International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

AI-based functional expense classification for nonprofit financial reporting and form 990 compliance

Ishrak Alim [1,*], Nure Washik Elahe [2], Hasin Ishrak [3] and Tasnia Farzana Matin [4]

[1] *University of New Haven, Connecticut, United States.*
[2] *Bangor University, Wales, United Kingdom.*
[3] *Arkansas State University, Arkansas, United States.*
[4] *Autism Society Habilitation Organization (ASHO), New York, United States*

International Journal of Science and Research Archive, 2026, 20(01), 158–182

Publication history: Received on 26 May 2026; revised on 05 July 2026; accepted on 07 July 2026

Article DOI: https://doi.org/10.30574/ijsra.2026.20.1.1437

Abstract

Preparing a Statement of Functional Expenses is an important but often time-consuming task for nonprofit organizations. Accountants must classify expenses not only by type, such as salaries, rent, supplies, and professional fees, but also by purpose, including Program Services, Management and General, and Fundraising. In practice, this process often depends on manual review, spreadsheet workpapers, allocation percentages, payroll records, class codes, grant codes, and supporting documents. As a result, nonprofits may face inconsistent classification, weak documentation, reporting delays, and reconciliation problems.

This study proposes an AI-based functional expense report automation framework for nonprofit financial reporting and Form 990 compliance. The framework uses accounting data from Profit and Loss Detail or General Ledger reports, along with supporting inputs such as vendor information, payroll records, memo descriptions, invoice text, class codes, grant codes, historical classifications, and editable allocation rules. The system classifies direct expenses, allocates shared costs, generates confidence scores, explains the basis for each classification, reconciles totals to accounting records, and produces a draft Statement of Functional Expenses for accountant review.

The framework is not designed to replace professional judgment. Instead, it supports accountants by reducing manual workload, improving consistency, strengthening audit documentation, and identifying transactions that require human review. By combining automation, explainable classification logic, editable allocation percentages, and reconciliation controls, the proposed model offers a practical solution for improving nonprofit reporting accuracy, Form 990 preparation, and transparency for donors, grantors, auditors, regulators, and the public.

Keywords: Artificial Intelligence; Nonprofit Accounting; Functional Expense Report Automation; Statement of Functional Expenses; Form 990 Compliance; Expense Allocation; Program Services; Management and General Expenses; Fundraising Expenses; Accounting Analytics; Financial Reporting Automation; Explainable AI

1. Introduction

Nonprofit organizations operate in a reporting environment where financial transparency is directly connected to public trust. Unlike for-profit entities, nonprofits are often evaluated by how effectively they use resources to support their mission. Donors, grantors, auditors, regulators, board members, and the public rely on nonprofit financial reports to understand whether funds are being used for program services, administration, or fundraising. For this reason, accurate and timely expense classification is an important part of nonprofit accountability.

* Corresponding author: Ishrak Alim

A key report in this area is the Statement of Functional Expenses. This report presents expenses by natural classification, such as salaries, rent, supplies, insurance, and professional fees, and by functional purpose, such as Program Services, Management and General, and Fundraising. FASB guidance requires nonprofit organizations to present an analysis of expenses by both nature and function, either in the financial statements or related notes.[1] This makes functional expense reporting a central part of nonprofit financial presentation.

Functional expense reporting is also connected to Form 990 compliance. Form 990 is an annual information return filed by many tax-exempt organizations, and Part IX of the form presents the Statement of Functional Expenses.[2] Because Form 990 is publicly available, expense classification affects how donors, grantors, watchdog groups, regulators, and the public interpret a nonprofit's efficiency, governance, and financial responsibility.

Although reporting guidance exists, preparing the Statement of Functional Expenses can be difficult and time-consuming. Many nonprofits prepare the report manually by using Profit and Loss reports, General Ledger details, payroll allocations, class codes, grant codes, vendor information, invoice descriptions, and spreadsheets. This process can create inconsistencies, especially when transactions have vague descriptions, expenses support multiple functions, or payroll and shared costs must be allocated across several categories.

These challenges create an opportunity for AI-assisted report automation. An AI-based system can import accounting data, classify expenses by function, allocate shared costs, generate confidence scores, explain classification decisions, reconcile totals to the Profit and Loss report, and produce a draft Statement of Functional Expenses for accountant review. The purpose of this study is not to replace professional judgment, but to propose a practical decision-support framework that can improve consistency, reduce manual preparation time, strengthen documentation, and support Form 990 compliance.

2. Functional Expense Reporting in Nonprofit Organizations

Functional expense reporting classifies nonprofit expenses based on the purpose they serve. The three major functional categories are Program Services, Management and General, and Fundraising. Program Services represent activities that directly support the nonprofit's mission. Management and General expenses relate to administration, governance, accounting, compliance, and overall organizational support. Fundraising expenses are connected to activities that help the organization raise donations, grants, sponsorships, or other contributions.

Table 1 Functional Expense Categories and Common Examples

Functional Category	Meaning	Common Examples
Program Services	Expenses directly related to carrying out the nonprofit's mission	Program staff salaries, client services, educational materials, program supplies, direct service costs, program-related rent or utilities
Management and General	Expenses related to administration, governance, compliance, and organizational support	Accounting fees, legal fees, insurance, office administration, board meeting costs, audit costs, general management salaries
Fundraising	Expenses related to raising donations, grants, sponsorships, or contributions	Fundraising event costs, donor campaign expenses, fundraising staff salaries, donor communication, grant proposal costs, fundraising software

Nonprofit organizations must report expenses by both natural classification and functional purpose. Natural classification explains what type of expense was incurred, such as salaries, rent, supplies, professional fees, or insurance. Functional classification explains why the expense was incurred, meaning whether it supported program delivery, administration, or fundraising. This dual presentation is important because it helps users understand not only what the organization spent, but also how those expenses supported its mission and operations.[3]

In practice, some expenses are easy to classify. For example, supplies used directly for a youth education program may be classified as Program Services, while costs related to a donor fundraising event may be classified as Fundraising. Other expenses are more complex because they support more than one function. Payroll, rent, utilities, software,

insurance, and professional fees may need to be allocated across multiple categories using a reasonable and consistent method.[4]

A well-designed functional expense report should therefore do more than list expenses. It should show a clear connection between accounting transactions, functional categories, allocation methods, and final report totals. This is where automation can be useful. If a nonprofit's accounting system contains transaction-level details, class codes, grant codes, vendor names, memos, and historical classifications, an AI-assisted framework can help convert that data into a structured draft report.

3. Problems in Traditional Functional Expense Report Preparation

Traditional functional expense report preparation is often manual, time-consuming, and dependent on spreadsheet-based workpapers. Many nonprofits begin with a Profit and Loss report or General Ledger detail, then manually review transactions, assign functional categories, apply allocation percentages, and reconcile the final report to accounting records. While this process can work, it becomes difficult when the organization has multiple programs, restricted grants, shared payroll costs, recurring vendors, and large transaction volumes.

