PROJECT REPORT

CS 6235 FALL 2014

REAL-TIME SYSTEMS

EZFAIR

An efficient career fair

Georgia Institute of Technology

Kapil Agarwal

Anantha Nithya Arunachalam

Priyanka Ganesan

{kapila, ananthanithya, priyanka.ganesan}@gatech.edu

TABLE OF CONTENTS

MOTIVATION

Just a month into the semester, and students find themselves juggling from company to company, to land a dream internship or full-time opportunity. Companies may miss out on some extraordinary talent only because the candidate was not able to visit their stall at the Career Fair, thanks to the never-ending queues. We aim to simplify this process by taking the career fair queue system to the virtual domain.

The project also gives companies a mechanism to evaluate potential candidates on the spot and segregate a select few candidates whom they consider to be positives. This is done by a feature on the application which allows students to upload their resume to the server, which can then be accessed and saved by recruiters if necessary.

GOALS

The project seeks to meet the following objectives:

- To implement a virtual queue system which would reduce time spent physically standing in line
- To facilitate digital sharing of resumes
- To provide recruiters a method to shortlist candidates based on resume

These ideas combined would be a powerful tool at college career fairs to smoothen the process of information exchange.

ARCHITECTURE

1. SYSTEM ARCHITECTURE

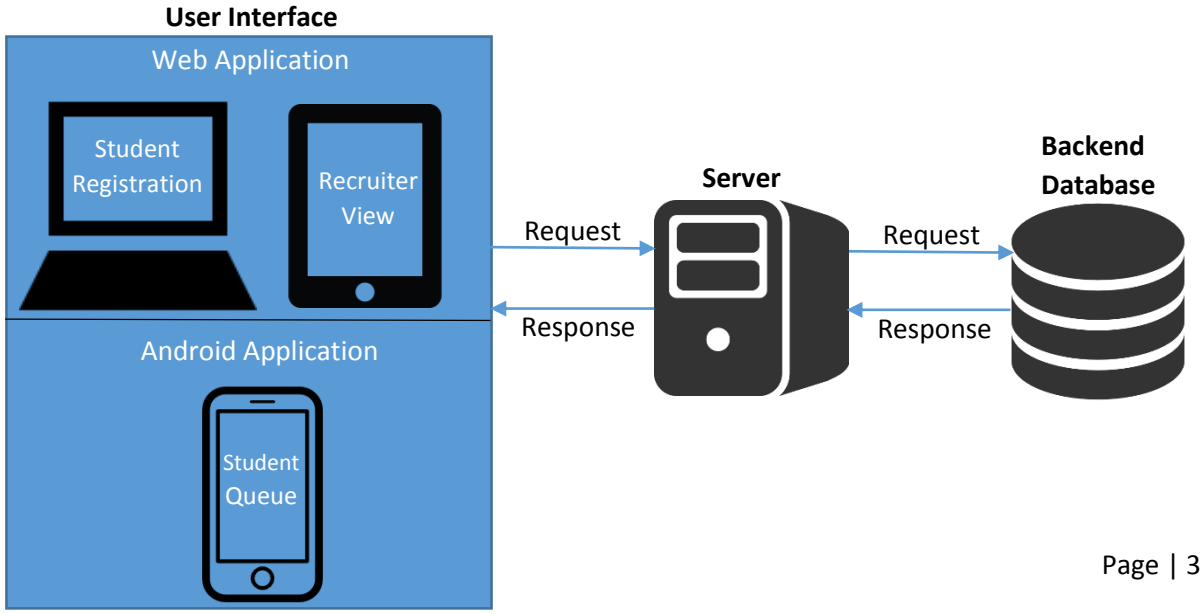


Figure 1: System Architecture Diagram

2. USER INTERFACE MODEL

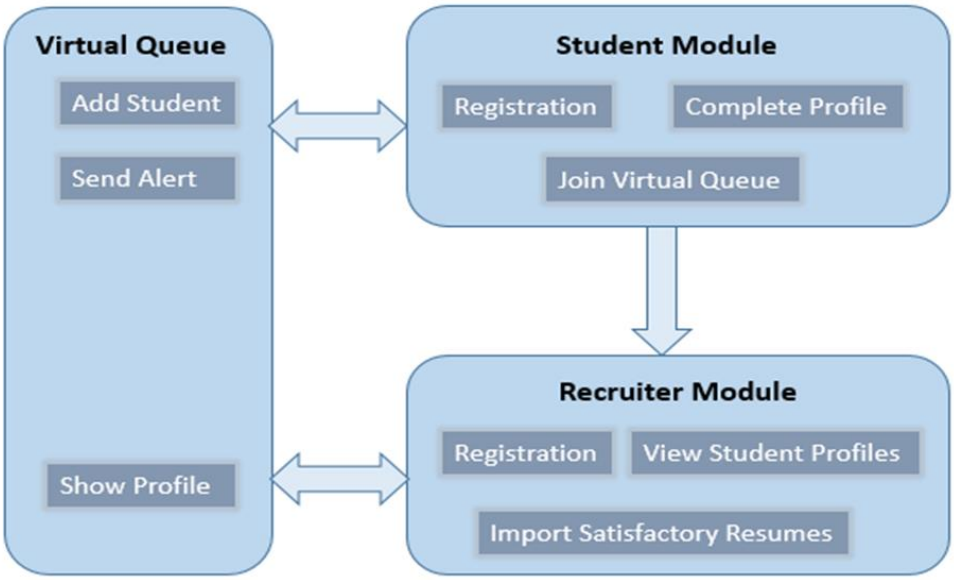


Figure 2: User Interface Model

DESIGN

1. FEATURES

The EZFair application can be used by both students and recruiters and there are several features that can help make the career fair more efficient by reducing the chaos that generally takes place.

1. Resume and profile upload
   The students are able to fill in their profiles and upload their resume before coming to the career fair. This helps students as they don't have to carry multiple copies of their resume to the career fair.

2. Join virtual queues
   Currently, students have to stand in long queues, potentially for several hours and yet attend only a few companies in a day. Using the EZFair android application, the students can join the virtual queues of the companies they wish to attend during the career fair. They get a token number when they join a queue from and they also know how many people are ahead of them

in the queue. They can also leave a queue at any time if they do not want to meet the recruiter from that company.

