Batch: D-2	**Roll No.:**16010122151
Experiment / assignment / tutorial No.____	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title:Mongo DB and Mongoose

AIM: To Implement the Concept of Advanced JavaScript

Problem Definition:

1. Implement CRUD (Create, Read, Update, Delete) operations in an NodeJS/Express.js application using Mongoose.
2. Implement Student Application form and store the data in MongoDB database.

*(Students have to perform the task assigned within group and demonstrate the same).
Resources used:

Expected OUTCOME of Experiment:

CO 1:.Build full stack applications in JavaScript using the MERN technologies.

Books/ Journals/ Websites referred:

1. Shelly Powers Learning Node O' Reilly 2 nd Edition, 2016.

Pre Lab/ Prior Concepts:

Implementation Details:

```
{
  "name": "mern-8",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.3",
    "express": "^4.21.1",
    "mongoose": "^8.7.1"
  }
}
```

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const studentRoutes = require('./routes/studentRoutes');

const app = express();
app.use(bodyParser.json());
app.use(studentRoutes);
```

```
mongoose.connect('mongodb://localhost:27017/studentDB');
```

```
const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

```
const express = require('express');
const router = express.Router();
const Student = require('../models/student');

// Create a new student (C)
router.post('/students', async (req, res) => {
  const student = new Student({
    name: req.body.name,
    age: req.body.age,
    course: req.body.course
  });
  try {
    await student.save();
    res.status(201).send(student);
  } catch (error) {
    res.status(400).send(error);
  }
});
// Read all students (R)
router.get('/students', async (req, res) => {
  try {
    const students = await Student.find();
    res.status(200).send(students);
  } catch (error) {
    res.status(500).send(error);
  }
});
// Read a single student by ID (R)
router.get('/students/:id', async (req, res) => {
  try {
    const student = await Student.findById(req.params.id);
    if (!student) return res.status(404).send('Student not found');
    res.status(200).send(student);
  } catch (error) {
    res.status(500).send(error);
  }
});
// Update a student by ID (U)
router.put('/students/:id', async (req, res) => {
  try {
    const student = await Student.findByIdAndUpdate(req.params.id, req.body, { new: true });
    if (!student) return res.status(404).send('Student not found');
    res.status(200).send(student);
  } catch (error) {
    res.status(400).send(error);
  }
});
// Delete a student by ID (D)
router.delete('/students/:id', async (req, res) => {
  try {
    const student = await Student.findByIdAndDelete(req.params.id);
```

```
    if (!student) return res.status(404).send('Student not found');
    res.status(200).send('Student deleted');
  } catch (error) {
    res.status(500).send(error);
  }
});
module.exports = router;
```

```
const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  age: {
    type: Number,
    required: true
  },
  course: {
    type: String,
    required: true
  }
});
module.exports = mongoose.model('Student', studentSchema);
```

All crud operations

Conclusion:

Learned mongoose and how to do crud operations with expressjs and nodejs