Practical No 01:

Aim: Write a program to implement MongoDB data models.

Step 1: Creating **index.js** and **model.js** file.

Step 2: Run the **“npm init”** command in the terminal to initialize the required files.

Step 3: Installing “mongoose” package using **“npm i mongoose”** or **“npm install mongoose”** in the project.

Step 4: Editing **index.js** file
Code:

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

mongoose.set("strictQuery", true);

mongoose.connect("mongodb://127.0.0.1:27017/test"
  )
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));

// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});

// Defining Student model
const Student = mongoose.model("Student", studentSchema);

// Create collection of Model
Student.createCollection().then(function () {
  console.log("Collection is created!");
});
```

Step 5: Editing **model.js** file

Code:

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

//Scheme for collection
const studentSchema = new mongoose.Schema(
  {
    name: String,
    rollNo: String,
    class: String,
    contactNo: String,
    email: String,
  },
  { collection: "students" }
);

//Exporting scheme
module.exports = mongoose.model("student", studentSchema);
```

Step 6: Executing **"node index.js"** command in terminal

Practical No 02:

Aim: Write a program to implement CRUD operations on MongoDB.

Step 1: Creating **createCollection.js, insertOne.js, insertmany.js, getdata.js, update.js, and delete.js** file.

Step 2: Run the **"npm init"** command in the terminal to initialize the required files.

Step 3: Installing "mongoose" package using **"npm i mongoose"** or "npm install mongoose" in the project.

Step 4: Editing **createCollection.js** file
Code:

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

mongoose.connect("mongodb://127.0.0.1:27017/test"
  )
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));
```

```
// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});

// Defining Student model
const Student = mongoose.model("Student", studentSchema);

// get reference to database
var db = mongoose.connection;

// function to create collection of Model
Student.createCollection().then(function () {
  console.log("Collection is created!");
});

// To Check error
db.on("error", console.error.bind(console, "connection error:"));
```

insertOne.js file:
<u>Code:</u>

```
const mongoose = require("mongoose");
mongoose.connect("mongodb://127.0.0.1:27017/test"
  )
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));
// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});
// Defining Student model
const Student = mongoose.model("Student", studentSchema);
```

```
// get reference to database
var Student1 = new Student({
  name: "Zaid",
  rollNo: 31,
  class: "SyCs",
  age: 19,
  email: "sybsccsz@gmail.com",
});

Student1.save()
  .then(result => {
    console.log("Data Inserted!");
  })
  .catch(err => {
    console.log(err);
  });
```

insertMany.js file:
Code:

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

mongoose.connect("mongodb://127.0.0.1:27017/test")
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));

// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});

// Defining Student model
const Student = mongoose.model("Student", studentSchema);

// To insert Multi data in db
// save model to database
Student.insertMany([
  {
    name: "test",
    rollNo: 31,
    class: "SyCs",
    age: 20,
    email: "testmail1@gmail.com",
  },
  {
    name: "test1",
    rollNo: 32,
    class: "SyCs",
    age: 18,
    email: "testmail2@gmail.com",
  },
  {
    name: "test2",
    rollNo: 33,
    class: "SyCs",
    age: 25,
    email: "testmail3@gmail.com",
  },
  {
    name: "test3",
    rollNo: 34,
    class: "SyCs",
    age: 21,
    email: "testmail4@gmail.com",
  },
])
  .then(function () {
    console.log("Data inserted"); // Success
  })
  .catch(function (error) {
    console.log(error); // Failure
  });
```

getData.js file:
Code:

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

mongoose.connect("mongodb://127.0.0.1:27017/test")
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));

// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});

// Defining Student model
const Student = mongoose.model("Student", studentSchema);

// To get All data from db
Student.find({})
  .then((data) => {
    console.log("Data:");
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });
```

update.js file:
Code:

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

mongoose
  .connect("mongodb://127.0.0.1:27017/test")
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));

// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});
```

```
// Defining Student model
const Student = mongoose.model("Student", studentSchema);

// To update data in db
Student.updateOne({ name: "test3", age: 30 })
  .then((result) => {
    console.log("Result:", result);
  })
  .catch((err) => {
    console.log(err);
  });
```

delete.js file:
Code:

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

mongoose.connect("mongodb://127.0.0.1:27017/test"
  )
  .then(() => console.log("Connected to database"))
  .catch((err) => console.error("Connection error:", err));

// Creating Schema
const studentSchema = new mongoose.Schema({
  name: String,
  rollNo: Number,
  class: String,
  age: Number,
  email: String,
});

// Defining Student model
const Student = mongoose.model("Student", studentSchema);

// get reference to database
var db = mongoose.connection;

// To update data in db
Student.deleteMany()
  .then(function () {
    console.log("Data deleted"); // Success
  })
  .catch(function (error) {
    console.log(error); // Failure
  });
```

Practical No 03:

Aim: Write a program to perform validation of a form using AngularJS

Step 1: Creating **index.html & welcome.html** file.

Step 2: Editing **index.html** file
Code:

```
<!DOCTYPE html>
<html>

<head>
    <title> AngularJs Form Validation </title>
    <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></scri
pt>
    <script>
        var app = angular.module('formApp', []);
        app.controller('formCtrl', function ($scope) {
            $scope.sendForm = function () {
                window.open("welcome.htm");
                $scope.msg = 'Form Submited Successfully';
            };
            $scope.getClass = function (color) {
                return color.toString();
            }
        });
    </script>

    <style>
        .valid.false {
            background: red;
        }

        .valid.true {
            background: green;
        }
        .error {
            color: red;
        }
    </style>
</head>

<body ng-app="formApp" ng-controller="formCtrl" bgcolor="white">
    <h3>Form validation demo app in AngularJs</h3>

    <form name="personForm" ng-submit="sendForm()">
        <label for="name">Name</label>
        <input id="name" name="name" type="text" ng-model="person.name"
required />
        <span class="error" ng-show="personForm.name.$error.required"> Required!
</span>
        <br /><br />

        <label for="adress">Adress</label>
        <input id="address" name="address" type="text" ng-
model="person.address" required />
        <span class="error" ng-show="personForm.address.$error.required">
Required! </span>
        <br /><br />

        <label for="contact">Contact No</label>
        <input id="mobile" name="mobile" type="number" ng-model="person.mobile"
required />
        <span class="error" ng-show="personForm.mobile.$error.required">Required
number!</span>
        <span class="error" ng-show="personForm.mobile.$error.mobile">Invalid
mobile!</span>
        <br /><br />

        <label for="email">Email</label>
        <input id="email" name="email" type="email" ng-model="person.email"
required />
        <span class="error" ng-
show="personForm.email.$error.required">Required!</span>
        <span class="error" ng-show="personForm.email.$error.email">Invalid
Email!</span>
```

```
      <br /><br />

      <input type="checkbox" ng-model="terms" name="terms" id="terms"
required />
      <label for="terms">I Agree to the terms.</label>
      <span class="error" ng-show="personForm.terms.$error.required">You must
agree to the terms</span>
      <br /><br />

      <button type="submit">Submit Form</button>
      <br /><br />

      <span>{{msg}}</span>
   </form>
</body>

</html>
```

Step 3: Editing **welcome.html** file

Code:

```
<html>

<head>
   <title>Welcome Page</title>
</head>

<body bgcolor="white">
   <h1>Record Successfully Submitted...........</h1>
</body>

</html>
```