**Module 4: Assignment 2: Creating a NoSQL database in MongoDB Atlas**

Saurabh Verma

Arizona State University

IFT 598: Middleware Programming & Database Security

Professor. Dinesh Sthapit

October 23, 2022

STEP 1:

A screenshot of a computer

Description automatically generated with medium confidence

Step 2:

Created the model folder and loanModel.js file:

A screenshot of a computer

Description automatically generated with medium confidence

**Code:**

const mongoose = require('mongoose');

const loanSchema = new mongoose.Schema(

  {

    customerName: {

        type: String,

        required: [true, 'A customer must have a name'],

        trim: true,

        maxlength: [40, 'A customer name must have less or equal then 40 characters'],

        minlength: [10, 'A customer name must have more or equal then 10 characters']

    },

    phoneNumber:{

        type: Number,

        required: [true, 'A customer must have a phone number'],

        unique: true,

        validate: {

            validator: function(value){

                return value.toString().length === 10;

            }

        }

    },

    address: {

        type: String,

        required: [true, 'A customer must have an address'],

        trim: true,

        maxlength: [70, 'An address  must have less or equal then 70 characters'],

        minlength: [10, 'A from  must have more or equal then 10 characters']

    },

    loanAmount: {

        type: Number,

        required: [true, 'A loan should have loan amount']

    },

    interest: {

        type: Number,

        required: [true, 'A loan should have loan interest'],

        validate: {

            validator: function(value){

                return value.toString().length < 4;

            }

        }

    },

    loanTermYears: {

        type: Number,

        required: [true, 'A loan should have loan term years'],

    },

    loanType: {

        type: String,

        required: [true, 'A loan must have a type'],

        trim: true,

        maxlength: [20, 'A loan type must have less or equal then 20 characters'],

        minlength: [10, 'A loan type must have more or equal then 10 characters']

    },

    description: {

        type: String,

        required: [true, 'A description must have a loan description'],

        trim: true,

        maxlength: [100, 'A description  must have less or equal then 100 characters'],

        minlength: [5, 'A description  must have more or equal then 5 characters']

    },

    createdDate: {

        type: Date,

        default: Date.now

    },

    insertedDate: {

        type: Date,

        default: Date.now

    }

});

const Loan = mongoose.model('Loan', loanSchema);

module.exports = Loan;

Step 3: Setup the server.js

A screenshot of a computer

Description automatically generated with medium confidence

**Code:**

const dotenv = require('dotenv');

dotenv.config({ path: './config.env' });

const app = require('./app');

//new code

const mongoose = require('mongoose');

const MONGO\_DATA\_BASE = process.env.DATABASE.replace('<password>', process.env.DB\_PASSWORD);

//Connect to database

mongoose.connect(MONGO\_DATA\_BASE,

  //connection recipie

  {

    useNewUrlParser: true,

    useunifiedTopology: true,

    //useCreateIndex: true

  }).then(con=>{

    console.log(con.connection);// log connection properties

    console.log(`The Database connection was successful with ${process.env.DATABASE}`);// log connection properties

  });

const port = process.env.PORT || 3000;

app.listen(port, () => {

  console.log(`App running on port ${port}...`);

});

Step 4:

Text

Description automatically generated

**Code:**

class APIFeatures {

    constructor(query, queryString) {

      this.query = query;

      this.queryString = queryString;

    }

    filter() {

      const queryObj = { ...this.queryString };

      const excludedFields = ['page', 'sort', 'limit', 'fields'];

      excludedFields.forEach(el => delete queryObj[el]);

      // 1B) Advanced filtering

      let queryStr = JSON.stringify(queryObj);

      queryStr = queryStr.replace(/\b(gte|gt|lte|lt)\b/g, match => `$${match}`);

      this.query = this.query.find(JSON.parse(queryStr));

      return this;

    }

    sort() {

      if (this.queryString.sort) {

        const sortBy = this.queryString.sort.split(',').join(' ');

        this.query = this.query.sort(sortBy);

      } else {

        this.query = this.query.sort('-createdAt');

      }

      return this;

    }

    limitFields() {

      if (this.queryString.fields) {

        const fields = this.queryString.fields.split(',').join(' ');

        this.query = this.query.select(fields);