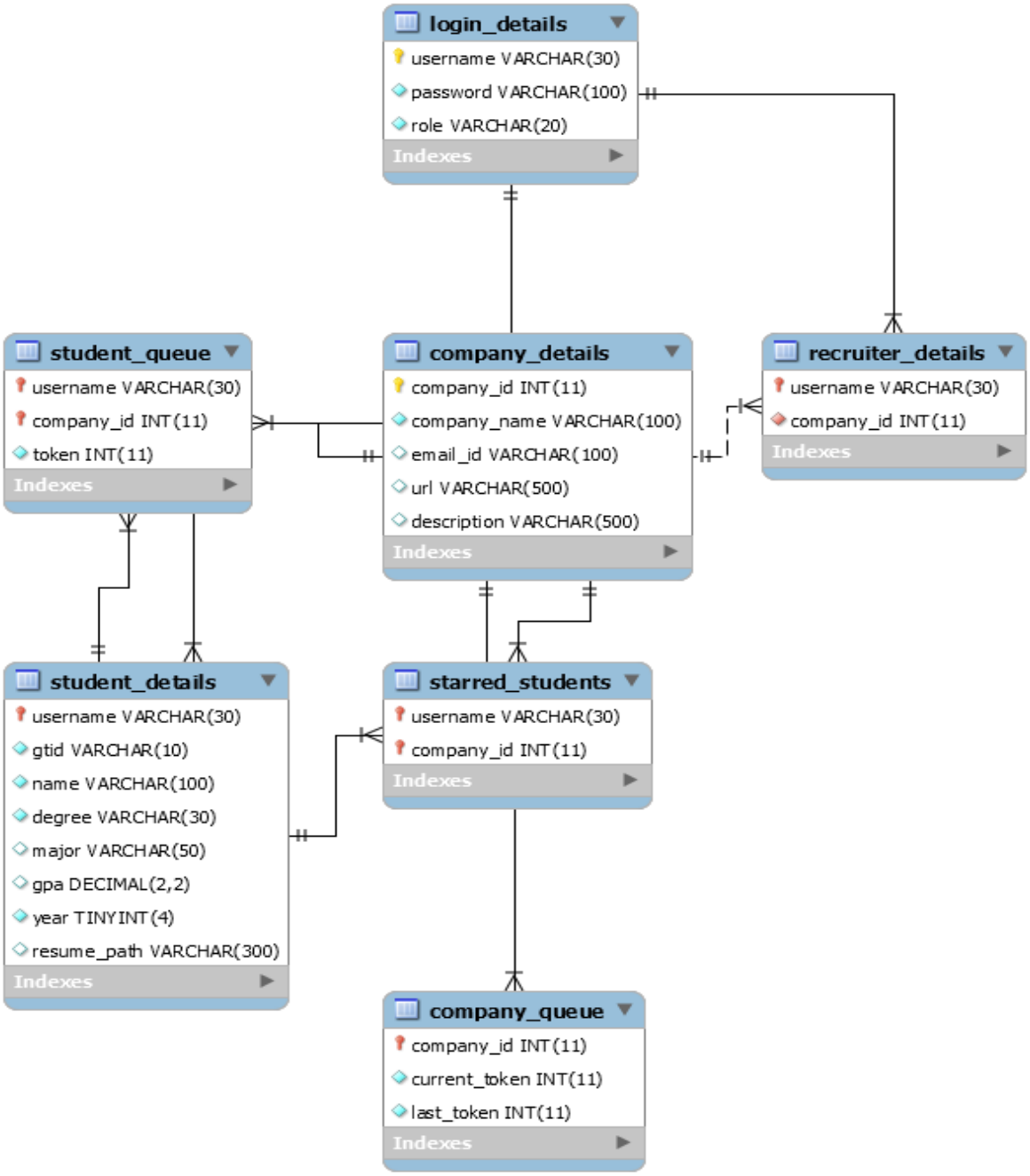
3. Notifications
   The students are notified about 15 minutes before their turn is about to arrive, and then they can go and join the company's queue physically. This saves a lot of the students' time and gives them a chance to attend more number of companies. The students can also decide to leave the queue at the time of notification if they feel that they will not be able to make it or are no longer interested. This will move the student out of the company's queue and another student will take their place

4. Student profile views
   During a career fair, a recruiter has only a minute or so to peruse through a student's resume and has to learn as much information about their interests and skills during a short talk. This is not an easy task and recruiters generally look for some basic criteria like degree, major, GPA, etc. Using EZFair, they can view these details directly on their tablets which helps to make the conversation with the student more concise and productive.

5. Student resume views
   Recruiters collect hard copies of resumes of interested students which they have to manually sort later. Sometimes, they even scan or take photos of the resume and upload them to the company database. EZFair provides them with the digital copies of the resumes for direct download which saves paper and time.

6. Star good candidates
   It is difficult for a recruiter to remember all the students they might have talked to during the career fair. There might have been some students who the recruiter liked and would like to know more about them. EZFair provides recruiters with an option to mark students as *Starred* which can be viewed later again. These students can then be contacted by the recruiter for proceeding further in the process.

2. SCHEMA DIAGRAM



3. FLOW DIAGRAMS

We have three main flows in the system.

