Copy Trading App: End-to-End Guide for New Developers

This guide takes you **from zero to deployed**—no prior technical knowledge needed. You'll set up your development environment, write and organize **all** the backend and frontend code (with full folder structures and copy-and-paste files), configure CI/CD, deploy to the cloud, and learn why each step matters. At the end, you'll have a working Copy Trading App with user registration, login, trade CRUD, error/performance monitoring with Sentry, and automated testing/deployment pipelines.

Table of Contents

Prerequisites

Accounts & Services

- **GitHub** (code hosting & CI)
- **MongoDB Atlas** (cloud database)
- **Render** (free Node.js hosting)
- **Vercel** (free React hosting)
- **Sentry** (error & performance monitoring)
- **Postman** (API testing)

Local Tools (Developer)

- Node.js & npm (v18+ recommended)
- Git
- VS Code or another code editor

Repository & Folder Structure

Create a parent folder and two Git repos:

```
mkdir copy-trading-app && cd copy-trading-app
```

Backend (backend)

```
backend/
├── .env             # Local env vars
├── .env.example
├── instrument.js
├── package.json
├── testMongo.js
├── src/
│   ├── server.js
│   ├── app.js
│   ├── models/
│   │   └── Trade.js
│   ├── controllers/
│   │   └── authController.js
│   ├── middleware/
│   │   └── authMiddleware.js
│   └── routes/
│       ├── authRoutes.js
│       └── tradeRoutes.js
└── .github/
    └── workflows/
        └── backend-ci.yml
```

Frontend (frontend)

```
frontend/
├── .env.local
├── package.json
├── public/
│   └── index.html
└── src/
    ├── api.js
    ├── App.js
    ├── App.css
    ├── index.js
    ├── TradeApp.js
    ├── components/
    │   └── PrivateRoute.js
    └── pages/
        ├── Register.js
        └── Login.js
```

Backend Setup

3.1 Environment Variables

Create `backend/.env` (never commit real secrets):

```
# MongoDB
MONGO_URI=mongodb+srv://<user>:<pass>@cluster0.mongodb.net/<dbname>?
retryWrites=true&w=majority

# JWT secret
JWT_SECRET=yourSuperSecretKey

# Sentry DSN
SENTRY_DSN=https://<publicKey>@o<org>.ingest.sentry.io/<projectId>
```

Also add `backend/.env.example`:

```
MONGO_URI=your_mongo_uri
JWT_SECRET=your_jwt_secret
SENTRY_DSN=your_sentry_dsn
```

3.2 Dependencies & `package.json`

In `backend/`, install:

```
npm init -y
npm install express mongoose cors dotenv bcryptjs jsonwebtoken @sentry/node
@sentry/tracing
npm install --save-dev nodemon
```

`backend/package.json` highlights:

```
{
  "name": "copy-trading-backend",
  "version": "1.0.0",
  "scripts": {
    "start": "node src/server.js",
    "dev": "nodemon src/server.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mongoose": "^7.0.0",
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "bcryptjs": "^2.4.3",
    "jsonwebtoken": "^9.0.0",
    "@sentry/node": "^7.0.0",
    "@sentry/tracing": "^7.0.0"
  }
}
```

3.3 Sentry Instrumentation (`instrument.js`)

Who: Developer
Where: `backend/instrument.js`

```
// Load env vars
require('dotenv').config();

// Initialize Sentry
const Sentry = require('@sentry/node');
Sentry.init({
  dsn: process.env.SENTRY_DSN,
  tracesSampleRate: 1.0,
});

module.exports = Sentry;
```

3.4 Express App Setup (`src/app.js`)

Who: Developer
Where: `backend/src/app.js`

```
const express = require('express');
const cors    = require('cors');
const Sentry  = require('@sentry/node');

const app = express();

// CORS for React dev server
app.use(cors({
  origin: 'http://localhost:3000',
  credentials: true
}));

// Sentry handlers
```

```
app.use(Sentry.Handlers.requestHandler());
app.use(Sentry.Handlers.tracingHandler());

// JSON parsing
app.use(express.json());

// Health route
app.get('/', (req,res) => res.send('Backend is working!'));

// API routes (mounted later)

module.exports = app;
```

3.5 Server Entry (`src/server.js`)

Who: Developer
Where: `backend/src/server.js`

```
// Initialize Sentry, load env
require('dotenv').config();
require('../instrument');

const app     = require('./app');
const mongoose = require('mongoose');
const Sentry   = require('@sentry/node');

// Mount routes
app.use('/api/auth',  require('./routes/authRoutes'));
app.use('/api/trades', require('./routes/tradeRoutes'));

// Debug route for Sentry
app.get('/debug-sentry', (req,res,next) => next(new Error('Sentry OK')));

// Sentry error handler (must come after routes)
app.use(Sentry.Handlers.errorHandler());

// Final Express error handler
app.use((err,req,res,next) => {
  console.error(err);
  res.status(err.status || 500).json({ error: err.message });
});

// Connect to MongoDB & launch
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true, useUnifiedTopology: true
})
.then(() => app.listen(process.env.PORT||5000, () =>
  console.log('� Server running')
))
.catch(err => console.error('DB connection error', err));
```

