



**Technological Institute of the Philippines
Manila Campus**

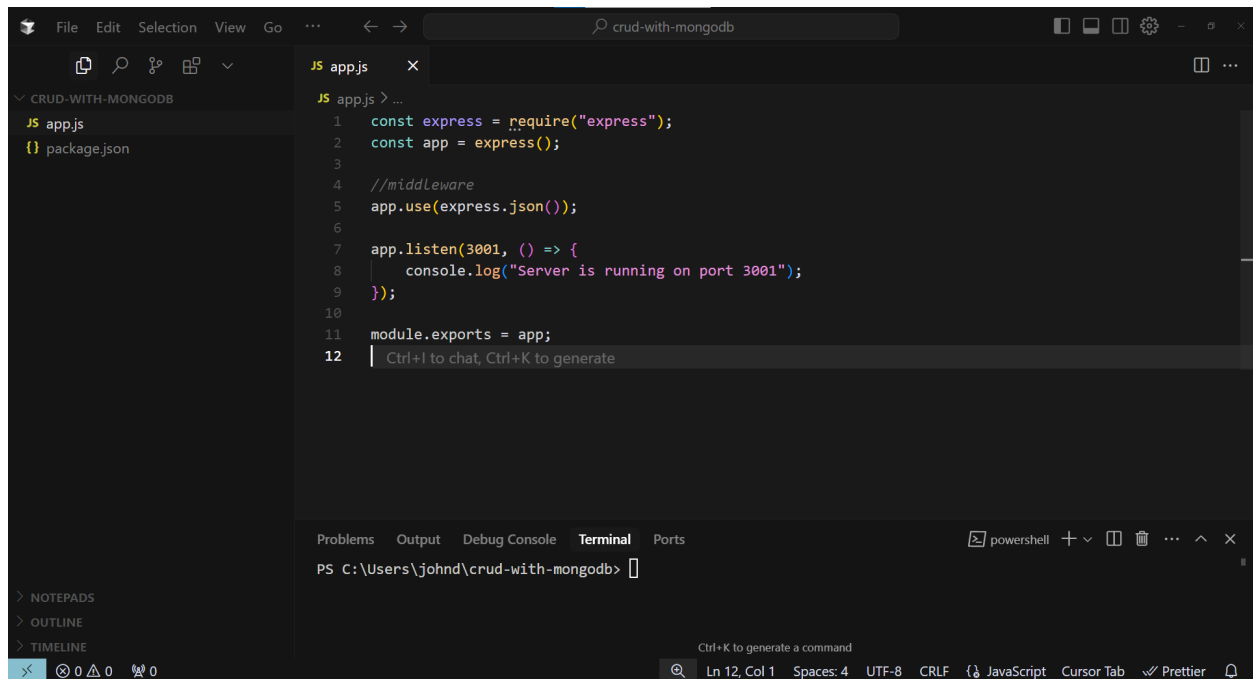
***College of Computer Studies
CITE 006-IT32S1 - Application Development and Emerging Technologies
Bachelor of Science in Information Technology***

***Laboratory Activity:
Build-a-CRUD-App-with-Node-js-and-MongoDB-AppSignal-Blog-1***

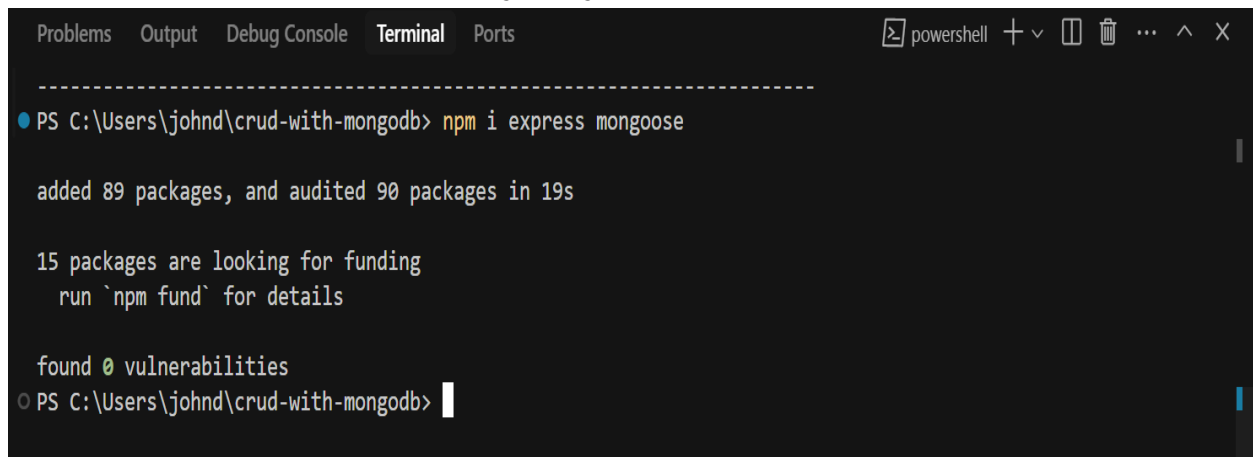
Submitted by:
Doctor, John Michael

Submitted to:
Prof. FRANCIS CARABUENA

Node.js Project Set-up



Install Dependencies for Your Node.js Project



The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the project structure: `crud-with-mongodb` > `node_modules`, `app.js`, `package-lock.json`, and `package.json`. The main editor area shows the `package.json` file with the following content:

```
{
  "name": "crud-with-mongodb",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node app.js",
    "test": "mocha --timeout 10000"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^4.21.2",
    "mongoose": "^8.12.1"
  },
  "devDependencies": {
    "chai": "^5.2.0",
    "chai-http": "^5.1.1",
    "mocha": "^11.1.0"
  }
}
```

At the bottom, the Terminal panel shows a PowerShell prompt: `PS C:\Users\johnd\crud-with-mongodb>`. The status bar at the bottom indicates the cursor is at line 15, column 26, with 2 spaces, UTF-8 encoding, LF line endings, and JSON syntax highlighting.

Set Up A MongoDB Database for Your Node.js App

The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the project structure: `crud-with-mongodb` > `node_modules`, `app.js`, `package-lock.json`, and `package.json`. The main editor area shows the `app.js` file with the following content:

```
const express = require("express");
const app = express();
const mongoose = require("mongoose");

// middleware
app.use(express.json());

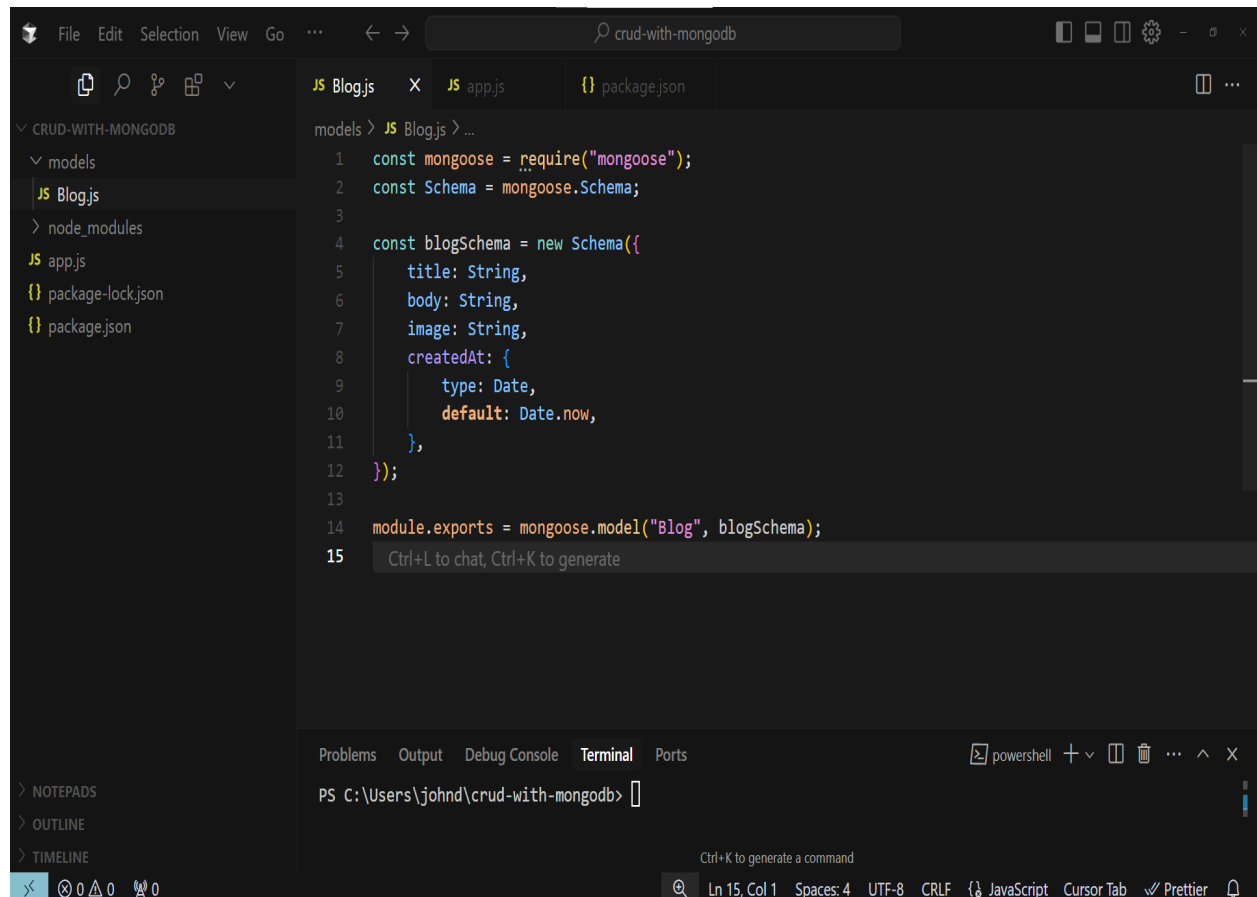
// Configure mongoose
mongoose.connect(
  process.env.MONGODB_URI || "mongodb://localhost/CRUD",
  {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  },
  (err) => {
    if (err) {
      console.log(err);
    } else {
      console.log("Connected to MongoDB");
    }
  }
);

app.listen(3001, () => {
  console.log("Server is running on port 3001");
});

module.exports = app;
```

At the bottom, the Terminal panel shows a PowerShell prompt: `PS C:\Users\johnd\crud-with-mongodb>`. The status bar at the bottom indicates the cursor is at line 4, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript syntax highlighting.

