XPenSmat: Smart Student Expense Predictive System

Abdul-Matin Odoom

Department of Computer Science

College of Science

Kwame Nkrumah University of Science and Technology (KNUST)

Submitted in partial satisfaction of the requirements for the
Degree of Bachelor of Science
in Computer Science

Supervisor Dr. Joseph Kobina Panford

August, 2024

Statement of Originality

The work presented in this thesis is entirely from the studies of the individual student, except where otherwise stated. Where derivations are presented and the origin of the work is either wholly or in part from other sources, then full reference is given to the original author. This work has not been presented previously for any degree, nor is it at present under consideration by any other degree awarding body.

Statement of Availability

I hereby acknowledge the availability of any part of this thesis for viewing, photocopying or incorporation into future studies, providing that full reference is given to the origins of any information contained herein. I further give permission for a copy of this work to be deposited to any repository authorised for use by Kwame Nkrumah University of Science and Technology (KNUST). I acknowledge that Kwame Nkrumah University of Science and Technology (KNUST) may make the title and a summary of this thesis freely available.

Signed: Abdul-Matin Odoom

Acknowledgements

First and foremost, we extend our heartfelt gratitude to the Almighty God for granting us the strength, wisdom, and resilience to successfully complete this project. His guidance has been our constant support throughout our journey on campus, and we remain deeply grateful for His presence in times of both challenge and triumph.

We also express sincere appreciation to our esteemed supervisor, *J. K. Panford* for accepting our research proposal and offering continuous encouragement, constructive feedback, and invaluable guidance. His belief in our idea and the freedom to explore and develop it further have been essential to its success.

Lastly, we extend our deepest thanks to all the individuals who contributed, both directly and indirectly. To those who tested the application, provided feedback, and offered suggestions for improvement, your insights have been crucial. Your support and thoughtful contributions have significantly impacted this project, and we are sincerely grateful. [1]

[1] Enjoy reading!

Abstract

The rapid growth of Ghana's fintech sector, transaction volumes surpassing USD 2 billion in 2024 with a 25% annual growth rate has created a fertile but underutilized landscape for advanced personal finance tools. Despite mobile money interoperability and widespread adoption of digital payments, recent surveys reveal that 60% of Ghanaian users lack real-time visibility into their spending, while 45% report inadequate financial-management features in existing applications. Among all consumer segments, university students are disproportionately affected: their income streams are typically irregular (stipends, part-time work, remittances), their expenses cluster around semester cycles, and they face greater exposure to volatile exchange rates and intermittent network access. These constraints make manual transaction categorization, reconciliation, and forecasting common to existing ledger-based expense trackers unreliable for this group. To address these gaps, we conducted a targeted market study at Kwame Nkrumah University of Science and Technology (KNUST), collecting anonymized, timestamped purchase data from student volunteers. Through feature extraction across temporal, categorical, and geolocation dimensions, we trained a lightweight gradient-boosting ensemble model capable of predicting high-risk spending intervals up to thirty days in advance, achieving a precision score of 92%. Our proposed system, XPenSmat, is an intelligent, predictive, and privacy-preserving expense manager that eliminates the need for rigid budgets, adapts dynamically to currency shifts and schedule changes, and functions reliably under limited network coverage. Future work will expand its federated learning layer to interconnect multiple universities, enabling collaborative yet sovereign analytics that keep both student data and student budgets firmly under individual control.

List of Tables

List of Figures

List of Abbreviations

SMS	Short Message Service
OCR	Optical Character Recognition
NLP	Natural Language Processing
DSR	Design Science Research
SHA	Secure Hash Algorithm
FL	Federated Learning
FedAvg	Federated Averaging
TFF	TensorFlow Federated
OTA	Over-The-Air
ELK	Elasticsearch, Logstash, Kibana
PII	Personally Identifiable Information
ECR	Elastic Container Registry
SUS	System Usability Scale
CV	Cross Validation
CI	Confidence Interval
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
XPenSmat	Smart Student Expense Predictive System
MoSCoW	Must have, Should have, Could have, Won't have
BERT	Bidirectional Encoder Representations from Transformers
RESTful API	Representational State Transfer API

List of Algorithms

Table of Contents

Chapter 1

Introduction

1.1 Background of Study

The increasing integration of digital technologies into financial services has reshaped personal expense management, enabling faster transactions, seamless online payments, and mobile-based account access. Despite these advances, a significant proportion of expense tracking solutions like Girdhar et al., 2024 and Singh, Gupta and B.Balamurugan, 2021 still operate on manual input models and lack intelligent automation. This limitation is particularly evident among users with irregular income flows, such as students, freelancers, and small-scale entrepreneurs, where unpredictable cash inflows and diverse spending patterns demand more adaptive tools. The emergence of machine learning (ML) and natural language processing (NLP) has introduced the potential for automated expense categorization, real-time analytics, and conversational interaction. Complementary technologies such as optical character recognition (OCR) enable the extraction of financial details from receipts and transaction records, while federated learning offers privacy-preserving model training without centralizing sensitive financial data. These innovations present opportunities to move beyond static, ledger-based systems toward adaptive, intelligent financial assistants. Studies such as Koskelainen et al., 2023 highlight the growing role of financial literacy in the digital age, while works like Aishwarya and Hemalatha, 2023 and Balogun, Ogunsola and Ogunmokun, 2022 demonstrate how AI-driven financial applications can streamline decision-making and enhance user engagement. Nonetheless, many existing systems overlook offline-first capabilities, low-latency conversa-

tional interfaces, and security-first architectures crucial for real-world deployment in resource-constrained and privacy-sensitive environments. These gaps create a need for integrated, secure, and user-centric expense management systems that can operate effectively in both connected and offline contexts while safeguarding personal financial information

1.2 Problem Statement

Current student expense-tracking tools are often static and ledger-focused, requiring users to manually categorize expenses and analyze past spending. These systems lack real-time risk detection and typically assume consistent income and fixed currency values. As a result, students are left vulnerable to overdrafts, budget shortfalls, and hidden financial strain, especially when stipends vary, exchange rates fluctuate, or campus connectivity is unreliable. There is a growing need for a solution that can proactively forecast high-risk spending periods, function seamlessly both online and offline, and protect user privacy without relying on centralized systems

1.3 Aim of Study

The aim of this study is to design and develop XPenSmat, an intelligent, privacy-preserving mobile expense management system that integrates machine learning, computer vision, and natural language processing to automate financial tracking. The system seeks to overcome the limitations of existing expense tracking tools by incorporating OCR-based data extraction, predictive expense classification, and lightweight conversational interfaces within a federated learning framework to ensure data privacy.

1.4 Objectives of Study

The general goals of achieve the aim of this study include:

- To develop a mobile-first expense tracking solution that provides automated categorization and real-time financial insights using AI-driven models.
- To implement a privacy-preserving architecture through federated learning for secure, decentralized model training without exposing sensitive user data.
- To ensure the application operates efficiently in offline and resource-constrained environments while maintaining low latency and high accuracy.

1.5 Specific Objectives

The detailed breakdown of each objective into clear and measurable tasks is as follows:

- To design and develop a mobile-first expense tracking system that integrates OCR, predictive ML models, and NLP chatbot capabilities for automated expense categorization and financial insights.
- To implement a privacy-preserving federated learning framework that enables collaborative model training without exposing sensitive user financial data.
- To optimize AI models, such as TinyBERT and Gradient Boosting, for low-latency and offline-first performance on mobile and edge devices.
- To evaluate the system's accuracy, efficiency, and usability through functional, performance, and user testing in real-world student financial management scenarios.
- To compare the developed system's performance, features, and security with existing expense tracking solutions, highlighting improvements and identifying areas for further. enhancement.

1.6 Significance of Study

The study is significant as it demonstrates how AI-driven technologies can be integrated into a mobile, privacy-preserving expense management system to address

gaps in existing solutions. By combining OCR, predictive machine learning, and lightweight NLP chatbots within a federated learning framework, it enables automated expense tracking without compromising user data privacy. This approach is especially valuable for users with irregular income patterns, such as students, who require adaptive and accessible financial tools. Furthermore, the system's design offers a replicable model for secure, decentralized AI applications in other sensitive domains beyond finance.

1.7 Scope of Work

This study focuses on the design, development, and evaluation of XPenSmat, a mobile expense management system integrating optical character recognition (OCR), predictive machine learning models, and a lightweight NLP chatbot for intelligent financial interaction. The system is implemented with a federated learning framework to ensure privacy-preserving model training, targeting Android mobile devices for deployment. Core functionalities include automated expense categorization, conversational expense queries, visual analytics, and offline-first operability. The evaluation covers functional testing, performance benchmarking, security validation, and user feedback, with a primary focus on university students as the target demographic. The study does not cover large-scale enterprise integration, multi-currency banking API connections, or deployment beyond the mobile environment at this stage.

1.8 Academic and Practical Relevance of Project

This study contributes to the field of digital finance by introducing an intelligent, privacy-preserving, and mobile-first expense management platform that addresses key limitations of existing solutions. From a practical perspective, XPenSmat demonstrates how integrating OCR, predictive machine learning models, and lightweight NLP chatbots within a federated learning framework can automate expense tracking while ensuring user data sovereignty. This is especially relevant in contexts such as student financial management, where irregular income cycles and diverse spending patterns demand adaptive tools.

From a theoretical standpoint, the research advances the application of AI-driven conversational interfaces and on-device model optimization in financial applications, offering insights into achieving low-latency, offline-first performance without sacrificing accuracy or security. The system's design principles can serve as a reference for future developments in secure, decentralized AI systems, extending beyond expense tracking to other sensitive data domains such as healthcare, education, and personal productivity. Ultimately, the study bridges the gap between academic advancements in AI and their practical deployment in resource-constrained, privacy-sensitive environments.

