Offline Transactions

Presentated by :

Sharvi Burse SY COMP A

Bhargavi Dange SY COMP B

Introduction

India’s UPI revolution :

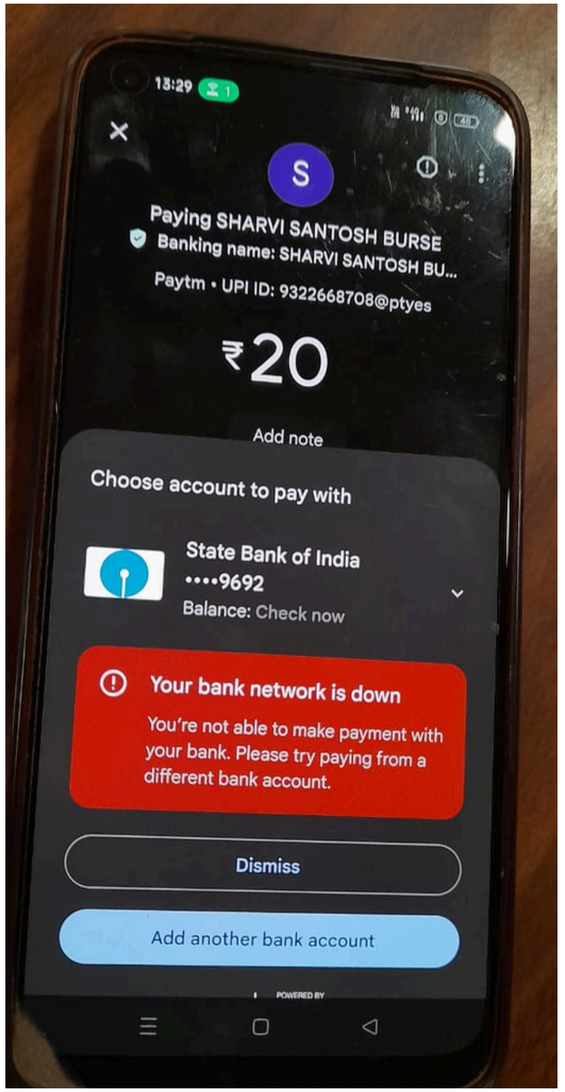
- 400+ million UPI users as of 2024
- Over 10 billion transactions in a single month
- UPI is powering India’s digital economy

But .....

- 60,000+ villages still lack stable internet
- Millions experience frequent network issues
- UPI growth is urban-centric, leaving rural India behind

Why it matters?

- Includes rural India in the digital economy
- Supports small businesses, farmers, and workers
- Boosts resilience during network outages
- A step forward in Digital India’s inclusive growth



Problem Statement

Digital payment systems like UPI rely heavily on internet connectivity, making them inaccessible in low or no-network areas. This limits the reach of cashless transactions and excludes individuals in such regions from participating in the digital economy. There is a need for a solution that enables secure and seamless UPI transactions even without internet access.

Solution

Challenges Faced

Problem 1

Normal Transactions and Database Restore:
If transactions are only stored and restored from a local database (when the sender is offline), there can be problems. For example, if the sender is never online to confirm the transaction, it might lead to inconsistencies in the data (like missed transactions or unprocessed ones).

Problem 2

Normal Random Verification Code (Security Issue):
Using a simple random verification code for confirming transactions can be risky because these codes can be easily guessed or intercepted by malicious parties. This could lead to unauthorized transactions or security breaches. To avoid this, encryption is used to protect the verification process and ensure only authorized parties can access and process the transaction.

Problem 3

Encrypted Code and User Interaction:
Encryption makes data secure but can be difficult for users to handle, as they might not understand how to deal with encrypted information directly. To solve this, a QR code is used. It's a simple way to display encrypted information (like a transaction code or details), which users can easily scan using their mobile devices for a smooth and secure interaction.

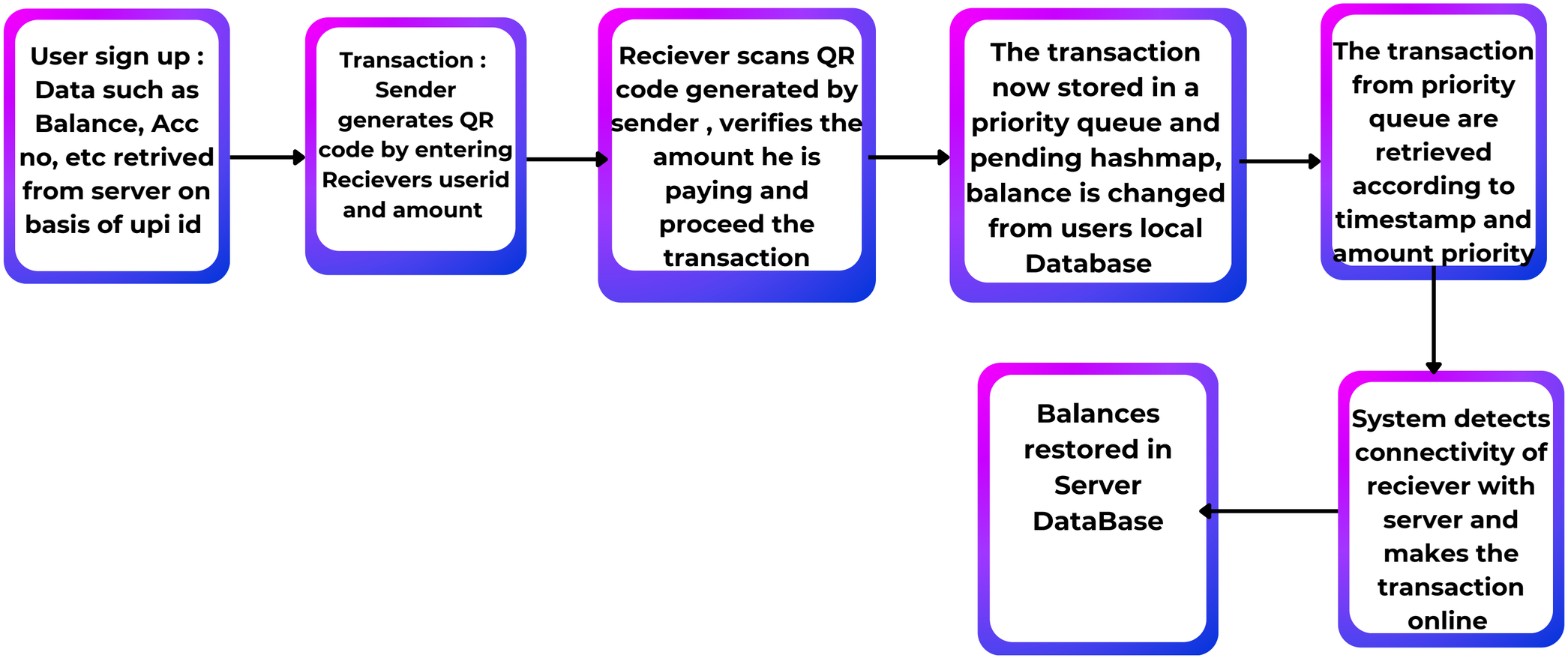
Problem 4

Retrieval from Receiver Side:
On the receiver's side, the transaction details are retrieved in an encrypted form. This ensures that even if the data is intercepted, it cannot be understood without the proper decryption key, ensuring privacy and security for both sender and receiver.

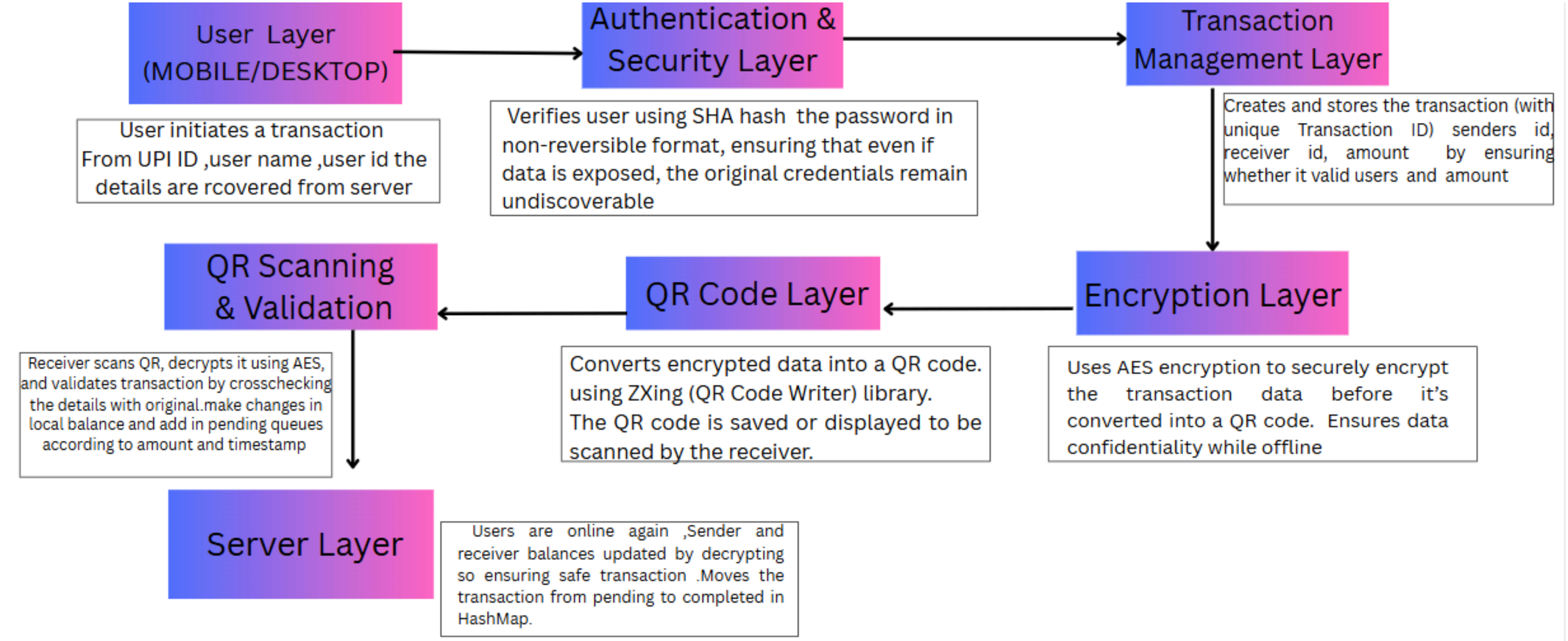
Problem 5

Password Hashing: Only hashed passwords are stored, ensuring the original password is never exposed

Process Flow :



Architecture Diagram



User → Authentication → Transaction Creation → AES Encryption → QR Generation → QR Scan → AES Decryption → Validation → Local Update →Server Update

Data Structures

Data Structure	Purpose	Why We Used It	Why Not Other Structures
HashMap	To store and retrieve user and transaction details like user_id, upi_id, pin,balance quickly by mapping them to unique keys. Ideal for managing large datasets with fast lookups.	Provides constant-time (O(1)) access and key-based mapping.	ArrayList/LinkedList: O(n) search. TreeMap: O(log n) with sorting overhead. Set: No key-value mapping.
Stack	To maintain transaction history per user, where the most recent transaction should be accessed or reversed first (Last-In-First-Out behavior).	Allows fast access to the most recent transaction and supports undo operations.	Queue: Uses FIFO, not suitable for reversal. ArrayList: Allows misuse of order. LinkedList: Higher memory usage.
PriorityQueue (Max Heap)	To handle offline transactions by processing the most urgent or highest priority ones first, ensuring critical operations are not delayed.	Automatically processes transactions based on priority using max-heap logic.	Queue/Stack: No support for priority. ArrayList: Needs frequent sorting. TreeSet: Doesn't support duplicates.
ArrayList	To store flexible and dynamic lists like registered users, recent UPI transactions, or contributors, with support for index-based access and iteration.	Offers dynamic resizing, fast element access, and easy traversal.	Array: Fixed size. LinkedList: Slower access time.Set: No indexing .

A **heap-based PriorityQueue** ensures **O(log n) for insertion** and **removal and O(1)** for accessing the highest/lowest priority element, making it efficient and balanced. In contrast, a **normal PriorityQueue** using **unsorted or sorted lists can require O(n) or O(log n) f**or operations, making it less efficient. Hence, heaps are preferred for better performance and scalability

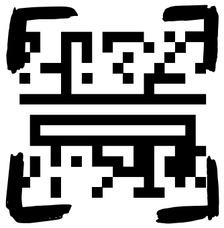
Algorithms

1. **AES (Advanced Encryption Standard)**

- Purpose: Encrypts transaction data to keeping it secure.
- Why: Protects sensitive information by ensuring only authorized people can read it.
- A strong encryption method that uses the same key for both encrypting and decrypting data.

2. **SHA-256 (Secure Hash Algorithm)**

- Purpose: Turns passwords into a unique string of numbers and letters.
- Why: Makes it hard to guess or reverse the original data, ensuring it stays safe.
- A function that creates a unique code (hash) for any input data.



3. **QR Code Generation (QRCodeWriter Algorithm)**

- Purpose: Turns transaction details into a QR code.
- Why: Lets users scan the code for offline transactions without needing the internet.
- A way to create barcodes that store information, which can be scanned by phones

USP :

- Our project offers a truly offline UPI payment solution using a encrypted code and QR, requiring no internet, SIM card, or special hardware.
- Existing solutions include:
- *NPCI’s 99# USSD service – works only on feature phones and requires a mobile network.
- ToneTag’s sound-based payments – relies on additional hardware or sound emitters.
- RBI’s NFC-based offline payment pilot – hardware-dependent and limited to specific banks.
- UPI Lite – allows low-value payments without PIN, but still requires periodic internet to load wallet and does not support peer-to-peer transactions offline.
- These systems are either network-dependent, bank-specific, or support only partial offline functionality.
- Our solution is device-agnostic, app-based, and supports offline peer-to-peer transactions using encrypted code verification — ideal for rural users, emergency situations, and low-connectivity zones.
- To date, no completely internet-independent UPI solution has been deployed at scale for smartphones in a generalized, app-based format — we aim to bridge that gap.

Future Scope and Security Enhancements

Unauthorized Access

➤ App-level biometric lock & user-device binding

Data Modification During Sync

➤ Use Merkle Trees & hash verification on reconnection

Man-in-the-Middle Attacks

➤ Encrypt code transmission & allow only local device input

Device Spoofing/Cloning

➤ Implement device fingerprinting & secure device ID checks

Thank you