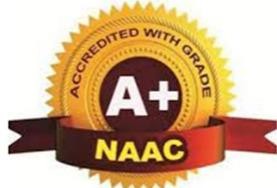


**BASAVARAJESWARI GROUP OF INSTITUTIONS**

**BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT**

NBA and NACC Accredited Institution\*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi) "JnanaGangotri" Campus, No.873/2,**

**Ballari-Hosapete Road, Allipur, Ballari-583 104 (Karnataka) (India) Ph:08392 – 237100 / 237190, Fax: 08392 – 237197**

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**A**

**Python-Project Report On**

## “CUSTOMER RELATIONSHIP MANAGEMENT SYSTEM”

***Submitted By***

**Ankitha S 3BR23AI011**

**Arpitha B N 3BR23AI012**

**Asha Thasleem 3BR23AI013**

**Bhavani Hemagudda 3BR23AI026**

**Hampi Gayathri 3BR23AI054**

## Under the Guidance of

Ms. Neha

# Visvesvaraya Technological University

### Belagavi, Karnataka

****

**2024-2025**

**BASAVARAJESWARI GROUP OF INSTITUTIONS**

**BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT**

NBA and NACC Accredited Institution\*

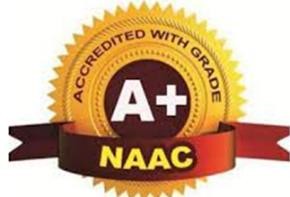
**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi) "JnanaGangotri" Campus, No.873/2,**

**Ballari-Hosapete Road, Allipur, Ballari-583 104 (Karnataka) (India) Ph:08392 – 237100 / 237190, Fax: 08392 – 237197**



**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

****

**CERTIFICATE**

This is to certify that **Seminar** entitled **“CUSTOMER RELATIONSHIP MANAGEMENT SYSTEM”** has been successfully completed by **Team Members Ankita S, Arpitha B N, Asha Thasleem, Bhavani Hemagudda, Hampi Gayathri,** bearing **USNs 3BR23AI011, 3BR23AI012, 3BR23AI013, 3BR23AI026, 3BR23AI054**

Bonafide students of Ballari Institute of Technology and Management, Ballari, for the partial fulfillment of the requirements for the **in Python Internship** during the academic year 2025-2026.

The participant demonstrated enhanced skills in observation, communication, critical thinking, teamwork, and presentation. The visit and seminar also fostered professionalism, confidence, and effective time management.

**Signature of Head of the Department Dr. B M Vidyavathi Department of AIML,**

**BITM, Ballari**

## ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of Mini-project work on “**CUSTOMER RELATIONSHIP MANAGEMENT SYSTEM**” would be

incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement, and support crowned our efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of this project work.

We are extremely grateful to our respective guide **Neha,** for her noble gesture, support, coordination and valuable suggestions given to us in completing the project work. We also thank **Dr. B M Vidyavathi**, H.O.D. Department of AIML, for her coordination and valuable suggestions given to us in completing the project. We also thank the Principal, Management, and non-teaching staff for their coordination and valuable suggestions given to us in completing the project work.

# ABSTRACT

The Customer Relationship Management (CRM) System is a software application designed to manage a company’s interactions with current and potential customers. The system allows users to store and maintain customer information, track interactions, record purchases, and schedule follow-ups efficiently.

This project implements key CRUD (Create, Read, Update, Delete) operations for managing customer data, along with features to log customer interactions and purchases, calculate lead scores, and schedule future follow-ups. The lead score helps in identifying high-priority customers based on their engagement and purchase history.

The CRM system is menu-driven, enabling users to dynamically add, update, delete, or view customers, log interactions and purchases, and manage follow-up dates. It provides a simple yet effective way for businesses to improve customer engagement, streamline sales processes, and prioritize leads.

This project is implemented in Python, using in-memory data storage for simplicity, making it easy to use and extend. The system demonstrates core CRM functionalities and can be further enhanced by integrating with databases for persistent storage or web-based interfaces for accessibility.

# INTRODUCTION OF THE PROJECT

Customer Relationship Management (CRM) refers to the strategies, practices, and technologies that companies use to manage and analyze customer interactions and data throughout the customer lifecycle. The goal of CRM is to improve business relationships with customers, assist in customer retention, and drive sales growth.

