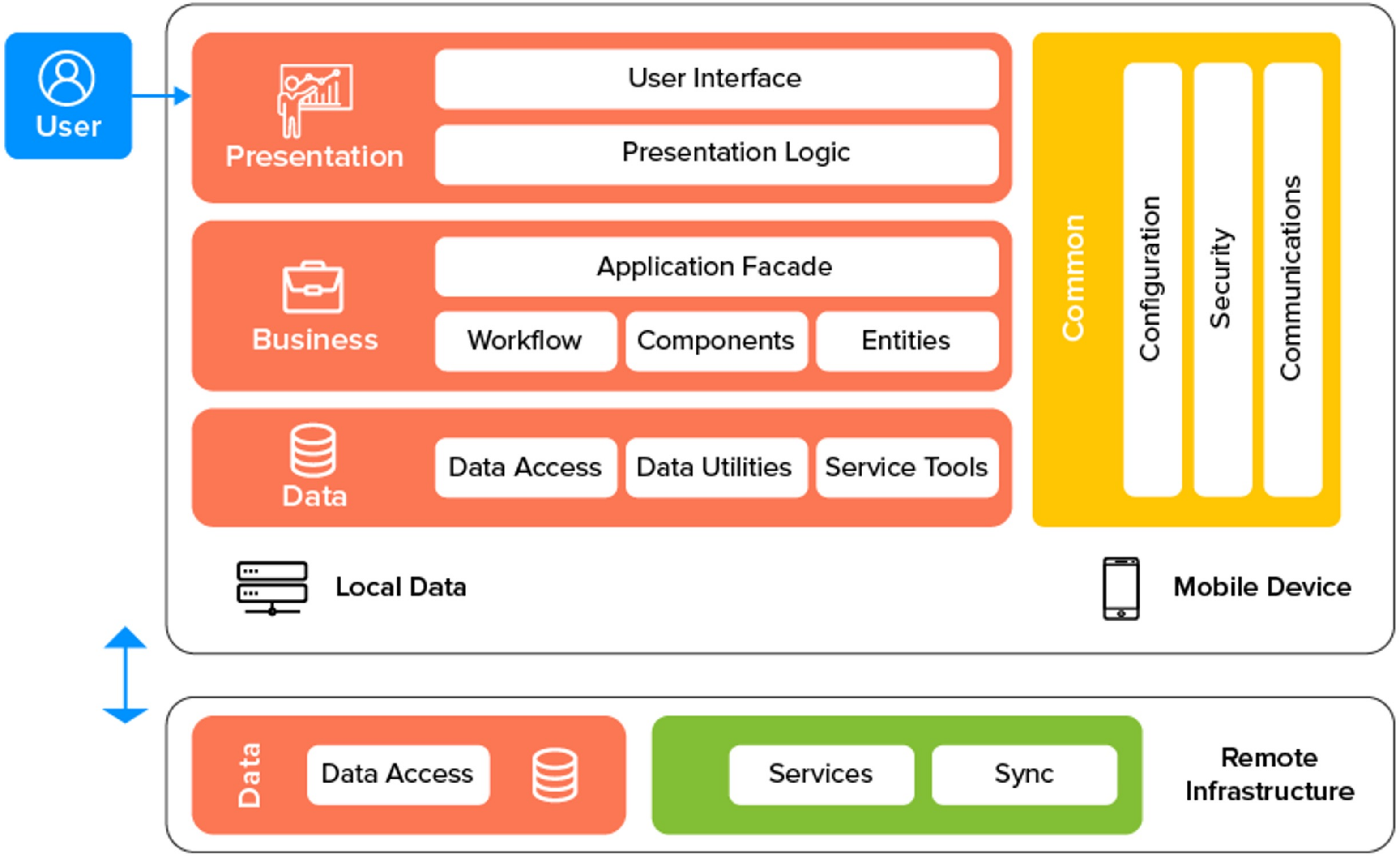
# Financial Institution example of the mobile banking app

