OpenBank Application Setup Guide

Prerequisites

Before you begin, ensure you have the following installed:

- [Node.js](https://nodejs.org/) (v14 or higher)
- [MongoDB](https://www.mongodb.com/try/download/community)
- [VS Code](https://code.visualstudio.com/) (or any code editor)
- Git (optional, for cloning)

Step-by-Step Setup Guide

1. Download and Extract the Project

- ✓ Download the OpenBank project folder
- ✓ Extract it to your preferred location
- ✓ Open VS Code and open the extracted folder

2. Open Three Terminals in VS Code

In VS Code, open three separate terminals:

- ❖ Terminal 1: Frontend
- ❖ Terminal 2: Backend
- ❖ Terminal 3: MongoDB

3. MongoDB Setup (Terminal 1)

```
# Navigate to MongoDB bin directory (Windows example)
cd "C:\Program Files\MongoDB\Server\8.2\bin"
# Start MongoDB server
mongod
# Leave this terminal running
```

Note for macOS/Linux users:

```
 # Usually MongoDB runs as a service
sudo systemctl start mongod
# or
mongod --config /usr/local/etc/mongod.conf
```

4. Backend Setup (Terminal 2)

```
 # Navigate to backend directory
cd backend
# Check if package.json exists
ls package.json
# Install all dependencies
npm install

# Install specific required packages
npm install express mongoose dotenv cors bcrypt jsonwebtoken validator nodemon --s
ave
# If you need to install bcryptjs separately
npm install bcryptjs --save
# For development dependencies
npm install --save-dev nodemon
# Verify node_modules exists
ls node_modules
# Create a .env file if it doesn't exist (ask project owner for variables)
# Add your environment variables (database URL, JWT secret, etc.)

# Start the backend server
npm start
# or if using nodemon
nodemon server.js
```

5. Frontend Setup (Terminal 3)

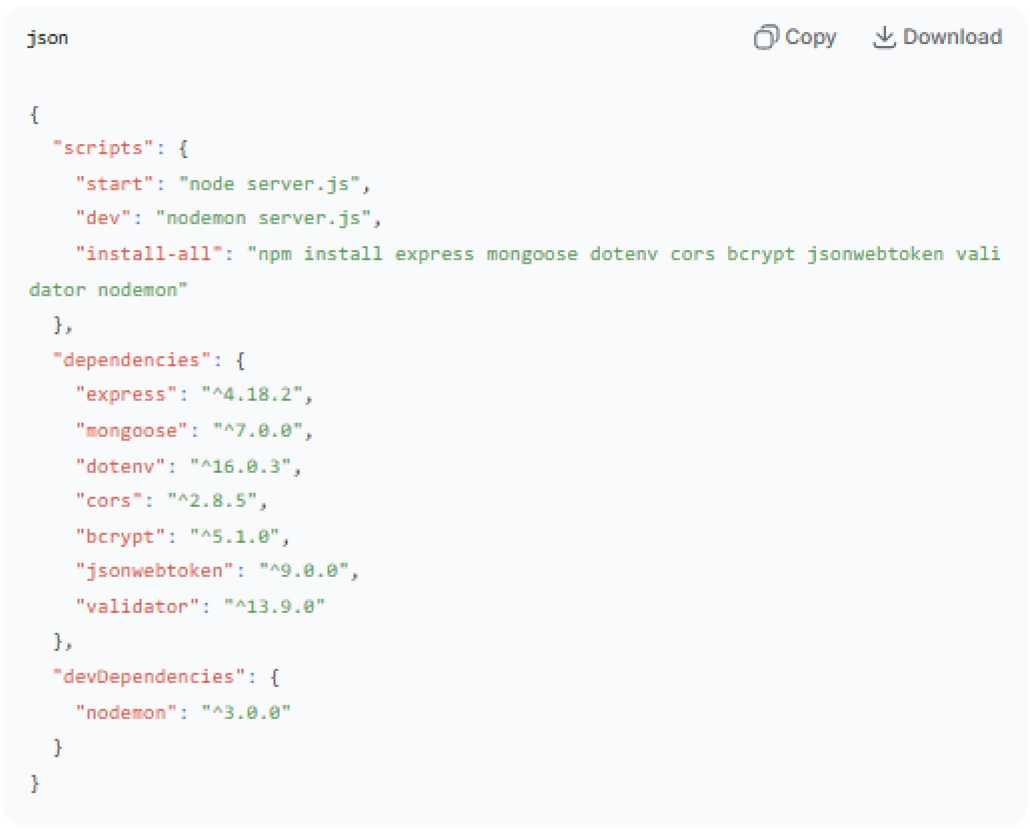
```
 # Navigate to frontend directory
cd frontend
# Check if package.json exists
ls package.json
# Install dependencies
npm install
# Verify node_modules exists
ls node_modules

# Start the frontend development server
npm start
# or
```

```
npm run dev
```

