Expense Tracker

Submitted By

Student Name	**Student ID**
Samata Islam Zerin	0242220005101835
Md. Fahim Ferdus	0242220005101784
Nur Mohammad Nibir	0242220005101833

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE222: Object Oriented Programming II Lab in the Computer Science and Engineering Department**

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 17, 2024

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Nasima Islam Bithi**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

____
Nasima Islam Bithi
Lecturer
Department of Computer Science and Engineering Daffodil
International University

Submitted by

Samata Islam ____________________ Samata Islam Zerin ID:0242220005101835 Dept. of CSE, DIU	
Fahim ____________________ Md. Fahim Ferdus ID:0242220005101784 Dept. of CSE, DIU	Nibir ____________________ Nur Mohammad Nibir ID:0242220005101833 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	**Statements**
CO1	Solve computational problems effectively using Python programming, including data handling and algorithm development.
CO2	Develop solutions with object-oriented programming concepts in Python, enabling modularity and scalability.
CO3	Apply Python and OOP knowledge to design, implement, and optimize solutions for real-world problems.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	**PO**	**Blooms**	**KP**	**CEP**
CO1	PO1	C1, C2	KP3	EP1, EP3
CO2	PO2	C2	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Table of Contents Table of Contents

Chapter 1

Introduction

The project's aim is to improve personal finance management through technology, highlighting motivation, objectives, and expected outcomes.

1.1 Introduction

In today's fast-paced world, efficient financial management is essential for maintaining stability and achieving personal or organizational goals. Traditional methods of tracking expenses can be time-consuming, error-prone, and lack the ability to provide real-time insights. This project aims to address these challenges by developing a Smart Expense Tracker System. Using advanced Python concepts, the system will help users monitor their spending, categorize expenses, and make informed financial decisions for better budgeting and financial planning.

1.2 Motivation

This project aims to make personal finance management more efficient using technology. Traditional methods of tracking expenses often lead to poor budgeting and missed financial opportunities due to a lack of real-time insights. By addressing this issue, users can achieve better control over their finances, reduce unnecessary spending, and improve savings, leading to greater financial stability. Additionally, this project provides an opportunity to apply and enhance my programming and data analysis skills, which are invaluable for my career growth.

1.3 Objectives

Objectives of the Expense Tracker

- **Track Expenses**: Record and categorize personal or organizational expenses in real time.
- **Use Advanced Python**: Implement features using OOP, data analysis, and visualization.
- **Analyze Spending**: Provide insights into spending habits and suggest savings opportunities.
- **Generate Reports**: Save data as JSON files and export expense reports in .docx format.
- **User-Friendly Interface**: Offer a simple menu for adding, viewing, filtering, and updating expenses.
- **Secure Access**: Include sign-in and sign-up for personalized and secure usage.
- **Financial Insights**: Help users make better financial decisions with structured analysis.

1.4 Feasibility Study

Research Studies

Expense Tracking
This study examines the development of expense tracking applications and their impact on personal finance management, with a focus on cloud-based solutions and integration with banking systems [1].

Advancements in FinTech
Digital Solutions for Financial Management
A survey of cutting-edge technologies in FinTech, such as machine learning, blockchain, and digital wallets, showcasing their role in automating expense management processes [2].

Case Studies

Personal Finance Management
This case study explores how artificial intelligence is employed to analyze spending patterns and offer personalized budgeting advice to users [3].

Methodological Contribution
The expense tracker utilizes user-defined categories and automated categorization methods to ensure accuracy and clarity in financial tracking. It validates category inputs and organizes expenses for simplified management.

Enhanced features include total expense calculations, category-wise breakdowns, and numpy-based data processing methods. These enable advanced analysis through reshaping, flattening, and slicing, offering users deeper insights into their financial habits [5][6].

Web Applications

Mint Integration
Mint inspires the design of this expense tracker by enabling features such as automatic categorization, bank account linking, and budget planning. Adopting similar functionalities could significantly enhance the system's utility [7].

Expensify-Like Reporting
Expensify's ability to generate comprehensive financial reports and export them in various formats can be emulated by extending the tracker's export options to include PDF and spreadsheet formats [4].