NTU CE7 Module 1.10 Assignment

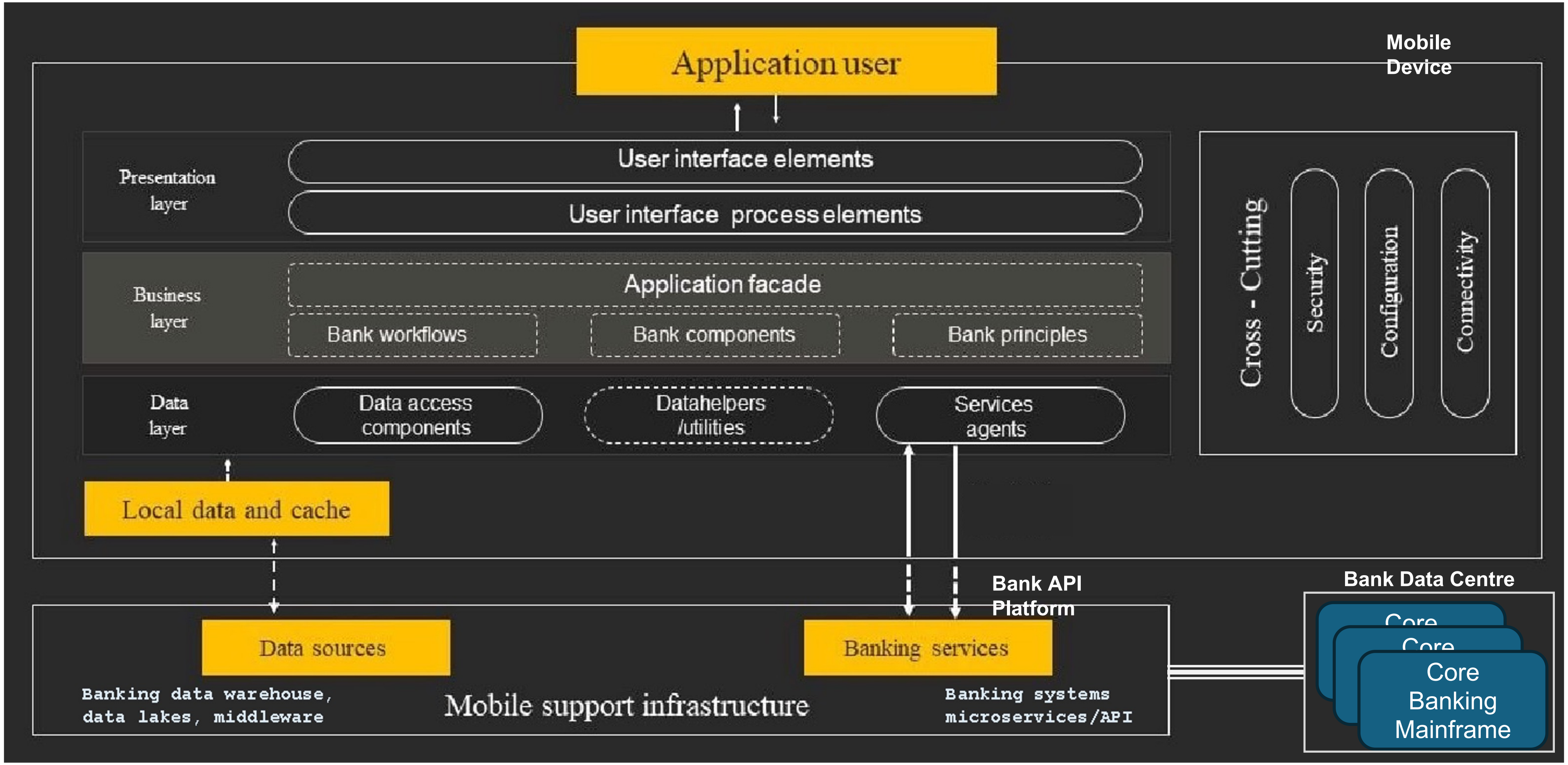
15 Jul 2024

# The Typical Mobile App Architecture Design



