The current mess management system at our University JawaharLal Nehru University is manual,decentralised and inefficient.

**CURRENT SITUATION:-**

* The students have to sign the Mess Register to get the plates of the Food,but this sometimes leads to duplicity as the manager is unable to identify the legit student and some students eats illegally which lead to inefficiency.
* The data of the Mess is Decentralised so a student of the Hostel1 have to eat in his/her respective hostel only,sometimes it leads to problem as he/she have to go to far and sometimes they miss their classes as well.
* This also sometimes leads to lack of transparency in billing system.

**DESIRED FUTURE STATE:-**

* If the mess management system will get Online then the student have to enter their respective Details only then he/she will be albe to access the Food.So this will make our Mess Management System more efficient.
* If the Data of the hostels get centralised then the Student will be able to Eat in any food so he/she doesn’t have to go to to long distance and can have food in nearby hostel.
* Our project will also lead to transparency in Mess Management System,accurate billing and reduction in wait times for Food.

A database management system (DBMS) refers to the technology for creating and

managing databases.Basically DBMS is a software tool to organize (create, retrieve, update

and manage) data in a database.

The main aim of a DBMS is to supply a way to store up and retrieve database

information that is both convenient and efficient. By data, we mean known facts that can

be recorded and that have embedded meaning. Normally people use software such as

DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database.

Database systems are meant to handle large collection of information. Management

of data involves both defining structures for storage of information and providing

mechanisms that can do the manipulation of those stored information. Moreover, the

database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

This project is aim at computerizing the manual process of Mess Management System.The database is implemented using MySql.

The project consists of namely four major entities

(i)Hostel,which will include details of hostels allot hostel to students and contains mess detials.

(ii)Student,which will have the details of the students,and mess.

(iii) Mess,it contains the meal details,student’s messid,status of student.

(iv)Meal :-This table includes the detail of meal in breakfast,lunch and dinner.

The services of a Mess Management Syste can include:

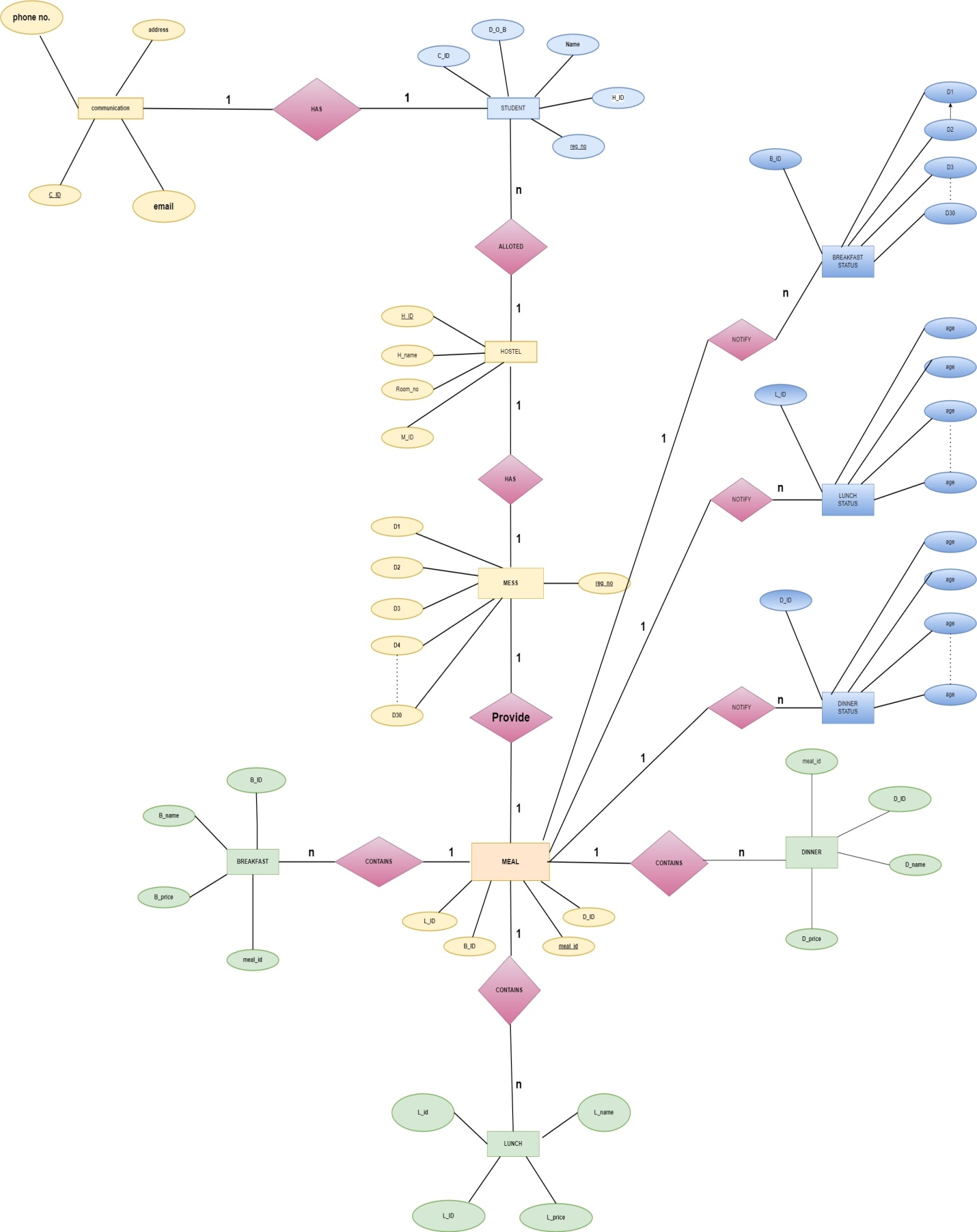
•Alloting the Hostel to a new Student.

