

1. Build a RESTful API using MongoDB.

Index.js -

```
// Import express
let express = require("express");
// Import Body parser
let bodyParser = require("body-parser");
// Import Mongoose
let mongoose = require("mongoose");
// Initialise the app
let app = express();

// Import routes
let apiRoutes = require("./api-routes");
// Configure bodyparser to handle post requests
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);
app.use(bodyParser.json());
// Connect to Mongoose and set connection variable
mongoose.connect("mongodb://localhost/student", { useNewUrlParser: true,
family: 4, });
var db = mongoose.connection;

// Added check for DB connection
if (!db) console.log("Error connecting db");
else console.log("Db connected successfully");

// Setup server port
var port = process.env.PORT || 8080;

// Send message for default URL
app.get("/", (req, res) => res.send("Hello World with Express"));

// Use Api routes in the App
```

```
app.use("/api", apiRoutes);
// Launch app to listen to specific port
app.listen(port, function () {
  console.log("Running RestHub on port " + port);
});
```

contactModel.js -

```
var mongoose = require("mongoose"); // Setup schema
var contactSchema = mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
  },
  gender: String,
  phone: String,
  create_date: {
    type: Date,
    default: Date.now,
  },
}); // Export Contact model
var Contact = (module.exports = mongoose.model("contact", contactSchema));
module.exports.get = function (callback, limit) {
  Contact.find(callback).limit(limit);
};
```

contactController.js -

```
// contactController.js
// Import contact model
Contact = require("./contactModel"); // Handle index actions
exports.index = function (req, res) {
  Contact.get(function (err, contacts) {
    if (err) {
      res.json({
```

```

        status: "error",
        message: err,
    });
}
res.json({
    status: "success",
    message: "Contacts retrieved successfully",
    data: contacts,
});
});
}; // Handle create contact actions
exports.new = function (req, res) {
    var contact = new Contact();
    contact.name = req.body.name ? req.body.name : contact.name;
    contact.gender = req.body.gender;
    contact.email = req.body.email;
    contact.phone = req.body.phone; // save the contact and check for errors
    contact.save(function (err) {
        if (err) res.json(err);
        res.json({
            message: "New contact created!",
            data: contact,
        });
    });
}; // Handle view contact info
exports.view = function (req, res) {
    Contact.findById(req.params.contact_id, function (err, contact) {
        if (err) res.send(err);
        res.json({
            message: "Contact details loading..",
            data: contact,
        });
    });
}; // Handle update contact info
exports.update = function (req, res) {
    Contact.findById(req.params.contact_id, function (err, contact) {
        if (err) res.send(err);
    });
};

```

```

    contact.name = req.body.name ? req.body.name : contact.name;
    contact.gender = req.body.gender;
    contact.email = req.body.email;
    contact.phone = req.body.phone; // save the contact and check for errors
    contact.save(function (err) {
        if (err) res.json(err);
        res.json({
            message: "Contact Info updated",
            data: contact,
        });
    });
});
}; // Handle delete contact
exports.delete = function (req, res) {
    Contact.remove(
        {
            _id: req.params.contact_id,
        },
        function (err, contact) {
            if (err) res.send(err);
            res.json({
                status: "success",
                message: "Contact deleted",
            });
        }
    );
};

```

api-routes.js-

```

// api-routes.js// Initialize express router
let router = require("express").Router(); // Set default API response
router.get("/", function (req, res) {
    res.json({
        status: "API Its Working",
        message: "Welcome to RESTHub crafted with love!",
    });
}); // Import contact controller

```

```

var contactController = require("./contactController"); // Contact routes
router
  .route("/contacts")
  .get(contactController.index)
  .post(contactController.new);
router
  .route("/contacts/:contact_id")
  .get(contactController.view)
  .patch(contactController.update)
  .put(contactController.update)
  .delete(contactController.delete); // Export API routes
module.exports = router;

