



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]

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).....8

Chapter two (Planning & Analysis).....13

Chapter three (Software Design)28

Chapter Four (Implementation).....37

Chapter Five (Testing).....82

Chapter Six (Results & Discussion)97

Chapter Seven (Conclusion)99

Chapter Eight (Future Work).....101

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 clients.

5. Improved client engagement: A website for jobs can help improve client Engagement by providing clients 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 client 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 client 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 client satisfaction. For freelancer, the Website can help reduce costs associated with administrative tasks, improve Client 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 - laptops and laptops.

- Codecs.
- Internet service.
- Design software.

Timetable:

- 1 month for gathering information and Diagrams.
- 2 months 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.

2.1.1 Estimated 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: Clint want to find a freelancer to book with. Post: User shows a list of freelancers related to his name that search by it. Description: Clint enters search name or category or both in search box and ask for result.
- 5.Make Reservation Actor: Clint . Pre: Clint want to make a Reservation. Post: Clint 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 Clint) . 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: Clint Pre: clinte search for the desired freelancer. Post :
Rate added successfully. Description: Clint 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.

2.4.3 Domain 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 a probability 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 client 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.

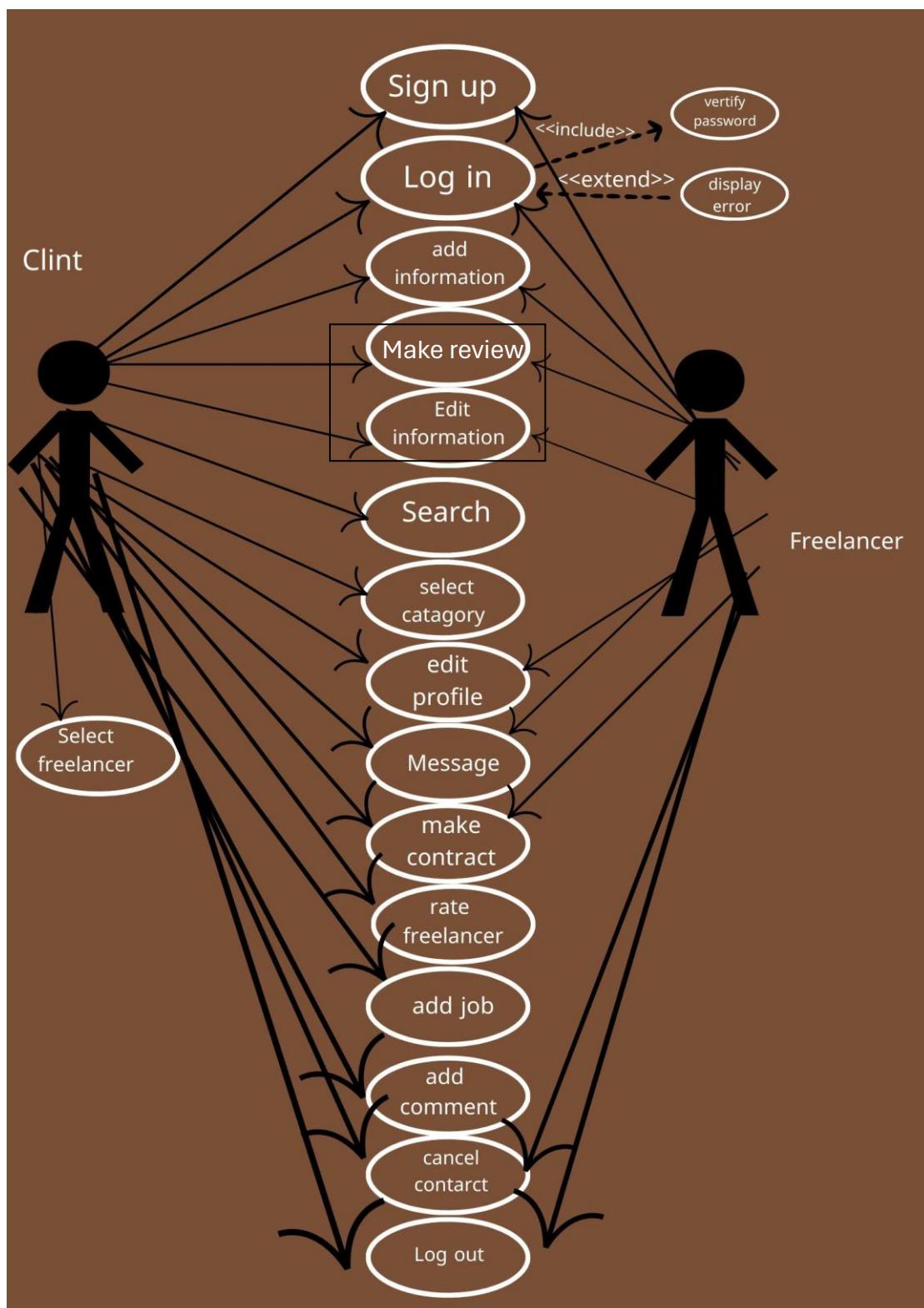
2.6 Risk and Risk Managements:

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 clients, 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 and procedures 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



