CS387: Project Deliverables 2

Team

Name	Email	Roll Number
Parth Laturia	parthlaturia@cse.iitb.ac.in	180050071
Anish Deshpande	anishdeshpande@cse.iitb.ac.in	180100013
Rajat Jain	rajat2001@cse.iitb.ac.in	180100091
Devansh Chandak	dchandak@cse.iitb.ac.in	180110027

Topic

Restaurant Management System

Project Deliverable 2 - Analysis

Requirements

The customer shall be able to:

1. View the menu of the restaurant, categorically (cuisine, starter-main-dessert).

1. View the cost and contents of each dish.
2. View the status of the restaurant (open/closed, or full/available).
3. Upon selecting the order option, he/she will be prompted to enter some personal information.
4. Then, he/she will select, one-by-one, the dishes desired.
5. Then, an option for either take-away or dine-in.
6. Finally, a payment option will be present. All of these details comprise the order. Upon completion, the order will be recorded in the order history.

The owner/administrator shall be able to:

1. View all the inventory (pantry) that the restaurant has.
2. View all orders placed.

3. View the menu and its components, employee details (salary, name, ID).
4. Allot an order to a chef, make a purchase order to restock ingredients.
5. View analytics i.e. ‘popular’ dishes and expenditure incurred on a day by day basis, the number of customers served and such analytics.
6. Edit database i.e. Update employee salary, update the menu

The Employee shall be able to:

1. View the orders allotted to them by the owner and fulfill those
2. Once done, allot the order to one of the free/less-occupied waiters
3. See the number and total cost of orders they’ve fulfilled on any given day
4. Browse through the menu and update
5. Update inventory - based on the food cooked or the ingredients received

Internally the database will:
Track the behaviour of a customer - the number of times a customer has ordered, what kind of dishes he/she prefers. Based on this information, when a customer selects the option to order, the system will provide a discount coupon for loyal customers, and will automatically suggest a few dishes the customer is likely to order.
If the stock of some ingredient falls below a threshold(manually set), then an automatic notification is put out for those ingredients. (i.e, addition in the ingredients table).

Analyse the data for insights into preferred dishes, best days for business and other metrics.

Entities and Relationships

(In the inherited tables, we assume that only the primary key of the parent tables are present in the child tables apart from the attributes mentioned)

Accounts -> (Date,Restaurant_profit, expenditure, Total_food_wasted (in kgs))

Chef: inherits People (Salary, Experience, TCDM(Total Cost of Dishes Made))

Contains -> (Ingredient_ID, dish_ID, Quantity_used)

Cooks -> (ID,dish_ID) //Note that though the attribute ID is of people relation, the ID here refers to a chef's ID since chef relation also derives the ID attribute from the people relation

Customers:inherits People -> (Order_Frequency)

Dishes -> (dish_ID, dish_name, cuisine(Chinese, Continental,...), Category(Starter, Main, Dessert, etc), cost)

Ingredients -> (Ingredient_ID, Quality)

Manager: inherits People (Salary)

Non_waiting customer: inherits Customers (Amount spent, Payment Method)

Order -> (ID, Dish_ID, Date, Time, Day, Quantity_ordered, Review {1,2,3,4,5}, Cost)

Owners: Inherits People -> () //no extra attributes apart from those in People

Pay By -> (ID, Bill_ID, Payment_mode, Invoice_ID, Invoice_description, Status {"Paid","In progress", "Failed"})

Payment -> (Bill_ID, Date, Discount_offered {Yes, No})

People -> (ID,name.FN(FirstName),name.MN(middle_name),name.LN(last_name),gender,Age)

Receptionist: inherits People (Salary)

Sitting -> (ID,Table_ID)

Table -> (Table_ID,Size,Status{Ordering, Ordered but not eating, eating, finished}, location{"Window Side", "Centre", "Door Side"})

Tracks -> (ID,Date)

Waiter: inherits People (Salary)

Waiting Customer: inherits Customers (Waiting_Number)

Entities:

Accounts, Chef, Customers, Dishes, Ingredients, Manager, Non_waiting customers, Owners, Payment, People, Receptionist, Table, Waiter, Waiting Customers

