Team Inferno Intellect

Our Exceptional Team

- Mir Inayat Ahmed
- Abdullah Shaikh
- Mohammed Muqeet Us Salam
- Armaan Rashid Pathan

Github Url: https://github.com/Mir-Inayat/inferno

SMARTDOCMATE

AI

Why Us?

Unique Features

- **Multilingual Support**
- **AI Assistant with Voice Capabilities-Communicates document details interactively using voice-based AI.**
- **Support for Various Formats**
- **Open-Source and Cost-Effective**
- **Batch Processing Capability**
- **Hierarchical Document Categorization**
- **AI-Powered Automation**
- **Drag-and-Drop Functionality**
- **OCR Integration for Text Extraction**
- **Categorization Results with Confidence Scores**
- **Free and Scalable Deployment**
- **User Feedback Integration**

Current Progress!!!

Features Implemented

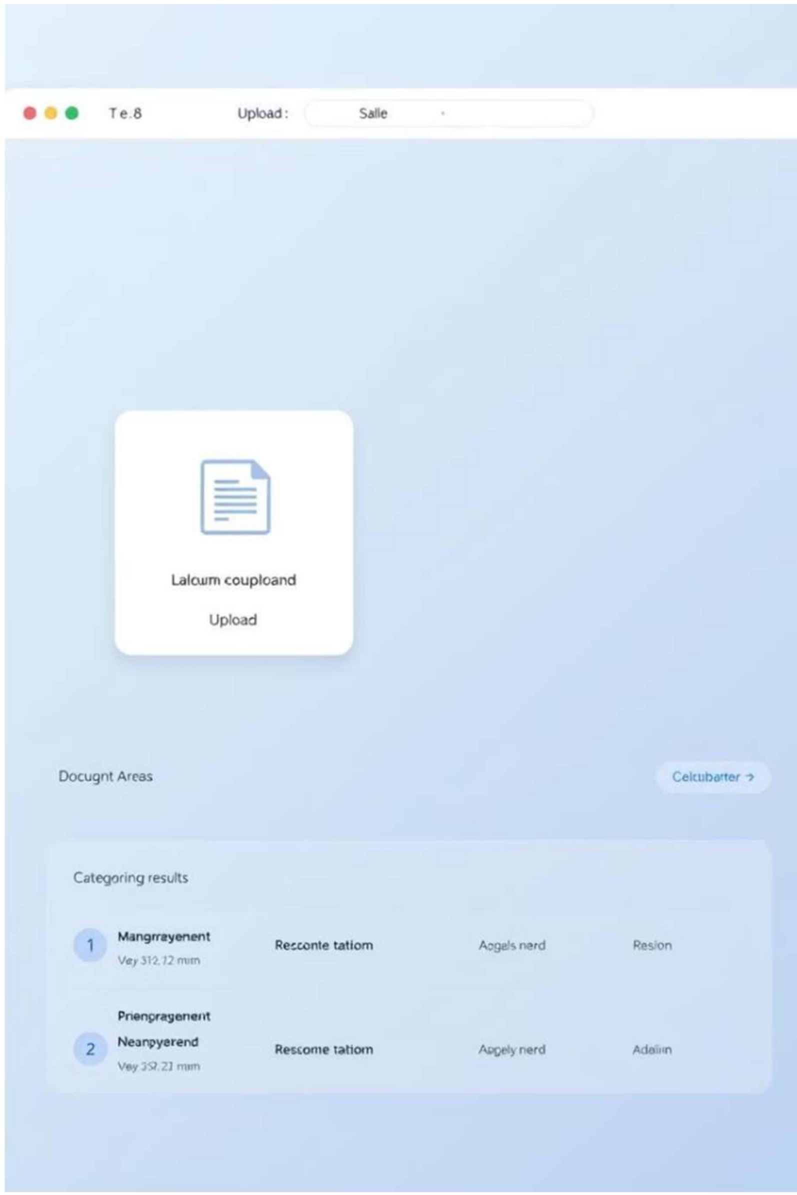
- **Support for Various Formats**
- **Open-Source and Cost-Effective**
- **Batch Processing Capability**
- **Hierarchical Document Categorization**
- **Drag-and-Drop Functionality**
- **AI-Powered Automation**
- **OCR Integration for Text Extraction**
- **Categorization Results with Confidence Scores**

