1. Initial Setup and Basic Structure

- **Goal**: Establish core platform skeleton.
- **Modules**:
    - **Set Up Project Repositories** (GitHub, GitLab)
    - **Set Up Django/Flask Backend**
    - **Set Up React/Vue.js Frontend** (Choose based on comfort)
    - **Cloud Infrastructure Setup** (AWS/GCP) for initial scalability.
- **Steps**:
    - Create a basic backend (Django/Flask) for API management.
    - Establish basic frontend structure (React or Vue) with static pages.
    - Set up basic database schema (PostgreSQL for structured data, MongoDB for unstructured).
    - Deploy basic cloud infrastructure for hosting.

Testing as You Go:

- Test API routes (GET, POST, PUT) and verify basic connectivity.
- Test frontend for basic routing (static pages).

Timeframe: 1–2 weeks for initial setup and structure.

---

2. User Management Module

- **Goal**: Create a functional user management system with profiles and role-based access.
- **Modules**:
    - User Registration/Login (Basic authentication).
    - User Profile Management (Create, Edit, View).
    - Role-based Access (Different views for individual users and corporate clients).
- **Steps**:
    - Implement authentication (JWT tokens, login/logout).
    - Create a user profile system (Django models).
    - Design the UI for registration and login.
    - Add user-specific data (goals, risk profile).

Testing as You Go:

- Test user registration and login.
- Ensure role-based access works (admin, investor, corporate user).
- Test data persistence for profiles in the database.

Timeframe: 2–3 weeks for a functional user management system.

---

3. Basic Data Aggregation

- **Goal**: Begin fetching financial data and displaying it on the platform.
- **Modules**:
    - API Integration for Market Data (Alpha Vantage, Polygon.io, etc.).

    - Basic Data Storage (balance sheets, stock prices).
    - Web Scraping (basic scrapers for local financial data).
- **Steps**:
    - Set up integrations with financial data APIs for local markets (start with Kenyan stocks, then global).
    - Build a basic scraper (BeautifulSoup/Scrapy) for extracting company data from websites.
    - Store and clean the data (ETL pipelines).

Testing as You Go:

- Test API data retrieval for specific stocks and assets.
- Verify that scraped data is correctly formatted and inserted into the database.
- Test the performance and update rate of data-fetching systems.

Timeframe: 2 weeks for API integration and basic scrapers.

---

4. Basic Portfolio Management

- **Goal**: Create a simple portfolio system where users can add, track, and monitor investments.
- **Modules**:
    - Portfolio Creation (Users create portfolios with stocks, crypto, bonds).
    - Basic Portfolio Dashboard (View portfolio summary and performance).
- **Steps**:
    - Build the backend to handle portfolio creation and tracking.
    - Develop a dashboard that displays basic portfolio metrics (value, returns).
    - Create simple calculations for portfolio value and historical performance.

Testing as You Go:

- Test portfolio creation and updating.
- Check if the portfolio values are being calculated correctly.
- Validate portfolio dashboard responsiveness.

Timeframe: 2 weeks for portfolio management.

---

5. Visualization and Dashboarding

- **Goal**: Build interactive charts and visualizations for user portfolios and market data.
- **Modules**:
    - Integrate D3.js or Plotly for dynamic data visualizations.
    - Build charts for stock prices, portfolio value, and financial metrics.
- **Steps**:
    - Set up React or Vue components for charts and graphs.
    - Integrate financial data with visualization tools (showing stock trends, portfolio breakdown).
    - Allow users to filter data by asset class, date range, etc.

Testing as You Go:

- Test if charts render correctly on the frontend.
- Validate if the charts update in real-time as data is updated.
- Ensure the data is pulling correctly from the backend.

Timeframe: 2–3 weeks for visualization module.

---

6. Basic Financial Modeling Tools

- **Goal**: Develop tools to perform financial analysis (e.g., DCF, ratios, etc.).
- **Modules**:
  - Implement DCF (Discounted Cash Flow) Calculator.
  - Implement Ratio Analysis Tools (P/E, P/B, ROI).
- **Steps**:
  - Write the logic for calculating DCF, NPV, and other financial models.
  - Build UI elements for users to input data and see calculations.
  - Connect to user portfolio to track performance metrics.

Testing as You Go:

- Test financial calculations for accuracy.
- Validate the UI for inputs and results display.
- Ensure calculations are updated as users modify their portfolio data.

Timeframe: 2–3 weeks for basic financial modeling.

---

7. Machine Learning & Predictive Models

- **Goal**: Introduce machine learning models for price predictions, sentiment analysis, and forecasting.
- **Modules**:
  - Implement stock price prediction using time series models (ARIMA, LSTM).
  - Implement sentiment analysis on financial news (NLP).
- **Steps**:
  - Train initial models on historical stock data.
  - Set up sentiment analysis using news articles and social media data.
  - Deploy models in backend (API calls to get predictions).

Testing as You Go:

- Test model predictions and accuracy.
- Monitor real-time performance of the models.
- Ensure models integrate smoothly with the frontend and provide actionable insights.

Timeframe: 4–6 weeks for integrating machine learning models.

---

8. Advanced Features & Integration

- **Goal**: Implement advanced tools like algorithmic trading and macroeconomic analysis.
- **Modules**:
    - Algorithmic Trading (backtesting, strategy creation).
    - Macroeconomic Data Integration (GDP, inflation, etc.).
- **Steps**:
    - Integrate with broker APIs for backtesting trading strategies.
    - Aggregate macroeconomic data from reliable sources (IMF, World Bank).
    - Display macroeconomic trends alongside financial data for more context.

Testing as You Go:

- Test trading strategies using historical data.
- Validate macroeconomic data integration and display.

Timeframe: 4–6 weeks for advanced features.

---

9. Global Expansion and API Integration

- **Goal**: Expand platform to support global markets and provide API access.
- **Modules**:
    - Add global market data (stocks, crypto, bonds, ETFs).
    - Develop API access for external applications.
- **Steps**:
    - Integrate with global financial APIs (Yahoo Finance, Bloomberg).
    - Develop API endpoints for financial data access by third parties.

Testing as You Go:

- Test integration with global data providers.
- Test API stability and speed.

Timeframe: 4–6 weeks for global data expansion and API.

Final Phase: User Testing and Refinement

- **Goal**: Conduct user testing, bug fixing, and refine the platform for scalability.
- **Modules**:
    - Conduct User Acceptance Testing (UAT).
    - Performance and stress testing for scalability.
- **Steps**:
    - Gather feedback from early users.
    - Optimize performance and fix bugs.
    - Finalize platform for launch.

Timeframe: 2–3 weeks for testing and refining before launch.