1. Student Registration (Web Application)

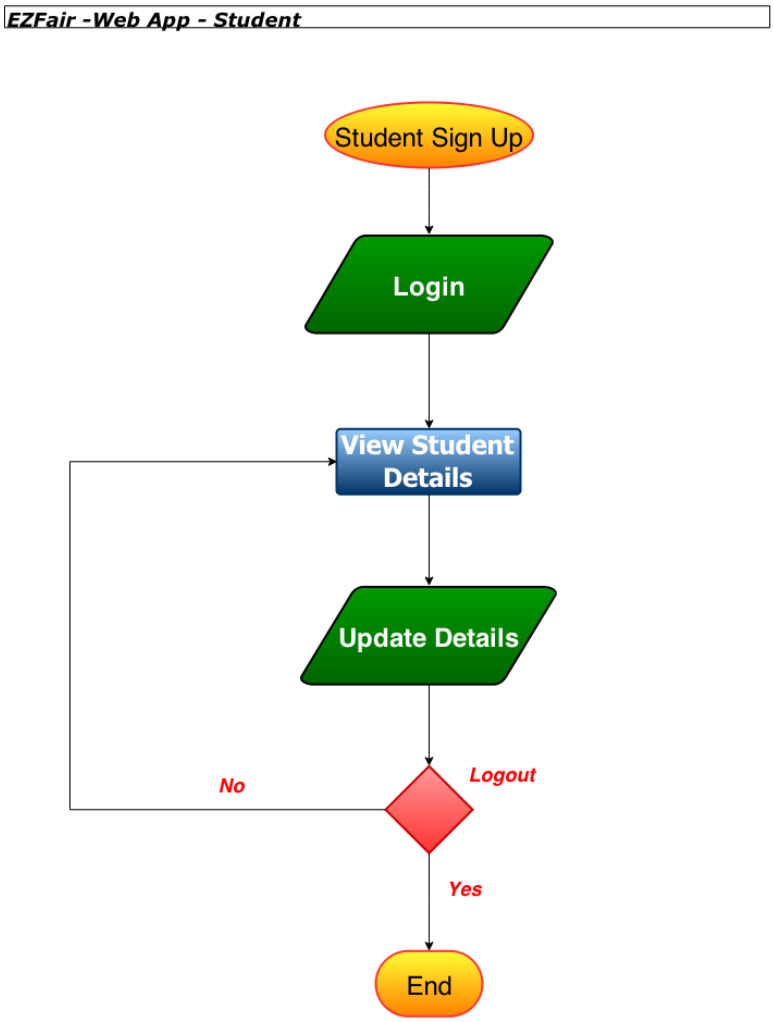


Figure 4: Student Web Module Workflow

The student first signs up with the system to get a username and password. Upon successful login, students can update their profile by providing their Major, GPA, and also upload their resumes in the system. These details can be seen by the companies prior to the career fair.

