MealManager

Team-2

Chaithra Lakshmi Sathyanarayana (Student ID: 012447565)
Thaijasa Badrinath Vijendranath (Student ID: 012470835)
Vineela Velicheti (Student ID: 012445186)

INTRODUCTION

- Interface between restaurants, dieticians and customers
- Customers:
  - People looking for meal subscriptions - breakfast, lunch and dinner
  - Option to enroll for a personalized diet plan planned by a affiliated dietician of their choosing
- Restaurants:
  - Orders details known in advance
  - Can offer food for lower rates and still profit
- Dieticians: Provide personalized diet plans
- Admin: Creates restaurant and dieticians users
- NoSQL MongoDB is used for storing contact form data

CUSTOMER

Major functionalities of a customer:

Customer without diet plan:

- Register to the application (Enter details and make payment)
- Login to the application with credentials
- Select the meals based on meal plan
- Place the order for selected meal and specify pick-up time
- Rate items
- Logout of the application

Customer with a diet plan:

- Register to the application (Enter details and make payment)
- Login to the application
- Select the meals based on meal plan and diet plan
- Place the order for selected meal and specify pick-up time
- View and update diet plan progress
- Rate items
- Rate dieticians
- Logout of the application

RESTAURANT

Major functionalities of a restaurant:

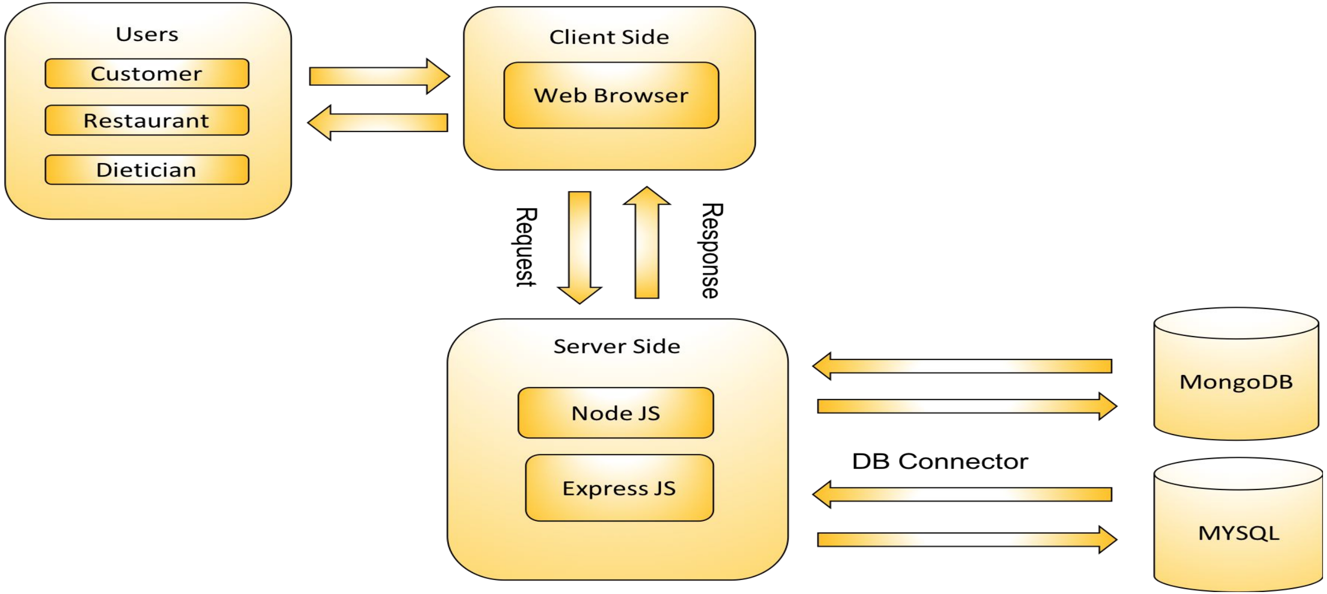
- Get registered to the application
- Login in to the application with credentials
- Add items with nutritional information to the menu
- Receive and view item orders from customers
- Update/delete items from menu
- Generate sales report for a specific time period
- Log out of the application

