Volume: 13 Issue: VI Month of publication: June 2025

DOI: https://doi.org/10.22214/ijraset.2025.72440

Design and Implementation of a Streamlit-based Web Application for Automated Debit Calculation from PDF Bank Statements

Indranil Banerjee[1], Ayush Dutta[2], Parna Debnath[3]
[1]Department of Computer Science Engineering, NSHM Knowledge Campus, Durgapur, West Bengal, India,
[2]Department of Computer Science Engineering, Adamas University, Kolkata, West Bengal, India,
[3]Department of Computer Science Engineering, Adamas University, Kolkata, West Bengal, India

Abstract:** In an era marked by the rapid advancement of digital finance, individuals, freelancers, and small business owners are increasingly reliant on electronic documentation for tracking and managing their financial transactions. Among the most ubiquitous of these documents are bank e-statements, typically issued monthly in the Portable Document Format (PDF). While these digital statements provide a convenient and standardized way of recording financial activity, extracting meaningful insights—such as total debit amounts—often requires tedious manual inspection. For users without access to advanced financial management tools or accounting software, reconciling transactions from such statements can be time-consuming and error-prone. This paper introduces the design and development of a lightweight, web-based application that automates the extraction and summation of debit transactions from bank statements provided in PDF format. Built entirely in Python, the tool leverages three primary open-source libraries: Streamlit for building an interactive and user-friendly interface, pdfplumber for parsing and extracting tabular data from PDF files, and pandas for efficient data manipulation and analysis. The combination of these libraries results in a streamlined, accessible application that requires no programming expertise on the part of the end user. The core functionality of the application revolves around the automatic identification of columns typically labeled "Debit" or "Withdrawals." Upon uploading a password-free bank e-statement, the application parses each page, extracts tables, and converts them into pandas DataFrames. It then scans these tables to detect column headers or data fields related to debit transactions. Only those rows containing valid debit entries—based on keyword recognition and numerical value validation—are retained. The application then computes the cumulative sum of the debit values and displays both the filtered transactions and the total debited amount in a clear and concise format. The motivation behind this solution is rooted in enhancing financial transparency and simplifying daily bookkeeping practices. Whether for tracking personal expenses or preparing monthly ledgers for small enterprises, this tool provides an accessible and cost-effective alternative to complex accounting software. Furthermore, by automating the extraction and calculation process, the risk of human error is significantly reduced, and users gain valuable time that would otherwise be spent combing through pages of transactional data. Another notable feature of this application is its adaptability. Given that bank statement formats can vary significantly between institutions, the implementation includes flexible parsing logic that searches for a variety of keyword patterns associated with debit activity. This ensures that the application remains robust across different formats and layouts, provided that the statements are text-based and contain recognizable table structures.

Keywords:** Debit transactions, bank e-statements, PDF parsing, financial automation, Python, Streamlit, pdfplumber, pandas, personal finance, small business accounting, transaction extraction, digital banking tools, data filtering, financial reconciliation, web-based application.

I. INTRODUCTION

Modern banking has undergone a profound transformation over the past decade, moving decisively from paper-based ledgers and monthly mailed statements to real-time, digital-first transaction records. Financial institutions now routinely issue electronic statements (e-statements) in PDF format, providing customers with convenient, secure access to their account activity through online banking portals or mobile apps. While this shift promises enhanced accessibility and environmental benefits, it has introduced new challenges for end users who wish to derive meaningful insights—particularly total debits, or "money out" figures—from these documents.

