

FPS Database Schema Guide

Overview

The FPS database contains **45 tables** organized into functional domains supporting a comprehensive UK-focused financial planning system. This guide maps database tables to application modules and shows how data flows through the system.

Table of Contents

- 1. [Core System Tables](#)
 - 2. [User & Authentication](#)
 - 3. [Estate Planning Module](#)
 - 4. [Protection Module](#)
 - 5. [Savings Module](#)
 - 6. [Investment Module](#)
 - 7. [Retirement Module](#)
 - 8. [Cross-Module Tables](#)
 - 9. [Table Relationships Map](#)
 - 10. [Data Flow Patterns](#)
-

Core System Tables

users (Central Hub)

Primary user account table - ALL other tables link back to this

Key Fields:

- id (PK)
- email, password, name
- date_of_birth, gender, marital_status
- spouse_id (FK → users.id) -- Links to spouse account
- household_id (FK → households.id)
- onboarding_completed, onboarding_focus_area
- annual_employment_income, annual_self_employment_income
- annual_rental_income, annual_dividend_income
- national_insurance_number, address fields

Relationships:

- Self-referencing: **spouse_id** links to another user
 - Parent to: ALL module-specific tables
 - Used by: All 5 modules + authentication
-

households

Groups related users (spouses, family members)

Key Fields:

- id (PK)
- household_name

Relationships:

- Referenced by: users, family_members, properties, cash_accounts, investment_accounts, trusts, business_interests, chattels

Usage:

- Enables joint/family asset tracking
 - Aggregates household net worth
 - Used in Estate & Net Worth calculations
-

family_members

Dependents and other family members

Key Fields:

- id (PK)
- user_id (FK → users.id)
- household_id (FK → households.id)
- relationship (spouse/child/parent/other_dependent)
- name, date_of_birth, gender, annual_income
- is_dependent, education_status

Relationships:

- Belongs to: users, households
 - Used by: Protection (dependents), Retirement (spouse planning), Estate (beneficiaries)
-

spouse_permissions

Manages spouse account linking and permissions

Key Fields:

- id (PK)
- user_id (FK → users.id)
- spouse_id (FK → users.id)
- status (pending/accepted/rejected)

Relationships:

- Links two user accounts bidirectionally
 - Created when spouse email entered in family members
 - Used for joint account visibility
-

User & Authentication

password_reset_tokens

Key Fields:

- email (PK)
- token
- created_at

personal_access_tokens

Laravel Sanctum API tokens

Key Fields:

- id (PK)
- tokenable_type, tokenable_id (polymorphic → users)
- token (hashed)
- abilities, expires_at, last_used_at

Usage:

- All API requests use Bearer token from this table
 - Token generated on login (/api/login)
 - Validated by Sanctum middleware
-

onboarding_progress

Tracks user onboarding flow

Key Fields:

- id (PK)
- user_id (FK → users.id)
- focus_area (estate/protection/retirement/investment/tax_optimisation)
- step_name, step_data (JSON)
- completed, skipped

Relationships:

- Belongs to: users
 - Used by: OnboardingService, OnboardingController
-

Estate Planning Module

Core Estate Tables

assets

Generic assets not in other specific tables

Key Fields:

- id (PK)
- user_id (FK → users.id)
- asset_type (property/pension/investment/business/other)
- asset_name, current_value
- ownership_type (individual/joint/trust)
- is_iht_exempt, exemption_reason

Usage:

- Aggregated for net worth calculation
 - Used in IHT calculation (EstateAgent)
-

properties

Real estate holdings (primary residence, buy-to-let, etc.)

Key Fields:

- id (PK)
- user_id (FK → users.id)
- joint_owner_id (FK → users.id) -- For joint ownership
- household_id (FK → households.id)
- trust_id (FK → trusts.id)
- property_type, ownership_type, ownership_percentage
- address_line_1, postcode
- purchase_date, purchase_price, current_value
- monthly_rental_income, annual_rental_income
- outstanding_mortgage
- annual_service_charge, annual_ground_rent, etc.

Relationships:

- Belongs to: users, households, trusts
- Has many: mortgages
- Used in: Estate (IHT), Net Worth, RNRB calculation

Code Files:

- Backend: `app/Models/Property.php`, `app/Services/Property/PropertyService.php`
 - Frontend: `resources/js/components/NetWorth/Property/*.vue`
-

mortgages**Mortgage loans linked to properties****Key Fields:**

- `id` (PK)
- `property_id` (FK → `properties.id`)
- `user_id` (FK → `users.id`)
- `joint_owner_id` (FK → `users.id`)
- `lender_name`, `outstanding_balance`
- `mortgage_type` (`repayment/interest_only/part_and_part`)
- `interest_rate`, `rate_type`, `monthly_payment`
- `remaining_term_months`

Relationships:

- Belongs to: `properties`, `users`
- Used in: Estate (liabilities for net estate), Net Worth

Code Files:

- Backend: `app/Models/Mortgage.php`,
`app/Http/Controllers/Api/MortgageController.php`
 - Frontend: `resources/js/components/NetWorth/Property/MortgageForm.vue`
-

liabilities**All other debts (loans, credit cards, etc.)****Key Fields:**

- `id` (PK)
- `user_id` (FK → `users.id`)
- `liability_type`
(`mortgage/secured_loan/personal_loan/credit_card/overdraft/etc.`)
- `current_balance`, `monthly_payment`, `interest_rate`
- `is_priority_debt`

