

CS 387: Database and Information Systems Lab

Design Plan: **Placement Portal**

Monday 4th April, 2022

Contents

1	Team members	3
2	Logical Schema	3
2.1	Schema Diagram	5
3	Integrity Constraints	6
3.1	Primary Constraints	6
3.2	Foreign Constraints	6
3.3	Other Constraints	7
3.4	NULL Attributes	7
4	SQL files: DDL and InsertData	8
5	Possible Transactions and SQL Statements	8
5.1	Common to All Users	8
5.2	Students	9
5.3	Recruiter	10
5.4	Coordinator	12
5.5	Superuser	13
6	Indexes for Query Optimization	14

7	User Forms	14
8	Business Logic Controller	18
8.1	User #1: Student	18
8.2	User #2: Recruiter	19
8.3	User #3: Coordinator	20

1 Team members

1. Aditya Badola (190050006)
2. Akshat Gupta (190050010)
3. Liza Dahiya (190050063)
4. Tanay Sharma (190050122)

2 Logical Schema

From the *entity-relation* model, we derive the following logical schema after normalizing to 3NF (***bolded*** attributes for a table form the primary key):

COORDINATOR
coordinator_id
coordinator_password
coordinator_name
coordinator_email
coordinator_contact

PROGRAM
program_id
program_name
program_duration
program_level

DEPARTMENT
department_id
department_name

PROFILE
profile_id
profile_name
profile_description

COMPANY
company_id
company_name
company_origin
company_coordinator

RECRUITER
recruiter_id
recruiter_password
recruiter_name
recruiter_contact
recruiter_email
recruiter_company

JAF
jaf_id
profile_id
company_id
jaf_jd
jaf_bond_duration
jaf_location_of_posting
jaf_currency
jaf_ctc
jaf_gross
jaf_cpi
jaf_bonus_allowed
jaf_opened_on
jaf_closed_on
laf_slot

STUDENT
student_rno
student_password
student_name
student_gender
student_dob
student_email
student_contact
year_of_enrollment
student_current_year
student_cpi
student_incentive_points
program_id
department_id
allocated_jaf
allocation_timestamp