Build A Model



```
1 const mongoose = require("mongoose");
2 const Schema = mongoose.Schema;
3
4 const blogSchema = new Schema({
5   title: String,
6   body: String,
7   image: String,
8   createdAt: {
9     type: Date,
10    default: Date.now,
11  },
12 });
13
14 module.exports = mongoose.model("Blog", blogSchema);
15
```

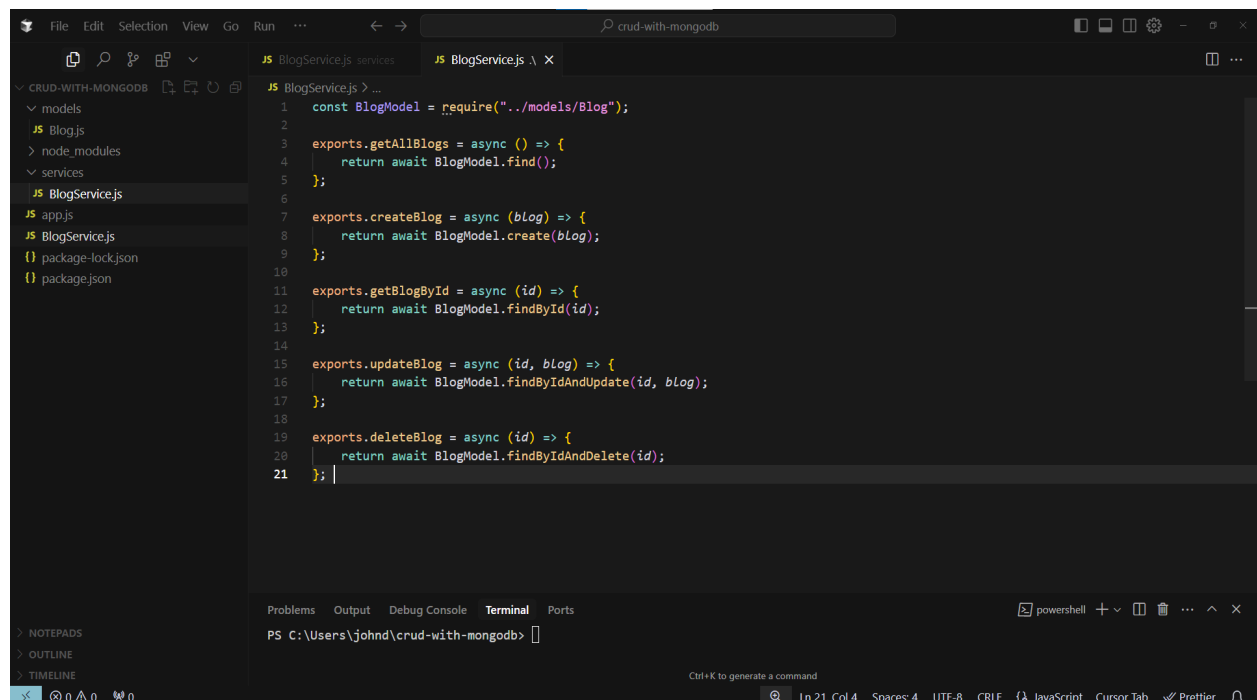
Ctrl+L to chat, Ctrl+K to generate

Problems Output Debug Console Terminal Ports

PS C:\Users\johnd\crud-with-mongodb>

Ln 15, Col 1 Spaces: 4 UTF-8 CRLF JavaScript Cursor Tab Prettier

Create Services



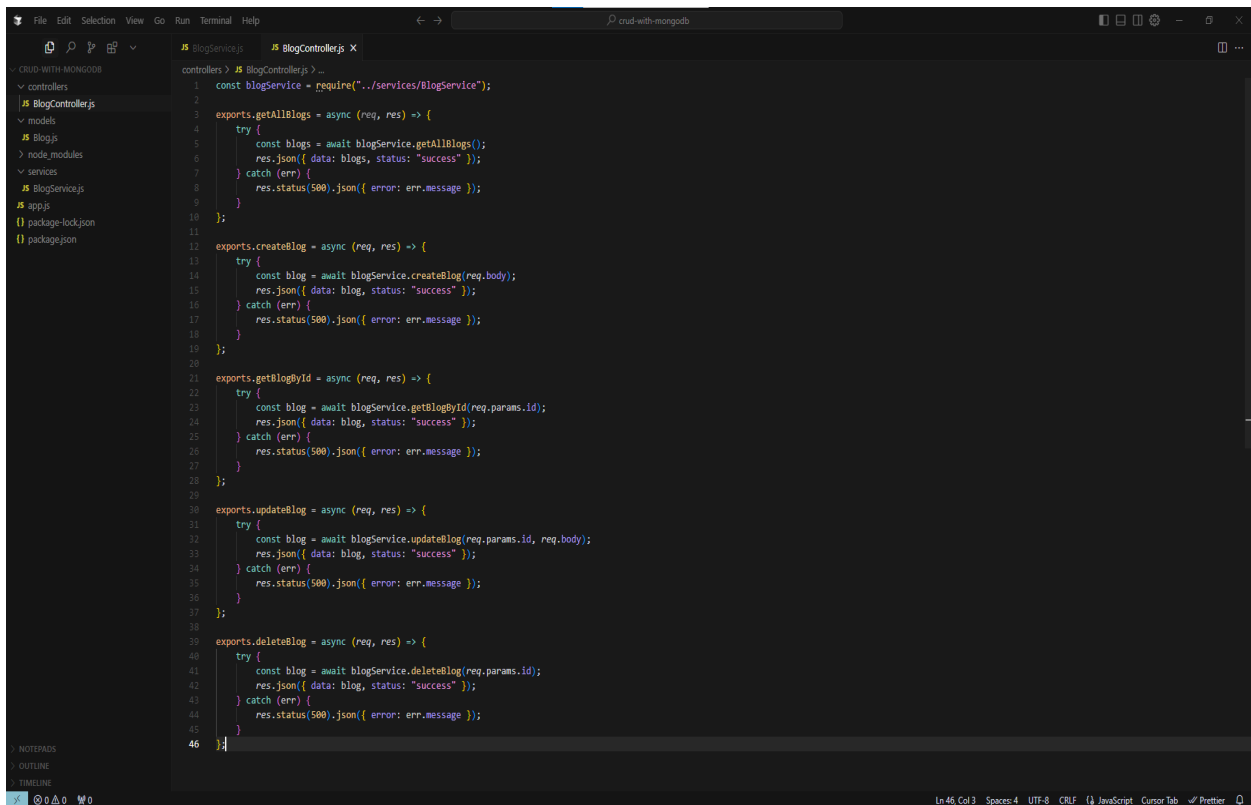
```
1 const BlogModel = require("../models/Blog");
2
3 exports.getAllBlogs = async () => {
4   return await BlogModel.find();
5 };
6
7 exports.createBlog = async (blog) => {
8   return await BlogModel.create(blog);
9 };
10
11 exports.getBlogById = async (id) => {
12   return await BlogModel.findById(id);
13 };
14
15 exports.updateBlog = async (id, blog) => {
16   return await BlogModel.findByIdAndUpdate(id, blog);
17 };
18
19 exports.deleteBlog = async (id) => {
20   return await BlogModel.findByIdAndDelete(id);
21 };
```

