

# Programing Paradigms Project Final Report:

Team Member Full Name	NetID
Katie Grace	kgrace2
Sarah Cullinan	scullin2
Sofia Dillhoff	sdillho2
Celina Ryan	cryan27

## Chosen Technology Stack

☐ Python + Django + Bootstrap

### Persistent Storage Design

We are using SQLite database to persist our data. Our database, as seen in *figure 1*, contains the following tables: CanidateProfile, RecruiterProfile, Post & Offer. These tables are connected as and participate in inheritance as a recruiter can make a post, and an offer. A candidate can view an offer and posts (from recruiters). Additionally, candidates can access posts through the offers they have received. For these reasons. all of these models are connected and are associated with eachother.

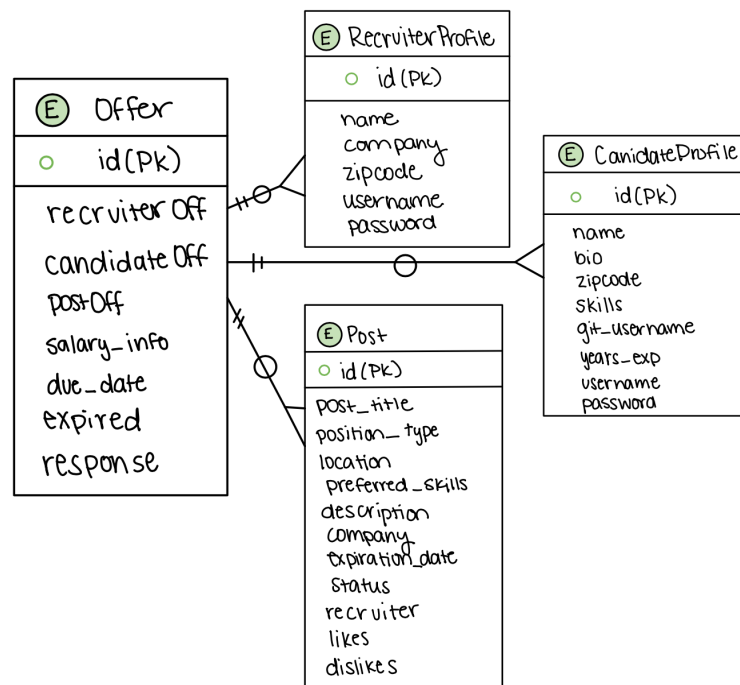


Figure 1 database schema

## Feature 1: Sign Up/ Sign In

### 1.1: Create New Candidate Profile

*Figure 2* shows a screenshot for creating a candidate profile. The user is directed to this page when they click on the “Sign up as a Candidate” link on the homepage of our TinDev website. The information requested from the candidate includes: name, bio, skills, GitHub username, years of experience, username and password. This allows the candidate to create an account with proficient information for Recruiters actively looking for candidates.

# Welcome to TinDev

Login

Don't have an account? Sign up below!

Sign up as Candidate

Sign up as Recruiter

*Figure 2 Screenshot for feature 1.1 showing the homepage of our TinDev website*

## Sign up as a candidate

Name:

Bio:

Zipcode:

Skills:

Git username:

Years exp:

Username:

Password:

Create

*Figure 2 Screenshot for feature 1.1 showing the information a candidate inputs when creating a TinDev account*

The website is designed so that no two users have the same username. If so, the candidate will get the following error notification shown in *figure 3* below.

# Sign up as a candidate

Error: Username is already in use.

[Try Again](#)

Figure 3 screenshot for feature 1.1 showing a sign up error notification when username is already in use

## 1.2: Create new Recruiter Profile

Figure 4 shows a screenshot for creating a recruiter profile. The user is directed to this page when they click on the “Sign up as a Recruiter” link on the homepage of our TinDev website. The information requested from the candidate includes: name, company, zip code, username and password. This allows the recruiter to create an account with proficient information that is valuable for candidates actively looking for jobs/internships etc.

# Sign up as a recruiter

Name:

Company:

Zipcode:

Username:

Password:

Figure 4 screenshot for feature 1.2 showing the information a recruiter inputs when creating a TinDev account

The website is designed so that no two users have the same username. If so, the recruiter will get the following error notification shown in figure 5 below.

