Placement Cell Application

Final Project Report

Anchit Gupta	Anand Dhoot	Shubham Yadav
13D100032	130070009	130050003

November 8, 2015

1 Introduction

We have made an end-to-end application for the college Placement Cell. We planned to make the new one from scratch and we have worked to the best of our ability to make it, given the time frame and resources. We have tried to make our application as complete and deployment-ready as possible

This report highlights the important aspects of our system.

2 Sample Data

We used a large dataset consisting of as many as -

- 500 Students
- 50 Coordinators
- 50 Companies
- 50 Seed JAFs

The Dataset we used was sourced from a freely available online toy dataset. The data was modified and expanded to our needs using Excel.

In the submitted files, the data is present in appropriate csv files. Scripts were made to populate the database using these files.

3 Overview of users and functionality

We mainly have three type of users - Students, Coordinators and Company representatives. Main functionality of each of these users is as below.

3.1 Student

All students get registered by default at the beginning of the academic year by an administrator. The students can update their basic details and upload their resume for verification.

Once verified, every student can see the list of Job Application Forms (JAFs) that they are currently eligible for. A student can also view the current status of all JAFs she has applied for. In terms of functionality, every student has the following interfaces

- Resume Upload / Download
- Edit Personal Details
- View Open JAFs
- Sign Eligible Open JAFs
- View Signed JAFs

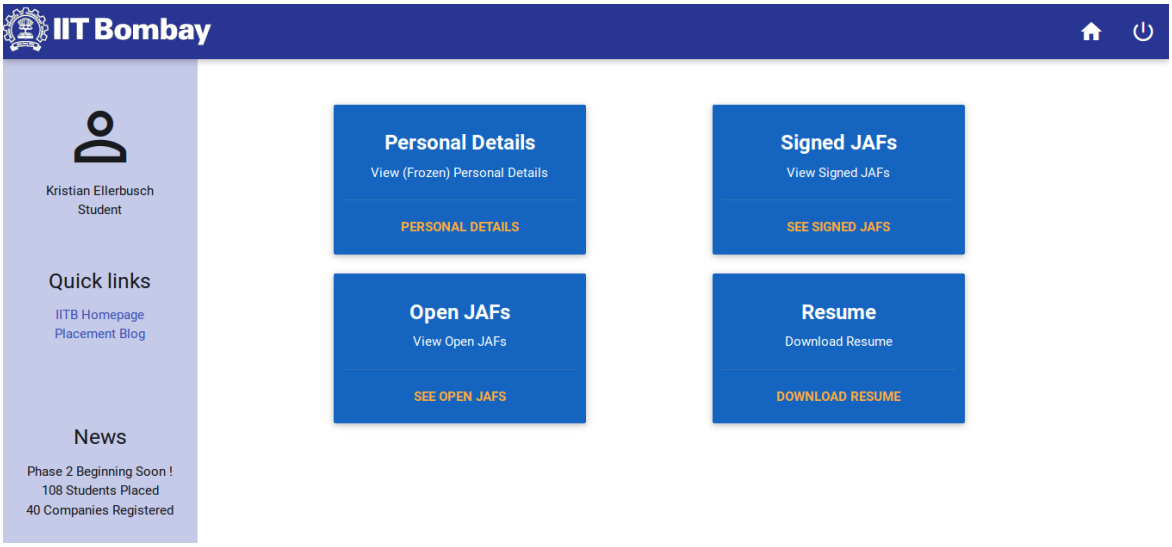

Figure 1: Student Interface

3.2 Co-ordinator

Each Coordinator is allotted a fixed number of students and companies by the administrator and would be responsible for them.

More specifically, a coordinator can see the list of students under his supervision and can approve students which whose resumes get verified, etc.. Also, the coordinator has

the responsibility of verifying details filled by companies in new JAFs and approving them.

Every coordinator has interfaces to -

- View Resume and profile of students allocated
- Verify Students
- View JAF Details & Verifying the same

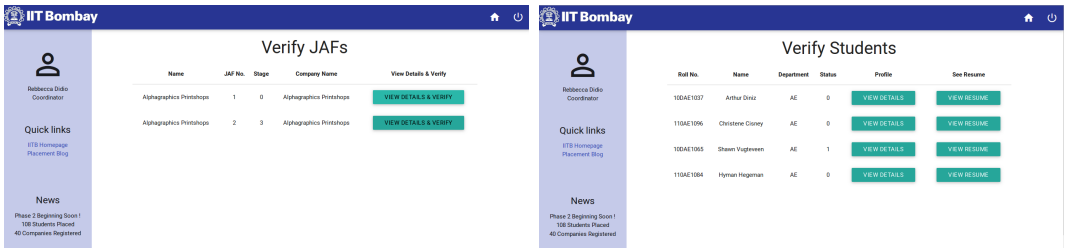

Figure 2: Coordinator Interfaces

3.3 Company

A company has the important role of creating a JAF and selecting students in further stages of the recruitment process. Once, the JAF created gets approved by its corresponding coordinator, the company can see the list of all students who have applied for the position and select/reject them for subsequent stages.

