

EXPERIMENT 4.2

AIM: Student Management System Using MongoDB and MVC Architecture

1. Create the following folders and files in vs-code:

student-mvc

```

.env
app.js
db.js
package.json

controllers
studentController.js

models
studentModel.js

routes
studentRoutes.js

views
└students
  index.ejs
  new.ejs
  edit.ejs
  show.ejs

public
  styles.css (optional)

```

2. **Create .env:** Create .env in student-mvc

```

MONGODB_URI=mongodb://127.0.0.1:27017
DB_NAME=studentsdb
PORT=3000

```

3. **Create db.js (Mongo connection helper):**

```

// db.js
const { MongoClient } = require("mongodb");
require("dotenv").config();

const uri = process.env.MONGODB_URI;
const dbName = process.env.DB_NAME;

let client;
let db;

async function connectDB() {
  if (db) return db;
  client = new MongoClient(uri);
  await client.connect();
}

```

```

db = client.db(dbName);
console.log("✅ MongoDB connected:", db.databaseName);
return db;
}

function getDB() {
  if (!db) throw new Error("DB not connected yet!");
  return db;
}

module.exports = { connectDB, getDB };

```

4. Create the Model (models/studentModel.js):

```

// models/studentModel.js
const { ObjectId } = require("mongodb");
const { getDB } = require("../db");

function collection() {
  return getDB().collection("students");
}

async function createStudent(data) {
  const doc = {
    rollNo: String(data.rollNo || "").trim(),
    name: String(data.name || "").trim(),
    age: data.age ? Number(data.age) : null,
    department: data.department?.trim() || null,
    email: data.email?.trim() || null,
    phone: data.phone?.trim() || null,
    createdAt: new Date(),
    updatedAt: new Date(),
  };
  const res = await collection().insertOne(doc);
  return { _id: res.insertedId, ...doc };
}

async function findAll(query = {}) {
  const filter = {};
  if (query.q) {
    filter.$or = [
      { name: { $regex: query.q, $options: "i" } },
      { rollNo: { $regex: query.q, $options: "i" } },
      { department: { $regex: query.q, $options: "i" } },
    ];
  }
  return collection().find(filter).sort({ createdAt: -1 }).toArray();
}

async function findById(id) {
  return collection().findOne({ _id: new ObjectId(id) });
}

```

```

        }

async function updateById(id, data) {
  const update = {
    $set: {
      rollNo: String(data.rollNo || "").trim(),
      name: String(data.name || "").trim(),
      age: data.age ? Number(data.age) : null,
      department: data.department?.trim() || null,
      email: data.email?.trim() || null,
      phone: data.phone?.trim() || null,
      updatedAt: new Date(),
    },
  };
  const res = await collection().findOneAndUpdate(
    { _id: new ObjectId(id) },
    update,
    { returnDocument: "after" }
  );
  return res.value;
}

async function deleteById(id) {
  const res = await collection().deleteOne({ _id: new ObjectId(id) });
  return res.deletedCount === 1;
}

module.exports = { createStudent, findAll, findById, updateById, deleteById };

```

5. Create the Controller (controllers/studentController.js)

```

// controllers/studentController.js
const Student = require("../models/studentModel");

exports.list = async (req, res) => {
  const students = await Student.findAll(req.query);
  res.render("students/index", { students, query: req.query.q || "" });
};

exports.newForm = (_req, res) => {
  res.render("students/new");
};

exports.create = async (req, res) => {
  try {
    if (!req.body.name || !req.body.rollNo) {
      return res.status(400).send("Name and Roll No are required");
    }
    await Student.createStudent(req.body);
    res.redirect("/students");
  } catch (e) {

```

```

    res.status(500).send(e.message);
}
};

exports.show = async (req, res) => {
  const student = await Student.findById(req.params.id);
  if (!student) return res.status(404).send("Not found");
  res.render("students/show", { student });
};

exports.editForm = async (req, res) => {
  const student = await Student.findById(req.params.id);
  if (!student) return res.status(404).send("Not found");
  res.render("students/edit", { student });
};

exports.update = async (req, res) => {
  try {
    await Student.findByIdAndUpdate(req.params.id, req.body);
    res.redirect("/students");
  } catch (e) {
    res.status(500).send(e.message);
  }
};

exports.remove = async (req, res) => {
  await Student.findByIdAndDelete(req.params.id);
  res.redirect("/students");
};