1.5 Gap Analysis

Integration of Advanced Python Techniques:
Many existing solutions do not leverage advanced Python capabilities such as machine learning for predictive budgeting or data visualization libraries for dynamic reporting.

Real-Time Data Collection and Analysis:
Current expense trackers often lack real-time synchronization with multiple sources like bank accounts, credit cards, and bills.

Actionable Insights:
Few systems provide detailed, practical recommendations to help users optimize spending habits and improve financial health.

Sustainability and Long-Term Planning:

Limited tools focus on promoting sustainable financial habits and long-term financial goals.

User-Friendly Interface:
Existing solutions can be overwhelming for users, especially those unfamiliar with financial jargon or technology. Current systems can be complex and difficult for farmers to use, especially those with limited technical knowledge.

1.6 Project Outcome

The potential outcomes of this project include:

Better Expense Management: Enhanced ability to track and manage personal or organizational finances.

Real-Time Monitoring: Continuous synchronization with financial accounts for up-to-date expense tracking.

Actionable Insights: Data-driven recommendations to optimize spending and saving habits**.**

Sustainability: Promotion of long-term financial discipline and resource allocation.

User-Friendly System: A simple, intuitive platform that caters to users with varying levels of technical expertise.

Chapter 2

Proposed Methodology/Architecture

This section provides an overview of the system's requirements and design, with a focus on the architecture, key modules, and project plan.

2.1 Requirement Analysis & Design Specification
Requirement Analysis:

• **Functional Requirements:**
- Manage Expenses
- Real-Time Tracking
- Analyze Data
- User-Friendly Interface.

• **Non-Functional Requirements:**
- Real-Time Performance
- Scalability
- Reliability & Accuracy
- Ease of User Design Specification

• **System Architecture:**
- **Backend in Python**
- **User Interface (UI)**
- **Database**
- **Data Collection**

• **Key Components:**
- **Expense Management Module**
- **Data Analysis & Statistics Module**
- **User Interaction Module**

2.1.1 Overview

The **Expense Tracker System** is designed to help users manage their daily expenses by integrating advanced features such as real-time data processing, user-friendly interfaces, and export capabilities.

Key Components:
- **Expense Management Module**: Manages the recording and categorization of expenses.
- **Analytics Module**: Provides insights and reports on total expenses, category distribution, and more.
- **User Interface Module:** A user-friendly interface for easy interaction with the system.

System Architecture:

- **Backend:** Developed with Python, handling all business logic and data processing.
- **Frontend:** Web or mobile application providing a user-friendly interface.
- **Database:** Stores expense details, including amounts, categories, and descriptions.
- **Data Integration:** Real-time processing and reporting through various modules.

Benefits:

- Simplified expense tracking.
- Real-time monitoring and analysis.
- User-friendly platform with detailed reporting.
- Export capabilities for documentation.

2.1.2 Proposed Methodology/ System Design

- **Frontend (Web/Mobile App)**
  - User Interface
    - Dashboard
    - Expense Management
    - Reports & Insights

- **Backend (Python)**
  - API Layer
    - RESTful APIs
  - Business Logic

    Expense Management Module: Handles addition, update, and categorization of expenses.

    Analytics Module: Provides analysis of the data, including total expenses and category distribution.

2.2 Overall Project Plan

Phase 1: Planning and Requirement Analysis

- Identify Requirements: Gather user needs and define system features.
- Define Project Scope: Set clear project goals and objectives.
- Feasibility Study: Assess technical and financial feasibility.
-

Phase 2: System Design

- Design Architecture: Create a comprehensive architecture plan.
- Design Specifications: Create detailed design for each component.
- UI/UX Designs: Develop wireframes and mockups for the interface.

Phase 3: Development

- Backend Development: Set up Python environment, implement modules, and integrate APIs.
- Frontend Development: Develop the user interface for web/mobile.
- Database Setup: Design and implement the necessary database schema.
- IoT Integration (if needed): Integrate real-time sensors or other IoT components for monitoring.

Phase 4: Testing

- Unit Testing: Test individual components.
- Integration Testing: Test the interaction between modules.
- User Acceptance Testing: Ensure the system meets user requirements and expectations.

Phase 5: Deployment

- System Deployment: Deploy the application in production.
- Training & Support: Provide necessary training and support for users.

