PROJECT -2 REPORT

DATABASE DESIGN

SPRING 2019

Professor

Kong Li

MEAL MANAGER

Date of Submission

05th May 2019

Submitted by

Team #2

Chaithra Lakshmi Sathyanarayana (Student ID: 012447565)

Thaijasa Badrinath Vijendranath (Student ID: 012470835)

Vineela Velicheti (Student ID: 012445186)

1. Choice of Database Project

The mini world chosen as part of this project - Meal Manager, provides an interface between restaurants, dieticians and customers. The potential customers to this website will be people looking for meal subscriptions - breakfast, lunch and dinner, along with an option to enroll for a personalized diet plan.

With the order details known in advance, restaurant managers will be able to plan efficiently and thereby avoid food wastage. Consequently, they will offer lower prices as compared to the traditional approach. Customers, on the other hand, will get access to good food at a lower price and also benefit from skipping wait lines. Health conscious customers can choose from a group of affiliated dieticians to get personalized diet plans, and customers can choose meals based on their diet plan.

1.1. Business Rules

1. Any restaurant/ Dietician will be registered to the application only by the admin.
2. Each customer will be provided with the following choices to select from
    - Meal type: Breakfast and/or Lunch and/or Dinner
    - Number of meals in a month: 15/30
    - Enroll with a dietician: yes/no
3. Restaurant orders must be placed by customers on the previous day before 11:59 PM.
4. In a day, a customer will be able to place an order only for one item per meal type.
5. Dietician will need a maximum of 2 days to issue a new diet plan.
6. A customer enrolled with dietician should wait for a minimum of 3 days after the payment, to start the diet plan.
7. A customer enrolled with dietician will by default be registered under all meal types with 30 as count for each.
8. If a diet plan change request is initiated by a customer, he/she should wait for 3 days to start the updated diet plan.
9. A restaurant can offer an item under only one meal type.
10. As part of the update operation, a restaurant can change only the days on which the items are being offered. Any changes with respect to the other attributes of the item can be done by the combination of delete and add operations. For example, if a restaurant wants to change name of item from “A” to “B”, restaurant should first delete item A and then add new entry with name as item B.

1.2. High Level Functioning of the Meal Manager Application

Customers to the Meal Manager application will at first create an account by giving the basic registration details such as name, email ID, password. During the next registration step, customers will be displayed with options –either to get enrolled with a dietician and a diet plan or not to get enrolled with a dietician. The next and final page of the registration will be based on the option customer chooses in previous section. Customer who chooses to get registered with a dietician will be prompted to enter a few of his lifestyle related details such as weight, height, any diseases that he/she is diagnosed with etc. After entering the respective data, he will make the payment to complete the registration process. On the other hand, if a customer chooses not to have dietician, he will not be prompted to enter any lifestyle related data. He will make the payment to complete the registration process.

Any restaurant will be registered in to the application by admin of the application. The location to which restaurant belongs is entered along with the other basic details like email and password. When the restaurant adds items to its menu, nutritional information calories, carbohydrate content, fat content, protein content will be added for each item.

Similarly, any dietician will be registered to the application by the admin of the application. A dietician will use the customer given lifestyle information to formulate the diet plan and uploads the plan for that customer. A diet plan will include the following intake the customer can take for each day (15 days or 30 days based on the option selected) and for each meal type: calories, carbohydrate content, fat content, protein content.

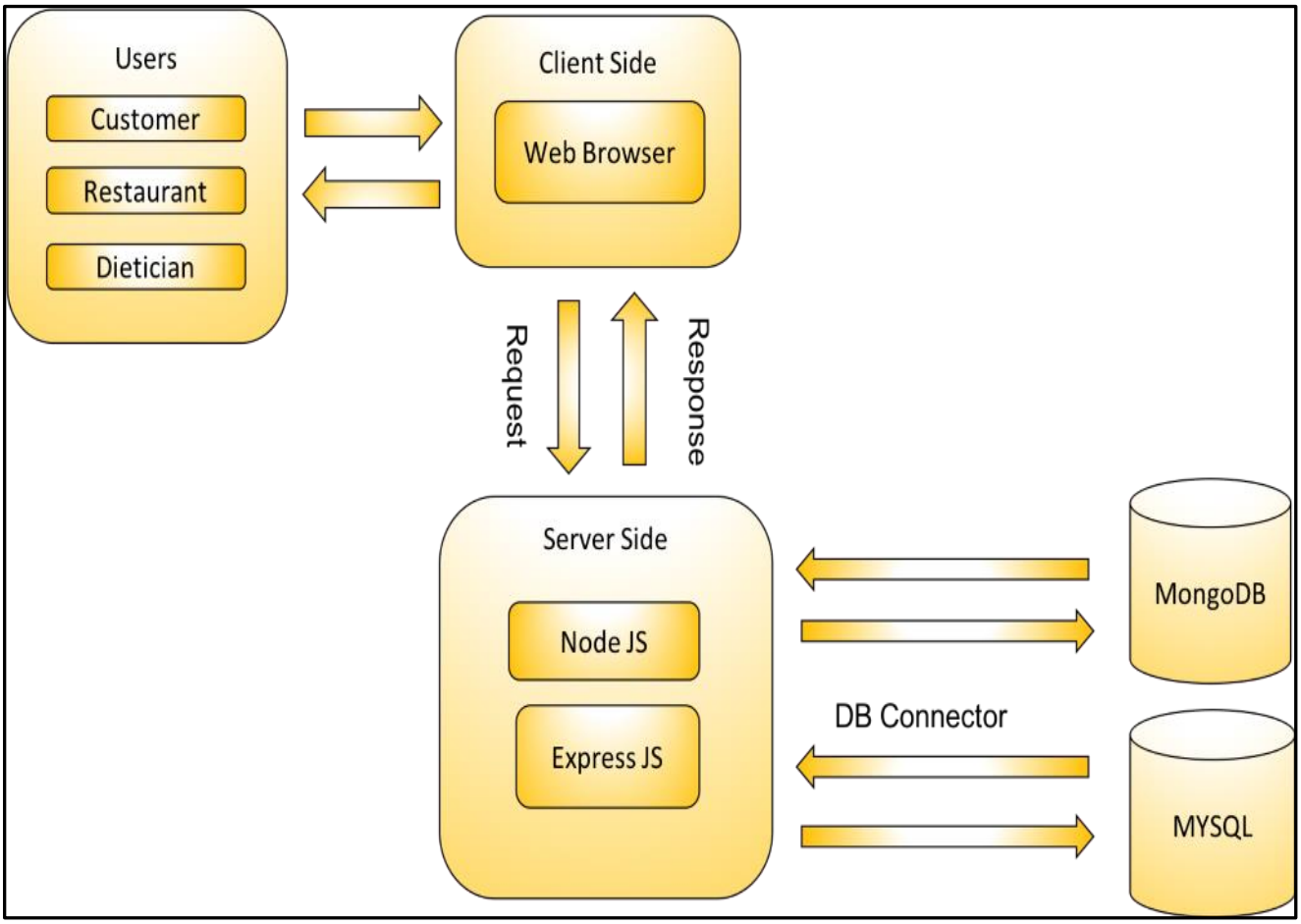
On any given day, a customer will be able to view menu items offered by the restaurant for the next day. Only those restaurants whose location match with the location of the customers will be displayed. Also, for each of the three meal types, breakfast/lunch/dinner, a customer will be able to view items from only those meal types to which he is enrolled for during the registration. Additionally, the menu items displayed will be further filtered for a customer enrolled with a dietician. To do this, application considers the prescribed diet plan provided by the dietician for that customer and matches with the item's nutritional information. Only those items which match the criteria will be displayed as item results.

As a last step, the customer chooses the items from the list displayed for him, selects the pick-up time slot and places the order.

2. Tech Stack

- Database Engine: MySQL, MongoDB
- Database Application Technologies
    - Front -end Technologies: HTML5, CSS3, JavaScript
    - Frameworks: Bootstrap, Express.js, ejs
    - Server-side Technologies: Node.js
- Operating System: Mac OS
- Languages: Javascript, SQL, HTML5, CSS3

3. Architecture



4. Final list of functionalities

4.1. Modifications from ERD

- Removed SELECTS_AND_PAYS_FOR binary relation and replaced it with SELECTS_AND_GENERATES ternary relation. This is done to maintain history of all the previous plan details chosen by all customers.
- Created a new weak entity CUSTOMER_BILLING to maintain all the previous billing information.
- Added new relation CHANGE_REQUEST where a customer can request to change his diet plan.
- Added New_Diet_Plan_Ind attribute to DIET_PLAN entity. This acts as an indicator with values as Y or N where Y implies that diet_plan_id is still new and not addressed by the dietician and N implies that diet_plan_ID is not new and has been addressed by the dietician already.
- Removed Quantity attribute from ORDERS relation
- Removed Calendar_Date from DIET_PROGRESS and made "Day" attribute as a partial key.
- Added Offered_Ind attribute in ITEM entity. This acts as an indicator with values as Y or N where Y implies item is offered currently by the restaurant and N implies item is not offered currently by restaurant for customers.
- Removed ORDERS binary relation and replaced with PLACES_ORDER ternary relation. This helps to maintain the order history.
- Created a new weak entity ORDERS to maintain order history.

4.2. Functionalities and Status of Completion

4.2.1. Admin

Functionality	Planned/New	Status
Admin must be able to log in	Planned	Complete
Admin must be able to register dieticians.	Planned	Complete
Admin must be able to register restaurants	Planned	Complete
Admin must be able to log out successfully	Planned	Complete

4.2.2. Customer

Functionality	Planned/New	Status
Customer must be able to register	Planned	Complete
Customer who has registered must be able to log in	Planned	Complete
Customer may choose a meal plan	Planned	Complete
Customer must be able to view the next day's menu	Planned	Complete
Customer may select items from one or more restaurants	Planned	Complete
Customer may order an item from the menu of the selected restaurant.	Planned	Complete
Customer must be able to choose a pick-up time slot while placing an order	Planned	Complete
Customer must be able to rate items that he has ordered before	Planned	Complete
Customer should be able to see item ratings	Planned	Complete
Customer may enroll for a dietician	Planned	Complete
Customer with a dietician must have a diet plan	Planned	Complete
Customer must be able to see his current diet plan	Planned	Complete
Customer with a dietician may input his progress for each day in the diet plan	Planned	Complete
Customer must be able to see his current diet progress	Planned	Complete
Customer may request for a change of diet plan from the enrolled dietician	Planned	Complete
Customer must be able to view his diet history	New	Complete
Customer must be able to rate dieticians with whom he had enrolled before	Planned	Complete
Customer should be able to see dietician ratings	Planned	Complete
Customer must be able to see his past item orders	New	Complete
Customer must be able to log out of the application	Planned	Complete

4.2.3. Restaurant

Functionality	Planned/New	Status
Restaurant must be registered via Admin	Planned	Complete
Restaurant must be able to log in	Planned	Complete
Restaurant must be able to view its item list as part of it's menu	Planned	Complete
Restaurant must be able to add an item to their menu	Planned	Complete
Restaurant must be able to update/delete an item from it's menu	Planned	Complete
Restaurant may access customer's orders for the current day	Planned	Complete
Restaurant may access customer's orders for the next day	Planned	Complete
Restaurant may request for sales report for a specific time period	Planned	Complete
Restaurant must be able to log out of the application	Planned	Complete

4.2.4. Dietician

Functionality	Planned/New	Status
Dietician must be registered via Admin	Planned	Complete
Dietician must be able to log in	Planned	Complete
Dietician must be able to view the list of customers currently enrolled under him/her	Planned	Complete
Dietician must be able to view the current diet plan of all enrolled customers	Planned	Complete
Dietician must be able to view current diet progress of all enrolled customers	Planned	Complete
Dietician must be able to view new diet plan requests from the customers	Planned	Complete

Dietician must be able to view new diet plan change requests from the customers	Planned	Complete
Dietician must provide diet plan for all new diet plan requests from customers	Planned	Complete
Dietician must be able to change customer's diet plan on request	Planned	Complete
Dietician must be able to log out of the application	Planned	Complete

5. Task Distribution

5.1. Timeline

Components	Duration
Proposal	February 20th
ERD	March 10th
ERD Changes	March 15th to March 20th
Normalization	March 22nd to March 29th; April 18th to April 22nd
Table Scripts	March 30th to April 7th
Insert Statements	March 30th to April 7th
Stored Procedures	April 8th to April 14th; April 18th to April 22nd
Views	April 8th to April 14th; April 18th to April 22nd
Trigger	April 8th to April 14th; April 18th to April 22nd
Index	March 30th to April 7th
Password Encryption	April 28th to April 30th
Multi statement transactions	April 28th to April 30th
UI Design Planning	March 20th - March21st; April 18th to April 22nd
Node JS installation and learning	March 30th to April 7th
Session and Logs	March 30th to April 7th
EJS Pages Creation	April 8th to April 14th; April 18th to April 22nd
CSS Styling	April 8th to April 14th; April 18th to April 22nd
Mongo DB	April 22nd to April 30th

Screenshots from DB and application	May 1st to May 5th
Normalization justification	May 1st to May 5th
Functionalities	May 1st to May 5th
Test cases	May 1st to May 5th
Report	May 1st to May 5th
Presentation slides	May 1st to May 5th

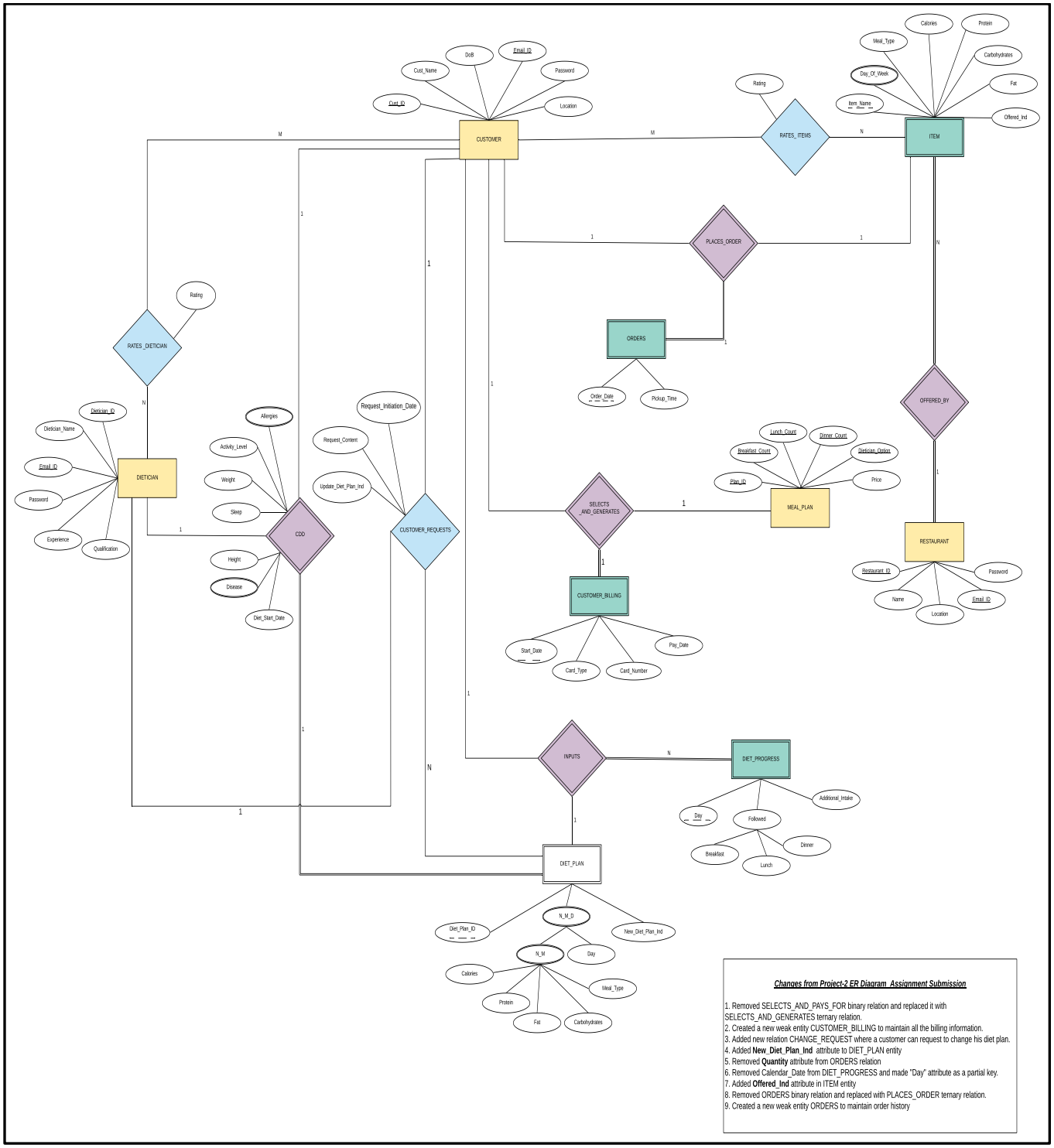
5.2. Individual Contributions

Student Name	Component	Task
Chaithra Lakshmi Sathyanarayana (ID: 012447565)	DB	Normalization, Table Scripts, Insert Statements, Stored Procedures, Views, Trigger, Index, Password Encryption, Multi statement transactions, MongoDB
	UI	Restaurant (view menu, add menu), Dietician - Full
	Documentation	Screenshots from DB and application, Normalization justification, Functionalities, Test cases, Report, Presentation Slides
Thaijasa Badrinath Vijendranath (ID: 012470835)	DB	Normalization, Table Scripts, Insert Statements, Stored Procedures, Views, Trigger, Index, Password Encryption, Multi statement transactions, MongoDB
	UI	Restaurant (update/delete menu, view report, view today's orders, view tomorrows orders), Customer Registration, Admin - Full
	Documentation	Screenshots from DB and application, Normalization justification, Functionalities, Test cases, Report, Presentation Slides
Vineela Velicheti (ID: 012445186)	DB	Normalization, Table Scripts, Insert Statements, Stored Procedures, Views, Trigger, Index, Password Encryption, Multi statement transactions, MongoDB
	UI	Customer - Full

	Documentation	Screenshots from DB and application, Normalization justification, Functionalities, Test cases, Report, Presentation Slides

6. Final Design of Database Portion

6.1. ER Diagram



Changes from Project-2 ER Diagram Assignment Submission

1. Removed SELECTS_AND_PAYS_FOR binary relation and replaced it with SELECTS_AND_GENERATES ternary relation.
2. Created a new weak entity CUSTOMER_BILLING to maintain all the billing information.
3. Added new relation CHANGE_REQUEST where a customer can request to change his diet plan.
4. Added **New_Diet_Plan_Ind** attribute to DIET_PLAN entity
5. Removed **Quantity** attribute from ORDERS relation
6. Removed Calendar_Date from DIET_PROGRESS and made "Day" attribute as a partial key.
7. Added **Offered_Ind** attribute in ITEM entity
8. Removed ORDERS binary relation and replaced with PLACES_ORDER ternary relation.
9. Created a new weak entity ORDERS to maintain order history

6.2. Specification of DB Objects

6.2.1. Tables

	Table Name	Column Name	Description of the Column
1	Customer: customer related basic details		
		cust_id	Unique identification number for customer
		cust_name	Name of the customer
		dob	Date of birth of the customer
		email_id	Email ID used by the customer for registration to the application
		password	Password used by the customer for registration to the application
		location	Location to which customer belongs to
2	Meal_plan: defined meal plans of the application		
		Plan_ID	Unique identification number for a unique combination of breakfast_count, lunch_count, dinner_count and dietician_option
		Breakfast_Count	Number of breakfast meals in a month
		Lunch_Count	Number of lunch meals in a month
		Dinner_Count	Number of dinner meals in a month
		Dietician_Option	"Yes or No options to signify dietician required in the plan and dietician not required in the plan"
		price	"total price based on breakfast_count, lunch_count, dinner_count and dietician_option"
3	customer_meal_plan: customers with selected with meal plan		
		cust_id	Unique identification number for customer
		start_date	Date on which selected meal plan to be started
		plan_id	Plan_ID selected by customer
4	Customer_Billing: payment details of customers for each enrollment		
		Cust_ID	Unique identification number for customer

		Start_Date	Date on which selected meal plan to be started
		Card_number	Card number of the card used for payment
		card_type	Category of the card type used for payment
		pay_date	Date on which payment is made for the mealplan
5	Restaurant: restaurant related basic details		
		restaurant_id	Unique identification number for restaurant
		name	Name of the restaurant
		email_id	Email ID used by the restaurant for registration to the application
		password	Password used by the restaurant for registration to the application
		location	Area in which restauarant is located
6	Item: food items names with nutritional value		
		Restaurant_ID	Unique identification number for restaurant
		Item_Name	Name of the food item
		Meal_Type	Category of meal in Breakfast/Lunch/Dinner
		Calories	Total calories in the food item
		Proteins	Total protein content in the food item
		Carbohydrates	Total carbohydrates content in the food item
		Fat	Total fat content in the food item
		offered_ind	Indicator with values as Y or N where Y implies Item_Name is offered currently by Restaurant_ID and N implies Item_Name is not offered currently by Restaurant_ID for customers
7	dietician: dietician related basic information		
		dietician_id	Unique identification number of dietician
		dietician_name	Name of the dietician
		email_id	Email ID used by the dietician for registration to the application
		password	Password used by the dietician for registration to the application

		experience	Number of years of professional experience that the dietician has
		qualification	Academic qualification of the dietician
8	restaurant_item_day: restaurant offered items on days		
		Restaurant_ID	Unique identification number of restaurant
		Item_Name	Name of the food item
		day_of_week	Day on which the Item_Name is offered by Restaurant_ID
9	diet_plan: customers with selected dietplans and basic lifetsyle details		
		cust_ID	Unique identification number of customer
		dietician_ID	Unique identification number of dietician
		diet_plan_id	Unique ID generated when each time a customer enrolls for a new plan_ID. The ID starts with 1 for new customer and increments there on
		diet_start_date	Date on which selected meal plan to be started
		new_diet_plan_ind	Indicator with values as Y or N where Y implies that diet_plan_id is still new and not addressed by the dietician_ID and N implies that diet_plan_ID is not new and has been addressed by the dietician_ID
		activity_level	Three levels of physical activity level indicators on an average daily basis which are low/medium/high
		weight	Measured weight in lbs of the customer
		sleep	"Three levels of sleep customer gets on an average daily basis which are low/medium/high
		height	Measured height in cms of the customer
10	diet_plan_details: details of each dietplan of customers		
		cust_ID	Unique identification number of customer
		dietician_ID	Unique identification number of dietician

		diet_plan_id	"Unique ID generated when each time a customer enrolls for a new plan_ID. The ID starts with 1 for new customer and increments there on"
		day	Day of the month. Starts from 1 and increments till 30
		meal_type	Category of meal in Breakfast/Lunch/Dinner
		calories	"Dietician suggested calories content for cust_ID with diet_plan_id for a day and for a meal_type"
		proteins	"Dietician suggested protein content for cust_ID with diet_plan_id for a day and for a meal_type"
		carbohydrates	"Dietician suggested carbohydrates content for cust_ID with diet_plan_id for a day and for a meal_type"
		fat	"Dietician suggested fat content for cust_ID with diet_plan_id for a day and for a meal_type"
11	allergy: food allergies if any associated with customers		
		cust_ID	Unique identification number of customer
		dietician_ID	Unique identification number of dietician
		diet_plan_id	Unique ID generated when each time a customer enrolls for a new plan_ID. The ID starts with 1 for new customer and increments there on"
		allergy	Type of allergy that cust_ID has when enrolled for diet_plan_ID with dietician_ID
12	disease: diseases if any associated with customers		
		cust_ID	Unique identification number of customer
		dietician_ID	Unique identification number of dietician
		diet_plan_id	Unique ID generated when each time a customer enrolls for a new plan_ID. The ID starts with 1 for new customer and increments there on

		disease	Type of disease that cust_ID has when enrolled for diet_plan_ID with dietician_ID
13	diet_progress: diet progess made by the customers of various dietplans		
		cust_ID	Unique identification number of customer
		dietician_ID	Unique identification number of dietician
		diet_plan_id	Unique ID generated when each time a customer enrolls for a new plan_ID. The ID starts with 1 for new customer and increments there on
		day	Day of the month. Starts from 1 and increments till 30
		breakfast	Indicator with values as Y or N which suggests whether or not cust_ID followed diet_plan_id on day for breakfast

6.2.2. Views

Customer

1. diet_ratings_view - This view is created to combine the average ratings received by each dietician from all the customers, with the other details of the dietician like name, email ID and qualification. If no ratings are provided to a particular dietician, then 'No Ratings' is displayed.
2. c_customer_remaining_meal_count: For each customer this view provides the number of Breakfast, Lunch and Dinner meals left in his current meal plan. This is calculated by taking the meal count from the meal plan and the Order details from Orders table. The information provided by this view is used as part of the c_menu_view.
3. c_menu_view: For each customer this view gives the list of items available from the restaurants in the same location as customer for the next day. For dietician enrolled customers, only those items which match the diet plan criteria are displayed. For other customers all the items available in the same location as the customer will be displayed
4. c_order_history_view: The purpose of this view is to give the complete history of orders for each customer. For each order the view returns customer id, meal type, Item name, Restaurant name, Order date and pick up time. This information is also displayed as part

of “rate an item” page, as a customer should be able to rate only those items he had ordered.

5. c_diet_plan_details: For all customers that are currently enrolled with a dietician, this view returns the current diet plan details. It provides the nutritional intake the customer is supposed to consume each day in his diet plan. If a customer is enrolled with a dietician in his past and currently has just a meal plan, there will be no records for that customer under this view.
6. c_diet_progress_details: For all customers that are currently enrolled with a dietician this view gives the current diet plan progress details. It will provide information if the customer has followed the suggested diet and if he had any additional food. For the view to provide this information, they must be submitted by customer first.
7. c_diet_plan_list_view: The purpose of this view is to return the list of past diet plans for each customer. It includes information like dietician name, diet plan id, diet start date etc.
8. c_diet_history_view: The purpose of this view is to return the complete information for the list of diet plans each customer has previously enrolled. It provides all the nutritional intake information the customer was supposed to take each day and also lists any additional calorie intake consumed by the customer. Along with this, the view also displays any diet progress updated by customer. If the customer did not update the progress for any completed day in the diet plan, this view doesn’t return any details for that day.
9. c_dietician_list_view: This view displays the list of dieticians each customer has enrolled, both in past and present. This information is displayed to the customer in the “rate a dietician” page, as a customer should be able to rate only those dieticians he had enrolled.

Dietician

1. d_customers_diet_plan_id_view – This view used to find all diet plan ids for a customer, along with customer details.

2. d_customer_diet_plan_details_view – This view gives the diet plan details for all 30 days for all 3 meal types – breakfast, lunch and dinner for all the diet plans ids of a customer.

3. d_customer_diet_plans_list_view – This view is used to find all diet plans associated with a customer along with start dates, pay dates, dietician of the diet plans.
4. d_customer_dietplan_progress_view – This view gives the diet plan details for all 30 days for all 3 meal types for all diet plans of a customer. It also shows the day-to-day diet progress details the customer has given.

5. d_diet_plans_cust_health_details_view – This view shows the customer health details like allergies and diseases associated with each diet plan of the customer.

6. diet_plan_day_meal_type_view – This view is used to get the days and meal types in a typical 30-day diet plan.

7. d_dietician_customers_view – This gives all the customers and customer details like diet start date, diet request date associated with a dietician.

8. d_update_diet_plan_dates_view – This gives the existing diet plan details for the diet days for which the diet plan update request is submitted.

9. d_update_diet_plans_list_view – This gives the list of diet plan update requests associated with a dietician.

Restaurant

1. r_restaurant_item_details_day_view – This gives the list of items and item details like calories, proteins, days of which the item is offered for a restaurant.

2. r_update_menu_view – This view gives the items list along with whether or not it is offered on a particular day of week for each restaurant.

6.2.3. Stored Procedures

Admin

1. a_check_admin_login_info: The procedure is called to validate the login credentials of the admin user. It takes as input email ID and password from the UI and performs a

database check. It gives as output the count matching the email ID and password, the email ID of the admin.

2. proc_d_ins_dietn_regn_details : The procedure is called to insert the details of the dietician from UI to database. This is called during the registration of the dietician by the admin. It takes as input name, email ID, password, experience in years and academic qualification. It gives as output as 1 if the insertion is successfully completed, If the email ID already exists, insertion will not happen and returns 0 as the output.

3. proc_r_ins_rest_details: The procedure is called to insert the details of the restaurant from UI to database. This is called during the registration of the restaurant by the admin. It takes as input name, email ID, password, and location. It gives as output as 1 if the insertion is successfully completed. If the email ID already exists, insertion will not happen and returns 0 as the output.

Customer

1. c_check_customer_login_info: This stored procedure is called to validate the login credentials of the customer. It takes as input email ID and password from the UI and performs a database check. It gives as output the count matching the email ID and password, customer type. Customer type will be one of the following depending on their respective scenarios:
    1. None: Not enrolled in any diet plan or meal plan
    2. Diet_Regular: Enrolled in either a diet plan or a meal plan with startdate is less than or equal to tomorrow
    3. Start_Date_Late: Enrolled in either a diet plan or meal plan but start date is greater than tomorrow
2. c_place_order: This stored procedure is used to place an order for the items selected by the customer. It takes as input item name, restaurant name and pick up time for each meal type along with the customer id. This procedure first checks if there is an existing record for the chosen meal type and if there is no record only then the order will be inserted. It returns as output three variables: bf_chk, ln_chk and din_chk. If there are no existing records with the specified details they will have a value of 0 indicating success else they will have a value of 1.
3. c_request_diet_plan_change: This stored procedure is used to insert the new change diet plan request made by customer. It takes as input customer id, initiation date,

request description. If there is an existing request from same customer for same diet plan, it will be overwritten and the output variable v_check value will be 1. If there are no existing records, v_check will be 0.

4. c_update_diet_progress: This stored procedure is used to insert the new diet progress information provided by customer. It takes as input customer id, log date, if customer has followed breakfast, lunch, dinner and if he had any additional intake. If there is an existing update from same customer for same date, it will be overwritten and the output variable v_cnt value will be 1 indicating the record is overwritten. If there are no existing records, v_cnt will be 0.

5. c_update_item_rating: This stored procedure is used to insert a new item rating provided by customer. It takes as input customer id, restaurant id, item name and rating. If there is an existing rating from same customer for same item, it will be overwritten and the output variable v_cnt value will be 1 indicating the record is overwritten. If there are no existing records, v_cnt will be 0.

6. c_update_dietician_rating: This stored procedure is used to insert a new dietician rating provided by customer. It takes as input customer id, dietician id and rating. If there is an existing rating from same customer for same dietician, it will be overwritten and the output variable v_cnt value will be 1 indicating the record is overwritten. If there are no existing records, v_cnt will be 0.

7. proc_c_ins_cust_details: The procedure is called to insert the details of the customer from UI to database. This is called during the registration of the customer. It takes as input name, email ID, password, date of birth and location. It gives as output as 1 if the insertion is successfully completed. If the email ID already exists, insertion will not happen and returns 0 as the output.

8. proc_dietician_yes: The procedure is called to insert the details of customer who wishes to enroll with a diet plan. It takes as input the customer enters and inserts record into five tables based on the respective fields. It checks for the previous diet plans the customer has had and increments the diet plan ID accordingly. If the insertions happen successfully then 1 is returned else 0 is returned.

9. proc_dietician_no: The procedure is called to insert the details of customer who does not enroll with a diet plan. Because this customer will have no additional information related to lifestyle this involves inserting data only to two tables. If the insertions happen successfully then 1 is returned else 0 is returned.

Dietician

1. d_check_dietician_login_info: The procedure is called to validate the login credentials of the dietician user. It takes as input email ID and password from the UI and performs a database check. It gives as output the count matching the email ID and password, the email ID of the dietician.
2. d_add_diet_details: The procedure is used to add and update diet plan details like calories, proteins for all meal types (Breakfast, Lunch and Dinner). If the diet plan details are present for a particular day, this procedure updates the details, otherwise inserts the diet plan. It gives as output as 1 if the operation is successfully completed, -1 if the insertion is not successful.
3. d_finalize_new_diet_plan: The procedure is used to finalize the new diet plan of a customer. If the dietician is happy with the diet plan he has prepared for a customer and does not want to make any further changes, then he can use the "Finalize Diet Plan" option from UI, and this procedure is called. It updates the new_diet_plan_ind column in diet_plan table to 'N'. It gives as output as 1 if the operation is successfully completed, -1 if the insertion is not successful.
4. d_finalize_update_diet_plan: The procedure is used to finalize the uddated diet plan of a customer. If the dietician is happy with the diet plan he has prepared for a customer and does not want to make any further changes, then he can use the "Finalize Diet Plan" option from UI, and this procedure is called. It updates the update_diet_plan_ind column in diet_customer_requests table to 'N'. It gives as output as 1 if the operation is successfully completed, -1 if the insertion is not successful.

Restaurant

1. r_check_restaurant_login_info: The procedure is called to validate the login credentials of the restaurant user. It takes as input email ID and password from the UI and performs a database check. It gives as output the count matching the email ID and password, the email ID of the restaurant.
2. r_add_menu_item: The procedure is called whenever a restauranr adds a new item to its menu. It takes the item name, nutritional information like calories,carbohydrates, meal type and the days of week on which the item is offered. The procedure checks if the item name is already present for that particular restauran and if it's a new item, then the new item is inserted in item table, and days of week on which the item is offered will be updated in restaurant_item_day table. It returns

output as 1 in case of successful previously mentioned operation. If the item is already present for the restaurant, then it returns 2 as output and does not insert the item. In case of DB errors, 0 is returned.

3. proc_r_generate_report: The procedure is called when the restaurant requests for a sales report for a specific time period. It takes the input as dates to which the report is required and returns the items ordered during that time period, the number of times each item is ordered, and the rating received for each item. This will perform only select operations from combination of three tables.

4. proc_update_menu: The procedure is called when the restaurant chooses to update the menu. Item to be updated along with days of week variables are taken as the input. Based on the updated menu, for each item, the item name and the day of the week on which it is offered is updated in the respective table. Similarly, if an item is removed being offered on a particular day, the respective entry is deleted from the table. It performs the check and corresponding insertion or deletion for each day of the week.

6.3. Functional Dependencies and Normalization

1. Customer

Table: Customer (Cust_ID,Cust_Name,DOB,Email_ID,Password,Location)

FDs:
Cust_ID → Cust_Name
Cust_ID → DOB
Cust_ID → Email_ID
Cust_ID → Password
Cust_ID → Location

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID
  - All the non-primary keys depend on primary key Cust_ID
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID → Cust_Name	All non-key attributes are fully functionally dependent on primary key
Cust_ID → DOB	All non-key attributes are fully functionally dependent on primary key
Cust_ID → Email_ID	All non-key attributes are fully functionally dependent on primary key
Cust_ID → Password	All non-key attributes are fully functionally dependent on primary key
Cust_ID → Location	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID → Cust_Name	All non-key attributes are directly dependent on primary key
Cust_ID → DOB	All non-key attributes are directly dependent on primary key
Cust_ID → Email_ID	All non-key attributes are directly dependent on primary key
Cust_ID → Password	All non-key attributes are directly dependent on primary key
Cust_ID → Location	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID → Cust_Name	Every determinant in the FD is a candidate key
Cust_ID → DOB	Every determinant in the FD is a candidate key
Cust_ID → Email_ID	Every determinant in the FD is a candidate key

Cust_ID → Password	Every determinant in the FD is a candidate key
Cust_ID → Location	Every determinant in the FD is a candidate key

2. ***Customer_Billing***

Table:
Customer_Billing(Cust_ID,Plan_ID,Start_Date,Card_Number,Card_Type,Pay_Date)

FDs:
Cust_ID, Start_Date → Card_Number
Cust_ID, Start_Date → Card_Type
Cust_ID, Start_Date → Pay_Date

Normalization:

- **1NF:** Yes
- **2NF**: This table is not in 2NF because some of the functional dependencies have partial dependency. Below are the details.

Functional Dependency	***Partial Dependency***
Cust_ID, Start_Date → Card_Num	Non-key attribute Card_Num is not fully functionally dependent on primary key. It depends only on part of the primary key Cust_ID, Start_Date
Cust_ID, Start_Date → Card_Type	Non-key attribute Card_Type is not fully functionally dependent on primary key. It depends only on part of the primary key Cust_ID, Start_Date
Cust_ID, Start_Date → Pay_Date	Non-key attribute Pay_Date is not fully functionally dependent on primary key. It depends only on part of the primary key Cust_ID, Start_Date

To normalize the table to 2NF, we have identified the problematic attributes: Card_Num, Card_Type, Pay_Date. These attributes along with the part of original primary key on which they are fully functionally dependent is formed as a new table: Customer_Billing.
The original primary key is preserved as part of Customer_Meal_Plan table.

Below are the new tables generated after doing 2NF:

2a. Customer_Billing (Cust_ID, Start_Date, Card_Number,Card_Type,Pay_Date)
2b. Customer_Meal_Plan (Cust_ID,Plan_ID,Start_Date)

- **3NF**: **Customer_Billing**: Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Start_Date → Card_Num	All non-key attributes are directly dependent on primary key
Cust_ID, Start_Date → Card_Type	All non-key attributes are directly dependent on primary key
Cust_ID, Start_Date → Pay_Date	All non-key attributes are directly dependent on primary key

Customer_Meal_Plan: As all the columns are part of primary key, there are no dependencies in this table.

- **BCNF**:

Customer_Meal_Plan : As all the columns are part of primary key, there are no dependencies in this table.
Customer_Billing : Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Start_Date → Card_Num	Every determinant in the FD is a candidate key
Cust_ID, Start_Date → Card_Type	Every determinant in the FD is a candidate key
Cust_ID, Start_Date → Pay_Date	Every determinant in the FD is a candidate key

3. Meal_Plan

Table: Meal_Plan(Plan_ID,Breakfast_Count,Lunch_Count,Dinner_Count,dietician_Option,Price)

FDs:

Plan ID → Breakfast_Count
Plan ID → Lunch_Count
Plan ID → Dinner_Count
Plan ID → dietician_Option
Plan ID → Price

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Plan_ID
  - All the non-primary keys depend on primary key Plan_ID
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Plan ID → Breakfast_Count	All non-key attributes are fully functionally dependent on primary key
Plan ID → Lunch_Count	All non-key attributes are fully functionally dependent on primary key
Plan ID → Dinner_Count	All non-key attributes are fully functionally dependent on primary key
Plan ID → dietician_Option	All non-key attributes are fully functionally dependent on primary key
Plan ID → Price	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Plan ID → Breakfast_Count	All non-key attributes are directly dependent on primary key
Plan ID → Lunch_Count	All non-key attributes are directly dependent on primary key
Plan ID → Dinner_Count	All non-key attributes are directly dependent on primary key
Plan ID → dietician_Option	All non-key attributes are directly dependent on primary key
Plan ID → Price	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Plan ID → Breakfast_Count	Every determinant in the FD is a candidate key
Plan ID → Lunch_Count	Every determinant in the FD is a candidate key
Plan ID → Dinner_Count	Every determinant in the FD is a candidate key
Plan ID → dietician_Option	Every determinant in the FD is a candidate key
Plan ID → Price	Every determinant in the FD is a candidate key

4. Restaurant

Table: Restaurant(Restaurant_ID,Name,Email_ID,Password,Location)

FDs:
Restaurant_ID→ Name
Restaurant_ID→ Email_ID
Restaurant_ID→ Password

Restaurant_ID→ Location

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Restaurant_ID
  - All the non-primary keys depend on primary key Restaurant_ID
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Restaurant_ID→ Name	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID→ Email_ID	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID→ Password	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID→ Location	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Restaurant_ID→ Name	All non-key attributes are directly dependent on primary key
Restaurant_ID→ Email_ID	All non-key attributes are directly dependent on primary key
Restaurant_ID→ Password	All non-key attributes are directly dependent on primary key
Restaurant_ID→ Location	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Restaurant_ID→ Name	Every determinant in the FD is a candidate key
Restaurant_ID→ Email_ID	Every determinant in the FD is a candidate key
Restaurant_ID→ Password	Every determinant in the FD is a candidate key
Restaurant_ID→ Location	Every determinant in the FD is a candidate key

5. Dietician

Table:
Dietician(Dietician_ID,Dietician_Name,Email_ID,Password,Experience,Qualification)

FDs:
Dietician_ID → Dietician_Name
Dietician_ID → Email_ID
Dietician_ID → Password
Dietician_ID → Experience
Dietician_ID → Qualification

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Dietician_ID
  - All the non-primary keys depend on primary key Dietician_ID
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Dietician_ID → Dietician_Name	All non-key attributes are fully functionally dependent on primary key
Dietician_ID → Email_ID	All non-key attributes are fully functionally dependent on primary key
Dietician_ID → Password	All non-key attributes are fully functionally dependent on primary key
Dietician_ID → Experience	All non-key attributes are fully functionally dependent on primary key
Dietician_ID → Qualification	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Dietician_ID → Dietician_Name	All non-key attributes are directly dependent on primary key
Dietician_ID → Email_ID	All non-key attributes are directly dependent on primary key
Dietician_ID → Password	All non-key attributes are directly dependent on primary key
Dietician_ID → Experience	All non-key attributes are directly dependent on primary key
Dietician_ID → Qualification	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Dietician_ID → Dietician_Name	Every determinant in the FD is a candidate key
Dietician_ID → Email_ID	Every determinant in the FD is a candidate key
Dietician_ID → Password	Every determinant in the FD is a candidate key

Dietician_ID → Experience	Every determinant in the FD is a candidate key
Dietician_ID → Qualification	Every determinant in the FD is a candidate key

6. Item

Table: Item(Restaurant_ID,Item_Name,Meal_type,Calories,Carbohydrates,Protein,Fat, Offered_Ind)

FDs:
Restaurant_ID, Item_Name → Meal_type
Restaurant_ID, Item_Name → Calories
Restaurant_ID, Item_Name → Carbohydrates
Restaurant_ID, Item_Name → Protein
Restaurant_ID, Item_Name → Fat
Restaurant_ID, Item_Name → Offered_Ind

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Restaurant_ID,Item_Name.
  - All the non-primary keys depend on primary key Restaurant_ID,Item_Name
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Restaurant_ID, Item_Name → Meal_type	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID, Item_Name → Calories	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID, Item_Name → Carbohydrates	All non-key attributes are fully functionally dependent on primary key

Restaurant_ID, Item_Name → Protein	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID, Item_Name → Fat	All non-key attributes are fully functionally dependent on primary key
Restaurant_ID, Item_Name → Offered_Ind	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Restaurant_ID, Item_Name → Meal_type	All non-key attributes are directly dependent on primary key
Restaurant_ID, Item_Name → Calories	All non-key attributes are directly dependent on primary key
Restaurant_ID, Item_Name → Carbohydrates	All non-key attributes are directly dependent on primary key
Restaurant_ID, Item_Name → Protein	All non-key attributes are directly dependent on primary key
Restaurant_ID, Item_Name → Fat	All non-key attributes are directly dependent on primary key
Restaurant_ID, Item_Name → Offered_Ind	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Restaurant_ID, Item_Name → Meal_type	Every determinant in the FD is a candidate key
Restaurant_ID, Item_Name → Calories	Every determinant in the FD is a candidate key
Restaurant_ID, Item_Name → Carbohydrates	Every determinant in the FD is a candidate key

Restaurant_ID, Item_Name → Protein	Every determinant in the FD is a candidate key
Restaurant_ID, Item_Name → Fat	Every determinant in the FD is a candidate key
Restaurant_ID, Item_Name → Offered_Ind	Every determinant in the FD is a candidate key

7. ***(Day_Of_Week) Restaurant_Item_Day***

Table: Day_Of_Week(Restaurant_ID,Item_Name,Day_Of_Week)

FDs: As all the columns are part of primary key, there are no dependencies in this table.

Normalization:
1NF: Yes
2NF: Yes
3NF: Yes
BCNF: Yes

8. ***Diet_Plan***

Table: Diet_Plan(Cust_ID, Dietician_Id, Diet_Plan_Id,New_Diet_Plan_Ind,Activity_level, Weight,Height,Sleep,Measurement_Date)

FDs:
Cust_ID, Dietician_Id, Diet_Plan_Id → New_Diet_Plan_Ind
Cust_ID, Dietician_Id, Diet_Plan_Id → Activity_level
Cust_ID, Dietician_Id, Diet_Plan_Id → Weight
Cust_ID, Dietician_Id, Diet_Plan_Id → Height
Cust_ID, Dietician_Id, Diet_Plan_Id → Sleep
Cust_ID, Dietician_Id, Diet_Plan_Id → Measurement_Date

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID, Dietician_Id, Diet_Plan_Id
  - All the non-primary keys depend on primary key Cust_ID, Dietician_Id, Diet_Plan_Id
  - All the values are atomic.

- There is no multivalued attribute
- There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Dietician_Id, Diet_Plan_Id → New_Diet_Plan_Ind	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Activity_level	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Weight	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Height	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Sleep	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Measurement_Date	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_Id, Diet_Plan_Id → New_Diet_Plan_Ind	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Activity_level	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Weight	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Height	All non-key attributes are directly dependent on primary key

Cust_ID, Dietician_Id, Diet_Plan_Id → Sleep	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_Id, Diet_Plan_Id → Measurement_Date	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_Id, Diet_Plan_Id → New_Diet_Plan_Ind	Every determinant in the FD is a candidate key
Cust_ID, Dietician_Id, Diet_Plan_Id → Activity_level	Every determinant in the FD is a candidate key
Cust_ID, Dietician_Id, Diet_Plan_Id → Weight	Every determinant in the FD is a candidate key
Cust_ID, Dietician_Id, Diet_Plan_Id → Height	Every determinant in the FD is a candidate key
Cust_ID, Dietician_Id, Diet_Plan_Id → Sleep	Every determinant in the FD is a candidate key
Cust_ID, Dietician_Id, Diet_Plan_Id → Measurement_Date	Every determinant in the FD is a candidate key

9. NMD (Diet_Plan_Details)

Table:
Diet_Plan_Details(Cust_ID,Dietician_Id,Diet_Plan_Id,Day,Meal_Type,Calories,Carbohydrates,Protein,Fat)

FDs:
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Calories
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Carbohydrates
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Protein
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Fat

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID,Dietician_Id,Diet_Plan_Id,Day,Meal_Type
  - All the non-primary keys depend on primary key Cust_ID,Dietician_Id,Diet_Plan_Id,Day,Meal_Type
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Calories	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Carbohydrates	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Protein	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Fat	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Calories	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Carbohydrates	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Protein	All non-key attributes are directly dependent on primary key

Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Fat	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Calories	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Carbohydrates	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Protein	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day, Meal_Type → Fat	Every determinant in the FD is a candidate key

10. Allergies

Table: Allergies(Cust_ID,Dietician_Id,Diet_Plan_Id,Allergy)

FDs: As all the columns are part of primary key, there are no dependencies in this table.

Normalization:
1NF: Yes
2NF: Yes
3NF: Yes
BCNF: Yes

11. Disease

Table:Disease(Cust_ID,Dietician_Id,Diet_Plan_Id,Disease)

FDs: As all the columns are part of primary key, there are no dependencies in this table.

Normalization:

1NF: Yes

2NF: Yes

3NF: Yes

BCNF: Yes

12. Diet Progress

Table:

Diet_Progress(Cust_ID,Dietician_Id,Diet_Plan_Id,Day,Breakfast,Lunch,Dinner,Additional_Intake)

FDs:

Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Breakfast

Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Lunch

Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Dinner

Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Additional_Intake

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID,Dietician_Id,Diet_Plan_Id,Day
  - All the non-primary keys depend on primary key Cust_ID,Dietician_Id,Diet_Plan_Id,Day
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Breakfast	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Lunch	All non-key attributes are fully functionally dependent on primary key

Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Dinner	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Additional_Intake	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Breakfast	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Lunch	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Dinner	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Additional_Intake	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Breakfast	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Lunch	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Dinner	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID, Day → Additional_Intake	Every determinant in the FD is a candidate key

13. Customer_Requests

Table:
Customer_Requests(Cust_ID,Dietician_Id,Diet_Plan_Id,Request_Initiation_Date,Request_Content,Update_Diet_Plan_Ind)

FDs:
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Initiation_Date
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Content
Cust_ID, Dietician_ID, Diet_Plan_ID → Update_Diet_Plan_Ind

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID,Dietician_Id,Diet_Plan_Id
  - All the non-primary keys depend on primary key Cust_ID,Dietician_Id,Diet_Plan_Id
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Initiation_Date	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Content	All non-key attributes are fully functionally dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID → Update_Diet_Plan_Ind	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Initiation_Date	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Content	All non-key attributes are directly dependent on primary key
Cust_ID, Dietician_ID, Diet_Plan_ID → Update_Diet_Plan_Ind	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Initiation_Date	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID → Request_Content	Every determinant in the FD is a candidate key
Cust_ID, Dietician_ID, Diet_Plan_ID → Update_Diet_Plan_Ind	Every determinant in the FD is a candidate key

14. Orders

Table: Orders(Cust_ID,Restaurant_ID,Item_Name,Order_Date,Pickup_Time)

FDs:
Cust_ID, Restaurant_ID, Item_Name, Order_Date → Pickup_Time

Normalization:

- **1NF:** Yes, this relation is in 1NF because:

- It has a primary key Cust_ID,Restaurant_ID,Item_Name,Order_Date
- All the non-primary keys depend on primary key Cust_ID,Restaurant_ID,Item_Name,Order_Date
- All the values are atomic.
- There is no multivalued attribute
- There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Restaurant_ID, Item_Name, Order_Date → Pickup_Time	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Restaurant_ID, Item_Name, Order_Date → Pickup_Time	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Restaurant_ID, Item_Name, Order_Date → Pickup_Time	Every determinant in the FD is a candidate key

15. Item_Ratings

Table: Item_Ratings(Cust_ID,Restaurant_ID,Item_Name,Rating)

FDs:
Cust_ID, Restaurant_ID, Item_Name → Rating

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID,Restaurant_ID,Item_Name
  - All the non-primary keys depend on primary key Cust_ID,Restaurant_ID,Item_Name.
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Restaurant_ID, Item_Name → Rating	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Restaurant_ID, Item_Name → Rating	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Restaurant_ID, Item_Name → Rating	Every determinant in the FD is a candidate key

16. Rates_Dietician

Table: Dietician_Ratings(<u>Cust_ID,Dietician_ID</u>,Rating)

FDs:
Cust_ID, Dietician_ID → Rating

Normalization:

- **1NF:** Yes, this relation is in 1NF because:
  - It has a primary key Cust_ID,Dietician_ID
  - All the non-primary keys depend on primary key Cust_ID,Dietician_ID
  - All the values are atomic.
  - There is no multivalued attribute
  - There is no nested relation

- **2NF:** Yes, this table is in 2NF because it is in 1NF and has no partial dependencies. Below are the details for each functional dependency.

Functional Dependency	***Partial Dependency***
Cust_ID, Dietician_ID → Rating	All non-key attributes are fully functionally dependent on primary key

- **3NF:** Yes, this table is in 3NF because it is in 2NF and has no transitive dependencies. Below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID → Rating	All non-key attributes are directly dependent on primary key

- **BCNF**: Yes, this table is in BCNF because it is in 3NF and below are the details for each functional dependency.