```

6. Create the Routes (routes/studentRoutes.js)

```

// routes/studentRoutes.js
const express = require("express");
const router = express.Router();
const ctrl = require("../controllers/studentController");

// Home -> /students
router.get("/", (_req, res) => res.redirect("/students"));

// HTML views
router.get("/students", ctrl.list);
router.get("/students/new", ctrl.newForm);
router.post("/students", ctrl.create);
router.get("/students/:id", ctrl.show);
router.get("/students/:id/edit", ctrl.editForm);
router.put("/students/:id", ctrl.update);
router.delete("/students/:id", ctrl.remove);

// (Optional) JSON API for Postman
/*

```

```

const Student = require("../models/studentModel");
router.get("/api/students", async (req, res) => res.json(await Student.findAll(req.query)));
router.post("/api/students", async (req, res) => res.status(201).json(await
Student.createStudent(req.body)));
router.get("/api/students/:id", async (req, res) => {
  const s = await Student.findById(req.params.id);
  if (!s) return res.status(404).json({ error: "Not found" });
  res.json(s);
});
router.put("/api/students/:id", async (req, res) => res.json(await
Student.updateById(req.params.id, req.body)));
router.delete("/api/students/:id", async (req, res) => { await Student.deleteById(req.params.id);
res.json({ success:true }); });
*/

```

module.exports = router;

7. Create app.js (server):

```

// app.js (stable, no layout plugin)
const express = require("express");
const path = require("path");
const methodOverride = require("method-override");
require("dotenv").config();

const { connectDB } = require("./db");
const routes = require("./routes/studentRoutes");

const app = express();

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(methodOverride("_method"));
app.use(express.static(path.join(__dirname, "public")));

connectDB()
.then(() => {
  app.use("/", routes);

  const port = process.env.PORT || 3000;
  app.listen(port, () => {
    console.log(`⚡ Server running at http://localhost:${port}`);
  });
})
.catch((err) => {
  console.error("Mongo connection error:", err);
  process.exit(1);
});

```

8. Create the Views (Bootstrap, self-contained)

- **views/students/index.ejs**

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Students</title>
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">
  <nav class="navbar navbar-dark bg-primary mb-4">
    <div class="container">
      <a class="navbar-brand" href="/students">Student Manager</a>
      <a href="/students/new" class="btn btn-light btn-sm">+ Add Student</a>
    </div>
  </nav>

  <main class="container">
    <div class="d-flex align-items-center mb-3">
      <h1 class="h3 mb-0">Students</h1>
      <a href="/students/new" class="btn btn-primary ms-auto">+ Add Student</a>
    </div>

    <form class="row g-2 mb-3" method="get" action="/students">
      <div class="col-sm-8">
        <input class="form-control" type="text" name="q" value="<% = query %>" placeholder="Search by name / roll / dept">
      </div>
      <div class="col-sm-4 d-grid d-sm-block">
        <button class="btn btn-outline-primary w-100">Search</button>
      </div>
    </form>

    <% if (students.length === 0) { %>
      <div class="alert alert-info">No students yet. Click "Add Student".</div>
    <% } else { %>
      <div class="card shadow-sm">
        <div class="table-responsive">
          <table class="table align-middle mb-0">
            <thead class="table-light">
              <tr>
                <th>Roll No</th><th>Name</th><th>Dept</th><th>Age</th><th>Email</th><th>Phone</th><th class="text-end">Actions</th>
              </tr>
            </thead>
            <tbody>
```

```

<% students.forEach(s => { %>
  <tr>
    <td><span class="badge text-bg-secondary"><%= s.rollNo %></span></td>
    <td><a href="/students/<%= s._id %>" class="link-primary fw-semibold"><%= s.name %></a></td>
      <td><%= s.department || '-' %></td>
      <td><%= s.age || '-' %></td>
      <td><%= s.email || '-' %></td>
      <td><%= s.phone || '-' %></td>
      <td class="text-end">
        <a class="btn btn-sm btn-outline-secondary" href="/students/<%= s._id %>/edit">Edit</a>
        <form method="post" action="/students/<%= s._id %>?_method=DELETE"
          class="d-inline" onsubmit="return confirm('Delete this student?')">
          <button class="btn btn-sm btn-outline-danger">Delete</button>
        </form>
      </td>
    </tr>
  <% } ) %>
</tbody>
</table>
</div>
</div>
<% } %>
</main>
</body>
</html>

```

• views/students/new.ejs