DIETICIAN

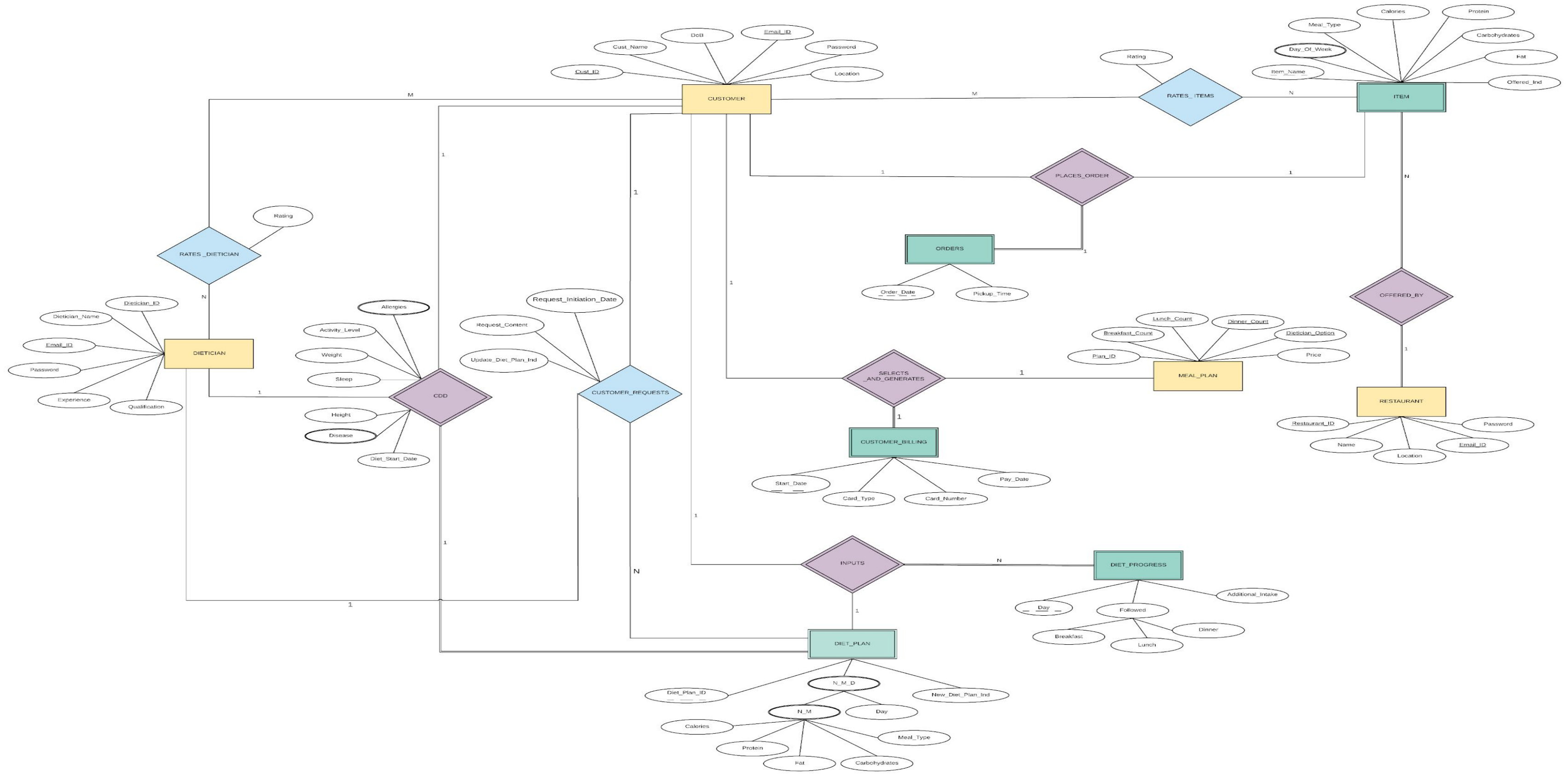
Major functionalities of a dietician:

- Get registered to the application
- Login in to the application with credentials
- Receive diet plan requests from customer
- Formulate diet plan for the requests and update them in the application
- View enrolled customer’s diet progress
- Cater to change requests orders from the customer
- Logout of the application

ARCHITECTURE

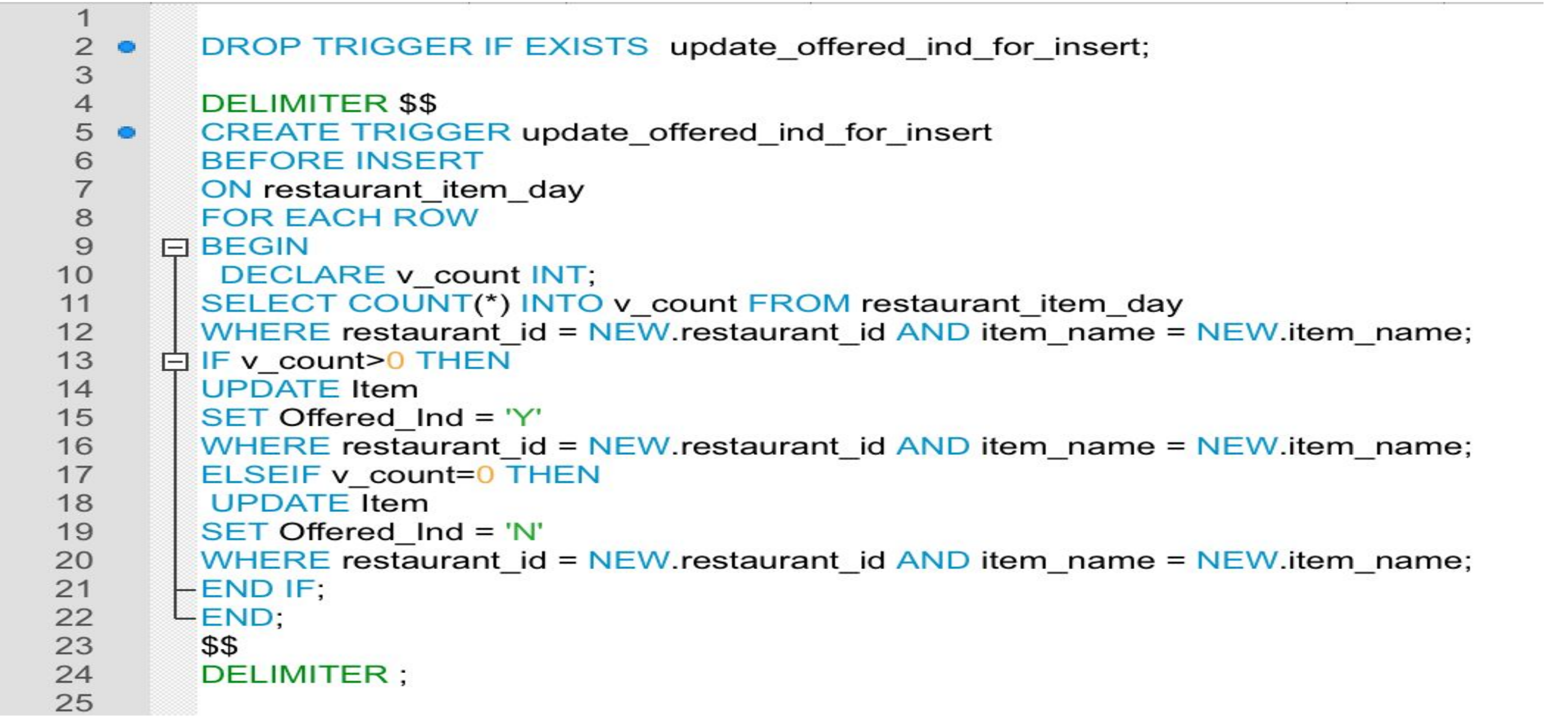


ER DIAGRAM



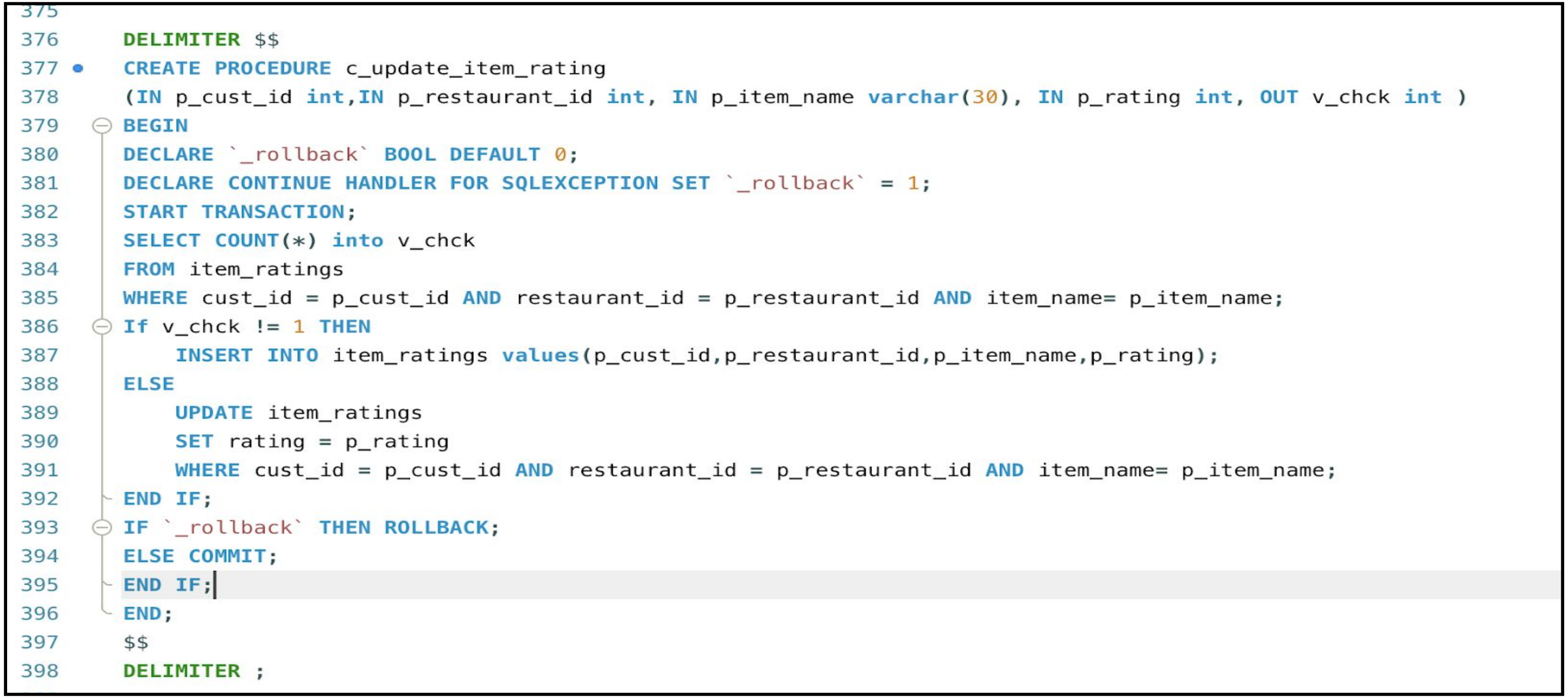
CUSTOMER
Cust_ID
Cust_Name
DoB
Email_ID
Password
Location
M
RATES_ ITEMS
Rating
M
N
ITEM
Meal_Type
Calories
Protein
Carbohydrates
Day_Of_Week
Fat
Item_Name
Offered_Ind
N
PLACES_ORDER
1
1
1
ORDERS
Order_Date
Pickup_Time
OFFERED_BY
1
RESTAURANT
Restaurant_ID
Name
Location
Email_ID
Password
RATES _DIETICIAN
Rating
N
DIETICIAN
Dietician_ID
Dietician_Name
Email_ID
Password
Experience
Qualification
1
1
CDD
Allergies
Activity_Level
Weight
Sleep
Height
Disease
Diet_Start_Date
1
CUSTOMER_REQUESTS
Request_Initiation_Date
Request_Content
Update_Diet_Plan_Ind
N
1
SELECTS _AND_GENERATES
1
1
MEAL_PLAN
Lunch_Count
Dinner_Count
Breakfast_Count
Dietician_Option
Plan_ID
Price
CUSTOMER_BILLING
Start_Date
Card_Type
Card_Number
Pay_Date
1
1
INPUTS
N
DIET_PROGRESS
Day
Followed
Additional_Intake
Breakfast
Lunch
Dinner
1
1
DIET_PLAN
Diet_Plan_ID
N_M_D
New_Diet_Plan_Ind
N_M
Day
Calories
Protein
Fat
Carbohydrates
Meal_Type

