TEAMNETWORK

SheCodes Hackathon Airtel

Team Name : Team Network

College Name: NIT Hamirpur

Team Leader's Name: Ravroop

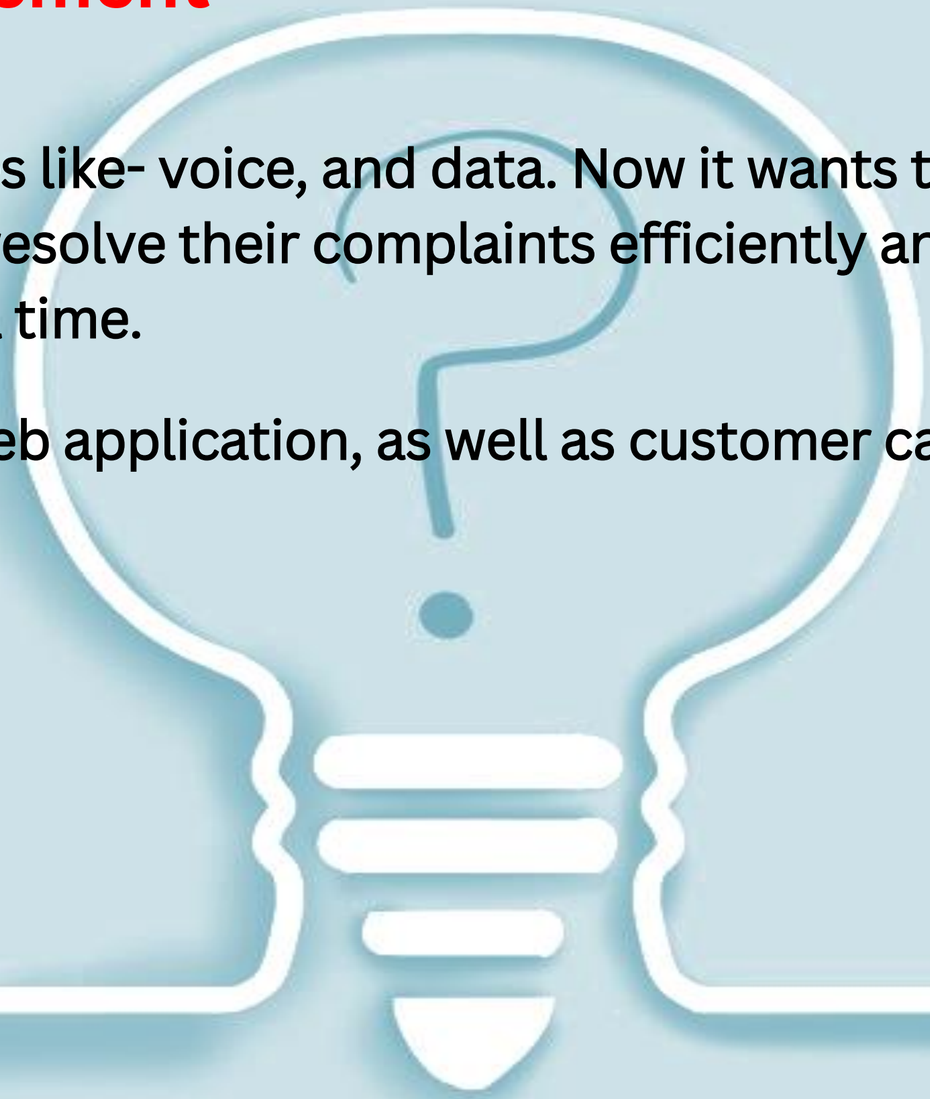
TEAMNETWORK

Member's Names	College Name	Email ID	Roll No.
Ravroop Kour	**NIT Hamirpur**	**21bee013@nith.ac.in**	**21BEE013**
Tanu Tanu	**NIT Hamirpur**	**21bcs055@nith.ac.in**	**21BCS055**

Problem Statement

Airtel provides customers with various services like- voice, and data. Now it wants to connect customers to skilled agents who can resolve their complaints efficiently and accurately in the shortest time possible in real time.

Airtel provides services through an app and web application, as well as customer care and store visits.