```

<!doctype html>
<html>
<head>
  <meta charset="utf-8"><title>Add Student</title>
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
    rel="stylesheet">
</head>
<body class="bg-light">
  <nav class="navbar navbar-dark bg-primary mb-4">
    <div class="container">
      <a class="navbar-brand" href="/students">Student Manager</a>
    </div>
  </nav>
  <main class="container">
    <div class="card shadow-sm">
      <div class="card-body">
        <h1 class="h4 mb-3">Add Student</h1>
        <form method="post" action="/students" class="row g-3">
          <div class="col-md-4"><label class="form-label">Roll No</label><input
            name="rollNo" class="form-control" required></div>

```

```

<div class="col-md-8"><label class="form-label">Name</label><input
name="name" class="form-control" required></div>
<div class="col-md-3"><label class="form-label">Age</label><input
type="number" name="age" class="form-control"></div>
<div class="col-md-3"><label class="form-label">Department</label><input
name="department" class="form-control"></div>
<div class="col-md-3"><label class="form-label">Email</label><input
type="email" name="email" class="form-control"></div>
<div class="col-md-3"><label class="form-label">Phone</label><input
name="phone" class="form-control"></div>
<div class="col-12 d-flex gap-2">
    <button class="btn btn-primary">Save</button>
    <a href="/students" class="btn btn-outline-secondary">Cancel</a>
</div>
</form>
</div>
</div>
</main>
</body>
</html>
views/students/edit.ejs
<!doctype html>
<html>
<head>
    <meta charset="utf-8"><title>Edit Student</title>
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
    <nav class="navbar navbar-dark bg-primary mb-4">
        <div class="container">
            <a class="navbar-brand" href="/students">Student Manager</a>
        </div>
    </nav>
    <main class="container">
        <div class="card shadow-sm">
            <div class="card-body">
                <h1 class="h4 mb-3">Edit Student</h1>
                <form method="post" action="/students/<%= student._id %>?_method=PUT"
class="row g-3">
                    <div class="col-md-4"><label class="form-label">Roll No</label><input
name="rollNo" class="form-control" value="<<%= student.rollNo %>" required></div>
                    <div class="col-md-8"><label class="form-label">Name</label><input
name="name" class="form-control" value="<<%= student.name %>" required></div>
                    <div class="col-md-3"><label class="form-label">Age</label><input
type="number" name="age" class="form-control" value="<<%= student.age || %>"/></div>
                    <div class="col-md-3"><label class="form-label">Department</label><input
name="department" class="form-control" value="<<%= student.department || %>"/></div>
                </form>
            </div>
        </div>
    </main>
</body>
</html>

```

```

    %>"></div>
    <div class="col-md-3"><label class="form-label">Email</label><input
type="email" name="email" class="form-control" value="<%= student.email || %
%"></div>
    <div class="col-md-3"><label class="form-label">Phone</label><input
name="phone" class="form-control" value="<%= student.phone || " %"></div>
    <div class="col-12 d-flex gap-2">
        <button class="btn btn-primary">Update</button>
        <a href="/students" class="btn btn-outline-secondary">Cancel</a>
    </div>
</form>
</div>
</div>
</main>
</body>
</html>

```

- **views/students/show.ejs**

```

<!doctype html>
<html>
<head>
    <meta charset="utf-8"><title><%= student.name %></title>
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
    <nav class="navbar navbar-dark bg-primary mb-4">
        <div class="container">
            <a class="navbar-brand" href="/students">Student Manager</a>
        </div>
    </nav>
    <main class="container">
        <div class="card shadow-sm">
            <div class="card-body">
                <h1 class="h4 mb-3"><%= student.name %></h1>
                <ul class="list-group">
                    <li class="list-group-item"><strong>Roll No:</strong> <%= student.rollNo
%></li>
                    <li class="list-group-item"><strong>Age:</strong> <%= student.age || '-' %></li>
                    <li class="list-group-item"><strong>Department:</strong> <%=
student.department || '-' %></li>
                    <li class="list-group-item"><strong>Email:</strong> <%= student.email || '-'
%></li>
                    <li class="list-group-item"><strong>Phone:</strong> <%= student.phone || '-' %
%></li>
                </ul>
                <div class="mt-3 d-flex gap-2">
                    <a href="/students/<%= student._id %>/edit" class="btn btn-outline-
secondary">Edit</a>

```

