HELWAN UNIVERSITY

Factually of computer science and artificial intelligence

Information system department

[Jobs]

A graduation project dissertation by:

Ezzeldin Ahmed Ezzeldin Sadeq (201900475)

Osama Abdelsalam fahmy Mohamed (201900122)

Nesreen Khaled fayez ataa elgenedy (201900901)

Sohila wael Ahmed Hamdy (201900358)

Mohamed Ashraf Helmy Shaaban (20160336)

Hassan Mekawy kayaty abd elbaky (20180199)

Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science in Computers & Artificial Intelligence, at the Information System Department, the Faculty of Computers & Artificial Intelligence, Helwan University Supervised by:

[Dr. Helal Ahmed]

JOBS

Supervised By: Dr. Helal Ahmed

Team Members

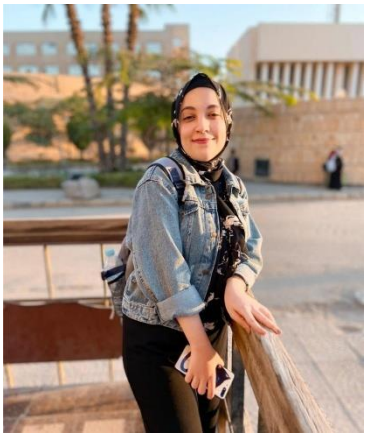
Ezzeldin Ahmed

Osama Abdelsalam

Hassan Mekawy

Mohamed Ashraf

Sohila wael



Nesreen Khaled

Acknowledgement

I would like to express my deepest gratitude to all those who have Contributed to the successful completion of this graduation project .First and foremost, I would like to thank my supervisor **[Dr. Helal Ahmed]** for her Invaluable guidance, support, and Encouragement throughout the entire research process. Their Expertise, constructive feedback, and mentorship have been Instrumental in shaping this project. I would also like to extend my Thanks to The faculty members, for their Insightful comments and Suggestions during the review process. Their contributions have Helped me to refine and Improve my work . Furthermore, I would like to thank the staff of My Department, for Providing me with the necessary resources, data, and equipment to Conduct my research. Their cooperation and assistance have been Essential in ensuring the success of this project .Finally, I would like to express my gratItude to my family and Friends, who have provided me with unwavering support ,Encouragement, and motivation throughout my academic journey .Their love and understanding have been the driving force behind my Achievements.Once again, I am deeply grateful to everyone who has contributed To this graduation project, and I hope that my work will be of value to The academic community and beyond

Abstract Freelancing websites have become increasingly important in the contemporary work landscape for several reasons:

1. **Flexibility:**

- Freelancing platforms offer flexibility in terms of working hours and location. Freelancers can work from anywhere with an internet connection, allowing for a better work-life balance.

2. **Global Talent Pool:**

- Businesses can tap into a global pool of talent and find individuals with specific skills that may not be readily available locally.

3. **Diverse Skill Sets:**

- Freelancing websites cover a wide array of skills and services, including writing, design, programming, marketing, and more. This diversity allows businesses to find specialists for various tasks.

4. **Cost-Effective Solutions:**

- For businesses, freelancing platforms can be a cost-effective solution. They can hire freelancers on a project basis, avoiding the need for full-time hires and associated overhead costs.

5. **Scalability:**

- Freelancing allows businesses to scale their workforce up or down based on project demands. This flexibility is particularly valuable for startups and small businesses.

6. **Access to Specialized Talent:**

- Businesses can access highly specialized talent for specific projects without the need for lengthy recruitment processes. Freelancers often bring niche skills and experiences to the table.

7. **Entrepreneurial Opportunities:**

- Freelancers have the opportunity to build their own businesses, market their skills, and work with a variety of clients. This entrepreneurial aspect is appealing to many professionals.

8. **Digital Transformation:**

- The rise of freelancing aligns with the ongoing digital transformation. As work becomes more digital and remote, freelancing platforms provide a space for this type of work to thrive.

9. **Innovation and Creativity:**

- Freelancers often bring fresh perspectives and innovative ideas to projects. They can introduce new approaches and solutions, contributing to a company's overall creativity.

10. **Reduced Geographic Barriers:**

- Freelancing platforms break down geographic barriers, enabling collaboration between individuals from different parts of the world. This diversity can lead to richer and more varied outcomes.

11. **Skill Development:**

- Freelancers continuously enhance their skills to stay competitive in the market. This culture of ongoing skill development benefits both freelancers and the businesses that hire them.

12. **Marketplace Efficiency:**

- Freelancing websites create efficient marketplaces where supply (freelancers) and demand (businesses) can quickly and easily connect, reducing the time and effort required to find suitable talent.

In summary, freelancing websites have become integral to the modern workforce, offering opportunities for businesses to access specialized talent, for individuals to build independent careers, and for the overall economy to adapt to changing work dynamics.

Table of Contents

Chapter ONE : INTRODUCTION

1.1.Overview

1.2.Objectives

1.3.Purpose

1.4.Scope

1.5.General constraints

1.1 Overview

Our freelancing website is an online platform that connects freelancers (individuals offering services) with clients or businesses seeking those services. These websites act as intermediaries, facilitating a marketplace for work, allowing freelancers to find projects and clients to find skilled professionals for specific tasks or projects. Here's an overview of key aspects:

1.2 Objectives

The objectives of our freelancing website are designed to create a dynamic and supportive environment for both freelancers and clients. Here are the key objectives:

1. **Connect Talent with Opportunities:**

Facilitate a platform where freelancers can showcase their skills and connect with clients seeking their expertise.

2. **Diverse Job Opportunities:**

Offer a broad range of job categories to ensure freelancers from various fields find opportunities that match their skills and interests.

3. **User-Friendly Interface**

Provide an intuitive and user-friendly interface that enhances the overall experience for both freelancers and clients.

4. **Security and Trust:**

Prioritize the security of user data and transactions to build trust among freelancers and clients, creating a safe environment .

1.3 Purpose

The purpose of a freelancing website is multifaceted, aiming to serve the needs of both freelancers and clients in the digital marketplace. Here are some key purposes:

1. **Connect Freelancers and Clients:**

   Facilitate the connection between skilled freelancers and clients seeking specific services, creating a platform for mutually beneficial collaborations.

2. **Provide Job Opportunities:**

   Offer a centralized space where freelancers can find diverse job opportunities across various industries and skill sets.

3. **Promote Professional Independence:**

   Empower freelancers to work independently, allowing them to choose projects that align with their skills, interests, and availability.

4. **Enable Remote Work:**

Cater to the growing trend of remote work by providing a platform where clients can find talented freelancers regardless of geographical location.

5. **Offer Cost-Effective Solutions:**

Provide cost-effective solutions for businesses by allowing them to hire freelancers for specific projects without the overhead costs associated with full-time employees.

6. **Create a Global Talent Pool:**

Build a diverse and global talent pool, giving clients access to a wide range of skills and expertise beyond their local vicinity.

7. **Facilitate Efficient Communication:**

Enhance communication between freelancers and clients through built-in messaging systems, ensuring clear project requirements and expectations.

8. **Ensure Fair Compensation:**

Establish a transparent and fair compensation system to ensure that freelancers are adequately rewarded for their skills and efforts.

9. **Encourage Entrepreneurship:**

Foster entrepreneurship by allowing freelancers to build their own businesses, gain experience, and develop a portfolio of work.

10. **Support Gig Economy:**

Contribute to the growth of the gig economy by providing a platform for short-term, project-based work that aligns with the changing nature of work.

11. **Promote Skill Development:**

Encourage continuous learning and skill development among freelancers by exposing them to a variety of projects and industries.

12. **Facilitate Reviews and Ratings:**

Enable clients and freelancers to leave reviews and ratings, establishing a reputation system that helps build trust within the community. .4. Scope

1.4 Scope

The scope of a freelancing website is to provide a digital platform that connects freelancers and clients, facilitating a diverse range of job opportunities across various industries and skill sets. It enables global collaboration, offers flexibility in work arrangements, supports remote work, fosters entrepreneurial opportunities, promotes skill development, ensures fair compensation, and includes features like reviews and ratings for reputation management. The scope extends to creating a dynamic and inclusive online marketplace that adapts to industry changes and fosters innovation in the gig economy.

1.5 General Constraints

Scope: project outcome as defined in the contract.

Risks: uncertainties associated with the project •

Time: deadline for delivering the output

Time management. • Indiscipline “Human factor” like being late in delivering tasks or attending meetings .

Chapter TWO : Planning & Analysis

1.Project planning

2. Analysis an existing system

3. Need for the new system

4. Analysis of the new system

5. Advantages of the new system

6. Risk and Risk Management

.Project Planning

2.1.1 Feasibility Study

To determine the realization of this project, we need to research these Aspects further:

Problems solved by the project:

1. Convenience: One of the primary benefits of a website for jobs is the Convenience it provides. Clinte can quickly and easily schedule appointments With their preferred freelancer without having to wait on hold or navigate Complicated phone systems.

2. Efficiency: A website for jobs can also improve the efficiency of the Appointment scheduling process. It can eliminate the need for manual Appointment booking, reducing errors and freeing up staff time to focus on Other tasks.

3. Access to information: A website for jobs can provide clinte with access to Important information about freelancers, such as their qualifications, Experience, and specialties. This can help clinte make informed decisions .

Who will benefit from the project ?

1. Clinte: Clinte can benefit from a website for jobs by being able to quickly And easily schedule appointments with their preferred freelancers, view their Projects records, and communicate with their freelancers. This can improve Clinte engagement and overall projects.

2. Freelancers: Freelancers can benefit from a website for jobs by reducing Administrative tasks, improving clinte engagement, and increasing efficiency. This can help freelancers focus on providing high-quality care to their clinte.

3. Organizations:Organizations can benefit from a website for jobs by Improving clinte satisfaction, reducing costs associated with manual Appointment scheduling, and increasing efficiency.

Reasons for choosing this project to work on:

1.Growing demand for freelancers:

The COVID-19 pandemic has accelerated the adoption of jobs , As more people seek

Services remotely, a website for jobs can help meet this growing demand.

2.Convenience: Many people find it difficult and time-consuming to schedule Appointments with their clinte, particularly when they have busy schedules. A

Website for jobs can make the process much more convenient and user-Friendly.

3.Improved efficiency: A website for jobs can help improve the efficiency of The appointment scheduling process, reducing errors and freeing up staff time To focus on other tasks.

4.Competitive advantage: In today's freelancing market, clinte are Increasingly looking for providers who offer modern, convenient, and user-Friendly

services. A website for jobs can help freelancers differentiate Themselves from their competitors and attract new clinte.

5.Improved clinte engagement: A website for jobs can help improve clinte Engagement by providing clinte with access to important information about Freelancer, allowing them to communicate with freelancers, and providing Them with a more streamlined and userfriendly experience.

Importance of the project:

The importance our website is that it can significantly improve the clinte Experience by allowing them to schedule appointments more easily and Quickly. The website can also improve the efficiency of the appointment Scheduling process, reducing errors, and freeing up staff time to focus on Other important tasks. Additionally, the website can improve clinte Engagement by allowing them to view and manage their appointments, Communicate with freelancers, and access their projects records. All of this Can help improve experience and clinte satisfaction. For freelancer, the Website can help reduce costs associated with administrative tasks, improve Clinte engagement, and increase efficiency.

Market analysis：

Main competitors:

• Freelancer.

•Clinte.

Target market:

• Age: 25 –90

• Gender: Both.

• Place: We will start in Egypt, then expand to the world on a large scale.

• Income level: 2000-3000

• Level of education: primary education and above.

• Related behavioral factor

Sales analysis:

Sales strategy: to increase sales, we must pay attention to important points

Like:

• Update data quickly.

• Checking and correcting all data.

• Checking user’s complaints and following them up regularly.

• Get regular tests.

• Upgrade the website offer enhanced Services.

Technical Analysis:

• No hardware required for the site.

• Key technologies and programs we associate with:

• App services, c#. .nt

• Microsoft SQL server

Resource and time analysis:

Devices for coding, operating and testing - laptopsand laptops.

• Codecs.

• Internet service.

• Design software.

Timetable:

• 1 month for gathering information and Diagrams.

• 2 month for planning and analysis.

• 1 months of training and studying.

• 5 months for development.

- 1 month for testing.

• **Optional requirements:**

1-the user can post comments on reports

2-The user can send his feedback or questions

3-The user can view and edit his profile.

4- The user can make a chat.

Internet connection:

Since we would need to keep the user's data in the cloud, we need an

internet connection to store and retrieve the data.

Storage space: To keep the user's data in addition to the application itself, we need a

fine storage.

2.1.1Estimated cost

2.1.1.1 Meeting at a public workplace 2 times.

2.2 Analysis of existing system

•Dashboard

•Wanted

•Graphs

• Reports

•Notification

•Connecting with people

• Locating the closest freelancer

•Alerting the abnormal inputs

2.3 Need for the new system Our website include:

- Recruitment.
- Screening and selection.
- Credentialing and privileging.
- Ongoing performance management.
- Search by Category.

2.4.4 Functional Requirements

1. Sign in Actor: (freelanc or clinte) . Pre: Opened our site then pressed “Sign In”. Description: The user gets his name, age, address, password, Email , then it’s saved in our database.Post: The User has an account, and he can use the rest of the functions provided by the site.

2. Sign in Actor: User (freelancer or clinte). Pre: The user opened the site, pressed “Log In:” and the user is registered before. Description: user enters his username and password so he can use the site. Post: The User can go to the home page.

3.Delete clinte : User. Pre: User is logging in, open the list of admins and press on “Delete User” button. Post: User Deleted successfully. Description: Admin choose other one and press on ‘delete’.

4.Search Actor: clinte . Pre: Clinte want to find a freelancer to book with. Post: User shows a list of freelancers related to his name that search by it. Description: Clinte enters search name or category or both in search box and ask for result.

5.Make Reservation Actor: Clinte . Pre: Clinte want to make a Reservation. Post: Clinte book an appointment successfully. Description: Patient enters search name or category or both in search box and choose the doctor and click on make appointment.

6. cancel Reservation Actor: Patient. Pre: Patient wants to cancel a Reservation. Post: patient cancel an appointment successfully. Description: Patient open my upcoming Appointment and click on cancel reservation.

7.Add Post Actor: User (Freelancer or Clinte . Pre: User login on the Website. Post: Post added successfully. Description: User can make post as he wants.

8.Add comment Actor: User (freelancer or clinte). Pre: User login on the website. Post: Comment added successfully. Description: User can add replay on any post.

9.add Rate Actor: Clinte Pre: clinte search for the desired freelancer. Post :

Rate added successfully. Description: Clinte can rate any doctor.

10.Edit profile Actor: user (freelancer or clinte). Pre: user open my profile. Post: profile edited successfully. Description: user can edit his profile.

.11view profile Actor: user (freelancer or clinte). Pre: user open my profile .Post: profile viewed successfully. Description: user can view his profile.

12. Message: Actor: user (freelancer or clinte). Pre: user search for another User. Post: user message successfully. Description: user can message any Other user.

13. Contact us Actor: User (freelancer or clinte) Pre: User is logged in the site And pressed on “contact us”. Post: An inquiry is recorded successfully. Description: User can add any inquiry and press submit.

14. Logout Actor: User (freelancer or clinte) Pre: User is logged in the site. Description: User presses on “Logout”. Post: The user’s session is ended, and A login\ register page is shown. Description: User presses on “Logout”.

2.4.1 User Requirements

Mandatory requirements:

• The user enters the name, category.

• The date he wants to receive the project At.

• His city and address in detail or His Place. Desirable requirements:

• Project and project details.

2.4.2**System Requirements Mobile & Desktop : windows,and Linux.**

• Internet connection:

It will work on mobile android and iOS, Since we would need to keep the User's data in the App so we need internet connection to store and Retrieve the data.

2.4.3Domain requirements

• Enough memory should be reserved on the database server to Accommodate for any number of users and their data.

• Each user must have a unique username/password combination.

• The website must verify all values before making the change in the Database.

• The database should be backed up every occasion in case the original Does become corrupt.

2.4.5 Non-Functional Requirements

1-Availability: The App must be contactable 24 hours.

2-Security: -Any unauthorized access to the App is denied, no one can use the Website without creating an authorized account. -All users are verified. – Passwords shall never be viewable at the point of entry or at any other time. –All data is reviewed before being displayed on the site by admin.

3-Usability -The App should be suitable for a certain age. -It should be easy to Use so that the user can understand it.

4-Flexibility: -The user can modify his data or declare anything he wants Through comments.

5-Scalability – The support manager shall support unlimited user's inquiries. – The number of users should not affect the App.

6-Efficiency -The user must get what he wants, whether he is viewing, sharing, Or Searching.

7-Maintainability. Maintainability defines the time required for a solution or its Component to be fixed, changed to increase performance or other qualities, Or adapted to a changing environment. Like reliability, it can be expressed as aprobability of repair during some time. For example, if you have 75 percent Maintainability for 24 hours, this means that there's a 75 percent chance the Component can be fixed in 24 hours.

8-Reliability -The official must verify the information before adding a clinte or Help find freelancer.

2.5 Advantages for the new system Our site involves:

• All free features.

• See all Doctors and divide them into categories for easy access

• makes it easy for the user to search by name and category to access more closely the doctor's features.

• easy to use by any one.

• Tracking all the inputs.

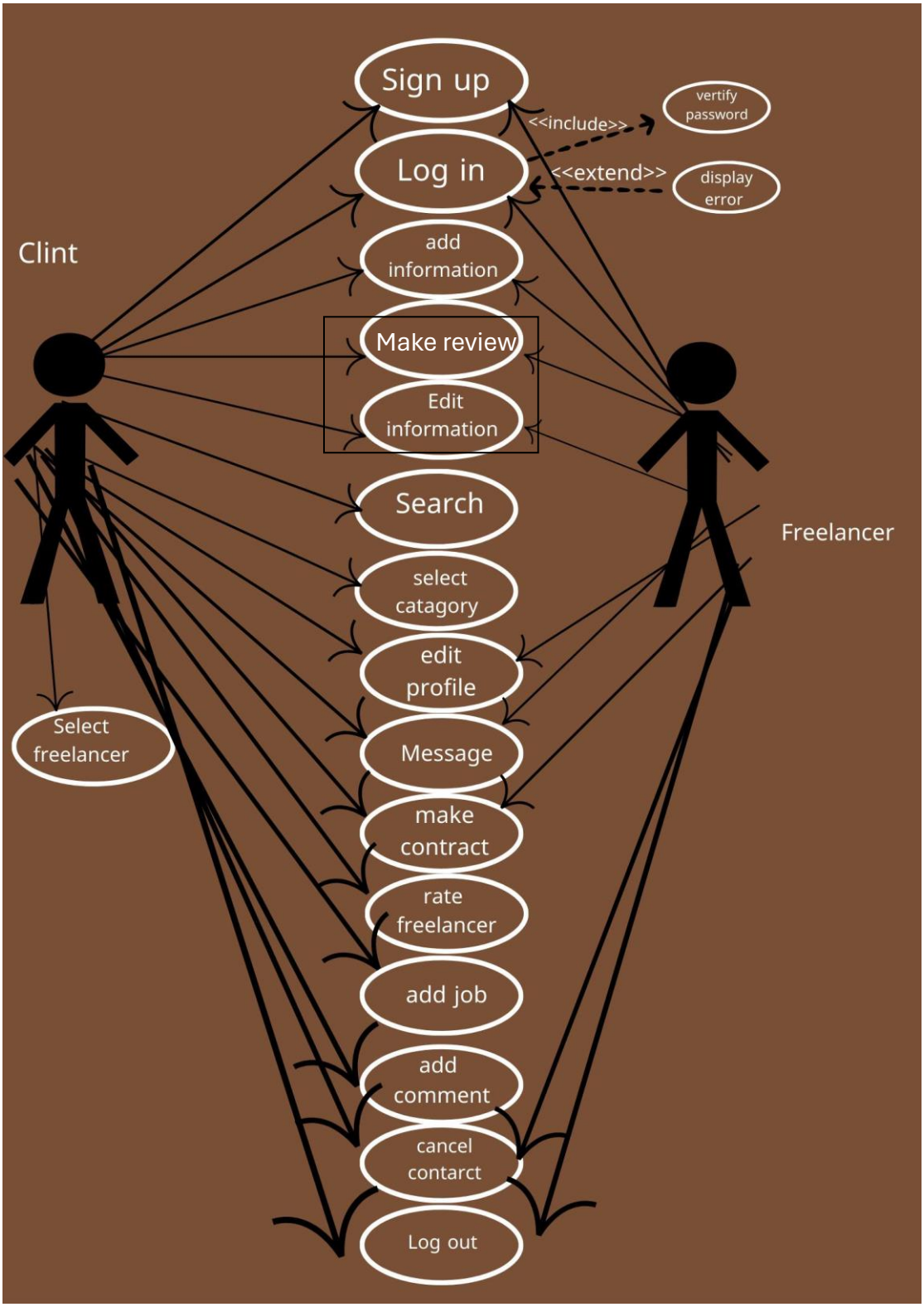
• User Friendly UI.

2.6 Risk and Risk Managements:

In the context of appointing a freelancer, risk refers to the potential for harm or negative outcomes that may result from the appointment of an unsuitable or incompetent freelancer. This can include harm to clintes, Damage to the reputation of the organization, and legal liability. Risk management involves identifying, assessing, and mitigating potential risks associated with the appointment of a freelancer. This may involve conducting thorough background checks and verifying the freelancer's credentials and qualifications to ensure that they are licensed to practice and have the necessary training and experience to perform the duties required of them. Other risk management strategies may include implementing policies andProcedures to ensure that freelancers adhere to established standards of Care, providing ongoing training and education to help freelancers stay up to Date with the

latest advances, and establishing protocols for how to handle Situations where a freelancer's performance or conduct is called into Question. Effective risk management can help

