Product Requirements Document

AI Financial Report Generator Agent

Version: 1.0
Date: December 15, 2024
Product: Terminal.AI - Automated Financial Report Generation

---

1. Executive Summary

1.1 Purpose

Create an autonomous AI agent that generates comprehensive, professional-grade financial research reports for any publicly traded stock, complete with visual analytics and downloadable PDF format.

1.2 User Experience

```
User enters "RELIANCE"
    ↓
Agent researches for 30-60 seconds
    ↓
Displays interactive report in UI
    ↓
User downloads PDF report
```

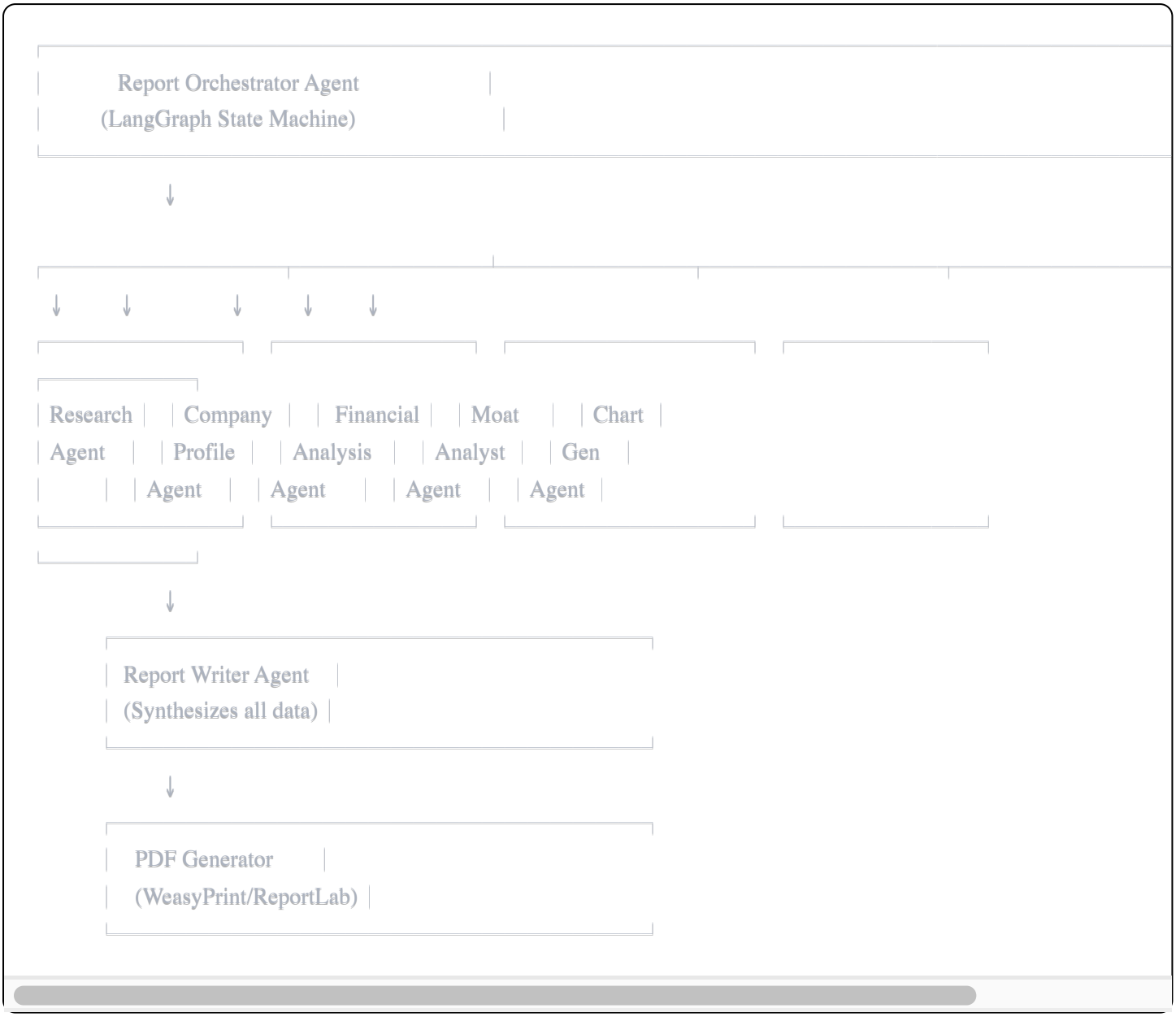
1.3 Report Contents

1. **Executive Summary** - AI-generated overview
2. **Company Overview** - Business model, products, markets
3. **Recent Developments** - Last 12 months activities
4. **Financial Analysis** - 5-year financial charts with insights
5. **Competitive Moat** - Sustainable competitive advantages
6. **Risk Assessment** - Key risks and challenges
7. **Investment Thesis** - Bull/bear case analysis
8. **Valuation** - P/E, DCF, price targets
9. **Technical Analysis** - Price charts, trends
10. **Recommendation** - Buy/Hold/Sell with rationale

---

2. Multi-Agent Architecture

2.1 Agent Hierarchy



2.2 Agent Responsibilities

1. Research Agent

LLM: GPT-4o or Claude Sonnet 4
Purpose: Web search for recent news, developments, announcements
Tools:

- Web search (Google, Bing)
- News APIs (NewsAPI, Finnhub)
- SEC EDGAR filings scraper
- Company press releases

Output:

```json
{
  "recent_news": [
    {
      "date": "2024-11-15",
      "headline": "Reliance announces $10B green energy investment",
      "source": "Reuters",
      "summary": "..."
    }
  ],
  "major_developments_12m": [
    "Launched Jio 5G nationwide",
    "Acquired XYZ company for $2B",
    "Expanded retail footprint to 500 new stores"
  ],
  "upcoming_events": [
    "Q4 earnings call on Jan 15, 2025",
    "AGM scheduled for March 2025"
  ]
}
```

2. Company Profile Agent

LLM: GPT-4o-mini (cost-effective)
Purpose: Extract/summarize company information
Data Sources:

- Wikipedia
- Company website
- Crunchbase / PitchBook
- Yahoo Finance company profile

Output:

```json
```

```
{
  "company_name": "Reliance Industries Ltd.",
  "founded": "1966",
  "founder": "Dhirubhai Ambani",
  "ceo": "Mukesh Ambani",
  "headquarters": "Mumbai, India",
  "industry": "Conglomerate",
  "sectors": ["Energy", "Petrochemicals", "Retail", "Telecom"],
  "employees": 347000,
  "business_description": "Reliance Industries is India's largest private sector company...",
  "key_products": ["Jio Telecom", "Reliance Retail", "Refining", "Petrochemicals"],
  "market_position": "Market leader in telecom and retail in India"
}
```


3. Financial Analysis Agent

LLM: GPT-4o (for complex reasoning)
Purpose: Analyze financial statements and trends
Data Sources:

- FMP API (Financial Modeling Prep)
- Yahoo Finance
- Alpha Vantage
- Polygon.io

Analysis Performed:

- Revenue growth trend (5 years)
- Profit margin analysis
- EPS growth trajectory
- CAGR calculations
- Cash flow health
- Debt-to-equity ratio
- Return on equity (ROE)
- Working capital management

Output:

```
json
```

```
{
  "financial_health_score": 8.5,
  "revenue_5y_cagr": 12.5,
  "eps_5y_cagr": 10.8,
  "current_pe_ratio": 24.3,
  "debt_to_equity": 0.45,
  "roe": 15.2,
  "fcf_margin": 18.5,
  "insights": [
    "Strong revenue growth driven by retail and telecom",
    "Improving profit margins YoY",
    "Conservative debt levels",
    "Consistent cash flow generation"
  ],
  "concerns": [
    "Slowing EPS growth in recent quarters",
    "High capex impacting free cash flow"
  ]
}
```


4. Competitive Moat Analyst Agent

LLM: Claude Sonnet 4 (best for nuanced analysis)
Purpose: Identify sustainable competitive advantages
Analysis Framework:

- **Network Effects:** Does the business get stronger with more users?
- **Brand Moat:** Strong brand recognition and customer loyalty?
- **Cost Advantages:** Economies of scale, proprietary tech?
- **Switching Costs:** Hard for customers to switch to competitors?
- **Regulatory Moat:** Licenses, patents, regulatory barriers?

