🚀 Catalyst Markets - Complete Startup Guide

⚡ Quick Start (5 Minutes)

Step 1: Prerequisites Check

```bash
# Verify installations
node --version    # Should be 18+
docker --version  # Should be 24+
git --version
```

Step 2: Start Docker Services

```bash
# Navigate to project root
cd catalyst-markets

# Start all services
docker-compose up -d

# Verify services are running
docker-compose ps

# You should see:
# ✅ catalyst-postgres    (port 5432)
# ✅ catalyst-redis       (port 6379)
# ✅ catalyst-adminer     (port 8080)
# ✅ catalyst-redis-commander (port 8081)
```

Troubleshooting:

```bash
# If port 5432 is already in use:
lsof -i :5432
kill -9 <PID>

# OR change port in docker-compose.yml:
postgres:
  ports:
    - "5433:5432"  # Use different port
```

Step 3: Setup Backend

```bash
cd backend

# Copy environment file
cp .env.example .env.development

# Install dependencies
npm install

# Generate Prisma client
npx prisma generate

# Run database migrations
npx prisma migrate dev --name init

# Seed database with sample data
npx prisma db seed
```

Expected Output:

```
🌱 Starting database seed...
📈 Seeding NSE stocks...
✅ Seeded 15 NSE stocks
📈 Seeding NASDAQ stocks...
✅ Seeded 10 NASDAQ stocks
💼 Seeding sample IPOs...
✅ Seeded 3 sample IPOs
📊 Seeding Fear & Greed history...
✅ Seeded 30 days of Fear & Greed history
✅ Database seed completed successfully!
```

Step 4: Setup Frontend

```bash
cd ../frontend

# Copy environment file
cp .env.example .env.local

# Install dependencies
npm install
```

Step 5: Start Development Servers

Terminal 1 - Backend:

```bash
cd backend
npm run dev
```

Expected Output:

```
✅ Database connected
✅ Redis connected
🚀 Catalyst Markets v0.1.0 running on port 3001
📝 Environment: development
🏥 Health check: http://localhost:3001/health
🔗 API: http://localhost:3001/api/v1
```

Terminal 2 - Frontend:

```bash
cd frontend
npm run dev
```

Expected Output:

```
▲ Next.js 14.1.0
- Local:        http://localhost:3000
- Ready in 2.3s
```

🎨 Visual Access Points

1. Main Application

URL: http://localhost:3000

What you'll see:

```
┌─────────────────────────────────────────┐
│ 🏠 Catalyst Markets Dashboard          │
├─────────────────────────────────────────┤
│                                         │
│ 📊 Stats Grid:                          │
│   ├─ Total Stocks: 250+                 │
│   ├─ Active IPOs: 3                     │
│   ├─ Market Status: Open                │
│   └─ Avg Response: 120ms                │
│                              │
│ 📈 Top Stocks (Clickable Cards):      │
│ ├─ RELIANCE - ₹2450.75 (+0.5%)       │
│ ├─ TCS - ₹3250.00 (+1.2%)            │
│ └─ ... (5 stocks)                    │
│                              │
│ 💼 Upcoming IPOs (3 cards):          │
│ ├─ TechInnovate Solutions            │
│ │   Opens: Tomorrow                  │
│ │   GMP: +25.5%                      │
│ │   Subscription: 0x (Not started)   │
│ └─ ...                               │
│                              │
│ 🎯 Features:                         │
│ ├─ Real-Time Pricing                 │
│ ├─ IPO Intelligence                  │
│ └─ Smart Screeners                   │
└──────────────────────────────────────┘
```

2. Stocks Page

URL: http://localhost:3000/stocks

What you'll see:

- Full table with 25 stocks (15 NSE + 10 NASDAQ)
- Search functionality (try searching "REL" or "APPLE")
- Exchange filter dropdown
- Real-time price updates
- Color-coded gains (green) / losses (red)

3. IPOs Page

URL: http://localhost:3000/ipos

What you'll see:

- 3 IPO cards with details
- Filter tabs: All / Upcoming / Open
- GMP percentage badges
- Subscription meters
- "Apply/Neutral/Avoid" verdicts
- Days remaining countdown

4. Database GUI (Adminer)

URL: http://localhost:8080

Login:

- System: PostgreSQL
- Server: postgres
- Username: catalyst_user
- Password: dev_password_123
- Database: catalyst_dev

What you'll see:

- Tables: stocks, ipos, feargreedhistory
- Can browse all seeded data
- Run SQL queries
- View relationships

5. Redis GUI (Redis Commander)

URL: http://localhost:8081

What you'll see:

- Cached stock prices (keys like `stock:iex:RELIANCE`)
- IPO list cache
- TTL values showing time remaining

6. Prisma Studio

Command:

```bash
cd backend
npx prisma studio
```

URL: http://localhost:5555

What you'll see:

- Visual database browser
- Edit records directly
- Add new IPOs
- View relationships

🧪 Test API Endpoints

Health Check

bash

```bash
curl http://localhost:3001/health
```

Expected Response:

json

```json
{
  "status": "ok",
  "timestamp": "2026-02-07T...",
  "uptime": 45.234,
  "environment": "development",
  "version": "0.1.0",
  "checks": {
    "database": "healthy",
    "redis": "healthy"
  }
}
```

Get Stocks

bash

```bash
curl http://localhost:3001/api/v1/stocks?limit=5
```

Expected Response:

json

```
{
  "data": [
    {
      "symbol": "RELIANCE",
      "name": "Reliance Industries Ltd",
      "exchange": "NSE",
      "currentPrice": 0,
      "dayChangePercent": 0,
      ...
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 5,
    "total": 25
  }
}
```

Get IPOs

```bash
curl http://localhost:3001/api/v1/ipos/upcoming
```

Get IPO Advisor Verdict

```bash
curl -X POST http://localhost:3001/api/v1/ipos/1/advisor
```

Expected Response:

```json
{
  "data": {
    "verdict": "APPLY",
    "score": 4,
    "reasons": [
      "Strong GMP of 25.5%",
      "QIB subscription 0x",
      ...
    ],
    "risks": []
  }
}
```

📊 Visual Features Demonstration

Feature 1: Real-Time Stock Updates

1. Open http://localhost:3000
2. Note the stock prices (currently showing 0)
3. The prices would update in real app when you add API keys

To test with real data:

- Add `IEX_CLOUD_API_KEY` to `backend/.env.development`
- Restart backend
- Prices will update every 15 seconds

Feature 2: IPO GMP Tracking

1. Go to http://localhost:3000/ipos
2. See 3 IPO cards with GMP percentages
3. Filter by "Open Now" or "Upcoming"
4. Click "View Details" button

Feature 3: Search & Filter

1. Go to http://localhost:3000/stocks
2. Type "REL" in search box
3. See filtered results (RELIANCE)
4. Change exchange filter to "NSE"
5. See only NSE stocks

Feature 4: Visual Indicators

Color Coding:

- 🟢 Green = Positive change
- 🔴 Red = Negative change
- 🟡 Yellow = Neutral verdict

Badges:

- "✓ Apply" = Green background
- "○ Neutral" = Yellow background

- "✗ Avoid" = Red background

🎬 Complete Demo Flow

Scenario: New User Exploring the App

Step 1: Dashboard (30 seconds)

```
1. Open http://localhost:3000
2. See overview stats
3. Scroll through top 5 stocks
4. Check upcoming IPOs section
```

Step 2: Browse All Stocks (1 minute)

```
1. Click "View All →" in stocks section
2. See full table with 25 stocks
3. Try search: type "TCS"
4. Filter by exchange: select "NSE"
5. Clear filters
```

Step 3: Explore IPOs (2 minutes)

```
1. Click "IPOs" in header
2. See 3 IPO cards
3. Click "Open Now" tab
4. Notice days remaining countdown
5. See GMP percentages
6. Check subscription meters
7. Click "View Details" on any IPO
```

