Jobseeker website with AI

Oleksandr Mustakaiev

N00202022

Supervisor: Cyril Connolly

Second Reader: John Montayne

Year 4 2023/24

DL836 BSc (Hons) in Creative Computing

Abstract

This Project compares React.js + Vue.js to establish which Front-End Framework is optimal for website development. The project involves research in User Experience and Artificial Intelligence. The final application is a job seeker website that includes ChatGPT AI and allows user registration easier and faster. Include technologies – front-end and back-end

Keywords: AI, Front-End, Back-End, React, Tailwind CSS, MongoDB, Express, ChatGPT API

Acknowledgements

I extend my thanks to my supervisor, Cyril Connolly, for his support and guidance. He was checking my progress every week to make sure I was doing it on time, helping me with decisions. I am also thankful to everyone else who gave their knowledge and experience.

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not to leave copies of your own files on a hard disk where they can be accessed by other. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student : Oleksandr Mustakaiev

Signed: Oleksandr Mustakaiev

Failure to complete and submit this form may lead to an investigation into your work.

Table of Contents

1 Introduction

The Overall aim is to create a Job Seeker Website with a faster and easier profile register system. For the Application area, users can view all jobs and apply for them if they have registered their profile with their CV. Employers can create, edit, delete their jobs, and view who applied for their job offer. Technologies – For the Front-End – React, Tailwind CSS. For the Back-End – MongoDB, Express, pdf-parse and ChatGPT API. For hosting – Vercel (Front-End and Back-End), S3 Amazon (Images and Files).

2 Research

2.1 Introduction

This Project compares React.js + Vue.js to establish which Front-End Framework is optimal for website development. The project involves research in User Experience and Artificial Intelligence. The final application is a job seeker website that includes ChatGPT AI and allows user registration easier and faster.

2.2 User Experience

2.2.1 What is User Experience

Don Norman, the inventor of the term user experience, commented about 15 years ago [is still valid today] “I invented the term because I thought human interface and usability were too narrow. I wanted to cover all aspects of the person’s experience with the system including industrial design, graphics, the interface, the physical interaction, and the manual. This view is still valid today.

Since then the term has spread widely, so much so that it is starting to lose its meaning[…] People mention the term user experience often without having any idea why, what the word means, its origin, history, or what it’s about.” This reflects the complex nature of UX. (Hellweger & Wang, 2015) [1]

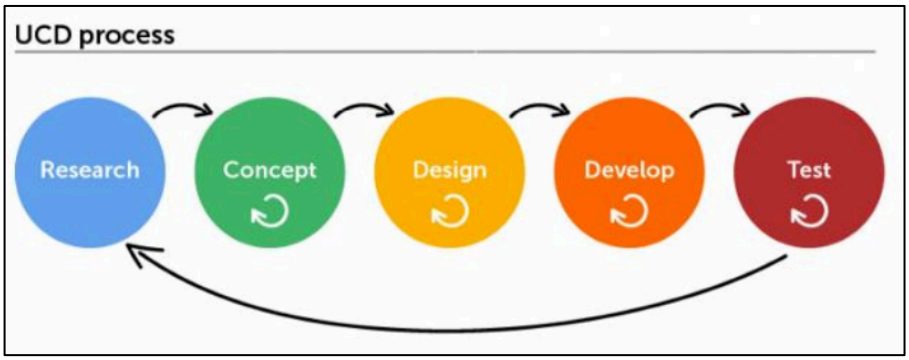

Figure 1 - *(Dang & Dat, 2020)* - Research to Concept to Design to Developer to Test and after testing go back to research and fix

It is no coincidence that UX, which includes both practice and research, is attracting a lot of attention right now. Our daily lives now contain a large number

of interactive products. Modern technology (sound, graphics, networking, miniaturisation, etc.) enables more than just functionality. Simultaneously, the parameters of demand for interactive products are being shifted by the expanding and evolving user base. From a UX standpoint, this change is taken very seriously. It is no accident that it emphasises positive, experiential, and emotional aspects in addition to functional ones. Commercial vendors who are aware of shifting business conditions, designers who value fresh design possibilities, and the scientific community's resurgence of interest in the affective system and its relationship to cognition are the main forces behind it. (Hassenzahl & Tractinsky, 2006)

2.2.2 UX applied to website development

The report describes that, similar to product design, web design is dependent on user experience. If the user experience (UX) of a website is different significantly from the expectations of its users, then no design will be able to convert those visitors into customers. Several UX elements in web design contribute to the ease of understanding and navigation of websites. (Goeva, 2019) [2]

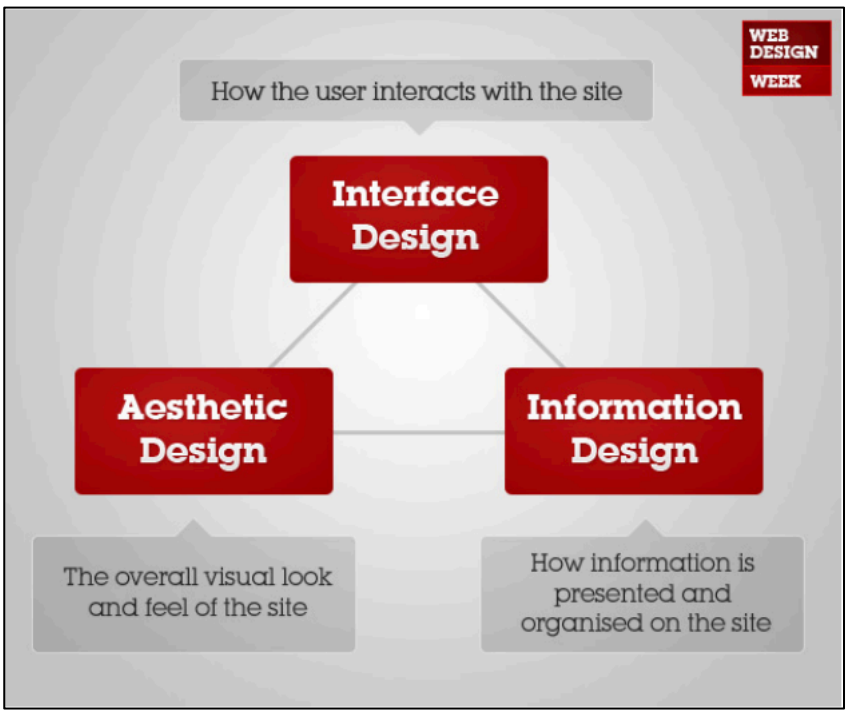

Figure 2 - Components of Web Design (Goeva, 2019) – How the user interacts with the site

According to the author, conversion rates on the website might rise by up to 200% with a well-designed user interface, and by up to 400% with an improved UX design. For this reason, certain businesses must adapt immediately and have a solid understanding of how UX design can transform their industry. (Dang & Dat, 2020)

2.2.3 Principles or Steps of UX Design

The author describes this research's aims of creating a system or mechanism that evaluates a web application according to set user experience (UX) parameters. The web application's overall rating will be determined by assigning a score to each of these parameters, which will then be used as the test standards for the web application's overall user experience. (Mistry & Rajan, 2019)

Investigating the locations where the product is used is the focus of the understanding context of the use phase. Developing scenarios for the use of the product helps in the development of an improved requirements list. The next phase focuses on defining user requirements, comprehending how the user uses the product, and figuring out why the user needs the product. In the following stage, design solutions are produced by following the specifications and guidelines established earlier in the development process. This section focuses on developing design solutions that are efficient without compromising usage. The next section's main objective is to compare the finished product to the requirements using a variety of evaluation techniques, including expert analysis, qualitative analysis, and quantitative analysis. After the product is released, this step focuses on identifying any design mistakes and errors. (Heikinmäki, 2020)

2.2.4 How UX works in React and Vue

The report describes React and Vue are libraries that are made to aid in creating interactive user interfaces for web pages. Both are modular and can be integrated with other JavaScript libraries. Modularity also enables to creation of

only parts of a web page with the library or the complete page can be created using these libraries. Pages with the majority if not all the page content created through these libraries are called Single-Page Applications (SPA). All the test subjects created for this thesis's test are Single-Page Applications by nature. (Pikkanen, 2021)**?**

2.2.5 Summary of User Experience (UX)

At the end of this section, the developer researched what is User Experience (UX), how it should work, how User Experience (UX) applied to website development and the steps of UX design.

2.3 React.js

2.3.1 What is React

The main of the study is that React is a library released in 2013 by META (formerly known as Facebook) and is a tool for creating reactive single-page applications. It was the first framework/library to utilize the Virtual DOM setup. It does not force any pattern and can be used with both class-based and functional programming. It utilizes a concept called JSX (JavaScript Syntax Extension) to structure its components. Throughout its time in existence, it has been frequently updated and has acquired a very large, third-party, library ecosystem. It refers to itself as Declarative, component-based and platform-independent with the more recent React Native. It is written in JavaScript but can utilize TypeScript through extensions. (Hidvég, 2022) Facebook engineers first developed React to address the difficulties they encountered when creating intricate user interfaces using dynamic datasets. This is no small task, and to function at Facebook's scale, it needs to be both scalable and maintainable. Facebook's ads division, which had been using a conventional client-side Model-View-Controller methodology, is actually where React got its start. (Gackenheimer, 2015)

2.3.2 Why React

React was designed to handle data visualisation in user interfaces. You would be justified in believing that the issue of displaying data in a user interface has already been resolved. The distinction lies in the fact that React was developed to support Facebook's and Instagram's expansive user interfaces, which involve dynamic data. Tools that are not part of React can be used to create and solve this kind of interface. These problems had to be resolved by Facebook before it developed React. However, Facebook did develop React because it had good logic and discovered that React could be applied to certain issues that arose during the development of intricate user interfaces. (Gackenheimer, 2015)

2.3.3 Development

React provides a lightweight and very efficient document object model. Despite not interacting with the program's generated DOM, it reacts to the document object model stored in memory. This causes the application to execute blazingly fast and powerfully. Most other web development frameworks allow for direct interaction with the program's DOM, resulting in full control over the DOM tree at each page activation point. Thus, when a large amount of data needs to be changed, the presentation is greatly impacted. ReactJS makes use of something called a virtual DOM, which is contrary to expectations. It functions in a very simple way. (Bhalla, Garg, & Singh, 2020)
React creates a dynamic and intelligent user interface for websites and mobile apps. Create fundamental views for every state in your application, and when your data changes, React will efficiently refresh and deliver the ideal portions. Having definitive perspectives improves the readability and debugging of your code. One can easily become used to the structure of ReactJS due to its simplicity and lack of complexity. React has a hand in many interesting moments.

Probably React's best feature is its comprehensibility. Even for those who are unfamiliar with it, it is quite understandable. Does the opposite, whereas other systems require learning a great deal about the structure itself, ignoring the

basics of language. Therefore, there are no barriers or problems that prevent the structure's initial skill levels from being further developed. (Rawat & Mahajan, 2020)

2.3.4 Summary

In this section developer, figure out what is React, why developers use React instead of other frameworks that are available and how it works while creating an app.

2.4 Vue.js

2.4.1 What is Vue

The report described Vue itself as “An approachable, performant and versatile framework for building web user interfaces”. It is a community-driven and developed web application framework sponsored by various organizations. The first release of Vue was in 2014 and has received several updates since its first release. It is built in TypeScript but uses the native JavaScript language and focuses on declarative rendering and component composition. However, it is possible, and recommended, to make use of TypeScript for Vue applications. (Hidvég, 2022)

After researching, the developers of Vue.js refer to it as a progressive framework. This is because the core Vue.js library only concentrates on the view layer, so you can start developing your app with less effort. As the needs increase over time, you can add more libraries to enhance functionality. (Nelson, 2018)

2.4.2 Why Vue

According to this article, the author found that Vue is a progressive framework for creating user interfaces that can power complex Single-Page Applications on their own when combined with its Single-File Components. It is also said to be easily learned, view layer-focused, incrementally adoptable, and easy to integrate with other libraries or existing projects. Through the use of a Virtual

DOM, Vue makes it possible to represent the real DOM as JavaScript objects that can be updated and changed intelligently without requiring a page reload. (Kumpulainen, Web application development with Vue.js, 2021)

2.4.3 Development

While using TypeScript with Vue 3 is still optional, it has near-perfect integration with the JavaScript combination and can be supported without the need for additional tooling. Previous versions of Vue were written in plain JavaScript. However, Vue 3 is a complete rewrite of the entire Vue codebase in TypeScript. While most of the theoretical concepts discussed remain valid for Vue 2, the provided configurations and code examples cannot be used as a reference when developing with older versions of Vue because the learning tool is specifically designed for Vue 3, and all related information has been validated using the Vue CLI version 4.5.11. Since all the technologies covered in the learning tool were used in its creation, the version of each specific API for which the data provided is valid can be found in the project's package.json file. (Kumpulainen, Web application development with Vue.js, 2021)

2.4.4 Summary

In this section, the developer researched what is Vue, why developers use Vue and how it works while creating an app.

2.5 Comparing Vue v React

According to the book, you can create a basic app using the same tools found in both Vue and React. Create-React-App is a feature of React that builds a React application by setting up a webpack but keeps the configuration hidden from the user. With the help of vue-cli (a globally installed npm package that provides the vue command in your terminal), which comes with several templates, you can install Vue using Webpack, Browserify, or no build tool at all. Unlike create-react-

app, which does not include vue-router, unit testing, and functional testing templates are also configurable. There are a lot of similarities between Vue and React. (Macrae, 2018)

2.6 Artificial Intelligence

2.6.1 What is Artificial Intelligence

According to this author's article, Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to biologically observable methods. (McCarthy, 2004)

It is a system that mimics various functions which a human can do. AI uses external data like big data to achieve excellent performance for the given tasks. Once, AI was just a concept that was seen in science fiction and debates which discussed the effect of technology in the modern world. But now, it has become a part of us in our day-to-day life. It has become the key function of many technical and various other sectors. Artificial Intelligence has a significant impact on industries like manufacturing, healthcare, supply chains etc. The ability of AI to do things which humans can', bring many applications which result in improved performance and productivity. (K, 2021)

2.6.2 What is ChatGPT

ChatGPT is an AI-powered chatbot platform developed by OpenAI that has the potential to revolutionize how people interact with technology. This revolutionary platform utilizes natural language processing (NLP) and machine learning (ML) algorithms to enable users to communicate with machines conversationally has recently been introduced to the public with great fanfare and promises to revolutionize how people interact with technology. With their ability to comprehend context, intent, sentiment, and more, ChatGPT enables

users of all ages and backgrounds to communicate naturally in a variety of languages without having any prior knowledge or experience in programming or computer science. It may be used for many things, like customer service, entertainment, education, finance, health care, and more. (Gabrio, Hovan, & Shaji, 2023)

3 Requirements

3.1 Introduction

The purpose of the requirements phase is to allow developers to work out what the application should be able to do. However, It is also important to understand what the users would like the application.

After researching which front-end framework is most suitable, a decision was taken to develop in React.js. The application involved the creation of a Job seeker website where people can upload their CVs and a profile will be automatically created. To make it happen, the chatGPT API will be connected to the Back-End.