      } else {

        this.query = this.query.select('-\_\_v');

      }

      return this;

    }

    paginate() {

      const page = this.queryString.page \* 1 || 1;

      const limit = this.queryString.limit \* 1 || 100;

      const skip = (page - 1) \* limit;

      this.query = this.query.skip(skip).limit(limit);

      return this;

    }

  }

  module.exports = APIFeatures;

Step 5:

A screenshot of a computer

Description automatically generated with medium confidence

**Code:**

const Loan = require('./../model/loanModel');

const APIFeatures = require('./../dataBaseManager/loanDbContext');

// Add a new Loan

exports.createLoan = async  (req, res) => {

    try {

      const newLoan = await Loan.create(req.body);

      res.status(201).json({

        status: 'success',

        data: {

          course: newLoan

        }

      });

    } catch (err) {

      res.status(400).json({

        status: 'fail',

        message: err

      });

    }

  };

// Get all loans

exports.getAllLoan =   async (req, res) => {

  try {

    // EXECUTE QUERY

    const features = new APIFeatures(Loan.find(), req.query)

      .filter()

      .sort()

      .limitFields()

      .paginate();

    const loans = await features.query;

    // SEND RESPONSE

    res.status(200).json({

      status: 'success',

      results: loans.length,

      data: {

        loans

      }

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

// Get a specific loan by ID

exports.getLoan = async (req, res) => {

  try {

    const loan = await Loan.findById(req.params.id);

    // Course.findOne({ \_id: req.params.id })

    res.status(200).json({

      status: 'success',

      data: {

        loan

      }

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

// Change an existing loan

exports.updateLoan = async (req, res) => {

  try {

    const loan = await Loan.findByIdAndUpdate(req.params.id, req.body, {

      new: true,

      runValidators: true

    });

    res.status(200).json({

      status: 'success',

      data: {

        loan

      }

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

// Delete a loan

exports.deleteLoan = async (req, res) => {

  try {

    await Loan.findByIdAndDelete(req.params.id);

    res.status(204).json({

      status: 'success',

      data: null

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

Step 6:

A screenshot of a computer

Description automatically generated with medium confidence

**Code:**

const express = require('express');

const loanController = require('../controllers/loanController');

const router = express.Router();

router

  .route('/')

  .get(loanController.getAllLoan)

  .post(loanController.createLoan);

router

  .route('/:id')

  .get(loanController.getLoan)

  .put(loanController.updateLoan)

  .delete(loanController.deleteLoan);

module.exports = router;

Modifying app.js file:

A screenshot of a computer

Description automatically generated

**Code:**

const express = require('express');

const morgan = require('morgan');

const loanRouter = require('./routes/loanRoutes');

const app = express();

// 1) MIDDLEWARES

if (process.env.NODE\_ENV === 'development') {

  app.use(morgan('dev'));

}

app.use(express.json());

app.use(express.static(`${\_\_dirname}/public`));

app.use((req, res, next) => {

  console.log('Hello from the middleware 👋');

  next();

});

app.use((req, res, next) => {

  req.requestTime = new Date().toISOString();

  next();

});

// 3) ROUTES

app.use('/api/v1/loan', loanRouter);

module.exports = app;

Step 7: Testing:

Successfully connected to MongoDB after running node server.js:

Text

Description automatically generated

Checking records on MongoDB:

Graphical user interface, text, email

Description automatically generated

Step 8:

1. Adding new loan

Graphical user interface, text, application

Description automatically generated

Screenshot of Postman requests:

Text

Description automatically generated

Step 10: Terminal Output:

The Database connection was successful with mongodb+srv://sverma67:<password>@cluster0.htmb1ie.mongodb.net/test

Hello from the middleware 👋

POST /api/v1/loan 400 18.173 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan 400 4.409 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 2.108 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 2.139 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 2.959 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 2.143 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 4.231 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 1.407 ms - 2258

Hello from the middleware 👋

POST /api/v1/loan/ 400 3.123 ms – 2258

**References**

[Module 4 Assignment 2 Canvas page](https://canvas.asu.edu/courses/132658/assignments/3526667?module_item_id=9146146)