Problems Output Debug Console Terminal Ports

PS C:\Users\johnd\crud-with-mongodb>

Ln 21, Col 4 Spaces: 4 UTF-8 CRLF JavaScript Cursor Tab Prettier

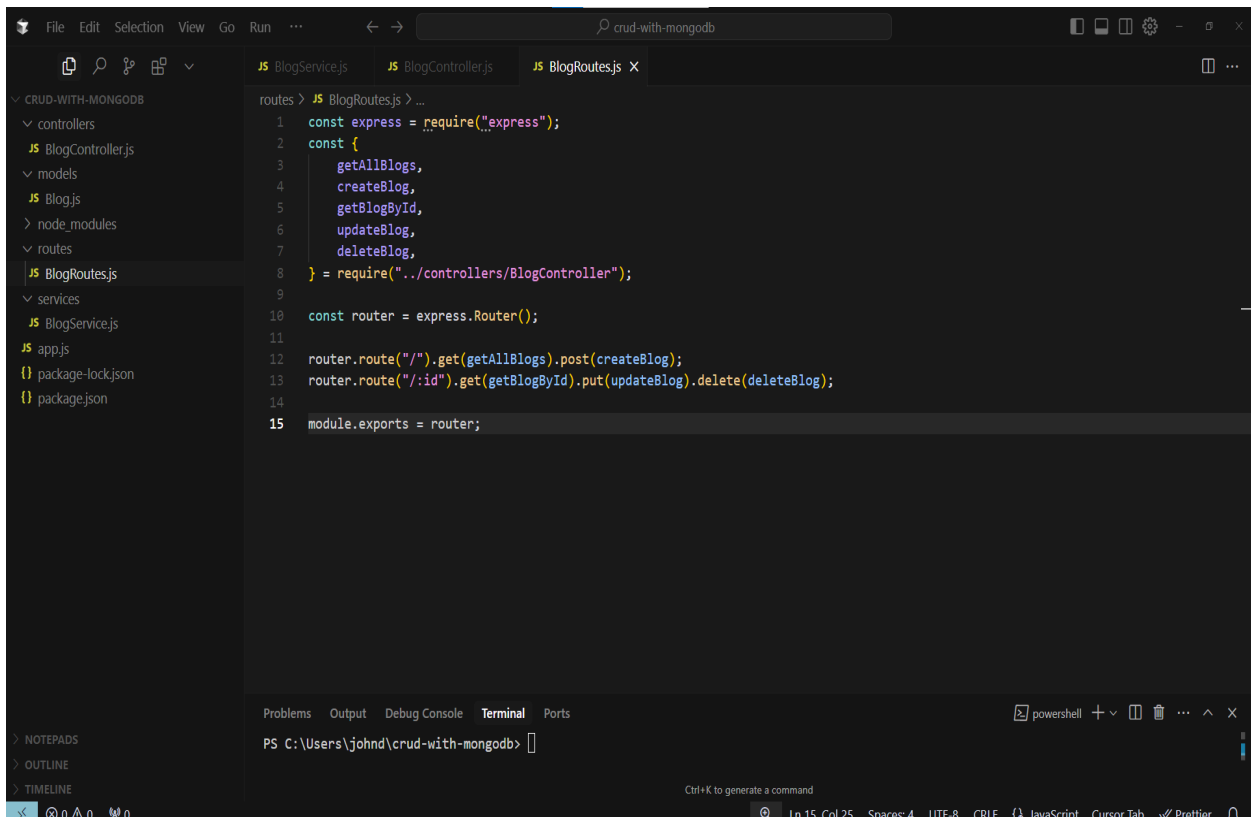
Set Up Controllers



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders for controllers, models, services, and app.js. The code editor displays the content of BlogController.js, which defines several asynchronous functions for handling blog data: getAllBlogs, createBlog, getBlogById, updateBlog, and deleteBlog. Each function uses the blogService module to interact with the database and returns a JSON response or an error.

```
1 const blogService = require("../services/BlogService");
2
3 exports.getAllBlogs = async (req, res) => {
4   try {
5     const blogs = await blogService.getAllBlogs();
6     res.json({ data: blogs, status: "success" });
7   } catch (err) {
8     res.status(500).json({ error: err.message });
9   }
10 };
11
12 exports.createBlog = async (req, res) => {
13   try {
14     const blog = await blogService.createBlog(req.body);
15     res.json({ data: blog, status: "success" });
16   } catch (err) {
17     res.status(500).json({ error: err.message });
18   }
19 };
20
21 exports.getBlogById = async (req, res) => {
22   try {
23     const blog = await blogService.getBlogById(req.params.id);
24     res.json({ data: blog, status: "success" });
25   } catch (err) {
26     res.status(500).json({ error: err.message });
27   }
28 };
29
30 exports.updateBlog = async (req, res) => {
31   try {
32     const blog = await blogService.updateBlog(req.params.id, req.body);
33     res.json({ data: blog, status: "success" });
34   } catch (err) {
35     res.status(500).json({ error: err.message });
36   }
37 };
38
39 exports.deleteBlog = async (req, res) => {
40   try {
41     const blog = await blogService.deleteBlog(req.params.id);
42     res.json({ data: blog, status: "success" });
43   } catch (err) {
44     res.status(500).json({ error: err.message });
45   }
46 };
```

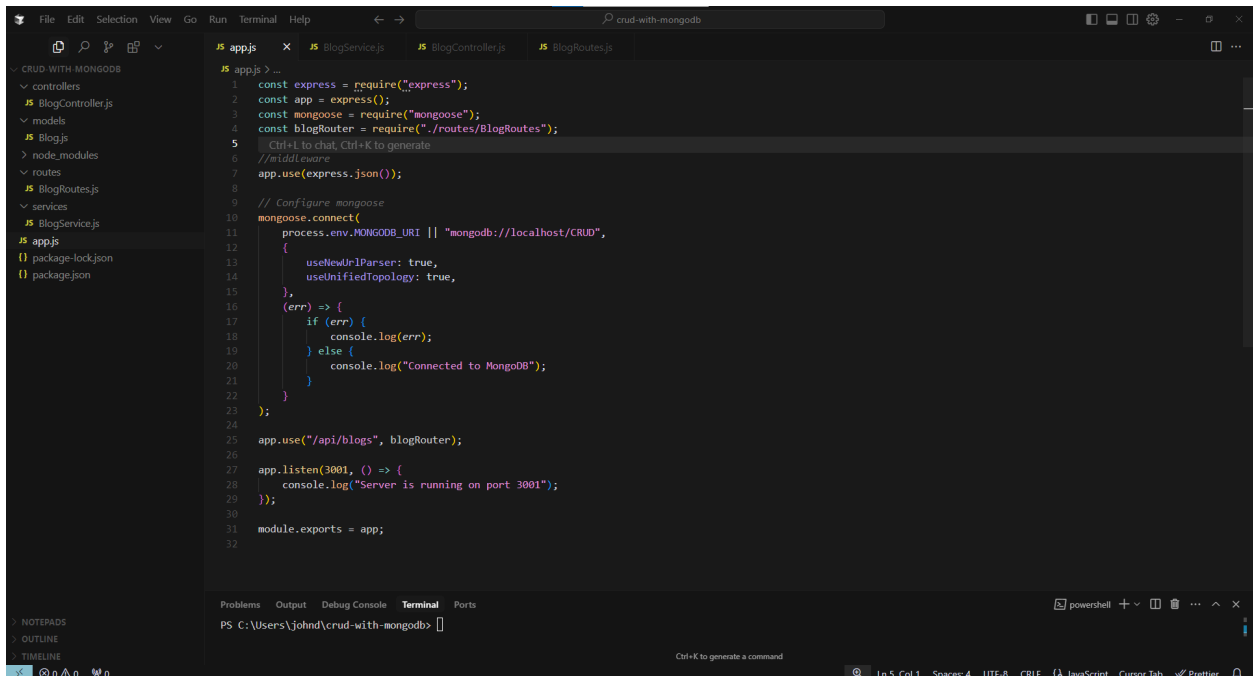
Create Routes for Controllers



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders for controllers, models, services, and app.js. The code editor displays the content of BlogRoutes.js, which sets up an Express.js router. It imports the express module, defines an array of route functions (getAllBlogs, createBlog, getBlogById, updateBlog, deleteBlog), and then uses the router module to define the actual routes. The routes are: a POST route for creating a blog, and GET, PUT, and DELETE routes for retrieving, updating, and deleting a blog by ID.

```
1 const express = require("express");
2 const {
3   getAllBlogs,
4   createBlog,
5   getBlogById,
6   updateBlog,
7   deleteBlog,
8 } = require("../controllers/BlogController");
9
10 const router = express.Router();
11
12 router.route("/").get(getAllBlogs).post(createBlog);
13 router.route("/:id").get(getBlogById).put(updateBlog).delete(deleteBlog);
14
15 module.exports = router;
```

Below the code editor, there is a terminal window showing the command prompt for PowerShell, indicating the current directory is C:\Users\johnd\crud-with-mongodb.

