**Software Project - Team 3**

**Project Development Manual**

##### **Jessica Chavoya**

##### **Arturo Granados**

##### **Raven Khaya**

##### **Xuwei Lin**

##### **David Palacios**

##### **Matthew Plicinski**

##### **Tyler Wolfe-Adam**

**Requirements Document**

**(functional and Non-functional)**

# **INTRODUCTION**

## **USER PROBLEM/ PROJECT BACKGROUND**

In the current market, many “job search” websites allow companies to post jobs and candidates to apply for said jobs. However, all of them are catered for a wide audience that spans all industries and employment levels. As such, they are limited in their search scope, only allowing for basic searches based on keywords, job titles, and distances.

When looking at people and companies in the IT industry, this poses a major problem. Between the hundreds of available programming languages and a diverse range in potential focus areas (software, hardware, data analytics, Cloud computing - just to name a few), there is an immense amount of nuance in the roles.

For employers, it’s difficult to adequately convey all of their specifically desired technical, business, and cultural attributes in a job description that is comprehensive yet concise. As no other job search websites cater specifically to those in IT, the employers have to spend time manually filtering out candidates that have no related expertise and on validating the candidate’s knowledge as much as they claim. Additionally, employers currently have no way to centralize and easily manage all applicants to their business.

This service would create a way for employers to list the skills they require for a particular job and then be able to directly compare applicants based on those skills - all in one centralized system.

**GOALS OF THE PROJECT**

The project needs to allow companies to rank applicants based on compatibility in three skill categories: technology, business, and cultural (work attitudes); the compatibility will be based on the applicant’s attributes compared to the job posting.

## **SYSTEM SCOPE**

The project needs to support three types of users: candidate (individual looking for a job), company (a single representative account), and administrator (for backend maintenance). There is intended to be one candidate account per individual and one company account per company. Profiles are available for candidates to display their bio, skills, attitudes, and optionally a result. Profiles are available for companies to display their bio and available jobs.

Both candidates and companies can perform searches and filters for one another with the option to save or delete those searches or filters. Results are returned based on a best-fit algorithm.

This project is not meant to be a social media site, instead, it is meant to directly connect employers with potential candidates. This connection can happen based on basic contact information at a minimum. It should also be noted that the application is meant to be web-based only. No installation should be required to use the application.

## **CLIENT, CUSTOMER, AND OTHER STAKEHOLDERS**

The client for the product is Dr. Muscarello.

The customers for the product consist of two parties:

1. Companies and employers who want to be able to create job listings and find the most relevant candidates.
2. Individuals who are looking for a job that best fits their personal preferences and abilities.

There are no other people or organizations who have a vested interest in the product.

## **USERS OF THE PRODUCT**

The people who will be interacting with the product include:

* Administrators
  + The role of the administrator is to maintain and manage the site. They can access and edit the database directly. They also have analytical tools for the website that allows them to view and filter companies, job listings, and candidates.
* Companies / Employers:
  + Companies should be able to create and edit their profile, create job listings, and search, filter, and analyze potential candidates for their business.
* Candidates/ Individuals seeking employment:
  + Candidates should be able to create and edit their profile, upload a resume, edit their three skill/ cultural listings. Search and filter companies where they could potentially seek employment.

# **ASSUMPTIONS AND OTHER RELEVANT FACTS**

Assumptions

* It is acceptable and still practical for users that skills (technical and business) and values are limited and not open to custom input. This was done to limit the amount of potential variance from different spellings, abbreviations, or ways to reference a skill/value (e.g., “Word” vs “Microsoft Word” vs “Word (Microsoft Office)”, etc.). A positive result from this is that search and matching scores are improved.
* All users are based out of the U.S.
* Companies are not interested in creating general posts to be listed on their profile, akin to a social media site. Instead, employer pages will only consist of the relevant profile information for that business and their job listings.
* Individuals using the filters are looking for options to be selected in the “OR” condition and not to specifically exclude the options not selected. For example, under “work type” filters, selecting “full time” and “part time” but not “contract” will show results that have the first two options included. If the result only has “contract”, this will be included since the selected option takes precedence.
* Admins do not have a profile, as they are not shown on the site.
* It’s acceptable for inactive jobs to be shown on the company’s profile page (with some visual or message indicator). The important point is that they’re not shown in the “Jobs” search page results.

# **BUSINESS REQUIREMENTS**

**SEE COMPANION FILE: BUSINESS\_REQUIREMENTS.xlsx**

# **USER REQUIREMENTS**

## **FUNCTIONALITY REQUIREMENTS**

### **Processing Requirements**

The process which will be implemented by the system include: new profile creation, login and logout, account deletion, account/ profile editing, analytical tools, browsing/ searching companies and candidates, creating, editing, and closing job listings, uploading documents, rank ordered sorting, and companies being able to contact candidates.

### **Information Requirements**N/A

## **USABILITY REQUIREMENTS**

### **Ease-of-use Requirements**

Some ease-of-use requirements include:

* Searching tools that can be very simple and search tools that allow more detailed filtering. Filtering includes location, skill requirements, cultural preferences, etc.
* Simple guided profile creation upon account registration as well as more detailed profile editing options after the account is created.
* Analytical tools for companies that range from simply listing candidates in ranked order to creating graphs based on the properties of candidates.

### **Documentation Requirements**

The documentation requirements include help pages/ tutorials on using the analytical tools that the application offers.

### **Safety Requirements**If the system were to be hacked, personal data may be leaked. To help reduce this risk, connections will be over HTTPS and access to the database will be limited.

## **PERFORMANCE REQUIREMENTS**

### **Availability Requirements**

The availability of the system is required to be twenty four hours a day, seven days a week.

### **Responsiveness Requirements**

The system’s responsiveness should be quite quick for simple use cases. For example loading/ viewing a profile page or doing a simple search should execute within a few seconds. The responsiveness of the more advanced analytical tools may be a bit slower but not excessively so. Users should not be waiting several minutes or more for analytical tools to finish executing.

### **Reliability Requirements**The expected reliability of the system is to be as close to 100% as possible.

### **Capacity Requirements**

Initially the capacity of the system should be able to handle a few hundred users including both companies and candidates. However it should be easily scalable to be able to handle thousands or more of users.

### **Scalability Requirements**

The expected increase of usage of the system over time is from several hundred users to thousands, tens of thousands, and beyond that while maintaining quick response time.

### **Disaster Recovery & Business Continuity Requirements**

In the case of disaster, the required operation is to recover the application data from backup databases and restore the functionality and data of the application as quickly as possible.

## **SECURITY REQUIREMENTS**

### **User Security Requirements**

User accounts will require a username and password to login. Only registered companies will have access to searching for candidates and candidates will not be able to see one another. Connections will be required to be made over HTTPS (encrypted).

### **Data Security Requirements**

The requirements for storing sensitive information include ensuring that only the appropriate users can view certain private information when it is appropriate. For example only a company should be able to view a candidate's information and they should only be able to view that candidate's information if they are a potential match for one of their job listings. Another example: A candidate should never be able to view another candidate's private information. Only the application administrators should have access to the full database of the website.

## **LEGAL COMPLIANCE REQUIREMENTS**

### **Notification Requirements**

The notification requirements of the system are to require users to Opt-In to an agreement about the intended use of the application.

### **Privacy Requirements**

Only registered companies will be able to openly search for candidate users. Any user is able to delete their account and no longer have their information publicly listed. Data will be stored in a secured database with limited access, with communication occurring over an established API.

### **Funding Requirements or Appropriation StipulationsN/A**

# **REQUIREMENT CONSTRAINTS & DEPENDENCIES**

## **DESIGN CONSTRAINT**

| **ID** | **Constraint** |
| --- | --- |
| C.1.1 | Web-based application only |
| C.1.2 |  |

## 

## **PROJECT CONSTRAINT**

| **ID** | **Constraint** |
| --- | --- |
| C.2.1 | Project preview due May 26 2021 |
| C.2.2 | Final delivery due June 9th 2021 |
| C.2.3 | Target budget of $0.00, Max budget of $50.00 |
|  |  |

# 

**Design/Architecture**

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **Scope**

Hardware: N/A

Software: Python v3.9

## **Audience**

## Below is a list of the site audience:

* Job seekers
  + For site usage, basic computer skills are required
  + Assumed to be knowledgeable about website registration, filling out forms and basic search queries, and the ideas behind “bookmarking”
* Companies
  + For site usage, basic computer skills are required.
  + Assumed to be knowledgeable about website registration and filling out forms and basic search queries.
  + It’s assumed they have basic data analytical skills.
* Administrators
  + For site usage, it’s required to be familiar with the idea behind relational databases (to best be able to manipulate database entries).
  + It’s assumed they have basic data analytical skills.

## **Related Documentation**

| File name | Purpose |
| --- | --- |
| SP2021 Project Requirements | Overview of what is required in the system; provided by the custom. |
| Business User Requirements | A formal declaration of requirements for both the business-case and the user-case. |
| Test Plan | Detail of test scope, methodologies, and validation requirements. |

## **Document Conventions**

## The database diagram uses typical entity-relationship formats. Other views are non-specific and described in-line.

# **System Overview**

# **Description**

# The system involves accommodating job seekers and IT companies. Job seekers will be able to create a profile that shows their skills and attitudes, browse companies, seek and save jobs they are interested in, list their own references, as well as upload their resume for companies to see. Companies will be able to create an environment where they can create, post, edit, or close jobs, browse candidates that match the company’s requirements, review any outstanding searches, and contact job seekers for follow-up. Overall, this software project should be able to accommodate both job seekers and companies looking to utilize an IT HR job site tool.

## **System Architecture**

## **Software Architecture**

## The website will be based in Python via the Flask framework. Built into this is the Jinja library, allowing template HTML files to be used to inject content into. SQLAlchemy will be used to abstract away from the underlying PostgreSQL database (allowing for the tables to be declared in Python).

## **Hardware Architectures**

## Hardware architecture is not applicable to this specific project. Heroku will be used and thus the specifications for hardware are abstracted away from the project.

## **Hardware Design**

## **Hardware Components**

## **Computer Systems**

## N/A

## **Peripherals**

## N/A

## **Networks**

## N/A

## **Project Specific Hardware Items**

## N/A

# **Hardware Integration**

### **Logical Design**

### N/A

### **Physical Design**

### N/A

### **Recovery Design**

### N/A

# **Software Design**

# **Software Packages**

“app”: Where the project is stored. It has modules for global constants and database models. It consists of four subpackages:

1. “api”: Provides a public endpoint for querying/updating the database regarding users and for the searching/matching algorithms
2. “auth”: Provides the HTTP routes and web forms related to authentication (registration, log-in)
3. “main”: Provides the HTTP routes and web forms related to the main sources of interaction (job posting, searching, editing/viewing profiles)
4. “errors”: Provides the handlers for assorted web errors (e.g., 404, 500)

## **Software Integration**

## 

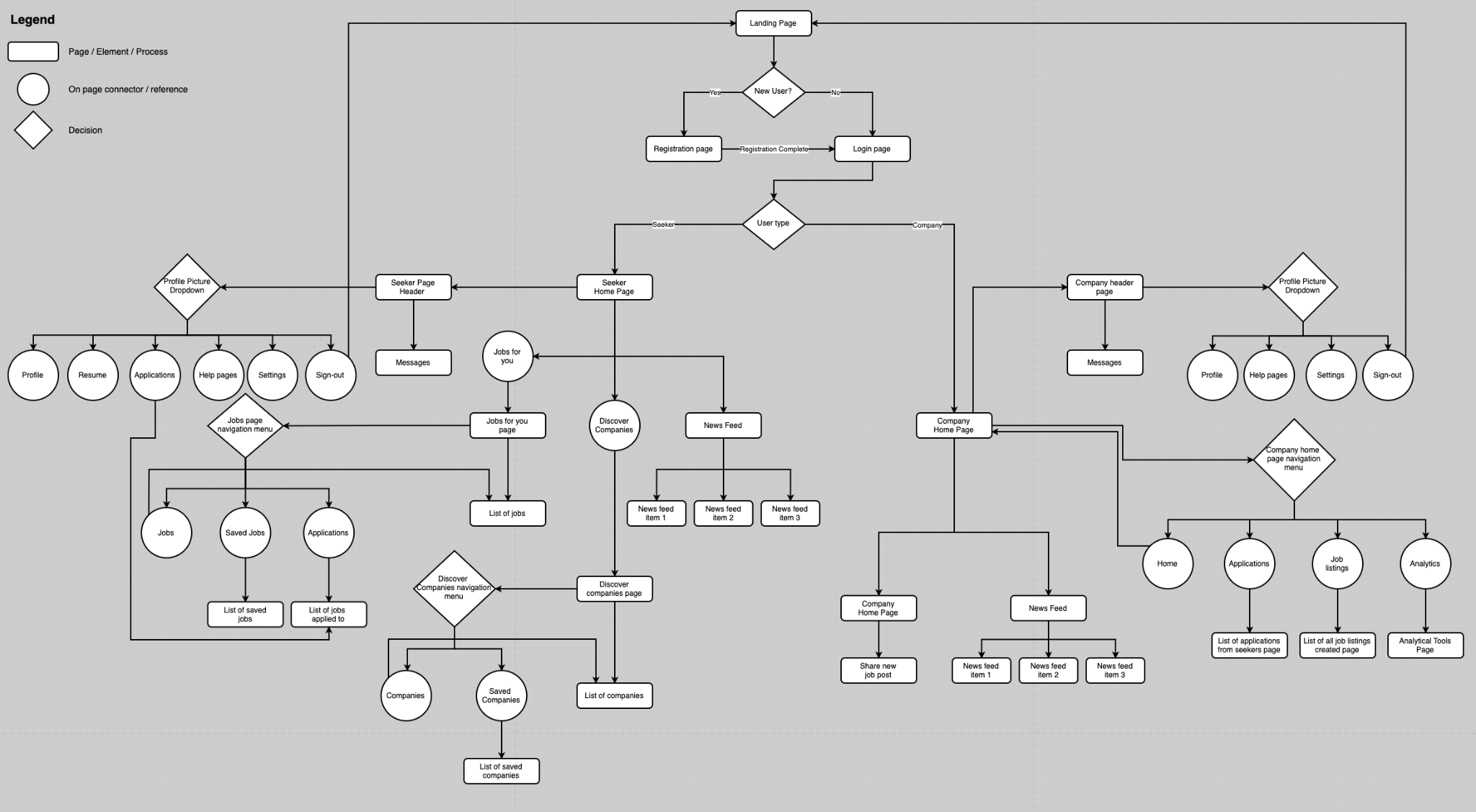
*Import interaction across packages (beige) and modules (blue)*

All links within the software are within its own codebase and the database, which is connected at the initialization of the “app” module.

