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
B.Tech. – Computer Science and Engineering

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SDG 4: Quality Education
Course Code : 10211CS207
Course Name : Database Management Systems

SlotNo :S6L6

DBMS PROJECT REPORT

Title: Public service records management

Submitted by:

VTUNO	REGISTER NUMBER	STUDENT NAME
VTU 29462	24UECS1413	V.Rupa Sri
VTU 27715	24UECS0661	K.Padmaja
VTU 27898	24UECS0407	A.Bhanu gomathi
VTU 29056	24UECS0113	G.Hitesh
VTU 28092	24UECS0335	T.Bhanu Prakash

Under the guidance of:
Dr. Uma Nandhini.D
Associate Professor

INDEX **PAGE**

1. Introduction

The Public Service Records Management System is a database management project designed to efficiently store, manage, and retrieve information related to public services provided to citizens. In traditional systems, maintaining citizen records, service details, and transaction history is often time-consuming and prone to errors. This project aims to overcome these challenges by implementing a computerized database that .

The system keeps track of citizens, the various public services offered, and the records of services availed by each individual. By using SQL-based queries and relational database concepts, the project supports data organization, retrieval, and manipulation in a structured manner. It enables administrators to perform operations such as adding new services, updating citizen details, monitoring service usage, and generating analytical reports.

Overall, this project demonstrates how database management techniques can be applied to improve the efficiency and transparency of public service administration, ensuring better service delivery and data integrity.

2. Problem Statement

In many government departments, maintaining public service records manually leads to issues such as data redundancy, inconsistency, and difficulty in retrieving information when needed. Managing details of citizens, the services they avail, and the corresponding records becomes inefficient and time-consuming with traditional paper-based systems.

There is a lack of a centralized and systematic approach to store and manage these records securely. As a result, tracking citizen information, updating service details, and generating reports often require significant effort and may lead to errors or loss of data.

To address these challenges, there is a need for a database-driven system that can efficiently handle all public service-related information, ensure data accuracy, and enable quick access and updates. The Public Service Records Management System aims to provide an organized, reliable, and user-friendly solution to manage and maintain these records digitally.

3. Objectives

The key objectives of the Public service records management System are:

1.To design a centralized database for storing and managing information related to citizens and the public services they avail.

2. To eliminate data redundancy and inconsistency by maintaining accurate and normalized records within the database.

3.To simplify data retrieval and updates through the use of SQL queries for efficient information management.

4. To enhance transparency and accountability in public service delivery by providing reliable and easily accessible records.

5. To enable quick report generation for analyzing service usage, citizen participation, and administrative efficiency.

6. To demonstrate the practical application of DBMS concepts such as relationships, constraints, and queries in managing real-world data.

4. System Requirements

Hardware Requirements:

- Processor: Intel i5 or higher
- RAM: 8 GB or more - Hard

Disk: 250 GB or more

Software Requirements:

- OS: Windows 10 or Linux or mac OS
- Database: Oracle 12c /My SQL or above
- Front-End: HTML,CSS, Javascript/
- NoSQL: MongoDB 6.0
- Tools: Oracle SQL Developer /mySQL workbench

5. System Analysis and Design

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

- Citizen
- Public_Service
- Public_Service_Record
- Application

6. ER Diagram (Conceptual Design)

Relationships:

1.Citizen — Submits — Application

Type: One-to-Many (1 : M)

Explanation:Explanation:Each Citizen can submit multiple Applications,but each Application is submitted by exactly one Citizen

2.Application — Has — Service

Type: Many-to-Many (M : M)

Explanation:

An Application can include multiple Services

3. Application — Recorded In — Public_Service_Record

Type: One-to-One (1 : 1)

