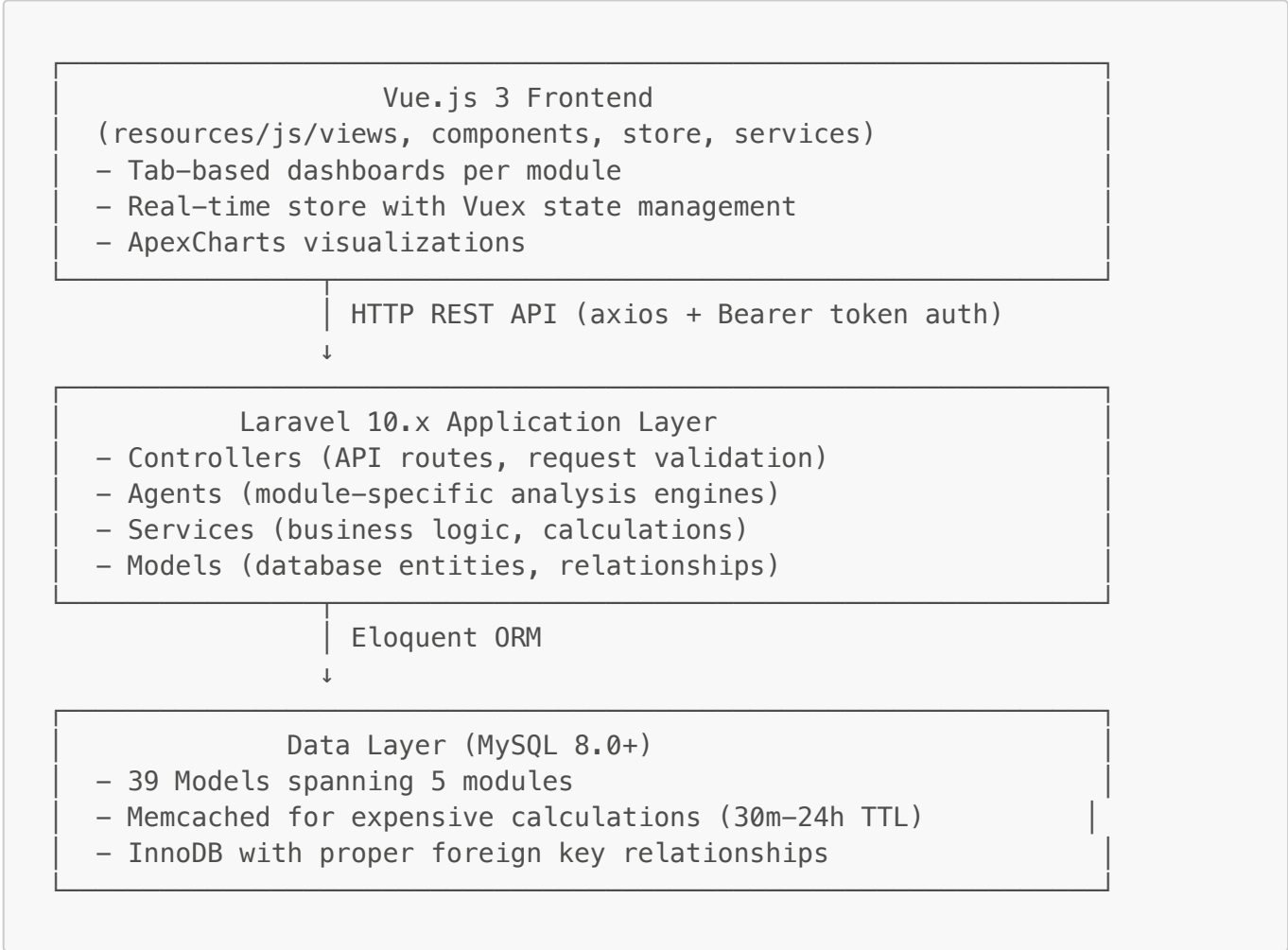
FPS (Financial Planning System) - Complete Codebase Structure & Architecture

Version: v0.1.2.2 **Last Updated**: October 2025 **Total Files**: 8,186 PHP, Vue, JS, and config files **Backend Services**: 4,811+ lines across Estate services alone

TABLE OF CONTENTS

ARCHITECTURE OVERVIEW

Three-Tier Architecture



Agent-Based Processing Pattern

Each module has an **Agent** that orchestrates analysis:

```
Vue Component
    ↓
calls service.method()
    ↓
JavaScript Service (resources/js/services/)
    ↓
HTTP POST/GET to /api/[module]/
    ↓
Controller (app/Http/Controllers/Api/)
    ↓
instantiates + calls Agent.analyze/generateRecommendations/buildScenarios
    ↓
Agent coordinates multiple Services
    ↓
Services call Models and perform calculations
    ↓
Response → Vuex Store → Component renders
```

BACKEND STRUCTURE

1. AGENTS (app/Agents/)

Core Agents - Each module has an agent that takes inputs and produces recommendations:

Agent	Responsibility	Key Services	Cache TTL
BaseAgent	Abstract base class, common utilities	N/A	3600s (1h default)
ProtectionAgent	Life/CI/IP coverage analysis	CoverageGapAnalyzer, AdequacyScorer, RecommendationEngine, ScenarioBuilder	3600s
SavingsAgent	Emergency fund, ISA tracking, savings goals	EmergencyFundCalculator, ISATracker, GoalProgressCalculator, LiquidityAnalyzer	3600s
InvestmentAgent	Portfolio analysis, Monte Carlo, asset allocation	PortfolioAnalyzer, MonteCarloSimulator, AssetAllocationOptimizer, TaxEfficiencyCalculator, FeeAnalyzer	3600s
RetirementAgent	Pension planning, contribution tracking, readiness	PensionProjector, AnnualAllowanceChecker, ContributionOptimizer, ReadinessScorer, DecumulationPlanner	3600s

Agent	Responsibility	Key Services	Cache TTL
EstateAgent	IHT, gifting strategies, net worth, second death planning	IHTCalculator, GiftingStrategy, NetWorthAnalyzer, CashFlowProjector	3600s
CoordinatingAgent	Cross-module holistic planning	HolisticPlanner, RecommendationsAggregatorService, CashFlowCoordinator	3600s

Key BaseAgent Methods:

```
abstract public function analyze(int $userId): array;
abstract public function generateRecommendations(array $analysisData):
array;
abstract public function buildScenarios(int $userId, array $parameters):
array;
protected function remember(string $key, callable $callback): mixed; //
Cache wrapper
protected function getCurrentTaxYear(): string; // Returns "2025/26"
protected function calculateAge($dateOfBirth): int;
protected function response($success, $message, $data): array; //
Standardized format
```

