WealthAlloc - AI-Powered Wealth Management Platform

Show Image

Show Image

Show Image

Show Image

> **AI-powered wealth management for everyone. $9/month flat fee. No percentage fees.**

WealthAlloc is a comprehensive fintech platform that combines AI-driven portfolio management, tax loss harvesting, and real-time trading capabilities through Interactive Brokers integration.

🌟 Features

Core Functionality

- **Portfolio Management**: Real-time tracking, multi-account aggregation, asset allocation analysis
- **AI Recommendations**: LSTM-based predictions with 87% accuracy, risk-adjusted suggestions
- **Tax Loss Harvesting**: Automated opportunity detection, saving $2,850/year on average
- **IBKR Integration**: Direct Interactive Brokers connection for real-time trading
- **Educational Content**: Curated investment education videos and learning paths

Technology Highlights

- **Backend**: FastAPI (Python 3.10+), PostgreSQL/CockroachDB, Redis Cache

- **Frontend**: React + TypeScript (Base44 framework)
- **AI/ML**: LSTM Autoencoder (based on Nature paper s41599-025-04412-y)
- **Scalability**: Designed for 500M+ users, <100ms API latency (p99)
- **Security**: Bank-grade encryption, SOC 2 compliant

📋 Table of Contents

🚀 Quick Start

1. Clone the Repository

```bash
git clone https://github.com/YOUR_USERNAME/wealthalloc.git
cd wealthalloc
```

2. Backend Setup

```bash
```

```bash
# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r backend/requirements.txt

# Set up environment variables
cp backend/.env.example backend/.env
# Edit .env with your configuration

# Initialize database
createdb wealthalloc
psql -d wealthalloc -f backend/database/schema.sql

# Start backend server
cd backend
python main.py
```

Backend will be running at: http://localhost:8000

3. Frontend Setup

```bash
# The frontend is deployed on Base44
# No local setup needed - it connects to your deployed backend

# Update the API base URL in the Base44 project settings to point to your backend
```

4. Verify Installation

```bash
# Check backend health
curl http://localhost:8000/health

# Expected response:
# {"status":"healthy","timestamp":"...","ibkr_connected":true}
```

📁 Project Structure

```
wealthalloc/
```

```
├── backend/                    # FastAPI Backend
│   ├── main.py                 # Main application entry point
│   ├── requirements.txt        # Python dependencies
│   ├── .env.example            # Environment template
│   ├── Dockerfile              # Container definition
│   │
│   ├── models/                 # Data models & ML
│   │   ├── entities.py         # Base44 entity models
│   │   ├── lstm_autoencoder.py    # LSTM model
│   │   └── similarity_engine.py   # Hybrid similarity engine
│   │
│   ├── services/               # Business logic
│   │   ├── ibkr_client.py         # IBKR integration
│   │   ├── portfolio_service.py   # Portfolio management
│   │   ├── tax_harvest_service.py  # Tax loss harvesting
│   │   └── ai_recommendations.py   # AI engine
│   │
│   ├── api/                    # API routes
│   │   └── routes.py           # Endpoint definitions
│   │
│   ├── database/               # Database
│   │   ├── schema.sql          # PostgreSQL schema
│   │   └── migrations/         # Database migrations
│   │
│   ├── tests/                  # Test suite
│   │   ├── test_api.py         # API tests
│   │   ├── test_entities.py       # Entity tests
│   │   └── test_e2e.py         # End-to-end tests
│   │
│   └── scripts/                # Automation scripts
│     ├── deploy.sh             # Deployment script
│     └── train_lstmae.py       # Model training
│
├── frontend/                   # Base44 Frontend (uploaded separately)
│   ├── LandingPage/            # Public website
│   │   ├── Pages/              # Home, About, Platform, Contact
│   │   └── Layout.js           # Navigation & footer
│   │
│   └── TradingPlatform/          # Authenticated app
│     ├── Pages/                # Dashboard, Portfolio, Trade, etc.
│     └── Components/           # Reusable UI components
│
├── kubernetes/                 # Kubernetes configs
│   ├── deployment.yaml         # Deployment config
│   ├── service.yaml           # Service config
│   ├── ingress.yaml           # Ingress config
│   └── hpa.yaml               # Auto-scaling config
│
├── docs/                      # Documentation
│   ├── API.md                 # API documentation
│   ├── ARCHITECTURE.md        # System architecture
│   └── SETUP.md               # Detailed setup guide
│
├── docker-compose.yml         # Local development
├── .github/                   # GitHub Actions
│   └── workflows/
│       └── deploy.yml         # CI/CD pipeline
│
└── README.md                  # This file
```

