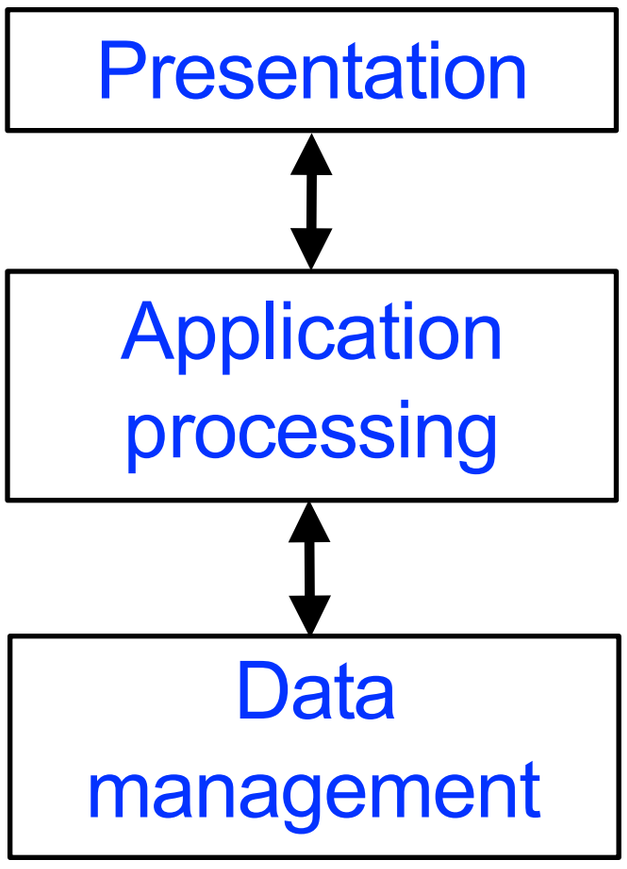
Problem : Design of a Client-Server System for Banking

Problem: Design of a Client-Server System

- Input: an informal description of an application (e.g., banking application)



concerned with user interface

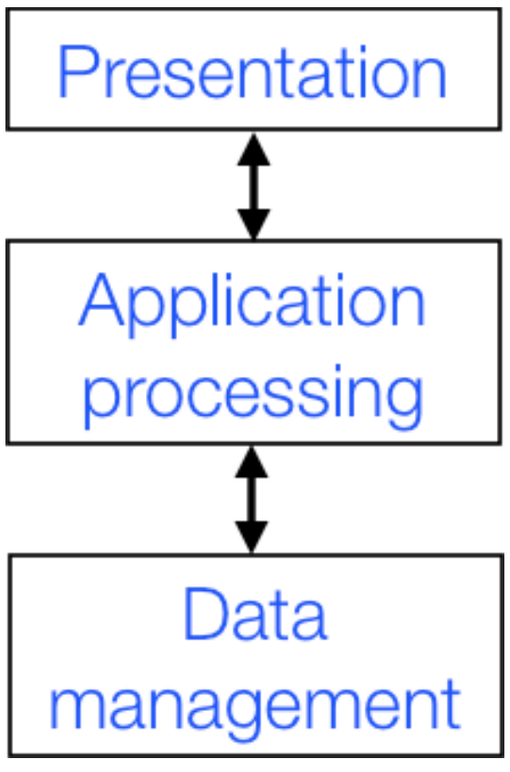
concerned with the detailed application-specific processing associated with the application

concerned with the persistent storage of the application (typically a DBMS)

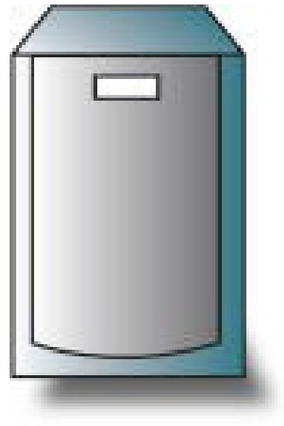
- Output: client-server implementation of the application