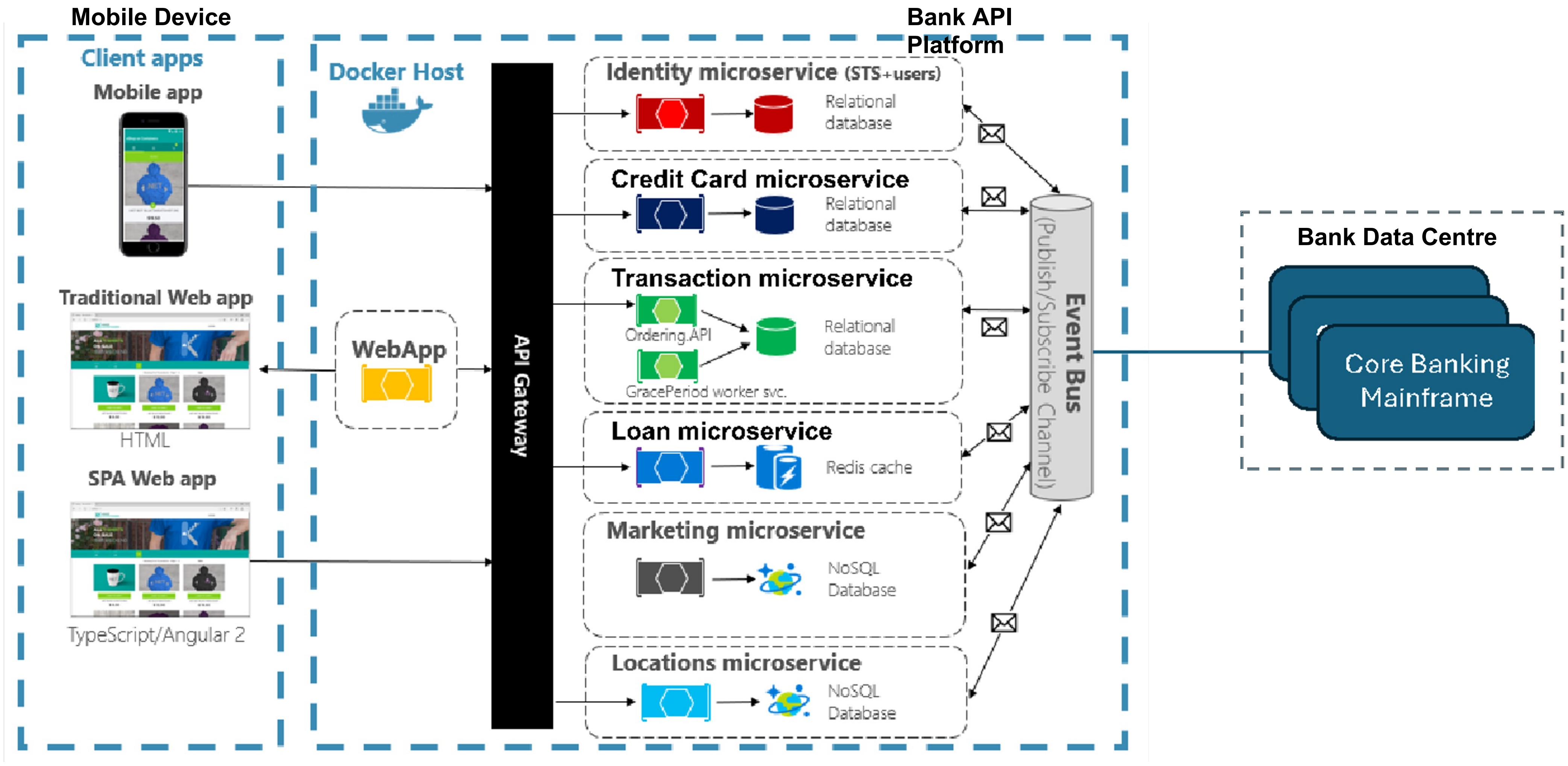
Source: https://www.webcluesinfotech.com/modern-mobile-apps-architecture-to-be-aware-of-in-2023/

