Food World

By:

Harshad Sathe

Preety Mishra

Vivekananda Shankar Archarya

Instructor:

Jose Annunziato

Problem statement:

As proposed earlier, in today’s world people want everything on the click of a button. Most of us find it difficult waiting outside a restaurant for our turn. What if, your are in a group with your parents and grandparents. Would you like them to wait outside the restaurant until your turn comes? Now a day, people have busy schedules and the last thing they want is to be in the waiting list standing outside a restaurant. Nowadays, there are a lot of restaurants and we prefer choosing the ones that our friends like and have had an amazing experience there. This is an interesting situation where people want to follow other people; may be their friends, family or someone with similar choice of restaurants.

Proposed solution:

We propose a online Food World application, where people can book their reservation for dinner and can connect with other foodies at the convenience of a click. Now, make a reservation prior and save yourself from waiting outside the restaurant for your turn. User will have their personal account where they will login and make a booking. A user can select the date, time and no. Of people for the dinner and can book the reservation. User can search the restaurant and make a reservation. User can see the restaurant’s logo and address to make sure that it’s the same restaurant they are searching for. After a reservation is made, they can see their reservation history too. Also, they can add a restaurant to their favorites. Adding in favorites saves user from making a search again and they can make a booking straight from their favorites. They can edit their profile and update it. A user can follow/ unfollow another user and can see their reviews.

Architecture:

- A user can create an account with a username and password. They can edit their profile once they login.
- After login in the user can see the reviews of all the users he is following.

- They can search the restaurant based on the location and the name of a restaurant, name of the restaurant only, location only or based on the zip-code only using Google Places API (Place search request). A list of restaurants is displayed on the Search results page with a few details and the restaurant logo on the right side.
- Clicking on the restaurant name, a restaurant details page will render and display all the details like name, address, phone no., website, price level, ratings and image of the restaurant.
- A reservation can be made from the details page by clicking the “Make Reservation” button, which will redirect the form to make reservation page. A user can enter the no. Of people, date and time of the reservation and can submit it. The data is stored in the database. The landing page now will be “My Reservations” page with the updated number of reservations the user has made.
- The restaurant details page also has “Add to favorites” and “Write a review” button with which he can add a restaurant to his favorites and with review he can write the reviews about the restaurant.
- In “My Reservations” tab the person can see a history of his reservations and delete button. A click on delete button will delete that reservation and the page will render to the “My Reservations” with the updated list of reservations.
- A user can follow other users by making a search based on the username of the other user. They can unfollow the user from “Unfollow” button. The user data will be removed from the following table.
- At the end, user can log out from their profile and the session expires.

APIs:

We are using **Google Places API’s** in our project. The Google Places API is a service that returns information about Places, defined within this API as establishments or geographic locations. It uses HTTP requests and returns a JSON object response.

We have used the Google Places APIs in our project to Search for Restaurants based on a Restaurant Name and/or Location. This is a TEXT based search, which receives a JSON object response from the API.

We are using the following Place requests:

Place Searches:
Returns a list of Places based on a user's location or search string.
Used in the Application (Home>Search a Restaurant) : The API expects Location and Restaurant Name as a Text based input, and returns a list of relevant Restaurant information for the input text.

URL:
https://maps.googleapis.com/maps/api/place/textsearch/xml?query="Place"&key="AddYourOwnKeyHere"

Required parameters

- Key — Application's API key.
- Location — The Place text for which the API retrieves place information.

Place Details:
Requests return more detailed information about a specific Place.
Based on the information received from Place Searches, our application displays all the relevant search results on searchResults.jsp page. If we click on a particular Restaurant, we trigger the **Place Details** request for the selected Restaurant and display specific details about the restaurant like the the Opening and Closing Time, Website, Phone number and display this information in restaurantdetails.jsp page.

