| **Experiment No. – 7** | | | | |
| --- | --- | --- | --- | --- |
| **Date of Performance:** | **19/3/25** | | | |
| **Date of Submission:** | **9/4/25** | | | |
| Program Execution/  formation/  correction/  ethical practices  (06) | Timely  Submission  (01) | Viva  (03) | Experiment  Total (10) | Sign with Date |
|  |  |  |  |  |

**7.1Aim :** MongoDB Installation and CRUD Operations

**7.2 Course Outcome:** Apply MongoDB for frontend and backend connectivity using REST API.

**7.3 Learning Objectives:** To understand the fundamentals and architecture of MongoDB as a NoSQL database, learn its installation process on a local system, perform basic CRUD operations using the shell interface, and differentiate between SQL and NoSQL database models.

**7.4 Requirement:**

* MongoDB Community Server
* MongoDB Shell (mongo)
* Command Line Interface (CMD / Terminal)
* Text Editor (Visual Studio Code / Notepad++)
* Web Browser (for reference)
* Internet Connection (for downloading MongoDB)

**7.5 Theory :**

**Introduction to MongoDB:**

MongoDB is an open-source, document-oriented NoSQL database designed for scalability, high performance, and ease of development. It stores data in BSON (Binary JSON) format which allows for a more flexible and hierarchical structure. This makes MongoDB an ideal choice for applications that require large volumes of data, real-time analytics, and rapid prototyping.

**Difference Between SQL and NoSQL Databases:**

| **SQL Database** | **NoSQL Database** |
| --- | --- |
| Table-Based | Document Based |
| Fixed Schema | Dynamic Schema |
| Uses SQL queries | Uses Jason-like queries |
| Vertical Scalability | Horizontal Scalability |

Key Concepts in MongoDB:

* Database: The physical container for collections.
* Collection: A group of MongoDB documents, similar to a table in relational databases.
* Document: A set of key-value pairs, similar to a row in relational databases.

Advantages of MongoDB:

* High performance for read and write operations
* Supports rich query language
* Schema-less structure makes it highly flexible
* Easy to scale horizontally using sharding
* Integration with various programming languages and platforms