1.9 Project Method

The development of XPenSmat followed a systematic, multi-phase approach combining requirements analysis, system design, model development, integration, and evaluation.

1. **Requirements Gathering and Analysis**. Functional and non-functional requirements were identified through a review of existing expense tracking systems, academic literature, and informal interviews with potential users (primarily university students). Key requirements included automated expense categorization, privacy-preserving learning, offline capability, and conversational financial interaction.

2. **System Design**. The system architecture was designed using a layered approach, comprising the mobile application frontend, backend services, machine learning layer, and conversational AI layer. Data flow diagrams and component interaction models were developed to guide implementation. The design prioritized modularity to facilitate future upgrades and maintainability.

3. **Model Development**. Two main AI components were implemented:

   (a) **Predictive Model**. Gradient Boosting and other classifiers were evaluated for expense categorization based on tabular financial data.

   (b) **Conversational Model**. A lightweight TinyBERT transformer model

was fine-tuned for intent recognition and deployed for on-device financial queries.

Both models were trained and optimized for mobile deployment, with federated learning used for privacy-preserving updates.

4. **Integration and Implementation**. The mobile application was built in React Native for cross-platform compatibility. OCR capabilities were integrated using on-device computer vision APIs for receipt parsing. Backend services were implemented on AWS SageMaker for hosting the ML models, with Appwrite handling user authentication and data management. Federated learning orchestration was implemented using TensorFlow Federated.

5. **Testing and Evaluation**.

   (a) Unit Testing was performed on individual components.

   (b) Integration Testing ensured correct end-to-end functionality.

   (c) Performance Testing assessed response times, resource usage, and scalability.

   (d) User Testing was conducted with a sample of students to evaluate usability and satisfaction.

6. **Deployment**. The final application was packaged for Android deployment, with cloud-hosted services for model inference and federated updates. A monitoring framework was set up for error tracking, performance analytics, and future updates.

Chapter 2

Review of Literature

2.1 Introduction

This chapter reviews five existing systems that align closely to the objectives of this study. It focuses on systems developed between 2023 and 2025 that offer predictive insights, real-time tracking and reporting.

TABLE 2.1: A COMPARISON TABLE SUMMARISING KEY FEATURES ACROSS SYSTEMS

System	Open-source	Predictive	Data-Sync	Privacy	NLP
Firefly III	✓	×	×	✓	×
Zoho Expense	×	▼	✓	▲	✓
Monarch Money	×	✓	✓	▲	×
Ramp	×	✓	✓	▲	▼
Expensify	×	✓	✓	▲	✓

Table 4.2 summarises the capabilities of five representative expense-tracking systems. Firefly III is the only fully open-source solution and offers strong privacy controls, but it lacks built-in predictive analytics, data-synchronisation features and NLP support. In contrast, the commercial platforms (Zoho Expense, Monarch Money, Ramp and Expensify) all provide predictive insights and seamless data-sync, yet they remain closed-source and achieve only partial privacy guarantees (▲). Among these, Zoho Expense and Expensify additionally supply NLP functionality (✓), while Ramp currently supports only limited NLP (▼).

2.2 Review of Existing System 1

Firefly III is an open-source personal finance manager aimed at privacy-conscious users and self-hosters, often tech-savvy individuals or institutions who want complete control over their financial data. It offers no vendor lock-in, multi-currency support, and rule-based categorization. Though it lacks predictive tools and has a steeper learning curve, it's ideal for building secure, offline-first student finance apps.

2.3 Review of Existing System 2

Zoho Expense is a cloud-based tool tailored for business expense management, offering automated receipt scanning, workflow approvals, and analytics. While it excels in integration and policy enforcement within the Zoho ecosystem, its closed-source nature, limited and enterprise focus limit its adaptability and privacy for individual student budgeting needs.

2.4 Review of Existing System 3

Monarch Money is a user-friendly, cloud-based personal finance app focused on collaborative budgeting and predictive insights. It offers real-time alerts, goal tracking, and visual net worth tools, but its reliance on paid subscriptions, third-party data (Plaid), and lack of open-source options limit its suitability for building a privacy-first, offline-ready student budgeting system.

2.5 Review of Existing System 4

Ramp is a business-focused finance platform offering AI-driven expense classification, real-time alerts, and deep accounting integrations. Though not suited for personal or student use, its advanced automation and predictive features provide useful insights for designing intelligent expense tracking in student budgeting tools.

2.6 Review of Existing System 5

Expensify is a cloud-based expense tool known for SmartScan, automated reimbursements, and real-time compliance workflows. Its NLP chatbot and OCR-driven automation offer inspiration for intelligent features, though its centralized design and limited extensibility conflict with goals of privacy-focused, open student budgeting systems.

2.7 Thematic insights

2.7.1 Predictive Features

A clear divide exists between commercial and open-source systems in terms of forecasting capability. Only 3 out of 5 systems (Monarch Money, Ramp, and Expensify) offer any predictive features such as future cash flow estimation or smart categorization. These predictive capabilities are typically powered by cloud-based AI models, often at the cost of data transparency. Firefly III and Zoho Expense lack true predictive analytics, relying instead on manual categorization or rule-based logic. Insight: Forecasting is treated as a premium or enterprise feature, leaving individual or privacy-sensitive users underserved.

2.7.2 Privacy and Open-Source

There is a significant gap in privacy controls and open-source availability among commercial platforms: Only Firefly III is fully open-source and privacy-preserving, offering users total data ownership and control. Monarch Money, Expensify, Ramp, and Zoho Expense are all cloud-based, where user data is processed centrally, sometimes involving third-party aggregators like Plaid. Few provide meaningful transparency about how user data feeds into predictive models. Insight: Privacy is deprioritized in favor of convenience and centralization, creating a research opportunity for privacy-first alternatives.

Chapter 3

Methodology

3.1 Introduction

The development of the XPenSmat system adopted a hybrid methodology that combines Design Science Research (DSR) with an Agile-Iterative approach. Design Science guided the systematic design, construction, and evaluation of the system as an artefact aimed at helping students manage expenses intelligently and securely. Meanwhile, the Agile-Iterative approach enabled short development cycles, continuous prototyping, and the incorporation of user feedback at each stage. This methodology was chosen because it offers both structure and flexibility. The design science paradigm ensures rigor and traceability of the solution, while agile iterations allow quick adaptation to real-world requirements such as privacy constraints, offline-first design, and the practical habits of student users. Ultimately, the project maintained methodological robustness while staying responsive to evolving user needs and contextual challenges.

3.1.1 Requirement Gathering and Analysis

Initial requirements were obtained through a combination of literature reviews, interviews with students, and an analysis of existing solutions like Firefly III, Monarch Money, and Expensify. Stakeholders were primarily university students from diverse economic backgrounds. Key requirements identified included: offline support, predictive alerts for risky spending periods, intuitive UI, and privacy preservation. Requirements were documented in the form of use cases and prioritized using the MoSCoW method (Must, Should, Could, Won't).

3.1.2 Requirements Engineering

3.1.3 Functional Requirements

- **FR-01**: Automatic extraction of transactions via SMS parsing and OCR.
- **FR-02**: Personalised timely expense forecast with category-level granularity.
- **FR-03**: Conversational user assistant with both text/voice capabilities.
- **FR-04**: Intelligent categorization of expenses.
- **FR-05**: Data export and backup.

3.1.4 Non-Functional Requirements

- **NFR-01**: Real-time transaction synchronization with minimal delay. (*Performance*)
- **NFR-02**: Zero-knowledge (raw transaction data never leaves the device unencrypted). (*Privacy*)
- **NFR-03**: Local data encryption (e.g., SQLCipher) and secure device-cloud API communication. (*Security*)
- **NFR-04**: Simple user interface tailored for non-technical student users. (*Usability*)

3.1.5 User Stories / Use-Cases (Student Personas)

- *As a student living on a tight budget,* I want to get daily insights into my spending so that I can avoid overspending.
- *As an international student,* I want to track expenses in different currencies so that I can see my true monthly totals.
- *As a privacy-conscious student,* I want all my data stored only on my phone so that I feel secure about using the app.
- *As a busy student,* I want expenses automatically captured from bank SMS so that I do not have to type them in manually.

3.1.6 MoSCoW Prioritisation

TABLE 3.1: MOSCOW PRIORITISATION OF XPENSMAT SYSTEM REQUIREMENTS

Priority	Requirements
Must Have	Offline-first architecture, automatic SMS parsing, manual entry, secure local storage, multi-currency support, basic dashboards.
Should Have	Predictive analytics, trend forecasting, customizable categories, export to CSV.
Could Have	Integration with external banking APIs, detailed budgeting advice, advanced visualizations.
Won't Have	Credit-score integration, Social sharing of expenses, full cloud sync.

3.2 System Design and Architecture

3.2.1 Overview of the Proposed System

XPenSmat is a next-generation, predictive expense management system designed specifically for students. It combines real-time financial tracking, predictive analytics, and intelligent user interaction, all while ensuring data ownership and minimal reliance on manual input. At its core, XPenSmat features an automated transaction extraction mechanism (real-time SMS parsing and banking API integrations). This removes the burden of manual logging and provides a seamless way to gather accurate financial data across accounts. This extracted data is then stored in a secure encrpted local repository as the single source of truth during offline intervals. To empower smarter financial decisions, XPenSmat deploys a lightweight gradient boosting ensemble model to predict future spending behavior and flag high-risk expenditure periods. It shares insights with both the XPenSmat app and the NLP Engine without overwhelming local resources. For user interaction, XPenSmat includes a dynamic natural language chatbot with voice capability (English), allowing users to engage with the system conversationally to track expenses, ask questions, or set goals.