Relationships

Contains: between Ingredients and Dishes

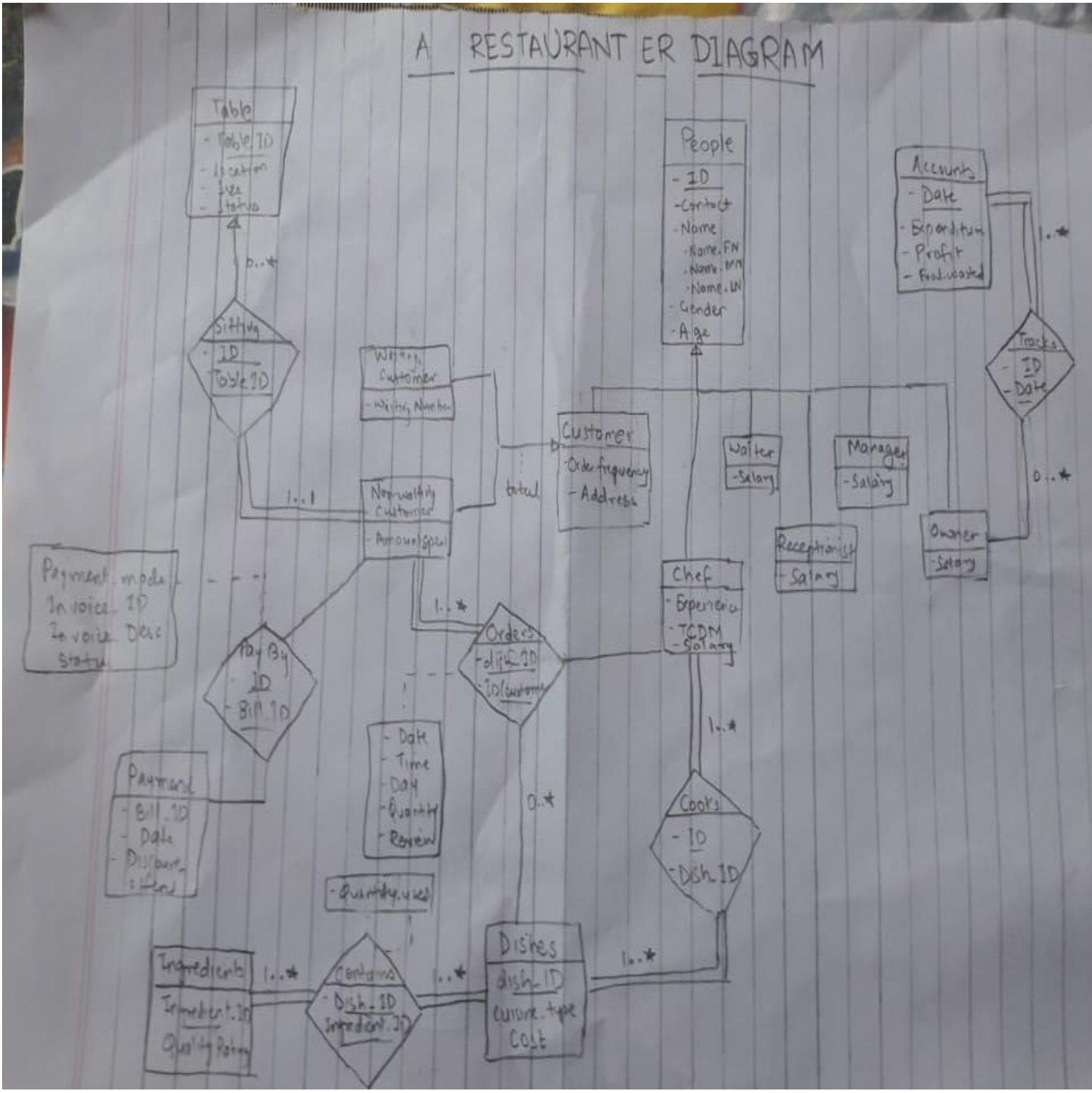
Cooks: between dishes and Chef

Order: between Dishes and Non_waiting customer

Pay by: between Non_waiting customer and Payment

Sitting: between Table and Non_waiting customer

Tracks: between Owner and Accounts


A RESTAURANT ER DIAGRAM
Table
- Table ID
- Location
- Size
- Status
People
- ID
- Contact
- Name
- Name.FN
- Name.MN
- Name.LN
- Gender
- Age
Accounts
- Date
- Expenditure
- Profit
- Food wasted
Tracks
- ID
- Date
Sitting
- ID
Table ID
Waiting Customer
- Waiting Number
Customer
- Order frequency
- Address
Waiter
- Salary
Manager
- Salary
Owner
- Salary
Receptionist
+ Salary
Chef
- Experience
- TCDM
- Salary
Cooks
- ID
- Dish ID
Normality Customer
- Amount spent
total
1..1
Payment mode
Invoice ID
Invoice Desc
status
Pay By
ID
- Bill ID
Payment
- Bill ID
- Date
- Discount
- Total
Orders
- Order ID
- ID (Customer)
- Date
- Time
- Dish
- Quantity
- Review
Quantity used
Ingredients
Ingredient ID
Quality Rating
Contains
- Dish ID
Ingredient ID
Dishes
dish ID
cuisine type
Cost
1..*
0..*

Use Cases

Enumerate all use cases for each of the roles in your system. For example, you may have a "Admin" role, a "Manager" role etc. A use case will outline the following details (stick to this template for all of them):

1. Title of Use case: for example "Logging on to the system" or "Querying for the most popular dish" or "adding a new user to the system"
2. Description of what the use case is meant to accomplish for the user.
3. Trigger event for the use case - human triggers or is triggered automatically (analytics use cases may be this way)
4. Primary actor - this is the user role who will be interacting with the system
5. Inputs to be supplied by primary actors and constraints on input.
6. Preconditions for the use case - state of data - what is the world state that the use case is relevant for.
7. Main success path - what is the set of interactions with the system and the final output.
8. Exceptions that may arise and how they will be compensated.
9. Post conditions - after executing this use case, what will be the state of the system?

List of Use Cases, elaborated on the bold ones after that -