RESUME
resume_id
resume_url
resume_npages
student_rno

APPLIES_FOR
student_rno
jaf_id
resume_id
date_time

SHORTLIST
student_rno
jaf_id
date_time

OFFER
student_rno
jaf_id
date_time

ELIGIBLE
jaf_id
department_id
program_id

2.1 Schema Diagram

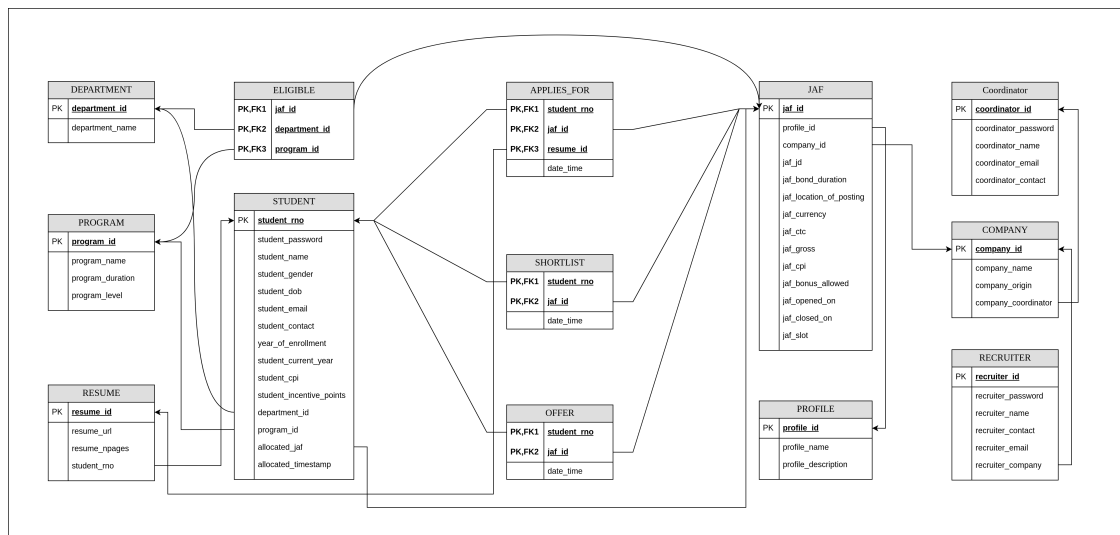


Figure 1: Schema

3 Integrity Constraints

3.1 Primary Constraints

Table	Primary key
COORDINATOR	coordinator_id
PROGRAM	program_id
DEPARTMENT	department_id
PROFILE	profile_id
COMPANY	company_id
RECRUITER	recruiter_id
JAF	jaf_id
STUDENT	student_rno
RESUME	resume_id
APPLIES_FOR	(student_rno, jaf_id, resume_id)
SHORTLIST	(student_rno, jaf_id)
OFFER	(student_rno, jaf_id)
ELIGIBLE	(jaf_id, department_id, program_id)

3.2 Foreign Constraints

Table	Attribute	References
COORDINATOR	company_coordinator	coordinator_id of COORDINATOR
RECRUITER	recruiter_company	company_id of COMPANY
JAF	profile_id	profile_id of PROFILE
JAF	company_id	company_id of COMPANY
STUDENT	allocated_jaf	jaf_id of JAF
STUDENT	program_id	program_id of PROGRAM
STUDENT	department_id	department_id of DEPARTMENT
RESUME	student_rno	student_rno of STUDENT
APPLIES_FOR	student_rno	student_rno of STUDENT
APPLIES_FOR	jaf_id	jaf_id of JAF
APPLIES_FOR	resume_id	resume_id of RESUME
SHORTLIST	student_rno	student_rno of STUDENT
SHORTLIST	jaf_id	jaf_id of JAF
OFFER	student_rno	student_rno of STUDENT
OFFER	jaf_id	jaf_id of JAF
ELIGIBLE	jaf_id	jaf_id of JAF
ELIGIBLE	department_id	department_id of DEPARTMENT
ELIGIBLE	program_id	program_id of PROGRAM

3.3 Other Constraints

1. If `jaf_opened_on` is not NULL, no attributes except `jaf_closed_on` and `jaf_slot` can be NULL and a coordinator must be assigned to the company with *id* `company_id`.
2. If `jaf_closed_on` is not NULL, no attributes except `jaf_slot` can be NULL and `jaf_closed_on` \geq `jaf_opened_on`.
3. A student can apply to a JAF if `jaf_opened_on` is not NULL && timestamp of applying is greater than `jaf_opened_on` && timestamp of applying is less than `jaf_closed_on` if `jaf_closed_on` is not NULL && timestamp of applying is less than `allocated_timestamp` if `allocated_jaf` is not NULL && `student_cpi` \geq `jaf_cpi` && student is eligible for the JAF (the JAF is open for his/her department program tuple).
4. *Shortlist*: A student can be shortlisted if the student has applied for the JAF && `data_time` of SHORTLIST \geq `jaf_closed_on` && `data_time` of SHORTLIST \leq `allocation_timestamp` if `allocated_jaf` is not NULL.
5. *Offer*: A student can be offered a job if the student has applied for the JAF && `data_time` of SHORTLIST \geq `jaf_closed_on` && `data_time` of SHORTLIST \leq `allocation_timestamp` if `allocated_jaf` is not NULL.
6. *Allocation*: For a student, a job is allocated (that is, `allocated_jaf` made NOT NULL) if he has been offered a job && `allocation_timestamp` \geq `date_time` of offer.

3.4 NULL Attributes

1. The `company_coordinator` of table COMPANY can be NULL if a coordinator is not yet assigned.
2. The `jaf_opened_on` of table JAF can be NULL if that JAF has not been opened for the students to apply for.
3. The `jaf_closed_on` of table JAF can be NULL if that closing timestamp of the JAF is not yet specified.
4. The `jaf_slot` of table JAF can be NULL if that JAF is not allocated an interview slot.
5. The attributes `student_gender`, `student_dob`, `student_contact`, `year_of_enrollment`, `student_current_year`, `student_cpi`, `student_incentive_points`, `program_id` and `department_id` of table STUDENT can be NULL if they have not been filled by the student or the superuser. During registration, the student is not mandated to fill the details.
6. The attributed `allocated_jaf` and `allocated_timestamp` can be NULL if the student is not allocated a job yet.