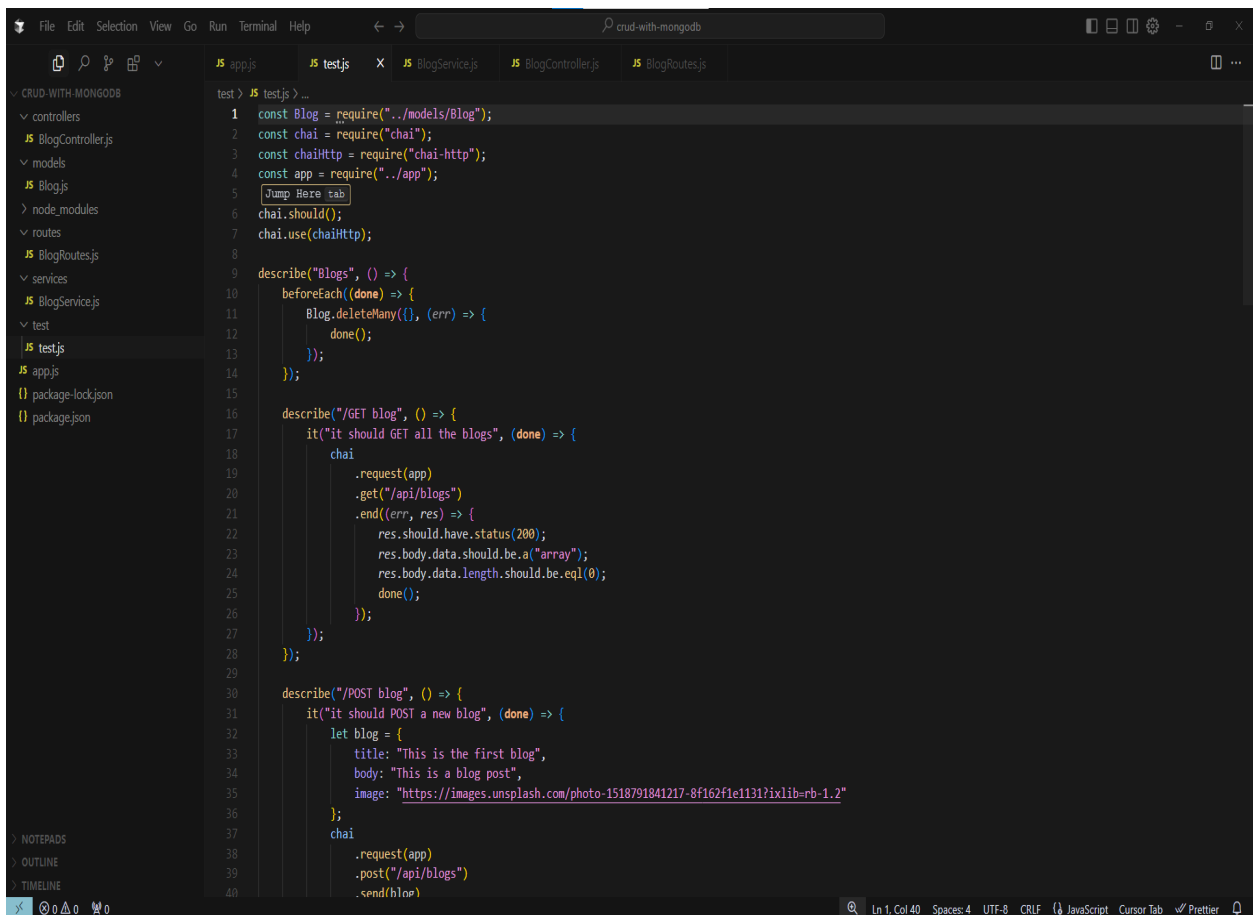


The screenshot shows the VS Code editor with the 'crud-with-mongodb' project open. The file explorer on the left shows the project structure, including controllers, models, routes, and services. The main editor displays the 'app.js' file, which contains the following code:

```
1 const express = require("express");
2 const app = express();
3 const mongoose = require("mongoose");
4 const blogRouter = require("./routes/BlogRoutes");
5 // Ctrl+I to chat, Ctrl+K to generate
6 // middleware
7 app.use(express.json());
8
9 // Configure mongoose
10 mongoose.connect(
11   process.env.MONGODB_URI || "mongodb://localhost/CRUD",
12   {
13     useNewUrlParser: true,
14     useUnifiedTopology: true,
15   },
16   (err) => {
17     if (err) {
18       console.log(err);
19     } else {
20       console.log("Connected to MongoDB");
21     }
22   }
23 );
24
25 app.use("/api/blogs", blogRouter);
26
27 app.listen(3001, () => {
28   console.log("Server is running on port 3001");
29 });
30
31 module.exports = app;
```

The terminal at the bottom shows the command prompt: `PS C:\Users\johnd\crud-with-mongodb>`.