3.2 Requirements gathering

3.2.1 Similar applications

1. "jobsireland.ie"

Home Page

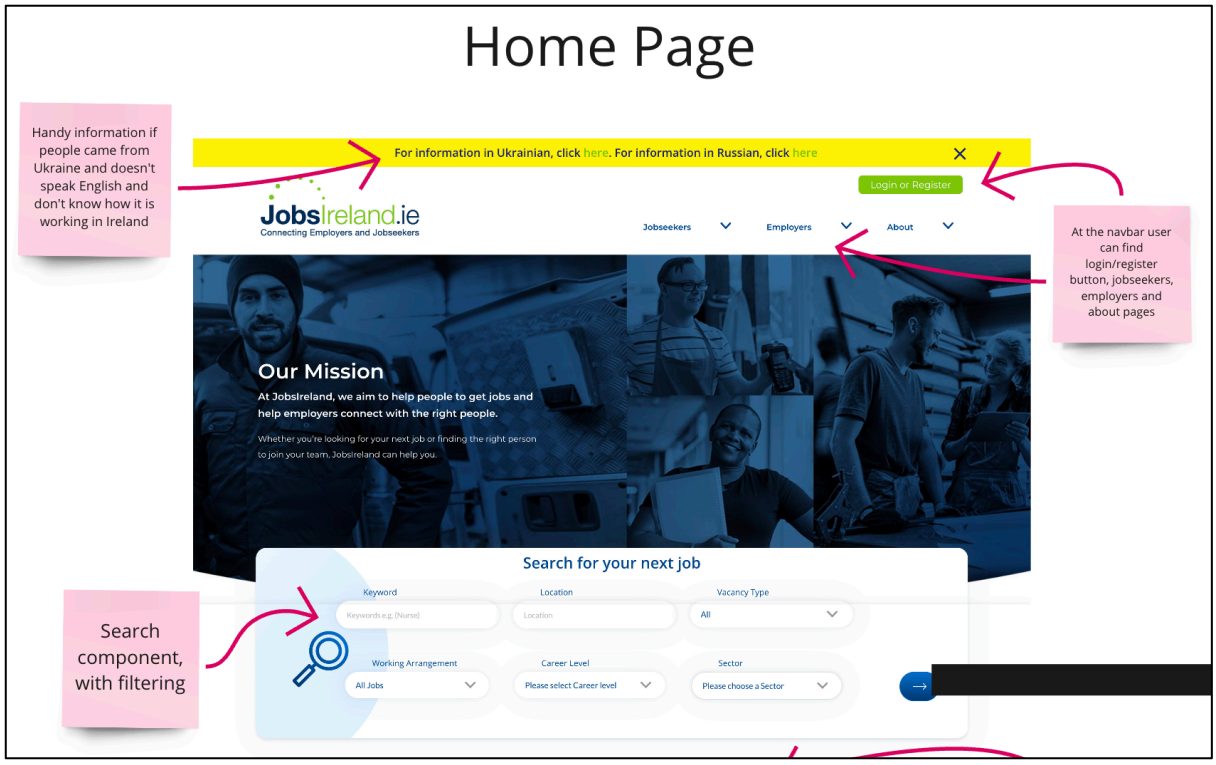

Figure 3 - jobsireland.ie - Home Page

- At the navbar user can find the Login/Register button, and sections such as ***Jobseekers***, ***Employers*** and ***About***.
- Search component with filtering, such as keywords, location, vacancy type, working arrangement, career level and sector.

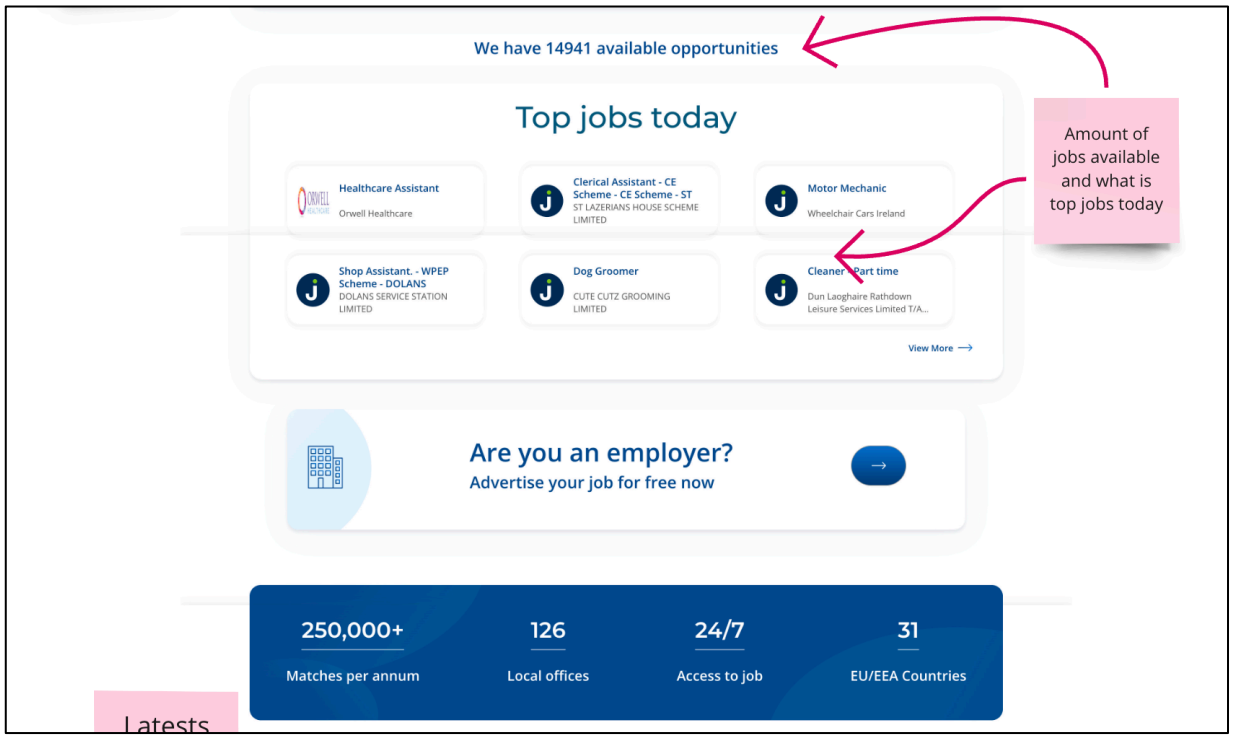

Figure 4 - jobsireland.ie - Home Page

- Amount of jobs available and what are the top jobs for today

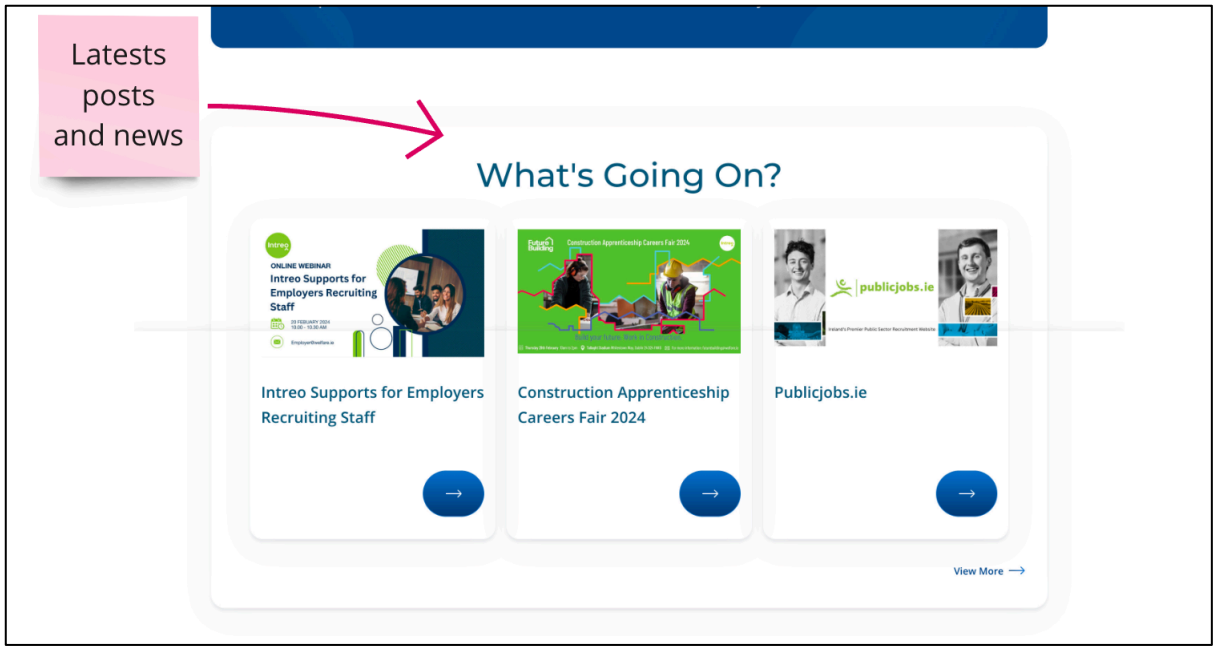

Figure 5 - jobsireland.ie - Home Page

- Latest posts and news

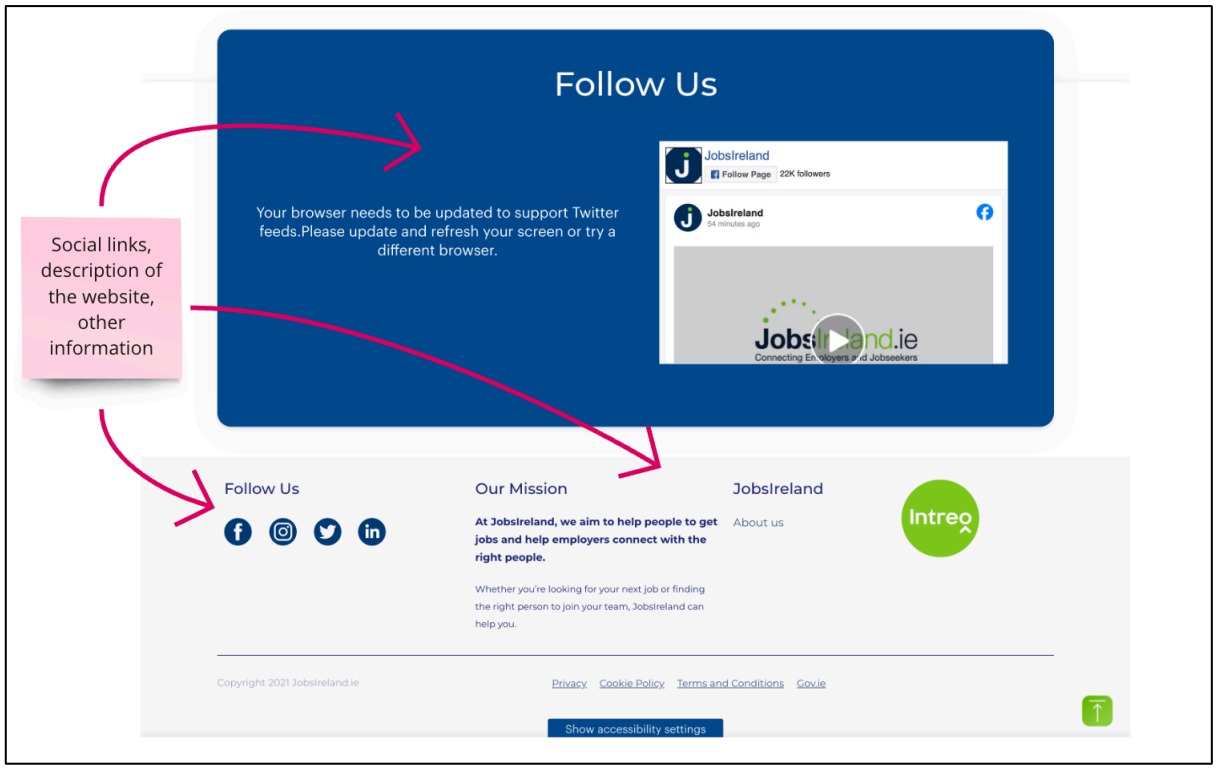

Figure 6 - jobsireland.ie - Home Page

- Social Links, description of the website and other information

Search Page

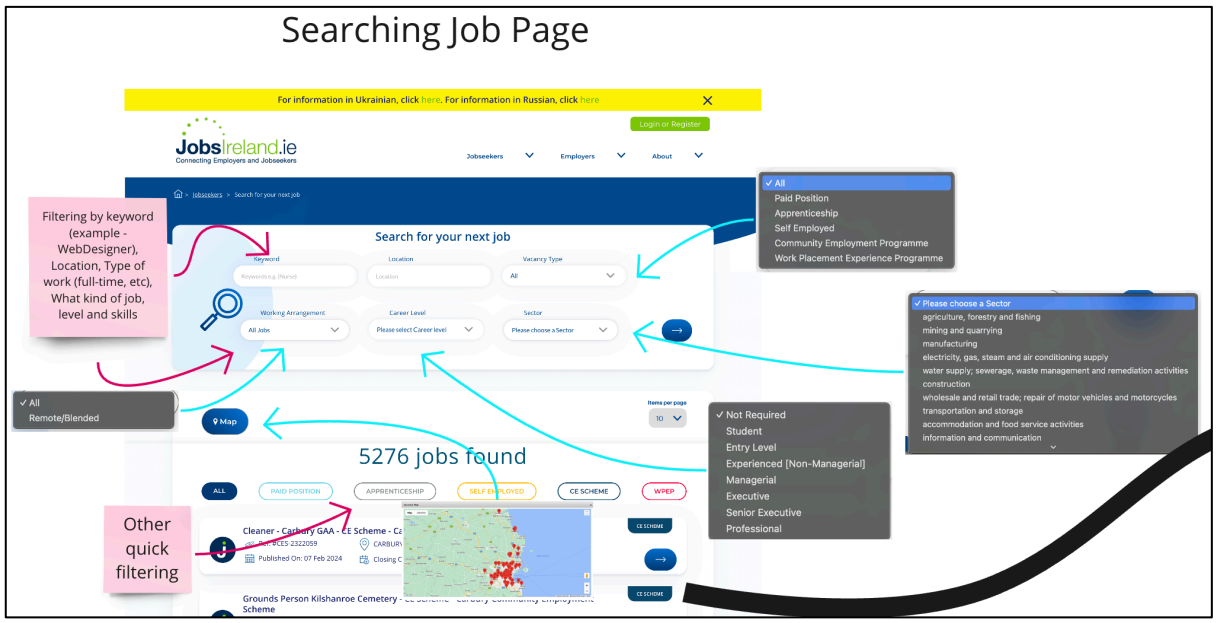

Figure 7 - jobsireland.ie - Search Page

- Searching and filtering by keywords (for example – Web Designer), location, type of work, what kind of job, level and skills
- Underneath some other quick filtering
- Map with jobs available

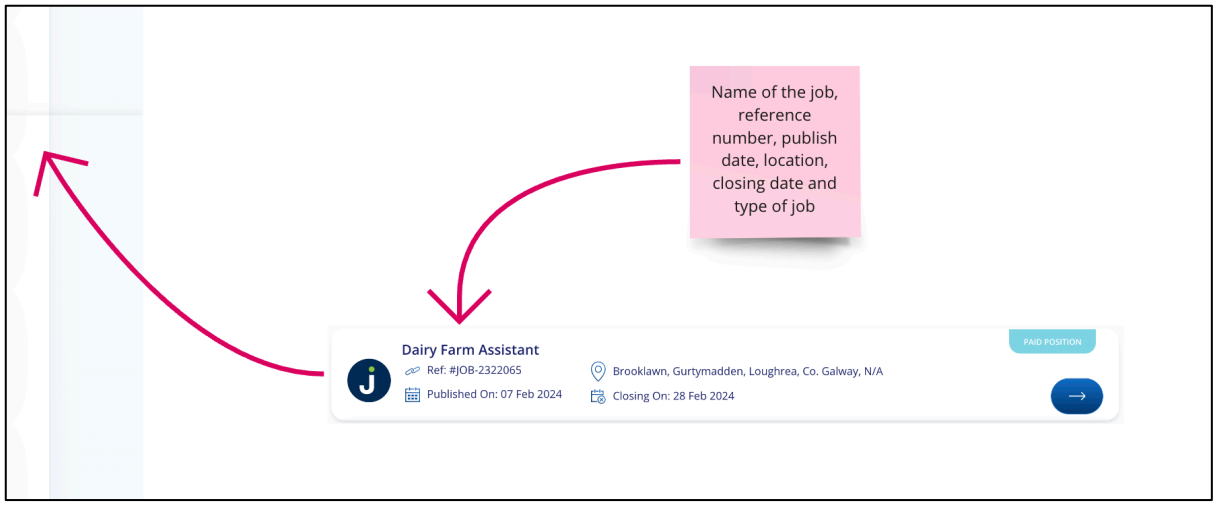

Figure 8 - jobsireland.ie - Search Page

- Each component of the job has: the name of the job, reference number, publish date, location, closing date and type of job

