

To create the described project in **Visual Studio Code (VS Code)**, follow these steps:

1. Set Up Your Project Directory

1. Open **VS Code**.
 2. Create a new folder for your project, e.g., `MyAPIProject`.
 3. Open the folder in VS Code (`File > Open Folder`).
-

2. Initialize Your Project

1. Open a terminal in VS Code (`Ctrl + ~` or `View > Terminal`).
2. Initialize a new Node.js project:

```
npm init -y
```

This creates a `package.json` file to manage your project's dependencies.

3. Install the required dependencies:

```
npm install express body-parser cors mongoose
```

3. Create Backend Files

1. **Inside the project folder**, create a folder named `backend`.
 2. Inside the `backend` folder, create a file named `server.js`.
 3. Copy the code into `server.js`:
-

4. Create Frontend Files

1. **Inside the project folder**, create a folder named `frontend`.
 2. Inside the `frontend` folder, create a file named `index.html`.
 3. Copy the code into `index.html`:
-

5. Start the Backend Server

1. Navigate to the `backend` directory in the terminal:

```
cd backend
```

2. Start the server:

```
node server.js
```

6. Open the Frontend

1. Navigate to the `frontend` folder and open `index.html` in a browser (drag and drop or open via file path).
 2. Test the form by submitting data and see it displayed in the "Fetched Data" section.
-

Errors and Issues: Resolved

Error 1:

On starting the server as mentioned in Step 5, I am getting the following error:

```
Error: listen EADDRINUSE: address already in use :::5000 at Server.setupListenHandle [as
_listen2] (node:net:1937:16) at listenInCluster (node:net:1994:12) at Server.listen
(node:net:2099:7) at Function.listen
<Project_Path>node_modules/express/lib/application.js:635:24) at Object.<anonymous>
<Project_Path>backend/server.js:47:5) at Module._compile
(node:internal/modules/cjs/loader:1740:14) at Object.js
(node:internal/modules/cjs/loader:1905:10) at Module.load
(node:internal/modules/cjs/loader:1474:32) at Function._load
(node:internal/modules/cjs/loader:1286:12) at TracingChannel.traceSync
(node:diagnostics_channel:322:14) Emitted 'error' event on Server instance at: at emitErrorNT
(node:net:1973:8) at process.processTicksAndRejections
(node:internal/process/task_queues:90:21) { code: 'EADDRINUSE', errno: -48, syscall: 'listen',
address: ':::', port: 5000 }
```

The error `EADDRINUSE: address already in use` occurs when the specified port (in this case, 5000) is already in use by another application or process. Here's how you can resolve this issue:

Error 1: Solutions

1. Identify and Kill the Process Using Port 5000

1. Open your terminal and check which process is using port 5000:

```
lsof -i :5000
```

2. This will return a list of processes using the port. Look for the `PID` (Process ID) of the process.
3. Kill the process using the `PID`:

```
kill -9 <PID>
```

Replace `<PID>` with the actual process ID.

4. Retry starting your server:

```
node server.js
```

2. Change the Port in Your Server

If killing the process is not feasible or you prefer a different approach, change the port in your `server.js` file:

1. Open `server.js`.
2. Update the `listen` method with a new port (e.g., 5001):

```
app.listen(5001, () => console.log('Server running on port 5001'));
```

3. Restart the server.
-

3. Check for Orphaned Processes

Sometimes, a previously crashed process may still hold onto the port:

- Restart your system to clear all processes if the issue persists.
-

4. Use a Dynamic Port

You can modify your server to use a dynamic port by leveraging the `process.env` object:

```
const PORT = process.env.PORT || 5000;  
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

This allows you to specify a port when starting the server:

```
PORT=5001 node server.js
```

Query:

I have resolved the issue using Dynamic Port. How should the `index.html` be updated to listen to this port?

To ensure the **frontend** (`index.html`) works dynamically with the backend using the updated dynamic port, you need to make some adjustments to fetch the correct port for API requests.

Solution

1. Use an Environment Variable or Configure Base URL

The frontend must know the backend's host and port. This can be achieved in two ways:

Option A: Use a Static Configuration

If the backend is hosted locally or on a specific environment, you can define the base URL for API requests in the JavaScript section of your `index.html` file.

Update in `index.html`:

```
const baseUrl = "http://localhost:5000"; // Replace 5000 with your dynamic port if known
```

Then, update the `fetch` calls in your JavaScript to use this base URL:

```
const response = await fetch(`${baseUrl}/api/data`, {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ name, email, message }),
});
```

Option B: Dynamically Detect the Port

If the port is entirely dynamic, the backend must inform the frontend of the correct port or allow requests from the same origin.

Steps:

1. **Allow Same-Origin Requests (Simplest for Localhost):** Use `window.location` to detect the hostname and port dynamically:

```
const baseUrl =
`${window.location.protocol}//${window.location.hostname}:${window.location.port || 5000}`;
```

2. **Proxy Setup (For Production):** If deploying to production, use a proxy to route API requests without hardcoding the backend's port. For example, in a production environment with tools like Nginx, configure a reverse proxy.

2. Updated `index.html` Example

Here's how your updated `index.html` would look with dynamic port handling:

```

<script>
  // Detect backend base URL dynamically
  const baseURL =
`${window.location.protocol}//${window.location.hostname}:${5000}`; //
Replace 5000 with your backend's fallback port

  document.getElementById('dataForm').addEventListener('submit', async (e)
=> {
    e.preventDefault();
    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const message = document.getElementById('message').value;

    try {
      const response = await fetch(`${baseURL}/api/data`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name, email, message }),
      });
      const result = await response.json();
      alert(result.message);
      fetchData();
    } catch (error) {
      alert('Error submitting data.');

```

3. Test Your Setup

- **Run the backend with the dynamic port.**
- Open the frontend in your browser and verify:
 1. You can submit data successfully.
 2. The fetched data displays correctly.

Error 2:

I am getting the following errors in console:

"[Error] Failed to load resource: Could not connect to the server. (data, line 0)

[Error] Not allowed to load local resource: <file:///favicon.ico>

[Error] Could not connect to the server.

[Error] Fetch API cannot load <http://localhost:5001/api/data> due to access control checks.

[Error] Failed to load resource: Could not connect to the server. (data, line 0)"

The errors you are encountering can be caused by a few different issues. Let's address them step-by-step:

Error 2: Solutions

Failed to Load Resource: Could Not Connect to the Server

This indicates that the server is not running or the frontend cannot reach the backend at the specified address (<http://localhost:5001/api/data>).

Steps to Resolve:

- Ensure the server is running:

```
node server.js
```
- Verify the server port:
 - If you used a dynamic port, ensure you know which port the backend is running on (e.g., 5001 or another).
 - Update the port in your `index.html` file if needed.

Check the Backend API

- Test the endpoint in your browser or with a tool like **Postman**:

```
http://localhost:5001/api/data
```

- If the backend responds, the issue lies in the connection between the frontend and backend.

Query:

Tested the Get method in Postman. Got the following error: "Error: connect ECONNREFUSED 127.0.0.1:5001"

The error `Error: connect ECONNREFUSED 127.0.0.1:5001` indicates that the Postman client is unable to connect to your backend server on port 5001. This typically means that:

1. The server is not running.
2. The server is running on a different port.
3. There is a network or configuration issue preventing the connection.

Steps to Resolve

1. Check if the Server is Running

- Open a terminal and check if the server is running.
- If the server is not running, start it:

```
node server.js
```

- Confirm in the terminal that the server is listening on the correct port (e.g., 5001):

```
Server running on port 5001
```

2. Verify the Port

- Ensure that the port in the `server.js` file matches the port you are trying to access in Postman (5001):

```
const PORT = process.env.PORT || 5001;  
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

- If you suspect the port might be dynamically assigned, check the actual port using a logging statement or confirm in the terminal where the server is running.

3. Check for Port Conflicts

- Use the following command to see if another process is using the port:

```
lsof -i :5001
```

- If another process is using the port, terminate it:

```
kill -9 <PID>
```

4. Test the Endpoint Locally

- Open your browser and navigate to:

`http://localhost:5001/api/data`

- If this doesn't work, your backend server might have an issue. Check the console logs for errors.

5. Network or Firewall Issues

- Ensure that no firewall or security settings are blocking access to the port.

6. Debugging the Backend

- Add a basic route to ensure the server is working:

```
app.get('/', (req, res) => {  
  res.send('Server is running!');  
});
```

- Access `http://localhost:5001/` in your browser or Postman to confirm the server is running.

