Troubleshooting Guide

Common issues and their solutions.

🔴 MongoDB Connection Issues

Problem: "Error connecting to MongoDB"

```
Error connecting to MongoDB: MongooseServerSelectionError
```

Solutions:

1. **Check if MongoDB is running**

```bash
# Windows
net start MongoDB

# macOS/Linux
sudo systemctl status mongod
# or
brew services start mongodb-community
```

2. **Verify MongoDB URI**
   - Default: `mongodb://localhost:27017/GO`
   - Check port number (default is 27017)
   - Ensure database name is correct

3. **Check MongoDB logs**

```bash
# Linux/macOS
tail -f /var/log/mongodb/mongod.log

# Windows
# Check Event Viewer or MongoDB log directory
```

4. **Test connection manually**

```bash
  mongosh mongodb://localhost:27017/GO
```

🔴 Application Won't Start

Problem: "Cannot find module"

```
Error: Cannot find module 'express'
```

Solution:

```bash
npm install
```

Problem: "Port already in use"

```
Error: listen EADDRINUSE: address already in use :::3000
```

Solutions:

1. **Find and kill the process**

```bash
  # Windows
  netstat -ano | findstr :3000
  taskkill /PID <PID> /F

  # macOS/Linux
  lsof -ti:3000 | xargs kill -9
```

2. **Use a different port**

```bash
```

```
# In server.js
const PORT = 3001; // Change port number
```

🔴 API Request Errors

Problem: 404 Not Found for valid routes

```json
{
  "error": {
    "code": "NOT_FOUND",
    "message": "Route not found"
  }
}
```

Checklist:

- [ ] Is the server running?
- [ ] Are you using the correct base URL? (`http://localhost:3000`)
- [ ] Is the route spelled correctly?
- [ ] Is the HTTP method correct? (GET, POST, PATCH, DELETE)

Problem: 400 Bad Request - Invalid ID

```json
{
  "message": "Employee not found"
}
```

Solution:

- Ensure you're using a valid MongoDB ObjectId (24 hex characters)
- Example valid ID: `507f1f77bcf86cd799439011`
- Copy IDs from previous responses, don't make them up

Problem: 409 Conflict - Duplicate key

```json
```

```
{
  "message": "SSN already exists"
}
```

Solution:

- This is working as intended (unique constraint)
- Use a different unique value (SSN, department number, project number)
- Or delete the existing record first

Problem: 400 Bad Request - Invalid fields

```json
{
  "message": "Invalid fields in request body"
}
```

Solution: Check allowed fields for each resource:

- **Employees**: ssn, name, bdate, address, sex, salary
- **Departments**: number, name, locations
- **Projects**: number, name, location
- **Dependents**: employeeId, name, sex, birthDate, relationship

---

🔴 Pagination Issues

Problem: Page size warning in console

```
PageSize must be less than 100
```

Solution:

- Maximum page size is 100
- Request: `GET /employees?pageSize=50` (valid)
- Request: `GET /employees?pageSize=150` (invalid, will use default 10)

Problem: Empty results with pagination

Check:

1. Total number of records
2. Page number might be too high
3. Try: `GET /employees?pageNumber=1&pageSize=10`

🔴 Sorting Issues

Problem: "Invalid sort fields" error

```json
{
  "message": "Invalid sort fields"
}
```

Solution: Only these sort combinations are allowed:

- `sort=salary,bdate`
- `sort=-salary,-bdate`
- `sort=salary,-bdate`
- `sort=-salary,bdate`

NOT allowed:

- `sort=name` ❌
- `sort=address` ❌
- `sort=ssn,salary` ❌

🔴 Dependent Creation Issues

Problem: Cannot create dependent

```json
```

```
{
  "message": "Validation error..."
}
```

Solutions:

1. **Ensure employee exists first**

bash

```bash
# First create employee
POST /employees
{
  "ssn": "123-45-6789",
  "name": {"fname": "John", "lname": "Doe"},
  "salary": 50000,
  "sex": "Male"
}

# Copy the returned _id
# Then create dependent using that _id
POST /dependents
{
  "employeeId": "COPIED_ID_HERE",
  "name": "Jane Doe",
  "sex": "Female",
  "relationship": "Daughter"
}
```

2. **Check employeeId format**
   - Must be a valid MongoDB ObjectId
   - Must reference an existing employee

