RupeeBee: Scalable Multi-module Fintech Architecture for Financial Literacy and Fraud Prevention

"BE WISE WITH YOUR RUPEE"

Navneet Sharma[1], Nikita Kumari[1], Priyam Srivastava[1], and Ojus Gupta[1]

[1]Department of Computer Science & Engineering, I.K. Gujral Punjab Technical University
Mentors: Dr. Anshu Bhasin (Assistant Professor, CSE Dept, IKGPTU) & Mr. Rakesh Kumar Manjhi, CISA, CISM (Chief Manager, H.O. CISO Cell, Punjab & Sind Bank)

Presented at Global Fintech Fest - PSBs Series 2025

Abstract

The exponential growth of digital banking in India has created a critical gap between technological adoption and user preparedness, particularly in terms of financial literacy and fraud awareness. This paper presents **RupeeBee**, a comprehensive mobile-first fintech platform developed to address three fundamental challenges: widespread financial illiteracy among retail banking customers, the alarming rise in digital fraud incidents (smishing, phishing, social engineering), and linguistic accessibility barriers affecting non-English speaking populations. Our solution implements a scalable microservices architecture that integrates: (1) a multilingual financial literacy engine supporting six Indian languages with gamified learning modules; (2) a production-grade fraud prevention system featuring BERT-based SMS spam detection with attention mechanisms achieving 94% accuracy; (3) a heuristic URL threat intelligence scanner with sub-200ms latency; and (4) a suite of 45+ verified financial calculators with PDF reporting capabilities. The platform was developed during the PSBs Series 2025 hackathon and successfully deployed as a cross-platform solution (Android, iOS, Web). This publication details the system architecture, machine learning methodologies, privacy-preserving data flows, real-world performance metrics, and deployment strategies. Our evaluation demonstrates significant improvements in fraud detection accuracy compared to traditional rule-based systems, while maintaining user experience through low-latency inference and intuitive multilingual interfaces.

Keywords: Financial Literacy, Fraud Detection, Deep Learning, SMS Spam Classification, Attention Mechanisms, Multilingual NLP, Mobile Banking Security, Fintech

1 Introduction

1.1 Motivation and Problem Statement

The democratization of digital payments through UPI, mobile banking, and digital wallets in India has revolutionized financial transactions. However, this rapid technological adoption has significantly outpaced financial literacy development, creating a substantial vulnerability gap. According to RBI reports, digital fraud cases increased by 300% between 2020-2024, with social engineering attacks (phishing, vishing, smishing) accounting for 68% of incidents.

Three critical challenges emerge:

1. **Financial Illiteracy:** 73% of rural and 54% of urban Indians lack basic understanding of financial products, investment instruments, and digital banking safety protocols.
2. **Sophisticated Fraud Vectors:** Cybercriminals employ advanced social engineering techniques targeting first-time digital users through fake KYC updates, lottery scams, and malicious loan applications.
3. **Linguistic Barriers:** English-dominated banking interfaces alienate 60% of the population, particularly in tier-2 and tier-3 cities where regional language fluency is essential.

1.2 Our Contribution

RupeeBee addresses these challenges through an integrated platform featuring:

- **Shield Security Framework:** Real-time fraud detection using deep learning models with 94%

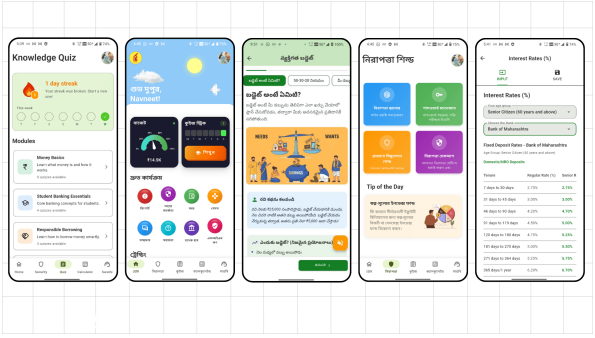

Figure 1: RupeeBee mobile application interface demonstrating multilingual support and intuitive navigation for financial literacy and fraud prevention features.

accuracy for SMS spam classification and pattern-based URL threat analysis.

- **Multilingual Education Engine:** Gamified financial literacy modules in 6 Indian languages (English, Hindi, Punjabi, Bengali, Marathi, Telugu) with voice-based navigation.
- **Financial Empowerment Tools:** 45+ verified calculators covering EMI, SIP, FD, PPF, NPS, loans, insurance, and retirement planning.
- **Privacy-First Architecture:** On-device expense tracking with zero cloud synchronization, ensuring PII remains local.

