Design Document for SwipeHire

Group 1_CW_4

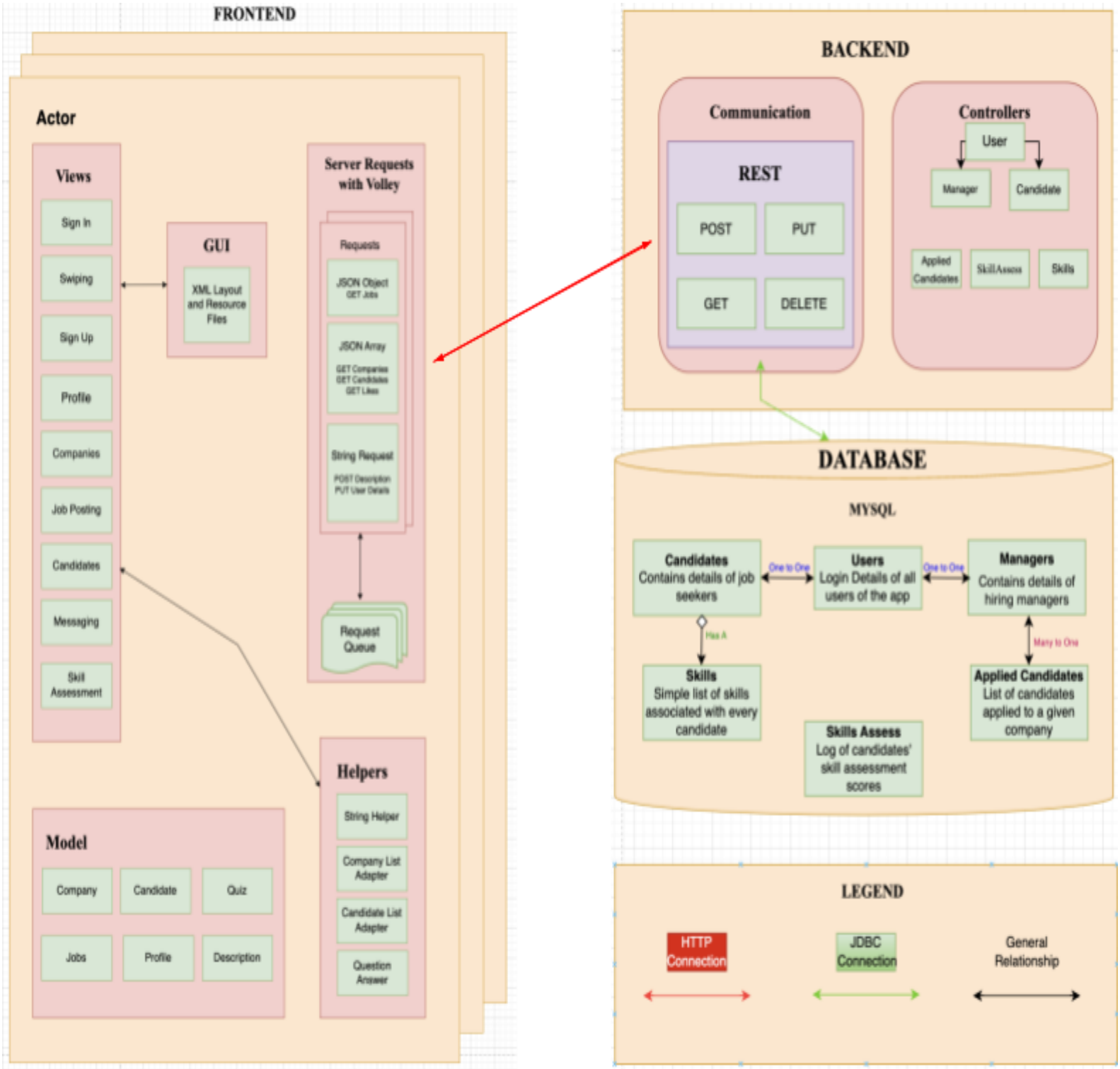
Akhilesh Nevatia: 25% contribution
Hrishikesha Kyathsandra: 25% contribution
Aryan Rao: 25% contribution
Kausshik Manojkumar: 25% contribution

API Documentation

Frontend
1_cw_4/Documents/javadoc/index.html

Backend
http://coms-309-017.class.las.iastate.edu:8080/swagger-ui/index.html#/

SwipeHire Block Diagram



Explanation of Complex Parts of Block Diagram

FRONTEND

The actor communicates with our application via Android views. These views control the user interface and input. Our software has many views to fit the different screens the actor might switch between.

Android User GUI

The Android applications GUI is all handled in the XML layout and resource files. Each activity will have a related XML file that describes how the page should be styled. However, for certain activities that share common instructions, we have constructed helpers, models, and interfaces that will be utilized throughout the activities. The activities will still contain the majority of the logic processing of user commands.

Android Code Helpers

The helper class string helper is used for the email verification that makes sure the user has entered a valid email id. The helper classes Candidate List Adapter and Company List adapters help create the custom lists respective to the candidate and company objects. The question and answer helper class is loosely coupled with the Quiz class, which makes adding and removing questions or adding tests of new skills to the app very simple.

Android Models

The Candidate and Profile classes handle the user's information and handle its logic. The Company, Description, and Jobs class handle the information of the Companies and their job postings. The Quiz class deals with the business and logic of the skill assessments and handles the users' scores and the type of assessment they took.

Android Communication

We are using the Volley library for almost all of the server and client communications. So far, we have mainly conducted JSON object requests, JSON array requests, and String requests using the CRUD specifiers to achieve full round trips between the client and the server. We will now be using web sockets to implement the chatting features and other features that require constant communication between the back end and the front end.

BACKEND

Communication

The Backend uses mapping to update databases based on information sent on the given URL. The mappings include :

Post: This action involves sending information about an item to be added to the database.

Get: By requesting information from the database, often with a specific identifier for the item requested, and this action retrieves information.

Put: This action involves sending information to update a specific item in the database.

Delete: This action sends an identifier to delete a specific item from the database.

Controllers

Controllers contain the mapping for communication between the backend and frontend of the app.

These include:

• **User**: Contains the above mappings to create users (fields: username and password), which contain one-to-one relationships with the below Candidate and Manager type users.

• **Candidate**: This adds a few candidate-specific fields, including the option to add their major, a bio about them, and their skills, and is connected by the userid/email-d to the User table.

• **Skills**: This is one of the fields of the Candidate table that stores the skills a candidate possesses.

• **Managers**: Similar to Candidates, it adds other fields such as jobs, the company they hire for, and information about the position/job they hire for(like salary, job location, etc.). This is also connected to the User table by means of userid/email-id.

• **Applied Candidates**: This has a *many-to-one relationship with the Managers table*, where

candidates can apply to companies, and the managers of every company can see all the candidates who have applied to that given role that a given hiring manager hires for.

• **Skills Assess**: This is a separate entity that deals with storing information of a feature of our App called Skill Assessments. This stores the skill assessment scores of all candidates for all different skill assessments present on the app.

SQL TABLE RELATIONSHIP DIAGRAM