EXPERIMENT – 11

Student Management System Using MongoDB and MVC Architecture

Code Implementation: -

The Student Management System using

MongoDB and MVC architecture is a backend application that allows you to add, view, update, and delete student records.

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

const mongoose = require('mongoose');

const bodyParser = require('body-parser');


mongoose.connect('mongodb://127.0.0.1:27017/
studentdb', {

  useNewUrlParser: true,

  useUnifiedTopology: true

});
```

```
const StudentSchema = new mongoose.Schema({
  firstName: { type: String, required: true },
  lastName: { type: String },
  rollNo: { type: String, required: true, unique: true
},
  email: { type: String },
  dob: { type: Date },
  course: { type: String },
  createdAt: { type: Date, default: Date.now }
});

const Student = mongoose.model("Student",
StudentSchema);

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

```
// Controller functions
const studentController = {
  getAll: async (req, res) => {
    try {
      const students = await Student.find();
      res.json(students);
    } catch (err) {
      res.status(500).json({ error: err.message });
    }
  },

  create: async (req, res) => {
    try {
      const student = new Student(req.body);
      await student.save();
      res.status(201).json(student);
    } catch (err) {
      res.status(400).json({ error: err.message });
    }
  },

  getById: async (req, res) => {
    try {
      const student = await
Student.findById(req.params.id);
      if (!student) return res.status(404).json({
message: "Student not found" });
      res.json(student);
    } catch (err) {
      res.status(500).json({ error: err.message });
    }
  },

  update: async (req, res) => {
    try {
      const student = await
Student.findByIdAndUpdate(req.params.id,
req.body, { new: true });
      if (!student) return res.status(404).json({
message: "Student not found" });
      res.json(student);
    } catch (err) {
      res.status(400).json({ error: err.message });
    }
  },

  delete: async (req, res) => {
    try {
      const student = await
Student.findByIdAndDelete(req.params.id);
      if (!student) return res.status(404).json({
message: "Student not found" });
      res.json({ message: "Student deleted
successfully" });
```

```
    } catch (err) {
      res.status(500).json({ error: err.message });
    }
  }
};

// Routes
app.get('/students', studentController.getAll);
app.post('/students', studentController.create);
app.get('/students/:id',
studentController.getById);
app.put('/students/:id', studentController.update);
app.delete('/students/:id',
studentController.delete);

const PORT = 3000;
app.listen(PORT, () => console.log(`Server running
at http://localhost:${PORT}`));
```