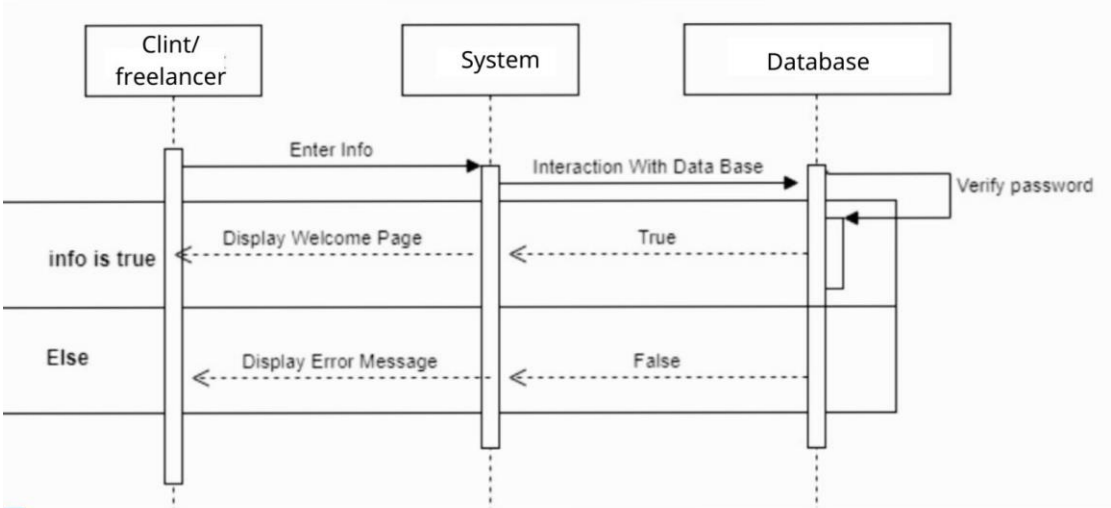
Chapter THREE : SoftWare Design



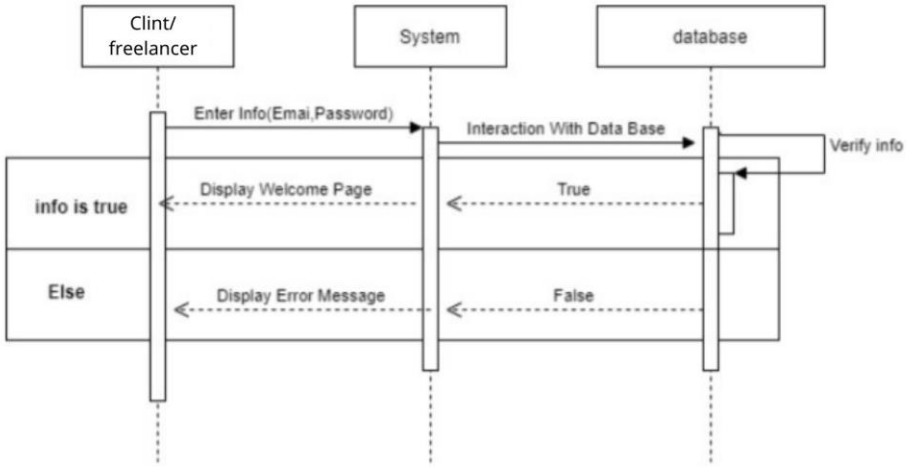
3.1 Use case digram

3.2 Sequence diagram

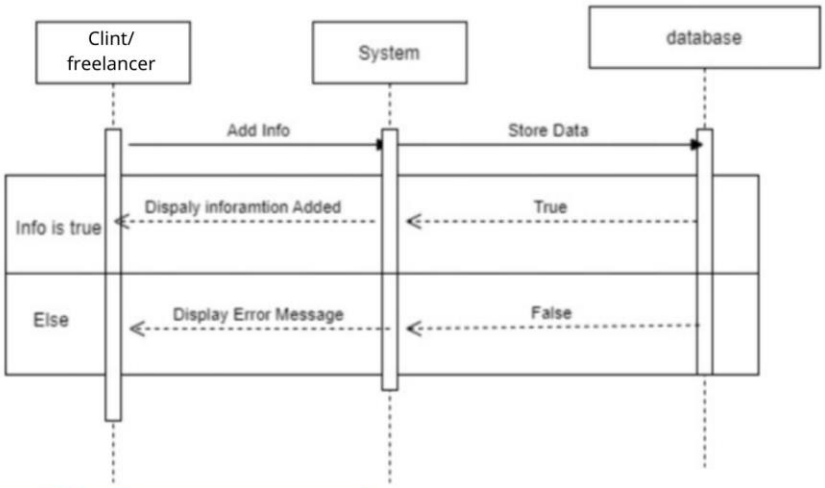
Sign up



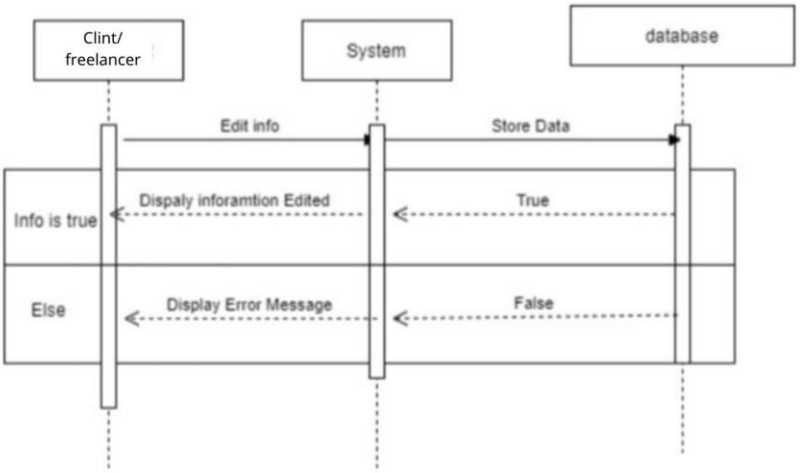
Log in



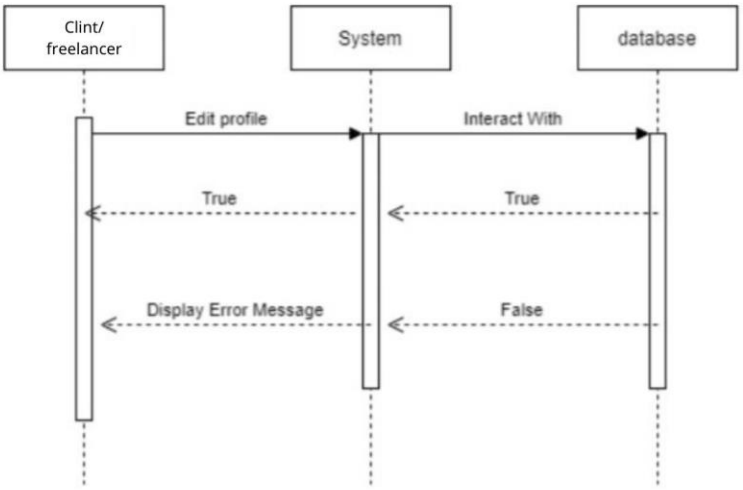
Add info



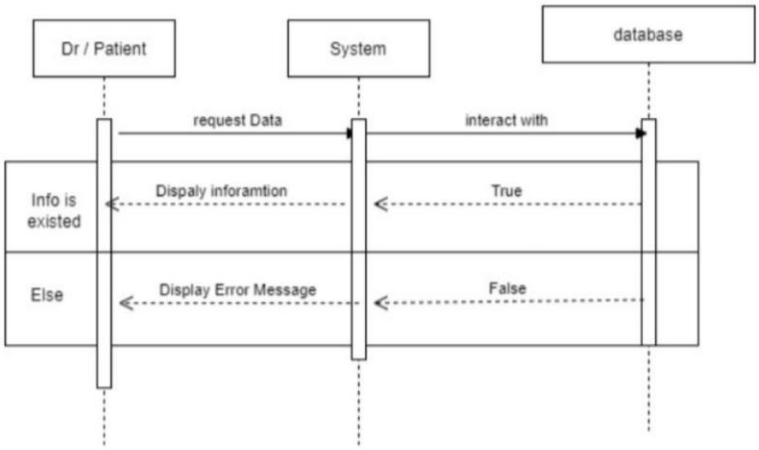
Edit Info



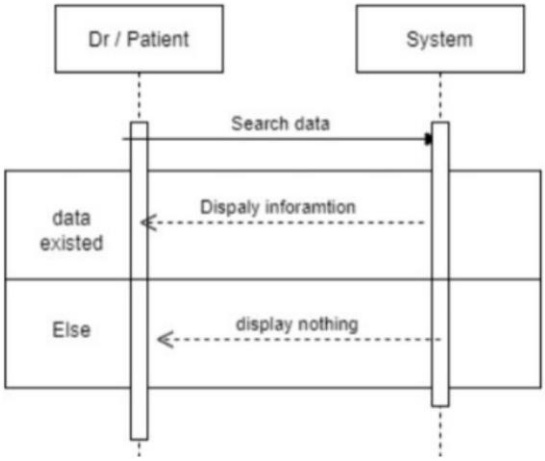
Edit profile



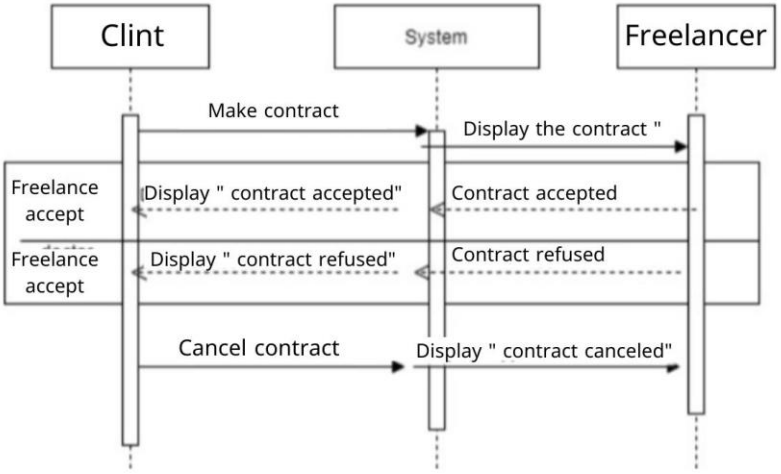
Make review



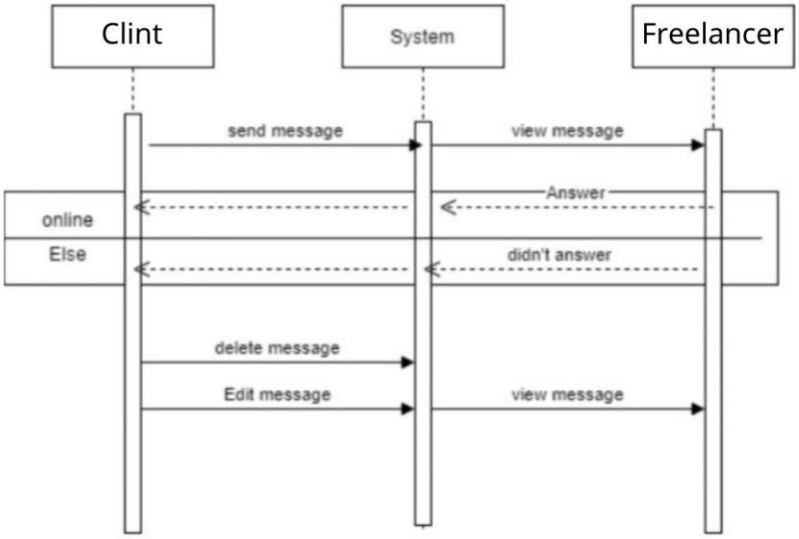
Search



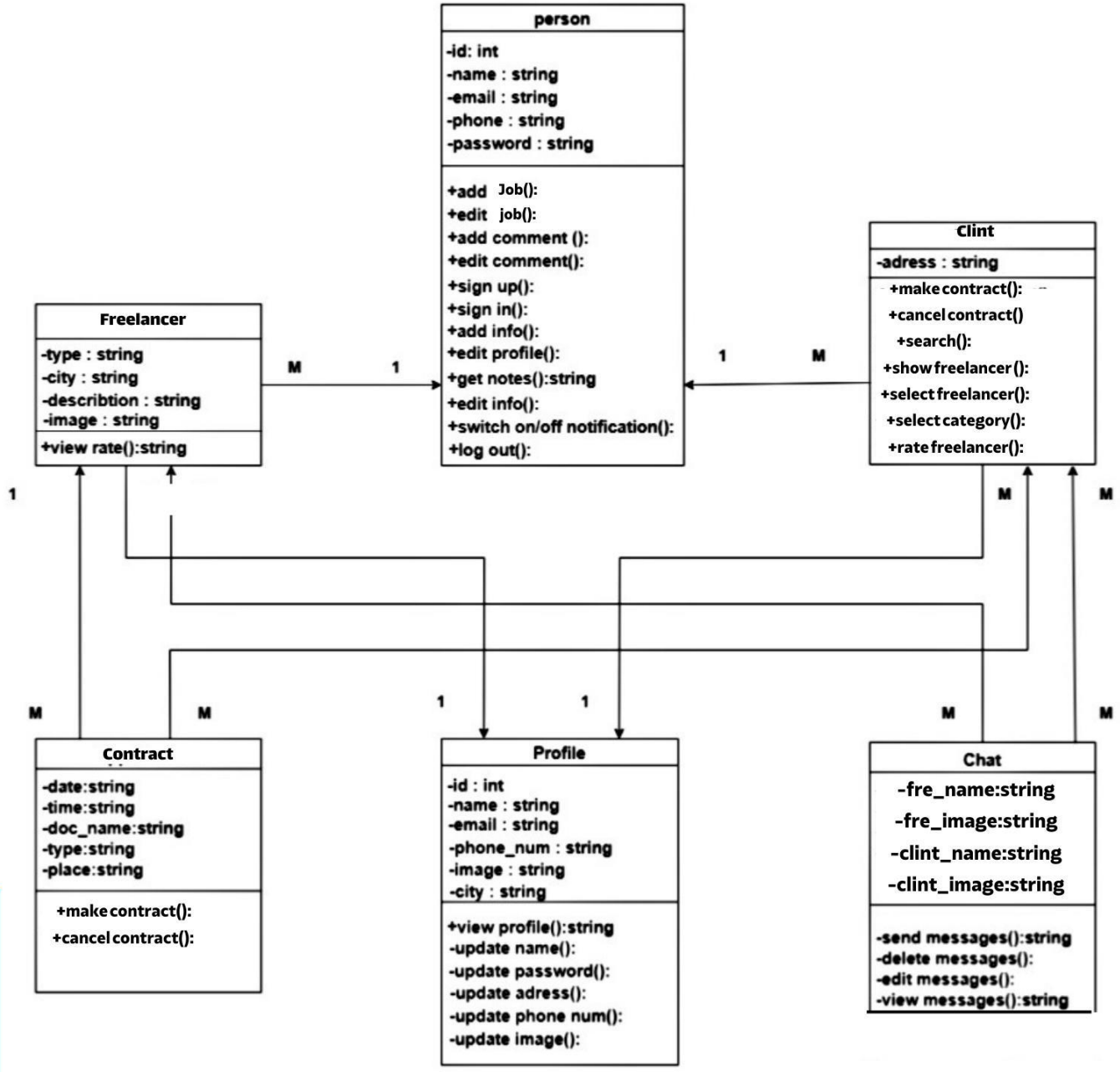
Make contract



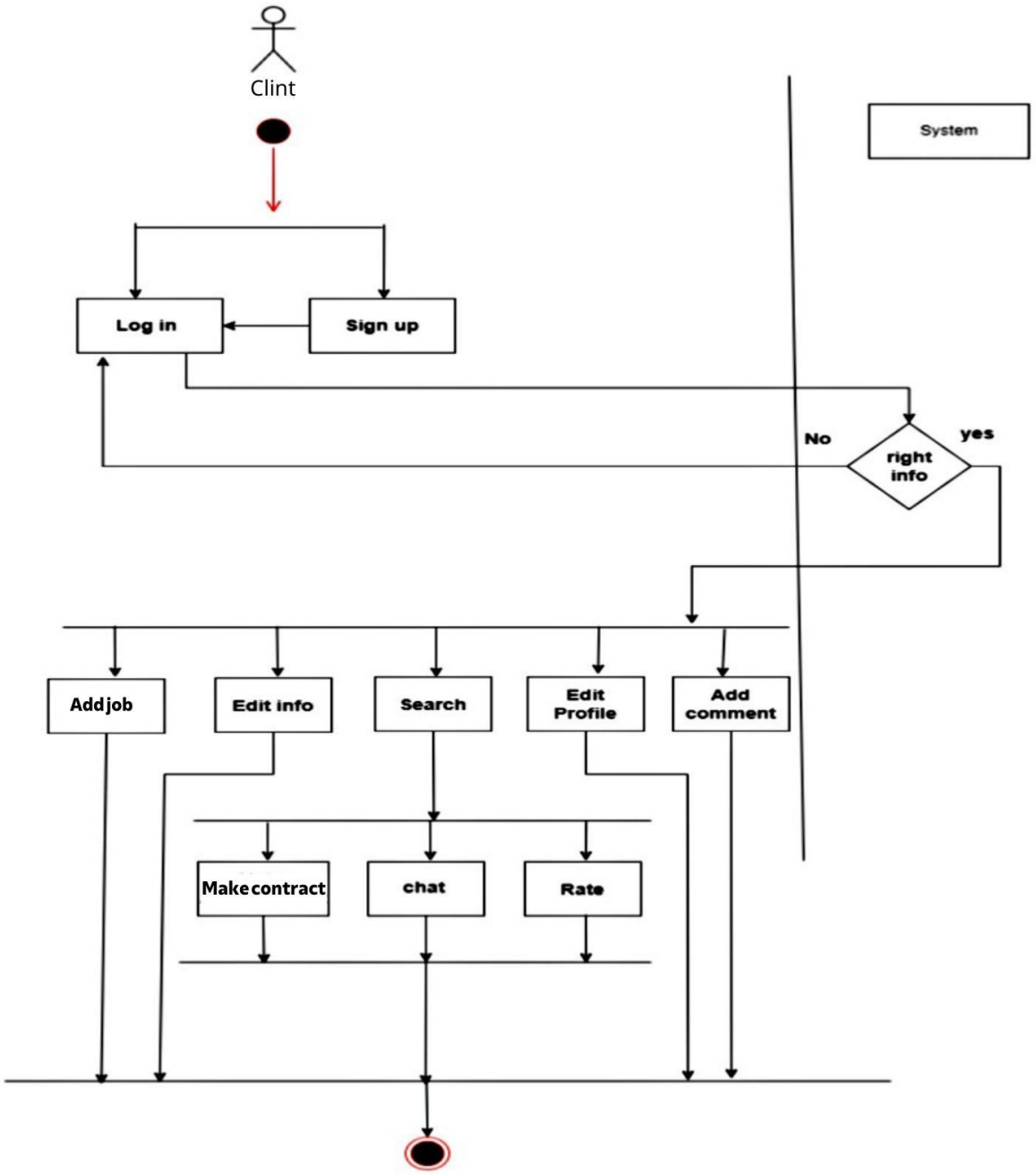
Message

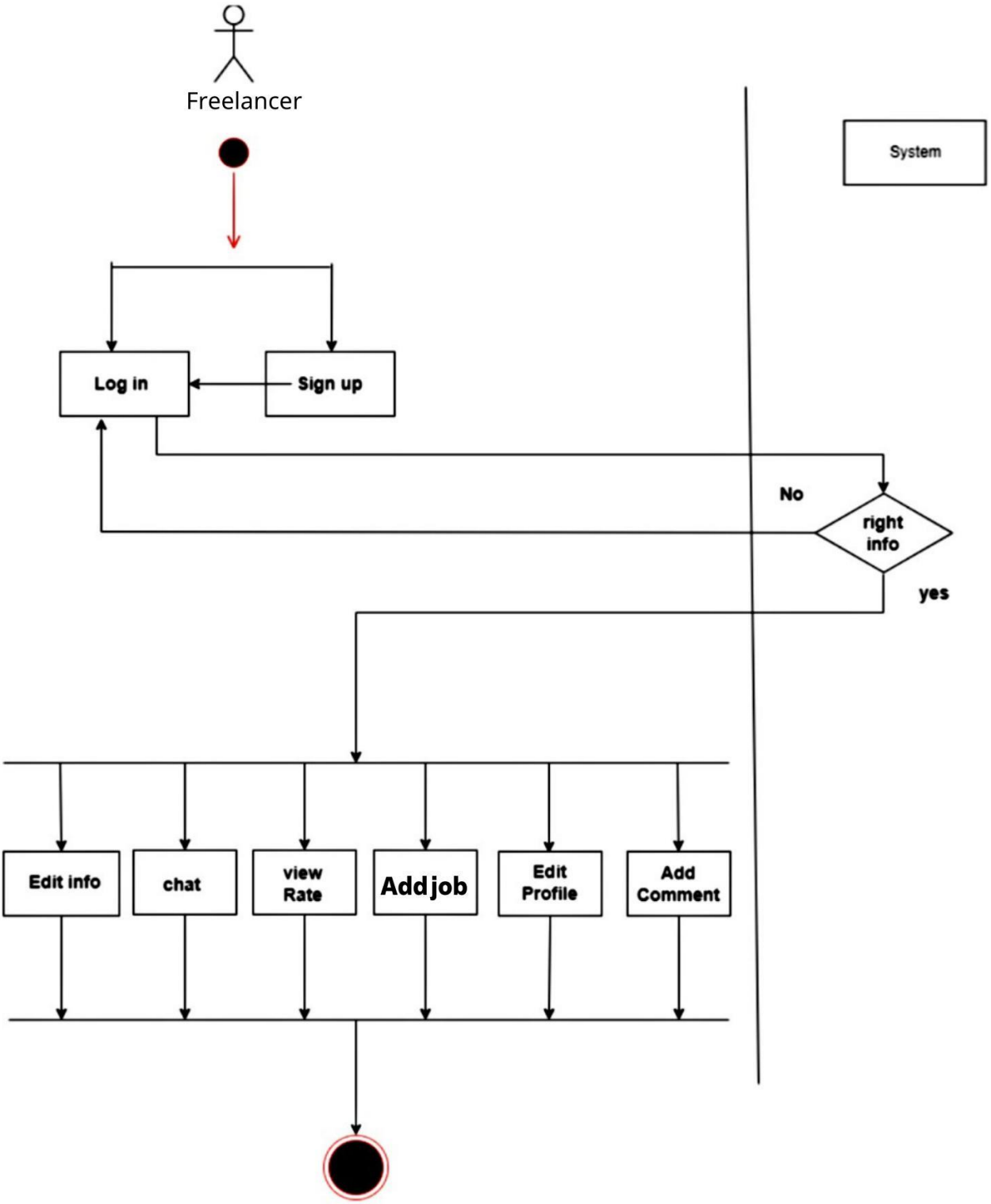


3.3 class Diagram



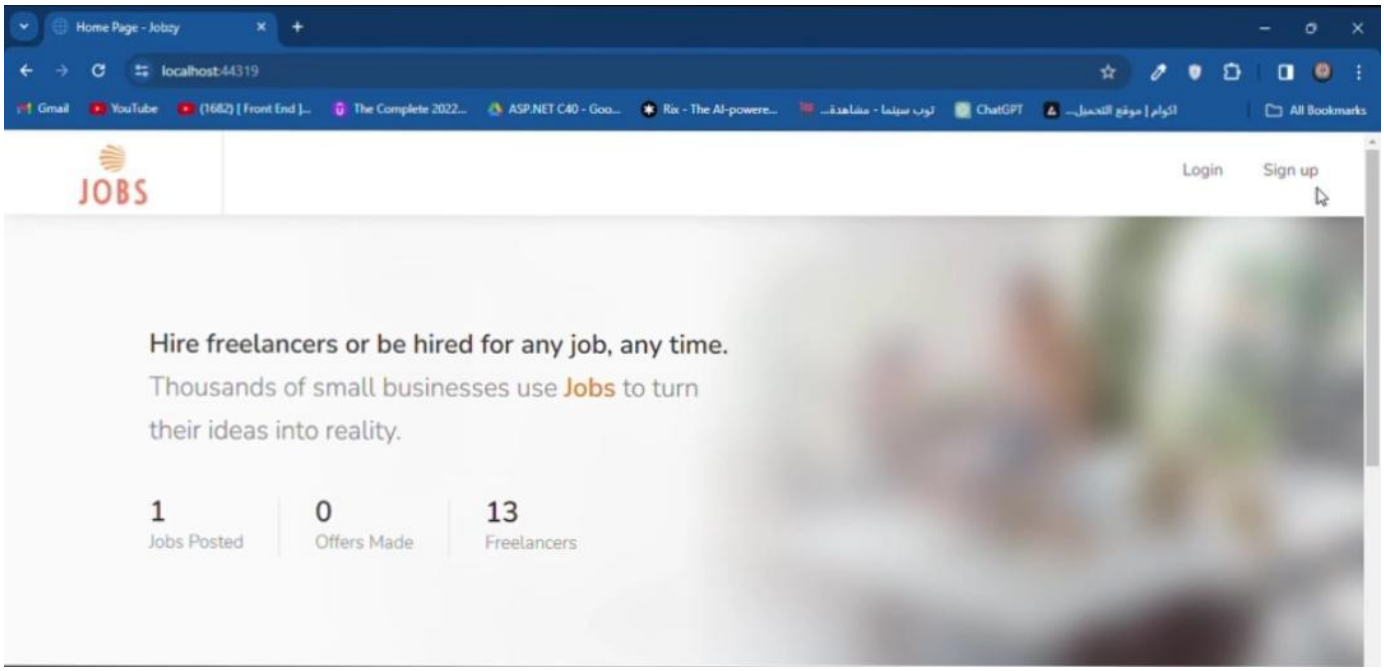
3.4 Activity Diagram



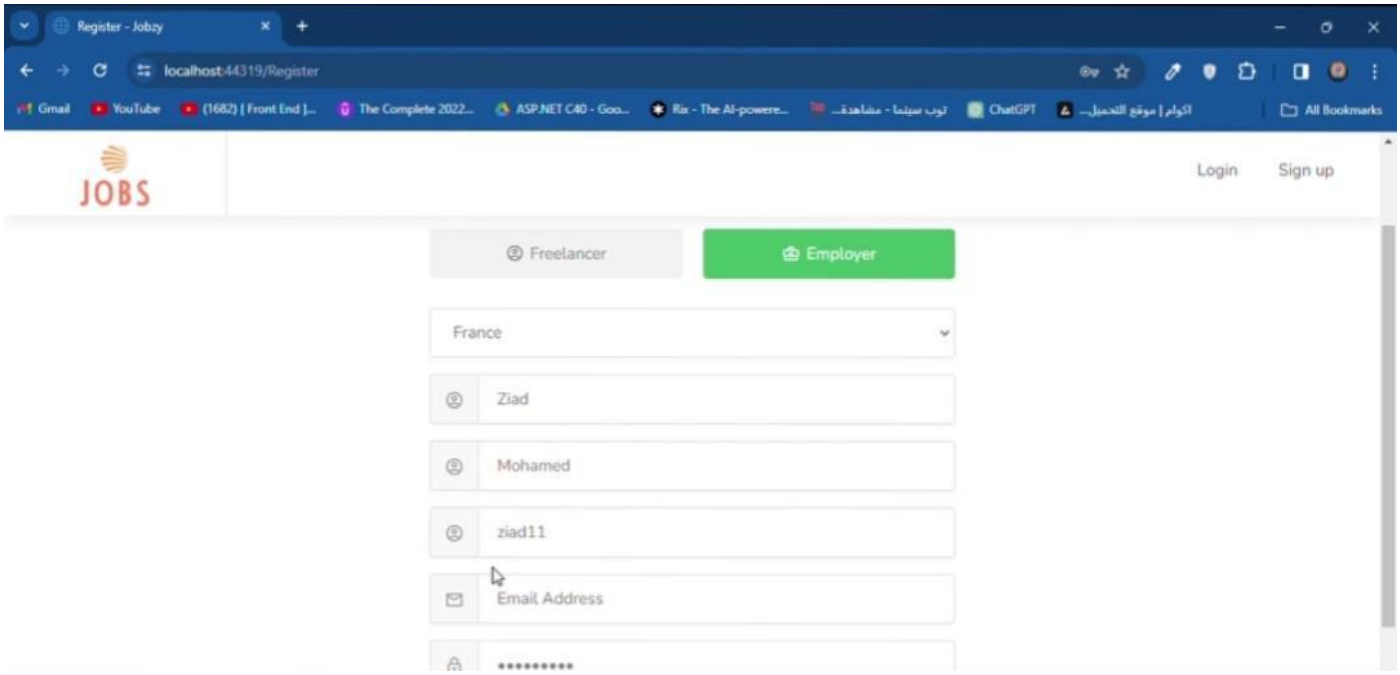

Freelancer
System
Log in
Sign up
No
right info
yes
Edit info
chat
view Rate
Addjob
Edit Profile
Add Comment

Chapter four :Work flow

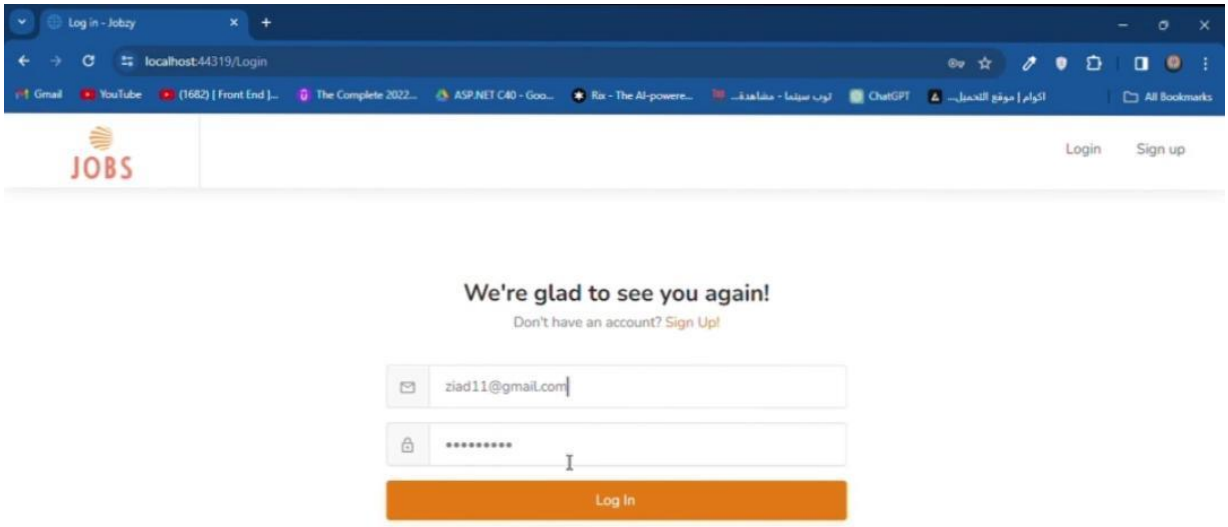
Welcome home



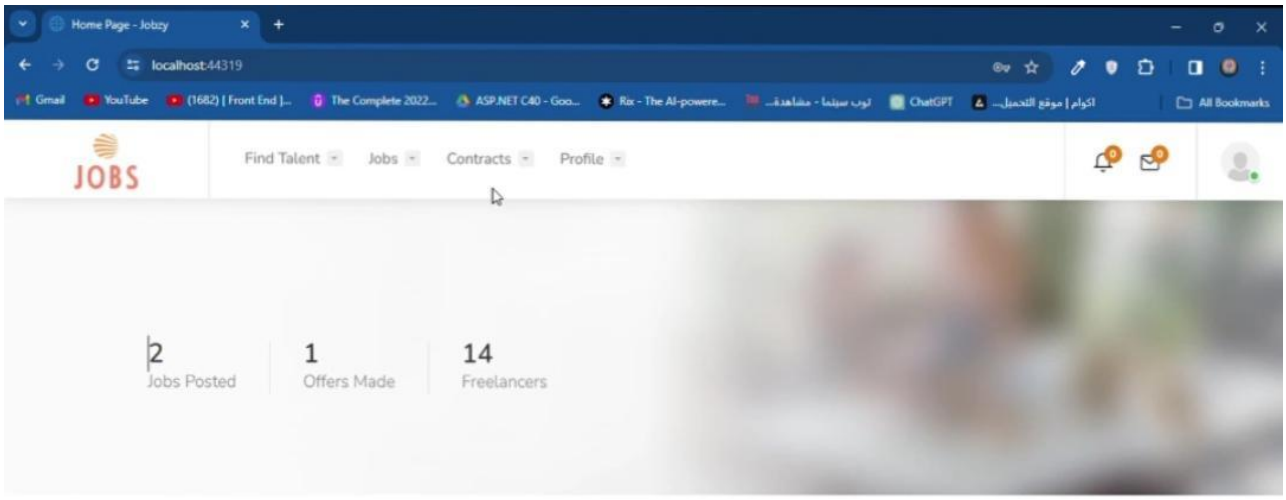
Sign up



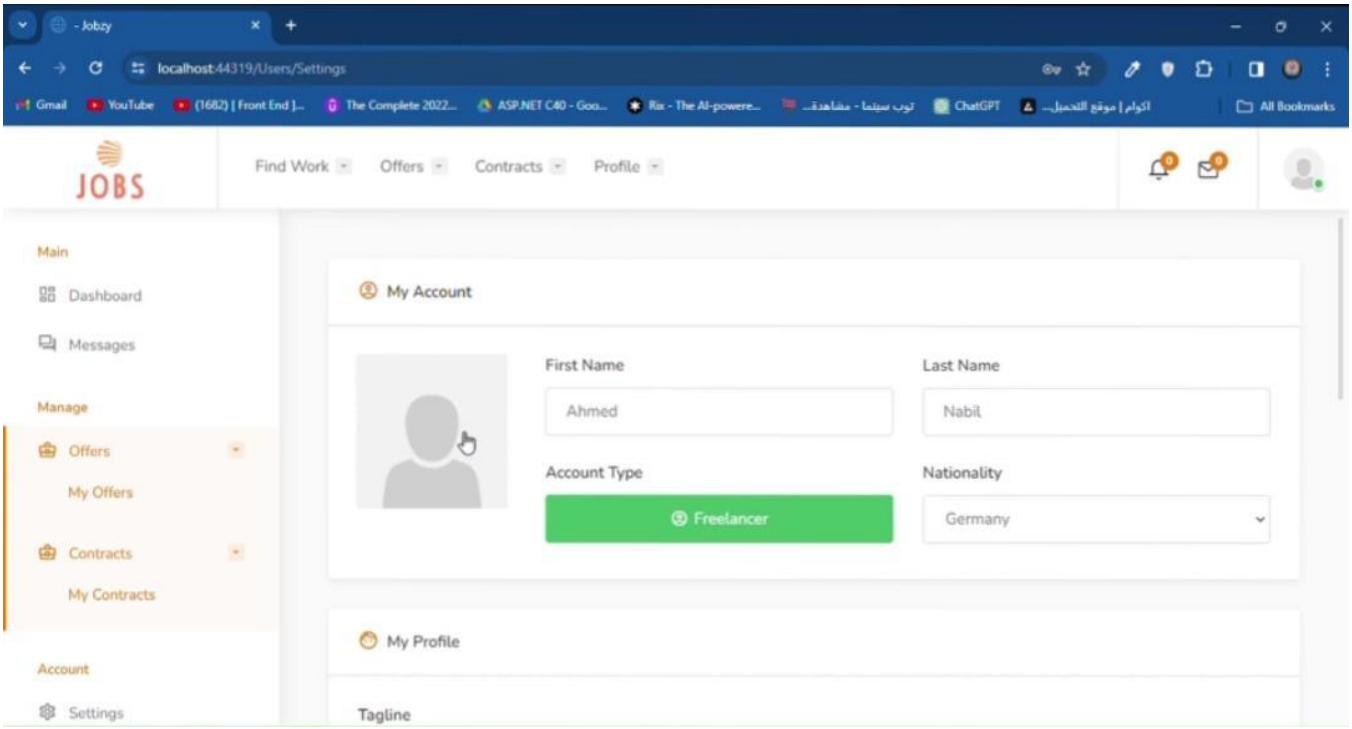
Log in



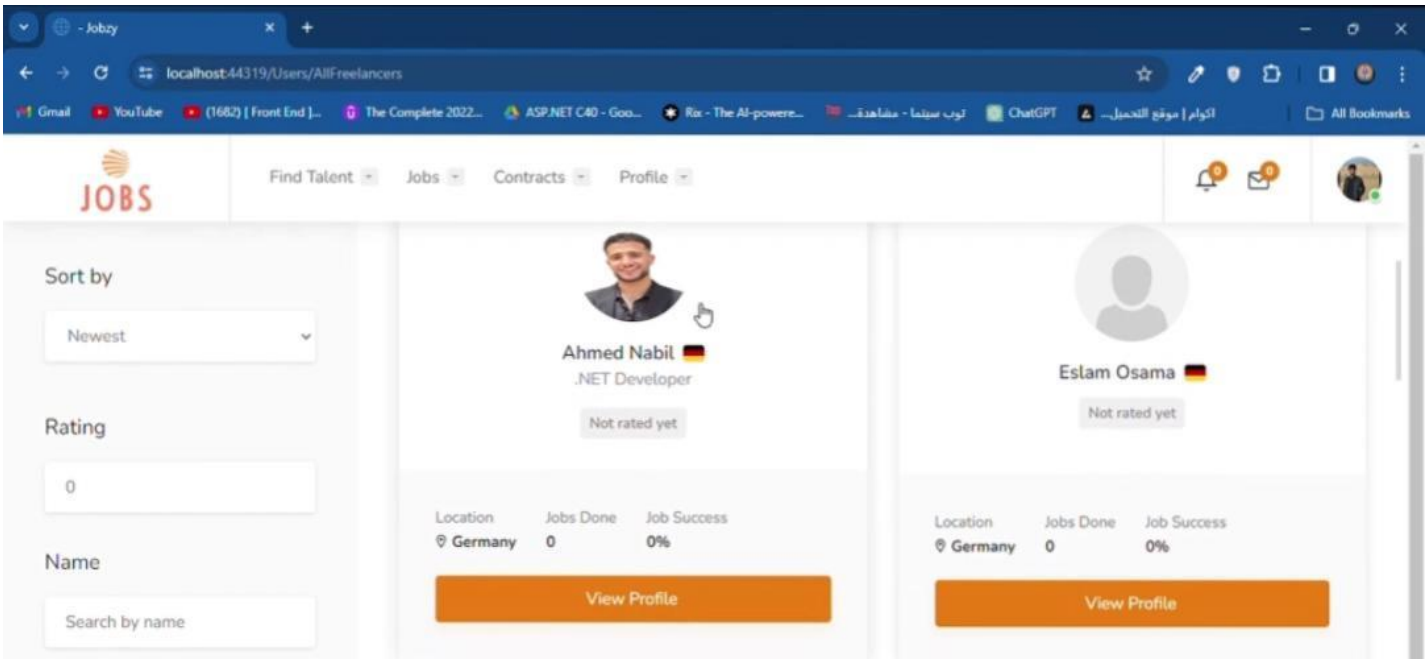
Home page



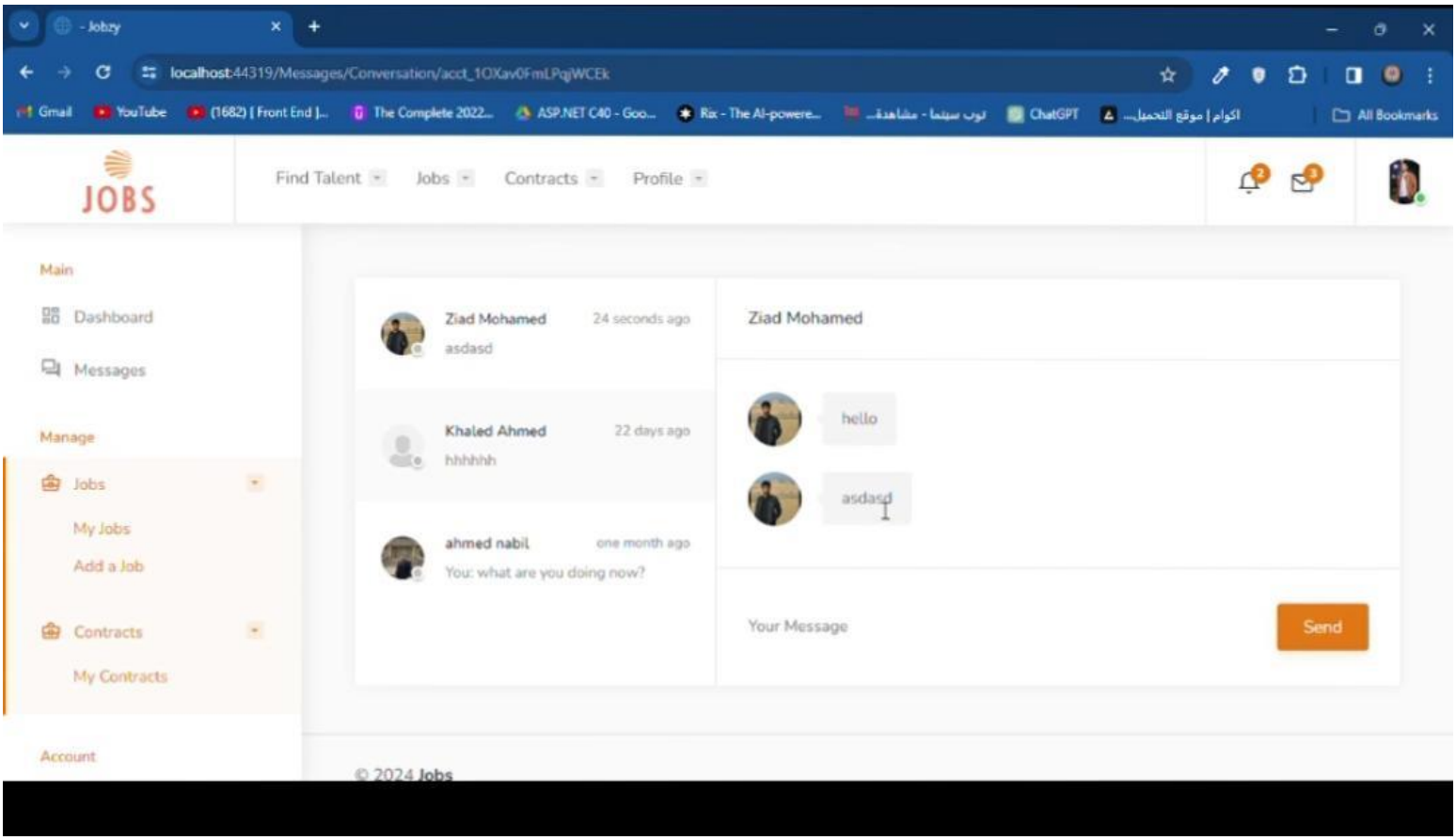
Add profile

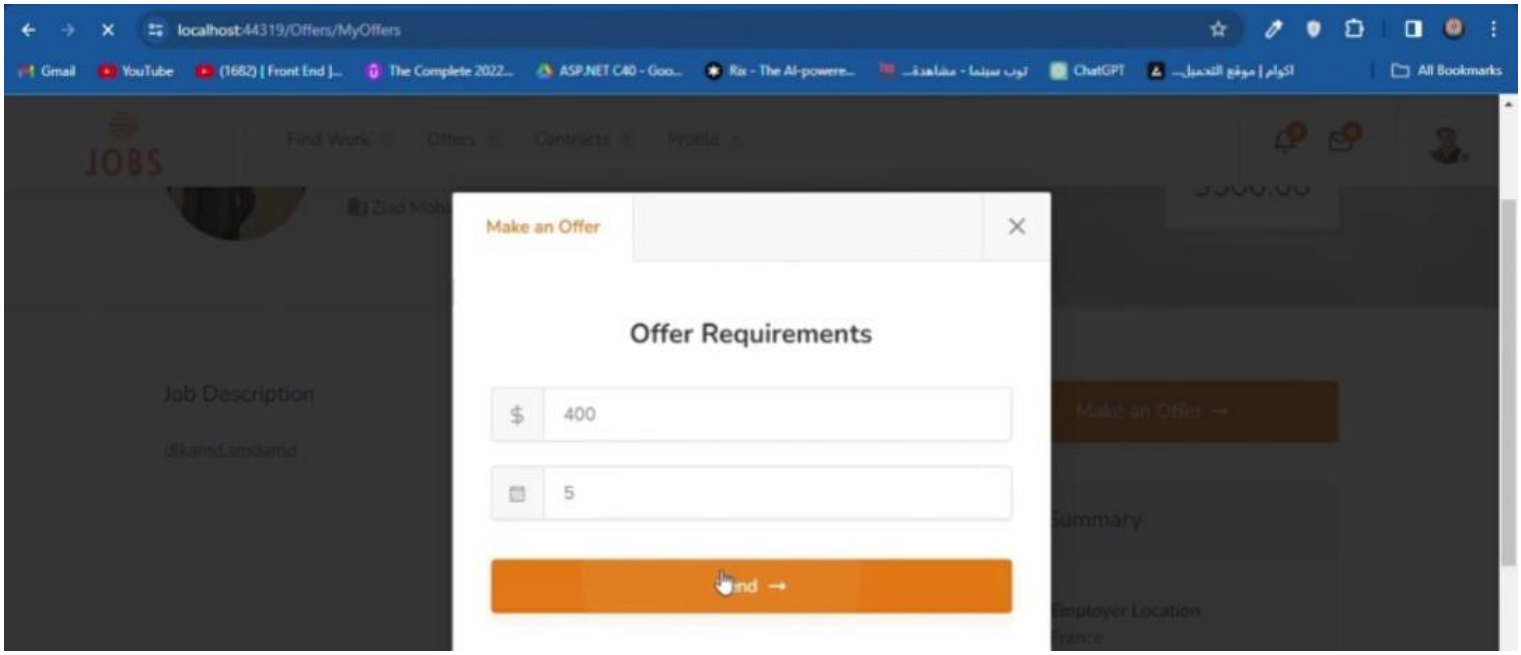


Freelancer profile


- Jobzy
localhost:44319/Users/AllFreelancers
Gmail
YouTube
(1682) | Front End |...
The Complete 2022...
ASP.NET C#0 - Goo...
Rix - The AI-powere...
ChatGPT
All Bookmarks
JOBS
Find Talent
Jobs
Contracts
Profile
Sort by
Newest
Rating
0
Name
Search by name
Ahmed Nabil
.NET Developer
Not rated yet
Location
Germany
Jobs Done
0
Job Success
0%
View Profile
Eslam Osama
Not rated yet
Location
Germany
Jobs Done
0
Job Success
0%
View Profile

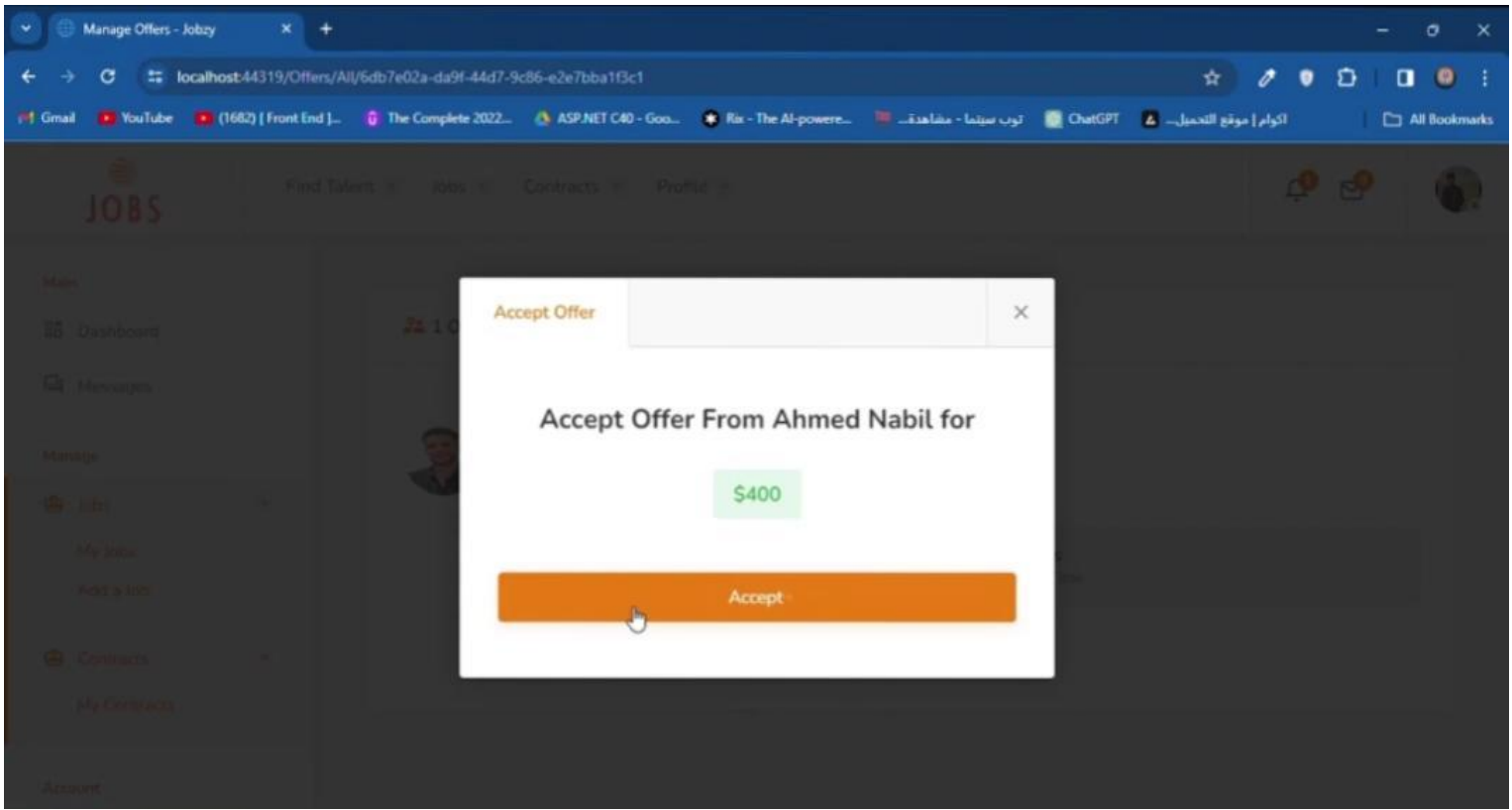
Message&chat



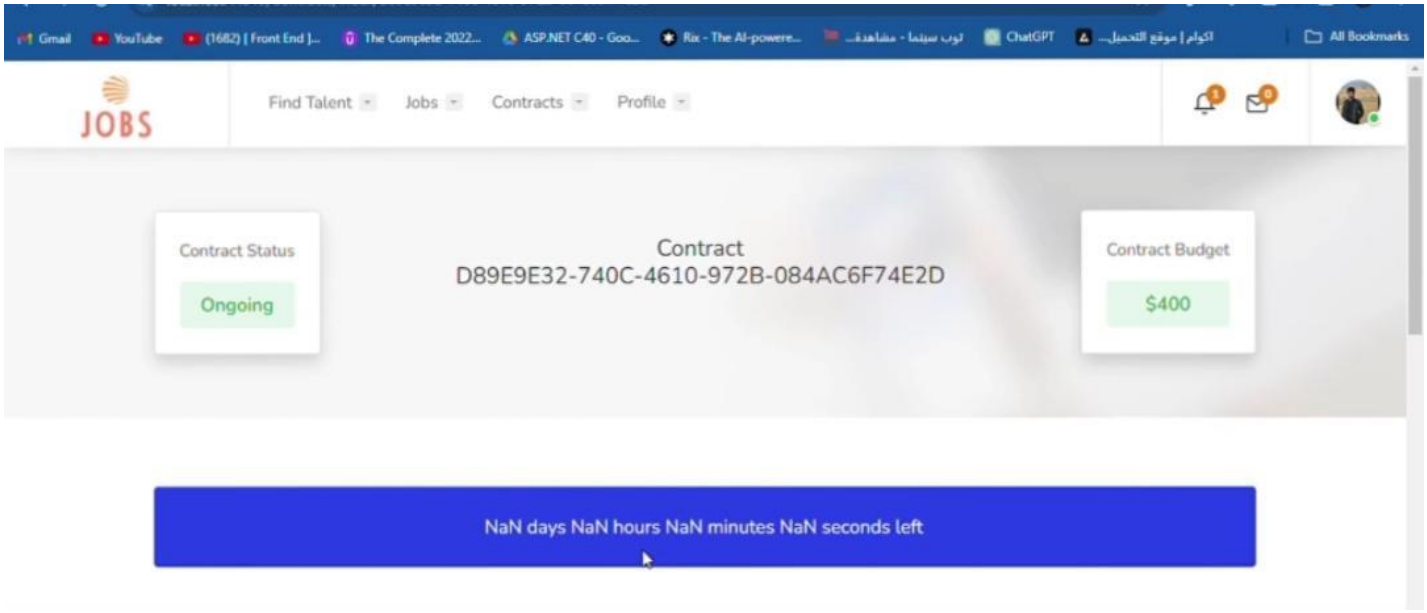


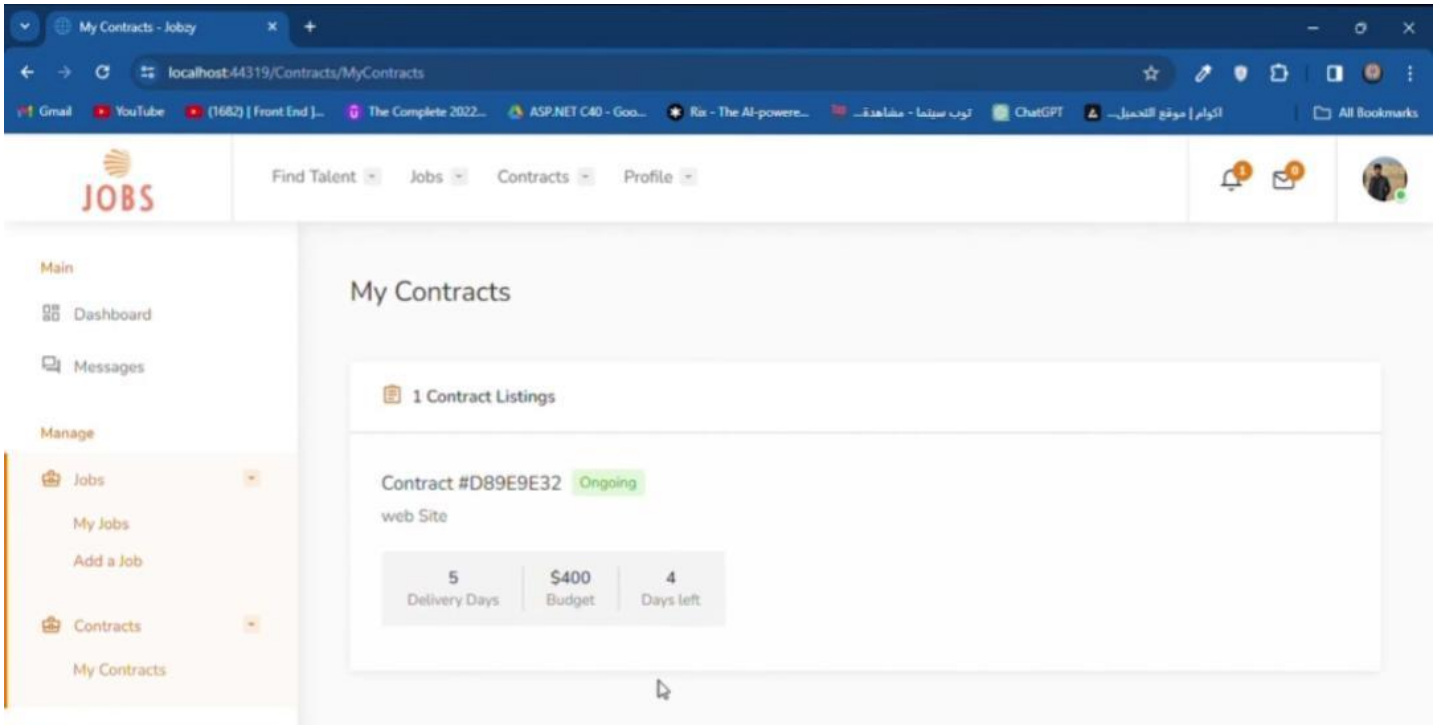
Make an offer

Accept offer

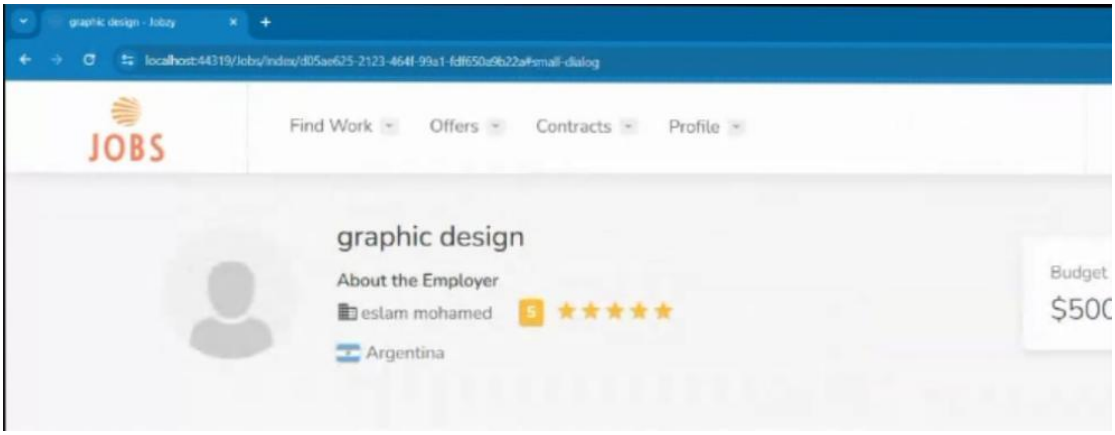


Contract

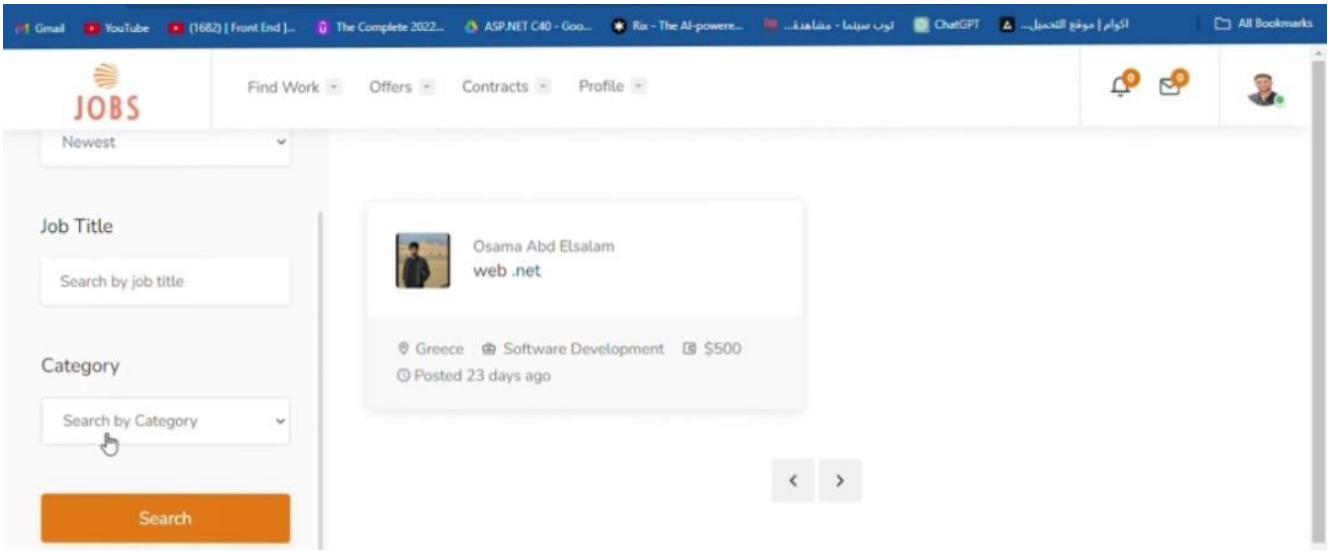

JOBS
Find Talent
Jobs
Contracts
Profile
Contract Status
Ongoing
Contract
D89E9E32-740C-4610-972B-084AC6F74E2D
Contract Budget
$400
NaN days NaN hours NaN minutes NaN seconds left