Output:

```
json
```

```
{
 "moat_rating": "Wide Moat",
 "moat_strength": 8,
 "moat_sources": [
  {
   "type": "Network Effects",
   "strength": 9,
   "description": "Jio's 400M+ subscribers create powerful network effects in telecom"
  },
  {
   "type": "Scale Advantages",
   "strength": 8,
   "description": "Largest refining capacity in India enables cost leadership"
  },
  {
   "type": "Integrated Ecosystem",
   "strength": 9,
   "description": "Vertical integration across energy, retail, and telecom"
  }
 ],
 "moat_durability": "Strong moat likely to persist for 10+ years",
 "competitive_threats": [
  "Airtel challenging Jio's telecom dominance",
  "Amazon/Flipkart in retail e-commerce"
 ]
}
```


5. Chart Generator Agent

LLM: GPT-4o-mini (for code generation)
Purpose: Generate Python/Matplotlib code for charts
Charts Created:

- Revenue & EPS trend (5 years)
- Profit margin evolution
- Debt-to-equity trend
- ROE comparison with peers
- Stock price chart (5 years)
- Valuation multiples (P/E, P/B over time)

Process:

1. LLM generates Python plotting code

2. Execute code to create chart images
3. Save as PNG files
4. Embed in PDF report

Example LLM Prompt:

```
Generate matplotlib code to create a professional-looking chart showing:
- Quarterly revenue for last 5 years
- Dual-axis with net profit margin %
- Use dark theme, gridlines
- Title: "Revenue Growth & Profit Margin Trend"
- Add data labels

Data:
Revenue (in Cr): [150000, 155000, 162000, ...]
Profit Margin %: [7.5, 7.8, 8.1, ...]
Quarters: ['Q1 2020', 'Q2 2020', ...]
```

6. Report Writer Agent

LLM: GPT-4o (best writing quality)
Purpose: Synthesize all agent outputs into coherent narrative
Writing Style: Professional, institutional investor-grade
Sections Written:

- Executive Summary (1 page)
- Company Overview (2 pages)
- Financial Analysis narrative (3 pages)
- Competitive Moat discussion (2 pages)
- Risk Assessment (1 page)
- Investment Thesis (2 pages)
- Conclusion & Recommendation (1 page)

Prompt Template:

```
You are a senior equity research analyst at a top investment bank.

Write a professional Executive Summary for a research report on {company_name}.

Include:
1. Current stock price and 52-week range
2. Investment recommendation (Buy/Hold/Sell)
3. Price target (12-month)
4. Key investment highlights (3-4 points)
5. Main risks (2-3 points)
6. Valuation summary

Data available:
{financial_data}
{company_profile}
{moat_analysis}
{recent_developments}

Tone: Professional, data-driven, balanced
Length: 250-300 words
```

---

3. Technology Stack

3.1 LLM Selection

Agent	Primary LLM	Reason	Cost per Report
Research Agent	GPT-4o	Best for web research synthesis	$0.15
Company Profile	GPT-4o-mini	Simple extraction task	$0.02
Financial Analysis	GPT-4o	Complex financial reasoning	$0.20
Moat Analyst	Claude Sonnet 4	Nuanced strategic analysis	$0.25
Chart Generator	GPT-4o-mini	Code generation	$0.03
Report Writer	GPT-4o	Best writing quality	$0.30
Total			**~$0.95/report**

