1. Approach and Workflow

The invoice extraction solution is built on an Object-Oriented Programming (OOP) foundation to ensure modularity and maintainability across the complex workflow.

The process is sequential, involving four main steps:

1. OCR Text Retrieval: Calls the Veryfi API to extract raw OCR text from the input PDF file.
2. Document Validation: Checks the document's structural integrity using generalized regex patterns.
3. Data Extraction: Parses the validated OCR text to extract required fields.
4. Result Consolidation: Saves the extracted data to a JSON output file.

2. Solution Modules and Responsibilities

The solution is divided into three key, cohesive modules, each with a defined responsibility:

OCR & Text Retrieval

This layer handles secure communication with the external Veryfi API via the `VeryfiOCRClient`. Its primary responsibility is converting the input PDF into a raw text string.

Document Validation

The `InvoiceParser` executes its `validate_format` method, which employs generalized regex patterns (fingerprints) to structurally inspect the document layout. Documents failing this check are skipped from further processing.

Data Extraction & Persistence

This module handles the core transformation. If the document is validated, the `InvoiceParser` extracts critical fields (date, totals, and line items, including handling multiline descriptions) and structures them into a Python dictionary. The `DataSaver` then converts this dictionary to a JSON file and persists it in the `extractedData` directory.

3. Coding Best Practices

- Modularity: The codebase is designed for high maintainability through clear, modular organization.

- Secure Credential Handling: Credentials are loaded and validated separately via `app_secrets.py`. This isolates secrets from the main codebase and allows for easy exclusion from version control (Git).
- Robust Error Handling: Instead of crashing, the custom `InvalidDocumentFormatError` is specifically caught by the `main.py` orchestrator. This prints a "SKIPPING" message, allowing the batch process to seamlessly continue to the next file without interruption.

4. Unit Testing

The project incorporates unit tests to verify core functionality:

- Full Data Extraction: Confirms that the parsing logic correctly converts mocked OCR text (designed to test multilane descriptions and multipage documents) into the correct JSON output structure.
- Format Validation: Ensures the system correctly rejects documents that do not match the expected format (tested with mocked incorrect text).

5. Assumptions

The implementation is based on the following assumptions regarding the input documents and output mapping:

- Line Item Field Mapping: The standard invoice fields Quantity, Rate, and Amount are mapped to the following JSON output keys:
  - `Quantity` → `quantity`
  - `Rate` → `price`
  - `Amount` → `total`
- Missing Fields: Fields such as `sku` and `tax_rate` are assumed not to be present in the document and are consequently assigned null values for each line item.
- Line Item Definition: An invoice line item is strictly defined as a single row in the document table containing the description, quantity, rate, and amount.