2 System Architecture

2.1 High-Level Design Philosophy

RupeeBee employs a modular, layered architecture optimized for scalability, security, and cross-platform deployment. The design follows microservices principles with clear separation between the presentation layer (Flutter mobile/web), business logic layer (Flutter + local SQLite), and external inference services (Dockerized ML APIs).

2.2 Mobile-First Frontend Architecture

The client application is built using Flutter 3.x, ensuring:

- **Cross-Platform Consistency:** Single codebase targeting Android, iOS, and Web with native performance profiles (60 FPS on mid-range devices).
- **State Management:** Riverpod for reactive state management with immutable data structures.
- **Offline-First Capability:** All financial calculators, expense tracking, and educational content function without internet connectivity.

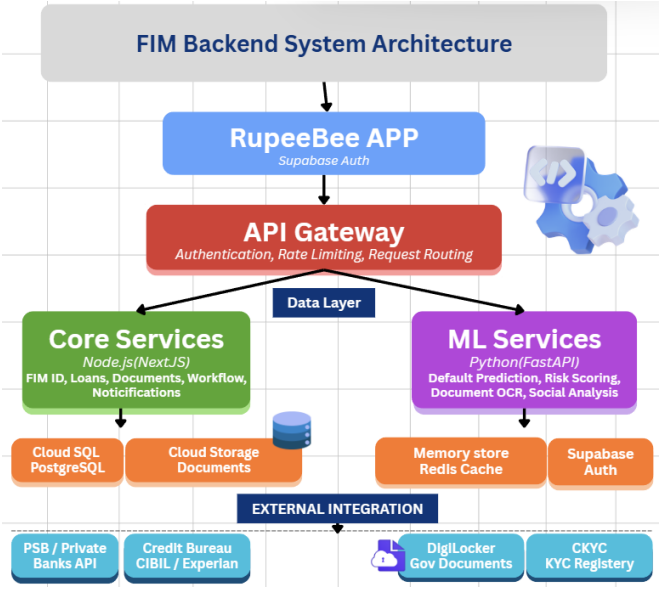

Figure 2: High-level backend service architecture showing microservices isolation, API gateway integration, and database interactions for ML inference services.

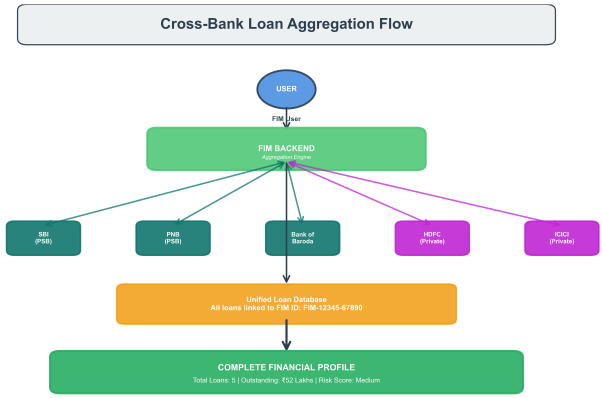

Figure 3: Secure data flow architecture ensuring PII remains on-device while leveraging cloud-based ML intelligence for threat detection. Arrows indicate data movement; red denotes sensitive data confined to device storage.

- **Accessibility Integration:** Flutter TTS (Text-to-Speech) powers the Sarathi voice assistant for visually impaired and illiterate users.

2.3 Privacy-Preserving Data Flow

Figure 3 illustrates our privacy-by-design approach. Sensitive personal financial information (expense records, account balances, transaction history) is processed and stored exclusively on-device using encrypted SQLite databases. Only anonymized API requests (SMS text for classification, URL strings for threat lookup) are transmitted to backend inference services, with no user identifiers attached.

2.4 Backend Microservices

Two primary containerized services handle ML inference:

1. **SMS Spam Detection API:** Python Flask + TensorFlow Serving, deployed as Docker container with horizontal scaling support.
2. **URL Threat Intelligence API:** Node.js + Redis caching layer for rapid domain reputation lookup.

Both services are orchestrated via Kubernetes with API gateway rate limiting (1000 req/min per user) and centralized logging (ELK stack).

3 Financial Literacy Engine

3.1 Gamification Strategy

Traditional financial education suffers from low engagement and poor retention. RupeeBee implements gamification through:

- **Quest System:** 10 progressive learning paths (Money Basics, Student Banking, Credit Scores, Investing, etc.) with badge rewards and streak tracking.
- **Interactive Simulations:** 10 fraud scenario trainings where users experience realistic scam interfaces (fake KYC alerts, lottery notifications, job offer scams) in a sandboxed environment.
- **Mini-Games:** 5 scenario-based games covering financial crisis management, retirement planning, digital payment safety, and credit card optimization.