Error 3:

I am getting the following error:

```
"err = new ServerSelectionError(); ^ MongooseServerSelectionError: connect  
ECONNREFUSED ::1:27017, connect ECONNREFUSED 127.0.0.1:27017 at  
_handleConnectionErrors <Project_Path>node_modules/mongoose/lib/connection.js:1110:11)  
at NativeConnection.openUri  
<Project_Path>node_modules/mongoose/lib/connection.js:1041:11) { reason:  
TopologyDescription { type: 'Unknown', servers: Map(1) { 'localhost:27017' =>  
ServerDescription { address: 'localhost:27017', type: 'Unknown', hosts: [], passives: [], arbiters:  
[], tags: {}, minWireVersion: 0, maxWireVersion: 0, roundTripTime: -1, minRoundTripTime: 0,  
lastUpdateTime: 779014886, lastWriteDate: 0, error: MongoNetworkError: connect  
ECONNREFUSED ::1:27017, connect ECONNREFUSED 127.0.0.1:27017 at  
Socket.<anonymous> <Project_Path>node_modules/mongodb/lib/cmap/connect.js:285:44) at  
Object.onceWrapper (node:events:628:26) at Socket.emit (node:events:513:28) at emitErrorNT  
(node:internal/streams/destroy:170:8) at emitErrorCloseNT  
(node:internal/streams/destroy:129:3) at process.processTicksAndRejections  
(node:internal/process/task_queues:90:21) { [Symbol(errorLabels)]: Set(1) { 'ResetPool' },  
[cause]: AggregateError [ECONNREFUSED]: at internalConnectMultiple (node:net:1139:18)  
at afterConnectMultiple (node:net:1712:7) { code: 'ECONNREFUSED', [errors]: [Array] } },  
topologyVersion: null, setName: null, setVersion: null, electionId: null,  
logicalSessionTimeoutMinutes: null, maxMessageSizeBytes: null, maxWriteBatchSize: null,  
maxBsonObjectSize: null, primary: null, me: null, '$clusterTime': null, iscryptd: false } }, stale:  
false, compatible: true, heartbeatFrequencyMS: 10000, localThresholdMS: 15, setName: null,  
maxElectionId: null, maxSetVersion: null, commonWireVersion: 0,  
logicalSessionTimeoutMinutes: null }, code: undefined }"
```

The error `MongooseServerSelectionError: connect ECONNREFUSED ::1:27017, connect ECONNREFUSED 127.0.0.1:27017` occurs because your application is unable to connect to the MongoDB server. This typically means the MongoDB service is not running or the connection string is misconfigured.

Error 3: Solutions

1. Start MongoDB

- Ensure that the MongoDB service is running. Use the following commands based on your system:

Windows:

1. Open Command Prompt or PowerShell with admin privileges.
2. Start the MongoDB service:

```
net start MongoDB
```

If the service is not installed, run:

```
mongod
```

macOS/Linux:

1. Open a terminal and start MongoDB:

```
brew services start mongodb-community
```

Or run it manually:

```
mongod
```

2. Confirm MongoDB is running by checking its status:

```
ps aux | grep mongod
```

2. Verify the Connection String

The default MongoDB connection string is `mongodb://localhost:27017/<database_name>`. In your `server.js` file, make sure you are using the correct connection string:

```
mongoose.connect('mongodb://localhost:27017/mydb', { useNewUrlParser: true, useUnifiedTopology: true });
```

If MongoDB is not running on the default port (27017), update the port accordingly.

3. Check IPv6 Compatibility

The error mentions `::1`, which is the IPv6 address for localhost. If your MongoDB is not configured for IPv6, you can force Mongoose to use IPv4 by modifying your connection string:

```
mongoose.connect('mongodb://127.0.0.1:27017/mydb', { useNewUrlParser: true, useUnifiedTopology: true });
```

4. Test MongoDB Connection

Manually test the connection to MongoDB using the MongoDB shell:

```
mongo --host localhost --port 27017
```

If you can connect successfully, the issue is likely in your application configuration.