Despite their ubiquity, PDF e-statements are not immediately analyzable: their visually formatted layouts, variable table structures, and embedded metadata defeat simple text searches, and each bank's unique styling conventions can frustrate one-size-fits-all parsing routines. Manually reviewing a PDF statement page by page to identify and sum debit transactions is both laborious and error-prone. An accountant reconciling multiple client accounts, a freelancer tracking project expenses, or a small business owner compiling monthly ledgers may spend hours on routine data entry—time better spent on strategic planning, client relationships, or revenue-generating activities. Even tech-savvy users with spreadsheet skills must first convert PDF tables into CSV or Excel formats, clean and standardize columns, and then apply filters or formulas to isolate debit entries. Each of these steps risks data loss, misalignment, or transcription errors, and the process must be repeated for every new statement. Additionally, casual users or those without access to paid conversion and analytics platforms face barriers that discourage regular financial review and undermine budgeting and cash-flow management. Against this backdrop, there is a clear and growing need for lightweight, user-friendly tools that automate the extraction and analysis of key financial metrics from standard e-statements. An ideal solution requires minimal setup, no specialized software purchases, and no manual preprocessing of PDF files. It should intelligently recognize debit-related columns—commonly labeled "Debit," "Withdrawal," or similar—and accurately filter out all relevant entries regardless of minor formatting differences across institutions. Moreover, this tool should present results in an intuitive interface that displays both the raw, filtered transactions and a computed sum of all debit amounts, empowering users with immediate insights into outflows without the overhead of traditional accounting systems. This research addresses these needs by presenting the design and implementation of an intelligent, web-based Python application that automates debit extraction from PDF bank statements. The application harnesses three mature open-source libraries:

- pdfplumber: A robust PDF parsing utility that can detect and extract structured tables directly from the document's internal layout data, bypassing common pitfalls of OCR-based approaches when statements are digitally generated.
- pandas: The de facto standard for data manipulation and analysis in Python, enabling efficient filtering, type conversion, and summation operations on tabular data once extracted.
- Streamlit: A modern framework for rapidly building interactive web apps in pure Python, allowing end users to upload statements, view filtered transactions, and inspect calculated totals in the browser without any front-end development.

By combining these components, the proposed application offers a drag-and-drop workflow: users simply upload their statement PDF, and behind the scenes, the app parses every page, identifies any table structures, converts them into DataFrames, and searches for columns matching a configurable set of debit indicators. Numeric validation ensures that only legitimate monetary values are considered, excluding empty cells, notes columns, or descriptive text. The filtered subset is then displayed in an interactive table, and the cumulative sum of all debit amounts—formatted in the user's preferred currency—is prominently shown. The core contributions of this work are twofold. First, it demonstrates that accurate, format-agnostic extraction of financial outflows from diverse bank statements is achievable with open-source tools, eliminating the need for costly proprietary services. Second, it showcases how a streamlined user interface built entirely in Python can democratize access to such functionality, catering to a broad audience:

- Individual users and finance enthusiasts who wish to monitor personal spending habits, set budget targets, and generate ad hoc reports without piecemeal manual methods.
- Freelancers and gig economy workers who receive payments and incur expenses across multiple banking platforms, requiring quick reconciliation to invoice clients or assess profitability per assignment.
- Small business owners operating lean accounting processes who must produce monthly bookkeeping entries or tax-ready summaries but lack in-house financial software or specialist staff.

Beyond the immediate convenience of automated total debit calculation, the application fosters greater financial transparency. Instead of vague recollections or aggregated monthly figures, users gain detailed visibility into each withdrawal, fund transfer, or debit card purchase. This granularity aids in categorizing expenses, identifying subscription overlaps, and spotting unauthorized transactions promptly. As a result, users can make informed decisions—whether trimming discretionary spending, adjusting pricing strategies, or reallocating budgetary allocations—to optimize cash flow and achieve financial goals. While the current version assumes text-based PDFs without password protection and relies on the presence of table structures, it sets the foundation for future enhancements. Subsequent iterations may integrate OCR engines like Tesseract to handle scanned or image-only statements, incorporate machine-learning classifiers to detect nonstandard debit descriptors, or support multi-currency aggregation for users with international accounts. Additionally, secure cloud deployment with user authentication and encrypted storage could extend its utility in enterprise settings and for regulated fintech solutions.