TRIGGER



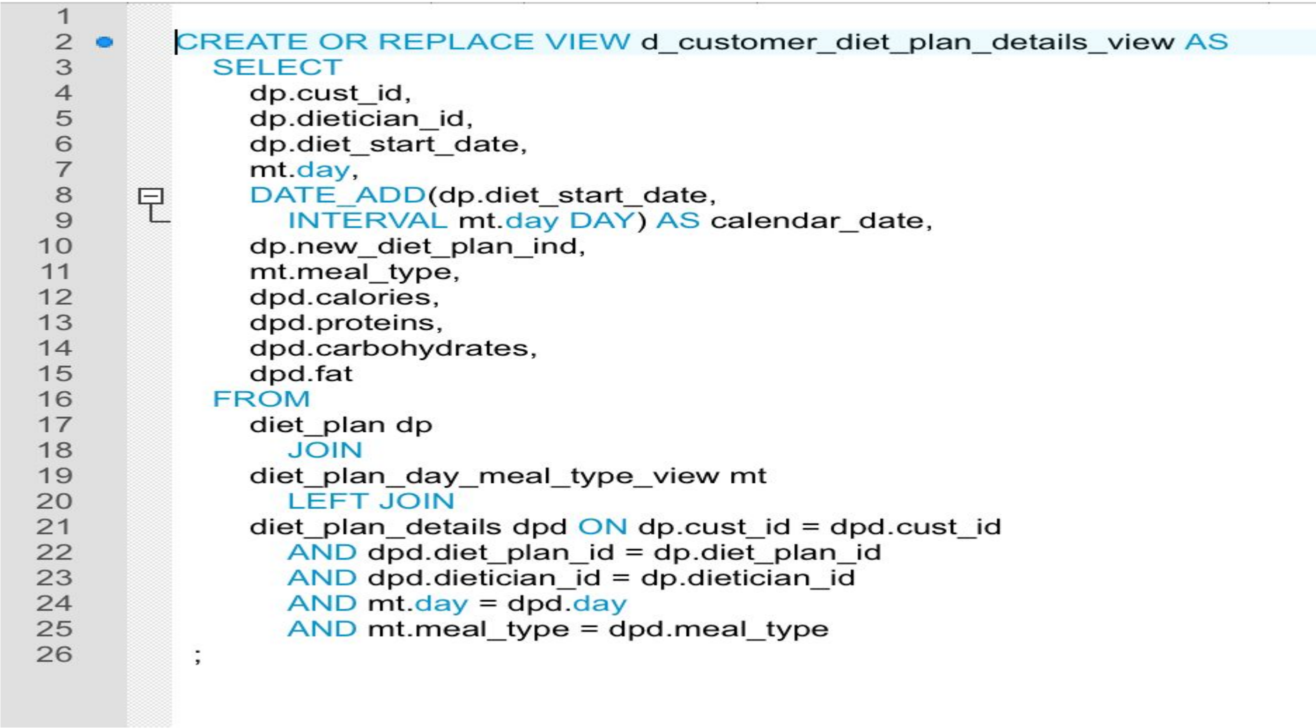
```

DROP TRIGGER IF EXISTS  update_offered_ind_for_insert;

DELIMITER $$
CREATE TRIGGER update_offered_ind_for_insert
BEFORE INSERT
ON restaurant_item_day
FOR EACH ROW
BEGIN
 DECLARE v_count INT;
SELECT COUNT(*) INTO v_count FROM restaurant_item_day
WHERE restaurant_id = NEW.restaurant_id AND item_name = NEW.item_name;
IF v_count>0 THEN
UPDATE Item
SET Offered_Ind = 'Y'
WHERE restaurant_id = NEW.restaurant_id AND item_name = NEW.item_name;
ELSEIF v_count=0 THEN
 UPDATE Item
SET Offered_Ind = 'N'
WHERE restaurant_id = NEW.restaurant_id AND item_name = NEW.item_name;
END IF;
END;
$$
DELIMITER ;