My Contracts - Jobzy
localhost:44319/Contracts/MyContracts
JOBS
Find Talent
Jobs
Contracts
Profile
Main
Dashboard
Messages
Manage
Jobs
My Jobs
Add a Job
Contracts
My Contracts
My Contracts
1 Contract Listings
Contract #D89E9E32
Ongoing
web Site
5
Delivery Days
$400
Budget
4
Days left

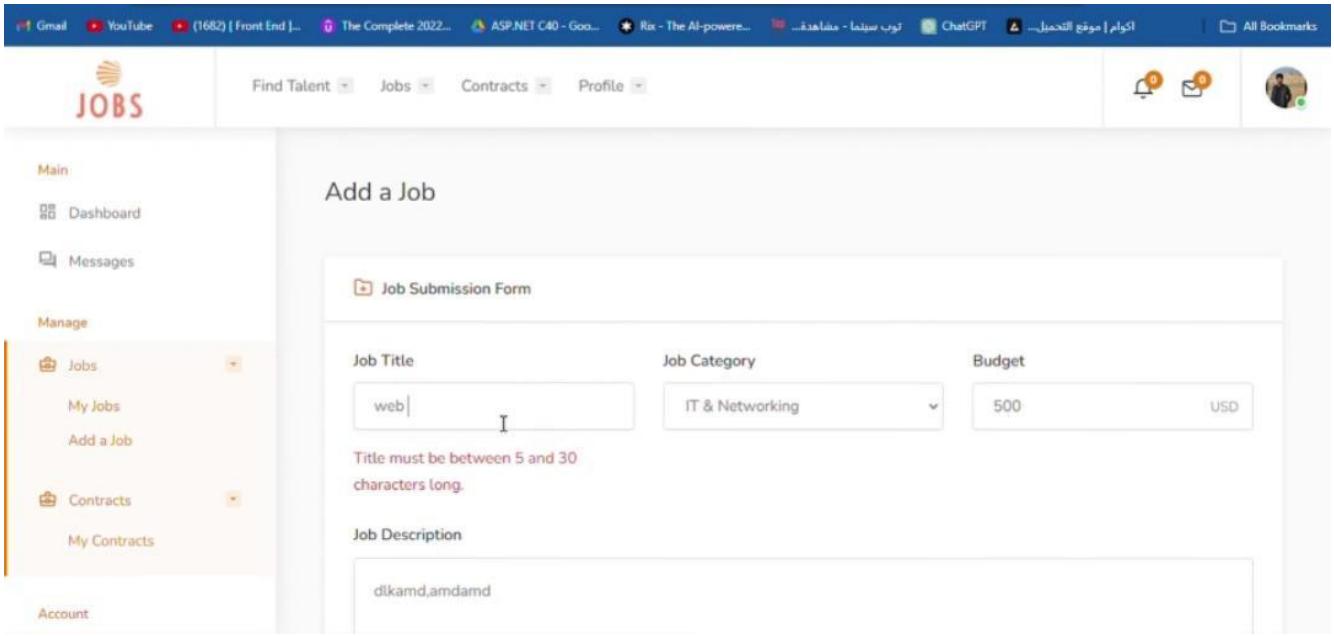
View rate



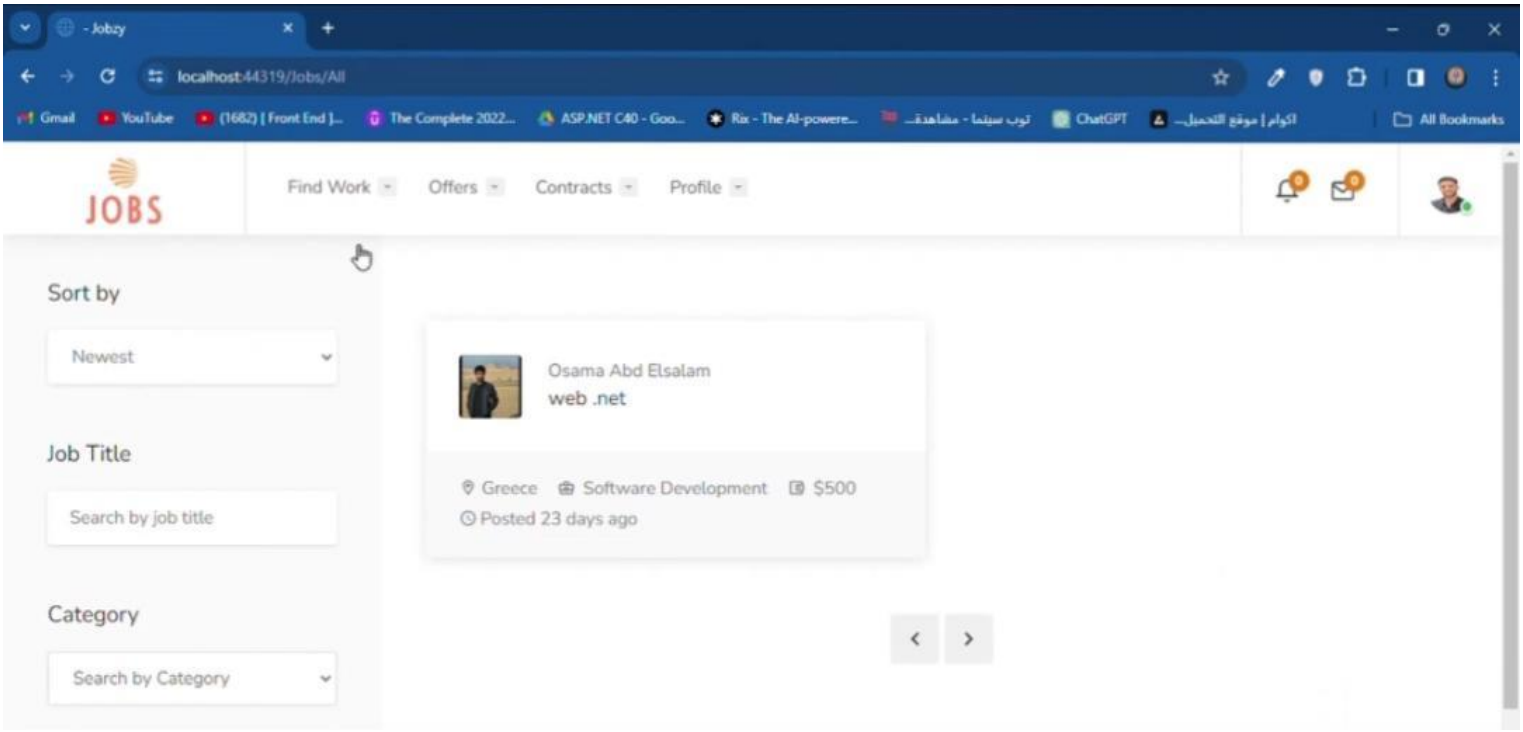
Search



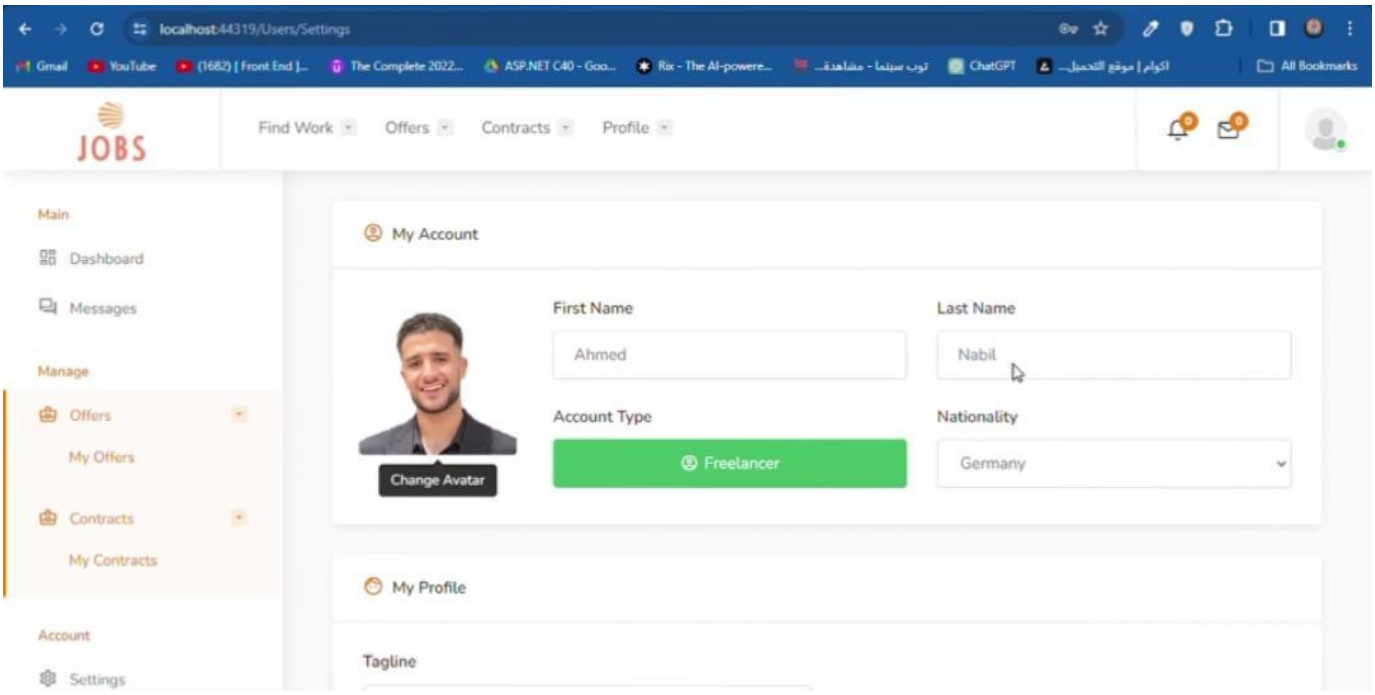
Add job


JOBS
Find Talent
Jobs
Contracts
Profile
Main
Dashboard
Messages
Manage
Jobs
My Jobs
Add a Job
Contracts
My Contracts
Account
Add a Job
Job Submission Form
Job Title
web
Job Category
IT & Networking
Budget
500
USD
Title must be between 5 and 30 characters long.
Job Description
dlkamd,amdamd

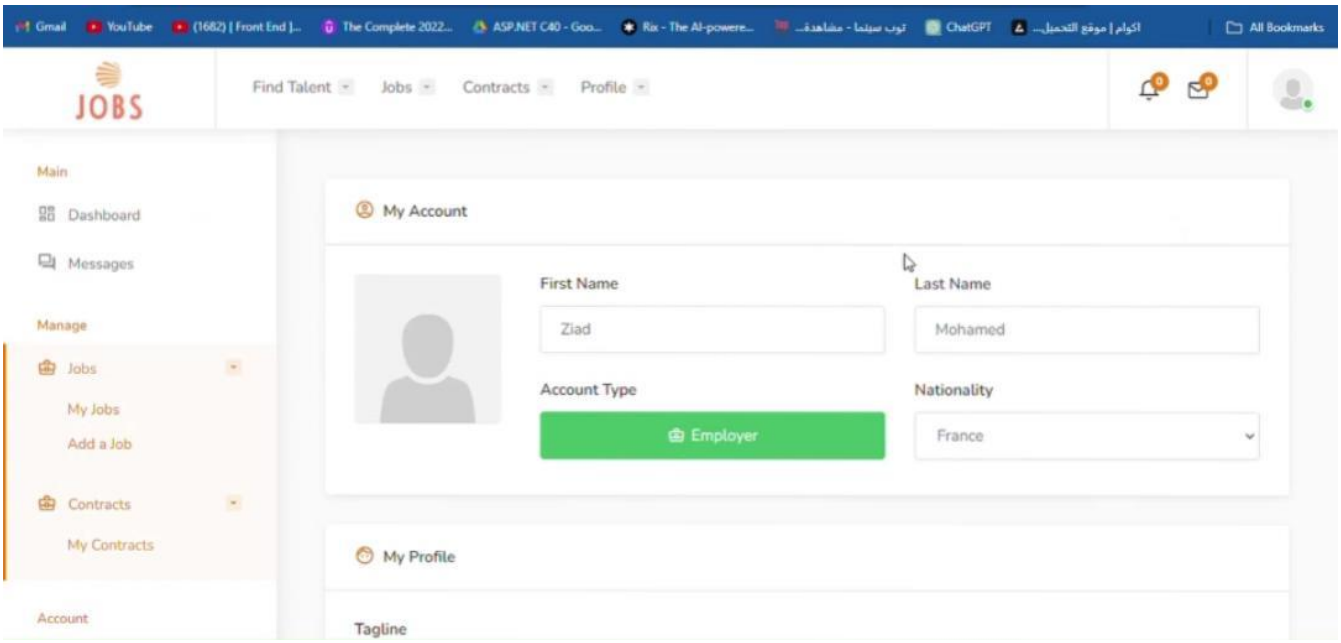
Job


localhost:44319/Jobs/All
JOBS
Find Work
Offers
Contracts
Profile
Sort by
Newest
Job Title
Search by job title
Category
Search by Category
Osama Abd Elsalam
web .net
Greece
Software Development
$500
Posted 23 days ago

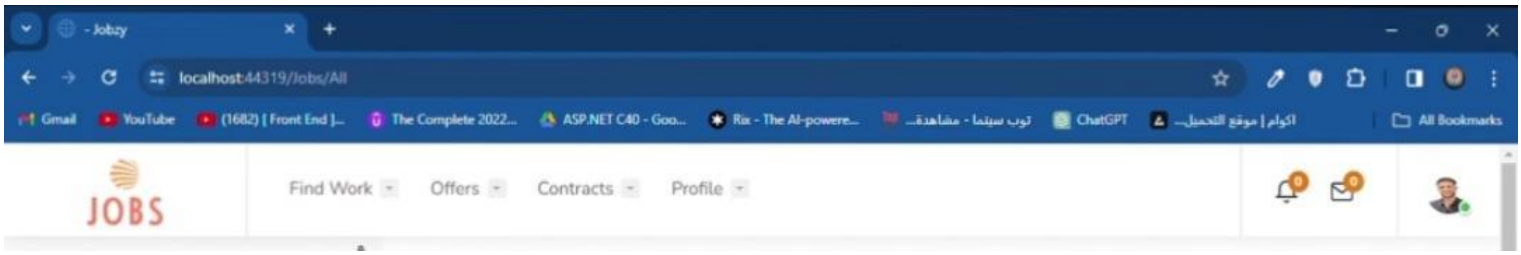
Edit freelancer profile



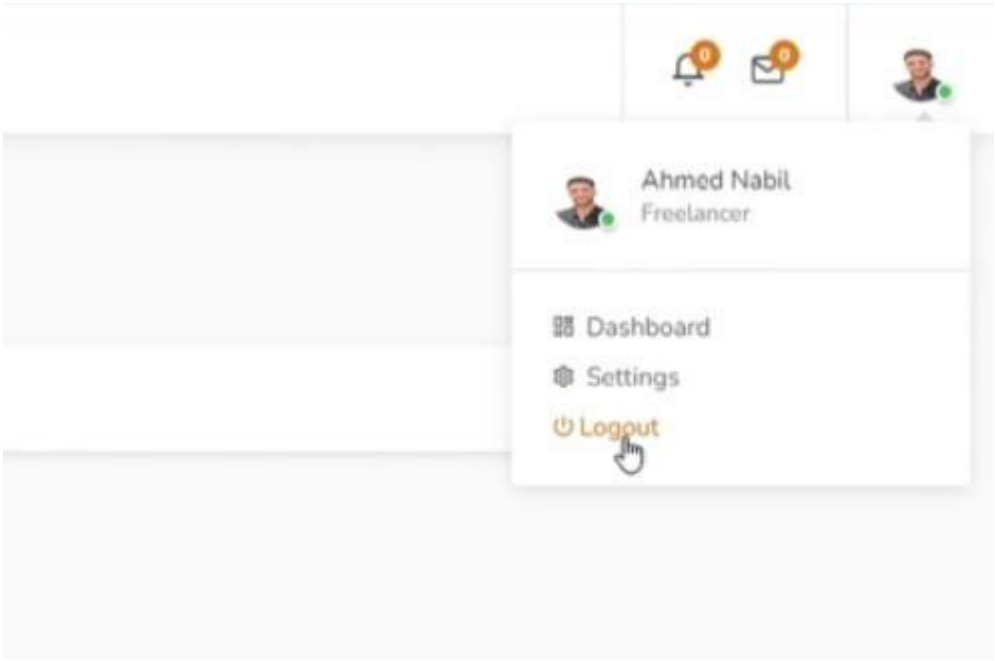
Edit Clint profile



Menu


- Jobzy
localhost:44319/Jobs/All
Gmail
YouTube
(1682) | Front End |...
The Complete 2022...
ASP.NET C40 - Goo...
Rix - The AI-powere...
ChatGPT
All Bookmarks
JOBS
Find Work
Offers
Contracts
Profile

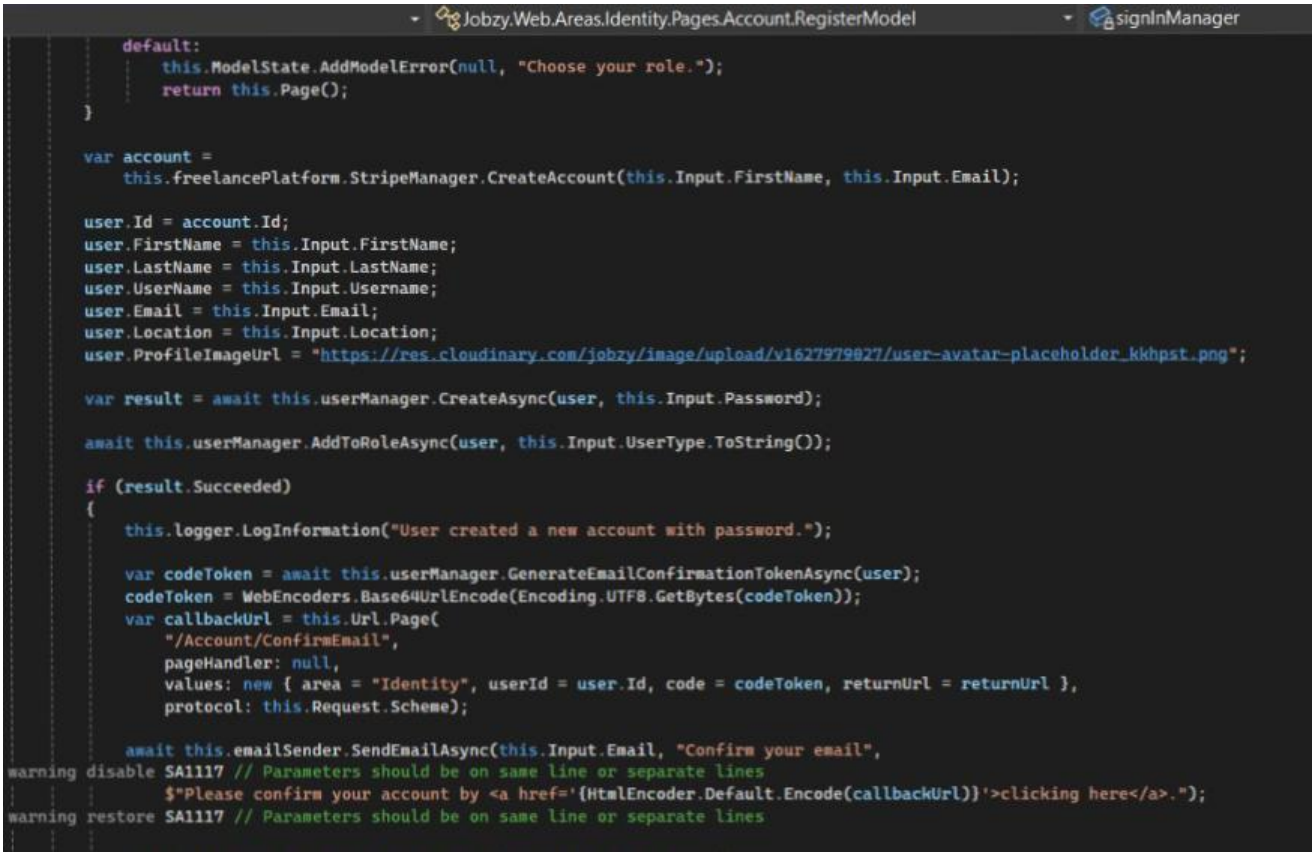
Log out

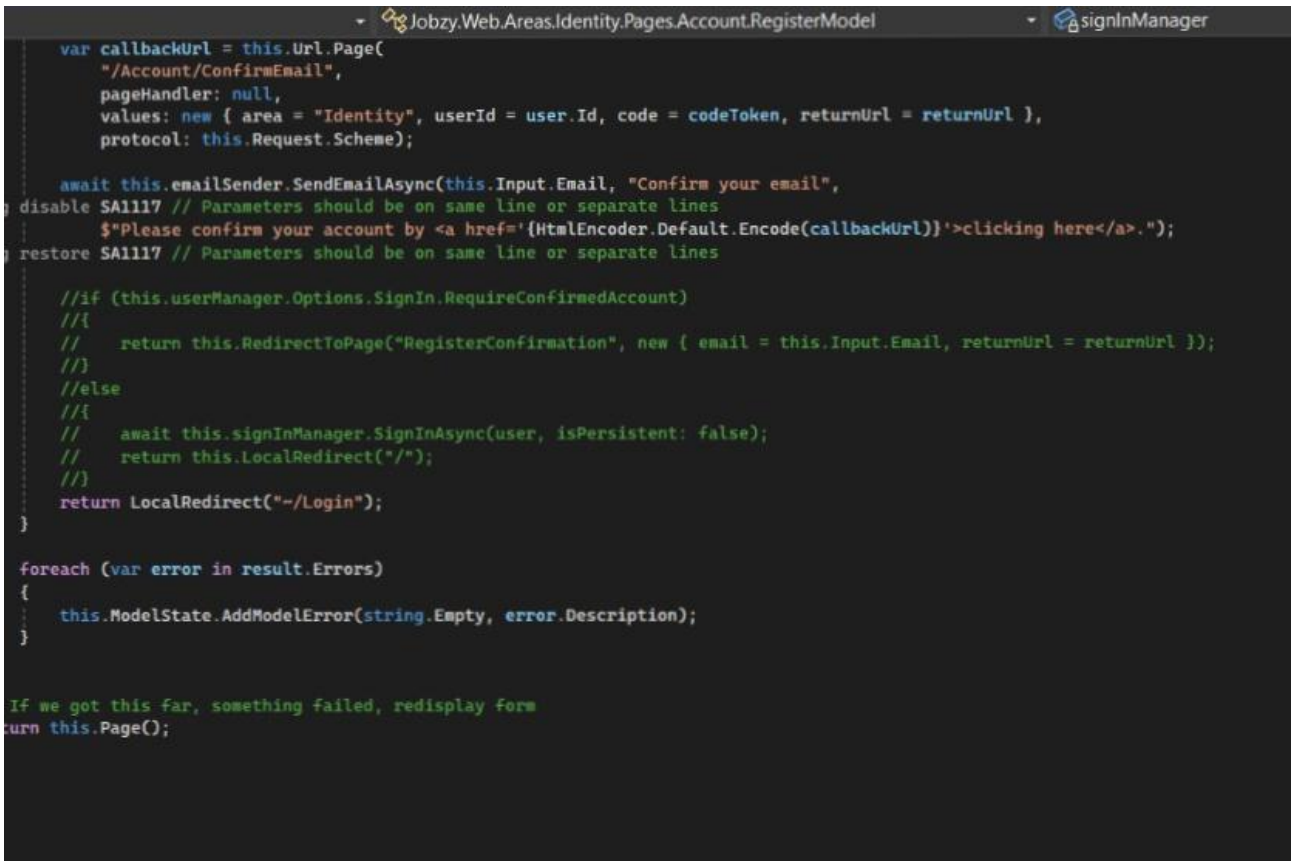

Ahmed Nabil
Freelancer
Dashboard
Settings
Logout

Software architecture .Sign up

```razor
@{
    ViewData["Title"] = "Register";
}

<div class="margin-top-30"></div>

<div class="container">
    <div class="row">
        <div class="col-xl-6 offset-xl-3">
            <div class="login-register-page">
                <!-- Welcome Text -->
                <div class="welcome-text">
                    <h3 style="font-size: 26px;">Let's create your account!</h3>
                    <span>Already have an account? <a href="Login">Log In!</a></span>
                </div>

                <div asp-validation-summary="All" class="text-danger"></div>

                <!-- Form -->
                <form asp-route-returnUrl="@Model.ReturnUrl" method="post" id="register-form">
                    <div class="account-type">
                        <div>
                            <input asp-for="Input.UserType" value="@UserType.Freelancer" type="radio" id="freelancer-radio" class="account-type-radio" />
                            <label asp-for="Input.UserType" for="freelancer-radio" class="ripple-effect-dark"><i class="icon-material-outline-account-circle"></i> Freelancer</label>
                        </div>

                        <div>
                            <input asp-for="Input.UserType" value="@UserType.Employer" type="radio" id="employer-radio" class="account-type-radio" />
                            <label asp-for="Input.UserType" for="employer-radio" class="ripple-effect-dark"><i class="icon-material-outline-business-center"></i> Employer</label>
                        </div>
                    </div>

                    <span asp-validation-for="Input.Location" class="text-danger"></span>
                    <div class="input-with-icon-left">
                        <select asp-for="Input.Location" asp-items="Html.GetEnumSelectList<Country>()" class="location-select dropdown with-border" data-size="5" title="Select Your Location">
                            <option value="0">Select Your Location</option>
                        </select>
                    </div>

                    <span asp-validation-for="Input.FirstName" class="text-danger"></span>
                    <div class="input-with-icon-left">
                        <i class="icon-material-outline-account-circle"></i>
                        <input asp-for="Input.FirstName" type="text" class="input-text with-border" placeholder="First Name" required />
                    </div>
```

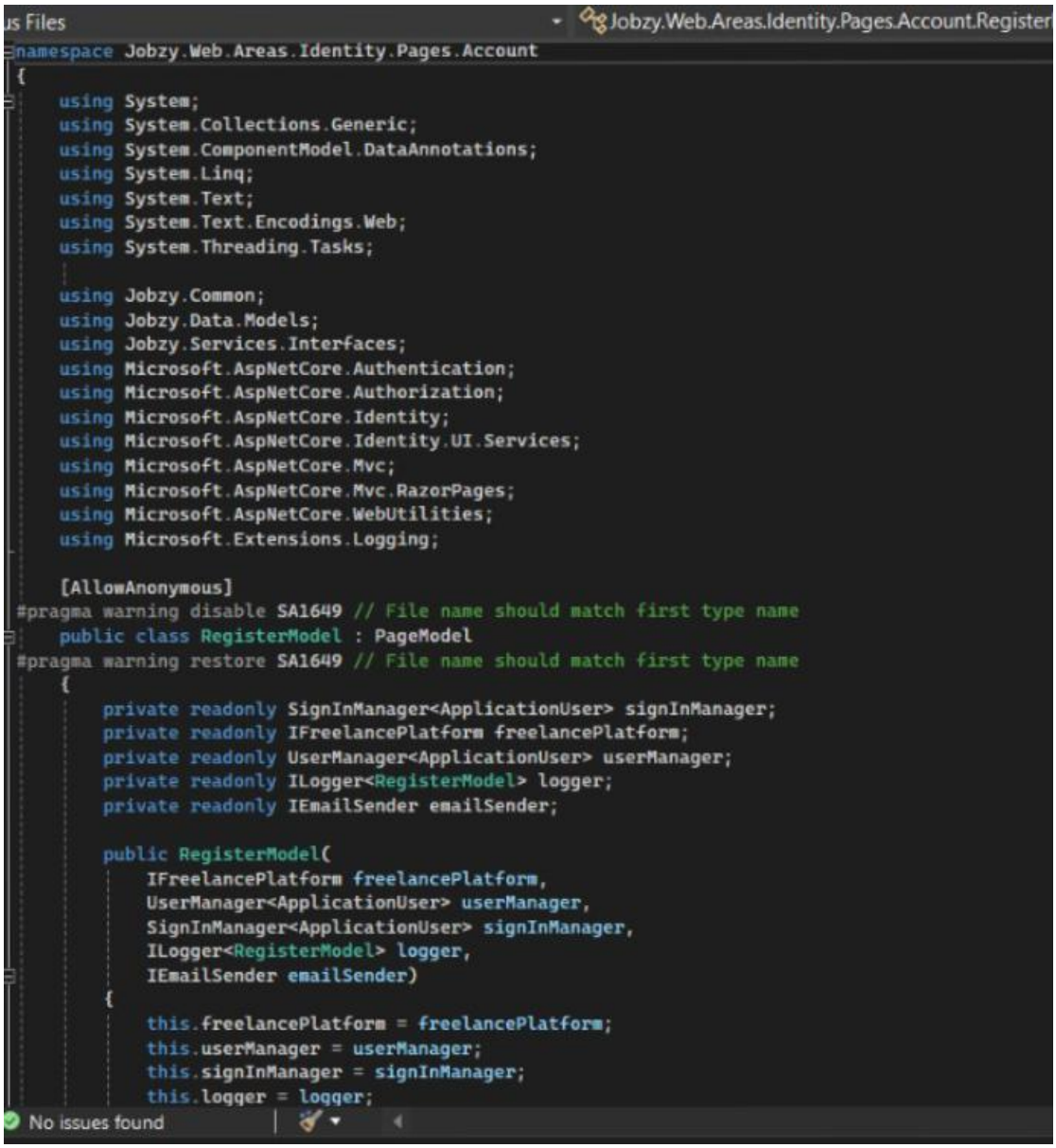
```
s Files                    Jobzy.Web.Areas.Identity.Pages.Account.RegisterModel          signInManager
            public string Email { get; set; }

            [Required]
            [Range(1, 40, ErrorMessage = "Choose a location.")]
            [Display(Name = "Location")]
            public Country Location { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be between {2} and {1} characters long.", MinimumLength = 6)]
            [DataType(DataType.Password)]
            [Display(Name = "Password")]
            public string Password { get; set; }

            [DataType(DataType.Password)]
            [Display(Name = "Confirm password")]
            [Compare("Password", ErrorMessage = "The Password and confirmation password do not match.")]
            public string ConfirmPassword { get; set; }
        }

#pragma warning disable SA1201 // Elements should appear in the correct order
        public async Task OnGetAsync(string code)
#pragma warning restore SA1201 // Elements should appear in the correct order
        {
            this.ExternalLogins = (await this.signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
        }

        public async Task<IActionResult> OnPostAsync(string returnUrl = null)
        {
            returnUrl ??= this.Url.Content("~/");
            this.ExternalLogins = (await this.signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
            if (this.ModelState.IsValid)
            {
                ApplicationUser user;

                switch (this.Input.UserType)
                {
                    case UserType.Freelancer:
                        user = new Freelancer();
                        break;
                    case UserType.Employer:
                        user = new Employer();
                        break;
                    default:
                        this.ModelState.AddModelError(null, "Choose your role.");
No issues found
```


Jobzy.Web.Areas.Identity.Pages.Account.RegisterModel
signInManager
default:
this.ModelState.AddModelError(null, "Choose your role.");
return this.Page();
}
var account =
this.freelancePlatform.StripeManager.CreateAccount(this.Input.FirstName, this.Input.Email);
user.Id = account.Id;
user.FirstName = this.Input.FirstName;
user.LastName = this.Input.LastName;
user.UserName = this.Input.Username;
user.Email = this.Input.Email;
user.Location = this.Input.Location;
user.ProfileImageUrl = "https://res.cloudinary.com/jobzy/image/upload/v1627979027/user-avatar-placeholder_kkhpst.png";
var result = await this.userManager.CreateAsync(user, this.Input.Password);
await this.userManager.AddToRoleAsync(user, this.Input.UserType.ToString());
if (result.Succeeded)
{
this.logger.LogInformation("User created a new account with password.");
var codeToken = await this.userManager.GenerateEmailConfirmationTokenAsync(user);
codeToken = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(codeToken));
var callbackUrl = this.Url.Page(
"/Account/ConfirmEmail",
pageHandler: null,
values: new { area = "Identity", userId = user.Id, code = codeToken, returnUrl = returnUrl },
protocol: this.Request.Scheme);
await this.emailSender.SendEmailAsync(this.Input.Email, "Confirm your email",
warning disable SA1117 // Parameters should be on same line or separate lines
$"Please confirm your account by clicking here.");
warning restore SA1117 // Parameters should be on same line or separate lines

