INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 2, February 2026 | ISSN: 2456-4184 | IJNRD.ORG

Clothing Shop Expense Tracker with Data Visualization

R. Bhuvaneswari[1], S. Indira[2] R. Bhuvaneswari[1],
M.Sc., M.Phil., MTech., SET., Assistant Professor,
Department of Computer Science and Applications
D.K.M. College for Women (Autonomous)
S. Indira[2], PG Student, Department of Computer Science and Applications
D.K.M. College for Women (Autonomous), Vellore, Tamil Nadu, India.

Abstract— Managing financial transactions and maintaining accurate records has become an essential requirement for small and medium-scale retail businesses, particularly clothing shops where daily transactions are frequent and diverse. Traditional methods such as manual bookkeeping or generic spreadsheet tools are error-prone, time-consuming, and lack analytical capabilities. This paper presents the design and development of a Clothing Shop Expense Tracker with Data Visualization — a web-based platform that automates financial tracking and delivers actionable insights through advanced visualization techniques. The system enables shop owners to create multiple shop profiles under a single account, record daily expenses across categories such as fabric purchases, tailoring charges, rent, wages, and utilities, and monitor income from sales. Profit is automatically computed as the difference between total income and total expenses, ensuring real-time accuracy and transparency. The platform integrates interactive data visualizations including bar charts, pie charts, donut charts, line graphs, polar area charts, radar charts, and horizontal bar charts using Chart.js on the frontend and Matplotlib on the backend, transforming raw financial data into strategic insights. Built on the Django 5.2.4 framework with SQLite as the database, Pandas and NumPy for data processing, and Tailwind CSS for a responsive user interface, the system demonstrates how modern web technologies can replace inefficient traditional practices in the retail sector and empower business owners with data-driven decision-making capabilities.

Keywords—Expense Tracker, Data Visualization, Django, Chart.js, Matplotlib, SQLite, Pandas, NumPy, Retail Management, Financial Analytics, Tailwind CSS.

I. INTRODUCTION

The retail clothing industry is characterised by a high volume of daily financial transactions involving diverse expense categories including raw material procurement, tailoring and labour charges, electricity and utility bills, rent, and employee wages. Shop owners simultaneously monitor income from garment sales across multiple product lines such as men's and women's clothing, adult and children's wear. This complexity makes accurate financial management a critical operational requirement, yet many small and medium-scale clothing businesses continue to rely on manual bookkeeping or generic tools like Microsoft Excel and QuickBooks that lack the specificity and analytical depth required for the sector.

The consequences of inadequate financial tracking are significant: delayed identification of cost overruns, inaccurate profit computations, difficulty in multi-shop management, and an inability to detect seasonal trends or spending patterns. Without visual analytical tools, even correct data remains difficult to interpret quickly and act upon.

This paper presents the Clothing Shop Expense Tracker with Data Visualization, a purpose-built web application that addresses these gaps. The system provides clothing shop owners with a unified digital platform to record income and expenses, automatically calculate profits, and gain visual insights through an integrated suite of eight interactive chart types. The application supports multiple shop profiles under a single user account, enforces secure role-based access, and is designed to be accessible to users with minimal technical expertise.

The backend is implemented using Django 5.2.4 — a high-level Python web framework — with SQLite as the relational database, Pandas and NumPy for data processing and aggregation, and Matplotlib for server-side chart generation. The frontend uses HTML5, CSS3, Tailwind CSS, and Chart.js for responsive, interactive visualizations. The system was developed under the guidance of Redback IT Solutions Private Limited and demonstrates how accessible web technologies, combined with thoughtful analytical features, can modernize financial management in the retail clothing sector.

The remainder of this paper is organized as follows. Section II reviews related literature. Section III describes the proposed system and its design. Section IV details the technology stack. Section V covers the implementation architecture. Section VI presents results and discussion. Section VII concludes with future enhancement directions.