Under Development

- **Multilingual Support(ToDo)**
- **AI Assistant with Voice Capabilities-Communicates document details interactively using voice-based AI.(Ready Just need to be integrated)**
- **Free and Scalable Deployment(Deployment pending for other feature implementations)**
- **User Feedback Integration(ToDo)**

Challenges in Manual Document Processing

•Financial institutions handle thousands of unstructured documents daily.
•Manual verification and organization are time-intensive and repetitive.
•Need for automation to enhance efficiency and reduce errors.



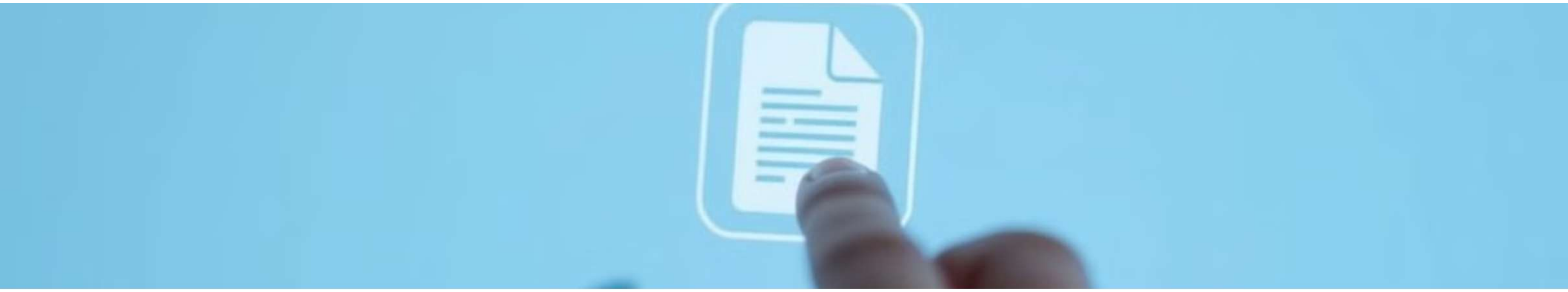
Focus Areas for MVP: Upload and Categorization

Document Upload

The core functionality of our MVP revolves around seamless document upload. We'll implement drag-and-drop functionality, allowing users to easily upload financial documents in PDF format. The application will seamlessly process these documents, extracting relevant information for categorization and summarization.

Categorization

Our MVP will incorporate a robust document categorization system. system. We'll leverage a free pretrained model, LayoutLMv3, from from Hugging Face. This model will analyze the uploaded documents documents and predict their categories based on content and layout. layout.



Prototype Features: Drag-and-Drop & Results Display

Drag-and-Drop Upload

Users will be able to simply drag and drop documents from their computer onto the app's interface, eliminating the need for traditional file selection menus. This intuitive feature will enhance user experience and make document upload a breeze.

Categorization Results Display

Upon processing, the application will display the predicted categories in categories in a clear and concise format. We'll implement a table or table or card-based display, presenting the categorization results along results along with confidence scores, providing users with a comprehensive overview of the document's classification.

Tools and Technologies: Frontend, Backend, Deployment Deployment

Frontend: React

For rapid web interface development, we'll utilize react, a Javascript framework known for its ease of use and ability to build interactive web apps quickly. This will ensure a user-friendly interface that facilitates document upload and results display.