2. Student Queues (Android Application)

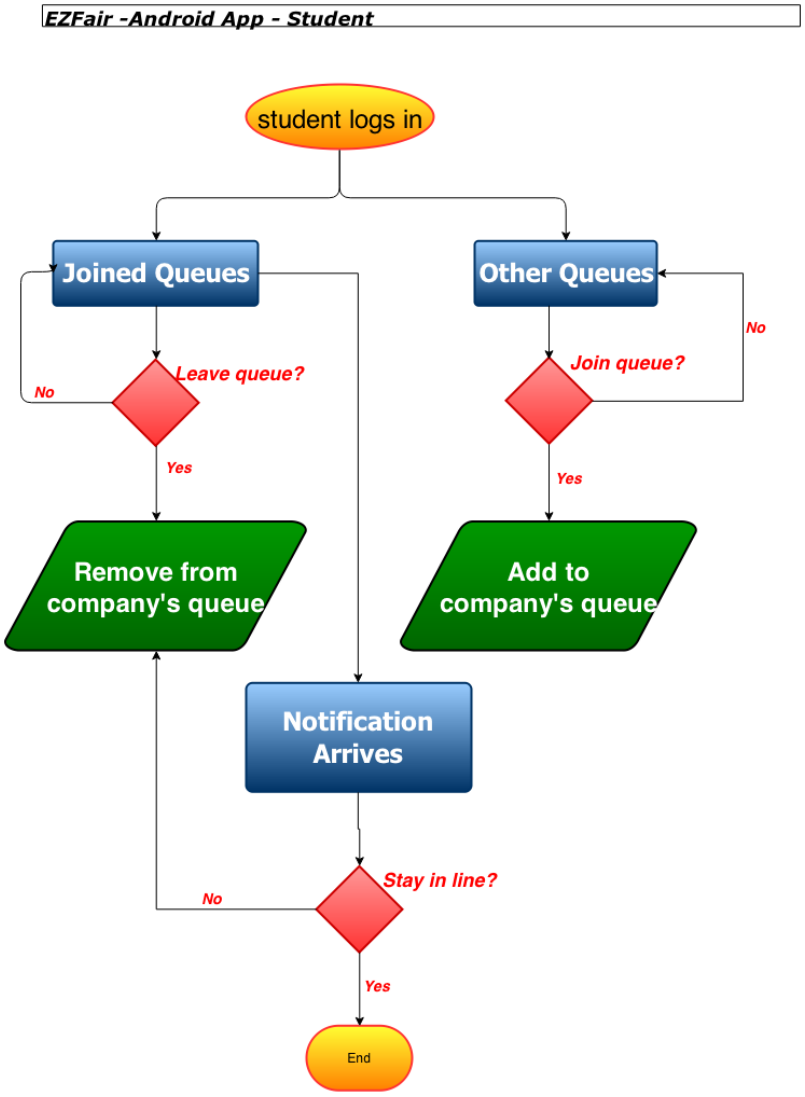


Figure 5: Student Android Module Workflow

The student signs in to the Android application on the day of the career fair and views the list of companies visiting the career fair for that day. The student can click on the company and join its virtual queue. Once joined, the student can see the company listed in the "Joined Queues" section. The companies which the student has not joined are listed in the "Other Companies" or "Other Queues". Once a student has joined a queue virtually, there is an option by which he/she can voluntarily leave the queue.

The student gets alerted (via a notification message) about 15 minutes prior to his/her turn at the head of the queue. At this point too, the student can choose to leave the queue. If not, this notification is the cue for the student to join the physical queue in front of the company's stall at the career fair.

