**Documentation Workflow Automation Agent**

**TEAM : 12**

**TeamLead: P.Satish – 22B21A4348**

**Teammates:**

**B.Laya 22B21A4306**

**K.Dharma Teja 22B21A4363**

**Y.Murali RaviKumar 22B21A4386**

**K.Sumanth 22B21A4375**

**Department: CSE-AI**

**Institution: Kakinada institute of engineering and technology**

**Academic year: 2025 - 2026**

**Abstract**

In **today’s world, every organization deals with a large number of documents every day. These documents can be in the form of invoices, resumes, contracts, or reports. When this work is done by people manually, it not only takes a lot of time but also increases the chances of errors such as missing information, wrong entries, or delays in approval.**

**This project, called the Documentation Workflow Automation Agent, is designed to make this process easier and faster. The system will automatically read documents, take out the important details, and check them against simple rules. For example, if an invoice is below a certain amount, it can be approved automatically, but if it is above the limit, it will be sent to a manager for review. After checking, the document will be stored in the right place or forwarded to the correct person.**

**The method used in this project is based on automation tools and rule-based techniques such as OCR for reading text from documents and logical conditions for decision-making. The expected result is a system that will reduce manual work, improve accuracy, and save time. By using this approach, organizations can handle documents in a smarter, faster, and more reliable way.**

**Introduction & Problem Statement :**

**Background**

**Companies receive hundreds of documents daily:**

* **Finance departments verify invoices and bills.**
* **HR teams collect and check resumes.**
* **Legal departments review contracts and agreements.**

**Doing all of this manually requires a lot of time and effort.**

**Importance**

**Errors in document handling can cause financial losses, work delays, and legal risks. Automation ensures that documents are processed consistently, faster, and with fewer errors, which improves productivity and reliability.**

**Problem Definition / Objectives**

**The main problem is that manual processing is slow, repetitive, and error-prone.**

**Objectives of the project are:**

* **To automatically read and process different types of documents.**
* **To extract key information such as invoice numbers, amounts, and dates.**
* **To apply simple rules (e.g., approve invoices below a limit, forward larger ones for review).**
* **To store or forward the results safely in an organized way.**

**Proposed Methodology :**

**Approach**

**The project uses a step-by-step approach:**

1. **Document Input: Collect documents in PDF, Word, or Excel formats.**
2. **Data Extraction: Use OCR (Optical Character Recognition), regex, or Excel functions to read text and tables.**
3. **Rule Application: Apply simple business rules.**
   * **Example:**
     + **If invoice amount < ₹50,000 → Auto approve**
     + **If invoice amount ≥ ₹50,000 → Send to manager**
4. **Action: Store details in Excel/database, send email notifications, and archive the documents.**

**Document Input**

**↓**

**Data Extraction (Text/Tables)**

**↓**

**Apply Rules (Approve / Reject / Forward)**

**↓**

**Save / Email / Archive**

**Document Identification and Feature Extraction:**

**Invoices**

* **Identified by keywords like Invoice No., Total Amount, Vendor.**
* **Structured documents → fields: invoice number, date, vendor name, total value.**
* **Extraction uses OCR + regex.**
* **Output: Excel/Database with Invoice No, Date, Vendor, Amount, Status.**

**Resumes**

* **Identified by sections like Education, Skills, Experience.**
* **Semi-structured → fields: name, contact details, education, work experience, skills.**
* **Extraction uses text parsing + NER.**
* **Output: Excel/Database with Name, Email, Phone, Education, Skills, Experience.**

**Contracts**

* **Identified by legal terms like Agreement, Parties, Effective Date.**
* **Unstructured → fields: contract title, parties, start date, duration, terms.**
* **Extraction uses rule-based search + entity extraction.**
* **Output: Excel/Database with Title, Parties, Effective Date, Expiry Date, Terms.**