Writing Automated Tests



The screenshot shows the VS Code editor with the 'crud-with-mongodb' project open. The file explorer on the left shows the project structure, including controllers, models, routes, and services. The main editor displays the 'test.js' file, which contains the following code:

```
test > JS testjs > ...
1 const Blog = require("../models/Blog");
2 const chai = require("chai");
3 const chaiHttp = require("chai-http");
4 const app = require("../app");
5 // Jump Here tab
6 chai.should();
7 chai.use(chaiHttp);
8
9 describe("Blogs", () => {
10   beforeEach((done) => {
11     Blog.deleteMany({}, (err) => {
12       done();
13     });
14   });
15
16   describe("/GET blog", () => {
17     it("it should GET all the blogs", (done) => {
18       chai
19         .request(app)
20         .get("/api/blogs")
21         .end((err, res) => {
22           res.should.have.status(200);
23           res.body.data.should.be.a("array");
24           res.body.data.length.should.be.eql(0);
25           done();
26         });
27     });
28   });
29
30   describe("/POST blog", () => {
31     it("it should POST a new blog", (done) => {
32       let blog = {
33         title: "This is the first blog",
34         body: "This is a blog post",
35         image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
36       };
37       chai
38         .request(app)
39         .post("/api/blogs")
40         .send(blog)
```

The terminal at the bottom shows the command prompt: `PS C:\Users\johnd\crud-with-mongodb>`.

```
File Edit Selection View Go Run Terminal Help crud-with-mongodb
JS app.js JS test.js x JS BlogService.js JS BlogController.js JS BlogRoutes.js

CRUD-WITH-MONGODB
  controllers
  JS BlogController.js
  models
  JS Blog.js
  > node_modules
  routes
  JS BlogRoutes.js
  services
  JS BlogService.js
  test
  JS test.js
  app.js
  () package-lock.json
  () package.json

test > JS test.js > ...
9 describe("Blogs", () => {
10   //
29
30   describe("/POST blog", () => {
31     it("it should POST a new blog", (done) => {
32       let blog = {
33         title: "This is the first blog",
34         body: "This is a blog post",
35         image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
36       };
37       chai
38         .request(app)
39         .post("/api/blogs")
40         .send(blog)
41         .end((err, res) => {
42           res.should.have.status(200);
43           res.body.data.should.be.a("object");
44           res.body.status.should.be.eql("success");
45           done();
46         });
47     });
48   });
49
50   describe("/GET/:id blog", () => {
51     it("it should GET a blog by the id", (done) => {
52       let blog = new Blog({
53         title: "This is the first blog",
54         body: "This is a blog post",
55         image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
56       });
57       blog.save((err, blog) => {
58         chai
59           .request(app)
60           .get("/api/blogs/" + blog.id)
61           .end((err, res) => {
62             res.should.have.status(200);
63             res.body.data.should.be.a("object");
64             res.body.status.should.be.eql("success");
65             done();
66           });
67     });
68   });
69 });

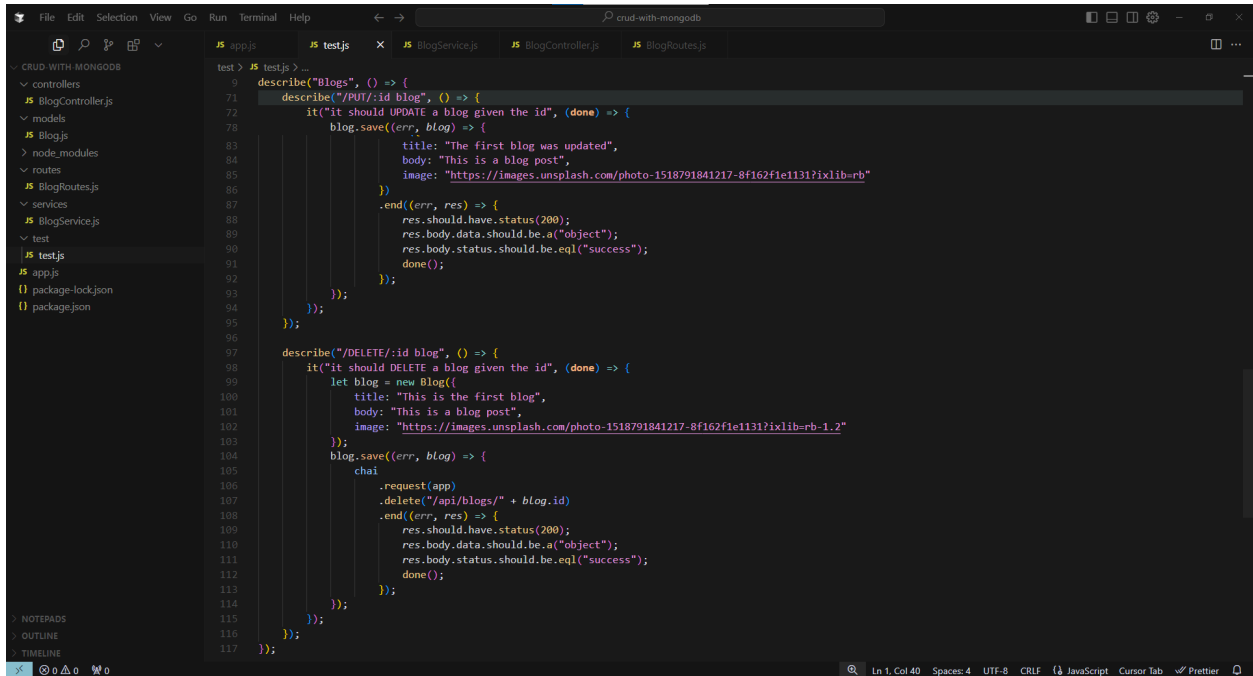
Ln 1, Col 40 Spaces: 4 UTF-8 CRLF JavaScript Cursor Tab Prettier
```

```
File Edit Selection View Go Run Terminal Help crud-with-mongodb
JS app.js JS test.js x JS BlogService.js JS BlogController.js JS BlogRoutes.js

CRUD-WITH-MONGODB
  controllers
  JS BlogController.js
  models
  JS Blog.js
  > node_modules
  routes
  JS BlogRoutes.js
  services
  JS BlogService.js
  test
  JS test.js
  app.js
  () package-lock.json
  () package.json

test > JS test.js > ...
9 describe("Blogs", () => {