4 SQL files: DDL and InsertData

We have generated the .csv files and loaded the same using python script. SQL files and python script are in this link : [drive link](#)

5 Possible Transactions and SQL Statements

5.1 Common to All Users

- **Logging on to the Portal**

For Students:

Input - r1:student_rno, p1:student_password

Query -

```
1      SELECT EXISTS(SELECT 1 FROM STUDENT
2      WHERE student_rno=r1 AND student_password=p1);
```

Similarly, recruiters, coordinators and superusers can also login using their credentials.

- **Viewing Personal Details**

For Students:

Input - r1:student_rno

Query -

```
1      SELECT
2      student_rno ,
3      student_name ,
4      student_gender ,
5      student_dob ,
6      student_email ,
7      student_contact ,
8      year_of_enrollment ,
9      student_current_year ,
10     student_cpi ,
11     student_incentive_points ,
12     program_name ,
13     department_name
14     FROM student INNER JOIN department INNER JOIN program ON
15     student.program_id = program.program_id ,
16     student.department_id = department.department_id
17     WHERE student_rno = r1;
```

Similarly coordinators and recruiters can also view their personal details.

5.2 Students

- **Registering on the Portal**

Input - r1:student_rno, p1:student_password, e1:student_email

Query -

```
1      INSERT INTO STUDENT
2      (student_rno, student_password, student_email)
3      VALUES(r1, p1, e1);
```

- **Editing Personal Details**

Input - r1:student_rno, n1:student_name, g1:student_gender, d1:student_dob, c1:student_contact

Query -

```
1      UPDATE student
2      SET student_name = n1,
3      SET student_gender = g1,
4      SET student_dob = d1,
5      SET student_contact = c1
6      WHERE student_rno = r1;
```

- **Viewing Resume**

Input - r1:student_rno, rid:resume_id

Query -

```
1      SELECT resume_url
2      FROM resume INNER JOIN student ON
3      resume.student_rno = student.student_rno
4      WHERE
5      resume.student_rno = r1 AND resume.resume_id = rid;
```

- **Uploading Resume**

Input - r1:student_rno, u1:resume_url, np:resume_npages

Query -

```
1      INSERT INTO
2      resume(student_rno, resume_url, resume_npages)
3      VALUES (r1, u1, np);
```

- **View list of open JAFs with eligibility**

Input - r1:student_rno:

Query -

```
1      SELECT
2      company.company_name, jaf.jaf_id, jaf.jaf_jd FROM
3      STUDENT
4      INNER JOIN ELIGIBLE
5      INNER JOIN COMPANY
6      INNER JOIN JAF
```

```

7      ON
8      STUDENT.program_id = ELIGIBLE.program_id,
9      STUDENT.department_id = ELIGIBLE.department_id,
10     JAF.jaf_id = ELIGIBLE.jaf_id,
11     STUDENT.student_cpi >= JAF.cpi_cutoff
12     WHERE
13     STUDENT.roll_number = r1
14     AND JAF.jaf_opened_on < CURRENT_TIMESTAMP()
15     AND (JAF.jaf_closed_on IS NULL
16     OR JAF.jaf_closed_on > CURRENT_TIMESTAMP());

```

- **Apply to open and eligible JAF**

Input - r1:student_rno, j1:jaf_id, rid:resume_id

Query -

```

1      INSERT INTO
2      APPLIES_FOR(student_rno, jaf_id, resume_id, date_time)
3      VALUES (r1, j1, rid, CURRENT_TIMESTAMP());

```

- **View list of applied JAFs with status**

Input - r1:student_rno

Query -

```

1      SELECT
2      JAF.jaf_id,
3      COMPANY.company_id
4      FROM
5      APPLIES_FOR INNER JOIN JAF INNER JOIN COMPANY
6      ON APPLIES_FOR.jaf_id = JAF.jaf_id,
7      JAF.company_id = COMPANY.company_id
8      WHERE
9      APPLIES_FOR.student_rno = r1