3. Recruiter View (Web Application)

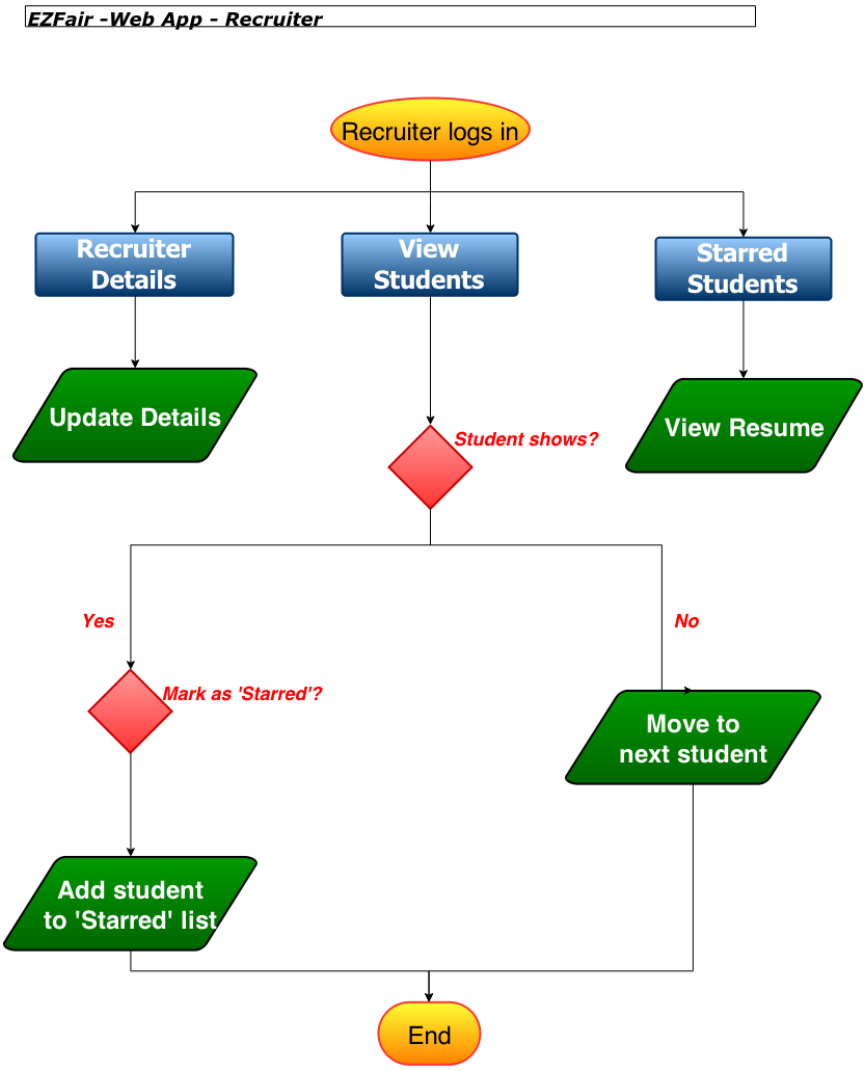


Figure 6: Recruiter Web Module Workflow

The recruiter signs up with the system and gets a username and password. Once the recruiter logs in, there are three options available. The recruiter can choose to update the company details like email address, or job description. Alternatively, the recruiter can view the students and their profiles in the company's virtual queue. If the student does not show in the physical queue, the recruiter can move on to the next student in line. Or, if the student does show up, and the recruiter was impressed after talking to the student, the recruiter can mark that student as "Starred". Later the "Starred" students' resumes can be imported to the company's database for further review.

IMPLEMENTATION

1. ANDROID

In our Android App, we have used 4 Activities and 2 services. There is a Login Activity, and after successful login, the username is stored in a SharedPreferences library and hence login is not needed for consecutive use of the app. The app then starts another Activity which uses an Asynchronous Task to connect to the database to retrieve the list of companies which the student has not joined. The companies are displayed using a ListView, which uses a dynamic Adapter to feed data to the list. The number of people already in the virtual queue is displayed for every company. The student can select one or more companies from the list, and choose to join the corresponding virtual queues. Once the student clicks Join, the student can view the Joined Queues in another Activity, on the click of a button in the Action Bar. The student's token number and the number of people ahead in the queue is also displayed for every joined queue.

Further, the student can select one or more companies from the 'Joined Queues' list and choose to leave them. In this case, the companies will get added to the 'Other Companies' list (i.e. the list of company queues which the student has not joined).

The app also periodically queries the server for the number of people ahead in each of the virtual queues that the student has joined. We assume (based on experience) that each recruiter talks to a student for about 5 minutes. So when there are three people ahead of the student in the virtual queue (who have confirmed that they will be joining the physical queue), a notification will arrive for the student to join the physical queue. Here, the student is again given an option to leave the virtual queue or to confirm that he/she will be joining the physical queue.

