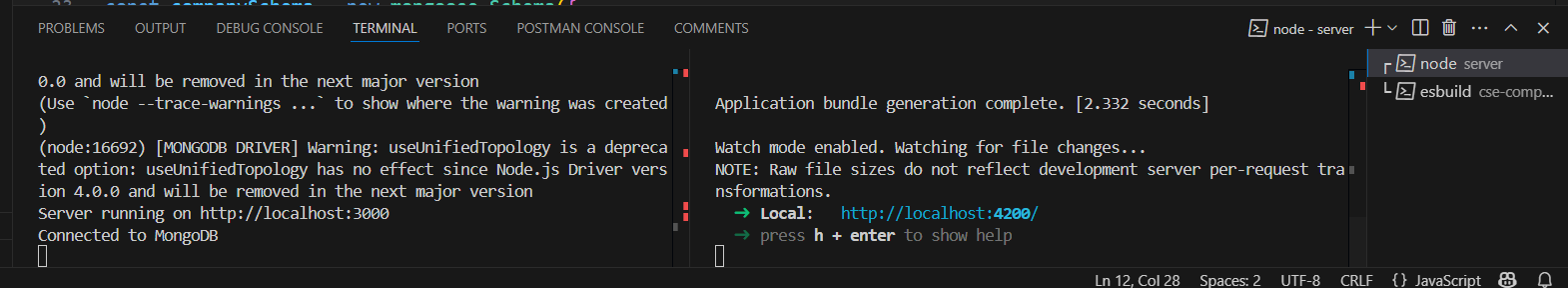
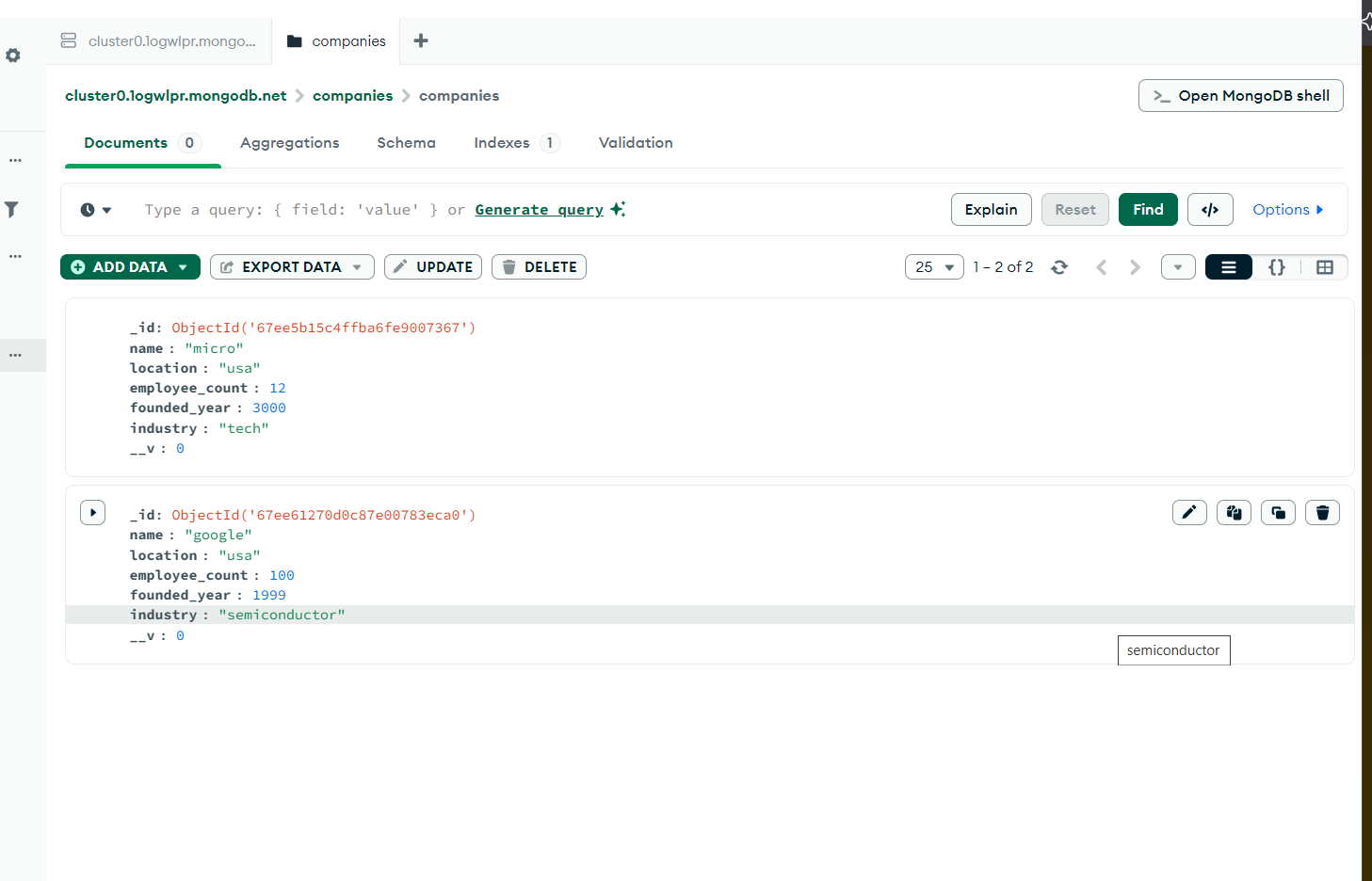