Job Page

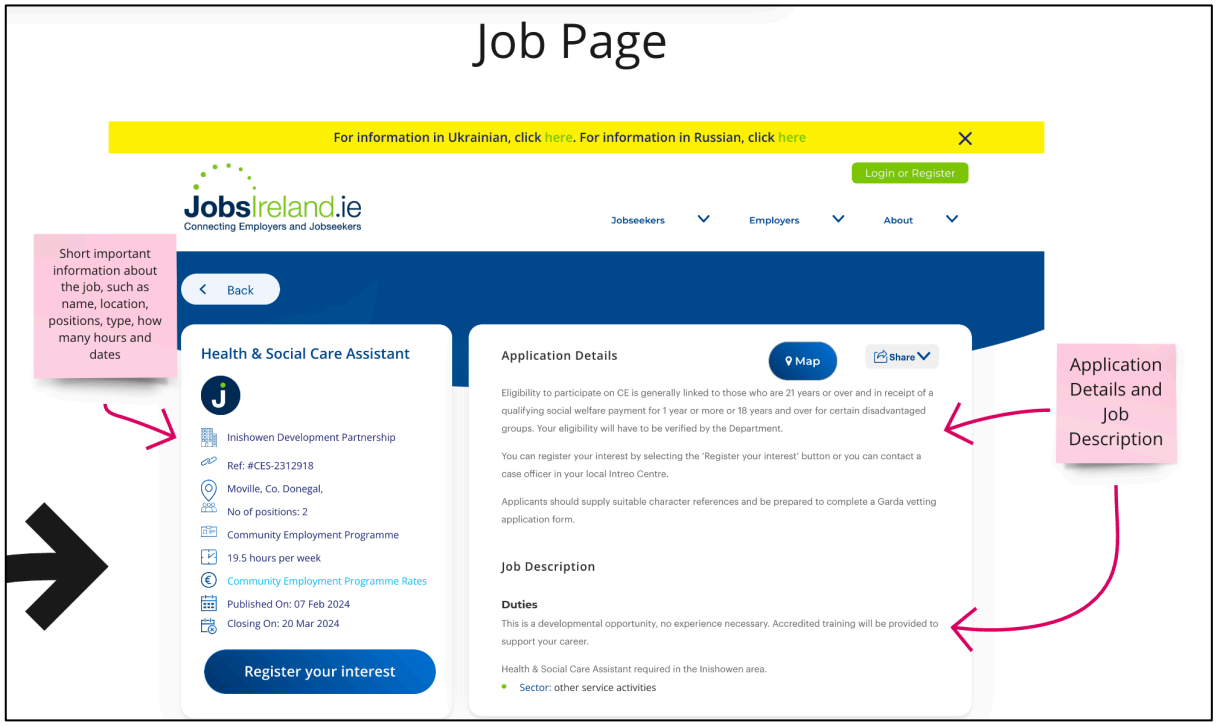

Figure 9 - jobsireland.ie - Job Page

- On the left-hand side, short important information about the job, such as name, location, position, type, how many hours and dates
- On the right-hand side, application details and job description

Advantages:

1. Easy and simple design
2. Good flow pages
3. Map in the search page with available jobs
4. Can retrieve a previously viewed job using a reference number
5. Share URL link

Disadvantages:

1. Doesn’t show the salary
2. No sort-by button

2. “ie.indeed.com”

Home Page

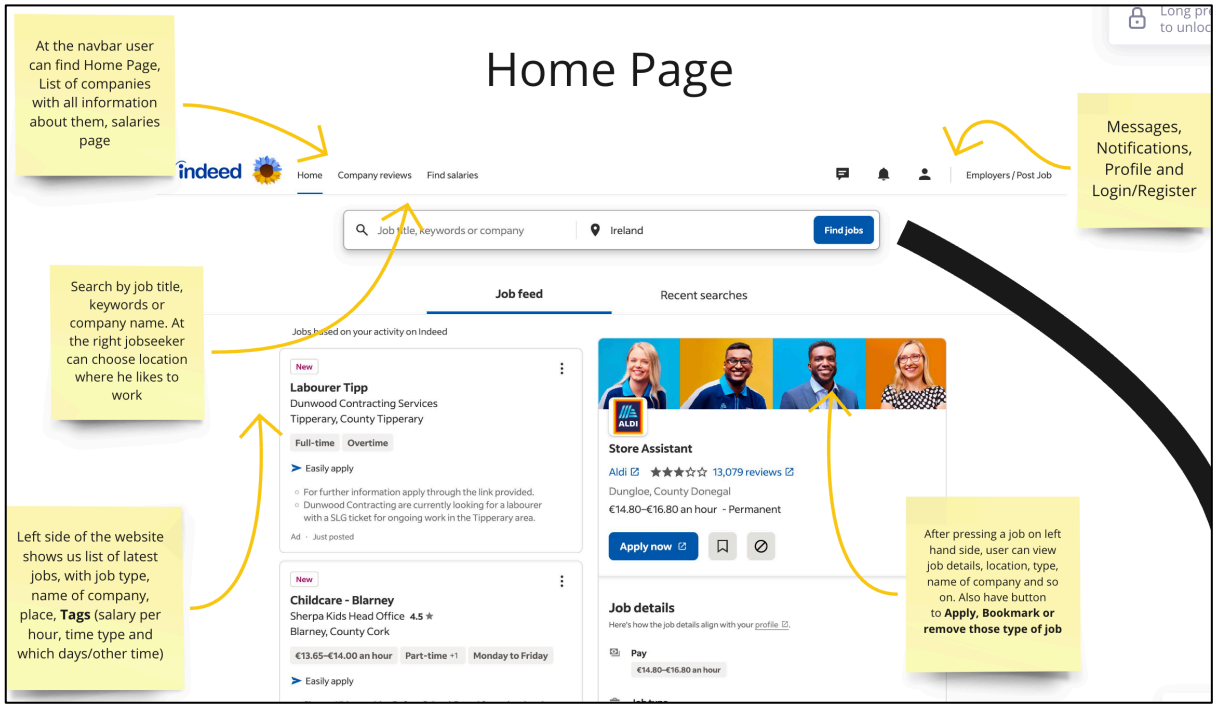

Figure 10 - ie.indeed.com - Home Page

- At the navbar, the user can find the ***Home Page***, a list of companies and a salaries page
- Search by job title, keywords or company name. At the right of the page, the jobseeker can choose a location to work
- The left side of the website shows us a list of the latest jobs, with job type, name of company, place, and Tags (salary per hour, time type and which days/other time)
- Messages, Notifications, Profile and Login/Register included
- After pressing a job on the left-hand side of the page, the user can view job details, location, type, name of the company and so on. Also have a button to **Apply, Bookmark or remove those types of job**

Search Page

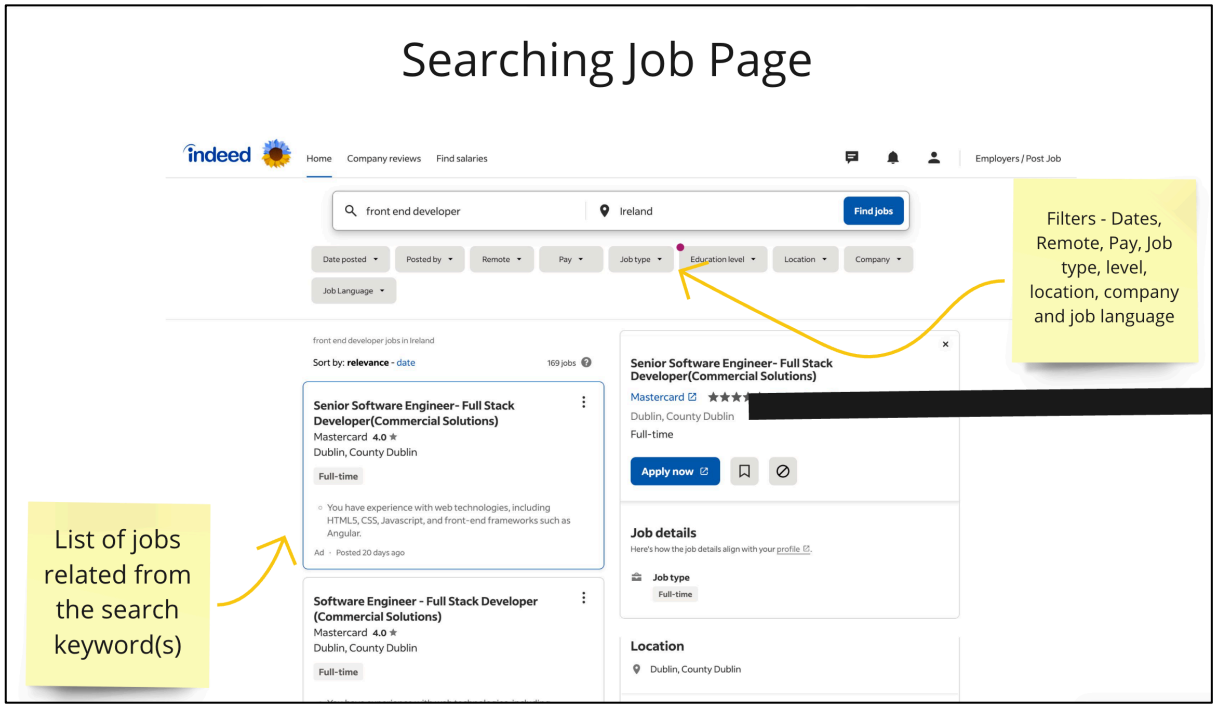

Figure 11 - ie.indeed.com - Search Page

- List of jobs related to the search keyword(s)
- Filters - Dates, Remote, Pay, Job type, level, location, company and job language

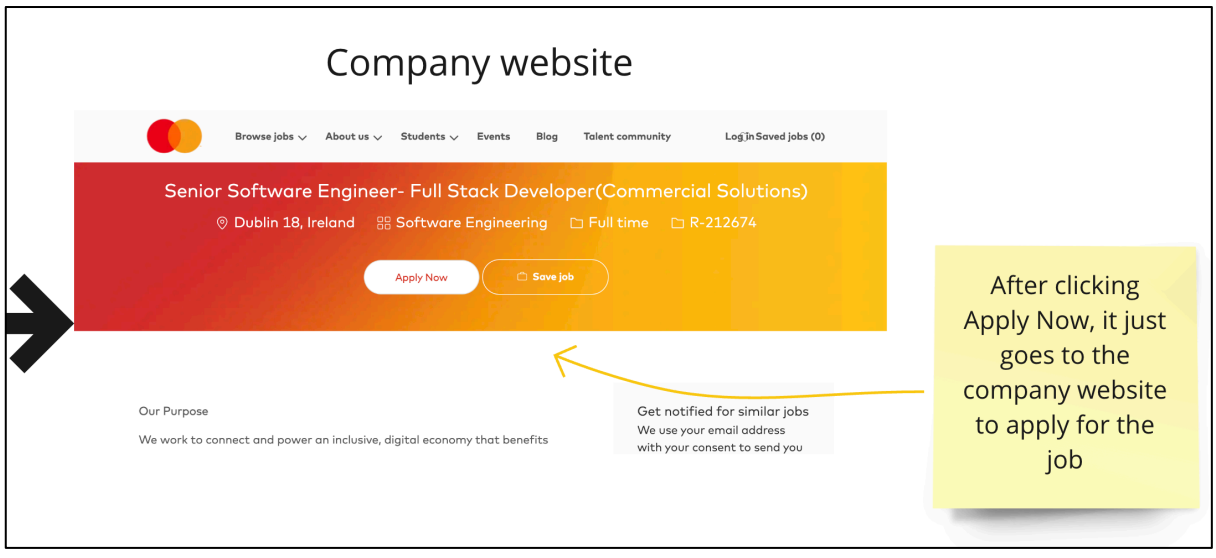

Figure 12 - ie.indeed.com - Company Website

- After clicking ***Apply Now*** on the search page, it goes to the company website to apply for the job

Advantages:

1. The small amount of pages saves time for job applicants
2. Simple design
3. Advertised Jobs are the first to be viewed, which means the website company makes money on this.
4. After clicking ***Apply Now***, the user is brought to the company website and then the user can apply for the post.

Disadvantages:

1. Doesn't show on the home page who are they and what the website is about
2. Easy to lose the jobs feed. The user needs to scroll up to view the next job
3. Need to create an account but the jobseeker can just go straight away to the company website to apply for the job

3. "jobs.ie"

Home Page

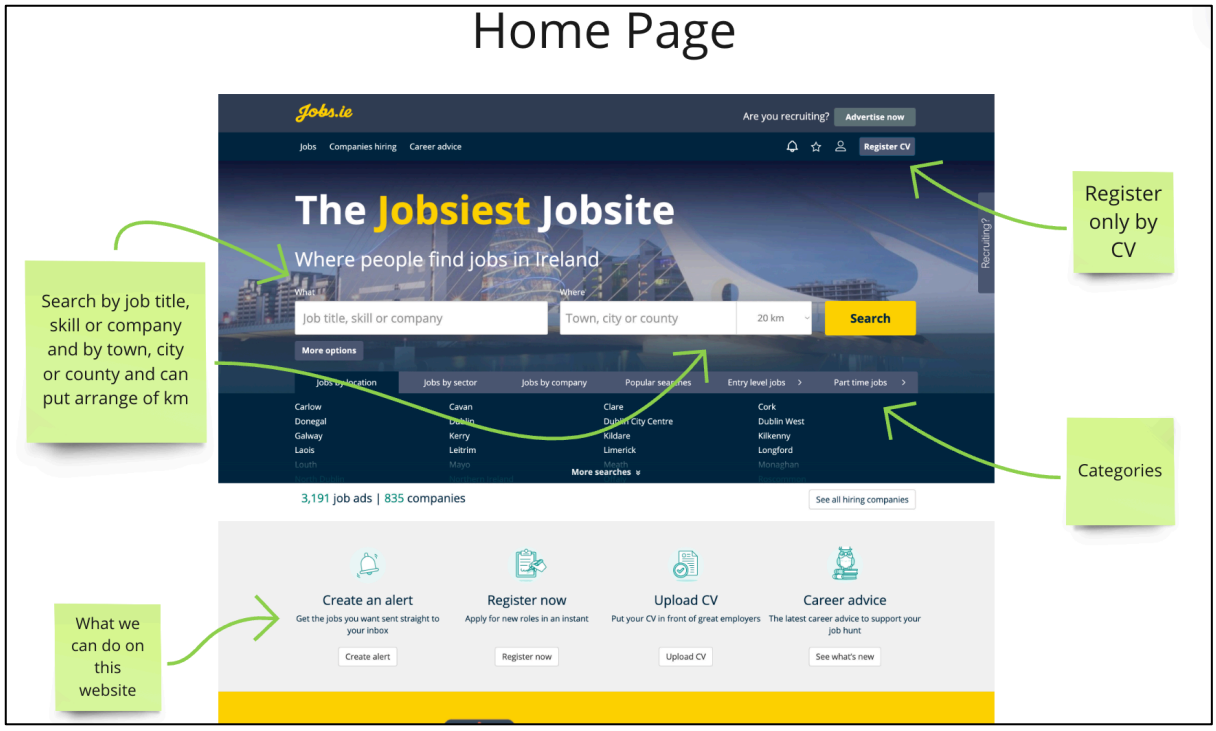

Figure 13 - jobs.ie - Home Page

- Search by job title, skill or company and by town, city or county and can put arrange of km
- What we can do on this website
- Register only by CV
- Categories

