**Mingus Development & Implementation Strategy (Updated)**

*Comprehensive Guide Starting from Week 9 - Post-MVP Foundation*

**Executive Summary**

Mingus has completed its foundational MVP development phase and is ready for feature enhancement and market launch. This strategy covers the next steps from Week 9 onward, including advanced feature development, landing page creation, marketing implementation, and user acquisition for African American professionals aged 25-35.

**Table of Contents**

1. [Current Status & Completed Foundation](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#current-status--completed-foundation)
2. [Development Roadmap (Week 9+)](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#development-roadmap-week-9)
3. [Landing Page Strategy & Implementation](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#landing-page-strategy--implementation)
4. [Job Security Feature Integration](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#job-security-feature-integration)
5. [Marketing & User Acquisition](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#marketing--user-acquisition)
6. [Testing & Validation Framework](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#testing--validation-framework)
7. [Implementation Timeline (Weeks 9-24)](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#implementation-timeline-weeks-9-24)
8. [Resource Requirements & Budget](https://claude.ai/chat/048d4ace-64fa-4457-ba45-5dfb550d6eae#resource-requirements--budget)

**Current Status & Completed Foundation**

**✅ Completed MVP Features**

* **Cash balance forecasting** with key milestone dates
* **Enhanced date generation system** (12-month horizon with business day logic)
* **Physical and mental health tracking** with weekly check-ins
* **Health-spending correlation analysis** showing impact on financial decisions
* **Basic career stress tracking** (foundation for job security features)
* **Optimized Supabase schema** with proper table structure
* **Fixed daily\_cashflow calculation** with income\_dates and expense\_dates tables

**Current Database Schema Status**

-- Operational Tables

user\_income\_due\_dates ✅ Working

user\_expense\_due\_dates ✅ Working

daily\_cashflow ✅ Fixed and operational

health\_checkins ✅ Implemented

relationship\_checkins ✅ Implemented

-- Ready for Enhancement

daily\_cashflow columns: forecast\_date, opening\_balance, income, expenses,

closing\_balance, net\_change, balance\_status, user\_id, id

**Technical Stack Status**

* **Backend**: Python/Django with Cursor AI assistance ✅
* **Database**: Supabase with optimized schema ✅
* **Frontend**: Vanilla JavaScript foundation ✅
* **Payment Processing**: Ready for Cash App integration
* **APIs**: Plaid integration planned, SEC EDGAR ready for implementation

**Development Roadmap (Week 9+)**

Based on my analysis of the codebase, here's a comprehensive outline of the missing screens and processes needed for a complete new user onboarding flow:

**Missing Screens and Processes for Complete New User Onboarding**

**1. Landing Page & Entry Points ✅ (EXISTS)**

* **Current**: landing.html - Main marketing page
* **Missing**: Direct link from landing page to registration/login

**2. Authentication Flow ✅ (EXISTS)**

* **Current**:
* templates/login.html - Login form
* templates/register.html - Registration form
* backend/routes/auth.py - Authentication endpoints
* **Status**: Functional but needs better integration

**3. Onboarding Questionnaire Flow ⚠️ (PARTIALLY EXISTS)**

* **Current**:
* backend/routes/onboarding.py - Backend endpoints
* src/components/questionnaires/QuestionnaireFlow.tsx - React components
* **Missing Screens**:
* **Welcome/Introduction Screen** - Explain the onboarding process
* **Profile Setup Screen** - Basic user information (age, location, employment)
* **Financial Profile Screen** - Income, expenses, debt, savings
* **Goals Setting Screen** - Financial goals and timelines
* **Health & Wellness Screen** - Initial health assessment
* ***Preferences Screen****- Risk tolerance, investment experience*
* **Data Input Screen** - Manual expense categories setup

**4. Questionnaire Templates ⚠️ (PARTIALLY EXISTS)**

* **Current**: React components for health, relationship, career questionnaires
* **Missing**:
* **Financial Questionnaire Template** - Income, expenses, debt structure
* **Lifestyle Questionnaire Template** - Spending habits, stress triggers
* **Goal Setting Template** - Short-term and long-term financial goals
* **Progress Tracking Template** - Onboarding completion status

**5. Data Input & Integration Screens ❌ (MISSING)**

* **Missing Screens**:
* **Bank Account Connection Screen** - Link financial accounts
* **Manual Data Entry Screen** - For users who don't want to connect accounts
* **Expense Categorization Screen** - Set up spending categories
* **Income Sources Screen** - Multiple income streams
* **Debt Management Screen** - Current debts and payment plans

**Here’s  what’s missing:**

**What’s MISSING or NOT INTEGRATED**

**Frontend**

1. **No Navigation/Links from Progress Tracker to Data Input Screens**

* The progress tracker does not allow users to click a step and go to the corresponding input screen.
* There are no React routes/pages for each onboarding step (profile, preferences, expenses, etc.)—only the goals setup and questionnaires are present.

1. **No React Forms for:**

* **Profile Input** (basic info, income, etc.)
* **Preferences Input** (risk tolerance, investment experience, etc.)
* **Expenses Input** (expense categories)
* **Financial Profile** (unless handled by a questionnaire)
* **Expense Profile** (unless handled by a questionnaire)
* **Financial Goals** (beyond the goals setup)
* **Onboarding Choice** (Go Deep vs. Keep it Brief)

1. **No Routing/Flow Logic**

* There is no React router logic to move the user through the onboarding steps based on progress or completion.
* No “Next Step” or “Continue” logic tied to backend progress.

**Backend**

* **All endpoints and templates exist.** The backend is ready for all onboarding steps.

**Summary Table**

| **Step** | **Backend Endpoint** | **Template Exists** | **React Page Exists** | **Integrated in Flow?** |
| --- | --- | --- | --- | --- |
| Welcome | Yes | Yes | No | No |
| Profile | Yes | Yes | No | No |
| Preferences | Yes | Yes | No | No |
| Expenses | Yes | Yes | No | No |
| Goals | Yes | Yes | Yes (GoalsSetup) | Partially |
| Financial Questionnaire | Yes | Yes | Yes (QuestionnaireFlow) | Partially |
| Lifestyle Questionnaire | Yes | Yes | Yes (QuestionnaireFlow) | Partially |
| Onboarding Choice | Yes | Yes | No | No |
| Progress Tracker | Yes | N/A | Yes | Yes (display only) |

**What You Need to Add**

1. **React pages/components for each onboarding step:**

* Profile input form
* Preferences input form
* Expenses input form
* Welcome/intro screen
* Onboarding choice screen

1. **Routing and navigation logic:**

* Use React Router to move between steps.
* Allow clicking steps in the progress tracker to jump to the relevant screen.

1. **API integration for each form:**

* POST data to the corresponding backend endpoint on submit.

1. **“Next Step” logic:**

* After submitting a step, update progress and move to the next step.