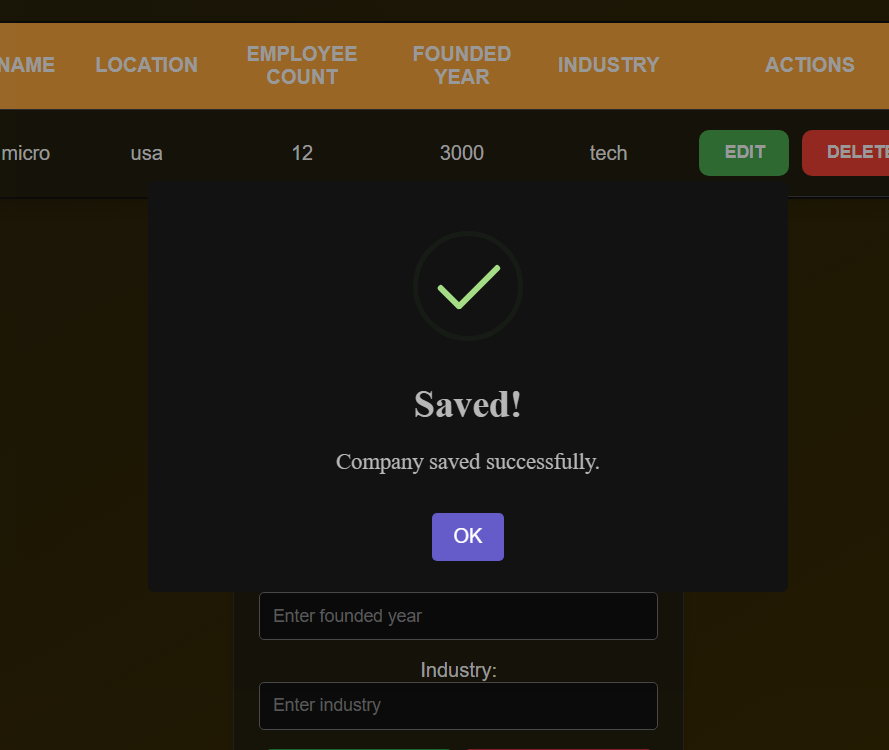
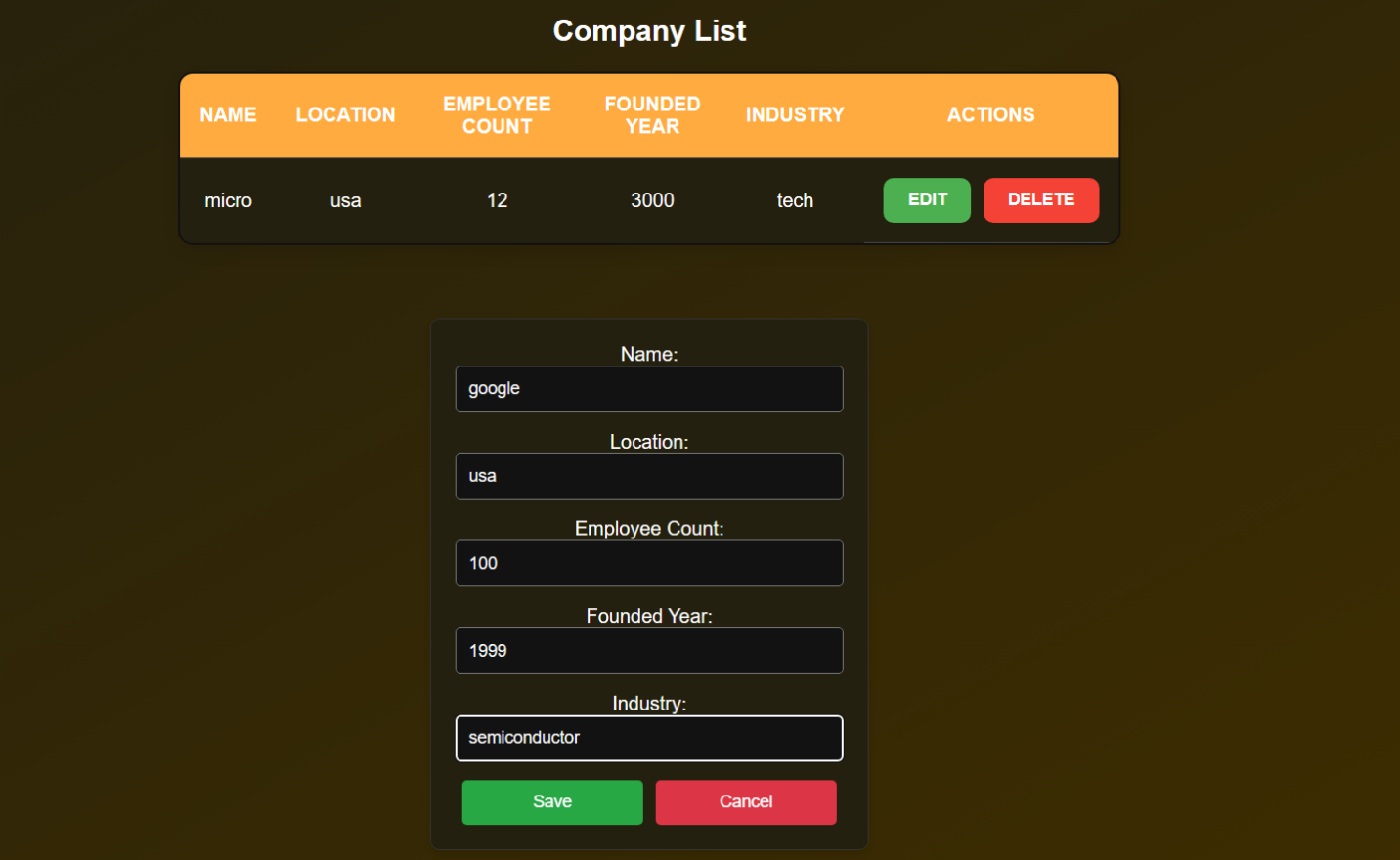