One common problem is inconsistent coding. The same type of expense may be classified differently across months, programs, preparers, or fiscal years. For example, a vendor that provides both program and administrative services may be coded one way in one period and differently in another. Without a structured method, classification can depend heavily on individual judgment rather than consistent evidence.

Vague transaction descriptions create another challenge. Accounting records may include descriptions such as "services," "supplies," "consulting," or "reimbursement," without explaining the actual purpose of the expense. A general ledger account may show the natural category, but it may not show whether the expense supported program work, management and general activities, or fundraising. This weakens the reliability of the final functional expense report.

Mixed-purpose expenses are also difficult to handle manually. Payroll, rent, utilities, insurance, software, and professional fees may support several functions at once. These costs require allocation based on time records, square footage, class codes, department usage, grant budgets, or management-approved percentages. If the allocation method is not documented clearly, the report may be difficult to support during audit or Form 990 preparation.

Another practical problem is reconciliation. After expenses are classified and allocated, the total functional expense report must agree with the organization's Profit and Loss report or General Ledger totals. Manual spreadsheets increase the risk of missing transactions, duplicate entries, formula errors, or unexplained differences. These issues can delay reporting, increase audit questions, and reduce confidence in the financial information.

These problems show that the issue is not simply classification. The larger problem is report generation. Nonprofits need a process that can take accounting data, classify and allocate expenses, explain the reasoning, preserve an audit trail, and reconcile the output to the original accounting records.

4. Research Gap and Study Objective

Existing nonprofit reporting guidance explains the need to report expenses by both nature and function, but many nonprofits still lack a consistent technology-supported method for preparing the Statement of Functional Expenses. Current practice often depends on manual review, spreadsheet-based allocations, staff judgment, and historical coding patterns. These methods may be manageable for small organizations, but they become less reliable as transaction volume, program complexity, and grant restrictions increase.

The research gap is the limited development of an AI-assisted system that can automate the preparation of a functional expense report from accounting data. Prior work in accounting automation often focuses on transaction processing, audit analytics, fraud detection, or financial forecasting. Less attention has been given to how AI can help nonprofits classify expenses, allocate shared costs, explain classification decisions, reconcile totals to the Profit and Loss report, and generate a draft Statement of Functional Expenses for review.

This gap is important because functional expense reporting affects Form 990 accuracy, audit documentation, donor transparency, grant compliance, and public accountability. A useful AI framework should therefore do more than suggest whether an expense is Program Services, Management and General, or Fundraising. It should also support

allocation percentages, allow user-defined functional categories, provide confidence scores, explain the reason for each classification, and create a reviewable audit trail.

The objective of this study is to propose an AI-based functional expense report automation framework for nonprofit financial reporting and Form 990 compliance. The framework is designed to import data from reports such as Profit and Loss Detail, General Ledger, payroll records, vendor lists, invoice descriptions, class or department codes, grant or project codes, and historical classification files. It then classifies direct expenses, allocates shared expenses, generates explanations, reconciles totals to accounting records, and produces a draft functional expense report for accountant review.

The proposed framework follows a responsible AI approach. It does not replace accountants, auditors, or professional judgment. Instead, it functions as a decision-support system that improves consistency, reduces manual workload, strengthens documentation, and supports human review. This design is consistent with AI risk management principles that emphasize transparency, reliability, accountability, and human oversight.[5] It also follows the logic of structured data-mining processes, where business understanding, data preparation, modeling, evaluation, and deployment are connected to a practical reporting outcome.[6]

This study contributes to nonprofit accounting analytics by shifting the focus from simple classification to full report automation. The expected contribution is a practical framework that can help nonprofits generate functional expense reports faster, reduce reporting errors, improve Form 990 preparation, support audit readiness, and increase transparency for donors, grantors, regulators, and the public.

5. Proposed AI-Based Functional Expense Report Automation Framework

This study proposes an AI-based functional expense report automation framework that moves beyond simple expense classification and focuses on generating a draft Statement of Functional Expenses from nonprofit accounting data. The framework is designed to import accounting reports, classify expenses by function, allocate shared costs, produce confidence scores, explain classification decisions, reconcile totals to the Profit and Loss report, and generate a draft functional expense report for accountant review.

The central purpose of the framework is to reduce the manual work involved in preparing functional expense reports while improving consistency, documentation, and reviewability. Nonprofit organizations are required to report expenses by both natural classification and functional purpose, which means that expense categories such as salaries, rent, supplies, insurance, and professional fees must be distributed across program services, management and general, and fundraising.[1 7] In practice, this process often requires accountants to review transaction details, payroll records, class codes, grant codes, vendor descriptions, and allocation schedules. The proposed framework automates much of this process while keeping the final decision under human review.

The model begins by importing accounting data from reports such as the Profit and Loss Detail, General Ledger, Transaction Detail by Account, chart of accounts, vendor list, payroll records, invoice text, and historical classification files. The Profit and Loss Detail or General Ledger is the primary source because it contains the expense transactions that must be classified. Balance Sheet data is not the main input, but it may support limited items such as prepaid expenses, accrued expenses, depreciation, or other adjustments that affect expense recognition.

After importing the data, the model cleans and standardizes the records. This includes normalizing vendor names, mapping general ledger accounts to natural expense categories, reading memo and invoice descriptions, identifying department or class codes, matching grant or project codes, and detecting prior-year classification patterns. This step is important because nonprofit accounting systems often contain inconsistent descriptions, incomplete memos, and different coding practices across programs or departments.

The next step is functional classification. The AI model evaluates each transaction and suggests whether it belongs to program services, management and general, or fundraising. The classification logic uses a hybrid approach. Rule-based accounting logic handles clear cases, such as audit fees being management and general or fundraising event costs being fundraising. Natural language processing helps interpret memo fields and invoice descriptions, especially when the purpose of the expense is written in unstructured text.[8] Historical pattern recognition helps maintain consistency by comparing current transactions with how similar expenses were classified in prior periods.

The framework also addresses shared expenses. Some expenses are directly assignable to one function, but others must be allocated. For example, payroll, rent, utilities, insurance, software, office supplies, and professional fees may support

multiple functions. The proposed system allows users to define allocation percentages, such as 70% program, 20% management and general, and 10% fundraising. These percentages can be based on time records, square footage, class codes, grant budgets, employee roles, or management-approved allocation policies. This makes the model flexible enough for different nonprofit structures.

For each transaction, the model produces three outputs: a suggested functional category or allocation split, a confidence score, and an explanation. A high-confidence item may be classified automatically for quick review when multiple indicators agree. A low-confidence or conflicting item is routed to manual review. This design follows responsible AI principles because the model supports the accounting process but does not replace professional judgment.[9]

Finally, the model generates a draft Statement of Functional Expenses. The report is presented in a matrix format where natural expense categories appear in rows and functional categories appear in columns. This structure allows the organization to compare total expenses by nature with total expenses by function. The report should reconcile back to the Profit and Loss report so that accountants can confirm that the functional expense report includes all relevant expenses and does not create unexplained differences.