II. LITERATURE REVIEW

Sharma and Kumar presented a web-based financial management system for small retail businesses using PHP and SQLite, featuring expense categorization and automated profit calculation. While effective for basic tracking, the system lacked interactive visualizations, relying on tabular data representations that limited

analytical depth. Their usability evaluation nonetheless confirmed high adoption among non-technical retail users, underscoring the importance of simple interfaces in this domain.

Lee and Gonzalez explored data visualization for retail financial decision-making using Python (Pandas and Matplotlib) with a Flask web interface and PostgreSQL backend. Their prototype demonstrated the analytical value of bar and line chart representations of expense and revenue patterns. However, the system operated on static datasets and lacked real-time processing, limiting its practical applicability in dynamic retail environments. The authors noted the need for JavaScript-based interactive libraries such as D3.js for improved frontend engagement.

Patel and Reddy proposed a Django-based expense tracking system for small enterprises using SQLite and Django's ORM, with emphasis on security features such as authentication and data validation. Their evaluation with a local retail chain confirmed high system reliability, but the absence of visualization features and multi-user dashboard support for individual business units restricted operational insights. This work is directly relevant as the closest prior system using the same backend technology stack.

Chen and Kumar developed interactive financial dashboards using Chart.js and Node.js with MongoDB storage. Their work highlighted the effectiveness of interactive pie, bar, and line charts for helping retailers understand financial trends. Performance degradation with large datasets was identified as a limitation, and the absence of robust Python-based backend analytics (comparable to Pandas/NumPy) was noted as a gap relative to server-side solutions.

Khan and Wong presented a cloud-based financial management system integrating machine learning (scikit-learn) for predictive analytics, with Plotly for visualization and AWS for storage. While delivering predictive insights into spending patterns, the system's high implementation complexity and cost made it unsuitable for the small business context. The proposed system occupies a practical middle ground: richer than basic tracking tools yet accessible and affordable for clothing shop owners.

In summary, existing solutions either lack visualization capabilities, are not tailored to the clothing retail context, rely on static datasets, or introduce complexity that is inaccessible to non-technical users. The proposed system addresses all of these gaps through an industry-specific, visualization-rich, Django-based web platform.

III. PROPOSED SYSTEM

A. Problem Definition and Existing Drawbacks

Existing financial management tools for clothing retail businesses share several critical limitations: lack of industry-specific expense categories (e.g., fabric, tailoring) requiring manual customization; absence of interactive data visualizations beyond tabular reports; time-consuming and error-prone manual entry workflows; poor scalability for multi-shop environments; and limited accessibility due to desktop-based deployment constraints.

B. System Overview

The proposed Clothing Shop Expense Tracker addresses these gaps through a unified web platform supporting four core functional areas. First, user-specific shop management allows owners to create and maintain multiple shop profiles under a single account, with shop-level financial isolation ensuring privacy and accuracy. Second, automated financial tracking enables structured recording of expenses across predefined clothing-industry categories — fabric purchase, tailoring charges, rent, wages, and utilities — alongside income from garment sales. Profit is computed automatically as: Profit = Total Income − Total Expenses, updated in real time with each transaction entry. Third, interactive data visualization delivers eight distinct chart types covering expense distribution, income breakdown, profit/loss by category, monthly trends, subcategory analysis, category performance, top cloth items, and cloth-level profit trends. Fourth, data export capabilities allow shop owners to download financial records as Excel or PDF reports for offline reference and accounting.

C. Module Architecture

The system is organized into four Django applications, each encapsulating a distinct domain of functionality:

- Accounts App: Manages user registration, authentication (login/logout), profile creation and updates, and password management. The Profile model extends Django's built-in User model via a OneToOneField, adding phone number and address fields. Forms use django-crispy-forms with Tailwind styling.
- Shops App: Handles creation, editing, and listing of shop profiles. The Shop model associates shops with users via a ForeignKey, supporting multi-shop management with ownership validation. ShopListView limits display to the authenticated user's own shops.
- Transactions App: Forms the financial core of the system. The Category → Subcategory → Cloth hierarchy organizes clothing items, while Expense and Income models record transactions linked to a user, shop, and cloth item with quantity, amount, date, and description. Django-filter enables dynamic filtering by category, date, and shop. Export views generate PDF (reportlab) and Excel (openpyxl) outputs.
- Dashboard App: Serves as the visualization hub. The DashboardView retrieves the past 12 months of transactions per user, processes them with Pandas (grouping by category and time period) and NumPy (trend computation), and passes JSON-serialized data to the frontend for rendering by Chart.js. Matplotlib generates backend chart images for report embedding.