```

STORED PROCEDURE



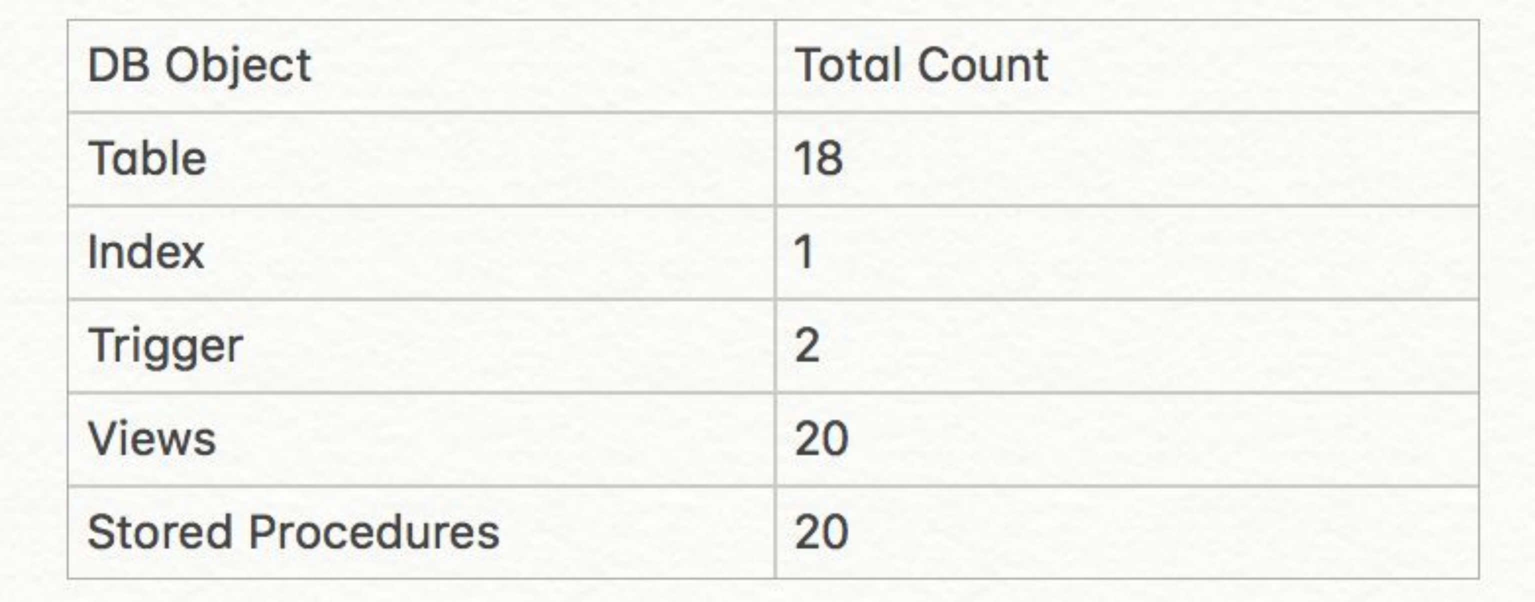
```
DELIMITER $$
CREATE PROCEDURE c_update_item_rating
(IN p_cust_id int,IN p_restaurant_id int, IN p_item_name varchar(30), IN p_rating int, OUT v_chck int )
BEGIN
DECLARE `_rollback` BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
START TRANSACTION;
SELECT COUNT(*) into v_chck
FROM item_ratings
WHERE cust_id = p_cust_id AND restaurant_id = p_restaurant_id AND item_name= p_item_name;
If v_chck != 1 THEN
    INSERT INTO item_ratings values(p_cust_id,p_restaurant_id,p_item_name,p_rating);
ELSE
    UPDATE item_ratings
    SET rating = p_rating
    WHERE cust_id = p_cust_id AND restaurant_id = p_restaurant_id AND item_name= p_item_name;
END IF;
IF `_rollback` THEN ROLLBACK;
ELSE COMMIT;
END IF;
END;
$$
DELIMITER ;
```