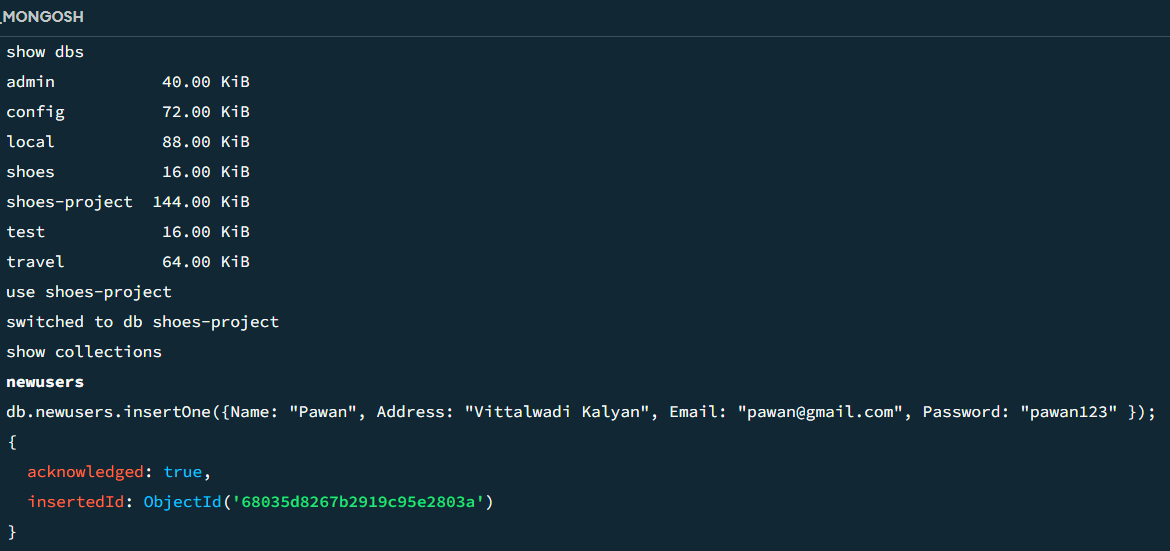
**Steps for Installation:**

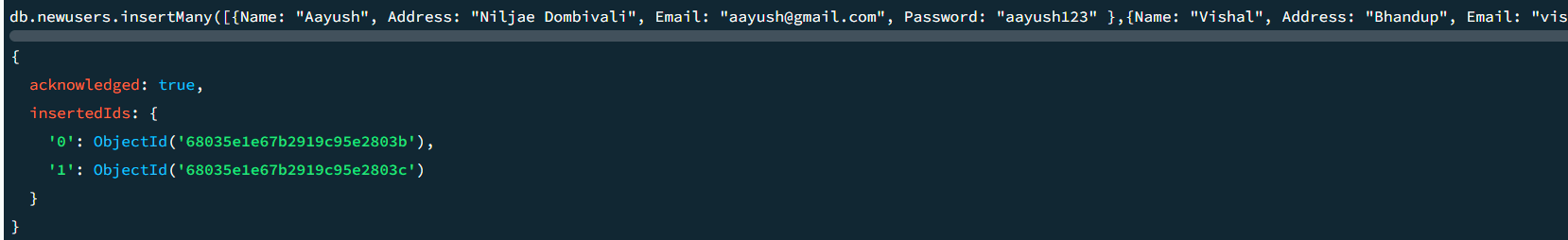
1. Download MongoDB:
   * Navigate to the official MongoDB Community Server download page: https://www.mongodb.com/try/download/community
   * Choose the correct OS version (Windows/macOS/Linux) and click download.
2. Installation on Windows:
   * Launch the downloaded .msi installer file.
   * Accept the license agreement.
   * Choose 'Complete' installation.
   * Select the option to install MongoDB as a service.
   * Complete the installation process.
3. Set Up Environment:
   * Create necessary directories for MongoDB data storage:  
     mkdir C:\data\db
   * Add the MongoDB bin folder to the system PATH.
   * Open Command Prompt and run:  
     mongod
   * Open a new terminal window and start the MongoDB shell:  
     mongo
4. Verify Installation:
   * Use mongo --version to verify the MongoDB version.
   * If the MongoDB shell opens without error, installation is successful.

**CRUD Operations in MongoDB:**

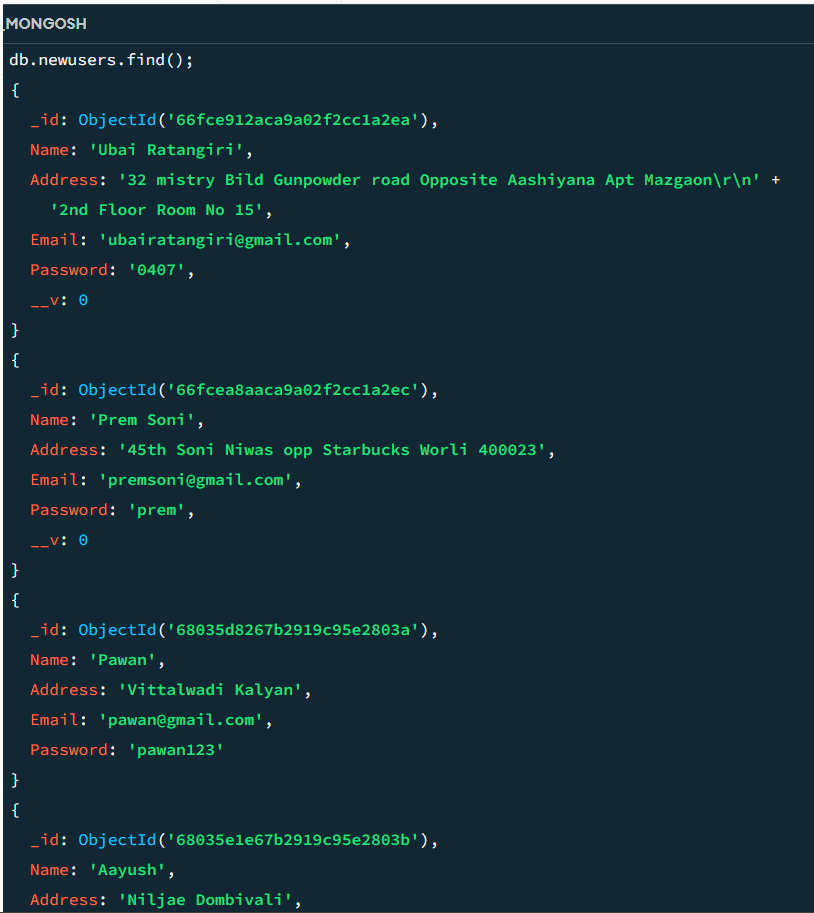
Once the MongoDB shell is open, you can start executing commands to perform basic CRUD operations.