3.2 Tools & Libraries

Backend:

```python
# LLM Orchestration
langchain==1.1.3
langgraph==0.2.x
langchain-openai==0.3.x
langchain-anthropic==0.3.x

# Data Sources
yfinance==0.2.32
alpha-vantage==2.3.1
newsapi-python==0.2.7
finnhub-python==2.4.18

# Web Scraping
beautifulsoup4==4.12.0
selenium==4.15.0
requests-html==0.10.0

# Chart Generation
matplotlib==3.8.0
seaborn==0.13.0
plotly==5.18.0

# PDF Generation
reportlab==4.0.7
weasyprint==60.1
pdfkit==1.0.0

# Image Processing
Pillow==10.1.0
```

Frontend:

```javascript
// PDF Viewer
react-pdf==7.5.0
pdfjs-dist==3.11.174

// UI Components
@mui/material==5.14.0
react-icons==4.11.0
```

---

4. LangGraph Workflow

4.1 State Schema

python

```python
from typing import TypedDict, List, Dict, Optional

class ReportState(TypedDict):
    # Input
    ticker: str
    exchange: str

    # Agent Outputs
    company_profile: Optional[Dict]
    recent_news: Optional[List[Dict]]
    financial_data: Optional[Dict]
    financial_analysis: Optional[Dict]
    moat_analysis: Optional[Dict]
    chart_paths: Optional[List[str]]

    # Report Sections
    executive_summary: Optional[str]
    company_overview: Optional[str]
    financial_section: Optional[str]
    moat_section: Optional[str]
    risk_section: Optional[str]
    investment_thesis: Optional[str]

    # Output
    report_html: Optional[str]
    report_pdf_path: Optional[str]

    # Metadata
    generated_at: str
    status: str
    errors: List[str]
```

4.2 Workflow Graph

python

```python
from langgraph.graph import StateGraph, END

workflow = StateGraph(ReportState)

# Add nodes
workflow.add_node("fetch_company_profile", fetch_company_profile_node)
workflow.add_node("research_recent_news", research_news_node)
workflow.add_node("analyze_financials", financial_analysis_node)
workflow.add_node("analyze_moat", moat_analysis_node)
workflow.add_node("generate_charts", chart_generation_node)
workflow.add_node("write_report", report_writing_node)
workflow.add_node("generate_pdf", pdf_generation_node)

# Define edges (execution order)
workflow.set_entry_point("fetch_company_profile")

# Parallel execution for data gathering
workflow.add_edge("fetch_company_profile", "research_recent_news")
workflow.add_edge("fetch_company_profile", "analyze_financials")

workflow.add_edge("research_recent_news", "analyze_moat")
workflow.add_edge("analyze_financials", "analyze_moat")

workflow.add_edge("analyze_moat", "generate_charts")
workflow.add_edge("generate_charts", "write_report")
workflow.add_edge("write_report", "generate_pdf")
workflow.add_edge("generate_pdf", END)

# Compile
report_agent = workflow.compile()
```

---

5. Agent Implementation Examples

5.1 Research Agent Node

python

```
from langchain_openai import ChatOpenAI
from langchain.tools import Tool
from langchain.agents import create_openai_functions_agent

async def research_news_node(state: ReportState) -> ReportState:
    """
    Research recent company news and developments
    """
    ticker = state["ticker"]
    company_name = state["company_profile"]["company_name"]

    # Initialize LLM
    llm = ChatOpenAI(model="gpt-4o", temperature=0.3)

    # Define tools
    tools = [
        Tool(
            name="web_search",
            func=search_web,
            description="Search the web for recent news about the company"
        ),
        Tool(
            name="news_api",
            func=fetch_news_api,
            description="Fetch news articles from NewsAPI"
        ),
        Tool(
            name="sec_filings",
            func=fetch_sec_filings,
            description="Get recent SEC filings (10-K, 10-Q, 8-K)"
        )
    ]

    # Create agent
    agent = create_openai_functions_agent(llm, tools)

    # Research prompt
    prompt = f"""
    Research {company_name} ({ticker}) and find:

    1. Major developments in the last 12 months
    2. Recent product launches or strategic initiatives
    3. M&A activity (acquisitions or partnerships)
    4. Leadership changes
    5. Regulatory issues or legal matters
    6. Upcoming events (earnings, AGM, etc.)
```

```
Focus on material information that would impact investors.
Provide sources for each finding.
"""

# Execute research
result = await agent.ainvoke({"input": prompt})

# Parse and structure results
recent_news = parse_research_output(result["output"])

state["recent_news"] = recent_news
state["status"] = "research_complete"

return state
```

