Dylan Breslaw	**dbreslaw**
Trevor Komeiji	**tkomeiji**
Jack Galassi	**jgalassi**
Aileen Dugan	**adugan2**

Chosen Technology Stack

X Python + Django Python + TkInter

Features Implemented for Phase 2

- **Feature 2: Recruiter's Dashboard**
  - **2.1 View all Posts**
  - **2.2 Creating a Post**
  - **2.3 Updating a Post**
  - **2.4 Deleting a Post**
  - **2.5 Make an offer to interested candidates**
  - **2.6 View compatibility with interested candidates**
- **Feature 3: Candidate's Dashboard**
  - **3.1 Viewing Job Postings**
  - **3.2 Demonstrating interest / not interested in a job**
  - **3.3 Viewing offers**
  - **3.4 Accepting/Declining offers**

Persistent Storage Design

We are using SQLite database to persist our data. Our database includes the tables shown in Figure 1. It contains 3 tables. The Post table which includes the attributes JOB_TYPE, position, type, location, skills, description, company, expiration, status, and interest. The Post table stores all the necessary information for Feature 2.2 which is where we allow recruiters to create posts. The interest attribute is what allows candidates to select which posts they are interested in and for recruiters to view posts with interested candidates. Our second table is the User table which

contains the attributes USER_TYPES, name, zipcode, username, password, user_type. It also contains the attributes profile_bio, education, github, experience, and skills for candidates; the User model then contains the company attribute for recruiters. The User table is used for collecting the necessary data that candidates and recruiters need to provide to the application in order to create their respective candidate and recruiter profile. Our last table includes the Offers table. The Offers table includes the attributes status, salary, expiration, user, and post. All of these attributes are necessary to allow the recruiters to select candidates for whom they are offering a job to. The status attribute is either pending, accepted, or rejected. The salary and expiration attributes allow the recruiter to disclose to the user this information. The user and post attributes are what connect the candidate and/or recruiter to the specific posting