```

5.3 Recruiter

- **Registering on the Portal**

Input - e1:recruiter_email, p1:recruiter_password

Query -

```

1      INSERT INTO RECRUITER(recruiter_email, recruiter_password)
2      VALUES (e1, p1);

```

- **View Company Details**

Input - r1:recruiter_id

Query -

```

1      SELECT
2      COMPANY.company_name ,
3      COMPANY.company_origin ,
4      COORDINATOR.coordinator_name ,
5      COORDINATOR.coordinator_contact
6      FROM RECRUITER
7      INNER JOIN COMPANY
8      INNER JOIN COORDINATOR
9      ON
10     RECRUITER.recruiter_company = COMPANY.company_id ,
11     COMPANY.company_coordinator = COORDINATOR.coordinator_id;

```

- **Edit Company Details**

Input - r1:recruiter_id, c1:company_name, o1:company_origin

Query -

```

1      UPDATE COMPANY
2      SET company_name = c1,
3      SET company_origin = o1
4      WHERE company_id =
5      (SELECT company_id from RECRUITER
6      WHERE RECRUITER.recruiter_id = r1);

```

- **Create new JAF**

Input - cid:company_id, jd:jaf_jd, p1:jaf_location_of_posting, bd:jaf_bond_duration, cr1:jaf_currency, ctc:jaf_ctc, gross:jaf_gross, profile:profile_id, pid:program_id, did:department_id, ba:jaf_bonus_allowed, cpi:jaf_cpi

Query -

```

1      INSERT INTO
2      JAF(profile_id, company_id, jaf_jd,
3      jaf_bond_duration, jaf_location_of_posting,
4      jaf_currency, jaf_ctc, jaf_gross,
5      jaf_cpi, jaf_bonus_allowed)
6      VALUES(profile, cid, jd, bd, p1, cr1, ctc, gross, cpi, ba);
7      /* we get the jaf_id of the JAF from the database */
8      INSERT INTO
9      ELIGIBLE(jaf_id, program_id, department_id)
10     VALUES(jid, pid, did);

```

- **Edit a JAF**

Editing a JAF is similar to creating a new JAF except that instead of inserting entry in the JAF table, we have to update the entry.

- **View list of created JAFs**

Input - r1:recruiter_id

Query -

```

1      SELECT
2      JAF.jaf_profile , JAF.jaf_jd
3      FROM
4      JAF INNER JOIN COMPANY INNER JOIN RECRUITER
5      ON JAF.company_id = COMPANY.company_id ,
6      RECRUITER.company_id = COMPANY.company_id
7      WHERE
8      RECRUITER.recruiter_id = r1;

```

- **View list of applicants to a JAF**

Input - rid:recruiter_id, jid:jaf_id

Query -

```

1      SELECT
2      STUDENT.student_name ,
3      STUDENT.student_rno ,
4      STUDENT.student_cpi
5      FROM RECRUITER
6      INNER JOIN JAF
7      INNER JOIN APPLIES_FOR
8      INNER JOIN STUDENT
9      ON JAF.jaf_id = APPLIES_FOR.jaf_id ,
10     APPLIES.student_rno = STUDENT.student_rno
11     WHERE JAF.jaf_id = jid
12     AND RECRUITER.recruiter_id = rid;

```

- **View Student Resume**

Input - jid:jaf_id, rid:resume_id, r1:student_rno

Query -

```

1      SELECT resume_url FROM RESUME
2      WHERE resume_id = rid AND student_rno = r1;

```

5.4 Coordinator

- **Registering on the Portal**

Input - n1:coordinator_name, e1:coordinator_email, c1:coordinator_contact,
p1:coordinator_password

Query -

```

1      INSERT INTO COORDINATOR
2      (coordinator_name , coordinator_email ,
3      coordinator_contact , coordinator_password)
4      VALUES(n1, e1, c1, p1);

```

- **View list of Allotted Companies**

Input - cid:coordinator_id

Query -

```
1      SELECT COMPANY.company_name
2      FROM COMPANY WHERE
3      COMPANY.company_coordinator = cid;
```

- **View list of JAFs of a Company**

This shall work exactly like for recruiter.