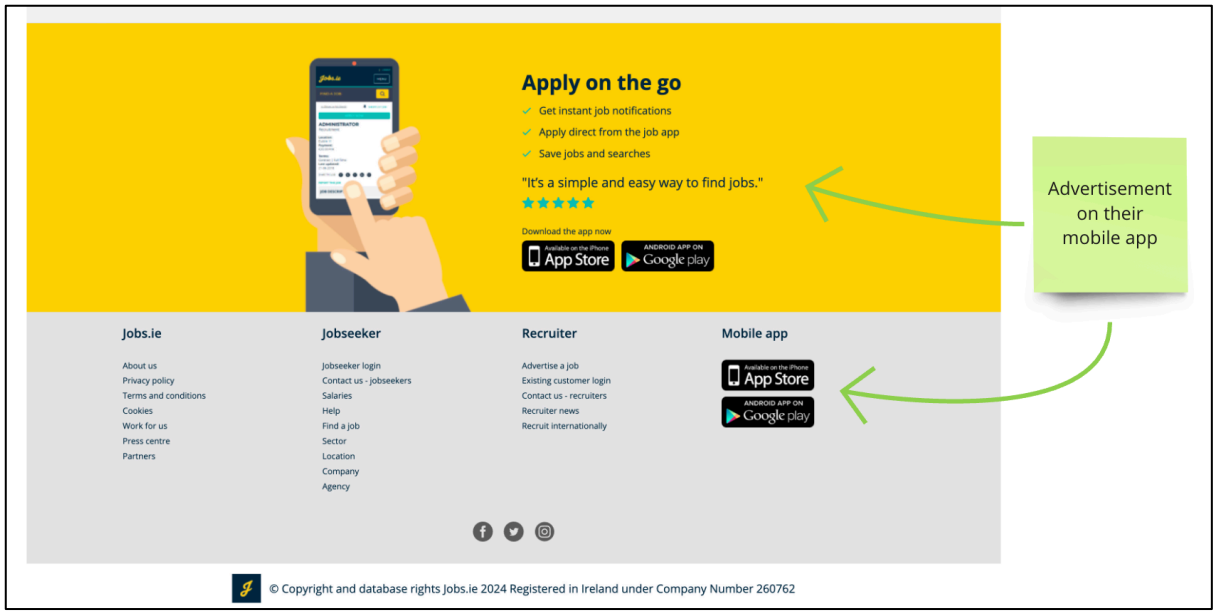

Figure 14 - jobs.ie - Home Page

- Advertisement on their mobile app

Search Page

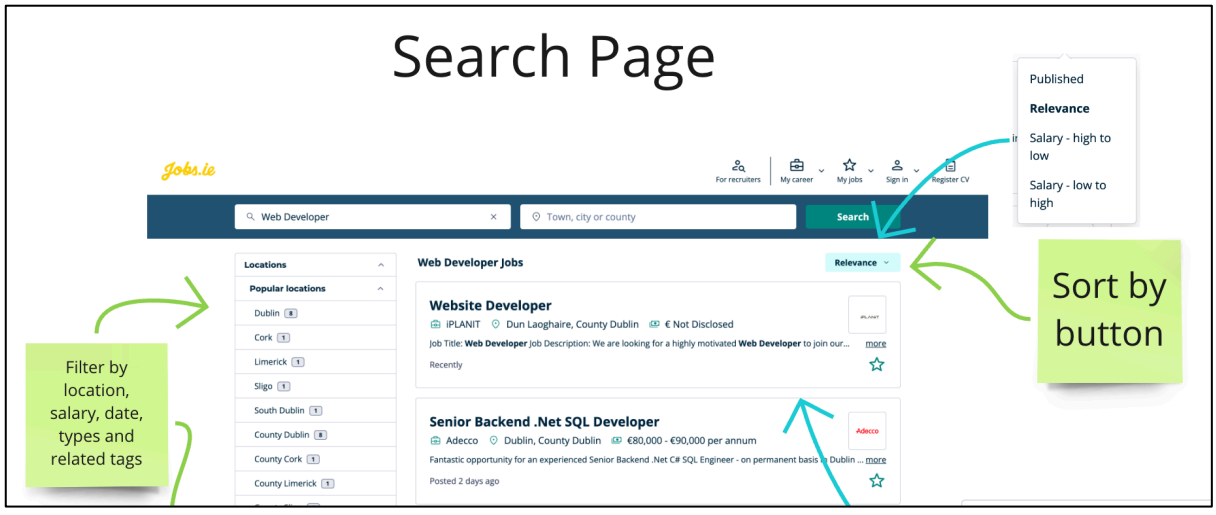

Figure 15 - jobs.ie - Search Page

- Filter by location, salary, date, types and related tags
- Sort by button

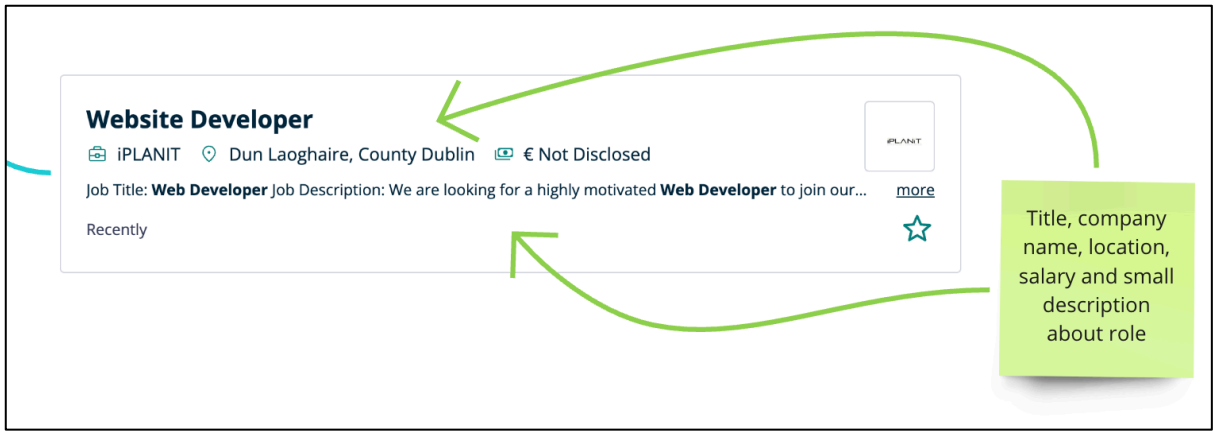

Figure 16 - jobs.ie - Search Page

- Title, company name, location, salary and a small description of the role

Job Page

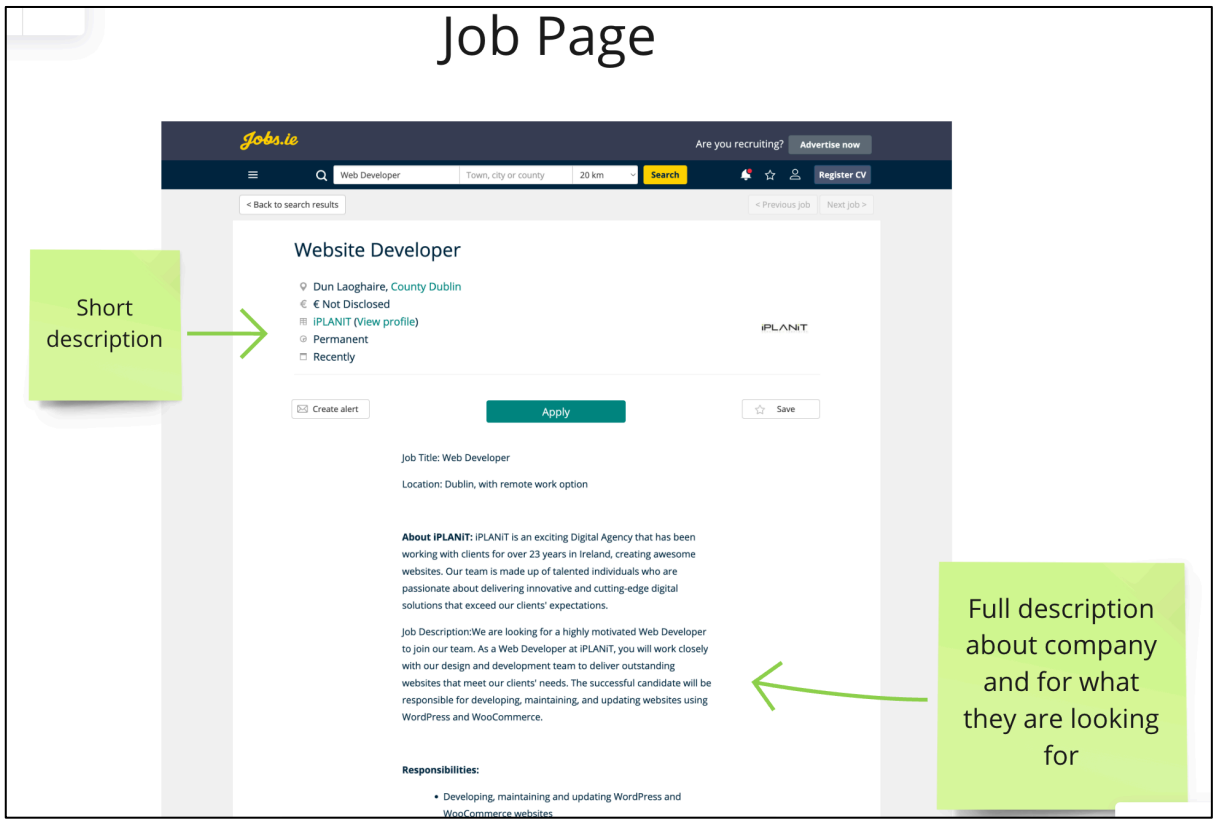

Figure 17 - jobs.ie - Job Page

- Short description at the top
- Full description of the company and what they are looking for

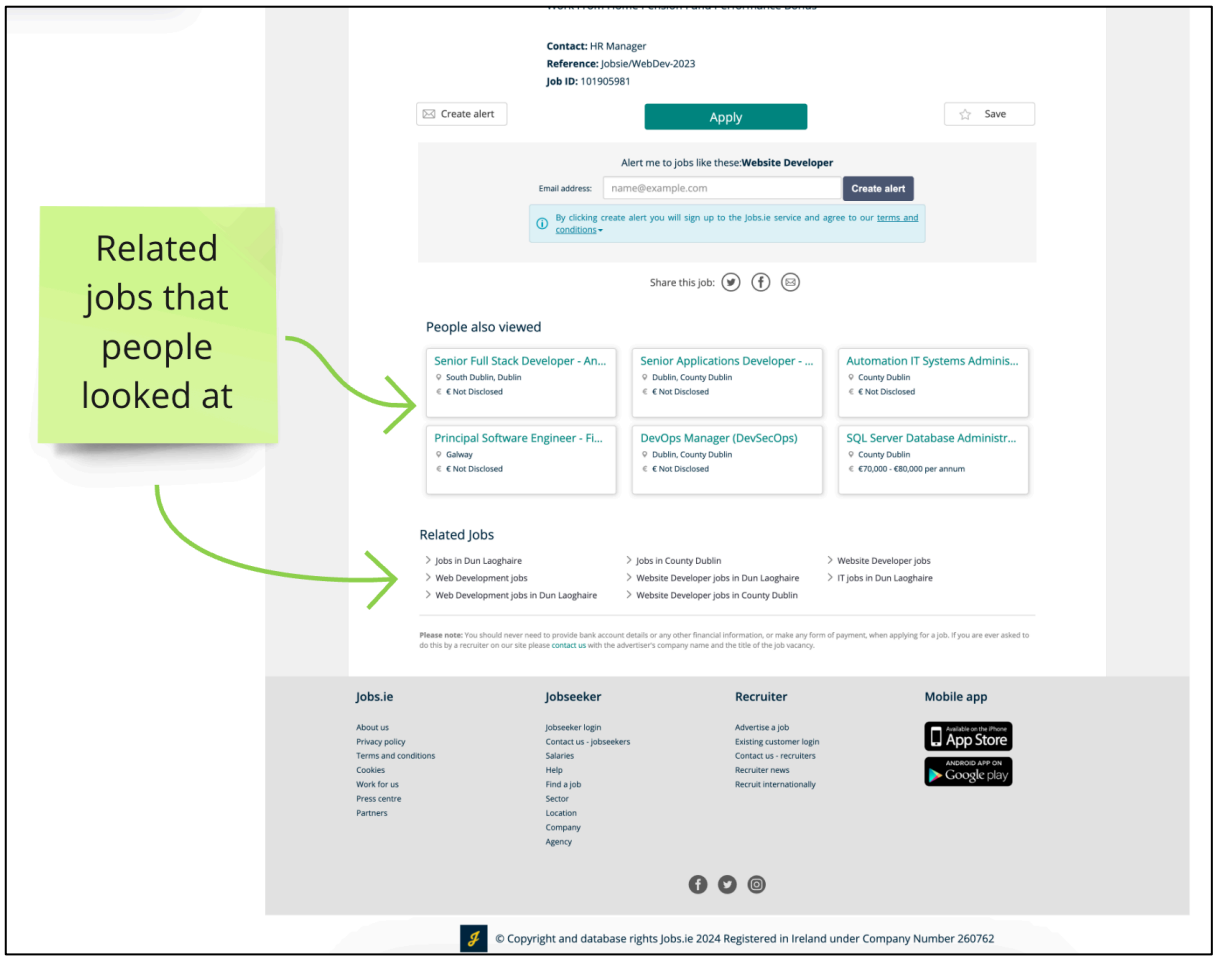

Figure 18 - jobs.ie - Job Page

- Related jobs that people looked at

Advantages:

1. Good flow
2. On the ***Home Page*** in the search bar, the user can put how far they want to look for the job
3. Sort by button on the ***Search Page***
4. Bookmark a job
5. Alert a person to jobs he viewed

Disadvantages:

1. An old design impression
2. Too much information on the ***Home Page*** when the user enters the website

3.3 Requirements modelling

3.3.1 Personas

These are fictional characters to help the developer understand the users' needs. They also help identify who the relevant users are.

3.3.2 Functional requirements

- When a job seeker/user uploads his CV it will create a profile, researched from their CV
- Admin section can delete all jobs
- Employer/Company can create/edit/delete jobs
- All users can login/register
- ChatGPT API functionality
- If a user doesn't have an account, it will ask to login/register to create a profile, apply for jobs, etc.

3.3.3 Non-functional requirements

- Security
- Performance
- Manageability
- Usability

3.3.4 Use Case Diagrams

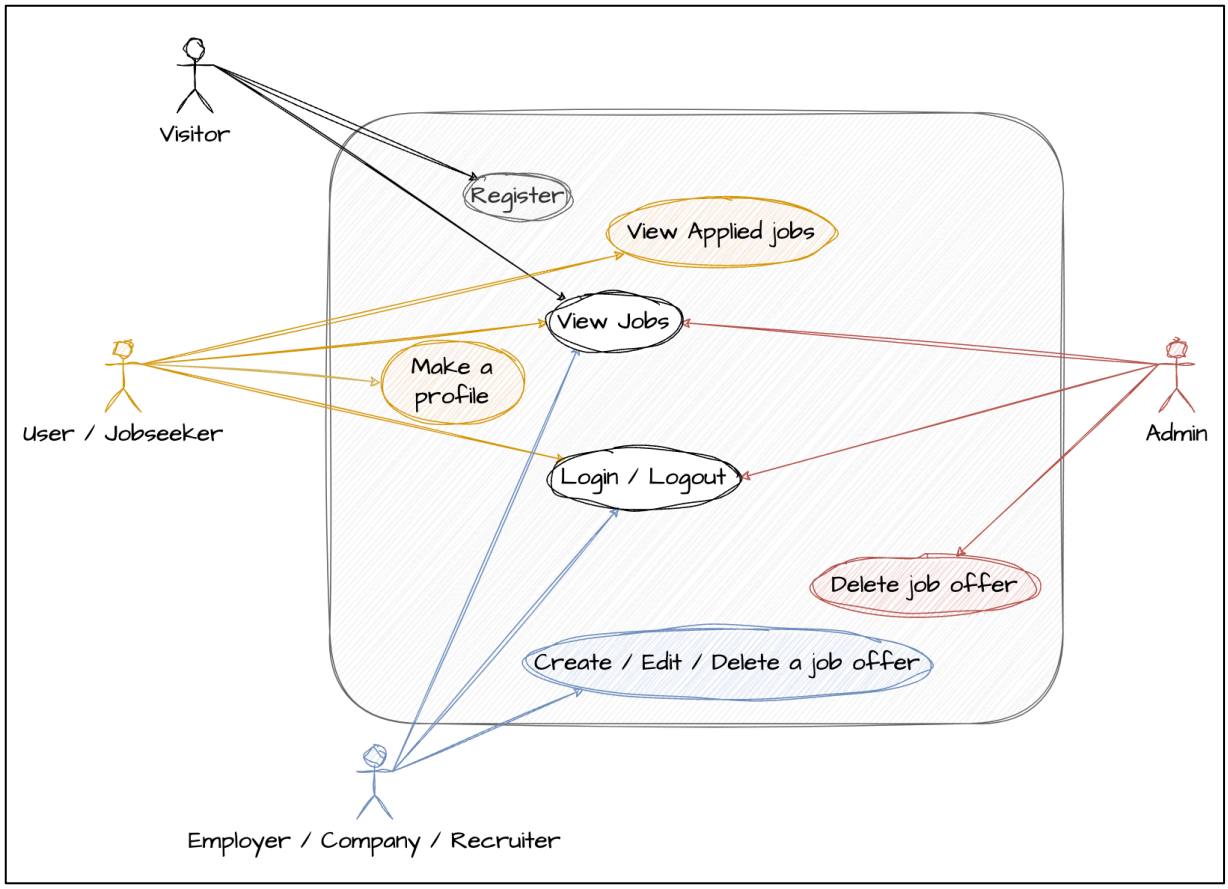

Figure 19 - Use Case Diagram – 4 Roles, Visitor, User, Employer and Admin

As a Visitor – Register, View Jobs

As a User/Job Seeker – Login/Logout, Make a profile, Apply to jobs, View Applied Jobs

As an Employer/Company – Login/Logout, View Jobs, Create/Edit/Delete their job offers

As an Admin – Login/Logout, View Jobs, Delete all job offers

3.4 Feasibility

The developer will use MongoDB and Express to create a database for the back end and Insomnia to test the API. The developer needs to extract text from the PDF file, connect chatGPT API with the extracted text and return a JSON file. For

the Front-End an app in React using Tailwind CSS will need to be created. Finally, the Back-End and Front-End will be connected. The developer will host the Back-End using Vercel and S3 Amazon for images. The Front-End hosting will involve Vercel or Google Firebase.

