**Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology**

**(Deemed to be University Estd. u/s 3 of UGC Act, 1956)**

**School of Computing**A blue sign with white text

AI-generated content may be incorrect.

**B.Tech. – Computer Science and Engineering**

A close up of numbers

AI-generated content may be incorrect.

Academic Year: 2025–2026

SDG 4: Quality Education  
Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S6

DBMS PROJECT REPORT

Title: A GIFT COUPON APPLICTATION HANDILES OFFERS PAYMENT-RELATED

**Submitted by:**

| **VTUNO** | **REGISTER NUMBER** | **STUDENT NAME** |
| --- | --- | --- |
| VTU28814 | 24UECS0791 | SHANMUKA SIVA N |
| VTU28830 | 24UECS0380 | VASU.A |
| VTU28886 | 24UECS1419 | PAVAN KISHORE .V |
| VTU28901 | 24UECS0006 | BHARGAV SAI RATNAGI.A |
| VTU28914 | 24UECS0545 | REDDY SANTHOSH .D |
| VTU28978 | 24UECS0939 | SURYA KARTHIKEYA .T |

Under the guidance of:  
Associate Professor: Dr. AMBIKA .S

**INDEX PAGE**

1. Introduction …………………………………………… 3

2. Problem Statement …………………………………………… 4

3. Objectives …………………………………………… 5

4. System Requirements …………………………………………… 6

5. System Analysis and Design……………………………………… 6

6. ER Diagram (Conceptual Design) ..……………………………… 7

7. Schema Design (Oracle)…………………………………………… 9

8. Normalization …………………………………………… 13

9. Implementation (SQL Queries)…………………………………… 13

10. Input and Output …………………………………………… 14

11. Integration with MongoDB (NoSQL)…………………………… 15

12. Results and Discussion ..…………………………………… 19

13. Conclusion …………………………………………… 20

14. References …………………………………………… 20

**1. Introduction :**

In this use case, the Gift Coupon Application manages offers and payment-related information, which involves handling a large volume of transactional data. Because financial transactions demand accuracy, integrity, and consistency, the choice of database system becomes critical.

A relational database management system (RDBMS) is chosen due to its strong ACID (Atomicity, Consistency, Isolation, Durability) properties. These ensure that every transaction—such as issuing, redeeming, or expiring a gift coupon—executes completely or not at all, maintains data consistency, and is durable even in case of system failures.

While many NoSQL databases provide eventual consistency and horizontal scalability, they often compromise on strict transactional guarantees. However, modern relational databases such as MySQL Cluster, PostgreSQL Cluster, or CockroachDB provide the ability to scale horizontally without losing ACID compliance, making them ideal for this application.

The system will be designed using normalized relational tables to eliminate redundancy and maintain data integrity. Normalization helps avoid anomalies during data updates, deletions, or insertions. In practical applications like a gift coupon system, normalization is typically applied up to the Third Normal Form (3NF) or Boyce-Codd Normal Form (BCNF).

**These forms ensure:**

Each table represents a single entity (e.g., Customer, Coupon, Transaction), Data dependencies are logical and minimal, The design supports concurrent transactions without deadlocks, when proper indexing and isolation levels are maintained.

Thus, by using a horizontally scalable relational database cluster that sustains normalization up to 3NF/BCNF, the gift coupon application can efficiently handle large-scale reads and wri-tes while ensuring transactional safety and maintaining.

**2. Problem Statement :**

In modern e-commerce systems, businesses frequently provide gift coupons and promotional offers to encourage customer loyalty and boost sales. However, manual tracking of coupons, their expiry dates, usage status, and associated payment information can lead to errors, data redundancy, and inefficiency.

To overcome these challenges, there is a need for a database-based Gift Coupon Application that can manage offers, customers, and payment-related information effectively. The system should allow businesses to store, retrieve, and update coupon and payment data efficiently while ensuring security and accuracy.

**3. Objectives :**

1. To design a relational database for managing gift coupons, customers, and payment transactions.

2. To maintain accurate records of coupon creation, validity, and usage.

3. To link customer and payment information for clear tracking of coupon redemption.

4. To retrieve useful insights such as: Active and expired offers Coupons used by specific customers Total spending by customers

5. To ensure data consistency and integrity using SQL constraints like primary keys and foreign keys.

**4. System Requirements :**

**Hardware Requirements:**

- Processor: Intel i5 or higher

- RAM: 8 GB or more

- Hard Disk: 250 GB

**Software Requirements:**

- OS: Windows 10 or Linux

- Database: Oracle 12c or above

- Front-End: Java / Web Interface

- NoSQL: MongoDB 6.0

- Tools: Oracle SQL Developer, MongoDB Compass

**5. System Analysis and Design :**

The system identifies the main entities and their relationships. Major entities include:

**ENTITIES:**

1. Customers

Attributes: customer\_id, name, email, phone

2. Coupons

Attributes: coupon\_id, code, description, discount\_amount, expiry\_date, status

3. Payments

Attributes: payment\_id, customer\_id, amount, payment\_date, payment\_method

4. Coupon\_Usage

Attributes: usage\_id, coupon\_id, customer\_id, payment\_id, used\_on

**RELATIONSHIPS:**

Customers → Payments :

One customer can make many payments.