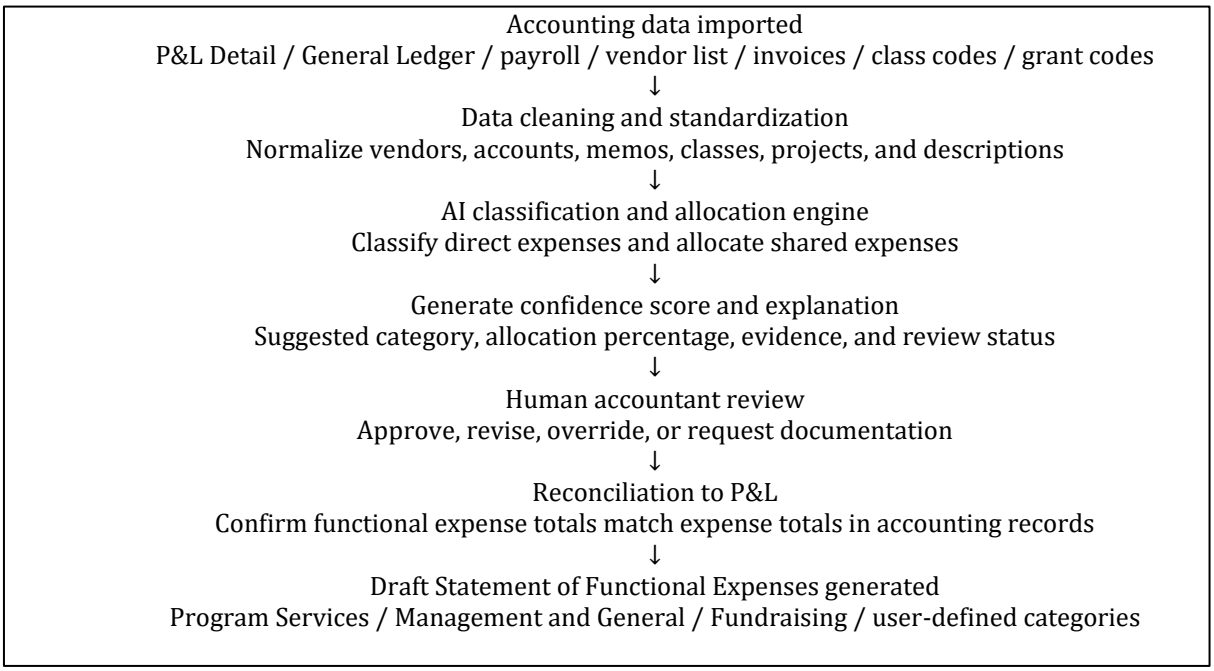

Figure 1 Proposed AI Workflow for Automated Functional Expense Report Generation

6. Data Inputs Required for Automated Report Generation

The proposed automation framework requires accounting data that nonprofit organizations already maintain in their bookkeeping or financial reporting systems. The most important input is the Profit and Loss Detail or General Ledger because these reports provide the transaction-level expense data that must be converted into functional categories. Each record should include the transaction date, vendor name, general ledger account, amount, memo or description, class or department, project or grant code, and any available supporting document reference.

The chart of accounts is also necessary because it identifies the natural classification of each expense. For example, salaries, rent, insurance, office supplies, professional fees, travel, and program materials are natural expense categories. The AI model uses these natural categories as rows in the final functional expense report. This is consistent with nonprofit reporting practice, where expenses are commonly presented by nature and function in a matrix format.[7]

Vendor data provides another important input. Vendor names can help the model identify likely expense purpose, especially when the same vendor is repeatedly associated with a specific function. For example, an audit firm may usually relate to management and general, while a program supply vendor may relate to program services. However, vendor name alone is not sufficient because the same vendor can support multiple functions. Therefore, vendor data must be evaluated together with account, memo, invoice, class, project, and historical classification data.

Payroll records are especially important because personnel costs are often the largest expense category for nonprofits. Payroll may need to be allocated across program services, management and general, and fundraising based on employee role, time records, department assignment, grant funding, or management-approved allocation percentages. Without payroll allocation data, the functional expense report may not accurately reflect how staff time supports different functions.

Invoice text and memo descriptions help explain the purpose of the expense. These fields are often unstructured, but they may include useful phrases such as "youth program supplies," "donor outreach," "audit preparation," "grant-funded training," or "board meeting." Natural language processing can help convert this text into classification signals.[8]

Historical classifications are also useful because they help maintain consistency across reporting periods. If a similar transaction from the same vendor, account, department, and memo description was classified as program services in prior periods, the model can use that pattern as supporting evidence. However, historical classifications should not be accepted automatically because prior coding may contain errors. The model should use historical treatment as one signal, not as the only basis for classification.

Table 2 Required Data Inputs for AI-Based Functional Expense Report Automation

Data Input	Example Fields	Purpose in the Automation Model
Profit and Loss Detail / General Ledger	Date, transaction type, account, vendor, amount, memo, class, project	Main source of expense transactions to be classified
Chart of Accounts	Account number, account name, natural expense category	Maps expenses into natural classifications for report rows
Vendor List	Vendor name, vendor type, prior classification history	Helps identify recurring vendors and likely functional purpose
Transaction Memo / Description	Memo text, internal notes, transaction description	Provides text evidence for program, management, or fundraising purpose
Invoice Text / Supporting Documents	Invoice description, service details, project name, event name	Strengthens classification evidence and audit documentation
Payroll Records	Employee role, department, time allocation, salary, benefits	Supports allocation of salaries and benefits across functions
Class / Department Codes	Program, administration, development, fundraising	Links transactions to internal functional responsibility
Grant / Project Codes	Restricted grant, program project, campaign code	Connects expenses to program or funding purpose
Historical Classification File	Prior-year category, prior allocation percentage, reviewer notes	Improves consistency and highlights unusual changes
Allocation Policy Table	Allocation basis, percentage split, effective date, approval status	Applies management-approved allocation percentages to shared costs

7. AI Classification and Allocation Logic

The proposed AI model classifies each expense by evaluating whether the transaction is directly assignable to one function or should be allocated across multiple functions. Direct classification applies when the evidence clearly supports one category. For example, a transaction coded to a youth program grant with an invoice description for program materials may be classified as program services. A donor campaign expense may be classified as fundraising. Audit fees, board meeting costs, and general accounting services may be classified as management and general.

Shared expenses require allocation rather than single-category classification. Many nonprofit expenses support more than one function, including rent, utilities, insurance, office software, telephone, payroll, and administrative support. The model should therefore allow users to define allocation percentages. For example, rent may be allocated based on

square footage, payroll may be allocated based on employee time records, and software costs may be allocated based on department usage. These allocation rules should be documented and applied consistently.

The classification logic should be flexible enough to support both the three standard functional categories and additional user-defined categories. The three primary categories are program services, management and general, and fundraising. However, some organizations may want to break program services into several program columns, such as Youth Program, Healthcare Program, Education Program, or Community Outreach. The proposed model should allow the nonprofit to add, remove, or rename functional categories while still reconciling the total functional expenses to the total expense amount in the accounting records.