Solution 1: Two-Tier Client-Server Architecture

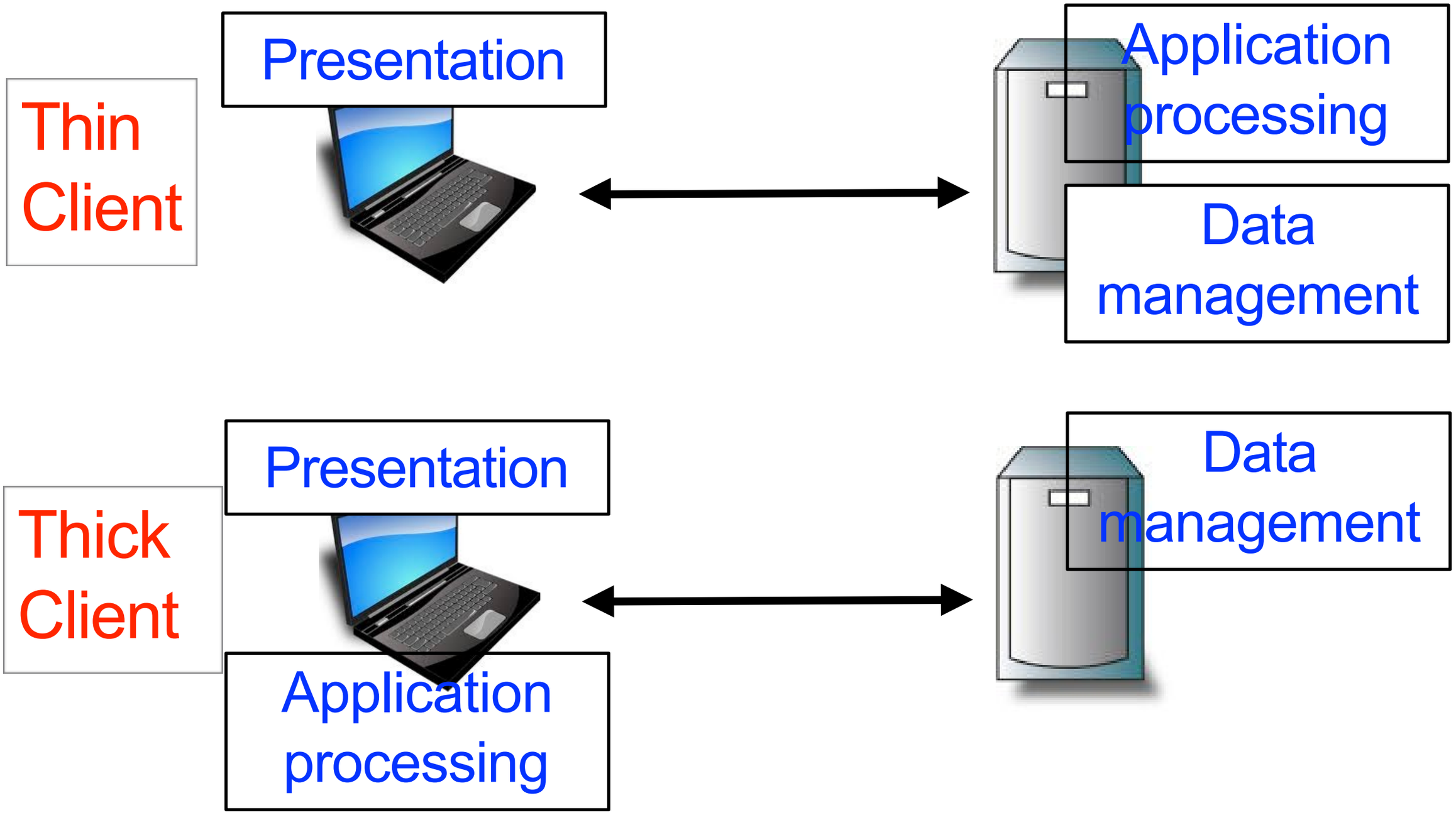
- Application organized as a server and a set of clients
- Two kinds of machines: client machines and server machines