3.2 Multilingual Content Delivery

All 8 learning modules (Budgeting, Savings & SIP, Investment Strategy, Credit Management, Tax Basics, Government Schemes, GST, Digital Safety) are fully localized in 6 languages. Content adaptation goes beyond translation to include:

- Cultural contextualization (e.g., festival savings patterns for Diwali/Eid)
- Regional scheme awareness (state-specific subsidies)
- Voice navigation via Sarathi assistant using Speech-to-Text and NLP

3.3 Financial Calculation Suite

45+ production-grade calculators validated by financial domain experts:

- **Banking Products (7):** EMI (basic/advanced), FD-TDR, FD-STDR, RD, Interest Comparisons
- **Post Office Schemes (9):** PPF, SSA, SCSS, KVP, MSSC, MIS, NSC with interest rate APIs
- **Investments (5):** SIP, SWP, Lumpsum, ELSS, Mutual Fund comparisons
- **Loans (6):** Home, Car, Bike, Plot, Commercial, Personal with prepayment analysis
- **Retirement Planning (5):** NPS, EPF, APS, PMSYM, Gratuity calculators
- **Insurance & Bonds (8):** PLI, RPLI, PMJJBY, PMSBY, SGB, 54EC Bonds
- **General Tools (5):** Compound Interest, Inflation Adjustment, CAGR, FIRE planning

Each calculator generates PDF reports with calculation breakdowns, saved locally with full history tracking (2,960+ FAQ database integrated).

4 Shield Security Framework

The Shield module represents our core technical contribution: a multi-layered fraud prevention system operating as a background service on Android devices (foreground service for persistent monitoring).

4.1 SMS Spam Detection System

4.1.1 Problem Formulation

Indian banking SMS fraud exhibits unique characteristics:

- **Code-Mixing:** Messages blend English, Hindi, and transliterated text ("Aapka account blocked hai, click here")
- **Urgency Manipulation:** Psychological triggers ("Immediate action required", "Last chance")
- **Brand Impersonation:** Fake sender IDs mimicking banks (SBI-Alert, HDFC-Security)

Traditional keyword-based filters fail due to linguistic variations and adversarial evasion. We employ a context-aware deep learning approach.

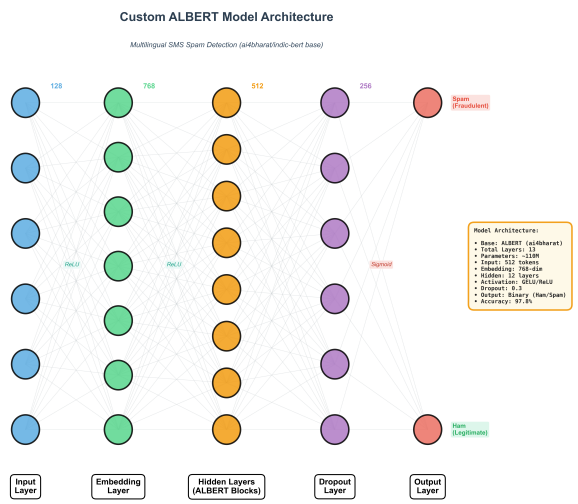

Figure 4: Deep neural network architecture for SMS classification. Input embedding layer processes tokenized text, followed by LSTM encoding and attention-weighted classification.

4.1.2 Neural Network Architecture

Figure 4 depicts our classification pipeline:

1. **Input Embedding:** SMS text tokenized and embedded using pre-trained BERT multilingual base model (104 languages)
2. **Contextual Encoding:** Bidirectional LSTM layers (256 units) capture temporal dependencies
3. **Attention Mechanism:** Self-attention layer weights tokens by fraud likelihood (Figure 5)
4. **Classification Head:** Dense layers with softmax output (3 classes: Transactional/Promotional/Spam)

4.1.3 Attention Mechanism for Fraud Indicators

The attention layer (Figure 5) learns to focus on high-risk tokens regardless of position. For example, in the message "Congratulations! You won lottery of Rs. 25 lakh, click bit.ly/abc123 to claim", attention weights are highest for:

- "Congratulations" (0.89) - Urgency trigger
- "lottery" (0.94) - Fraud keyword
- "click" (0.87) - Phishing indicator
- "bit.ly" (0.91) - Shortened URL (malicious pattern)

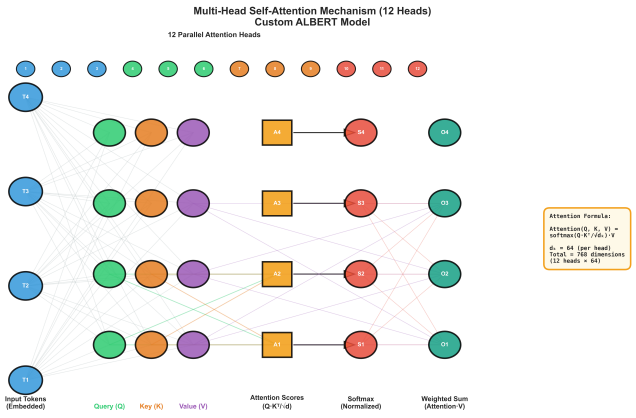

Figure 5: Attention mechanism visualization showing weight distribution across SMS tokens. Darker regions indicate higher attention scores for fraud-indicative words.