# Sign up as a recruiter

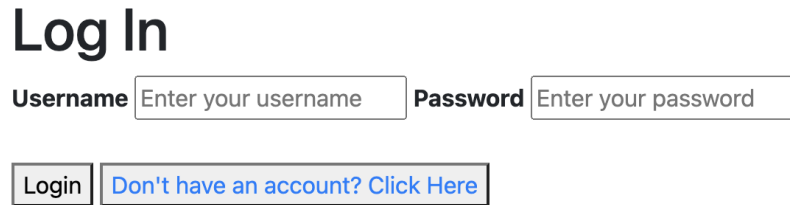
Error: Username is already in use.

[Try Again](#)

Figure 5 screenshot for feature 1.2 showing a sign up error notification when username is already in use

### 1.3: Log-in

Figure 6 shows a screenshot for our login page. This allows users who have already created an account to login and view job postings and interact with other users on our TinDev website.



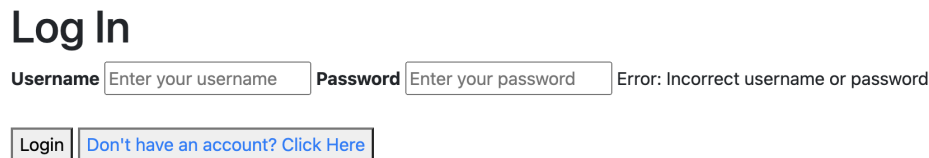
**Log In**

Username  Password

[Don't have an account? Click Here](#)

Figure 6 screenshot for feature 1.3 showing the login page

If an user attempts to login with the incorrect username or password, they will receive the error notification seen in figure 7.



**Log In**

Username  Password  Error: Incorrect username or password

[Don't have an account? Click Here](#)

Figure 7 screenshot for feature 1.3 showing login error notification when they have entered an incorrect username/password

### 1.4: Log-out

Figure 8 shows screenshots of both the Recruiter and Candidate Dashboards (which are separate links). Users are taken to this webpage when they log-in. Once at the dashboard page, users can click the “Logout” button and they will be redirected to the login page as seen in figure 6.



Recruiter Dashboard	Candidate Dashboard
<input type="button" value="Create Post"/> <input type="button" value="View all posts"/> <input type="button" value="Logout"/>	<input type="button" value="View All Posts"/> <input type="button" value="View Interested Jobs"/> <input type="button" value="View Offers"/> <input type="button" value="Logout"/>

Figure 8 screenshots for feature 1.4 showing the logout buttons on the dashboards

## Feature 2: Recruiter's Dashboard

### 2.1: View all posts

Figure 10 shows a screenshot of the Job Postings page, which the user is redirected to when they click the “View all posts” button on the recruiter dashboard home page as seen in figure 9 below. Note that a recruiter can only see the jobs that they have posted, and not the jobs other recruiters have posted. For example, let's say that Grigorii works for the Cincinnati Reds. This means that he can only see the jobs that he posted and not what other recruiters have posted as seen in figure 10.

## Recruiter Dashboard



Figure 9 screenshot of the Recruiter Dashboard home page

Recruiter Dashboard Create Post All Posts Logout

### Job Postings

Filter Posts:

- **Video & Technology Specialist**, Cincinnati Reds, Cincinnati, OH
- **Account Executive, Retention & Sales**, Cincinnati Reds, Cincinnati, OH

Figure 10 screenshot for feature 2.1 showing Job Postings page

If Grigorii tries to “View all Posts” and he has not made any posts, he will be redirected to the page seen in figure 11.

No job postings.



Figure 11 screenshot of the Recruiter Dashboard home page

Finally, if a recruiter has made posts, they can filter them with the dropdown menu shown in figure 12.