(1-to-many relationship)

Customers → Coupon\_Usage :

One customer can use many coupons.

(1-to-many relationship)

Coupons → Coupon\_Usage :

One coupon can be used many times (if reusable), or just once depending on business rules.

(1-to-many relationship)

Payments → Coupon\_Usage :

One payment may be associated with a coupon usage.

(1-to-1 or 1-to-many depending on design)

**6. ER Diagram (Conceptual Design) :**

* **Entities and Attributes :**

1. Customers

Attributes: customer\_id, name, email, phone

2. Coupons

Attributes: coupon\_id, code, description, discount\_amount, expiry\_date, status

3. Payments

Attributes: payment\_id, customer\_id, amount, payment\_date, payment\_method

4. Coupon\_Usage

Attributes: usage\_id, coupon\_id, customer\_id, payment\_id, used\_on

* **Relationships:**

Customers → Payments :

One customer can make many payments.

(1-to-many relationship)

Customers → Coupon\_Usage :

One customer can use many coupons.

(1-to-many relationship)

Coupons → Coupon\_Usage :

One coupon can be used many times (if reusable), or just once depending on business rules.

(1-to-many relationship)

Payments → Coupon\_Usage :

One payment may be associated with a coupon usage.

(1-to-1 or 1-to-many depending on design)

**Block Diagram :**

A diagram of a payment method

AI-generated content may be incorrect.

ER Diagram (Text Format) :

+-------------+ +------------+ +------------+

| Customers |1-------∞ | Payments | | Coupons |

|-------------| |------------| |------------|

| customer\_id |<-------∞ | customer\_id| | coupon\_id |

| name | | amount | | code |

| email | | date | | discount |

| phone | | method | | expiry |

+-------------+ +------------+ +------------+

| | |

| | |

∞ ∞ ∞

\|/

|

|

+----------------+

| Coupon\_Usage |

|----------------|

| usage\_id |

| coupon\_id |

| customer\_id |

| payment\_id

| used\_on |

+----------------+

**7. Schema Design (Oracle) :**

SQL>CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100) UNIQUE,

phone VARCHAR(15)

);

Output:

A computer screen with white text

AI-generated content may be incorrect.  
  
SQL> CREATE TABLE Coupons (

coupon\_id INT PRIMARY KEY,

code VARCHAR(50) UNIQUE,

description VARCHAR(255),

discount\_amount DECIMAL(10,2),

expiry\_date DATE,

status VARCHAR(20) CHECK (status IN ('Active', 'Expired', 'Used'))

);

Output:

A screen shot of a computer

AI-generated content may be incorrect.

SQL> CREATE TABLE Payments (

payment\_id INT PRIMARY KEY,

customer\_id INT,

amount DECIMAL(10,2),

payment\_date DATE,

payment\_method VARCHAR(50),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

Output:

A screen shot of a computer program

AI-generated content may be incorrect.

SQL> CREATE TABLE Coupon\_Usage (

usage\_id INT PRIMARY KEY,

coupon\_id INT,

customer\_id INT,

payment\_id INT,

used\_on DATE,

FOREIGN KEY (coupon\_id) REFERENCES Coupons(coupon\_id),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id),

FOREIGN KEY (payment\_id) REFERENCES Payments(payment\_id)

);

Output:

A screen shot of a computer program

AI-generated content may be incorrect.

**8. Normalization :**

The schema design follows normalization principles upto 3NF:

1NF (Remove repeating groups): Each attribute holds atomic values only (no lists or sets).

2NF (Remove partial dependencies): Every non-key attribute depends on the whole primary key.

3NF (Remove transitive dependencies): No non-key attribute depends on another non-key attribute.

**9. Implementation (SQL Queries) :**  
  
INSERT INTO Customers (customer\_id, name, email, phone) VALUES(1, 'Ravi Kumar', 'ravi@gmail.com', '9876543210');

INSERT INTO Customers (customer\_id, name, email, phone) VALUES(2, 'Priya Sharma', 'priya@gmail.com', '9876501234');

INSERT INTO Customers (customer\_id, name, email, phone) VALUES(3, 'Arun Verma', 'arun@gmail.com', '9876512345');  
A black screen with white text

AI-generated content may be incorrect.

