Practical: 01 Create a user-friendly interface with a clean and proper design.

```
<!DOCTYPE html>
<html ng-app="todoApp">
        <head>
   <meta charset="UTF-8">
   <title>To-Do List</title>
   <style>
      /* CSS */
    body { font-family: Arial; display: flex; justify-content: center; align-items:
center;   height: 100vh; margin: 0; }
  .container { width: 300px; padding: 20px; box-shadow: 0 0 10px rgba(0,0,0,0.1); }
   input, button { padding: 10px; margin: 5px; }
      .done { text-decoration: line-through; }
  </style>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"> </script>
   <!-- javaScript -->
  <script>
  angular.module('todoApp', []).controller('TodoController', function() {
         this.todos = [{text: 'Learn AngularJS', done: false}, {text: 'Build a to-do
app', done: false}];
  this.addTodo = () => { if (this.newTodo) { this.todos.push({text:
this.newTodo, done: false}); this.newTodo = ''; } };
         this.removeTask = (index) => { this.todos.splice(index, 1); };
       });
   </script>
```

```
</head>
<body ng-controller="TodoController as ctrl">
   <div class="container">
<form ng-submit="ctrl.addTodo()"><input ng-model="ctrl.newTodo" placeholder="Add
task"><button>Add</button></form>
  <ul><li ng-repeat="todo in ctrl.todos"><input type="checkbox" ng
model="todo.done"><span ng-class="{'done': todo.done}">{{todo.text}}</span><button
ng-click="ctrl.removeTask($index)">Remove</button></li></ul>
        </div>
</body>
</html>
```

Practical 02;- Develop AngularJS program to create a login form, with validation for the username and password fields.

```
<!DOCTYPE html>
<html ng-app="loginApp">
         <head>
             <title>Login</title>
 <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
 <style>
 body { background: burlywood; display: flex; justify-content: center; align-items: center;
height: 100vh; margin: 0; }
   .form-container { background: chocolate; padding: 70px; border-radius: 8px; box
shadow: 0 0 10px black; text-align: center; }
 .error { color: red; }
    form.ng-pristine.ng-invalid.ng-invalid-required { display: flex; flex-direction: column; }
    button { color: red; padding: 9px; border-radius: 20px; }
```

```
 </style>
</head>
            <body ng-controller="loginCtrl">
 <div class="form-container">
  <h2>Firewall Authentication</h2>
  <form name="loginForm" ng-submit="submitForm()" novalidate>
   <label>Username: <input type="text" name="username" ng-model="user.username"
required>
  <span class="error" ng-show="loginForm.username.$dirty &&
loginForm.username.$invalid">Required</span>
```

```
 </label>
    <label>Password: <input type="password" name="password" ng
model="user.password" required>
 <span class="error" ng-show="loginForm.password.$dirty &&
loginForm.password.$invalid">Required</span>
 </label>
    <button type="submit" ng-disabled="loginForm.$invalid">Login</button>
    <div ng-show="loginSuccess" style="color: black;">Login Successful!</div>
            </form>
            </div>
 <script>
 angular.module('loginApp', []).controller('loginCtrl', function($scope) {
    $scope.submitForm = function() {
      $scope.loginSuccess = $scope.user.username === 'student' && $scope.user.password
=== 'icoet';
      if (!$scope.loginSuccess) $scope.user = {};
```

```
    };
  });
 </script>
</body>
</html>
```

Practical:-03 To demonstrate AngularJS services and data bind.

```
 <!DOCTYPE html>
<html ng-app="myApp">
      <head>
  <title>Data Binding Example</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<script>
      angular.module('myApp', [])
      .service('UserService', function() {
        this.getUserData = () => ({ name: 'SSBT COET', email: 'ssbt
coer@example.com' });
      })
 .controller('MainController', ['$scope', 'UserService', function($scope, UserService) {
        $scope.title = 'AngularJS Example';
        $scope.userData = UserService.getUserData();
        $scope.user = { name: '' };
      }]);
 </script>
</head>
<body ng-controller="MainController">
  <h1>{{ title }}</h1>
  <label>Enter your name:</label>
  <input type="text" ng-model="user.name">
```

```
  <h2>Welcome, {{ user.name }}!</h2>
 <h3>Stored Data from Service:</h3>
  <p>Name: {{ userData.name }}</p>
  <p>Email: {{ userData.email }}</p>
</body>
</html>
```

Practical 4:- To display tasks in a list with checkboxes for marking completion as per user choice.

```
<!DOCTYPE html>
<html lang="en">
      <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Task List</title>
            <style>
    body { font-family: Arial, sans-serif; }
    .task-list { list-style: none; padding: 0; }
    .task-item { margin: 5px 0; display: flex; align-items: center; }
    .task-item label { margin-left: 10px; }
    .completed label { text-decoration: line-through; color: gray; }
    .add-task, .subject-select { margin: 10px 0; }
  </style>
</head>
      <body>
  <h1>Task List</h1>
  <div class="subject-select">
    <label for="subject-select">Select Subject:</label>
    <select id="subject-select">
      <option>AngularJS</option>
      <option>E-Commerce</option>
```

```
        <option>Cloud Computing</option>
      </select>
    </div>
    <input type="text" id="task-input" placeholder="New task">
    <button onclick="addTask()">Add Task</button>
    <h2>Tasks</h2>
    <ul id="task-list" class="task-list"></ul>
<script>
 function addTask() {
          const taskText = document.getElementById('task-input').value.trim();
          if (!taskText) return;
         const subject = document.getElementById('subject-select').value;
          const taskItem = document.createElement('li');
          taskItem.className = 'task-item';
          taskItem.innerHTML = `
             <input type="checkbox" onclick="toggleCompletion(this)">
             <label>${taskText} - <strong>${subject}</strong></label>
             <button onclick="removeTask(this)">Remove</button>
          `; document.getElementById('task-list').appendChild(taskItem);
 document.getElementById('task-input').value = '';
      }
          function toggleCompletion(checkbox) {
          checkbox.parentElement.classList.toggle('completed', checkbox.checked);
      }
      function removeTask(button) {
          button.parentElement.remove();
      }
    </script>
</body>
```

```
</html>
```

Practical 5:- To use ng-if directive to display tasks in the UI.

```
 <!DOCTYPE html>
<html ng-app="taskApp">
<head>
  <meta charset="UTF-8">
  <title>Task List</title>
<scriptsrc="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"
</script>
  <script>
    // AngularJS application setup
    var app = angular.module('taskApp', []);
    app.controller('TaskController', function($scope) {
      $scope.tasks = [
        { name: 'Complete assignment', done: true },
        { name: 'Read a book', done: true },
        { name: 'Go for a walk', done: false }
      ];
    });
  </script>
</head>
<body ng-controller="TaskController">
  <h2>Tasks to Complete</h2>
  <ul>
    <li ng-repeat="task in tasks" ng-if="!task.done">
      {{ task.name }}
    </li>
  </ul>
  <h2>Completed Tasks</h2>
```

```
  <ul>
 <li ng-repeat="task in tasks" ng-if="task.done">
        {{ task.name }}
```



```
      </li>
   </ul>
</body>
 </html>
```

Prqctical 6:- To create database and structure in MongoDB.

1. Install MongoDB (community server) MSI package

2. Install MongoDB Shell (Command line)

3. After installing this we will check the version of MongoDB Mongod –version

4. Open a MongoDB shell using CMD Mongosh

5. Creating Database in MongoDB use bca Output:

6. Creating a Collection in MongoDB Db.createCollection(“mongodb”)

Practical 7:- To demonstrate insert, update, delete, select operations in MongoDB

1. Create A database:

2. Create A Collection

3. Insert into the collection There are Two ways to insert the data into the collection i. insertOne ii. insertMany

4. Update the data that we inserted There are Two ways to update a data i. updateOne ii. Updatemany

5. Select the data

6. Delete the data There are Two ways to delete the data i. deleteOne ii. deleteMany

Practical 8:- To create collection and insert records in collections.

1. Create A database:

2. Create A Collection

3. Insert into the collection There are Two ways to insert the data into the collection i. insertOne ii. insertMany

Practical 9:- Develop a task manager application using AngularJS for the frontend and MongoDB for the backend

1. Open visual studio code

2. Install some necessary Extensions that required to perform this practical

i. Live Server by ritwick dey

ii. Angular Snippets by john papa

iii. Html-CSS-JavaScript Support by ecmel

iv. Prettier by prettier

v. Npm

vi. Mongodb by mongodb

3. Install some necessary packages through npm (node package manager) to do this practical. This Packages can be installed using visual studio code terminal

i. npm init -y

ii. npm install angular

iii. npm install express mongoose body-parse cors

iv. npm install mongoose

v. npm install cors body-parser

4. Now Create a Directory in visual studio code terminal

mkdir (Directory Name)

5. After creating a Directory we need to change the directory

cd (Directory Name)

6. After changing there is a link of your directory open it using (ctrl+click)

7. After downloading the mongodb extension → click on the extension →Connect the server with the visual studio code using a connection string→ mongodb://localhost:27017 → connect

8. After connecting the localhost we can create a database where we insert a data that shown in our application

9. Creating database and inserting a data

i. Creating databse

ii. Create collection

iii. Insert the data into the collection

10. Open visual studio code →open your folder→create a files

i. Server.js

ii. Index.html

i. Server.js

```
        const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');
        const app = express();
const port = 3000;
       app.use(cors());
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));
        // Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/student')
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('Failed to connect to MongoDB', err));
          const studentSchema = new mongoose.Schema({
  name: String,
  age: Number,
  course: String,
  year: String
}, { collection: 'stu' }); // Collection name is 'stu'
        const Student = mongoose.model('Student', studentSchema);
        // Get all students
app.get('/students', async (req, res) => {
  try {
```

```
const students = await Student.find();
      res.json(students);
    } catch (error) {
      res.status(500).json({ error: 'Failed to fetch students' });
    }
});
            // Add a new student
app.post('/students', async (req, res) => {
    try {
      const student = new Student(req.body);
      await student.save();
      res.json(student);
    } catch (error) {
      res.status(500).json({ error: 'Failed to add student' });
    }
});
               // Update an existing student
app.put('/students/:id', async (req, res) => {
    try {
      const student = await Student.findByIdAndUpdate(req.params.id,
req.body, { new: true });
      res.json(student);
    } catch (error) {
      res.status(500).json({ error: 'Failed to update student' });
    }
});
            // Delete a student
app.delete('/students/:id', async (req, res) => {
```

```
  try {

    await Student.findByIdAndDelete(req.params.id);
    res.json({ message: 'Student deleted' });
  } catch (error) {
    res.status(500).json({ error: 'Failed to delete student' });
  }
});
        // Start the server
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

11. Run this program using command

```
Node server.js
```

ii. Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial
scale=1.0">
  <title>Student Manager</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
 </script>
  <script>
    const app = angular.module('studentApp', []);
```

```
app.controller('studentController', function($scope, $http) {
        $scope.students = [];
        $scope.newStudent = {};



        $scope.editStudent = null;
        // Load students
        $scope.loadStudents = function() {
           $http.get('http://localhost:3000/students')
              .then(function(response) {
                 $scope.students = response.data;
              });
        };
       // Add student
        $scope.addStudent = function() {
           $http.post('http://localhost:3000/students', $scope.newStudent)
              .then(function(response) {
                 $scope.loadStudents();
                 $scope.newStudent = {}; // Clear the form
              });
      //
Edit student
        $scope.editStudentData = function(student) {
           $scope.editStudent = angular.copy(student);
        };
        // Update student
        $scope.updateStudent = function() {
$http.put(`http://localhost:3000/students/${$scope.editStudent._id}`,
$scope.editStudent)
```

```
            .then(function(response) {
              $scope.loadStudents();
              $scope.editStudent = null; // Clear the edit form
            });
```



```
        };
        // Delete student
        $scope.deleteStudent = function(id) {
          $http.delete(`http://localhost:3000/students/${id}`)
            .then(function(response) {
              $scope.loadStudents();
            });
        };
                // Initialize
        $scope.loadStudents();
      });
    </script>
</head>
<body ng-controller="studentController">
    <h1>Student Manager</h1>
    <div>
      <h2>Add New Student</h2>
      <form ng-submit="addStudent()">
        <label>Name:</label>
        <input type="text" ng-model="newStudent.name" required><br>
        <label>Age:</label>
        <input type="number" ng-model="newStudent.age" required><br>
        <label>Course:</label>
```

```
  <input type="text" ng-model="newStudent.course" required><br>
        <button type="submit">Add Student</button>
      </form>
   </div>                                                    43
  <div>
      <h2>Students</h2>
      <ul>
        <li ng-repeat="student in students">
          <strong>{{ student.name }}</strong> - Age: {{ student.age }} -
Course: {{ student.course }}
          <button ng-click="editStudentData(student)">Edit</button>
          <button ng-click="deleteStudent(student._id)">Delete</button>
        </li>
      </ul>
   </div>
    <div ng-if="editStudent">
      <h2>Edit Student</h2>
      <form ng-submit="updateStudent()">
        <label>Name:</label>
        <input type="text" ng-model="editStudent.name" required><br>
        <label>Age:</label>
        <input type="number" ng-model="editStudent.age" required><br>
        <label>Course:</label>
        <input type="text" ng-model="editStudent.course" required><br>
        <button type="submit">Update Student</button>
      </form>
   </div>
</body>
</html>
```

Practical 10:- To Create Student interface to stored and update the information.

