Complete RAG Desktop Application Directory Structure

With Phase Mapping & Purpose Documentation

text

```
rag-desktop-app/
├── README.md              [phase: 1, 15][documentation & setup guide, final API docs]
├── .gitignore             [phase: 1][version control exclusions for Python/Docker/builds]
├── requirements.txt            [phase: 2, 3, 4, 5, 6, 8, 9, 10, 11, 14][Python dependencies - updated across
phases]
├── .env.example           [phase: 2, 9, 12][environment variables template for all configs]
├── docker-compose.yml          [phase: 2][service orchestration - FastAPI, PostgreSQL, Qdrant, Ollama]
│
├── backend/               [folder][API server with FastAPI, RAG pipeline, authentication]
│   ├── main.py            [phase: 3, 12][FastAPI app initialization, middleware, CORS setup]
│   ├── config.py          [phase: 3, 6, 9, 12][environment configuration, database URLs, API keys]
│   ├── api_routes.py          [phase: 3, 7, 8, 10, 12][all REST endpoints, request/response handling]
│   ├── schemas.py            [phase: 3, 7, 10][Pydantic models for request/response validation]
│   ├── utils.py           [phase: 3, 4, 5, 9][file handling, chunking algorithms, embedding utilities]
│   ├── documents.py            [phase: 4, 5, 6, 10][document processing, chunking, embedding, metadata
storage]
│   ├── rag.py             [phase: 6, 7, 8][Qdrant integration, semantic search, RAG pipeline]
│   ├── llm.py             [phase: 8, 9][Ollama/Gemma integration, response streaming, TAVILY fallback]
│   ├── database.py           [phase: 10, 12][PostgreSQL models, SQLAlchemy setup, user management]
│   └── auth.py            [phase: 12][Google OAuth implementation, JWT tokens, session management]
│
├── frontend/              [folder][PyQt6 desktop application with modern UI]
│   ├── main.py            [phase: 11, 13][PyQt6 app initialization, system tray integration]
│   ├── main_window.py          [phase: 11, 12, 13][main UI, chat widget, document panel, login interface]
│   ├── api_client.py          [phase: 11, 12][HTTP client for backend communication, auth handling]
│   ├── session_manager.py        [phase: 11, 12, 13][local session persistence, offline mode, background
sync]
│   ├── styles.qss            [phase: 11][modern CSS-like styling, dark theme, responsive design]
│   └── resources/            [folder][UI assets and application resources]
│       ├── icons/         [folder][phase: 11][app icons, system tray icons, UI button icons]
│       └── app.ico        [phase: 11, 14][main application icon for Windows/macOS builds]
│
├── deployment/              [folder][containerization, build automation, packaging]
│   ├── Dockerfile.backend       [phase: 2][FastAPI application containerization]
│   ├── Dockerfile.qdrant        [phase: 2][Qdrant vector store container setup]
│   ├── build_installer.py       [phase: 14][PyInstaller automation for .exe/.dmg creation]
│   └── deploy.sh             [phase: 2, 14, 15][Docker orchestration, container builds, final deployment]
│
├── tests/                 [folder][quality assurance and automated testing]
│   ├── test_auth.py           [phase: 15][OAuth flow testing, JWT validation, session management]
│   ├── test_rag.py           [phase: 15][document processing, retrieval, generation pipeline testing]
│   └── test_api.py           [phase: 15][endpoint testing, request/response validation]
│
└── scripts/               [folder][development automation and setup utilities]
    ├── setup_dev.py          [phase: 1, 2][development environment automation, initial setup]
    ├── init_models.py          [phase: 5, 8][download Sentence Transformers, setup Ollama/Gemma
models]
    ├── cythonize.py          [phase: 15][Cython compilation for performance optimization]
    └── deploy.sh            [phase: 15][final deployment automation and release management]
```

Phase Evolution Summary

Early Foundation (Phases 1-3)

- **Structure Setup**: Basic directories, Git, Docker foundation
- **Core Files**: `main.py`, `config.py`, `api_routes.py`, basic requirements

Backend Core Development (Phases 4-10)

- **Document Processing**: `documents.py`, `utils.py` for chunking and embedding
- **Vector Storage**: `rag.py` for Qdrant integration
- **LLM Integration**: `llm.py` for Ollama and TAVILY fallback
- **Data Layer**: `database.py` for PostgreSQL and metadata

Authentication & Frontend (Phases 11-12)

- **Desktop UI**: Complete `frontend/` folder creation
- **Authentication**: `auth.py` backend + frontend login integration
- **Session Management**: Cross-platform session persistence

System Integration (Phase 13)

- **System Tray**: Updates to `main.py` and `main_window.py`
- **Background Operations**: Enhanced `session_manager.py`

Production Ready (Phases 14-15)

- **Packaging**: `build_installer.py` and deployment automation
- **Testing**: Complete `tests/` folder with comprehensive coverage
- **Optimization**: `cythonize.py` for performance enhancement

Key File Interdependencies

Configuration Chain

- `.env.example` → `config.py` → All backend modules
- `requirements.txt` → Docker files → All Python modules

Backend Pipeline

- `utils.py` → `documents.py` → `rag.py` → `llm.py` → `api_routes.py`
- `database.py` → `auth.py` → `api_routes.py`

Frontend Integration

- `api_client.py` → `main_window.py` → `main.py`
- `session_manager.py` → All frontend components

Deployment Chain

- All source files → `build_installer.py` → Final executables
- Docker files → `deploy.sh` → Production deployment

This structure ensures systematic development with clear phase progression and minimal rework across the 15-phase development cycle.