INSERT INTO Payments (payment\_id, customer\_id, amount, payment\_date, payment\_method) VALUES(201, 1, 1500.00,TO\_DATE('2025-10-15', 'YYYY-MM-DD'), 'UPI'); INSERT INTO Payments (payment\_id, customer\_id, amount, payment\_date, payment\_method) VALUES(202, 2, 800.00,TO\_DATE('2025-10-18', 'YYYY-MM-DD'), 'Credit Card'); INSERT INTO Payments (payment\_id, customer\_id, amount, payment\_date, payment\_method) VALUES(203, 1, 950.00,TO\_DATE('2025-10-20', 'YYYY-MM-DD'), 'Cash');

A black background with many lines

AI-generated content may be incorrect.

**10. Input and Output :**

\*\*Sample Input Queries:\*\*

\*\*Sample Output Query and Result:\*\*

**Query:**  
SELECT s.Student\_Name, c.Course\_Name FROM Student s JOIN Enrollment e ON s.Student\_ID = e.Student\_ID JOIN Course c ON e.Course\_ID = c.Course\_ID;  
  
**Output:**

**A computer screen with text on it

AI-generated content may be incorrect.**

**11. Integration with MongoDB (NoSQL) :**

MongoDB is a NoSQL, document-oriented database used to store large volumes of unstructured or semi-structured data. In contrast to relational databases like Oracle, MongoDB stores data in JSON-like documents instead of rows and columns. This makes it highly flexible, scalable, and efficient for applications that deal with dynamic content, such as online learning platforms.

MongoDB stores unstructured data such as lecture videos, PDFs, and feedback threads.

MongoDB Database Structure for the Project

Database Name: OnlineLearningDB

Collections and Examples

**1.customers collection**{

"\_id": ObjectId("6743f001a7c4b4f0a1e1d101"),

"customer\_id": 101,

"name": "Nandha Kishore",

"email": "nandha@example.com",

"phone": "9876543210"

}

**2.couponscollections**{

"\_id": ObjectId("6743f010a7c4b4f0a1e1d102"),

"coupon\_id": 201,

"code": "FEST500",

"description": "Festival Offer ₹500 off",

"discount\_amount": 500.00,

"expiry\_date": "2025-12-31",

"status": "Active"

}

**3.paymentscollections**  
{

"\_id": ObjectId("671b0d5fe3b2f9c69b7d9129"),

"payment\_id": 1,

"customer\_id": ObjectId("671b0c2fe3b2f9c69b7d9123"),

"amount": 2500.75,

"payment\_date": ISODate("2025-10-25T00:00:00Z"),

"payment\_method": "Credit Card"

}

**4. coupoun usage Collection**

{

"\_id": ObjectId("671b0f4fe3b2f9c69b7d9139"),

"usage\_id": 1,

"coupon\_id": ObjectId("671b0e9ce3b2f9c69b7d9121"),

"customer\_id": ObjectId("671b0c2fe3b2f9c69b7d9123"),

"payment\_id": ObjectId("671b0d5fe3b2f9c69b7d9129"),

"used\_on": ISODate("2025-10-25T00:00:00Z")

}

**1. Inserting Data in customers collection**.

db.Customers.insertOne({

\_id: ObjectId("6743f001a7c4b4f0a1e1d101"),

customer\_id: 101,

name: "Ganesh",

email: "ganesh@example.com",

phone: "9876543110"

});

**2 ubdate ammount in payments**

db.Payments.updateOne(

{ \_id: ObjectId("671b0d5fe3b2f9c69b7d9129") },

{ $set: { amount: 5000.00 } }

);

**12. Results and Discussion streamlines :**

The gift coupon application handles offers payments related information successfully done in sql and mongo db with the customers, coupons, payments, coupon usage. Customers can use coupon and they make payments we can also check the coupon usage by the ids. The payment amount we can see in it. The system manages customer payments and applies coupon discounts automatically. It allows tracking of transactions, offers, and user activity. The application successfully records payments, applies coupon discounts, and maintains customer offer history efficiently in a NoSQL database.

**13. Conclusion :**

The Gift Coupon Application efficiently manages customer, coupon, and payment information using MongoDB’s NoSQL architecture. By storing data in flexible JSON documents, it simplifies handling of real-time offers and payment updates. The system allows easy insertion, updation, and retrieval of payment and coupon data while maintaining strong consistency and performance.

Overall, the application provides a scalable, fast, and reliable solution for managing digital payments and promotional offers, making it well-suited for modern e-commerce and customer reward systems.

**14. References :**

1. Oracle Database Documentation – Oracle Corporation  
2. MongoDB Official Documentation

GeeksforGeeks – MongoDB Basics-

[https://www.geeksforgeeks.org/mongodb-tutorial/](%20https://www.geeksforgeeks.org/mongodb-tutorial/)

→ Clear explanations of NoSQL concepts and MongoDB commands with examples.

3. Oracle, SQL Language Reference

→ Official Oracle SQL documentation for syntax and commands..