A practical AI-assisted model can use three layers of decision logic. The first layer is rule-based mapping. This applies clear accounting rules and organization-approved allocation policies. The second layer is text-based classification, where the model uses memo descriptions and invoice text to identify functional purpose. The third layer is historical pattern recognition, where the model compares the current transaction with prior classifications. When all three layers agree, the confidence score should be high. When the signals conflict, the transaction should be sent for accountant review.

The system should also produce an explanation for each classification or allocation. For example, the explanation may state that an expense was classified as program services because it used a program class code, a restricted grant code, and invoice text related to direct service delivery. For a shared expense, the explanation may state that the transaction was allocated 70% to program services, 20% to management and general, and 10% to fundraising based on the approved rent allocation policy. This explanation supports audit review and helps users understand the reasoning behind the AI recommendation.

Table 3 Classification and Allocation Rules Used by the AI Model

Expense Type	Example Transaction	Suggested Treatment	Decision Logic
Direct program expense	Program supplies for youth workshop	Program Services	Program class code, program memo, grant/project code
Administrative expense	Audit fee, legal fee, insurance, board meeting cost	Management and General	Account type and vendor purpose indicate governance or administration
Fundraising expense	Donor event, campaign design, fundraising software	Fundraising	Memo or invoice text indicates donor solicitation or fundraising activity
Shared payroll	Employee works across program and administration	Allocate by percentage	Allocation based on time records, role, or approved payroll allocation
Shared rent or utilities	Office space used by program and administrative staff	Allocate by percentage	Allocation based on square footage, department usage, or approved policy
Shared software	Accounting, CRM, or communication platform used across departments	Allocate by usage or policy	Allocation based on user count, department, or management-approved split
Unclear transaction	Vague memo such as “services” or “supplies”	Manual review	Low confidence due to insufficient description or conflicting indicators
Historical inconsistency	Same vendor classified differently from prior period	Review required	Model flags unusual change for accountant review

8. Automated Functional Expense Report Generation Process

The automated report generation process begins with data extraction. The nonprofit exports transaction-level accounting data from its accounting system, such as a P&L Detail, General Ledger, or Transaction Detail by Account.

Payroll records, vendor files, class codes, grant codes, invoice descriptions, and historical classification files may also be imported. The objective is to gather enough evidence to classify each expense by function and reconcile the output to the organization's accounting records.

The second step is data cleaning. The system standardizes vendor names, account names, date formats, amounts, class codes, project codes, and memo fields. It removes duplicate records, flags missing descriptions, and identifies transactions that do not contain enough information for reliable classification. This step is necessary because poor data quality can reduce the reliability of the AI-generated report.

The third step is classification and allocation. The model applies rules, text analysis, historical patterns, and allocation policies to each expense transaction. Direct expenses are assigned to a functional category. Shared expenses are split across functional categories using approved percentages. The framework should allow accountants to manage percentages manually, such as 80% program and 20% management, or define allocation rules by vendor, account, department, project, or employee role.

The fourth step is confidence scoring and explanation generation. Each transaction receives a confidence score based on the strength and agreement of the available evidence. A high-confidence transaction may have consistent account, memo, class, and historical treatment. A low-confidence transaction may have vague text, missing class codes, conflicting prior treatment, or no supporting document. The explanation identifies the main reason for the recommendation so the accountant can review it efficiently.

The fifth step is human review. The accountant reviews all low-confidence and conflicting transactions, approves or edits the suggested categories, changes allocation percentages if necessary, and documents any override. This step is important because functional expense classification remains a professional accounting judgment, especially when expenses support multiple functions.

The sixth step is reconciliation. The system compares the total expenses in the generated functional expense report with the total expenses in the P&L Detail or General Ledger. Any difference should be flagged for review. This reconciliation control is important because the final functional expense report should not omit expenses, duplicate expenses, or produce totals that do not agree with accounting records.

The final step is report output. After review and reconciliation, the model generates a draft Statement of Functional Expenses. The output should show natural expense categories in rows and functional categories in columns. The report can be exported to Excel, PDF, or a Form 990 workpaper format. The system should also preserve a detailed classification log showing the original transaction, AI suggestion, confidence score, explanation, reviewer decision, final classification, and allocation basis.

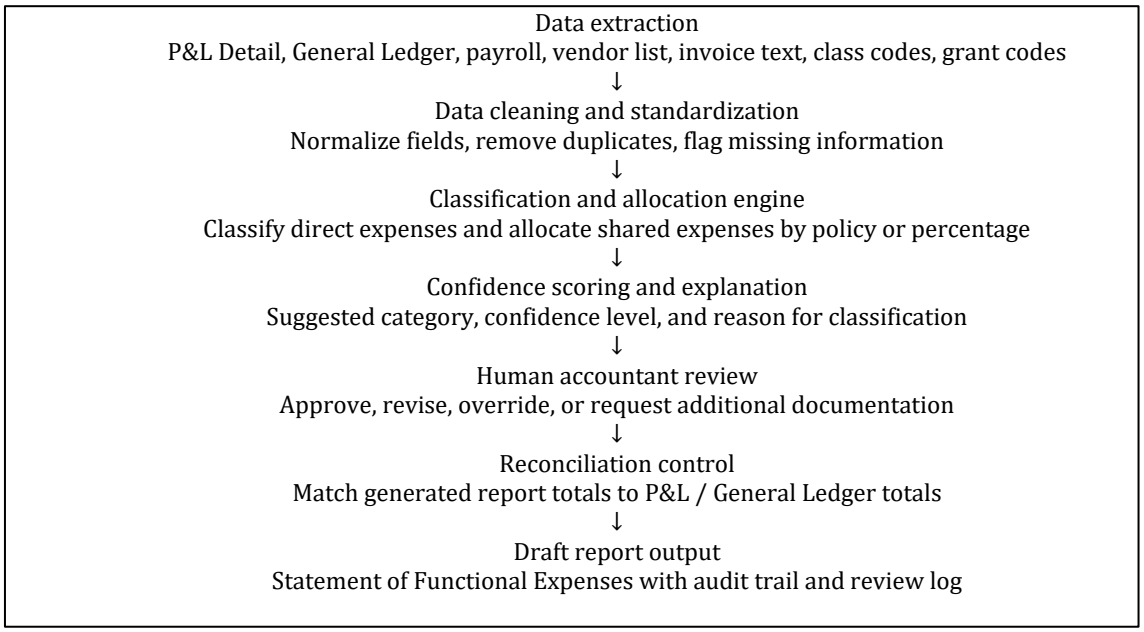

Figure 2 AI-Based Functional Expense Report Generation Pipeline

9. Confidence Score, Human Review, and Compliance Control

The proposed automation framework should not allow AI to make the final functional expense decision independently. Functional expense reporting involves professional judgment, documentation, and compliance responsibility. Although AI can classify transactions faster than manual review, the final classification should remain subject to accountant review because nonprofit expenses may support more than one function and may require context that is not fully captured in the accounting data.

