CS387 Project: Scope Document

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

Scope Description

High Level Overview

We are planning to add 3 views: one for the customer, one for the administrator or the owner, and one for the employees (waiters and chefs and delivery persons).

Customer View:
Scroll through the Menu swiftly using the search feature which will use tags like “italian”, “spicy”, “beverage” to display relevant results. The order can be placed instantly via a form (with name/Sl No. of dish and the quantity demanded) for online delivery. He/She has to provide the address and contact number also. The address code in the address field will be compulsory. If the order is made offline, i.e, at the restaurant, then no address is required (as it will be either a take away or a dine-in).

Admin/Owner View:
The admin will have the superuser view of the restaurant. He will be able to edit existing employee details or add a new employee to the table with employee details. There will be a table specifically for employees who are chefs. He will manage a pending orders table which has pending orders and will be able to delete from it once he places the order. We can have a table whose order is placed but not yet delivered. Apart from this, there will be a booking reservation system for the customers that want to eat in the restaurant. So, when the capacity gets filled, the next customer will have to wait until a table in the restaurant is vacant and available. This won’t be applicable for the customers that order food online or on call.

The admin view will also have a table which has all dishes details (a subset of which is the menu visible to the employee) - price, name, expected quantity that can be delivered, and the chef(s) who cook that particular dish. The chef_id will be a foreign key referencing the chef_employee table.

Inventory Management

It will also have an inventory management component. This system needs to track ingredients purchased, dishes cooked and the ingredients used in each dish for the day. The perishable nature/expiry of the goods will also be taken into account, and we will store the date/time an ingredient is acquired. The restaurant provides a fixed number of serialized tables on its premises for which occupancy data needs to be tracked by time of day and day of week. It accepts booking on the phone for these tables. So a table is either reserved/occupied/unoccupied.

There exists a table which stores delivery details of different areas with employee assigned per area and list of sub area codes under him

A customer record is maintained where premium customers can be given discounts

Relations

Note: This is a rough sketch -- there may be significant changes

Restaurants(ID, area, rating),
Employees(ID,name,category) //a master employee table for all types of employees
Staff(ID, works_at, job {Manager, waiter, chef, watchman, washer,...},salary) -- can do specialization and make relationships like cooks between chef and dishes
Dishes(dish_id, cuisine_type, cost, ingredient_ID, chef_id -- will be many), ingredients(ingredient_ID, quantity, rate, quality)
delivery_system(area_primary_code, area_secondary_code), customers(Id, orders, discounts, gender)

Table(ID, Status, size)
Waiting_Customers(ID, Name, waiting number, gender)

I feel we will be required to add restaurant_id in every other relation

What we will NOT be doing

- Food Recommender System - Recommend the user some dishes that he likes based on his/her restaurant related history i.e. past choices and ordering of dishes
- Chatbot support for queries
- Voice to text translation- A customer speaks up what he/she wants to order and the system will apply automatic speech recognition techniques to convert it into text that will be used to further place the orders accordingly.
- Computing the expected delivery time of the orders placed
- Customer support - in-app chat for complaints and suggestions
- No voice call support for queries

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.

Our world in Data - from data
K nearest neighbour - Recommender system
Middle tier - Modern view controller paradigm
Test planning - Plan on how to test
Testing, Incorporating fixes - a loop
Jmeter - generate load by simulating multiple users

Tuning - the database

Should this include reviews of service, food delivery?
Geospatial database for delivery feasibility
Postgis - Geospatial equivalent of Postgres

Transaction system - the one with which users interact, not much in bg
Don’t work on all - Delivery + Inventory + Reviews
Start Slow