We have two services to handle the action to join the physical queue or leave the virtual queue from the notification. These services also contact the server to perform the action of leaving the virtual queue or confirming to join the physical queue.

2. WEB APPLICATION

The web application part of EZFair consists of two parts. First is the student module and the other is the recruiter module. The user interface has been developed using HTML, CSS, Javascript, Jquery and AJAX technologies. Also, the web pages are dynamic in nature i.e. they are not hard-coded as static HTML pages. We have used PHP for the backend development which dynamically generates the web pages to be rendered. First, there is the Sign-up page and the Login page through which the user registers into the EZFair application and then can login to view their details. HTML forms have been created that send GET/POST requests to the server which if return successfully lets the user proceed to the next page.

We have assigned a role to every user which can be either *student* or *recruiter*. Based on the role and if the login credentials are validated, the user is taken to the student or the recruiter module. In the case of student, the student can now view or update his personal information and upload his resume. Currently, the upload limit is set to 2 MB. The student can also view his resume online which is displayed as a HTML5 embedded object. In case of recruiter, the user can update the company information and on the day of the career fair, they can see the list of students present in the queue and mark them as *starred* if they like a candidate. Also, the recruiters only see those students that are eligible to join the queue physically based on a heuristic that on average, a recruiter talks to a student for about 5 minutes and a threshold of 15 minutes required by a student to join the physical queue. In case a student does not turn up despite being in the queue, the recruiter can mark them as *No Show* in which case the student is removed from the queue and another student is selected to join the physical queue. As we don't want the recruiter to refresh after every student, we have used AJAX to make requests to the server that allow asynchronous calls and on return, the queue is updated.

