Project Specification: TaxPal RAG Chatbot

**Status**

**MVP Planning Phase**

**Audience**

* Accountants, Tax Professionals, Tax Students, Tax & Finance Departments in businesses. General Public (secondary)

# 1. Project Overview

* **Purpose:** Provide a subscription-based chatbot that uses Retrieval-Augmented Generation (RAG) to deliver accurate information on Irish tax legislation.
* **Brand Name / Placeholder:** TaxPal
* **Core Objective:** Offer streamlined answers and references to the relevant sections of Irish tax legislation, with disclaimers that it is not formal legal advice.

# 2. Core Stack & Components

## Frontend

* **Framework:** React (Next.js)
* **Styling:** Chakra UI
* **Auth:** Supabase Auth with OAuth (Google)
* **Hosting:** Vercel (auto-deploy via Git integration)
* **Key Pages:**
  1. Landing / Home
  2. Sign Up / Log In
  3. Main Chat (chatbot + citations)
  4. Profile & Subscription
  5. Chat History
  6. Admin Dashboard

## Backend

### Language

* Python

### Framework

FastAPI

### Hosting

Fly.io (with auto-scaling via Gunicorn/Uvicorn)

### CI/CD

GitHub Actions (test, build, deploy, coverage enforcement)

#### Deployment Pipeline Details

CI/CD Workflow:

1. Pull request triggers tests and linting
2. Merge to main triggers staging deployment
3. Manual promotion from staging to production
4. Database migrations run automatically before deployment

Environments:

* Development: Local development environments
* Staging: Mirrors production with test data
* Production: Live environment with real user data

Monitoring:

* Application performance monitoring via Datadog or similar
* Error tracking via Sentry
* Usage metrics dashboard for business stakeholders
* Automated alerts for high error rates or performance issues

### Key Services

#### RAG pipeline

* Embedding Model: gemini-embedding-exp-03-07
* Vector Store: Pinecone
* Completion Model: Gemini Flash 2.0

#### Subscription Management

Stripe + Webhooks

#### Auth Integration

* Leverage Supabase Auth directly via client SDK
* Custom backend auth endpoints only for specialized needs not covered by Supabase SDK
* Token validation middleware for protected routes

#### Usage Tracking

Token-based, background job resets

#### Chat History & Logging

#### CLI Document Ingestion (admin-managed)

## Supabase

### Auth

* Email/password + Google OAuth

### Token-Based Protection

* for all private endpoints

### Database

Users, usage counters, subscriptions, chat logs, feedback

### Scheduled Tasks

Usage resets (weekly/monthly)

### Technical Specifications

#### Database Schema Details

<https://d.docs.live.net/5162816083b4594e/Project%20Support%20Material/Python%20Projects/Tax_Rag/documentation/Specifications/database-schema.mermaid>

## Infrastructure

### Environment

Dev, Staging, Prod

### Secrets Management

* Vercel (frontend)
* Fly.io (backend)

### Testing

* Backend: Pytest (unit + integration)
* Frontend: Jest + Playwright (E2E)
* Initial coverage thresholds enforced in CI: 50% for MVP, increasing to 80% post-MVP
* Focus testing on critical RAG pipeline components and authentication flows

#### Comprehensive Testing Strategy

##### Unit Testing:

* RAG pipeline components: chunking, embedding, retrieval, generation
* Authentication flows
* Subscription management
* Rate limiting and quota enforcement

##### Integration Testing:

* End-to-end chat flow with mock LLM responses
* Authentication with Supabase
* Subscription management with Stripe
* Document ingestion pipeline

##### Performance Testing:

* Response time under various loads (target < 3s for chat responses)
* Concurrent user simulation (start with 100 concurrent users)
* Token usage tracking accuracy
* Rate limiting effectiveness

##### User Acceptance Testing:

* Testing with sample tax queries from different categories
* Verification of citation accuracy
* Subscription upgrade flow
* Mobile responsiveness

# 3. RAG Implementation

* **Chunking Strategy:** Semantic/topic-based by legislative section (Tax Consolidation Act)
* **Chunk Size:** Up to 8,192 tokens for embeddings
* **Retrieval Pipeline:** Hybrid - Vector + BM25
* **Top-K Retrieval:** 5 chunks
* **Context Window:** Gemini Flash 2.0 supports ~1 million tokens
* **Future:** Summarization for large context windows
* **Citations:** Document name + paragraph shown in UI & response

# 4. Data & Document Management

* **Update Frequency:** As needed for MVP (formalize monthly schedule post-MVP)
* **Document Source:** Core Irish tax legislation (Tax Consolidation Act prioritized)
* **Ingestion Method:** Simple Python scripts for initial document processing
* **Document Removal:** Manual process during MVP phase

## Document Structure and Organization

### Actual Tax Legislation Samples

<https://d.docs.live.net/5162816083b4594e/Project%20Support%20Material/Python%20Projects/Tax_Rag/documentation/Specifications/irish-tax-legislation-samples.md>

# 5. Compliance & Legal

* **Disclaimer:**
  + Visible in UI and chatbot replies
  + Explicit: "This is not formal legal advice"
* **GDPR & Data Privacy:**
  + Hosting in the EU
  + Consider consent for storing chat logs / user data (future item)
  + GDPR compliance for EU data protection
  + Secure storage of user data with encryption at rest
  + Data retention policy: Chat history stored for 12 months, then anonymized

# 6. Subscription & Billing

* **Plans:**
  + Free Tier: 10 messages/week (token-based limit)
  + Paid Tier: 1,000 messages/month (to start)
* **Billing & Enforcement:**
  + Stripe integration
  + No refunds; cancel stops future billing, access continues until end of cycle
  + Stripe webhooks update subscription state
  + Show upgrade modal when usage exceeded
  + Leverage Stripe Checkout and Customer Portal for faster implementation instead of building custom payment UI