5.2 Moat Analysis Agent

python

```
from langchain_anthropic import ChatAnthropic

async def moat_analysis_node(state: ReportState) -> ReportState:
    """
    Analyze company's competitive moat using Claude
    """
    # Use Claude for nuanced strategic analysis
    llm = ChatAnthropic(model="claude-sonnet-4-20250514", temperature=0.2)

    # Gather context
    company_profile = state["company_profile"]
    financial_data = state["financial_data"]
    recent_news = state["recent_news"]

    prompt = f"""
    You are a senior strategy consultant analyzing competitive moats.

    Company: {company_profile['company_name']}
    Industry: {company_profile['industry']}
    Business Model: {company_profile['business_description']}

    Financial Metrics:
    - Revenue 5Y CAGR: {financial_data['revenue_5y_cagr']}%
    - ROE: {financial_data['roe']}%
    - Market Share: {company_profile.get('market_position', 'N/A')}

    Recent Developments:
    {format_recent_news(recent_news)}

    Analyze the company's competitive moat:

    1. Identify all sources of competitive advantage:
       - Network effects
       - Brand strength
       - Cost advantages / economies of scale
       - Switching costs
       - Intangible assets (patents, licenses)
       - Efficient scale (natural monopoly)
       - Data moat

    2. Rate moat strength: 0-10 scale

    3. Assess moat durability: Will it last 10+ years?

    4. Identify threats to the moat
```

```
5. Overall moat classification:
   - No Moat
   - Narrow Moat
   - Wide Moat

Provide detailed reasoning for your assessment.
Format output as JSON.
"""

response = await llm.ainvoke(prompt)
moat_analysis = parse_json_response(response.content)

state["moat_analysis"] = moat_analysis

return state
```

5.3 Chart Generation Agent

python

```python
async def chart_generation_node(state: ReportState) -> ReportState:
    """
    Generate financial charts using LLM-generated plotting code
    """
    llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)

    financial_data = state["financial_data"]
    ticker = state["ticker"]

    charts_to_generate = [
        "revenue_eps_trend",
        "profit_margin_evolution",
        "roe_comparison",
        "stock_price_chart",
        "valuation_multiples"
    ]

    chart_paths = []

    for chart_type in charts_to_generate:
        # LLM generates plotting code
        code_prompt = f"""
        Generate Python matplotlib code to create a professional financial chart:

        Chart Type: {chart_type}
        Company: {ticker}
        Data: {financial_data}

        Requirements:
        - Use dark theme (facecolor='#0f172a')
        - Professional styling
        - Gridlines for readability
        - Proper labels and title
        - Data labels on key points
        - Save as PNG at 300 DPI

        Output only executable Python code, no explanations.
        """

        response = await llm.ainvoke(code_prompt)
        plot_code = extract_code_from_response(response.content)

        # Execute plotting code safely
        chart_path = execute_plot_code_safely(
            code=plot_code,
            output_path=f"charts/{ticker}_{chart_type}.png"
```

```
        )

        chart_paths.append(chart_path)

    state["chart_paths"] = chart_paths

    return state

def execute_plot_code_safely(code: str, output_path: str) -> str:
    """
    Execute LLM-generated plotting code in sandboxed environment
    """
    import matplotlib
    matplotlib.use('Agg')  # Non-interactive backend

    import matplotlib.pyplot as plt
    import numpy as np

    # Create namespace with only safe modules
    safe_globals = {
        'plt': plt,
        'np': np,
        '__builtins__': {}
    }

    try:
        exec(code, safe_globals)
        plt.savefig(output_path, dpi=300, bbox_inches='tight')
        plt.close()
        return output_path
    except Exception as e:
        logging.error(f"Chart generation failed: {e}")
        return None
```

