NextgenLead_5

OVERALL PROCESS FLOW

1. **Customer Request** → Customer submits a callback request.
2. **Customer Care Follows Up** → Customer care updates the system with additional details.
3. **Dealer Information** → Dealer retrieves customer details using the interaction ID.
4. **Dealer Calls** → Dealer calls the customer (manual).
5. **Customer Visit** → The customer visits the showroom (manual).
6. **Salesperson Updates** → The salesperson records the visit and next steps.
7. **Head Office Report** → The head office generates reports to track all customer interactions

Step 1: Create the `customer_interactions` table

```
CREATE TABLE customer_interactions (
    interaction_id SERIAL PRIMARY KEY,  -- Unique identifier for each interaction, aut
    customer_name VARCHAR(100) NOT NULL,  -- Customer's name
    phone_number VARCHAR(20) NOT NULL,  -- Customer's phone number
    request_type VARCHAR(50) NOT NULL,  -- Type of request (e.g., product inquiry, ser
    preferred_time TIMESTAMP,  -- Preferred callback time
    additional_info TEXT,  -- Additional info from the customer
    dealer_name VARCHAR(100),  -- Assigned dealer's name
    dealer_phone_number VARCHAR(20),  -- Dealer's phone number
    interaction_summary TEXT,  -- Summary of dealer-customer interaction
    next_steps VARCHAR(255),  -- Agreed next steps for follow-up
    customer_status VARCHAR(50) NOT NULL DEFAULT 'Pending',  -- Status (e.g., Pending,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  -- Timestamp of record creation
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  -- Timestamp of last update to the
);


ALTER TABLE customer_interactions
ADD COLUMN preferred_dealer VARCHAR(100);
```

EXPLANATION OF FIELDS:

- **`interaction_id`**: A `SERIAL` column that auto-increments and serves as the primary key.
- **`customer_name`**: Stores the name of the customer (e.g., John Doe).
- **`phone_number`**: Stores the customer's phone number.
- **`request_type`**: The type of the request (e.g., product inquiry, service issue).
- **`preferred_time`**: The time the customer prefers for a callback (timestamp format).
- **`additional_info`**: Any extra information provided by the customer.
- **`dealer_name`**: The name of the dealer who is assigned to follow up with the customer.
- **`dealer_phone_number`**: The phone number of the assigned dealer.

- **`interaction_summary`**: A summary of the interaction between the dealer and the customer.
- **`next_steps`**: The next steps agreed upon during the interaction.
- **`customer_status`**: The current status of the interaction (defaults to "Pending").
- **`created_at`**: Automatically records when the row is first created.
- **`updated_at`**: Automatically updates the timestamp when the row is modified.