Functional Dependency	***Transitive Dependency***
Cust_ID, Dietician_ID → Rating	Every determinant in the FD is a candidate key

6.4. Multi statement DB Transactions (Initiated from DB Server Side)

6.4.1. Admin

1. proc_d_ins_dietn_regn_details

```
-- PROCEDURE TO INSERT DIETICIAN REGISTRATION DETAILS

drop procedure if exists proc_d_ins_dietn_regn_details;
DELIMITER $$
create procedure proc_d_ins_dietn_regn_details
(in dietn_name varchar(50),
in dietn_email varchar(50),
in dietn_pwd varchar(50),
in dietn_exp decimal(5,2),
in dietn_qua varchar(50),
out vc int,
out eid varchar(50))
begin
   DECLARE `_rollback` BOOL DEFAULT 0;
   DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;

   START TRANSACTION;
   set eid = NULL;
   set vc = 0;
   select email_id into eid from dietician where email_id = dietn_email;
   if (eid is NULL) then
      insert into dietician (dietician_name, email_id, password, experience, qualification) values (dietn_name, dietn_email, SHA1(dietn_pwd), dietn_exp, dietn_qua);
      set vc = 1;
   end if;

   IF `_rollback` THEN ROLLBACK;
   ELSE COMMIT;
   END IF;
end
$$
```

2. proc_r_ins_rest_details

```
-- PROCEDURE TO INSERT RESTAURANT REGISTRATION DETAILS

drop procedure if exists proc_r_ins_rest_details;
DELIMITER $$
create procedure proc_r_ins_rest_details
(in rest_name varchar(50),
in rest_email varchar(50),
in rest_pwd varchar(50),
in rest_location varchar(30),
out vc int,
out eid varchar(50))
begin
  DECLARE `_rollback` BOOL DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
  START TRANSACTION;

  set eid = NULL;
  set vc = 0;
  select email_id into eid from restaurant where email_id = rest_email;
  if (eid is NULL) then
    insert into restaurant (name, email_id, password, location) values (rest_name, rest_email, SHA1(rest_pwd), rest_location);
    set vc = 1;
  end if;

  IF `_rollback` THEN ROLLBACK;
  ELSE COMMIT;
  END IF;
end
$$
DELIMITER ;
```

3. proc_c_ins_cust_details

```
-- PROCEDURE TO INSERT CUSTOMER REGISTRATION DETAILS
drop procedure if exists proc_c_ins_cust_details;
DELIMITER $$
create procedure proc_c_ins_cust_details
(in cust_name varchar(50),
in cust_dob date,
in cust_email_id varchar(50),
in cust_password varchar(50),
in cust_location varchar(50),
out vc int,
out eid varchar(50))
begin
  DECLARE `_rollback` BOOL DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
  START TRANSACTION;

  set eid = NULL;
  set vc = 0;
  select email_id into eid from customer where email_id = cust_email_id;
  if (eid is null) then
    insert into customer (cust_name, dob, email_id, password, location) values (cust_name, cust_dob, cust_email_id, SHA1(cust_password),
    cust_location);
    set vc = 1;
  end if;

  IF `_rollback` THEN ROLLBACK;
  ELSE COMMIT;
  END IF;
end
$$
DELIMITER ;
```

6.4.2. Customer

1. proc_dietician_yes

```
33
34 •  -- Dietician_YES
35
36    drop procedure if exists proc_dietician_yes;
37    DELIMITER $$
38 •  create procedure proc_dietician_yes(
39    in v_customer_id int,
40    in stdt date,
41    in cardtype varchar (10),
42    in cardnumb bigint(20),
43    in dieticianid int(11),
44    in measr_date date,
45    in activitylevel varchar(10),
46    in weight decimal(6,2),
47    in height decimal(5,2),
48    in sleep int(11),
49    in disease1 varchar(25),
50    in disease2 varchar(25),
51    in disease3 varchar(25),
52    in disease4 varchar(25),
53    in disease5 varchar(25),
54    in allergy1 varchar(30),
55    in allergy2 varchar(30),
56    in allergy3 varchar(30),
57    in allergy4 varchar(30),
58    in allergy5 varchar(30),
59    out res int,
60    out cnt int)
61    begin
62       DECLARE `_rollback` BOOL DEFAULT 0;
63       DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
64       START TRANSACTION;
65
66       set res = 0;
```

```
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
START TRANSACTION;

set res = 0;
set cnt= 0;

insert into customer_meal_plan(cust_id, start_date, plan_id)
values (v_customer_id, stdt, '1');

insert into customer_billing (cust_ID, start_date, card_number, card_type, pay_date)
values(v_customer_id, stdt, cardnumb, cardtype, curdate());

select count(cust_ID) into cnt from diet_plan
where cust_ID = v_customer_id;
set cnt = cnt + 1;

insert into diet_plan
values(v_customer_id, dieticianid, cnt, stdt, 'Y', activitylevel, weight, sleep, height);

if (disease1 != 'undefined') then
   insert into disease values(v_customer_id, dieticianid, cnt, disease1);
end if;
if (disease2 != 'undefined') then
   insert into disease values(v_customer_id, dieticianid, cnt, disease2);
end if;
if (disease3 != 'undefined') then
   insert into disease values(v_customer_id, dieticianid, cnt, disease3);
end if;
if (disease4 != 'undefined') then
   insert into disease values(v_customer_id, dieticianid, cnt, disease4);
end if;
if (disease5 != 'undefined') then
   insert into disease values(v_customer_id, dieticianid, cnt, disease5);
end if;
 if (allergy1 != 'undefined') then
   insert into allergy values(v_customer_id, dieticianid, cnt, allergy1);
end if;
if (allergy2 != 'undefined') then
```

```
    end if;
    if (allergy2 != 'undefined') then
        insert into allergy values(v_customer_id, dieticianid, cnt, allergy2);
    end if;
    if (allergy3 != 'undefined') then
        insert into allergy values(v_customer_id, dieticianid, cnt, allergy3);
    end if;
    if (allergy4 != 'undefined') then
        insert into allergy values(v_customer_id, dieticianid, cnt, allergy4);
    end if;
    if (allergy5 != 'undefined') then
        insert into allergy values(v_customer_id, dieticianid, cnt, allergy5);
    end if;
    set res =1;

    IF `_rollback` THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
  end
  $$
  DELIMITER ;
```

2. proc_dietician_no

```
22
23 •  -- Dietician_NO
24    drop procedure if exists proc_dietician_no;
25    DELIMITER $$
26 •  create procedure proc_dietician_no(
27    in v_customer_id int,
28    in stdt DATE,
29    in mealtypeid int(11),
30    in cardtype varchar(10),
31    in cardnumb bigint(20),
32    out res int
33    )
34    begin
35       DECLARE `_rollback` BOOL DEFAULT 0;
36       DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
37       START TRANSACTION;
38
39       set res = 0;
40       insert into customer_meal_plan
41       values (v_customer_id, stdt, mealtypeid);
42       insert into customer_billing
43       values(v_customer_id, stdt,cardnumb,cardtype, curdate());
44       set res = 1;
45
46       IF `_rollback` THEN ROLLBACK;
47       ELSE COMMIT;
48       END IF;
49    end
50    $$
51    DELIMITER ;
52
```

3. c_place_order


```
DELIMITER $$
CREATE PROCEDURE c_place_order
(IN p_cust_id int, IN restaurant_name_bf varchar(30),  IN item_name_bf varchar(30), IN restaurant_name_ln varchar(30),
IN item_name_ln varchar(30), IN restaurant_name_din varchar(30),  IN item_name_din varchar(30),
IN p_bf_time time,IN p_ln_time time, IN p_din_time time, OUT bf_chk int,OUT ln_chk int, OUT din_chk int )
BEGIN
DECLARE restaurant_id_bf int DEFAULT 0;
DECLARE restaurant_id_ln int DEFAULT 0;
DECLARE restaurant_id_din int DEFAULT 0;
DECLARE `_rollback` BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
START TRANSACTION;

SELECT COUNT(o.cust_id) into bf_chk
FROM orders as o
JOIN item as i
ON i.restaurant_id = o.restaurant_id AND i.item_name = o.item_name
WHERE cust_id = p_cust_id AND order_date = DATE(NOW()) AND meal_type = 'breakfast' AND  restaurant_name_bf!= ' ';
SELECT COUNT(o.cust_id) into ln_chk
FROM orders as o
JOIN item as i
ON i.restaurant_id = o.restaurant_id AND i.item_name = o.item_name
WHERE cust_id = p_cust_id AND order_date = DATE(NOW()) AND meal_type = 'lunch' AND restaurant_name_ln!= ' ' ;
SELECT COUNT(o.cust_id) into din_chk
FROM orders as o
JOIN item as i
ON i.restaurant_id = o.restaurant_id AND i.item_name = o.item_name
WHERE cust_id = p_cust_id AND order_date = DATE(NOW()) AND meal_type = 'dinner' AND restaurant_name_din!= ' ';
SELECT restaurant_id into restaurant_id_bf
FROM restaurant
WHERE name = restaurant_name_bf;
SELECT restaurant_id into restaurant_id_ln
FROM restaurant
WHERE name = restaurant_name_ln;
SELECT restaurant_id into restaurant_id_din
FROM restaurant
WHERE name = restaurant_name_din;

IF restaurant_id_bf != 0  AND bf_chk = 0 THEN
 INSERT INTO orders values(p_cust_id,restaurant_id_bf,item_name_bf,NOW(),p_bf_time);
END IF;
IF restaurant_id_ln != 0 AND ln_chk = 0 THEN
 INSERT INTO orders values(p_cust_id,restaurant_id_ln,item_name_ln,NOW(),p_ln_time);
END IF;
IF restaurant_id_din != 0 AND din_chk = 0 THEN
 INSERT INTO orders values(p_cust_id,restaurant_id_din,item_name_din,NOW(),p_din_time);
END IF;
IF `_rollback` THEN ROLLBACK;
ELSE COMMIT;
END IF;
END;
$$
DELIMITER ;
```

4. c_request_diet_plan_change

```
297   DELIMITER $$
298 • CREATE PROCEDURE c_request_diet_plan_change(IN p_cust_id int, IN input_date date, IN p_request varchar(50),OUT v_check int)
299   BEGIN
300   DECLARE v_cust_id int;
301   DECLARE v_dietician_id int;
302   DECLARE v_diet_plan_id int;
303   DECLARE `_rollback` BOOL DEFAULT 0;
304   DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
305   START TRANSACTION;
306   SELECT DISTINCT cust_id, dietician_id, diet_plan_id into v_cust_id,v_dietician_id,v_diet_plan_id
307   FROM c_diet_plan_details
308   WHERE cust_id = p_cust_id;
309   -- To check if there is an existing request change
310   SELECT COUNT(*) into v_check
311   FROM customer_requests
312   WHERE cust_id= v_cust_id AND dietician_id = v_dietician_id AND diet_plan_id = v_diet_plan_id;
313   If v_check != 1 THEN
314       INSERT INTO customer_requests values(v_cust_id,v_dietician_id,v_diet_plan_id,input_date,p_request,'Y');
315   ELSE
316       UPDATE customer_requests
317       SET request_initiation_date = input_date,request_content = p_request,update_diet_plan_ind='Y'
318       WHERE cust_id= v_cust_id AND dietician_id = v_dietician_id AND diet_plan_id = v_diet_plan_id;
319   END IF;
320   IF `_rollback` THEN ROLLBACK;
321   ELSE COMMIT;
322   END IF;
323   END;
324   $$
325   DELIMITER ;
326
```

5. c_update_diet_progress

```
331
332    DELIMITER $$
333 •  CREATE PROCEDURE c_update_diet_progress (IN p_cust_id int,IN log_date date, IN breakfast varchar(10), IN lunch varchar(10),
334    IN dinner varchar(10),IN additional_intake varchar(50),OUT v_cnt int)
335    BEGIN
336    DECLARE v_day int;
337    DECLARE v_cust_id int;
338    DECLARE v_dietician_id int;
339    DECLARE v_diet_plan_id int;
340    DECLARE v_start_date date;
341    DECLARE `_rollback` BOOL DEFAULT 0;
342    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
343    START TRANSACTION;
344    SELECT DISTINCT cust_id, dietician_id, diet_plan_id,diet_start_date into v_cust_id,v_dietician_id,v_diet_plan_id,v_start_date
345    FROM c_diet_plan_details
346    WHERE cust_id = p_cust_id;
347    SELECT DATEDIFF(log_date,v_start_date)+1 into v_day;
348    -- To check if the progress is already logged for tht day
349    SELECT COUNT(*) into v_cnt
350    FROM diet_progress
351    WHERE cust_id= v_cust_id AND dietician_id = v_dietician_id AND diet_plan_id = v_diet_plan_id AND day = v_day;
352    If v_cnt != 1 THEN
353        INSERT INTO diet_progress values(v_cust_id,v_dietician_id,v_diet_plan_id,v_day,breakfast,lunch,dinner,additional_intake);
354    ELSE
355        UPDATE diet_progress
356        SET breakfast = breakfast,lunch = lunch,dinner=dinner,additional_intake=additional_intake
357        WHERE cust_id= v_cust_id AND dietician_id = v_dietician_id AND diet_plan_id = v_diet_plan_id AND day = v_day;
358    END IF;
359    IF `_rollback` THEN ROLLBACK;
360    ELSE COMMIT;
361    END IF;
362    END;
363    $$
364    DELIMITER ;
```

6. c_update_item_rating

```
375
376     DELIMITER $$
377 •   CREATE PROCEDURE c_update_item_rating
378     (IN p_cust_id int,IN p_restaurant_id int, IN p_item_name varchar(30), IN p_rating int, OUT v_chck int )
379     BEGIN
380     DECLARE `_rollback` BOOL DEFAULT 0;
381     DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
382     START TRANSACTION;
383     SELECT COUNT(*) into v_chck
384     FROM item_ratings
385     WHERE cust_id = p_cust_id AND restaurant_id = p_restaurant_id AND item_name= p_item_name;
386     If v_chck != 1 THEN
387         INSERT INTO item_ratings values(p_cust_id,p_restaurant_id,p_item_name,p_rating);
388     ELSE
389         UPDATE item_ratings
390         SET rating = p_rating
391         WHERE cust_id = p_cust_id AND restaurant_id = p_restaurant_id AND item_name= p_item_name;
392     END IF;
393     IF `_rollback` THEN ROLLBACK;
394     ELSE COMMIT;
395     END IF;
396     END;
397     $$
398     DELIMITER ;
```

7. c_update_dietician_rating

```
404
405    DELIMITER $$
406 •  CREATE PROCEDURE c_update_dietician_rating (IN p_cust_id int,IN p_dietician_id int, IN p_rating int, OUT v_chck int)
407    BEGIN
408    DECLARE `_rollback` BOOL DEFAULT 0;
409    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
410    START TRANSACTION;
411    SELECT COUNT(*) into v_chck
412    FROM dietician_ratings
413    WHERE cust_id = p_cust_id AND dietician_id = p_dietician_id;
414    If v_chck != 1 THEN
415        INSERT INTO dietician_ratings values(p_cust_id,p_dietician_id,p_rating);
416    ELSE
417    UPDATE dietician_ratings
418    SET rating = p_rating
419    WHERE cust_id = p_cust_id AND dietician_id = p_dietician_id;
420    END IF;
421    IF `_rollback` THEN ROLLBACK;
422    ELSE COMMIT;
423    END IF;
424    END;
425    $$
426    DELIMITER ;
```

6.4.3. Dietician

1. d_add_diet_details

```
DELIMITER $$
CREATE PROCEDURE d_add_diet_details(
    IN      v_custid_dietstartdate  VARCHAR(50),
    IN      v_day                   VARCHAR(10),
    IN      v_breakfast_calories    VARCHAR(10),
    IN      v_breakfast_proteins    VARCHAR(10),
    IN      v_breakfast_carbohydrates   VARCHAR(10),
    IN      v_breakfast_fat         VARCHAR(10),
    IN      v_lunch_calories        VARCHAR(10),
    IN      v_lunch_proteins        VARCHAR(10),
    IN      v_lunch_carbohydrates   VARCHAR(10),
    IN      v_lunch_fat             VARCHAR(10),
    IN      v_dinner_calories       VARCHAR(10),
    IN      v_dinner_proteins       VARCHAR(10),
    IN      v_dinner_carbohydrates  VARCHAR(10),
    IN      v_dinner_fat            VARCHAR(10),
      OUT v_result                  INT)
BEGIN
   DECLARE v_count_breakfast   INT DEFAULT 0;
   DECLARE v_count_lunch       INT DEFAULT 0;
   DECLARE v_count_dinner      INT DEFAULT 0;
declare v_cust_id INT default 0;
declare v_diet_start_date date default now();
declare v_dietician_id int default 0;
declare v_diet_Plan_id int default 0;
   DECLARE `_rollback` BOOL DEFAULT 0;
 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
   START TRANSACTION;

select cust_id,diet_start_date,dietician_id,diet_Plan_id into v_cust_id,v_diet_start_date,v_dietician_id,v_diet_Plan_id
 from diet_plan where concat(cust_id,';',diet_start_date)=v_custid_dietstartdate;

  SELECT count(1)
    INTO v_count_breakfast
    FROM diet_plan_details
   WHERE     cust_id = v_cust_id
and dietician_id = v_dietician_id
and diet_Plan_id = v_diet_Plan_id
        AND day =v_day
        AND meal_type = 'Breakfast';

  IF v_count_breakfast = 0
  THEN
    INSERT INTO diet_plan_details(cust_id,dietician_id,diet_plan_id,
                          day,
                          meal_type,
```

```
44      INSERT INTO diet_plan_details(cust_id,dietician_id,diet_plan_id,
45                                    day,
46                                    meal_type,
47                                    calories,
48                                    proteins,
49                                    carbohydrates,
50                                    fat)
51          VALUES (v_cust_id,v_dietician_id,v_diet_plan_id,
52                  v_day,
53                  'Breakfast',
54                  v_breakfast_calories,
55                  v_breakfast_proteins,
56                  v_breakfast_carbohydrates,
57                  v_breakfast_fat);
58    ELSEIF v_count_breakfast > 0
59    THEN
60      UPDATE diet_plan_details
61        SET calories = v_breakfast_calories,
62            proteins = v_breakfast_proteins,
63            carbohydrates = v_breakfast_carbohydrates,
64            fat = v_breakfast_fat
65       WHERE       cust_id = v_cust_id
66  and dietician_id = v_dietician_id
67  and diet_Plan_id = v_diet_Plan_id
68          AND day =v_day
69          AND meal_type = 'Breakfast';
70    END IF;
71
72    SELECT count(1)
73      INTO v_count_lunch
74      FROM diet_plan_details
75     WHERE     cust_id = v_cust_id
76  and dietician_id = v_dietician_id
77  and diet_Plan_id = v_diet_Plan_id
78          AND day =v_day
79          AND meal_type = 'Lunch';
80
81    IF v_count_lunch = 0
82    THEN
83      INSERT INTO diet_plan_details(cust_id,dietician_id,diet_plan_id,
84                                    day,
85                                    meal_type,
86                                    calories,
87                                    proteins,
88                                    carbohydrates,
89                                    fat)
```

```
88                              carbohydrates,
89                              fat)
90          VALUES (v_cust_id,v_dietician_id,v_diet_plan_id,
91                  v_day,
92                  'Lunch',
93                  v_lunch_calories,
94                  v_lunch_proteins,
95                  v_lunch_carbohydrates,
96                  v_lunch_fat);
97      ELSEIF v_count_lunch > 0
98      THEN
99        UPDATE diet_plan_details
100         SET calories = v_lunch_calories,
101           proteins = v_lunch_proteins,
102           carbohydrates = v_lunch_carbohydrates,
103           fat = v_lunch_fat
104       WHERE      cust_id = v_cust_id
105   and dietician_id = v_dietician_id
106   and diet_Plan_id = v_diet_Plan_id
107         AND day =v_day
108         AND meal_type = 'Lunch';
109     END IF;
110
111
112       SELECT count(1)
113      INTO v_count_dinner
114      FROM diet_plan_details
115     WHERE   cust_id = v_cust_id
116   and dietician_id = v_dietician_id
117   and diet_Plan_id = v_diet_Plan_id
118         AND day =v_day
119         AND meal_type = 'Dinner';
120
121     IF v_count_dinner = 0
122     THEN
123       INSERT INTO diet_plan_details(cust_id,dietician_id,diet_plan_id,
124                               day,
125                               meal_type,
126                               calories,
127                               proteins,
128                               carbohydrates,
129                               fat)
130         VALUES (v_cust_id,v_dietician_id,v_diet_plan_id,
131                 v_day,
132                 'Dinner'
```

```
                                carbohydrates,
                                fat)
            VALUES (v_cust_id,v_dietician_id,v_diet_plan_id,
                    v_day,
                    'Dinner',
                    v_dinner_calories,
                    v_dinner_proteins,
                    v_dinner_carbohydrates,
                    v_dinner_fat);
    ELSEIF v_count_dinner > 0
    THEN
      UPDATE diet_plan_details
        SET calories = v_dinner_calories,
            proteins = v_dinner_proteins,
            carbohydrates = v_dinner_carbohydrates,
            fat = v_dinner_fat
       WHERE       cust_id = v_cust_id
  and dietician_id = v_dietician_id
  and diet_Plan_id = v_diet_Plan_id
          AND day =v_day
          AND meal_type = 'Dinner';
    END IF;
  IF `_rollback` THEN ROLLBACK;     SET v_result = -1;
  ELSE COMMIT;    SET v_result = 1;
  END IF;
 END;
 $$
 DELIMITER ;
```

2. d_finalize_new_diet_plan

```
DELIMITER $$
CREATE PROCEDURE d_finalize_new_diet_plan(
  IN     v_custid_dietstartdate  VARCHAR(50),
    OUT v_result                        INT)
BEGIN

 DECLARE `_rollback` BOOL DEFAULT 0;
 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
  START TRANSACTION;
  update diet_plan set new_diet_plan_ind='N' where
  concat(cust_id,';',diet_start_date)=v_custid_dietstartdate;
 IF `_rollback` THEN ROLLBACK;    SET v_result = -1;
 ELSE COMMIT;    SET v_result = 1;
 END IF;
END;
$$
DELIMITER ;
```

3. d_finalize_update_diet_plan

```
DELIMITER $$
CREATE PROCEDURE d_finalize_update_diet_plan(
  IN     v_custid_dietstartdate  VARCHAR(50),
    OUT v_result                        INT)
BEGIN

  DECLARE `_rollback` BOOL DEFAULT 0;
 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
  START TRANSACTION;
update customer_requests set update_diet_plan_ind='N'
where exists (select * from diet_plan where diet_plan.cust_id=customer_requests.cust_id
and  customer_requests.dietician_id=diet_plan.dietician_id and
customer_requests.diet_plan_id=diet_plan.diet_plan_id
 and concat(cust_id,';',diet_start_date)=v_custid_dietstartdate);
 IF `_rollback` THEN
  ROLLBACK;
    SET v_result = -1;
 ELSE
  COMMIT;
   SET v_result = 1;
 END IF;
END;
$$
```

6.4.4. Restaurant

1. proc_update_menu

```
-- Procedure to update Menu
drop procedure if exists proc_update_menu;
DELIMITER $$
create procedure proc_update_menu(
IN     v_restaurant_id  INT,
                        IN     v_item_name      VARCHAR(10),
                        IN     v_monday         VARCHAR(1),
                        IN     v_tuesday        VARCHAR(1),
                        IN     v_wednesday      VARCHAR(1),
                        IN     v_thursday       VARCHAR(1),
                        IN     v_friday         VARCHAR(1),
                        IN     v_saturday       VARCHAR(1),
                        IN     v_sunday         VARCHAR(1),
                          OUT v_result         INT)
BEGIN



  DECLARE v_count   INT DEFAULT 0;
  DECLARE v_item    INT DEFAULT 0;
  DECLARE `_rollback` BOOL DEFAULT 0;
   DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
   SET v_result = 0;
   START TRANSACTION;

  SELECT count(*)
    INTO v_item
    FROM restaurant_item_day
   WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Monday';
   if v_monday='Y' and v_item=0 then
      INSERT INTO restaurant_item_day(restaurant_id,
                              item_name,
                              day_of_week)
     VALUES (v_restaurant_id, v_item_name, 'Monday');
   elseif v_monday='N' and v_item=1 then
      delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Monday';
   end if;

```

```
SELECT count(*)
 INTO v_item
 FROM restaurant_item_day
WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Tuesday';
if v_tuesday='Y' and v_item=0 then
    INSERT INTO restaurant_item_day(restaurant_id,
                                    item_name,
                                    day_of_week)
   VALUES (v_restaurant_id, v_item_name, 'Tuesday');
elseif v_tuesday='N' and v_item=1 then
   delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Tuesday';
end if;

SELECT count(*)
INTO v_item
FROM restaurant_item_day
WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Wednesday';
if v_wednesday='Y' and v_item=0 then
    INSERT INTO restaurant_item_day(restaurant_id,
                                    item_name,
                                    day_of_week)
   VALUES (v_restaurant_id, v_item_name, 'Wednesday');
elseif v_wednesday='N' and v_item=1 then
   delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Wednesday';
end if;


SELECT count(*)
INTO v_item
FROM restaurant_item_day
WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Thursday';
if v_thursday='Y' and v_item=0 then
    INSERT INTO restaurant_item_day(restaurant_id,
                                    item_name,
                                    day_of_week)
   VALUES (v_restaurant_id, v_item_name, 'Thursday');
elseif v_thursday='N' and v_item=1 then
   delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Thursday';
```

```
    delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Thursday';
  end if;

  SELECT count(*)
  INTO v_item
  FROM restaurant_item_day
  WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Friday';
  if v_friday='Y' and v_item=0 then
     INSERT INTO restaurant_item_day(restaurant_id,
                                item_name,
                                day_of_week)
    VALUES (v_restaurant_id, v_item_name, 'Friday');
  elseif v_friday='N' and v_item=1 then
    delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Friday';
  end if;

  SELECT count(*)
  INTO v_item
  FROM restaurant_item_day
  WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Saturday';
  if v_saturday='Y' and v_item=0 then
     INSERT INTO restaurant_item_day(restaurant_id,
                                item_name,
                                day_of_week)
    VALUES (v_restaurant_id, v_item_name, 'Saturday');
  elseif v_saturday='N' and v_item=1 then
    delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Saturday';
  end if;

  SELECT count(*)
  INTO v_item
  FROM restaurant_item_day
  WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Sunday';
  if v_sunday='Y' and v_item=0 then
     INSERT INTO restaurant_item_day(restaurant_id,
                                item_name,
                                day_of_week)
    VALUES (v_restaurant_id, v_item_name, 'Sunday');
```

```
    delete from restaurant_item_day where restaurant_id = v_restaurant_id AND item_name = v_item_name and day_of_week='Sunday';
  end if;
  commit;

  IF `_rollback` THEN ROLLBACK;
  ELSE COMMIT;
  END IF;


END;
$$

DELIMITER ;
```

2. r_add_menu_item

```
45      THEN
46        INSERT INTO restaurant_item_day(restaurant_id,
47                                        item_name,
48                                        day_of_week)
49              VALUES (v_restaurant_id, v_item_name,'Wednesday');
50      END IF;
51    IF v_thursday = 'Y'
52      THEN
53        INSERT INTO restaurant_item_day(restaurant_id,
54                                        item_name,
55                                        day_of_week)
56            VALUES (v_restaurant_id, v_item_name, 'Thursday');
57      END IF;
58   IF v_friday = 'Y'
59      THEN
60        INSERT INTO restaurant_item_day(restaurant_id,
61                                        item_name,
62                                        day_of_week)
63            VALUES (v_restaurant_id, v_item_name, 'Friday');
64      END IF;
65     IF v_saturday = 'Y'
66      THEN
67        INSERT INTO restaurant_item_day(restaurant_id,
68                                        item_name,
69                                        day_of_week)
70            VALUES (v_restaurant_id, v_item_name, 'Saturday');
71      END IF;
72    IF v_sunday = 'Y'
73      THEN
74        INSERT INTO restaurant_item_day(restaurant_id,
75                                        item_name,
76                                        day_of_week)
77            VALUES (v_restaurant_id, v_item_name, 'Sunday');
78      END IF;
79      COMMIT;
80      SET v_result = 1;
81   elseif v_item > 0
82     THEN
83        SET v_result = 2;
84    END IF;
85    IF `_rollback` THEN ROLLBACK;
86    ELSE COMMIT;
87    END IF;
88   END;
89   $$
90   DELIMITER
```

```
DROP PROCEDURE IF EXISTS r_add_menu_item;
DELIMITER $$
CREATE PROCEDURE r_add_menu_item(IN      v_restaurant_id   INT,
  IN      v_item_name       VARCHAR(10),  IN      v_meal_type       VARCHAR(10),
  IN      v_calories        VARCHAR(10),IN      v_proteins        VARCHAR(10),
   IN      v_carbohydrates   VARCHAR(10), IN      v_fat             VARCHAR(10),
   IN      v_monday          VARCHAR(1),  IN      v_tuesday         VARCHAR(1),
  IN      v_wednesday       VARCHAR(1), IN      v_thursday        VARCHAR(1),
   IN      v_friday          VARCHAR(1), IN      v_saturday        VARCHAR(1),
  IN      v_sunday          VARCHAR(1),  OUT v_result          INT)
BEGIN
  DECLARE v_count   INT DEFAULT 0;
  DECLARE v_item    INT DEFAULT 0;
  DECLARE `_rollback` BOOL DEFAULT 0;
 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET `_rollback` = 1;
   SET v_result = 0;
 START TRANSACTION;
SELECT count(*)
    INTO v_item
    FROM item
   WHERE restaurant_id = v_restaurant_id AND item_name = v_item_name;
IF v_item = 0
  THEN
    INSERT INTO item(restaurant_id, item_name, meal_type,
                calories, proteins,carbohydrates,
                fat, offered_ind)
        VALUES (v_restaurant_id,  v_item_name,  v_meal_type,
              v_calories, v_proteins, v_carbohydrates,
              v_fat,   'N');
    IF v_monday = 'Y'
    THEN
      INSERT INTO restaurant_item_day(restaurant_id,
                                item_name,
                                day_of_week)
          VALUES (v_restaurant_id, v_item_name, 'Monday');
    END IF;
IF v_tuesday = 'Y'
    THEN
      INSERT INTO restaurant_item_day(restaurant_id,
                                item_name,
                                day_of_week)
          VALUES (v_restaurant_id, v_item_name, 'Tuesday');
    END IF;
    IF v_wednesday = 'Y'
    THEN
      INSERT INTO restaurant_item_day(restaurant_id,
```

6.5. Additional DB Objects

6.5.1. Triggers

Triggers update_offered_ind_for_insert and update_offered_ind_for_delete exist on restaurant_item_day table and are used manipulate offered_ind in Item table. The item table has an offered_ind column to indicate if the restaurant is offering the particular item on any day. If the item is being offered on any day, then the day on which the item is offered will be in restaurant_item_day table m and in item table offered_ind should be Y, else it will be N. If the restaurant wants to stop offering the item, we cannot delete the item from item table due to the necessity to maintain order history. The offered_ind helps in determining if the item is being offered on any day or not.

1. **update_offered_ind_for_insert**: If the restaurant is not offering the item currently on any day, and decides that it wants to offer the item, then records will be inserted in restaurant_item_day table. This trigger updates offered_ind to Y from N in item table.

```
1
2   DROP TRIGGER IF EXISTS  update_offered_ind_for_insert;
3
4   DELIMITER $$
5   CREATE TRIGGER update_offered_ind_for_insert
6   BEFORE INSERT
7   ON restaurant_item_day
8   FOR EACH ROW
9   BEGIN
10    DECLARE v_count INT;
11  SELECT COUNT(*) INTO v_count FROM restaurant_item_day
12  WHERE restaurant_id = NEW.restaurant_id AND item_name = NEW.item_name;
13  IF v_count>0 THEN
14  UPDATE Item
15  SET Offered_Ind = 'Y'
16  WHERE restaurant_id = NEW.restaurant_id AND item_name = NEW.item_name;
17  ELSEIF v_count=0 THEN
18   UPDATE Item
19  SET Offered_Ind = 'N'
20  WHERE restaurant_id = NEW.restaurant_id AND item_name = NEW.item_name;
21  END IF;
22  END;
23  $$
24  DELIMITER ;
25
```

2. **update_offered_ind_for_delete**: If the restaurant is currently offering an item, and wants to stop offering the item, then all records for the item in restaurant_item_day table will be deleted. In this case, the trigger updates offered_ind to N in item table.

```
DROP TRIGGER IF EXISTS update_offered_ind_for_delete;
DELIMITER $$
CREATE TRIGGER update_offered_ind_for_delete
AFTER DELETE
ON restaurant_item_day
FOR EACH ROW
BEGIN
  DECLARE v_count INT;
SELECT
   COUNT(*)
INTO v_count FROM
   restaurant_item_day
WHERE
   restaurant_id = OLD.restaurant_id
      AND item_name = OLD.item_name;
IF v_count>0 THEN
UPDATE Item
SET Offered_Ind = 'Y'
WHERE restaurant_id = OLD.restaurant_id AND item_name = OLD.item_name;
ELSEIF v_count=0 THEN
 UPDATE Item
SET Offered_Ind = 'N'
WHERE restaurant_id = OLD.restaurant_id AND item_name = OLD.item_name;
END IF;
END;
$$
DELIMITER ;
```

6.5.2. Index

Index customer_meal_plan_idx1 is created on customer_meal_plan table on cust_ID, start_date columns as these columns are parent referential keys and are part of composite primary keys of the table.

```
CREATE INDEX customer_meal_plan_idx1 ON
 customer_meal_plan (cust_ID, start_date);
```

6.6. DB Execution Screenshots

Below are DB execution screenshots for some important operations for the application:

6.6.1. Admin

1. proc_d_ins_dietn_regn_details: The procedure is called to insert the details of the dietician from UI to database. This is called during the registration of the dietician by the admin. The below screenshot shows the output from dietician table with the newly inserted dietician ‘dietician102’

```
30
31 • CALL proc_d_ins_dietn_regn_details ('dietician102', 'dietician102@mm.com', '602c57ffb51af99d6f3b54c0ee9587bb110fb990',
32     10, 'M.S.', @vc, @eid);
33
34
35 • SELECT * FROM dietician;
```

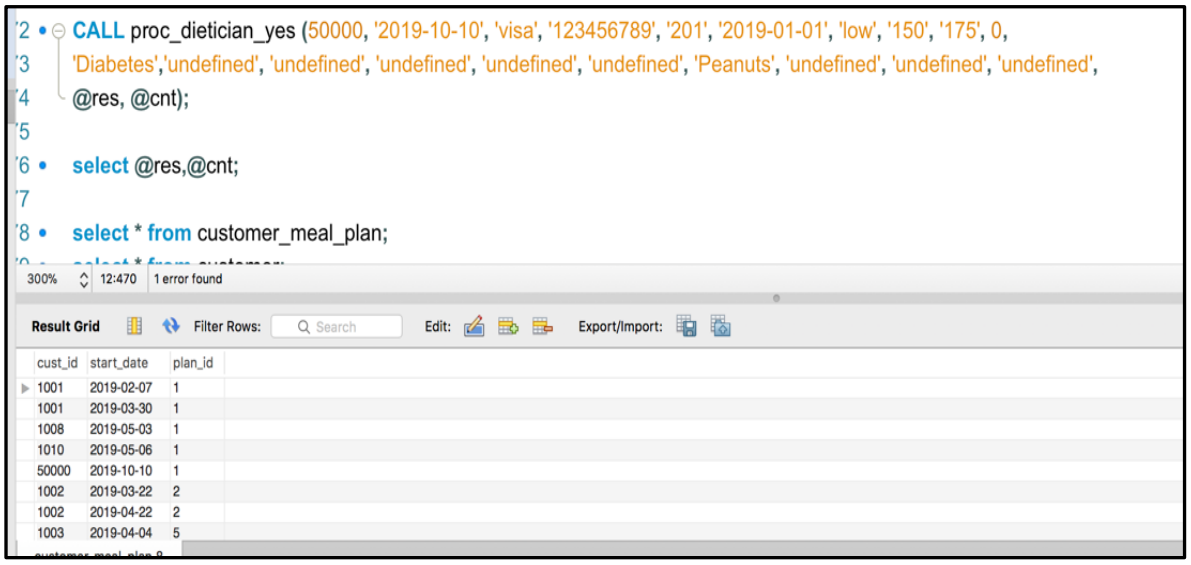
250% 12:79

Result Grid | Filter Rows: Search | Edit: | Export/Import:

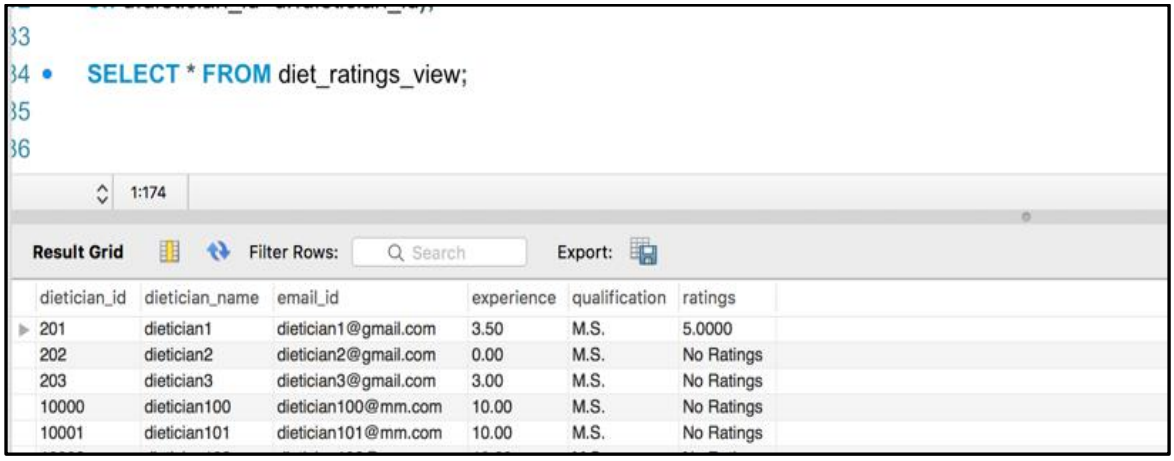
dietician_id	dietician_name	email_id	password	experience	qualification
201	dietician1	dietician1@gmail.com	88ea39439e74fa27c09a4fc0bc8ebe6d00978392	3.50	M.S.
202	dietician2	dietician2@gmail.com	88ea39439e74fa27c09a4fc0bc8ebe6d00978392	0.00	M.S.
203	dietician3	dietician3@gmail.com	88ea39439e74fa27c09a4fc0bc8ebe6d00978392	3.00	M.S.
10000	dietician100	dietician100@mm.com	2bdba55ec2c947b1c5d0f68385f2cc5a78488c24	10.00	M.S.
10001	dietician101	dietician101@mm.com	2bdba55ec2c947b1c5d0f68385f2cc5a78488c24	10.00	M.S.
10002	dietician102	dietician102@mm.com	2bdba55ec2c947b1c5d0f68385f2cc5a78488c24	10.00	M.S.
NULL	NULL	NULL	NULL	NULL	NULL

6.6.2. Customer

1. proc_dietician_yes: The procedure is called to insert the details of customer who wishes to enroll with a diet plan. The below screenshot shows the output from one of the updated tables which is customer_meal_plan with the new cust_ID 50000.



2. diet_ratings_view: This view is created to combine the average ratings received by each dietician from all the customers, with the other details of the dietician like name, email ID and qualification. If no ratings are provided to a particular dietician, then 'No Ratings' is displayed. The below screenshot shows the sample data executed from the view. It is seen that only dieticain1 has the ratings.



3. c_customer_remaining_meal_count: For each customer this view provides the number of Breakfast, Lunch and Dinner meals left in his current meal plan. Below screenshot shows the remaining meal count for the current users from sample data.

```
234 •   SELECT * FROM c_customer_remaining_meal_count;
235
236
```

100% 47:234

Result Grid Filter Rows: Search Export:

cust_id	start_date	rem_breakfast_count	rem_lunch_count	rem_dinner_count
1008	2019-05-03	30	30	30
1010	2019-05-06	30	30	30
1002	2019-04-22	0	0	15
1003	2019-05-04	0	15	15
1004	2019-05-03	0	30	30
1005	2019-05-01	0	30	15
1009	2019-05-06	30	15	0
1001	2019-04-30	30	30	30
1007	2019-04-08	30	30	30

4. c_diet_history_view: The purpose of this view is to return the complete information for the list of diet plans each customer has previously enrolled. Below screenshot shows the complete diet plan details of first 9 days along with the diet progress with respect to cust id 1001 and dietician id 201.

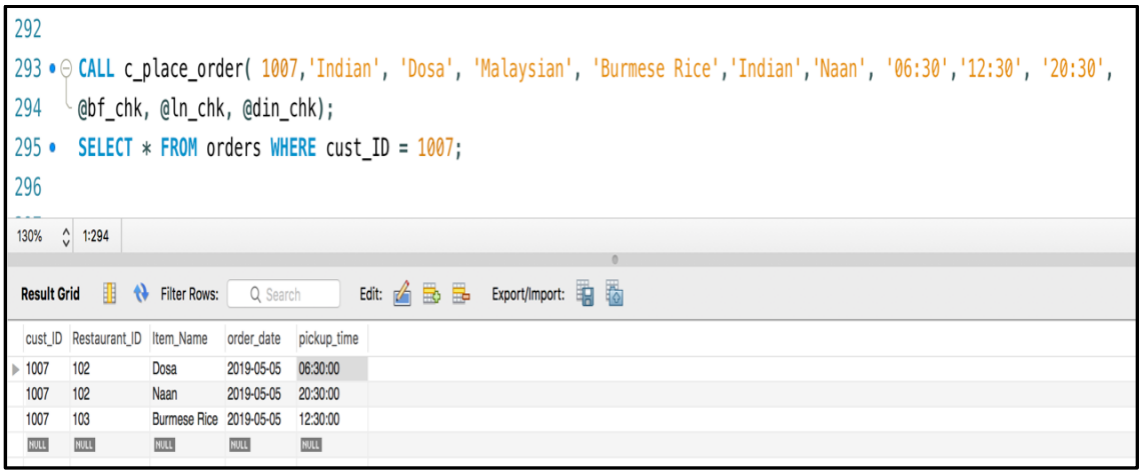
```
406
407 •  SELECT * FROM c_diet_history_view;
408
409
```

100% 1:408

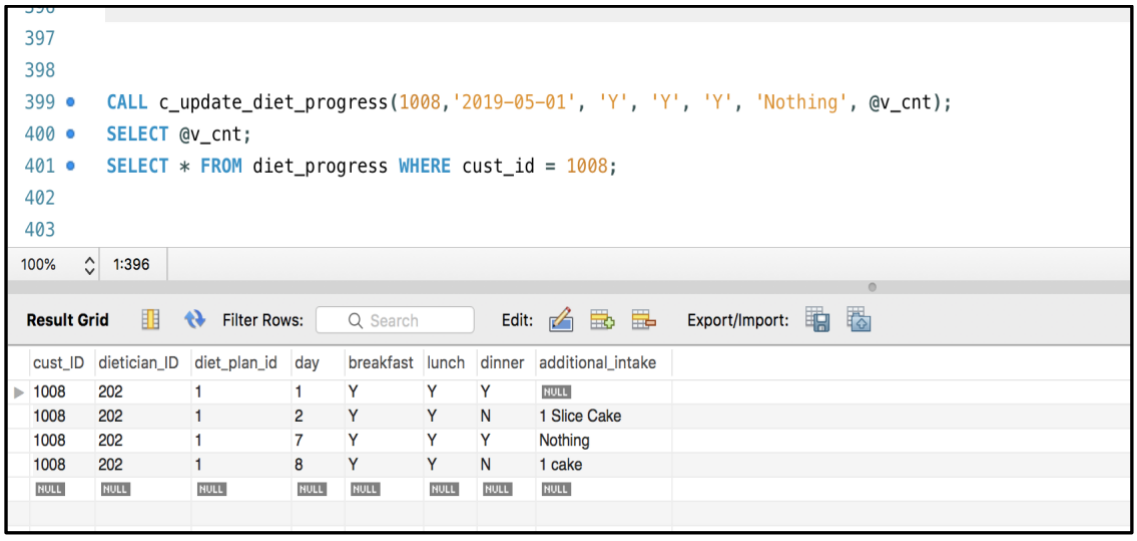
Result Grid | Filter Rows: Search | Export:

cust_id	dietician_id	diet_plan_id	day	meal_type	calories	proteins	carbohydrates	fat	followed	additional_intake
1001	201	1	1	Breakfast	100.00	20.00	20.00	30.00	Y	NULL
1001	201	1	1	Dinner	100.00	40.00	100.00	30.00	Y	NULL
1001	201	1	1	Lunch	180.00	20.00	80.00	30.00	Y	NULL
1001	201	1	2	Breakfast	120.00	20.00	20.00	30.00	Y	Biryani
1001	201	1	2	Dinner	130.00	45.00	100.00	30.00	N	Biryani
1001	201	1	2	Lunch	150.00	30.00	70.00	30.00	N	Biryani
1001	201	1	3	Breakfast	120.00	30.00	30.00	30.00	NULL	NULL
1001	201	1	3	Dinner	130.00	45.00	10.00	30.00	NULL	NULL
1001	201	1	3	Lunch	150.00	30.00	75.00	30.00	NULL	NULL
1001	201	1	4	Breakfast	100.00	20.00	20.00	10.00	Y	NULL
1001	201	1	4	Dinner	130.00	45.00	80.00	30.00	Y	NULL
1001	201	1	4	Lunch	140.00	30.00	70.00	20.00	N	NULL
1001	201	1	5	Breakfast	170.00	20.00	20.00	30.00	Y	Apple Pie
1001	201	1	5	Dinner	130.00	45.00	70.00	30.00	N	Apple Pie
1001	201	1	5	Lunch	150.00	35.00	70.00	30.00	Y	Apple Pie
1001	201	1	6	Breakfast	120.00	20.00	20.00	10.00	Y	NULL
1001	201	1	6	Dinner	130.00	45.00	60.00	30.00	Y	NULL
1001	201	1	6	Lunch	150.00	75.00	70.00	20.00	Y	NULL
1001	201	1	7	Breakfast	100.00	20.00	20.00	30.00	Y	NULL
1001	201	1	7	Dinner	110.00	45.00	60.00	30.00	Y	NULL
1001	201	1	7	Lunch	150.00	85.00	70.00	30.00	Y	NULL
1001	201	1	8	Breakfast	170.00	35.00	20.00	10.00	NULL	NULL
1001	201	1	8	Dinner	130.00	45.00	85.00	30.00	NULL	NULL
1001	201	1	8	Lunch	150.00	35.00	65.00	30.00	NULL	NULL
1001	201	1	9	Breakfast	120.00	35.00	50.00	10.00	Y	Pastry
1001	201	1	9	Dinner	130.00	25.00	85.00	30.00	Y	Pastry
1001	201	1	9	Lunch	150.00	35.00	75.00	30.00	Y	Pastry
1001	201	1	10	Breakfast	130.00	35.00	50.00	20.00	Y	NULL

5. c_place_order: This stored procedure is used to place an order for the items selected by the customer. Below screenshot shows the execution while customer 1008 places an order for breakfast, lunch and dinner meal types.

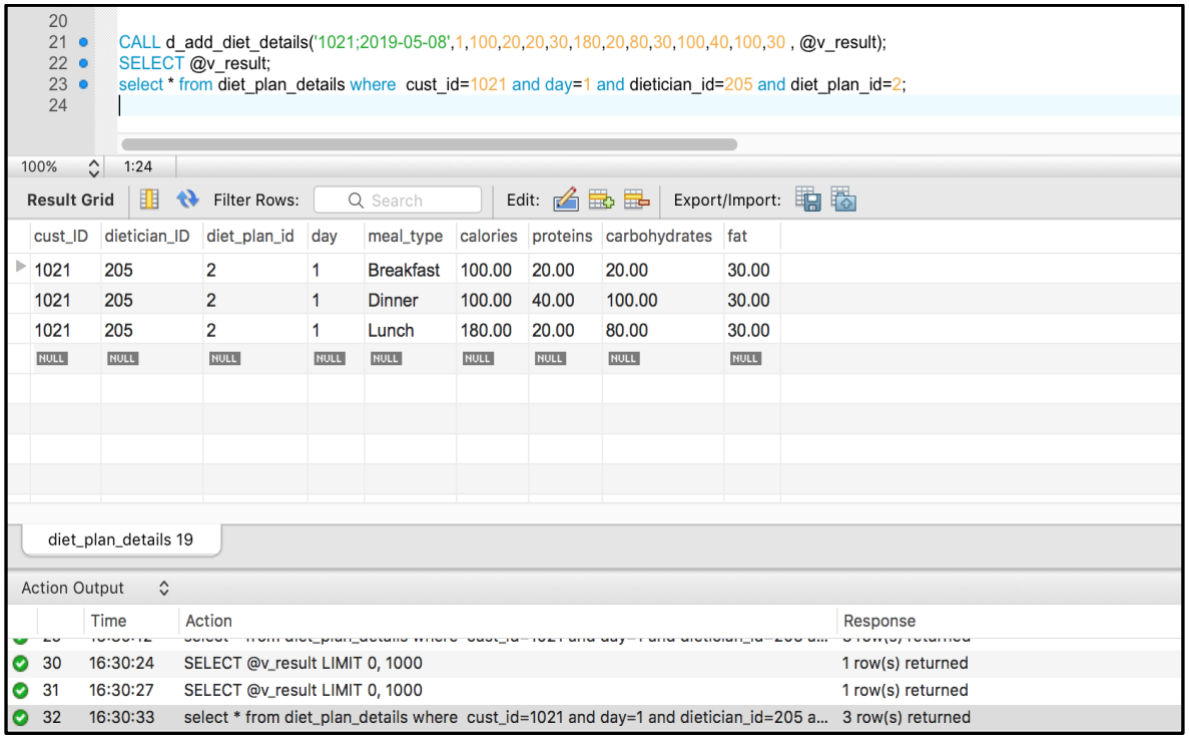


6. c_update_diet_progress: This stored procedure is used to insert the new diet progress information provided by customer. Below screenshot shows execution when customer 1008 tries to update his progress for 1st May.



6.6.3. Dietician

7. d_add_diet_details: This procedure is used by dietician to add diet plan details for a customer for a particular day for all 3 meal types.



2. d_customer_diet_plan_details_view - This view gives the diet plan details for all 30 days for all 3 meal types – breakfast, lunch and dinner for all the diet plans ids of a customer.

```
1  select * from d_customer_diet_plan_details_view;
```

100% 49:1

Result Grid | Filter Rows: Search | Export:

cust...	dietician...	diet_start_d...	day	calendar_date	new_di...	meal_type	calories	proteins	carbohydrates	fat
1001	201	2019-02-07	1	2019-02-08	Y	Breakfast	100.00	20.00	20.00	30.00
1001	201	2019-02-07	1	2019-02-08	Y	Lunch	180.00	20.00	80.00	30.00
1001	201	2019-02-07	1	2019-02-08	Y	Dinner	100.00	40.00	100.00	30.00
1001	201	2019-02-07	2	2019-02-09	Y	Breakfast	120.00	20.00	20.00	30.00
1001	201	2019-02-07	2	2019-02-09	Y	Lunch	150.00	30.00	70.00	30.00
1001	201	2019-02-07	2	2019-02-09	Y	Dinner	130.00	45.00	100.00	30.00
1001	201	2019-02-07	3	2019-02-10	Y	Breakfast	120.00	30.00	30.00	30.00
1001	201	2019-02-07	3	2019-02-10	Y	Lunch	150.00	30.00	75.00	30.00
1001	201	2019-02-07	3	2019-02-10	Y	Dinner	130.00	45.00	10.00	30.00
1001	201	2019-02-07	4	2019-02-11	Y	Breakfast	100.00	20.00	20.00	10.00
1001	201	2019-02-07	4	2019-02-11	Y	Lunch	140.00	30.00	70.00	20.00
1001	201	2019-02-07	4	2019-02-11	Y	Dinner	130.00	45.00	80.00	30.00
1001	201	2019-02-07	5	2019-02-12	Y	Breakfast	170.00	20.00	20.00	30.00
1001	201	2019-02-07	5	2019-02-12	Y	Lunch	150.00	35.00	70.00	30.00
1001	201	2019-02-07	5	2019-02-12	Y	Dinner	130.00	45.00	70.00	30.00
1001	201	2019-02-07	6	2019-02-13	Y	Breakfast	120.00	20.00	20.00	10.00
1001	201	2019-02-07	6	2019-02-13	Y	Lunch	150.00	75.00	70.00	20.00
1001	201	2019-02-07	6	2019-02-13	Y	Dinner	130.00	45.00	60.00	30.00

3. d_customer_diet_plans_list_view – This view is used to find all diet plans associated with a customer along with start dates, pay dates, dietician of the diet plans.

```
1  select * from d_customer_diet_plans_list_view;
```

100% 15:1

Result Grid | Filter Rows: Search | Export:

cust_id	cust_n...	cust_email_id	cu...	dieti...	height	weight	acti...	sleep	di...	diet_start_d...	new_diet_plan_request_date	new_diet_plan_ind
1001	Sara	Sara@gmail.com	26	201	150.00	200.00	low	0	1	2019-02-07	2019-02-04	Y
1001	Sara	Sara@gmail.com	26	202	153.00	100.00	me...	2	1	2019-03-30	2019-03-27	Y
1021	James	James@gmail.com	25	204	150.00	200.00	low	7	1	2019-03-09	2019-03-06	Y
1021	James	James@gmail.com	25	205	150.00	195.00	low	7	1	2019-04-08	2019-04-05	Y
1021	James	James@gmail.com	25	205	150.00	180.00	low	8	2	2019-05-08	2019-05-05	Y
1022	Joseph	Joseph@gmail.com	33	205	150.00	200.00	low	6	1	2019-04-25	2019-04-22	Y

4. d_diet_plans_cust_health_details_view – This view shows the customer health details like allergies and diseases associated with each diet plan of the customer.

```
select * from d_diet_plans_cust_health_details_view;
```

Result Grid

cust_id	cust_name	cust_email_id	cust_age	dietician_id	height	weight	activity...	sleep	diet_pl...	diet_start_date	new_diet_plan_...	new_...	a_d_type	allergy_disease
1001	Sara	Sara@gmail.com	26	201	150.00	200.00	low	0	1	2019-02-07	2019-02-04	Y	Allergy	Lactose
1001	Sara	Sara@gmail.com	26	201	150.00	200.00	low	0	1	2019-02-07	2019-02-04	Y	Allergy	Peanuts
1001	Sara	Sara@gmail.com	26	202	153.00	100.00	medium	2	1	2019-03-30	2019-03-27	Y	Allergy	No Allergy
1021	James	James@gmail.com	25	204	150.00	200.00	low	7	1	2019-03-09	2019-03-06	Y	Allergy	Lactose
1021	James	James@gmail.com	25	204	150.00	200.00	low	7	1	2019-03-09	2019-03-06	Y	Allergy	Peanuts
1021	James	James@gmail.com	25	205	150.00	195.00	low	7	1	2019-04-08	2019-04-05	Y	Allergy	Lactose
1021	James	James@gmail.com	25	205	150.00	195.00	low	7	1	2019-04-08	2019-04-05	Y	Allergy	Peanuts
1021	James	James@gmail.com	25	205	150.00	180.00	low	8	2	2019-05-08	2019-05-05	Y	Allergy	Lactose
1021	James	James@gmail.com	25	205	150.00	180.00	low	8	2	2019-05-08	2019-05-05	Y	Allergy	Peanuts
1022	Joseph	Joseph@gmail.com	33	205	150.00	200.00	low	6	1	2019-04-25	2019-04-22	Y	Allergy	No Allergy
1001	Sara	Sara@gmail.com	26	201	150.00	200.00	low	0	1	2019-02-07	2019-02-04	Y	Disease	HBP

6.6.4. Restaurant

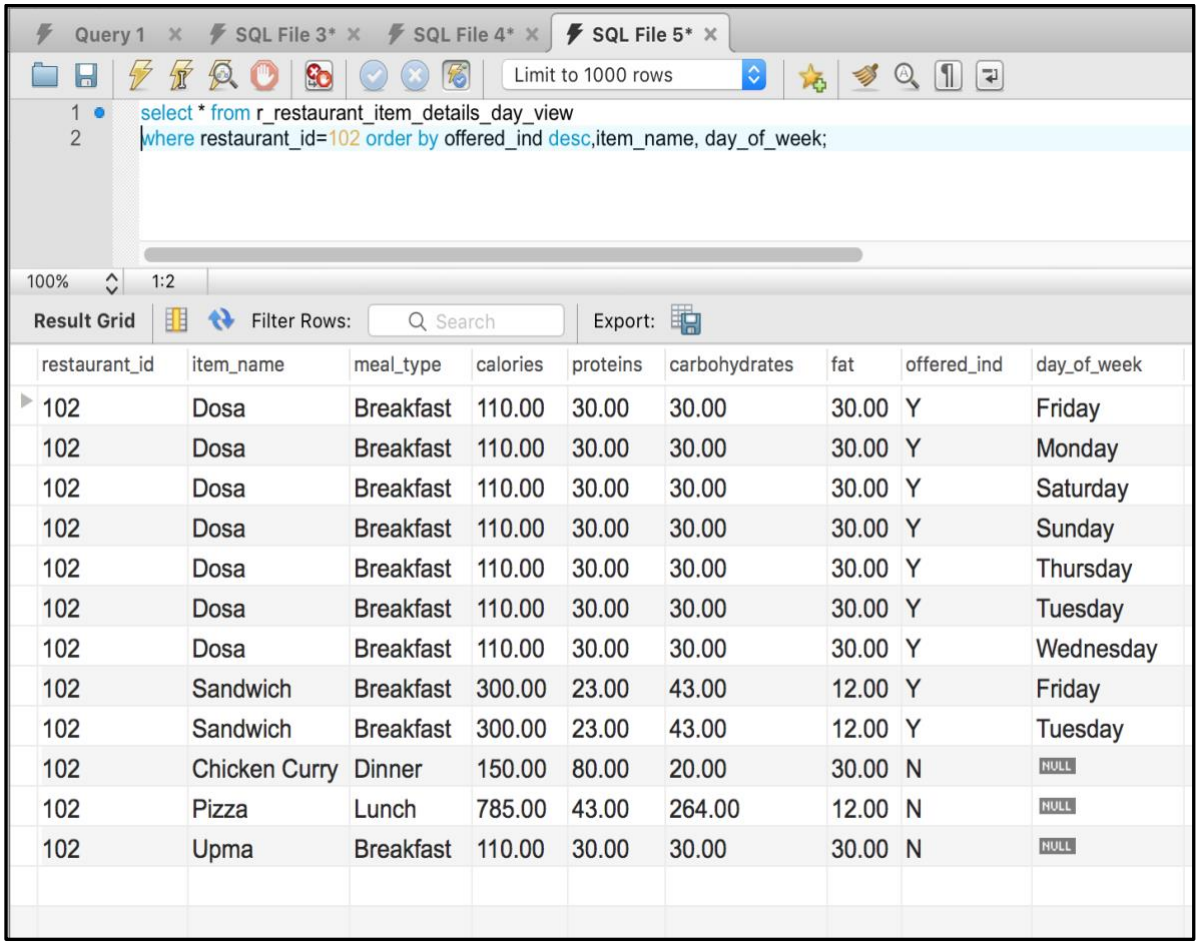
1. r_add_menu_item - This procedure is used to add menu item for a particular restaurant, and input the item name, nutritional information and days of week on which the item will offered.

```
SET @v_result = 0;
CALL r_add_menu_item(102 ,'Apple Pie','Breakfast',468,23,43,12,'Y','N','Y','Y','N','N','N',@v_result);
SELECT @v_result;
select * from item i join restaurant_item_day r on i.item_name=r.item_name
and i.restaurant_id=r.restaurant_id where r.restaurant_id=102 and r.item_name='Apple Pie';
```

Result Grid

Restaurant_ID	Item_Name	Meal_Type	Calories	Proteins	Carbohydrates	Fat	offered_ind	Restaurant_ID	Item_Name	day_of_week
102	Apple Pie	Breakfast	468.00	23.00	43.00	12.00	Y	102	Apple Pie	Monday
102	Apple Pie	Breakfast	468.00	23.00	43.00	12.00	Y	102	Apple Pie	Thursday
102	Apple Pie	Breakfast	468.00	23.00	43.00	12.00	Y	102	Apple Pie	Wednesday

2. r_restaurant_item_details_day_view – This gives the list of items and item details like calories, proteins, days of which the item is offered for a restaurant.