2. CONTROLLERS (app/Http/Controllers/Api/)

Authentication & User Management:

- `AuthController` - Login, register, password change
- `UserProfileController` - Profile updates, income/occupation
- `FamilyMembersController` - Spouse/family management
- `PersonalAccountsController` - P&L, Cashflow, Balance Sheet
- `SpousePermissionController` - Cross-spouse data access permissions

Module Controllers (25 total):

Estate Module Controllers

```
EstateController (main CRUD for assets/liabilities/gifts)
└── Estate/
    ├── IHTController (IHT calculations, spouse exemption, second death)
    ├── GiftingController (gifting strategy, timeline, discount
calculations)
    ├── LifePolicyController (life cover strategy, whole-of-life vs self-
insurance)
    ├── TrustController (trust CRUD, tax analysis, IHT impact)
    └── WillController (will planning, bequests, probate)
```

Other Modules

```
ProtectionController → policies (life, CI, income protection, disability)
SavingsController → accounts, goals, ISA tracking
InvestmentController → accounts, holdings, risk profile, Monte Carlo
RetirementController → DC/DB pensions, state pension, contributions
PropertyController → properties, mortgages
MortgageController → mortgage calculations, amortization schedules
NetWorthController → net worth aggregation
DashboardController → financial health score, alerts
HolisticPlanningController → cross-module recommendations
RecommendationsController → unified recommendations tracking
```

Pattern: Each controller method typically:

1. Gets authenticated user via `$request->user()`
2. Validates input (custom Form Request classes)
3. Instantiates agent or service via dependency injection
4. Calls `agent->analyze()` → `agent->generateRecommendations()` → formats response
5. Returns standardized JSON response with caching

3. SERVICES (app/Services/ - 50+ files)

Services are organized by domain and provide reusable business logic:

```
Services/
├── Shared/
│   └── CrossModuleAssetAggregator.php (aggregates assets from all
modules)
│
├── Estate/ (20+ files - most complex)
│   ├── IHTCalculator.php (core IHT calculation engine)
│   ├── NetWorthAnalyzer.php (asset aggregation, net worth trends)
│   ├── CashFlowProjector.php (P&L, cash flow by tax year)
│   ├── GiftingStrategy.php (PET/CLT analysis)
│   ├── GiftingStrategyOptimizer.php (optimal gifting timeline)
│   ├── GiftingTimelineService.php (7-year PET tracking)
│   ├── SecondDeathIHTCalculator.php (married couple IHT planning)
│   ├── LifePolicyStrategyService.php (life cover recommendations)
│   ├── FutureValueCalculator.php (actuarial projections)
│   ├── ActuarialLifeTableService.php (mortality/life expectancy)
│   ├── TrustService.php (trust lifecycle management)
│   ├── SpouseNRBTrackerService.php (NRB transfer tracking)
│   ├── EstateAssetAggregatorService.php (cross-module asset collection)
│   ├── IHTStrategyGeneratorService.php (mitigation strategies)
│   └── LifeCoverCalculator.php (life cover adequacy)
│
├── Protection/ (5 files)
```

```
│   ├── CoverageGapAnalyzer.php (needs vs coverage)
│   ├── AdequacyScorer.php (% adequacy, score insights)
│   ├── RecommendationEngine.php (gap-based recommendations)
│   └── ScenarioBuilder.php (death, CI, disability scenarios)
│
├── Savings/ (5 files)
│   ├── EmergencyFundCalculator.php (runway calculation)
│   ├── ISATracker.php (£20k allowance tracking across modules)
│   ├── GoalProgressCalculator.php (goal achievement timeline)
│   ├── LiquidityAnalyzer.php (liquidity assessment)
│   └── RateComparator.php (account rate comparison)
│
├── Investment/ (5 files)
│   ├── PortfolioAnalyzer.php (allocation, performance, holdings)
│   ├── MonteCarloSimulator.php (queued simulation job)
│   ├── AssetAllocationOptimizer.php (rebalancing suggestions)
│   ├── TaxEfficiencyCalculator.php (tax-loss harvesting)
│   └── FeeAnalyzer.php (fee impact on returns)
│
├── Retirement/ (5 files)
│   ├── PensionProjector.php (accumulation/drawdown projections)
│   ├── AnnualAllowanceChecker.php (£60k limit, 3-year carry forward)
│   ├── ContributionOptimizer.php (optimal contribution strategy)
│   ├── ReadinessScorer.php (retirement readiness %)
│   └── DecumulationPlanner.php (drawdown strategy)
│
├── Property/ (3 files)
│   ├── MortgageService.php (loan calculations, schedules)
│   ├── PropertyService.php (property data management)
│   └── PropertyTaxService.php (SDLT, CGT, rental income tax)
│
├── Trust/ (2 files)
│   ├── TrustAssetAggregatorService.php (trust assets from multiple
sources)
│   └── IHTPeriodicChargeCalculator.php (trust periodic charges)
│
├── Coordination/ (5 files)
│   ├── HolisticPlanner.php (cross-module planning)
│   ├── RecommendationsAggregatorService.php (unified recommendations)
│   ├── CashFlowCoordinator.php (household cash flow)
│   ├── PriorityRanker.php (recommendation prioritization)
│   └── ConflictResolver.php (conflicting recommendations)
│
├── Onboarding/ (2 files)
│   ├── OnboardingService.php (wizard flow management)
│   └── EstateOnboardingFlow.php (estate-specific onboarding)
│
├── Dashboard/ (1 file)
│   └── DashboardAggregator.php (main dashboard data)
│
├── UserProfile/ (2 files)
│   ├── UserProfileService.php (profile updates, validation)
│   └── PersonalAccountsService.php (P&L, cashflow, balance sheet)
│
```

```
├── NetWorth/ (1 file)
│   └── NetWorthService.php (net worth aggregation)
│
└── UKTaxCalculator.php (shared - income tax, NI calculation for all
income types)
```

Key Service Patterns:

- Most public methods accept `int $userId` as first parameter
- Complex calculations split into focused methods
- Return standardized arrays (not objects)
- Use cache `Cache::remember()` for expensive operations
- Validate inputs via type hints
- Config-driven rules via `config('uk_tax_config')`

4. MODELS (app/Models/ - 39 files)

Core Models:

```
User Model
├── Relationships: spouse_id (self), household_id, many protection
policies
├── Key attributes: date_of_birth, marital_status, annual_*_income
└── Methods: getAge(), getIncomeByType(), getTotalAnnualIncome()

FamilyMember Model
├── User relationship + role (spouse, child, dependent, etc)
└── Spouse Permission support

Household Model
└── Aggregates household-level data (joint assets, expenses, etc)
```

Protection Module Models:

- `ProtectionProfile` - Income, expenses, dependents, human capital
- `LifeInsurancePolicy` - Sum assured, term, premium, type (term/whole-life)
- `CriticalIllnessPolicy` - Coverage details
- `IncomeProtectionPolicy` - Deferment, benefit period
- `DisabilityPolicy` - Disability cover
- `SicknessIllnessPolicy` - Sickness cover
- `ExpenditureProfile` - Monthly/annual living expenses

Savings Module Models:

- `SavingsAccount` - Institution, balance, type (regular/ISA), interest rate
- `SavingsGoal` - Target, deadline, current progress
- `ISAAllowanceTracking` - Annual tracking of £20k allowance

Investment Module Models:

- `InvestmentAccount` - Provider, balance, type (S&S ISA, GIA, SIPP, etc)
- `Holding` - Individual securities within account
- `InvestmentGoal` - Target, growth rate, rebalancing rules
- `RiskProfile` - Risk attitude questionnaire results

Retirement Module Models:

- `DCPension` - Pension name, value, contribution rate
- `DBPension` - Scheme name, accrual, pension in payment
- `StatePension` - Forecast amount, state pension age
- `RetirementProfile` - Retirement age, expenses, income needs

Estate Module Models (in Estate/ subdirectory):

```
Asset.php
├── Types: property, pension, investment, savings, business,
life_insurance, personal, other
├── Ownership: sole, joint_tenants, tenants_in_common, trust
└── IHT fields: is_iht_exempt, exemption_reason

Liability.php
├── Types: mortgage, secured_loan, personal_loan, credit_card, overdraft,
etc
└── IHT deductible

Gift.php (PETs/CLTs)
├── Gift type: pet, clt, exempt, small_gift, annual_exemption
├── Status: within_7_years, survived_7_years
└── Taper relief tracking

IHTProfile.php
├── Marital status, home ownership, NRB transferred
├── Charitable giving percentage
└── Spouse exemption fields

Trust.php
├── Types: bare, settlement, discretionary, interest_in_possession,
life_insurance
├── Assets: current_value, IHT-related fields
├── Periodic charge tracking

Will.php
├── Death scenario: user_only, both_die (second death)
├── Spouse as primary beneficiary (%)
└── Probate executor fields

Bequest.php (items in will)
├── Beneficiary, asset, percentage/amount
└── Contingency beneficiary

NetWorthStatement.php
└── Aggregated snapshot of assets - liabilities at point in time
```

Net Worth Module Models:

- `Property` - Address, value, ownership %, mortgages
- `Mortgage` - Lender, balance, rate, term, type
- `CashAccount` - Cash holdings
- `PersonalAccount` - P&L, cashflow, balance sheet items
- `BusinessInterest` - Business ownership, valuation
- `Chattel` - Personal items (art, jewelry, etc)

