### **Intelligent Document Processing (IDP) POC: Business Requirements Document**

**1.0 Executive Summary** This document outlines the business requirements for a Proof of Concept (POC) for an Intelligent Document Processing (IDP) system. The primary objective is to demonstrate how AI can significantly accelerate the mortgage underwriting process by intelligently handling and extracting key information from a bundle of borrower documents. The system will automate the tedious tasks of document classification, data extraction, and summarization, allowing underwriters to focus on high-value, complex tasks and decision-making.

**2.0 Business Goals**

* **Increase Efficiency:** Drastically reduce the time underwriters and loan officers spend on manual data entry and document review.
* **Improve Accuracy:** Minimize human error in data extraction and document organization.
* **Enhance Underwriter Focus:** Allow underwriters to shift their focus from administrative tasks to critical risk assessment and borrower relationships.
* **Demonstrate Agentic AI:** Showcase a powerful, integrated AI workflow where a single model performs multiple, chained tasks to provide a cohesive solution.

**3.0 Stakeholders**

* **Primary Users:** Mortgage Underwriters, Loan Officers.
* **Business Sponsor:** Head of Lending/Mortgage Division.
* **Development Team:** POC Engineering Team.

**4.0 Functional Requirements** The system must be able to perform the following actions:

* **Document Upload:** Allow a user to upload a single PDF file that may contain multiple mortgage-related documents (e.g., W-2s, pay stubs, bank statements).
* **Automated Document Breakdown:** The system must automatically split the uploaded PDF bundle into individual, logically separate documents.
* **Key Data Extraction:** From each split document, the system must extract specific, predefined data points relevant to the underwriting process.
* **AI-Generated Summary:** The system must provide a concise, professional, AI-generated summary for each document and the entire document bundle from an underwriter's perspective.
* **Human-in-the-Loop Review:** Present the AI's output in a clean, intuitive user interface for a human to review and validate. The interface must clearly show confidence scores and flag potential issues.
* **Output Management:** Allow the user to download individual split documents and copy extracted data or summaries to the clipboard.

**5.0 Non-Functional Requirements**

* **Performance:** The AI processing of a typical document bundle should complete within a reasonable timeframe (e.g., under 60 seconds).
* **Reliability:** The system should handle errors gracefully and provide clear feedback to the user.
* **Security:** As this is a POC for a local environment, basic security measures are acceptable. All uploaded files should be handled and deleted securely after processing.
* **Usability:** The user interface must be clean and intuitive, requiring minimal training for a new user.

**6.0 Success Criteria** The POC will be considered a success if it can:

* Successfully process and split a sample document bundle containing at least three different document types.
* Accurately extract key data points from each document with a verifiable high confidence score.
* Generate a relevant and useful underwriter-centric summary of the documents.
* Demonstrate a clear reduction in manual data entry and review time for the user.

### **Intelligent Document Processing (IDP) POC: Technical Solutioning Document**

**1.0 Architecture Overview** The application will be built on a three-tier architecture: **Front-End**, **Back-End**, and an **AI Agent**. This modular approach ensures that each component can be developed and tested independently.

**2.0 Technology Stack**

* **Front-End:** **React**. A component-based JavaScript library for building the user interface.
* **Back-End:** **Python** and **Flask**. A simple, yet powerful, web framework for handling API requests and serving the application.
* **AI Agent:** **Google Gemini API** (gemini-2.5-flash). This model will be used with a structured JSON schema to ensure predictable and reliable output.
* **PDF Processing:** **PyPDF2**. A pure-Python library for manipulating PDF files.