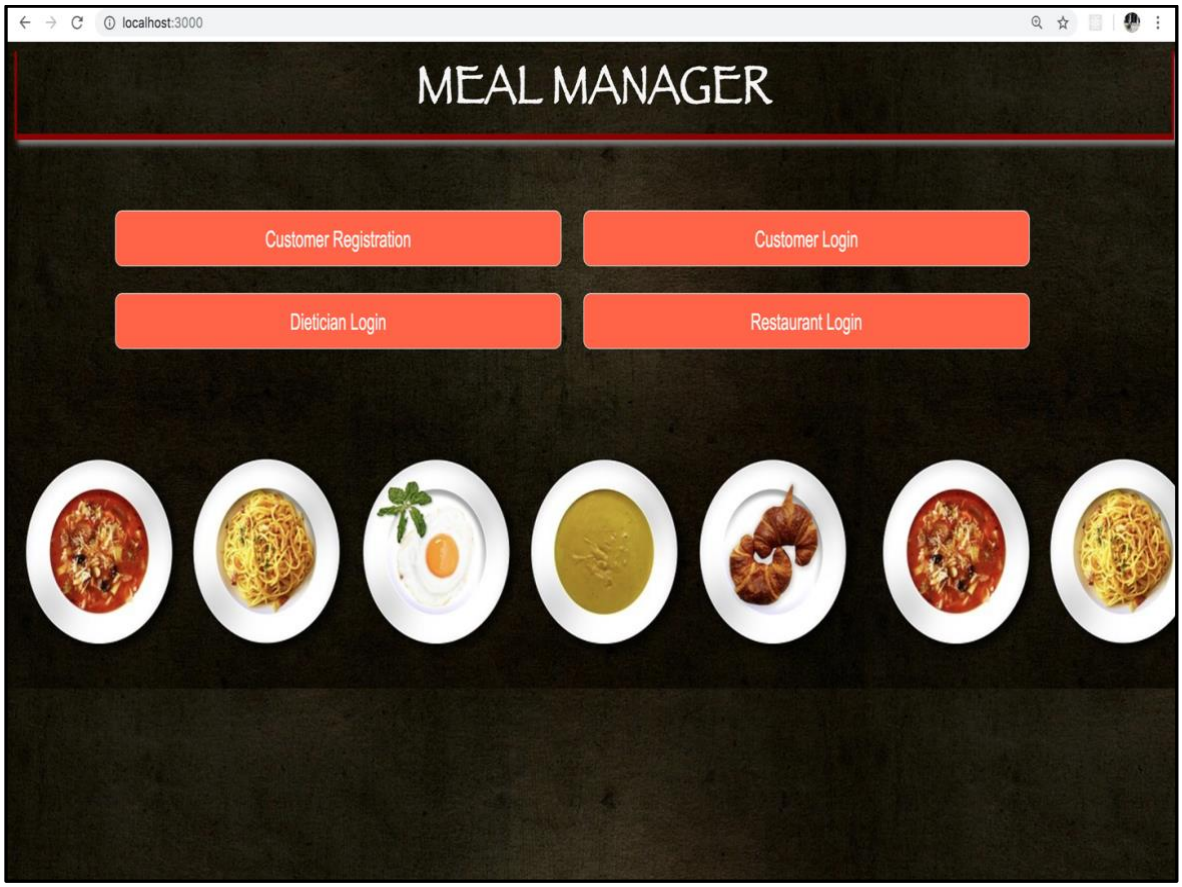
restaurant_id	item_name	meal_type	calories	proteins	carbohydrates	fat	offered_ind	day_of_week
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Friday
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Monday
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Saturday
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Sunday
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Thursday
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Tuesday
102	Dosa	Breakfast	110.00	30.00	30.00	30.00	Y	Wednesday
102	Sandwich	Breakfast	300.00	23.00	43.00	12.00	Y	Friday
102	Sandwich	Breakfast	300.00	23.00	43.00	12.00	Y	Tuesday
102	Chicken Curry	Dinner	150.00	80.00	20.00	30.00	N	NULL
102	Pizza	Lunch	785.00	43.00	264.00	12.00	N	NULL
102	Upma	Breakfast	110.00	30.00	30.00	30.00	N	NULL

7. Final Design of DB Application

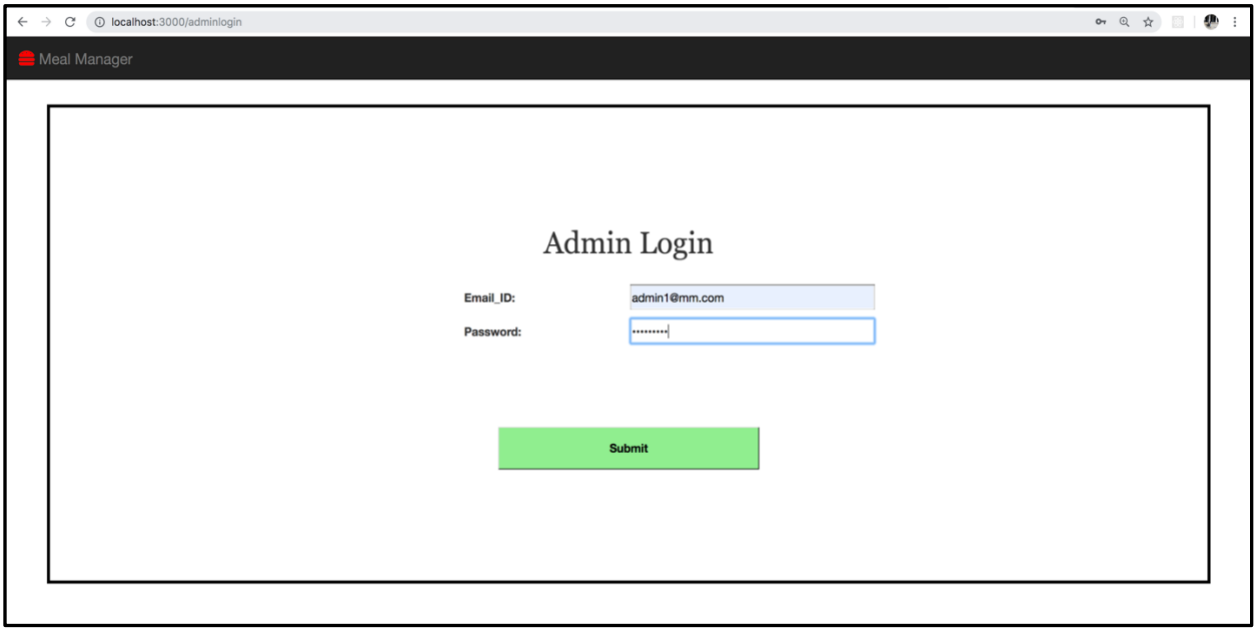
7.1. Application Screenshots

7.1.1. Admin Operations

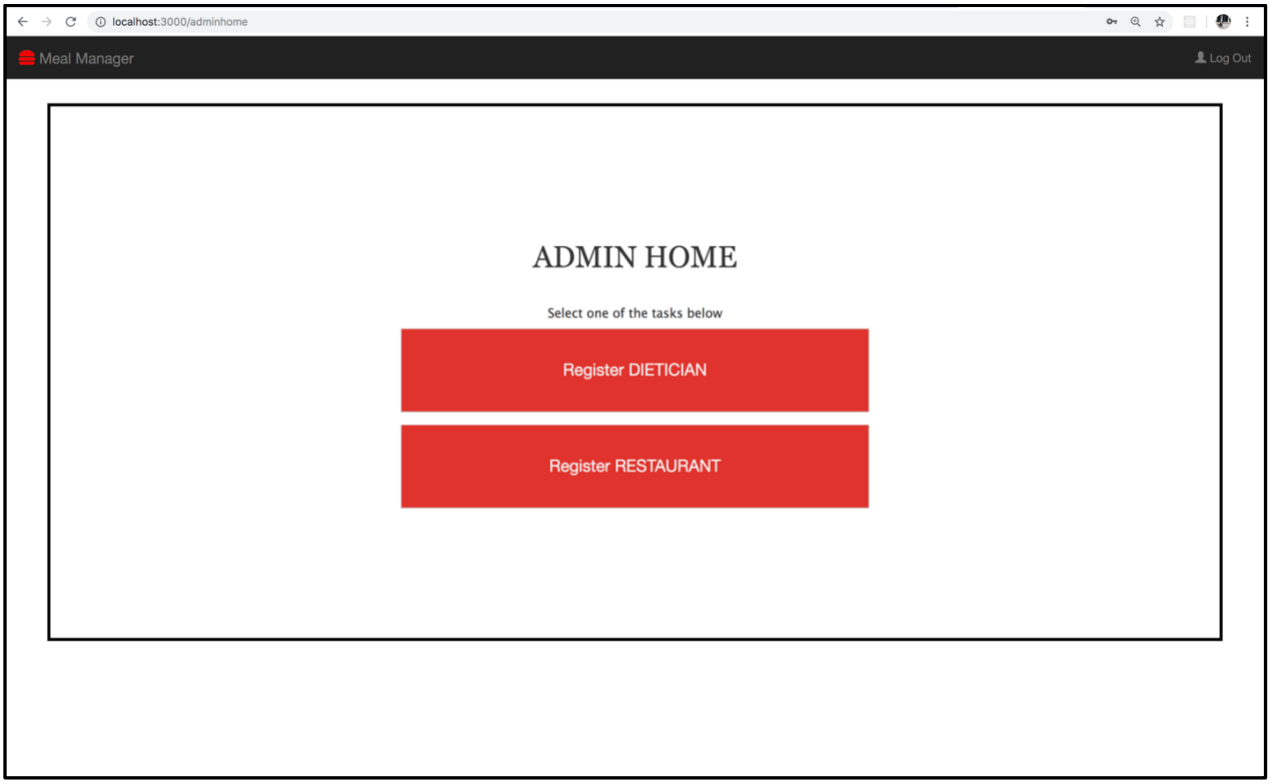
1. Application home page: onload of application homepage



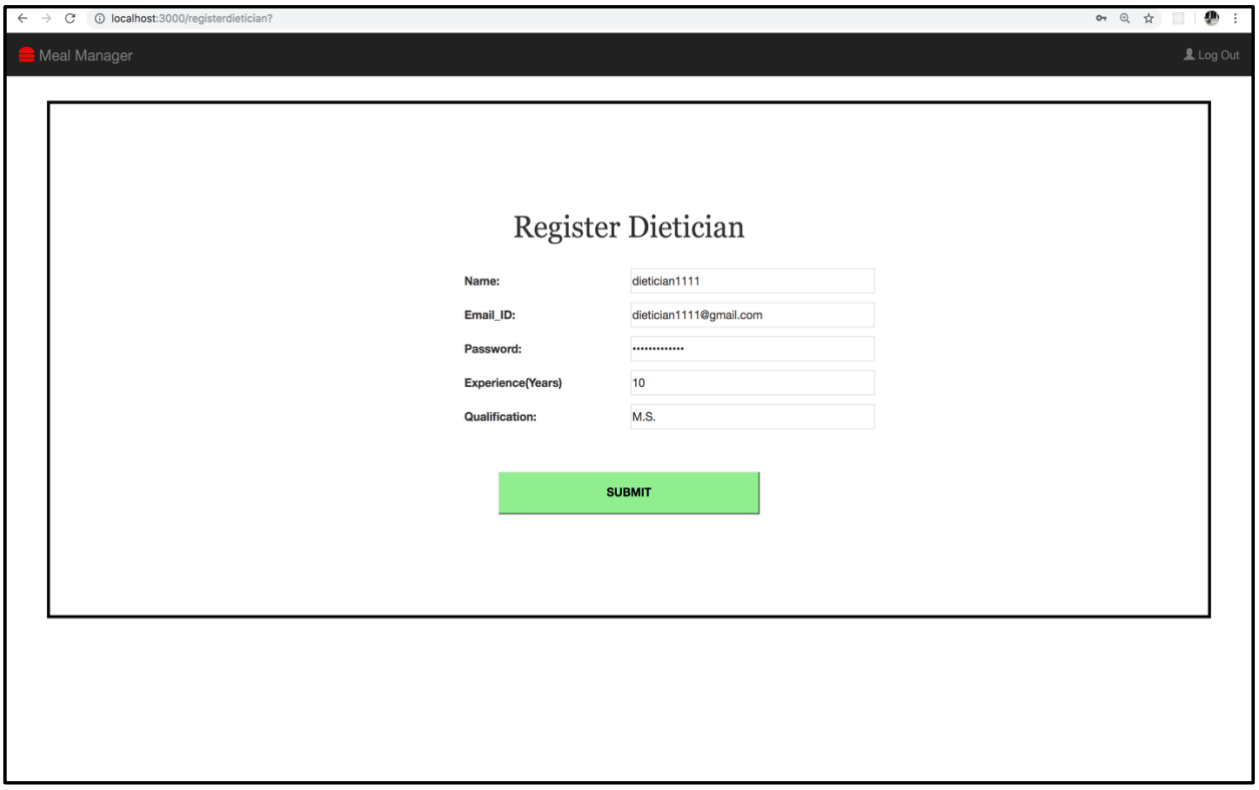
1. Admin login page: page displayed when localhost:3000/adminlogin is selected as url. Details entered after page load.



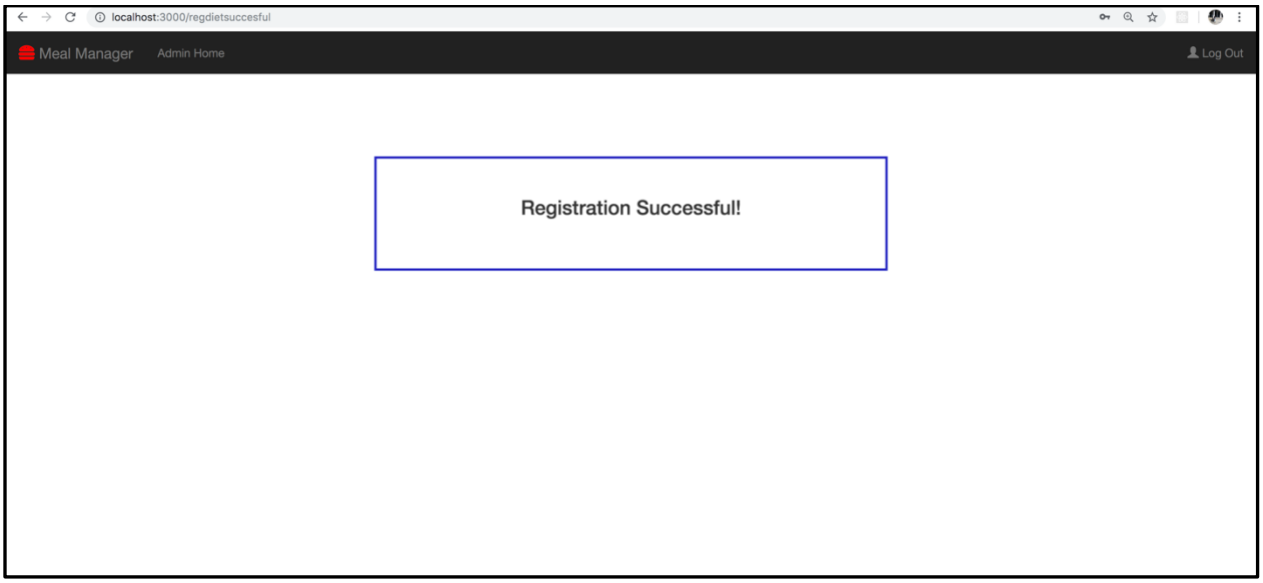
2. Admin home page: page displayed when admin successfully logs in



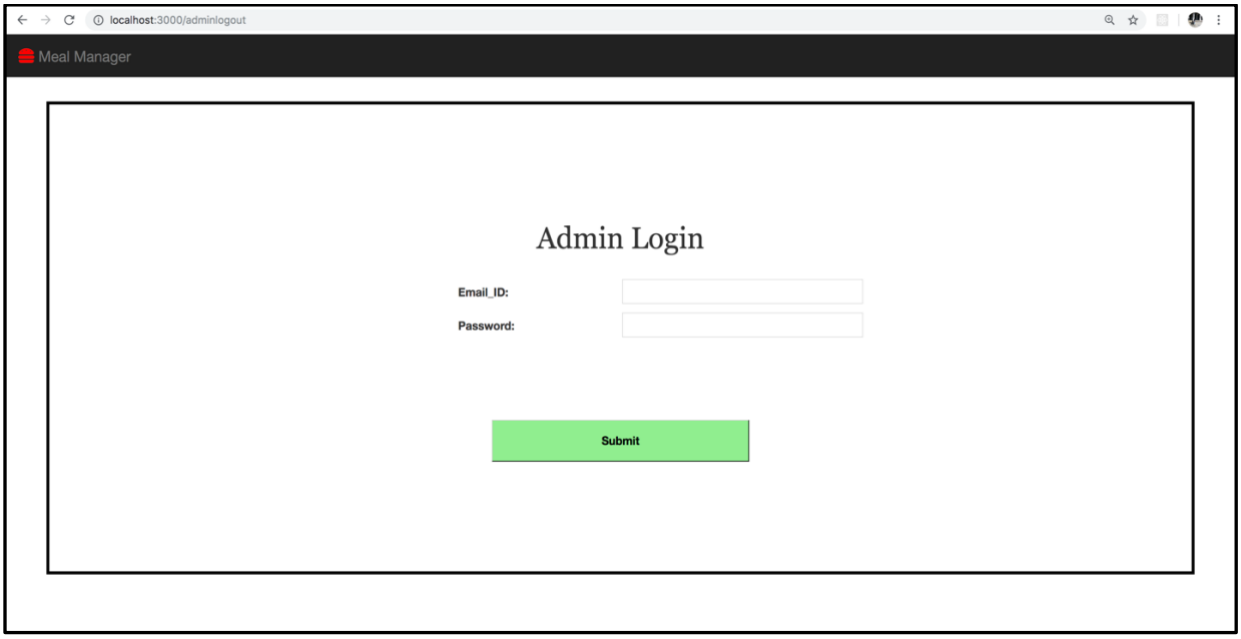
3. Register dietician page: page displayed when clicked on ‘Register Dietician’ button. Details entered after page load.



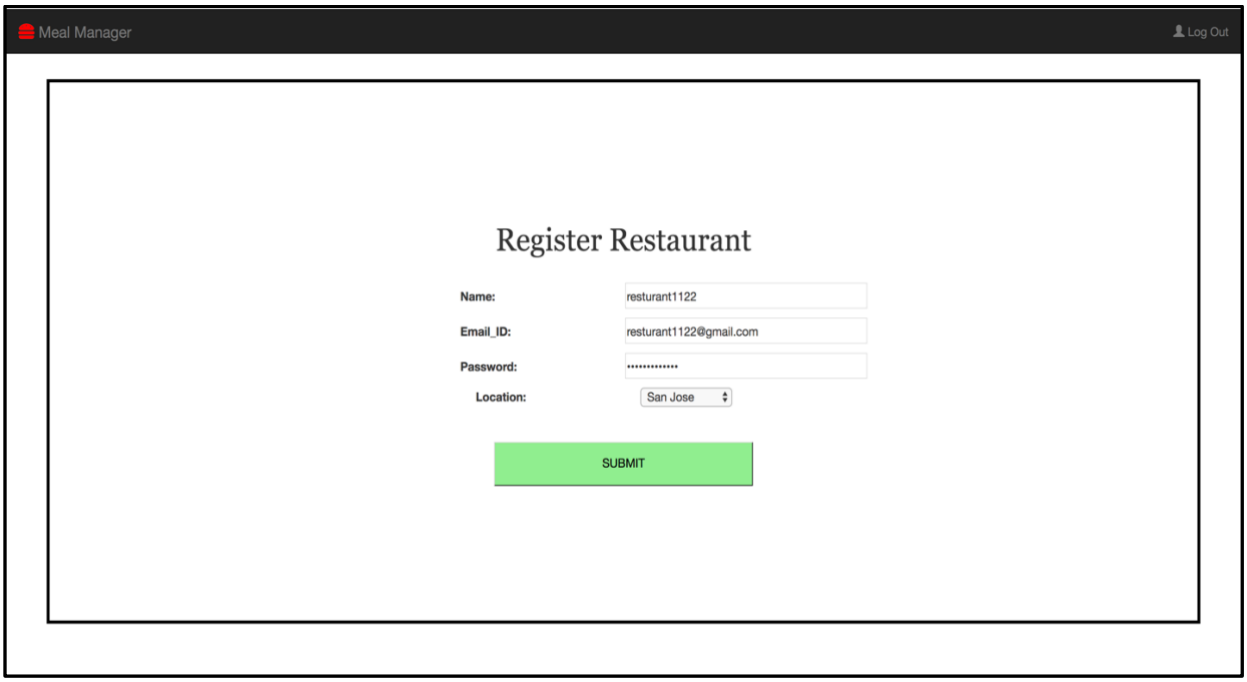
4. Register dietician successful page: page displayed on successful registration of dietician



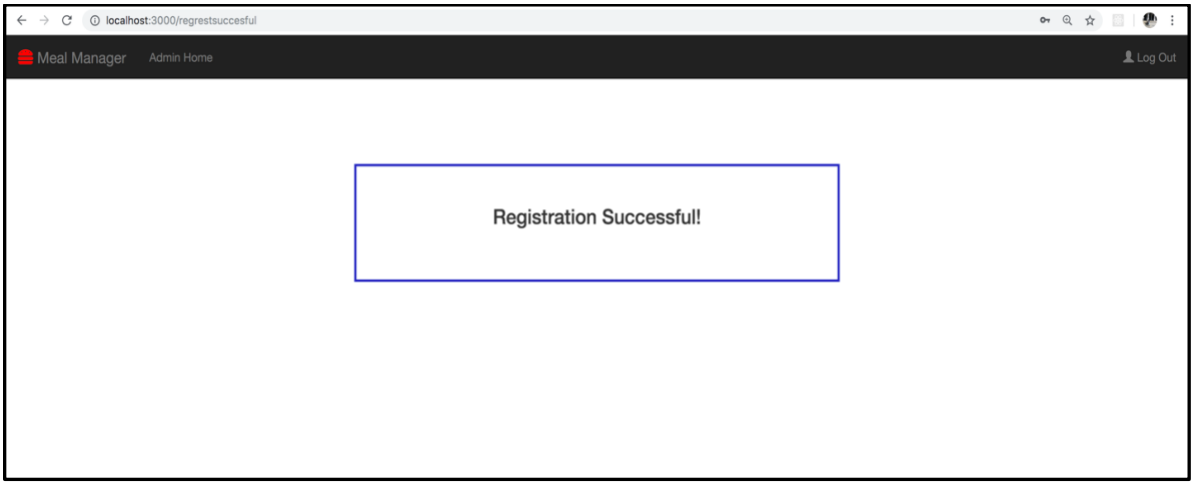
5. Admin logout page: page displayed when admin clicks on logout button



6. Register restaurant page: page displayed when Register RESTAURANT button is clicked (admin relogins with admin1). Details entered after page load.

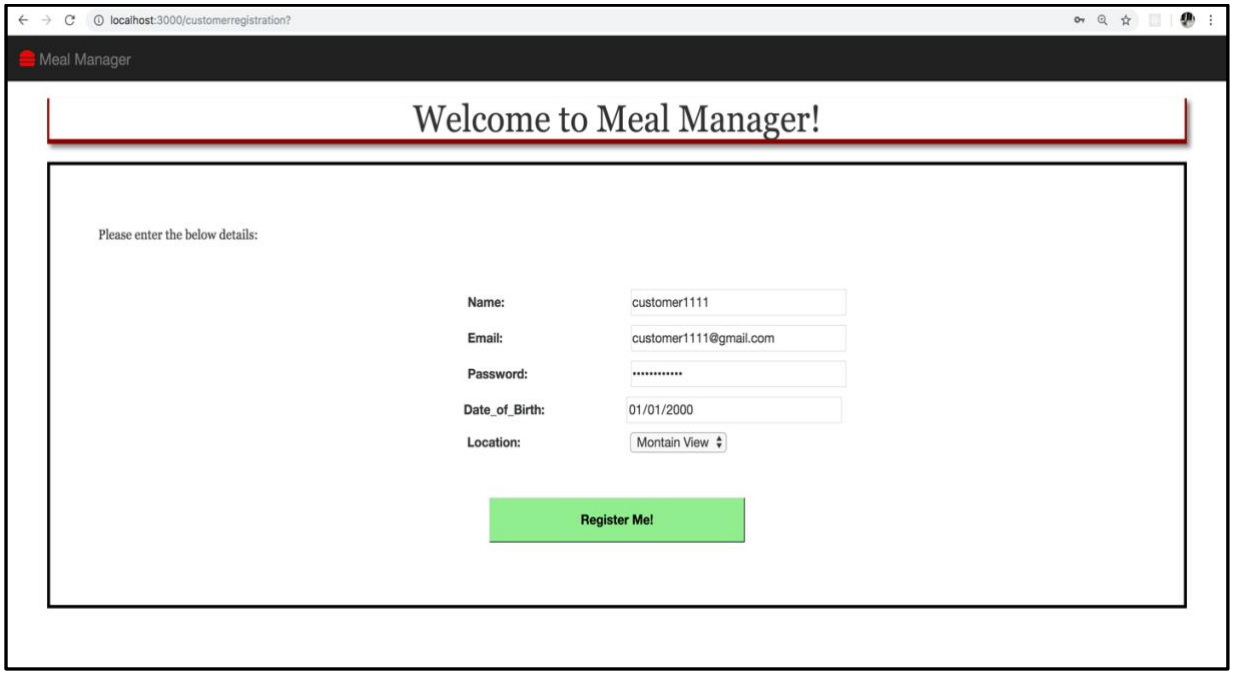


7. Register restaurant successful page: page displayed on successful registration of restaurant

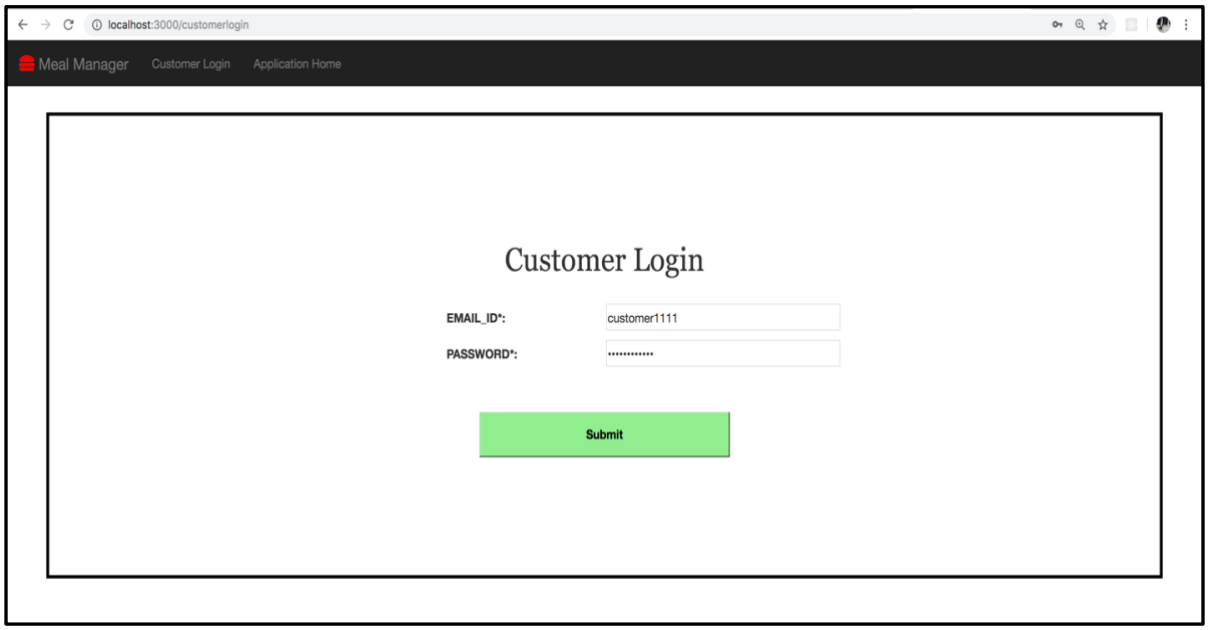


7.1.2. Customer

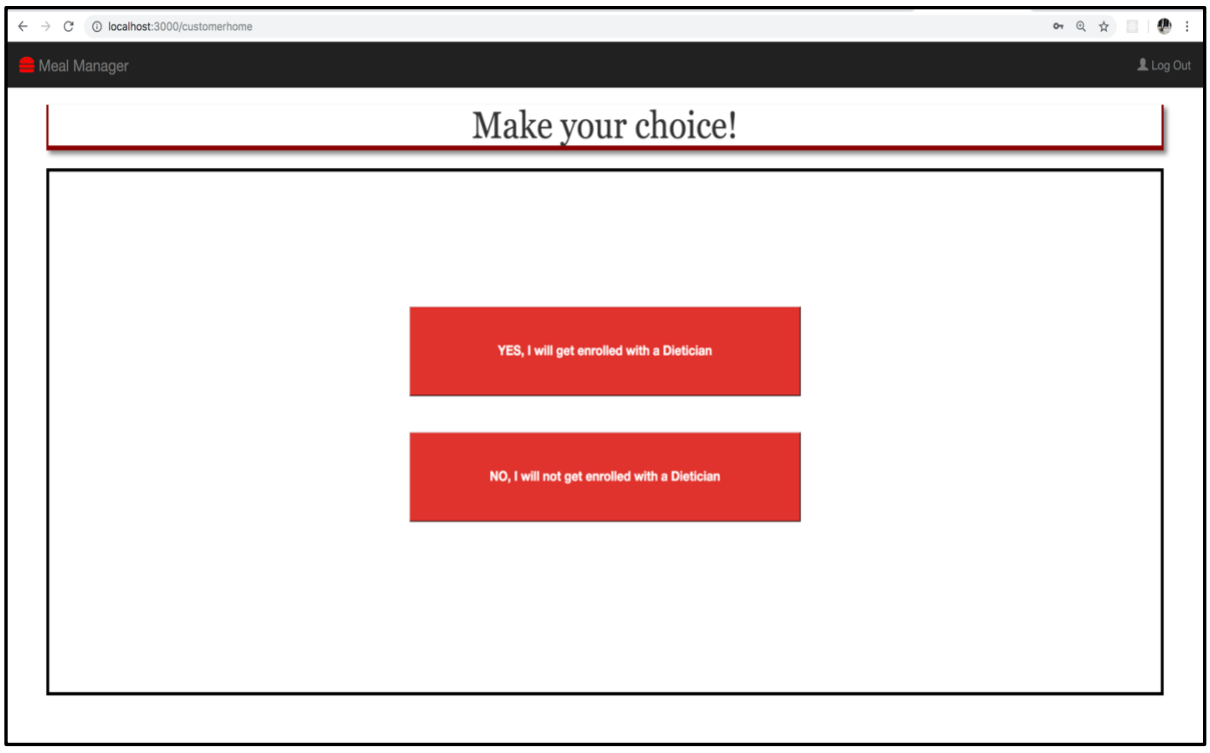
1. Customer registration page: page loaded when ‘Customer Registration’. Button is clicked in application homepage. Details entered after page load.



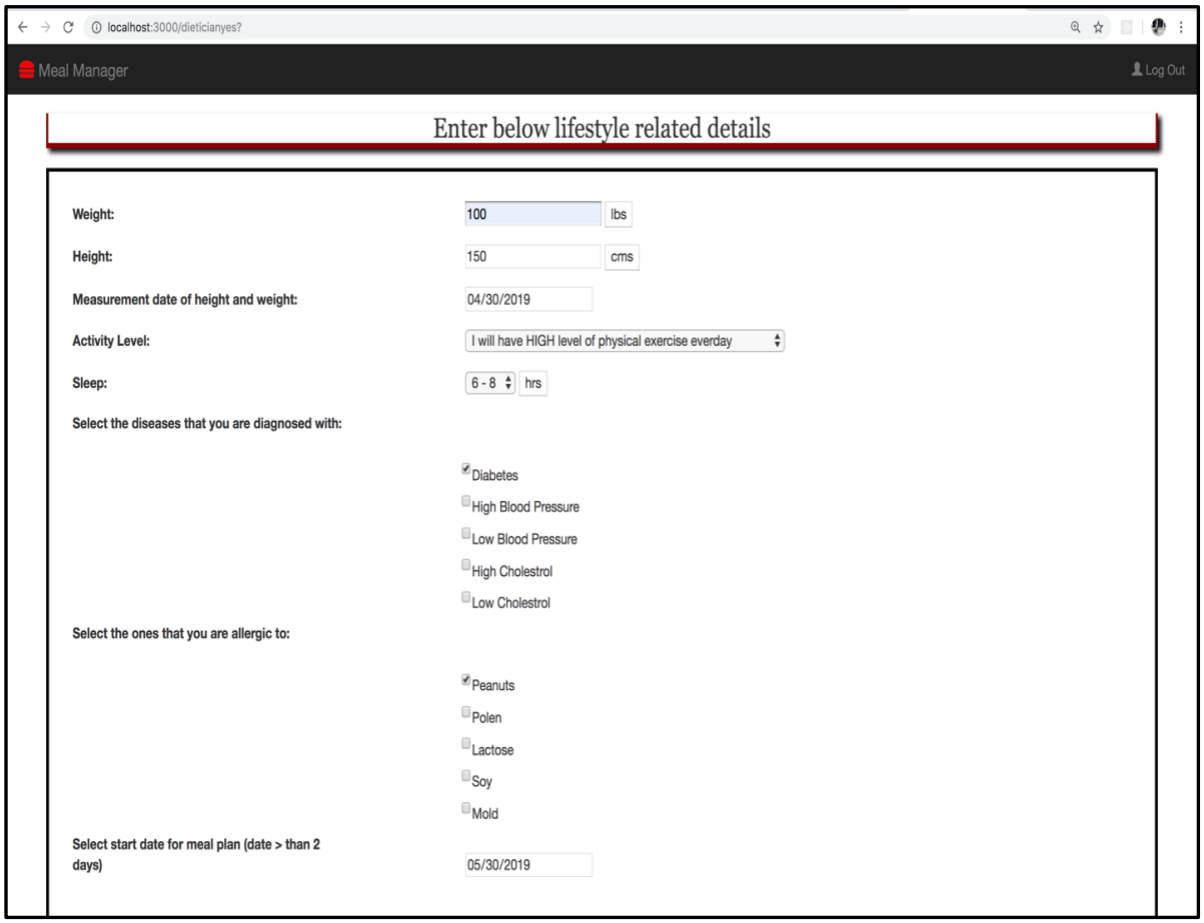
2. Customer login page: page displayed after successful registration of customer. Details entered after page load.

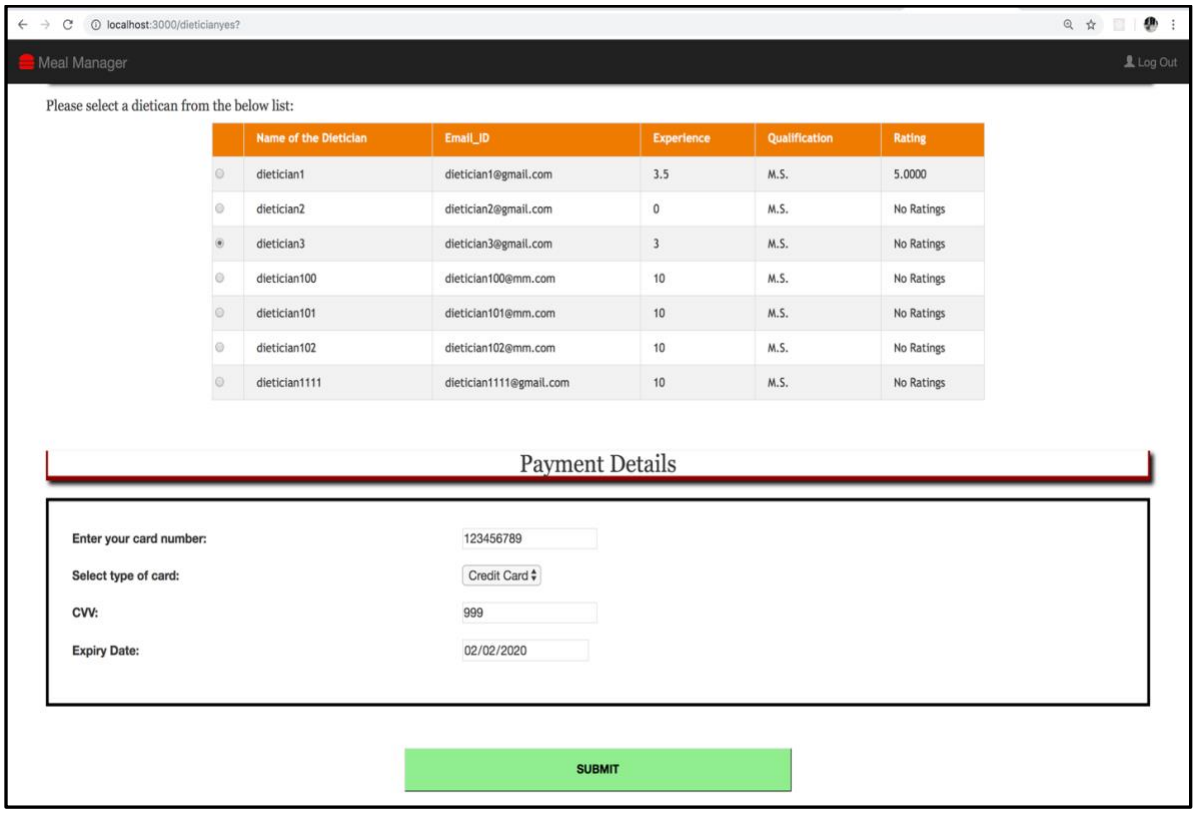


3. Customer home page: page displayed when customer credentials are verified and logged in successfully

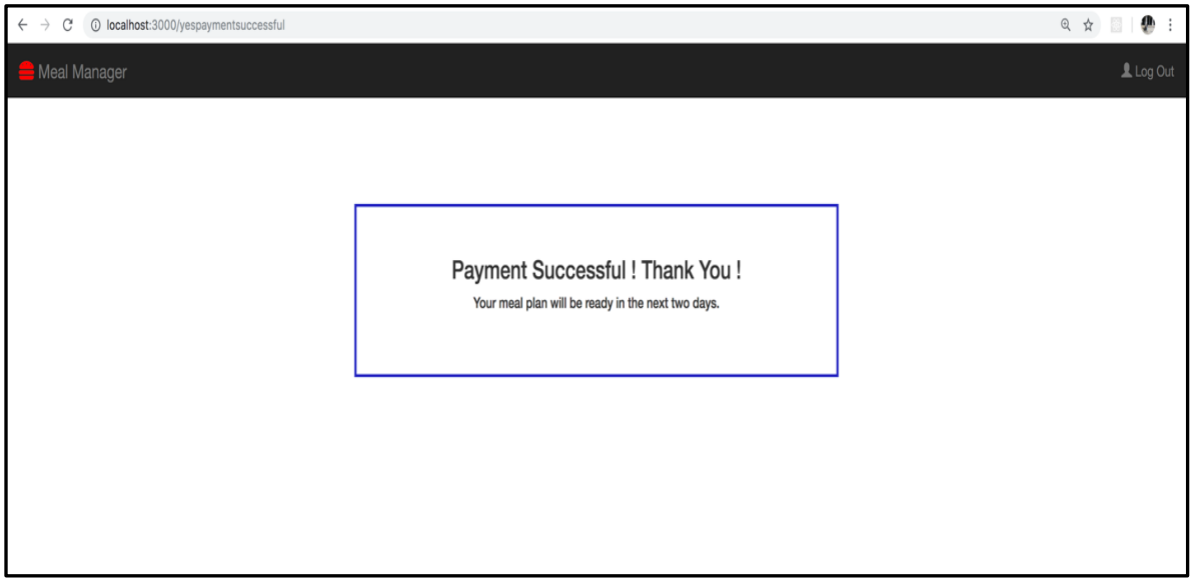


4. Dieticianyes page: page displayed when ‘Yes, I will get enrolled with a dietician’ button is clicked. Details entered after page load.



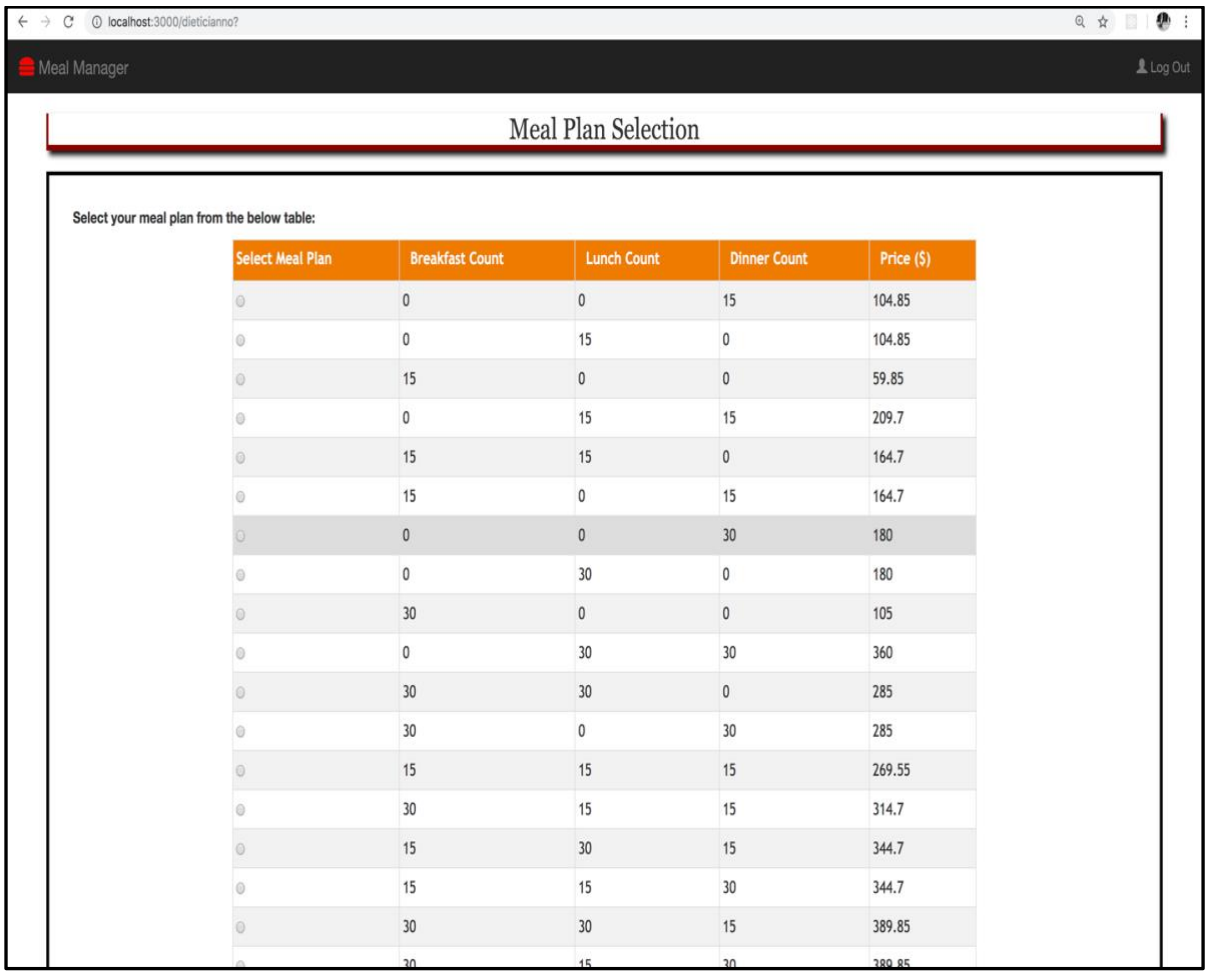


5. Yespayementsuccessful page: page displayed when details entered by customer is successfully stored in database

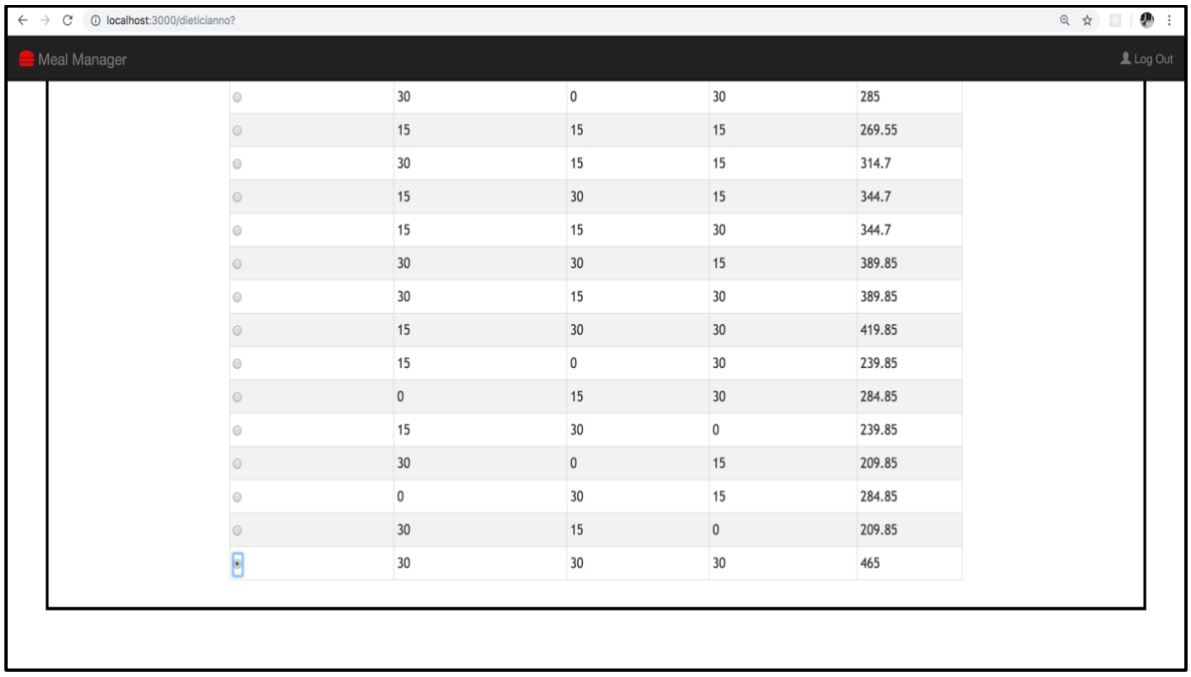


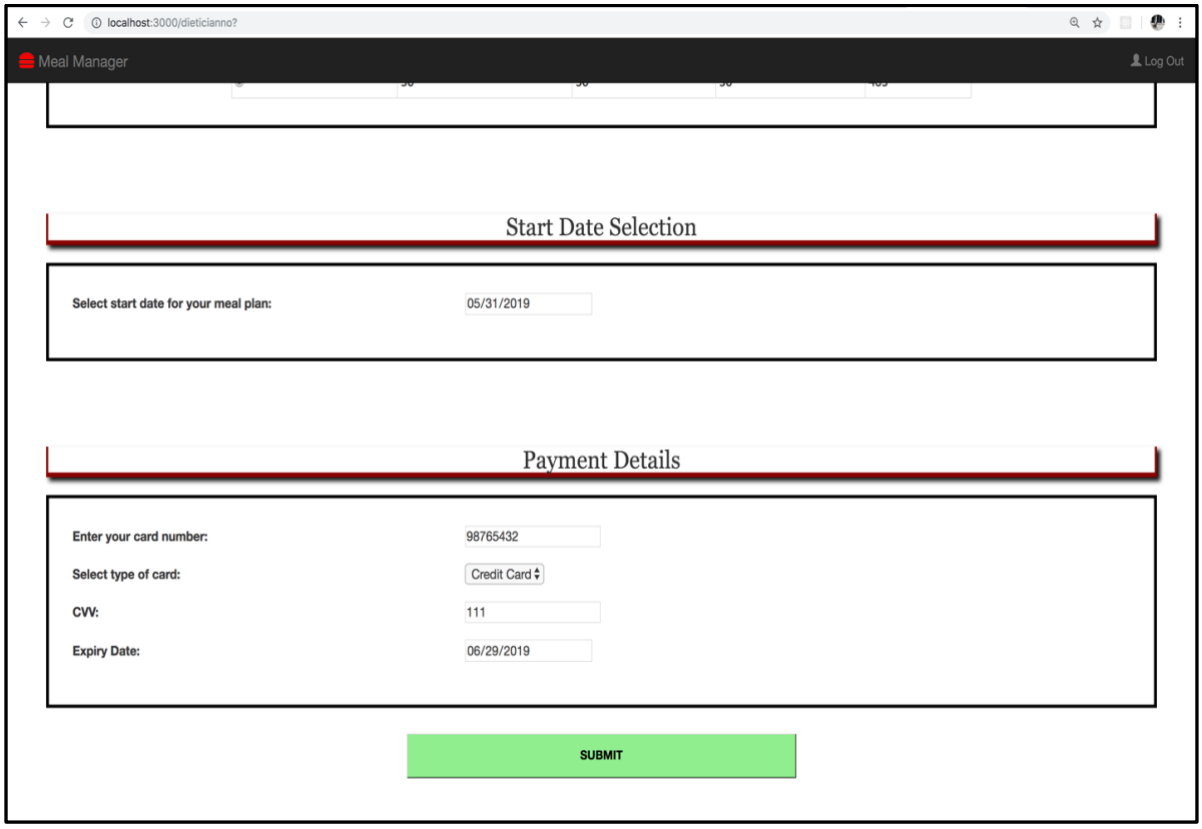
6. After registration of new customer customer2222:

Dieticianno page: page displayed when 'NO, I will not get enrolled with a dietician' button is clicked. Details entered after page load.

