



Campus Return - info and supervision

DBMS FINAL PROJECT

-INDUJA KALA (195013)
-DHRUV PANWAR (195048)
-SHIVANKIT DHIMAN (195003)
-MANAV DODA (195057)

GitHub Repository: [Click Here!](https://github.com/dhruvininfo28/campus-return) (https://github.com/dhruvininfo28/campus-return)

Deployed Website: [Click Here!](https://campus-return.herokuapp.com/) (https://campus-return.herokuapp.com/)

INDEX

1.OUR MOTIVATION

2.PROJECT BRIEF

3. ER DIAGRAM-ENTITIES

ATTRIBUTES

RELATIONSHIP TYPES

DIAGRAM

TOTAL PARTICIPATION AND CARDINALITY RATIO

4.RELATIONAL TABLES

5. FUNCTIONAL DEPENDENCIES

6 NORMALIZATION (tables normalized up to BCNF)

7. CREATING THE TABLES (SQL queries)

8. TRIGGERS USED

9. WORKING

10. FEATURES

11. TOOLS AND TECHNOLOGIES

12. CONTRIBUTIONS

Why did we choose this:

- Seeing the current situation of covid we think this is something every institution needs
- As students return from their home to the campus, we believe that there is a need of a database management system that tracks their application status as well as makes sure that the database of the students has been taken proper care of
- For e.g., the RTPCR reports, Vaccination certificates, application of the student and basic data of a college student

WHAT IS OUR PROJECT ABOUT:

-We have designed a portal for students to log into and submit all the basic information as well as covid related documents like the RTPCR reports and vaccination certificates.

-It also enables the administration to approve or deny (based on the documents submitted by students) a student's application for returning to the campus.

ER DIAGRAM:

ENTITIES AND ATTRIBUTES:

1. **Student** -Who will be applying through our portal to return to campus.

ATTRIBUTES: Roll_no,Name,Branch,Email_id>Password,Year,Programe

2. **First_dose**-It contains the covid vaccination first dose certificate.

ATTRIBUTES: FD_Document, Benificiary_id, Fdose_id, Roll_no

3. **Second_dose**-It contains the covid vaccination second dose certificate.

ATTRIBUTES: FD_Document,Benificiary_id,Sdose_id,Roll_no

4. RTPCR-It contains the RTPCR report of the test taken by the student before returning to campus.

ATTRIBUTES: Roll_no,FD_Document,RTPCR_id,Expiration_date

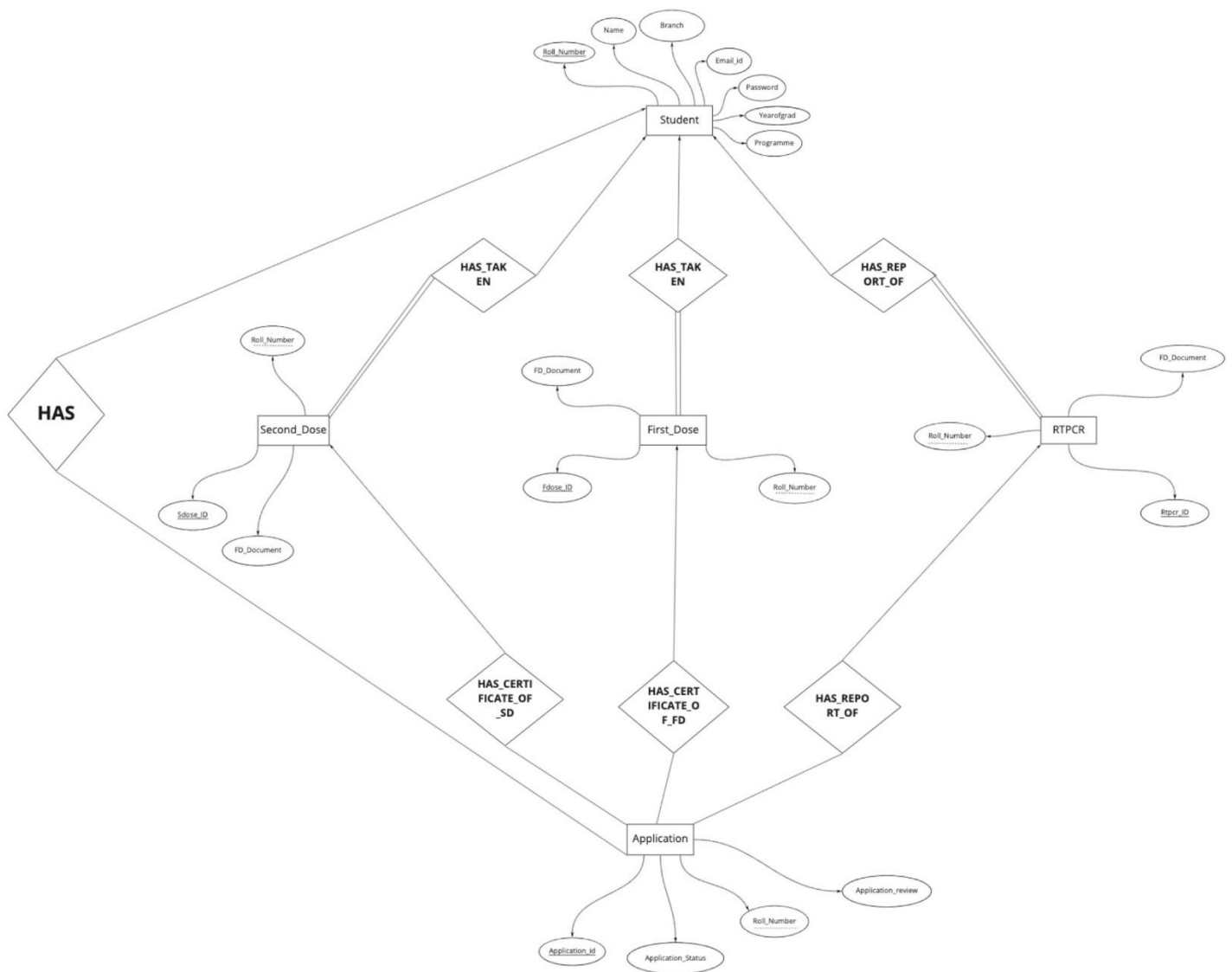
5. Application-This entity will be created once the student uploads all his basic information and the required documents.

