lancing

Lancing Beta Version

Students
James Bell
Negin Jahanbakhsh
Nada Saadi
Fidel Paredes Sánchez

Instructor
Dr. Fairouz Medjahed

Supervisor
Dr. Jorge Martínez Ladrón de Guevara

Saint Louis University - Madrid

Spring 2022
CSCI 4962

Table of Contents

I. Introduction

A. Background

Now more than ever, remote work is on the rise thanks to our advancement in technology. As shown in Figure 1, with the coronavirus pandemic, remote work skyrocketed to just about 70%. Although we are still technically in the pandemic, it has been two years since its peak and things are relatively returning to a new normal. However, this trend of remote work still exists and there are still just under 50% of people working from home. There are many benefits attributed to working from home for the employer as well as the employee. Many companies are going fully remote to save costs on business offices as they have seen that people are just as, if not more productive working from home than at the office. From an employee's perspective, it's no longer required to dress fully professional or be obliged to sit at the desk all day. In addition, breaks from work can be healthier, such as going for a run or doing yoga in the living room. As of April 2022, 16% of companies claim to be fully remote worldwide, while 66% of them admit to have installed a work from home policy for at least two days per week. We strongly believe that remote work is not a temporary trend, rather a solution to the modern workplace that is here to stay.

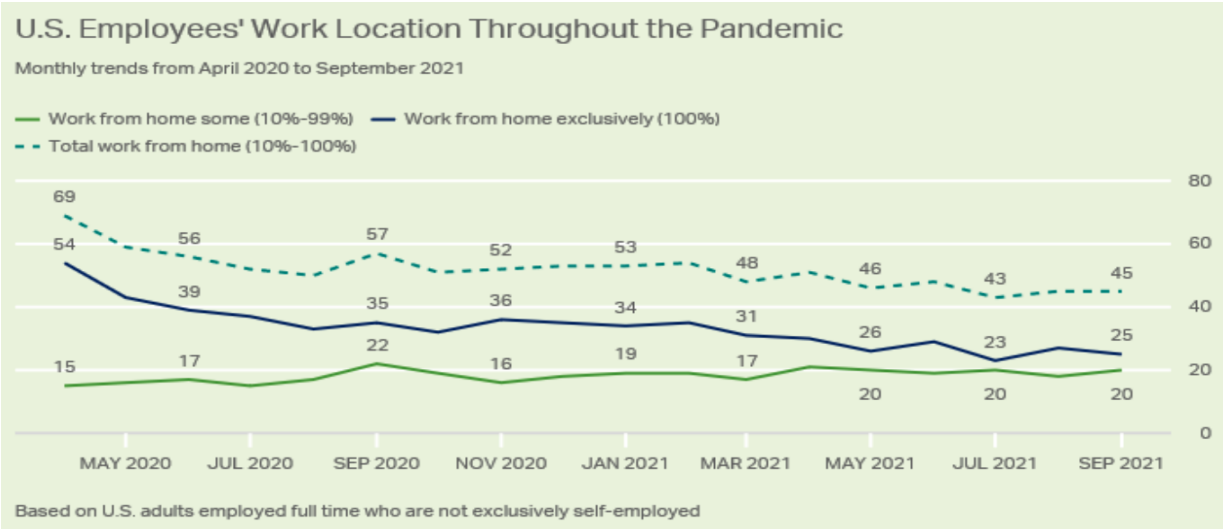


Figure 1: Work from home growth in the US

B. Problem Statement

The whole idea that remote work is a permanent trend was the main motivation for this project. Lancing will be providing opportunities to people from other countries who do not have the means to travel to the location of the job. We are aware of and admire other similar freelancer marketplace sites such as UpWork and Fiverr, but companies do not like the fact that anyone can apply to their jobs. That is why we created Lancing, freelancers are matched with jobs depending on their skillset and they are only given the option to apply if they are compatible. This way, companies are getting less applications for the same jobs but the applicants are guaranteed to be potential employees given their skills. Applicants are hired from different cities and countries around the globe based on their talent and qualifications.

C. Purpose

The aim of this document is to demonstrate the beta version of the Lancing application, including the application interface, system architecture, and demo website. The initial design we had in the earlier version had a different layout and framework for the login and registration pages. The website shows the main added value of the application, a brief description about what Lancing is about as well as simulated user reviews. The site also introduces the cofounders, and a contact form to receive any opinion, complaint or suggestion.

D. Scope

The scope of this project is a full scale application from scratch. Beginning with the initial idea in our minds through design to finally implementation. Lancing will adopt many existing features from third party applications to fit our needs. The idea for Lancing is to provide companies with top talent that are guaranteed to be compatible with the skills they are searching for.

E. Glossary

Node.js	Node.js is primarily used for non-blocking, event-driven servers, due to its single-threaded nature. It's used to build scalable network applications and for traditional web sites and back-end API services.
React Native	React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android.
Android Studio	Android Studio is the official integrated development environment (IDE) for android app development. It is used to view the changes to the application interface with real time changes on an emulator.
API	Application Programming Interface or API is a set of rules that define how applications or devices can connect to and communicate with each other.

GitHub	**GitHub, Inc.** is our choice for version control using Git. It offers the distributed version control and Source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking and feature requests.
MySQL	MySQL is an open source SQL relational database management system that's developed and supported by Oracle.
Visual Studio	VS is a powerful source code editor. It comes with built-in support for JavaScript and Node.js and has a rich ecosystem of extensions for other languages. We used VSCode in our app for the development, along with the android studio for the virtual emulator.
Android Studio	Android Studio simulates Android on your computer so that you can test your app on a variety of devices. It allows a flexible building system based on Gradle, a fast and feature packed emulator and a unified environment where you can develop for all Android devices.

II. Models

A. Domain Class Diagram

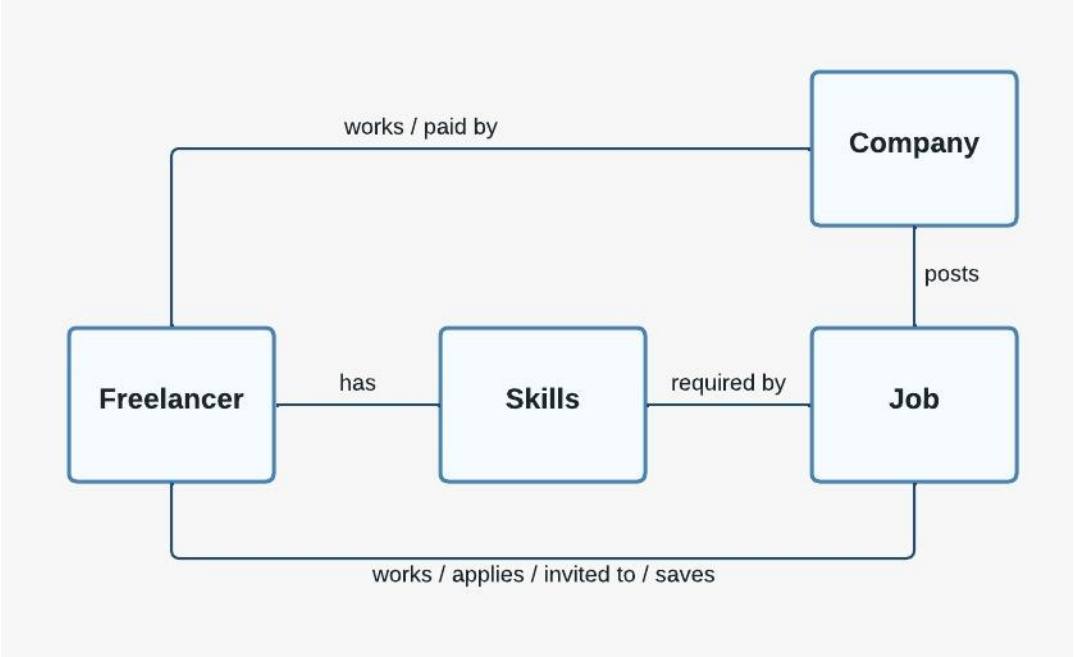


Figure 2: Domain Class Diagram

A freelancer will have some specific skills that a project will require. Therefore, depending on the skills of the freelancer, the freelancer will meet the skills requirements for one project or another. Skills will be uniquely identified by their category, subcategory, and skill.