app.py

```
import psycopg2
from flask import Flask, request, jsonify
from flask_restful import Api
from sqlalchemy import create_engine, Column, String, Integer, DateTime, Text, TIMESTA
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from sqlalchemy.pool import NullPool

app = Flask(__name__)
api = Api(app)

# Database setup
Base = declarative_base()
database_url = "postgresql://postgres:1234@localhost:5432/postgres"
engine = create_engine(database_url, echo=True, poolclass=NullPool)
Session = sessionmaker(bind=engine)
session = Session()

# Define the customer_interactions model
class CustomerInteraction(Base):
    __tablename__ = 'customer_interactions'

    interaction_id = Column(Integer, primary_key=True, autoincrement=True)
    customer_name = Column(String(100), nullable=False)
    phone_number = Column(String(20), nullable=False)
    request_type = Column(String(50), nullable=False)
    preferred_time = Column(DateTime)
    additional_info = Column(Text)
    dealer_name = Column(String(100))
    dealer_phone_number = Column(String(20))
    interaction_summary = Column(Text)
    next_steps = Column(String(255))
    customer_status = Column(String(50), nullable=False, default='Pending')
    created_at = Column(TIMESTAMP, server_default=func.now())
    updated_at = Column(TIMESTAMP, server_default=func.now(), onupdate=func.now())

# Create the table if it doesn't exist
Base.metadata.create_all(engine)

# POST: Submit Callback Request (Step 1)
@app.route('/api/callback-request', methods=['POST'])
def create_callback_request():
```

```
    data = request.json
    new_request = CustomerInteraction(
        customer_name=data.get('customer_name'),
        phone_number=data.get('phone_number'),
        request_type=data.get('request_type'),
        preferred_time=data.get('preferred_time'),
        customer_status='Pending'
    )
    session.add(new_request)
    session.commit()
    return jsonify({
        "status": "success",
        "message": "Callback request created",
        "request_id": new_request.interaction_id
    })

# PUT: Update Callback Request (Step 2)
@app.route('/api/callback-request/<int:request_id>', methods=['PUT'])
def update_callback_request(request_id):
    data = request.json
    # Find the existing callback request by the request_id
    callback_request = session.query(CustomerInteraction).filter_by(interaction_id=rec

    if not callback_request:
        return jsonify({
            "status": "error",
            "message": f"Request with id {request_id} not found"
        }), 404

    # Update fields based on the provided data
    callback_request.customer_name = data.get('customer_name', callback_request.custom
    callback_request.additional_info = data.get('additional_info', callback_request.ad

    session.commit()

    return jsonify({
        "status": "success",
        "message": "Request updated successfully"
    })

# GET: Retrieve Customer Info for Dealers (Step 3)
@app.route('/api/customer-info/<int:interaction_id>', methods=['GET'])
def get_customer_info(interaction_id):
    # Query to fetch customer information for a specific interaction_id
    customer = session.query(CustomerInteraction).filter_by(interaction_id=interactior

    if not customer:
        return jsonify({
            "status": "error",
            "message": f"No customer information found for interaction ID {interactior
        }), 404

    # Serialize customer data
    customer_data = {
        "customer_name": customer.customer_name,
```

```
        "phone_number": customer.phone_number,
        "request_type": customer.request_type,
        "additional_info": customer.additional_info,
        "preferred_time": customer.preferred_time,
        "dealer_name": customer.dealer_name
    }

    return jsonify(customer_data)

# PUT: Record Dealer-Customer Interaction (Step 6)
@app.route('/api/customer-interaction/<int:interaction_id>', methods=['PUT'])
def update_customer_interaction(interaction_id):
    data = request.json
    interaction = session.query(CustomerInteraction).filter_by(interaction_id=interact

    if not interaction:
        return jsonify({
            "status": "error",
            "message": f"Interaction with id {interaction_id} not found"
        }), 404

    # Update interaction details
    interaction.interaction_summary = data.get('interaction_summary', interaction.inte
    interaction.next_steps = data.get('next_steps', interaction.next_steps)
    interaction.customer_status = data.get('customer_status', interaction.customer_sta

    session.commit()

    return jsonify({
        "status": "success",
        "message": "Interaction details updated successfully"
    })


@app.route('/api/sales-interaction/<int:interaction_id>', methods=['PUT'])
def record_sales_interaction(interaction_id):
    data = request.json
    interaction = session.query(CustomerInteraction).filter_by(interaction_id=interact

    if not interaction:
        return jsonify({
            "status": "error",
            "message": f"Interaction with id {interaction_id} not found"
        }), 404

    # Update salesperson interaction details
    interaction.interaction_summary = data.get('interaction_summary', interaction.inte
    interaction.next_steps = data.get('next_steps', interaction.next_steps)
    interaction.customer_status = data.get('customer_status', interaction.customer_sta

    session.commit()

    return jsonify({
        "status": "success",
        "message": "Salesperson interaction details recorded successfully"
```

```
    })


@app.route('/api/reports/customer-status', methods=['GET'])
def get_customer_status_report():
    interactions = session.query(CustomerInteraction).all()

    # Serialize interaction data for reporting
    report_data = [{
        "customer_name": interaction.customer_name,
        "status": interaction.customer_status,
        "interaction_summary": interaction.interaction_summary,
        "dealer_name": interaction.dealer_name
    } for interaction in interactions]

    return jsonify(report_data)

if __name__ == '__main__':
    app.run(debug=True)
```

1. REQUEST FOR RADHA:

- **Endpoint**: `/api/callback-request`
- **Method**: `POST`
- **Request Body (JSON)**:

```
json
Copy code
{
"customer_name": "Radha",
"phone_number": "+1234567890",
"request_type": "Product Inquiry",
"preferred_time": "2024-09-08 10:00:00"
}
```

2. REQUEST FOR KRISHNA:

- **Endpoint**: `/api/callback-request`
- **Method**: `POST`
- **Request Body (JSON)**:

```
json
Copy code
{
"customer_name": "Krishna",
"phone_number": "+0987654321",
"request_type": "Service Issue",
"preferred_time": "2024-09-09 11:00:00"
}
```

step1

step 2 Put Method

http://127.0.0.1:5000/api/callback-request/1

```
@app.route('/api/callback-request/<int:request_id>', methods=['PUT'])
def update_callback_request(request_id):
    data = request.json
    # Find the existing callback request by the request_id
    callback_request = session.query(CustomerInteraction).filter_by(interaction_id=rec

    if not callback_request:
        return jsonify({
            "status": "error",
            "message": f"Request with id {request_id} not found"
        }), 404

    # Update fields based on the provided data
    callback_request.customer_name = data.get('customer_name', callback_request.custom
    callback_request.additional_info = data.get('additional_info', callback_request.ad

    session.commit()

    return jsonify({
        "status": "success",
        "message": "Request updated successfully"
    })
```

step 3 Now get info by get method

http://127.0.0.1:5000/api/customer-info/1

STEP 4: UPDATE INTERACTION DETAILS

CREATE AN ENDPOINT TO RECORD AND UPDATE THE DETAILS OF THE INTERACTION BETWEEN THE DEALER AND THE CUSTOMER. THIS STEP WILL BE USEFUL FOR TRACKING THE OUTCOME OF THE FOLLOW-UP CALL AND PLANNING NEXT STEPS.

http://127.0.0.1:5000/api/customer-interaction/1

STEP 5: CUSTOMER VISITS SHOWROOM (IF AGREED)

THIS STEP INVOLVES THE CUSTOMER VISITING THE SHOWROOM FOR FURTHER DISCUSSIONS OR PRODUCT DEMONSTRATIONS. SINCE THIS STEP IS ALSO MANUAL AND INVOLVES NO API, YOU'LL HANDLE IT WITH THE SYSTEM BY ENSURING YOU HAVE AN ENDPOINT FOR TRACKING CUSTOMER VISITS IF NEEDED.