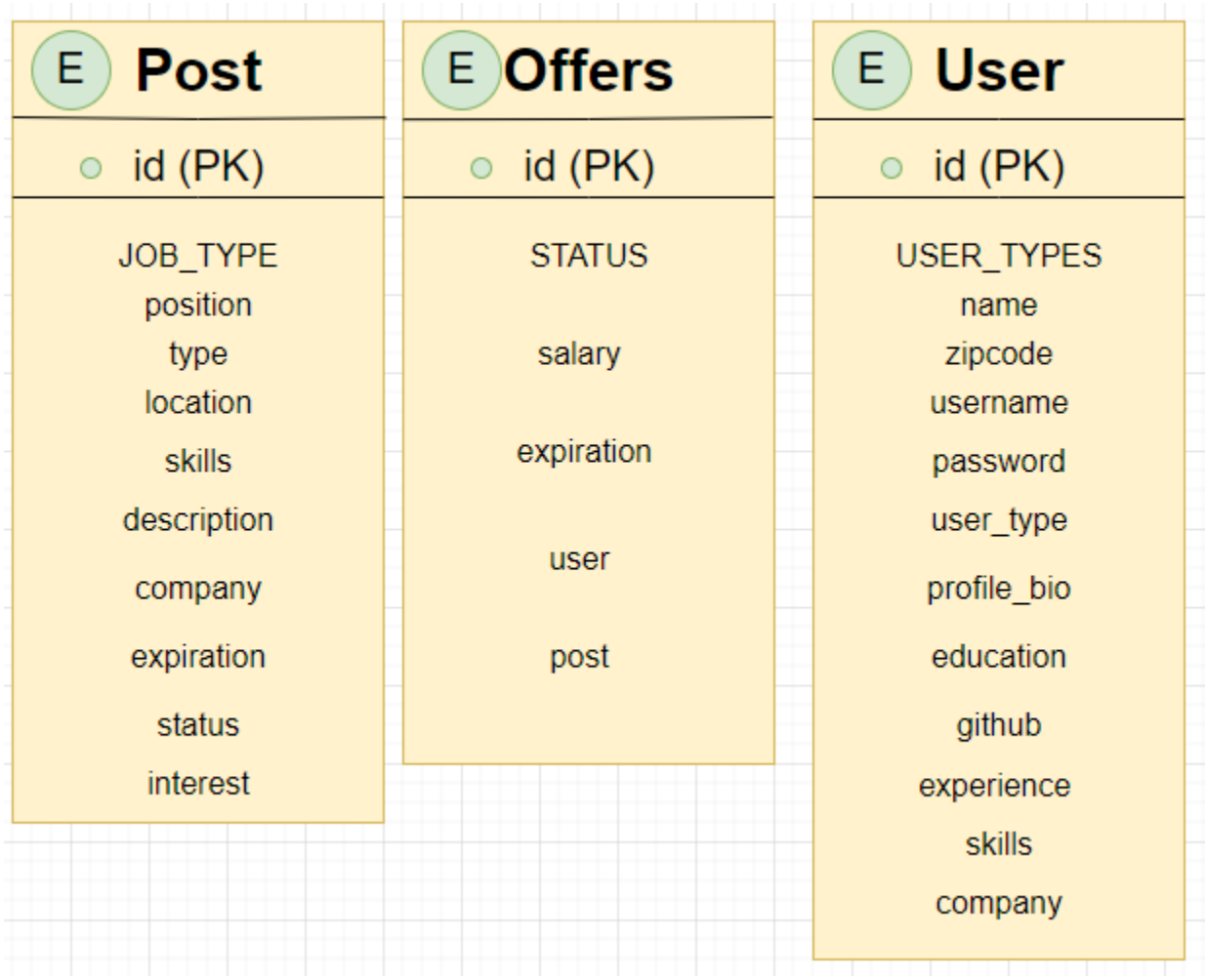


Figure 1 database schema

Demonstration of the Features Implemented for Phase 2

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 are readable when printed).

Feature 2.1

Figure 2.1 demonstrates how a recruiter shall be able to view all of its previously posted jobs and how those job posts can be filtered such that they can view all posts, active posts only, inactive posts only, or posts with at least 1 interested candidate. This feature was implemented to ensure a level of organization to recruiters such that they can keep track of all their job postings and any candidate users available and interested for potential hiring. The job postings, as seen in Figure 2.1 , display the Position Title, Company, Status, Type, and number of Interested Candidates to the recruiter user.

