**Features Document: Dual-Country Financial Planning Platform**

**EXECUTIVE SUMMARY: COMPLETE FEATURES DOCUMENT**

This comprehensive Product Requirements Document and Features Specification covers a complete dual-country (UK and South Africa) financial planning web application with the following major components:

**1. CORE INFRASTRUCTURE**

* **User Authentication & Profile Management**: 2FA, session management, secure login
* **User Information Module**: Tax status, domicile tracking, income management, family details
* **Central Dashboard**: Net worth aggregation, real-time updates, goal tracking

**2. FINANCIAL MODULES**

**Protection Module**

* Life assurance, critical illness, income protection tracking
* Coverage gap analysis
* Tax treatment (UK trusts, SA estate duty)
* Beneficiary management
* Premium tracking and reminders

**Savings Module**

* Multi-currency cash account tracking
* ISA and TFSA allowance management
* Emergency fund assessment
* Interest income tax calculation
* Bank account aggregation capability

**Investment Module**

* Portfolio management (stocks, funds, ETFs)
* ISA, GIA, VCT, EIS, SEIS tracking
* SA unit trusts and equity investments
* Asset allocation analysis
* Tax lot tracking for CGT
* Dividend income management

**Retirement Module**

* **UK**: DB/DC pensions, SIPP, personal pensions, state pension
* **SA**: Pension funds, provident funds, retirement annuities, preservation funds
* **Cross-border**: QROPS/ROPS tracking
* Annual allowance monitoring (UK)
* Section 10C deduction tracking (SA)
* Retirement income projection

**Inheritance Tax Planning Module**

* Comprehensive asset register (all jurisdictions)
* Liabilities register with deductibility rules
* UK IHT calculation (NRB, RNRB, reliefs)
* SA Estate Duty calculation
* Lifetime gifts register (PETs tracking, 7-year rule)
* Estate value projection
* DTA relief application

**3. TAX INTELLIGENCE ENGINE**

**Core Tax Calculations**

* UK Income Tax (including Scottish rates)
* UK National Insurance (Class 1, 2, 4)
* UK Capital Gains Tax
* UK Dividend Tax
* SA Income Tax with rebates
* SA Capital Gains Tax (inclusion rate)
* SA Dividend Withholding Tax

**Double Tax Agreement (DTA) Relief**

* Automated DTA provision application
* Foreign tax credit calculation
* Source vs residence taxation determination
* Tie-breaker rules for dual residents
* Income categorization by DTA article

**Tax Residency Determination**

* UK Statutory Residence Test (SRT) automation
* SA Physical Presence Test
* UK domicile and deemed domicile tracking
* Split year treatment identification
* Historical residency tracking

**4. AI ADVISORY ENGINE**

**Recommendation Generation**

* Protection recommendations (coverage gaps, trust structures)
* Savings recommendations (emergency fund, ISA/TFSA usage)
* Investment recommendations (diversification, tax efficiency, CGT harvesting)
* Retirement recommendations (contribution optimization, carry forward)
* Tax optimization recommendations (allowance usage, marriage allowance)
* IHT planning recommendations (gifting strategies, estate equalization)
* Prioritization by urgency and impact
* Estimated benefit calculations

**Goal-Based Financial Planning**

* SMART goal creation and validation
* Required contribution calculations
* Savings vehicle recommendations
* Feasibility assessment
* Progress tracking with milestones
* Monte Carlo simulation for success probability
* Goal conflict resolution
* Alternative scenario generation

**Scenario Analysis & What-If Modeling**

* Major life event modeling (retirement, relocation, business sale)
* Side-by-side scenario comparison
* Impact analysis across all modules
* Sensitivity analysis
* Risk assessment
* Probability-weighted outcomes
* Trade-off identification

**Personalization Engine**

* Behavioral tracking and learning
* User profile building (demographics, behavior, preferences)
* Recommendation personalization
* Collaborative filtering
* A/B testing framework
* Feedback loop processing
* Model retraining pipeline
* Explainable AI

**5. KEY TECHNICAL FEATURES**

**Architecture**

* Modular design (each module independent)
* API-first approach
* Multi-currency support (GBP, ZAR, others)
* Historical data tracking (temporal queries)
* Real-time calculations
* Async job processing
* Caching strategies

**Data Management**

* Versioned tax configurations
* Audit trails for all calculations
* Document storage and OCR
* Integration with external modules
* Bank aggregation (Open Banking)
* Currency conversion services

**Security & Compliance**

* Bank-level encryption
* GDPR and POPIA compliance
* Role-based access control
* Two-factor authentication
* Data sovereignty considerations
* Audit logging

**Performance**

* Target response times: <1s dashboard, <200ms tax calc, <3s recommendations
* Caching strategies (Redis, materialized views)
* Async processing for heavy computations
* Batch jobs for periodic updates
* Scalable infrastructure

**6. USER EXPERIENCE**

**Key User Flows**

* Progressive onboarding with guided setup
* Intuitive data entry with validation
* Real-time feedback and calculations
* Visual dashboards and charts
* Mobile-responsive design
* Contextual help and education

**Notifications & Alerts**

* Tax year-end reminders
* Allowance usage warnings
* Goal milestone achievements
* Recommendation updates
* Document expiry alerts
* Multi-channel delivery (email, in-app, SMS)

**7. REPORTING & EXPORT**

* Net worth statements
* Tax preparation reports (self-assessment, provisional tax)
* Portfolio performance reports
* Goal progress reports
* Estate planning summaries
* Scenario comparison reports
* PDF export functionality
* CSV/Excel data export

**8. INTEGRATION CAPABILITIES**

* Bank account aggregation
* Investment platform APIs
* Pension scheme data imports
* HMRC/SARS integration (future)
* Financial advisor collaboration
* Third-party financial tools

**9. ROADMAP CONSIDERATIONS**

**Phase 1: Foundation**

Core authentication and user management

* User Information Module
* Central Dashboard (basic)
* One module (Savings - simplest) fully built

**Phase 2: Core Modules**

* Protection Module
* Investment Module
* Tax Intelligence Engine (basic)
* Basic recommendations

**Phase 3: Advanced Features**

* Retirement Module (all products)
* IHT Planning Module
* DTA Relief Calculator
* Enhanced Tax Intelligence

**Phase 4: Intelligence**

* Goal-Based Planning
* Scenario Analysis
* Advanced Recommendations
* Personalization Engine

**Phase 5: Enhancement**

* Machine learning optimization
* Additional jurisdictions
* Advanced integrations
* Mobile app development

**10. SUCCESS METRICS**

**User Engagement**

* Daily/Monthly Active Users
* Average session duration
* Feature adoption rates
* Recommendation acceptance rates

**Value Delivery**

* Tax savings identified
* Goals achieved on time
* IHT liability reduction
* Investment performance vs benchmarks

**Quality**

* Calculation accuracy (>99.9%)
* User satisfaction (NPS >50)
* Support ticket volume (trend down)
* System uptime (>99.5%)

**11. RISK MITIGATION**

**Technical Risks**

* Complex tax calculations: Professional review, extensive testing
* Data security: Best practices, regular audits, insurance
* Performance at scale: Scalable architecture, load testing
* Integration failures: Robust error handling, fallback mechanisms

**Regulatory Risks**

* Crossing into regulated advice: Clear disclaimers, legal review
* Tax law changes: Regular updates, versioned configurations
* Data protection: GDPR/POPIA compliance by design

**Operational Risks**

* Tax calculation errors: Multi-layer validation, professional oversight
* User adoption: Excellent UX, comprehensive onboarding
* Maintaining currency: Partnerships with tax professionals, regular updates

**Table of Contents**

1. [User Authentication & Profile Management](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#1-user-authentication--profile-management)
2. [User Information Module](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#2-user-information-module)
3. [Central Dashboard](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#3-central-dashboard)
4. [Protection Module](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#4-protection-module)
5. [Savings Module](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#5-savings-module)
6. [Investment Module](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#6-investment-module)
7. [Retirement Module](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#7-retirement-module)
8. [Inheritance Tax Planning Module](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#8-inheritance-tax-planning-module)
9. [Tax Intelligence Engine](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#9-tax-intelligence-engine)
10. [AI Advisory Engine](https://claude.ai/chat/0358691f-6ed6-4fbc-9efb-ee9f6a8e6412#10-ai-advisory-engine)

**1. USER AUTHENTICATION & PROFILE MANAGEMENT**

**Feature 1.1: User Registration**

**Feature Name:** User Registration and Onboarding

**User Story:** As a new user, I want to create an account with email and password so that I can securely access the financial planning platform.

**Acceptance Criteria:**

* User can register with email and password
* Email verification required before full access
* Password must meet security requirements (min 12 chars, uppercase, lowercase, number, special char)
* User receives welcome email with onboarding guide
* Account created with default settings
* GDPR/POPIA consent captured
* User can select preferred country (UK/SA/Both)

**Technical Requirements:**

* Encryption: bcrypt or Argon2 for password hashing
* JWT tokens for session management
* Email service integration (SendGrid, AWS SES, or similar)
* Rate limiting on registration endpoint (max 5 attempts per IP per hour)
* CAPTCHA integration to prevent bot registrations

**Constraints:**

* Must comply with UK GDPR and SA POPIA
* No Node.js backend
* Session timeout: 30 minutes inactivity, 24 hours absolute

**Implementation Approach:**

ENDPOINT: POST /api/v1/auth/register

REQUEST BODY:

{

email: string (valid email format),

password: string (meets complexity requirements),

firstName: string,

lastName: string,

country: enum['UK', 'SA', 'BOTH'],

termsAccepted: boolean,

marketingConsent: boolean

}

PROCESS:

1. Validate input data

2. Check if email already exists

3. Hash password using Argon2

4. Generate email verification token (UUID, expires in 24h)

5. Create user record with status='PENDING\_VERIFICATION'

6. Store verification token in cache/DB

7. Send verification email

8. Return success response (do not reveal if email exists)

RESPONSE:

{

success: true,

message: "Registration successful. Please check your email.",

userId: uuid

}

**User Flow:**

[Landing Page] → [Register Button]

↓

[Registration Form]

- Email input

- Password input (with strength indicator)

- Confirm password

- First name / Last name

- Country selection

- Terms checkbox

- Marketing consent checkbox

↓

[Submit] → [Validation]

↓

[Success Message] → [Email Sent Notification]

↓

[User checks email] → [Clicks verification link]

↓

[Account Activated] → [Redirect to Login]

**API Endpoints:**

* POST /api/v1/auth/register - Register new user
* GET /api/v1/auth/verify-email?token={token} - Verify email
* POST /api/v1/auth/resend-verification - Resend verification email

**Data Models:**

TABLE: users

- id: UUID (PK)

- email: VARCHAR(255) UNIQUE NOT NULL

- password\_hash: VARCHAR(255) NOT NULL

- first\_name: VARCHAR(100)

- last\_name: VARCHAR(100)

- country\_preference: ENUM('UK', 'SA', 'BOTH')

- status: ENUM('PENDING\_VERIFICATION', 'ACTIVE', 'SUSPENDED', 'DELETED')

- email\_verified: BOOLEAN DEFAULT FALSE

- terms\_accepted\_at: TIMESTAMP

- marketing\_consent: BOOLEAN

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

- last\_login\_at: TIMESTAMP

TABLE: email\_verification\_tokens

- id: UUID (PK)

- user\_id: UUID (FK to users)

- token: VARCHAR(255) UNIQUE

- expires\_at: TIMESTAMP

- used: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

INDEX on users.email

INDEX on email\_verification\_tokens.token

INDEX on email\_verification\_tokens.expires\_at

**Error Handling:**

ERROR CASES:

1. Email already registered

- Response: 409 Conflict

- Message: "An account with this email already exists"

2. Invalid email format

- Response: 400 Bad Request

- Message: "Please provide a valid email address"

3. Password too weak

- Response: 400 Bad Request

- Message: "Password must be at least 12 characters and include uppercase, lowercase, number, and special character"

4. Terms not accepted

- Response: 400 Bad Request

- Message: "You must accept the terms and conditions"

5. Rate limit exceeded

- Response: 429 Too Many Requests

- Message: "Too many registration attempts. Please try again later"

6. Email service failure

- Response: 500 Internal Server Error

- Log error, queue for retry

- Message: "Registration successful but verification email delayed"

EDGE CASES:

- User tries to register with existing email: Return generic success message (security)

- Token expired: Allow resend with new token

- User clicks verify link multiple times: Idempotent operation

- Simultaneous registrations with same email: DB unique constraint handles it

**Performance Considerations:**

* Password hashing is CPU intensive: Use async hashing, consider queuing
* Email sending should be async (message queue)
* Cache email existence checks (5 min TTL) to reduce DB queries
* Verification tokens in Redis for fast lookup
* Rate limiting via Redis counter
* Expected load: 1000 registrations/day, <500ms response time

**Feature 1.2: User Login with 2FA**

**Feature Name:** Secure Login with Two-Factor Authentication

**User Story:** As a registered user, I want to log in securely with my email and password, and enable two-factor authentication for enhanced security.

**Acceptance Criteria:**

* User can login with email and password
* Optional 2FA via authenticator app (TOTP)
* Remember device option (30 days)
* Account lockout after 5 failed attempts (30 min cooldown)
* Notification sent on new device login
* Session management with JWT tokens
* Refresh token mechanism for extended sessions

**Technical Requirements:**

* TOTP library for 2FA (pyotp, Google Authenticator compatible)
* JWT with RS256 signing
* Redis for session management and rate limiting
* Device fingerprinting for "remember device"
* IP logging for security monitoring

**Constraints:**

* Access token lifetime: 15 minutes
* Refresh token lifetime: 7 days
* 2FA code valid for 30 seconds (1 time-step)
* Maximum 5 concurrent sessions per user

**Implementation Approach:**

ENDPOINT: POST /api/v1/auth/login

REQUEST BODY:

{

email: string,

password: string,

deviceId: string (optional, for remember device),

totpCode: string (optional, if 2FA enabled)

}

PROCESS:

1. Rate limit check (max 5 attempts per IP per 15 min)

2. Fetch user by email

3. Check account status (must be ACTIVE)

4. Verify password hash

5. If 2FA enabled and not remembered device:

a. If totpCode provided, verify it

b. If totpCode not provided, return 2FA\_REQUIRED

6. Check device trust status

7. Generate access token (JWT, 15 min expiry)

8. Generate refresh token (JWT, 7 days expiry)

9. Store session in Redis

10. Update last\_login\_at

11. Log login event

12. Send notification if new device

13. Return tokens

RESPONSE SUCCESS:

{

accessToken: string,

refreshToken: string,

expiresIn: 900,

user: {

id: uuid,

email: string,

firstName: string,

lastName: string,

twoFactorEnabled: boolean

}

}

RESPONSE 2FA REQUIRED:

{

requiresTwoFactor: true,

message: "Please provide your 2FA code"

}

**User Flow:**

[Login Page]

↓

[Enter Email & Password] → [Submit]

↓

[Backend Validation]

↓

[2FA Enabled?] → YES → [2FA Code Entry Page]

| ↓

| [Enter TOTP Code]

| ↓

| [Verify Code]

| ↓

NO ← ← ← ← ← ← ← ← ← ← [Success]

↓

[Generate Tokens]

↓

[Redirect to Dashboard]

**API Endpoints:**

* POST /api/v1/auth/login - User login
* POST /api/v1/auth/refresh - Refresh access token
* POST /api/v1/auth/logout - Invalidate session
* POST /api/v1/auth/logout-all - Invalidate all sessions
* GET /api/v1/auth/sessions - List active sessions
* POST /api/v1/auth/enable-2fa - Enable 2FA
* POST /api/v1/auth/disable-2fa - Disable 2FA
* POST /api/v1/auth/verify-2fa-setup - Verify 2FA during setup

**Data Models:**

TABLE: user\_sessions

- id: UUID (PK)

- user\_id: UUID (FK to users)

- access\_token\_jti: VARCHAR(255) UNIQUE (JWT ID)

- refresh\_token\_jti: VARCHAR(255) UNIQUE

- device\_id: VARCHAR(255)

- device\_name: VARCHAR(255)

- ip\_address: VARCHAR(45)

- user\_agent: TEXT

- is\_trusted: BOOLEAN DEFAULT FALSE

- last\_activity: TIMESTAMP

- created\_at: TIMESTAMP

- expires\_at: TIMESTAMP

TABLE: user\_2fa

- user\_id: UUID (PK, FK to users)

- secret: VARCHAR(255) ENCRYPTED

- enabled: BOOLEAN DEFAULT FALSE

- backup\_codes: JSON ENCRYPTED (array of hashed codes)

- created\_at: TIMESTAMP

- last\_used\_at: TIMESTAMP

TABLE: login\_attempts

- id: UUID (PK)

- email: VARCHAR(255)

- ip\_address: VARCHAR(45)

- success: BOOLEAN

- failure\_reason: VARCHAR(100)

- timestamp: TIMESTAMP

INDEX on user\_sessions.user\_id

INDEX on user\_sessions.access\_token\_jti

INDEX on user\_sessions.expires\_at

INDEX on login\_attempts.ip\_address, timestamp

**Error Handling:**

ERROR CASES:

1. Invalid credentials

- Response: 401 Unauthorized

- Message: "Invalid email or password"

- Log attempt in login\_attempts table

2. Account locked (5 failed attempts)

- Response: 423 Locked

- Message: "Account temporarily locked due to multiple failed login attempts. Try again in 30 minutes"

3. Account not verified

- Response: 403 Forbidden

- Message: "Please verify your email address before logging in"

4. 2FA code invalid

- Response: 401 Unauthorized

- Message: "Invalid or expired 2FA code"

5. Rate limit exceeded

- Response: 429 Too Many Requests

- Message: "Too many login attempts. Please try again later"

6. Session limit exceeded

- Response: 409 Conflict

- Message: "Maximum number of active sessions reached. Please logout from another device"

- Provide option to force logout oldest session

EDGE CASES:

- Concurrent logins: Allow up to 5 sessions

- Token refresh while access token still valid: Allow, extend session

- Logout with expired refresh token: Return success (idempotent)

- Time synchronization issues with TOTP: Accept codes from previous/next time window (90 sec total window)

- User changes password: Invalidate all existing sessions except current

**Performance Considerations:**

* Redis session store for fast token validation
* JWT signature verification is CPU intensive: Cache public key
* Rate limiting counters in Redis (sliding window algorithm)
* Async notification sending (queue)
* Expected load: 10,000 logins/day, <200ms response time
* Session validation on every API call must be <10ms
* Consider token payload size (keep under 1KB)

**2. USER INFORMATION MODULE**

**Feature 2.1: Tax Status & Domicile Management**

**Feature Name:** Comprehensive Tax Status and Domicile Tracking

**User Story:** As a user with ties to both UK and SA, I want to input and track my tax residency status, domicile information, and years of residency so that the system can provide accurate tax advice.

**Acceptance Criteria:**

* User can input current tax residency (UK, SA, both, neither)
* User can track domicile status (UK, SA, or non-domiciled)
* System tracks years of residency in each country
* UK Statutory Residence Test (SRT) calculator available
* SA physical presence test calculator available
* Historical changes tracked with effective dates
* Deemed domicile status automatically calculated (UK)
* Remittance basis vs arising basis selection
* Future domicile projections based on planned residence

**Technical Requirements:**

* Complex business logic for SRT calculation
* Date-based calculations for residency tracking
* Historical state management (temporal data)
* Rules engine for deemed domicile determination
* Integration with Income Tax calculation engine

**Constraints:**

* Must support backdating for historical accuracy
* Cannot delete historical records (audit trail)
* SRT rules change periodically (versioned rule sets)
* SA physical presence test: 91 days in current year + 91 days in previous 5 years average

**Implementation Approach:**

ENDPOINT: POST /api/v1/user/tax-status

REQUEST BODY:

{

effectiveDate: date,

ukTaxResident: boolean,

saTaxResident: boolean,

domicile: enum['UK\_DOMICILED', 'SA\_DOMICILED', 'NON\_UK\_DOMICILED', 'NON\_SA\_DOMICILED'],

domicileOfOrigin: enum['UK', 'SA', 'OTHER'],

ukResidenceBasis: enum['ARISING', 'REMITTANCE'] (if non-UK domiciled),

yearsInUk: integer,

yearsInSa: integer,

notes: text

}

BUSINESS LOGIC:

1. Calculate deemed domicile status:

UK\_DEEMED\_DOMICILE = (yearsInUk >= 15 out of last 20 years) OR

(UK domicile of origin AND yearsInUk >= 1 out of last 2 years)

2. Determine applicable tax regimes:

IF ukTaxResident AND domicile = 'UK\_DOMICILED' THEN

worldwide\_income\_taxed\_in\_uk = TRUE

ELSE IF ukTaxResident AND domicile = 'NON\_UK\_DOMICILED' AND basis = 'REMITTANCE' THEN

only\_uk\_source\_and\_remitted\_income\_taxed = TRUE

3. Validate residency status:

IF both UK and SA tax resident THEN

apply\_DTA\_tie\_breaker\_rules()

4. Store with temporal validity

5. Recalculate tax implications across all modules

6. Trigger AI advisory engine to review recommendations

RESPONSE:

{

id: uuid,

effectiveDate: date,

calculatedStatus: {

ukDeemedDomicile: boolean,

ukTaxLiability: enum['WORLDWIDE', 'UK\_SOURCE\_ONLY', 'REMITTANCE\_ONLY'],

saTaxLiability: enum['WORLDWIDE', 'SA\_SOURCE\_ONLY'],

dualResident: boolean,

dtaTieBreakerResult: enum['UK\_RESIDENT', 'SA\_RESIDENT', 'N/A']

}

}

**User Flow:**

[User Information Dashboard] → [Tax Status Section]

↓

[Edit Tax Status Button]

↓

[Tax Status Form]

- Effective date selector

- "Are you UK tax resident?" toggle

- "Are you SA tax resident?" toggle

- Domicile status dropdown

- Domicile of origin dropdown

- Years in UK input (auto-calculated if dates provided)

- Years in SA input (auto-calculated if dates provided)

- If non-UK domiciled: Remittance basis selection

↓

[Calculate Button] → [Show calculated status preview]

- Deemed domicile status

- Tax liability scope

- DTA implications

↓

[Save] → [Confirmation]

↓

[Trigger recalculation across modules]

↓

[Show updated recommendations]

**API Endpoints:**

* POST /api/v1/user/tax-status - Create/update tax status
* GET /api/v1/user/tax-status - Get current tax status
* GET /api/v1/user/tax-status/history - Get historical tax status
* GET /api/v1/user/tax-status/at-date?date={date} - Get status at specific date
* POST /api/v1/user/tax-status/srt-calculator - UK SRT calculator
* POST /api/v1/user/tax-status/sa-presence-test - SA physical presence test

**Data Models:**

TABLE: user\_tax\_status

- id: UUID (PK)

- user\_id: UUID (FK to users)

- effective\_from: DATE NOT NULL

- effective\_to: DATE (NULL = current)

- uk\_tax\_resident: BOOLEAN

- sa\_tax\_resident: BOOLEAN

- domicile: ENUM('UK\_DOMICILED', 'SA\_DOMICILED', 'NON\_UK\_DOMICILED', 'OTHER')

- domicile\_of\_origin: ENUM('UK', 'SA', 'OTHER')

- uk\_residence\_basis: ENUM('ARISING', 'REMITTANCE') NULL

- years\_in\_uk: INTEGER

- years\_in\_sa: INTEGER

- uk\_deemed\_domicile: BOOLEAN (calculated)

- dual\_resident: BOOLEAN (calculated)

- dta\_tie\_breaker\_result: ENUM('UK\_RESIDENT', 'SA\_RESIDENT', 'N/A')

- notes: TEXT

- created\_at: TIMESTAMP

- created\_by: UUID (FK to users)

TABLE: uk\_srt\_data

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: VARCHAR(7) (e.g., '2024/25')

- days\_in\_uk: INTEGER

- tie\_1\_family: BOOLEAN

- tie\_2\_accommodation: BOOLEAN

- tie\_3\_work: BOOLEAN

- tie\_4\_90\_days\_previous\_years: BOOLEAN

- tie\_5\_more\_days\_uk\_than\_other: BOOLEAN

- sufficient\_ties\_count: INTEGER

- automatic\_overseas\_test: BOOLEAN

- automatic\_uk\_test: BOOLEAN

- sufficient\_ties\_test\_result: BOOLEAN

- final\_result: ENUM('UK\_RESIDENT', 'NON\_RESIDENT')

- created\_at: TIMESTAMP

TABLE: sa\_presence\_data

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: VARCHAR(9) (e.g., '2024/2025')

- days\_in\_sa: INTEGER

- days\_in\_sa\_previous\_5\_years: JSON (array of {year, days})

- average\_days\_previous\_5\_years: DECIMAL

- physically\_present\_test\_result: BOOLEAN

- ordinarily\_resident: BOOLEAN

- created\_at: TIMESTAMP

INDEX on user\_tax\_status(user\_id, effective\_from, effective\_to)

INDEX on uk\_srt\_data(user\_id, tax\_year)

INDEX on sa\_presence\_data(user\_id, tax\_year)

**Error Handling:**

ERROR CASES:

1. Effective date in future beyond reasonable period

- Response: 400 Bad Request

- Message: "Effective date cannot be more than 1 year in the future"

2. Overlapping effective periods

- Response: 409 Conflict

- Message: "A tax status record already exists for this period"

- Auto-adjust previous record's effective\_to date

3. Invalid domicile + residence combination

- Response: 400 Bad Request

- Message: "Cannot be UK domiciled and non-UK tax resident for more than 5 years"

4. Remittance basis selected but UK domiciled

- Response: 400 Bad Request

- Message: "Remittance basis only available for non-UK domiciled individuals"

5. Years in UK exceeds age

- Response: 400 Bad Request

- Message: "Years of UK residency cannot exceed your age"

EDGE CASES:

- User lived in UK before birth (parents): Allow with warning

- Split year treatment: Create two records for same tax year

- Dual resident with no DTA: Manual tie-breaker input required

- Non-dom claiming remittance basis: Track remittance basis claim fee payment

- Temporary non-residence rules: Flag if left UK and returning within 5 years

- Crown employees: Override SRT rules (special status flag)

**Performance Considerations:**

* SRT calculation is complex: Cache results for tax year
* Historical queries use temporal query optimization
* Deemed domicile calculation runs on save: <100ms
* Recalculation triggers: Use async job queue for module updates
* Expected usage: Multiple updates per user per year
* Index on effective\_from and effective\_to for fast temporal queries

**Feature 2.2: Income Information Management**

**Feature Name:** Multi-Jurisdiction Income Tracking

**User Story:** As a user with income from multiple sources and countries, I want to record all my income streams with their source country and type so that the system can calculate my tax liability accurately in both jurisdictions.

**Acceptance Criteria:**

* User can add multiple income sources
* Each income has: type, amount, currency, country of source, frequency
* Support for employment, self-employment, rental, dividend, interest, pension income
* Foreign income flagged and tracked
* PAYE/PASE details captured (tax already paid at source)
* Tax year allocation for income received
* Gross and net income recording
* Tax withheld at source tracked

**Technical Requirements:**

* Multi-currency support with historical exchange rates
* Income categorization aligned with UK and SA tax categories
* Integration with Tax Intelligence Engine
* Support for grossing up net income
* Temporal tracking (income changes over time)

**Constraints:**

* Must handle both UK tax year (April-April) and SA tax year (March-February)
* Currency conversion uses HMRC/SARS official rates where applicable
* Historic income cannot be modified after tax return filed flag set

**Implementation Approach:**

ENDPOINT: POST /api/v1/user/income

REQUEST BODY:

{

incomeType: enum['EMPLOYMENT', 'SELF\_EMPLOYMENT', 'RENTAL', 'DIVIDEND',

'INTEREST', 'PENSION', 'CAPITAL\_GAINS', 'OTHER'],

sourceCountry: enum['UK', 'SA', 'OTHER'],

description: string,

amount: decimal,

currency: string (ISO 4217),

frequency: enum['ANNUAL', 'MONTHLY', 'QUARTERLY', 'ONE\_OFF'],

taxYear: string (e.g., 'UK-2024/25' or 'SA-2024/2025'),

grossIncome: boolean (true if gross, false if net),

taxWithheldAtSource: decimal (optional),

payeReference: string (optional, for employment),

startDate: date,

endDate: date (optional, NULL = ongoing),

relatedEntity: string (employer name, rental property, etc.)

}

BUSINESS LOGIC:

1. Validate income type and source country combination

2. Convert to GBP and ZAR using exchange rate for date

3. Determine tax treatment based on:

- Income type

- Source country

- User's tax residency

- DTA provisions

4. Calculate expected tax liability

5. If net income provided, gross up using applicable tax rate

6. Allocate to correct tax year(s) based on receipt date

7. Store with currency conversion metadata

8. Trigger recalculation of total income for tax purposes

PROCESS:

calculate\_tax\_treatment(income):

IF user.uk\_tax\_resident AND (income.sourceCountry = 'UK' OR user.domicile\_basis = 'ARISING'):

uk\_taxable = TRUE

apply\_uk\_tax\_rates(income.type)

IF user.sa\_tax\_resident AND (income.sourceCountry = 'SA' OR income.sourceCountry = 'OTHER'):

sa\_taxable = TRUE

apply\_sa\_tax\_rates(income.type)

IF uk\_taxable AND sa\_taxable:

apply\_DTA\_relief() // Avoid double taxation

RETURN tax\_summary

RESPONSE:

{

id: uuid,

incomeDetails: {...},

taxTreatment: {

ukTaxable: boolean,

ukTaxAmount: decimal,

saTaxable: boolean,

saTaxAmount: decimal,

dtaRelief: decimal,

effectiveTaxRate: decimal

},

conversionDetails: {

gbpAmount: decimal,

zarAmount: decimal,

exchangeRate: decimal,

rateDate: date

}

}

**User Flow:**

[User Information Dashboard] → [Income Section]

↓

[Add Income Button]

↓

[Income Entry Form - Step 1: Type]

- Select income type (visual cards)

↓

[Income Entry Form - Step 2: Details]

- Source country

- Description/name

- Amount and currency

- Frequency

- Is this gross or net? toggle

- Start date (end date if applicable)

↓

[Income Entry Form - Step 3: Tax Details]

- Tax withheld at source (if applicable)

- PAYE reference (if employment)

- Related entity details

↓

[Preview Calculation]

- Show UK tax treatment

- Show SA tax treatment

- Show DTA relief

- Total expected tax

↓

[Save] → [Confirmation]

↓

[Income List View - Updated]

- Grouped by type

- Total annual income display

- Tax liability summary

**API Endpoints:**

* POST /api/v1/user/income - Add new income source
* PUT /api/v1/user/income/{id} - Update income source
* DELETE /api/v1/user/income/{id} - Delete income source (soft delete)
* GET /api/v1/user/income - List all income sources
* GET /api/v1/user/income/summary - Income summary by type and country
* GET /api/v1/user/income/tax-year/{taxYear} - Income for specific tax year
* POST /api/v1/user/income/gross-up - Calculate gross from net

**Data Models:**

TABLE: user\_income

- id: UUID (PK)

- user\_id: UUID (FK to users)

- income\_type: ENUM('EMPLOYMENT', 'SELF\_EMPLOYMENT', 'RENTAL', 'DIVIDEND',

'INTEREST', 'PENSION', 'CAPITAL\_GAINS', 'OTHER')

- source\_country: ENUM('UK', 'SA', 'OTHER')

- description: VARCHAR(255)

- amount: DECIMAL(15,2)

- currency: CHAR(3)

- frequency: ENUM('ANNUAL', 'MONTHLY', 'QUARTERLY', 'ONE\_OFF')

- uk\_tax\_year: VARCHAR(7) (e.g., '2024/25')

- sa\_tax\_year: VARCHAR(9) (e.g., '2024/2025')

- is\_gross: BOOLEAN

- tax\_withheld\_at\_source: DECIMAL(15,2)

- paye\_reference: VARCHAR(50)

- start\_date: DATE

- end\_date: DATE (NULL = ongoing)

- related\_entity: VARCHAR(255)

- gbp\_amount: DECIMAL(15,2) (calculated)

- zar\_amount: DECIMAL(15,2) (calculated)

- exchange\_rate\_used: DECIMAL(10,6)

- exchange\_rate\_date: DATE

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: income\_tax\_treatment

- id: UUID (PK)

- income\_id: UUID (FK to user\_income)

- uk\_taxable: BOOLEAN

- uk\_tax\_estimated: DECIMAL(15,2)

- sa\_taxable: BOOLEAN

- sa\_tax\_estimated: DECIMAL(15,2)

- dta\_relief\_claimed: DECIMAL(15,2)

- effective\_tax\_rate: DECIMAL(5,2)

- calculation\_date: TIMESTAMP

- tax\_status\_id: UUID (FK to user\_tax\_status - for audit)

TABLE: exchange\_rates

- id: UUID (PK)

- from\_currency: CHAR(3)

- to\_currency: CHAR(3)

- rate: DECIMAL(10,6)

- rate\_date: DATE

- source: VARCHAR(50) (e.g., 'HMRC', 'SARS', 'ECB')

- created\_at: TIMESTAMP

UNIQUE INDEX on exchange\_rates(from\_currency, to\_currency, rate\_date)

INDEX on user\_income(user\_id, uk\_tax\_year)

INDEX on user\_income(user\_id, sa\_tax\_year)

INDEX on user\_income(user\_id, income\_type)

**Error Handling:**

ERROR CASES:

1. Invalid currency code

- Response: 400 Bad Request

- Message: "Invalid currency code. Please use ISO 4217 format (e.g., GBP, ZAR)"

2. Exchange rate not available for date

- Response: 404 Not Found

- Message: "Exchange rate not available for {date}. Using nearest available rate from {actualDate}"

- Warning flag set on income record

3. Negative income amount

- Response: 400 Bad Request

- Message: "Income amount must be positive. Use expenses to record negative amounts"

4. End date before start date

- Response: 400 Bad Request

- Message: "End date must be after start date"

5. Tax withheld exceeds income

- Response: 400 Bad Request

- Message: "Tax withheld cannot exceed income amount"

6. Attempt to modify locked tax year

- Response: 423 Locked

- Message: "Cannot modify income for tax year {year} as tax return has been filed"

EDGE CASES:

- Income received in foreign currency: Store original + converted amounts

- Income spanning multiple tax years (arrears): Allocate based on receipt date

- Bonus payments: Create separate one-off income entry

- Pension splitting for tax: Create separate entries for UK and SA portions

- Grossing up net dividends: Apply correct dividend tax credit rate

- Foreign dividend with withholding tax: Record both gross and net, claim DTA relief

- Self-employment: Separate entry for each source if needed

- Rental income: Net of expenses, link to property records in future

**Performance Considerations:**

* Exchange rate lookups: Cache daily rates (1 day TTL)
* Tax calculation: Cache results per income record
* Income summary queries: Materialized view for tax year totals
* Recalculation triggers: Debounce multiple income entries (batch process)
* Expected usage: 5-20 income sources per user
* Summary calculation: <500ms for all income sources
* Indexed queries on tax\_year fields for fast filtering

**3. CENTRAL DASHBOARD**

**Feature 3.1: Net Worth Summary Dashboard**

**Feature Name:** Comprehensive Net Worth Aggregation and Visualization

**User Story:** As a user, I want to see my complete net worth across all accounts, investments, properties, and liabilities in one place, with breakdowns by country, currency, and asset type.

**Acceptance Criteria:**

* Display total net worth (assets - liabilities)
* Breakdown by country (UK, SA, Offshore)
* Breakdown by currency (with conversion to base currency)
* Breakdown by asset class (Cash, Investments, Property, Pensions, Other)
* Liability summary
* Net worth trend over time (line chart)
* Asset allocation pie chart
* Currency exposure visualization
* Quick links to each module
* Refresh data button with last updated timestamp

**Technical Requirements:**

* Aggregate data from all modules
* Real-time currency conversion
* Historical net worth tracking (snapshots)
* Caching for performance
* Responsive charts library (Chart.js, D3, or Recharts)
* WebSocket for real-time updates (optional)

**Constraints:**

* Dashboard must load in <2 seconds
* Data refresh max every 5 minutes for external data
* Support for up to 1000 line items per user
* Mobile responsive design required

**Implementation Approach:**

ENDPOINT: GET /api/v1/dashboard/net-worth

QUERY PARAMS:

- baseCurrency: string (default: GBP)

- asOfDate: date (optional, default: today)

BUSINESS LOGIC:

1. Fetch all assets from modules:

- Savings accounts → SavingsModule

- Investments → InvestmentModule

- Pensions → RetirementModule

- Properties → IHTModule.assets

- Protection policies (cash value) → ProtectionModule

- Other assets → IHTModule.assets

2. Fetch all liabilities from modules:

- Mortgages → IHTModule.liabilities

- Loans → IHTModule.liabilities

- Credit cards → SavingsModule (negative balance)

3. Convert all amounts to baseCurrency using latest rates

4. Calculate totals:

total\_assets = SUM(all\_asset\_values\_in\_base\_currency)

total\_liabilities = SUM(all\_liability\_values\_in\_base\_currency)

net\_worth = total\_assets - total\_liabilities

5. Group and aggregate:

BY country: UK, SA, Offshore, Other

BY asset\_class: Cash, Investments, Property, Pensions, Protection, Other

BY currency: Original currency exposure

6. Fetch historical snapshots for trend

7. Calculate changes:

net\_worth\_change\_30d = current\_net\_worth - net\_worth\_30\_days\_ago

net\_worth\_change\_percent = (change / previous) \* 100

RESPONSE:

{

netWorth: {

total: decimal,

totalAssets: decimal,

totalLiabilities: decimal,

baseCurrency: string,

asOfDate: date,

lastUpdated: timestamp

},

byCountry: [

{country: 'UK', amount: decimal, percentage: decimal},

{country: 'SA', amount: decimal, percentage: decimal},

...

],

byAssetClass: [

{class: 'Cash', amount: decimal, percentage: decimal},

{class: 'Investments', amount: decimal, percentage: decimal},

...

],

byCurrency: [

{currency: 'GBP', amount: decimal, percentage: decimal},

{currency: 'ZAR', amount: decimal, percentage: decimal},

...

],

trend: [

{date: date, netWorth: decimal},

{date: date, netWorth: decimal},

... (last 12 months)

],

changes: {

day: {amount: decimal, percentage: decimal},

month: {amount: decimal, percentage: decimal},

year: {amount: decimal, percentage: decimal}

}

}

**User Flow:**

[User Login] → [Dashboard Landing]

↓

[Net Worth Summary - Hero Section]

- Large total net worth display

- Color-coded change indicator (green up, red down)

- Period selector (1D, 1M, 3M, 1Y, All)

↓

[Asset Allocation Section]

- Pie chart (by asset class)

- Bar chart (by country)

- Toggle between views

↓

[Net Worth Trend]

- Line chart with date range selector

- Hover to see specific date values

↓

[Quick Module Access Cards]

- Protection: Total cover amount

- Savings: Total saved

- Investments: Portfolio value

- Retirement: Pension pot

- IHT: Estate value

- Click card → Navigate to module

↓

[Refresh Button] → [Reload data]

**API Endpoints:**

* GET /api/v1/dashboard/net-worth - Get net worth summary
* GET /api/v1/dashboard/net-worth/trend - Historical trend data
* GET /api/v1/dashboard/snapshot - Create manual snapshot
* GET /api/v1/dashboard/currency-exposure - Detailed currency breakdown

**Data Models:**

TABLE: net\_worth\_snapshots

- id: UUID (PK)

- user\_id: UUID (FK to users)

- snapshot\_date: DATE

- total\_assets\_gbp: DECIMAL(15,2)

- total\_liabilities\_gbp: DECIMAL(15,2)

- net\_worth\_gbp: DECIMAL(15,2)

- total\_assets\_zar: DECIMAL(15,2)

- total\_liabilities\_zar: DECIMAL(15,2)

- net\_worth\_zar: DECIMAL(15,2)

- snapshot\_type: ENUM('AUTO', 'MANUAL', 'TAX\_YEAR\_END')

- created\_at: TIMESTAMP

TABLE: net\_worth\_by\_category

- id: UUID (PK)

- snapshot\_id: UUID (FK to net\_worth\_snapshots)

- category\_type: ENUM('COUNTRY', 'ASSET\_CLASS', 'CURRENCY')

- category\_value: VARCHAR(50)

- amount\_gbp: DECIMAL(15,2)

- amount\_zar: DECIMAL(15,2)

- percentage\_of\_total: DECIMAL(5,2)

TABLE: dashboard\_cache

- user\_id: UUID (PK)

- cache\_key: VARCHAR(255)

- cache\_data: JSONB

- expires\_at: TIMESTAMP

- created\_at: TIMESTAMP

VIEW: v\_user\_net\_worth\_current (materialized view, refreshed hourly)

- user\_id

- total\_cash (from savings)

- total\_investments (from investments)

- total\_pensions (from retirement)

- total\_property (from IHT assets)

- total\_liabilities (from IHT liabilities)

- net\_worth\_gbp

- net\_worth\_zar

- last\_calculated: TIMESTAMP

INDEX on net\_worth\_snapshots(user\_id, snapshot\_date DESC)

INDEX on net\_worth\_by\_category(snapshot\_id, category\_type)

INDEX on dashboard\_cache(user\_id, cache\_key, expires\_at)

**Error Handling:**

ERROR CASES:

1. No data available (new user)

- Response: 200 OK

- Return zero values with message: "Add your first account to see your net worth"

2. Exchange rate unavailable

- Response: 200 OK (partial data)

- Message: "Some currency conversions unavailable. Showing in original currencies"

- Flag affected items

3. Module data fetch timeout

- Response: 206 Partial Content

- Return available data with warning: "Some data unavailable. Showing cached values from {timestamp}"

4. Cache expired and refresh in progress

- Response: 200 OK

- Return cached data with indicator: "Data refreshing in background"

5. Snapshot creation failure

- Response: 500 Internal Server Error (log and retry)

- User sees previous snapshot

- Admin alert triggered

EDGE CASES:

- Negative net worth: Display normally with messaging

- First-time user: Show onboarding prompts

- Data inconsistency between modules: Flag for review, show best estimate

- Large portfolio (1000+ items): Implement pagination in detail views

- Multiple base currencies: User can switch, recalculate on fly

- Assets in cryptocurrencies: Fetch current rates, high volatility warning

- Liabilities exceeding assets: Risk indicator shown

**Performance Considerations:**

* Cache dashboard data for 5 minutes (Redis)
* Materialized view for net worth calculation (refresh hourly)
* Async aggregation for large portfolios
* Lazy loading for historical trend data
* Snapshot creation: Daily automated job (off-peak hours)
* Manual snapshots: Rate limited to 1 per hour
* Expected load: 50,000 dashboard views/day
* Target response time: <1 second (from cache)
* Cold load (no cache): <2 seconds
* Optimize queries with proper indexing
* Consider GraphQL for flexible data fetching
* Implement pagination for large datasets (>100 items)

**Feature 3.2: AI Recommendations Summary**

**Feature Name:** Personalized AI-Driven Action Recommendations

**User Story:** As a user, I want to see prioritized, actionable recommendations on my dashboard so that I know what financial actions to take next to improve my situation.

**Acceptance Criteria:**

* Display top 5 prioritized recommendations
* Recommendations categorized by urgency (Critical, High, Medium, Low)
* Each recommendation includes: title, description, estimated impact, required action
* User can dismiss recommendations
* User can mark recommendations as "in progress" or "completed"
* Recommendations update based on user actions
* Link to relevant module for action
* Explanation of why recommendation is made
* Estimated tax savings or benefit amount where applicable

**Technical Requirements:**

* Integration with AI Advisory Engine
* Rules engine for recommendation priority
* ML model for personalization (optional future enhancement)
* Recommendation tracking and effectiveness measurement
* Real-time recalculation when data changes

**Constraints:**

* Max 10 active recommendations per user
* Recommendations expire after 90 days if not acted upon
* Recalculate recommendations max once per day
* Must explain reasoning (no black box AI)

**Implementation Approach:**

ENDPOINT: GET /api/v1/dashboard/recommendations

QUERY PARAMS:

- limit: integer (default: 5, max: 20)

- priority: enum['CRITICAL', 'HIGH', 'MEDIUM', 'LOW'] (optional filter)

- category: string (optional: 'TAX', 'INVESTMENT', 'PROTECTION', etc.)

BUSINESS LOGIC:

1. Fetch user profile and all financial data

2. Run recommendation engine rules:

RULE EXAMPLES:

// Critical: Urgent action needed

IF user.protectionModule.lifeCover < (user.income.annual \* 10) AND user.dependents > 0:

CREATE recommendation(

priority: CRITICAL,

category: PROTECTION,

title: "Insufficient life cover for dependents",

description: "Your life cover is below recommended levels for your family",

estimatedImpact: "Protect your family's financial future",

action: "Review life cover options",

linkTo: "/protection"

)

// High: Significant opportunity

IF user.taxStatus.uk\_tax\_resident AND

user.income.annual < 100000 AND

user.savings.isa\_unused\_allowance > 10000:

CREATE recommendation(

priority: HIGH,

category: TAX,

title: "Use your £{amount} unused ISA allowance",

description: "Tax year ends {date}. ISA allowances don't roll over",

estimatedImpact: "Save up to £{tax\_saved} in tax on investment returns",

action: "Transfer savings to ISA",

linkTo: "/savings"

)

// Medium: Good practice

IF user.retirement.uk\_pension\_contributions < user.retirement.annual\_allowance \* 0.5:

CREATE recommendation(

priority: MEDIUM,

category: RETIREMENT,

title: "Consider increasing pension contributions",

description: "You're using only {percent}% of your pension allowance",

estimatedImpact: "Tax relief up to £{amount} available",

action: "Review pension contributions",

linkTo: "/retirement"

)

3. Score and rank recommendations:

score = base\_priority\_score + estimated\_impact\_value + urgency\_factor + personalization\_score

4. Filter dismissed and completed recommendations

5. Apply limit and return top N

RESPONSE:

{

recommendations: [

{

id: uuid,

priority: 'CRITICAL',

category: 'PROTECTION',

title: string,

description: string,

reasoning: string (why this recommendation),

estimatedImpact: {

description: string,

monetaryValue: decimal (optional),

currency: string

},

action: {

description: string,

linkTo: string (module URL),

externalLink: string (optional)

},

dueDate: date (optional, e.g., tax year end),

status: enum['NEW', 'VIEWED', 'IN\_PROGRESS', 'COMPLETED', 'DISMISSED'],

createdAt: timestamp,

lastUpdated: timestamp

},

...

],

metadata: {

totalRecommendations: integer,

byCriticalityCount: {critical: int, high: int, medium: int, low: int},

lastCalculated: timestamp,

nextCalculation: timestamp

}

}

**User Flow:**

[Dashboard Landing] → [Recommendations Section]

↓

[Prioritized List Display]

Each recommendation card shows:

- Priority badge (color-coded)

- Title

- Brief description

- Estimated impact (highlighted)

- Action button

↓

[User Actions per Card]

1. Click "Take Action" → Navigate to module

2. Click "Learn More" → Expand card with full reasoning

3. Mark as "In Progress" → Status updated

4. Dismiss → Card removed, feedback optional

↓

[Filter/Sort Options]

- By priority

- By category

- By estimated impact

↓

[View All Recommendations] → [Full recommendations page]

- Historical recommendations

- Completed recommendations with outcomes

- Effectiveness tracking

**API Endpoints:**

* GET /api/v1/dashboard/recommendations - Get active recommendations
* GET /api/v1/dashboard/recommendations/{id} - Get specific recommendation
* PUT /api/v1/dashboard/recommendations/{id}/status - Update status
* POST /api/v1/dashboard/recommendations/{id}/dismiss - Dismiss recommendation
* POST /api/v1/dashboard/recommendations/recalculate - Trigger recalculation
* GET /api/v1/dashboard/recommendations/history - Get historical recommendations

**Data Models:**

TABLE: ai\_recommendations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- priority: ENUM('CRITICAL', 'HIGH', 'MEDIUM', 'LOW')

- category: ENUM('TAX', 'INVESTMENT', 'PROTECTION', 'RETIREMENT', 'IHT', 'SAVINGS', 'GENERAL')

- title: VARCHAR(255)

- description: TEXT

- reasoning: TEXT

- estimated\_impact\_description: TEXT

- estimated\_impact\_value: DECIMAL(15,2)

- estimated\_impact\_currency: CHAR(3)

- action\_description: TEXT

- action\_link\_to: VARCHAR(255)

- action\_external\_link: VARCHAR(500)

- due\_date: DATE (optional)

- status: ENUM('NEW', 'VIEWED', 'IN\_PROGRESS', 'COMPLETED', 'DISMISSED')

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

- expires\_at: TIMESTAMP (created\_at + 90 days)

- completed\_at: TIMESTAMP

- dismissed\_at: TIMESTAMP

- score: DECIMAL(10,2) (for ranking)

TABLE: recommendation\_rules

- id: UUID (PK)

- rule\_name: VARCHAR(100) UNIQUE

- rule\_code: TEXT (stored rule logic reference)

- category: ENUM(...)

- base\_priority: ENUM(...)

- active: BOOLEAN

- version: INTEGER

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: recommendation\_feedback

- id: UUID (PK)

- recommendation\_id: UUID (FK to ai\_recommendations)

- user\_id: UUID (FK to users)

- action\_taken: ENUM('FOLLOWED', 'DISMISSED', 'PARTIALLY\_FOLLOWED')

- feedback\_text: TEXT

- rating: INTEGER (1-5)

- created\_at: TIMESTAMP

TABLE: recommendation\_effectiveness

- id: UUID (PK)

- recommendation\_id: UUID (FK to ai\_recommendations)

- user\_id: UUID (FK to users)

- predicted\_impact: DECIMAL(15,2)

- actual\_impact: DECIMAL(15,2) (measured post-action)

- accuracy\_score: DECIMAL(5,2)

- measurement\_date: TIMESTAMP

INDEX on ai\_recommendations(user\_id, status, priority, due\_date)

INDEX on ai\_recommendations(user\_id, created\_at DESC)

INDEX on ai\_recommendations(expires\_at) (for cleanup job)

**Error Handling:**

ERROR CASES:

1. Recommendation engine failure

- Response: 200 OK

- Return cached recommendations with flag: "Showing previous recommendations"

- Log error for investigation

- Retry engine in background

2. No recommendations available (all completed/dismissed)

- Response: 200 OK

- Message: "Great job! You're on top of your finances. Check back tomorrow for new insights"

3. Stale data (user changed something affecting recommendations)

- Response: 200 OK with flag: "Recommendations updating based on your recent changes"

- Trigger async recalculation

4. Invalid status update

- Response: 400 Bad Request

- Message: "Cannot mark dismissed recommendation as in progress"

5. Recommendation expired

- Response: 410 Gone

- Message: "This recommendation has expired. Refresh to see current recommendations"

EDGE CASES:

- Conflicting recommendations: Engine deduplicates and prioritizes

- Recommendation becomes invalid after user action: Auto-mark completed

- Seasonal recommendations (e.g., tax year end): Increase priority as deadline approaches

- User dismisses repeatedly: Reduce frequency of similar recommendations

- Circular recommendations: Detect and prevent (e.g., "invest more" vs "save more")

- Cross-module recommendations: Ensure recommendation considers all modules

- Rapid user data changes: Debounce recalculation (max once per hour)

**Performance Considerations:**

* Recommendation calculation: Async job, run nightly
* Trigger on-demand recalculation: Rate limited to once per hour
* Cache recommendation list: 1 hour TTL
* Rules engine: Optimize for <5 seconds execution time
* Expected recommendations per user: 5-15 active at any time
* Status updates: Real-time (no caching)
* Recommendation scoring: Pre-calculate during generation
* Historical recommendations: Archive after 1 year (keep summary only)
* Machine learning enhancement (future): Batch train models weekly
* A/B testing framework: Track which recommendations drive action

**4. PROTECTION MODULE**

**Feature 4.1: Life Assurance Policy Management**

**Feature Name:** Comprehensive Life Assurance Policy Tracking

**User Story:** As a user, I want to track all my life assurance policies across UK and SA, including coverage amounts, beneficiaries, and tax implications, so I can ensure my family is adequately protected.

**Acceptance Criteria:**

* User can add multiple life assurance policies
* Each policy includes: provider, policy type, cover amount, premium, term, beneficiaries
* Support for UK and SA policies with different tax treatments
* Track if policy written in trust (UK)
* Track estate duty implications (SA)
* Coverage gap analysis based on family needs
* Premium comparison across providers
* Beneficiary management with percentages
* Policy renewal reminders
* Claims process guidance

**Technical Requirements:**

* File upload for policy documents (PDF)
* OCR for policy data extraction (optional enhancement)
* Beneficiary relationship tracking
* Premium payment reminder system
* Integration with IHT module for estate planning
* Currency support (GBP, ZAR, USD, EUR)

**Constraints:**

* Policy documents: Max 10MB per file
* Beneficiaries: Max 10 per policy
* Historical policy records retained indefinitely (audit trail)
* Premium reminders: Email and in-app notification

**Implementation Approach:**

ENDPOINT: POST /api/v1/protection/life-assurance

REQUEST BODY:

{

policyNumber: string,

provider: string,

providerCountry: enum['UK', 'SA', 'OTHER'],

policyType: enum['TERM', 'WHOLE\_OF\_LIFE', 'DECREASING\_TERM', 'LEVEL\_TERM',

'INCREASING\_TERM', 'FAMILY\_INCOME\_BENEFIT', 'OTHER'],

coverAmount: decimal,

currency: string,

premiumAmount: decimal,

premiumFrequency: enum['MONTHLY', 'ANNUALLY', 'SINGLE'],

startDate: date,

endDate: date (NULL for whole of life),

writtenInTrust: boolean (UK only),

trustDetails: {

trustType: enum['BARE', 'DISCRETIONARY', 'INTEREST\_IN\_POSSESSION'],

trustees: array[string],

beneficiaries: array[{name: string, relationship: string, percentage: decimal}]

},

beneficiariesDirectPolicy: array[{

name: string,

dateOfBirth: date,

relationship: enum['SPOUSE', 'CHILD', 'PARENT', 'SIBLING', 'OTHER'],

percentage: decimal,

address: string

}],

indexationRate: decimal (optional, for increasing policies),

criticalIllnessRider: boolean,

waiverOfPremium: boolean,

notes: text

}

BUSINESS LOGIC:

1. Validate policy data:

- Sum of beneficiary percentages = 100%

- Cover amount > 0

- End date > start date (if applicable)

- Premium amount reasonable for cover amount

2. Determine tax treatment:

IF providerCountry = 'UK' AND writtenInTrust = TRUE:

outside\_uk\_estate\_for\_iht = TRUE

ELSE IF providerCountry = 'UK':

in\_uk\_estate\_for\_iht = TRUE

IF providerCountry = 'SA':

apply\_sa\_estate\_duty\_rules()

// SA policies generally part of estate

3. Calculate recommended cover (family needs analysis):

recommended\_cover = (annual\_income \* income\_multiplier) +

outstanding\_debts +

(children\_count \* education\_cost\_per\_child) +

funeral\_costs -

existing\_assets

coverage\_gap = recommended\_cover - total\_current\_cover

4. Store policy with encrypted document reference

5. Set up premium payment reminders

6. Link to IHT module if not in trust

RESPONSE:

{

id: uuid,

policyDetails: {...},

taxTreatment: {

ukIhtImpact: boolean,

saEstateDutyImpact: boolean,

writtenInTrust: boolean

},

coverageAnalysis: {

currentTotalCover: decimal,

recommendedCover: decimal,

coverageGap: decimal,

gapPercentage: decimal

},

annualPremiumCost: decimal

}

**User Flow:**

[Protection Dashboard] → [Life Assurance Tab]

↓

[Add Policy Button]

↓

[Policy Entry Form - Step 1: Provider Details]

- Policy number

- Provider name (autocomplete with popular providers)

- Provider country

- Policy type (visual cards with descriptions)

↓

[Policy Entry Form - Step 2: Cover Details]

- Cover amount and currency

- Premium amount and frequency

- Start date and end date (if applicable)

- Optional riders (CI, waiver)

↓

[Policy Entry Form - Step 3: Beneficiaries]

- Written in trust? (UK policies) → If YES: Trust details

- Beneficiary list (add multiple)

- Name, relationship, percentage

- Validation: percentages total 100%

↓

[Policy Entry Form - Step 4: Upload Document]

- Upload policy document (optional)

- OCR extraction attempt (if available)

- Review and confirm extracted data

↓

[Coverage Analysis Display]

- Current total cover

- Recommended cover (calculated)

- Gap visualization

- Recommendation: "Consider increasing cover by £X"

↓

[Save Policy]

↓

[Policy List View]

- Card view: Each policy with key details

- Total cover amount (header)

- Coverage gap indicator

- Filter: By provider, type, country

- Sort: By cover amount, premium, end date

**API Endpoints:**

* POST /api/v1/protection/life-assurance - Add policy
* PUT /api/v1/protection/life-assurance/{id} - Update policy
* DELETE /api/v1/protection/life-assurance/{id} - Delete policy (soft delete)
* GET /api/v1/protection/life-assurance - List all policies
* GET /api/v1/protection/life-assurance/{id} - Get specific policy
* POST /api/v1/protection/life-assurance/coverage-analysis - Run coverage needs analysis
* POST /api/v1/protection/life-assurance/{id}/upload-document - Upload policy document
* GET /api/v1/protection/life-assurance/{id}/document - Download policy document

**Data Models:**

TABLE: life\_assurance\_policies

- id: UUID (PK)

- user\_id: UUID (FK to users)

- policy\_number: VARCHAR(100)

- provider: VARCHAR(255)

- provider\_country: ENUM('UK', 'SA', 'OTHER')

- policy\_type: ENUM('TERM', 'WHOLE\_OF\_LIFE', 'DECREASING\_TERM', 'LEVEL\_TERM',

'INCREASING\_TERM', 'FAMILY\_INCOME\_BENEFIT', 'OTHER')

- cover\_amount: DECIMAL(15,2)

- currency: CHAR(3)

- cover\_amount\_gbp: DECIMAL(15,2) (calculated)

- cover\_amount\_zar: DECIMAL(15,2) (calculated)

- premium\_amount: DECIMAL(10,2)

- premium\_frequency: ENUM('MONTHLY', 'ANNUALLY', 'SINGLE')

- annual\_premium: DECIMAL(10,2) (calculated)

- start\_date: DATE

- end\_date: DATE (NULL for whole of life)

- written\_in\_trust: BOOLEAN DEFAULT FALSE

- trust\_type: ENUM('BARE', 'DISCRETIONARY', 'INTEREST\_IN\_POSSESSION')

- critical\_illness\_rider: BOOLEAN DEFAULT FALSE

- waiver\_of\_premium: BOOLEAN DEFAULT FALSE

- indexation\_rate: DECIMAL(5,2)

- uk\_iht\_impact: BOOLEAN (calculated)

- sa\_estate\_duty\_impact: BOOLEAN (calculated)

- document\_reference: UUID (FK to documents table)

- status: ENUM('ACTIVE', 'LAPSED', 'CLAIMED', 'MATURED')

- notes: TEXT

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: policy\_beneficiaries

- id: UUID (PK)

- policy\_id: UUID (FK to life\_assurance\_policies)

- name: VARCHAR(255)

- date\_of\_birth: DATE

- relationship: ENUM('SPOUSE', 'CHILD', 'PARENT', 'SIBLING', 'OTHER')

- percentage: DECIMAL(5,2)

- address: TEXT

- created\_at: TIMESTAMP

TABLE: policy\_trustees (for trust policies)

- id: UUID (PK)

- policy\_id: UUID (FK to life\_assurance\_policies)

- name: VARCHAR(255)

- relationship\_to\_policyholder: VARCHAR(100)

- created\_at: TIMESTAMP

TABLE: coverage\_needs\_analysis

- id: UUID (PK)

- user\_id: UUID (FK to users)

- analysis\_date: DATE

- annual\_income: DECIMAL(15,2)

- income\_multiplier: DECIMAL(3,1) (typically 10)

- outstanding\_debts: DECIMAL(15,2)

- number\_of\_children: INTEGER

- education\_cost\_per\_child: DECIMAL(15,2)

- funeral\_costs: DECIMAL(10,2)

- existing\_liquid\_assets: DECIMAL(15,2)

- recommended\_cover: DECIMAL(15,2)

- current\_total\_cover: DECIMAL(15,2)

- coverage\_gap: DECIMAL(15,2)

- notes: TEXT

- created\_at: TIMESTAMP

TABLE: premium\_reminders

- id: UUID (PK)

- policy\_id: UUID (FK to life\_assurance\_policies)

- reminder\_date: DATE

- reminder\_sent: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

INDEX on life\_assurance\_policies(user\_id, status)

INDEX on policy\_beneficiaries(policy\_id)

INDEX on premium\_reminders(reminder\_date, reminder\_sent)

CONSTRAINT: SUM(policy\_beneficiaries.percentage WHERE policy\_id = X) = 100

**Error Handling:**

ERROR CASES:

1. Beneficiary percentages don't total 100%

- Response: 400 Bad Request

- Message: "Beneficiary percentages must total 100%. Current total: {calculated\_total}%"

2. Invalid trust setup (trust selected but no trustees)

- Response: 400 Bad Request

- Message: "Trust policies must have at least one trustee"

3. End date before start date

- Response: 400 Bad Request

- Message: "Policy end date must be after start date"

4. Duplicate policy number for same provider

- Response: 409 Conflict

- Message: "A policy with this number already exists for {provider}"

- Option to update existing or confirm as separate policy

5. Document upload too large

- Response: 413 Payload Too Large

- Message: "Document size exceeds 10MB limit. Please compress or split the file"

6. Invalid currency code

- Response: 400 Bad Request

- Message: "Invalid currency code. Supported: GBP, ZAR, USD, EUR"

EDGE CASES:

- Whole of life policies: No end date (NULL), validation skip

- Single premium policies: Annual premium = premium\_amount, frequency = 'SINGLE'

- Multiple beneficiaries with same name: Allow, distinguish by relationship and DOB

- Policy in trust but user not UK resident: Warning (unusual but valid)

- Child beneficiary: Flag for review (may need trust structure)

- Policy matured: Status = 'MATURED', show in separate view

- Policy lapsed: Status = 'LAPSED', keep for reference but exclude from coverage total

- Indexation rate: Apply annually to cover amount (background job)

- Critical illness claim: Reduce life cover by CI payout amount

- Currency fluctuation: Recalculate GBP/ZAR values daily

**Performance Considerations:**

* Document storage: Use cloud storage (S3, Azure Blob)
* OCR processing: Async job (if implemented), may take 30-60 seconds
* Coverage analysis: Cache results for 30 days (recalculate if significant data change)
* Currency conversion: Daily batch job for all policies
* Premium reminders: Daily cron job, send 7 days before due date
* Expected policies per user: 1-5
* Policy list query: <500ms
* Document upload: Progress indicator for files >2MB
* Beneficiary validation: Client-side + server-side

**5. SAVINGS MODULE**

**Feature 5.1: Cash Account Management**

**Feature Name:** Multi-Currency Cash Account Tracking

**User Story:** As a user with bank accounts in UK and SA, I want to track all my cash accounts, see total savings, and understand the tax treatment of interest earned.