```

    <a href="/students" class="btn btn-primary">Back</a>
  </div>
</div>
</div>
</main>
</body>
</html>
(Optional) public/styles.css:
/* public/styles.css */
.card { border-radius: 14px; }

```

9. Run the app

In **Command Prompt** inside student-mvc:

node app.js

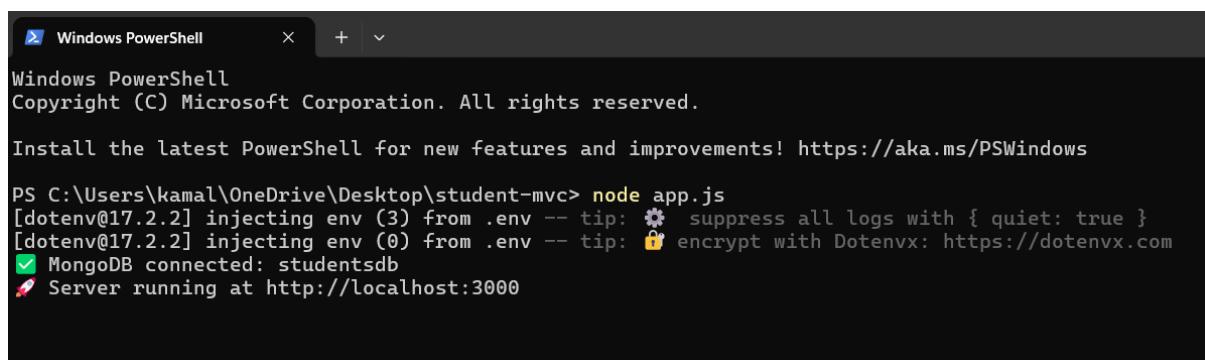
You should see:

-  MongoDB connected: studentsdb
-  Server running at http://localhost:3000

Open your browser:

- **List:** <http://localhost:3000/students>
- **Add new:** <http://localhost:3000/students/new>

MongoDB DB/collection auto-create on first insert. Check in **Compass** after adding a student.



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\kamal\OneDrive\Desktop\student-mvc> node app.js
[dotenv@17.2.2] injecting env (3) from .env -- tip: ⚙ suppress all logs with { quiet: true }
[dotenv@17.2.2] injecting env (0) from .env -- tip: 🔒 encrypt with Dotenvx: https://dotenvx.com
✔ MongoDB connected: studentsdb
🚀 Server running at http://localhost:3000

```

Figure 1: The cmd showing the server is running in mongoDB.

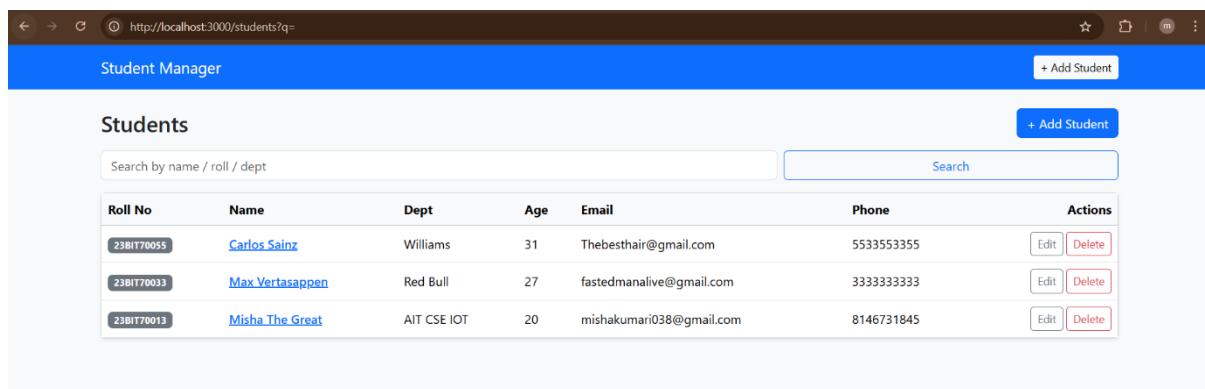


Figure 2 : Student Management System running on browser

Student Manager

+ Add Student

Students

+ Add Student

max vertasappen

Search

| Roll No | Name | Dept | Age | Email | Phone | Actions |
|------------|---------------------------------|----------|-----|--------------------------|------------|---|
| 23BIT70033 | Max Vertasappen | Red Bull | 27 | fastedmanalive@gmail.com | 3333333333 | Edit Delete |

Figure 3: running serach operation on the student management system