# **Wellness Companion AI - Project Summary**

## **Architecture Overview**

6-layer desktop AI application with local processing, RAG capabilities, and cloud fallback.

## **Core Features**

* **Local Document Processing**: PDF/DOCX upload and indexing
* **Semantic Search**: Vector-based similarity search
* **Multi-modal AI**: Text and document analysis
* **Web Search Fallback**: Low confidence triggers Tavily API
* **System Tray Integration**: Floating overlay like SliderAI
* **Privacy-First**: Local processing with optional cloud sync

## **Technology Stack by Layer**

### **1. Desktop Layer**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| UI Framework | PyQt6 + QSS | Desktop interface |
| Packaging | Briefcase | Cross-platform deployment |
| Entry Point | Core Orchestrator | API trigger management |
| Local LLM | Ollama | Offline processing |
| System Integration | System Tray | Background operation |

### **2. Core Backend Layer**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| API Framework | FastAPI | REST API server |
| Authentication | Google OAuth + AWS Cognito | User management |
| Rate Limiting | Redis | API protection |
| Health Checks | Custom endpoints | System monitoring |
| RAG Framework | LangChain/LlamaIndex | Document processing |

### **3. AI/ML Orchestration Layer**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| Vector Search | Cosine similarity | Document retrieval |
| Local LLM | Ollama (Gemma 3B/Mistral 7B) | Response generation |
| Web Search | Tavily API | Fallback mechanism |
| Response Synthesis | Custom pipeline | Output formatting |
| Embeddings | Sentence Transformers | Vector generation |

### **4. Data Layer**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| Vector DB | Qdrant (primary) | Semantic search |
| Document Store | AWS S3 | File storage |
| Metadata DB | PostgreSQL | Structured data |
| Cache | Redis | Performance optimization |
| Embedding Pipeline | Custom | Document processing |

### **5. Infrastructure & Deployment**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| Cloud Platform | AWS (EC2, RDS, ElastiCache) | Hosting |
| Containerization | Docker | Deployment |
| API Gateway | NGINX | Security & routing |
| Database | PostgreSQL on RDS | Production data |
| Security | HTTPS/SSL | Data protection |

### **6. CI/CD & Logging**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| Version Control | GitHub Actions | Automation |
| Monitoring | Prometheus + Sentry | Error tracking |
| Packaging | PyInstaller + Briefcase | Executable creation |
| Deployment | Docker + AWS | Production deployment |

## **Model Recommendations**

### **Embedding Models**

| **Model** | **Dimensions** | **Accuracy** | **Speed** | **Size** |
| --- | --- | --- | --- | --- |
| all-MiniLM-L12-v2 ⭐ | 384 | 87% | Fast | 120MB |
| all-MiniLM-L6-v2 | 384 | 85% | Fast | 90MB |

### **LLM Models**

| **Model** | **Size** | **RAM** | **Accuracy** | **Speed** |
| --- | --- | --- | --- | --- |
| LLaMA2:7b ⭐ | 3.8GB | 8GB | 85% | Medium |
| Gemma:3b 🎯 | 2GB | 4GB | 82% | Fast |
| Mistral:7b | 4.1GB | 8GB | 87% | Medium |

### **Vector Databases**

| **Database** | **Performance** | **Setup** | **Scalability** | **Features** |
| --- | --- | --- | --- | --- |
| Qdrant ⭐ | Excellent | Easy | High | Full-featured |
| FAISS 🎯 | Excellent | Medium | High | Minimal |
| ChromaDB 🎯 | Good | Easy | Medium | Simple |

## **Performance Optimizations**

* **Cythonization**: Core files for faster processing
* **Quantization**: FP32 → INT8 (4x faster, 4x less memory)
* **Batch Processing**: 3x faster indexing
* **Async Pipeline**: 2x throughput
* **GPU Acceleration**: 10x faster inference

## **Deployment Strategy**

1. **Development**: Local environment setup
2. **Containerization**: Docker for consistency
3. **Production**: AWS deployment with auto-scaling
4. **Distribution**: DMG/EXE installers via Briefcase
5. **Monitoring**: Real-time health checks and logging

## **Key Workflows**

* **Query Processing**: Desktop → FastAPI → RAG/LLM → Response
* **Document Ingestion**: Upload → Chunking → Embedding → Vector Storage
* **Authentication**: OAuth → JWT → Keyring Storage
* **Fallback**: Low confidence → Web search → Enhanced response