3.6 Data Model (`src/models/Trade.js`)

Who: Developer
Where: `backend/src/models/Trade.js`

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

const tradeSchema = new mongoose.Schema({
  symbol:     { type: String, required: true },
  action:     {
    type: String,
    enum: ['buy','sell'],
    required: true,
    lowercase: true
  },
  price:      { type: Number, required: true },
  userId:     { type: String, required: true },
  createdAt:  { type: Date,   default: Date.now }
});

module.exports = mongoose.model('Trade', tradeSchema);
```

3.7 Authentication

3.7.1 Controller (`src/controllers/authController.js`)

```
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');
let users = []; // demo

exports.registerUser = async (req,res) => {
  const { email, password } = req.body;
  if (users.find(u=>u.email===email))
    return res.status(400).json({ error:'User exists' });
  const hash = await bcrypt.hash(password,10);
  users.push({ id:users.length+1, email, password:hash });
  res.status(201).json({ message:'Registered' });
};

exports.loginUser = async (req,res) => {
  const { email,password } = req.body;
  const user = users.find(u=>u.email===email);
  if (!user || !(await bcrypt.compare(password,user.password)))
    return res.status(401).json({ error:'Invalid credentials' });
  const token = jwt.sign({ id:user.id }, process.env.JWT_SECRET, { expiresIn:'1h' });
  res.json({ token });
};
```

3.7.2 Routes (`src/routes/authRoutes.js`)

```
const express = require('express');
const { registerUser, loginUser } = require('../controllers/authController');
const router = express.Router();

router.post('/register', registerUser);
router.post('/login',    loginUser);

module.exports = router;
```

3.7.3 Middleware (`src/middleware/authMiddleware.js`)

```
const jwt = require('jsonwebtoken');

module.exports = (req,res,next) => {
  const header = req.headers.authorization;
  if (!header) return res.status(401).json({ error:'No token' });
  const token = header.split(' ')[1];
  jwt.verify(token, process.env.JWT_SECRET, (err,decoded) => {
    if(err) return res.status(401).json({ error:'Invalid token' });
    req.userId = String(decoded.id);
    next();
  });
};
```

3.8 Trade CRUD Routes (`src/routes/tradeRoutes.js`)

```
const express = require('express');
const auth    = require('../middleware/authMiddleware');
const Trade   = require('../models/Trade');
const router  = express.Router();

// List
router.get('/', async (req,res,next) => {
  try { res.json(await Trade.find().sort({createdAt:-1})); }
  catch(e){ next(e); }
});

// Create
router.post('/', auth, async (req,res,next) => {
  try {
    const {symbol,action,price} = req.body;
    res.status(201).json(
      await Trade.create({ symbol, action, price, userId:req.userId })
    );
  } catch(e){ next(e); }
});

// Update
router.put('/:id', auth, async (req,res,next) => {
  try {
    const t = await Trade.findById(req.params.id);
    if(!t) return res.status(404).json({ error:'Not found' });
    if(t.userId!==req.userId) return res.status(403).json({ error:'Forbidden' });
    Object.assign(t, req.body);
    await t.save();
    res.json({ message:'Updated', trade:t });
  } catch(e){ next(e); }
});

// Delete
router.delete('/:id', auth, async (req,res,next) => {
  try {
    const t = await Trade.findById(req.params.id);
    if(!t) return res.status(404).json({ error:'Not found' });
    if(t.userId!==req.userId) return res.status(403).json({ error:'Forbidden' });
    await t.deleteOne();
    res.json({ message:'Deleted' });
  } catch(e){ next(e); }
});

module.exports = router;
```

3.9 `testMongo.js` Connectivity Check

```
// backend/testMongo.js
require('dotenv').config();
const { MongoClient, ServerApiVersion } = require('mongodb');

const uri = process.env.MONGO_URI;
const client = new MongoClient(uri, { serverApi: ServerApiVersion.v1 });

async function run() {
  try {
    await client.connect();
    await client.db("admin").command({ ping: 1 });
    console.log("□ MongoDB connected!");
  } catch (err) {
    console.error("MongoDB error:", err);
  } finally {
    await client.close();
  }
}
run();
```

Run locally: `node testMongo.js`

3.10 CI Pipeline (`.github/workflows/backend-ci.yml`)