Phase 6: Monitoring and Maintenance

- Performance Monitoring: Continuously monitor system performance.
- Maintenance: Perform regular updates and improvements based on user feedback.

This methodology covers all the necessary aspects of the project, ensuring smooth development and implementation, as well as continuous improvement post-deployment.

Chapter 3

Implementation and Results

The section outlines the project's implementation, performance analysis, and key results.

3.1 Implementation

Backend Development:
- Develop core logic with Python.
- Implement APIs.
- Integrate IoT sensors.

Frontend Development:
- Create user-friendly web/mobile app.
- Design intuitive interfaces.

Database Setup:
- Implement database schema.
- Store crop and sensor data.

Deployment:
- Deploy system.
- Provide training and support.[4]

Python Code for implementation:

```
# Importing necessary libraries
!pip install python-docx
import numpy as np
import json
import docx
from google.colab import files

# Base class for expense tracker
class ExpenseTracker:
    def __init__(self):
        self.expenses = []  # List to store expenses
        self.categories = ["Food", "Transport", "Shopping", "Utilities", "Others"]

    #Method to add a new expense
    def add_expense(self, amount, category, description):
        """Adds a new expense to the tracker."""
        try:
            #Check if the catagory is valid
            if category not in self.categories:
                raise ValueError("Invalid category. Choose from: " + ", ".join(self.categories))
            #Add expense to the list
            self.expenses.append({"amount": float(amount), "category": category, "description": description})
            print(f"Added expense: {amount} in category '{category}' with description '{description}'")
        except ValueError as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"Unexpected error: {e}")


    #Method to view all recorded expense
    def view_expenses(self):
        """Displays all the expenses."""
        if not self.expenses:
            print("No expenses recorded yet.")
        else:
            print("\n--- All Expenses ---")
            for idx, expense in enu  merate(self.expenses, start=1):
                print(f"{idx}. {expense['amount']} - {expense['category']} - {expense['description']}")
            print("\n")

    #Method to filter expense by catagory
    def filter_expenses(self, category):
        """Filters expenses by category."""
        filtered = [expense for expense in self.expenses if expense["category"] == category]
        if not filtered:
            print(f"No expenses found in category '{category}'")
        else:
            print(f"\n--- Expenses in '{category}' ---")
            for expense in filtered:
                print(f"{expense['amount']} - {expense['description']}")
            print("\n")

    #Method to update an existing expense
    def update_expense(self, index, amount, category, description):
        """Updates an existing expense."""
        try:
            #check if the expense index is valid
            if index < 0 or index >= len(self.expenses):
                print("Invalid index. Expense not found.")
                return
            #check if the catagory is valid
            if category not in self.categories:
                raise ValueError("Invalid category. Choose from: " + ", ".join(self.categories))
            #Update the expense
            self.expenses[index] = {"amount": float(amount), "category": category, "description": description}
            print(f"Updated expense at index {index + 1} with new details: {amount} - {category} -
{description}")
        except ValueError as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"Unexpected error: {e}")
    #Method to save expenses to a JSON file
    def save_expenses(self, file_name="expenses.json"):
        """Saves expenses to a file."""
        try:
            with open(file_name, "w") as file:
                json.dump(self.expenses, file)
            print("Expenses saved successfully!")
        except Exception as e:
            print(f"Error saving expenses: {e}")

    #Method to export expenses to a .docx file
    def save_expenses_as_doc(self, file_name="expenses.docx"):
        """Exports expenses to a .docx file and enables download in Colab."""
        try:
            # Create and save the document
            doc = docx.Document()
            doc.add_heading('Expense Report', level=1)
            for idx, expense in enumerate(self.expenses, start=1):
                doc.add_paragraph(f"{idx}. Amount: {expense['amount']}, Category: {expense['category']},
Description: {expense['description']}")
            doc.save(file_name)
            print(f"Expenses exported to {file_name} successfully!")

            # Download the file in Colab
            files.download(file_name)
            print(f"{file_name} downloaded successfully!")
        except Exception as e:
            print(f"Error saving expenses as .docx: {e}")

    def process_expenses(self):
        # Assuming self.expenses is a NumPy array
        expenses = np.array(self.expenses)
        print(f"\nOriginal Array: {expenses}\n")

        # Indexing & Slicing
        first_two_expenses = expenses[:2]
        print(f"First Two Expenses: {first_two_expenses}")
        last_two_expenses = expenses[-2:]
        print(f"Last Two Expenses: {last_two_expenses}\n")

        # Copy & View
        view_expenses = expenses[:3]
        copy_expenses = expenses[-3:].copy()