Customer:

User registration, Logging into the system, **Searching for dishes(Search Engine)**, Add to Cart, **Place an Order**, Previous Orders, Cancel Order, My Preferred Order, Get Discount, Recommender System

Owner:

Add Employee, **Manage Employee Info**, **Analytics** of: Sales, Customer Visits, **Allot Order to Chef**, **View Order History**, Employee Leaderboard

Employee:

Mark completed orders, **Manage(update) inventory**, **Allot order to waiter(Serving Food)**, Orders completed, **Automation**

Search Engine

1. Search Engine
2. The user can search for dishes by name, id of dish
3. Human trigger
4. Customer will be the primary actor, even employees can use it to show the menu to the visitors
5. Input will be the kind of Cuisine or Category, Dish name or Dish id
6. Users must be logged in.
7. Based on keyword matching the database will return relevant results
8. In case of wrong input, “Sorry, no matching dishes!” will be displayed
9. The user can go through the results displayed, check the ingredients and choose food/ search again

Place an Order

1. Place an Order
2. The user can place an order using the menu in the restaurant
3. Human Trigger
4. The customer who places the order
5. The list of orders (specified by the dish_ID) that the user will provide using a form. The dish in the order must be present in the dishes entity
6. The customer must be logged in.
7. The main set of interactions consist of filling the form and inputting the order dishes into the system. The final output is successful placing of order
8. The user places order for some dishes not in the menu
9. The order will have been placed and sent to the manager for further processing

Manage Employee Information:

1. Manage employee info
2. Update the salary, add employee, find best performing employees, other personal information
3. Human trigger
4. The owner of the restaurant
5. Enter employee name to edit their details, monitor their performance
6. Must have admin rights+logged in
7. Navigate to the employee tab on the admin interface
8. None

9. If the owner makes any edits and saves them then those changes will be pushed to the database and will compromise the new state