Projects will be identified by a specific project ID, and will also be related to the skill they require. Freelancers are able to apply, be invited to, work, and save jobs to their favorites. Companies will be identified by their company id and each company will have a specific contact that will have the responsibility of the login information. Companies are the entities responsible for posting jobs and are able to pay and save freelancers to their favorites.

B. Interaction Overview Diagram

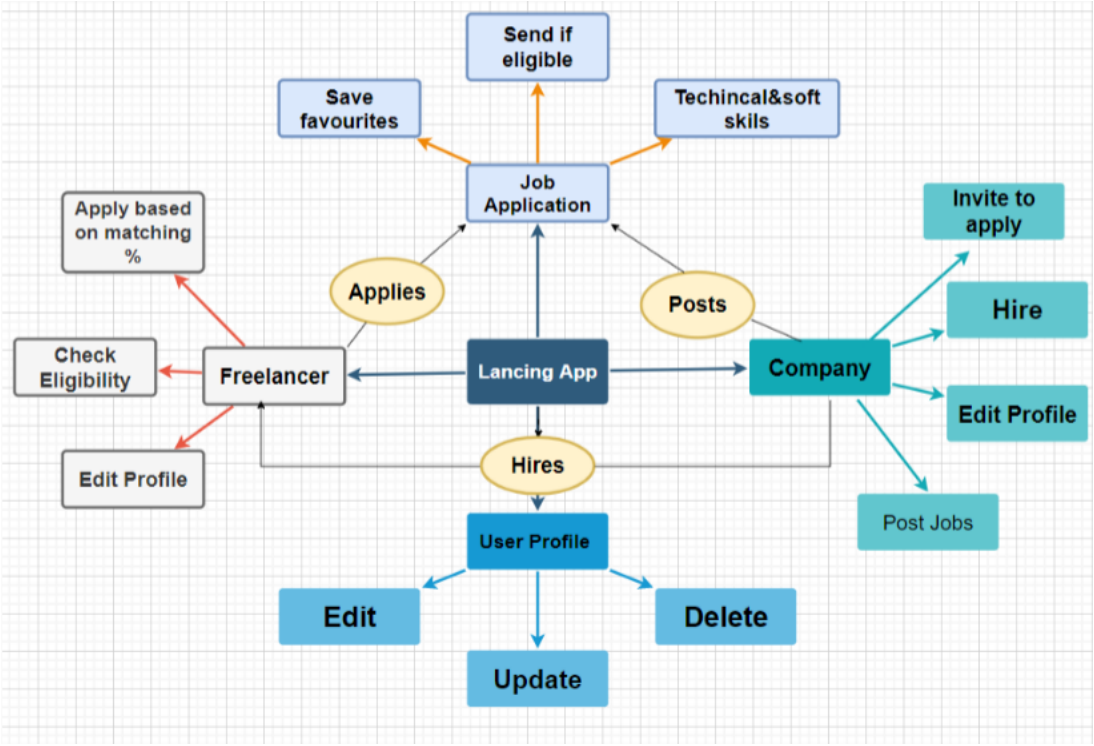


Figure 3: Interaction Overview Diagram

The interaction overview diagram shows the main four components of our application: User profile, company, freelancer and job. It is a more detailed description of the main use cases of each user and the relationships between them.

C. Database Physical Model

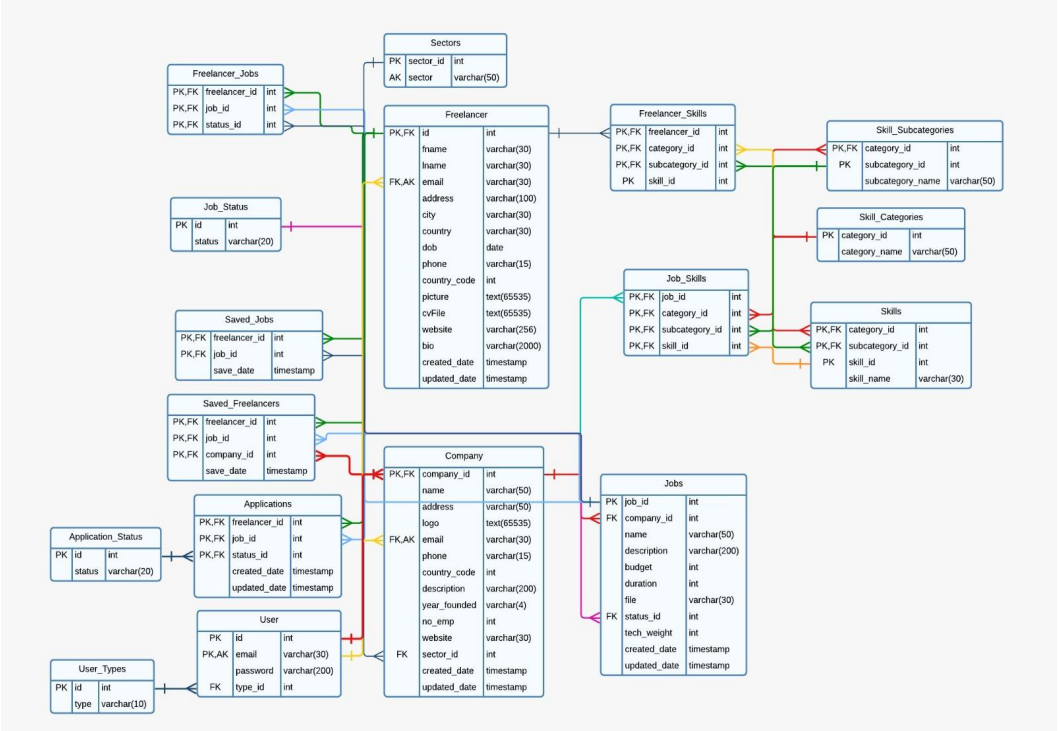


Figure 4: ER diagram

In figure 4, we can see the most recent version of the physical model of the database. The entities designed in the logical model become tables as do some weak entities that came from some of the relations. The main entities for the application are Freelancer, Jobs, Company, and Skills which are in the center of all the tables.

'Saved_Freelancers' acts as the "favorites" list for the company. It holds the freelancers that a company has saved on their profile.

'Skills' as mentioned earlier are identified by their category, subcategory, and skill which also have tables of their own. This makes the skills have a hierarchical structure rather than the previously designed linear model.

'Freelancer_Skills' records all the skills a freelancer chooses to add to their profile.

‘Job_Skills’ records the skills required for a specific job.

Moving to the left, ‘Saved_Jobs’ acts as another “favorites” list, this time for the freelancer. It holds the jobs a freelancer saves to their account.

‘Applications’ records the applications from freelancers to jobs and its status, which can be retrieved from ‘Application_Status’.

‘Freelancer_Jobs’ contains all the jobs both in progress and finished that a freelancer works.

The changes we made in the database from the alpha version include:

We removed the company contact table and directly added the email address and phone to the company table because since there was a 1:1 relationship between the contact and the company we thought it was best to join them for increased simplicity among the entities.

We added a Users table to keep track of all the users, both freelancers and companies in a single table. These users are identified by their email and type id. The type id is related to the type of user they are registered as (freelancer or company). The purpose of this change was for a more straightforward implementation of the registration and to show the respective pages depending on the

D. Use Case Diagram

Our use-case consists of two users: the freelancer and the business representative. A freelancer user can apply for a project, save a project, get notified, and contact a business. A business owner can post a project, hire and invite a freelancer, save a profile, send payment, and contact a freelancer.

