Assignment #2

Online Banking System / Digital Wallet

1. extended architecture design

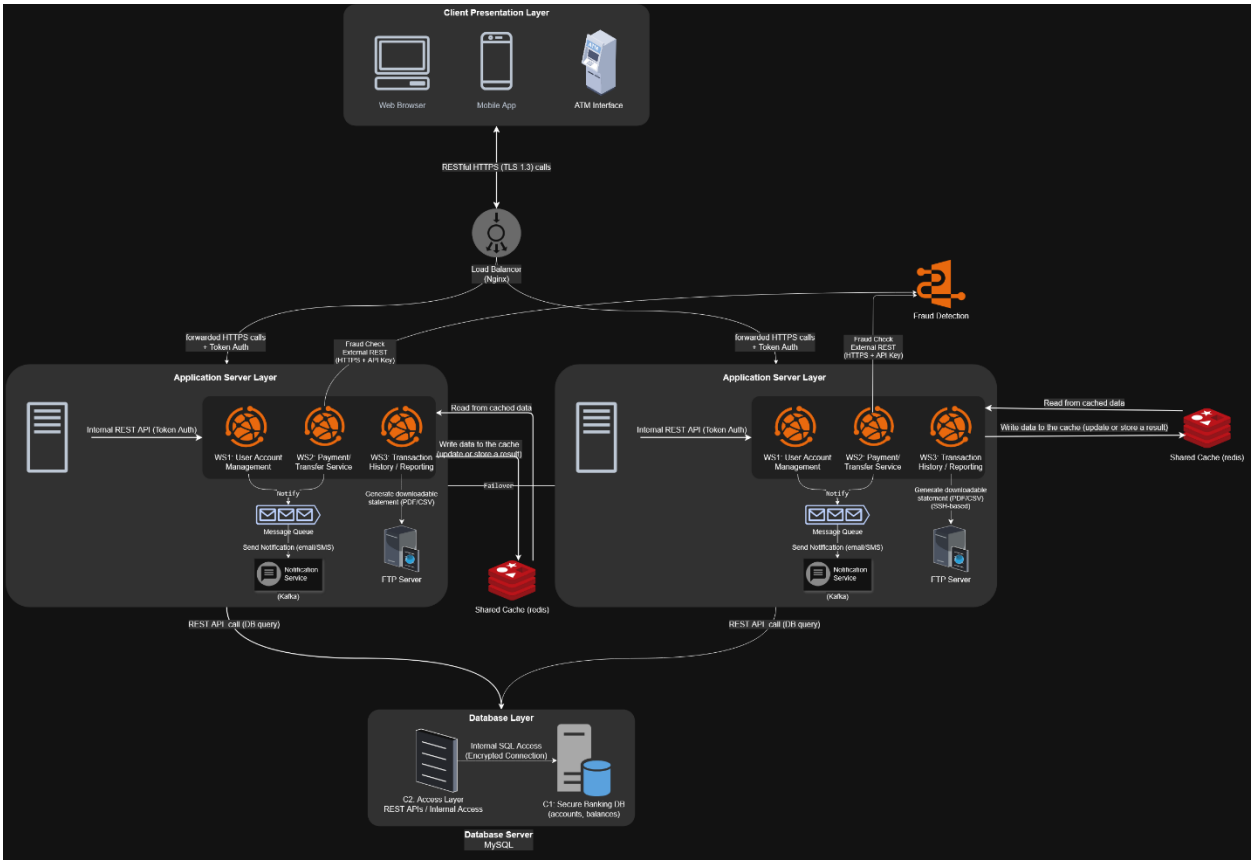
new components added

component	description
fraud detection api	external api that checks if a transaction looks suspicious before approval.
message queue	sends events async between services (I used kafka in my diagrams)
notification service	listens to events and sends alerts to users.
redundant app servers	backup app servers to keep system running if one goes down.

interactions and flow

- ws2 checks with fraud detection before sending to db.
- if approved, it continues and saves to db.
- after that, ws2 and ws3 send event messages.
- the notification service listens and sends the alert.
- the load balancer decides which app server to send the request to.