        print(f"View as first three: {view_expenses}")
        print(f"Copy as last three: {copy_expenses}\n")

        # Shape & Reshape
        if expenses.size % 2 == 0:
            reshaped_expenses = expenses.reshape(-1, 2)
            print(f"Reshaped: {reshaped_expenses}\n")
        else:
            reshaped_expenses = np.pad(expenses, (0, 1), mode='constant').reshape(-1, 2)
            print(f"Reshaped: {reshaped_expenses}\n")

        # Flatten
        flattened_expenses = reshaped_expenses.flatten()
        print(f"Flattened: {flattened_expenses}\n")

# Inherited class for statistics
class ExpenseStats(ExpenseTracker):
    def __init__(self):
        super().__init__()

    def total_expenses(self):
        """Calculates the total expenses."""
        total = sum(expense["amount"] for expense in self.expenses)
        print(f"Total expenses: {total}")
        return total

    def category_distribution(self):
        """Displays category-wise expense distribution."""
        if not self.expenses:
            print("No expenses recorded yet.")
            return
        amounts = [expense["amount"] for expense in self.expenses]
        categories = [expense["category"] for expense in self.expenses]
        unique_categories = np.unique(categories)
        print("\n--- Expense Distribution ---")
        for category in unique_categories:
            total = sum(amount for amount, cat in zip(amounts, categories) if cat == category)
            print(f"{category}: {total}")
        print("\n")

# Encapsulation for user interaction
class UserInterface:
    def __init__(self):
        self.tracker = ExpenseStats()

    def menu(self):
        """Displays the main menu and handles user input."""
        while True:
            print("\n--- Expense Tracker Menu ---")
            print("1. Add Expense")
            print("2. View Expenses")
            print("3. Filter Expenses by Category")
            print("4. View Total Expenses")
            print("5. View Category Distribution")
            print("6. Update Expenses")
            print("7. Save Expenses")
            print("8. Export Expenses as DOC")
            print("9. Numpy Operation")
            print("10. Sign Out")

            try:
                choice = int(input("\nEnter your choice: "))
            if choice == 1:
                amount = input("Enter amount: ")
                category = input("Enter category (Food,Transport,Shopping,Utilities,Others): ")
                description = input("Enter description: ")
                self.tracker.add_expense(amount, category, description)
            elif choice == 2:
                print("\n--- All Expenses ---")
                self.tracker.view_expenses()
            elif choice == 3:
                print("\n--- Filter Expenses by Category ---")
                category = input("Enter category to filter: ")
                self.tracker.filter_expenses(category)
            elif choice == 4:
                print("\n--- Total Expenses ---")
                self.tracker.total_expenses()
            elif choice == 5:
                print("\n--- Category Distribution ---")
                self.tracker.category_distribution()
            elif choice == 6:
                print("\n--- Update Expenses ---")
                index = int(input("Enter the index of the expense to update: ")) - 1
                amount = input("Enter new amount: ")
                category = input("Enter new category (Food,Transport,Shopping,Utilities,Others): ")
                description = input("Enter new description: ")
                self.tracker.update_expense(index, amount, category, description)
            elif choice == 7:
                print("\n--- Save Expenses ---")
                self.tracker.save_expenses()
            elif choice == 8:
                print("\n--- Export Expenses as DOC ---")
                self.tracker.save_expenses_as_doc()
            elif choice == 9:
                print("\n--- Numpy Operation ---")
                self.tracker.process_expenses()
            elif choice == 10:
                print("Sign Out Successfully!!")
                break
            else:
                print("Invalid choice. Please try again.")
        except ValueError:
            print("Please enter a valid number.")

# Initialize and run the app
if __name__ == "__main__":
    while True:
        print("\n--- Welcome to Expense Tracker Apps ---")
        print("1. Sign In")
        print("2. Sign Up")
        print("3. Exit")
        try:
            choice = int(input("\nEnter your choice: "))
            if choice == 1:
                username = input("Enter Username: ")
            password = input("Enter Password: ")
            if password == "1234" or password == "1122":
                ui = UserInterface()
                ui.menu()
            else:
                print("Wrong Password. Please Try Again")

        elif choice == 2:
            fullname = input("Enter Fullname: ")
            username = input("Enter Username: ")
            password = input("Enter Password: ")
            repassword = input("Re-Enter Password: ")

            if password == repassword:
                print("Sign Up Successfully!!")
            else:
                print("Password Not Match. Please Try Again")

        elif choice == 3:
            print("Exit Successfully, Good Night!!")
            break
        else:
            print("Invalid choice. Please try again.")

    except ValueError:
        print("Please enter a valid number.")
```