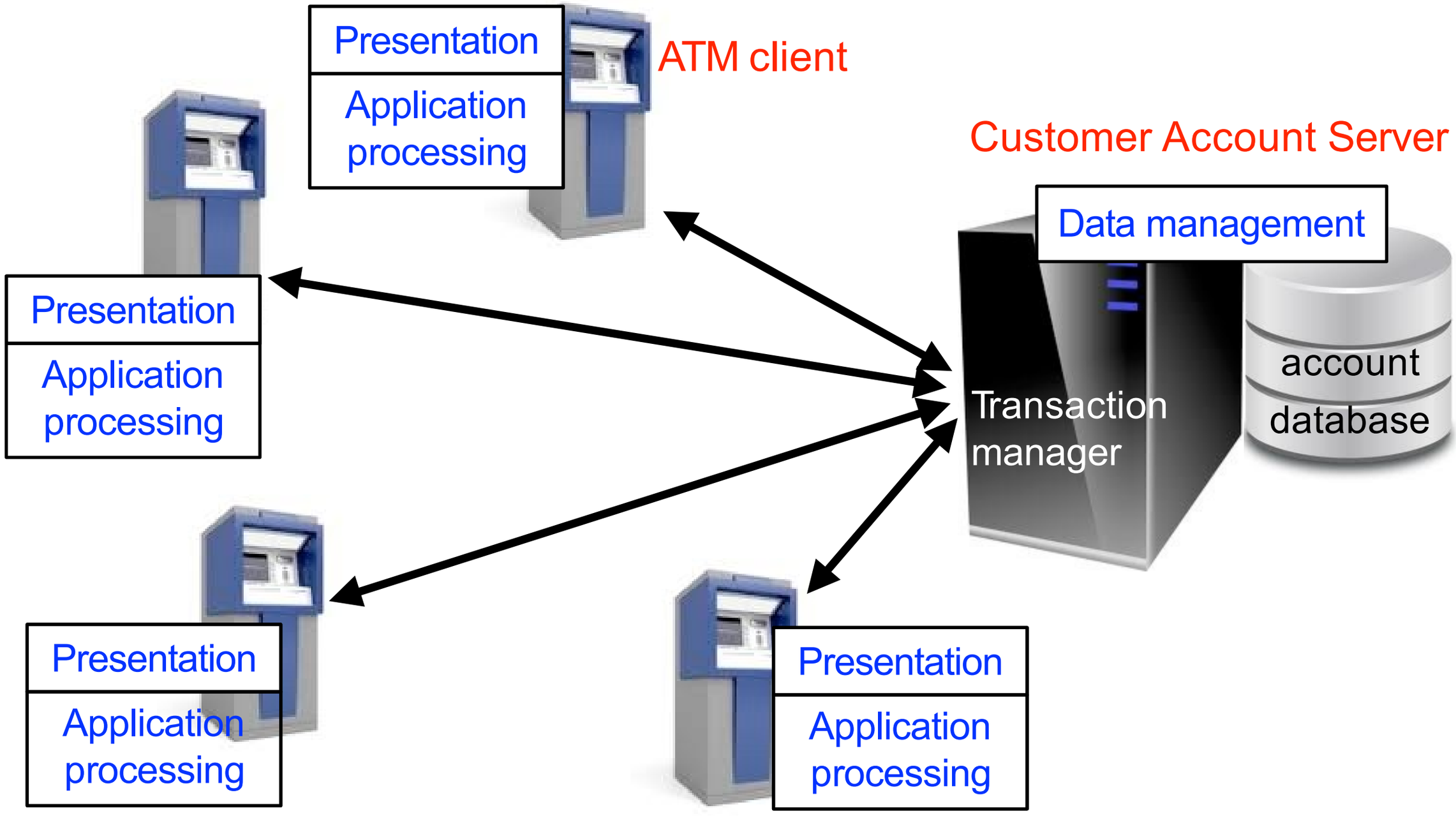
how to map
3 application layers
into a 2-tier architecture?



