Team Member Full Name	NetID
Andres Gonzalez	agonza42
Mariana Gonzalez	mgonza32
Anna Koziol	akoziol
Gabe Elbling	gelbling

Chosen Technology Stack

- Python + Django

Persistent Storage Design

For our project's data, we are using SQLite database to persist our data. Our database currently includes the tables shown below in Figures a, b, c, and d. These tables consist of the candidate profiles, recruiter profiles, job postings, and offers, respectively, and holds all of their information within the database. We used forms to register users onto the platform, using a recruiter form for recruiters and a candidate form for candidates, which are then stored in separate tables. We used this same form process for both the posts and the offers. Hence, by doing this, we are able to keep track of all the profiles in the system, both for candidates and recruiters, as well as posts and offers, and will then be able to store all of the different posts and interactions that occur between the users of our platform. We then store the username and passwords of both recruiters and candidates in users in order to be able to authenticate each user individually, while also being able to connect them for the functionality of the posts and the offers which connects both types of users.

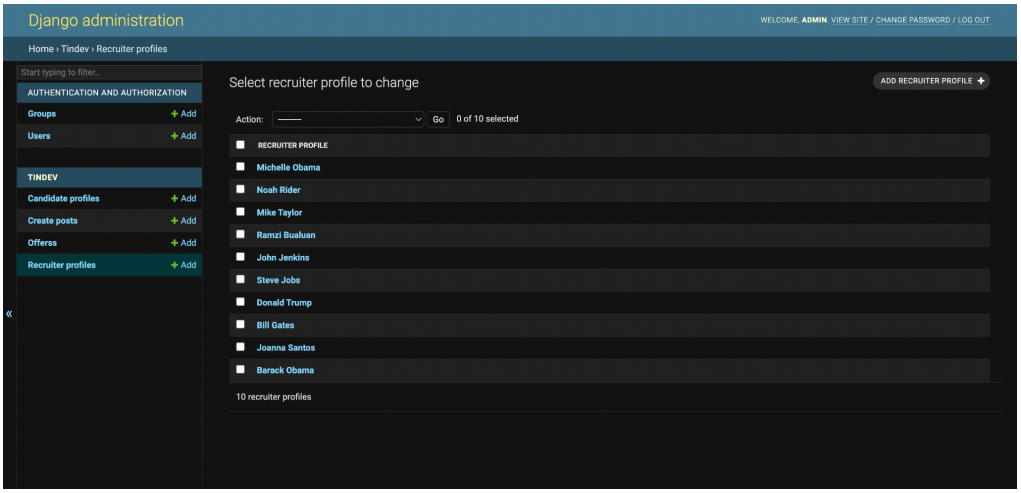

Figure a: Recruiter database storage

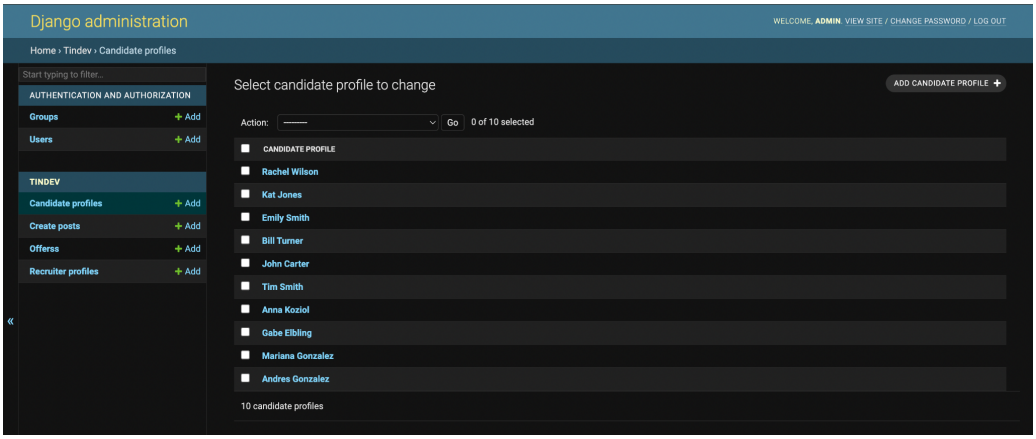

Figure b: Candidate database storage

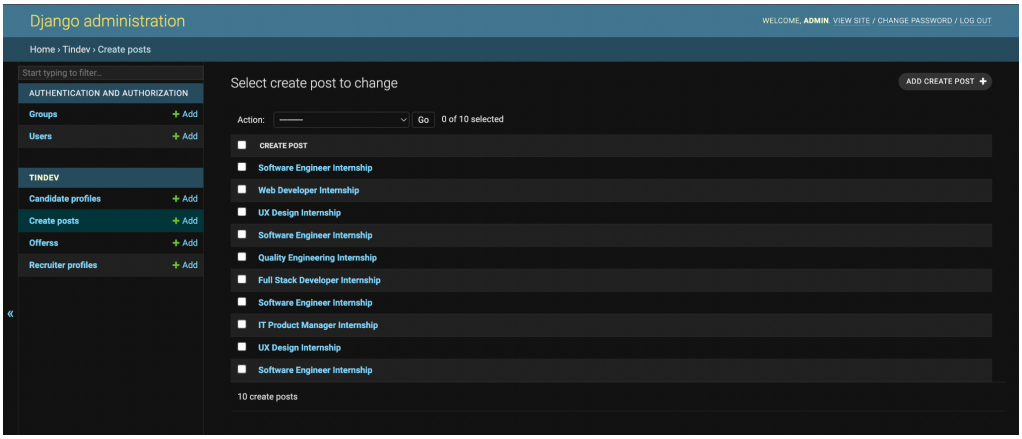

Figure c: Job postings database storage

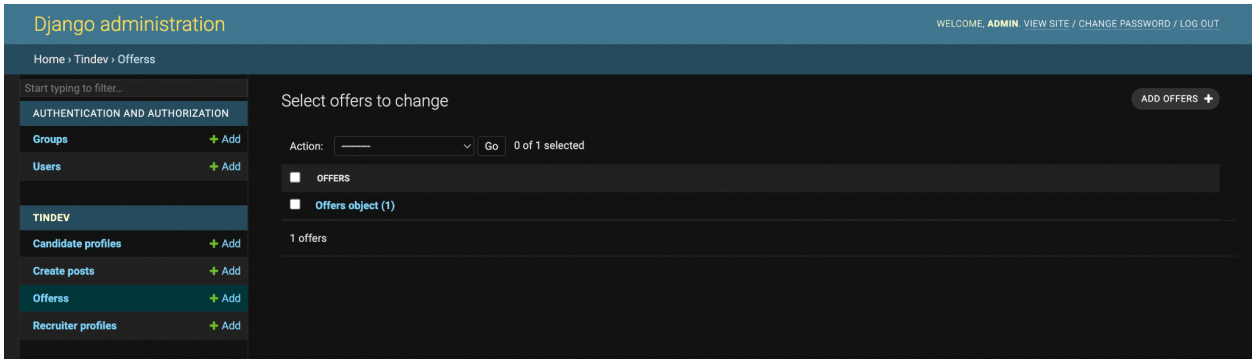

Figure d: Offers database storage

Demonstration of the Features

It should have at least one screenshot per feature. This should include not only the screenshots but also an explanation for them! Make sure the screenshots have good resolution (i.e., they can be readable when printed).

Feature A

For Feature A, we focused on giving users the ability to create a new candidate profile to use the platform. This is done through a form they have to fill out with important information and the data is then stored directly into our SQLite database. By creating their candidate profile, candidates are able to then use the platform to look at different job postings, connect with others, and hopefully land their dream job. Below, figure 1 shows a screenshot for the form that is filled out for candidates when registering, and figure 2 shows a screenshot of how their information is stored in the database.