1. Open visual studio code

2. Install some necessary Extensions that required to perform this practical

i. Live Server by ritwick dey

ii. Angular Snippets by john papa

iii. Html-CSS-JavaScript Support by ecmel

iv. Prettier by prettier

v. Npm

vi. Mongodb by mongodb

3. Install some necessary packages through npm (node package manager) to do this practical. This Packages can be installed using visual studio code terminal

i. npm init -y

ii. npm install angular

iii. npm install express mongoose body-parse cors

iv. npm install mongoose

v. npm install cors body-parser

4. Now Create a Directory in visual studio code terminal

mkdir (Directory Name)

5. After creating a Directory we need to change the directory

cd (Directory Name)

6. After changing there is a link of your directory open it using (ctrl+click)

7. After downloading the mongodb extension → click on the extension →Connect the server with the visual studio code using a connection string→ mongodb://localhost:27017 → connect

8. After connecting the localhost we can create a database where we insert a data that shown in our application

9. Creating database and inserting a data

i. Creating a databse

ii. Creating a collection

iii. Insert the data into the collection 10. Open visual studio code →open your folder→create a files ii. Index.html

i. Server.js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');
const app = express();
const port = 3000;
app.use(cors());
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));
// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/student')
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('Failed to connect to MongoDB', err));
const studentSchema = new mongoose.Schema({
  name: String,
  age: Number,
  course: String,
  year: String
}, { collection: 'stu' }); // Collection name is 'stu'
const Student = mongoose.model('Student', studentSchema);
// Get all students
app.get('/students', async (req, res) => {
  try {
    const students = await Student.find();
    res.json(students);
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch students' });
```

```
  }
});
        // Add a new student
app.post('/students', async (req, res) => {
  try {
    const student = new Student(req.body);
    await student.save();
    res.json(student);
  } catch (error) {
    res.status(500).json({ error: 'Failed to add student' });
  }
});
          // Update an existing student
app.put('/students/:id', async (req, res) => {
  try {
    const student = await Student.findByIdAndUpdate(req.params.id,
req.body, { new: true });
    res.json(student);
  } catch (error) {
    res.status(500).json({ error: 'Failed to update student' });
  }
});
        // Delete a student
app.delete('/students/:id', async (req, res) => {
  try {

    await Student.findByIdAndDelete(req.params.id);
    res.json({ message: 'Student deleted' });
  } catch (error) {
```

```
      res.status(500).json({ error: 'Failed to delete student' });
    }
});
          // Start the server
app.listen(port, () => {
   console.log(`Server running at http://localhost:${port}`);
});
```

11. Run this program using command

