SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

COURSE CODE: DJS22ITL604 **DATE: 28-01-2025**

COURSE NAME: Full Stack Web Development Laboratory **CLASS: TYBTech**

NAME: ANISH SHARMA **ROLL: I011** **DIV: IT1-1**

DEPARTMENT OF INFORMATION TECHNOLOGY

EXPERIMENT NO. 01

CO/LO: CO1-Develop a full stack web application.

AIM / OBJECTIVE: Setting Up MERN Stack Environment Install necessary software/tools and verify basic functionality for each component.

THEORY:

The MERN stack is a popular JavaScript-based web development framework used to build full-stack applications. It stands for:

- **M**: MongoDB (Database) A NoSQL database for storing application data in a flexible, JSON-like format.
- **E**: Express.js (Backend Framework) A lightweight web application framework for Node.js, used to build server-side applications.
- **R**: React.js (Frontend Framework) A JavaScript library for building dynamic, responsive user interfaces.
- **N**: Node.js (Runtime Environment) A JavaScript runtime environment for executing server-side code.
- **SQL Databases Vs NoSQL Databases:**

SQL Databases	**NoSQL Databases**
Relational databases with a structured schema (e.g., MySQL, PostgreSQL).	Non-relational databases designed for flexibility and scalability (e.g., MongoDB, CouchDB)
Data is stored in tables with rows and columns	Data is stored in formats like JSON, key-value pairs, or graphs
Suitable for applications requiring ACID (Atomicity, Consistency, Isolation, Durability) properties	Ideal for applications with unstructured or semi-structured data

Basic MongoDB Operations

- **Create**: Insert new documents into a collection.
- **Read**: Retrieve documents from a collection.
- **Update**: Modify existing documents.

- **Delete**: Remove documents from a collection.

Key Components of the MERN Stack

1. MongoDB (Database Layer)

- **Purpose**: Stores application data in a flexible, document-oriented, NoSQL format.
- **Key Features**:
  - Stores data in JSON-like documents.
  - Scalable and supports distributed databases.
  - Allows for easy integration with Node.js applications via libraries like Mongoose.
- **Role in MERN**: Acts as the database to persist application data.

2. Express.js (Backend Framework)

- **Purpose**: A lightweight, flexible web application framework for Node.js.
- **Key Features**:
  - Simplifies the process of building APIs and managing server logic.
  - Supports middleware to handle HTTP requests, responses, and errors.
  - Integrates seamlessly with MongoDB for database operations.
- **Role in MERN**: Handles routing, server-side logic, and API endpoints.

3. React.js (Frontend Framework)

- **Purpose**: A JavaScript library for building user interfaces.
- **Key Features**:
  - Component-based architecture for reusable UI elements.
  - Virtual DOM for efficient updates and rendering.
  - Strong community support and extensive ecosystem.
- **Role in MERN**: Builds the dynamic, responsive user interface (frontend).

4. Node.js (Runtime Environment)

- **Purpose**: Executes JavaScript code on the server side.

- **Key Features**:
  - Built on Chrome's V8 engine for fast execution.
  - Non-blocking, event-driven architecture for handling concurrent requests.
  - Enables the use of JavaScript for both client and server sides.
- **Role in MERN**: Provides the runtime environment for running the server-side application.

How MERN Architecture Works

1. **Frontend (React)**:
   - The user interacts with the React.js frontend, which renders components and handles user actions.
   - React communicates with the backend via HTTP requests to perform CRUD operations.
2. **Backend (Express and Node.js)**:
   - Express.js, running on Node.js, handles incoming requests, processes business logic, and routes the requests to the appropriate endpoints.
   - It also handles communication with the MongoDB database for data storage and retrieval.
3. **Database (MongoDB)**:
   - MongoDB stores application data in collections as JSON-like documents.
   - The backend uses libraries like Mongoose to perform database operations efficiently.
4. **Data Flow**:
   - The React frontend sends API requests to the Express.js server.
   - Express processes these requests and interacts with MongoDB for any required data.
   - The server returns the requested data or status to the frontend, which updates the user interface dynamically.