In summary, this research presents a practical, open-source approach to a common yet underserved challenge in personal and small-business finance. By leveraging Python's ecosystem and focusing on a single, high-value metric—the total amount debited—this web application streamlines recurring accounting tasks, reduces errors, and empowers users with rapid, actionable insights. As digital banking continues to evolve, such automated tools will play an increasingly vital role in translating raw transaction data into strategic financial intelligence.

II. LITERATURE REVIEW

The paper [4] investigates the challenges and potential solutions related to the interoperability of bank statements. Through a case study, the authors likely explore the difficulties arising from disparate formats and standards used by different banking institutions for their statements. The research probably highlights the need for standardized approaches to facilitate easier data exchange and processing for individuals and businesses who may deal with multiple banks. The study may propose a framework or model to enhance interoperability, potentially discussing the benefits such as improved efficiency in financial management and accounting processes. In reference to paper [5], this recent preprint introduces "TabSniper," a system aimed at achieving accurate table detection and structure recognition specifically for bank statements. Given the complex and varied layouts of bank statements, this research likely focuses on developing robust algorithms, possibly leveraging machine learning or deep learning techniques, to identify tabular data and understand its underlying structure (rows, columns, headers). The paper's contribution would be a method to reliably extract transactional data and other key information presented in tables within bank statements, which is crucial for automated data entry, analysis, and downstream financial applications.

The forthcoming journal article [6] explores the application of neural networks to classify bank statement transactions for accounting purposes. The authors likely investigate how machine learning models, specifically neural networks, can automate the process of categorizing financial transactions (e.g., income, expenses, transfers) based on the information present in bank statements. The research would aim to demonstrate the potential of AI to reduce manual effort, improve accuracy, and provide timely insights for accounting and financial reporting. The paper may also discuss the types of neural network architectures best suited for this task and the features extracted from bank statements for classification.

The chapter [7] within the book "Banking Transactions and Services" examines current accounts not just as transactional tools but as key instruments for managing relationships between banks and their customers. Proto likely analyzes how the data generated from current account activity can provide banks with valuable insights into customer behavior, preferences, and financial health. This understanding can then be leveraged to offer personalized services, targeted financial products, and ultimately strengthen customer loyalty and engagement. The paper may discuss strategies for utilizing current account information ethically and effectively for relationship management.

The conference paper [8] presents a knowledge-based approach for recognizing tables within images of Chinese bank statements. The authors likely address the specific challenges posed by the language and layout characteristics of Chinese bank statements. Their method probably incorporates domain-specific knowledge about the common structures and keywords found in these documents to improve the accuracy of table detection and information extraction from scanned or image-based statements. The research contributes to the field of document image analysis, particularly for financial documents in non-Latin scripts.

Similar to Wong & Hanne (2025), the publication [9] also focuses on leveraging artificial intelligence for classifying bank statement transactions to support accounting processes. Lecci and Hanne likely explore various AI techniques, potentially comparing different machine learning algorithms or ensemble methods for this task. The paper might delve into the practical implementation challenges, data preprocessing requirements, and the potential impact of AI-driven classification on the efficiency and accuracy of accounting workflows. It may also offer insights into the interpretability of these AI models in an accounting context.

The paper [10] published in "Pattern Recognition" focuses on the broader problem of table detection in various business document images, with likely applicability to bank statements. The authors propose a method utilizing message passing networks, a type of graph neural network, for this task. This approach likely models the relationships between different elements within a document (e.g., text blocks, lines) to identify and delineate table structures. The research contributes to advancing the state-of-the-art in document analysis by employing sophisticated deep learning techniques for robust table detection in complex layouts.

The paper [11] explores the use of AI-powered Optical Character Recognition (OCR) technology for detecting fraud within FinTech income verification systems, where bank statements are often a key document. Jain likely discusses how advanced OCR can accurately extract data from financial documents and how AI algorithms can then analyze this data for anomalies, inconsistencies, or patterns indicative of fraudulent activity. The research would highlight the role of AI in enhancing the security and reliability of income verification processes, which are critical in lending and other financial services.