3. BACKEND

There are two parts for the backend, one is the backend for the web application and the other is the backend for the android application. The backend is developed using PHP programming language and MySQL database. The requests are received as GET/POST and the server does the processing like database transactions on the inputs to generate an output that is returned to the web page or the android app. Data is communicated between the backend and the android app using JSON data structure.

Now, we describe the implementation of the virtual queue system in the backend. There is a queue for every company registered for the career fair. We maintain two variables two describe a queue - the current token number i.e. the token number at the front of the queue and the last token number i.e. the token number at the end of the virtual queue. Every time a student joins the virtual queue of a company, the last token number gets incremented and assigned to that student. We use PHP transaction management to implement MySQL atomic instructions to avoid any data races that may arise due to two students joining the same queue at the same time. When a student leaves a queue, the token number assigned to the student is invalidated. However, no change is done to the current token and last token needed to maintain the queue as the next student who joins will just get the next token number, thus assigning a unique token to each student for a company. As we can see, after a student has left the queue, there can be voids between the current token and last token. So, we have used the *count ahead* metric which is the number of people that are currently ahead of a student in the queue rather than just the last token minus the current token. Depending on the number of recruiters and the average heuristic of 5 minutes per person per recruiter, we display those students whose token numbers are in the range of current token and current token plus three times the number of recruiters. These students also get a notification as they are the ones whose turn is next to join the physical queue.

The JSON structure returned to display the company queues on the android application:


```
{
   "joined_queue": [
       {
           "company_id": "1",
           "company_name": "Microsoft",
           "token": "14",
           "count_ahead": "0",
           "num_recruiter": "1"
       }, {
           "company_id": "2",
           "company_name": "Google",
           "token": "5",
           "count_ahead": "0",
           "num_recruiter": "2"
       }
    ],
   "other_queue": [
       {
           "company_id": "3",
           "company_name": "Facebook",
           "count_ahead": "0"
       },
       {
           "company_id": "4",
           "company_name": "Amazon",
           "count_ahead": "0"
       }
   ]
}
```


Figure 7: JSON Queue Response from Server

4. SERVER

We have used an Apache/2.4.4 server running on a HP Pavilion dv6 64-bit machine with Ubuntu 12.04.5 LTS operating system. We have also used MySQL version 14.14 distribution 5.5.40 and PHP version 5.3.10-1ubuntu3.1.

RESULTS

The application reaches the goals it set out to solve. The results are depicted using the screenshots shown below.