FRONTEND STRUCTURE

1. COMPONENTS (resources/js/components/ - 150+ .vue files)

Organization by Module:

```
components/
├── Admin/
│   ├── AdminDashboard.vue (user management, backup/restore)
│   ├── DatabaseBackup.vue (backup creation/restoration)
│   ├── TaxSettings.vue (tax configuration panel)
│   ├── UserFormModal.vue (create/edit users)
│   └── UserManagement.vue (user list, admin actions)
│
├── Auth/
│   └── ChangePasswordModal.vue
│
├── Common/
│   ├── ConfirmDialog.vue (delete confirmations)
│   └── ConfirmationModal.vue
│
├── Dashboard/
│   ├── AlertsPanel.vue (financial alerts)
│   ├── FinancialHealthScore.vue (overall health gauge)
│   ├── NetWorthOverviewCard.vue (summary card)
│   ├── NetWorthSummary.vue (breakdown)
│   ├── QuickActions.vue (shortcut tiles)
│   └── UKTaxesOverviewCard.vue & UKTaxesAllowancesCard.vue
│
├── Estate/ (45+ components for estate planning)
│   ├── IHTPlanning.vue (main IHT calculation component)
│   ├── WillPlanning.vue (will and bequest management)
│   ├── GiftingStrategy.vue (PET/CLT strategy with timeline)
│   ├── TrustPlanning.vue (trust lifecycle)
│   ├── LifePolicyStrategy.vue (life cover recommendations)
│   ├── NetWorth.vue (net worth details)
│   ├── AssetsLiabilities.vue (CRUD interface)
│   ├── AssetForm.vue & LiabilityForm.vue & GiftForm.vue
│   ├── CashFlow.vue (P&L statement)
│   ├── Recommendations.vue (estate recommendations)
```

```
│   ├── WhatIfScenarios.vue (scenario builder)
│   ├── NRBRNRBTracker.vue (spouse allowance tracking)
│   ├── SpouseExemptionNotice.vue (spouse calculation notice)
│   ├── SurvivingSpouseIHTPlanning.vue (second death scenarios)
│   ├── IHTMitigationStrategies.vue (strategy suggestions)
│   ├── LifeCoverRecommendations.vue (life policy gaps)
│   └── Charts:
│       ├── NetWorthWaterfallChart.vue
│       ├── IHTLiabilityGauge.vue
│       ├── GiftingTimelineChart.vue
│       ├── EstateProjectionComparison.vue
│       └── CashFlowProjectionChart.vue
│
├── Protection/ (20+ components)
│   ├── CurrentSituation.vue (policy listing)
│   ├── GapAnalysis.vue (coverage gaps)
│   ├── Recommendations.vue (coverage recommendations)
│   ├── WhatIfScenarios.vue (scenario modeling)
│   ├── PolicyFormModal.vue (CRUD for policies)
│   ├── PolicyCard.vue & PolicyDetails.vue
│   └── Charts:
│       ├── CoverageAdequacyGauge.vue
│       ├── CoverageGapChart.vue
│       ├── CoverageTimelineChart.vue
│       └── PremiumBreakdownChart.vue
│
├── Savings/ (15+ components)
│   ├── CurrentSituation.vue (account listing)
│   ├── EmergencyFund.vue (runway calculation)
│   ├── SavingsGoals.vue (goal tracking)
│   ├── ISAAllowanceTracker.vue (cross-module tracking)
│   ├── SaveAccountModal.vue & SaveGoalModal.vue
│   ├── Recommendations.vue
│   ├── WhatIfScenarios.vue
│   └── Charts:
│       ├── EmergencyFundGauge.vue
│       └── InterestRateComparisonChart.vue
│
├── Investment/ (20+ components)
│   ├── Accounts.vue (account listing)
│   ├── Holdings.vue & HoldingsTable.vue (security listing)
│   ├── Goals.vue (investment goal tracking)
│   ├── Performance.vue (performance analysis)
│   ├── PortfolioOverview.vue
│   ├── AccountForm.vue & HoldingForm.vue & GoalForm.vue
│   ├── MonteCarloResults.vue (simulation results)
│   ├── TaxFees.vue (tax efficiency, fee analysis)
│   ├── Recommendations.vue
│   ├── WhatIfScenarios.vue
│   └── Charts:
│       ├── AssetAllocationChart.vue
│       ├── GeographicAllocationMap.vue
│       └── PerformanceLineChart.vue
│
```

```
├── Retirement/ (15+ components)
│   ├── PensionInventory.vue (pension listing)
│   ├── DBPensionForm.vue & DCPensionForm.vue & StatePensionForm.vue
│   ├── Projections.vue (accumulation/drawdown)
│   ├── AnnualAllowanceTracker.vue (£60k limit + carry forward)
│   ├── DecumulationPlanning.vue (retirement income)
│   ├── ReadinessGauge.vue (retirement readiness %)
│   ├── Recommendations.vue
│   ├── WhatIfScenarios.vue
│   └── Charts:
│       ├── AccumulationChart.vue
│       ├── IncomeProjectionChart.vue
│       └── DrawdownSimulator.vue
│
├── NetWorth/ (12+ components)
│   ├── NetWorthOverview.vue (summary)
│   ├── Property/
│   │   ├── PropertyDetail.vue (single property view)
│   │   ├── PropertyForm.vue (CRUD for properties)
│   │   ├── PropertyFinancials.vue (rental income, tax)
│   │   ├── MortgageForm.vue (CRUD for mortgages)
│   │   ├── PropertyTaxCalculator.vue (SDLT, CGT, rental tax)
│   │   └── AmortizationScheduleView.vue
│   ├── BusinessInterestCard.vue & BusinessInterestsList.vue
│   ├── ChattelCard.vue & ChattelsList.vue
│   └── Charts:
│       ├── AssetAllocationDonut.vue
│       ├── AssetBreakdownBar.vue
│       └── NetWorthTrendChart.vue
│
├── Onboarding/ (10+ components)
│   ├── OnboardingWizard.vue (main flow)
│   ├── OnboardingStep.vue (step wrapper)
│   ├── FocusAreaSelection.vue (focus area picker)
│   ├── SkipConfirmationModal.vue
│   └── steps/
│       ├── PersonalInfoStep.vue
│       ├── FamilyInfoStep.vue
│       ├── IncomeStep.vue
│       ├── AssetsStep.vue
│       ├── LiabilitiesStep.vue
│       ├── ProtectionPoliciesStep.vue
│       ├── TrustInfoStep.vue
│       ├── WillInfoStep.vue
│       └── CompletionStep.vue
│
├── UserProfile/ (12+ components)
│   ├── PersonalInformation.vue (name, DOB, address)
│   ├── IncomeOccupation.vue (employment details)
│   ├── FamilyMembers.vue (spouse/family list)
│   ├── FamilyMemberFormModal.vue
│   ├── PersonalAccounts.vue (P&L, cashflow, balance sheet)
│   ├── SpouseDataSharing.vue (permission management)
│   ├── Settings.vue (user settings)
```

```
│   └── Views:
│       ├── AssetsOverview.vue
│       ├── LiabilitiesOverview.vue
│       ├── BalanceSheetView.vue
│       ├── CashflowView.vue
│       └── ProfitAndLossView.vue
│
├── Holistic/ (5+ components)
│   ├── ExecutiveSummary.vue (high-level plan)
│   ├── ModuleSummaries.vue (module snapshots)
│   ├── PrioritizedRecommendations.vue (unified recommendations)
│   ├── RiskAssessment.vue (overall risk)
│   ├── CashFlowAllocationChart.vue
│   └── NetWorthProjectionChart.vue
│
├── Trusts/ (3 components)
│   ├── TrustCard.vue
│   ├── TrustFormModal.vue
│   └── TrustsOverviewCard.vue
│
├── Shared/
│   └── ISAAllowanceSummary.vue (reused in multiple modules)
│
├── Actions/
│   └── RecommendationFilters.vue
│
├── UKTaxes/
│   └── CalculationsTab.vue (tax calculator)
│
└── Global/
    ├── Navbar.vue (main navigation)
    ├── Footer.vue
    └── App.vue (root layout)
```

Component Patterns:

- **Multi-word naming** (PascalCase filenames)
- **Tabs for module dashboards**: "Current Situation" | "Analysis" | "Recommendations" | "What-If" | "Details"
- **Modal forms for CRUD** (custom event: `@save`, NOT `@submit` to avoid double submissions)
- **Props with type validation** for data passing
- **Computed properties** for filtering/sorting
- **Methods** grouped by functionality
- **Lifecycle hooks** (mounted, watch) for data fetching

2. VIEWS (resources/js/views/ - 25 files)

Page-level components (one per major section):

```
views/
├── Dashboard.vue (main dashboard - calls DashboardController)
├── HolisticPlan.vue (cross-module holistic planning)
├── Login.vue (authentication)
├── Register.vue (user registration)
├── Version.vue (app version display)
├── UserProfile.vue (user profile management)
├── Settings.vue (global settings)
│
├── Estate/
│   └── EstateDashboard.vue (main estate page with tabs)
│
├── Protection/
│   └── ProtectionDashboard.vue
│
├── Savings/
│   └── SavingsDashboard.vue
│
├── Investment/
│   └── InvestmentDashboard.vue
│
├── Retirement/
│   ├── RetirementDashboard.vue (main retirement view)
│   ├── PensionInventory.vue (pension details tab)
│   ├── Projections.vue (accumulation/drawdown)
│   ├── ContributionsAllowances.vue (annual allowance tracking)
│   ├── DecumulationPlanning.vue (retirement income planning)
│   ├── RetirementReadiness.vue (readiness assessment)
│   ├── Recommendations.vue (retirement recommendations)
│   └── WhatIfScenarios.vue (scenario modeling)
│
├── NetWorth/
│   └── NetWorthDashboard.vue (net worth overview, properties, mortgages)
│
├── Trusts/
│   └── TrustsDashboard.vue (trust overview)
│
├── UKTaxes/
│   └── UKTaxesDashboard.vue (tax rules, allowances, calculator)
│
├── Admin/
│   └── AdminPanel.vue (user management, backups, tax settings)
│
├── Actions/
│   └── ActionsDashboard.vue (recommendations tracking)
│
└── Onboarding/
    └── OnboardingView.vue (onboarding wizard entry point)
```