The conference proceeding [12] investigates the detection of financial statement fraud specifically within Shariah banks in Indonesia. The authors examine the influence of factors such as political connections, financial stability of the bank, and the effectiveness of monitoring mechanisms in identifying fraudulent activities. While not solely focused on bank statements, this research is relevant as financial statements are a key output that could be manipulated. The paper likely employs statistical analysis or case studies to understand the interplay of these factors in the context of Shariah banking principles and regulatory environments.
The journal article [13] focuses on predicting fraud in financial statements by comparing various data mining techniques. The authors likely evaluate the performance of different algorithms (e.g., classification models, anomaly detection methods) in identifying red flags and predicting the likelihood of fraudulent reporting. Their research would contribute to the development of more effective and proactive fraud detection systems by providing insights into which data mining approaches are most suitable for analyzing financial data and uncovering subtle patterns of fraud.
The journal article [14] provides a broader perspective on how finance technology (FinTech), driven by data science and artificial intelligence, is transforming traditional banking systems. Jindal likely discusses the wide-ranging impact of these technologies, from enhancing customer experience and operational efficiency to improving risk management and enabling new business models. The paper would likely cover areas such as automated financial advice, algorithmic trading, fraud prevention (as seen in other papers), and the use of big data analytics to gain deeper insights into market trends and customer behavior, with bank statement data being a significant input for many of these advancements.

III. METHODOLOGY

The proposed solution employs a clear, modular three-step pipeline—Data Ingestion, Data Extraction & Transformation, and Data Presentation—to automate the calculation of total debit transactions from bank e-statement PDFs. Each stage is defined and implemented to ensure robustness, extensibility, and user-friendliness.

A. Data Ingestion

Data ingestion refers to the process of importing or uploading raw data into a system so that downstream components can operate on it. In the context of our application, it encompasses the reception of PDF files via a user interface and preliminary validation to ensure compatibility.

1. User Interface with Streamlit
    - File Uploader: We leverage Streamlit's st.file_uploader widget to provide an intuitive drag-and-drop area or a browse button for selecting PDF statements.
    - Supported File Types: The uploader is configured to accept only ".pdf" extensions, preventing non-PDF uploads at the source.
    - Unlocked Statements Only: Since the pdfplumber library can only parse non-encrypted PDFs, the interface immediately rejects password-protected or encrypted files. An informative error message guides users to generate an unlocked copy from their bank portal if necessary.
    - Client-Side Preview (Optional): For additional usability, the interface can render a thumbnail of the first page or display basic metadata (number of pages, file size) after upload, reassuring users that the correct document has been received.

B. Data Extraction & Transformation

Data extraction is the process of programmatically retrieving specific information—in this case, tabular transaction data—from a larger document. Transformation refers to the subsequent cleaning, normalization, and preparation of that data for analysis or presentation.

1. Parsing with pdfplumber
    - Page Iteration: The application opens the uploaded PDF with pdfplumber.open() and loops over each page object.
    - Table Detection: On each page, page.extract_tables() is called to detect any embedded tables based on the document's internal layout streams. This approach capitalizes on "born-digital" PDFs, circumventing the need for OCR.
    - Raw Table Structures: Each returned table is a nested list of rows, where the first sublist represents column headers and the subsequent sublists represent row values.