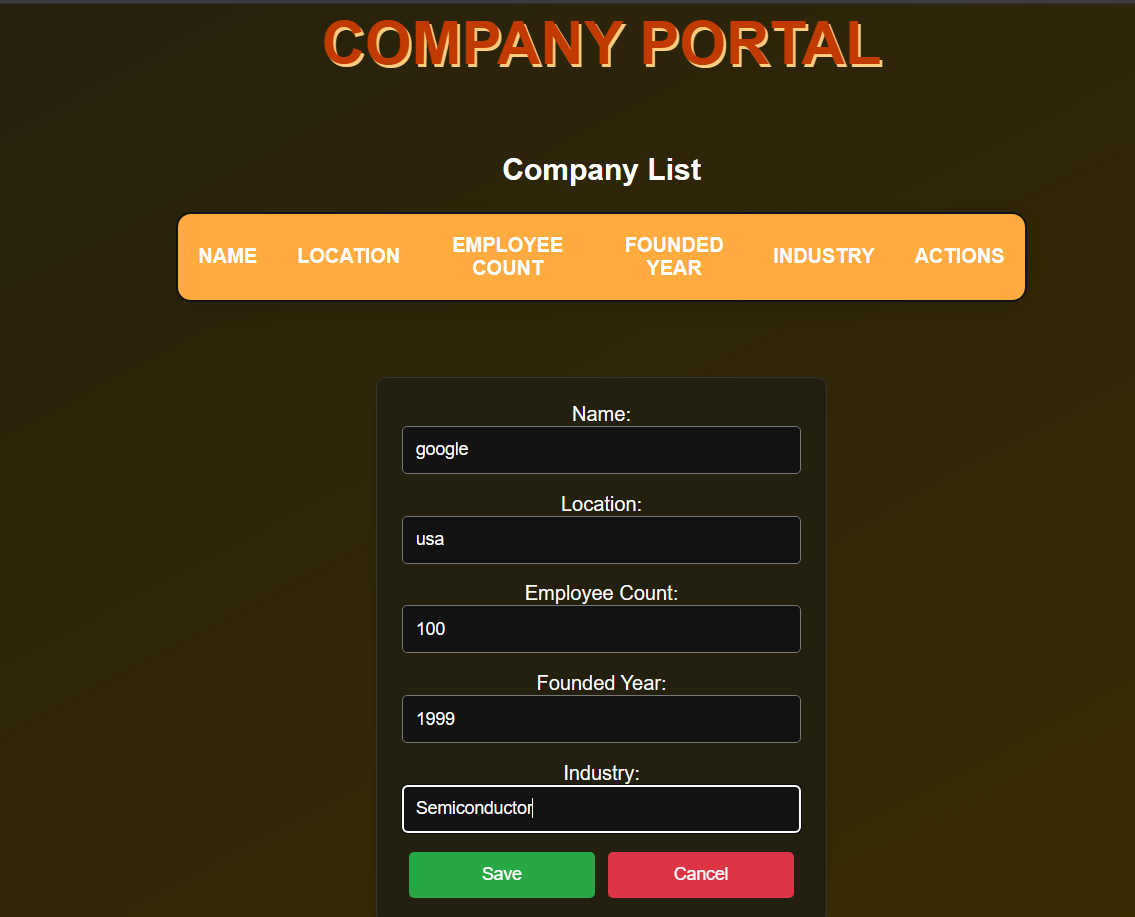
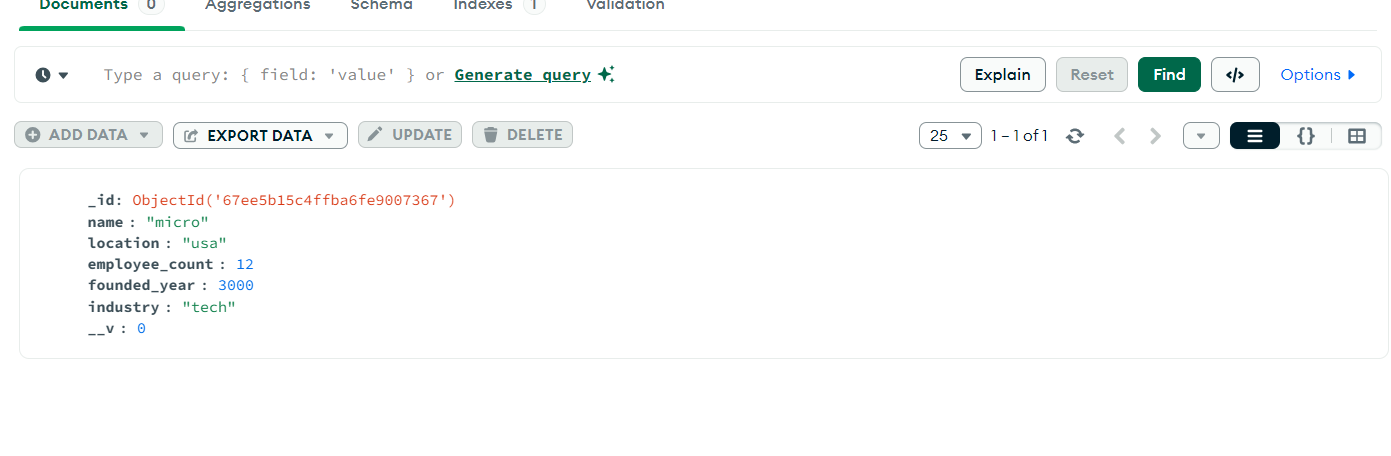
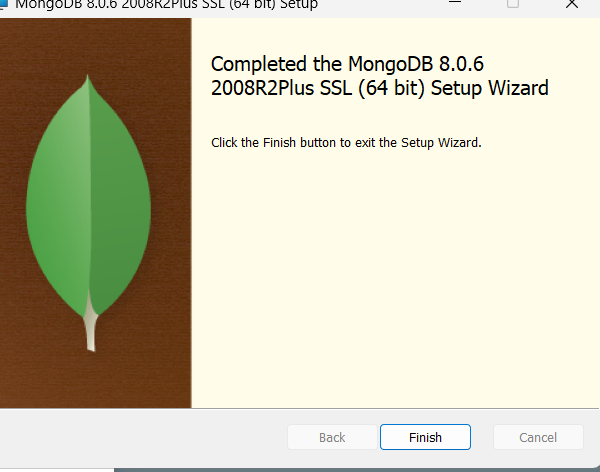