The interactions shown in the above image depict both variable and function imports. Some variables that are accessible to the module are located within the initialization code (\_\_init\_\_.py) located in the module; this is the reason why some links are pointing towards modules

**Data / Database / Files**

## **Data Flow Diagrams**

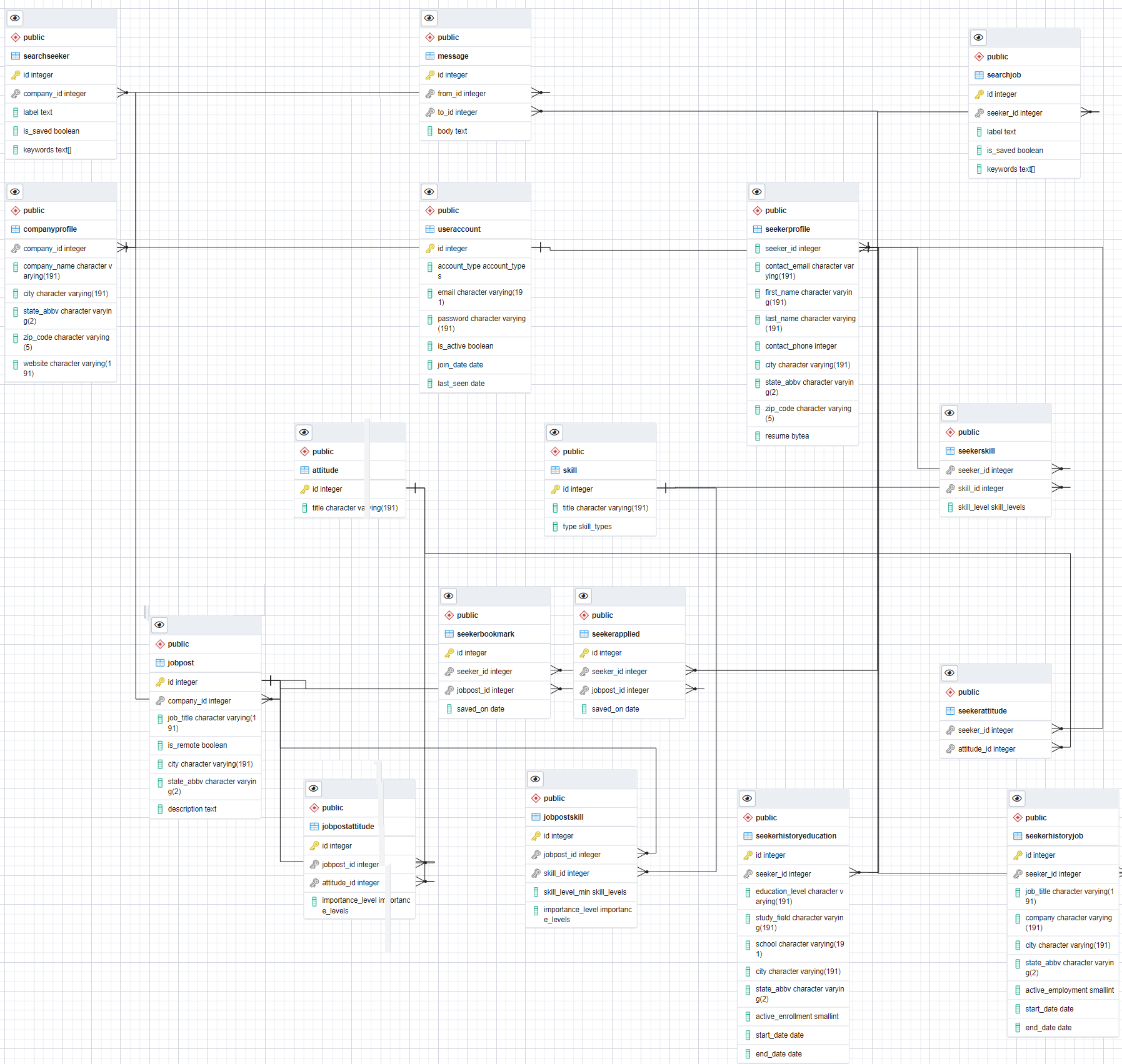


## **Database Design**

A PostgreSQL database is used with the following tables. Relationships can be seen in the image below.

| **List of relations** | | |
| --- | --- | --- |
| **Schema** | **Name** | **Type** |
| public | alembic\_version | table |
| public | attitude | table |
| public | company\_profile | table |
| public | company\_seeker\_search | table |
| public | jobpost | table |
| public | jobpost\_attitude | table |
| public | jobpost\_skill | table |
| public | location\_coordinates | table |
| public | match\_scores | table |
| public | seeker\_application | table |
| public | seeker\_attitude | table |
| public | seeker\_bookmark | table |
| public | seeker\_history\_education | table |
| public | seeker\_history\_job | table |
| public | seeker\_job\_search | table |
| public | seeker\_profile | table |
| public | seeker\_skill | table |
| public | skill | table |
| public | user | table |
| public | user\_picture | table |

| **Table "public.attitude"** | | | | |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('attitude\_id\_seq'::regclass) | plain |
| title | character varying(191) | not null |  | extended |
| **Indexes:** |  |  |  |  |
| attitude\_pkey PRIMARY KEY, btree (id) | | | |  |
| attitude\_title\_key UNIQUE CONSTRAINT, btree (title) | | | | |
| **Referenced by:** | |  |  |  |
| TABLE "jobpost\_attitude" CONSTRAINT "jobpost\_attitude\_attitude\_id\_fkey" FOREIGN KEY (attitude\_id) REFERENCES attitude(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_attitude" CONSTRAINT "seeker\_attitude\_attitude\_id\_fkey" FOREIGN KEY (attitude\_id) REFERENCES attitude(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.company\_profile"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('company\_profile\_id\_seq'::regclass) | plain |
| user\_id | integer | not null |  | plain |
| name | character varying(191) | not null |  | extended |
| city | character varying(191) | | | extended |
| state | character varying(2) | |  | extended |
| website | character varying(191) | | | extended |
| tagline | character varying(100) | | | extended |
| summary | character varying | |  | extended |
| **Indexes:** |  |  |  |  |
| company\_profile\_pkey PRIMARY KEY, btree (id) | | | | |
| company\_profile\_user\_id\_key UNIQUE CONSTRAINT, btree (user\_id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| company\_profile\_user\_id\_fkey FOREIGN KEY (user\_id) REFERENCES "user"(id) ON DELETE CASCADE | | | | |
| **Referenced by:** | |  |  |  |
| TABLE "jobpost" CONSTRAINT "jobpost\_company\_id\_fkey" FOREIGN KEY (company\_id) REFERENCES company\_profile(id) | | | | |
|  |  |  |  |  |
| **Table "public.company\_seeker\_search"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('company\_seeker\_search\_id\_seq'::regclass) | plain |
| user\_id | integer | not null |  | plain |
| label | character varying | not null |  | extended |
| query | character varying | not null |  | extended |
| **Indexes:** |  |  |  |  |
| company\_seeker\_search\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| company\_seeker\_search\_user\_id\_fkey FOREIGN KEY (user\_id) REFERENCES "user"(id) | | | | |
|  |  |  |  |  |
| **Table "public.jobpost"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('jobpost\_id\_seq'::regclass) | plain |
| company\_id | integer | not null |  | plain |
| job\_title | character varying(191) | not null |  | extended |
| city | character varying(191) | | | extended |
| state | character varying(2) | |  | extended |
| description | text |  |  | extended |
| work\_type | worktypes | |  | plain |
| is\_remote | boolean |  |  | plain |
| salary\_min | integer |  |  | plain |
| salary\_max | integer |  |  | plain |
| created\_timestamp | timestamp without time zone | | | plain |
| active | boolean |  |  | plain |
| **Indexes:** |  |  |  |  |
| jobpost\_pkey PRIMARY KEY, btree (id) | | | |  |
| **Foreign-key constraints:** | | |  |  |
| jobpost\_company\_id\_fkey FOREIGN KEY (company\_id) REFERENCES company\_profile(id) | | | | |
| **Referenced by:** | |  |  |  |
| TABLE "jobpost\_attitude" CONSTRAINT "jobpost\_attitude\_jobpost\_id\_fkey" FOREIGN KEY (jobpost\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
| TABLE "jobpost\_skill" CONSTRAINT "jobpost\_skill\_jobpost\_id\_fkey" FOREIGN KEY (jobpost\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
| TABLE "match\_scores" CONSTRAINT "match\_scores\_jobpost\_id\_fkey" FOREIGN KEY (jobpost\_id) REFERENCES jobpost(id) | | | | |
| TABLE "seeker\_application" CONSTRAINT "seeker\_application\_job\_id\_fkey" FOREIGN KEY (job\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_bookmark" CONSTRAINT "seeker\_bookmark\_job\_id\_fkey" FOREIGN KEY (job\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.jobpost\_attitude"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('jobpost\_attitude\_id\_seq'::regclass) | plain |
| jobpost\_id | integer | not null |  | plain |
| attitude\_id | integer | not null |  | plain |
| importance\_level | importancelevel | |  | plain |
| **Indexes:** |  |  |  |  |
| jobpost\_attitude\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| jobpost\_attitude\_attitude\_id\_fkey FOREIGN KEY (attitude\_id) REFERENCES attitude(id) ON DELETE CASCADE | | | | |
| jobpost\_attitude\_jobpost\_id\_fkey FOREIGN KEY (jobpost\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.jobpost\_skill"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('jobpost\_skill\_id\_seq'::regclass) | plain |
| jobpost\_id | integer | not null |  | plain |
| skill\_id | integer | not null |  | plain |
| skill\_level\_min | skilllevels | |  | plain |
| importance\_level | importancelevel | |  | plain |
| **Indexes:** |  |  |  |  |
| jobpost\_skill\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| jobpost\_skill\_jobpost\_id\_fkey FOREIGN KEY (jobpost\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
| jobpost\_skill\_skill\_id\_fkey FOREIGN KEY (skill\_id) REFERENCES skill(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.location\_coordinates"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| location | character varying | not null |  | extended |
| latitude | numeric |  |  | main |
| longitude | numeric |  |  | main |
| **Indexes:** |  |  |  |  |
| location\_coordinates\_pkey PRIMARY KEY, btree (location) | | | | |
|  |  |  |  |  |
| **Table "public.match\_scores"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('match\_scores\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| jobpost\_id | integer | not null |  | plain |
| score | numeric | not null |  | main |
| **Indexes:** |  |  |  |  |
| match\_scores\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| match\_scores\_jobpost\_id\_fkey FOREIGN KEY (jobpost\_id) REFERENCES jobpost(id) | | | | |
| match\_scores\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_application"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_application\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| job\_id | integer | not null |  | plain |
| **Indexes:** |  |  |  |  |
| seeker\_application\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_application\_job\_id\_fkey FOREIGN KEY (job\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
| seeker\_application\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_attitude"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_attitude\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| attitude\_id | integer | not null |  | plain |
| **Indexes:** |  |  |  |  |
| seeker\_attitude\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_attitude\_attitude\_id\_fkey FOREIGN KEY (attitude\_id) REFERENCES attitude(id) ON DELETE CASCADE | | | | |
| seeker\_attitude\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_bookmark"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_bookmark\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| job\_id | integer | not null |  | plain |
| **Indexes:** |  |  |  |  |
| seeker\_bookmark\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_bookmark\_job\_id\_fkey FOREIGN KEY (job\_id) REFERENCES jobpost(id) ON DELETE CASCADE | | | | |
| seeker\_bookmark\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_history\_education"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_history\_education\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| school | character varying | not null |  | extended |
| education\_lvl | educationlevel | not null |  | plain |
| study\_field | character varying | not null |  | extended |
| **Indexes:** |  |  |  |  |
| seeker\_history\_education\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_history\_education\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_history\_job"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_history\_job\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| job\_title | character varying(191) | | | extended |
| years\_employed | integer |  |  | plain |
| **Indexes:** |  |  |  |  |
| seeker\_history\_job\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_history\_job\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_job\_search"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_job\_search\_id\_seq'::regclass) | plain |
| user\_id | integer | not null |  | plain |
| label | character varying | not null |  | extended |
| query | character varying | not null |  | extended |
| **Indexes:** |  |  |  |  |
| seeker\_job\_search\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_job\_search\_user\_id\_fkey FOREIGN KEY (user\_id) REFERENCES "user"(id) | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_profile"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_profile\_id\_seq'::regclass) | plain |
| user\_id | integer | not null |  | plain |
| first\_name | character varying(191) | not null |  | extended |
| last\_name | character varying(191) | not null |  | extended |
| phone\_number | character varying(10) | | | extended |
| city | character varying(191) | | | extended |
| state | character varying(2) | |  | extended |
| work\_wanted | worktypes | |  | plain |
| remote\_wanted | boolean |  |  | plain |
| tagline | character varying(100) | | | extended |
| summary | character varying | |  | extended |
| resume | bytea |  |  | extended |
| **Indexes:** |  |  |  |  |
| seeker\_profile\_pkey PRIMARY KEY, btree (id) | | | | |
| seeker\_profile\_user\_id\_key UNIQUE CONSTRAINT, btree (user\_id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_profile\_user\_id\_fkey FOREIGN KEY (user\_id) REFERENCES "user"(id) | | | | |
| **Referenced by:** | |  |  |  |
| TABLE "match\_scores" CONSTRAINT "match\_scores\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) | | | | |
| TABLE "seeker\_application" CONSTRAINT "seeker\_application\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_attitude" CONSTRAINT "seeker\_attitude\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_bookmark" CONSTRAINT "seeker\_bookmark\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_history\_education" CONSTRAINT "seeker\_history\_education\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_history\_job" CONSTRAINT "seeker\_history\_job\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_skill" CONSTRAINT "seeker\_skill\_seeker\_id\_fkey" FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.seeker\_skill"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('seeker\_skill\_id\_seq'::regclass) | plain |
| seeker\_id | integer | not null |  | plain |
| skill\_id | integer | not null |  | plain |
| skill\_level | skilllevels | not null |  | plain |
| **Indexes:** |  |  |  |  |
| seeker\_skill\_pkey PRIMARY KEY, btree (id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| seeker\_skill\_seeker\_id\_fkey FOREIGN KEY (seeker\_id) REFERENCES seeker\_profile(id) ON DELETE CASCADE | | | | |
| seeker\_skill\_skill\_id\_fkey FOREIGN KEY (skill\_id) REFERENCES skill(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.skill"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('skill\_id\_seq'::regclass) | plain |
| title | character varying(191) | not null |  | extended |
| type | skilltypes | not null |  | plain |
| **Indexes:** |  |  |  |  |
| skill\_pkey PRIMARY KEY, btree (id) | | | |  |
| skill\_title\_key UNIQUE CONSTRAINT, btree (title) | | | | |
| **Referenced by:** | |  |  |  |
| TABLE "jobpost\_skill" CONSTRAINT "jobpost\_skill\_skill\_id\_fkey" FOREIGN KEY (skill\_id) REFERENCES skill(id) ON DELETE CASCADE | | | | |
| TABLE "seeker\_skill" CONSTRAINT "seeker\_skill\_skill\_id\_fkey" FOREIGN KEY (skill\_id) REFERENCES skill(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.user"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| id | integer | not null | nextval('user\_id\_seq'::regclass) | plain |
| account\_type | accounttypes | not null |  | plain |
| email | character varying(191) | not null |  | extended |
| password | character varying(191) | not null |  | extended |
| is\_active | boolean |  |  | plain |
| join\_date | timestamp without time zone | not null |  | plain |
| last\_login | timestamp without time zone | not null |  | plain |
| **Indexes:** |  |  |  |  |
| user\_pkey PRIMARY KEY, btree (id) | | | |  |
| user\_email\_key UNIQUE CONSTRAINT, btree (email) | | | | |
| **Referenced by:** | |  |  |  |
| TABLE "company\_profile" CONSTRAINT "company\_profile\_user\_id\_fkey" FOREIGN KEY (user\_id) REFERENCES "user"(id) ON DELETE CASCADE | | | | |
| TABLE "company\_seeker\_search" CONSTRAINT "company\_seeker\_search\_user\_id\_fkey" FOREIGN KEY (user\_id) REFERENCES "user"(id) | | | | |
| TABLE "seeker\_job\_search" CONSTRAINT "seeker\_job\_search\_user\_id\_fkey" FOREIGN KEY (user\_id) REFERENCES "user"(id) | | | | |
| TABLE "seeker\_profile" CONSTRAINT "seeker\_profile\_user\_id\_fkey" FOREIGN KEY (user\_id) REFERENCES "user"(id) | | | | |
| TABLE "user\_picture" CONSTRAINT "user\_picture\_user\_id\_fkey" FOREIGN KEY (user\_id) REFERENCES "user"(id) ON DELETE CASCADE | | | | |
|  |  |  |  |  |
| **Table "public.user\_picture"** | | | | |
| **Column** | **Type** | **Nullable** | **Default** | **Storage** |
| width | integer | not null |  | plain |
| height | integer | not null |  | plain |
| mimetype | character varying(255) | not null |  | extended |
| original | boolean | not null |  | plain |
| created\_at | timestamp with time zone | not null |  | plain |
| user\_id | integer | not null |  | plain |
| **Indexes:** |  |  |  |  |
| user\_picture\_pkey PRIMARY KEY, btree (width, height, user\_id) | | | | |
| **Foreign-key constraints:** | | |  |  |
| user\_picture\_user\_id\_fkey FOREIGN KEY (user\_id) REFERENCES "user"(id) ON DELETE CASCADE | | | | |



## **Registry / System Parameters**

Two system variables are used:

1. “FLASK\_APP”: refers to the entry point for the program.
2. “DATABASE\_URL”: the URI of the database for the program to connect to.

# **System Interfaces**

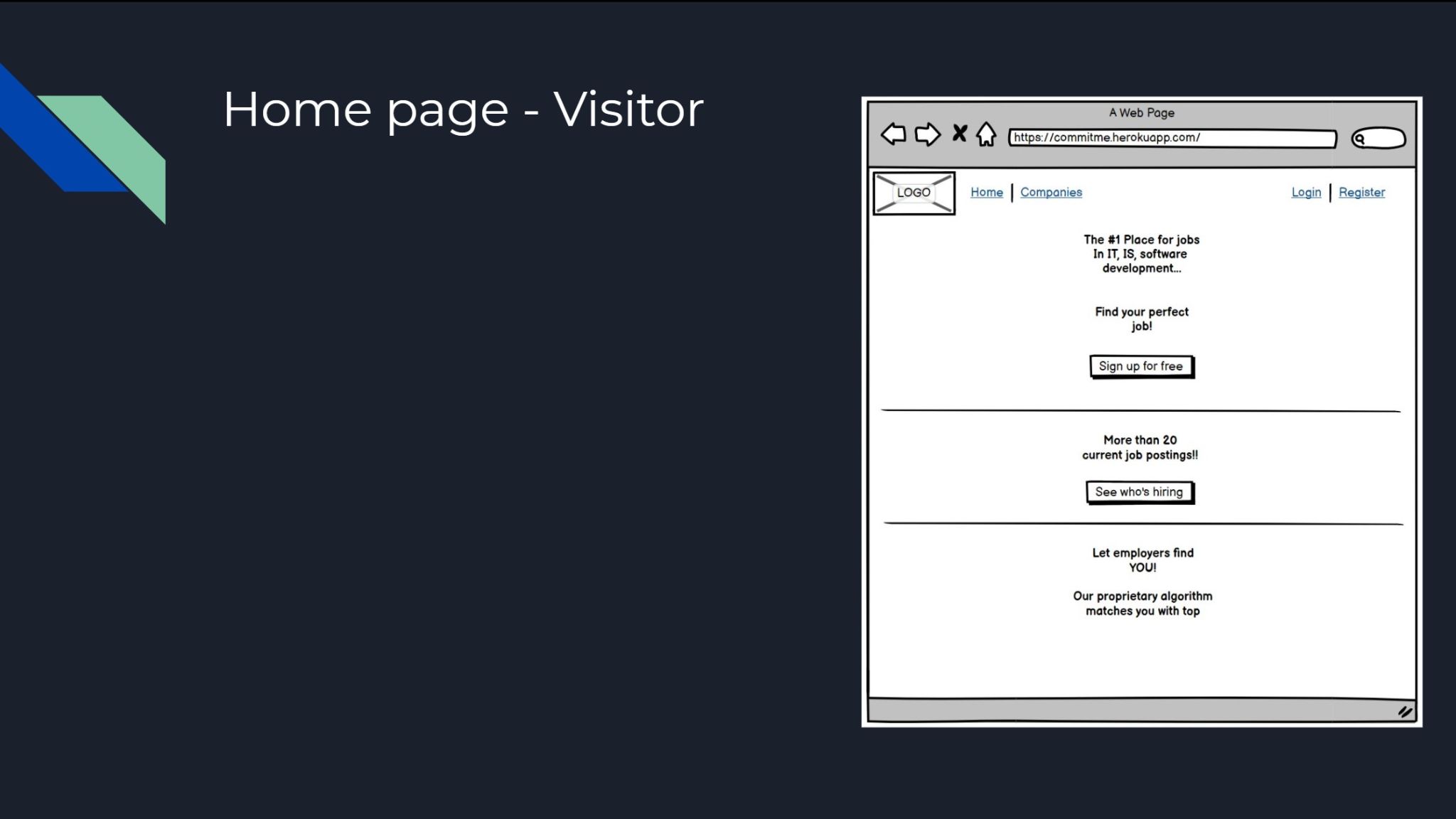
## **Interface**

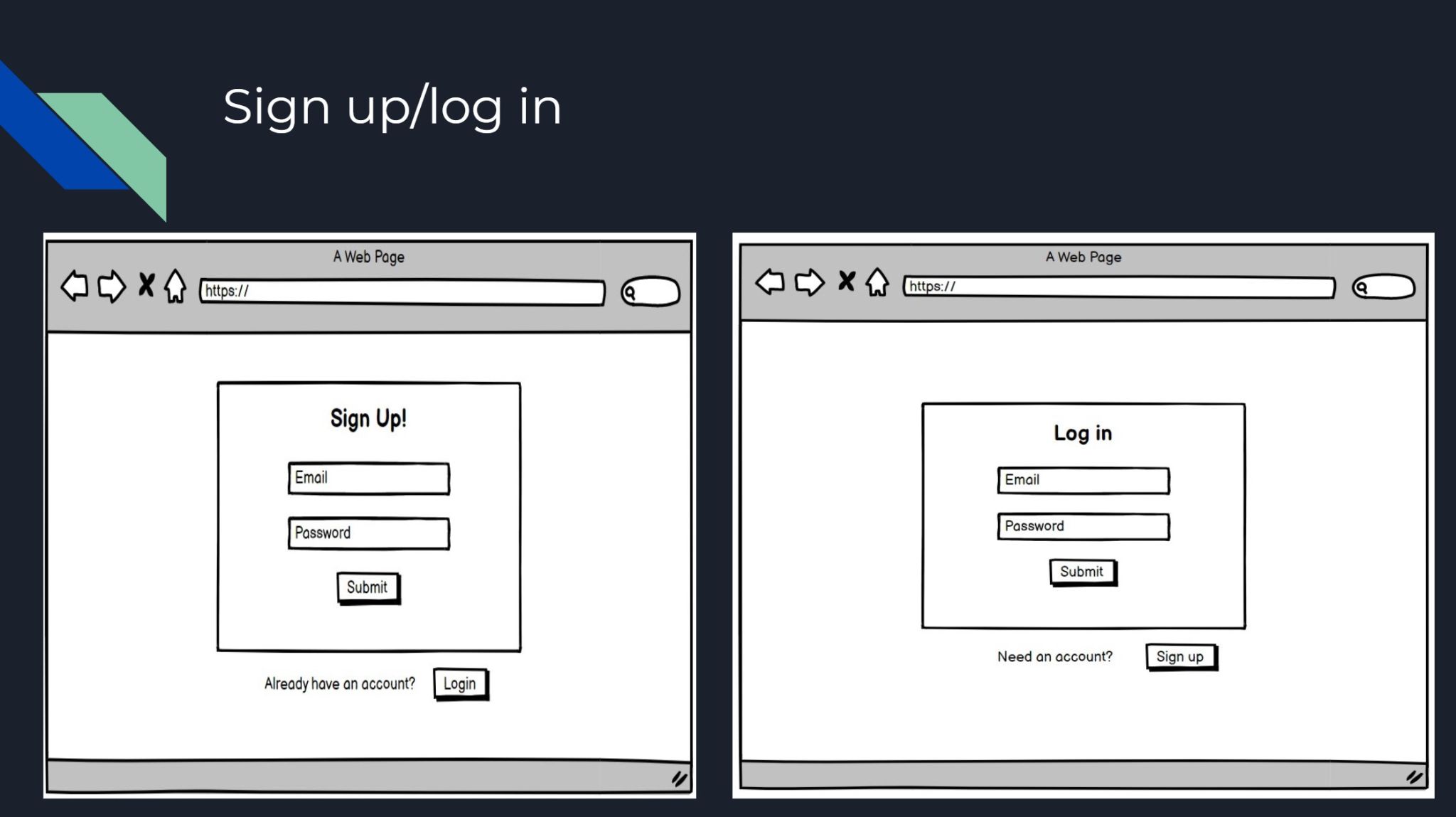
N/A

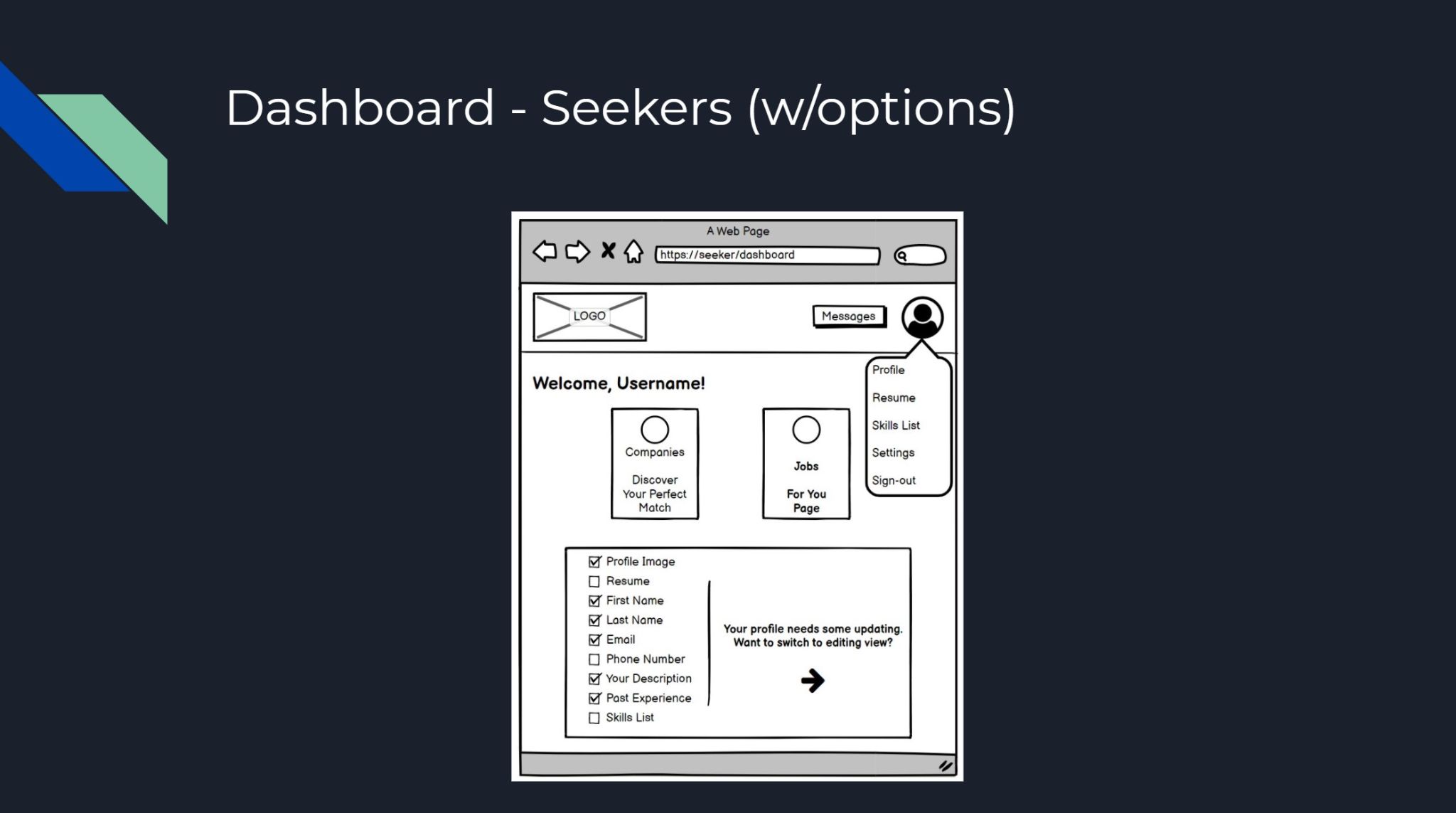
# **System Performance**

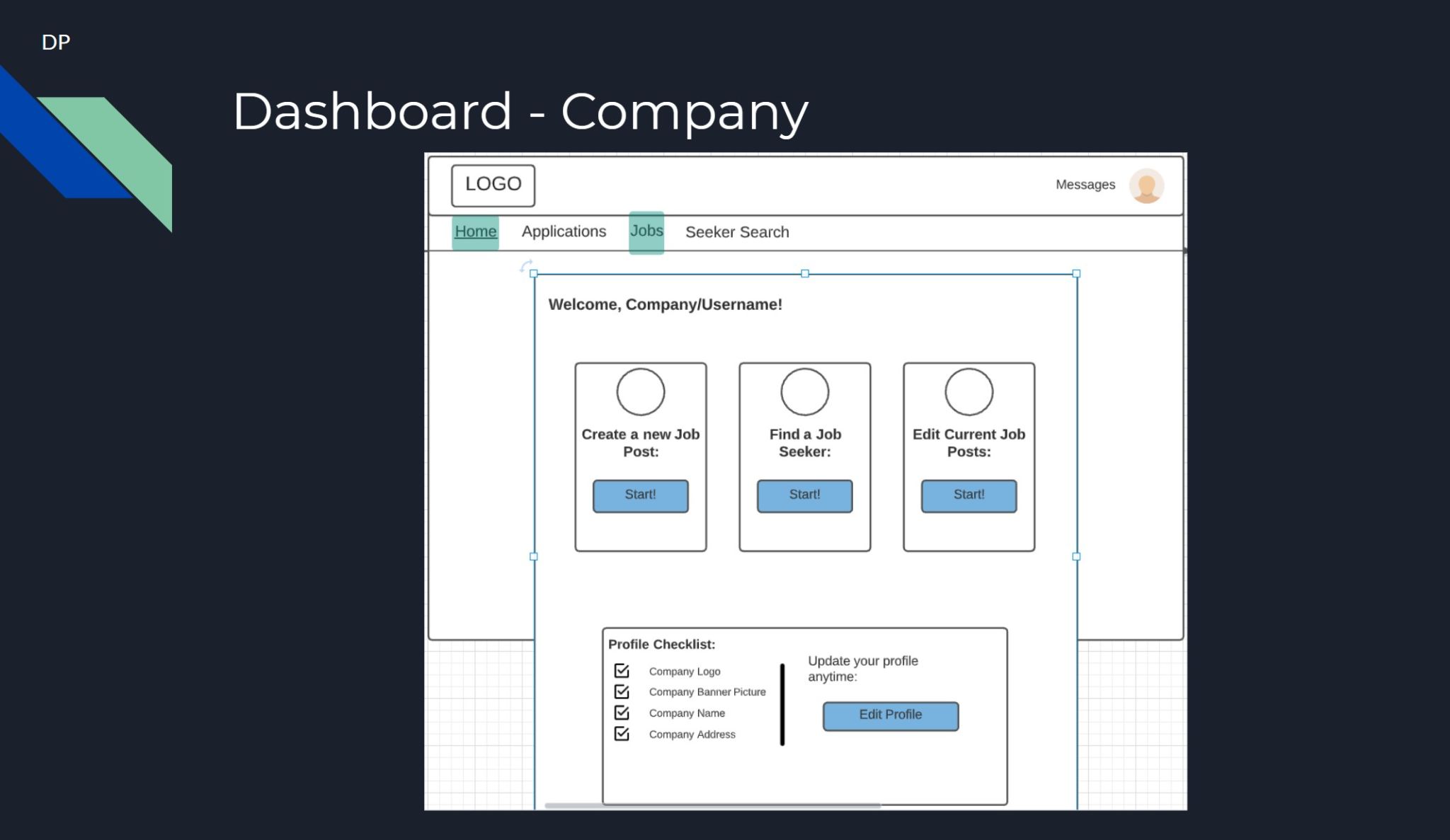
N/A

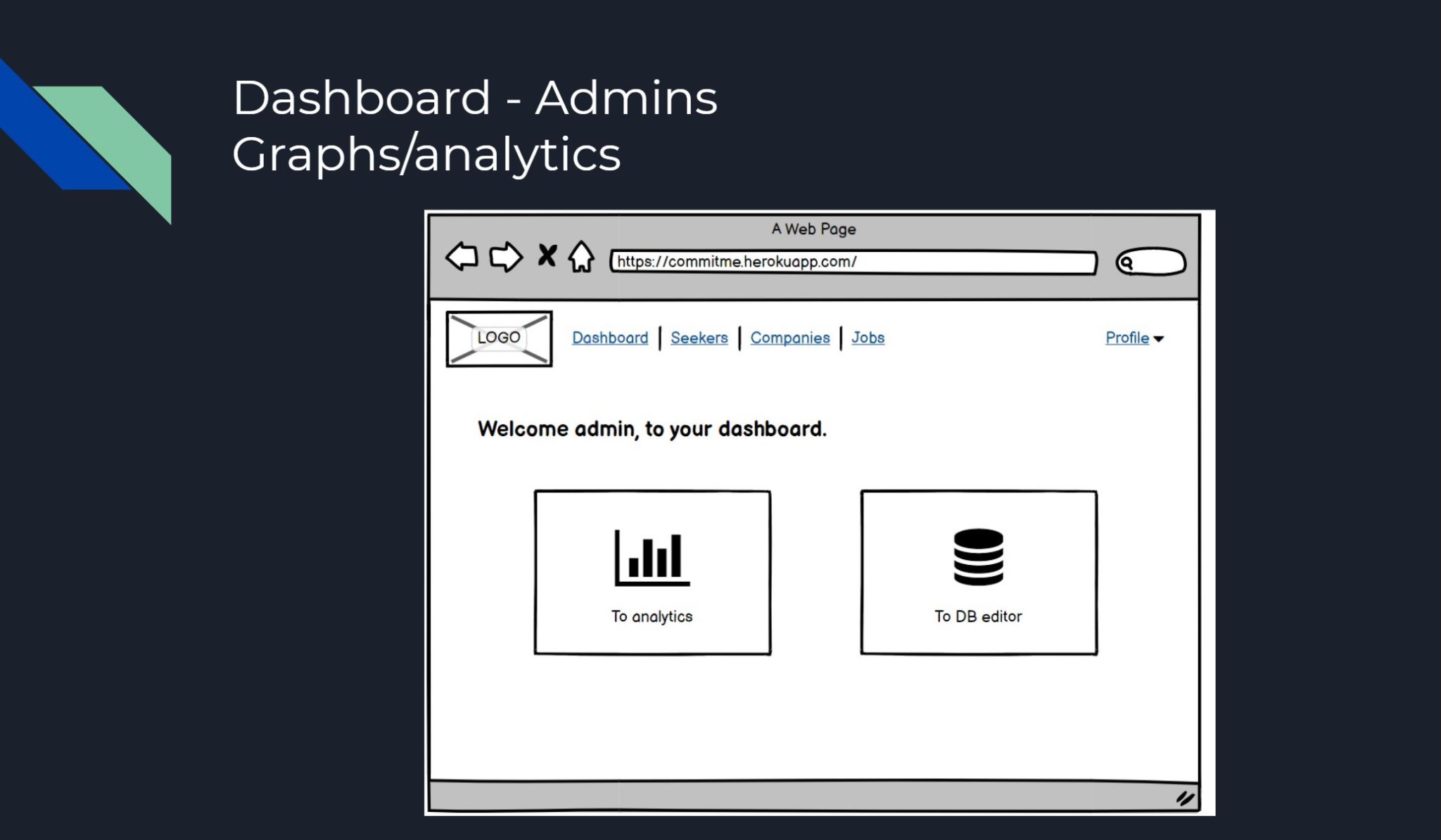
**WireFrames**

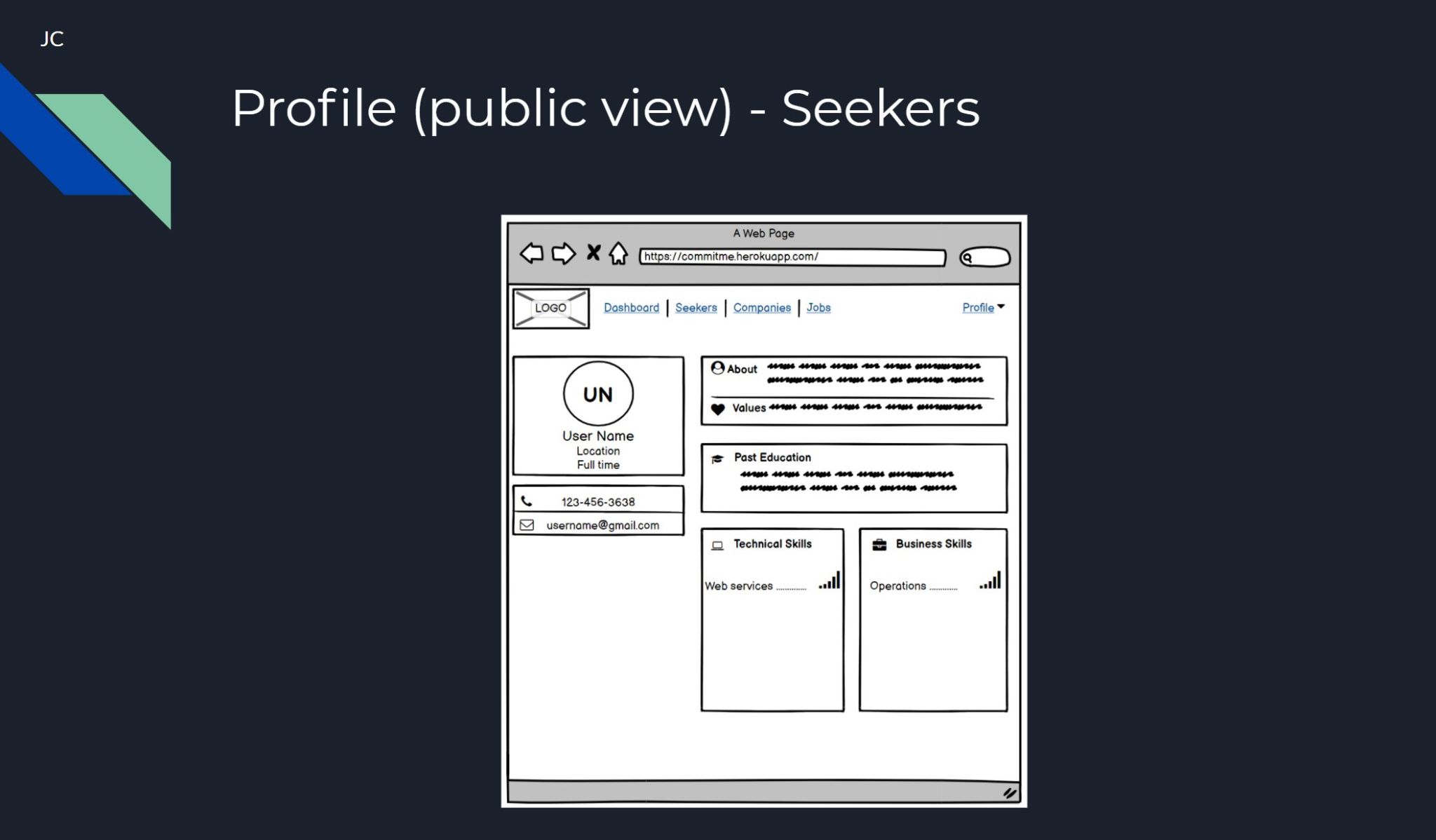
****

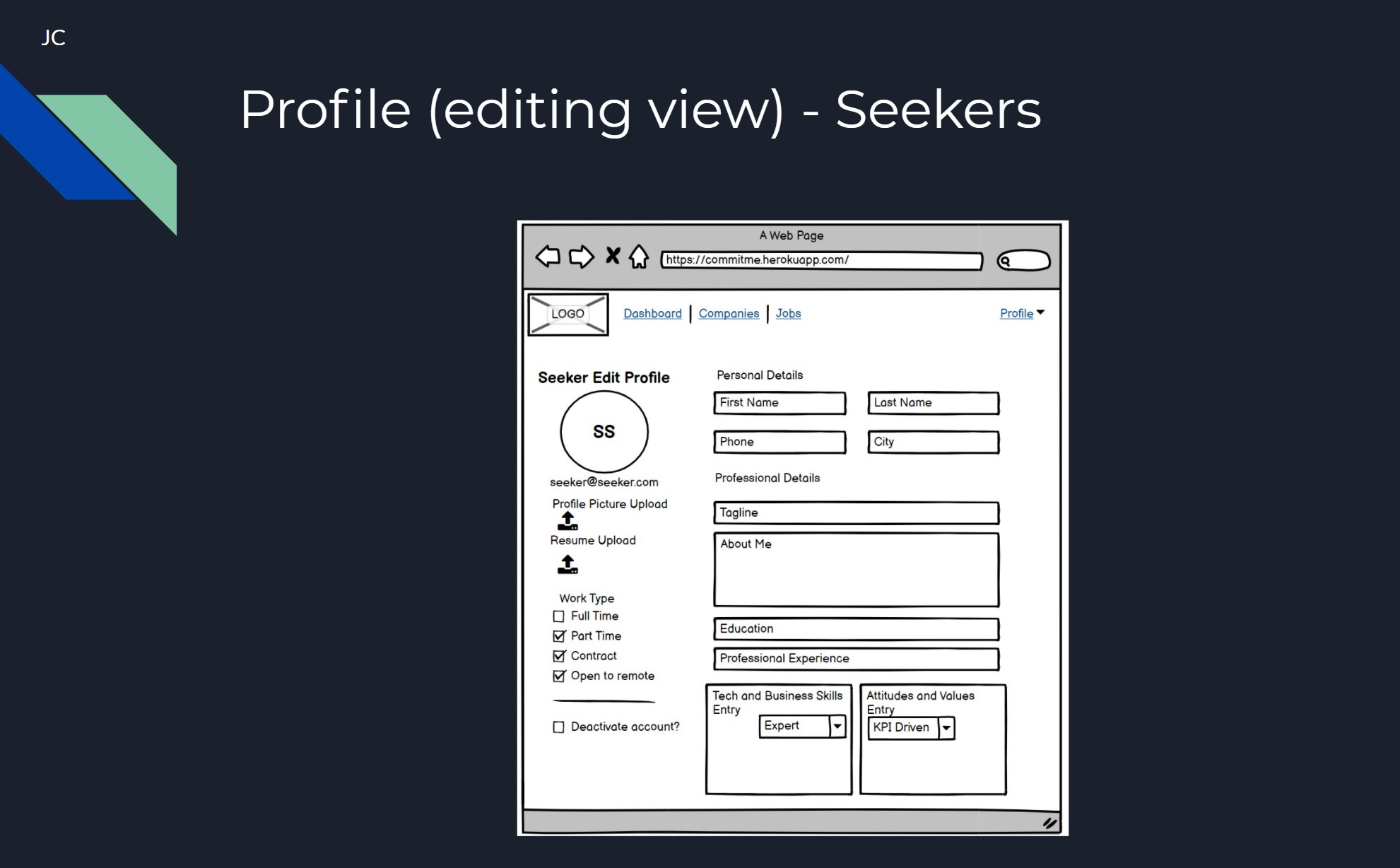
****

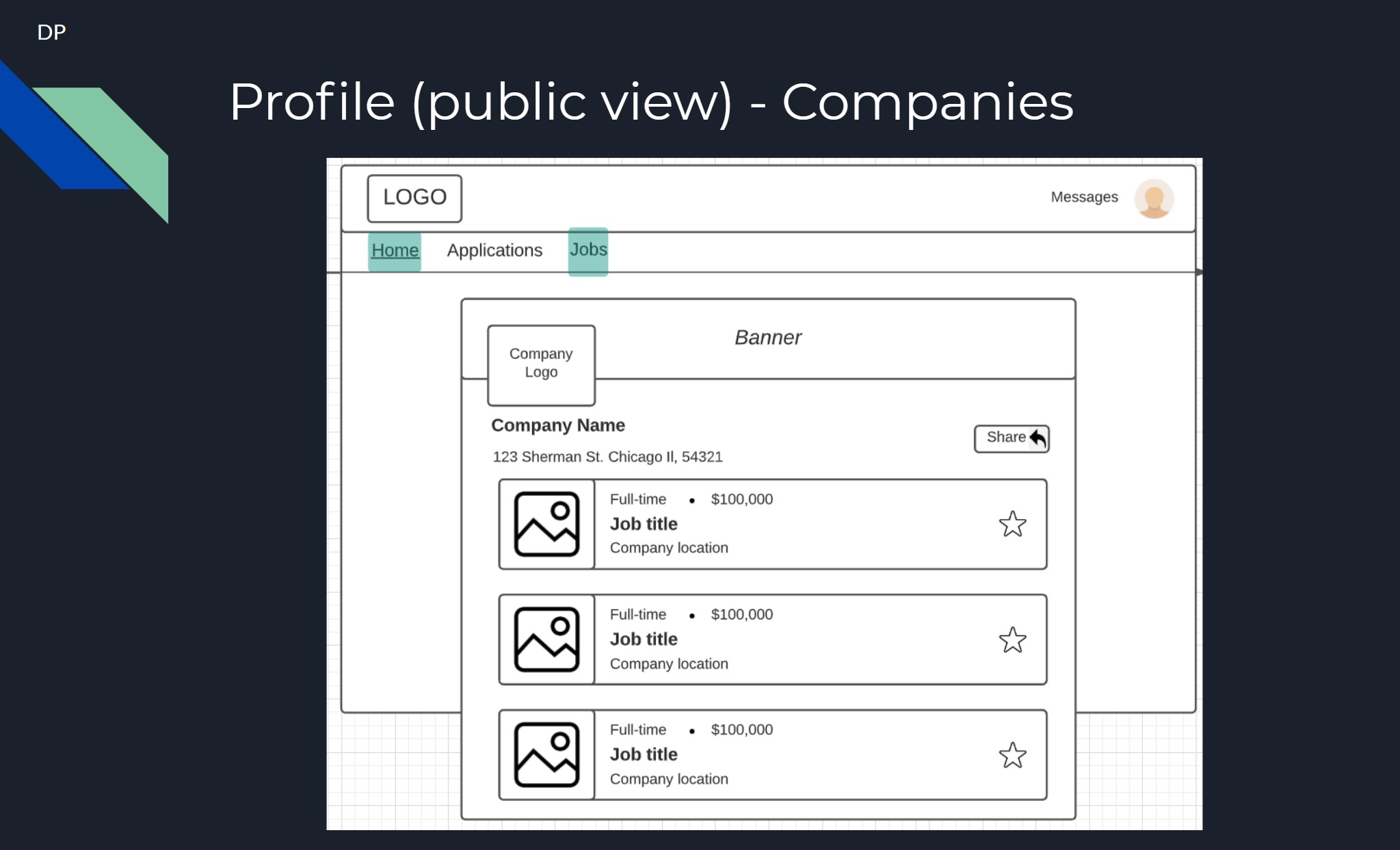
****

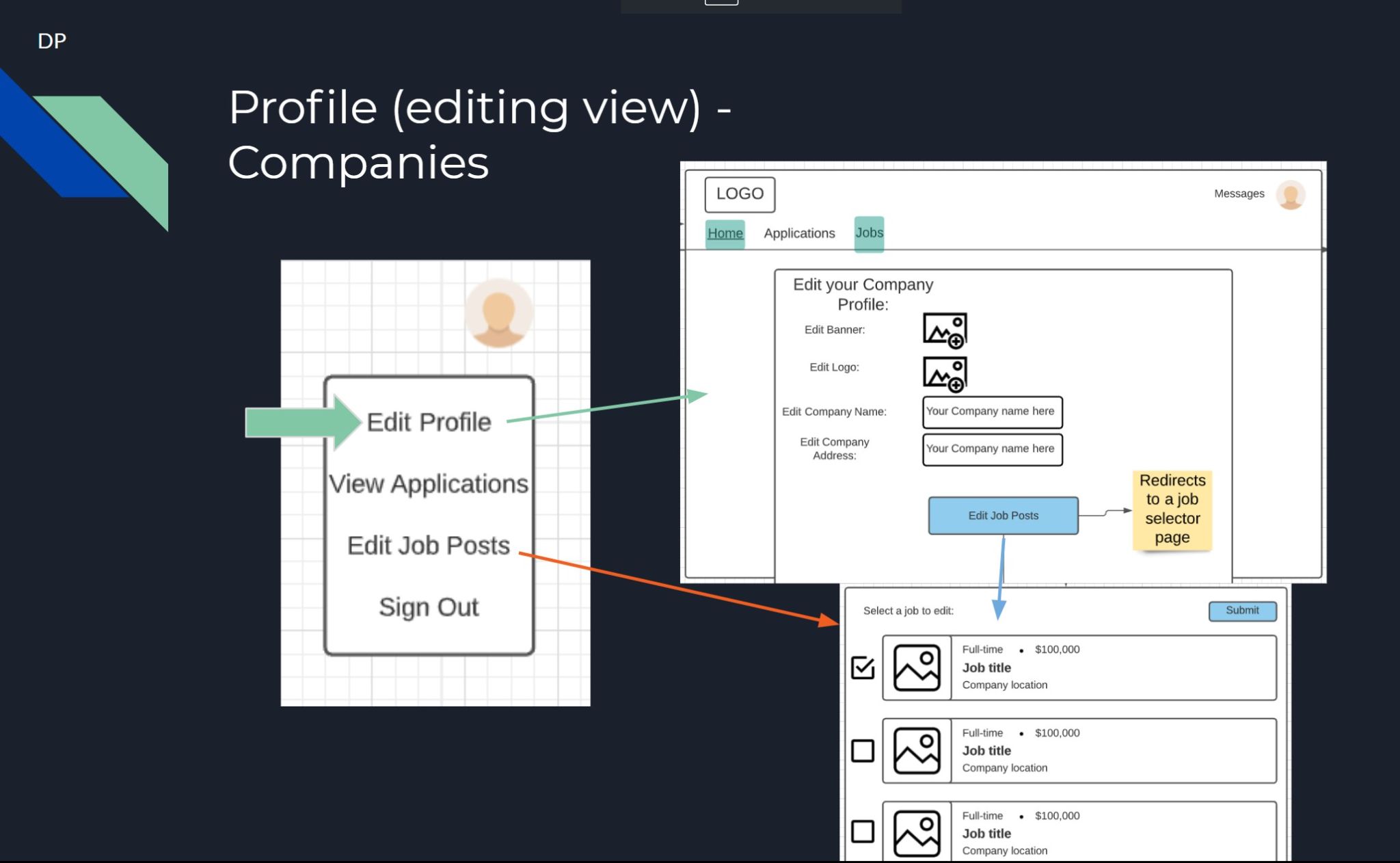
****

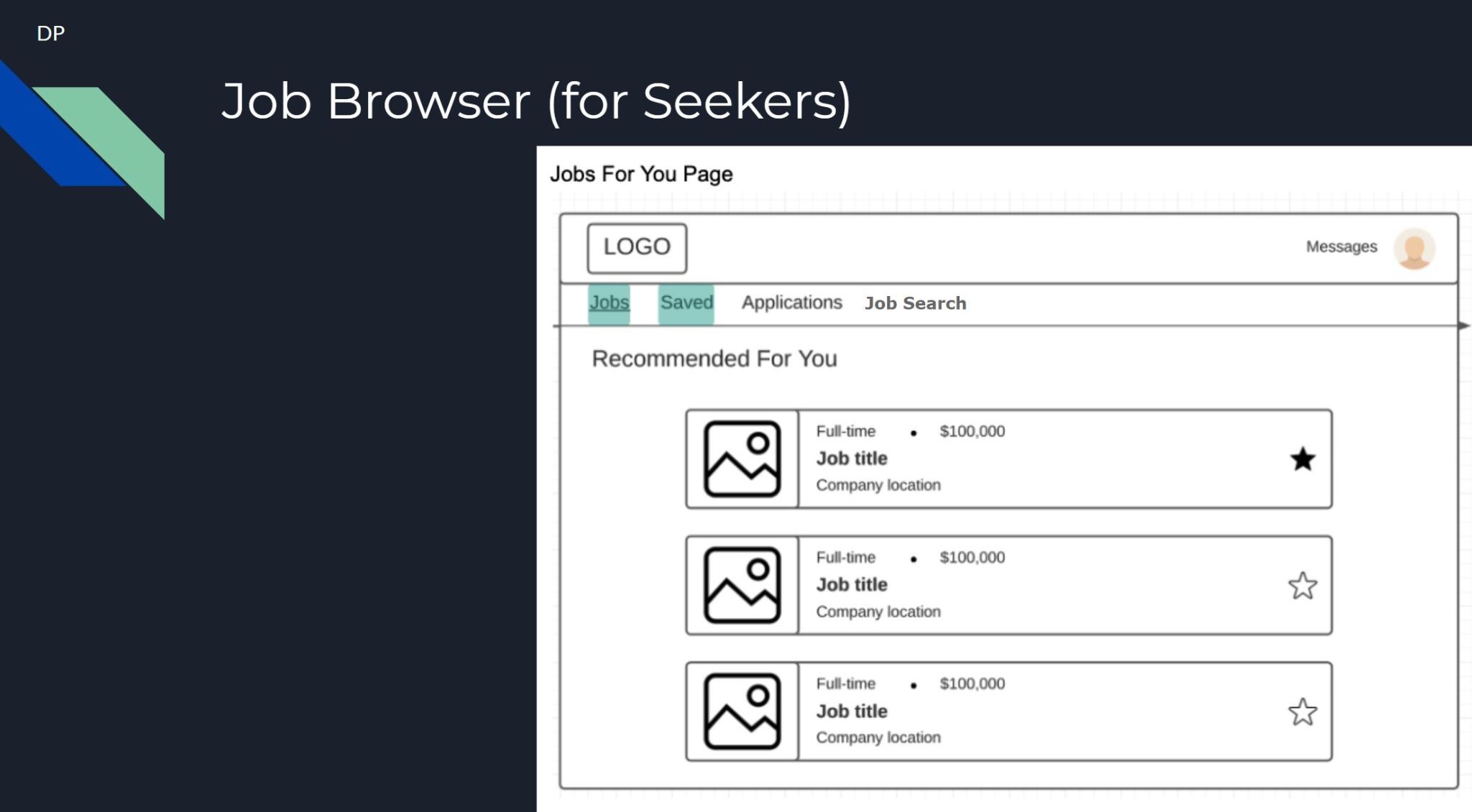
****

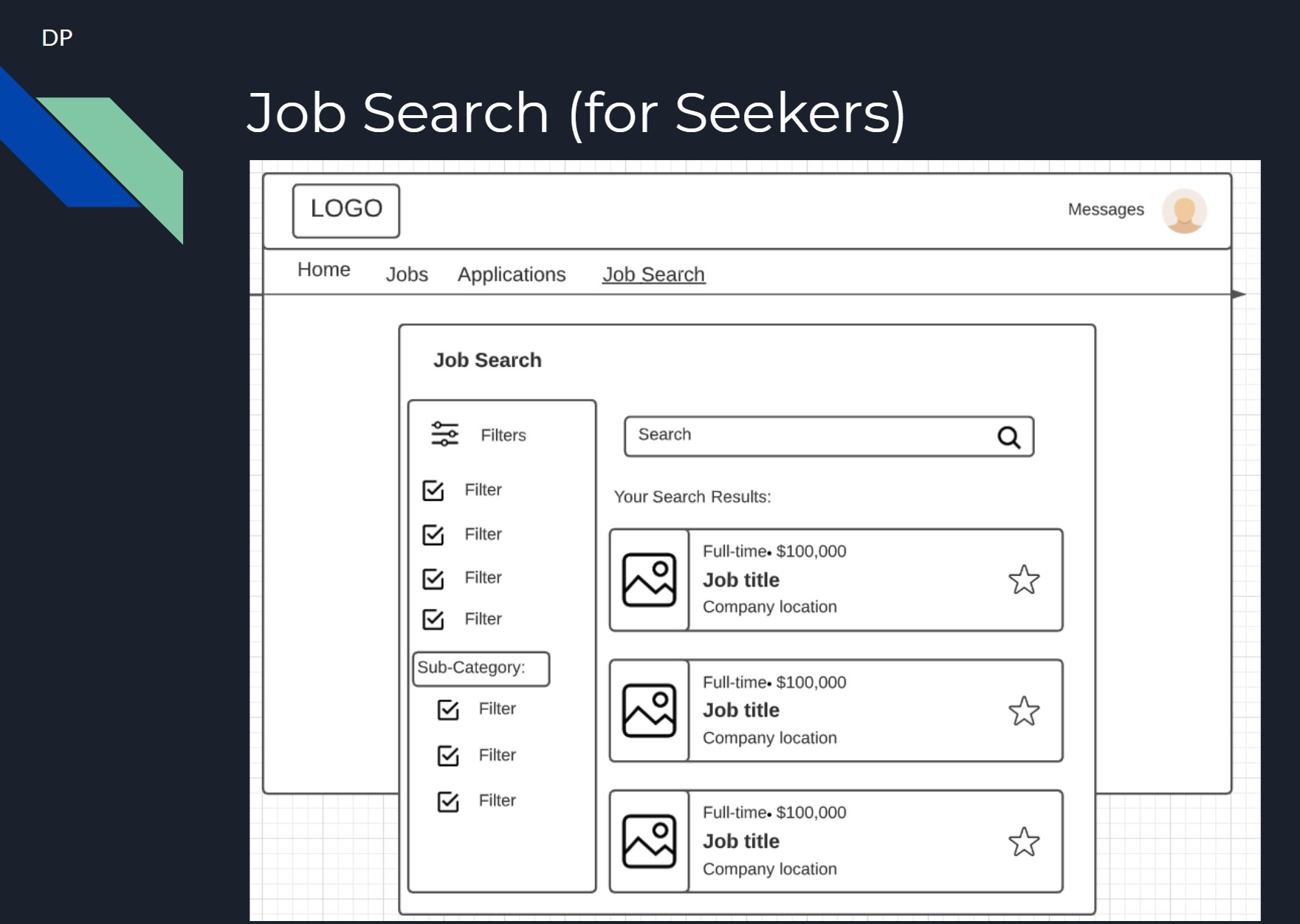
****

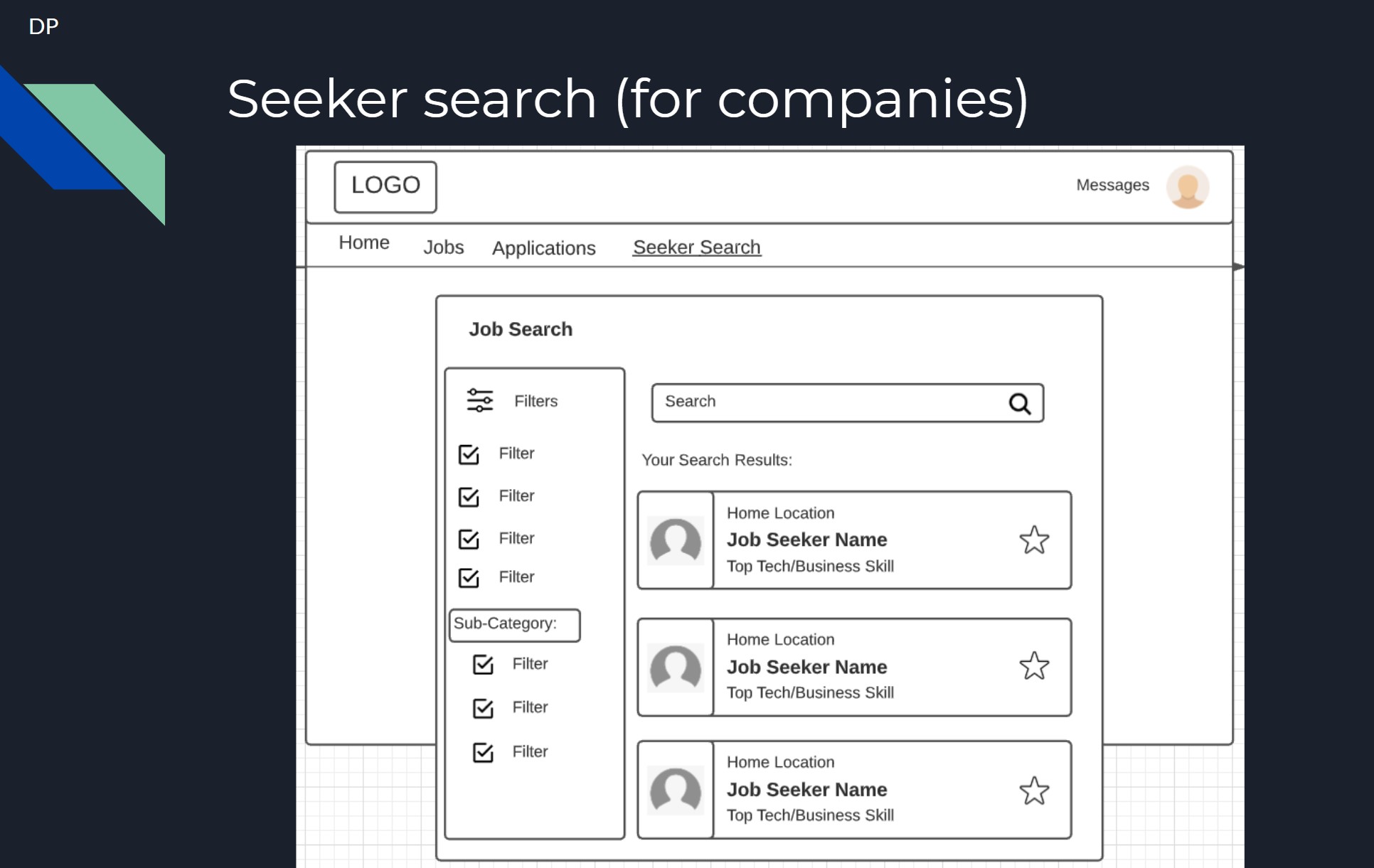
****

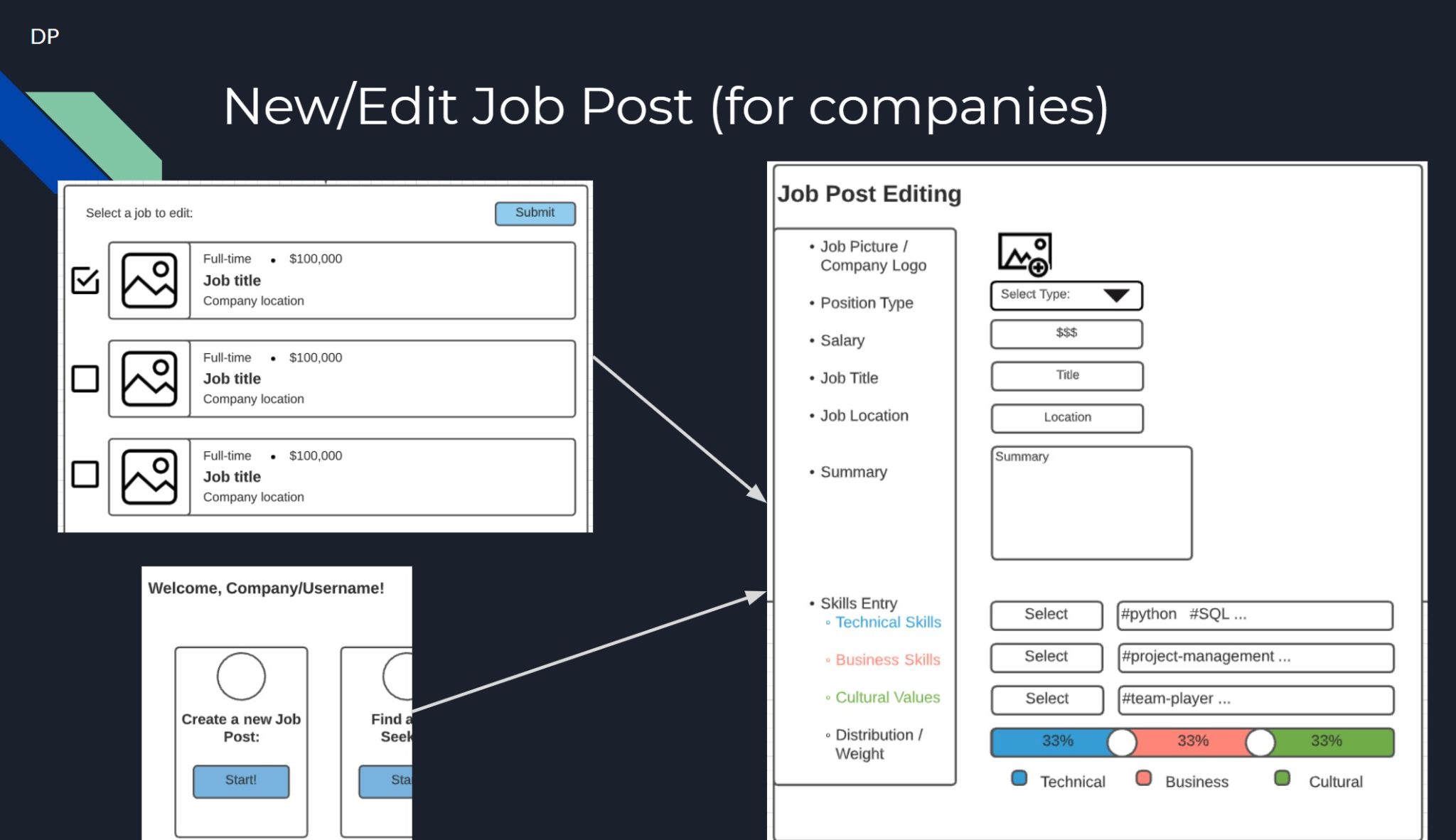
****

****

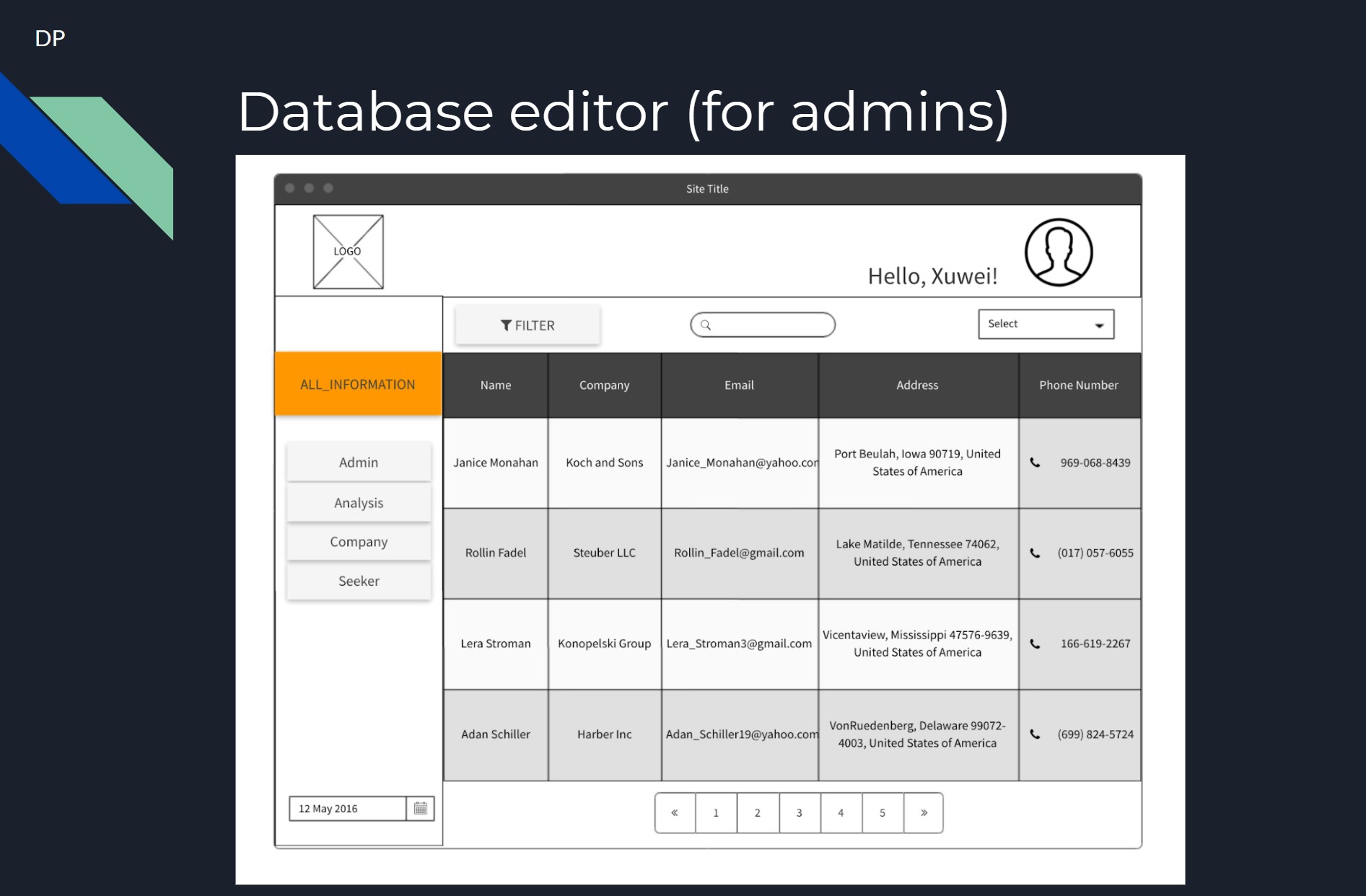
****

****

****

****

****

****

**Any test design plan and results**

# **Test Plan Overview**

## **PROJECT OBJECTIVES**

* Create a job site tool that utilizes an algorithm to connect companies and businesses with specific candidates that have the necessary skills and techniques.
* Create a job site tool that connects candidates with specific jobs that fulfil their wants and needs.
* Create a job site tool that will have different functionality based on what type of user you are.

## **SYSTEM DESCRIPTION**

This software project intends to create an IT human resources job site tool. The site created will allow job candidates to enter descriptive data regarding skills and interests, as well as, adding an official resume. Companies and businesses who use the site should be able to enter descriptive data and post jobs including necessary candidate skills. Additionally, this site will use this data to serve customized and relevant job listings to candidates.

## **PLAN OBJECTIVES**

The objective of this Project Test Plan is to develop a strategy to help us ensure our product functions as intended and to help plan on how to find vulnerabilities in our code. As a team we will either have dedicated testers for particular features, or we will pair program such that two people could alternate between writing code and testing each other's code. In terms of test methodologies, we would like to utilize an array of functional and nonfunctional testing strategies to ensure the application meets the business requirements and operational aspects. Phase one will be to make a prioritized list of system aspects that need to be tested. Phase two will be to actually develop the test cases. Phase three will be to set up a test environment. Phase four will be to execute the tests. The test environment will be somewhere where we can simulate the product running on different machines and under a variety of circumstances that might impact its functionality. The planned project testing may change depending on what features are updated, added, or removed. It may also change due to updated business plans.

## **REFERENCES**

The sources of information used to develop this plan include the team 3 business user requirements document and the team 3 test strategy document.

## **OUTSTANDING ISSUES, ASSUMPTIONS, RISKS AND CONTINGENCIES**

As of this document revision, there are currently no issues or problems relevant to testing. However, possible issues and or risks that might arise are:

Hosting Issues

* As we are relying on a service to host our site, some issues might arise with this.
* This will affect test planning because it will cause the team to find an alternate hosting service. Ultimately affecting project timing and planning.

Interface Issues

* We will have to test our software in a variety of browsers to ensure no issues arise.
* This will affect test planning because it will cause the team to change certain parts of the software to become compatible with various interfaces. Ultimately becoming an issue of project timing and planning by delaying the schedule.

Communication Issues

* With most large projects, communication between developers and testers can be difficult. For this reason, team 3 plans on discussing frequently and planning ahead to mitigate any possible risks associated with communication.
* This will affect test planning by creating project inconsistencies if certain project aspects are not communicated effectively.

# **TEST SCOPE**

## **FEATURES TO BE TESTED**

Features that will be tested:

* Algorithm portion of software
  + Candidates finding job matches
  + Companies finding candidate matches
* Basic site functionality
  + Ability to smoothly create an account
  + Ability to smoothly login and log out
  + Company/business ability to post, edit, or close jobs
  + Candidate ability to upload resume and skills list
  + Database portion
  + Ability to edit, add, or delete records

Successful performance of tested features will allow the team to evaluate what additionally needs to be added or removed. Unsuccessful performance of tested features will allow the team to evaluate what needs to be edited and changed.

## **FEATURES NOT TO BE TESTED**

Not applicable. All features of the site will be tested as soon as they are available for testing.

# **TEST METHODOLOGIES**

## **TESTING APPROACH**

The following is a list of the major group features or feature combinations to be tested. Additionally, this list outlines the activities or type of testing that is required. Moreover, it is important to mention that the in-house staff/ team 3 should and will perform initial testing.

* Basic Site Functionality
  + A usability test should be performed for the listed basic site functionality features
* Algorithm Portion
  + Once this feature is applied to site, an additional usability test should be performed
* Database Portion
  + Once a proper database is created, being able to add, edit, or delete records will be tested.

## **TEST DATA**

The sources of test data will be available and utilized once the test cases document is provided and created for this software project. Test data will ensure adequacy of testing since it is used for verification purposes.

## **TEST DOCUMENTS**

The team 3 test strategy document, team 3 use case specification, and the team 3 system design document will be created throughout the project lifecycle that will further define the system and any relevant material associated with testing.

# **REQUIREMENTS VALIDATION**

We will make a suite of tests for each requirement and make sure they pass those tests. Individuals responsible for testing will then provide the test results to the rest of the team to follow up with any issues or recommendations.

## **CONTROL PROCEDURES**

Test results will be documented individually by the software project testers. These test results should outline and record any issues and or approval of the software. Furthermore, the recording of this information will then be shared with the rest of the team for possible resolution or approval.

