Money Flow

Comprehensive Recurring Payment Management

A modern, AI-powered application for tracking all types of recurring financial obligations with natural language commands and intelligent insights.

Version	2.0 (Money Flow Complete)
Last Updated	December 13, 2025
Status	Production Ready
Tests	400+ passing

Executive Summary

Project Overview

Money Flow (formerly Subscription Tracker) is a comprehensive recurring payment management application featuring an agentic interface that allows natural language commands to manage all types of recurring payments. The system combines a modern Next.js frontend with a FastAPI backend, PostgreSQL database, Redis caching, and Qdrant vector database for RAG capabilities.

Key Achievements

- Multi-container Docker setup with 5 services (PostgreSQL, FastAPI, Next.js, Redis, Qdrant)
- Agentic interface with Claude Haiku 4.5 and XML-based prompting
- Support for 9 payment types (subscriptions, housing, utilities, debts, savings, etc.)
- RAG implementation complete with semantic search and conversation context
- Modern glassmorphism UI with Tailwind CSS v4 and Framer Motion animations
- Payment card tracking with funding chain support
- Import/Export functionality (JSON & CSV v2.0 format)
- 400+ automated tests with comprehensive coverage

Business Value

Money Flow provides users with a unified view of all recurring financial obligations, enabling better financial planning and management. The natural language interface reduces friction in data entry, while AI-powered insights help identify spending patterns and optimization opportunities.

Technology Stack

Backend Technologies

Technology	Version	Purpose
Python	3.11+	Core backend language
FastAPI	Latest	REST API framework
SQLAlchemy	2.0	Async ORM with PostgreSQL
Pydantic	v2	Data validation and schemas
PostgreSQL	15	Primary database
Redis	Latest	Caching and session storage
Qdrant	1.7+	Vector database for RAG

Frontend Technologies

Technology	Version	Purpose
Next.js	16.0.5	React framework with Turbopack
React	19.2.0	UI component library
TypeScript	Strict	Type-safe JavaScript
Tailwind CSS	4.1.17	CSS-first utility classes
Framer Motion	12.23.24	Animation library
React Query	5.90.11	Server state management

AI & Machine Learning

Technology	Model/Version	Purpose
Claude API	Haiku 4.5	Intent classification & NL parsing

Sentence Transformers	all-MiniLM-L6-v2	Text embeddings (384 dim)
Qdrant	1.7+	Vector similarity search

System Architecture

6-Layer Architecture

The application follows Domain-Driven Design principles with a clean separation of concerns across six distinct layers:

Layer	Technology	Responsibility
Presentation	Next.js + React	User interface, client state
API Gateway	FastAPI	Request handling, validation
Business Logic	Python Services	Domain logic, transactions
Agentic	Claude AI + RAG	NL parsing, context, insights
Data Access	SQLAlchemy 2.0	ORM, queries, relationships
Database	PostgreSQL + Qdrant	Data persistence, vectors

Docker Services

Service	Container	Port	Purpose
Database	subscription-db	5433	PostgreSQL data storage
Backend	subscription-backend	8001	FastAPI application
Frontend	subscription-frontend	3001	Next.js web app
Cache	subscription-redis	6379	Redis caching
Vectors	subscription-qdrant	6333	Qdrant vector DB

Data Flow

1. User interacts via web UI or natural language chat
2. Frontend sends requests to Next.js API routes (proxied to backend)
3. FastAPI validates requests with Pydantic schemas
4. For NL commands: Parser + Claude AI extracts intent and entities
5. RAG service provides conversation context and semantic search

6. Service layer executes business logic with database
7. Response serialized and returned to frontend

Payment Types & Features

Supported Payment Types

Money Flow supports 9 distinct payment types, each with specialized tracking fields:

Type	Examples	Special Features
SUBSCRIPTION	Netflix, Spotify, Claude AI	Standard recurring tracking
HOUSING	Rent, mortgage	Auto-classified from keywords
UTILITY	Electric, water, council tax	Auto-classified from keywords
PROFESSIONAL	Therapist, coach, trainer	Service provider tracking
INSURANCE	Health, AppleCare, vehicle	Policy management
DEBT	Credit cards, loans, personal	total_owed, remaining_balance, creditor
SAVINGS	Goals, regular transfers	target_amount, current_saved, recipient
TRANSFER	Family support, gifts	recipient tracking
ONE_TIME	Legal fees, single purchases	Non-recurring with end_date

Payment Card System

Track which card pays for each subscription with the payment card system:

- Card types: Debit, Credit, Prepaid, Bank Account
- Visual card display with brand colors and logos
- Balance tracking per card (this month / next month)
- Funding chain support (e.g., PayPal funded by Monzo)
- Unassigned payment tracking and warnings

Multi-Currency Support

Currency	Symbol	Flag	Status
GBP	£	■■	Default
EUR	€	■■	Supported

USD	$	■■	Supported
UAH	■	■■	Supported

AI Agent & Natural Language Interface

Agentic Architecture

The AI agent uses Claude Haiku 4.5 for fast, cost-effective intent classification and entity extraction. The system implements dual-mode parsing with AI as primary and regex patterns as fallback for reliability.

Agent Components

Component	File	Purpose
CommandParser	src/agent/parser.py	NL → intent + entities
AgentExecutor	src/agent/executor.py	Intent → service calls
PromptLoader	src/agent/prompt_loader.py	XML prompt management
ConversationalAgent	src/agent/conversational_agent.py	Tool-use based agent

Example Commands

- "Add Netflix for £15.99 monthly" → Creates subscription
- "Add rent payment £1137.50 monthly" → Creates housing payment
- "Add debt to John £500, paying £50 monthly" → Creates debt with creditor
- "Add savings goal £10000 for holiday" → Creates savings with target
- "I paid £200 off my credit card" → Updates debt balance
- "How much am I spending per month?" → Returns summary
- "What's due this week?" → Lists upcoming payments
- "Show my total debt" → Aggregates debt balances

XML-Based Prompting

Prompts are organized in structured XML files for maintainability:

- system.xml - System role and capabilities
- command_patterns.xml - Intent patterns with examples
- currency.xml - Currency detection configuration
- response_templates.xml - Response format templates

RAG Implementation

What is RAG?

RAG (Retrieval-Augmented Generation) enhances the AI agent with memory and context awareness. The agent can remember conversations, search semantically, and provide intelligent insights based on historical data.

RAG Services

Service	Purpose	Status
EmbeddingService	Generate text embeddings (384-dim)	■ Complete
VectorStore	Qdrant CRUD + similarity search	■ Complete
RAGService	Context retrieval, reference resolution	■ Complete
ConversationService	Session management, history	■ Complete
InsightsService	Spending patterns, recommendations	■ Complete
HistoricalQueryService	Temporal parsing, date queries	■ Complete
CacheService	Redis embedding cache	■ Complete
RAGAnalyticsService	Query monitoring, metrics	■ Complete

Key RAG Features

- Reference Resolution: 'Cancel it' → 'Cancel Netflix' (from context)
- Semantic Note Search: Find subscriptions by meaning, not just keywords
- Conversation Memory: Multi-turn conversations with session tracking
- Hybrid Search: Combines semantic similarity with keyword boosting
- Spending Insights: Trend analysis, category breakdown, predictions
- Historical Queries: 'What did I add last month?' with temporal parsing
- Embedding Cache: 60%+ cache hit rate for performance
- Analytics Dashboard: Query latency tracking, health monitoring

Architecture Decisions

- Vector DB: Qdrant (self-hosted, Docker-native, excellent filtering)

- Embedding Model: all-MiniLM-L6-v2 (local, 50ms inference, 80MB)
- Caching: Redis with TTL-based expiration
- Context: Hybrid approach (recent turns + semantic search)
- Data Isolation: User-level filtering on all queries

Frontend & User Interface

Modern Design System

The frontend uses cutting-edge 2025 CSS features with Tailwind CSS v4, featuring a glassmorphism design language with OKLCH color space for perceptually uniform colors.

