BNPL Debt & Refund Service

Fintech Microservice Project

Project Overview

- Role: Middle Python Backend Developer
- Domain: Fintech / BNPL / Payments / Risk Management
- Timeline: 7 days
- Repository: GitHub link
- Author:Jurabek Tojiddinov

Objectives

Design and implement a robust Debt & Refund Management microservice for a BNPL (Buy Now, Pay Later) fintech platform.

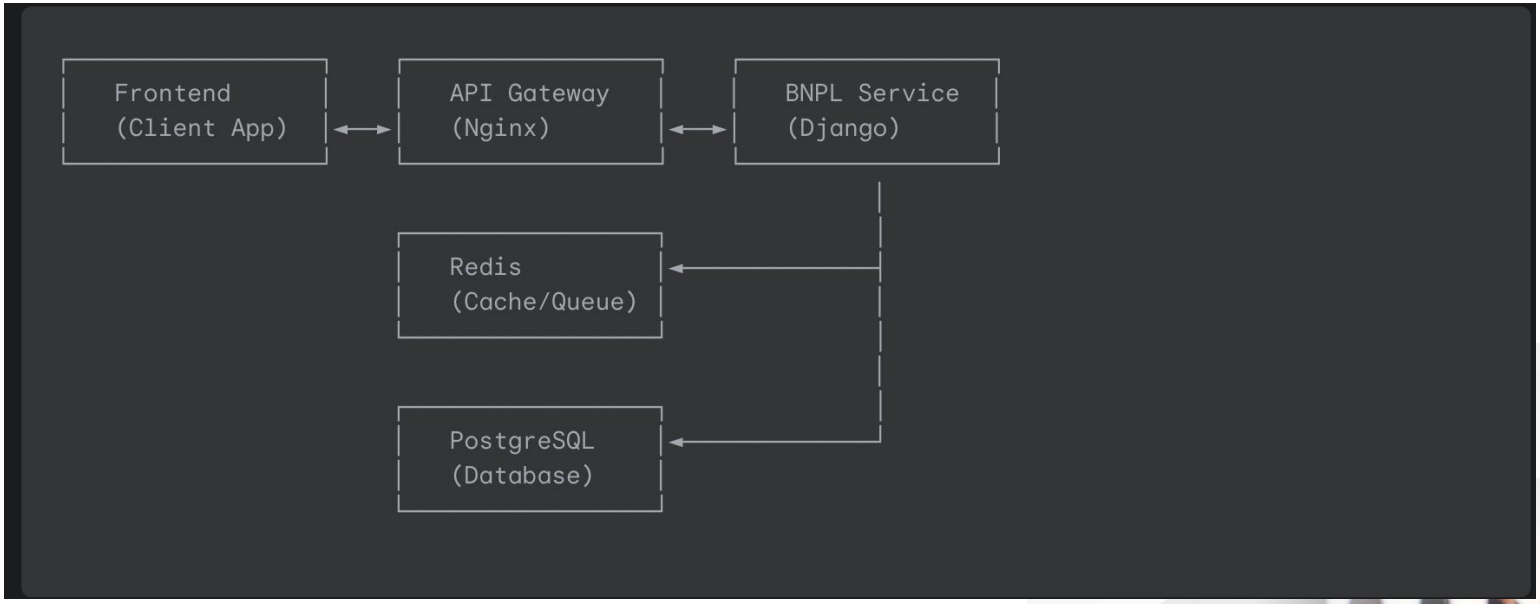
Core Requirements:

- Manage BNPL installment plans
- Monitor and enforce debt status
- Handle refund requests with idempotency
- Protect sensitive user data with masking
- Provide reliable and secure endpoints

Architecture

Architecture Overview

- The system is designed as a microservice, with a clear separation of concerns.



Database Models

The core of the service is built on four key models:

Core Models:

- User Model: Stores user details and their debt status.
- BNPL Plan Model: Manages the overall plan, total amount, and links to the user.
- Installment Model: Tracks individual payments, due dates, and statuses.
- Refund Model: Handles refund requests and their processing status.

Security & Data Protection

Key Security Features:

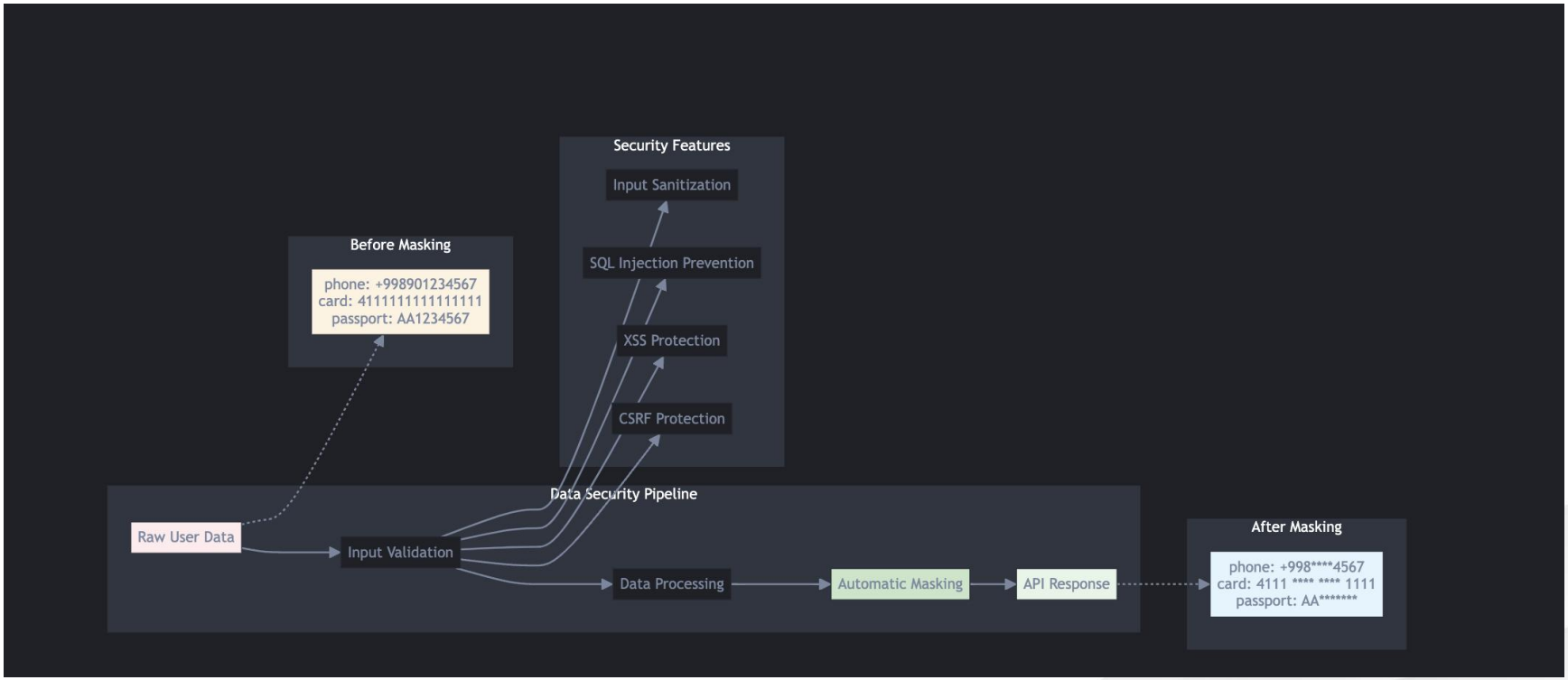
- Automatic Data Masking: Sensitive data is automatically masked in API responses to protect user privacy.
- No Sensitive Data Logging: Prevents sensitive information from being stored in logs.
- Input Validation & Sanitization: Protects against common vulnerabilities like SQL injection.
- CORS Configuration: Ensures secure communication between different origins.