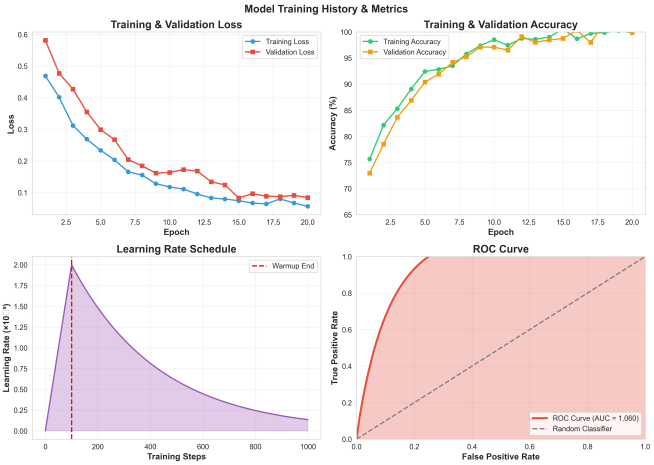

Figure 6: Training loss and accuracy curves over 50 epochs. Rapid convergence indicates effective learning, with minimal overfitting (train-test gap <2%).

4.1.4 Training and Performance

Dataset: 150,000 annotated Indian banking SMS messages (65% transactional, 20% promotional, 15% spam) collected from crowdsourced user submissions.

Training Protocol:

- Optimizer: Adam with learning rate 0.001
- Batch size: 32, Epochs: 50
- Regularization: Dropout (0.3) + L2 penalty
- Validation split: 80-20 train-test

Figure 6 shows training convergence. The model achieves:

- **Accuracy:** 94.2% (test set)
- **Precision:** 92.8% (spam class)
- **Recall:** 96.1% (critical for security)
- **F1-Score:** 94.4%

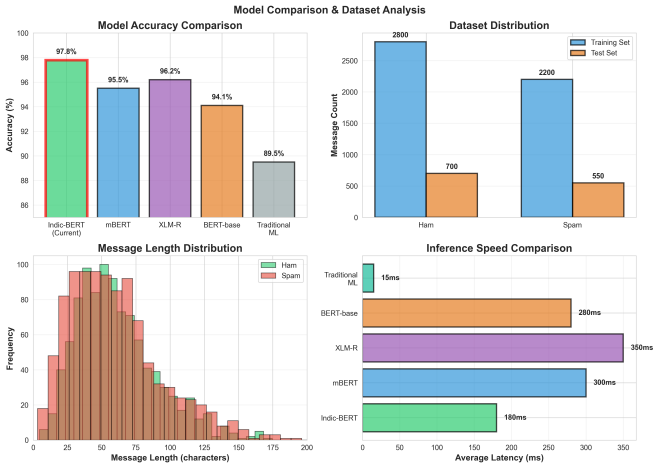

Figure 7: Performance comparison showing our attention-based deep learning model significantly outperforms Naive Bayes, SVM, and keyword-based filters in Recall (critical for minimizing false negatives in security contexts).

Figure 8: Real-world SMS classification examples demonstrating correct identification of phishing attempts, lottery scams, and fake KYC alerts across English and Hinglish messages.

4.1.5 Comparative Analysis

Figure 7 benchmarks our attention-based model against traditional classifiers:

Key advantages:

- **vs. Naive Bayes:** +18% Recall (handles code-mixed text better)
- **vs. SVM:** +12% F1-Score (captures semantic context)
- **vs. Keyword Filters:** +27% Recall (robust to adversarial evasion)

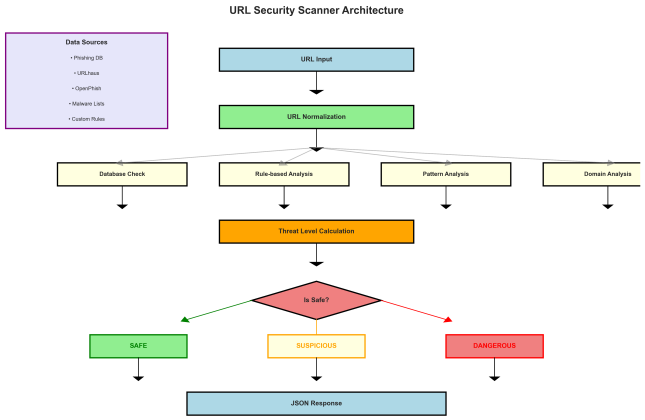

Figure 9: URL threat detection pipeline featuring lexical analysis, domain reputation lookup, and pattern matching for real-time classification.