# 7. Usage Tracking & Quotas

* **Metric:** Token usage (primary cost driver)
* **Reset Jobs**: Implement via Supabase Edge functions (selected for simplified deployment)
* **Authorization**: Implement via Supabase Row Level Security policies instead of custom middleware
* **Rate Limits:** 60 requests/minute per user
* **UI Behavior:** Show upgrade modal instead of silent block

# 8. Chatbot Behavior

* **Memory:** Session-based
* **Tone:** Accurate, formal, legal-style
* **Citation:** Document + section output with every answer
* **Errors:** Graceful fallback on RAG failure or quota limits

# 9. Feedback & Moderation

* **User Feedback:**
  + Users can report inaccurate or offensive responses
  + Stored in Supabase feedback table
* **Admin Tools:**
  + Dashboard to review flags, users, usage
  + Freeze/ban user accounts
* **Automated Moderation:** Future roadmap (manual only for MVP)

# 10. Security & Access Control

## Secrets Management

Environment variables (Fly.io, Vercel)

## Authorization & Authentication

Authentication Flow:

* Supabase handles authentication (login/signup/user storage)
* Supabase issues JWTs following security best practices
* Backend validates these JWTs using middleware
* This hybrid approach combines managed authentication security with custom validation flexibility
  + Role-based access control (User, Admin)
  + Session timeout after 24 hours of inactivity
  + Rate limiting per user and IP address

## Session Strategy

Use Supabase sessions with JWT validation middleware for backend API routes

## Compliance

## Infrastructure Security

* All traffic over HTTPS
* Regular dependency scanning for vulnerabilities
* Firewall and rate limiting to prevent abuse
* Database backups performed daily

## LLM Security Considerations

* Input validation and sanitization to prevent prompt injection attacks
* Character and token limits on user inputs
* Blocking of specific patterns that could lead to prompt manipulation
* Regular security audits of prompt templates
* Monitoring for unusual query patterns that may indicate exploitation attempts

# 11. LLM Handling & Performance

## Rate Limiting

Per-user + optional IP throttling

## Concurrency

Scale via multiple Gunicorn/Uvicorn workers

## Streaming

Preferred if supported by Gemini Flash; fallback to blocking

### Model Resilience

* Primary model: Gemini Flash 2.0
* Fallback models: Configured backup models (e.g., GPT-3.5-turbo) in case of API disruption
* Circuit breaker pattern implementation to handle API degradation gracefully
* Clear error messaging to users when model services are unavailable

### Content Generation Controls

* Output filtering for potentially harmful or non-compliant content
* Confidence thresholds to reduce hallucinations
* System prompts that emphasize accuracy over completeness

### Error Handling Strategy

* Categorized error codes with user-friendly messages
* Specific handling for: quota exceeded, service unavailable, invalid input, retrieval failure
* Graceful degradation to standard Q&A when RAG retrieval fails
* Automatic logging of all errors for review and improvement

# 12. Branding & UX

* **Name:** TaxPal (placeholder)
* **Design System:** Chakra UI
* **Onboarding:** Show sample queries or a brief tour on first login

## Technical Specifications

### UI/UX Design Requirements

#### Brand Element

The application would follow these design principles:

* Clean, professional interface suitable for tax professionals
* Responsive design for desktop and mobile use
* Primary colors: Blues and grays (professional, trustworthy)
* Typography: Sans-serif font family for readability

#### User Flows

Key user flows:

* Landing → Sign up → Free tier usage → Quota reached → Subscription upgrade
* Landing → Login → Chat interface → Review history
* Admin → Document management → Review feedback

#### Interface Design

Chat interface would feature:

* Prominent disclaimer about not providing formal legal advice
* Citation display with clickable references to source documents
* Token usage display showing current quota
* Clear message history functionality
* Feedback options on each response

# 14. Implementation Next Steps

1. Scaffold FastAPI backend:
   * Routes: /chat/ask, /auth/\*, /billing/\*, /admin/\*
   * RAG pipeline integration with Gemini + Pinecone
   * Usage tracking
   * Stripe webhook listener
2. Scaffold frontend app (Next.js):
   * Auth flow with Supabase
   * Chat interface
   * Profile/subscription view
   * Landing page & upgrade modal
3. CI/CD setup:
   * GitHub Actions for test + deploy
   * Coverage enforcement in pipelines
4. Add disclaimer and token usage display to UI
5. Deploy initial environments: Dev, Staging, Prod

# 15. RAG Quality Evaluation

Evaluation Metrics

* Accuracy: Correctness of tax information compared to source documents
* Citation precision: Relevance of cited sources to the question
* Response completeness: Coverage of all aspects of the query
* Consistency: Maintaining the same answers for similar questions

Evaluation Process

* Automated evaluation using predefined test cases covering common tax scenarios
* Regular manual review by tax domain experts
* Feedback collection mechanism from users with classification of issues
* Quarterly comprehensive audit of randomly selected responses

Performance Benchmarks

* Response generation time: < 3 seconds for typical queries
* Retrieval precision: > 85% relevant document chunks retrieved
* Citation accuracy: > 90% citations should be directly relevant to the query
* User satisfaction: Track positive feedback rate with target of > 85%

# 16. API Specification

Authentication Endpoints:

* Only custom endpoints not covered by Supabase client SDK:
* POST /auth/admin-login
* POST /auth/verify-token

**Document Metadata**

* **Last Updated:** [Insert Date]
* **Owner:** [Insert Product/Tech Lead Name]
* **Status:** MVP Planning