ATTRIBUTES: Application_id,Application_Status,Roll_no

RELATIONSHIP_TYPES:

Entity_type	Relationship_type	Entity_type
Student	Has	Application
Student	Has_taken	First_dose
Student	Has_taken	Second_dose
Student	Has_report_of	RTPCR
Application	Has_certificate_of_sd	Second_dose
Application	Has_certificate_of_fd	First_dose
Application	Has_report_of	RTPCR

DIAGRAM:



Cardinality Ratio and Total Participation:

-So, talking about the relationships in the table as we can see in this table that we have drawn that each entity is related to one another and their relationships have been described. Now talking about the total participation and cardinality ratios:

- If we closely look at the ER diagram, we see total participation from second dose, first dose and RTPCR towards student. We can conclude from here that if there is a second dose entity or a certificate it would exist only when student is there while a student can or cannot have a second dose report. Same is the case with First dose and RTPCR.

--Now talking about student and application, a student can have multiple applications hence the relationship here has been defined as one to many. There is also many to one relationship from application to first dose second dose and RTPCR which means an application is associated with at most one first dose one RTPCR and one second dose or we can say same first dose can exist for multiple applications described clearly in the diagram.

RELATIONAL TABLES:

--Now talking about converting ER DIAGRAMS to relational tables. we know A strong entity set with only simple attributes or any no of composite attributes will require only one table in relational model. Hence the tables Student, Application, First dose, Second DOSE and RTPCR .

--Also, for many to one relationship from application to first dose, second dose and RTPCR their relationship has been made as separate tables shown here.

First_Dose

FD_Document	<u>Fdose_ID</u>	Roll_number
-------------	-----------------	----------------------

Student

<u>Roll_Number</u>	Name	Branch	Email_id	Password	Yearofgrad	Programme
--------------------	------	--------	----------	----------	------------	-----------

Second_dose

FD_Document	<u>Sdose_ID</u>	Roll_number
-------------	-----------------	----------------------

RTPCR

Roll_Number	<u>Rtpcr_id</u>	Fd_document
----------------------	-----------------	-------------

Application

<u>Application_id</u>	Application_status	Roll_number	Application_Review
-----------------------	--------------------	----------------------	--------------------

Has_certificate_of_fd

Fdose_ID	<u>Application_Id</u>
-------------------	-----------------------

HAS_CERTIFICATE_OF_SD

Sdose_ID	<u>Application_Id</u>
-------------------	-----------------------

HAS_REPORT_OF

Rtpcr_ID	<u>Application_Id</u>
-------------------	-----------------------

FUNCTIONAL DEPENDENCIES:

1. Student table:

Roll_Number \rightarrow Name

Roll_Number \rightarrow branch

Roll_Number \rightarrow email_id

Roll_Number \rightarrow Password

Roll_Number \rightarrow yearofgrad

Roll_Number \rightarrow Programme

email_id \rightarrow Roll_number

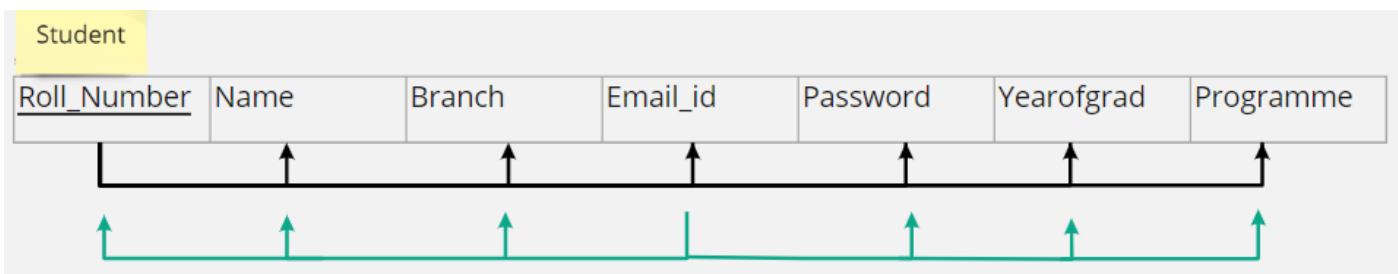
email_id \rightarrow Name

email_id \rightarrow Branch

email_id \rightarrow Password

email_id \rightarrow Yearodgrad

email_id \rightarrow Programme

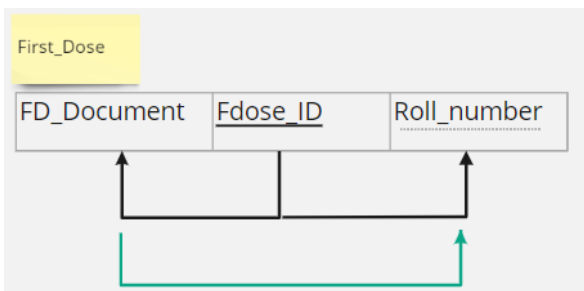


2. First_dose:

Fdose_id \rightarrow FD_Document

Fdose_id \rightarrow Roll_number

FD_Document \rightarrow Roll_number

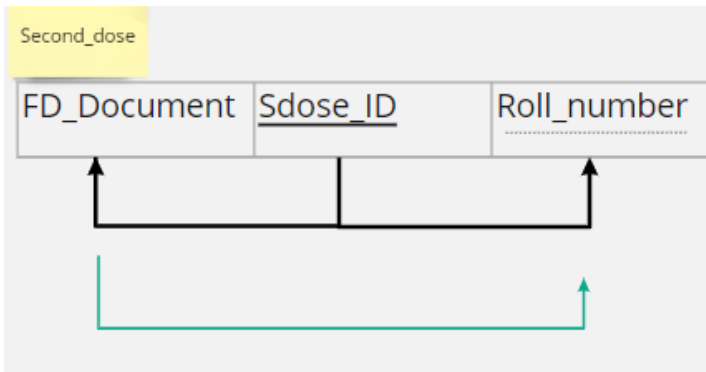


3. Second_dose:

Sdose_id \rightarrow FD_Document

Sdose_id \rightarrow Roll_number

FD_Document \rightarrow Roll_number

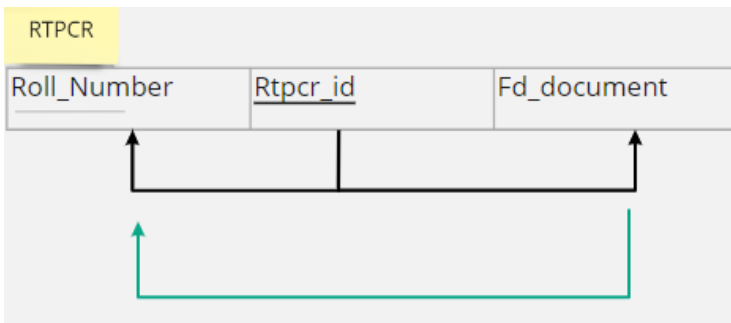


4. RTPCR:

Rtpcr_id → Roll_number

Rtpcr_id → Fd_document

Fd_document → Roll_number

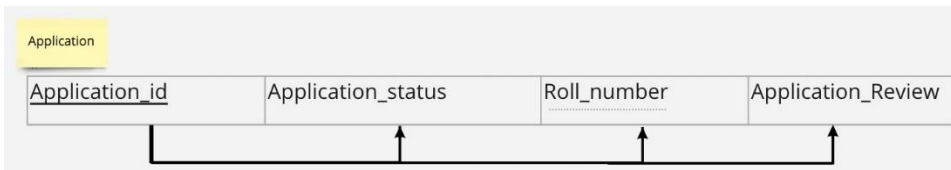


5. Application:

Application_id → Application_status

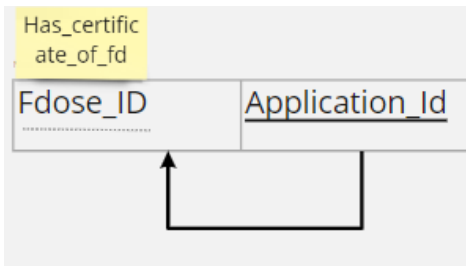
Application-id → Roll_number

Application-id → Application_Review



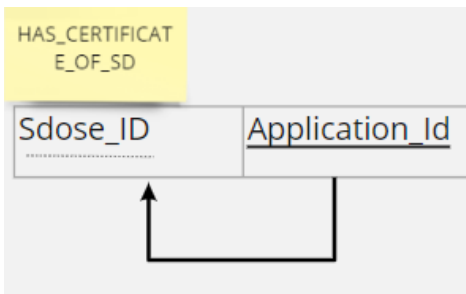
6. Has_certificate_of_fd:

Application_id → FDose_id



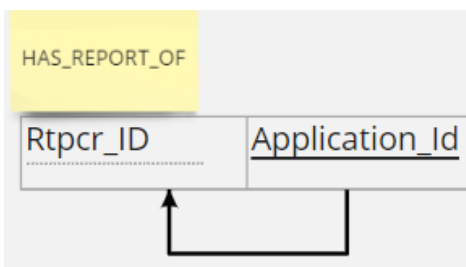
7. Has_certificate_of_sd:

Application_id → SDose_id



8. Has_report_of:

Application_id → Rtpcr_id

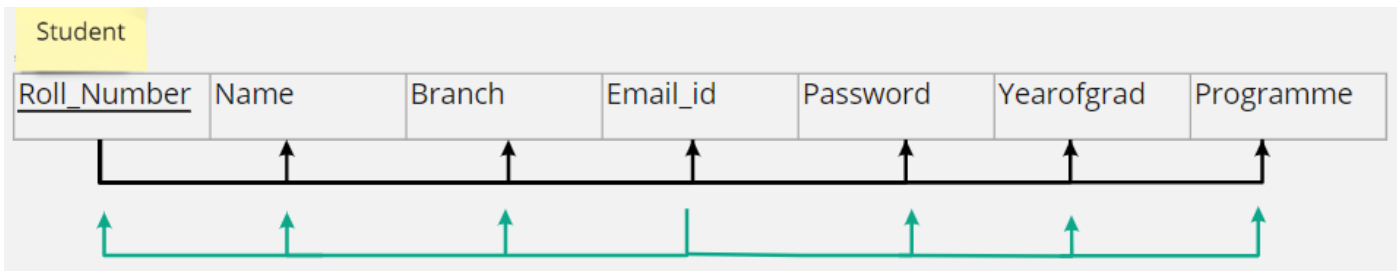


NORMALIZATION:

Functional dependencies have been defined above:

1. Student table:

Candidate Keys: Roll_Number
Email_id



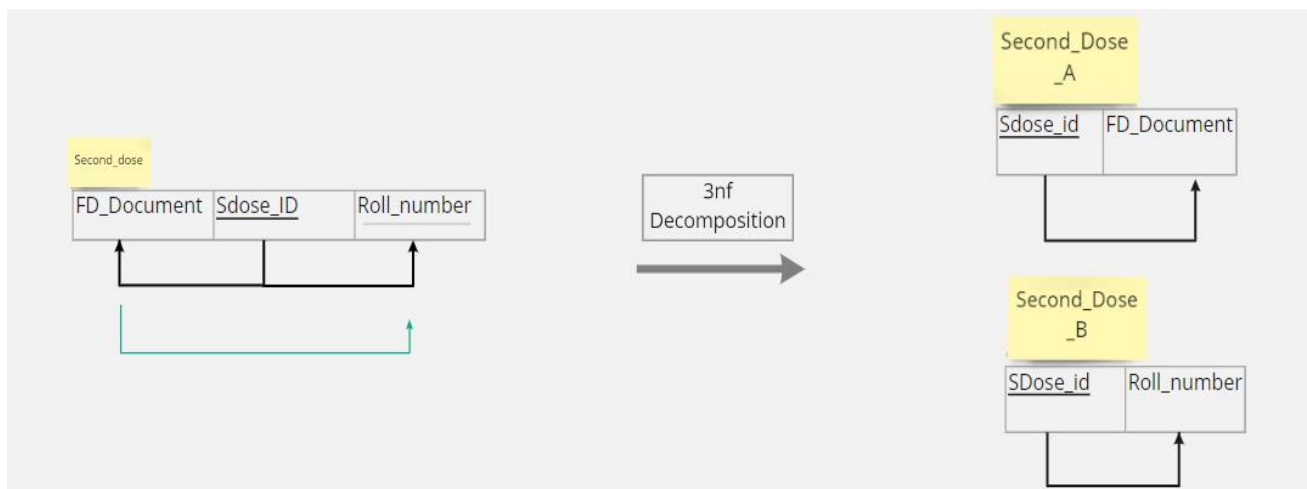
→ So, beginning from the student table, we have two candidate keys in our student table, one is roll number of student and the other one is email id of student because we can determine all the other attributes of student table by using any one of them explicitly. Since there are no multi-valued attributes in our student table the table satisfies the properties of 1NF. Here we have taken roll number as our primary key and 2NF states that every non-prime attribute should be fully functionally dependent on the primary key and as we can see here every attribute can be determined by roll number. So, the table also complies with the properties of 2NF. Now talking about the 3NF, 3NF states that no non-prime attribute should be transitively dependent on the primary key. And as here we can see no non-prime attribute is determining any of the other non-prime attributes. Therefore, the table is also in 3NF. Lastly, talking about the BCNF, BCNF is a special form of 3NF which states that if any non-trivial dependency $X \rightarrow A$ holds then X should be a super key. And here we can see that all the attributes are determined by our candidate keys roll number and email id and as we know each candidate key is also a super key. Therefore, the table is also in BCNF form.

2. First_dose:

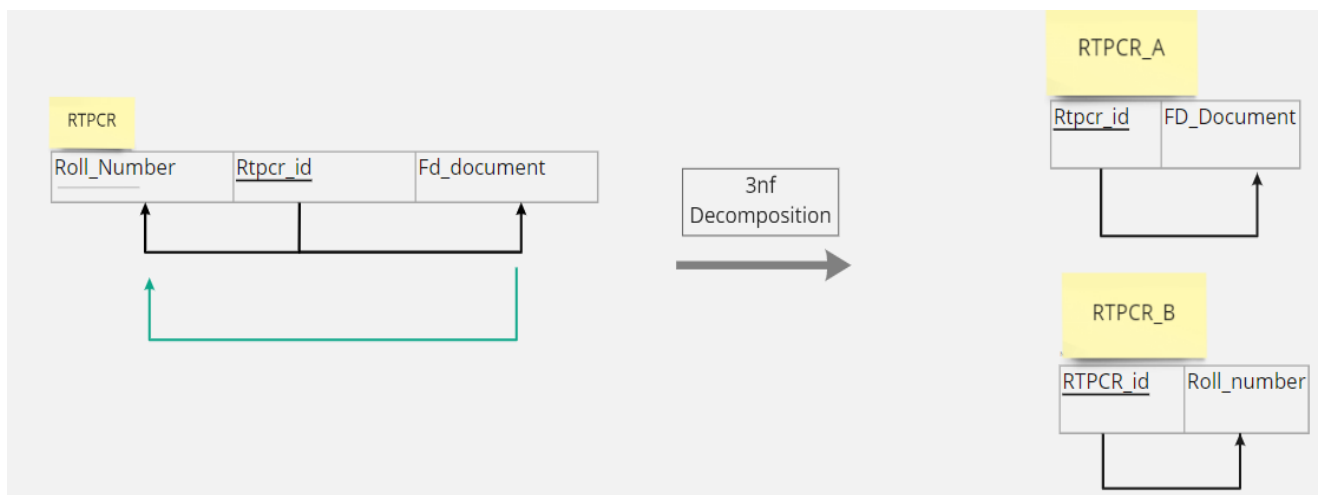
Candidate Key: Fdose_ID



3. Second_dose:
Candidate Key: Sdose_ID



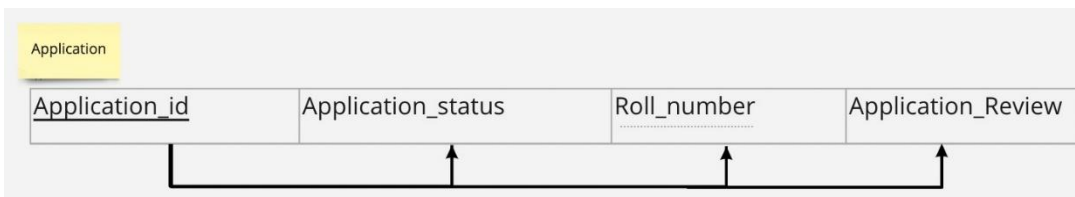
4. RTPCR:
Candidate Key: RTPCR_id



→ In these tables we have Fdose_id, Sdose_id and RTPCR_id as our candidate keys respectively and since we are assigning unique ids every time someone uploads a new document therefore each id is mapped to exactly one doc and one roll number therefore the table satisfies the 1NF. Since the primary key is made up of only a single attribute therefore there can be no partial dependency on primary key and hence it also satisfies 2NF. Further moving on to 3NF we see that FD_document which is a non-prime attribute can determine roll number which is also a non-prime attribute. Therefore, this is a case of transitive dependency and hence the table is not in 3NF form. Therefore, to convert it into 3NF we'll need to decompose the table into two smaller tables which are as shown here. Similarly, we have broken Second Dose & RTPCR tables to convert them into 3NF. Now in the new tables there are only two attributes and the primary key is determining the other attribute therefore they are also in BCNF.