**Would you like a starter template for the missing React forms and routing logic?**Or do you want to focus on a specific step first (e.g., profile input or onboarding choice)?

**6. Dashboard Access & Navigation ✅ (EXISTS)**

* **Current**:
* backend/templates/dashboard.html - Main dashboard
* dashboard.html - Alternative dashboard view
* **Status**: Functional but needs onboarding completion check

**7.**

**B. User Data**

**sql**

**Apply to business\_int...**

*-- Missing tables for user data*

CREATE TABLE user\_financial\_data (

    id INTEGER PRIMARY KEY,

    user\_id TEXT NOT NULL,

    income\_sources TEXT,

    expense\_categories TEXT,

    debt\_info TEXT,

    savings\_info TEXT,

    created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

**12. Missing Validation & Error Handling**

**A. Form Validation**

* Client-side validation for all onboarding forms
* Server-side validation for data integrity
* Progress saving for incomplete onboarding

**B. Error Recovery**

* Offline support for onboarding completion
* Data recovery for interrupted sessions
* Graceful error handling and user feedback

**13. Missing Integration Points**

**A. Authentication Integration**

* Seamless flow from registration to onboarding
* Session management during onboarding
* Progress persistence across sessions

**B. Dashboard Integration**

* Onboarding completion check before dashboard access
* Personalized dashboard based on onboarding data
* Quick access to complete incomplete onboarding

**Priority Implementation Order**

1. **High Priority**: Welcome screen, profile setup, financial profile
2. **Medium Priority**: Goals setup, health assessment, data input
3. **Low Priority**: Preferences, dashboard intro, advanced features

This outline provides a complete roadmap for implementing the missing screens and processes needed for a seamless new user onboarding experience from sign-up to dashboard access.

Review changes

1 file

+8-5

business\_intelligence.py

**Phase 1: Advanced Features Development (Weeks 9-12)**

*Budget: $1,400 from remaining funds*

**Week 9-10: Job Security Analytics Foundation**

**Goal**: Add comprehensive job security scoring to existing career stress tracking

**Development Tasks:**

1. **Enhanced Career Questions Integration**
2. # Extend existing career stress check-in
3. enhanced\_career\_checkin = {
4. 'job\_stress\_level': existing\_question, # Already implemented
5. 'job\_security\_feeling': 'How secure do you feel in your current position? (1-10)',
6. 'employer\_warnings': 'Has your employer mentioned layoffs or budget cuts recently?',
7. 'employer\_name': 'Current employer name (for financial analysis)',
8. 'industry\_sector': 'What industry do you work in?'
9. }
10. **Basic Job Security Score Algorithm**
11. def calculate\_job\_security\_score(user\_responses, external\_data=None):
12. """Enhanced scoring combining user input with external data"""
13. base\_score = 100
15. # User perception factors (60% weight)
16. stress\_impact = (user\_responses['job\_stress\_level'] - 1) \* 4
17. security\_feeling = (user\_responses['job\_security\_feeling'] - 1) \* 6
18. warning\_penalty = 25 if user\_responses['employer\_warnings'] == 'Yes' else 0
20. user\_score = base\_score - stress\_impact + security\_feeling - warning\_penalty
22. # External data factors (40% weight) - when available
23. if external\_data:
24. financial\_health = external\_data.get('employer\_financial\_health', 50)
25. industry\_risk = external\_data.get('industry\_risk\_score', 50)
26. local\_layoffs = external\_data.get('local\_layoff\_activity', 0)
28. external\_score = (financial\_health + industry\_risk - (local\_layoffs \* 2)) / 2
29. final\_score = (user\_score \* 0.6) + (external\_score \* 0.4)
30. else:
31. final\_score = user\_score
33. return max(0, min(100, final\_score))
34. **Job Security Dashboard Card**
35. <!-- Add to existing dashboard alongside health correlations -->
36. <div class="dashboard-card job-security-card">
37. <h3>📊 Job Security Analysis</h3>
39. <div class="security-score-circle">
40. <div class="score-value" id="job-security-score">85/100</div>
41. <div class="score-label">Security Score</div>
42. </div>
44. <div class="security-factors">
45. <div class="factor">
46. <span class="factor-label">Your Confidence:</span>
47. <span class="factor-value positive" id="confidence-score">+42</span>
48. </div>
49. <div class="factor">
50. <span class="factor-label">Stress Impact:</span>
51. <span class="factor-value negative" id="stress-impact">-16</span>
52. </div>
53. <div class="factor">
54. <span class="factor-label">Warning Signs:</span>
55. <span class="factor-value" id="warning-status">None detected</span>
56. </div>
57. </div>
59. <div class="recommendations">
60. <h4>💡 Job Security Tips</h4>
61. <ul id="job-security-recommendations">
62. <!-- Dynamically populated -->
63. </ul>
64. </div>
65. </div>

**Upwork Allocation**: $400 for advanced job security algorithms and UI components

**Week 11-12: SEC EDGAR Integration & Enhanced Analytics**

**Goal**: Add employer financial health analysis using free government data

**Development Tasks:**

1. **SEC EDGAR API Integration**
2. class EmployerAnalyzer:
3. def \_\_init\_\_(self):
4. self.sec\_base\_url = "https://data.sec.gov/api/xbrl/"
6. def analyze\_employer\_financial\_health(self, employer\_name):
7. """Get employer financial health from SEC filings"""
8. try:
9. # Find company CIK (Central Index Key)
10. cik = self.find\_company\_cik(employer\_name)
11. if not cik:
12. return self.handle\_private\_company(employer\_name)
14. # Get latest financial facts
15. financial\_url = f"{self.sec\_base\_url}companyfacts/{cik}.json"
16. response = requests.get(financial\_url, headers={'User-Agent': 'Mingus App'})
18. if response.status\_code == 200:
19. data = response.json()
20. return self.calculate\_financial\_health\_score(data)
21. else:
22. return {'status': 'no\_data', 'score': 50}
24. except Exception as e:
25. return {'status': 'error', 'score': 50, 'message': str(e)}
27. def calculate\_financial\_health\_score(self, sec\_data):
28. """Calculate financial health score from SEC data"""
29. try:
30. facts = sec\_data.get('facts', {})
32. # Key financial indicators
33. revenue\_data = self.extract\_revenue\_trend(facts)
34. debt\_data = self.extract\_debt\_metrics(facts)
35. cash\_data = self.extract\_cash\_position(facts)
37. # Calculate component scores
38. revenue\_score = self.score\_revenue\_trend(revenue\_data)
39. debt\_score = self.score\_debt\_health(debt\_data)
40. cash\_score = self.score\_cash\_position(cash\_data)
42. # Weighted average
43. overall\_score = (revenue\_score \* 0.4 + debt\_score \* 0.3 + cash\_score \* 0.3)
45. return {
46. 'status': 'success',
47. 'overall\_score': overall\_score,
48. 'components': {
49. 'revenue\_health': revenue\_score,
50. 'debt\_health': debt\_score,
51. 'cash\_position': cash\_score
52. },
53. 'last\_filing': self.get\_last\_filing\_date(sec\_data)
54. }
56. except Exception as e:
57. return {'status': 'calculation\_error', 'score': 50}
58. **State WARN Database Integration**
59. def get\_local\_warn\_activity(self, user\_zip\_code, radius\_miles=25):
60. """Check for recent layoff activity in user's area"""
61. user\_state = self.get\_state\_from\_zip(user\_zip\_code)
63. # Priority states with good APIs/data
64. warn\_handlers = {
65. 'CA': self.fetch\_california\_warn\_data,
66. 'WA': self.fetch\_washington\_warn\_data,
67. 'MD': self.fetch\_maryland\_warn\_data,
68. 'IL': self.fetch\_illinois\_warn\_data
69. }
71. if user\_state in warn\_handlers:
72. try:
73. local\_layoffs = warn\_handlers[user\_state](user\_zip\_code, radius\_miles)
74. return self.analyze\_local\_layoff\_trends(local\_layoffs)
75. except Exception as e:
76. return {'status': 'no\_local\_data', 'layoff\_count': 0}
78. return {'status': 'state\_not\_supported', 'layoff\_count': 0}