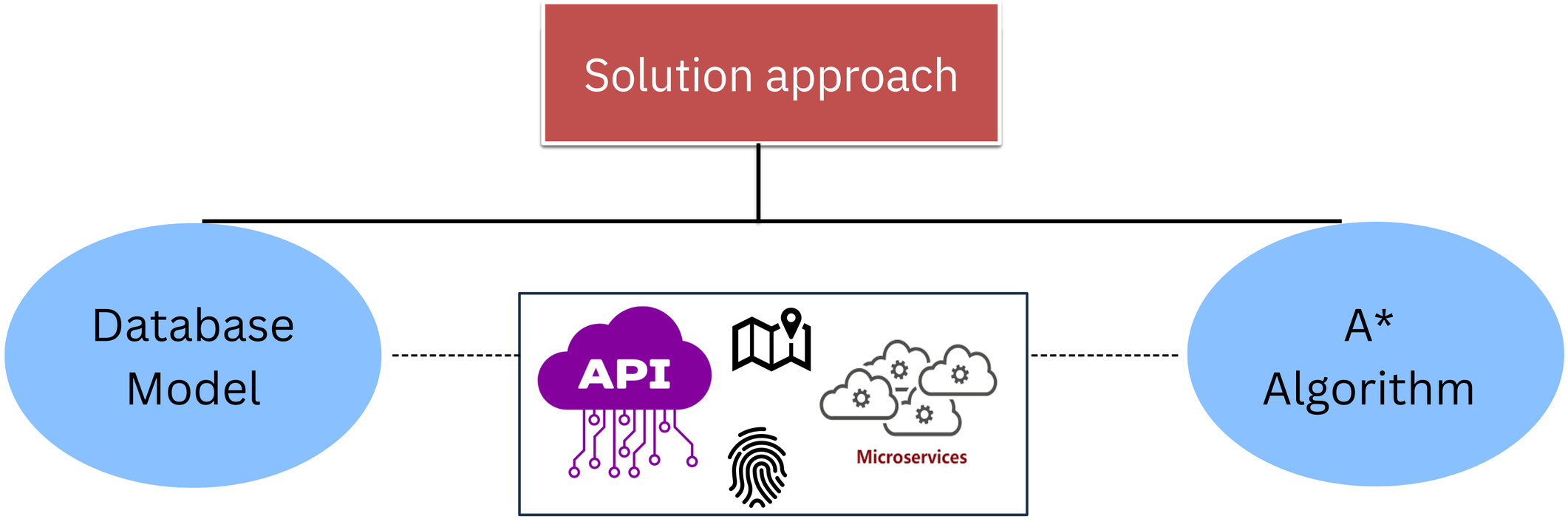
Choosing this Problem Statement

- Customer satisfaction is the most important factor for the growth of a company.
- Great products do not live up to their potential if proper customer services are not provided.
- We as customers of this service find this as a major pain point.
- If Airtel wants to pioneer this market, the problem statement needs efficient and scalable implementation.

Solution Approach:

1. Define problem- A skilled agent must reach the desired customer location in shortest path and time efficiently and accurately.

2. Solution Design- The problem statement is analogous to the food delivery system where the delivery person must reach the customer within a time frame. Companies like Zomato, and Swiggy have been successful in doing this and the same learning can be used in this service.



Assumptions:

- We have the list of agents with the requisite skills for the service requested.
- Agents are provided with devices that have inbuilt GPS which will help to locate their current location in real time.
- Customer make calls to nearest branch.

Database Model:

5 tables are created for storing:

1. customer information
2. agent information
3. branch where customer will file complaint
4. Address of customers
5. Complaints

Implement location based queries by using efficient data structure such as spatial indexing.

- Use a priority queue to manage incoming requests and prioritize tasks based on urgency and agent availability.
- Use a relational Database Management system(RDBMS) like PostgreSQL as it supports spatial indexing with PostGIS extension.

Agent Connecting Algorithm

- Use GIS and A* search algorithm to find the shortest path from agents to customers.
- Implement a real-time tracking system to monitor agent locations and dynamically assign tasks based on priority to the customers.
- Choose agents based on current workload, skillset, and previous rating to ensure the best available service is provided to customers.

