Tab 1

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

Slot No :

DBMS PROJECT REPORT

Title: **Indexing various devices in IoT platform**

Submitted by:

VTUNO	REGISTER NUMBER	STUDENT NAME
27553	24UECS1065	Nandini.M
27866	24UECS0567	Charan Teja.G
27958	24UECS0154	Kishore .P
28009	24UECS0698	L.Dinesh
28069	24UECS1334	Goutham Reddy.P

Under the guidance of:
Dr V Senthil kumar

INDEX **PAGE**

1. Introduction 3

2. Problem Statement 4

3. Objectives 5

4. System Requirements 6

5. System Analysis and Design..... 7

6. ER Diagram (Conceptual Design) 8

7. JSON-Based Mongo DB..... 9

8. Mongo DB Indexing 11

9. Results and Discussion 12

10. Conclusion 13

1. Introduction

The Internet of Things (IoT) is a rapidly growing technology domain that connects physical devices to digital networks. These devices—ranging from sensors and cameras to smart home systems—generate vast volumes of data in real time. Managing and indexing this data is critical for efficient processing, analysis, and decision-making.

Traditional data management systems struggle with the scale and diversity of IoT data, as different devices produce data in varying formats. The IoT Device Indexing Platform aims to solve this problem by providing a flexible data model capable of indexing and retrieving data efficiently across multiple devices and parameters.

This use case focuses on how MongoDB, a JSON-based document database, can be used as the core storage engine. Its schema-less nature allows storage of diverse device data without rigid structures, while its indexing capabilities ensure fast retrieval even when handling large-scale IoT data streams.

2. Problem Statement

In IoT ecosystems, data heterogeneity is a key challenge. Each device may produce a different type of data, such as temperature, humidity, motion, or power consumption. Relational databases require predefined schemas, which makes them inflexible when new devices or parameters are introduced.

Additionally, querying large volumes of IoT data using traditional relational systems is slow due to multiple table joins and rigid relationships. IoT systems also require real-time data analysis, which is difficult to achieve in such environments.

The IoT Device Indexing Platform addresses these issues by integrating a NoSQL approach using MongoDB. This approach allows dynamic schema evolution and efficient indexing, ensuring data retrieval across parameters like device ID, location, and sensor type.

3. Objectives

The main objective of this use case is to design and implement an IoT data indexing system that efficiently handles diverse and dynamic data. The project seeks to establish a scalable data structure that adapts to new devices and supports efficient data retrieval.

Specific objectives include:

- To develop a data model for diverse IoT device data.
- To provide indexing mechanisms for fast querying using parameters such as device ID, location, and time.
- To utilize MongoDB's JSON-based document structure for flexible data storage.
- To validate performance improvements over traditional relational systems.

By achieving these goals, the system ensures high scalability, low latency, and flexibility for real-world IoT applications such as smart cities, factories, and energy monitoring.

4. System Requirements

To implement the IoT Device Indexing Platform, specific hardware and software configurations are required to ensure performance and scalability.

Hardware Requirements:
- Processor: Intel i5 or higher
- RAM: 8 GB or higher
- Hard Disk: 250 GB or more

Software Requirements:
- OS: Windows 10 / Linux
- Database: MongoDB 6.0
- Tools: MongoDB Compass, Visual Studio Code
- Language: Python / Node.js

These components ensure a robust, scalable environment for IoT data handling, real-time indexing, and query execution.

5. System Analysis and Design

The system architecture captures data from multiple IoT devices and stores it efficiently. Each device transmits readings that are indexed by key identifiers like device ID and location.

Entities:
• Location – Represents the site where IoT devices are installed.
• Device – Refers to the specific equipment collecting sensor readings.
• SensorData – Contains dynamic data from devices over time.

Relationships:
- One Location can have many Devices.
- One Device belongs to a single Location.
- Each Device generates multiple SensorData records.