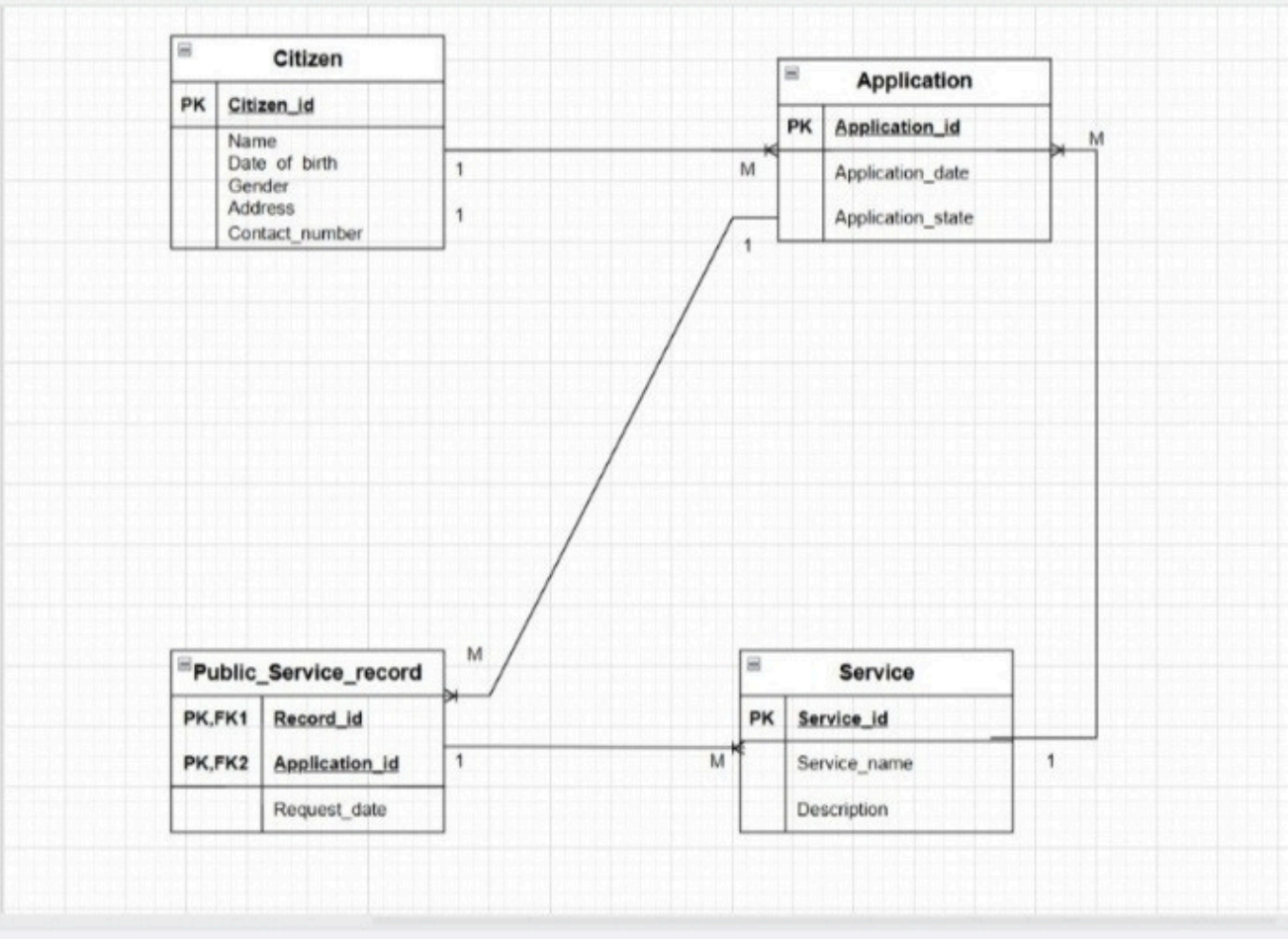
Each Application has one corresponding Public Service Record entry for tracking&history, and each Record belongs to a single Application.

4. Public_Service_Record — Belongs To — Citizen

Type: Many-to-One (M : 1)

Multiple Public Service Records may belong to one Citizen (since a citizen can make many service requests).

Figure: ER Diagram



Entities and Attributes:

1.citizen:citizen_id,date_of_birth,name,gender,address,contact

2.Application: application_id, application_date, application_status, citizen_id

3.Service: service_id, service_name

4.Public_service_record:record_id, application_id, request date

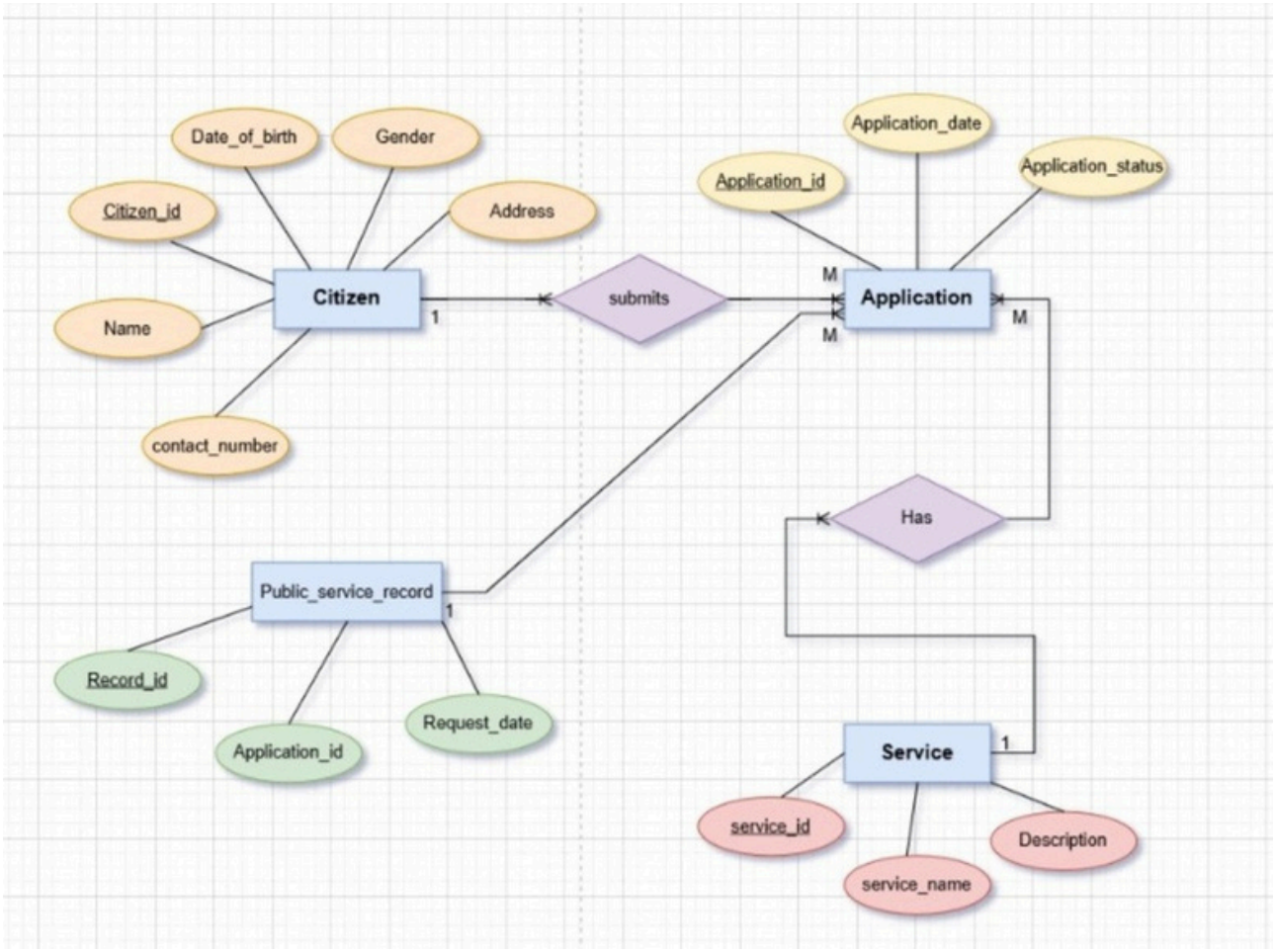
Relationships:

1. Citizen — Submits — Application

2. Application — Has — Service

3. Application — Recorded In — Public_Service_Record

4. Public_Service_Record — Belongs To — Citizen


Date_of_birth
Gender
Citizen_id
Address
Citizen
Name
contact_number
1
submits
M
M
Application
Application_date
Application_id
Application_status
M
Has
Public_service_record
1
Record_id
Application_id
Request_date
Service
1
service_id
service_name
Description

7. Schema Design (Oracle)

```
CREATE TABLECitizen (
Citizen_id INT PRIMARY KEY,
Name VARCHAR(100) NOT NULL,
Date_of_birth DATE,
Gender VARCHAR(10),
Address VARCHAR(255),
Contact_number VARCHAR(15)
);
```

Output:

```
+-----------------+---------------+------+-
| Field           | Type          | Null |
+-----------------+---------------+------+-
| Citizen_id      | int           | NO   |
| Name            | varchar(50)   | YES  |
| Date_of_birth   | date          | YES  |
```