VIEW



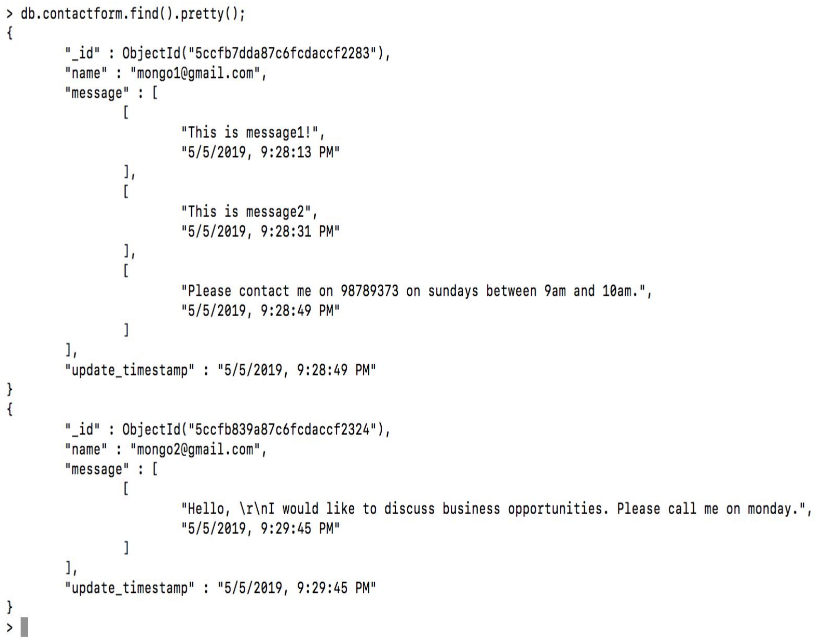
```
CREATE OR REPLACE VIEW d_customer_diet_plan_details_view AS
  SELECT
    dp.cust_id,
    dp.dietician_id,
    dp.diet_start_date,
    mt.day,
    DATE_ADD(dp.diet_start_date,
      INTERVAL mt.day DAY) AS calendar_date,
    dp.new_diet_plan_ind,
    mt.meal_type,
    dpd.calories,
    dpd.proteins,
    dpd.carbohydrates,
    dpd.fat
  FROM
    diet_plan dp
      JOIN
    diet_plan_day_meal_type_view mt
      LEFT JOIN
    diet_plan_details dpd ON dp.cust_id = dpd.cust_id
      AND dpd.diet_plan_id = dp.diet_plan_id
      AND dpd.dietician_id = dp.dietician_id
      AND mt.day = dpd.day
      AND mt.meal_type = dpd.meal_type
 ;
```

