Ministry Category: Ministry of Road Transport and Highways

Problem Statement:

Team Leader Name: Shubham Vishwakarma

Problem Code: #MRT5

College Code:

Problem Statement:

The Idea is to develop an application which will create a database of the Eating Joints falling with the Origin and Destination Points of the journey and placing the orders as per the available Menu at particular Eating Joint with time of Service.

Brief Solution:

We propose to develop a platform comprising of an android application and website which will enable a traveller to search and order food from eating joints which are located throughout the route to their destination.

Our platform would include user authentication and thus would enable us to maintain a user profile to deliver a more personalised experience to our users. We will get the current location of the user using GPS (with the option of entering it manually) and will ask for a destination, after which we will display the list of eating joints which are located near their current route arranged in increasing order of their time of arrival. The user can select the preferred eating joint based on their requirements and will be able to place their order with the time of service after selecting food items from the menu of the selected eating joint.

Description of Technology:

Structure of Database:

- We will be using MySQL database management system for our data storing requirements.
- Our data base will consist of tables for storing information about the users and various eating joints.
- The user specific table will consist of data points such as name, mobile, email, unique ID etc.
- Information about eating joints will comprise of name, unique code and location (in the form of latitude and longitude).
- We will be storing the menus of various eating joints in separate JSON files whose link will be stored along with other information of the eating joints in the concerned table.

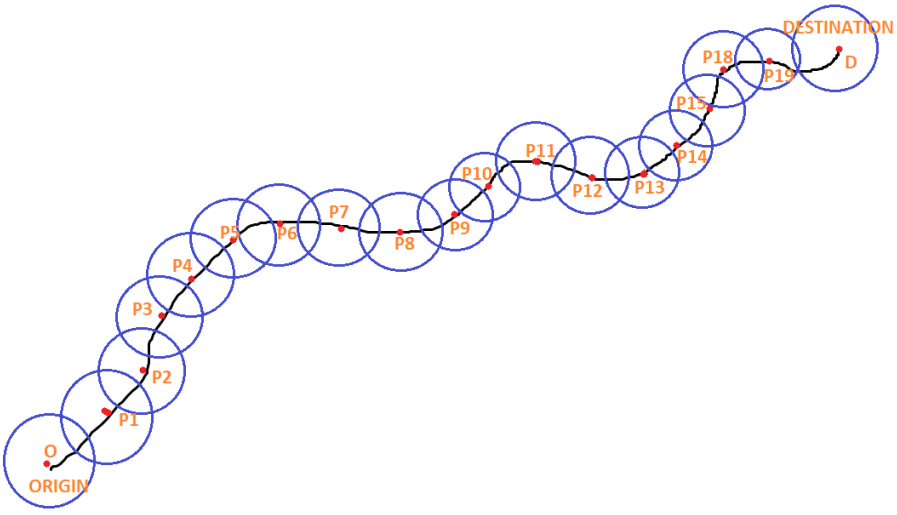
Functionality of app:

- The user will have the option of logging-in via their mobile number or through their Facebook/Google account.
- Upon successful authentication, the user will have to enter their destination by Entering address or by placing marker on the map.
- The current location will be fetched by our app through GPS or it can be Entered manually by the user.
- The shortest path will then be calculated by the Google directions API. The polylines that the google’s API will throw would enable us to calculate nodes along the path.
- Considering these nodes as centres of circles of radius 2km, we will search for eating joints located within them.We would achieve this by getting the latitude and longitude of these nodes and then searching in our database for the qualified eating joints.
- These joints will be displayed in a map and the user will be to view the list of all the eating joints sorted according to their distance from the current location.