5. Application table:

Candidate key: Application_id



Here we have application_id as our candidate key and in this table, we have no multivalued attribute. Therefore, it is in 1NF. Now talking about 2NF since there is only one attribute in the primary key therefore all the non-prime attributes are fully functionally dependent on primary key. Therefore, the table satisfies 2NF form too. Now the application_status, roll_number and Application_Review are not related to each other in any form. Therefore, there is no transitive dependency and both roll number and application_status are dependent only on the primary key i.e., application_id therefore the table is also in BCNF.

CREATING THE TABLES (SQL Queries):

1. Student table:

```
create table if not exists student
(student_roll_number varchar(20) primary key,
student_name varchar(50),
student_branch varchar(60),
student_email_id varchar(60),
```

```
student_password varchar(150),
student_year int,
student_programme varchar(70),
student_token varchar(150),
student_verified_status tinyint(1));
```

2. Second dose:

```
create table if not exists Second_dose_A
(Sdose_id varchar(75) primary key,
FD_Document mediumblob);
```

```
create table if not exists Second_dose_B
(Sdose_id varchar(75) primary key,
student_roll_number varchar(20),
constraint sd_certificate_of
foreign key (student_roll_number)
references student (student_roll_number)
on update cascade
on delete cascade);
```

3. First dose:

```
create table if not exists First_Dose_A
(Fdose_id varchar(75) primary key,
FD_Document mediumblob);
```

```
create table if not exists First_Dose_B
(FDose_id varchar(75) primary key,
student_roll_number varchar(20),
constraint fd_certificate_of
foreign key(student_roll_number) references
student(student_roll_number)
on update cascade
```

on delete cascade);

4. RTPCR:

```
create table if not exists rtpcr_a
(Rtpcr_id varchar(75) primary key,
FD_Document mediumblob);
```

```
create table if not exists rtpcr_b
(Rtpcr_id varchar(75) primary key,
student_roll_number varchar(20),
constraint has_report_of
foreign key(student_roll_number)
references student(student_roll_number)
on update cascade
on delete cascade);
```

5. Application:

```
create table if not exists Application" +
(Application_id int primary key auto_increment,
Application_status int not null,
student_roll_number varchar(20) not null,
```



```
application_review varchar(500),  
constraint applied_by  
foreign key(student_roll_number)  
references student(student_roll_number)  
on update cascade  
on delete cascade);
```

6. Has_certificate_of_fd (Application to first dose certificate mapping)

```
create table if not exists has_certificate_of_fd  
(Application_id int primary key  
Fdose_id varchar(75),  
constraint fd_belonging_to  
foreign key(Application_id) references Application(Application_id)  
on update cascade  
on delete cascade,  
constraint __doc1  
foreign key(Fdose_id) references First_Dose_A(Fdose_id)  
on update cascade  
on delete set null);
```

7. Has_certificate_of_sd (Application to second dose certificate mapping)

```
create table if not exists has_certificate_of_sd  
Application_id int primary key,  
Sdose_id varchar(75),
```

```
constraint sd_belonging_to  
foreign key(Application_id) references Application(Application_id)  
on update cascade  
on delete cascade,  
constraint __doc2  
foreign key(Sdose_id) references Second_Dose_A(Sdose_id)  
on update cascade  
on delete set null);
```

8. Has_report_of (Application to Rtpcr report mapping)

```
create table if not exists has_report_of  
Application_id int primary key,  
Rtpcr_id varchar(75),  
constraint report_belonging_to  
foreign key(Application_id) references Application(Application_id)  
on update cascade  
on delete cascade,  
constraint __doc3  
foreign key(Rtpcr_id) references rtpcr_a(Rtpcr_id)  
on update cascade  
on delete set null);
```

CREATING THE TRIGGER (SQL Query):

delimiter //

create trigger if not exists on_application_insert

after insert

on Application For each row

begin

declare roll_number varchar(20) default '';

declare curr_application_id int default 0;

declare rowcount int default 0;

declare req_fdose_id varchar(75) default '';

declare req_sdose_id varchar(75) default '';

declare req_rtpcr_id varchar(75) default '';

-- Selecting student roll number

set roll_number = new.student_roll_number;

set curr_application_id = new.application_id;

select count(*) into rowcount from first_dose_b where student_roll_number = roll_number;

set rowcount = rowcount - 1 ;

if rowcount <> -1 then

set req_fdose_id = (select Fdose_id from first_dose_b where student_roll_number = roll_number limit 1 offset rowcount);

end if;

select count(*) into rowcount from second_dose_b where student_roll_number = roll_number;

set rowcount = rowcount - 1 ;

if rowcount <> -1 then

select Sdose_id into req_sdose_id from second_dose_b where student_roll_number = roll_number limit 1 offset rowcount;

end if;

select count(*) into rowcount from rtpcr_b where student_roll_number = roll_number;