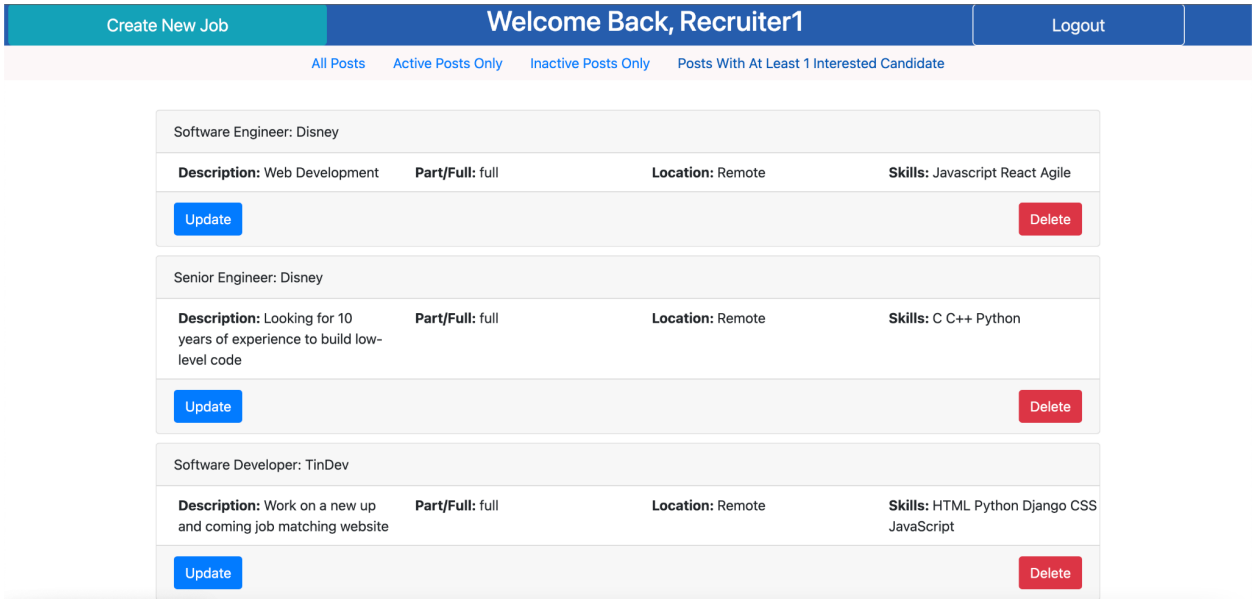


Figure 2.1

Feature 2.2

The implementation of this feature allows recruiter users, to create a job post with the necessary information such as Position title, Type (Full Time or Part Time), Location, List of Preferred Skills, Description, Company, Expiration Date (aka the date at which the job posting will automatically become inactive), and Status. These details that can be added when creating a job posting for recruiter users and the format of which job posting creation is done can be seen in Figure 2.2.

Figure 2.2

Feature 2.3

Figure 2.3 shows the implementation of how a recruiter user can select one of the job postings they previously created and update any of its information. This is useful for a recruiter user for many reasons such as changing the expiration date to keep the job posting active longer if the recruiter user wishes to keep looking for candidates past their original deadline or edit the list of preferred skills to try to bring in candidates that better fit what the recruiter user is looking for.

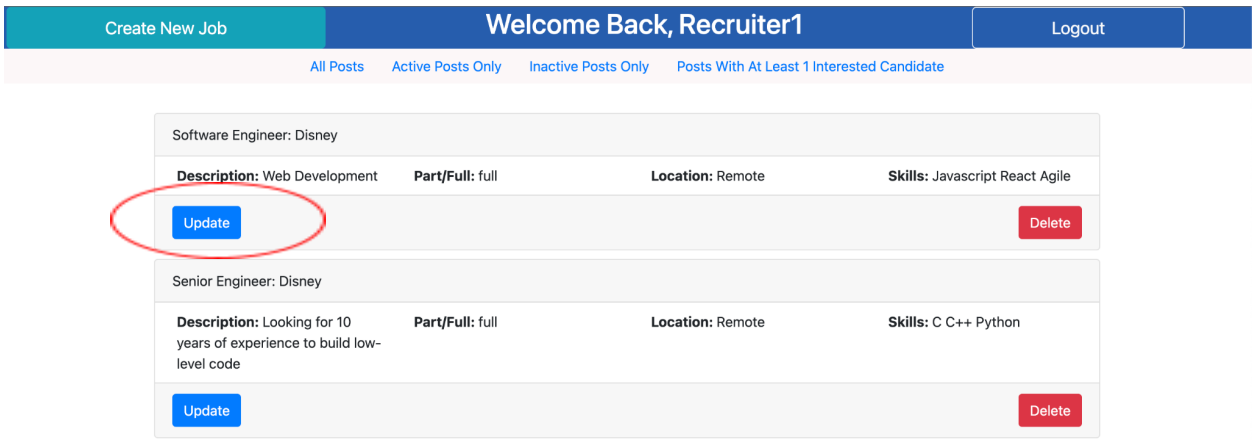


Figure 2.3

Feature 2.4

This feature, shown in Figure 2.4, was implemented to enable recruiter users to select a job posting and delete it entirely. This is different from a job posting going inactive/expiring as it will be erased and not shown on any job post viewing lists. However, the post ID for that job posting will not be reused as the ID is unique and non-transferrable. This allows the recruiter user to delete a posting that they might have made a mistake on or other reasons one might have.

