**CRUD Operations in MongoDB using Mongoose and Node.js**

**Student Marks**:-

const mongoose = require("mongoose");

// Connect to MongoDB

mongoose.connect("mongodb://127.0.0.1:27017/schoolDB")

  .then(() => {

    console.log("Connected to MongoDB");

    return main(); // start program

  })

  .catch((err) => console.error("MongoDB connection error:", err));

// Define Schema

const studentSchema = new mongoose.Schema({

  name: String,

  age: Number,

  grade: String,

  createdAt: { type: Date, default: Date.now }

});

// Create Model

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

// --- CREATE ---

function createStudent(name, age, grade) {

  const student = new Student({ name, age, grade });

  return student.save()

    .then(result => {

      console.log("Student created:", result);

      return result;

    })

    .catch(err => console.error("Error creating student:", err));

}

// --- READ ---

function readStudents() {

  return Student.find()

    .then(students => {

      console.log("All Students:");

      students.forEach(s => {

        console.log(`- ${s.\_id}: ${s.name}, Age: ${s.age}, Grade: ${s.grade}`);

      });

      return students;

    })

    .catch(err => console.error("Error reading students:", err));

}

// --- UPDATE ---

function updateStudent(id, newData) {

  return Student.findByIdAndUpdate(id, newData, { new: true })

    .then(result => {

      if (result) console.log("Student updated:", result);

      else console.log("Student not found");

      return result;

    })

    .catch(err => console.error("Error updating student:", err));

}

// --- DELETE ---

function deleteStudent(id) {

  return Student.findByIdAndDelete(id)

    .then(result => {

      if (result) console.log("Student deleted:", result);

      else console.log("Student not found");

      return result;

    })

    .catch(err => console.error("Error deleting student:", err));

}

// Main Flow (Promise Chaining)

function main() {

  createStudent("John Doe", 18, "12th")

    .then(() => createStudent("Alice Smith", 17, "11th"))

    .then(() => readStudents())

    // Example update (uncomment and replace ID after first run)

    // .then(() => updateStudent("6522f123abcde4567890f123", { grade: "Graduated" }))

    // Example delete (uncomment and replace ID after first run)

    // .then(() => deleteStudent("6522f456def7890123456789"))

    .then(() => readStudents())

    .then(() => mongoose.connection.close())

    .then(() => console.log("Connection closed"))

    .catch(err => console.error("Error in main flow:", err));

}

**E- Commerce Cart**

const mongoose = require("mongoose");

const readline = require("readline");

// MongoDB Connection using Mongoose

mongoose

  .connect("mongodb://localhost:27017/ecommerce\_cart")

  .then(() => console.log("MongoDB Connected"))

  .catch((err) => console.log("Connection Error:", err));

//Schema

const cartSchema = new mongoose.Schema({

  itemId: { type: Number, required: true, unique: true },

  itemName: { type: String, required: true },

  price: { type: Number, required: true },

  quantity: { type: Number, required: true },

  total: { type: Number, required: true },

});

const Cart = mongoose.model("Cart", cartSchema);

//Input

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout,

});

function showMenu() {

  console.log("\n======= E-Commerce Cart Menu =======");

  console.log("1. Add Item to Cart");

  console.log("2. View Cart Items");

  console.log("3. Update Item");

  console.log("4. Delete Item");

  console.log("5. Exit");

  rl.question("Enter your choice: ", handleMenu);

}

// CRUD Operations

async function handleMenu(choice) {

  switch (choice) {

    case "1":

      await addItem();

      break;

    case "2":

      await viewCart();

      break;

    case "3":

      await updateItem();

      break;

    case "4":

      await deleteItem();

      break;

    case "5":

      console.log("Exiting...");

      mongoose.connection.close();

      rl.close();

      return;

    default:

      console.log("Invalid choice. Try again.");

  }

  showMenu();

}

// CREATE

async function addItem() {

  const item = {};

  item.itemId = await ask("Enter Item ID: ");

  item.itemName = await ask("Enter Item Name: ");

  item.price = parseFloat(await ask("Enter Price: "));

  item.quantity = parseInt(await ask("Enter Quantity: "));

  item.total = item.price \* item.quantity;

  try {

    const newItem = new Cart(item);

    await newItem.save();

    console.log("Item added to cart successfully!");

  } catch (err) {

    console.log("Error adding item:", err.message);

  }

}

// READ

async function viewCart() {

  const items = await Cart.find();

  console.log("\n--- Cart Items ---");

  if (items.length === 0) {

    console.log("Cart is empty.");

  } else {

    let grandTotal = 0;

    items.forEach((i) => {

      console.log(

        ID: ${i.itemId}, Name: ${i.itemName}, Price: ₹${i.price}, Qty: ${i.quantity}, Total: ₹${i.total}

      );

      grandTotal += i.total;