D. Data Flow

Data flows from user input through Django form validation into the SQLite database via Django ORM. The Dashboard View retrieves and aggregates this data using Pandas DataFrames, applies NumPy computations for statistical measures, and serializes results as JSON context variables. Chart.js reads these variables from the rendered HTML template to initialize interactive charts. For static report generation, Matplotlib

processes the same aggregated data and outputs chart images embedded in PDF or Excel exports.

IV. TECHNOLOGY STACK

A. Backend Framework: Django 5.2.4

Django's Model-View-Template (MVT) architecture separates data logic, business logic, and presentation, enabling rapid development and maintainability. Its built-in ORM eliminates raw SQL for standard operations, providing database abstraction that supports future migration from SQLite to PostgreSQL or MySQL with minimal code changes. Django's authentication system, CSRF protection, and SQL injection prevention provide a robust security foundation. The asgiref library supports asynchronous operation for future real-time features.

B. Database: SQLite

SQLite is used as the relational database, providing ACID-compliant transaction support essential for accurate financial record-keeping. Django ORM maps models to SQLite tables: auth_user (built-in), accounts_profile (extended profile), shops_shop (shop records), transactions_category, transactions_subcategory, transactions_cloth (clothing hierarchy), and transactions_expense and transactions_income (financial records). Foreign key constraints enforce referential integrity across all relational links.

C. Data Processing: Pandas and NumPy

Pandas 2.3.1 provides DataFrame-based data manipulation for grouping expenses and income by category, subcategory, cloth type, and time period. NumPy 2.3.2 supports numerical computations including profit trend analysis and statistical aggregation. The combination enables real-time generation of all chart datasets from raw transaction records within a single dashboard view request.

D. Visualization: Chart.js and Matplotlib

Chart.js 3.x delivers eight interactive frontend chart types rendered via HTML5 Canvas: Pie Chart (expense category distribution), Donut Chart (income distribution), Bar Chart (profit/loss by category), Line Chart (monthly income vs. expense trends), Polar Area Chart (subcategory expenses, top 8), Graph/Line Chart (category performance), Horizontal Bar Chart (top 6 cloth items by total amount), and Radar Chart (cloth profit trends). Matplotlib 3.10.3 generates static server-side charts for report embedding, with Pillow 11.3.0 supporting image processing for high-quality outputs.

E. Frontend: Tailwind CSS and Django Templates

Tailwind CSS 3.x provides utility-first styling for a responsive, accessible interface without custom CSS overhead. Django Crispy Forms with crispy-bootstrap4 renders form fields with consistent Bootstrap 4 styling. Django-widget-tweaks enables fine-grained field-level customization. The base.html master template provides a consistent layout with navigation, and app-specific templates extend it for each functional page.

Table I: Software Requirements Summary

Component	Tool/Library	Version
Web Framework	Django	5.2.4
Database	SQLite	8.0+
Language	Python	3.9+
Data Analysis	Pandas / NumPy	2.3.1 / 2.3.2
Backend Charts	Matplotlib	3.10.3
Frontend Charts	Chart.js	3.x
CSS Framework	Tailwind CSS	3.x
PDF Export	ReportLab	4.4.3
Excel Export	OpenPyXL	3.1.5

V. IMPLEMENTATION

A. Project Setup and Configuration

The project is structured under the clothing_expense_tracker directory, with a virtual environment (env) isolating all dependencies. The settings.py file configures the SQLite database, registers the four apps (accounts, shops, transactions, dashboard), enables middleware for security and session management, and configures static file handling for Tailwind CSS and Chart.js assets. The urls.py root configuration routes requests to app-specific URL patterns (e.g., /transactions/add/, /shops/list/, /dashboard/). The manage.py utility handles migrations, local server startup, and static file collection.