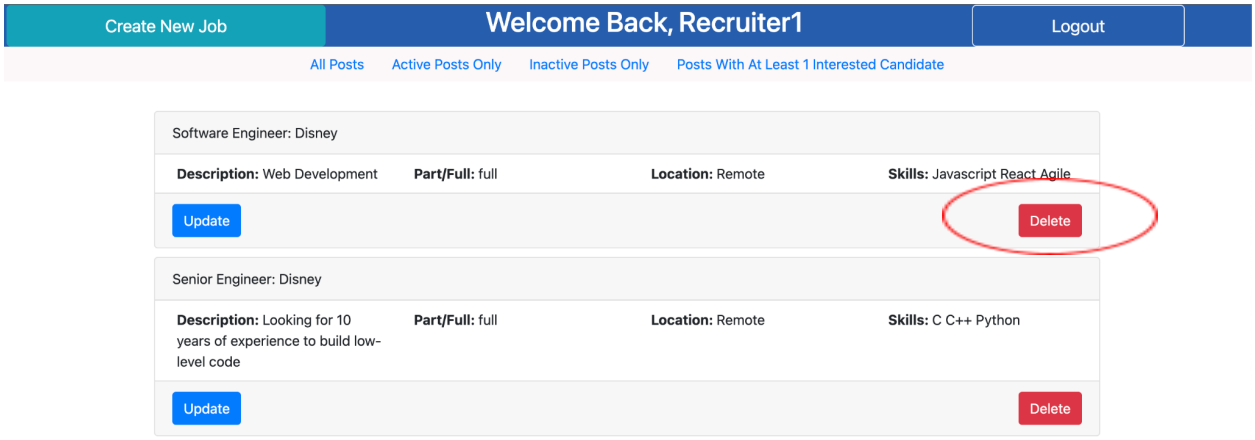
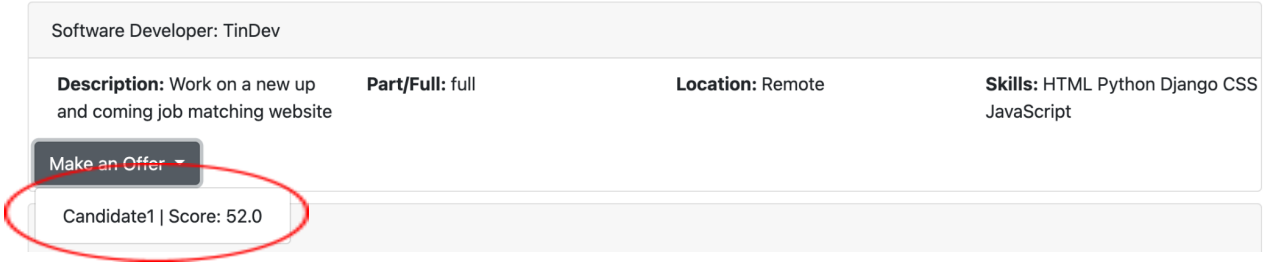


Figure 2.4

Feature 2.5

Figure 2.5 shows our implementation that allows a recruiter user to be able to select one of their posted jobs and see which candidates are interested in them. The recruiter user can then select 1 or more candidates that are interested to make an offer. Figure 2.5 for feature 2.5 shows the format of forming an offer to candidates which includes the yearly salary information and the due date (aka the date at which the candidate must accept the offer). This feature is important for the recruiter user to reach out to the proper candidates that they might want to hire for the job they have posted and important for the candidates to be able to hear back from the recruiter user on the jobs they are interested in.



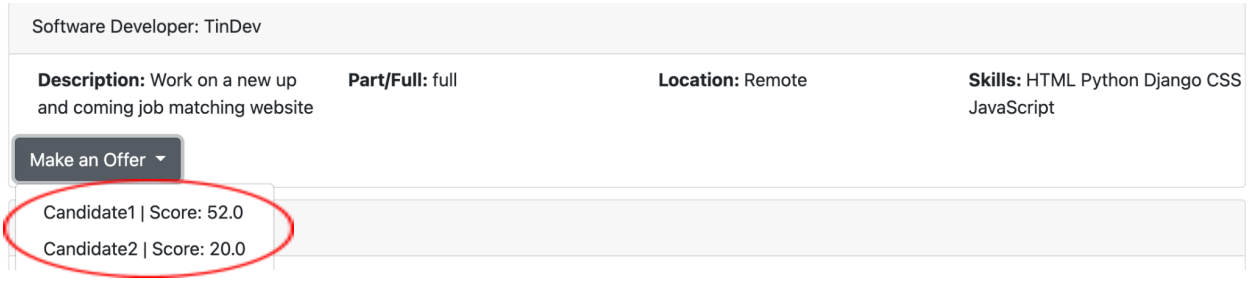


Figure 2.5

Feature 2.6

The implementation of this feature, shown in Figure 2.6 , allows the recruiter user to select one of the job postings, view all the candidates interested in it , and see a compatibility score as a percentage of how closely the candidate's interests generally fit the job post. The way this compatibility percentage was chosen to be calculated was by 20% location and 80% keyword matching between skills required for the job and skills the candidate has. We decided to calculate it this way because we felt that it gives the most accurate representation of the correlation between the candidate user’s interests and the job posting.