Figure 1: Form filled out by candidates

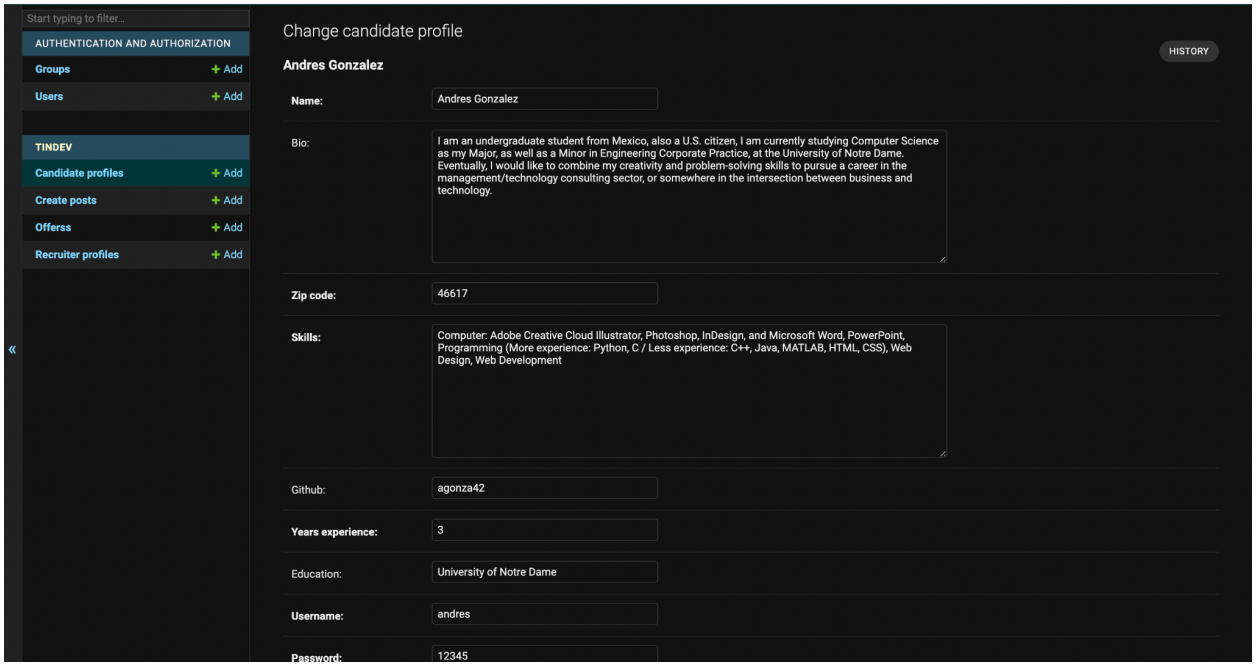

Figure 2: Candidate information being stored in the database

Feature B

For Feature B, we focused on giving users the ability to create a new recruiter profile to use the platform, which is done through a form they have to fill out with important information and that data is then stored directly into our SQLite database. By creating their recruiter profile, recruiters are able to then use the platform to post different job opportunities for potential candidates, connect with others, and hopefully be able to attain lots of talented people. Below, figure 3 shows a screenshot for the form that is filled out for recruiters when registering, and figure 4 shows a screenshot of how their information is stored in the database.

Figure 3: Form filled out by Recruiters

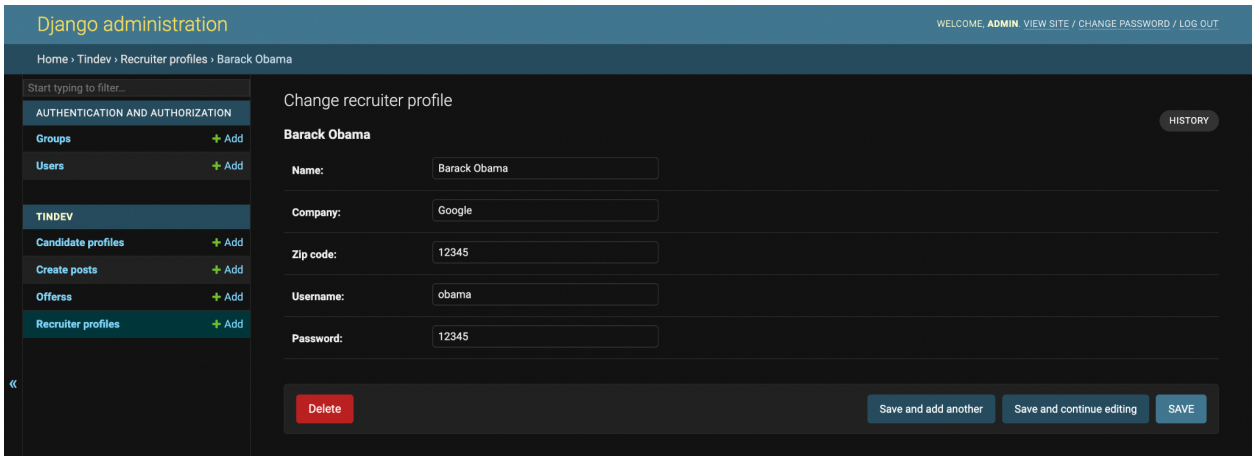

Figure 4: Recruiter information being stored in the database

Feature C

For Feature C, we focused on giving users the ability to log-in to the platform with their profile, which would be done by providing their username and password, as well as their account type. The way this is done is by using the username and password that the user enters to compare to the existing users in our SQLite database, which will be the credentials that were stored when they registered using either the candidate or recruiters. If the log-in credentials match what's in our database, they will be sent to our loggedIn page, but if not, then they will be prompted to re-enter their information or create an account if they haven't already. Below, figure 5 shows the form that users would use to log-in to the platform.