API Highlights

The service provides a comprehensive set of RESTful API endpoints for all core functions.

Key Endpoint Groups:

- User & BNPL Management: Endpoints to create and manage users and their BNPL plans.
- Debt Management: APIs to check debt status and process repayments.
- Refund Management: Endpoints for creating, approving, and canceling refund requests.
- System Endpoints: Provides access to health checks and API documentation.


Security Features
Input Sanitization
SQL Injection Prevention
XSS Protection
CSRF Protection
Before Masking
phone: +998901234567
card: 4111111111111111
passport: AA1234567
Data Security Pipeline
Raw User Data
Input Validation
Data Processing
Automatic Masking
API Response
After Masking
phone: +998****4567
card: 4111 **** **** 1111
passport: AA*******

Business Logic

The service includes sophisticated business logic to handle key operations:

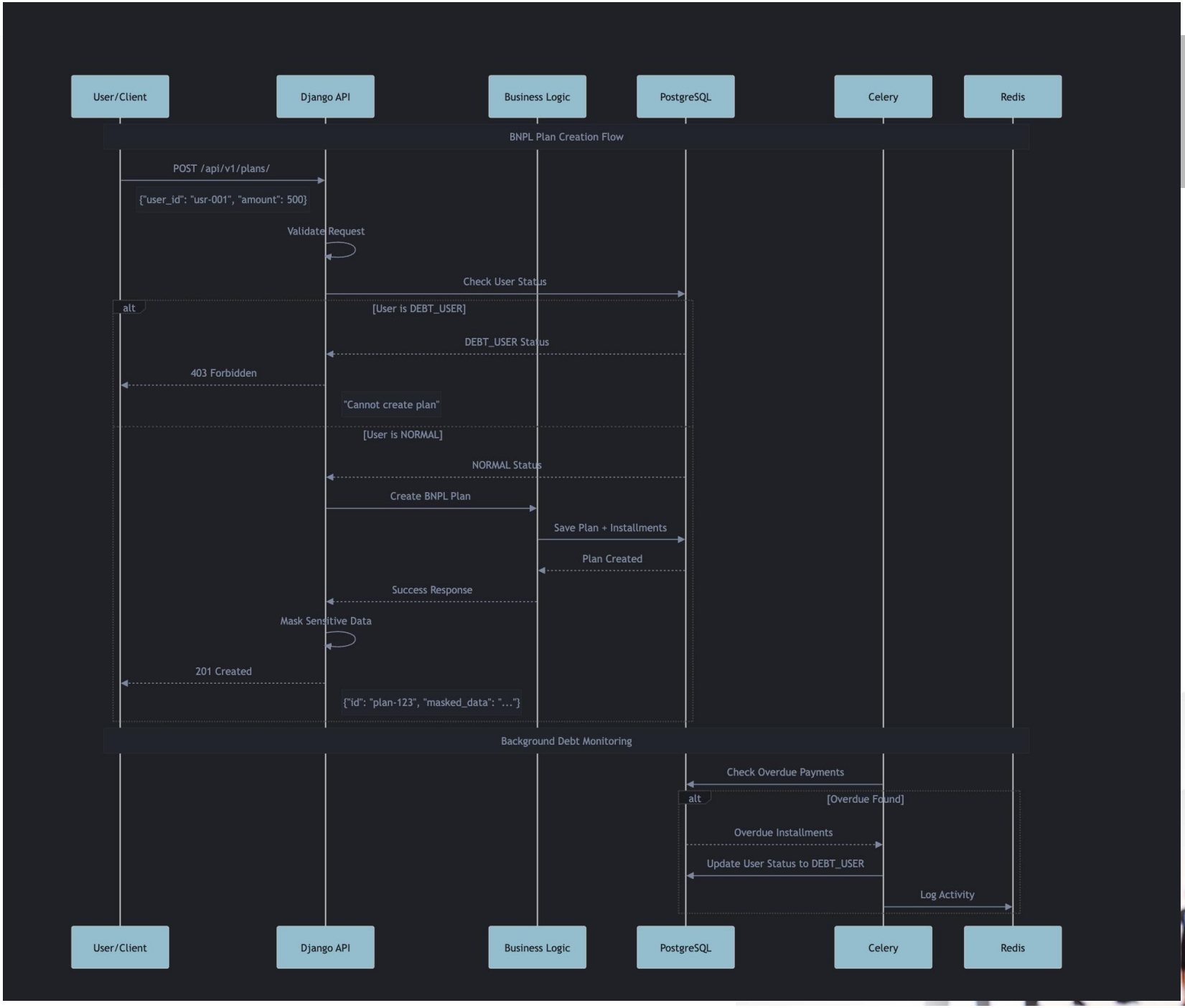
- BNPL Plan Creation: Includes validation for user status and automatic installment generation.
- Debt Management: Automatically detects overdue payments and updates user status.
- Refund Workflow: Utilizes idempotency to prevent duplicate refund requests and integrates with asynchronous webhooks for processing.

