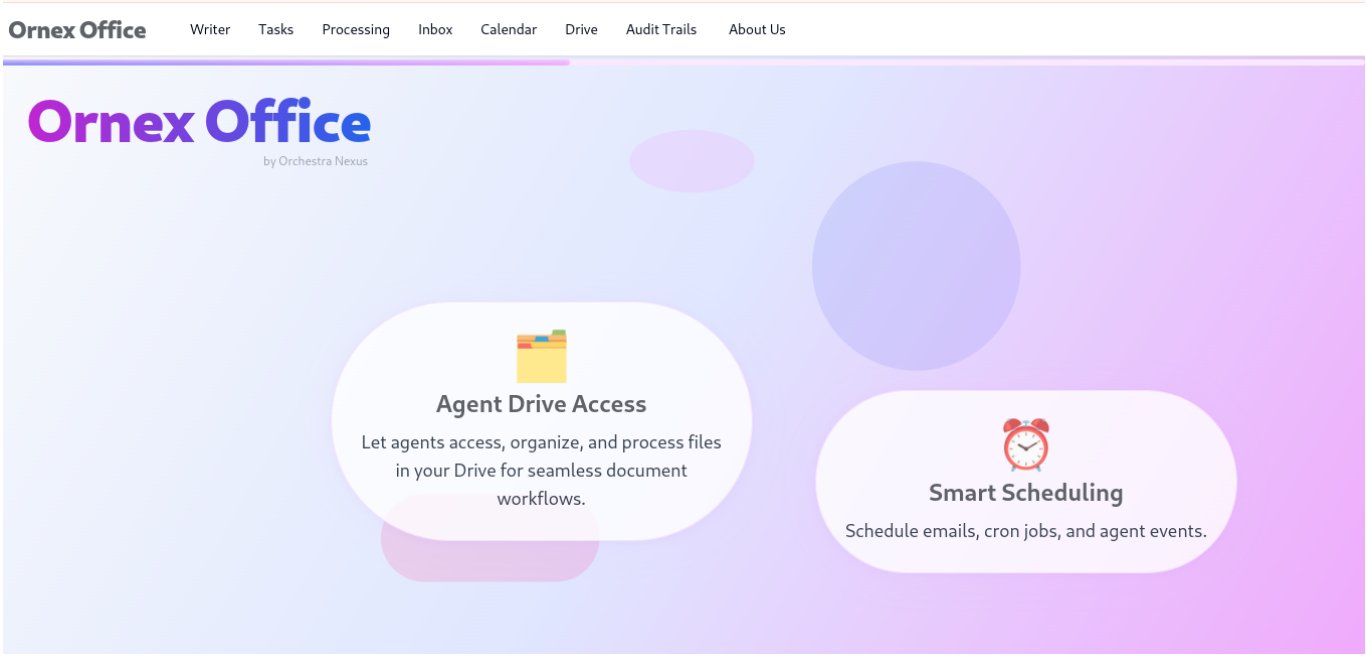
Ornex Office – Modern Agent Workspace

Overview

Ornex Office is a next-generation workspace platform that unifies email, document writing, task management, scheduling, and file automation—all powered by intelligent agents. Designed for modern teams and professionals, Ornex Office enables you to:

- Automate email categorization, labeling, and processing with AI agents.
- Write, summarize, and rewrite documents or emails using an AI-powered writing assistant.
- Build and run custom document processing workflows (e.g., validation, agent review, export to Drive/Sheets).
- Manage tasks, schedule events, and automate multi-step workflows visually.
- Integrate with Google Calendar and Drive for seamless scheduling and file management.
- Chat in real time with agents to get help, assign tasks, or trigger automations.
- Track every action with a secure, transparent audit trail.
- Enjoy a beautiful, responsive, and modern interface with a modern, responsive UI/UX.

Ornex Office is fully dockerized for easy development and production deployment, and is built to be extensible with more agent-powered tools and integrations coming soon.



Office Tools Overview

- **Email (Gmail):** Full-featured inbox, agent-powered automation, and chat-based actions.
- **Sheets:** Data export, reporting, and spreadsheet automation.
- **Calendar:** Event scheduling, reminders, and agent-managed bookings.
- **Drive:** File storage, document processing, and workflow integration.
- **Writer:** AI-powered document and email composition, with export to Sheets, PDF, or email.
- **Processing:** Document upload, validation, agent review, and multi-step automation.
- **Tasks:** Assign, track, and automate tasks across all connected tools.
- **Audit Trails:** Track every action for transparency and compliance.

How Connectors Work

- **In the Chat:** You can ask agents to perform actions like "Send this as an email", "Save to Google Sheets", or "Schedule a meeting". The chat interface routes these requests to the appropriate connector.
- **In Workflows:** The visual workflow builder lets you chain together actions across connectors (e.g., process a PDF, validate, save to Drive, notify via email).
- **In Tasks:** Assign tasks to agents that involve one or more connectors, such as "Summarize this document and email it to my team" or "Extract data from PDF and update my spreadsheet".

Ornex Office is designed to be extensible—more connectors and tools (e.g., Slack, Google Docs) can be added to further automate your workflows.

Tasks & Workflows

- **Tasks Page:** Manage and track tasks assigned to agents or users. (Planned: advanced task states, assignment, and notifications.)
- **Workflow Automation:** Visual workflow builder lets you create, connect, and orchestrate multi-step automations (e.g., validation, agent processing, export to Drive/Sheets).
- **Drag & Drop:** Build workflows by dragging nodes (validation, agent, save, etc.) and connecting them visually.
- **Agent Orchestration:** Combine multiple agents and tools in a single workflow for complex document or email processing.
- **Integration:** Workflows can interact with email, Drive, Sheets, and more—enabling end-to-end automation.

Document Processing & Workflow Builder

A new, modern Document Processing page lets users upload or select a document and visually build custom workflows. The main area features a workflow builder (visual builder, ready for react-flow or similar) with draggable node types:

- **Human/Agent Validation:** Add manual or agent-based review steps.
- **Agent Processing:** Let agents analyze, extract, or summarize documents.
- **Advanced Processing:** Placeholder for future advanced logic or integrations.
- **Start Agent Task:** Trigger agent-powered automations as part of the workflow.
- **Save to Drive:** Store processed documents directly in Google Drive.
- **Save to Sheets:** Export extracted or processed data to Google Sheets.

This enables end-to-end automation for document-centric workflows, combining validation, agent intelligence, and seamless integration with your office tools.

Architecture

Connectors & Office Tools

Ornex Office integrates multiple office tools and services through agent-powered connectors (MCP servers). These connectors allow you to automate, orchestrate, and interact with your favorite productivity tools directly from the chat, workflows, and task pages:

Scheduler System

- **Scheduler Page:** New `/scheduler` page in the frontend for creating, pausing, and deleting scheduled tasks (email, cronjob, agent event).
- **Backend API:** Endpoints for managing scheduler tasks (`/api/scheduler/tasks`, `/api/scheduler/task`, pause/delete). Currently uses in-memory store; DB persistence planned.
- **AgentScheduler:** Backend logic for scheduling and (soon) executing tasks.

Audit Trail

- **Audit Trail Page:** New `/audit` page in the frontend displays audit logs from the backend.
- **Backend:** `/api/audit` endpoint returns audit logs from PostgreSQL.
- **Audit Logging:** Actions like labeling emails are logged to the audit trail.

WebSocket Agent Chat

- **Real-time Chat:** `/ws/agent` endpoint enables chat with the AI agent (Gemini/MCP integration) via WebSocket.
- **Error Handling:** Improved error/traceback reporting to frontend.

Security & Secrets

- **Environment Variables:** All secrets (e.g., `GEMINI_API_KEY`, DB credentials) are handled via `.env` and passed securely in Docker Compose.
- **.gitignore:** Updated to ignore all sensitive files, build artifacts, and node_modules.

Usage

- **Scheduler:** Go to `/scheduler` to manage scheduled tasks. (Currently demo; real execution and DB persistence coming soon.)
- **Audit Trail:** Go to `/audit` to view recent audit logs (actions, labels, etc.).
- **Agent Chat:** Use the chat interface (if present) to interact with the AI agent in real time.

Planned & Upcoming Features

Ornex Office evolves from a Gmail dashboard to a full-featured agent workspace for modern office automation. The following features are planned or in development:

- **Multi-Agent Workspace:** Unified dashboard for email, document writing, task management, scheduling, and file workflows.
- **AI Writing Assistant:** Compose, summarize, and rewrite documents or emails with AI support. Export to Google Sheets, PDF, or email draft.
- **Document Processing:** Upload/select documents, build custom workflows (validation, agent processing, save to Drive/Sheets, etc.).
- **Inbox & Email Automation:** Categorize, label, and automate emails with agent support. Real-time chat with agents.
- **Task & Workflow Automation:** Visual workflow builder for multi-step automations (drag, connect, orchestrate agents and tools).
- **Calendar Integration:** Book meetings, sync events, and let agents manage your schedule.
- **Drive Integration:** Agents can access, organize, and process files in your Drive for seamless document workflows.
- **Audit Trails:** Full transparency for all agent and user actions, with secure logging and review.
- **About Us & Branding:** Consistent Ornex Office branding, modern navigation, and responsive UI/UX.
- **Extensible Integrations:** More tools and agent-powered features coming soon (e.g., Slack, Google Docs, advanced scheduling).

Roadmap / TODO

- [x] Modern animated landing page with carousel and SVG backgrounds
- [x] Multi-agent workspace: Writer, Tasks, Processing, Inbox, Calendar, Drive, Audit Trails, About Us
- [x] Dockerized dev/prod workflows (compose, build, hot-reload)
- [x] AI Writing Assistant page with export options
- [x] Document Processing page with workflow builder
- [x] Real-time agent chat (WebSocket)
- [x] PostgreSQL for audit trails and scheduler
- [x] Automated backend image build & push
- [ ] Advanced authentication (OAuth, SSO, etc.)
- [ ] Scheduler: advanced timing, recurrence, error handling
- [ ] More integrations (Google Calendar, Slack, Google Docs, etc.)
- [ ] Improved test coverage (unit/integration)
- [ ] Deployment documentation for cloud environments
- [ ] User/role management, permissions
- [ ] Mobile/responsive UI polish

Next steps

- [ ] Advanced authentication (OAuth, SSO, etc.)
- [ ] Expand scheduler features
- [ ] More integrations (e.g. Google Calendar, Slack)
- [ ] Increase test coverage
- [ ] Automate cloud deployment

Achieved milestones

- FastAPI backend with WebSocket and REST API
- PostgreSQL database for audit trail and scheduler

- Modern React/Vite frontend with scheduler and audit trail pages
- Agent integration (Gemini/MCP) with WebSocket chat
- Docker Compose for development and production
- Automated image build & push to Docker Hub

Technical Setup & Architecture

Architecture

- **Frontend:** React/Vite (already present)
- **Backend:** FastAPI (Python)
- **Database:** PostgreSQL (for audit trails)
- **Agent Communication:** WebSockets (chat with AI agent)
- **MCP Server:** Already running, integration as in `test.py`
- **Docker Compose:** Orchestrates backend, database, and MCP server

Quick Start with Docker Compose

1. **Requirements:**
    - Docker and Docker Compose installed
2. **Start the project:**

    ```
    docker compose up --build
    ```

    - The backend will be available at http://localhost:8000
    - The database runs on port 5432
    - The MCP server (gmail) runs as configured in the compose file
3. **Start the frontend locally (optional):**
    - Change to the `frontend/` directory and start the frontend as usual (e.g., with `npm run dev`).
4. **Database initialization:**
    - The SQL script `db/db_init.sql` is executed automatically on first start.

Services in docker-compose.yml

- **backend:** FastAPI app (Python, port 8000)
- **db:** PostgreSQL database (port 5432)
- **gmail:** MCP server for Gmail bridge

Example: Backend FastAPI structure

- `/api/emails` – List, details, label change

- `/api/audit` – Audit trail logs
- `/ws/agent` – WebSocket for chat

Development

- Changes to the backend may require a container restart (`docker compose restart backend`).
- You can use a `.env` file for environment variables.

Environment Variables

- Place a `.env` file in the root with at least:

```
GEMINI_API_KEY=your_gemini_api_key
DATABASE_URL=postgresql://postgres:postgres@db:5432/mailwhisperer
```

- All services in Docker Compose will pick up these variables.

Build and Development Setup

Build Images and Production

The build and Dockerfiles are located in the `build/` folder.

- To build the images and start the containers for production or build tests:

```
docker compose -f build/docker-compose.build.yml build
docker compose -f build/docker-compose.build.yml up
```

Development (Hot-Reload, Local Source Code)

For local development, prepared images are used and the actual development servers (e.g., Vite, Uvicorn) run outside of Docker or in their own dev images.

- Start the development environment with:

```
docker compose -f docker-compose.dev.yml up
```

- Adjust the environment variables in a `.env` file if necessary.

Notes

- The old Dockerfiles and Compose files have been moved to the `build/` folder.
- For local development, it is recommended to start the frontend directly with `npm run dev` and the backend with `uvicorn` to use hot-reload.
- The database remains persistent in the `db_data` volume.

Build and Push Backend Image

To build the backend image and push it to the registry, run:

```
# Log in once (only needed if you are not already logged in)
docker login

# Build and push the image (replace TAG, e.g., "latest" or "v1")
./build_and_push_backend.sh <TAG>
```

The image will be built and pushed as `orchestranexus/agentbox:<TAG>`.

Then adjust the backend service in your `docker-compose.yml`:

```
backend:
  image: orchestranexus/agentbox:<TAG>
  ...
```

You can then start as usual with Compose:

```
docker compose up
```

File Structure

```
mail-whisperer-drafts-labeller/
├── backend/                # FastAPI backend (Python)
│   ├── agent_ws.py         # Main agent logic
│   ├── agent_scheduler.py  # Scheduler logic
│   ├── ws_manager.py       # WebSocket manager
│   ├── tools_wrapper.py    # Tool integration
│   ├── backend_main.py     # (Entry point, if used)
│   └── test/               # Backend tests
│       ├── test_agent_manager.py
│       └── test_mcp_tools.py
├── frontend/               # React/Vite frontend (TypeScript)
│   ├── src/                # Main source code
│   │   ├── components/     # UI components
│   │   ├── pages/          # App pages (Scheduler, AuditTrail, etc.)
│   │   └── ...             # Other frontend code
│   ├── public/             # Static assets
│   └── ...                 # Configs, package.json, etc.
├── db/                     # Database initialization
│   └── db_init.sql
├── mcp/                    # MCP server scripts and config
│   ├── config/
│   └── scripts/
```

```
├── build/                  # Docker build files
│   ├── Dockerfile.backend
│   ├── Dockerfile.frontend
│   └── docker-compose.build.yml
├── docker-compose.yml      # Main compose file (uses pushed images)
├── docker-compose.dev.yml  # Dev compose file (for local dev servers)
├── build_and_push_backend.sh # Script to build & push backend image
├── requirements.txt        # Python dependencies
├── README.md
└── ...
```

Questions/Feedback welcome!