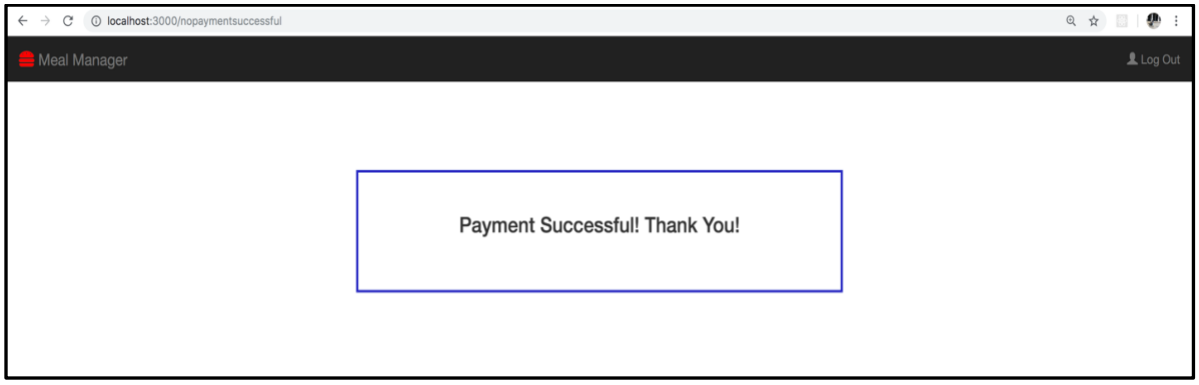


Select Meal Plan	Breakfast Count	Lunch Count	Dinner Count	Price ($)
	0	0	15	104.85
	0	15	0	104.85
	15	0	0	59.85
	0	15	15	209.7
	15	15	0	164.7
	15	0	15	164.7
	0	0	30	180
	0	30	0	180
	30	0	0	105
	0	30	30	360
	30	30	0	285
	30	0	30	285
	15	15	15	269.55
	30	15	15	314.7
	15	30	15	344.7
	15	15	30	344.7
	30	30	15	389.85
	30	15	30	389.85


localhost:3000/dieticianno?
Meal Manager
Log Out
30 0 30 285
15 15 15 269.55
30 15 15 314.7
15 30 15 344.7
15 15 30 344.7
30 30 15 389.85
30 15 30 389.85
15 30 30 419.85
15 0 30 239.85
0 15 30 284.85
15 30 0 239.85
30 0 15 209.85
0 30 15 284.85
30 15 0 209.85
30 30 30 465

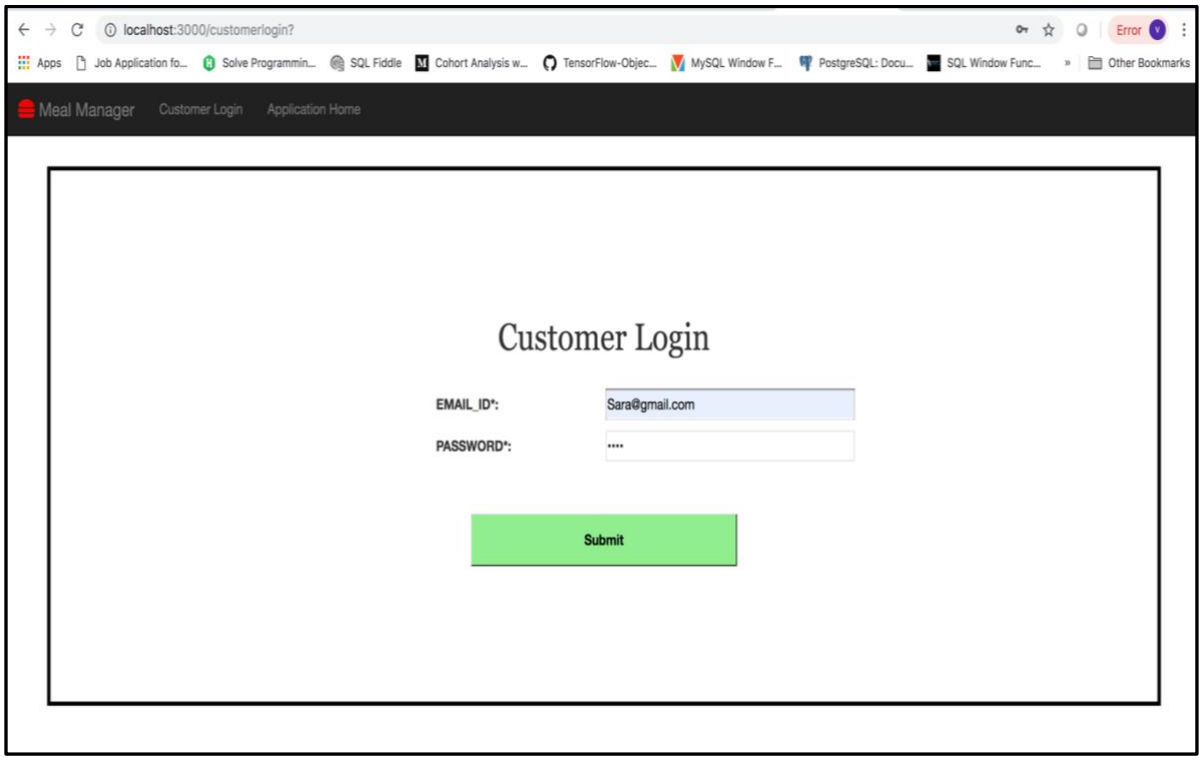

localhost:3000/dieticianno?
Meal Manager
Log Out
Start Date Selection
Select start date for your meal plan:
05/31/2019
Payment Details
Enter your card number:
98765432
Select type of card:
Credit Card
CVV:
111
Expiry Date:
06/29/2019
SUBMIT

7. Nopaymentsuccessful page: page displayed when details entered by customer is successfully stored in database

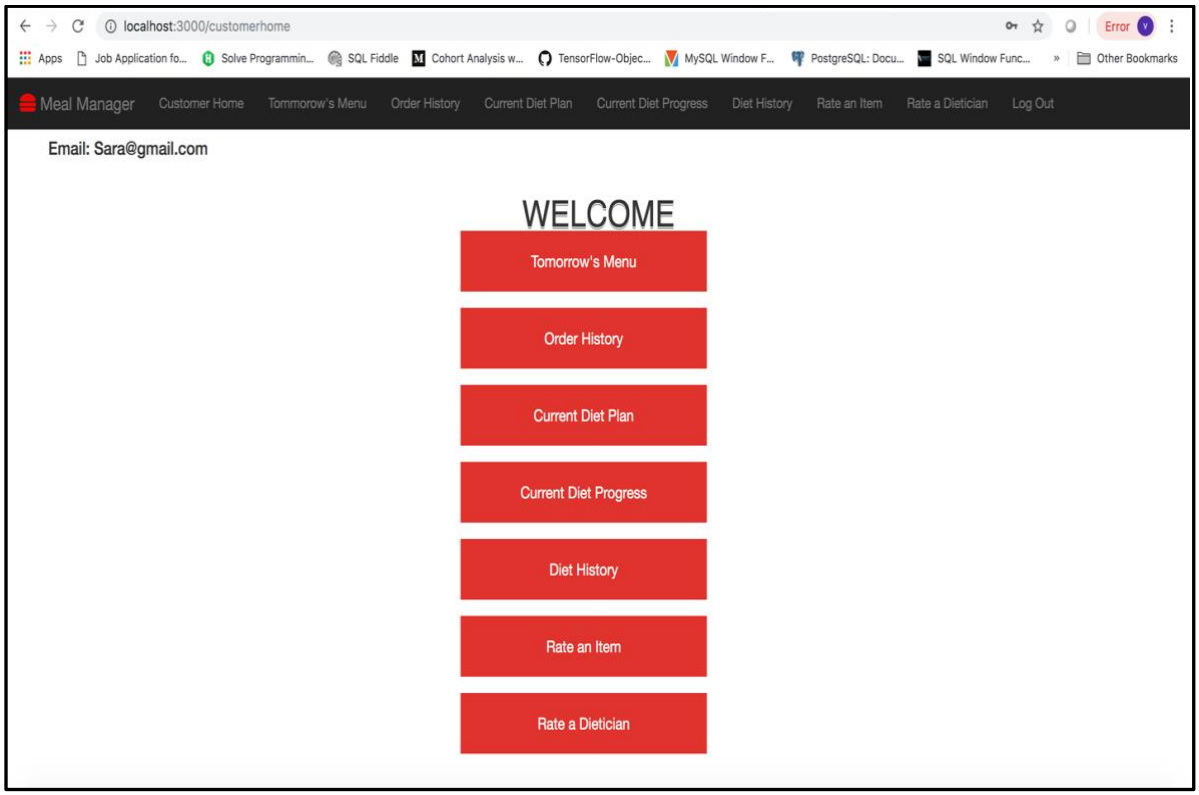


8. Customer Login Page: Customer logs in with email-id Sara@gmail.com and password "Sara".

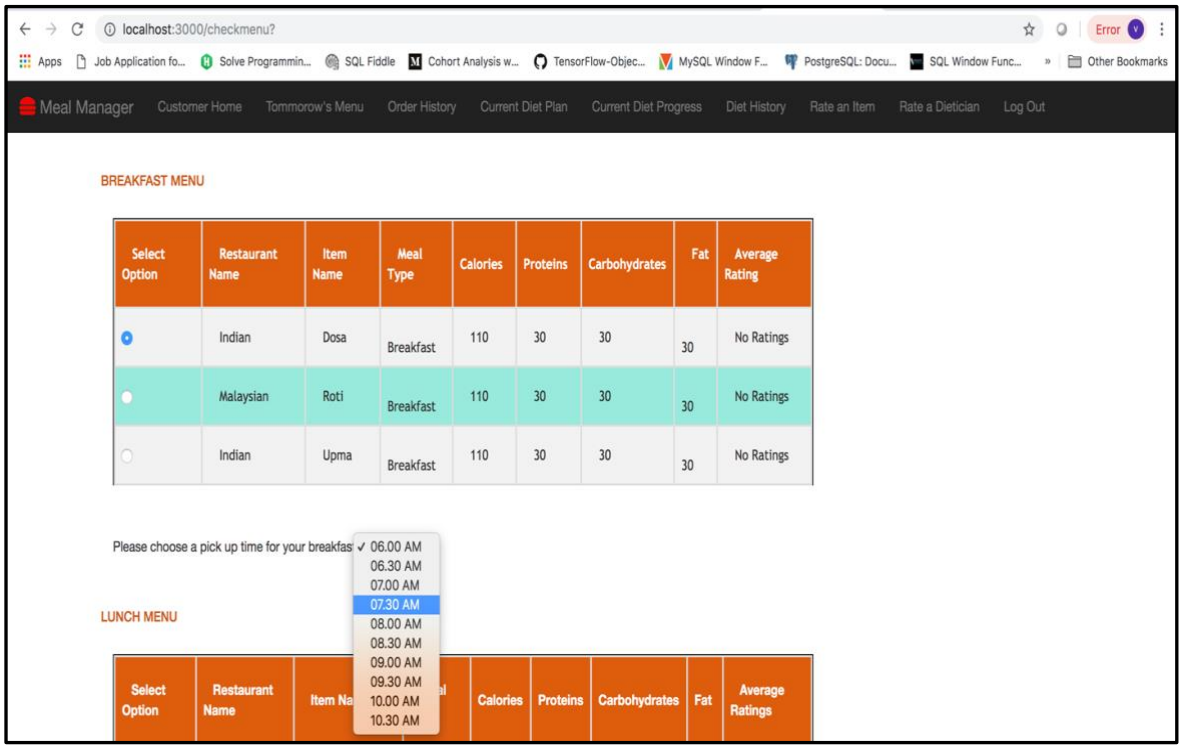
Customer Sara is currently enrolled in a regular meal plan but has two diet plans in her past.

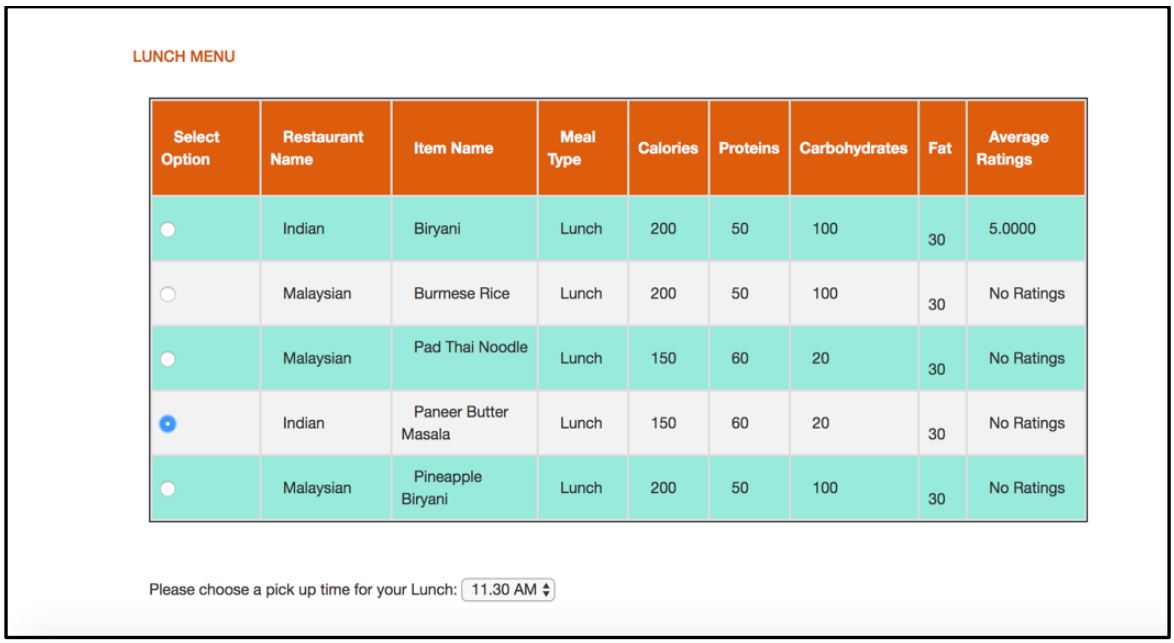


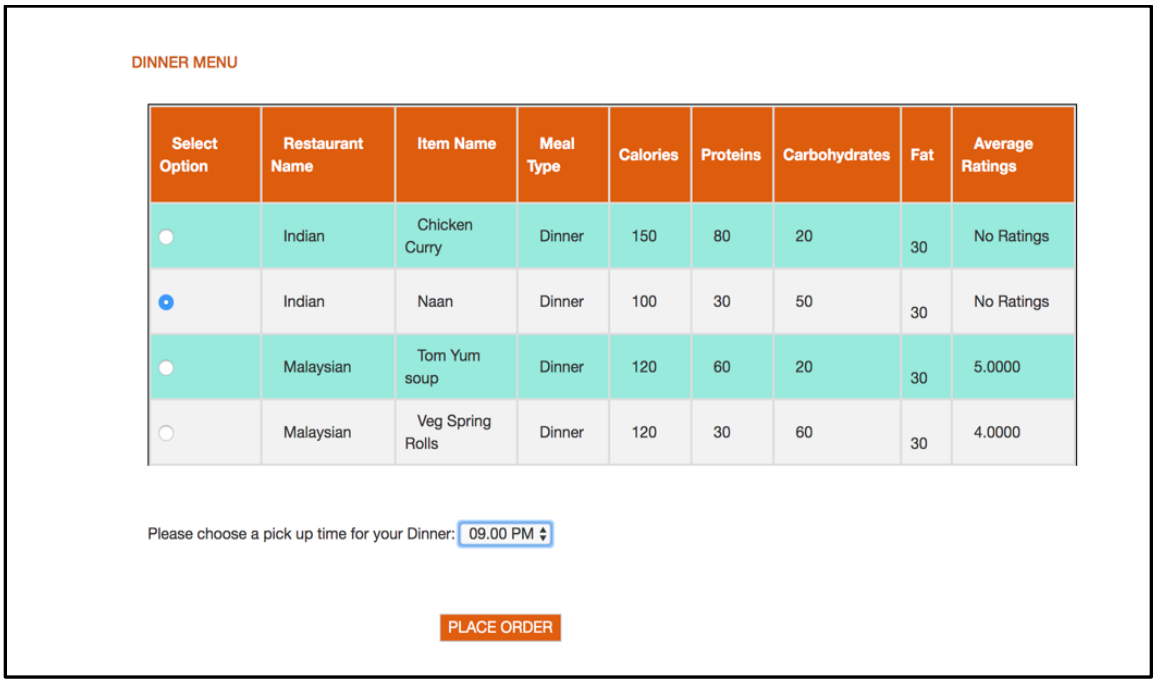
9. Customer Home Page after Sara logs in successfully.



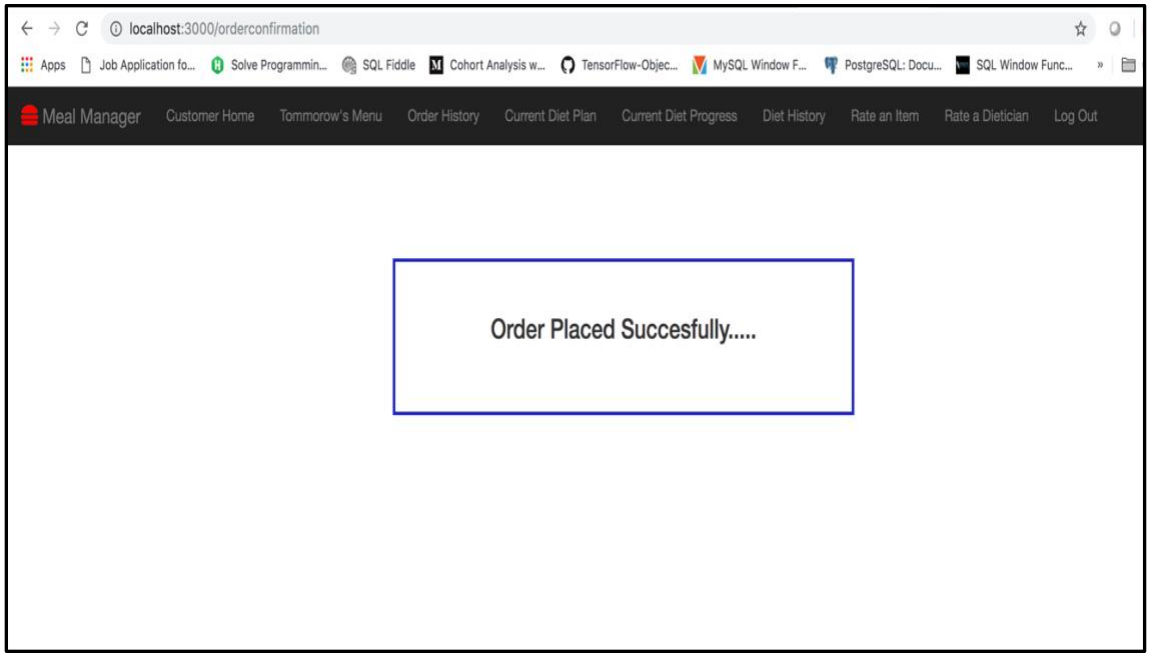
10. Click on Tomorrow's Menu button and you will get the below page with the personalized menu for next day. You can select any item you want to order in each table along with the pick-up time slot from drop down and click on place order button. (similar to below three screenshots)



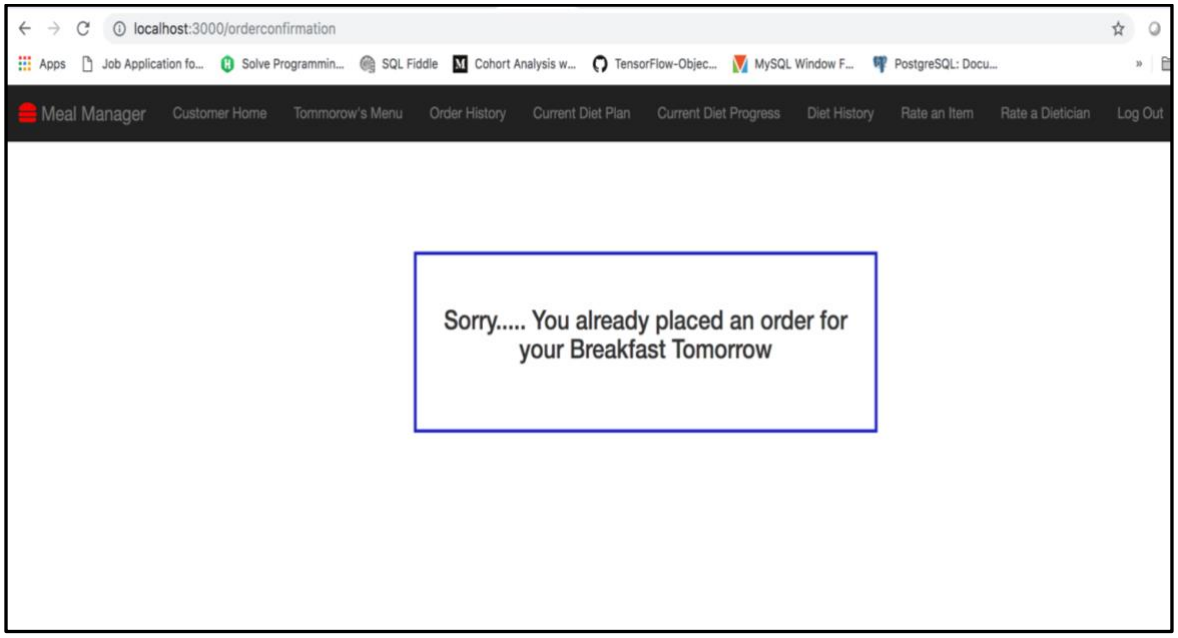




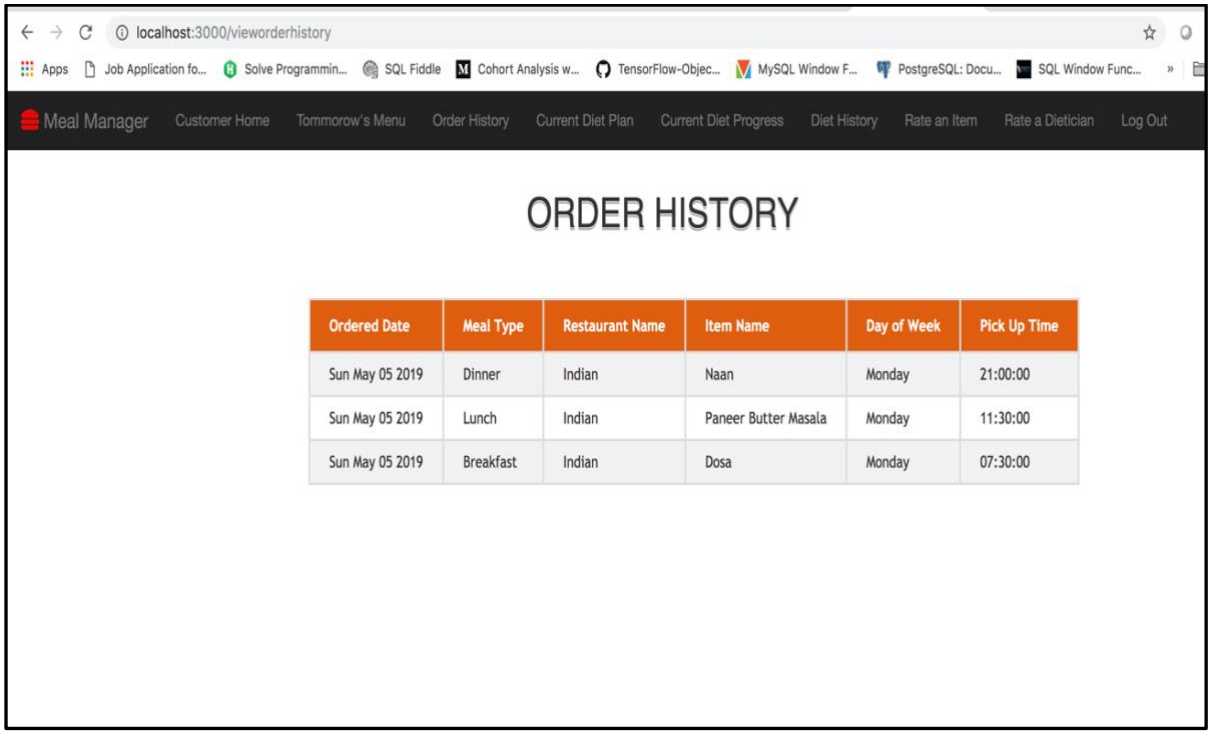
11. Once clicked on place order button, you get the message that the order is successfully placed.



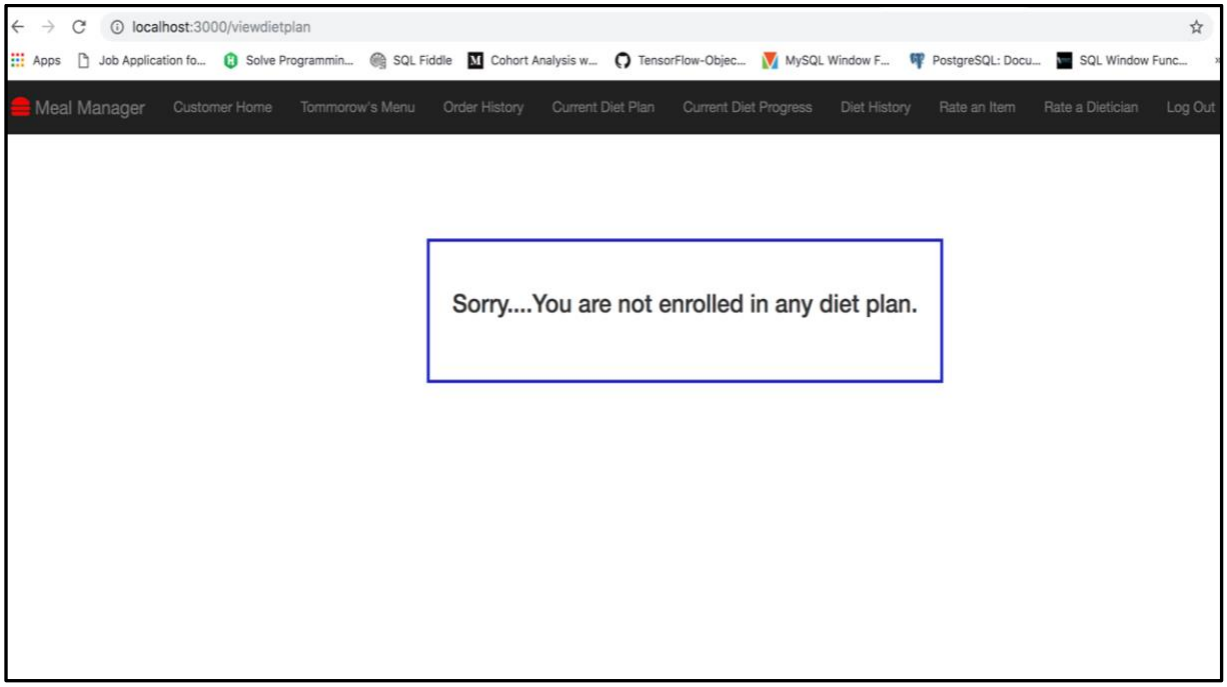
12. If you go to the Menu page again and try placing an order again for breakfast, the below message will be displayed because one customer can order only one item per meal type each day according to the business rule.



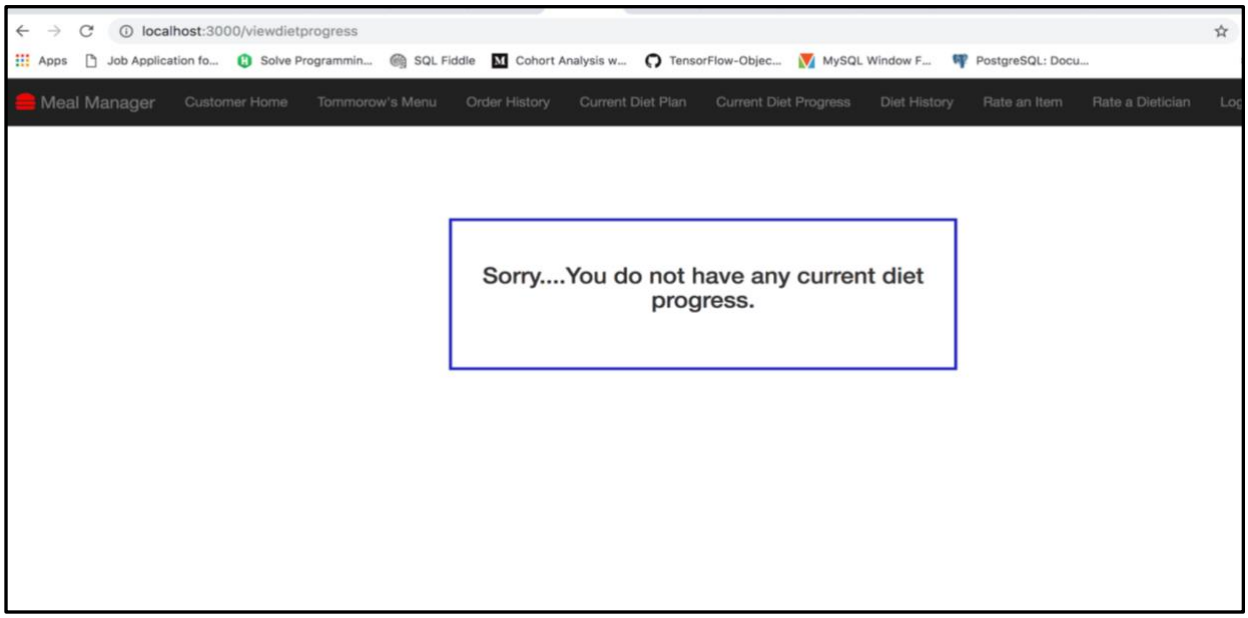
13. Go to home page and click on order history button. You will be able to get all the orders previously placed by Sara.



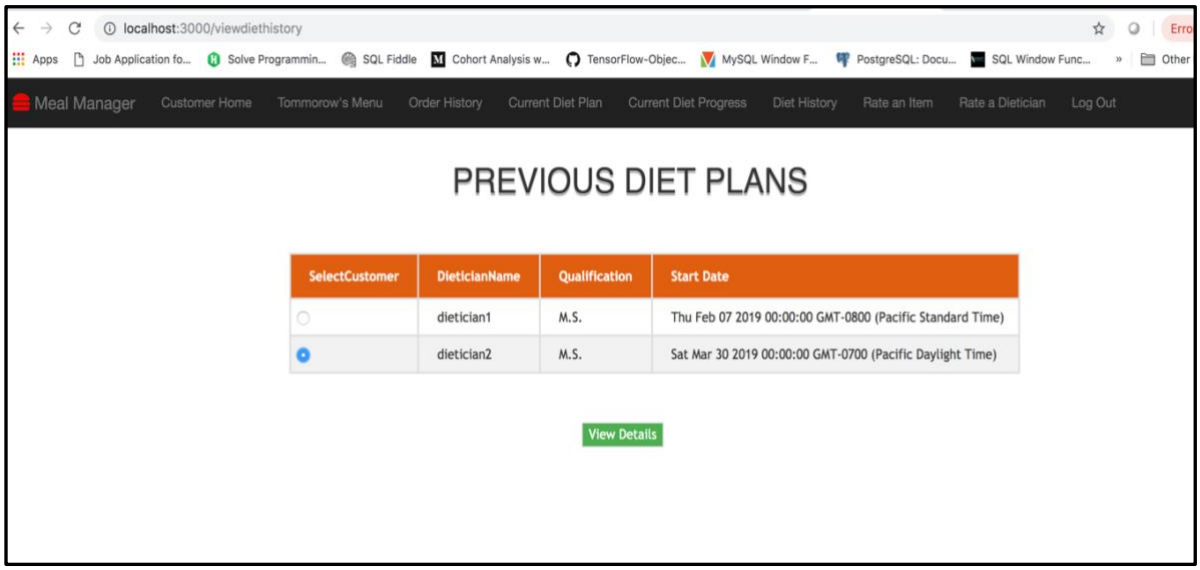
14. If you go to customer home page and click on current diet plan button, you get the below message as Sara is not enrolled with any dietician currently.



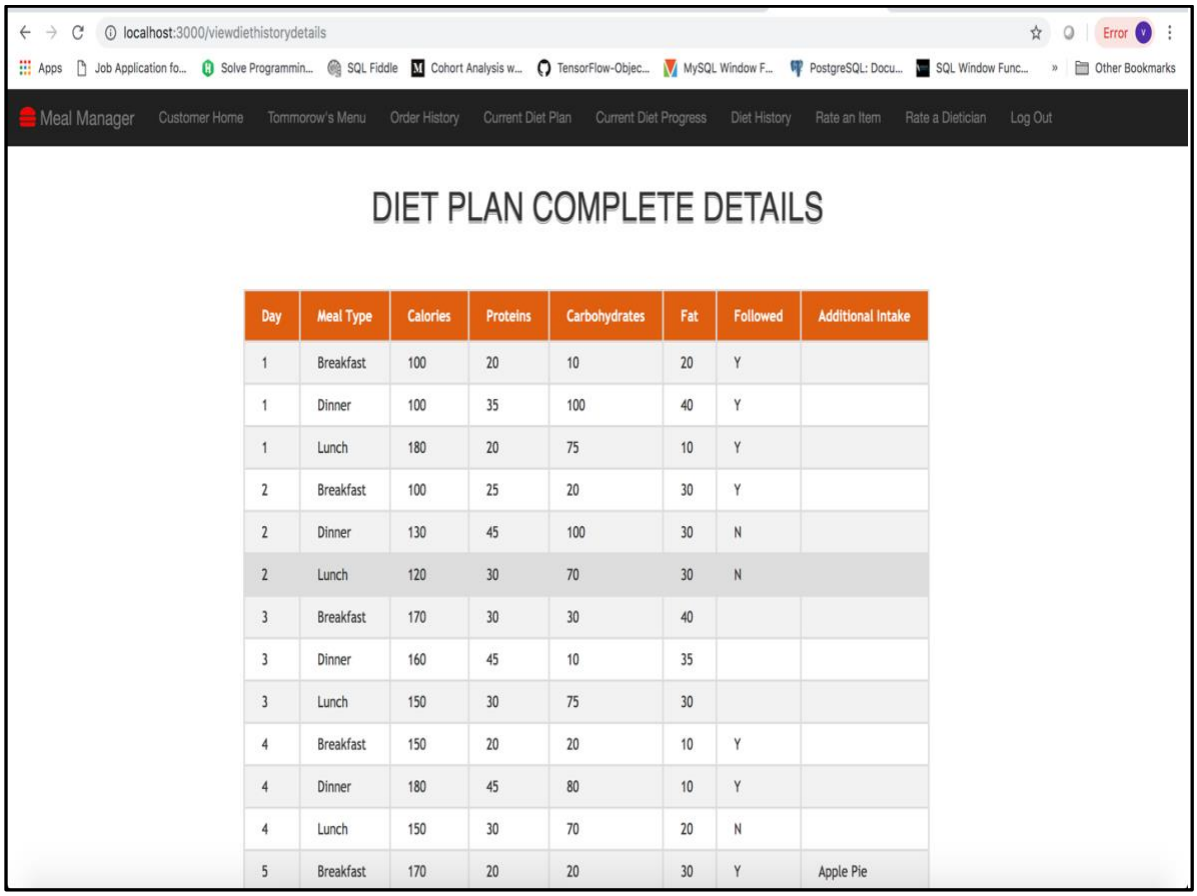
15. If you go to customer home page and click on current diet progress button, you get the below message as Sara is not enrolled with any dietician currently.



16. If you go to customer home page and click on diet history button, you get the list of previous diet plans which Sara has.

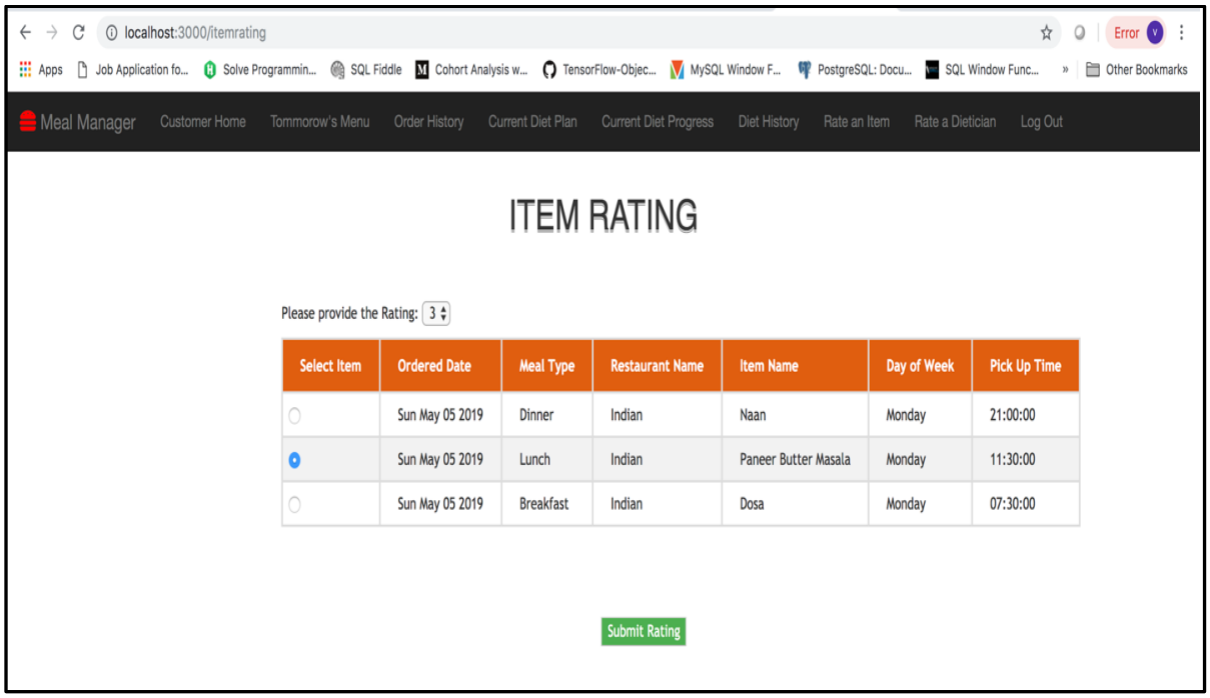


17. You can select one diet plan and click on view details button to get the complete information. (Shown in screenshot below)

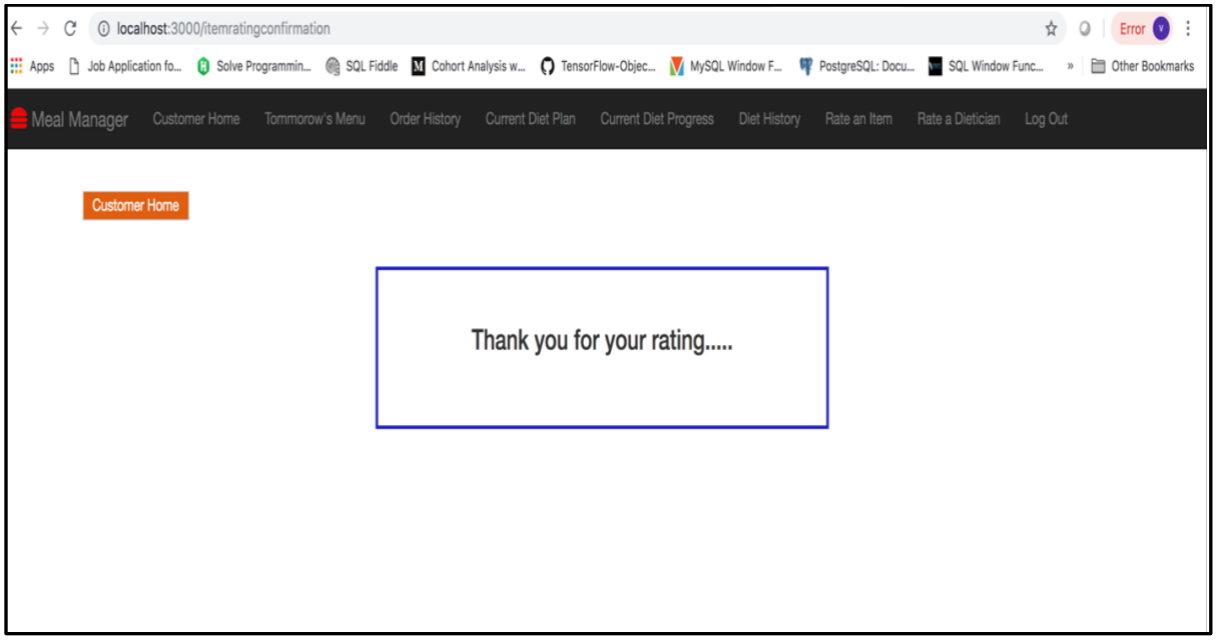


Day	Meal Type	Calories	Proteins	Carbohydrates	Fat	Followed	Additional Intake
1	Breakfast	100	20	10	20	Y	
1	Dinner	100	35	100	40	Y	
1	Lunch	180	20	75	10	Y	
2	Breakfast	100	25	20	30	Y	
2	Dinner	130	45	100	30	N	
2	Lunch	120	30	70	30	N	
3	Breakfast	170	30	30	40		
3	Dinner	160	45	10	35		
3	Lunch	150	30	75	30		
4	Breakfast	150	20	20	10	Y	
4	Dinner	180	45	80	10	Y	
4	Lunch	150	30	70	20	N	
5	Breakfast	170	20	20	30	Y	Apple Pie

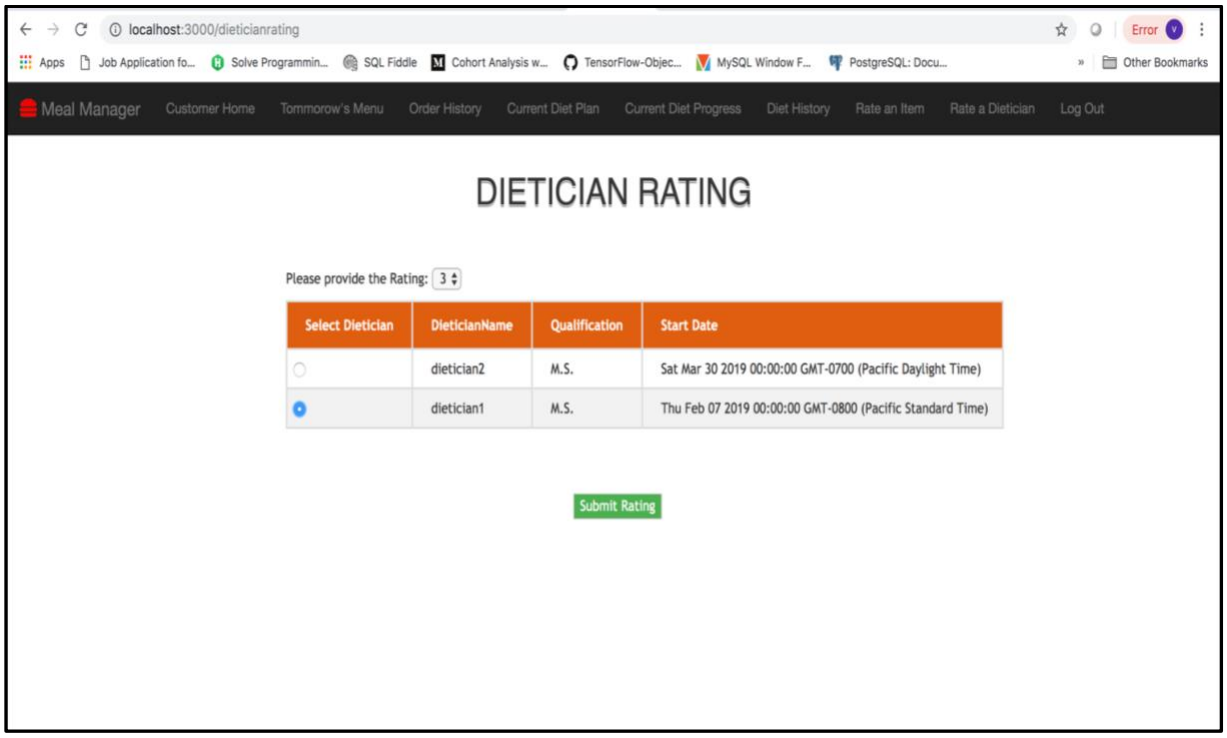
18. If you go to customer home page and click on Rate an item button, you get the list of items which Sara has previously ordered. You can select one of the items from this list, then a rating from the dropdown and click on submit rating button.



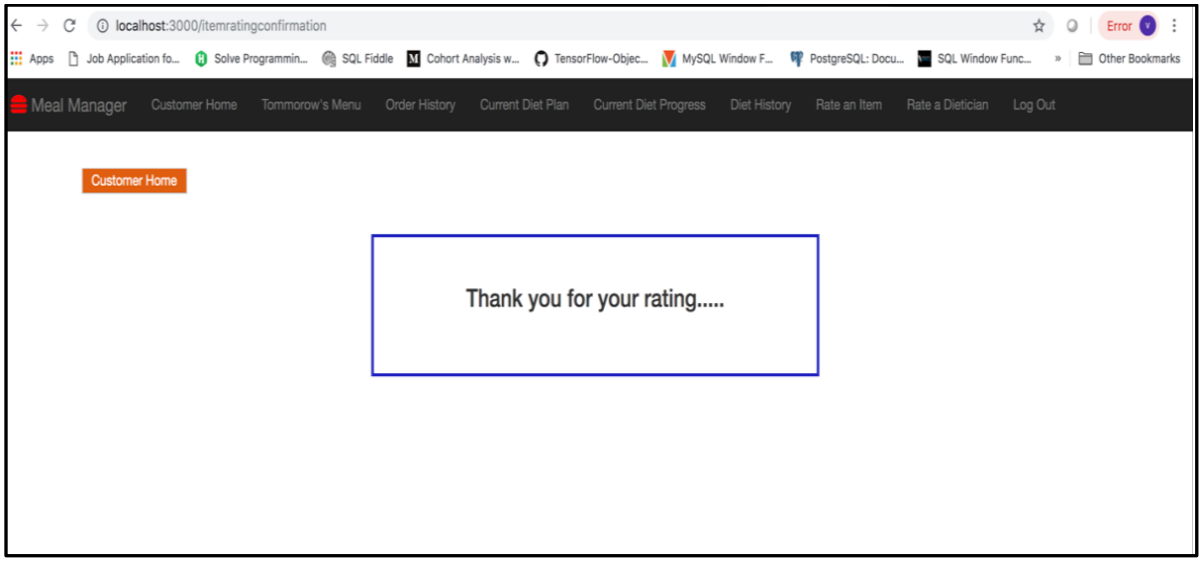
19. You will get the below message indicating success for the operation.



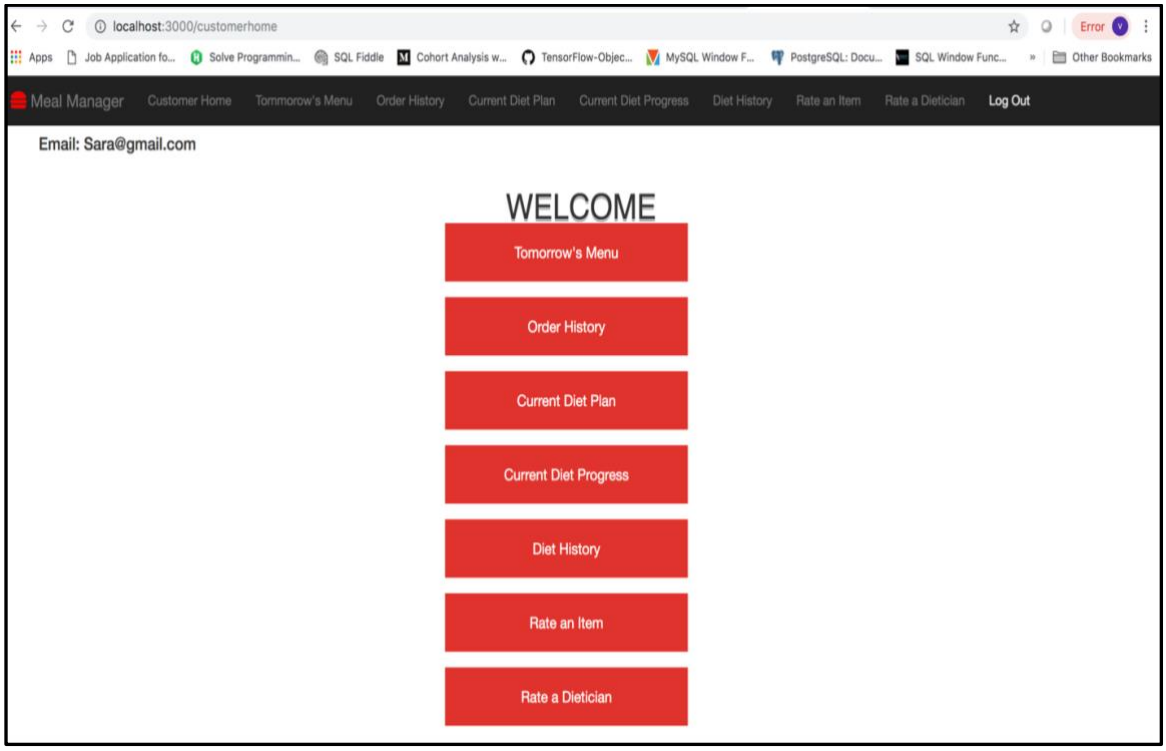
20. If you go to customer home page and click on Rate a Dietician button, you get the list of dieticians to whom Sara has previously enrolled. You can select one of the dieticians from this list, then a rating from the dropdown and click on submit rating button.

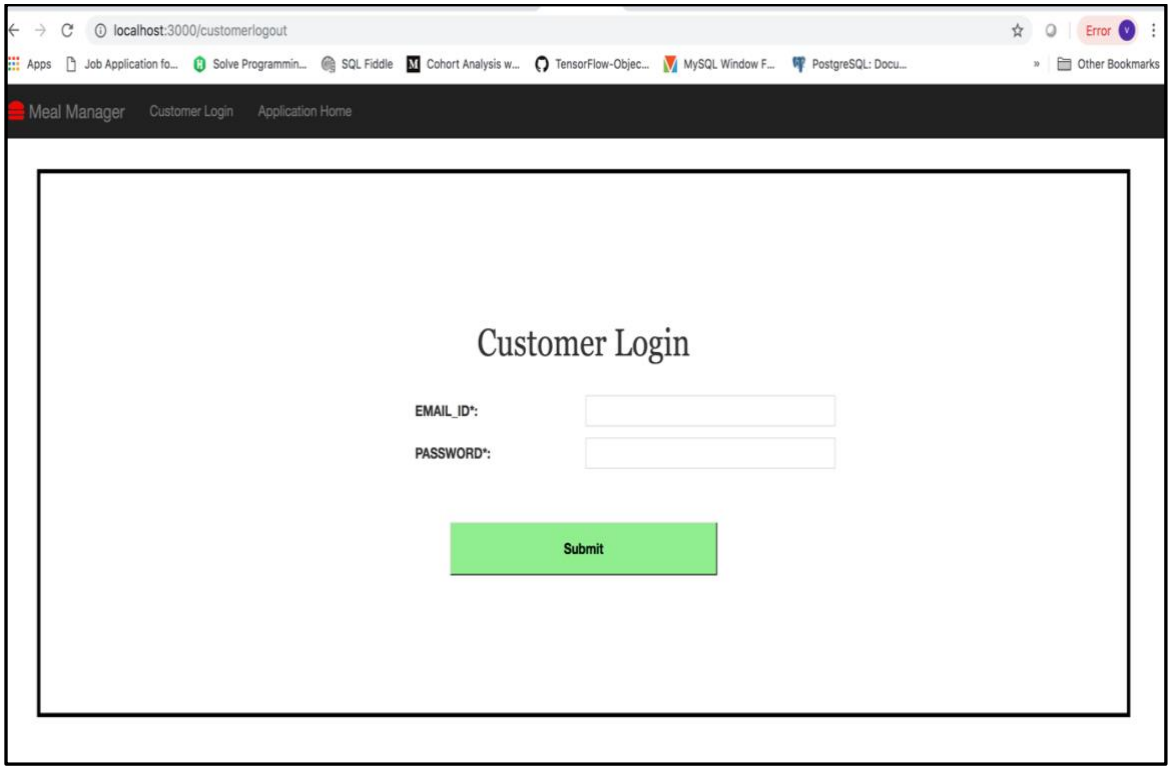


21. You will get the below message indicating success for the operation.

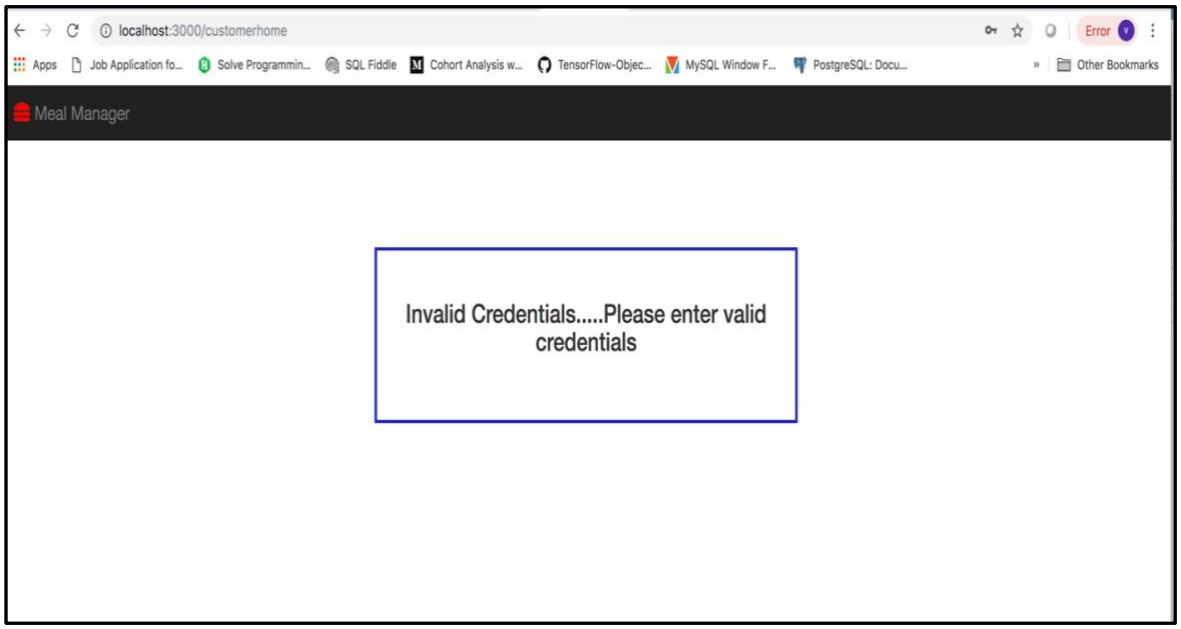


22. If you go to customer home page and click on Log out button you will be logged out and redirected to customer login page again.