- **View list of applicants for a JAF**

This shall also work exactly like for recruiter.

- **Open JAF**

Input - jid:jaf_id

Query -

```
1      UPDATE JAF
2      SET opened_on = CURRENT_TIMESTAMP()
3      WHERE JAF.jaf_id = jid;
```

- **Close JAF**

Input - jid:jaf_id

Query -

```
1      UPDATE JAF
2      SET closed_on = CURRENT_TIMESTAMP()
3      WHERE JAF.jaf_id = jid;
```

5.5 Superuser

- **Add Coordinator**

Input - n1:coordinator_name, e1:coordinator_email, c1:coordinator_contact, p1:coordinator_password

Query -

```
1      INSERT INTO COORDINATOR
2      (coordinator_name, coordinator_email,
3      coordinator_contact, coordinator_password)
4      VALUES(n1, e1, c1, p1);
```

- **Create a Company entity and allocate to a Coordinator**

Input - n1:company_name, c1:company_coordinator, o1:company_origin

Query -

```

1      INSERT INTO COMPANY
2      (company_name, company_coordinator,
3      company_origin)
4      VALUES(n1, c1, o1);

```

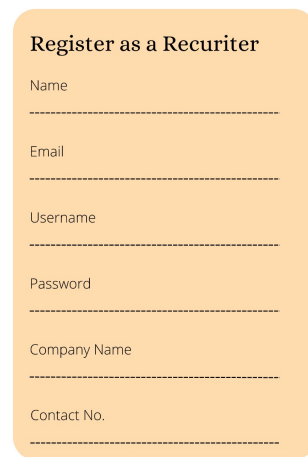
6 Indexes for Query Optimization

Following are the indices we have identified to improve the performance of SQL queries:

Table	Attribute as index	Associated query
JAF	profile_id	view all JAFs of a profile
JAF	company_id	view all JAFs created by a company
JAF	jaf_currency, jaf_ctc	view all JAFs in a range of CTC for categorising them
JAF	jaf_currency, jaf_gross	view all JAFs in a range of gross compensation
JAF	jaf_cpi	view all JAFs with a certain CPI cutoff
JAF	jaf_opened_on	view all JAFs open currently
JAF	jaf_closed_on	view all JAFs closing today
JAF	jaf_slot	view all JAFs of a slot
STUDENT	year_of_enrollment	view all students of a particular year
STUDENT	student_cpi	view the students above a certain CPI threshold
STUDENT	program_id	view the students of a particular program
STUDENT	department_id	view the students of a department
STUDENT	allocated_jaf	view all students allocated a specific job role
APPLIES_FOR	jaf_id	view list of applicants of a JAF
SHORTLIST	student_rno	view the JAFs a student is shortlisted in
SHORTLIST	jaf_id	view the students shortlisted by a JAF
OFFER	student_rno	view the offers made to a student
OFFER	jaf_id	view the offers made by a JAF
ELIGIBLE	jaf_id	view the departments and programs eligible for a JAF

7 User Forms

Here are the user forms for the use cases. We will be using Angular for the UI form interface. **Note:** Do not consider the given wireframes as final. They are only for demonstrative/visual purposes.



Register as a Recruiter

Name

Email

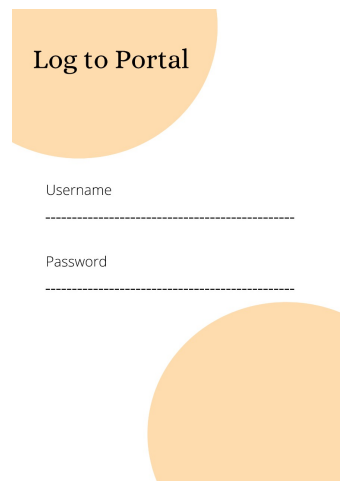
Username

Password

Company Name

Contact No.

Figure 2: Registering on the Portal (shown for recruiter)



Log to Portal

Username

Password

Figure 3: Logging on to the Portal (shown for recruiter)



Edit Recuritor Details

Name

Email

Contact No.