set rowcount = rowcount - 1 ;

```

if rowcount <> -1 then
select rtPCR_id into req_rtPCR_id from rtPCR_b where student_roll_number = roll_number limit 1 offset
    rowcount;
end if;

if req_fdose_id <> " then
insert into has_certificate_of_fd values(curr_application_id,req_fdose_id);
end if;

if req_sdose_id <> " then
insert into has_certificate_of_sd values(curr_application_id,req_sdose_id);
end if;

if req_rtPCR_id <> " then
insert into has_report_of values(curr_application_id,req_rtPCR_id);
end if;
end //
delimiter ;

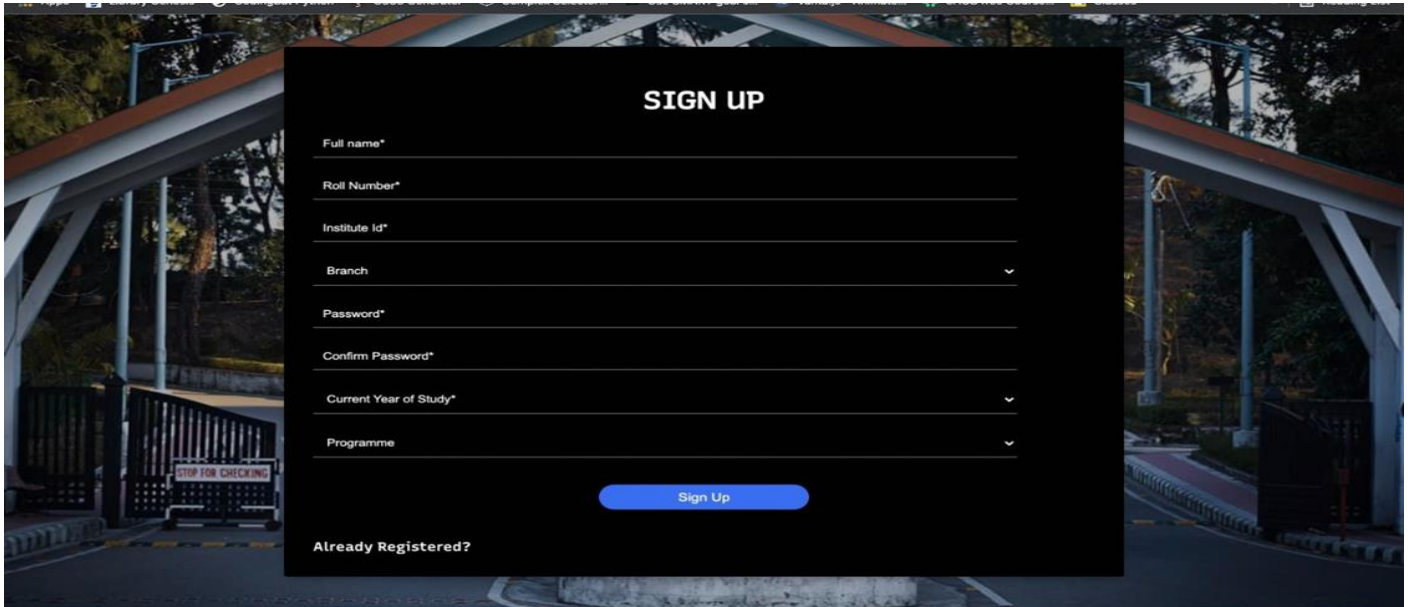
```

The above **trigger** runs **after** an entry is made in the **application** table.

It extracts the **latest** First Dose certificate from First_Dose_B, the latest Second Dose certificate from Second_Dose_B and latest RtPCR from RtPCR_B of the student who submitted the application and makes corresponding entries in has_certificate_of_fd, has_certificate_of_sd and has_report_of tables respectively.

WORKING (CAMPUS_RETURN)

1) **Registration:** The students will have to create their account by filling up their professional details like name, roll number, branch and programme on the Sign Up page. The students are only allowed to sign in using their official college ID and so only NIT-Hamirpur students will be able to register.