```
    Node server.js
```

ii. Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial
scale=1.0">
   <title>Student Manager</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
 </script>
   <script>
      const app = angular.module('studentApp', []);
  app.controller('studentController', function($scope, $http) {
         $scope.students = [];
         $scope.newStudent = {};
         $scope.editStudent = null;
         // Load students
```

```
    $scope.loadStudents = function() {
      $http.get('http://localhost:3000/students')
        .then(function(response) {
          $scope.students = response.data;
        });
    };
   // Add student
    $scope.addStudent = function() {
      $http.post('http://localhost:3000/students', $scope.newStudent)
        .then(function(response) {
          $scope.loadStudents();
          $scope.newStudent = {}; // Clear the form
        });
  //
 Edit student
    $scope.editStudentData = function(student) {
      $scope.editStudent = angular.copy(student);
    };
    // Update student
    $scope.updateStudent = function() {
$http.put(`http://localhost:3000/students/${$scope.editStudent._id}`,
$scope.editStudent)
        .then(function(response) {
          $scope.loadStudents();
          $scope.editStudent = null; // Clear the edit form
        });
    };
    // Delete student
    $scope.deleteStudent = function(id) {
```

```
            $http.delete(`http://localhost:3000/students/${id}`)
              .then(function(response) {
                $scope.loadStudents();
              });
          };
                // Initialize
          $scope.loadStudents();
        });
    </script>
</head>
<body ng-controller="studentController">
    <h1>Student Manager</h1>
    <div>
      <h2>Add New Student</h2>
      <form ng-submit="addStudent()">
        <label>Name:</label>
        <input type="text" ng-model="newStudent.name" required><br>
        <label>Age:</label>
        <input type="number" ng-model="newStudent.age" required><br>
        <label>Course:</label>
        <input type="text" ng-model="newStudent.course" required><br>
        <button type="submit">Add Student</button>
      </form>
    </div>
    <div>
      <h2>Students</h2>
      <ul>
        <li ng-repeat="student in students">
          <strong>{{ student.name }}</strong> - Age: {{ student.age }} -
```

```
Course: {{ student.course }}
            <button ng-click="editStudentData(student)">Edit</button>
            <button ng-click="deleteStudent(student._id)">Delete</button>
          </li>
        </ul>
    </div>
     <div ng-if="editStudent">
        <h2>Edit Student</h2>
        <form ng-submit="updateStudent()">
          <label>Name:</label>
          <input type="text" ng-model="editStudent.name" required><br>
          <label>Age:</label>
          <input type="number" ng-model="editStudent.age" required><br>
          <label>Course:</label>
          <input type="text" ng-model="editStudent.course" required><br>
          <button type="submit">Update Student</button>
        </form>
    </div>
</body>
</html>
```

Practical 11-:To display students information reports (All, Parametized)

1. Open visual studio code

2. Install some necessary Extensions that required to perform this practical

3. Install some necessary packages through npm (node package manager) to do this practical. This Packages can be installed using visual studio code terminal

4. Now Create a Directory in visual studio code terminal mkdir (Directory Name)

5. After creating a Directory we need to change the directory cd (Directory Name)

6. After changing there is a link of your directory open it using (ctrl+click)

7. After downloading the mongodb extension → click on the extension →Connect the server with the visual studio code using a connection string→ mongodb://localhost:27017 → connect

8. After connecting the localhost we can create a database where we insert a data that shown in our application
9. Creating database and inserting a data

iv. Creating a databse

v. Creating a collection

vi. Insert the data into the collection

vii. Sort data in parameterized using commands • 2 Way using AngularJS as frontend and MongoDB as backend
10. Open visual studio code →open your folder→create a files i. Server.js ii. Index.html

i. Server.js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');
const app = express();
const port = 3000;
app.use(cors());
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));
// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/student')
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('Failed to connect to MongoDB', err));
const studentSchema = new mongoose.Schema({
name: String,
age: Number,
course: String,
year: String
}, { collection: 'stu' }); // Collection name is 'stu'
const Student = mongoose.model('Student', studentSchema);
```

```
// Get all students with sorting
app.get('/students', async (req, res) => {
try {
// Default to ascending if no sortOrder provided
const sortOrder = req.query.sortOrder === 'desc' ? -1 : 1;
const students = await Student.find().sort({ age: sortOrder });
res.json(students);
} catch (error) {
res.status(500).json({ error: 'Failed to fetch students' });
}
});
// Add a new student
app.post('/students', async (req, res) => {
try {
const student = new Student(req.body);
await student.save();
res.json(student);
} catch (error) {
res.status(500).json({ error: 'Failed to add student' });
}
});
// Update an existing student
app.put('/students/:id', async (req, res) => {
try {
const student = await Student.findByIdAndUpdate(req.params.id, req.body, {
new: true });
res.json(student);
} catch (error) {
res.status(500).json({ error: 'Failed to update student' });
```

```
}
});
```



```
// Delete a student
app.delete('/students/:id', async (req, res) => {
try {
await Student.findByIdAndDelete(req.params.id);
res.json({ message: 'Student deleted' });
} catch (error) {
res.status(500).json({ error: 'Failed to delete student' });
}
});
// Start the server
app.listen(port, () => {
console.log(`Server running at http://localhost:${port}`);
});
```

11.Run this code using command