Async Processing (Celery)

Celery is integrated to handle background tasks and improve performance.

Celery Benefits:

- Non-blocking Operations: Handles tasks like refund processing without blocking the main API thread.
- Scheduled Background Jobs: Automatically runs tasks, such as checking for overdue payments, on a set schedule.
- Real-time Webhook Processing: Ensures instant updates from merchants.

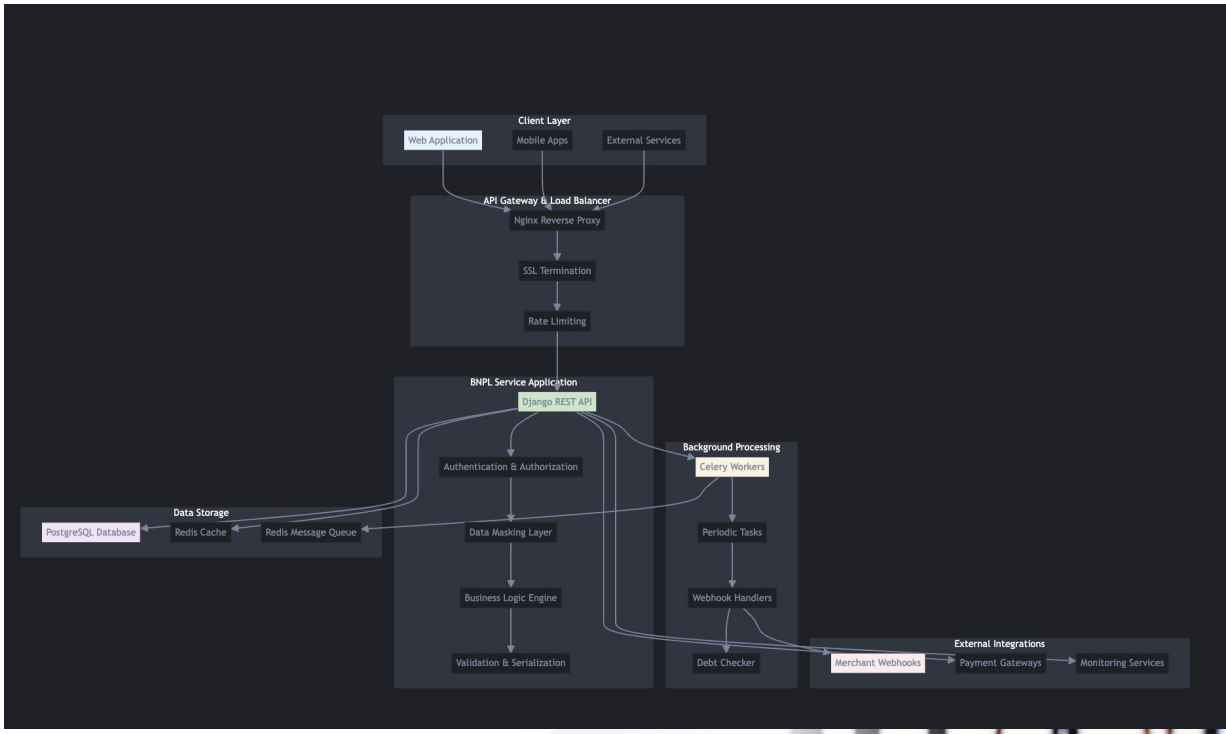

User/Client
Django API
Business Logic
PostgreSQL
Celery
Redis
BNPL Plan Creation Flow
POST /api/v1/plans/
{"user_id": "usr-001", "amount": 500}
Validate Request
Check User Status
alt
[User is DEBT_USER]
DEBT_USER Status
403 Forbidden
"Cannot create plan"
[User is NORMAL]
NORMAL Status
Create BNPL Plan
Save Plan + Installments
Plan Created
Success Response
Mask Sensitive Data
201 Created
{"id": "plan-123", "masked_data": "..."}
Background Debt Monitoring
Check Overdue Payments
alt
[Overdue Found]
Overdue Installments
Update User Status to DEBT_USER
Log Activity
User/Client
Django API
Business Logic
PostgreSQL
Celery
Redis

Testing & QA

A strong focus on testing ensures the reliability and security of the service.

Test Coverage:
Total Tests: 18

- Categories: Unit Tests, Integration Tests, Business Logic Tests, and Security Tests.
- Coverage: All core functionality is thoroughly tested.



Bonus Features

The project goes beyond the core requirements with several advanced features.

p.s we have got some billing issue so we couldn’t manage workflow you can see problem next page

Bonus Features:

- Async Overdue Checker
- Refund Status Webhook
- Debt Amount Endpoint
- Interactive Swagger Documentation
- Automated CI/CD Pipeline

All workflows

Showing runs from all workflows

Filter workflow runs

4 workflow runs	Event	Status	Branch	Actor
Create ultra-minimal workflow to avoid billing issues **BNPL Service CI/CD Pipeline (Minimal)** #4: Commit 8fd09f9 pushed by Tojiddinov	main		45 minutes ago 7s	...