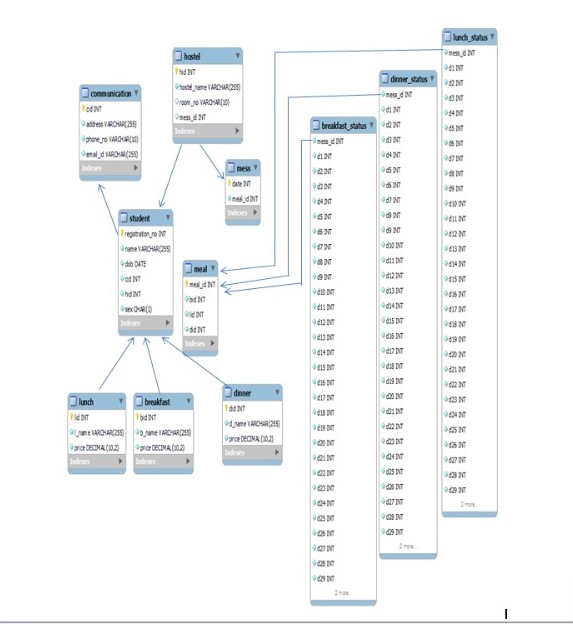
•Preparing the meal to for the month for respective BreakFast,Lunch and Dinner.

•Serving the food to student.

• Checking the status of the student i.e; whether he/she have taken the meal.

•Preparing the Monthly Bill of the Student.





1.**HOSTEL-STUDENT(1:N)**:-A hostel may have many stuents.

2.**HOSTEL-MESS(1:1)**:-A hostel will have only one mess.

3.**HOSTEL-MEAL(1:N)**:-A hostel provides many Meals.

4.**MEAL-BREAKFAST(1:N):-**A meal contains many breakfast items.

5.**MEAL-LUNCH(1:N)**:-A meal contains many lunch items.

6.**MEAL-BREAKFAST\_STATUS(1:N)**:-A meal contains multiple breakfast\_status.

7.**MEAL-LUNCH\_STATUS(1:N)**:-A meal contains multiple lunch\_status.