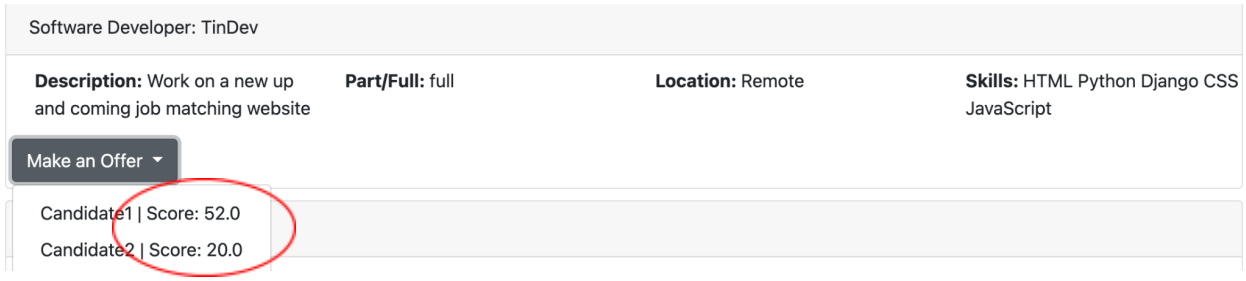


Figure 2.6

Feature 3.1

This feature is the first one implemented out of the rest that are candidate user focused. Figure 3.1 shows our implementation of this first candidate focused feature which allows a candidate to access their dashboard where they can see Job Postings and Job Postings they have expressed interest in once the candidate user has logged in.

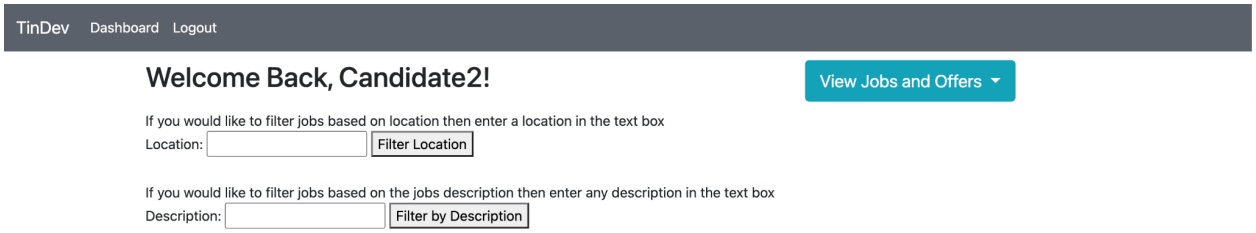


Figure 3.1

Feature 3.2

Figure 3.2 shows the implementation of our feature that enables a candidate user to select a posted active job and demonstrate their interest or disinterest in a job. This is a useful tool for candidate users so they can communicate with recruiter users that they are interested in a job and would want to be given an offer.