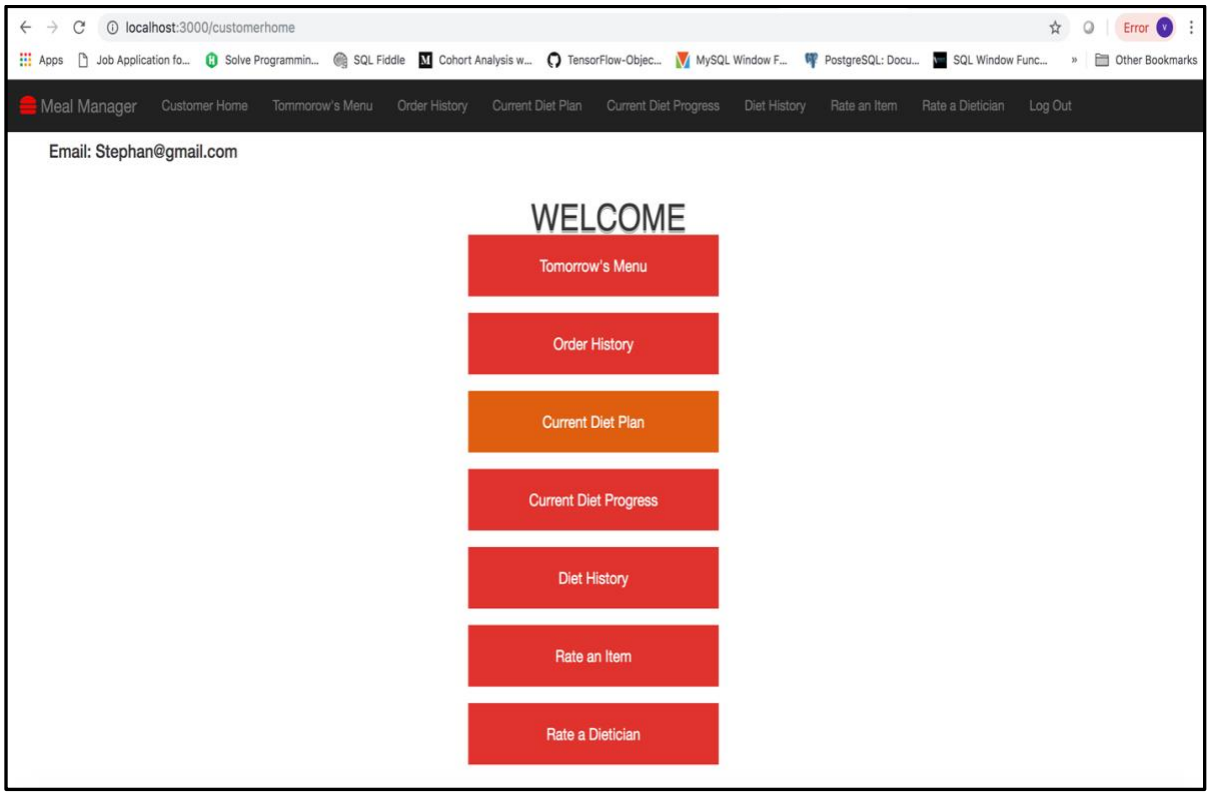




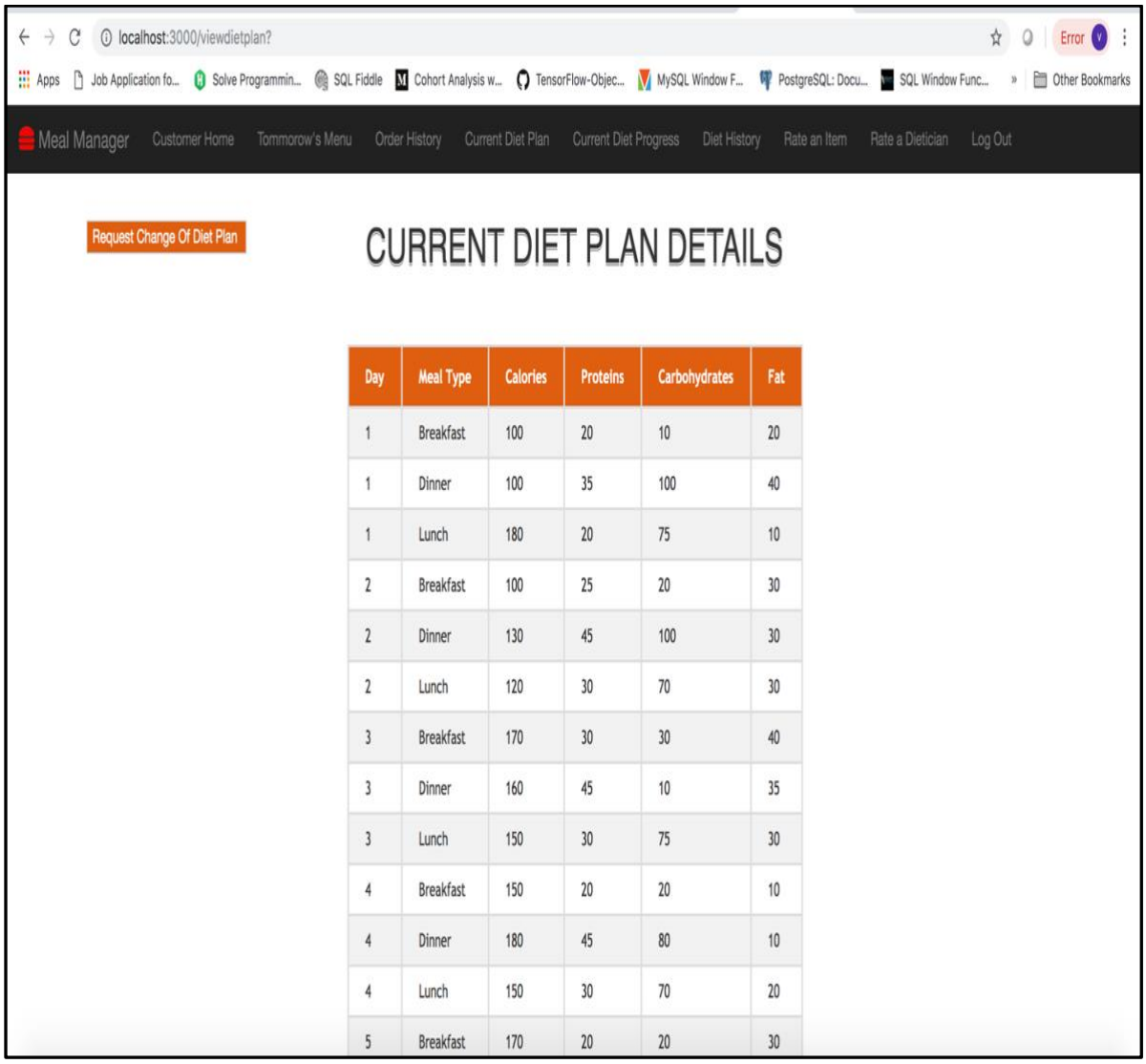
23. Now, if you enter email-id as Sara@gmail.com and password as “sara” (instead of “Sara”), you will get the below message as you entered wrong password.



24. If you go to customer login page and enter email-id as Stephan@gmail.com and password as “Stephan” you will enter the customer home page. Customer Stephan is a dietician enrolled person.

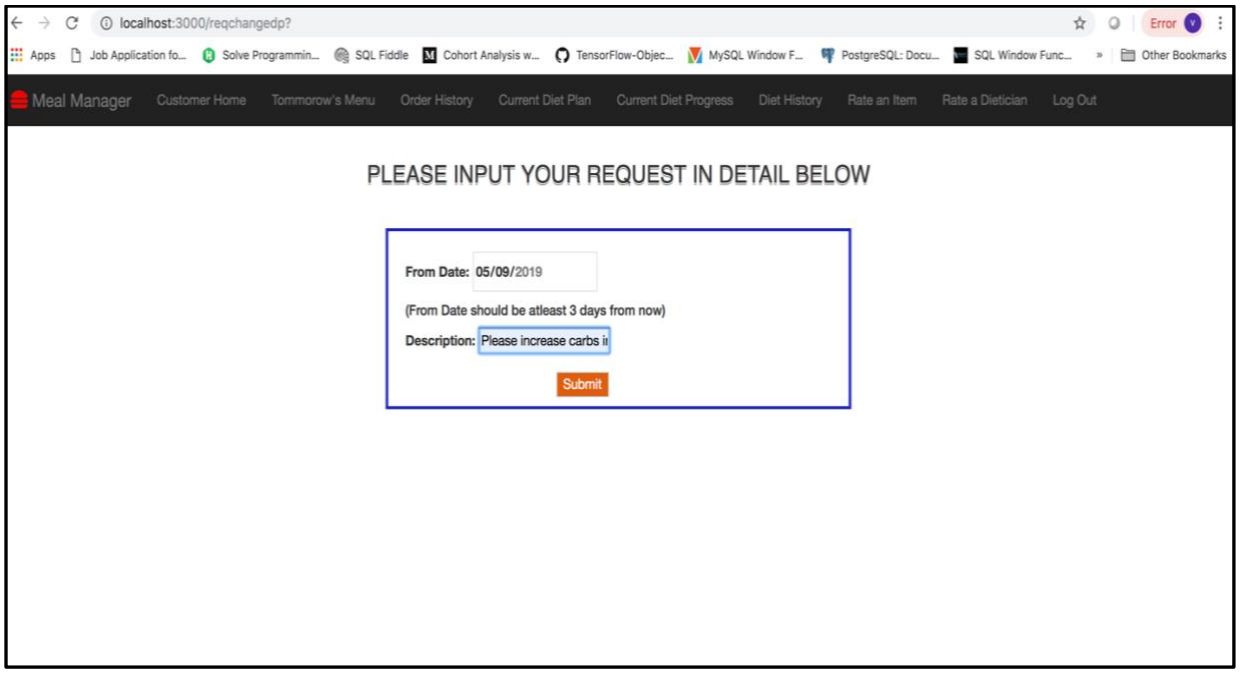


25. If you click on Current Diet Plan button, you will get the below page with the current diet plan details for Stephan. Now, if you want to request dietician to make some change in your current diet plan, you can click on Request Change of Diet Plan button in the top left corner of the page.

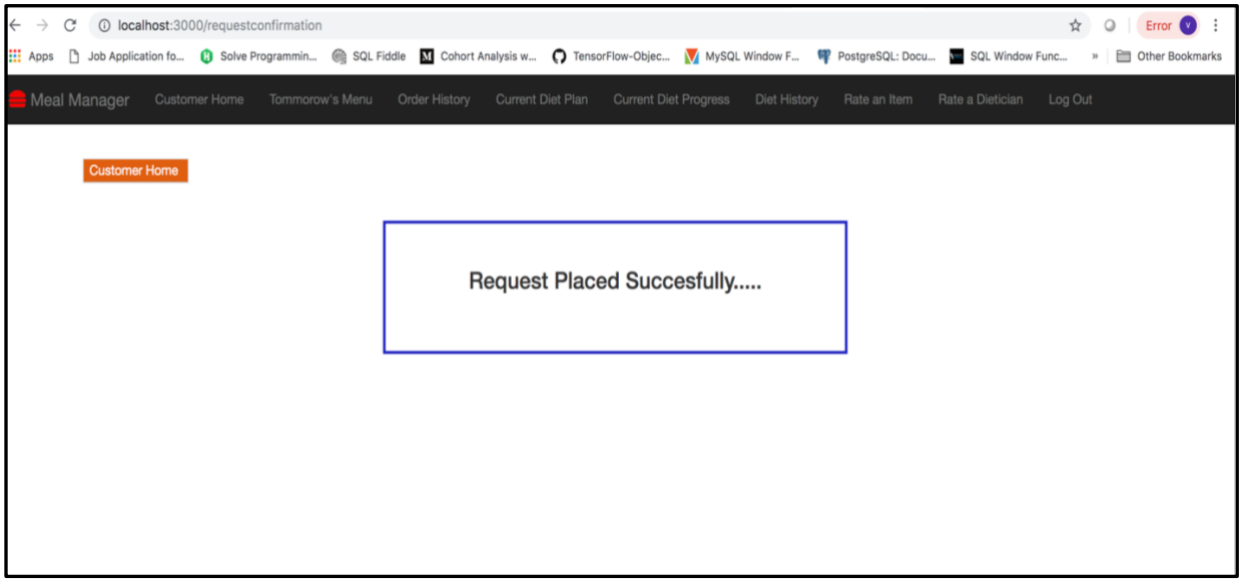


Day	Meal Type	Calories	Proteins	Carbohydrates	Fat
1	Breakfast	100	20	10	20
1	Dinner	100	35	100	40
1	Lunch	180	20	75	10
2	Breakfast	100	25	20	30
2	Dinner	130	45	100	30
2	Lunch	120	30	70	30
3	Breakfast	170	30	30	40
3	Dinner	160	45	10	35
3	Lunch	150	30	75	30
4	Breakfast	150	20	20	10
4	Dinner	180	45	80	10
4	Lunch	150	30	70	20
5	Breakfast	170	20	20	30

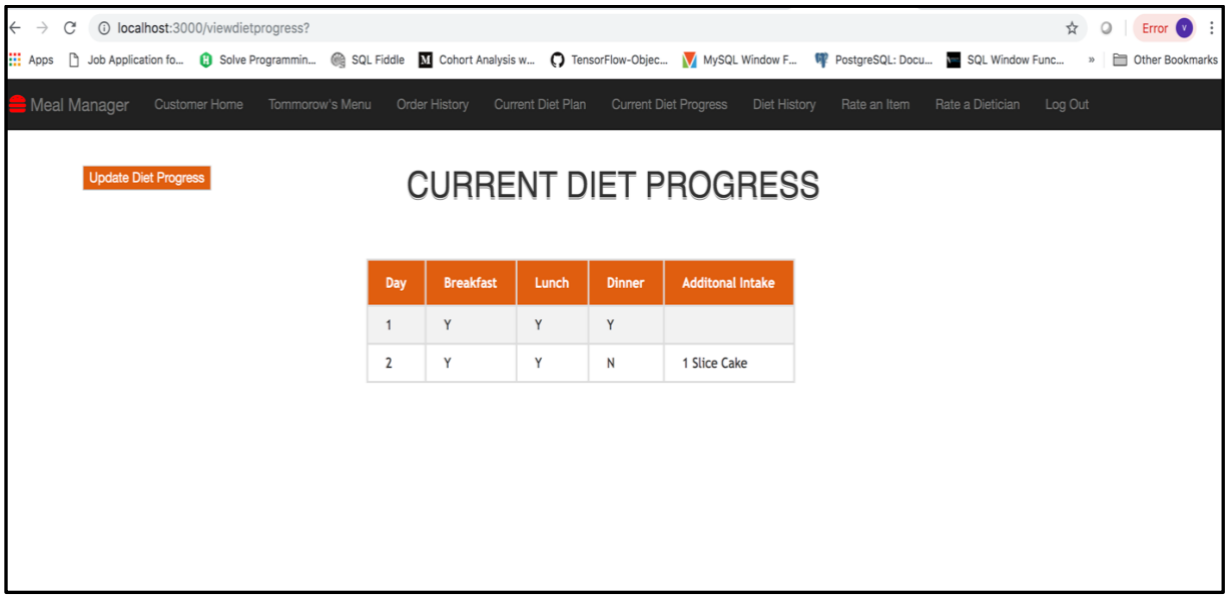
26. You will get the below form in which you have to enter the date from which you want to start your new diet plan and also you should provide some description on the change you want to have. Once you enter the date and description, you can click on submit button.



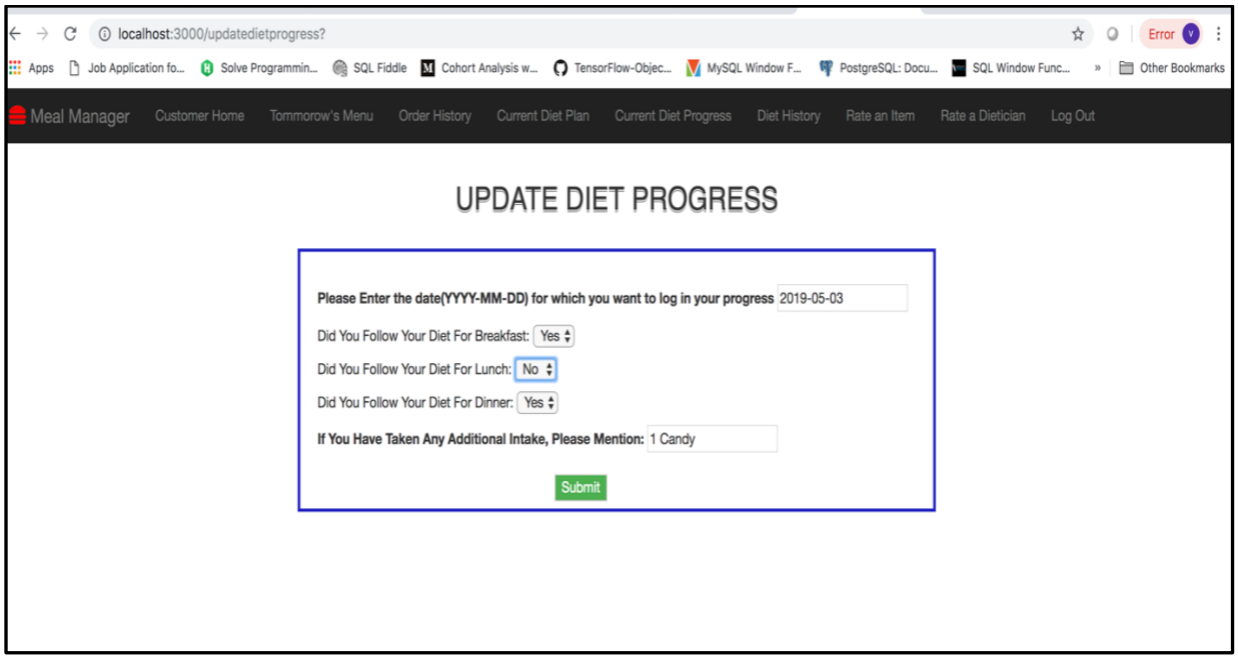
27. You will get the below message indicating that the request is placed successfully.



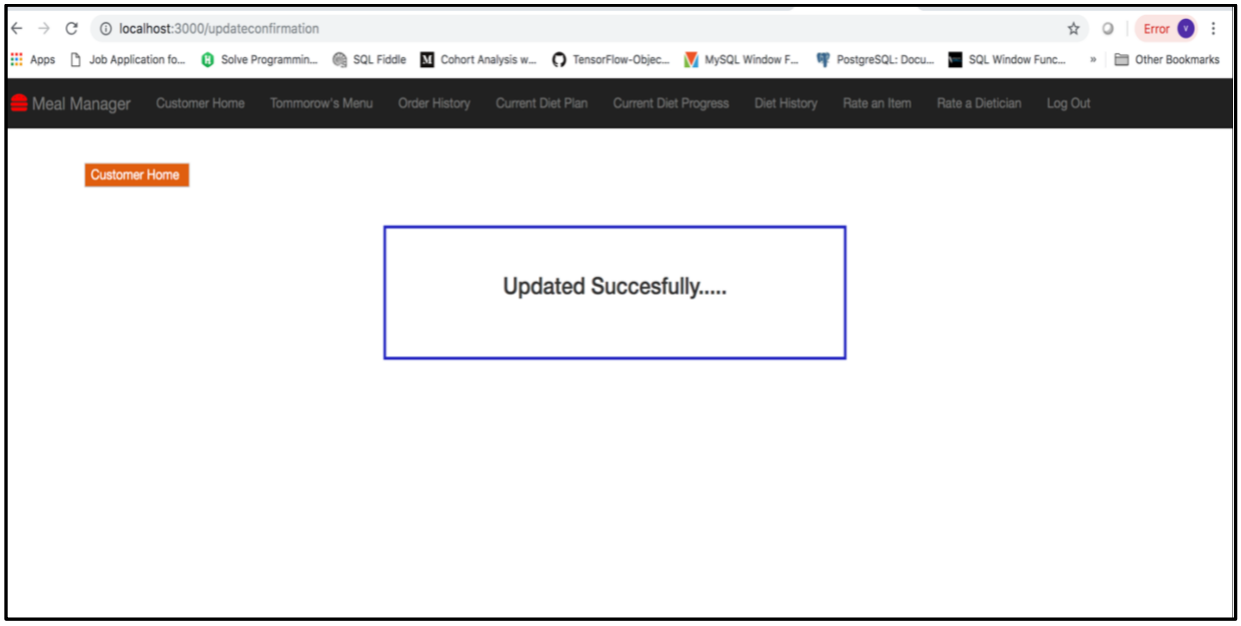
28. If you go to customer home page and click on current diet progress button, you will get a new page with the data Stephan has logged previously. If you want to add new diet progress, you can click on Update Diet Progress button on top left corner of the page.



29. You can enter the date for which you want to update your progress, then update if you have followed the diet for each meal type. Then click on submit button.



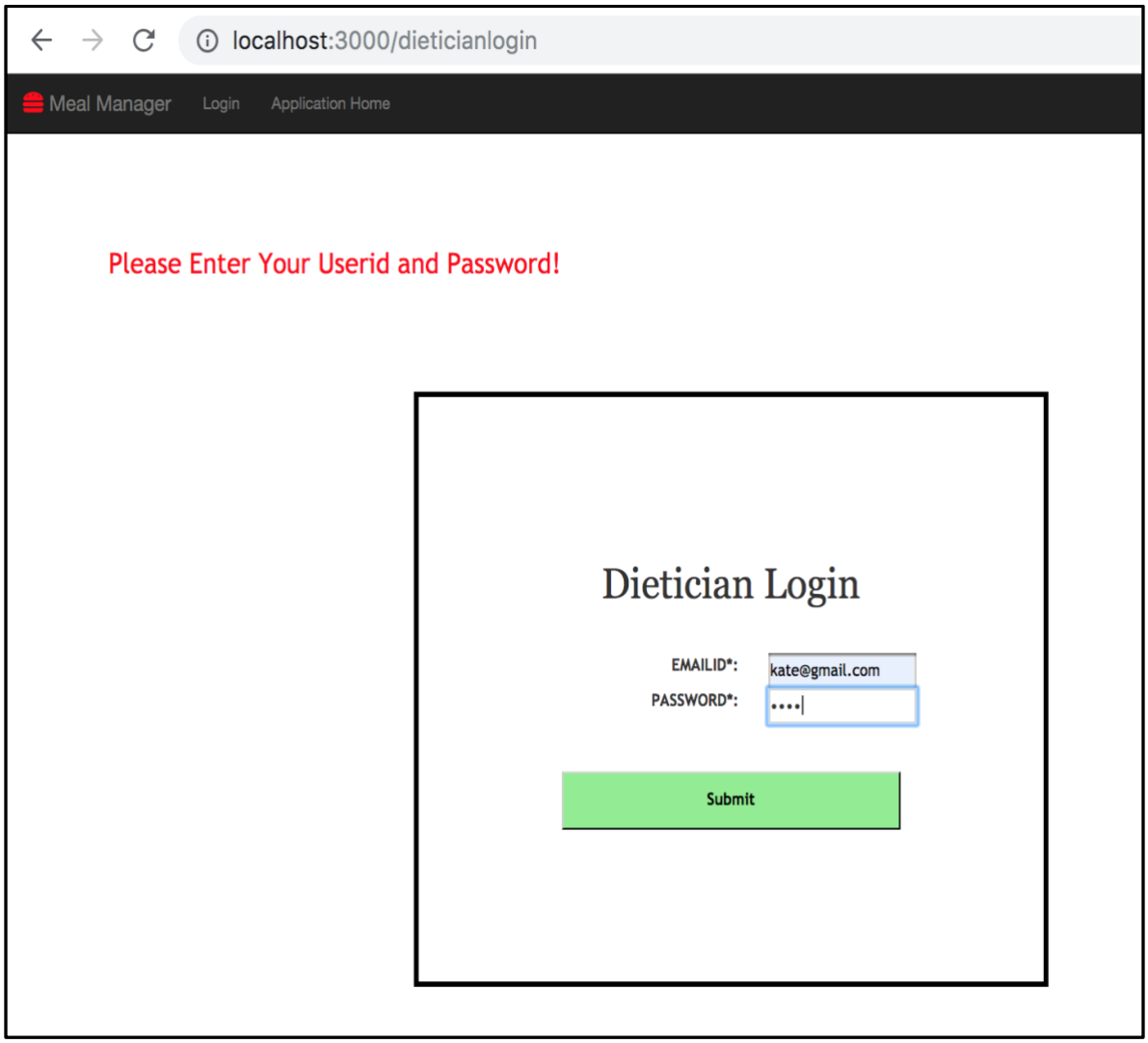
30. Once you click on submit button, you get the below message indicating that you update is successfully done.



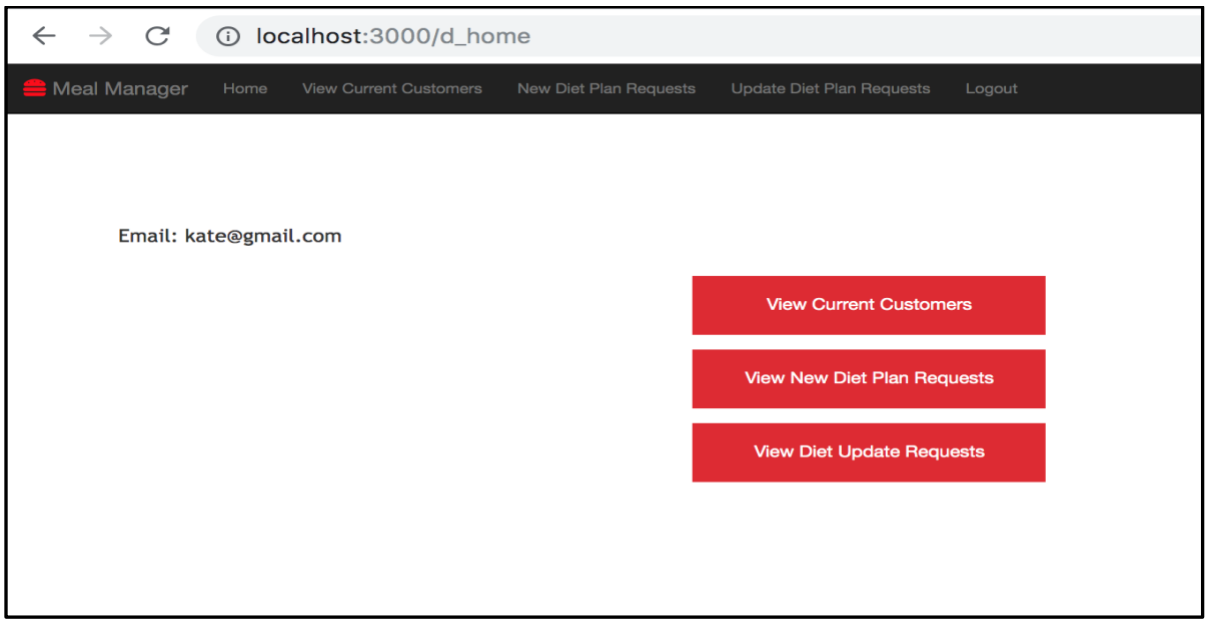
7.1.3. Dietician

1. Dietician Login Page:
   Login Page: http://localhost:3000/dieticianlogin

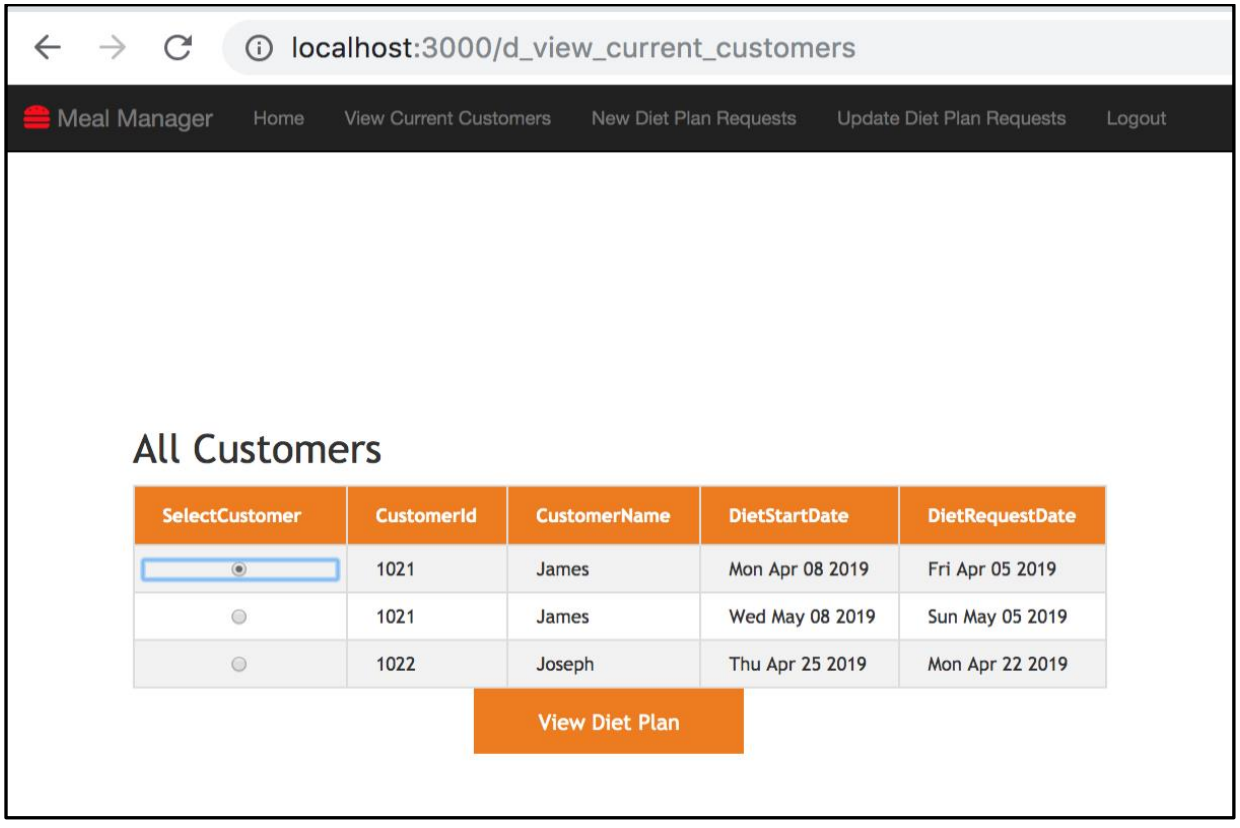
2. Dietician Login Page: Dietician logs in with email-id kate@gmail.com and password "Kate".



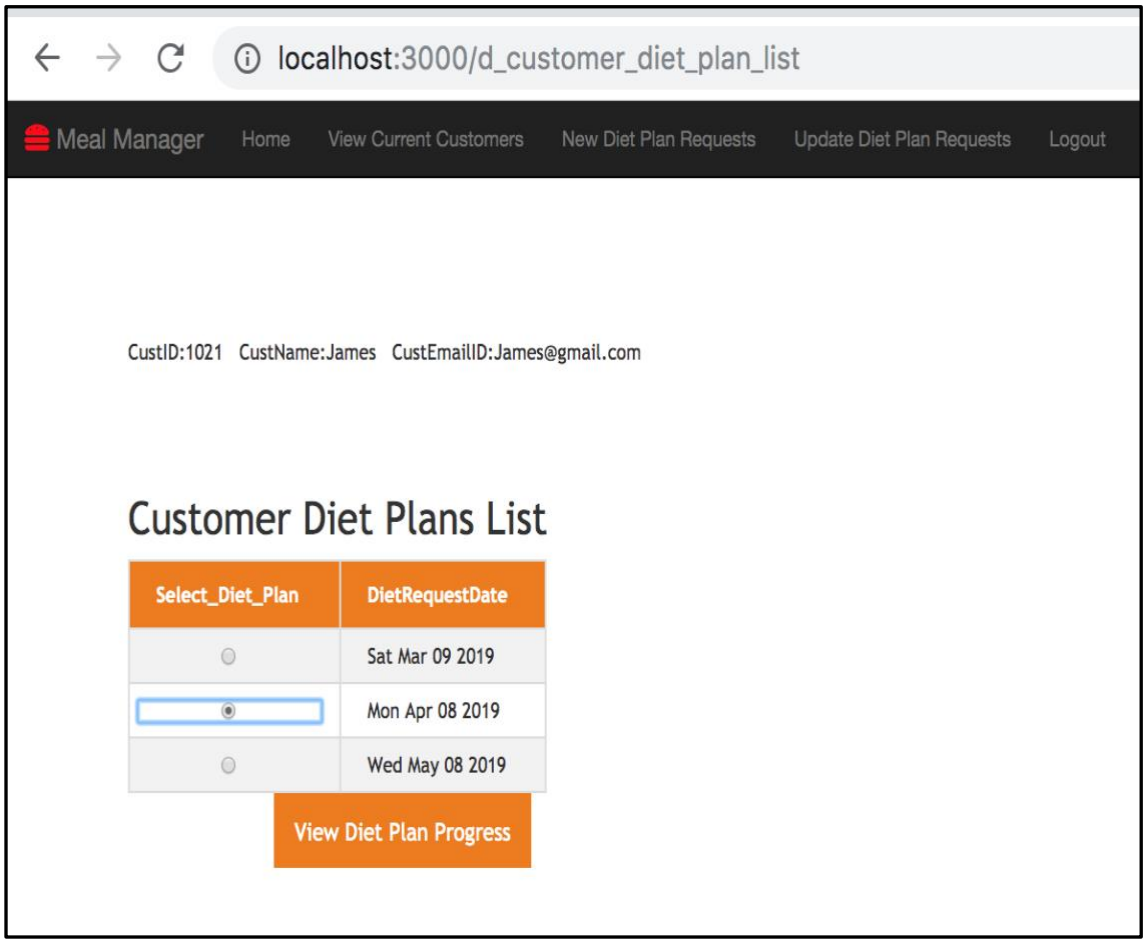
3. Dietician home page when dietician Kate logs in successfully.



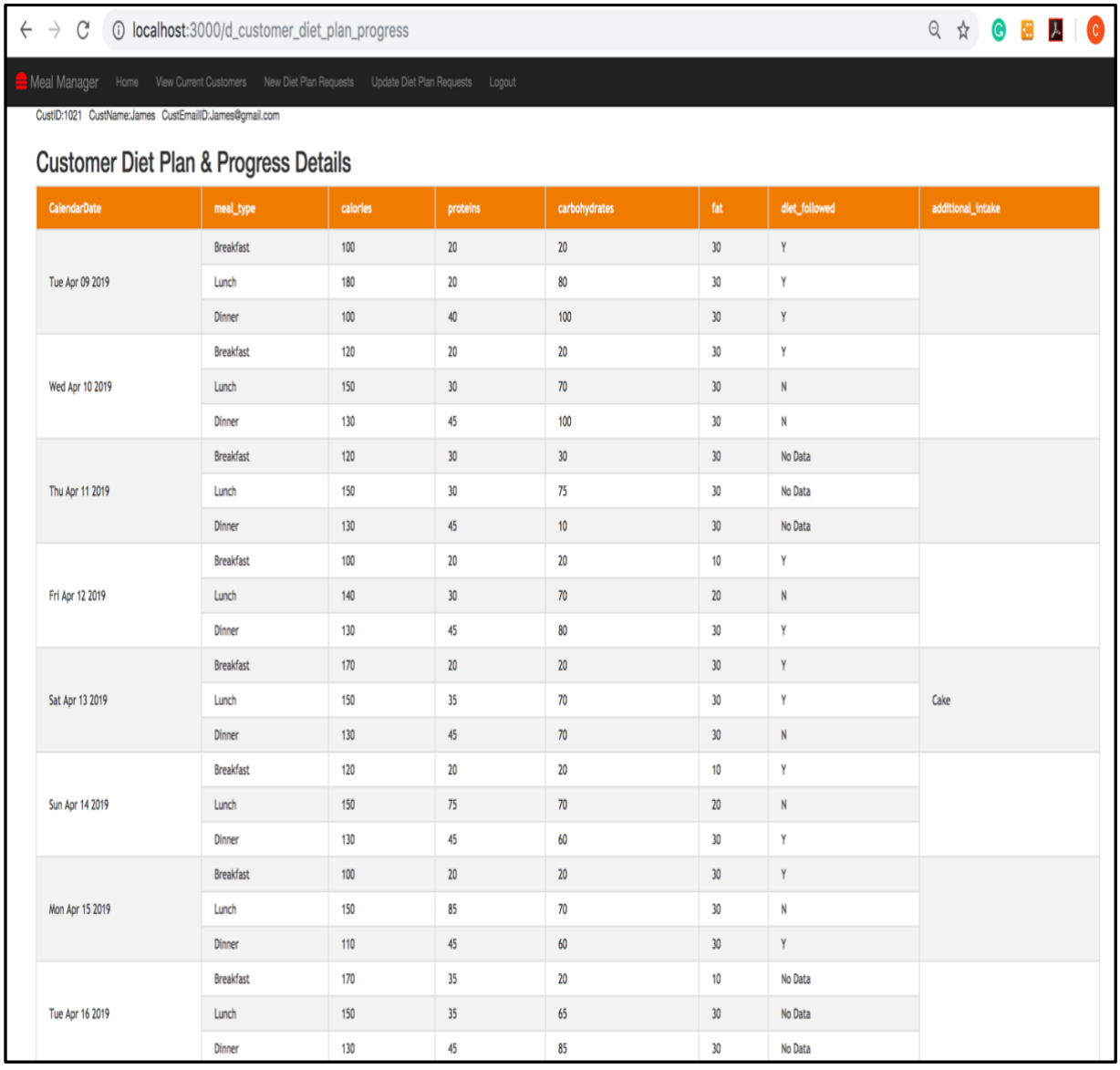
4. List of current customers when clicked on “View Current Customers” button on home page or navigation bar.



5. When “View Diet Plan” button is clicked for James, list of diet plan for James is displayed.

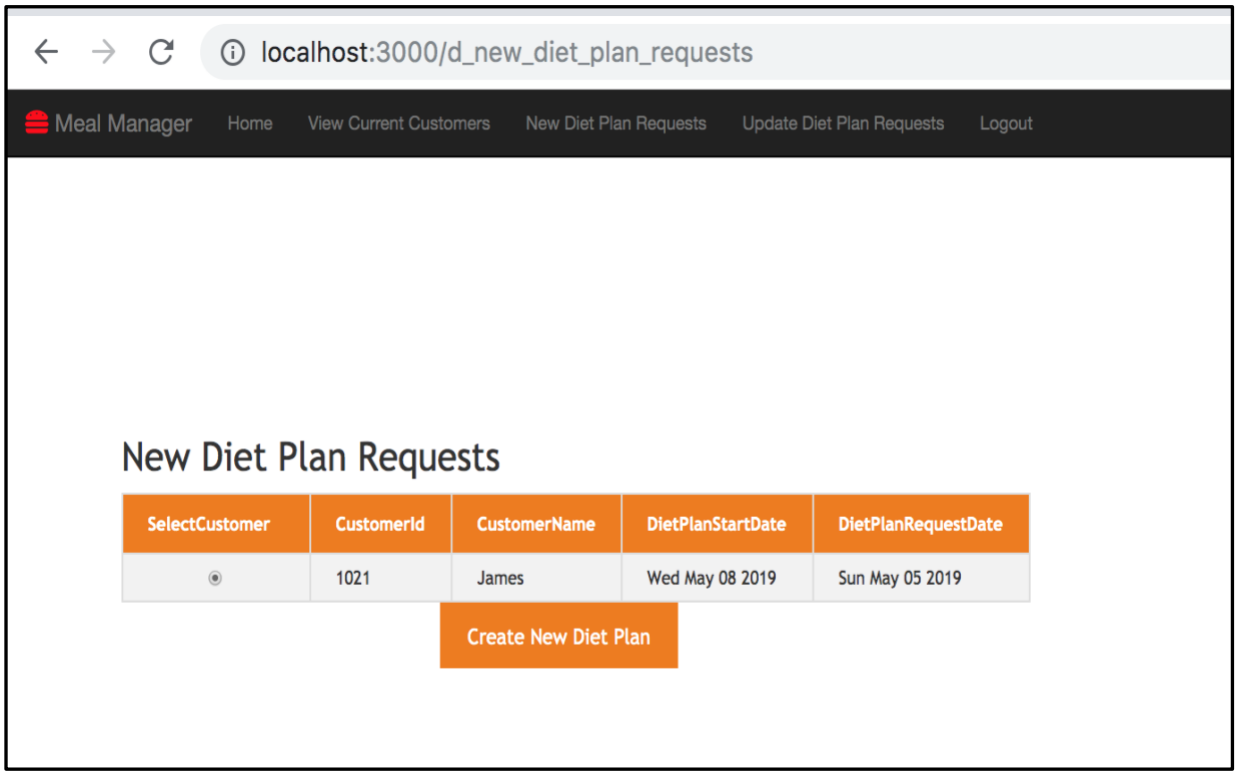


6. When Diet Plan for Apr 8 2019 is selected for James and “view Diet Plan Progress” is clicked, diet plan details and progress can be seen for that particular diet plan. The diet plan given by dietician can be viewed. If James has updated diet plan progress, we can see Y or N in diet_followed column, otherwise ‘No Data’ is seen. When James has added any additional_intake we can the additional_intake data. This helps dietician monitor the diet plan.

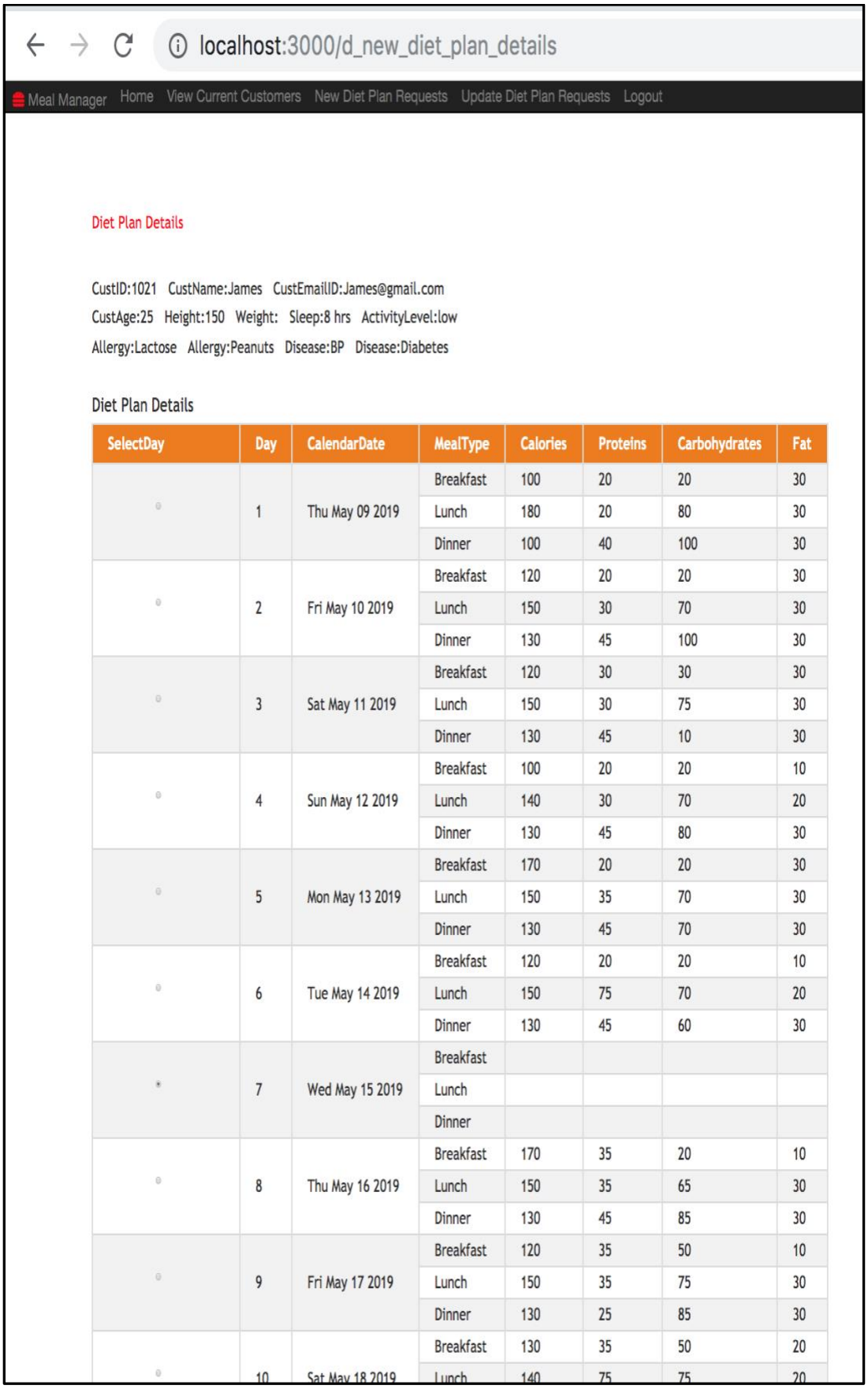


CalendarDate	meal_type	calories	proteins	carbohydrates	fat	diet_followed	additional_intake
Tue Apr 09 2019	Breakfast	100	20	20	30	Y	
	Lunch	180	20	80	30	Y	
	Dinner	100	40	100	30	Y	
Wed Apr 10 2019	Breakfast	120	20	20	30	Y	
	Lunch	150	30	70	30	N	
	Dinner	130	45	100	30	N	
Thu Apr 11 2019	Breakfast	120	30	30	30	No Data	
	Lunch	150	30	75	30	No Data	
	Dinner	130	45	10	30	No Data	
Fri Apr 12 2019	Breakfast	100	20	20	10	Y	
	Lunch	140	30	70	20	N	
	Dinner	130	45	80	30	Y	
Sat Apr 13 2019	Breakfast	170	20	20	30	Y	
	Lunch	150	35	70	30	Y	Cake
	Dinner	130	45	70	30	N	
Sun Apr 14 2019	Breakfast	120	20	20	10	Y	
	Lunch	150	75	70	20	N	
	Dinner	130	45	60	30	Y	
Mon Apr 15 2019	Breakfast	100	20	20	30	Y	
	Lunch	150	85	70	30	N	
	Dinner	110	45	60	30	Y	
Tue Apr 16 2019	Breakfast	170	35	20	10	No Data	
	Lunch	150	35	65	30	No Data	
	Dinner	130	45	85	30	No Data	

7. When “New Diet Plan Requests” button from home page or link for navigation bar, can see new diet plan requests for dietician Kate.



8. To create or to continue creating the new diet plan for James, “Create New Diet Plan” is clicked. Customer details like age, height, weight, sleep, activity level, allergies and diseases can be seen at the top. The days for which Kate has already entered diet plan details, we can see the diet plan details. If the diet plan details is not entered, then diet day is displayed, diet plan details will not be displayed, for example Day -7.



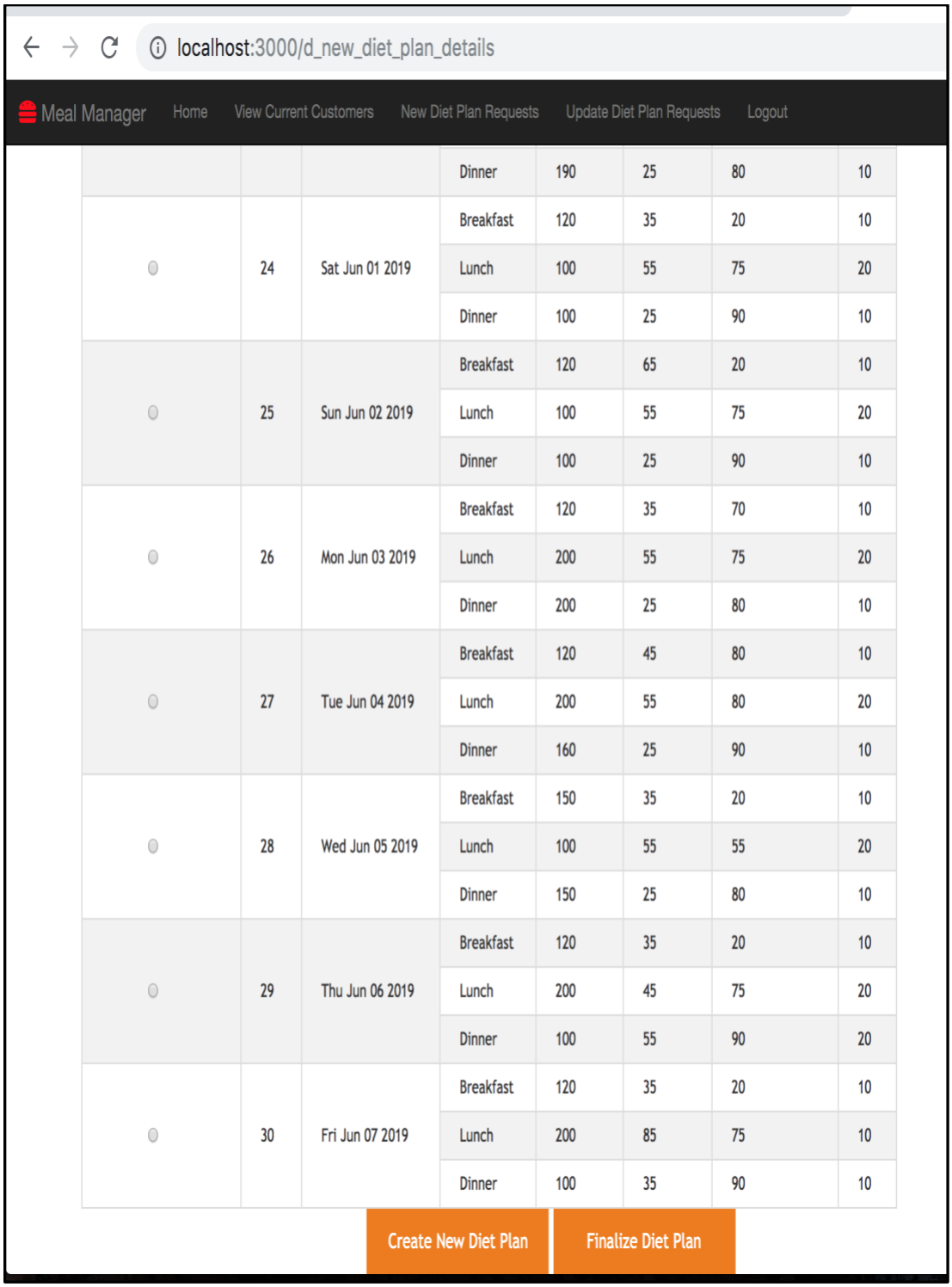
Diet Plan Details

CustID:1021 CustName:James CustEmailID:James@gmail.com
CustAge:25 Height:150 Weight: Sleep:8 hrs ActivityLevel:low
Allergy:Lactose Allergy:Peanuts Disease:BP Disease:Diabetes

Diet Plan Details

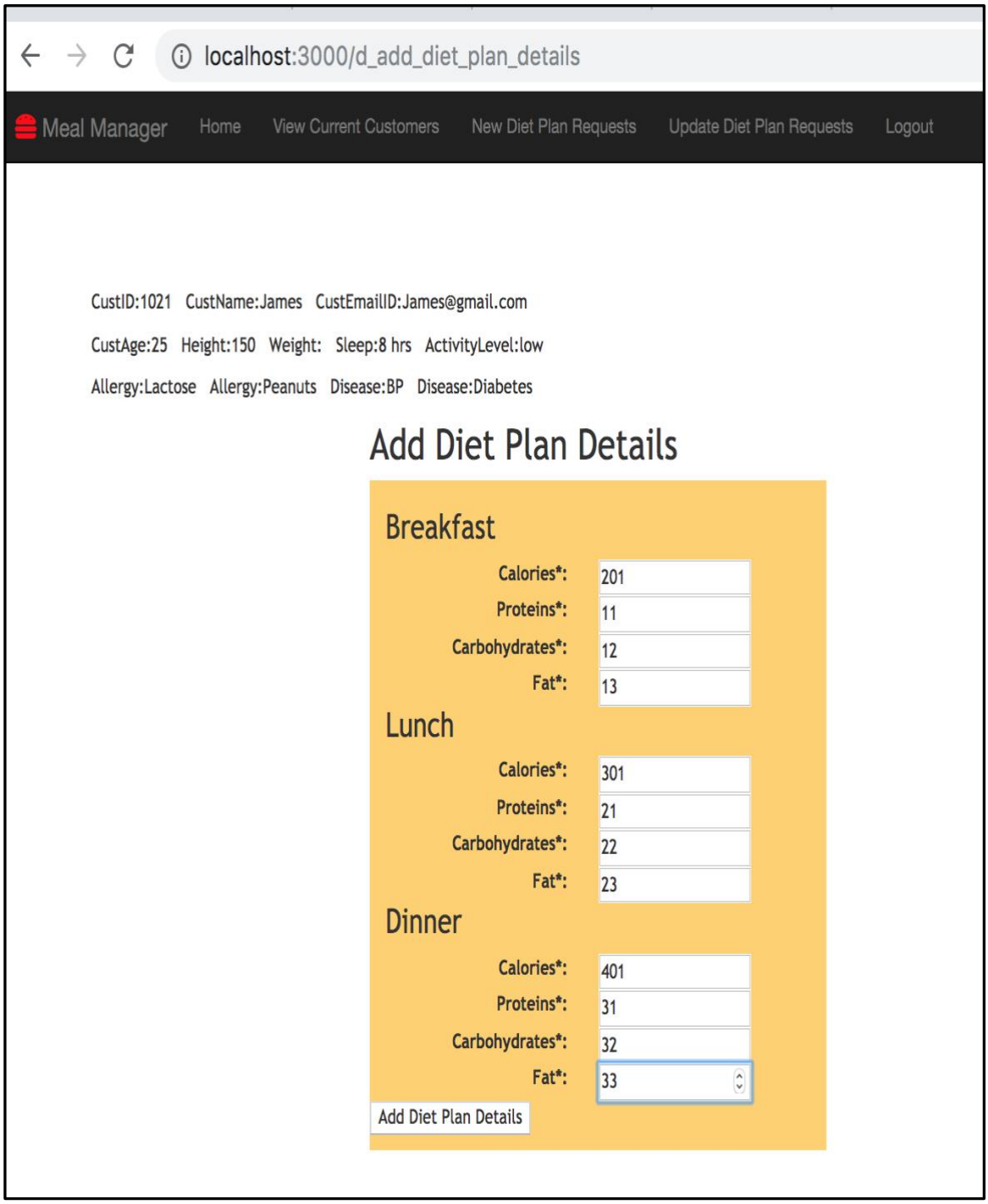
SelectDay	Day	CalendarDate	MealType	Calories	Proteins	Carbohydrates	Fat
	1	Thu May 09 2019	Breakfast	100	20	20	30
			Lunch	180	20	80	30
			Dinner	100	40	100	30
	2	Fri May 10 2019	Breakfast	120	20	20	30
			Lunch	150	30	70	30
			Dinner	130	45	100	30
	3	Sat May 11 2019	Breakfast	120	30	30	30
			Lunch	150	30	75	30
			Dinner	130	45	10	30
	4	Sun May 12 2019	Breakfast	100	20	20	10
			Lunch	140	30	70	20
			Dinner	130	45	80	30
	5	Mon May 13 2019	Breakfast	170	20	20	30
			Lunch	150	35	70	30
			Dinner	130	45	70	30
	6	Tue May 14 2019	Breakfast	120	20	20	10
			Lunch	150	75	70	20
			Dinner	130	45	60	30
	7	Wed May 15 2019	Breakfast				
			Lunch				
			Dinner				
	8	Thu May 16 2019	Breakfast	170	35	20	10
			Lunch	150	35	65	30
			Dinner	130	45	85	30
	9	Fri May 17 2019	Breakfast	120	35	50	10
			Lunch	150	35	75	30
			Dinner	130	25	85	30
	10	Sat May 18 2019	Breakfast	130	35	50	20
			Lunch	140	75	75	20