8.**MEAL-DINNER\_STATUS(1:N)**:-A meal contains multiple dinner\_status.

9.**STUDENT-COMMUNICATION(1:1)**:-A student may have related one communication entity

**Student** : To determine the normal forms (NF) for the given set of attributes reg\_id, name, D\_O\_B, H\_ID, and C\_ID, we need to analyze the functional dependencies and dependencies among the attributes. Based on the provided information, let's examine each normal form:

1. First Normal Form (1NF): • 1NF requires that each attribute in a relation must have atomic values (no multivalued attributes or repeating groups).
2. • From the given set of attributes, it appears that each attribute contains atomic values.
3. • Therefore, the relation satisfies 1NF.

2. Second Normal Form (2NF):

• 2NF states that the relation should be in 1NF and there should be no partial dependencies.

• To identify partial dependencies, we need to know the functional dependencies between attributes.

• However, the provided information does not include any explicit functional dependencies.

• Without explicit dependencies, we cannot determine if there are any partial dependencies.

• Assuming there are no partial dependencies, we can consider the relation to satisfy 2NF. 3. Third Normal Form (3NF):

• 3NF states that the relation should be in 2NF, and there should be no transitive dependencies.

• To identify transitive dependencies, we need to know the functional dependencies between attributes. • From the given information, we can assume the following functional dependencies:

• reg\_id → name • reg\_id → D\_O\_B • reg\_id → H\_ID • reg\_id → C\_ID

• There are no transitive dependencies present since each non-key attribute depends solely on the primary key attribute, reg\_id.

• Therefore, the relation satisfies 3NF. 4. Boyce-Codd Normal Form (BCNF):

• BCNF is a stronger form of normalization that eliminates all non-trivial dependencies in a relation.

• To determine BCNF, we need to identify the candidate keys and non-trivial dependencies.

• From the given information, assuming reg\_id is the primary key, the candidate key is {reg\_id}.

• The functional dependencies are:

• reg\_id → name

• reg\_id → D\_O\_B

• reg\_id → H\_ID

• reg\_id → C\_ID

• The dependencies are non-trivial, and each non-key attribute depends on the candidate key.

• Therefore, the relation satisfies BCNF. In summary, based on the provided information and assumptions, the relation with attributes reg\_id, name, D\_O\_B, H\_ID, and C\_ID satisfies 1NF, 2NF, 3NF, and BCNF. Hostel : 1. First Normal Form (1NF): • From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming H\_ID is the candidate key, we have the following functional dependencies:

• H\_ID -> H\_name, M\_ID, room\_no

• There are no partial dependencies since each non-key attribute depends on the entire candidate key.

• Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming H\_ID is the candidate key, we have the following functional dependencies:

• H\_ID -> H\_name, M\_ID, room\_no • There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming H\_ID is the candidate key, the functional dependencies are as follows:

• H\_ID -> H\_name, M\_ID, room\_no

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies H\_ID -> H\_name, M\_ID, room\_no, the relation satisfies 1NF, 2NF, 3NF, and BCNF. communication: 1. First Normal Form (1NF):

• From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming C\_ID is the candidate key; we have the following functional dependencies:

• C\_ID -> phone\_no, email, address

• There are no partial dependencies since each non-key attribute depends on the entire candidate key.

• Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming C\_ID is the candidate key, we have the following functional dependencies:

• C\_ID -> phone\_no, email, address

• There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming C\_ID is the candidate key, the functional dependencies are as follows:

• C\_ID -> phone\_no, email, address

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies C\_ID -> phone\_no, email, address, the relation satisfies 1NF, 2NF, 3NF, and BCNF. Mess: 1. First Normal Form (1NF):

• From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming reg\_id is the candidate key, we have the following functional dependencies:

• reg\_id -> D1, D2, D3, ..., D31

• There are no partial dependencies since each non-key attribute depends on the entire candidate key. • Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming reg\_id is the candidate key, we have the following functional dependencies:

• reg\_id -> D1, D2, D3, ..., D31

• There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming reg\_id is the candidate key, the functional dependencies are as follows:

• reg\_id -> D1, D2, D3, ..., D31

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies reg\_id -> D1, D2, D3, ..., D31, the relation satisfies 1NF, 2NF, 3NF, and BCNF. Meal: 1. First Normal Form (1NF):

• From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> B\_ID, D\_ID, L\_ID

• There are no partial dependencies since each non-key attribute depends on the entire candidate key.

• Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> B\_ID, D\_ID, L\_ID

• There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming meal\_id is the candidate key, the functional dependencies are as follows:

• meal\_id -> B\_ID, D\_ID, L\_ID

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies meal\_id -> B\_ID, D\_ID, L\_ID, the relation satisfies 1NF, 2NF, 3NF, and BCNF. Breakfast: 1. First Normal Form (1NF):

• From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> price, B\_ID, B\_name

• There are no partial dependencies since each non-key attribute depends on the entire candidate key.

• Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> price, B\_ID, B\_name

• There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming meal\_id is the candidate key, the functional dependencies are as follows:

• meal\_id -> price, B\_ID, B\_name

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies meal\_id -> price, B\_ID, B\_name, the relation satisfies 1NF, 2NF, 3NF, and BCNF. Lunch: 1. First Normal Form (1NF):

• From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> L\_price, L\_ID, L\_name

• There are no partial dependencies since each non-key attribute depends on the entire candidate key.

• Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> L\_price, L\_ID, L\_name

• There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming meal\_id is the candidate key, the functional dependencies are as follows:

• meal\_id -> L\_price, L\_ID, L\_name

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies meal\_id -> L\_price, L\_ID, L\_name, the relation satisfies 1NF, 2NF, 3NF, and BCNF. Dinner: To evaluate the functional dependencies meal\_id -> D\_price, D\_ID, D\_name, we need to analyze each dependency and assess the normal forms. Here's the analysis: 1. First Normal Form (1NF):

• From the given functional dependencies, it appears that each attribute contains atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> D\_price, D\_ID, D\_name

• There are no partial dependencies since each non-key attribute depends on the entire candidate key.

• Therefore, the functional dependencies satisfy 2NF. 3. Third Normal Form (3NF):

• Assuming meal\_id is the candidate key, we have the following functional dependencies:

• meal\_id -> D\_price, D\_ID, D\_name

• There are no transitive dependencies since each non-key attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF): • Assuming meal\_id is the candidate key, the functional dependencies are as follows:

• meal\_id -> D\_price, D\_ID, D\_name

• There are no non-trivial dependencies since each non-key attribute depends on the candidate key.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies meal\_id -> D\_price, D\_ID, D\_name, the relation satisfies 1NF, 2NF, 3NF, and BCNF. Breakfast Status: 1. First Normal Form (1NF):

• The given functional dependencies do not violate 1NF since each attribute appears to contain atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming B\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, it seems that there is a partial dependency.

• Each attribute D1, D2, D3, ..., D31 depends on B\_ID partially since it depends on a specific day (e.g., D1 depends on B\_ID for the first day, D2 depends on B\_ID for the second day, and so on).

• Therefore, the functional dependencies do not satisfy 2NF due to partial dependencies. 3. Third Normal Form (3NF):

• Assuming B\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, there are no transitive dependencies since each attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming B\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, there are no non-trivial dependencies. • Each attribute D1, D2, D3, ..., D31 depends entirely on the candidate key B\_ID.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies B\_ID -> D1, D2, D3, ..., D31, the relation satisfies 1NF, does not satisfy 2NF, satisfies 3NF, and satisfies BCNF.

Lunch Status :

1. First Normal Form (1NF): a. The given functional dependencies do not violate 1NF since each attribute appears to contain atomic values.

b. Therefore, the functional dependencies satisfy 1NF.