Relationships:

- Belongs to: `users`
- Used in: Estate (reduces net estate), Protection (debts to cover), Net Worth

Code Files:

- Backend: `app/Models/Estate/Liability.php`
 - Used by: EstateAgent, NetWorthAnalyzer
-

trusts**Trust structures for estate planning****Key Fields:**

- id (PK)
- user_id (FK → users.id)
- household_id (FK → households.id)
- trust_name
- trust_type (bare/interest_in_possession/discretionary/etc.)
- trust_creation_date, initial_value, current_value
- is_relevant_property_trust, last_periodic_charge_date
- beneficiaries, trustees (TEXT)

Relationships:

- Belongs to: users, households
- Referenced by: properties, investment_accounts, cash_accounts, business_interests, chattels
- Used in: Estate (IHT calculations, 10-year charge)

Code Files:

- Backend: `app/Models/Trust.php`, `app/Services/Estate/TrustService.php`
 - Frontend: `resources/js/components/Estate/TrustForm.vue`
-

gifts**Record of lifetime gifts (PETs, CLTs)****Key Fields:**

- id (PK)
- user_id (FK → users.id)
- gift_date, recipient
- gift_type (pet/clt/exempt/small_gift/annual_exemption)
- gift_value
- status (within_7_years/survived_7_years)
- taper_relief_applicable

Relationships:

- Belongs to: users

- Used in: IHT calculation (7-year rule, taper relief)

Code Files:

- Backend: `app/Models/Gift.php`, `app/Services/Estate/GiftingService.php`
 - Used by: IHTCalculator for CLT calculation
-

wills

Will information**Key Fields:**

- id (PK)
- user_id (FK → users.id)
- death_scenario (user_only/both_simultaneous)
- spouse_primary_beneficiary, spouse_bequest_percentage
- last_reviewed_date

Relationships:

- Belongs to: users
 - Has many: bequests
-

bequests

Specific bequests in a will**Key Fields:**

- id (PK)
- will_id (FK → wills.id)
- user_id (FK → users.id)
- beneficiary_name, beneficiary_user_id (FK → users.id)
- bequest_type (percentage/specific_amount/specific_asset/residuary)
- percentage_of_estate, specific_amount
- asset_id, priority_order

Relationships:

- Belongs to: wills, users
 - Used in: Estate planning scenarios
-

iht_profiles

IHT-specific settings

Key Fields:

- id (PK)
- user_id (FK → users.id)
- marital_status, has_spouse, own_home
- nrb_transferred_from_spouse (£325k transferable)
- charitable_giving_percent (reduces IHT by 10% if ≥10%)

Relationships:

- Belongs to: users (one-to-one)
- Used by: IHTCalculator

Supporting Estate Tables

business_interests - Business ownership **chattels** - Valuable movable property (cars, art, jewelry)

cash_accounts - Cash holdings (current, savings, ISAs) - also used by Savings module

net_worth_statements - Historical net worth snapshots

Protection Module

protection_profiles

Core protection needs data**Key Fields:**

- id (PK)
- user_id (FK → users.id) - ONE-TO-ONE
- annual_income, monthly_expenditure
- mortgage_balance, other_debts
- number_of_dependents, dependents_ages (JSON)
- retirement_age, occupation, smoker_status, health_status

Relationships:

- Belongs to: users (one-to-one)
- Used by: ProtectionAgent for human capital calculation

Code Files:

- Backend: `app/Models/ProtectionProfile.php`, `app/Agents/ProtectionAgent.php`
- Frontend: `resources/js/components/Protection/ProtectionProfileForm.vue`

Insurance Policy Tables

life_insurance_policies

Key Fields:

- id (PK)
- user_id (FK → users.id)
- policy_type (term/whole_of_life/decreasing_term/family_income_benefit)
- provider, sum_assured, premium_amount
- policy_start_date, policy_term_years
- in_trust (boolean - IHT exempt if true)

Usage:

- Aggregated for total life cover
- Gap analysis: Required cover - Existing policies
- Used in Estate if in_trust (not part of estate)

critical_illness_policies**Key Fields:**

- id (PK)
- user_id (FK → users.id)
- policy_type (standalone/accelerated/additional)
- sum_assured, premium_amount
- conditions_covered (JSON)

income_protection_policies**Key Fields:**

- id (PK)
- user_id (FK → users.id)
- benefit_amount, benefit_frequency (monthly/weekly)
- deferred_period_weeks, benefit_period_months

disability_policies**Personal Accident & Disability cover**

sickness_illness_policies**Specific sickness cover**

expenditure_profiles

Monthly expenditure breakdown

Key Fields:

- id (PK)
- user_id (FK → users.id)
- monthly_housing, monthly_utilities, monthly_food
- monthly_transport, monthly_insurance, monthly_loans
- monthly_discretionary
- total_monthly_expenditure (calculated)

Relationships:

- Belongs to: users (one-to-one)
- Used by: ProtectionAgent (income replacement need), SavingsAgent (emergency fund calculation)