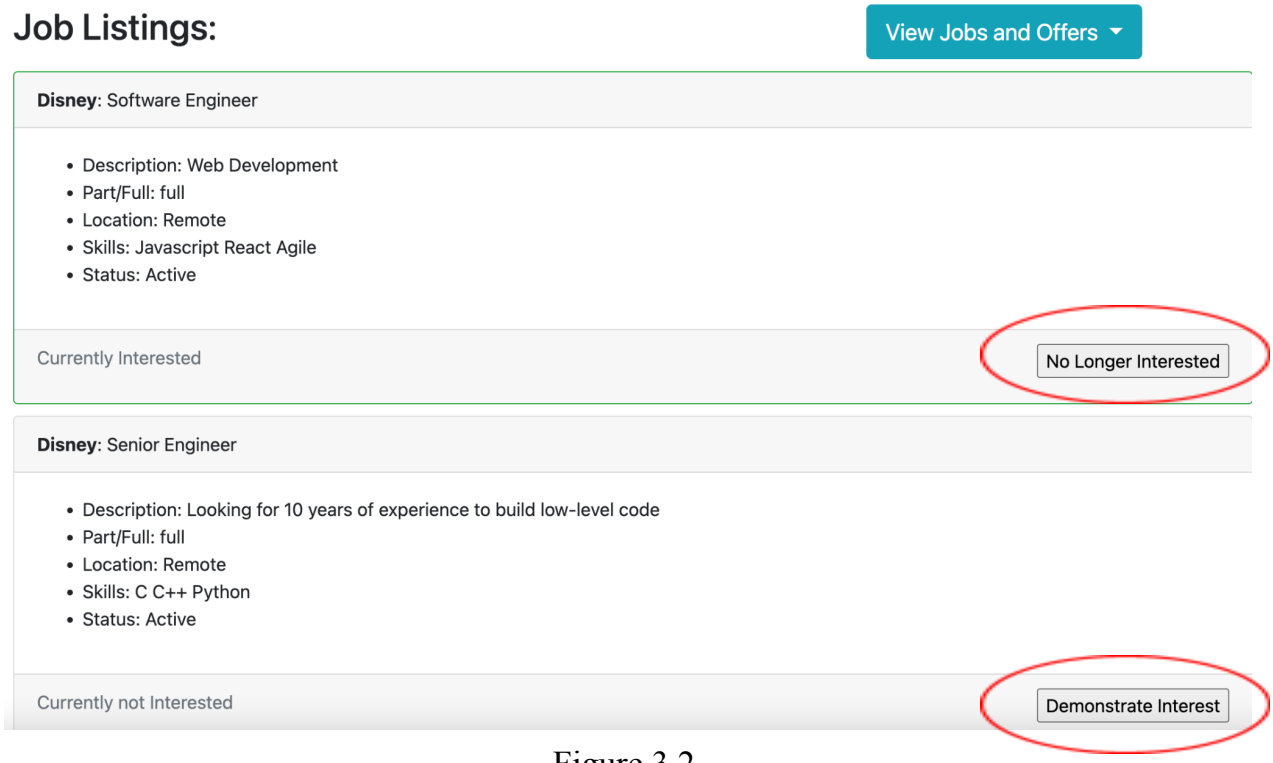


Figure 3.2

Feature 3.3

Our implementation of this feature, which allows a candidate to be able to see all the job postings that they received an offer from, can be seen in Figure 3.3. For expired offers, the system clearly indicates to the candidate user what day the offer expired on. This is

helpful for candidate users to be able to check offers they are given by recruiters from the job postings they have expressed interest in and keeps this organized.

Figure 3.3

Feature 3.4

This feature, shown in Figure 3.4, allows a candidate user to see and accept/decline an offer. A candidate user may only have this ability to accept or decline an offer if they are doing so before the offer's expiration date otherwise they are unable to do so. This is an important feature as this is how a candidate user will be able to accept a job offer that they want which is an integral part of the recruiting and hiring process.