2. Second Normal Form (2NF):

a. Assuming L\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, it seems that there is a partial dependency.

b. Each attribute D1, D2, D3, ..., D31 depends on L\_ID partially since it depends on a specific day (e.g., D1 depends on L\_ID for the first day, D2 depends on L\_ID for the second day, and so on).

c. Therefore, the functional dependencies do not satisfy 2NF due to partial dependencies.

3. Third Normal Form (3NF):

a. Assuming L\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, there are no transitive dependencies since each attribute depends directly on the candidate key.

b. Therefore, the functional dependencies satisfy 3NF.

4. Boyce-Codd Normal Form (BCNF):

a. Assuming L\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, there are no non-trivial dependencies.

b. Each attribute D1, D2, D3, ..., D31 depends entirely on the candidate key L\_ID.

c. Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies L\_ID -> D1, D2, D3, ..., D31, the relation satisfies 1NF, does not satisfy 2NF, satisfies 3NF, and satisfies BCNF.

Dinner Status:

1. First Normal Form (1NF):

• The given functional dependencies do not violate 1NF since each attribute appears to contain atomic values.

• Therefore, the functional dependencies satisfy 1NF. 2. Second Normal Form (2NF):

• Assuming D\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, it seems that there is a partial dependency.

• Each attribute D1, D2, D3, ..., D31 depends on D\_ID partially since it depends on a specific day (e.g., D1 depends on D\_ID for the first day, D2 depends on D\_ID for the second day, and so on).

• Therefore, the functional dependencies do not satisfy 2NF due to partial dependencies. 3. Third Normal Form (3NF):

• Assuming D\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, there are no transitive dependencies since each attribute depends directly on the candidate key.

• Therefore, the functional dependencies satisfy 3NF. 4. Boyce-Codd Normal Form (BCNF):

• Assuming D\_ID is the candidate key, and D1, D2, D3, ..., D31 are the attributes, there are no non-trivial dependencies.

• Each attribute D1, D2, D3, ..., D31 depends entirely on the candidate key D\_ID.

• Therefore, the functional dependencies satisfy BCNF. In summary, based on the given functional dependencies D\_ID -> D1, D2, D3, ..., D31, the relation satisfies 1NF, does not satisfy 2NF, satisfies 3NF, and satisfies BCNF.

1)Student(registeration\_no,name,dob,cid,hid,sex)

+-----------------+-------------------------+------------+-----+-----+-----+

| registration\_no | name | dob | cid | hid | sex |

+-----------------+-------------------------+------------+-----+-----+-----+

| 202301 | Swagatika Hial | 2002-02-16 | 1 | 1 | F |

| 202302 | Harshita Yadav | 2001-11-26 | 2 | 2 | F |

| 202303 | Mamta Singh | 2000-10-02 | 3 | 3 | F |

| 202304 | Shailendra Mohan Sharma | 2002-03-02 | 4 | 4 | M |

| 202305 | Priyanshi Agnihotri | 2008-05-04 | 5 | 5 | F |

+-----------------+-------------------------+------------+-----+-----+-----+

2)Hostel(hid,hostel\_name,room\_no,mess\_id)

+-----+--------------+---------+---------+

| hid | hostel\_name | room\_no | mess\_id |

+-----+--------------+---------+---------+

| 1 | Koyna | 08 | 1 |

| 2 | chandrabhaga | 09 | 2 |

| 3 | chandrabhaga | 10 | 3 |

| 4 | mahi | 26 | 4 |

| 5 | koyna | 01 | 5 |

+-----+--------------+---------+---------+

3)Mess(date,meal\_id)

+------+---------+

| date | meal\_id |

+------+---------+

| 1 | 7 |

| 2 | 5 |

| 3 | 4 |

| 4 | 2 |

| 5 | 4 |

| 6 | 3 |

| 7 | 1 |

| 8 | 2 |

| 9 | 2 |

| 10 | 4 |

| 11 | 3 |

| 12 | 5 |

| 13 | 1 |

| 14 | 3 |

| 15 | 5 |

| 16 | 7 |

| 17 | 4 |

| 18 | 6 |

| 19 | 5 |

| 20 | 3 |

| 21 | 5 |

| 22 | 4 |

| 23 | 7 |

| 24 | 6 |

| 25 | 1 |

| 26 | 3 |

| 27 | 4 |

| 28 | 5 |

| 29 | 4 |

| 30 | 5 |

| 31 | 5 |

+------+---------+

4)Meal(mead\_id,bid,lid,did)