```

a. GET -

Query	Headers ²	Auth	Body ¹	Tests	Pre Run	Status: 200 OK	Size: 367 Bytes	Time: 36 ms
<div> <div>GET</div> <div>http://localhost:8080/api/contacts</div> <div>Send</div> </div>								
<div> <div>JSON</div> <div>XML</div> <div>Text</div> <div>Form</div> <div><u>Form-encode</u></div> <div>GraphQL</div> <div>Binary</div> </div>						<div> <div>Response</div> <div>Headers ⁶</div> <div>Cookies</div> <div>Results</div> <div>Docs</div> </div>		
<div>Form Encoded</div> <div> <input checked="" type="checkbox"/> name Mikil <input checked="" type="checkbox"/> email mikil@gmail.com <input checked="" type="checkbox"/> gender male <input checked="" type="checkbox"/> phone 1234567890 <input type="checkbox"/> name value </div>						<pre> 1 { 2 "status": "success", 3 "message": "Contacts retrieved successfully", 4 "data": [5 { 6 "_id": "6434d64abada02c44721e18b", 7 "create_date": "2023-04-11T03:38:50.935Z", 8 "name": "a", 9 "gender": "m", 10 "email": "b", 11 "phone": "1", 12 "__v": 0 13 }, 14 { 15 "_id": "6434d69ebada02c44721e18f", 16 "create_date": "2023-04-11T03:40:14.440Z", 17 "name": "Mikil", 18 "gender": "male", 19 "email": "mikil@gmail.com", 20 "phone": "1234567890", 21 "__v": 0 22 } 23] 24 } </pre>		

b. PUT -

POST ▾	http://localhost:8080/api/contacts	Send				
Query	Headers ²	Auth	Body ¹	Tests	Pre Run	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary
Form Encoded						
<input checked="" type="checkbox"/>	name	Mikil				
<input checked="" type="checkbox"/>	email	mikil@gmail.com				
<input checked="" type="checkbox"/>	gender	male				
<input checked="" type="checkbox"/>	phone	1234567890				
<input type="checkbox"/>	name	value				

Status: 200 OK Size: 203 Bytes Time: 5 ms

Response	Headers ⁶	Cookies	Results	Docs
<pre>1 { 2 "message": "New contact created!", 3 "data": { 4 "_id": "6434d69ebada02c44721e18f", 5 "create_date": "2023-04-11T03:40:14.440Z", 6 "name": "Mikil", 7 "gender": "male", 8 "email": "mikil@gmail.com", 9 "phone": "1234567890", 10 "_v": 0 11 } 12 }</pre>				

c. PUSH -

PUT ▾	http://localhost:8080/api/contacts/6434d69ebada02c44721e18f	Send				
Query	Headers ²	Auth	Body ¹	Tests	Pre Run	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary
Form Encoded						
<input checked="" type="checkbox"/>	name	Mikil Lalwani				
<input checked="" type="checkbox"/>	email	mikil.lalwani@gmail.com				
<input checked="" type="checkbox"/>	gender	male				
<input checked="" type="checkbox"/>	phone	9876543210				
<input type="checkbox"/>	name	value				

Status: 200 OK Size: 219 Bytes Time: 15 ms

Response	Headers ⁶	Cookies	Results	Docs
<pre>1 { 2 "message": "Contact Info updated", 3 "data": { 4 "_id": "6434d69ebada02c44721e18f", 5 "create_date": "2023-04-11T03:40:14.440Z", 6 "name": "Mikil Lalwani", 7 "gender": "male", 8 "email": "mikil.lalwani@gmail.com", 9 "phone": "9876543210", 10 "_v": 0 11 } 12 }</pre>				

d. DELETE -

DELETE ▾	http://localhost:8080/api/contacts/6434d64abada02c44721e18f	Send				
Query	Headers ²	Auth	Body	Tests	Pre Run	
JSON	XML	Text	Form	Form-encode	GraphQL	Binary

Status: 200 OK Size: 48 Bytes Time: 7 ms

Response	Headers ⁶	Cookies	Results
<pre>1 { 2 "status": "success", 3 "message": "Contact deleted"</pre>			