5. Inspect Logs for MongoDB

If MongoDB fails to start, check its logs for errors:

- **Default log path:** `/var/log/mongodb/mongod.log` (Linux/macOS) or the path specified in the `mongod.conf` file.
-

6. Reinstall MongoDB (if necessary)

If MongoDB still fails to start, reinstall it:

- **Windows:** Download the installer from the [MongoDB website](#).
 - **macOS/Linux:** Use a package manager like `brew` or `apt-get`.
-

7. Update Your Application Logs

Add error handling to your Mongoose connection to log detailed errors:

```
javascript  
Copy code
```

```
mongoose.connect('mongodb://127.0.0.1:27017/mydb', { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch((err) => console.error('MongoDB connection error:', err));
```

Query:

On running brew services start mongodb-community the following error is received. "No available formula with the name "mongodb-community". Did you mean mongodb-atlas-cli?" What to do?

The error indicates that the **Homebrew** formula for `mongodb-community` is not available in your installation. This could happen because MongoDB was removed from Homebrew's core formula list. Instead, you can add MongoDB's official tap or use an alternative method.

Solution 1: Add MongoDB's Official Tap

MongoDB maintains its own Homebrew tap for installing `mongodb-community`. To add it:

1. Tap the MongoDB formula:

```
brew tap mongodb/brew
```

2. Install `mongodb-community`:

```
brew install mongodb-community
```

3. Start the service:

```
brew services start mongodb/brew/mongodb-community
```

Solution 2: Install MongoDB Community Edition Manually

If you face issues with Homebrew, you can manually install MongoDB Community Edition:

1. **Download MongoDB:** Visit the [MongoDB Download Center](#) and download the version for macOS.
2. **Install MongoDB:** Follow the instructions for installation provided on the download page.
3. **Run MongoDB:** Start the `mongod` service manually:

```
mongod --dbpath /your/data/directory
```

Solution 3: Use Docker

If you are comfortable with Docker, you can run MongoDB in a container:

1. Install Docker if not already installed:

```
brew install --cask docker
```

2. Pull the MongoDB image:

```
docker pull mongo
```

3. Run MongoDB:

```
docker run --name mongoddb -d -p 27017:27017 mongo
```

Solution 4: Use MongoDB Atlas

If local installation is unnecessary, you can use **MongoDB Atlas**, a cloud-based solution:

1. Sign up at [MongoDB Atlas](#).
 2. Create a free cluster and connect your application to it.
 3. Update your connection string in the application to the Atlas URI.
-

Verify Installation

After installing MongoDB using any of the above methods:

- Test if MongoDB is running:

```
mongo
```

- Confirm the connection in your application.
-

Error 4:

The above error was resolved. Now going back to the previous steps, on trying to test MongoDB Connection using "mongo --host localhost --port 27017", the following message is received in the terminal: "zsh: command not found: mongo"

Error 4: Solutions

The error `zsh: command not found: mongo` indicates that the `mongo` shell is not available in your system's `PATH`, or it is not installed. The `mongo` shell is a command-line client used to interact with MongoDB. Here's how to resolve this issue:

1. Verify MongoDB Installation

If you recently installed MongoDB via Homebrew or another method, ensure the installation includes the `mongo` shell. In recent MongoDB versions, the `mongo` shell is replaced with the **MongoDB Database Tools**. You might need to install it separately.

2. Install MongoDB Shell

If the `mongo` shell is missing, follow these steps to install it:

Option A: Using Homebrew

1. Install the MongoDB shell (`mongosh`):

```
brew install mongosh
```

2. Test the installation:

```
mongosh --host localhost --port 27017
```

Option B: Using MongoDB's Download Center

1. Visit the [MongoDB Database Tools Download Page](#).
 2. Download the tools for your operating system.
 3. Follow the installation instructions for your system.
-

3. Update the PATH Variable

If the `mongo` or `mongosh` binary is installed but not found in the terminal, add its location to your system's `PATH`:

1. Locate the installation directory of MongoDB:

```
which mongosh
```

2. Add the directory to your PATH in the ~/.zshrc file:

```
export PATH="/path/to/mongodb/bin:$PATH"
```

Replace /path/to/mongodb/bin with the actual directory path.

3. Reload the terminal:

```
source ~/.zshrc
```

4. Verify Connection

After installing the shell and ensuring it's in your PATH:

- Test the MongoDB connection:

```
mongosh --host localhost --port 27017
```

- If successful, you'll enter the MongoDB shell prompt.
-