**Upwork Allocation**: $500 for SEC API integration and WARN data scraping

**Phase 2: Special Events & Enhanced Forecasting (Weeks 13-16)**

*Budget: $1,000 from remaining funds*

**Week 13-14: Special Events Planning System**

**Goal**: Add milestone planning with red/green coverage indicators

**Database Enhancement:**

-- Add special events tables to existing schema

CREATE TABLE user\_special\_events (

id uuid PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id uuid REFERENCES auth.users(id) NOT NULL,

event\_name text NOT NULL,

event\_type text CHECK (event\_type IN (

'child\_birthday', 'spouse\_birthday', 'parent\_birthday', 'anniversary',

'solo\_vacation', 'group\_trip\_girls', 'group\_trip\_boys', 'wedding',

'tax\_refund', 'property\_tax', 'car\_registration', 'holiday\_spending'

)),

event\_date date NOT NULL,

estimated\_amount decimal(10,2) NOT NULL,

frequency text DEFAULT 'annually',

priority\_level integer DEFAULT 3 CHECK (priority\_level BETWEEN 1 AND 5),

is\_active boolean DEFAULT true,

created\_at timestamp DEFAULT now()

);

-- Enhance daily\_cashflow table

ALTER TABLE daily\_cashflow

ADD COLUMN special\_events\_expense decimal(12,2) DEFAULT 0.00,

ADD COLUMN uncovered\_events\_count integer DEFAULT 0;

**Special Events Interface:**

// Special Events Manager

class SpecialEventsManager {

constructor(userId) {

this.userId = userId;

}

async addSpecialEvent(eventData) {

const event = {

user\_id: this.userId,

event\_name: eventData.name,

event\_type: eventData.type,

event\_date: eventData.date,

estimated\_amount: parseFloat(eventData.amount),

priority\_level: eventData.priority || 3

};

try {

const response = await fetch('/api/special-events', {

method: 'POST',

headers: {'Content-Type': 'application/json'},

body: JSON.stringify(event)

});

if (response.ok) {

await this.refreshCashFlowForecast();

this.displayEvents();

return {success: true};

}

} catch (error) {

return {success: false, error: error.message};

}

}

displayEvents() {

const container = document.getElementById('special-events-list');

fetch(`/api/special-events/${this.userId}`)

.then(response => response.json())

.then(events => {

let html = '';

events.forEach(event => {

const statusClass = this.getEventStatusClass(event.coverage\_status);

const statusIcon = this.getEventStatusIcon(event.coverage\_status);

html += `

<div class="event-card ${statusClass}">

<div class="event-header">

<span class="event-status">${statusIcon}</span>

<h4>${event.event\_name}</h4>

<span class="event-amount">$${event.estimated\_amount.toLocaleString()}</span>

</div>

<div class="event-date">${this.formatDate(event.event\_date)}</div>

<div class="event-countdown">${this.getCountdown(event.event\_date)}</div>

</div>

`;

});

container.innerHTML = html;

});

}

getEventStatusClass(status) {

switch(status) {

case 'covered': return 'status-green';

case 'at\_risk': return 'status-yellow';

case 'uncovered': return 'status-red';

default: return 'status-gray';

}

}

getEventStatusIcon(status) {

switch(status) {

case 'covered': return '✅';

case 'at\_risk': return '⚠️';

case 'uncovered': return '❌';

default: return '❓';

}

}

}

**Week 15-16: Enhanced Cash Flow with Special Events**

**Goal**: Integrate special events into cash flow forecasting with coverage analysis

**Enhanced Cash Flow Calculator:**

def calculate\_enhanced\_daily\_cashflow(user\_id, days\_ahead=365):

"""Enhanced calculation including special events and job security factors"""

# Get base data

opening\_balance = get\_user\_opening\_balance(user\_id)

income\_schedule = get\_income\_due\_dates(user\_id, days\_ahead)

expense\_schedule = get\_expense\_due\_dates(user\_id, days\_ahead)

special\_events = get\_special\_events\_schedule(user\_id, days\_ahead)

# Calculate daily balances

daily\_records = []

current\_balance = opening\_balance

current\_date = datetime.now()

for day in range(days\_ahead):

date\_str = current\_date.strftime("%Y-%m-%d")

# Calculate day's transactions

day\_income = sum(i['amount'] for i in income\_schedule if i['due\_date'] == date\_str)

day\_expenses = sum(e['amount'] for e in expense\_schedule if e['due\_date'] == date\_str)

day\_events = sum(e['amount'] for e in special\_events if e['event\_date'] == date\_str)

# Calculate balances

total\_outflow = day\_expenses + day\_events

net\_change = day\_income - total\_outflow

closing\_balance = current\_balance + net\_change

# Analyze special events coverage

events\_today = [e for e in special\_events if e['event\_date'] == date\_str]

uncovered\_events = [e for e in events\_today if closing\_balance < e['amount']]

# Determine status

if closing\_balance < 0:

status = 'critical'

elif uncovered\_events:

status = 'at\_risk'

elif closing\_balance < 500: # Configurable threshold

status = 'caution'

else:

status = 'positive'

daily\_record = {

'user\_id': user\_id,

'forecast\_date': date\_str,

'opening\_balance': current\_balance,

'income': day\_income,

'expenses': day\_expenses,

'special\_events\_expense': day\_events,

'closing\_balance': closing\_balance,

'net\_change': net\_change,

'balance\_status': status,

'uncovered\_events\_count': len(uncovered\_events)

}

daily\_records.append(daily\_record)

current\_balance = closing\_balance

current\_date += timedelta(days=1)

# Update special events coverage status

update\_events\_coverage\_status(user\_id, special\_events, daily\_records)

return daily\_records

**Upwork Allocation**: $600 for special events system and enhanced forecasting

**Landing Page Strategy & Implementation**

**Landing Page Objectives**

1. **Convert visitors to email leads** (primary goal)
2. **Communicate unique value proposition** (health + finance integration)
3. **Build trust** with target demographic
4. **Drive trial signups** across all tiers

**Landing Page Structure & Script**

**Above-the-Fold Section**

<!-- Hero Section -->

<section class="hero-section">

<div class="hero-content">

<h1 class="hero-headline">

Finally, A Finance App That Gets

<span class="highlight">Your Real Life</span>

</h1>

<p class="hero-subheadline">

Mingus connects your health, relationships, and career stress

to your financial decisions. See how your whole life affects

your money—and plan accordingly.