B. User Authentication Workflow

New users register via the register view using CustomUserCreationForm, which extends Django's UserCreationForm with help text removed for a cleaner interface. On successful registration, users are automatically authenticated and redirected to the dashboard. The login_view processes credentials via Django's AuthenticationForm. The profile view displays account details, while profile_update supports simultaneous profile and password changes, using update_session_auth_hash to maintain the active session after password modification. The logout_view terminates the session and redirects to the login page.

C. Shop and Transaction Management

Authenticated users create shops via ShopCreateView with validation enforcing unique shop names per user. TransactionCreateView handles expense and income entry with Django form validation ensuring numeric amounts and valid dates. TransactionListView displays paginated transaction records with django-filter support for filtering by category, date range, and shop. TransactionExportView generates PDF reports via ReportLab and Excel files via OpenPyXL, embedding summary statistics and Matplotlib-generated charts.

D. Dashboard and Visualization

The dashboard view retrieves all transactions from the past 12 months for the authenticated user. It uses Django's TruncMonth and Coalesce aggregation functions alongside Pandas groupby

operations to compute monthly totals, category-level breakdowns, subcategory distributions, and cloth-level profit figures. All computed datasets are serialized to JSON and injected into the template context. On the frontend, dedicated JavaScript functions initialize each of the eight Chart.js charts: createExpensePieChart, createIncomeDonutChart, createProfitBarChart, createTrendsLineChart, createSubcategoryPolarChart, createPerformanceRadarChart, createClothHorizontalChart, and createClothTrendsChart. A GET parameter (period=day or period=month) adjusts temporal granularity dynamically.

VI. RESULTS AND DISCUSSION

A. Functional Outcomes

The implemented system successfully delivers all proposed functional requirements. Data owners can create and manage multiple clothing shop profiles, each with independent financial records. Expenses are recorded against a three-level hierarchy (Category → Subcategory → Cloth), providing granularity appropriate for clothing retail operations. Income entries follow the same hierarchy, enabling cloth-item-level profitability analysis. Profit is computed automatically with each transaction update, eliminating manual calculation errors.

The eight-chart dashboard provides comprehensive financial visibility. The Pie and Donut charts enable instant identification of dominant expense and income categories. The Bar chart highlights category-level profit and loss for strategic resource allocation. The Line chart reveals monthly income and expense trends, supporting seasonal planning. The Polar Area chart focuses attention on the highest-cost subcategories. The Horizontal Bar chart ranks cloth items by total financial impact, while the Radar chart delivers multi-dimensional cloth profit analysis.

B. Comparison with Existing Systems

Table II: Comparison with Existing Solutions

Feature	Manual/Excel	QuickBooks	Patel et al.	Chen et al.	Proposed
Industry-specific	No	No	No	No	Yes
Interactive Charts	No	Limited	No	Yes	Yes (8)
Multi-shop	No	Partial	No	No	Yes
Auto Profit Calc	Manual	Yes	Yes	Yes	Yes
Web-based	No	Yes	Yes	Yes	Yes
PDF/Excel Export	Manual	Yes	No	No	Yes
Open Source	Yes	No	Yes	Yes	Yes

C. System Performance

The Django development server demonstrated consistent sub-second response times for dashboard page loads with up to 500 transaction records per shop, attributed to efficient Pandas aggregation and Django ORM query optimisation. SQLite handled concurrent read operations reliably within the single-server deployment model, with the virtual environment ensuring consistent dependency resolution. Export operations generating PDF and Excel files completed within two to three seconds for typical datasets, meeting practical usability requirements.