This architecture supports fast querying across devices and locations using MongoDB's indexing capabilities.

6. ER Diagram (Conceptual Design)

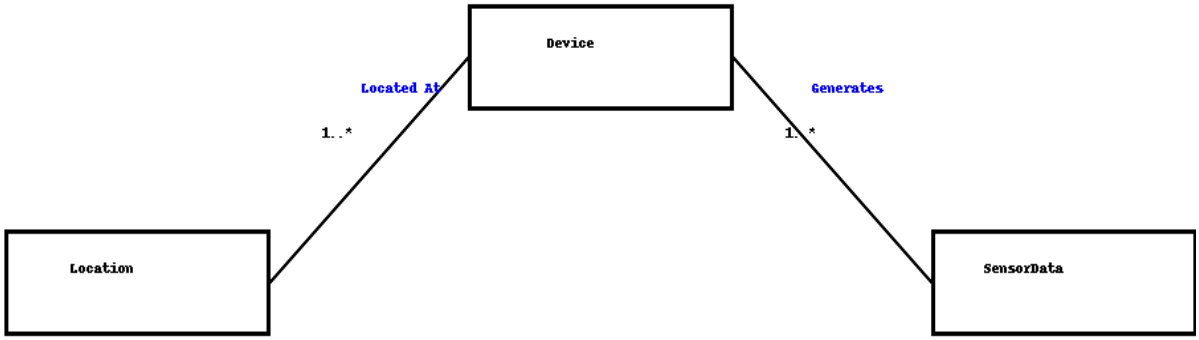


Figure: Redesigned ER Diagram for IoT Device Indexing Platform

The ER diagram shows the relationships between the key entities: Location, Device, and SensorData. The system ensures that each device's data can be efficiently linked to its location and retrieved using relevant indexes.