9. To create diet plan for day 7, select the radio button, and click on “Create New Diet Plan” button shown below.

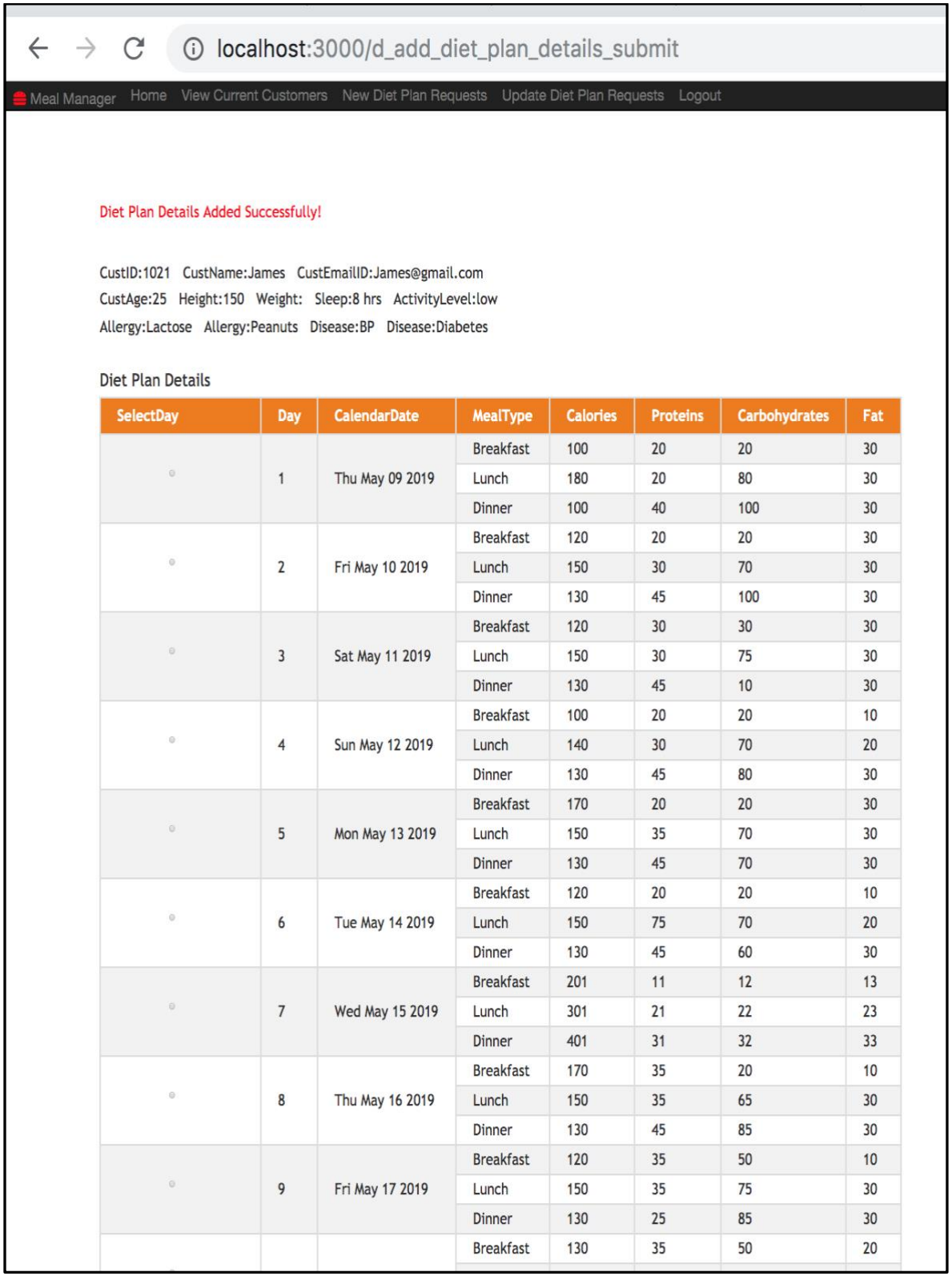


			Dinner	190	25	80	10
	24	Sat Jun 01 2019	Breakfast	120	35	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
	25	Sun Jun 02 2019	Breakfast	120	65	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
	26	Mon Jun 03 2019	Breakfast	120	35	70	10
			Lunch	200	55	75	20
			Dinner	200	25	80	10
	27	Tue Jun 04 2019	Breakfast	120	45	80	10
			Lunch	200	55	80	20
			Dinner	160	25	90	10
	28	Wed Jun 05 2019	Breakfast	150	35	20	10
			Lunch	100	55	55	20
			Dinner	150	25	80	10
	29	Thu Jun 06 2019	Breakfast	120	35	20	10
			Lunch	200	45	75	20
			Dinner	100	55	90	20
	30	Fri Jun 07 2019	Breakfast	120	35	20	10
			Lunch	200	85	75	10
			Dinner	100	35	90	10

10. Add Diet Plan details is displayed for day-7. Fill the details and click on “Add Diet Plan Details” button.

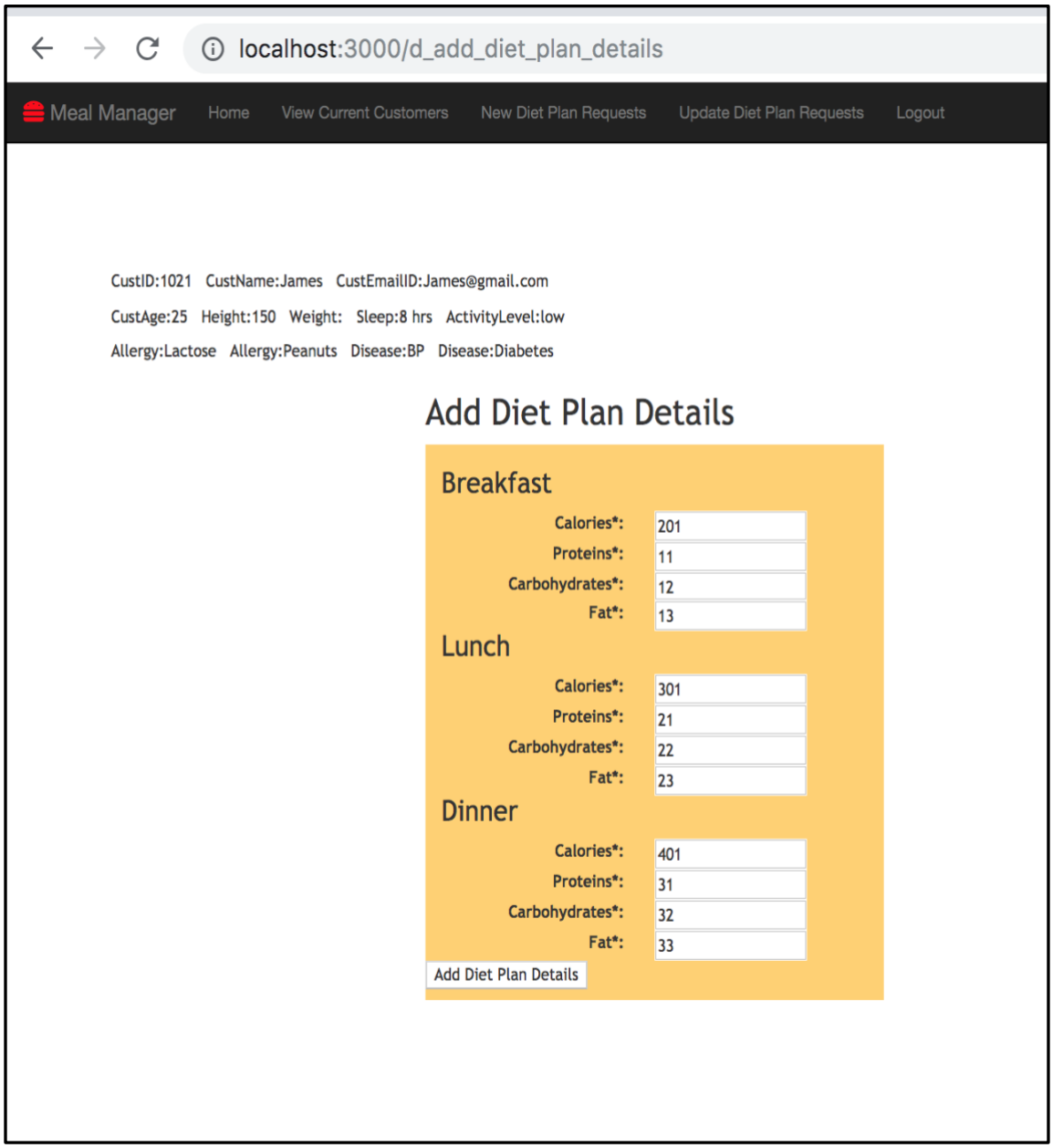


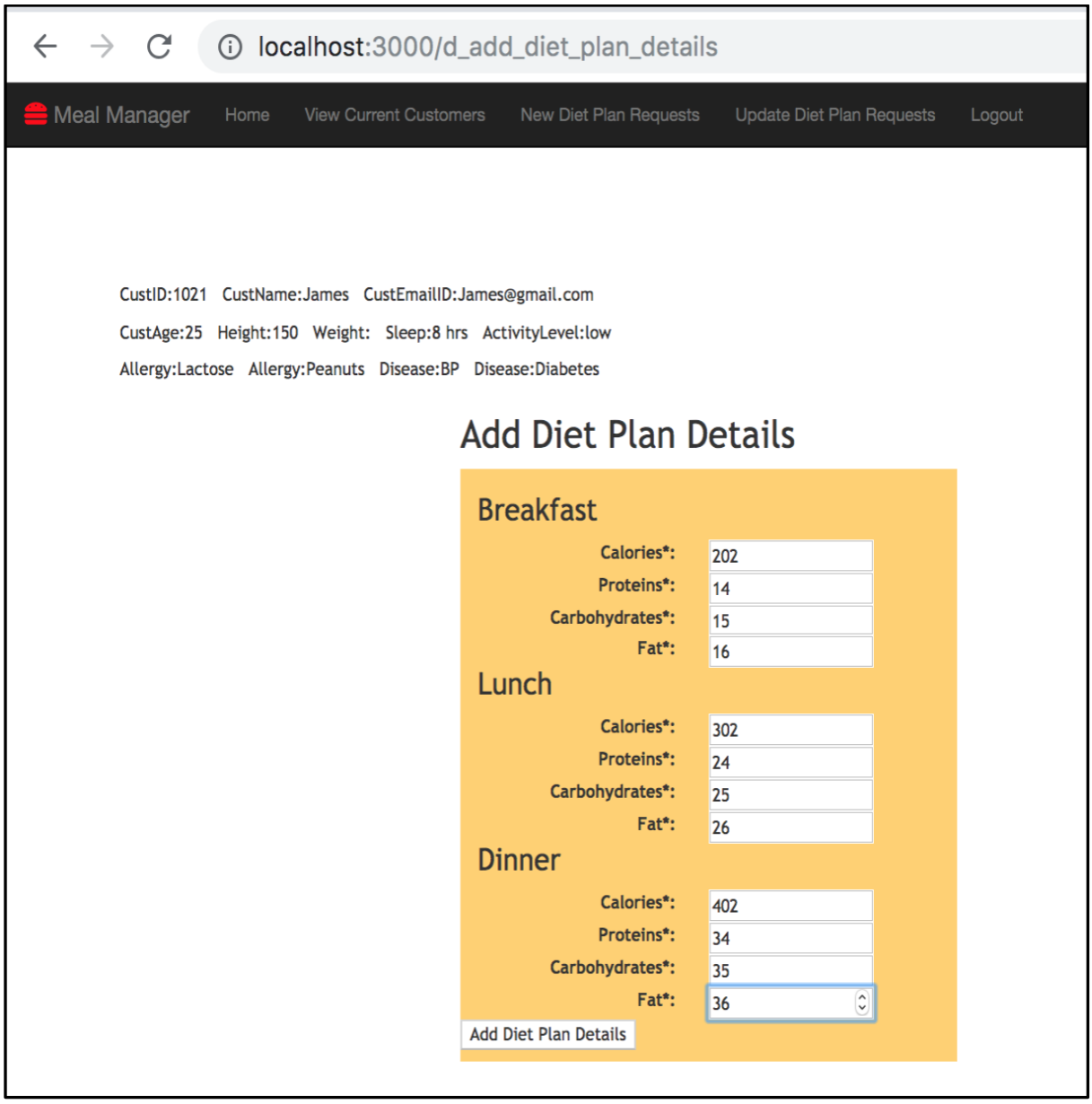
11. Diet plan details for Day-7 is added, we can see the success message at the top. Also, for Day-7, the diet plan details can be seen.



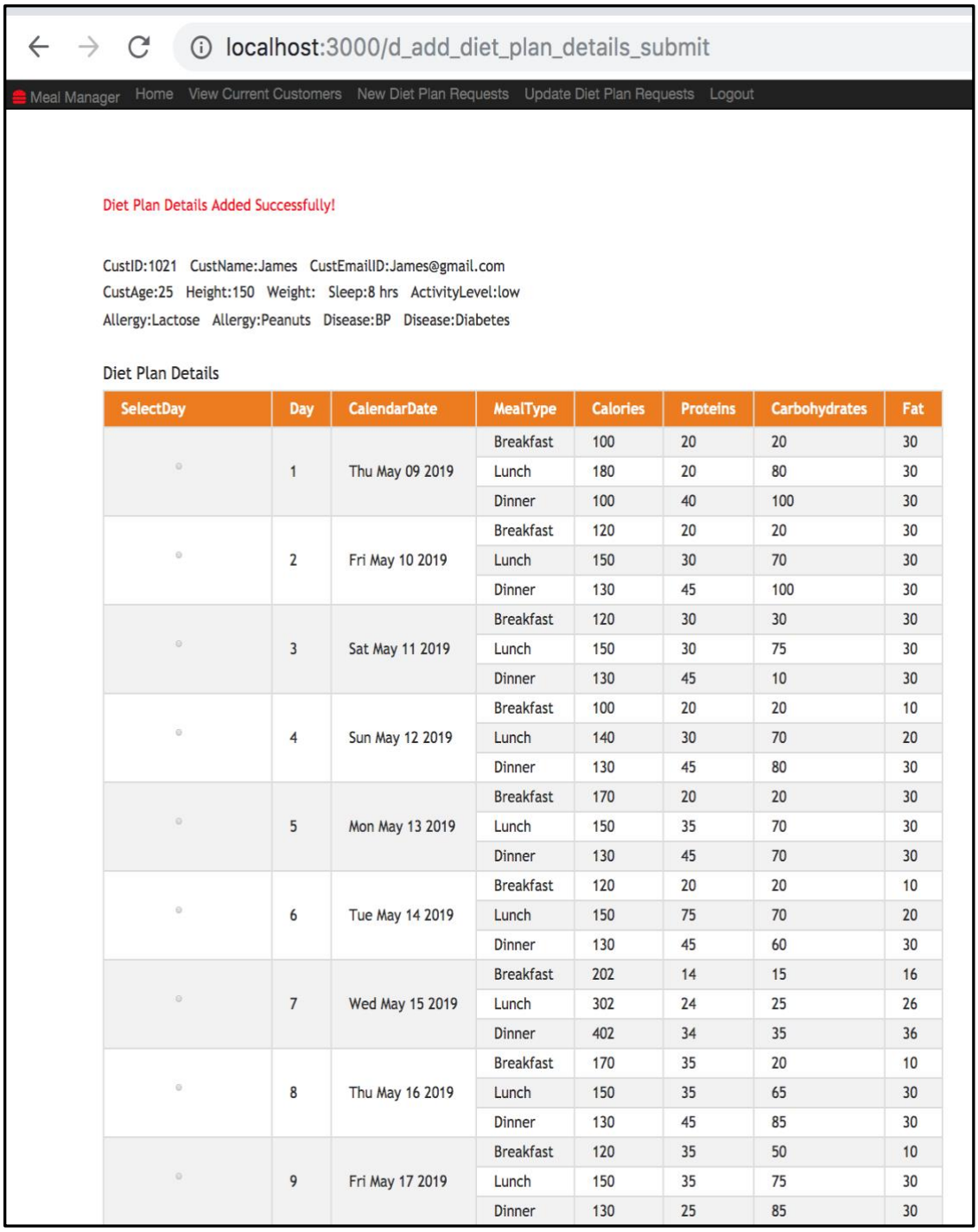
SelectDay	Day	CalendarDate	MealType	Calories	Proteins	Carbohydrates	Fat
	1	Thu May 09 2019	Breakfast	100	20	20	30
			Lunch	180	20	80	30
			Dinner	100	40	100	30
	2	Fri May 10 2019	Breakfast	120	20	20	30
			Lunch	150	30	70	30
			Dinner	130	45	100	30
	3	Sat May 11 2019	Breakfast	120	30	30	30
			Lunch	150	30	75	30
			Dinner	130	45	10	30
	4	Sun May 12 2019	Breakfast	100	20	20	10
			Lunch	140	30	70	20
			Dinner	130	45	80	30
	5	Mon May 13 2019	Breakfast	170	20	20	30
			Lunch	150	35	70	30
			Dinner	130	45	70	30
	6	Tue May 14 2019	Breakfast	120	20	20	10
			Lunch	150	75	70	20
			Dinner	130	45	60	30
	7	Wed May 15 2019	Breakfast	201	11	12	13
			Lunch	301	21	22	23
			Dinner	401	31	32	33
	8	Thu May 16 2019	Breakfast	170	35	20	10
			Lunch	150	35	65	30
			Dinner	130	45	85	30
	9	Fri May 17 2019	Breakfast	120	35	50	10
			Lunch	150	35	75	30
			Dinner	130	25	85	30
			Breakfast	130	35	50	20

12. If not happy with the diet plan for any day, select the day and click on “Create new diet plan” button at the bottom of the page. Let’s update for Day-7. Diet plan details for day-7 is prefilled. Modify the details as required, and click on “Add Diet Plan Details” button.



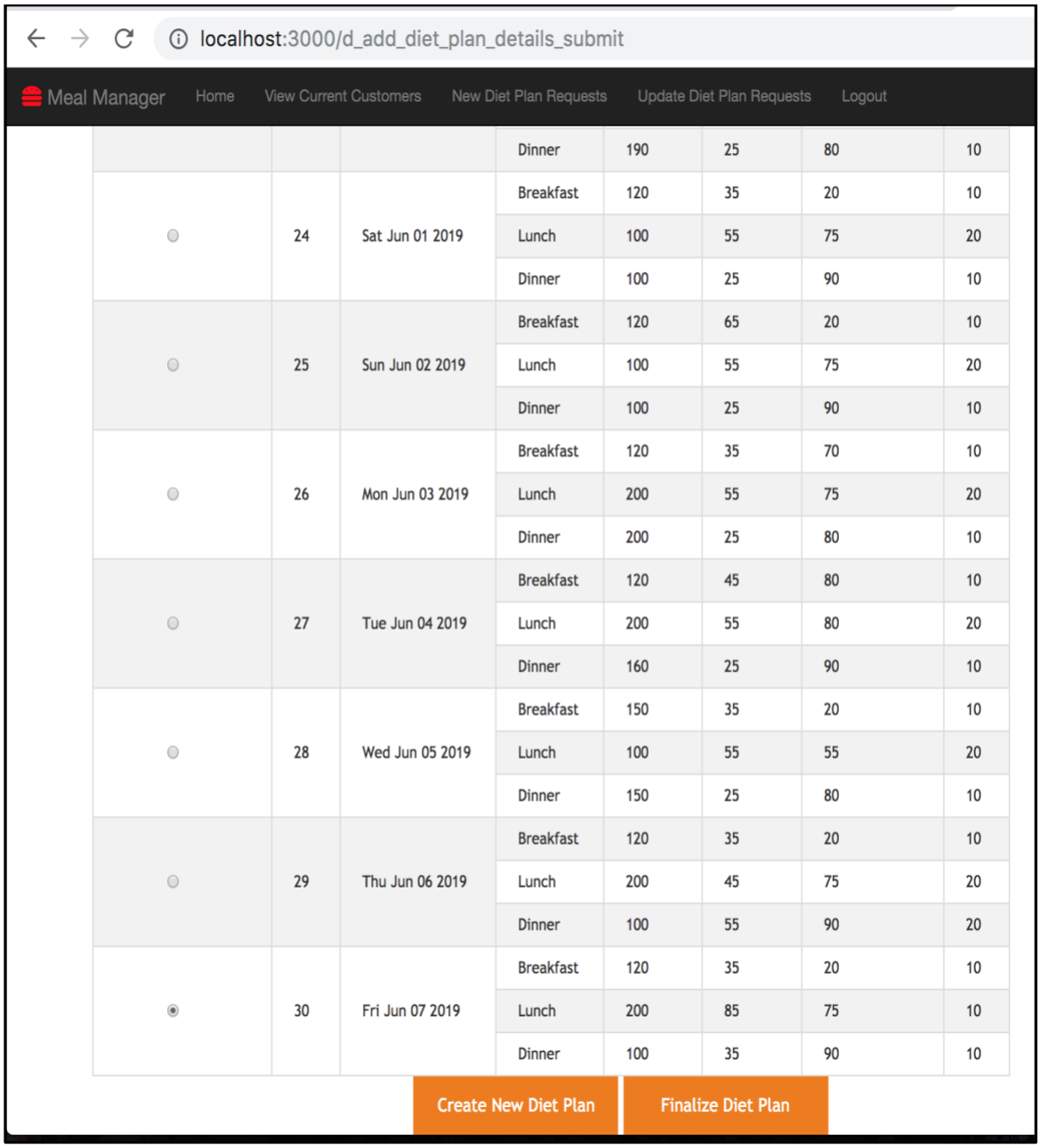

localhost:3000/d_add_diet_plan_details
Meal Manager
Home
View Current Customers
New Diet Plan Requests
Update Diet Plan Requests
Logout
CustID:1021 CustName:James CustEmailID:James@gmail.com
CustAge:25 Height:150 Weight: Sleep:8 hrs ActivityLevel:low
Allergy:Lactose Allergy:Peanuts Disease:BP Disease:Diabetes
Add Diet Plan Details
Breakfast
Calories*: 202
Proteins*: 14
Carbohydrates*: 15
Fat*: 16
Lunch
Calories*: 302
Proteins*: 24
Carbohydrates*: 25
Fat*: 26
Dinner
Calories*: 402
Proteins*: 34
Carbohydrates*: 35
Fat*: 36
Add Diet Plan Details

13. Diet plan details are successfully added. Success message with modified diet plan day 7 can be seen.



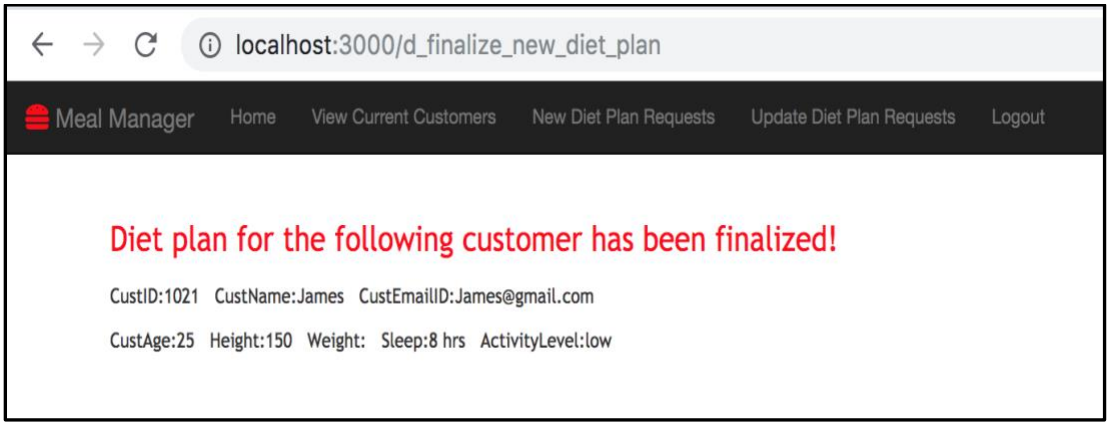
SelectDay	Day	CalendarDate	MealType	Calories	Proteins	Carbohydrates	Fat
	1	Thu May 09 2019	Breakfast	100	20	20	30
			Lunch	180	20	80	30
			Dinner	100	40	100	30
	2	Fri May 10 2019	Breakfast	120	20	20	30
			Lunch	150	30	70	30
			Dinner	130	45	100	30
	3	Sat May 11 2019	Breakfast	120	30	30	30
			Lunch	150	30	75	30
			Dinner	130	45	10	30
	4	Sun May 12 2019	Breakfast	100	20	20	10
			Lunch	140	30	70	20
			Dinner	130	45	80	30
	5	Mon May 13 2019	Breakfast	170	20	20	30
			Lunch	150	35	70	30
			Dinner	130	45	70	30
	6	Tue May 14 2019	Breakfast	120	20	20	10
			Lunch	150	75	70	20
			Dinner	130	45	60	30
	7	Wed May 15 2019	Breakfast	202	14	15	16
			Lunch	302	24	25	26
			Dinner	402	34	35	36
	8	Thu May 16 2019	Breakfast	170	35	20	10
			Lunch	150	35	65	30
			Dinner	130	45	85	30
	9	Fri May 17 2019	Breakfast	120	35	50	10
			Lunch	150	35	75	30
			Dinner	130	25	85	30

14. If the dietician is happy with diet plan for all 30 days, click on “Finalize Diet Plan” at the bottom of the page. This will finalize the diet plan, and the request will be removed from new diet plans list.

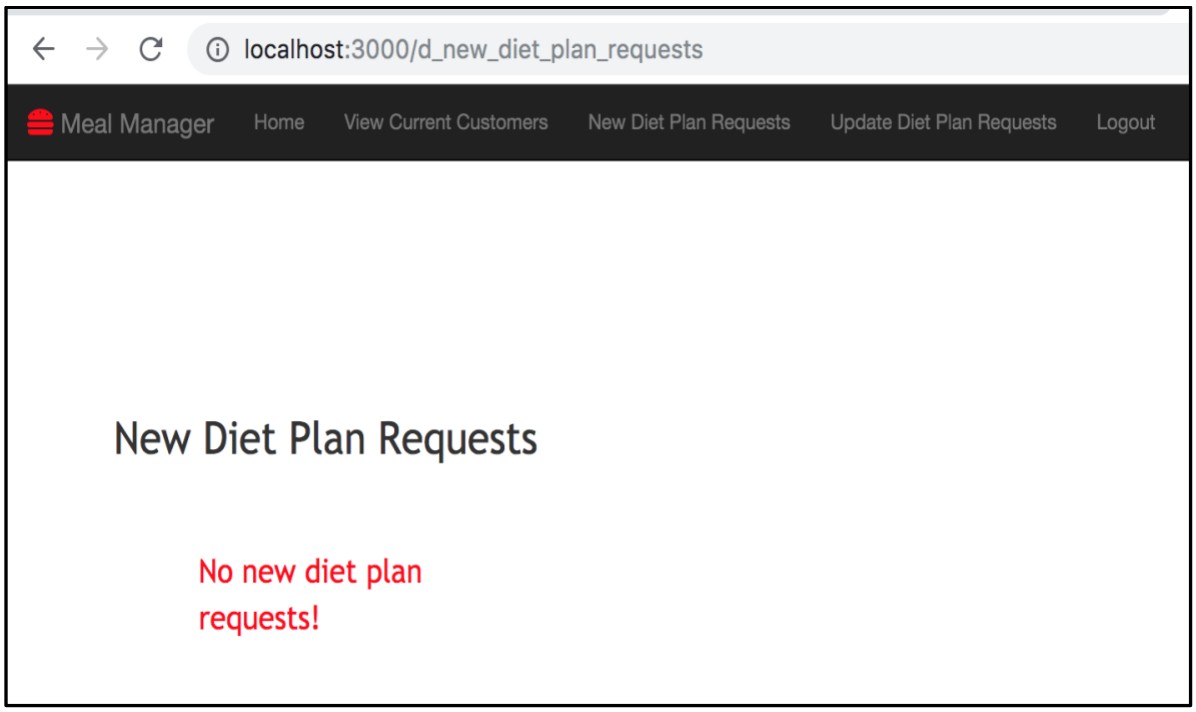


			Dinner	190	25	80	10
○	24	Sat Jun 01 2019	Breakfast	120	35	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
○	25	Sun Jun 02 2019	Breakfast	120	65	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
○	26	Mon Jun 03 2019	Breakfast	120	35	70	10
			Lunch	200	55	75	20
			Dinner	200	25	80	10
○	27	Tue Jun 04 2019	Breakfast	120	45	80	10
			Lunch	200	55	80	20
			Dinner	160	25	90	10
○	28	Wed Jun 05 2019	Breakfast	150	35	20	10
			Lunch	100	55	55	20
			Dinner	150	25	80	10
○	29	Thu Jun 06 2019	Breakfast	120	35	20	10
			Lunch	200	45	75	20
			Dinner	100	55	90	20
◉	30	Fri Jun 07 2019	Breakfast	120	35	20	10
			Lunch	200	85	75	10
			Dinner	100	35	90	10

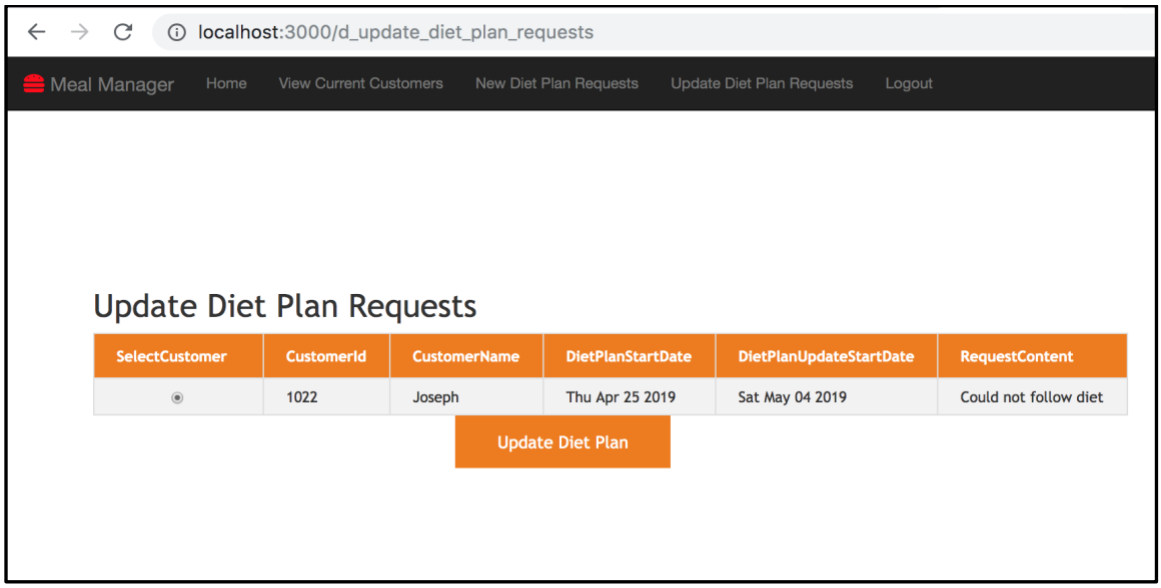
15. We get the following success message.



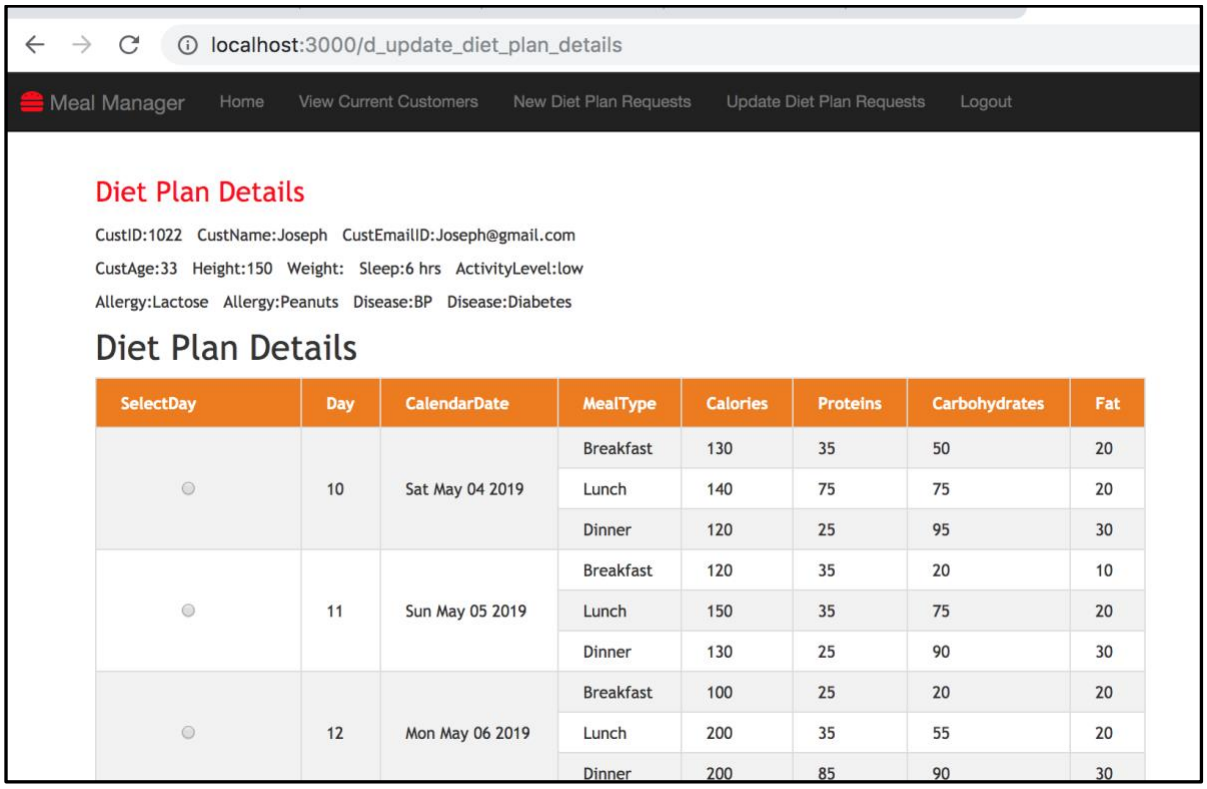
16. When “New Diet Plan Requests” is clicked again, the new diet plan request from James will be removed. Also, since, Kate does not have any other new diet plan requests, “No New Diet Plan Requests” message will be displayed.



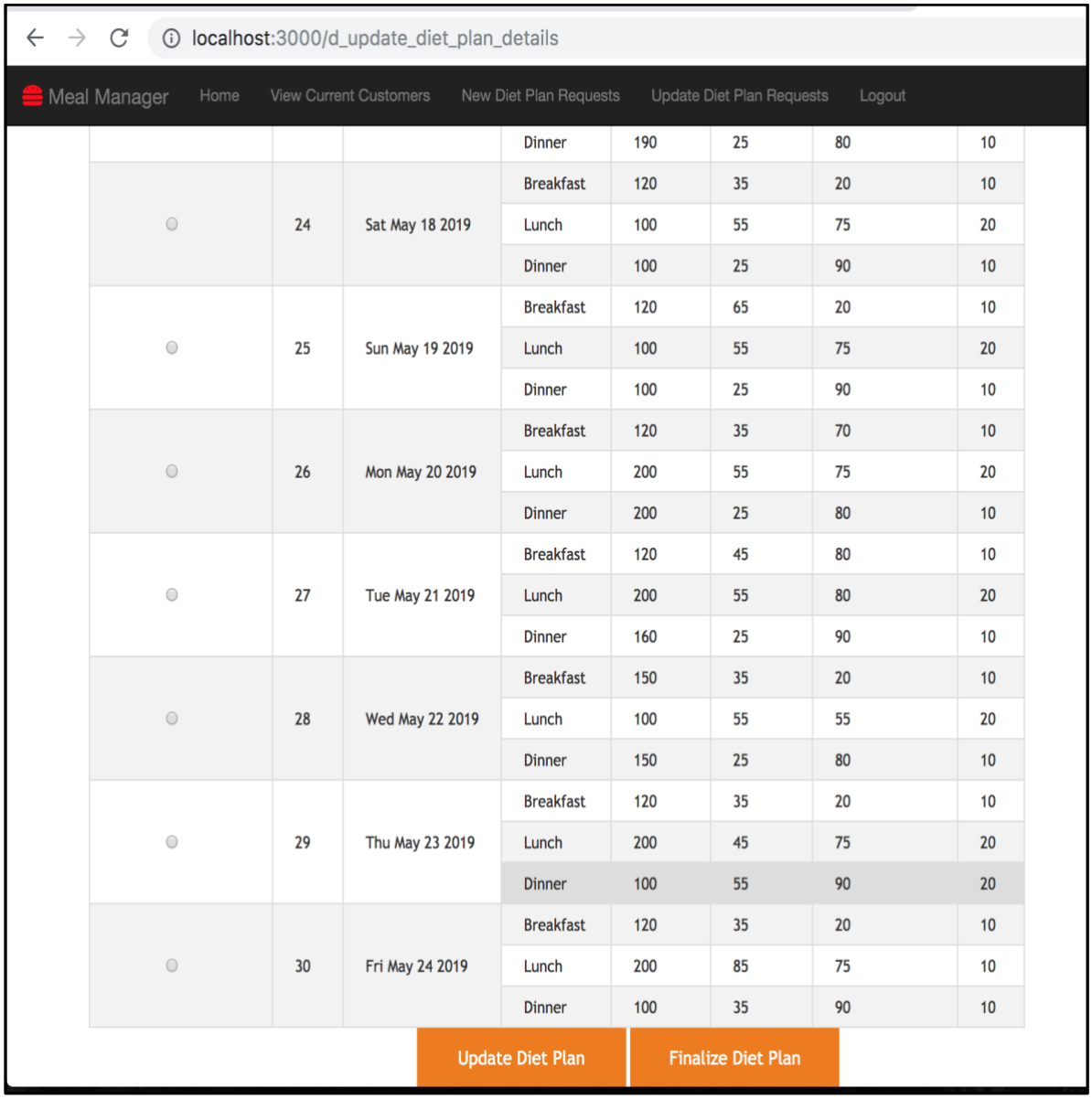
17. Update diet plan requests can be viewed from “Update Diet Plan Requests” button on home page or from link in navigation bar. This can be used to either update or continue updating a diet plan.



18. Kate has 1 update diet plan request from Joseph. The diet start date was Apr 25th 2019. He needs the plan to be updated from May 4th 2019. When the radio button is selected and “Update Diet Plan” button is clicked, the diet plan details from May 4th 2019 till the end of diet plan will be displayed.

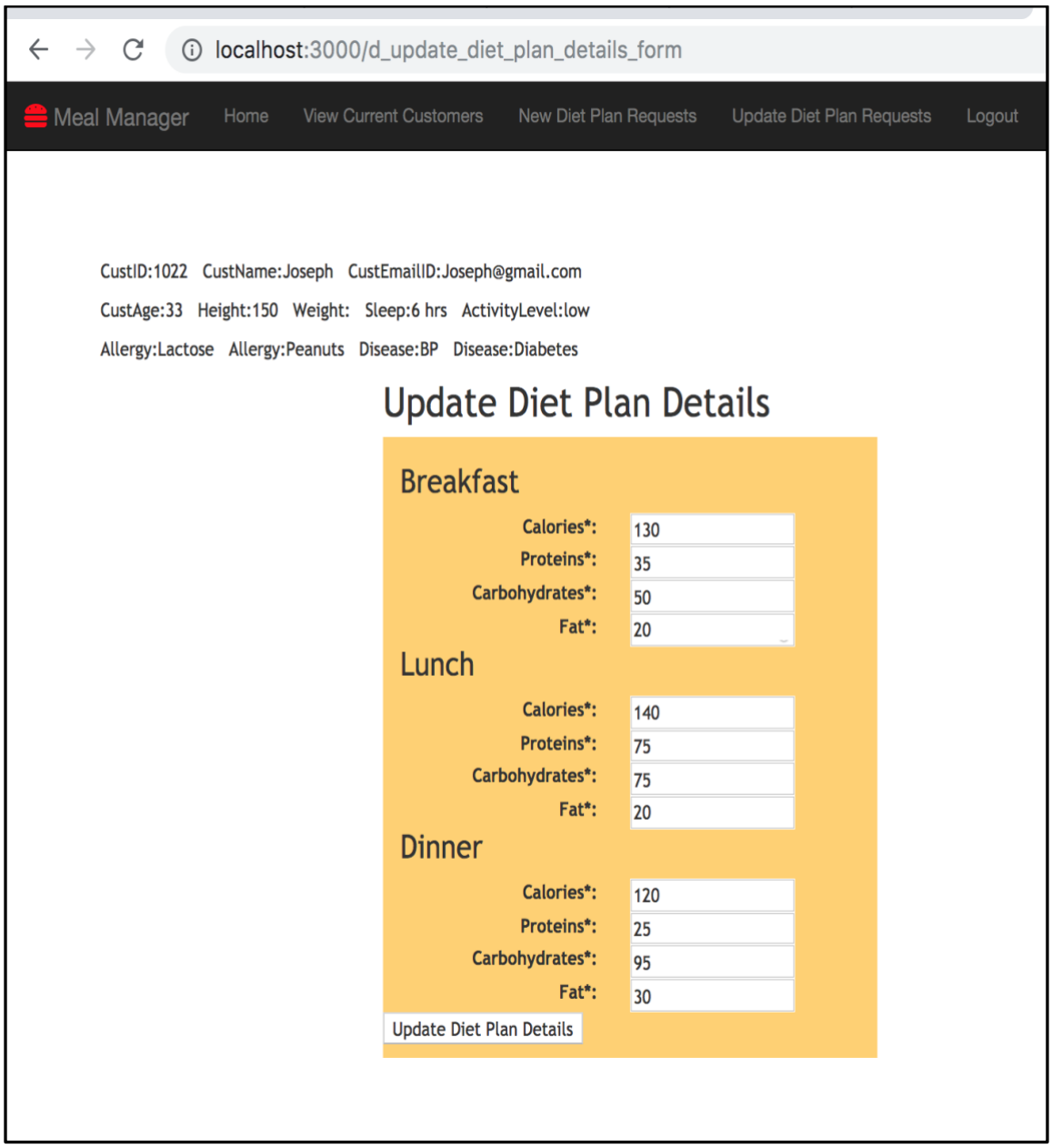

localhost:3000/d_update_diet_plan_details
Meal Manager
Home
View Current Customers
New Diet Plan Requests
Update Diet Plan Requests
Logout
Diet Plan Details
CustID:1022 CustName:Joseph CustEmailID:Joseph@gmail.com
CustAge:33 Height:150 Weight: Sleep:6 hrs ActivityLevel:low
Allergy:Lactose Allergy:Peanuts Disease:BP Disease:Diabetes
Diet Plan Details
SelectDay
Day
CalendarDate
MealType
Calories
Proteins
Carbohydrates
Fat
10
Sat May 04 2019
Breakfast 130 35 50 20
Lunch 140 75 75 20
Dinner 120 25 95 30
11
Sun May 05 2019
Breakfast 120 35 20 10
Lunch 150 35 75 20
Dinner 130 25 90 30
12
Mon May 06 2019
Breakfast 100 25 20 20
Lunch 200 35 55 20
Dinner 200 85 90 30

19. To update the diet plan for May 4th 2019, select the corresponding button and click on the “Update Diet Plan” button at the bottom of the page.

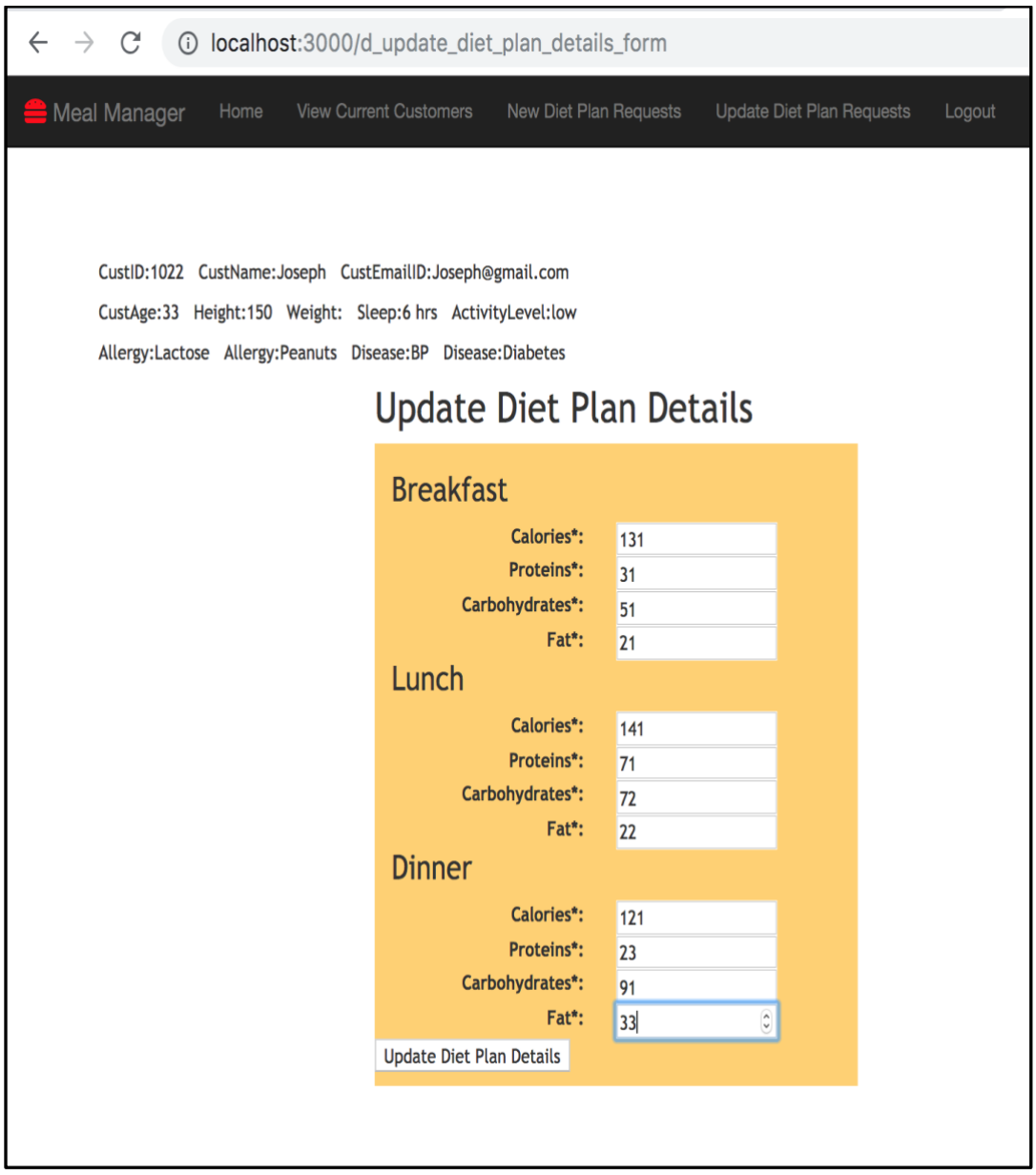


			Dinner	190	25	80	10
○	24	Sat May 18 2019	Breakfast	120	35	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
○	25	Sun May 19 2019	Breakfast	120	65	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
○	26	Mon May 20 2019	Breakfast	120	35	70	10
			Lunch	200	55	75	20
			Dinner	200	25	80	10
○	27	Tue May 21 2019	Breakfast	120	45	80	10
			Lunch	200	55	80	20
			Dinner	160	25	90	10
○	28	Wed May 22 2019	Breakfast	150	35	20	10
			Lunch	100	55	55	20
			Dinner	150	25	80	10
○	29	Thu May 23 2019	Breakfast	120	35	20	10
			Lunch	200	45	75	20
			Dinner	100	55	90	20
○	30	Fri May 24 2019	Breakfast	120	35	20	10
			Lunch	200	85	75	10
			Dinner	100	35	90	10

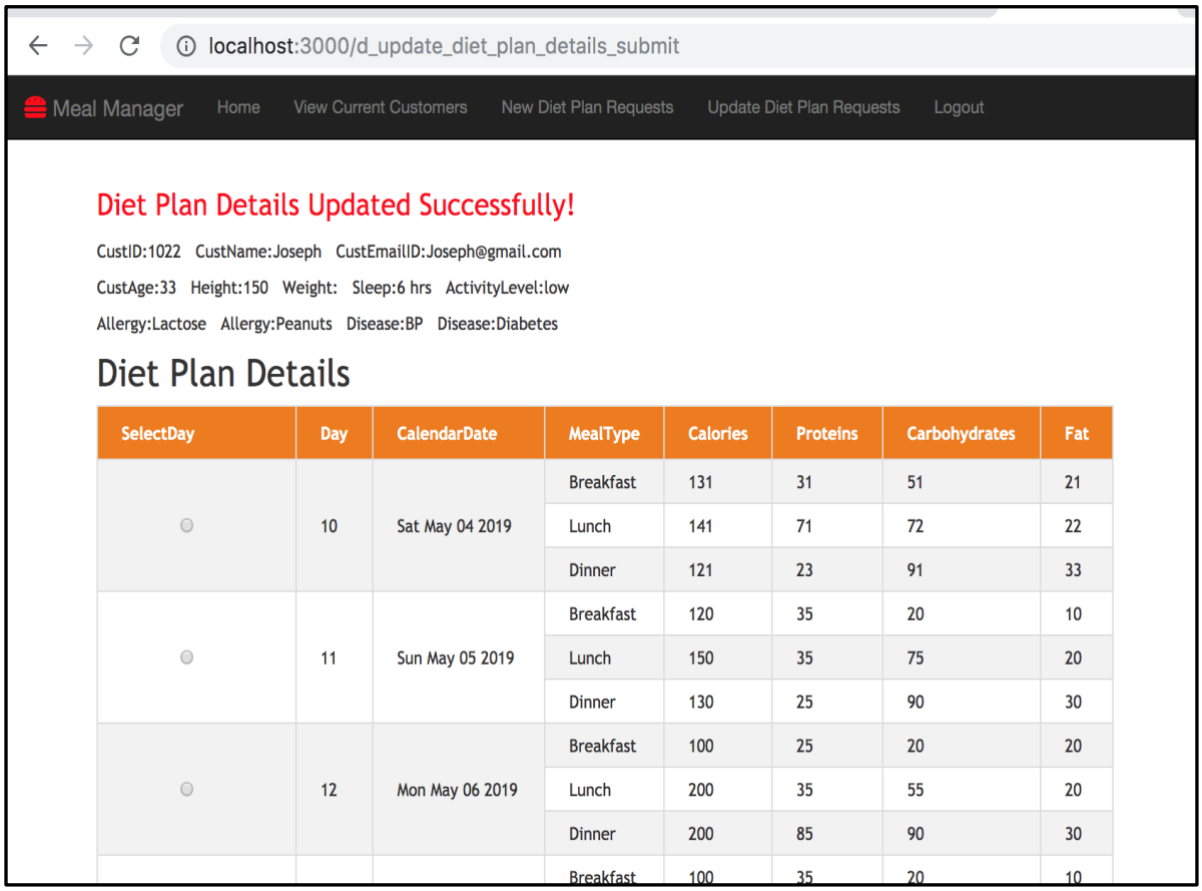
20. Update diet plan form for May 4th along with existing diet plan details is fetched.



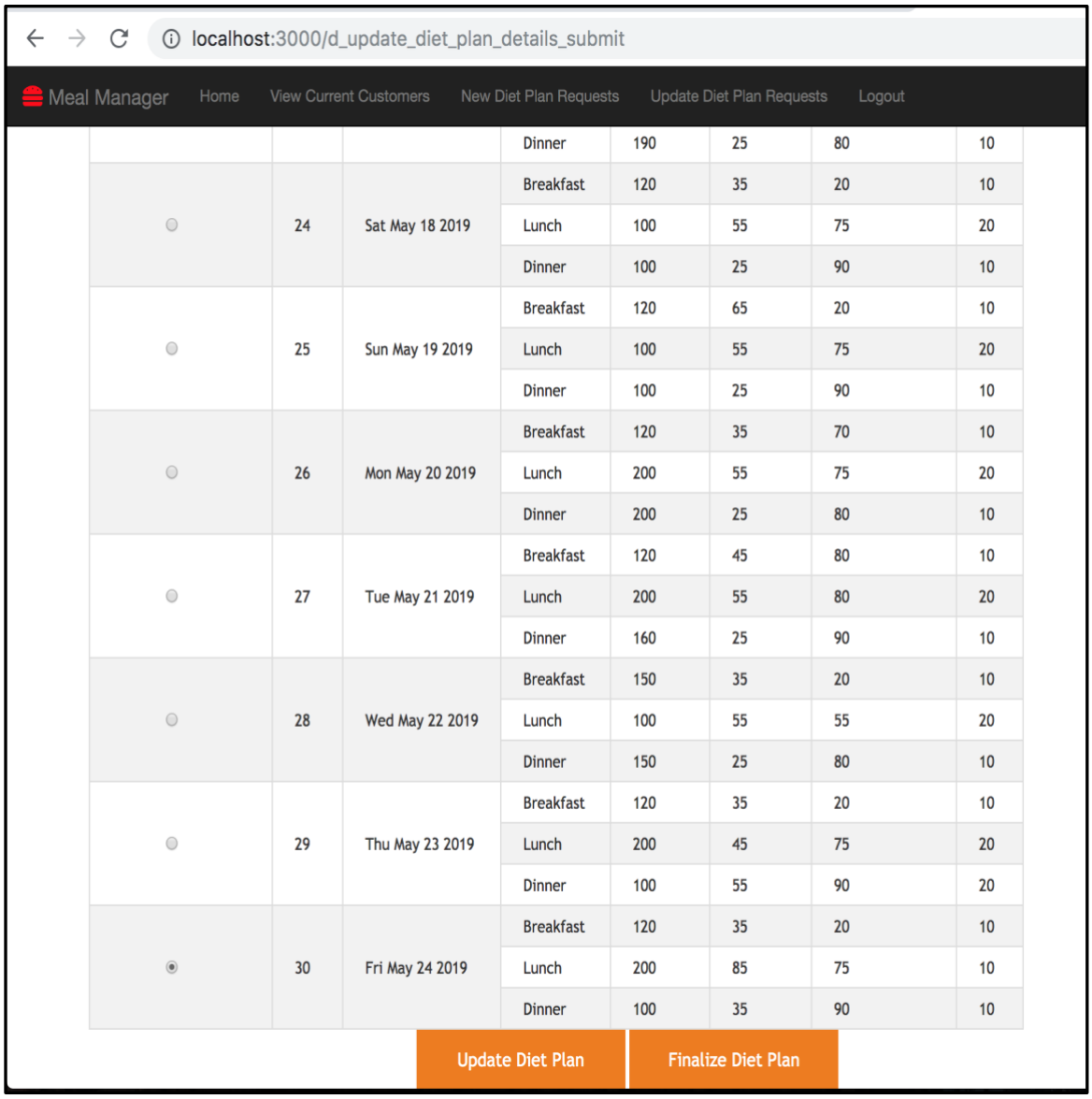
21. Update the diet plan details and click on “Update Diet Plan Details” button.



22. Diet plan details is updated for May 4th and a success message is displayed at the top. The page also shows the updated plan details.

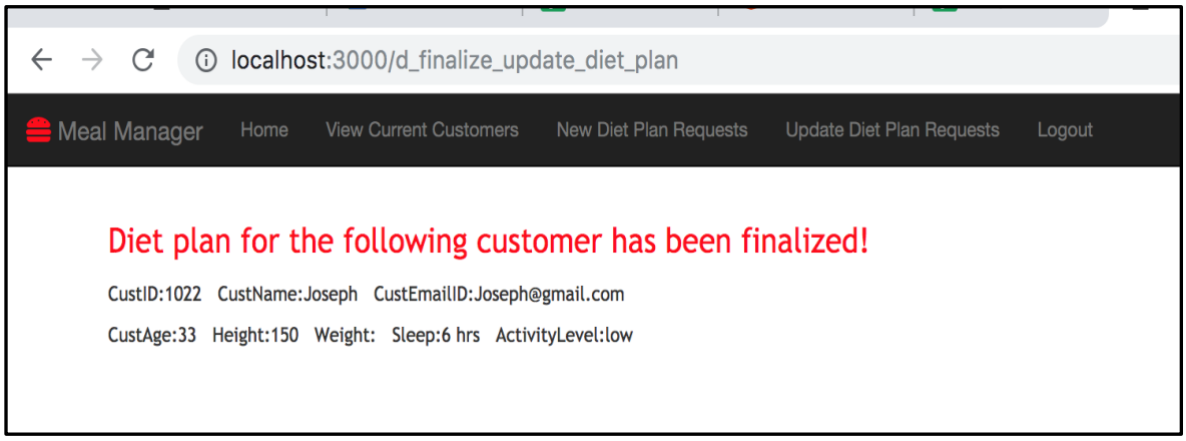


23. This process of updating can be continued till the dietician is happy with the diet plan. Once the diet plan is updated completely, “Finalize Diet Plan” button at the bottom of the page can be clicked. This will remove the request from the update diet plan request list.