**3.0 Data Flow and Logic**

1. **User Action:** The user uploads a bundled PDF file to the React front-end via a drag-and-drop interface.
2. **Front-End to Back-End:** The React app sends the PDF file to the back-end's /process\_document endpoint via a **POST** request. The file is sent as multipart form data.
3. **Back-End to AI Agent:**
   * The Back-End server receives the file, saves it, and converts its contents into a Base64-encoded string.
   * It then constructs a JSON payload that includes the system instructions, the responseSchema, and the user's prompt with the Base64-encoded PDF data.
   * This payload is sent to the Gemini API endpoint.
4. **AI Agent Response:** The Gemini API returns a single, structured JSON object containing all the information requested: a list of classified documents with their page ranges, key data highlights, titles, and summaries.
5. **Back-End Orchestration:**
   * The Back-End receives the JSON response from the AI.
   * Using the page ranges provided by the AI (page\_start and page\_end), it uses PyPDF2 to physically split the original PDF file into separate documents.
   * These split documents are saved to a local folder and made available for download via a dedicated endpoint.
6. **Back-End to Front-End:** The Back-End constructs a final JSON response for the Front-End. This response includes the AI-generated data (summaries, highlights) and the URL paths for downloading the newly split documents.
7. **Front-End Rendering:** The React app receives this JSON and updates its state to display the parsed information, allowing the user to interact with the data and download the split files.

### **Intelligent Document Processing (IDP) POC: Gemini AI Requirements**

This document specifies the precise requirements for the interaction with the Gemini AI model, including the endpoint, system instructions, and the expected output schema. This is the most critical part of the AI-driven workflow.

**1.0 Gemini Endpoint and Request Body** The following endpoint and request body will be used for all AI interactions:

* **Endpoint:** https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent

**Request Body (JSON Payload):**JSON  
{

"systemInstruction": {

"parts": [

{

"text": "You are a mortgage document pack classifier and summarizer. Return ONLY JSON that matches the schema. Infer doc\_type yourself (no predefined list). Normalize doc\_type to short snake\_case (e.g., w2, paystub, bank\_statement). Identify page ranges within the bundle (1-based inclusive). Provide a concise paragraph summary and a list of key highlight fields as key-value pairs. Follow privacy rules: mask SSNs and full account numbers (use last4); minimize PII; use ISO dates (YYYY-MM-DD). If a type is unclear, set doc\_type='other' and explain briefly in summary."

}

]

},

"generationConfig": {

"responseMimeType": "application/json",

"responseSchema": {

"type": "object",

"properties": {

"documents": {

"type": "array",

"items": {

"type": "object",

"properties": {

"doc\_type": { "type": "string" },

"title": { "type": "string" },

"page\_start": { "type": "integer", "minimum": 1 },

"page\_end": { "type": "integer", "minimum": 1 },

"confidence": { "type": "number", "minimum": 0, "maximum": 1 },

"summary": { "type": "string" },

"key\_highlights": {

"type": "array",

"items": {

"type": "object",

"properties": {

"field": { "type": "string" },

"value": { "type": "string" }

},

"required": ["field", "value"]

}

}

},

"required": ["doc\_type", "page\_start", "page\_end", "summary", "key\_highlights"]

}

}

},

"required": ["documents"]

},

"maxOutputTokens": 10000

},

"contents": [

{

"role": "user",

"parts": [

{

"text": "Classify all sub-documents in this file and return only JSON per the schema. For images, set page\_start=1 and page\_end=1. Title should be a human-friendly heading found on the document. summary: one short paragraph describing what the document is and any noteworthy context. key\_highlights: include 3–10 key fields as {field, value} pairs relevant to mortgage processing (examples if present: tax\_year, employer, wages\_box1, federal\_withholding\_box2; pay\_period\_start, gross\_pay, net\_pay; statement\_period\_start, ending\_balance, account\_last4; property\_address, est\_value\_or\_loan\_amount, issue\_or\_closing\_date). Use last4 for any identifiers; omit full SSNs and full account numbers."

},

{

"inline\_data": {

"mime\_type": "application/pdf",

"data": "{{ $json.data }}"

}

}

]

}

]

}

**2.0 Key Requirements for the API Call**

* **Structured Output:** The generationConfig with responseMimeType and responseSchema ensures that the model's response is a valid JSON object, preventing parsing errors on the back-end.
* **System Instructions:** The systemInstruction acts as a highly specific persona for the AI, guiding its behavior and ensuring it follows privacy and formatting rules.
* **Comprehensive Prompt:** The contents section combines the user prompt with the document itself, allowing the model to perform multiple, complex tasks (classification, extraction, and summarization) in a single request. This is the core of the agentic approach.