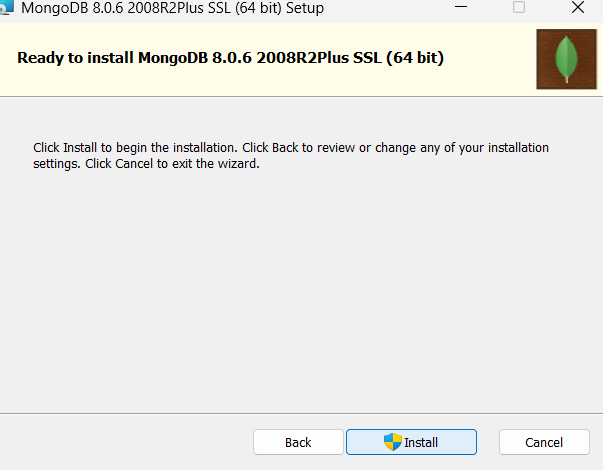
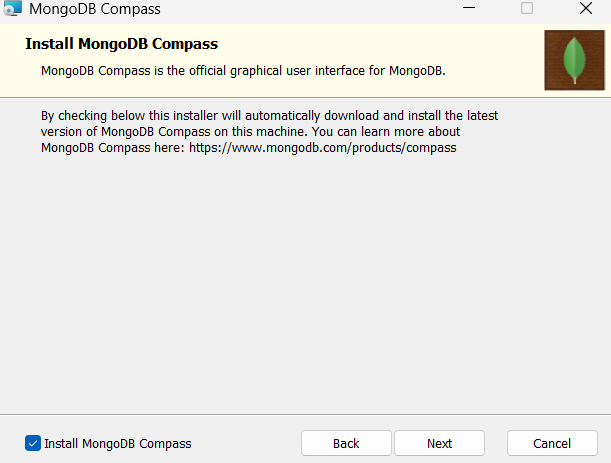