STEP 6: RECORD SALESPERSON INTERACTION

CREATE AN ENDPOINT TO RECORD DETAILS OF THE INTERACTION BETWEEN THE SALESPERSON AND THE CUSTOMER AFTER THE SHOWROOM VISIT. THIS WILL HELP YOU TRACK AND MANAGE CUSTOMER INTERACTIONS EFFECTIVELY.

http://127.0.0.1:5000/api/sales-interaction/1
Flask Endpoint for Recording Salesperson Interaction

STEP 7: GENERATE REPORT AT HEAD OFFICE

CREATE AN ENDPOINT TO GENERATE A REPORT THAT PROVIDES THE STATUS OF ALL CUSTOMER INTERACTIONS. THIS HELPS THE HEAD OFFICE VIEW AND ANALYZE CUSTOMER ENGAGEMENT.

Flask Endpoint for Generating Reports
http://127.0.0.1:5000/api/reports/customer-status

IN SHORTS IT IS DEFINED AS :

STEP 1: SUBMIT CALLBACK REQUEST

- **Action**: The customer submits a request for a callback via web or mobile for product inquiries or service issues.
- **API Endpoint**: `/api/callback-request`
- **HTTP Method**: `POST`
- **Implementation**: A new callback request is created and saved in the system using the customer's details.
- **Postman Example (POST)**:

```
json
Copy code
{
"customer_name": "Radha",
"phone_number": "+1234567890",
"request_type": "Product Inquiry",
"preferred_time": "2024-09-08 10:00:00"
}
```

STEP 2: CUSTOMER CARE FOLLOWS UP

- **Action**: A customer care representative calls the customer and updates the request with additional details.
- **API Endpoint**: `/api/callback-request/<int:request_id>`
- **HTTP Method**: `PUT`
- **Implementation**: Updates an existing callback request with additional information like the dealer's name or interaction details.
- **Postman Example (PUT)**:

```
json
Copy code
{
"additional_info": "Interested in car model ABC, looking for a test drive",
"preferred_dealer": "ABC Showroom"
}
```

STEP 3: INFORMATION SHARED WITH DEALERS

- **Action**: The updated customer information is retrieved based on `interaction_id` and shared with the dealer.
- **API Endpoint**: `/api/customer-info/<int:interaction_id>`
- **HTTP Method**: `GET`
- **Implementation**: Retrieves customer information based on `interaction_id`.
- **Postman Example (GET)**:
  Request URL: `http://localhost:5000/api/customer-info/1`
  Response Example:

```
json
Copy code
{
"customer_name": "Radha",
"phone_number": "+1234567890",
"request_type": "Product Inquiry",
"additional_info": "Interested in car model ABC, looking for a test drive",
"preferred_time": "2024-09-08 10:00:00"
}
```

STEP 4: DEALER CALLS CUSTOMER

- **Action**: The dealer calls the customer to follow up.
- **Implementation**: No API needed for this step (manual step). However, the dealer can use the information retrieved in Step 3 to call the customer.

STEP 5: CUSTOMER VISITS SHOWROOM (IF AGREED)

- **Action**: The customer visits the dealership for further discussions or product demonstrations.
- **Implementation**: No API required for this step as it happens offline.

STEP 6: SALESPERSON RECORDS CONVERSATION

- **Action**: The salesperson records the details of the conversation and updates the system.
- **API Endpoint**: `/api/sales-interaction/<int:interaction_id>`
- **HTTP Method**: `PUT`
- **Implementation**: The salesperson records the interaction details, such as conversation summary, next steps, and the updated customer status.
- **Postman Example (PUT)**:

```
json
Copy code
{
"interaction_summary": "Customer visited showroom, test drove car model ABC.",
"next_steps": "Follow-up call in 2 weeks.",
"customer_status": "In Progress"
}
```

STEP 7: REPORT AT HEAD OFFICE

- **Action**: The head office generates a report to view the status of all customers and their interactions with dealerships.
- **API Endpoint**: `/api/reports/customer-status`
- **HTTP Method**: `GET`
- **Implementation**: Fetches all customer interactions for reporting.
- **Postman Example (GET)**:
  Request URL: `http://localhost:5000/api/reports/customer-status`
  Response Example:

```
json
Copy code
[
{
"customer_name": "Radha",
"status": "Follow-up Scheduled",
"interaction_summary": "Customer visited showroom, test drove car model ABC.",
"dealer_name": "ABC Showroom"
},
{
"customer_name": "Krishna",
"status": "Completed",
"interaction_summary": "Customer purchased car model XYZ.",
"dealer_name": "XYZ Showroom"
}
]
```