5.4 Report Writer Agent

python

```
async def report_writing_node(state: ReportState) -> ReportState:
    """
    Synthesize all data into comprehensive report
    """
    llm = ChatOpenAI(model="gpt-4o", temperature=0.3)

    # Gather all context
    context = {
        "ticker": state["ticker"],
        "company_profile": state["company_profile"],
        "financial_analysis": state["financial_analysis"],
        "moat_analysis": state["moat_analysis"],
        "recent_news": state["recent_news"],
        "chart_paths": state["chart_paths"]
    }

    sections = {}

    # Generate each section
    section_prompts = {
        "executive_summary": get_executive_summary_prompt(context),
        "company_overview": get_company_overview_prompt(context),
        "financial_section": get_financial_analysis_prompt(context),
        "moat_section": get_moat_discussion_prompt(context),
        "risk_section": get_risk_assessment_prompt(context),
        "investment_thesis": get_investment_thesis_prompt(context)
    }

    for section_name, prompt in section_prompts.items():
        response = await llm.ainvoke(prompt)
        sections[section_name] = response.content
        state[section_name] = response.content

    return state

def get_executive_summary_prompt(context: Dict) -> str:
    return f"""
    You are a senior equity research analyst at Goldman Sachs.

    Write a 1-page Executive Summary for {context['company_profile']['company_name']}.

    Structure:
    1. Investment Recommendation (Buy/Hold/Sell) with 12-month price target
    2. Current Valuation (P/E, P/B vs peers and historical average)
    3. Key Investment Highlights (3-4 bullet points)
    4. Main Risks (2-3 bullet points)
```

```
5. Thesis in One Sentence

Data:
- Current Price: {context['financial_analysis']['current_price']}
- Financial Health Score: {context['financial_analysis']['financial_health_score']}/10
- Moat Rating: {context['moat_analysis']['moat_rating']}
- Recent Developments: {format_developments(context['recent_news'])}

Tone: Professional, data-driven, institutional investor-grade
Length: 300-350 words

Do not use phrases like "In conclusion" or "To summarize".
Start directly with the recommendation.
"""
```