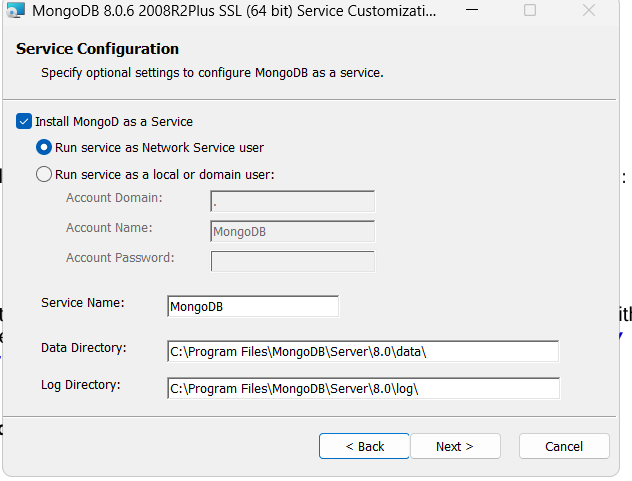
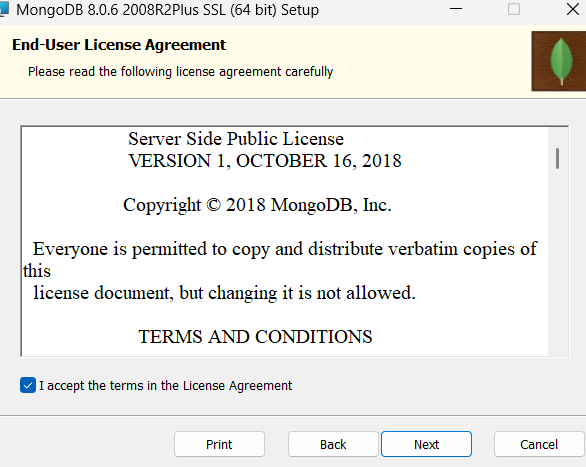