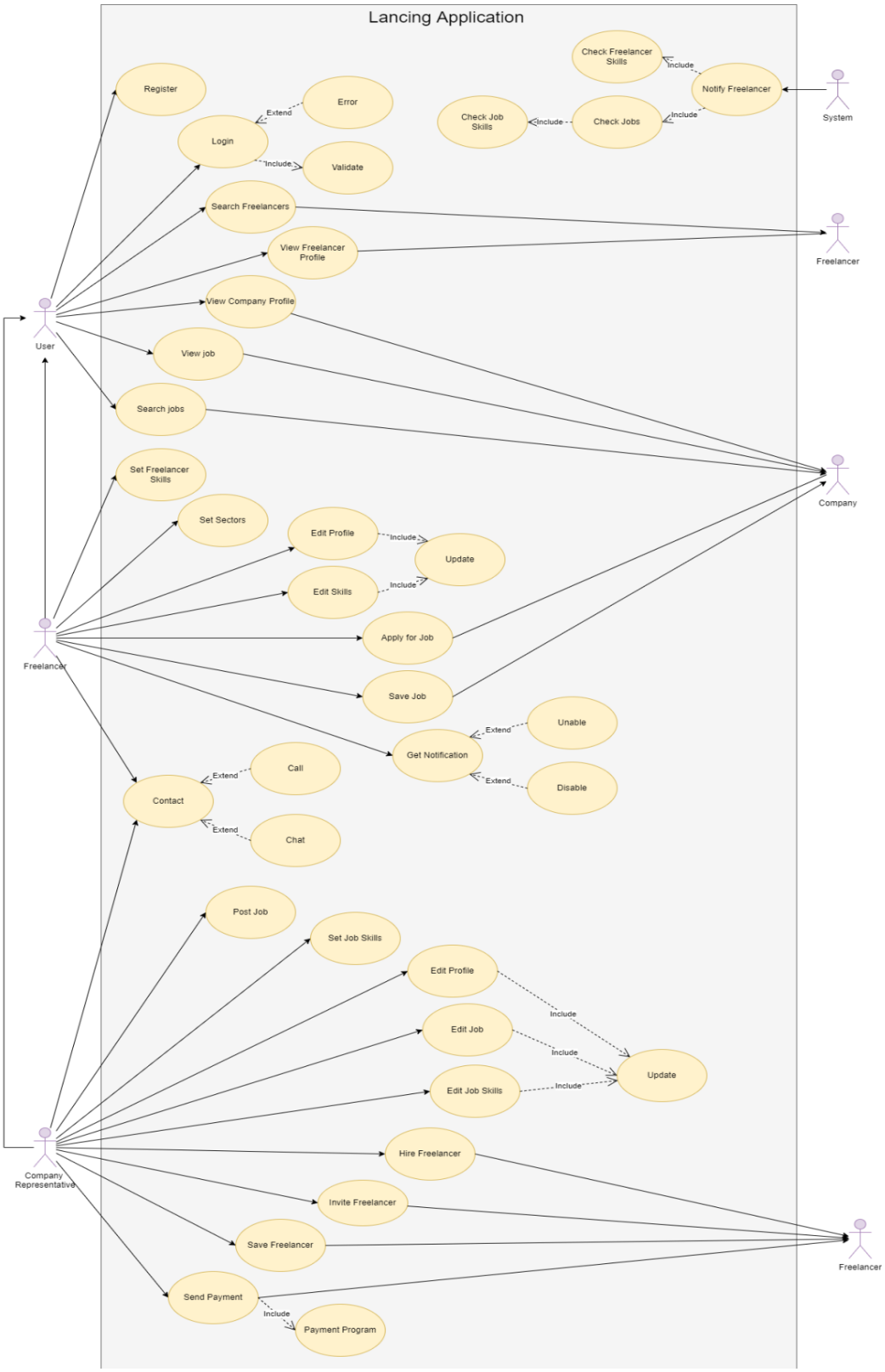


Figure 5: Use case diagram

This use case shows the functionalities of our Application. In this use case diagram we have three different primary actors: ‘User’, ‘Freelancer’, and ‘Company representative’. These functionalities include ‘Search for a freelancer’, ‘View freelancer profile’, ‘Search for job’, ‘View job’, and ‘View company profile’. A ‘User’ can login or register as a ‘Freelancer’ or as a ‘Company’. As the diagram shows, all the functionalities of the actor ‘user’ will be inherited by the actors ‘Freelancer’ and ‘Company’ .The ‘Freelancer’ will be able to perform the inherited as well as the following functionalities: ‘Set freelancers skills’, ‘Set sector’, ‘Edit profile’, ‘Edit skills’, ‘Apply for job’, ‘Save job’, ‘Get notification’, and ‘Contact’. The included use case ‘Update’ will update the profile and the freelancer skills whenever there is an edit in these two areas. Whenever the freelancer applies or saves a job, the use case will interact with the company as a secondary actor. The use case ‘Get notification’ has two extended use cases, so that the user will be able to ‘Enable’ or ‘Disable’ the notifications. The use case contact has two extensions, to ‘call’ and to ‘chat’. The ‘Company representative’ will be able to perform the inherited functionalities of ‘User’ as well as the following functionalities: ‘Post job’, ‘Set job skills’, ‘Edit profile’, ‘Edit job’, ‘Edit job skills’, ‘Hire Freelancer’, ‘Invite Freelancer’, ‘Save Freelancer’, and ‘Send payment’. The included use case ‘Update’ will update the job and the company respresentative’s profile whenever there is an edit in the profile, job, or job skills. The secondary class ‘freelancer’ will react whenever the following use cases are performed: ‘Hire freelancer’, ‘Invite freelancer’, ‘Save freelancer’, and ‘Send payment’. The included use case ‘Payment program’ will be performed when ‘Send payment’ is performed.

This use case represents the final design of the application. As we will not have the means to implement all the features we initially brainstormed, some of the added features of the application will be left out; for example, the payment feature was an idea we initially thought to implement in the app but will not in the current scope.

E. Skills Hierarchy

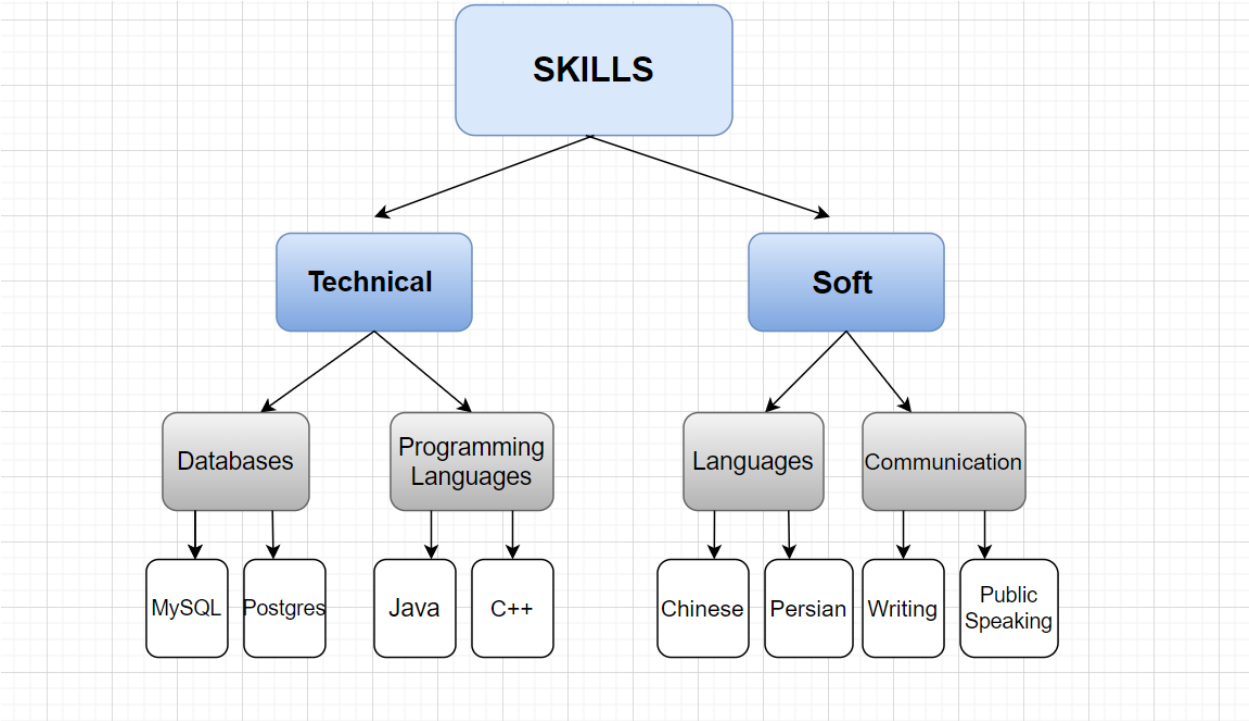


Figure 6: Skills Hierarchy

We designed the skills to have this hierarchical structure due to its ease of categorization since jobs will require skills of different categories and subcategories. This way we can also calculate the different compatibility levels between a freelancer and a job such as the compatibility for the technical skills and the compatibility for the soft skills.