Prerequisites

DEPARTMENT OF INFORMATION TECHNOLOGY

- Install Node.js: https://nodejs.org/ - Install MongoDB: https://www.mongodb.com/try/download/community
- Install a code editor (e.g., Visual Studio Code): https://code.visualstudio.com/

DEPARTMENT OF INFORMATION TECHNOLOGY

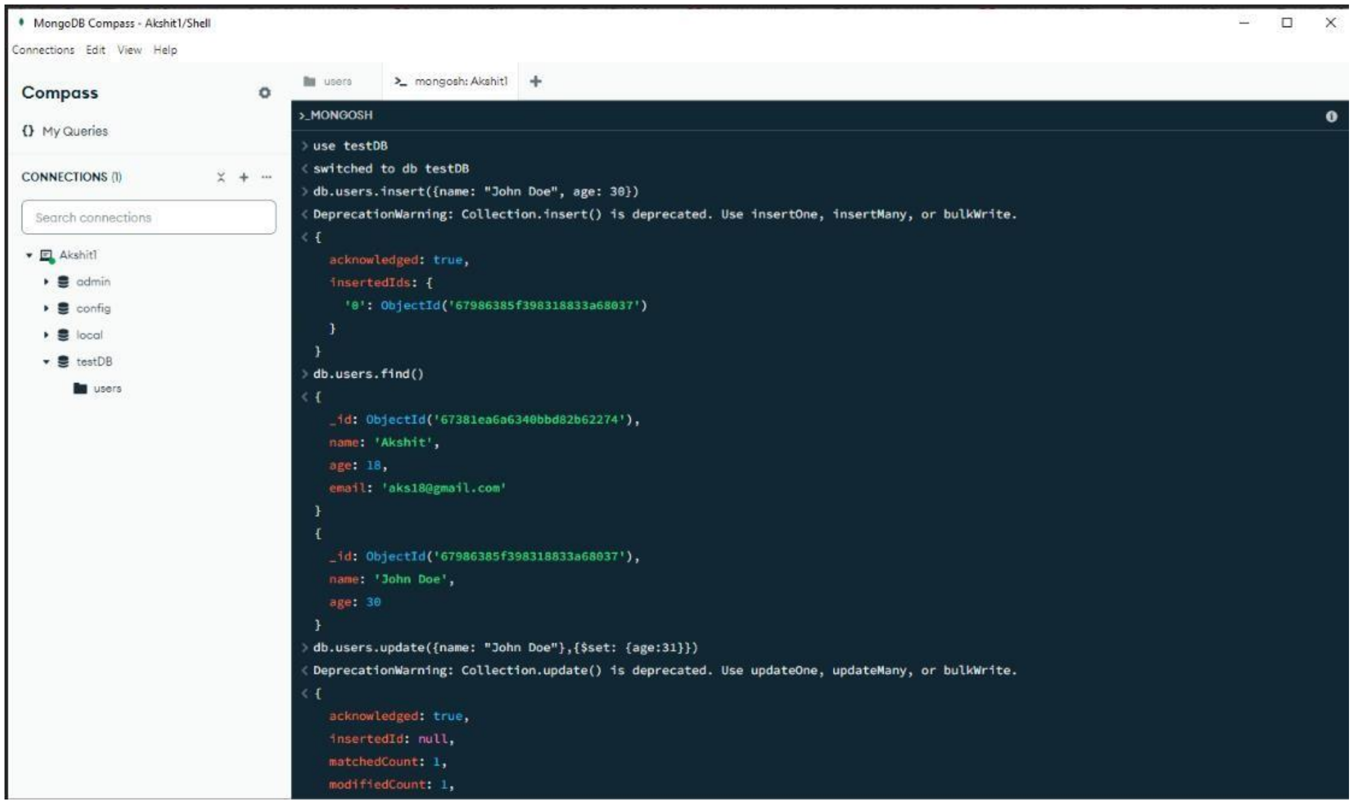
 Install MongoDB Compass

Setting Up MongoDB

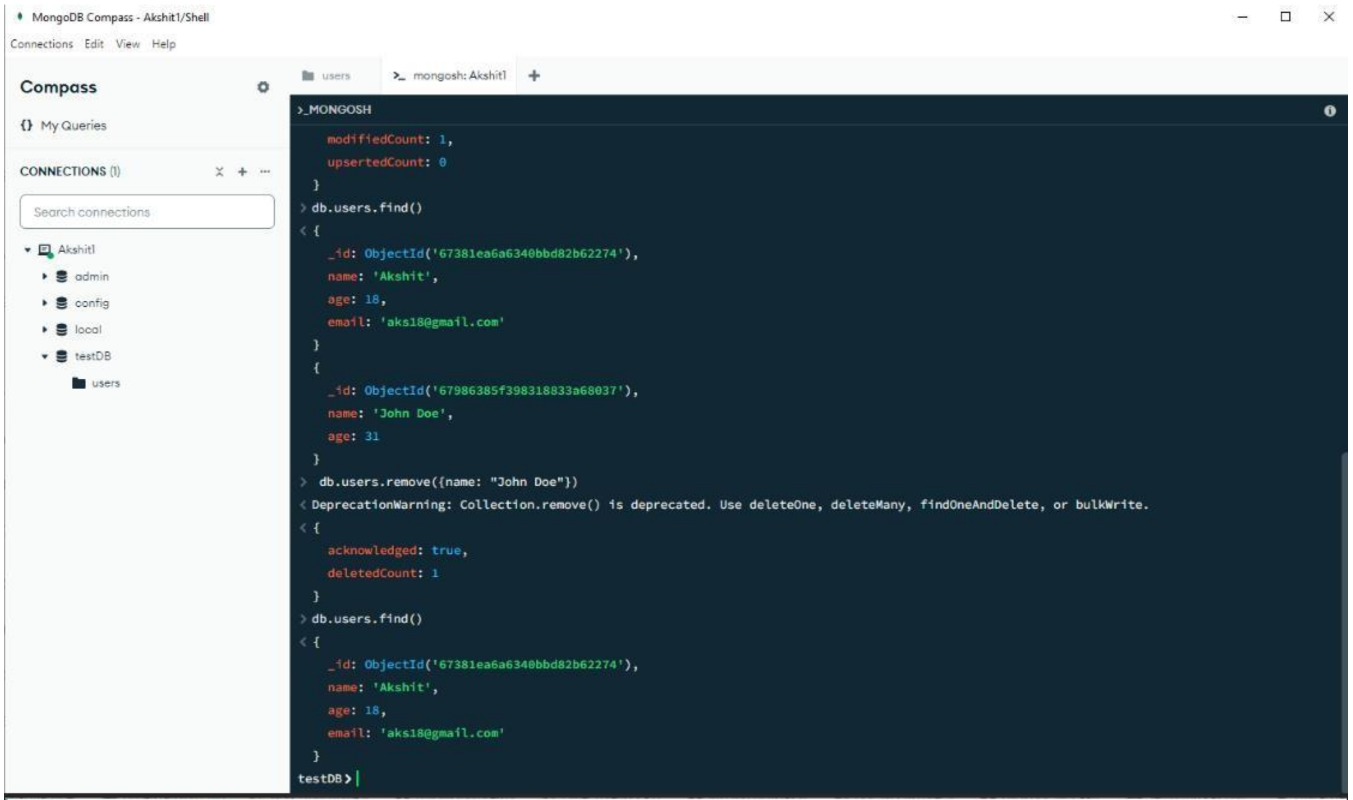
1. Download and install MongoDB.
3. Start the MongoDB server: mongod 5.