Code Files:

- Used in: [app/Services/Protection/CoverageGapAnalyzer.php](#)
-

Savings Module

savings_accounts

Savings accounts (including Cash ISAs)

Key Fields:

- id (PK)
- user_id (FK → users.id)
- joint_owner_id (FK → users.id)
- ownership_type (individual/joint/trust)
- account_type, institution, current_balance
- interest_rate, access_type (immediate/notice/fixed)
- is_isa, is_emergency_fund
- isa_type, isa_subscription_year, isa_subscription_amount

Relationships:

- Belongs to: users
- Has many: savings_goals (via linked_account_id)
- Used in: ISA allowance tracking, Emergency fund calculation, Net Worth

Code Files:

- Backend: [app/Models/SavingsAccount.php](#)
 - Frontend: [resources/js/components/Savings/*.vue](#)
 - Used by: SavingsAgent, ISATracker
-

savings_goals

Savings targets

Key Fields:

- id (PK)
- user_id (FK → users.id)
- linked_account_id (FK → savings_accounts.id)
- goal_name, target_amount, current_saved
- target_date, priority

Relationships:

- Belongs to: users, savings_accounts
-

isa_allowance_tracking

Cross-module ISA allowance tracking

Key Fields:

- id (PK)
- user_id (FK → users.id)
- tax_year (e.g., "2025/26")
- cash_isa_used, stocks_shares_isa_used, lisa_used
- total_used, total_allowance (£20,000)

Relationships:

- Belongs to: users
- **CRITICAL:** Updated by BOTH Savings and Investment modules
- Tracks combined ISA usage across modules

Code Files:

- Backend: `app/Services/Shared/ISATracker.php`
- Used by: SavingsAgent, InvestmentAgent
- Ensures total ISA subscriptions ≤ £20k per tax year

Data Flow:

```
User adds Cash ISA (Savings) → ISATracker.recordSubscription()  
User adds S&S ISA (Investment) → ISATracker.recordSubscription()  
ISATracker checks: cash_isa_used + stocks_shares_isa_used ≤ £20,000
```

Investment Module

investment_accounts

Investment wrapper accounts

Key Fields:

- id (PK)
- user_id (FK → users.id)
- joint_owner_id, household_id, trust_id
- ownership_type (individual/joint/trust)
- account_type (isa/gia/onshore_bond/offshore_bond/vct/eis)
- provider, platform, current_value
- isa_type, isa_subscription_current_year (for ISAs)
- platform_fee_percent

Relationships:

- Belongs to: users, households, trusts
- Has many: holdings
- Used in: ISA tracking (if account_type = 'isa'), Portfolio analysis, Net Worth

Code Files:

- Backend: [app/Models/InvestmentAccount.php](#)
 - Frontend: [resources/js/components/Investment/*.vue](#)
-

holdings

Individual securities/funds within investment accounts

Key Fields:

- id (PK)
- investment_account_id (FK → investment_accounts.id)
- asset_type (equity/bond/fund/etf/uk_equity/us_equity/cash/property)
- allocation_percent
- security_name, ticker, isin
- quantity, purchase_price, current_price, current_value
- dividend_yield, ocf_percent (ongoing charges)

Relationships:

- Belongs to: investment_accounts
- Used in: Asset allocation analysis, Fee calculation, Monte Carlo projections

Code Files:

- Backend: [app/Models/Holding.php](#), [app/Services/Investment/PortfolioAnalyzer.php](#)
-

investment_goals

Investment objectives

Key Fields:

- id (PK)
- user_id (FK → users.id)
- goal_name, goal_type (retirement/education/wealth/home)
- target_amount, target_date, priority
- linked_account_ids (JSON array of investment_account.id)

Relationships:

- Belongs to: users
 - Links to: investment_accounts (via JSON array)
-

risk_profiles

Investment risk assessment

Key Fields:

- id (PK)
- user_id (FK → users.id)
- risk_tolerance (cautious/balanced/adventurous)
- capacity_for_loss_percent
- time_horizon_years, knowledge_level
- esg_preference

Relationships:

- Belongs to: users
 - Used by: InvestmentAgent for portfolio recommendations
-

Retirement Module

retirement_profiles

Retirement planning parameters

Key Fields:

- id (PK)
- user_id (FK → users.id)
- current_age, target_retirement_age
- current_annual_salary, target_retirement_income
- essential_expenditure, lifestyle_expenditure

- life_expectancy, spouse_life_expectancy
- risk_tolerance

Relationships:

- Belongs to: users
 - Used by: RetirementAgent for decumulation planning
-

dc_pensions

Defined Contribution pensions**Key Fields:**

- id (PK)
- user_id (FK → users.id)
- scheme_name, scheme_type (workplace/sipp/personal)
- provider, current_fund_value
- employee_contribution_percent, employer_contribution_percent
- monthly_contribution_amount
- retirement_age, projected_value_at_retirement

Relationships:

- Belongs to: users
- Used in: Retirement projections, Annual allowance tracking, Net Worth

Code Files:

- Backend: `app/Models/DCPension.php`
 - Used by: RetirementAgent, NetWorthAnalyzer
-

db_pensions

Defined Benefit pensions (final salary, career average)**Key Fields:**

- id (PK)
- user_id (FK → users.id)
- scheme_name, scheme_type (final_salary/career_average/public_sector)
- accrued_annual_pension
- pensionable_service_years, pensionable_salary
- normal_retirement_age, spouse_pension_percent
- inflation_protection (cpi/rpi/fixed/none)

Relationships:

- Belongs to: users
 - Used in: Retirement income projections (NOT transfer value in net worth)
-

state_pensions

UK State Pension forecast

Key Fields:

- id (PK)
- user_id (FK → users.id)
- ni_years_completed, ni_years_required (35 for full pension)
- state_pension_forecast_annual, state_pension_age
- ni_gaps (JSON), gap_fill_cost

Relationships:

- Belongs to: users
 - Used in: Retirement income projections
-

Cross-Module Tables

recommendation_tracking

Unified recommendations from all agents

Key Fields:

- id (PK)
- user_id (FK → users.id)
- recommendation_id (unique string)
- module (Protection/Savings/Investment/Retirement/Estate)
- recommendation_text, priority_score
- timeline (immediate/short_term/medium_term/long_term)
- status (pending/in_progress/completed/dismissed)

Relationships:

- Belongs to: users
- Aggregated from: All 5 module agents + HolisticCoordinatingAgent

Code Files:

- Backend: `app/Services/Coordination/RecommendationsAggregatorService.php`
- Frontend: `resources/js/views/Dashboard.vue` (recommendations feed)

Data Flow:

Each Agent generates recommendations →
RecommendationsAggregatorService.aggregate() →
Conflict resolution & prioritization →
Stored in recommendation_tracking →
Displayed on Dashboard

tax_configurations

UK tax rules by tax year

Key Fields:

- id (PK)
- tax_year (e.g., "2025/26")
- effective_from, effective_to
- config_data (JSON) - All tax bands, thresholds, allowances
- is_active

JSON Structure:

```
{
  "income_tax": { "bands": [...], "personal_allowance": 12570 },
  "national_insurance": { "class_1": {...}, "class_2": {...} },
  "capital_gains": { "annual_exemption": 3000 },
  "iht": { "nil_rate_band": 325000, "residence_nil_rate_band": 175000 },
  "pensions": { "annual_allowance": 60000, "lifetime_allowance": null },
  "isa": { "overall_limit": 20000 }
}
```

Relationships:

- No foreign keys - referenced by tax_year
- **CRITICAL**: Used by ALL modules for tax calculations

Code Files:

- Backend: `app/Services/UKTaxCalculator.php`, `config/uk_tax_config.php`
- Seeded by: `database/seeder/TaxConfigurationSeeder.php`
- Cached with Memcached (1 hour TTL)

uk_life_expectancy_tables

ONS life expectancy data

Key Fields:

- id (PK)
- age, gender (male/female)
- life_expectancy_years
- table_version (ONS_2020_2022), data_year

Relationships:

- No user linkage - reference data
- Used by: RetirementAgent (decumulation planning), EstateAgent (second death projections)

personal_accounts

Self-employed P&L, cash flow, balance sheet tracking

Key Fields:

- id (PK)
- user_id (FK → users.id)
- account_type (profit_and_loss/cashflow/balance_sheet)
- period_start, period_end
- line_item, category, amount

Relationships:

- Belongs to: users
- Used for: Self-employed income tracking

System Tables

jobs

Laravel queue jobs (Monte Carlo simulations, etc.)