Alternative Installation Method (Using package.json)

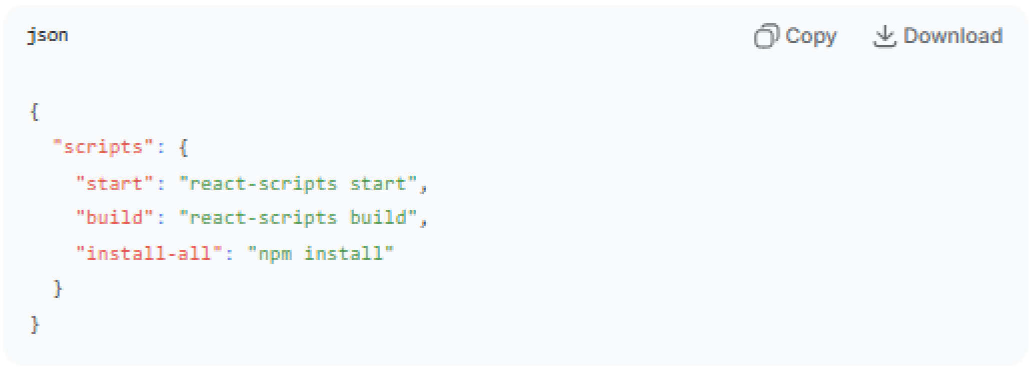
If you have a complete package.json file, you can simplify installation:

For Backend:



```json
{
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "install-all": "npm install express mongoose dotenv cors bcrypt jsonwebtoken validator nodemon"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mongoose": "^7.0.0",
    "dotenv": "^16.0.3",
    "cors": "^2.8.5",
    "bcrypt": "^5.1.0",
    "jsonwebtoken": "^9.0.0",
    "validator": "^13.9.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.0"
  }
}
```

For Frontend:



```json
{
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "install-all": "npm install"
  }
}
```

Troubleshooting Common Issues

1. MongoDB Connection Issues

```bash
# Check if MongoDB is running
mongo --eval "db.adminCommand('ping')"

# If not running, start MongoDB service
sudo service mongod start  # Linux/Mac
# or
net start MongoDB  # Windows
```

2. Port Conflicts

- Backend typically runs on: `http://localhost:5000
- Frontend typically runs on: `http://localhost:3000
- MongoDB runs on: `mongodb://localhost:27017

3. CORS Issues

Ensure your backend has CORS configured. If using the provided code snippet:

```javascript
const corsOptions = {
    origin: 'http://localhost:3000', // Your frontend URL
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
    allowedHeaders: ['Content-Type', 'Authorization'],
    credentials: true
};
app.use(cors(corsOptions));
```

4. Missing node_modules

```
# Delete node_modules and package-lock.json
rm -rf node_modules package-lock.json  # Linux/Mac
# or
rd /s /q node_modules & del package-lock.json  # Windows

# Reinstall
npm install
```

Verification Steps

1. Check MongoDB: Should be running on port 27017
2. Check Backend: Should respond at http://localhost:5000
3. Check Frontend: Should open at http://localhost:3000
4. Check Connection: Frontend should successfully communicate with backend

Startup Order (CRITICAL)

Always start services in this order:

text Copy Download

```text
1. MongoDB (Terminal 3) - Leave running
2. Backend (Terminal 2) - Wait for "Server running" message
3. Frontend (Terminal 1) - Will open browser automatically
```

Environment Variables

env

```env
PORT=5000
MONGODB_URI=mongodb://localhost:27017/openbank
JWT_SECRET=your_jwt_secret_key_here
```