- Glassmorphism cards with backdrop blur and subtle borders
- OKLCH color definitions for consistent lightness perception
- Framer Motion animations with spring physics
- Scroll-driven animations and CSS anchor positioning
- Container style queries and :has() parent selectors
- Service icon library with 70+ popular subscription icons
- Brand colors for recognized services (Netflix, Spotify, etc.)

Key Components

Component	Purpose
Header	Navigation, branding with gradient glow
StatsPanel	Spending summary, debt/savings progress
SubscriptionList	Payment list with service icons, filtering
AddSubscriptionModal	Smart form with service suggestions
PaymentCalendar	Calendar view of upcoming payments
CardsDashboard	Payment card management, balance tracking
AgentChat	Natural language interface with markdown
ImportExportModal	JSON/CSV import and export

Service Icon Library

The application includes icons for 70+ popular services across categories:

- Streaming: Netflix, Disney+, Hulu, HBO Max, Apple TV+, YouTube
- Music: Spotify, Apple Music, Tidal, Deezer, Amazon Music
- Gaming: Xbox Game Pass, PlayStation Plus, Steam, GeForce Now

- Productivity: Microsoft 365, Google One, Notion, Slack, Figma
- Development: GitHub, GitLab, JetBrains, Vercel, AWS, Heroku
- AI Tools: ChatGPT Plus, Claude Pro, Midjourney, Grammarly

API Endpoints

Subscriptions API

Method	Endpoint	Description
GET	/api/subscriptions	List all (with payment_type filter)
GET	/api/subscriptions/{id}	Get single subscription
POST	/api/subscriptions	Create subscription
PUT	/api/subscriptions/{id}	Update subscription
DELETE	/api/subscriptions/{id}	Delete subscription
GET	/api/subscriptions/summary	Spending summary by period
GET	/api/subscriptions/upcoming	Upcoming payments

Import/Export API

Method	Endpoint	Description
GET	/api/subscriptions/export/json	Export as JSON v2.0
GET	/api/subscriptions/export/csv	Export as CSV v2.0
POST	/api/subscriptions/import/json	Import from JSON
POST	/api/subscriptions/import/csv	Import from CSV

Cards & Analytics API

Method	Endpoint	Description
GET	/api/cards	List payment cards
POST	/api/cards	Create payment card
GET	/api/cards/balance-summary	Card balance summary
GET	/api/analytics/daily	Daily RAG metrics

GET	/api/analytics/health	System health check
GET	/api/insights/	Complete spending insights
POST	/api/insights/historical	Historical query with parsing
POST	/api/search/notes	Semantic note search
POST	/api/agent/execute	Execute NL command

Project Metrics & Status

Development Metrics

Metric	Value	Notes
Total Tests	400+	Unit + Integration
Python Files	51+	Backend codebase
TypeScript Files	18+	Frontend codebase
API Endpoints	45+	REST API coverage
Database Migrations	8	Alembic managed
Payment Types	9	Full Money Flow coverage
Service Icons	70+	Popular subscriptions
Lines of Code	~7,500+	Python backend

Completion Status

Feature	Status	Phase
Core CRUD Operations	■ Complete	Initial
AI Agent (Claude Haiku)	■ Complete	Initial
Multi-Currency Support	■ Complete	Initial
Money Flow Refactor	■ Complete	Phase 1-3
RAG Implementation	■ Complete	Phase 1-4
Payment Cards System	■ Complete	Enhancement
Import/Export v2.0	■ Complete	Enhancement
Modern UI (Tailwind v4)	■ Complete	Polish
GCP Deployment	■ Planned	Future
User Authentication	■ Planned	Future

Access URLs (Local Development)

Service	URL
Frontend	http://localhost:3001
Backend API	http://localhost:8001
API Documentation	http://localhost:8001/docs
Database	localhost:5433
Qdrant Dashboard	http://localhost:6333/dashboard

Generated on December 13, 2025 at 19:13 | Money Flow v2.0