SQL>CREATE TABLE Service (Service_id INT PRIMARY KEY, Service_name VARCHAR(100) NOT NULL,Description VARCHAR(255));

OUTPUT:

Field	Type
Service_id	int
Service_name	varchar(100)
Description	varchar(255)

SQL>CREATE TABLE Application (
Application_id INT PRIMARY KEY,
Application_date DATE,
Application_status VARCHAR(50),
Citizen_id INT,
(Citizen_id));

OUTPUT:

Field	Type	Null	Key	Default	Extra
Application_id	int	NO	PRI	NULL	
Application_date	date	YES		NULL	
Application_status	varchar(50)	YES		NULL	
Citizen_id	int	YES	MUL	NULL	

SQL>CREATE TABLE Public_Service_Record (
Record_id INT PRIMARY KEY,
Application_id INT,
Request_date DATE,
FOREIGN KEY (Application_id) REFERENCES
Application(Application_id)
);

OUTPUT:

Field	Type	Null	Key	Default	Extra
Record_id	int	NO	PRI	NULL	
Application_id	int	YES	MUL	NULL	
Request_date	date	YES		NULL	

8. Normalization

1NF:

Each table must have a primary key.

All attributes must contain only atomic (indivisible) values.

No repeating groups or arrays are allowed.

Step 2: Achieve Second Normal Form (2NF)

The table must be in 1NF.

All non-key attributes must depend on the

entire primary key (no partial dependency).

In Our Project:Each table with a composite key (like Application_Service) has all attributes fully dependent on the combination of Application_id and Service_id.

3. Third Normal Form (3NF):

Rule:The table must be in 2NF.

There should be no transitive dependency (non-key attributes should not depend on other non-key attributes).

Conclusion:

The Public Service Records Management System database is normalized up to Third Normal Form (3NF).

This ensures:Elimination of redundant data,Efficient data storage,Easy maintenance and updates,Strong referential integrity across all entities

9. Implementation (SQL Queries)

INSERT INTO Citizen (Citizen_id, Name, Date_of_birth, Gender, Address, Contact_number)
VALUES
(101, 'Aarav Sharma', '1998-04-12', 'Male', 'Delhi, India', '9876543210'),
(102, 'Priya Verma', '2000-07-25', 'Female', 'Mumbai, India', '9823456789'),
(103, 'Rahul Mehta', '1995-11-08', 'Male', 'Pune, India', '9812345678'),
(104, 'Ananya Rao', '1999-03-14', 'Female', 'Hyderabad, India', '9908765432');

Citizen_id	Name	Date_of_birth	Gender	Address	Contact_number
101	Aarav Sharma	1998-04-12	Male	Delhi, India	9876543210
102	Priya Verma	2000-07-25	Female	Mumbai, India	9823456789
103	Rahul Mehta	1995-11-08	Male	Pune, India	9812345678
104	Ananya Rao	1999-03-14	Female	Hyderabad, India	9908765432

INSERT INTO Service (Service_id, Service_name, Description)
VALUES
(201, 'Birth Certificate', 'Issuance of official birth certificate'),
(202, 'Passport Renewal', 'Renewal of expired or expiring passports'),
(203, 'Voter ID Registration', 'Application for new voter ID card'),
(204, 'Pension Scheme', 'Enrollment in government pension scheme');

Service_id	Service_name	Description
101	Passport Renewal	Renewal of expired passports
102	Driving License	Issuance of new driving license
103	Aadhar Update	Correction of personal details in Aadhar
201	Birth Certificate	Issuance of official birth certificate
202	Passport Renewal	Renewal of expired or expiring passports
203	Voter ID Registration	Application for new voter ID card
204	Pension Scheme	Enrollment in government pension scheme

INSERT INTO Application (Application_id, Application_date,

Application_status, Citizen_id)

VALUES

(301, '2025-01-10', 'Approved', 101),

(302, '2025-02-05', 'Pending', 102),

(303, '2025-02-15', 'Rejected', 103),

(304, '2025-03-01', 'Approved', 104);

Application_id	Application_date	Application_status	Citizen_id
301	2025-01-10	Approved	101
302	2025-02-05	Pending	102
303	2025-02-15	Rejected	103
304	2025-03-01	Approved	104