Client-Server Architecture



---

2. architectural pattern analysis

microservices architecture

- scalability: every service can scale alone depending on load.
- availability: if one fails the rest still work.
- security: different rules for different services.

layered architecture

- scalability: not that flexible, u gotta scale whole layers.
- availability: if a layer fails, it may affect all services.
- security: it's easier to control everything in one place.

I recommend to go with microservices, it's more flexible and modern for this system.

---

3. quality attribute analysis

quality attr.	metric	strategy	tradeoffs
performance	keep api response < 200ms	use cache, parallel fraud check	memory usage may increase
scalability	handle 10x traffic	scale components horizontally	more infrastucture cost
reliability	99.9% uptime	add backup servers + health checks	bit more system complexity
security	full HTTPS + token auth	tls 1.3, api keys, jwt, validations	slight latency but worth it

---

4. Comprehensive Documentation

component and connector documentation

diagrams

- *client-server architecture*
- *workflow architecture*
- *communication & protocol patterns*

diagram: workflow architecture

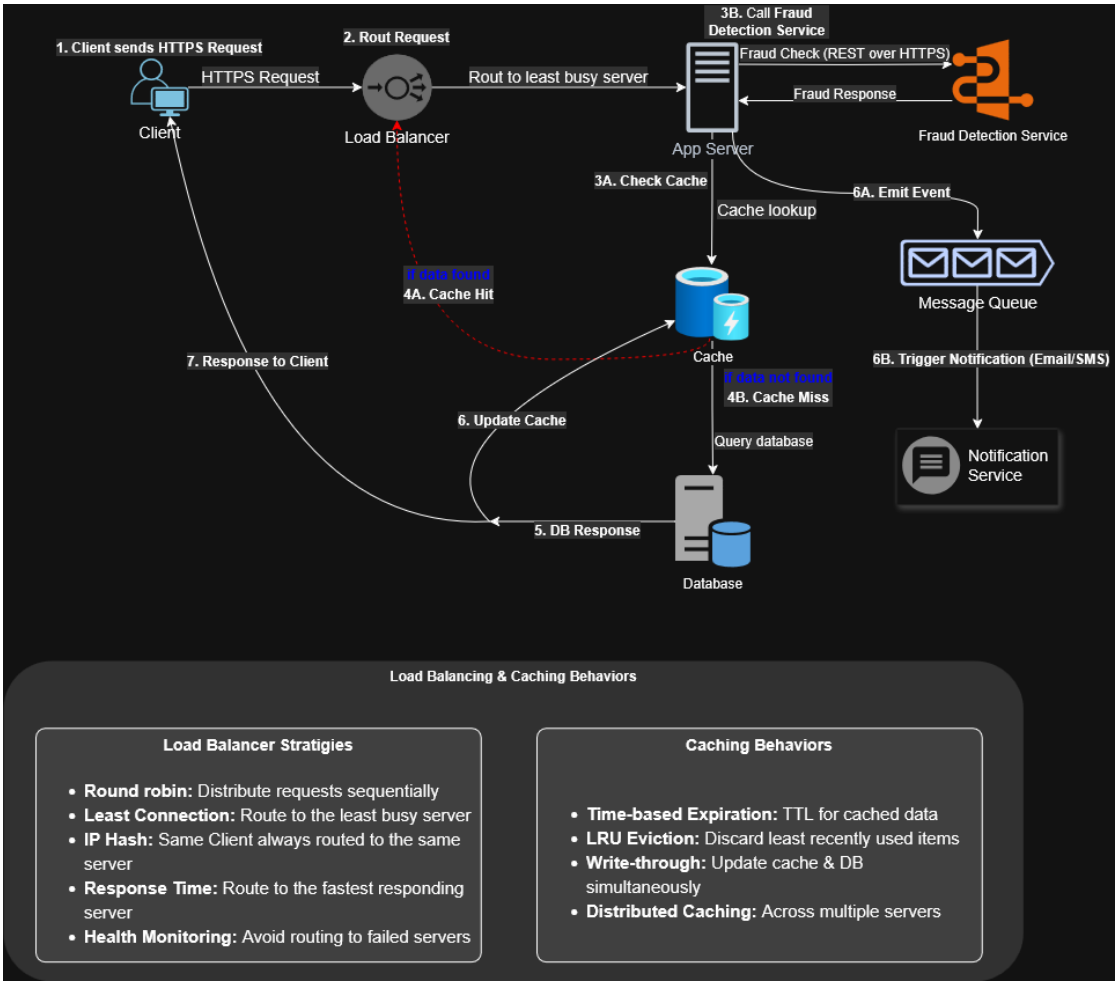
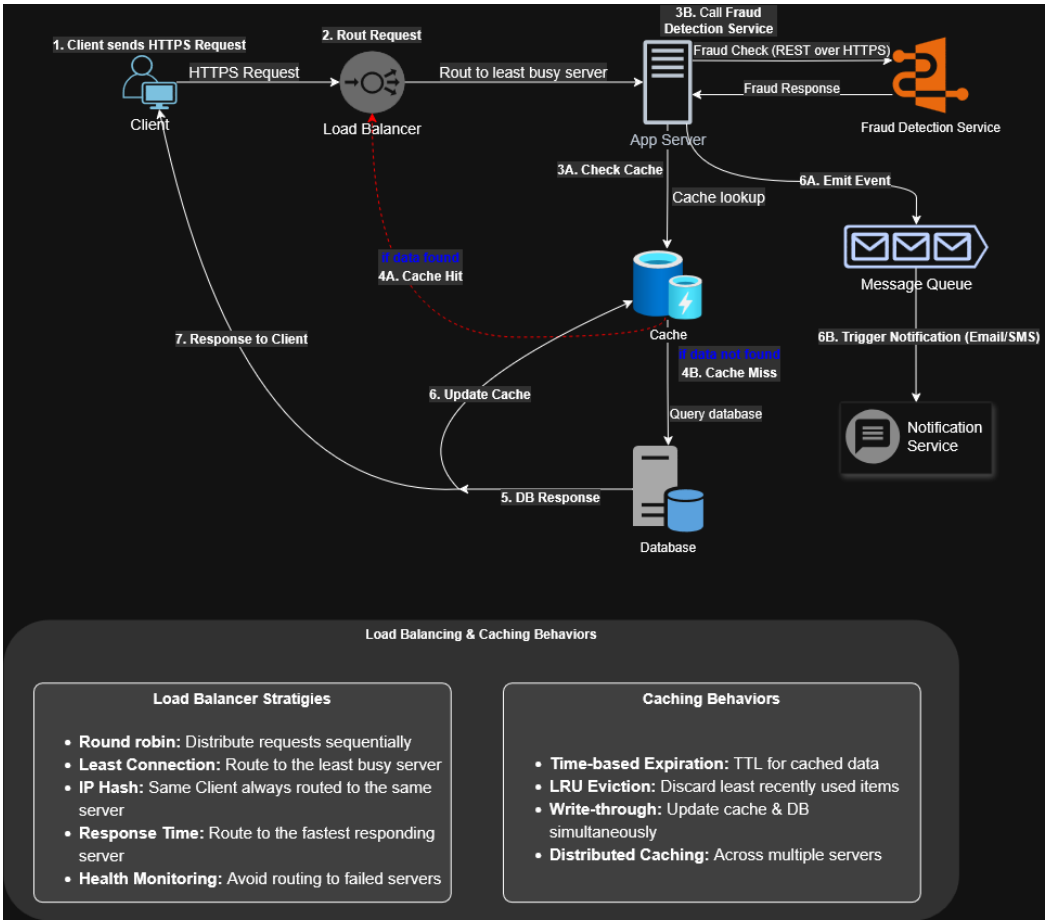


diagram: communication patterns



New Components Description

fraud detection API

- checks if transaction is fake or fishy
- uses https with api key, synchronous

notification service

- listens to events from queue
- triggers email or sms alerts

event bus / message queue

- helps services talk async (I used kafka)
- lets system scale without crashing

redundant app servers

- if one app server fails, the other still works
- both are behind the load balancer

---

connector protocols

from → to	protocol / security	sync?	description
client → load balancer	https (tls 1.3)	sync	secure client request
app server → ws*	rest + jwt	sync	secure internal calls
ws2 → fraud api	https + api key	sync	fraud check external
ws2 → event bus	kafka / amqp	async	emits transaction completed event
event bus → notification	internal dispatch	async	sends alert when event is received