**From AIA:**  
mongo

use chats

db.dropDatabase()  
  
  
**connection:**

const mongoose = require('mongoose');

mongoose.connect(

'mongodb://127.0.0.1:27017/chats',

{ useUnifiedTopology: true, useNewUrlParser: true },

(err) => {

if (err) console.log(err);

else console.log('moogodb connect successfully');

}

);

**Queries:**[**https://docs.mongodb.com/manual/crud/**](https://docs.mongodb.com/manual/crud/)<https://docs.mongodb.com/manual/tutorial/query-documents/>

show all data bases-> show databases  
use this database-> use dbname  
  
  
  
**Mongo DB:**Mongodb is cross platform and open source document oriented database.Mongo Db stores data into the form of BJSON. Its similar to json but BJSON is a binary representation of JSON.

**Components:**

* **Mongod:**

[mongod](https://docs.mongodb.com/manual/reference/program/mongod/" \l "mongodb-binary-bin.mongod) is the primary daemon process for the MongoDB system. It handles data requests, manages data access, and performs background management operations.

* **Mongo:**

**mongo** is an interactive JavaScript shell interface **to MongoDB**, which provides a powerful interface for system administrators as well as a way for developers **to** test queries and operations directly **with** the database. **mongo** also provides a fully functional JavaScript environment for use **with** a **MongoDB**.  
we use mongo shell to communicate with db.

* **Mongoose:**

Object Mapping between Node and MongoDB managed via Mongoose.  
Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.  
Mongoose is a **[MongoDB](https://www.mongodb.org/)** object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

install mongo through “npm install mongoose ”  
we this for connecting to DB and define sachems etc.

* **Replica:**

A **replica** set in **MongoDB** is a group of mongod processes that maintain the same data set. **Replica** sets provide redundancy and high availability, and are the basis for all production deployments. This section introduces **replication in MongoDB** as well as the components and architecture of **replica** sets.  
<https://docs.mongodb.com/manual/replication/>

* **Sharding:**

[Sharding](https://docs.mongodb.com/manual/reference/glossary/" \l "std-term-sharding) is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.  
<https://www.guru99.com/mongodb-sharding-implementation.html>  
<https://docs.mongodb.com/manual/sharding/>

* **Sharded cluster:**

The set of nodes comprising a [sharded](https://docs.mongodb.com/manual/reference/glossary/#std-term-sharding) MongoDB deployment. A sharded cluster consists of config servers, shards, and one or more [mongos](https://docs.mongodb.com/manual/reference/program/mongos/#mongodb-binary-bin.mongos) routing processes.

* **Shard:**

A shard is a single MongoDB instance that holds a subset of the sharded data. Shards can be deployed as replica sets to [increase availability and provide redundancy](https://www.bmc.com/blogs/redundancy-impact-availability/). The combination of multiple shards creates a complete data set. For example, a 2 TB data set can be broken down into four shards, each containing 500 GB of data from the original data set.

* **Config Server/ file:**

Config servers store the metadata for a [sharded cluster](https://docs.mongodb.com/manual/reference/glossary/#std-term-sharded-cluster). The metadata reflects state and organization for all data and components within the sharded cluster. The metadata includes the list of chunks on every shard and the ranges that define the chunks.

Configuration servers store the metadata and the configuration settings for the whole cluster.

**Replication**: A replica set in **MongoDB** is a group of mongod processes that maintain the same data set. **Sharding**: **Sharding** is a method for storing data across multiple machines.

**Important Links:**<https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/>

<https://www.bmc.com/blogs/mongodb-sharding-explained/>