3. VUEX STORE (resources/js/store/)

Structure:

```
store/
├── index.js (root store configuration)
│
└── modules/ (16 store modules)
    ├── auth.js (authentication state, token, user)
    ├── user.js (user profile state)
    ├── userProfile.js (extended profile info)
    ├── onboarding.js (wizard progress)
    ├── dashboard.js (dashboard data)
    ├── netWorth.js (net worth aggregation)
    │
    ├── protection.js (state: policies, analysis, recommendations)
    ├── savings.js (state: accounts, goals, ISA tracking)
    ├── investment.js (state: accounts, holdings, goals)
    ├── retirement.js (state: pensions, projections)
    ├── estate.js (state: assets, liabilities, gifts, trusts, IHT)
    │
    ├── holistic.js (cross-module holistic data)
    ├── recommendations.js (unified recommendations)
    ├── trusts.js (trust-specific state)
    └── spousePermission.js (permission status)
```

Store Module Pattern:

```
const state = { ... }                    // Reactive data
const getters = { ... }                  // Computed selectors
const actions = { async methods }        // Async operations (API calls)
const mutations = { sync updates }       // State mutations
```

Key Stores:

auth.js - Authentication

- State: token, user, loading, error
- Actions: login, register, logout, fetchUser
- Mutations: setToken, setUser, setLoading

estate.js - Estate Module

- State: assets, liabilities, gifts, trusts, ihtProfile, analysis, recommendations, secondDeathPlanning
- Getters: allAssets, totalAssets, totalLiabilities, netWorthValue, ihtLiability, assetsByType, etc.
- Actions: fetchEstateData, analyzeEstate, getRecommendations, runScenario, calculateIHT, etc.

investment.js - Investment Module

- State: accounts, holdings, goals, riskProfile, monteCarlo, analysis
- Getters: totalValue, allocation, performance, etc.
- Actions: fetchAccounts, analyzePortfolio, startMonteCarloSimulation, etc.

onboarding.js - Onboarding Wizard

- State: currentStep, completedSteps, focusArea, stepData
- Actions: fetchStepData, saveStepProgress, skipStep, completeOnboarding

4. JAVASCRIPT SERVICES (resources/js/services/ - 17 files)

Purpose: Encapsulate API communication for each module

```
services/
├── api.js (axios instance with auth interceptor)
│
├── authService.js
│   ├── login(email, password)
│   ├── register(userData)
│   ├── logout()
│   └── changePassword()
│
├── estateService.js
│   ├── getEstateData()
│   ├── analyzeEstate()
│   ├── calculateIHT()
│   ├── calculateSecondDeathIHTPlanning()
│   ├── getRecommendations()
│   ├── storeAsset(), updateAsset(), deleteAsset()
│   ├── storeLiability(), updateLiability(), deleteLiability()
│   ├── storeGift(), updateGift(), deleteGift()
│   ├── getTrusts(), createTrust(), updateTrust(), deleteTrust()
│   └── runScenario()
│
├── protectionService.js
│   ├── analyzeProtection()
│   ├── getRecommendations()
│   ├── storePolicy(), updatePolicy(), deletePolicy() [life, CI, IP, etc]
│   └── runScenario()
│
├── savingsService.js
│   ├── analyzeSavings()
│   ├── getRecommendations()
│   ├── storeAccount(), updateAccount(), deleteAccount()
│   ├── storeGoal(), updateGoal(), deleteGoal()
│   ├── getISAAllowance()
│   └── runScenario()
│
├── investmentService.js
│   ├── analyzePortfolio()
│   ├── getRecommendations()
│   ├── storeAccount(), updateAccount(), deleteAccount()
│   ├── storeHolding(), updateHolding(), deleteHolding()
│   ├── startMonteCarloSimulation()
│   ├── getMonteCarloResults()
│   └── storeRiskProfile()
```

```
│
├── retirementService.js
│   ├── analyzeRetirement()
│   ├── getRecommendations()
│   ├── storeDCPension(), storeDCPension(), storeStatePension()
│   ├── checkAnnualAllowance()
│   └── runScenario()
│
├── holisticService.js
│   ├── analyzeHolistic()
│   ├── getRecommendations()
│   ├── getCashFlowAnalysis()
│   └── markRecommendationDone/InProgress/dismiss()
│
├── dashboardService.js
│   ├── getDashboard()
│   ├── getFinancialHealthScore()
│   ├── getAlerts()
│   └── dismissAlert()
│
├── userProfileService.js
│   ├── getProfile()
│   ├── updatePersonalInfo()
│   ├── updateIncomeOccupation()
│   └── operations for family members, personal accounts
│
├── netWorthService.js
│   ├── getOverview()
│   ├── getBreakdown()
│   ├── getTrend()
│   └── refresh()
│
├── propertyService.js
│   ├── CRUD for properties
│   ├── calculateSDLT()
│   ├── calculateCGT()
│   └── calculateRentalIncomeTax()
│
├── mortgageService.js
│   ├── CRUD for mortgages
│   ├── calculatePayment()
│   └── amortizationSchedule()
│
├── onboardingService.js
│   ├── getOnboardingStatus()
│   ├── getSteps(), getStepData()
│   ├── saveStepProgress()
│   ├── completeOnboarding()
│   └── restartOnboarding()
│
├── spousePermissionService.js
│   ├── getStatus()
│   ├── requestPermission()
│   ├── acceptPermission(), rejectPermission()
```

```
│   └── revokePermission()
│
├── adminService.js
│   ├── getUsers(), createUser(), updateUser(), deleteUser()
│   ├── createBackup(), listBackups(), restoreBackup()
│   └── getDashboard()
│
├── taxSettingsService.js
│   ├── getCurrent()
│   ├── getAll()
│   ├── create(), update()
│   └── setActive()
│
└── utils/ (utility functions, helpers)
    └── (date formatting, number formatting, validation helpers)
```

Pattern: Each service wraps axios API calls:

```
async method() {
    const response = await api.get/post/put/delete('/endpoint');
    return response.data; // Extract data from response wrapper
}
```

MODULE BREAKDOWN

ESTATE PLANNING MODULE (Most Complex)

Frontend → Backend → Data Flow:

1. **Vue Components** (Estate dashboard)
   - IHTPlanning.vue → estateService.calculateIHT()
   - GiftingStrategy.vue → estateService.getPlannedGiftingStrategy()
   - WillPlanning.vue → willController.storeOrUpdateWill()
   - LifePolicyStrategy.vue → lifePolicyController.getLifePolicyStrategy()

2. **JavaScript Services** (estateService.js)

```
async calculateIHT(data) {
    const response = await api.post('/estate/calculate-iht', data);
    return response.data;
}
```

3. **API Routes** (routes/api.php)

```
Route::post('/estate/calculate-iht', [IHTController::class,
'calculateIHT']);
```

4. **Controllers** (IHTController.php)

```
public function calculateIHT(Request $request): JsonResponse {
    $this->ihtCalculator->calculateIHTLiability($assets, $ihtProfile,
...);
}
```