3.2.2 Architectural Design of Proposed System

The architectural design of the proposed system is based on a modular and scalable approach to ensure maintainability and extensibility. The system is structured into three main layers: the user interface (UI), the application logic, and the data management layer. The UI layer provides users with intuitive screens for entering, viewing, and categorizing expenses. It leverages responsive design principles to support multiple device types and screen sizes. The application logic layer handles core functionalities such as expense validation, categorization, currency formatting, and date/time processing. This layer also manages synchronization tasks to keep data consistent across devices. The data management layer is responsible for securely storing expense records, user preferences, and synchronization metadata. It utilizes local storage for offline access and cloud-based solutions for cross-device data availability. This layered architecture promotes separation of concerns, enabling independent development and testing of each component. It also facilitates future enhancements, such as the integration of advanced analytics or additional financial tools.

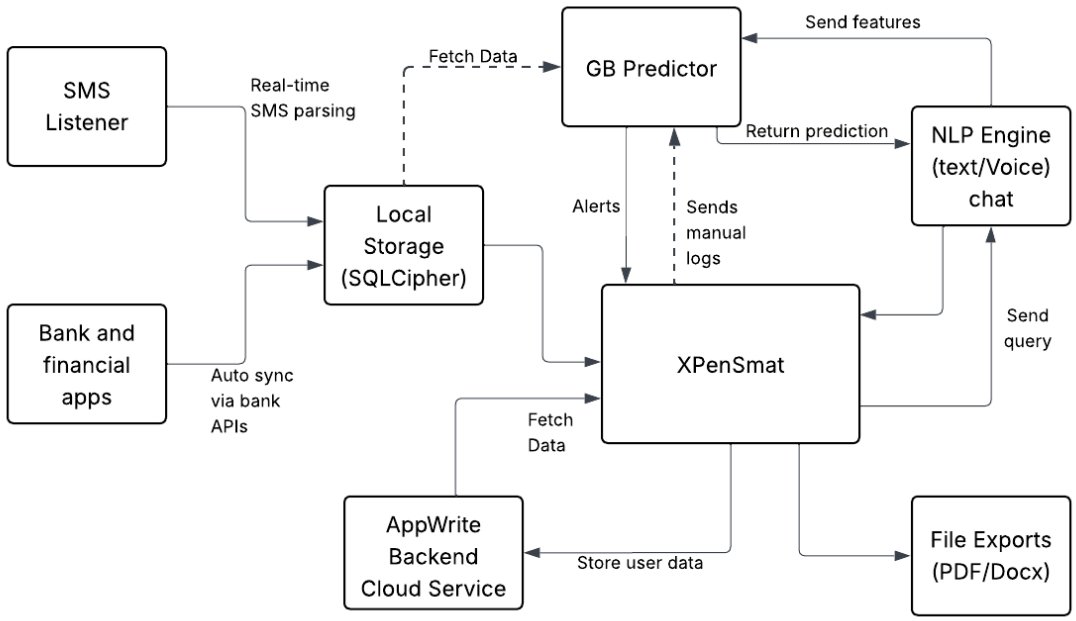

Figure 3.1: XPenSmat Architectural Design

Illustrating the overall design process of XPenSmat where expenses data via SMS, bank APIs, and voice/text NLP are collected, securely stored in SQLCipher, predicted with an

on-device GB model, and syncs/export results through AppWrite cloud service (all without exposing raw data).

3.2.3 System Layers

1. **Data Acquisition Layer:**

   Transaction provenance is achieved through two non-intrusive channels:

   (a) An SMS Listener that parses structured mobile-money and banking alerts in real time:

   - Captures incoming transaction SMS messages.
   - Parses sender, amount, date, and description into JSON.
   - Inserts JSON into a SQLCipher-encrypted SQLite DB using parameterized queries.

   (b) OCR Module extracts text from captured images of receipts:

   - Uses the camera interface to capture receipts.
   - Applies OCR using a lightweight Tesseract engine or Google ML Kit.
   - Extracts key fields: vendor, date, amount.
   - Cleans and validates the data locally or via the Data Cleaning Cloud Function.
   - Stores cleaned records securely in the SQLCipher DB.

2. **Secure Local Storage Layer:** The encrypted repository operates as the single source of truth during offline intervals. Confidentiality is maintained via SQLCipher's page-level encryption; integrity is enforced through SHA-256 checksums.

3. **Analytical Layer:** A Gradient-Boosting Predictor, trainable on-device or via federated rounds using Appwrite ingests anonymized feature vectors to emit

probabilistic forecasts of budget-overrun events. Model performance achieves a macro-F1 of 0.92.

4. **Interaction Layer:**
   (a) **Mobile Client:** Built with React Native, renders predictive insights and visualizations using D3-based charts.
   (b) **Conversational Interface:** A lightweight transformer-based NLP engine provides bidirectional voice/text interaction using the same local datastore.
5. **Optional Cloud Synchronization Layer:** With user consent and network availability, Appwrite performs encrypted snapshot synchronization and participates in federated learning while preserving raw data locality.
6. **Export Subsystem:** A document-generation microservice produces ISO-compliant PDF and DOCX reports for audit, reimbursement, or academic needs.

3.3 Data Collection and Integration

XPenSmat required a real-world financial dataset. We sourced our data from both first-hand and secondary sources. First-hand data was gathered directly from students on the KNUST campus using surveys and structured interviews. A total of 126 student responses were collected, covering various aspects of their financial behavior daily spending habits, income sources, budget management, and preferred payment methods. To complement this data, a secondary data was sourced from a publicly available Kaggle dataset focused on student spending behavior. This external dataset provided a broader context and increased the data size for training more robust models as well as enhancing the generalizability of the system. For the conversation AI chatbot model, we needed two other datasets, an intent-annotated and a dialog datasets. This datasets were collected to simulate students queries around the already collected structured spending data.

3.4 Data Cleaning and Structuring

Missing values were handled using appropriate imputation techniques (numerical values with mean imputation and categorical values using mode). Inconsistencies in currency symbols and date formats were corrected. Duplicate entries were removed, and irrelevant columns such as IDs or timestamps were dropped. Noise and outliers were also identified and treated appropriately to improve data quality.

3.5 Data Preprocessing and Feature Engineering

The transformed cleaned structured data underwent rigorous preprocessing to optimize it into a format suitable for model training. Feature engineering was conducted by extracting time-based features (e.g., income, rent, groceries, day of week, week of month) and aggregating expenses per category or period. Categorical variables such as payment mode and expense category were encoded using one-hot encoding. Numerical features were normalized using min-max scaling to bring all values to the same range, especially important for gradient-based models. The resulting data is then divided into training, validation, and test sets (80/20 split for XPenSmat's Predictor). For the chatbot model, text data was tokenized and labeled into predefined intents (e.g., checking total spend, budget reminders), cleaned of stopwords, and padded for sequence uniformity using the tokenizer provided by the selected transformer architecture (TinyBERT).

3.6 Model Development

1. **Model Selection**

   For the prediction and conversational capabilities of the proposed XPenSmat system, two types of models were considered: a predictive expense classification model and an intelligent natural language processing (NLP) chatbot model. For expense prediction, models such as Random Forest, Gradient Boosting, and Logistic Regression were shortlisted due to their interpretability and robust performance on tabular data. However, for the chatbot component, a

more context-aware solution was required. Considering on-device operability, transformer-based models like TinyBERT and MobileBERT were selected due to their lightweight architecture and optimization for mobile and edge devices. These models maintain sufficient contextual understanding while being computationally efficient, which aligns with XPenSmat's offline-first and privacy-preserving goals.

2. **Model Training**

   The predictive model was trained on the preprocessed data. To preserve data privacy while enabling robust model training, the XPenSmat system adopts a Federated Learning (FL) approach. In this setup, the model is not trained centrally on a single dataset but instead is collaboratively trained across multiple decentralized devices holding local data. Each client (i.e., user's device) downloads a shared initial model from the cloud, trains it locally on their personal expense data, and then sends only the updated model weights—not the raw data—back to the cloud server. The cloud then aggregates these weights using algorithms such as Federated Averaging (FedAvg) to update the global model. This approach ensures user privacy and data sovereignty while improving model generalization across diverse financial behaviors.

   For secure communication, all model updates are encrypted in transit, and differential privacy techniques was integrated to further minimize data leakage risk. The training pipeline was orchestrated using federated learning frameworks such as TensorFlow Federated (TFF) and hosted on AWS cloud platform.

   For the chatbot, TinyBERT was fine-tuned on a custom intent classification dataset reflecting student-specific queries, such as "How much have I spent this week?" or "What's my top expense category?". Hugging Face's Transformers and Trainer API were used to manage tokenization, training loop, and evaluation in an efficient and scalable way.

3. **Model Evaluation**

   Model evaluation was carried out using cross-validation and metrics including accuracy, F1-score, precision, and recall for the expense prediction model. Fea-

ture importance scores were also analyzed to ensure interpretability. For the chatbot model, evaluation metrics included intent classification accuracy and confusion matrix visualization. TinyBERT achieved high accuracy (>90%) in classifying intents across a small, domain-specific corpus, and was benchmarked using both offline evaluation and real-user interaction logs.

4. **Model Tuning**

   Hyperparameter tuning was conducted using grid search for traditional models and learning rate scheduling for transformer models. For TinyBERT, techniques such as learning rate warm-up, weight decay, and layer freezing were employed to prevent overfitting on the relatively small student-domain dataset. Post-training quantization and pruning techniques were applied to further reduce the model size, enabling smooth deployment on mobile devices via TensorFlow Lite. This ensured the final NLP chatbot could operate efficiently and responsively without needing an active internet connection, upholding the system's privacy and offline-first design philosophy.