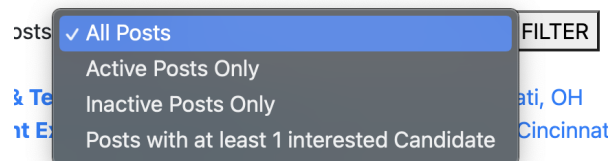
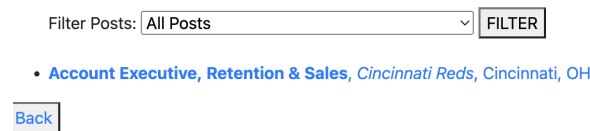


Figure 12 screenshot to filter the Job Postings in feature 2.1

In this example, if Grigorri selected to filter his posts to only see active posts, he would only see the post for the Account Executive (and not the video and technology specialist) since it is the only active post as seen in *figure 13* below. If no posts fall under the filter Grigorri chose, then he would be redirected to the page in *figure 11*.

## Job Postings



Filter Posts:

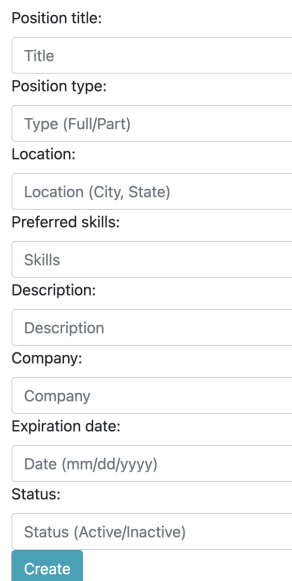
- Account Executive, Retention & Sales, Cincinnati Reds, Cincinnati, OH

Figure 13 screenshot of filtered job posting for feature 2.1

## 2.2: Creating a post

From the recruiter dashboard, when a recruiter clicks on the “Create Post” button, they are led to this form as seen in *figure 14*. Recruiters then input the necessary information to create a post for candidates to view.

### Create a Job Post



Position title:

Position type:

Location:

Preferred skills:

Description:

Company:

Expiration date:

Status:

Figure 14 screenshot for feature 2.2 of the Create a Job Post form

If a recruiter attempts to submit a form and leaves a field blank, they receive the error notification seen in *figure 15*.



Position title:

Position type:

Location:

Preferred skills:

Description:

Company:

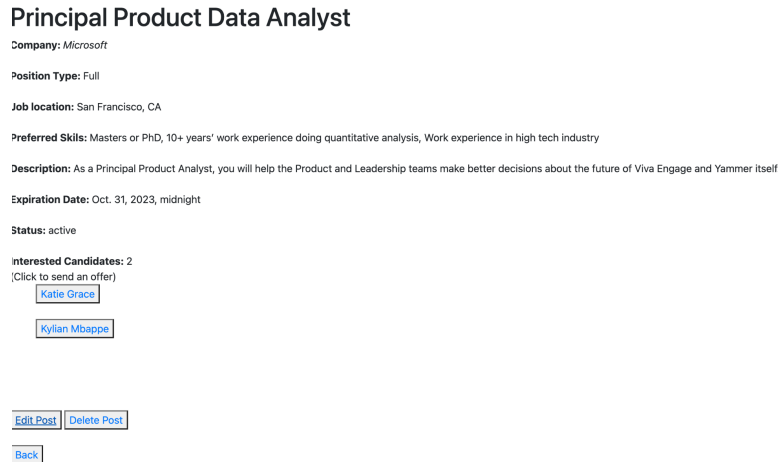
Expiration date:

Status:

Figure 15 screenshots for feature 2.2 when a recruiter does not properly fill out the form

## 2.3 Updating a post

When a recruiter wants to update a post, they must click on the “View all Posts” button on the recruiter dashboard (*figure 9*). Then they must click on the specific post they want to update and they will be redirected to the page shown below in *figure 16*. At the bottom of this page is a “Edit Post” button. The recruiter will finally be redirected to the webpage shown in *figure 17* where the recruiter can update and make changes to the job postings they have created.



**Principal Product Data Analyst**

Company: Microsoft

Position Type: Full

Job location: San Francisco, CA

Preferred Skills: Masters or PhD, 10+ years' work experience doing quantitative analysis, Work experience in high tech industry

Description: As a Principal Product Analyst, you will help the Product and Leadership teams make better decisions about the future of Viva Engage and Yammer itself.