Figure 4: Edit Details (shown for recruiter)

Create a New JAF

Jaf Description

Place of Posting

Bond Duration

Eligibility

Profile

Currency

CTC

Gross

Figure 5: Create a New JAF

Edit a JAF

Jaf Description

Place of Posting

Bond Duration

Eligibility

Profile

Currency

CTC

Gross

Figure 6: Edit a JAF

Change Student Status

JAF ID

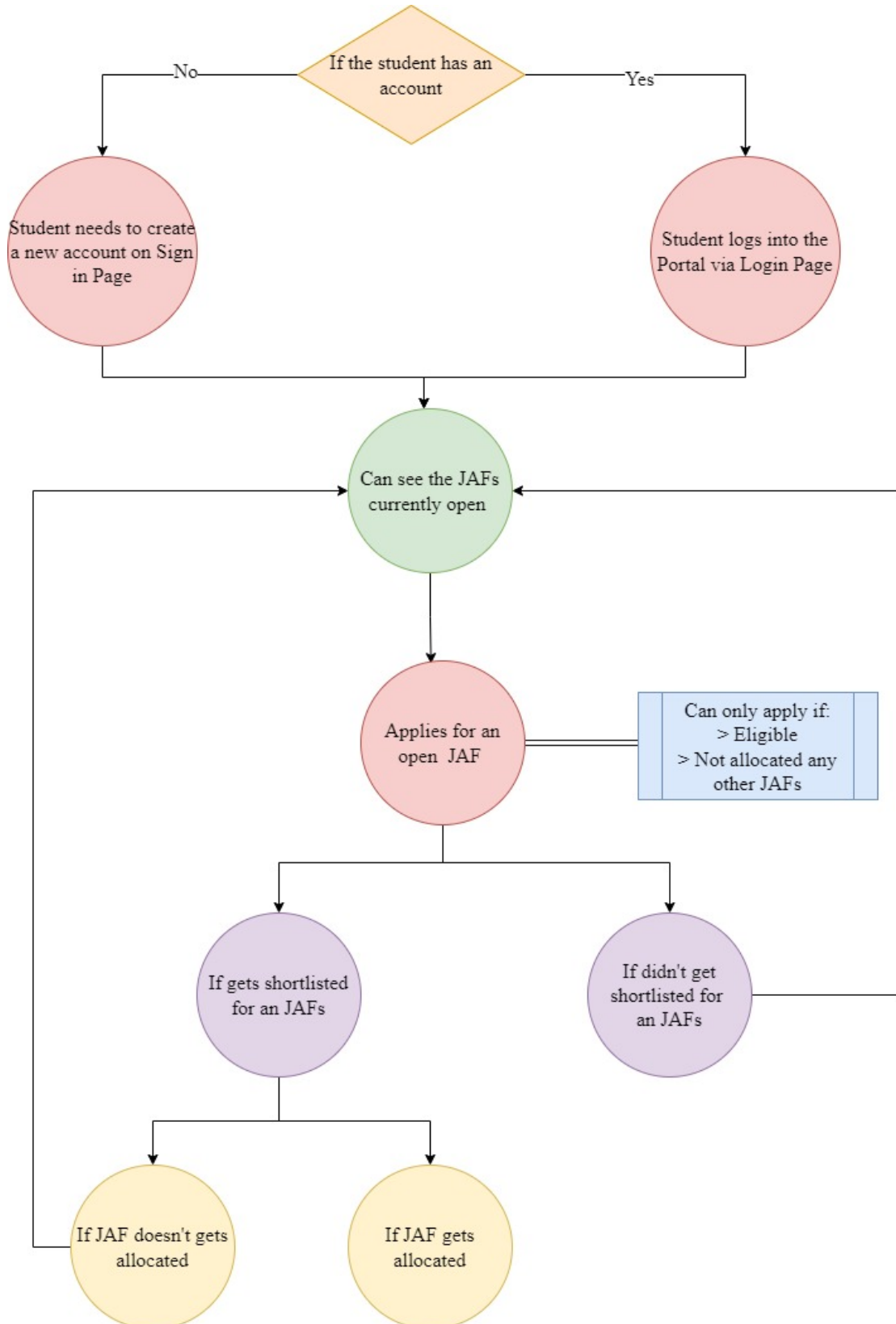
Roll Number

Status

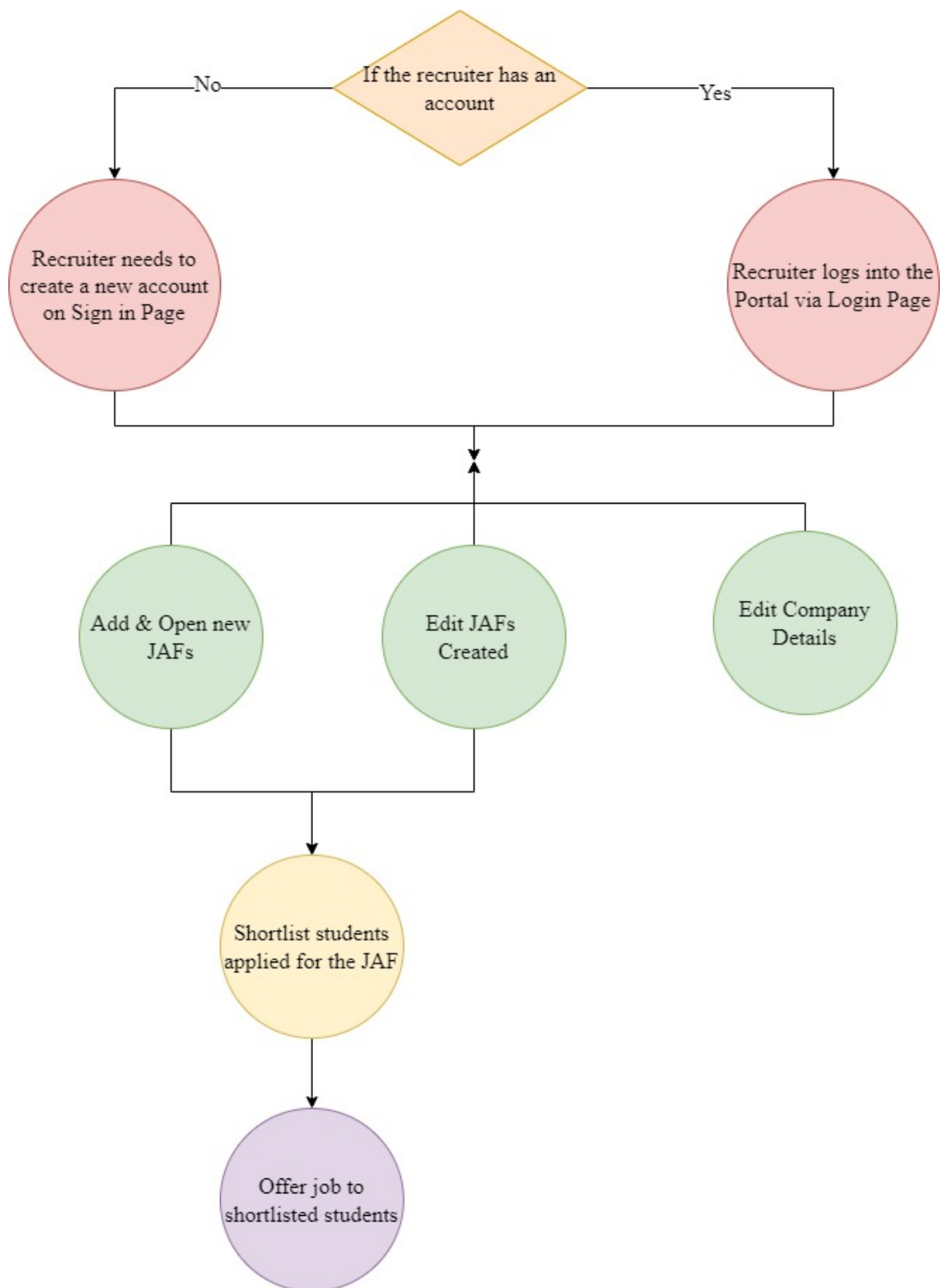
Figure 7: Change Student Status

8 Business Logic Controller

8.1 User #1: Student



8.2 User #2: Recruiter



8.3 User #3: Coordinator