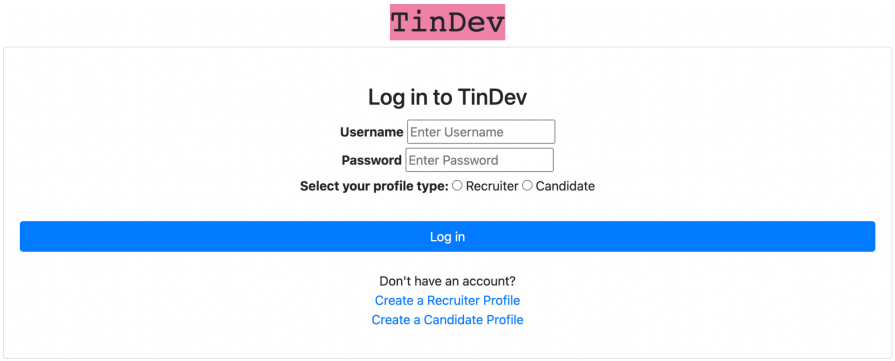

Figure 5: Log-in form required to enter the platform

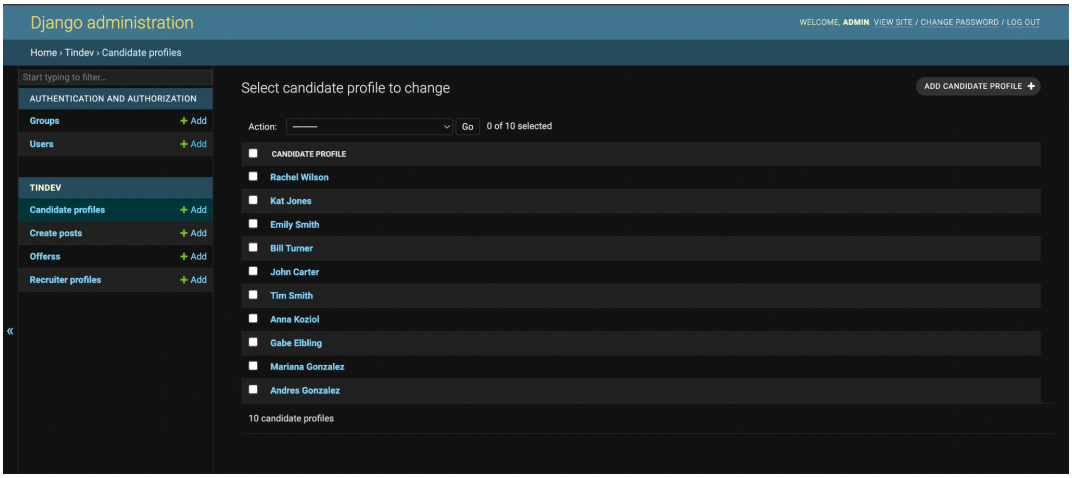

Figure 6: Database used to authenticate users

Feature D

For Feature D, we focused on giving users the ability to log-out of the platform, which would essentially only allow them to access the content on the homepage, but wouldn't allow them to access the recruiter and candidate dashboards. The way this is done is by using the username and password that the user enters to compare to the existing users in our SQLite database, which will be the credentials that were stored when they registered using either the candidate or recruiters. Once the user clicks the log-out button, then Django's authentication system will detect that the user is not authenticated anymore. Hence, when they try to enter another page, such as the dashboards, the system will show that they aren't logged in, and won't be able to see the page's contents as a result. Below, figure 7 shows the button that users would use to log-out of the platform.

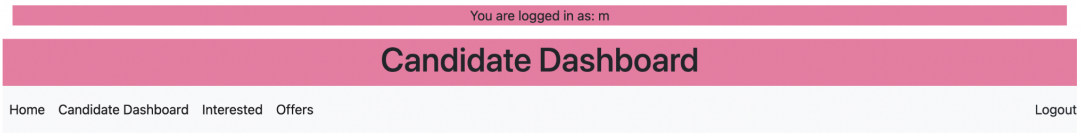

Figure 7: Log-out button in navigation bar

Feature E

For Feature E, we focused on allowing recruiters to be able to view all of their previously posted jobs on their recruiter dashboard, so that they can keep track of the opportunities they are offering. Also, this feature allows recruiters to filter job postings so that they are able to view all posts, only active or inactive posts, and posts with at least 1 interested candidate. We do this by using our database to filter the posts to only include those created by the recruiter that's logged-in to the system, and then the recruiter can filter these posts because the system itself filters them by analyzing comparing the criteria the user selected to the posts stored within the database. Below, figure 8 and 9 show how these posts can be displayed using distinct filters.

Figure 8: Recruiter posts showing all posts

Figure 9: Recruiter posts showing those with at least 1 interested candidate

Feature F

For Feature F, we focused on giving recruiters the ability to create a post, and these job postings would include the position title, type, location, preferred skills, description, company, expiration date, and status. As a result, this is the core feature which both the candidate and recruiter dashboards will revolve around, since the entire purpose of the application is to connect recruiters to candidates through these job postings. We do this through a form the recruiters have to fill out and that data is then stored directly into our SQLite database. Then, these stored posts are analyzed in different ways to make sure the functionality in the other features works as intended. Below, figure 10 and 11 illustrate how the form works and how the posts are stored in the database.