Expiration Date: Oct. 31, 2023, midnight

Status: active

Interested Candidates: 2  
(Click to send an offer)

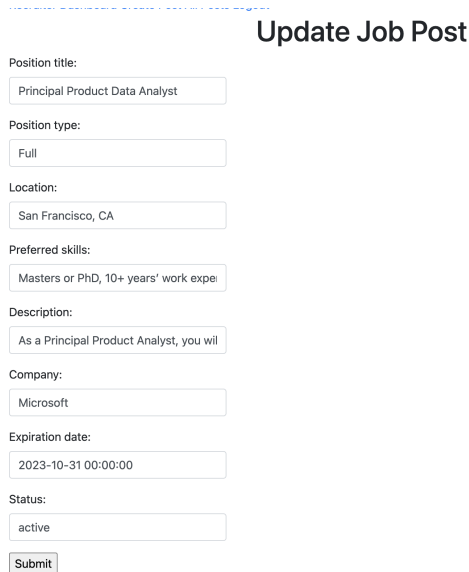
[Katie Grace](#)

[Kylian Mbappe](#)

[Edit Post](#) [Delete Post](#)

[Back](#)

Figure 16 screenshot for feature 2.3



### Update Job Post

Position title:

Position type:

Location:

Preferred skills:

Description:

Company:

Expiration date:

Status:

Figure 17 screenshot for feature 2.3 to update a post

## 2.4 Deleting a post

To delete a post, the recruiter will follow a similar process that they did to update one. Once the recruiter gets to the Job Postings page, they must click on the job posting they would like to delete and click the “Delete Post” button as seen in *figure 16*. Once the recruiter does so, the

specified post will be deleted from the database and no longer seen on the job postings page (see figure 18).

### Job Postings

Filter Posts:

- [Video & Technology Specialist, Cincinnati Reds, Cincinnati, OH](#)
- [Account Executive, Retention & Sales, Cincinnati Reds, Cincinnati, OH](#)
- [BAD POST, bad post, New York, NY](#)

### Job Postings

Filter Posts:

- [Video & Technology Specialist, Cincinnati Reds, Cincinnati, OH](#)
- [Account Executive, Retention & Sales, Cincinnati Reds, Cincinnati, OH](#)

Figure 18 screenshot for feature 2.4 before (left) and after (right) deleting “BAD POST”

### 2.5 Make an offer to interested candidates

To make an offer to interested candidates, the recruiter will click on the job posting of the job they want to offer, then click on the “Send Offer” button (see figure 16). Once they do so, they are redirected to the page shown in figure 19, they can then input the offer informations and click create to send the candidate an offer.

## Create an Offer

Salary info:

Due date:

Figure 19 screenshot for feature 2.5 when a recruiter enters in offer details

### 2.6 View compatibility with interested candidates

When a recruiter clicks on on of their job postings and gets a detailed view, at the bottom of the page, when the recruiter see the interested candidates (for that specific job posting) they will also see the compatibility score next to the user’s name as seen in figure 20. This can help a recruiter better evaluate their decision on giving a candidate an offer.

**Interested Candidates: 2**  
(Click to send an offer)

[Katie Grace](#)

Compatibility: 21.21

[John](#)

Compatibility: 20.0

Figure 20 screenshot for feature 2.6



## Feature 3: Candidates's Dashboard

### 3.1: Viewing Job Postings

When a candidate logs in, they are directed to the Candidate Dashboard (*figure 21*), to view the job postings, the candidate must click on the “View All Posts” button where they will be directed to the job postings page as seen in *figure 22*.

## Candidate Dashboard



Figure 21 screenshot for feature 3.1, the candidate dashboard

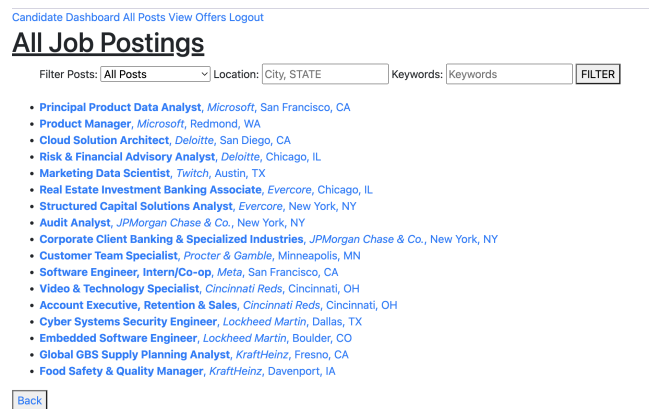


Figure 22 screenshot for feature 3.1 of the Job Postings page for candidates

If the candidate wants to, they can filter these posts with the drop down menu (*figure 23*) to see active and inactive job postings. Additionally they can enter in keywords, and a location to refine their job search. The user can choose to filter one, multiple or all categories.