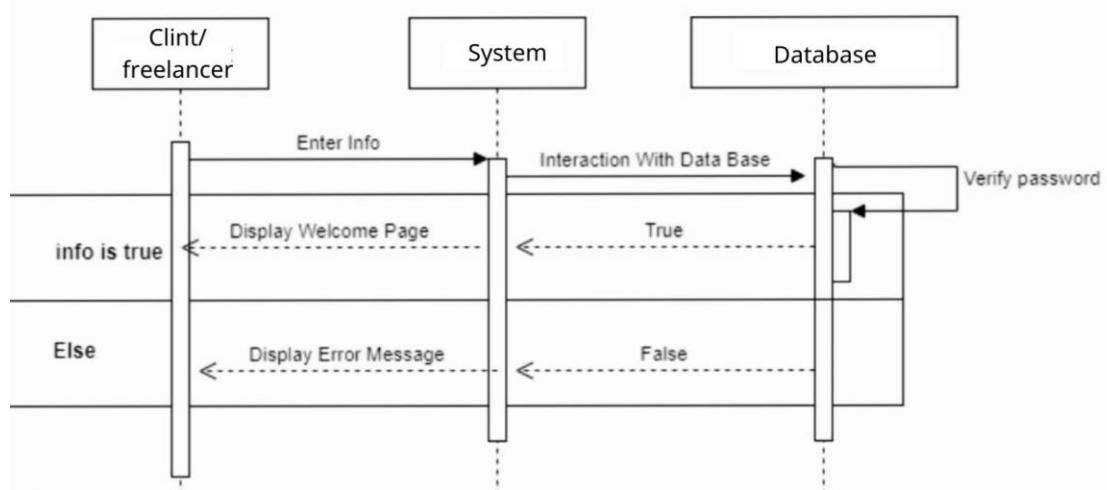
Chapter THREE : SoftWare Design



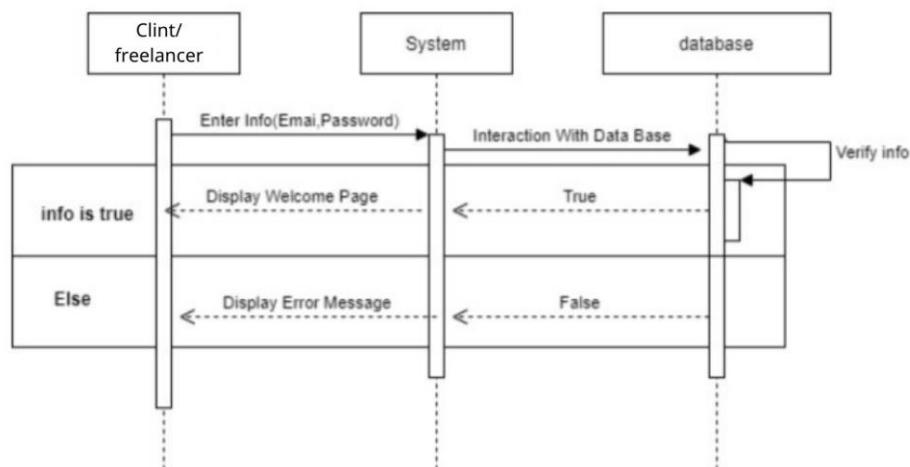
3.1 Use case diagram

3.2 Sequence diagram

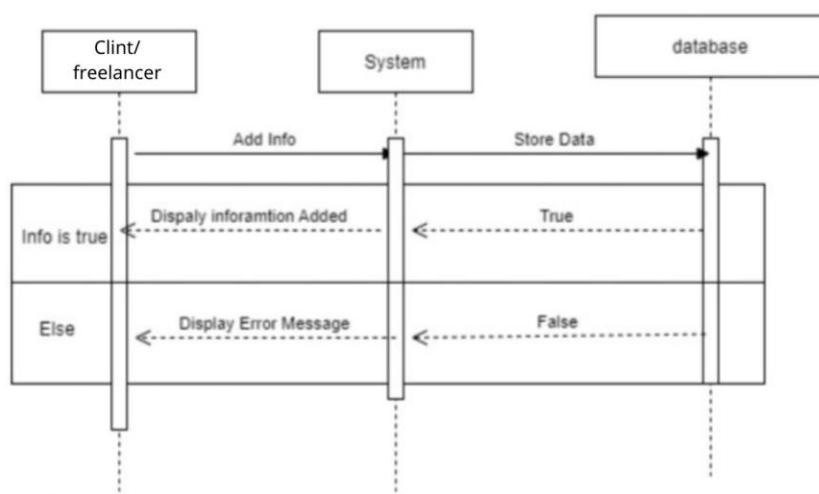
Signup



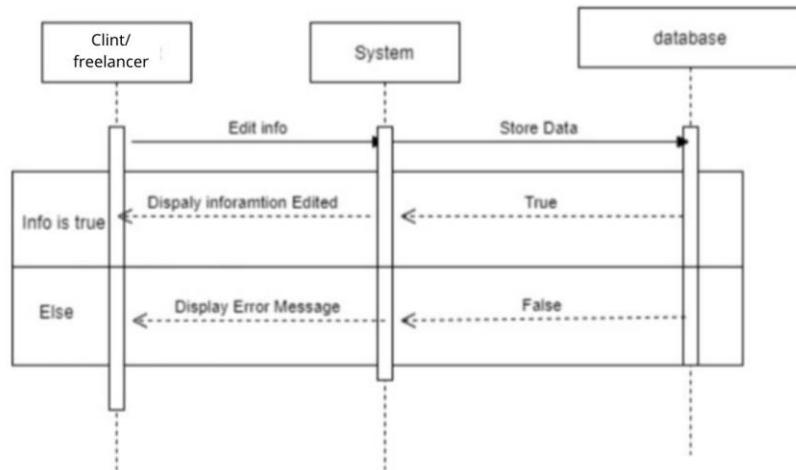
Log in



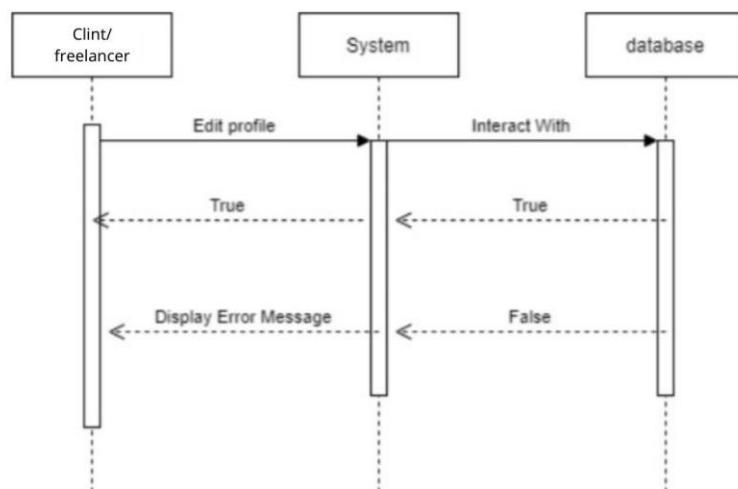
Add info



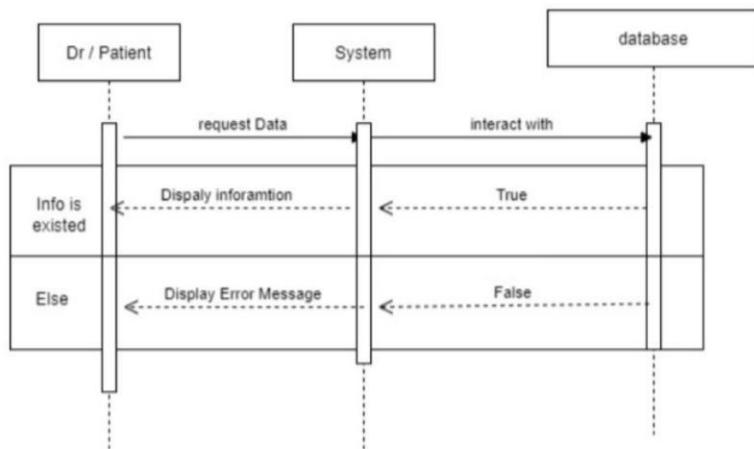
Edit Info



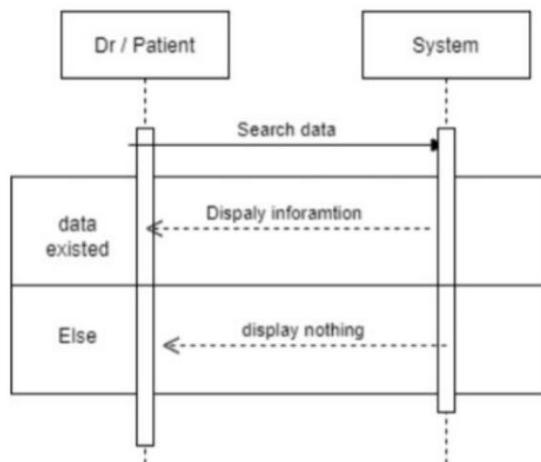
Edit profile



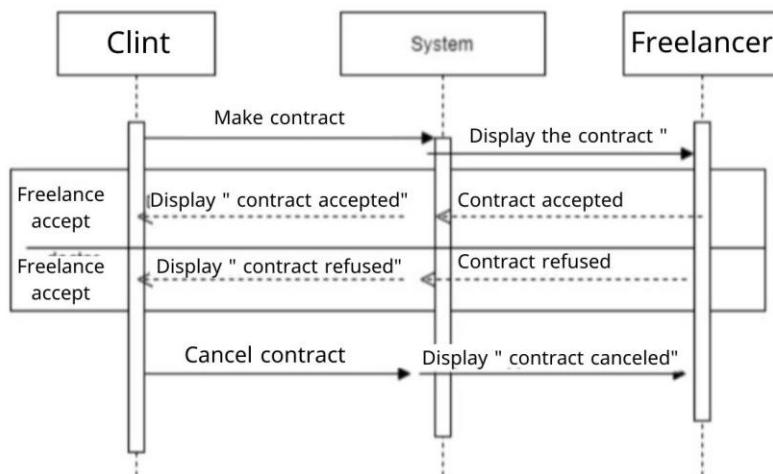
Make review



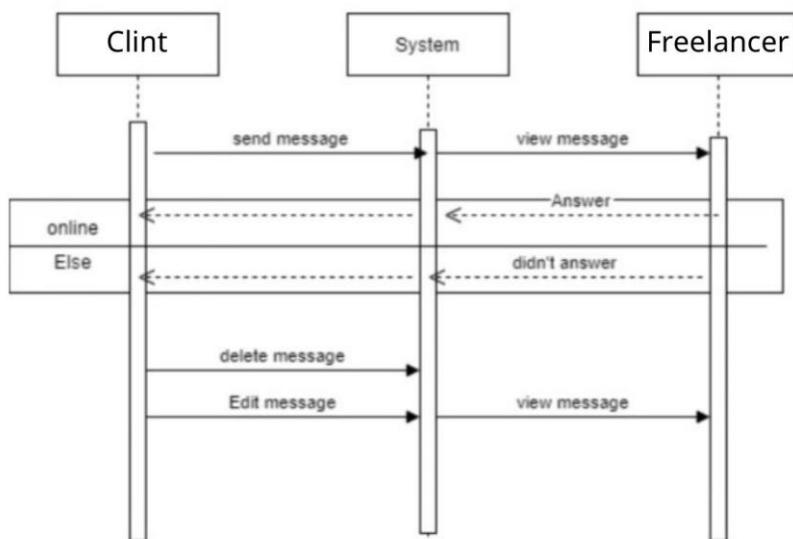
Search



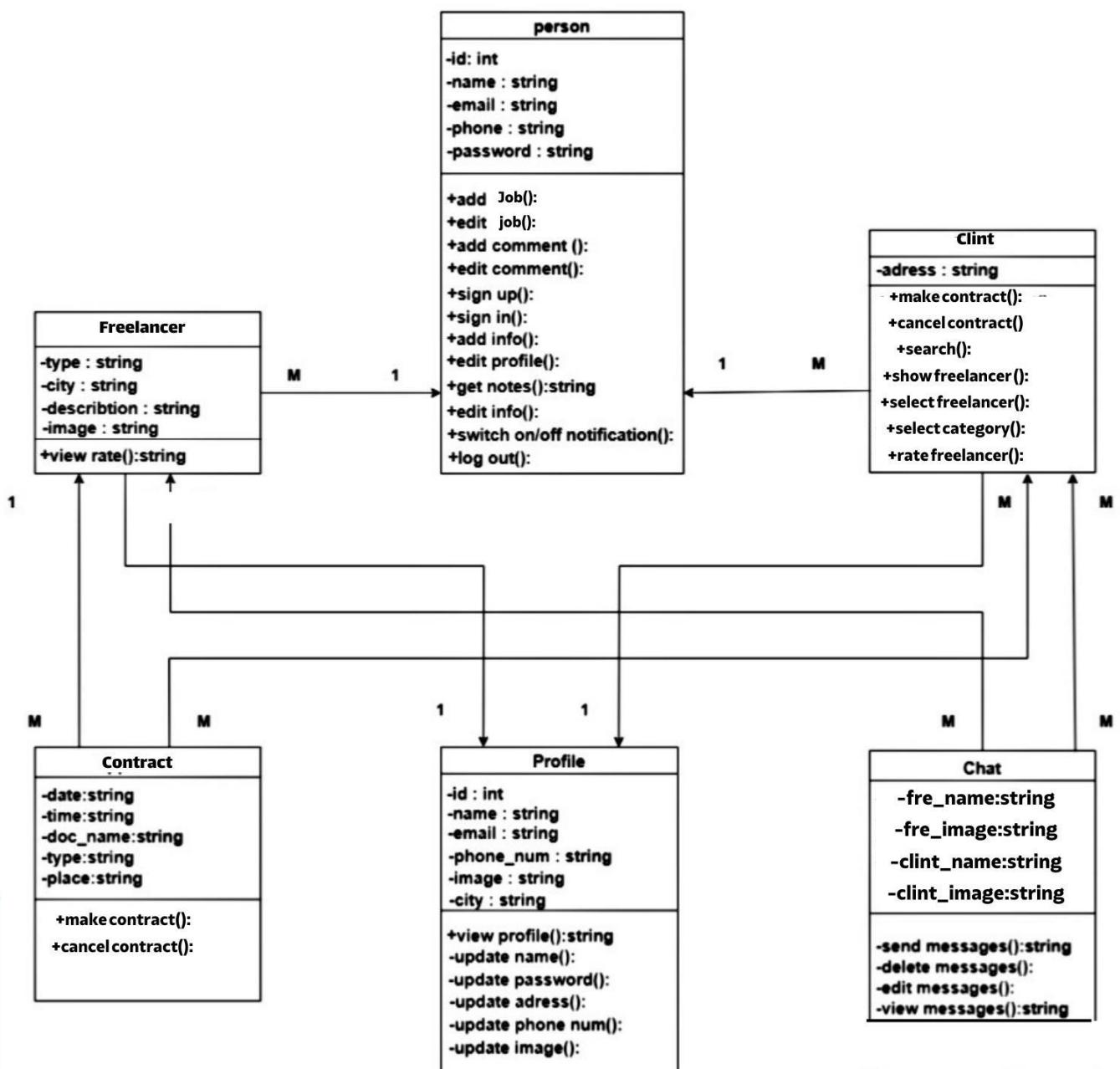
Make contract



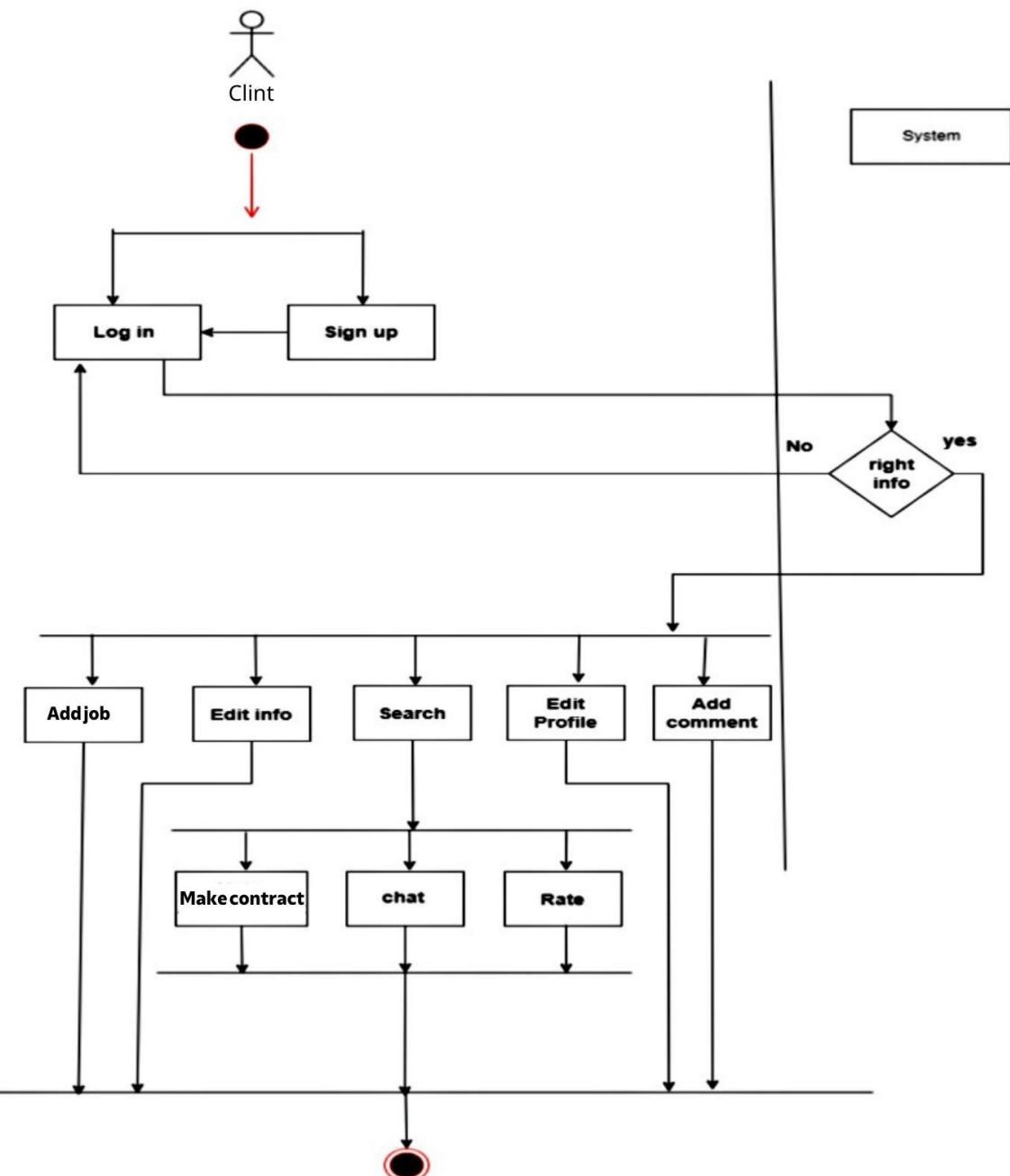
Message

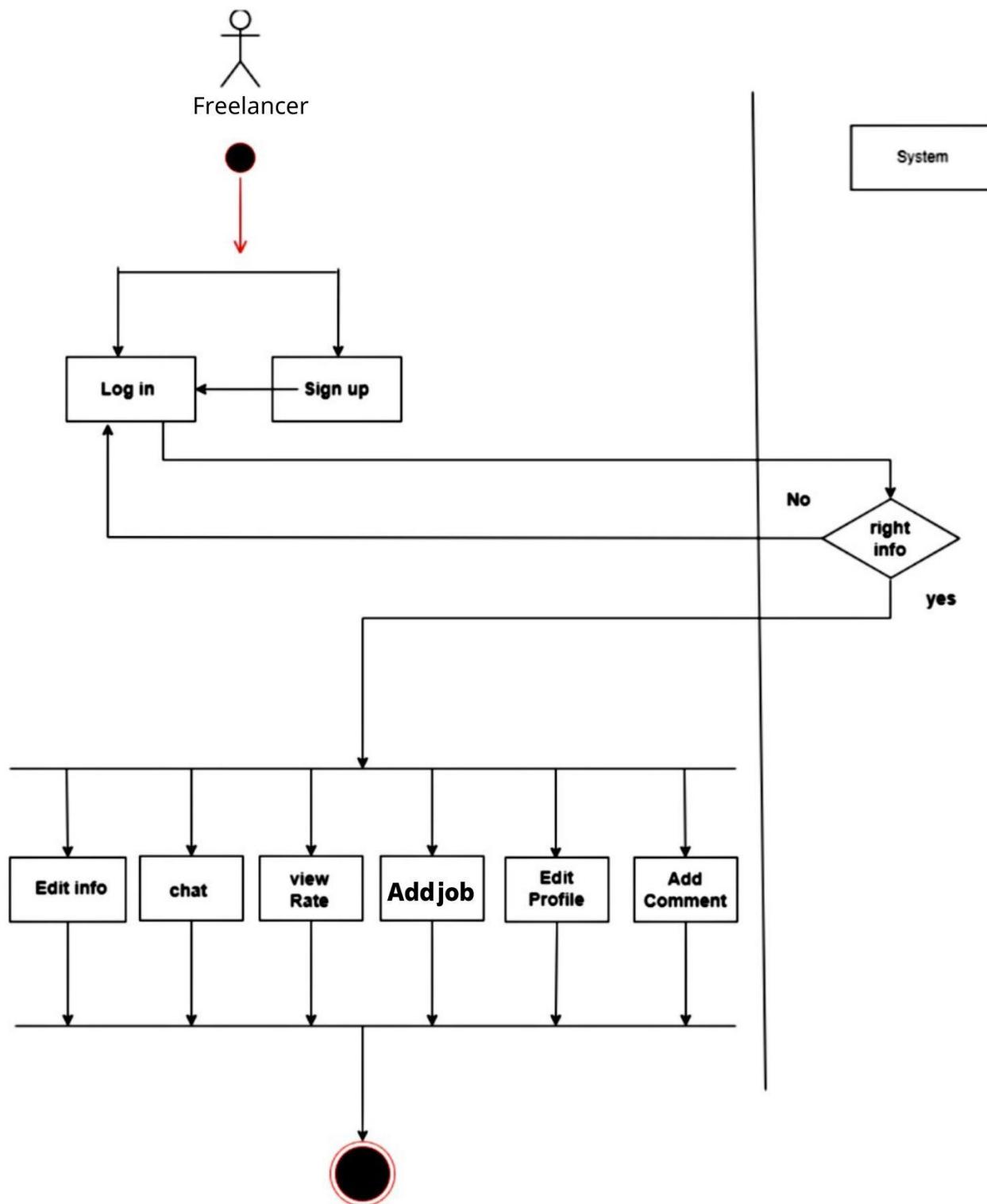


3.3 class Diagram



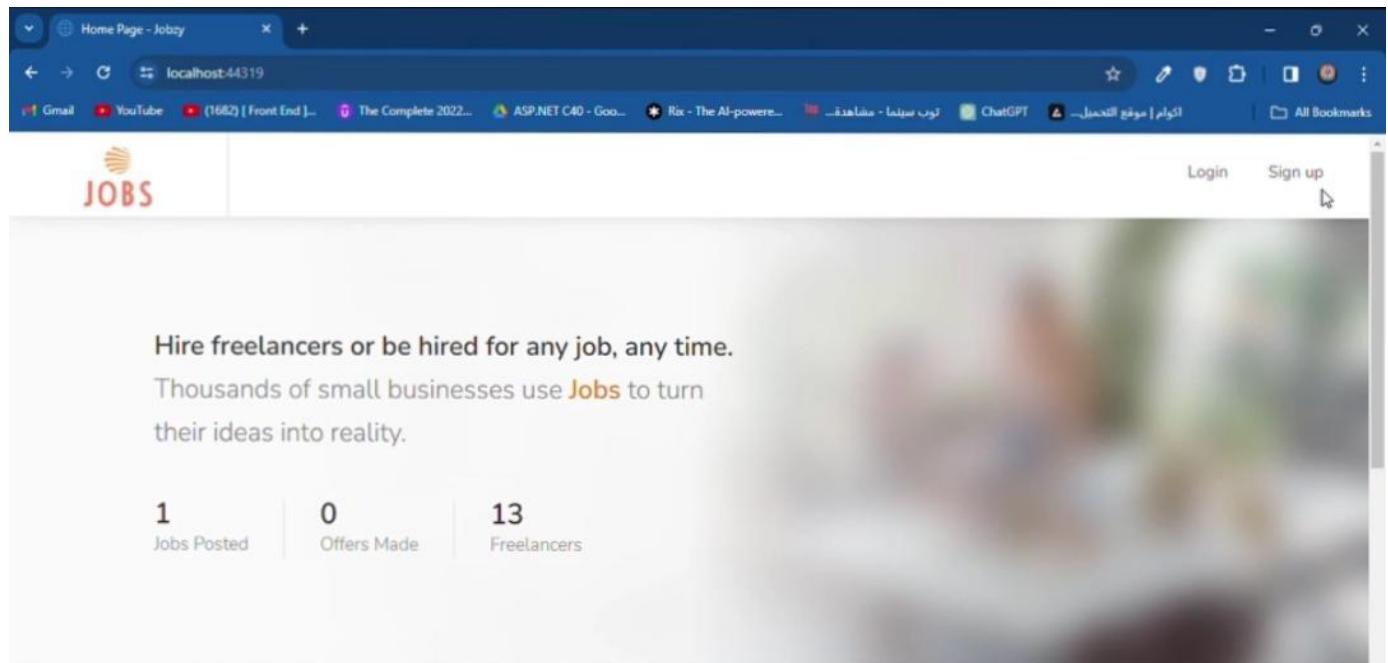
3.4 Activity Diagram





Chapter four :Work flow

Welcome home



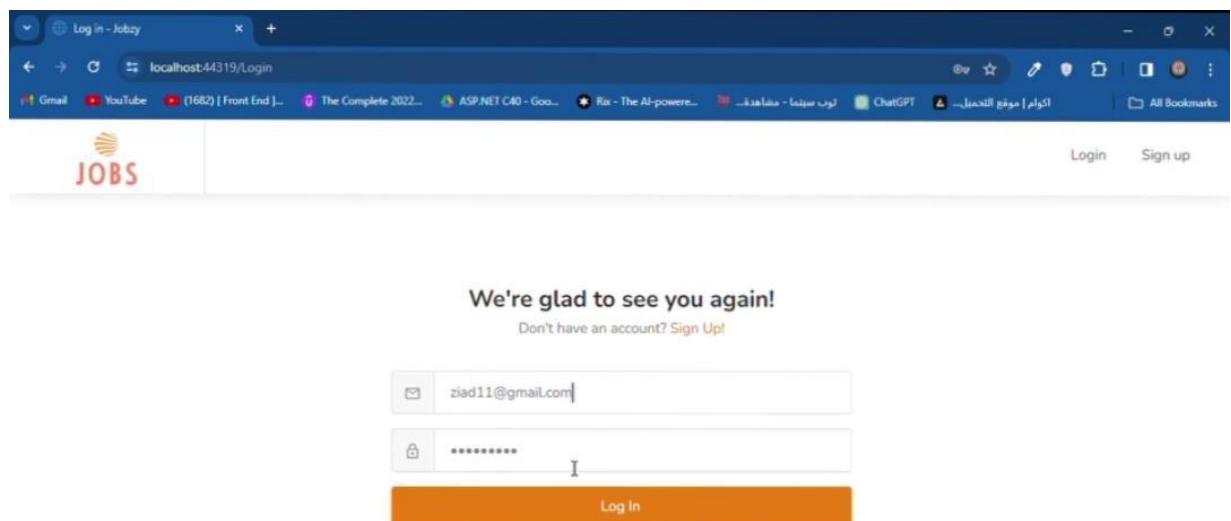
Sign up

The screenshot shows the registration form for Jobzy. The URL in the browser is 'localhost:44319/Register'. The form has two main tabs: 'Freelancer' (grayed out) and 'Employer' (highlighted in green). The 'Employer' tab is active. The form fields include:

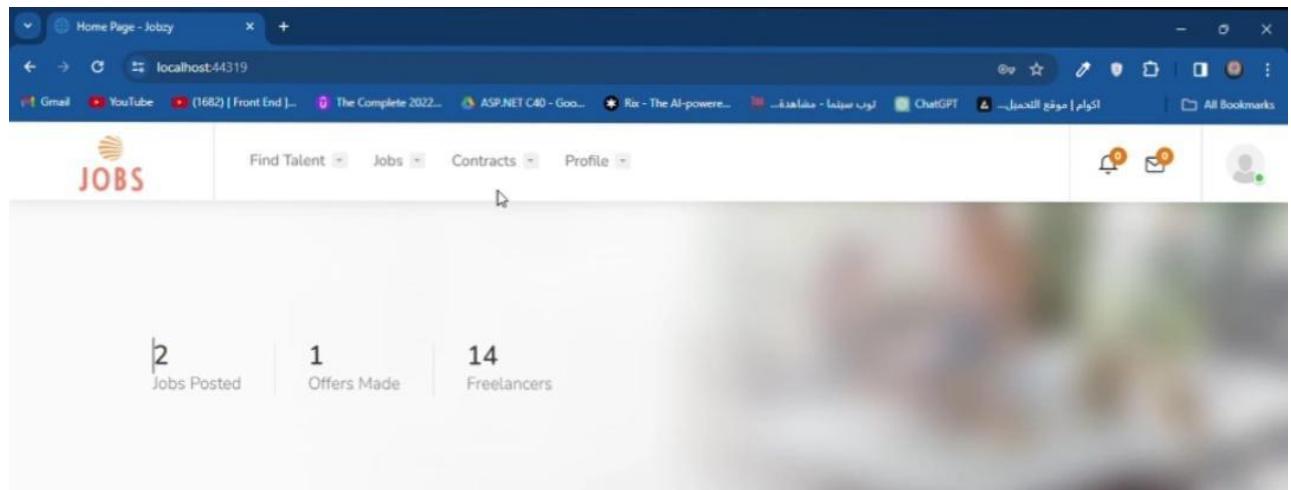
- A dropdown menu for 'Country' set to 'France'.
- A field for 'First Name' containing 'Ziad'.
- A field for 'Last Name' containing 'Mohamed'.
- A field for 'Username' containing 'ziad11'.
- A field for 'Email Address' with a placeholder 'Email Address' and a file input field.
- A password field with the placeholder '*****'.

At the bottom of the form, there are 'Next Step' and 'Cancel' buttons.

Log in



Home page



Add profile

The screenshot shows a web application interface for managing a user profile. The main content area is titled "My Account". It displays the following information:

- First Name: Ahmed
- Last Name: Nabil
- Account Type: Freelancer
- Nationality: Germany

The sidebar on the left contains the following navigation items:

- Main: Dashboard, Messages
- Manage: Offers (My Offers), Contracts (My Contracts)
- Account: Settings

Freelancer profile

- Jobzy

localhost:44319/Users/AllFreelancers

Gmail YouTube (1682) [Front End]... The Complete 2022... ASP.NET C# - Go... Rx - The AI-power... دوام | موقع المعلم... ChatGPT All Bookmarks

JOBS Find Talent Jobs Contracts Profile

Sort by Newest

Rating 0

Name Search by name

Ahmed Nabil .NET Developer

Location Germany Jobs Done 0 Job Success 0% View Profile

Eslam Osama

Location Germany Jobs Done 0 Job Success 0% View Profile

Newest

Not rated yet

Not rated yet

View Profile

View Profile

Message&chat

The screenshot shows a web browser window with the URL `localhost:44319/Messages/Conversation/acct_10XavOFmlPgjWCEk`. The page is titled "Jobzy". The main content area displays a conversation between the user and three other individuals: Ziad Mohamed, Khaled Ahmed, and ahmed nabil. The messages are as follows:

- Ziad Mohamed: 24 seconds ago - asdasd
- Khaled Ahmed: 22 days ago - hhhhhh
- ahmed nabil: one month ago - You: what are you doing now?
- Ziad Mohamed: hello
- ahmed nabil: asdard

The sidebar on the left includes sections for Main (Dashboard, Messages), Manage (Jobs, My Jobs, Add a Job), Contracts (Contracts, My Contracts), and Account. The footer contains the text "© 2024 Jobzy".