# **TEST PHASES**

The following list outlines the testing phases:

Prototype

* Once a prototype of the site is provided, team members will together share feedback

Implementation

* Once the software system is created and implemented for testing, individuals who will perform testing will start by testing the basic site functionality, database usability, algorithm performance, and then overall usability of the site.

System Acceptance

* Once testers approve of the site and find no errors the software will be finalized

## **DEFINITION**

Unit testing will be used to ensure that individual components of the code are free of errors or bugs. Integration testing will be used to test larger units of the application, such as the UI or database. System testing will be used to blackbox specific requirements. Acceptance testing will be used to test the system as a whole and make sure all units function properly together. Prototype testing will be used to determine some of the application’s limits.

## **PARTICIPANTS**

All developers a part of the project team will share responsibility to write tests for their code where relevant.

## **SOURCES OF DATA**

The data will be a model of possible users and use cases for our application.

## **ENTRANCE AND EXIT CRITERIA**

Test results will be evaluated based on their overall usability and functionality. Feedback that is in approval of the software as a whole will be the exit criteria that must be met.

## **REQUIREMENTS**

Those requirements listed in the team 3 requirements document will be validated in the testing phase.

## **WORK PRODUCTS**

Not applicable

## **TEST COMPLETION ACCEPTANCE**

The test is to be considered complete once the testers have approved the site and find no flaws and or errors that need to be changed.

# **TEST ENVIRONMENT**

## **HARDWARE**

Individuals who will work on testing will need a device that connects to the internet and can display web pages.

## **SOFTWARE**

Individuals who will work on testing should have access to a variety of web browsers, operating systems, and reliable internet connectivity. Testing libraries built into Python will be used.

## **LOCATION**

All testing done for this software system will be done remotely.

## **STAFFING AND TRAINING**

**Identify any staffing or training needs.**

The following are all the members associated with this software project (staffing)

* Arturo Granados
* David Palacios
* Jessica Chavoya
* Raven Khaya
* Tyler Wolfe-Adam
* Xuwei Lin
* Matthew Plicinski

At the moment, no training is required.

# **SCHEDULE**

Some functional tests will be written during development to ensure that the code works properly. However most testing will be performed near the completion of each large unit. For example, the UI will be tested for usability upon near-completion of the UI. The same goes for the Backend processes and Database.

# **APPROVALS AND DISTRIBUTION**

Approvers:

The approval process will be in the form of a pull request approval. This means that anytime someone wants to contribute their code to the main branch of our project, a different team member will have to read through the code and test it to make sure it functions as expected. Then the approver can accept the pull request and the code will be added to the main project branch.