**Acceptance Criteria:**

* User can add bank accounts from UK, SA, and other countries
* Each account includes: bank name, account type, balance, currency, interest rate
* Support for current accounts, savings accounts, fixed deposits
* Track interest earned and tax treatment
* Calculate total savings across all accounts
* Emergency fund assessment
* Account categorization (emergency, short-term goals, long-term)
* Interest rate comparison and alerts for better rates
* Manual balance updates with historical tracking
* Bank account aggregation via Open Banking (future enhancement)
* ISA allowance tracking (UK)
* TFSA contribution tracking (SA)

**Technical Requirements:**

* Multi-currency support with real-time conversion
* Interest calculation engine (simple and compound)
* Historical balance tracking (snapshots)
* Tax calculation integration for interest income
* Open Banking API integration capability (Phase 2)
* Encryption for sensitive bank details

**Constraints:**

* Balance updates: Max 10 per day per account (prevent abuse)
* Historical data: Retain 7 years minimum (regulatory requirement)
* Interest rates: Percentage format, max 2 decimal places
* Account deletion: Soft delete only (audit trail)

**Implementation Approach:**

ENDPOINT: POST /api/v1/savings/accounts

REQUEST BODY:

{

bankName: string,

accountType: enum['CURRENT', 'SAVINGS', 'FIXED\_DEPOSIT', 'MONEY\_MARKET',

'CASH\_ISA', 'TFSA', 'NOTICE\_ACCOUNT', 'OTHER'],

accountNumber: string (last 4 digits only for security),

country: enum['UK', 'SA', 'OTHER'],

currency: string,

currentBalance: decimal,

interestRate: decimal (annual percentage),

interestPaymentFrequency: enum['MONTHLY', 'QUARTERLY', 'ANNUALLY', 'MATURITY'],

accountPurpose: enum['EMERGENCY\_FUND', 'SHORT\_TERM\_GOAL', 'LONG\_TERM\_SAVINGS', 'GENERAL'],

openDate: date,

maturityDate: date (optional, for fixed deposits),

noticePeriod: integer (days, for notice accounts),

accessRestrictions: text,

isJointAccount: boolean,

jointAccountHolder: string (if applicable)

}

BUSINESS LOGIC:

1. Validate account data:

- Balance >= 0

- Interest rate between 0 and 20% (sanity check)

- Account number encrypted before storage

2. Determine tax treatment:

IF country = 'UK':

apply\_uk\_interest\_tax\_rules()

// PSA: £1000 for basic rate, £500 for higher rate, £0 for additional rate

// Starting rate for savings: £5000 if income < £17,570

IF accountType = 'CASH\_ISA':

interest\_tax\_free = TRUE

track\_isa\_subscription()

IF country = 'SA':

apply\_sa\_interest\_exemption()

// First R23,800 exempt (under 65), R34,500 (65+)

IF accountType = 'TFSA':

interest\_tax\_free = TRUE

track\_tfsa\_contribution()

3. Calculate projected interest:

IF interestPaymentFrequency = 'MONTHLY':

monthly\_rate = annual\_rate / 12

projected\_annual = balance \* (1 + monthly\_rate)^12 - balance

ELSE IF interestPaymentFrequency = 'ANNUALLY':

projected\_annual = balance \* annual\_rate

4. Assess emergency fund adequacy:

IF accountPurpose = 'EMERGENCY\_FUND':

recommended\_emergency = user.monthly\_expenses \* 6

current\_emergency = SUM(accounts WHERE purpose = 'EMERGENCY\_FUND')

emergency\_fund\_status = current\_emergency / recommended\_emergency

5. Create balance snapshot

6. Check for better rates (background job)

RESPONSE:

{

id: uuid,

accountDetails: {...},

taxTreatment: {

interestTaxFree: boolean,

applicableTaxRate: decimal,

annualExemptionUsed: decimal,

annualExemptionRemaining: decimal

},

projections: {

projectedAnnualInterest: decimal,

projectedMonthlyInterest: decimal,

effectiveRate: decimal

},

conversionDetails: {

balanceGbp: decimal,

balanceZar: decimal,

exchangeRate: decimal

}

}

**User Flow:**

[Savings Dashboard] → [Cash Accounts Tab]

↓

[Account Summary Cards]

- Total savings (prominent)

- By currency breakdown

- By purpose breakdown

- Emergency fund status indicator

↓

[Add Account Button]

↓

[Account Entry Form - Step 1: Bank Details]

- Bank name (autocomplete with popular banks)

- Country

- Account type (visual cards with descriptions)

↓

[Account Entry Form - Step 2: Balance & Interest]

- Current balance and currency

- Interest rate (with comparison to market average)

- Interest payment frequency

- Account number (last 4 digits)

↓

[Account Entry Form - Step 3: Purpose & Dates]

- Account purpose (Emergency, Goals, etc.)

- Open date

- Maturity date (if applicable)

- Access restrictions/notice period

↓

[Tax Treatment Preview]

- Show if interest is taxable

- Exemption status (PSA/SA exemption)

- ISA/TFSA allowance impact

↓

[Save Account]

↓

[Account List View]

- Card view with key metrics per account

- Quick balance update buttons

- Filter: By type, country, purpose

- Sort: By balance, interest rate

- Visual indicators: High interest (green), Low interest (amber)

**API Endpoints:**

* POST /api/v1/savings/accounts - Add account
* PUT /api/v1/savings/accounts/{id} - Update account
* DELETE /api/v1/savings/accounts/{id} - Delete account (soft delete)
* GET /api/v1/savings/accounts - List all accounts
* GET /api/v1/savings/accounts/{id} - Get specific account
* POST /api/v1/savings/accounts/{id}/update-balance - Update balance
* GET /api/v1/savings/accounts/{id}/balance-history - Get balance history
* GET /api/v1/savings/summary - Get savings summary with totals
* GET /api/v1/savings/emergency-fund-status - Emergency fund assessment
* POST /api/v1/savings/compare-rates - Compare rates with market

**Data Models:**

TABLE: savings\_accounts

- id: UUID (PK)

- user\_id: UUID (FK to users)

- bank\_name: VARCHAR(255)

- account\_type: ENUM('CURRENT', 'SAVINGS', 'FIXED\_DEPOSIT', 'MONEY\_MARKET',

'CASH\_ISA', 'TFSA', 'NOTICE\_ACCOUNT', 'OTHER')

- account\_number\_encrypted: VARCHAR(255) (last 4 digits only, encrypted)

- country: ENUM('UK', 'SA', 'OTHER')

- currency: CHAR(3)

- current\_balance: DECIMAL(15,2)

- balance\_gbp: DECIMAL(15,2) (calculated)

- balance\_zar: DECIMAL(15,2) (calculated)

- interest\_rate: DECIMAL(5,2)

- interest\_payment\_frequency: ENUM('MONTHLY', 'QUARTERLY', 'ANNUALLY', 'MATURITY')

- account\_purpose: ENUM('EMERGENCY\_FUND', 'SHORT\_TERM\_GOAL', 'LONG\_TERM\_SAVINGS', 'GENERAL')

- open\_date: DATE

- maturity\_date: DATE

- notice\_period\_days: INTEGER

- access\_restrictions: TEXT

- is\_joint\_account: BOOLEAN DEFAULT FALSE

- joint\_account\_holder: VARCHAR(255)

- status: ENUM('ACTIVE', 'CLOSED', 'MATURED')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: account\_balance\_history

- id: UUID (PK)

- account\_id: UUID (FK to savings\_accounts)

- balance: DECIMAL(15,2)

- balance\_date: DATE

- update\_type: ENUM('MANUAL', 'AUTO', 'INTEREST\_PAYMENT', 'WITHDRAWAL', 'DEPOSIT')

- notes: TEXT

- created\_at: TIMESTAMP

TABLE: interest\_payments

- id: UUID (PK)

- account\_id: UUID (FK to savings\_accounts)

- payment\_date: DATE

- gross\_interest: DECIMAL(10,2)

- tax\_withheld: DECIMAL(10,2)

- net\_interest: DECIMAL(10,2)

- tax\_year\_uk: VARCHAR(7)

- tax\_year\_sa: VARCHAR(9)

- created\_at: TIMESTAMP

TABLE: isa\_contributions

- id: UUID (PK)

- user\_id: UUID (FK to users)

- account\_id: UUID (FK to savings\_accounts)

- tax\_year: VARCHAR(7) (e.g., '2024/25')

- contribution\_amount: DECIMAL(10,2)

- contribution\_date: DATE

- contribution\_type: ENUM('CASH\_ISA', 'STOCKS\_ISA', 'LISA', 'JUNIOR\_ISA')

- created\_at: TIMESTAMP

TABLE: tfsa\_contributions

- id: UUID (PK)

- user\_id: UUID (FK to users)

- account\_id: UUID (FK to savings\_accounts)

- tax\_year: VARCHAR(9) (e.g., '2024/2025')

- contribution\_amount: DECIMAL(10,2)

- contribution\_date: DATE

- lifetime\_contributions: DECIMAL(10,2) (running total)

- created\_at: TIMESTAMP

TABLE: emergency\_fund\_settings

- user\_id: UUID (PK, FK to users)

- monthly\_expenses: DECIMAL(10,2)

- target\_months: INTEGER (typically 3-6)

- target\_amount: DECIMAL(15,2) (calculated)

- updated\_at: TIMESTAMP

INDEX on savings\_accounts(user\_id, status)

INDEX on account\_balance\_history(account\_id, balance\_date DESC)

INDEX on interest\_payments(account\_id, payment\_date)

INDEX on isa\_contributions(user\_id, tax\_year)

INDEX on tfsa\_contributions(user\_id, tax\_year)

**Error Handling:**

ERROR CASES:

1. Negative balance entered

- Response: 400 Bad Request

- Message: "Account balance cannot be negative"

2. Interest rate exceeds reasonable threshold

- Response: 400 Bad Request

- Message: "Interest rate seems unusually high. Please verify (max 20%)"

3. ISA contribution exceeds annual allowance

- Response: 400 Bad Request

- Message: "This contribution would exceed your £{allowance} ISA allowance for {tax\_year}"

- Show: Current contributions: £{current}, Allowance remaining: £{remaining}

4. TFSA lifetime contribution exceeds R500,000

- Response: 400 Bad Request

- Message: "This contribution would exceed the R500,000 lifetime TFSA limit"

- Show: Current lifetime contributions: R{current}, Remaining: R{remaining}

5. Too many balance updates in a day

- Response: 429 Too Many Requests

- Message: "Maximum 10 balance updates per day reached. Please try again tomorrow"

6. Maturity date before open date

- Response: 400 Bad Request

- Message: "Maturity date must be after account opening date"

7. Closed account balance update attempt

- Response: 400 Bad Request

- Message: "Cannot update balance for a closed account"

EDGE CASES:

- Multiple ISAs of same type in same tax year: Warn user (HMRC rules allow but must track total)

- ISA transfer from previous year: Don't count against current year allowance

- TFSA withdrawal: Doesn't free up contribution room (lifetime limit)

- Joint account: Assume 50/50 ownership unless specified otherwise

- Foreign currency accounts: Daily exchange rate updates for GBP/ZAR equivalents

- Fixed deposit early withdrawal: Penalty calculation and note

- Notice account withdrawal: Flag if within notice period

- Account closure: Move to 'CLOSED' status, retain historical data

- Negative interest rates: Rare but valid (some currencies)

- Interest paid gross (no tax withheld): User responsible for declaring

- Accounts in cryptocurrencies: Not supported (too volatile for savings)

**Performance Considerations:**

* Balance history: Paginate if >100 entries
* Currency conversion: Cache exchange rates (1 day TTL)
* Interest calculations: Pre-calculate and store monthly
* Emergency fund assessment: Cache for 1 week or until balance changes
* Balance update: Real-time, <200ms response
* Account list query with balances: <500ms
* Expected accounts per user: 3-10
* Historical balance queries: Index on date field
* ISA/TFSA allowance checks: Cache current tax year data
* Interest payment tracking: Background job (monthly)
* Market rate comparison: Daily batch job, cache results

**Feature 5.2: ISA and TFSA Allowance Tracking**

**Feature Name:** Tax-Advantaged Savings Allowance Management

**User Story:** As a UK tax resident, I want to track my ISA allowance usage across all ISA types, and as an SA tax resident, I want to track my TFSA contributions, so I can maximize my tax-efficient savings.

**Acceptance Criteria:**

* Track ISA allowance for current tax year (UK: April-April)
* Support Cash ISA, Stocks & Shares ISA, LISA, Junior ISA
* Show allowance used and remaining
* Alert when approaching limit (90% used)
* Track TFSA contributions for SA (March-February tax year)
* TFSA lifetime limit tracking (R500,000)
* Historical allowance usage by tax year
* Suggest optimal use of remaining allowance
* ISA transfer tracking (doesn't count against allowance)
* Flexible ISA rule support (withdrawals and recontributions)

**Technical Requirements:**

* Tax year calculation logic (UK and SA)
* Running total calculations
* ISA transfer vs new contribution differentiation
* Flexible ISA withdrawal tracking
* Alert/notification system
* Historical data migration when allowances change

**Constraints:**

* ISA allowance (2024/25): £20,000 across all ISAs
* LISA: £4,000 per year (part of overall £20,000)
* Junior ISA: £9,000 per year (separate from adult allowance)
* TFSA annual limit: R36,000 (as of last update)
* TFSA lifetime limit: R500,000
* Cannot backdate contributions to previous tax years

**Implementation Approach:**

ENDPOINT: GET /api/v1/savings/tax-efficient-allowances

QUERY PARAMS:

- taxYear: string (e.g., 'UK-2024/25' or 'SA-2024/2025')

- country: enum['UK', 'SA']

BUSINESS LOGIC:

1. Determine current tax year:

UK: April 6 to April 5

SA: March 1 to February 28/29

2. Fetch allowance limits (from configuration):

uk\_isa\_limit = 20000 // £20,000 for 2024/25

uk\_lisa\_limit = 4000 // Within overall limit

uk\_junior\_isa\_limit = 9000

sa\_tfsa\_annual\_limit = 36000 // R36,000

sa\_tfsa\_lifetime\_limit = 500000 // R500,000

3. Calculate usage:

FOR UK:

total\_isa\_contributions = SUM(isa\_contributions WHERE tax\_year = current\_tax\_year)

lisa\_contributions = SUM(isa\_contributions WHERE type = 'LISA' AND tax\_year = current\_tax\_year)

isa\_allowance\_remaining = uk\_isa\_limit - total\_isa\_contributions

lisa\_allowance\_remaining = uk\_lisa\_limit - lisa\_contributions

// Flexible ISA adjustments

IF has\_flexible\_isa:

flexible\_isa\_withdrawals = SUM(withdrawals WHERE tax\_year = current\_tax\_year)

additional\_allowance = flexible\_isa\_withdrawals

adjusted\_allowance\_remaining = isa\_allowance\_remaining + additional\_allowance

FOR SA:

tfsa\_contributions\_this\_year = SUM(tfsa\_contributions WHERE tax\_year = current\_tax\_year)

tfsa\_lifetime\_total = SUM(tfsa\_contributions WHERE created\_at <= NOW())

tfsa\_annual\_remaining = sa\_tfsa\_annual\_limit - tfsa\_contributions\_this\_year

tfsa\_lifetime\_remaining = sa\_tfsa\_lifetime\_limit - tfsa\_lifetime\_total

4. Check alert thresholds:

IF isa\_allowance\_remaining < (uk\_isa\_limit \* 0.1):

trigger\_alert("90% of ISA allowance used")

IF days\_until\_tax\_year\_end < 30 AND isa\_allowance\_remaining > 5000:

trigger\_alert("Significant ISA allowance unused, tax year ending soon")

5. Generate recommendations:

recommend\_isa\_usage()

recommend\_tfsa\_usage()

RESPONSE:

{

country: 'UK',

taxYear: '2024/25',

taxYearStart: date,

taxYearEnd: date,

daysRemainingInTaxYear: integer,

isaAllowances: {

overallLimit: 20000,

used: decimal,

remaining: decimal,

percentageUsed: decimal,

byType: [

{type: 'CASH\_ISA', used: decimal},

{type: 'STOCKS\_ISA', used: decimal},

{type: 'LISA', used: decimal, limit: 4000, remaining: decimal}

],

flexibleIsaWithdrawals: decimal (if applicable),

additionalAllowanceFromWithdrawals: decimal

},

tfsaAllowances: {

annualLimit: 36000,

annualUsed: decimal,

annualRemaining: decimal,

lifetimeLimit: 500000,

lifetimeUsed: decimal,

lifetimeRemaining: decimal,

percentageOfLifetimeUsed: decimal

},

alerts: [

{

severity: enum['INFO', 'WARNING', 'CRITICAL'],

message: string,

actionRequired: string

}

],

recommendations: [

{

title: string,

description: string,

estimatedBenefit: decimal

}

],

historicalUsage: [

{taxYear: string, isaUsed: decimal, tfsaUsed: decimal},

...

]

}

**User Flow:**

[Savings Dashboard] → [Tax-Efficient Allowances Card]

↓

[Allowance Overview Display]

- ISA Allowance (if UK resident):

- Progress bar: Used vs Remaining

- Breakdown by ISA type

- Days remaining in tax year

- TFSA Allowance (if SA resident):

- Annual: Progress bar

- Lifetime: Progress bar with milestone markers

↓

[Contribution Actions]

- "Add ISA Contribution" button → Links to add account or update balance

- "Add TFSA Contribution" button → Links to add account or update balance

↓

[Alerts Section]

- Critical: Red banner (e.g., "Only £2,000 ISA allowance left, 45 days until tax year end")

- Warning: Amber banner

- Info: Blue banner

↓

[Recommendations Section]

- AI-generated suggestions

- "Transfer cash savings to ISA" with estimated tax saving

- "Maximize LISA for home purchase bonus" (25% government bonus)

↓

[Historical Tab]

- Table: Tax year, ISA used, TFSA used

- Chart: Allowance usage trend over years

↓

[Settings/Configuration]

- Set notification preferences

- Customize alert thresholds

**API Endpoints:**

* GET /api/v1/savings/tax-efficient-allowances - Get allowance summary
* GET /api/v1/savings/tax-efficient-allowances/history - Historical usage
* POST /api/v1/savings/isa-contribution - Record ISA contribution
* POST /api/v1/savings/tfsa-contribution - Record TFSA contribution
* POST /api/v1/savings/isa-transfer - Record ISA transfer (doesn't count against allowance)
* POST /api/v1/savings/flexible-isa-withdrawal - Record flexible ISA withdrawal
* GET /api/v1/savings/allowance-alerts - Get current alerts
* PUT /api/v1/savings/allowance-alerts/preferences - Update alert preferences

**Data Models:**

TABLE: tax\_efficient\_allowances\_config

- id: UUID (PK)

- country: ENUM('UK', 'SA')

- tax\_year: VARCHAR(10)

- allowance\_type: ENUM('ISA\_OVERALL', 'LISA', 'JUNIOR\_ISA', 'TFSA\_ANNUAL', 'TFSA\_LIFETIME')

- limit\_amount: DECIMAL(10,2)

- currency: CHAR(3)

- effective\_from: DATE

- effective\_to: DATE

- notes: TEXT

TABLE: isa\_transfers

- id: UUID (PK)

- user\_id: UUID (FK to users)

- from\_provider: VARCHAR(255)

- to\_provider: VARCHAR(255)

- amount: DECIMAL(10,2)

- isa\_type: ENUM('CASH\_ISA', 'STOCKS\_ISA', 'LISA')

- tax\_year\_transferred: VARCHAR(7)

- transfer\_date: DATE

- transfer\_reference: VARCHAR(100)

- created\_at: TIMESTAMP

TABLE: flexible\_isa\_withdrawals

- id: UUID (PK)

- account\_id: UUID (FK to savings\_accounts)

- withdrawal\_amount: DECIMAL(10,2)

- withdrawal\_date: DATE

- tax\_year: VARCHAR(7)

- recontribution\_allowance\_created: DECIMAL(10,2)

- recontribution\_allowance\_used: DECIMAL(10,2)

- expires\_at: DATE (end of tax year)

- created\_at: TIMESTAMP

TABLE: allowance\_alerts

- id: UUID (PK)

- user\_id: UUID (FK to users)

- alert\_type: ENUM('ISA\_90\_PERCENT', 'ISA\_TAX\_YEAR\_END', 'TFSA\_90\_PERCENT',

'TFSA\_LIFETIME\_APPROACHING', 'TFSA\_TAX\_YEAR\_END')

- severity: ENUM('INFO', 'WARNING', 'CRITICAL')

- message: TEXT

- action\_required: TEXT

- triggered\_at: TIMESTAMP

- dismissed: BOOLEAN DEFAULT FALSE

- dismissed\_at: TIMESTAMP

TABLE: allowance\_alert\_preferences

- user\_id: UUID (PK, FK to users)

- isa\_allowance\_alerts\_enabled: BOOLEAN DEFAULT TRUE

- isa\_alert\_threshold\_percentage: DECIMAL(5,2) DEFAULT 90.0

- tfsa\_allowance\_alerts\_enabled: BOOLEAN DEFAULT TRUE

- tfsa\_alert\_threshold\_percentage: DECIMAL(5,2) DEFAULT 90.0

- tax\_year\_end\_reminder\_days: INTEGER DEFAULT 30

- notification\_channels: JSON ['EMAIL', 'IN\_APP', 'SMS']

- updated\_at: TIMESTAMP

VIEW: v\_current\_allowance\_usage (materialized view, refreshed daily)

- user\_id

- country

- current\_tax\_year

- isa\_overall\_used

- isa\_overall\_remaining

- lisa\_used

- lisa\_remaining

- tfsa\_annual\_used

- tfsa\_annual\_remaining

- tfsa\_lifetime\_used

- tfsa\_lifetime\_remaining

- last\_updated: TIMESTAMP

INDEX on isa\_contributions(user\_id, tax\_year)

INDEX on tfsa\_contributions(user\_id, tax\_year)

INDEX on isa\_transfers(user\_id, tax\_year\_transferred)

INDEX on allowance\_alerts(user\_id, triggered\_at, dismissed)

**Error Handling:**

ERROR CASES:

1. ISA contribution exceeds remaining allowance

- Response: 400 Bad Request

- Message: "This contribution of £{amount} would exceed your ISA allowance by £{excess}"

- Suggestion: "Maximum contribution allowed: £{remaining}"

2. TFSA contribution exceeds annual limit

- Response: 400 Bad Request

- Message: "This contribution of R{amount} would exceed your TFSA annual limit by R{excess}"

3. TFSA contribution exceeds lifetime limit

- Response: 400 Bad Request

- Message: "This contribution would exceed the R500,000 lifetime TFSA limit"

- Show: "You have R{remaining} lifetime allowance remaining"

4. LISA contribution for user over 50

- Response: 400 Bad Request

- Message: "LISA contributions not permitted after age 50"

5. Multiple adult ISAs of same type in same tax year

- Response: 400 Bad Request

- Message: "You can only contribute to one Cash ISA per tax year"

- Suggestion: "Consider an ISA transfer instead"

6. Backdated contribution to previous tax year

- Response: 400 Bad Request

- Message: "Cannot add contributions to previous tax year {year}"

7. ISA transfer amount doesn't match provider statement

- Response: 400 Bad Request (validation warning)

- Message: "Transfer amount seems inconsistent with typical ISA values. Please verify"

EDGE CASES:

- User turns 18 mid-tax-year: Junior ISA converts to adult ISA, track both allowances

- User becomes UK resident mid-tax-year: Pro-rata ISA allowance (actually full allowance available)

- User becomes SA resident mid-tax-year: Pro-rata TFSA allowance (actually annual limit applies)

- ISA provider failure: Transferred ISAs may have delays, don't count against new allowance

- Flexible ISA withdrawal and recontribution in same tax year: Track carefully

- LISA government bonus: 25% bonus paid by government, doesn't count against allowance

- LISA withdrawal before 60 (except first home/terminal illness): 25% penalty

- Help to Buy ISA (closed to new savers): Existing accounts can continue

- Bed and ISA: Selling shares to buy within ISA, counts as new contribution

- In-specie ISA transfer: Value at transfer date, not original purchase value

- TFSA withdrawal: Never creates additional contribution room (unlike UK pension annual allowance)

- Death of ISA holder: Becomes a "continuing ISA" with additional permitted subscription for spouse

**Performance Considerations:**

* Allowance calculation: Use materialized view, refresh daily
* Real-time contribution validation: Cache current tax year allowances
* Alert generation: Daily batch job at midnight
* Historical usage queries: Pre-aggregate by tax year
* Expected API calls: High frequency during tax year-end (March-April)
* Response time target: <200ms for allowance summary
* Alert check: <50ms (from cache)
* Flexible ISA calculation complexity: Cache withdrawal allowances
* Tax year-end surge: Scale infrastructure, expect 10x normal traffic
* Notification dispatch: Queue-based async processing

**6. INVESTMENT MODULE**

**Feature 6.1: Portfolio Management**

**Feature Name:** Comprehensive Investment Portfolio Tracking

**User Story:** As an investor with holdings across UK and SA markets, I want to track all my investments including stocks, funds, ISAs, and tax-advantaged schemes so that I can monitor performance and manage tax efficiently.

**Acceptance Criteria:**

* Track holdings in Stocks & Shares ISA, General Investment Account (GIA), Unit Trusts, ETFs
* Support for VCTs, EIS, SEIS investments with tax relief tracking
* SA investments: Unit Trusts, ETFs, JSE-listed stocks
* Record purchase price, current value, unrealized gains/losses
* Track dividend income by source country
* Capital gains tracking (realized and unrealized)
* Asset allocation analysis
* Performance metrics vs benchmarks
* Currency exposure reporting
* Tax lot tracking (FIFO, average cost methods)

**Technical Requirements:**

* Market data integration for pricing (manual or API)
* Capital gains calculation engine (UK and SA rules)
* Dividend tax treatment calculation
* Asset allocation algorithms
* Performance calculation (time-weighted returns)
* Corporate action tracking (splits, mergers, dividends)
* EIS/SEIS/VCT holding period tracking (for tax relief retention)

**Constraints:**

* Market data refresh: Real-time for premium, daily for standard
* Historical price data: 10 years minimum
* Tax lot tracking: Required for accurate CGT calculation
* EIS/SEIS minimum holding: 3 years for tax relief retention
* VCT minimum holding: 5 years for tax relief retention

**Implementation Approach:**

ENDPOINT: POST /api/v1/investments/holdings

REQUEST BODY:

{

accountType: enum['STOCKS\_ISA', 'GIA', 'UNIT\_TRUST', 'VCT', 'EIS', 'SEIS',

'SA\_UNIT\_TRUST', 'SA\_ETF', 'SA\_DIRECT\_SHARES', 'OFFSHORE\_BOND'],

accountProvider: string,

accountNumber: string (last 4 digits),

country: enum['UK', 'SA', 'OFFSHORE'],

holdings: [

{

securityType: enum['STOCK', 'FUND', 'ETF', 'BOND', 'VCT', 'EIS\_SHARE', 'SEIS\_SHARE'],

ticker: string,

name: string,

quantity: decimal,

purchaseDate: date,

purchasePrice: decimal,

purchaseCurrency: string,

currentPrice: decimal,

currentValue: decimal,

unrealizedGain: decimal,

assetClass: enum['EQUITY', 'FIXED\_INCOME', 'PROPERTY', 'COMMODITY', 'CASH', 'ALTERNATIVE'],

region: enum['UK', 'US', 'EUROPE', 'ASIA', 'EMERGING', 'GLOBAL'],

sector: string

}

],

taxReliefClaimed: { // For EIS/SEIS/VCT

reliefType: enum['INCOME\_TAX\_RELIEF', 'CGT\_EXEMPTION', 'CGT\_DEFERRAL'],

reliefAmount: decimal,

taxYear: string,

holdingPeriodRequired: integer (years),

holdingPeriodEndDate: date

}

}

BUSINESS LOGIC:

1. Validate holding data:

- Quantity > 0

- Purchase price >= 0

- Purchase date <= today

2. Calculate metrics:

unrealized\_gain = (current\_price - purchase\_price) \* quantity

unrealized\_gain\_percentage = (unrealized\_gain / (purchase\_price \* quantity)) \* 100

IF accountType = 'STOCKS\_ISA':

gains\_tax\_free = TRUE

dividends\_tax\_free = TRUE

ELSE IF accountType = 'GIA':

apply\_uk\_cgt\_rules()

apply\_uk\_dividend\_tax\_rules()

IF country = 'SA':

apply\_sa\_cgt\_rules() // Inclusion rate method

apply\_sa\_dividend\_tax\_rules() // Dividend withholding tax

3. Track tax lots for CGT:

// FIFO method for UK

// Average cost method allowed for SA

tax\_lot = {

purchase\_date: date,

quantity: decimal,

cost\_basis: decimal,

disposal\_method: 'FIFO' or 'AVERAGE\_COST'

}

4. VCT/EIS/SEIS tracking:

IF accountType IN ['VCT', 'EIS', 'SEIS']:

holding\_period\_remaining = required\_period - years\_held

at\_risk\_of\_losing\_relief = (holding\_period\_remaining > 0)

IF at\_risk\_of\_losing\_relief:

CREATE alert("Hold for {remaining} more years to retain tax relief")

5. Asset allocation calculation:

total\_portfolio\_value = SUM(all\_holdings.current\_value)

FOR EACH holding:

allocation\_percentage = (holding.current\_value / total\_portfolio\_value) \* 100

GROUP BY asset\_class, region, sector

6. Performance calculation:

time\_weighted\_return = calculate\_twr(cash\_flows, valuations)

vs\_benchmark = portfolio\_return - benchmark\_return

RESPONSE:

{

id: uuid,

accountDetails: {...},

holdings: [...],

portfolioMetrics: {

totalValue: {gbp: decimal, zar: decimal},

totalCost: {gbp: decimal, zar: decimal},

totalUnrealizedGain: {amount: decimal, percentage: decimal},

totalRealizedGain: decimal (from previous disposals),

assetAllocation: [

{assetClass: string, value: decimal, percentage: decimal},

...

],

regionAllocation: [...],

sectorAllocation: [...],

currencyExposure: [...]

},

taxImplications: {

isaHoldings: {value: decimal, gains: decimal, taxFree: true},

giaHoldings: {

value: decimal,

unrealizedGains: decimal,

estimatedCgtLiability: decimal,

cgtAllowanceUsed: decimal,

cgtAllowanceRemaining: decimal

},

taxReliefAtRisk: [

{type: string, amount: decimal, holdUntil: date}

]

},

performance: {

timeWeightedReturn: decimal,

moneyWeightedReturn: decimal,

benchmarkComparison: {

portfolioReturn: decimal,

benchmarkReturn: decimal,

outperformance: decimal

}

}

}

**User Flow:**

[Investment Dashboard] → [Portfolio Overview]

↓

[Portfolio Summary Cards]

- Total Portfolio Value (prominent)

- Unrealized Gain/Loss (color-coded)

- Asset Allocation Pie Chart

- Performance vs Benchmark

↓

[Add Investment Account Button]

↓

[Account Entry - Step 1: Account Type]

- Select account type (visual cards):

- Stocks & Shares ISA

- General Investment Account

- VCT/EIS/SEIS

- SA Unit Trust / ETF

- Offshore Bond

↓

[Account Entry - Step 2: Provider Details]

- Provider name (autocomplete)

- Account number (last 4 digits)

- Country

↓

[Account Entry - Step 3: Add Holdings]

- Manual entry:

- Search security by ticker/name

- Enter quantity, purchase price, date

- Bulk import:

- Upload CSV file

- Map columns to fields

- Validate and import

- For VCT/EIS/SEIS:

- Tax relief details

- Holding period tracker

↓

[Holdings List View]

- Table with sortable columns:

- Security, Quantity, Cost, Value, Gain/Loss, %

- Filter: By account, asset class, region

- Color coding: Gains (green), Losses (red)

↓

[Detailed Holding View] (click on any holding)

- Purchase history (tax lots)

- Dividend history

- Corporate actions

- Performance chart

- Tax lot tracking

- CGT calculator (if GIA)

↓

[Portfolio Analysis Tab]

- Asset allocation (multiple views):

- By asset class

- By region

- By sector

- By currency

- Diversification score

- Risk metrics

- Rebalancing recommendations

**API Endpoints:**

* POST /api/v1/investments/accounts - Add investment account
* PUT /api/v1/investments/accounts/{id} - Update account
* DELETE /api/v1/investments/accounts/{id} - Delete account (soft delete)
* POST /api/v1/investments/accounts/{accountId}/holdings - Add holding
* PUT /api/v1/investments/holdings/{id} - Update holding
* DELETE /api/v1/investments/holdings/{id} - Delete holding (soft delete)
* POST /api/v1/investments/holdings/{id}/update-price - Update current price
* GET /api/v1/investments/portfolio - Get complete portfolio summary
* GET /api/v1/investments/portfolio/performance - Performance metrics
* GET /api/v1/investments/portfolio/asset-allocation - Asset allocation breakdown
* GET /api/v1/investments/cgt-calculator - CGT liability calculator
* POST /api/v1/investments/holdings/bulk-import - Bulk import holdings
* GET /api/v1/investments/dividends - Dividend income report
* GET /api/v1/investments/tax-relief-tracker - VCT/EIS/SEIS tracker

**Data Models:**

TABLE: investment\_accounts

- id: UUID (PK)

- user\_id: UUID (FK to users)

- account\_type: ENUM('STOCKS\_ISA', 'GIA', 'UNIT\_TRUST', 'VCT', 'EIS', 'SEIS',

'SA\_UNIT\_TRUST', 'SA\_ETF', 'SA\_DIRECT\_SHARES', 'OFFSHORE\_BOND')

- provider: VARCHAR(255)

- account\_number\_encrypted: VARCHAR(255)

- country: ENUM('UK', 'SA', 'OFFSHORE')

- base\_currency: CHAR(3)

- account\_open\_date: DATE

- status: ENUM('ACTIVE', 'CLOSED')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: investment\_holdings

- id: UUID (PK)

- account\_id: UUID (FK to investment\_accounts)

- security\_type: ENUM('STOCK', 'FUND', 'ETF', 'BOND', 'VCT', 'EIS\_SHARE', 'SEIS\_SHARE')

- ticker: VARCHAR(20)

- isin: VARCHAR(12)

- security\_name: VARCHAR(255)

- quantity: DECIMAL(15,4)

- average\_cost\_basis: DECIMAL(15,4) (calculated)

- current\_price: DECIMAL(15,4)

- price\_currency: CHAR(3)

- current\_value: DECIMAL(15,2) (calculated)

- unrealized\_gain: DECIMAL(15,2) (calculated)

- unrealized\_gain\_percentage: DECIMAL(10,2) (calculated)

- asset\_class: ENUM('EQUITY', 'FIXED\_INCOME', 'PROPERTY', 'COMMODITY', 'CASH', 'ALTERNATIVE')

- region: ENUM('UK', 'US', 'EUROPE', 'ASIA', 'EMERGING', 'GLOBAL')

- sector: VARCHAR(100)

- last\_price\_update: TIMESTAMP

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: tax\_lots

- id: UUID (PK)

- holding\_id: UUID (FK to investment\_holdings)

- purchase\_date: DATE

- quantity: DECIMAL(15,4)

- purchase\_price: DECIMAL(15,4)

- purchase\_currency: CHAR(3)

- cost\_basis\_gbp: DECIMAL(15,2)

- cost\_basis\_zar: DECIMAL(15,2)

- exchange\_rate: DECIMAL(10,6)

- disposal\_date: DATE (NULL if not disposed)

- disposal\_quantity: DECIMAL(15,4)

- disposal\_proceeds: DECIMAL(15,2)

- realized\_gain: DECIMAL(15,2) (calculated on disposal)

- cgt\_tax\_year: VARCHAR(7) (UK) or VARCHAR(9) (SA)

- disposal\_method: ENUM('FIFO', 'AVERAGE\_COST', 'SPECIFIC\_IDENTIFICATION')

- created\_at: TIMESTAMP

TABLE: dividends

- id: UUID (PK)

- holding\_id: UUID (FK to investment\_holdings)

- payment\_date: DATE

- ex\_dividend\_date: DATE

- dividend\_per\_share: DECIMAL(10,4)

- total\_dividend\_gross: DECIMAL(10,2)

- withholding\_tax: DECIMAL(10,2)

- total\_dividend\_net: DECIMAL(10,2)

- currency: CHAR(3)

- source\_country: ENUM('UK', 'SA', 'US', 'OTHER')

- uk\_tax\_year: VARCHAR(7)

- sa\_tax\_year: VARCHAR(9)

- qualified\_dividend: BOOLEAN (for US stocks)

- created\_at: TIMESTAMP

TABLE: corporate\_actions

- id: UUID (PK)

- holding\_id: UUID (FK to investment\_holdings)

- action\_type: ENUM('SPLIT', 'REVERSE\_SPLIT', 'MERGER', 'SPINOFF', 'RIGHTS\_ISSUE',

'BONUS\_ISSUE', 'DELISTING')

- action\_date: DATE

- ratio: VARCHAR(20) (e.g., '2:1' for split)

- description: TEXT

- quantity\_before: DECIMAL(15,4)

- quantity\_after: DECIMAL(15,4)

- cost\_basis\_adjustment: DECIMAL(15,2)

- created\_at: TIMESTAMP

TABLE: tax\_relief\_schemes

- id: UUID (PK)

- holding\_id: UUID (FK to investment\_holdings)

- scheme\_type: ENUM('VCT', 'EIS', 'SEIS')

- investment\_date: DATE

- investment\_amount: DECIMAL(15,2)

- income\_tax\_relief\_claimed: DECIMAL(10,2)

- income\_tax\_relief\_percentage: DECIMAL(5,2) (30% EIS, 50% SEIS, 30% VCT)

- tax\_year\_claimed: VARCHAR(7)

- minimum\_holding\_period\_years: INTEGER (5 for VCT, 3 for EIS/SEIS)

- holding\_period\_end\_date: DATE (calculated)

- cgt\_deferral\_claimed: DECIMAL(10,2) (EIS only)

- cgt\_exemption\_eligible: BOOLEAN (EIS/SEIS after holding period)

- relief\_withdrawn: BOOLEAN DEFAULT FALSE

- relief\_withdrawal\_reason: TEXT

- created\_at: TIMESTAMP

TABLE: portfolio\_snapshots

- id: UUID (PK)

- user\_id: UUID (FK to users)

- snapshot\_date: DATE

- total\_value\_gbp: DECIMAL(15,2)

- total\_value\_zar: DECIMAL(15,2)

- total\_cost\_basis\_gbp: DECIMAL(15,2)

- unrealized\_gain\_gbp: DECIMAL(15,2)

- realized\_gain\_ytd\_gbp: DECIMAL(15,2)

- asset\_allocation\_json: JSON

- performance\_metrics\_json: JSON

- created\_at: TIMESTAMP

VIEW: v\_portfolio\_summary (materialized view, refreshed on demand)

- user\_id

- total\_portfolio\_value\_gbp

- total\_portfolio\_value\_zar

- total\_unrealized\_gain

- total\_realized\_gain\_ytd

- isa\_holdings\_value

- gia\_holdings\_value

- vct\_eis\_seis\_value

- estimated\_cgt\_liability

- last\_updated: TIMESTAMP

INDEX on investment\_holdings(account\_id, deleted)

INDEX on tax\_lots(holding\_id, disposal\_date)

INDEX on dividends(holding\_id, payment\_date)

INDEX on tax\_relief\_schemes(holding\_id, holding\_period\_end\_date)

INDEX on portfolio\_snapshots(user\_id, snapshot\_date DESC)

**Error Handling:**

ERROR CASES:

1. Negative quantity or price

- Response: 400 Bad Request

- Message: "Quantity and price must be positive values"

2. Purchase date in future

- Response: 400 Bad Request

- Message: "Purchase date cannot be in the future"

3. ISA holding exceeds subscription limits

- Response: 400 Bad Request

- Message: "This holding would exceed your ISA subscription limit for {tax\_year}"

4. VCT/EIS/SEIS disposal before holding period

- Response: 400 Bad Request (validation warning)

- Message: "Disposing before {date} will result in loss of £{amount} tax relief"

- Allow user to confirm and proceed

5. Tax lot disposal quantity exceeds available

- Response: 400 Bad Request

- Message: "Cannot dispose {quantity} shares. Only {available} shares available from this tax lot"

6. Invalid ticker symbol

- Response: 404 Not Found

- Message: "Unable to find security with ticker {ticker}. Please verify or enter manually"

7. Currency mismatch

- Response: 400 Bad Request

- Message: "Security {name} trades in {currency}, but you entered {entered\_currency}"

8. Duplicate holding in same account

- Response: 409 Conflict (warning)

- Message: "You already have a holding of {security} in this account. Merge or add as separate lot?"

EDGE CASES:

- Stock split: Adjust all tax lots proportionally, maintain cost basis

- Reverse split: Consolidate shares, maintain cost basis

- Merger/acquisition: Create new holdings for acquirer, close acquired

- Spinoff: Allocate cost basis between parent and spun-off entity

- Rights issue: New tax lot at subscription price

- Bonus issue: Additional shares, zero cost basis, adjust average cost

- Bed and ISA: Dispose from GIA (CGT event), purchase in ISA (new tax lot)

- Same-day rule (UK CGT): Disposals matched with acquisitions on same day first

- 30-day rule (UK CGT): Disposals matched with acquisitions within 30 days (prevent bed and breakfasting)

- Section 104 holding (UK): Pooled holding with average cost

- Dividend reinvestment: New tax lot at reinvestment price

- Foreign dividends with withholding tax: Claim DTA relief

- Accumulation funds: No dividend distribution, increases share price

- Offshore bonds: Complex tax treatment (top-slicing relief)

- EIS loss relief: Can claim against income tax if EIS shares sold at loss

- VCT dividend: Tax-free regardless of investor's tax band

- Fractional shares: Support decimal quantities for ETFs/funds

- Currency hedged funds: Track hedge separately from underlying

**Performance Considerations:**

* Price updates: Batch process, cache for 15 minutes (real-time)
* Portfolio calculation: Materialized view, refresh on material changes
* CGT calculation: Complex, cache per holding, recalculate on disposal
* Asset allocation: Pre-calculate, store in snapshot
* Performance metrics: Time-intensive, calculate daily (background job)
* Expected holdings per user: 10-100
* Portfolio summary: Target <1 second response
* Historical performance: Pre-aggregate monthly/quarterly
* Dividend income report: Index on payment\_date
* Tax lot queries: Optimize FIFO lookups with proper indexing
* Bulk import: Process asynchronously, max 1000 holdings per import
* Market data API: Rate limiting, caching, fallback to manual updates

**7. RETIREMENT MODULE**

**Feature 7.1: UK Pension Management**

**Feature Name:** Comprehensive UK Pension Tracking and Planning

**User Story:** As a UK pension saver, I want to track all my pension schemes including workplace pensions, personal pensions, SIPPs, and state pension entitlement so that I can plan for retirement and optimize my contributions.

**Acceptance Criteria:**

* Track multiple UK pension schemes (Occupational DB/DC, Personal Pensions, SIPP)
* Record contributions (employee, employer, personal)
* Track pension pot values (current and projected)
* Annual Allowance tracking and carry forward
* Lifetime Allowance tracking (historical and current rules)
* State Pension forecast integration
* Pension Commencement Lump Sum (PCLS) projection
* Retirement income modeling
* Tax relief tracking (net pay vs relief at source)
* Pension freedom options analysis (drawdown vs annuity)

**Technical Requirements:**

* Defined Benefit (DB) pension valuation
* Defined Contribution (DC) projection modeling
* Annual Allowance calculator with taper for high earners
* Lifetime Allowance calculator (noting 2023 changes)
* State Pension API integration (HMRC Gateway - future)
* Retirement income tax calculator
* Inflation adjustment for projections
* Investment return modeling (Monte Carlo optional)

**Constraints:**

* Annual Allowance (2024/25): £60,000
* Money Purchase Annual Allowance (MPAA): £10,000 (if triggered)
* Tapered Annual Allowance: Reduced for adjusted income >£260,000
* Lifetime Allowance: Abolished April 2024, but lump sum limits apply
* Lump Sum Allowance: £268,275
* Lump Sum and Death Benefit Allowance: £1,073,100
* State Pension Age: Dynamic based on DOB
* Minimum pension access age: 55 (rising to 57 in 2028)

**Implementation Approach:**

ENDPOINT: POST /api/v1/retirement/uk-pensions

REQUEST BODY:

{

pensionType: enum['OCCUPATIONAL\_DB', 'OCCUPATIONAL\_DC', 'PERSONAL\_PENSION',

'SIPP', 'STAKEHOLDER', 'GROUP\_PERSONAL\_PENSION'],

provider: string,

schemeReference: string,

employerName: string (for occupational),

// For Defined Contribution

currentValue: decimal,

contributionDetails: {

employeeContribution: {

amount: decimal,

frequency: enum['MONTHLY', 'ANNUALLY'],

type: enum['PERCENTAGE\_SALARY', 'FIXED\_AMOUNT']

},

employerContribution: {

amount: decimal,

frequency: enum['MONTHLY', 'ANNUALLY'],

type: enum['PERCENTAGE\_SALARY', 'FIXED\_AMOUNT']

},

personalContribution: {

amount: decimal,

frequency: enum['MONTHLY', 'ANNUALLY']

},

taxReliefMethod: enum['NET\_PAY', 'RELIEF\_AT\_SOURCE']

},

// For Defined Benefit

definedBenefitDetails: {

accrualRate: decimal (e.g., 1/60, 1/80),

pensionableService: decimal (years),

finalSalary: decimal (or CARE: career average),

schemeType: enum['FINAL\_SALARY', 'CAREER\_AVERAGE', 'CASH\_BALANCE'],

normalRetirementAge: integer,

guaranteedPension: decimal (annual amount at NRA),

spousePension: decimal (percentage, e.g., 50%),

lumpSumEntitlement: decimal (if applicable),

indexation: enum['CPI', 'RPI', 'FIXED\_PERCENTAGE', 'NONE']

},

startDate: date,

expectedRetirementDate: date,

investmentStrategy: enum['CONSERVATIVE', 'BALANCED', 'AGGRESSIVE', 'CUSTOM'],

assumedGrowthRate: decimal (annual percentage),

assumedInflationRate: decimal,

mpaaTriggered: boolean (money purchase annual allowance),

mpaaDate: date (when triggered)

}

BUSINESS LOGIC:

1. Validate pension data:

- Current value >= 0

- Contributions >= 0

- Expected retirement date > today

- For DB: Validate accrual rate and service years

2. Calculate Annual Allowance usage:

total\_annual\_contribution = employee\_contribution + employer\_contribution +

personal\_contribution + tax\_relief

IF mpaaTriggered:

annual\_allowance = 10000 // MPAA

ELSE:

base\_allowance = 60000

// Taper for high earners

IF adjusted\_income > 260000:

reduction = MIN((adjusted\_income - 260000) / 2, 56000)

annual\_allowance = MAX(base\_allowance - reduction, 10000)

ELSE:

annual\_allowance = base\_allowance

allowance\_used = total\_annual\_contribution

allowance\_remaining = annual\_allowance - allowance\_used

// Carry forward unused allowance (previous 3 years)

carry\_forward\_available = calculate\_carry\_forward()

3. Calculate Lifetime Allowance impact (for pre-2024 protections):

// Post-April 2024: No LTA charge, but lump sum limits apply

lump\_sum\_allowance = 268275 // 25% of old LTA

lump\_sum\_death\_benefit\_allowance = 1073100

projected\_pension\_value = calculate\_projection()

available\_tax\_free\_cash = MIN(projected\_pension\_value \* 0.25, lump\_sum\_allowance)

4. Project future pension pot (DC):

FOR year IN range(current\_year, retirement\_year):

annual\_contribution = calculate\_contributions(year)

investment\_return = pot\_value \* growth\_rate

pot\_value = pot\_value + annual\_contribution + investment\_return

projected\_pot\_at\_retirement = pot\_value

5. Calculate DB pension value:

// For Annual Allowance purposes

pension\_input\_amount = (pension\_accrued\_this\_year \* 16) +

(lump\_sum\_accrued\_this\_year)

// Current value for net worth

transfer\_value\_equivalent = request\_from\_scheme\_or\_estimate()

6. Retirement income projection:

// Drawdown scenario

sustainable\_withdrawal = projected\_pot \* safe\_withdrawal\_rate // e.g., 4%

// Annuity scenario (estimate)

annuity\_rate = get\_current\_annuity\_rates(age, health\_status)

annuity\_income = projected\_pot \* annuity\_rate

// DB pension income

db\_annual\_income = pensionable\_service \* accrual\_rate \* final\_salary

7. Tax relief calculation:

IF tax\_relief\_method = 'NET\_PAY':

tax\_relief = contribution \* user.marginal\_tax\_rate

ELSE IF tax\_relief\_method = 'RELIEF\_AT\_SOURCE':

basic\_rate\_relief = contribution \* 0.20 // Automatic

higher\_rate\_relief = contribution \* (user.marginal\_tax\_rate - 0.20) // Claim via tax return

8. State Pension integration:

state\_pension\_amount = fetch\_or\_estimate\_state\_pension()

state\_pension\_age = calculate\_spa(user.date\_of\_birth)

RESPONSE:

{

id: uuid,

pensionDetails: {...},

currentStatus: {

currentValue: decimal (DC) or transferValue: decimal (DB),

totalContributions: decimal,

investmentReturns: decimal

},

annualAllowance: {

taxYear: string,

allowance: decimal,

used: decimal,

remaining: decimal,

carryForwardAvailable: decimal,

mpaaApplies: boolean

},

projection: {

projectedValueAtRetirement: decimal,

retirementAge: integer,

yearsToRetirement: integer,

assumptions: {

growthRate: decimal,

inflationRate: decimal,

contributions: string

}

},

retirementIncome: {

taxFreeCash: decimal,

annuityOption: {

estimatedAnnualIncome: decimal,

assumptions: string

},

drawdownOption: {

sustainableAnnualIncome: decimal,

withdrawalRate: decimal

},

dbPensionIncome: decimal (if applicable)

},

taxImplications: {

taxReliefReceived: decimal,

taxOnRetirementIncome: decimal (estimated),

lumpSumAllowanceUsed: decimal,

lumpSumAllowanceRemaining: decimal

},

recommendations: [

{message: string, estimatedBenefit: decimal}

]

}

**User Flow:**

[Retirement Dashboard] → [UK Pensions Tab]

↓

[Pension Summary]

- Total pension pot value (all schemes combined)

- Projected retirement income

- Annual Allowance usage this year

- Years to retirement countdown

↓

[Add Pension Button]

↓

[Pension Entry - Step 1: Type]

- Select pension type (visual cards):

- Workplace Pension (DB or DC)

- Personal Pension / SIPP

- State Pension (tracked separately)

↓

[Pension Entry - Step 2: Provider & Scheme]

- Provider name

- Scheme reference

- Employer (if workplace)

- Start date

↓

[Pension Entry - Step 3: Contributions] (DC)

- Current pot value

- Your contribution (£ or %)

- Employer contribution (£ or %)

- Personal contributions

- Tax relief method

↓

[Pension Entry - Step 3: Benefit Details] (DB)

- Accrual rate

- Pensionable service

- Final/Career average salary

- Normal retirement age

- Guaranteed annual pension

↓

[Pension Entry - Step 4: Projections]

- Expected retirement age

- Investment strategy (DC)

- Assumed growth rate

- Inflation assumption

↓

[Projection Display]

- Pot value at retirement

- Tax-free cash available

- Income options

- Visual timeline chart

↓

[Annual Allowance Check]

- This year's usage

- Carry forward available

- Warnings if exceeding

↓

[Save Pension]

↓

[Pension List View]

- Card view: Each pension with key details

- Total retirement income projection

- Filter: By type, employer

- Sort: By value, contribution

↓

[State Pension Section]

- NI record summary

- Forecast amount

- State Pension Age

- Voluntary contribution calculator

**API Endpoints:**

* POST /api/v1/retirement/uk-pensions - Add pension
* PUT /api/v1/retirement/uk-pensions/{id} - Update pension
* DELETE /api/v1/retirement/uk-pensions/{id} - Delete pension (soft delete)
* GET /api/v1/retirement/uk-pensions - List all pensions
* GET /api/v1/retirement/uk-pensions/{id} - Get specific pension
* POST /api/v1/retirement/uk-pensions/{id}/update-value - Update pot value
* GET /api/v1/retirement/annual-allowance - Annual Allowance summary
* POST /api/v1/retirement/annual-allowance/calculate - Calculate AA with carry forward
* GET /api/v1/retirement/projection - Combined retirement projection
* POST /api/v1/retirement/retirement-income-calculator - Model retirement income
* GET /api/v1/retirement/state-pension - State Pension details
* POST /api/v1/retirement/state-pension/voluntary-contributions - Calculate voluntary NI

**Data Models:**

TABLE: uk\_pensions

- id: UUID (PK)

- user\_id: UUID (FK to users)

- pension\_type: ENUM('OCCUPATIONAL\_DB', 'OCCUPATIONAL\_DC', 'PERSONAL\_PENSION',

'SIPP', 'STAKEHOLDER', 'GROUP\_PERSONAL\_PENSION')

- provider: VARCHAR(255)

- scheme\_reference: VARCHAR(100)

- employer\_name: VARCHAR(255)

- start\_date: DATE

- expected\_retirement\_date: DATE

- normal\_retirement\_age: INTEGER

- status: ENUM('ACTIVE', 'DEFERRED', 'IN\_PAYMENT', 'TRANSFERRED\_OUT')

- mpaa\_triggered: BOOLEAN DEFAULT FALSE

- mpaa\_trigger\_date: DATE

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: dc\_pension\_details

- pension\_id: UUID (PK, FK to uk\_pensions)

- current\_value: DECIMAL(15,2)

- employee\_contribution\_amount: DECIMAL(10,2)

- employee\_contribution\_frequency: ENUM('MONTHLY', 'ANNUALLY')

- employee\_contribution\_type: ENUM('PERCENTAGE\_SALARY', 'FIXED\_AMOUNT')

- employee\_contribution\_percentage: DECIMAL(5,2)

- employer\_contribution\_amount: DECIMAL(10,2)

- employer\_contribution\_frequency: ENUM('MONTHLY', 'ANNUALLY')

- employer\_contribution\_type: ENUM('PERCENTAGE\_SALARY', 'FIXED\_AMOUNT')

- employer\_contribution\_percentage: DECIMAL(5,2)

- personal\_contribution\_amount: DECIMAL(10,2)

- personal\_contribution\_frequency: ENUM('MONTHLY', 'ANNUALLY')

- tax\_relief\_method: ENUM('NET\_PAY', 'RELIEF\_AT\_SOURCE')

- investment\_strategy: ENUM('CONSERVATIVE', 'BALANCED', 'AGGRESSIVE', 'CUSTOM')

- assumed\_growth\_rate: DECIMAL(5,2)

- assumed\_inflation\_rate: DECIMAL(5,2)

- last\_value\_update: DATE

- updated\_at: TIMESTAMP

TABLE: db\_pension\_details

- pension\_id: UUID (PK, FK to uk\_pensions)

- accrual\_rate: VARCHAR(10) (e.g., '1/60', '1/80')

- pensionable\_service\_years: DECIMAL(5,2)

- scheme\_type: ENUM('FINAL\_SALARY', 'CAREER\_AVERAGE', 'CASH\_BALANCE')

- final\_salary: DECIMAL(10,2) (or career average)

- guaranteed\_annual\_pension: DECIMAL(10,2)

- spouse\_pension\_percentage: DECIMAL(5,2)

- lump\_sum\_entitlement: DECIMAL(10,2)

- indexation\_type: ENUM('CPI', 'RPI', 'FIXED\_PERCENTAGE', 'NONE')

- indexation\_rate: DECIMAL(5,2) (if fixed)

- transfer\_value\_equivalent: DECIMAL(15,2) (CETV)

- cetv\_date: DATE

- updated\_at: TIMESTAMP

TABLE: pension\_value\_history

- id: UUID (PK)

- pension\_id: UUID (FK to uk\_pensions)

- value\_date: DATE

- pension\_value: DECIMAL(15,2)

- contributions\_this\_period: DECIMAL(10,2)

- investment\_return: DECIMAL(10,2)

- value\_source: ENUM('MANUAL', 'PROVIDER\_STATEMENT', 'PROVIDER\_API')

- created\_at: TIMESTAMP

TABLE: annual\_allowance\_tracking

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: VARCHAR(7)

- annual\_allowance: DECIMAL(10,2) (60000 or tapered)

- total\_contributions: DECIMAL(10,2)

- allowance\_used: DECIMAL(10,2)

- allowance\_remaining: DECIMAL(10,2)

- carry\_forward\_year\_1: DECIMAL(10,2) (previous year unused)

- carry\_forward\_year\_2: DECIMAL(10,2)

- carry\_forward\_year\_3: DECIMAL(10,2)

- total\_carry\_forward\_available: DECIMAL(10,2)

- excess\_contributions: DECIMAL(10,2) (if over limit)

- annual\_allowance\_charge: DECIMAL(10,2)

- adjusted\_income: DECIMAL(15,2) (for taper calculation)

- threshold\_income: DECIMAL(15,2) (for taper calculation)

- mpaa\_applies: BOOLEAN

- created\_at: TIMESTAMP

TABLE: pension\_contributions

- id: UUID (PK)

- pension\_id: UUID (FK to uk\_pensions)

- contribution\_date: DATE

- employee\_contribution: DECIMAL(10,2)

- employer\_contribution: DECIMAL(10,2)

- personal\_contribution: DECIMAL(10,2)

- tax\_relief\_received: DECIMAL(10,2)

- total\_contribution: DECIMAL(10,2)

- tax\_year: VARCHAR(7)

- created\_at: TIMESTAMP

TABLE: state\_pension

- user\_id: UUID (PK, FK to users)

- ni\_number: VARCHAR(13) ENCRYPTED

- qualifying\_years: INTEGER

- years\_needed\_for\_full\_pension: INTEGER (typically 35)

- forecast\_weekly\_amount: DECIMAL(10,2)

- forecast\_annual\_amount: DECIMAL(10,2)

- state\_pension\_age: INTEGER

- state\_pension\_date: DATE (calculated)

- gaps\_in\_record: INTEGER

- voluntary\_contribution\_cost: DECIMAL(10,2) (to fill gaps)

- last\_updated: DATE

- data\_source: ENUM('HMRC\_API', 'MANUAL', 'ESTIMATED')

TABLE: retirement\_projections

- id: UUID (PK)

- user\_id: UUID (FK to users)

- projection\_date: DATE

- retirement\_age: INTEGER

- total\_pension\_pot\_dc: DECIMAL(15,2)

- total\_db\_annual\_income: DECIMAL(15,2)

- state\_pension\_annual: DECIMAL(15,2)

- tax\_free\_lump\_sum: DECIMAL(15,2)

- annuity\_income\_estimate: DECIMAL(15,2)

- drawdown\_income\_estimate: DECIMAL(15,2)

- total\_retirement\_income: DECIMAL(15,2)

- income\_after\_tax: DECIMAL(15,2)

- assumptions\_json: JSON

- created\_at: TIMESTAMP

INDEX on uk\_pensions(user\_id, status)

INDEX on pension\_value\_history(pension\_id, value\_date DESC)

INDEX on annual\_allowance\_tracking(user\_id, tax\_year)

INDEX on pension\_contributions(pension\_id, tax\_year)

**Error Handling:**

ERROR CASES:

1. Contributions exceed Annual Allowance

- Response: 400 Bad Request (warning)

- Message: "Your contributions of £{amount} exceed your Annual Allowance of £{allowance}"

- Details: "This may result in an Annual Allowance charge of £{charge}"

- Allow user to proceed with acknowledgment

2. MPAA triggered but high contributions entered

- Response: 400 Bad Request

- Message: "Money Purchase Annual Allowance (£10,000) applies. Your contributions of £{amount} exceed this"

3. Retirement age below minimum (55, soon 57)

- Response: 400 Bad Request

- Message: "Minimum pension access age is {age}. You've entered {entered\_age}"

4. DB accrual rate invalid

- Response: 400 Bad Request

- Message: "Accrual rate must be in format like '1/60' or '1/80'"

5. Negative pension value

- Response: 400 Bad Request

- Message: "Pension value cannot be negative"

6. Transfer value date too old (>3 months for DB)

- Response: 400 Bad Request (warning)

- Message: "CETV dated {date} is more than 3 months old. Consider requesting updated valuation"

7. State Pension forecast: insufficient NI years

- Response: 200 OK with warning

- Message: "You have {years} qualifying years. You need {required} for full State Pension"

- Recommendation: "Consider voluntary NI contributions (cost: £{amount})"

EDGE CASES:

- Multiple pensions from same employer: Allow, track separately

- Pension transfer in: Create new pension, close old one, link transfer

- Pension transfer out: Mark status as 'TRANSFERRED\_OUT', retain history

- DB to DC transfer: Complex, requires CETV and conversion

- Pension recycling rules: If take TFC and reinvest, may trigger MPAA

- Protected pension age (e.g., 50 for police): Override minimum age check

- LTA protection (Fixed Protection, Individual Protection): Track protection type

- Small pots rule: Can take 3 pensions under £10k without triggering MPAA

- Trivial commutation: Full pension as lump sum if total < £30k

- Flexible drawdown: Track if taking income (triggers MPAA for other pensions)

- Death before retirement: Nominee drawdown or lump sum (usually tax-free if under 75)

- Scheme Pays: Employer pays AA charge, reduces pension

- Salary sacrifice: Reduces net pay, increases employer contribution

- Auto-enrolment: Minimum contributions (8% total: 3% employer, 5% employee)

- Career Average Revalued Earnings (CARE): Annual accrual revalued by inflation

- Pension commencement: Options at 75 (no BCE, but rules change)

**Performance Considerations:**

* Projection calculations: Complex, cache for 24 hours
* Annual Allowance calc: Pre-calculate at each contribution, cache current tax year
* State Pension API: Rate limited, cache for 30 days
* Pension value updates: Manual or via provider API (daily batch)
* Expected pensions per user: 2-8
* Projection query: Target <1 second
* Retirement income calculator: Multiple scenarios, may take 2-3 seconds
* Historical value tracking: Index on date for trend queries
* Carry forward calculation: Requires 3 previous tax years, pre-aggregate
* DB pension valuation: Complex actuarial calc, use cached CETV

**Feature 7.2: SA Retirement Fund Management**

**Feature Name:** South African Retirement and Preservation Fund Tracking

**User Story:** As a South African resident or someone with SA retirement funds, I want to track my pension funds, provident funds, retirement annuities, and preservation funds so I can plan for retirement and manage my contributions.