localhost:44319/Offers/MyOffers

JOBS

Find Work | Offers | Contracts | Profile |

Make an Offer

Offer Requirements

Job Description

\$ 400

5

Send →

Make an Offer →

Summary

Employer Location

Make an offer

Accept offer

localhost:44319/Offers/All/6db7e02a-da9f-44d7-9c86-e2e7bba1f3c1

Manage Offers - Jobsy

JOBS

Find Talent | Jobs | Contracts | Profile |

Main

Dashboard

Messages

Manage

Jobs

My Jobs

Add a Job

Contracts

My Contracts

Account

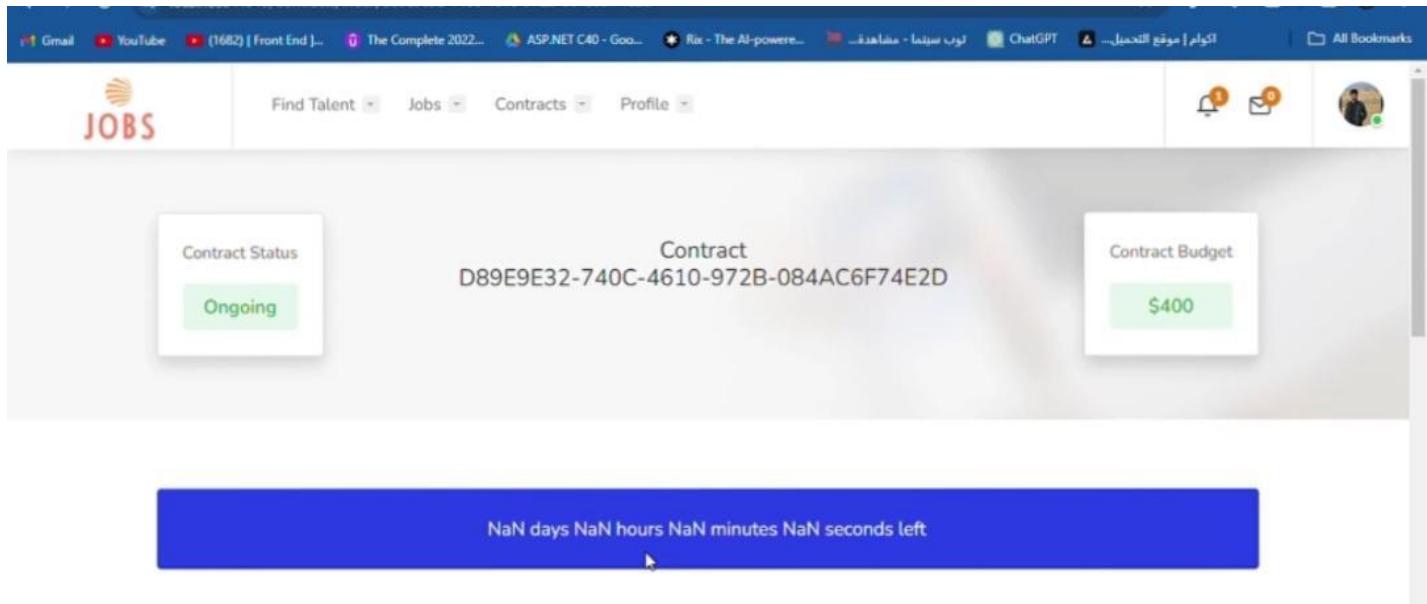
Accept Offer

Accept Offer From Ahmed Nabil for

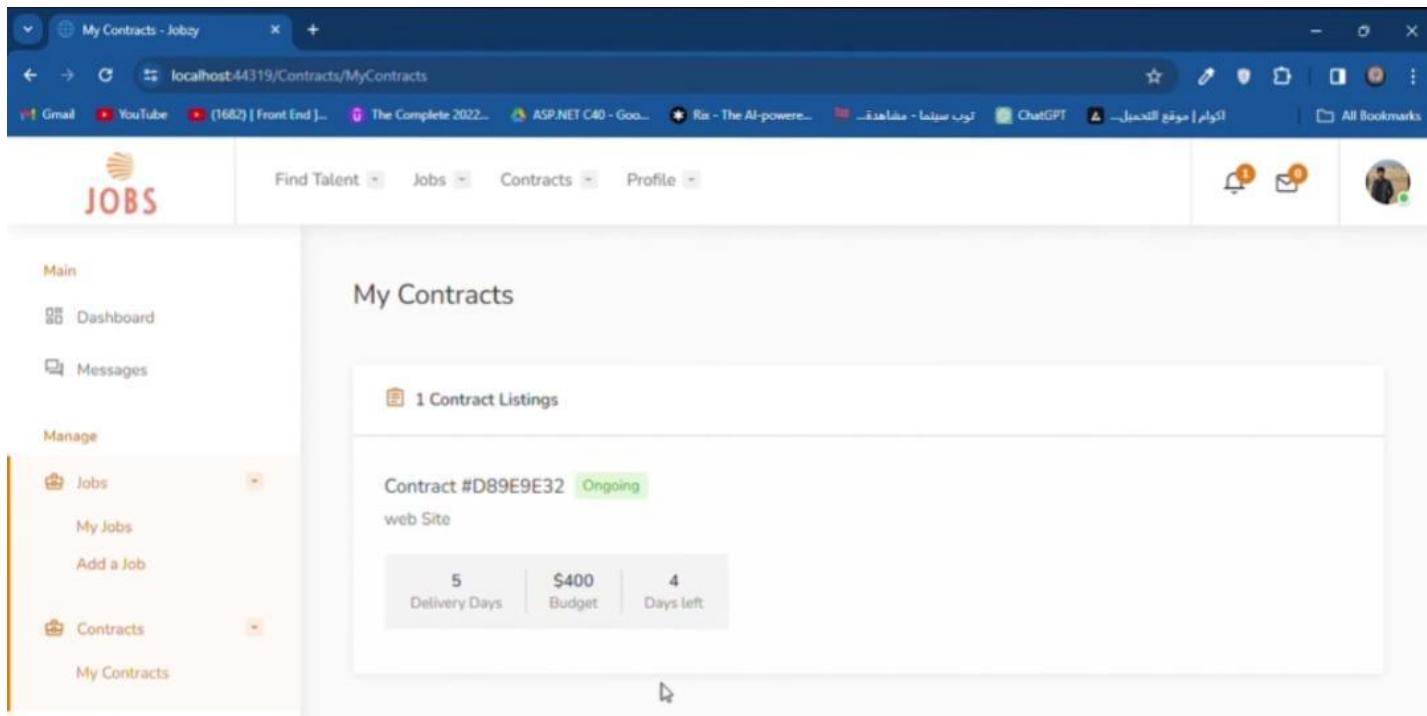
\$400

Accept

Contract

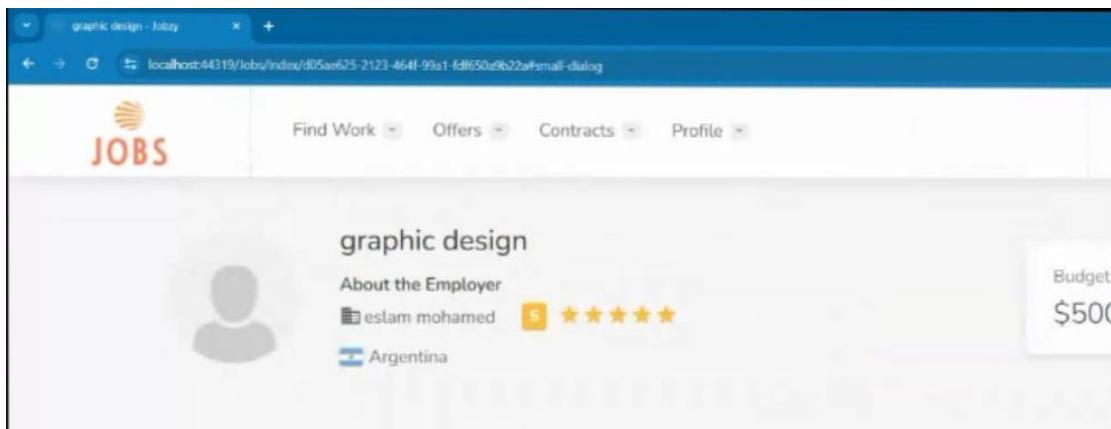


A screenshot of the Jobzy contract dashboard. At the top center, it displays "Contract D89E9E32-740C-4610-972B-084AC6F74E2D". To the left, a box shows "Contract Status Ongoing". To the right, a box shows "Contract Budget \$400". Below this, a large blue bar contains placeholder text: "NaN days NaN hours NaN minutes NaN seconds left". The browser's address bar shows "localhost:44319/Contracts/MyContracts". The left sidebar includes links for Main, Dashboard, Messages, Manage (Jobs, My Jobs, Add a Job), and Contracts (My Contracts).



A screenshot of the Jobzy contract listing page. It shows a single contract entry: "Contract #D89E9E32 Ongoing web Site". Below the entry, a summary box shows "5 Delivery Days", "\$400 Budget", and "4 Days left". The browser's address bar shows "localhost:44319/Contracts/MyContracts". The left sidebar includes links for Main, Dashboard, Messages, Manage (Jobs, My Jobs, Add a Job), and Contracts (My Contracts).

View rate



Search

A screenshot of a web browser window showing search results for "web .net" on the JOBS platform. The results list a job for "Osama Abd Elsalam" in Greece, posted 23 days ago, with a budget of \$500. The search interface includes fields for "Job Title" and "Category", and an orange "Search" button. The top navigation bar is visible, along with browser bookmarks and notifications.

Add job

The screenshot shows a web application interface for adding a new job. On the left, there's a sidebar with navigation links: Main (Dashboard, Messages), Manage (Jobs - My Jobs, Add a Job; Contracts - My Contracts), and Account. The main content area is titled 'Add a Job'. It contains a 'Job Submission Form' with fields for 'Job Title' (containing 'web'), 'Job Category' (set to 'IT & Networking'), and 'Budget' (\$500 USD). A note below the title field states: 'Title must be between 5 and 30 characters long.' The 'Job Description' field contains the text 'dlkamda,amdamd'.

Job

The screenshot shows a web application interface for searching jobs. At the top, there are filters: 'Find Work' (Offers, Contracts, Profile) and a user profile icon. Below these are sections for 'Sort by' (Newest), 'Job Title' (Search by job title), and 'Category' (Search by Category). The main content area displays a single job listing for 'Osama Abd Elsalam' with the job title 'web .net'. The listing includes a profile picture, location 'Greece', category 'Software Development', budget '\$500', and a timestamp 'Posted 23 days ago'. Navigation arrows are visible at the bottom right of the listing.

Edit freelancer profile

This screenshot shows the 'Edit freelancer profile' page. The top navigation bar includes links for Gmail, YouTube, The Complete 2022..., ASP.NET C# - Go... Rix - The AI-power... ChatGPT, and All Bookmarks. The main interface features a sidebar with 'Main' (Dashboard, Messages), 'Manage' (Offers, My Offers, Contracts, My Contracts), and 'Account' (Settings). The central area displays 'My Account' with fields for First Name (Ahmed), Last Name (Nabil), Account Type (Freelancer), and Nationality (Germany). A placeholder image for the avatar is shown with a 'Change Avatar' button.

Edit Clint profile

This screenshot shows the 'Edit Clint profile' page. The top navigation bar includes links for Gmail, YouTube, The Complete 2022..., ASP.NET C# - Go... Rix - The AI-power... ChatGPT, and All Bookmarks. The main interface features a sidebar with 'Main' (Dashboard, Messages), 'Manage' (Jobs, My Jobs, Add a Job, Contracts, My Contracts), and 'Account' (Settings). The central area displays 'My Account' with fields for First Name (Ziad), Last Name (Mohamed), Account Type (Employer), and Nationality (France). A placeholder image for the avatar is shown with a 'Change Avatar' button.

Menu

The screenshot shows a web browser window with the URL `localhost:44319/jobs/All`. The browser's toolbar includes links for Gmail, YouTube, and ChatGPT, along with a bookmark for 'الدوم | موقع التحفيظ'.

The main content area displays a user profile for 'Ahmed Nabil' (Freelancer). The profile includes a small profile picture, a bell icon with a '0' notification, an envelope icon with a '0' notification, and a green circular badge.

A dropdown menu is open from the profile picture, containing the following options:

- Dashboard
- Settings
- Logout** (The 'Logout' option is highlighted with a yellow background and a hand cursor icon).

A large rectangular box highlights the 'Logout' button.

Software architecture .Sign up

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



</div>



<div class="row">
        <div class="col-12 offset-xl-3">
            <!-- Welcome Text -->
            <div class="welcome-text">
                <h3 style="font-size: 26px;">Let's create your account!</h3>
                <span>Already have an account? <a href="Log In">Log In!</a></span>
            </div>

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

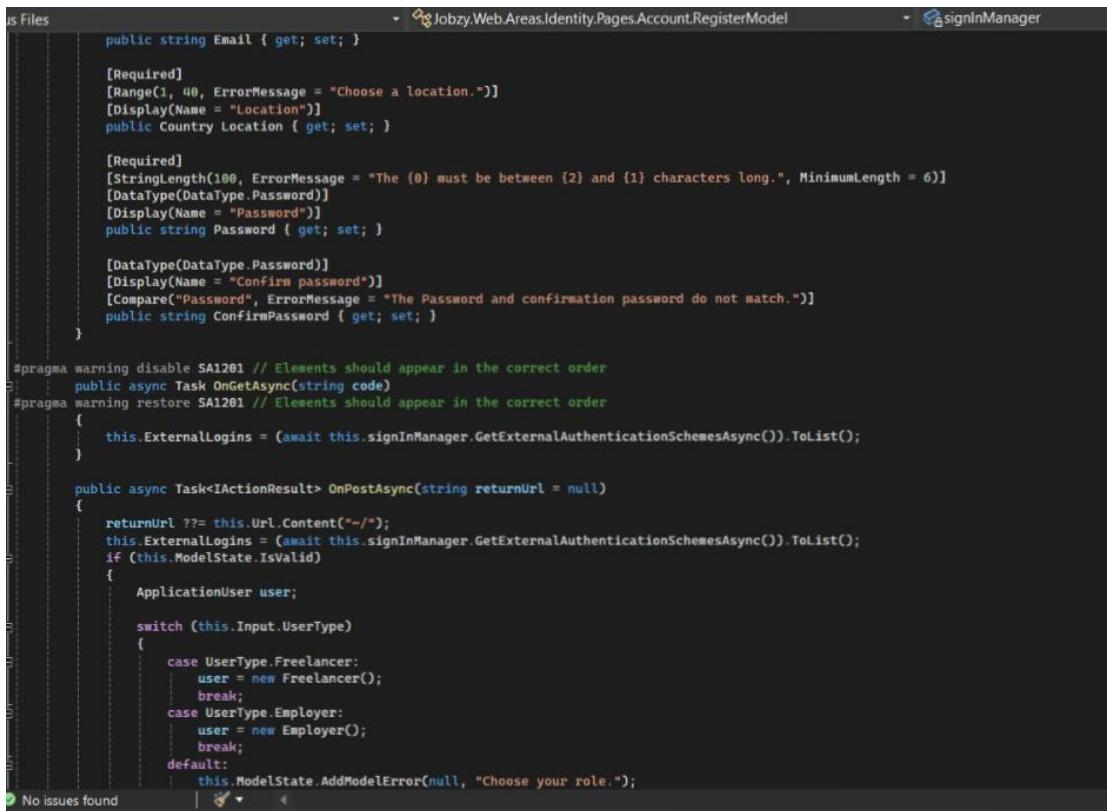
            <!-- Form -->
            <form asp-route-returnUrl="@Model.ReturnUrl" method="post" id="register-form">
                <div class="account-type">
                    <div>
                        <input asp-for="Input.UserType" value="@UserType.Freelancer" type="radio" id="freelancer-radio" class="account-type-radio" />
                        <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>
            </form>
        </div>
    </div>


```



```
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.");
            }
        }
    }
}
```

```
    - 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/v1627979827/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 S11117 // Parameters should be on same line or separate lines
        $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}>clicking here</a>.");
warning restore S11117 // 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 <a href='{HtmlEncoder.Default.Encode(callbackUrl)}>clicking here</a>.");
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
    </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-right"></i>
</button>
</div>

```

```
us Files Jobzy.Web.Areas.Identity.Pages.Account.Register.cs
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;

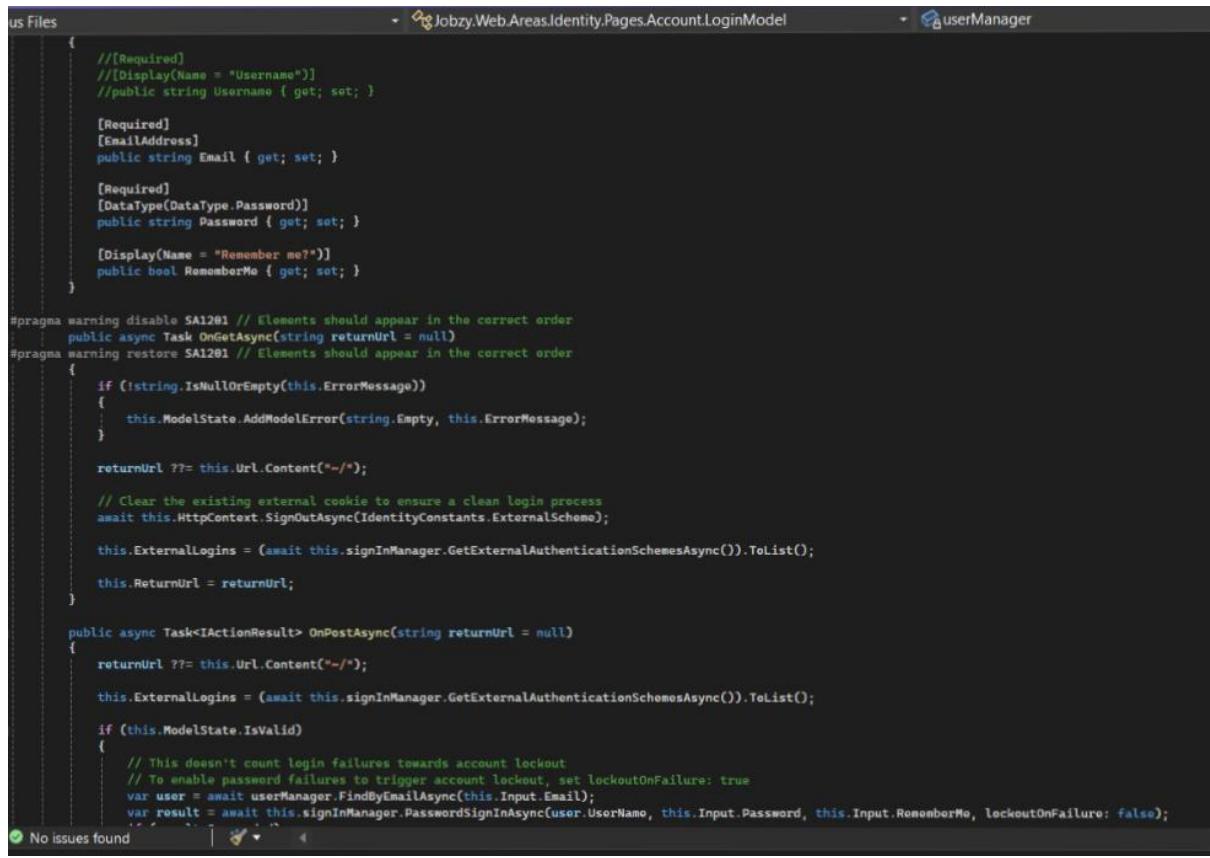
        public RegisterModel(
            IFreelancePlatform freelancePlatform,
            UserManager<ApplicationUser> userManager,
            SignInManager<ApplicationUser> signInManager,
            ILogger<RegisterModel> logger,
            IEmailSender emailSender)
        {
            this.freelancePlatform = freelancePlatform;
            this.userManager = userManager;
            this.signInManager = signInManager;
            this.logger = logger;
        }
    }
}
```

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 class="margin-bottom-100"></div>
<partial name="BigFooterPartial" />
@section Scripts {
  <partial name="ValidationScriptsPartial" />
}
```

No issues found | ln 22 Ch 1



```
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
```

```
cellaneous Files + Jobzy.Web.Areas.Identity.Pages.Account.LoginMode
1  namespace Jobzy.Web.Areas.Identity.Pages.Account
2  {
3      using System.Collections.Generic;
4      using System.ComponentModel.DataAnnotations;
5      using System.Linq;
6      using System.Threading.Tasks;
7
8      using Jobzy.Data.Models;
9      using Microsoft.AspNetCore.Authentication;
10     using Microsoft.AspNetCore.Authorization;
11     using Microsoft.AspNetCore.Identity;
12     using Microsoft.AspNetCore.Mvc;
13     using Microsoft.AspNetCore.Mvc.RazorPages;
14     using Microsoft.Extensions.Logging;
15
16     [AllowAnonymous]
17     #pragma warning disable SA1649 // File name should match first type name
18     public class LoginModel : PageModel
19     #pragma warning restore SA1649 // File name should match first type name
20     {
21         private readonly UserManager< ApplicationUser> userManager;
22         private readonly SignInManager< ApplicationUser> signInManager;
23         private readonly ILogger< LoginModel > logger;
24
25         public LoginModel(
26             SignInManager< ApplicationUser > signInManager,
27             ILogger< LoginModel > logger,
28             UserManager< ApplicationUser > userManager)
29         {
30             this.userManager = userManager;
31             this.signInManager = signInManager;
32             this.logger = logger;
33         }
34
35         [BindProperty]
36         public InputModel Input { get; set; }
37
38         public IList< AuthenticationScheme > ExternalLogins { get; set; }
39
40         public string ReturnUrl { get; set; }
41
42         [TempData]
43         public string ErrorMessage { get; set; }
44
45         public class InputModel
46         {
47             // [Required]
48             // [Display( Name = " Username" )]
```

```
es          - OG 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();
        }
    }

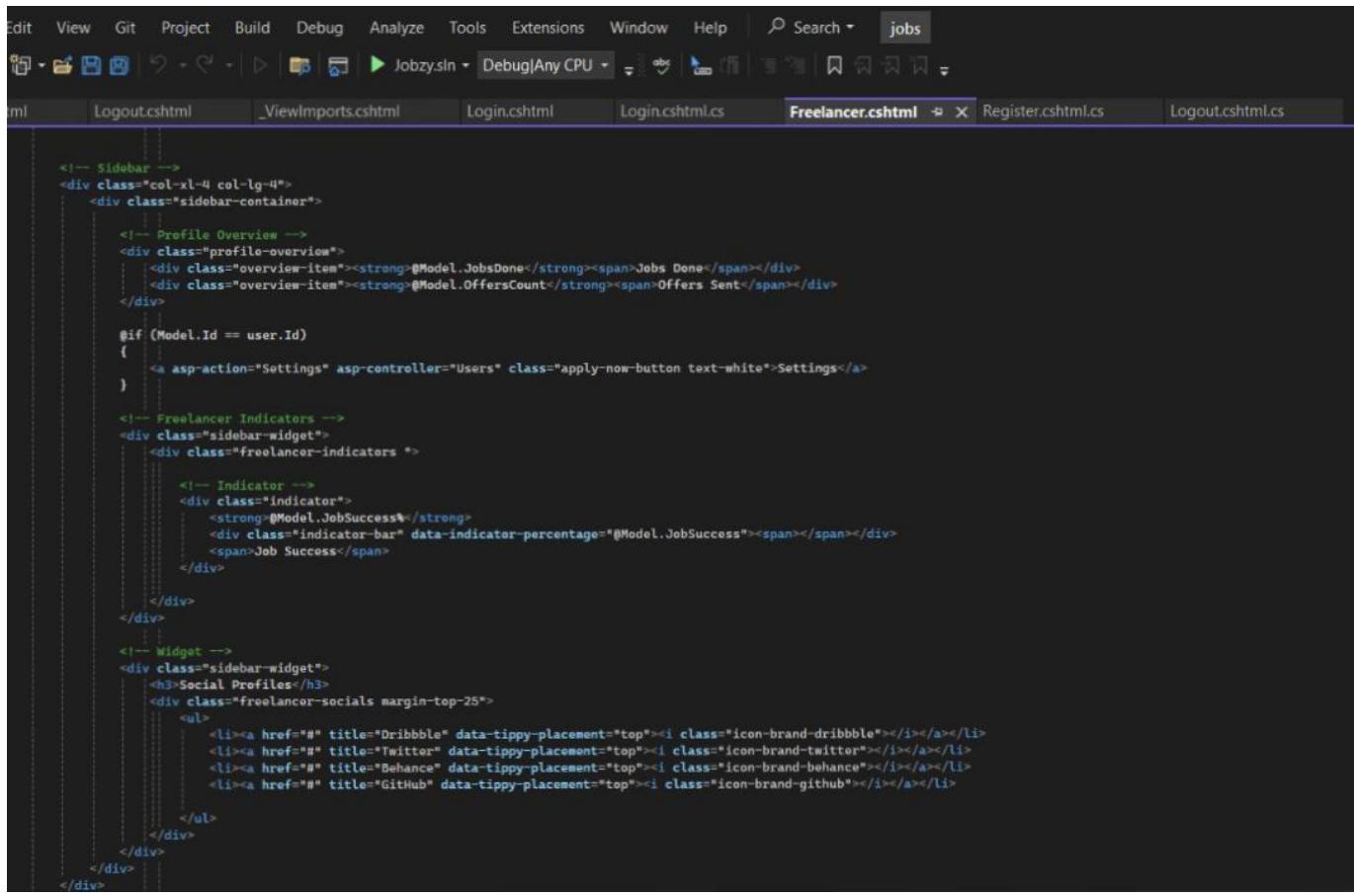
    // If we got this far, something failed, redisplay form
    return this.Page();
}

0 issues found
```