## Customer-Relationship Management project information

1. It is user-friendly - Provides an intuitive interface that requires minimal training for sales teams to use effectively
2. It helps reduce workload on employees - Automates repetitive tasks like data entry, follow-up reminders, and report generation
3. It streamlines business processes - Reduces paperwork, manual data entry, communication gaps, and coordination efforts
4. It improves data accuracy and quality - Eliminates duplication, inconsistencies, and incomplete customer information through centralized data management
5. It facilitates collaboration among team members - Enables sales, marketing, and support teams to share customer insights and coordinate efforts
6. It helps prevent errors and missed opportunities - Minimizes mistakes like duplicate contacts, forgotten follow-ups, and miscommunication with customer.

# Features of Customer-Relationship Management

### Manage Customers

In this feature manage all the information of Customers and each Customer's interaction history with the company. Track complete customer lifecycle from lead to loyal customer with detailed profiling and segmentation capabilities.

### Manage Leads & Opportunities

CRM makes it possible to get all of a lead's information through a system with just a few clicks. The user can see information like the lead's interaction history, current status, sales representative assigned, deal value, conversion probability, and much more.

### Sales Pipeline Management

Track every deal through customizable sales stages from initial contact to closed- won or closed-lost. Visual pipeline dashboard shows where each opportunity stands, helping sales teams prioritize efforts and forecast revenue accurately.

### Interaction Tracking

Automatically record all customer interactions including emails, calls, meetings, and notes. Maintain complete communication history to ensure continuity and personalized engagement across all touchpoints.

### Automated Lead Scoring

Intelligent scoring algorithm automatically ranks leads based on engagement level, demographic data, and behaviour patterns. Identify hot leads that need immediate attention and nurture colder leads with targeted campaigns.

### Follow-up & Task Management

Schedule and track customer follow-ups with automated reminders. Assign tasks to team members, set priorities, and ensure no customer interaction opportunity is missed through systematic task management.

1. **Reporting & Analytics**

Comprehensive dashboards and reports provide insights into sales performance, team productivity, conversion rates, and customer behaviour. Make data-driven decisions with real-time analytics and customizable reports.

# MODULE DESCRIPTION

Module Name: crm\_system.py Purpose:

This module implements a basic Customer Relationship Management (CRM) system that allows the user to manage customer data, track interactions and purchases, schedule follow-ups, and calculate lead scores. It provides a command- line interface to perform CRUD operations and view customer details.

Key Components

1. Data Structures

customers\_data – Dictionary to store customer objects using customer\_id as the key.

followup\_data – Dictionary to store scheduled follow-up dates for customers.

1. Classes

Customer – Represents a customer.

Attributes:

customer\_id: Unique ID for each customer. name: Customer name.

address: Customer address. email: Customer email.

phone: Customer phone number. interaction\_history: List of customer interactions. purchase\_history: List of customer purchases.

Note: init should be corrected to init for proper object initialization.

1. CRUD Functions

create\_customer(customer\_id, name, address, email, phone) – Adds a new customer.

get\_customer\_details(customer\_id) – Retrieves a customer object.

update\_customer(customer\_id, name, address, email, phone) – Updates customer details.

delete\_customer(customer\_id) – Deletes a customer record.

1. Customer Interaction & Purchase Logging

log\_customer\_interaction(customer\_id, interaction) – Logs interactions with the customer.

log\_customer\_purchase(customer\_id, purchase\_amount) – Logs purchases with amount.

1. Follow-up Scheduling

schedule\_followup(customer\_id, followup\_date) – Schedules a future follow-up date.

1. Display Functions

display\_customer(customer\_id) – Displays full customer details. display\_all\_customers() – Displays details of all customers in the system.

1. Menu System

main\_menu() – Interactive command-line menu to: Add Customer

Delete Customer Update Customer View Customer Display All Customers Log Interaction

Log Purchase Schedule Follow-up Exit

# ALGORITHM

1. Initialize System Import datetime module.

Initialize two global data structures:

customers\_data → dictionary to store customer objects. followup\_data → dictionary to store follow-up dates by customer ID.

1. Define Customer Class Attributes:

customer\_id, name, address, email, phone interaction\_history → list of interactions purchase\_history → list of purchase records Method:

init to initialize all attributes. Define CRM Functions

1. create\_customer(customer\_id, name, address, email, phone)

If customer\_id exists in customers\_data, print warning and return. Else, create a Customer object and store it in customers\_data.

1. get\_customer\_details(customer\_id)

Return the customer object if it exists, else return None.

1. update\_customer(customer\_id, name, address, email, phone) If customer exists:

Update only provided (non-None) fields. Else, print error.

1. delete\_customer(customer\_id) If customer exists:

Remove from customers\_data and followup\_data.

Else, print error.

1. log\_customer\_interaction(customer\_id, interaction) If customer exists:

Append interaction to interaction\_history.