D. Testing

The system underwent unit testing of individual model methods and form validators; integration testing of the ORM-to-view data pipeline; functional testing of all eight chart types with varied datasets; and user acceptance testing confirming that shop owners could complete end-to-end workflows (register, create shop, record transactions, view dashboard, export report) without technical guidance. All critical paths passed testing with no blocking defects.

VII. CONCLUSION

This paper presented the design and implementation of a Clothing Shop Expense Tracker with Data Visualization — a web-based financial management platform purpose-built for the retail clothing industry. The system successfully replaces error-prone manual bookkeeping and generic accounting tools with an automated, industry-specific solution that provides real-time profit computation, multi-shop management, and a comprehensive eight-chart interactive dashboard covering expense distribution, income analysis, profit trends, subcategory breakdowns, and cloth-level performance.

Built on Django 5.2.4 with SQLite, Pandas, NumPy, Matplotlib, Chart.js, and Tailwind CSS, the platform achieves a balance of analytical capability and user accessibility that makes advanced financial insights available to non-technical clothing shop owners. The open-source technology stack ensures zero licensing costs, supporting adoption in small and medium-scale retail businesses.

Comparative evaluation confirms that the proposed system uniquely combines industry-specific expense categorization, interactive multi-chart visualization, multi-shop profile support, and PDF/Excel export in a single web-based platform — capabilities not simultaneously available in any single existing solution reviewed.

Future enhancements include mobile application development (React Native/Flutter), inventory management integration with stock-level tracking, AI/ML-powered predictive analytics for sales forecasting and anomaly detection, multi-user role-based access control, cloud deployment with PostgreSQL migration, multi-currency and multi-language support, and integration with third-party accounting tools such as Tally and QuickBooks API.

REFERENCES

[1] P. Sharma and A. Kumar, "A Web-Based Financial Management System for Small Retail Businesses," International Journal of Computer Applications, vol. 175, no. 12, pp. 1–8, 2020.

[2] J. Lee and M. Gonzalez, "Data Visualization for Financial Decision-Making in Retail," Journal of Business Analytics, vol. 5, no. 2, pp. 45–58, 2021.

[3] R. Patel and S. Reddy, "A Django-Based Expense Tracking System for Small Enterprises," International Journal of Engineering and Technology, vol. 8, no. 4, pp. 112–119, 2019.

[4] E. Chen and R. Kumar, "Interactive Dashboards for Retail Financial Analysis Using Chart.js," IEEE Access, vol. 9, pp. 78234–78245, 2021.

[5] A. Khan and L. Wong, "Cloud-Based Financial Management for Retail Using Machine Learning," Journal of Cloud Computing, vol. 10, no. 1, p. 23, 2021.

[6] Django Software Foundation, "Django Documentation – Version 5.2," 2024. [Online]. Available: https://docs.djangoproject.com/en/5.2/

[7] W. McKinney, "Data Structures for Statistical Computing in Python," in Proc. 9th Python in Science Conference (SciPy), Austin, TX, USA, 2010, pp. 51–56.

[8] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.

[9] C. R. Harris et al., "Array Programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.

[10] Chart.js Contributors, "Chart.js Documentation – Version 3.x," 2023. [Online]. Available: https://www.chartjs.org/docs/latest/

[11] A. Holovaty and J. Kaplan-Moss, The Definitive Guide to Django: Web Development Done Right, 2nd ed. Apress, 2009.

[12] Tailwind Labs, "Tailwind CSS Documentation – Version 3.x," 2023. [Online]. Available: https://tailwindcss.com/docs/

[13] K. Reitz and T. Schlusser, The Hitchhiker's Guide to Python. O'Reilly Media, 2016.

[14] ReportLab Group, "ReportLab PDF Library User Guide," 2023. [Online]. Available: https://www.reportlab.com/docs/reportlab-userguide.pdf

[15] OpenPyXL Contributors, "OpenPyXL Documentation – Version 3.1," 2023. [Online]. Available: https://openpyxl.readthedocs.io/