    });

    console.log(Grand Total: ₹${grandTotal});

  }

}

// UPDATE

async function updateItem() {

  const id = await ask("Enter Item ID to update: ");

  const item = await Cart.findOne({ itemId: id });

  if (!item) {

    console.log("Item not found.");

    return;

  }

  const name = await ask(`Enter new name (${item.itemName}): `);

  const price = await ask(`Enter new price (${item.price}): `);

  const quantity = await ask(`Enter new quantity (${item.quantity}): `);

  item.itemName = name || item.itemName;

  item.price = price ? parseFloat(price) : item.price;

  item.quantity = quantity ? parseInt(quantity) : item.quantity;

  item.total = item.price \* item.quantity;

  await item.save();

  console.log("Item updated successfully!");

}

// DELETE

async function deleteItem() {

  const id = await ask("Enter Item ID to delete: ");

  const result = await Cart.findOneAndDelete({ itemId: id });

  if (result) console.log("Item deleted successfully!");

  else console.log("Item not found.");

}

function ask(question) {

  return new Promise((resolve) => rl.question(question, resolve));

}

showMenu();

**Employee PayRoll**

const mongoose = require("mongoose");

const readline = require("readline");

// Connect DB

mongoose

  .connect("mongodb://localhost:27017/employee\_payroll")

  .then(() => console.log("MongoDB Connected"))

  .catch((err) => console.log("Error:", err.message));

// Schema

const employeeSchema = new mongoose.Schema({

  empId: { type: Number, required: true, unique: true },

  name: { type: String, required: true },

  department: { type: String, required: true },

  salary: { type: Number, required: true },

});

const Employee = mongoose.model("Employee", employeeSchema);

// Input setup

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout,

});

// Menu

function showMenu() {

  console.log("\n=== Employee Payroll ===");

  console.log("1. Insert");

  console.log("2. View");

  console.log("3. Update");

  console.log("4. Delete");

  console.log("5. Exit");

  rl.question("Enter choice: ", handleMenu);

}

// Handle menu choice

async function handleMenu(choice) {

  switch (choice) {

    case "1":

      await insertEmployee();

      break;

    case "2":

      await viewEmployees();

      break;

    case "3":

      await updateEmployee();

      break;

    case "4":

      await deleteEmployee();

      break;

    case "5":

      console.log("Bye!");

      mongoose.connection.close();

      rl.close();

      return;

    default:

      console.log("Invalid choice.");

  }

  showMenu();

}

// Insert

async function insertEmployee() {

  const emp = {};

  emp.empId = await ask("ID: ");

  emp.name = await ask("Name: ");

  emp.department = await ask("Dept: ");

  emp.salary = await ask("Salary: ");

  try {

    await new Employee(emp).save();

    console.log("Added!");

  } catch (err) {

    console.log("Error:", err.message);

  }

}

// View

async function viewEmployees() {

  const employees = await Employee.find();

  console.log("\n--- Employee List ---");

  if (employees.length === 0) console.log("No data.");

  else {

    employees.forEach((e) => {

      console.log(

        `ID: ${e.empId}, Name: ${e.name}, Dept: ${e.department}, Salary: ₹${e.salary}`

      );

    });

  }

}

// Update

async function updateEmployee() {

  const id = await ask("Enter ID: ");

  const emp = await Employee.findOne({ empId: id });

  if (!emp) return console.log("Not found.");

  const name = await ask(`New name (${emp.name}): `);

  const dept = await ask(`New dept (${emp.department}): `);

  const salary = await ask(`New salary (${emp.salary}): `);

  emp.name = name || emp.name;

  emp.department = dept || emp.department;

  emp.salary = salary || emp.salary;

  await emp.save();

  console.log("Updated!");

}

// Delete

async function deleteEmployee() {

  const id = await ask("Enter ID: ");

  const res = await Employee.findOneAndDelete({ empId: id });

  console.log(res ? "Deleted!" : "Not found.");

}

function ask(q) {

  return new Promise((resolve) => rl.question(q, resolve));

}

showMenu();

Hospital Management System

const mongoose = require("mongoose");

// Connect to MongoDB

mongoose.connect("mongodb://127.0.0.1:27017/hospitalDB")

.then(() => {

console.log("Connected to MongoDB");

return main(); // Start program

})

.catch((err) => console.error("MongoDB connection error:", err));

// Define Schema

const patientSchema = new mongoose.Schema({

name: String,

age: Number,

gender: String,

disease: String,

doctorAssigned: String,

admittedDate: { type: Date, default: Date.now },

discharged: { type: Boolean, default: false }

});

// Create Model

const Patient = mongoose.model("Patient", patientSchema);

// --- CREATE ---

function createPatient(name, age, gender, disease, doctorAssigned) {

const patient = new Patient({ name, age, gender, disease, doctorAssigned });

return patient.save()