**Acceptance Criteria:**

* Track Pension Funds, Provident Funds, Retirement Annuities (RA)
* Track Preservation Funds (pension and provident)
* Record contributions and employer contributions
* Section 10C tax deduction tracking (27.5% of income, max R350,000)
* Fund value and investment choice tracking
* Retirement age options (55+ for most funds)
* One-third lump sum option tracking
* Two-thirds annuity requirement (pension funds)
* Full cash withdrawal option (provident funds, with limits)
* Regulation 28 compliance monitoring
* Withdrawal rules and penalties

**Technical Requirements:**

* SA tax calculation for retirement contributions
* Section 10C deduction calculator
* Lump sum tax tables (2024: R550,000 tax-free, then progressive)
* Annuity income tax calculation
* Regulation 28 compliance checker (max 75% equity, max 30% offshore)
* Integration with SA income tax module

**Constraints:**

* Section 10C deduction: 27.5% of remuneration or taxable income, capped at R350,000 per tax year
* Retirement age: Typically 55 minimum
* Pension fund: Min 2/3 to annuity, max 1/3 lump sum
* Provident fund: Full withdrawal option (subject to limits for recent contributions)
* Preservation fund: One withdrawal allowed before retirement
* Tax-free lump sum (2024): R550,000 (first R550k tax-free, then tiered rates)

**Implementation Approach:**

ENDPOINT: POST /api/v1/retirement/sa-retirement-funds

REQUEST BODY:

{

fundType: enum['PENSION\_FUND', 'PROVIDENT\_FUND', 'RETIREMENT\_ANNUITY',

'PRESERVATION\_FUND\_PENSION', 'PRESERVATION\_FUND\_PROVIDENT'],

provider: string,

fundName: string,

employerName: string (for pension/provident),

memberNumber: string,

currentValue: decimal,

currency: string (typically ZAR),

contributionDetails: {

employeeContribution: {

amount: decimal,

frequency: enum['MONTHLY', 'ANNUALLY'],

percentage: decimal (if percentage of salary)

},

employerContribution: {

amount: decimal,

frequency: enum['MONTHLY', 'ANNUALLY']

}

},

investmentChoice: {

portfolioName: string,

assetAllocation: {

equityPercentage: decimal,

bondsPercentage: decimal,

cashPercentage: decimal,

propertyPercentage: decimal,

offshorePercentage: decimal

},

regulation28Compliant: boolean

},

startDate: date,

expectedRetirementAge: integer,

normalRetirementAge: integer (fund specific),

// For preservation funds

preservationFundSource: enum['PENSION\_FUND', 'PROVIDENT\_FUND'],

transferDate: date,

withdrawalTaken: boolean,

withdrawalDate: date,

withdrawalAmount: decimal,

// For provident funds (new rules)

contributionsBeforeMarch2021: decimal,

contributionsAfterMarch2021: decimal

}

BUSINESS LOGIC:

1. Validate fund data:

- Current value >= 0

- Retirement age >= 55 (typically)

- Contributions >= 0

2. Calculate Section 10C tax deduction:

annual\_contribution = employee\_contribution \* frequency\_multiplier

// 27.5% of the greater of remuneration or taxable income

max\_deductible = MIN(user.annual\_income \* 0.275, 350000)

deduction\_claimed = MIN(annual\_contribution, max\_deductible)

tax\_saving = deduction\_claimed \* user.marginal\_tax\_rate

3. Check Regulation 28 compliance:

IF fundType IN ['PENSION\_FUND', 'PROVIDENT\_FUND', 'RETIREMENT\_ANNUITY']:

reg28\_equity\_limit = 75

reg28\_offshore\_limit = 30

reg28\_property\_limit = 25

IF equity\_percentage > reg28\_equity\_limit:

non\_compliant = TRUE

violation = "Equity allocation exceeds 75%"

IF offshore\_percentage > reg28\_offshore\_limit:

non\_compliant = TRUE

violation = "Offshore allocation exceeds 30%"

4. Calculate retirement options:

// Pension Fund and RA

IF fundType IN ['PENSION\_FUND', 'RETIREMENT\_ANNUITY']:

max\_lump\_sum = fund\_value \* (1/3)

min\_annuity = fund\_value \* (2/3)

// Unless fund value < R247,500 (trivial amount)

IF fund\_value < 247500:

can\_take\_full\_cash = TRUE

// Provident Fund (complex post-2021 rules)

IF fundType = 'PROVIDENT\_FUND':

// Contributions before 1 March 2021: Full cash withdrawal

// Contributions after 1 March 2021: Subject to 1/3 rule

cash\_portion = contributionsBeforeMarch2021 +

(contributionsAfterMarch2021 \* (1/3))

annuity\_portion = contributionsAfterMarch2021 \* (2/3)

5. Calculate lump sum tax (SA 2024 tax tables):

tax\_free\_portion = 550000 // First R550k tax-free

IF lump\_sum <= tax\_free\_portion:

tax = 0

ELSE:

taxable\_amount = lump\_sum - tax\_free\_portion

// Progressive rates

IF taxable\_amount <= 200000:

tax = taxable\_amount \* 0.18

ELSE IF taxable\_amount <= 350000:

tax = 36000 + (taxable\_amount - 200000) \* 0.27

ELSE IF taxable\_amount <= 700000:

tax = 76500 + (taxable\_amount - 350000) \* 0.36

ELSE:

tax = 202500 + (taxable\_amount - 700000) \* 0.45

6. Project fund value at retirement:

years\_to\_retirement = retirement\_age - current\_age

annual\_contribution = calculate\_annual\_contributions()

assumed\_growth\_rate = 0.08 // 8% default

projected\_value = calculate\_future\_value(

current\_value,

annual\_contribution,

assumed\_growth\_rate,

years\_to\_retirement

)

7. Calculate retirement income:

// Annuity options (life annuity vs living annuity)

life\_annuity\_rate = get\_annuity\_rates(age, gender) // Typically 8-12% at 65

life\_annuity\_income = annuity\_portion \* life\_annuity\_rate

living\_annuity\_withdrawal\_min = 2.5 // 2.5% minimum

living\_annuity\_withdrawal\_max = 17.5 // 17.5% maximum

living\_annuity\_income\_range = {

min: annuity\_portion \* 0.025,

max: annuity\_portion \* 0.175

}

RESPONSE:

{

id: uuid,

fundDetails: {...},

currentStatus: {

currentValue: decimal,

currentValueZAR: decimal,

totalContributions: decimal,

investmentReturns: decimal

},

taxDeduction: {

taxYear: string,

annualContribution: decimal,

maxDeductible: decimal,

deductionClaimed: decimal,

taxSaving: decimal,

remainingAllowance: decimal

},

regulation28: {

compliant: boolean,

violations: [string],

currentAllocation: {...},

limits: {...}

},

projection: {

projectedValueAtRetirement: decimal,

retirementAge: integer,

yearsToRetirement: integer,

assumptions: {

growthRate: decimal,

contributions: string

}

},

retirementOptions: {

lumpSumOption: {

maxLumpSum: decimal,

estimatedTax: decimal,

netLumpSum: decimal

},

annuityRequirement: {

minAnnuityAmount: decimal,

canTakeFullCash: boolean (if trivial)

},

incomeProjection: {

lifeAnnuity: {

estimatedMonthlyIncome: decimal,

assumptions: string

},

livingAnnuity: {

minMonthlyIncome: decimal,

maxMonthlyIncome: decimal,

recommendedRate: decimal

}

}

}

}

**User Flow:**

[Retirement Dashboard] → [SA Retirement Funds Tab]

↓

[Fund Summary]

- Total retirement savings (all SA funds)

- Section 10C deduction used this year

- Projected retirement income

- Regulation 28 compliance status

↓

[Add Fund Button]

↓

[Fund Entry - Step 1: Type]

- Select fund type (visual cards):

- Pension Fund

- Provident Fund

- Retirement Annuity (RA)

- Preservation Fund

↓

[Fund Entry - Step 2: Provider & Details]

- Provider name

- Fund name

- Member number

- Employer (if applicable)

- Start date

↓

[Fund Entry - Step 3: Value & Contributions]

- Current fund value

- Your contribution (R or %)

- Employer contribution

- Contribution frequency

↓

[Fund Entry - Step 4: Investment Choice]

- Portfolio name

- Asset allocation sliders:

- Equity %

- Bonds %

- Cash %

- Property %

- Offshore %

- Regulation 28 compliance checker (real-time)

↓

[Fund Entry - Step 5: Retirement Planning]

- Expected retirement age

- Growth rate assumption

↓

[Tax Deduction Preview]

- Annual contribution total

- Section 10C deduction

- Tax saving estimate

- Remaining deduction allowance

↓

[Projection Display]

- Fund value at retirement

- Lump sum options

- Annuity income options

↓

[Save Fund]

↓

[Fund List View]

- Card view: Each fund with key details

- Total Section 10C deduction tracker

- Regulation 28 compliance indicators

- Filter: By type, employer

- Sort: By value, contribution

**API Endpoints:**

* POST /api/v1/retirement/sa-retirement-funds - Add fund
* PUT /api/v1/retirement/sa-retirement-funds/{id} - Update fund
* DELETE /api/v1/retirement/sa-retirement-funds/{id} - Delete fund (soft delete)
* GET /api/v1/retirement/sa-retirement-funds - List all funds
* GET /api/v1/retirement/sa-retirement-funds/{id} - Get specific fund
* POST /api/v1/retirement/sa-retirement-funds/{id}/update-value - Update value
* GET /api/v1/retirement/sa-section-10c - Section 10C deduction summary
* POST /api/v1/retirement/sa-lump-sum-tax-calculator - Calculate lump sum tax
* POST /api/v1/retirement/sa-regulation-28-checker - Check Reg 28 compliance
* GET /api/v1/retirement/sa-retirement-income-calculator - Model retirement income

**Data Models:**

TABLE: sa\_retirement\_funds

- id: UUID (PK)

- user\_id: UUID (FK to users)

- fund\_type: ENUM('PENSION\_FUND', 'PROVIDENT\_FUND', 'RETIREMENT\_ANNUITY',

'PRESERVATION\_FUND\_PENSION', 'PRESERVATION\_FUND\_PROVIDENT')

- provider: VARCHAR(255)

- fund\_name: VARCHAR(255)

- employer\_name: VARCHAR(255)

- member\_number\_encrypted: VARCHAR(255)

- current\_value: DECIMAL(15,2)

- currency: CHAR(3) DEFAULT 'ZAR'

- start\_date: DATE

- expected\_retirement\_age: INTEGER

- normal\_retirement\_age: INTEGER

- status: ENUM('ACTIVE', 'PRESERVED', 'PAID\_OUT', 'TRANSFERRED')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: sa\_fund\_contributions

- fund\_id: UUID (PK, FK to sa\_retirement\_funds)

- employee\_contribution\_amount: DECIMAL(10,2)

- employee\_contribution\_frequency: ENUM('MONTHLY', 'ANNUALLY')

- employee\_contribution\_percentage: DECIMAL(5,2)

- employer\_contribution\_amount: DECIMAL(10,2)

- employer\_contribution\_frequency: ENUM('MONTHLY', 'ANNUALLY')

- contributions\_before\_march\_2021: DECIMAL(15,2) (for provident funds)

- contributions\_after\_march\_2021: DECIMAL(15,2) (for provident funds)

- updated\_at: TIMESTAMP

TABLE: sa\_fund\_investment\_choice

- fund\_id: UUID (PK, FK to sa\_retirement\_funds)

- portfolio\_name: VARCHAR(255)

- equity\_percentage: DECIMAL(5,2)

- bonds\_percentage: DECIMAL(5,2)

- cash\_percentage: DECIMAL(5,2)

- property\_percentage: DECIMAL(5,2)

- offshore\_percentage: DECIMAL(5,2)

- regulation\_28\_compliant: BOOLEAN

- last\_rebalance\_date: DATE

- updated\_at: TIMESTAMP

TABLE: sa\_preservation\_fund\_details

- fund\_id: UUID (PK, FK to sa\_retirement\_funds)

- source\_fund\_type: ENUM('PENSION\_FUND', 'PROVIDENT\_FUND')

- transfer\_date: DATE

- transfer\_value: DECIMAL(15,2)

- withdrawal\_allowed: BOOLEAN DEFAULT TRUE

- withdrawal\_taken: BOOLEAN DEFAULT FALSE

- withdrawal\_date: DATE

- withdrawal\_amount: DECIMAL(15,2)

- updated\_at: TIMESTAMP

TABLE: sa\_section\_10c\_tracking

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: VARCHAR(9) (e.g., '2024/2025')

- total\_contributions: DECIMAL(15,2)

- remuneration: DECIMAL(15,2)

- taxable\_income: DECIMAL(15,2)

- max\_deductible: DECIMAL(15,2)

- deduction\_claimed: DECIMAL(15,2)

- tax\_saving: DECIMAL(15,2)

- created\_at: TIMESTAMP

TABLE: sa\_fund\_value\_history

- id: UUID (PK)

- fund\_id: UUID (FK to sa\_retirement\_funds)

- value\_date: DATE

- fund\_value: DECIMAL(15,2)

- contributions\_this\_period: DECIMAL(10,2)

- investment\_return: DECIMAL(10,2)

- created\_at: TIMESTAMP

TABLE: sa\_retirement\_projections

- id: UUID (PK)

- user\_id: UUID (FK to users)

- projection\_date: DATE

- retirement\_age: INTEGER

- total\_fund\_value: DECIMAL(15,2)

- lump\_sum\_option: DECIMAL(15,2)

- lump\_sum\_tax: DECIMAL(15,2)

- annuity\_portion: DECIMAL(15,2)

- life\_annuity\_income: DECIMAL(15,2)

- living\_annuity\_income\_min: DECIMAL(15,2)

- living\_annuity\_income\_max: DECIMAL(15,2)

- assumptions\_json: JSON

- created\_at: TIMESTAMP

INDEX on sa\_retirement\_funds(user\_id, status)

INDEX on sa\_section\_10c\_tracking(user\_id, tax\_year)

INDEX on sa\_fund\_value\_history(fund\_id, value\_date DESC)

**Error Handling:**

ERROR CASES:

1. Contributions exceed Section 10C limit

- Response: 400 Bad Request (warning)

- Message: "Your contributions of R{amount} exceed the deductible limit"

- Details: "Max deductible: R{max}. Excess: R{excess} (no tax benefit)"

2. Regulation 28 violation

- Response: 400 Bad Request (warning)

- Message: "Asset allocation violates Regulation 28"

- Details: "Equity: {equity}% (max 75%), Offshore: {offshore}% (max 30%)"

- Allow user to save with acknowledgment

3. Preservation fund second withdrawal attempt

- Response: 400 Bad Request

- Message: "Only one withdrawal allowed from preservation funds before retirement"

- Details: "Previous withdrawal: R{amount} on {date}"

4. Retirement age below minimum

- Response: 400 Bad Request

- Message: "Minimum retirement age for {fund\_type} is 55"

5. Negative fund value

- Response: 400 Bad Request

- Message: "Fund value cannot be negative"

6. Asset allocation doesn't total 100%

- Response: 400 Bad Request

- Message: "Asset allocation must total 100%. Current total: {total}%"

EDGE CASES:

- Provident fund to preservation fund: Track original contributions split (pre/post March 2021)

- Multiple employers: Multiple pension funds, track separately

- Fund switch within RA: Update investment choice, no tax event

- Emigration (financial): Tax implications on withdrawal (complex)

- Retrenchment/resignation: Options to preserve or withdraw

- Early retirement (50-55): May be allowed by fund rules, penalties may apply

- Disability retirement: May have different tax treatment

- Death before retirement: Lump sum to beneficiaries (usually tax-free)

- Divorce: Pension interest awarded to ex-spouse (no tax if clean break)

- Retirement reform: Provident fund rules changed March 2021

- Living annuity: Can change income drawdown rate annually

- Guaranteed life annuity: Fixed income for life, no flexibility

- With-profit annuity: Potential for income increases

- Living annuity depletion: Risk of running out if withdrawal rate too high

- Offshore pension to SA: Tax implications, may need SARS approval

**Performance Considerations:**

* Section 10C calculation: Pre-calculate for current tax year, cache
* Regulation 28 checking: Real-time validation, simple calculation
* Lump sum tax calculation: Use lookup table for tax brackets
* Fund value updates: Manual or via provider statement (monthly/quarterly)
* Expected funds per user: 1-4
* Projection query: Target <1 second
* Retirement income calculator: Multiple scenarios, <2 seconds
* Historical value tracking: Index on date
* Asset allocation visualization: Pre-calculate percentages

**Feature 7.3: QROPS / ROPS Management**

**Feature Name:** Cross-Border Pension Transfer Tracking (QROPS/ROPS)

**User Story:** As someone who has transferred or is considering transferring a UK pension overseas, I want to track QROPS (Qualifying Recognised Overseas Pension Scheme) or ROPS (Recognised Overseas Pension Scheme in SA) so I can manage cross-border pension arrangements and understand tax implications.

**Acceptance Criteria:**

* Track QROPS/ROPS pension schemes
* Record UK pension transfer details (source scheme, transfer value, date)
* Track overseas tax charges (25% if not EEA and resident <5 years)
* Monitor reporting requirements to HMRC
* Track contributions post-transfer (if allowed)
* Dual-country tax treatment
* Currency exposure management
* Retirement benefit options (based on receiving country rules)

**Technical Requirements:**

* UK-SA pension transfer rules engine
* Overseas Transfer Charge calculator (25% if applicable)
* HMRC reporting requirement tracker (5 years post-transfer)
* Exchange rate tracking at transfer date
* Integration with both UK and SA retirement modules
* DTA application for pension income

**Constraints:**

* Overseas Transfer Charge: 25% if transferring to non-EEA and member not resident in destination country
* HMRC reporting: Annual reporting required for 5 years post-transfer (until April 2024 rules)
* SA ROPS: Must be SARS-approved
* Minimum transfer value: Typically £30,000 (scheme rules)
* Transfer testing: Protection of benefits

**Implementation Approach:**

ENDPOINT: POST /api/v1/retirement/qrops-rops

REQUEST BODY:

{

schemeType: enum['QROPS', 'ROPS', 'OTHER\_OVERSEAS\_PENSION'],

destinationCountry: enum['SA', 'OTHER'],

schemeName: string,

schemeProvider: string,

qropsReferenceNumber: string,

schemeAddress: text,

ukSourcePension: {

ukPensionId: uuid (FK to uk\_pensions),

sourceSchemeName: string,

sourceProvider: string,

transferValue: decimal,

transferDate: date,

transferCurrency: string,

exchangeRate: decimal (GBP to destination currency)

},

overseasTransferCharge: {

applicable: boolean,

chargeRate: decimal (typically 25%),

chargeAmount: decimal,

reasonForCharge: string,

paidBy: enum['MEMBER', 'SCHEME']

},

memberResidency: {

residencyAtTransfer: enum['UK', 'SA', 'OTHER'],

taxResidencyAtTransfer: enum['UK', 'SA', 'BOTH'],

yearsOutOfUk: integer

},

currentValue: decimal,

destinationCurrency: string,

reportingRequirements: {

hmrcReportingRequired: boolean,

reportingPeriodEnd: date (transfer\_date + 10 years),

lastReportDate: date,

nextReportDue: date

},

benefitOptions: {

retirementAge: integer,

lumpSumAvailable: boolean,

lumpSumPercentage: decimal,

annuityRequired: boolean,

drawdownAvailable: boolean

},

notes: text

}

BUSINESS LOGIC:

1. Validate QROPS/ROPS data:

- Transfer date <= today

- Transfer value > 0

- Destination country valid

2. Calculate Overseas Transfer Charge:

// 25% charge applies if:

// - Transfer to non-EEA QROPS, AND

// - Member not tax resident in same country as QROPS for 5 full UK tax years before transfer

IF destinationCountry NOT IN ['EEA\_COUNTRIES']:

IF member\_tax\_resident\_in\_destination\_country < 5\_years:

overseas\_transfer\_charge\_applicable = TRUE

charge\_amount = transfer\_value \* 0.25

ELSE:

overseas\_transfer\_charge\_applicable = FALSE

// Exception: UK/SA DTA may provide relief

IF destinationCountry = 'SA' AND uk\_sa\_dta\_applies:

// Check specific DTA provisions

may\_avoid\_charge = TRUE

3. Determine reporting requirements:

// Pre-April 2024: 10 years of reporting

// Post-April 2024: 5 years for certain transfers

IF transfer\_date < '2024-04-06':

reporting\_period\_years = 10

ELSE:

reporting\_period\_years = 5

reporting\_end\_date = transfer\_date + reporting\_period\_years \* 365

// Report required events:

// - Receiving scheme losing QROPS status

// - Member becoming UK resident again

// - Payments made from scheme

// - Transfers out of scheme

4. Track currency exposure:

value\_in\_gbp = current\_value / current\_exchange\_rate

value\_in\_zar = current\_value \* current\_zar\_gbp\_rate

currency\_gain\_loss = value\_in\_gbp - (transfer\_value / transfer\_exchange\_rate)

5. Calculate retirement benefits (destination country rules):

IF destinationCountry = 'SA':

// SA ROPS follows SA retirement fund rules

apply\_sa\_retirement\_rules()

max\_lump\_sum = calculate\_sa\_lump\_sum()

annuity\_requirement = calculate\_sa\_annuity\_requirement()

ELSE:

// Other jurisdictions

apply\_destination\_country\_rules()

6. Tax treatment of pension income:

// DTA determines which country taxes pension

// Typically: Country of residence at time of payment

IF member\_tax\_resident = 'SA':

sa\_tax\_applies = TRUE

apply\_sa\_pension\_tax\_rates()

IF member\_tax\_resident = 'UK':

uk\_tax\_applies = TRUE

apply\_uk\_pension\_tax\_rates()

// DTA relief to avoid double taxation

apply\_uk\_sa\_dta()

7. Integration with UK and SA modules:

// Link to original UK pension (now transferred out)

UPDATE uk\_pensions

SET status = 'TRANSFERRED\_OUT',

transferred\_to\_scheme = qrops\_id

WHERE id = ukSourcePension.ukPensionId

// Potentially create SA retirement fund entry if ROPS

IF destinationCountry = 'SA':

CREATE sa\_retirement\_fund\_entry()

RESPONSE:

{

id: uuid,

schemeDetails: {...},

transferDetails: {

ukSourceScheme: string,

transferValue: {gbp: decimal, destinationCurrency: decimal},

transferDate: date,

exchangeRate: decimal

},

overseasTransferCharge: {

applicable: boolean,

chargeAmount: decimal,

paidDate: date,

reason: string

},

currentStatus: {

currentValue: decimal,

valueGbp: decimal,

valueZar: decimal,

currencyGainLoss: decimal

},

reportingRequirements: {

hmrcReportingRequired: boolean,

nextReportDue: date,

reportingPeriodEnds: date,

eventsToReport: [string]

},

taxTreatment: {

currentResidence: string,

pensionIncomeTaxedIn: enum['UK', 'SA', 'BOTH'],

dtaRelief: string

},

retirementProjection: {

retirementAge: integer,

projectedValue: decimal,

lumpSumOption: decimal,

annuityRequirement: decimal,

estimatedIncome: decimal

}

}

**User Flow:**

[Retirement Dashboard] → [International Pensions Tab]

↓

[QROPS/ROPS Summary]

- Overseas pension value

- Reporting status (HMRC)

- Currency exposure

- Years until reporting ends

↓

[Add QROPS/ROPS Button]

↓

[Transfer Entry - Step 1: Destination Scheme]

- Scheme name and provider

- Destination country

- QROPS reference number

- Scheme address

↓

[Transfer Entry - Step 2: UK Source Pension]

- Select from existing UK pensions OR enter manually

- Source scheme name

- Transfer value (GBP)

- Transfer date

↓

[Transfer Entry - Step 3: Transfer Charge]

- System calculates if charge applies

- "Were you tax resident in {country} for 5 years before transfer?"

- If charge applies: Amount, paid date, paid by

↓

[Transfer Entry - Step 4: Current Position]

- Current scheme value

- Currency

- Investment choices (if applicable)

↓

[Transfer Entry - Step 5: Reporting]

- HMRC reporting required? (auto-calculated)

- Reporting period

- Next report due date

↓

[Tax Treatment Display]

- Current residency

- Which country will tax pension income

- DTA provisions

↓

[Currency Exposure Analysis]

- Transfer exchange rate

- Current exchange rate

- Currency gain/loss

- Hedging recommendations

↓

[Save QROPS/ROPS]

↓

[Scheme Detail View]

- All scheme details

- Reporting checklist

- Currency tracking chart

- Link to UK source pension (now transferred out)

↓

[Reporting Section]

- Reportable events tracker

- Submission history

- Upcoming deadlines

**API Endpoints:**

* POST /api/v1/retirement/qrops-rops - Add QROPS/ROPS
* PUT /api/v1/retirement/qrops-rops/{id} - Update scheme
* DELETE /api/v1/retirement/qrops-rops/{id} - Delete scheme (soft delete)
* GET /api/v1/retirement/qrops-rops - List all schemes
* GET /api/v1/retirement/qrops-rops/{id} - Get specific scheme
* POST /api/v1/retirement/qrops-rops/{id}/update-value - Update value
* POST /api/v1/retirement/qrops-rops/transfer-charge-calculator - Calculate OTC
* POST /api/v1/retirement/qrops-rops/{id}/report-event - Log reportable event
* GET /api/v1/retirement/qrops-rops/{id}/reporting-checklist - Get reporting status
* GET /api/v1/retirement/qrops-rops/{id}/currency-exposure - Currency analysis

**Data Models:**

TABLE: qrops\_rops\_schemes

- id: UUID (PK)

- user\_id: UUID (FK to users)

- scheme\_type: ENUM('QROPS', 'ROPS', 'OTHER\_OVERSEAS\_PENSION')

- destination\_country: VARCHAR(100)

- scheme\_name: VARCHAR(255)

- scheme\_provider: VARCHAR(255)

- qrops\_reference\_number: VARCHAR(100)

- scheme\_address: TEXT

- current\_value: DECIMAL(15,2)

- destination\_currency: CHAR(3)

- current\_value\_gbp: DECIMAL(15,2) (calculated)

- current\_value\_zar: DECIMAL(15,2) (calculated)

- status: ENUM('ACTIVE', 'TRANSFERRED\_OUT', 'IN\_PAYMENT')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: qrops\_transfers

- id: UUID (PK)

- qrops\_scheme\_id: UUID (FK to qrops\_rops\_schemes)

- uk\_source\_pension\_id: UUID (FK to uk\_pensions, nullable)

- source\_scheme\_name: VARCHAR(255)

- source\_provider: VARCHAR(255)

- transfer\_value\_gbp: DECIMAL(15,2)

- transfer\_value\_destination: DECIMAL(15,2)

- transfer\_currency: CHAR(3)

- transfer\_date: DATE

- exchange\_rate\_at\_transfer: DECIMAL(10,6)

- member\_residency\_at\_transfer: VARCHAR(100)

- member\_tax\_residency\_at\_transfer: VARCHAR(100)

- years\_out\_of\_uk: INTEGER

- created\_at: TIMESTAMP

TABLE: overseas\_transfer\_charges

- id: UUID (PK)

- transfer\_id: UUID (FK to qrops\_transfers)

- charge\_applicable: BOOLEAN

- charge\_rate: DECIMAL(5,2)

- charge\_amount\_gbp: DECIMAL(15,2)

- charge\_amount\_destination: DECIMAL(15,2)

- charge\_paid\_by: ENUM('MEMBER', 'SCHEME')

- charge\_paid\_date: DATE

- reason\_for\_charge: TEXT

- exemption\_claimed: BOOLEAN

- exemption\_reason: TEXT

- created\_at: TIMESTAMP

TABLE: qrops\_reporting\_requirements

- id: UUID (PK)

- qrops\_scheme\_id: UUID (FK to qrops\_rops\_schemes)

- hmrc\_reporting\_required: BOOLEAN

- reporting\_period\_years: INTEGER (10 or 5)

- reporting\_start\_date: DATE

- reporting\_end\_date: DATE

- last\_report\_submitted\_date: DATE

- next\_report\_due\_date: DATE

- reporting\_complete: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: qrops\_reportable\_events

- id: UUID (PK)

- qrops\_scheme\_id: UUID (FK to qrops\_rops\_schemes)

- event\_type: ENUM('SCHEME\_LOST\_QROPS\_STATUS', 'MEMBER\_BECAME\_UK\_RESIDENT',

'PAYMENT\_MADE', 'TRANSFER\_OUT', 'DEATH', 'OTHER')

- event\_date: DATE

- event\_description: TEXT

- reported\_to\_hmrc: BOOLEAN DEFAULT FALSE

- report\_date: DATE

- hmrc\_reference: VARCHAR(100)

- created\_at: TIMESTAMP

TABLE: qrops\_value\_history

- id: UUID (PK)

- qrops\_scheme\_id: UUID (FK to qrops\_rops\_schemes)

- value\_date: DATE

- scheme\_value: DECIMAL(15,2)

- value\_currency: CHAR(3)

- exchange\_rate\_gbp: DECIMAL(10,6)

- exchange\_rate\_zar: DECIMAL(10,6)

- value\_gbp: DECIMAL(15,2) (calculated)

- value\_zar: DECIMAL(15,2) (calculated)

- created\_at: TIMESTAMP

TABLE: qrops\_benefit\_options

- qrops\_scheme\_id: UUID (PK, FK to qrops\_rops\_schemes)

- retirement\_age: INTEGER

- lump\_sum\_available: BOOLEAN

- lump\_sum\_percentage: DECIMAL(5,2)

- annuity\_required: BOOLEAN

- annuity\_percentage: DECIMAL(5,2)

- drawdown\_available: BOOLEAN

- death\_benefits: TEXT

- updated\_at: TIMESTAMP

INDEX on qrops\_rops\_schemes(user\_id, status)

INDEX on qrops\_transfers(qrops\_scheme\_id)

INDEX on qrops\_reporting\_requirements(next\_report\_due\_date)

INDEX on qrops\_reportable\_events(qrops\_scheme\_id, event\_date)

INDEX on qrops\_value\_history(qrops\_scheme\_id, value\_date DESC)

**Error Handling:**

ERROR CASES:

1. Transfer date in future

- Response: 400 Bad Request

- Message: "Transfer date cannot be in the future"

2. Invalid QROPS reference number format

- Response: 400 Bad Request

- Message: "QROPS reference number must be in format: QROPS######"

3. Transfer value below minimum

- Response: 400 Bad Request (warning)

- Message: "Transfer value {value} is below typical minimum of £30,000. Please verify"

4. Overseas transfer charge not paid when applicable

- Response: 400 Bad Request

- Message: "Overseas Transfer Charge of £{amount} was applicable but not recorded as paid"

5. HMRC report overdue

- Response: 200 OK with alert

- Alert: "HMRC report was due on {date}. Submit report immediately to avoid penalties"

6. Scheme lost QROPS status

- Response: 400 Bad Request (critical)

- Message: "This scheme is no longer a registered QROPS. Tax charges may apply"

- Action: "Report to HMRC immediately"

EDGE CASES:

- Multiple transfers to same QROPS: Track each separately

- Transfer from QROPS to another QROPS: Another reportable event, may trigger new charges

- Member returns to UK within 5 years: May trigger Overseas Transfer Charge retrospectively

- QROPS becomes ROPS: SA-based schemes after Brexit

- Death while in QROPS: Death benefits subject to destination country rules, report to HMRC

- Taking benefits before UK minimum pension age (55/57): May trigger UK tax charge

- Unapproved payments from QROPS: Unauthorized payments, report to HMRC, tax charge

- Currency hedging within scheme: Track separately, affects value

- Protected pension age: May lose protection on transfer

- Enhanced protection / Fixed protection: May lose on transfer

- Transfer to non-QROPS: Not allowed without severe tax consequences

- EEA transfers (pre-Brexit): No OTC, but still reporting requirements

- Gibraltar QROPS: Special rules, still part of HMRC QROPS list

**Performance Considerations:**

* Exchange rate updates: Daily batch job for currency conversions
* QROPS status verification: Weekly check against HMRC QROPS list (API or manual)
* Reporting deadline calculations: Pre-calculate, set reminders 30/60/90 days before
* Transfer charge calculation: Complex, cache result with transfer
* Expected schemes per user: 0-2 (relatively uncommon)
* Value tracking: Manual updates or annual statements
* Currency exposure analysis: Calculate on demand, cache for 24 hours
* Integration with UK pensions: Link source pension, mark as transferred out

**8. INHERITANCE TAX PLANNING MODULE**

**Feature 8.1: Assets Register**

**Feature Name:** Comprehensive Multi-Jurisdiction Asset Tracking for Estate Planning

**User Story:** As a user with assets in UK and SA, I want to maintain a complete register of all my assets including their location, ownership structure, and value so that I can plan my estate and understand potential inheritance tax/estate duty liabilities.

**Acceptance Criteria:**

* Track all assets by type (property, investments, cash, business interests, personal possessions)
* Record asset location (UK-situs, SA-situs, offshore)
* Track ownership structure (sole, joint, trust, company)
* Support for excluded property (non-UK domiciled assets)
* Valuation tracking with historical updates
* Asset categorization for IHT/Estate Duty purposes
* Integration with other modules (investments, savings, property)
* Document attachment for valuations and ownership proof
* Beneficiary designation where applicable

**Technical Requirements:**

* Complex ownership structure modeling
* Situs determination rules engine
* Excluded property calculator
* Asset valuation history tracking
* Integration with Investment and Savings modules
* Document management system
* Multi-currency support with conversion

**Constraints:**

* UK IHT: Applies to UK-situs assets regardless of domicile
* UK IHT: Applies to worldwide assets if UK domiciled/deemed domiciled
* SA Estate Duty: Applies based on residency and asset location
* Excluded property rules: Complex, based on domicile and asset type
* Valuation date: Date of death (or alternate valuation date)
* Joint ownership: Determines how assets pass (joint tenants vs tenants in common)

**Implementation Approach:**

ENDPOINT: POST /api/v1/iht/assets

REQUEST BODY:

{

assetType: enum['PROPERTY', 'INVESTMENT', 'CASH', 'BUSINESS\_INTEREST',

'LIFE\_POLICY', 'PENSION', 'PERSONAL\_POSSESSION', 'OTHER'],

assetCategory: enum['RESIDENTIAL\_PROPERTY', 'COMMERCIAL\_PROPERTY', 'LAND',

'QUOTED\_SHARES', 'UNQUOTED\_SHARES', 'UNIT\_TRUSTS',

'BANK\_ACCOUNT', 'ISA', 'SIPP', 'ARTWORK', 'JEWELRY',

'VEHICLE', 'INTELLECTUAL\_PROPERTY', 'OTHER'],

description: string,

location: enum['UK', 'SA', 'OFFSHORE', 'OTHER'],

situs: enum['UK\_SITUS', 'SA\_SITUS', 'NON\_UK\_NON\_SA\_SITUS', 'MOVEABLE'],

valuation: {

currentValue: decimal,

currency: string,

valuationDate: date,

valuationMethod: enum['MARKET\_VALUE', 'PROBATE\_VALUE', 'PROFESSIONAL\_VALUATION',

'COST\_BASIS', 'ESTIMATED'],

valuationSource: string (e.g., professional valuer name)

},

ownership: {

ownershipType: enum['SOLE', 'JOINT\_TENANTS', 'TENANTS\_IN\_COMMON',

'TRUST', 'COMPANY', 'PARTNERSHIP'],

ownershipPercentage: decimal (if not 100%),

jointOwners: [

{

name: string,

relationship: string,

percentage: decimal

}

],

trustDetails: {

trustName: string,

trustType: enum['BARE', 'DISCRETIONARY', 'INTEREST\_IN\_POSSESSION', 'OTHER'],

settlor: string,

beneficiaries: [string]

},

companyDetails: {

companyName: string,

registrationNumber: string,

jurisdiction: string,

ownershipPercentage: decimal

}

},

acquisition: {

acquisitionDate: date,

acquisitionCost: decimal,

acquisitionMethod: enum['PURCHASE', 'INHERITANCE', 'GIFT', 'TRANSFER']

},

taxation: {

ukIhtApplicable: boolean,

saEstateDutyApplicable: boolean,

excludedProperty: boolean,

excludedPropertyReason: string,

businessPropertyRelief: boolean,

bprPercentage: decimal (50% or 100%),

agriculturalPropertyRelief: boolean,

aprPercentage: decimal (50% or 100%)

},

mortgageDebt: {

hasMortgage: boolean,

outstandingBalance: decimal,

lender: string,

linkedLiabilityId: uuid (FK to liabilities)

},

beneficiaryDesignation: {

hasBeneficiaries: boolean,

beneficiaries: [

{

name: string,

relationship: string,

percentage: decimal

}

],

passesByWill: boolean

},

notes: text,

documentReferences: [uuid] // FKs to document storage

}

BUSINESS LOGIC:

1. Validate asset data:

- Current value >= 0 (can be 0 for depreciated assets)

- Ownership percentages total 100% (if joint ownership)

- Acquisition date <= today

- Valuation date <= today or reasonable projection

2. Determine situs (location for tax purposes):

// UK situs assets

IF assetCategory IN ['RESIDENTIAL\_PROPERTY', 'COMMERCIAL\_PROPERTY', 'LAND'] AND location = 'UK':

situs = 'UK\_SITUS'

IF assetCategory IN ['QUOTED\_SHARES'] AND company\_registered\_uk = TRUE:

situs = 'UK\_SITUS'

IF assetCategory IN ['BANK\_ACCOUNT', 'CASH'] AND bank\_branch\_uk = TRUE:

situs = 'UK\_SITUS'

// SA situs assets

IF location = 'SA' AND assetCategory IN ['PROPERTY', 'LAND']:

situs = 'SA\_SITUS'

// Moveable property (jewelry, artwork, vehicles)

IF assetCategory IN ['JEWELRY', 'ARTWORK', 'VEHICLE']:

situs = 'MOVEABLE' // Situs = location of owner at death

3. Determine IHT applicability:

user\_domicile = get\_user\_domicile()

// UK IHT applies if:

IF user\_domicile IN ['UK\_DOMICILED', 'UK\_DEEMED\_DOMICILED']:

uk\_iht\_applicable = TRUE // Worldwide assets

ELSE IF situs = 'UK\_SITUS':

uk\_iht\_applicable = TRUE // UK assets even if non-dom

ELSE:

uk\_iht\_applicable = FALSE

// Check for excluded property

IF user\_domicile = 'NON\_UK\_DOMICILED' AND situs != 'UK\_SITUS':

excluded\_property = TRUE

uk\_iht\_applicable = FALSE

4. Determine SA Estate Duty applicability:

// SA Estate Duty applies to:

// - Worldwide assets of SA residents

// - SA-situs assets of non-residents

IF user.sa\_resident:

sa\_estate\_duty\_applicable = TRUE // Worldwide

ELSE IF situs = 'SA\_SITUS':

sa\_estate\_duty\_applicable = TRUE

ELSE:

sa\_estate\_duty\_applicable = FALSE

5. Calculate reliefs (UK):

// Business Property Relief (BPR)

IF assetCategory = 'UNQUOTED\_SHARES' AND held\_for >= 2\_years:

bpr\_available = TRUE

bpr\_percentage = 100 // 100% relief

IF assetCategory = 'BUSINESS\_INTEREST' AND held\_for >= 2\_years:

bpr\_available = TRUE

bpr\_percentage = 100 // 100% relief for sole trader/partnership

IF assetCategory = 'QUOTED\_SHARES' AND company\_qualifies:

bpr\_available = TRUE

bpr\_percentage = 50 // 50% relief for controlling shareholding

// Agricultural Property Relief (APR)

IF assetCategory = 'LAND' AND agricultural\_use AND ownership\_or\_occupation\_criteria\_met:

apr\_available = TRUE

apr\_percentage = 100 or 50 // Depends on occupation

6. Calculate net asset value:

gross\_value = current\_value \* (ownership\_percentage / 100)

IF hasMortgage:

net\_value = gross\_value - outstanding\_mortgage\_balance

ELSE:

net\_value = gross\_value

IF bpr\_available:

relievable\_value = net\_value \* (bpr\_percentage / 100)

taxable\_value = net\_value - relievable\_value

ELSE:

taxable\_value = net\_value

7. Integration with other modules:

// Link to existing investment holdings

IF assetType = 'INVESTMENT':

linked\_investment\_id = find\_matching\_investment()

sync\_valuation\_with\_investment\_module()

// Link to savings accounts

IF assetType = 'CASH':

linked\_savings\_account\_id = find\_matching\_account()

sync\_valuation\_with\_savings\_module()

// Link to protection policies

IF assetType = 'LIFE\_POLICY':

linked\_policy\_id = find\_matching\_policy()

check\_if\_written\_in\_trust()

8. Currency conversion:

value\_gbp = convert\_to\_gbp(current\_value, currency, valuation\_date)

value\_zar = convert\_to\_zar(current\_value, currency, valuation\_date)

RESPONSE:

{

id: uuid,

assetDetails: {...},

valuation: {

currentValue: decimal,

currency: string,

valueGbp: decimal,

valueZar: decimal,

lastValuationDate: date

},

ownership: {

type: string,

userPercentage: decimal,

jointOwners: [...],

netValueToUser: decimal

},

taxation: {

situs: string,

ukIhtApplicable: boolean,

saEstateDutyApplicable: boolean,

excludedProperty: boolean,

reliefs: {

bpr: {available: boolean, percentage: decimal, relievedAmount: decimal},

apr: {available: boolean, percentage: decimal, relievedAmount: decimal}

},

taxableValue: {

uk: decimal,

sa: decimal

}

},

linkedAssets: {

investmentId: uuid,

savingsAccountId: uuid,

policyId: uuid

},

netValue: decimal (after mortgages/debts)

}

**User Flow:**

[IHT Dashboard] → [Assets Register Tab]

↓

[Assets Overview]

- Total estate value (gross and net)

- Assets by type (pie chart)

- Assets by location (UK, SA, Offshore)

- IHT/Estate Duty exposure

↓

[Add Asset Button]

↓

[Asset Entry - Step 1: Type & Category]

- Select asset type (visual cards):

- Property

- Investments

- Cash & Bank Accounts

- Business Interests

- Life Insurance

- Personal Possessions

- Other

- Select specific category

↓

[Asset Entry - Step 2: Location & Description]

- Asset description/name

- Physical location

- Situs (auto-calculated, can override)

↓

[Asset Entry - Step 3: Valuation]

- Current value and currency

- Valuation date

- Valuation method

- Valuation source (if professional)

- Upload valuation documents

↓

[Asset Entry - Step 4: Ownership]

- Ownership type selection:

- Sole ownership

- Joint ownership → Add co-owners with %

- Trust → Trust details

- Company → Company details

- Your ownership percentage

↓

[Asset Entry - Step 5: Acquisition]

- When acquired

- How acquired (purchase/gift/inheritance)

- Original cost (if applicable)

↓

[Asset Entry - Step 6: Tax Treatment]

- System calculates IHT applicability

- System calculates Estate Duty applicability

- Excluded property? (auto-determined)

- Eligible for BPR/APR? (if applicable)

- Mortgage/debt against asset?

↓

[Asset Entry - Step 7: Beneficiaries]

- Passes by will? OR

- Specific beneficiary designation

- Beneficiary details and percentages

↓

[Tax Treatment Summary]

- UK IHT impact

- SA Estate Duty impact

- Reliefs available

- Net taxable value

↓

[Save Asset]

↓

[Assets List View]

- Table/Card view with filters:

- Filter by type, location, ownership

- Sort by value, date acquired

- Quick actions: Edit, Update valuation, View details

- Color coding: UK IHT (red), SA Estate Duty (blue), Both (purple)

↓

[Asset Detail View]

- Complete asset information

- Valuation history chart

- Ownership structure diagram

- Tax treatment breakdown

- Linked liabilities

- Documents attached

- Edit/Delete options

**API Endpoints:**

* POST /api/v1/iht/assets - Add asset
* PUT /api/v1/iht/assets/{id} - Update asset
* DELETE /api/v1/iht/assets/{id} - Delete asset (soft delete)
* GET /api/v1/iht/assets - List all assets
* GET /api/v1/iht/assets/{id} - Get specific asset
* POST /api/v1/iht/assets/{id}/update-valuation - Update valuation
* GET /api/v1/iht/assets/{id}/valuation-history - Get valuation history
* GET /api/v1/iht/assets/summary - Assets summary by type/location
* POST /api/v1/iht/assets/bulk-import - Bulk import assets
* GET /api/v1/iht/assets/tax-treatment - Tax treatment summary
* POST /api/v1/iht/assets/{id}/upload-document - Upload document
* GET /api/v1/iht/assets/{id}/documents - Get asset documents

**Data Models:**

TABLE: iht\_assets

- id: UUID (PK)

- user\_id: UUID (FK to users)

- asset\_type: ENUM('PROPERTY', 'INVESTMENT', 'CASH', 'BUSINESS\_INTEREST',

'LIFE\_POLICY', 'PENSION', 'PERSONAL\_POSSESSION', 'OTHER')

- asset\_category: VARCHAR(100)

- description: VARCHAR(500)

- location: ENUM('UK', 'SA', 'OFFSHORE', 'OTHER')

- situs: ENUM('UK\_SITUS', 'SA\_SITUS', 'NON\_UK\_NON\_SA\_SITUS', 'MOVEABLE')

- current\_value: DECIMAL(15,2)

- currency: CHAR(3)

- value\_gbp: DECIMAL(15,2) (calculated)

- value\_zar: DECIMAL(15,2) (calculated)

- valuation\_date: DATE

- valuation\_method: VARCHAR(100)

- valuation\_source: VARCHAR(255)

- acquisition\_date: DATE

- acquisition\_cost: DECIMAL(15,2)

- acquisition\_method: ENUM('PURCHASE', 'INHERITANCE', 'GIFT', 'TRANSFER')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: asset\_ownership

- id: UUID (PK)

- asset\_id: UUID (FK to iht\_assets)

- ownership\_type: ENUM('SOLE', 'JOINT\_TENANTS', 'TENANTS\_IN\_COMMON',

'TRUST', 'COMPANY', 'PARTNERSHIP')

- user\_ownership\_percentage: DECIMAL(5,2) DEFAULT 100.00

- joint\_ownership: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

TABLE: asset\_joint\_owners

- id: UUID (PK)

- asset\_id: UUID (FK to iht\_assets)

- owner\_name: VARCHAR(255)

- relationship: VARCHAR(100)

- ownership\_percentage: DECIMAL(5,2)

- created\_at: TIMESTAMP

TABLE: asset\_trust\_details

- asset\_id: UUID (PK, FK to iht\_assets)

- trust\_name: VARCHAR(255)

- trust\_type: ENUM('BARE', 'DISCRETIONARY', 'INTEREST\_IN\_POSSESSION', 'OTHER')

- settlor\_name: VARCHAR(255)

- trustee\_names: JSON (array)

- beneficiary\_names: JSON (array)

- trust\_deed\_reference: UUID (FK to documents)

- created\_at: TIMESTAMP

TABLE: asset\_company\_ownership

- asset\_id: UUID (PK, FK to iht\_assets)

- company\_name: VARCHAR(255)

- company\_registration\_number: VARCHAR(100)

- jurisdiction: VARCHAR(100)

- user\_ownership\_percentage: DECIMAL(5,2)

- created\_at: TIMESTAMP

TABLE: asset\_taxation

- asset\_id: UUID (PK, FK to iht\_assets)

- uk\_iht\_applicable: BOOLEAN

- sa\_estate\_duty\_applicable: BOOLEAN

- excluded\_property: BOOLEAN

- excluded\_property\_reason: TEXT

- business\_property\_relief: BOOLEAN

- bpr\_percentage: DECIMAL(5,2)

- bpr\_relieved\_amount: DECIMAL(15,2)

- agricultural\_property\_relief: BOOLEAN

- apr\_percentage: DECIMAL(5,2)

- apr\_relieved\_amount: DECIMAL(15,2)

- uk\_taxable\_value: DECIMAL(15,2)

- sa\_taxable\_value: DECIMAL(15,2)

- updated\_at: TIMESTAMP

TABLE: asset\_valuation\_history

- id: UUID (PK)

- asset\_id: UUID (FK to iht\_assets)

- valuation\_date: DATE

- value: DECIMAL(15,2)

- currency: CHAR(3)

- value\_gbp: DECIMAL(15,2)

- value\_zar: DECIMAL(15,2)

- valuation\_method: VARCHAR(100)

- valuation\_source: VARCHAR(255)

- notes: TEXT

- created\_at: TIMESTAMP

TABLE: asset\_beneficiary\_designation

- id: UUID (PK)

- asset\_id: UUID (FK to iht\_assets)

- beneficiary\_name: VARCHAR(255)

- beneficiary\_relationship: VARCHAR(100)

- percentage: DECIMAL(5,2)

- passes\_by\_will: BOOLEAN DEFAULT TRUE

- created\_at: TIMESTAMP

TABLE: asset\_documents

- id: UUID (PK)

- asset\_id: UUID (FK to iht\_assets)

- document\_type: ENUM('VALUATION', 'DEED', 'TITLE', 'CERTIFICATE', 'OTHER')

- document\_name: VARCHAR(255)

- document\_reference: UUID (FK to document\_storage)

- upload\_date: DATE

- created\_at: TIMESTAMP

TABLE: asset\_module\_links

- asset\_id: UUID (PK, FK to iht\_assets)

- investment\_holding\_id: UUID (FK to investment\_holdings)

- savings\_account\_id: UUID (FK to savings\_accounts)

- life\_policy\_id: UUID (FK to life\_assurance\_policies)

- uk\_pension\_id: UUID (FK to uk\_pensions)

- sa\_retirement\_fund\_id: UUID (FK to sa\_retirement\_funds)

- sync\_enabled: BOOLEAN DEFAULT TRUE

- last\_sync: TIMESTAMP

VIEW: v\_estate\_summary (materialized view, refreshed daily)

- user\_id

- total\_gross\_estate\_gbp

- total\_gross\_estate\_zar

- total\_net\_estate\_gbp (after liabilities)

- total\_net\_estate\_zar

- uk\_iht\_liable\_assets\_gbp

- sa\_estate\_duty\_liable\_assets\_zar

- excluded\_property\_value\_gbp

- bpr\_relief\_total\_gbp

- apr\_relief\_total\_gbp

- asset\_count

- last\_updated: TIMESTAMP

INDEX on iht\_assets(user\_id, deleted)

INDEX on iht\_assets(asset\_type, location)

INDEX on asset\_ownership(asset\_id)

INDEX on asset\_valuation\_history(asset\_id, valuation\_date DESC)

INDEX on asset\_taxation(asset\_id)

CONSTRAINT: SUM(asset\_joint\_owners.ownership\_percentage WHERE asset\_id = X) +

asset\_ownership.user\_ownership\_percentage = 100

**Error Handling:**

ERROR CASES:

1. Negative asset value

- Response: 400 Bad Request

- Message: "Asset value cannot be negative. Use zero for worthless assets"

2. Valuation date in future

- Response: 400 Bad Request

- Message: "Valuation date cannot be in the future"

3. Ownership percentages don't total 100%

- Response: 400 Bad Request

- Message: "Total ownership percentages must equal 100%. Current total: {total}%"

4. Acquisition date after valuation date

- Response: 400 Bad Request

- Message: "Acquisition date must be before valuation date"

5. BPR claimed but holding period < 2 years

- Response: 400 Bad Request (warning)

- Message: "Business Property Relief requires 2 years ownership. Current ownership: {years} years"

- Allow user to save with acknowledgment that relief not yet available

6. Asset in trust but trust details missing

- Response: 400 Bad Request

- Message: "Please provide trust details for assets held in trust"

7. Mortgage exceeds asset value

- Response: 400 Bad Request (warning)

- Message: "Mortgage of £{mortgage} exceeds asset value of £{value} (negative equity)"

- Allow to proceed with warning

8. Duplicate asset detection

- Response: 409 Conflict (warning)

- Message: "Similar asset already exists: {description}. Continue adding?"