</p>

<div class="hero-cta">

<button class="cta-primary" onclick="openTrialModal()">

Start Your Free Trial

</button>

<p class="cta-subtext">45-day free trial • No credit card required</p>

</div>

<div class="social-proof">

<p>"The only finance app that actually understands my life as a young professional"</p>

<span class="testimonial-author">- Marcus, Atlanta</span>

</div>

</div>

<div class="hero-visual">

<div class="phone-mockup">

<!-- Animated dashboard preview -->

<img src="/images/dashboard-preview.png" alt="Mingus Dashboard" />

</div>

</div>

</section>

**Problem Agitation Section**

<section class="problem-section">

<div class="container">

<h2>Tired of Finance Apps That Don't Get It?</h2>

<div class="problems-grid">

<div class="problem-card">

<div class="problem-icon">😤</div>

<h3>Generic Advice</h3>

<p>Other apps give you cookie-cutter budgets that ignore your real life and cultural context.</p>

</div>

<div class="problem-card">

<div class="problem-icon">🤯</div>

<h3>Stress Spending</h3>

<p>You know job stress affects your spending, but no app helps you plan for it.</p>

</div>

<div class="problem-card">

<div class="problem-icon">🎯</div>

<h3>Missing the Big Picture</h3>

<p>Your money decisions involve relationships, health, and career—but apps only track dollars.</p>

</div>

</div>

</div>

</section>

**Solution/Features Section**

<section class="solution-section">

<div class="container">

<h2>Mingus: Financial Planning That Fits Your Life</h2>

<div class="features-showcase">

<div class="feature-item">

<div class="feature-visual">

<img src="/images/health-correlation.png" alt="Health Correlation" />

</div>

<div class="feature-content">

<h3>🏃‍♀️ Health + Money Connection</h3>

<p>See how your stress levels, physical activity, and mental health impact your spending patterns. Plan better by understanding your triggers.</p>

<ul>

<li>Weekly health check-ins</li>

<li>Spending correlation analysis</li>

<li>Stress-based budget adjustments</li>

</ul>

</div>

</div>

<div class="feature-item reverse">

<div class="feature-visual">

<img src="/images/job-security.png" alt="Job Security Analysis" />

</div>

<div class="feature-content">

<h3>📊 Job Security Analytics</h3>

<p>Know your employment risk before it hits. We analyze your employer's financial health and local layoff trends.</p>

<ul>

<li>Employer financial health scoring</li>

<li>Local layoff activity monitoring</li>

<li>Emergency fund recommendations</li>

</ul>

</div>

</div>

<div class="feature-item">

<div class="feature-visual">

<img src="/images/milestone-planning.png" alt="Milestone Planning" />

</div>

<div class="feature-content">

<h3>🎯 Life Milestone Planning</h3>

<p>Plan for what matters: birthdays, anniversaries, vacations, and major life events. See if you'll have the funds when you need them.</p>

<ul>

<li>Visual coverage indicators (green/red)</li>

<li>Automatic milestone forecasting</li>

<li>Priority-based planning</li>

</ul>

</div>

</div>

</div>

</div>

</section>

**Social Proof Section**

<section class="social-proof-section">

<div class="container">

<h2>Join Hundreds of Young Professionals Taking Control</h2>

<div class="testimonials-grid">

<div class="testimonial-card">

<div class="testimonial-content">

<p>"Finally! An app that gets that my relationship affects my spending. The milestone planning saved my anniversary."</p>

</div>

<div class="testimonial-author">

<img src="/images/avatar-1.jpg" alt="Testimonial" />

<div>

<h4>Jasmine K.</h4>

<span>Marketing Manager, Atlanta</span>

</div>

</div>

</div>

<div class="testimonial-card">

<div class="testimonial-content">

<p>"The job security feature helped me see the risk at my company before layoffs happened. Built up my emergency fund just in time."</p>

</div>

<div class="testimonial-author">

<img src="/images/avatar-2.jpg" alt="Testimonial" />

<div>

<h4>Marcus T.</h4>

<span>Software Engineer, Houston</span>

</div>

</div>

</div>

<div class="testimonial-card">

<div class="testimonial-content">

<p>"Love how it shows me when I stress-spend. Now I can budget for those rough weeks at work."</p>

