9530

ST.MOTHER THERESA ENGINEERING COLLEGE
COMPUTER SCIENCE AND ENGINEERING

NM-ID:A63174B3B81ACDA9
2C64BB148209D943

REG NO:953023104130
DATE:14-09-2025

Completed the project named as

PHASE -2

Node.js Backend for Contact Form

SUBMITTED BY
N.THANGA ESWARI

PH NO: 7010355144

TITLE

Node.js Backend for Contact Form

Problem statement

Many websites rely on contact forms to collect feedback, queries, or requests from users. Without a secure backend, form submissions may be lost, vulnerable to spam, or hard to manage. Improper validation and lack of storage result in data inaccuracy, poor user experience, and increased risk of attacks such as SQL injection and spamming. Therefore, a backend system is required to securely handle, validate, and store form submissions.

Objective

The objective of this project is to design and implement a Node.js backend that ensures secure handling of contact form data. The backend validates user inputs (name, email, message), prevents invalid data, stores submissions in a MongoDB database, and optionally sends email notifications to the admin.

AIM 1

To develop a secure REST API endpoint that accepts and validates contact form submissions

AIM 2

To implement data storage in MongoDB and provide admin access to view submissions using JWT authentication.

Flowchart

User (Contact Form) → POST /api/contact → Express Router → Validation → Save to MongoDB →

Success Response (JSON) → (Optional) Nodemailer sends email to admin.

PROGRAM

```
// server.js (simplified)
```

```
const express = require('express');
const mongoose = require('mongoose');
const { body, validationResult } = require('express-validator');
const app = express();
app.use(express.json());
mongoose.connect('mongodb://localhost:27017/contact_form_db');
const Contact = mongoose.model('Contact', {
 name: String, email: String, message: String
});
app.post('/api/contact', [
 body('name').notEmpty(),
 body('email').isEmail(),
 body('message').notEmpty()
], async (req, res) => {
 const errors = validationResult(req);
 if (!errors.isEmpty()) return res.status(400).json({ errors: errors.array() });
 const contact = new Contact(req.body);
 await contact.save();
 res.json({ success: true, message: 'Submission saved' });
});
app.listen(5000, () => console.log('Server running'));
```

Project hurdles

1 Spam Protection – preventing bots from flooding the API.

2 Validation – ensuring only valid emails and messages are accepted.

3 Database Scaling – handling large volumes of submissions efficiently.

4 Authentication – securing admin access with JWT.

5 Deployment Issues – configuring environment variables for SMTP and MongoDB.

Design & Architecture

1. Tech Stack Selection: Node.js (Express), MongoDB, JWT, Nodemailer.

2. UI Structure: Contact form with Name, Email, Message fields.

3. API Schema: { 'name': 'string', 'email': 'string', 'message': 'string' }

4. Data Handling Approach: Validate → Sanitize → Store in MongoDB → Respond → Notify

Admin.

5. Component / Module Diagram: Frontend Form → API Route → Validation → Database →

(Optional) Notification.

Tech Stack Selection

- **Backend Framework:** Node.js with Express.js
- **Database:** MongoDB (for storing contact form submissions)
- **Validation:** Express-validator / Joi
- **Authentication (Admin):** JWT (JSON Web Tokens)
- **Email Notifications:** Nodemailer (optional)
- **Security Tools:** Helmet, xss-clean, express-rate-limit
- **Deployment:** Heroku / Render / AWS

UI Structure / API Schema Design

- **UI Structure (Frontend Contact Form):**

    - Input fields: Name, Email, Message
    - Submit button → sends POST request to backend API
- **API Schema:**
    - **POST `/api/contact`**
        - Request Body:
        - ```
          {
            "name": "John Doe",
            "email": "john@example.com",
            "message": "Hello! I want to know more."
          }
          ```
        - Response:
        - ```
          {
            "success": true,
            "message": "Submission saved"
          }
          ```
    - **GET `/api/contact` (Admin only)**
        - Returns list of all submissions with pagination

Data Handling Approach

1. **Validation**: Ensure all required fields are provided and valid (email format check).
2. **Sanitization**: Remove any malicious scripts (XSS protection).
3. **Database Storage**: Save submissions in MongoDB collection (`contacts`).
4. **Optional Email**: Notify admin about new submissions via Nodemailer.
5. **Admin Access**: Use JWT-based authentication to restrict viewing submissions

---

✅ Program Outputs (Node.js Backend for Contact Form)

1) Starting the Server

When you run:

```
npm start
```

Console Output:

```
MongoDB connected
Server running on port 5000
```

---

2) POST Request – Submit Contact Form

Request (Frontend → Backend API):

```
POST http://localhost:5000/api/contact
Content-Type: application/json

{
  "name": "Alice",
  "email": "alice@example.com",
  "message": "Hello, I am interested in your
services."
}
```

Response (Backend → Frontend):

```
{
  "success": true,
  "message": "Submission saved"
}
```

Database (MongoDB document stored):

```
{
  "_id": "650123abcd456ef7890",
  "name": "Alice",
  "email": "alice@example.com",
  "message": "Hello, I am interested in your
services.",
  "createdAt": "2025-09-15T10:15:00.000Z",
  "updatedAt": "2025-09-15T10:15:00.000Z",
  "__v": 0
}
```

---

3) POST Request – With Invalid Email

Request:

```
POST http://localhost:5000/api/contact
Content-Type: application/json

{
  "name": "Bob",
```

```
  "email": "bob@@wrong.com",
  "message": "This should fail"
}
```

Response:

```
{
  "errors": [
    {
      "msg": "Invalid email",
      "param": "email",
      "location": "body"
    }
  ]
}
```

---

4) GET Request – Admin Views Submissions

Request:

```
GET http://localhost:5000/api/contact
Authorization: Bearer <ADMIN_JWT_TOKEN>
```

Response:

```
{
  "page": 1,
  "limit": 10,
  "total": 2,
  "items": [
    {
      "_id": "650123abcd456ef7890",
      "name": "Alice",
      "email": "alice@example.com",
      "message": "Hello, I am interested in your
services.",
      "createdAt": "2025-09-15T10:15:00.000Z"
    },
    {
      "_id": "650124efgh567ij8901",
      "name": "Charlie",
      "email": "charlie@example.com",
      "message": "Need more details about pricing.",
      "createdAt": "2025-09-15T11:00:00.000Z"
    }
  ]
}
```

Conclusion

The Node.js backend for the contact form ensures secure handling of user inputs, improves datareliability, and allows administrators to manage submissions effectively. By using Express,MongoDB, and JWT, the project demonstrates modern web backend development practices.