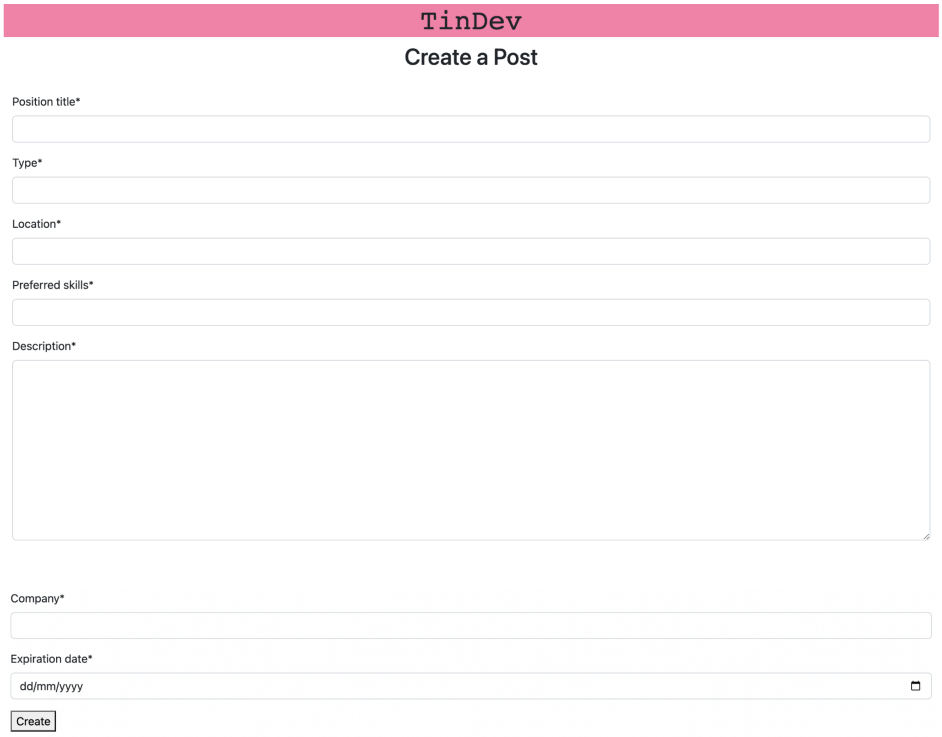

Figure 10: Form that recruiters fill out to create job postings

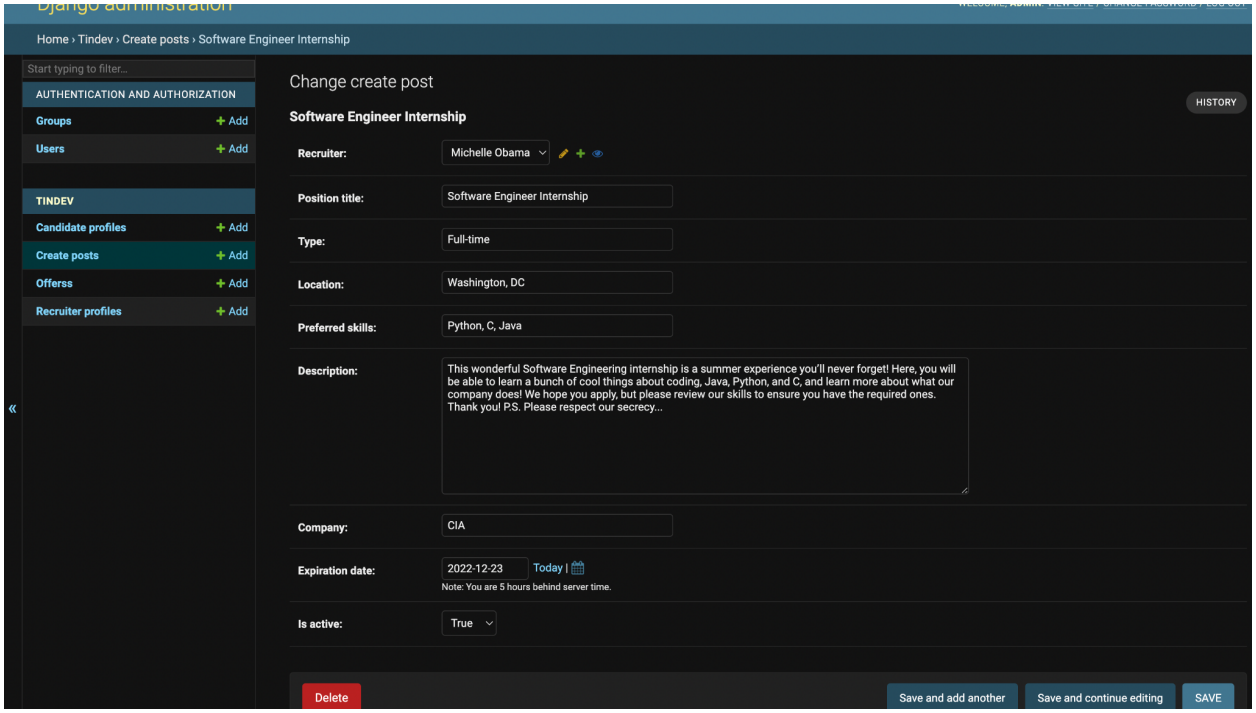

Figure 11: How the job postings are stored in the database

Feature G

For Feature G, we focused on giving recruiters the ability to update a post, which essentially means that they would be able to select one of the job postings they previously created and update any of its information, except the ID of the actual post. Of course, this is a very important feature, since job postings are subject to change and giving recruiters this functionality is essential because of that. We did this by adding a button that would allow them to change their previous posts, which would then be done by filling out a form that contained the original information to simply edit what you want to update before clicking to update button. Below, figure 12 shows the button to update the posts and the form would be the same as shown in figure 10 above.