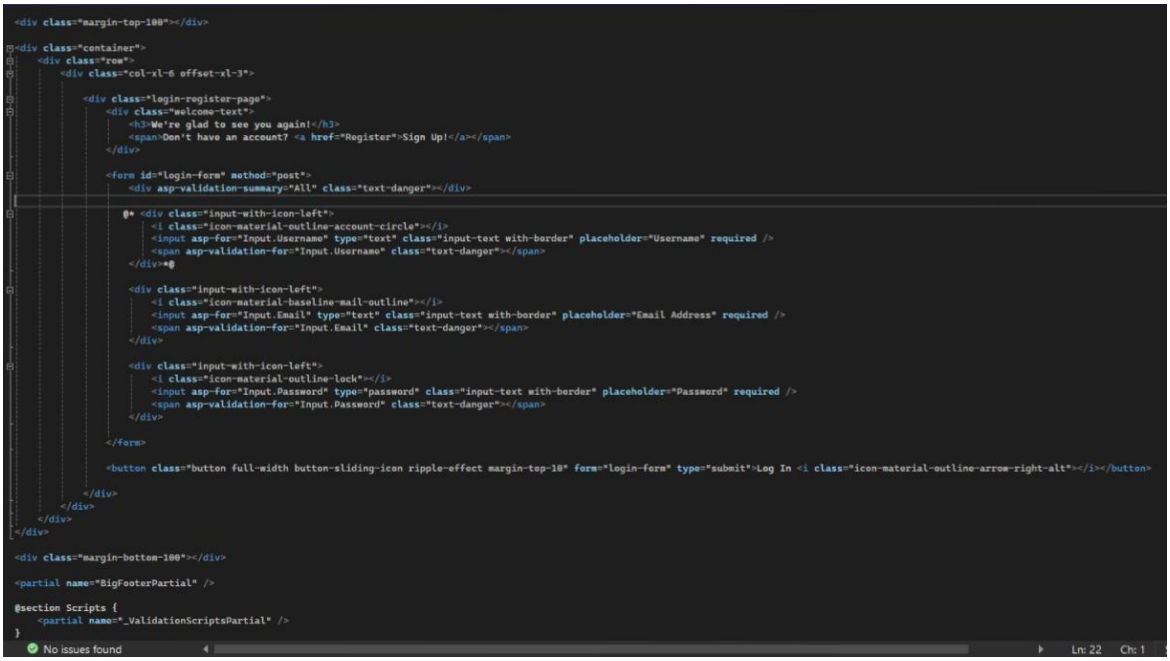

Jobzy.Web.Areas.Identity.Pages.Account.RegisterModel
signInManager
var callbackUrl = this.Url.Page(
"/Account/ConfirmEmail",
pageHandler: null,
values: new { area = "Identity", userId = user.Id, code = codeToken, returnUrl = returnUrl },
protocol: this.Request.Scheme);
await this.emailSender.SendEmailAsync(this.Input.Email, "Confirm your email",
disable SA1117 // Parameters should be on same line or separate lines
$"Please confirm your account by clicking here.");
restore SA1117 // Parameters should be on same line or separate lines
//if (this.userManager.Options.SignIn.RequireConfirmedAccount)
//{
// return this.RedirectToPage("RegisterConfirmation", new { email = this.Input.Email, returnUrl = returnUrl });
//}
//else
//{
// await this.signInManager.SignInAsync(user, isPersistent: false);
// return this.LocalRedirect("/");
//}
return LocalRedirect("~/Login");
}
foreach (var error in result.Errors)
{
this.ModelState.AddModelError(string.Empty, error.Description);
}
If we got this far, something failed, redisplay form
turn this.Page();

```
                        <input asp-for="Input.ConfirmPassword" type="password" class="input-text with-border" placeholder="Repeat Password"
                    </div>

                    <button class="button full-width button-sliding-icon ripple-effect margin-top-10" type="submit" form="register-form">Re
                </form>
            </div>

        </div>
    </div>
</div>

<div class="margin-bottom-100"></div>

<partial name="BigFooterPartial"/>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />

    <script>
        let freelancerInput = document.getElementById("freelancer-radio");
        let employerInput = document.getElementById("employer-radio");

        freelancerInput.setAttribute("checked", true);

        employerInput.addEventListener("click", () => {
            freelancerInput.removeAttribute("checked");
            employerInput.setAttribute("checked", true);
        })

        freelancerInput.addEventListener("click", () => {
            employerInput.removeAttribute("checked");
            freelancerInput.setAttribute("checked", true);
        })
    </script>
}
```

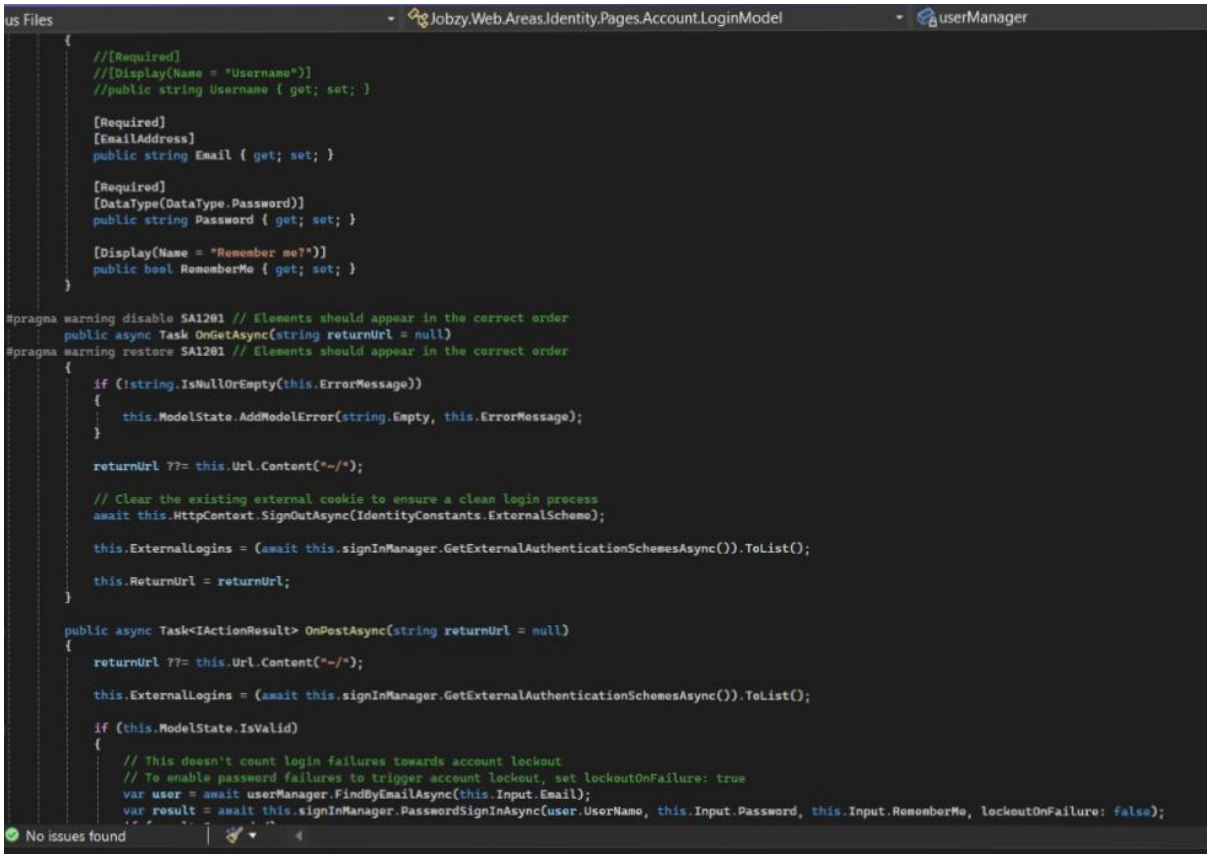
```
        <span asp-validation-for="Input.FirstName" class="text-danger"></span>
        <div class="input-with-icon-left">
            <i class="icon-material-outline-account-circle"></i>
            <input asp-for="Input.FirstName" type="text" class="input-text with-border" placeholder="First Name" required />
        </div>

        <span asp-validation-for="Input.LastName" class="text-danger"></span>
        <div class="input-with-icon-left">
            <i class="icon-material-outline-account-circle"></i>
            <input asp-for="Input.LastName" type="text" class="input-text with-border" placeholder="Last Name" required />
        </div>


        <span asp-validation-for="Input.Username" class="text-danger"></span>
        <div class="input-with-icon-left">
            <i class="icon-material-outline-account-circle"></i>
            <input asp-for="Input.Username" type="text" class="input-text with-border" placeholder="Username" required />
        </div>


        <span asp-validation-for="Input.Email" class="text-danger"></span>
        <div class="input-with-icon-left">
            <i class="icon-material-baseline-mail-outline"></i>
            <input asp-for="Input.Email" type="text" class="input-text with-border" placeholder="Email Address" required />
        </div>

        <span asp-validation-for="Input.Password" class="text-danger"></span>
        <div class="input-with-icon-left" title="Should be at least 8 characters long" data-tippy-placement="bottom">
            <i class="icon-material-outline-lock"></i>
            <input asp-for="Input.Password" type="password" class="input-text with-border" placeholder="Password" required />
        </div>


        <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
        <div class="input-with-icon-left">
            <i class="icon-material-outline-lock"></i>
            <input asp-for="Input.ConfirmPassword" type="password" class="input-text with-border" placeholder="Repeat Password" required />
        </div>

        <button class="button full-width button-sliding-icon ripple-effect margin-top-10" type="submit" form="register-form">Register <i class="icon-material-outline-arrow-
    </form>
  </div>

</div>
ues found
```

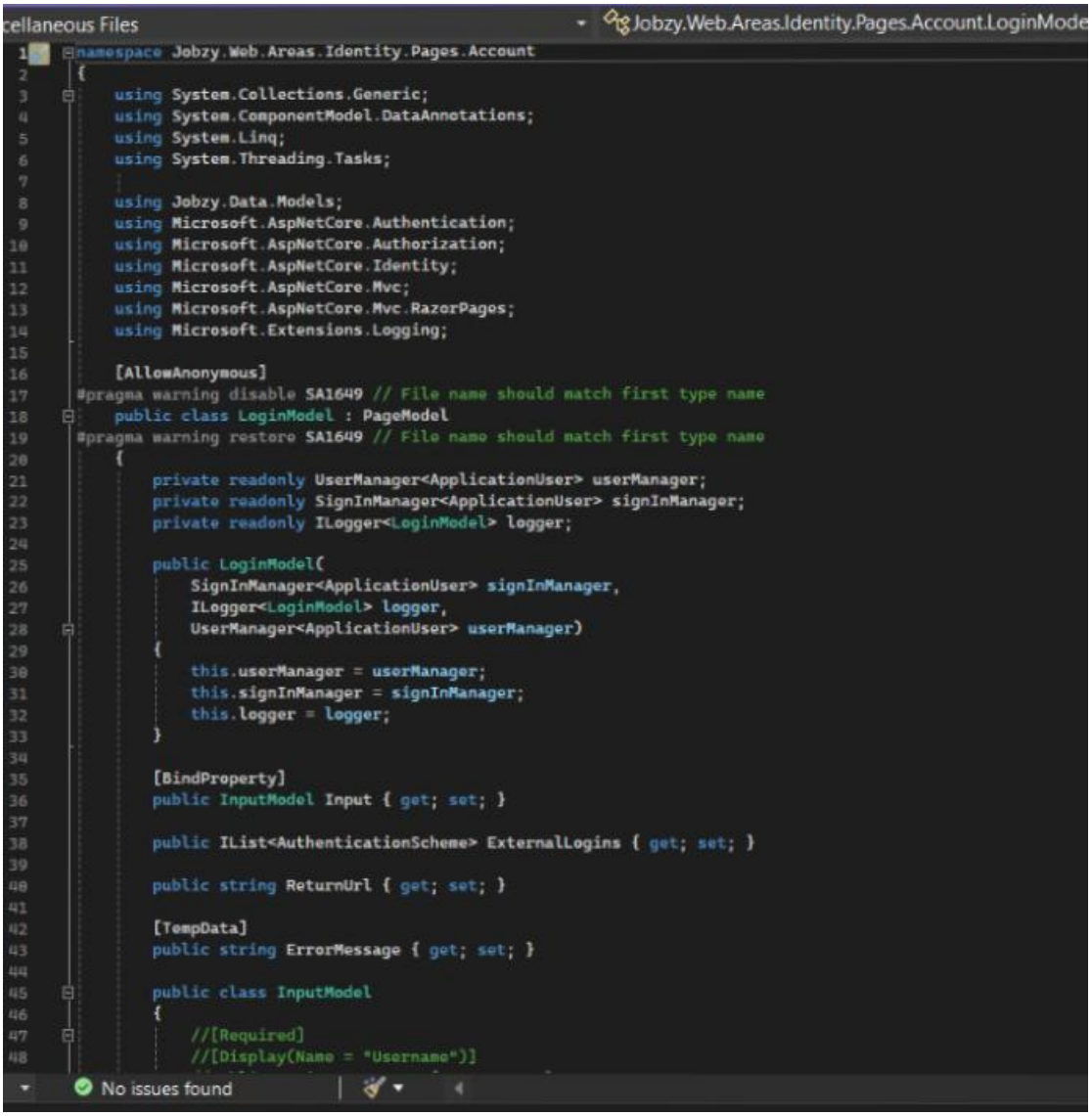

Jobzy.Web.Areas.Identity.Pages.Account.Register
namespace Jobzy.Web.Areas.Identity.Pages.Account
{
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Text.Encodings.Web;
using System.Threading.Tasks;
using Jobzy.Common;
using Jobzy.Data.Models;
using Jobzy.Services.Interfaces;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.WebUtilities;
using Microsoft.Extensions.Logging;
[AllowAnonymous]
#pragma warning disable SA1649 // File name should match first type name
public class RegisterModel : PageModel
#pragma warning restore SA1649 // File name should match first type name
{
private readonly SignInManager<ApplicationUser> signInManager;
private readonly IFreelancePlatform freelancePlatform;
private readonly UserManager<ApplicationUser> userManager;
private readonly ILogger<RegisterModel> logger;
private readonly IEmailSender emailSender;
public RegisterModel(
IFreelancePlatform freelancePlatform,
UserManager<ApplicationUser> userManager,
SignInManager<ApplicationUser> signInManager,
ILogger<RegisterModel> logger,
IEmailSender emailSender)
{
this.freelancePlatform = freelancePlatform;
this.userManager = userManager;
this.signInManager = signInManager;
this.logger = logger;
No issues found

Log in

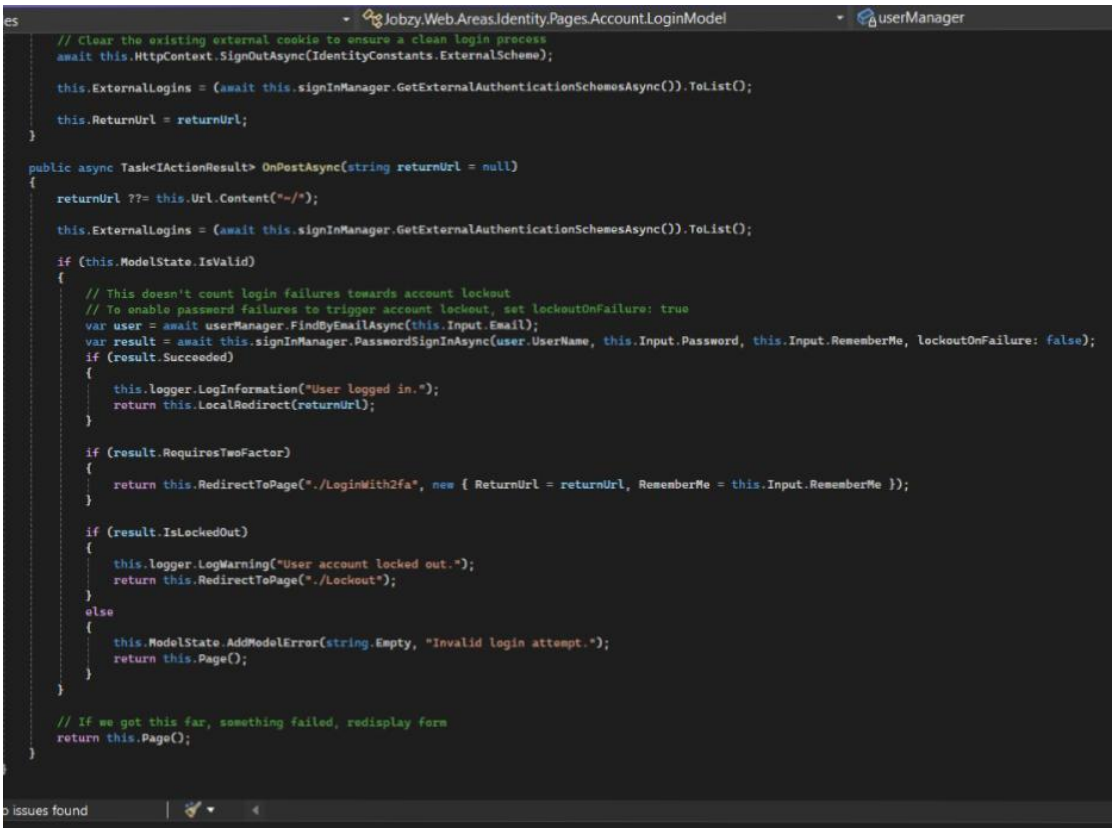


```
<div class="margin-top-100"></div>

<div class="container">
    <div class="row">
        <div class="col-xl-6 offset-xl-3">

            <div class="login-register-page">
                <div class="welcome-text">
                    <h3>We're glad to see you again!</h3>
                    <span>Don't have an account? <a href="Register">Sign Up!</a></span>
                </div>

                <form id="login-form" method="post">
                    <div asp-validation-summary="All" class="text-danger"></div>

                    @* <div class="input-with-icon-left">
                        <i class="icon-material-outline-account-circle"></i>
                        <input asp-for="Input.Username" type="text" class="input-text with-border" placeholder="Username" required />
                        <span asp-validation-for="Input.Username" class="text-danger"></span>
                    </div>*@

                    <div class="input-with-icon-left">
                        <i class="icon-material-baseline-mail-outline"></i>
                        <input asp-for="Input.Email" type="text" class="input-text with-border" placeholder="Email Address" required />
                        <span asp-validation-for="Input.Email" class="text-danger"></span>
                    </div>

                    <div class="input-with-icon-left">
                        <i class="icon-material-outline-lock"></i>
                        <input asp-for="Input.Password" type="password" class="input-text with-border" placeholder="Password" required />
                        <span asp-validation-for="Input.Password" class="text-danger"></span>
                    </div>

                </form>

                <button class="button full-width button-sliding-icon ripple-effect margin-top-10" form="login-form" type="submit">Log In <i class="icon-material-outline-arrow-right-alt"></i></button>

            </div>
        </div>
    </div>
</div>

<div class="margin-bottom-100"></div>

<partial name="BigFooterPartial" />

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

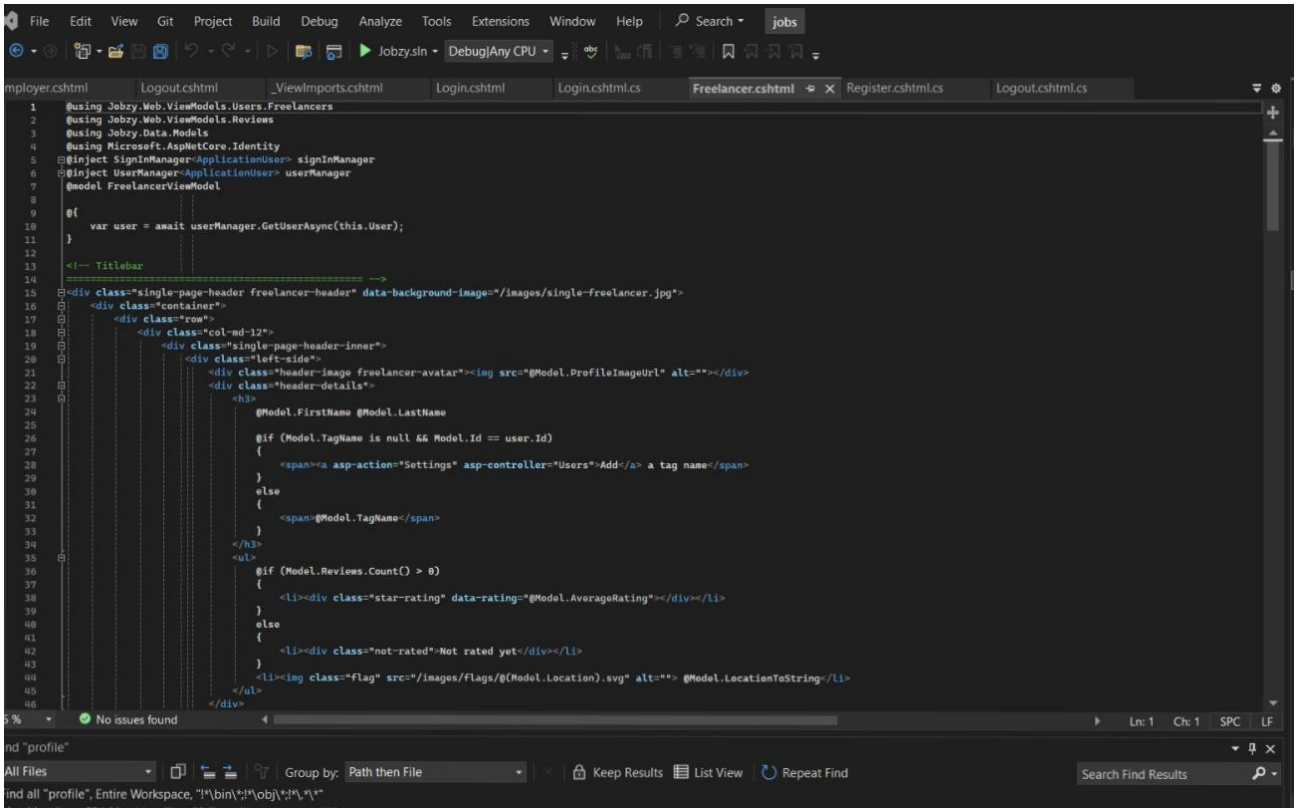

us Files
Jobzy.Web.Areas.Identity.Pages.Account.LoginModel
userManager
{
//[Required]
//[Display(Name = "Username")]
//public string Username { get; set; }
[Required]
[EmailAddress]
public string Email { get; set; }
[Required]
[DataType(DataType.Password)]
public string Password { get; set; }
[Display(Name = "Remember me?")]
public bool RememberMe { get; set; }
}
#pragma warning disable SA1201 // Elements should appear in the correct order
public async Task OnGetAsync(string returnUrl = null)
#pragma warning restore SA1201 // Elements should appear in the correct order
{
if (!string.IsNullOrEmpty(this.ErrorMessage))
{
this.ModelState.AddModelError(string.Empty, this.ErrorMessage);
}
returnUrl ??= this.Url.Content("~/");
// Clear the existing external cookie to ensure a clean login process
await this.HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);
this.ExternalLogins = (await this.signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
this.ReturnUrl = returnUrl;
}
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
returnUrl ??= this.Url.Content("~/");
this.ExternalLogins = (await this.signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
if (this.ModelState.IsValid)
{
// This doesn't count login failures towards account lockout
// To enable password failures to trigger account lockout, set lockoutOnFailure: true
var user = await userManager.FindByEmailAsync(this.Input.Email);
var result = await this.signInManager.PasswordSignInAsync(user.UserName, this.Input.Password, this.Input.RememberMe, lockoutOnFailure: false);
No issues found



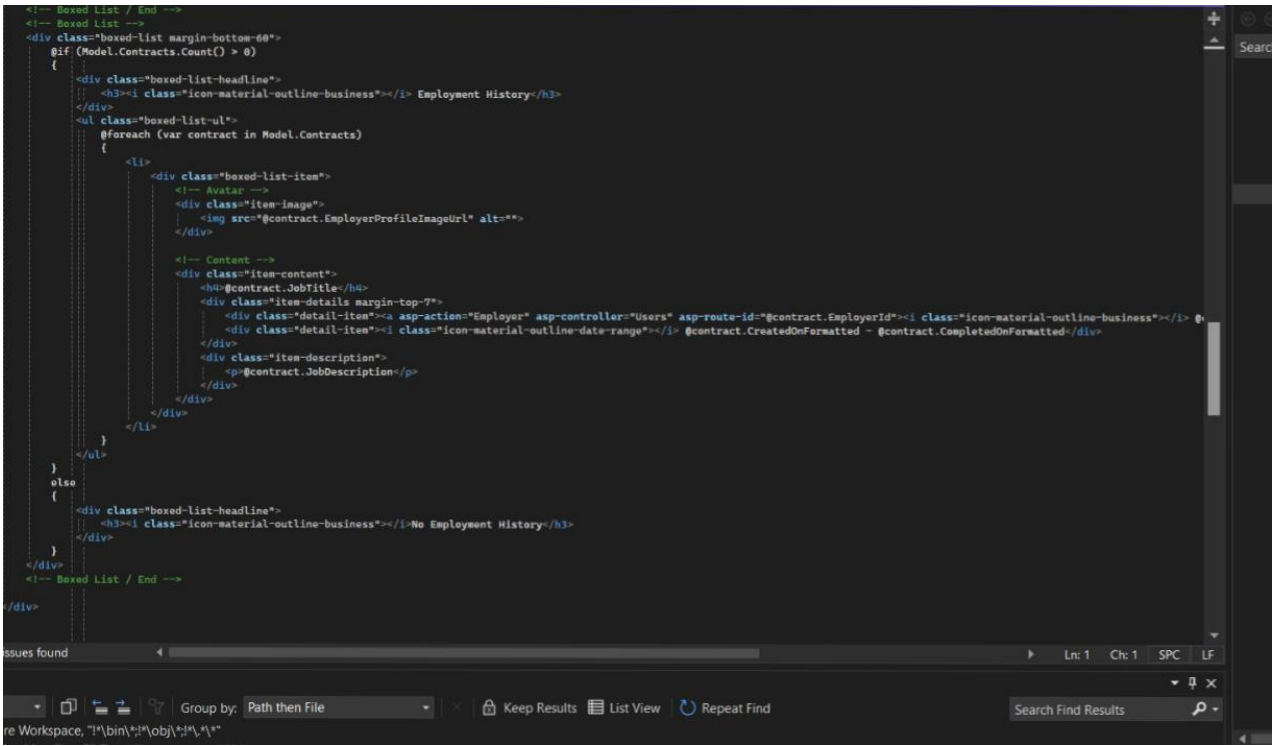

```
namespace Jobzy.Web.Areas.Identity.Pages.Account
{
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.Linq;
    using System.Threading.Tasks;

    using Jobzy.Data.Models;
    using Microsoft.AspNetCore.Authentication;
    using Microsoft.AspNetCore.Authorization;
    using Microsoft.AspNetCore.Identity;
    using Microsoft.AspNetCore.Mvc;
    using Microsoft.AspNetCore.Mvc.RazorPages;
    using Microsoft.Extensions.Logging;

    [AllowAnonymous]
#pragma warning disable SA1649 // File name should match first type name
    public class LoginModel : PageModel
#pragma warning restore SA1649 // File name should match first type name
    {
        private readonly UserManager<ApplicationUser> userManager;
        private readonly SignInManager<ApplicationUser> signInManager;
        private readonly ILogger<LoginModel> logger;

        public LoginModel(
            SignInManager<ApplicationUser> signInManager,
            ILogger<LoginModel> logger,
            UserManager<ApplicationUser> userManager)
        {
            this.userManager = userManager;
            this.signInManager = signInManager;
            this.logger = logger;
        }

        [BindProperty]
        public InputModel Input { get; set; }

        public IList<AuthenticationScheme> ExternalLogins { get; set; }

        public string ReturnUrl { get; set; }

        [TempData]
        public string ErrorMessage { get; set; }

        public class InputModel
        {
            //[Required]
            //[Display(Name = "Username")]
```


es
Jobzy.Web.Areas.Identity.Pages.Account.LoginModel
userManager
// Clear the existing external cookie to ensure a clean login process
await this.HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);
this.ExternalLogins = (await this.signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
this.ReturnUrl = returnUrl;
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
returnUrl ??= this.Url.Content("~/");
this.ExternalLogins = (await this.signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
if (this.ModelState.IsValid)
// This doesn't count login failures towards account lockout
// To enable password failures to trigger account lockout, set lockoutOnFailure: true
var user = await userManager.FindByEmailAsync(this.Input.Email);
var result = await this.signInManager.PasswordSignInAsync(user.UserName, this.Input.Password, this.Input.RememberMe, lockoutOnFailure: false);
if (result.Succeeded)
this.logger.LogInformation("User logged in.");
return this.LocalRedirect(returnUrl);
if (result.RequiresTwoFactor)
return this.RedirectToPage("./LoginWith2fa", new { ReturnUrl = returnUrl, RememberMe = this.Input.RememberMe });
if (result.IsLockedOut)
this.logger.LogWarning("User account locked out.");
return this.RedirectToPage("./Lockout");
else
this.ModelState.AddModelError(string.Empty, "Invalid login attempt.");
return this.Page();
// If we got this far, something failed, redisplay form
return this.Page();
o issues found

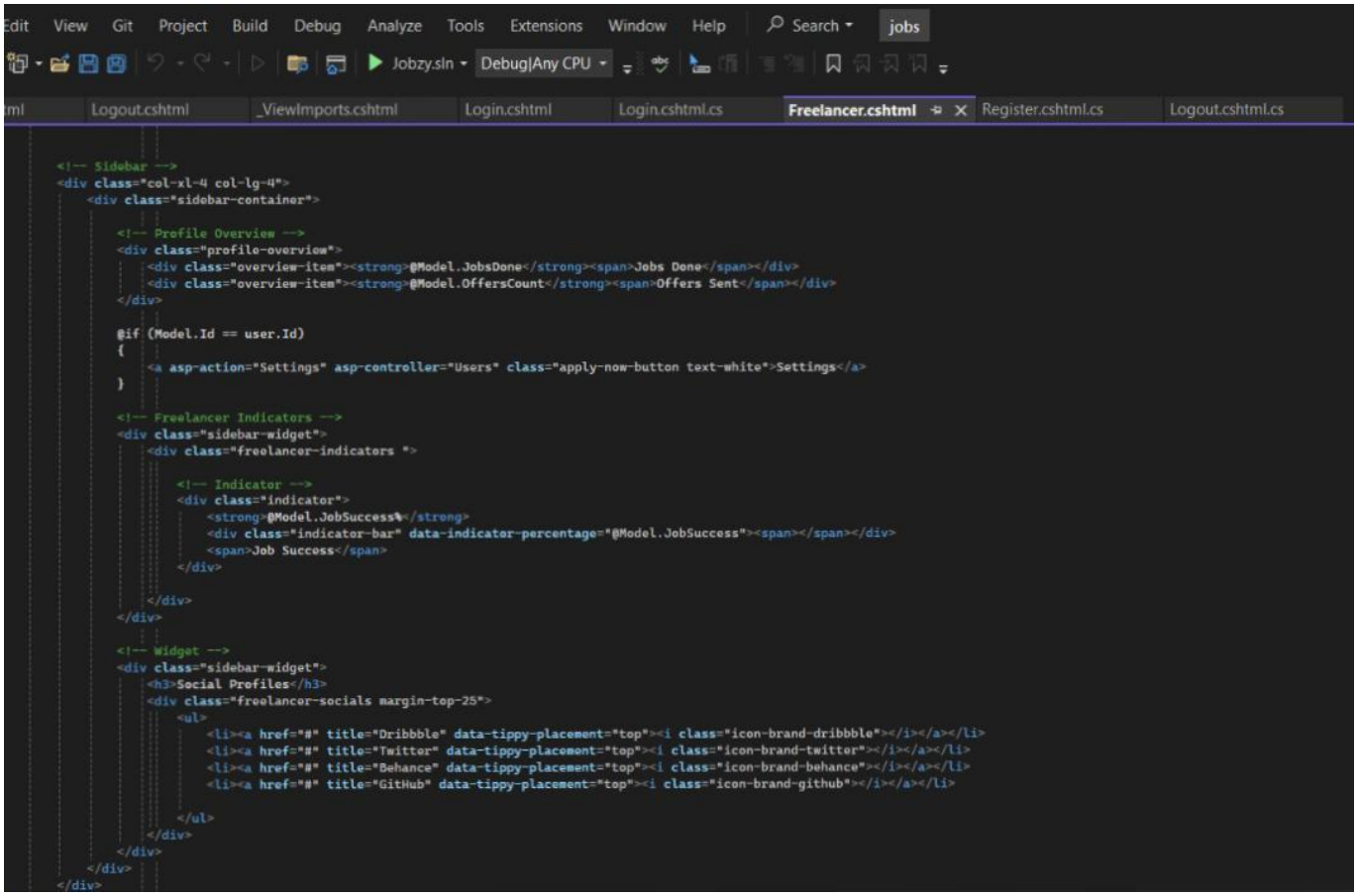
Freelancer profile



```
@using Jobzy.Web.ViewModels.Users.Freelancers
@using Jobzy.Web.ViewModels.Reviews
@using Jobzy.Data.Models
@using Microsoft.AspNetCore.Identity
@inject SignInManager<ApplicationUser> signInManager
@inject UserManager<ApplicationUser> userManager
@model FreelancerViewModel

@{
    var user = await userManager.GetUserAsync(this.User);
}

<!-- Titlebar
============================================ -->
<div class="single-page-header freelancer-header" data-background-image="/images/single-freelancer.jpg">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <div class="single-page-header-inner">
                    <div class="left-side">
                        <div class="header-image freelancer-avatar"><img src="@Model.ProfileImageUrl" alt=""></div>
                        <div class="header-details">
                            <h3>
                                @Model.FirstName @Model.LastName

                                @if (Model.TagName is null && Model.Id == user.Id)
                                {
                                    <span><a asp-action="Settings" asp-controller="Users">Add</a> a tag name</span>
                                }
                                else
                                {
                                    <span>@Model.TagName</span>
                                }
                            </h3>
                            <ul>
                                @if (Model.Reviews.Count() > 0)
                                {
                                    <li><div class="star-rating" data-rating="@Model.AverageRating"></div></li>
                                }
                                else
                                {
                                    <li><div class="not-rated">Not rated yet</div></li>
                                }
                                <li><img class="flag" src="/images/flags/@(Model.Location).svg" alt=""> @Model.LocationToString</li>
                            </ul>
                        </div>
```



```
<!-- Boxed List / End -->
<!-- Boxed List -->
<div class="boxed-list margin-bottom-60">
    @if (Model.Contracts.Count() > 0)
    {
        <div class="boxed-list-headline">
            <h3><i class="icon-material-outline-business"></i> Employment History</h3>
        </div>
        <ul class="boxed-list-ul">
            @foreach (var contract in Model.Contracts)
            {
                <li>
                    <div class="boxed-list-item">
                        <!-- Avatar -->
                        <div class="item-image">
                            <img src="@contract.EmployerProfileImageUrl" alt="">
                        </div>