🔴 Validation Errors

Problem: "Employee validation failed"

json

```
{
  "message": "Employee validation failed: name.fname: Path `name.fname` is required."
}
```

Solution: Check all required fields:

Employee:

```json
{
  "ssn": "required",
  "name": {
    "fname": "required",
    "lname": "required"
  },
  "salary": "required",
  "sex": "required (Male/Female)"
}
```

Department:

```json
{
  "number": "required (unique)",
  "name": "required"
}
```

Project:

```json
{
  "number": "required (unique)",
  "name": "required"
}
```

Dependent:

```json
```

```
{
  "employeeId": "required (valid ObjectId)",
  "name": "required",
  "sex": "required (Male/Female)"
}
```

🔴 JSON Parsing Errors

Problem: "Unexpected token in JSON"

```json
{
  "message": "Unexpected token..."
}
```

Solutions:

1. **Ensure Content-Type header**

```
Content-Type: application/json
```

2. **Validate JSON syntax**
    - Use a JSON validator
    - Check for trailing commas
    - Ensure proper quotes (double quotes only)
    - Example valid JSON:

```json
  {
    "name": "test",
    "value": 123
  }
```

🔴 Performance Issues

Problem: Slow queries

Solutions:

1. **Check indexes**

```javascript
// In MongoDB shell
db.employees.getIndexes()
db.departments.getIndexes()
db.projects.getIndexes()
db.dependents.getIndexes()
```

2. **Use projection to limit fields**

```
GET /employees?project=name,salary
```

3. **Use pagination**

```
GET /employees?pageSize=20
```

🔴 Data Inconsistency

Problem: Stale or incorrect data

Solutions:

1. **Clear the database**

```javascript
```

```
// MongoDB shell
use GO
db.employees.deleteMany({})
db.departments.deleteMany({})
db.projects.deleteMany({})
db.dependents.deleteMany({})
```

2. **Drop and recreate indexes**

```javascript
// MongoDB shell
db.employees.dropIndexes()
db.departments.dropIndexes()
db.projects.dropIndexes()
db.dependents.dropIndexes()
```

Then restart the application to recreate indexes.

🔴 Import/Export Issues

Problem: "Cannot find module" after moving files

Solution: Check all import paths are correct:

```javascript
// Correct
import Employee from '../Models/Emps.model.js';

// Incorrect
import Employee from '../models/Emps.model.js'; // wrong case
import Employee from './Models/Emps.model.js'; // wrong relative path
```

🛠️ Debugging Tips

Enable Detailed Mongoose Logging

```javascript
```

```
// In server.js
import mongoose from 'mongoose';
mongoose.set('debug', true);
```

Check Request Body

```javascript
// In any controller
console.log('Request body:', req.body);
console.log('Request params:', req.params);
console.log('Request query:', req.query);
```

Verify Database Records

```javascript
// MongoDB shell
use GO
db.employees.find().pretty()
db.departments.find().pretty()
```

Test with curl

```bash
# Test if server is running
curl http://localhost:3000/employees

# Test POST with data
curl -X POST http://localhost:3000/employees \
  -H "Content-Type: application/json" \
  -d '{"ssn":"123-45-6789","name":{"fname":"Test","lname":"User"},"salary":50000,"sex":"Male"}'
```

📞 Still Having Issues?

1. **Check server console** for error messages
2. **Check MongoDB logs** for database errors
3. **Use MongoDB Compass** to verify data

4. **Test with Postman** for better error messages
5. **Clear node_modules and reinstall**

```bash
rm -rf node_modules package-lock.json
npm install
```

✅ Health Check Endpoints (Optional Enhancement)

Add these to App.js for debugging:

```javascript
```

```
// Basic health check
app.get('/health', (req, res) => {
  res.status(200).json({
    status: 'OK',
    timestamp: new Date().toISOString(),
    uptime: process.uptime()
  });
});

// Database health check
app.get('/health/db', async (req, res) => {
  try {
    const dbState = mongoose.connection.readyState;
    const states = {
      0: 'disconnected',
      1: 'connected',
      2: 'connecting',
      3: 'disconnecting'
    };

    res.status(dbState === 1 ? 200 : 503).json({
      database: states[dbState],
      timestamp: new Date().toISOString()
    });
  } catch (error) {
    res.status(503).json({ error: error.message });
  }
});
```