🔧 Prerequisites

Required Software

- **Python 3.10+** - Download
- **PostgreSQL 14+** - Download
- **Redis 7.0+** - Download
- **Node.js 18+** - Download (for local frontend testing)
- **Docker & Docker Compose** - Download (optional)

IBKR Requirements

- Interactive Brokers account
- IB Gateway or TWS installed
- API access enabled in account settings

Accounts Needed

- **Base44 Account** - Sign up (for frontend hosting)
- **GitHub Account** - For repository hosting
- **Cloud Provider** (optional):
  - AWS, GCP, or Azure for production deployment
  - Render.com, Heroku, or Railway for quick deployment

💻 Installation

Backend Installation

1. Clone Repository

```bash
git clone https://github.com/YOUR_USERNAME/wealthalloc.git
cd wealthalloc/backend
```

2. Create Virtual Environment

```bash
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install Dependencies

```bash
pip install -r requirements.txt
```

4. Database Setup

```bash
# Create PostgreSQL database
createdb wealthalloc

# Run schema
psql -d wealthalloc -f database/schema.sql

# Verify tables created
psql -d wealthalloc -c "\dt"
```

5. Redis Setup

```bash
```

```
# Start Redis server (if not running)
redis-server

# Verify Redis is running
redis-cli ping
# Should return: PONG
```

Frontend Setup

The frontend is hosted on Base44. To deploy:

1. **Create Base44 Account**: https://base44.app
2. **Create New Project**: Import your frontend code
3. **Configure API URL**: Set environment variable `REACT_APP_API_URL` to your backend URL
4. **Deploy**: Base44 handles the deployment automatically

⚙️ Configuration

Backend Configuration

1. Environment Variables

Copy the example environment file:

```bash
cp .env.example .env
```

Edit `.env` with your settings:

```bash
```

```
# Application
APP_NAME=WealthAlloc
ENVIRONMENT=development
DEBUG=True

# Database
DATABASE_URL=postgresql+asyncpg://user:password@localhost:5432/wealthalloc

# Redis
REDIS_URL=redis://localhost:6379/0

# IBKR
IBKR_HOST=127.0.0.1
IBKR_PORT=7497  # 7497 for paper trading, 7496 for live
IBKR_CLIENT_ID=1
IBKR_PAPER_TRADING=True

# Security
SECRET_KEY=your-secret-key-here-generate-with-openssl-rand-hex-32
JWT_SECRET_KEY=your-jwt-secret-here

# CORS (add your frontend URL)
ALLOWED_ORIGINS=http://localhost:3000,https://your-base44-app.base44.app
```

Generate Secret Keys:

```bash
# Generate SECRET_KEY
openssl rand -hex 32

# Generate JWT_SECRET_KEY
openssl rand -hex 32
```

2. IBKR Gateway Configuration

1. **Download IB Gateway**: Interactive Brokers
2. **Configure Gateway**:
   - Enable API connections
   - Set port to 7497 (paper trading) or 7496 (live)
   - Add localhost to trusted IPs

- Disable automatic logout

3. **Test Connection**:

```bash
python -c "
from services.ibkr_client import IBKRClient
import asyncio

async def test():
    client = IBKRClient()
    await client.connect()
    print('✓ Connected to IBKR Gateway')

asyncio.run(test())
"
```

Frontend Configuration

In your Base44 project, set these environment variables:

```bash
# Backend API URL
REACT_APP_API_URL=http://localhost:8000  # For local development
# or
REACT_APP_API_URL=https://your-backend.herokuapp.com  # For production

# Optional: Analytics
REACT_APP_GOOGLE_ANALYTICS_ID=your-ga-id
```

🏃 Running Locally

Option 1: Manual Start

Terminal 1 - Database & Cache

```bash
# Start PostgreSQL (if not running as service)
pg_ctl -D /usr/local/var/postgres start

# Start Redis
redis-server
```

Terminal 2 - IBKR Gateway

```bash
# Start IB Gateway manually or run:
# (On macOS/Linux)
/Applications/IB\ Gateway.app/Contents/MacOS/ibgateway

# (On Windows)
C:\Program Files\IB Gateway\ibgateway.exe
```

Terminal 3 - Backend

```bash
cd backend
source venv/bin/activate
python main.py
```

Backend will be available at:

- **API**: http://localhost:8000
- **API Docs**: http://localhost:8000/api/docs
- **Health Check**: http://localhost:8000/health

Terminal 4 - Frontend (if running locally)

```bash
cd frontend
npm install
npm start
```

Frontend will be at: http://localhost:3000

Option 2: Docker Compose (Recommended)

```bash
```

```
# Start all services
docker-compose up -d

# View logs
docker-compose logs -f

# Stop all services
docker-compose down
```

Services will be available at:

- **Backend**: http://localhost:8000
- **Frontend**: http://localhost:3000
- **PostgreSQL**: localhost:5432
- **Redis**: localhost:6379

Verify Everything is Running

```bash
# Check backend health
curl http://localhost:8000/health

# Check database connection
psql -d wealthalloc -c "SELECT COUNT(*) FROM users;"

# Check Redis connection
redis-cli ping

# Test API endpoint
curl http://localhost:8000/api/v1/dashboard
```

🚀 Deployment

Deploy to Render.com (Recommended for Quick Start)

1. Backend Deployment

1. **Push to GitHub**:

```bash
```

```
git add .
git commit -m "Initial commit"
git push origin main
```

2. **Create Render Account**: https://render.com

3. **Create PostgreSQL Database**:
   - New → PostgreSQL
   - Name: `wealthalloc-db`
   - Copy the Internal Database URL

4. **Create Redis Instance**:
   - New → Redis
   - Name: `wealthalloc-redis`
   - Copy the Internal Redis URL

5. **Create Web Service**:
   - New → Web Service
   - Connect your GitHub repository
   - Name: `wealthalloc-api`
   - Root Directory: `backend`
   - Build Command: `pip install -r requirements.txt`
   - Start Command: `python main.py`
   - Add environment variables from `.env`

6. **Run Database Migrations**:

```bash
# SSH into Render shell
psql $DATABASE_URL < database/schema.sql
```

7. **Get your backend URL**: `https://wealthalloc-api.onrender.com`

2. Frontend Deployment (Base44)

1. **Upload to Base44**:
   - Login to Base44

    - Create new project
    - Upload frontend files

2. **Configure Environment**:
    - Set `REACT_APP_API_URL` to your Render backend URL

3. **Deploy**: Base44 handles the rest!

Deploy to AWS (Production)

See docs/DEPLOYMENT_AWS.md for detailed AWS deployment instructions.

Deploy to Kubernetes

```bash
# Configure kubectl
kubectl config use-context your-cluster

# Deploy
cd kubernetes
./deploy.sh

# Verify
kubectl get pods -n wealthalloc
kubectl get svc -n wealthalloc
```

📚 API Documentation

Interactive API Docs

Once the backend is running, visit:

- **Swagger UI**: http://localhost:8000/api/docs
- **ReDoc**: http://localhost:8000/api/redoc

Key Endpoints

Endpoint	Method	Description
`/health`	GET	Health check
`/api/v1/dashboard`	GET	Dashboard data
`/api/v1/portfolio`	GET	Portfolio details
`/api/v1/trade`	POST	Execute trade
`/api/v1/recommendations`	GET	AI recommendations

Endpoint	Method	Description
/api/v1/tax-harvesting	GET	Tax opportunities

Example API Call