# 

# **Test Strategy Overview**

# **1 INTRODUCTION**

The objectives of this document are:

* Communicate the scope and phases of testing
* Communicate the components that will be tested
* Communicate the dependencies between testing phases and other projects
* Communicate how the testing processes will be completed and managed

# **2 TEST OBJECTIVES**

# Verify database gets updated when a new user signs up

* Verify database retains log in details until deconstructed
* Ensure a single build works across multiple devices
* Verify recommendation algorithm correctly displays applicable jobs
* Ensure displays change depending on what role user is signed in as

# **3 IDENTIFY TEST TYPES**

| **Type of test** | **Definition** |
| --- | --- |
| Unit Testing | Testing conducted to verify the implementation of the design for one software element (e.g., unit, module) |
| Acceptance Testing | Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system |
| Performance Testing | Performed to confirm that the system meets performance goals such as turnaround times, maximum delays, peak performance, etc. |
| Data Conversion and Load Testing | Performed to verify the correctness of automated or manual conversions and/or loads of data in preparation for implementing the new system |

# **4 SCOPE OF TESTING**

* Ensure users can find a job based on their skills
* Ensure employers can see users with desirable skills
* Ensure different information is displayed based on sign in role
* Focus heavily on making sure data can be retrieved from the database and displayed to the users

# **5 TEST PREPARATION AND EXECUTION PROCESS**

## **5.1 TEST PREPARATION**

Preparation of software test cases is the most important step. A test case is a description that guides engineers to test execution, software engineering refers to requirement documents like prototype diagram, and outline design to perform Ctest to help prove software functionality or find software defects

1. TEST REQUIREMENT ANALYSIS:

| **Actors** | **Requirement** |
| --- | --- |
| Employer | Post Job, Create Profile, Browse Candidate |
| User | Browse Job Listings, Create Resume, Profile |
| Admain | Access Database, Delete\Inserting\Updating Accounts |

1. Process analysis:
   1. Test environment preparation: ***Python Unit Test***
   2. Performance test case design: ***Equivalence, Boundary, Cause and effect diagram***
   3. Performance test monitoring program design: ***SQL Database***
   4. Performance test scripts: ***Text Editor***
2. Test Cases:

| **User** | **Employer** | **Admain** |
| --- | --- | --- |
| *Minimum or Maximum of User input.* | *Creating its profile and Posting job title tag* | *Adding/ Inserting/ Updating/ Deleting Account* |
| *Tag function: make sure the user perspective finds the target job title listings.* | *Acceptance of maximum number of candidates to the job.* | *Distinguish Company account and employee account.* |
| *User adding favorite company* | *Filter out the employee who does not meet requirements.* | *Null value tests* |
| etc... | etc... | etc... |

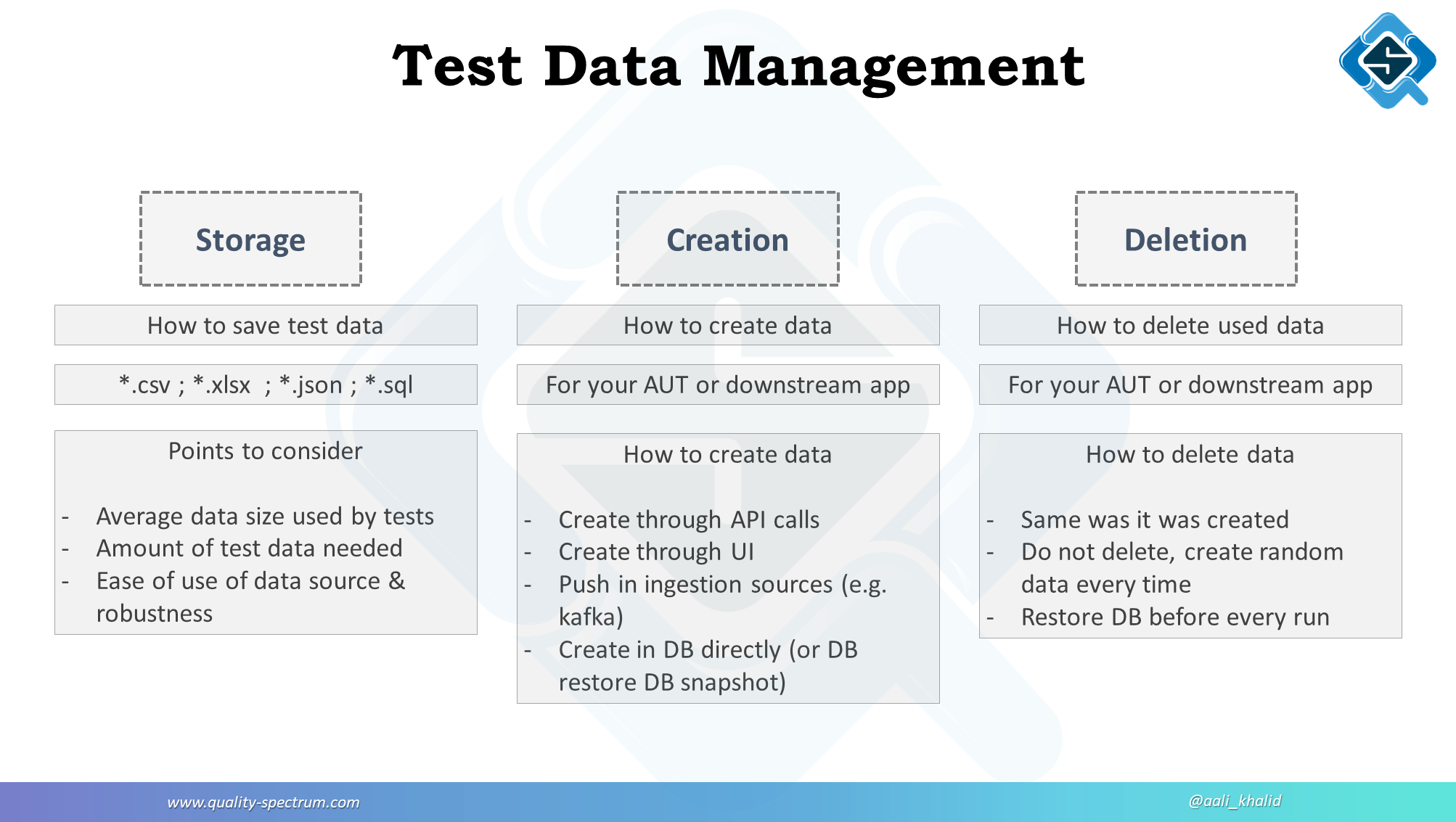
1. Self Inspections:
   1. Make sure all the test cases are included.
   2. Test cases are executed in order of priority.
   3. Commenting unclear coding parts.