```
name: Backend CI

on: [push]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node
        uses: actions/setup-node@v3
        with: node-version: '18'
      - name: Install deps
        run: npm install
        working-directory: backend
      - name: Lint & Test
        run: |
          npm run lint || true
          npm test || true
        working-directory: backend
```

Frontend Setup

4.1 Environment Variables (`.env.local`)

```
REACT_APP_API_URL=http://localhost:5000
REACT_APP_SENTRY_DSN=https://<publicKey>@o<org>.ingest.sentry.io/<projectId>
```

4.2 Dependencies & `package.json`

```
cd frontend
npm install axios react-router-dom @sentry/react @sentry/tracing
```

Key entries in `frontend/package.json` :

```
"dependencies": {
  "axios": "^1.2.0",
  "react-router-dom": "^6.3.0",
  "@sentry/react": "^7.0.0",
  "@sentry/tracing": "^7.0.0"
}
```

4.3 API Helper (`src/api.js`)

```
import axios from 'axios';
const api = axios.create({ baseURL: process.env.REACT_APP_API_URL });
api.interceptors.request.use(cfg => {
  const token = localStorage.getItem('token');
  if(token) cfg.headers.Authorization = `Bearer ${token}`;
  return cfg;
});
export default api;
```

4.4 Private Route Wrapper (`src/components/PrivateRoute.js`)

```
import React from 'react';
import { Navigate } from 'react-router-dom';

export default function PrivateRoute({ children }) {
  return localStorage.getItem('token') ? children : <Navigate to="/login" />;
}
```

4.5 Pages

4.5.1 Register (`src/pages/Register.js`)

```
import React, { useState } from 'react';
import api from '../api';
import { useNavigate, Link } from 'react-router-dom';

export default function Register() {
  const [form, setForm] = useState({ email:'', password:'' });
  const [error, setError] = useState('');
  const navigate = useNavigate();

  const onChange = e => setForm(f=>({...f,[e.target.name]:e.target.value}));
  const onSubmit = async e => {
    e.preventDefault();
    try {
      await api.post('/api/auth/register', form);
      navigate('/login');
    } catch(err) {
      setError(err.response?.data?.error || 'Registration failed');
    }
  };

  return (
    <div>
      <h2>Register</h2>
      {error && <p style={{color:'red'}}>{error}</p>}
      <form onSubmit={onSubmit}>
        <input name="email" type="email" placeholder="Email"
          value={form.email} onChange={onChange} required /><br/>
        <input name="password" type="password" placeholder="Password"
          value={form.password} onChange={onChange} required /><br/>
        <button type="submit">Register</button>
      </form>
      <p>Have an account? <Link to="/login">Login</Link></p>
    </div>
  );
}
```

4.5.2 Login (`src/pages/Login.js`)

```
import React, { useState } from 'react';
import api from '../api';
import { useNavigate, Link } from 'react-router-dom';

export default function Login() {
  const [form, setForm] = useState({ email:'', password:'' });
  const [error, setError] = useState('');
  const navigate = useNavigate();

  const onChange = e => setForm(f=>({...f,[e.target.name]:e.target.value}));
  const onSubmit = async e => {
    e.preventDefault();
    try {
      const res = await api.post('/api/auth/login', form);
      localStorage.setItem('token', res.data.token);
      navigate('/');
    } catch(err) {
      setError(err.response?.data?.error || 'Login failed');
    }
  };

  return (
    <div>
      <h2>Login</h2>
      {error && <p style={{color:'red'}}>{error}</p>}
      <form onSubmit={onSubmit}>
        <input name="email" type="email" placeholder="Email"
          value={form.email} onChange={onChange} required /><br/>
        <input name="password" type="password" placeholder="Password"
          value={form.password} onChange={onChange} required /><br/>
        <button type="submit">Login</button>
      </form>
      <p>New user? <Link to="/register">Register</Link></p>
    </div>
  );
}
```

4.5.3 Trade App (`src/TradeApp.js`)

```
import React, { useEffect, useState } from 'react';
import api from './api';

export default function TradeApp() {
  const [trades, setTrades] = useState([]);
  const [newTrade, setNewTrade] = useState({ symbol:'', action:'', price:'' });
  const [error, setError] = useState('');

  useEffect(() => { fetchTrades(); }, []);

  const fetchTrades = async () => {
    try { const res = await api.get('/api/trades'); setTrades(res.data); }
    catch { setError('Fetch failed'); }
  };
  const handleChange = e => setNewTrade(t=>({...t,[e.target.name]:e.target.value}));
  const handleAdd = async e => {
    e.preventDefault();
    try {
      const res = await api.post('/api/trades', newTrade);
      setTrades(t=>[...t,res.data]);
      setNewTrade({ symbol:'', action:'', price:'' });
    } catch { setError('Add failed'); }
  };