A company representative has the following interfaces

- Create new JAF
- View Students That Have Applied to open JAFs
- Recruit Students

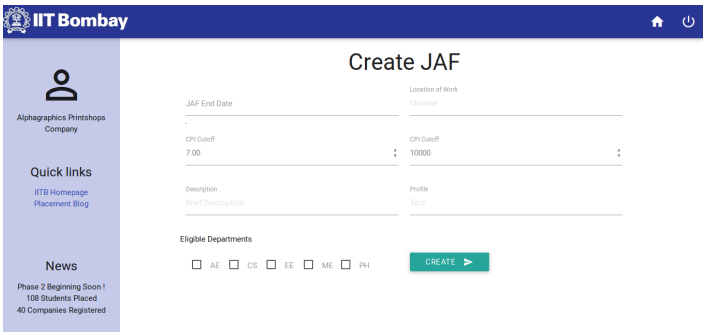

Figure 3: Company Interface to create a new JAF

4 Salient Features

4.1 Authentication

A lot of time was spent to meticulously ensure that adventurous users are not able to exploit the system by doing hacks & tricks like Inspect Element and changing URL parameters. Placement Cell Interface, being a Mission Critical application, ensuring security features and no foul-play by all parties is essential.

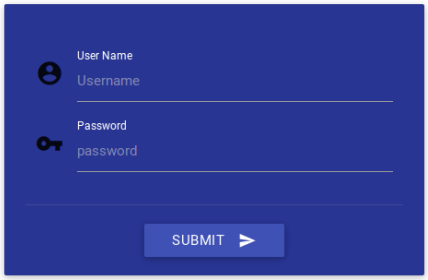

Figure 4: Login Page

This was ensured by using sessions to store critical parameters and putting authorization checks using filters every time data is served or updated in the database. We implemented a "logout" feature as well so that users can securely exit the system without the headache of any security risk.

4.2 Placement Cell Rules

As a part of this application, we have taken care to implement these constraints of the PT Cell, which are not explicitly a part of the database schema such as -

- A student can get selected for at most one job i.e. if he gets selected by a company, we needed to make sure he can neither apply to, nor be selected by any other company from that moment onward.
- After getting approved, a student can neither edit his profile nor upload his resume.
- A student can only apply for a JAF only if $CPI > CPICutOff$ and if his $Department \in Depts.Eligible$

4.3 Sorting data in frontend dynamically

We wanted to provide as much flexibility to the user as possible in terms of making decisions from the data available. We have implemented the stable sort algorithm with

the "sorttable" javascript library.

This not only enables us to dynamically sort any table on any column, but also enables sorting on multiple attributes.

4.4 Storing Resumes in PDF format

This involved storing and fetching PDFs from the database. Not being familiar with the many utilities, we needed several packages to accomplish this.

We decided to store PDF's into the Database as a byte array. We converted the PDF into a byte array for storing and converted the byte array into a PDF during fetch. This needed the usage of the apache.commons library functions. Thus, this seemingly straightforward functionality took quite some time to get just right.

5 Libraries used

5.1 SortTable

This is used to make tables sortable. This is a very important functionality which will be demanded by most companies. This makes possible things like:

- Sort According to CPI
- Group by Department
- Group and Sort
- Sort by Name etc.

Initially we had thought of using SQL directly to answer these queries. But we realised that this was a highly inefficient as this would require a huge latency in extra database calls along with the Network Overhead for large sizes. A good alternative to this is to do client side sorting.

5.2 Materialize CSS Bootstrap

This has been used to provide a good UI. This is a library which closely follows Google's Material Design Philosphy which is both minimalist and feature rich. Some elements to note from this library

- Buttons- Raised Buttons with fluid click animation
- Consistent Color Scheme along with icons to match
- Text Boxes with intuitive display of labels and default values

6 Processing and Interfacing

Below given is a brief description of the various classes and JSPs implemented.

6.1 JSPs

- Main Login page - Auto detects user type. From the username, our application automatically detects whether the user is a student, coordinator or a company representative.
- Landing pages - Depending on the type of entity "AuthRedirect" will open the corresponding landing page, depending on the type of user. The three landing pages are -
  - Student landing page - contains the form for data entry, Summary of Signed and Eligible JAFs, and the relevant data about those JAFs and upload / download utility for Resumes
  - Company landing page - contains the JAF entry form, Student filtering(CPI or selection basis) and the selection page for the selected students. Also ability to Download Resume of Student's which apply to it
  - Coordinator landing page - contains information regarding a Student and the verification pages. Also has the ability to download Resumes of applied students for verification.
- Result pages for the three types of users. These pages display the details of the queries requested from the landing pages.

6.2 Other important packages

- Dbutils: This package contains all utility functions to maintain the connections regarding the DBMS and maintaining SQL related queries like update, insert, etc...
- UI: Functions to take care of user interface on various JSPs. We made it centralised so editing the UI is easier and faster.
- Authorization: This contains filter and redirect servlets which implement all the logging in and authorization capabilities. The filter ensures all pages are secured and one entity can not access pages pertaining another entity type.

6.3 Frontend

We followed material design guidelines to make the UI look pleasant to the eyes and intuitive.To make the UI feel fresh and modern.

- Uniform color palette
- A navigation bar at the top with intuitive Home and Logout Buttons

- Side bar containing custom information for each user on the left, containing the name, image, entity type. This also contains news and some live statistics regarding the Placement Cell
- Fast and fluid animations.
- User Friendly Form with a graphical flair

References

[1] http://www.kryogenix.org/code/browser/sorttable/

[2] http://materializecss.com/

[3] http://www.java-tips.org/other-api-tips-100035/69-jdbc.html

[4] https://commons.apache.org/proper/commons-io/