EDGE CASES:

- Joint tenants: Passes automatically to survivor, not by will

- Tenants in common: Passes by will, specify percentage

- Asset partly in UK, partly overseas: Split into separate assets

- Foreign property with UK mortgage: Complex situs rules, both may apply

- Reversionary interest in trust: May not have current value but future entitlement

- Life insurance in trust: Not part of estate if properly structured

- Pension death benefits: Normally discretionary, not estate assets

- Business assets used partly for business: Partial BPR available

- Agricultural land with development potential: May affect APR

- Quoted shares in suspended company: Valuation difficult, may be zero

- Art, antiques, collectibles: Professional valuation recommended

- Intellectual property: Difficult to value, may need specialist

- Cryptocurrency: Valuation at date of death, high volatility

- Foreign currency accounts: Value in GBP/ZAR at death

- Jointly owned property: Severance of joint tenancy affects IHT

- Property owned through company: Shares are the asset, not property

- Overseas property: May have local death taxes in addition to UK IHT

- Excluded property becoming liable: If non-dom becomes deemed domiciled

- BPR/APR withdrawal: If asset changes use before death

- Lifetime gifts with reservation: May still be in estate

**Performance Considerations:**

* Asset list with calculations: Use materialized view, refresh daily
* Valuation history: Paginate if >50 entries per asset
* Currency conversions: Batch update daily, cache exchange rates
* Tax treatment calculation: Complex, cache result per asset
* BPR/APR eligibility: Business rules engine, <100ms execution
* Expected assets per user: 5-50
* Estate summary calculation: Pre-aggregate, <500ms response
* Document uploads: Async processing for large files
* Bulk import: Process asynchronously, max 100 assets per import
* Integration sync: Daily batch job to sync with other modules
* Search and filter: Full-text search on description, indexed queries

**Feature 8.2: Liabilities Register**

**Feature Name:** Comprehensive Debt and Liability Tracking for Estate Calculation

**User Story:** As a user planning my estate, I want to record all my liabilities including mortgages, loans, credit cards, and other debts so that my net estate value is accurately calculated for inheritance tax purposes.

**Acceptance Criteria:**

* Track all liabilities (mortgages, personal loans, credit cards, business debts)
* Record liability details (creditor, amount, interest rate, repayment terms)
* Link liabilities to specific assets (e.g., mortgage to property)
* Deductibility determination for IHT/Estate Duty
* Track payment history and outstanding balance
* Project future liability reduction
* Alert for liabilities that may not be deductible
* Support for joint liabilities

**Technical Requirements:**

* Liability amortization calculator
* Deductibility rules engine (UK and SA)
* Asset-liability linking
* Payment tracking system
* Interest calculation
* Future projection modeling

**Constraints:**

* UK IHT: Liabilities deductible if legally owed at death
* UK IHT: Liabilities to "connected persons" may be restricted
* UK IHT: Foreign liabilities only deductible against foreign assets (non-doms)
* SA Estate Duty: Similar deductibility rules
* Double deduction: Cannot deduct same liability in both jurisdictions
* Contingent liabilities: May or may not be deductible

**Implementation Approach:**

ENDPOINT: POST /api/v1/iht/liabilities

REQUEST BODY:

{

liabilityType: enum['MORTGAGE', 'PERSONAL\_LOAN', 'CREDIT\_CARD', 'BUSINESS\_LOAN',

'OVERDRAFT', 'TAX\_LIABILITY', 'LEGAL\_OBLIGATION', 'GUARANTEE', 'OTHER'],

description: string,

creditor: string,

accountReference: string,

financial: {

outstandingBalance: decimal,

currency: string,

originalAmount: decimal,

interestRate: decimal (annual percentage),

startDate: date,

endDate: date (for term loans),

repaymentFrequency: enum['MONTHLY', 'QUARTERLY', 'ANNUALLY', 'BULLET', 'REVOLVING'],

monthlyPayment: decimal (if applicable)

},

linkedAsset: {

isSecured: boolean,

linkedAssetId: uuid (FK to iht\_assets),

securityType: enum['MORTGAGE', 'CHARGE', 'PLEDGE', 'UNSECURED']

},

ownership: {

jointLiability: boolean,

userPercentage: decimal,

jointDebtors: [

{

name: string,

relationship: string,

percentage: decimal

}

]

},

deductibility: {

ukIhtDeductible: boolean,

ukIhtDeductibleReason: text,

ukIhtDeductibleAmount: decimal,

saEstateDutyDeductible: boolean,

saEstateDutyDeductibleReason: text,

saEstateDutyDeductibleAmount: decimal,

connectedPerson: boolean,

connectedPersonDetails: string

},

status: enum['ACTIVE', 'PAID\_OFF', 'DEFAULTED', 'DISPUTED'],

notes: text

}

BUSINESS LOGIC:

1. Validate liability data:

- Outstanding balance >= 0

- Interest rate reasonable (0-30%)

- Start date <= today

- End date > start date (if applicable)

- If joint: percentages total 100%

2. Calculate user's share:

user\_liability\_share = outstanding\_balance \* (user\_percentage / 100)

3. Determine deductibility for UK IHT:

// General rule: Deductible if legally owed

deductible = TRUE

// Restrictions:

IF creditor\_is\_connected\_person:

// May be restricted or disallowed

deductible = FALSE or RESTRICTED

reason = "Liability to connected person - may not be deductible"

IF user.domicile = 'NON\_UK\_DOMICILED' AND liability\_location = 'OVERSEAS':

// Can only deduct against overseas assets, not UK assets

deduction\_limited\_to\_overseas\_assets = TRUE

IF liability\_type = 'GUARANTEE' AND not\_yet\_called:

deductible = FALSE

reason = "Contingent liability - only deductible if called upon"

IF liability\_incurred\_to\_acquire\_excluded\_property:

// May not be deductible for UK IHT

deductible = FALSE

reason = "Liability to acquire excluded property"

// Calculate deductible amount

IF deductible:

uk\_iht\_deductible\_amount = user\_liability\_share

ELSE:

uk\_iht\_deductible\_amount = 0

4. Determine deductibility for SA Estate Duty:

// Similar principles to UK

sa\_deductible = TRUE

IF liability\_to\_connected\_person:

sa\_deductible = RESTRICTED

IF contingent\_liability:

sa\_deductible = FALSE

sa\_estate\_duty\_deductible\_amount = calculate\_sa\_deduction()

5. Link to asset (if secured):

IF isSecured AND linkedAssetId:

asset = get\_asset(linkedAssetId)

// Check if liability exceeds asset value

IF user\_liability\_share > asset.net\_value:

alert = "Liability exceeds asset value (negative equity)"

// Update asset with linked liability

UPDATE iht\_assets

SET linked\_liability\_id = liability\_id,

net\_value = asset\_value - liability

WHERE id = linkedAssetId

6. Calculate projected payoff:

IF repaymentFrequency = 'MONTHLY':

months\_remaining = (end\_date - today) / 30

projected\_balance = calculate\_amortization(

outstanding\_balance,

interest\_rate,

monthly\_payment,

months\_remaining

)

// Project balance at expected death (actuarial age)

expected\_death\_age = user.life\_expectancy

years\_to\_expected\_death = expected\_death\_age - user.current\_age

balance\_at\_death = project\_balance(years\_to\_expected\_death)

7. Currency conversion:

liability\_gbp = convert\_to\_gbp(outstanding\_balance, currency)

liability\_zar = convert\_to\_zar(outstanding\_balance, currency)

RESPONSE:

{

id: uuid,

liabilityDetails: {...},

financial: {

outstandingBalance: decimal,

currency: string,

balanceGbp: decimal,

balanceZar: decimal,

userShare: decimal,

monthlyPayment: decimal,

interestRate: decimal

},

linkedAsset: {

assetId: uuid,

assetDescription: string,

assetValue: decimal,

loanToValue: decimal (percentage)

},

deductibility: {

ukIht: {

deductible: boolean,

deductibleAmount: decimal,

reason: string,

restrictions: [string]

},

saEstateDuty: {

deductible: boolean,

deductibleAmount: decimal,

reason: string,

restrictions: [string]

}

},

projection: {

yearsRemaining: decimal,

projectedPayoffDate: date,

balanceAtExpectedDeath: decimal

}

}

**User Flow:**

[IHT Dashboard] → [Liabilities Register Tab]

↓

[Liabilities Overview]

- Total liabilities (gross and net to user)

- Liabilities by type (pie chart)

- Secured vs unsecured

- Deductible vs non-deductible

↓

[Add Liability Button]

↓

[Liability Entry - Step 1: Type]

- Select liability type (visual cards):

- Mortgage

- Personal Loan

- Credit Card

- Business Loan

- Other Debt

↓

[Liability Entry - Step 2: Creditor & Details]

- Liability description

- Creditor name

- Account/reference number

↓

[Liability Entry - Step 3: Financial Details]

- Outstanding balance and currency

- Original amount

- Interest rate

- Start date and end date

- Repayment frequency

- Monthly payment amount

↓

[Liability Entry - Step 4: Security]

- Is this secured against an asset?

- If YES: Select asset from list

- Security type (mortgage, charge, etc.)

- System shows asset value and LTV

↓

[Liability Entry - Step 5: Ownership]

- Is this a joint liability?

- If YES: Add joint debtors with %

- Your percentage share

↓

[Liability Entry - Step 6: Deductibility]

- System calculates deductibility

- Is creditor a connected person?

- If YES: Warn about potential restrictions

- Show UK IHT deductibility

- Show SA Estate Duty deductibility

↓

[Deductibility Summary]

- UK IHT: Deductible £X (or not deductible with reason)

- SA Estate Duty: Deductible RX (or not deductible with reason)

- Any restrictions or warnings

↓

[Projection Display]

- Payoff timeline chart

- Balance at expected death (based on life expectancy)

- Effect on net estate

↓

[Save Liability]

↓

[Liabilities List View]

- Table/Card view with key details

- Filter: By type, creditor, deductibility

- Sort: By balance, interest rate, payoff date

- Color coding: Deductible (green), Non-deductible (red), Restricted (amber)

- Quick actions: Make payment, Update balance, View details

↓

[Liability Detail View]

- Complete liability information

- Payment history

- Amortization schedule

- Linked asset details

- Deductibility breakdown

- Edit/Delete options

**API Endpoints:**

* POST /api/v1/iht/liabilities - Add liability
* PUT /api/v1/iht/liabilities/{id} - Update liability
* DELETE /api/v1/iht/liabilities/{id} - Delete liability (soft delete)
* GET /api/v1/iht/liabilities - List all liabilities
* GET /api/v1/iht/liabilities/{id} - Get specific liability
* POST /api/v1/iht/liabilities/{id}/payment - Record payment
* GET /api/v1/iht/liabilities/{id}/amortization-schedule - Get schedule
* GET /api/v1/iht/liabilities/summary - Liabilities summary
* POST /api/v1/iht/liabilities/{id}/update-balance - Update balance
* GET /api/v1/iht/liabilities/deductibility-analysis - Deductibility summary

**Data Models:**

TABLE: iht\_liabilities

- id: UUID (PK)

- user\_id: UUID (FK to users)

- liability\_type: ENUM('MORTGAGE', 'PERSONAL\_LOAN', 'CREDIT\_CARD', 'BUSINESS\_LOAN',

'OVERDRAFT', 'TAX\_LIABILITY', 'LEGAL\_OBLIGATION', 'GUARANTEE', 'OTHER')

- description: VARCHAR(500)

- creditor: VARCHAR(255)

- account\_reference: VARCHAR(100) ENCRYPTED

- outstanding\_balance: DECIMAL(15,2)

- currency: CHAR(3)

- balance\_gbp: DECIMAL(15,2) (calculated)

- balance\_zar: DECIMAL(15,2) (calculated)

- original\_amount: DECIMAL(15,2)

- interest\_rate: DECIMAL(5,2)

- start\_date: DATE

- end\_date: DATE

- repayment\_frequency: ENUM('MONTHLY', 'QUARTERLY', 'ANNUALLY', 'BULLET', 'REVOLVING')

- monthly\_payment: DECIMAL(10,2)

- status: ENUM('ACTIVE', 'PAID\_OFF', 'DEFAULTED', 'DISPUTED')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: liability\_security

- liability\_id: UUID (PK, FK to iht\_liabilities)

- is\_secured: BOOLEAN

- linked\_asset\_id: UUID (FK to iht\_assets)

- security\_type: ENUM('MORTGAGE', 'CHARGE', 'PLEDGE', 'UNSECURED')

- loan\_to\_value: DECIMAL(5,2) (calculated)

- created\_at: TIMESTAMP

TABLE: liability\_ownership

- id: UUID (PK)

- liability\_id: UUID (FK to iht\_liabilities)

- joint\_liability: BOOLEAN DEFAULT FALSE

- user\_percentage: DECIMAL(5,2) DEFAULT 100.00

- user\_liability\_share: DECIMAL(15,2) (calculated)

- created\_at: TIMESTAMP

TABLE: liability\_joint\_debtors

- id: UUID (PK)

- liability\_id: UUID (FK to iht\_liabilities)

- debtor\_name: VARCHAR(255)

- relationship: VARCHAR(100)

- percentage: DECIMAL(5,2)

- created\_at: TIMESTAMP

TABLE: liability\_deductibility

- liability\_id: UUID (PK, FK to iht\_liabilities)

- uk\_iht\_deductible: BOOLEAN

- uk\_iht\_deductible\_amount: DECIMAL(15,2)

- uk\_iht\_deductible\_reason: TEXT

- uk\_iht\_restrictions: JSON (array of restrictions)

- sa\_estate\_duty\_deductible: BOOLEAN

- sa\_estate\_duty\_deductible\_amount: DECIMAL(15,2)

- sa\_estate\_duty\_deductible\_reason: TEXT

- sa\_estate\_duty\_restrictions: JSON

- connected\_person: BOOLEAN DEFAULT FALSE

- connected\_person\_details: TEXT

- contingent\_liability: BOOLEAN DEFAULT FALSE

- updated\_at: TIMESTAMP

TABLE: liability\_payments

- id: UUID (PK)

- liability\_id: UUID (FK to iht\_liabilities)

- payment\_date: DATE

- payment\_amount: DECIMAL(10,2)

- principal\_paid: DECIMAL(10,2)

- interest\_paid: DECIMAL(10,2)

- balance\_after\_payment: DECIMAL(15,2)

- payment\_reference: VARCHAR(100)

- created\_at: TIMESTAMP

TABLE: liability\_projections

- liability\_id: UUID (PK, FK to iht\_liabilities)

- projection\_date: DATE

- years\_remaining: DECIMAL(5,2)

- projected\_payoff\_date: DATE

- balance\_at\_expected\_death: DECIMAL(15,2)

- total\_interest\_to\_pay: DECIMAL(15,2)

- last\_calculated: TIMESTAMP

VIEW: v\_liabilities\_summary (materialized view)

- user\_id

- total\_liabilities\_gbp

- total\_liabilities\_zar

- total\_secured\_liabilities

- total\_unsecured\_liabilities

- uk\_iht\_deductible\_total

- sa\_estate\_duty\_deductible\_total

- non\_deductible\_total

- liability\_count

- last\_updated: TIMESTAMP

INDEX on iht\_liabilities(user\_id, status)

INDEX on liability\_security(linked\_asset\_id)

INDEX on liability\_payments(liability\_id, payment\_date DESC)

CONSTRAINT: SUM(liability\_joint\_debtors.percentage) +

liability\_ownership.user\_percentage = 100

**Error Handling:**

ERROR CASES:

1. Negative outstanding balance

- Response: 400 Bad Request

- Message: "Outstanding balance cannot be negative"

2. Interest rate unreasonable

- Response: 400 Bad Request (warning)

- Message: "Interest rate of {rate}% seems unusually high. Please verify"

- Allow if user confirms

3. Monthly payment insufficient for interest

- Response: 400 Bad Request (warning)

- Message: "Monthly payment of £{payment} does not cover monthly interest of £{interest}. Loan will never be repaid"

- Allow with warning (negative amortization)

4. End date before start date

- Response: 400 Bad Request

- Message: "Loan end date must be after start date"

5. Linked asset not found

- Response: 404 Not Found

- Message: "Asset with ID {id} not found. Please select a valid asset"

6. Joint liability percentages don't total 100%

- Response: 400 Bad Request

- Message: "Total liability percentages must equal 100%. Current total: {total}%"

7. Secured liability exceeds asset value

- Response: 400 Bad Request (warning)

- Message: "Liability of £{liability} exceeds asset value of £{asset}. This is negative equity"

- Allow with warning

8. Connected person liability without justification

- Response: 400 Bad Request (warning)

- Message: "Liabilities to connected persons may not be deductible for IHT. Please provide details"

EDGE CASES:

- Revolving credit: Balance fluctuates, use current balance

- Interest-only mortgage: No principal repayment, balloon payment at end

- Payment holiday: Adjust amortization schedule

- Early repayment charge: Factor into payoff calculation

- Variable interest rate: Use current rate for projection, note assumption

- Foreign currency loan: Exchange rate risk affects balance

- Joint and several liability: Each debtor liable for full amount

- Liability paid off: Mark as PAID\_OFF, retain for history

- Disputed liability: May not be deductible until resolved

- Contingent liability (guarantee): Only deductible if called upon

- Mortgage with life insurance: May be paid off at death (note this)

- Tax liability: Deductible for estate duty if legally due

- Funeral expenses: Deductible from estate (estimate)

- Business loan with personal guarantee: Track separately

- Shareholder loan: May be treated as part of business valuation

- Liability in trust: May not be personal liability

- Post-death liabilities: Probate costs, executor fees (estimate separately)

**Performance Considerations:**

* Amortization calculation: Complex, cache schedule
* Payment history: Paginate if >100 payments
* Deductibility determination: Business rules engine, <100ms
* Currency conversion: Daily batch update
* Expected liabilities per user: 2-15
* Liabilities summary: Use materialized view, <500ms response
* Projection calculations: Pre-calculate monthly, store results
* Integration with assets: Real-time LTV calculation
* Bulk operations: Support batch payment recording

**Feature 8.3: Estate Calculation & IHT/Estate Duty Projection**

**Feature Name:** Comprehensive Estate Valuation and Tax Liability Calculation

**User Story:** As a user, I want to see my complete estate calculation including UK Inheritance Tax and SA Estate Duty projections so that I can understand my potential tax liability and plan accordingly.

**Acceptance Criteria:**

* Calculate gross estate value (all assets)
* Calculate net estate value (assets minus liabilities)
* Apply UK IHT calculation with Nil Rate Band, Residence NRB, and reliefs
* Apply SA Estate Duty calculation with abatement
* Show tax liability in both jurisdictions
* Account for Double Tax Agreement provisions
* Show available allowances and reliefs
* Project estate value at expected death
* Scenario modeling (different death ages, asset values)
* Recommendations for tax reduction

**Technical Requirements:**

* Complex multi-jurisdiction estate calculation engine
* NRB and RNRB calculator
* Transferable NRB from deceased spouse tracking
* BPR and APR application
* SA Estate Duty calculator with Section 4(q) deductions
* DTA relief calculator
* Monte Carlo simulation for projections (optional)
* Scenario modeling engine

**Constraints:**

* UK IHT: 40% on estate above NRB (£325,000) + RNRB (up to £175,000)
* UK RNRB: Tapered away if estate >£2 million
* UK IHT: Reduced to 36% if 10%+ left to charity
* SA Estate Duty: 20% on dutiable amount over R30 million (2024)
* SA Abatement: R3.5 million (2024)
* DTA: Relief for assets taxed in both jurisdictions
* Calculations at date of death (not current date)

**Implementation Approach:**

ENDPOINT: GET /api/v1/iht/estate-calculation

QUERY PARAMS:

- asOfDate: date (default: today, or projected death date)

- scenarioType: enum['CURRENT', 'EXPECTED\_DEATH', 'CUSTOM']

- customAge: integer (for custom scenarios)

BUSINESS LOGIC:

1. Gather all estate components:

total\_assets = SUM(iht\_assets WHERE user\_id = user.id AND deleted = FALSE)

total\_liabilities = SUM(iht\_liabilities WHERE user\_id = user.id AND deleted = FALSE)

gross\_estate = total\_assets

net\_estate = total\_assets - total\_liabilities

2. Separate assets by jurisdiction:

uk\_assets = SUM(iht\_assets WHERE uk\_iht\_applicable = TRUE)

sa\_assets = SUM(iht\_assets WHERE sa\_estate\_duty\_applicable = TRUE)

excluded\_property = SUM(iht\_assets WHERE excluded\_property = TRUE)

// Adjust for excluded property

uk\_taxable\_estate = uk\_assets - excluded\_property

sa\_taxable\_estate = sa\_assets

3. Calculate UK Inheritance Tax:

// Nil Rate Band (NRB)

current\_nrb = 325000 // £325k as of 2024

// Transferable NRB from deceased spouse

transferable\_nrb = get\_transferable\_nrb\_from\_spouse()

total\_nrb = current\_nrb + transferable\_nrb

// Residence Nil Rate Band (RNRB)

max\_rnrb = 175000 // £175k as of 2024

// RNRB conditions:

// - Must own qualifying residential property

// - Must pass to direct descendants

// - Tapered if estate > £2m

IF has\_qualifying\_residence AND passes\_to\_descendants:

IF net\_estate <= 2000000:

rnrb = max\_rnrb

ELSE:

taper = (net\_estate - 2000000) / 2 // £1 reduction for every £2 over

rnrb = MAX(max\_rnrb - taper, 0)

ELSE:

rnrb = 0

// Transferable RNRB from deceased spouse

transferable\_rnrb = get\_transferable\_rnrb\_from\_spouse()

total\_rnrb = rnrb + transferable\_rnrb

// Total nil rate bands

total\_nil\_rate\_bands = total\_nrb + total\_rnrb

// Apply reliefs (BPR and APR)

bpr\_relief = SUM(iht\_assets.bpr\_relieved\_amount WHERE user\_id = user.id)

apr\_relief = SUM(iht\_assets.apr\_relieved\_amount WHERE user\_id = user.id)

total\_reliefs = bpr\_relief + apr\_relief

// Calculate taxable estate

taxable\_estate\_after\_reliefs = uk\_taxable\_estate - total\_reliefs

chargeable\_estate = MAX(taxable\_estate\_after\_reliefs - total\_nil\_rate\_bands, 0)

// Standard IHT rate: 40%

standard\_rate = 0.40

// Reduced rate: 36% if 10%+ left to charity

IF charitable\_legacy\_percentage >= 10:

iht\_rate = 0.36

ELSE:

iht\_rate = standard\_rate

uk\_iht\_liability = chargeable\_estate \* iht\_rate

4. Calculate SA Estate Duty:

// SA Estate Duty calculation

gross\_estate\_sa = sa\_taxable\_estate

// Deductions under Section 4(q)

section\_4q\_deductions = [

funeral\_expenses,

executor\_fees,

master\_fees,

valuator\_fees,

other\_admin\_costs

]

total\_section\_4q = SUM(section\_4q\_deductions)

// Property in spouse's estate (Section 4A)

property\_in\_spouse\_estate = calculate\_spouse\_estate\_property()

// Deductible liabilities

deductible\_liabilities = SUM(iht\_liabilities.sa\_estate\_duty\_deductible\_amount)

// Net value of estate

net\_estate\_sa = gross\_estate\_sa - total\_section\_4q - deductible\_liabilities

// Abatement (R3.5 million as of 2024)

abatement = 3500000

// Dutiable amount

dutiable\_amount = MAX(net\_estate\_sa - abatement, 0)

// Estate duty threshold (R30 million as of 2024)

estate\_duty\_threshold = 30000000

IF dutiable\_amount > estate\_duty\_threshold:

// 20% on amount over R30 million

sa\_estate\_duty = (dutiable\_amount - estate\_duty\_threshold) \* 0.20

ELSE:

sa\_estate\_duty = 0

5. Apply Double Tax Agreement relief:

// Assets taxed in both jurisdictions

dual\_taxed\_assets = identify\_dual\_taxed\_assets()

// DTA relief: Credit for tax paid in other jurisdiction

// Typically, country of situs taxes first, residence country gives credit

FOR EACH dual\_taxed\_asset:

IF asset.situs = 'UK':

uk\_tax\_on\_asset = calculate\_uk\_tax(asset)

sa\_tax\_on\_asset = calculate\_sa\_tax(asset)

// SA gives credit for UK tax paid

sa\_dta\_relief += MIN(uk\_tax\_on\_asset, sa\_tax\_on\_asset)

ELSE IF asset.situs = 'SA':

sa\_tax\_on\_asset = calculate\_sa\_tax(asset)

uk\_tax\_on\_asset = calculate\_uk\_tax(asset)

// UK gives credit for SA tax paid

uk\_dta\_relief += MIN(sa\_tax\_on\_asset, uk\_tax\_on\_asset)

// Apply relief

uk\_iht\_liability\_after\_dta = uk\_iht\_liability - uk\_dta\_relief

sa\_estate\_duty\_after\_dta = sa\_estate\_duty - sa\_dta\_relief

total\_death\_taxes = uk\_iht\_liability\_after\_dta + sa\_estate\_duty\_after\_dta

6. Calculate effective tax rate:

effective\_uk\_rate = (uk\_iht\_liability / uk\_taxable\_estate) \* 100

effective\_sa\_rate = (sa\_estate\_duty / sa\_taxable\_estate) \* 100

effective\_overall\_rate = (total\_death\_taxes / net\_estate) \* 100

7. Project future estate value:

IF scenarioType = 'EXPECTED\_DEATH':

years\_to\_death = user.life\_expectancy - user.current\_age

// Project asset growth

projected\_assets = project\_assets(current\_assets, years\_to\_death)

// Project liability reduction

projected\_liabilities = project\_liabilities(current\_liabilities, years\_to\_death)

// Recalculate with projections

projected\_estate\_calculation = calculate\_estate(projected\_assets, projected\_liabilities)

8. Identify planning opportunities:

recommendations = []

IF uk\_iht\_liability > 0:

IF not\_using\_full\_nrb:

recommendations.add("Consider equalizing estates with spouse to use both NRBs")

IF rnrb = 0 AND has\_direct\_descendants:

recommendations.add("Consider purchasing qualifying residence to utilize RNRB")

IF has\_business\_assets AND not\_claiming\_bpr:

recommendations.add("Review business assets for potential BPR")

IF charitable\_legacy\_percentage < 10:

recommendations.add("Consider 10% charitable legacy for reduced IHT rate (36%)")

IF sa\_estate\_duty > 0:

IF not\_using\_spouse\_benefits:

recommendations.add("Consider Section 4A deductions for property in spouse's estate")

RESPONSE:

{

calculationDate: date,

scenarioType: string,

estateValuation: {

grossEstate: {gbp: decimal, zar: decimal},

totalLiabilities: {gbp: decimal, zar: decimal},

netEstate: {gbp: decimal, zar: decimal}

},

assetBreakdown: {

ukAssets: {gbp: decimal, percentage: decimal},

saAssets: {zar: decimal, percentage: decimal},

excludedProperty: {gbp: decimal},

byCategoryGbp: {...},

byCategoryZar: {...}

},

ukInheritanceTax: {

taxableEstate: decimal,

nilRateBand: {

current: decimal,

transferredFromSpouse: decimal,

total: decimal,

unused: decimal

},

residenceNilRateBand: {

maximum: decimal,

taperReduction: decimal,

available: decimal,

transferredFromSpouse: decimal,

total: decimal

},

reliefs: {

businessPropertyRelief: decimal,

agriculturalPropertyRelief: decimal,

total: decimal

},

chargeableEstate: decimal,

taxRate: decimal,

grossTaxLiability: decimal,

dtaRelief: decimal,

netTaxLiability: decimal,

effectiveRate: decimal

},

saEstateDuty: {

grossEstate: decimal,

section4qDeductions: decimal,

deductibleLiabilities: decimal,

netEstate: decimal,

abatement: decimal,

dutiableAmount: decimal,

threshold: decimal,

taxRate: decimal (20%),

grossTaxLiability: decimal,

dtaRelief: decimal,

netTaxLiability: decimal,

effectiveRate: decimal

},

totalDeathTaxes: {

ukIht: decimal,

saEstateDuty: decimal,

total: {gbp: decimal, zar: decimal},

effectiveOverallRate: decimal

},

netEstateAfterTax: {

beforeTax: {gbp: decimal, zar: decimal},

totalTax: {gbp: decimal, zar: decimal},

afterTax: {gbp: decimal, zar: decimal},

percentageReduction: decimal

},

projection: {

yearsToProjectedDeath: integer,

projectedGrossEstate: {gbp: decimal, zar: decimal},

projectedTaxLiability: {gbp: decimal, zar: decimal},

assumptions: {...}

},

recommendations: [

{

category: string,

title: string,

description: string,

estimatedSaving: {gbp: decimal, zar: decimal},

priority: enum['HIGH', 'MEDIUM', 'LOW']

}

],

comparisonWithPreviousCalculation: {

previousDate: date,

estateValueChange: {amount: decimal, percentage: decimal},

taxLiabilityChange: {amount: decimal, percentage: decimal}

}

}

**User Flow:**

[IHT Dashboard] → [Estate Calculation Tab]

↓

[Estate Summary (Hero Section)]

- Net estate value (prominent)

- Total tax liability (UK + SA)

- Effective tax rate %

- Net estate after tax

↓

[Scenario Selector]

- Current position (today's values)

- At expected death (life expectancy projection)

- Custom age scenario

↓

[UK Inheritance Tax Section]

- Taxable estate £X

- Nil Rate Bands:

- Your NRB: £325,000

- Transferred from spouse: £X

- RNRB: £X

- Total: £X

- Reliefs applied:

- BPR: £X

- APR: £X

- Chargeable estate: £X

- Tax rate: 40% (or 36% if charity)

- IHT liability: £X

↓

[SA Estate Duty Section]

- Gross estate: RX

- Deductions: RX

- Abatement: R3.5m

- Dutiable amount: RX

- Estate Duty: RX

↓

[DTA Relief Section]

- Assets taxed in both jurisdictions

- Relief applied

- Net tax in each country

↓

[Visual Breakdown]

- Waterfall chart: Gross estate → Reliefs → NRBs → Tax → Net estate

- Pie chart: Where tax is payable (UK vs SA)

- Comparison: Before tax vs After tax

↓

[Planning Opportunities Section]

- AI-generated recommendations

- Prioritized by potential saving

- Action buttons for each

↓

[Projection Timeline]

- Chart: Estate value over time

- Chart: Tax liability over time

- Slider: Adjust age at death for scenarios

↓

[Detailed Breakdown (Expandable)]

- All assets listed with values

- All liabilities listed

- Full calculation methodology

- Assumptions stated

↓

[Actions]

- Download PDF report

- Share with advisor

- Schedule review reminder

- Model "what-if" scenarios

**API Endpoints:**

* GET /api/v1/iht/estate-calculation - Get estate calculation
* POST /api/v1/iht/estate-calculation/scenario - Run scenario analysis
* GET /api/v1/iht/estate-calculation/history - Historical calculations
* POST /api/v1/iht/estate-calculation/compare - Compare scenarios
* GET /api/v1/iht/nil-rate-bands - Get NRB and RNRB details
* POST /api/v1/iht/transferable-nrb - Calculate transferable NRB
* GET /api/v1/iht/planning-opportunities - Get recommendations
* POST /api/v1/iht/estate-projection - Project future estate
* GET /api/v1/iht/estate-calculation/pdf - Generate PDF report

**Data Models:**

TABLE: estate\_calculations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- calculation\_date: DATE

- calculation\_type: ENUM('CURRENT', 'EXPECTED\_DEATH', 'CUSTOM\_SCENARIO', 'WHAT\_IF')

- scenario\_age: INTEGER (if custom)

- gross\_estate\_gbp: DECIMAL(15,2)

- gross\_estate\_zar: DECIMAL(15,2)

- total\_liabilities\_gbp: DECIMAL(15,2)

- total\_liabilities\_zar: DECIMAL(15,2)

- net\_estate\_gbp: DECIMAL(15,2)

- net\_estate\_zar: DECIMAL(15,2)

- uk\_iht\_liability: DECIMAL(15,2)

- sa\_estate\_duty\_liability: DECIMAL(15,2)

- total\_death\_taxes\_gbp: DECIMAL(15,2)

- total\_death\_taxes\_zar: DECIMAL(15,2)

- net\_estate\_after\_tax\_gbp: DECIMAL(15,2)

- net\_estate\_after\_tax\_zar: DECIMAL(15,2)

- effective\_tax\_rate: DECIMAL(5,2)

- calculation\_details\_json: JSON (full calculation breakdown)

- created\_at: TIMESTAMP

TABLE: uk\_nil\_rate\_bands

- user\_id: UUID (PK, FK to users)

- current\_nrb: DECIMAL(15,2) DEFAULT 325000

- transferable\_nrb\_available: DECIMAL(15,2)

- transferable\_nrb\_source: TEXT (spouse details)

- transferable\_nrb\_date: DATE

- current\_rnrb: DECIMAL(15,2)

- max\_rnrb: DECIMAL(15,2) DEFAULT 175000

- rnrb\_taper\_reduction: DECIMAL(15,2)

- transferable\_rnrb\_available: DECIMAL(15,2)

- total\_nrb: DECIMAL(15,2) (calculated)

- total\_rnrb: DECIMAL(15,2) (calculated)

- updated\_at: TIMESTAMP

TABLE: estate\_planning\_recommendations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- calculation\_id: UUID (FK to estate\_calculations)

- recommendation\_category: ENUM('NRB\_OPTIMIZATION', 'RNRB\_PLANNING', 'RELIEF\_CLAIMING',

'CHARITABLE\_GIVING', 'GIFT\_PLANNING', 'TRUST\_STRUCTURE', 'OTHER')

- recommendation\_title: VARCHAR(255)

- recommendation\_description: TEXT

- estimated\_saving\_gbp: DECIMAL(15,2)

- estimated\_saving\_zar: DECIMAL(15,2)

- priority: ENUM('HIGH', 'MEDIUM', 'LOW')

- status: ENUM('NEW', 'IN\_PROGRESS', 'COMPLETED', 'DISMISSED')

- action\_taken: TEXT

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: estate\_projections

- id: UUID (PK)

- user\_id: UUID (FK to users)

- projection\_date: DATE

- years\_projected: INTEGER

- projected\_age\_at\_death: INTEGER

- projected\_gross\_estate\_gbp: DECIMAL(15,2)

- projected\_net\_estate\_gbp: DECIMAL(15,2)

- projected\_tax\_liability\_gbp: DECIMAL(15,2)

- assumptions: JSON (growth rates, asset changes, etc.)

- created\_at: TIMESTAMP

TABLE: dta\_relief\_calculations

- id: UUID (PK)

- calculation\_id: UUID (FK to estate\_calculations)

- asset\_id: UUID (FK to iht\_assets)

- asset\_description: VARCHAR(255)

- asset\_situs: VARCHAR(50)

- uk\_tax\_on\_asset: DECIMAL(15,2)

- sa\_tax\_on\_asset: DECIMAL(15,2)

- relief\_given\_by: ENUM('UK', 'SA')

- relief\_amount: DECIMAL(15,2)

- created\_at: TIMESTAMP

VIEW: v\_current\_estate\_summary (materialized view, refreshed on asset/liability change)

- user\_id

- gross\_estate\_gbp

- net\_estate\_gbp

- uk\_iht\_estimate

- sa\_estate\_duty\_estimate

- total\_tax\_estimate

- net\_after\_tax

- last\_updated: TIMESTAMP

INDEX on estate\_calculations(user\_id, calculation\_date DESC)

INDEX on estate\_planning\_recommendations(user\_id, status, priority)

INDEX on estate\_projections(user\_id, projection\_date DESC)

**Error Handling:**

ERROR CASES:

1. Insufficient data for calculation

- Response: 400 Bad Request

- Message: "Cannot calculate estate tax. Please add at least one asset"

2. Missing critical tax status information

- Response: 400 Bad Request

- Message: "Domicile status required for accurate I

HT calculation. Please complete your tax status information"

1. RNRB calculation error - no qualifying residence
   * Response: 200 OK (with warning)
   * Warning: "RNRB not applied: No qualifying residential property found in estate"
   * Recommendation: "Consider property ownership for RNRB eligibility"
2. Transferable NRB claimed but no spouse details
   * Response: 400 Bad Request
   * Message: "Cannot calculate transferable NRB without deceased spouse details"
3. BPR/APR claimed on assets not meeting holding period
   * Response: 200 OK (with warning)
   * Warning: "BPR not applied to {asset}: Held for {period}, requires 2 years"
4. Scenario projection beyond reasonable age
   * Response: 400 Bad Request
   * Message: "Custom age {age} exceeds reasonable life expectancy. Maximum: 120"
5. DTA relief calculation error
   * Response: 200 OK (with warning)
   * Warning: "Unable to calculate DTA relief for some assets. Manual review recommended"
6. Negative net estate (liabilities exceed assets)
   * Response: 200 OK
   * Message: "Net estate is negative. No inheritance tax liability"
   * Alert: "Consider reviewing liability levels"

EDGE CASES:

* Estate exactly at NRB threshold: Tax = £0, but planning still beneficial
* Estate £1 over threshold: Marginal rate applies to that £1
* RNRB taper: Precise calculation needed, £1 for £2 reduction
* Multiple deaths in quick succession: Consider quick succession relief
* Non-UK domiciled with UK assets: Only UK assets taxed
* Deemed domicile acquired mid-year: Pro-rata calculation
* Assets in trust: May or may not be in estate (depends on trust type)
* Life insurance in trust: Outside estate if properly structured
* Joint tenancy: Passes outside will, but still in estate for IHT
* Business assets sold before death: BPR lost
* Gifts with reservation: Back in estate for IHT
* Potentially exempt transfers (PETs): Taxable if death within 7 years
* Taper relief on gifts: Reduces tax if death 3-7 years after gift
* Charitable legacy exactly 10%: Qualifies for reduced rate
* Charitable legacy 9.9%: Doesn't qualify, consider rounding up
* Assets in different currencies: Exchange rate at death applies
* Valuation disputes: Probate value may differ from current estimate
* Business assets: Valuation complex, may need professional
* Quoted shares: Valuation = lower of quarter-up or average
* Unquoted shares: Professional valuation required
* Property: Market value, not mortgage outstanding
* Jointly owned assets: Only deceased's share in estate
* Foreign assets: May have local death taxes too
* Pension death benefits: Usually discretionary, not in estate
* ISA on death: Becomes a "continuing ISA" for spouse
* Offshore bonds: Complex tax treatment on death
* Intellectual property: Valuation challenging
* Digital assets: Cryptocurrency, domain names, etc.

\*\*Performance Considerations:\*\*

- Estate calculation: Complex, cache results for 24 hours

- Trigger recalculation when: Assets change, liabilities change, tax status changes

- NRB/RNRB lookup: Fast, simple calculation

- BPR/APR application: Iterate through assets, apply rules

- DTA relief: Most complex part, may take 1-2 seconds

- Scenario modeling: Multiple calculations, async processing for >3 scenarios

- Expected calculation time: <3 seconds for complete estate

- Projection calculations: Time-intensive, use background job for long projections

- PDF generation: Async job, 10-30 seconds depending on estate complexity

- Historical comparison: Pre-aggregate monthly snapshots

- Real-time updates: WebSocket for live recalculation as user adds assets

---

### Feature 8.4: Lifetime Gifts Register & PETs Tracking

\*\*Feature Name:\*\* Lifetime Gifts and Potentially Exempt Transfers Management

\*\*User Story:\*\*

As a user planning to reduce my estate through gifting, I want to track all lifetime gifts and understand their IHT implications, including the 7-year rule and taper relief, so I can manage my gift planning effectively.

\*\*Acceptance Criteria:\*\*

- Record all lifetime gifts with date, recipient, and value

- Track Potentially Exempt Transfers (PETs)

- Track Chargeable Lifetime Transfers (CLTs) to trusts

- Monitor 7-year clock for each gift

- Calculate taper relief if death within 7 years

- Track annual exemption usage (£3,000 per year)

- Track small gifts exemption (£250 per person)

- Track wedding/civil partnership gifts exemptions

- Track gifts out of income exemption

- Alert when gifts may become chargeable

- Integration with estate calculation

\*\*Technical Requirements:\*\*

- Gift tracking with date-based calculations

- 7-year running total calculator

- Taper relief calculator

- Exemption tracker (annual, small gifts, normal expenditure)

- Gift categorization engine

- PET/CLT classification logic

- Integration with IHT calculation

\*\*Constraints:\*\*

- PET becomes exempt if donor survives 7 years

- Taper relief: Reduces tax (not value) if death 3-7 years after gift

- Annual exemption: £3,000 per year (can carry forward 1 year unused)

- Small gifts: £250 per person per year (not if used annual exemption)

- Wedding gifts: £5,000 (child), £2,500 (grandchild), £1,000 (other)

- Normal expenditure out of income: Must be regular and leave donor with adequate income