Open the MongoDB shell: mongosh

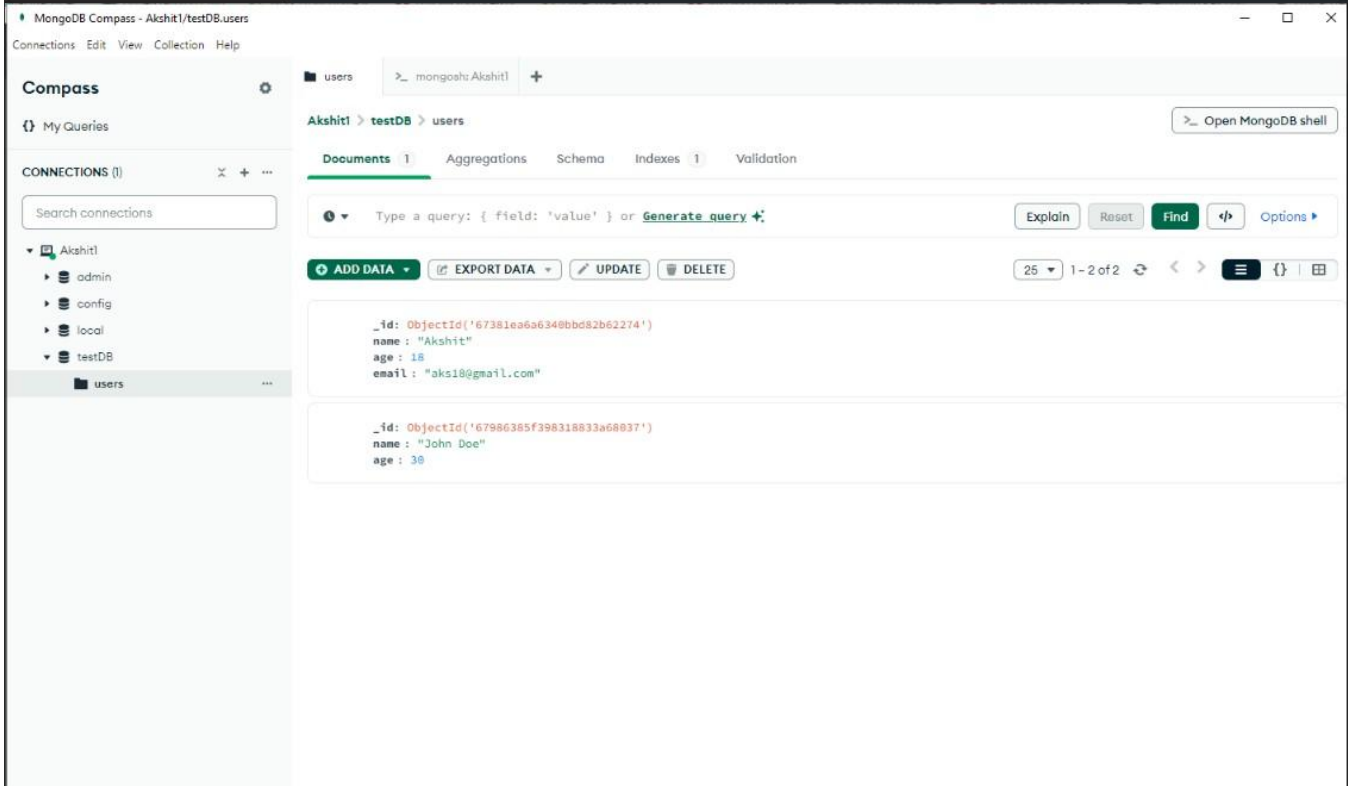
6. Perform basic operations:
   - Insert a document: db.users.insert({name: "John Doe", age: 30})
   - Query documents: db.users.find()
   - Update a document: db.users.update({name: "John Doe"}, {$set: {age: 31}})
   - Delete a document: db.users.remove({name: "John Doe"})



```
> use testDB
< switched to db testDB
> db.users.insert({name: "John Doe", age: 30})
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('67986385f398318833a68037')
    }
  }
> db.users.find()
< {
    _id: ObjectId('67381ea6a6340bbd82b62274'),
    name: 'Akshit',
    age: 18,
    email: 'aks18@gmail.com'
  }
  {
    _id: ObjectId('67986385f398318833a68037'),
    name: 'John Doe',
    age: 30
  }
> db.users.update({name: "John Doe"},{$set: {age:31}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
```