Freelancer profile

```
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml.cs
1 @using Jobzy.Web.ViewModels.Users.Freelancers
2 @using Jobzy.Web.ViewModels.Reviews
3 @using Jobzy.Data
4 @using Microsoft.AspNetCore.Identity
5 @inject SignInManager< ApplicationUser > signInManager
6 @inject UserManager< ApplicationUser > userManager
7 @model FreelancerViewModel
8
9 @{
10     var user = await userManager.GetUserAsync(this.User);
11 }
12
13 <!-- Titlebar -->
14 <div class="single-page-header freelancer-header" data-background-image="/images/single-freelancer.jpg">
15     <div class="container">
16         <div class="row">
17             <div class="col-md-12">
18                 <div class="single-page-header-inner">
19                     <div class="left-side">
20                         <div class="header-avatar"></div>
21                         <div class="header-details">
22                             <h3>
23                                 @Model.FirstName @Model.LastName
24
25                                 @if (Model.TagName == null && Model.Id == user.Id)
26                                 {
27                                     <span><a href="#" asp-action="Settings" asp-controller="Users" data-id="Add">Add</a> a tag name</span>
28                                 }
29                                 else
30                                 {
31                                     <span>@Model.TagName</span>
32                                 }
33                             </h3>
34                             <ul>
35                                 @if (Model.Reviews.Count() > 0)
36                                 {
37                                     <li><div class="star-rating" data-rating="@Model.AverageRating"></div></li>
38                                 }
39                                 else
40                                 {
41                                     <li><div class="not-rated">Not rated yet</div></li>
42                                 }
43                             <li> @Model.LocationToString</li>
44                         </ul>
45                     </div>
46                 </div>
47             </div>
48         </div>
49     </div>
50 </div>
51
52 No issues found
53
54 <!-- Boxed List / End -->
55 <!-- Boxed List -->
56 <div class="boxed-list margin-bottom-60">
57     @if (Model.Contracts.Count() > 0)
58     {
59         <div class="boxed-list-headline">
60             <h3><i class="icon-material-outline-business"></i> Employment History</h3>
61         </div>
62         <ul class="boxed-list-ul">
63             @foreach (var contract in Model.Contracts)
64             {
65                 <li>
66                     <div class="boxed-list-item">
67                         <!-- Avatar -->
68                         <div class="item-image">
69                             
70                         </div>
71
72                         <!-- Content -->
73                         <div class="item-content">
74                             <h4>@contract.JobTitle</h4>
75                             <div class="item-details margin-top-7">
76                                 <div class="detail-item"><a href="#" asp-action="Employer" asp-controller="Users" data-route-id="@contract.EmployerId" data-id="Edit" class="icon-material-outline-business"></a> @contract.CreatedOnFormatted - @contract.CompletedOnFormatted</div>
77                             </div>
78                             <div class="item-description">
79                                 <p>@contract.JobDescription</p>
80                             </div>
81                         </div>
82                     </div>
83                 </li>
84             }
85         </ul>
86     }
87     else
88     {
89         <div class="boxed-list-headline">
90             <h3><i class="icon-material-outline-business"></i> No Employment History</h3>
91         </div>
92     }
93 </div>
94 <!-- Boxed List / End -->
95
96
97 No issues found
98
99 <!-- Boxed List / End -->
```

```
<!-- 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">
                            
                        </div>
                        <!-- Content -->
                        <div class="item-content">
                            <h4>@contract.JobTitle</h4>
                            <div class="item-details margin-top-7">
                                <div class="detail-item"><a href="#" asp-action="Employer" asp-controller="Users" data-route-id="@contract.EmployerId" data-id="Edit" class="icon-material-outline-business"></a> @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 -->
<!-- Boxed List / End -->
```



The screenshot shows the Visual Studio IDE interface with the 'Freelancer.cshtml' file open in the editor. The file contains C# Razor code for a sidebar section. It includes sections for 'Profile Overview', 'Freelancer Indicators', and 'Social Profiles'. The 'Social Profiles' section lists links to Dribbble, Twitter, Behance, and GitHub, each with a corresponding icon.

```
<!-- Sidebar -->
<div class="col-1x-4 col-lg-4">
  <div class="sidebar-container">

    <!-- Profile Overview -->
    <div class="profile-overview">
      <div class="overview-item"><strong>@Model.JobsDone</strong><span>Jobs Done</span></div>
      <div class="overview-item"><strong>@Model.OffersCount</strong><span>Offers Sent</span></div>
    </div>

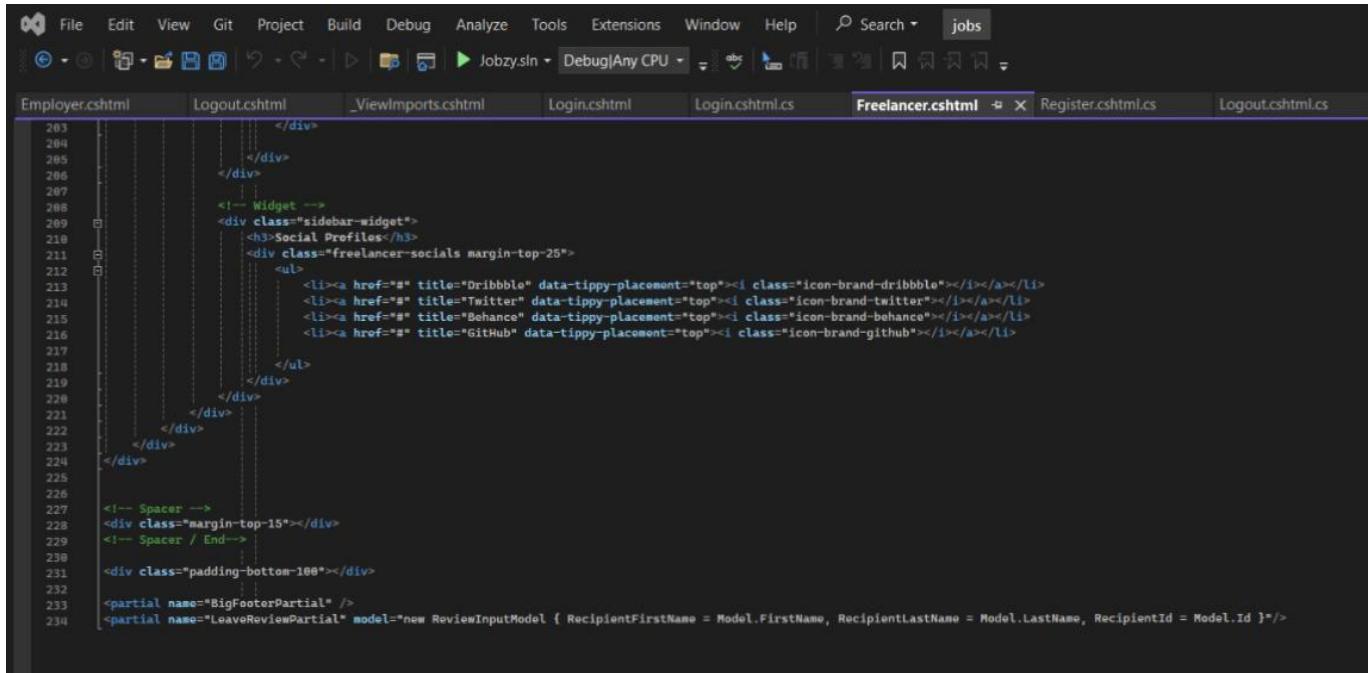
    @if (Model.Id == user.Id)
    {
      <a href="#" asp-action="Settings" asp-controller="Users" class="apply-now-button text-white">Settings</a>
    }

    <!-- Freelancer Indicators -->
    <div class="sidebar-widget">
      <div class="freelancer-indicators">

        <!-- Indicator -->
        <div class="indicator">
          <strong>@Model.JobSuccess</strong>
          <div class="indicator-bar" data-indicator-percentage="@Model.JobSuccess"><span></span></div>
          <span>Job Success</span>
        </div>

      </div>
    </div>

    <!-- 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>
```



The screenshot shows the Visual Studio IDE interface with the 'Employer.cshtml' file open in the editor. This file is a partial view that includes parts of the 'Logout.cshtml' and 'Freelancer.cshtml' files. It features a sidebar with social media links (Dribbble, Twitter, Behance, GitHub) and a spacer section at the bottom.

```
203   </div>
204
205   </div>
206
207   <!-- Widget -->
208   <div class="sidebar-widget">
209     <h3>Social Profiles</h3>
210     <div class="freelancer-socials margin-top-25">
211       <ul>
212         <li><a href="#" title="Dribbble" data-tippy-placement="top"><i class="icon-brand-dribbble"></i></a></li>
213         <li><a href="#" title="Twitter" data-tippy-placement="top"><i class="icon-brand-twitter"></i></a></li>
214         <li><a href="#" title="Behance" data-tippy-placement="top"><i class="icon-brand-behance"></i></a></li>
215         <li><a href="#" title="GitHub" data-tippy-placement="top"><i class="icon-brand-github"></i></a></li>
216       </ul>
217     </div>
218   </div>
219
220   </div>
221
222   </div>
223
224 </div>
225
226
227 <!-- Spacer -->
228 <div class="margin-top-15"></div>
229 <!-- Spacer / End-->
230
231 <div class="padding-bottom-100"></div>
232
233 <partial name="BigFooterPartial" />
234 <partial name="LeaveReviewPartial" model="new ReviewInputModel { RecipientFirstName = Model.FirstName, RecipientLastName = Model.LastName, RecipientId = Model.Id }"/>
```

```
<div class="item-content">
    <h4><span>@review.SenderFirstName @review.SenderLastName</span></h4>
    <div class="item-details margin-top-18">
        <div class="star-rating" data-rating="@review.Rating"></div>
        <div class="detail-item"><i class="icon-material-outline-date-range"></i> @review.DateFormatted</div>
    </div>
    <div class="item-description">
        <p>@review.Text</p>
    </div>
</div>
</li>
</ul>
}
else
{
    <div class="boxed-list-headline">
        <h3><i class="icon-material-outline-thumb-up"></i> 0 Reviews</h3>
    </div>

    <ul class="boxed-list-ul">
        <li>
            <div class="boxed-list-item">
                <!-- Content -->
                <div class="item-content">
                    <div class="item-description">
                        <p>Be the first one to leave a review.</p>
                    </div>
                </div>
            </div>
        </li>
    </ul>
}
@if (Model.Id != user.Id)
{
    <div class="centered-button margin-top-35">
        <a href="#small-dialog" class="popup-with-zoom-anim button button-sliding-icon">Leave a Review <i class="icon-material-outline-arrow-right-alt"></i></a>
    </div>
}
</div>

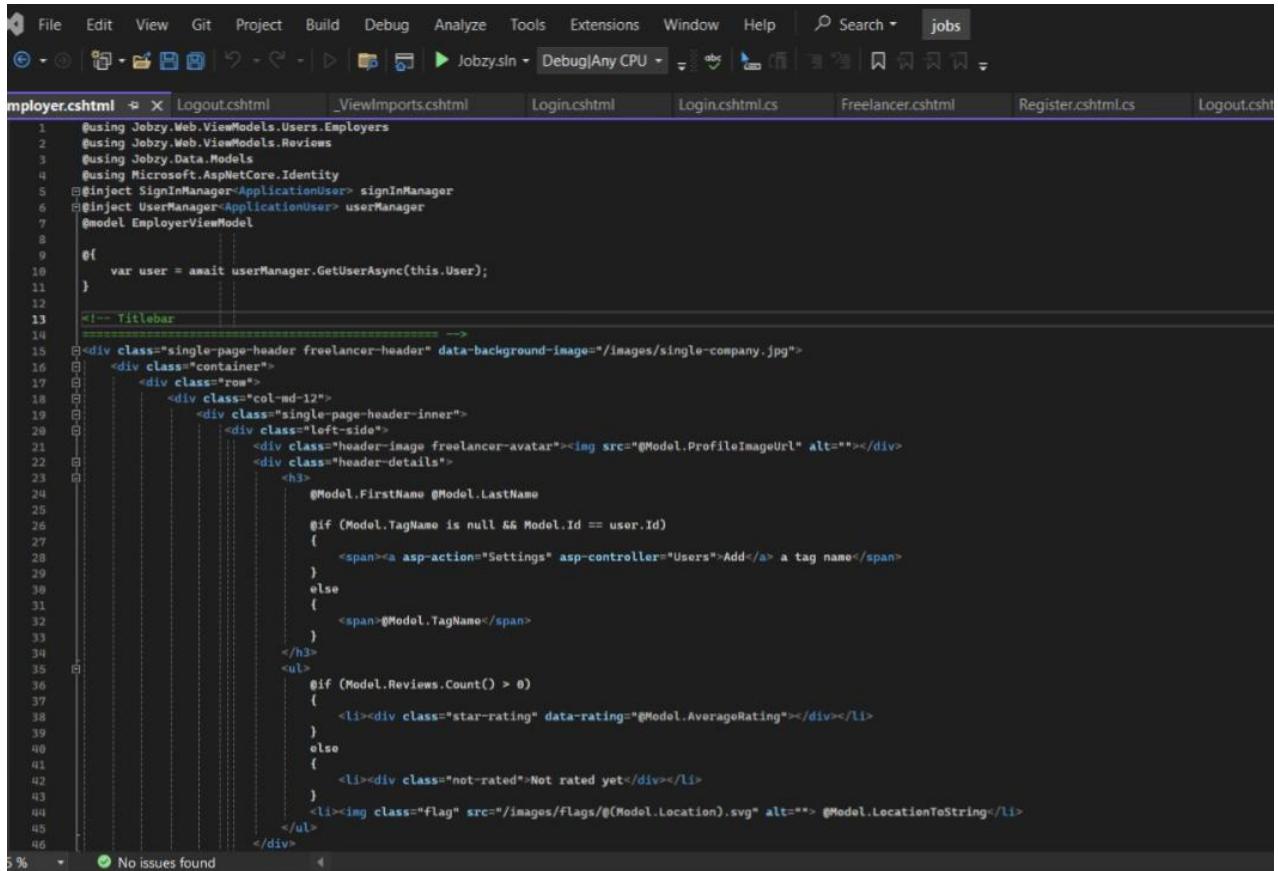
<!-- Boxed List / End -->
<!-- Boxed List -->
<div class="boxed-list margin-bottom-60">
```

```
<div class="item-content">
    <h4><span>@review.SenderFirstName @review.SenderLastName</span></h4>
    <div class="item-details margin-top-18">
        <div class="star-rating" data-rating="@review.Rating"></div>
        <div class="detail-item"><i class="icon-material-outline-date-range"></i> @review.DateFormatted</div>
    </div>
    <div class="item-description">
        <p>@review.Text</p>
    </div>
</div>
</li>
</ul>
}
else
{
    <div class="boxed-list-headline">
        <h3><i class="icon-material-outline-thumb-up"></i> 0 Reviews</h3>
    </div>

    <ul class="boxed-list-ul">
        <li>
            <div class="boxed-list-item">
                <!-- Content -->
                <div class="item-content">
                    <div class="item-description">
                        <p>Be the first one to leave a review.</p>
                    </div>
                </div>
            </div>
        </li>
    </ul>
}
@if (Model.Id != user.Id)
{
    <div class="centered-button margin-top-35">
        <a href="#small-dialog" class="popup-with-zoom-anim button button-sliding-icon">Leave a Review <i class="icon-material-outline-arrow-right-alt"></i></a>
    </div>
}
</div>

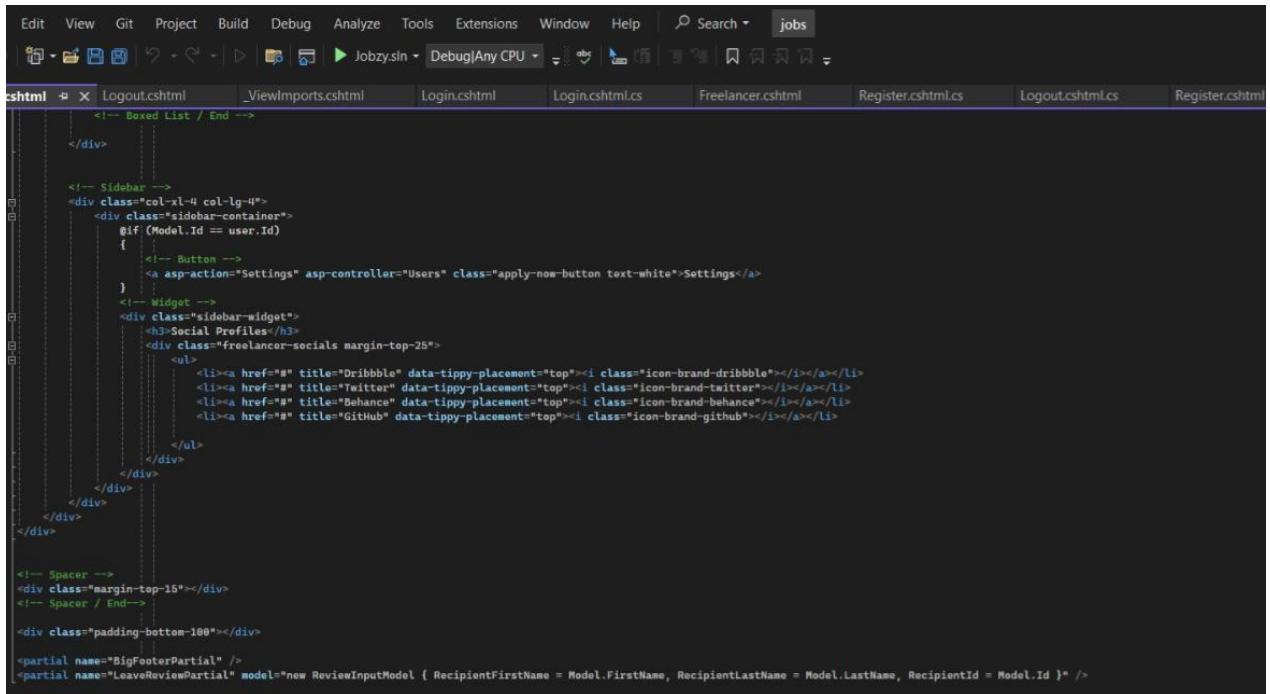
<!-- Page Content -->
<div class="container">
    <div class="row">
        <!-- Content -->
        <div class="col-xl-8 col-lg-8 content-right-offset">
            <!-- Page Content -->
            <div class="single-page-section">
                <h3>About Me</h3>
                @if (Model.Description is null && Model.Id == user.Id)
                {
                    <p class="not-verified-badge"><a href="#" asp-action="Settings" asp-controller="Users">Add</a> a description.</p>
                }
                else
                {
                    <p>@(Model.Description is not null ? Model.Description : "No description yet.")</p>
                }
            </div>
            <!-- Boxed List -->
            <div class="boxed-list margin-bottom-60">
                @if (Model.Reviews.Count() > 0)
                {
                    <div class="boxed-list-headline">
                        <h3><i class="icon-material-outline-thumb-up"></i> @Model.Reviews.Count() Reviews</h3>
                    </div>
                    <ul class="boxed-list-ul">
                        @foreach (var review in Model.Reviews)
                        {
                            <li>
                                <div class="boxed-list-item">
                                    <!-- Content -->
                                    <div class="item-content">
                                        <h4>@review.SenderFirstName @review.SenderLastName</h4>
```

Employer profile



A screenshot of the Visual Studio IDE showing the code editor for the `Employer.cshtml` file. The file contains C# Razor code for displaying an employer's profile. It includes imports for `Jobzy.Web.ViewModels`, `Jobzy.Data.Models`, and `Microsoft.AspNetCore.Identity`. It injects `SignInManager<ApplicationUser>` and `UserManager<ApplicationUser>` into the constructor. The code then checks if the user is authenticated and retrieves the user from the database. It then displays a header with a background image, a left sidebar with the user's profile picture and details, and a main content area showing reviews and social profiles. The code uses Bootstrap's grid system and includes logic for average rating and review counts.

```
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Employer.cshtml Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml
1  @using Jobzy.Web.ViewModels.Users.Employers
2  @using Jobzy.Web.ViewModels.Reviews
3  @using Jobzy.Data.Models
4  @using Microsoft.AspNetCore.Identity
5  @inject SignInManager< ApplicationUser > signInManager
6  @inject UserManager< ApplicationUser > userManager
7  @model EmployerViewModel
8
9  @{
10     var user = await userManager.GetUserAsync(this.User);
11 }
12
13 <!-- Titlebar -->
14 <div class="single-page-header freelancer-header" data-background-image="/images/single-company.jpg">
15     <div class="container">
16         <div class="row">
17             <div class="col-md-12">
18                 <div class="single-page-header-inner">
19                     <div class="left-side">
20                         <header-image freelancer-avatar></div>
21                         <div class="header-details">
22                             <h3>
23                                 @Model.FirstName @Model.LastName
24
25                            @if (Model.TagName is null && Model.Id == user.Id)
26                             {
27                                 <span><a href="#" asp-action="Settings" asp-controller="Users">Add a tag name</a></span>
28                             }
29                             else
30                             {
31                                 <span>@Model.TagName</span>
32                             }
33                         </h3>
34                         <ul>
35                             @if (Model.Reviews.Count() > 0)
36                             {
37                                 <li><div class="star-rating" data-rating="@Model.AverageRating"></div></li>
38                             }
39                             else
40                             {
41                                 <li><div class="not-rated">Not rated yet</div></li>
42                             }
43                         <li> @Model.LocationToString()
44                         </li>
45                     </ul>
46                 </div>
47             </div>
48         </div>
49     </div>
50     <div class="padding-bottom-100"></div>
51     <partial name="BigFooterPartial" />
52     <partial name="LeaveReviewPartial" model="new ReviewInputModel { RecipientFirstName = Model.FirstName, RecipientLastName = Model.LastName, RecipientId = Model.Id }" />
53
54 %>
55 No issues found
```