2. DataFrame Construction with pandas
   - Header Assignment: For every raw table, we instantiate a pandas DataFrame by treating the first row as column names.
   - Column Identification: The system scans the header strings to detect any columns whose names include key substrings—specifically "Debit" or "Withdrawal" (case-insensitive). We maintain a small, configurable list of synonyms (e.g., "Debit Amt," "Withdrawals," "Dr") to maximize compatibility.
   - Value Cleaning: Within identified columns, each cell is processed as follows:
     - Strip Formatting: Remove thousands separators (commas, spaces) and currency symbols (₹, $, etc.).
     - Normalize Decimal Marks: Ensure periods (.) are used as decimal separators; replace commas if necessary.
     - Convert to Numeric: Use pandas.to_numeric() with errors='coerce' to cast strings to floats, automatically setting invalid parses to NaN.
   - Row Filtering: Any row where all debit-related columns are NaN or zero is discarded. Valid debit entries—those with a positive numeric amount—are retained for summation.

3. Aggregation
   - Per-Page Summation (Optional): For performance transparency, the application can compute both per-page and overall debit totals, enabling users to trace back aggregated sums to individual pages.
   - Cumulative Total: The final debit total is computed via a simple df[debit_columns].sum().sum(), summing across all identified debit columns and rows.

C. Data Presentation

Data presentation is the delivery of processed information to the end user in an accessible, interpretable format. It includes both tabular displays and summary metrics.

1. Interactive Table Display
   - Streamlit DataFrame Rendering: The filtered DataFrame—containing only rows with valid debit values—is passed to st.dataframe(), which renders it as a scrollable, sortable table in the browser.
   - Column Highlights: To further draw attention, debit columns can be visually emphasized (e.g., bold headers or background shading), guiding the user directly to the critical fields.

2. Summary Metrics
   - Total Debited Amount: Using st.metric(), the cumulative debit sum is displayed prominently, formatted to two decimal places and annotated with the user's preferred currency symbol.
   - Additional Insights (Optional):
     - Average Debit: Calculated via total_debit / number_of_debits, offering users insight into typical transaction size.
     - Transaction Count: Displaying the total number of debits parsed reinforces transparency and helps users verify completeness.

3. Responsiveness and Accessibility
   - Cross-Platform Compatibility: As a lightweight web app, the interface automatically adapts to different screen sizes—desktop browsers, tablets, and even mobile phones—thanks to Streamlit's responsive design.
   - Error Handling and Feedback: If no debit columns are found, or if the uploaded PDF contains no valid debit entries, the app presents a clear warning (st.warning) with suggestions (e.g., checking synonyms, verifying PDF format).
   - User Guidance: A sidebar or tooltip section can provide concise instructions on generating compatible bank statements, with screenshots or links to common bank portals.

By structuring the workflow into ingestion, extraction/transformation, and presentation phases, the application achieves a balance of simplicity and power. Users experience a straightforward upload-and-go process, while developers benefit from clear modular boundaries that facilitate future enhancements—such as OCR integration for scanned statements, support for additional transaction types (credits, balances), or customizable reporting exports (CSV, Excel, PDF). This three-step approach ensures maintainability, extensibility, and, most importantly, an accessible financial automation tool for a wide variety of users.

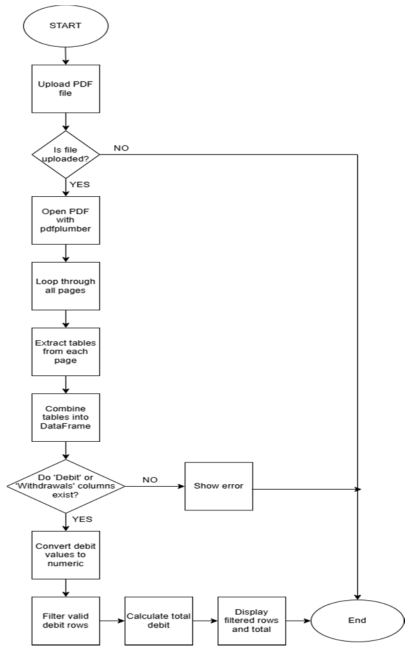
IV. IMPLEMENTATION

The application was developed using a cohesive stack of open-source tools, each selected for its suitability to a specific task in the PDF-to-insight pipeline:

- Python 3.10: A modern, high-level programming language renowned for its readability and extensive ecosystem. Version 3.10 introduces structural pattern matching and other enhancements that streamline data-processing logic.
- pdfplumber: A specialized library that parses "born-digital" PDF files by inspecting their internal layout objects rather than relying on OCR. It exposes tables, text boxes, and positional metadata, enabling precise extraction of tabular transaction data.
- pandas: The leading data-analysis library in Python, pandas provides the DataFrame abstraction—a two-dimensional, mutable table structure with powerful indexing, filtering, and aggregation methods. It underpins the transformation and computation steps of the workflow.
- Streamlit: A rapid-development framework for building interactive web applications purely in Python. Streamlit handles UI rendering, widget events, and responsive design, allowing developers to focus on core functionality without frontend boilerplate.

A. Functional Flow

- File Upload (Data Ingestion): Users interact with a Streamlit file-uploader widget to supply an unlocked PDF bank statement.
- Page-by-Page Parsing (Extraction): The application opens the PDF via pdfplumber and iterates through each page, invoking table-detection routines.
- DataFrame Assembly (Transformation): Extracted tables are converted into pandas DataFrames. These per-page DataFrames are concatenated into a single "master" table for unified processing.
- Column Identification & Standardization: The software scans header labels for the keywords "Debit" and "Withdrawals," cleans formatting (removing commas, currency symbols), and casts entries to numeric types.
- Computation & Presentation: It computes the total of all valid debit entries and renders both the filtered table and the aggregate sum in the Streamlit interface, offering users immediate, transparent insight into their outflows.


START
Upload PDF file
Is file uploaded?
NO
YES
Open PDF with pdfplumber
Loop through all pages
Extract tables from each page
Combine tables into DataFrame
Do 'Debit' or 'Withdrawals' columns exist?
NO
Show error
YES
Convert debit values to numeric
Filter valid debit rows
Calculate total debit
Display filtered rows and total
End
Figure: Flowchart of the model

V. RESULTS AND DISCUSSION

The application was tested extensively using a diverse set of bank statements obtained from multiple banking institutions, each with its own formatting style, table structure, and column labeling conventions. These statements varied in terms of page layout, table column order, date formatting, and numerical representations (e.g., comma separators, currency symbols). This variety was intentional and necessary to assess the robustness and adaptability of the parsing logic under real-world conditions. During the evaluation phase, the application demonstrated strong performance in detecting and summing debit transactions from PDF bank statements that adhered to standard digital formatting practices. In particular, the system proved effective when the columns containing debit-related amounts were clearly labeled using terms like “Debit,” “Withdrawals,” or similar recognizable synonyms. The flexible logic implemented using keyword matching and case-insensitive search allowed the application to detect relevant columns even when minor variations in naming existed, such as “Debit Amt,” “Withdrawal Amount,” or “Dr.” This adaptability significantly improved the application's compatibility across different bank formats. The use of both “Debit” and “Withdrawals” as primary search terms proved particularly beneficial, as some banks use “Debit” to refer to transactions while others prefer “Withdrawals.” By including both terms in the search logic, the application achieved better coverage without requiring custom configurations for each bank. Furthermore, the filtered outputs and final calculated debit totals were consistent with manual calculations, confirming the computational accuracy of the system. However, several limitations were also identified during testing. First, the application currently does not support password-protected PDFs. As many banks automatically encrypt their statements for security purposes, this limitation could prevent some users from processing their files unless they manually remove the password protection beforehand. Second, the system does not support scanned (image-based) bank statements, which often appear as rasterized pages without embedded text. In such cases, pdfplumber fails to detect any extractable tables because there is no underlying machine-readable structure.

To address these shortcomings, future development could include integration of Optical Character Recognition (OCR) capabilities using tools like Tesseract to process scanned or image-based documents. Additionally, implementing an optional password input feature would allow users to unlock encrypted PDFs within the application, broadening accessibility and making the tool more versatile in practical use scenarios.