</div>

<div class="testimonial-author">

<img src="/images/avatar-3.jpg" alt="Testimonial" />

<div>

<h4>Alicia R.</h4>

<span>Nurse, Dallas</span>

</div>

</div>

</div>

</div>

<div class="stats-row">

<div class="stat">

<h3>87%</h3>

<p>Improve their spending awareness in 30 days</p>

</div>

<div class="stat">

<h3>$1,247</h3>

<p>Average amount saved in first year</p>

</div>

<div class="stat">

<h3>93%</h3>

<p>Successfully plan for major life events</p>

</div>

</div>

</div>

</section>

**Pricing Section**

<section class="pricing-section">

<div class="container">

<h2>Choose Your Financial Wellness Journey</h2>

<div class="pricing-grid">

<div class="pricing-card">

<div class="card-header">

<h3>Essentials</h3>

<div class="price">

<span class="currency">$</span>

<span class="amount">12</span>

<span class="period">/month</span>

</div>

<p class="price-subtitle">Perfect for getting started</p>

</div>

<ul class="features-list">

<li>✅ 12-month cash flow forecasting</li>

<li>✅ Health-spending correlations</li>

<li>✅ Basic job security score</li>

<li>✅ Milestone planning (5 events)</li>

<li>✅ Weekly health check-ins</li>

<li>✅ Email support</li>

</ul>

<button class="cta-secondary" onclick="selectPlan('essentials')">

Start 45-Day Free Trial

</button>

</div>

<div class="pricing-card featured">

<div class="popular-badge">Most Popular</div>

<div class="card-header">

<h3>Professional</h3>

<div class="price">

<span class="currency">$</span>

<span class="amount">25</span>

<span class="period">/month</span>

</div>

<p class="price-subtitle">For ambitious professionals</p>

</div>

<ul class="features-list">

<li>✅ Everything in Essentials, plus:</li>

<li>✅ 24-month forecasting</li>

<li>✅ Advanced job security analytics</li>

<li>✅ Unlimited milestone planning</li>

<li>✅ Employer financial health analysis</li>

<li>✅ Weekly personalized insights</li>

<li>✅ Priority support</li>

</ul>

<button class="cta-primary" onclick="selectPlan('professional')">

Start 30-Day Free Trial

</button>

</div>

<div class="pricing-card">

<div class="card-header">

<h3>Executive</h3>

<div class="price">

<span class="currency">$</span>

<span class="amount">45</span>

<span class="period">/month</span>

</div>

<p class="price-subtitle">Complete optimization</p>

</div>

<ul class="features-list">

<li>✅ Everything in Professional, plus:</li>

<li>✅ 36-month forecasting</li>

<li>✅ Industry layoff trend analysis</li>

<li>✅ Monthly 1-on-1 coaching (15 min)</li>

<li>✅ Custom financial reports</li>

<li>✅ Phone support</li>

<li>✅ Executive peer network access</li>

</ul>

<button class="cta-secondary" onclick="selectPlan('executive')">

Start 60-Day Free Trial

</button>

</div>

</div>

<p class="pricing-guarantee">

💰 30-day money-back guarantee • 🔒 Cancel anytime • 📱 Works on all devices

</p>

</div>

</section>

**FAQ Section**

<section class="faq-section">

<div class="container">

<h2>Frequently Asked Questions</h2>

<div class="faq-grid">

<div class="faq-item">

<h3>How is Mingus different from other budgeting apps?</h3>

<p>Mingus is the only app that connects your health, relationships, and career stress to your financial decisions. We provide personalized insights based on your complete life, not just your transactions.</p>

</div>

<div class="faq-item">

<h3>Is my financial data secure?</h3>

<p>Absolutely. We use bank-level encryption and never store your banking credentials. We're also SOC 2 compliant and follow all financial data protection standards.</p>

</div>

<div class="faq-item">

<h3>How does the job security analysis work?</h3>

<p>We analyze your employer's financial health using public SEC filings and monitor local layoff activity through government WARN databases. This gives you early warning of potential employment risks.</p>

</div>

<div class="faq-item">

<h3>Can I cancel anytime?</h3>

<p>Yes! Cancel anytime with no fees or penalties. You'll continue to have access until your current billing period ends.</p>

</div>

<div class="faq-item">

<h3>Do you offer student discounts?</h3>

<p>Yes! Students get 25% off the Essentials plan with valid student email verification.</p>

</div>

<div class="faq-item">

<h3>What if I need help getting started?</h3>

<p>Professional and Executive plans include onboarding sessions. All users get access to our comprehensive help center and email support.</p>

</div>

</div>

</div>

</section>

**Final CTA Section**

<section class="final-cta-section">

<div class="container">

<div class="cta-content">

<h2>Ready to Take Control of Your Complete Financial Picture?</h2>

<p>Join hundreds of young professionals who've already transformed their relationship with money.</p>

<div class="cta-buttons">

<button class="cta-primary large" onclick="openTrialModal()">

Start Your Free Trial Today

</button>

<button class="cta-secondary large" onclick="downloadEbook()">

Download Free E-book

</button>

</div>

<p class="cta-disclaimer">

No credit card required • Free e-book: "The Holistic Money Method"

</p>

</div>

</div>

</section>

**Landing Page Technical Implementation**

**Lead Capture Modal**

function openTrialModal() {

const modal = document.getElementById('trial-modal');

modal.style.display = 'flex';

// Track modal open event

gtag('event', 'trial\_modal\_opened', {

event\_category: 'engagement',

event\_label: 'free\_trial\_interest'

});

}

function submitTrialForm() {

const formData = {

name: document.getElementById('trial-name').value,

email: document.getElementById('trial-email').value,

plan: document.getElementById('selected-plan').value,

source: 'landing\_page'

};

fetch('/api/trial-signup', {

method: 'POST',

headers: {'Content-Type': 'application/json'},

body: JSON.stringify(formData)

})

.then(response => response.json())

.then(data => {

if (data.success) {

// Redirect to onboarding

window.location.href = `/onboarding?email=${formData.email}&plan=${formData.plan}`;

} else {

showError('Something went wrong. Please try again.');

}

})

.catch(error => {

showError('Network error. Please check your connection.');

});

}

**Analytics Integration**

// Google Analytics 4 Events

function trackLandingPageEvents() {

// Track scroll depth

let scrollDepths = [25, 50, 75, 100];

let trackedDepths = [];

window.addEventListener('scroll', () => {

const scrollPercent = Math.round((window.scrollY / (document.body.scrollHeight - window.innerHeight)) \* 100);

scrollDepths.forEach(depth => {

if (scrollPercent >= depth && !trackedDepths.includes(depth)) {

trackedDepths.push(depth);

gtag('event', 'scroll\_depth', {

event\_category: 'engagement',

event\_label: `${depth}%`,

value: depth

});

}

});

});

// Track pricing card interactions

document.querySelectorAll('.pricing-card').forEach(card => {

card.addEventListener('click', (e) => {

const planName = card.querySelector('h3').textContent;

gtag('event', 'pricing\_card\_click', {

event\_category: 'conversion',

event\_label: planName.toLowerCase(),

plan\_type: planName.toLowerCase()

});

});

});

// Track feature section views

const observerOptions = {

threshold: 0.5,

rootMargin: '0px 0px -100px 0px'

};

const sectionObserver = new IntersectionObserver((entries) => {

entries.forEach(entry => {

if (entry.isIntersecting) {

const sectionName = entry.target.className.split('-')[0];

gtag('event', 'section\_view', {

event\_category: 'engagement',

event\_label: sectionName

});

}

});

}, observerOptions);

document.querySelectorAll('section').forEach(section => {

sectionObserver.observe(section);

});

}

**Landing Page CSS Framework**

/\* Mingus Landing Page Styles \*/

:root {

--mingus-primary: #1a1a2e;

--mingus-secondary: #16213e;

--mingus-accent: #00d9ff;

--mingus-success: #4caf50;

--mingus-warning: #ffa726;

--mingus-danger: #ff4757;

--mingus-gray-800: #1a1a1a;

--mingus-gray-600: #404040;

--mingus-gray-300: #cccccc;

--mingus-white: #ffffff;

}

/\* Hero Section \*/

.hero-section {

background: linear-gradient(135deg, var(--mingus-primary) 0%, var(--mingus-secondary) 100%);

color: var(--mingus-white);

padding: 100px 0;

display: flex;

align-items: center;

min-height: 100vh;

