Product Requirements Document (PRD): Simple Currency Exchange Service (CES) - Django

1. Introduction and Goals

Project Name: Simple Currency Exchange Service (CES)
Target Audience: Internal systems (e.g., Treasury, Billing microservices) requiring real-time currency conversion rates.
Version: 1.0 (Python/Django Edition)

1.1 Project Objective

To build a highly reliable, low-latency, and auditable service for handling foreign exchange (FX) rates using a Python/Django backend. The service must abstract external provider dependency and provide strict data logging.

1.2 Core Goal: Data Timeliness and Auditability

The service must ensure that the FX rates provided are always the most recently available and that every conversion is logged with an immutable reference to the exact rate used.

1.3 Key Functional Requirements (What it Does)

ID	Feature	Description	Priority
FR1.1	**Scheduled Rate Ingestion**	The service must use **Celery Beat** to automatically and resiliently fetch the latest FX rates from an external provider (via a dedicated API Client) at a configurable interval (default: hourly).	P1
FR1.2	**Rate Query API**	Expose a REST endpoint to query the latest rate for a specific pair (e.g., GET /api/rates?from=USD&to=NGN). This must be	P1

		served from the cache.	
FR1.3	**Conversion API**	Expose a REST endpoint to calculate the converted amount, applying a pre-defined spread/margin.	P1
FR1.4	**Conversion Audit Trail**	Every executed conversion must generate a persistent log entry, recording the input, output, and the specific immutable ExchangeRate record ID used.	P1
FR1.5	**Django Admin Visibility**	All ingested rates and conversion audit logs must be immediately viewable and manageable via the Django Admin interface.	P2

1.4 Non-Functional Requirements (How Well it Works)

Category	**Requirement**	**Target**
Performance	**Latency (Read)**	Rate lookups served from the **Redis Cache** must complete in under **10 milliseconds** (accounting for Django overhead).
Reliability	**Ingestion Resilience**	If the external FX API fails, the service must use the last successfully stored rate from the cache and issue a log warning. Celery must handle automatic retries for failed tasks.
Auditability	**Immutability**	Stored ExchangeRate records must be immutable (no updates, only new insertions).
Integrity	**Precision**	All rates and amounts must be handled using Python's

		Decimal type (mapped to high-precision DB fields).