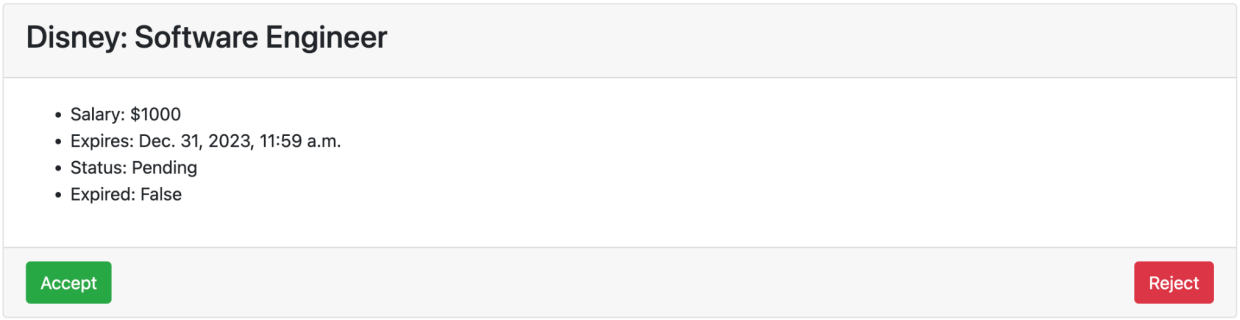


Figure 3.4

Project's Learned Lessons

In this section you will reflect on the group's activities and performance throughout the entire project, and capture your reflections, observations and thoughts. It should also 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?
    a. We chose to use a more imperative paradigm for displaying content. However, we used a declarative paradigm for managing the data. We think it made sense for the content to be declared imperatively so that we could manage exactly how it looked. For the data it made sense to use the declarative methods because we only cared about the end product. We didn't care about how the data was being filtered.
    b. Starting from scratch, we would definitely take advantage of the object-oriented nature of the Django templates. Often times, we were copying and pasting code from different portions of the templates. Instead of copy and pasting, we could have used more template inheritance and including blocks.
    c. I think the paradigms we chose helped in developing the project because it felt like it was the easiest way for us to understand the code. However, it did have some drawbacks once we really understood Django. At times, it felt like we were fighting against the nature of Django and that really slowed us down.
2. What were the most intellectually challenging aspects of the project?
    a. Some of the more intellectually challenging aspects of the project I felt dealt with the database management. First of all, when designing some of the models or database tables it was difficult to figure out how some of the attributes should be used. For example, navigating and understanding how ManytoOne or foreign keys should be implemented was a difficult but rewarding aspect of the project.
    b. We also thought that Django was a bit difficult to use at first. Sometimes we were trying to work only on the front end like filtering the jobs, but then we realized that it made more sense to complete some of these tasks in the applications back-end.
3. What aspects of your process or your group's organization had the largest positive effect on the project's outcome?
    a. The Git workflow was very helpful because it allowed us to work on separate portions of the code and then merge it into one final product.

Delete later: logins and names of 10 recruiter users and 10 candidate users
(most of these are jokes based on characters from Silicon Valley and Mythic Quest)
RECRUITER USERS:

1. Gavin Belson of Hooli - Gbelson200 password: Money1234$$
    a. Product engineer for Hooli XYZ
    b. Lead Engineer Hooli XYZ
    c. Figure Head Hooli XYZ
2. Erlich Bachman of Aviato - Bachmanity420 password: iAMgreat22
    a. Incube funded by Aviato
3. Richard Hendricks or Pied Piper - Rhendricks44 password: TABSnotSPACES
    a. Network engineer for pied piper
    b. Java developer for pied piper
    c. COO for Pied Piper
4. Peter Gregory of Raviga Capital - PeterGregory password: TheInternetWeDeserve
5. Laurie Bream of Bream Hall Capital - Breams34 password: WellYes&&634

    a. Associate partner for Bream Hall Capital
6. Ian Grimm of Mythic Quest - Grimm99 password: SharpMindShark44
    a. Game Tester for Mythic Quest
7. Poppy Li of GrimmPop Studios - PoppyL1 password: IdontNeedIan!0!0
    a. Publicity Manager for GrimmPop Studio's game Hera
    b. Software Intern at GrimmPop Studio
8. Michael Doc of Oubliette Studios - DocM222 password: MeanBeanMachine<3
9. David Brittlesbee of Montreal - DavidBb717 password: LoneWolfBoy444
10. Jack Barker of Triangle - ActionJack password: TriangleOfSuccess101

CANDIDATE USERS:

1. Nelson Bigetti - BigHead202 password: ohThatsCool111
2. Dinesh Chugtai - DinnyChug password: WishIwasRich99$$
3. Bertram Gilfoyle - GilfoyleB66 password: SatanistEG111000
4. Monica Hall - MonicaH24 password: doBetterRichard1!1!
5. Brad Bakshi - Bakshi999 password: MoneyPigCa$hCow808
6. Rachel Ramone - RRamone2 password: Feminism4Life!!!
7. Dana De'Lores - DanaD01 password: Gaming4Life!!
8. Aileen Dugan - adugan02 password: _____
9. Carol Carman - CarolCar83 password: HRviolationsEverywhere
10. Carl Weathers Longbottom - CWLongBottom password: IwroteAbookOnce222