SIGN UP

Full name*

Roll Number*

Institute Id*

Branch

Password*

Confirm Password*

Current Year of Study*

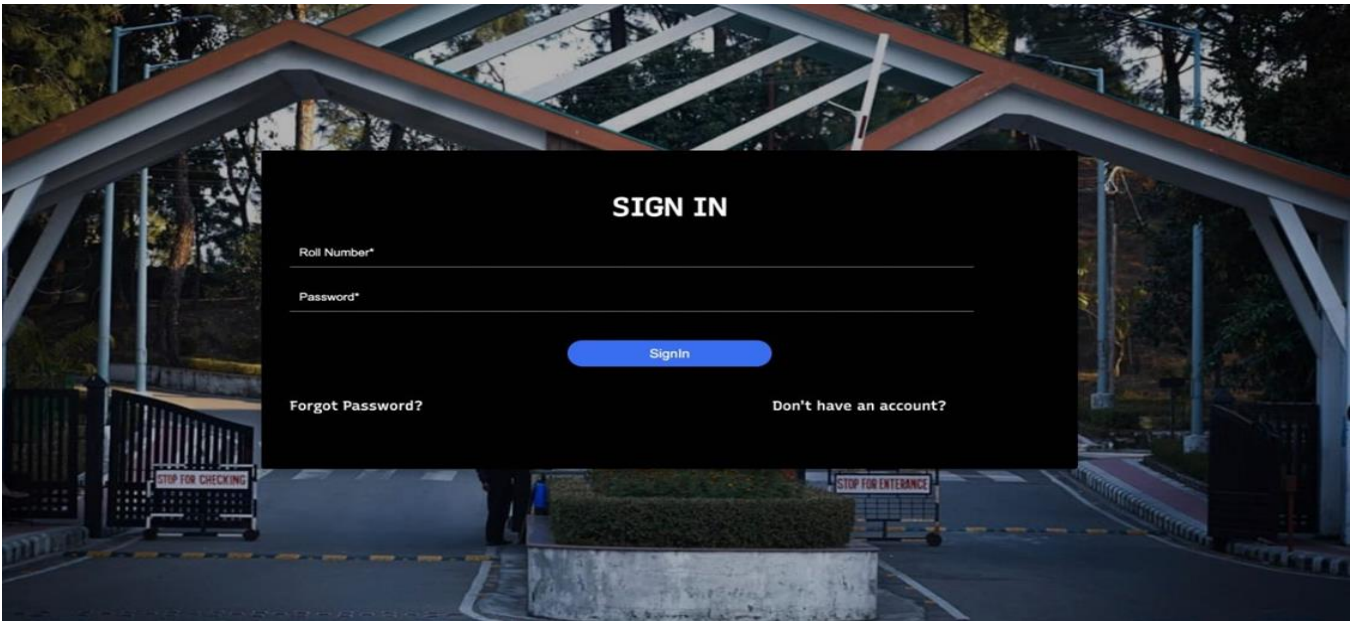
Programme

Sign Up

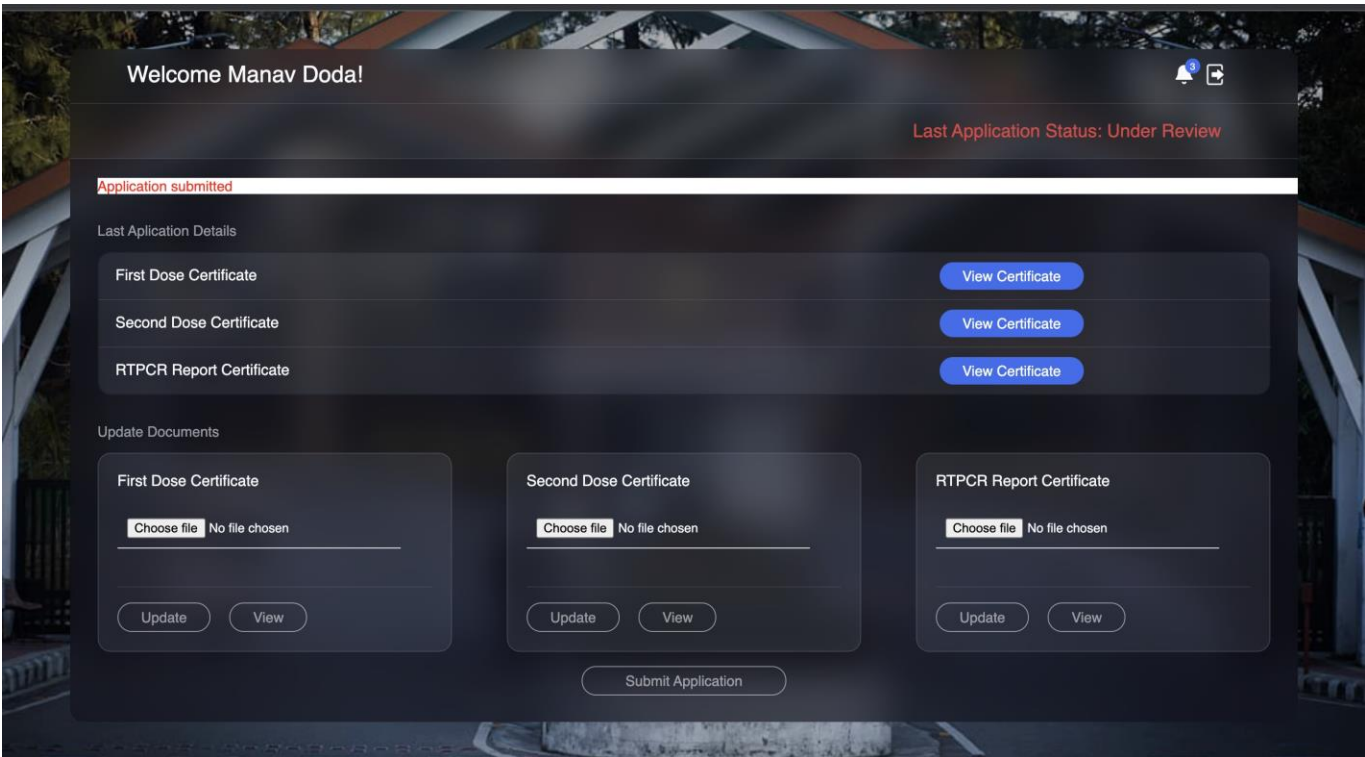
Already Registered?

2) Once the student has registered, he/she/other can **Sign In** using their roll number and password anytime they want.

A **student's session** will be saved until they logout, so even if they switch tabs or close this tab they will be logged in and now can access the dashboard without the hassle of logging in again.

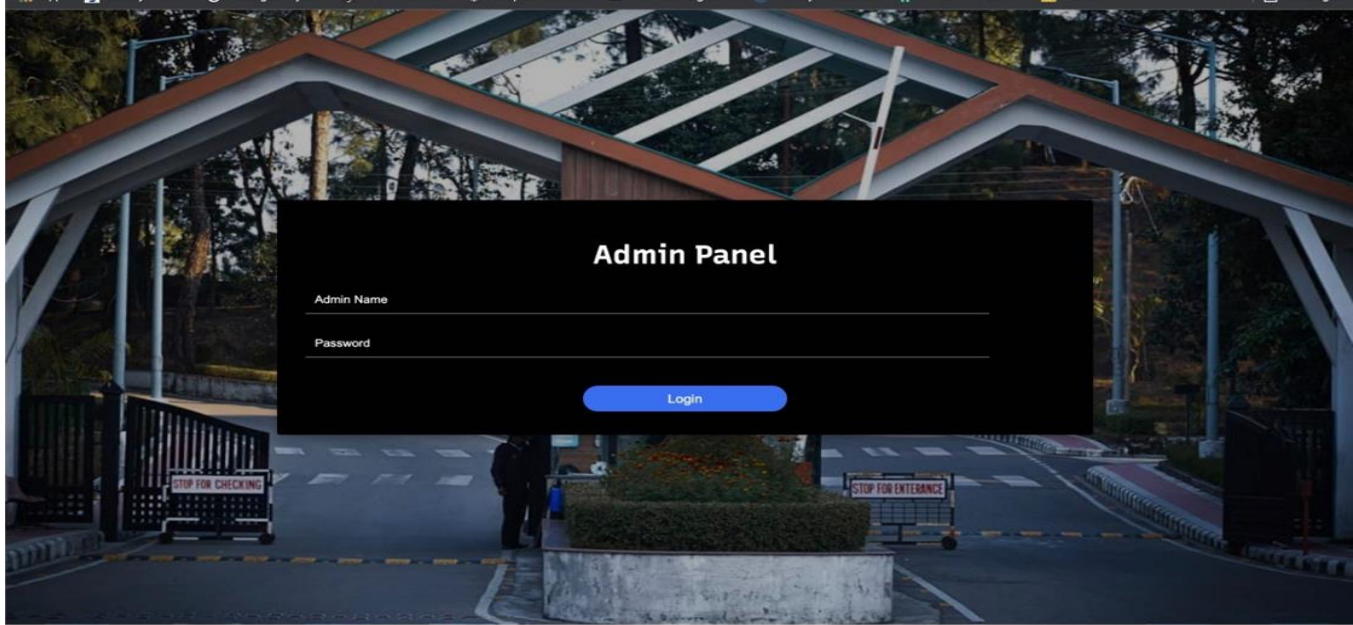


3) The **student dashboard** shows status of the previous application the student submitted. A student can also view the certificates(first and second dose) and rtPCR report which he/she/other submitted for the same(last application). Also, the dashboard provides students' with the option of **updating** their documents in case their previous application was rejected and submit a new application.



