Placement Cell Application

Detailed Design Report

Anchit Gupta 13D100032 Anand Dhoot 130070009 Shubham Yadav 130050003

October 19, 2015

1 Introduction

This design report lists down the various design aspects of our database and forms the base for further development and making the complete application. This reports contains both the work which we have already done and things we plan to do. While most of the design aspects get covered in this report, in case some minor things get left out, they would be added to the database/interface design as needed.

2 Approach

We first list down the entities and their corresponding attributes which would constitute our application. Subsequently, we list the relationships between the entities and establish their cardinality. This would involve setting up primary and foreign keys, adding weak entity sets and other components of the Entity-Relationship diagram. Finally, we reduce our ER diagram to a Database Schema. Then from the Schema we obtain the DDL Code using some well-known tool.

Apart from designing the database schema, we would have to design several other aspects which include, but are not limited to

- Ensuring there are no data redundancies in the Final ER-Diagram. This was ensured by verifying that our model is in **BCNF**. Initially the model we had thought of didn't have a separate Application table, but after applying the BCNF Rules we were able to split our JAF Table into two tables to avoid data redundancies.
- Expressing various constraints such as check, primary-key, foreign-key, etc. using triggers or other alternative methods.
- The cardinalities of each relation will be needed to be identified according to the semantics of the application. These are very important as they will decide how the schema will look like when we reduce it.

3 ER Diagram

3.1 Entities and Attributes

Our application can naturally be thought of as a interaction between the following four entities. A few relevant Attributes have been listed for each entity. Refer the Schema for a detailed listing of all attributes.

Entity	**Attributes**
Student	Name, E-mail ID, Password, Roll No., Mobile No.
Company	Password, Category, Name, Company ID
Coordinator	Name, E-mail ID, Password, Coordinator ID, Mobile No.
Job Application Form	Description, Salary, End time, CPI cut-off, Eligibility

3.2 Relations

We basically have 4 relations.

1. Applies between Student and a JAF
2. Opens between Job Application Form (JAF) and Company
3. Manages between Company and Coordinator
4. Represents between Coordinator and Student.

These relations along with their respective cardinalities have been shown in the ER Diagram.

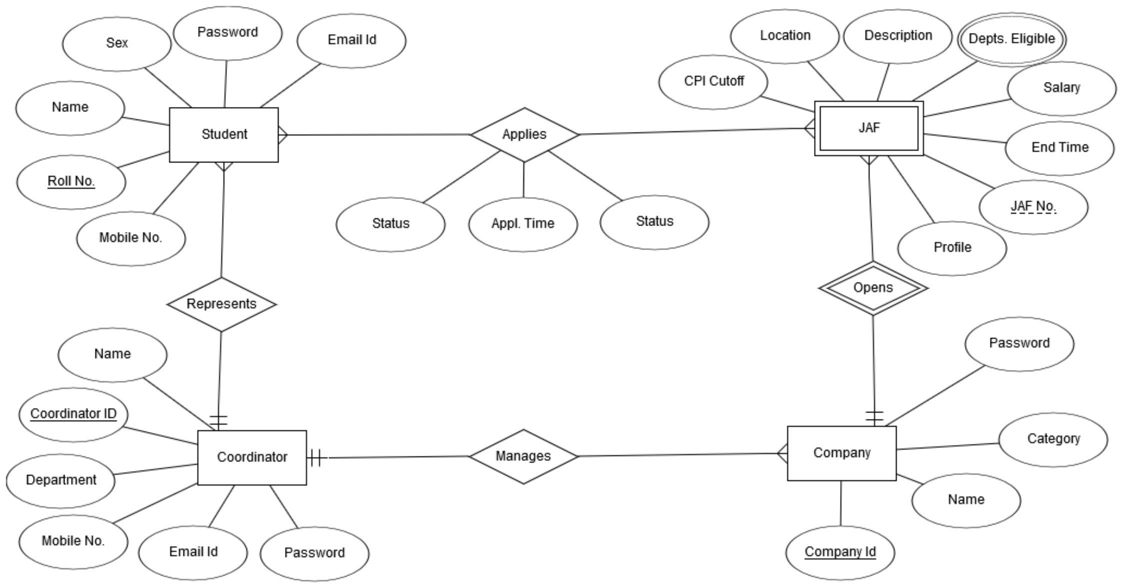


Figure 1: ER Diagram

3.3 Constraints

These are some extra constraints that are not part of the ER/Schema . Enforcing these are critical for our application. These will be enforced either directly through SQL triggers etc, or through the JDBC code. Some of these include