---

6. PDF Generation

6.1 HTML Template Design

```
html
```

```
<!-- report_template.html -->
<!DOCTYPE html>
<html>
<head>
  <style>
    @page {
      size: A4;
      margin: 1in;
      @top-center {
        content: "Terminal.AI Research Report";
      }
      @bottom-right {
        content: "Page " counter(page) " of " counter(pages);
      }
    }

    body {
      font-family: 'Helvetica Neue', Arial, sans-serif;
      font-size: 11pt;
      line-height: 1.6;
      color: #333;
    }

    .cover-page {
      page-break-after: always;
      text-align: center;
      padding-top: 200px;
    }

    h1 {
      font-size: 28pt;
      color: #1e40af;
      margin-bottom: 20px;
    }

    h2 {
      font-size: 18pt;
      color: #1e40af;
      border-bottom: 2px solid #1e40af;
      padding-bottom: 5px;
      page-break-after: avoid;
    }

    .chart {
      width: 100%;
      page-break-inside: avoid;
```

```
            margin: 20px 0;
        }

        .metric-card {
            display: inline-block;
            width: 30%;
            padding: 15px;
            background: #f0f9ff;
            border-left: 4px solid #1e40af;
            margin: 10px 1%;
        }

        .recommendation-box {
            background: #dcfce7;
            border: 2px solid #16a34a;
            padding: 20px;
            margin: 20px 0;
            page-break-inside: avoid;
        }

        .risk-warning {
            background: #fee2e2;
            border-left: 4px solid #dc2626;
            padding: 15px;
            margin: 15px 0;
        }

        table {
            width: 100%;
            border-collapse: collapse;
            margin: 20px 0;
        }

        th, td {
            border: 1px solid #ddd;
            padding: 12px;
            text-align: left;
        }

        th {
            background-color: #1e40af;
            color: white;
        }
    </style>
</head>
<body>
    <!-- Cover Page -->
```

```
<div class="cover-page">
    <h1>{{ company_name }}</h1>
    <h2>Equity Research Report</h2>
    <p style="font-size: 14pt; margin-top: 50px;">
        Ticker: <strong>{{ ticker }}</strong><br>
        Exchange: {{ exchange }}<br>
        Report Date: {{ report_date }}
    </p>
    <img src="{{ company_logo }}" style="max-width: 200px; margin-top: 50px;">
    <p style="position: absolute; bottom: 50px; width: 100%; text-align: center;">
        Generated by Terminal.AI<br>
        Powered by Advanced AI Analytics
    </p>
</div>

<!-- Executive Summary -->
<div class="section">
    <h2>Executive Summary</h2>

    <div class="recommendation-box">
        <h3 style="margin-top: 0;">Investment Recommendation: {{ recommendation }}</h3>
        <p><strong>Price Target:</strong> {{ price_target }} ({{ upside_potential }}% upside)</p>
        <p><strong>Current Price:</strong> {{ current_price }}</p>
    </div>

    {{ executive_summary_content }}

    <h3>Key Metrics at a Glance</h3>
    <div class="metric-card">
        <strong>Market Cap</strong><br>
        {{ market_cap }}
    </div>
    <div class="metric-card">
        <strong>P/E Ratio</strong><br>
        {{ pe_ratio }}
    </div>
    <div class="metric-card">
        <strong>5Y EPS CAGR</strong><br>
        {{ eps_cagr }}%
    </div>
</div>

<!-- Company Overview -->
<div class="section">
    <h2>Company Overview</h2>
    {{ company_overview_content }}
</div>
```

```
    <!-- Financial Analysis -->
    <div class="section">
        <h2>Financial Analysis</h2>
        {{ financial_analysis_content }}

        <img src="{{ chart_revenue_eps }}" class="chart">
        <img src="{{ chart_profit_margin }}" class="chart">
    </div>

    <!-- Competitive Moat -->
    <div class="section">
        <h2>Competitive Moat Analysis</h2>
        <p><strong>Moat Rating:</strong> {{ moat_rating }} ({{ moat_strength }}/10)</p>
        {{ moat_analysis_content }}
    </div>

    <!-- Risk Assessment -->
    <div class="section">
        <h2>Risk Assessment</h2>
        {{ risk_assessment_content }}

        <div class="risk-warning">
            <strong>⚠ Key Risks:</strong>
            <ul>
            {% for risk in key_risks %}
                <li>{{ risk }}</li>
            {% endfor %}
            </ul>
        </div>
    </div>

    <!-- Investment Thesis -->
    <div class="section">
        <h2>Investment Thesis</h2>
        {{ investment_thesis_content }}
    </div>

    <!-- Disclaimer -->
    <div class="section" style="font-size: 9pt; color: #666; border-top: 1px solid #ccc; padding-top: 20px;">
        <h3>Disclaimer</h3>
        <p>This report is generated by AI and is for informational purposes only. It should not be considered as investment advic
    </div>
</body>
</html>
```

6.2 PDF Generation Code

python

```python
from weasyprint import HTML, CSS
from jinja2 import Template
import base64

async def pdf_generation_node(state: ReportState) -> ReportState:
    """
    Generate PDF from report sections
    """
    ticker = state["ticker"]

    # Load HTML template
    with open("report_template.html", "r") as f:
        template = Template(f.read())

    # Embed charts as base64
    chart_data = {}
    for chart_path in state["chart_paths"]:
        with open(chart_path, "rb") as img_file:
            chart_data[chart_path] = base64.b64encode(img_file.read()).decode()

    # Prepare template context
    context = {
        "ticker": ticker,
        "company_name": state["company_profile"]["company_name"],
```