+---------+-----+-----+-----+

| meal\_id | bid | lid | did |

+---------+-----+-----+-----+

| 1 | 1 | 2 | 1 |

| 2 | 2 | 3 | 1 |

| 3 | 2 | 2 | 1 |

| 4 | 3 | 1 | 2 |

| 5 | 1 | 2 | 3 |

| 6 | 3 | 2 | 1 |

| 7 | 2 | 1 | 2 |

5)Lunch(lid,l\_name,price)

+-----+---------------+-------+

| lid | l\_name | price |

+-----+---------------+-------+

| 1 | curry chaawal | 35.00 |

| 2 | dahi baigan | 30.00 |

| 3 | soya bean | 35.00 |)

6)Breakfast(bid,b\_name,price)

+-----+----------------+-------+

| bid | b\_name | price |

+-----+----------------+-------+

| 1 | chana chura | 25.00 |

| 2 | sprout | 25.00 |

| 3 | chhole bhature | 30.00 |

| 4 | poori sabzi | 30.00 |

| 5 | parantha | 20.00 |

7)Dinner(did,d\_name,price)

+-----+----------------------+-------+

| did | d\_name | price |

+-----+----------------------+-------+

| 1 | paneer and rasogulla | 50.00 |

| 2 | aloo katahal | 40.00 |

| 3 | chana and custard | 50.00 |

| 4 | karela | 40.00 |

8)breakfast\_status(mess\_id,d1,d2,d3………d31);

9)lunch\_status(mess\_id,d1,d2,d3…………….d31);

10)dinner\_status(mess\_id,d1,d2,d3………….d31);

Student table:

CREATE TABLE student (

registration\_no INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

dob DATE NOT NULL,

cid INT NOT NULL,

hid INT NOT NULL,

sex CHAR(1) NOT NULL,

PRIMARY KEY (registration\_no)

);

Communication table:

CREATE TABLE communication (

cid INT NOT NULL,

address VARCHAR(255) NOT NULL,

phone\_no VARCHAR(10) NOT NULL,

email\_id VARCHAR(255) NOT NULL,

PRIMARY KEY (cid)

);

Hostel table:

CREATE TABLE hostel (

hid INT NOT NULL AUTO\_INCREMENT,

hostel\_name VARCHAR(255) NOT NULL,

room\_no VARCHAR(10),

mess\_id INT,

PRIMARY KEY (hid)

);

Mess table:

CREATE TABLE mess (

date INT NOT NULL AUTO\_INCREMENT PRIMARY KEY MAXVALUE 31,

meal\_id INT NOT NULL

);

Breakfast status table:

CREATE TABLE breakfast\_cost (

mess\_id INT NOT NULL,

d1 BOOLEAN NOT NULL,

d2 BOOLEAN NOT NULL,

d3 BOOLEAN NOT NULL,

d4 BOOLEAN NOT NULL,

d5 BOOLEAN NOT NULL,

d6 BOOLEAN NOT NULL,

d7 BOOLEAN NOT NULL,

d8 BOOLEAN NOT NULL,

d9 BOOLEAN NOT NULL,

d10 BOOLEAN NOT NULL,

d11 BOOLEAN NOT NULL,

d12 BOOLEAN NOT NULL,

d13 BOOLEAN NOT NULL,

d14 BOOLEAN NOT NULL,

d15 BOOLEAN NOT NULL,

d16 BOOLEAN NOT NULL,

d17 BOOLEAN NOT NULL,

d18 BOOLEAN NOT NULL,

d19 BOOLEAN NOT NULL,

d20 BOOLEAN NOT NULL,

d21 BOOLEAN NOT NULL,

d22 BOOLEAN NOT NULL,

d23 BOOLEAN NOT NULL,

d24 BOOLEAN NOT NULL,

d25 BOOLEAN NOT NULL,

d26 BOOLEAN NOT NULL,

d27 BOOLEAN NOT NULL,

d28 BOOLEAN NOT NULL,

d29 BOOLEAN NOT NULL,

d30 BOOLEAN NOT NULL,

d31 BOOLEAN NOT NULL,

PRIMARY KEY (mess\_id)