- CLT: Immediate IHT charge if exceeds NRB

\*\*Implementation Approach:\*\*

```pseudo

ENDPOINT: POST /api/v1/iht/gifts

REQUEST BODY:

{

giftDate: date,

recipient: {

name: string,

relationship: enum['SPOUSE', 'CHILD', 'GRANDCHILD', 'PARENT', 'SIBLING',

'FRIEND', 'CHARITY', 'TRUST', 'OTHER'],

age: integer (optional)

},

giftType: enum['CASH', 'PROPERTY', 'SHARES', 'OTHER\_ASSET'],

giftDescription: string,

giftValue: decimal,

currency: string,

giftCategory: enum['OUTRIGHT\_GIFT', 'GIFT\_TO\_TRUST', 'GIFT\_WITH\_RESERVATION'],

trustDetails: {

trustName: string,

trustType: enum['BARE', 'DISCRETIONARY', 'INTEREST\_IN\_POSSESSION'],

beneficiaries: [string]

},

exemptions: {

annualExemption: {

claimCurrent: boolean,

claimPreviousYear: boolean,

amountClaimed: decimal

},

smallGiftsExemption: boolean, // £250 per person

weddingGiftExemption: {

applicable: boolean,

amount: decimal // £5000/£2500/£1000

},

normalExpenditureOutOfIncome: boolean,

spouseExemption: boolean, // Unlimited if spouse

charityExemption: boolean // Unlimited to charity

},

reservationOfBenefit: {

hasReservation: boolean,

reservationDetails: text

},

notes: text

}

BUSINESS LOGIC:

1. Validate gift data:

- Gift date <= today

- Gift value > 0

- Recipient details provided

- Gift date not more than 7 years in past (for tracking purposes)

2. Classify gift (PET vs CLT):

IF recipient.relationship = 'SPOUSE':

classification = 'SPOUSE\_EXEMPT'

potentially\_exempt = FALSE

immediately\_exempt = TRUE

ELSE IF recipient.relationship = 'CHARITY':

classification = 'CHARITY\_EXEMPT'

potentially\_exempt = FALSE

immediately\_exempt = TRUE

ELSE IF giftCategory = 'OUTRIGHT\_GIFT':

classification = 'PET' // Potentially Exempt Transfer

potentially\_exempt = TRUE

immediately\_exempt = FALSE

ELSE IF giftCategory = 'GIFT\_TO\_TRUST':

IF trustType = 'BARE':

classification = 'PET'

potentially\_exempt = TRUE

ELSE: // Discretionary or IIP trusts

classification = 'CLT' // Chargeable Lifetime Transfer

potentially\_exempt = FALSE

calculate\_immediate\_iht\_charge()

3. Apply exemptions:

gift\_value\_before\_exemptions = gift\_value

// Spouse exemption (unlimited)

IF exemptions.spouseExemption:

taxable\_value = 0

RETURN // Fully exempt

// Charity exemption (unlimited)

IF exemptions.charityExemption:

taxable\_value = 0

RETURN // Fully exempt

// Annual exemption (£3,000 per year)

IF exemptions.annualExemption.claimCurrent:

current\_year\_exemption\_used = get\_annual\_exemption\_used(gift\_date.year)

current\_year\_exemption\_available = MAX(3000 - current\_year\_exemption\_used, 0)

exemption\_applied = MIN(gift\_value, current\_year\_exemption\_available)

gift\_value -= exemption\_applied

// Carry forward previous year's unused exemption

IF exemptions.annualExemption.claimPreviousYear AND gift\_value > 0:

previous\_year\_unused = get\_previous\_year\_unused\_exemption(gift\_date.year - 1)

previous\_exemption\_applied = MIN(gift\_value, previous\_year\_unused)

gift\_value -= previous\_exemption\_applied

// Small gifts exemption (£250 per person per year)

IF exemptions.smallGiftsExemption:

IF gift\_value <= 250 AND not\_already\_claimed\_annual\_exemption\_for\_recipient:

gift\_value = 0 // Fully exempt

// Wedding gift exemption

IF exemptions.weddingGiftExemption.applicable:

IF recipient.relationship = 'CHILD':

wedding\_exemption = MIN(5000, gift\_value)

ELSE IF recipient.relationship = 'GRANDCHILD':

wedding\_exemption = MIN(2500, gift\_value)

ELSE:

wedding\_exemption = MIN(1000, gift\_value)

gift\_value -= wedding\_exemption

// Normal expenditure out of income

IF exemptions.normalExpenditureOutOfIncome:

// This is a qualitative exemption, mark for evidence

requires\_income\_evidence = TRUE

// If proven, fully exempt

IF can\_prove\_normal\_expenditure:

gift\_value = 0

taxable\_value = MAX(gift\_value, 0)

4. Check for gift with reservation:

IF reservationOfBenefit.hasReservation:

// Gift with reservation treated as still in estate

gift\_with\_reservation = TRUE

alert = "This gift may remain in your estate for IHT purposes"

5. Calculate 7-year status:

years\_since\_gift = (today - gift\_date) / 365

years\_remaining = MAX(7 - years\_since\_gift, 0)

IF years\_remaining = 0:

pet\_status = 'EXEMPT' // Survived 7 years

ELSE:

pet\_status = 'POTENTIALLY\_EXEMPT'

pet\_becomes\_exempt\_date = gift\_date + 7\_years

6. Calculate potential IHT on gift if death now:

// Only relevant for PETs or CLTs

IF classification IN ['PET', 'CLT']:

// Check 7-year cumulation

gifts\_in\_previous\_7\_years = get\_gifts\_in\_previous\_7\_years(gift\_date)

cumulative\_total = SUM(gifts\_in\_previous\_7\_years) + taxable\_value

// Apply NRB

nrb\_at\_gift\_date = 325000 // Use historical NRB if older gift

IF cumulative\_total > nrb\_at\_gift\_date:

excess = cumulative\_total - nrb\_at\_gift\_date

potential\_iht = excess \* 0.20 // 20% lifetime rate (half death rate)

// Taper relief if death 3-7 years after gift

IF years\_since\_gift >= 3 AND years\_since\_gift < 7:

taper\_relief\_percentage = calculate\_taper\_relief(years\_since\_gift)

potential\_iht = potential\_iht \* (1 - taper\_relief\_percentage)

ELSE:

potential\_iht = 0

7. Calculate taper relief:

calculate\_taper\_relief(years\_since\_gift):

IF years\_since\_gift < 3:

RETURN 0 // No taper relief

ELSE IF years\_since\_gift >= 3 AND years\_since\_gift < 4:

RETURN 0.20 // 20% relief

ELSE IF years\_since\_gift >= 4 AND years\_since\_gift < 5:

RETURN 0.40 // 40% relief

ELSE IF years\_since\_gift >= 5 AND years\_since\_gift < 6:

RETURN 0.60 // 60% relief

ELSE IF years\_since\_gift >= 6 AND years\_since\_gift < 7:

RETURN 0.80 // 80% relief

ELSE:

RETURN 1.00 // 100% relief (exempt)

8. Integration with estate calculation:

// Gifts within 7 years added back to estate

IF pet\_status = 'POTENTIALLY\_EXEMPT':

add\_to\_estate\_calculation(taxable\_value, potential\_iht)

9. Currency conversion:

gift\_value\_gbp = convert\_to\_gbp(gift\_value, currency, gift\_date)

gift\_value\_zar = convert\_to\_zar(gift\_value, currency, gift\_date)

RESPONSE:

{

id: uuid,

giftDetails: {...},

classification: {

type: enum['PET', 'CLT', 'SPOUSE\_EXEMPT', 'CHARITY\_EXEMPT'],

potentiallyExempt: boolean,

immediatelyExempt: boolean

},

exemptions: {

applied: [

{type: string, amount: decimal}

],

totalExemptions: decimal,

taxableValue: decimal

},

sevenYearStatus: {

yearsElapsed: decimal,

yearsRemaining: decimal,

becomesExemptDate: date,

currentStatus: enum['POTENTIALLY\_EXEMPT', 'EXEMPT', 'CHARGEABLE']

},

potentialIht: {

ifDeathToday: decimal,

taperRelief: {

applicable: boolean,

percentage: decimal,

reliefAmount: decimal

},

cumulativeGifts: decimal

},

alerts: [

{severity: string, message: string}

],

recommendations: [

{message: string}

]

}

**User Flow:**

[IHT Dashboard] → [Gifts Register Tab]

↓

[Gifts Overview]

- Total gifts in last 7 years

- PETs still within 7-year period

- Gifts becoming exempt soon

- Annual exemption used/available

↓

[Add Gift Button]

↓

[Gift Entry - Step 1: When & To Whom]

- Gift date

- Recipient name

- Relationship to you

↓

[Gift Entry - Step 2: What Was Given]

- Gift type (cash, property, shares, etc.)

- Description

- Value and currency

↓

[Gift Entry - Step 3: Gift Structure]

- Outright gift OR

- Gift to trust → Trust details

- Did you keep any benefit? (reservation)

↓

[Gift Entry - Step 4: Exemptions]

System suggests applicable exemptions:

- Spouse? → Unlimited exemption

- Charity? → Unlimited exemption

- Use annual exemption (£3,000)? → Check available

- Small gift (≤£250)?

- Wedding gift?

- Normal expenditure out of income?

↓

[Exemption Calculator]

- Shows gift value before exemptions

- Shows exemptions applied

- Shows taxable value after exemptions

↓

[7-Year Timeline Display]

- Current date

- Gift date marked

- 7-year countdown

- Date gift becomes exempt

- Timeline visualization

↓

[Potential IHT Calculation]

- "If you died today, potential IHT: £X"

- Taper relief: £X (if 3-7 years)

- Cumulative gifts: £X

↓

[Save Gift]

↓

[Gifts List View]

- Timeline view: All gifts on 7-year timeline

- Card view: Each gift with key details

- Filter: By status, recipient, year

- Sort: By date, value, years remaining

- Color coding:

- Red: Within 3 years (no taper)

- Amber: 3-7 years (partial taper)

- Green: >7 years (exempt)

↓

[Gift Detail View]

- Complete gift information

- 7-year timeline

- Exemptions breakdown

- Potential IHT calculation

- Edit/Delete options

↓

[Annual Exemption Tracker]

- Current year: £X used of £3,000

- Previous year: £X unused (can carry forward)

- Historical usage chart

↓

[Gift Planning Tools]

- "Plan a Gift" wizard

- Calculates optimal gift strategy

- Shows impact on estate

**API Endpoints:**

* POST /api/v1/iht/gifts - Record gift
* PUT /api/v1/iht/gifts/{id} - Update gift
* DELETE /api/v1/iht/gifts/{id} - Delete gift (soft delete)
* GET /api/v1/iht/gifts - List all gifts
* GET /api/v1/iht/gifts/{id} - Get specific gift
* GET /api/v1/iht/gifts/seven-year-summary - 7-year gifts summary
* GET /api/v1/iht/gifts/annual-exemption - Annual exemption status
* POST /api/v1/iht/gifts/calculate-iht - Calculate IHT on gifts
* GET /api/v1/iht/gifts/timeline - Get gifts timeline visualization
* POST /api/v1/iht/gifts/planning-tool - Gift planning calculator

**Data Models:**

TABLE: lifetime\_gifts

- id: UUID (PK)

- user\_id: UUID (FK to users)

- gift\_date: DATE

- recipient\_name: VARCHAR(255)

- recipient\_relationship: ENUM('SPOUSE', 'CHILD', 'GRANDCHILD', 'PARENT', 'SIBLING',

'FRIEND', 'CHARITY', 'TRUST', 'OTHER')

- gift\_type: ENUM('CASH', 'PROPERTY', 'SHARES', 'OTHER\_ASSET')

- gift\_description: TEXT

- gift\_value: DECIMAL(15,2)

- currency: CHAR(3)

- gift\_value\_gbp: DECIMAL(15,2) (calculated)

- gift\_value\_zar: DECIMAL(15,2) (calculated)

- gift\_category: ENUM('OUTRIGHT\_GIFT', 'GIFT\_TO\_TRUST', 'GIFT\_WITH\_RESERVATION')

- classification: ENUM('PET', 'CLT', 'SPOUSE\_EXEMPT', 'CHARITY\_EXEMPT')

- potentially\_exempt: BOOLEAN

- becomes\_exempt\_date: DATE (gift\_date + 7 years)

- pet\_status: ENUM('POTENTIALLY\_EXEMPT', 'EXEMPT', 'CHARGEABLE')

- deleted: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

TABLE: gift\_trust\_details

- gift\_id: UUID (PK, FK to lifetime\_gifts)

- trust\_name: VARCHAR(255)

- trust\_type: ENUM('BARE', 'DISCRETIONARY', 'INTEREST\_IN\_POSSESSION')

- beneficiaries: JSON (array)

- trust\_deed\_reference: UUID (FK to documents)

- created\_at: TIMESTAMP

TABLE: gift\_exemptions

- id: UUID (PK)

- gift\_id: UUID (FK to lifetime\_gifts)

- exemption\_type: ENUM('ANNUAL\_EXEMPTION\_CURRENT', 'ANNUAL\_EXEMPTION\_PREVIOUS',

'SMALL\_GIFTS', 'WEDDING', 'NORMAL\_EXPENDITURE', 'SPOUSE', 'CHARITY')

- exemption\_amount: DECIMAL(15,2)

- exemption\_year: INTEGER (for annual exemption tracking)

- created\_at: TIMESTAMP

TABLE: gift\_reservation\_of\_benefit

- gift\_id: UUID (PK, FK to lifetime\_gifts)

- has\_reservation: BOOLEAN

- reservation\_details: TEXT

- gift\_with\_reservation\_flag: BOOLEAN

- created\_at: TIMESTAMP

TABLE: gift\_iht\_calculations

- gift\_id: UUID (PK, FK to lifetime\_gifts)

- taxable\_value: DECIMAL(15,2)

- cumulative\_gifts\_7\_years: DECIMAL(15,2)

- nil\_rate\_band\_used: DECIMAL(15,2)

- potential\_iht\_if\_death\_today: DECIMAL(15,2)

- taper\_relief\_applicable: BOOLEAN

- taper\_relief\_percentage: DECIMAL(5,2)

- taper\_relief\_amount: DECIMAL(15,2)

- iht\_after\_taper: DECIMAL(15,2)

- last\_calculated: TIMESTAMP

TABLE: annual\_exemption\_tracking

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: INTEGER

- annual\_exemption\_limit: DECIMAL(10,2) DEFAULT 3000

- exemption\_used: DECIMAL(10,2)

- exemption\_available: DECIMAL(10,2)

- carry\_forward\_from\_previous: DECIMAL(10,2)

- created\_at: TIMESTAMP

TABLE: gift\_planning\_scenarios

- id: UUID (PK)

- user\_id: UUID (FK to users)

- scenario\_name: VARCHAR(255)

- planned\_gifts: JSON (array of planned gifts)

- projected\_estate\_reduction: DECIMAL(15,2)

- projected\_iht\_saving: DECIMAL(15,2)

- created\_at: TIMESTAMP

VIEW: v\_active\_pets (gifts within 7 years)

- user\_id

- gift\_id

- gift\_date

- recipient\_name

- gift\_value\_gbp

- taxable\_value

- years\_elapsed

- years\_remaining

- becomes\_exempt\_date

- potential\_iht

INDEX on lifetime\_gifts(user\_id, gift\_date DESC)

INDEX on lifetime\_gifts(user\_id, pet\_status, becomes\_exempt\_date)

INDEX on gift\_exemptions(gift\_id, exemption\_type)

INDEX on annual\_exemption\_tracking(user\_id, tax\_year)

**Error Handling:**

ERROR CASES:

1. Gift date in future

- Response: 400 Bad Request

- Message: "Gift date cannot be in the future"

2. Gift date more than 7 years ago

- Response: 400 Bad Request (warning)

- Message: "Gift is more than 7 years old and should be exempt. Still record?"

- Allow with confirmation (for historical records)

3. Annual exemption exceeded

- Response: 400 Bad Request (warning)

- Message: "Annual exemption of £{available} exceeded. Claim amount: £{claimed}"

- Allow with warning that excess is taxable

4. Small gifts exemption misapplied

- Response: 400 Bad Request

- Message: "Small gifts exemption (£250) cannot be used with annual exemption for same recipient"

5. Spouse/charity exemption with other exemptions

- Response: 400 Bad Request (warning)

- Message: "Spouse/charity gifts are fully exempt. Other exemptions not needed"

6. Normal expenditure out of income without evidence

- Response: 200 OK (with alert)

- Alert: "Normal expenditure exemption requires evidence of regular gifts from income. Ensure records kept"

7. Wedding gift exemption without wedding context

- Response: 400 Bad Request

- Message: "Wedding gift exemption requires wedding/civil partnership date"

8. Gift to discretionary trust exceeding NRB

- Response: 200 OK (with alert)

- Alert: "CLT of £{amount} exceeds NRB. Immediate IHT charge of £{tax} may apply"

EDGE CASES:

- Gift to spouse who is non-UK domiciled: Limited to £325,000

- Gift to charity via will (not lifetime): Not a PET, handled in will planning

- Gift of property with mortgage: Value = property value - mortgage

- Gift of shares: Valuation on date of gift

- Gift of business asset: May qualify for BPR (50% or 100%)

- Gift of agricultural land: May qualify for APR

- Gifts to political parties: Exempt if party meets criteria

- Gifts to museums/galleries: Exempt if qualifying

- Gifts to housing associations: Exempt

- Conditional exemption for heritage assets: Complex rules

- Potentially exempt transfer failing: Becomes chargeable if death within 7 years

- CLT to discretionary trust: Periodic charges every 10 years

- Exit charges from trusts: Complex trust taxation

- Gift and leaseback: May be gift with reservation

- Gift of property but continue living there: Gift with reservation

- Gift with reservation released before death: May still be in estate

- Failed PETs: Tax calculated at death, not gift date

- Taper relief: Reduces tax, not value

- Cumulation: Gifts cumulate backwards 7 years from each gift

- Multiple gifts same day: Treated as single gift for cumulation

- Death within 7 years of multiple PETs: All become chargeable

- Quick succession relief: If recipient dies within 5 years

**Performance Considerations:**

* 7-year calculation: Simple date arithmetic, <10ms
* Cumulative gifts calculation: Query last 7 years, <50ms
* Annual exemption lookup: Index on tax\_year, <20ms
* IHT calculation on gifts: More complex, cache results
* Timeline visualization: Pre-calculate data points
* Expected gifts per user: 5-50 over lifetime
* Gifts list query: <500ms with filters
* Background job: Daily check for gifts becoming exempt
* Alert generation: When gift enters final year before exemption
* Integration with estate calc: Real-time for active PETs

**9. TAX INTELLIGENCE ENGINE**

**Feature 9.1: Core Tax Calculation Service**

**Feature Name:** Multi-Jurisdiction Tax Calculation Engine

**User Story:** As the system, I need a centralized tax calculation engine that can accurately compute income tax, capital gains tax, dividend tax, and other taxes for both UK and SA jurisdictions, applying the correct rates, bands, and allowances based on user circumstances.

**Acceptance Criteria:**

* Calculate UK Income Tax (including Scottish rates where applicable)
* Calculate SA Income Tax (PAYE and provisional tax)
* Calculate UK Capital Gains Tax with annual exemption
* Calculate SA Capital Gains Tax (inclusion rate method)
* Calculate UK Dividend Tax with dividend allowance
* Calculate SA Dividend Withholding Tax
* Apply correct tax bands and rates for each jurisdiction
* Handle personal allowance tapering (UK)
* Support rebates and tax credits (SA)
* Version control for historical tax rates
* Real-time calculation API

**Technical Requirements:**

* Tax rules engine with versioned rate tables
* Progressive tax band calculator
* Allowance and relief calculator
* High-precision decimal arithmetic
* Historical tax rate database
* Calculation audit trail
* Performance optimization for multiple calculations
* Stateless calculation service

**Constraints:**

* Tax rates change annually (April for UK, March for SA)
* Must support historical calculations for past tax years
* Calculations must be reproducible (audit trail)
* Performance: <50ms per calculation
* Precision: 2 decimal places for currency
* Must handle edge cases (£0 income, negative adjustments)

**Implementation Approach:**

SERVICE: TaxCalculationService

CLASS TaxCalculationService:

# ===== UK INCOME TAX =====

FUNCTION calculate\_uk\_income\_tax(

income: decimal,

tax\_year: string,

scotland\_resident: boolean,

personal\_allowance\_restriction: decimal = 0

) -> UkIncomeTaxResult:

# Get tax year configuration

config = get\_uk\_tax\_config(tax\_year)

# Personal Allowance calculation

personal\_allowance = config.personal\_allowance # £12,570 for 2024/25

# Taper personal allowance if income > £100,000

IF income > 100000:

taper = (income - 100000) / 2

personal\_allowance = MAX(personal\_allowance - taper, 0)

# Apply any additional restrictions

personal\_allowance = MAX(personal\_allowance - personal\_allowance\_restriction, 0)

# Taxable income

taxable\_income = MAX(income - personal\_allowance, 0)

# Get appropriate tax bands (England/Wales/NI or Scotland)

IF scotland\_resident:

bands = config.scottish\_income\_tax\_bands

ELSE:

bands = config.uk\_income\_tax\_bands

# Example bands for England/Wales 2024/25:

# £0 - £37,700: 20% (Basic rate)

# £37,701 - £125,140: 40% (Higher rate)

# £125,141+: 45% (Additional rate)

# Calculate tax by band

tax\_by\_band = []

remaining\_income = taxable\_income

cumulative\_threshold = 0

FOR EACH band IN bands:

band\_size = band.upper\_limit - cumulative\_threshold

IF remaining\_income <= 0:

BREAK

IF band.upper\_limit = INFINITY: # Top band

taxable\_in\_band = remaining\_income

ELSE:

taxable\_in\_band = MIN(remaining\_income, band\_size)

tax\_in\_band = taxable\_in\_band \* band.rate

tax\_by\_band.append({

band\_name: band.name,

taxable\_amount: taxable\_in\_band,

rate: band.rate,

tax: tax\_in\_band

})

remaining\_income -= taxable\_in\_band

cumulative\_threshold = band.upper\_limit

total\_tax = SUM(tax\_by\_band[].tax)

# Calculate effective rate

effective\_rate = (total\_tax / income) \* 100 IF income > 0 ELSE 0

# Calculate marginal rate

marginal\_rate = determine\_marginal\_rate(income, bands)

RETURN {

gross\_income: income,

personal\_allowance: personal\_allowance,

taxable\_income: taxable\_income,

tax\_by\_band: tax\_by\_band,

total\_tax: total\_tax,

net\_income: income - total\_tax,

effective\_rate: effective\_rate,

marginal\_rate: marginal\_rate,

tax\_year: tax\_year

}

# ===== UK NATIONAL INSURANCE =====

FUNCTION calculate\_uk\_national\_insurance(

income: decimal,

tax\_year: string,

employment\_type: enum['EMPLOYED', 'SELF\_EMPLOYED']

) -> UkNationalInsuranceResult:

config = get\_uk\_tax\_config(tax\_year)

IF employment\_type = 'EMPLOYED':

# Class 1 NI (Employees)

# 2024/25: 8% on £12,570 - £50,270, 2% above

primary\_threshold = config.ni\_primary\_threshold # £12,570

upper\_earnings\_limit = config.ni\_upper\_earnings\_limit # £50,270

ni\_class\_1 = 0

IF income > primary\_threshold:

# 8% band

band\_1\_income = MIN(income - primary\_threshold,

upper\_earnings\_limit - primary\_threshold)

ni\_class\_1 += band\_1\_income \* 0.08

# 2% band (above UEL)

IF income > upper\_earnings\_limit:

band\_2\_income = income - upper\_earnings\_limit

ni\_class\_1 += band\_2\_income \* 0.02

RETURN {

gross\_income: income,

class\_1\_ni: ni\_class\_1,

total\_ni: ni\_class\_1,

effective\_rate: (ni\_class\_1 / income) \* 100

}

ELSE IF employment\_type = 'SELF\_EMPLOYED':

# Class 2 and Class 4 NI

class\_2\_threshold = config.ni\_class\_2\_threshold # £12,570

class\_4\_lower\_limit = config.ni\_class\_4\_lower # £12,570

class\_4\_upper\_limit = config.ni\_class\_4\_upper # £50,270

# Class 2: Flat rate if profits > threshold

class\_2\_ni = 0

IF income > class\_2\_threshold:

class\_2\_ni = config.ni\_class\_2\_weekly\_rate \* 52 # £3.45/week \* 52

# Class 4: 6% on £12,570 - £50,270, 2% above

class\_4\_ni = 0

IF income > class\_4\_lower\_limit:

band\_1\_income = MIN(income - class\_4\_lower\_limit,

class\_4\_upper\_limit - class\_4\_lower\_limit)

class\_4\_ni += band\_1\_income \* 0.06

IF income > class\_4\_upper\_limit:

band\_2\_income = income - class\_4\_upper\_limit

class\_4\_ni += band\_2\_income \* 0.02

total\_ni = class\_2\_ni + class\_4\_ni

RETURN {

gross\_income: income,

class\_2\_ni: class\_2\_ni,

class\_4\_ni: class\_4\_ni,

total\_ni: total\_ni,

effective\_rate: (total\_ni / income) \* 100

}

# ===== UK CAPITAL GAINS TAX =====

FUNCTION calculate\_uk\_capital\_gains\_tax(

capital\_gain: decimal,

asset\_type: enum['RESIDENTIAL\_PROPERTY', 'OTHER'],

tax\_year: string,

income: decimal,

basic\_rate\_band\_remaining: decimal

) -> UkCapitalGainsTaxResult:

config = get\_uk\_tax\_config(tax\_year)

# Annual Exempt Amount (AEA)

annual\_exemption = config.cgt\_annual\_exemption # £3,000 for 2024/25

taxable\_gain = MAX(capital\_gain - annual\_exemption, 0)

# CGT rates depend on asset type and tax band

IF asset\_type = 'RESIDENTIAL\_PROPERTY':

basic\_rate\_cgt = 0.18

higher\_rate\_cgt = 0.24

ELSE: # Other assets

basic\_rate\_cgt = 0.10

higher\_rate\_cgt = 0.20

# Determine how much gain falls in basic rate band

gain\_in\_basic\_rate = MIN(taxable\_gain, basic\_rate\_band\_remaining)

gain\_in\_higher\_rate = MAX(taxable\_gain - basic\_rate\_band\_remaining, 0)

tax\_at\_basic\_rate = gain\_in\_basic\_rate \* basic\_rate\_cgt

tax\_at\_higher\_rate = gain\_in\_higher\_rate \* higher\_rate\_cgt

total\_cgt = tax\_at\_basic\_rate + tax\_at\_higher\_rate

RETURN {

capital\_gain: capital\_gain,

annual\_exemption\_used: MIN(capital\_gain, annual\_exemption),

taxable\_gain: taxable\_gain,

gain\_at\_basic\_rate: gain\_in\_basic\_rate,

gain\_at\_higher\_rate: gain\_in\_higher\_rate,

tax\_at\_basic\_rate: tax\_at\_basic\_rate,

tax\_at\_higher\_rate: tax\_at\_higher\_rate,

total\_cgt: total\_cgt,

effective\_rate: (total\_cgt / capital\_gain) \* 100 IF capital\_gain > 0 ELSE 0

}

# ===== UK DIVIDEND TAX =====

FUNCTION calculate\_uk\_dividend\_tax(

dividend\_income: decimal,

tax\_year: string,

other\_income: decimal,

basic\_rate\_band\_remaining: decimal

) -> UkDividendTaxResult:

config = get\_uk\_tax\_config(tax\_year)

# Dividend Allowance

dividend\_allowance = config.dividend\_allowance # £500 for 2024/25

taxable\_dividends = MAX(dividend\_income - dividend\_allowance, 0)

# Dividend tax rates (2024/25):

# Basic rate: 8.75%

# Higher rate: 33.75%

# Additional rate: 39.35%

# Determine tax band utilization

dividends\_at\_basic\_rate = MIN(taxable\_dividends, basic\_rate\_band\_remaining)

# Calculate remaining income after basic rate band used

remaining\_after\_basic = MAX(taxable\_dividends - dividends\_at\_basic\_rate, 0)

# Higher rate band threshold (£125,140 - £37,700 = £87,440)

higher\_rate\_band\_size = 125140 - 37700

income\_in\_higher\_band = other\_income - 37700

higher\_rate\_band\_remaining = MAX(higher\_rate\_band\_size - income\_in\_higher\_band, 0)

dividends\_at\_higher\_rate = MIN(remaining\_after\_basic, higher\_rate\_band\_remaining)

dividends\_at\_additional\_rate = MAX(remaining\_after\_basic - dividends\_at\_higher\_rate, 0)

# Calculate tax

tax\_at\_basic\_rate = dividends\_at\_basic\_rate \* 0.0875

tax\_at\_higher\_rate = dividends\_at\_higher\_rate \* 0.3375

tax\_at\_additional\_rate = dividends\_at\_additional\_rate \* 0.3935

total\_dividend\_tax = tax\_at\_basic\_rate + tax\_at\_higher\_rate + tax\_at\_additional\_rate

RETURN {

dividend\_income: dividend\_income,

dividend\_allowance\_used: MIN(dividend\_income, dividend\_allowance),

taxable\_dividends: taxable\_dividends,

dividends\_at\_basic\_rate: dividends\_at\_basic\_rate,

dividends\_at\_higher\_rate: dividends\_at\_higher\_rate,

dividends\_at\_additional\_rate: dividends\_at\_additional\_rate,

tax\_at\_basic\_rate: tax\_at\_basic\_rate,

tax\_at\_higher\_rate: tax\_at\_higher\_rate,

tax\_at\_additional\_rate: tax\_at\_additional\_rate,

total\_tax: total\_dividend\_tax,

effective\_rate: (total\_dividend\_tax / dividend\_income) \* 100 IF dividend\_income > 0 ELSE 0

}

# ===== SA INCOME TAX =====

FUNCTION calculate\_sa\_income\_tax(

taxable\_income: decimal,

tax\_year: string,

age\_group: enum['UNDER\_65', '65\_TO\_74', '75\_PLUS']

) -> SaIncomeTaxResult:

config = get\_sa\_tax\_config(tax\_year)

# SA Income Tax Bands (2024/2025):

# R0 - R237,100: 18%

# R237,101 - R370,500: 26%

# R370,501 - R512,800: 31%

# R512,801 - R673,000: 36%

# R673,001 - R857,900: 39%

# R857,901+: 45%

bands = config.income\_tax\_bands

# Calculate tax by band

tax\_by\_band = []

remaining\_income = taxable\_income

FOR EACH band IN bands:

IF remaining\_income <= 0:

BREAK

IF band.upper\_limit = INFINITY:

taxable\_in\_band = remaining\_income

ELSE:

band\_size = band.upper\_limit - band.lower\_limit

taxable\_in\_band = MIN(remaining\_income, band\_size)

tax\_in\_band = taxable\_in\_band \* band.rate

tax\_by\_band.append({

band\_name: band.name,

taxable\_amount: taxable\_in\_band,

rate: band.rate,

tax: tax\_in\_band

})

remaining\_income -= taxable\_in\_band

gross\_tax = SUM(tax\_by\_band[].tax)

# Apply Primary Rebate (age-dependent)

IF age\_group = 'UNDER\_65':

primary\_rebate = config.primary\_rebate # R17,235 for 2024/25

ELSE IF age\_group = '65\_TO\_74':

primary\_rebate = config.secondary\_rebate # R19,500

ELSE: # 75+

primary\_rebate = config.tertiary\_rebate # R21,720

# Tax payable after rebate

tax\_payable = MAX(gross\_tax - primary\_rebate, 0)

# Calculate effective rate

effective\_rate = (tax\_payable / taxable\_income) \* 100 IF taxable\_income > 0 ELSE 0

# Determine marginal rate

marginal\_rate = determine\_sa\_marginal\_rate(taxable\_income, bands)

RETURN {

taxable\_income: taxable\_income,

tax\_by\_band: tax\_by\_band,

gross\_tax: gross\_tax,

primary\_rebate: primary\_rebate,

tax\_payable: tax\_payable,

net\_income: taxable\_income - tax\_payable,

effective\_rate: effective\_rate,

marginal\_rate: marginal\_rate,

tax\_year: tax\_year

}

# ===== SA CAPITAL GAINS TAX =====

FUNCTION calculate\_sa\_capital\_gains\_tax(

capital\_gain: decimal,

tax\_year: string,

taxable\_income: decimal

) -> SaCapitalGainsTaxResult:

config = get\_sa\_tax\_config(tax\_year)

# SA CGT uses inclusion rate method

# Annual exclusion: R40,000 (2024/25)

# Inclusion rate: 40% for individuals

annual\_exclusion = config.cgt\_annual\_exclusion # R40,000

# Apply annual exclusion

gain\_after\_exclusion = MAX(capital\_gain - annual\_exclusion, 0)

# Apply inclusion rate (40%)

inclusion\_rate = 0.40

taxable\_capital\_gain = gain\_after\_exclusion \* inclusion\_rate

# This taxable gain is added to income and taxed at marginal rate

combined\_taxable\_income = taxable\_income + taxable\_capital\_gain

# Calculate tax on combined income

tax\_on\_combined = calculate\_sa\_income\_tax(

combined\_taxable\_income,

tax\_year,

age\_group

).tax\_payable

# Calculate tax on income only

tax\_on\_income\_only = calculate\_sa\_income\_tax(

taxable\_income,

tax\_year,

age\_group

).tax\_payable

# CGT is the difference

cgt = tax\_on\_combined - tax\_on\_income\_only

RETURN {

capital\_gain: capital\_gain,

annual\_exclusion\_used: MIN(capital\_gain, annual\_exclusion),

gain\_after\_exclusion: gain\_after\_exclusion,

inclusion\_rate: inclusion\_rate,

taxable\_capital\_gain: taxable\_capital\_gain,

cgt: cgt,

effective\_rate: (cgt / capital\_gain) \* 100 IF capital\_gain > 0 ELSE 0

}

# ===== SA DIVIDEND TAX =====

FUNCTION calculate\_sa\_dividend\_tax(

dividend\_income: decimal,

dividend\_type: enum['LOCAL', 'FOREIGN'],

tax\_year: string

) -> SaDividendTaxResult:

config = get\_sa\_tax\_config(tax\_year)

# SA Dividend Tax (2024/25):

# Local dividends: 20% Dividends Tax (withheld at source)

# Foreign dividends: Included in income, taxed at marginal rate

IF dividend\_type = 'LOCAL':

# Dividends Tax withheld at source (20%)

dividend\_tax = dividend\_income \* 0.20

net\_dividend = dividend\_income - dividend\_tax

# Exemptions (first R23,800 exempt for individuals under 65)

# But this is interest exemption, not dividend

# Dividends Tax applies regardless

RETURN {

gross\_dividend: dividend\_income,

dividend\_tax\_rate: 0.20,

dividend\_tax\_withheld: dividend\_tax,

net\_dividend: net\_dividend,

included\_in\_taxable\_income: FALSE

}

ELSE: # FOREIGN

# Foreign dividends included in taxable income

# Taxed at marginal rate

# May qualify for foreign tax credit under DTA

RETURN {

gross\_dividend: dividend\_income,

included\_in\_taxable\_income: TRUE,

taxed\_at\_marginal\_rate: TRUE,

foreign\_tax\_credit\_may\_apply: TRUE

}

# ===== HELPER FUNCTIONS =====

FUNCTION determine\_marginal\_rate(income: decimal, bands: array) -> decimal:

# Find which band the next £1 of income falls into

cumulative = 0

FOR EACH band IN bands:

IF income <= band.upper\_limit:

RETURN band.rate

cumulative = band.upper\_limit

RETURN bands[last].rate

FUNCTION get\_uk\_tax\_config(tax\_year: string) -> UkTaxConfig:

# Retrieve tax configuration for specified year

# Versioned in database

RETURN db.query("SELECT \* FROM uk\_tax\_config WHERE tax\_year = ?", tax\_year)

FUNCTION get\_sa\_tax\_config(tax\_year: string) -> SaTaxConfig:

# Retrieve SA tax configuration for specified year

RETURN db.query("SELECT \* FROM sa\_tax\_config WHERE tax\_year = ?", tax\_year)

# ===== COMPOSITE TAX CALCULATION =====

FUNCTION calculate\_total\_tax\_liability(

user\_id: uuid,

tax\_year: string

) -> CompleteTaxLiabilityResult:

# Gather all user income sources

income\_sources = get\_user\_income(user\_id, tax\_year)

# UK Tax Calculations

uk\_employment\_income = SUM(income WHERE source\_country = 'UK' AND type = 'EMPLOYMENT')

uk\_self\_employment\_income = SUM(income WHERE source\_country = 'UK' AND type = 'SELF\_EMPLOYMENT')

uk\_rental\_income = SUM(income WHERE source\_country = 'UK' AND type = 'RENTAL')

uk\_pension\_income = SUM(income WHERE source\_country = 'UK' AND type = 'PENSION')

uk\_dividend\_income = SUM(income WHERE source\_country = 'UK' AND type = 'DIVIDEND')

uk\_interest\_income = SUM(income WHERE source\_country = 'UK' AND type = 'INTEREST')

# Calculate UK Income Tax

total\_uk\_income = uk\_employment\_income + uk\_self\_employment\_income +

uk\_rental\_income + uk\_pension\_income

uk\_income\_tax = calculate\_uk\_income\_tax(total\_uk\_income, tax\_year, FALSE)

# Calculate UK NI

uk\_ni = calculate\_uk\_national\_insurance(uk\_employment\_income, tax\_year, 'EMPLOYED')

# Calculate UK Dividend Tax

uk\_dividend\_tax = calculate\_uk\_dividend\_tax(

uk\_dividend\_income,

tax\_year,

total\_uk\_income,

37700 - total\_uk\_income # Basic rate band remaining

)

# Calculate UK CGT (if any capital gains)

uk\_capital\_gains = get\_user\_capital\_gains(user\_id, tax\_year, 'UK')

uk\_cgt = calculate\_uk\_capital\_gains\_tax(

uk\_capital\_gains,

'OTHER',

tax\_year,

total\_uk\_income,

37700 - total\_uk\_income

)

# SA Tax Calculations

sa\_employment\_income = SUM(income WHERE source\_country = 'SA' AND type = 'EMPLOYMENT')

sa\_dividend\_income = SUM(income WHERE source\_country = 'SA' AND type = 'DIVIDEND')

sa\_interest\_income = SUM(income WHERE source\_country = 'SA' AND type = 'INTEREST')

total\_sa\_taxable\_income = sa\_employment\_income + sa\_interest\_income

sa\_income\_tax = calculate\_sa\_income\_tax(total\_sa\_taxable\_income, tax\_year, 'UNDER\_65')

# SA Dividend Tax (withheld at source for local dividends)

sa\_dividend\_tax = calculate\_sa\_dividend\_tax(sa\_dividend\_income, 'LOCAL', tax\_year)

# SA CGT

sa\_capital\_gains = get\_user\_capital\_gains(user\_id, tax\_year, 'SA')

sa\_cgt = calculate\_sa\_capital\_gains\_tax(sa\_capital\_gains, tax\_year, total\_sa\_taxable\_income)

# Apply DTA relief (if applicable)

dta\_relief = calculate\_dta\_relief(user\_id, uk\_taxes, sa\_taxes)

RETURN {

uk\_taxes: {

income\_tax: uk\_income\_tax.total\_tax,

national\_insurance: uk\_ni.total\_ni,

dividend\_tax: uk\_dividend\_tax.total\_tax,

capital\_gains\_tax: uk\_cgt.total\_cgt,

total: uk\_income\_tax.total\_tax + uk\_ni.total\_ni + uk\_dividend\_tax.total\_tax + uk\_cgt.total\_cgt

},

sa\_taxes: {

income\_tax: sa\_income\_tax.tax\_payable,

dividend\_tax: sa\_dividend\_tax.dividend\_tax\_withheld,

capital\_gains\_tax: sa\_cgt.cgt,

total: sa\_income\_tax.tax\_payable + sa\_dividend\_tax.dividend\_tax\_withheld + sa\_cgt.cgt

},

dta\_relief: dta\_relief,

total\_tax\_liability: (uk\_taxes.total - dta\_relief.uk\_credit) +

(sa\_taxes.total - dta\_relief.sa\_credit)

}

**API Endpoints:**

# UK Tax Calculations

POST /api/v1/tax/uk/income-tax

POST /api/v1/tax/uk/national-insurance

POST /api/v1/tax/uk/capital-gains-tax

POST /api/v1/tax/uk/dividend-tax

# SA Tax Calculations

POST /api/v1/tax/sa/income-tax

POST /api/v1/tax/sa/capital-gains-tax

POST /api/v1/tax/sa/dividend-tax

# Composite Calculations

POST /api/v1/tax/calculate-total-liability

POST /api/v1/tax/compare-scenarios

# Tax Configuration

GET /api/v1/tax/config/uk/{taxYear}

GET /api/v1/tax/config/sa/{taxYear}

# Tax Planning

POST /api/v1/tax/optimization-analysis

POST /api/v1/tax/marginal-rate-calculator

**Data Models:**

TABLE: uk\_tax\_config

- tax\_year: VARCHAR(7) (PK) (e.g., '2024/25')

- personal\_allowance: DECIMAL(10,2)

- basic\_rate\_band\_upper: DECIMAL(10,2)

- higher\_rate\_band\_upper: DECIMAL(10,2)

- basic\_rate: DECIMAL(5,4)

- higher\_rate: DECIMAL(5,4)

- additional\_rate: DECIMAL(5,4)

- cgt\_annual\_exemption: DECIMAL(10,2)

- cgt\_basic\_rate\_other: DECIMAL(5,4)

- cgt\_higher\_rate\_other: DECIMAL(5,4)

- cgt\_basic\_rate\_property: DECIMAL(5,4)

- cgt\_higher\_rate\_property: DECIMAL(5,4)

- dividend\_allowance: DECIMAL(10,2)

- dividend\_basic\_rate: DECIMAL(5,4)

- dividend\_higher\_rate: DECIMAL(5,4)

- dividend\_additional\_rate: DECIMAL(5,4)

- ni\_primary\_threshold: DECIMAL(10,2)

- ni\_upper\_earnings\_limit: DECIMAL(10,2)

- ni\_class\_1\_rate: DECIMAL(5,4)

- ni\_class\_1\_additional\_rate: DECIMAL(5,4)

- effective\_from: DATE

- effective\_to: DATE

TABLE: scottish\_income\_tax\_bands

- tax\_year: VARCHAR(7) (FK to uk\_tax\_config)

- band\_order: INTEGER

- band\_name: VARCHAR(50)

- lower\_limit: DECIMAL(10,2)

- upper\_limit: DECIMAL(10,2)

- rate: DECIMAL(5,4)

TABLE: sa\_tax\_config

- tax\_year: VARCHAR(9) (PK) (e.g., '2024/2025')

- primary\_rebate: DECIMAL(10,2)

- secondary\_rebate: DECIMAL(10,2)

- tertiary\_rebate: DECIMAL(10,2)

- interest\_exemption\_under\_65: DECIMAL(10,2)

- interest\_exemption\_65\_plus: DECIMAL(10,2)

- cgt\_annual\_exclusion: DECIMAL(10,2)

- cgt\_inclusion\_rate: DECIMAL(5,4)

- dividend\_tax\_rate: DECIMAL(5,4)

- medical\_tax\_credit: DECIMAL(10,2)

- effective\_from: DATE

- effective\_to: DATE

TABLE: sa\_income\_tax\_bands

- tax\_year: VARCHAR(9) (FK to sa\_tax\_config)

- band\_order: INTEGER

- band\_name: VARCHAR(50)

- lower\_limit: DECIMAL(15,2)

- upper\_limit: DECIMAL(15,2)

- rate: DECIMAL(5,4)

TABLE: tax\_calculations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- calculation\_type: ENUM('UK\_INCOME\_TAX', 'SA\_INCOME\_TAX', 'UK\_CGT', 'SA\_CGT', etc.)

- tax\_year: VARCHAR(10)

- input\_data: JSON

- calculation\_result: JSON

- calculated\_at: TIMESTAMP

- calculation\_version: VARCHAR(10)

INDEX on tax\_calculations(user\_id, tax\_year, calculation\_type)

INDEX on uk\_tax\_config(tax\_year)

INDEX on sa\_tax\_config(tax\_year)

**Error Handling:**

ERROR CASES:

1. Tax year configuration not found

- Response: 404 Not Found

- Message: "Tax configuration for year {year} not available"

2. Negative income value

- Response: 400 Bad Request

- Message: "Income cannot be negative. Use losses/deductions separately"

3. Invalid tax year format

- Response: 400 Bad Request

- Message: "Invalid tax year format. Use 'YYYY/YY' for UK or 'YYYY/YYYY' for SA"

4. Historical calculation with unavailable rates

- Response: 404 Not Found

- Message: "Tax rates for {year} not available in system"

EDGE CASES:

- Income exactly at band threshold: Apply higher rate to next £1

- Personal allowance fully tapered: Calculate without allowance

- Zero income: Return zero tax, full allowances unused

- Income above additional rate threshold: All excess at top rate

- Scottish resident with mixed income: Apply Scottish rates to earned income only

- Non-resident with UK income: Different tax treatment

- Savings starting rate: First £5,000 at 0% if income low enough

- Marriage allowance transfer: 10% of personal allowance transferable

**Performance Considerations:**

* All calculations stateless: No database dependency during calc
* Tax config cached in memory: <5ms lookup
* Single calculation: Target <50ms
* Batch calculations: Optimize for bulk processing
* Precision: Use Decimal type, avoid floating point
* Audit trail: Log all calculations asynchronously
* Historical calculations: Pre-load tax configs for common years

Feature 9.2: Double Tax Agreement (DTA) Relief Calculator

**Feature Name:** UK-SA Double Tax Agreement Relief Application

**User Story:** As a user with income or assets in both UK and SA, I need the system to automatically calculate Double Tax Agreement relief so that I don't pay tax twice on the same income or gains.

**Acceptance Criteria:**

* Identify income/gains taxed in both jurisdictions
* Apply UK-SA DTA provisions correctly
* Calculate foreign tax credit
* Determine which country has primary taxing rights
* Apply tie-breaker rules for dual residents
* Handle different income types (employment, business, dividends, interest, pensions, capital gains)
* Show relief calculation breakdown
* Integration with tax calculation engine

**Technical Requirements:**

* DTA rules engine
* Source vs residence taxation logic
* Foreign tax credit calculator
* Tie-breaker rules for dual residency
* Income categorization by DTA article
* Relief limitation calculator (cannot exceed tax on that income)

**Constraints:**

* UK-SA DTA: Effective from 2002
* Relief limited to lower of: tax paid abroad or tax due in residence country
* Different rules for different income types
* Residency determined by DTA tie-breaker if dual resident
* Must track which country taxes first (source vs residence)

**Implementation Approach:**

SERVICE: DtaReliefCalculator

FUNCTION calculate\_dta\_relief(

user: User,

income\_items: array[IncomeItem],

tax\_year: string

) -> DtaReliefResult:

# 1. Determine tax residency under DTA

dta\_residence = determine\_dta\_residence(user)

# 2. Categorize income by DTA treatment

categorized\_income = categorize\_income\_by\_dta\_article(income\_items)

# 3. Calculate relief for each income type

relief\_by\_income\_type = []

FOR EACH income\_category IN categorized\_income:

relief = calculate\_relief\_for\_category(

income\_category,

dta\_residence,

user.uk\_tax\_resident,

user.sa\_tax\_resident

)

relief\_by\_income\_type.append(relief)

# 4. Aggregate total relief

total\_uk\_relief = SUM(relief\_by\_income\_type WHERE relief\_country = 'UK')

total\_sa\_relief = SUM(relief\_by\_income\_type WHERE relief\_country = 'SA')

RETURN {

dta\_residence\_determination: dta\_residence,

relief\_by\_income\_type: relief\_by\_income\_type,

total\_uk\_foreign\_tax\_credit: total\_uk\_relief,

total\_sa\_foreign\_tax\_credit: total\_sa\_relief,

net\_tax\_saving: total\_uk\_relief + total\_sa\_relief

}

FUNCTION determine\_dta\_residence(user: User) -> DtaResidenceResult:

# UK-SA DTA Article 4: Residence tie-breaker rules

IF NOT user.uk\_tax\_resident AND NOT user.sa\_tax\_resident:

RETURN {residence: 'NEITHER', reason: 'Not resident in either country'}

IF user.uk\_tax\_resident AND NOT user.sa\_tax\_resident:

RETURN {residence: 'UK', reason: 'UK resident only'}

IF user.sa\_tax\_resident AND NOT user.uk\_tax\_resident:

RETURN {residence: 'SA', reason: 'SA resident only'}

# Dual resident - apply tie-breaker rules

IF user.uk\_tax\_resident AND user.sa\_tax\_resident:

# Article 4(2) tie-breaker cascade:

# (a) Permanent home available

IF has\_permanent\_home\_in('UK') AND NOT has\_permanent\_home\_in('SA'):

RETURN {residence: 'UK', reason: 'Permanent home in UK only'}

IF has\_permanent\_home\_in('SA') AND NOT has\_permanent\_home\_in('UK'):

RETURN {residence: 'SA', reason: 'Permanent home in SA only'}

# (b) Centre of vital interests (stronger personal/economic ties)

IF has\_permanent\_home\_in('UK') AND has\_permanent\_home\_in('SA'):

IF centre\_of\_vital\_interests = 'UK':

RETURN {residence: 'UK', reason: 'Centre of vital interests in UK'}

ELSE IF centre\_of\_vital\_interests = 'SA':

RETURN {residence: 'SA', reason: 'Centre of vital interests in SA'}

# (c) Habitual abode (where person normally lives)

IF habitual\_abode\_country = 'UK':

RETURN {residence: 'UK', reason: 'Habitual abode in UK'}

ELSE IF habitual\_abode\_country = 'SA':

RETURN {residence: 'SA', reason: 'Habitual abode in SA'}

# (d) Nationality

IF user.nationality = 'UK' AND user.nationality != 'SA':

RETURN {residence: 'UK', reason: 'UK national'}

ELSE IF user.nationality = 'SA' AND user.nationality != 'UK':

RETURN {residence: 'SA', reason: 'SA national'}

# (e) Mutual agreement procedure

RETURN {

residence: 'UNDETERMINED',

reason: 'Requires mutual agreement between UK and SA tax authorities',

action\_required: 'Contact tax advisor'

}

FUNCTION categorize\_income\_by\_dta\_article(

income\_items: array[IncomeItem]

) -> array[CategorizedIncome]:

categorized = []

FOR EACH income IN income\_items:

category = {

income: income,

dta\_article: determine\_dta\_article(income.type),

taxing\_rights: determine\_taxing\_rights(income),

relief\_method: determine\_relief\_method(income)

}

categorized.append(category)

RETURN categorized

FUNCTION determine\_dta\_article(income\_type: string) -> integer:

# Map income types to DTA articles

MATCH income\_type:

CASE 'EMPLOYMENT':

RETURN 15 # Article 15: Employment income

CASE 'SELF\_EMPLOYMENT':

RETURN 7 # Article 7: Business profits

CASE 'DIVIDEND':

RETURN 10 # Article 10: Dividends

CASE 'INTEREST':

RETURN 11 # Article 11: Interest

CASE 'ROYALTY':

RETURN 12 # Article 12: Royalties

CASE 'PENSION':

RETURN 17 # Article 17: Pensions

CASE 'GOVERNMENT\_SERVICE':

RETURN 19 # Article 19: Government service

CASE 'CAPITAL\_GAIN':

RETURN 13 # Article 13: Capital gains

DEFAULT:

RETURN 21 # Article 21: Other income

FUNCTION determine\_taxing\_rights(income: IncomeItem) -> TaxingRights:

# Determine which country can tax under DTA

dta\_article = determine\_dta\_article(income.type)

MATCH dta\_article:

CASE 7: # Business profits

# Taxable only in residence country unless PE in source country

IF income.has\_permanent\_establishment\_in\_source\_country:

RETURN {

source\_country\_can\_tax: TRUE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'SOURCE'

}

ELSE:

RETURN {

source\_country\_can\_tax: FALSE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'RESIDENCE'

}

CASE 10: # Dividends

# Both can tax, but source limited to 15% (10% if >10% shareholding)

IF income.shareholding\_percentage >= 10:

source\_country\_rate\_limit = 0.10

ELSE:

source\_country\_rate\_limit = 0.15

RETURN {

source\_country\_can\_tax: TRUE,

source\_country\_rate\_limit: source\_country\_rate\_limit,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'SOURCE',

relief\_method: 'CREDIT'

}

CASE 11: # Interest

# Both can tax, but source limited to 10%

RETURN {

source\_country\_can\_tax: TRUE,

source\_country\_rate\_limit: 0.10,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'SOURCE',

relief\_method: 'CREDIT'

}

CASE 13: # Capital gains

# Immovable property: Taxed in country where situated

# Moveable property of PE: Taxed in PE country

# Shares: Taxed in residence country

# Other: Taxed in residence country

IF income.asset\_type = 'IMMOVABLE\_PROPERTY':

RETURN {

source\_country\_can\_tax: TRUE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'SOURCE'

}

ELSE IF income.asset\_type = 'SHARES\_DERIVING\_VALUE\_FROM\_PROPERTY':

# Shares deriving >50% value from immovable property

RETURN {

source\_country\_can\_tax: TRUE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'SOURCE'

}

ELSE:

RETURN {

source\_country\_can\_tax: FALSE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'RESIDENCE'

}

CASE 15: # Employment income

# Taxed in country where employment exercised

# Exceptions if: <183 days, paid by non-resident employer, no PE

IF income.days\_worked\_in\_source < 183 AND

income.employer\_not\_resident\_in\_source AND

income.not\_borne\_by\_pe\_in\_source:

# Exempt in source country

RETURN {

source\_country\_can\_tax: FALSE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'RESIDENCE'

}

ELSE:

RETURN {

source\_country\_can\_tax: TRUE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'SOURCE'

}

CASE 17: # Pensions

# Private pensions: Taxed only in residence country

# Government pensions: Taxed in source country (unless national of other state)

IF income.pension\_type = 'GOVERNMENT':

IF income.beneficiary\_national\_of\_residence\_country:

RETURN {

source\_country\_can\_tax: FALSE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'RESIDENCE'

}

ELSE:

RETURN {

source\_country\_can\_tax: TRUE,

residence\_country\_can\_tax: FALSE,

primary\_taxing\_rights: 'SOURCE'

}

ELSE: # Private pension

RETURN {

source\_country\_can\_tax: FALSE,

residence\_country\_can\_tax: TRUE,

primary\_taxing\_rights: 'RESIDENCE'

}

FUNCTION calculate\_relief\_for\_category(

income\_category: CategorizedIncome,

dta\_residence: string,

uk\_tax\_resident: boolean,

sa\_tax\_resident: boolean

) -> ReliefCalculation:

income = income\_category.income

taxing\_rights = income\_category.taxing\_rights

# Calculate tax in source country

IF income.source\_country = 'UK':

source\_tax = calculate\_uk\_tax\_on\_income(income)

ELSE IF income.source\_country = 'SA':

source\_tax = calculate\_sa\_tax\_on\_income(income)

# Calculate tax in residence country

IF dta\_residence = 'UK':

residence\_tax = calculate\_uk\_tax\_on\_income(income)

relief\_country = 'UK'

ELSE IF dta\_residence = 'SA':

residence\_tax = calculate\_sa\_tax\_on\_income(income)

relief\_country = 'SA'

# Determine relief

IF taxing\_rights.primary\_taxing\_rights = 'SOURCE':

# Source country taxes first

# Residence country gives credit

foreign\_tax\_paid = source\_tax

# Relief limited to lower of: foreign tax or domestic tax on that income

relief\_amount = MIN(foreign\_tax\_paid, residence\_tax)

RETURN {

income\_description: income.description,

income\_amount: income.amount,

source\_country: income.source\_country,

source\_country\_tax: source\_tax,

residence\_country: dta\_residence,

residence\_country\_tax\_before\_relief: residence\_tax,

foreign\_tax\_credit: relief\_amount,

residence\_country\_tax\_after\_relief: MAX(residence\_tax - relief\_amount, 0),

relief\_given\_by: relief\_country,

net\_tax\_on\_income: source\_tax + (residence\_tax - relief\_amount)

}

ELSE IF taxing\_rights.primary\_taxing\_rights = 'RESIDENCE':

# Only residence country taxes

# No relief needed

RETURN {

income\_description: income.description,

income\_amount: income.amount,

taxed\_in: dta\_residence,

tax\_amount: residence\_tax,

foreign\_tax\_credit: 0,

no\_relief\_needed: TRUE

}

FUNCTION calculate\_uk\_tax\_on\_income(income: IncomeItem) -> decimal:

# Calculate UK tax on specific income item

# Use appropriate UK tax calculation based on income type

MATCH income.type:

CASE 'EMPLOYMENT', 'SELF\_EMPLOYMENT', 'PENSION':

result = calculate\_uk\_income\_tax(income.amount, income.tax\_year, FALSE)

RETURN result.total\_tax

CASE 'DIVIDEND':

result = calculate\_uk\_dividend\_tax(income.amount, income.tax\_year, 0, 37700)

RETURN result.total\_tax

CASE 'INTEREST':

# Interest taxed as income at marginal rate

result = calculate\_uk\_income\_tax(income.amount, income.tax\_year, FALSE)

RETURN result.total\_tax

CASE 'CAPITAL\_GAIN':

result = calculate\_uk\_capital\_gains\_tax(income.amount, 'OTHER', income.tax\_year, 0, 37700)

RETURN result.total\_cgt

FUNCTION calculate\_sa\_tax\_on\_income(income: IncomeItem) -> decimal:

# Calculate SA tax on specific income item

MATCH income.type:

CASE 'EMPLOYMENT', 'SELF\_EMPLOYMENT', 'PENSION', 'INTEREST':

result = calculate\_sa\_income\_tax(income.amount, income.tax\_year, 'UNDER\_65')

RETURN result.tax\_payable

CASE 'DIVIDEND':

result = calculate\_sa\_dividend\_tax(income.amount, 'FOREIGN', income.tax\_year)

# Foreign dividends taxed at marginal rate

# Approximate using income tax calculation

result = calculate\_sa\_income\_tax(income.amount, income.tax\_year, 'UNDER\_65')

RETURN result.tax\_payable

CASE 'CAPITAL\_GAIN':

result = calculate\_sa\_capital\_gains\_tax(income.amount, income.tax\_year, 0)

RETURN result.cgt

**API Endpoints:**

POST /api/v1/tax/dta/calculate-relief

POST /api/v1/tax/dta/determine-residence

POST /api/v1/tax/dta/categorize-income

GET /api/v1/tax/dta/treaty-provisions/{article}

POST /api/v1/tax/dta/foreign-tax-credit

**Data Models:**

TABLE: dta\_provisions

- id: UUID (PK)

- treaty: VARCHAR(50) DEFAULT 'UK\_SA'

- article\_number: INTEGER

- article\_title: VARCHAR(255)

- provision\_text: TEXT

- income\_type: VARCHAR(100)

- source\_taxing\_rights: BOOLEAN

- residence\_taxing\_rights: BOOLEAN

- source\_rate\_limit: DECIMAL(5,4)

- relief\_method: ENUM('CREDIT', 'EXEMPTION', 'DEDUCTION')

- effective\_from: DATE

- effective\_to: DATE

TABLE: dta\_relief\_calculations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- income\_id: UUID (FK to user\_income)

- tax\_year: VARCHAR(10)

- dta\_article: INTEGER

- source\_country: ENUM('UK', 'SA')

- residence\_country: ENUM('UK', 'SA')

- income\_amount: DECIMAL(15,2)

- source\_country\_tax: DECIMAL(15,2)

- residence\_country\_tax\_before\_relief: DECIMAL(15,2)

- foreign\_tax\_credit: DECIMAL(15,2)

- residence\_country\_tax\_after\_relief: DECIMAL(15,2)

- relief\_given\_by: ENUM('UK', 'SA')

- calculation\_date: TIMESTAMP

TABLE: dta\_residence\_determinations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- determination\_date: DATE

- tax\_year: VARCHAR(10)

- uk\_tax\_resident: BOOLEAN

- sa\_tax\_resident: BOOLEAN

- permanent\_home\_uk: BOOLEAN

- permanent\_home\_sa: BOOLEAN

- centre\_of\_vital\_interests: ENUM('UK', 'SA', 'UNCLEAR')

- habitual\_abode: ENUM('UK', 'SA', 'BOTH')

- nationality: VARCHAR(50)

- dta\_residence\_conclusion: ENUM('UK', 'SA', 'UNDETERMINED')

- tie\_breaker\_rule\_applied: VARCHAR(100)

- reasoning: TEXT

- created\_at: TIMESTAMP

INDEX on dta\_relief\_calculations(user\_id, tax\_year)

INDEX on dta\_provisions(article\_number, income\_type)

**Error Handling:**

ERROR CASES:

1. Dual residence cannot be determined

- Response: 200 OK with warning

- Message: "DTA residence cannot be automatically determined. Manual review required"

- Recommendation: "Contact tax advisor for mutual agreement procedure"

2. Income type not covered by DTA

- Response: 400 Bad Request

- Message: "Income type {type} not explicitly covered by UK-SA DTA"

- Fallback: "Apply general 'Other Income' provisions (Article 21)"

3. Conflicting tax residence claims

- Response: 409 Conflict

- Message: "Tax residence conflict detected. Both countries may claim taxing rights"

- Action: "Apply DTA tie-breaker rules"

4. Missing required information for DTA determination

- Response: 400 Bad Request

- Message: "Insufficient information to determine DTA treatment"

- Required: "Please provide: {list of missing data}"

EDGE CASES:

- Triple residence (UK, SA, and third country): Not handled, manual review

- Income from third country: No DTA relief for UK-SA treaty

- Treaty shopping: Beneficial ownership requirements

- Permanent establishment determination: Complex, may need manual review

- Employment income split across countries: Apportion by days worked

- Pension from third country: DTA may not apply

- Government service pensions: Special rules apply

- Students and trainees: Special exemptions (Article 20)

- Artists and sportspersons: Special provisions (Article 16)

- Offshore structures: Anti-avoidance provisions

- Limitation of benefits clause: May restrict DTA access

**Feature 9.3: Tax Residency Status Determination**

**Feature Name:** Automated Tax Residency and Domicile Status Assessment

**User Story:** As a user with ties to multiple countries, I need the system to accurately determine my tax residency and domicile status so that I understand which tax rules apply to me.

**Acceptance Criteria:**

* UK Statutory Residence Test (SRT) automation
* SA Physical Presence Test automation
* UK domicile status determination
* Deemed domicile calculation (UK)
* Split year treatment identification (UK)
* Temporary non-residence rules (UK)
* Residency history tracking
* Tax certificate generation support

**Technical Requirements:**

* Rules engine for SRT (complex logic)
* Day counting algorithms
* Tie-breaker logic
* Historical residency tracking
* Date-based calculations
* Supporting documentation references

**Constraints:**

* UK SRT: Based on days in UK, ties, and automatic tests
* SA test: 91 days in current year + average 91 days over 5 years
* Domicile: Long-term concept, difficult to change
* Deemed domicile: 15 of last 20 years UK resident OR UK domicile of origin + 1 of last 2 years resident
* Split year: Only applies in specific circumstances

**Implementation Approach:**

SERVICE: TaxResidencyDetermination

# ===== UK STATUTORY RESIDENCE TEST =====

FUNCTION calculate\_uk\_srt(

user\_id: uuid,

tax\_year: string,

days\_in\_uk: integer,

ties: UkTiesData

) -> UkSrtResult:

# UK SRT has three parts:

# 1. Automatic Overseas Test (definitely NOT resident)

# 2. Automatic UK Test (definitely resident)

# 3. Sufficient Ties Test (depends on days and ties)

# Part 1: Automatic Overseas Test

automatic\_overseas = check\_automatic\_overseas\_test(user\_id, tax\_year, days\_in\_uk)

IF automatic\_overseas.result = TRUE:

RETURN {

uk\_resident: FALSE,

determination\_method: 'AUTOMATIC\_OVERSEAS\_TEST',

test\_met: automatic\_overseas.test\_met,

days\_in\_uk: days\_in\_uk,

reasoning: automatic\_overseas.reasoning

}

# Part 2: Automatic UK Test

automatic\_uk = check\_automatic\_uk\_test(user\_id, tax\_year, days\_in\_uk)

IF automatic\_uk.result = TRUE:

RETURN {

uk\_resident: TRUE,

determination\_method: 'AUTOMATIC\_UK\_TEST',

test\_met: automatic\_uk.test\_met,

days\_in\_uk: days\_in\_uk,

reasoning: automatic\_uk.reasoning

}

# Part 3: Sufficient Ties Test

sufficient\_ties = check\_sufficient\_ties\_test(user\_id, tax\_year, days\_in\_uk, ties)

RETURN {

uk\_resident: sufficient\_ties.result,

determination\_method: 'SUFFICIENT\_TIES\_TEST',

days\_in\_uk: days\_in\_uk,

ties\_count: sufficient\_ties.ties\_count,

ties\_needed: sufficient\_ties.ties\_needed,

ties\_detail: sufficient\_ties.ties\_detail,

reasoning: sufficient\_ties.reasoning

}

FUNCTION check\_automatic\_overseas\_test(

user\_id: uuid,

tax\_year: string,

days\_in\_uk: integer

) -> AutomaticOverseasTestResult:

# Test 1: Present in UK for fewer than 16 days

IF days\_in\_uk < 16:

RETURN {

result: TRUE,

test\_met: 'FEWER\_THAN\_16\_DAYS',

reasoning: 'Present in UK for fewer than 16 days'

}

# Test 2: Not UK resident in all 3 preceding tax years AND present < 46 days

was\_non\_resident\_previous\_3\_years = check\_previous\_residence(user\_id, 3)

IF was\_non\_resident\_previous\_3\_years AND days\_in\_uk < 46:

RETURN {

result: TRUE,

test\_met: 'NON\_RESIDENT\_3\_YEARS\_PLUS\_UNDER\_46\_DAYS',