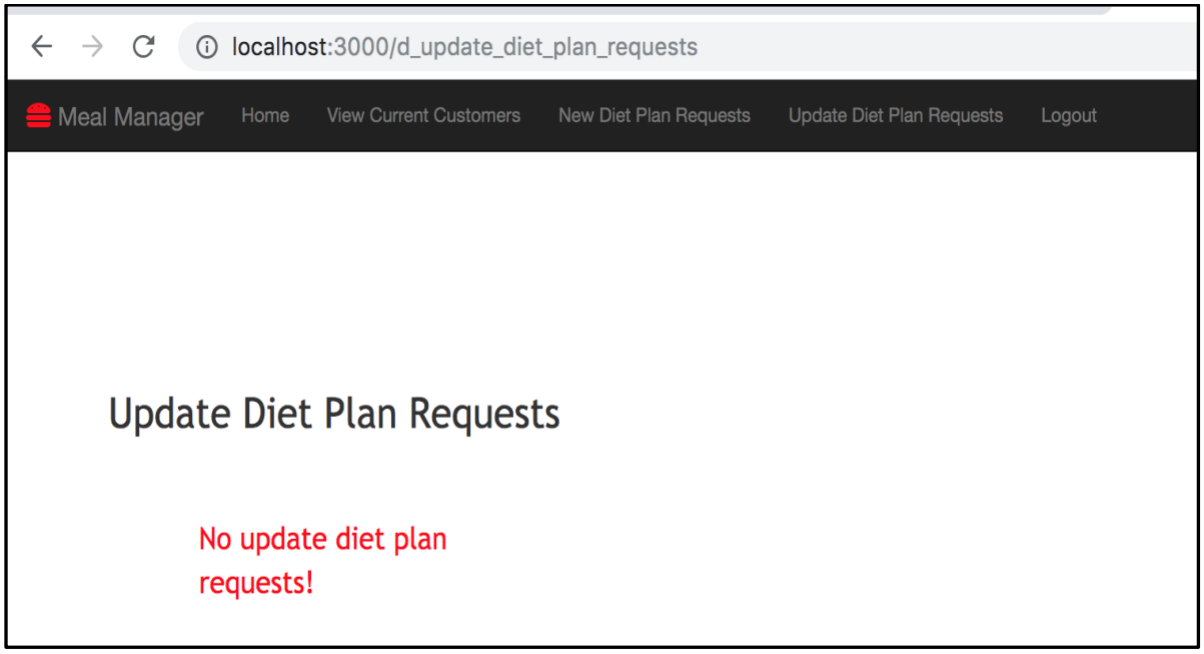


			Dinner	190	25	80	10
	24	Sat May 18 2019	Breakfast	120	35	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
	25	Sun May 19 2019	Breakfast	120	65	20	10
			Lunch	100	55	75	20
			Dinner	100	25	90	10
	26	Mon May 20 2019	Breakfast	120	35	70	10
			Lunch	200	55	75	20
			Dinner	200	25	80	10
	27	Tue May 21 2019	Breakfast	120	45	80	10
			Lunch	200	55	80	20
			Dinner	160	25	90	10
	28	Wed May 22 2019	Breakfast	150	35	20	10
			Lunch	100	55	55	20
			Dinner	150	25	80	10
	29	Thu May 23 2019	Breakfast	120	35	20	10
			Lunch	200	45	75	20
			Dinner	100	55	90	20
	30	Fri May 24 2019	Breakfast	120	35	20	10
			Lunch	200	85	75	10
			Dinner	100	35	90	10

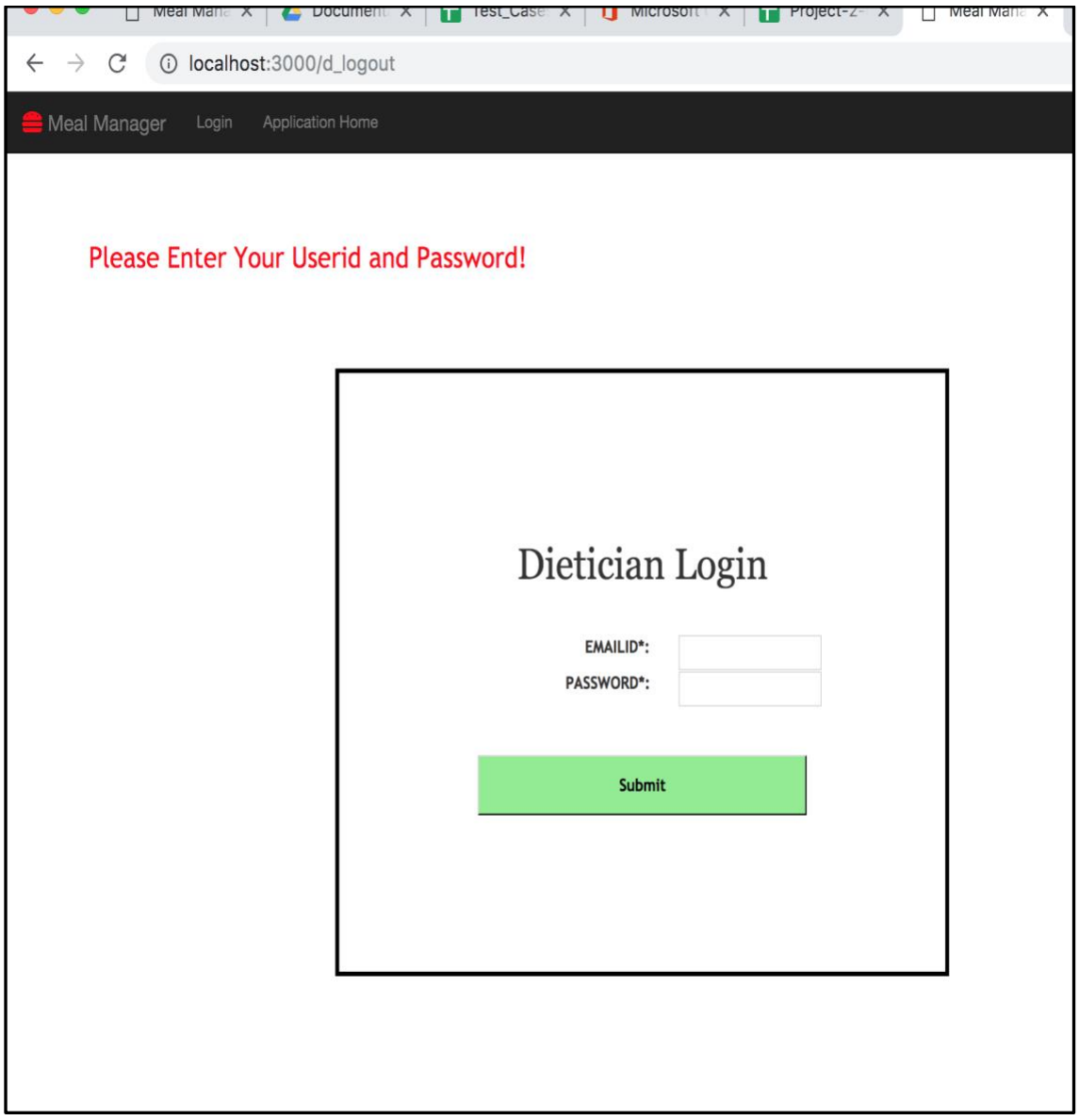
24. On finalizing the diet plan, the following success message Is seen, and Joseph's request is removed from the list.



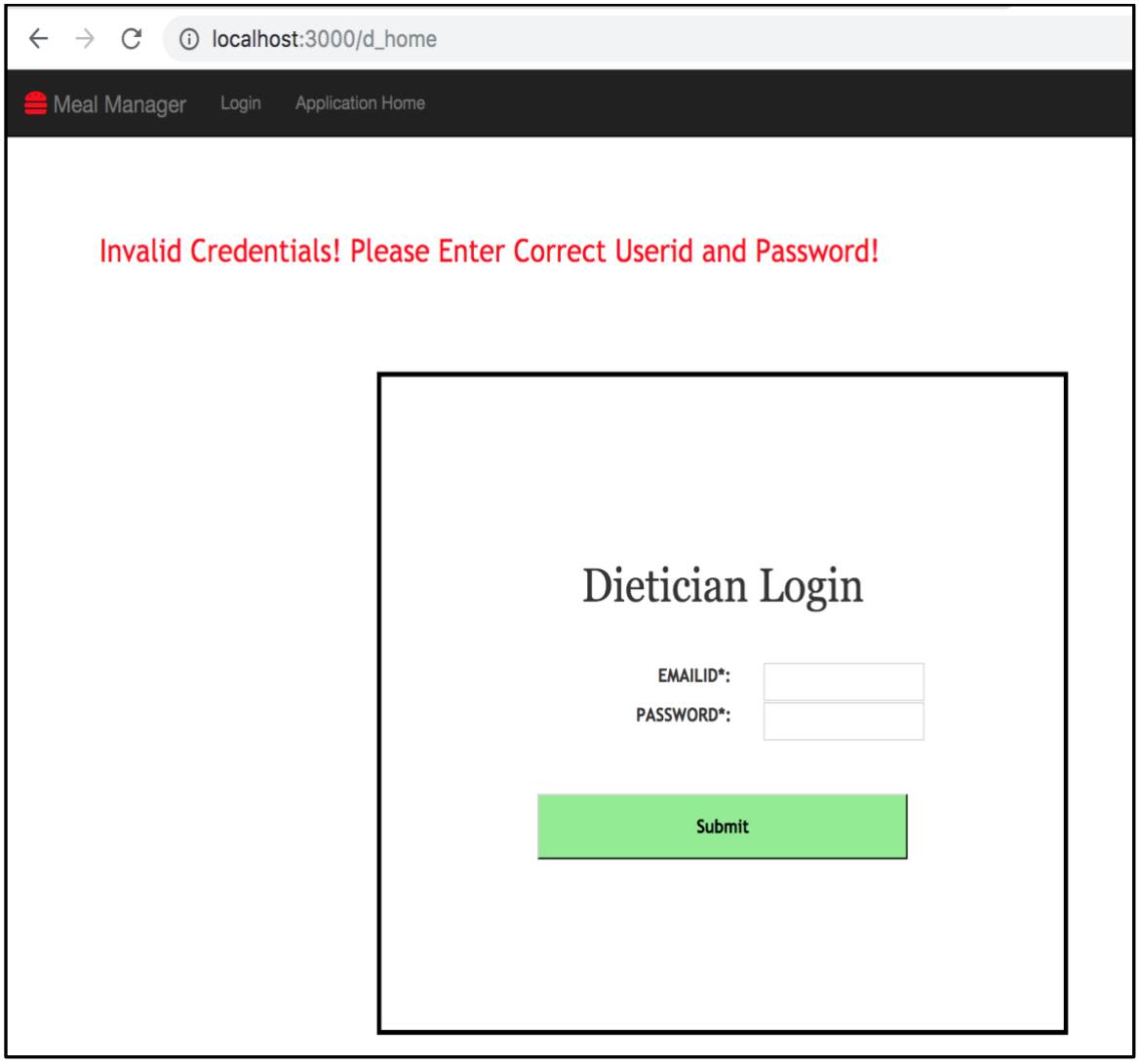
25. Joseph's request is removed from the "Update Diet Plan Requests Page". And since Kate has no more update requests, "No update diet plan requests" message is displayed.



26. On clicking on logout, Kate is logged out and login form is displayed.



27. If invalid credentials are entered in login form, error message is displayed along with login form.



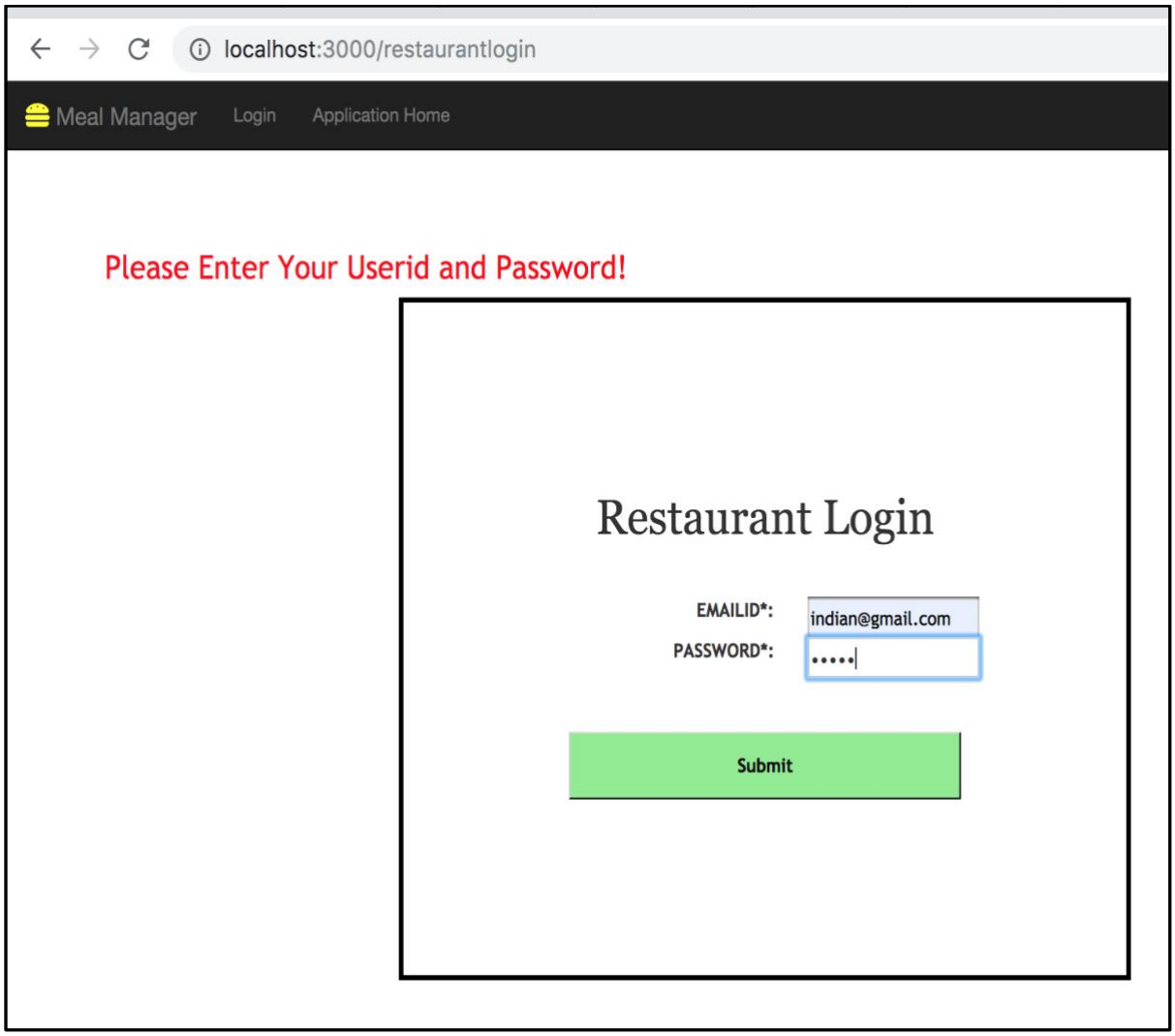
7.1.4. Restaurant

1. Restaurant Login Page:

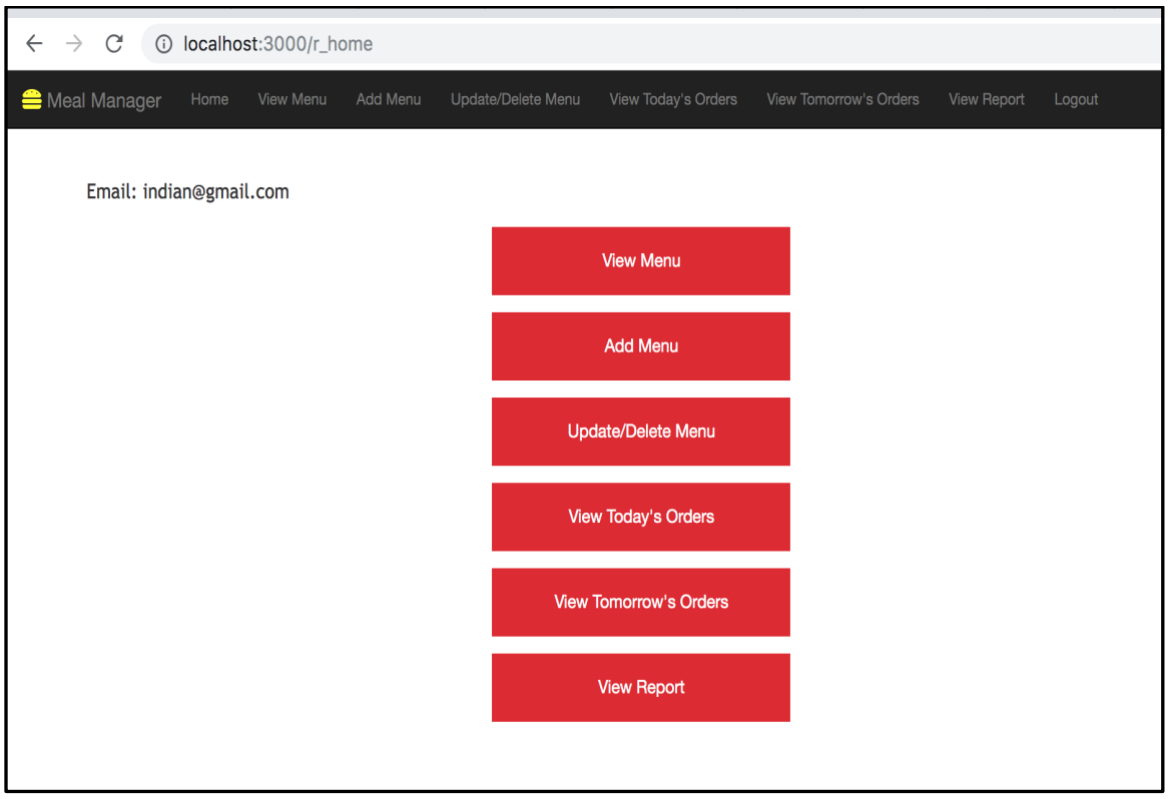
Login Page: http://localhost:3000/restaurantlogin

Login in to restaurant module with email id indian@gmail.com and password “Indian”

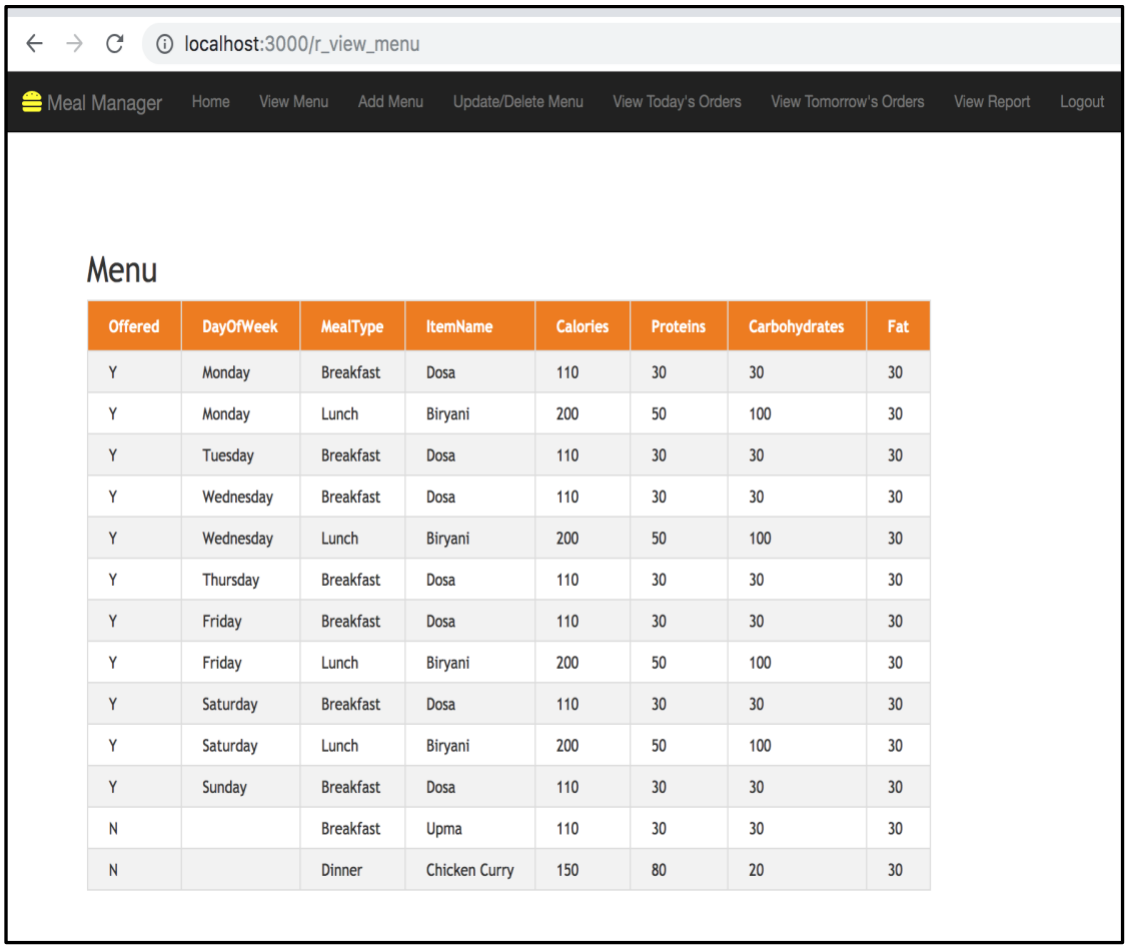
Click on submit.



2. If the login credentials are valid, restaurant home page is displayed.

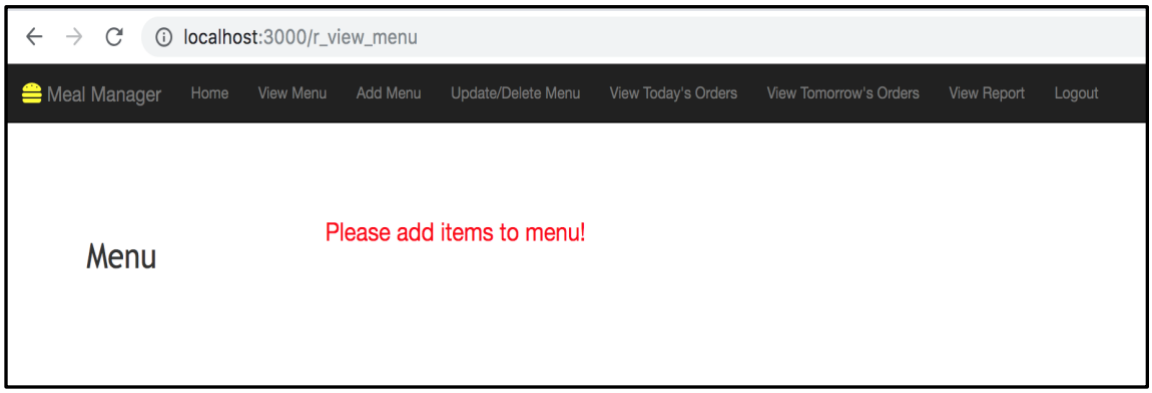


3. Click on “View Menu” button from home page or from navigation bar. The items the restaurant is offering is displayed on Menu Page, along with nutritional information. An item can be added into the menu without offering on any days. This can be because the restaurant wants to offer the item in future or because it stopped offering an item. For those records offered is N and DayOfWeek is empty.

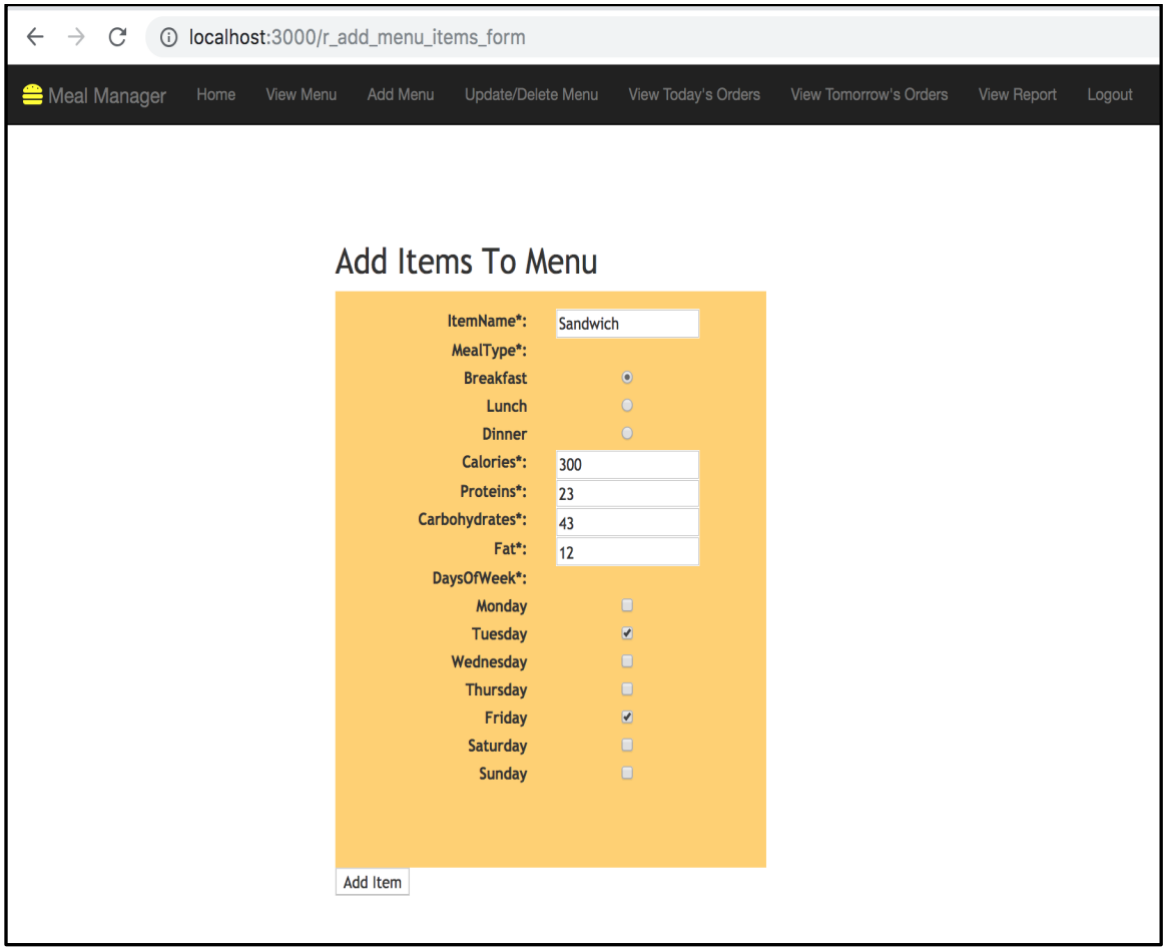


Offered	DayOfWeek	MealType	ItemName	Calories	Proteins	Carbohydrates	Fat
Y	Monday	Breakfast	Dosa	110	30	30	30
Y	Monday	Lunch	Biryani	200	50	100	30
Y	Tuesday	Breakfast	Dosa	110	30	30	30
Y	Wednesday	Breakfast	Dosa	110	30	30	30
Y	Wednesday	Lunch	Biryani	200	50	100	30
Y	Thursday	Breakfast	Dosa	110	30	30	30
Y	Friday	Breakfast	Dosa	110	30	30	30
Y	Friday	Lunch	Biryani	200	50	100	30
Y	Saturday	Breakfast	Dosa	110	30	30	30
Y	Saturday	Lunch	Biryani	200	50	100	30
Y	Sunday	Breakfast	Dosa	110	30	30	30
N		Breakfast	Upma	110	30	30	30
N		Dinner	Chicken Curry	150	80	20	30

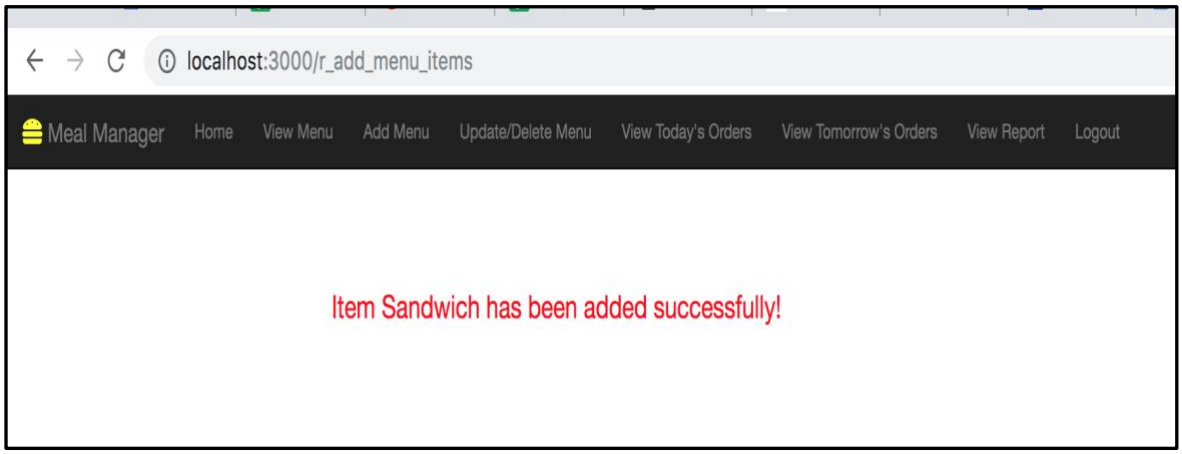
4. If the restaurant does not have any items added, “Please add item to menu!” message is displayed.



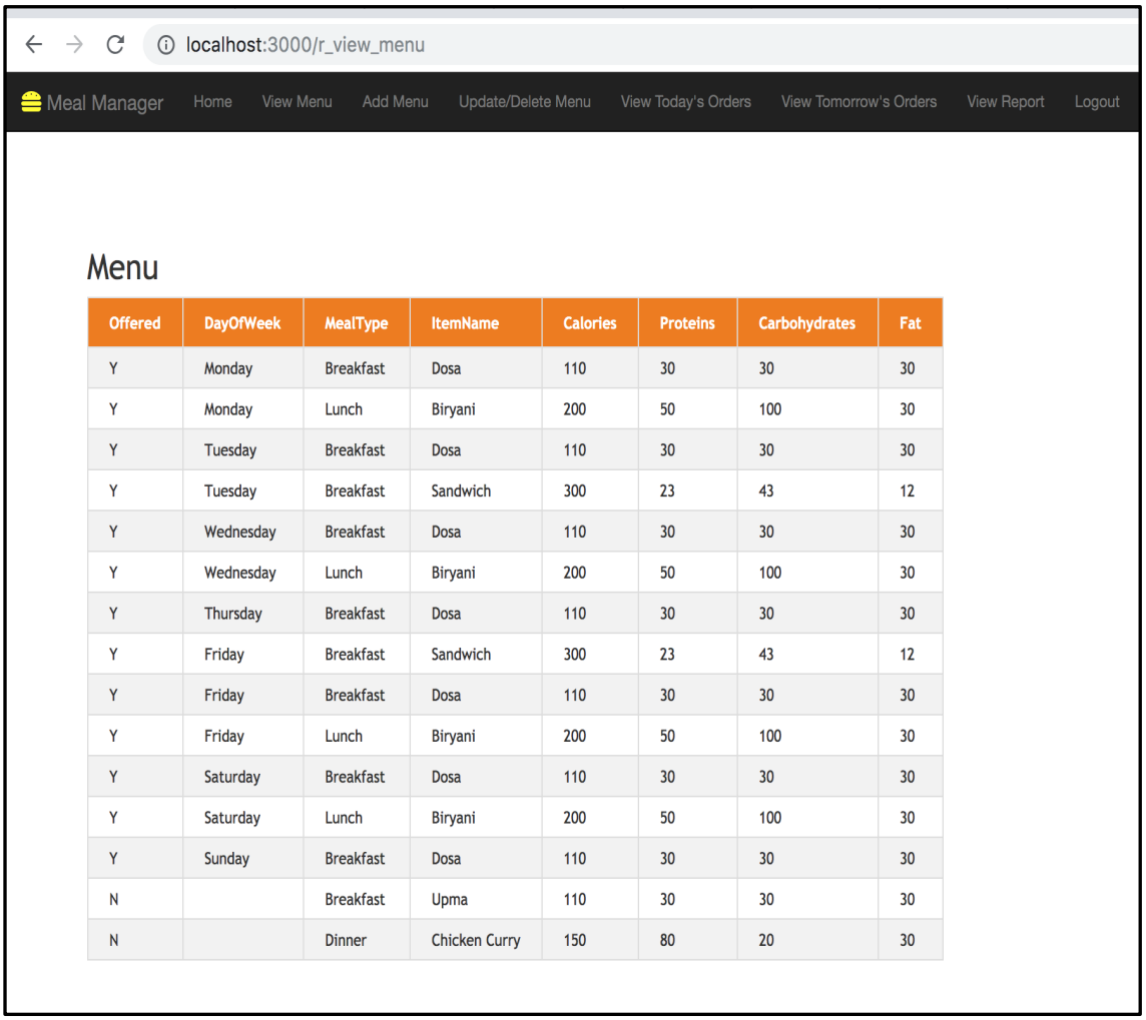
5. To add a new item to the menu, click on “Add Menu” button from home page or link from navigation bar. Add Menu form is displayed. Add all the details required and click on “Add Item” button.



6. Success message is displayed that the item Sandwich is added successfully.



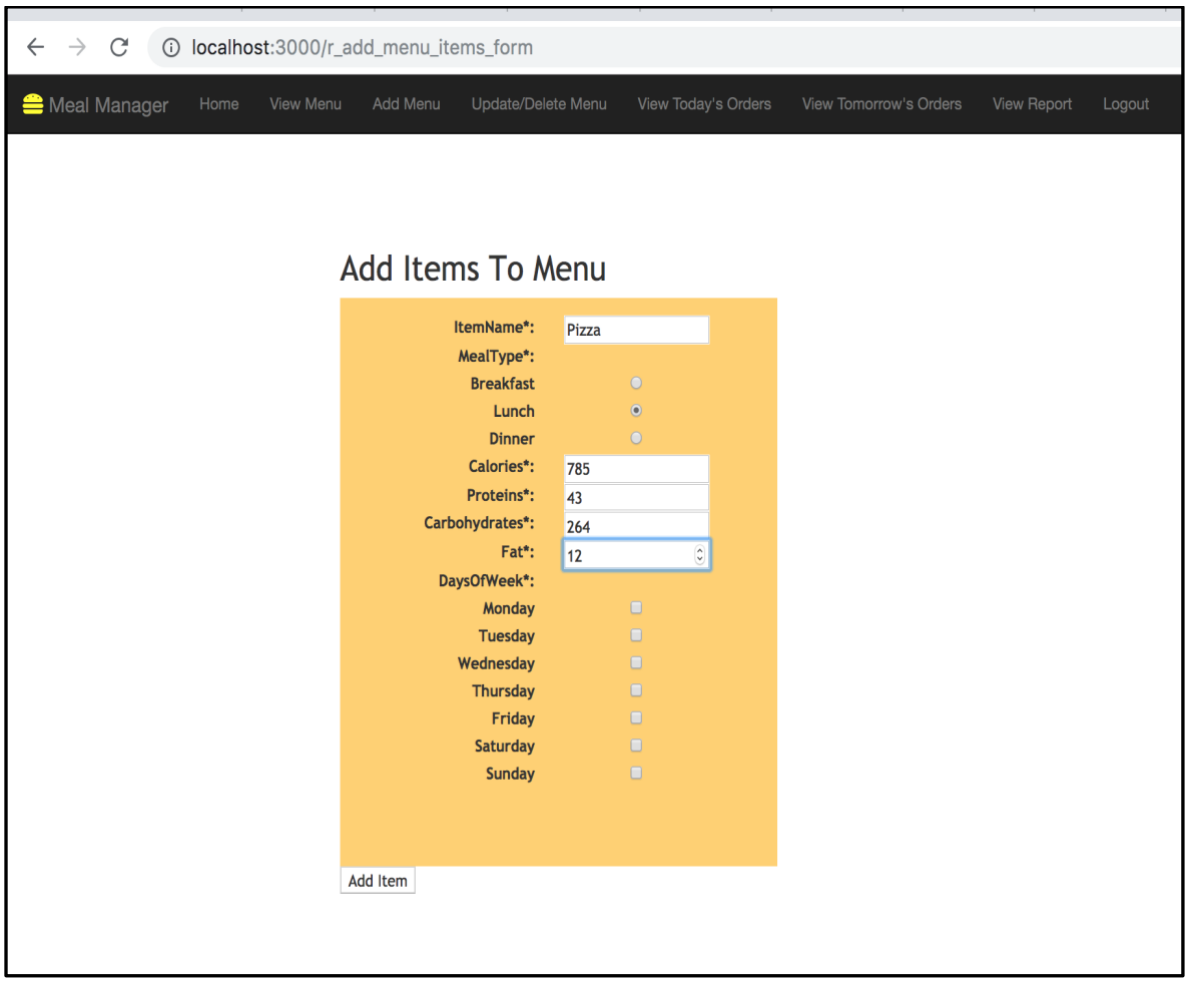
7. We can see the item added on Menu page with offered as Y since it is being offered.



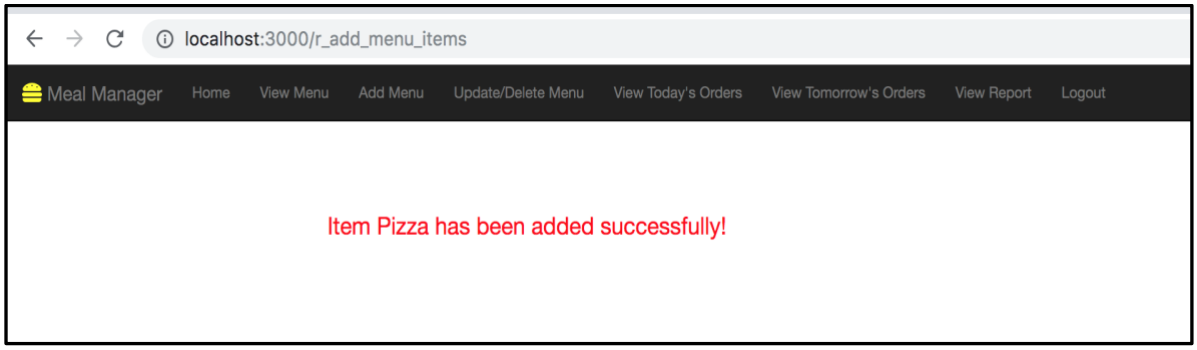
Menu

Offered	DayOfWeek	MealType	ItemName	Calories	Proteins	Carbohydrates	Fat
Y	Monday	Breakfast	Dosa	110	30	30	30
Y	Monday	Lunch	Biryani	200	50	100	30
Y	Tuesday	Breakfast	Dosa	110	30	30	30
Y	Tuesday	Breakfast	Sandwich	300	23	43	12
Y	Wednesday	Breakfast	Dosa	110	30	30	30
Y	Wednesday	Lunch	Biryani	200	50	100	30
Y	Thursday	Breakfast	Dosa	110	30	30	30
Y	Friday	Breakfast	Sandwich	300	23	43	12
Y	Friday	Breakfast	Dosa	110	30	30	30
Y	Friday	Lunch	Biryani	200	50	100	30
Y	Saturday	Breakfast	Dosa	110	30	30	30
Y	Saturday	Lunch	Biryani	200	50	100	30
Y	Sunday	Breakfast	Dosa	110	30	30	30
N		Breakfast	Upma	110	30	30	30
N		Dinner	Chicken Curry	150	80	20	30

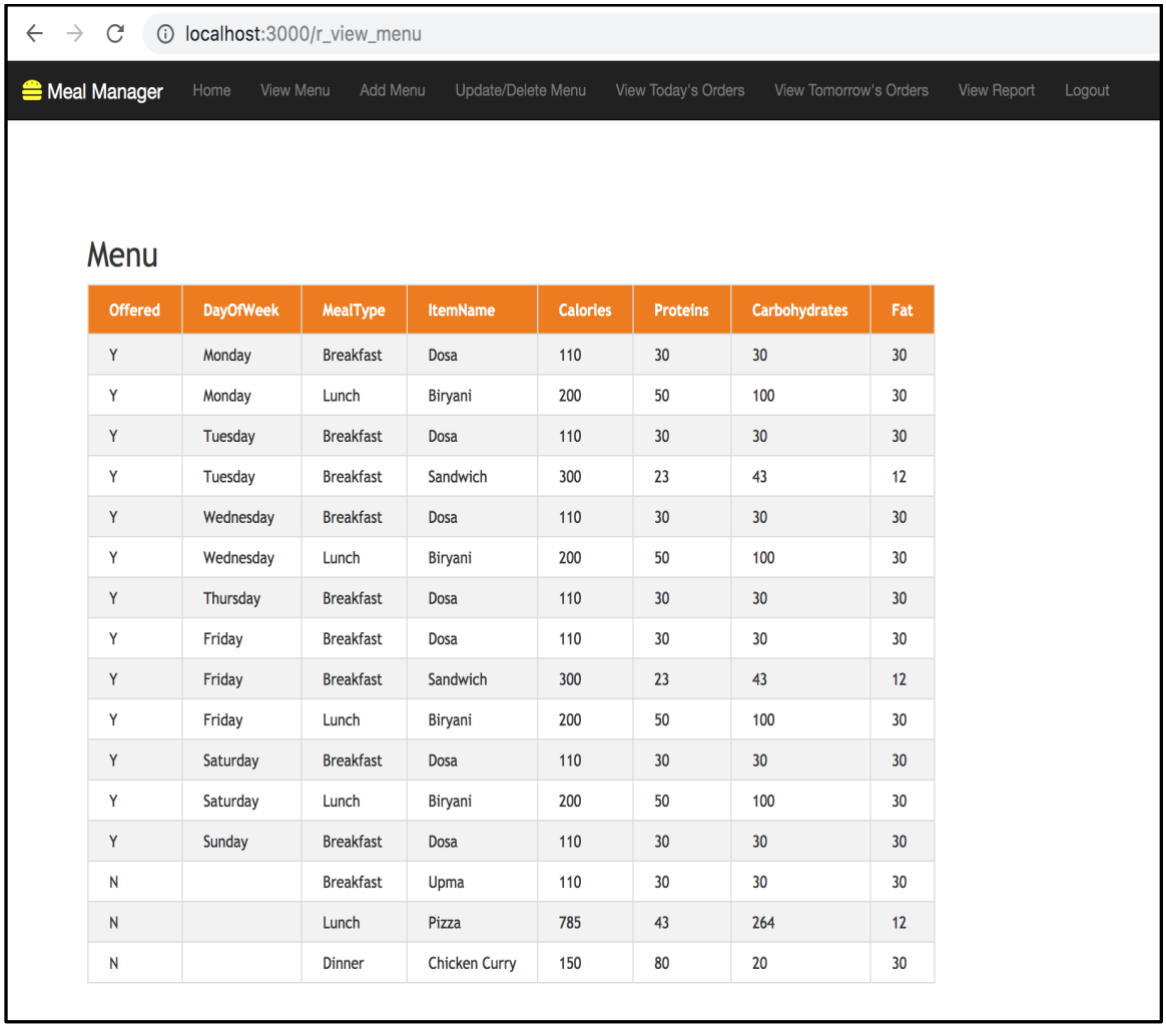
8. Let’s add an item Pizza and not offer on any day.



9. We get the success message that the item is added.



10. In Menu, we can find the item Pizza with offered as N and DayOfWeek empty.

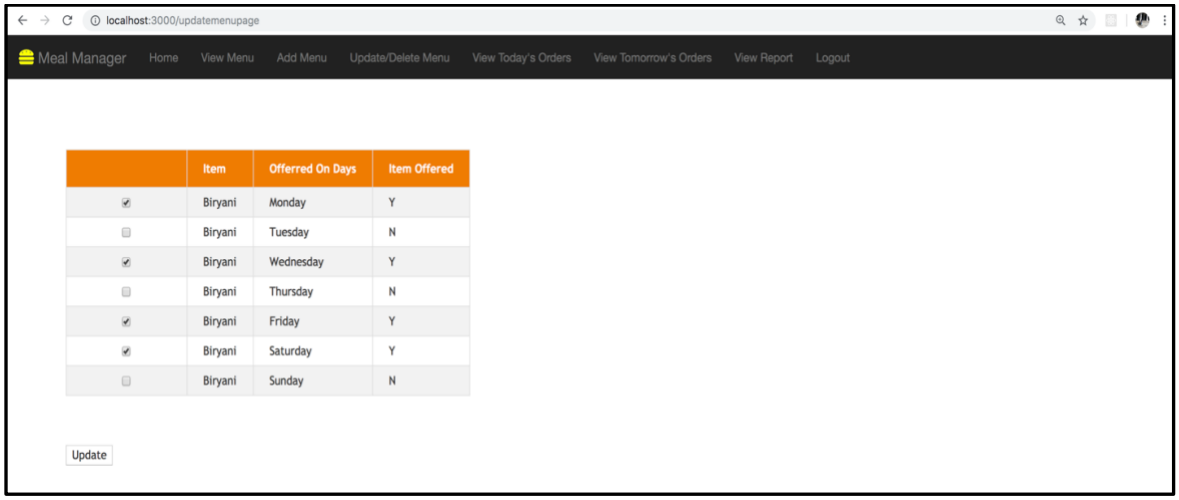


Menu

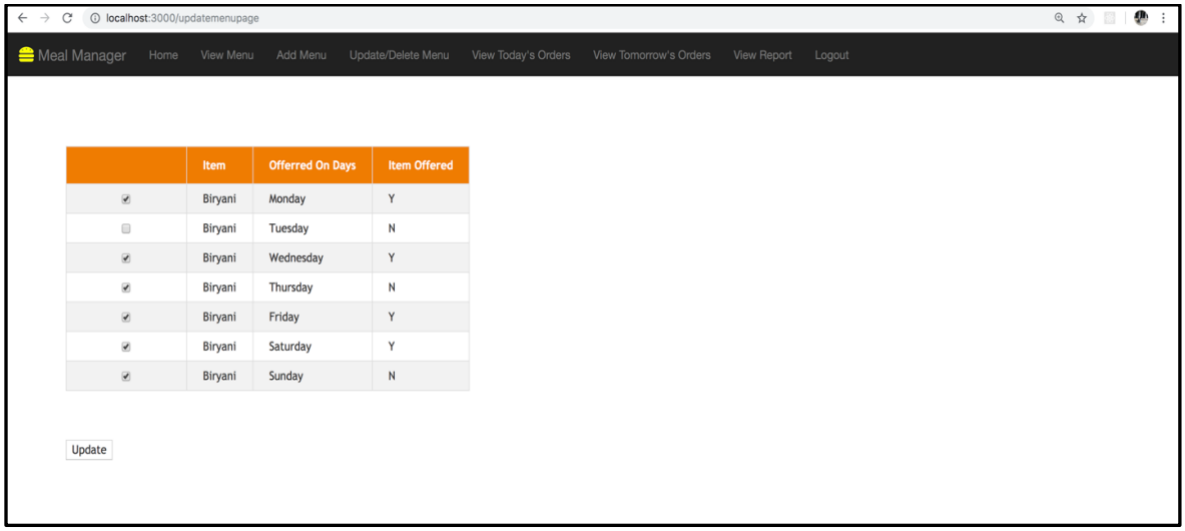
Offered	DayOfWeek	MealType	ItemName	Calories	Proteins	Carbohydrates	Fat
Y	Monday	Breakfast	Dosa	110	30	30	30
Y	Monday	Lunch	Biryani	200	50	100	30
Y	Tuesday	Breakfast	Dosa	110	30	30	30
Y	Tuesday	Breakfast	Sandwich	300	23	43	12
Y	Wednesday	Breakfast	Dosa	110	30	30	30
Y	Wednesday	Lunch	Biryani	200	50	100	30
Y	Thursday	Breakfast	Dosa	110	30	30	30
Y	Friday	Breakfast	Dosa	110	30	30	30
Y	Friday	Breakfast	Sandwich	300	23	43	12
Y	Friday	Lunch	Biryani	200	50	100	30
Y	Saturday	Breakfast	Dosa	110	30	30	30
Y	Saturday	Lunch	Biryani	200	50	100	30
Y	Sunday	Breakfast	Dosa	110	30	30	30
N		Breakfast	Upma	110	30	30	30
N		Lunch	Pizza	785	43	264	12
N		Dinner	Chicken Curry	150	80	20	30

11. Restaurant update menu page: page displayed when ‘UPDATE MENU’ is selected. Data entered after page load

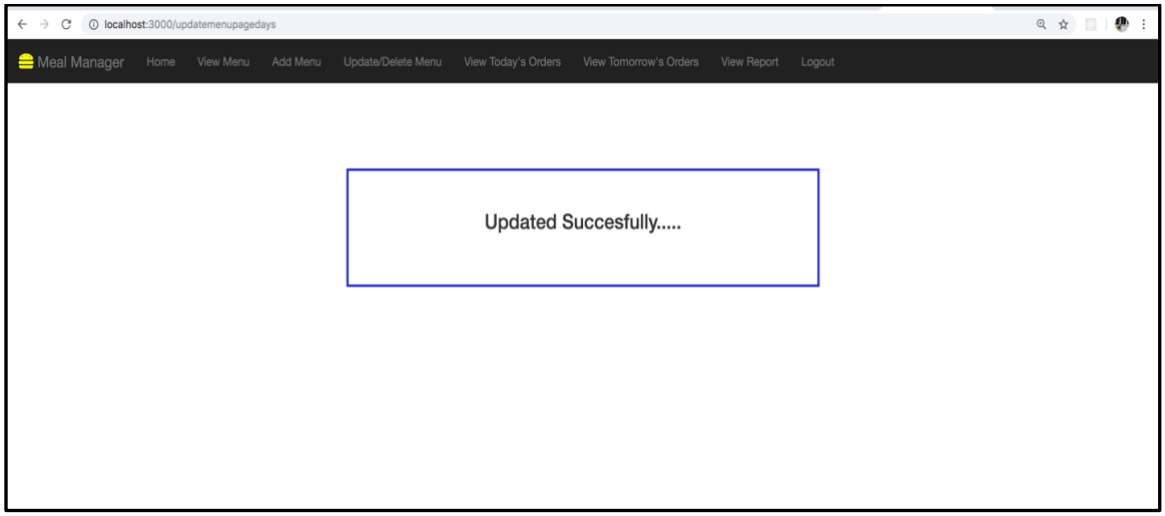
12. Item update menu page: page displayed when ‘Update selected item is selected.