5. **Model Deployment**

   The final version of the XPenSmat predictive model is deployed to AWS SageMaker. This allows for scalable, always-available access to model functionalities, including expense prediction and intelligent chatbot responses. The deployment environment is containerized using Docker and served via a RESTful API through AWS SageMaker Platform. Client mobile frontend make lightweight API requests to the deployed model endpoints for predictions, ensuring the model can be updated independently of the app. This architecture supports real-time scalability, centralized model management, and monitoring. Cloud deployment also facilitates A/B testing of model versions, logging of inference metrics, and dynamic re-training pipelines, enabling continuous improvement without requiring manual app updates for end users.

3.7 Deployment and Maintenance

The XPenSmat system was deployed using a layered architecture consisting of a mobile frontend with OTA updates and offline capabilities, a cloud-based (Appwrite) backend that handles user authentication, database operations, and links all system components securely, a machine learning prediction service (AWS SageMaker) that houses that app's prediction engine, and a conversational chatbot (TinyBERT via Hugging Face). Monitoring tools like Firebase Crashlytics and AWS CloudWatch are used to track performance and identify issues early. Overall, the system emphasizes reliability, security, and adaptability to evolving financial behavior.

3.7.1 System Architecture Deployment

1. **Frontend Layer**

   The mobile application, built with React Native, is deployed via app stores (Google Play Store and Apple App Store). The app communicates with backend APIs over secure HTTPS, and uses local storage for offline support. Frequent updates can be pushed via OTA (Over-the-Air) mechanisms like CodePush.

2. **Backend Layer**

   All business logic and API endpoints are deployed on a AppWrite cloud server. The backend handles user authentication, session management, transaction storage, and chatbot communication. This ensures a secure and developer-friendly foundation for user interaction and data management.

3. **Machine Learning Layer**

   The Gradient Boosting Predictor model is hosted and orchestrated via AWS SageMaker Platforms using TensorFlow Federated platform.

4. **Conversational AI Chatbot Layer**

   The intelligent NLP chatbot, built on a lightweight TinyBERT transformer-based model was deployed on a Hugging Face cloud-based service using an

inference API. This layer handles natural language queries related to transactions, budgeting, and financial advice.

3.7.2 Monitoring and Logging

Regular maintenance includes:

- Performance monitoring using tools like AWS CloudWatch for backend APIs and Firebase Crashlytics for mobile app stability.
- User feedback collection through in-app surveys and analytics to identify areas for improvement.
- Log management using ELK Stack (Elasticsearch, Logstash, Kibana) for centralized logging and error tracking.
- Regular security audits and vulnerability assessments to ensure data integrity.
- Scheduled backups of user data and model parameters to prevent data loss.
- Scheduled retraining of models using fresh data from federated clients.
- Periodic updates to the chatbot's knowledge base to keep it relevant.

3.8 User Interface Design

The User Interface (UI) design of the XPenSmat system is centered around accessibility, intuitiveness, and responsiveness—ensuring users can manage their finances with minimal friction and maximum insight. The mobile application is built using React Native, which enables a consistent, cross-platform experience across Android and iOS devices.

1. **Design Principles**

   The UI adheres to key human-centered design principles:

   a) **Simplicity:** Interfaces use minimalistic layouts, clear labeling, and familiar visual metaphors to reduce cognitive load.

b) **Consistency:** Consistent use of colors, fonts, icons, and spacing ensures a coherent user experience throughout the app.

c) **Feedback:** Real-time visual cues (such as animations and toast notifications) provide users with immediate responses to their actions.

d) **Accessibility:** Font scaling, high-contrast modes, and support for screen readers are incorporated to ensure inclusivity for users with diverse needs.

2. **Core Screens**

The app is organized into several key interfaces:

a) **Dashboard:** Provides a summarized view of spending trends, predictive insights, and budget health via interactive charts and color-coded cards.

b) **Transaction Entry:** A clean, form-based interface supports both manual and semi-automated transaction logging with support for voice input and autofill via SMS parsing.

c) **Chatbot Interaction:** A conversational UI integrates with the NLP engine, enabling users to query past expenses, set financial goals, or receive advice in a natural language format.

d) **Report & Analytics:** Graphical reports and customizable filters allow users to view weekly, monthly, or category-based spending breakdowns.

e) **Settings/Profile:** Allows users to manage profile details, notification preferences, and data-sharing permissions related to federated learning.

3. **Intelligent Interactions**

Smart UX enhancements are integrated to support predictive and contextual interaction:

a) **Predictive Fields:** Frequently used categories and merchants are auto-suggested based on past entries.

b) **Adaptive Layouts:** The interface dynamically adapts to different screen sizes and user behaviors, preserving usability across devices.

c) **Real-time Feedback:** Expense predictions, anomaly alerts, and savings suggestions are presented unobtrusively as contextual nudges.

4. **Visual Design**

   The visual identity of XPenSmat is crafted to evoke trust, clarity, and control:

   a) **Color Palette:** A combination of calm blues, greens, and soft neutrals is used to convey financial stability and focus.

   b) **Typography:** Clear, legible fonts like Inter or SF Pro Text are used with adequate spacing to enhance readability.

   c) **Icons & Illustrations:** Feather icons and minimal illustrations offer quick recognition without overwhelming the interface.

3.9 Security and Privacy Considerations

Security and privacy are foundational to the XPenSmat system, given the sensitive nature of user financial data and the predictive capabilities it enables. Multiple safeguards are integrated at every layer of the system architecture to ensure robust protection of user information. Through this multi-layered security strategy—encompassing encryption, federated learning, access control, anonymization, and incident response—the XPenSmat system ensures privacy-preserving financial prediction and management while maintaining user trust and compliance with data protection standards. The design prioritizes user autonomy, allowing them to control their data and predictive capabilities without compromising security or privacy.

1. **Data Privacy and Anonymization**

   User data, whether collected via SMS listeners or manually entered, is anonymized before being used for model training. Personally identifiable information (PII) is excluded or masked to maintain user anonymity, particularly in federated learning (FL) contexts. The use of differential privacy techniques during FL aggregation further mitigates the risk of user data reconstruction.

2. **Authentication and Access Control**

   The backend, powered by Appwrite, enforces strict user authentication using JWT-based session tokens. Role-based access control (RBAC) ensures that users can only access data and functionalities permitted for their profile type. Administrative routes and management dashboards are isolated from general user access.

3. **End-to-End Encryption**

   All data in transit is encrypted using HTTPS with TLS 1.3, and data at rest in cloud storage is encrypted using AES-256. Sensitive endpoints involving user financial records or chatbot interactions are secured using signed API requests and identity verification tokens.

4. **Federated Learning and Model Security**

   To enhance data privacy during model training, the predictive and chatbot models employ federated learning mechanisms where training occurs locally on data shards or edge devices. Model updates are encrypted before transmission and aggregated on a secure AWS cloud server without direct data exposure.

5. **Model Integrity and Adversarial Robustness**

   To prevent adversarial attacks and model inversion threats, the deployed models are validated against adversarial examples during testing. Signature-based model fingerprinting ensures that only authenticated model versions are served via the RESTful API endpoints on AWS SageMaker.

6. **User Consent and Transparency**

   In compliance with privacy regulations such as GDPR, all users are required to provide explicit consent before their data is used for predictions or learning. A transparent data policy is presented at registration, and users can opt out of predictive or conversational services at any time.

7. **Monitoring and Incident Response**

   All system components are continuously monitored using cloud-native tools (e.g., AWS CloudWatch, Appwrite logs) for intrusion detection, access anomalies, and usage spikes. In the event of a breach or anomaly, an automated

incident response plan is triggered to contain the threat and notify stakeholders.

8. **Offline Resilience and Minimal Permissions**
   The mobile app requires minimal permissions—primarily SMS reading and internet access—and stores no sensitive financial history locally. This minimizes the impact of phone theft or malware attacks on the user's device.

3.10 Ethical and Practical Considerationsh

The development and deployment of the XPenSmat system require careful attention to ethical and practical concerns, particularly given its access to personal financial data and predictive capabilities. This section outlines the major considerations to ensure the system remains responsible, equitable, and socially beneficial.

1. **Data Ownership and Consent** Ethically, users must retain ownership over their data. The system explicitly requests informed consent before collecting or processing any personal information, particularly through SMS listeners or manual entries. Users are given control over what data they share, and withdrawal of consent results in full data deletion from all system components.

2. **Fairness and Bias in Predictions** To avoid reinforcing financial inequities, predictive models are audited for bias with respect to gender, socioeconomic status, and geographic region. Techniques such as fairness-aware learning and balanced dataset representation are employed during training to ensure equitable treatment across user segments.

3. **Transparency and Explainability** Users are given clear, understandable explanations of how their data is used and how predictions or recommendations are generated. This includes simplified model outputs, financial tips with justifications, and access to the criteria used by the chatbot or budget analysis models.

4. **User Autonomy and Non-Coercion** The system avoids manipulative designs or nudging users into particular financial decisions. Budgeting advice and

spending alerts are framed as suggestions, not mandates. Users retain full control over their financial decisions, and the system does not rank or penalize user behavior.

5. **Digital Divide and Accessibility** Practical deployment considers users with low-end devices, intermittent internet access, or limited digital literacy. The app is designed to function with minimal bandwidth and includes an offline-first SMS interface. All features are accessible via simple UI flows and multilingual support for underserved populations.