Analytics

1. Analytics
2. The system generates some insightful analytics that provides a realistic view of the restaurant system and its quantitative features to the user.
3. Automatic Triggered
4. System and its computational logic
5. The statistical measures and values of restaurant's attributes and features (like expenditure, Food_wasted,...) on a daily basis
6. The user is logged in and must be an owner of the restaurant
7. The values are represented in an elegant and understandable format and successfully capture the trends of the provided data
8. The data type and attributes are not matching as per the conditions
9. The owner has captured the required analysis

Allot Order to Chef

1. Allot Order to Chef
2. The Order will be allotted to the designated chef
3. Human Triggered
4. The manager is involved as he/she will assign the corresponding orders and so manager will be the main actor interacting with the system
5. The orders received from the users. Though the user may order multiple dishes, the ordered list will be broken down into smaller orders such that each order has only 1 dish in it. This will be done for each user and this will finally become the input to the system
6. The corresponding user must have given some order that the manager is currently transferring to the chef(s)
7. The chef(s) receive the order
8. No chef is alloted the given order/ is allotted to an absent employee
9. Those chefs who received the order will now start preparing it

Order History:

1. Order History
2. Get a list of all previous orders made by customers
3. Triggered automatically
4. Employee/Owner

5. The user can input a particular date/ dish to see the order history
6. User must have non-customer status
7. Main set of interactions involved choosing the right filter for viewing the relevant history. The result is a table consisting of all relevant orders, their date, quantity, revenue etc
8. None
9. The state doesn't change, since order history is view-only

Update Inventory:

1. Update Inventory
2. Based on the dishes prepared and the incoming ingredients, update the inventory
3. Human triggered
4. Employee/Owner
5. Choose the ingredient to update and enter the new quantity
6. User must have non-customer status
7. Main set of interactions involve entering the right amount added/subtracted. The final result is the update of the database.
8. The quantity cannot be negative
9. The new changes will be pushed to the database

Serving Food:

1. Employee allots order to waiter
2. The Order will be allotted to the designated waiter
3. Human Triggered
4. The employee is involved as he/she will assign the corresponding orders and so employee will be the main actor interacting with the system
5. As soon as the order is completed by the employee, they can allot the order to a waiter, who will serve the client then
6. Employee login
7. Employee checks available waiters and allots the task. The output is that the task is transferred to the waiter and marked completed by the employee.
8. Order is allotted to an absent waiter
9. The order moves from employee interface to waiter interface, marked done on the employee interface, gets added to employee performance table

Automation:

1. Personalization
2. Track customer behaviour, ease ordering by recommending frequently ordered/ famous foods, notification to owner if some ingredient's quantity falls below a threshold
3. Automatically triggered
4. Main actor is the backend, the customer is the beneficiary
5. User does not give any specific input, the recommendation is based on the history of orders the user has made so it happens automatically
6. Previous user data is required
7. While searching food, placing an order - the automation will help reduce hassle by remembering previous information.
8. No serious complications, just that initially the recommendations may not be useful for users due to small size of training data
9. User can order on their own or choose from the recommendations, order ingredients if value falls below threshold

Technology Choices

7. Any preliminary technology choices you would like to commit to and why - you may change these later.

Possible Frameworks to be used:

- ASP.NET + MS-SQL + C#
- Django + Angular/Node/React

We will decide on one of the above combinations in due course of time before starting development. We will use MVC architecture or ASP.NET WebForms for our application.

Reason for shortlisting the above two is the robustness and ease of use. Also, our team has previous internships and projects experiences on ASP.NET and Django/Angular. We will be learning Node in this course so that is also a good option.

In our experience with ASP.NET it is extremely easy and fast to use and add complex features. The Toolbox in Visual Studio allows rapid additions of basic HTML elements. The CRUD view allows different views of the inherent database and allows users to create/edit/update/delete entries (directly or through forms) which is extremely useful in any Database application. ASP.NET has a separate MVC architecture framework which we will be able to incorporate into our project.

Django: we have used in CS251 and our familiarity with Python encouraged us to consider this option. It supports the standard MVC architecture and can be used with MySQL/PostgreSQL.