Scalability :

- Design the system with microservices architecture to allow horizontal scaling by adding more instances as user base grows.
- Implement load balancing and caching mechanisms to distribute complaints
- Use Amazon AWS for elastic scalability.

Location Accuracy:

- Implement an error correction algorithm to mitigate inaccuracy in GPS signals.
- Store additional metadata such as signal strength and timestamp to enhance location accuracy and reliability.

Structure of required tables :

```sql
CREATE TABLE Customers (
    Customer_id INT PRIMARY KEY,
    Customer_name VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    Customer_location VARCHAR(255) NOT NULL ,
    latitude DECIMAL(10,8),
    longitude DECIMAL(11,8)

);
```

```sql
CREATE TABLE Agents (
    Agent_id INT PRIMARY KEY,
    Branch_id INT,
   Agent_NAME VARCHAR(255) ,
   Skill VARCHAR(255),
   Available boolean,
   current_location VARCHAR(255),
   phone VARCHAR(20) NOT NULL,
   FOREIGN KEY(Branch_id) REFERENCES branches (Branch_id),
    latitude DECIMAL(10,8),
    longitude DECIMAL(11,8)
);
```

```sql
CREATE TABLE branches (
    Branch_id INT PRIMARY KEY,
    Branch_NAME VARCHAR(255) ,
    Branch_loc VARCHAR(255),
    phone VARCHAR(20) NOT NULL,
);
```

```sql
CREATE TABLE Complaints (
    complaint_id INT PRIMARY KEY,
    Customer_id INT,
    Product_type VARCHAR(255),
    Branch_id INT,
    complaint_status VARCHAR(255),
    FOREIGN KEY (Customer_id) REFERENCES Customers(Customer_id),
    FOREIGN KEY (Branch_id) REFERENCES branches(Branch_id),

);
```

```sql
CREATE TABLE Assignments (
    Assignment_id INT PRIMARY KEY ,
    Agent_id INT,
    complaint_id INT,
    Assignment_status VARCHAR(255),
    FOREIGN KEY (Agent_id) REFERENCES Agents(Agent_id),
    FOREIGN KEY (complaint_id) REFERENCES Complaints(complaint_id),

);
```

Let’s create some SQL Queries to retrieve information from the Database schema:

1. To select the name, location and phone of all customers who have placed a complaint:

```
SELECT Customers.Customer_name, Customers.Customer_location, Customers.phone
FROM Customers
JOIN Complaints ON Customers.Customer_id = Complaints.Customer_id;
```

2. To select the complaint id, product type and status of all complaints that have been placed by a customers.

```
SELECT Complaints.complaint_id, Complaints.Product_type, Complaints.complaint_status, complaints.Customer_id
FROM complaints
JOIN Customers ON complaints.Customer_id = Customer.Customer_id
```

3. To assign a complaint to an agent based on their skill and location proximity to the customer using the A* algorithm, we'll need to perform several steps:

- Identify agents with the required skill.
- Calculate the distance between each agent's current location and the customer's location using GPS system.