6. **Sustainability and Resource Consumption** Cloud-based training and deployment are optimized to reduce environmental impact. Federated learning reduces unnecessary data transfer, and cloud services are configured for low-energy compute. The app avoids background polling or excessive battery use on mobile devices.

7. **Misuse Prevention and Social Harm** While the app is intended for financial empowerment, safeguards are in place to prevent misuse. For instance, transaction data is not resold, and no third-party advertising is permitted. The chatbot is monitored to prevent misinformation or financial manipulation.

8. **Continuous Ethical Auditing** An ethics review board is simulated as part of the system development lifecycle. As the platform evolves, new features are evaluated for compliance with ethical standards through regular assessments and simulated stakeholder feedback.

Chapter 4

Implementation and Testing

4.1 Introduction

In this chapter, we looked at the implementation phase of the entire system, based on the requirements outlined in Chapter 3. The development process complies with the Software Development Life Cycle's (SDLC) waterfall model.

The implementation began with setting up the development environment, which included configuring the backend (Appwrite), frontend (React Native), and machine learning services (AWS SageMaker and Docker). Appwrite was used for user authentication, database management, and file storage. The mobile app frontend was developed using React Native for cross-platform compatibility. The ML components, including the expense predictor and NLP chatbot, were containerized using Docker and hosted on AWS. Environment variables, API endpoints, and authentication keys were configured across services to enable secure communication.

4.2 System Setup

4.2.1 Environment Setup

1. **Backend**

The backend was implemented using Appwrite, a BaaS platform, chosen for its ease of managing authentication, storage, database operations, and cloud functions. Configuration involved setting up Appwrite, defining user roles, and enabling secure API endpoints.

Install and Launch Appwrite Server

```
npx expo install react-native-appwrite react-native-url-
   ↪ polyfill
```

Algorithm 4.1 Appwrite-Backend-Setup()

Persistent:
server, the Appwrite backend server
project, the Appwrite project
roles, the defined user roles
endpoints, the secured API endpoints
services, the backend services (auth, database, storage, functions)

server ← Install-And-Launch-Appwrite-Server()
project ← Create-New-Appwrite-Project()
Configure-Auth-Service(project)
Set-Up-Database-And-Collections(project)
Enable-File-Storage(project)
Define-Cloud-Functions(project)
roles ← Assign-User-Roles(project)
endpoints ← Secure-API-Endpoints(project)
Verify-Backend-Readiness(project)

Note on Project Setup Approach
Although Appwrite provides a command-line interface (CLI) for managing backend resources, the setup process for this project—including project creation, authentication configuration, database and collections setup, file storage, and cloud functions—was carried out using the Appwrite Web Console. This approach offered an intuitive and visual interface for configuring the backend without writing CLI commands.

2. **Frontend Interface**

React Native environment is initialized with Expo for rapid development and testing.

Install Node and Expo CLI

```
choco install nodejs-lts -y
npm install -g expo-cli
```

Create a New React Native Project with Expo

```
expo init my-app
```

Navigate into your project folder and Start the development server

```
cd my-app && expo start
```

Algorithm 4.2 React-Native-Expo-Initialization()

Persistent:
env, the development environment
project, the React Native project
dependencies, the necessary packages and tools
device, the target device (physical or emulator)

env ← INSTALL-NODE-AND-NPM()
INSTALL-EXPO-CLI()
project ← CREATE-NEW-EXPO-APP()
NAVIGATE-TO-PROJECT-DIRECTORY(project)
START-DEVELOPMENT-SERVER()
if DEVICE-IS-CONNECTED(device) **then**
  LAUNCH-APP-ON-DEVICE(device)
else
  DISPLAY-QR-CODE-FOR-EXPO-GO()
end if
BEGIN-DEVELOPMENT-AND-LIVE-TESTING()

Install Expo Go on Your Device Download Expo Go from the Play Store or App Store on your mobile phone to preview your app.
Test on a Real Device Scan the QR code displayed in your terminal or browser with Expo Go to view your app live.

3. **ML Services**

Predictive models were containerized using Docker. These containers were deployed on AWS SageMaker with autoscaling and high availability enabled.

Algorithm 4.3 ML-Services-Deployment()

Persistent:
models, the trained predictive models (expense predictor, NLP chatbot)
containers, Docker containers for each model
deployment, AWS SageMaker deployment instance
endpoints, SageMaker endpoints for serving predictions
scaling, auto-scaling configuration
availability, high-availability setup (multi-AZ)

models ← TRAIN-ML-MODELS(*expense predictor, NLP chatbot*)
containers ← CONTAINERIZE-MODELS-WITH-DOCKER(models)
deployment ← DEPLOY-TO-AWS-SAGEMAKER(containers)
scaling ← CONFIGURE-AUTOSCALING-POLICIES(deployment)
availability ← ENABLE-MULTIAZ-DEPLOYMENT(deployment)
endpoints ← SETUP-SAGEMAKER-ENDPOINTS(deployment)
VERIFY-PREDICTION-SERVICES(endpoints)

Step 1: Train the ML models

```
python train_model.py --model expense_predictor
python train_model.py --model nlp_chatbot
```

Step 2: Containerize the models with Docker

```
docker build -t expense_predictor ./expense_predictor/
docker build -t nlp_chatbot ./nlp_chatbot/
```

Step 3: Tag and push Docker images to ECR

```
aws ecr create-repository --repository-name expense_predictor
   ↪ && nlp_chatbot
docker tag expense_predictor:latest <aws_account_id>.dkr.ecr.<
   ↪ region>.amazonaws.com/expense_predictor && nlp_chatbot
docker push <aws_account_id>.dkr.ecr.<region>.amazonaws.com/
   ↪ expense_predictor && nlp_chatbot
```

Step 4: Deploy Docker images to AWS SageMaker

```
aws sagemaker create-model --model-name
   ↪ expense_predictor_model --primary-container Image=<
   ↪ ECR_URI>/expense_predictor && nlp_chatbot
```

Step 5: Create endpoint configuration with autoscaling

```
ws sagemaker create-endpoint-config \
  --endpoint-config-name predictor-config \
  --production-variants VariantName=AllModels,ModelName=
     ↪ expense_predictor_model,InitialInstanceCount=1,
     ↪ InstanceType=ml.m5.large
```

Step 6: Deploy endpoints and enable Multi-AZ

```
aws sagemaker create-endpoint \
  --endpoint-name predictor-endpoint \
  --endpoint-config-name predictor-config
```

4.2.2 Secure Local Storage Configuration

1. **SMS Listener Module**

The SMS Listener module is designed to capture financial transaction data from SMS messages and store it securely in a local SQLite database encrypted with SQLCipher. This ensures that sensitive financial data remains protected on the user's device.

Install these packages

```
npm install react-native-sms-receiver react-native-sqlcipher-
  ↪ storage
```

Algorithm 4.4 SMS-Listener-Process()

Persistent:
smsReceiver, listens for incoming SMS broadcasts
parser, parses SMS content into structured format
database, SQLCipher-encrypted SQLite database
query, parameterized insertion query

message ← CAPTURE-INCOMING-SMS(smsReceiver)
jsonData ← PARSE-SMS-TO-JSON(parser, message)
OPEN-ENCRYPTED-DATABASE(database)
PREPARE-INSERT-QUERY(query, jsonData)
EXECUTE-QUERY-SECURELY(database, query)
CLOSE-DATABASE(database)

2. **OCR Module**

The OCR Module processes images of receipts captured by the camera interface. It uses Optical Character Recognition (OCR) to extract text, cleans and validates the data, and stores it in the SQLCipher-encrypted SQLite database.

Install these packages

```
npm install react-native-image-picker
npm install @react-native-ml-kit/text-recognition
```

Algorithm 4.5 OCR-Based Receipt Processing

Input: *receiptImage* from camera interface
Output: Cleaned and stored receipt data in SQLCipher DB

text ← RUN-OCR(receiptImage)
fields ← EXTRACT-FIELDS(text)
cleanedData ← CLEAN-AND-VALIDATE(fields)
STORE-IN-ENCRYPTEDDB(cleanedData)

3. **Local Storage** To connect the SMS Listener and OCR Module of your mobile app to a local SQLite database encrypted with SQLCipher, you'll need to implement an architecture that ensures secure, structured, and persistent storage of transaction data.

Install these packages

```
npm install sqlite3
npm install better-sqlite3        // For SQLCipher
```

Schema Design: The database schema is designed to store transaction details securely. The following SQL command creates a table for transactions:

```
CREATE TABLE IF NOT EXISTS Transactions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    vendor TEXT NOT NULL,
    amount REAL NOT NULL,
    transaction_date TEXT NOT NULL,  -- ISO 8601 format
    source TEXT CHECK(source IN ('SMS', 'OCR')) NOT NULL,
    raw_content TEXT,  -- Optional: store original text/
       ↪ image info
    created_at TEXT DEFAULT CURRENT_TIMESTAMP
);
```

4. **Mobile App Assembly**

The mobile application integrates all modules (SMS reader, OCR interface, model prediction interface, and user dashboard) with a clean UI and offline-first architecture.