5. **Agents** (EstateAgent.php)
    - analyze() → coordinates multiple services
    - generateRecommendations() → uses analysis results
    - buildScenarios() → models what-if outcomes
6. **Services** (Estate/*)
    - IHTCalculator → NRB, RNRB, spouse exemption, gifts, trusts
    - NetWorthAnalyzer → asset aggregation, net worth calculation
    - GiftingStrategy → PET/CLT analysis, taper relief
    - SecondDeathIHTCalculator → married couple scenarios
    - LifePolicyStrategyService → life cover recommendations
    - FutureValueCalculator → growth projections
    - ActuarialLifeTableService → mortality data, life expectancy
7. **Models** (Estate/*)
    - Asset, Liability, Gift, Trust, Will, IHTProfile, Bequest, etc.
8. **Database** (MySQL)
    - Assets table (thousands of records, indexed by user_id)
    - Liabilities table
    - Gifts table
    - Trusts table
    - IHT profiles
    - Wills & Bequests

PROTECTION MODULE (Second Most Complex)

Key Data Flow: Vue → protectionService → ProtectionController → ProtectionAgent → Services → Models

Protection Agent Orchestration:

```
ProtectionAgent.analyze()
├── CoverageGapAnalyzer.calculateProtectionNeeds()
```

```
├── CoverageGapAnalyzer.calculateTotalCoverage()
├── CoverageGapAnalyzer.calculateCoverageGap()
├── AdequacyScorer.calculateAdequacyScore()
├── RecommendationEngine.generateRecommendations()
└── ScenarioBuilder.modelDeathScenario() / modelCriticalIllnessScenario()
```

Key Components:

- PolicyCard.vue - Display individual policies
- CurrentSituation.vue - List all policies
- GapAnalysis.vue - Coverage gap visualization
- CoverageAdequacyGauge.vue - Visual gauge of adequacy %

SAVINGS MODULE

Key Data Flow: Vue → savingsService → SavingsController → SavingsAgent → Services → Models

Cross-Module Integration: ISA Tracker service aggregates ISA usage from both Savings (Cash ISA) and Investment (S&S ISA) modules

Key Components:

- SaveAccountModal.vue - Add/edit savings accounts
- EmergencyFund.vue - Emergency fund runway calculation
- ISAAllowanceTracker.vue - £20k total allowance tracking
- SavingsGoals.vue - Goal progress tracking

INVESTMENT MODULE

Key Data Flow: Vue → investmentService → InvestmentController → InvestmentAgent → Services → Models

Background Jobs: Monte Carlo simulations run as queued jobs

Key Components:

- AccountForm.vue - Investment account CRUD
- Holdings.vue - Security listing
- MonteCarloResults.vue - Simulation visualization
- TaxFees.vue - Tax efficiency and fee analysis

RETIREMENT MODULE

Key Data Flow: Vue → retirementService → RetirementController → RetirementAgent → Services → Models

Key Components:

- PensionInventory.vue - DC/DB/State pension listing
- AnnualAllowanceTracker.vue - £60k annual limit + 3-year carry forward
- Projections.vue - Accumulation/decumulation charts
- ReadinessGauge.vue - Retirement readiness percentage

PROTECTION, SAVINGS, INVESTMENT, RETIREMENT

Shared Characteristics:

- Each has a dashboard view with tabs
- Each calls an Agent for analysis
- Each returns canned recommendations
- Each supports what-if scenario building
- Each integrates with cross-module dashboard

DATA FLOW PATTERNS

COMPLETE REQUEST FLOW (Example: Calculate IHT)

```
1. Vue Component (IHTPlanning.vue)
   ↓ User clicks "Calculate IHT"
2. Component calls: estateService.calculateIHT(profileData)
   ↓
3. Service (estateService.js)
   ↓ Makes HTTP request:
4. API Call: POST /api/estate/calculate-iht
   ↓ Bearer token in Authorization header
5. Controller (IHTController::calculateIHT)
   ├─ Validate input via Form Request
   ├─ Get authenticated user
   └─ Call service/agent methods
   ↓
6. Services Layer
   ├─ IHTCalculator.calculateIHTLiability()
   ├─ NetWorthAnalyzer.generateSummary()
   ├─ FutureValueCalculator.getLifeExpectancy()
   └─ Each service queries Models via Eloquent
   ↓
7. Models/Database
   ├─ Asset::where('user_id', $userId)->get()
   ├─ Liability::where('user_id', $userId)->get()
   ├─ Gift::where('user_id', $userId)->get()
   ├─ InvestmentAccount::where('user_id', $userId)->get()
   ├─ Property::where('user_id', $userId)->get()
   └─ SavingsAccount::where('user_id', $userId)->get()
   ↓
8. Calculation in Service
   ├─ Aggregate assets from all modules
   ├─ Deduct liabilities
   ├─ Apply spouse exemption (if married)
   ├─ Calculate NRB/RNRB use
   ├─ Factor in gifts and trusts
   └─ Calculate IHT at 40%
   ↓
9. Cache Results (if configured)
   └─ Cache::remember("iht_calculation_{$userId}", 3600, ...)
```

```
   ↓
10. Return JSON Response
    └─ {success: true, data: {iht_liability: £X, breakdown: {...}}}
    ↓
11. JavaScript Service Returns data.data
    ↓
12. Vue Component Receives data
    ├─ Commits to Vuex store: estate.mutations.setIHTCalculation()
    ├─ Updates component data
    └─ Re-renders template
    ↓
13. Template Renders
    ├─ Displays IHT liability in gauge
    ├─ Shows breakdown table
    ├─ Displays recommendations
    └─ Charts update with ApexCharts
```

CROSS-MODULE INTEGRATION FLOW (Example: Estate Gathering Investment Accounts)

```
EstateController.index()
├─ Get manual Estate Assets
├─ Get InvestmentAccounts from Investment module
│  └─ InvestmentAccount::where('user_id', $userId)->get()
├─ Get Properties from NetWorth module
│  └─ Property::where('user_id', $userId)->get()
├─ Get SavingsAccounts from Savings module
│  └─ SavingsAccount::where('user_id', $userId)->get()
└─ Return aggregated data
   └─ {assets: [...], investment_accounts: [...], properties: [...], ...}

Frontend (IHTPlanning.vue)
├─ Calls estateService.calculateIHT()
└─ Backend IHTCalculator automatically includes:
   ├─ Manual assets from estate_assets table
   ├─ Investment accounts (S&S ISAs, GIAs, etc)
   ├─ Properties (with ownership %)
   └─ Savings accounts (even though ISAs are IHT taxable, not exempt)
```