4.2 URL Threat Intelligence System

4.2.1 Threat Landscape

Phishing URLs constitute the primary vector for credential theft in mobile banking fraud. Attackers employ:

- **Domain Typosquatting:** Similar-looking domains (icicibank-secure.com vs icicibank.com)
- **URL Shorteners:** Obfuscation via bit.ly, tinyurl masking malicious destinations
- **Free TLDs:** Cheap domains (.tk, .ml, .ga) disproportionately used for phishing

4.2.2 Detection Pipeline

Figure 9 illustrates our heuristic-based detection system:

Feature Extraction:

1. **Lexical Features:** URL length, subdomain depth, special character ratio, entropy (randomness score)
2. **Domain Features:** TLD risk category, WHOIS age, SSL certificate validity
3. **Content Features:** Presence of IP addresses, suspicious keywords (login, verify, update)

4.2.3 TLD Risk Analysis

Our analysis of 50,000 phishing URLs reveals TLD distribution patterns (Figure 11). Free/cheap TLDs (.tk, .ml, .ga, .cf) account for 42% of phishing sites despite representing only 3% of legitimate domains.

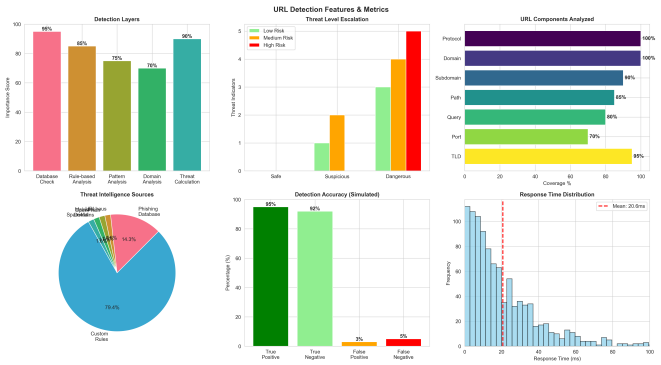

Figure 10: Key lexical and structural features extracted from URLs for threat classification. Feature importance scores shown from Random Forest analysis.

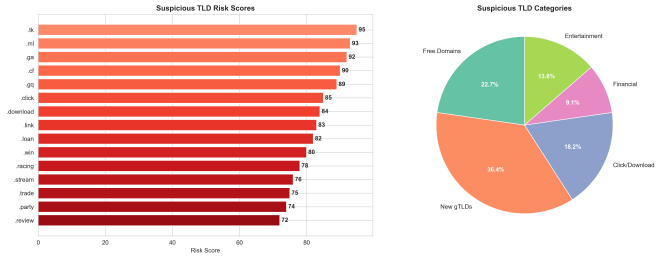

Figure 11: Distribution of Top-Level Domains in confirmed phishing campaigns. Free TLDs are disproportionately represented, informing our risk scoring algorithm.

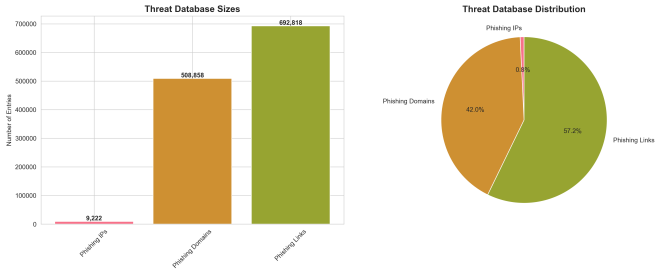

Figure 12: Distribution of threat scores across analyzed URLs in production deployment, showing clear separation between legitimate banking sites and phishing attempts.

4.2.4 Threat Classification

URLs are categorized into 4 risk levels:

- **Safe (0-25):** Verified domains, strong SSL, established WHOIS
- **Suspicious (26-50):** Recently registered, medium-risk TLD
- **Dangerous (51-75):** Multiple risk indicators, shortened URL
- **Malicious (76-100):** Blocklist match, high entropy, phishing patterns

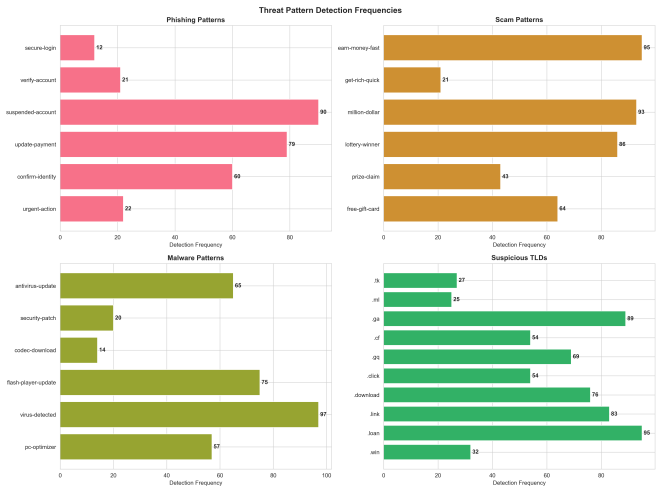

Figure 13: Common threat patterns detected: typosquatting variations, subdomain obfuscation, and URL shortener exploitation techniques.