Step 4: Check Database (1 minute)

```
1. Open http://localhost:8080
2. Login to Adminer
3. Click "stocks" table
4. See all 25 seeded stocks
5. Click "ipos" table
6. See 3 sample IPOs
```

Step 5: API Testing (1 minute)

```bash
```

```
# Get all stocks
curl http://localhost:3001/api/v1/stocks

# Get specific stock
curl http://localhost:3001/api/v1/stocks/RELIANCE

# Get IPO recommendation
curl -X POST http://localhost:3001/api/v1/ipos/1/advisor
```

🐛 Common Issues & Visual Fixes

Issue 1: "Cannot connect to backend"

Symptom: Frontend shows loading spinner forever

Visual Check:

```bash
# Terminal 1: Is backend running?
cd backend
npm run dev
# Should see: ✅ Database connected
```

Fix: Make sure backend is running on port 3001

Issue 2: "No data showing"

Symptom: Empty tables, no stocks/IPOs

Visual Check:

```bash
# Check if database is seeded
npx prisma studio
# Open http://localhost:5555
# Check if tables have data
```

Fix:

```bash
cd backend
npx prisma db seed
```

Issue 3: "Stock prices are all 0"

Issue 3: "Stock prices are all 0"

Symptom: Stocks show ₹0.00

Visual Check: This is expected! Seed data has 0 prices.

Fix (Optional): Add API keys to fetch real prices

```bash
# backend/.env.development
IEX_CLOUD_API_KEY=your_key_here
ALPHA_VANTAGE_API_KEY=your_key_here
```

Issue 4: "Port already in use"

Symptom:

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

Visual Check:

```bash
lsof -i :3000
# See what's using the port
```

Fix:

```bash
kill -9 <PID>
# OR use different port:
PORT=3001 npm run dev
```

📸 Screenshot Guide

What Each Page Should Look Like:

Homepage (/):

- Blue gradient header
- 4 stat cards (grid)
- 2-column layout: Stocks (left) + IPOs (right)
- 3 feature cards at bottom
- White background with shadows

Stocks Page (/stocks):

- Search bar + filter dropdown
- Full-width table
- Alternating row colors on hover
- Color-coded change percentages

IPOs Page (/ipos):

- Filter tabs at top
- 3-column grid of IPO cards
- Cards have: header, details, subscription meter, button
- Green/yellow/red badges for verdicts

✅ Success Checklist

After startup, you should have:

☑ **Backend running** on http://localhost:3001
☑ **Frontend running** on http://localhost:3000
☑ **PostgreSQL** on port 5432
☑ **Redis** on port 6379
☑ **Adminer** on http://localhost:8080
☑ **Redis Commander** on http://localhost:8081
☑ **25 stocks** in database
☑ **3 IPOs** in database
☑ **API responding** to curl requests
☑ **UI displaying** data correctly

🎯 Next Steps

1. **Add Real API Keys** (optional for v1):

```bash
# backend/.env.development
IEX_CLOUD_API_KEY=pk_your_key_here
ALPHA_VANTAGE_API_KEY=your_key_here
```

2. **Test All Features:**

- Search stocks
- Filter by exchange
- Check IPO verdicts
- Browse database in Adminer

3. **Customize Data:**
    - Add more stocks via Prisma Studio
    - Create new IPOs
    - Modify GMP percentages
4. **Deploy (optional):**
    - Backend → AWS ECS
    - Frontend → Vercel
    - Database → AWS RDS

🆘 Getting Help

Check logs:

```bash
# Backend logs
cd backend && npm run dev

# Docker logs
docker-compose logs -f postgres
docker-compose logs -f redis

# Check database connection
docker exec -it catalyst-postgres psql -U catalyst_user -d catalyst_dev
```

Restart everything:

```bash
```

```
# Stop all
docker-compose down
# Ctrl+C in both terminal windows

# Start fresh
docker-compose up -d
cd backend && npm run dev    # Terminal 1
cd frontend && npm run dev   # Terminal 2
```

🎉 You're all set! Your app is now running with visual UI and working APIs!