III. Software Architecture

A. Backend

The backend of the application is implemented in node.js. We chose node.js because of its wide usability in the labor market and its various functionalities and offered libraries. The structure of the backend is as follows:

- Database scripts
- Controllers
- Routes
- Services
- Validation

This type of structure allows an easy and simple way to organize the code. In the database scripts you can find the DDL for the database as well as a sql script to load sample data into the tables. The controllers directory includes the functions that call the queries from the database. The routes include the http methods and the function to call with the respective request method. The services directory has the more logic heavy code, including the login service which handles the log in with the validation called from the Validation directory, and the registration service which after checking to make sure everything is validated like the password length, unique email, password confirmation, then calls the query to insert the data into the User table.

Matching Algorithm

The matching algorithm we designed and implemented is a simplified algorithm which works in the following way. To give some context, when a company posts a job, they are given the option to give a custom weight to the category of skills. For example, a technical job like an iOS Developer will prioritize their technical skills, so the job post can give an 80% weight to those types of skills and a 20% weight to soft skills such as communication and languages whereas for a job like graphic designer, the job post could give a 20% weight to technical skills and an 80% weight to soft skills such as creativity and artistic ability. Going back to the algorithm, first the

skills required for a given job are retrieved as well as the freelancer skills that match up with the skills required for the job. The number of matched skills is divided by the total number of skills required for the job, and the result is multiplied by the category's weight. This way, the matched freelancers can get a good idea that they are fit for the job.

Tools

As mentioned in the glossary, we used many libraries in the backend to further the functionality and security of the app. To further elaborate on a few, we used Passport for our authentication and authorization service especially in the implementation of the login and registration. The useful aspect of Passport is that it also includes functionality for user sessions, so if a user logs in and makes changes and logs out, the next time they log in those changes will still be there. Babel was used to make the program backwards compatible with previous javascript versions, something that is useful at compile time. Multer was used to parse form data, in our case for parsing the data sent during login and registration. Bcrypt was used to hash user passwords in the registration service to add security to the application and keep user's data safe. Swagger was used for documentation due to its user-friendly interface that is provided when documenting the program. To access the documentation, navigate to the local host's server /docs. In order to access functions in the backend such as handling login and registration, we used Axios to send requests to the REST endpoint and perform CRUD operations.

B. Frontend

The frontend of this application was implemented in Flutter at first, but due to some difficulties in integrating the backend with frontend, we have decided to implement the application interface in React Native. This transition made it a lot easier for us to communicate with JavaScript files and develop our application since all the files in React Native are in javascript. The entire codebase was written in Visual Studio Code editor and we used many libraries and dependencies in the frontend such as handler, fontAwesome5, CreateStackNavigator, React components, etc. To run our codes, we used the Android Studio emulator. This led us to view and improve our application interface on the Android mobile phones. Each screen of the application is written in separate files and exported to the App.js file. We also used Stack in order to navigate between

screens. We were able to pass the data from parents to child functions by using props. The React is utilized for both layout and designing of this application.