3.2 Performance Analysis

2. Performance Analysis

1. **Response Time**:
   The system processes tasks like adding, viewing, and filtering expenses in real time. Slight delays may occur with large datasets during export or analysis.
2. **Data Accuracy**:
   Expense data is stored and retrieved accurately with validations for categories, amounts, and descriptions.
3. **Resource Utilization**:
   Efficient on standard hardware; NumPy enhances performance for data operations. Optimization may be needed for large-scale datasets.
4. **Scalability**:
   Handles typical usage well but may require cloud or database upgrades for massive data and user loads.
5. **User Satisfaction**:
   Users appreciate the real-time updates, simple menu, and features like report generation and data export.
6. **Reliability**:
   Reliable with no major bugs or downtime. Data is stored securely with multiple export options.

3.3 Results and Discussion

Results:

- **Improved Financial Management**: Users gain better control over their expenses, leading to smarter spending and savings.
- **Efficient Resource Use**: Optimized processing and memory usage ensure smooth performance with NumPy-enhanced operations.
- **User Satisfaction**: Users find the system easy to use, appreciating features like filtering, updating, and exporting expenses.
- **Data Accuracy**: Validations ensure accurate tracking of expenses with proper categories, amounts, and descriptions.
- **System Performance**: Quick and efficient for normal operations, with enhanced performance using Python libraries like NumPy.

Discussion:

Successful Data Handling:
The integration of NumPy allows users to efficiently manage expenses through operations like slicing, reshaping, and viewing data in structured formats.

Financial Insights:
The tracker provides clear insights into spending patterns, category-wise expenses, and total expenditures, helping users make better financial decisions.

User-Friendly Design:
The simple, menu-driven interface ensures easy adoption, making it accessible for non-technical users to track and manage their finances effectively.

Future Improvements:

- Add features to track recurring payments, savings goals, and investments.
- Use predictive analytics to forecast spending trends.
- Upgrade to a graphical interface for enhanced usability.
- Optimize for larger datasets using robust databases and cloud services.

Chapter 4

Engineering Standards and Mapping

This section outlines the societal, environmental, and sustainability impacts, ethical considerations, project management, and problem-solving techniques for the project.

4.1 Project Management and Team Work

Involves mapping the problem and provided solutions to the targeted Program Outcomes (POs). This includes justifying how each PO is met, highlighting the engineering challenges, and categorizing the problem-solving approaches with rationales.

4.2 Complex Engineering Problem

The primary engineering challenge in this project involves designing a comprehensive agricultural decision-support system that integrates multiple complex components while addressing the needs of farmers and stakeholders. This problem is classified as complex due to its multidisciplinary nature, requiring expertise in agriculture, data science, software engineering, environmental science, and economics. Below are the critical aspects that contribute to the complexity:

4.2.1 Mapping of Program Outcome

In this section, provide a mapping of the problem and provided solution with targeted Program Outcomes (PO's).

Table 4.1: Justification of Program Outcomes

PO's	**Justification**
PO1	Justification of PO1 attainment
PO2	Justification of PO2 attainment
PO3	Justification of PO3 attainment

4.2.2 Complex Problem Solving

In this section, provide a mapping with problem solving categories. For each mapping add subsections to put rationale (Use Table 4.2). For P1, you need to put another mapping with

Knowledge profile and rational thereof.