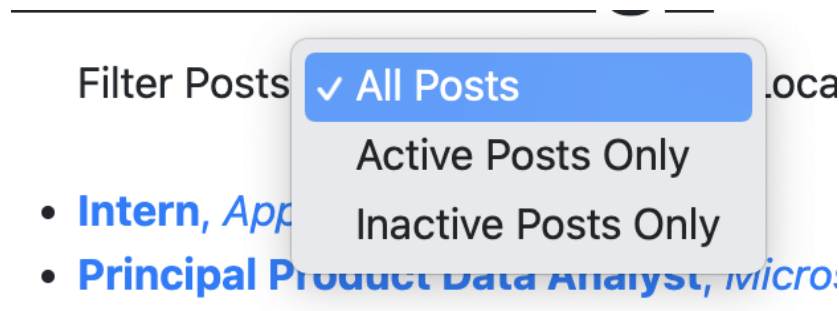


Figure 23 screenshot for feature 3.1 of the filter dropdown menu for candidates

### 3.2: Demonstrating Interests/ not interested in a job

When a candidate wants to demonstrate interest in a job, they will first click on the index view of a job from the job postings page (*figure 22*). This will allow the candidate to view the job posting with more detail (detail view) as seen in *figure 24*, and at the bottom of the post there are “interested” and “not interested” buttons. When the candidate clicks on either one, they will be redirected to the job postings page.

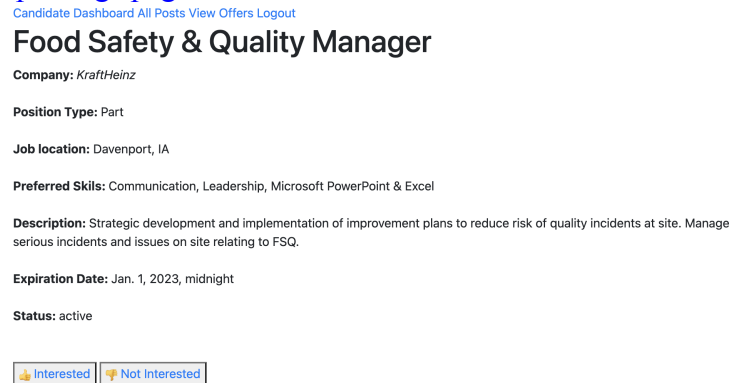


Figure 24 screenshot for feature 3.2

From the candidate dashboard, a candidate can click on the “View Interested Jobs” button to see the job postings they have liked/ demonstrated interest in as seen in *figure 25* below.

## All Interested Postings

- **Cloud Solution Architect**, Deloitte, San Diego, CA
- **Software Engineer, Intern/Co-op**, Meta, San Francisco, CA
- **Food Safety & Quality Manager**, KraftHeinz, Davenport, IA

[Back](#)

Figure 25 screenshot for feature 3.2

Additionally, on the recruiter side, when a recruiter clicks on one of their job postings, they can see if any candidates are interested in the posting and they will see their names as well (see *figure 26 & 27*).

**Interested Candidates:**  
(Click to send an offer)

[Celina](#)

[Chiara Thrum](#)

Figure 26 screenshot for feature 3.2 illustrating what a recruiter sees when there are interested candidates

**No interested candidates.**

Figure 27 screenshot for feature 3.2 illustrating what a recruiter sees when there are NO interested candidates

### 3.3: Viewing offers

On the candidate dashboard, candidates can click on the “View Offers” button to see if they have any offers in index view as seen in *figure 28*. They can click on the index view to get a detail view of the offer including salary information and expiration date (*figure 29*). If a candidate does not have any offers, they will see a “no offers available” message (*figure 30*).