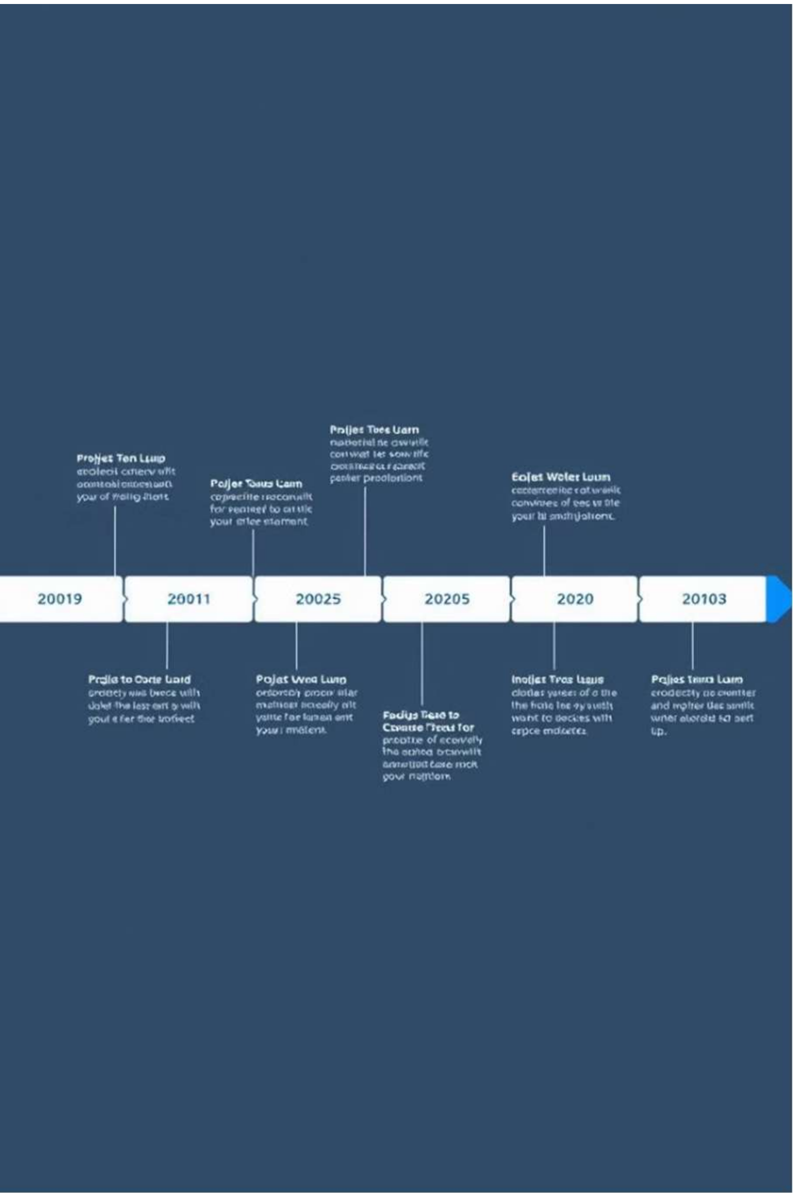
Backend/Processing: Hugging Face Transformers

•Automates categorization and summarization using advanced AI models.
•Hierarchical classification of documents based on type and associated individual.
•Tools: Hugging Face LayoutLMv3, Streamlit, Tesseract OCR.
•Supports various document formats (PDF, images, text files).
•Multilingual support for diverse datasets.

Deployment: Streamlit Community Community Cloud

For deployment, we'll use Streamlit Community Cloud, a free platform that makes makes it easy to host Streamlit applications. applications. Alternatively, we can explore explore options like Render or Heroku, which which offer free tiers for hosting Python web web applications.

Batch Processing: Redis and Celery for handling large volumes of documents efficiently.



Timeline: MVP Development and Deployment

1

Day 1: Setup & Initial Testing

We'll start by setting up the basic Streamlit app with file upload functionality. We'll then then integrate a pretrained classification model (LayoutLMv3) and conduct initial testing to testing to validate the functionality.

2

Day 2: Deployment & Presentation

We'll deploy the MVP application on Streamlit Community Cloud, making it accessible for testing and demonstration. We'll also create a comprehensive presentation highlighting the problem statement, solution, implementation details, and future extensions.

