**Francis Institute of Technology Borivali (West), Mumbai-400103**

**(Autonomous Institute)**

**Department of Information Technology**

**Academic Year: 2024-25**

**Class: TE-ITA/B                                                                            Semester: VI**

**Subject: Web Lab**

**Experiment – 7: Node.js and MongoDB connectivity and using Mongoose for Structured Schema and Validation.**

1. **Aim:** To connect MongoDB with NodeJS and use mongoose for structured schema.
2. **Objectives:** Aim of this experiment is that, the students will be able

* To connect MongoDB with NodeJS
* Create a structured schema and collection

1. **Outcomes:** After study of this experiment, the students will be able

* To perform CRUD operations
* to handle coding and syntax error

1. **Prerequisite:** Basic understanding of database, MongoDB data types and commands, NodeJS etc.
2. **Requirements:** Personal Computer, Windows operating system, VSCode, TypeScript, browser, Internet Connection, google doc, Node JS, MongoDB.
3. **Pre-Experiment Exercise:**

**Brief Theory:** Refer shared material

1. **Laboratory Exercise**
   * + 1. **Procedure:**

**a. Answer the following:**

## **What is the use of mongoose library?**

**ANS:**

1. MongoDB ODM – Acts as an Object Data Modeling (ODM) library for MongoDB.
2. Schema Definition – Defines structured schemas for MongoDB collections.
3. Data Validation – Ensures data integrity with built-in validation.
4. Middleware Support – Allows pre and post hooks for operations.
5. Query Building – Provides powerful query functions like .find(), .populate().
6. Model Methods – Enables creation of custom instance and static methods.
7. Relationship Handling – Supports references (ref) between collections.
8. Indexing – Optimizes performance using indexing features.
9. Transactions – Supports MongoDB transactions for atomic operations.

## **2. How to create a schema using mongoose?**

**ANS:**

Import Mongoose – const mongoose = require('mongoose');  
Define Schema – const userSchema = new mongoose.Schema({...});  
Set Field Types – Define properties like String, Number, Date, etc.  
Add Validation – Use required fields, min/max values, and regex patterns.  
Default Values – Set default values for fields.  
Reference Other Collections – Use ref for relationships.  
Timestamps – Enable timestamps: true to auto-track creation and updates.  
Virtual Fields – Define computed properties not stored in DB.  
Schema Methods – Attach custom functions to the schema.

### **3. Why to use await inside the async () method?**

**ANS:**

1. Handles Asynchronous Operations – Waits for a Promise to resolve.
2. Avoids Callbacks – Eliminates callback hell.
3. Improves Readability – Code looks more synchronous and cleaner.
4. Ensures Sequential Execution – Executes code in order without then().
5. Error Handling – Works well with try...catch for error management.
6. Prevent Race Conditions – Ensures one operation completes before another starts.
7. Reduces Nesting – Avoids deeply nested .then() chains.
8. Better Debugging – Stack traces are clearer than with .then().
9. Ensures Proper Data Flow – Waits for dependent operations to complete.
10. Mandatory for Promise Return Values – Needed when working with Promises.

**b**. **Attach screenshots:**

* MongoDB code and output with your own comments.

**8. Post-Experiments Exercise**

1. **Extended Theory:**

Nil

1. **Questions:**

### What are the benefits of mongoose in MongoDB?

### How to create a model?

1. **Conclusion:**

* Write what was performed in the experiment.
* Write the significance of the topic studied in the experiment.

1. **References:**

***1. MongoDB in Action, Second Edition, by Kyle Banker, Peter Bakkum***

***2. https://docs.mongodb.com/manual/***

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |