### Project Title:

**Automated Hillsborough County Crawler with PDF Intelligence and Firebase + Pinecone Integration**

### Objective:

Develop a Python-based pipeline that:

1. Scrapes Hillsborough County mortgage records via document type filter.
2. Saves structured record data to Firebase.
3. Visits each Instrument’s detailed page to:
   * Extract additional document information.
   * Download the full mortgage PDF.
4. Extracts insights from the PDF using OpenAI Vision.
5. Saves the output to Pinecone for intelligent search and retrieval.

### Target URLs:

* Main search: <https://publicaccess.hillsclerk.com/oripublicaccess/>
* Document detail: [https://publicaccess.hillsclerk.com/oripublicaccess/?instrument={INSTRUMENT\_ID}](https://publicaccess.hillsclerk.com/oripublicaccess/?instrument=%7bINSTRUMENT_ID%7d)

### Functional Requirements

### 1. Web Automation & Search

* Open portal using Selenium/Playwright.
* Navigate to:
  + **Document Type** filter.
  + Enter:
    - Document Type: (MTG) MORTGAGE
    - Start Date: 07/01/2025
    - End Date: 07/09/2025
* Click **Search** and wait for results to populate.

2. Navigate to **Export to Spreadsheet**

* Save the extracted data as a .csv or .xlsx file locally.

3**. Extract excel Data**

1. For each row, extract:

• Instrument

• Grantor

• Grantee

• RecordDate

• DocType

• BookType (optional)

• BookNum (optional)

• PageNum (optional)

• Legal (optional)

• SalesPrice (optional)

1. Save all results to: Firebase Firestore under collection /{DOCUMENT\_TYPE} /

/mortgage\_records/

3. **Visit Instrument Page**

1. For each Instrument ID:

https://publicaccess.hillsclerk.com/oripublicaccess/?instrument={INSTRUMENT\_ID}

**Extract left panel fields:**

o Document Type

o Recording Date

o Grantor

o Grantee

o Book

o Page

o Legal

o Direct link

1. Update status in Firebase as **Visited.**

**4. Download PDF**

• Locate the PDF viewer iframe on the right.

• Click the Download button (you will need to inspect its HTML to get its ID).

• Download the PDF and save it locally using InstrumentID.pdf.

5. Process PDF using OpenAI Vision

* Load the saved PDF file.
* Use openai Vision model (e.g., GPT-4V) to extract key content:
* Property details
* Loan amount
* Insurance limits
* Legal obligations
* Borrower/Lender terms
* Generate structured response.

6. Push Data to Pinecone

Convert OpenAI Vision output to vector embeddings using text-embedding-3-small or text-embedding-ada-002.

Upsert into Pinecone under namespace {DOCUMENT\_TYPE}using

**Folder Structure**

project/

│

├── data/

│ └── instrument\_csv.xlsx

│ └── pdfs/

│ └── 2025295324.pdf

│

├── firebase/

│ └── serviceAccount.json

│

├── src/

│ ├── search\_scraper.py

│ ├── detail\_scraper.py

│ ├── pdf\_downloader.py

│ ├── vision\_extractor.py

│ ├── firebase\_uploader.py

│ └── pinecone\_uploader.py

│

├── requirements.txt

└── README.md

**Firebase Firestore Database Structure**

**Top-level Collection:**

* records\_by\_document\_type/

Document Type Subcollections (Dynamic)

Each document\_type is stored as a dynamic document ID (key), e.g.:

**records\_by\_document\_type**/

├── (MTG) MORTGAGE/

│ ├── 2025295324/

│ ├── 2025295317/

│ └── ...

├── (D) DEED/

│ ├── 2025297890/

│ └── ...

└── ...

**Document Schema (per Instrument ID)**

Each Instrument ID (e.g., 2025295324) will be a document with fields like:

{

"instrument\_id": "2025295324",

"grantor": "KIRSCH INVESTMENT PARTNERS LLC",

"grantee": "TAYLOR MADE LENDING LLC",

"recording\_date": "2025-07-08T17:05:00",

"document\_type": "(MTG) MORTGAGE",

"book": null,

"page": null,

"legal": null,

"direct\_link": "https://publicaccess.hillsclerk.com/oripublicaccess/?instrument=2025295324",

"pdf\_downloaded": true,

"pdf\_path": "pdfs/2025295324.pdf",

"status": "completed", // one of: pending, visited, pdf\_downloaded, completed

"vision\_summary": {

"loan\_amount": "$619,000.00",

"insurance\_limit": "$2,000,000.00",

"obligations": "To pay taxes, assessments, obtain insurance...",

"raw\_text": "PROVIDED ALWAYS that if the Mortgagor shall..."

},

}

**Status Lifecycle (for Tracking)**

| **Status** | **Meaning** |
| --- | --- |
| pending | Just discovered from search |
| visited | Detail page visited, info scraped |
| pdf\_downloaded | PDF file downloaded |
| completed | AI summary + Pinecone vector generated and uploaded |

### Pinecone structure:

### {

### "grantor": "KIRSCH INVESTMENT PARTNERS LLC",

### "grantee": "TAYLOR MADE LENDING LLC",

### "recording\_date": 1752042300, // This is the Unix timestamp for 2025-07-08T17:05:00

### "loan\_amount": 619000.00,

### "document\_type": "(MTG) MORTGAGE",

### "direct\_link": "https://publicaccess.hillsclerk.com/oripublicaccess/?instrument=2025295324",

### "insurance\_limit": 2000000.00,

### "instrument\_id": "2025295324"

### }

### Epic 1: Project Setup & Configuration

*(This remains the same as it's a foundational step)*

* **Task 1.1: Initialize Project Environment**
  + Subtask 1.1.1: Create the specified project directory structure (project/, data/, src/, etc.).
  + Subtask 1.1.2: Set up a Python virtual environment and requirements.txt.
* **Task 1.2: Configure Service Integrations**
  + Subtask 1.2.1: Set up Firebase, get the serviceAccount.json.
  + Subtask 1.2.2: Set up OpenAI and configure the API key.
  + Subtask 1.2.3: Set up Pinecone, create an index, and configure API keys.

### Epic 2: Initial Search & Data Export (One-Time Step)

This epic covers the first step which happens once per run to get the list of all instruments to be processed.

* **Task 2.1: Develop Initial Search & Export Script (search\_scraper.py)**
  + Subtask 2.1.1: Implement logic to launch a browser and navigate to the search portal.
  + Subtask 2.1.2: Enter search criteria (Document Type, Start Date, End Date).
  + Subtask 2.1.3: Execute the search and wait for results.
  + Subtask 2.1.4: Navigate to the "Export to Spreadsheet" function and download the results to data/instrument\_csv.xlsx.

### Epic 3: Sequential Record Processing Pipeline

This epic defines the main, continuous workflow that processes each record individually from start to finish.

* **Task 3.1: Develop the Main Orchestrator**
  + Subtask 3.1.1: Write the main script that will orchestrate the entire process.
  + Subtask 3.1.2: Read the exported instrument\_csv.xlsx file into a list or DataFrame.
  + Subtask 3.1.3: Initialize all required clients (Firebase, OpenAI, Pinecone, Selenium/Playwright).
  + Subtask 3.1.4: Create a master loop to iterate through **each row** (each instrument) from the spreadsheet.
* **Task 3.2: Process a Single Record (Inside the Loop)**
  + **Subtask 3.2.1: Initial Data Ingestion (firebase\_uploader.py function)**
    - For the current instrument, extract the initial data (Instrument ID, Grantor, Grantee, etc.).
    - Save this initial data to a new document in Firebase under records\_by\_document\_type/(MTG) MORTGAGE/{InstrumentID}.
    - Update the document status to pending.
  + **Subtask 3.2.2: Visit Detail Page & Scrape Info (detail\_scraper.py function)**
    - Construct the detail page URL for the current instrument.
    - Navigate to the URL.
    - Scrape the additional fields from the left panel.
    - Update the same Firebase document with the newly scraped fields.
    - Update the status to visited.
  + **Subtask 3.2.3: Download the PDF (pdf\_downloader.py function)**
    - While on the detail page, locate the PDF viewer iframe and download button.
    - Download the PDF, saving it locally as data/pdfs/{InstrumentID}.pdf.
    - Update the Firebase document with pdf\_downloaded: true and the pdf\_path.
    - Update the status to pdf\_downloaded.
  + **Subtask 3.2.4: Process PDF with OpenAI Vision (vision\_extractor.py function)**
    - Load the PDF file that was just downloaded.
    - Convert PDF to an image format suitable for the Vision API.
    - Send the image to the OpenAI API with the prompt to extract key information.
    - Parse the structured response.
    - Update the Firebase document with the vision\_summary.
  + **Subtask 3.2.5: Vectorize and Upsert to Pinecone (pinecone\_uploader.py function)**
    - Take the vision\_summary text.
    - Generate a vector embedding using the OpenAI embeddings model.
    - Upsert the vector and metadata (using Instrument ID as the vector ID) to the Pinecone index.
  + **Subtask 3.2.6: Finalize Record Status**
    - After the successful Pinecone upsert, update the status in the Firebase document to completed.
    - Log the completion of the instrument and move to the next iteration of the loop.