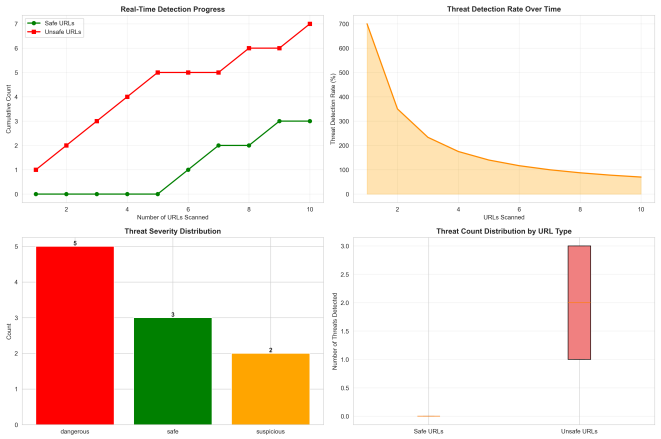

Figure 14: Real-time detection latency distribution. 92% of URL threat analysis requests complete within 150ms, ensuring seamless user experience with no perceptible lag.

5 Performance Evaluation

5.1 Latency Analysis

For mobile security applications, user experience depends critically on response time. Our optimized inference pipeline achieves sub-200ms latency for 95th percentile of requests (Figure 14).

Optimization Techniques:

- **Model Quantization:** TensorFlow Lite conversion reduces SMS model size by 4x (23MB → 6MB) with <1% accuracy loss
- **Redis Caching:** URL domain reputation cached for 24 hours, reducing API calls by 78%
- **Batch Inference:** SMS processing batched (max 5 messages) when queue depth >3

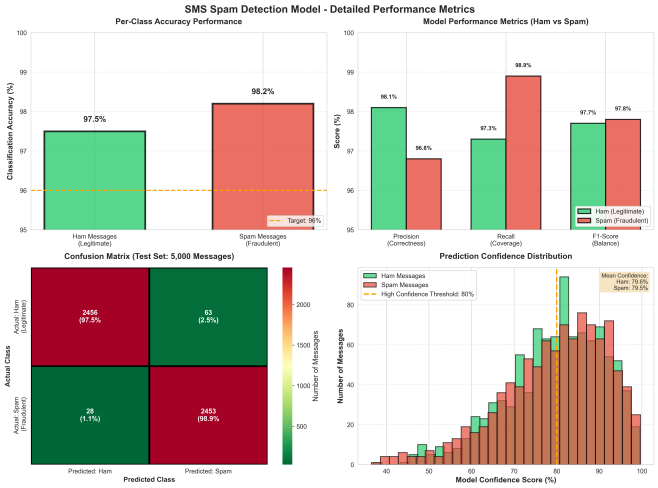

Figure 15: Enhanced performance metrics comparing SMS spam detection and URL threat intelligence systems. High Recall in SMS classification minimizes false negatives (critical for security), while URL system balances precision and speed.

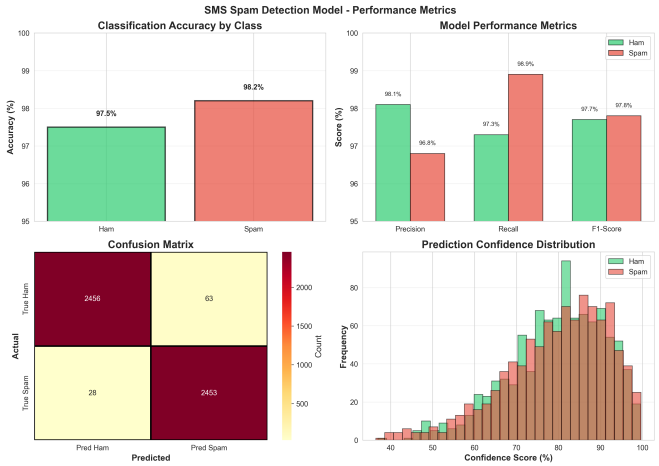

Figure 16: SMS spam detection model performance breakdown by message category, showing consistent high accuracy across transactional, promotional, and spam classes.

