TSES, ABUJA

Backend Developer Technical Assessment.

Issued: 30 December 2025 | Deadline: 3 calendar days after receipt

1. Overview

Build a small authentication service that issues email-based one-time passwords (OTPs) with Redis-backed rate limiting, asynchronous processing via Celery, JWT authentication, and complete OpenAPI documentation via drf-spectacular. You will demo the API live via screen share, endpoint by endpoint.

2. Required Tooling

You must use the following:

- Django
- Django REST Framework (DRF)
- Redis
- Celery
- djangorestframework-simplejwt
- drf-spectacular (Swagger UI)
- Docker and docker-compose

3. Architecture Requirements

The solution must be split into multiple Django apps (modular design). A typical structure is below; you may rename apps, but responsibilities must remain clearly separated.

- apps/accounts: OTP request/verify flows, JWT issuance, user creation/update
- apps/audit: audit log model, filters, and read-only endpoints

Business logic should not be dumped entirely inside views. Use serializers plus a small service layer or utilities to keep code understandable during the demo.

4. Functional Requirements

4.1 OTP Request

Endpoint: POST /api/v1/auth/otp/request

Request body:

{"email": "user@example.com"}

Behavior:

- Generate a 6-digit OTP.
- Store the OTP in Redis with a 5-minute TTL.
- Rate limit using Redis counters with TTLs:
    - Max 3 OTP requests per email per 10 minutes
    - Max 10 OTP requests per IP per 1 hour
- Return 429 Too Many Requests when throttled, including a helpful error payload (and retry-after seconds if possible).
- Enqueue a Celery task to 'send' the OTP email (console output is fine).
- Enqueue a separate Celery task to create an audit log entry (event = OTP_REQUESTED).

Response:

- 202 Accepted on success with OTP expiry information.

4.2 OTP Verify

Endpoint: POST /api/v1/auth/otp/verify

Request body:

{"email": "user@example.com", "otp": "123456"}

Behavior:

- Validate the OTP against Redis.

- OTP must be one-time-use: delete the Redis OTP key after successful verification.
- Track failed attempts in Redis with TTL: max 5 failed attempts per email per 15 minutes.
- After lockout, return 423 Locked (or 429) consistently, with an unlock ETA (seconds).
- On success, create or update a user record and issue JWT tokens using SimpleJWT.
- Enqueue audit logs asynchronously via Celery for OTP_VERIFIED, OTP_FAILED, and OTP_LOCKED events.

4.3 Audit Logs

Endpoint: GET /api/v1/audit/logs

Requirements:

- JWT authentication required.
- Return a paginated list of audit entries.
- Support filtering via query params: email, event, from, to (date/time).
- Ordering by newest first by default.

5. Data Model Requirements

Minimum required model(s):

- AuditLog(id, event, email, ip_address, user_agent, metadata(JSON), created_at)
- User model: you may use Django's default User or a custom user. Email must be used for OTP flows.

6. Redis Requirements

Redis must be used for OTP storage and counters. Use atomic operations (e.g., INCR with EXPIRE) so rate limits are correct even under concurrent requests.

7. Celery Requirements

Must include at least these Celery tasks:

- send_otp_email(email, otp): may print to console; must run asynchronously

- write_audit_log(event, email, ip, meta): inserts AuditLog row asynchronously

8. API Documentation Requirements (drf-spectacular)

Swagger UI must be enabled and show accurate schemas. Each endpoint must include summary/description and correct status codes. Include example requests and example responses in the OpenAPI schema.

9. Docker Requirements

A docker-compose setup is required with official images of required dependencies. At minimum:

- web (Django)
- postgresql
- redis
- celery_worker

The project must run from a clean machine with a single command: docker compose up --build

10. What You Will Demo (Screen Share)

Please demo in this order:

1. Start the system via docker compose up --build
2. Open Swagger UI and show the documented schemas
3. OTP request - happy path (show 202 response)
4. OTP request - email rate limit triggered (show 429)
5. OTP verify - wrong OTP (show failure and counter behavior)
6. OTP verify - lockout triggered (show 423 and unlock ETA)
7. OTP verify - success (show JWT tokens)
8. Audit logs - authenticated access, filtering, pagination, newest-first ordering
9. Show Celery worker logs proving tasks executed (email task and audit task)

11. Deliverables

- A public GitHub repository link (emailed back to the hiring team).
- README with setup steps, env vars, and how to reach Swagger UI.
- A .env.example file with all required environment variables.
- No deployment is required (we don't want you using infrastructure setup as an excuse for not completing the task within the given timeframe).

12. Deadline

Deadline is three (3) calendar days after you receive this assignment. Please submit the public repository URL by email.

13. Evaluation Criteria

- Correctness: OTP TTL, one-time-use, rate limiting, lockout behavior
- Asynchronous design: requests do not block on Celery tasks; tasks run and are observable in logs
- Redis correctness: atomic counters with TTL; no fragile race conditions
- Modularity: clean split into Django apps with clear responsibilities
- Documentation quality: drf-spectacular schemas, examples, correct status codes
- Developer experience: docker compose is reliable; README is clear

To submit this assessment, send an email containing the URL to the repo and other deliverables to

victoriaoladosu.tses@gmail.com and copy info@tsesltd.com.ng

Best of Luck!