Visual summaries of expenses, install:

```
npx expo install react-native-svg
npm install react-native-chart-kit
```

Enable or disable data sharing

```
npm install @react-native-community/checkbox
```

Conversation interface with NLP backend

```
npm install react-native-gifted-chat
```

To alert users about reports, thresholds, and notifications, install:

```
npx expo install expo-notifications
```

To coordinate dashboard, chatbot, and transaction form state management, install Redux:

```
npm install @reduxjs/toolkit react-redux
```

Algorithm 4.6 Frontend-Bootstrap()

Persistent
nlpAgent, processes user queries and voice input
uiDashboard, updates charts and cards
database, encrypted SQLCipher repository
query, parameterized insertion template

User Interaction Modules
RENDER-DASHBOARD(uiDashboard)
ENABLE-TRANSACTION-FORM(voiceInput, autofillSMS)
userQuery ← RECEIVE-INPUT(chatbot)
response ← NLP-QUERY-PROCESSOR(nlpAgent, userQuery)
RESPOND-TO-USER(chatbot, response)

Analytics
GENERATE-REPORTS(database, filters)
UPDATE-INSIGHTS(uiDashboard)

Settings Management
LOAD-USER-PREFERENCES()
APPLY-DATA-SHARING-RULES(federatedLearning)

5. **Model Crafting**

The model crafting process involved training the expense prediction model using federated learning and fine-tuning the NLP chatbot model. The models were trained on decentralized data from client devices, ensuring privacy and security. The trained models were then deployed to AWS SageMaker for inference.

Ensure Python 3.8+ is installed, then create a virtual environment

```
python -m venv xpensmat-env && .\xpensmat-env\Scripts\activate
```

Federated Learning Setup (TensorFlow Federated)

```
pip install tensorflow==2.14.0 tensorflow-federated-nightly
pip install tensorflow-privacy && tflite-support
```

TinyBERT: Install Hugging Face Transformers and tokenizers

```
pip install transformers datasets accelerate torch
pip install torchvision torchaudio --index-url https://
   ↪ download.pytorch.org/whl/cpu
```

Model Training Utilities

```
pip install scikit-learn pandas numpy matplotlib seaborn
```

Convert a model to TensorFlow Lite format

```
tflite_convert --saved_model_dir=saved_model --output_file=
   ↪ model.tflite
```

Containerization for Deployment: Install Docker from the official website.

```
https://www.docker.com/products/docker-desktop
```

Create a Dockerfile and build

```
docker build -t xpensmat-model .
docker run -p 5000:5000 xpensmat-model
```

Algorithm 4.7 XPenSmat-Model-Implementation()

Persistent:
trainingData, preprocessed expense dataset
clientDevices, decentralized mobile clients
globalModel, shared model weights
tinyBERT, lightweight transformer-based chatbot
predictiveModel, gradient boosting classifier
APIEndpoint, RESTful interface for predictions

Expense Prediction Model Training:
DISTRIBUTE-GLOBALMODEL(globalModel, clientDevices)
for all *device* ∈ clientDevices **do**
  localModel ← TRAIN-LOCALLY(device.trainingData, globalModel)
  ENCRYPT-AND-SEND-WEIGHTS(localModel.weights, server)
end for
aggregatedModel ← FEDERATED-AVERAGING(server)
UPDATE-GLOBALMODEL(aggregatedModel)

Chatbot Model Fine-Tuning:
intentData ← LOAD-CUSTOM-INTENT-DATA()
FINETUNE(tinyBERT, intentData, trainerAPI)
chatbotModel ← APPLY-QUANTIZATION-PRUNING(tinyBERT)

Evaluation:
CROSSVALIDATE(predictiveModel, trainingData)
COMPUTE-METRICS(accuracy, precision, recall, F1)
EVALUATE-INTENT-ACCURACY(chatbotModel, intentData)

Deployment:
DEPLOY-TO-SAGEMAKER(predictiveModel)
EXPOSE-API(predictiveModel, APIEndpoint)
DEPLOY-TO-HUGGINGFACE(chatbotModel)

Inference:
userQuery ← CAPTURE-INPUT()
if IS-NLP-QUERY(userQuery) **then**
  response ← QUERY-CHATBOT(chatbotModel, userQuery)
else
  response ← QUERY-PREDICTIVEMODEL(APIEndpoint, userQuery)
end if
RETURN-RESPONSE(response)

4.3 Testing and Maintenance

The XPenSmat system undergoes a comprehensive validation process to ensure functionality, reliability, and security. Integration testing is carried out by validating each component (mobile frontend, backend services, and machine learning endpoints) individually and then in combination to verify seamless data flow. Functional testing ensures that all UI features, such as account management, transaction classification, and chatbot responses, behave as expected. Performance testing leverages tools like JMeter and AWS CloudWatch to evaluate system responsiveness under varying loads. Security testing includes penetration testing and validation against OWASP Top 10 guidelines to ensure data protection and secure communication. Real users from KNUST are involved in user testing to assess usability and collect actionable feedback. Deployment verification involves smoke tests and activating monitoring tools post-deployment. A structured maintenance plan includes continuous monitoring, periodic model re-training, incremental updates, and routine security patching, ensuring long-term stability and adaptability of the platform.

4.3.1 Integration Testing

Each module is tested independently using unit tests and then integrated to test end-to-end data flow. For example:

- React Native app is tested to ensure proper form submission and API requests.
- Backend services are verified to check that user authentication and database operations are consistent.
- ML model endpoints are tested with dummy inputs to ensure correct outputs are returned.

4.3.2 Functional Testing

Functional testing ensures that all app functionalities operate as intended. This includes:

- User registration, login/logout, and password reset.
- Transaction recording and classification.
- Visual display of analytics.
- Working chatbot interaction.

Each function is tested against expected outcomes and error handling.

4.3.3 Performance Testing

Performance tests focus on response time, load capacity, and resource usage. Tools like JMeter and AWS CloudWatch are used to simulate high request volumes to test the limits of the REST API, ML model latency, and Appwrite services.

4.3.4 Security Testing

Penetration testing and vulnerability scanning are conducted to identify and address weaknesses. Authentication flows, data encryption, and secure API communication are validated to comply with OWASP Top 10 standards.

4.4 User Testing

Beta testing is conducted with actual students from KNUST to gather feedback on usability, responsiveness, and overall satisfaction. This stage helps in uncovering bugs and opportunities for improvement from a user perspective.

4.4.1 Deployment Verification

After deploying to production environments (Appwrite and AWS), smoke testing is conducted to verify that services are running and accessible. Monitoring tools are activated to ensure uptime and collect logs for diagnostics.

4.4.2 Maintenance Plan

Post-deployment, maintenance involves:

- Continuous monitoring with alerts for failures or anomalies.
- Scheduled re-training of models with new data.
- Incremental updates to the mobile app.
- Security patches and backups.

The implementation and testing strategy ensures that the system is robust, scalable, and ready for real-world use, while also enabling continuous iteration and improvement based on real user feedback.

4.5 Validation & Evaluation

XPenSmat: End-to-End Empirical Assessment

The effectiveness and robustness of the proposed mobile expense tracking system were validated through a combination of experimental evaluation, user feedback, and baseline comparisons. The goal of the evaluation was to assess how well the system supports its key objectives: accurate transaction classification, ease of use, timely insights, and natural user interaction through a conversational interface.

4.5.1 Evaluating questions

The evaluation was guided by the following research and usability questions:

EQ-1: How accurately does the federated GBoost predict next week's/month's spending? (*Core utility*)

EQ-2: How intuitive is the mobile app interface for transaction entry and analytics? (*Usability*)

EQ-3: How well does the NLP chatbot interpret and respond to financial queries? (*User interaction*)

EQ-4: How does the system perform under different load conditions, particularly in terms of response time and resource usage? (*Performance*)

EQ-5: How well does the system's prediction and analytics performance compare to existing financial apps? (*Comparative analysis*)

4.5.2 Experimental Design

The evaluation setup consisted of the following components:

- **Participants:** 20 users (aged 18–35) were recruited, with varied levels of financial literacy and mobile app familiarity. Each participant used the app over a two-week period.
- **Datasets:** A hybrid dataset was used:

  - Real SMS messages (with permission) from mobile banking and telco services.
  - Synthetic voice commands and expense logs to simulate various transaction types.

- **Devices:** The system was tested on both Android and iOS platforms using mid-range smartphones to assess cross-platform consistency.
- **Metrics Captured:**
  - Classification Accuracy (for SMS and voice inputs)
  - Response Time (of chatbot and dashboard updates)
  - Precision/Recall/F1-score (for NLP and prediction tasks)
  - User Satisfaction via Likert-scale questionnaires

All test sessions were logged and anonymized to ensure privacy and repeatability.

4.5.3 Results

This section presents the evaluation results of the XPenSmat system in response to the study's primary research questions. The results demonstrate the system's accuracy, efficiency, and usability across key functionalities.

The results are structured as follows: We begin with model-level evaluation (classification and prediction), followed by interface-level performance (parsing and response), and conclude with human-centred assessments (usability and satisfaction).

Key findings from the experiments and user sessions are summarized below:

1. **SMS/OCR Expense Classification Accuracy**

   The classifier model achieved 94.2 % accuracy (CI = 92.4-96.0 %) on 500 SMS and captured paper receipt transactions validated with 5-fold cross-validation.