For each transaction, the system should provide four outputs: a suggested functional category, a confidence score, an explanation, and a review status. The suggested category identifies whether the expense should be reported as Program Services, Management and General, Fundraising, or another user-defined functional category. The confidence score shows how strongly the available evidence supports that recommendation. The explanation identifies the main factors behind the classification, such as the general ledger account, vendor name, class code, grant code, memo description, invoice text, or historical treatment. The review status tells the accountant whether the item can be reviewed quickly, requires additional support, or should be manually reviewed before reporting.

A high-confidence classification may occur when several indicators point to the same result. For example, a transaction may be classified as Program Services when the account is program supplies, the class code is a program department, the grant code relates to a restricted program, and the invoice description refers to direct service activity. A lower-confidence classification may occur when the transaction has a vague memo, missing class code, conflicting prior treatment, or a vendor that supports multiple functions.

Human review is essential for compliance control. The accountant should be able to accept the AI recommendation, revise the classification, change the allocation percentage, request additional documentation, or override the recommendation with a written explanation. This creates an audit trail showing how the final classification was determined. Responsible AI guidance emphasizes transparency, accountability, reliability, and human oversight, especially when AI supports decisions that affect organizational reporting and compliance.[14]

The confidence score also helps prioritize review time. Instead of reviewing every transaction with the same level of attention, accountants can focus on low-confidence and conflicting items. This makes the reporting process more efficient while preserving professional judgment. The goal is not to remove human responsibility, but to make the review process faster, more consistent, and better documented.

AI Confidence Score	Suggested Review Action	Compliance Purpose
85–100%	Accept classification after quick review	Maintains efficiency when evidence is strong and consistent
60–84%	Accountant reviews supporting details	Confirms classification when evidence is reasonable but not complete
Below 60%	Manual review required	Prevents weak or unsupported classification from entering the report
Conflicting signals	Supervisor or accountant approval required	Ensures professional judgment when data indicators disagree

Figure 3 AI Confidence Score and Human Review Decision Matrix

10. Prototype Design and Pseudocode for Report Automation

The proposed framework can be implemented as a practical prototype using spreadsheet-based accounting exports and Python-supported data processing. This design is suitable for nonprofit organizations because many nonprofits already prepare financial reports in Excel and rely on exported reports from bookkeeping systems. Modern spreadsheet tools

now support more advanced data analysis and automation. Microsoft describes Python in Excel as a feature that allows users to apply Python-based analysis directly within Excel, while Copilot in Excel can assist users with analyzing data, generating formulas, and working with structured datasets.[15] [16] For small and mid-sized nonprofits that may not have advanced reporting software, an Excel-based prototype can therefore provide a realistic and accessible automation pathway.

The prototype begins with exported accounting data. The primary input should be the Profit and Loss Detail, General Ledger, or Transaction Detail by Account report because these reports contain the expense activity that must be classified by function. Additional inputs may include the chart of accounts, vendor list, payroll records, invoice descriptions, class or department codes, grant or project codes, historical classifications, and an allocation policy table. These files allow the model to connect each expense transaction with its natural category, functional purpose, supporting documentation, and prior treatment.

The Balance Sheet should not be treated as the primary source for functional expense reporting because it does not normally show expense activity by function. However, Balance Sheet-related information may be useful as an adjustment file when it affects the final expense report. For example, depreciation, prepaid expense amortization, accrued expenses, or reclassification entries may need to be added as adjusting items. In this prototype, the Balance Sheet is therefore used only to support adjustment entries, while the Profit and Loss Detail or General Ledger remains the main source for expense classification and report generation.

The technical design is intentionally simple so that nonprofit accounting teams can understand and review the process. First, the system imports the accounting data. Second, it cleans the data by standardizing vendor names, account names, dates, amounts, memo descriptions, class codes, and project codes. Third, it applies classification rules to expenses that can be directly assigned to one functional category. Fourth, it uses text-based analysis to interpret memo fields and invoice descriptions. Fifth, it applies allocation rules to shared expenses such as payroll, rent, utilities, insurance, and software. Sixth, it generates a confidence score and explanation for each classification. Finally, it exports a draft Statement of Functional Expenses, a classification log, a review list, and a reconciliation summary.

The prototype should also allow users to manage allocation percentages. For example, the organization may decide that rent should be allocated 70% to Program Services, 20% to Management and General, and 10% to Fundraising. Payroll may be allocated based on time records or employee roles, while software or utility costs may be allocated based on department usage or an approved internal policy. These percentages should be editable in the allocation rule table so that management can update the model when programs, staffing, or operating conditions change.

Although some nonprofits may choose to add separate program columns, the basic version of the prototype uses only three functional expense categories: Program Services, Management and General, and Fundraising. This keeps the model aligned with the standard functional expense presentation used in nonprofit reporting and Form 990 workpapers. If needed, future versions of the prototype can allow additional program-level categories, but the simplified model focuses on the three main functional expense categories to keep the automation clear and practical.

The pseudocode below presents the core logic of the proposed automation model. It is not intended to replace a complete accounting system. Instead, it demonstrates how exported accounting reports can be converted into a draft functional expense report using classification rules, allocation percentages, confidence scoring, reconciliation controls, and accountant review.

Prototype Pseudocode for AI-Based Functional Expense Report Automation

Import Profit and Loss Detail or General Ledger report.

Import supporting files, if available:
Chart of accounts
Vendor list
Payroll records
Invoice descriptions
Class or department codes
Grant or project codes
Historical classifications
Allocation policy table

Import optional Balance Sheet adjustment file only when adjustment entries affect expenses, such as depreciation, prepaid expense amortization, accrued expenses, or reclassification items.

Define the three functional expense categories:
Program Services
Management and General
Fundraising

Clean and standardize the accounting data:
Standardize vendor names
Standardize account names
Standardize transaction dates
Standardize transaction amounts
Clean memo and invoice descriptions
Check for missing class, project, or grant codes

For each expense transaction:
Identify the natural expense category from the chart of accounts.
Check whether the account, vendor, class, project, or grant code directly maps to a functional category.
Analyze memo and invoice descriptions for program, administrative, or fundraising indicators.
Compare the transaction with historical classifications, if available.

If the expense is directly assignable:
Assign the expense to Program Services, Management and General, or Fundraising.
Calculate a confidence score.
Generate a short explanation for the classification.

If the expense is shared:
Apply the approved allocation percentage from the allocation policy table.
Split the transaction amount across Program Services, Management and General, and Fundraising.
Generate an explanation showing the allocation basis.

If the transaction has incomplete or conflicting information:
Mark the item for accountant review.
Assign a lower confidence score.
Require documentation or manual confirmation before final reporting.

Generate a classification log showing:
Original transaction details
Suggested functional category
Allocation percentage
Confidence score
Explanation
Review status

Aggregate expenses by natural expense category and functional expense category.

Reconcile the total functional expense report to the Profit and Loss Detail or General Ledger total.

If a difference exists:
Flag the reconciliation difference for review.
Check for missing transactions, duplicate entries, adjustment errors, or incorrect allocation rules.