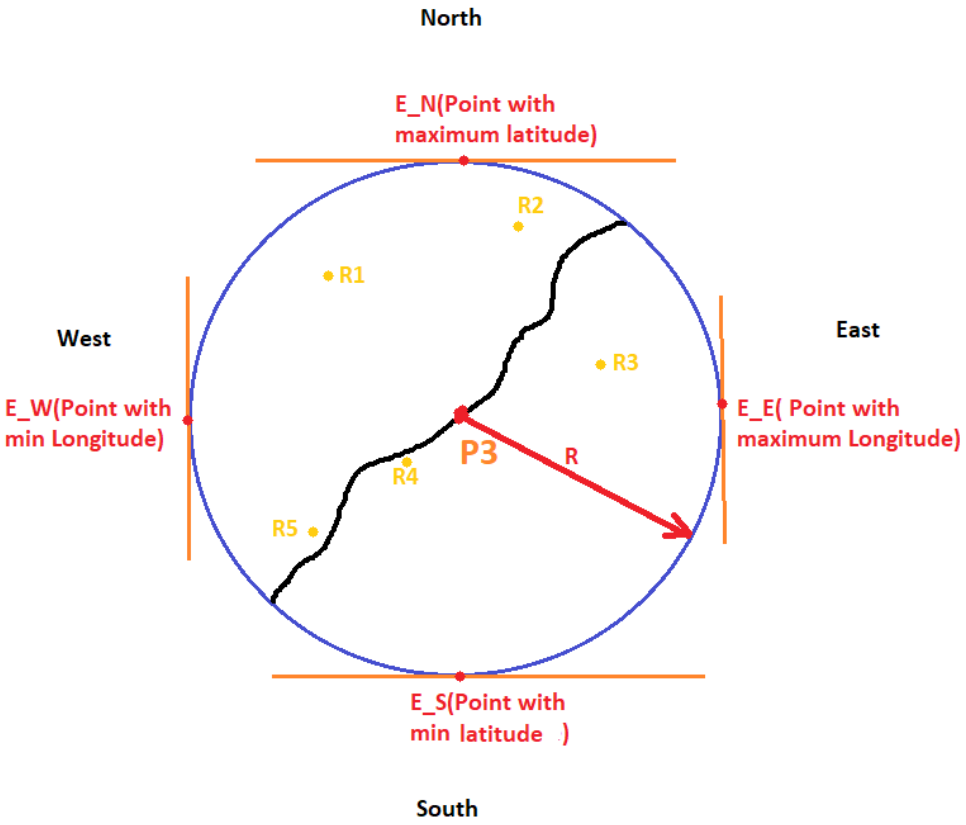
- After selecting the preferred eating joint, the user will be provided with the menu of the selected eating joint.
- The user will then be able to select food items and place their order with the eating joint.
- The order will be confirmed by the respective eating joint through confirmation call. The user can then directly start navigation to the concerned eating joint.

Algorithm to find Eating Joints near the selected route:

Let O and D be the origin and destination points of the route respectively. Let P1, P2, P3 ... Pn be the points lying on the route at distance y, 2y, 3y, ... , ny respectively from the origin such that distance of O and D is (n+1)y.



Now, considering each of the above point as a centre of circle of radius ‘R’ km. Let us consider the scenario of one such circle.



Suppose R1, R2, R3, R4 etc. are the locations of eating joints lying within the circle. We will then find the points having latitudemin(E_S), latitudemax(E_N), longitudemin(E_W) and longitudemax(E_E) lying on the circumference of circle using the following formula:

$\varphi 2 = \text{asin}(\sin \varphi 1 \cdot \cos \delta + \cos \varphi 1 \cdot \sin \delta \cdot \cos \theta) \; \lambda 2 = \lambda 1 + \text{atan2}(\sin \theta \cdot \sin \delta \cdot \cos \varphi 1, \cos \delta - \sin \varphi 1 \cdot \sin \varphi 2)$

Bearing = 0 gives max longitude Bearing = 180 gives min longitude bearing = 90 gives min latitude bearing = -90 gives max latitude (all angles in radians)

where φ is latitude, λ is longitude, θ is the bearing (clockwise from north), δ is the angular distance x/R; x being the radius of circle, R the earth's radius.

Following is the implementation of above formula in JavaScript:

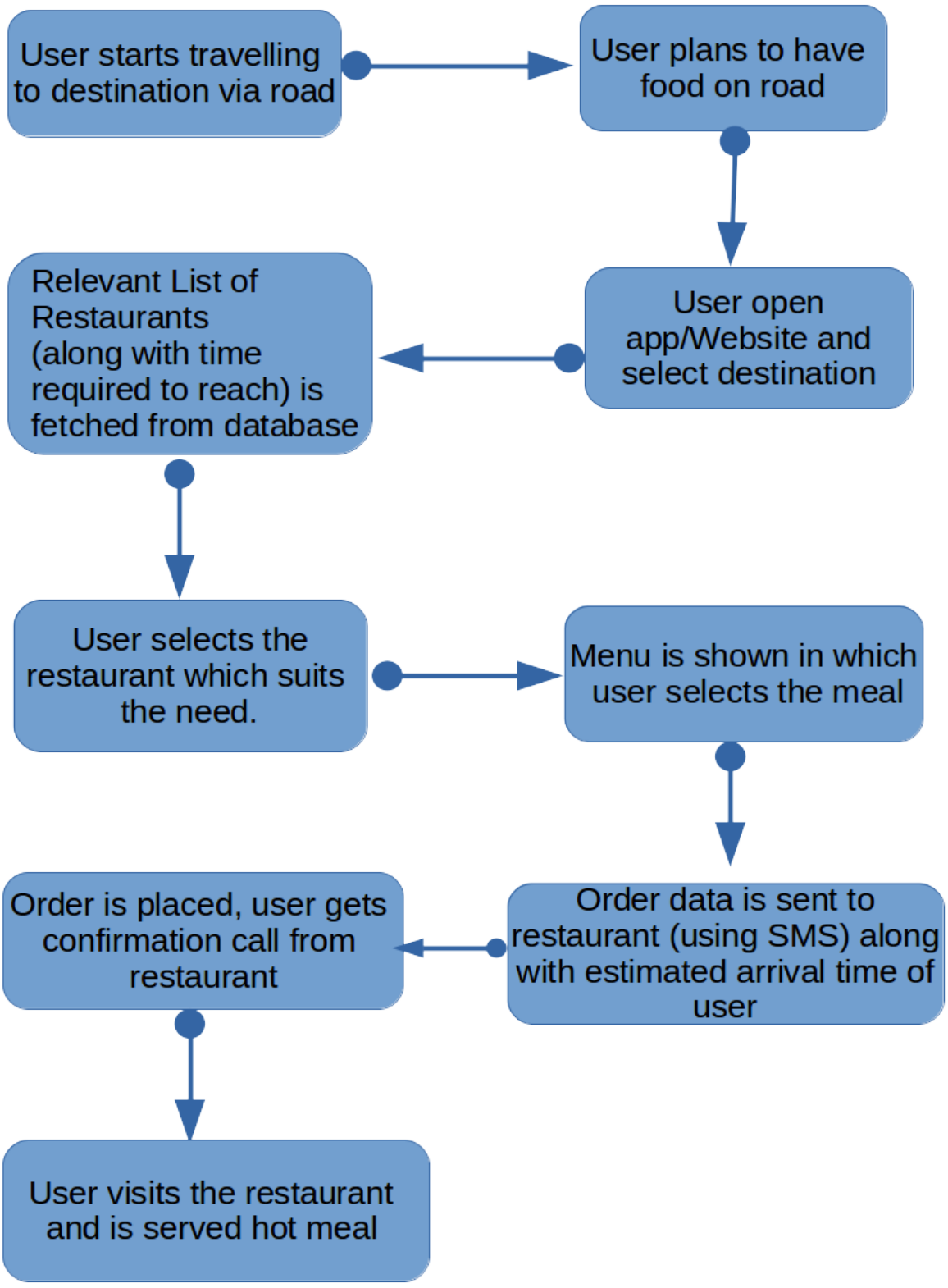
```
var φ2 = Math.asin( Math.sin(φ1)*Math.cos(d/R) +
          Math.cos(φ1)*Math.sin(d/R)*Math.cos(brng) );

var λ2 = λ1 + Math.atan2(Math.sin(brng)*Math.sin(d/R)*Math.cos(φ1),
              Math.cos(d/R)-Math.sin(φ1)*Math.sin(φ2));
```

Now, we have lat_{min}(E_S), lat_{max}(E_S), $long_{min}$(E_W) and $long_{max}$(E_E) of the circle of radius x encircling the eating joints with centre ‘p’. We’ll find minimum latitude, minimum longitude, maximum latitude, maximum longitude of all points P1 to Pn.

Now, we’ll find restaurants for each point lying within the circle by using binary search algorithm which will search all the available restaurants in logarithm time. The algorithm for binary search is implemented by MySQL database itself while using normal queries.

First database will try to search for minimum and maximum lattitude which is already stored in increasing orderin. When the said range is found then the eating joints in the range of longitude is searched which are then returned by the database.

Use Case:


User starts travelling to destination via road
User plans to have food on road
User open app/Website and select destination
Relevant List of Restaurants (along with time required to reach) is fetched from database
User selects the restaurant which suits the need.
Menu is shown in which user selects the meal
Order data is sent to restaurant (using SMS) along with estimated arrival time of user
Order is placed, user gets confirmation call from restaurant
User visits the restaurant and is served hot meal

Technology Stack Used:

Front-End:

App:

- Java and Kotlin for front-end logic
- XML for layout designing
- Firebase for user authentication and notifications

Website:

- HTML
- CSS
- JAVASCRIPT
- React.JS with Redux (For State Management)

Back-end:

(Will be same for app and website)

- PHP/Node.js
- MySQL (for Database purpose)
- Firebase

Showstoppers:

- Challenges in collecting the data for the database from all the eating joints all over the country.
- Challenges in marketing of the app and user awareness.
- Challenges to maintain the availabilty of each food item shown on menu and uncertain opening and closing of Eating Joints. This can be achieved using a Merchant App by which every Eating Joint can update there Informations in real time.