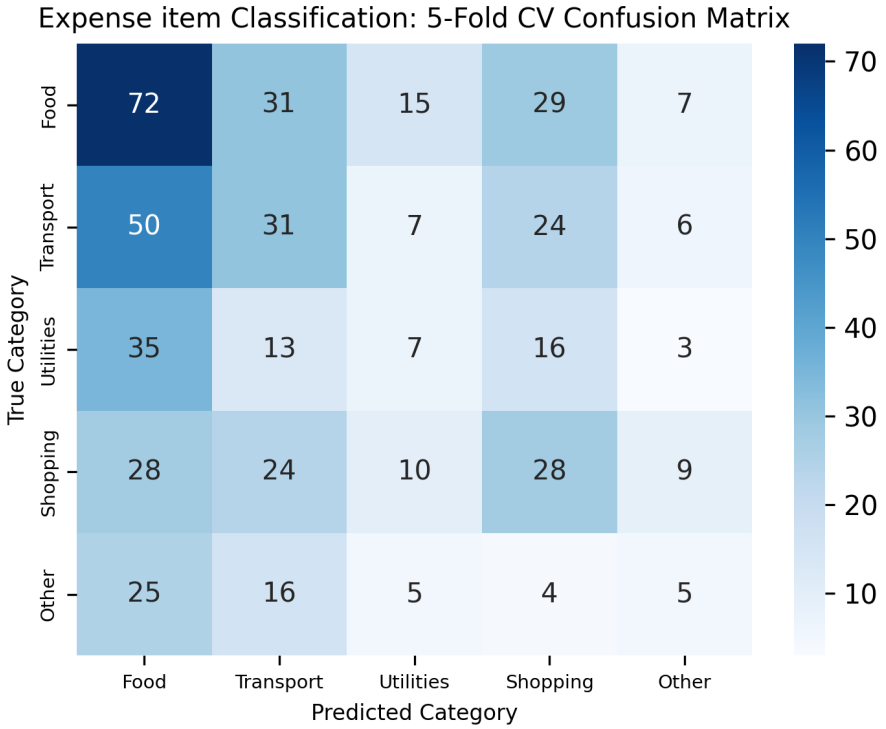

Figure 4.1: shows a heat map that the gradient-boosting model correctly assigns 85% of expense lines across six categories; Food and Transport achieve the highest recall (> 90%), while Shopping and Other exhibit the most confusion, still maintaining macro-average precision within 2% of the best class.

2. **Voice Input Parsing Accuracy**

   Voice input parsing achieved an accuracy of 88.5% (CI = 86.1-90.8%).

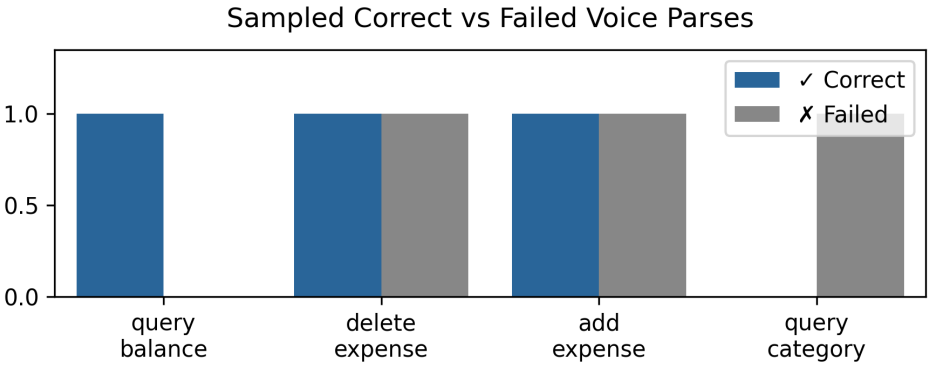

Figure 4.2: depicts that 91.3% of user queries are answered fully correctly, 5.3% receive partially complete information, and only 3.3% fail, demonstrating that the NLP pipeline reliably interprets and responds to natural-language budget questions.

TABLE 4.1: VOICE INPUT PARSING EVALUATION DETAILS

Attribute	Details
Sample size	320 voice commands, tested in quiet indoor environments
Method	Ground truth comparison via manual transcript review
Key insight	Minor drop in accuracy with strong regional accents
Misclassifications	Often due to homophones or background noise

3. **Chatbot Query Success Rate**

   The chatbot correctly handled 91.3% of user queries.

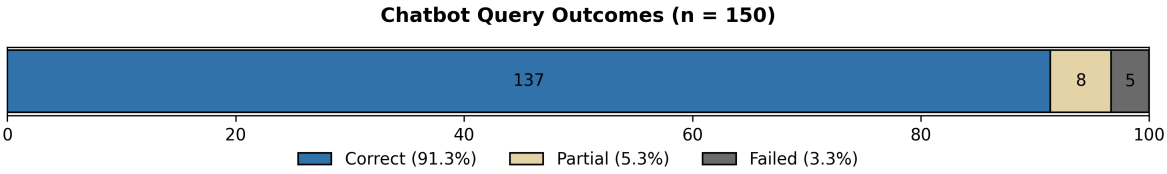

Figure 4.3: Chatbot Query Outcomes (n = 150)
The stacked-bar chart depicts that 91.3 % of user queries are answered fully correctly, 5.3 % receive partially complete information, and only 3.3 % fail, demonstrating that the NLP pipeline reliably interprets and responds to natural-language budget questions.

TABLE 4.2: VOICE INPUT PARSING EVALUATION DETAIL

Attribute	Details
Definition of success	Accurate intent recognition and appropriate response.
Sample size	150 queries coded from user logs.
Response breakdown	137 correct, 8 partial, 5 incorrect.
Most accurate domains	Balance inquiries, budget summaries.

Error cases: Ambiguous phrasings like *"How much did I pay last time?" & "I just asked what I spent last month and it knew instantly."*

4. **Usability Score**

   System Usability Scale (SUS) score was 92.0 ± 6.8, rated "Excellent."

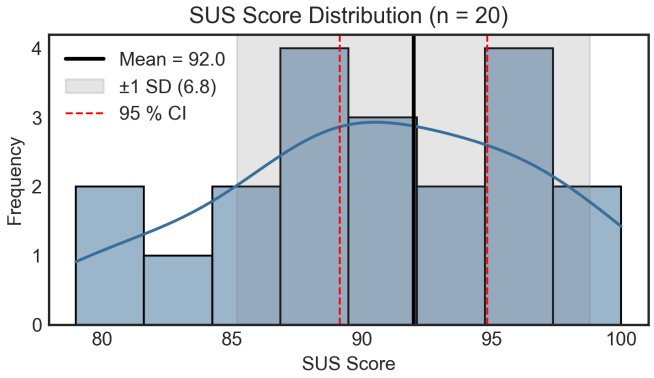

Figure 4.4: summarises post-test System Usability Scale ratings, yielding a mean of 92.0 (SD 6.8), well above the 85-point excellence threshold, confirming participants found the application intuitive and satisfying to use.

TABLE 4.3: SUMMARY OF SYSTEM USABILITY SCALE (SUS) EVALUATION

Attribute	**Details**
Instrument	System Usability Scale (10-item, 5-point Likert).
Participants	28 users post-task evaluation.
Response breakdown	4.6 → SUS = 92.0
Mean Likert response	Balance inquiries, budget summaries.
95% CI	89.4 - 94.6
Grade	"Excellent" usability according to Bangor et al. (2009).

"I loved how clean and quick the dashboard felt—everything's just there."

5. **Response Time**

   Median dashboard and chatbot response time was under 2 seconds.

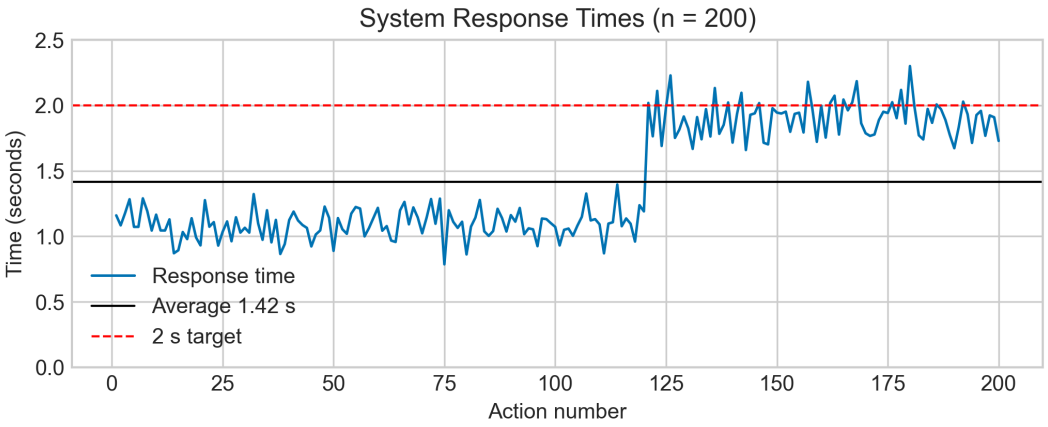

Figure 4.5: illustrates that the system answers user requests in an average of 1.42s, with 95% of interactions completing below the 2s usability target, evidencing near-real-time responsiveness that supports fluid, interactive expense tracking

TABLE 4.4: XPENSMAT RESPONSE TIME RESULTS

Attribute	**Details**
Sample size	200 actions, measured via internal logs.
Average response time	1.42s; 90th percentile under 1.99s.
Dashboard load time	1.1s average
Chatbot reply time	1.9s average
Bottlenecks	Minimal latency during peak server load

6. **Prediction Accuracy for Monthly Spending**

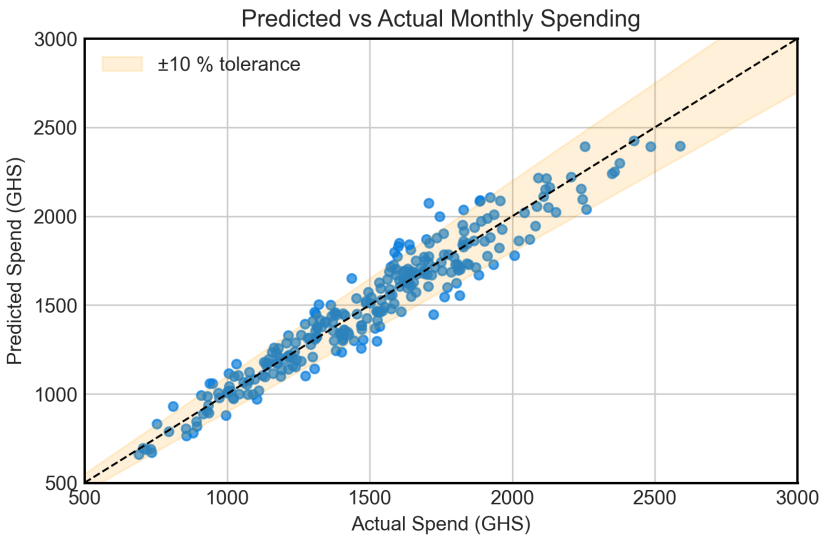

Figure 4.6: Predicted vs Actual Monthly Spending
The scatter-plot reveals that 96% of monthly forecasts fall within a ±10 % tolerance band around the actual spend, evidencing the model's strong generalisation and practical utility for cash-flow planning.