#### All Offers

Video & Technology Specialist, Cincinnati Reds, \$120,000, Dec. 31, 2022, midnight  
Cyber Systems Security Engineer, Lockheed Martin, \$200,000, Dec. 8, 2022, midnight

[Back](#)

*Figure 28 screenshot for feature 3.3 for a candidate viewing their offer*

#### Cyber Systems Security Engineer

Company: Lockheed Martin

Position Type: Full

Job location: Dallas, TX

Preferred Skills: Active Security Clearance, Python experience, Experience with Agile practices, Experienced Programmer

Description: By bringing together people that use their passion for purposeful innovation, at Lockheed Martin we keep people safe and solve the world's most complex challenges.

Salary: 200,000

Due Date: Dec. 8, 2022, midnight

[✓ Accept](#) [✗ Decline](#)

[Back](#)

*Figure 29 screenshot for feature 3.3*

No offers available.

[Back](#)

*Figure 30 screenshot for feature 3.3 when a candidate has no offers*

### 3.4 Accepting/Declining offers

After a candidate clicks on the “View Offers” button on the candidate dashboard, they can click on specific offers to get a more detailed view of the offer (salary, expiration date etc) and they can either click on the “Accept” or “Reject” buttons to make a decision on a job to take (*figure 31*).

# Video & Technology Specialist

**Company:** Cincinnati Reds

**Position Type:** Part

**Job location:** Cincinnati, OH

**Preferred Skills:** Videography Skills, Film Analysis, Communication

**Description:** This individual will contribute to an efficient process for collecting video, creating video content, and distributing video to players and staff

**Salary:** 120,000

**Due Date:** Dec. 31, 2022, midnight

*Figure 31 screenshot for feature 3.4 of a Candidate's options to accept/decline an offer*

## Project's Learned Lessons

For this project, we used django which is a high-level Python web framework. The paradigms we used include: imperative programming, object oriented programming, and event-driven programming.

Python is an imperative language with a light and uncluttered feel to it. Implementing imperative programming allowed us to optimize our code and it was easier to understand and debug vs implementing declarative programming.

The event driven paradigm allowed us to change our program based on user actions. We chose this so that we could get user input and so that users could have an easy time interacting with our TinDev web application. This allowed users to enter their personal information and view postings/ others profiles. This was done through HTML DOM events.

Finally, python used class based OOP. This allowed us to use class inheritance which was helpful when connecting objects and models.

The paradigms we chose helped in developing our project as we were able to utilize DOM and user interactions with the webpage and optimize our code and create inheritance to connect the different aspects of our project.

One of the major factors of this project that made it so intellectually challenging was the steep learning curve we had. Starting off was difficult. We hadn't seen examples as complex as this project in class and there were not many examples online to refer to, and if there were examples, they were hard to follow and relate to our project. Therefore beginning this project and fully understanding how Django works was difficult (so difficult we had to re-do our phase 1), but we eventually got the hang of things and were able to speed up our work.

Another difficult portion of our project was that we could not connect the frontend and backend variables. As we were going through candidate and recruiter profiles, there were certain objects we could not access on the front end. This is done for safety reasons (as we do not want random accounts editing others accounts, posts, and interactions), but it made things frustrating as we couldn't access and modify objects as easily.

Additionally, comprehending the models and what connections to make was intellectually challenging. For example, an offer has a recruiter, candidate, and post all associated with it. Comprehending how to make these connections and how to code them was even more difficult but with the help of the TAs and Professor Santos, we were able to understand.

Our group's organization and focus contributed to our success and learning. We were very organized in planning our meeting times and created a shared google calendar to coordinate these daily times. For some of the project features, we worked together on them, rather than splitting up the work. This made us more efficient as working on these alone would be difficult. It was helpful to get multiple eyes on the code and different opinions/ideas on how to add the different features to our project and get them fully functioning. Additionally, going to office hours, asking questions before/after class and reaching out to TA's (especially Grigorii) and Professor Santos via CampusWire helped tremendously. There were many times that we did not understand the bugs in our code, but we were able to get help. Also this was all of our first times collaborating on a project via a GitHub Repo. GitHub was super easy to use and it was helpful to see the history of our code changes as well as share this with Professor Santos & the TA's for them to

look and interact with our code. Overall, a key factor to our group's success was our organization and consistent meeting times. This collaboration allowed us to learn off each other and test out our different ideas.