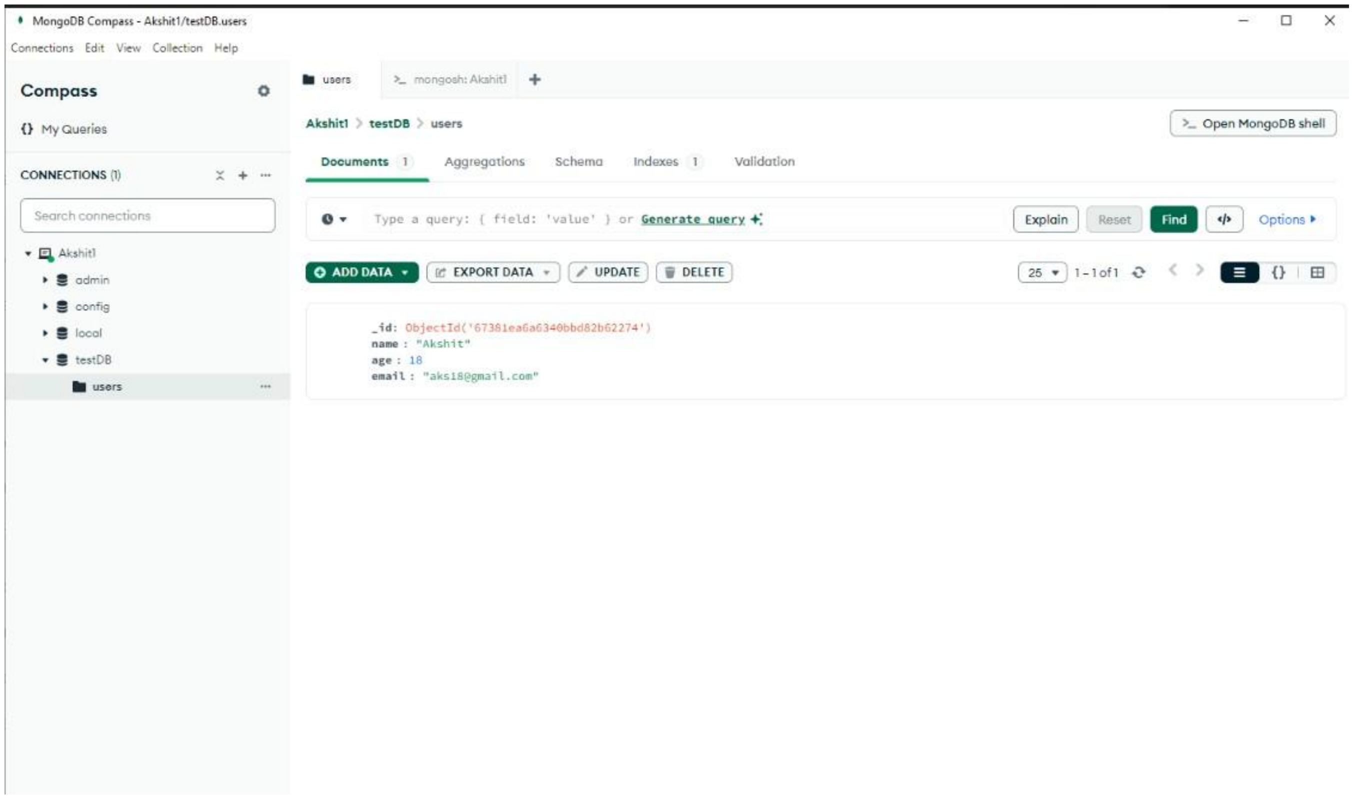


```
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.users.find()
< {
    _id: ObjectId('67381ea6a6340bbd82b62274'),
    name: 'Akshit',
    age: 18,
    email: 'aks18@gmail.com'
  }
  {
    _id: ObjectId('67986385f398318833a68037'),
    name: 'John Doe',
    age: 31
  }
>  db.users.remove({name: "John Doe"})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.users.find()
< {
    _id: ObjectId('67381ea6a6340bbd82b62274'),
    name: 'Akshit',
    age: 18,
    email: 'aks18@gmail.com'
  }
testDB >
```



```
_id: ObjectId('67381ea6a6340bbd82b62274')
name : "Akshit"
age : 18
email : "aks18@gmail.com"

_id: ObjectId('67986385f398318833a68037')
name : "John Doe"
age : 30
```



```
_id: ObjectId('67381ea6a6340bbd82b62274')
name : "Akshit"
age : 18
email : "aks18@gmail.com"
```

BOOKS AND WEB RESOURCES:

1. Installing MongoDB Tutorial Online
   Available:https://www.youtube.com/playlist?list=PL4cUxeGkcC9h77dJ-QJlwGlZlTd4ecZOA
2. Learning React by Alex Banks and Eve Porcello
3. MongoDB: The Definitive Guide by Kristina Chodorow
4. Node.js Design Patterns by Mario Casciaro
5. Express in Action by Evan Hahn

Web Resources

1. MongoDB Documentation: https://www.mongodb.com/docs/
2. React Official Documentation: https://reactjs.org/docs/
3. Node.js Documentation: https://nodejs.org/docs/
4. Express.js Guide: https://expressjs.com/

Videos and Blogs

1. Traversy Media: MERN Stack Tutorial (YouTube Channel)
2. Academind: MERN Stack Crash Course (YouTube Channel)
3. FreeCodeCamp MERN Tutorial (FreeCodeCamp Blog)