A screenshot of the Visual Studio IDE showing the code editor for the `Logout.cshtml` file. The file contains C# Razor code for a logout page. It includes imports for `Jobzy.Web` and `Jobzy.Data`. The code uses a `Boxed List` component to wrap the content. It then handles the logout process by signing out the user and redirecting them to the login page. The code also includes logic for social profiles and a footer partial view.

```
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Logout.cshtml _ViewImports.cshtml Login.cshtml Login.cshtml.cs Freelancer.cshtml Register.cshtml.cs Logout.cshtml
1  <!-- Boxed List / End -->
2
3  <!-- Sidebar -->
4  <div class="col-1x-4 col-lg-4">
5      <div class="sidebar-container">
6          @if (Model.Id == user.Id)
7          {
8              <!-- Button -->
9              <a href="#" asp-action="Settings" asp-controller="Users" class="apply-now-button text-white">Settings</a>
10         <!-- Widget -->
11         <div class="sidebar-widget">
12             <h3>Social Profiles</h3>
13             <div class="freelancer-socials margin-top-25">
14                 <ul>
15                     <li><a href="#" title="Dribbble" data-tippy-placement="top"><i class="icon-brand-dribbble"></i></a></li>
16                     <li><a href="#" title="Twitter" data-tippy-placement="top"><i class="icon-brand-twitter"></i></a></li>
17                     <li><a href="#" title="Behance" data-tippy-placement="top"><i class="icon-brand-behance"></i></a></li>
18                     <li><a href="#" title="GitHub" data-tippy-placement="top"><i class="icon-brand-github"></i></a></li>
19                 </ul>
20             </div>
21         </div>
22     </div>
23 </div>
24
25 <!-- Spacer -->
26 <div class="margin-top-15"></div>
27 <!-- Spacer / End-->
28
29 <div class="padding-bottom-100"></div>
30
31 <partial name="BigFooterPartial" />
32 <partial name="LeaveReviewPartial" model="new ReviewInputModel { RecipientFirstName = Model.FirstName, RecipientLastName = Model.LastName, RecipientId = Model.Id }" />
```

The screenshot shows the Visual Studio IDE interface with the 'Logout.cshtml' file open. The top menu bar includes 'Edit', 'View', 'Git', 'Project', 'Build', 'Debug', 'Analyze', 'Tools', 'Extensions', 'Window', 'Help', and a search bar. Below the menu is a toolbar with various icons. The tabs at the top show 'Logout.cshtml', '_ViewImports.cshtml', 'Login.cshtml', 'Login.cshtml.cs', 'Freelancer.cshtml', 'Register.cshtml.cs', and 'Logout.cshtml.cs'. The main code editor area contains C# Razor code for a login page. A status bar at the bottom indicates 'No issues found'.

```
<!-- Details -->
<div class="job-listing-description">
    <h3 class="job-listing-title">@job.Title</h3>

    <!-- Job Listing Footer -->
    <div class="job-listing-footer">
        <ul>
            <li><i class="icon-material-outline-location-on"></i> @job.LocationToString</li>
            <li><i class="icon-material-outline-business-center"></i> @job.CategoryName</li>
            <li><i class="icon-material-outline-access-time"></i> Posted @job.DateFormatted</li>
        </ul>
    </div>
</div>

</a>
}
else
{
    <div class="boxed-list-headline">
        <h3><i class="icon-material-outline-business-center"></i> 0 Open Positions</h3>
    </div>
}
<!-- Boxed List / End -->
<!-- Boxed List -->
<div class="boxed-list margin-bottom-60">
    @if (Model.Reviews.Count() > 0)
    {
        <div class="boxed-list-headline">
            <h3><i class="icon-material-outline-thumb-up"></i> @Model.Reviews.Count() Reviews</h3>
        </div>
        <ul class="boxed-list-ul">
            #foreach (var review in Model.Reviews)
            {
                <li>
                    <div class="boxed-list-item">
                        <!-- Content -->
                        <div class="item-content">
                            <h4><span>@review.SenderFirstName @review.SenderLastName</span></h4>
                            <div class="item-details margin-top-10">
                                <div class="star-rating" data-rating="@review.Rating"></div>
                            
```

The screenshot shows the Visual Studio IDE interface with the 'Logout.cshtml' file open. The top menu bar includes 'Edit', 'View', 'Git', 'Project', 'Build', 'Debug', 'Analyze', 'Tools', 'Extensions', 'Window', 'Help', and a search bar. Below the menu is a toolbar with various icons. The tabs at the top show 'Logout.cshtml', '_ViewImports.cshtml', 'Login.cshtml', 'Login.cshtml.cs', 'Freelancer.cshtml', 'Register.cshtml.cs', and 'Logout.cshtml.cs'. The main code editor area contains C# Razor code for a login page. A status bar at the bottom indicates 'No issues found'.

```
<div class="star-rating" data-rating="@review.Rating"></div>
<div class="detail-item"><i class="icon-material-outline-date-range"></i> @review.DateFormatted</div>
</div>
<div class="item-description">
    <p>@review.Text</p>
</div>
</div>
</li>
}
</ul>
}
else
{
    <div class="boxed-list-headline">
        <h3><i class="icon-material-outline-thumb-up"></i> 0 Reviews</h3>
    </div>
    @if (Model.Id != user.Id)
    {
        <ul class="boxed-list-ul">
            <li>
                <div class="boxed-list-item">
                    <!-- Content -->
                    <div class="item-content">
                        <div class="item-description">
                            <p>Be the first one to leave a review.</p>
                        </div>
                    </div>
                </div>
            </li>
        </ul>
    }
    @if (Model.Id != user.Id)
    {
        <div class="centered-button margin-top-35">
            <a href="#small-dialog" class="popup-with-zoom-anim button button-sliding-icon">Leave a Review <i class="icon-material-outline-arrow-right-alt"></i></a>
        </div>
    }
}
<!-- Boxed List / End -->
```

```
47      </div>
48      </div>
49      </div>
50    </div>
51  </div>
52</div>
53
54
55  <!-- Page Content -->
56  <div class="container">
57    <div class="row">
58
59      <!-- Content -->
60      <div class="col-8 col-lg-8 content-right-offset">
61
62        <div class="single-page-section">
63          <h3 class="margin-bottom-25">About The Employer</h3>
64
65         @if (Model.Description == null && Model.Id == user.Id)
66          {
67            <p class="not-verified-badge"><a href="#" asp-action="Settings" asp-controller="Users">Add</a> a description.</p>
68          }
69          else
70          {
71            <p>@(Model.Description != null ? Model.Description : "No description yet.")</p>
72          }
73      </div>
74
75      <!-- Boxed List -->
76      <div class="boxed-list margin-bottom-60">
77        @if (Model.OpenJobs.Count > 0)
78        {
79
80          <div class="boxed-list-headline">
81            <h3><i class="icon-material-outline-business-center"></i> @Model.OpenJobs.Count Open Positions</h3>
82          </div>
83
84          <div class="listings-container compact-list-layout">
85            @foreach (var job in Model.OpenJobs)
86            {
87
88              <!-- Job Listing -->
89              <a href="#" asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing">
90
91                <!-- Job Listing Details -->
92                <div class="job-listing-details">
```

Sitting (Edit profile , Upload image& Delete image)

Screenshot of the Visual Studio code editor showing the file Privacy.cshtml. The code is a C# Razor view for a job search application. It includes imports for Jobzy.Data.Models, Microsoft.AspNetCore.Identity, and SignInManager<ApplicationUser>. The view defines ViewData["Title"] as "Home Page". It contains sections for an intro banner (with a background image and headline) and stats (job counts). The code uses Bootstrap classes like .row, .col-md-12, and .margin-top-45.

```

1  @using Jobzy.Data.Models
2  @using Microsoft.AspNetCore.Identity
3  @inject SignInManager< ApplicationUser > signInManager
4
5  #model Jobzy.Web.ViewModels.Home.HomeViewModel
6
7  @{
8      this.ViewData["Title"] = "Home Page";
9  }
10
11  <!-- Intro Banner
12  =====>
13  <!-- add class="disable-gradient" to enable consistent background overlay -->
14  <div class="intro-banner" data-background-image="/images/home-background.jpg">
15  <div class="container">
16
17     @if (!this.signInManager.IsSignedIn(this.User))
18      {
19          <!-- Intro Headline -->
20          <div class="row">
21              <div class="col-md-12">
22                  <div class="banner-headline">
23                      <h3>
24                          <strong>Hire freelancers or be hired for any job, any time.</strong>
25                            

26                          <span>Thousands of small businesses use <strong>Jobs</strong> to turn their ideas into reality.</span>
27                      </h3>
28                  </div>
29              </div>
30          </div>
31      }
32
33      <!-- Stats -->
34      <div class="row">
35          <div class="col-md-12">
36              <ul class="intro-stats margin-top-45 hide-under-992px">
37                  <li>
38                      <strong class="counter">@Model.JobsCount</strong>
39                      <span>Jobs Posted</span>
40                  </li>
41                  <li>
42                      <strong class="counter">@Model.OffersCount</strong>
43                      <span>Offers Made</span>
44                  </li>
45                  <li>
46                      <strong class="counter">@Model.FreelancersCount</strong>
47                      <span>Freelancers</span>
48                  </li>
49              </ul>
50          </div>
51      </div>
52  </div>
53
54  75% No issues found 4

```

Screenshot of the Visual Studio code editor showing the file Index.cshtml. This is a C# Razor view for displaying job listings. It starts with a section header and a link to browse all jobs. The main content is a foreach loop over Model.Jobs, which contains job listing details, footer information, and an apply button. The code uses Bootstrap classes like .row, .col-xl-12, and .listings-container.

```

1  <div class="row">
2      <div class="col-xl-12">
3
4          <!-- Section Heading -->
5          <div class="section-headline margin-top-0 margin-bottom-35">
6              <h3>Featured Jobs</h3>
7              <a href="#" asp-action="All" asp-controller="Jobs" class="headline-link">Browse All Jobs</a>
8          </div>
9
10         <!-- Jobs Container -->
11         <div class="listings-container compact-list-layout margin-top-35">
12             &#foreach (var job in Model.Jobs)
13             {
14
15                 <!-- Job Listing -->
16                 <a href="#" asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing with-apply-button">
17
18                     <!-- Job Listing Details -->
19                     <div class="job-listing-details">
20
21                         <!-- Logo -->
22                         <div class="job-listing-company-logo">
23                             
24                         </div>
25
26                         <!-- Details -->
27                         <div class="job-listing-description">
28                             <h3 class="job-listing-title">@job.Title</h3>
29
30                             <!-- Job Listing Footer -->
31                             <div class="job-listing-footer">
32                                 <ul>
33                                     <li><i class="icon-material-outline-business"></i> @job.EmployerFirstName @job.EmployerLastName</li>
34                                     <li><i class="icon-material-outline-location-on"></i> @job.EmployerLocationToString</li>
35                                     <li><i class="icon-material-outline-business-center"></i> @job.CategoryName</li>
36                                     <li><i class="icon-material-outline-access-time"></i> @job.Posted @job.TimeAgo</li>
37                                 </ul>
38                             </div>
39
40                         <!-- Apply Button -->
41                         <span class="list-apply-button ripple-effect">Apply Now</span>
42                     </div>
43                 </a>
44             }
45
46             <!-- Jobs Container / End -->
47         </div>
48     </div>
49
50  
```

```
<!-- Job Listing Footer -->
<div class="job-listing-footer">
    <ul>
        <li><i class="icon-material-outline-business"></i> @job.EmployerFirstName @job.EmployerLastName</li>
        <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-access-time"></i> Posted @job.TimeAgo</li>
    </ul>
</div>

<!-- Apply Button -->
<span class="list-apply-button ripple-effect">Apply Now</span>

}
</div>
<!-- Jobs Container / End -->
</div>
</div>
<!-- Featured Jobs / End -->
}
<div class="margin-bottom-100"></div>
<partial name="BigFooterPartial" />
```

```
</div>
</div>
<!-- Icon Boxes / End -->
<div class="padding-bottom-100"></div>
}
else
{
    if (this.User.IsInRole("Employer") && Model.Freelancers.Any())
    {
        <!-- Highest Rated Freelancers -->
        <div class="section gray padding-top-65 padding-bottom-70 full-width-carousel-fix">
            <div class="container">
                <div class="row">
                    <div class="col-xl-12">
                        <!-- Section Headline -->
                        <div class="section-headline margin-top-0 margin-bottom-25">
                            <h3>Highest Rated Freelancers</h3>
                            <a asp-action="AllFreelancers" asp-controller="Users" class="headline-link">Browse All Freelancers</a>
                        </div>
                    </div>
                    <div class="col-xl-12">
                        <div class="default-slick-carousel freelancers-container freelancers-grid-layout">
                            <#foreach (var freelancer in Model.Freelancers)>
                                <!--Freelancer -->
                                <div class="freelancer">
                                    <!-- Overview -->

```

```
<!-- Overview -->
<div class="freelancer-overview">
    <div class="freelancer-overview-inner">
        <!-- Avatar -->
        <div class="freelancer-avatar">
            <a asp-action="Freelancer" asp-controller="Users" asp-route-id="@freelancer.Id"></a>
        </div>

        <!-- Name -->
        <div class="freelancer-name">
            <@* asp-action="Freelancer" asp-controller="Users" asp-route-id="@freelancer.Id" *@>@freelancer.FirstName @freelancer.LastName 
        </div>

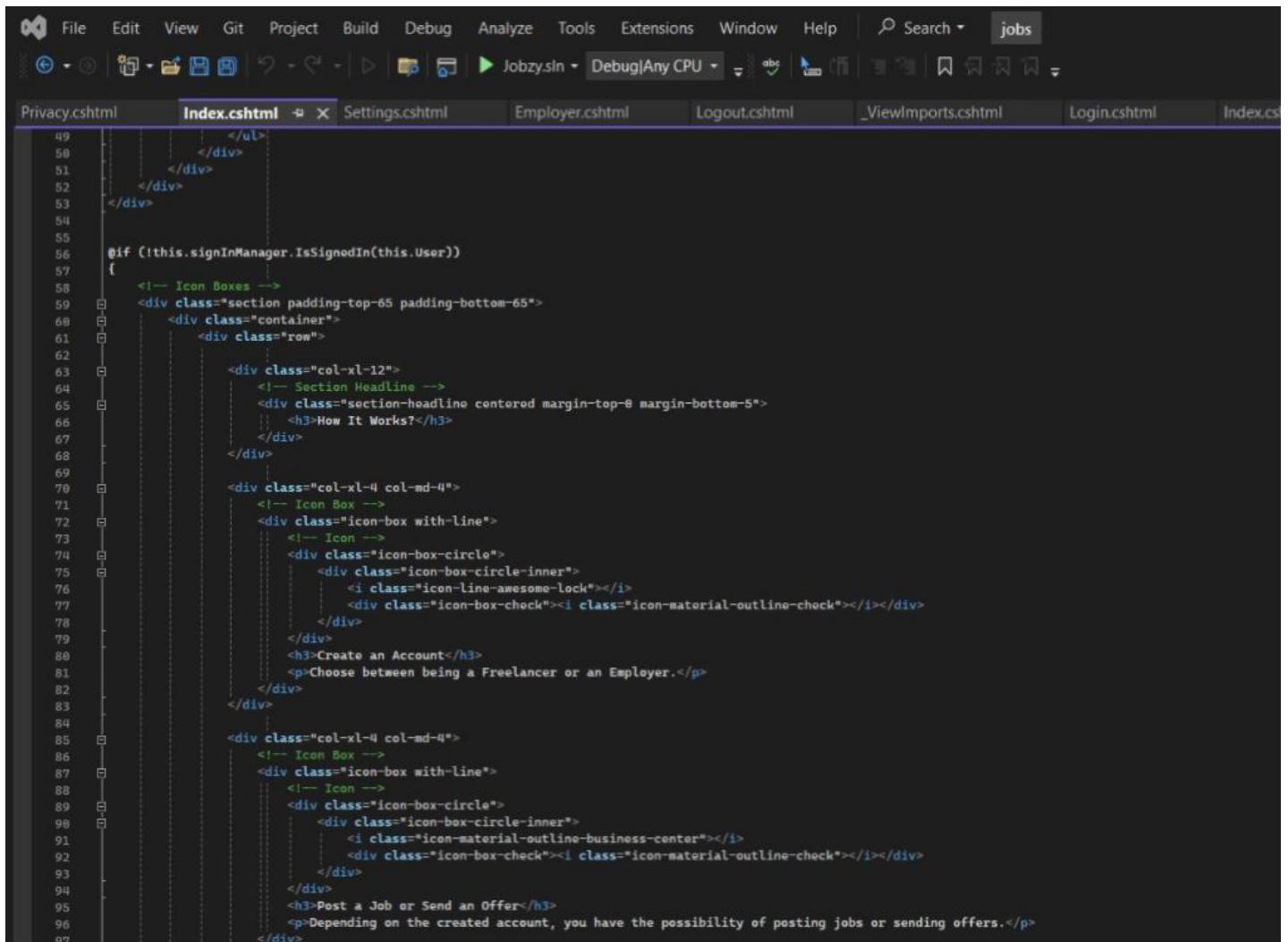
        <!-- Rating -->
        <div class="freelancer-rating">
            <div class="stac-rating" data-rating="@freelancer.AverageRating"></div>
        </div>
    </div>
</div>

<!-- Details -->
<div class="freelancer-details">
    <div class="freelancer-details-list">
        <ul>
            <li>Location <strong><i class="icon-material-outline-location-on"></i> @freelancer.LocationToString</strong></li>
            <li>Jobs Done <strong>@freelancer.JobsDone</strong></li>
            <li>Job Success <strong>@freelancer.JobSuccess</strong></li>
        </ul>
    </div>
    <@* asp-action="Freelancer" asp-controller="Users" asp-route-id="@freelancer.Id" *@>View Profile <i class="icon-material-outline"></i>
</div>
<!-- Freelancer / End -->
</div>
</div>
<!-- Highest Rated Freelancers / End-->
}

else if (this.User.IsInRole("Freelancer") && Model.Jobs.Any())
{
    <!-- Features Jobs -->
    <div class="section gray margin-top-45 padding-top-65 padding-bottom-75">
        <div class="container">
            <div class="row">

```

Home



```
49     </ul>
50   </div>
51 </div>
52 </div>
53 </div>
54
55
56 @if (!this.signInManager.IsSignedIn(this.User))
{
    <!-- Icon Boxes -->
59   <div class="section padding-top-65 padding-bottom-65">
60     <div class="container">
61       <div class="row">
62
63         <div class="col-xl-12">
64           <!-- Section Headline -->
65           <div class="section-headline centered margin-top-0 margin-bottom-5">
66             <h3>How It Works?</h3>
67           </div>
68
69         <div class="col-xl-4 col-md-4">
70           <!-- Icon Box -->
71           <div class="icon-box with-line">
72             <!-- Icon -->
73             <div class="icon-box-circle">
74               <div class="icon-box-circle-inner">
75                 <i class="icon-line-awesome-lock"></i>
76                 <div class="icon-box-check"><i class="icon-material-outline-check"></i></div>
77               </div>
78             </div>
79             <h3>Create an Account</h3>
80             <p>Choose between being a Freelancer or an Employer.</p>
81           </div>
82
83         <div class="col-xl-4 col-md-4">
84           <!-- Icon Box -->
85           <div class="icon-box with-line">
86             <!-- Icon -->
87             <div class="icon-box-circle">
88               <div class="icon-box-circle-inner">
89                 <i class="icon-material-outline-business-center"></i>
90                 <div class="icon-box-check"><i class="icon-material-outline-check"></i></div>
91               </div>
92             </div>
93             <h3>Post a Job or Send an Offer</h3>
94             <p>Depending on the created account, you have the possibility of posting jobs or sending offers.</p>
95           </div>
96
97         </div>
98       </div>
99     </div>
100   </div>
101 </div>
```

```
        <span asp-validation-for="Budget" class="text-danger"></span>
    </div>
</div>

<div class="col-xl-12">
    <div class="submit-field">
        <h5>Job Description</h5>
        <textarea asp-for="Description" cols="30" rows="5" class="with-border"></textarea>
        <span asp-validation-for="Description" class="text-danger"></span>
    </div>
</div>

<div class="col-xl-12">
    <button asp-route="/Dashboard/Jobs/Add" type="submit" class="button ripple-effect big margin-top-30"><i class="icon-feather-plus"></i> Add a Job</button>
</div>
</div>
</div>
<!-- Row / End -->
<partial name="SmallFooterPartial" />

</div>
</div>
<!-- Dashboard Content / End -->
</div>
<!-- Dashboard Container / End -->
```

The screenshot shows the 'Index.cshtml' file in Visual Studio. The code is a form for adding a job, structured with Bootstrap grid classes like 'col-12', 'col-4', and 'col-12'. It includes input fields for Title, Category, Budget, Description, and a submit button. A partial view 'SmallFooterPartial' is included at the bottom.

```
<input asp-for="Title" type="text" class="with-border">
</span>
```

```
<div class="col-12">
<div class="submit-field">
<h5>Job Category</h5>
<select asp-for="CategoryId" class="location-select dropdown with-border" asp-items="categories" data-size="7" title="Select Category">
</select>
</span>
</div>
</div>
```

```
<div class="col-12">
<div class="submit-field">
<h5>Budget</h5>
<div class="input-with-icon">
<input asp-for="Budget" min="10" class="currency with-border" type="number" placeholder="Budget">
<i class="currency">USD</i>
</div>
</span>
</div>
</div>
```

```
<div class="col-12">
<div class="submit-field">
<h5>Job Description</h5>
<textarea asp-for="Description" cols="30" rows="5" class="with-border"></textarea>
</span>
</div>
</div>
```

```
<div class="col-12">
<button asp-route="/Dashboard/Jobs/Add" type="submit" class="button ripple-effect big margin-top-30"><i class="icon-feather-plus"></i> Add a Job</button>
</div>
</div>
</div>
<!-- Row / End -->
<partial name="SmallFooterPartial" />
```

```
</div>
</div>
<!-- Dashboard Content / End -->
```