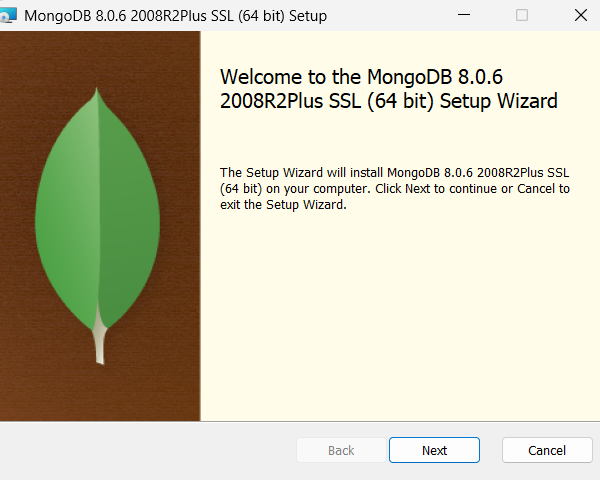
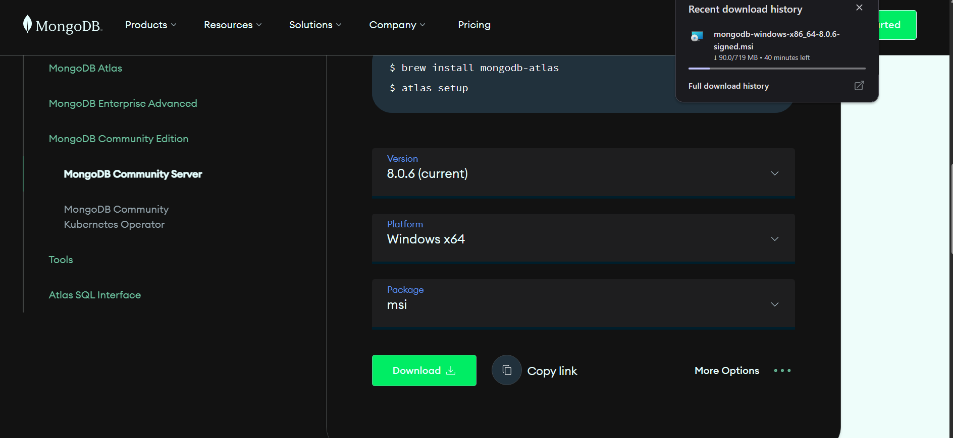
**Batch T4**