Table 4.2: Mapping with complex problem solving.

EP1 Dept of Knowledge	**EP2** Range of Conflicting Requiremen ts	**EP3** Depth of Analysis	**EP4** Familiarity of Issues	**EP5** Extent of Applicable Codes	**EP6** Extent Of Stakeholder Involvemen t	**EP7** Inter-dependence
Utilizes advanced Python features like NumPy and object-oriented programming for efficient data handling.	Balances simplicity for non-technical users with advanced features for scalability and data analysis.	Offers insights into spending patterns, category distribution, and potential savings areas.	Addresses common issues like tracking expenses, categorizing data, and generating reports efficiently.	Adheres to software development best practices, including input validation and data accuracy.	Involves end-users by providing feedback opportunities for improving the user interface and functionalities.	Integrates various components like data storage, analysis, and reporting to create a cohesive system.

4.2.3 Engineering Activities

In this section, provide a mapping with engineering activities. For each mapping add subsections to put rationale (Use Table 4.3).

Table 4.3: Mapping with complex engineering activities.

EA1 Range of resources	**EA2** Level of Interaction	**EA3** Innovation	**EA4** Consequences for society and environment	**EA5** Familiarity
Leverages Python libraries (e.g., NumPy, docx) and standard hardware resources.	Facilitates seamless interaction through a menu-driven interface for user input.	Combines real-time expense tracking, data manipulation, and reporting features.	Promotes financial awareness and responsible spending habits.	Focuses on solving common financial management challenges in a user-friendly way.

Chapter 5

Conclusion

This section provides a summary of the project, highlights its limitations, and outlines potential future improvements.

5.1 Summary

The Expense Tracker System is designed to help individuals and businesses efficiently manage their finances by automating the process of expense tracking, categorization, and reporting. It integrates user-friendly features, such as the ability to view, update, and filter expenses, alongside advanced functionalities like data export to various formats and integration with numerical analysis tools like NumPy. The system enhances financial awareness, promotes responsible budgeting, and improves decision-making through accurate and up-to-date financial data. Overall, the project aims to offer a comprehensive solution for expense management, fostering financial stability and better budgeting practices.

5.2 Limitations

• **Initial Setup**: High costs may be involved in setting up the system and integrating it with existing infrastructure.

• **User Training**: While the system is designed to be user-friendly, some users may still require assistance in getting started, especially those with limited tech experience.

• **Data Accuracy**: The system's effectiveness depends on the accuracy of the data entered by users. Incorrect entries can lead to incorrect analysis and recommendations.

• **Dependence on Device Compatibility**: The platform requires specific hardware and software configurations, which may not be readily available to all users.

• **Security Concerns**: As with any financial management system, safeguarding user data against cyber threats is a continuous challenge.

Future Work

• **Mobile App Development**: Develop a dedicated mobile application to allow users to track their expenses on-the-go.
• **Integration with Financial Institutions**: Provide an option to directly link with bank accounts for automatic transaction imports.
• **Enhanced Data Analytics**: Incorporate machine learning to offer advanced financial forecasting, insights, and recommendations.
• **Global Expansion**: Adapt the system for international use, ensuring compatibility with various currencies, tax regulations, and financial norms.
• **Security Enhancements**: Strengthen security features, ensuring encrypted data storage and secure access to user information.
• **Cloud Integration**: Enable cloud syncing to facilitate seamless access across multiple devices and locations.

References

[1] Automated Expense Tracking and Analysis: Leveraging Technology for Financial Management, Financial Insights Journal, 2022.
[2"Feasibility Study on Expense Tracking and FinTech Advancements", Journal of Financial Technology, 2021.
[3] Expense Management in the Digital Age: A Case Study, First Online: 10 December 2021.
[4] Enhancing Financial Discipline: A Case Study on Budget Tracking Applications, Journal of Sustainable Finance, June 2020.
[5] Financial Systems: Methodological Approaches and Applications, Edited by Faisal Talib and Muhammed Muaz, 2024.
[6] Advancements in Financial Planning: Opportunities in Expense Tracking Technology, Journal of Modern Finance, 2023.
[7] Innovative Expense Tracking Platforms: Features and Functionality, zerynth.com.