VUEX STORE FLOW

```
Vue Component
├─ Calls: this.$store.dispatch('estate/analyzeEstate')
│  ↓
│  Actions (async)
│  ├─ API call via service
│  ├─ Process response
│  └─ Commit mutations
│
└─ Commits: this.$store.commit('estate/setIHTCalculation', data)
```

```
   ↓
   Mutations (sync)
   ├─ Update state.analysis
   ├─ Update state.recommendations
   └─ Update state.secondDeathPlanning
   ↓
   Getters (computed)
   ├─ Read: this.$store.getters['estate/ihtLiability']
   ├─ Read: this.$store.getters['estate/allAssets']
   └─ Component watches for changes
   ↓
   Component Re-renders
   └─ Using computed: return this.$store.getters[...] or
this.$store.state[...]
```

CACHING STRATEGY

```
Frontend (Vue)
├─ Component data (reactive)
└─ Vuex store (persists until browser refresh)

Backend (Laravel)
├─ Memcached (primary)
│  ├─ Tax config: 3600s (1h)
│  ├─ Analysis results: 3600s (1h)
│  └─ Monte Carlo: 86400s (24h)
│
└─ Database Queries
   ├─ Indexed by user_id
   └─ Eager loading via Eloquent relations

Controller Cache Invalidation
├─ Cache::forget("estate_analysis_{$userId}") after updates
├─ Cache::flush() in admin for tax config updates
└─ Automatic TTL expiration
```

CROSS-MODULE INTEGRATION

ISA ALLOWANCE TRACKING (Savings + Investment)

Challenge: UK tax rule: £20,000 total ISA allowance across all accounts

Solution: Centralized ISATracker service

```
Frontend
├─ Savings Dashboard: Shows Cash ISA contribution (e.g., £8,000)
├─ Investment Dashboard: Shows S&S ISA contribution (e.g., £12,000)
└─ Both call: savingsService.getISAAllowance('2025/26')
```

```
Backend
├─ ISATracker.calculateAllowanceUsage($userId, $taxYear)
│  ├─ Sum SavingsAccount.current_balance where account_type = 'isa'
│  ├─ Sum InvestmentAccount.current_value where account_type = 's&s_isa'
│  ├─ Calculate total: £20,000
│  ├─ Calculate remaining: £20,000 - used
│  └─ Flag warnings if exceeds
│
└─ Database
   ├─ ISAAllowanceTracking table (audit trail)
   └─ Tracks by user_id + tax_year
```

NET WORTH AGGREGATION (All Modules)

Challenge: Net worth must include assets and liabilities from all modules

Solution: NetWorthService + CrossModuleAssetAggregator

```
NetWorthService.generateNetWorth($userId)
├─ Assets
│  ├─ Estate Assets (manual entry)
│  ├─ Investment Accounts
│  ├─ Savings Accounts
│  ├─ Properties
│  ├─ Pensions (estimated value)
│  ├─ Business Interests
│  └─ Chattels
│
├─ Liabilities
│  ├─ Mortgages
│  ├─ Personal Loans
│  ├─ Credit Cards
│  ├─ Other Debt
│  └─ Loans against trusts
│
└─ Net Worth = Total Assets - Total Liabilities

Trend Tracking
├─ NetWorthStatement table (historical snapshots)
└─ Calculated monthly/quarterly for trend analysis
```

SECOND DEATH IHT PLANNING (Estate Module for Married Couples)

Challenge: Calculate optimal gifting strategy when both spouses die

Solution: SecondDeathIHTCalculator + ActuarialLifeTableService

```
Frontend (SurvivingSpouseIHTPlanning.vue)
├─ User enters: spouse details, assets, gifting timeline
└─ Calls: estateService.calculateSecondDeathIHTPlanning()

Backend (SecondDeathIHTCalculator)
├─ Spouse 1 death scenario
│  ├─ Surviving spouse gets spouse exemption (unlimited)
│  └─ Spouse 2 later inherits: full spouse NRB + original NRB = £650k
│
├─ Spouse 2 death scenario
│  ├─ Estate includes: remaining assets + inherited assets
│  └─ Available NRB: original £325k + transferred spouse NRB £325k = £650k
│
├─ Gifting Strategy Optimizer
│  ├─ Calculate optimal annual gifting (£3k + surplus income exempt)
│  ├─ Model PET survival probabilities (actuarial tables)
│  ├─ Project estate growth (4.5% annually)
│  └─ Recommend life cover amount (to cover IHT on second death)
│
└─ Actuarial Life Tables
   ├─ Age + gender → life expectancy
   ├─ Probability of surviving N years
   └─ Used for gifting timeline modeling
```

COORDINATING AGENT (Holistic Planning)

Responsibility: Coordinate analysis across all modules

```
CoordinatingAgent.analyze($userId)
├─ Call each module Agent.analyze():
│  ├─ ProtectionAgent.analyze()
│  ├─ SavingsAgent.analyze()
│  ├─ InvestmentAgent.analyze()
│  ├─ RetirementAgent.analyze()
│  └─ EstateAgent.analyze()
│
├─ Aggregate recommendations from all agents
│
├─ Identify conflicts:
│  └─ e.g., "Invest £10k ISA" vs "Pay off credit card" → credit card wins
│
├─ Prioritize across modules:
│  └─ Priority matrix: impact × urgency
│
├─ Generate holistic cash flow:
│  └─ Monthly: Income - Expenses - Debt servicing - Recommended savings
│
└─ Return unified plan with module synergies highlighted
```

KEY TECHNOLOGIES & PATTERNS

BACKEND TECH STACK

- **Framework**: Laravel 10.x (PSR-12 coding standards)
- **Database**: MySQL 8.0+ (InnoDB, `DECIMAL(15,2)` for currency)
- **Cache**: Memcached (for expensive calculations)
- **Auth**: Laravel Sanctum (token-based API auth)
- **Validation**: Form Requests (custom validation classes)
- **Jobs**: Laravel Queue (database-backed for Monte Carlo)
- **Testing**: Pest (unit & feature tests)

FRONTEND TECH STACK

- **Framework**: Vue.js 3 (Composition API support)
- **State Management**: Vuex 4
- **Build Tool**: Vite (HMR enabled)
- **CSS**: Tailwind CSS (utility-first)
- **Charts**: ApexCharts (interactive, responsive)
- **HTTP Client**: Axios (with interceptors for auth)

ARCHITECTURAL PATTERNS

1. Agent Pattern

- Each module has an agent orchestrating analysis
- Standardized interface: `analyze()`, `generateRecommendations()`, `buildScenarios()`
- Cache at agent level for performance

2. Service Layer Pattern

- Services encapsulate business logic
- Split by domain (Protection, Savings, etc.)
- Services call Models, not controllers
- Testable and reusable

3. Model Layer Pattern

- Eloquent relationships define data structure
- Type casting for safety (`'current_value' => 'float'`)
- Local scopes for common queries (`scope whereActive()`)
- No business logic in models (kept in services)