failed_jobs

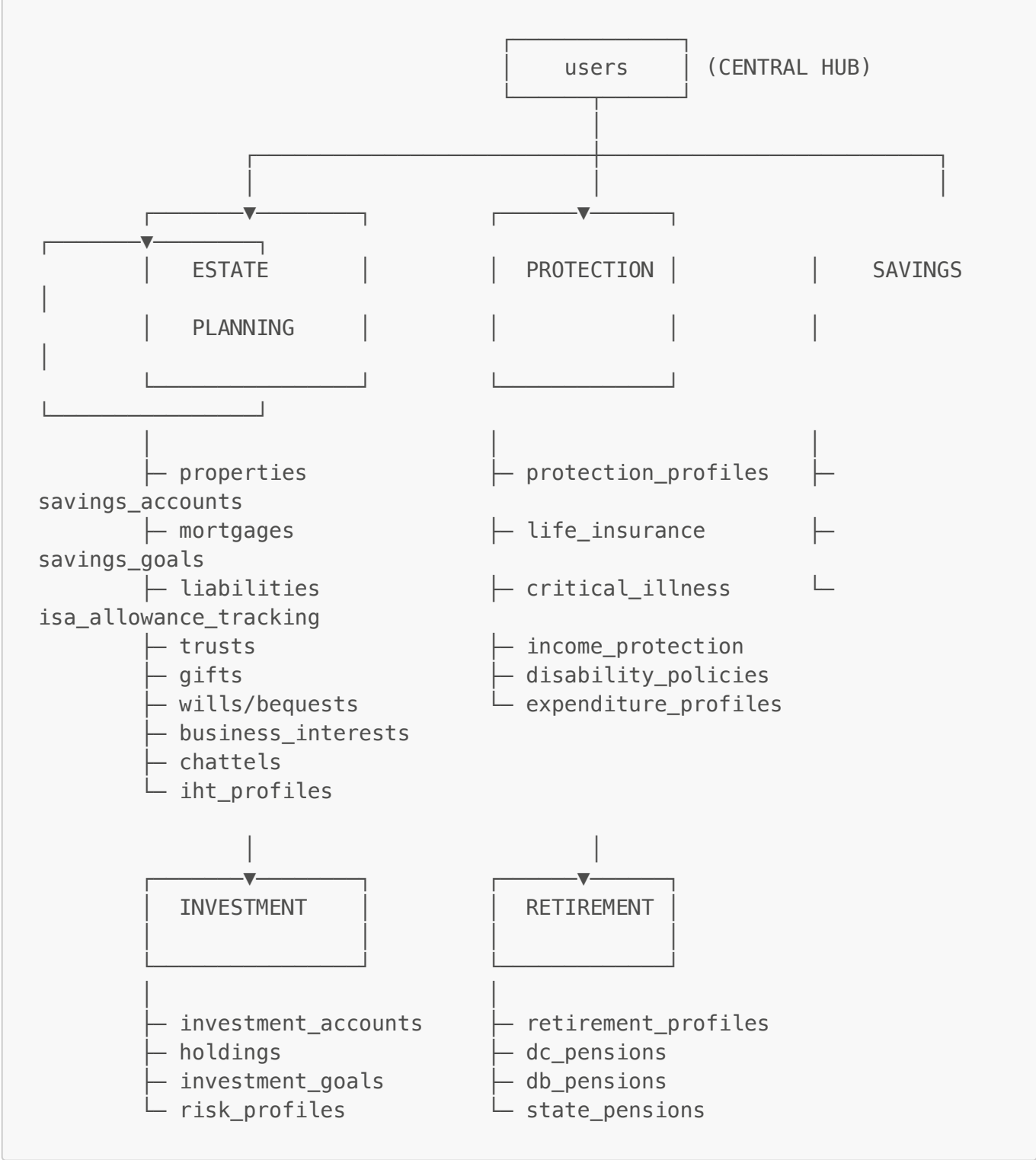
Failed queue jobs for debugging

migrations

Laravel migration history

Table Relationships Map

User-Centric Model (Hub & Spoke)



Cross-Module Relationships

Joint Ownership Pattern

users (id: 1)

users (id: 2)

→ properties (user_id: 1, joint_owner_id: 2)

Backend auto-creates reciprocal record:

properties (user_id: 2, joint_owner_id: 1)

Both users see the property in their accounts.

ISA Allowance Tracking (Cross-Module)

```
savings_accounts (is_isa: true) └─┐
                                └─┴─> isa_allowance_tracking (user_id: 1)
investment_accounts (account_type: 'isa') └─┐
```

ISATracker ensures: $\text{cash_isa_used} + \text{stocks_shares_isa_used} \leq \text{£20,000/year}$

Net Worth Aggregation

```
EstateAgent.calculateNetWorth() aggregates:
├─ Properties (current_value)
├─ DC Pensions (current_fund_value)
├─ Investment Accounts (current_value)
├─ Savings Accounts (current_balance)
├─ Business Interests (current_valuation)
├─ Chattels (current_value)
├─ Assets (current_value)
├─ MINUS Liabilities (current_balance)
└─ MINUS Mortgages (outstanding_balance)
```

Data Flow Patterns

1. User Registration & Onboarding

```
POST /api/register
┆
┆ AuthController → User created in `users` table
┆
┆ Sanctum token generated in `personal_access_tokens`
┆
┆ POST /api/onboarding
┆
┆ OnboardingService → Creates records in module-specific tables based on
┆ focus_area
┆
┆ users.onboarding_completed = true
```

2. Estate IHT Calculation Flow

```

GET /api/estate/calculate-ihl
↓
EstateController
↓
EstateAgent.calculateIHL()
↓
Aggregates data from:
├─ properties (current_value × ownership_percentage)
├─ investment_accounts (current_value × ownership_percentage)
├─ savings_accounts (current_balance × ownership_percentage)
├─ dc_pensions (current_fund_value)
├─ business_interests (current_valuation × ownership_percentage)
├─ chattels (current_value × ownership_percentage)
└─ assets (current_value)
↓
Minus:
├─ mortgages (outstanding_balance)
└─ liabilities (current_balance)
↓
IHLCalculator.calculate(netEstate, iht_profiles settings)
↓
Applies:
├─ NRB (£325k + transferred from spouse)
├─ RNRB (£175k if own_home)
├─ Gifts (7-year rule, taper relief)
└─ Charitable giving (10% reduction if ≥10%)
↓
Returns: { iht_liability, taxable_estate, breakdown }

```

3. Protection Gap Analysis Flow

```

GET /api/protection/analysis
↓
ProtectionController
↓
ProtectionAgent.analyze()
↓
Queries:
├─ protection_profiles (income, expenditure, debts, dependents)
├─ users (date_of_birth → age)
├─ family_members (dependents_ages)
├─ life_insurance_policies (sum_assured)
├─ critical_illness_policies (sum_assured)
└─ income_protection_policies (benefit_amount)
↓
UKTaxCalculator.calculateNetIncome(annual_income)
↓
CoverageGapAnalyzer:
├─ Human Capital = NPV(net_income until retirement_age)

