Technical Design Document (TDD)

1. Document Control

- **Version:** 1.0
- **Author:** Richard Baah, Jose Lamela, Saksham Mehta
- **Date:** 2025-09-04
- **Reviewers:** Development Team
- **GitHub:** https://github.com/Rich-T-kid/micro-lending

2. Introduction

This document outlines the **technical design and implementation strategy** for the **Microlending Platform** described in the Functional Requirements Document (FRD).

The system enables borrowers to request small loans and lenders to fund them, while enforcing credit evaluation, repayment tracking, and compliance reporting.

Linked Document: Microlending FRD

3. System Architecture

3.1 Overview

The system follows a **three-tier architecture**:

Layer	Description	Components
Presentation Layer	User-facing web interface for borrowers, lenders, and admins.	HTML/CSS/JS frontend
Application Layer	Business logic for loan processing, scoring, payments, and compliance.	FastAPI (Python)
Data Layer	Persistent relational storage for structured financial and user data.	PostgreSQL database hosted on Railway

All layers communicate via RESTful APIs secured with JWT authentication.

3.2 Architectural Diagram

📄 *See:* Architectural Diagram

4. Data Model

The data model is designed for **referential integrity**, **normalized structure**, and **auditability**. Each entity follows a clear ownership and foreign key hierarchy to maintain consistency across borrower, lender, and transaction records.

4.1 Core Tables

Entity	Description
`UserAccount`	Stores user identity and credentials.
`UserRole` / `UserAccountRole`	Defines role-based access.
`Institution`	Represents financial or lending institutions.
`UserProfile`	Holds borrower demographics and credit information.
`KYCVerification`	Captures verification results for compliance.
`LoanApplication`	Tracks borrower loan requests and risk decisions.
`LoanProduct`	Defines loan offerings by institutions.
`Loan`	Represents active funded loans.
`Payment` / `RepaymentSchedule`	Tracks repayments and outstanding balances.
`P2PLoanRequest` / `P2POffer`	Manages peer-to-peer loan postings.
`Notification`	Logs outbound user communications.
`AuditLog`	Records every user or system action for compliance.

4.2 Normalization and Integrity

- **Normalization Level:** 3rd Normal Form (3NF)
  - Prevents duplication across entities (e.g., users vs. institutions).
- **Foreign Keys:** Enforced between user, loan, and payment tables.
- **Referential Integrity:** Cascading deletes for dependent records.
- **Indexing:**
  - `CREATE INDEX idx_user_email ON UserAccount(email);`
  - `CREATE INDEX idx_loan_status ON Loan(status);`
  - `CREATE INDEX idx_payment_loan_id ON Payment(loan_id);`
- **Data Types:**
  - `UUID` for primary keys
  - `JSONB` for flexible metadata fields (e.g., fees, audit data)
  - `TIMESTAMP` for lifecycle events

---

5. Technology Stack

Component	Technology

Component	Technology
Frontend	HTML, CSS, JavaScript (responsive design)
Backend Framework	Python with FastAPI
Database	PostgreSQL (hosted on Railway)
Hosting	Railway (auto-deploy via GitHub Actions)
Authentication	JWT (Bearer tokens)
External Services	Stripe/PayPal (payments), Persona/Onfido (KYC), SendGrid (notifications)

6. API Design

Attribute	Description
Style	REST (JSON-based)
Base URL	`/api/v1/`
Authentication	`Authorization: Bearer <JWT>`
Content-Type	`application/json`

Example Endpoints

Method	Endpoint	Description
`POST /auth/login`	User authentication	
`POST /loans/apply`	Submit borrower loan application	
`GET /loans/{id}`	Retrieve loan details	
`POST /payments/{loan_id}`	Make loan repayment	
`GET /admin/users`	Admin view of all users	

7. Security Approach

Area	Technique
Authentication	JWT-based token issuance on login
Authorization	Role-based access (Borrower, Lender, Admin, Compliance)
Data Protection	Password hashing (bcrypt/SHA256), TLS encryption
SQL Injection Prevention	Parameterized queries & ORM input validation
Audit Logging	Track all CRUD and authentication actions
Compliance	Adheres to KYC/AML requirements with secure data storage

8. Entitlements and Role Mapping

Role	Accessible Functions
Borrower	Apply for loans, view repayment status, send messages
Lender	Fund loans, view investments, withdraw balance
Administrator	Manage users, approve loans, generate system reports
Compliance Officer	View audit trails, export reports, view flagged accounts

All entitlements are enforced via middleware intercepting JWT claims before controller execution.

---

9. Interface Design

Interface	Description
Web Interface (Frontend)	Lightweight HTML/JS dashboard for end users
Admin Console	Role-based admin dashboard with data tables and filters
API Interface	JSON-based API for integrations and mobile extensions
CLI (optional)	Developer or DevOps tools for migrations, logs, or testing

Each view layer interacts with the FastAPI backend using secured REST endpoints.

---

10. Scalability Considerations

Area	Strategy
Database Scaling	Vertical scaling via Railway Postgres tiers, read replicas for analytics
Application Scaling	Horizontal scaling via Railway auto-scaling instances
Caching	Redis layer for frequent lookups (e.g., user sessions, exchange rates)
Load Balancing	Railway-managed load balancing for multiple backend pods
Data Sharding (Future)	Partition loans/payments by region or institution for distributed writes

---

11. Performance Optimization

- Index frequent query columns (`email`, `status`, `loan_id`).
- Use database connection pooling to reduce latency.
- Cache static configurations (currencies, rates).
- Optimize SELECT queries with joins instead of subqueries.
- Asynchronous I/O via FastAPI for concurrent requests.

---

12. Risks and Mitigations

Risk	Mitigation
Fraud or abuse	KYC verification, admin approval of lenders
Data breach	Encryption at rest + TLS, RBAC, least privilege DB access
Duplicate payments	Transactional idempotency keys
Performance degradation	Index tuning, Redis caching, async processing
Service downtime	Health checks and Railway auto-restarts

---

13. Testing Strategy

Type	Description
Unit Tests	Validate business logic (loan calculations, validation rules)
Integration Tests	API and database interactions
E2E Tests	Full loan lifecycle flow (apply → approve → fund → repay)
Security Tests	SQL injection, token forgery, input fuzzing
Load Testing	Simulate concurrent users to ensure scalability

Testing pipeline runs via **GitHub Actions CI**.

---

14. Deployment & Monitoring

Category	Description
CI/CD	Automated pipeline (build → test → deploy) via GitHub Actions
Hosting	Railway container deployment
Monitoring	Structured logging (JSON), Railway metrics, alert hooks
Error Tracking	Logged via centralized service (e.g., Sentry)
Metrics	API latency, DB health, active users, error rates
Alerts	Triggered on failed builds, downtime, or anomalies

---

15. Future Enhancements

- Add ML-based credit scoring for improved risk evaluation.
- Introduce mobile-native app integration via REST endpoints.
- Multi-language support for global borrower inclusion.
- Event-driven architecture (Kafka) for scaling notifications and transactions.