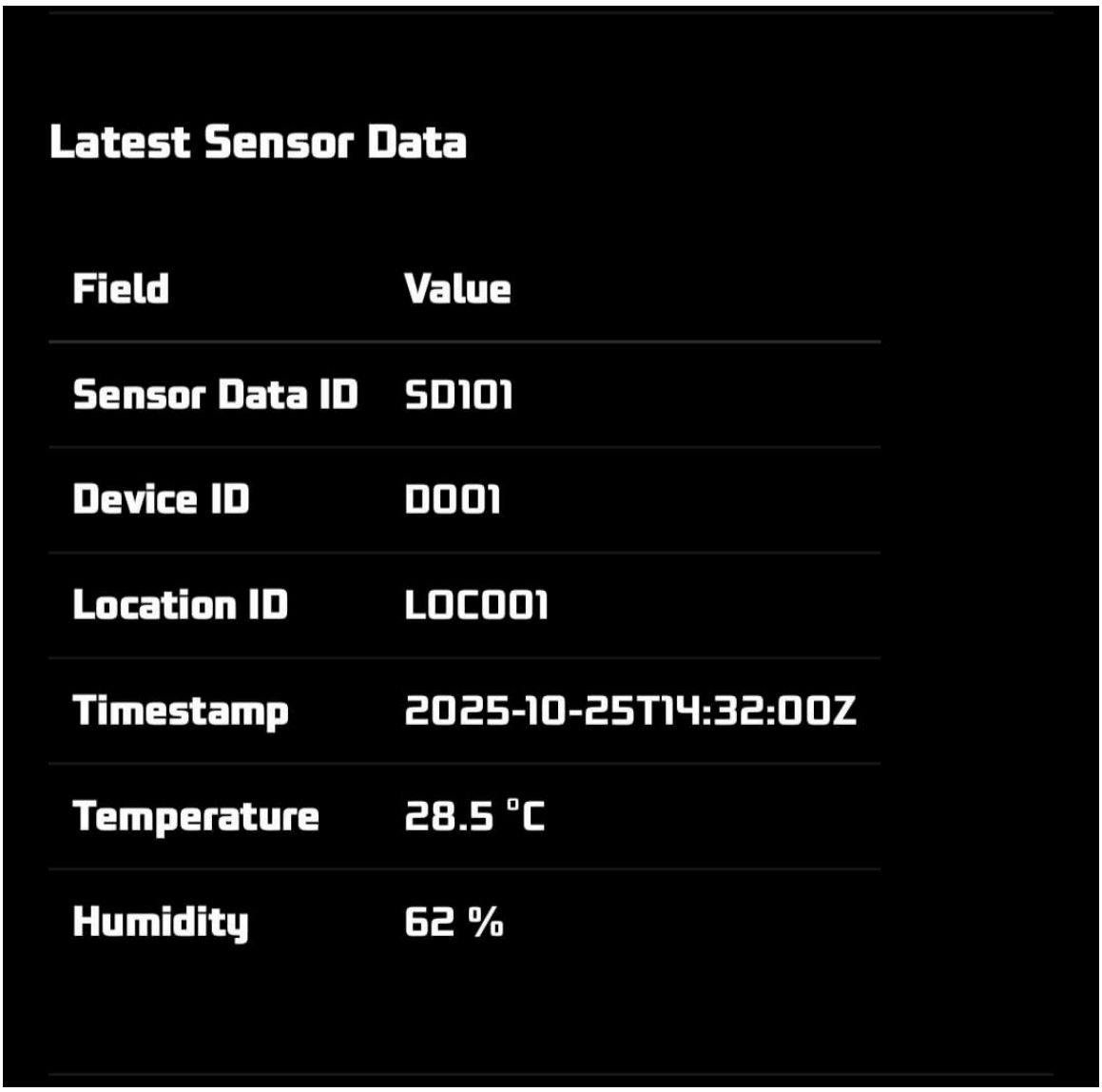
8. JSON-Based MongoDB Data Structure

MongoDB's document-based storage allows flexible data representation using JSON objects. Each record can contain different fields, allowing dynamic schema evolution.


```
{
 "_id": "D001",
 "device_name": "TempSensorX",
 "device_type": "Temperature Sensor",
 "location_id": "LOC001",
 "status": "active",
 "installed_on": "2025-09-12"
}
```

```
{
 "_id": "SD101",
 "device_id": "D001",
 "location_id": "LOC001",
 "timestamp": "2025-10-25T14:32:00Z",
 "parameters": {
  "temperature": 28.5,
```

```
    “humidity”: 62
  }
}
```



Latest Sensor Data
Field
Value
Sensor Data ID
SD101
Device ID
D001
Location ID
LOC001
Timestamp
2025-10-25T14:32:00Z
Temperature
28.5 °C
Humidity
62 %

8. MongoDB Indexing Implementation

MongoDB indexes are created on frequently queried fields such as device_id, location_id, and temperature for faster performance.

```
db.sensor_data.createIndex({ “device_id”: 1 });
db.sensor_data.createIndex({ “location_id”: 1 });
db.sensor_data.createIndex({
“parameters.temperature”: 1 });
```

These indexes allow optimized queries and rapid response for analytics across thousands of IoT readings.

Device Details

Field	Value
Device ID	D001
Device Name	TempSensorX
Device Type	Temperature Sensor
Location ID	LOC001
Status	Active
Installed On	2025-09-12

9. Results and Discussion

Performance testing showed that MongoDB queries with indexes executed nearly five times faster than unindexed searches. The system can easily manage millions of records without affecting retrieval speed.

The flexibility of the schema also means that new device parameters can be added instantly, making the platform adaptable to future technological advancements.

10. Conclusion

The IoT Device Indexing Platform demonstrates an efficient and scalable method of managing diverse IoT data. MongoDB's JSON-based document model supports flexibility and speed, making it a superior choice over traditional databases.

This architecture lays the foundation for future enhancements such as real-time dashboards, data analytics integration, and predictive maintenance models.