50   describe("/GET/:id blog", () => {
51     it("it should GET a blog by the id", (done) => {
57       blog.save((err, blog) => {
61         .end((err, res) => {
66           });
67       });
68     });
69   });
70
71   describe("/PUT/:id blog", () => {
72     it("it should UPDATE a blog given the id", (done) => {
73       let blog = new Blog({
74         title: "This is the first blog",
75         body: "This is a blog post",
76         image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
77       });
78       blog.save((err, blog) => {
79         chai
80           .request(app)
81           .put("/api/blogs/" + blog.id)
82           .send({
83             title: "The first blog was updated",
84             body: "This is a blog post",
85             image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
86           })
87           .end((err, res) => {
88             res.should.have.status(200);
89             res.body.data.should.be.a("object");
90             res.body.status.should.be.eql("success");
91             done();
92           });
93       });
94     });
95   });
96
97   describe("/DELETE/:id blog", () => {
98     it("it should DELETE a blog given the id", (done) => {
99       let blog = new Blog({
100         title: "This is the first blog"
101       });
102       blog.save((err, blog) => {
103         chai
104           .request(app)
105           .delete("/api/blogs/" + blog.id)
106           .end((err, res) => {
107             res.should.have.status(200);
108             res.body.data.should.be.a("object");
109             res.body.status.should.be.eql("success");
110             done();
111           });
112       });
113     });
114   });
115 });

Ln 1, Col 40 Spaces: 4 UTF-8 CRLF JavaScript Cursor Tab Prettier
```



```
test > JS test.js > ...
9 describe("Blogs", () => {
10   describe("/PUT/:id blog", () => {
11     it("it should UPDATE a blog given the id", (done) => {
12       blog.save((err, blog) => {
13         title: "The first blog was updated",
14         body: "This is a blog post",
15         image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
16       });
17       .end((err, res) => {
18         res.should.have.status(200);
19         res.body.data.should.be.a("object");
20         res.body.status.should.be.eql("success");
21         done();
22       });
23     });
24   });
25 });
26
27 describe("/DELETE/:id blog", () => {
28   it("it should DELETE a blog given the id", (done) => {
29     let blog = new Blog({
30       title: "This is the first blog",
31       body: "This is a blog post",
32       image: "https://images.unsplash.com/photo-1518791841217-8f162f1e1131?ixlib=rb-1.2"
33     });
34     blog.save((err, blog) => {
35       chai
36         .request(app)
37         .delete("/api/blogs/" + blog.id)
38         .end((err, res) => {
39           res.should.have.status(200);
40           res.body.data.should.be.a("object");
41           res.body.status.should.be.eql("success");
42           done();
43         });
44     });
45   });
46 });
47
48 });
49
50 });
```

```
Blogs API
MongoDB Connection Error: bad auth : authentication failed
PS C:\Users\johnd\crud-with-mongodb> npm run test

> crud-with-mongodb@1.0.0 test
> mocha --timeout 30000

(node:6472) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver
version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:6472) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js D
river version 4.0.0 and will be removed in the next major version
Server is running on port 3001

Blogs API
Connected to MongoDB
  ✓ should GET all blogs (87ms)
  ✓ should POST a blog (121ms)
  ✓ should GET a blog by ID (132ms)
  ✓ should UPDATE a blog by ID (139ms)
  ✓ should DELETE a blog by ID (136ms)

5 passing (2s)
```


