LAIKIPIA UNIVERSITY

UNIVERSITY EXAMINATIONS

FIRST SEMESTER 2025/2026 ACADEMIC YEAR

SECOND YEAR EXAMINATION FOR THE DEGREE OF BACHELOR OF SCIENCE IN INFORMATION COMMUNICATION AND TECHNOLOGY

BICT 213: EVENT – DRIVEN PROGRAMMING

STREAM: R ***TIME: 2 HRS***

DAY: TUESDAY [11.30 – 13.30 P.M] ***DATE: 03/02/2026***

THIS QUESTION PAPER CONSISTS OF SIX (6) PAGES

PLEASE DO NOT OPEN UNTIL THE INVIGILATOR SAYS SO.

QUESTION ONE (COMPULSORY)

a) Explain the role of the event loop in event-driven programming. [**3 Marks**]

b) Compare and contrast event-driven programming with procedural programming. [**4 Marks**]

c) Describe the Publisher-Subscriber pattern. [**2 Marks**]

d) Differentiate between the Observer pattern and the Publisher-Subscriber pattern. [**3 Marks**]

e) Write a simple code snippet demonstrating the use of callbacks in event handling. [**3 Marks**]

f) Explain the inversion of control in the context of event-driven programming. **[3 Marks]**

g) List three common event types in GUI programming. **[1 Mark]**

h) Explain how event listeners are registered in a GUI framework like JavaFX. **[2 Marks]**

i) What is the DOM? How does it relate to event handling? **[1 Mark]**

j) Why is testing event-driven applications challenging? **[2 Marks]**

k) Describe a strategy for unit testing event handlers. **[3 marks]**

l) Evaluate the effectiveness of logging in debugging distributed event systems. (**3 marks**)

m) What are WebSockets? How do they enable real-time communication? (**1 mark**)

QUESTION TWO

"You are the lead architect for 'FlashSale Frenzy,' an e-commerce platform that runs limited-time, high-discount sales on popular items. During a 10-minute flash sale for a new smartphone, the platform must handle an anticipated 50,000 concurrent users all trying to purchase a limited stock of 1,000 units.

The system is built as a set of microservices using an event-driven architecture with a message broker (Kafka/RabbitMQ).

The Core Services and Flow:

1) **API Gateway**: Receives all "Place Order" requests from users.

2) **Order Service**: Receives order requests, publishes an OrderRequested event containing userId, productId, and orderId.

3) **Inventory Service**: Listens for OrderRequested events. It checks and reserves stock using an atomic operation to prevent overselling. It then publishes either an InventoryReserved event or an InventoryDenied event.

4) **Payment Service**: Listens for InventoryReserved events. It calls an external payment gateway to process the payment. Based on the result, it publishes a PaymentConfirmed or PaymentFailed event.

5) **Order Fulfilment Service**: Listens for PaymentConfirmed events to begin the shipping process.

6) **Notification Service**: Listens for all events (Inventory Reserved, Inventory Denied, Payment Confirmed, Payment Failed) and sends real-time updates to the user accordingly.

A critical business rule is that an item can only be reserved for 15 minutes after an InventoryReserved event is published. If a PaymentConfirmed event is not received within that window, the reservation must be automatically released."

a) During the flash sale, the system fails. The Inventory Service is overwhelmed, the message broker queue is growing uncontrollably, and users are experiencing significant delays, resulting in numerous failed transactions and inventory overselling. Analyze the depicted architecture and identify two potential bottlenecks in the event flow that could cause this failure. For each bottleneck, explain why it occurs under extreme load. [**5 marks**]

b) The product manager suggests replacing the event-driven flow for payment with a direct, synchronous API call from the Order Service to the Payment Service to "simplify the process." Evaluate this suggestion by comparing the trade-offs between the proposed synchronous call and the existing event-driven approach. Your answer should consider the impact on user responsiveness, system resilience, and scalability during the flash sale. [**5 marks**]

c)To prevent the failure in Question a, a redesign is necessary. Propose a new event flow to handle the initial inventory check and reservation more efficiently. Your solution must: [**10 marks**]

d) Explain how to use the Publisher-Subscriber and Message Queue patterns effectively to distribute load.

e) Describe how to implement a pre-check or throttling mechanism at the API Gateway before an OrderRequested event is published.

f) Justify how your redesign prevents the system from being overwhelmed and ensures inventory is not oversold.

QUESTION THREE

"You are the lead developer for 'Project Canvas,' a web-based collaborative design tool similar to Figma or Miro. Multiple users can be on the same digital canvas simultaneously, manipulating objects (such as shapes, text, and sticky notes) in real-time.

The core technical challenge is managing the complex, asynchronous flow of user interactions and synchronising the state across all connected clients instantly.