Fix GitHub Actions workflow syntax error **BNPL Service CI/CD Pipeline (Minimal)** #3: Commit b0f41eb pushed by Tojiddinov	main		48 minutes ago 5s	...
Fix health check test and update CI/CD workflow for free tier **BNPL Service CI/CD Pipeline (Minimal)** #2: Commit 9b1d4d2 pushed by Tojiddinov	main		49 minutes ago *Failure*	...
Complete all bonus tasks - 100% bonus implementation **BNPL Service CI/CD Pipeline (Minimal)** #1: Commit e9d3c24 pushed by Tojiddinov	main		9 hours ago 6s	...

Deployment

The service is designed for production with a robust Docker-based setup.

Deployment Features:

- Docker Configuration: A multi-service setup including Django, PostgreSQL, Redis, Celery, and Nginx.
- Kubernetes Ready: Designed with manifests for easy deployment to Kubernetes.
- Production Best Practices: Includes SSL/TLS encryption, rate limiting, and health monitoring.

Performance & Scalability

The architecture is built to be high-performing and scalable.

Key Strategies:

- Database Optimization: Strategic indexing and optimized queries.
- Caching: Utilizes Redis for sessions and task queues.
- Scalability: Microservice architecture and async processing allow for independent and horizontal scaling.

Key Achievements

Delivery Excellence:

Timeline: Completed ahead of schedule.

- Requirements: 100% of core requirements met.
- Quality: All tests passing, demonstrating a high-quality implementation.

Technical Excellence

Technical Excellence:

- Architecture: Production-ready and scalable.
- Security: Implements industry best practices.
- Bonus Achievement: All 5 bonus tasks were completed, demonstrating innovation and excellence.

Future Enhancements

- ML-based risk scoring
- Real-time notifications
- Analytics dashboard
- Multi-tenant support
- Event-driven architecture

Thank you

Questions & Discussion Welcome!