Export the final workpapers:
Draft Statement of Functional Expenses
Classification Log
Items Requiring Review
Reconciliation Summary

Table 4 Sample Output Format for AI-Generated Functional Expense Report

Natural Expense Category	Program Services	Management and General	Fundraising	Total Expenses
Salaries and Wages	$80,000	$15,000	$5,000	$100,000
Payroll Taxes and Benefits	$16,000	$3,000	$1,000	$20,000
Rent	$42,000	$12,000	$6,000	$60,000
Professional Fees	$5,000	$20,000	$2,000	$27,000
Program Supplies	$15,000	$0	$0	$15,000
Fundraising Events	$0	$0	$10,000	$10,000
Insurance	$4,000	$5,000	$1,000	$10,000
Total Expenses	$162,000	$55,000	$25,000	$242,000

This output format shows how natural expense categories can be presented across the three major functional expense categories. It allows the accountant to review whether salaries, rent, professional fees, supplies, fundraising costs, and other expenses have been properly classified or allocated. The total expense column also provides a reconciliation point against the Profit and Loss Detail or General Ledger. Behind the summary report, the classification log should be retained as supporting documentation for management review, audit preparation, and Form 990 reporting.

11. Expected Contribution to Nonprofit Reporting and Form 990 Compliance

This study contributes to nonprofit reporting by proposing a practical AI-assisted method for generating the Statement of Functional Expenses from accounting data. The contribution is not limited to classifying individual transactions. The framework provides a full reporting workflow that imports accounting data, classifies direct expenses, allocates shared costs, produces confidence scores, explains decisions, supports human review, reconciles totals, and generates a draft report.

The framework can reduce manual preparation time because accountants no longer need to classify every expense from scratch in spreadsheets. Instead, the system can perform the first review, apply approved allocation rules, and identify items that require attention. This allows accounting teams to focus on exceptions, unclear transactions, and higher-risk classifications. For small and mid-sized nonprofits with limited accounting staff, this can make the reporting process more efficient.

The framework can also improve consistency. Similar transactions can be classified using the same logic across months, programs, and fiscal years. Historical patterns can be used to identify unusual classification changes, while allocation policy tables can help ensure that shared expenses are distributed consistently. This consistency is important because Form 990 and financial statement users often compare nonprofit spending patterns across reporting periods.[17]

The framework supports audit documentation by preserving the reasoning behind each classification and allocation. Each transaction can include the original accounting data, AI recommendation, confidence score, explanation, reviewer decision, and final classification. This creates a stronger audit trail and makes it easier for auditors, management, and board members to understand how the report was prepared.

The framework can also improve donor and grantor transparency. Functional expense reporting affects how users interpret program spending, administrative costs, and fundraising efficiency. When the report is prepared consistently and supported by documentation, donors and grantors can have greater confidence that the organization's financial reports fairly represent how resources were used. In this way, the proposed framework supports both compliance and public accountability.

Limitations and Future Research

This study is limited by its conceptual and prototype-based design. The proposed framework explains how AI can support automated functional expense report generation, but it has not yet been tested using a large real-world nonprofit dataset. Nonprofit accounting data can vary widely by organization, mission, funding source, chart of

accounts, program structure, grant requirements, and accounting system. A model that performs well in one nonprofit may need adjustment before it can be used in another.

Data quality is another major limitation. The framework depends on accounting data such as memo descriptions, class codes, project codes, invoice text, payroll records, and allocation policies. If these inputs are missing, inconsistent, or inaccurate, the model may produce weaker recommendations. For example, a vague memo such as "services" may not provide enough evidence for classification. Missing invoices or weak class coding may also reduce the reliability of the AI-generated report.

The framework may also produce classification errors. Some expenses are straightforward, but many nonprofit costs require judgment. Payroll, rent, utilities, software, insurance, and professional fees may support multiple functions and require allocation. The AI model may suggest a classification or percentage split, but the accountant must still evaluate whether the result is reasonable and properly documented. AI should therefore remain a decision-support tool, not the final reporting authority.

Future research should test the framework using real or synthetic nonprofit accounting datasets. Researchers can compare rule-based models, natural language processing, supervised machine learning, and hybrid classification methods to determine which approach produces the most accurate and explainable results. Future studies can also evaluate whether confidence scores reduce review time and improve the quality of audit documentation.

Additional research should examine sector-specific differences. Healthcare nonprofits, education nonprofits, religious organizations, social service agencies, arts organizations, and grant-funded community programs may use different expense structures and allocation methods. Future studies can examine whether the model should be customized by sector, funding type, or organization size. Research should also explore how spreadsheet-based tools, Python in Excel, and AI assistants can be safely used in nonprofit financial reporting while maintaining human oversight and data governance.[18] [19]

12. Conclusion

Functional expense reporting plays an important role in nonprofit financial reporting and Form 990 compliance because it shows how an organization uses its resources across Program Services, Management and General, and Fundraising. Although reporting guidance exists, many nonprofits still prepare this report manually using spreadsheets, P&L details, general ledger reports, payroll allocations, and judgment-based classifications. This manual process can take significant time and may create inconsistency, weak documentation, allocation errors, and reconciliation issues.

This study proposed an AI-based functional expense report automation framework to make the process more efficient and reliable. The framework imports accounting data, classifies direct expenses, allocates shared costs, generates confidence scores, explains classification decisions, supports accountant review, reconciles totals to the P&L or General Ledger, and produces a draft Statement of Functional Expenses.

The framework is not intended to replace accountants, auditors, or professional judgment. Instead, it works as a decision-support tool that helps nonprofit accounting teams prepare functional expense reports faster and with stronger documentation. By combining accounting logic, text analysis, allocation rules, confidence scoring, reconciliation controls, and human review, the model offers a practical solution to a common nonprofit reporting challenge.

Overall, this study contributes to nonprofit accounting analytics by showing how AI can move beyond basic expense classification and support automated report generation. The proposed framework can help nonprofits reduce manual workload, improve reporting consistency, strengthen Form 990 preparation, support audit readiness, and increase transparency for donors, grantors, regulators, and the public.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Financial Accounting Standards Board. (2016). Accounting Standards Update No. 2016-14: Not-for-Profit Entities (Topic 958): Presentation of Financial Statements of Not-for-Profit Entities. Financial Accounting Standards Board.

[2] Internal Revenue Service. (2025). Instructions for Form 990: Return of Organization Exempt From Income Tax. U.S. Department of the Treasury.

[3] AICPA & CIMA. (2024). Functional expense classification: An overview for not-for-profit entities. Association of International Certified Professional Accountants.

[4] AICPA & CIMA. (2022). Functional expense schedule best practices for not-for-profits. Association of International Certified Professional Accountants.

[5] National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

[6] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc.

[7] Financial Accounting Standards Board. (2016). Accounting Standards Update No. 2016-14: Not-for-Profit Entities (Topic 958): Presentation of Financial Statements of Not-for-Profit Entities. Financial Accounting Standards Board.