);

Lunch status table:

CREATE TABLE lunch\_status (

mess\_id INT NOT NULL,

d1 INT NOT NULL,

d2 INT NOT NULL,

d3 INT NOT NULL,

d4 INT NOT NULL,

d5 INT NOT NULL,

d6 INT NOT NULL,

d7 INT NOT NULL,

d8 INT NOT NULL,

d9 INT NOT NULL,

d10 INT NOT NULL,

d11 INT NOT NULL,

d12 INT NOT NULL,

d13 INT NOT NULL,

d14 INT NOT NULL,

d15 INT NOT NULL,

d16 INT NOT NULL,

d17 INT NOT NULL,

d18 INT NOT NULL,

d19 INT NOT NULL,

d20 INT NOT NULL,

d21 INT NOT NULL,

d22 INT NOT NULL,

d23 INT NOT NULL,

d24 INT NOT NULL,

d25 INT NOT NULL,

d26 INT NOT NULL,

d27 INT NOT NULL,

d28 INT NOT NULL,

d29 INT NOT NULL,

d30 INT NOT NULL,

d31 INT NOT NULL

);

Dinner status table:

CREATE TABLE dinner\_status (

mess\_id INT NOT NULL,

d1 INT NOT NULL,

d2 INT NOT NULL,

d3 INT NOT NULL,

d4 INT NOT NULL,

d5 INT NOT NULL,

d6 INT NOT NULL,

d7 INT NOT NULL,

d8 INT NOT NULL,

d9 INT NOT NULL,

d10 INT NOT NULL,

d11 INT NOT NULL,

d12 INT NOT NULL,

d13 INT NOT NULL,

d14 INT NOT NULL,

d15 INT NOT NULL,

d16 INT NOT NULL,

d17 INT NOT NULL,

d18 INT NOT NULL,

d19 INT NOT NULL,

d20 INT NOT NULL,

d21 INT NOT NULL,

d22 INT NOT NULL,

d23 INT NOT NULL,

d24 INT NOT NULL,

d25 INT NOT NULL,

d26 INT NOT NULL,

d27 INT NOT NULL,

d28 INT NOT NULL,

d29 INT NOT NULL,

d30 INT NOT NULL,

d31 INT NOT NULL

);

Meal Table:

CREATE TABLE meal (

meal\_id INT NOT NULL AUTO\_INCREMENT,

bid INT NOT NULL,

lid INT NOT NULL,

did INT NOT NULL,

PRIMARY KEY (meal\_id)

);

Breakfast Table:

CREATE TABLE breakfast (

bid INT NOT NULL AUTO\_INCREMENT,

b\_name VARCHAR(255) NOT NULL,

price DECIMAL(10,2) NOT NULL,

PRIMARY KEY (bid)

);

Lunch Table:

CREATE TABLE lunch (

lid INT NOT NULL AUTO\_INCREMENT,

l\_name VARCHAR(255) NOT NULL,

price DECIMAL(10,2) NOT NULL,

PRIMARY KEY (lid)

);

Dinner Table:

CREATE TABLE dinner (

did INT NOT NULL AUTO\_INCREMENT,

d\_name VARCHAR(255) NOT NULL,

price DECIMAL(10,2) NOT NULL,

PRIMARY KEY (did)

);

In this project we have created one database which is easy to access and user friendly.

The database keeps a backup of the mess management data which includes their

details. A mess management is a professional app which assists with the messmeals,accurate bills etc.