3.5 Conclusion

The developer researched 3 similar websites and created 3 personas related to the website's UX problem. ERD define and use case diagrams were also created using draw.io.

4 Design

4.1 Introduction

For the design section, the developer will create ERD, Low Fidelity and High Fidelity designs for the website. There are 9 tables in ERD, 6 pages and 2 modals

4.2 Program Design

4.2.1 Technologies

The technologies being used to design this application are Figma and draw.io. This technology was chosen because, Figma is a collaborative web application for interface design, with additional offline features enabled by desktop applications for macOS and Windows.
Diagrams.net (draw.io) is a cross-platform graph drawing software developed in HTML5 and JavaScript. Its interface can be used to create diagrams such as flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams.

4.2.2 Structure of Express MongoDB

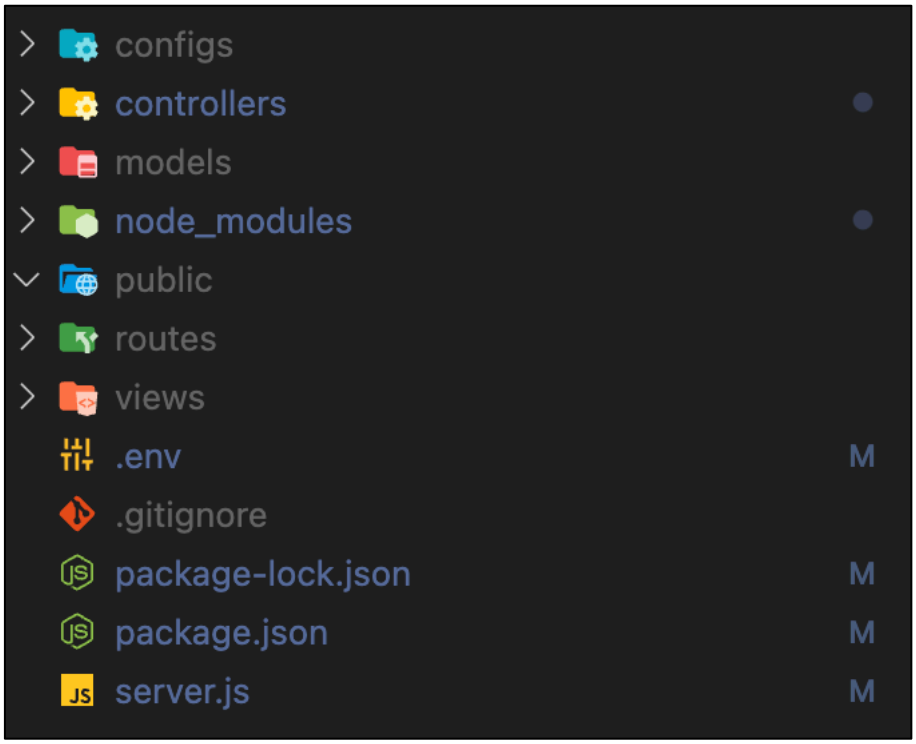

Figure 20 - Structure of Express MongoDB

In the config folder, there is a db.js file, which where developer uses to connect MongoDB with Express. The controller's folder, where all controller files are stored, is the same as for models and routes. Views folder includes index.html to test extracting text from PDF files. Server.js is the main file where all links, connections and middleware are located.

4.2.3 Database design

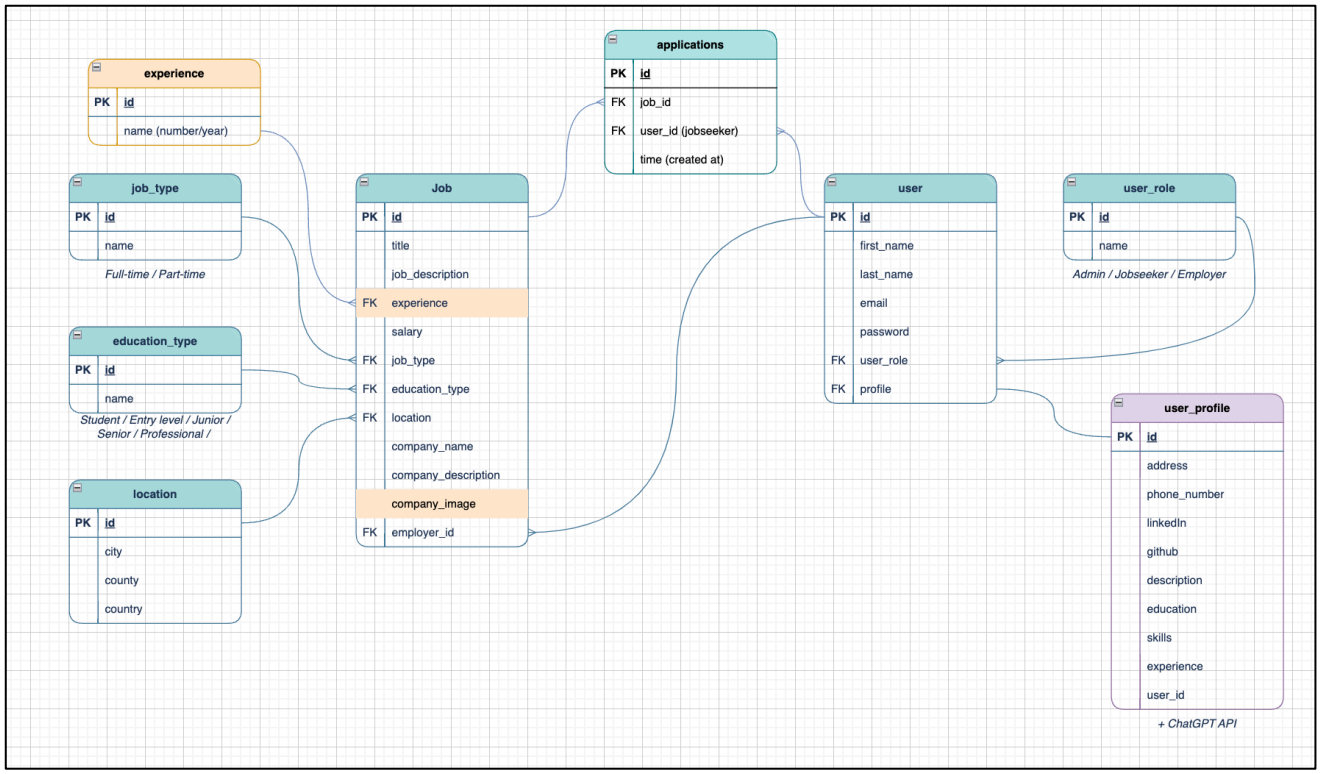

Figure 21 - Database Design

4.3 User interface design

The user interface must be simple for users and easy to use. It should be quick for users to find what they're looking for. This might include a logical content setup, clear search functionality, and simple navigation menus.

Give users clear feedback when they complete tasks, such as updating their profile or submitting a job application. Error messages should be informative and help users in finding out how to fix their errors.

The website should work smoothly and load fast. Reduce the number of useless animations and large file sizes that can make the user experience delay.

4.3.1 Wireframe

Figure 22, At the ***Home Page***, there is welcoming text and the user can look for the job including ***searching***, ***filtering*** and ***sorting*** system.

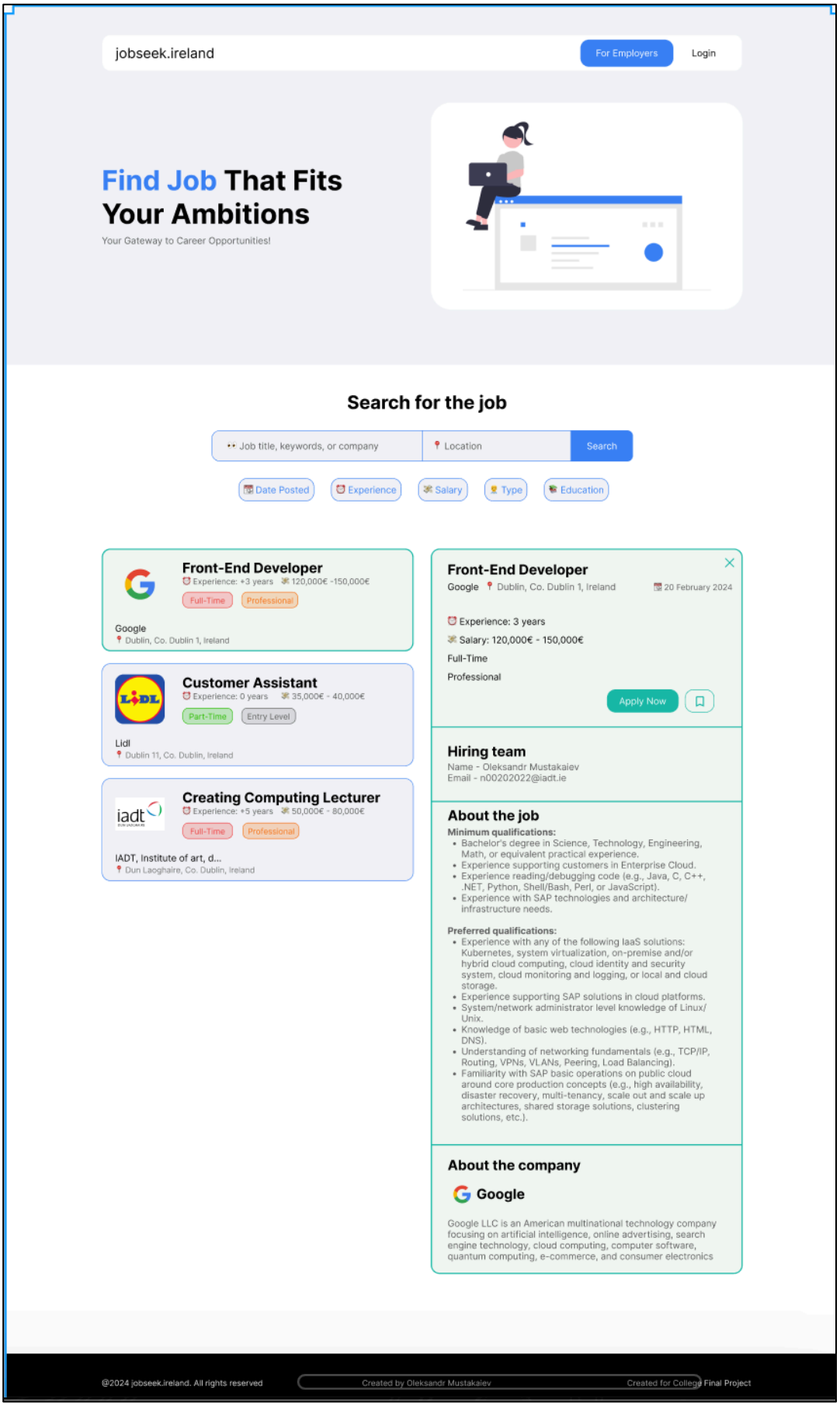
jobseek.ireland
For Employers
Login
Find Job That Fits Your Ambitions
Your Gateway to Career Opportunities!
Search for the job
Job title, keywords, or company
Location
Search
Date Posted
Experience
Salary
Type
Education
Front-End Developer
Experience: +3 years
120,000€ -150,000€
Full-Time
Professional
Google
Dublin, Co. Dublin 1, Ireland
Customer Assistant
Experience: 0 years
35,000€ - 40,000€
Part-Time
Entry Level
Lidl
Dublin 11, Co. Dublin, Ireland
Creating Computing Lecturer
Experience: +5 years
50,000€ - 80,000€
Full-Time
Professional
IADT, Institute of art, d...
Dun Laoghaire, Co. Dublin, Ireland
Front-End Developer
Google
Dublin, Co. Dublin 1, Ireland
20 February 2024
Experience: 3 years
Salary: 120,000€ - 150,000€
Full-Time
Professional
Apply Now
Hiring team
Name - Oleksandr Mustakaiev
Email - n00202022@iadt.ie
About the job
Minimum qualifications:
Bachelor's degree in Science, Technology, Engineering, Math, or equivalent practical experience.
Experience supporting customers in Enterprise Cloud.
Experience reading/debugging code (e.g., Java, C, C++, .NET, Python, Shell/Bash, Perl, or JavaScript).
Experience with SAP technologies and architecture/ infrastructure needs.
Preferred qualifications:
Experience with any of the following IaaS solutions: Kubernetes, system virtualization, on-premise and/or hybrid cloud computing, cloud identity and security system, cloud monitoring and logging, or local and cloud storage.
Experience supporting SAP solutions in cloud platforms.
System/network administrator level knowledge of Linux/ Unix.
Knowledge of basic web technologies (e.g., HTTP, HTML, DNS).
Understanding of networking fundamentals (e.g., TCP/IP, Routing, VPNs, VLANs, Peering, Load Balancing).
Familiarity with SAP basic operations on public cloud around core production concepts (e.g., high availability, disaster recovery, multi-tenancy, scale out and scale up architectures, shared storage solutions, clustering solutions, etc.).
About the company
Google
Google LLC is an American multinational technology company focusing on artificial intelligence, online advertising, search engine technology, cloud computing, computer software, quantum computing, e-commerce, and consumer electronics
@2024 jobseek.ireland. All rights reserved
Created by Oleksandr Mustakaiev
Created for College Final Project

Figure 22 - Home Page

Figure 23***, Authentication modal*** for Employer. It is the same for Users, but it ***gives a role for a user and employer***.