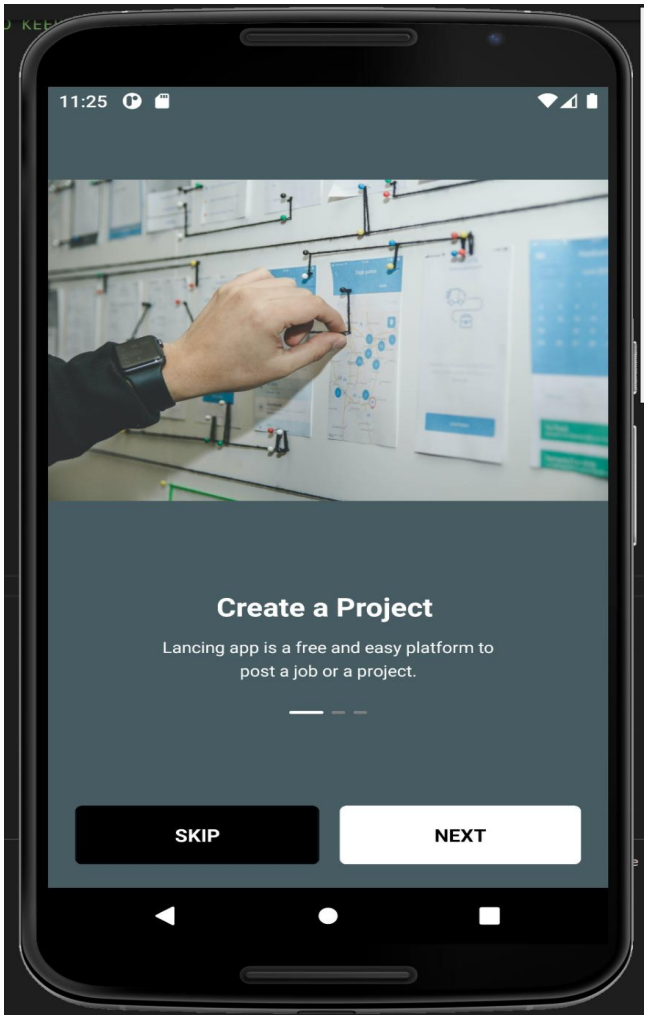
IV. Interface Design

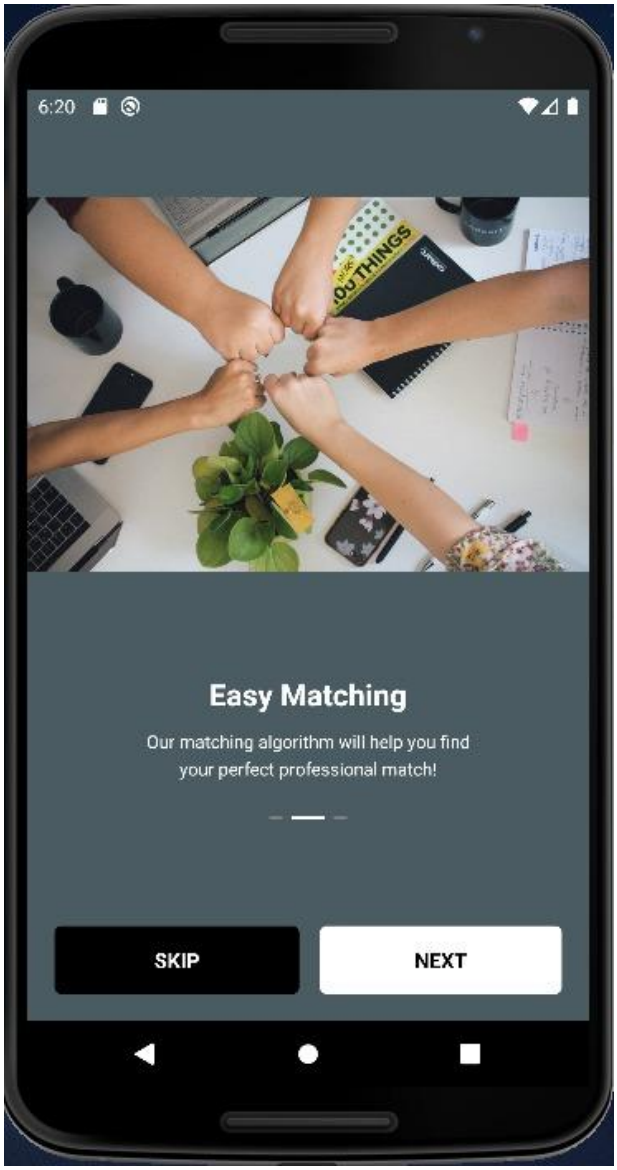
A. Application Interface

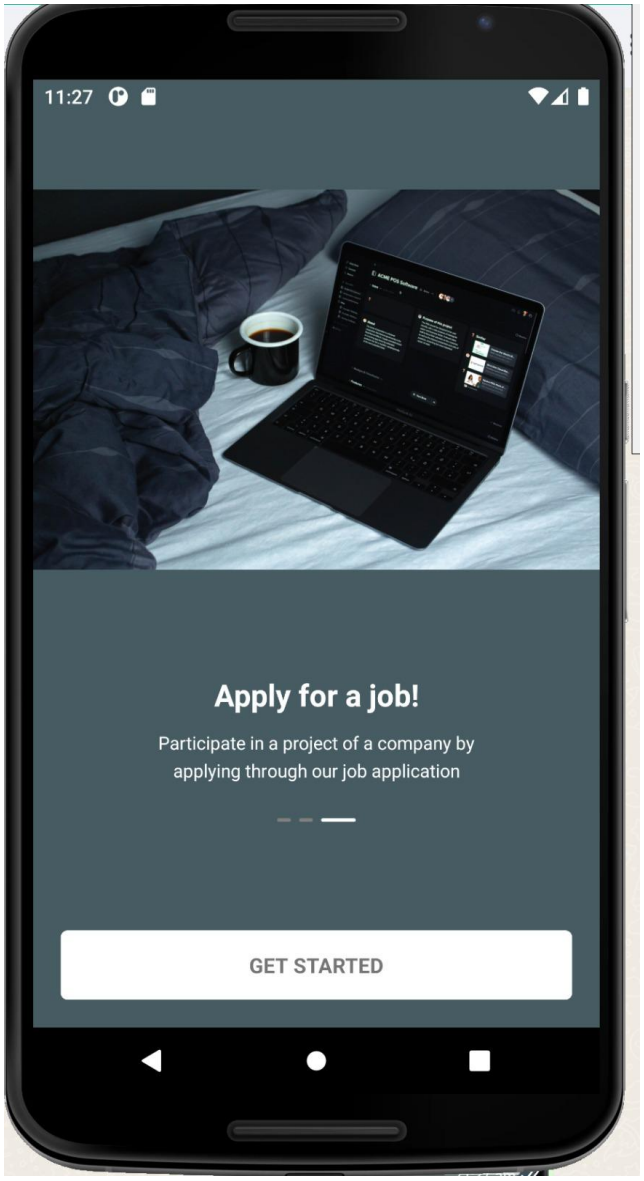
As shown in the figure, for our application interface, we have decided to give a brief introduction about the application's functionality once the users install the app on their phone. Afterward, the users will be directed to the login and registration page to create an account if they don't have one or login to an existing account. From the company's point of view, by clicking on any of the searched profiles, they can get information about the freelancer's professional life such as experience and project history as well as their full name, location, number of projects that they have done, and etc. Accordingly, if the company is interested in working with the freelancer, they can send a request to ask for joining in any project they are working on or contact them for any inquiry.

On the other hand, from a freelancer's point of view, we can view the company's information such as number of employees, number of projects, and the projects in which they are working on. If the user would like to join, they have to meet the necessary matching compatibility percentage for the skills related to the project. If so, they will be able to apply for the job.

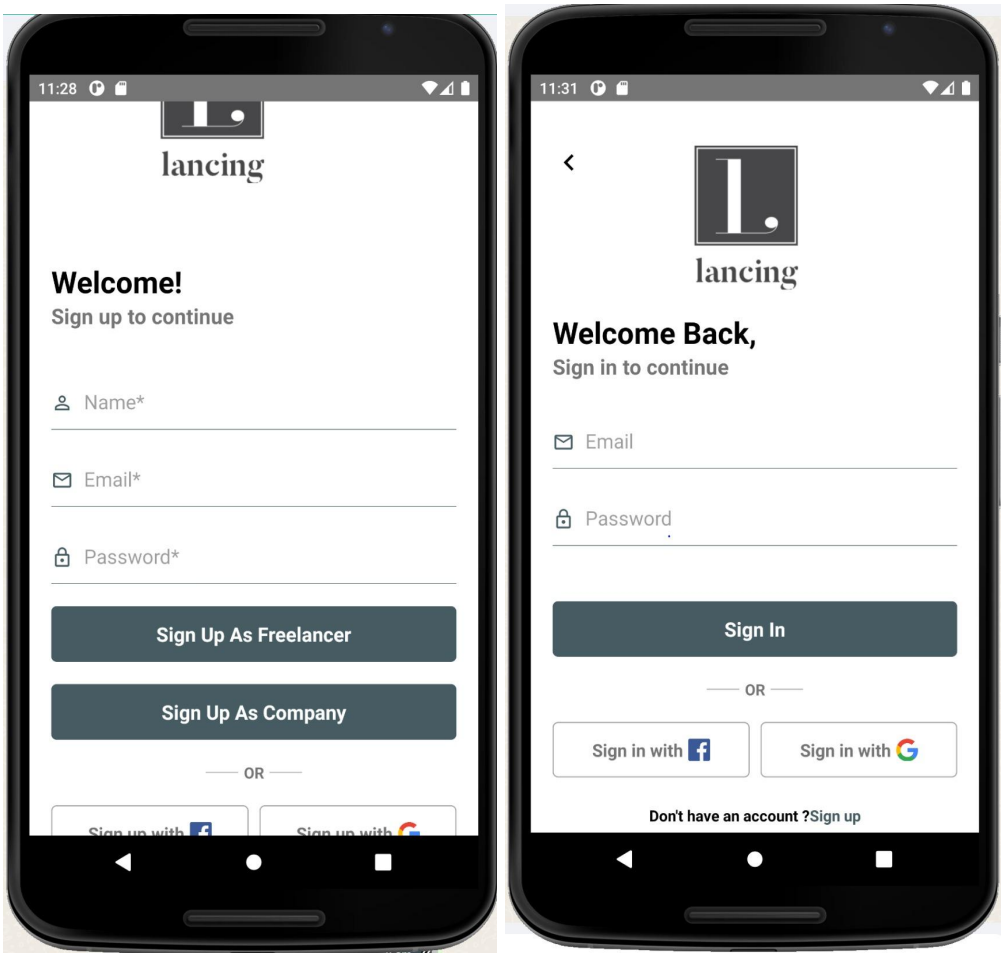
B. <u>On boarding Pages</u>



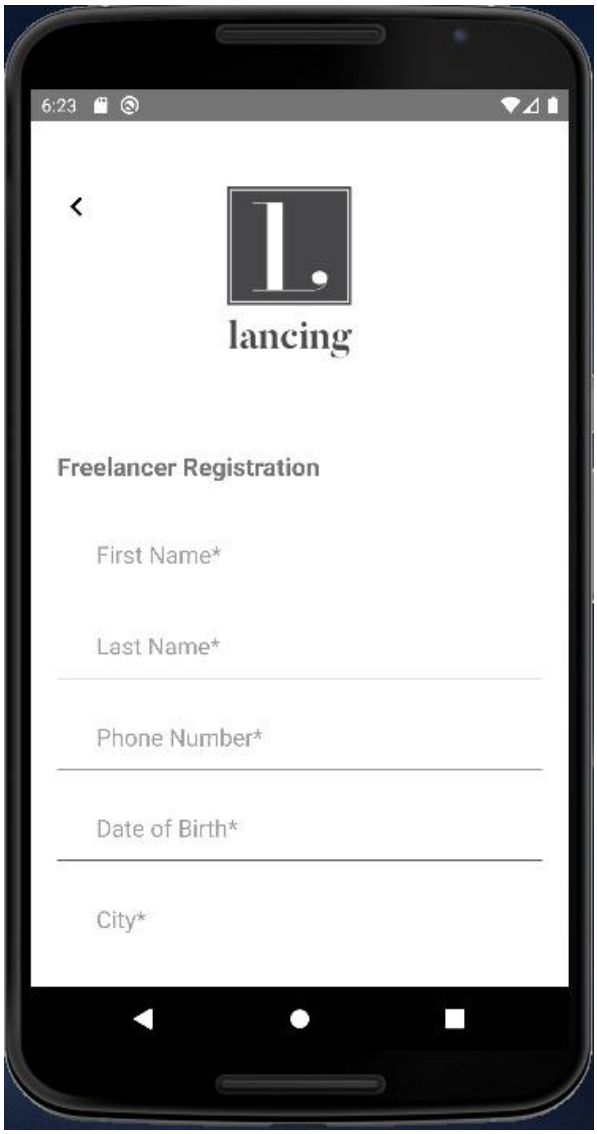

6:20
Easy Matching
Our matching algorithm will help you find your perfect professional match!
SKIP
NEXT