[8] Jurafsky, D., & Martin, J. H. (2026). Speech and language processing (3rd ed. draft). Stanford University.

[9] National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

[10] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc.

[11] Alim, I., Meghla, R. N., Matin, T. F., & Prodhan, T. M. M. H. (2026). A semantic and behavioral AI framework for detecting invoice fraud in automated accounts payable. ResearchGate.

[12] Microsoft. (2024). Copilot in Excel: Transforming data analysis. Microsoft Tech Community.

[13] Microsoft. (2025). Introduction to Python in Excel. Microsoft Support.

[14] Microsoft. (2025). Get started with Copilot in Excel. Microsoft Support.

[15] National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

[16] Microsoft. (2025). Introduction to Python in Excel. Microsoft Support.

[17] Microsoft. (2025). Get started with Copilot in Excel. Microsoft Support.

[18] Internal Revenue Service. (2025). Instructions for Form 990: Return of Organization Exempt From Income Tax. U.S. Department of the Treasury.

[19] Microsoft. (2025). Get started with Python in Excel. Microsoft Support.

[20] Microsoft. (2025). COPILOT function. Microsoft Support.

Appendix A. Simplified Python Prototype for AI-Assisted Functional Expense Report Automation

"""AI-Assisted Functional Expense Report Generator

For Nonprofit Financial Reporting and Form 990 Workpapers

Purpose:

This prototype generates a draft Statement of Functional Expenses from:

1. Profit and Loss Detail / General Ledger export

2. Optional Balance Sheet adjustment file

3. Editable allocation percentage rules

Main functional expense categories:

1. Program Service Expenses

2. Management and General / Administrative Expenses

3. Fundraising Expenses

Important accounting note:

The Profit and Loss Detail or General Ledger is the main source for expense transactions.

The Balance Sheet is optional and should only be used for adjustments such as prepaid

expense amortization, accrued expenses, depreciation, or reclassification entries.

"""

```
import pandas as pd
import re
from pathlib import Path
# ============================================================
# 1. USER SETTINGS
# ============================================================
INPUT_FOLDER = Path(".")
PL_FILE = INPUT_FOLDER / "profit_and_loss_detail.csv"
BALANCE_SHEET_ADJUSTMENTS_FILE = INPUT_FOLDER / "balance_sheet_adjustments.csv"
ALLOCATION_RULES_FILE = INPUT_FOLDER / "allocation_rules.csv"
OUTPUT_FILE = INPUT_FOLDER / "functional_expense_report_output.xlsx"
FUNCTIONAL_CATEGORIES = [
    "Program Service Expenses",
    "Management and General / Administrative Expenses",
    "Fundraising Expenses"
]
# ============================================================
# 2. REQUIRED INPUT FORMAT
```

==========

"""

profit_and_loss_detail.csv should include these columns:

Date

Account

Vendor

Amount

Memo

Class

Project

Example:

Date,Account,Vendor,Amount,Memo,Class,Project

01/15/2026,Program Supplies,ABC Supplies,1200,Youth workshop materials,Program,DYCD Youth Program

01/20/2026,Professional Fees,Smith CPA,2500,Audit preparation,Admin,General

01/25/2026,Event Expense,City Venue,1800,Donor fundraising event,Fundraising,Annual Gala

02/01/2026,Rent,Property LLC,6000,Monthly office rent,Shared,General

allocation_rules.csv should include these columns:

Match_Field

Match_Value

Program Service Expenses

Management and General / Administrative Expenses

Fundraising Expenses

Example:

Match_Field,Match_Value,Program Service Expenses,Management and General / Administrative Expenses,Fundraising Expenses

Account,Rent,70,20,10

Account,Utilities,70,20,10

Vendor,Smith CPA,0,100,0

Class,Fundraising,0,0,100

Class,Program,100,0,0

balance_sheet_adjustments.csv is optional.

Use it only when Balance Sheet accounts affect expense reporting.

Required columns:

Date

Account

Vendor

Amount

Memo

Class

Project

Example:

Date,Account,Vendor,Amount,Memo,Class,Project

12/31/2026,Depreciation Expense,Fixed Asset Register,3000,Annual depreciation allocation,Shared,General