The screenshot shows the 'All.cshtml' file in Visual Studio. It contains a navigation menu with links for 'All', 'Jobs', 'Sorting', 'Category', and 'CurrentPage'. Below the menu is a 'clearFix' div and a padding section. A partial view 'BigFooterPartial' is included at the bottom.

```
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 href="#">
```

```
<li class="pagination-arrow">
<a href="#" 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)">
<i class="icon-material-outline-keyboard-arrow-right"></i>
</a>
</li>
```

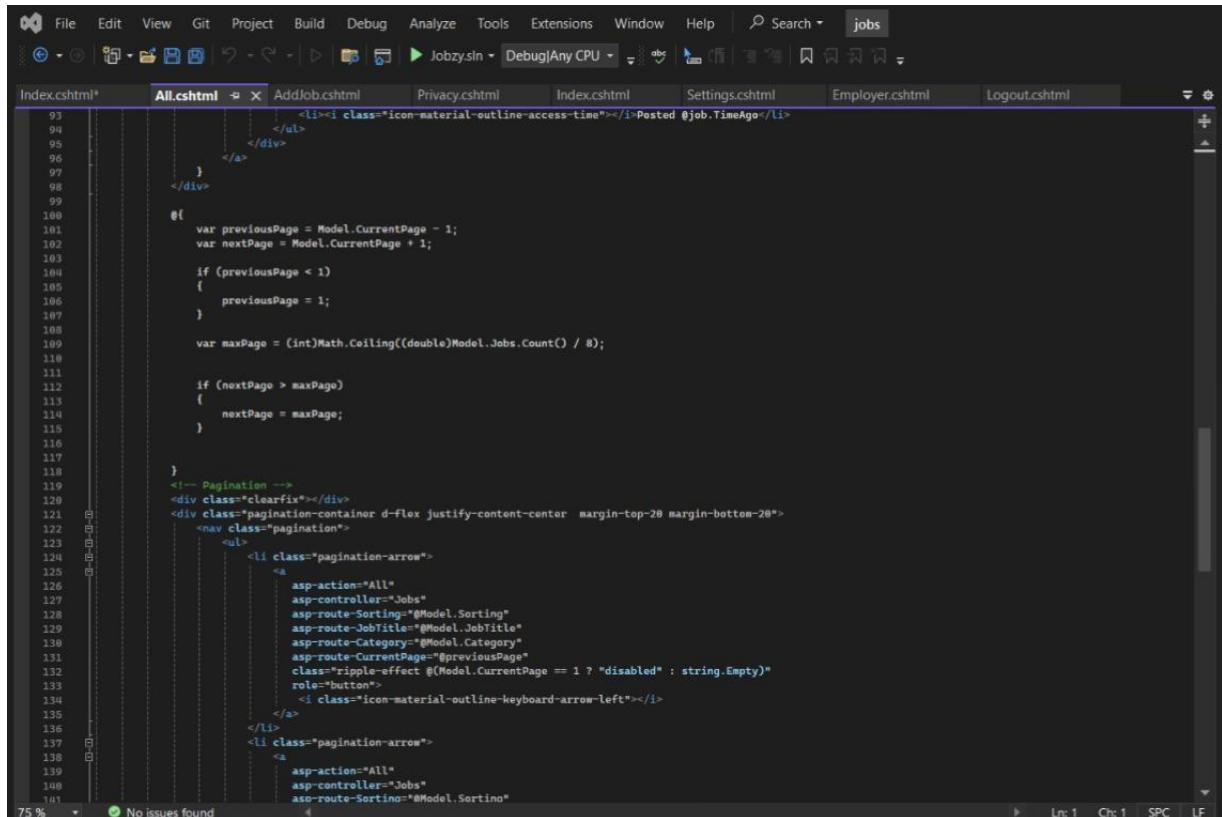
```
</ul>
</div>
<div class="clearfix"></div>
<!-- Pagination / End -->
<div class="padding-bottom-100"></div>
</div>
<!-- Full Page Content / End -->
</div>
```

```
<div class="padding-bottom-100"></div>
<partial name="BigFooterPartial" />
```

The screenshot shows a Microsoft Visual Studio interface with the code editor open. The title bar says "File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs". The tabs at the top include "Index.cshtml", "All.cshtml" (which is the active tab), "AddJob.cshtml", "Privacy.cshtml", "Index.cshtml", "Settings.cshtml", "Employer.cshtml", and "Logout.cshtml". The code in the editor is as follows:

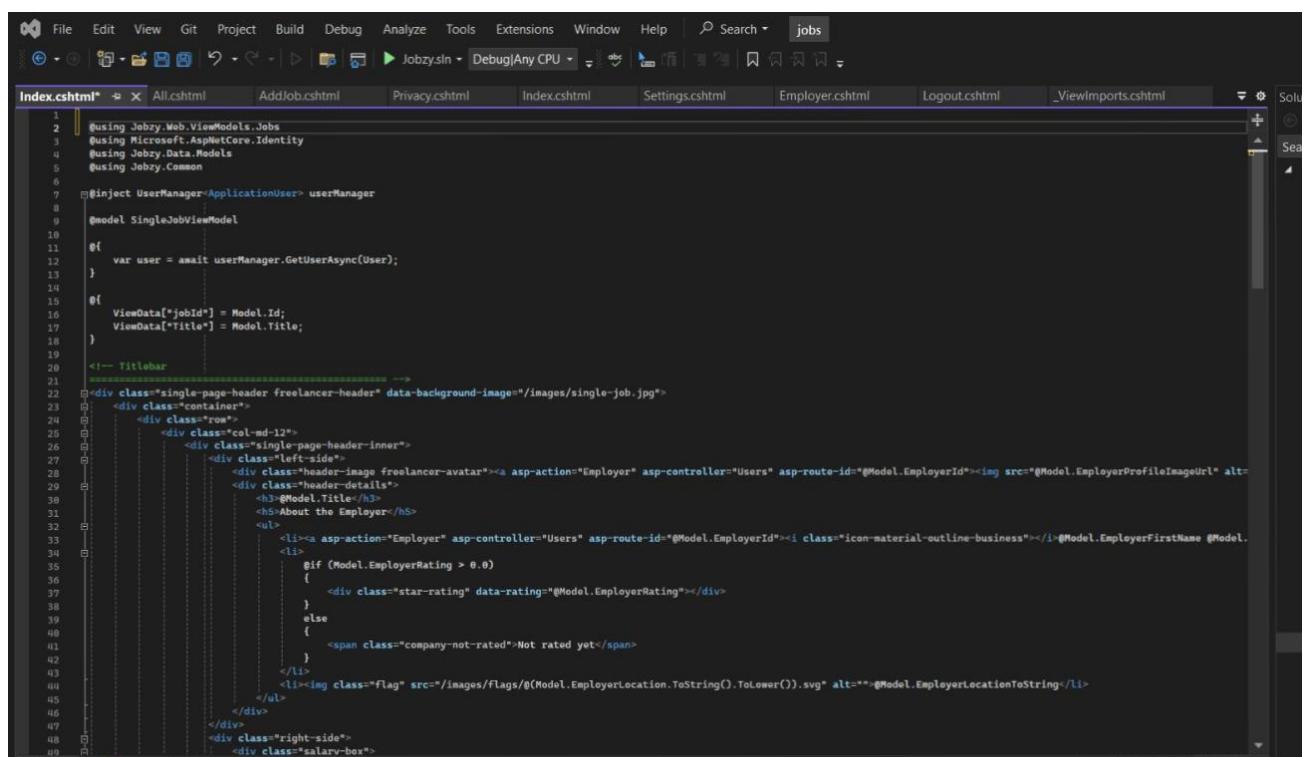
```
1  @using Jobzy.Common
2  @model Jobzy.Web.ViewModels.Jobs.AllJobsQueryModel
3
4  @{
5      var categories = Model.Categories.Select(category => new SelectListItem
6      {
7          Text = category.Name,
8          Value = category.Id
9      });
10 }
11 <!-- Page Content
12 =====>
13 <div class="full-page-container">
14     <div class="full-page-sidebar">
15         <div class="full-page-sidebar-inner" data-simplebar>
16             <div class="sidebar-container">
17                 <form id="search-query-form" method="get" asp-action="All" asp-controller="Jobs">
18                     <!-- Sort by -->
19                     <div class="sidebar-widget">
20                         <h3>Sort by</h3>
21                         <select asp-for="Sorting" class="location-select dropdown" asp-items="Html.GetEnumSelectList<Sorting>()" data-size="7">
22                             <option value="">Search by Category</option>
23                         @foreach (var category in Model.Categories)
24                         {
25                             <option value="@category.Name">@category.Name</option>
26                         }
27                         </select>
28                     </div>
29                     <!-- Keywords -->
30                     <div class="sidebar-widget">
31                         <h3>Job Title</h3>
32                         <div class="keywords-container">
33                             <div class="keyword-input-container">
34                                 <input asp-for="JobTitle" type="text" class="keyword-input" placeholder="Search by job title" />
35                             </div>
36                             <div class="clearfix"></div>
37                         </div>
38                     </div>
39                     <!-- Category -->
40                     <div class="sidebar-widget">
41                         <h3>Category</h3>
42                         <select asp-for="Category" class="location-select dropdown with-border" data-size="7">
43                             <option value="">Search by Category</option>
44                             @foreach (var category in Model.Categories)
45                             {
46                                 <option value="@category.Name">@category.Name</option>
47                             }
48                         </select>
49                     </div>
50                 <div class="sidebar-widget d-flex justify-content-center">
51                     <!-- Content -->
52                 </div>
53             </div>
54         </div>
55     </div>
56 </div>
```

Add job



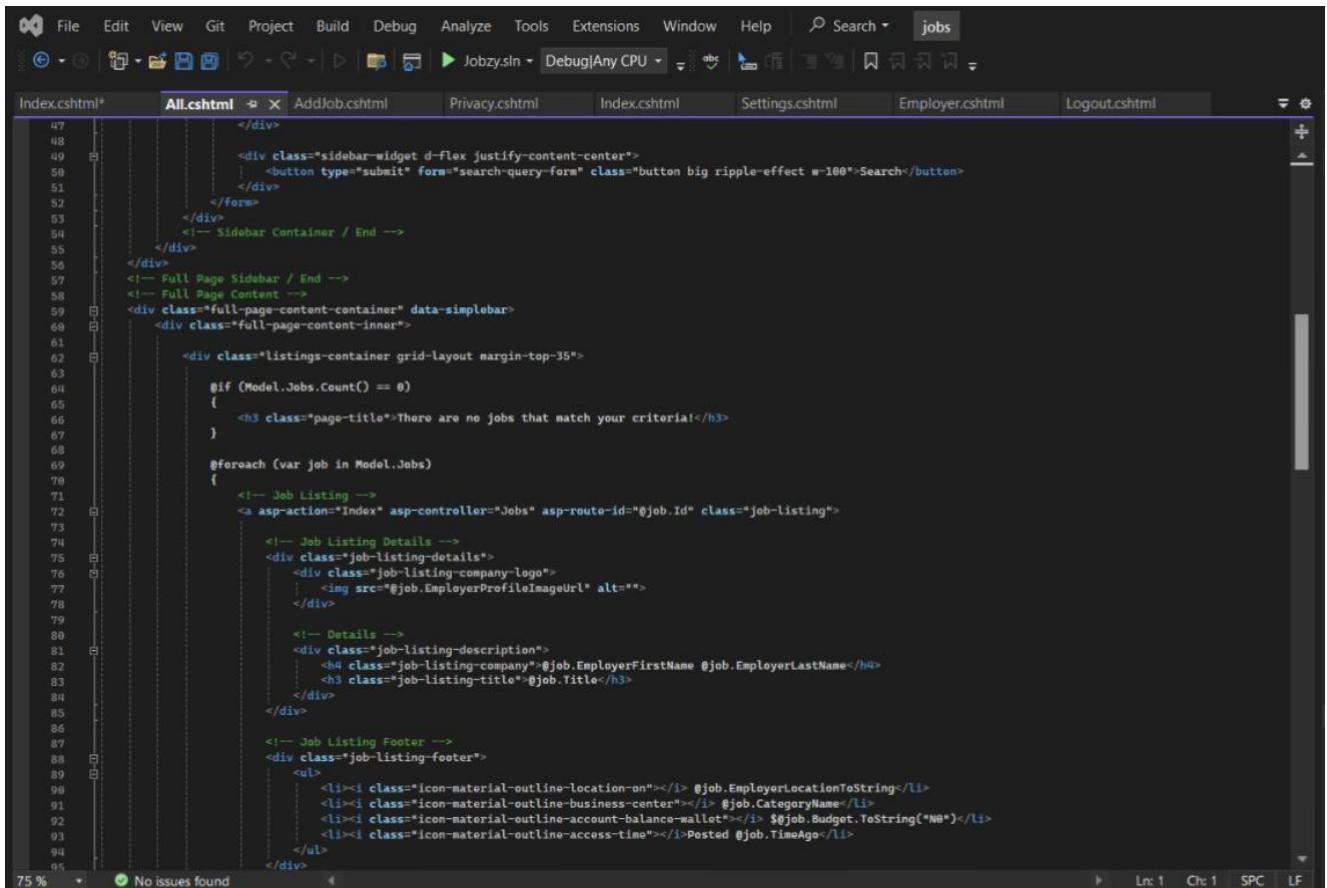
This screenshot shows the Visual Studio IDE with the Index.cshtml file open. The code is part of a pagination component. It includes logic to calculate previous and next page numbers, and then generates navigation links using ASP.NET Core routing attributes. The code uses Bootstrap classes like 'd-flex' and 'justify-content-center' for layout.

```
93     <li><a href="#">...</a>
94   </ul>
95 </div>
96 
97   <div>
98     ...
99   </div>
100 
101  @{
102    var previousPage = Model.CurrentPage - 1;
103    var nextPage = Model.CurrentPage + 1;
104 
105    if (previousPage < 1)
106    {
107      previousPage = 1;
108    }
109 
110    var maxPage = (int)Math.Ceiling((double)Model.Jobs.Count() / 8);
111 
112    if (nextPage > maxPage)
113    {
114      nextPage = maxPage;
115    }
116 
117  }
118 
119  <!-- Pagination -->
120 <div class="clearfix"></div>
121 <div class="pagination-container d-flex justify-content-center margin-top-20 margin-bottom-20">
122   <nav class="pagination">
123     <ul>
124       <li class="pagination-arrow">
125         <a href="#">...</a>
126       <li class="pagination-arrow">
127         <a href="#">...</a>
128       <li class="pagination-arrow">
129         <a href="#">...</a>
130       <li class="pagination-arrow">
131         <a href="#">...</a>
132       <li class="pagination-arrow">
133         <a href="#">...</a>
134       <li class="pagination-arrow">
135         <a href="#">...</a>
136       <li class="pagination-arrow">
137         <a href="#">...</a>
138       <li class="pagination-arrow">
139         <a href="#">...</a>
140       <li class="pagination-arrow">
141         <a href="#">...</a>
142     </ul>
143   </nav>
144 </div>
```



This screenshot shows the Visual Studio IDE with the Index.cshtml file open. The code is for a single job view. It starts with injecting the UserManager service. It then retrieves the user associated with the job. The page header features a background image of a freelancer. The main content area contains details about the employer, including their rating, location, and a flag icon representing their location.

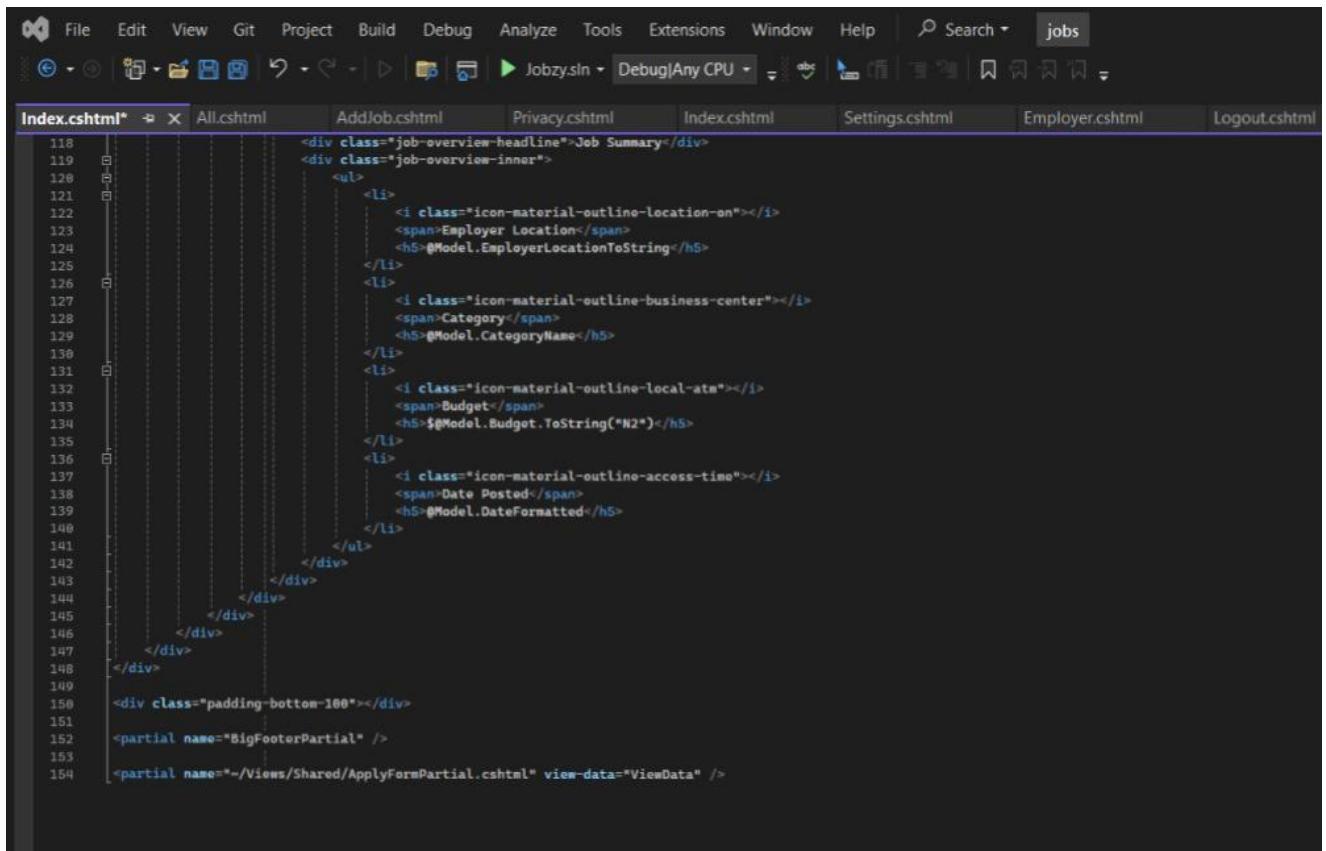
```
1  @using Jobzy.Web.ViewModels.Jobs
2  @using Microsoft.AspNetCore.Identity
3  @using Jobzy.Data.Models
4  @using Jobzy.Common
5 
6  @inject UserManager< ApplicationUser > userManager
7 
8  @model SingleJobViewModel
9 
10 @{
11   var user = await userManager.GetUserAsync(User);
12 }
13 
14 @{
15   ViewData["jobId"] = Model.Id;
16   ViewData["title"] = Model.Title;
17 }
18 
19 
20 <!-- Titlebar
21 =====-->
22 <div class="single-page-header freelancer-header" data-background-image="/images/single-job.jpg">
23   <div class="container">
24     <div class="row">
25       <div class="col-sm-12">
26         <div class="single-page-header-inner">
27           <div class="left-side">
28             <div class="header-imgae freelancer-avatar"><a href="#" asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"></a></div>
29             <div class="header-details">
30               <h2>@Model.Title</h2>
31               <h3>About the Employer</h3>
32               <ul>
33                 <li><a href="#" asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"><i class="icon-material-outline-business"></i> @Model.EmployerFirstName @Model.EmployerLastName</a></li>
34                 <li>
35                   &#x2022; If (Model.EmployerRating > 0.0)
36                   {
37                     <div class="star-rating" data-rating="@Model.EmployerRating"></div>
38                   }
39                   else
40                   {
41                     <span class="company-not-rated">Not rated yet</span>
42                   }
43                 </li>
44               </ul>
45             </div>
46           </div>
47           <div class="right-side">
48             <div class="salarv-box">
49               ...
50             </div>
51           </div>
52         </div>
53       </div>
54     </div>
55   </div>
56 </div>
```



The screenshot shows a Microsoft Visual Studio code editor window. The title bar includes the project name "Jobzy.sln" and the current file "All.cshtml". The editor displays the following C# Razor code:

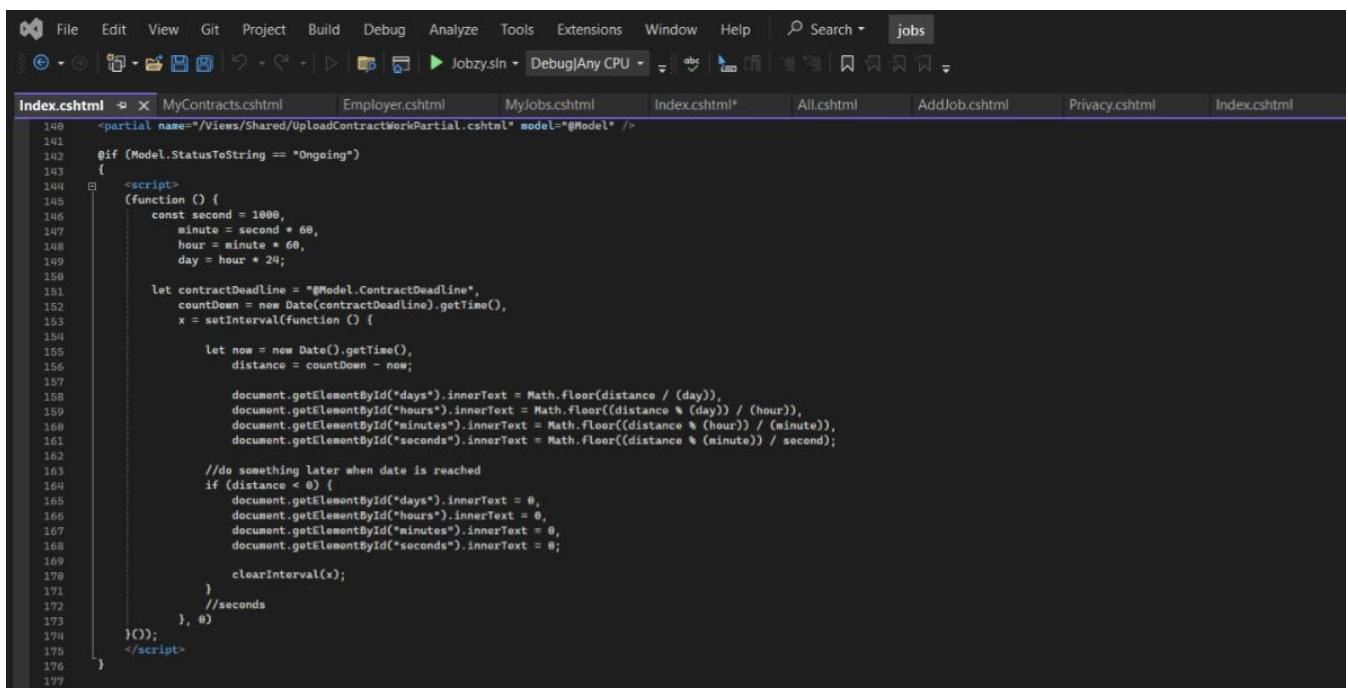
```
47
48
49     <div class="sidebar-widget d-flex justify-content-center">
50         <button type="submit" form="search-query-form" class="button big ripple-effect w-100">Search</button>
51     </div>
52     </form>
53     </div>
54     <!-- Sidebar Container / End -->
55 </div>
56     <!-- Full Page Sidebar / End -->
57     <!-- Full Page Content -->
58     <div class="full-page-content-container" data-simplebar>
59         <div class="full-page-content-inner">
60             <div class="listings-container grid-layout margin-top-35">
61
62                 @if (Model.Jobs.Count() == 0)
63                 {
64                     <h3 class="page-title">There are no jobs that match your criteria!</h3>
65                 }
66
67                 @foreach (var job in Model.Jobs)
68                 {
69                     <!-- Job Listing -->
70                     <a href="#" asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing">
71
72                         <!-- Job Listing Details -->
73                         <div class="job-listing-details">
74                             <div class="job-listing-company-logo">
75                                 
76                             </div>
77
78                             <!-- Details -->
79                             <div class="job-listing-description">
80                                 <h4 class="job-listing-company">@job.EmployerFirstName @job.EmployerLastName</h4>
81                                 <h3 class="job-listing-title">@job.Title</h3>
82                             </div>
83                         </div>
84
85                         <!-- Job Listing Footer -->
86                         <div class="job-listing-footer">
87                             <ul>
88                                 <li><i class="icon-material-outline-location-on"></i> @job.EmployerLocationToString</li>
89                                 <li><i class="icon-material-outline-business-center"></i> @job.CategoryName</li>
90                                 <li><i class="icon-material-outline-account-balance-wallet"></i> $@job.Budget.ToString("N0")</li>
91                                 <li><i class="icon-material-outline-access-time"></i> Posted @job.TimeAgo</li>
92                             </ul>
93                         </div>
94
95                     </a>
96                 </div>
97             </div>
98         </div>
99     </div>
100
```

All jobs



This screenshot shows the Visual Studio IDE with the 'Index.cshtml' file open in the editor. The file is part of the 'Jobzy.sln' project, specifically under the 'All' view. The code is a partial view that displays a summary of job details. It includes sections for location, category, budget, and date posted, each with an associated icon and text. The code uses C# syntax for model binding (@Model) and Razor directives (@if). The file ends with two partial view declarations.

```
118     <div class="job-overview-headline">Job Summary</div>
119     <div class="job-overview-inner">
120         <ul>
121             <li>
122                 <i class="icon-material-outline-location-on"></i>
123                 <span>Employer Location</span>
124                 <h5>@Model.EmployerLocationToString</h5>
125             </li>
126             <li>
127                 <i class="icon-material-outline-business-center"></i>
128                 <span>Category</span>
129                 <h5>@Model.CategoryName</h5>
130             </li>
131             <li>
132                 <i class="icon-material-outline-local-atm"></i>
133                 <span>Budget</span>
134                 <h5>$@Model.Budget.ToString("N2")</h5>
135             </li>
136             <li>
137                 <i class="icon-material-outline-access-time"></i>
138                 <span>Date Posted</span>
139                 <h5>@Model.DateFormatted</h5>
140             </li>
141         </ul>
142     </div>
143     </div>
144     </div>
145     </div>
146     </div>
147     </div>
148 </div>
149 <div class="padding-bottom-100"></div>
150 <partial name="BigFooterPartial" />
151 <partial name="~/Views/Shared/ApplyFormPartial.cshtml" view-data="ViewData" />
```



This screenshot shows the Visual Studio IDE with the 'Index.cshtml' file open in the editor. The file is part of the 'Jobzy.sln' project, specifically under the 'MyContracts' view. The code contains a script block that performs a countdown from a specified deadline. It calculates the time difference between the current date and the deadline, then updates four `document.getElementById` elements ('days', 'hours', 'minutes', 'seconds') every second until the deadline is reached. Once the deadline is passed, it clears the interval and sets all four elements to zero.

```
140 <partial name="/Views/Shared/UploadContractWorkPartial.cshtml" model="@Model" />
141 @if (Model.StatusToString == "Ongoing")
142 {
143     <script>
144         (function () {
145             const second = 1000,
146                 minute = second * 60,
147                 hour = minute * 60,
148                 day = hour * 24;
149
150             let contractDeadline = "@Model.ContractDeadline",
151                 countDown = new Date(contractDeadline).getTime(),
152                 x = setInterval(function () {
153
154                 let now = new Date().getTime(),
155                     distance = countDown - now;
156
157                 document.getElementById("days").innerText = Math.floor(distance / (day)),
158                 document.getElementById("hours").innerText = Math.floor((distance % (day)) / (hour)),
159                 document.getElementById("minutes").innerText = Math.floor((distance % (hour)) / (minute)),
160                 document.getElementById("seconds").innerText = Math.floor((distance % (minute)) / second);
161
162                 //do something later when date is reached
163                 if (distance < 0) {
164                     document.getElementById("days").innerText = 0,
165                     document.getElementById("hours").innerText = 0,
166                     document.getElementById("minutes").innerText = 0,
167                     document.getElementById("seconds").innerText = 0;
168
169                     clearInterval(x);
170                 }
171                 //seconds
172             }, 0)
173         })();
174     </script>
175 }
```

```
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml
49     <div class="salary-box">
50         <div class="salary-type">Budget</div>
51         <div class="salary-amount">$Model.Budget.ToString("N2")</div>
52     </div>
53     </div>
54     </div>
55     </div>
56     </div>
57 </div>
58 </div>
59
60    !-- Page Content
61    ===== -->
62 <div class="container">
63     <div class="row">
64
65        <!-- Content -->
66        <div class="col-xl-8 col-lg-8 content-right-offset">
67            <div class="single-page-section">
68                <h3>Job Description</h3>
69                <p>@Model.Description</p>
70            </div>
71        </div>
72
73
74        <!-- Sidebar -->
75        <div class="col-xl-4 col-lg-4">
76            <div class="sidebar-container">
77
78                &if (this.User.IsInRole("Freelancer"))
79                {
80                    &if (Model.Status == JobStatus.InContract && Model.ContractsFreelancerIds.Contains(user.Id))
81                    {
82                        <a href="#" 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>
83                    }
84                    else if (Model.Status == JobStatus.Open && !Model.OffersFreelancerIds.Contains(user.Id))
85                    {
86                        <a href="#small-dialog" class="apply-now-button popup-with-zoom-anm" href="#">Make an Offer<i class="icon-material-outline-arrow-right-alt"></i></a>
87                    }
88                    else if (Model.Status == JobStatus.Open && Model.OffersFreelancerIds.Contains(user.Id))
89                    {
90                        <a role="button" class="apply-now-button bg-secondary text-white">Offer already sent</a>
91                    }
92                    else
93                    {
94                        <a role="button" class="apply-now-button bg-secondary text-white">Job is closed</a>
95                    }
96                }
97            </div>
98        </div>
99    </div>
100
101    <!-- Sidebar Widget -->
102    <div class="sidebar-widget">
103        <div class="job-overview">
104            <div class="job-overview-headline">Job Summary</div>
105            <div class="job-overview-inner">
106                <ul>
107                    <li>
108                        <i class="icon-material-outline-location-on"></i>
109                        <span>Employer Location:</span>
110                        <h5>@Model.EmployerLocationToString</h5>
111                    </li>
112                    <li>
113                        <i class="icon-material-outline-business-center"></i>
114                        <span>Category:</span>
115                        <h5>@Model.CategoryName</h5>
116                    </li>
117                    <li>
118                        <i class="icon-material-outline-local-atm"></i>
119                        <span>Budget:</span>
120                        <h5>$Model.Budget.ToString("N2")</h5>
121                    </li>
122                    <li>
123                        <i class="icon-material-outline-access-time"></i>
124                        <span>Date Posted:</span>
125                        <h5>@Model.DateFormatted</h5>
126                    </li>
127                </ul>
128            </div>
129        </div>
130    </div>
131
```

```
File Edit View Git Project Build Debug Analyze Tools Extensions Window Help Search jobs
Index.cshtml* All.cshtml AddJob.cshtml Privacy.cshtml Index.cshtml Settings.cshtml Employer.cshtml Logout.cshtml _ViewImports.cshtml
}
&if (this.User.IsInRole("Employer"))
{
    &if (Model.Status == JobStatus.InContract && Model.EmployerId == user.Id)
    {
        <a href="#" 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 href="#" 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>
```

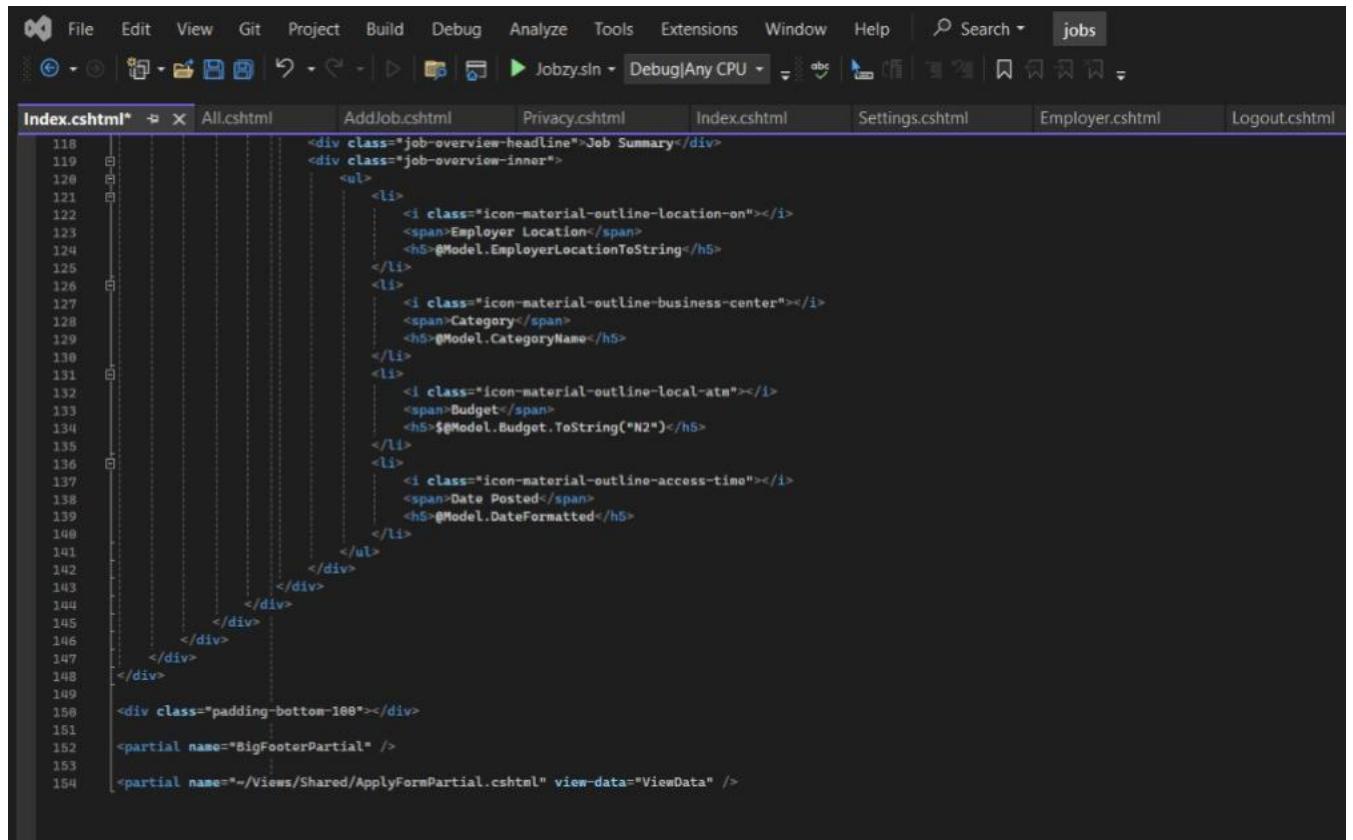
Job details

The screenshot shows the Visual Studio code editor with the file `Index.cshtml` open. The code is a Razor view for displaying job listings. It includes sections for sidebar navigation, job listing details, and footer information. The code uses Bootstrap classes and Material Design icons.

```
47     </div>
48     <!-- Sidebar Container / End -->
49   </div>
50   <!-- Full Page sidebar / End -->
51   <!-- Full Page Content -->
52   <div class="full-page-content-container" data-simplebar>
53     <div class="full-page-content-inner">
54       <div class="listings-container grid-layout margin-top-35">
55         &if (Model.Jobs.Count() == 0)
56         {
57           <h3 class="page-title">There are no jobs that match your criteria!</h3>
58         }
59         &foreach (var job in Model.Jobs)
60         {
61           <!-- Job Listing -->
62           <a asp-action="Index" asp-controller="Jobs" asp-route-id="@job.Id" class="job-listing">
63             <!-- Job Listing Details -->
64             <div class="job-listing-details">
65               <div class="job-listing-company-logo">
66                 
67               </div>
68               <!-- Details -->
69               <div class="job-listing-description">
70                 <h4 class="job-listing-company">@job.EmployerFirstName @job.EmployerLastName</h4>
71                 <h3 class="job-listing-title">@job.Title</h3>
72               </div>
73             </div>
74             <!-- Job Listing Footer -->
75             <div class="job-listing-footer">
76               <ul>
77                 <li><i class="icon-material-outline-location-on"></i> @job.EmployerLocationToString</li>
78                 <li><i class="icon-material-outline-business-center"></i> @job.CategoryName</li>
79                 <li><i class="icon-material-outline-account-balance-wallet"></i> $@job.Budget.ToString("N2")</li>
80                 <li><i class="icon-material-outline-access-time"></i> Posted @job.TimeAgo</li>
81               </ul>
82             </div>
83           </a>
84         </div>
85       </div>
86     </div>
87   </div>
88   <!-- Sidebar Widget -->
89   <div class="sidebar-widget">
90     <div class="job-overview">
91       <div class="job-overview-headline">Job Summary</div>
92       <div class="job-overview-inner">
93         <ul>
94           <li><i class="icon-material-outline-location-on"></i>
95             <span>Employer Location</span>
96             <h5>@Model.EmployerLocationToString</h5>
97           </li>
98           <li><i class="icon-material-outline-business-center"></i>
99             <span>Category</span>
100            <h5>@Model.CategoryName</h5>
101          </li>
102          <li><i class="icon-material-outline-local-atm"></i>
103            <span>Budget</span>
104            <h5>$@Model.Budget.ToString("N2")</h5>
105          </li>
106          <li><i class="icon-material-outline-access-time"></i>
107            <span>Date Posted</span>
108            <h5>@Model.DateFormatted</h5>
109          </li>
110        </ul>
111      </div>
112    </div>
113  </div>;
114
```

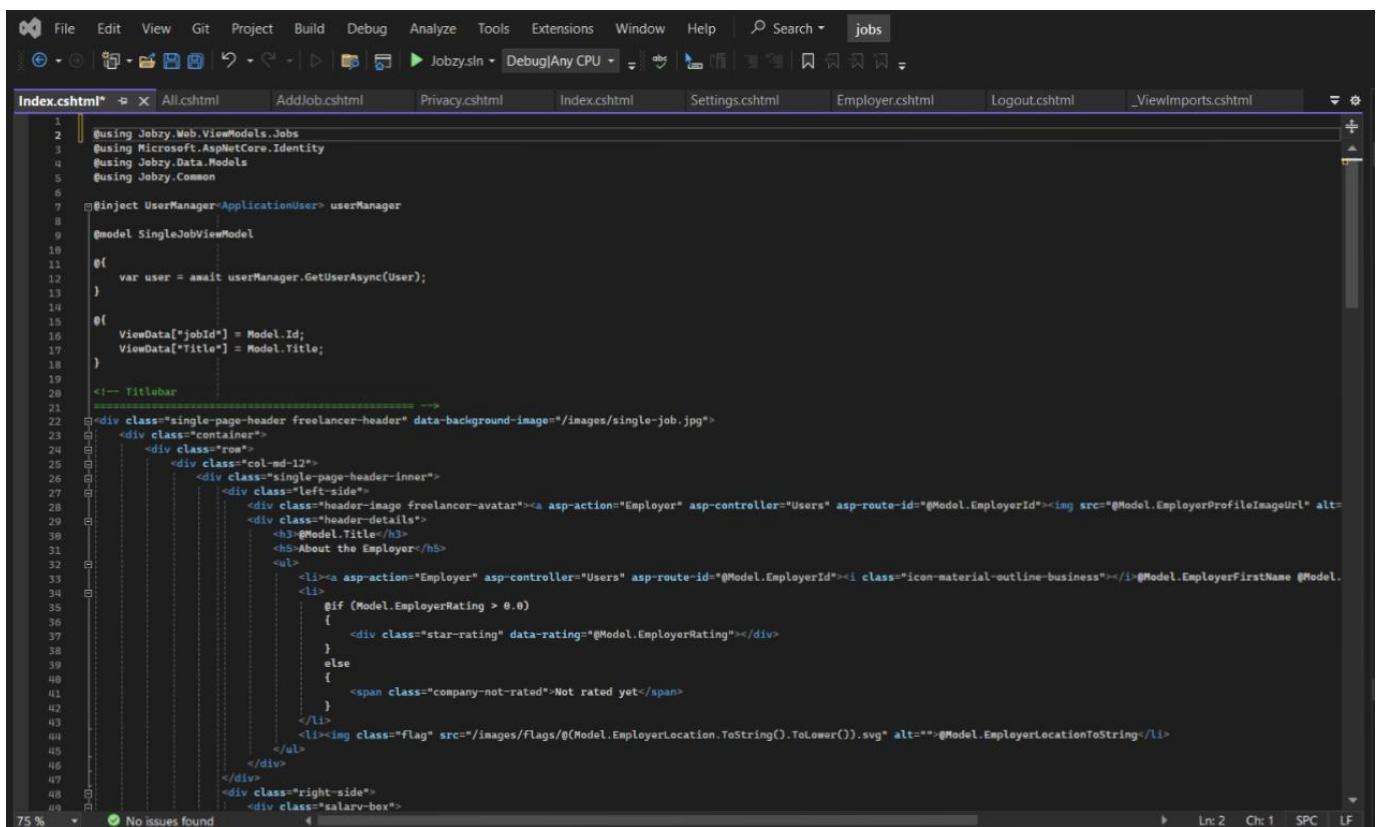
The screenshot shows the Visual Studio code editor with the file `Index.cshtml` open. The code is a Razor view for displaying job details. It includes sections for sidebar navigation, job summary, and detailed job information. The code uses Bootstrap classes and Material Design icons.

```
97     &if (this.User.IsInRole("Employer"))
98     {
99       if (Model.Status == JobStatus.InContract && Model.EmployerId == user.Id)
100       {
101         <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>
102       }
103       else if (Model.Status != JobStatus.InContract && Model.EmployerId == user.Id)
104       {
105         <a asp-action="MyJobs" asp-controller="Jobs" class="apply-now-button">My Jobs<i class="icon-material-outline-arrow-right-alt"></i></a>
106       }
107       else
108       {
109         <a role="button" class="apply-now-button bg-secondary text-white">Employers cannot apply to jobs</a>
110       }
111     }
112   </div>;
113
114   <!-- Sidebar Widget -->
115   <div class="sidebar-widget">
116     <div class="job-overview">
117       <div class="job-overview-headline">Job Summary</div>
118       <div class="job-overview-inner">
119         <ul>
120           <li>
121             <i class="icon-material-outline-location-on"></i>
122             <span>Employer Location</span>
123             <h5>@Model.EmployerLocationToString</h5>
124           </li>
125           <li>
126             <i class="icon-material-outline-business-center"></i>
127             <span>Category</span>
128             <h5>@Model.CategoryName</h5>
129           </li>
130           <li>
131             <i class="icon-material-outline-local-atm"></i>
132             <span>Budget</span>
133             <h5>$@Model.Budget.ToString("N2")</h5>
134           </li>
135           <li>
136             <i class="icon-material-outline-access-time"></i>
137             <span>Date Posted</span>
138             <h5>@Model.DateFormatted</h5>
139           </li>
140         </ul>
141       </div>
142     </div>;
143   </div>;
144
```



The screenshot shows the Visual Studio IDE interface with the 'Index.cshtml' file open in the editor. The file contains C# Razor code for rendering job details. The code includes a list of icons representing job information such as location, category, budget, and date posted.