- Apply the A* algorithm to find the shortest path from each eligible agent to the customer's location.
- Select the agent with the shortest path.

```
import psycopg2
from math import radians, sin, cos, sqrt, atan2

# Function to fetch data from the database
def fetch_data():
    try:
        conn = psycopg2.connect('Airtel database.db')
        cursor = conn.cursor()

        # Fetch customer data with their complaints
        cursor.execute("SELECT Customers.Customer_id, Customers.Customer_name, Customers.Customer_location,\
                        Customers.latitude, Customers.longitude, Complaints.Product_type \
                        FROM Customers \
                        JOIN Complaints ON Customers.Customer_id = Complaints.Customer_id;")
        customer_data = cursor.fetchall()

        # Fetch available agents data with their required skills to resolve the product type error
        cursor.execute("SELECT Agents.Agent_id, Agents.Agent_NAME, Agents.current_location,  \
                       Agents.latitude, Agents.longitude, Agents.Skill \
                        FROM Agents \
                        JOIN Assignments ON Agents.Agent_id = Assignments.Agent_id \
                        LEFT JOIN Complaints ON Assignments.complaint_id = Complaints.complaint_id \
                        WHERE Agents.Available = 1\
                       AND Agents.Skill = '<Product_type_error>' ")
        agent_data = cursor.fetchall()

        return customer_data, agent_data
```

```
    except psycopg2.Error as e:
        print("Database error:", e)
    finally:
        if conn:
            conn.close()

def haversine_dist(lat1, long1, lat2, long2):
    lat1,long1,lat2,long2= map(radians,[lat1,long1,lat2,long2])

def haversine_dist(lat1, long1, lat2, long2):
    lat1,long1,lat2,long2= map(radians,[lat1,long1,lat2,long2])

    # Haversine formula
    dlat = lat2 - lat1
    dlon = long2 - long1
    a = sin(dlat / 2)*2 + cos(lat1) * cos(lat2) * sin(dlon / 2)*2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance = 6371 * c  # Radius of Earth in kilometers
    return distance

# A utility function to calculate the g: movement cost to move from agent to goal node
def calculate_g_value(agent, customer):
    # Compute the actual distance from agent's current location to customer's location
    return haversine_dist(agent[3], agent[4], customer[3], customer[4])



# A utility function to calculate the 'h' heuristics  with haversine distance
def calculate_h_value(agent, customer):
        # estimate the remaining distance from the agent's current location to customer's location
        #3 refers to agents and customer latitude and 4 refers to  agents and customer longitutde
    return haversine_dist(agent[3], agent[4], customer[3], customer[4])
```

```python
# A Function to find the shortest path between a given source cell to a destination cell according to A* Search Algorithm
def a_star_search(customer_data, agent_data):
    assignments = {}

    for customer in customer_data:
        min_distance = float('inf')
        closest_agent = None

        for agent in agent_data:
            if agent[5] == customer[5]:  # Match agent's skill with customer's complaint product type
                g_value = calculate_g_value(agent, customer)
                h_value = calculate_h_value(agent, customer)
                total_cost = g_value + h_value
                if total_cost < min_distance:
                    min_distance = total_cost
                    closest_agent = agent

        if closest_agent:
            assignments[customer]= closest_agent

        print(f"Assign Agent {closest_agent[1]} to Customer {customer[1]} at {customer[2]}")

# Fetch data from the database
customer_data, agent_data = fetch_data()

# If data is fetched successfully, proceed with A* search
if customer_data and agent_data:
    a_star_search(customer_data, agent_data)
else:
    print("Inform to branch that no agent is available, contact to next nearest branch")
```

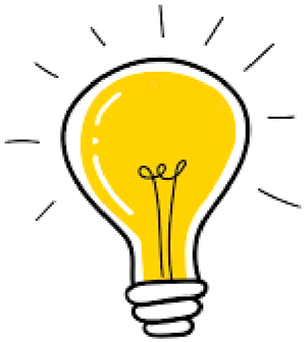
4. Now the agent with shortest path is selected and assigned the task.

```
SELECT *
FROM Agents
WHERE Agent_id = closest_agent;
```

5. To retrieve assigned Agent and Complaint information for a specific Branch.

```
SELECT Agents.*, Leads.*
FROM Agents
JOIN Assignments ON Agents.Agent_id = Assignments.Agent_id
JOIN Complaints ON Assignments.complaint_id = Complaints.complaint_id
WHERE Agents.Branch_id = '<Branch_id>';
```

IDEAS :



- If the current branch where the complaint is filed, fails to provide an agent, the request must be sent to the nearest branch to send a skilled agent.
- Rating of the agent is also an important factor that can be added in database preparation.
- We can also consider the following factors for further improving the efficiency of the above algorithm and better customer experience:

1. Fuel Capacity
2. Speed of the agent’s vehicle
3. Type of vehicle used

Links:

https://medium.com/towards-data-engineering/database-design-for-a-food-delivery-app-like-zomato-swiggy-86c16319b5c5

https://sandesh-deshmane.medium.com/architecture-and-design-principles-for-online-food-delivery-system-33bfda73785d

Thank You !!!