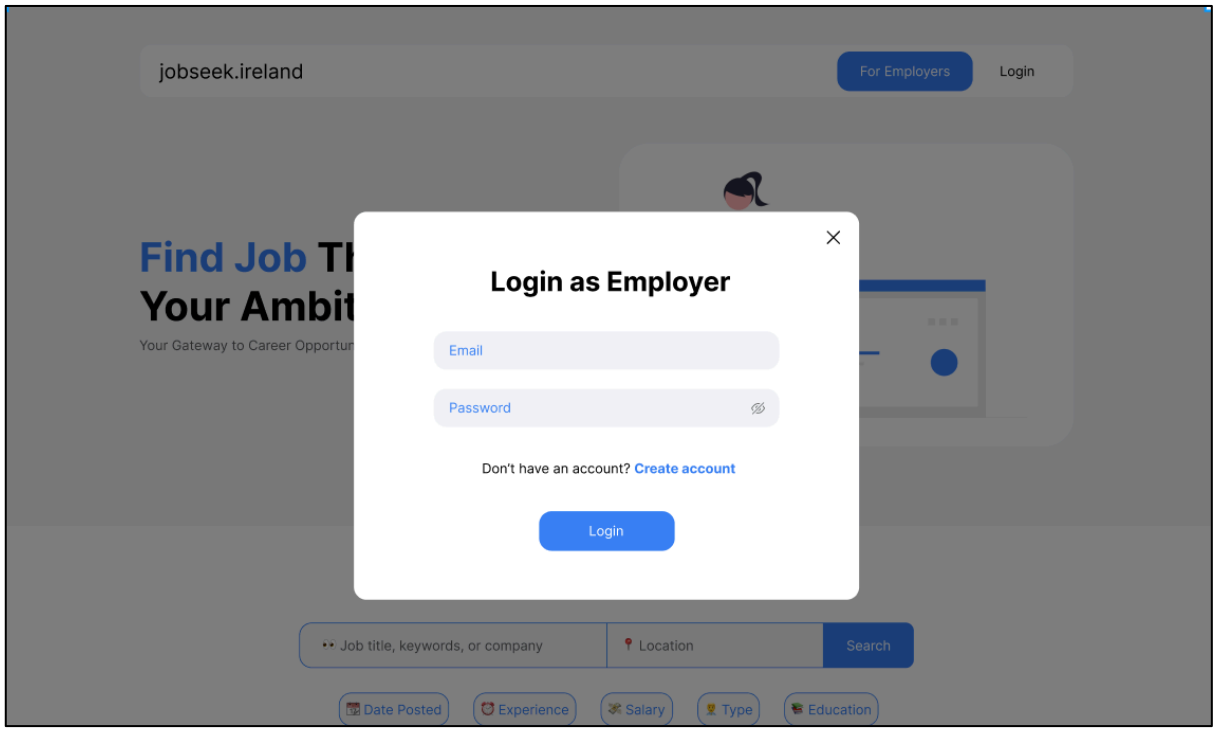

Figure 23 - Employer/User Login/Register modal

Figure 24, After logging in as a user, the jobseeker needs to ***make a profile*** to apply for the jobs. There are ***2 ways*** to create a profile: 1. Create using their CV 2. Create manually

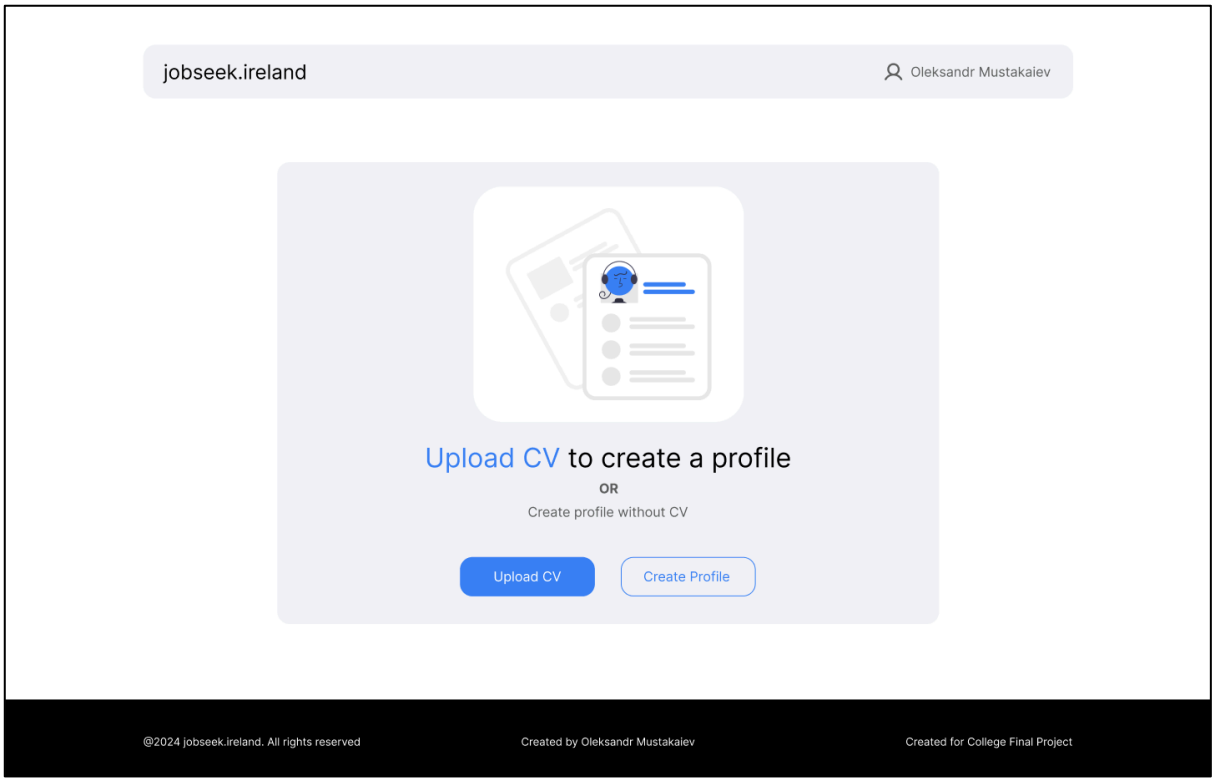

Figure 24 - Upload CV Page

Figure 25, A ***form for creating a profile*** for a user. If the job seeker uses “Upload CV” it will ***auto-fill*** their profile, using their CV information.

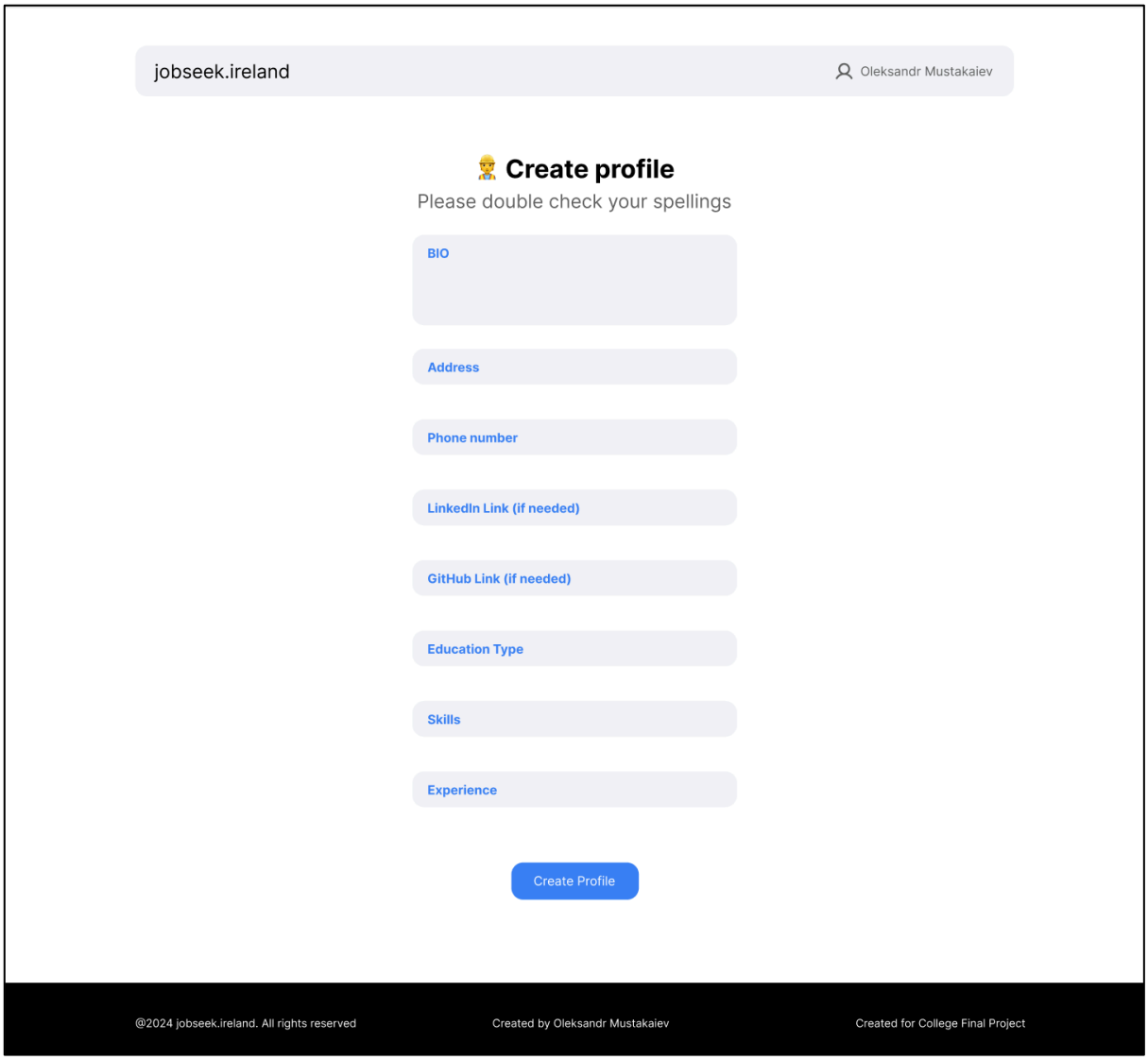

Figure 25 - Upload CV (2)/Profile Page

Figure 26, shows the ***user's profile*** and ***applied jobs***.

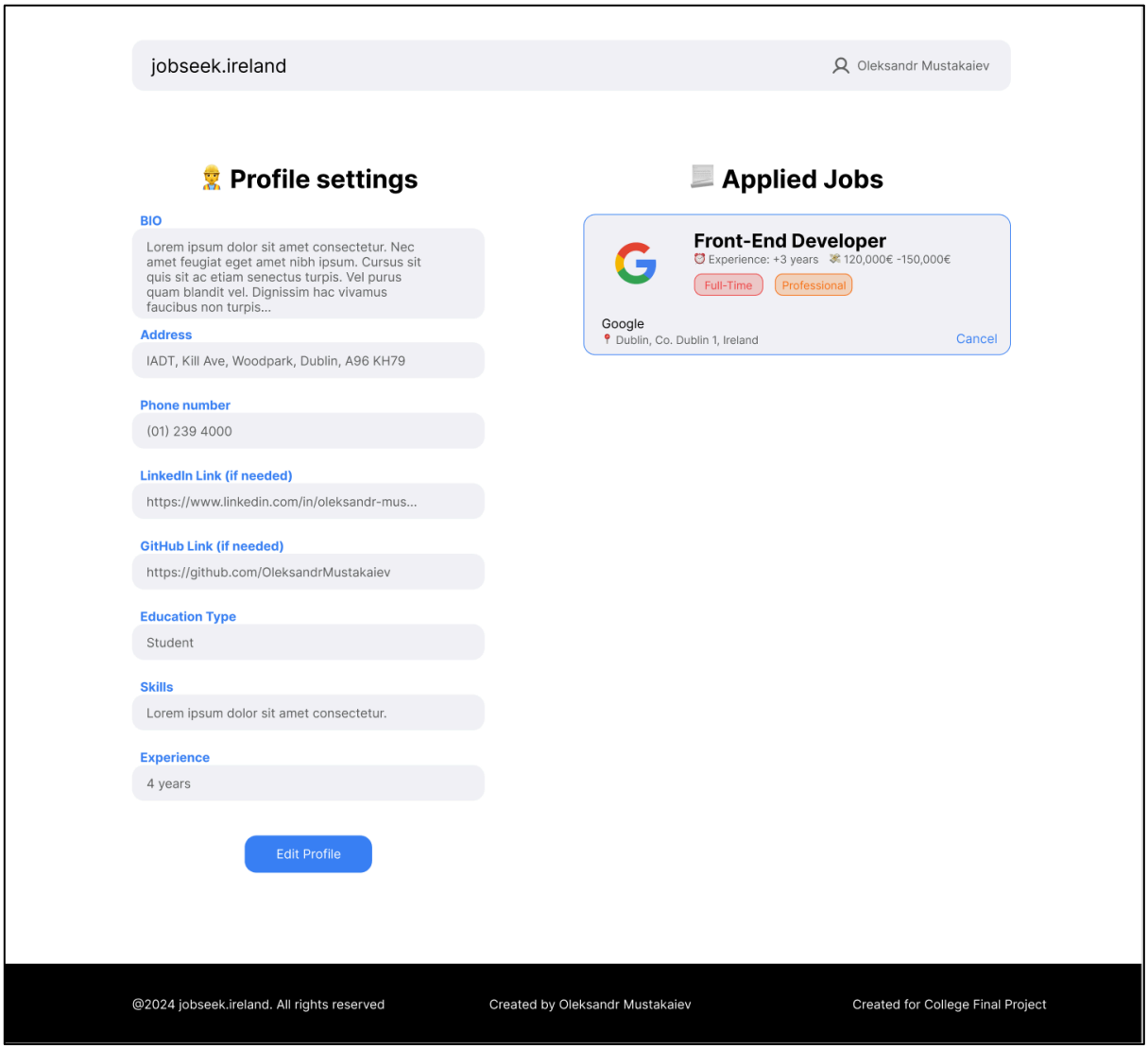

Figure 26 - User Profile Page

Figure 27, ***Employer's Dashboard***, where they can ***create new*** job offers, ***view*** their job offers and ***who applied*** to their job offer.

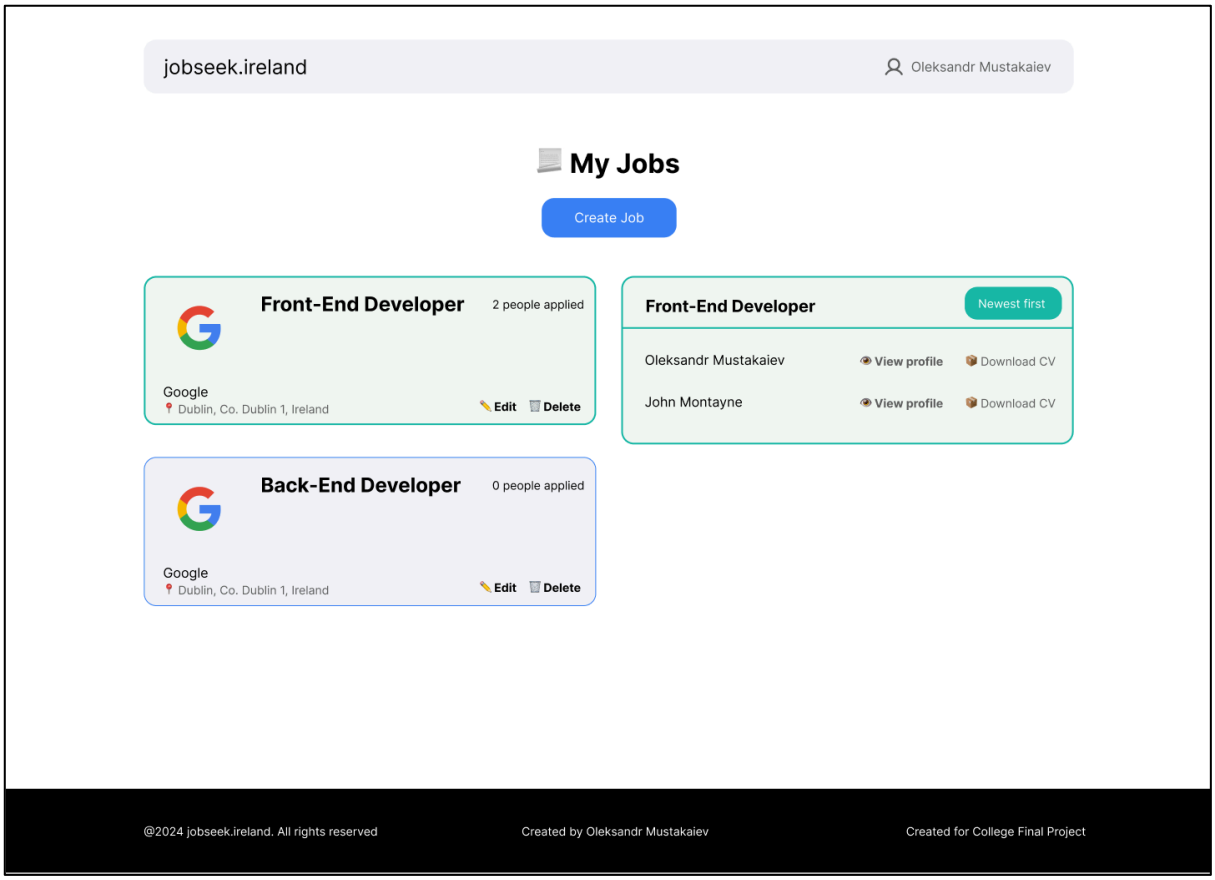

Figure 27 - Employers Page

Figure 28, when someone applies for a job offer, they can click on the user's ***profile view their profile*** and ***install their CV***.