# Example of a Mobile Banking App Architecture



Source: https://www.slidegeeks.com/diagram-highlighting-mobile-banking-application-architecture-ppt-summary-demonstration-pdf

# Example of a Mobile Banking App Architecture



*Source: Microsoft*

# Implementation Patterns

- **Clustering**
  - Yes, because the execution of banking transaction processing is mission-critical that must guarantee delivery so Clustering of servers is needed for high availability, scalability and reliability (including fault-tolerance).
- **Load Balancing**
  - Yes, because application LB is needed to ensure any sudden, high demand spikes of payment transactions like during periods of music event ticketing (eg. Taylor Swift).

# Implementation Patterns

- **Containerization or Virtualization**
  - Yes, to achieve cost-effective performance through server scalability and high fault-tolerance with site reliability engineering (SRE[*]), the clustering of application containerisation (using Docker + Kubernetes technology) can be a key enabler towards the design implementation of a cloud-native, microservices architecture[**] for the mobile banking app.
- **Whitelisting**
  - Yes, as adopting a zero trust security model[***] can provide greater risk assurance against cybersecurity threats and vulnerabilities in particularly of the application and data.

# Implementation Patterns

- **Blacklisting**
  - Yes, to ensure the least security risk to the mobile banking app, it is best-practice to implement firewall blacklisting that can protect against unwanted traffic accesses at the OSI layers 2-4 of data link, network and transport.
    - Suspicious IP Addresses: Blocking IP addresses known for malicious activities to prevent fraud attempts.
  - There may also be application-level blacklisting for:
    - **Fraud Prevention:**
      - Compromised Devices: Blacklisting devices identified in fraudulent transactions to prevent further unauthorized access.
      - User Accounts: Blocking user accounts involved in suspicious activities until further investigation.
    - **Transaction Security (eg. for credit cards):**
      - Card Number Blacklisting: Blocking stolen or compromised credit/debit card numbers to prevent unauthorized transactions.
      - Merchant Blacklisting: Blocking transactions with merchants known for fraudulent activities or disputes.
    - **Access Control (who can access the app, what they can do once they are in, ensuring that only authorized users have the right level of access.) :**
      - Geographic Restrictions: Blocking access from regions where the bank does not operate or has higher incidences of fraud.
      - Unauthorized Access Attempts: Blacklisting users or devices after multiple failed login attempts to prevent brute-force attacks.

# *Site Reliability Engineering

- Site Reliability Engineering (SRE) is a set of principles and practices that applies aspects of software engineering to IT infrastructure and operations. SRE claims to create highly reliable and scalable software systems. Although they are closely related, SRE is slightly different from DevOps.

*Source: https://g.co/kgs/p5Cmbpf*

# **Microservices in Cloud Native Architecture

- Microservices in the Cloud Native Architecture where Microservices and Containers are two key components of cloud-native architecture that enable developers to build and deploy applications that are scalable, resilient, and flexible.

*Source: https://medium.com/@iamsoumyadip/microservices-in-the-cloud-native-architecture-723eb65f2f33*

# ***Zero Trust Security Model

- Zero trust security model whose main concept is "never trust, always verify", which means that users and devices should not be trusted by default, even if they are connected to a permissioned network such as a corporate LAN and even if they were previously verified. According to Microsoft, the zero trust security architecture framework is based upon the zero trust principles to verify explicitly, use least-privilege access and assume breach.
    - Verify explicitly: Always authenticate and authorize based on all available data points, including user identity, location, device health, service or workload, data classification, and anomalies.
    - Use least-privilege access: Limit user access with just-in-time and just-enough access (JIT/JEA), risk-based adaptive polices, and data protection to help secure both data and productivity.
    - Assume breach: Minimize blast radius and segment access. Verify end-to-end encryption and use analytics to get visibility, drive threat detection, and improve defenses.

*Source: https://en.wikipedia.org/wiki/Zero_trust_security_mode*
*https://www.microsoft.com/en-us/security/business/zero-trust*