This document contains possible high level set of solutions/tools/technologies for implementing integration architecture for user's activity events.

Table of contents:

1. eCom Mobile App

Application produces events about user activity. For example: "Category viewed", "Product viewed", "Product added to the cart".
Events can be transferred from application to the server **synchronously or asynchronously**.
Usually tracking activity events don't require immediate action, that's why I would propose to send events **asynchronously**.
In the application it would be nice to have a **queue of events**. It will help to transfer them in case of **bad internet connection or offline usage**.
Events can be sent **one by one or in batches**.
Both ways have advantages and disadvantages. If application sends them **one by one** we can offer **faster consuming** that events by other systems downstream. Also in case of failure of transferring or any other error only one event will be lost.
With **batching events** downstream application will get them with the **delay**, but processing them might be **cheaper**.
With balancing **number of events in batch** and **conditions for releasing** events it is possible to get **optimal performance** for sending and processing events downstream.
For transferring **HTTP API** can be used or **websockets**. Both ways have advantages and disadvantages and it should be careful analysis before choosing proper technology for that.

2. Distribution Hub

When event (or events) received on the backend side it is necessary to pass it very quickly downstream.
For that I would recommend to use **data streaming platforms** such as **Amazon Kinesis** or **Kafka**. For activity events **no heavy and complex processing** is needed. Only relatively simple mapping from one contract to another. So it should be fast task.
The main question is how to organize **channels (topics)**. One way is to create a **separate topic for each event type**. For example ProductViewedTopic,

ProductViewedTopic etc. or **group similar events** in general topic. I would recommend **separate topics** because it is cheap and it is very difficult to predict what will happen with the data later. How it will change with the time. And how it will be used in future. Also **separate topics** will allow to **control subscribers separately** for each topic.

3. NoSQL database

I would recommend having a separate service (or combination of services) for saving data. Service should **listen to the topic** it is responsible for and **save data from them**.
Depends on **business model, conversion** should be performed. For activity events I don't expect much logic. But maybe data from events should be combined with other sources. This should happen it this service.
For storing I would propose distributed databases such as **Cassandra or DynamoDb** or any other database which satisfies requirement of **resilience, cost and speed**. Service should be designed in a way so **changing database is easy**.

4. Warehouse

Data warehouses is not my speciality and I've never worked close enough with it. I can imagine some solutions for the problem, but it will probably be quite naive.
There are technologies to dump raw data from data streams. For example for **Kafka** topics there is a way to dump the snapshot **to the file** and then **index it with Athena**. Also there is service **Amazon Redshift**. It is a well known solution for this task.
Maybe it would be nice to store the very **raw data** which comes directly from the **mobile application**, but it depends on requirements.