  const handleDelete = async id => {
    try {
      await api.delete(`/api/trades/${id}`);
      setTrades(t=>t.filter(x=>x._id!==id));
    } catch { setError('Delete failed'); }
  };

  return (
    <div>
      <h2>Trades</h2>
      {error && <p style={{color:'red'}}>{error}</p>}
      <form onSubmit={handleAdd}>
        <input name="symbol" placeholder="Symbol"
          value={newTrade.symbol} onChange={handleChange} required/>
        <input name="action" placeholder="Action"
          value={newTrade.action} onChange={handleChange} required/>
        <input name="price" type="number" placeholder="Price"
          value={newTrade.price} onChange={handleChange} required/>
        <button type="submit">Add</button>
      </form>
      <ul>
        {trades.map(t => (
          <li key={t._id}>
            {t.symbol} - {t.action} @ ${t.price}{' '}
            <button onClick={()=>handleDelete(t._id)}>Delete</button>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

4.6 Root App (`src/App.js`)

```
import React from 'react';
import { Routes, Route, Link } from 'react-router-dom';
import PrivateRoute from './components/PrivateRoute';
import Register from './pages/Register';
import Login    from './pages/Login';
import TradeApp from './TradeApp';
```

```
export default function App() {
  return (
    <div style={{ padding: 20 }}>
      <nav>
        <Link to="/">Home</Link> |{' '}
        <Link to="/register">Register</Link> |{' '}
        <Link to="/login">Login</Link>
      </nav>
      <Routes>
        <Route path="/" element={
          <PrivateRoute><TradeApp/></PrivateRoute>
        }/>
        <Route path="/register" element={<Register/>}/>
        <Route path="/login"    element={<Login/>}/>
      </Routes>
    </div>
  );
}
```

4.7 Entry Point (`src/index.js`)

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import './index.css';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(<BrowserRouter><App/></BrowserRouter>);
```

4.8 CI Pipeline (`.github/workflows/frontend-ci.yml`)

```
name: Frontend CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node
        uses: actions/setup-node@v3
        with: node-version: '18'
      - name: Install & Test
        run: |
          npm ci
          npm test || true
        working-directory: frontend
```

---

5. Deployments

5.1 MongoDB Atlas

- Create a free cluster, whitelist your IP, get connection string, update `MONGO_URI`.

5.2 Render (Backend)

- New Web Service → Connect GitHub → select `backend` → Build: `npm install` → Start: `npm start` → set ENV vars → Deploy.

5.3 Vercel (Frontend)

- New Project → Connect GitHub → select `frontend` → set ENV vars (`REACT_APP_API_URL`, `REACT_APP_SENTRY_DSN`) → Deploy.

---

6. Testing & Usage

6.1 Postman Collection

- Import endpoints (register, login, trades CRUD, health, debug-sentry).

6.2 Debugging with Sentry

- Hit `/debug-sentry` → see error in Sentry Issues.

---

7. When to Commit & Push

- **After** each logical unit: env files, instrumentation, models, controllers, routes, UI pages
- Commit message convention:

```
feat: add auth middleware
fix: correct trade delete endpoint
chore: configure CI pipeline
```

- Push to `main` (or protection branches, PRs, etc.)

---

8. Future Enhancements

- Replace in-memory users with real User model
- Add “Edit Trade” UI
- Real-time updates (WebSockets)
- Rate limiting & input validation
- Production-grade logging & monitoring
- CI/CD tests & coverage

---

9. Problems Faced & Solutions

Issue	Why	Solution
DNS error on Mongo SRV	Missing MongoDB SRV record	Switched to full connection URI from Atlas

Sentry `requestHandler()` undefined	Incorrect SDK version/integration	Installed `@sentry/node@7.x` and used built-in `Handlers`
CORS blocks	No CORS middleware	Added `cors()` before routes
React missing `key` warning	List items without `key` prop	Used `trade._id` as `key`
Delete route 500 (undefined ID)	Frontend sent wrong ID	Updated `handleDelete(id)` to use `trade._id`
Mongoose `Cast to ObjectId failed`	`userId` schema expected ObjectId but got number	Changed `userId` type to `String`
Mongoose enum validation for uppercase	Input `'SELL'` didn’t match lowercase enum	Added `lowercase: true` to schema
401 Unauthorized on protected endpoints	No JWT sent	Stored token in `localStorage` and added Axios interceptor
Blank page (frontend fetch blocked)	CORS + missing fallback UI	Fixed CORS, added error messages in UI

Each of these fixes was **committed** immediately with descriptive messages and tested locally before pushing.

Congratulations! You now have a **full-stack**, **cloud-deployed**, **monitored**, and **tested** Copy Trading App—built step by step from the ground up.