## **5.2 TEST EXECUTION**

Setting up a test environment and using many testing tools to help engineers test all possible scenarios. If the test prediction is successful, enter the system test directly. If the prediction fails, cooperate with our monitoring tools to analyze the software performance and defects, and track the occurrence of bugs based on the source of the data, and record the address in the Excel table.

# **6 TEST DATA MANAGEMENT**

After all the defects are closed, then the regression test is carried out to release the version, and a detailed evaluation of the entire test process and version quality is made to confirm whether it can be launched.



* The data obtained from the test will be stored in the SQL Database in the form of SQL file, and it is expected to use **Total Test cases**=[amount of employee uses cases\*Amount of employer uses case\*Amount of Admains uses cases], and it will be automatically generated for each test case Serial number, easy to track.
* As part of the test, the software engineer will fill in the data through the UI page of the software together with the design team, and the obtained data will appear in the background. Similarly, engineers will use AI to automatically generate data to test the software coverage, and the test data will also be stored in the SQL Database.
* Operational readiness test – a copy of system/user acceptance test data could be used for the operational readiness test. Since the focus of the test is on operational procedures, a low number of transactions will be required and data integrity is not critical.
* Generate HTML report file to judge the coverage of all functions.

**Any Use Cases**

# USE CASE SPECIFICATION TEMPLATE

# 

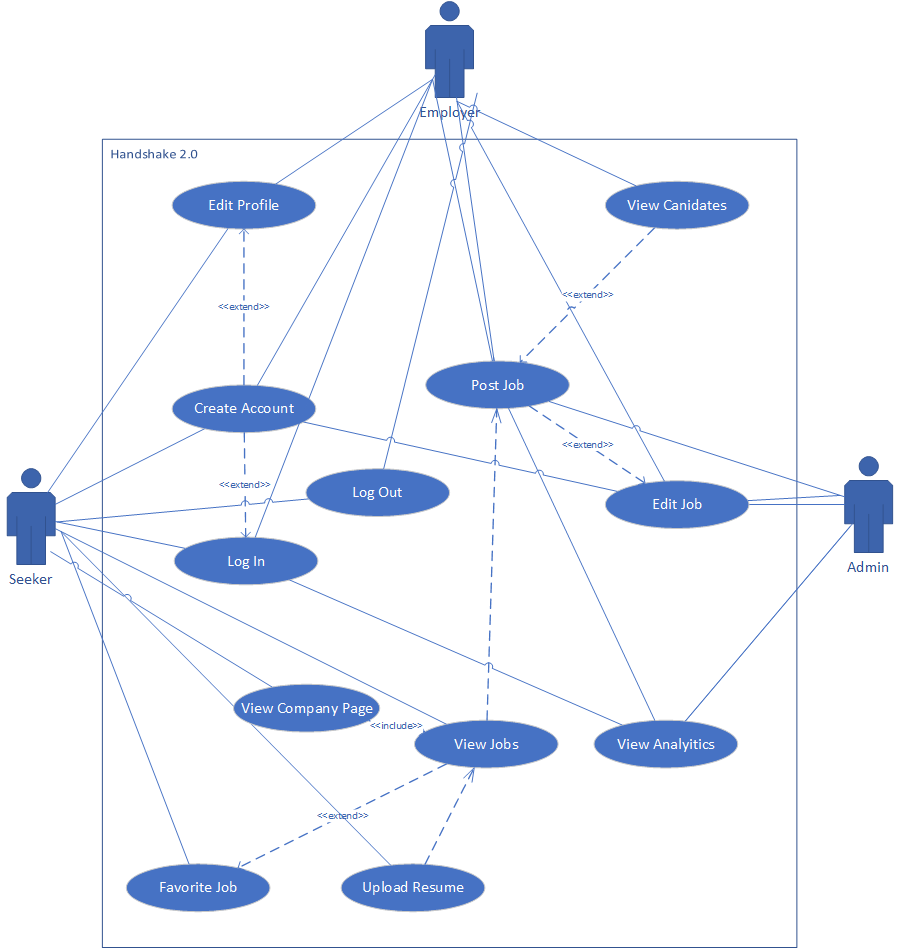
# **ACTORS, GOALS AND USE CASE BRIEFS**

| **Actors** | **Task-level Goal** | **Priority** | **Brief** |
| --- | --- | --- | --- |
| job seeker, company, admin | Log-in/Register | High | The user will need to submit their log-in credentials and they must match the credentials in our profile database. If they are a new user, they will be entered into the system |
| job seeker, company | Edit profile | High | Users should be able to access their profile information and edit their skills, resumes, and applications as needed. |
| job seeker | View relevant jobs | High | Job seekers will request to view the curated list of jobs for them. |
| company | View candidate matches | High | Companies will request a list of job seekers who fulfill their job requirements. |
| company, admins | post jobs/edit jobs | High | Companies will see the option to create and post a job post that can then be recommended by our algorithm to our seekers. |
| admins | view analytics | High | Companies and admins should be able to see statistics and logs in order to maintain the system |
| Job seekers | submit their resume to apply to jobs | High | Seekers should be able to submit their applications to open job posts and the companies would see this as a reflection of incoming applications. |
| job seeker, companies, admins | log out | High | all users should be able to log out in order to keep their account secure |
| job seeker | view a company discovery page | Medium | seekers should be able to view a company overview and description |
| companies | send a message | Medium | companies should be able to message seekers through some medium |

## 

## 

## **USE CASE DIAGRAM**



**Project Code Module List**

“app”: Where the project is stored. It has modules for global constants and database models. It consists of four subpackages:

1. “api”: Provides a public endpoint for querying/updating the database regarding users and for the searching/matching algorithms
2. “auth”: Provides the HTTP routes and web forms related to authentication (registration, log-in)
3. “main”: Provides the HTTP routes and web forms related to the main sources of interaction (job posting, searching, editing/viewing profiles)
4. “errors”: Provides the handlers for assorted web errors (e.g., 404, 500)

**Version and Development History**

Deployments:

* v01: May 01 at 1:21 PM
* v02: May 01 at 1:21 PM
* v03: May 01 at 1:21 PM
* v04: May 01 at 1:21 PM
* v05: May 01 at 1:55 PM
* v06: May 26 at 11:34 AM
* v07: May 26 at 11:53 AM
* v08: Jun 03 at 3:58 PM
* v09: Jun 03 at 4:10 PM
* v10: Jun 03 at 4:59 PM
* v11: Jun 04 at 2:12 PM
* v12: Jun 04 at 2:24 PM
* v13: Jun 07 at 1:27 PM
* v14: Jun 08 at 4:35 PM
* v15: Jun 08 at 7:23 PM
* v16: Jun 08 at 9:11 PM
* v17: Jun 09 at 12:02 AM
* v18: Jun 09 at 12:05 AM
* v19: Jun 09 at 12:14 AM
* v20: Jun 09 at 11:45 AM

Full network history can be seen here: <https://github.com/shirtandtieler/Job-Website-Project/network>

**User Manual**

# **INTRODUCTION**

## **INTRODUCTION AND PURPOSE**

The purpose of this Systems Administration Manual is to create a user manual that is short, intuitive, and that describes the features/functions for end-users. The system to which this document applies is to team 3’s software project, Commit Me. This system was created as a solution for a new HR job site tool that can be accessed from any type of computer through a simple web browser.

## **PROJECT REFERENCES**

Available systems documentation include:

* Business user requirements document
* Test plan and test strategy documents
* Use case specification and template document
* System design document
* System deployment document

These documents will then be combined to create the project development manual.

## **GLOSSARY**

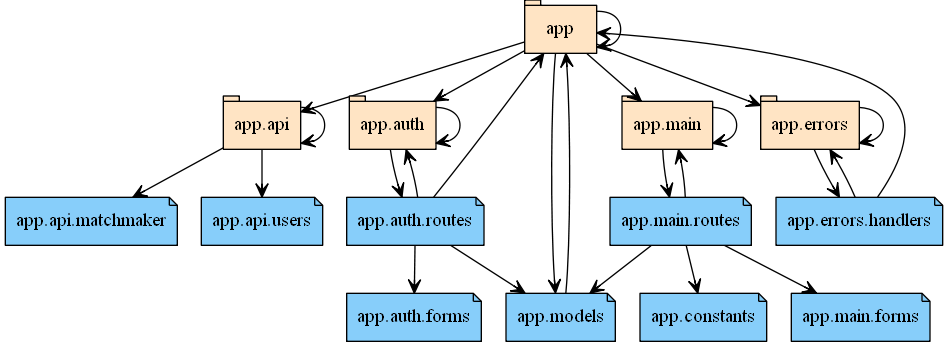
* “User” - Any visitor accessing the website
* “Seeker” - A job seeker or candidate
* “Company” - A user representing a company account
* “Admin” - An administrator account

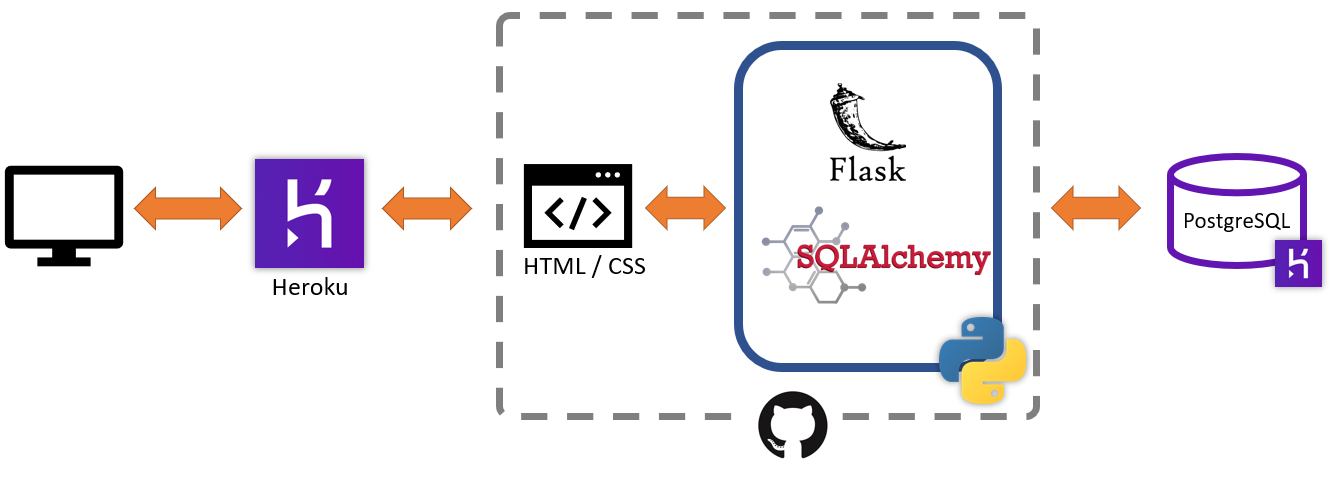
# **SYSTEM OVERVIEW**

## **SYSTEM APPLICATION**

The purpose of this system is to create an IT human resources job site tool. The site created will allow job candidates to enter descriptive data regarding skills and interests, as well as, adding an official resume. Companies and businesses who use the site should be able to enter descriptive data and post jobs including necessary candidate skills. Additionally, this site will use collected data to provide customized and relevant job listings to candidates and customized and relevant candidates to companies.

## **SYSTEM ORGANIZATION**





## **INFORMATION INVENTORY**

This section provides information about data files and the databases that are produced or referenced by the system.

### **Resource Inventory**

A PostgreSQL database is referenced by the website and hosted on Heroku.

### **Report Inventory**

N/A

## **PROCESSING OVERVIEW**

Operational restrictions: Each page on the website can only be accessed if you are an appropriate user for that particular example. For example, a seeker or company cannot view the admin dashboard page.

No other systems are communicated with.

## **COMMUNICATIONS OVERVIEW**

Other than the ability to find company or job seeker contact information, the system does not have any communication functions in it.

## **SECURITY**

Seeker and Company information can only be accessed by registered users. The website is accessed with HTTPS connection to encrypt data sent to or from the website.

# **SITE PROFILE(S)**

This section contains information pertaining to the site(s) where the application is running. That information includes the information contained in the subsequent sections.

## **SITE LOCATION(S)**

<https://commitme.herokuapp.com>

## **PRIMARY SITE**

Contacts for the site designated as primary include all team three personnel.

* Jessica Chavoya
  + jchavoya@depaul.edu
* Arturo Granados
  + agranad3@depaul.edu
* Raven Khaya
  + rkhaya@depaul.edu
* Xuwei Lin
  + xlin19@depaul.edu
* David Palacios
  + dpalaci6@depaul.edu
* Matthew Plicinski
  + mplicins@depaul.edu
* Tyler Wolfe-Adam
  + twolfead@depaul.edu

# **SYSTEMS ADMINISTRATION**

## **USER AND GROUP ACCOUNTS**

### **Adding/Deleting Users**

Seekers can create an account (with both a login and password identification) via the sign-up page. A link is provided on the homepage. To delete user login and password account, seekers need to contact a system administrator. They can deactivate their profile (not showing up on search results) through the editing view page of their personal account.

Companies can create an account (with both a login and password identification) via the sign-up page. A link is provided on-site the homepage. To delete user login and password account, companies need to contact a system administrator.

### **Setting User Permissions**

N/A – Only the site administrator can have access to certain restricted files.

### **Adding/Deleting User Groups**

N/A

### **Setting User Roles/Responsibilities**

Admins are created manually by the site administrators. The seekers & companies pick their role upon creating their account on the website. The logic behind what each user type has access to is built into the product.

## **SERVER ADMINISTRATION**

This section describes procedures to set up servers, including naming conventions and standards.

### **Creating Directories**

“app”: Where the main code is stored. It has modules for global constants and database models. It consists of the following subdirectories:

1. “api”: Provides a public endpoint for querying/updating the database regarding users and for the searching/matching algorithms
2. “auth”: Provides the HTTP routes and web forms related to authentication (registration, log-in)
3. “main”: Provides the HTTP routes and web forms related to the main sources of interaction (job posting, searching, editing/viewing profiles)
4. “errors”: Provides the handlers for assorted web errors (e.g., 404, 500)
5. “static”: Contains typical static files for a website, including style files, scripting files, and images.
6. “templates”: Contains all HTML templates used to render the website.

The process of creating more directories requires modifying the source code and re-deploying the website. Any directories created while deployed (e.g., within the console on Heroku or connected remotely from a command prompt) will be automatically removed by Heroku within 24 hours.

### **Building Drive Mappings**

N/A

## **SYSTEM BACKUP PROCEDURES**

This section describes procedures for regularly scheduled backups of the entire network, including program and data storage, and the creation and storage of backup logs.

### **Maintenance Schedule (Daily, Weekly)**

As the website is currently in a non-production state, it does not have any backups scheduled. If desired, the database could be scheduled to be backed up on a daily basis automatically. At its current plan, up to 2 backups can be saved on Heroku’s servers. This process could also be setup to be done “manually” through a script.

Further information can be found at: <https://devcenter.heroku.com/articles/heroku-postgres-backups>

### **Off-Site Storage Procedures**

The website is stored entirely off-site, on Heroku’s servers.

### **Maintaining Backup Log**

System logs can be accessed in multiple ways, via a CLI or the Heroku dashboard. More information can be found here: <https://devcenter.heroku.com/articles/logging>

## **PRINTER SUPPORT**

### **Maintenance**

N/A

### **PRINT JOBS**

N/A

## **SYSTEM MAINTENANCE**

### **Monitoring Performance and System Activity**

Performance & Activity of the application is automatically monitored through Heroku, the cloud service we are using the application.

### **Installing Programs and Operating System Updates**

N/A

### **Maintaining Audit Records of System Operation**

N/A

### **Maintenance Reports**

N/A

## **SECURITY PROCEDURES**

This section describes the process for obtaining identifications (IDs) and passwords. It includes information concerning network access and confidentiality requirements.