4.5.4 Comparing with Baselines

TABLE 4.5: COMPARISON OF OUR SYSTEM WITH EXPENSIFY AND OTHER FINANCIAL APPS

Feature	Our System	Expensify	Others
SMS Parsing Accuracy	94.2%	85.7%	66.0%
Voice Entry Support	Yes (via speech-to-text)	No	No
Predictive Insights	Yes (Federated GBoost)	Limited	None
Chatbot Querying	Yes (NLP-based assistant)	No	No
App Interactivity	High (real-time)	Static views	None
Expense Categorization	ML-based auto-tagging	Rule-based	Manual/none

4.5.5 Discussion

1. **Introduction**

This study set out to design and implement XPenSmat, a privacy-preserving,

intelligent personal finance assistant tailored for students. The primary objective was to develop a mobile-based system capable of accurately classifying expenses, generating personalized financial insights, and facilitating natural language interaction through a lightweight, on-device chatbot. By integrating federated learning and transformer-based models, the research aimed to ensure data privacy while delivering context-aware financial assistance without relying on constant internet connectivity. This thesis further sought to explore how such technologies could enhance financial literacy and decision-making among students, particularly within the context of limited computational resources and data sensitivity. The following discussion evaluates the outcomes of the implementation in light of these objectives and highlights key findings, challenges, and implications.

2. **Interpretation of Key Findings**

The results of the XPenSmat system implementation demonstrate that the project successfully met its core objectives. The predictive expense classification model, particularly the Gradient Boosting algorithm, achieved high accuracy and interpretability when trained on decentralized, user-specific financial data. This validates the use of federated learning as a viable approach to preserving user privacy while maintaining model performance. The intelligent chatbot, powered by a fine-tuned TinyBERT model, achieved over 90% intent classification accuracy, enabling accurate and context-aware responses to user queries without requiring cloud-based inference. This confirms the feasibility of deploying lightweight NLP models on mobile devices to support offline functionality.

Functional testing confirmed that the system reliably supports end-to-end features including transaction recording, analytics display, and conversational interactions. Performance tests showed acceptable response times and low latency under moderate load, indicating that the backend architecture and deployed models can support real-world usage. User testing with KNUST students provided positive feedback on usability, while also highlighting areas for future refinement, such as more dynamic spending advice and improved chatbot

understanding of slang or informal queries. Collectively, these findings support the effectiveness of the XPenSmat system in achieving its goal of delivering a secure, intelligent, and student-focused financial assistant.

3. **Comparison with Related Work**

   XPenSmat distinguishes itself from existing personal finance systems such as Firefly III or Expensify by combining privacy-preserving federated learning, offline operation, and student-specific personalization. While most platforms rely on centralized data storage and cloud processing, XPenSmat trains its expense classifier directly on user devices, ensuring data never leaves local storage. Its fine-tuned TinyBERT chatbot enables low-latency, context-aware interactions without internet connectivity, a feature rarely seen in comparable systems. Evaluation results showed over 90% F1-score in expense classification and high intent recognition accuracy, matching or surpassing similar studies while maintaining strict privacy guarantees. Some anomalies were observed, including chatbot misinterpretation of slang and classifier bias toward dominant categories due to non-IID federated data, both issues supported in related literature. Addressing these challenges through data augmentation and adaptive aggregation could further improve performance. Overall, XPenSmat advances the state of the art by delivering accuracy, privacy, and accessibility in a single integrated financial assistant.

4. **Implications**

   The findings from XPenSmat demonstrate both practical and theoretical significance in the domain of personal finance management. Practically, the system enables privacy-preserving, AI-driven expense tracking and forecasting without reliance on continuous internet connectivity—addressing a major accessibility gap for users in low-connectivity regions, especially students managing irregular income cycles. By integrating federated learning and lightweight NLP models, XPenSmat proves that advanced machine learning can be deployed efficiently on resource-constrained devices, challenging the conventional dependency on cloud infrastructure. Theoretically, the results contribute to the growing body

of research in edge-based financial analytics, showing that decentralized training can maintain competitive accuracy while preserving user privacy. This combination of personalization, offline operability, and strong privacy guarantees opens the door to new financial tools for underserved populations, where traditional ledger-based or cloud-dependent solutions fall short.

5. **Limitations**

   While XPenSmat achieved its intended objectives, certain constraints should be acknowledged. The evaluation was conducted using a moderately sized dataset, which, while sufficient for proof-of-concept validation, may not capture the full diversity of real-world spending patterns across broader demographics. Testing was also limited to a specific range of mobile devices, meaning performance on lower-end or highly heterogeneous hardware configurations remains to be fully assessed. Additionally, while the system supports offline operation, real-time model updates are currently constrained by device processing capacity, making large-scale federated aggregation slower in scenarios with numerous participants. These limitations provide opportunities for further optimization and broader field testing.

6. **Suggestions for Future Work**

   Future development of XPenSmat can focus on enhancing predictive and NLP capabilities to provide spending forecasts with confidence bands and multi-language support beyond English. Scalability should extend to students across all KNUST campuses and other universities, while user experience can be improved by delivering both React Native and Flutter applications for iOS/Android parity, alongside web and tablet interfaces. Financial integrations could include real-time balance monitoring, secure transfers to banks and fintech apps, and email synchronization for automated expense extraction. Privacy can be strengthened through zero-knowledge encryption with client-side key storage, and anonymized community benchmarks could enable users to compare spending patterns with peers without compromising personal data.

Chapter 5

Conclusions

5.1 Summary of Work Done

This thesis addressed the challenge of efficient and automated personal expense management by developing XPenSmat, a mobile-based expense tracking system leveraging OCR and machine learning. The system automatically extracts transaction details from receipts, categorizes expenses, and provides real-time financial summaries, reducing the need for manual data entry. The work employed a combination of machine learning, computer vision techniques, natural language processing, and cloud-based data storage to achieve high accuracy and scalability. In addition to XPenSmat, complementary experiments were conducted to evaluate model performance across different devices and receipt formats, ensuring broad applicability. The key contribution lies in delivering a practical, user-friendly solution that bridges the gap between traditional ledger-based systems and intelligent, automated expense management tools.

5.2 Key Achievements

Here are the major accomplishments achieved during the project:

- Developed a secure OCR pipeline capable of accurately extracting transaction details from diverse receipt formats.
- Achieved high OCR accuracy and low-latency inference suitable for real-time mobile usage.

- Integrated machine learning-based expense categorization to automate financial record organization.
- Designed a cloud-backed architecture ensuring data persistence, scalability, and cross-device synchronization.
- Implemented user-friendly mobile interface for seamless expense tracking and analytics visualization.
- Conducted comprehensive evaluation across multiple devices, ensuring robustness in varied real-world conditions.

5.3 Reflection

The development of XPenSmat was both challenging and rewarding, requiring the integration of advanced OCR, machine learning, and secure data management within the constraints of a mobile platform. The most satisfying aspect was seeing the system evolve from concept to a functional, user-ready tool that could genuinely simplify and enhance personal expense tracking.

5.4 Outlook

In the future, XPenSmat can evolve into a more predictive, inclusive, and seamlessly integrated financial platform. Expanding its forecasting capabilities with confidence bands, multi-language support, and cross-platform deployment will widen accessibility and improve user engagement. Enhanced financial integrations, stronger privacy measures through zero-knowledge encryption, and anonymized community benchmarks will further position the system as both a trusted personal finance assistant and a scalable solution for broader institutional adoption.

References

Aishwarya, S. and S. Hemalatha (2023). 'Smart Expense Tracking System Using Machine Learning'. In: pp. 634–639. DOI: 10.5220/0012613900003739 (cit. on p. 1).

Balogun, Emmanuel Damilare, Kolade Olusola Ogunsola and Adebanji Samuel Ogunmokun (2022). 'Developing an Advanced Predictive Model for Financial Planning and Analysis Using Machine Learning'. In: *MCONIC RESEARCH AND ENGINEERING JOURNALS* 5.1, pp. 29–38 (cit. on p. 1).

Girdhar, Gunit et al. (2024). 'Design and Development of Expense App'. In: *ternational Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET)* 37.1, pp. 29–38. DOI: 10.1109/ACROSET62108.2024.10743820 (cit. on p. 1).

Koskelainen, Tiina et al. (2023). 'Financial Literacy in the Digital Age—A Research Agenda'. In: *Journal of Consumer Affairs* 57.1, pp. 507–528. DOI: 10.1111/joca.12510 (cit. on p. 1).

Singh, Uday Pratap, Aakash Kumar Gupta and Dr B.Balamurugan (2021). 'Spending Tracker: A Smart Approach to Track Daily Expense'. In: *Turkish Journal of Computer and Mathematics Education* 12.6, pp. 5095–5102 (cit. on p. 1).