SHINGIRAI CHANAKIRA
sechanakira@yahoo.com
sechanakira@gmail.com

1.(a) System Design:

Assumptions:

1. Near real time sync means data will constantly flow from ABC Work Management System to the H.R and Payroll System. Periodically, data will be moved between the two systems.

2. Both systems use a persistant data stores such as RDBMS databases.

Proposed solution:

The proposed solution is to stream changes from the ABC Work Management databases. A database listening software will be used to listen to changes on the ABC Work Management database and publish them to a Kafka topic. Debezium will be used for listening to these database events. An intermediary simple service will listen to the topic and persist changes on the HR/Payroll database. This service can also do any necessary transformations if needed. This solution has been chosen as it does not require any of the systems to have APIs exposed. This solution also allows the systems to change independantly.

Technologies: Debezium(https://debezium.io/) and Kafka

1.(b)

I would test the system by applying changes on the ABC Work Management System test database and checking if those changes have been published to Kafka.

1.(c)

1. Debezium/Kafka could be down and data will not sync, however when they come back online, they will capture all changes.
2. The consumer(intermediary service) can stop or crush.
3. Monitored databases can stop or crash.
4. Duplicate events may be recored when things go wrong.

1.(d) ;

When an error occurs, the intermediary service would throw an exception and the change event will be put back on the topic. The consumer is configured to periodically record its position (aka, offset) in each topic. When an application stops gracefully and closes the consumer, the consumer will record the offsets for the last event in each topic. When the application restarts at any later time, the consumer looks up those offsets and starts reading the very next events in each topic. Therefore, under normal operating scenarios, the application sees every event exactly one time.

I would also add logic to handle duplicate events.

1.(e)

1. Add more consumers to process data fatser.
2. Spawn new instances of Debezium.
3. Configure Kafka partitions for high data rates.

2. https://github.com/sechanakira/transitive-dependency-check