```bash
# Get dashboard data
curl -X GET http://localhost:8000/api/v1/dashboard \
  -H "Content-Type: application/json"

# Execute a trade
curl -X POST http://localhost:8000/api/v1/trade \
  -H "Content-Type: application/json" \
  -d '{
    "portfolio_id": "portfolio_1",
    "symbol": "AAPL",
    "trade_type": "buy",
    "order_type": "market",
    "shares": 10
  }'
```

See docs/API.md for complete API documentation.

🧪 Testing

Run All Tests

```bash
cd backend
pytest tests/ -v
```

Run Specific Test Suite

```bash
```

```
# API tests
pytest tests/test_api.py -v

# Entity tests
pytest tests/test_entities.py -v

# End-to-end tests
pytest tests/test_e2e.py -v
```

Run with Coverage

```bash
pytest --cov=. --cov-report=html
open htmlcov/index.html
```

Manual Testing

```bash
# Test health endpoint
curl http://localhost:8000/health

# Test dashboard
curl http://localhost:8000/api/v1/dashboard

# Test recommendations
curl http://localhost:8000/api/v1/recommendations
```

🔍 Troubleshooting

Backend Won't Start

Problem: `ModuleNotFoundError: No module named 'fastapi'`

Solution:

```bash
pip install -r requirements.txt
```

Problem: `Cannot connect to database`

Solution:

IBKR Connection Issues

Problem: `Cannot connect to IBKR Gateway`

Solutions:

1. Verify Gateway is running: `netstat -an | grep 7497`
2. Check API settings in Gateway configuration
3. Verify client ID is unique
4. Check firewall settings

Frontend Not Loading Data

Problem: `Network Error` or `Failed to fetch`

Solutions:

1. Verify backend is running: `curl http://localhost:8000/health`
2. Check CORS settings in `main.py`
3. Verify API URL in frontend environment variables
4. Check browser console for errors

Database Migration Errors

Problem: `relation "users" already exists`

Solution:

```bash
# Drop and recreate database
dropdb wealthalloc
createdb wealthalloc
psql -d wealthalloc -f backend/database/schema.sql
```

For more troubleshooting, see docs/TROUBLESHOOTING.md

🤝 Contributing

We welcome contributions! Please follow these steps:

1. **Fork the repository**
2. **Create a feature branch**: `git checkout -b feature/amazing-feature`
3. **Commit your changes**: `git commit -m 'Add amazing feature'`
4. **Push to the branch**: `git push origin feature/amazing-feature`
5. **Open a Pull Request**

Development Guidelines

- Follow PEP 8 for Python code
- Use ESLint/Prettier for JavaScript/React code
- Write tests for new features
- Update documentation as needed

See CONTRIBUTING.md for detailed guidelines.

📄 License

🙏 Acknowledgments

- **LSTM Autoencoder** methodology based on research published in *Nature* (s41599-025-04412-y)
- **Interactive Brokers** for trading API
- **Base44** for frontend hosting platform
- Michigan startup community for beta testing support

📞 Support & Contact

- **Email**: support@wealthalloc.com
- **GitHub Issues**: Report a bug
- **Documentation**: Full Docs
- **Discord**: Join our community

🗺️ Roadmap

Version 1.1 (Q1 2025)

- [ ] Automated rebalancing execution
- [ ] SMS/Email alerts
- [ ] Mobile app (iOS/Android)

Version 1.2 (Q2 2025)

- [ ] Additional broker integrations (Robinhood, Schwab)
- [ ] Advanced backtesting interface
- [ ] Social trading features

Version 2.0 (Q3 2025)

- [ ] Cryptocurrency trading
- [ ] Options trading
- [ ] International markets

⭐ Star History

Show Image

Made in Michigan 🏭 | Powered by AI 🤖 | Built for Investors 📈