                        <!-- Content -->
                        <div class="item-content">
                            <h4>@contract.JobTitle</h4>
                            <div class="item-details margin-top-7">
                                <div class="detail-item"><a asp-action="Employer" asp-controller="Users" asp-route-id="@contract.EmployerId"><i class="icon-material-outline-business"></i> @
                                <div class="detail-item"><i class="icon-material-outline-date-range"></i> @contract.CreatedOnFormatted - @contract.CompletedOnFormatted</div>
                            </div>
                            <div class="item-description">
                                <p>@contract.JobDescription</p>
                            </div>
                        </div>
                    </div>
                </li>
            }
        </ul>
    }
    else
    {
        <div class="boxed-list-headline">
            <h3><i class="icon-material-outline-business"></i>No Employment History</h3>
        </div>
    }
</div>
<!-- Boxed List / End -->
</div>
```


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs
<!-- Sidebar -->

<!-- Profile Overview -->

@Model.JobsDoneJobs Done
@Model.OffersCountOffers Sent

@if (Model.Id == user.Id)
{
<a asp-action="Settings" asp-controller="Users" class="apply-now-button text-white">Settings
}
<!-- Freelancer Indicators -->

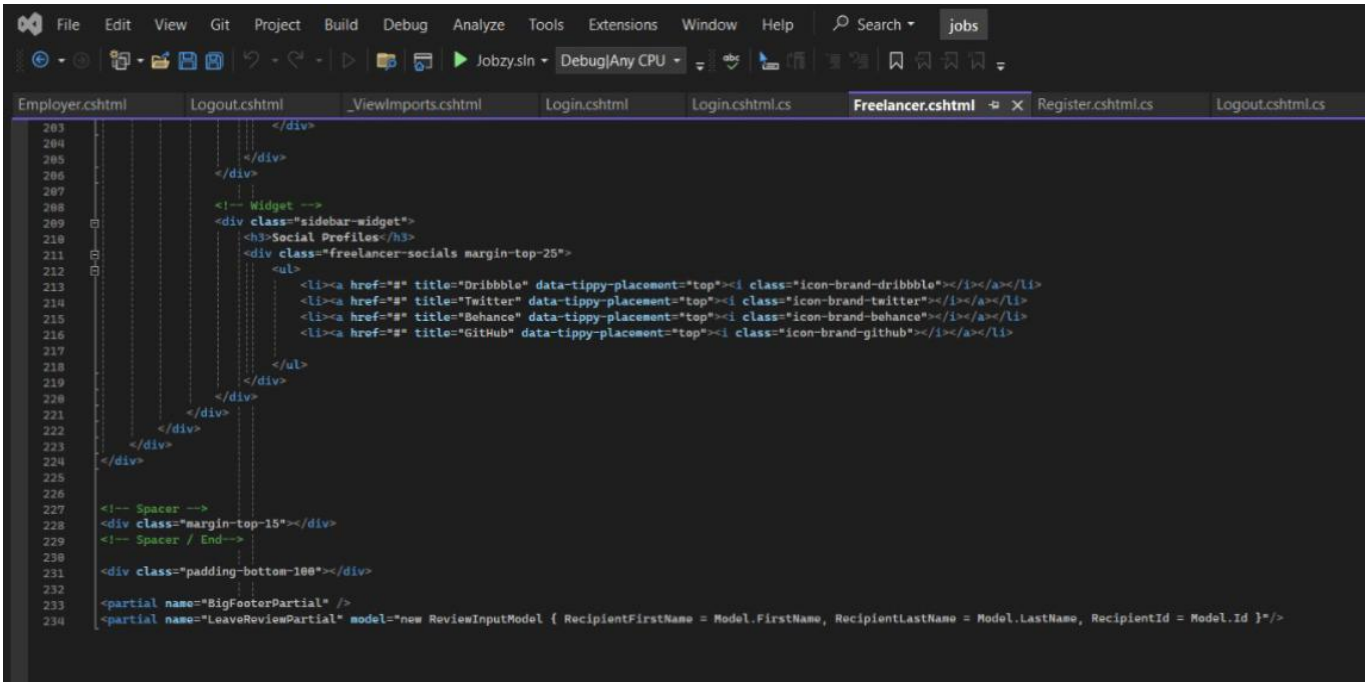
<!-- Indicator -->

@Model.JobSuccess%

Job Success

<!-- Widget -->

<h3>Social Profiles</h3>

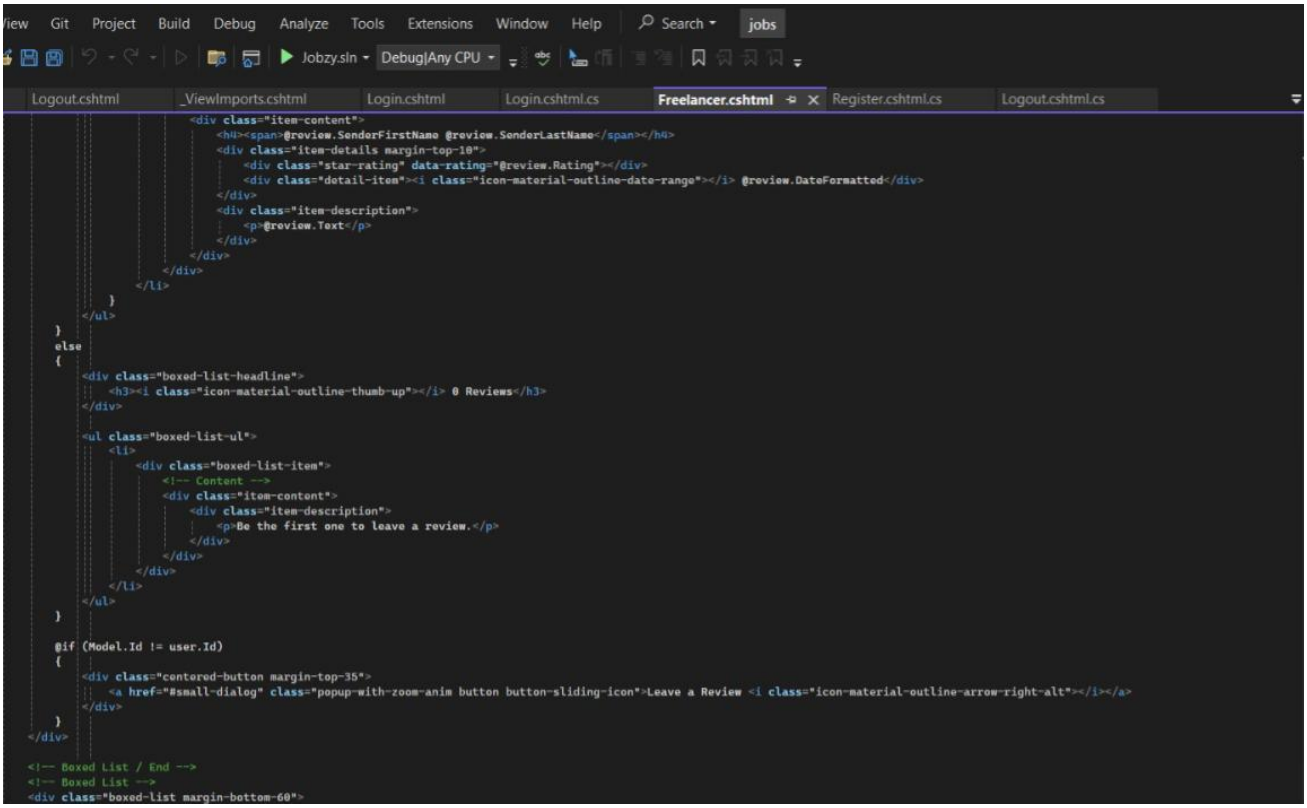

File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs
<!-- Widget -->

<h3>Social Profiles</h3>

<!-- Spacer -->

<!-- Spacer / End-->

<partial name="BigFooterPartial" />
<partial name="LeaveReviewPartial" model="new ReviewInputModel { RecipientFirstName = Model.FirstName, RecipientLastName = Model.LastName, RecipientId = Model.Id }"/>


Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs

<h4>@review.SenderFirstName @review.SenderLastName</h4>

 @review.DateFormatted

@review.Text

}

}
else
{

<h3> 0 Reviews</h3>

<ul class="boxed-list-ul">

<!-- Content -->

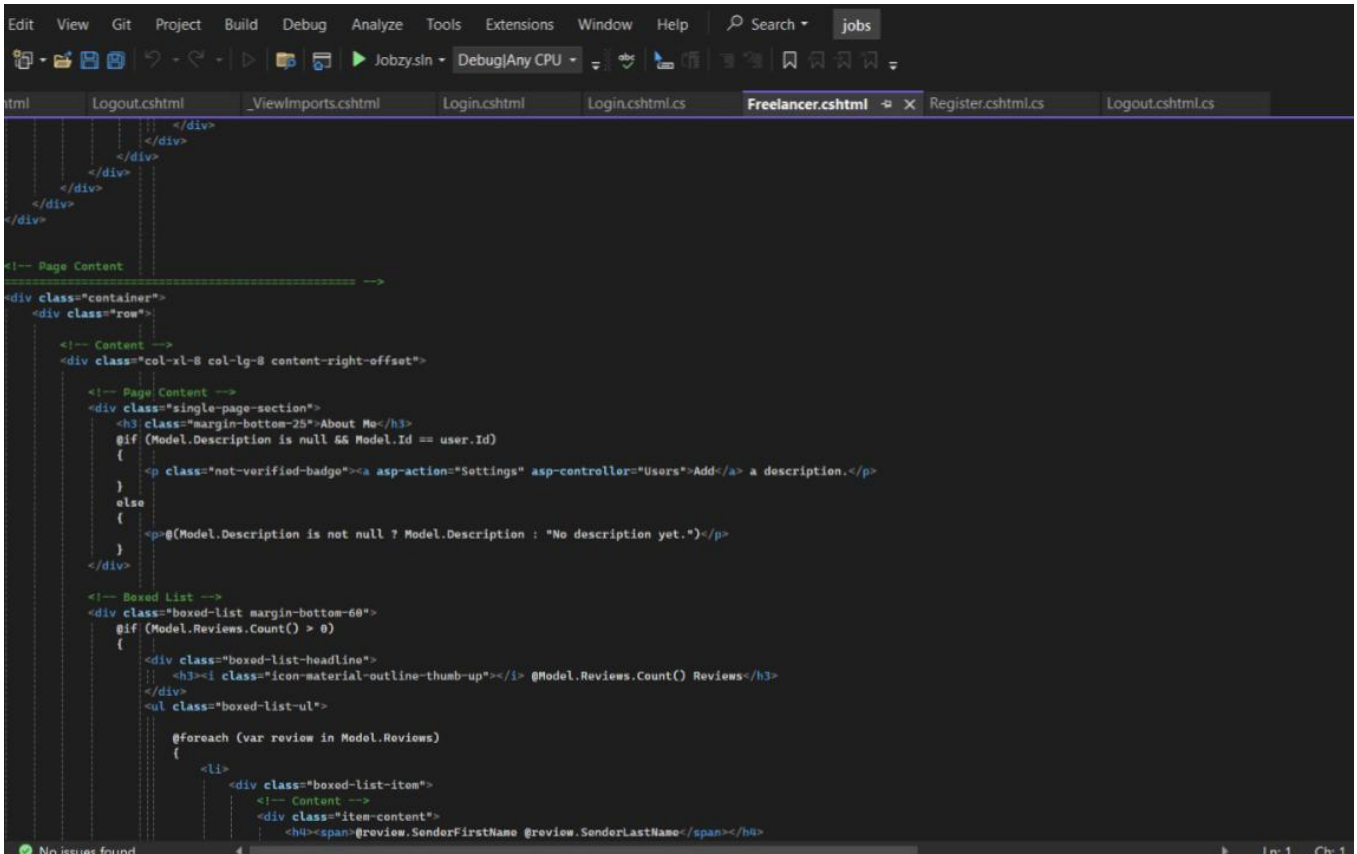
Be the first one to leave a review.

}
@if (Model.Id != user.Id)
{

Leave a Review

}

<!-- Boxed List / End -->
<!-- Boxed List -->
"


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs
<!-- Page Content

<!-- Content -->

<!-- Page Content -->

<h3 class="margin-bottom-25">About Me</h3>
@if (Model.Description is null && Model.Id == user.Id)
{
<a asp-action="Settings" asp-controller="Users">Add a description.
}
else
{
@(Model.Description is not null ? Model.Description : "No description yet.")
}

<!-- Boxed List -->

@if (Model.Reviews.Count() > 0)
{

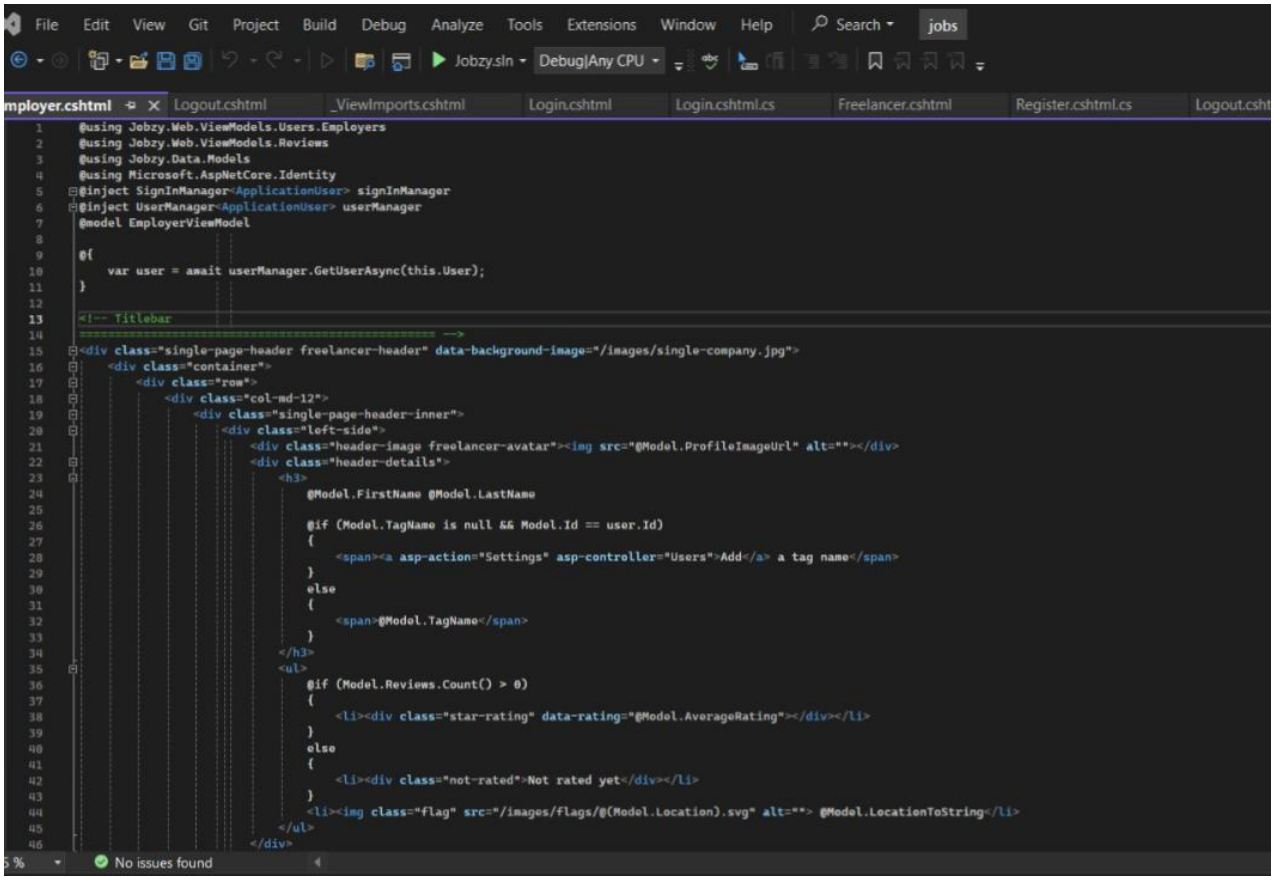
<h3> @Model.Reviews.Count() Reviews</h3>

<ul class="boxed-list-ul">
@foreach (var review in Model.Reviews)
{

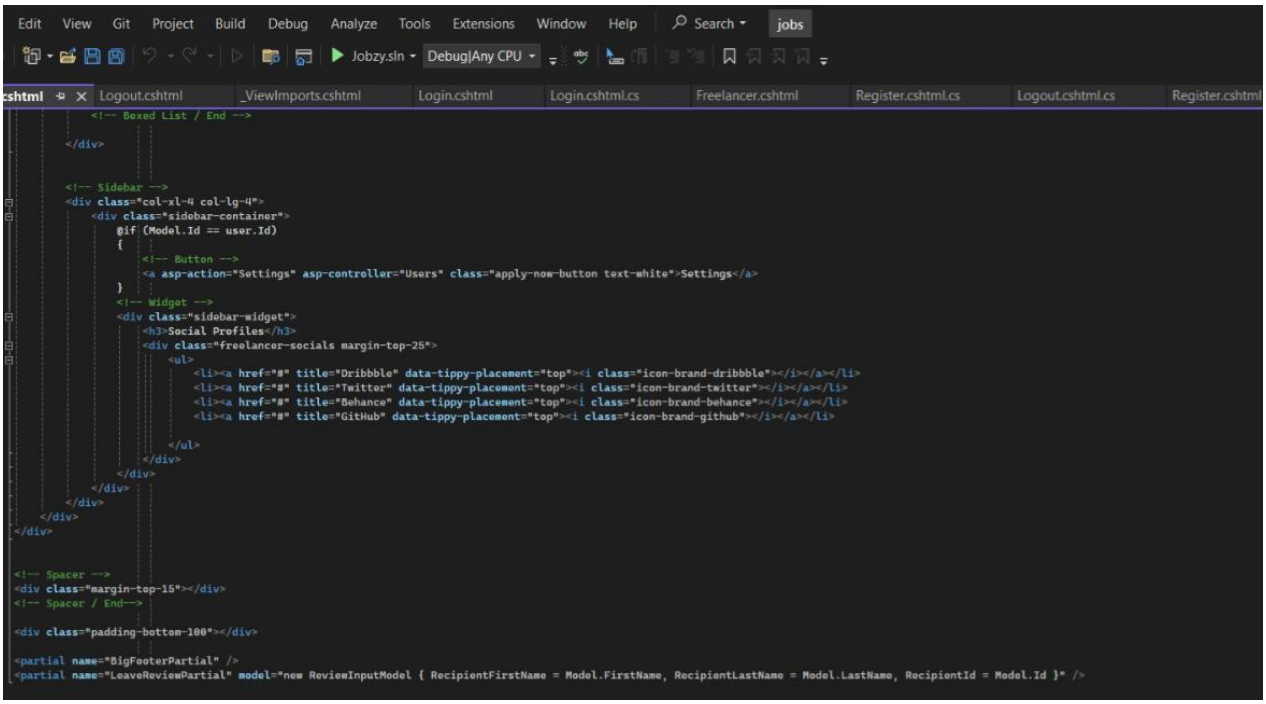
<!-- Content -->

<h4>@review.SenderFirstName @review.SenderLastName</h4>
No issues found

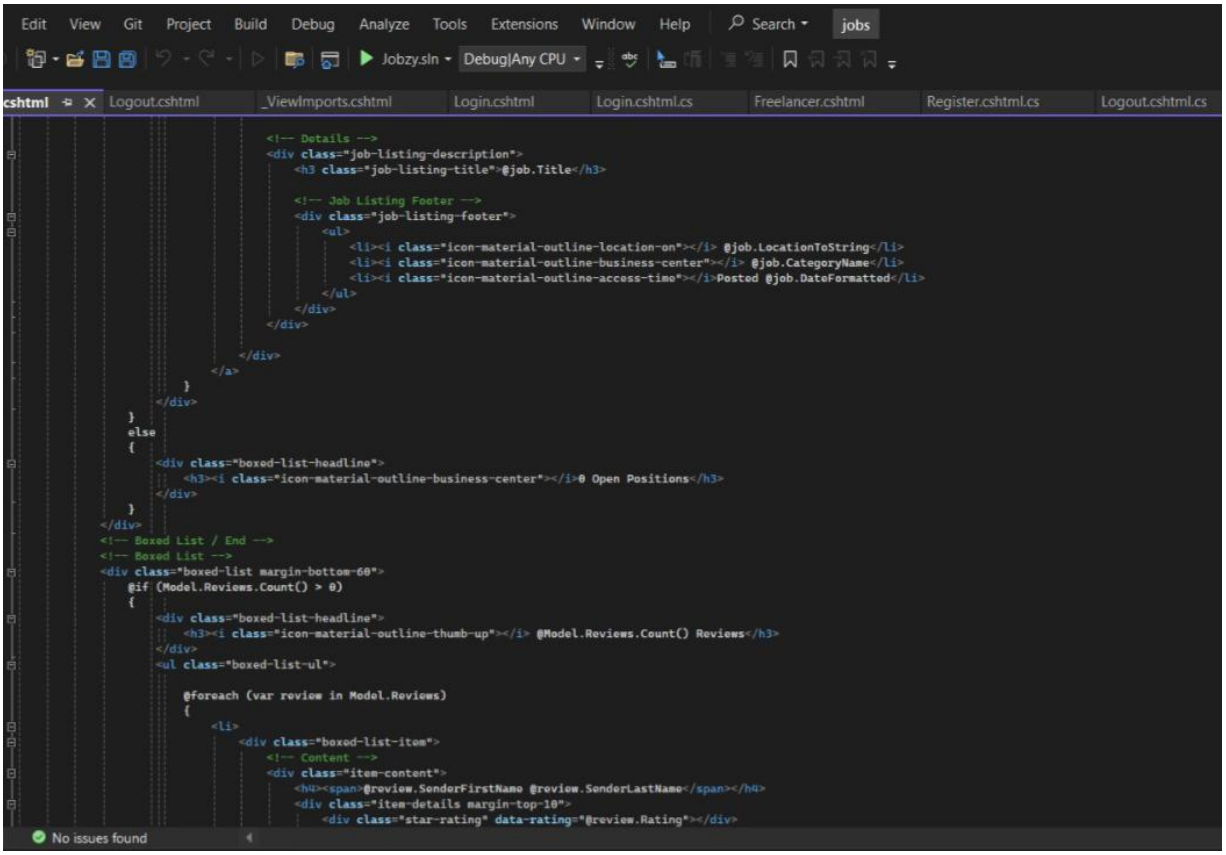
Employer profile



```
@using Jobzy.Web.ViewModels.Users.Employers
@using Jobzy.Web.ViewModels.Reviews
@using Jobzy.Data.Models
@using Microsoft.AspNetCore.Identity
@inject SignInManager<ApplicationUser> signInManager
@inject UserManager<ApplicationUser> userManager
@model EmployerViewModel

@{
    var user = await userManager.GetUserAsync(this.User);
}

<!-- Titlebar
============================================= -->
<div class="single-page-header freelancer-header" data-background-image="/images/single-company.jpg">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <div class="single-page-header-inner">
                    <div class="left-side">
                        <div class="header-image freelancer-avatar"><img src="@Model.ProfileImageUrl" alt=""></div>
                        <div class="header-details">
                            <h3>
                                @Model.FirstName @Model.LastName

                                @if (Model.TagName is null && Model.Id == user.Id)
                                {
                                    <span><a asp-action="Settings" asp-controller="Users">Add</a> a tag name</span>
                                }
                                else
                                {
                                    <span>@Model.TagName</span>
                                }
                            </h3>
                            <ul>
                                @if (Model.Reviews.Count() > 0)
                                {
                                    <li><div class="star-rating" data-rating="@Model.AverageRating"></div></li>
                                }
                                else
                                {
                                    <li><div class="not-rated">Not rated yet</div></li>
                                }
                                <li><img class="flag" src="/images/flags/@(Model.Location).svg" alt=""> @Model.LocationToString</li>
                            </ul>
                        </div>
```



```
            <!-- Boxed List / End -->

        </div>


        <!-- Sidebar -->
        <div class="col-xl-4 col-lg-4">
            <div class="sidebar-container">
                @if (Model.Id == user.Id)
                {
                    <!-- Button -->
                    <a asp-action="Settings" asp-controller="Users" class="apply-now-button text-white">Settings</a>
                }
                <!-- Widget -->
                <div class="sidebar-widget">
                    <h3>Social Profiles</h3>
                    <div class="freelancer-socials margin-top-25">
                        <ul>
                            <li><a href="#" title="Dribbble" data-tippy-placement="top"><i class="icon-brand-dribbble"></i></a></li>
                            <li><a href="#" title="Twitter" data-tippy-placement="top"><i class="icon-brand-twitter"></i></a></li>
                            <li><a href="#" title="Behance" data-tippy-placement="top"><i class="icon-brand-behance"></i></a></li>
                            <li><a href="#" title="GitHub" data-tippy-placement="top"><i class="icon-brand-github"></i></a></li>