}

.hero-headline {

font-size: 3.5rem;

font-weight: 700;

line-height: 1.2;

margin-bottom: 1.5rem;

}

.highlight {

color: var(--mingus-accent);

background: linear-gradient(135deg, var(--mingus-accent), #4dffff);

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

}

.hero-subheadline {

font-size: 1.25rem;

color: #b0b0b0;

line-height: 1.6;

margin-bottom: 2rem;

max-width: 600px;

}

.cta-primary {

background: linear-gradient(135deg, var(--mingus-accent), #4dffff);

color: var(--mingus-primary);

padding: 18px 36px;

border: none;

border-radius: 12px;

font-size: 1.1rem;

font-weight: 600;

cursor: pointer;

transition: all 0.3s ease;

box-shadow: 0 4px 16px rgba(0, 217, 255, 0.3);

}

.cta-primary:hover {

transform: translateY(-2px);

box-shadow: 0 8px 24px rgba(0, 217, 255, 0.4);

}

.cta-secondary {

background: transparent;

color: var(--mingus-accent);

border: 2px solid var(--mingus-accent);

padding: 16px 32px;

border-radius: 12px;

font-size: 1rem;

font-weight: 500;

cursor: pointer;

transition: all 0.3s ease;

}

.cta-secondary:hover {

background: var(--mingus-accent);

color: var(--mingus-primary);

}

/\* Problem Section \*/

.problem-section {

background: #f8f9fa;

padding: 80px 0;

}

.problems-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 2rem;

margin-top: 3rem;

}

.problem-card {

background: white;

padding: 2rem;

border-radius: 16px;

text-align: center;

box-shadow: 0 4px 16px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease;

}

.problem-card:hover {

transform: translateY(-4px);

}

.problem-icon {

font-size: 3rem;

margin-bottom: 1rem;

}

/\* Features Section \*/

.features-showcase {

margin-top: 3rem;

}

.feature-item {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 3rem;

align-items: center;

margin-bottom: 4rem;

padding: 2rem 0;

}

.feature-item.reverse {

direction: rtl;

}

.feature-item.reverse > \* {

direction: ltr;

}

.feature-visual img {

width: 100%;

max-width: 500px;

border-radius: 16px;

box-shadow: 0 8px 32px rgba(0, 0, 0, 0.15);

}

.feature-content h3 {

font-size: 1.75rem;

margin-bottom: 1rem;

color: var(--mingus-primary);

}

.feature-content ul {

list-style: none;

padding: 0;

margin-top: 1rem;

}

.feature-content li {

padding: 0.5rem 0;

color: #666;

}

.feature-content li:before {

content: "✓ ";

color: var(--mingus-success);

font-weight: bold;

margin-right: 0.5rem;

}

/\* Pricing Section \*/

.pricing-section {

background: var(--mingus-gray-800);

color: var(--mingus-white);

padding: 80px 0;

}

.pricing-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));

gap: 2rem;

margin-top: 3rem;

}

.pricing-card {

background: rgba(255, 255, 255, 0.05);

border: 1px solid rgba(255, 255, 255, 0.1);

border-radius: 20px;

padding: 2rem;

position: relative;

backdrop-filter: blur(10px);

transition: all 0.3s ease;

}

.pricing-card.featured {

border: 2px solid var(--mingus-accent);

transform: scale(1.05);

}

.pricing-card:hover {

transform: translateY(-4px);

box-shadow: 0 12px 48px rgba(0, 0, 0, 0.3);

}

.popular-badge {

position: absolute;

top: -12px;

left: 50%;

transform: translateX(-50%);

background: linear-gradient(135deg, var(--mingus-accent), #4dffff);

color: var(--mingus-primary);

padding: 6px 20px;

border-radius: 20px;

font-size: 0.875rem;

font-weight: 600;

}

.price {

display: flex;

align-items: baseline;

margin: 1rem 0;

}

.currency {

font-size: 1.5rem;

color: var(--mingus-accent);

}

.amount {

font-size: 3rem;

font-weight: 700;

color: var(--mingus-white);

margin: 0 0.25rem;

}

.period {

font-size: 1rem;

color: #b0b0b0;

}

.features-list {

list-style: none;

padding: 0;

margin: 2rem 0;

}

.features-list li {

padding: 0.75rem 0;

border-bottom: 1px solid rgba(255, 255, 255, 0.1);

}

/\* Social Proof Section \*/

.testimonials-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 2rem;

margin: 3rem 0;

}

.testimonial-card {

background: white;

padding: 2rem;

border-radius: 16px;

box-shadow: 0 4px 16px rgba(0, 0, 0, 0.1);

}

.testimonial-author {

display: flex;

align-items: center;

margin-top: 1.5rem;

gap: 1rem;

}

.testimonial-author img {

width: 50px;

height: 50px;

border-radius: 50%;

}

.stats-row {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 2rem;

text-align: center;

margin-top: 3rem;

padding-top: 3rem;

border-top: 1px solid #e0e0e0;

}

.stat h3 {

font-size: 2.5rem;

color: var(--mingus-accent);

margin-bottom: 0.5rem;

}

/\* Modal Styles \*/

.modal-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.8);

display: none;

justify-content: center;

align-items: center;

z-index: 1000;

}

.modal-card {

background: white;

padding: 3rem;

border-radius: 20px;

width: 90%;

max-width: 500px;

max-height: 90vh;

overflow-y: auto;

}

.form-group {

margin-bottom: 1.5rem;

}

.form-group label {

display: block;

margin-bottom: 0.5rem;

font-weight: 500;

color: var(--mingus-primary);

}

.form-group input,

.form-group select {

width: 100%;

padding: 12px;

border: 2px solid #e0e0e0;

border-radius: 8px;

font-size: 1rem;

transition: border-color 0.3s ease;

}

.form-group input:focus,

.form-group select:focus {

outline: none;

border-color: var(--mingus-accent);

box-shadow: 0 0 0 3px rgba(0, 217, 255, 0.1);

}

/\* Responsive Design \*/

@media (max-width: 768px) {

.hero-headline {

font-size: 2.5rem;

}

.hero-section {

padding: 60px 0;

text-align: center;

}

.feature-item {

grid-template-columns: 1fr;

text-align: center;

}

.feature-item.reverse {

direction: ltr;

}

.pricing-card.featured {

transform: none;

}

.problems-grid,

.testimonials-grid {

grid-template-columns: 1fr;

}

}

/\* Loading and Animation States \*/

.fade-in {

opacity: 0;

transform: translateY(20px);

transition: all 0.6s ease;

}

.fade-in.visible {

opacity: 1;

transform: translateY(0);

}

.cta-primary.loading {

position: relative;

color: transparent;

}

.cta-primary.loading::after {

content: "";