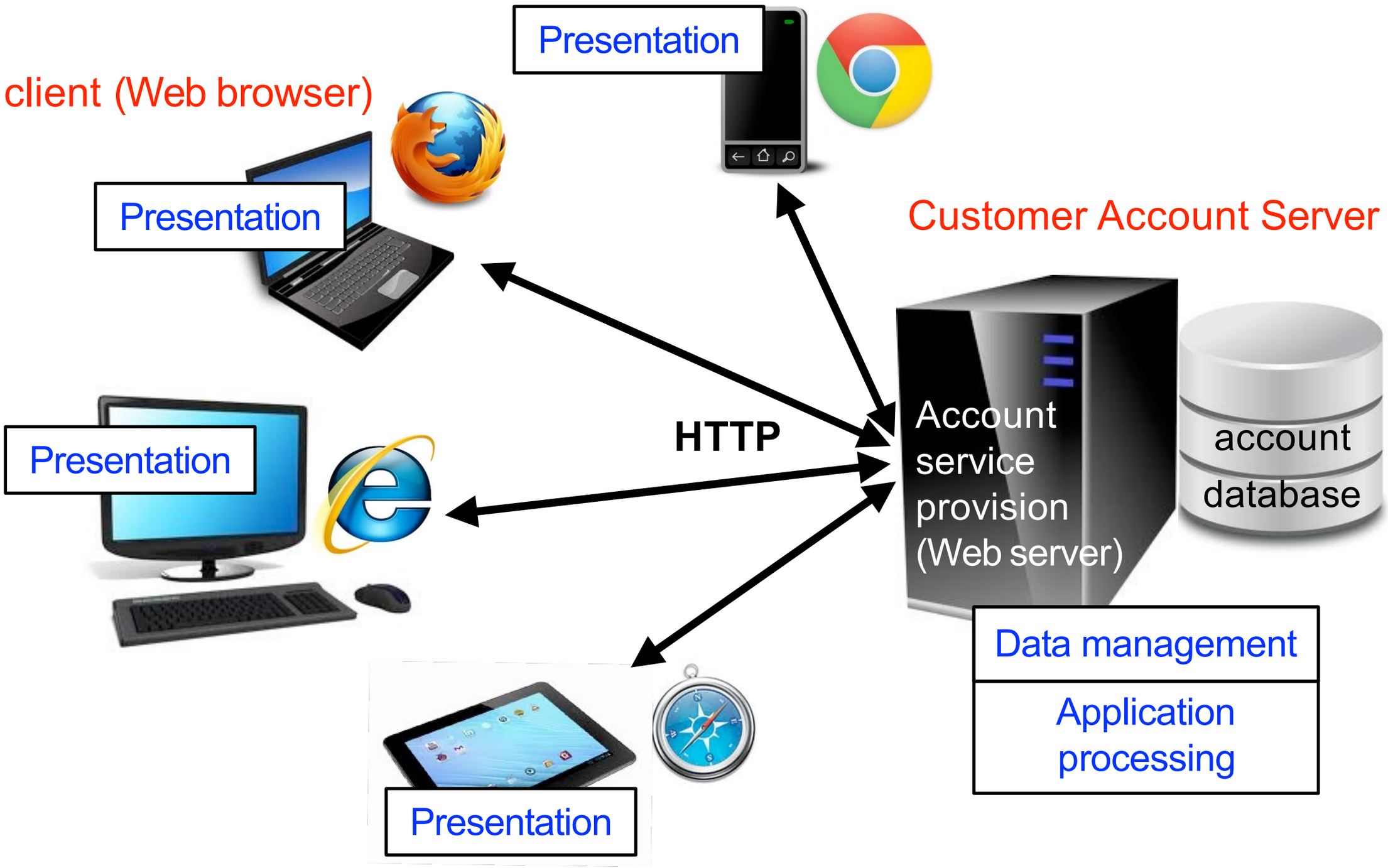
Thin VS Thick Client Model