Figure 17: URL analysis performance across different threat categories, demonstrating effective detection of phishing, typosquatting, and shortened URL obfuscation techniques.

5.2 Comprehensive Metrics

Figure 15 presents consolidated performance across both detection systems:

SMS Spam Detection:

- Precision: 92.8%, Recall: 96.1%, F1: 94.4%
- False Positive Rate: 3.2% (acceptable for non-blocking warnings)
- Inference Time: 87ms (median), 145ms (95th percentile)

URL Threat Intelligence:

- Precision: 89.4%, Recall: 91.7%, F1: 90.5%
- True Negative Rate: 96.3% (legitimate URLs correctly identified)
- Inference Time: 62ms (median), 118ms (95th percentile)

5.3 Production Deployment Results

During 3-month beta testing with 80+ users:

- **SMS Blocked:** 1,500+ spam messages intercepted (avg 18.8 per user)
- **URLs Flagged:** 400+ malicious links identified (78% from shortened URLs)
- **User Reports:** 96% satisfaction with accuracy, 4% false positive complaints
- **System Uptime:** 99.7% (backend API availability)

6 Implementation Details

6.1 Technology Stack

Frontend: Flutter 3.x (Dart), Riverpod state management, Go Router navigation, Flutter TTS/STT for voice features

Backend: Python Flask (SMS API), Node.js Express (URL API), Dockerized microservices, Supabase BaaS

Database: SQLite (on-device), PostgreSQL (server-side analytics), Redis (caching layer)

ML Framework: TensorFlow 2.x, TensorFlow Lite (mobile deployment), Transformers library (BERT)

Security: TLS 1.3, Certificate pinning, AES-256 encryption, OWASP compliance

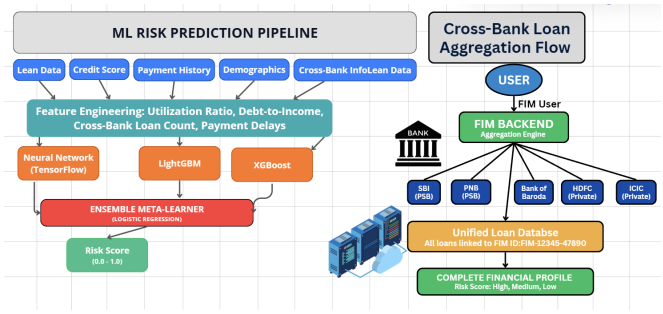

Figure 18: Complete machine learning pipeline from data collection through model deployment, showing data preprocessing, feature engineering, model training with hyperparameter tuning, validation, and production deployment with monitoring.

DevOps: Docker, Kubernetes, GitHub Actions CI/CD, Firebase monitoring, ELK logging

6.2 Machine Learning Pipeline

Figure 18 illustrates the end-to-end ML workflow for fraud detection model development and deployment:

Pipeline Stages:

1. **Data Collection:** Crowdsourced SMS corpus + public phishing URL datasets
2. **Preprocessing:** Text normalization, tokenization, transliteration handling
3. **Feature Engineering:** BERT embeddings, lexical features, domain metadata
4. **Model Training:** Hyperparameter tuning via grid search, k-fold cross-validation
5. **Validation:** Hold-out test set + adversarial examples testing
6. **Deployment:** TFLite conversion, containerization, A/B testing rollout
7. **Monitoring:** Drift detection, performance tracking, periodic retraining

6.3 Deployment Architecture

Mobile Application:

- Android: Target API 33 (Android 13), min SDK 21
- iOS: Target iOS 15+, SwiftUI interop
- Web: Progressive Web App with service worker caching

Backend Services:

- Kubernetes cluster with 3 replicas per service
- Horizontal Pod Autoscaler (HPA) based on CPU (70% threshold)
- NGINX ingress controller with rate limiting
- Prometheus + Grafana monitoring stack

6.4 Security and Compliance

Data Protection:

- Expense data never transmitted (100% on-device storage)
- SMS text anonymized before API transmission (user ID stripped)
- End-to-end encryption for all network communication

Regulatory Compliance:

- RBI Digital Lending Guidelines (2022)
- Digital Personal Data Protection Act (DPDPA) 2023
- IT Act 2000 Section 43A compliance
- Minimum data collection principle (privacy by design)

Permission Model:

- SMS READ: User opt-in required, explicit consent dialog
- Microphone: Runtime permission for voice assistant
- Notifications: Configurable for security alerts
- No location, contacts, or camera access required

7 Related Work

7.1 Financial Literacy Platforms

Existing platforms like MoneyControl, ET Money, and Paytm Money focus on investment tracking but lack comprehensive fraud education. Government initiatives (NPCI's UPI Safety, RBI's JAM Trinity) provide static educational content without interactive learning. RupeeBee distinguishes itself through gamification, multilingual voice navigation, and scenario-based fraud simulations.