URL:https://maps.googleapis.com/maps/api/place/details/json?placeid={PLACE}&key=AIzaSyCTxX10Hznx4ta5ZvlCS1BFXxDOwNJIQ-s

Required parameters:

- Key — Application's API key.
- PlaceId— The placeId, which is available from the place search API service, for which the API retrieves detailed place information.

Place Photos: Gives you access to the millions of Place related photos stored in **Google's Place database.**
For every Restaurant which we search and display in the search results page, a Google Places Photo service executes, and gets an image of the restaurant which is then displayed on the results page. This service responds with an image for a given place.

URL:
https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photoreference={PHOTO_REF}&key=AIzaSyBK7757GStqJx2Xudqnm5IMViOBmGIdJFM
Required parameters:

- Key — Application's API key.
- Photo reference— The photo reference for the Place,which is available with the Google Place Search API service.

Technology stack used:
Frontend:

- JSP + JavaScript + CSS
- AJAX
- JWS (Webservices)
  1. Address
  2. Restaurant
  3. User

Backend:

- JPA

- mySQL

<u>Server:</u>

- Apache Tomcat Server 7

<u>Operating System:</u>

- Windows OS

Use cases:

1) User Login
2) Incorrect Login credentials
3) User Logout
4) Register User
5) Edit/Update Profile
6) Delete User
7) Display User Profile
8) Find a User
9) Incorrect user search-Display error string
10) Display Searched User profile page
11) Follow a User
12) Unfollow a User
13) HOME>Find a user>Favorites>Click on a Restaurant for more Details
14) My Following-Displays all users whom the logged-in user is following.
15) My Following > Click on User>Following Users Profile page
16) My Reviews-Displays all the reviews of the logged in user.
17) My Review>Click on Restaurant> Restaurant Detail Page
18) My Review>Click on Restaurant> Restaurant Detail Page>Add a Reviews
19) My Favorites: Displays all the favorite restaurants
20) My Favorites>Click on Restaurant > Restaurant Detail Page
21) My Favorites>Click on Restaurant > Restaurant Detail Page>Add to Favorites