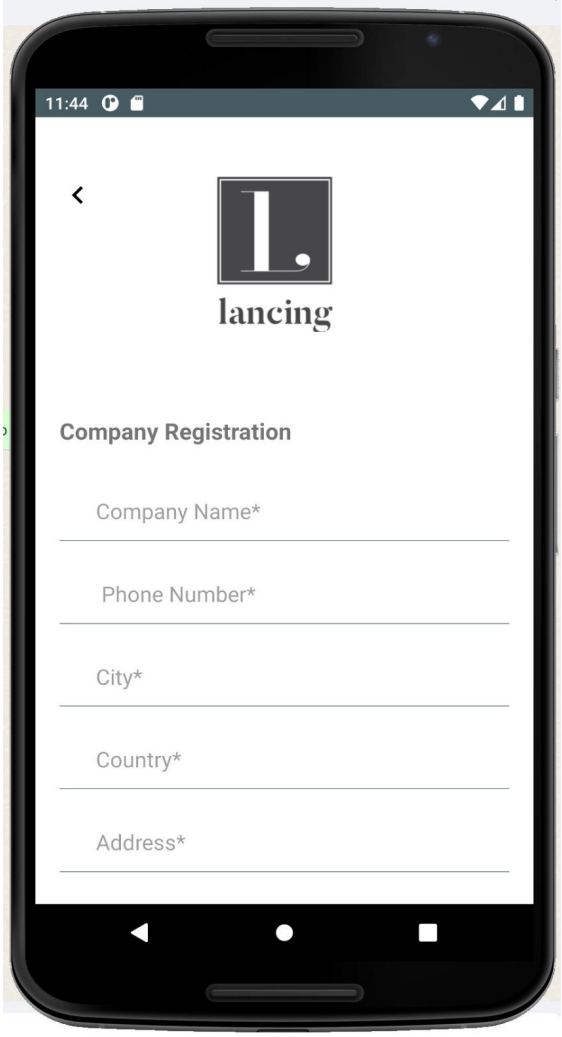

11:27
Apply for a job!
Participate in a project of a company by applying through our job application
GET STARTED

C. Log in and Sign Up



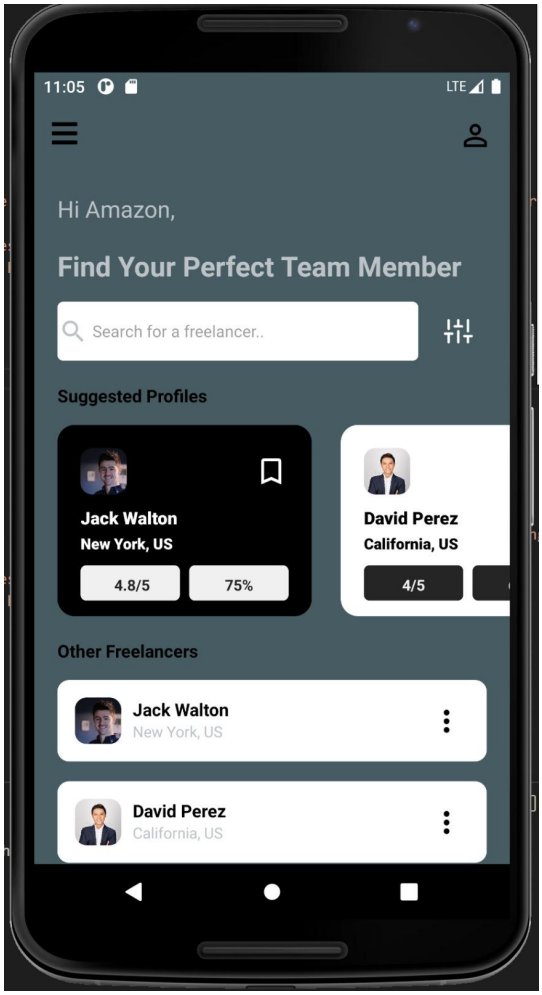
D. Registration



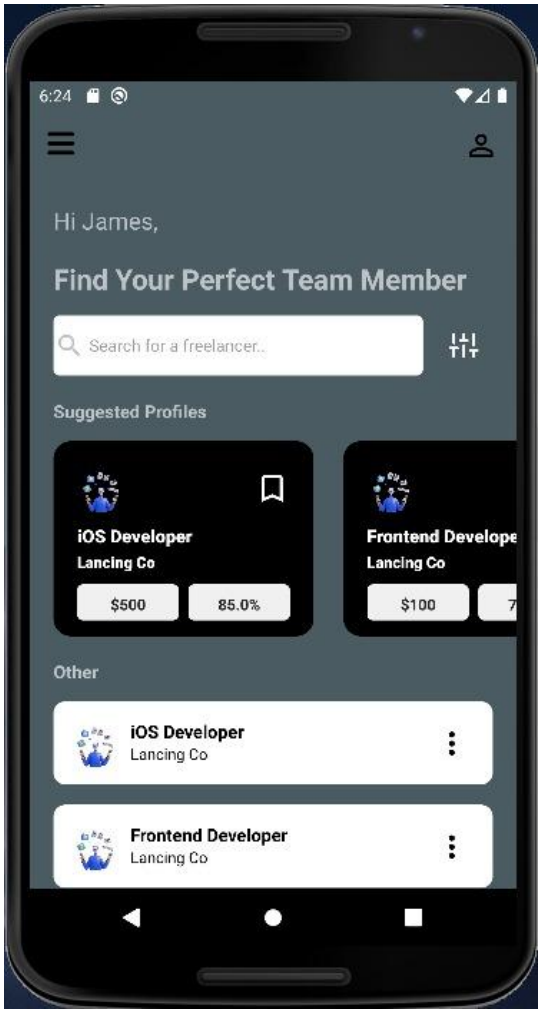

11:44
lancing
Company Registration
Company Name*
Phone Number*
City*
Country*
Address*