Figure 12: Button to update recruiter's job postings

Feature H

For Feature H, we focused on giving recruiters the ability to delete a post, which consists of giving users the ability to select one of the job postings they'd previously created and delete it entirely. Still, it is important to note that the job posting ID is not reused, as it is meant to be unique and non-transferrable, This is a pivotal feature, as some job postings might suddenly not be applicable or available for some unanticipated reason, meaning that recruiters must be able to delete previous posts. This is done by adding a button that allows the recruiter to delete a specific post, and clicking it would remove the post from the database entirely, but the ID would not be reused. Below, figure 13 shows the button to delete a specific post.

Figure 13: Button to delete a specific job posting

Feature I

For Feature I, we focused on allowing recruiters to make an offer to interested candidates, as they'll be able to select one of their posted jobs and see which candidates are interested in them. Also, this feature gives the recruiter the ability to select one or more candidates to make an offer, and the offer would include the yearly salary information and a due date by which the candidate must accept the offer. The way we do this is by showing the interested candidates in the individual job postings and adding a button which takes the recruiter to a form to send out offers to candidates. These forms are stored in the database, which would allow the system to connect those offers to the candidates so that they can access them as well. Below, figure 14 shows the button to access the offer form, figure 15 shows the form which needs to be filled out, and figure 16 shows how the offers would be stored within the database.

Figure 14: Button to access job offer form

Figure 15: Offer form to be filled out by recruiters

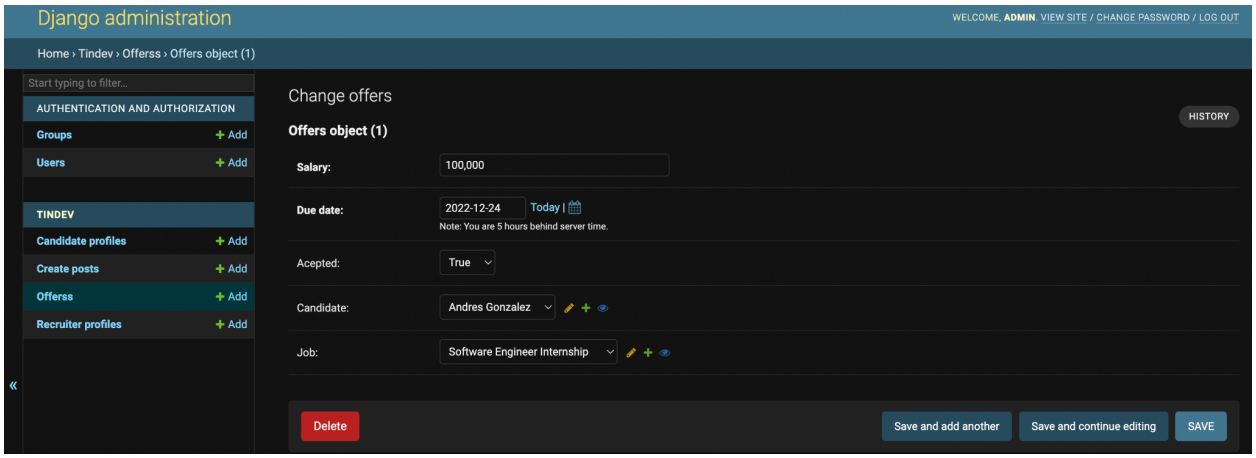

Figure 16: How offers are stored in the database

Feature J

For Feature J, we focused on allowing the recruiter to select one of the job postings and see the compatibility score of all the candidates interested in it. Basically, the system computes a compatibility score, given as a percentage (where 100% means the candidate is a perfect fit, 0% means the candidate's profile does not match the job posting). This compatibility score is done by comparing metrics in the job posting specifications and the candidate's profile to see how compatible they are with the job. Essentially, the factors we compared were keyword-based matching between the job's description and the candidate's bio, matching the post's required skills against the candidate's skills, and analyzing their years of experience. For each of the criteria that matched, the candidates would receive points, and would then be given a percentage score out of 100. Below, figure 17 illustrates how the different compatibility scores are shown for interested candidates.

Figure 17: Different compatibility scores for interested candidates

Feature K

For Feature K, we focused on allowing candidates to view all of the previously posted jobs on their candidate dashboard, so that they can keep track of the opportunities that recruiters are offering. Also, this feature allows candidates to filter job postings so that they are able to view posts by job post status, by location, or by different keywords in the post's description. We do this by using our database to filter the posts to include all the posts created by recruiters, and then the candidate can filter these posts because the system itself filters them by analyzing comparing the criteria the user selected to the posts stored within the database. Below, figure 18 and 19 show how these posts can be displayed using distinct filters.