                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>


<!-- Spacer -->
<div class="margin-top-15"></div>
<!-- Spacer / End-->

<div class="padding-bottom-100"></div>

<partial name="BigFooterPartial" />
<partial name="LeaveReviewPartial" model="new ReviewInputModel { RecipientFirstName = Model.FirstName, RecipientLastName = Model.LastName, RecipientId = Model.Id }" />
```


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs
<!-- Details -->

<h3 class="job-listing-title">@job.Title</h3>
<!-- Job Listing Footer -->

 @job.LocationToString
 @job.CategoryName
Posted @job.DateFormatted

}

}
else
{

<h3>0 Open Positions</h3>

}

<!-- Boxed List / End -->
<!-- Boxed List -->

@if (Model.Reviews.Count() > 0)
{

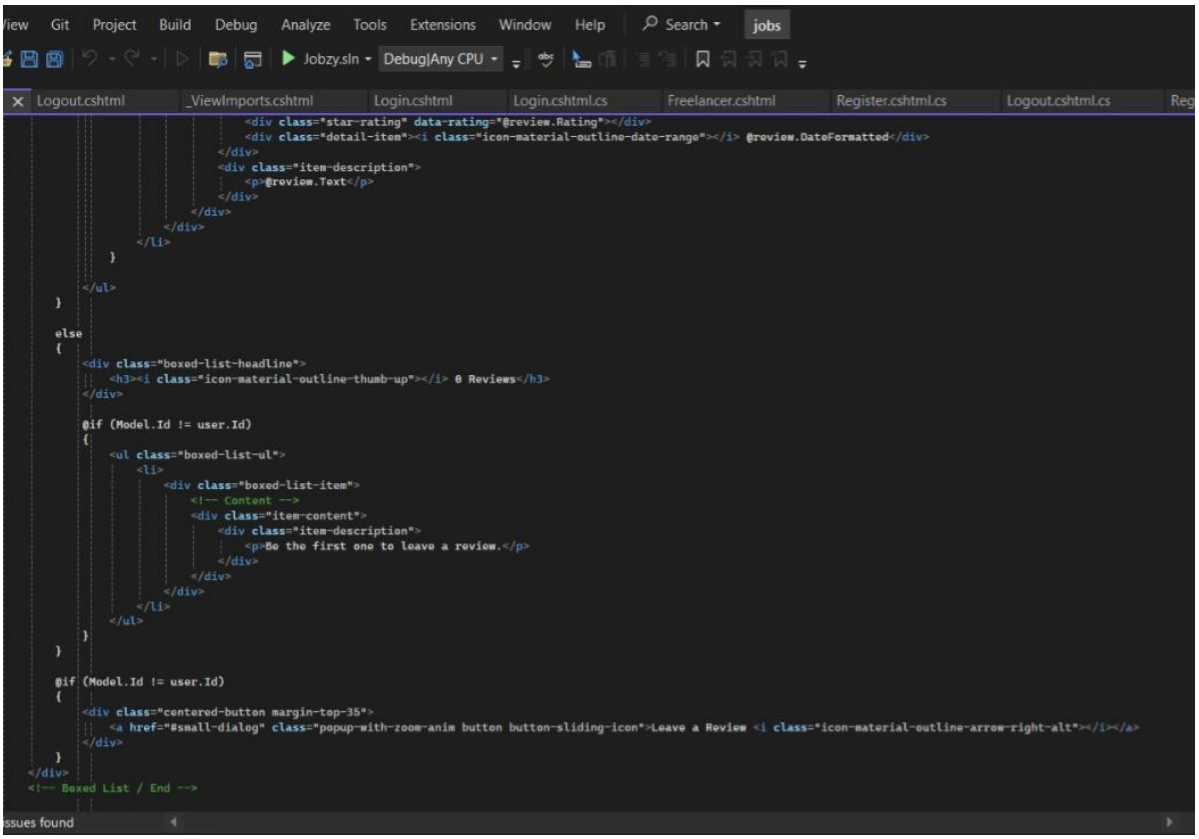
<h3> @Model.Reviews.Count() Reviews</h3>

<ul class="boxed-list-ul">
@foreach (var review in Model.Reviews)
{

<!-- Content -->

<h4>@review.SenderFirstName @review.SenderLastName</h4>

No issues found


Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs

 @review.DateFormatted

@review.Text

}
else
{

<h3> 0 Reviews</h3>

@if (Model.Id != user.Id)
{
<ul class="boxed-list-ul">

<!-- Content -->

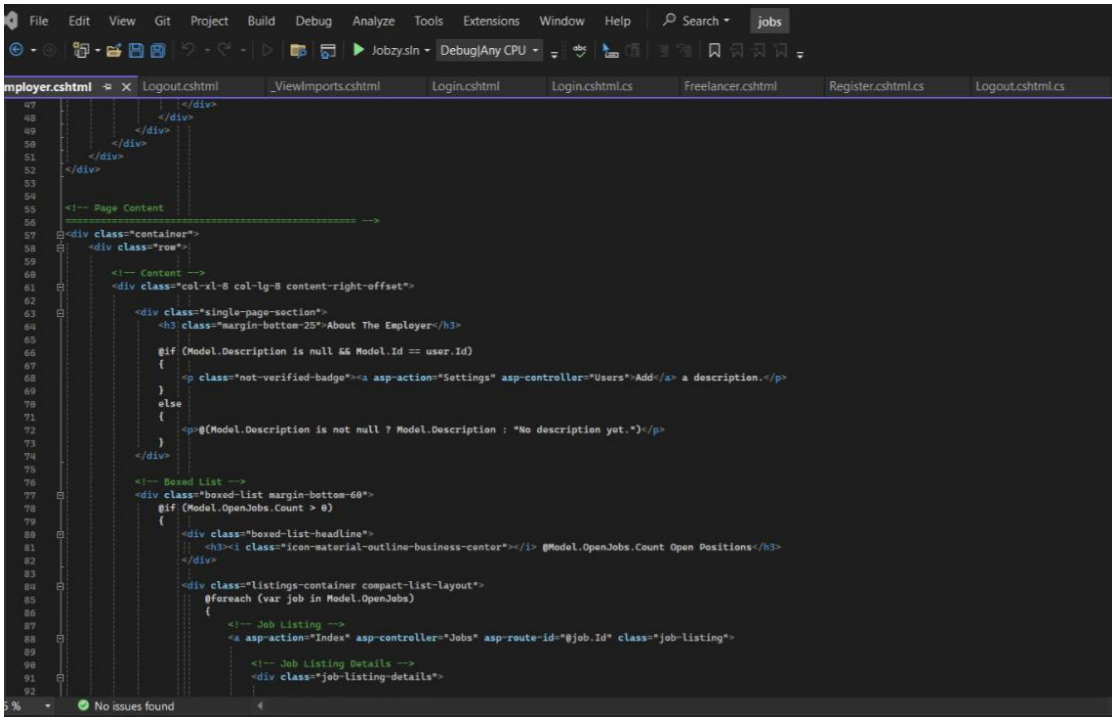
Be the first one to leave a review.

}
@if (Model.Id != user.Id)
{

Leave a Review

}

<!-- Boxed List / End -->
ssues found


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
jobs
Jobzy.sln
Debug|Any CPU
mployer.cshtml
Logout.cshtml
_ViewImports.cshtml
Login.cshtml
Login.cshtml.cs
Freelancer.cshtml
Register.cshtml.cs
Logout.cshtml.cs
<!-- Page Content

<!-- Content -->

<h3 class="margin-bottom-25">About The Employer</h3>
@if (Model.Description is null && Model.Id == user.Id)
<a asp-action="Settings" asp-controller="Users">Add a description.
else
@(Model.Description is not null ? Model.Description : "No description yet.")
<!-- Boxed List -->

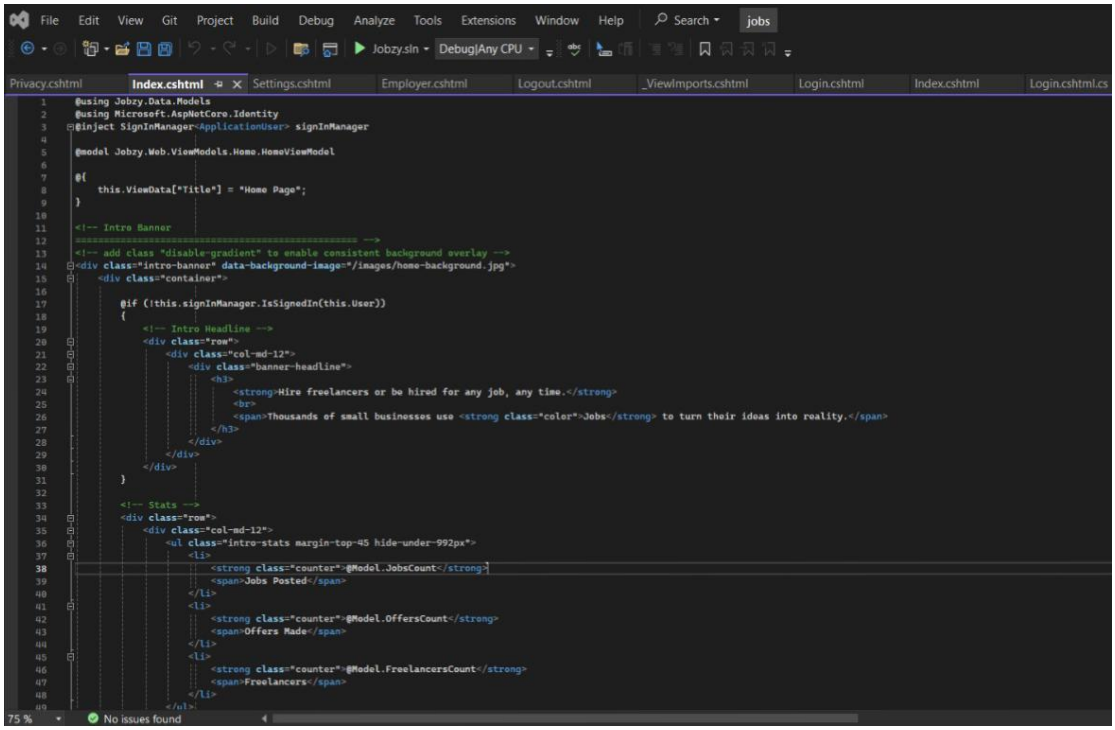
@if (Model.OpenJobs.Count > 0)

<h3> @Model.OpenJobs.Count Open Positions</h3>

@foreach (var job in Model.OpenJobs)
<!-- Job Listing -->
<a asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing">
<!-- Job Listing Details -->

No issues found

Sitting (Edit profile , Upload image& Delete image)


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Index.cshtml Login.cshtml.cs
@using Jobzy.Data.Models
@using Microsoft.AspNetCore.Identity
@inject SignInManager<ApplicationUser> signInManager
@model Jobzy.Web.ViewModels.Home.HomeViewModel
@{
this.ViewData["Title"] = "Home Page";
}
<!-- Intro Banner
<!-- add class "disable-gradient" to enable consistent background overlay -->

@if (!this.signInManager.IsSignedIn(this.User))
{
<!-- Intro Headline -->

<h3>
Hire freelancers or be hired for any job, any time.

Thousands of small businesses use Jobs to turn their ideas into reality.
</h3>

}
<!-- Stats -->

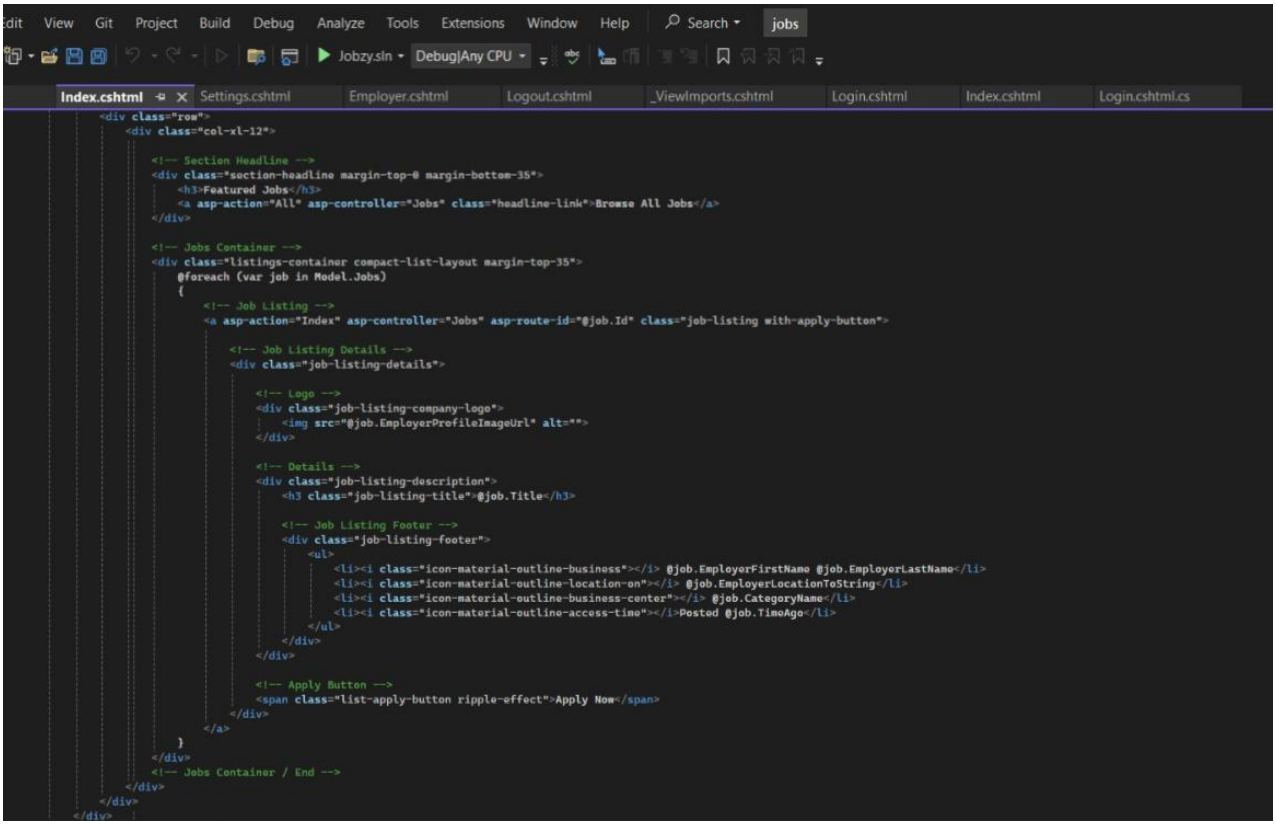
<ul class="intro-stats margin-top-45 hide-under-992px">

@Model.JobsCount
Jobs Posted

@Model.OffersCount
Offers Made

@Model.FreelancersCount
Freelancers

75 %
No issues found


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Index.cshtml Login.cshtml.cs

<!-- Section Headline -->

<h3>Featured Jobs</h3>
<a asp-action="All" asp-controller="Jobs" class="headline-link">Browse All Jobs

<!-- Jobs Container -->

@foreach (var job in Model.Jobs)
{
<!-- Job Listing -->
<a asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing with-apply-button">
<!-- Job Listing Details -->

<!-- Logo -->

<img src="@job.EmployerProfileImageUrl" alt="">

<!-- Details -->

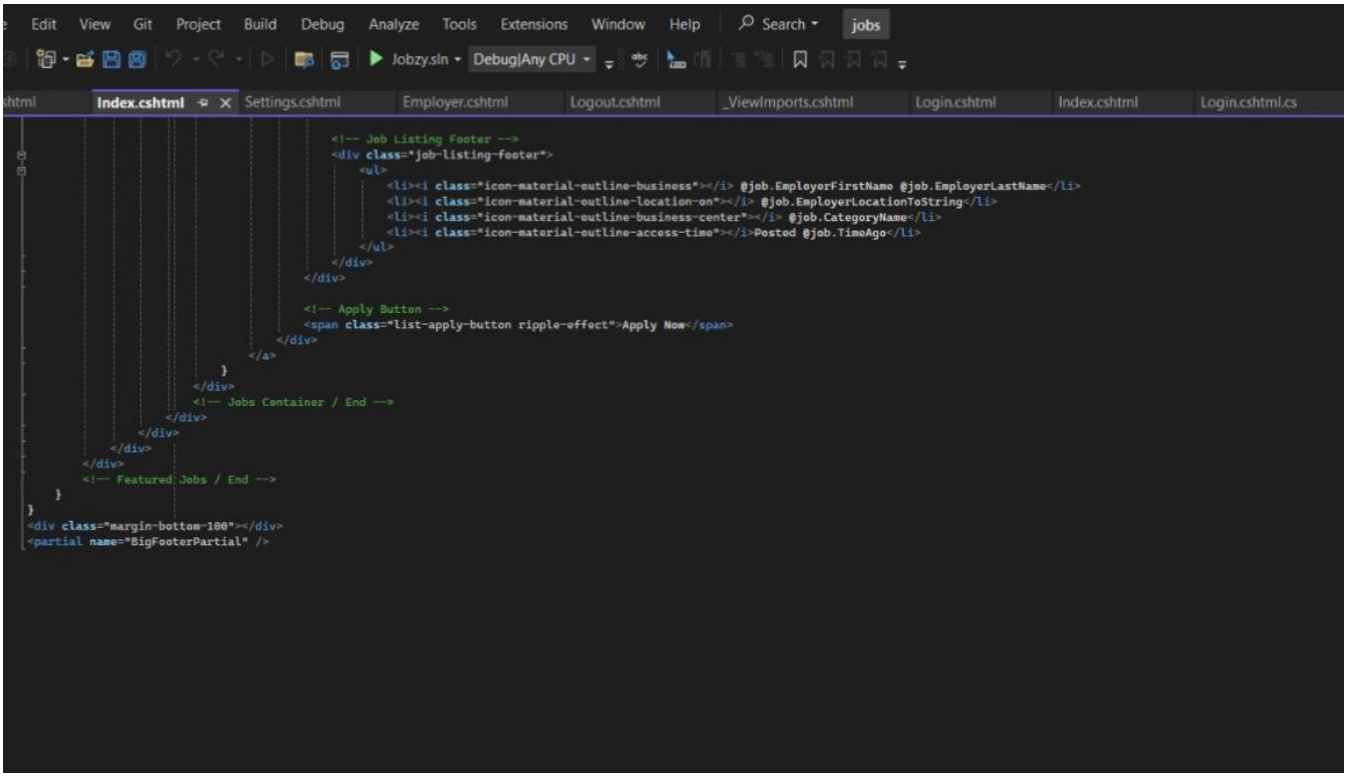
<h3 class="job-listing-title">@job.Title</h3>
<!-- Job Listing Footer -->

 @job.EmployerFirstName @job.EmployerLastName
 @job.EmployerLocationToString
 @job.CategoryName
Posted @job.TimeAgo

<!-- Apply Button -->
Apply Now

}

<!-- Jobs Container / End -->


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Index.cshtml Login.cshtml.cs
<!-- Job Listing Footer -->

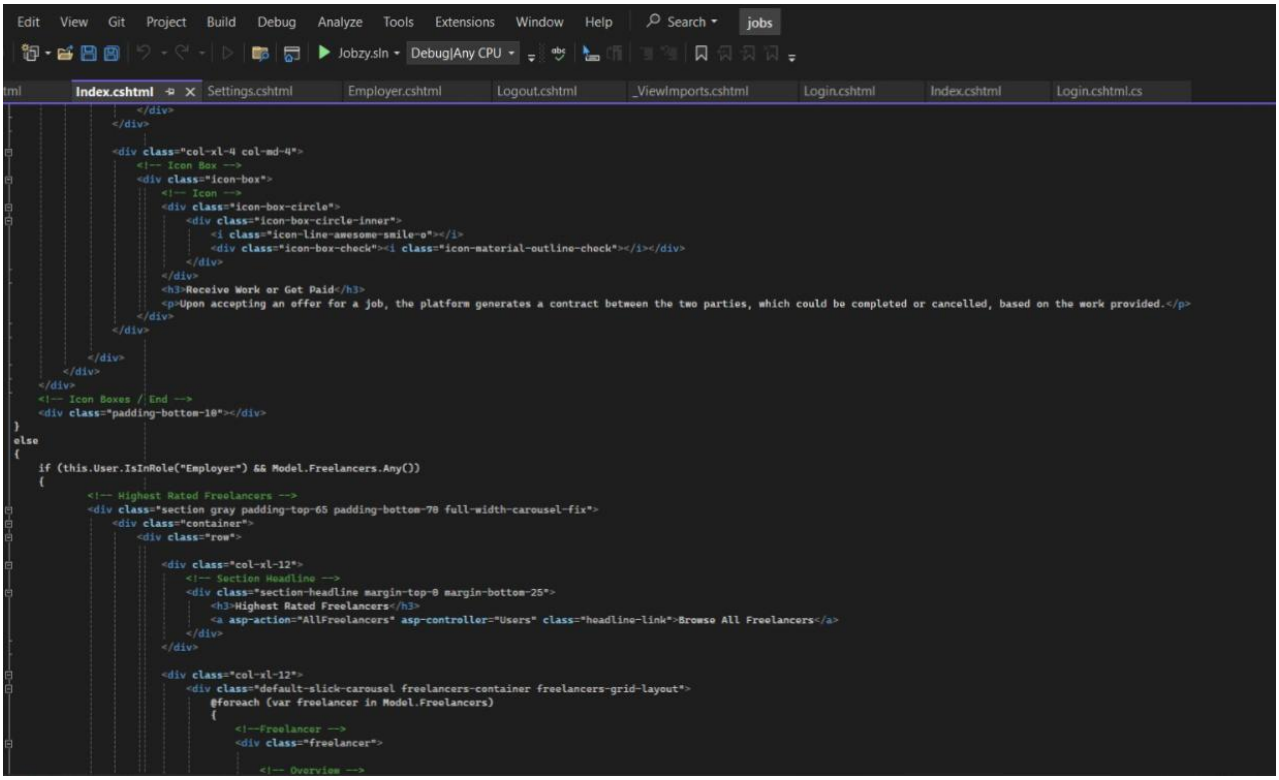
 @job.EmployerFirstNamo @job.EmployerLastName
 @job.EmployerLocationToString
 @job.CategoryName
Posted @job.TimeAgo

<!-- Apply Button -->
Apply Now

}

<!-- Jobs Container / End -->
<!-- Featured Jobs / End -->
}
}

<partial name="BigFooterPartial" />


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Index.cshtml Login.cshtml.cs

<!-- Icon Box -->

<!-- Icon -->

<h3>Receive Work or Get Paid</h3>
Upon accepting an offer for a job, the platform generates a contract between the two parties, which could be completed or cancelled, based on the work provided.

<!-- Icon Boxes / End -->

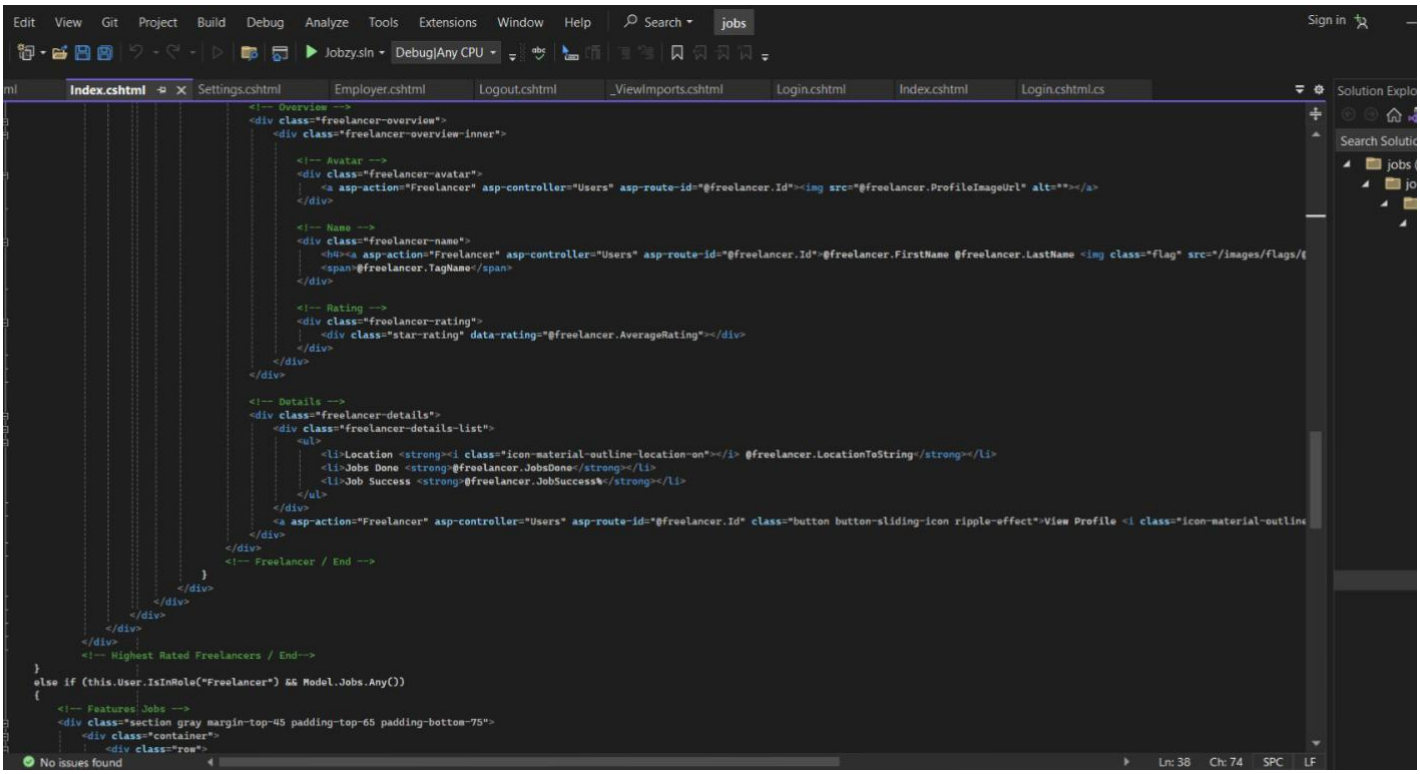
}
else
{
if (this.User.IsInRole("Employer") && Model.Freelancers.Any())
{
<!-- Highest Rated Freelancers -->

<!-- Section Headline -->

<h3>Highest Rated Freelancers</h3>
<a asp-action="AllFreelancers" asp-controller="Users" class="headline-link">Browse All Freelancers

@foreach (var freelancer in Model.Freelancers)
{
<!--Freelancer -->

<!-- Overview -->


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs Sign in
Jobzy.sln Debug|Any CPU
Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Index.cshtml Login.cshtml.cs
Solution Explo
Search Solutio
jobs
<!-- Overview -->

<!-- Avatar -->

<a asp-action="Freelancer" asp-controller="Users" asp-route-id="@freelancer.Id"><img src="@freelancer.ProfileImageUrl" alt="">

<!-- Name -->

<h4><a asp-action="Freelancer" asp-controller="Users" asp-route-id="@freelancer.Id">@freelancer.FirstName @freelancer.LastName <img class="flag" src="/images/flags/
@freelancer.TagName

<!-- Rating -->

<!-- Details -->

Location @freelancer.LocationToString
Jobs Done @freelancer.JobsDone
Job Success @freelancer.JobSuccess%

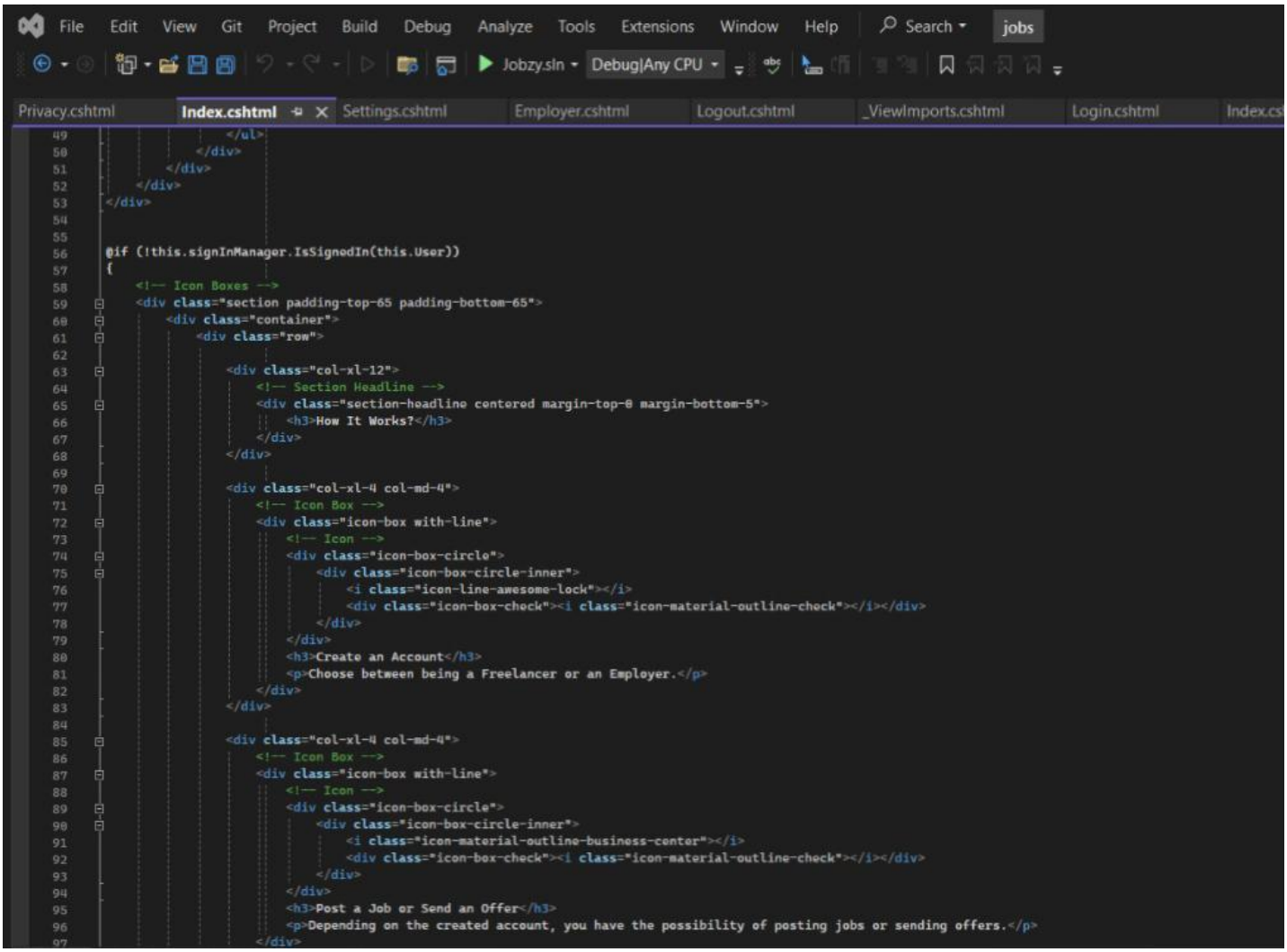
<a asp-action="Freelancer" asp-controller="Users" asp-route-id="@freelancer.Id" class="button button-sliding-icon ripple-effect">View Profile <i class="icon-material-outlin

<!-- Freelancer / End -->
}

<!-- Highest Rated Freelancers / End-->
}
else if (this.User.IsInRole("Freelancer") && Model.Jobs.Any())
{
<!-- Features Jobs -->

No issues found
Ln: 38 Ch: 74 SPC LF

Home


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml
@if (!this.signInManager.IsSignedIn(this.User))
<!-- Icon Boxes -->

<!-- Section Headline -->

<h3>How It Works?</h3>

<!-- Icon Box -->

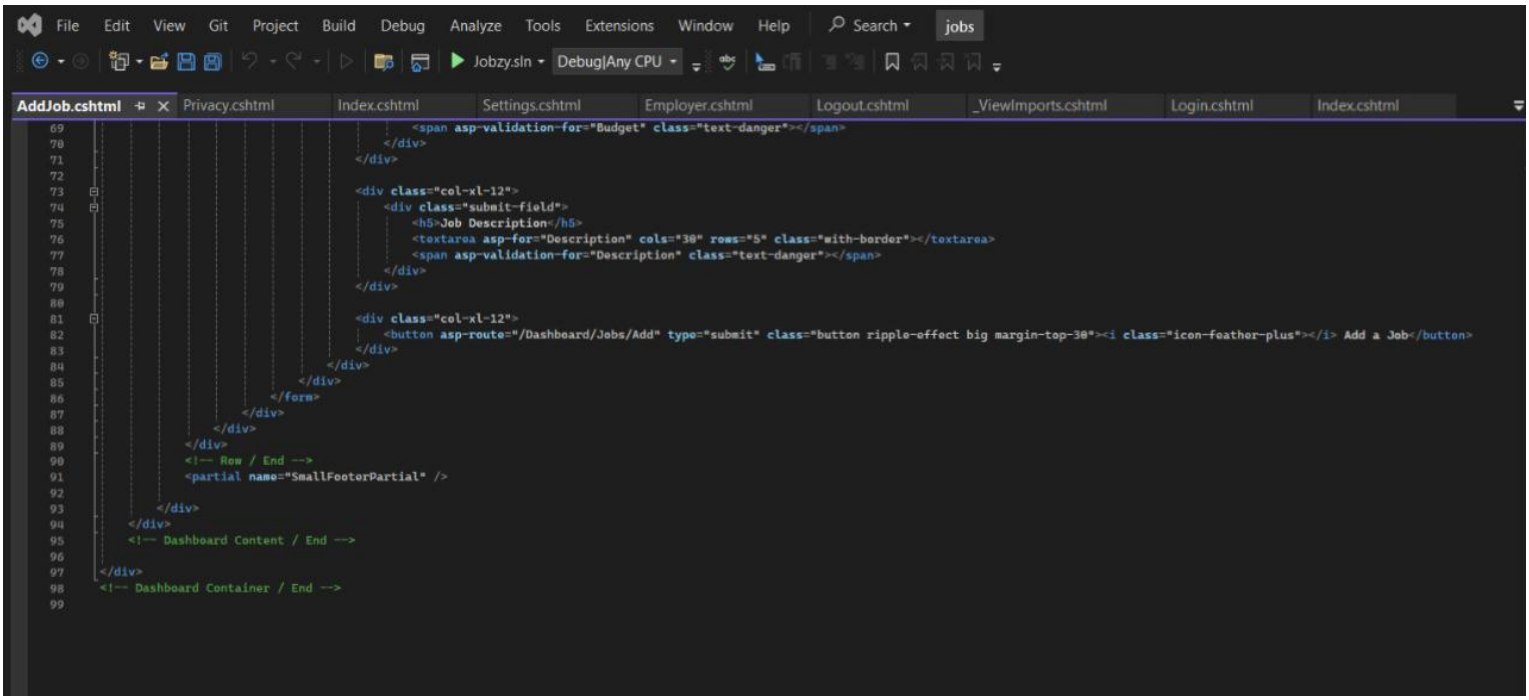
<!-- Icon -->

<h3>Create an Account</h3>
Choose between being a Freelancer or an Employer.

<!-- Icon Box -->

<!-- Icon -->

<h3>Post a Job or Send an Offer</h3>
Depending on the created account, you have the possibility of posting jobs or sending offers.


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
jobs
Jobzy.sln
Debug|Any CPU
AddJob.cshtml
Privacy.cshtml
Index.cshtml
Settings.cshtml
Employer.cshtml
Logout.cshtml
_ViewImports.cshtml
Login.cshtml
Index.cshtml

<h5>Job Description</h5>
<textarea asp-for="Description" cols="30" rows="5" class="with-border"></textarea>

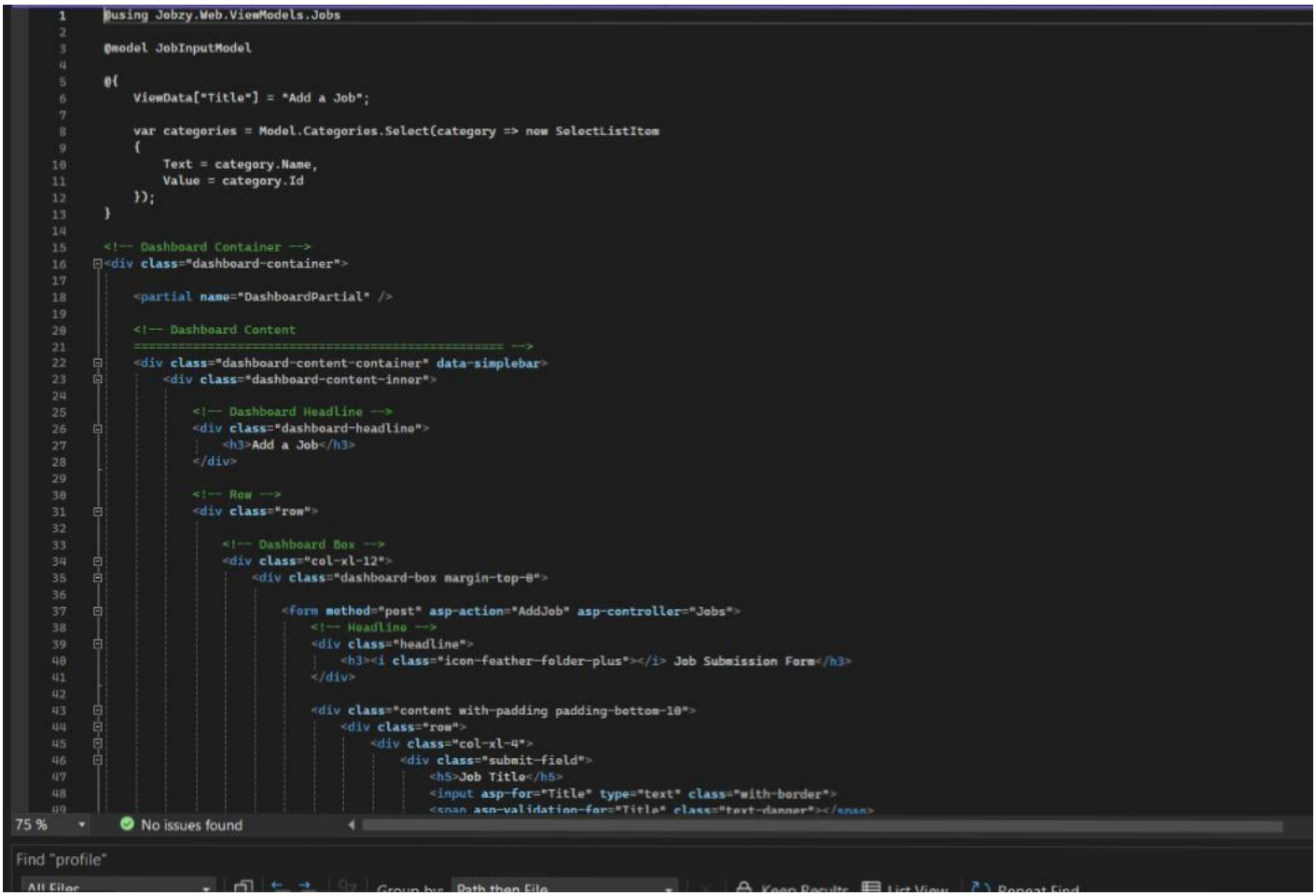
<button asp-route="/Dashboard/Jobs/Add" type="submit" class="button ripple-effect big margin-top-30"> Add a Job</button>

</form>

<!-- Row / End -->
<partial name="SmallFooterPartial" />

<!-- Dashboard Content / End -->

<!-- Dashboard Container / End -->


@using Jobzy.Web.ViewModels.Jobs
@model JobInputModel
@{
ViewData["Title"] = "Add a Job";
var categories = Model.Categories.Select(category => new SelectListItem
{
Text = category.Name,
Value = category.Id
});
}
<!-- Dashboard Container -->

<partial name="DashboardPartial" />
<!-- Dashboard Content
<div class="dashboard-content-container" data-simplebar>

<!-- Dashboard Headline -->

<h3>Add a Job</h3>

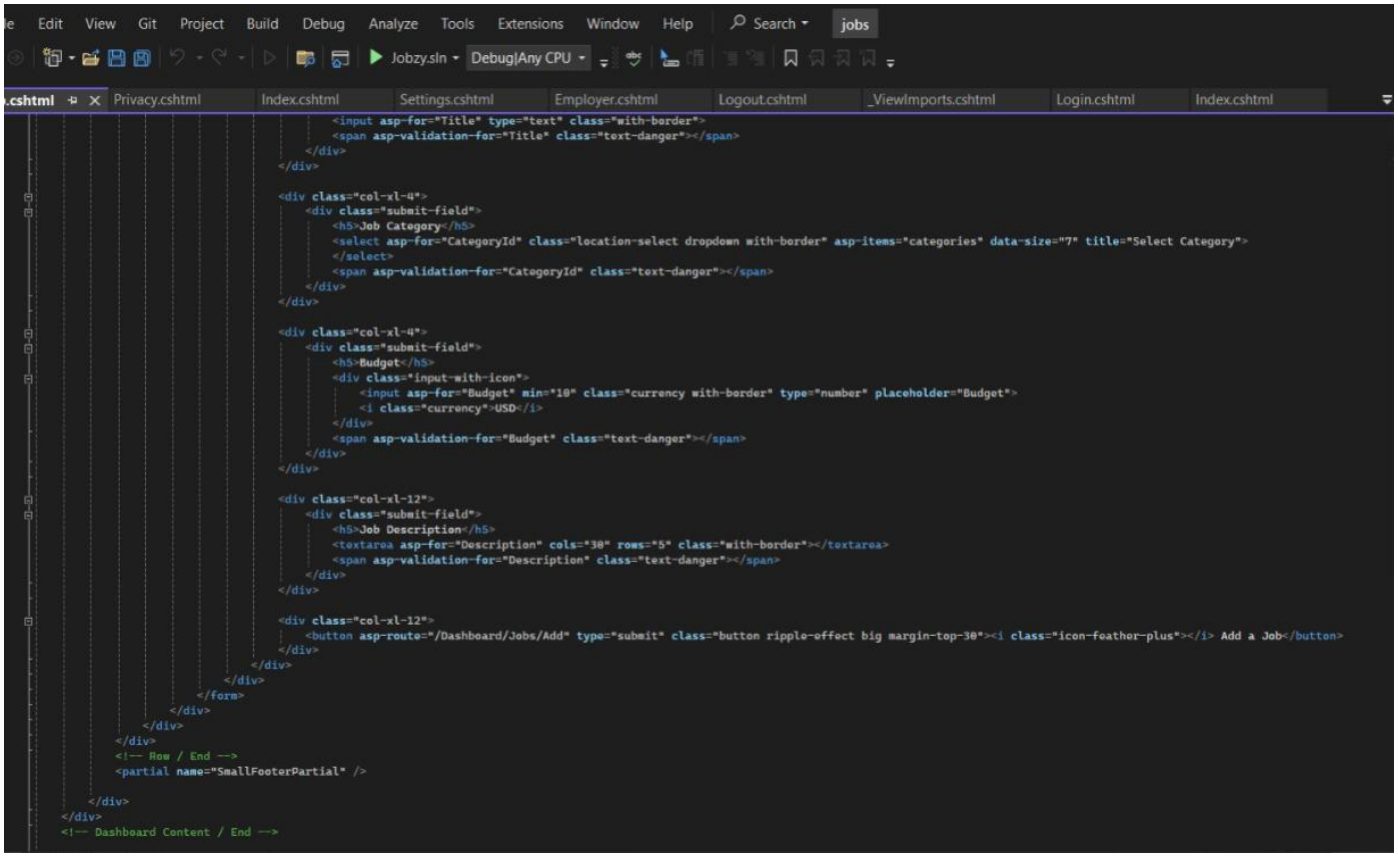
<!-- Row -->

<!-- Dashboard Box -->

<form method="post" asp-action="AddJob" asp-controller="Jobs">
<!-- Headline -->

<h3> Job Submission Form</h3>

<h5>Job Title</h5>
<input asp-for="Title" type="text" class="with-border">
75 %
No issues found
Find "profile"


Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Index.cshtml
<input asp-for="Title" type="text" class="with-border">

<h5>Job Category</h5>
<select asp-for="CategoryId" class="location-select dropdown with-border" asp-items="categories" data-size="7" title="Select Category">
</select>

<h5>Budget</h5>

<input asp-for="Budget" min="10" class="currency with-border" type="number" placeholder="Budget">
USD

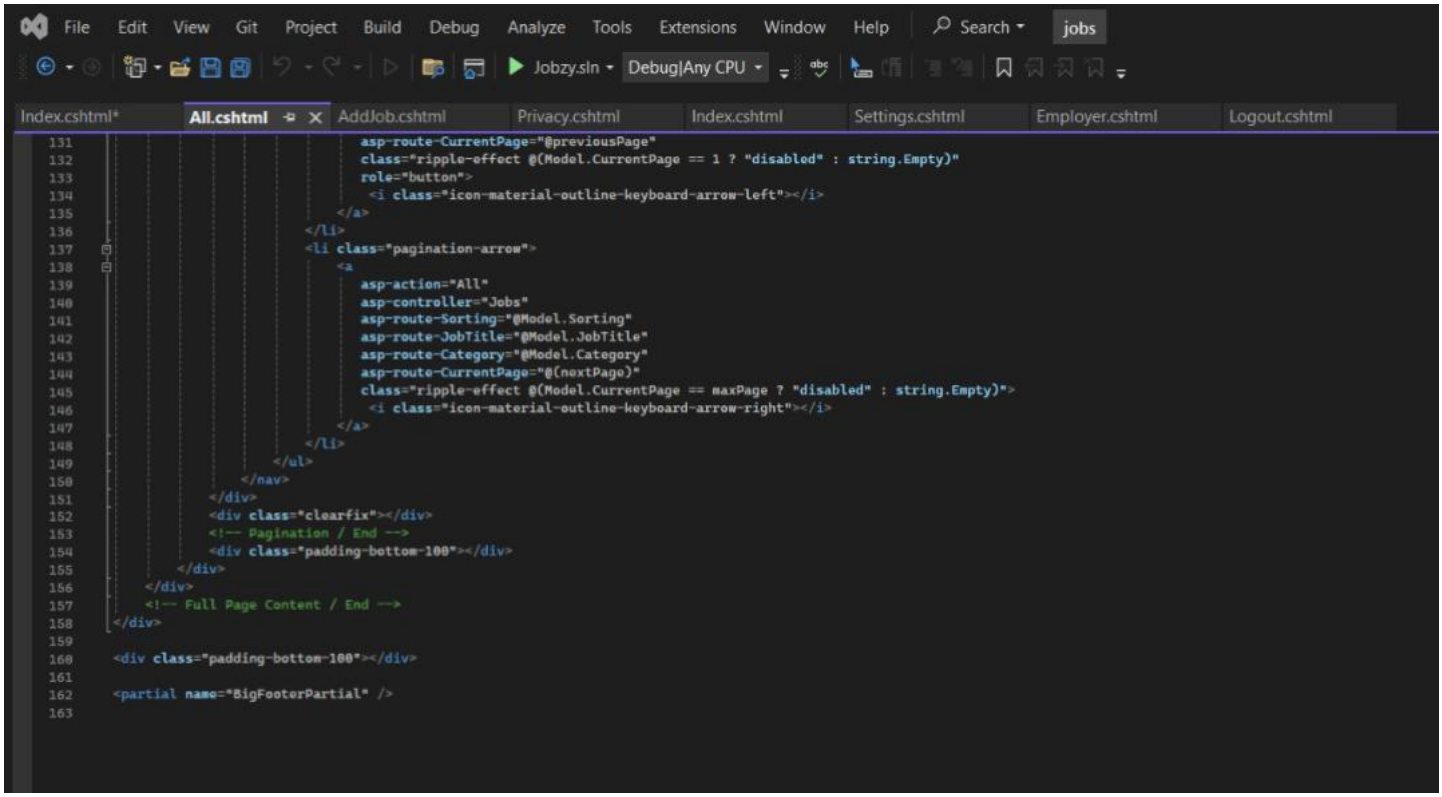
<h5>Job Description</h5>
<textarea asp-for="Description" cols="30" rows="5" class="with-border"></textarea>

<button asp-route="/Dashboard/Jobs/Add" type="submit" class="button ripple-effect big margin-top-30"> Add a Job</button>

</form>

<!-- Row / End -->
<partial name="SmallFooterPartial" />

<!-- Dashboard Content / End -->


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml
asp-route-CurrentPage="@previousPage"
class="ripple-effect @(Model.CurrentPage == 1 ? "disabled" : string.Empty)"
role="button">

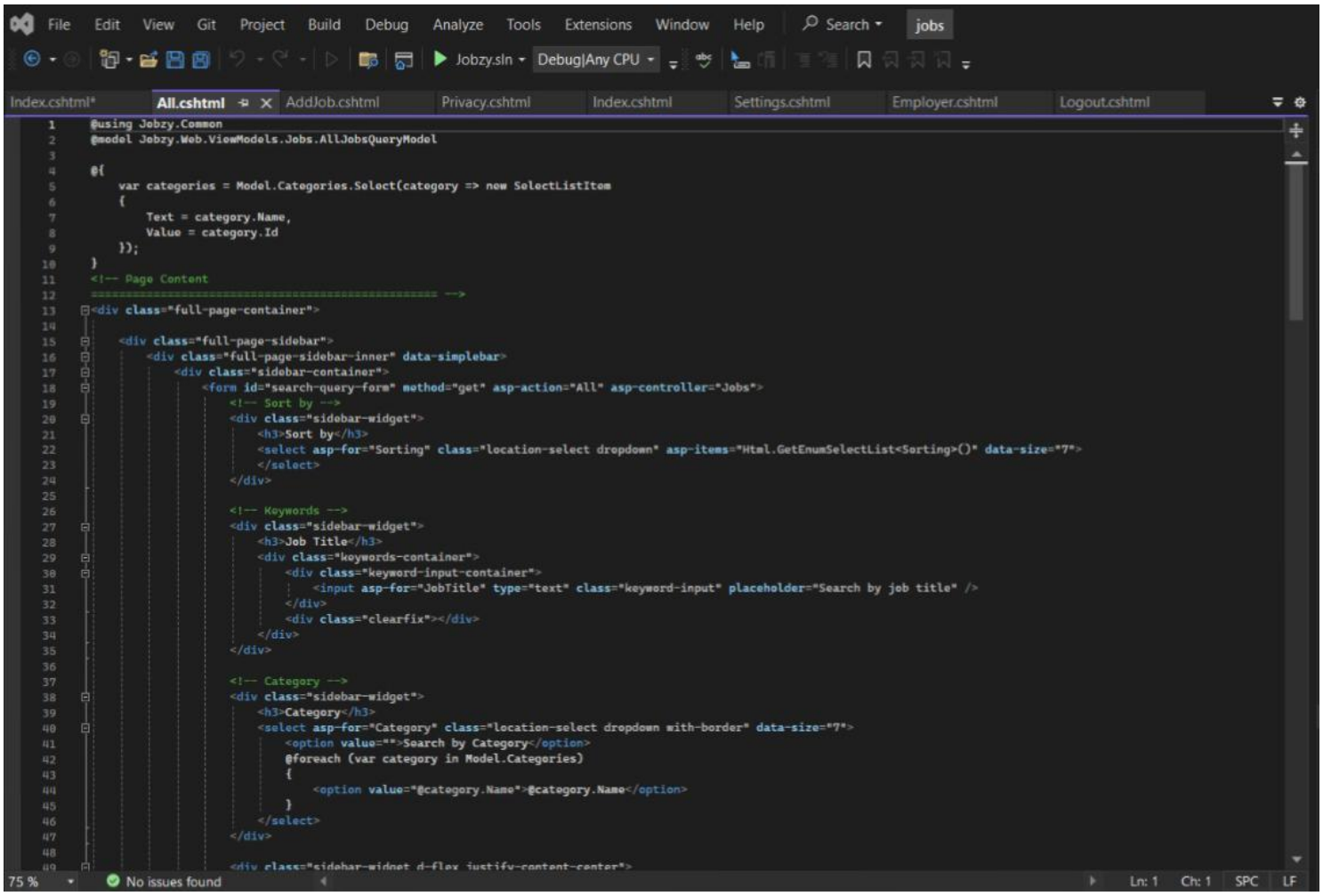
<li class="pagination-arrow">
<a
asp-action="All"
asp-controller="Jobs"
asp-route-Sorting="@Model.Sorting"
asp-route-JobTitle="@Model.JobTitle"
asp-route-Category="@Model.Category"
asp-route-CurrentPage="@(nextPage)"
class="ripple-effect @(Model.CurrentPage == maxPage ? "disabled" : string.Empty)">

</nav>

<!-- Pagination / End -->

<!-- Full Page Content / End -->

<partial name="BigFooterPartial" />


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml
@using Jobzy.Common
@model Jobzy.Web.ViewModels.Jobs.AllJobsQueryModel
@{
var categories = Model.Categories.Select(category => new SelectListItem
{
Text = category.Name,
Value = category.Id
});
}
<!-- Page Content

<div class="full-page-sidebar-inner" data-simplebar>

<form id="search-query-form" method="get" asp-action="All" asp-controller="Jobs">
<!-- Sort by -->

<h3>Sort by</h3>
<select asp-for="Sorting" class="location-select dropdown" asp-items="Html.GetEnumSelectList<Sorting>()" data-size="7">
</select>

<!-- Keywords -->

<h3>Job Title</h3>

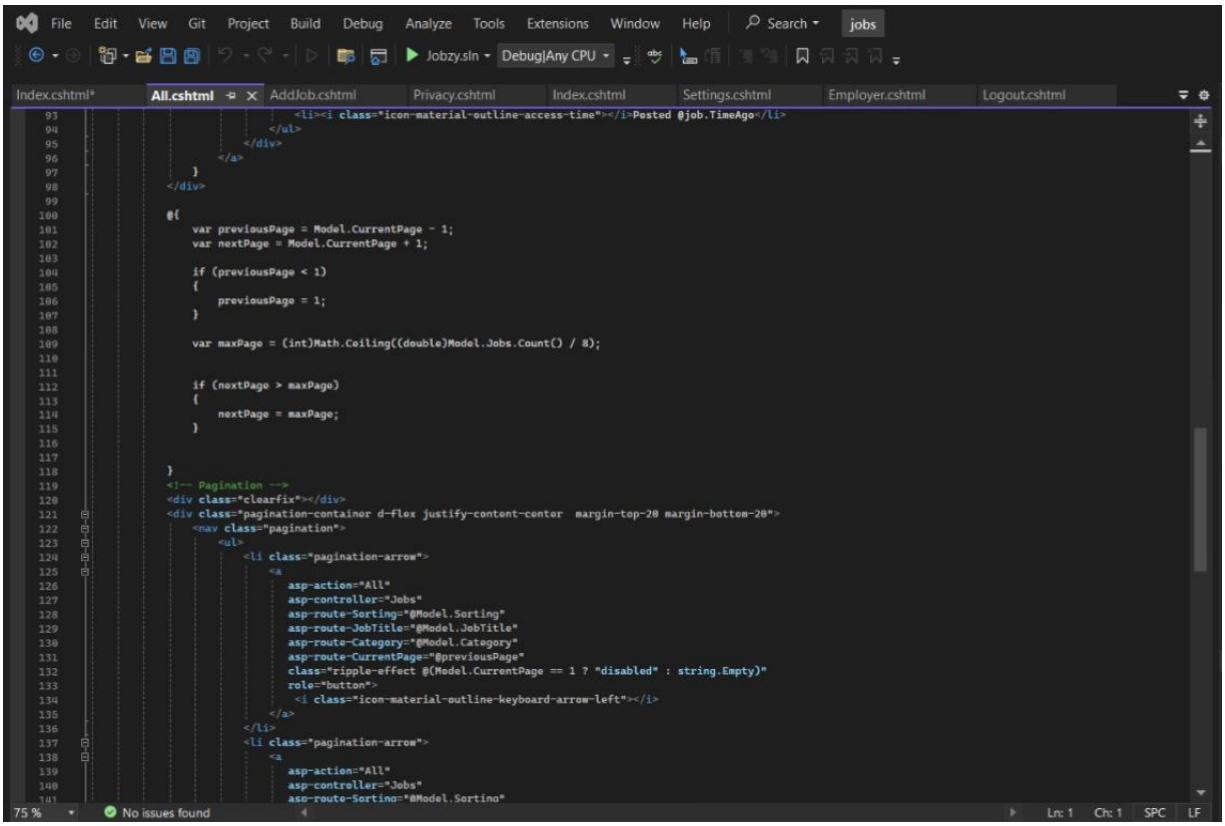
<input asp-for="JobTitle" type="text" class="keyword-input" placeholder="Search by job title" />

<!-- Category -->

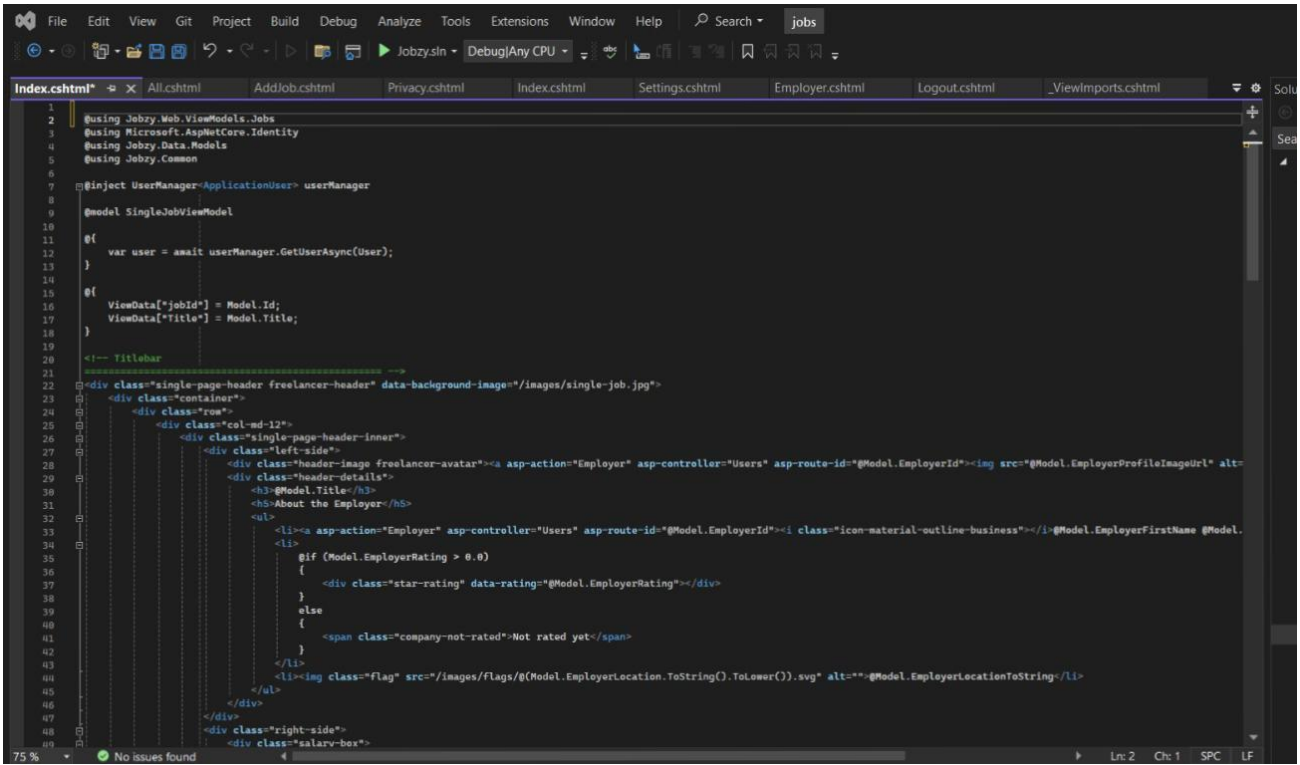
<h3>Category</h3>
<select asp-for="Category" class="location-select dropdown with-border" data-size="7">
<option value="">Search by Category</option>
@foreach (var category in Model.Categories)
{
<option value="@category.Name">@category.Name</option>
}
</select>

75 % No issues found Ln: 1 Ch: 1 SPC LF

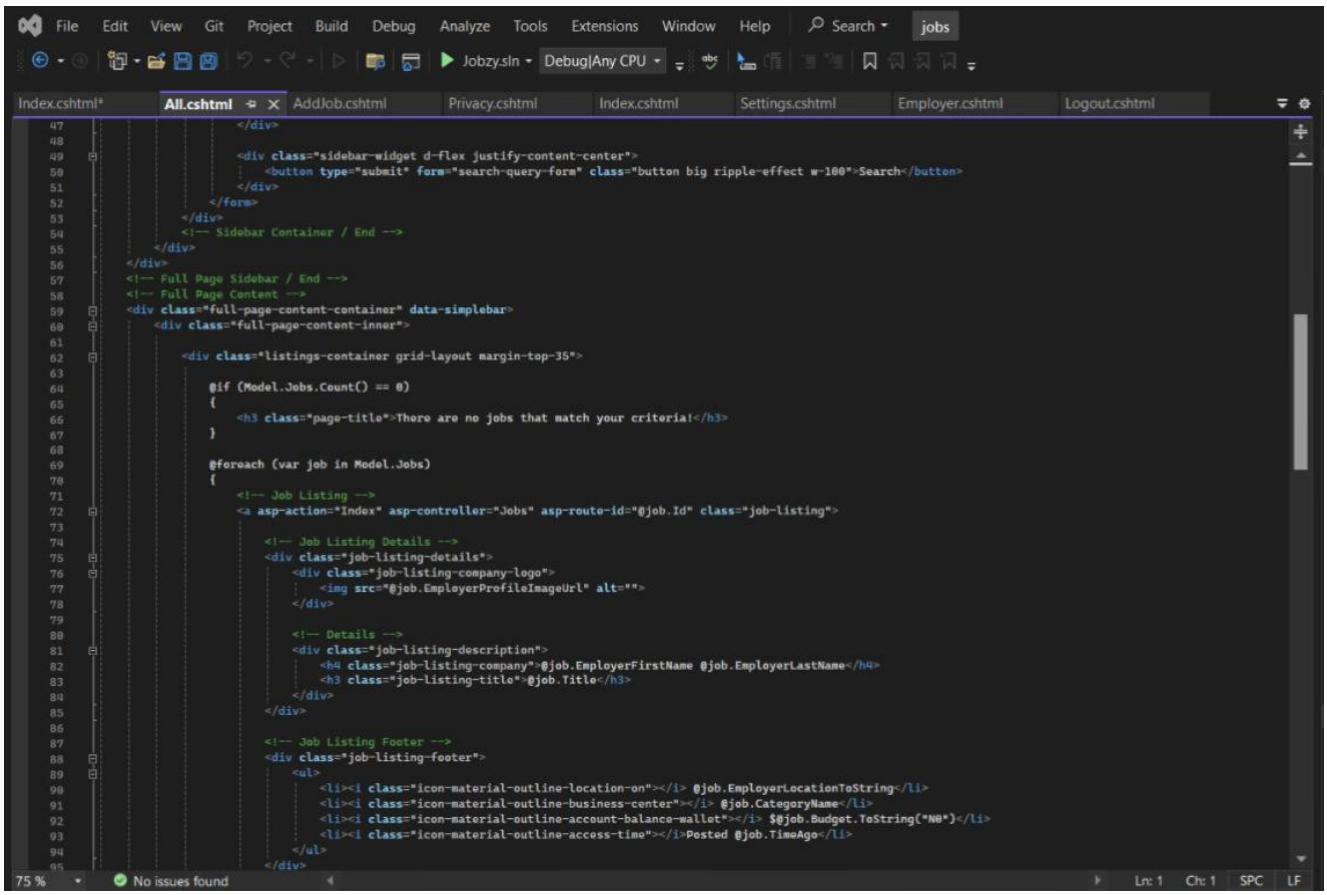
Add job