**Practical No. 9**

**Title of Assignment :**

**Student Name: Parshwa Herwade**

**Student PRN: 22510064**

Install & deploy the following cloud databases on Windows / Linux platform : A] MongoDB B] CassandraDB Create the web form in angular 19, to demonstrate the CRUD operations with above backend cloud databases. Assume any database. Create database by your PRN.

**Objective / Aim**

1. Install and configure MongoDB and CassandraDB on Windows/Linux.
2. Deploy both cloud databases locally or on a cloud VM.
3. Build an Angular 19 web application that performs Create, Read, Update, and Delete operations on the chosen database backend.
4. Demonstrate end-to-end connectivity and data flow between frontend and backend databases.

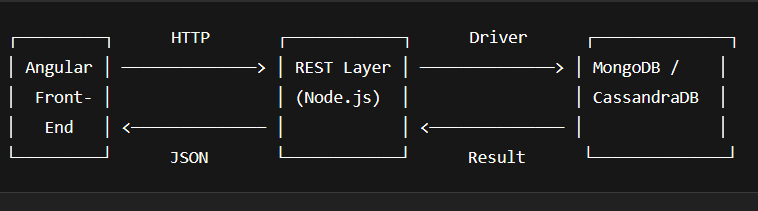
**Introduction**  
NoSQL databases such as MongoDB and CassandraDB offer flexible schema designs and high scalability. MongoDB is a document-oriented database storing JSON-like documents; CassandraDB is a wide-column store optimized for high availability and write throughput. This lab explores installing both, understanding their data models, and integrating them with a modern frontend (Angular 19).

**Theory / Algorithms**

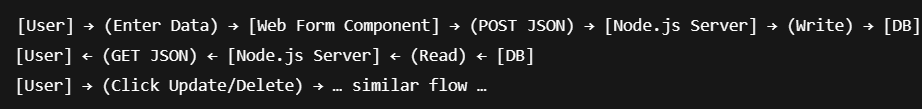
* **MongoDB**
  + Document Model: Collections of BSON documents
  + CRUD operations via mongo shell or drivers (e.g., Mongoose)
  + Indexing, replication, and sharding fundamentals
* **CassandraDB**
  + Column-Family Model: Keyspace → Tables (Column Families)
  + Partitioning and clustering keys for data distribution
  + CRDT-based eventual consistency, tunable consistency levels
* **Angular 19 HTTP CRUD Flow**
  + User action in component →
  + Service call using HttpClient →
  + REST API endpoint →
  + Database driver executes operation →
  + Response propagates back to UI

**Documentation**

1. **Functional Block Diagram**



1. **Data Flow Diagram (DFD Level 0)**



text

CopyEdit

[User] → (Enter Data) → [Web Form Component] → (POST JSON) → [Node.js Server] → (Write) → [DB]

[User] ← (GET JSON) ← [Node.js Server] ← (Read) ← [DB]

[User] → (Click Update/Delete) → … similar flow …

**Procedure**

1. **Environment Setup**
   * Install Node.js (v18.x) and Angular CLI (v19).
   * Install Java 11 (required by Cassandra).
2. **MongoDB Installation**
   * Download MongoDB Community Edition.
   * sudo apt-get install -y mongodb-org (Linux) or Windows MSI.
   * Start service: sudo systemctl start mongod.
3. **CassandraDB Installation**
   * Download Apache Cassandra.
   * Unzip and configure cassandra.yaml.
   * Start service: bin/cassandra -f.
4. **Angular 19 Project**
   * ng new cs371-db-lab9 --routing=true --style=css
   * Generate service & components:

bash

CopyEdit

ng generate service services/db

ng generate component components/crud

1. **Backend REST API**
   * Initialize Node.js project: npm init -y.
   * Install Express, body-parser, database drivers:

bash

CopyEdit

npm install express body-parser mongoose cassandra-driver cors

* + Create endpoints in app.js for /api/items supporting GET, POST, PUT, DELETE.
  + For MongoDB: connect via Mongoose URI mongodb://localhost:27017/<PRN>.
  + For Cassandra: connect via new cassandra.Client({contactPoints:['127.0.0.1'], keyspace: '<PRN>'}).

1. **Angular Service**
   * In db.service.ts, implement methods getAll(), create(item), update(id, data), delete(id) using HttpClient.
2. **CRUD Component**
   * Build form for add/edit; display table of items with Edit/Delete buttons.
   * Bind form submission to service calls; refresh list on success.
3. **Run & Test**
   * Start backend: node app.js.
   * Start frontend: ng serve.
   * Access at http://localhost:4200, perform CRUD and observe database with mongo shell or cqlsh.

**Conclusion**

* Successfully installed and deployed MongoDB and CassandraDB on the target platform.
* Built an Angular 19 application that performs full CRUD on both NoSQL backends.
* Observed differences in data modeling and query syntax between MongoDB’s document store and Cassandra’s column-family design.
* Both integrations demonstrated stable, bi‑directional data flow between frontend and database layers