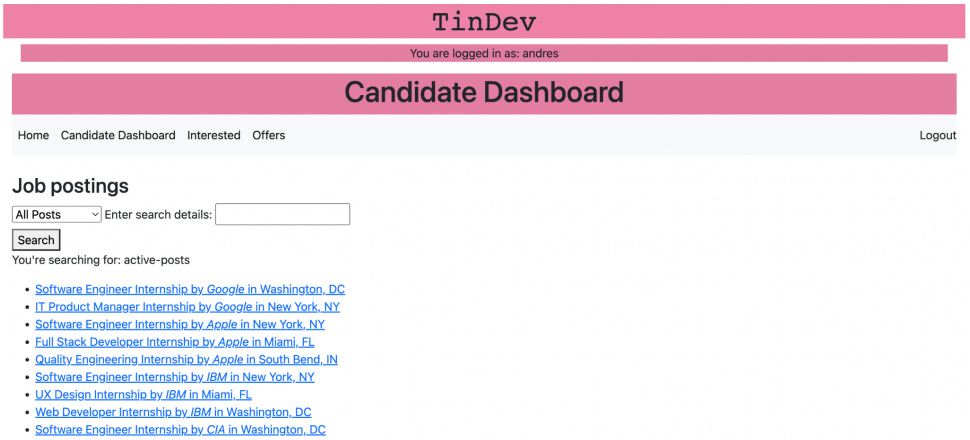

Figure 18: Job postings showing all active posts

Figure 19: Job postings showing posts by location

Feature L

For Feature L, we focused on giving candidates the ability to select a posted job which is still active, and demonstrate whether they are interested in the job or not. This interaction with the post would then be stored in the system and would associate the candidate with the job posting, and thus with the recruiter. Hence, the database would be connected in a way that would allow the recruiter to see which candidates are interested, as we mentioned in the previous features. Then, the candidate will see all the posts he's interested in by clicking on the "Interested" tab. Below, figure 20 illustrates how candidates can interact with these posts to demonstrate interest.

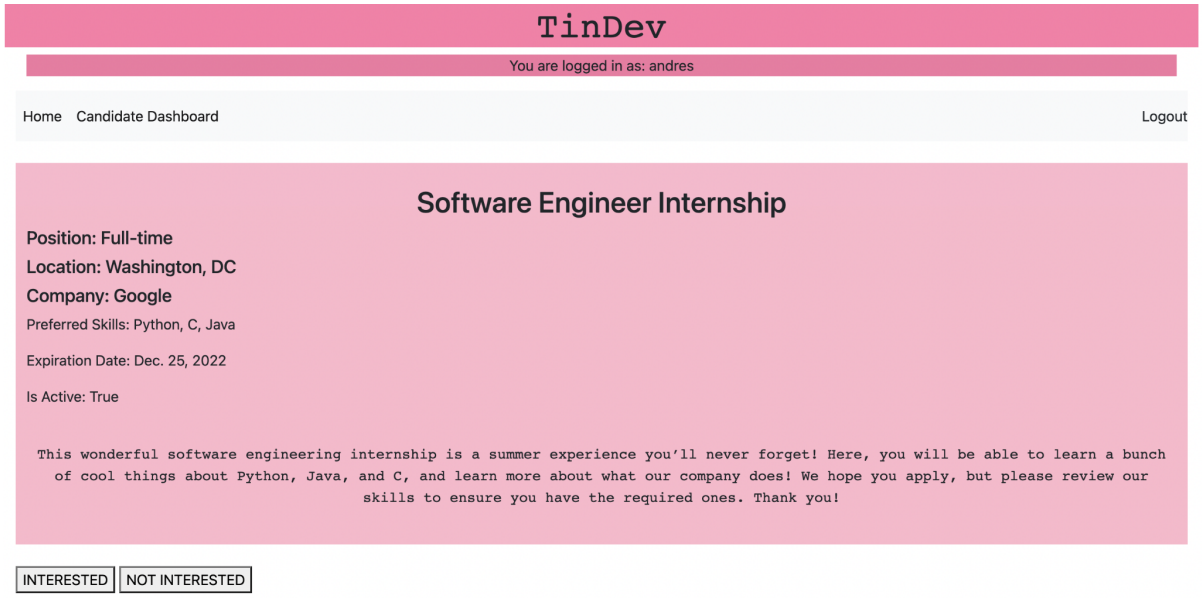

Figure 20: Candidate interaction with job postings to demonstrate interest

Feature M

For Feature M, we focused on allowing candidates to see all of the job postings that they had received an offer from. Hence, all these postings are displayed on the page in an organized manner so that they can go over them, but if an offer has already expired, then the system indicates that the offer expired on a specific date X. The way this works is by accessing the connected database which holds the interested candidates that have received offers from employers, which is done through the offer forms, and then the candidate can go over the offer they received to understand what they're being offered. Below, figure 21 shows how the candidate can view all of the job offers they have with their details.