### **Issuing IDs and Passwords**

N/A

### **License Agreements**

N/A

## **NETWORK MAINTENANCE**

This section describes procedures to maintain and monitor the data communications network.

### **LAN Design**

N/A

### **Communications Equipment**

N/A

## **INVENTORY MANAGEMENT**

### **Maintaining Hardware and Software Configurations**

N/A

### **Maintaining Floor Plans**

N/A

### **Installing Software/Hardware (New, Upgrades)**

N/A

### **Maintaining Lists of Serial Numbers**

N/A

### **Maintain Property Inventory**

N/A

## **TRAINING BACKUP ADMINISTRATOR**

This section describes how to train a backup administrator.

### **End-User Support - Procedures for Support and Contract Information**

Matthew Plicinski - [mplicinski@gmail.com](mailto:mplicinski@gmail.com)

Xuwei Lin - [lxw824178965@gmail.com](mailto:lxw824178965@gmail.com)

### **Escalation Procedures**

N/A

## **DOCUMENTATION**

### **Troubleshooting Issues**

Team three should be made aware of any troubleshooting issues and conduct any troubleshooting activities. Therefore, the team will document this information on their own.

## **DATABASE MAINTENANCE**

### **Database User/Group Access**

Tyler is the owner of the Heroku Application Hosting account and would grant access to any other team members.

### **Adding/Deleting Users to the Database**

Those who are granted a designated admin account manage users in the database. They are responsible for all database maintenance of the application.

### **Setting User Permissions for Database**

N/A (User Permissions are built into the website)

### **Adding/Deleting Groups for Database**

N/A

### **Re-indexing Database**

After database changes have been made in a local development environment, the database hosted on the server will need to be updated. To do this, the developer should export their instance of the database to a “.dump” file using PostgreSQL’s “pg\_dump” command. The following command shows an example (the developer should replace the username and database name based on their environment).

pg\_dump -Fc --no-acl --no-owner -h localhost -U user jobsite > mydb.dump

The developer should then upload this dump file to a hosted data environment (e.g., Amazon S3 or a site such as tmpfiles.org). Finally, the developer should tell Heroku to pull in the database from the URL provided in the data environment. This requires the developer to have collaborator permissions on the Heroku page. For example:

heroku pg:backups:restore https://tmpfiles.org/dl/..../mydb.dump DATABASE\_URL --app=commitme --confirm commitme

### **Packing/Compressing Database**

N/A

### **Data Entry/Modification/ Deletion**

Anyone with collaborator access on the Heroku app can access the database. Admin accounts on the site can interface with the database directly through the website.

### **Database Reporting**

N/A

### **Database Backup and Restore**

Anyone with collaborator access on the Heroku app can perform a manual backup. Scheduled backups can be performed as well.

## **APPLICATION MAINTENANCE**

### **Application User/Group Access**

Tyler the owner of the Heroku account hosting the application would grant access to any other members.

### **Adding/Deleting Application users**

Administrators should be responsible for adding or deleting users. However, administrator accounts must be created by the developer.

### **Setting User Application Permissions**

N/A

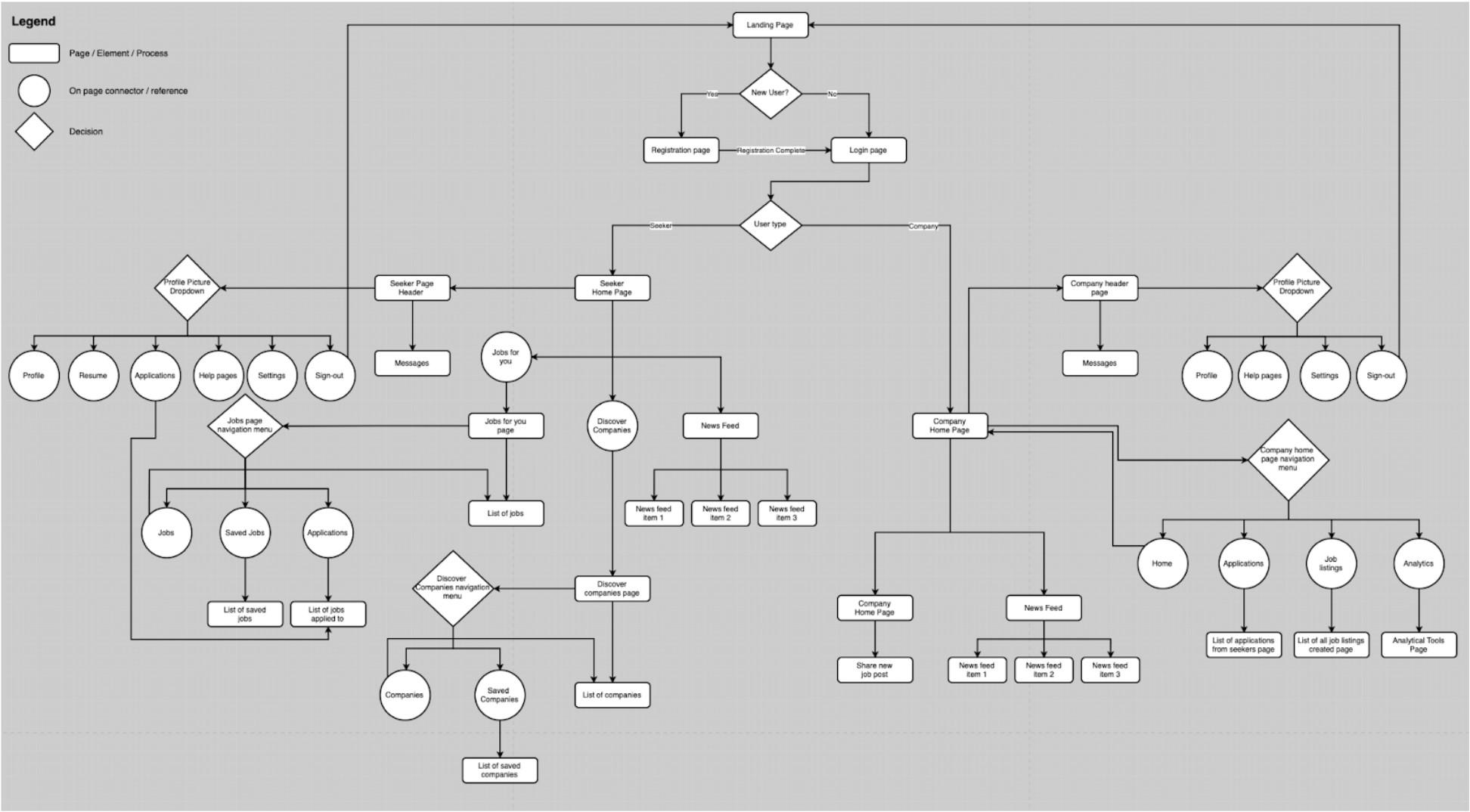
### **Adding/Deleting Application Groups**

N/A

### **Procedures to Start and Stop the Application**

Tyler is the owner of the Heroku account which is hosting the application so he would be directly responsible for starting and stopping the application. If we had to stop the application we would notify users of a scheduled maintenance time and do any problem fixing then. Ideally, this maintenance time would be done during a time of the day when user activity is on average low.

### **Application Flow Chart**



### **Description of Major Program or Sub-program Modules**

“app”: Where the main code is stored. It has modules for global constants and database models. It consists of the following subdirectories:

1. “api”: Provides a public endpoint for querying/updating the database regarding users and for the searching/matching algorithms
2. “auth”: Provides the HTTP routes and web forms related to authentication (registration, log-in)
3. “main”: Provides the HTTP routes and web forms related to the main sources of interaction (job posting, searching, editing/viewing profiles)
4. “errors”: Provides the handlers for assorted web errors (e.g., 404, 500)

**Weekly Team Log**

Logs track what the team discussed with the professor during the weekly meetings.

Team 3 Software Project

MEETING #1 03/31

horizontal line

**31 March 2021 / 6:30 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, & Jessica Chavoya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* Not applicable
  + This is the first meeting

## **New Business**

* Review course expectations and requirements.

# NOTES

* Email the professor if and when needed.
* Each group gets 25 minutes each week.
* Tool Suggestions
  + Trello
  + Google Docs
  + GitHub for code
* The goal is to build a tool for three users.
  + People looking for a job
  + People posting a job
  + Administrator
  + Users should be able to input skills and resume
  + Give users a chance to describe and list what they are good at
  + Recommendations should be provided for job seekers and companies
  + Be creative!
* Use provided New Mexico IT templates
  + Simple and easy to use
  + If something does not apply use N/A
  + Bullet points or paragraphs are okay
* **HW1 is due week 3**

# ACTION ITEMS

1. There are no old action items
2. New action items:
   1. Review project description and requirements

# Key Decisions Made

* When to contact team members and begin the project.

# Issues Identified

* Not applicable, this was the first meeting.

Team 3 Software Project

MEETING #2 04/07

horizontal line

**07 April 2021 / 6:30 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, & Jessica Chavoya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* Not applicable.

## **New Business**

* Notify the professor that we are missing our seventh team member.
* Discuss HW1 requirements and details.

# NOTES

* For the hardware, use a server for the site.
* Discussed overall ideas for the project.
* **HW1 is due next week**
  + Use provided templates.
  + Present HW1 as a slide show presentation
* Homeworks
  + Homeworks will be updated throughout the course as things change

# ACTION ITEMS

1. There are no old action items
2. New action items:
   1. Start HW1

# Key Decisions Made

* Allow professor to locate the seventh team member

# Issues Identified

* Not applicable

Team 3 Software Project

MEETING #3 04/14

horizontal line

**14 April 2021 / 6:31 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* The seventh team member has now joined.

## **New Business**

* Go over HW1 presentation

# NOTES

* Google Slide
  + Requirements → Delivered by Jessica
  + Architecture → Delivered by Tyler
  + Data Design → Delivered by Arturo
  + Hardware/Software → Delivered by Tyler
  + GUI → Delivered by Tyler since David could not attend the meeting.
* Professor Feedback
  + Keep at what we are doing
  + The team is planning properly
  + Remember to use templates for homework
    - Can place N/A in sections that do not apply
  + Remember homework input will change as the project progresses. Go back and make changes if and where needed.
* **HW2 is due next week.**

# ACTION ITEMS

1. Old action items:
   1. Contact with the seventh team member → resolved
2. New action items:
   1. Present HW1
   2. Accept feedback

# Key Decisions Made

* Progress with the project to HW2

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #4 04/21

horizontal line

**21 April 2021 / 6:32 PM - 6:53 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* There was nothing to follow up on.

## **New Business**

* Go over HW2 documents
* Ask any relevant questions

# NOTES

* Tyler presented the requirements document
  + Addressed what the team as a whole might want to do moving forward
  + Question:
    - In terms of site capacity/ site ability/ site traffic, what should the team prepare?
      * Professor Answer: Just make it work. Don’t anticipate hundreds of users, but make it available to scale up if need be. This is not the main focus of the project.
* David presented the test plan document
* Professor feedback
  + The presentation of requirements was done well
  + Focus on user requirements!
    - Business requirements will become included
      * Don’t worry too much about business requirements
  + Since this is only a ten-week course, testing can not be long
  + Now is the best time to start coding the project!
* **HW3 is due in two weeks.**

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Present HW2
   2. Accept feedback

# Key Decisions Made

* Progress with the project to HW3
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #5 04/28

horizontal line

**28 April 2021 / 6:31 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* There was nothing to follow up on.

## **New Business**

* Go over what is expected for HW3
* Go over what the team accomplished this week
* Get feedback on preliminary wireframes
* Ask any relevant questions

# NOTES

* Tyler discussed the technical aspect of the project and what the team did/accomplished this week
* David and Jessica presented preliminary wireframes to get feedback
* Professor Feedback
  + Wireframes
    - Well thought out
    - Likes the idea of separate login and sign up pages
    - Clean approach
  + HW3 Comments
    - Just pick one use case to write about
      * Describe it
      * No diagram, charts, or tables needed, just pick one to complete
    - Spend more time on coding!
    - Documents should be placed on the back burner compared to getting coding completed
* **HW3 is due next week.**

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Discuss HW3
   2. Discuss where the team should be in project development
   3. Present prelim wireframes and accept feedback

# Key Decisions Made

* Progress with the project to HW3
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #6 05/05

horizontal line

**05 May 2021 / 6:31 PM - 6:42 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* There was nothing to follow up on.

## **New Business**

* Present HW3 documents
* Accept feedback
* Discuss HW4 expectations

# NOTES

* David presented the use case document
* Tyler presented the system document
* HW4
  + Looking for wireframes only
* Professor Feedback
  + Both documents presented today have the right idea
  + The effort put in is well
  + Overall, the homework looks good, it’s just a matter of implementing it now
  + Start getting the pieces together
  + The team is on target
* **HW4 is due next week.**

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Discuss HW4 expectations
   2. Discuss where the team should be in project development
   3. Present HW3 documents and work

# Key Decisions Made

* Progress with the project to HW4
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #7 05/12

horizontal line

**12 May 2021 / 6:33 PM - 6:46 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* There was nothing to follow up on.

## **New Business**

* Present HW4 Wireframe and document
* Accept feedback

# NOTES

* Wireframes delivered by David and Jessica
* System deployment plan document delivered by Tyler
* Professor Feedback
  + Wireframes:
    - Looks good
    - Good ideas and design
    - Hopes we can implement weight value feature
* **HW5 is due in two weeks.**

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Present HW4 details
   2. Discuss where the team should be in project development

# Key Decisions Made

* Progress with the project to HW5
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #8 05/19

horizontal line

**19 May 2021 / 6:32 PM - 6:37 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* There was nothing to follow up on.

## **New Business**

* Discuss project progress
* Discuss what was accomplished this past week
* Discuss HW5 expectations

# NOTES

* Jessica and David discussed where the team is at in project development
* Question: Does HW5 have to be delivered as a video?
  + Professor answer: Yes, just so it saves time and mitigates any possible issues that might arise.
* Professor comments on HW5:
  + The complete system does not have to be done
  + Show what is complete up to that date
  + Idea is to present what the system is looking like now and what it should look like
* **HW5 is due next week.**

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Discuss where the team should be and is in project development

# Key Decisions Made

* Progress with the project to HW5
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #9 05/26

horizontal line

**26 May 2021 / 6:32 PM - 6:42 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* There was nothing to follow up on.

## **New Business**

* Present HW5
* Discuss how far along the team is with the project
  + Any other project updates

# NOTES

* David presented the video our team created for HW5
* Professor Feedback
  + Looks pretty good
  + Should be able to complete between now and the due date
  + Likes the fairly simple layout
  + For reference, look at other job sites like Linkedin
* **HW6, HW7, HW8, and HW9 are due in two weeks.**
  + HW8 can be written with bullet points
    - If something does not apply to you, N/A is acceptable.

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Discuss where the team should be and is in project development

# Key Decisions Made

* Progress with the project to complete on June 9, 2021.
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

Team 3 Software Project

MEETING #10 06/02

horizontal line

**02 June 2021 / 6:33 PM - 6:40 PM / Zoom Meeting**

# ATTENDEES

Thomas Muscarello, David Palacios, Matthew Plicinski, Xuwei Lin, Arturo Granados, Tyler Wolfe-Adam, Jessica Chavoya, & Raven Khaya

# New Items To Be Discussed

## **Last Meeting Follow-up**

* Update professor
  + Present what work has been made from the last meeting till now

## **New Business**

* Discuss how far along the team is with the project
  + Any other project updates

# NOTES

* Professor Feedback
  + N/A
* **HW6, HW7, HW8, and HW9 are due next week at 5:45 pm.**
  + HW8 can be written with bullet points
    - If something does not apply to you, N/A is acceptable.

# ACTION ITEMS

1. Old action items:
   1. Not applicable
2. New action items:
   1. Discuss where the team should be and is in project development
   2. Update professor with the projects current progress

# Key Decisions Made

* Progress with the project to complete on June 9, 2021.
* The team is progressing well, continuing with the plan of action currently in place.

# Issues Identified

* Not applicable.

**Team Work Log**

# **Information regarding team three’s work log**

The following links are utilized to represent teamwork.

* GitHub: <https://github.com/shirtandtieler/Job-Website-Project>
  + Commit log: <https://github.com/shirtandtieler/Job-Website-Project/commits/main>
* Trello: <https://trello.com/b/6An0iTcf>