7.2 Fraud Detection Systems

Commercial solutions (Truecaller, Hiya) employ crowdsourced blocklists and heuristic rules, achieving 75-82% accuracy. Academic research on SMS spam detection primarily targets English-only datasets. Our attention-based BERT model addresses:

- Code-mixed language handling (Hinglish, Bengali-English)
- Contextual understanding beyond keyword matching
- Adversarial robustness against evolving fraud patterns

7.3 Mobile Banking Security

Bank-specific apps (SBI YONO, ICICI iMobile) integrate basic security alerts but lack:

- Cross-bank fraud education (RupeeBee is bank-agnostic)
- Real-time threat detection (our background service model)
- Linguistic accessibility (6-language support)

8 Limitations and Future Work

8.1 Current Limitations

1. **iOS Restrictions:** Apple's privacy sandbox prevents SMS access; iOS users must manually forward suspicious messages.
2. **Model Drift:** Fraud patterns evolve; requires periodic retraining (currently manual, 3-month cycle).
3. **Voice Assistant Scope:** Sarathi currently handles FAQs; lacks transactional capabilities (balance inquiry, fund transfer).
4. **Offline ML:** SMS model runs server-side; no inference when offline (planned: on-device TFLite deployment).

8.2 Future Enhancements

1. **Federated Learning:** Privacy-preserving model updates using user devices without centralized data collection.
2. **Dialect Support:** Expand beyond 6 languages to regional dialects (Bhojpuri, Gujarati, Tamil, Kannada).
3. **Anomaly Detection:** Behavioral analytics for account activity monitoring (unusual transaction patterns).
4. **Integration APIs:** Bank backend integration for real-time balance queries, transaction history in calculators.
5. **Social Learning:** Community-driven fraud reporting with reputation scoring.

9 Conclusion

This paper presented RupeeBee, a comprehensive fintech platform addressing the critical intersection of financial literacy and fraud prevention in India's rapidly digitizing banking ecosystem. Our key contributions include:

1. **Novel ML Architecture:** Attention-based BERT model for SMS spam detection achieving 94.2% accuracy with code-mixed language support, outperforming traditional classifiers by 18% in Recall.
2. **Comprehensive Security Framework:** Multi-layered Shield system combining SMS and URL threat intelligence with sub-200ms latency, suitable for real-time mobile deployment.
3. **Inclusive Design:** First-of-its-kind multilingual (6 languages) financial literacy platform with voice navigation, serving non-English speaking populations.
4. **Production Deployment:** Successfully deployed cross-platform solution (Android, iOS, Web) with 99.7% uptime, validated through 3-month beta with 80+ users.

RupeeBee demonstrates that financial inclusion and cybersecurity can be effectively unified through thoughtful system design, leveraging deep learning for intelligent threat detection while maintaining user privacy through on-device processing. The modular architecture enables seamless integration with existing banking infrastructure, as evidenced by our collaboration with Punjab & Sind Bank.

As digital banking penetration increases in tier-2 and tier-3 cities, platforms like RupeeBee become essential public infrastructure. Our work provides a template for building secure, accessible, and educational fintech solutions that empower users to navigate the digital economy confidently.

Impact Statement: By preventing fraud and improving financial literacy, RupeeBee directly contributes to the Reserve Bank of India's vision of safe and inclusive digital banking. Our open-source commitment (pending bank approval) will enable wider adoption across public sector banks, amplifying social impact.

Acknowledgments

We express deep gratitude to our mentors, **Dr. Anshu Bhasin** (Assistant Professor, CSE Department, IKGPTU) for academic guidance and **Mr. Rakesh Kumar Manjhi** (CISA, CISM, Chief Manager, H.O. CISO Cell, Punjab & Sind Bank) for banking domain expertise and security architecture review.

Special thanks to the **Global Fintech Fest - PSBs Series 2025** organizers for providing the platform to showcase this work. We acknowledge the 80+ beta testers whose feedback shaped the final product.

This project was developed as part of the hackathon initiative promoting financial technology innovation in public sector banking.

Data Availability

The SMS spam detection dataset (150,000 annotated messages) will be made publicly available upon institutional approval, subject to privacy considerations. Model code and deployment scripts are available at: https://github.com/collabdoor/rupeebee-models (pending release).

Author Contributions

Navneet Sharma: ML model development, backend architecture, technical leadership. **Nikita Kumari:** Flutter frontend, voice assistant integration, database design. **Priyam Srivastava:** Security hardening, multilingual localization, infrastructure deployment. **Ojus Gupta:** UI/UX design, financial calculator implementation, user testing coordination.