E. Home Pages

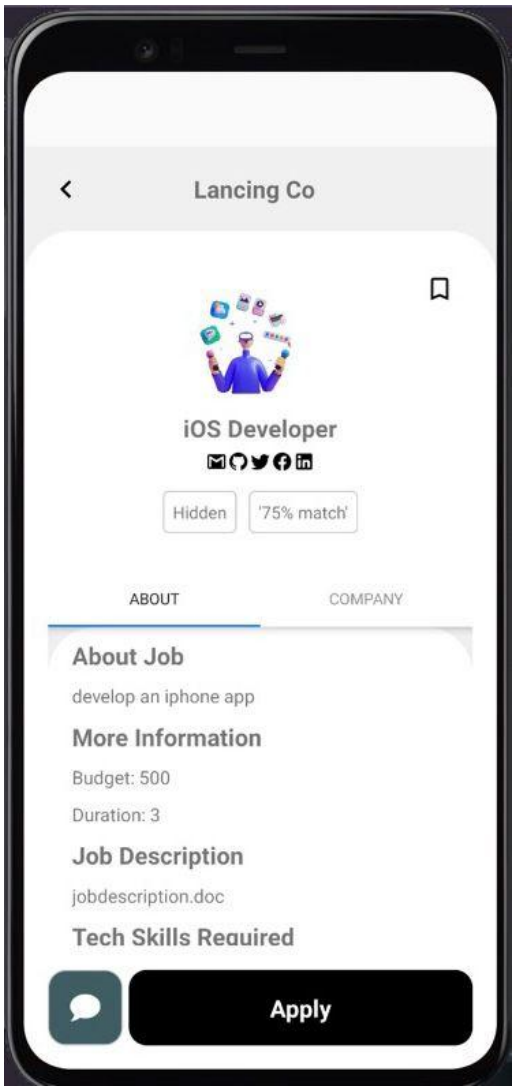
Company's Home Page



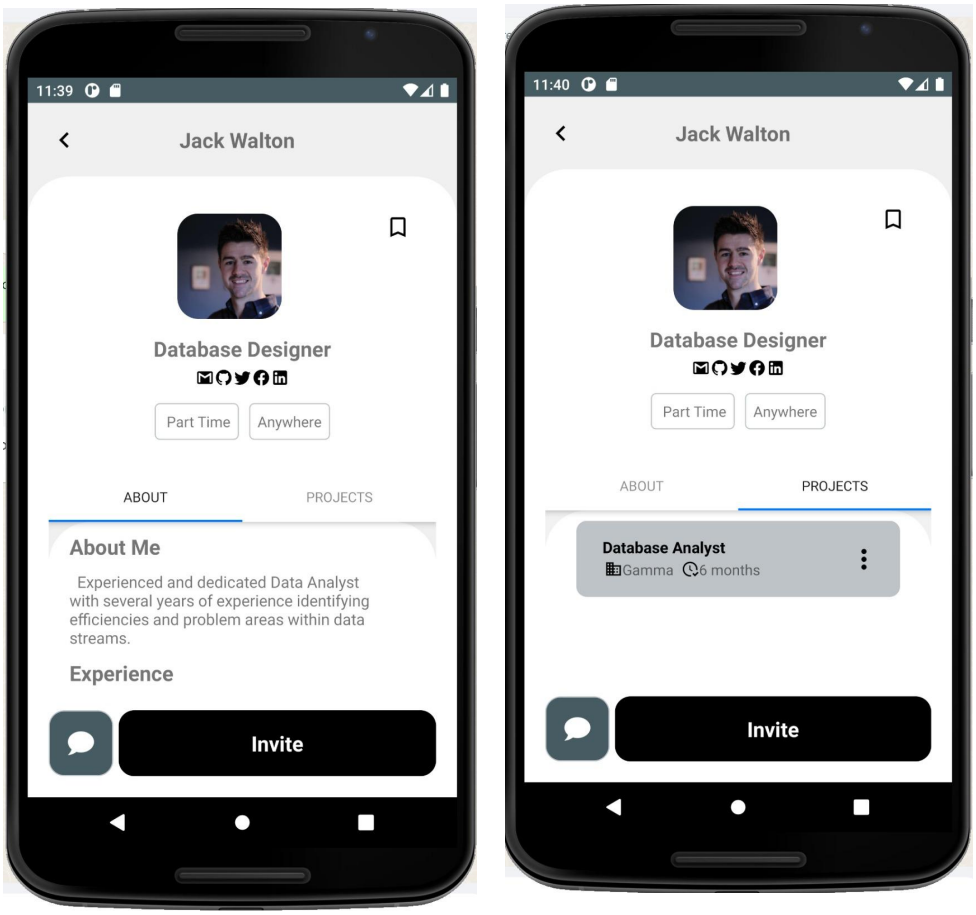
Freelancer's Home Page



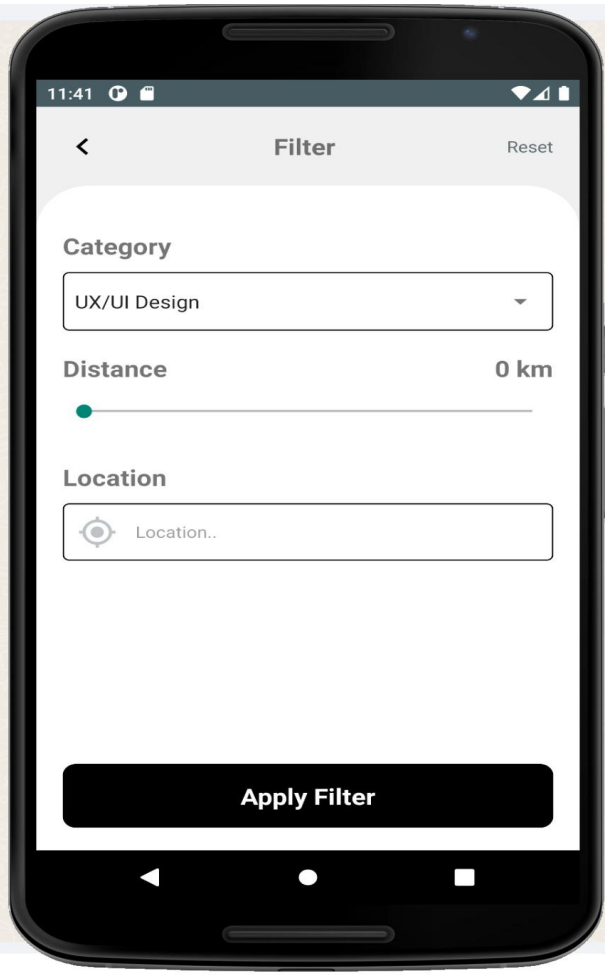
F. Job Profiles:



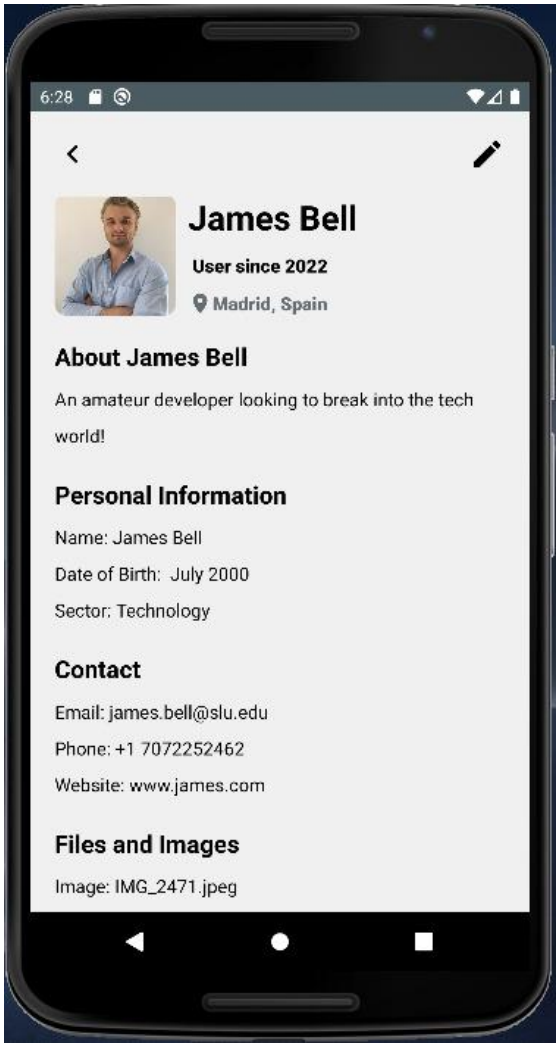
D. Freelancers Profiles



E. Search Filters



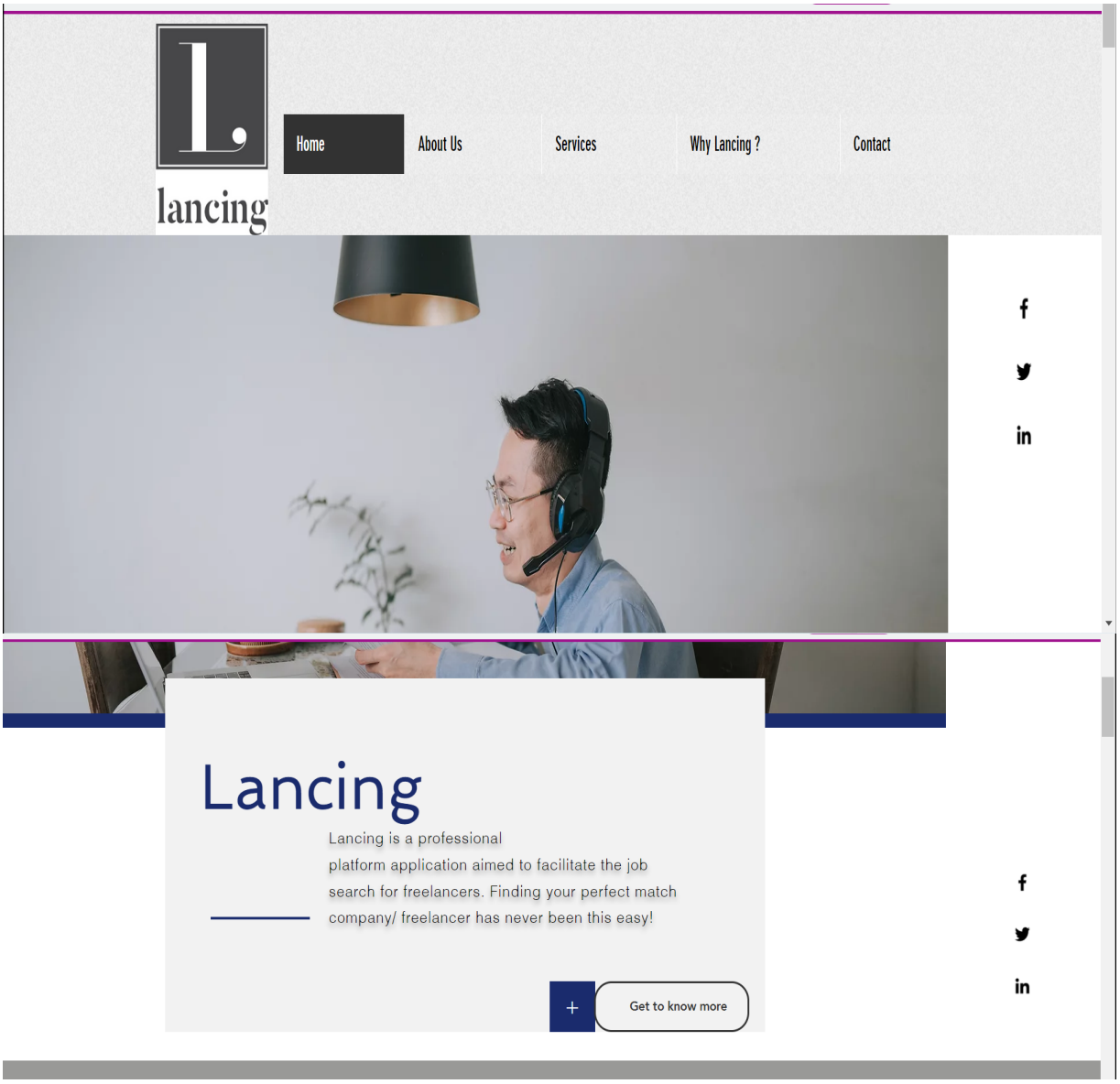
E. User Profile



F. Website Interface

As shown in the interface below, we chose to have a simple, and aesthetically pleasing design for the web page front end. The website will include four main parts: the Home page, About us, Services and Industries. Services will mainly showcase the most important functionalities that the Lancing application is offering. The added value of Lancing can be resumed in the matching system enabled by checking the boxes. The accuracy of this method will be possible thanks to a link between the databases of the freelancer's skills and the company's skill requirements for that