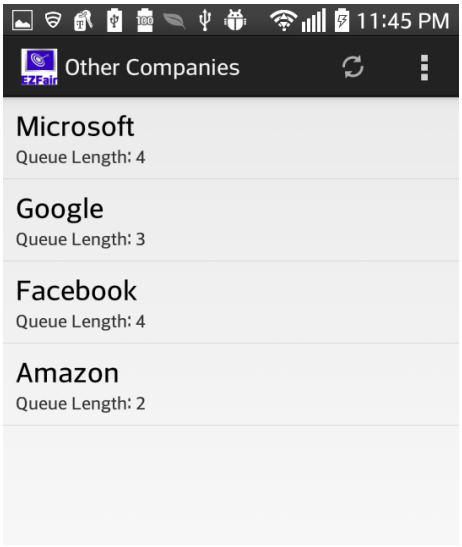


Figure 7(a): Companies not joined

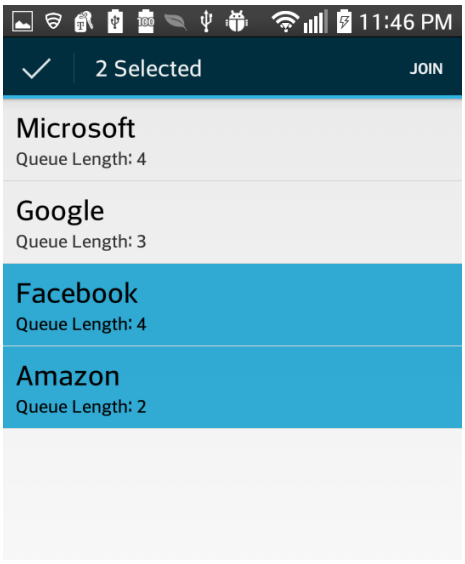


Figure 7(b): Student joins 2 companies

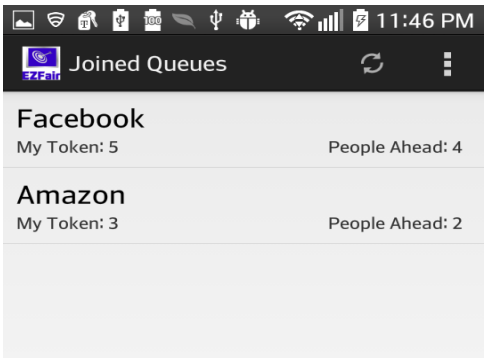


Figure 7(c):Student's Joined Queues

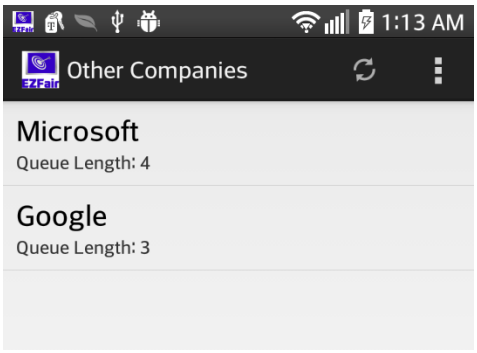


Figure 7(d):Student's Other Queues after joining two companies

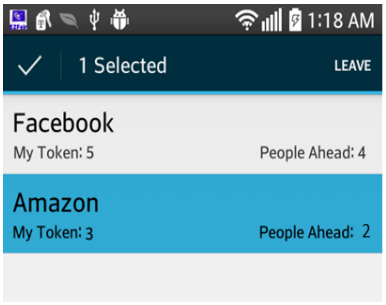


Figure 8:Student leaves a joined queue

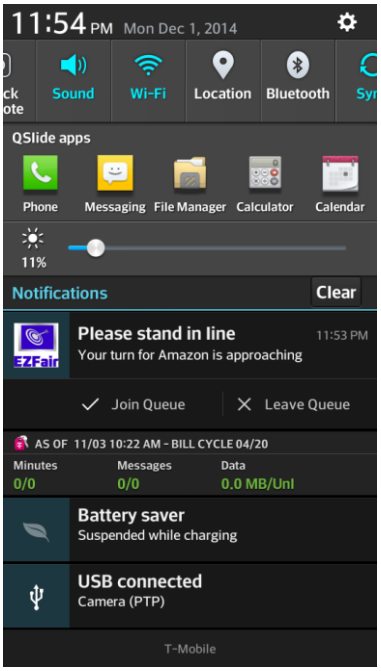


Figure 9:Student gets a notification to join the physical queue

FUTURE WORK

Our proposed system attempts to solve the problems of long queues during the career fair and reduces the waiting time for the students so that they can meet with more number of recruiters during the available time. Also, our system helps the recruiters to get better information about the candidate apriori leading to better interaction. The scope for improvements in this system include:

1. Extending the app to allow recruiters to schedule interviews with students

2. Applying data mining techniques like clustering on the information provided by candidates to form groups based on parameters like skills, specializations, or education. This can help the company officials filter out candidates based on the requirements of the open positions available in the company.
3. Exploring different algorithms for more efficient queue maintenance
4. Integrating students' resumes with their LinkedIn profiles

REFERENCES

[1] GT Career Fair Plus App
https://play.google.com/store/apps/details?id=com.careerfairplus.gt_ga

[2] QLess
https://play.google.com/store/apps/details?id=qmanager.qless.android.ap
https://play.google.com/store/apps/details?id=qkiosk.qless.android.app

[3] Android Application Development
http://developer.android.com/index.html

[4]WAMP server
http://www.wampserver.com/en/