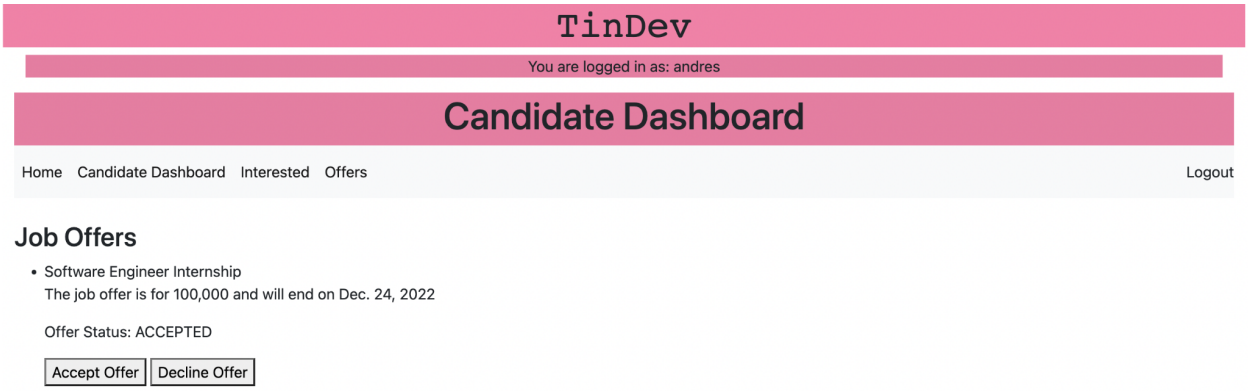

Figure 21: All job offers sent to the candidate

Feature N

For Feature N, we focused on giving candidates the ability to either accept or decline an offer sent by a recruiter. Still, it is important to note that the system will only allow the candidate to do so that the if the decision was made prior to the offer expiration date. If the offer expiration date has already passed, then the system shall not allow the candidate to accept or decline the offer. This is done by adding an accept and a decline button on the pages where the candidate is looking over the specified job offer. Thus, accepting the offer will indicate that the offer has been accepted, and declining the offer will indicate that the offer has been declined. Below, figure 22 illustrates the accept and decline buttons, figure 23 shows the offer status.

Figure 22: Accept and decline buttons on the specific job offers

Figure 23: How the offers page shows the status of the offer

Project's Learned Lessons

In this section, you will reflect on the group's activities and performance through the entire project, and capture your reflections, observations and thoughts. It should address the following items, but feel free to expand on these in light of your group's unique experiences.

1. What programming paradigm(s) have you chosen to use and why? If you were to start the project from scratch now, would you make different choices? Do you think the paradigm(s) chosen helped (or not) in developing the project?

We used a high-level Python web framework known as Django for our project. We employed several programming paradigms, such as imperative programming, object-oriented programming, and event-driven programming. Also, we used Python, which is an imperative language that has a very light and uncluttered feel to it, which makes it effective to work with on this sort of project of this scale. Additionally, the event-driven programming we implemented allowed us to change the program and our system based on how the user interacted with it. Essentially, this is a pivotal part of all platforms for users, since user input such as log-in, post creation, post interactions, and eventually job offers, would all affect the experience users would get, and how the system would interact with them. This shifting between pages and experiences based on user input was created using HTML DOM, since the movement between the application's pages depended on what these users wanted, and our program would respond and shift accordingly. Finally, the object-oriented programming of our program was essential, as recruiter and candidate profiles, as well as job postings, were all different types of objects and models that were stored in our database, which were then accessed and analyzed to ensure that the program functionality was as intended.

2. What were the most intellectually challenging aspects of the project?

The most intellectually challenging part of the project was connecting the different models and forms that we were using throughout the project to make sure we had the proper functionality. Basically, if the project was done using the Django administration page, then it would be much easier to work through and create. However, creating all of these complex features and components, as well as making sure that they would all be connected to each other was quite difficult. For example, making sure that the job posts could be connected through the database to both the recruiters and candidates, as well as connecting recruiters and candidates through interests and job offers was very complex since there were so many different aspects of the data to keep track of. Still, we managed to pull through in the end, but it required lots of time and effort to get there.

3. What aspects of your process or your group's organization had the largest positive effect on the project's outcome?

Our communication and collaboration with one another had the largest positive impact on the project's outcome. We were consistently able to set up times to meet that worked for everyone and put in effort to make substantial progress in each meeting. We bounced ideas off one another and split up tasks

amongst each member of the group. For example, while one person was pulling the code to edit, the others would be researching and watching videos to help the coding process. We displayed the code on a TV and were all able to contribute at once. Hence, we successfully worked together in an efficient way to produce a final project that we are all proud of, and it couldn't have been done without the collaboration we had throughout the process.