```

```

├ Debt Coverage = mortgage_balance + other_debts
├ Dependent Support = years_to_independence × annual_expenditure
└ Required Cover = Human Capital + Debt Coverage + Dependent Support
↓
Gap = Required Cover - Existing Policies
↓
Returns: { adequacy_score, gaps, recommendations }

```

4. Savings Emergency Fund Flow

```

GET /api/savings/dashboard
↓
SavingsController
↓
SavingsAgent.analyze()
↓
Queries:
├ savings_accounts WHERE is_emergency_fund = true
└ expenditure_profiles (total_monthly_expenditure)
↓
Emergency Fund Runway =  $\Sigma(\text{emergency fund balances}) \div \text{monthly\_expenditure}$ 
↓
Recommended: 3–6 months (depends on employment status)
↓
Returns: { current_runway, recommended_runway, shortfall }

```

5. ISA Allowance Cross-Module Tracking

```

POST /api/savings/accounts (is_isa: true, isa_subscription: £5,000)
↓
SavingsController → SavingsAgent.store()
↓
ISATracker::recordSubscription(user_id, 'cash_isa', £5,000)
↓
Updates: isa_allowance_tracking.cash_isa_used += £5,000

POST /api/investment/accounts (account_type: 'isa', contributions_ytd:
£10,000)
↓
InvestmentController → InvestmentAgent.store()
↓
ISATracker::recordSubscription(user_id, 'stocks_shares_isa', £10,000)
↓
Updates: isa_allowance_tracking.stocks_shares_isa_used += £10,000

GET /api/isa/remaining-allowance
↓

```

```
ISATracker::getRemainingAllowance(user_id, tax_year)
↓
Returns: £20,000 - (£5,000 + £10,000) = £5,000 remaining
```

6. Recommendation Aggregation Flow

```
GET /api/dashboard
↓
DashboardController
↓
HolisticCoordinatingAgent.generateRecommendations()
↓
Calls each agent:
├─ ProtectionAgent.analyze() → recommendations[]
├─ SavingsAgent.analyze() → recommendations[]
├─ InvestmentAgent.analyze() → recommendations[]
├─ RetirementAgent.analyze() → recommendations[]
└─ EstateAgent.analyze() → recommendations[]
↓
RecommendationsAggregatorService.aggregate(allRecommendations)
↓
├─ Removes duplicates
├─ Resolves conflicts (e.g., emergency fund vs. pension contributions)
├─ Prioritizes by urgency & impact
└─ Stores in recommendation_tracking table
↓
Returns: Unified prioritized list to frontend
```

7. Joint Account Linking Flow

```
User 1 (id: 5) adds spouse via FamilyMemberFormModal
↓
POST /api/family-members
Body: { relationship: 'spouse', email: 'spouse@example.com' }
↓
FamilyMembersController → StoreFamilyMemberRequest
↓
Check if User 2 exists (email = 'spouse@example.com')
↓
IF NOT EXISTS:
├─ Create User 2 (id: 10) with random password
├─ Send welcome email
└─ Create family_members record (user_id: 5, name: "Spouse")

IF EXISTS:
└─ Link accounts bidirectionally:
    └─ users.spouse_id = 10 WHERE id = 5
```

```
| users.spouse_id = 5 WHERE id = 10  
| users.marital_status = 'married' for both  
| Create spouse_permissions record
```

User 2 can now see joint assets WHERE joint_owner_id = 10 OR user_id = 10

Key Code Files by Table

Properties & Mortgages

Backend:

- `app/Models/Property.php`
- `app/Models/Mortgage.php`
- `app/Http/Controllers/Api/PropertyController.php`
- `app/Http/Controllers/Api/MortgageController.php`
- `app/Services/Property/PropertyService.php`

Frontend:

- `resources/js/components/NetWorth/Property/PropertyCard.vue`
- `resources/js/components/NetWorth/Property/PropertyForm.vue`
- `resources/js/components/NetWorth/Property/MortgageForm.vue`
- `resources/js/services/propertyService.js`
- `resources/js/store/modules/netWorth.js`

Investment Accounts & Holdings

Backend:

- `app/Models/InvestmentAccount.php`
- `app/Models/Holding.php`
- `app/Http/Controllers/Api/InvestmentController.php`
- `app/Services/Investment/PortfolioAnalyzer.php`

Frontend:

- `resources/js/components/Investment/InvestmentAccountCard.vue`
- `resources/js/components/Investment/HoldingsList.vue`
- `resources/js/services/investmentService.js`

Savings & ISA Tracking

Backend:

- `app/Models/SavingsAccount.php`
- `app/Models/ISAAallowanceTracking.php`
- `app/Services/Shared/ISATracker.php`

- `app/Http/Controllers/Api/SavingsController.php`

Frontend:

- `resources/js/components/Savings/SaveAccountModal.vue`
 - `resources/js/components/Savings/ISAAllowanceCard.vue`
 - `resources/js/services/savingsService.js`
-

Protection Policies

Backend:

- `app/Models/LifeInsurancePolicy.php`
- `app/Models/CriticalIllnessPolicy.php`
- `app/Models/IncomeProtectionPolicy.php`
- `app/Services/Protection/CoverageGapAnalyzer.php`
- `app/Agents/ProtectionAgent.php`

Frontend:

- `resources/js/components/Protection/PolicyForm.vue`
 - `resources/js/components/Protection/GapAnalysisChart.vue`
-

Estate & IHT

Backend:

- `app/Models/Estate/Liability.php`
- `app/Models/Gift.php`
- `app/Models/Trust.php`
- `app/Services/Estate/IHTCalculator.php`
- `app/Services/Estate/NetWorthAnalyzer.php`
- `app/Agents/EstateAgent.php`

Frontend:

- `resources/js/components/Estate/EstateOverviewCard.vue`
 - `resources/js/components/Estate/IHTCalculationBreakdown.vue`
 - `resources/js/components/Estate/GiftingTimeline.vue`
-

Retirement Pensions

Backend:

- `app/Models/DCPension.php`
- `app/Models/DBPension.php`
- `app/Models/StatePension.php`
- `app/Agents/RetirementAgent.php`

Frontend:

- [resources/js/components/Retirement/PensionInventory.vue](#)
 - [resources/js/components/Retirement/ProjectionChart.vue](#)
-

Database Design Patterns

1. Ownership Pattern

All major asset tables support individual/joint/trust ownership:

```
ownership_type ENUM('individual', 'joint', 'trust')
ownership_percentage DECIMAL(5,2) DEFAULT 100.00
joint_owner_id BIGINT (FK to users)
trust_id BIGINT (FK to trusts)
```

Tables using this:

- properties
 - investment_accounts
 - savings_accounts
 - business_interests
 - chattels
-

2. User-Centric Foreign Keys

Every user-data table has `user_id` foreign key:

```
user_id BIGINT UNSIGNED NOT NULL
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
INDEX idx_user_id (user_id)
```

Enables:

- Fast user data queries
 - Cascade deletion for GDPR compliance
 - Row-level security
-

3. Profile One-to-One Pattern

Module profiles are one-to-one with users:

```
CREATE TABLE protection_profiles (
  id BIGINT UNSIGNED PRIMARY KEY,
```

```
    user_id BIGINT UNSIGNED UNIQUE NOT NULL,  
    ...  
)
```

Tables:

- protection_profiles
 - retirement_profiles
 - iht_profiles
 - expenditure_profiles
-

4. Tax Year Tracking

Tax-year-sensitive data includes `tax_year` field:

```
tax_year VARCHAR(10) -- e.g., "2025/26"
```

Tables:

- isa_allowance_tracking
 - investment_accounts
 - tax_configurations
-

5. JSON Flexible Data

Complex/evolving data stored as JSON:

```
dependents_ages JSON -- Protection  
linked_account_ids JSON -- Investment goals  
conditions_covered JSON -- Insurance policies  
step_data JSON -- Onboarding  
config_data JSON -- Tax configurations
```

Query Performance Notes

Key Indexes

users table:

- PRIMARY KEY (id)
- UNIQUE (email)
- INDEX (spouse_id)
- INDEX (household_id)

All asset tables:

- INDEX (user_id) -- Most common query
- INDEX (joint_owner_id) -- For joint assets
- INDEX (household_id) -- For household aggregation

investment_accounts:

- INDEX (user_id, account_type) -- Filter by ISA
- INDEX (household_id)

isa_allowance_tracking:

- INDEX (user_id, tax_year) -- Fast current year lookup

recommendation_tracking:

- INDEX (user_id, status) -- Pending recommendations query
- INDEX (recommendation_id) -- Unique constraint

Common Query Patterns**Get User's Net Worth Assets**

```
-- Properties
SELECT SUM(current_value * ownership_percentage / 100)
FROM properties WHERE user_id = ?

-- Investments
SELECT SUM(current_value * ownership_percentage / 100)
FROM investment_accounts WHERE user_id = ?

-- Savings
SELECT SUM(current_balance * ownership_percentage / 100)
FROM savings_accounts WHERE user_id = ?

-- Pensions
SELECT SUM(current_fund_value) FROM dc_pensions WHERE user_id = ?
```

Get User's Liabilities

```
-- Mortgages
SELECT SUM(outstanding_balance) FROM mortgages WHERE user_id = ?

-- Other Debts
SELECT SUM(current_balance) FROM liabilities WHERE user_id = ?
```

Get ISA Usage for Tax Year

```
SELECT cash_isa_used, stocks_shares_isa_used, total_used
FROM isa_allowance_tracking
WHERE user_id = ? AND tax_year = '2025/26'
```

Get Pending Recommendations

```
SELECT * FROM recommendation_tracking
WHERE user_id = ? AND status = 'pending'
ORDER BY priority_score DESC, created_at ASC
```

Data Integrity Rules

Critical Constraints

1. ISA Individual Ownership Only

- `savings_accounts`: IF `is_isa = true` THEN `ownership_type = 'individual'`
- `investment_accounts`: IF `account_type = 'isa'` THEN `ownership_type = 'individual'`
- Enforced: Backend validation (StoreSavingsAccountRequest, StoreInvestmentAccountRequest)