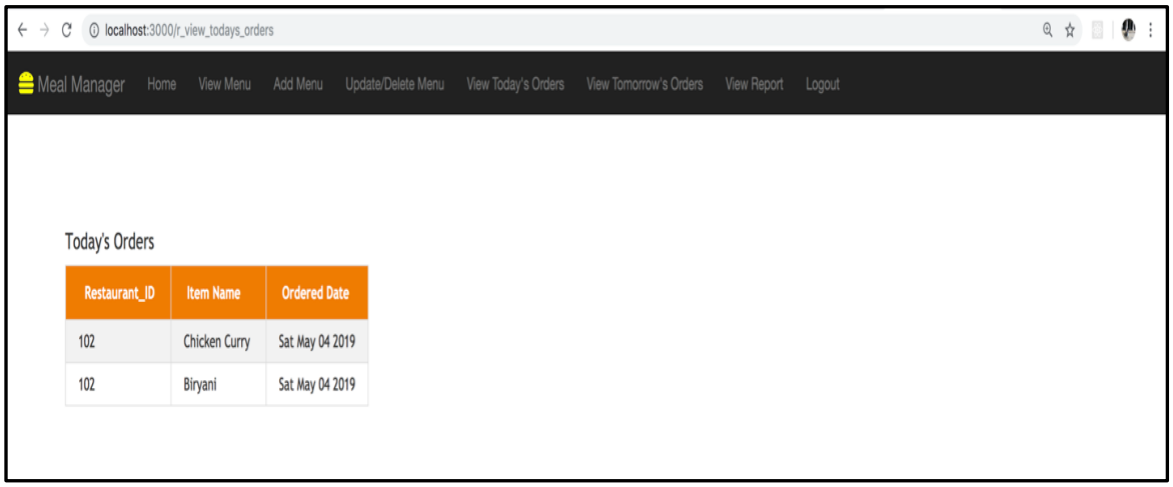
13. Check operations for Thursday and Sunday:



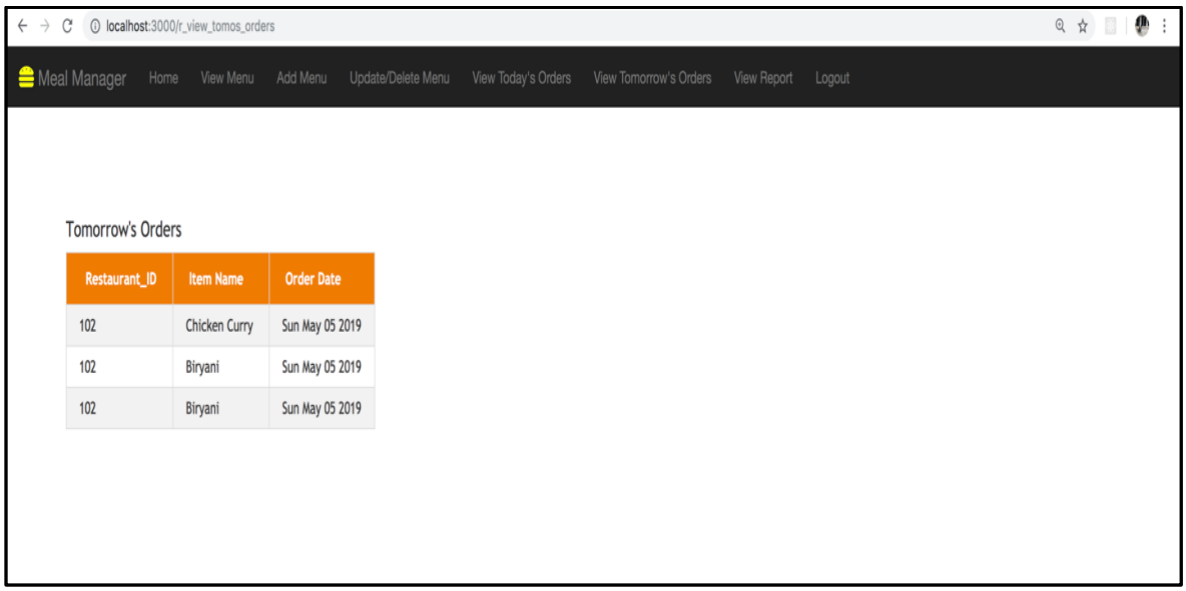
14. Click on update



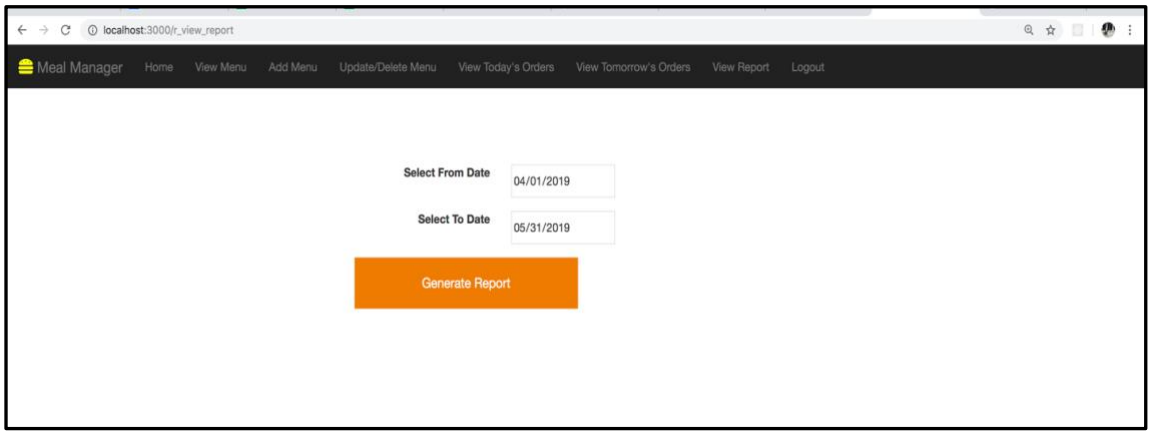
15. View todays orders page: page displayed when view todays order is clicked on header



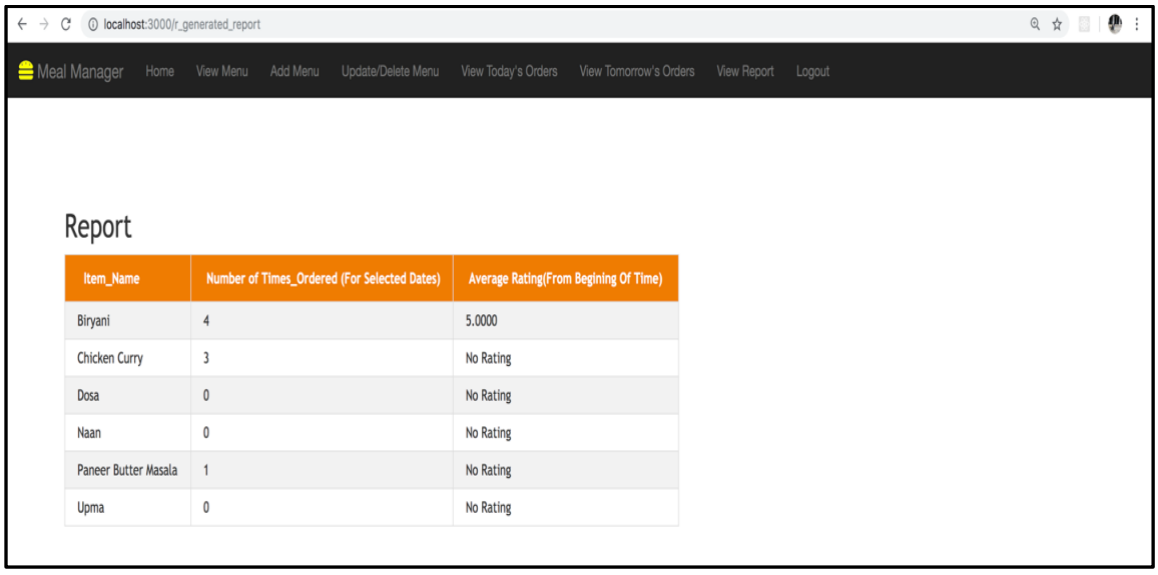
16. View tomorrows orders page: page displayed when view tomorrows order is clicked on header



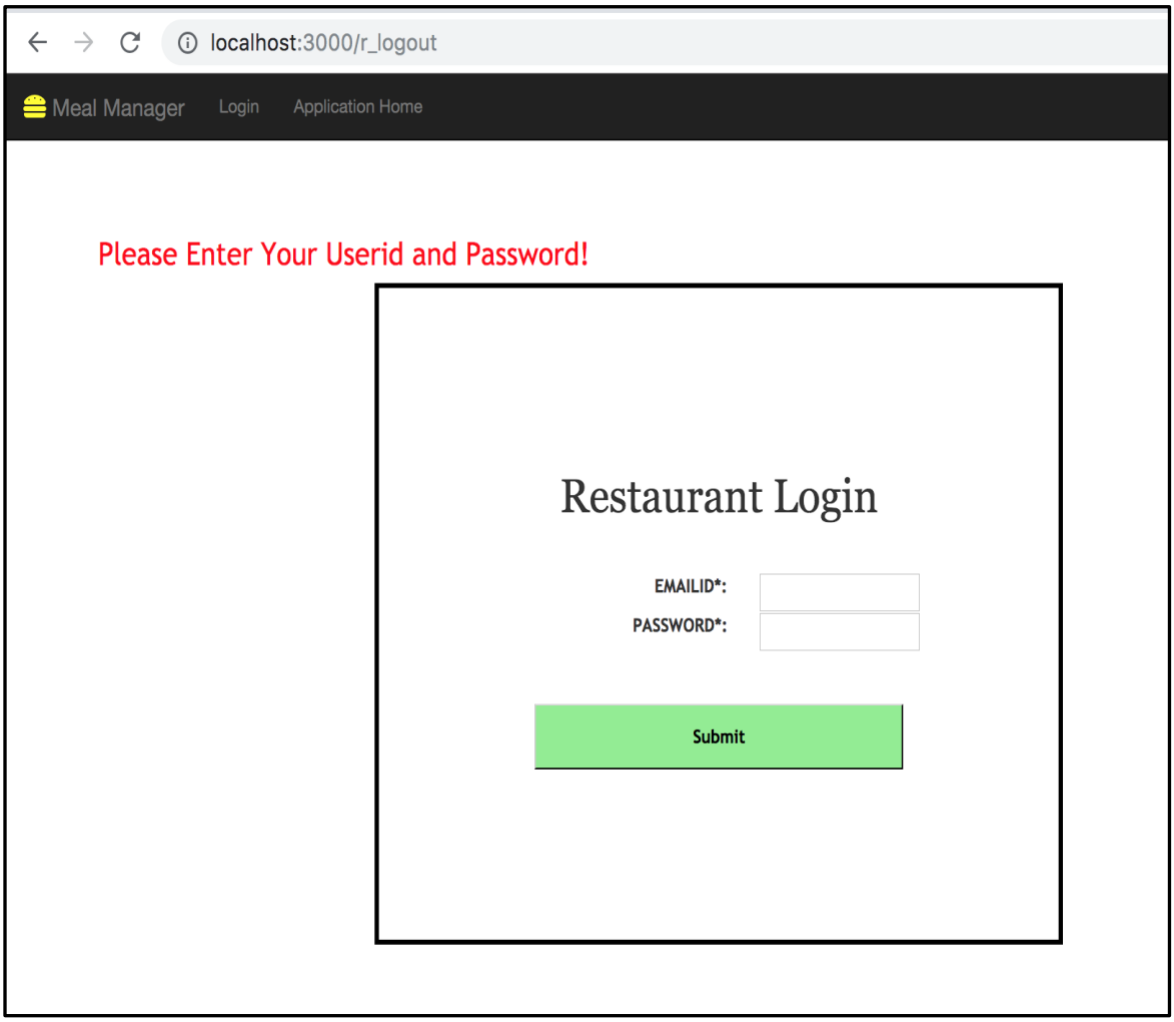
17. View report page: page displayed when view report is clicked on header. Data entered after page load



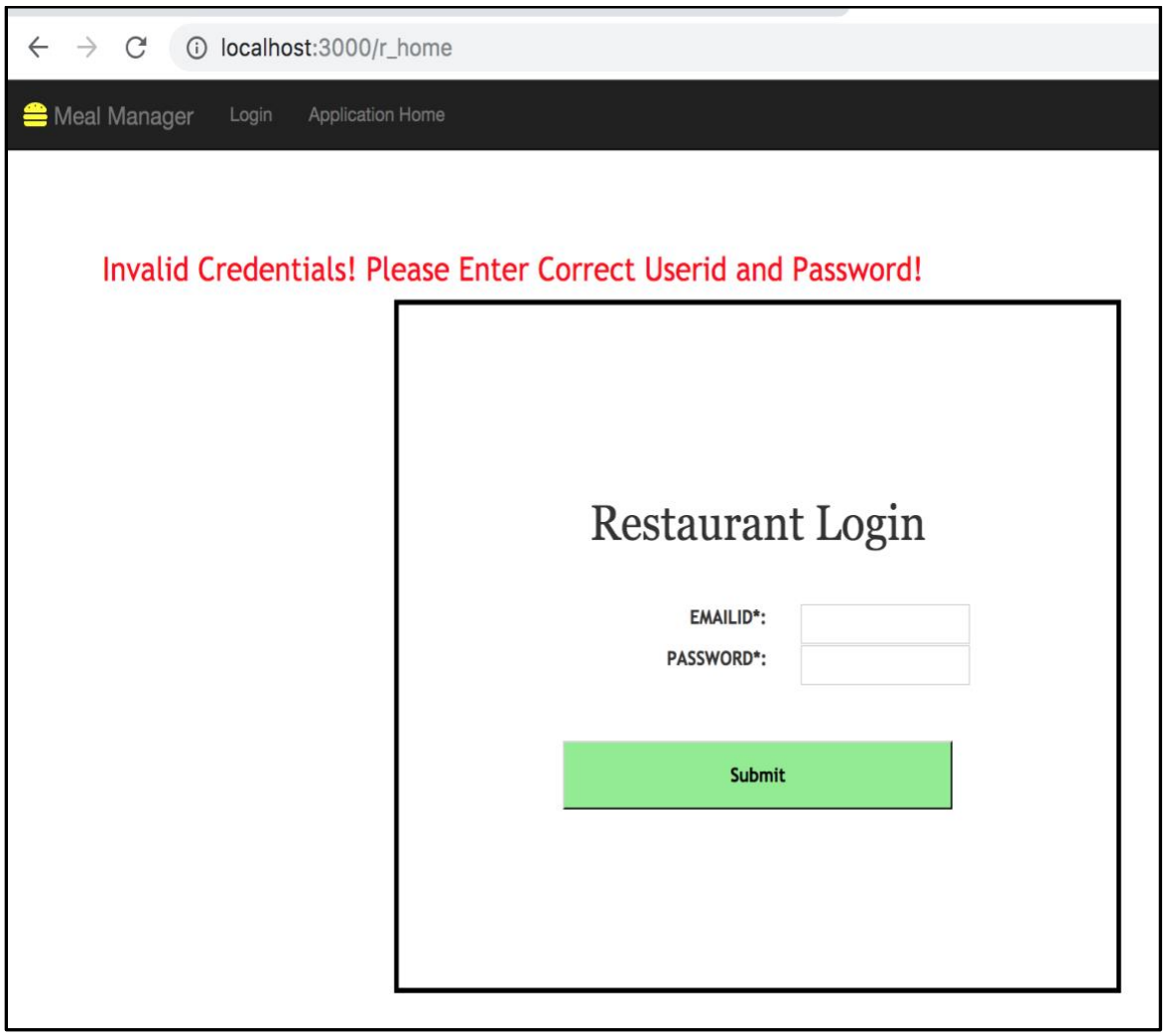
18. Page displayed when clicked on Generate Report



20. When logout is clicked in navigation bar, the restaurant is logged out and login form appears.



21. If the credentials provided are invalid, then login form is displayed with error message.



8. Major Design Decisions

For the purpose of maintaining history, we have changed some binary relations in the originally submitted ER diagram to ternary relations.

- To maintain order history for each customer, we converted orders binary relation to a ternary relation and added a new weak entity orders with order date as partial key.
- To maintain customer meal plan history i.e. to keep track of different meal plans, each customer had from the month they first registered to the application, we have created a weak entity customer billing with start date as partial key.

9. Test Cases and Test Plan Execution

Step #	Step Details	Inputs	Expected Results	Actual Results	Status
Admin					
1	Load admin page		Admin page should be loaded	Admin Page is loaded	Pass
2	Enter Email and Password fields and click Submit button	Email: admin1@mm.com Password: iamadmin1	Admin home page should be loaded	Admin home page is loaded	Pass
3	Click on Register Dietician button		Register dietician page should be loaded	Register dietician page is loaded	Pass
4	Enter Name, Email ID, Password, Experience and Qualification fields and click on Submit button	Name: dietician10 Email ID: dietician10@mm.com Password: 10101010 Experience: 10 Qualification: M.S.	Registration successful page should be displayed	Registration successful page is displayed	Pass
5	Click on Admin Home in the header		Admin home page should be loaded	Admin home page is loaded	Pass
6	Click on Register Restaurant button		Register restaurant page should be loaded	Register restaurant page is loaded	Pass
7	Enter Name, Email ID, Password, and Location fields and click on Submit button	Name: restaurant10 Email ID: restaurant10@gmail.com.com Password: 20202020	Registration successful page should be displayed	Registration successful page is displayed	Pass

		Location: San Jose			
8	Click on Logout button		admin1 should be logged out and admin login page should be displayed	admin1 is logged out and admin login page is displayed	Pass
Customer Registration					
1	Load home page of the application		Application home page should be loaded	Application home page is loaded	Pass
2	Click on Customer Registration button		Customer registration page should be displayed	Customer registration page is displayed	Pass
3	Enter Name, Email ID, Password, Date of Birth and Location fields and click on Register Me button	Name: customer10 Email ID: customer10@gmail.com Password: 30303030 Date of birth: 01/01/1995 Location: San Jose	Customer login page should be displayed	Customer login page is displayed	Pass
4	Enter Email ID and Password and click on Submit button	EmailID: customer10@gmail.com Password: 30303030	Dieticianoption should be displayed	Dieticianoption is displayed	Pass
5	Click on *Yes, I will get enrolled with a dietician* button		Dieticianyes page should be displayed	Dieticianyes page is displayed	Pass
6	Enter the details given in the input filed		Yespaymentsuccessful page should be displayed	Yespaymentsuccessful	Pass

	and click on Submit button	*Enter below lifestyle related details:* Weight: 150 Height: 175 Measurement date of height and weight: 01/01/2019 Activity level: I will have high level of physical exercise everyday Sleep: 6 -8 hrs Select the diseases that you are diagnosed with: High Cholestrol Select the ones that you are allergic to: Soy Select start date for meal plan (date > than 2 days) : 05/10/2019 *Select A Dietician:* dietician1 *Payment Details:* Enter your card number: 1111222233334444		page is displayed	

		Select type of card: Visa CVV: 9999 Expiry Date: 10/10/2020			
7	Click on Logout button		customer10 should be logged out and customer login page should be displayed	customer10 is logged out and customer login page is displayed	Pass
8	Repeat Step 2 and Step 3 with data as given in the input fields	Name: customer20 EmailID: customer20@gmail.com Password: 40404040 Date of birth: 02/02/1992 Location: Sunnyvale			Pass
9	Click on No *I will get enrolled with a dietician* button		Dieticianno page should be displayed	Dieticianno page is displayed	Pass
10	Enter the details given in the input fields and click on Submit button	Meal plan selection: PlanID: 27 Start date selection: Select start date for your meal plan: (Date of execution + 2 days) Payment details: Enter your card number:	Nopaymentsuccessful page should be displayed	Nopaymentsuccessful page is displayed	Pass

		4444333322221111 Select type of card: Credit Card CVV: 8888 Expiry Date: 10/10/2022			
11	Click on Logout button		customer20 should be logged out and customerlogin page should be displayed	customer20 is be logged out and customerlogin page is be displayed	Pass
12	Enter Email ID and Password and click on Submit button	EmailID: customer20@gmail.com Password: 40404040	"Sorry....Your Meal Plan Start Date is not tomorrow. Please Login the day before your start date." message should be displayed.	"Sorry....Your Meal Plan Start Date is not tomorrow. Please Login the day before your start date." message is be displayed.	Pass
Customer Logged In Functionalities					
Sara	**Given:** This customer is not enrolled with a dietician. Does not have a diet plan currently but has two diet plans in the past. Location of customer - "San Jose" **Assumptions Made:** Day of execution - Sunday				

	Date of Execution - May 5th 2019				
1	Customer Login: Enter UserID and Password	UserId:Sara@gmail.com Password: Sara	Input credentials will be validated and customer home page should be displayed if they are correct	Input credentials are validated and customer home page is displayed if they are correct	Pass
2	Click on Tommorow's Menu button		Personalized menu for each customer depending on his meal plan should be displayed	Personalized menu for each customer depending on his meal plan is displayed	Pass
3	Select one item in each meal type (breakfast, lunch, dinner) and respective pick up time slots and click on Place Order button	Select the following options: 1. Breakfast: Restaurant Name: Indian Item Name: Dosa Pick up time slot: 8:30A.M 2. Lunch: Restaurant Name: Malaysian Item Name: Pad Thai Noodle Pick up time slot: 1:00P.M 3. Dinner:	"Order Placed Succesfully" Message should be displayed	"Order Placed Succesfully" Message is displayed	

		Restaurant Name: Indian Item Name: Naan Pick up time slot: 8:30P.M			
4	Click on Order History button		Customer should be able to see their previous orders	Customer is shown his/her previous orders	Pass
5	Click on Current Diet Plan button		"Sorry you are not enrolled in any diet plan currently" Message should be displayed	"Sorry you are not enrolled in any diet plan currently" Message is displayed	Pass
6	Click on Diet Progress button		"Sorry you do not have any current diet progress" Message should be displayed	"Sorry you do not have any current diet progress" Message is displayed	Pass
7	Click on Diet History button		Customer should be displayed the list of previous diet plans	Customer is displayed the list of previous diet plans	Pass
8	Select one of the diet plans and click on View Details button		Customer should be displayed complete details of that diet plan along with his progress.	Customer is displayed complete details of that diet plan along with his progress.	Pass
9	Click on Rate an Item button		Customer should be displayed the	Customer is displayed the list of	Pass

			list of previously ordered items.	previously ordered items.	
10	Select one of the items and provide a rating for that item	Select the following options: Ordered date: Sun May 05 2019 Meal Type: Lunch Restaurant Name :Malaysian Item Name : Pad Thai Noodle Day of Week : Monday Pick up time slot : 1:00P.M Rating : 3	"Thank you for your rating" Message should be displayed	"Thank you for your rating" Message is displayed	Pass
11	Click on Rate a Dietician button		Customer should be displayed the list of previously enrolled dieticians.	Customer is displayed the list of previously enrolled dieticians.	Pass
12	Select one of the dieticians and provide a rating for that dietician	Select the following options: Dietician Name : dietician2 Qualification: M.S Start Date: March 30 2019	"Thank you for your rating" Message should be displayed	"Thank you for your rating" Message is displayed	Pass

		Rating : 3			
13	Click on Log Out button		Customer should be logged out and redirected to login page	Customer is logged out and redirected to login page	Pass
Stephan	**Given:** This customer is enrolled with a dietician. Has a current diet plan No diet plans in past Location of customer - "San Jose" **Assumptions Made:** Day of execution - Sunday Date of Execution - May 5th 2019				
14	Customer Login: Enter User ID and Password	UserId:Stephan@gmail.com Password: Stephan	Input credentials will be validated and customer home page should be displayed if they are correct		Pass
15	Click on Current Diet Plan button		Customer should be displayed complete details of current diet plan	Customer is displayed complete details of current diet plan	Pass
16	Click on Request change of Diet Plan button		Customer should be displayed new diet plan request page	Customer is displayed new diet plan request page	Pass
17	Enter From Date and Description of the	Enter the following details:	"Request Placed Succesfully"	"Request Placed Succesfully"	Pass

	request and click on Submit button	From Date: May 9th 2019 Description: Please increase protein in my diet plan	Message should be displayed	Message is displayed	
18	Click on Current Diet Progress button		Customer should be displayed complete details of current diet progress	Customer is displayed complete details of current diet progress	Pass
19	Click on Update Diet Progress button		Customer should be displayed update diet progress page	Customer is displayed update diet progress page	Pass
20	Enter the details given in input fields and click on submit button	Enter the following details: Log Date: May 1st 2019 Did you follow your diet for breakfast: Yes Did you follow your diet for lunch: Yes Did you follow your diet for dinner: Yes Any Additional Intake: Nothing	"Update Succesfully" Message should be displayed	"Update Succesfully" Message is displayed	Pass
21	Click on Log Out button		Customer should be logged out and redirected to login page	Customer is logged out and redirected to login page	Pass
Restaurant Logged in Functionalities					

	Given: Restaurant considered in this section is already registered and has items added in the menu				
1	Load home page of the application		Application home page should be loaded	Application home page is loaded	Pass
2	Click on Restaurant Login button		Restaurant login page should be displayed	Restaurant login page is displayed	Pass
3	Enter Email ID and Password and click on Submit button	EmailID: Indian@gmail.com Password: Indian	r_home page should be displayed.	r_home page is be displayed.	Pass
4	Click on View Menu button		Item names, day in which it is offered, nutritional in formation, offerred currently or not should be displayed	Item names, day in which it is offered, nutritional in formation, offerred currently or not is displayed	Pass
5	Click on Add Menu button from the header and enter the details given in the input field and click add item	Item Name: Item10 Breakfast Calories: 100 Carbohydrates:100 Protein: 80 Fat: 50 Monday	"Item Item10 has been added successfully!" message should be displayed	"Item Item10 has been added successfully!" message is displayed	Pass

6	Click on Update/Delete Menu button from the header and enter the details given in the input field and click on Update Selected Item button	List of items offered : select Item10	Item10 should be displayed with Item offered as Y for Monday and N for other six days	Item10 is displayed with Item offered as Y for Monday and N for other six days	Pass
7	Select the checkbox as given in the input field and click on Update button	Day: Thursday	Updated successfully' message should be displayed	Updated successfully' message is displayed	Pass
8	Click on View Todays Orders button from the header		Any order placed with data as [Date of execution - 1] should be displayed	Any order placed with data as [Date of execution - 1] is displayed	Pass
9	Click on View Tomorrows Orders button from the header		Any order placed with data as [Date of execution] should be displayed	Any order placed with data as [Date of execution] is displayed	Pass
10	Click on View Report button from the header		r_view_report page should be displayed	r_view_repor t page is displayed	Pass
11	Select the dates as given in the input and click on Generate report	From Date: 01/01/2019 To Date: Date of execution	All the items offerred by the restauarant should be displyed along with the number of times each item is ordered and ratings if any	All the items offerred by the restauarant is displyed along with the number of times each item is ordered and ratings if any	Pass

12	Click on Logout button		Restaurant should be logged out of the application and restaurant login page should be displayed	Restaurant is logged out of the application and restaurant login page is displayed	Pass
Dietician Logged IN Functionalities					
1	Load Dietician Login Page at http://localhost:3000/dieticianlogin		Dietician login page should be loaded	Dietician login page is loaded	Pass
2	Enter Emailid and Password fields and click Submit button	Email: kate@gmail.com Password: Kate	Dietician home page should be loaded	Dietician home page is loaded	Pass
3	Click on "View Current Customers" button on homepage or in navigation-bar		It should show the list of customers for who the current date is less than or equal to (diet_start_date+30) days. If the customer already has a diet plan for this month, and has renewed his membership for a new diet plan next month, then 2 records should come for this customer. 2 records for James - 1 for existing diet plan, and another for new diet plan where diet start	2 records for James - 1 for existing diet plan, and another for new diet plan where diet start date is be in future and 1 record for Joseph is be seen.	Pass

			date will be in future and 1 record for Joseph will be seen.		
4	Select the radio-button of the customer whose diet plan you want to view and click on "View Diet Plan" button	Select radio-button for customerId=1021 and dietStartDate<today and click on "View Diet Plan" button	It should show the list of all diet plans the customer=1021 has.	3 diet plan list for customer 1021 is seen	Pass
5	Select 1 of the diet plans and click on "View Diet Plan Progress" button	Select radio-button for customerId=1021 and select the middle radio button and click on "View Diet Plan Progress" button	Now the diet plan details to be followed by CustomerId=1021 will be shown for all days in that particular diet plan. If the customer has enetered any diet progress, that information will also be seen. If the customer has not enetered any progress for a particular day, "No Data" will be seen for DietFollowed column. If the customer has added any additional intake, it will be displayed.	Diet plan details to be followed by CustomerId=1021 is be shown for all 30 days in the selected diet plan	Pass
6	Go to "New Diet Plan Requests" in the		Diet plan requests page will all new	1 record for customerid=1	Pass

	navigation bar or "New Diet Plan Requests" button from home-page		diet plan requests will be seen. 1 record for customerid=1021 with diet start date in future will be seen	021 with diet start date in future is seen	
7	Select the radio-button of the customerid=1021 and click on "Create New Diet Plan" button.		Diet Plan Details for the 30 days must be seen for customerid=1021. The customer's health details like age, height, weight, activity level, sleep, allergies, diseases information for the new diet plan must be seen. Diet plan details like calories, proteins are seen for entire month, except for day-7	Customer health details is seen. Diet plan details is seen for all 30 days. Day 7 has details empty.	Pass
8	To continue creating new diet plan select day 7 for customerid=1021 and click on "Create New Diet Plan" button at the buttom of the page		The customer's health details like age, height, weight, activity level, sleep, allergies, diseases information for the new diet plan must be seen. The form to add diet details will be displayed.	Customer health details is seen. Add diet plan details form is displayed	Pass
9	In "Add Diet Plan Details" page, add all	Breakfast: Calories:201,	After entering the details, when the	Diet Plan Details added	Pass

	the information and submit to add the diet details for the seected day. After entering the details, when click on "Add Diet Plan Details" button.	Proteins: 11, Carbohydates: 12, fat: 13 Lunch: Calories:301, Proteins: 21, Carbohydates: 22, fat: 23 Dinner: Calories:401, Proteins: 31, Carbohydates: 32, fat: 33	"Add Diet Plan Details" button is clicked, the diet plan details will be added in the database. The "DIet Plan Details" page with the message "Diet Plan Details added successfully will be displayed". The added diet plan details can be seen for Day 7 for customerid=1021	successfully is displayed and diet plan details added can be seen for day 7	
10	To edit the created diet plan for Day 7, select day 7 for customerid=1021 and click on "Create New Diet Plan" button at the button of the page	Breakfast: Calories:101, Proteins: 11, Carbohydates: 15, fat: 16 Lunch: Calories:201, Proteins: 21, Carbohydates: 25, fat: 26 Dinner: Calories:301, Proteins: 31, Carbohydates: 35, fat: 36	The page will be displayed with existing deit plan details information. After updating the diet plan details, when the "Add Diet Plan Details" button is clicked, the diet plan details will be updated in the database. The "DIet Plan Details" page with the message "Diet Plan Details added successfully will be displayed". The added diet plan details can be seen	Customer health details is displayed. The add diet plan details form with prefilled information is displayed. After submitting the form, "DIet Plan Details" page with the message "Diet Plan Details added successfully is displyed with updated	Pass

			for Day 7 for customerid=1021	details for day 7.	
11	If the dietician is satisfied with the diet plan for all 30 days, click on "Finalize Diet Plan" button		"Diet plan for the following customer has been finalized! " with customer name will be displayed.	"Diet plan for the following customer has been finalized! " with customer name is displayed.	Pass
12	Go to "New Diet Plan Requests" in the navigation bar or "New Diet Plan Requests" button from home-page		The new diet plan request for customerid=1021 will not be seen	New diet plan request for customerid=1021 is removed	Pass
13	Go to "New Diet Plan Requests" in the navigation bar or "New Diet Plan Requests" button from home-page		If there are no new diet plan requests, "No new diet plan requests!" message willl be seen.	"No new diet plan requests!" message is displayed.	Pass
14	Go to "Update Diet Plan Requests" in the navigation bar or "Update Diet Plan Requests" button from home-page		1 record for customerid=1022 with dietupdatestartdate in future will be seen.	1 record for customerid=1022 with dietupdatestartdate in future is seen.	Pass
15	Select the radio button for customerid=1022 and click "Update Diet Plan" button.		The diet plan details for the selected customer and diet plan for all calendardates >= dietplanupdatestartdate will be displayed. The customer's health	The diet plan details for the selected customer and diet plan for all calendardates >= dietplanupdat	Pass

			details like age, height, weight, activity level, sleep, allergies, diseases information for the diet plan must be seen.	estartdate along with customer health details is displayed.	
16	select the radio button for day 10 to edit the diet plan details for day 10 for customerid=1022		The update diet plan details with prefilled values of the existing diet plan will be displayed for customerid=1022. The customer's health details like age, height, weight, activity level, sleep, allergies, diseases information for the diet plan must be seen.	The previous diet diet plan details is present in the form and customer health information is also displayed.	Pass
17	Enter the diet plan details for day 10 for customerid=1022 and click on "Update Diet Plan Details" button.	Breakfast: Calories:101, Proteins: 11, Carbohydates: 15, fat: 16 Lunch: Calories:201, Proteins: 21, Carbohydates: 25, fat: 26 Dinner: Calories:301, Proteins: 31,	The diet plan details will be updated in the database for customerid=1022 for day 10. The "DIet Plan Details" page with the message "Diet Plan Details updated successfully will be displayed". The updated diet plan details can be seen	The "DIet Plan Details" page with the message "Diet Plan Details updated successfully is displayed and diet plan details for day 10 can be seen.	Pass

		Carbohydates: 35, fat: 36	for Day 10 for customerid=1022		
18	If the dietician is satisfied with the diet plan for all days, click on "Finalize Diet Plan" button		"Diet plan for the following customer has been finalized! " with customer name will be displayed.	"Diet plan for the following customer has been finalized! " with customer name is displayed.	Pass
19	Go to "Update Diet Plan Requests" in the navigation bar or "Update Diet Plan Requests" button from home-page		The update diet plan request for customerid=1022 will not be seen	update diet plan request for customerid=1022 is removed.	Pass
20	Go to "Update Diet Plan Requests" in the navigation bar or "Update Diet Plan Requests" button from home-page		If there are no new diet plan requests, "No new update diet plan requests!" message will be seen.	"No new update diet plan requests!" message is seen.	Pass
21	Click on Logout button		Dietician Kate should be logged out and dietician login page should be displayed	Dietician Kate is logged out and dietician login page is displayed	Pass

10. Project Post Mortem

10.1. Issues Uncovered

1. We realized we had to maintain customer meal plan history and order history, to provide some functionalities we listed. So we converted some binary relations to ternary relations in ER diagram.
2. Creating an additional unique column in case of composite keys helps in easy maintenance of foreign keys. However, the new key affects the normalization process and resulted in further decomposition of the tables. Consequently, we removed the unique identifier and used the composite keys together.

10.2. Potential Improvements

1. Add images of items in menu page
2. Add recipes for items, so that customers will have an option to either order the item from restaurant or even cook by themselves if they have time.
3. Based on the ratings received to restaurant-items best restaurants can be announced on a monthly basis.

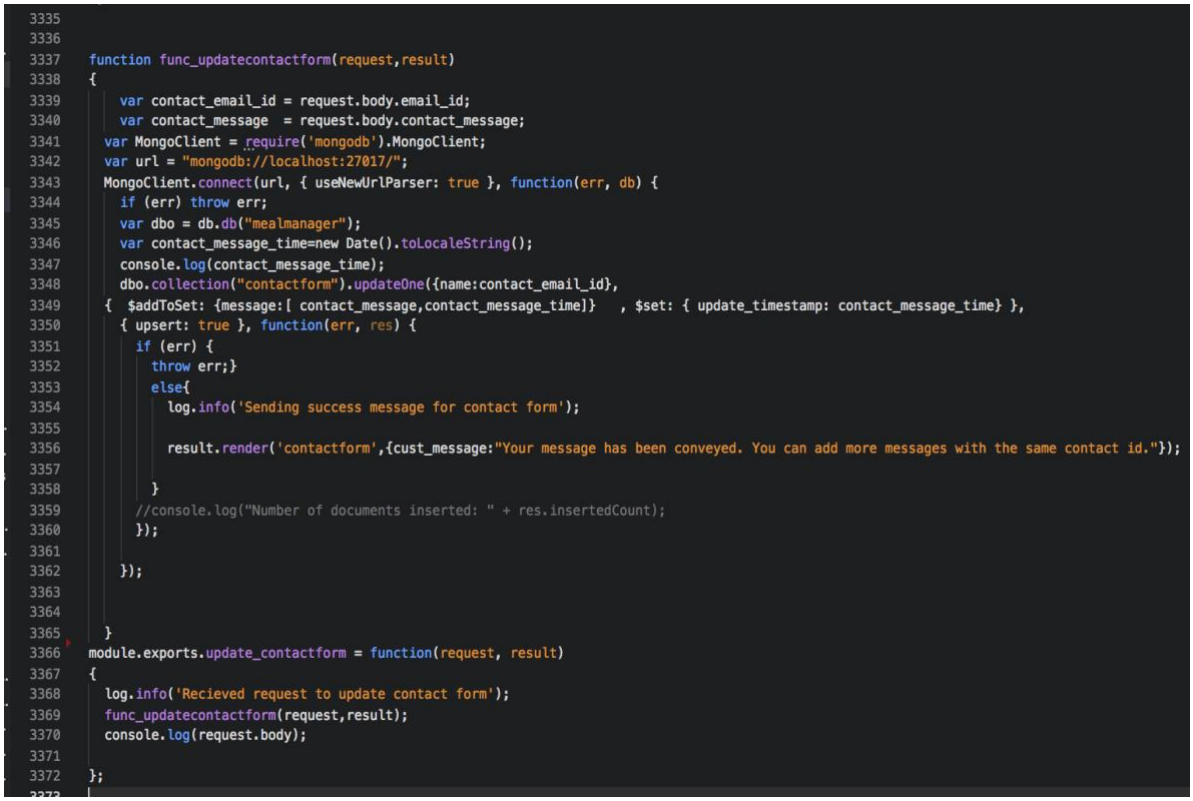
11. Mongo DB (Extra Credit)

We have used the nosql database MongoDB in our application to store the information from contact forms. Any user can interact with the admin through contact forms. They can send their grievances, concerns, business opportunities. The user must give a contact id and message. Whenever the user submits multiple messages over a period of time, the messages and the message timestamp will be stored for the corresponding contact id in mongo db. There is no limit on the message length.

The admin can view the messages he has got from users using “Contact Messages” button. He will get all contact ids that have sent him a message along with the timestamp of the last message from that particular user. He can further view all messages from the user, and decide how he wants to address the message.

11.1. MongoDB application code screenshots

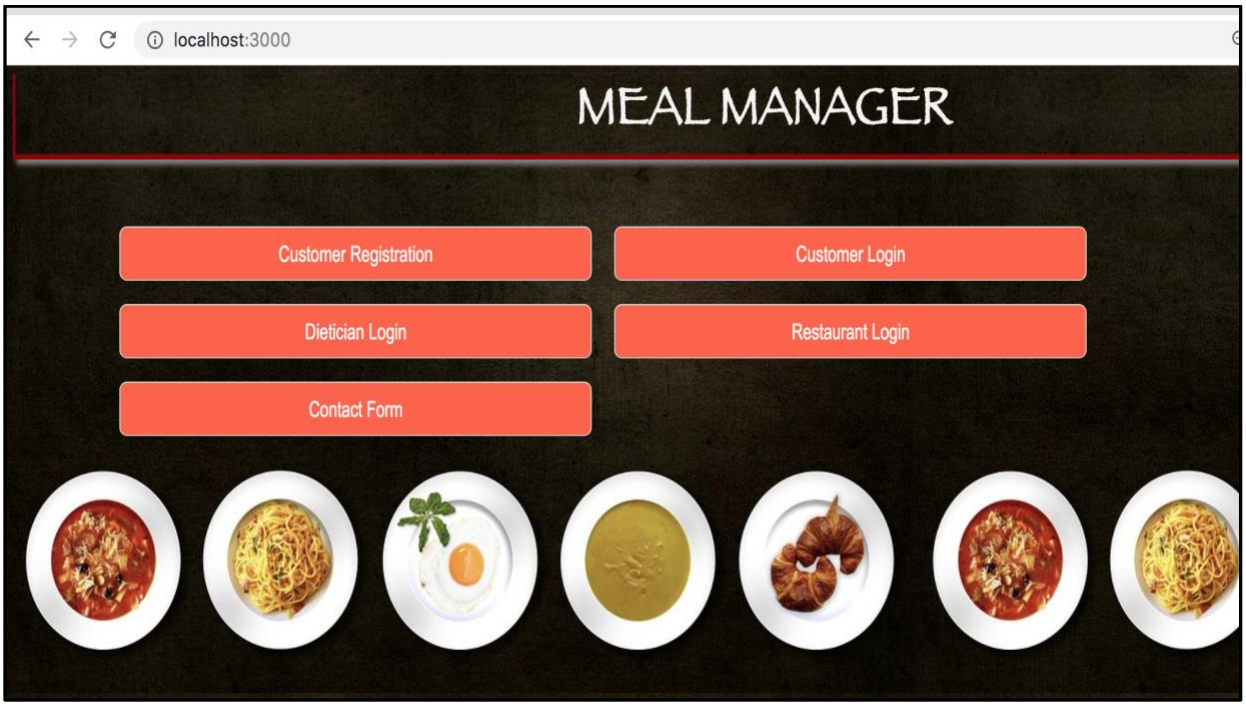
Screenshot of mongo db code used to update contact form.



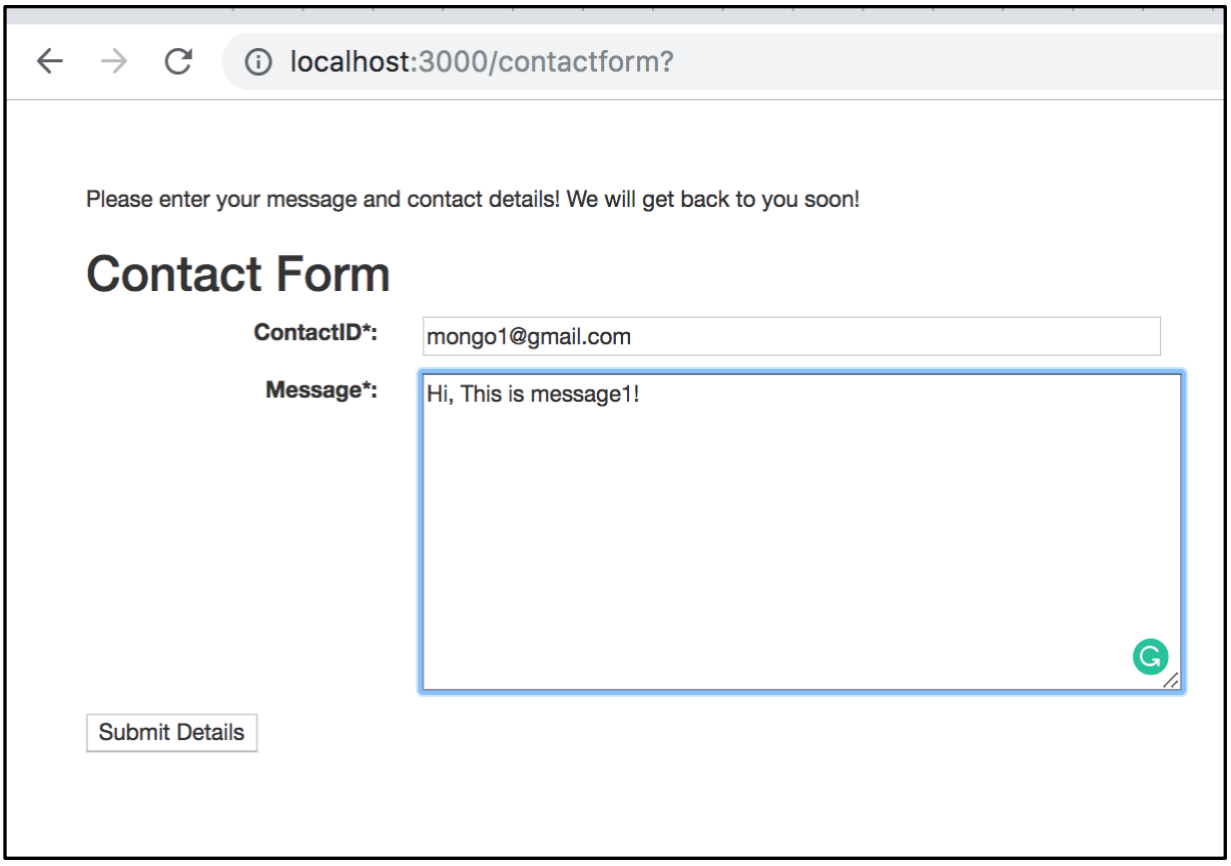
```
3335
3336
3337  function func_updatecontactform(request,result)
3338  {
3339      var contact_email_id = request.body.email_id;
3340      var contact_message  = request.body.contact_message;
3341    var MongoClient = require('mongodb').MongoClient;
3342    var url = "mongodb://localhost:27017/";
3343    MongoClient.connect(url, { useNewUrlParser: true }, function(err, db) {
3344      if (err) throw err;
3345      var dbo = db.db("mealmanager");
3346      var contact_message_time=new Date().toLocaleString();
3347      console.log(contact_message_time);
3348      dbo.collection("contactform").updateOne({name:contact_email_id},
3349    {  $addToSet: {message:[ contact_message,contact_message_time]}   , $set: { update_timestamp: contact_message_time} },
3350      { upsert: true }, function(err, res) {
3351        if (err) {
3352          throw err;}
3353          else{
3354            log.info('Sending success message for contact form');
3355
3356            result.render('contactform',{cust_message:"Your message has been conveyed. You can add more messages with the same contact id."});
3357
3358          }
3359        //console.log("Number of documents inserted: " + res.insertedCount);
3360        });
3361
3362      });
3363
3364
3365    }
3366  module.exports.update_contactform = function(request, result)
3367  {
3368    log.info('Recieved request to update contact form');
3369    func_updatecontactform(request,result);
3370    console.log(request.body);
3371
3372  };
```

11.2. MongoDB application screenshots

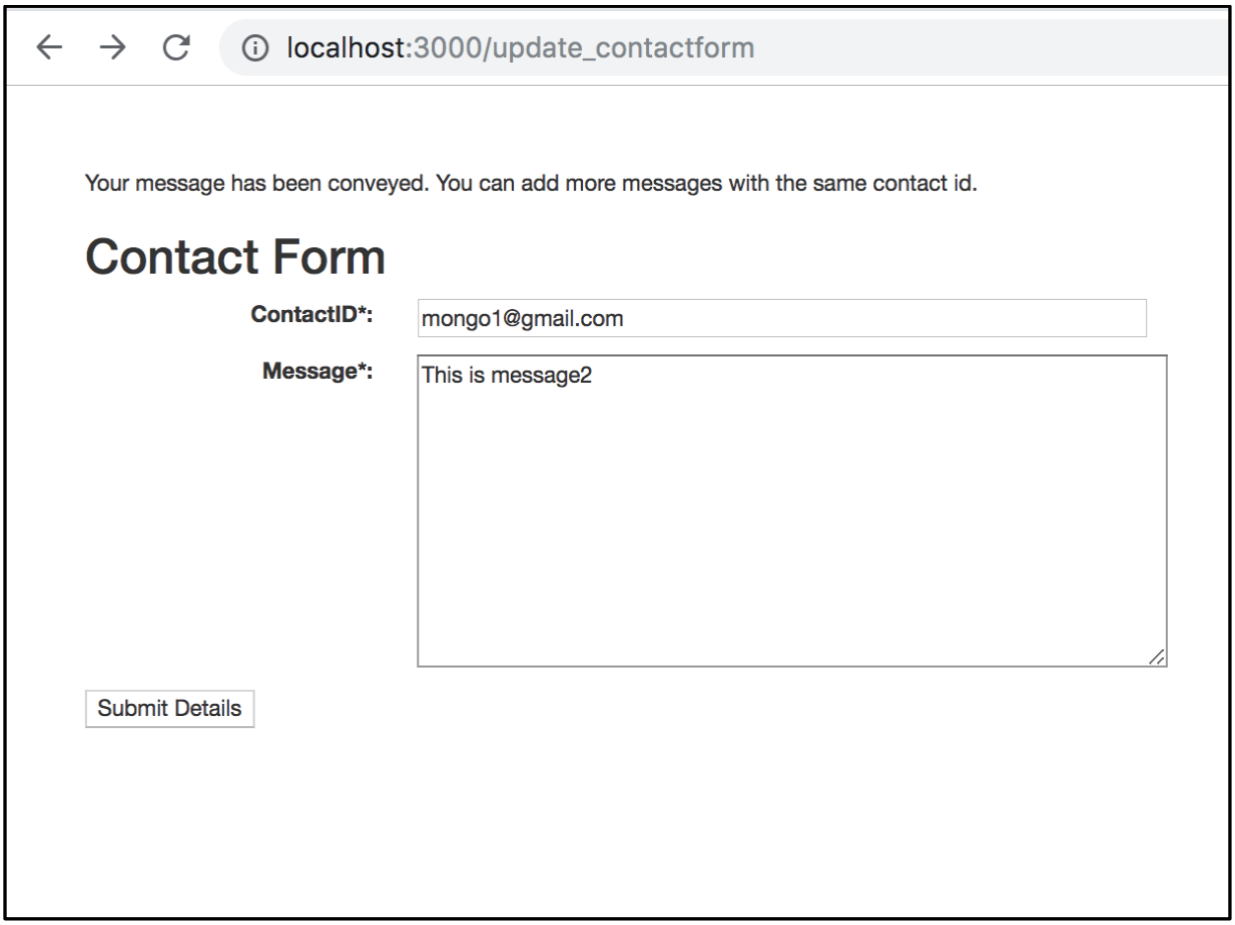
1. Go to home page http://localhost:3000/ and click on “Contact Form” button. Any user can access the contact form to contact the meal manager and does not have to login.



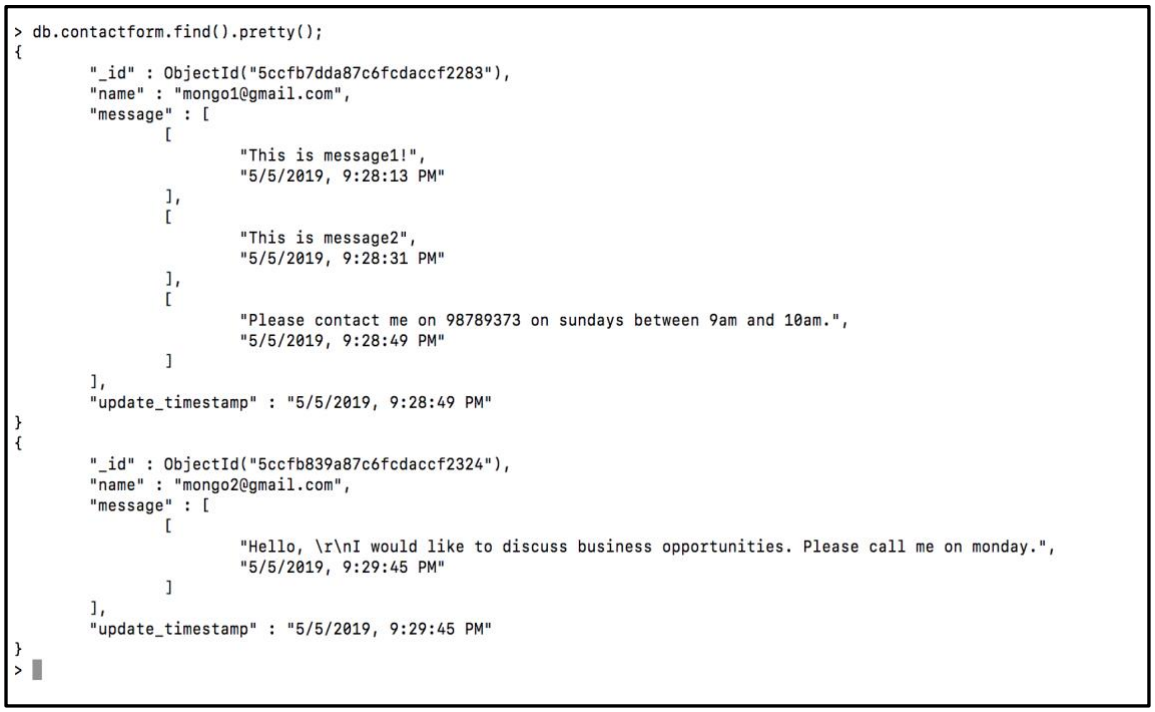
2. The user can enter the following details and submit.



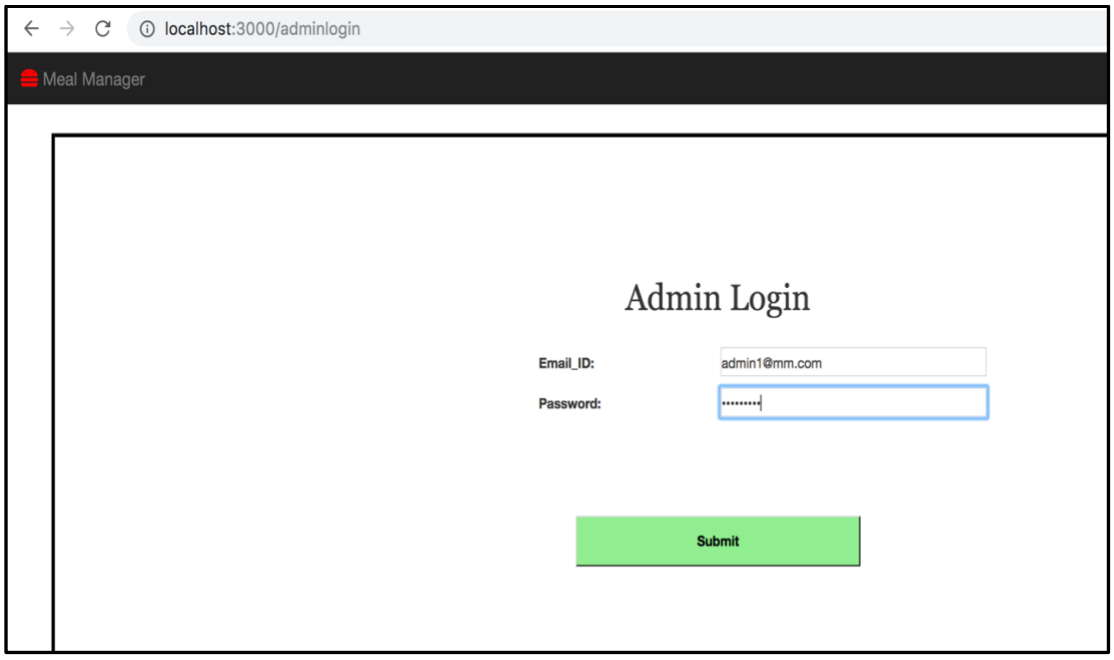
3. The user can send as many messages as he wants. All messages received from a particular contacted will be stored together in mongo db, along with the timestamp of the last message for that particular user.



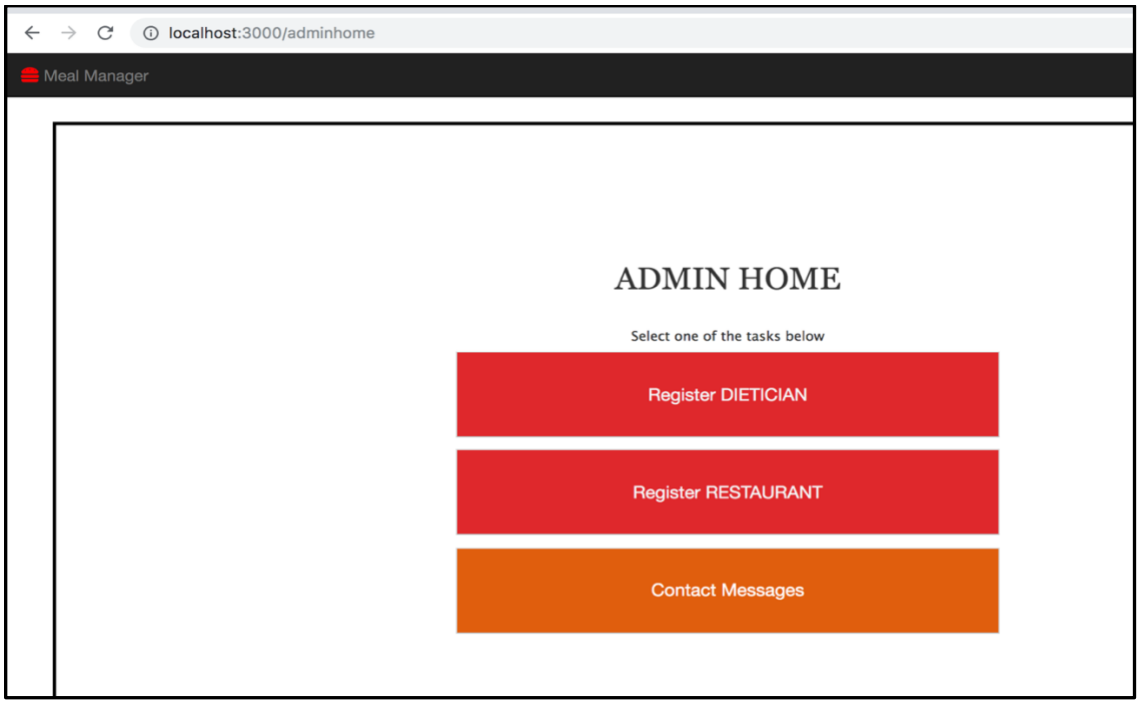
4. Screenshot of mongodb data:



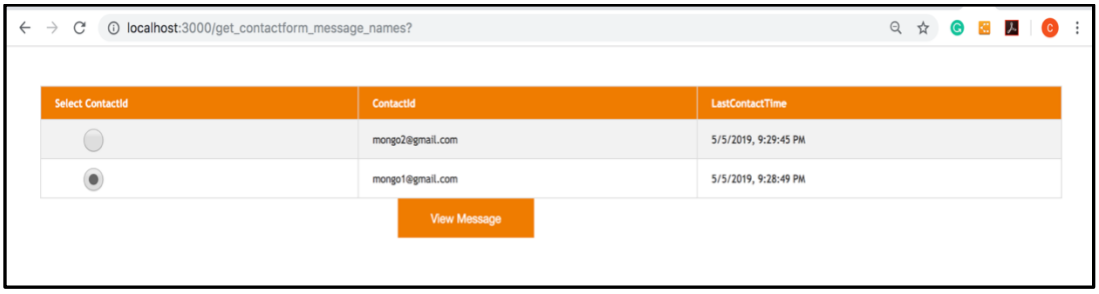
5. Admin will be able to view the contact form data. For that, he must login as an admin from http://localhost:3000/adminlogin. Give the email_id as admin1@mm.com and password as “iamadmin1”



6. When the admin logs in, on the home page he has to click on “Contact Messages” button to view contact form messages.



7. He can view the contact ids from whom he has received messages, and the data is ordered by timestamp in descending order.



8. Select mongo1@gmail.com user and click on “View Message”. He can see all messages received from user mongo1@gmail.com from the contact form.