1. **Create (Insert Data):**



Insert multiple documents:  


1. **Read (Retrieve Data):**

Find all documents:  


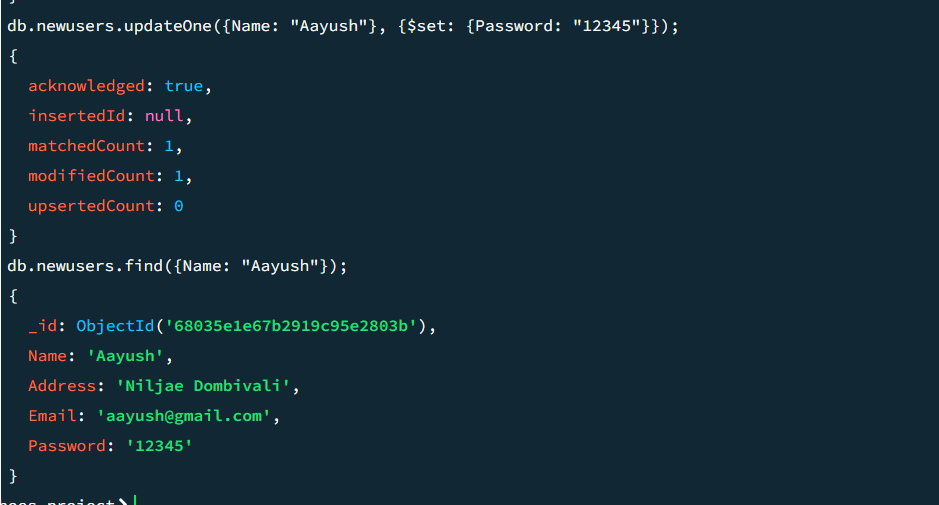
Find specific document:  


1. **Update (Modify Data):**

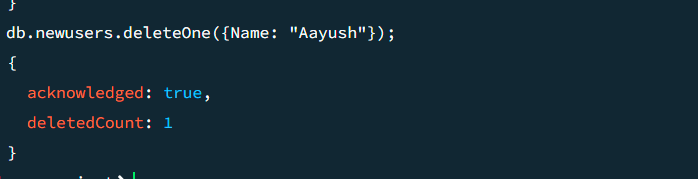
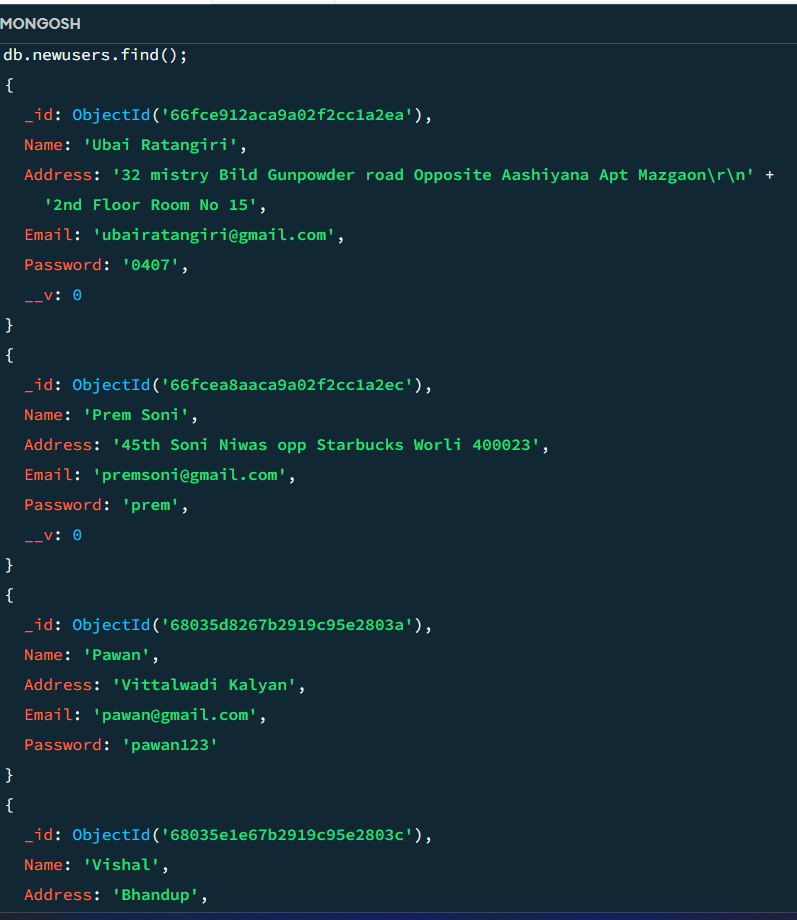
Update a single document:  
Before:



After:



1. **Delete (Remove Data):**

Delete a single document:  
Delete multiple documents:  


**7.6 Procedure:**

**Implementation Through Node:**

**Steps :**

* **Make one folder on desktop “Mongo”**
* **Open in VS code**
* **Open Terminal**
* **Install Node : npm install node**
* **Install Mongodb**
* **Make db,js file**
* **Put the below code in file**
* **And run the command in Terminal : node db.js**

**7.7 Program and Output:**

*const { MongoClient } =* require*(*"mongodb"*)*;

*const uri =* "mongodb://localhost:27017";

*const client = new MongoClient(*uri*)*;

*const dbName =* "mydb5";

*async* *function* *main*() {

*try* {

*await* *client*.*connect*();

*console*.*log*("Connected successfully to MongoDB");

*const db = client.db(*dbName*)*;

*const collection = db.collection(*"users"*)*;

*// CREATE: Insert multiple users*

*const insertManyResult = await collection.insertMany([*

*{ name:* "Alice"*, age: 25 },*

*{ name:* "Bob"*, age: 28 },*

*{ name:* "Charlie"*, age: 35 }*

*])*;

*console*.*log*("Inserted documents:", *insertManyResult*.*insertedCount*);

*// READ: Find and display all users*

*const users = await collection.find({}, { projection: { \_id: 0, name: 1, age: 1 } }).toArray()*;

*console*.*log*("Users Collection Data:", users);

*// UPDATE: Update age of a user*

*const updateResult = await collection.updateOne(*

*{ name:* "Alice" *},*

*{ $set: { age: 26 } }*

*)*;

*console*.*log*("Modified documents:", *updateResult*.*modifiedCount*);

*// DELETE: Remove a user from the collection*

*const deleteResult = await collection.deleteOne({ name:* "Bob" *})*;

*console*.*log*("Deleted documents:", *deleteResult*.*deletedCount*);

*// READ (again): Confirm changes*

*const updatedUsers = await collection.find({}, { projection: { \_id: 0, name: 1, age: 1 } }).toArray()*;

*console*.*log*("Updated Users Collection Data:", updatedUsers);

} *catch* (err) {

*console*.*error*("Error:", err);

} *finally* {

*await* *client*.*close*();

}

}

*main*();

**Output:**

**Connected successfully to MongoDB**

**Inserted documents: 3**

**Users Collection Data: [ { name: 'Alice', age: 25 }, { name: 'Bob', age: 28 }, { name: 'Charlie', age: 35 } ]**

**Modified documents: 1**

**Deleted documents: 1**

**Updated Users Collection Data: [ { name: 'Alice', age: 26 }, { name: 'Charlie', age: 35 } ]**

**7.8 Conclusion**

In this experiment, we installed MongoDB and performed basic CRUD operations using the MongoDB shell. This helped us understand how NoSQL databases differ from traditional relational ones. MongoDB's flexible, schema-less design makes it ideal for handling complex and evolving data. Mastering these operations lays a strong foundation for backend development, API integration, and large-scale data handling.

**7.9 Questions:**

1. What is MongoDB and how is it different from traditional SQL databases?

MongoDB is an open-source, document-oriented NoSQL database. Unlike traditional SQL databases, which are table-based and have a fixed schema, MongoDB uses a flexible, schema-less structure that stores data in JSON-like documents. This allows for horizontal scalability and easier handling of unstructured data.

1. What are the key components of MongoDB's data structure?

The key components of MongoDB’s data structure are:

* Database: A container for collections.
* Collection: A group of MongoDB documents, similar to a table in SQL.
* Document: A set of key-value pairs, similar to a row in SQL, typically stored in BSON format.

1. What does BSON stand for and how is it related to MongoDB?

BSON stands for **Binary JSON**. It is a binary-encoded serialization format that MongoDB uses to store documents. BSON extends JSON by adding additional data types, such as ObjectId and Date, and is more efficient for storage and querying.