DB OBJECTS

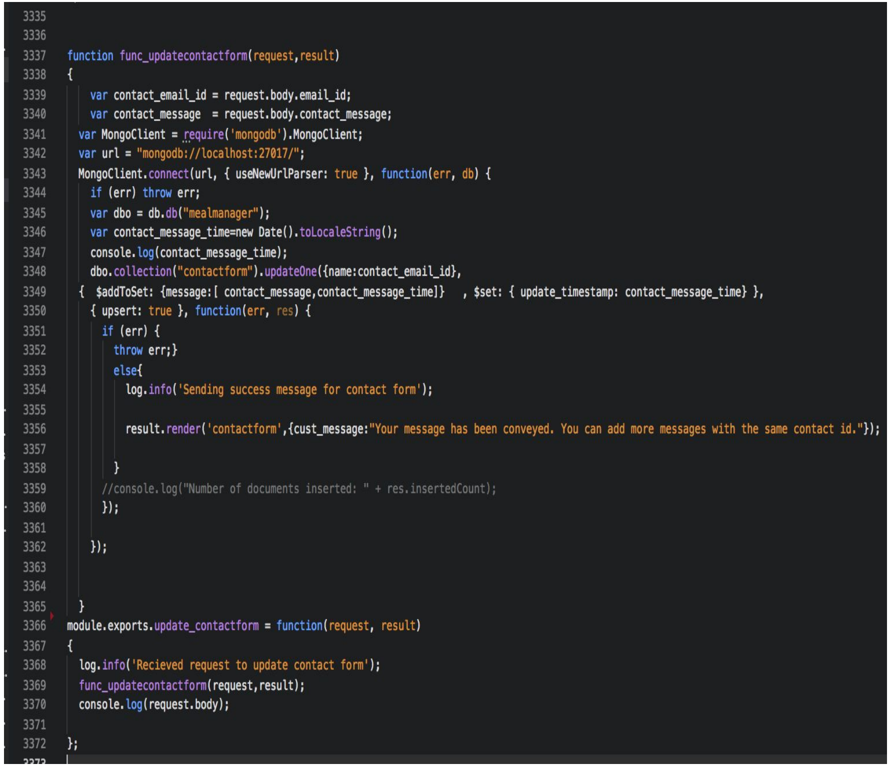


DB Object	Total Count
Table	18
Index	1
Trigger	2
Views	20
Stored Procedures	20

NoSQL: MONGODB for Contact Form



```
> db.contactform.find().pretty();
{
        "_id" : ObjectId("5ccfb7dda87c6fcdaccf2283"),
        "name" : "mongo1@gmail.com",
        "message" : [
                [
                        "This is message1!",
                        "5/5/2019, 9:28:13 PM"
                ],
                [
                        "This is message2",
                        "5/5/2019, 9:28:31 PM"
                ],
                [
                        "Please contact me on 98789373 on sundays between 9am and 10am.",
                        "5/5/2019, 9:28:49 PM"
                ]
        ],
        "update_timestamp" : "5/5/2019, 9:28:49 PM"
}
{
        "_id" : ObjectId("5ccfb839a87c6fcdaccf2324"),
        "name" : "mongo2@gmail.com",
        "message" : [
                [
                        "Hello, \r\nI would like to discuss business opportunities. Please call me on monday.",
                        "5/5/2019, 9:29:45 PM"
                ]
        ],
        "update_timestamp" : "5/5/2019, 9:29:45 PM"
}
> 
```



```


function func_updatecontactform(request,result)
{
    var contact_email_id = request.body.email_id;
    var contact_message  = request.body.contact_message;
  var MongoClient = require('mongodb').MongoClient;
  var url = "mongodb://localhost:27017/";
  MongoClient.connect(url, { useNewUrlParser: true }, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mealmanager");
    var contact_message_time=new Date().toLocaleString();
    console.log(contact_message_time);
    dbo.collection("contactform").updateOne({name:contact_email_id},
  {  $addToSet: {message:[ contact_message,contact_message_time]}   , $set: { update_timestamp: contact_message_time} },
    { upsert: true }, function(err, res) {
      if (err) {
        throw err;}
        else{
          log.info('Sending success message for contact form');

          result.render('contactform',{cust_message:"Your message has been conveyed. You can add more messages with the same contact id."});

        }
      //console.log("Number of documents inserted: " + res.insertedCount);
      });

    });


  }
module.exports.update_contactform = function(request, result)
{
  log.info('Recieved request to update contact form');
  func_updatecontactform(request,result);
  console.log(request.body);

};
```

RETROSPECTIVE

What we did well?

- Implementation and integration from frontend to backend of the application
- Team work and communication

What we learned new?

- Understand forming complex SQL stored procedures
- Using session concepts in application and generating logs during application run

What we can improve on?

- Gathering and understanding business requirements

Demo Time!

Thank You!