particular job or project. When the user searches for a job or project by entering a keyword in the search bar, the application will only generate suggested profiles that are highly likely to be of the user's interests.



OUR SERVICES

In Lancing-App we offer a unique way of applying to a project or/and freelancer's job, to both the freelancers and companies. Through a tailored matching algorithm, the freelancer is able to assess their eligibility to apply based on technical and soft skills matching.
From the company´s perspective,

WHAT PEOPLE SAY

Scott James. CS senior Student

« As a senior student, I found Lancing application to be particularly useful for making the first steps in the professional world. »

Fatima Said, CEO of Scotch

« I used Lancing platform to hire two programmers for a database project and was very satisfied with the application's usability and the quality of the freelancers profiles. »

Carla Rodriguez, IoS Engineer

« Lancing was a really useful tool in the recruitment process, given a detailed work experience, my team and I were easily able to hire candidates. »

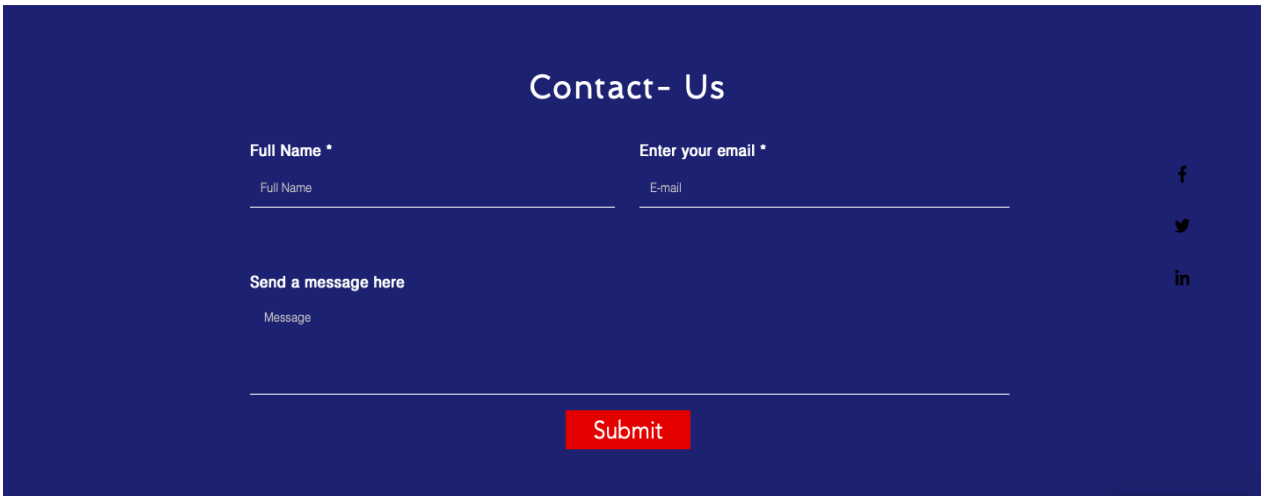
Meet The Team

Nada Saadi

Jose Fidel Paredes

James Bell

Negin Jahanbaskh



V. Testing Modules

We were able to show in the application demonstration the complete functioning of the following testing modules, as well as the connection with the database and the future implementation steps.

- ❖ Signing in/Registration:

The user of Lancing application, whether a freelancer or a company, is able to sign in to the platform by simply entering their email address and password.

- ❖ User Profile:

Once logged in, when clicking on the settings in the top right corner of the freelancers´ home page, the user is able to view their profile, contact information, technical and soft skills details.

- ❖ Job Application:

Once their profile is compatible with the job or project posted, the freelancer can make the choice to apply to this position. Once applied, an ¨Applied successfully¨ screen will popup on the user's screen, (see Figure 7). To confirm this process from the backend part, we showed the addition of this application in the database related to this job posting.

VI. Conclusion & Future Implementation

To conclude, we have implemented the application interface, onboarding pages, login and registration, and home pages for the freelancers and companies perspectives. The matching algorithm, based on the matching percentage that will show on the job application page, is calculated based on a custom weight determined by the job posting.

The connection between the backend and frontend has been successfully implemented by making the apply button work. In addition to that, the database is successfully updated with the new application registered, as can be seen in the Figure 7 below.

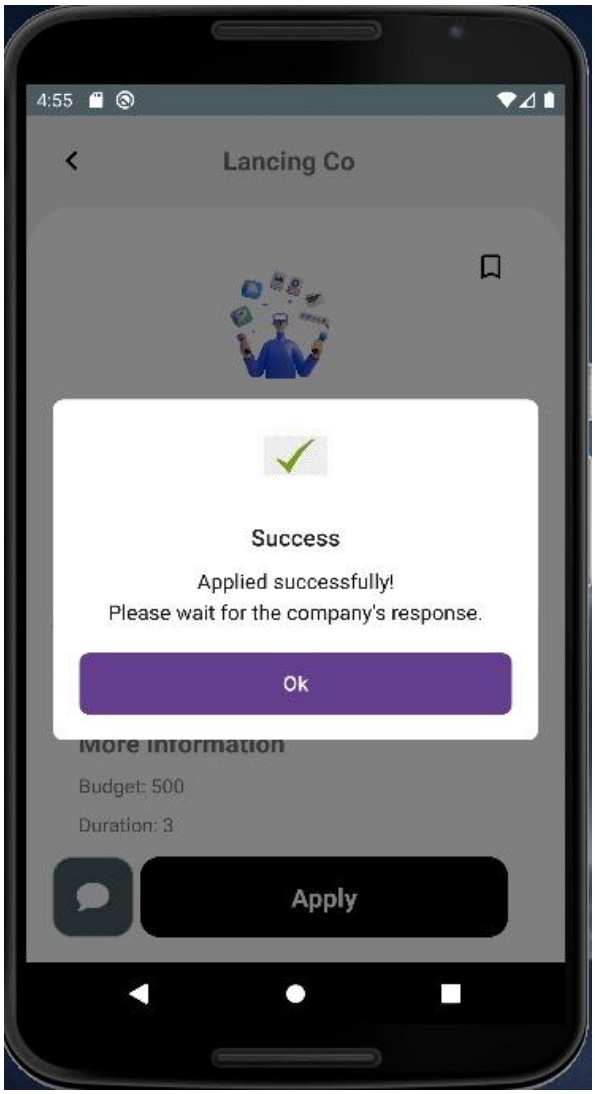


Figure 7: Application Pop Up

For future implementation, we would like to add a user menu in the freelancers home page where they will be able to view settings and update their profile. Moreover, we will also be working on having a section for freelancers to see applications submitted or saved to view later, and for the company to have the profiles of applicants interested in their jobs or projects posted saved in one section.

References

Node.js. “About.” *Node.js*, https://nodejs.org/en/about/.

“React Native · Learn Once, Write Anywhere.” *React Native RSS*, https://reactnative.dev/.

“Meet Android Studio : Android Developers.” *Android Developers*, https://developer.android.com/studio/intro.

“Technology Innovation & Integration: Software AG's Webmethods.” *Software AG*, https://www.softwareag.com

“How Does Upwork Work for Clients.” *Upwork*, https://www.upwork.com

“Meet Android Studio : Android Developers.” *Android Developers*, https://developer.android.com/studio/intro.