1. log\_customer\_purchase(customer\_id, purchase\_amount) If customer exists:

Append a dictionary {'amount': purchase\_amount} to purchase\_history.

1. schedule\_followup(customer\_id, followup\_date)

If customer exists and followup\_date is today or in the future: Save in followup\_data.

1. display\_customer(customer\_id) If customer exists:

Print all customer details, interactions, purchases, lead score, and follow-up date.

1. display\_all\_customers()

Iterate over all entries in customers\_data and call display\_customer().

1. Main Menu Loop Loop While True:

Display the main menu options:

Add, Delete, Update, View, Display All, Log Interaction, Log Purchase, Schedule Follow-up, Exit.

Read user choice as input. Based on the choice:

If Choice is '1':

Prompt user for all customer details. Call create\_customer(...).

'2':

Prompt for customer ID.

Call delete\_customer(...). '3':

Prompt for customer ID and any fields to update. Pass non-empty fields to update\_customer(...). '4':

Prompt for customer ID. Call display\_customer(...). '5':

Call display\_all\_customers(). '6':

Prompt for customer ID and interaction text. Call log\_customer\_interaction(...).

'7':

Prompt for customer ID and purchase amount. Call log\_customer\_purchase(...).

'8':

Prompt for customer ID and follow-up date (YYYY-MM-DD). Parse the date and call schedule\_followup(...).

Handle invalid date formats. '9':

Print exit message.Break the loop. Else:

Print invalid choice message.

1. Program Entry Point If \_name\_ == "\_main\_":

Call main\_menu() to start the system.

# CODE

import datetime # Data Structures

customers\_data = {} followup\_data = {} # Customer Class class Customer:

def \_init\_(self, customer\_id, name, address, email, phone): self.customer\_id = customer\_id

self.name = name self.address = address self.email = email self.phone = phone self.interaction\_history = [] self.purchase\_history = []

# CRUD Functions

def create\_customer(customer\_id, name, address, email, phone): if customer\_id in customers\_data:

print("Customer ID already exists!") return None

customer = Customer(customer\_id, name, address, email, phone) customers\_data[customer\_id] = customer

print(f"Customer {name} added successfully.") return customer

def get\_customer\_details(customer\_id):

return customers\_data.get(customer\_id, None

def update\_customer(customer\_id, name=None, address=None, email=None, phone=None):

customer = customers\_data.get(customer\_id, None) if not customer:

print("Customer not found!") return False

if name: customer.name = name

if address: customer.address = address if email: customer.email = email

if phone: customer.phone = phone print("Customer updated successfully.") return True

def delete\_customer(customer\_id): if customer\_id in customers\_data:

del customers\_data[customer\_id] followup\_data.pop(customer\_id, None) print("Customer deleted successfully.") return True

print("Customer not found!") return False

# Interaction & Purchase Logging

def log\_customer\_interaction(customer\_id, interaction): customer = customers\_data.get(customer\_id, None) if not customer:

print("Customer not found!") return False

customer.interaction\_history.append(interaction)

print("Interaction logged.") return True

def log\_customer\_purchase(customer\_id, purchase\_amount): customer = customers\_data.get(customer\_id, None)

if not customer: print("Customer not found!") return False

customer.purchase\_history.append({'amount': purchase\_amount}) print("Purchase logged.")

return True # Lead Scoring

def generate\_lead\_score(customer\_id):

customer = customers\_data.get(customer\_id, None) if not customer:

print("Customer not found!") return None

interaction\_score = len(customer.interaction\_history)

purchase\_score = sum(p['amount'] for p in customer.purchase\_history) lead\_score = interaction\_score + purchase\_score

return lead\_score

# Follow-up Scheduling

def schedule\_followup(customer\_id, followup\_date):

if customer\_id in customers\_data and followup\_date >= datetime.date.today(): followup\_data[customer\_id] = followup\_date

print("Follow-up scheduled.") return True

print("Invalid follow-up date or customer not found!")

return False

# Display Customer Details

def display\_customer(customer\_id):

customer = customers\_data.get(customer\_id, None) if not customer:

print("Customer not found!") return

print(f"\nCustomer ID: {customer.customer\_id}") print(f"Name: {customer.name}") print(f"Address: {customer.address}") print(f"Email: {customer.email}")

print(f"Phone: {customer.phone}") print(f"Interactions: {customer.interaction\_history}") print(f"Purchases: {customer.purchase\_history}")

print(f"Lead Score: {generate\_lead\_score(customer\_id)}")

print(f"Follow-up Date: {followup\_data.get(customer\_id, 'Not scheduled')}\n") def display\_all\_customers():

if not customers\_data:

print("No customers available.") return

for customer\_id in customers\_data: display\_customer(customer\_id)