"""

```
# ============================================================
# 3. HELPER FUNCTIONS
# ============================================================
def clean_text(value):
    """Clean text for matching."""
    if pd.isna(value):
        return ""
    value = str(value).lower().strip()
    value = re.sub(r"[^a-z0-9\s]", " ", value)
    value = re.sub(r"\s+", " ", value)
    return value
def clean_amount(value):
    """Convert accounting amount to number."""
    if pd.isna(value):
        return 0.0
```

```
    value = str(value).replace("$", "").replace(",", "").strip()
    # Handle accounting negative format such as (1,000)
    if value.startswith("(") and value.endswith(")"):
        value = "-" + value[1:-1]
    try:
        return float(value)
    except ValueError:
        return 0.0
def load_csv_if_exists(path):
    """Load file only if available."""
    if path.exists():
        return pd.read_csv(path)
    return pd.DataFrame(
def validate_columns(df, required_columns, file_name):
    """Check required columns."""
    missing = [col for col in required_columns if col not in df.columns]
    if missing:
        raise ValueError(f"{file_name} is missing required columns: {missing}")
# ============================================================
# 4. LOAD PROFIT AND LOSS DETAIL
# ============================================================
required_columns = ["Date", "Account", "Vendor", "Amount", "Memo", "Class", "Project"]
pl = pd.read_csv(PL_FILE)
validate_columns(pl, required_columns, "profit_and_loss_detail.csv")
pl["Source"] = "Profit and Loss Detail"
# ============================================================
# 5. LOAD OPTIONAL BALANCE SHEET ADJUSTMENTS
# ============================================================
bs_adjustments = load_csv_if_exists(BALANCE_SHEET_ADJUSTMENTS_FILE)
```

```
if not bs_adjustments.empty:
    validate_columns(bs_adjustments, required_columns, "balance_sheet_adjustments.csv")
    bs_adjustments["Source"] = "Balance Sheet Adjustment"
    # Add only adjustment entries, not full Balance Sheet balances.
    data = pd.concat([pl, bs_adjustments], ignore_index=True)
else:
    data = pl.copy()
# ============================================================
# 6. CLEAN DATA
# ============================================================
data["Amount"] = data["Amount"].apply(clean_amount)
for col in ["Account", "Vendor", "Memo", "Class", "Project"]:
    data[f"clean_{col}"] = data[col].apply(clean_text)
data["combined_text"] = (
    data["clean_Account"] + " " +
    data["clean_Vendor"] + " " +
    data["clean_Memo"] + " " +
    data["clean_Class"] + " " +
    data["clean_Project"]
)
# ============================================================
# 7. LOAD EDITABLE ALLOCATION RULES
# ============================================================
rules = pd.read_csv(ALLOCATION_RULES_FILE)
required_rule_columns = ["Match_Field", "Match_Value"] + FUNCTIONAL_CATEGORIES
validate_columns(rules, required_rule_columns, "allocation_rules.csv")
# Convert percentages into decimals
for category in FUNCTIONAL_CATEGORIES:
    rules[category] = rules[category].fillna(0).astype(float) / 100
```

```
# ============================================================
# 8. DEFAULT KEYWORD LOGIC
# ============================================================
KEYWORDS = {
    "Program Service Expenses": [
        "program", "client", "service", "case management", "training",
        "workshop", "education", "youth", "healthcare", "participant",
        "beneficiary", "community outreach", "direct service"
    ],
    "Management and General / Administrative Expenses": [
        "admin", "administration", "management", "accounting", "audit",
        "legal", "insurance", "board", "office", "compliance",
        "bookkeeping", "governance", "general"
    ],
    "Fundraising Expenses": [
        "fundraising", "fundraiser", "donor", "donation", "campaign",
        "gala", "sponsor", "solicitation", "development", "grant writing",
        "appeal", "contribution"
    ]
}
# ============================================================
# 9. CLASSIFICATION FUNCTIONS
# ============================================================
def apply_allocation_rule(row):
    """
    Apply user-approved allocation rules first.
    Example: Rent = 70% Program, 20% Admin, 10% Fundraising.
    """
    for _, rule in rules.iterrows():
```

```
        match_field = rule["Match_Field"]
        if match_field not in row.index:
            continue
        match_value = clean_text(rule["Match_Value"])
        row_value = clean_text(row[match_field])
        if match_value and match_value in row_value:
            allocation = {category: float(rule[category]) for category in FUNCTIONAL_CATEGORIES}
            total = sum(allocation.values())
            if total > 0:
                # Normalize in case percentages do not exactly equal 100%.
                allocation = {k: v / total for k, v in allocation.items()}
                explanation = (
                    f"Applied allocation rule where {match_field} matched "
                    f"'{rule['Match_Value']}'."
                )
                return allocation, 0.95, explanation, "Quick review"
    return None, None, None, None
def keyword_classification(row):
    """
    If no allocation rule applies, classify based on keywords from:
    Account, Vendor, Memo, Class, and Project.
    """
    scores = {category: 0 for category in FUNCTIONAL_CATEGORIES}
    text = row["combined_text"]
    for category, words in KEYWORDS.items():
        for word in words:
            if clean_text(word) in text:
                scores[category] += 1
    total_score = sum(scores.values())
```

```
if total_score == 0:
    allocation = {category: 0 for category in FUNCTIONAL_CATEGORIES}
    explanation = "No strong classification signal was found."
    return allocation, 0.30, explanation, "Manual review required"
top_category = max(scores, key=scores.get)
top_score = scores[top_category]
sorted_scores = sorted(scores.values(), reverse=True)
second_score = sorted_scores[1] if len(sorted_scores) > 1 else 0
# Direct classification if one category is clearly strongest.
if top_score >= second_score + 2:
    allocation = {category: 0 for category in FUNCTIONAL_CATEGORIES}
    allocation[top_category] = 1.0
    confidence = min(0.90, 0.50 + (top_score / max(total_score, 1)) * 0.40)
    explanation = (
        f"Classified as {top_category} based on keyword and coding signals "
        f"from account, vendor, memo, class, and project fields."
    )
    if confidence >= 0.85:
        review_status = "Quick review"
    else:
        review_status = "Accountant review"
    return allocation, confidence, explanation, review_status
# If signals are mixed, allocate by score but require review.
allocation = {
    category: scores[category] / total_score
    for category in FUNCTIONAL_CATEGORIES
}
explanation = (
    "Mixed classification signals were found. Amount was allocated based on "
```

```
        "relative keyword and coding strength, but accountant review is required."
    )
    return allocation, 0.55, explanation, "Accountant review required"
def classify_row(row):
    """
    Full classification process:
    1. Apply editable allocation rules
    2. If no rule applies, use keyword-based classification
    """
    allocation, confidence, explanation, review_status = apply_allocation_rule(row)
    if allocation is not None:
        return allocation, confidence, explanation, review_status
    return keyword_classification(row)
# ============================================================
# 10. APPLY CLASSIFICATION
# ============================================================
classified_rows = []
for _, row in data.iterrows():
    allocation, confidence, explanation, review_status = classify_row(row)
    output_row = row.to_dict()
    top_category = max(allocation, key=allocation.get) if sum(allocation.values()) > 0 else "Manual Review"
    output_row["Suggested Category"] = top_category
    output_row["Confidence Score"] = round(confidence, 2)
    output_row["Explanation"] = explanation
    output_row["Review Status"] = review_status
    for category in FUNCTIONAL_CATEGORIES:
        output_row[f"{category} %"] = round(allocation.get(category, 0) * 100, 2)
        output_row[f"{category} Amount"] = row["Amount"] * allocation.get(category, 0)
    classified_rows.append(output_row)
```

```
classified = pd.DataFrame(classified_rows)
# ============================================================
# 11. GENERATE FUNCTIONAL EXPENSE REPORT
# ============================================================
amount_columns = [f"{category} Amount" for category in FUNCTIONAL_CATEGORIES]
report = classified.groupby("Account")[amount_columns].sum().reset_index()
rename_columns = {
   f"{category} Amount": category
   for category in FUNCTIONAL_CATEGORIES
}
report = report.rename(columns=rename_columns)
report["Total Expenses"] = report[FUNCTIONAL_CATEGORIES].sum(axis=1)
# Add total row
total_row = {"Account": "Total Expenses"}
for category in FUNCTIONAL_CATEGORIES:
   total_row[category] = report[category].sum()
total_row["Total Expenses"] = report["Total Expenses"].sum()
report = pd.concat([report, pd.DataFrame([total_row])], ignore_index=True)
# ============================================================
# 12. RECONCILIATION
# ============================================================
source_total = data["Amount"].sum()
report_total = report.loc[report["Account"] == "Total Expenses", "Total Expenses"].iloc[0]
difference = source_total - report_total
reconciliation = pd.DataFrame({
   "Reconciliation Item": [
      "Total expenses from source data",
      "Total expenses from generated functional expense report",
      "Difference"
```

```
    ],
    "Amount": [
        source_total,
        report_total,
        difference
    ]
})
# ============================================================
# 13. ITEMS FOR REVIEW
# ============================================================
review_items = classified[
    classified["Review Status"].str.contains("review", case=False, na=False)
].copy()
# ============================================================
# 14. EXPORT FINAL WORKBOOK
# ============================================================
with pd.ExcelWriter(OUTPUT_FILE, engine="xlsxwriter") as writer:
    report.to_excel(writer, sheet_name="Functional Expense Report", index=False)
    classified.to_excel(writer, sheet_name="Classification Log", index=False)
    review_items.to_excel(writer, sheet_name="Items for Review", index=False)
    reconciliation.to_excel(writer, sheet_name="Reconciliation", index=False)
print("Functional Expense Report generated successfully.")
print(f"Output file: {OUTPUT_FILE}")
print(f"Source total: {source_total:,.2f}")
print(f"Report total: {report_total:,.2f}")
print(f"Difference: {difference:,.2f}")
```