Key Features & Technical Context:

- The UI is built using HTML5 Canvas and JavaScript.
- User interactions (drag, drop, resize, type, delete) on any object must be immediately reflected on their own screen and broadcast to other users.
- Real-time synchronisation is handled via WebSockets, which are event-driven by nature.
- The application must also manage local state efficiently to ensure performance and a smooth user experience.

A user performs the following action:

They click on a sticky note, drag it across the canvas, and drop it in a new location."

a) Deconstruct the single-user action of "dragging and dropping a sticky note" into a sequence of at least four distinct DOM events that the browser will fire. [**8 Marks**]

For each event in your sequence:

i) Name the event (e.g., click).

ii) Describe what specific data or state (e.g., mouse coordinates, target element) is crucial to capture in the event object for this specific application.

iii) Explain how this captured data is used to update the local canvas state before any data is sent to the server.

b) Consider a situation where two users, Alex and Bailey, try to move the same sticky note at almost the exact same time. On Alex's screen, he moves it to position X. On Bailey's screen, she moves it to position Y.

i) Using the concepts of event-driven systems, explain the potential conflict that arises and what the user experience would be if the application simply overwrote the object's position with the last received event. [**3 Marks**]

ii) Propose a strategy to handle this conflict. Your strategy should leverage the event payload and a central authority (the WebSocket server) to decide the "single source of truth" for the object's position, ensuring all clients eventually converge to the same state. [**4 Marks**]

c) During a usability test, it was observed that the canvas becomes laggy when more than 20 objects are frequently manipulated. Analyse this performance issue from an event-driven perspective. Propose and justify one specific code-level optimisation (e.g., debouncing, throttling, or a more efficient event listener strategy) that could be applied to the event handlers to mitigate this lag, without breaking the real-time collaboration feature. [**5 Marks**]

QUESTION FOUR

"You are architecting the core order processing system for 'QuickBite,' a food delivery platform. The system must handle the entire order lifecycle, from a customer placing an order on their phone to the restaurant preparing it and a driver delivering it.

The backend is built using a microservices architecture with Node.js. Each service uses the EventEmitter class internally and communicates with other services via a central message broker (Apache Kafka) using events.

The Core Services:

- Order Service: Receives and validates new orders. It orchestrates the entire order flow.
- Restaurant Service: Manages menu items and receives orders for specific restaurants.
- Delivery Service: Assigns drivers and tracks delivery status.
- Notification Service: Sends SMS and push notifications to customers, restaurants, and drivers.

A customer places a successful order for a pizza. The initial flow is:

- Order Service receives the request, validates it, and emits a 'order:created' event internally.
- A listener within the Order Service catches this event, persists the order to the database with a status of 'PENDING', and then publishes an 'OrderCreated' event to a Kafka topic.
- The Restaurant Service consumes the OrderCreated event, acknowledges the order, and publishes a RestaurantAccepted event

a) The customer is allowed to cancel their order, provided the restaurant has not yet started preparing it. This requires a coordinated "transaction" across the Order, Restaurant, and Delivery services. Design a Saga pattern to handle an OrderCancelRequested event. [**10 Marks**]

Your answer should:

a) List the sequence of events and compensating events (e.g., RestaurantCancel, CompensateRestaurantAccept).

b) Explain the final state of the order if the cancellation fails at the Restaurant Service because preparation has already begun.

c) Describe the role of the Saga Orchestrator (which could be a dedicated service or the Order Service itself) in managing this flow.

b) A critical bug has been identified: occasionally, when the Order Service publishes the OrderCreated event to Kafka, it crashes after sending the event but before updating the local database's order status to PENDING. This leaves the order in an inconsistent state.

a) Explain the specific problem this crash causes. What happens when the service restarts and the restaurant tries to accept an order that doesn't exist in the Order Service's database? [**4 Marks**]

b) Design a solution using the Outbox Pattern to prevent this inconsistency. Describe the steps the Order Service must take to ensure that writing to the local database and publishing the event to Kafka are an atomic operation. [**6 Marks**]

QUESTION FIVE

a) Explain the difference between bidirectional binding and unidirectional binding in JavaFX properties. Provide a practical example where bidirectional binding would be preferable over unidirectional binding. [**4 Marks**]

b) Given an ObservableList<Student> where Student has nameProperty() and gradeProperty(), write code to:

i. Create a TableView that automatically updates when students are added/removed

ii. Bind a label to show the average grade that updates in real-time

iii. Filter the table to only show students with grades above 70 [**6 Marks**]

c) Design a custom event class, DataUpdateEvent, that can carry information about what type of data was updated and the new value. Include event type constants for different update types.[**3 Marks**]

d) Differentiate between event bubbling and event capturing in JavaFX. Provide code examples to illustrate each concept. [**3 Marks**]

e) Write a complete event handler for a button that, when clicked, changes the background color of a pane to a random color. The handler should also prevent multiple rapid clicks by disabling the button for 2 seconds after each click. [**4 Marks)**