Real-Time Assessment Tracker (Frontend Perspective)

Kafka + WebSocket Integration with FastAPI Backend

Revonera Team

July 2, 2025

Overview

This document outlines the frontend-level theoretical design for real-time event tracking in the Revonera assessment tool. The architecture connects Kafka (message queue) and WebSocket (real-time updates) to support candidate activity tracking, using FastAPI in the backend.

Objective

To provide real-time updates to employers or admins during live assessments, including:

- Assessment lifecycle events (start, submit, disconnect)
- Answer submission tracking
- Warning alerts for suspicious candidate behavior

System Architecture (Frontend-Focused)

Step 1. Candidate performs an action on the test UI (e.g., submits an answer, changes tab, presses Ctrl+C).

Step 2. The frontend calls FastAPI backend, which processes the activity.

Step 3. FastAPI produces a Kafka event to a topic (e.g., `assessment_activity`).

Step 4. A Kafka consumer running in a WebSocket server receives the event.

Step 5. The WebSocket server pushes the event to all connected frontend clients (e.g., admin dashboards).

Step 6. The frontend dashboard displays the event in real-time.

Frontend Responsibilities

- Establish and maintain a live WebSocket connection with the server.
- Listen for events emitted by the WebSocket server.
- Update UI components with incoming events (e.g., status changes, warnings).
- Optionally filter or highlight critical warnings for review.

Event Types and Descriptions

1. Assessment Lifecycle Events

- **assessment_started** – Candidate begins the test.
- **assessment_submitted** – Candidate completes the test.
- **user_disconnected** – Candidate disconnects unexpectedly (e.g., closes tab).

2. Answer Submission Events

- **answer_submitted** – A candidate submits an answer to a question.

3. User Warning Events (Suspicious Behavior)

Triggered when the frontend detects abnormal behavior and notifies the backend, which emits Kafka warnings.

- **tab_switch_detected** – Candidate switched tabs during the test.
- **copy_attempt** – Copy action (Ctrl+C or right-click copy) was attempted.
- **paste_attempt** – Paste action was attempted.
- **inspect_element_opened** – Developer tools opened.
- **fullscreen_exit** – Candidate exited fullscreen mode.

Each warning event contains metadata such as:

- Candidate ID
- Assessment ID
- Timestamp
- Event type

Benefits

- Provides live visibility into candidate behavior.
- Helps employers monitor engagement and test integrity.
- Scales easily with Kafka's distributed nature.
- Frontend remains reactive with lightweight WebSocket listeners.

Technologies Used

- **Frontend:** Next.js, React, Socket.IO (WebSocket)
- **Backend:** FastAPI (Python)
- **Real-Time Transport:** WebSocket Server (Node.js)
- **Message Broker:** Apache Kafka

Conclusion

This frontend integration ensures that admins and employers receive accurate, live updates about ongoing assessments. By combining Kafka and WebSocket, the system is scalable, fast, and ideal for high-stakes test environments.

Note: This document focuses solely on frontend theory. Implementation details and backend configurations (Kafka producers/consumers) are covered separately.