.then(result => {

console.log("🩺 Patient added:", result);

return result;

})

.catch(err => console.error("Error creating patient:", err));

}

// --- READ ---

function readPatients() {

return Patient.find()

.then(patients => {

console.log("\n🏥 All Patients:");

patients.forEach(p => {

console.log(`- ${p.\_id}: ${p.name}, Age: ${p.age}, Disease: ${p.disease}, Doctor: ${p.doctorAssigned}, Discharged: ${p.discharged}`);

});

return patients;

})

.catch(err => console.error("Error reading patients:", err));

}

// --- UPDATE ---

function updatePatient(id, newData) {

return Patient.findByIdAndUpdate(id, newData, { new: true })

.then(result => {

if (result) console.log("✅ Patient updated:", result);

else console.log("❌ Patient not found");

return result;

})

.catch(err => console.error("Error updating patient:", err));

}

// --- DELETE ---

function deletePatient(id) {

return Patient.findByIdAndDelete(id)

.then(result => {

if (result) console.log("🗑️ Patient deleted:", result);

else console.log("❌ Patient not found");

return result;

})

.catch(err => console.error("Error deleting patient:", err));

}

// --- MAIN FLOW ---

function main() {

createPatient("John David", 45, "Male", "Fever", "Dr. Sharma")

.then(() => createPatient("Priya R", 30, "Female", "Migraine", "Dr. Patel"))

.then(() => readPatients())

// Example update (uncomment and replace ID after first run)

// .then(() => updatePatient("6522f123abcde4567890f123", { discharged: true }))

// Example delete (uncomment and replace ID after first run)

// .then(() => deletePatient("6522f456def7890123456789"))

.then(() => readPatients())

.then(() => mongoose.connection.close())

.then(() => console.log("\n Connection closed"))

.catch(err => console.error("Error in main flow:", err));

}

Train Booking Management System

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017/trainBookingDB")

.then(() => {

console.log("🚂 Connected to MongoDB (Train Booking DB)");

return main(); // Start program

})

.catch((err) => console.error("MongoDB connection error:", err));

const bookingSchema = new mongoose.Schema({

passengerName: String,

age: Number,

gender: String,

trainNumber: String,

trainName: String,

source: String,

destination: String,

travelDate: Date,

seatNumber: String,

classType: String, // e.g., Sleeper, AC, General

bookingStatus: { type: String, default: "Booked" }, // Booked / Cancelled

bookingDate: { type: Date, default: Date.now }

});

const Booking = mongoose.model("Booking", bookingSchema);

function createBooking(

passengerName,

age,

gender,

trainNumber,

trainName,

source,

destination,

travelDate,

seatNumber,

classType

) {

const booking = new Booking({

passengerName,

age,

gender,

trainNumber,

trainName,

source,

destination,

travelDate,

seatNumber,

classType

});

return booking

.save()

.then((result) => {

console.log("🎟️ Booking added:", result);

return result;

})

.catch((err) => console.error("Error creating booking:", err));

}

function readBookings() {

return Booking.find()

.then((bookings) => {

console.log("\n🚆 All Bookings:");

bookings.forEach((b) => {

console.log(

`- ${b.\_id}: ${b.passengerName} | Train: ${b.trainName} (${b.trainNumber}) | ${b.source} ➝ ${b.destination} | Date: ${b.travelDate.toDateString()} | Seat: ${b.seatNumber} | Class: ${b.classType} | Status: ${b.bookingStatus}`

);

});

return bookings;

})

.catch((err) => console.error("Error reading bookings:", err));

}

function updateBooking(id, newData) {

return Booking.findByIdAndUpdate(id, newData, { new: true })

.then((result) => {

if (result) console.log("✅ Booking updated:", result);

else console.log("❌ Booking not found");

return result;

})

.catch((err) => console.error("Error updating booking:", err));

}

function deleteBooking(id) {

return Booking.findByIdAndDelete(id)

.then((result) => {

if (result) console.log("🗑️ Booking deleted:", result);

else console.log("❌ Booking not found");

return result;

})

.catch((err) => console.error("Error deleting booking:", err));

}

function main() {

createBooking(

"Rahul Kumar",

28,

"Male",

"12678",

"Chennai Express",

"Chennai",

"Delhi",

new Date("2025-10-15"),

"S3-45",

"Sleeper"

)

.then(() =>

createBooking(

"Anjali Sharma",

35,

"Female",

"12345",

"Rajdhani Express",

"Mumbai",

"Kolkata",

new Date("2025-10-20"),

"A1-12",

"AC"

)

)

.then(() => readBookings())

.then(() => readBookings())

.then(() => mongoose.connection.close())

.then(() => console.log("\n🔒 Connection closed"))

.catch((err) => console.error("Error in main flow:", err));

}