4) For the concerned authority to view all the applications submitted and review them an **Admin Panel** has been designed.

The concerned authority can login to the admin panel as admin and view all the applications submitted so far.



5) This is the **dashboard for the Admin** where they can view all the applications in brief and then with the view application button , thoroughly review the specific application.

Welcome to Admin Panel

Search for Student or Roll No.

Filters: Filter by Branch Name Filter by Year of Study* Filter by Programme Apply filter

Roll No.	Student Name	Branch	Year	Programme	Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application
195057	Manav Doda	CSE	3rd	B.Tech	View Application

6) This is the **thorough view of an application** where the admin can view the whole application and applicant details and can approve/deny it with a descriptive review.

Application Number: 1

Applicant Details

Name of Applicant:	Manav Doda
Roll Number of Applicant:	195057
Email ID of Applicant:	195057@nith.ac.in
Applicant Branch:	Computer Science and Engineering
Programme Enrolled in:	B.Tech
Current Year:	3
First Dose Certificate:	View Certificate
Second Dose Certificate:	View Certificate
RTPCR Report:	View Certificate

Application Number: 1

Programme Enrolled in:

B.Tech

Current Year:

3

First Dose Certificate:

View Certificate

Second Dose Certificate:

View Certificate

RTPCR Report:

View Certificate

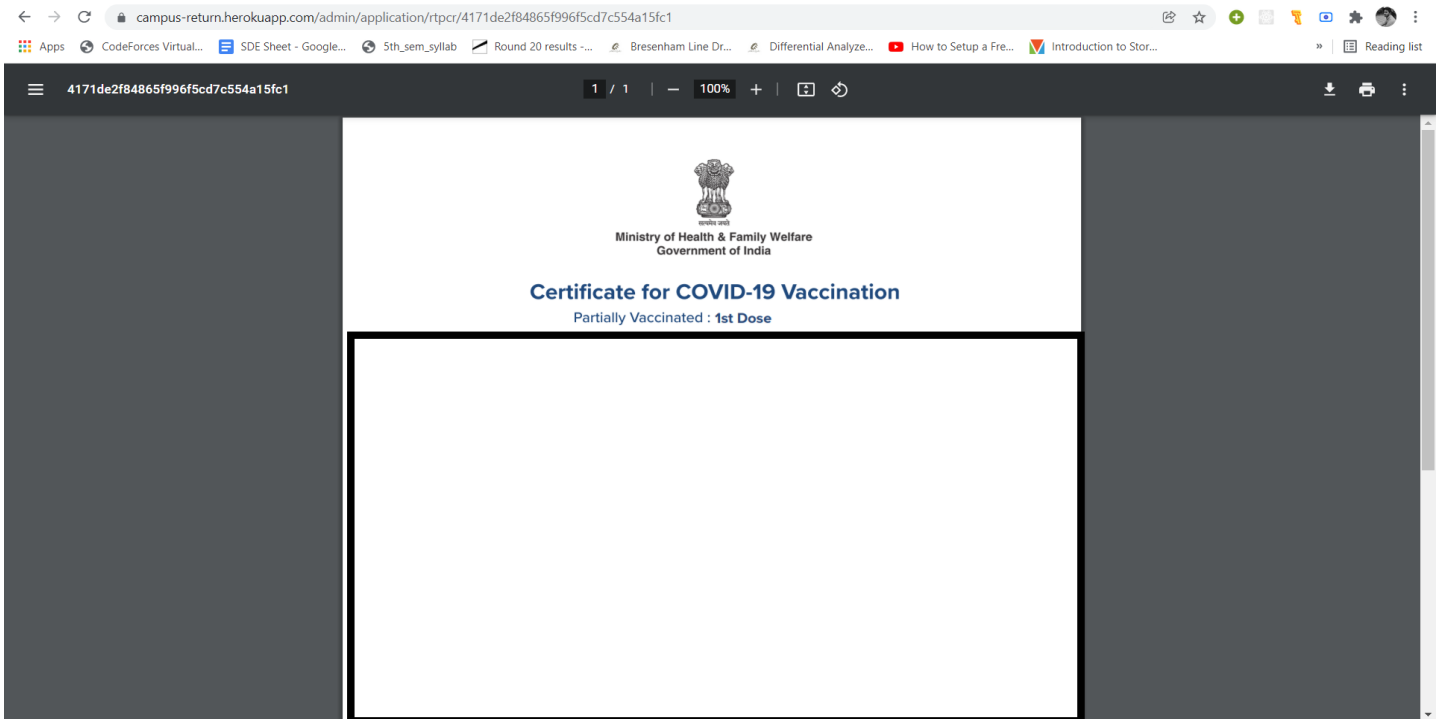
Response Area

Application Status

Enter your review here...

Submit Review

7) On clicking the **view certificate** button in thorough view of an application the admin can view the certificate as the screenshot show below:



FEATURES:

Email verification

Password Encryption

Session management Both for admin as well as students

CSRF Protection

Blob data management

TOOLS AND TECHNOLOGIES USED:

Frontend: HTML5, CSS3, JavaScript

Backend: Node.js

Database: MySQL, Microsoft Azure MySQL flexible server

Version Control: Git

Deployment: Heroku

GitHub Repository: [Click Here!](https://github.com/dhruvinfo28/campus-return) (https://github.com/dhruvinfo28/campus-return)

Deployed Website: [Click Here!](https://campus-return.herokuapp.com/) (https://campus-return.herokuapp.com/)

CONTRIBUTIONS:

Induja Kala	ER Diagrams, Relationship Design & Normalization of Tables
Shivankit Raj Dhiman	UI Design & Normalization of Tables
Manav Doda	UI Design, Normalization of Tables, ER Diagram & Relationship Design
Dhruv Panwar	Normalization, Backend Development, database implementation and deployment