# API-mongo

---

# Node.js & MongoDB User API with v1 and v2

## What is this project?

This project is a **web application** where you can:

- Submit **users’ details** (name, email, phone)
- Save them in a **MongoDB database**
- View all users
- Update or delete a user

We also have **two versions of the API:**

- **v1** → simple, only name + email
- **v2** → advanced, includes phone too

---

## Project Structure

```
project/
│
├─ controllers/          # Contains logic for APIs
│   ├─ v1/userController.js
│   └─ v2/userController.js
│
├─ models/
│   └─ userModel.js      # Database schema
│
├─ routes/
│   ├─ v1/userRoutes.js
│   └─ v2/userRoutes.js
│
├─ config/
│   └─ db.js             # Connect to MongoDB
│
```

```
│
├─ public/
│   └─ index.html         # The form to submit data
│
└─ server.js              # Main server file
```

---

## Database (MongoDB)

- **Database Name:** `formdb`
- **Collection Name:** `users`

**Schema** (User model):

| Field | Type | Required | Notes |
|---|---|---|---|
| name | String | Yes | User's name |
| email | String | Yes | User's email |
| phone | String | Optional | Required only in v2 |
| createdAt | Date | Yes | Automatically set |

> MongoDB automatically creates the database and collection when we insert the first document.

**MongoDB Shell Commands:**

```
use formdb
db.createCollection("users")
db.users.insertOne({name:"Alice", email:"alice@example.com",
phone:"123", createdAt:new Date()})
```

---

## server.js (Main Entry Point)

- **Purpose:** Starts the server, connects to MongoDB, and loads routes.
- **Key parts:**

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const connectDB = require('./config/db');

const v1Routes = require('./routes/v1/userRoutes');
const v2Routes = require('./routes/v2/userRoutes');

const app = express();
connectDB();

app.use(cors());
app.use(bodyParser.json());

app.use('/api/v1', v1Routes); // v1 routes
app.use('/api/v2', v2Routes); // v2 routes

app.get('/', (req,res)=>res.sendFile(__dirname +
'/public/index.html'));

app.listen(3000, () => console.log('Server running on
http://localhost:3000'));
```

---

## Controllers (Business Logic)

### v1 Controller

- Handles **name + email only**

```
const createUser = async (req,res)=>{
  const { name, email } = req.body;
  if(!name || !email) return res.status(400).json({message:'Name &
Email required'});
  const user = new User({name,email});
  await user.save();
  res.json({message:'User added (v1)'});
}
```

### v2 Controller

- Handles **name + email + phone**

```
const createUser = async (req,res)=>{
  const { name, email, phone } = req.body;
  if(!name || !email || !phone) return
res.status(400).json({message:'All fields required'});
  const user = new User({name,email,phone});
  await user.save();
  res.json({message:'User added (v2)', user});
}
```

✅ **Other methods (getUsers, getUserById, updateUser, deleteUser)** are similar but v2 includes **phone**.

---

## 6️⃣ Routes

- **v1 Routes:** `/api/v1/...`

```
router.post('/submit', createUser);
router.get('/all', getUsers);
router.get('/:id', getUserById);
router.put('/:id', updateUser);
router.delete('/:id', deleteUser);
```

- **v2 Routes:** `/api/v2/...` → same endpoints but use **v2 controller**.

  The version difference is **mainly in the controller and fields.**

---

## 7️⃣ index.html (Form)

### For v1 (name + email only)

```
<input type="text" id="name" name="name" required>
<input type="email" id="email" name="email" required>
```

### For v2 (name + email + phone)

```
<input type="text" id="name" name="name" required>
<input type="email" id="email" name="email" required>
<input type="text" id="phone" name="phone" required>
```

- **JS snippet to send data to API:**

```
const data = {
  name: document.getElementById('name').value,
  email: document.getElementById('email').value,
  phone: document.getElementById('phone')?.value // optional for v1
};

fetch(`/api/${API_VERSION}/submit`, {
  method:'POST',
  headers:{'Content-Type':'application/json'},
  body:JSON.stringify(data)
})
```

---

## How API Versions Work

| Version | Fields | Notes |
|---|---|---|
| v1 | name, email | Simple, old clients still work |
| v2 | name, email, phone | New feature, phone required |

- **Switch frontend** between v1 and v2 using:

```
const API_VERSION = 'v2'; // or 'v1'
```

---

## Summary / Step-by-Step Workflow

1. Start MongoDB (`mongod`)
2. Run `node server.js`
3. Open browser → `http://localhost:3000`

4. Fill form → Submit
5. Data is stored in **MongoDB**
6. v1 → only saves name + email
7. v2 → saves name + email + phone
8. GET /all → fetches all users
9. PUT /:id → updates user
10. DELETE /:id → deletes user

---

## Extra Tips

- Always check **API_VERSION** in frontend
- Make **phone optional in schema** for v1 to work
- Use **Postman** to test API endpoints separately
- **v2 is just an improved version of v1** — new fields, validation, and responses

---