22)My Reservation: Display all the reservation

23)My Reservation> Update reservation

24)My Reservation > Delete a Reservation

25)Search Restaurants by Name and Location

26)Search Restaurants by Name only

27)Search Restaurant by Location Name/Neighborhood only

28)Search Restaurant by Zip Code only

29)Display Search Results

30)Back to Search Page Button

31)Search Result Page> Click on Restaurant>Display Restaurant Details

32)Restaurant Details>Click on Restaurant Website

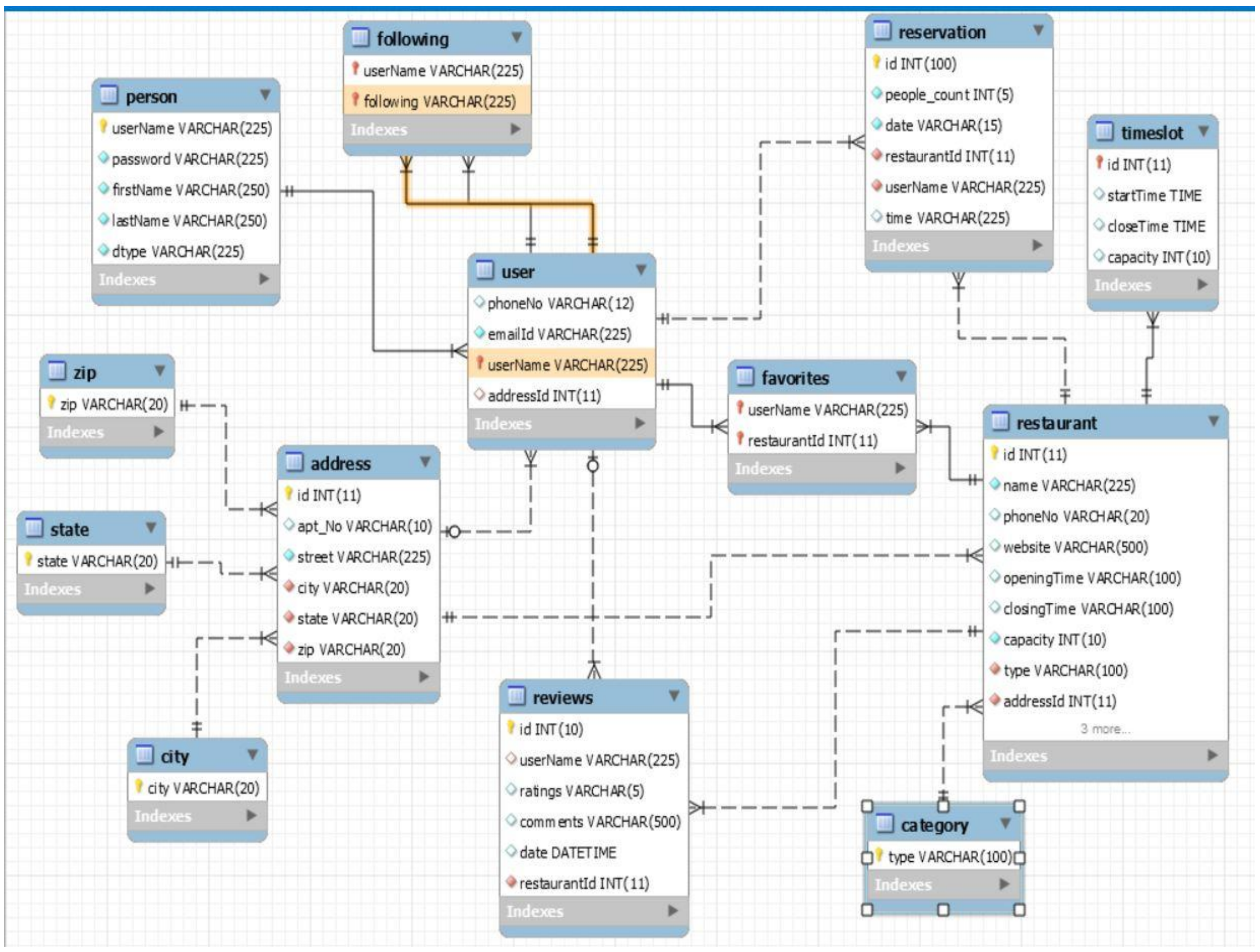
33)Restaurant Details>Click on Restaurant Website>Press Browser Back Button

34)Restaurant Details> Make a Reservation

35)Enter Reservation Details and Reserve a Table.

36)View reviews of the users, the logged in user is following on Timeline.

Data model (UML Class Diagram)



Category			
Relational Model or MongoDB			
Tables (count)	13		
Fields (count)	51		
Foreign Keys (count)	16		
Mapping Tables (count)	15		

Association Tables (count)			
Enumeration Tables (count)			
Object Model	13		
Entity Classes (count)	46		
Entity Fields (count)	13		
DAOs (count)	13		
Create Methods (count)	21		
Find One Methods (count)			
Find All Methods (count)			
Find Other Methods (count)			
Delete Methods (count)			
Update Methods (count)			
JWS or Express Web Services			
End Points (Mapped Methods) (count)			
JSON Producers (count)			
JSON Consumers (count)			
GET Methods (count)	2		
POST Methods (count)	11		
PUT Methods (count)	4		
DELETE Methods (count)	2		
Server Web Service Client			
Browser Web Service Client			

Online Web Services (list names)	AddressWebService UserWebService ResturntSearchWebService		
Online Web Services (count)	3		
AJAX GET (count)	2		
AJAX POST JSON (count)	11		
AJAX PUT JSON (count)	4		
AJAX DELETE (count)	2		
User Interface			
Pages (count)	16		
Input Fields (count)			
Links (count)			
Buttons (count)			
Data Tables (count)			
Data Lists (count)			
JavaScript Libraries (count)			
CSS Libraries (count)			
Look and Feel (count)			