# Menu Loop

def main\_menu(): while True:

print("\n--- CRM SYSTEM MENU ---")

print("1. Add Customer")

print("2. Delete Customer") print("3. Update Customer") print("4. View Customer") print("5. Display All Customers") print("6. Log Interaction") print("7. Log Purchase")

print("8. Schedule Follow-up") print("9. Exit")

choice = input("Enter your choice (1-9): ") if choice == '1':

cid = int(input("Enter Customer ID: ")) name = input("Enter Name: ")

address = input("Enter Address: ") email = input("Enter Email: ") phone = input("Enter Phone: ")

create\_customer(cid, name, address, email, phone) elif choice == '2':

cid = int(input("Enter Customer ID to delete: ")) delete\_customer(cid)

elif choice == '3':

cid = int(input("Enter Customer ID to update: "))

name = input("Enter new Name (or press Enter to skip): ") address = input("Enter new Address (or press Enter to skip): ") email = input("Enter new Email (or press Enter to skip): ") phone = input("Enter new Phone (or press Enter to skip): ")

update\_customer(cid, name or None, address or None, email or None, phone or None

elif choice == '4':

cid = int(input("Enter Customer ID to view: ")) display\_customer(cid)

elif choice == '5': display\_all\_customers()

elif choice == '6':

cid = int(input("Enter Customer ID for interaction: ")) interaction = input("Enter interaction details: ") log\_customer\_interaction(cid, interaction)

elif choice == '7':

cid = int(input("Enter Customer ID for purchase: ")) amount = float(input("Enter purchase amount: ")) log\_customer\_purchase(cid, amount)

elif choice == '8':

cid = int(input("Enter Customer ID for follow-up: ")) date\_input = input("Enter follow-up date (YYYY-MM-DD): ") try:

f\_date = datetime.datetime.strptime(date\_input, "%Y-%m-%d").date() schedule\_followup(cid, f\_date)

except ValueError: print("Invalid date format!")

elif choice == '9':

print("Exiting CRM System. Goodbye!") break

else:

print("Invalid choice. Try again!"

if \_name\_ == "\_main\_": main\_menu()

# OUTPUT

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 1 Enter Customer ID: 1 Enter Name: bhavani Enter Address: kampli

Enter Email: [bhavani@gmail.com](mailto:bhavani@gmail.com) Enter Phone: 1234567890

Customer bhavani added successfully.

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 6

Enter Customer ID for interaction: 1

Enter interaction details: called for product availability Interaction logged.

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 7

Enter Customer ID for purchase: 1 Enter purchase amount: 250 Purchase logged.

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 8

Enter Customer ID for follow-up: 1

Enter follow-up date (YYYY-MM-DD): 2025-09-30 Follow-up scheduled.

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 4 Enter Customer ID to view: 1 Customer ID: 1

Name: bhavani Address: kampli

Email: [bhavani@gmail.com](mailto:bhavani@gmail.com) Phone: 1234567890

Interactions: ['called for product availability'] Purchases: [{'amount': 250.0}]

Lead Score: 251.0

Follow-up Date: 2025-09-30

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 5

Customer ID: 1 Name: bhavani Address: kampli

Email: [bhavani@gmail.com](mailto:bhavani@gmail.com) Phone: 1234567890

Interactions: ['called for product availability'] Purchases: [{'amount': 250.0}]

Lead Score: 251.0

Follow-up Date: 2025-09-30

--- CRM SYSTEM MENU ---

1. Add Customer
2. Delete Customer
3. Update Customer
4. View Customer
5. Display All Customers
6. Log Interaction
7. Log Purchase
8. Schedule Follow-up
9. Exit

Enter your choice (1-9): 9 Exiting CRM System. Goodbye!

# CONCLUSION

In conclusion, this project successfully demonstrates the development of a simple, console-based Customer Relationship Management (CRM) system using Python. It includes essential functionalities such as creating, viewing, updating, and deleting customer records, along with logging customer interactions and purchases, calculating lead scores, and scheduling follow-ups. The project makes effective use of object-oriented programming principles and Python’s built-in data structures, providing a clear and modular structure. Through this system, users can manage customer data efficiently via a user-friendly menu interface

# REFERENCES

* + Python Classes & Objects – https://docs.python.org/3/tutorial/classes.html
  + Python Data Structures-https://docs.python.org/3/tutorial/datastructures.html
  + CRUD Concept-https://en.wikipedia.org/wiki/Create,\_read,\_update\_and\_delete
  + CRM Basics – https:/[/www.ibm.com/topics/crm](http://www.ibm.com/topics/crm)