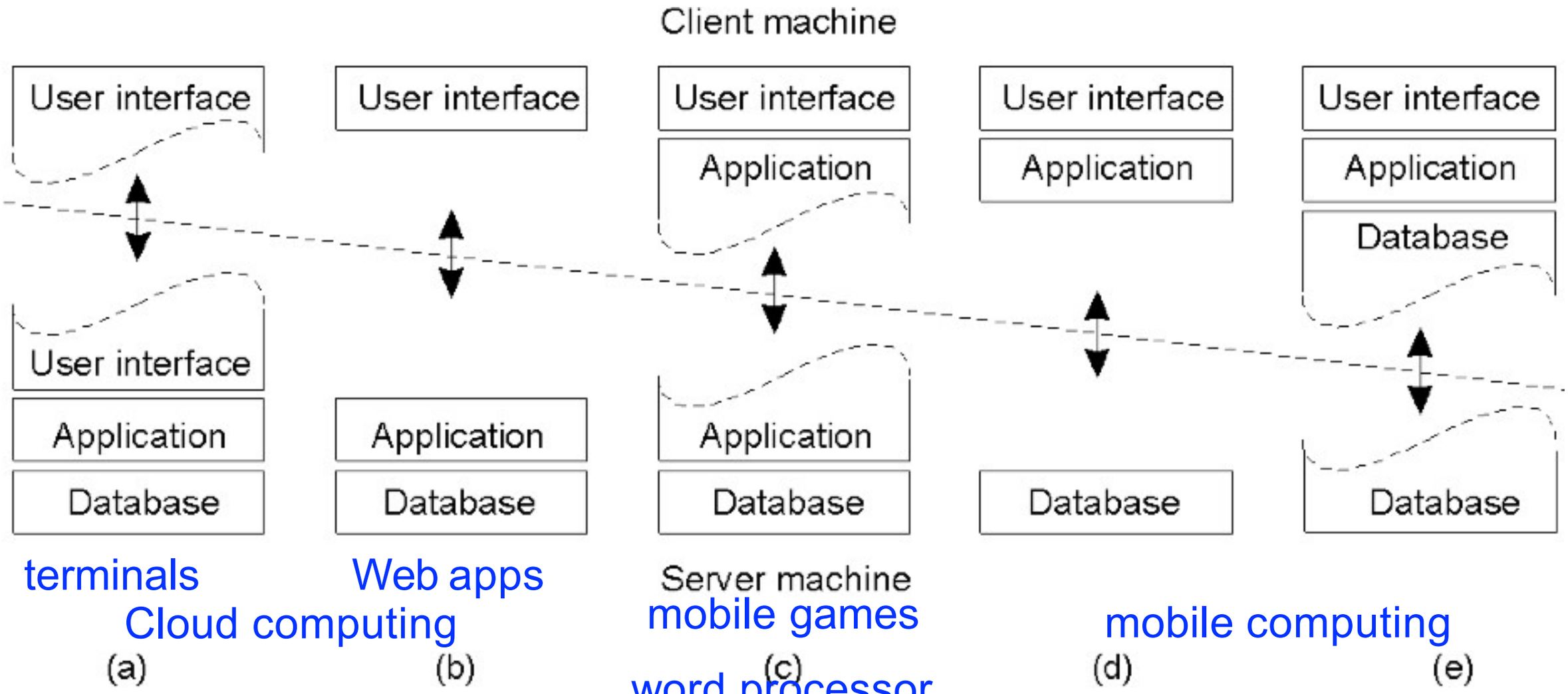
Example of Thick Client: ATM Banking System