VI. CONCLUSION

This study illustrates the feasibility of constructing an efficient, lightweight, and user-friendly web application leveraging the Python ecosystem and open-source libraries to automate the extraction and computation of debit transactions from PDF bank statements. By integrating pdfplumber for precise table parsing, pandas for robust data manipulation, and Streamlit for rapid deployment of an interactive interface, the application eliminates manual spreadsheet entry and accelerates financial review workflows. Users benefit from immediate visibility into their spending patterns, as the tool not only calculates the aggregate of debit and withdrawal entries but also displays each relevant transaction, thereby enhancing transparency and auditability. Moreover, the modular architecture supports future extensibility: incorporating OCR engines such as Tesseract would enable processing of scanned or image-based PDFs; embedding secure password prompts can accommodate encrypted statements; and developing RESTful APIs could facilitate seamless integration with existing accounting platforms. Additional enhancements—such as interactive data visualizations, export-to-Excel functionality, and multi-currency support—would further broaden the tool’s applicability to diverse financial contexts. In sum, this web application represents a foundational step toward a comprehensive, automated financial management solution, providing both immediate utility for end users and a flexible framework for ongoing innovation.

REFERENCES

[1] Streamlit Documentation. (https://docs.streamlit.io)
[2] pdfplumber GitHub Repository. (https://github.com/jsvine/pdfplumber)
[3] pandas Documentation. (https://pandas.pydata.org/docs/)
[4] Chejkova-Nikolov, R., Gusev, M., Kostoska, M., & Ristov, S. (2015, May). Interoperablity of bank statements: A case study. In 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1505-1510). IEEE.
[5] Trivedi, A., Mukherjee, S., Singh, R. K., Agarwal, V., Ramakrishnan, S., & Bhatt, H. S. (2024). TabSniper: Towards Accurate Table Detection & Structure Recognition for Bank Statements. arXiv preprint arXiv:2412.12827.
[6] Wong, K., & Hanne, T. (2025). Support of Accounting by Bank Statement Classification Using Neural Networks. Journal of Emerging Technologies in Accounting, 22(1), 137-152.
[7] Proto, A. (2023). Current Accounts as a Tool for Bank and Customer Relationships Management. In Banking Transactions and Services (Vol. 1, pp. 17-28). Giappichelli Editore.

[8] Xu, L., Fan, W., Sun, J., Li, X., & Naoi, S. (2016, September). A knowledge-based table recognition method for Chinese bank statement images. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3279-3283). IEEE.
[9] Lecci, M., & Hanne, T. (2025). Accounting Support Using Artificial Intelligence for Bank Statement Classification. Computers, 14(5), 193.
[10] Riba, P., Goldmann, L., Terrades, O. R., Rusticus, D., Fornés, A., & Lladós, J. (2022). Table detection in business document images by message passing networks. Pattern Recognition, 127, 108641.
[11] Jain, J. (2024). AI-Driven Optical Character Recognition for Fraud Detection in FinTech Income Verification Systems.
[12] Arinta, Y. N., Rahman, T., & Khilmiyah, I. (2024, September). DETECTION FINANCIAL STATEMENT FRAUD SHARIAH BANK IN INDONESIA: ROLE OF POLITIC CONNECTION, FINANCIAL STABILITY, IN EFECTIVE MONITORING. In Proceedings of the International Conference of Islamic Economics and Business (ICONIES) (Vol. 10, No. 1, pp. 1317-1328).
[13] Nemati, Z., Mohammadi, A., Bayat, A., & Mirzaei, A. (2025). Fraud Prediction in Financial Statements through Comparative Analysis of Data Mining Methods. International Journal of Finance & Managerial Accounting, 10(38), 151-166.
[14] Jindal, G. (2024). The role of finance tech in revolutionizing traditional banking systems through data science and AI. Journal Of Applied Sciences, 4(11), 10-21.