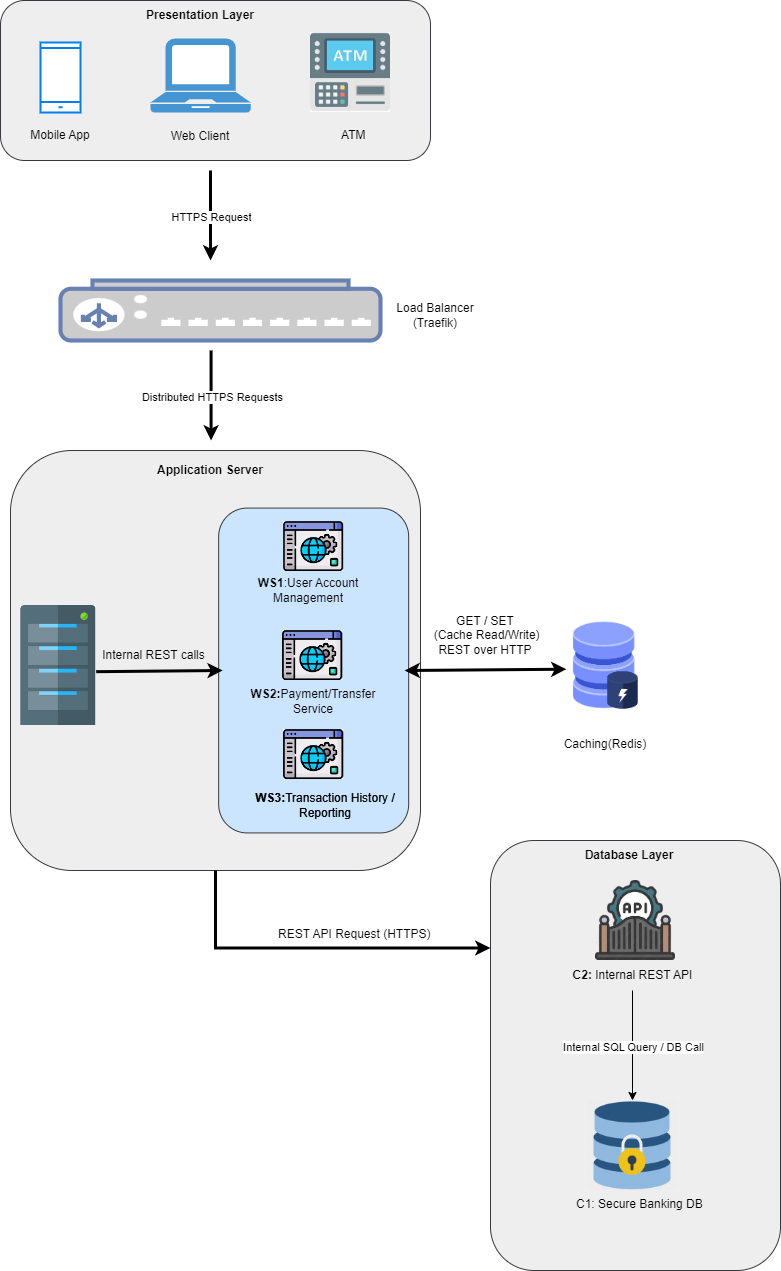
Software Architecture (40353103-1)

**Assignment #1- Graded (15%)**

Sdra Osama Awameh 202210368

# Q1: Software Architecture Design Tasks. You have to achieve the following 4 tasks with considering the applicable architecture and optimal QA

**Client-Server Based Architecture Design**

# Workflow Architecture: From Request to Reply

# 

# Module Description:

# Presentation Layer:

# 1.Web clint:

# The website user can access using a browser like chrome or firefox

# I use it because in my opinion people prefer use laptops when they need to view details and show in large screen

# 2.Mobile App:

# I used it because most people today prefer to use their mobile phones for everything, especially banking transactions.

# The mobile phone is always with the user, making it easy to check the balance, transfer money, or review recent transactions.

# I can also use notifications to alert the user if something happens to their account.

# 3.ATM:

# ATM provides direct access to basic services such as withdrawals and deposits.

# App Server:

# 1. WS1 User Account Management:

# This handles logging in creating a new account and changing your password

# in other words, everything related to the user himself.

# 2.WS2 Payment/Transfer Service:

# It handles financial transfers, such as sending money or paying a bill.

# It has security features to ensure the transaction is valid.

# 3.WS3 Transaction History / Reporting:

# This is responsible for displaying past transactions and reports.

# The user can see where they spent their money and download an account statement

# in PDF or Excel format.

# Cache:

# Cache is a place where we temporarily store information for a short period of time, so that we can quickly access it without having to ask the database again and again.

# Load Balancer:

# It decides which server should handle the user request so no single server gets too busy it help system to be faster and always available

# DB Server:

# Safe and organized place to keep all the banking data

# C1: to store sensitive data like account information, transaction logs, balances

# C2: APIs used by the App Server to talk to the database securely.

# 

# Component & Connector Communication Patterns (C&C)

# A diagram of a software flow AI-generated content may be incorrect.

# Component & Connector View – Detailed Explanation:

# Client (Mobile App, Web, ATM):

# Interface for user to access the system send HTTP requests to load balancer ,ATM use secure internal connection like (VPN).

# Load balancer ;

# use to distribute user request accept HTTPS request from clints then sent the request to App server (WS1 ,WS2 ,WS3 ).

# App server : accept REST calls from load balance rand can communication with cache and internal API(C2).

# Cache : store frequently accessed data connected internally with WS3 ,WS2 via API calls.

# C2: API gateway to access data base communication with C1(DB) using SQL accepts REST calls from App server.

# C1: store all permanent data responds to queries from C2 APIs.

# NOTE

# Why I didn’t connect WS1 to cache ?

# WS1 dont use cache directly because it deals with sensitive information and we must always get it from the source (database) to ensure security and accuracy.

# Q3: Mapping Quality Attributes to Architecture

Explain how your architecture supports key **software quality attributes (if applicable)**:

|  |  |
| --- | --- |
| **Quality Attribute** | **Architectural Feature Supporting It** |
| **Performance** | For performance I use cache to reduce database load and speed up the responses |
| **Scalability** | I use load balancer to handle multiple requests if we have more than one request in same time by distribute traffic across multiple services |
| **Availability** | If one of services fail load balancer can redirect to healthy services and system can run normally and available for use any time |
| **Security** | I use HTTPS / secure DB access and no caching for sensitive data like account info |
| **Modifiability** | (WS1,WS2,WS3) each service is independent so I can update any one without break others |
| **Maintainability** | When I design ACH I use models and components and it have clear responsibilities clear communication so developers can fix or update any parts without affecting the whole system |
| **Cost Efficiency** | Use cache to reduce the number of queries that go to the database.  Load Balancer:Distributes requests to servers as needed  I can start with one server, and if the load increases, I can add more. |