1. A student can get selected for at most one job.
2. Student can only apply for a JAF only if $CPI > CPICutOff$.
3. Student can only apply for a JAF only if $Department \in Depts.Eligible$
4. To make the login process user-friendly we have a single login page for all three types of people. Our backend JAVA code detects which type of user is trying to login. This setup naturally assumes that the UserIds of users across all three tables (Company, Student, Coordinators) are distinct.

   In practice this can be easily ensured by setting the UserID of a student as his Roll. Number and the remaining two follow a distinct naming scheme.

4 Database Schema and related aspects

4.1 Tables

We create tables for individual entities - Students, Company (Representative), JAF, Coordinator. In case of relations, we create a table for the Application relation because it is many-many (other one-many or one-one relations get handled by the foreign key constraints) and also has some extra attributes.

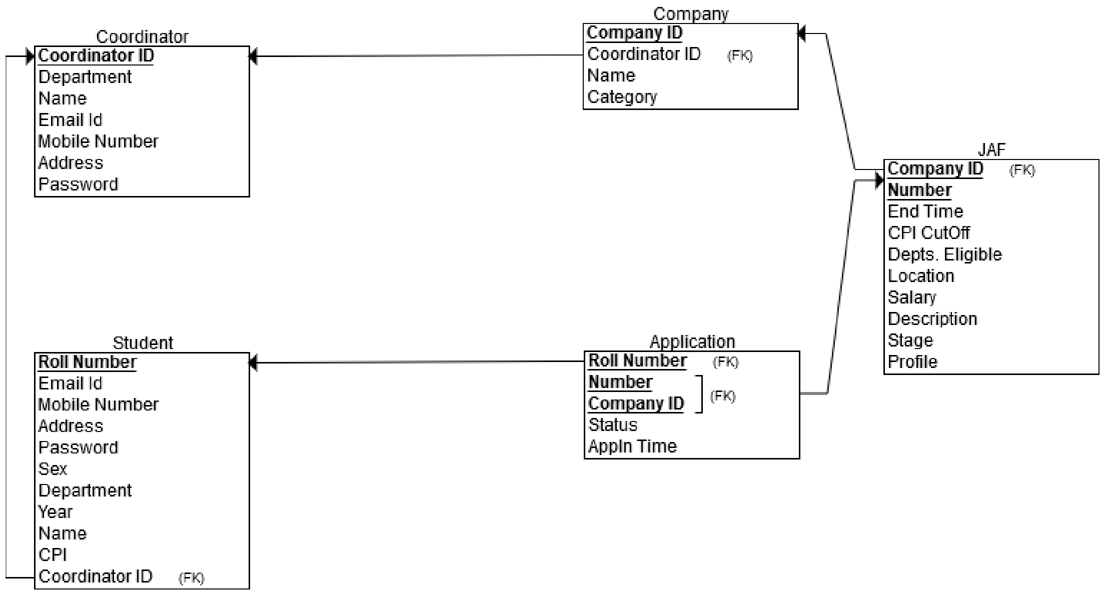


Figure 2: Database Schema

4.2 Users and Access Control

We realize that the Placement Application is a critical application and it is completely responsible for our future once we graduate. This means that every record is critical. Hence, we make two users who can login to the database -

- The database administrator - Who has access and modify permissions for the entire database and can delete any record or table.
- The application user which has only access (and possibly updation) rights of the database.

This is done to ensure that even by mistake, there is no possibility of the application deleting any data from any table in the database.

