Pandar Node.js Backend Assessment

In-Memory Wallet API (No Database)

Objective

Build a Node.js API with 5 main endpoints.

Endpoints

1) POST /user

Body:

{ "email": “user@abc.xyz” }

- The user must have an initial balance of **10,000**.
- This initial balance must be recorded
- It must return a JWT token that will be used for the other endpoints

2) GET /balance

Header: Bearer token

Returns the current wallet balance of a logged-in user.

3) POST /add_balance

Header: Bearer token

Body:

{ "amount": 5000 }

Requirements:

- amount must be a positive integer

- Must require Idempotency-Key header

- Must be idempotent

- Must update balance correctly

- Must create a transaction record in a double-entry ledger

4) POST /withdraw

Header: Bearer token

Body:

{ "amount": 1000 }

Requirements:

- amount must be a positive integer

- Cannot exceed balance

- Must require Idempotency-Key header

- Must be idempotent

- Must be concurrency safe (balance must never go negative)

- Must create a transaction record

5) GET /transactions

Header: Bearer token

Returns a paginated transaction history (most recent first).

Transaction fields:

- type (credit | withdraw)

- amount (integer)

- reference (string)

- createdAt (ISO timestamp)

General Requirements

- In-memory storage only (no database, no Redis)

- Strict input validation

- Properly structured error responses

- No stack traces in production
- Must handle concurrent requests correctly
- Clean project structure
- Implement basic rate limiting on mutating endpoints
- Return consistent JSON error structure
- Proper organization of codebase
- Provide a clear README with setup and test instructions