3

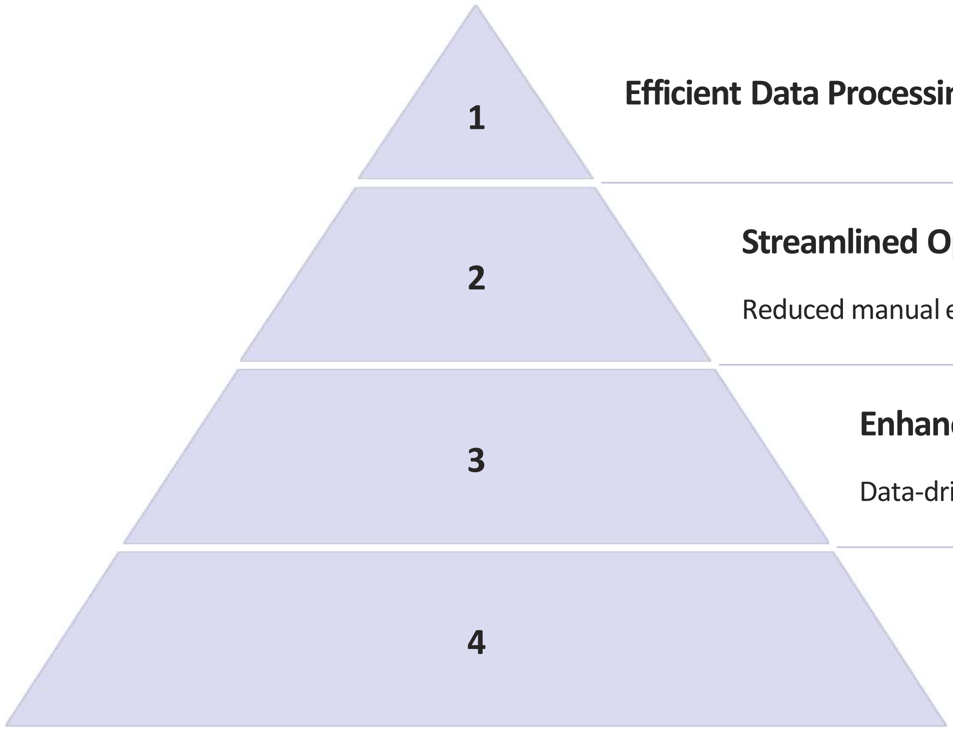
Day 7: Refinement & Feature Enhancements

Based on feedback and insights gained from initial testing and user interaction, we'll refine the we'll refine the application. We'll also incorporate features like user correction feedback and feedback and batch processing to enhance the user experience and functionality.

Code Snippet: A Glimpse into the Implementation

```
import streamlit as st
from transformers import pipeline
import pytesseract
from pdf2image import convert_from_path

# Load pretrained models
categorizer = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
summarizer = pipeline("summarization")

# Function to extract text from PDF
def extract_text_from_pdf(file):
  images = convert_from_path(file)
  text = ""
  for image in images:
    text += pytesseract.image_to_string(image)
  return text

# Streamlit app
st.title("Document Categorization and Summarization")
st.write("Upload financial documents for categorization and summary.")

uploaded_file = st.file_uploader("Upload a PDF", type=["pdf"])
if uploaded_file is not None:
  # Extract text with st.spinner("Processing document..."):
  text = extract_text_from_pdf(uploaded_file)

  st.subheader("Extracted Text")
  st.text_area("Document Text", text, height=200)

  # Categorize
  st.subheader("Categorization")
  categories = ["Bank Application", "Identity Document", "Financial Document", "Receipt"]
  result = categorizer(text, categories)
  st.write("Predicted Category:", result["labels"][0])

  # Summarize
  st.subheader("Summarization")
  summary = summarizer(text, max_length=100, min_length=25, do_sample=False)
  st.write("Summary:", summary[0]["summary_text"])
```

Key Takeaways and Next Steps



Efficient Data Processing

Streamlined Operations

Reduced manual effort and improved accuracy.

Enhanced Decision-Making

Data-driven insights for better informed decisions.

Future Extensions

User feedback integration, cloud storage, advanced compliance.