4.3 Indexing

In our application, there would be a large number of records in the tables JAF and Application. We believe that primarily, this would be the bottleneck of our application. Hence, to speed up accessing, we create indices on the primary key(s) of these tables. Specifically, we create indices on

- (Company ID, Number) for the table JAF. This ensures fast lookups of JAFs from a specific company.
- (Roll Number, Company ID, Number) for the table Application. This ensures fast lookups of JAFs signed by a specific student.

5 Processing and Interfacing

The Servlets and classes are what we have decided so far and are subject to change to improve the efficiency and functionality of the application.

5.1 Servlets

- Main login page - auto detect login type. From the username, it will automatically detect whether the user is a student, coordinator or a company representative
- Landing pages- If the password matches the password of the username, it will open the corresponding landing page, depending on the type of user.
- Student landing page will contain the form for data entry, Summary of Signed and Eligible JAFs, and the relevant data about those JAFs
- company landing page will contain the JAF entry form, Student filtering(CPI or selection basis) and the selection page for the selected students.
- Coordinator landing page will contain information regarding a Student and the verification pages

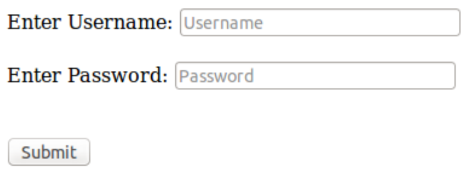
5.2 Other important classes

We have 3 classes, namely Student and company, Coordinator to deal with their relavant operation logic. These classes will implement all the tasks needed like managing data related to JAFs, student data and its submission in time, and company data like which JAFs they are holding, approval of JAFs , student profiles by representatives etc.

Regarding security, we made a login class to deal with detecting type of user(student, co-ordinator or company representative) his authentication based on his username and password and subsequent redirecting to the relavant landing page.

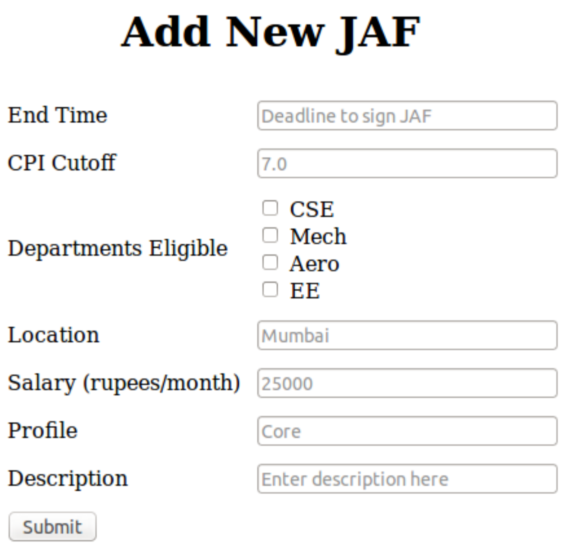
5.3 Frontend

Basic bare-bones interface for some pages are as below. Please note that this is a basic version without CSS and we intend to use standard libraries such as Bootstrap to provide an intuitive UI.



Enter Username:	Username
Enter Password:	Password

Figure 3: Login Page



Add New JAF

End Time	Deadline to sign JAF
CPI Cutoff	7.0
Departments Eligible	☐ CSE ☐ Mech ☐ Aero ☐ EE
Location	Mumbai
Salary (rupees/month)	25000
Profile	Core
Description	Enter description here

Figure 4: Adding a new JAF

6 Testing

We plan to create a sample data file having a minimum of 50 students, 10 student coordinators and 5-6 companies. Testing would involve running the application (from opening a JAF to final selection) for a company/set of companies. We would also be testing new user registrations, for both students and company representatives.

We also intend to make SQL scripts that would be used at the beginning of an academic year when the entire database has to be reset.

References

[1] https://erdplus.com Tool to Draw ER Diagrams and DataBase Schemas

[2] Database Systems Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan

[3] http://getbootstrap.com/ For making a User Friendly Front-End UI