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

width: 20px;

height: 20px;

border: 2px solid transparent;

border-top: 2px solid var(--mingus-primary);

border-radius: 50%;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: translate(-50%, -50%) rotate(0deg); }

100% { transform: translate(-50%, -50%) rotate(360deg); }

}

**Landing Page Performance Optimization**

// Page load optimization

document.addEventListener('DOMContentLoaded', function() {

// Lazy load images

const images = document.querySelectorAll('img[data-src]');

const imageObserver = new IntersectionObserver((entries, observer) => {

entries.forEach(entry => {

if (entry.isIntersecting) {

const img = entry.target;

img.src = img.dataset.src;

img.classList.remove('lazy');

imageObserver.unobserve(img);

}

});

});

images.forEach(img => imageObserver.observe(img));

// Animate elements on scroll

const animateElements = document.querySelectorAll('.fade-in');

const animateObserver = new IntersectionObserver((entries) => {

entries.forEach(entry => {

if (entry.isIntersecting) {

entry.target.classList.add('visible');

}

});

}, { threshold: 0.1 });

animateElements.forEach(el => animateObserver.observe(el));

// Initialize tracking

trackLandingPageEvents();

});

**Job Security Feature Integration**

**Phase 1: Enhanced Career Questions (Week 9-10)**

Building on your existing career stress question, we'll add comprehensive job security analytics.

**Database Schema Updates**

-- Enhance existing career data

ALTER TABLE health\_checkins

ADD COLUMN job\_security\_feeling integer CHECK (job\_security\_feeling BETWEEN 1 AND 10),

ADD COLUMN employer\_warnings boolean DEFAULT false,

ADD COLUMN employer\_name text,

ADD COLUMN industry\_sector text;

-- New job security analysis table

CREATE TABLE job\_security\_analysis (

id uuid PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id uuid REFERENCES auth.users(id) NOT NULL,

analysis\_date date NOT NULL,

user\_perception\_score integer,

employer\_financial\_score integer,

local\_risk\_score integer,

overall\_job\_security\_score integer CHECK (overall\_job\_security\_score BETWEEN 0 AND 100),

recommendations text[],

last\_updated timestamp DEFAULT now()

);

**Enhanced Weekly Check-in Flow**

// Updated career section in weekly check-in

const enhancedCareerCheckin = {

section: 'career\_wellness',

questions: [

{

id: 'job\_stress\_level',

question: 'How stressful has work been this week?',

type: 'scale',

scale: '1-10 (1=No stress, 10=Extremely stressful)',

existing: true

},

{

id: 'job\_security\_feeling',

question: 'How secure do you feel in your current position?',

type: 'scale',

scale: '1-10 (1=Very insecure, 10=Very secure)',

new: true

},

{

id: 'employer\_warnings',

question: 'Have you heard talk of layoffs, budget cuts, or restructuring at work recently?',

type: 'multiple\_choice',

options: ['Yes, definitely', 'Some rumors/hints', 'No, nothing', 'Not sure'],

new: true

},

{

id: 'work\_environment\_changes',

question: 'Any major changes at work this week?',

type: 'multiple\_choice',

options: ['New management', 'Team changes', 'Policy updates', 'Budget discussions', 'Nothing notable'],

conditional: true, // Only if previous answers indicate concern

new: true

}

],

conditional\_questions: {

trigger\_condition: 'job\_stress\_level > 7 OR job\_security\_feeling < 5',

additional\_questions: [

{

id: 'stress\_source',

question: 'What\'s your biggest work concern right now?',

type: 'multiple\_choice',

options: ['Job security', 'Workload', 'Management', 'Career growth', 'Work-life balance', 'Other']

}

]

}

};

**Basic Job Security Algorithm Implementation**

class JobSecurityAnalyzer:

def \_\_init\_\_(self):

self.weight\_user\_perception = 0.6 # User's own assessment

self.weight\_external\_data = 0.4 # External indicators

def calculate\_job\_security\_score(self, user\_data, external\_data=None):

"""

Calculate comprehensive job security score

"""

# User perception component (60% weight)

user\_score = self.calculate\_user\_perception\_score(user\_data)

# External data component (40% weight)

if external\_data:

external\_score = self.calculate\_external\_risk\_score(external\_data)

final\_score = (user\_score \* self.weight\_user\_perception +

external\_score \* self.weight\_external\_data)

else:

# If no external data, use only user perception

final\_score = user\_score

return {

'overall\_score': max(0, min(100, int(final\_score))),

'user\_perception\_score': user\_score,

'external\_data\_score': external\_score if external\_data else None,

'confidence\_level': self.calculate\_confidence\_level(user\_data, external\_data),

'risk\_factors': self.identify\_risk\_factors(user\_data, external\_data),

'recommendations': self.generate\_recommendations(final\_score, user\_data)

}

def calculate\_user\_perception\_score(self, user\_data):

"""Calculate score based on user's own assessment"""

base\_score = 100

# Job stress impact (higher stress = lower security feeling usually)

stress\_level = user\_data.get('job\_stress\_level', 5)

stress\_penalty = (stress\_level - 1) \* 3 # 0-27 point penalty

# Security feeling (direct correlation)

security\_feeling = user\_data.get('job\_security\_feeling', 5)

security\_bonus = (security\_feeling - 1) \* 7 # 0-63 point bonus

# Warning signs penalty

warnings = user\_data.get('employer\_warnings', 'No, nothing')

warning\_penalty = {

'Yes, definitely': 30,

'Some rumors/hints': 15,

'No, nothing': 0,

'Not sure': 10

}.get(warnings, 0)

user\_score = base\_score - stress\_penalty + security\_bonus - warning\_penalty

return max(0, min(100, user\_score))

def calculate\_external\_risk\_score(self, external\_data):

"""Calculate score based on external risk factors"""

base\_score = 100

# Employer financial health (if available)

employer\_health = external\_data.get('employer\_financial\_health', 50)

health\_adjustment = (employer\_health - 50) \* 0.6 # ±30 point impact

# Local layoff activity

local\_layoffs = external\_data.get('recent\_local\_layoffs', 0)

layoff\_penalty = min(local\_layoffs \* 2, 25) # Up to 25 point penalty

# Industry trends

industry\_risk = external\_data.get('industry\_risk\_score', 50)

industry\_adjustment = (industry\_risk - 50) \* 0.4 # ±20 point impact

external\_score = base\_score + health\_adjustment - layoff\_penalty - industry\_adjustment

return max(0, min(100, external\_score))

def generate\_recommendations(self, score, user\_data):

"""Generate personalized recommendations based on score"""

recommendations = []

if score < 40: # High risk

recommendations.extend([

"💰 Build 6-month emergency fund immediately",

"📄 Update resume and LinkedIn profile",

"🔍 Begin discreet job search",

"📚 Develop in-demand skills in your field",

"🤝 Network with industry professionals"