```
Node server.js
```

ii. Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Manager</title>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<script>
const app = angular.module('studentApp', []);
```

```
app.controller('studentController', function($scope, $http) {
$scope.students = [];
$scope.sortOrder = 'asc'; // Default sort order
WEB DEVELOPMENT TECHNOLOGY-III 50
// Load students with the specified sorting order
$scope.loadStudents = function() {
$http.get(`http://localhost:3000/students?sortOrder=${$scope.sortOrder}`)
.then(function(response) {
$scope.students = response.data;
});
};
// Change sort order and reload students
$scope.changeSortOrder = function(order) {
$scope.sortOrder = order;
$scope.loadStudents();
};
// Initialize
$scope.loadStudents();
});
</script>
</head>
<body ng-controller="studentController">
<h1>Student Manager</h1>
<div>
<h2>Sort By Age:</h2>
<button ng-click="changeSortOrder('asc')">Ascending</button>
<button ng-click="changeSortOrder('desc')">Descending</button>
</div>
<div>
```

```
<h2>Students</h2>
<ul>
<li ng-repeat="student in students">
<strong>{{ student.name }}</strong> - {{ student.age }} - {{ student.course }} - {{
student.year }}
</li>
</ul>
</div>
</body>
</html>
```

Practical: 12 ; : Implement a user interface where users can view, add, update, and delete tasks with MongoDB Database.

1. Open visual studio code

2. Install some necessary Extensions that required to perform this

3. Install some necessary packages through npm (node package manager) to do this practical. This Packages can be installed using visual studio code terminal

4. Now Create a Directory in visual studio code terminal

mkdir (Directory Name)

5. After creating a Directory we need to change the directory

cd (Directory Name)

6. After changing there is a link of your directory open it using (ctrl+click)

7. After downloading the mongodb extension → click on the extension →Connect the server with the visual studio code using a connection string→ mongodb://localhost:27017 → connect

8. After connecting the localhost we can create a database where we insert a data that shown in our application

9. Creating database and inserting a data

i. Server.js

```
 const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');
 const app = express();
const port = 3000;
app.use(cors());
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));
// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/student')
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('Failed to connect to MongoDB', err));
const studentSchema = new mongoose.Schema({
name: String,
age: Number,
course: String,
year: String
}, { collection: 'stu' }); // Collection name is 'stu'
const Student = mongoose.model('Student', studentSchema);
// Get all students with sorting
app.get('/students', async (req, res) => {
try {
// Default to ascending if no sortOrder provided
const sortOrder = req.query.sortOrder === 'desc' ? -1 : 1;
const students = await Student.find().sort({ age: sortOrder });
```

```
res.json(students);
} catch (error) {
res.status(500).json({ error: 'Failed to fetch students' });
}
});
// Add a new student
app.post('/students', async (req, res) => {
try {
```

```
const student = new Student(req.body);
await student.save();
res.json(student);
} catch (error) {
res.status(500).json({ error: 'Failed to add student' });
}
});
// Update an existing student
app.put('/students/:id', async (req, res) => {
try {
const student = await Student.findByIdAndUpdate(req.params.id, req.body,
{ new: true });
res.json(student);
} catch (error) {
res.status(500).json({ error: 'Failed to update student' });
}
});
// Delete a student
app.delete('/students/:id', async (req, res) => {
try {
```

```
await Student.findByIdAndDelete(req.params.id);
res.json({ message: 'Student deleted' });
} catch (error) {
res.status(500).json({ error: 'Failed to delete student' });
}
});
// Start the server
app.listen(port, () => {
console.log(`Server running at http://localhost:${port}`);
});
```

ii. Index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="studentApp">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Manager</title>
```



```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<script>
const app = angular.module('studentApp', []);
app.controller('studentController', function($scope, $http) {
$scope.students = [];
$scope.sortOrder = 'asc'; // Default sort order
 // Load students with the specified sorting order
$scope.loadStudents = function() {
$http.get(`http://localhost:3000/students?sortOrder=${$scope.sortOrder}`)
.then(function(response) {
$scope.students = response.data;
```

```
});
};
 // Change sort order and reload students
$scope.changeSortOrder = function(order) {
$scope.sortOrder = order;
$scope.loadStudents();
};
 // Initialize
$scope.loadStudents();
});
</script>
</head>
<body ng-controller="studentController">
<h1>Student Manager</h1>
<div>
<h2>Sort By Age:</h2>
<button ng-click="changeSortOrder('asc')">Ascending</button>
<button ng-click="changeSortOrder('desc')">Descending</button>
</div>
<div>
<h2>Students</h2>
 <ul>
<li ng-repeat="student in students">
<strong>{{ student.name }}</strong> - {{ student.age }} - {{ student.course }} -
{{ student.year }}
</li>
</ul>
</div>
</body>
```

</html>