reasoning: 'Non-resident in previous 3 years and present fewer than 46 days'

}

# Test 3: Full-time work abroad AND present < 91 days AND no more than 30 working days in UK

full\_time\_work\_abroad = check\_full\_time\_work\_abroad(user\_id, tax\_year)

working\_days\_in\_uk = get\_working\_days\_in\_uk(user\_id, tax\_year)

IF full\_time\_work\_abroad AND days\_in\_uk < 91 AND working\_days\_in\_uk <= 30:

RETURN {

result: TRUE,

test\_met: 'FULL\_TIME\_WORK\_ABROAD',

reasoning: 'Full-time work abroad with fewer than 91 days in UK and no more than 30 working days'

}

RETURN {

result: FALSE,

reasoning: 'No automatic overseas test met'

}

FUNCTION check\_automatic\_uk\_test(

user\_id: uuid,

tax\_year: string,

days\_in\_uk: integer

) -> AutomaticUkTestResult:

# Test 1: Present in UK for 183 days or more

IF days\_in\_uk >= 183:

RETURN {

result: TRUE,

test\_met: '183\_DAYS\_OR\_MORE',

reasoning: 'Present in UK for 183 days or more'

}

# Test 2: Only home in UK (or no home anywhere)

home\_status = check\_home\_status(user\_id, tax\_year)

IF home\_status.only\_home\_in\_uk:

IF home\_status.present\_in\_home\_for\_at\_least\_30\_days:

RETURN {

result: TRUE,

test\_met: 'ONLY\_HOME\_IN\_UK',

reasoning: 'Only home in UK and present for at least 30 days in tax year'

}

# Test 3: Full-time work in UK

full\_time\_work\_uk = check\_full\_time\_work\_uk(user\_id, tax\_year)

IF full\_time\_work\_uk.qualifies:

RETURN {

result: TRUE,

test\_met: 'FULL\_TIME\_WORK\_UK',

reasoning: 'Full-time work in UK for at least 365 days with no significant breaks'

}

RETURN {

result: FALSE,

reasoning: 'No automatic UK test met'

}

FUNCTION check\_sufficient\_ties\_test(

user\_id: uuid,

tax\_year: string,

days\_in\_uk: integer,

ties: UkTiesData

) -> SufficientTiesTestResult:

# Five UK ties:

# 1. Family tie: Spouse/civil partner or minor children resident in UK

# 2. Accommodation tie: Available accommodation in UK (used during year)

# 3. Work tie: 40+ days doing >3 hours work in UK

# 4. 90-day tie: Spent 90+ days in UK in either of previous 2 tax years

# 5. Country tie: Present in UK more than any other single country

ties\_count = 0

ties\_detail = []

# Tie 1: Family

IF ties.has\_uk\_resident\_spouse OR ties.has\_uk\_resident\_minor\_children:

ties\_count += 1

ties\_detail.append({

tie: 'FAMILY',

met: TRUE,

reason: 'Spouse or minor children resident in UK'

})

ELSE:

ties\_detail.append({tie: 'FAMILY', met: FALSE})

# Tie 2: Accommodation

IF ties.has\_uk\_accommodation\_available AND ties.spent\_at\_least\_one\_night:

ties\_count += 1

ties\_detail.append({

tie: 'ACCOMMODATION',

met: TRUE,

reason: 'UK accommodation available and used'

})

ELSE:

ties\_detail.append({tie: 'ACCOMMODATION', met: FALSE})

# Tie 3: Work

IF ties.uk\_working\_days >= 40:

ties\_count += 1

ties\_detail.append({

tie: 'WORK',

met: TRUE,

reason: '40+ days working more than 3 hours in UK'

})

ELSE:

ties\_detail.append({tie: 'WORK', met: FALSE})

# Tie 4: 90-day

days\_in\_previous\_years = get\_days\_in\_previous\_years(user\_id, 2)

IF days\_in\_previous\_years[0] >= 90 OR days\_in\_previous\_years[1] >= 90:

ties\_count += 1

ties\_detail.append({

tie: '90\_DAY',

met: TRUE,

reason: '90+ days in UK in one of previous 2 tax years'

})

ELSE:

ties\_detail.append({tie: '90\_DAY', met: FALSE})

# Tie 5: Country (only for "leavers" - UK resident in 1+ of previous 3 years)

was\_uk\_resident\_in\_previous\_3\_years = check\_previous\_residence(user\_id, 3)

IF was\_uk\_resident\_in\_previous\_3\_years:

IF ties.days\_in\_uk\_greater\_than\_any\_other\_country:

ties\_count += 1

ties\_detail.append({

tie: 'COUNTRY',

met: TRUE,

reason: 'More days in UK than any other single country'

})

ELSE:

ties\_detail.append({tie: 'COUNTRY', met: FALSE})

ELSE:

ties\_detail.append({

tie: 'COUNTRY',

met: 'N/A',

reason: 'Not a "leaver" - not UK resident in previous 3 years'

})

# Determine if sufficient ties

# Different thresholds for "arrivers" vs "leavers"

IF was\_uk\_resident\_in\_previous\_3\_years:

# "Leaver" - lower thresholds

IF days\_in\_uk < 16:

ties\_needed = 4 # 4 ties needed (impossible - max 4 ties apply)

ELSE IF days\_in\_uk >= 16 AND days\_in\_uk <= 45:

ties\_needed = 4

ELSE IF days\_in\_uk >= 46 AND days\_in\_uk <= 90:

ties\_needed = 3

ELSE IF days\_in\_uk >= 91 AND days\_in\_uk <= 120:

ties\_needed = 2

ELSE IF days\_in\_uk >= 121:

ties\_needed = 1

ELSE:

# "Arriver" - higher thresholds (country tie doesn't apply)

IF days\_in\_uk < 46:

ties\_needed = 4 # 4 ties needed (impossible - max 4 ties for arrivers)

ELSE IF days\_in\_uk >= 46 AND days\_in\_uk <= 90:

ties\_needed = 4

ELSE IF days\_in\_uk >= 91 AND days\_in\_uk <= 120:

ties\_needed = 3

ELSE IF days\_in\_uk >= 121:

ties\_needed = 2

is\_uk\_resident = (ties\_count >= ties\_needed)

RETURN {

result: is\_uk\_resident,

ties\_count: ties\_count,

ties\_needed: ties\_needed,

ties\_detail: ties\_detail,

status: IF was\_uk\_resident\_in\_previous\_3\_years THEN 'LEAVER' ELSE 'ARRIVER',

reasoning: IF is\_uk\_resident THEN

"UK resident: {ties\_count} ties, {ties\_needed} needed"

ELSE

"Not UK resident: {ties\_count} ties, {ties\_needed} needed"

}

# ===== SA PHYSICAL PRESENCE TEST =====

FUNCTION calculate\_sa\_residence(

user\_id: uuid,

tax\_year: string,

days\_in\_sa: integer

) -> SaResidenceResult:

# SA Physical Presence Test:

# Resident if:

# 1. Physically present for > 91 days in current year, AND

# 2. Physically present for > 91 days on average over current + previous 5 years

# OR

# 3. Ordinarily resident (habitual residence)

# Test 1: Current year

IF days\_in\_sa <= 91:

RETURN {

sa\_resident: FALSE,

determination\_method: 'PHYSICAL\_PRESENCE',

days\_current\_year: days\_in\_sa,

reasoning: 'Not present for more than 91 days in current year'

}

# Test 2: Average over 5 years

days\_previous\_5\_years = get\_sa\_days\_previous\_years(user\_id, 5)

total\_days\_6\_years = days\_in\_sa + SUM(days\_previous\_5\_years)

average\_days = total\_days\_6\_years / 6

IF average\_days <= 91:

RETURN {

sa\_resident: FALSE,

determination\_method: 'PHYSICAL\_PRESENCE',

days\_current\_year: days\_in\_sa,

average\_days\_6\_years: average\_days,

reasoning: 'Average days over 6 years (current + 5 previous) not more than 91'

}

# Both tests passed

RETURN {

sa\_resident: TRUE,

determination\_method: 'PHYSICAL\_PRESENCE',

days\_current\_year: days\_in\_sa,

days\_by\_year: days\_previous\_5\_years,

average\_days\_6\_years: average\_days,

reasoning: 'Present more than 91 days current year AND average > 91 days over 6 years'

}

# ===== UK DOMICILE DETERMINATION =====

FUNCTION determine\_uk\_domicile(

user\_id: uuid,

current\_date: date

) -> UkDomicileResult:

user = get\_user(user\_id)

# Domicile of Origin

domicile\_of\_origin = user.domicile\_of\_origin

# Current domicile (may have been changed by choice)

current\_domicile = user.current\_domicile

# UK Deemed Domicile (for IHT purposes)

# Deemed domiciled if:

# 1. UK resident for 15 of last 20 tax years, OR

# 2. UK domicile of origin AND UK resident in 1 of previous 2 tax years

residence\_history = get\_residence\_history(user\_id, 20)

uk\_resident\_years = COUNT(residence\_history WHERE uk\_resident = TRUE)

deemed\_domicile\_rule\_1 = (uk\_resident\_years >= 15)

previous\_2\_years = get\_residence\_history(user\_id, 2)

uk\_resident\_previous\_2 = ANY(previous\_2\_years WHERE uk\_resident = TRUE)

deemed\_domicile\_rule\_2 = (domicile\_of\_origin = 'UK' AND uk\_resident\_previous\_2)

is\_deemed\_domiciled = (deemed\_domicile\_rule\_1 OR deemed\_domicile\_rule\_2)

RETURN {

domicile\_of\_origin: domicile\_of\_origin,

current\_domicile: current\_domicile,

uk\_deemed\_domicile: is\_deemed\_domiciled,

deemed\_domicile\_reason: IF deemed\_domicile\_rule\_1 THEN

'15 of last 20 years UK resident'

ELSE IF deemed\_domicile\_rule\_2 THEN

'UK domicile of origin + UK resident in previous 2 years'

ELSE

'N/A',

uk\_resident\_years\_in\_last\_20: uk\_resident\_years,

for\_iht\_purposes: IF is\_deemed\_domiciled THEN

'Treated as UK domiciled - worldwide assets in scope'

ELSE IF current\_domicile = 'UK' THEN

'UK domiciled - worldwide assets in scope'

ELSE

'Non-UK domiciled - only UK assets in scope (excluded property applies)'

}

# ===== SPLIT YEAR TREATMENT =====

FUNCTION check\_split\_year\_treatment(

user\_id: uuid,

tax\_year: string

) -> SplitYearResult:

# Split year treatment applies in 8 specific cases:

# Cases 1-3: Arriving in UK

# Cases 4-8: Leaving UK

# This is complex - simplified version

residence\_status = calculate\_uk\_srt(user\_id, tax\_year, days\_in\_uk, ties)

IF NOT residence\_status.uk\_resident:

RETURN {

split\_year\_applies: FALSE,

reasoning: 'Not UK resident - split year not relevant'

}

# Check for qualifying circumstances

circumstances = analyze\_split\_year\_circumstances(user\_id, tax\_year)

IF circumstances.started\_full\_time\_work\_abroad:

# Case 1: Starting full-time work abroad

split\_date = circumstances.departure\_date

RETURN {

split\_year\_applies: TRUE,

case: 'CASE\_1\_STARTING\_FULL\_TIME\_WORK\_ABROAD',

split\_date: split\_date,

uk\_part: {from: tax\_year\_start, to: split\_date},

overseas\_part: {from: split\_date + 1, to: tax\_year\_end},

reasoning: 'Started full-time work abroad'

}

IF circumstances.ceased\_uk\_residence\_on\_leaving:

# Case 8: Ceasing to have a home in UK

# ... other cases

RETURN {

split\_year\_applies: FALSE,

reasoning: 'No qualifying circumstances for split year treatment'

}

**API Endpoints:**

POST /api/v1/tax/residency/uk-srt

POST /api/v1/tax/residency/sa-physical-presence

POST /api/v1/tax/residency/uk-domicile

POST /api/v1/tax/residency/split-year-check

GET /api/v1/tax/residency/history/{userId}

POST /api/v1/tax/residency/day-count-calculator

**Data Models:**

TABLE: residency\_determinations

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: VARCHAR(10)

- country: ENUM('UK', 'SA')

- determination\_date: DATE

- days\_in\_country: INTEGER

- resident: BOOLEAN

- determination\_method: VARCHAR(100)

- test\_details: JSON

- reasoning: TEXT

- created\_at: TIMESTAMP

TABLE: uk\_srt\_tie\_data

- id: UUID (PK)

- user\_id: UUID (FK to users)

- tax\_year: VARCHAR(7)

- family\_tie: BOOLEAN

- accommodation\_tie: BOOLEAN

- work\_tie: BOOLEAN

- ninety\_day\_tie: BOOLEAN

- country\_tie: BOOLEAN

- ties\_count: INTEGER

- uk\_working\_days: INTEGER

- accommodation\_details: TEXT

- updated\_at: TIMESTAMP

TABLE: day\_count\_records

- id: UUID (PK)

- user\_id: UUID (FK to users)

- country: ENUM('UK', 'SA', 'OTHER')

- entry\_date: DATE

- exit\_date: DATE

- days\_count: INTEGER

- purpose: VARCHAR(255)

- notes: TEXT

- created\_at: TIMESTAMP

TABLE: domicile\_history

- id: UUID (PK)

- user\_id: UUID (FK to users)

- effective\_from: DATE

- effective\_to: DATE

- domicile\_of\_origin: VARCHAR(50)

- domicile\_of\_choice: VARCHAR(50)

- deemed\_domicile\_uk: BOOLEAN

- reasoning: TEXT

- created\_at: TIMESTAMP

INDEX on residency\_determinations(user\_id, tax\_year, country)

INDEX on day\_count\_records(user\_id, country, entry\_date, exit\_date)

**Performance Considerations:**

* SRT calculation: Complex logic, cache result per tax year
* Day counting: Optimize queries with date range indexes
* Historical lookups: Pre-aggregate yearly totals
* Tie determination: May require multiple sub-queries
* Expected calculations: Once per tax year per user
* Target: <500ms for complete SRT determination

**10. AI ADVISORY ENGINE**

**Feature 10.1: Recommendation Generation System**

**Feature Name:** Intelligent Financial Recommendations Based on User Context

**User Story:** As a user, I want to receive personalized, actionable financial recommendations based on my complete financial situation, goals, tax status, and life circumstances so that I can make informed decisions to improve my financial position.

**Acceptance Criteria:**

* Generate recommendations across all modules (protection, savings, investment, retirement, IHT)
* Prioritize recommendations by impact and urgency
* Provide clear reasoning for each recommendation
* Calculate estimated financial benefit (tax savings, returns, protection gap closure)
* Consider user's risk tolerance and life stage
* Avoid conflicting recommendations
* Update recommendations when user data changes
* Track recommendation acceptance and outcomes

**Technical Requirements:**

* Rules engine for recommendation logic
* Machine learning model for personalization (optional future)
* Multi-criteria decision analysis for prioritization
* Natural language generation for explanations
* Integration with all financial modules
* A/B testing framework for recommendation effectiveness
* Recommendation versioning and audit trail

**Constraints:**

* Must not provide regulated financial advice (information only)
* Recommendations must be explainable (no black box)
* Must consider user's complete situation (holistic)
* Update frequency: Real-time on data change, daily batch for periodic checks
* Maximum 20 active recommendations per user
* Performance: Generate recommendations in <3 seconds

**Implementation Approach:**

SERVICE: RecommendationEngine

FUNCTION generate\_recommendations(user\_id: uuid) -> RecommendationSet:

# 1. Gather complete user context

context = build\_user\_context(user\_id)

# 2. Run all recommendation rules

raw\_recommendations = []

# Protection recommendations

raw\_recommendations.extend(generate\_protection\_recommendations(context))

# Savings recommendations

raw\_recommendations.extend(generate\_savings\_recommendations(context))

# Investment recommendations

raw\_recommendations.extend(generate\_investment\_recommendations(context))

# Retirement recommendations

raw\_recommendations.extend(generate\_retirement\_recommendations(context))

# Tax optimization recommendations

raw\_recommendations.extend(generate\_tax\_recommendations(context))

# IHT planning recommendations

raw\_recommendations.extend(generate\_iht\_recommendations(context))

# Cross-cutting recommendations

raw\_recommendations.extend(generate\_cross\_cutting\_recommendations(context))

# 3. Score and prioritize

scored\_recommendations = score\_recommendations(raw\_recommendations, context)

# 4. Filter and deduplicate

filtered\_recommendations = filter\_recommendations(scored\_recommendations, context)

# 5. Rank by priority

ranked\_recommendations = rank\_recommendations(filtered\_recommendations)

# 6. Limit to top N

top\_recommendations = ranked\_recommendations[0:20]

# 7. Generate explanations

final\_recommendations = generate\_explanations(top\_recommendations, context)

# 8. Store recommendations

store\_recommendations(user\_id, final\_recommendations)

RETURN {

recommendations: final\_recommendations,

generated\_at: NOW(),

context\_snapshot: context.summary,

total\_potential\_benefit: SUM(final\_recommendations.estimated\_benefit)

}

FUNCTION build\_user\_context(user\_id: uuid) -> UserContext:

user = get\_user(user\_id)

RETURN {

# Demographics

age: calculate\_age(user.date\_of\_birth),

life\_stage: determine\_life\_stage(user),

dependents: get\_dependents(user\_id),

# Financial position

income: get\_total\_income(user\_id),

net\_worth: get\_net\_worth(user\_id),

liquid\_assets: get\_liquid\_assets(user\_id),

liabilities: get\_total\_liabilities(user\_id),

# Tax status

uk\_tax\_resident: user.uk\_tax\_resident,

sa\_tax\_resident: user.sa\_tax\_resident,

domicile: user.domicile,

marginal\_tax\_rate\_uk: calculate\_marginal\_rate\_uk(user\_id),

marginal\_tax\_rate\_sa: calculate\_marginal\_rate\_sa(user\_id),

# Module-specific

protection: {

life\_cover: get\_total\_life\_cover(user\_id),

critical\_illness\_cover: get\_ci\_cover(user\_id),

income\_protection: get\_income\_protection(user\_id)

},

savings: {

emergency\_fund: get\_emergency\_fund(user\_id),

isa\_allowance\_used: get\_isa\_allowance\_used(user\_id),

tfsa\_allowance\_used: get\_tfsa\_allowance\_used(user\_id),

total\_cash: get\_total\_cash(user\_id)

},

investments: {

portfolio\_value: get\_portfolio\_value(user\_id),

asset\_allocation: get\_asset\_allocation(user\_id),

unrealized\_gains: get\_unrealized\_gains(user\_id),

cgt\_allowance\_used: get\_cgt\_allowance\_used(user\_id)

},

retirement: {

total\_pension\_pot: get\_total\_pension\_pot(user\_id),

annual\_allowance\_used: get\_annual\_allowance\_used(user\_id),

years\_to\_retirement: user.expected\_retirement\_age - calculate\_age(user.date\_of\_birth),

on\_track: assess\_retirement\_readiness(user\_id)

},

iht: {

estate\_value: get\_estate\_value(user\_id),

iht\_liability: get\_iht\_liability(user\_id),

gifts\_in\_7\_years: get\_gifts\_in\_7\_years(user\_id),

nrb\_used: get\_nrb\_utilization(user\_id)

},

# Goals and preferences

goals: get\_user\_goals(user\_id),

risk\_tolerance: user.risk\_tolerance,

investment\_horizon: user.investment\_horizon,

# Behavioral

recommendation\_acceptance\_rate: calculate\_acceptance\_rate(user\_id),

preferred\_recommendation\_types: get\_preferred\_types(user\_id)

}

# ===== PROTECTION RECOMMENDATIONS =====

FUNCTION generate\_protection\_recommendations(context: UserContext) -> array[Recommendation]:

recommendations = []

# Rule 1: Insufficient life cover

recommended\_life\_cover = calculate\_recommended\_life\_cover(context)

current\_life\_cover = context.protection.life\_cover

IF current\_life\_cover < recommended\_life\_cover:

gap = recommended\_life\_cover - current\_life\_cover

recommendations.append({

category: 'PROTECTION',

type: 'INCREASE\_LIFE\_COVER',

priority: 'HIGH',

title: "Increase life cover to protect your family",

description: "Your current life cover of £{current\_life\_cover} may not be sufficient. We recommend £{recommended\_life\_cover} based on your income, debts, and family needs.",

estimated\_benefit: {

type: 'RISK\_MITIGATION',

description: "Protect your family's financial future",

coverage\_gap: gap

},

action\_required: "Review life insurance options",

reasoning: [

"Current cover: £{current\_life\_cover}",

"Recommended: £{recommended\_life\_cover}",

"Based on: {calculation\_factors}"

],

urgency\_score: 90,

impact\_score: 95

})

# Rule 2: No income protection with dependents

IF context.protection.income\_protection = 0 AND context.dependents.count > 0:

recommendations.append({

category: 'PROTECTION',

type: 'ADD\_INCOME\_PROTECTION',

priority: 'HIGH',

title: "Consider income protection insurance",

description: "You have {dependents} dependents but no income protection. This could help maintain your family's lifestyle if you're unable to work due to illness or injury.",

estimated\_benefit: {

type: 'RISK\_MITIGATION',

description: "Protect {percentage}% of your income",

monthly\_benefit: context.income.monthly \* 0.65

},

action\_required: "Explore income protection policies",

reasoning: [

"You have {dependents} dependents",

"No income protection in place",

"Typical replacement: 65% of income"

],

urgency\_score: 80,

impact\_score: 85

})

# Rule 3: Life insurance not in trust (UK)

IF context.uk\_tax\_resident:

policies\_not\_in\_trust = get\_policies\_not\_in\_trust(context.user\_id)

IF policies\_not\_in\_trust.count > 0:

total\_value = SUM(policies\_not\_in\_trust.cover\_amount)

potential\_iht = total\_value \* 0.40

recommendations.append({

category: 'PROTECTION',

type: 'WRITE\_POLICY\_IN\_TRUST',

priority: 'MEDIUM',

title: "Write life insurance policies in trust",

description: "You have {count} life insurance policies not written in trust. Writing them in trust would remove them from your estate for IHT purposes.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Potential IHT saving",

amount\_gbp: potential\_iht,

currency: 'GBP'

},

action\_required: "Contact insurers to set up trust arrangements",

reasoning: [

"{count} policies not in trust",

"Total cover: £{total\_value}",

"Potential IHT at 40%: £{potential\_iht}"

],

urgency\_score: 60,

impact\_score: 70

})

RETURN recommendations

# ===== SAVINGS RECOMMENDATIONS =====

FUNCTION generate\_savings\_recommendations(context: UserContext) -> array[Recommendation]:

recommendations = []

# Rule 1: Inadequate emergency fund

recommended\_emergency\_fund = context.income.monthly \* 6 # 6 months expenses

current\_emergency\_fund = context.savings.emergency\_fund

IF current\_emergency\_fund < recommended\_emergency\_fund:

shortfall = recommended\_emergency\_fund - current\_emergency\_fund

recommendations.append({

category: 'SAVINGS',

type: 'BUILD\_EMERGENCY\_FUND',

priority: 'HIGH',

title: "Build up your emergency fund",

description: "Your emergency fund is currently £{current} but we recommend £{recommended} (6 months of expenses).",

estimated\_benefit: {

type: 'FINANCIAL\_RESILIENCE',

description: "Protection against unexpected expenses",

target\_amount: recommended\_emergency\_fund

},

action\_required: "Set up regular savings of £{monthly\_target} per month",

monthly\_target: shortfall / 12, # Build over 12 months

reasoning: [

"Current emergency fund: £{current}",

"Recommended: £{recommended} (6 months)",

"Shortfall: £{shortfall}"

],

urgency\_score: 85,

impact\_score: 90

})

# Rule 2: Unused ISA allowance (UK)

IF context.uk\_tax\_resident:

isa\_allowance = 20000 # 2024/25

isa\_used = context.savings.isa\_allowance\_used

isa\_remaining = isa\_allowance - isa\_used

IF isa\_remaining > 5000: # Threshold for recommendation

days\_until\_year\_end = calculate\_days\_until\_uk\_tax\_year\_end()

# Calculate potential tax saving

assumed\_return = 0.05 # 5% return assumption

annual\_return = isa\_remaining \* assumed\_return

tax\_on\_return = annual\_return \* context.marginal\_tax\_rate\_uk

recommendations.append({

category: 'SAVINGS',

type: 'USE\_ISA\_ALLOWANCE',

priority: IF days\_until\_year\_end < 60 THEN 'HIGH' ELSE 'MEDIUM',

title: "Use your remaining ISA allowance",

description: "You have £{isa\_remaining} of unused ISA allowance for this tax year. ISA allowances don't carry forward.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Tax-free returns on investment",

annual\_tax\_saving: tax\_on\_return,

currency: 'GBP'

},

action\_required: "Transfer £{isa\_remaining} to ISA before April 5",

deadline: calculate\_uk\_tax\_year\_end(),

reasoning: [

"ISA allowance: £{isa\_allowance}",

"Used: £{isa\_used}",

"Remaining: £{isa\_remaining}",

"Tax year ends in {days} days"

],

urgency\_score: IF days\_until\_year\_end < 60 THEN 90 ELSE 60,

impact\_score: 70

})

# Rule 3: Unused TFSA allowance (SA)

IF context.sa\_tax\_resident:

tfsa\_annual\_limit = 36000 # 2024/25

tfsa\_lifetime\_limit = 500000

tfsa\_used\_this\_year = context.savings.tfsa\_allowance\_used.annual

tfsa\_lifetime\_used = context.savings.tfsa\_allowance\_used.lifetime

tfsa\_annual\_remaining = tfsa\_annual\_limit - tfsa\_used\_this\_year

tfsa\_lifetime\_remaining = tfsa\_lifetime\_limit - tfsa\_lifetime\_used

IF tfsa\_annual\_remaining > 10000 AND tfsa\_lifetime\_remaining > tfsa\_annual\_remaining:

days\_until\_year\_end = calculate\_days\_until\_sa\_tax\_year\_end()

recommendations.append({

category: 'SAVINGS',

type: 'USE\_TFSA\_ALLOWANCE',

priority: IF days\_until\_year\_end < 60 THEN 'HIGH' ELSE 'MEDIUM',

title: "Maximize your Tax-Free Savings Account",

description: "You have R{tfsa\_annual\_remaining} of unused TFSA allowance this year. TFSA returns are completely tax-free.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Tax-free investment returns",

currency: 'ZAR'

},

action\_required: "Contribute to TFSA before February 28",

deadline: calculate\_sa\_tax\_year\_end(),

reasoning: [

"Annual limit: R{tfsa\_annual\_limit}",

"Used this year: R{tfsa\_used\_this\_year}",

"Remaining: R{tfsa\_annual\_remaining}",

"Lifetime remaining: R{tfsa\_lifetime\_remaining}"

],

urgency\_score: IF days\_until\_year\_end < 60 THEN 85 ELSE 55,

impact\_score: 65

})

RETURN recommendations

# ===== INVESTMENT RECOMMENDATIONS =====

FUNCTION generate\_investment\_recommendations(context: UserContext) -> array[Recommendation]:

recommendations = []

# Rule 1: Poor diversification

allocation = context.investments.asset\_allocation

# Check for concentration risk

IF allocation.equity > 90:

recommendations.append({

category: 'INVESTMENT',

type: 'IMPROVE\_DIVERSIFICATION',

priority: 'MEDIUM',

title: "Reduce concentration risk in your portfolio",

description: "Your portfolio is {equity\_pct}% equities. Consider diversifying across asset classes to reduce risk.",

estimated\_benefit: {

type: 'RISK\_REDUCTION',

description: "Lower portfolio volatility"

},

action\_required: "Rebalance portfolio to target allocation",

reasoning: [

"Current equity allocation: {equity\_pct}%",

"Recommended max for your risk profile: 80%",

"Consider bonds, property, or cash"

],

urgency\_score: 50,

impact\_score: 60

})

# Rule 2: CGT harvesting opportunity (UK)

IF context.uk\_tax\_resident:

unrealized\_gains = context.investments.unrealized\_gains

cgt\_allowance = 3000 # 2024/25

cgt\_allowance\_used = context.investments.cgt\_allowance\_used

cgt\_allowance\_remaining = cgt\_allowance - cgt\_allowance\_used

IF unrealized\_gains > 0 AND cgt\_allowance\_remaining > 1000:

# Opportunity to harvest gains tax-free

recommendations.append({

category: 'INVESTMENT',

type: 'CGT\_HARVESTING',

priority: 'MEDIUM',

title: "Use your Capital Gains Tax allowance",

description: "You have £{cgt\_allowance\_remaining} of unused CGT allowance. Consider selling and rebuying investments to 'realize' gains tax-free and reset the cost base.",

estimated\_benefit: {

type: 'TAX\_EFFICIENCY',

description: "Future tax savings by resetting cost base",

amount\_gbp: cgt\_allowance\_remaining \* 0.20 # Approx future saving

},

action\_required: "Review portfolio for bed-and-breakfast opportunities",

reasoning: [

"Unrealized gains: £{unrealized\_gains}",

"CGT allowance remaining: £{cgt\_allowance\_remaining}",

"Realize gains tax-free before April 5"

],

urgency\_score: 55,

impact\_score: 50

})

# Rule 3: Investment in GIA when ISA allowance available

IF context.uk\_tax\_resident:

gia\_holdings = get\_gia\_holdings(context.user\_id)

isa\_allowance\_remaining = 20000 - context.savings.isa\_allowance\_used

IF gia\_holdings.value > 0 AND isa\_allowance\_remaining > 5000:

# Calculate potential tax saving

annual\_return\_assumption = gia\_holdings.value \* 0.06

tax\_on\_return = annual\_return\_assumption \* context.marginal\_tax\_rate\_uk

recommendations.append({

category: 'INVESTMENT',

type: 'TRANSFER\_TO\_ISA',

priority: 'MEDIUM',

title: "Transfer investments from GIA to ISA",

description: "You have £{gia\_value} in a General Investment Account that could be sheltered in an ISA.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Annual tax saving on returns",

amount\_gbp: tax\_on\_return,

currency: 'GBP'

},

action\_required: "Consider 'Bed and ISA' to transfer holdings",

reasoning: [

"GIA holdings: £{gia\_value}",

"ISA allowance available: £{isa\_allowance\_remaining}",

"Estimated annual tax saving: £{tax\_on\_return}"

],

urgency\_score: 50,

impact\_score: 60

})

# Rule 4: No EIS/SEIS investments for higher rate taxpayers (UK)

IF context.uk\_tax\_resident AND context.marginal\_tax\_rate\_uk >= 0.40:

eis\_seis\_investments = get\_eis\_seis\_investments(context.user\_id)

IF eis\_seis\_investments.total\_value = 0 AND context.net\_worth > 250000:

recommendations.append({

category: 'INVESTMENT',

type: 'CONSIDER\_EIS\_SEIS',

priority: 'LOW',

title: "Consider EIS/SEIS investments for tax relief",

description: "As a higher-rate taxpayer with substantial assets, EIS and SEIS investments offer attractive tax reliefs (30-50% income tax relief plus CGT exemptions).",

estimated\_benefit: {

type: 'TAX\_RELIEF',

description: "Income tax relief + CGT benefits",

potential\_relief: "30-50% of investment"

},

action\_required: "Speak with financial advisor about EIS/SEIS opportunities",

warnings: [

"High risk investments",

"Illiquid - 3-5 year hold required",

"Only suitable for experienced investors",

"Loss of capital is possible"

],

reasoning: [

"Higher-rate taxpayer: {marginal\_rate}%",

"EIS: 30% income tax relief",

"SEIS: 50% income tax relief",

"CGT exemption after 3 years"

],

urgency\_score: 20,

impact\_score: 40

})

RETURN recommendations

# ===== RETIREMENT RECOMMENDATIONS =====

FUNCTION generate\_retirement\_recommendations(context: UserContext) -> array[Recommendation]:

recommendations = []

# Rule 1: Not on track for retirement

IF NOT context.retirement.on\_track:

shortfall = calculate\_retirement\_shortfall(context)

additional\_monthly\_contribution = calculate\_required\_monthly\_contribution(shortfall, context.retirement.years\_to\_retirement)

recommendations.append({

category: 'RETIREMENT',

type: 'INCREASE\_PENSION\_CONTRIBUTIONS',

priority: 'HIGH',

title: "Increase pension contributions to meet retirement goals",

description: "Based on your retirement age of {target\_age}, you're projected to have a shortfall of £{shortfall}. Increasing contributions by £{additional} per month would get you on track.",

estimated\_benefit: {

type: 'RETIREMENT\_ADEQUACY',

description: "Close retirement savings gap",

shortfall\_closed: shortfall

},

action\_required: "Increase pension contributions by £{additional}/month",

monthly\_increase: additional\_monthly\_contribution,

reasoning: [

"Current pension pot: £{current\_pot}",

"Projected at retirement: £{projected}",

"Target: £{target}",

"Shortfall: £{shortfall}",

"Years to retirement: {years}"

],

urgency\_score: 80,

impact\_score: 90

})

# Rule 2: Unused pension annual allowance (UK)

IF context.uk\_tax\_resident:

annual\_allowance = calculate\_pension\_annual\_allowance(context)

allowance\_used = context.retirement.annual\_allowance\_used

allowance\_remaining = annual\_allowance - allowance\_used

IF allowance\_remaining > 10000:

# Calculate tax relief

tax\_relief = allowance\_remaining \* context.marginal\_tax\_rate\_uk

recommendations.append({

category: 'RETIREMENT',

type: 'USE\_PENSION\_ALLOWANCE',

priority: 'MEDIUM',

title: "Maximize pension tax relief",

description: "You have £{allowance\_remaining} of unused pension annual allowance. Pension contributions receive tax relief at your marginal rate of {marginal\_rate}%.",

estimated\_benefit: {

type: 'TAX\_RELIEF',

description: "Tax relief on contributions",

amount\_gbp: tax\_relief,

currency: 'GBP'

},

action\_required: "Consider additional pension contribution before April 5",

reasoning: [

"Annual allowance: £{annual\_allowance}",

"Used: £{allowance\_used}",

"Remaining: £{allowance\_remaining}",

"Tax relief at {marginal\_rate}%: £{tax\_relief}"

],

urgency\_score: 65,

impact\_score: 75

})

# Rule 3: Consider carry forward (UK)

IF context.uk\_tax\_resident:

carry\_forward\_available = calculate\_carry\_forward\_available(context.user\_id)

IF carry\_forward\_available > 10000:

recommendations.append({

category: 'RETIREMENT',

type: 'USE\_CARRY\_FORWARD',

priority: 'MEDIUM',

title: "Use pension carry forward allowance",

description: "You have £{carry\_forward} of unused allowance from previous years that you can carry forward for pension contributions.",

estimated\_benefit: {

type: 'TAX\_RELIEF',

description: "Additional tax relief opportunity",

amount\_gbp: carry\_forward\_available \* context.marginal\_tax\_rate\_uk

},

action\_required: "Consider one-off contribution using carry forward",

reasoning: [

"Carry forward from previous 3 years: £{carry\_forward}",

"Expires if not used",

"Tax relief available: £{tax\_relief}"

],

urgency\_score: 60,

impact\_score: 70

})

# Rule 4: Section 10C deduction not maximized (SA)

IF context.sa\_tax\_resident:

max\_deductible = MIN(context.income.annual \* 0.275, 350000)

current\_contributions = get\_sa\_retirement\_contributions(context.user\_id)

deduction\_unused = max\_deductible - current\_contributions

IF deduction\_unused > 20000:

tax\_saving = deduction\_unused \* context.marginal\_tax\_rate\_sa

recommendations.append({

category: 'RETIREMENT',

type: 'MAXIMIZE\_SECTION\_10C',

priority: 'MEDIUM',

title: "Maximize Section 10C retirement tax deduction",

description: "You can deduct up to R{max\_deductible} for retirement contributions (27.5% of income, max R350k). You have R{deduction\_unused} unused.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Tax saving on additional contributions",

amount\_zar: tax\_saving,

currency: 'ZAR'

},

action\_required: "Increase retirement annuity contributions",

reasoning: [

"Maximum deductible: R{max\_deductible}",

"Current contributions: R{current\_contributions}",

"Unused: R{deduction\_unused}",

"Tax saving potential: R{tax\_saving}"

],

urgency\_score: 60,

impact\_score: 70

})

# Rule 5: QROPS transfer consideration

IF context.uk\_tax\_resident = FALSE AND context.sa\_tax\_resident = TRUE:

uk\_pensions = get\_uk\_pensions(context.user\_id)

IF uk\_pensions.total\_value > 50000:

recommendations.append({

category: 'RETIREMENT',

type: 'CONSIDER\_QROPS',

priority: 'LOW',

title: "Consider QROPS transfer to South Africa",

description: "You have UK pensions worth £{uk\_pension\_value}. If you're permanently resident in SA, a QROPS transfer might provide benefits.",

estimated\_benefit: {

type: 'FLEXIBILITY',

description: "Currency matching and flexibility",

considerations: [

"Match pension to spending currency",

"SA retirement rules may be more flexible",

"Avoid UK IHT on pension death benefits"

]

},

action\_required: "Speak with cross-border pension specialist",

warnings: [

"Complex area - professional advice essential",

"Overseas Transfer Charge may apply (25%)",

"HMRC reporting requirements",

"Consider costs vs benefits carefully"

],

reasoning: [

"UK pension value: £{uk\_pension\_value}",

"Resident in SA",

"QROPS can provide currency matching"

],

urgency\_score: 30,

impact\_score: 50

})

RETURN recommendations

# ===== TAX OPTIMIZATION RECOMMENDATIONS =====

FUNCTION generate\_tax\_recommendations(context: UserContext) -> array[Recommendation]:

recommendations = []

# Rule 1: Personal allowance taper (UK)

IF context.uk\_tax\_resident AND context.income.annual > 100000:

excess = context.income.annual - 100000

allowance\_lost = excess / 2

effective\_tax\_rate\_on\_excess = 0.60 # 40% tax + 40% from lost allowance

# Pension contribution could reduce income below £100k

contribution\_needed = excess + 1000 # Slightly over to be safe

tax\_saved = contribution\_needed \* effective\_tax\_rate\_on\_excess

recommendations.append({

category: 'TAX',

type: 'AVOID\_ALLOWANCE\_TAPER',

priority: 'HIGH',

title: "Avoid personal allowance taper with pension contribution",

description: "Your income of £{income} triggers personal allowance taper. A pension contribution of £{contribution\_needed} would save £{tax\_saved} in tax (effective rate: 60%).",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Save tax and restore personal allowance",

amount\_gbp: tax\_saved,

currency: 'GBP'

},

action\_required: "Make pension contribution to reduce adjusted net income below £100,000",

reasoning: [

"Income: £{income}",

"Excess over £100k: £{excess}",

"Personal allowance lost: £{allowance\_lost}",

"Effective tax rate: 60%"

],

urgency\_score: 85,

impact\_score: 90

})

# Rule 2: Marriage allowance (UK)

IF context.uk\_tax\_resident:

spouse = get\_spouse(context.user\_id)

IF spouse EXISTS:

user\_uses\_full\_allowance = (context.income.annual > 12570)

spouse\_uses\_full\_allowance = (spouse.income.annual > 12570)

IF user\_uses\_full\_allowance XOR spouse\_uses\_full\_allowance:

# One uses full allowance, one doesn't - marriage allowance beneficial

transferable\_amount = 12570 \* 0.10 # £1,257

tax\_saving = transferable\_amount \* 0.20 # £251

recommendations.append({

category: 'TAX',

type: 'CLAIM\_MARRIAGE\_ALLOWANCE',

priority: 'MEDIUM',

title: "Claim Marriage Allowance",

description: "You can transfer £{transferable} of unused personal allowance to your spouse, saving £{saving} per year.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Annual tax saving",

amount\_gbp: tax\_saving,

currency: 'GBP'

},

action\_required: "Apply via HMRC to transfer allowance",

reasoning: [

"One spouse not using full allowance",

"10% transferable (£{transferable})",

"Tax saving: £{saving}/year"

],

urgency\_score: 50,

impact\_score: 40

})

# Rule 3: Dividend tax planning

IF context.uk\_tax\_resident AND context.investments.dividend\_income > 500:

dividend\_allowance = 500 # 2024/25

excess\_dividends = context.investments.dividend\_income - dividend\_allowance

IF excess\_dividends > 0:

# Could reduce dividends or move to ISA

recommendations.append({

category: 'TAX',

type: 'DIVIDEND\_TAX\_PLANNING',

priority: 'LOW',

title: "Optimize dividend tax strategy",

description: "You're paying dividend tax on £{excess} of dividends above the £500 allowance. Consider moving dividend-paying investments to an ISA.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Avoid dividend tax",

annual\_saving: excess\_dividends \* 0.0875 # Basic rate

},

action\_required: "Transfer dividend-paying investments to ISA",

reasoning: [

"Dividend income: £{dividend\_income}",

"Dividend allowance: £500",

"Taxable: £{excess}",

"Consider ISA wrapper"

],

urgency\_score: 40,

impact\_score: 45

})

RETURN recommendations

# ===== IHT PLANNING RECOMMENDATIONS =====

FUNCTION generate\_iht\_recommendations(context: UserContext) -> array[Recommendation]:

recommendations = []

# Rule 1: Significant IHT liability

IF context.iht.iht\_liability > 50000:

recommendations.append({

category: 'IHT',

type: 'IHT\_PLANNING\_NEEDED',

priority: 'HIGH',

title: "Reduce Inheritance Tax liability",

description: "Your estate has a projected IHT liability of £{iht\_liability}. There are strategies to reduce this through gifting, trusts, and reliefs.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Potential IHT reduction",

amount\_gbp: context.iht.iht\_liability \* 0.50 # Could reduce by 50%

},

action\_required: "Consult estate planning specialist",

suggested\_strategies: [

"Lifetime gifting (7-year rule)",

"Write life insurance in trust",

"Consider Business Property Relief assets",

"Utilize spousal exemption",

"Charitable giving (reduce rate to 36%)"

],

reasoning: [

"Estate value: £{estate\_value}",

"IHT liability: £{iht\_liability}",

"Multiple planning options available"

],

urgency\_score: 75,

impact\_score: 85

})

# Rule 2: Not using NRB efficiently (spouse)

IF context.uk\_tax\_resident:

spouse = get\_spouse(context.user\_id)

IF spouse EXISTS:

nrb\_utilization = assess\_nrb\_utilization(context.user\_id, spouse.id)

IF nrb\_utilization.inefficient:

recommendations.append({

category: 'IHT',

type: 'EQUALIZE\_ESTATES',

priority: 'MEDIUM',

title: "Equalize estates to use both NRBs",

description: "Your estates are unbalanced. Equalizing can ensure both spouses' Nil Rate Bands (£325,000 each) are utilized effectively.",

estimated\_benefit: {

type: 'TAX\_EFFICIENCY',

description: "Better use of allowances",

potential\_saving: "Up to £130,000"

},

action\_required: "Review asset ownership and consider rebalancing",

reasoning: [

"Imbalanced estates",

"Both NRBs should be utilized",

"Transferable NRB available but not optimal"

],

urgency\_score: 55,

impact\_score: 65

})

# Rule 3: No RNRB planning (UK)

IF context.uk\_tax\_resident:

has\_qualifying\_residence = check\_for\_qualifying\_residence(context.user\_id)

has\_direct\_descendants = (context.dependents.children > 0)

IF NOT has\_qualifying\_residence AND has\_direct\_descendants:

rnrb = 175000 # Additional IHT saving potential

iht\_saving = rnrb \* 0.40

recommendations.append({

category: 'IHT',

type: 'RNRB\_PLANNING',

priority: 'MEDIUM',

title: "Consider Residence Nil Rate Band planning",

description: "You have children but no qualifying residential property. Owning a qualifying residence that passes to descendants could save up to £{iht\_saving} in IHT.",

estimated\_benefit: {

type: 'TAX\_SAVING',

description: "Additional RNRB (£175,000)",

amount\_gbp: iht\_saving

},

action\_required: "Consider residential property ownership for RNRB",

reasoning: [

"RNRB available: £175,000",

"Must own qualifying residence",

"Must pass to direct descendants",

"Potential IHT saving: £{iht\_saving}"

],

urgency\_score: 50,

impact\_score: 60

})

# Rule 4: Gifts within 7 years approaching exempt status

gifts\_approaching\_7\_years = get\_gifts\_approaching\_7\_years(context.user\_id, months\_threshold: 12)

IF gifts\_approaching\_7\_years.count > 0:

recommendations.append({

category: 'IHT',

type: 'GIFTS\_BECOMING\_EXEMPT',

priority: 'INFO',

title: "Gifts approaching 7-year exemption",

description: "You have {count} gifts that will become fully exempt from IHT in the next 12 months (total value: £{total\_value}).",

estimated\_benefit: {

type: 'TAX\_EXEMPTION',

description: "Gifts becoming IHT-exempt",

total\_value: SUM(gifts\_approaching\_7\_years.value)

},

action\_required: "No action needed - informational",

gifts\_list: gifts\_approaching\_7\_years,

reasoning: [

"{count} gifts approaching 7 years",

"Will be fully exempt from IHT",

"Total value: £{total\_value}"

],

urgency\_score: 10,

impact\_score: 20

})

RETURN recommendations

# ===== SCORING AND PRIORITIZATION =====

FUNCTION score\_recommendations(

recommendations: array[Recommendation],

context: UserContext

) -> array[ScoredRecommendation]:

scored = []

FOR EACH rec IN recommendations:

# Base score from urgency and impact

base\_score = (rec.urgency\_score \* 0.4) + (rec.impact\_score \* 0.6)

# Adjust for user preferences

IF rec.category IN context.preferred\_recommendation\_types:

base\_score \*= 1.2

# Adjust for estimated benefit

benefit\_score = calculate\_benefit\_score(rec.estimated\_benefit)

# Adjust for user's historical acceptance rate of similar recommendations

acceptance\_adjustment = get\_category\_acceptance\_rate(context.user\_id, rec.category)

# Final score

final\_score = base\_score \* (1 + benefit\_score) \* acceptance\_adjustment

scored.append({

recommendation: rec,

score: final\_score,

base\_score: base\_score,

benefit\_score: benefit\_score,

acceptance\_adjustment: acceptance\_adjustment

})

RETURN scored

FUNCTION rank\_recommendations(recommendations: array[ScoredRecommendation]) -> array:

# Sort by score descending

RETURN SORT(recommendations, BY score DESC)

FUNCTION filter\_recommendations(

recommendations: array[ScoredRecommendation],

context: UserContext

) -> array:

filtered = []

FOR EACH rec IN recommendations:

# Remove duplicates

IF NOT already\_exists(filtered, rec.recommendation.type):

# Remove conflicting recommendations

IF NOT conflicts\_with\_existing(filtered, rec):

# Remove if already actioned recently

IF NOT recently\_actioned(context.user\_id, rec.recommendation.type):

# Remove if dismissed recently

IF NOT recently\_dismissed(context.user\_id, rec.recommendation.type):

filtered.append(rec)

RETURN filtered

**API Endpoints:**

POST /api/v1/advisory/generate-recommendations

GET /api/v1/advisory/recommendations/{userId}

PUT /api/v1/advisory/recommendations/{id}/status

POST /api/v1/advisory/recommendations/{id}/dismiss

POST /api/v1/advisory/recommendations/{id}/accept

GET /api/v1/advisory/recommendation-history

POST /api/v1/advisory/recalculate

**Data Models:**

TABLE: ai\_recommendations (reusing from earlier, expanded)

- id: UUID (PK)

- user\_id: UUID (FK to users)

- priority: ENUM('CRITICAL', 'HIGH', 'MEDIUM', 'LOW', 'INFO')

- category: ENUM('PROTECTION', 'SAVINGS', 'INVESTMENT', 'RETIREMENT', 'TAX', 'IHT', 'CROSS\_CUTTING')

- recommendation\_type: VARCHAR(100)

- title: VARCHAR(255)

- description: TEXT

- reasoning: JSON (array of reasoning points)

- estimated\_benefit: JSON

- action\_required: TEXT

- deadline: DATE (optional)

- urgency\_score: INTEGER (0-100)

- impact\_score: INTEGER (0-100)

- final\_score: DECIMAL(10,2)

- status: ENUM('NEW', 'VIEWED', 'IN\_PROGRESS', 'COMPLETED', 'DISMISSED')

- generated\_at: TIMESTAMP

- expires\_at: TIMESTAMP

- completed\_at: TIMESTAMP

- dismissed\_at: TIMESTAMP

- dismissal\_reason: TEXT

TABLE: recommendation\_user\_actions

- id: UUID (PK)

- recommendation\_id: UUID (FK to ai\_recommendations)

- user\_id: UUID (FK to users)

- action\_type: ENUM('VIEWED', 'ACCEPTED', 'DISMISSED', 'COMPLETED')

- action\_date: TIMESTAMP

- notes: TEXT

TABLE: recommendation\_effectiveness

- id: UUID (PK)

- recommendation\_type: VARCHAR(100)

- user\_id: UUID (FK to users)

- acceptance\_rate: DECIMAL(5,2)

- completion\_rate: DECIMAL(5,2)

- average\_time\_to\_action: INTEGER (days)

- measured\_benefit: DECIMAL(15,2)

- last\_calculated: TIMESTAMP

INDEX on ai\_recommendations(user\_id, status, priority, generated\_at DESC)

INDEX on recommendation\_user\_actions(user\_id, recommendation\_id, action\_type)

**10. AI ADVISORY ENGINE (Continued)**

**Feature 10.2: Goal-Based Financial Planning**

**Feature Name:** Comprehensive Goal Setting and Progress Tracking with AI-Driven Planning

**User Story:** As a user, I want to set specific financial goals (e.g., buy a house, retire comfortably, children's education) and receive a personalized plan showing whether I'm on track, what actions to take, and how to prioritize competing goals.

**Acceptance Criteria:**

* User can create multiple financial goals with target amounts and dates
* System calculates required monthly savings/contributions for each goal
* Track progress toward each goal in real-time
* Prioritize goals based on urgency and feasibility
* Identify conflicts between goals (insufficient resources)
* Generate actionable plans to achieve goals
* Adjust plans when circumstances change
* Show trade-offs between different goal scenarios
* Integration with all financial modules
* Multi-currency goal support

**Technical Requirements:**

* Goal modeling engine with time-value-of-money calculations
* Monte Carlo simulation for probability of success
* Optimization algorithm for resource allocation
* Constraint satisfaction solver for competing goals
* Real-time progress tracking
* Scenario comparison engine
* Inflation adjustment
* Investment return modeling

**Constraints:**

* Goals must be SMART (Specific, Measurable, Achievable, Relevant, Time-bound)
* Maximum 10 active goals per user
* Minimum goal timeline: 6 months
* Maximum goal timeline: 50 years
* Must consider existing commitments (expenses, debt payments)
* Cannot allocate more than 100% of available income

**Implementation Approach:**

SERVICE: GoalPlanningEngine

# ===== GOAL CREATION AND VALIDATION =====

FUNCTION create\_goal(user\_id: uuid, goal\_data: GoalInput) -> Goal:

# Validate goal data

validate\_goal\_data(goal\_data)

# Create goal

goal = {

id: generate\_uuid(),

user\_id: user\_id,

goal\_type: goal\_data.goal\_type,

title: goal\_data.title,

description: goal\_data.description,

target\_amount: goal\_data.target\_amount,

currency: goal\_data.currency,

target\_date: goal\_data.target\_date,

priority: goal\_data.priority,

current\_progress: 0,

linked\_accounts: goal\_data.linked\_accounts,

created\_at: NOW(),

status: 'ACTIVE'

}

# Calculate plan

plan = generate\_goal\_plan(user\_id, goal)

# Store goal and plan

store\_goal(goal)

store\_goal\_plan(goal.id, plan)

# Recalculate all goals (check for conflicts)

recalculate\_all\_goals(user\_id)

RETURN {

goal: goal,

plan: plan,

feasibility: plan.feasibility\_assessment

}

FUNCTION validate\_goal\_data(goal\_data: GoalInput) -> boolean:

# Validate target amount

IF goal\_data.target\_amount <= 0:

THROW ValidationError("Target amount must be positive")

# Validate target date

months\_to\_goal = calculate\_months\_between(NOW(), goal\_data.target\_date)

IF months\_to\_goal < 6:

THROW ValidationError("Goal must be at least 6 months in the future")

IF months\_to\_goal > 600: # 50 years

THROW ValidationError("Goal cannot be more than 50 years in the future")

# Check goal limit

active\_goals\_count = count\_active\_goals(goal\_data.user\_id)

IF active\_goals\_count >= 10:

THROW ValidationError("Maximum 10 active goals allowed")

RETURN TRUE

# ===== GOAL TYPES =====

ENUM GoalType:

EMERGENCY\_FUND

HOUSE\_PURCHASE

HOME\_IMPROVEMENT

DEBT\_REPAYMENT

VEHICLE\_PURCHASE

WEDDING

HOLIDAY\_TRAVEL

EDUCATION\_CHILDREN

EDUCATION\_SELF

RETIREMENT

BUSINESS\_START

INHERITANCE\_PLANNING

FINANCIAL\_INDEPENDENCE

CHARITABLE\_GIVING

OTHER

# ===== GOAL PLAN GENERATION =====

FUNCTION generate\_goal\_plan(user\_id: uuid, goal: Goal) -> GoalPlan:

# Get user financial context

context = build\_user\_context(user\_id)

# Calculate time to goal

months\_to\_goal = calculate\_months\_between(NOW(), goal.target\_date)

years\_to\_goal = months\_to\_goal / 12

# Determine appropriate savings vehicle

savings\_vehicle = determine\_savings\_vehicle(goal, years\_to\_goal, context)

# Estimate investment return

expected\_return = estimate\_expected\_return(savings\_vehicle, years\_to\_goal)

# Calculate required monthly contribution

required\_monthly = calculate\_required\_monthly\_contribution(

target\_amount: goal.target\_amount,

months: months\_to\_goal,

annual\_return: expected\_return,

current\_savings: goal.current\_progress

)

# Assess feasibility

feasibility = assess\_goal\_feasibility(user\_id, required\_monthly, context)

# Generate milestones

milestones = generate\_goal\_milestones(goal, months\_to\_goal)

# Generate recommendations

recommendations = generate\_goal\_recommendations(goal, required\_monthly, feasibility, context)

# Calculate probability of success

probability\_of\_success = calculate\_success\_probability(

goal: goal,

monthly\_contribution: required\_monthly,

expected\_return: expected\_return,

time\_horizon: years\_to\_goal,

market\_volatility: get\_market\_volatility(savings\_vehicle)

)

RETURN {

goal\_id: goal.id,

target\_amount: goal.target\_amount,

current\_progress: goal.current\_progress,

required\_monthly\_contribution: required\_monthly,

recommended\_savings\_vehicle: savings\_vehicle,

expected\_annual\_return: expected\_return,

months\_to\_goal: months\_to\_goal,

projected\_final\_amount: calculate\_future\_value(

required\_monthly,

months\_to\_goal,

expected\_return

),

probability\_of\_success: probability\_of\_success,

feasibility\_assessment: feasibility,

milestones: milestones,

recommendations: recommendations,

alternative\_scenarios: generate\_alternative\_scenarios(goal, context)

}

FUNCTION determine\_savings\_vehicle(

goal: Goal,

years\_to\_goal: decimal,

context: UserContext

) -> SavingsVehicle:

# Short-term goals (<2 years): Cash savings

IF years\_to\_goal < 2:

IF context.uk\_tax\_resident AND has\_isa\_allowance(context):

RETURN 'CASH\_ISA'

ELSE IF context.sa\_tax\_resident AND has\_tfsa\_allowance(context):

RETURN 'TFSA\_CASH'

ELSE:

RETURN 'SAVINGS\_ACCOUNT'

# Medium-term goals (2-5 years): Conservative investments

ELSE IF years\_to\_goal >= 2 AND years\_to\_goal < 5:

IF context.risk\_tolerance = 'LOW':

RETURN 'CASH\_ISA'

ELSE:

IF context.uk\_tax\_resident AND has\_isa\_allowance(context):

RETURN 'STOCKS\_SHARES\_ISA\_CONSERVATIVE'

ELSE:

RETURN 'BALANCED\_PORTFOLIO'

# Long-term goals (5+ years): Growth investments

ELSE:

IF goal.goal\_type = 'RETIREMENT':

IF context.uk\_tax\_resident:

RETURN 'PENSION\_SIPP'

ELSE IF context.sa\_tax\_resident:

RETURN 'RETIREMENT\_ANNUITY'

ELSE:

IF context.uk\_tax\_resident AND has\_isa\_allowance(context):

RETURN 'STOCKS\_SHARES\_ISA\_GROWTH'

ELSE IF context.sa\_tax\_resident AND has\_tfsa\_allowance(context):

RETURN 'TFSA\_EQUITY'

ELSE:

RETURN 'GROWTH\_PORTFOLIO'

FUNCTION estimate\_expected\_return(

vehicle: SavingsVehicle,

years\_to\_goal: decimal

) -> decimal:

# Conservative estimates (post-inflation, post-tax where applicable)

MATCH vehicle:

CASE 'SAVINGS\_ACCOUNT':

RETURN 0.01 # 1% real return

CASE 'CASH\_ISA', 'TFSA\_CASH':

RETURN 0.02 # 2% real return

CASE 'STOCKS\_SHARES\_ISA\_CONSERVATIVE', 'BALANCED\_PORTFOLIO':

RETURN 0.04 # 4% real return

CASE 'STOCKS\_SHARES\_ISA\_GROWTH', 'TFSA\_EQUITY', 'GROWTH\_PORTFOLIO':

RETURN 0.06 # 6% real return

CASE 'PENSION\_SIPP', 'RETIREMENT\_ANNUITY':

IF years\_to\_goal > 20:

RETURN 0.07 # 7% real return (long-term equity)

ELSE IF years\_to\_goal > 10:

RETURN 0.06

ELSE:

RETURN 0.05 # More conservative as approaching retirement

FUNCTION calculate\_required\_monthly\_contribution(

target\_amount: decimal,

months: integer,

annual\_return: decimal,

current\_savings: decimal

) -> decimal:

# Future value of current savings

monthly\_rate = annual\_return / 12

fv\_current\_savings = current\_savings \* POWER(1 + monthly\_rate, months)

# Required additional savings

additional\_needed = target\_amount - fv\_current\_savings

IF additional\_needed <= 0:

RETURN 0 # Already have enough

# Calculate monthly payment using future value of annuity formula

# FV = PMT × [((1 + r)^n - 1) / r]

# PMT = FV / [((1 + r)^n - 1) / r]

IF monthly\_rate = 0:

# No growth assumption

monthly\_contribution = additional\_needed / months

ELSE:

numerator = additional\_needed \* monthly\_rate

denominator = POWER(1 + monthly\_rate, months) - 1

monthly\_contribution = numerator / denominator

RETURN ROUND(monthly\_contribution, 2)

# ===== FEASIBILITY ASSESSMENT =====

FUNCTION assess\_goal\_feasibility(

user\_id: uuid,

required\_monthly: decimal,

context: UserContext

) -> FeasibilityAssessment:

# Calculate available monthly income

monthly\_income = context.income.monthly

# Calculate committed expenses

essential\_expenses = calculate\_essential\_expenses(user\_id)

existing\_goal\_contributions = calculate\_existing\_goal\_contributions(user\_id)

debt\_payments = calculate\_debt\_payments(user\_id)

committed\_monthly = essential\_expenses + existing\_goal\_contributions + debt\_payments

# Calculate available for new goals

available\_monthly = monthly\_income - committed\_monthly

# Calculate percentage of income required

percentage\_of\_income = (required\_monthly / monthly\_income) \* 100

# Feasibility assessment

IF required\_monthly <= 0:

feasibility = 'ALREADY\_ACHIEVED'

recommendation = "You already have sufficient savings for this goal"

ELSE IF required\_monthly <= available\_monthly:

IF percentage\_of\_income < 10:

feasibility = 'HIGHLY\_FEASIBLE'

recommendation = "This goal is easily achievable with your current income"

ELSE IF percentage\_of\_income < 20:

feasibility = 'FEASIBLE'

recommendation = "This goal is achievable with disciplined saving"

ELSE:

feasibility = 'CHALLENGING'

recommendation = "This goal will require significant commitment"

ELSE IF required\_monthly <= available\_monthly \* 1.2:

feasibility = 'REQUIRES\_ADJUSTMENT'

recommendation = "Consider reducing expenses or extending timeline"

ELSE:

feasibility = 'NOT\_FEASIBLE'

recommendation = "Goal not achievable with current income. Consider alternative approach"

RETURN {

feasibility\_level: feasibility,

required\_monthly: required\_monthly,

available\_monthly: available\_monthly,

shortfall: MAX(required\_monthly - available\_monthly, 0),

percentage\_of\_income: percentage\_of\_income,

recommendation: recommendation,

suggested\_adjustments: generate\_feasibility\_adjustments(

required\_monthly,

available\_monthly,

context

)

}

FUNCTION generate\_feasibility\_adjustments(

required\_monthly: decimal,

available\_monthly: decimal,

context: UserContext

) -> array[Adjustment]:

adjustments = []

shortfall = required\_monthly - available\_monthly

IF shortfall <= 0:

RETURN [] # No adjustments needed

# Adjustment 1: Extend timeline

current\_months = context.goal.months\_to\_goal

extended\_months = calculate\_months\_needed\_for\_affordable\_payment(

context.goal.target\_amount,

available\_monthly,

context.expected\_return

)

IF extended\_months > current\_months:

adjustments.append({

type: 'EXTEND\_TIMELINE',

description: "Extend goal timeline to {extended\_months} months",

new\_timeline\_months: extended\_months,

new\_monthly\_required: available\_monthly,

impact: "Makes goal achievable within current budget"

})

# Adjustment 2: Reduce target amount

affordable\_target = calculate\_affordable\_target\_amount(

available\_monthly,

current\_months,

context.expected\_return,

context.goal.current\_progress

)

IF affordable\_target < context.goal.target\_amount:

adjustments.append({

type: 'REDUCE\_TARGET',

description: "Reduce target from {original} to {affordable\_target}",

original\_target: context.goal.target\_amount,

adjusted\_target: affordable\_target,

reduction: context.goal.target\_amount - affordable\_target,

impact: "Achievable with current budget and timeline"

})