INSERT INTO Public_Service_Record (Record_id, Application_id, Request_date)
VALUES(401, 301, '2025-01-12'),(402, 302, '2025-02-06'),(403, 303, '2025-02-16'),
(404, 304, '2025-03-02');

Record_id	Application_id	Request_date
401	301	2025-01-12
402	302	2025-02-06
403	303	2025-02-16
404	304	2025-03-02

10. Input and Output

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

```
INSERT INTO Citizen (Citizen_id, Name, Date_of_birth, Gender, Address, Contact_number)
VALUES
(101, 'Aarav Sharma', '1998-04-12', 'Male', 'Delhi, India', '9876543210'),
(102, 'Priya Verma', '2000-07-25', 'Female', 'Mumbai, India', '9823456789'),
(103, 'Rahul Mehta', '1995-11-08', 'Male', 'Pune, India', '9812345678'),
(104, 'Ananya Rao', '1999-03-14', 'Female', 'Hyderabad, India', '9908765432');
```

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

** Query:

```
SELECT

SELECT Application_id, Application_date, Citizen_id

FROM Application

WHERE Application_status = 'Approved';
```

OUTPUT:

```
+----------------+------------------+------------+
| Application_id | Application_date | Citizen_id |
+----------------+------------------+------------+
|            301 | 2025-01-10       |        101 |
|            304 | 2025-03-01       |        104 |
+----------------+------------------+------------+
```

11. Integration with MongoDB (NoSQL)

MongoDB commands:

```
use Public_Service_Records
db.citizens.insertMany([
                {
            citizen_id: 1,
        name: "Rupa",
 date_of_birth: "2005-03-15",
      gender: "Female",
      address: "Hyderabad",
contact_number: "9876543210"
                },
                {
            citizen_id: 2,
        name: "Aarav",
 date_of_birth: "2002-07-10",
      gender: "Male",
        address: "Delhi",
contact_number: "9123456780"
                },
                {
            citizen_id: 3,
        name: "Meera",
 date_of_birth: "1999-12-25",
      gender: "Female",
        address: "Chennai",
contact_number: "9988776655"

                }
                ]
                )
                ;
```

```
db.services.insertMany([
{ service_id: 101, service_name: "Passport Renewal", description: "Renewal of expired passports" },
{ service_id: 102, service_name: "Driving License", description: "Issuance of new driving license" },
{ service_id: 103, service_name: "Aadhar Update", description: "Correction of personal details in Aadhar" }
]);
db.citizens.find();
db.citizens.find({ address: "Hyderabad" });
db.citizens.updateOne(
{ citizen_id: 1 },
{ $set: { contact_number: "9000011122" } }
);
db.citizens.deleteOne({ citizen_id: 3 });
```

12. Results and Discussion

The Public Service Records Management System was successfully designed and implemented using SQL. The database consists of interrelated tables such as Citizen, Service, Application, Public_Service_Record, and Application_Service. These tables collectively manage citizen information, service details, and records of public service applications efficiently.

Through normalization up to Third Normal Form (3NF), the database eliminates redundancy and ensures data integrity. The ER diagram and schema clearly define the relationships between entities like citizens, services, and applications, providing a well-structured relational model.

When executing the SQL queries, the expected results were obtained:

13. Conclusion:

ThePublic Service Records Management System project successfully demonstrates the use of database management concepts to handle real-world data efficiently. The system provides a structured way to store, manage, and retrieve information about citizens and the public services they avail. By organizing the data into multiple related tables, it ensures accuracy, reduces redundancy, and enhances data consistency.

Through the implementation of SQL commands, relationships, and normalization, the project achieves reliable data storage and faster access to records. It simplifies administrative work by enabling quick updates, report generation, and tracking of services. Overall, this project highlights the importance of a well-designed DBMS in improving the efficiency, transparency, and reliability of public service delivery.

14. References

1. Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson Education, 2017.

2. Silberschatz, Korth, and Sudarshan, Database System Concepts, 7th Edition, McGraw Hill Education, 2020.

3. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw Hill, 2014.

4. Oracle Documentation — SQL Language Reference, Oracle Corporation.

5. MySQL Developer Guide — Structured Query Language (SQL), MySQL Reference Manual.

6. Online resources from GeeksforGeeks, TutorialsPoint, and W3Schools for SQL synta
x and examples.