ApliTrack

Progress Report 3 - Functional Design

Group 21

Prempreet Brar - 30112576

Jayden Yeo - 30113049

Fedor Prokopchuk - 30122005

CPSC 471: Data Base Management Systems

Ahmed Al Marouf

November 15, 2023

Use Case Diagram

Below is a small explanation of some of the modelling decisions in the use case diagram. Note that we only modelled the main functions/features for brevity (ie. some miscellaneous functionality may not be modelled).

1. There are 4 main actions an applicant can do: View, Create, Update, Delete. In addition, they can also assign some of their created entities to other parts of the system (such as assigning an offer to a job posting).
    a. When viewing components of the system, users can filter, sort, or view. Note that not all filter/sorting functionality will be available for each part of the system. For example, a user cannot "Filter by Category" for non-categorized items (such as referrals).
    b. Users do not have free reign to update certain parts of the system. For example, on the "Update Interview" use case, the user can only update the Interview Stage or Interview DateTime; they cannot update the ID. We used our discretion regarding what values should be updatable in the diagram.
    c. When deleting, certain use cases provide the user the option to delete specific parts of a component. For example, a user can "Delete Whole Contact" or "Delete Certain Email(s)." We used our discretion regarding what values should be deletable (ie. if a contact no longer uses an email, the applicant may wish to delete their inactive email but still keep them in the system). This was typically for multi-valued attributes.
2. There are no multiplicities listed from the applicant to the use cases (which means that the multiplicity is implicitly 1). This is because while an applicant, for example, can create multiple interviews, they have to create each interview one-by-one (ie. they cannot create multiple interviews simultaneously). In other words, the multiplicity is for use case participation, NOT the Enhanced-Entity-Relationship diagram. An applicant can only participant in one instance of each use case at a given moment in time.
    a. There **are** multiplicity listed from the database to each use case; this is because the database is dealing with multiple view, create, update, delete, and assign operations simultaneously.

3. The admin has additional functionality not shown in our EERD or our Relational Model. This is because we felt the admin would otherwise not be too differentiable from a regular user.
    a. Additional functionality includes creating, deactivating, reactivating, and deleting users, giving permissions to certain user accounts (to "promote" them into admins), and resetting user passwords.
    b. Note that jobs, contacts are shared across all users to prevent duplication in the database. Consequently, only the admin is allowed to delete jobs, contacts, and companies (not users); this is because we do not want malicious users deleting jobs, contacts, or companies from the database. Users can however, update the job's, contact's, or company's attributes; this includes deleting job qualifications and responsibilities, and contact email(s) and phone number(s).
4. The user has additional functionality not shown in our EERD or our Relational Model (such as logging in, forgetting password, updating account info, and deleting account).
5. There is an arrow going from the "Email sent" use case to the "Update Password" use case; this is because after a user sends their "Forgot Password" link, the system will then allow them to update their password using that link (bypassing the preconditions of authentication).
6. In our EERD diagram, we modelled applicants and admins as being disjoint (separate); however, we have now decided to allow admins to perform all actions allowed by applicants (which is why you see admin inheriting from applicant in the Use Case diagram).

Sequence Diagram

Below is a small explanation of some of the modelling decisions in the sequence diagram. Note that we only modelled the main functions/features for brevity (ie. some miscellaneous functionality or rare exceptions may not be modelled).

1. There are differences across some diagrams where boolean status codes are obtained from the database (in the actual system this will be from the API). For example, in the "ManageUser" sequence diagram, boolean status codes are named SuccessOrFail. In the

"UpdateAccountInfo" sequence diagram, boolean status codes are named Success; this is because in the "UpdateAccountInfo" diagram, the case of failure is handled in an exception. In other words, where there are exceptions, the regular sequence is only labelled with success.

2. Note that you may see slight differences in syntax across sequence diagrams; this is because multiple group members worked on the diagrams (each using their own style). However, when looking at a sequence diagram in isolation, it is still correct (ie. even though group members used different forms of syntax, both forms still follow UML standards).