# Adjustment 3: Increase income

required\_income\_increase = shortfall

percentage\_increase = (required\_income\_increase / context.income.monthly) \* 100

adjustments.append({

type: 'INCREASE\_INCOME',

description: "Increase monthly income by {amount}",

amount\_needed: required\_income\_increase,

percentage\_increase: percentage\_increase,

suggestions: [

"Seek salary increase or promotion",

"Take on additional work or side income",

"Monetize skills or hobbies"

]

})

# Adjustment 4: Reduce expenses

potential\_savings = identify\_expense\_reduction\_opportunities(context.user\_id)

IF potential\_savings.total >= shortfall:

adjustments.append({

type: 'REDUCE\_EXPENSES',

description: "Reduce monthly expenses by {amount}",

amount\_to\_reduce: shortfall,

potential\_areas: potential\_savings.areas,

suggestions: potential\_savings.suggestions

})

# Adjustment 5: Deprioritize other goals

lower\_priority\_goals = get\_lower\_priority\_goals(context.user\_id, context.goal.priority)

IF lower\_priority\_goals.count > 0:

freed\_up\_budget = SUM(lower\_priority\_goals.monthly\_contribution)

IF freed\_up\_budget >= shortfall:

adjustments.append({

type: 'DEPRIORITIZE\_GOALS',

description: "Pause contributions to lower priority goals",

goals\_to\_pause: lower\_priority\_goals,

budget\_freed: freed\_up\_budget,

impact: "Allows focus on higher priority goal"

})

RETURN adjustments

# ===== GOAL MILESTONES =====

FUNCTION generate\_goal\_milestones(

goal: Goal,

months\_to\_goal: integer

) -> array[Milestone]:

milestones = []

# Create quarterly milestones

milestone\_count = CEIL(months\_to\_goal / 3) # Every 3 months

FOR i FROM 1 TO milestone\_count:

milestone\_date = ADD\_MONTHS(NOW(), i \* 3)

IF milestone\_date > goal.target\_date:

milestone\_date = goal.target\_date

progress\_percentage = (i / milestone\_count) \* 100

target\_amount\_at\_milestone = goal.target\_amount \* (progress\_percentage / 100)

milestones.append({

milestone\_number: i,

milestone\_date: milestone\_date,

target\_progress\_percentage: progress\_percentage,

target\_amount: target\_amount\_at\_milestone,

description: "{progress\_percentage}% toward goal",

achieved: FALSE

})

RETURN milestones

# ===== GOAL PROGRESS TRACKING =====

FUNCTION update\_goal\_progress(goal\_id: uuid) -> GoalProgress:

goal = get\_goal(goal\_id)

# Calculate current progress from linked accounts

current\_progress = 0

FOR EACH account\_id IN goal.linked\_accounts:

account\_balance = get\_account\_balance(account\_id)

current\_progress += account\_balance

# Update goal

goal.current\_progress = current\_progress

goal.progress\_percentage = (current\_progress / goal.target\_amount) \* 100

goal.last\_updated = NOW()

# Check milestones

milestones = get\_goal\_milestones(goal\_id)

FOR EACH milestone IN milestones:

IF NOT milestone.achieved AND current\_progress >= milestone.target\_amount:

milestone.achieved = TRUE

milestone.achieved\_date = NOW()

# Trigger celebration notification

send\_milestone\_notification(goal.user\_id, goal, milestone)

# Recalculate plan if significantly off track

plan = get\_goal\_plan(goal\_id)

# Calculate expected progress at this point

months\_elapsed = calculate\_months\_between(goal.created\_at, NOW())

expected\_progress = calculate\_expected\_progress(plan, months\_elapsed)

variance = current\_progress - expected\_progress

variance\_percentage = (variance / expected\_progress) \* 100

# If more than 10% off track, recalculate

IF ABS(variance\_percentage) > 10:

new\_plan = generate\_goal\_plan(goal.user\_id, goal)

update\_goal\_plan(goal\_id, new\_plan)

# Notify user of plan adjustment

send\_plan\_adjustment\_notification(goal.user\_id, goal, new\_plan)

store\_goal(goal)

RETURN {

goal\_id: goal\_id,

current\_progress: current\_progress,

progress\_percentage: goal.progress\_percentage,

target\_amount: goal.target\_amount,

on\_track: variance\_percentage >= -10,

variance: variance,

variance\_percentage: variance\_percentage,

milestones\_achieved: COUNT(milestones WHERE achieved = TRUE),

next\_milestone: FIRST(milestones WHERE achieved = FALSE),

updated\_plan: IF plan\_was\_updated THEN new\_plan ELSE NULL

}

# ===== MONTE CARLO SIMULATION FOR SUCCESS PROBABILITY =====

FUNCTION calculate\_success\_probability(

goal: Goal,

monthly\_contribution: decimal,

expected\_return: decimal,

time\_horizon: decimal,

market\_volatility: decimal

) -> decimal:

# Run Monte Carlo simulation (10,000 iterations)

simulation\_count = 10000

success\_count = 0

FOR iteration FROM 1 TO simulation\_count:

simulated\_final\_value = simulate\_investment\_outcome(

initial\_amount: goal.current\_progress,

monthly\_contribution: monthly\_contribution,

months: goal.months\_to\_goal,

expected\_annual\_return: expected\_return,

annual\_volatility: market\_volatility

)

IF simulated\_final\_value >= goal.target\_amount:

success\_count += 1

probability = (success\_count / simulation\_count) \* 100

RETURN ROUND(probability, 1)

FUNCTION simulate\_investment\_outcome(

initial\_amount: decimal,

monthly\_contribution: decimal,

months: integer,

expected\_annual\_return: decimal,

annual\_volatility: decimal

) -> decimal:

balance = initial\_amount

FOR month FROM 1 TO months:

# Generate random monthly return using normal distribution

monthly\_return = generate\_normal\_random(

mean: expected\_annual\_return / 12,

std\_dev: annual\_volatility / SQRT(12)

)

# Apply return to current balance

balance = balance \* (1 + monthly\_return)

# Add monthly contribution

balance = balance + monthly\_contribution

RETURN balance

# ===== GOAL PRIORITIZATION =====

FUNCTION prioritize\_user\_goals(user\_id: uuid) -> array[PrioritizedGoal]:

goals = get\_active\_goals(user\_id)

context = build\_user\_context(user\_id)

# Calculate priority score for each goal

scored\_goals = []

FOR EACH goal IN goals:

score = calculate\_goal\_priority\_score(goal, context)

scored\_goals.append({

goal: goal,

priority\_score: score.total\_score,

urgency\_score: score.urgency,

importance\_score: score.importance,

feasibility\_score: score.feasibility

})

# Sort by priority score (descending)

prioritized = SORT(scored\_goals, BY priority\_score DESC)

RETURN prioritized

FUNCTION calculate\_goal\_priority\_score(

goal: Goal,

context: UserContext

) -> GoalScore:

# Urgency score (based on time remaining)

months\_remaining = calculate\_months\_between(NOW(), goal.target\_date)

IF months\_remaining < 12:

urgency\_score = 100

ELSE IF months\_remaining < 24:

urgency\_score = 80

ELSE IF months\_remaining < 60:

urgency\_score = 60

ELSE:

urgency\_score = 40

# Importance score (based on goal type and user priority)

base\_importance = {

'EMERGENCY\_FUND': 100,

'DEBT\_REPAYMENT': 95,

'RETIREMENT': 90,

'HOUSE\_PURCHASE': 85,

'EDUCATION\_CHILDREN': 85,

'WEDDING': 70,

'VEHICLE\_PURCHASE': 60,

'HOLIDAY\_TRAVEL': 40,

'OTHER': 50

}

importance\_score = base\_importance[goal.goal\_type]

# Adjust for user-specified priority

IF goal.priority = 'HIGH':

importance\_score \*= 1.2

ELSE IF goal.priority = 'LOW':

importance\_score \*= 0.8

# Feasibility score (how achievable)

plan = get\_goal\_plan(goal.id)

MATCH plan.feasibility\_assessment.feasibility\_level:

CASE 'HIGHLY\_FEASIBLE':

feasibility\_score = 100

CASE 'FEASIBLE':

feasibility\_score = 80

CASE 'CHALLENGING':

feasibility\_score = 60

CASE 'REQUIRES\_ADJUSTMENT':

feasibility\_score = 40

CASE 'NOT\_FEASIBLE':

feasibility\_score = 20

# Life stage relevance

life\_stage\_multiplier = calculate\_life\_stage\_relevance(goal, context.age)

# Calculate total score (weighted average)

total\_score = (

urgency\_score \* 0.3 +

importance\_score \* 0.4 +

feasibility\_score \* 0.3

) \* life\_stage\_multiplier

RETURN {

urgency: urgency\_score,

importance: importance\_score,

feasibility: feasibility\_score,

life\_stage\_multiplier: life\_stage\_multiplier,

total\_score: total\_score

}

# ===== GOAL CONFLICT RESOLUTION =====

FUNCTION identify\_goal\_conflicts(user\_id: uuid) -> array[Conflict]:

goals = get\_active\_goals(user\_id)

context = build\_user\_context(user\_id)

# Calculate total required contributions

total\_required\_monthly = 0

FOR EACH goal IN goals:

plan = get\_goal\_plan(goal.id)

total\_required\_monthly += plan.required\_monthly\_contribution

# Calculate available budget

available\_monthly = calculate\_available\_monthly\_income(context)

conflicts = []

# Conflict 1: Insufficient income

IF total\_required\_monthly > available\_monthly:

shortfall = total\_required\_monthly - available\_monthly

conflicts.append({

conflict\_type: 'INSUFFICIENT\_INCOME',

severity: 'HIGH',

description: "Total required contributions (£{total\_required\_monthly}) exceed available income (£{available\_monthly})",

shortfall: shortfall,

affected\_goals: goals,

resolution\_options: [

{

option: 'PRIORITIZE\_GOALS',

description: "Focus on highest priority goals only",

action: generate\_prioritization\_recommendation(goals, available\_monthly)

},

{

option: 'EXTEND\_TIMELINES',

description: "Extend timelines for lower priority goals",

action: generate\_timeline\_extension\_recommendation(goals, shortfall)

},

{

option: 'INCREASE\_INCOME',

description: "Increase income by £{shortfall}/month",

required\_increase: shortfall

}

]

})

# Conflict 2: Competing deadlines

near\_term\_goals = FILTER(goals, WHERE months\_to\_goal < 12)

IF near\_term\_goals.count > 2:

conflicts.append({

conflict\_type: 'COMPETING\_DEADLINES',

severity: 'MEDIUM',

description: "{count} goals have deadlines within 12 months",

affected\_goals: near\_term\_goals,

resolution\_options: [

{

option: 'STAGGER\_GOALS',

description: "Extend some goal deadlines to spread out timing",

recommendation: generate\_staggering\_recommendation(near\_term\_goals)

}

]

})

# Conflict 3: Retirement underfunded while pursuing other goals

retirement\_goals = FILTER(goals, WHERE goal\_type = 'RETIREMENT')

IF retirement\_goals.exists:

retirement\_goal = retirement\_goals[0]

retirement\_plan = get\_goal\_plan(retirement\_goal.id)

IF retirement\_plan.feasibility\_assessment.feasibility\_level IN ['REQUIRES\_ADJUSTMENT', 'NOT\_FEASIBLE']:

non\_retirement\_contribution = total\_required\_monthly - retirement\_plan.required\_monthly\_contribution

IF non\_retirement\_contribution > retirement\_plan.required\_monthly\_contribution:

conflicts.append({

conflict\_type: 'RETIREMENT\_UNDERPRIORITIZED',

severity: 'HIGH',

description: "Retirement goal underfunded while pursuing short-term goals",

affected\_goals: [retirement\_goal],

resolution\_options: [

{

option: 'INCREASE\_RETIREMENT\_PRIORITY',

description: "Reduce other goal contributions to adequately fund retirement",

recommended\_allocation: {

retirement: retirement\_plan.required\_monthly\_contribution,

other\_goals: available\_monthly - retirement\_plan.required\_monthly\_contribution

}

}

]

})

RETURN conflicts

# ===== GOAL RECOMMENDATIONS =====

FUNCTION generate\_goal\_recommendations(

goal: Goal,

required\_monthly: decimal,

feasibility: FeasibilityAssessment,

context: UserContext

) -> array[Recommendation]:

recommendations = []

# Recommendation 1: Open appropriate account

IF goal.linked\_accounts.count = 0:

suggested\_vehicle = determine\_savings\_vehicle(goal, goal.years\_to\_goal, context)

recommendations.append({

type: 'OPEN\_ACCOUNT',

priority: 'HIGH',

title: "Open a {vehicle} for this goal",

description: "We recommend a {vehicle} to maximize returns and tax efficiency",

vehicle: suggested\_vehicle,

action: "Open account and link to this goal"

})

# Recommendation 2: Set up automatic contributions

has\_automatic\_transfer = check\_automatic\_transfers(goal.id)

IF NOT has\_automatic\_transfer:

recommendations.append({

type: 'AUTOMATE\_SAVINGS',

priority: 'HIGH',

title: "Set up automatic monthly transfer of £{required\_monthly}",

description: "Automate your savings to ensure consistent progress",

amount: required\_monthly,

action: "Set up standing order"

})

# Recommendation 3: One-off contribution if windfall

IF context.liquid\_assets > goal.target\_amount \* 0.5:

lump\_sum\_contribution = MIN(

goal.target\_amount \* 0.25,

context.liquid\_assets \* 0.2

)

recommendations.append({

type: 'LUMP\_SUM\_CONTRIBUTION',

priority: 'MEDIUM',

title: "Consider one-off contribution of £{lump\_sum}",

description: "You have liquid assets that could accelerate this goal",

amount: lump\_sum\_contribution,

impact: "Would reduce monthly contribution to £{new\_monthly}",

new\_monthly: calculate\_required\_monthly\_with\_lump\_sum(

goal,

lump\_sum\_contribution

)

})

# Recommendation 4: Tax-efficient approach

IF goal.goal\_type = 'RETIREMENT':

IF context.uk\_tax\_resident:

tax\_relief = required\_monthly \* context.marginal\_tax\_rate\_uk

recommendations.append({

type: 'MAXIMIZE\_TAX\_RELIEF',

priority: 'HIGH',

title: "Maximize pension tax relief",

description: "Pension contributions receive {rate}% tax relief",

tax\_relief\_rate: context.marginal\_tax\_rate\_uk \* 100,

monthly\_tax\_relief: tax\_relief,

action: "Ensure contributions made via pension scheme"

})

# Recommendation 5: If not feasible, suggest adjustments

IF feasibility.feasibility\_level IN ['REQUIRES\_ADJUSTMENT', 'NOT\_FEASIBLE']:

FOR EACH adjustment IN feasibility.suggested\_adjustments:

recommendations.append({

type: 'ADJUST\_GOAL',

priority: 'HIGH',

title: adjustment.description,

description: adjustment.impact,

adjustment: adjustment

})

RETURN recommendations

# ===== ALTERNATIVE SCENARIOS =====

FUNCTION generate\_alternative\_scenarios(

goal: Goal,

context: UserContext

) -> array[Scenario]:

scenarios = []

# Scenario 1: Aggressive (higher contributions)

aggressive\_monthly = plan.required\_monthly\_contribution \* 1.5

aggressive\_timeline = calculate\_months\_needed(goal.target\_amount, aggressive\_monthly, plan.expected\_annual\_return)

scenarios.append({

scenario\_name: 'AGGRESSIVE',

description: "Achieve goal faster with higher contributions",

monthly\_contribution: aggressive\_monthly,

timeline\_months: aggressive\_timeline,

timeline\_reduction: goal.months\_to\_goal - aggressive\_timeline,

total\_contributed: aggressive\_monthly \* aggressive\_timeline,

final\_amount: goal.target\_amount,

feasible: aggressive\_monthly <= context.available\_monthly

})

# Scenario 2: Conservative (lower contributions, longer timeline)

conservative\_monthly = plan.required\_monthly\_contribution \* 0.75

conservative\_timeline = calculate\_months\_needed(goal.target\_amount, conservative\_monthly, plan.expected\_annual\_return)

scenarios.append({

scenario\_name: 'CONSERVATIVE',

description: "Lower monthly commitment with extended timeline",

monthly\_contribution: conservative\_monthly,

timeline\_months: conservative\_timeline,

timeline\_extension: conservative\_timeline - goal.months\_to\_goal,

total\_contributed: conservative\_monthly \* conservative\_timeline,

final\_amount: goal.target\_amount,

feasible: TRUE # Lower contribution always more feasible

})

# Scenario 3: Stretch target (achieve more than target)

stretch\_target = goal.target\_amount \* 1.25

stretch\_monthly = calculate\_required\_monthly\_contribution(

stretch\_target,

goal.months\_to\_goal,

plan.expected\_annual\_return,

goal.current\_progress

)

scenarios.append({

scenario\_name: 'STRETCH',

description: "Exceed your target by 25%",

monthly\_contribution: stretch\_monthly,

timeline\_months: goal.months\_to\_goal,

total\_contributed: stretch\_monthly \* goal.months\_to\_goal,

final\_amount: stretch\_target,

additional\_amount: stretch\_target - goal.target\_amount,

feasible: stretch\_monthly <= context.available\_monthly

})

# Scenario 4: Current pace (if contributing but not enough)

IF goal.current\_contributions > 0 AND goal.current\_contributions < plan.required\_monthly\_contribution:

current\_pace\_timeline = calculate\_months\_needed(

goal.target\_amount,

goal.current\_contributions,

plan.expected\_annual\_return

)

current\_pace\_final = calculate\_future\_value(

goal.current\_contributions,

goal.months\_to\_goal,

plan.expected\_annual\_return

)

scenarios.append({

scenario\_name: 'CURRENT\_PACE',

description: "Continue at current contribution rate",

monthly\_contribution: goal.current\_contributions,

timeline\_months: current\_pace\_timeline,

final\_amount: current\_pace\_final,

shortfall: goal.target\_amount - current\_pace\_final,

feasible: TRUE,

warning: "Will not reach target on time"

})

RETURN scenarios

**API Endpoints:**

# Goal Management

POST /api/v1/goals

PUT /api/v1/goals/{id}

DELETE /api/v1/goals/{id}

GET /api/v1/goals/{userId}

GET /api/v1/goals/{id}

# Goal Planning

GET /api/v1/goals/{id}/plan

POST /api/v1/goals/{id}/recalculate-plan

GET /api/v1/goals/{id}/scenarios

POST /api/v1/goals/{id}/select-scenario

# Progress Tracking

POST /api/v1/goals/{id}/update-progress

GET /api/v1/goals/{id}/progress

GET /api/v1/goals/{id}/milestones

POST /api/v1/goals/{id}/milestones/{milestoneId}/complete

# Goal Prioritization

GET /api/v1/goals/{userId}/prioritized

POST /api/v1/goals/{userId}/resolve-conflicts

GET /api/v1/goals/{userId}/conflicts

# Contributions

POST /api/v1/goals/{id}/contributions

GET /api/v1/goals/{id}/contributions/history

POST /api/v1/goals/{id}/link-account

DELETE /api/v1/goals/{id}/unlink-account/{accountId}

# Recommendations

GET /api/v1/goals/{id}/recommendations

POST /api/v1/goals/{id}/recommendations/{recId}/accept

**Data Models:**

TABLE: financial\_goals

- id: UUID (PK)

- user\_id: UUID (FK to users)

- goal\_type: ENUM('EMERGENCY\_FUND', 'HOUSE\_PURCHASE', 'RETIREMENT', etc.)

- title: VARCHAR(255)

- description: TEXT

- target\_amount: DECIMAL(15,2)

- currency: CHAR(3)

- target\_date: DATE

- priority: ENUM('HIGH', 'MEDIUM', 'LOW')

- current\_progress: DECIMAL(15,2)

- progress\_percentage: DECIMAL(5,2)

- status: ENUM('ACTIVE', 'COMPLETED', 'PAUSED', 'CANCELLED')

- created\_at: TIMESTAMP

- updated\_at: TIMESTAMP

- completed\_at: TIMESTAMP

TABLE: goal\_plans

- goal\_id: UUID (PK, FK to financial\_goals)

- required\_monthly\_contribution: DECIMAL(10,2)

- recommended\_savings\_vehicle: VARCHAR(100)

- expected\_annual\_return: DECIMAL(5,4)

- months\_to\_goal: INTEGER

- projected\_final\_amount: DECIMAL(15,2)

- probability\_of\_success: DECIMAL(5,2)

- feasibility\_level: ENUM('HIGHLY\_FEASIBLE', 'FEASIBLE', 'CHALLENGING', 'REQUIRES\_ADJUSTMENT', 'NOT\_FEASIBLE')

- plan\_generated\_at: TIMESTAMP

- last\_recalculated: TIMESTAMP

TABLE: goal\_milestones

- id: UUID (PK)

- goal\_id: UUID (FK to financial\_goals)

- milestone\_number: INTEGER

- milestone\_date: DATE

- target\_progress\_percentage: DECIMAL(5,2)

- target\_amount: DECIMAL(15,2)

- description: VARCHAR(255)

- achieved: BOOLEAN DEFAULT FALSE

- achieved\_date: DATE

- created\_at: TIMESTAMP

TABLE: goal\_linked\_accounts

- id: UUID (PK)

- goal\_id: UUID (FK to financial\_goals)

- account\_type: ENUM('SAVINGS\_ACCOUNT', 'ISA', 'INVESTMENT\_ACCOUNT', 'PENSION')

- account\_id: UUID (FK to respective account tables)

- linked\_at: TIMESTAMP

TABLE: goal\_contributions

- id: UUID (PK)

- goal\_id: UUID (FK to financial\_goals)

- contribution\_date: DATE

- contribution\_amount: DECIMAL(10,2)

- contribution\_type: ENUM('MANUAL', 'AUTOMATIC', 'LUMP\_SUM', 'WINDFALL')

- source\_account\_id: UUID

- notes: TEXT

- created\_at: TIMESTAMP

TABLE: goal\_scenarios

- id: UUID (PK)

- goal\_id: UUID (FK to financial\_goals)

- scenario\_name: VARCHAR(100)

- monthly\_contribution: DECIMAL(10,2)

- timeline\_months: INTEGER

- final\_amount: DECIMAL(15,2)

- probability\_of\_success: DECIMAL(5,2)

- is\_selected: BOOLEAN DEFAULT FALSE

- created\_at: TIMESTAMP

TABLE: goal\_conflicts

- id: UUID (PK)

- user\_id: UUID (FK to users)

- conflict\_type: ENUM('INSUFFICIENT\_INCOME', 'COMPETING\_DEADLINES', 'RETIREMENT\_UNDERPRIORITIZED')

- severity: ENUM('HIGH', 'MEDIUM', 'LOW')

- description: TEXT

- affected\_goals: JSON (array of goal\_ids)

- resolution\_options: JSON

- resolved: BOOLEAN DEFAULT FALSE

- resolved\_at: TIMESTAMP

- created\_at: TIMESTAMP

VIEW: v\_goal\_progress\_summary

- goal\_id

- user\_id

- title

- target\_amount

- current\_progress

- progress\_percentage

- required\_monthly\_contribution

- months\_to\_goal

- months\_elapsed

- on\_track: BOOLEAN

- next\_milestone\_date

- next\_milestone\_amount

INDEX on financial\_goals(user\_id, status)

INDEX on goal\_contributions(goal\_id, contribution\_date DESC)

INDEX on goal\_milestones(goal\_id, achieved, milestone\_date)

INDEX on goal\_conflicts(user\_id, resolved, created\_at DESC)

**Performance Considerations:**

* Monte Carlo simulation: 10,000 iterations, target <1 second
* Goal plan recalculation: Cache unless data changes significantly
* Progress updates: Real-time for linked accounts
* Conflict detection: Run daily batch job + on-demand
* Expected goals per user: 1-10 active
* Milestone checks: Trigger on contribution or weekly batch
* Scenario generation: Pre-calculate common scenarios

**Feature 10.3: Scenario Analysis & What-If Modeling**

**Feature Name:** Interactive Financial Scenario Modeling and Comparison

**User Story:** As a user, I want to model different financial scenarios (e.g., "What if I retire at 60 vs 65?", "What if I move to SA permanently?", "What if I sell my business?") and compare outcomes so that I can make informed decisions about major life changes.

**Acceptance Criteria:**

* Model major life events and financial decisions
* Compare multiple scenarios side-by-side
* Show impact across all modules (tax, retirement, IHT, etc.)
* Save and revisit scenarios
* Share scenarios with advisors
* Export scenario comparisons
* Real-time recalculation as assumptions change
* Risk analysis for each scenario
* Probability-weighted outcomes

**Technical Requirements:**

* Scenario engine with deep copy of user financial state
* Parallel calculation for multiple scenarios
* Differential analysis (what changed)
* Sensitivity analysis
* Monte Carlo simulation for uncertainty
* Scenario versioning and history
* Complex tax rule application across scenarios

**Constraints:**

* Maximum 5 active scenarios per user
* Scenarios expire after 90 days if not accessed
* Calculation time: <5 seconds per scenario
* Can model up to 30 years into future
* Must maintain data consistency across scenarios

**Implementation Approach:**

SERVICE: ScenarioAnalysisEngine

# ===== SCENARIO TYPES =====

ENUM ScenarioType:

RETIREMENT\_AGE\_CHANGE

RELOCATION

CAREER\_CHANGE

BUSINESS\_SALE

PROPERTY\_PURCHASE

PROPERTY\_SALE

INHERITANCE\_RECEIVED

MAJOR\_EXPENSE

INVESTMENT\_STRATEGY\_CHANGE

TAX\_RESIDENCY\_CHANGE

DIVORCE\_SEPARATION

MARRIAGE

CHILD\_BIRTH

CUSTOM

# ===== SCENARIO CREATION =====

FUNCTION create\_scenario(

user\_id: uuid,

scenario\_config: ScenarioConfig

) -> Scenario:

# Validate scenario count

active\_scenarios = count\_active\_scenarios(user\_id)

IF active\_scenarios >= 5:

THROW ValidationError("Maximum 5 active scenarios allowed")

# Create baseline (current state)

baseline = create\_baseline\_snapshot(user\_id)

# Create scenario with modifications

scenario = {

id: generate\_uuid(),

user\_id: user\_id,

name: scenario\_config.name,

description: scenario\_config.description,

scenario\_type: scenario\_config.type,

baseline\_snapshot\_id: baseline.id,

assumptions: scenario\_config.assumptions,

modifications: scenario\_config.modifications,

created\_at: NOW(),

last\_accessed: NOW(),

expires\_at: NOW() + 90\_DAYS

}

# Apply scenario modifications and calculate

scenario\_state = apply\_scenario\_modifications(baseline, scenario.modifications)

scenario\_results = calculate\_scenario\_outcomes(scenario\_state, scenario.assumptions)

# Store scenario

store\_scenario(scenario)

store\_scenario\_results(scenario.id, scenario\_results)

RETURN {

scenario: scenario,

baseline: baseline,

results: scenario\_results,

comparison: compare\_to\_baseline(baseline, scenario\_results)

}

FUNCTION create\_baseline\_snapshot(user\_id: uuid) -> BaselineSnapshot:

# Capture complete current state

context = build\_user\_context(user\_id)

baseline = {

id: generate\_uuid(),

user\_id: user\_id,

snapshot\_date: NOW(),

# Demographics

age: context.age,

dependents: context.dependents,

# Tax status

tax\_status: {

uk\_resident: context.uk\_tax\_resident,

sa\_resident: context.sa\_tax\_resident,

domicile: context.domicile,

deemed\_domicile: context.deemed\_domicile

},

# Financial position

income: context.income,

net\_worth: context.net\_worth,

assets: get\_all\_assets(user\_id),

liabilities: get\_all\_liabilities(user\_id),

# Module data

protection: context.protection,

savings: context.savings,

investments: context.investments,

retirement: context.retirement,

iht: context.iht,

# Goals

goals: get\_all\_goals(user\_id),

# Calculated metrics

current\_tax\_liability: calculate\_total\_tax\_liability(user\_id, current\_tax\_year()),

retirement\_readiness: assess\_retirement\_readiness(user\_id),

iht\_liability: get\_iht\_liability(user\_id)

}

store\_baseline\_snapshot(baseline)

RETURN baseline

# ===== SCENARIO MODIFICATIONS =====

FUNCTION apply\_scenario\_modifications(

baseline: BaselineSnapshot,

modifications: array[Modification]

) -> ScenarioState:

# Deep copy baseline

scenario\_state = deep\_copy(baseline)

# Apply each modification

FOR EACH mod IN modifications:

MATCH mod.type:

CASE 'CHANGE\_AGE':

scenario\_state.age = mod.new\_age

scenario\_state.years\_elapsed = mod.new\_age - baseline.age

CASE 'CHANGE\_INCOME':

scenario\_state.income = mod.new\_income

CASE 'CHANGE\_TAX\_RESIDENCY':

scenario\_state.tax\_status.uk\_resident = mod.uk\_resident

scenario\_state.tax\_status.sa\_resident = mod.sa\_resident

# Recalculate domicile based on new residency

scenario\_state.tax\_status = determine\_tax\_status\_future(

scenario\_state,

mod.change\_date

)

CASE 'RETIRE':

scenario\_state.retirement.retired = TRUE

scenario\_state.retirement.retirement\_date = mod.retirement\_date

scenario\_state.retirement.retirement\_age = calculate\_age\_at\_date(

baseline.date\_of\_birth,

mod.retirement\_date

)

# Stop pension contributions

scenario\_state.retirement.active\_contributions = 0

# Start drawing pension income

scenario\_state.retirement.drawing\_income = TRUE

scenario\_state.income.pension = calculate\_pension\_income(scenario\_state)

# Remove employment income

scenario\_state.income.employment = 0

CASE 'SELL\_ASSET':

# Remove asset from portfolio

asset\_to\_sell = find\_asset(scenario\_state.assets, mod.asset\_id)

scenario\_state.assets = remove\_asset(scenario\_state.assets, mod.asset\_id)

# Calculate CGT on sale

capital\_gain = mod.sale\_price - asset\_to\_sell.cost\_basis

cgt = calculate\_cgt(capital\_gain, scenario\_state.tax\_status)

# Add proceeds to cash (minus CGT)

net\_proceeds = mod.sale\_price - cgt

scenario\_state.savings.cash += net\_proceeds

CASE 'PURCHASE\_ASSET':

# Add new asset

new\_asset = {

id: generate\_uuid(),

type: mod.asset\_type,

description: mod.description,

value: mod.purchase\_price,

purchase\_date: mod.purchase\_date,

cost\_basis: mod.purchase\_price

}

scenario\_state.assets.append(new\_asset)

# Reduce cash

scenario\_state.savings.cash -= mod.purchase\_price

# Add mortgage if applicable

IF mod.mortgage\_amount > 0:

new\_mortgage = {

id: generate\_uuid(),

linked\_asset\_id: new\_asset.id,

amount: mod.mortgage\_amount,

interest\_rate: mod.mortgage\_rate,

term\_months: mod.mortgage\_term

}

scenario\_state.liabilities.append(new\_mortgage)

CASE 'RECEIVE\_INHERITANCE':

scenario\_state.savings.cash += mod.inheritance\_amount

scenario\_state.net\_worth += mod.inheritance\_amount

CASE 'CHANGE\_INVESTMENT\_STRATEGY':

# Rebalance portfolio to new allocation

scenario\_state.investments.asset\_allocation = mod.new\_allocation

scenario\_state.investments.expected\_return = calculate\_portfolio\_return(mod.new\_allocation)

CASE 'ADD\_GOAL':

new\_goal = mod.goal\_data

scenario\_state.goals.append(new\_goal)

CASE 'MODIFY\_GOAL':

goal = find\_goal(scenario\_state.goals, mod.goal\_id)

goal.target\_amount = mod.new\_target\_amount

goal.target\_date = mod.new\_target\_date

CASE 'SELL\_BUSINESS':

# Remove business asset

business\_asset = find\_business\_asset(scenario\_state.assets, mod.business\_id)

scenario\_state.assets = remove\_asset(scenario\_state.assets, mod.business\_id)

# Calculate tax (BPR may apply if held >2 years)

IF business\_asset.held\_years >= 2:

bpr\_relief = business\_asset.value \* 1.00 # 100% BPR

ELSE:

bpr\_relief = 0

taxable\_gain = (mod.sale\_price - business\_asset.cost\_basis) - bpr\_relief

cgt = calculate\_cgt(taxable\_gain, scenario\_state.tax\_status)

# Add proceeds

net\_proceeds = mod.sale\_price - cgt

scenario\_state.savings.cash += net\_proceeds

# Remove business income

scenario\_state.income.business = 0

CASE 'RELOCATE':

# Change country of residence

scenario\_state.location = mod.new\_country

scenario\_state.tax\_status = determine\_tax\_status\_after\_relocation(

scenario\_state,

mod.new\_country,

mod.relocation\_date

)

# Update asset situs where relevant

scenario\_state.assets = update\_asset\_situs(scenario\_state.assets, mod.new\_country)

CASE 'GIFT\_ASSET':

# Remove asset from estate

gifted\_asset = find\_asset(scenario\_state.assets, mod.asset\_id)

scenario\_state.assets = remove\_asset(scenario\_state.assets, mod.asset\_id)

# Record PET (Potentially Exempt Transfer)

pet = {

gift\_date: mod.gift\_date,

value: gifted\_asset.value,

recipient: mod.recipient,

becomes\_exempt: mod.gift\_date + 7\_YEARS

}

scenario\_state.iht.pets.append(pet)

CASE 'MARRIAGE':

scenario\_state.marital\_status = 'MARRIED'

scenario\_state.spouse = mod.spouse\_data

# Potential tax changes (marriage allowance, IHT spouse exemption)

scenario\_state.tax\_benefits.marriage\_allowance\_eligible = TRUE

CASE 'DIVORCE':

scenario\_state.marital\_status = 'DIVORCED'

scenario\_state.spouse = NULL

# Asset division

IF mod.asset\_split\_percentage:

scenario\_state.assets = apply\_asset\_split(

scenario\_state.assets,

mod.asset\_split\_percentage

)

scenario\_state.net\_worth \*= (1 - mod.asset\_split\_percentage)

CASE 'CHILD\_BIRTH':

scenario\_state.dependents.children += 1

scenario\_state.dependents.list.append({

name: mod.child\_name,

date\_of\_birth: mod.birth\_date,

dependent: TRUE

})

# Increase recommended life cover

scenario\_state.protection.recommended\_life\_cover += 100000

# Add education goal

education\_goal = {

type: 'EDUCATION\_CHILDREN',

target\_amount: 50000, # Estimate

target\_date: mod.birth\_date + 18\_YEARS

}

scenario\_state.goals.append(education\_goal)

CASE 'CUSTOM\_MODIFICATION':

# Allow arbitrary modifications via custom logic

apply\_custom\_modification(scenario\_state, mod.custom\_logic)

RETURN scenario\_state

# ===== SCENARIO CALCULATION =====

FUNCTION calculate\_scenario\_outcomes(

scenario\_state: ScenarioState,

assumptions: Assumptions

) -> ScenarioResults:

results = {

scenario\_state: scenario\_state,

projections: {},

comparisons: {},

risks: {}

}

# Project forward

projection\_years = assumptions.projection\_years OR 30

results.projections = {

net\_worth: project\_net\_worth(scenario\_state, projection\_years, assumptions),

income: project\_income(scenario\_state, projection\_years, assumptions),

expenses: project\_expenses(scenario\_state, projection\_years, assumptions),

tax: project\_tax\_liability(scenario\_state, projection\_years, assumptions),

retirement: project\_retirement\_income(scenario\_state, assumptions),

iht: project\_iht\_liability(scenario\_state, assumptions),

goals: assess\_goal\_achievement(scenario\_state, assumptions)

}

# Calculate key metrics

results.metrics = {

lifetime\_tax\_paid: SUM(results.projections.tax),

total\_retirement\_income: SUM(results.projections.retirement),

retirement\_adequacy\_ratio: calculate\_retirement\_adequacy(results.projections),

estate\_value\_at\_death: calculate\_estate\_value\_at\_death(scenario\_state, assumptions),

iht\_liability\_at\_death: calculate\_iht\_at\_death(scenario\_state, assumptions),

goals\_achieved\_count: COUNT(results.projections.goals WHERE achieved = TRUE),

goals\_achieved\_percentage: (goals\_achieved / total\_goals) \* 100

}

# Risk analysis

results.risks = {

longevity\_risk: assess\_longevity\_risk(scenario\_state, results.projections),

market\_risk: assess\_market\_risk(scenario\_state, assumptions),

inflation\_risk: assess\_inflation\_risk(scenario\_state, assumptions),

tax\_change\_risk: assess\_tax\_change\_risk(scenario\_state),

health\_risk: assess\_health\_risk(scenario\_state)

}

# Monte Carlo simulation for uncertainty

results.probability\_distribution = run\_monte\_carlo\_simulation(

scenario\_state,

assumptions,

simulations: 5000

)

RETURN results

FUNCTION project\_net\_worth(

state: ScenarioState,

years: integer,

assumptions: Assumptions

) -> array[YearlyNetWorth]:

projections = []

current\_state = state

FOR year FROM 1 TO years:

# Project asset growth

investment\_return = assumptions.investment\_return OR 0.06

assets\_growth = current\_state.assets.total \* investment\_return

# Project income and savings

annual\_income = current\_state.income.total

annual\_expenses = current\_state.expenses.total

annual\_savings = annual\_income - annual\_expenses - current\_state.tax.total

# Update net worth

new\_net\_worth = current\_state.net\_worth + assets\_growth + annual\_savings

# Age by one year

current\_state.age += 1

current\_state.net\_worth = new\_net\_worth

# Check for life events

IF current\_state.age = current\_state.retirement.planned\_retirement\_age:

current\_state = apply\_retirement\_changes(current\_state)

# Adjust for inflation

inflation\_rate = assumptions.inflation\_rate OR 0.025

current\_state.expenses.total \*= (1 + inflation\_rate)

projections.append({

year: year,

age: current\_state.age,

net\_worth: new\_net\_worth,

assets: current\_state.assets.total,

liabilities: current\_state.liabilities.total,

income: annual\_income,

expenses: annual\_expenses,

savings: annual\_savings

})

RETURN projections

FUNCTION project\_retirement\_income(

state: ScenarioState,

assumptions: Assumptions

) -> RetirementProjection:

retirement\_age = state.retirement.planned\_retirement\_age

# Pension pot at retirement

years\_to\_retirement = retirement\_age - state.age

IF years\_to\_retirement <= 0:

# Already retired

pension\_pot = state.retirement.current\_pension\_pot

ELSE:

# Project pension pot growth

annual\_contributions = state.retirement.annual\_contributions

growth\_rate = assumptions.pension\_growth\_rate OR 0.06

pension\_pot = calculate\_future\_value(

current\_value: state.retirement.current\_pension\_pot,

annual\_contribution: annual\_contributions,

years: years\_to\_retirement,

growth\_rate: growth\_rate

)

# Calculate tax-free lump sum (25%)

tax\_free\_lump\_sum = pension\_pot \* 0.25

remaining\_pension = pension\_pot \* 0.75

# Calculate sustainable income (4% rule)

sustainable\_annual\_income = remaining\_pension \* 0.04

# Add state pension

IF state.tax\_status.uk\_resident:

state\_pension\_annual = estimate\_uk\_state\_pension(state)

ELSE IF state.tax\_status.sa\_resident:

state\_pension\_annual = 0 # SA has no state pension

ELSE:

state\_pension\_annual = 0

total\_retirement\_income = sustainable\_annual\_income + state\_pension\_annual

# Calculate replacement ratio

current\_income = state.income.employment + state.income.self\_employment

replacement\_ratio = (total\_retirement\_income / current\_income) \* 100

RETURN {

retirement\_age: retirement\_age,

pension\_pot\_at\_retirement: pension\_pot,

tax\_free\_lump\_sum: tax\_free\_lump\_sum,

remaining\_for\_income: remaining\_pension,

sustainable\_annual\_income: sustainable\_annual\_income,

state\_pension\_annual: state\_pension\_annual,

total\_annual\_income: total\_retirement\_income,

monthly\_income: total\_retirement\_income / 12,

replacement\_ratio: replacement\_ratio,

adequate: replacement\_ratio >= 70 # 70% is typical target

}

# ===== SCENARIO COMPARISON =====

FUNCTION compare\_scenarios(

scenario\_ids: array[uuid]

) -> ScenarioComparison:

IF scenario\_ids.length < 2:

THROW ValidationError("Need at least 2 scenarios to compare")

scenarios = []

FOR EACH id IN scenario\_ids:

scenario = get\_scenario(id)

results = get\_scenario\_results(id)

scenarios.append({scenario: scenario, results: results})

# Compare key metrics

comparison = {

scenarios: scenarios,

metric\_comparisons: {},

winner\_by\_metric: {},

trade\_offs: [],

recommendations: []

}

# Compare net worth

comparison.metric\_comparisons.net\_worth = compare\_metric(

scenarios,

'projections.net\_worth[final]',

higher\_is\_better: TRUE

)

# Compare retirement income

comparison.metric\_comparisons.retirement\_income = compare\_metric(

scenarios,

'projections.retirement.total\_annual\_income',

higher\_is\_better: TRUE

)

# Compare lifetime tax paid

comparison.metric\_comparisons.lifetime\_tax = compare\_metric(

scenarios,

'metrics.lifetime\_tax\_paid',

higher\_is\_better: FALSE

)

# Compare IHT liability

comparison.metric\_comparisons.iht\_liability = compare\_metric(

scenarios,

'metrics.iht\_liability\_at\_death',

higher\_is\_better: FALSE

)

# Compare goals achieved

comparison.metric\_comparisons.goals\_achieved = compare\_metric(

scenarios,

'metrics.goals\_achieved\_percentage',

higher\_is\_better: TRUE

)

# Compare retirement age

comparison.metric\_comparisons.retirement\_age = compare\_metric(

scenarios,

'scenario\_state.retirement.retirement\_age',

higher\_is\_better: FALSE # Earlier retirement is better

)

# Identify trade-offs

comparison.trade\_offs = identify\_trade\_offs(scenarios)

# Generate recommendations

comparison.recommendations = generate\_scenario\_recommendations(scenarios, comparison)

# Overall best scenario

comparison.overall\_best = determine\_best\_scenario(scenarios, comparison)

RETURN comparison

FUNCTION compare\_metric(

scenarios: array,

metric\_path: string,

higher\_is\_better: boolean

) -> MetricComparison:

values = []

FOR EACH scenario IN scenarios:

value = extract\_value\_by\_path(scenario.results, metric\_path)

values.append({

scenario\_id: scenario.scenario.id,

scenario\_name: scenario.scenario.name,

value: value

})

# Sort

IF higher\_is\_better:

sorted\_values = SORT(values, BY value DESC)

ELSE:

sorted\_values = SORT(values, BY value ASC)

# Calculate differences

best\_value = sorted\_values[0].value

worst\_value = sorted\_values[last].value

FOR EACH item IN sorted\_values:

item.difference\_from\_best = item.value - best\_value

item.percentage\_difference = ((item.value - best\_value) / best\_value) \* 100

RETURN {

metric\_name: metric\_path,

higher\_is\_better: higher\_is\_better,

values: sorted\_values,

best\_scenario: sorted\_values[0].scenario\_id,

worst\_scenario: sorted\_values[last].scenario\_id,

range: worst\_value - best\_value,

average: AVERAGE(values.value)

}

FUNCTION identify\_trade\_offs(scenarios: array) -> array[TradeOff]:

trade\_offs = []

# Example: Early retirement vs higher net worth

early\_retirement\_scenario = find\_scenario\_with\_earliest\_retirement(scenarios)

highest\_net\_worth\_scenario = find\_scenario\_with\_highest\_net\_worth(scenarios)

IF early\_retirement\_scenario.id != highest\_net\_worth\_scenario.id:

retirement\_age\_diff = highest\_net\_worth\_scenario.retirement\_age -

early\_retirement\_scenario.retirement\_age

net\_worth\_diff = highest\_net\_worth\_scenario.final\_net\_worth -

early\_retirement\_scenario.final\_net\_worth

trade\_offs.append({

trade\_off\_type: 'RETIREMENT\_AGE\_VS\_NET\_WORTH',

description: "Retiring {retirement\_age\_diff} years later results in £{net\_worth\_diff} higher net worth",

scenario\_a: early\_retirement\_scenario,

scenario\_b: highest\_net\_worth\_scenario,

decision\_factors: [

"Personal preference for leisure vs wealth",

"Health considerations",

"Legacy intentions"

]

})

# Example: Tax residency change vs complexity

uk\_resident\_scenario = find\_uk\_resident\_scenario(scenarios)

sa\_resident\_scenario = find\_sa\_resident\_scenario(scenarios)

IF uk\_resident\_scenario AND sa\_resident\_scenario:

tax\_diff = uk\_resident\_scenario.lifetime\_tax - sa\_resident\_scenario.lifetime\_tax

trade\_offs.append({

trade\_off\_type: 'TAX\_RESIDENCY\_CHANGE',

description: "Moving to SA saves £{tax\_diff} in lifetime tax but adds complexity",

scenario\_a: uk\_resident\_scenario,

scenario\_b: sa\_resident\_scenario,

decision\_factors: [

"Tax savings: £{tax\_diff}",

"Complexity of dual compliance",

"Family and lifestyle preferences",

"Healthcare considerations",

"Currency risk"

]

})

# Example: Property purchase vs investment

property\_scenario = find\_property\_purchase\_scenario(scenarios)

investment\_scenario = find\_increased\_investment\_scenario(scenarios)

IF property\_scenario AND investment\_scenario:

trade\_offs.append({

trade\_off\_type: 'PROPERTY\_VS\_INVESTMENT',

description: "Property provides stability but lower returns vs flexible investments",

scenario\_a: property\_scenario,

scenario\_b: investment\_scenario,

decision\_factors: [

"Property: Tangible asset, housing security",

"Investments: Higher potential returns, liquidity",

"Property: Maintenance costs, less flexible",

"Investments: Market volatility, no housing benefit"

]

})

RETURN trade\_offs

# ===== WHAT-IF MODELING =====

FUNCTION run\_what\_if\_analysis(

user\_id: uuid,

base\_scenario\_id: uuid,

what\_if\_variables: array[WhatIfVariable]

) -> WhatIfResults:

base\_scenario = get\_scenario(base\_scenario\_id)

base\_results = get\_scenario\_results(base\_scenario\_id)

what\_if\_results = []

FOR EACH variable IN what\_if\_variables:

# Create temporary scenario with variable changed

modified\_scenario = deep\_copy(base\_scenario)

# Apply variable change

MATCH variable.type:

CASE 'INCOME\_CHANGE':

modified\_scenario.state.income.total \*= (1 + variable.percentage\_change / 100)

CASE 'INVESTMENT\_RETURN':

modified\_scenario.assumptions.investment\_return = variable.new\_value

CASE 'INFLATION\_RATE':

modified\_scenario.assumptions.inflation\_rate = variable.new\_value

CASE 'RETIREMENT\_AGE':

modified\_scenario.state.retirement.planned\_retirement\_age = variable.new\_value

CASE 'PROPERTY\_VALUE\_CHANGE':

property = find\_property(modified\_scenario.state.assets)

property.value \*= (1 + variable.percentage\_change / 100)

CASE 'MARKET\_CRASH':

modified\_scenario.state.investments.portfolio\_value \*= (1 - variable.crash\_percentage / 100)

CASE 'INTEREST\_RATE\_CHANGE':

FOR EACH liability IN modified\_scenario.state.liabilities:

IF liability.type = 'MORTGAGE':

liability.interest\_rate += variable.rate\_change

# Recalculate scenario

what\_if\_result = calculate\_scenario\_outcomes(modified\_scenario.state, modified\_scenario.assumptions)

# Compare to base

impact = {

variable: variable,

base\_value: extract\_base\_value(base\_results, variable),

new\_value: extract\_base\_value(what\_if\_result, variable),

difference: calculate\_difference(base\_results, what\_if\_result, variable),

percentage\_impact: calculate\_percentage\_impact(base\_results, what\_if\_result, variable)

}

what\_if\_results.append({

variable: variable,

results: what\_if\_result,

impact: impact

})

RETURN {

base\_scenario: base\_scenario,

base\_results: base\_results,

what\_if\_results: what\_if\_results,

sensitivity\_analysis: generate\_sensitivity\_analysis(base\_results, what\_if\_results)

}

FUNCTION generate\_sensitivity\_analysis(

base\_results: ScenarioResults,

what\_if\_results: array[WhatIfResult]

) -> SensitivityAnalysis:

# Identify which variables have biggest impact on outcomes

sensitivity\_rankings = []

FOR EACH what\_if IN what\_if\_results:

# Measure impact on key metrics

net\_worth\_impact = ABS(what\_if.impact.difference.net\_worth)

retirement\_income\_impact = ABS(what\_if.impact.difference.retirement\_income)

tax\_impact = ABS(what\_if.impact.difference.lifetime\_tax)

# Combined sensitivity score

sensitivity\_score = (

net\_worth\_impact \* 0.4 +

retirement\_income\_impact \* 0.4 +

tax\_impact \* 0.2

) / base\_results.metrics.net\_worth # Normalize

sensitivity\_rankings.append({

variable: what\_if.variable,

sensitivity\_score: sensitivity\_score,

impacts: {

net\_worth: net\_worth\_impact,

retirement\_income: retirement\_income\_impact,

lifetime\_tax: tax\_impact

}

})

# Sort by sensitivity

ranked = SORT(sensitivity\_rankings, BY sensitivity\_score DESC)

# Classify

FOR EACH item IN ranked:

IF item.sensitivity\_score > 0.10: # >10% impact

item.sensitivity\_level = 'HIGH'

ELSE IF item.sensitivity\_score > 0.05:

item.sensitivity\_level = 'MEDIUM'

ELSE:

item.sensitivity\_level = 'LOW'

RETURN {

ranked\_sensitivities: ranked,

most\_sensitive\_variable: ranked[0].variable,

least\_sensitive\_variable: ranked[last].variable,

high\_sensitivity\_variables: FILTER(ranked, WHERE sensitivity\_level = 'HIGH'),

recommendations: generate\_sensitivity\_recommendations(ranked)

}

FUNCTION generate\_sensitivity\_recommendations(

sensitivity\_rankings: array

) -> array[Recommendation]:

recommendations = []

high\_sensitivity\_vars = FILTER(sensitivity\_rankings, WHERE sensitivity\_level = 'HIGH')

FOR EACH var IN high\_sensitivity\_vars:

MATCH var.variable.type:

CASE 'INVESTMENT\_RETURN':

recommendations.append({

title: "Your outcome is highly sensitive to investment returns",

description: "A 1% change in returns significantly impacts your retirement. Consider diversification and risk management.",

priority: 'HIGH',

actions: [

"Review investment strategy with advisor",

"Consider diversifying across asset classes",

"Don't rely on optimistic return assumptions"

]

})

CASE 'RETIREMENT\_AGE':

recommendations.append({

title: "Retirement age has major impact on outcomes",

description: "Working even 1-2 years longer significantly improves your position.",

priority: 'HIGH',

actions: [

"Keep retirement age flexible",

"Consider phased retirement",

"Build multiple income streams"

]

})

CASE 'INFLATION\_RATE':

recommendations.append({

title: "Outcomes sensitive to inflation",

description: "Higher inflation erodes purchasing power and retirement adequacy.",

priority: 'MEDIUM',

actions: [

"Consider inflation-protected investments",

"Build buffer into retirement plans",

"Review spending assumptions regularly"

]

})

RETURN recommendations

# ===== MONTE CARLO SIMULATION =====

FUNCTION run\_monte\_carlo\_simulation(

scenario\_state: ScenarioState,

assumptions: Assumptions,

simulations: integer

) -> ProbabilityDistribution:

# Run multiple simulations with randomized returns

simulation\_results = []

FOR iteration FROM 1 TO simulations:

# Randomize key variables

random\_assumptions = {

investment\_return: generate\_random\_return(

mean: assumptions.investment\_return,

volatility: 0.15 # 15% volatility

),

inflation\_rate: generate\_random\_inflation(

mean: assumptions.inflation\_rate,

volatility: 0.02

),

longevity: generate\_random\_age\_at\_death(

scenario\_state.age,

scenario\_state.gender,

scenario\_state.health\_status

)

}

# Run simulation

sim\_result = calculate\_scenario\_outcomes(

scenario\_state,

{

...assumptions,

...random\_assumptions

}

)

simulation\_results.append({

iteration: iteration,

final\_net\_worth: sim\_result.projections.net\_worth[final],

retirement\_income: sim\_result.projections.retirement.total\_annual\_income,

estate\_value: sim\_result.metrics.estate\_value\_at\_death,

goals\_achieved: sim\_result.metrics.goals\_achieved\_count

})

# Analyze distribution

net\_worth\_distribution = analyze\_distribution(simulation\_results, 'final\_net\_worth')

retirement\_distribution = analyze\_distribution(simulation\_results, 'retirement\_income')

# Calculate confidence intervals

confidence\_95 = {

net\_worth: calculate\_percentile\_range(net\_worth\_distribution, 2.5, 97.5),

retirement\_income: calculate\_percentile\_range(retirement\_distribution, 2.5, 97.5)

}

# Probability of success

success\_criteria = {

retirement\_income\_adequate: assumptions.target\_retirement\_income,

net\_worth\_positive: 0,

goals\_all\_achieved: scenario\_state.goals.length

}

success\_count = COUNT(simulation\_results WHERE

retirement\_income >= success\_criteria.retirement\_income\_adequate AND

final\_net\_worth > success\_criteria.net\_worth\_positive AND

goals\_achieved = success\_criteria.goals\_all\_achieved

)

probability\_of\_success = (success\_count / simulations) \* 100

RETURN {

simulations\_run: simulations,

distributions: {

net\_worth: net\_worth\_distribution,

retirement\_income: retirement\_distribution

},

confidence\_intervals: confidence\_95,

probability\_of\_success: probability\_of\_success,

percentiles: {

p10: calculate\_percentile(net\_worth\_distribution, 10),

p25: calculate\_percentile(net\_worth\_distribution, 25),

p50: calculate\_percentile(net\_worth\_distribution, 50), # Median

p75: calculate\_percentile(net\_worth\_distribution, 75),

p90: calculate\_percentile(net\_worth\_distribution, 90)

},

worst\_case: MIN(simulation\_results.final\_net\_worth),

best\_case: MAX(simulation\_results.final\_net\_worth),

expected\_value: AVERAGE(simulation\_results.final\_net\_worth)

}

FUNCTION analyze\_distribution(results: array, field: string) -> Distribution:

values = EXTRACT(results, field)

RETURN {

mean: AVERAGE(values),

median: MEDIAN(values),

std\_dev: STDEV(values),

min: MIN(values),

max: MAX(values),

skewness: calculate\_skewness(values),

kurtosis: calculate\_kurtosis(values),

histogram: generate\_histogram(values, bins: 20)

}

**API Endpoints:**

# Scenario Management

POST /api/v1/scenarios

PUT /api/v1/scenarios/{id}

DELETE /api/v1/scenarios/{id}

GET /api/v1/scenarios/{userId}

GET /api/v1/scenarios/{id}

# Scenario Calculation

POST /api/v1/scenarios/{id}/calculate

POST /api/v1/scenarios/{id}/recalculate

GET /api/v1/scenarios/{id}/results

# Scenario Comparison

POST /api/v1/scenarios/compare

GET /api/v1/scenarios/comparison/{comparisonId}

# What-If Analysis

POST /api/v1/scenarios/{id}/what-if

POST /api/v1/scenarios/{id}/sensitivity-analysis

POST /api/v1/scenarios/{id}/monte-carlo

# Baseline

POST /api/v1/scenarios/baseline/create

GET /api/v1/scenarios/baseline/{userId}/current

**Data Models:**

TABLE: baseline\_snapshots

- id: UUID (PK)

- user\_id: UUID (FK to users)

- snapshot\_date: TIMESTAMP

- age: INTEGER

- financial\_state: JSON (complete state)

- tax\_status: JSON

- created\_at: TIMESTAMP

TABLE: scenarios

- id: UUID (PK)

- user\_id: UUID (FK to users)

- name: VARCHAR(255)

- description: TEXT

- scenario\_type: ENUM(...)

- baseline\_snapshot\_id: UUID (FK to baseline\_snapshots)

- assumptions: JSON

- modifications: JSON

- status: ENUM('DRAFT', 'CALCULATED', 'ARCHIVED')

- created\_at: TIMESTAMP

- last\_accessed: TIMESTAMP

- expires\_at: TIMESTAMP

TABLE: scenario\_results

- scenario\_id: UUID (PK, FK to scenarios)

- projections: JSON

- metrics: JSON

- risks: JSON

- probability\_distribution: JSON

- calculated\_at: TIMESTAMP

TABLE: scenario\_comparisons

- id: UUID (PK)

- user\_id: UUID (FK to users)

- name: VARCHAR(255)

- scenario\_ids: JSON (array of UUIDs)

- comparison\_data: JSON

- created\_at: TIMESTAMP

TABLE: what\_if\_analyses

- id: UUID (PK)

- base\_scenario\_id: UUID (FK to scenarios)

- variables\_tested: JSON

- results: JSON

- sensitivity\_analysis: JSON

- created\_at: TIMESTAMP

INDEX on scenarios(user\_id, status, last\_accessed)

INDEX on scenario\_results(scenario\_id)

This completes Scenario Analysis & What-If Modeling. Now proceeding to the Personalization Engine...

**Feature 10.4: Personalization Engine**

**Feature Name:** Adaptive Learning and Personalization System

**User Story:** As a user, I want the system to learn from my behavior, preferences, and feedback so that recommendations and advice become increasingly relevant and tailored to my specific situation over time.

**Acceptance Criteria:**

* Learn from user interactions and behavior
* Adapt recommendation style to user preferences
* Track which recommendations users act on
* Adjust priority and content based on success rates
* Personalize communication frequency and channels
* Adapt to user's financial literacy level
* Remember user preferences across sessions
* A/B test recommendation approaches
* Provide transparency into personalization

**Technical Requirements:**

* Machine learning models (collaborative filtering, content-based)
* Behavioral tracking system
* Feedback loop mechanism
* A/B testing framework
* Preference learning algorithms
* User segmentation
* Model retraining pipeline
* Explainable AI for transparency

**Constraints:**

* Must respect user privacy (GDPR/POPIA compliant)
* Cannot make fully automated regulated financial decisions
* Must allow user to override personalization
* Model updates: Weekly retraining
* Minimum data: 30 days of interactions before full personalization
* Performance: Recommendations generated in <3 seconds

**Implementation Approach:**

SERVICE: PersonalizationEngine

# ===== USER PROFILE =====

FUNCTION build\_user\_profile(user\_id: uuid) -> UserProfile:

# Demographic factors

demographics = {

age: get\_user\_age(user\_id),

life\_stage: determine\_life\_stage(user\_id),

country: get\_primary\_country(user\_id),

income\_level: categorize\_income\_level(user\_id),

net\_worth\_level: categorize\_net\_worth(user\_id)

}

# Behavioral factors

behavior = {

engagement\_level: calculate\_engagement\_level(user\_id),

login\_frequency: calculate\_login\_frequency(user\_id),

feature\_usage: track\_feature\_usage(user\_id),

recommendation\_interaction\_rate: calculate\_interaction\_rate(user\_id),

goal\_completion\_rate: calculate\_goal\_completion\_rate(user\_id),

average\_session\_duration: calculate\_avg\_session\_duration(user\_id)

}

# Preference factors

preferences = {

risk\_tolerance: get\_risk\_tolerance(user\_id),

investment\_style: infer\_investment\_style(user\_id),

communication\_preference: get\_communication\_preference(user\_id),

detail\_level\_preference: infer\_detail\_preference(user\_id),

recommendation\_categories\_preferred: get\_preferred\_categories(user\_id),

notification\_frequency: get\_notification\_frequency(user\_id)

}

# Financial sophistication

sophistication = {

financial\_literacy\_score: assess\_financial\_literacy(user\_id),

complexity\_comfortable\_with: infer\_complexity\_level(user\_id),

terminology\_familiarity: assess\_terminology\_knowledge(user\_id),

self\_reported\_expertise: get\_self\_reported\_expertise(user\_id)

}

# Historical performance

history = {

recommendations\_accepted: get\_accepted\_recommendations(user\_id),

recommendations\_dismissed: get\_dismissed\_recommendations(user\_id),

average\_time\_to\_action: calculate\_avg\_time\_to\_action(user\_id),

most\_successful\_recommendation\_types: identify\_successful\_types(user\_id),

abandoned\_features: identify\_abandoned\_features(user\_id)

}

RETURN {

user\_id: user\_id,

demographics: demographics,

behavior: behavior,

preferences: preferences,

sophistication: sophistication,

history: history,

profile\_completeness: calculate\_profile\_completeness(demographics, behavior, preferences),

last\_updated: NOW()

}

# ===== BEHAVIORAL TRACKING =====

FUNCTION track\_user\_interaction(

user\_id: uuid,

interaction: UserInteraction

) -> void:

# Record interaction

interaction\_record = {

user\_id: user\_id,

interaction\_type: interaction.type,

target\_id: interaction.target\_id,

target\_type: interaction.target\_type,

action: interaction.action,

context: interaction.context,

timestamp: NOW()

}

store\_interaction(interaction\_record)

# Update real-time profile elements

MATCH interaction.type:

CASE 'RECOMMENDATION\_VIEWED':

increment\_metric(user\_id, 'recommendations\_viewed')

CASE 'RECOMMENDATION\_ACCEPTED':

increment\_metric(user\_id, 'recommendations\_accepted')

record\_successful\_recommendation(

user\_id,

interaction.target\_id,

interaction.recommendation\_category

)

CASE 'RECOMMENDATION\_DISMISSED':

increment\_metric(user\_id, 'recommendations\_dismissed')

record\_dismissal\_reason(user\_id, interaction.target\_id, interaction.reason)

CASE 'FEATURE\_USED':

record\_feature\_usage(user\_id, interaction.feature\_name)

CASE 'GOAL\_CREATED':

record\_goal\_type\_preference(user\_id, interaction.goal\_type)

CASE 'CONTENT\_READ':

record\_content\_interest(user\_id, interaction.content\_topic)

CASE 'TIME\_SPENT':