Testing Your Node.js App with Postman GET

The screenshot shows the Postman interface with a GET request to `localhost:3001/api/blogs`. The request body is a JSON object:

```
1 {
2   "title": "The first blog was updated",
3   "body": "This is a blog post",
4   "image": "https://images.unsplash.com/photo-1518791841217-8f162f1e1131"
5 }
6
```

The response is a 200 OK status with a JSON body:

```
1 {
2   "status": "success",
3   "data": [
4     {
5       "_id": "67d17a2288b99c7f7b22fc53",
6       "title": "First Blog",
7       "body": "This is a post",
8       "image": "https://placeholder.co/600x400",
9       "createdAt": "2025-03-12T12:12:18.102Z",
10      "__v": 0
11    }
12  ]
13 }
```

POST

The screenshot shows the Postman interface with a POST request to `localhost:3001/api/blogs`. The request body is a JSON object:

```
1 {
2   "title": "The first blog was updated",
3   "body": "This is a blog post",
4   "image": "https://images.unsplash.com/photo-1518791841217-8f162f1e1131"
5 }
6
```

The response is a 201 Created status with a JSON body:

```
1 {
2   "status": "success",
3   "data": {
4     "title": "The first blog was updated",
5     "body": "This is a blog post",
6     "image": "https://images.unsplash.com/photo-1518791841217-8f162f1e1131",
7     "_id": "67d17ab588b99c7f7b22fc5f",
8     "createdAt": "2025-03-12T12:14:45.437Z",
9     "__v": 0
10   }
11 }
```