Generating JWT Secret Key

Method 1: Using Node.js (Recommended)

In Backend Terminal/Command Prompt:

```
# Run Node.js command to generate secret
node -e "console.log(require('crypto').randomBytes(64).toString('hex'))"
```

Alternative Node.js methods:

```
# Method A: One-liner
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Method B: Using util
node -e "console.log(require('crypto').randomBytes(48).toString('hex'))"

# Method C: Generate 256-bit key
node -e "console.log(require('crypto').randomBytes(256/8).toString('hex'))"
```

Method 2: Using OpenSSL (Mac/Linux/Windows with Git Bash)

```
# Generate 64-character hex string
openssl rand -hex 32

# Generate 256-bit key in base64
openssl rand -base64 32

# Generate stronger 512-bit key
openssl rand -hex 64
```

Method 3: Create a Simple Node Script

Create generateSecret.js

```
const crypto = require('crypto');

console.log('Hex (64 chars):', crypto.randomBytes(32).toString('hex'));
console.log('Base64 (44 chars):', crypto.randomBytes(32).toString('base64'));
console.log('Hex (128 chars):', crypto.randomBytes(64).toString('hex'));
```

Then run: on terminal

```
node generateSecret.js
```

For OpenBank Project:

1. Generate the Secret

```
# Run in your backend terminal
node -e "console.log('JWT_SECRET=' + require('crypto').randomBytes(64).toString('he
x'))"
```

2. Add to Your .env File

Copy the output and add it to your .env file:

```
JWT_SECRET=your_generated_secret_here_64_characters_long
```

Postman Collection Setup for OpenBank

1. Locate the Postman Collection File

```
/openbank-folder/postman_collection.json
```

2. Import into Postman

```
1. Open Postman desktop application
2. Click "Import" button (top-left corner)
3. Click "Upload Files" or drag & drop
4. Navigate to: openbank-folder/postman_collection.json
5. Click "Import"
```

3. Set Up Environment Variables

Create a new environment in Postman called "OpenBank Development":

1. Click "Environments" (left sidebar)
2. Click "+" to create new environment
3. Name it: `OpenBank Development`
4. Add these variables:

Variable	Initial Value	Current Value
`baseUrl`	`http://localhost:5000`	`http://localhost:5000`
`authToken`	(leave empty)	(will auto-fill)
`userId`	(leave empty)	(will auto-fill)

5. Click "Save"

4. Select Environment

```
- Click the environment dropdown (top-right)
- Select "OpenBank Development"
```

Collection Structure Overview

The collection includes:

1. Authentication Routes
2. User Routes
3. Account Routes
4. Transaction Routes

Running Tests - Step by Step

Prerequisites

```
Ensure these are running:
1. MongoDB: mongod
2. Backend: npm start (on port 5000)
3. Frontend: npm start (optional for Postman)
```

Step 1: Register a Test User

```
1. Open "Authentication" folder
2. Select "Register User"
3. Click "Body" tab, ensure raw JSON is selected
4. Example JSON:
{
    "name": "Test User",
    "email": "test@example.com",
    "password": "password123",
    "confirmPassword": "password123"
}
5. Click "Send"
6. Should receive 201 Created response
```

Step 2: Login to Get Token

```
1. Select "Login User"
2. Use same credentials:
{
    "email": "test@example.com",
    "password": "password123"
}
3. Click "Send"
4. Check "Tests" tab - token is automatically saved!
5. Verify token saved:
   - Click eye icon (top-right) next to environment
   - Check "authToken" has value
```

Step 3: Run Automated Test Collection

```
1. Click "Runner" button (top-left rocket icon)
2. Select "OpenBank Collection"
3. Select "OpenBank Development" environment
4. Click "Start Run"
5. Watch tests execute in sequence
```

Step 4: Individual Endpoint Testing

```
For each endpoint:
1. Select request in collection
2. Click "Send"
3. Check response status (should be 200/201)
4. View response body for data
```

Need Help?

If Postman tests fail:

1. Check backend console for errors
2. Verify MongoDB is running
3. Check .env file has correct JWT_SECRET
4. Clear Postman cache: File → Settings → Clear cache and restart
5. Update collection if API changed: Right-click collection → "Update from file"