**Pre-trained Models:**

1. **BERT  
   BERT is used in this project to identify the type of document, such as invoice, resume, or contract. Since it has been trained on a large collection of text, it understands language patterns and context very well. This makes it effective for classifying documents correctly before sending them for further processing.**
2. **spaCy NER  
   spaCy’s Named Entity Recognition (NER) model is applied to extract key details from documents. It can find important fields such as names, dates, amounts, and organizations. The model is already pre-trained, and we can also add our own labels like “Invoice Number” or “Total Amount” to make it more accurate for our specific use.**

**Approach To Build From The Scratch:**

**1. Data gathering**

* **Collected real invoices in both PDF and image formats.**
* **Dataset size ranged between 50 and 100 documents.**
* **Included a variety of layouts and styles from different vendors.**
* **Covered both scanned copies and digital PDFs.**
* **Organized all files systematically in folders with clear and consistent naming conventions.**

**2. Data labeling (annotation)**

* **Use a simple tool: Label Studio or Doccano (easy to learn).**
* **For each document mark the exact text spans or bounding boxes for each field (e.g., highlight the invoice number text and tag it invoice\_no).**
* **Save labels in JSON/JSONL format.**
* **Split labeled data into: Train (70%) / Validation (15%) / Test (15%). Keep the test set unseen.**

**3. Preprocessing & OCR**

* **For scanned or image PDFs, run OCR (Tesseract) to get text and token positions.**
* **Clean text: remove extra whitespace, unify date formats (DD-MM-YYYY → YYYY-MM-DD), remove page headers that repeat.**
* **For each token record: text, page, x1,y1,x2,y2 (bounding box) — useful if you later use layout info.**

**4. Design & build the models (from scratch)**

**You have two main choices:**

**A — Lightweight from-scratch models :**

* **NER with BiLSTM + CRF: classic model for field extraction.**
  + **Convert each token to word embedding (use pre-trained word2vec or train your own).**
  + **Feed embeddings to a BiLSTM layer, then CRF layer on top to predict entity tags (B-INV\_NUM, I-INV\_NUM, O).**
* **Document classifier: simple bag-of-words + logistic regression or small feed-forward network.**

**B — Transformer-based :**

* **Build a small Transformer encoder (like BERT architecture) and train from scratch on your labeled data for NER and classification tasks.**
* **Or initialize from a small pre-trained embedding and continue training—this is hybrid.**

**5. Prepare training data for model**

* **Convert labeled spans into token-level labels (BIO format).**
* **Create vocabulary & embeddings (either pre-trained word2vec/GloVe or learn embeddings during training).**
* **Make train/val/test data loaders (batches of token ids + labels).**

**6. Training steps**

* **Choose hyperparameters:**
  + **learning\_rate = 1e-3 (for BiLSTM)**
  + **batch\_size = 16–32**
  + **epochs = 10–30 (monitor validation)**
* **Train on the train set, evaluate on validation after each epoch.**
* **Save the model that gives best validation F1.**

**7. Evaluation (testing)**

* **Run the saved model on the test set (unseen data).**
* **Report metrics per field: Precision, Recall, F1-score.**
* **Also report overall accuracy for document classification.**

**8. Post-processing & business rules**

* **Normalize extracted fields (convert amounts to numeric, dates to YYYY-MM-DD).**
* **Apply rules:**
* **if total\_amount < 50000 => status = "Auto-Approved"**
* **else => status = "Send-to-Manager"**
* **If model confidence for any field is low (e.g., softmax < 0.7 or CRF score low), put document into a manual review queue.**

**9. Deployment**

* **Wrap model into a REST API using Flask or FastAPI: send document (or OCR text) → get structured JSON response.**
* **Create a small script : uploads docs → calls API → runs rules → saves to Excel / DB and sends email.**

**Conclusion :**

**The Documentation Workflow Automation Agent helps organizations handle documents faster and with fewer errors. By automating tasks like reading invoices, processing resumes, or organizing contracts, the system saves time and reduces manual effort. It improves accuracy, speed, and reliability, making daily work easier and more efficient for different departments. In the long run, this project shows how automation can support smarter business processes and better resource management.**

**References :**

1. **UiPath Documentation – Document Understanding Framework**
2. **Tesseract OCR – Open-source OCR Tool**
3. **Microsoft Excel Automation Features**