2. Create a new database to store student details and perform the following on the database.

- 1. Create a database.**
- 2. Create a collection.**
- 3. Create a document and insert one document and insert many documents at once.**
- 4. Search with conditions.**
- 5. Change the roll no of a student**
- 6. Delete one or many documents**

1. Create a database.

```
test> show dbs
admin    40.00 KiB
config  12.00 KiB
local    40.00 KiB
test> use student
switched to db student
```

2. Create a collection.

```
student> db.createCollection("details")
{ ok: 1 }
```

3. Insert one document.

```
student> db.details.insertOne({first_name:"Mikil", last_name:"Lalwani", roll_no:37, divsion:"D15B"})
{
  acknowledged: true,
  insertedId: ObjectId("6433f33b40a86808b180202f")
}
```

4. Insert many documents at once.

```
student> db.details.insertMany([
  {first_name:"A", last_name:"a", roll_no:1, divsion:"D15B"},
  {first_name:"B", last_name:"b", roll_no:2, divsion:"D15B"},
  {first_name:"C", last_name:"c", roll_no:1, divsion:"D15A"},
  {first_name:"D", last_name:"d", roll_no:3, divsion:"D15A"}
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6433f41040a86808b1802030"),
    '1': ObjectId("6433f41040a86808b1802031"),
    '2': ObjectId("6433f41040a86808b1802032"),
    '3': ObjectId("6433f41040a86808b1802033")
  }
}
```

5. Find all documents.

```
student> db.details.find()
[
  { _id: ObjectId("6433f2f440a86808b180202e") },
  {
    _id: ObjectId("6433f33b40a86808b180202f"),
    first_name: 'Mikil',
    last_name: 'Lalwani',
    roll_no: 37,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802030"),
    first_name: 'A',
    last_name: 'a',
    roll_no: 1,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802031"),
    first_name: 'B',
    last_name: 'b',
    roll_no: 2,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802032"),
    first_name: 'C',
    last_name: 'c',
    roll_no: 1,
    divsion: 'D15A'
  },
  {
    _id: ObjectId("6433f41040a86808b1802033"),
    first_name: 'D',
    last_name: 'd',
    roll_no: 3,
    divsion: 'D15A'
  }
]
```


6. Find documents using conditions.

```
student> db.details.find({roll_no:1})
[
  {
    _id: ObjectId("6433f41040a86808b1802030"),
    first_name: 'A',
    last_name: 'a',
    roll_no: 1,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802032"),
    first_name: 'C',
    last_name: 'c',
    roll_no: 1,
    divsion: 'D15A'
  }
]
```

```
student> db.details.find({roll_no:1, divsion:"D15B"})
[
  {
    _id: ObjectId("6433f41040a86808b1802030"),
    first_name: 'A',
    last_name: 'a',
    roll_no: 1,
    divsion: 'D15B'
  }
]
```

7. Delete one document.

```
student> db.details.deleteOne({roll_no:3})
{ acknowledged: true, deletedCount: 1 }
```

```

student> db.details.find()
[
  { _id: ObjectId("6433f2f440a86808b180202e") },
  {
    _id: ObjectId("6433f33b40a86808b180202f"),
    first_name: 'Mikil',
    last_name: 'Lalwani',
    roll_no: 37,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802030"),
    first_name: 'A',
    last_name: 'a',
    roll_no: 1,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802031"),
    first_name: 'B',
    last_name: 'b',
    roll_no: 2,
    divsion: 'D15B'
  },
  {
    _id: ObjectId("6433f41040a86808b1802032"),
    first_name: 'C',
    last_name: 'c',
    roll_no: 1,
    divsion: 'D15A'
  }
]

```

8. Delete many documents at once.

```

student> db.details.deleteMany({divsion: 'D15B'})
{ acknowledged: true, deletedCount: 2 }
student> db.details.find()
[
  {
    _id: ObjectId("6433f41040a86808b1802032"),
    first_name: 'C',
    last_name: 'c',
    roll_no: 1,
    divsion: 'D15A'
  }
]

```