2. ISA Annual Allowance

- Total across all ISAs ≤ £20,000 per tax year
- Enforced: ISATracker service throws exception if exceeded

3. Joint Ownership Reciprocity

- IF `properties.joint_owner_id = X` THEN reciprocal record exists with `user_id = X`
- Enforced: PropertyService creates both records

4. Spouse Bidirectional Link

- IF `users.spouse_id = X` THEN `users.spouse_id = Y` WHERE `id = X`
- Enforced: FamilyMembersController updates both records

5. Ownership Percentage

- For joint assets: `ownership_percentage` typically 50.00
- Sum of percentages for same asset should = 100.00
- Enforced: Application logic

Caching Strategy

Memcached Keys

Tax Configurations:

```
Key: tax_config_{tax_year}
Value: config_data JSON
TTL: 3600 seconds (1 hour)
```

Net Worth Calculation:

```
Key: net_worth_{user_id}
Value: { total_assets, total_liabilities, net_worth }
TTL: 1800 seconds (30 minutes)
```

Dashboard Data:

```
Key: dashboard_{user_id}
Value: Complete dashboard JSON
TTL: 1800 seconds (30 minutes)
```

Invalidation:

- Any asset/liability update → Clear `net_worth_{user_id}`, `dashboard_{user_id}`
- Property/savings/investment change → Clear respective module caches

Migration History

Key Migrations

- `2014_10_12_000000_create_users_table.php`
- `2025_01_15_create_properties_table.php`
- `2025_01_20_create_mortgages_table.php`
- `2025_10_14_075637_create_assets_table.php`
- `2025_10_23_154600_update_assets_ownership_type_to_individual.php` (Latest)

Ownership Type Migration

Migration: `2025_10_23_154600_update_assets_ownership_type_to_individual.php`

Purpose: Change 'sole' → 'individual' for consistency across all tables

Affected Tables:

- assets
- properties

- investment_accounts
 - savings_accounts
 - business_interests
 - chattels
-

Testing Database Queries

Test User Data

```
-- Admin user
id: 1016
email: admin@fps.com
password: admin123456

-- Test user
id: 1
email: test@example.com
```

Useful Queries

See all tables:

```
SHOW TABLES FROM laravel;
```

See user's assets:

```
SELECT 'Properties' as type, COUNT(*) as count, SUM(current_value) as
total
FROM properties WHERE user_id = 1
UNION
SELECT 'Investments', COUNT(*), SUM(current_value)
FROM investment_accounts WHERE user_id = 1
UNION
SELECT 'Savings', COUNT(*), SUM(current_balance)
FROM savings_accounts WHERE user_id = 1;
```

See ISA usage:

```
SELECT * FROM isa_allowance_tracking WHERE user_id = 1 AND tax_year =
'2025/26';
```

Summary

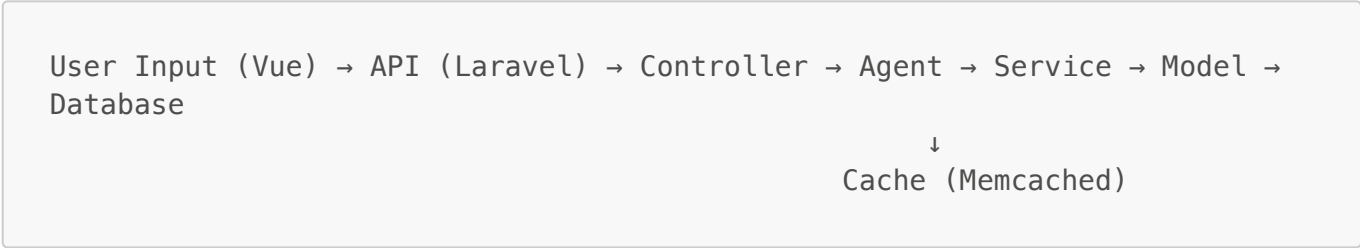
Database Statistics

- **45 tables** total
- **5 core modules** (Estate, Protection, Savings, Investment, Retirement)
- **39 user-data tables** (with **user_id** FK)
- **6 system tables** (auth, jobs, config)
- **~150 foreign key relationships**

Key Relationships

1. **users** → ALL module tables (hub & spoke)
2. **households** → Multi-user asset aggregation
3. **trusts** → Asset ownership wrapper
4. **isa_allowance_tracking** → Cross-module tracking (Savings + Investment)
5. **recommendation_tracking** → Cross-module recommendations

Data Flow



Critical Services

- **ISATracker** - Enforces £20k ISA allowance across modules
- **NetWorthAnalyzer** - Aggregates all assets/liabilities
- **IHTCalculator** - Estate planning calculations
- **UKTaxCalculator** - Income tax/NI calculations
- **RecommendationsAggregatorService** - Unified recommendations

Related Documentation

- **CODEBASE_STRUCTURE.md** - Full backend/frontend code structure
- **CODEBASE_FILE_MAP.md** - File locations and dependencies
- **FPS_Features_TechStack.md** - API specs and schemas
- **fpflow.md** - System architecture diagrams

Last Updated: October 24, 2025 **Version:** v0.1.2.2