update\_avg\_session\_duration(user\_id, interaction.duration)

# Trigger personalization update if threshold met

IF should\_update\_personalization(user\_id):

async\_update\_user\_personalization(user\_id)

# ===== RECOMMENDATION PERSONALIZATION =====

FUNCTION personalize\_recommendations(

user\_id: uuid,

base\_recommendations: array[Recommendation]

) -> array[PersonalizedRecommendation]:

profile = build\_user\_profile(user\_id)

personalized = []

FOR EACH rec IN base\_recommendations:

# Calculate personalization score

personalization\_score = calculate\_personalization\_score(rec, profile)

# Adjust recommendation based on profile

personalized\_rec = {

...rec,

personalization\_score: personalization\_score,

# Adjust title based on sophistication

title: adapt\_title\_to\_sophistication(rec.title, profile.sophistication),

# Adjust description detail level

description: adapt\_description\_detail(rec.description, profile.preferences.detail\_level\_preference),

# Adjust tone

tone: adapt\_tone(profile.demographics.age, profile.sophistication),

# Add personalized reasoning

personalized\_reasoning: generate\_personalized\_reasoning(rec, profile),

# Adjust estimated benefit format

benefit\_presentation: adapt\_benefit\_presentation(rec.estimated\_benefit, profile),

# Add relevant examples

examples: generate\_relevant\_examples(rec, profile),

# Adjust priority based on user history

adjusted\_priority: adjust\_priority\_for\_user(rec.priority, rec.category, profile)

}

personalized.append(personalized\_rec)

# Re-rank based on personalization

ranked = rank\_by\_personalization(personalized, profile)

# Filter out recommendations user consistently dismisses

filtered = filter\_consistently\_dismissed(ranked, profile)

RETURN filtered

FUNCTION calculate\_personalization\_score(

recommendation: Recommendation,

profile: UserProfile

) -> decimal:

score = 0.0

# Category preference weight (40%)

category\_preference = profile.history.most\_successful\_recommendation\_types[recommendation.category] OR 0.5

score += category\_preference \* 0.40

# Sophistication match (20%)

sophistication\_match = assess\_sophistication\_match(recommendation, profile.sophistication)

score += sophistication\_match \* 0.20

# Life stage relevance (20%)

life\_stage\_relevance = assess\_life\_stage\_relevance(recommendation, profile.demographics)

score += life\_stage\_relevance \* 0.20

# Timing appropriateness (10%)

timing\_score = assess\_timing(recommendation, profile.behavior)

score += timing\_score \* 0.10

# Recent interaction patterns (10%)

recency\_score = assess\_recency\_relevance(recommendation, profile.history)

score += recency\_score \* 0.10

RETURN score

FUNCTION adapt\_description\_detail(

description: string,

detail\_preference: enum['CONCISE', 'MODERATE', 'DETAILED']

) -> string:

MATCH detail\_preference:

CASE 'CONCISE':

# Extract key sentence only

RETURN extract\_key\_sentence(description) + " [Show more]"

CASE 'MODERATE':

# Keep 2-3 sentences

RETURN extract\_summary(description, sentences: 3)

CASE 'DETAILED':

# Full description plus additional context

RETURN description + "\n\n" + generate\_additional\_context(description)

FUNCTION generate\_personalized\_reasoning(

recommendation: Recommendation,

profile: UserProfile

) -> array[string]:

reasoning = []

# Add life-stage specific reasoning

MATCH profile.demographics.life\_stage:

CASE 'EARLY\_CAREER':

IF recommendation.category = 'RETIREMENT':

reasoning.append("Starting early gives your investments decades to grow through compounding")

CASE 'MID\_CAREER':

IF recommendation.category = 'PROTECTION':

reasoning.append("At your stage, protecting your family's financial security is crucial")

CASE 'PRE\_RETIREMENT':

IF recommendation.category = 'TAX':

reasoning.append("Tax planning now can significantly impact your retirement income")

# Add country-specific reasoning

IF profile.demographics.country = 'UK':

reasoning.append("This takes advantage of UK tax allowances and reliefs")

ELSE IF profile.demographics.country = 'SA':

reasoning.append("This optimizes for South African tax efficiency")

# Add historical success reasoning

IF profile.history.similar\_recommendations\_successful:

reasoning.append("Similar recommendations have worked well for you in the past")

RETURN reasoning

# ===== COLLABORATIVE FILTERING =====

FUNCTION get\_collaborative\_recommendations(user\_id: uuid) -> array[Recommendation]:

# Find similar users

similar\_users = find\_similar\_users(user\_id, count: 20)

# Get recommendations that similar users accepted

collaborative\_recommendations = []

FOR EACH similar\_user IN similar\_users:

accepted\_recs = get\_accepted\_recommendations(similar\_user.user\_id)

FOR EACH rec IN accepted\_recs:

# Check if this user hasn't seen this recommendation type yet

IF NOT user\_has\_seen\_recommendation\_type(user\_id, rec.type):

# Calculate relevance score

relevance = calculate\_collaborative\_relevance(

recommendation: rec,

target\_user\_id: user\_id,

source\_user\_similarity: similar\_user.similarity\_score

)

collaborative\_recommendations.append({

recommendation: rec,

relevance\_score: relevance,

source: 'COLLABORATIVE\_FILTERING',

similar\_user\_count: COUNT(similar\_users WHERE accepted this rec)

})

# Rank and return top recommendations

ranked = SORT(collaborative\_recommendations, BY relevance\_score DESC)

RETURN ranked[0:5] # Top 5

FUNCTION find\_similar\_users(target\_user\_id: uuid, count: integer) -> array[SimilarUser]:

target\_profile = build\_user\_profile(target\_user\_id)

all\_users = get\_all\_active\_users()

similarities = []

FOR EACH user IN all\_users:

IF user.id = target\_user\_id:

CONTINUE # Skip self

user\_profile = build\_user\_profile(user.id)

# Calculate similarity score

similarity = calculate\_profile\_similarity(target\_profile, user\_profile)

IF similarity > 0.5: # Threshold

similarities.append({

user\_id: user.id,

similarity\_score: similarity,

common\_attributes: identify\_common\_attributes(target\_profile, user\_profile)

})

# Sort by similarity

sorted\_similar = SORT(similarities, BY similarity\_score DESC)

RETURN sorted\_similar[0:count]

FUNCTION calculate\_profile\_similarity(

profile\_a: UserProfile,

profile\_b: UserProfile

) -> decimal:

# Weighted similarity across dimensions

# Demographic similarity (30%)

demo\_similarity = (

age\_similarity(profile\_a.demographics.age, profile\_b.demographics.age) \* 0.4 +

(profile\_a.demographics.life\_stage = profile\_b.demographics.life\_stage ? 1.0 : 0.0) \* 0.3 +

income\_similarity(profile\_a.demographics.income\_level, profile\_b.demographics.income\_level) \* 0.3

)

# Behavioral similarity (30%)

behavior\_similarity = (

engagement\_similarity(profile\_a.behavior.engagement\_level, profile\_b.behavior.engagement\_level) \* 0.5 +

feature\_usage\_overlap(profile\_a.behavior.feature\_usage, profile\_b.behavior.feature\_usage) \* 0.5

)

# Preference similarity (20%)

pref\_similarity = (

(profile\_a.preferences.risk\_tolerance = profile\_b.preferences.risk\_tolerance ? 1.0 : 0.5) \* 0.5 +

category\_preference\_overlap(profile\_a.preferences, profile\_b.preferences) \* 0.5

)

# Financial sophistication similarity (20%)

soph\_similarity = ABS(profile\_a.sophistication.financial\_literacy\_score -

profile\_b.sophistication.financial\_literacy\_score) / 10.0

soph\_similarity = 1.0 - soph\_similarity # Invert so closer = higher score

# Weighted total

total\_similarity = (

demo\_similarity \* 0.30 +

behavior\_similarity \* 0.30 +

pref\_similarity \* 0.20 +

soph\_similarity \* 0.20

)

RETURN total\_similarity

# ===== A/B TESTING FRAMEWORK =====

FUNCTION assign\_ab\_test\_variant(

user\_id: uuid,

test\_name: string

) -> string:

# Check if user already assigned to this test

existing\_assignment = get\_ab\_test\_assignment(user\_id, test\_name)

IF existing\_assignment:

RETURN existing\_assignment.variant

# Get test configuration

test\_config = get\_ab\_test\_config(test\_name)

IF NOT test\_config.active:

RETURN 'CONTROL' # Default if test not active

# Assign variant based on consistent hash

hash = consistent\_hash(user\_id + test\_name)

variant\_index = hash MOD 100

cumulative = 0

FOR EACH variant IN test\_config.variants:

cumulative += variant.traffic\_percentage

IF variant\_index < cumulative:

selected\_variant = variant.name

BREAK

# Record assignment

record\_ab\_test\_assignment(user\_id, test\_name, selected\_variant)

RETURN selected\_variant

FUNCTION apply\_ab\_test\_personalization(

recommendation: Recommendation,

user\_id: uuid

) -> Recommendation:

# Example tests

# Test 1: Benefit emphasis

benefit\_test\_variant = assign\_ab\_test\_variant(user\_id, 'BENEFIT\_EMPHASIS')

MATCH benefit\_test\_variant:

CASE 'CONTROL':

# Standard benefit presentation

PASS

CASE 'MONETARY\_FOCUS':

# Emphasize monetary benefits

IF recommendation.estimated\_benefit.amount:

recommendation.title = "Save £{amount}: " + recommendation.title

CASE 'PERCENTAGE\_FOCUS':

# Emphasize percentage improvements

IF recommendation.estimated\_benefit.percentage:

recommendation.title = "Improve by {percentage}%: " + recommendation.title

# Test 2: Urgency framing

urgency\_test\_variant = assign\_ab\_test\_variant(user\_id, 'URGENCY\_FRAMING')

MATCH urgency\_test\_variant:

CASE 'CONTROL':

PASS

CASE 'HIGH\_URGENCY':

IF recommendation.deadline:

days\_remaining = calculate\_days\_until(recommendation.deadline)

recommendation.description = "⚠️ Only {days} days left! " + recommendation.description

CASE 'LOW\_PRESSURE':

# Remove urgency language

recommendation.description = remove\_urgency\_words(recommendation.description)

# Test 3: Social proof

social\_proof\_variant = assign\_ab\_test\_variant(user\_id, 'SOCIAL\_PROOF')

MATCH social\_proof\_variant:

CASE 'CONTROL':

PASS

CASE 'SOCIAL\_PROOF':

# Add social proof element

similar\_users\_count = count\_similar\_users\_who\_accepted(recommendation)

IF similar\_users\_count > 10:

recommendation.description += "\n\n✓ {count} users in similar situations have acted on this."

RETURN recommendation

FUNCTION record\_ab\_test\_outcome(

user\_id: uuid,

test\_name: string,

recommendation\_id: uuid,

outcome: enum['VIEWED', 'ACCEPTED', 'DISMISSED', 'IGNORED']

) -> void:

variant = get\_ab\_test\_assignment(user\_id, test\_name).variant

outcome\_record = {

test\_name: test\_name,

variant: variant,

user\_id: user\_id,

recommendation\_id: recommendation\_id,

outcome: outcome,

timestamp: NOW()

}

store\_ab\_test\_outcome(outcome\_record)

FUNCTION analyze\_ab\_test\_results(test\_name: string) -> ABTestAnalysis:

test\_config = get\_ab\_test\_config(test\_name)

variant\_performance = []

FOR EACH variant IN test\_config.variants:

outcomes = get\_ab\_test\_outcomes(test\_name, variant.name)

total = outcomes.count

accepted = COUNT(outcomes WHERE outcome = 'ACCEPTED')

dismissed = COUNT(outcomes WHERE outcome = 'DISMISSED')

viewed = COUNT(outcomes WHERE outcome = 'VIEWED')

acceptance\_rate = (accepted / total) \* 100

dismissal\_rate = (dismissed / total) \* 100

engagement\_rate = ((viewed + accepted) / total) \* 100

variant\_performance.append({

variant\_name: variant.name,

sample\_size: total,

acceptance\_rate: acceptance\_rate,

dismissal\_rate: dismissal\_rate,

engagement\_rate: engagement\_rate,

statistical\_significance: calculate\_statistical\_significance(

variant,

test\_config.variants[0], # Compare to control

outcomes

)

})

# Determine winner

winner = MAX(variant\_performance, BY acceptance\_rate)

# Calculate lift over control

control\_performance = FIRST(variant\_performance WHERE variant\_name = 'CONTROL')

lift = ((winner.acceptance\_rate - control\_performance.acceptance\_rate) /

control\_performance.acceptance\_rate) \* 100

RETURN {

test\_name: test\_name,

variant\_performance: variant\_performance,

winner: winner.variant\_name,

lift\_over\_control: lift,

recommendation: IF lift > 10 AND winner.statistical\_significance > 0.95 THEN

"Roll out winning variant to all users"

ELSE

"Continue testing or refine hypothesis"

}

# ===== FEEDBACK LOOP =====

FUNCTION process\_user\_feedback(

user\_id: uuid,

feedback: Feedback

) -> void:

# Store feedback

store\_feedback(feedback)

# Update personalization based on feedback

MATCH feedback.type:

CASE 'RECOMMENDATION\_RATING':

# User rated a recommendation (1-5 stars)

update\_recommendation\_type\_preference(

user\_id,

feedback.recommendation\_category,

feedback.rating

)

CASE 'FEATURE\_RATING':

# User rated a feature

update\_feature\_preference(

user\_id,

feedback.feature\_name,

feedback.rating

)

CASE 'CONTENT\_PREFERENCE':

# User indicated preference for content style

update\_content\_style\_preference(

user\_id,

feedback.preferred\_style

)

CASE 'NOTIFICATION\_PREFERENCE':

# User adjusted notification settings

update\_notification\_preferences(

user\_id,

feedback.notification\_settings

)

CASE 'GENERAL\_FEEDBACK':

# Qualitative feedback - analyze sentiment

sentiment = analyze\_sentiment(feedback.text)

update\_satisfaction\_score(user\_id, sentiment)

# Trigger model retraining if significant feedback accumulated

IF should\_retrain\_model(user\_id):

schedule\_model\_retraining(user\_id)

# ===== MODEL RETRAINING =====

FUNCTION retrain\_personalization\_models() -> void:

# Run weekly (scheduled job)

# Get all users with sufficient interaction history

eligible\_users = get\_users\_with\_min\_interactions(min\_interactions: 50)

FOR EACH user IN eligible\_users:

# Extract features

features = extract\_user\_features(user.id)

# Extract labels (successful recommendations)

labels = extract\_recommendation\_outcomes(user.id)

# Train user-specific model

model = train\_recommendation\_model(features, labels)

# Evaluate model

performance = evaluate\_model(model, validation\_data)

# Store model if performance acceptable

IF performance.accuracy > 0.70:

store\_user\_model(user.id, model, performance)

# Train global collaborative filtering model

global\_cf\_model = train\_collaborative\_filtering\_model(all\_user\_interactions)

store\_global\_model('COLLABORATIVE\_FILTERING', global\_cf\_model)

# Log retraining metrics

log\_model\_retraining\_metrics({

users\_retrained: eligible\_users.count,

average\_accuracy: AVERAGE(all\_models.accuracy),

timestamp: NOW()

})

# ===== EXPLAINABILITY =====

FUNCTION explain\_personalization(

user\_id: uuid,

recommendation\_id: uuid

) -> PersonalizationExplanation:

recommendation = get\_recommendation(recommendation\_id)

profile = build\_user\_profile(user\_id)

# Explain why this recommendation was shown

explanation = {

primary\_reasons: [],

contributing\_factors: [],

how\_to\_improve\_relevance: []

}

# Analyze recommendation selection

IF recommendation.category IN profile.history.most\_successful\_recommendation\_types:

explanation.primary\_reasons.append(

"You've successfully acted on similar {category} recommendations before"

)

IF recommendation matches profile.demographics.life\_stage:

explanation.primary\_reasons.append(

"This is particularly relevant for your current life stage"

)

IF recommendation.estimated\_benefit.amount > profile.typical\_benefit\_threshold:

explanation.primary\_reasons.append(

"The potential benefit (£{amount}) is significant for your situation"

)

# Contributing factors

IF recommendation selected via collaborative filtering:

similar\_count = count\_similar\_users\_who\_accepted(recommendation\_id)

explanation.contributing\_factors.append(

"{count} users with similar profiles have benefited from this"

)

IF recommendation.urgency\_score > 80:

explanation.contributing\_factors.append(

"This is time-sensitive and requires prompt action"

)

# How to improve relevance

explanation.how\_to\_improve\_relevance = [

"Rate recommendations to help us understand your preferences",

"Complete your financial goals for more targeted advice",

"Provide feedback on what types of recommendations you find most valuable"

]

RETURN explanation

**API Endpoints:**

# Profile Management

GET /api/v1/personalization/profile/{userId}

PUT /api/v1/personalization/profile/{userId}/preferences

POST /api/v1/personalization/profile/{userId}/refresh

# Interaction Tracking

POST /api/v1/personalization/track-interaction

POST /api/v1/personalization/track-event

# Feedback

POST /api/v1/personalization/feedback

GET /api/v1/personalization/feedback/{userId}/history

# A/B Testing

GET /api/v1/personalization/ab-test/{testName}/variant

POST /api/v1/personalization/ab-test/{testName}/outcome

# Explainability

GET /api/v1/personalization/explain/{recommendationId}

GET /api/v1/personalization/why-seeing-this

# Admin (for monitoring)

GET /api/v1/personalization/model-performance

POST /api/v1/personalization/retrain-models

GET /api/v1/personalization/ab-test/{testName}/results

**Data Models:**

TABLE: user\_personalization\_profiles

- user\_id: UUID (PK, FK to users)

- engagement\_level: ENUM('LOW', 'MEDIUM', 'HIGH')

- financial\_literacy\_score: INTEGER (1-10)

- risk\_tolerance: ENUM('LOW', 'MEDIUM', 'HIGH')

- detail\_preference: ENUM('CONCISE', 'MODERATE', 'DETAILED')

- preferred\_categories: JSON (array)

- dismissed\_categories: JSON (array)

- communication\_style: VARCHAR(50)

- notification\_frequency: ENUM('REAL\_TIME', 'DAILY', 'WEEKLY', 'MONTHLY')

- preferred\_channels: JSON (array: 'EMAIL', 'IN\_APP', 'SMS')

- profile\_completeness: DECIMAL(5,2)

- last\_updated: TIMESTAMP

- created\_at: TIMESTAMP

TABLE: user\_interactions

- id: UUID (PK)

- user\_id: UUID (FK to users)

- interaction\_type: ENUM('RECOMMENDATION\_VIEWED', 'RECOMMENDATION\_ACCEPTED',

'RECOMMENDATION\_DISMISSED', 'FEATURE\_USED', 'GOAL\_CREATED',

'CONTENT\_READ', 'TIME\_SPENT', 'SEARCH\_PERFORMED')

- target\_id: UUID

- target\_type: VARCHAR(50)

- action: VARCHAR(100)

- context: JSON

- session\_id: UUID

- timestamp: TIMESTAMP

- device\_type: VARCHAR(50)

TABLE: recommendation\_feedback

- id: UUID (PK)

- recommendation\_id: UUID (FK to ai\_recommendations)

- user\_id: UUID (FK to users)

- feedback\_type: ENUM('RATING', 'HELPFUL', 'NOT\_HELPFUL', 'ALREADY\_DONE', 'NOT\_RELEVANT')

- rating: INTEGER (1-5, nullable)

- feedback\_text: TEXT

- timestamp: TIMESTAMP

TABLE: recommendation\_outcomes

- id: UUID (PK)

- recommendation\_id: UUID (FK to ai\_recommendations)

- user\_id: UUID (FK to users)

- outcome: ENUM('ACCEPTED', 'PARTIALLY\_ACCEPTED', 'DISMISSED', 'IGNORED', 'EXPIRED')

- outcome\_date: TIMESTAMP

- time\_to\_action\_days: INTEGER

- actual\_benefit\_realized: DECIMAL(15,2) (measured post-action)

- notes: TEXT

TABLE: category\_preferences

- user\_id: UUID (FK to users)

- category: VARCHAR(100)

- preference\_score: DECIMAL(5,2) (0-1 scale)

- acceptance\_count: INTEGER

- dismissal\_count: INTEGER

- total\_shown: INTEGER

- last\_interaction: TIMESTAMP

- PRIMARY KEY (user\_id, category)

TABLE: similar\_user\_mappings

- user\_id: UUID (FK to users)

- similar\_user\_id: UUID (FK to users)

- similarity\_score: DECIMAL(5,4)

- common\_attributes: JSON

- calculated\_at: TIMESTAMP

- PRIMARY KEY (user\_id, similar\_user\_id)

TABLE: ab\_test\_configurations

- test\_name: VARCHAR(100) (PK)

- description: TEXT

- hypothesis: TEXT

- start\_date: DATE

- end\_date: DATE

- active: BOOLEAN

- variants: JSON (array of {name, description, traffic\_percentage})

- success\_metric: VARCHAR(100)

- minimum\_sample\_size: INTEGER

- created\_by: UUID (FK to users - admin)

- created\_at: TIMESTAMP

TABLE: ab\_test\_assignments

- id: UUID (PK)

- user\_id: UUID (FK to users)

- test\_name: VARCHAR(100) (FK to ab\_test\_configurations)

- variant: VARCHAR(50)

- assigned\_at: TIMESTAMP

- UNIQUE (user\_id, test\_name)

TABLE: ab\_test\_outcomes

- id: UUID (PK)

- test\_name: VARCHAR(100) (FK to ab\_test\_configurations)

- variant: VARCHAR(50)

- user\_id: UUID (FK to users)

- recommendation\_id: UUID (FK to ai\_recommendations)

- outcome: ENUM('VIEWED', 'ACCEPTED', 'DISMISSED', 'IGNORED', 'COMPLETED')

- timestamp: TIMESTAMP

- context: JSON

TABLE: personalization\_models

- user\_id: UUID (PK, FK to users)

- model\_type: ENUM('RECOMMENDATION\_RANKING', 'CONTENT\_PREFERENCE', 'TIMING\_OPTIMIZATION')

- model\_data: BYTEA (serialized model)

- model\_version: VARCHAR(20)

- training\_date: TIMESTAMP

- performance\_metrics: JSON

- feature\_importance: JSON

- active: BOOLEAN DEFAULT TRUE

TABLE: user\_feature\_vectors

- user\_id: UUID (PK, FK to users)

- features: JSON (feature vector for ML models)

- calculated\_at: TIMESTAMP

TABLE: personalization\_events\_log

- id: UUID (PK)

- user\_id: UUID (FK to users)

- event\_type: VARCHAR(100)

- event\_data: JSON

- timestamp: TIMESTAMP

- processing\_status: ENUM('PENDING', 'PROCESSED', 'FAILED')

INDEX on user\_interactions(user\_id, timestamp DESC)

INDEX on user\_interactions(interaction\_type, timestamp DESC)

INDEX on recommendation\_outcomes(user\_id, outcome, outcome\_date)

INDEX on ab\_test\_outcomes(test\_name, variant, outcome)

INDEX on category\_preferences(user\_id, preference\_score DESC)

INDEX on personalization\_events\_log(processing\_status, timestamp)

**Error Handling:**

ERROR CASES:

1. Insufficient interaction data for personalization

- Response: 200 OK

- Behavior: Fall back to general recommendations

- Message: "Building your personalized profile. Complete more actions for tailored advice"

2. Model training failure

- Response: 500 Internal Server Error (logged internally)

- Behavior: Use previous model version or fallback

- User impact: None (transparent failover)

3. A/B test assignment conflict

- Response: 200 OK

- Behavior: Use existing assignment

- Log: Warning for investigation

4. Feature extraction failure

- Response: 200 OK

- Behavior: Use partial features or defaults

- Log: Error for debugging

5. User opts out of personalization

- Response: 200 OK

- Behavior: Disable personalization, use standard recommendations

- Store: User preference permanently

EDGE CASES:

- New user: Use demographic-based recommendations until sufficient data

- Inactive user returning: Check for stale profile, update before personalizing

- User behavior changes dramatically: Detect concept drift, retrain model

- Privacy mode: Limit tracking, use aggregated patterns only

- Multiple devices: Merge interaction data across sessions

- Shared account: Detect multiple user patterns, suggest separate profiles

- Extreme outlier user: Fall back to robust general recommendations

- Testing environment: Separate A/B test assignments from production

**Performance Considerations:**

* Profile building: Cache for 1 hour, recalculate on significant events
* Interaction tracking: Async processing via message queue
* Model inference: <100ms for recommendation scoring
* Similarity calculation: Pre-compute weekly for active users
* A/B test assignment: Hash-based, deterministic, <5ms
* Feature extraction: Batch process daily, cache results
* Model retraining: Weekly batch job, off-peak hours
* Expected interactions per user: 10-100 per session
* Collaborative filtering: Use approximate nearest neighbors for scale
* Real-time personalization: Hot path <200ms end-to-end

**11. TAX INFORMATION MODULE**

**Feature 11.1: Comprehensive Tax Reference Library**

**Feature Name:** Interactive Tax Rates, Allowances, and Rules Reference

**User Story:** As a user, I want to access a comprehensive, up-to-date reference of all tax rates, allowances, bands, and rules for both UK and South Africa so that I understand the tax treatment of my financial decisions and can verify calculations made by the system.

**Acceptance Criteria:**

* Display current tax rates and allowances for both UK and SA
* Show historical tax rates (previous 5 years minimum)
* Include detailed explanations of each tax type
* Provide examples and calculators for each tax
* Tax year selector (switch between years)
* Country toggle (UK/SA/Both)
* Search functionality for specific tax topics
* Links to official government resources
* Export tax reference data
* Comparison tool (year-over-year changes)
* Personal view (shows user's applicable rates based on status)
* Educational content about tax planning

**Technical Requirements:**

* Tax data repository (versioned by tax year)
* Content management system for educational content
* Search indexing for tax topics
* Calculation widgets embedded in page
* Historical data storage and retrieval
* Real-time updates when tax year changes
* Responsive layout for mobile
* Integration with Tax Intelligence Engine

**Constraints:**

* Data must be sourced from official government publications
* Updates required at start of each tax year (April for UK, March for SA)
* Historical data retained indefinitely
* Must include disclaimer about using official sources for filing
* Cannot provide tax filing advice (information only)

**Implementation Approach:**

SERVICE: TaxInformationService

# ===== TAX INFORMATION RETRIEVAL =====

FUNCTION get\_tax\_information\_page(

user\_id: uuid,

country: enum['UK', 'SA', 'BOTH'],

tax\_year: string

) -> TaxInformationPage:

# Get user context for personalization

user\_context = get\_user\_context(user\_id)

# Retrieve tax data

IF country = 'BOTH':

uk\_data = get\_uk\_tax\_data(tax\_year)

sa\_data = get\_sa\_tax\_data(tax\_year)

RETURN {

view\_mode: 'COMPARISON',

uk\_tax\_data: uk\_data,

sa\_tax\_data: sa\_data,

comparison: generate\_uk\_sa\_comparison(uk\_data, sa\_data),

personalized\_view: generate\_personalized\_view(user\_context, uk\_data, sa\_data)

}

ELSE IF country = 'UK':

uk\_data = get\_uk\_tax\_data(tax\_year)

RETURN {

view\_mode: 'UK\_ONLY',

tax\_data: uk\_data,

personalized\_view: generate\_personalized\_uk\_view(user\_context, uk\_data)

}

ELSE IF country = 'SA':

sa\_data = get\_sa\_tax\_data(tax\_year)

RETURN {

view\_mode: 'SA\_ONLY',

tax\_data: sa\_data,

personalized\_view: generate\_personalized\_sa\_view(user\_context, sa\_data)

}

# ===== UK TAX DATA STRUCTURE =====

FUNCTION get\_uk\_tax\_data(tax\_year: string) -> UkTaxData:

config = get\_uk\_tax\_config(tax\_year)

RETURN {

tax\_year: tax\_year,

tax\_year\_dates: {

start: format\_date(tax\_year, 'start'), # 6 April

end: format\_date(tax\_year, 'end') # 5 April

},

# Income Tax

income\_tax: {

personal\_allowance: {

standard: config.personal\_allowance, # £12,570 for 2024/25

description: "Amount you can earn tax-free",

taper: {

threshold: 100000,

rate: "£1 lost for every £2 over £100,000",

fully\_lost\_at: 125140

},

restrictions: [

"Reduced if income over £100,000",

"Not available if income over £125,140",

"May be restricted if claiming certain benefits"

]

},

tax\_bands: [

{

name: "Basic Rate",

band: "£0 - £37,700",

rate: 0.20,

rate\_display: "20%",

description: "First £37,700 of taxable income",

applies\_to: "Taxable income after personal allowance"

},

{

name: "Higher Rate",

band: "£37,701 - £125,140",

rate: 0.40,

rate\_display: "40%",

description: "Taxable income between £37,701 and £125,140"

},

{

name: "Additional Rate",

band: "Over £125,140",

rate: 0.45,

rate\_display: "45%",

description: "Taxable income over £125,140"

}

],

scottish\_tax\_bands: [

{

name: "Starter Rate",

band: "£0 - £2,162",

rate: 0.19,

rate\_display: "19%",

note: "Applies to Scottish residents only"

},

{

name: "Basic Rate",

band: "£2,163 - £13,118",

rate: 0.20,

rate\_display: "20%"

},

{

name: "Intermediate Rate",

band: "£13,119 - £31,092",

rate: 0.21,

rate\_display: "21%"

},

{

name: "Higher Rate",

band: "£31,093 - £125,140",

rate: 0.42,

rate\_display: "42%"

},

{

name: "Top Rate",

band: "Over £125,140",

rate: 0.47,

rate\_display: "47%"

}

],

savings\_rates: {

starting\_rate: {

amount: 5000,

rate: 0.00,

condition: "Available if income below £17,570"

},

personal\_savings\_allowance: {

basic\_rate: 1000,

higher\_rate: 500,

additional\_rate: 0,

description: "Tax-free interest on savings"

}

},

marriage\_allowance: {

transferable\_amount: 1260, # 10% of personal allowance

tax\_saving: 252, # 20% of transferable amount

eligibility: "One partner not using full personal allowance, other a basic rate taxpayer"

},

examples: [

{

scenario: "£30,000 salary",

calculation: {

gross\_income: 30000,

personal\_allowance: 12570,

taxable\_income: 17430,

tax\_due: 3486,

effective\_rate: "11.62%",

breakdown: [

"£17,430 @ 20% = £3,486"

]

}

},

{

scenario: "£60,000 salary",

calculation: {

gross\_income: 60000,

personal\_allowance: 12570,

taxable\_income: 47430,

basic\_rate\_tax: 7540,

higher\_rate\_tax: 3892,

total\_tax: 11432,

effective\_rate: "19.05%",

breakdown: [

"£37,700 @ 20% = £7,540",

"£9,730 @ 40% = £3,892"

]

}

}

]

},

# National Insurance

national\_insurance: {

class\_1\_employee: {

primary\_threshold: config.ni\_primary\_threshold, # £12,570

upper\_earnings\_limit: config.ni\_upper\_earnings\_limit, # £50,270

rates: [

{

band: "£12,570 - £50,270",

rate: 0.08,

rate\_display: "8%"

},

{

band: "Over £50,270",

rate: 0.02,

rate\_display: "2%"

}

]

},

class\_1\_employer: {

secondary\_threshold: 9100,

rate: 0.138,

rate\_display: "13.8%",

note: "Paid by employer on earnings above £9,100"

},

class\_2\_self\_employed: {

small\_profits\_threshold: 12570,

weekly\_rate: 3.45,

annual\_equivalent: 179.40,

note: "Flat rate if profits over £12,570"

},

class\_4\_self\_employed: {

lower\_profits\_limit: 12570,

upper\_profits\_limit: 50270,

rates: [

{

band: "£12,570 - £50,270",

rate: 0.06,

rate\_display: "6%"

},

{

band: "Over £50,270",

rate: 0.02,

rate\_display: "2%"

}

]

}

},

# Capital Gains Tax

capital\_gains\_tax: {

annual\_exempt\_amount: config.cgt\_annual\_exemption, # £3,000 for 2024/25

rates: {

residential\_property: {

basic\_rate: 0.18,

higher\_rate: 0.24,

rates\_display: "18% / 24%"

},

other\_assets: {

basic\_rate: 0.10,

higher\_rate: 0.20,

rates\_display: "10% / 20%"

}

},

exemptions: [

"Principal Private Residence (main home)",

"Personal possessions worth £6,000 or less",

"ISAs and PEPs",

"UK Government Gilts",

"Qualifying corporate bonds",

"Betting, lottery or pools winnings"

],

reliefs: {

business\_asset\_disposal\_relief: {

lifetime\_limit: 1000000,

rate: 0.10,

rate\_display: "10%",

description: "On disposal of business or business assets",

conditions: [

"Must have owned business for 2 years",

"Business must be trading company",

"Must be disposing of whole or part of business"

]

},

investors\_relief: {

lifetime\_limit: 10000000,

rate: 0.10,

rate\_display: "10%",

description: "On disposal of unlisted trading company shares",

conditions: [

"Shares must be newly issued",

"Must have held for 3 years from April 2016",

"Must be employee or officer of company"

]

}

}

},

# Dividend Tax

dividend\_tax: {

dividend\_allowance: config.dividend\_allowance, # £500 for 2024/25

rates: [

{

band: "Basic Rate",

rate: 0.0875,

rate\_display: "8.75%"

},

{

band: "Higher Rate",

rate: 0.3375,

rate\_display: "33.75%"

},

{

band: "Additional Rate",

rate: 0.3935,

rate\_display: "39.35%"

}

],

note: "Dividends received within ISAs are tax-free"

},

# ISA Allowances

isa\_allowances: {

overall\_limit: 20000,

types: [

{

name: "Cash ISA",

limit: 20000,

description: "Tax-free savings account"

},

{

name: "Stocks & Shares ISA",

limit: 20000,

description: "Tax-free investment account"

},

{

name: "Lifetime ISA (LISA)",

limit: 4000,

description: "For first home or retirement (age 60+)",

bonus: "25% government bonus",

restrictions: "Must be 18-39 to open, can contribute up to age 50"

},

{

name: "Innovative Finance ISA",

limit: 20000,

description: "Peer-to-peer lending"

},

{

name: "Junior ISA",

limit: 9000,

description: "For children under 18",

note: "Separate from adult allowance"

}

],

rules: [

"Can only contribute to one of each type per tax year",

"Can split £20,000 across types",

"LISA £4,000 counts toward overall £20,000",

"Can transfer between ISAs without affecting allowance"

]

},

# Pension Allowances

pension\_allowances: {

annual\_allowance: {

standard: 60000,

description: "Maximum pension contributions with tax relief",

money\_purchase\_annual\_allowance: {

amount: 10000,

triggered\_by: "Accessing pension flexibly",

description: "Reduced allowance if you've accessed pension"

},

tapered\_annual\_allowance: {

threshold\_income: 200000,

adjusted\_income: 260000,

taper\_rate: "£1 for every £2 over £260,000",

minimum: 10000,

description: "Reduced for high earners"

},

carry\_forward: {

years: 3,

description: "Can carry forward unused allowance from previous 3 years",

conditions: "Must have been member of pension scheme in those years"

}

},

lifetime\_allowance: {

note: "Abolished from April 2024",

replaced\_by: {

lump\_sum\_allowance: 268275,

lump\_sum\_death\_benefit\_allowance: 1073100

},

historical: {

"2023/24": 1073100,

"2020/21 - 2022/23": 1073100,

"2018/19 - 2019/20": 1055000

}

}

},

# Inheritance Tax

inheritance\_tax: {

nil\_rate\_band: {

amount: 325000,

transferable: true,

description: "Tax-free threshold for estates",

transferable\_note: "Unused portion can transfer to spouse"

},

residence\_nil\_rate\_band: {

maximum: 175000,

description: "Additional allowance for main residence left to direct descendants",

taper: {

threshold: 2000000,

rate: "£1 reduction for every £2 over £2m",

fully\_lost\_at: 2350000

},

transferable: true,

conditions: [

"Must leave main residence to direct descendants",

"Includes children, grandchildren, step-children",

"Property must have been residence at some point"

]

},

rate: {

standard: 0.40,

rate\_display: "40%",

reduced: 0.36,

reduced\_display: "36%",

reduced\_condition: "If 10% or more of estate left to charity"

},

exemptions: [

"Spouse/civil partner (unlimited)",

"Charity (unlimited)",

"Political parties (meeting certain conditions)",

"Annual exemption: £3,000",

"Small gifts: £250 per person",

"Wedding gifts: £5,000 (child), £2,500 (grandchild), £1,000 (other)"

],

pets\_and\_7\_year\_rule: {

description: "Potentially Exempt Transfers (PETs)",

rule: "Gifts become exempt if you survive 7 years",

taper\_relief: [

{years: "3-4", relief: "20%"},

{years: "4-5", relief: "40%"},

{years: "5-6", relief: "60%"},

{years: "6-7", relief: "80%"},

{years: "7+", relief: "100% (fully exempt)"}

],

note: "Taper relief reduces tax, not value"

},

business\_property\_relief: {

rates: {

"100%": [

"Business or interest in business",

"Shares in unlisted company"

],

"50%": [

"Shares controlling >50% of listed company",

"Land, buildings, machinery owned and used in business"

]

},

holding\_period: "Must be owned for 2 years"

},

agricultural\_property\_relief: {

rates: {

"100%": "Agricultural property with vacant possession or let after 1 Sept 1995",

"50%": "Other agricultural property"

},

holding\_period: "Must be owned for 2 years (or 7 if not occupied)"

}

},

# Stamp Duty Land Tax

stamp\_duty: {

residential: [

{

band: "£0 - £250,000",

rate: 0.00,

rate\_display: "0%",

first\_time\_buyer\_band: "£0 - £425,000"

},

{

band: "£250,001 - £925,000",

rate: 0.05,

rate\_display: "5%"

},

{

band: "£925,001 - £1,500,000",

rate: 0.10,

rate\_display: "10%"

},

{

band: "Over £1,500,000",

rate: 0.12,

rate\_display: "12%"

}

],

additional\_property\_surcharge: {

rate: 0.03,

rate\_display: "3%",

description: "Additional 3% on all bands for second homes and buy-to-let"

},

first\_time\_buyer\_relief: {

threshold: 625000,

relief\_up\_to: 425000,

rate\_after: 0.05,

description: "No SDLT up to £425,000 for first-time buyers (if property ≤£625,000)"

}

},

# Other Allowances

other\_allowances: {

blind\_persons\_allowance: {

amount: 3070,

transferable\_to\_spouse: true

},

trading\_allowance: {

amount: 1000,

description: "Tax-free trading income allowance"

},

property\_allowance: {

amount: 1000,

description: "Tax-free property income allowance"

}

}

}

# ===== SA TAX DATA STRUCTURE =====

FUNCTION get\_sa\_tax\_data(tax\_year: string) -> SaTaxData:

config = get\_sa\_tax\_config(tax\_year)

RETURN {

tax\_year: tax\_year,

tax\_year\_dates: {

start: format\_date(tax\_year, 'start'), # 1 March

end: format\_date(tax\_year, 'end') # 28/29 February

},

# Income Tax

income\_tax: {

tax\_brackets: [

{

band: "R0 - R237,100",

rate: 0.18,

rate\_display: "18%",

tax\_calculation: "18% of taxable income"

},

{

band: "R237,101 - R370,500",

rate: 0.26,

rate\_display: "26%",

tax\_calculation: "R42,678 + 26% of taxable income above R237,100"

},

{

band: "R370,501 - R512,800",

rate: 0.31,

rate\_display: "31%",

tax\_calculation: "R77,362 + 31% of taxable income above R370,500"

},

{

band: "R512,801 - R673,000",

rate: 0.36,

rate\_display: "36%",

tax\_calculation: "R121,475 + 36% of taxable income above R512,800"

},

{

band: "R673,001 - R857,900",

rate: 0.39,

rate\_display: "39%",

tax\_calculation: "R179,147 + 39% of taxable income above R673,000"

},

{

band: "Over R857,900",

rate: 0.45,

rate\_display: "45%",

tax\_calculation: "R251,258 + 45% of taxable income above R857,900"

}

],

rebates: {

primary: {

amount: config.primary\_rebate, # R17,235 for 2024/25

description: "Primary rebate for all individuals under 65",

age\_requirement: "Under 65"

},

secondary: {

amount: config.secondary\_rebate, # R19,500

description: "Additional rebate for individuals 65-74",

age\_requirement: "65-74 years",

total\_rebate: config.primary\_rebate + 2265

},

tertiary: {

amount: config.tertiary\_rebate, # R21,720

description: "Additional rebate for individuals 75+",

age\_requirement: "75 years and over",

total\_rebate: config.primary\_rebate + 4485

}

},

tax\_thresholds: {

below\_65: {

amount: 95750,

description: "Income below this amount is effectively tax-free for under 65s"

},

age\_65\_to\_74: {

amount: 148217,

description: "Income below this amount is effectively tax-free for 65-74s"

},

age\_75\_plus: {

amount: 165689,

description: "Income below this amount is effectively tax-free for 75+"

}

},

interest\_exemption: {

under\_65: {

amount: 23800,

description: "First R23,800 of local interest income is exempt"

},

age\_65\_plus: {

amount: 34500,

description: "First R34,500 of local interest income is exempt for 65+"

}

},

medical\_tax\_credits: {

main\_member: {

monthly: 364,

annual: 4368,

description: "Medical scheme fees tax credit for main member"

},

first\_dependent: {

monthly: 364,

annual: 4368

},

additional\_dependents: {

monthly: 246,

annual: 2952,

per: "each additional dependent"

},

additional\_credit: {

threshold\_age\_65\_plus: "4 × (medical scheme contributions - tax credits)",

threshold\_disabled: "4 × (medical scheme contributions - tax credits)",

rate: 0.33,

rate\_display: "33.3%",

description: "Additional credit for qualifying medical expenses"

}

},

examples: [

{

scenario: "R500,000 salary (age 40)",

calculation: {

taxable\_income: 500000,

tax\_before\_rebate: 86611,

primary\_rebate: 17235,

tax\_payable: 69376,

effective\_rate: "13.88%"

}

},

{

scenario: "R500,000 salary (age 70)",

calculation: {

taxable\_income: 500000,

tax\_before\_rebate: 86611,

secondary\_rebate: 19500,

tax\_payable: 67111,

effective\_rate: "13.42%"

}

}

]

},

# Capital Gains Tax

capital\_gains\_tax: {

annual\_exclusion: config.cgt\_annual\_exclusion, # R40,000 for 2024/25

inclusion\_rate: {

individuals: 0.40,

rate\_display: "40%",

description: "40% of capital gain is included in taxable income"

},

method: "Inclusion rate method",

description: "CGT is not a separate tax. Capital gains are added to taxable income after applying inclusion rate",

calculation\_example: {

capital\_gain: 100000,

annual\_exclusion: 40000,

net\_gain: 60000,

inclusion\_rate: 0.40,

taxable\_amount: 24000,

description: "R24,000 added to taxable income, taxed at marginal rate"

},

exemptions: [

"Primary residence (main dwelling)",

"Personal use assets (furniture, clothing, etc.)",

"Retirement fund lump sums",

"Proceeds from life insurance policies",

"Compensation for personal injury, illness or defamation",

"Annual exclusion: R40,000"

],

special\_rules: {

primary\_residence: {

exemption: 2000000,

description: "R2m exclusion on primary residence",

condition: "Must be primary residence throughout ownership"

},

small\_business\_assets: {

age\_55\_plus\_exclusion: 1800000,

lifetime\_limit: 1800000,

conditions: [

"Must be 55 or older",

"Business owned for 5 years minimum",

"Active business asset"

]

}

}

},

# Dividends Tax

dividends\_tax: {

rate: config.dividend\_tax\_rate, # 20% for 2024/25

rate\_display: "20%",

description: "Withholding tax on dividends",

exemptions: [

"Dividends from retirement funds",

"Dividends from foreign companies",

"Dividends in specie (non-cash)",

"Dividends from shares held in tax-free savings account",

"Certain dividends between companies (inter-company)"

],

treatment: {

local: "Withheld at source by company paying dividend",

foreign: "Included in taxable income, may qualify for foreign tax credit"

}

},

# Tax-Free Savings Account

tfsa: {

annual\_contribution\_limit: 36000,

lifetime\_contribution\_limit: 500000,

rules: [

"No age limit",

"Contributions are not tax deductible",

"Returns (interest, dividends, capital gains) are tax-free",

"Withdrawals do not create additional contribution room",

"Lifetime limit is cumulative (not per year)",

"Penalties for excess contributions: 40% of excess amount"

],

benefits: [

"Interest tax-free",

"Dividends tax-free (no dividends tax)",

"Capital gains tax-free",

"No tax on withdrawals",

"Estate duty free"

]

},

# Retirement Fund Contributions

retirement\_contributions: {

section\_10c\_deduction: {

limit: "27.5% of remuneration or taxable income (whichever is higher)",

annual\_cap: 350000,

description: "Tax deduction for retirement fund contributions",

calculation: "MIN(contributions, 27.5% × MAX(remuneration, taxable income), R350,000)"

},

types\_of\_funds: [

"Pension Fund",

"Provident Fund",

"Retirement Annuity Fund (RA)"

],

lump\_sum\_tax\_tables: {

retirement: [

{

band: "R0 - R550,000",

rate: 0.00,

rate\_display: "0%"

},

{

band: "R550,001 - R770,000",

rate: 0.18,

rate\_display: "18%",

calculation: "18% of amount above R550,000"

},

{

band: "R770,001 - R1,155,000",

rate: 0.27,

rate\_display: "27%",

calculation: "R39,600 + 27% of amount above R770,000"

},

{

band: "Over R1,155,000",

rate: 0.36,

rate\_display: "36%",

calculation: "R143,550 + 36% of amount above R1,155,000"

}

],

withdrawal: [

{

band: "R0 - R27,500",

rate: 0.00,

rate\_display: "0%"

},

{

band: "R27,501 - R726,000",

rate: 0.18,

rate\_display: "18%"

},

{

band: "R726,001 - R1,089,000",

rate: 0.27,

rate\_display: "27%"

},

{

band: "Over R1,089,000",

rate: 0.36,

rate\_display: "36%"

}

],

note: "Retirement lump sums and withdrawal lump sums taxed separately"

}

},

# Estate Duty

estate\_duty: {

rate: 0.20,

rate\_display: "20%",

description: "Levied on dutiable amount of deceased estate",

abatement: {

amount: 3500000,

description: "First R3.5 million of estate is exempt"

},

calculation: "20% of (Estate value - R3.5m)",

deductions: {

section\_4q: [

"Funeral expenses",

"Costs of administration",

"Debts owed by deceased",

"Bequests to public benefit organizations",

"Property accruing to surviving spouse"

],

section\_4a: {

description: "Deduction for property deemed to be in deceased's estate but accruing to surviving spouse",

effect: "Effective estate splitting with spouse"

}

},

exemptions: [

"Property accruing to surviving spouse (unlimited)",

"Property left to public benefit organization (Section 18A approved)",

"Abatement: R3.5 million"

],

example: {

gross\_estate: 10000000,

deductions: 1000000,

net\_estate: 9000000,

abatement: 3500000,

dutiable\_amount: 5500000,

estate\_duty: 1100000,

calculation\_display: "20% of (R9m - R3.5m) = R1.1m"

}

},

# Transfer Duty

transfer\_duty: {

description: "Tax on transfer of property",

rates: [

{

band: "R0 - R1,100,000",

rate: 0.00,

rate\_display: "0%"

},

{

band: "R1,100,001 - R1,512,500",

rate: 0.03,

rate\_display: "3%",

calculation: "3% of value above R1,100,000"

},

{

band: "R1,512,501 - R2,117,500",

rate: 0.06,

rate\_display: "6%",

calculation: "R12,375 + 6% of value above R1,512,500"

},

{

band: "R2,117,501 - R2,722,500",

rate: 0.08,

rate\_display: "8%",

calculation: "R48,675 + 8% of value above R2,117,500"

},

{

band: "R2,722,501 - R12,100,000",

rate: 0.11,

rate\_display: "11%",

calculation: "R97,075 + 11% of value above R2,722,500"

},

{

band: "Over R12,100,000",

rate: 0.13,

rate\_display: "13%",

calculation: "R1,128,625 + 13% of value above R12,100,000"

}

],

exemptions: [

"Transfers to/from spouse",

"Transfers to certain public benefit organizations",

"Certain government-to-government transfers"

]

},

# Donations Tax

donations\_tax: {

rate: 0.20,

rate\_display: "20%",

description: "Tax on value of property donated",

annual\_exemption: {

amount: 100000,

description: "R100,000 per year tax-free donations"

},

exemptions: [

"Donations to spouse",

"Donations to public benefit organizations (Section 18A)",

"Donations not exceeding R100,000 per annum",

"Bona fide maintenance payments",

"Donations to political parties"

],

casual\_gifts: {

exemption: 5000,

description: "Casual gifts under R5,000 exempt"

}

},

# Other Taxes

other\_taxes: {

securities\_transfer\_tax: {

rate: 0.0025,

rate\_display: "0.25%",

description: "On transfer of listed securities (shares)",

maximum: "Uncapped"

},

vat: {

standard\_rate: 0.15,

rate\_display: "15%",

description: "Value-Added Tax",

registration\_threshold: 1000000,

note: "Compulsory registration if turnover exceeds R1 million in 12 months"

}

}

}

# ===== PERSONALIZED VIEW =====

FUNCTION generate\_personalized\_view(

user\_context: UserContext,

uk\_data: UkTaxData,

sa\_data: SaTaxData

) -> PersonalizedTaxView:

personalized = {

your\_tax\_rates: {},

applicable\_allowances: {},

tax\_planning\_tips: []

}

# UK personalization

IF user\_context.uk\_tax\_resident:

# Determine user's tax band

IF user\_context.income.annual <= 12570:

personalized.your\_tax\_rates.uk\_income\_tax = "0% (within personal allowance)"

ELSE IF user\_context.income.annual <= 50270:

personalized.your\_tax\_rates.uk\_income\_tax = "20% (basic rate taxpayer)"

ELSE IF user\_context.income.annual <= 125140:

personalized.your\_tax\_rates.uk\_income\_tax = "40% (higher rate taxpayer)"

ELSE:

personalized.your\_tax\_rates.uk\_income\_tax = "45% (additional rate taxpayer)"

# Personal allowance status

IF user\_context.income.annual > 100000:

allowance\_lost = MIN((user\_context.income.annual - 100000) / 2, 12570)

remaining\_allowance = 12570 - allowance\_lost

personalized.applicable\_allowances.personal\_allowance = {

amount: remaining\_allowance,

note: "Reduced due to income over £100,000"

}

ELSE:

personalized.applicable\_allowances.personal\_allowance = {

amount: 12570,

note: "Full personal allowance available"

}

# Savings allowance

IF user\_context.income.annual <= 50270:

personalized.applicable\_allowances.personal\_savings\_allowance = {

amount: 1000,

note: "Basic rate taxpayer"

}

ELSE IF user\_context.income.annual <= 125140:

personalized.applicable\_allowances.personal\_savings\_allowance = {

amount: 500,

note: "Higher rate taxpayer"

}

ELSE:

personalized.applicable\_allowances.personal\_savings\_allowance = {

amount: 0,

note: "Not available for additional rate taxpayers"

}

# Dividend allowance

personalized.applicable\_allowances.dividend\_allowance = {

amount: 500,

note: "Applies to all taxpayers"

}

# ISA allowance

personalized.applicable\_allowances.isa\_allowance = {

amount: 20000,

used: user\_context.isa\_used,

remaining: 20000 - user\_context.isa\_used

}

# Pension annual allowance

IF user\_context.income.adjusted > 260000:

tapered\_allowance = MAX(60000 - ((user\_context.income.adjusted - 260000) / 2), 10000)

personalized.applicable\_allowances.pension\_annual\_allowance = {

amount: tapered\_allowance,

note: "Tapered due to high income"

}

ELSE:

personalized.applicable\_allowances.pension\_annual\_allowance = {

amount: 60000,

note: "Full annual allowance available"

}

# SA personalization

IF user\_context.sa\_tax\_resident:

# Determine user's marginal rate

IF user\_context.income.annual <= 237100:

personalized.your\_tax\_rates.sa\_income\_tax = "18% (marginal rate)"

ELSE IF user\_context.income.annual <= 370500:

personalized.your\_tax\_rates.sa\_income\_tax = "26% (marginal rate)"

ELSE IF user\_context.income.annual <= 512800:

personalized.your\_tax\_rates.sa\_income\_tax = "31% (marginal rate)"

ELSE IF user\_context.income.annual <= 673000:

personalized.your\_tax\_rates.sa\_income\_tax = "36% (marginal rate)"

ELSE IF user\_context.income.annual <= 857900:

personalized.your\_tax\_rates.sa\_income\_tax = "39% (marginal rate)"

ELSE:

personalized.your\_tax\_rates.sa\_income\_tax = "45% (marginal rate)"

# Tax rebate

IF user\_context.age < 65:

personalized.applicable\_allowances.sa\_rebate = {

amount: 17235,

type: "Primary rebate"

}

ELSE IF user\_context.age < 75:

personalized.applicable\_allowances.sa\_rebate = {

amount: 19500,

type: "Secondary rebate (65-74)"

}

ELSE:

personalized.applicable\_allowances.sa\_rebate = {

amount: 21720,

type: "Tertiary rebate (75+)"

}

# Interest exemption

IF user\_context.age < 65:

personalized.applicable\_allowances.interest\_exemption = {

amount: 23800,

note: "First R23,800 of interest tax-free"

}

ELSE:

personalized.applicable\_allowances.interest\_exemption = {

amount: 34500,

note: "First R34,500 of interest tax-free (65+)"

}

# TFSA

personalized.applicable\_allowances.tfsa = {

annual\_limit: 36000,

lifetime\_limit: 500000,

annual\_used: user\_context.tfsa\_used.annual,

lifetime\_used: user\_context.tfsa\_used.lifetime,

annual\_remaining: 36000 - user\_context.tfsa\_used.annual,

lifetime\_remaining: 500000 - user\_context.tfsa\_used.lifetime

}

# Retirement contribution deduction

max\_deduction = MIN(user\_context.income.annual \* 0.275, 350000)

personalized.applicable\_allowances.retirement\_deduction = {

max\_deductible: max\_deduction,

used: user\_context.retirement\_contributions,

remaining: max\_deduction - user\_context.retirement\_contributions

}

# Generate personalized tips

personalized.tax\_planning\_tips = generate\_tax\_tips(user\_context, personalized)

RETURN personalized

# ===== HISTORICAL COMPARISON =====

FUNCTION generate\_historical\_comparison(

country: enum['UK', 'SA'],

metric: string,

years: integer

) -> HistoricalComparison:

historical\_data = []

FOR year FROM (current\_tax\_year - years) TO current\_tax\_year:

IF country = 'UK':

config = get\_uk\_tax\_config(format\_tax\_year(year))

value = extract\_metric\_value(config, metric)

ELSE:

config = get\_sa\_tax\_config(format\_tax\_year(year))

value = extract\_metric\_value(config, metric)

historical\_data.append({

tax\_year: format\_tax\_year(year),

value: value

})

# Calculate changes

year\_over\_year\_changes = []

FOR i FROM 1 TO historical\_data.length - 1:

change = historical\_data[i].value - historical\_data[i-1].value

percentage\_change = (change / historical\_data[i-1].value) \* 100

year\_over\_year\_changes.append({

from\_year: historical\_data[i-1].tax\_year,

to\_year: historical\_data[i].tax\_year,

absolute\_change: change,

percentage\_change: percentage\_change

})

RETURN {

metric\_name: metric,

historical\_values: historical\_data,

changes: year\_over\_year\_changes,

trend: determine\_trend(historical\_data)

}

**User Interface Structure:**

TAX INFORMATION PAGE LAYOUT:

┌─────────────────────────────────────────────────────────┐

│ TAX INFORMATION CENTER │

│ │

│ [Country: UK ▼] [Tax Year: 2024/25 ▼] [👤 Personal View]│

└─────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────┐

│ QUICK NAVIGATION │

│ • Income Tax • National Insurance • CGT • Dividends │

│ • ISAs • Pensions • Inheritance Tax • Other │

└─────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────┐

│ YOUR TAX STATUS (Personalized) │

│ ┌─────────────┬──────────────┬────────────────────────┐│

│ │Your Marginal│Personal │ISA Allowance ││

│ │Rate: 40% │Allowance:£12,570│Used: £8,000/£20,000││

│ └─────────────┴──────────────┴────────────────────────┘│

└─────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────┐

│ INCOME TAX [Show Details ▼]│

│ │

│ Personal Allowance: £12,570 │

│ • Tax-free amount you can earn │

│ • Reduced if income over £100,000 │

│ │

│ TAX BANDS (2024/25) │

│ ├─ Basic Rate: £0-£37,700 @ 20% │

│ ├─ Higher Rate: £37,701-£125,140 @ 40% ◄ You are here│

│ └─ Additional: Over £125,140 @ 45% │

│ │

│ [Try Calculator] [View Examples] [Historical Rates] │

└─────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────┐

│ TAX CALCULATOR WIDGET │

│ Enter your income: £ [\_\_\_\_\_\_] │

│ ┌────────────────────────────────────────────┐ │

│ │ Gross Income: £60,000 │ │

│ │ Personal Allowance: -£12,570 │ │

│ │ Taxable Income: £47,430 │ │

│ │ │ │

│ │ Tax Due: │ │

│ │ £37,700 @ 20% = £7,540 │ │

│ │ £9,730 @ 40% = £3,892 │ │

│ │ Total Tax: £11,432 │ │

│ │ Effective Rate: 19.05% │ │

│ └────────────────────────────────────────────┘ │

└─────────────────────────────────────────────────────────┘

[Similar sections for each tax type...]

┌─────────────────────────────────────────────────────────┐

│ COMPARISON TOOLS │

│ • Year-over-Year Changes │

│ • UK vs SA Comparison │

│ • Historical Trends (5 years) │

└─────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────┐

│ EDUCATIONAL RESOURCES │

│ • Tax Planning Guides │

│ • Official Government Links (HMRC / SARS) │

│ • Glossary of Terms │

│ • Video Tutorials │

└─────────────────────────────────────────────────────────┘

**API Endpoints:**

GET /api/v1/tax-information/uk/{taxYear}

GET /api/v1/tax-information/sa/{taxYear}

GET /api/v1/tax-information/comparison/{taxYear}

GET /api/v1/tax-information/personalized/{userId}/{taxYear}

GET /api/v1/tax-information/historical/{country}/{metric}

GET /api/v1/tax-information/search?q={query}

POST /api/v1/tax-information/calculate/{taxType}

GET /api/v1/tax-information/changes/{taxYear} // What changed this year

GET /api/v1/tax-information/export/{format}

**Data Models:**

TABLE: tax\_information\_content

- id: UUID (PK)

- country: ENUM('UK', 'SA')

- tax\_type: VARCHAR(100)

- section: VARCHAR(100)

- content: TEXT

- content\_type: ENUM('DESCRIPTION', 'EXAMPLE', 'RULE', 'TIP')

- effective\_tax\_year: VARCHAR(10)

- last\_updated: TIMESTAMP

- source\_reference: TEXT (official source URL)

TABLE: tax\_year\_changes

- id: UUID (PK)

- tax\_year: VARCHAR(10)

- country: ENUM('UK', 'SA')

- change\_type: ENUM('RATE\_CHANGE', 'ALLOWANCE\_CHANGE', 'NEW\_RULE', 'ABOLISHED')

- metric\_affected: VARCHAR(100)

- old\_value: DECIMAL(15,2)

- new\_value: DECIMAL(15,2)

- description: TEXT

- impact\_assessment: TEXT

- announced\_date: DATE

- effective\_date: DATE

TABLE: user\_tax\_page\_preferences

- user\_id: UUID (PK, FK to users)

- default\_country: ENUM('UK', 'SA', 'BOTH')

- show\_personalized\_view: BOOLEAN DEFAULT TRUE

- favorite\_sections: JSON (array)

- collapsed\_sections: JSON (array)

- last\_viewed\_tax\_year: VARCHAR(10)

INDEX on tax\_information\_content(country, tax\_type, effective\_tax\_year)

INDEX on tax\_year\_changes(tax\_year, country)

**CONCLUSION**

This Features Document provides comprehensive technical specifications for building a sophisticated dual-country financial planning platform. The modular architecture allows for phased development while the detailed specifications ensure consistency and completeness across all features.

**Key Differentiators:**

1. **True dual-country integration**: Not just two separate systems, but genuine cross-border tax optimization
2. **Comprehensive coverage**: All major financial areas in both jurisdictions
3. **Intelligent automation**: AI-driven recommendations with personalization
4. **Goal-oriented approach**: Users plan for life goals, not just track numbers
5. **Scenario modeling**: Empowers users to make informed major decisions
6. **Tax optimization**: Built-in DTA application and cross-border planning

**Development Priorities:**

* Start with framework and one complete module
* Build tax engine early (dependencies across modules)
* Iterate based on user feedback
* Maintain modularity for independent development
* Focus on accuracy and regulatory compliance
* Invest in automation and intelligence features

This specification provides all the detail needed for development teams to begin implementation while maintaining flexibility for refinement during the build process.