4. Controller Pattern

- Thin controllers (validation + service coordination)
- Request validation via Form Request classes
- Return standardized JSON responses
- Cache invalidation after writes

5. Store Pattern (Vuex)

- State: Single source of truth
- Getters: Computed selectors (memoized)
- Actions: Async operations (API calls)
- Mutations: Synchronous state updates
- Clear naming: `module/ACTION_NAME`

6. Component Pattern

- Smart components (pages, coordinate logic)
- Dumb components (UI presentation)
- Props for input, events for output
- Avoid prop drilling (use store for shared state)
- Computed properties for reactive data

7. Service Pattern (JavaScript)

- Wrapper around axios API calls
- One service per module
- Consistent method naming
- Error handling via interceptors

CONFIGURATION MANAGEMENT

Centralized Tax Rules (`config/uk_tax_config.php`):

```
'inheritance_tax' => [
    'nil_rate_band' => 325000,
    'residence_nil_rate_band' => 175000,
    'rate' => 0.40,
    'annual_exemption' => 3000,
    'small_gifts_exemption' => 250,
    'pet_taper_relief' => [...3-7 year bands...],
],
'income_tax' => [
    'personal_allowance' => 12570,
    'basic_rate_band' => [12571, 50270],
    'higher_rate_band' => [50271, 125140],
    'additional_rate' => 0.45,
    ...
],
'national_insurance' => [...],
'pensions' => [
    'annual_allowance' => 60000,
    'mpaa_threshold' => 10000,
    'carry_forward_years' => 3,
],
'savings' => [
    'isa_allowance' => 20000,
    'lisa_contribution_limit' => 4000,
]
```

VALIDATION PATTERNS

Form Requests (`app/Http/Requests/`):

```
class StorePropertyRequest extends FormRequest {
    public function authorize() { return true; }

    public function rules() {
        return [
            'address_line_1' => 'required|string|max:255',
            'current_value' => 'required|numeric|min:0',
            'ownership_type' => 'required|in:individual,joint,trust',
            ...
        ];
    }
}
```

TESTING PATTERNS

Pest Test Structure:

```
describe('Estate Agent', function () {
    it('calculates iht liability correctly', function () {
        $user = User::factory()->create();
        $agent = new EstateAgent(...);

        $result = $agent->analyze($user->id);

        expect($result['iht_liability'])->toBe(/*...*/);
    });
});
```

ERROR HANDLING

Backend:

- Try-catch in controllers
- Standardized error responses
- Validation errors: 422 with field-level errors
- Auth errors: 401 with redirect to login
- Not found: 404 with message

Frontend:

- API interceptor catches all errors
- localStorage cleared on 401
- User redirected to login
- Toast/snackbar for error messages

- Form validation before submission

SECURITY PATTERNS

Authentication:

- Laravel Sanctum tokens (no sessions)
- Bearer token in Authorization header
- Token stored in localStorage
- Token passed to all authenticated requests

Authorization:

- Middleware `auth:sanctum` protects routes
- User ID from `$request->user()`
- Always filter by `user_id` in queries
- Spouse permissions for cross-spouse access

Data Validation:

- Form Requests validate all inputs
- Type hints on service methods
- Database constraints (foreign keys, NOT NULL)

Secrets:

- `.env` file for database, API keys
- Not committed to git
- Admin password in tinker only

KEY FILES REFERENCE

CRITICAL BACKEND FILES

File	Purpose	LOC
`app/Agents/EstateAgent.php`	Estate analysis orchestration	200+
`app/Services/Estate/IHTCalculator.php`	IHT liability calculation	400+
`app/Services/Estate/NetWorthAnalyzer.php`	Net worth aggregation	300+
`app/Services/Estate/CashFlowProjector.php`	Cash flow statements	250+
`app/Http/Controllers/Api/EstateController.php`	Estate CRUD & routes	400+
`config/uk_tax_config.php`	Centralized tax rules	200+
`routes/api.php`	All API route definitions	400+

CRITICAL FRONTEND FILES

File	Purpose
`resources/js/components/Estate/IHTPlanning.vue`	Main IHT calculation UI
`resources/js/views/Estate/EstateDashboard.vue`	Estate tab-based dashboard
`resources/js/store/modules/estate.js`	Estate state management
`resources/js/services/estateService.js`	Estate API service
`resources/js/components/Holistic/ExecutiveSummary.vue`	Holistic planning summary

SUMMARY STATISTICS

- **Total PHP Files**: ~400
- **Total Vue Components**: ~150
- **Total JavaScript Services**: 17
- **Total Vuex Stores**: 16
- **Database Models**: 39
- **Agent Classes**: 7 (1 base + 6 module-specific)
- **Service Classes**: 50+
- **API Controllers**: 25+
- **Form Request Classes**: 30+
- **Test Files**: 50+
- **Database Tables**: 40+
- **Lines of Backend Code**: 4,800+
- **Lines of Frontend Code**: 15,000+

QUICK START FOR DEVELOPERS

Adding a New Feature to Estate Module

1. **Define Model** (app/Models/Estate/NewFeature.php)
    - Add fillable array
    - Define relationships
    - Add type casting
2. **Create Database Migration**
    - Run: `php artisan make:migration create_new_features_table`
    - Define schema with proper types
    - Add indexes for user_id and foreign keys
3. **Add Service** (app/Services/Estate/NewFeatureService.php)
    - Accept $userId as first parameter
    - Use dependency injection for other services
    - Cache expensive operations

4. **Update Agent** (app/Agents/EstateAgent.php)
    - Call new service in analyze() method
    - Include results in return array
5. **Create Controller Endpoint** (app/Http/Controllers/Api/EstateController.php)
    - Add method to EstateController
    - Validate input with Form Request
    - Inject and call agent/service
    - Return JSON response
6. **Create Form Request** (app/Http/Requests/Estate/StoreNewFeatureRequest.php)
    - Define rules()
    - Define authorize()
7. **Add Route** (routes/api.php)
    - Add under `estate` prefix group
    - Use FormRequest for validation
8. **Create JavaScript Service** (resources/js/services/estateService.js)
    - Add async method
    - Make API call
    - Return response.data
9. **Update Vuex Store** (resources/js/store/modules/estate.js)
    - Add state field
    - Add getter
    - Add action (calls JS service)
    - Add mutation (updates state)
10. **Create Vue Component** (resources/js/components/Estate/NewFeature.vue)
    - Import service
    - Define props, data, computed, methods
    - Call store action on mount
    - Render with Tailwind + ApexCharts
11. **Integrate into Dashboard** (resources/js/views/Estate/EstateDashboard.vue)
    - Add tab or section
    - Import component
    - Include in template