Examples of Thin Client: Internet Banking System



Alternative Two-Tier Client Server Organizations

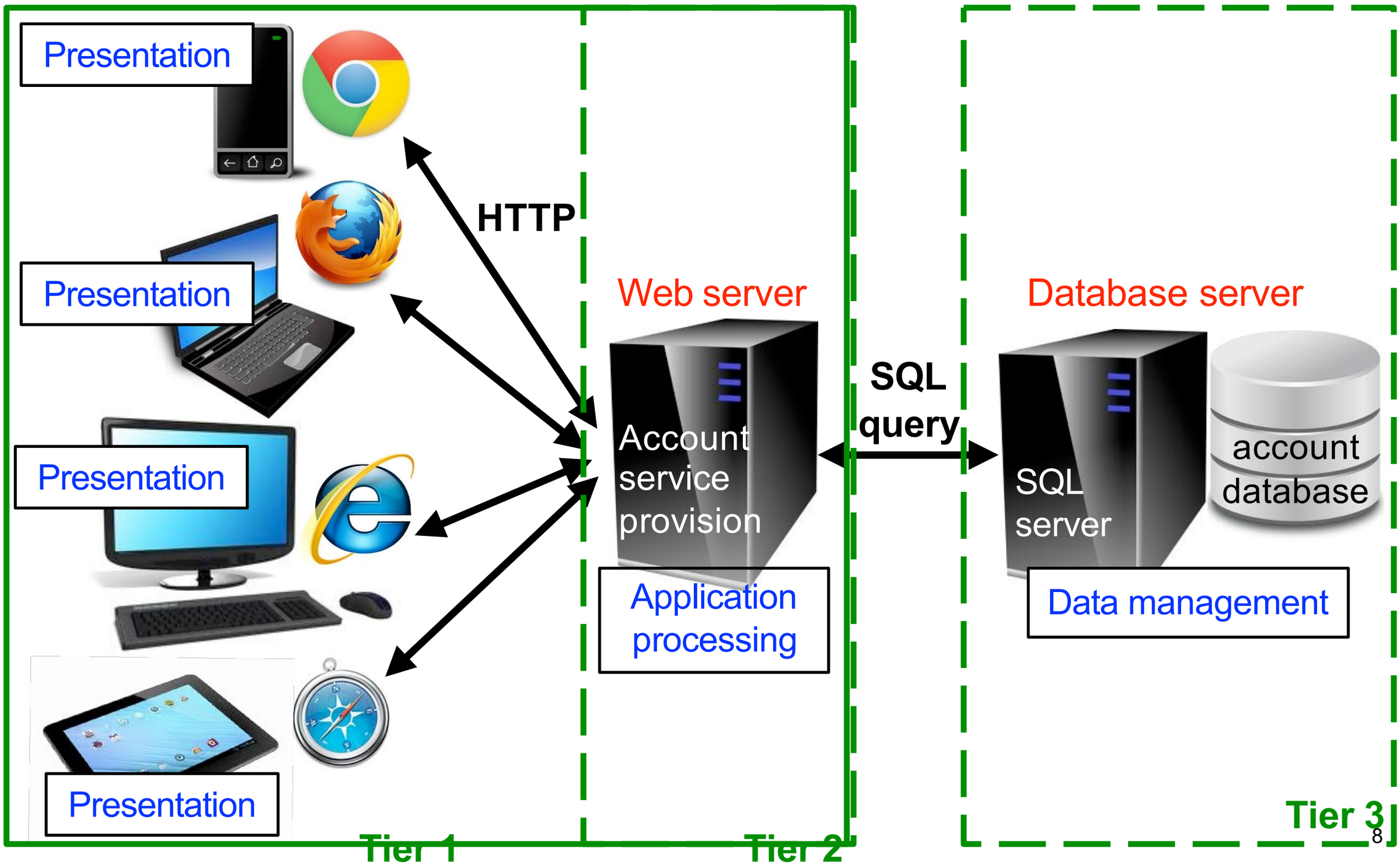


Thin Client

Hybrid Client

Thick Client

Internet Banking System... in Practice



Thin or Thick Client? Thin

pros

- Devices significantly enhanced with a plethora of networked services
- Access to legacy systems
- System management and administration
  - from admin perspective: system maintenance, security
  - from user perspective: not hassle with administrative aspects or constant upgrades
- More security
- Green IT (power saving --> cost saving)

cons

- Heavy processing load on both server and network
- Less client-perceived performance (in highly interactive graphical activities such as CAD and image processing)
- Need to be always connected

Thin or Thick Client? Thick

- Better client-perceived performance (especially, in terms of image & video processing)
- (Partly) available offline
- Distributed computing (no single point of failures)
- Devices are becoming ever faster and cheaper:
  what is the point of off-loading computation on a server when the client is amply capable of performing it without burdening the server or forcing the user to deal with network latencies?

pros

cons

- System management and related costs
- Having more functionality on the client makes client-side software more prone to errors and more dependent on the client's underlying platform

Use of Client–Server Architectural Patterns

Two-tier client-server architecture with thin clients

- Legacy system applications that are used when separating application processing and data management is impractical; clients may access these as services
- Computationally intensive applications such as compilers with little or no data management
- Data-intensive applications (browsing and querying) with non-intensive application processing (example: browsing the Web)

Two-tier client-server architecture with fat clients

- Applications where application processing is provided by off-the-shelf software (e.g., Microsoft Excel) on the client
- Applications where computationally intensive processing of data (e.g., data visualization) is required
- Mobile applications where internet connectivity cannot be guaranteed
- Some local processing using cached information from the database is therefore possible

Multi-tier client-server architecture

- Large-scale applications with hundreds or thousands of clients
- Applications where both the data and the application are volatile
- Applications where data from multiple sources are integrated

Task!

Consider a hypothetical car hire company and sketch out a three-tier solution to the provision of their underlying distributed car hire service. Use this to illustrate the benefits and drawbacks of a three-tier solution considering issues such as performance, scalability, dealing with failure and also maintaining the software over time.