# 🛠️ Implementation Path for Full Stack .NET Financial Management System

---

## ✅ Tech Stack Overview

- **Frontend**: React.js (with Tailwind CSS / Chart.js)
- **Backend**: ASP.NET Core Web API
- **Database**: SQL Server
- **Authentication**: JWT (JSON Web Tokens)
- **Stock API**: Alpha Vantage or Yahoo Finance API
- **Deployment**: Azure / Vercel / Railway

---

## 🚀 Sprint Planning (6 Sprints – 1 Week Each)

---

### 🟢 Sprint 1 – Project Setup & Authentication

**Goals:**

- Set up solution architecture (frontend + backend + DB)
- Configure React project with routing
- Set up ASP.NET Core Web API
- Implement basic user registration/login (JWT)
- Setup SQL Server DB schema for Users

**Deliverables:**

- Working user auth module with token-based login
- Postman tested API endpoints
- Environment setup complete (local/dev)

---

### 🔵 Sprint 2 – Expense & Income Management

**Goals:**

- Create models and APIs for income & expense
- Implement CRUD endpoints
- Connect backend with frontend (using Axios/Fetch)
- UI forms for entering transactions
- Basic dashboard showing totals

**Deliverables:**

- Fully functional income/expense entry system
- Local storage of transactions per user

---

## 🟠 Sprint 3 – Budgeting & Financial Reports

**Goals:**

- Add budget module (monthly/yearly)
- Visual reports using Chart.js (pie chart for categories, bar graph for month-wise spending)
- Backend APIs for budget data
- Connect budget logic with transactions

**Deliverables:**

- Budget setup screen
- Dynamic charts showing user insights

---

## 🟡 Sprint 4 – Stock Market Integration

**Goals:**

- Integrate stock market API (Alpha Vantage / Yahoo Finance)
- Create endpoints for searching tickers & displaying real-time stock data
- UI module for viewing stock info & charts
- Add user watchlist feature (optional)

**Deliverables:**

- Stock module in dashboard with search & charts
- API-tested real-time stock lookup

---

## 🟣 Sprint 5 – Testing, Security & Notifications

**Goals:**

- Unit testing (xUnit for .NET, Jest for frontend)
- Add input validation and sanitization
- Implement budget alerts or stock price notifications (email/Toast)
- Error handling middleware for backend
- Secure sensitive API keys via ENV variables

**Deliverables:**

- Tested system with role-based access
- Error-resilient and secure API

---

## ⚫ Sprint 6 – Final UI Polish & Deployment

**Goals:**

- Refactor code and enhance UI/UX
- Deploy backend on Azure App Service
- Deploy frontend on Vercel/Azure Static Web App
- Write README, setup documentation
- Create demo video (optional)

**Deliverables:**

- Fully deployed live system
- GitHub repo with clean documentation
- Optional: Add feature flags for enhancements

---

## 🧩 Optional Enhancements (Post-Launch)

- Email reports (daily/weekly summary)
- OAuth login (Google)
- Multi-currency support
- CSV import/export
- Real-time stock alerts with Twilio or Push API

---

## 📌 Tools Recommended:

- **Trello or Jira** – Task & sprint tracking
- **Postman** – API testing
- **Azure DevOps/GitHub Projects** – CI/CD pipelines
- **Figma/Whimsical** – UI planning and ER diagrams