```
                        <li><i class="icon-material-outline-access-time"></i>Posted @job.TimeAgo</li>
                    </ul>
                </div>
            </a>
        }
    </div>

    @{
        var previousPage = Model.CurrentPage - 1;
        var nextPage = Model.CurrentPage + 1;

        if (previousPage < 1)
        {
            previousPage = 1;
        }

        var maxPage = (int)Math.Ceiling((double)Model.Jobs.Count() / 8);


        if (nextPage > maxPage)
        {
            nextPage = maxPage;
        }


    }
    <!-- Pagination -->
    <div class="clearfix"></div>
    <div class="pagination-container d-flex justify-content-center  margin-top-20 margin-bottom-20">
        <nav class="pagination">
            <ul>
                <li class="pagination-arrow">
                    <a
                       asp-action="All"
                       asp-controller="Jobs"
                       asp-route-Sorting="@Model.Sorting"
                       asp-route-JobTitle="@Model.JobTitle"
                       asp-route-Category="@Model.Category"
                       asp-route-CurrentPage="@previousPage"
                       class="ripple-effect @(Model.CurrentPage == 1 ? "disabled" : string.Empty)"
                       role="button">
                        <i class="icon-material-outline-keyboard-arrow-left"></i>
                    </a>
                </li>
                <li class="pagination-arrow">
                    <a
                       asp-action="All"
                       asp-controller="Jobs"
                       asp-route-Sorting="@Model.Sorting"
```



```
@using Jobzy.Web.ViewModels.Jobs
@using Microsoft.AspNetCore.Identity
@using Jobzy.Data.Models
@using Jobzy.Common

@inject UserManager<ApplicationUser> userManager

@model SingleJobViewModel

@{
    var user = await userManager.GetUserAsync(User);
}

@{
    ViewData["jobId"] = Model.Id;
    ViewData["Title"] = Model.Title;
}

<!-- Titlebar
================================================== -->
<div class="single-page-header freelancer-header" data-background-image="/images/single-job.jpg">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <div class="single-page-header-inner">
                    <div class="left-side">
                        <div class="header-image freelancer-avatar"><a asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"><img src="@Model.EmployerProfileImageUrl" alt=
                        <div class="header-details">
                            <h3>@Model.Title</h3>
                            <h5>About the Employer</h5>
                            <ul>
                                <li><a asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"><i class="icon-material-outline-business"></i>@Model.EmployerFirstName @Model.
                                <li>
                                    @if (Model.EmployerRating > 0.0)
                                    {
                                        <div class="star-rating" data-rating="@Model.EmployerRating"></div>
                                    }
                                    else
                                    {
                                        <span class="company-not-rated">Not rated yet</span>
                                    }
                                </li>
                                <li><img class="flag" src="/images/flags/@(Model.EmployerLocation.ToString().ToLower()).svg" alt="">@Model.EmployerLocationToString</li>
                            </ul>
                        </div>
                    </div>
                    <div class="right-side">
                        <div class="salary-box">
```


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml

<button type="submit" form="search-query-form" class="button big ripple-effect w-100">Search</button>

</form>

<!-- Sidebar Container / End -->

<!-- Full Page Sidebar / End -->
<!-- Full Page Content -->
<div class="full-page-content-container" data-simplebar>

@if (Model.Jobs.Count() == 0)
{
<h3 class="page-title">There are no jobs that match your criteria!</h3>
}
@foreach (var job in Model.Jobs)
{
<!-- Job Listing -->
<a asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing">
<!-- Job Listing Details -->

<img src="@job.EmployerProfileImageUrl" alt="">

<!-- Details -->

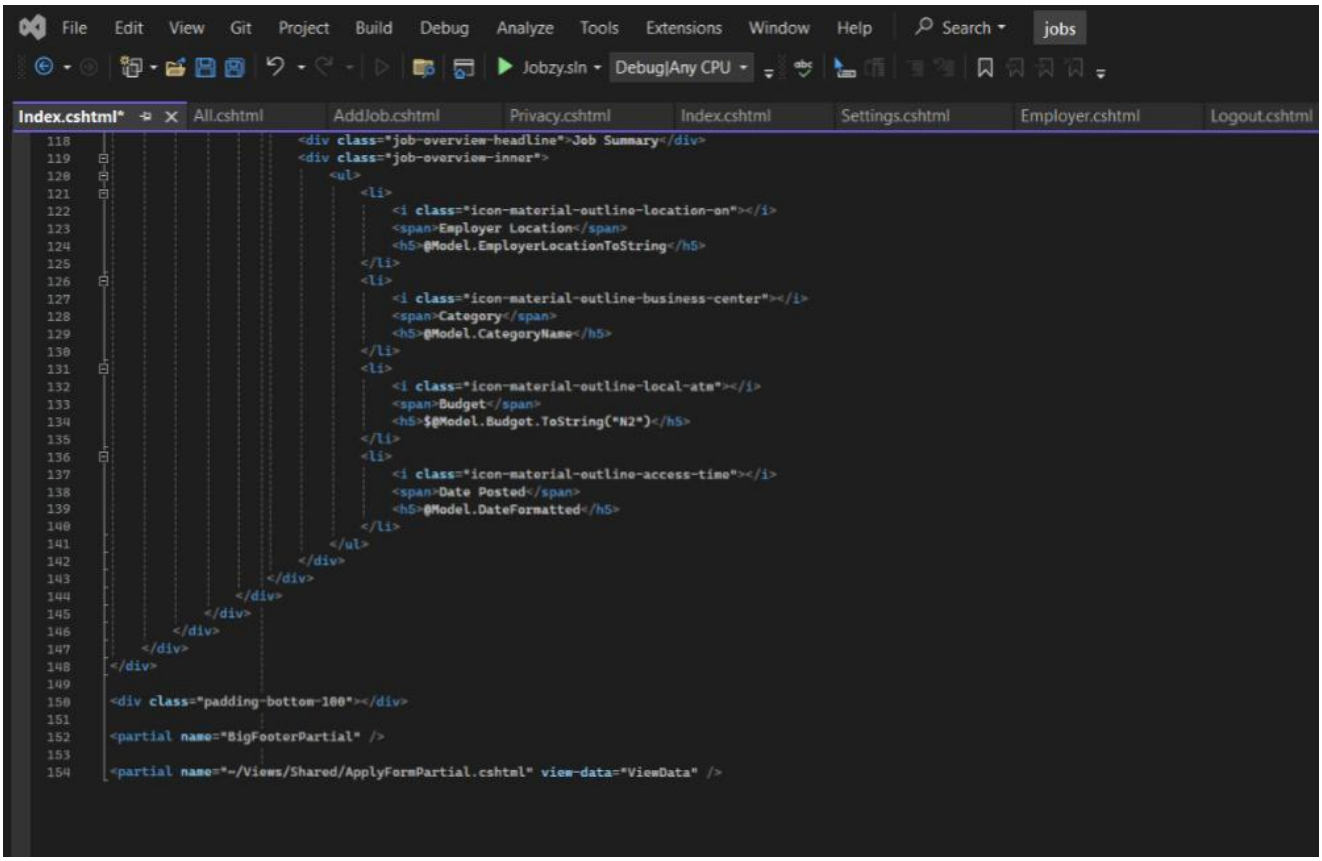
<h4 class="job-listing-company">@job.EmployerFirstName @job.EmployerLastName</h4>
<h3 class="job-listing-title">@job.Title</h3>

<!-- Job Listing Footer -->

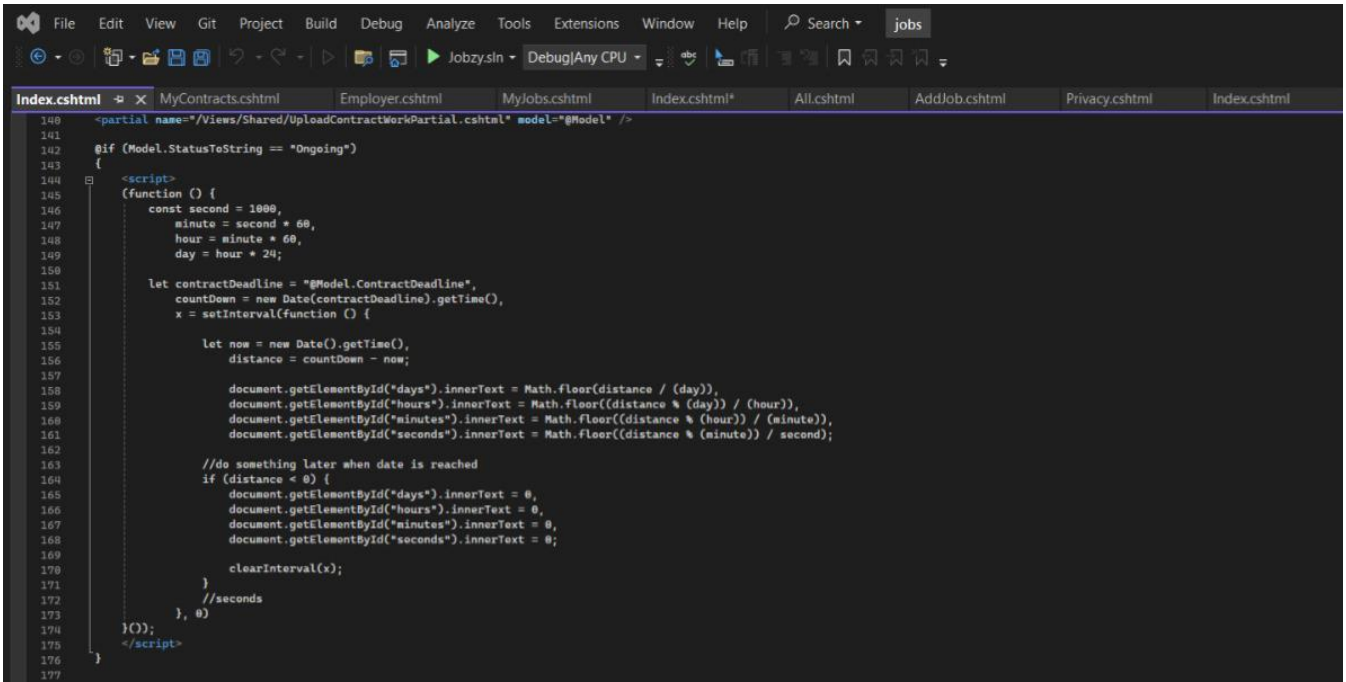
 @job.EmployerLocationToString
 @job.CategoryName
 $@job.Budget.ToString("N0")
Posted @job.TimeAgo

75 % No issues found
Ln: 1 Ch: 1 SPC LF

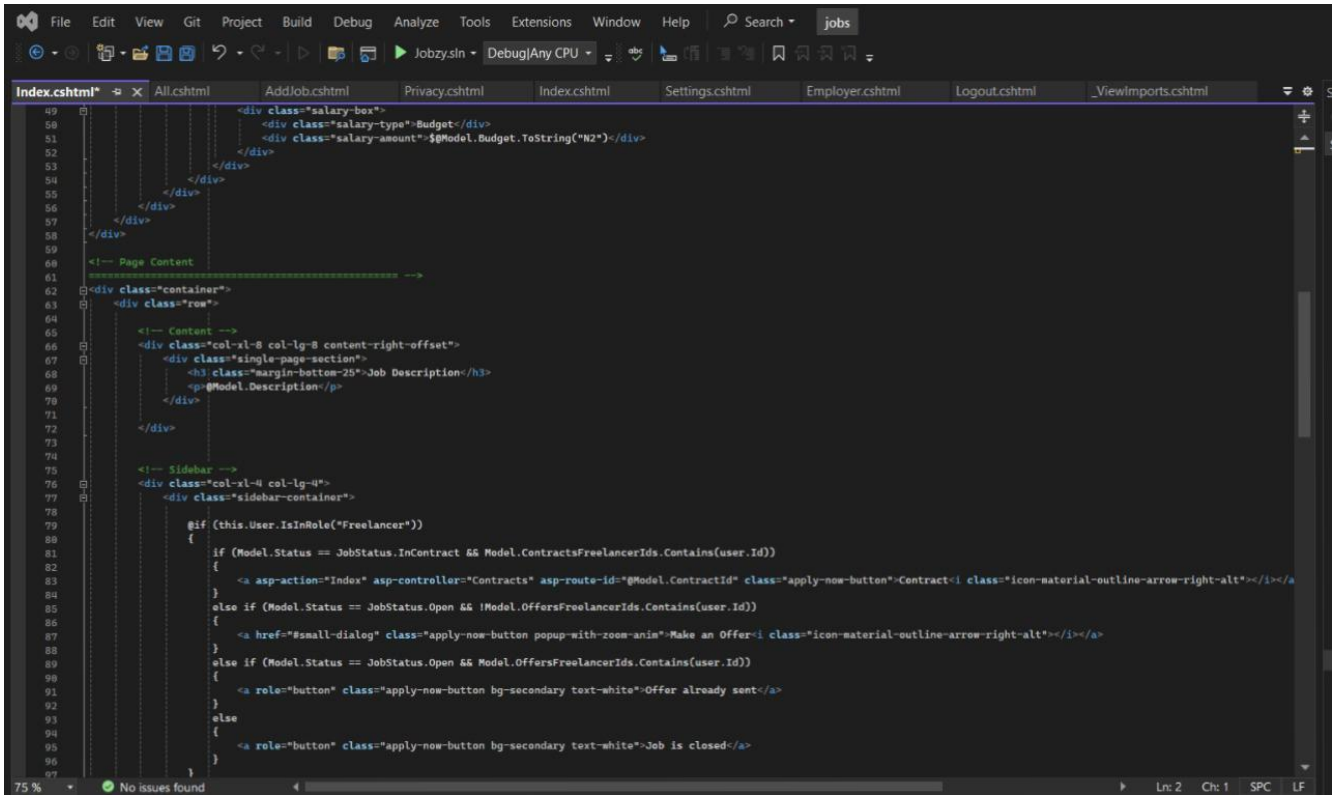
All jobs



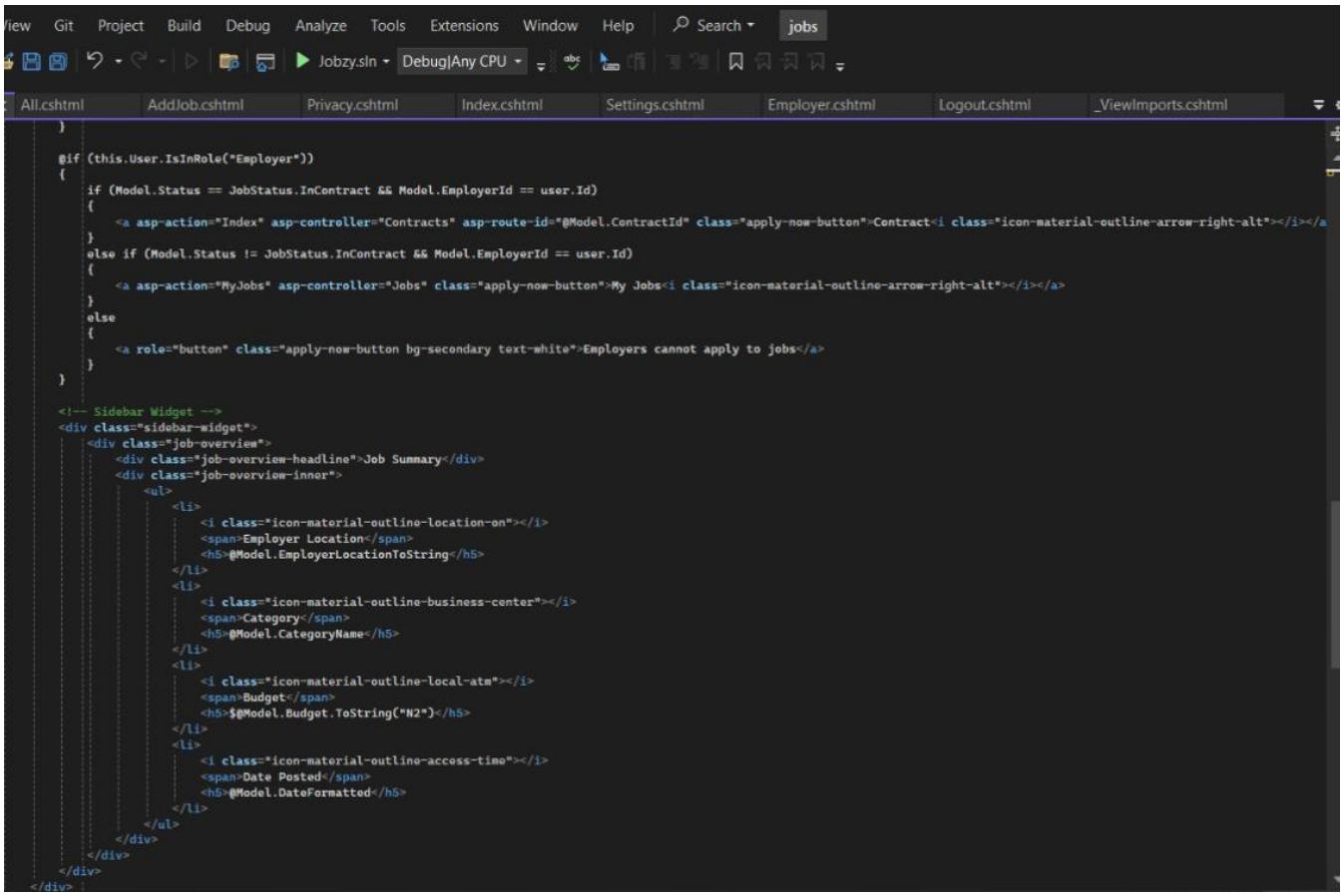
```
<div class="job-overview-headline">Job Summary</div>
<div class="job-overview-inner">
    <ul>
        <li>
            <i class="icon-material-outline-location-on"></i>
            <span>Employer Location</span>
            <h5>@Model.EmployerLocationToString</h5>
        </li>
        <li>
            <i class="icon-material-outline-business-center"></i>
            <span>Category</span>
            <h5>@Model.CategoryName</h5>
        </li>
        <li>
            <i class="icon-material-outline-local-atm"></i>
            <span>Budget</span>
            <h5>$@Model.Budget.ToString("N2")</h5>
        </li>
        <li>
            <i class="icon-material-outline-access-time"></i>
            <span>Date Posted</span>
            <h5>@Model.DateFormatted</h5>
        </li>
    </ul>
</div>
</div>
</div>
</div>
</div>
</div>

<div class="padding-bottom-100"></div>

<partial name="BigFooterPartial" />

<partial name="~/Views/Shared/ApplyFormPartial.cshtml" view-data="ViewData" />
```



```
<partial name="/Views/Shared/UploadContractWorkPartial.cshtml" model="@Model" />

@if (Model.StatusToString == "Ongoing")
{
    <script>
    (function () {
        const second = 1000,
            minute = second * 60,
            hour = minute * 60,
            day = hour * 24;

        let contractDeadline = "@Model.ContractDeadline",
            countDown = new Date(contractDeadline).getTime(),
            x = setInterval(function () {

                let now = new Date().getTime(),
                    distance = countDown - now;

                document.getElementById("days").innerText = Math.floor(distance / (day)),
                document.getElementById("hours").innerText = Math.floor((distance % (day)) / (hour)),
                document.getElementById("minutes").innerText = Math.floor((distance % (hour)) / (minute)),
                document.getElementById("seconds").innerText = Math.floor((distance % (minute)) / second);

                //do something later when date is reached
                if (distance < 0) {
                    document.getElementById("days").innerText = 0,
                    document.getElementById("hours").innerText = 0,
                    document.getElementById("minutes").innerText = 0,
                    document.getElementById("seconds").innerText = 0;

                    clearInterval(x);
                }
                //seconds
            }, 0)
    }());
    </script>
}
```

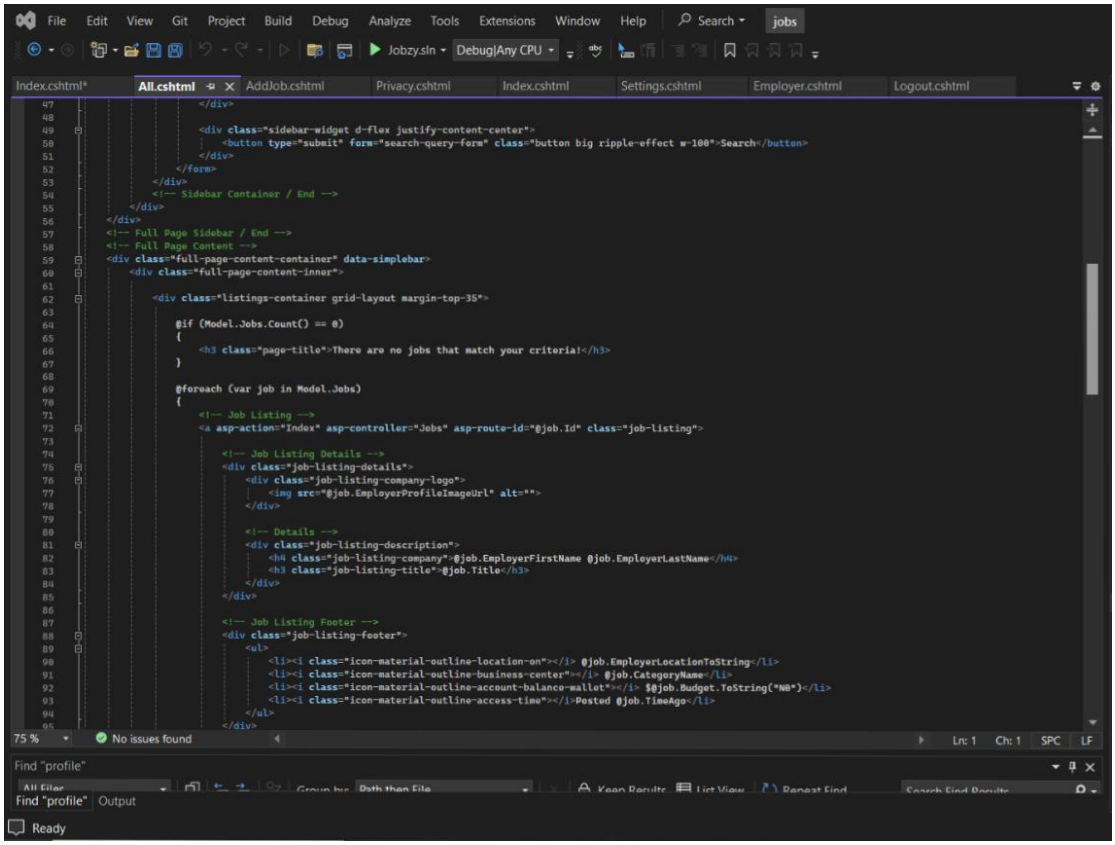


```
                        <div class="salary-box">
                            <div class="salary-type">Budget</div>
                            <div class="salary-amount">$@Model.Budget.ToString("N2")</div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Page Content
    ================================================== -->
    <div class="container">
        <div class="row">

            <!-- Content -->
            <div class="col-xl-8 col-lg-8 content-right-offset">
                <div class="single-page-section">
                    <h3 class="margin-bottom-25">Job Description</h3>
                    <p>@Model.Description</p>
                </div>

            </div>


            <!-- Sidebar -->
            <div class="col-xl-4 col-lg-4">
                <div class="sidebar-container">

                    @if (this.User.IsInRole("Freelancer"))
                    {
                        if (Model.Status == JobStatus.InContract && Model.ContractsFreelancerIds.Contains(user.Id))
                        {
                            <a asp-action="Index" asp-controller="Contracts" asp-route-id="@Model.ContractId" class="apply-now-button">Contract<i class="icon-material-outline-arrow-right-alt"></i></a
                        }
                        else if (Model.Status == JobStatus.Open && !Model.OffersFreelancerIds.Contains(user.Id))
                        {
                            <a href="#small-dialog" class="apply-now-button popup-with-zoom-anim">Make an Offer<i class="icon-material-outline-arrow-right-alt"></i></a>
                        }
                        else if (Model.Status == JobStatus.Open && Model.OffersFreelancerIds.Contains(user.Id))
                        {
                            <a role="button" class="apply-now-button bg-secondary text-white">Offer already sent</a>
                        }
                        else
                        {
                            <a role="button" class="apply-now-button bg-secondary text-white">Job is closed</a>
                        }
                    }
```

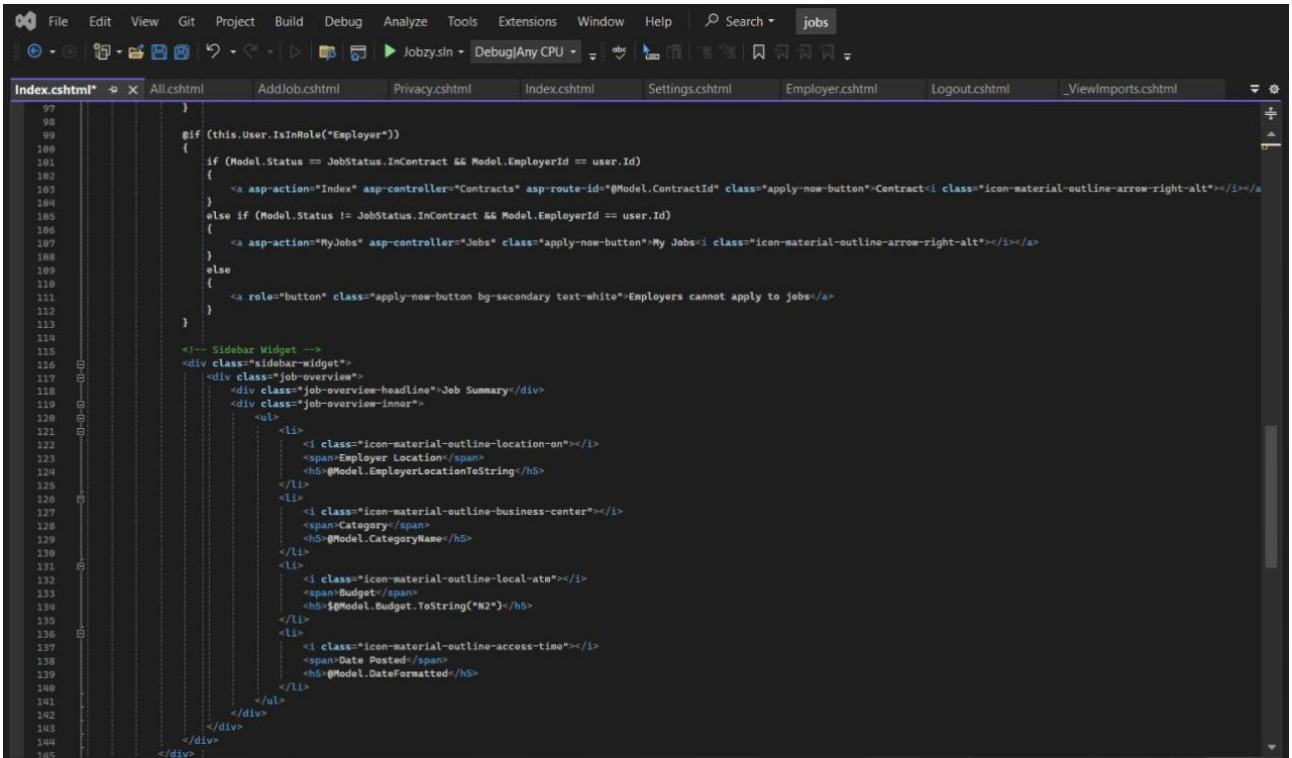


```
    }

    @if (this.User.IsInRole("Employer"))
    {
        if (Model.Status == JobStatus.InContract && Model.EmployerId == user.Id)
        {
            <a asp-action="Index" asp-controller="Contracts" asp-route-id="@Model.ContractId" class="apply-now-button">Contract<i class="icon-material-outline-arrow-right-alt"></i></a
        }
        else if (Model.Status != JobStatus.InContract && Model.EmployerId == user.Id)
        {
            <a asp-action="MyJobs" asp-controller="Jobs" class="apply-now-button">My Jobs<i class="icon-material-outline-arrow-right-alt"></i></a>
        }
        else
        {
            <a role="button" class="apply-now-button bg-secondary text-white">Employers cannot apply to jobs</a>
        }
    }

    <!-- Sidebar Widget -->
    <div class="sidebar-widget">
        <div class="job-overview">
            <div class="job-overview-headline">Job Summary</div>
            <div class="job-overview-inner">
                <ul>
                    <li>
                        <i class="icon-material-outline-location-on"></i>
                        <span>Employer Location</span>
                        <h5>@Model.EmployerLocationToString</h5>
                    </li>
                    <li>
                        <i class="icon-material-outline-business-center"></i>
                        <span>Category</span>
                        <h5>@Model.CategoryName</h5>
                    </li>
                    <li>
                        <i class="icon-material-outline-local-atm"></i>
                        <span>Budget</span>
                        <h5>$@Model.Budget.ToString("N2")</h5>
                    </li>
                    <li>
                        <i class="icon-material-outline-access-time"></i>
                        <span>Date Posted</span>
                        <h5>@Model.DateFormatted</h5>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</div>
```

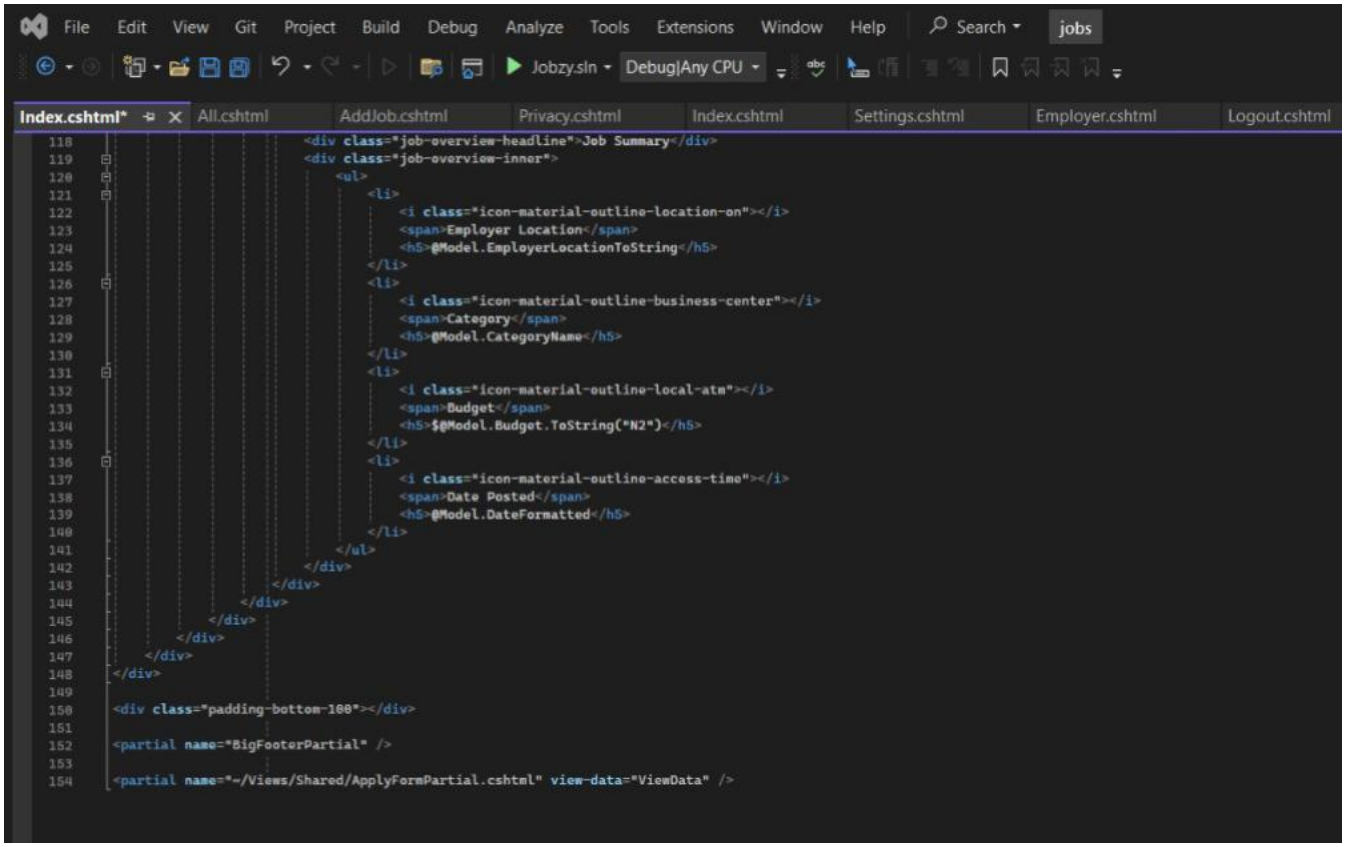
Job details



```
</div>

<div class="sidebar-widget d-flex justify-content-center">
    <button type="submit" form="search-query-form" class="button big ripple-effect w-100">Search</button>
</div>
</form>
</div>
<!-- Sidebar Container / End -->
</div>
</div>
<!-- Full Page Sidebar / End -->
<!-- Full Page Content -->
<div class="full-page-content-container" data-simplebar>
    <div class="full-page-content-inner">

        <div class="listings-container grid-layout margin-top-35">

            @if (Model.Jobs.Count() == 0)
            {
                <h3 class="page-title">There are no jobs that match your criteria!</h3>
            }

            @foreach (var job in Model.Jobs)
            {
                <!-- Job Listing -->
                <a asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing">

                    <!-- Job Listing Details -->
                    <div class="job-listing-details">
                        <div class="job-listing-company-logo">
                            <img src="@job.EmployerProfileImageUrl" alt="">
                        </div>

                        <!-- Details -->
                        <div class="job-listing-description">
                            <h4 class="job-listing-company">@job.EmployerFirstName @job.EmployerLastName</h4>
                            <h3 class="job-listing-title">@job.Title</h3>
                        </div>
                    </div>

                    <!-- Job Listing Footer -->
                    <div class="job-listing-footer">
                        <ul>
                            <li><i class="icon-material-outline-location-on"></i> @job.EmployerLocationToString</li>
                            <li><i class="icon-material-outline-business-center"></i> @job.CategoryName</li>
                            <li><i class="icon-material-outline-account-balance-wallet"></i> $@job.Budget.ToString("N0")</li>
                            <li><i class="icon-material-outline-access-time"></i>Posted @job.TimeAgo</li>
                        </ul>
                    </div>
```