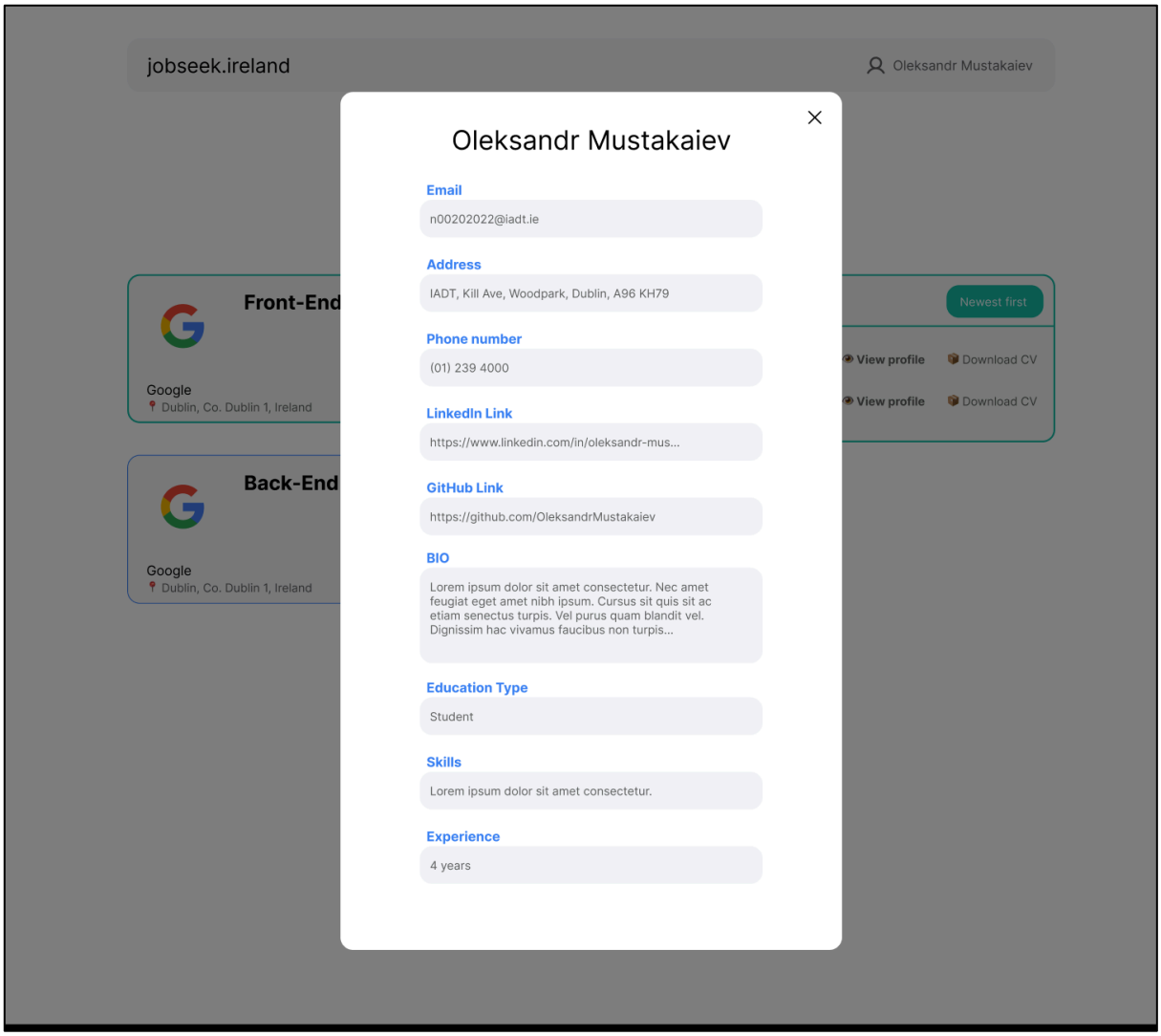

Figure 28 - Applied Jobseeker modal

Figure 29, Form for creating a ***job offer***.

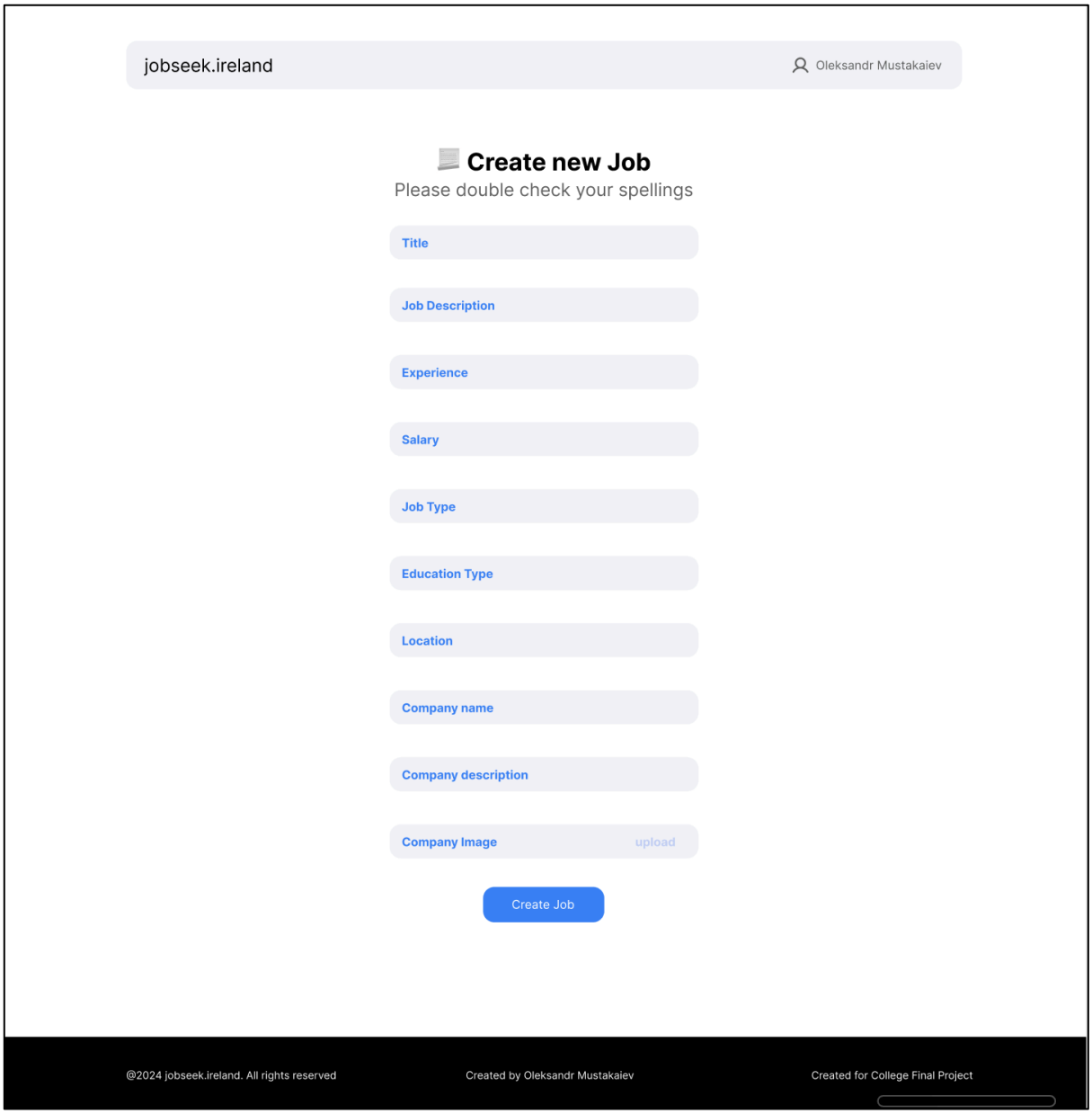

Figure 29 - Create/Edit Job Offer Page

4.3.2 User Flow Diagram

User: Home Page -> User Login modal -> User Register modal -> Upload CV Page -> Upload CV (2)/Profile Page -> User Page -> Home Page -> Apply Job -> Notification

Employer: Home Page -> Employer Login modal -> Employer Register modal -> Employers Page -> Create new Job Offer Page

4.4 Conclusion

In summary, the job seeker website's design includes both its functionality and style. For data handling, developer use tools such as Express for the back-end to keep a well-organized database. Users can find what they need on the website with a simple well-planned design.

5 Implementation

5.1 Introduction

This chapter describes the implementation of the application. The application has been developed using the following technologies:

- **Figma**

  Figma is a collaborative web application for interface design, with additional offline features enabled by desktop applications for macOS and Windows.

- **Express**

  Back-end web application framework for building RESTful APIs with Node.

- **React**

  MongoDB is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB utilizes JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and current versions are licensed under the Server Side Public License.

- **Tailwind CSS**

  Tailwind CSS is an open-source CSS framework. The main feature of this library is that, unlike other CSS frameworks like Bootstrap, it does not provide a series of predefined classes for elements such as buttons or tables.

The application for this project is to design a web application using Figma, to do Low Fidelity and High Fidelity. Express is used to create RESTful API and also connect ChatGPT API to the “User Profile” Table and host it using S3 Amazon for storing Images and Files (CV), and Vercel to host itself API.

5.2 Sprint 1 – Back-End (All API CRUD and Extracted Text with ChatGPT API)

5.2.1 Goal

Create the back end using Express and MongoDB, creating 9 tables in total.

5.2.2 Item 1

The Developer commenced with the back end to create basic tables. On the back end MongoDB Compass was used to create and log into an account while a blank database connect using Express was developed. The components server.js and db.js were created to start a server and a connection with the blank database created earlier. Later when the database was connected and running the Developer started to create routes, models and controllers for the Job table and to get basic *Create*, *Edit* and *Delete* functions working. After creating the Job Table, the Developer started to do other tables, such as *Experience*, *Education Type* and *Job Type*, so that later it would be easier to filter jobs for User Experience. For *one to many* joins the Developer had to get an ID from each additional table related to the Job Table to get their ID and display in *Job Respond* all details from Experience, Education and Job Type.

Thereafter the Developer started to create all other tables. In *User*, the Developer created the *User Profile* and *User Role* with *one to many* joins. This meant that when the user or Employer created an account it would display the relevant pages. For the User page, this includes the Upload CV and Profile pages. For the Employer page, an Employers Dashboard will be available.

To make it work, the Developer had to decode the *Role* and *Profile* tables to display them in response to the *User* table. The *Application* table is updated when the User applies for a job. Employers will see all applications on the employer's dashboard including the user's profile and CV.

5.2.3 Item 2

The next important step involved connecting ChatGPT API to the User Profile Table. To make it work, the Developer had to research how to extract text from the file and pass this text to the ChatGPT. For extracting files, a package called “pdf-parse” was used to get text only from the PDF file. The file index.html was created in Express to test extracting text in Front-End, as the React App wasn’t ready to be tested yet.

```
// Extract text from the PDF file
const pdfData = req.files.pdfFile.data;
const pdfText = await pdfParse(pdfData);
```

Figure 30 - Code of extracted text function

```
<!-- <input type="file" id="inpFile">
<button type="button" id="btnUpload">Upload</button>
<br>
<br>
<textarea style="width: 300px; height: 150px;" id="resultText" placeholder="Your PDF text will appear here"></textarea>
<script>
    const inpFile = document.getElementById("inpFile");
    const btnUpload = document.getElementById("btnUpload");
    const resultText = document.getElementById("resultText");

//      let test = "{\n  \"bio\": \"Oleksandr Mustakaiev is a skilled professional with experience in brewing coffee beverages and prov
// console.log(JSON.parse(test));

    btnUpload.addEventListener("click", () => {
        const formData = new FormData();
        formData.append("pdfFile", inpFile.files[0]);

        fetch("/extract-text", {
            method: "post",
            body: formData
        }).then(response => {
            return response.text();
        }).then(extractedText => {
            resultText.value = extractedText;
        }).catch(err => {
            console.error(err);
        });
```

Figure 31 - Code of extracted text on the front-end

In this code, the Developer had to write JS code to give ID to the input, button and button upload. Afterwards, to make it work, the Developer created a POST Request to get a response back from the extraction and to get a response with extracted text. The Developer began researching the ChatGPT API, and how to add and customise it. After installing packages from the official documentation

of ChatGPT API, work commenced on integrating ChatGPT API to the server.js, to call ChatpGPT API and pass extracted text from “pdf-parse”.

```
// Call ChatGPT API with the extracted text
const response = await axios.post(
    'https://api.openai.com/v1/chat/completions',
    {
        model: 'gpt-3.5-turbo',
        messages: [
          { role: 'user', content: pdfText.text },
          { role: 'user', content: "the above is a cv. give me the contents
      ],
    },
    {
        headers: {
            'Content-Type': 'application/json',
            'Authorization': 'Bearer sk-ByzyaVQXvYAa8kLsXt12T3BlbkFJVf4KKuBI
        },
    }
);
```

Figure 32 - Function to call chatGPT and give message with extracted text

At the request, the Developer passed the extracted text to the ChatGPT and told the AI that it is a CV, and required a return in JSON format with all attributes for the “User Profile” table. Afterwards, the Developer had to modify the code in index.html, to pass extracted text to ChatGPT and return the JSON text.

5.3 Sprint 2 – Front-End

5.3.1 Goal

After creating simple tables in the Back-End, the Developer started to work on the Front-End to create pages, components, a navbar and a footer.

5.3.2 Item 1

For the Front-End the Developer used React. After creating the app, the developer tested if it ran and worked. The Home Page, Navbar and Footer were then created. The developer used Tailwind CSS to style the react app and also used Google Fonts to use the “Inter” font. The developer used “Axios” to connect API and make calls. The first connection with API was Login as a User. After connecting the API developer had an error 404. To fix this bug involved checking the Back-End in the Login response. An IF statement was coded so that when the user logged in their name was displayed. A button to go to a profile page and log out, else show login with modal. Work then started on other pages such as Profile, UploadCV, Create Profile, Register Modal, Employers Login and Register Modals. Later, the Developer connected all Login and Register components with API and successfully tested them.

The other main work component in this sprint was to design the *Jobs* component in the Home Page, using a 6-6 cols grid. On the left-hand side, all jobs were shown and on the other side, more detail was provided when the user clicked on a job. After researching how to implement this design grid cols and overflow-scroll were employed to make it like 2 blocks with a scroll. Later, the Developer connected the API.

When a user clicks on the job it should show details associated with the job. After researching this requirement, the developer added the code “selected={i === selectedJobID}” to the array of job cards. In the Card component, the developer added the onClick function, so when the user clicks on a job card, it will take the ID and will display job details on the right side of the page and it will repeat after clicking on another job card. The same styling was used, so when the Job Card is selected, it changes colour to Teal (green) to tell the user that this card is selected. The Developer then connected the API to the UploadCV Page and used the same function to connect the chatGPT API call, same as in

Back-End -> index.html. In UploadCV, the Developer stores the response in the local storage and after successful storing, it navigates to the create profile.

```
const displayResponse = (responseData) => {
  // Store profile data in localStorage
  localStorage.setItem('profileData', JSON.stringify(responseData));

  // Navigate to create profile page
  navigate('/createprofile');
};
```

Figure 33 - Save response from chatGPT to local storage to pass it later on to create a profile page

At the Create profile, user be able to create a profile manually and also using their CV. Here is some code on how it gets a response from local storage.

```
useEffect(() => {
    // Check if profile data is passed from UploadCV
    const profileData = localStorage.getItem('profileData');
    if (profileData) {
        // Parse profile data and extract content
    const profileContent = JSON.parse(profileData).choices[0].message.content;

    // Log the profile content to ensure it's retrieved correctly
    let parsedData = JSON.parse(profileContent);
    console.log(parsedData);
    // console.log(profileContent);


    // Populate form fields with profile content
    setForm((prevData) => {
        return {
            ...prevData,
            address: parsedData.address,
            phone_number: parsedData.phone_number,
            linkedIn: parsedData.linkedIn,
            gitHub: parsedData.gitHub,
            description: parsedData.description,
            education: parsedData.education,
            skills: parsedData.skills,
            experience: parsedData.experience
        };
    });
    }
}, []);
```

Figure 34 - Getting a response from local storage in creating a profile page

In useEffect, it checks if profile data is passed from local storage parses data and extracts content, and then populates form fields with profile content. Also in inputs developer added value={form.address}, etc. After that users can check what response they got and add or edit some spellings. After the Developer begins to create an Employer Dashboard Page for employers it will store employer's job offers and who applied for them, and also be able to create, edit or delete their job offers. It uses similar job cards from the job cards on the Home Page, but on the right-hand side, it will display applicants with their profiles and CVs.

For linking pages, the developer used “Link” from “react-router-dom”. It works in cooperation with React Router to enable client-side navigation in single-page apps, avoid full-page reloads, and offer extra features for a better and smoother user experience.

