📘 TRUSTLESS PEER-TO-PEER LENDING NETWORK (TPLN)

A Cryptographically Secure Informal Lending System

Flagship Engineering Project – Flutter + FinTech + Security

---

1. Project Vision & Real-World Impact

1.1 The Problem

In India and many developing countries, people lend money informally:

- Friends
- Relatives
- Shopkeepers
- Tenants
- Freelancers

These loans:

- Have **no legal proof**
- Are tracked mentally or on paper
- Lead to disputes, money loss, and stress
- Are completely outside the banking system

Despite UPI growth, **trust is still manual**.

1.2 The Solution

TPLN converts informal loans into:

- **Digitally signed**
- **Tamper-proof**
- **Cryptographically verifiable**
  loan contracts.

No courts.
No banks.
No middlemen.

Just **math + software = trust**.

1.3 Impact

- Protects lenders

- Reduces disputes
- Enables financial inclusion
- Introduces real cryptography into everyday life

This is **financial infrastructure**, not just an app.

---

2. System Overview (Big Picture)

2.1 High-Level Architecture

Flutter App (UI)

↓

Crypto Layer (RSA, AES, Hashing)

↓

Secure Storage (Local + Cloud)

↓

Backend (Firebase)

2.2 Trust Model

- Users don’t trust each other
- They trust **cryptography**
- Contracts cannot be altered
- Signatures cannot be denied

2.3 Threat Model

System protects against:

- Contract modification
- Payment denial
- Fake agreements
- Data tampering
- Replay attacks

---

3. Technology Stack

Layer	**Technology**	**Purpose**
IDE	VS Code	Development
Frontend	Flutter (Dart)	Cross-platform UI
Backend	Firebase	Auth + Database
Crypto	RSA, AES, SHA-256	Security
Storage	Secure Local + Firestore	Offline + Sync
Auth	Firebase Auth	Identity
Version Control	Git + GitHub	Collaboration

---

4. Installation & Setup Guide

4.1 Required Software

- VS Code
- Flutter SDK
- Android Studio (for emulator only)
- Git

4.2 VS Code Extensions

- Flutter
- Dart
- GitLens
- Error Lens (optional)

4.3 Verify Setup

Run:

```
flutter doctor
```

All checks should be green.

---

5. Core App Features

5.1 MVP Features

- User registration & login
- Loan creation
- Digital contract generation
- Digital signatures
- Encrypted storage
- Repayment tracking

5.2 Advanced Features

- Late payment penalties
- Partial repayments
- Contract history
- Risk score per borrower
- Audit logs

---

6. Cryptography (Explained Simply)

6.1 Hashing (SHA-256)

- Converts contract → fixed fingerprint
- Even 1 character change = different hash

Used for:

- Integrity verification

6.2 Encryption (AES)

- Protects contract data
- Fast & secure

Used for:

- Storing sensitive information

6.3 Digital Signatures (RSA)

- Each user has a key pair
- Private key signs contract
- Public key verifies signature

Prevents:

- Denial
- Forgery
- Modification

6.4 Crypto Flow

1. Contract created
2. Hash generated
3. Hash signed by lender
4. Hash signed by borrower
5. Stored securely

---

7. Database Design (Firestore)

7.1 Collections

users/

contracts/

repayments/

keys/

audit_logs/

7.2 Contract Document Example

contractId

lenderId

borrowerId

amount

interest

dueDate

contractHash

lenderSignature

borrowerSignature

status

timestamps

---

8. Flutter App Architecture

8.1 Folder Structure

```
lib/
 ├── auth/
 ├── contracts/
 ├── crypto/
 ├── services/
 ├── models/
 ├── screens/
 └── utils/
```

8.2 Architecture Principles

- UI ≠ Logic
- Crypto isolated
- Services reusable
- Clean code

---

9. Backend Logic

9.1 Authentication

- Firebase Auth
- Email / Phone login
- User identity verification

9.2 Validation Rules

- Only involved parties can access contracts

- No write after signing
- Repayment rules enforced

---

10. Security Design

10.1 Key Management

- Private keys stored locally
- Encrypted with device security
- Never sent to server

10.2 Attack Prevention

- Hash verification on every read
- Signature validation
- Timestamp checks

10.3 Lost Phone Scenario

- Re-authentication
- Contract recovery via public keys
- Old device revoked

---

11. Development Roadmap (8 Weeks)

Week	**Goal**
1	Setup + Auth
2	Contract UI
3	Cryptography
4	Secure Storage
5	Repayments

Week	Goal
6	Security hardening
7	Testing
8	Polish & deploy

---

12. Testing Strategy

- Unit tests for crypto
- Contract integrity tests
- Edge cases
- Fraud simulation

---

13. Deployment & Demo

- Build APK
- Demo video (5–7 min)
- Screenshots
- Backend rules review