```
}

@if (this.User.IsInRole("Employer"))
{
    if (Model.Status == JobStatus.InContract && Model.EmployerId == user.Id)
    {
        <a asp-action="Index" asp-controller="Contracts" asp-route-id="@Model.ContractId" class="apply-now-button">Contract<i class="icon-material-outline-arrow-right-alt"></i></a
    }
    else if (Model.Status != JobStatus.InContract && Model.EmployerId == user.Id)
    {
        <a asp-action="MyJobs" asp-controller="Jobs" class="apply-now-button">My Jobs<i class="icon-material-outline-arrow-right-alt"></i></a>
    }
    else
    {
        <a role="button" class="apply-now-button bg-secondary text-white">Employers cannot apply to jobs</a>
    }
}

<!-- Sidebar Widget -->
<div class="sidebar-widget">
    <div class="job-overview">
        <div class="job-overview-headline">Job Summary</div>
        <div class="job-overview-inner">
            <ul>
                <li>
                    <i class="icon-material-outline-location-on"></i>
                    <span>Employer Location</span>
                    <h5>@Model.EmployerLocationToString</h5>
                </li>
                <li>
                    <i class="icon-material-outline-business-center"></i>
                    <span>Category</span>
                    <h5>@Model.CategoryName</h5>
                </li>
                <li>
                    <i class="icon-material-outline-local-atm"></i>
                    <span>Budget</span>
                    <h5>$@Model.Budget.ToString("N2")</h5>
                </li>
                <li>
                    <i class="icon-material-outline-access-time"></i>
                    <span>Date Posted</span>
                    <h5>@Model.DateFormatted</h5>
                </li>
            </ul>
        </div>
    </div>
</div>
</div>
```


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml
Job Summary

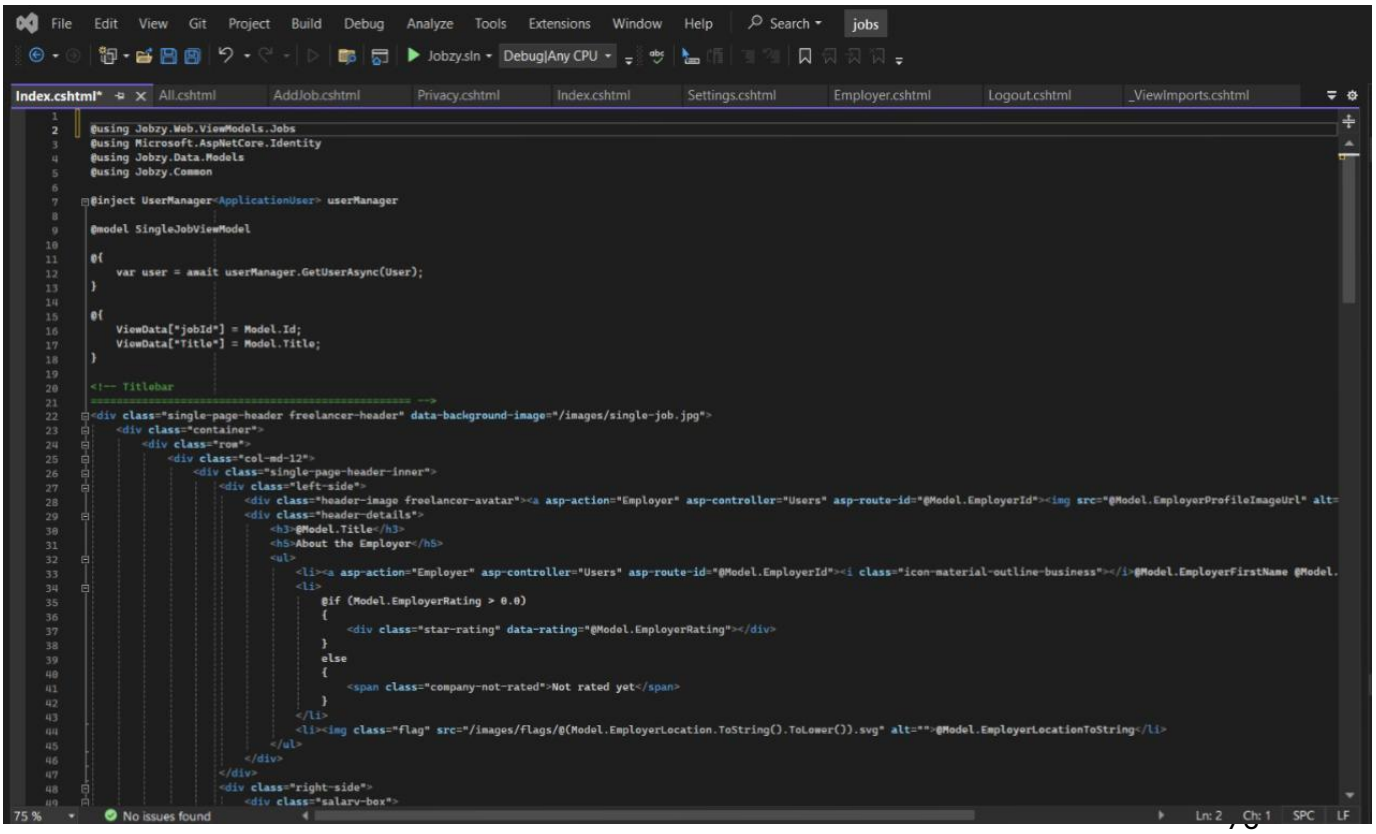
Employer Location
<h5>@Model.EmployerLocationToString</h5>

Category
<h5>@Model.CategoryName</h5>

Budget
<h5>$@Model.Budget.ToString("N2")</h5>

Date Posted
<h5>@Model.DateFormatted</h5>

<partial name="BigFooterPartial" />
<partial name="~/Views/Shared/ApplyFormPartial.cshtml" view-data="ViewData" />
jobs


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml
@using Jobzy.Web.ViewModels.Jobs
@using Microsoft.AspNetCore.Identity
@using Jobzy.Data.Models
@using Jobzy.Common
@inject UserManager<ApplicationUser> userManager
@model SingleJobViewModel
@{
var user = await userManager.GetUserAsync(User);
}
@{
ViewData["jobId"] = Model.Id;
ViewData["Title"] = Model.Title;
}
<!-- Titlebar
== -->

<a asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"><img src="@Model.EmployerProfileImageUrl" alt=

<h3>@Model.Title</h3>
<h5>About the Employer</h5>

<a asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId">@Model.EmployerFirstName @Model.

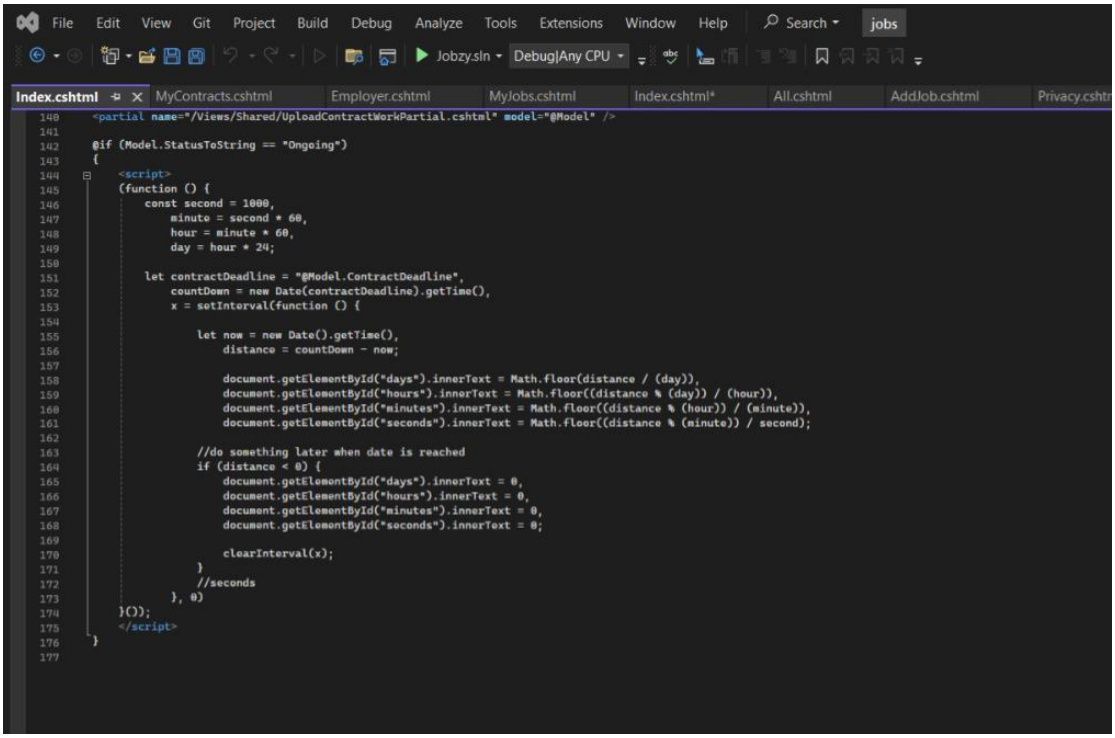
@if (Model.EmployerRating > 0.0)
{

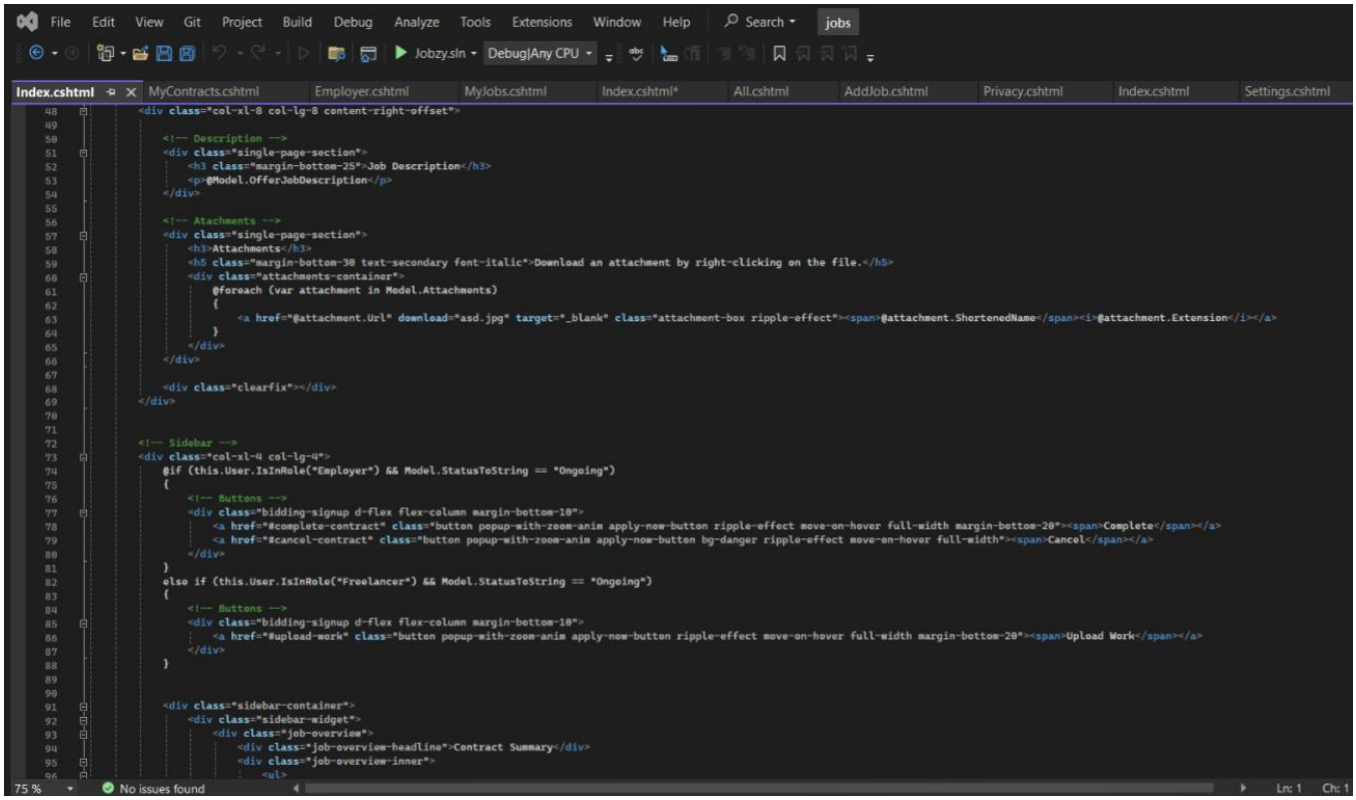
}
else
{
Not rated yet
}

<img class="flag" src="/images/flags/@(Model.EmployerLocation.ToString().ToLower()).svg" alt="">@Model.EmployerLocationToString

75 %
No issues found
Ln: 2 Ch: 1 SPC LF

Contract


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
jobs
Jobzy.sln
Debug|Any CPU
Index.cshtml
MyContracts.cshtml
Employer.cshtml
MyJobs.cshtml
Index.cshtml*
All.cshtml
AddJob.cshtml
Privacy.cshtml
<partial name="/Views/Shared/UploadContractWorkPartial.cshtml" model="@Model" />
@if (Model.StatusToString == "Ongoing")
{
<script>
(function () {
const second = 1000,
minute = second * 60,
hour = minute * 60,
day = hour * 24;
let contractDeadline = "@Model.ContractDeadline",
countDown = new Date(contractDeadline).getTime(),
x = setInterval(function () {
let now = new Date().getTime(),
distance = countDown - now;
document.getElementById("days").innerText = Math.floor(distance / (day)),
document.getElementById("hours").innerText = Math.floor((distance % (day)) / (hour)),
document.getElementById("minutes").innerText = Math.floor((distance % (hour)) / (minute)),
document.getElementById("seconds").innerText = Math.floor((distance % (minute)) / second);
//do something later when date is reached
if (distance < 0) {
document.getElementById("days").innerText = 0,
document.getElementById("hours").innerText = 0,
document.getElementById("minutes").innerText = 0,
document.getElementById("seconds").innerText = 0;
clearInterval(x);
}
//seconds
}, 0)
}());
</script>
}


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Jobzy.sln Debug|Any CPU
Index.cshtml MyContracts.cshtml Employer.cshtml MyJobs.cshtml Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml

<!-- Description -->

<h3 class="margin-bottom-25">Job Description</h3>
@Model.OfferJobDescription

<!-- Atachments -->

<h3>Attachments</h3>
<h5 class="margin-bottom-30 text-secondary font-italic">Download an attachment by right-clicking on the file.</h5>

@foreach (var attachment in Model.Attachments)
{
@attachment.ShortenedName@attachment.Extension
}

<!-- Sidebar -->

@if (this.User.IsInRole("Employer") && Model.StatusToString == "Ongoing")
{
<!-- Buttons -->

Complete
Cancel

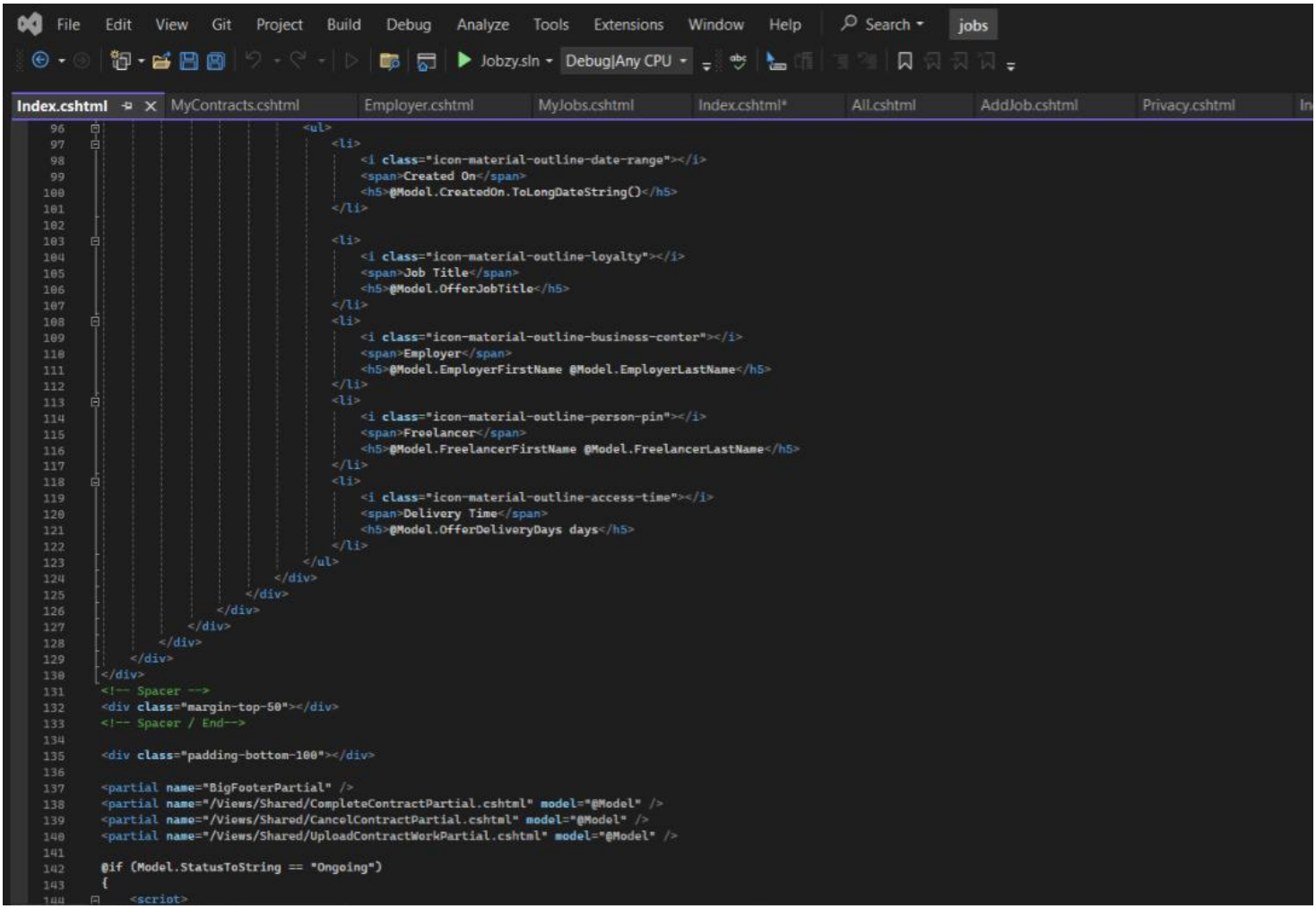
}
else if (this.User.IsInRole("Freelancer") && Model.StatusToString == "Ongoing")
{
<!-- Buttons -->

Upload Work

}

Contract Summary

75 %
No issues found
Ln: 1 Ch: 1


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
jobs
Jobzy.sln
Debug|Any CPU
Index.cshtml
MyContracts.cshtml
Employer.cshtml
MyJobs.cshtml
Index.cshtml*
All.cshtml
AddJob.cshtml
Privacy.cshtml

Created On
<h5>@Model.CreatedOn.ToLongDateString()</h5>

Job Title
<h5>@Model.OfferJobTitle</h5>

Employer
<h5>@Model.EmployerFirstName @Model.EmployerLastName</h5>

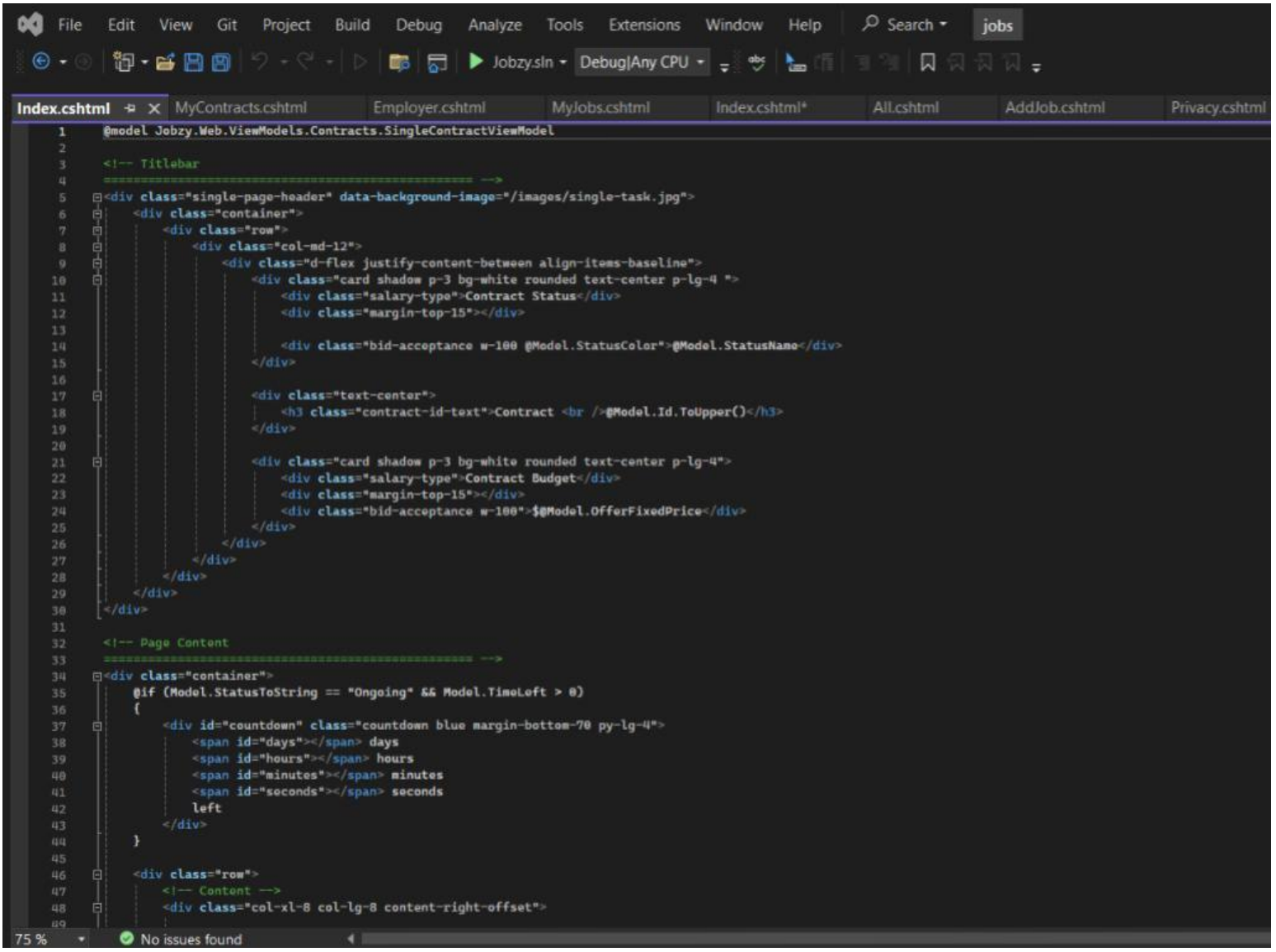
Freelancer
<h5>@Model.FreelancerFirstName @Model.FreelancerLastName</h5>

Delivery Time
<h5>@Model.OfferDeliveryDays days</h5>

<!-- Spacer -->

<!-- Spacer / End-->

<partial name="BigFooterPartial" />
<partial name="/Views/Shared/CompleteContractPartial.cshtml" model="@Model" />
<partial name="/Views/Shared/CancelContractPartial.cshtml" model="@Model" />
<partial name="/Views/Shared/UploadContractWorkPartial.cshtml" model="@Model" />
@if (Model.StatusToString == "Ongoing")
{
<script>


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
jobs
Jobzy.sln
Debug|Any CPU
Index.cshtml
MyContracts.cshtml
Employer.cshtml
MyJobs.cshtml
Index.cshtml*
All.cshtml
AddJob.cshtml
Privacy.cshtml
@model Jobzy.Web.ViewModels.Contracts.SingleContractViewModel
<!-- Titlebar

Contract Status

@Model.StatusName

<h3 class="contract-id-text">Contract
@Model.Id.ToUpper()</h3>

Contract Budget

$@Model.OfferFixedPrice

<!-- Page Content

@if (Model.StatusToString == "Ongoing" && Model.TimeLeft > 0)
{

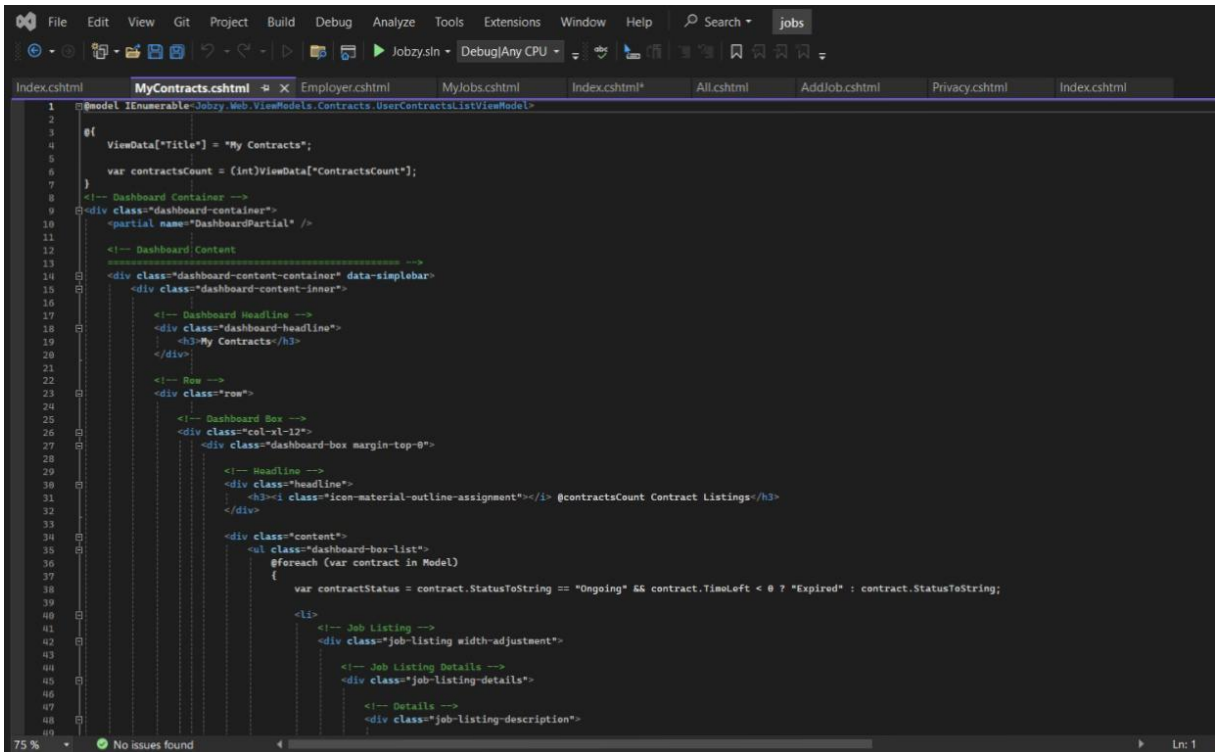
 days
 hours
 minutes
 seconds
left

}

<!-- Content -->

75 %
No issues found

User contract



Chapter Five (Testing)

5.1 Freelance Website Test Case

5.2 Usability Test Cases

5.3 Security Test Cases

5.4 Performance Test Cases

5.5 Accessibility Test Cases

5.6 Content Test Cases

5.7 Integration Test Cases

5.8 Search Engine Test Cases

5.9 Notification Test Cases

5.10 Offer Management Test Cases

5.11 Freelancer Profile Test Cases

5.12 Future Enhancement

5.13 Contract Management Test Cases

Freelance Website Test Documentation

Functional Test Cases

1. User Registration

Test Case ID: TC001

• **Test Name**: User Registration

• **Test Steps:**

1. Enter valid data in the registration form.
2. Submit the registration form.
3. Verify successful registration.

• Result: User has be registered successfully, and a confirmation email is

received.

2. Login Functionality

Test Case ID: TC002

• **Test Name**: Login Functionality

• **Test Steps:**

1. Enter valid login credentials.
2. Submit the login form.
3. Verify successful login.

• Result: User logged in successfully.

3. Profile Management

Test Case ID: TC003

• Test Name: Editing User Profile Information

• Test Steps:

1. Navigate to Profile Page:
    - Log in with valid credentials.
    - Go to the user profile page.

2. Update Profile Information:

- Click on the "Edit Profile" button.
- Modify fields such as bio, skills, or contact information.
- Save the changes.

3. Verify Changes:
    - Navigate back to the profile page.
    - Confirm that the updated information is displayed correctly.

Result: User able to successfully update their profile information, and the

    - changes should be accurately reflected on the profile page.
- Notes:
    - This test case assumes that users can edit various aspects of their profile,
      including bio, skills, and contact details.
    - Ensure that the website provides appropriate feedback to users during the update process.

Test Case Execution Results:

Test Case ID : TC003 -

Test Result : pass

Usability Test Cases

4. Cross-Browser Compatibility

Test Case ID: TC004

- **Test Name**: Cross-Browser Compatibility
- **Test Steps:**

1. Access the website using Google Chrome.
2. Access the website using Mozilla Firefox.
3. Access the website using Apple Safari.
4. Access the website using Microsoft Edge.

• **Result**: The website displays correctly on all tested browsers.

5. Responsive Design

Test Case ID: TC005

• **Test Name**: Responsive Design

• **Test Steps:**

1. Access the website on a desktop computer.
2. Access the website on a tablet device.
3. Access the website on a mobile phone.

Result: The website is responsive and displays correctly on all tested devices

Security Test Cases

6. SSL Certificate

Test Case ID: TC006

• **Test Name**: SSL Certificate

• **Test Steps**:

1. Check for the presence of SSL.

• **Result**: The website uses SSL to secure data transmission.

7. Password Security

Test Case ID: TC007

• **Test Name**: Password Security

• **Test Steps:**

1. Test password strength requirements during registration.
2. Test password strength requirements during password change.

• **Result**: Password strength requirements are enforced.

8. Session Management

Test Case ID: TC008

• **Test Name**: Session Management

• **Test Steps**:

1. Log in with valid credentials.
2. Stay inactive for a specified period.
3. Verify automatic logout.

• **Result**: User is automatically logged out after a period of inactivity.

Performance Test Cases

9. Page Load Time

Test Case ID: TC009

• **Test Name**: Page Load Time

• **Test Steps**:

1. Load key pages on the website, such as the homepage and user
1. dashboard.
2. Measure the time it takes for each page to load.

• **Result**: Pages load within an acceptable time frame, as per performance

benchmarks.

10. Scalability

Test Case ID: TC010

• **Test Name:** Scalability

• **Test Steps:**

1. Simulate a large number of concurrent users accessing the website.
2. Observe system behavior and response times.

• **Result**: The website scales effectively without significant performance

degradation.

Accessibility Test Cases

11. Screen Reader Compatibility

Test Case ID: TC011

• **Test Name**: Screen Reader Compatibility

• **Test Steps:**

1. Navigate through key pages using a screen reader.
2. Verify that all essential information is conveyed audibly.

• **Result**: The website is accessible to users with visual impairments using screen

readers.

12. Keyboard Navigation

Test Case ID: TC012

• **Test Name:** Keyboard Navigation

• **Test Steps:**

1. Navigate through the website using only the keyboard.

• **Result**: All interactive elements are accessible and usable via keyboard

navigation.

Content Test Cases

13. Content Accuracy

Test Case ID: TC013

• **Test Name:** Content Accuracy

• **Test Steps:**

1. Review all textual and graphical content on the website.

• **Result**: All content is accurate, up-to-date, and aligned with the website's purpose.

14. Error Handling

Test Case ID: TC014

• **Test Name:** Error Handling

• **Test Steps:**

1. Intentionally trigger errors by entering invalid data or navigating to non-existent pages.

• **Result**: Meaningful error messages are displayed, guiding users on corrective actions.

Integration Test Cases

15. Chat Section

Test Case ID: TC015

• **Test Name**: Chat Functionality

• **Test Steps:**

1. Initiate a chat with another user.
2. Send messages and verify real-time delivery.
3. Test multimedia sharing (if applicable).

• **Result:** Chat functionality works smoothly, supporting real-time communication between users.

16. Contract Section

Test Case ID: TC016

• **Test Name**: Creating and Managing Contracts

• **Test Steps:**

1. Create a new contract with another user.
2. Verify contract details, terms, and parties involved.
3. Edit and update contract details.

• **Result**: Users can create, manage, and edit contracts seamlessly.

Search Engine Test Cases

17. Basic Search Functionality

Test Case ID: TC017

• **Test Name:** Basic Search Functionality

• **Test Steps:**

1. Enter a keyword in the search bar.
2. Review search results.

• **Result**: Relevant results are displayed based on the entered keyword.

18. Advanced Search Options

Test Case ID: TC018

- **Test Name**: Advanced Search Options
- **Test Steps:**
    1. Use advanced search filters such as category, date, or location.
- **Result**: Search results are refined based on the selected advanced search options.

Notification Test Cases

19. Notification Delivery

Test Case ID: TC019

- **Test Name:** Notification Delivery
- **Test Steps:**
    1. Trigger events that generate notifications (e.g., new message, contract update).
    2. Verify timely delivery of notifications to the user.
- **Result**: Users receive notifications promptly for relevant events.

20. Notification Preferences

Test Case ID: TC020

- **Test Name:** Notification Preferences
- **Test Steps:**
    1. Adjust notification preferences in user settings.

2. Test if preferences are applied to incoming notifications.

• **Result**: Users can customize their notification settings, and changes are reflected accordingly.

Offer Management Test Cases

21. Creating Offers

Test Case ID: TC021

• **Test Name:** Creating Offers

• **Test Steps:**

1. Navigate to the offer creation section.
2. Fill in the required details for a new offer.
3. Submit the offer.

• **Result**: Offers can be created successfully with accurate details.

22. Reviewing Offers

Test Case ID: TC022

• **Test Name:** Reviewing Offers

• **Test Steps:**

1. Access the list of offers created.
2. Verify that details such as title, description, and pricing are accurate.

• **Result**: Offer details are correctly displayed for review.

23. Accepting/Declining Offers

Test Case ID: TC023

• **Test Name:** Accepting/Declining Offers

• **Test Steps:**

1. Receive an offer.
2. Accept or decline the offer.
3. Verify that the status of the offer is updated accordingly.

• **Result:** The user can successfully accept or decline offers, and the status is reflected.

Freelancer Profile Test Cases

24. Creating and Updating Freelancer Profile

Test Case ID: TC024

• **Test Name**: Creating and Updating Freelancer Profile

• **Test Steps:**

1. Navigate to the freelancer profile creation/editing section.
2. Fill in or update the required information (e.g., skills, experience).
3. Save the changes.

• **Result**: Freelancers can create and update their profiles successfully.

25. Viewing Freelancer Profiles

Test Case ID: TC025

• **Test Name:** Viewing Freelancer Profiles

• **Test Steps:**

1. Search for a freelancer's profile.
2. Verify that details such as skills, bio, and portfolio items are accurate.

• **Result**: Freelancer profiles are displayed with accurate information.

Future Enhancement

26. Adding Payment Method (Future Enhancement)

Test Case ID: TC026

• **Test Name:** Adding Payment Method (Future Enhancement)

• **Test Steps:**

1. Navigate to the payment method section (not available).
2. Verify the absence of active payment methods.

• **Result:** As of now, the website does not have an active payment method feature.

Note:

• The addition of a payment method is planned as a future enhancement. Users

are currently not required to add payment methods.

Contract Management Test Cases

27. Initiating a Contract

Test Case ID: TC027

• **Test Name**: Initiating a Contract

• **Test Steps:**

1. Select a job or project to propose a contract.
2. Fill in contract details, including terms and milestones.
3. Submit the contract proposal.

• **Result**: Users can initiate a contract proposal successfully.

28. Reviewing Contract Details

Test Case ID: TC028

• **Test Name:** Reviewing Contract Details

• **Test Steps:**

1. Access the list of active contracts.
2. Verify that contract details, including terms and milestones, are accurate.

• **Result**: Contract details are correctly displayed for review.

29. Modifying Contract Terms

Test Case ID: TC029

• **Test Name:** Modifying Contract Terms

• **Test Steps:**

1. Access an active contract.

2. Request or discuss modifications to contract terms with the client/freelancer.
3. Confirm changes and update the contract.

- **Result**: Contract terms can be modified with mutual agreement.

Chapter SIX : Results & Discussion

Results:

Our freelancing website has experienced significant growth since its launch, with a steady increase in registered freelancers and posted projects. The diverse job categories have attracted professionals from various fields, contributing to a vibrant and dynamic community.

Client satisfaction metrics indicate a high rate of successful project completions, showcasing the effectiveness of our platform in connecting clients with skilled freelancers. Additionally, positive feedback and testimonials highlight the positive experiences of both freelancers and clients.

Discussion:

The success of our freelancing website can be attributed to several key factors. The user-friendly interface has played a crucial role in attracting and retaining users, providing an accessible platform for freelancers to showcase their talents

and for clients to find the right expertise.

The implementation of robust security measures has fostered trust among our users, ensuring a safe environment

for transactions and interactions. Furthermore, our commitment to maintaining a diverse range of job categories has resulted in a rich ecosystem where freelancers can explore various opportunities.

Looking ahead, continuous efforts will be directed towards refining user experiences, expanding job categories, and exploring innovative features to further enhance the platform's functionality. The positive momentum observed in the early stages sets a promising foundation for the continued success and growth of our freelancing website.

Chapter SEVEN : Conclusion

In conclusion, a freelancing website serves as a transformative force in the modern workforce, reshaping how professionals connect and collaborate in the digital era. By providing a dynamic platform that transcends geographical boundaries, freelancing websites empower individuals to unleash their potential, offering diverse job opportunities and entrepreneurial pathways.

The ability to work remotely and flexibly not only benefits freelancers but also caters to the evolving needs of businesses seeking specialized skills without the commitment of traditional employment. The scope of a freelancing website encompasses a global talent pool, promoting diversity and fostering a vibrant community where freelancers can learn, grow, and thrive.

Moreover, the emphasis on secure transactions, fair compensation, and transparent communication builds trust among users. The inclusion of reviews and ratings ensures a reputation system that adds an extra layer of credibility to the freelancing ecosystem.

As we navigate the ever-changing landscape of work, freelancing websites continue to adapt, offering innovative solutions and staying at the forefront of the gig economy. In essence, these platforms play a pivotal role in shaping a future of work that is flexible, collaborative, and driven by individual empowerment.

Chapter EIGHT : Future Work

Since the project purpose does not end with the final discussion, we are planning to develop more items to make the applications work the best afterwards:

1.Adding the expected missing features

2.Adding new features such as a section for tips and emergencies.

3.Adding helpful features.

4. Enabling the data conducted from the app to help the research \ campaigns done, or decisions made by the government.

5.As with any application that deals with sensitive personal information, it's important to prioritize data privacy and security. Future work could focus on improving the application's security measures, such as implementing encryption, access controls, and other measures to protect users' information.

7. Adding new results like the alerting system.

8. Making the basic model faster and make it trained in more data in various fields and in various form to increase

performance.

9. Making a team in constant continues to keep our site in permanent maintenance and make it at an appropriate speed for users.

THANK YOU.