5.3.3 Item 2

The other main features to implement are Searching, Filtering and Sorting By. This feature was added to the Home Page for jobs. Firstly, the Developer state variables for filtered jobs to an array and term to empty string.

```
const [filteredJobs, setFilteredJobs] = useState([]);
const [term, setTerm] = useState("");
```

Figure 35 – Adding State Variables

In useEffect, where the developer connected API, the following code was added “setFilteredJobs(response.data)” in “.then”, so later when the user searches, jobs will be changed.

```
useEffect(() => {
    axios
    .get(`job`)
    .then((response) => {
        // console.log(response.data.data);
        setJobs(response.data);
        setFilteredJobs(response.data);

    })
    .catch((error) => {
        console.log(error);
    });
}, []);
```

Figure 36 - Adding "setFilteredJobs" to API call

Instead of “setJobs”, the developer changed to “setFilteredJobs”.

```
// array of jobs card
const jobsList = filteredJobs.map((jobs, i) => {
    return <JobCard setId={() => setSelectedJobId(i)} key={i} jobs={jobs} selected={i === selectedJobId}/>;
});
```

Figure 37 - Changing "setJobs" to "setFilteredJobs" an array of job cards

The main function was to create a handle search by job title’. In the “if” statement, when the code:

if in search input is blank, show all jobs, else run filtering by job title. Was implemented while also allowing lowercase letters.

```
// search job by the name
const handleSearch = (event) => {
    setTerm(event.target.value);
    // if search term was changed to blank, show all jobs
    if(event.target.value === ""){
        setFilteredJobs(jobs);
    }
    // else if search term is less than or = 1, do nothing
    else if(event.target.value <= 1){
        return;
    }
    else{
        let filter = jobs.filter((job) => {
            return job.title.toLowerCase().includes(term.toLowerCase());
        });
        setFilteredJobs(filter);
    }
};
```

Figure 38 - Searching by job title

The developer did 3 types of filtering Experience, Job Type and Education. Here is the code using useEffect, it checks if it’s not pressed it will show all jobs, only

will be changed after filtering it by experience, job type or/and education by using the job table, for example, job.experience.name.

```
// filtering by experience, job type and education
useEffect(() => {
    let filtered = jobs.filter(job => {
        // Filter by experience
        if (selectedExperience !== "all" && job.experience.name !== selectedExperience) {
            return false;
        }
        // Filter by job type
        if (selectedJobType !== "all" && job.job_type.name !== selectedJobType) {
            return false;
        }
        // Filter by education
        if (selectedEducation !== "all" && job.education_type.name !== selectedEducation) {
            return false;
        }
        return true;
    });
    setFilteredJobs(filtered);
}, [jobs, selectedExperience, selectedJobType, selectedEducation]);
```

Figure 39 - Demo of code how filtering works

In the dropdown, it will check the onClick function which one was clicked.

```
<Menu.Item>
    <p onClick={() => setSelectedExperience("all")} classN
</Menu.Item>
<Menu.Item>
    <p onClick={() => setSelectedExperience("0-1 years")}
</Menu.Item>
<Menu.Item>
    <p onClick={() => setSelectedExperience("1-2 years")}
</Menu.Item>
<Menu.Item>
    <p onClick={() => setSelectedExperience("2-3 years")}
</Menu.Item>
<Menu.Item>
    <p onClick={() => setSelectedExperience("3-4 years")}
```

Figure 40 - onClick function to select experience. Same for education and job type

Sort by the system works similarly to filtering, it checks which event was clicked and just sorts jobs by the event click. It sorts by job name a-z or z-a or latest or oldest.

```
// sort jobs by name and date
const handleSort = (event) => {
    let sorted = [...filteredJobs];

    if (event === "name_a-z") {
        sorted.sort((a, b) => a.title.localeCompare(b.title));
    }
    if (event === "name_z-a") {
        sorted.sort((a, b) => b.title.localeCompare(a.title));
    }
    if (event === "latest") {
        sorted.sort((a, b) => new Date(b.createdAt) - new Date(a.createdAt));
    }
    if (event === "oldest") {
        sorted.sort((a, b) => new Date(a.createdAt) - new Date(b.createdAt));
    }

    setSelectedSortOption(event); // Update selected sort option
    setFilteredJobs(sorted);
};
```

Figure 41 - handle sorting function

5.4 Sprint 3 (Hosting Back-End)

5.4.1 Goal

To host Back-End.

5.4.2 Item 1

The developer used S3 Amazon to host Images for “job_company” and Files for the “User Profile” table to store the applicant's CV. For Images and Files, the Developer used “multer” and “@aws-sdk/client-s3” to give settings and

requirements for storing files and images, and also add to the .env file. The developer also needed to create an account in AWS Amazon, to create a bucket in S3 and use IMP to create a user for using that bucket from S3. After creating and giving settings for the bucket such as permission and properties, the Developer started to modify the Job Model, adding “image_path” and in the controller adding the *delete image* function. In the create job, a developer used the if statement to check where to add images, on the local host or s3 bucket, depending on the .env setting. For the update, it is something similar, but it deletes the old image and adds a new one. For delete, this just deletes the image using the “deleteImage” function.

After successful testing, the developer started to do the same with files for the “User Profile” table, but in requirements is pdf only, as in image it is any type of image, such as PNG, etc. After testing the upload file in the “User Profile” table, the Developer installed a package called “vercel” to host API. To do this, the developer needs to create a “vercel.json” file to store the path for the API link “/(.*)” and then run a command using vercel to host API.

5.5 Sprint 4 (Hosting)

The developer used Vercel to host the Back-end and Front-end. To do this for the Back-end, the developer installed the package *vercel and* also added a vercel.json file to give a default path of API and run the *npm vercel* command to host online. To do this for the Front-end, the developer decided to host it using the GitHub repository, so after some changes, it will automatically update a hosted version.

5.6 Conclusion

The developer achieved most of the work that he planned. There were some difficulties during implementation, for example developer had to research chatGPT documentation, to connect it to the back end and make it work. In the

end, the developer created the Back-end with chatGPT API and created a react app, which allows users to register, log in, create a profile, apply for jobs, register and log in as an employer, create job offers, also edit and delete them, and view applicants.

6 Testing

6.1 Introduction

This chapter describes the testing that has been undertaken for the application. This chapter is presented in two sections:

1. Functional Testing
2. User Testing

6.2 Functional Testing

This section describes the functional tests which were carried out on the app. The developer will test the chatGPT request and respond.

At the start, the developer passed extracted text from the user's CV to the chatGPT and asked:

{ role: 'user', content: "the above is a cv. give me the contents of the CV in the following format. {name: 'value', description: 'write a short description about this person here', address: 'value', phone_number: 'value', linkedIn_link: 'value', github_link: 'value', education_type: ['values'], skills: ['values'], experience: [{job_title: 'value', company: 'value', start_date: 'value', end_date: 'value', description: 'value'}]}"}

The result was bad, as for the address it was email instead of the home address of the person, for education, skills and experience chatGPT was giving only 1 of each. Also if it can't find something, it gives a response as “Not Provided in CV”, sometimes for LinkedIn and GitHub URLs, was always gives different sizing characters, so it would cause a problem at the Front-end passing it.

After the developer gave all the settings to fix those errors and bugs, the final request was:

{ role: 'user', content: "the above is a cv. give me the contents of the CV in the following format. {name: value, description: write a short description about this person here, address: value, phone_number: value, linkedIn: value, gitHub: value, education: values, skills: values, experience: values}, use double quotes in the keys and values. for address use full home address from the cv, not email. List all experience, education and skills, for example: education[array of all listed from cv].string. If there is no value for any key, leave it empty string. Use same sizing letters, for example gitHub is gitHub and linkedIn is linkedIn, nothing else."}

The result was successful and corrected. If users haven't included something in their CV, it is blank now, users can see what they need to write down. Fixed sizing characters and for experience, skills and education it gives an array from the CV.

6.2.1 Discussion of Functional Testing Results

There were some errors in the chatGPT response, to make it correct the developer has fixed all the problems. Sometimes chatGPT gives a different response from the same CV, always changing the sizing of characters, giving only 1 or a few of the education, skills or/and experience. There are a lot of ways that the developer can make a good request to get a good response. In summary, it is hard to get the expected response from the chatGPT.

6.3 User Testing

For the user testing, the developer used a website called *Maze* to do 2 questions about the design of the app, 7 people attended the user testing. For the introduction, the developer asked if users used before job seek websites, to see if they had experience before using them. 71% YES and 29% NO.

The first task was to create an account as a user and create a profile without a CV. The feedback was positive, here are some comments from users.

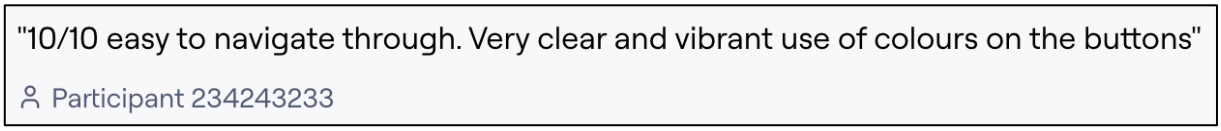

Figure 42 - 1 of the feedback from task 1

Figure 43 - Other feedback from the task 1

The second task was to log in as an employer, create a job and after creating, view the user's profile from the first job to the first person. The feedback was positive, here are some comments from the users.

Figure 44 - 2 feedbacks from task 2

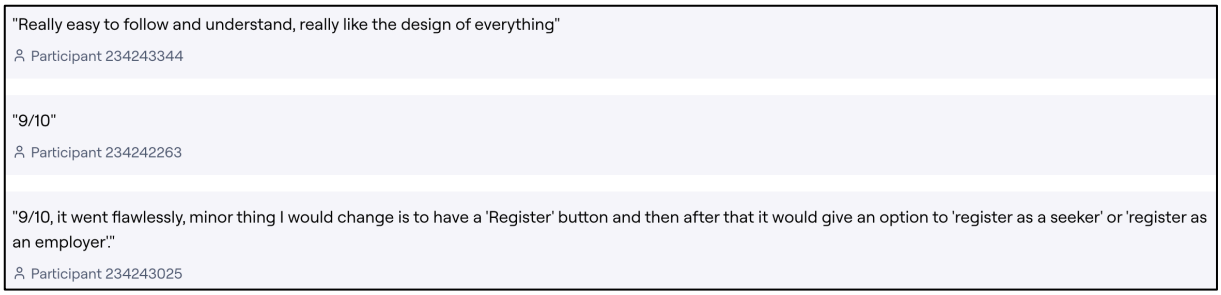

Figure 45 - some other feedback from the users for task 2

The last question was to give feedback about the design including colours, spacing and overall design of the app. Mostly it was positive, some users offered some changes. Here are some comments from the users.

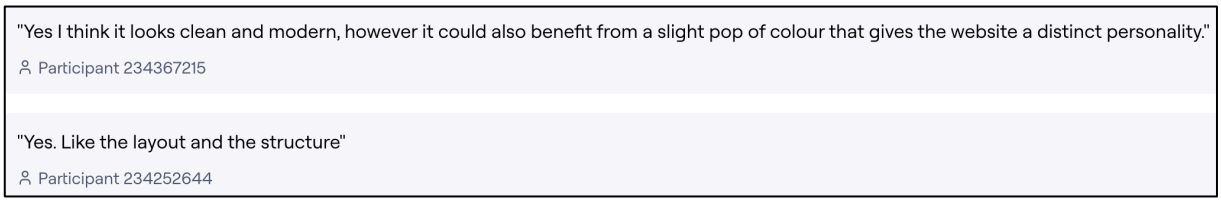

Figure 46 - Feedback from question about the design

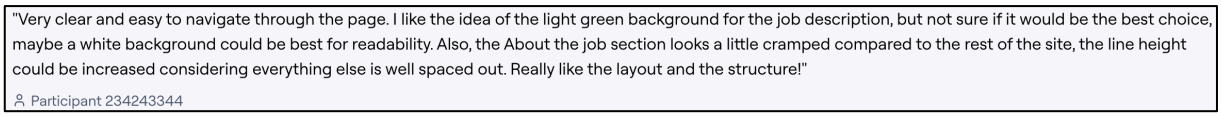

Figure 47 - Feedback from the users from the last question about the design

6.4 Conclusion

In summary, the developer got positive feedback about user testing and design, there is no need for any critical or major changes.

7 Conclusion

In conclusion, the developer has successfully created a job seeker website, where users can create an account, create a profile using their CV or manually, search for jobs including sorting, filtering system and applying for them, while employers can manage their job offers and view applicants with their CV and profile.

There were no problems with using technologies, the developer got more experience with working *React*, *Express*, *Tailwind CSS*, *ChatGPT API* and testing *API* in *Insomnia*, hosting *API* and *app* in *Vercel*.

After researching which framework for the Front-end is more popular and manageable, the developer created a successful website using *React,* learned how AI works and successfully integrated it into the Back-end.

The developer had to research a few similar websites, to get all the advantages and disadvantages of the designs, after the developer had successfully designed a website, after testing the feedback was very positive.

During implementation, the developer had some difficulties with connecting ChatGPT API and passing extracted text, after research and testing the developer implemented successfully the function. There were also some small errors and bugs during implementation, but the developer had no difficulties to fix them.

There were no difficulties doing user testing, as the developer had done user testing before, for functional testing, the developer had to test many variants to give the best request and get a good response to pass to the "*Create a Profile Page*".

How the project could be further developed

There are a lot of ways to make this project better and to make it public, the developer had some ideas, and here are some of them:

- Upgrade security
- Make a payment system for employers to advertise job offers, that can show jobs at the top of the list for a certain time
- Create Admin panel
- Get users to use this product
- Register or log in using Google API, LinkedIn API and so on
- Upgrade the filtering system, to find more related jobs that the user is looking for

8 References

The Department of Technology and Psychology in IADT uses APA referencing style.

Use alphabetical order for your references.

This site gives details about how to cite websites using APA:

https://www.wikihow.com/Cite-a-Website-in-APA

The following is a useful site for creating citations for APA for websites.

http://www.citationmachine.net/apa/cite-a-website

You can also use the Referencing tab within Microsoft Word to enter reference information manually. Word then creates an APA style reference.

8.1 Research:

Bhalla, A., Garg, S., & Singh, P. (2020). *PRESENT DAY WEB-DEVELOPMENT USING REACTJS.* International Research Journal of Engineering and Technology (IRJET).

Dang, & Dat. (2020). *Developing a website with user experience.* LAB UNIVERSITY OF APPLIED SCIENCES LTD.

Gabrio, M., Hovan, G., & Shaji, G. (2023). A Review of ChatGPT AI's Impact on Several Business Sectors. *Puiij*, 1-15.

Gackenheimer, C. (2015). *Introduction to React.* New York: Apress.

Goeva, A. (2019). *The role of website analytics in identifying user experience (UX) improvements.* Metropolia University of Applied Sciences.

Hassenzahl, M., & Tractinsky, N. (2006). User experience - a research agenda. *Taylor & Francis Online*, 1-8.

Heikinmäki, T. (2020). *PRINCIPLES OF UX DESIGN.* LAB UNIVERSITY OF APPLIED SCIENCES LTD.

Hellweger, S., & Wang, X. (2015). *What is User Experience Really: towards a UX Conceptual Framework.* Free University of Bolzano Bolzano, Italy: Free University of Bolzano.

Hidvég, T. M.-R. (2022). *Are the frameworks good enough?* Sweden: Tomas Marx-Raacz von Hidvég.

K, F. A. (2021). ARTIFICIAL INTELLIGENCE. *ARTIFICIAL INTELLIGENCE*, 76.

Kumpulainen, T. (2021). *Web application development with Vue.js.* tbc: JAMK University of Applied Sciences.

Kumpulainen, T. (2021). *Web application development with Vue.js.* JAMK University of Applied Science.

Macrae, C. (2018). *Vue.js Up & Running.* Sebastopol: O'Reilly.

McCarthy, J. (2004). WHAT IS ARTIFICIAL INTELLIGENCE? *Formal Stanford*, 1-14.

Mistry, A., & Rajan, A. P. (2019). Evaluation of web applications based on UX parameters. *International Journal of Electrical and Computer Engineering (IJECE)*, 1-7.

Nelson, B. (2018). *Getting to know Vue.js.* USA: Apress.

Pikkanen, M. (2021). *React and Vue performance comparison.* Metropolia University of Applied Sciences: Markus Pikkanen.

Rawat, P., & Mahajan, A. (2020). ReactJS: A Modern Web Development Framework. *International Journal of Innovative Science and Research Technology*, 1-5.