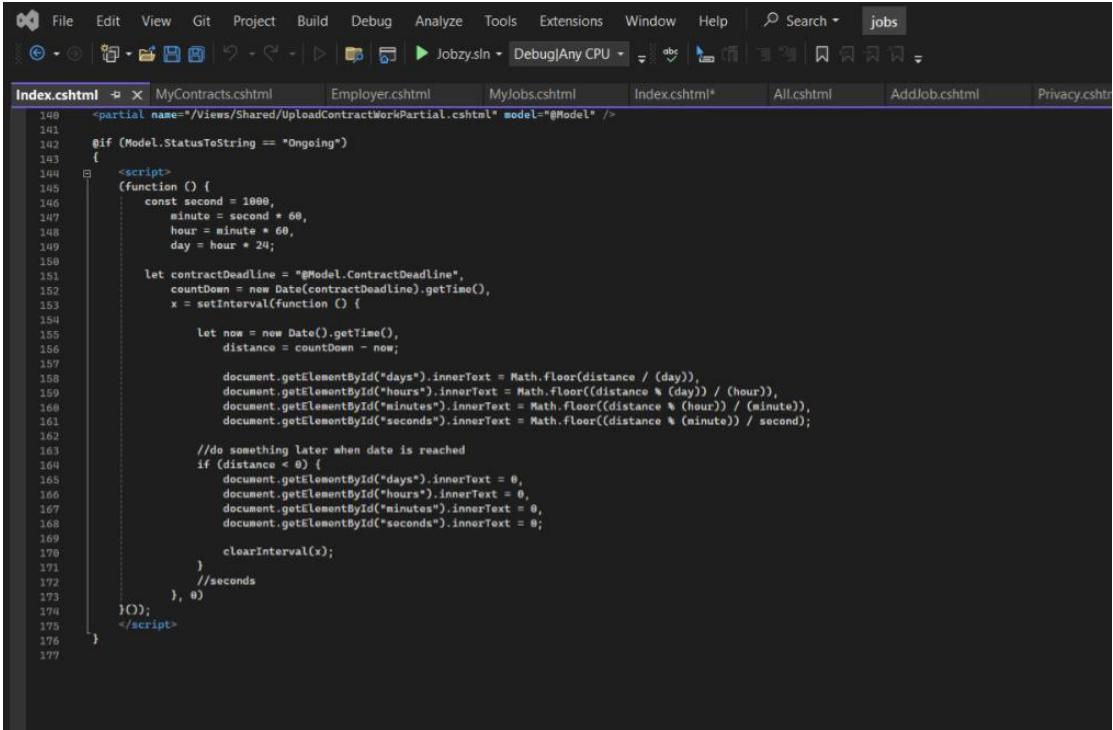
```
118 <div class="job-overview-headline">Job Summary</div>
119 <div class="job-overview-inner">
120   <ul>
121     <li>
122       <i class="icon-material-outline-location-on"></i>
123       <span>Employer Location</span>
124       <h5>@Model.EmployerLocationToString</h5>
125     </li>
126     <li>
127       <i class="icon-material-outline-business-center"></i>
128       <span>Category</span>
129       <h5>@Model.CategoryName</h5>
130     </li>
131     <li>
132       <i class="icon-material-outline-local-atm"></i>
133       <span>Budget</span>
134       <h5>$@Model.Budget.ToString("N2")</h5>
135     </li>
136     <li>
137       <i class="icon-material-outline-access-time"></i>
138       <span>Date Posted</span>
139       <h5>@Model.DateFormatted</h5>
140     </li>
141   </ul>
142 </div>
143 </div>
144 </div>
145 </div>
146 </div>
147 </div>
148 </div>
149 <div class="padding-bottom-100"></div>
150 <partial name="BigFooterPartial" />
151 <partial name="~/Views/Shared/ApplyFormPartial.cshtml" view-data="ViewData" />
```



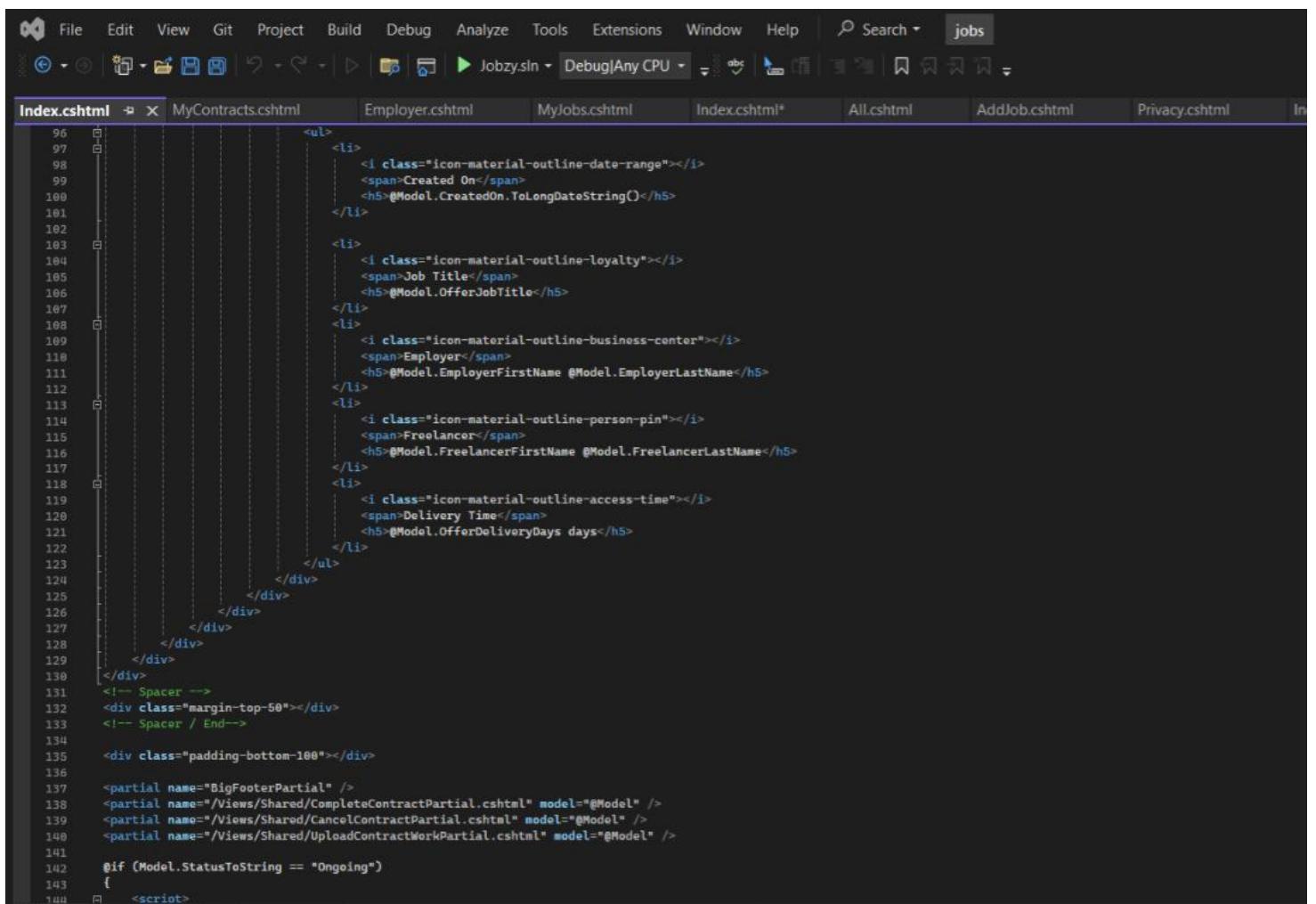
The screenshot shows the Visual Studio IDE interface with the 'Index.cshtml' file open in the editor. The file contains C# Razor code for rendering job details, including imports for Jobzy.Web.ViewModels, Microsoft.AspNetCore.Identity, Jobzy.Data.Models, and Jobzy.Common. The code uses the UserManager to get user details and sets ViewData for job ID and title.

```
1 @using Jobzy.Web.ViewModels.Jobs
2 @using Microsoft.AspNetCore.Identity
3 @using Jobzy.Data.Models
4 @using Jobzy.Common
5
6 @inject UserManager< ApplicationUser > userManager
7
8 @model SingleJobViewModel
9
10 @{
11   var user = await userManager.GetUserAsync(User);
12 }
13
14 @{
15   ViewData[" jobId "] = Model.Id;
16   ViewData[" title "] = Model.Title;
17 }
18
19 <!-- Toolbar -->
20 <div class="single-page-header freelancer-header" data-background-image="/images/single-job.jpg">
21   <div class="container">
22     <div class="row">
23       <div class="col-md-12">
24         <div class="single-page-header-inner">
25           <div class="left-side">
26             <div class="header-image freelancer-avatar"><a href="#" asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"></a></div>
27             <div class="header-details">
28               <h3>@Model.Title</h3>
29               <h5>About the Employer</h5>
30               <ul>
31                 <li><a href="#" asp-action="Employer" asp-controller="Users" asp-route-id="@Model.EmployerId"><i class="icon-material-outline-business"></i> @Model.EmployerFirstName @Model.EmployerLastName</a></li>
32                 <li>
33                   @if (Model.EmployerRating > 0.0)
34                   {
35                     <div class="star-rating" data-rating="@Model.EmployerRating"></div>
36                   }
37                   else
38                   {
39                     <span class="company-not-rated">Not rated yet</span>
40                   }
41                 </li>
42               </ul>
43             </div>
44             <div class="right-side">
45               <div class="salarr-box">
46                 
47               </div>
48             </div>
49           </div>
50         </div>
51       </div>
52     </div>
53   </div>
54 </div>
```

Contract



```
140 <partial name="/Views/Shared/UploadContractWorkPartial.cshtml" model="@Model" />
141
142 @if (Model.StatusToString == "Ongoing")
143 {
144     <script>
145         (function () {
146             const second = 1000,
147                 minute = second * 60,
148                 hour = minute * 60,
149                 day = hour * 24;
150
151             let contractDeadline = "@Model.ContractDeadline",
152                 countDown = new Date(contractDeadline).getTime(),
153                 x = setInterval(function () {
154
155                 let now = new Date().getTime(),
156                     distance = countDown - now;
157
158                 document.getElementById("days").innerText = Math.floor(distance / (day)),
159                 document.getElementById("hours").innerText = Math.floor((distance % (day)) / (hour)),
160                 document.getElementById("minutes").innerText = Math.floor((distance % (hour)) / (minute)),
161                 document.getElementById("seconds").innerText = Math.floor((distance % (minute)) / second);
162
163                 //do something later when date is reached
164                 if (distance < 0) {
165                     document.getElementById("days").innerText = 0,
166                     document.getElementById("hours").innerText = 0,
167                     document.getElementById("minutes").innerText = 0,
168                     document.getElementById("seconds").innerText = 0;
169
170                     clearInterval(x);
171                 }
172                 //seconds
173             }, 1000)
174         })();
175     </script>
176 }
177 }
```

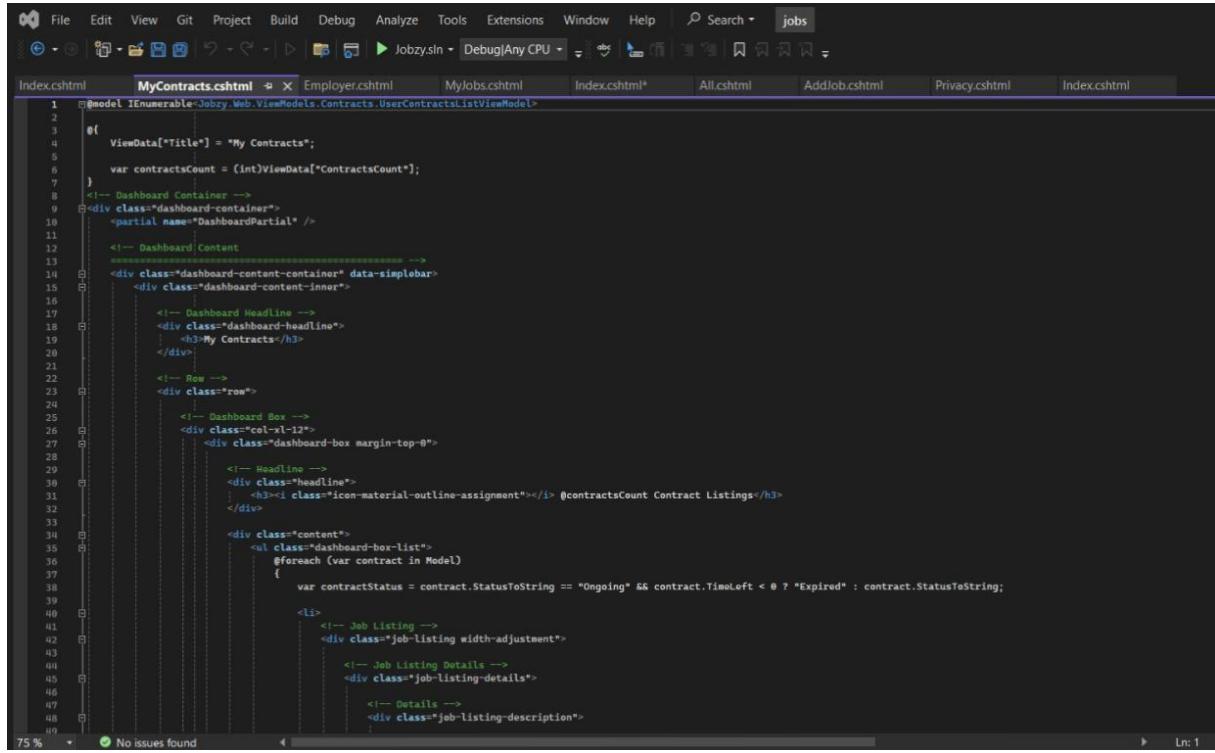
```
96     <ul>
97         <li>
98             <i class="icon-material-outline-date-range"></i>
99             <span>Created On</span>
100            <h5>@Model.CreatedOn.ToString("yyyy-MM-dd")</h5>
101        </li>
102
103        <li>
104            <i class="icon-material-outline-loyalty"></i>
105            <span>Job Title</span>
106            <h5>@Model.OfferJobTitle</h5>
107        </li>
108
109        <li>
110            <i class="icon-material-outline-business-center"></i>
111            <span>Employer</span>
112            <h5>@Model.EmployerFirstName @Model.EmployerLastName</h5>
113        </li>
114
115        <li>
116            <i class="icon-material-outline-person-pin"></i>
117            <span>Freelancer</span>
118            <h5>@Model.FreelancerFirstName @Model.FreelancerLastName</h5>
119        </li>
120
121        <li>
122            <i class="icon-material-outline-access-time"></i>
123            <span>Delivery Time</span>
124            <h5>@Model.OfferDeliveryDays days</h5>
125        </li>
126    </ul>
127 </div>
128 </div>
129 </div>
130
131 <!-- Spacer -->
132 <div class="margin-top-50"></div>
133 <!-- Spacer / End-->
134
135 <div class="padding-bottom-100"></div>
136
137 <partial name="BigFooterPartial" />
138 <partial name="/Views/Shared/CompleteContractPartial.cshtml" model="@Model" />
139 <partial name="/Views/Shared/CancelContractPartial.cshtml" model="@Model" />
140 <partial name="/Views/Shared/UploadContractWorkPartial.cshtml" model="@Model" />
141
142 if (Model.StatusToString == "Ongoing")
143 {
144     <script>
```

The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** File, Edit, View, Git, Project, Build, Debug, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search bar with the text "jobs".
- Toolbars:** Standard toolbar with icons for New, Open, Save, Print, etc.
- Project Explorer:** Shows "Jobzy.sln" and "Debug|Any CPU".
- Solution Explorer:** Shows files like Index.cshtml, MyContracts.cshtml, Employer.cshtml, MyJobs.cshtml, Index.cshtml+, All.cshtml, AddJob.cshtml, and Privacy.cshtml.
- Editor Area:** Displays the code for "Index.cshtml".
- Status Bar:** Shows "75 %", "No issues found", and a green checkmark icon.

```
1 @model Jobzy.Web.ViewModels.Contracts.SingleContractViewModel
2
3 <!-- Titlebar
4 =====>
5 <div class="single-page-header" data-background-image="/images/single-task.jpg">
6   <div class="container">
7     <div class="row">
8       <div class="col-md-12">
9         <div class="d-flex justify-content-between align-items-baseline">
10          <div class="card shadow p-3 bg-white rounded text-center p-lg-4">
11            <div class="salary-type">Contract Status</div>
12            <div class="margin-top-15"></div>
13
14            <div class="bid-acceptance w-100 @Model.StatusColor">@Model.StatusName</div>
15
16
17          <div class="text-center">
18            <h3 class="contract-id-text">Contract <br />@Model.Id.ToUpper()</h3>
19          </div>
20
21          <div class="card shadow p-3 bg-white rounded text-center p-lg-4">
22            <div class="salary-type">Contract Budget</div>
23            <div class="margin-top-15"></div>
24            <div class="bid-acceptance w-100" style="background-color: #007bff; color: white; padding: 5px; border-radius: 5px">$@Model.OfferFixedPrice</div>
25
26        </div>
27      </div>
28    </div>
29  </div>
30
31  <!-- Page Content
32 =====>
33 <div class="container">
34   &if (Model.StatusToString == "Ongoing" && Model.TimeLeft > 0)
35   {
36     <div id="countdown" class="countdown blue margin-bottom-70 py-lg-4">
37       <span id="days"></span> days
38       <span id="hours"></span> hours
39       <span id="minutes"></span> minutes
40       <span id="seconds"></span> seconds
41       left
42     </div>
43   }
44
45   <div class="row">
46     <!-- Content -->
47     <div class="col-xl-8 col-lg-8 content-right-offset">
```

User contract



The screenshot shows a code editor with the following details:

- File Menu:** File, Edit, View, Git, Project, Build, Debug, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search with dropdown for "jobs".
- Tab Bar:** Index.cshtml, MyContracts.cshtml, Employer.cshtml, MyJobs.cshtml, Index.cshtml*, All.cshtml, AddJob.cshtml, Privacy.cshtml, Index.cshtml.
- Content Area:** The code is for a dashboard page named "Index.cshtml". It uses Bootstrap classes like "dashboard-content", "row", "col-12", and "margin-top-0". It includes a partial view named "DashboardPartial". The code handles a model of type "IEnumerable<Jobzy.Web.ViewModels.Contracts.UserContractsListViewModel>". It sets the title to "My Contracts" and initializes a variable "contractsCount". The main content area contains a heading "My Contracts" and a list of contracts. Each contract item includes a status indicator (e.g., "icon-material-outline-assignment") and a link to its details.
- Status Bar:** Shows "75 %" completion, "No issues found", and "Ln: 1".

```
1 @model IEnumerable<Jobzy.Web.ViewModels.Contracts.UserContractsListViewModel>
2
3 @{
4     ViewData["Title"] = "My Contracts";
5 }
6     var contractsCount = (int)ViewData["ContractsCount"];
7 }
8 <!-- Dashboard Container -->
9 <div class="dashboard-container">
10    <partial name="DashboardPartial" />
11 <!-- Dashboard Content -->
12 <div class="dashboard-content-container" data-simplebar>
13    <div class="dashboard-content-inner">
14        <!-- Dashboard Headline -->
15        <div class="dashboard-headline">
16            <h2>My Contracts</h2>
17        </div>
18        <!-- Row -->
19        <div class="row">
20            <!-- Dashboard Box -->
21            <div class="col-12">
22                <div class="dashboard-box margin-top-0">
23                    <!-- Headline -->
24                    <div class="headline">
25                        <h3><i class="icon-material-outline-assignment"/> @contractsCount Contract Listings</h3>
26                    </div>
27                    <div class="content">
28                        <ul class="dashboard-box-list">
29                            @foreach (var contract in Model)
30                            {
31                                var contractStatus = contract.StatusToString == "Ongoing" && contract.TimeLeft < 0 ? "Expired" : contract.StatusToString;
32
33                                <li>
34                                    <!-- Job Listing -->
35                                    <div class="job-listing width-adjustment">
36                                        <!-- Job Listing Details -->
37                                        <div class="job-listing-details">
38                                            <!-- Details -->
39                                            <div class="job-listing-description">
```

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 been 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 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.