])

elif score < 60: # Moderate risk

recommendations.extend([

"💡 Increase emergency fund to 4-6 months expenses",

"📈 Monitor industry trends and company news",

"🎯 Consider additional income streams",

"📱 Keep professional network active"

])

elif score < 80: # Low-moderate risk

recommendations.extend([

"✅ Maintain 3-4 month emergency fund",

"📊 Stay informed about company performance",

"🚀 Focus on skill development and career growth"

])

else: # Low risk

recommendations.extend([

"🎯 Consider long-term career advancement opportunities",

"💰 Focus on retirement and investment goals",

"🌟 Mentor others and build leadership skills"

])

return recommendations

**Phase 2: External Data Integration (Week 11-12)**

**SEC EDGAR Integration for Employer Analysis**

class EmployerFinancialAnalyzer:

def \_\_init\_\_(self):

self.sec\_base\_url = "https://data.sec.gov/api/xbrl/"

self.headers = {'User-Agent': 'Mingus Financial App (contact@mingus.app)'}

def analyze\_employer\_financial\_health(self, employer\_name):

"""

Analyze employer's financial health using SEC filings

"""

try:

# Step 1: Find company CIK

cik = self.find\_company\_cik(employer\_name)

if not cik:

return self.handle\_private\_company(employer\_name)

# Step 2: Get latest financial data

financial\_data = self.get\_company\_financial\_facts(cik)

# Step 3: Calculate health score

health\_score = self.calculate\_financial\_health\_score(financial\_data)

return {

'status': 'success',

'company\_name': financial\_data.get('company\_name'),

'cik': cik,

'financial\_health\_score': health\_score['overall\_score'],

'risk\_indicators': health\_score['risk\_indicators'],

'last\_filing\_date': financial\_data.get('last\_filing\_date'),

'analysis\_date': datetime.now().isoformat()

}

except Exception as e:

return {

'status': 'error',

'error\_message': str(e),

'fallback\_score': 50 # Neutral score when analysis fails

}

def find\_company\_cik(self, employer\_name):

"""Find company's CIK using SEC ticker/company lookup"""

# First try exact name match

search\_url = f"https://www.sec.gov/include/ticker.txt"

try:

response = requests.get(search\_url, headers=self.headers)

ticker\_data = response.text

# Parse ticker data to find matches

lines = ticker\_data.strip().split('\n')

for line in lines:

if employer\_name.lower() in line.lower():

parts = line.split('\t')

if len(parts) >= 2:

return parts[1].zfill(10) # Pad CIK to 10 digits

except Exception as e:

print(f"Error searching for CIK: {e}")

return None

def get\_company\_financial\_facts(self, cik):

"""Retrieve company financial facts from SEC EDGAR"""

url = f"{self.sec\_base\_url}companyfacts/{cik}.json"

try:

response = requests.get(url, headers=self.headers)

response.raise\_for\_status()

data = response.json()

# Extract key information

return {

'company\_name': data.get('entityName'),

'cik': cik,

'facts': data.get('facts', {}),

'last\_filing\_date': self.extract\_latest\_filing\_date(data)

}

except requests.RequestException as e:

raise Exception(f"Failed to retrieve SEC data: {e}")

def calculate\_financial\_health\_score(self, financial\_data):

"""Calculate financial health score from SEC data"""

facts = financial\_data.get('facts', {})

# Initialize component scores

revenue\_score = 50 # Default neutral

profitability\_score = 50

debt\_score = 50

cash\_score = 50

try:

# Revenue analysis

revenue\_trend = self.analyze\_revenue\_trend(facts)

revenue\_score = self.score\_revenue\_health(revenue\_trend)

# Profitability analysis

profit\_data = self.analyze\_profitability(facts)

profitability\_score = self.score\_profitability(profit\_data)

# Debt analysis

debt\_metrics = self.analyze\_debt\_position(facts)

debt\_score = self.score\_debt\_health(debt\_metrics)

# Cash position analysis

cash\_data = self.analyze\_cash\_position(facts)

cash\_score = self.score\_cash\_health(cash\_data)

except Exception as e:

print(f"Error in financial analysis: {e}")

# Calculate weighted overall score

weights = {

'revenue': 0.3,

'profitability': 0.25,

'debt': 0.25,

'cash': 0.2

}

overall\_score = (

revenue\_score \* weights['revenue'] +

profitability\_score \* weights['profitability'] +

debt\_score \* weights['debt'] +

cash\_score \* weights['cash']

)

# Identify risk indicators

risk\_indicators = []

if revenue\_score < 40:

risk\_indicators.append("Declining revenue trend")

if profitability\_score < 40:

risk\_indicators.append("Profitability concerns")

if debt\_score < 40:

risk\_indicators.append("High debt levels")

if cash\_score < 40:

risk\_indicators.append("Low cash position")

return {

'overall\_score': max(0, min(100, int(overall\_score))),

'component\_scores': {

'revenue\_health': revenue\_score,

'profitability': profitability\_score,

'debt\_health': debt\_score,

'cash\_position': cash\_score

},

'risk\_indicators': risk\_indicators,

'confidence\_level': self.calculate\_analysis\_confidence(facts)

}

def analyze\_revenue\_trend(self, facts):

"""Analyze revenue trend over recent periods"""

try:

# Look for revenue in different possible fields

revenue\_fields = [

'us-gaap:Revenues',

'us-gaap:RevenueFromContractWithCustomerExcludingAssessedTax',

'us-gaap:SalesRevenueNet'

]

revenue\_data = None

for field in revenue\_fields:

if field in facts.get('us-gaap', {}):

revenue\_data = facts['us-gaap'][field]['units']['USD']

break

if not revenue\_data:

return {'trend': 'no\_data', 'recent\_revenues': []}

# Get recent annual revenues

annual\_revenues = []

for entry in revenue\_data:

if entry.get('frame') and 'CY' in entry['frame']: # Annual data

annual\_revenues.append({

'year': entry['frame'],

'value': entry['val'],

'date': entry['end']

})

# Sort by year and calculate trend

annual\_revenues.sort(key=lambda x: x['date'])

recent\_revenues = annual\_revenues[-3:] # Last 3 years

if len(recent\_revenues) >= 2:

# Calculate growth rate

latest = recent\_revenues[-1]['value']

previous = recent\_revenues[-2]['value']

growth\_rate = ((latest - previous) / previous) \* 100

if growth\_rate > 5:

trend = 'growing'

elif growth\_rate > -5:

trend = 'stable'

else:

trend = 'declining'

else:

trend = 'insufficient\_data'

return {

'trend': trend,

'growth\_rate': growth\_rate if 'growth\_rate' in locals() else 0,

'recent\_revenues': recent\_revenues

}

except Exception as e:

return {'trend': 'analysis\_error', 'recent\_revenues