COS730 Assignment 2

MUKWEVHO MULWELI PATIENCE

Student number:23877716

May 5, 2023

Contents

1 Introduction

PayPal is a popular secure platform that facilitates payments in different currencies between individuals and businesses,which is primarily used in the financial technology industry by business owners,buyers,sellers,senders ,receivers and other professionals who need to make transactions in different currencies.PayPal system has more than 435 million users ,so it needs system architecture that is made to be scalable, flexible, maintainable and reliable.The purpose of this document is to provide a thorough overview of the system architecture and design, architectural patterns, how the system interact and architectural design standard for the PayPal system application and website.

2 Non-functional requirements

2.1 Quality requirement

Security:

The platform should provide secure authentication, data encryption, and protection against unauthorized access or malicious attacks on users bank credentials.

Performance:

The platform should provide fast and secure transactions.

Reliability:

The platform should be highly reliable,fault tolerance and available at all times, without any downtime or service interruptions.

Scalability:

As the user base expands, the system should be able to support a rising number of users while maintaining performance.

Usability:

The platform should be easy to use and navigate for users of all technical abilities, with an intuitive interface and clear instructions.

Compatibility:

The platform should be compatible with a wide range of devices, operating systems, and web browsers, to ensure that users can access it from anywhere, at any time.

Accessibility:

The platform should be designed to be accessible to users with disabilities, including those who are visually impaired or have limited mobility, is where we can think of voice recognizing to those tho can not type.

Internationalization:

The platform should support multiple languages, currencies, and time zones, to cater to users from different regions of the world.

Performance efficiency:

The system should be able to securely transfer money electronically between people or organizations.

Maintainability:

The platform should be easy to maintain, upgrade, and modify, to ensure that it remains uptodate and competitive in a rapidly evolving market.

Availability:

The system must run continuously, without interruption for repairs or updates.

2.2 Quantification of the quality requirements

Security:

To ensure that data is kept private and secure, the system should use secure protocols. Any unwanted access attempts should be able to be found and stopped by the system. The system must follow generally accepted security best practices.

Performance:

The platform should provide fast and secure transactions.

Reliability:

The platform should be highly reliable,fault tolerance and available at all times, without any downtime or service interruptions.

Scalability:

The system should use secure protocols to ensure that data is kept confidential and secure. As the user base expands, the system should be able to support a rising number of users while maintaining performance.

Usability:

The system should have benchmark for high customer satisfaction which is around 90 percent or higher.The system should then be put to the test with both technical and non-technical users to determine its usability. For new users, the system's learning curve should be no longer than 10 minutes.

Compatibility:

The platform should be compatible with a wide range of devices, operating systems, and web browsers.

Accessibility:

The system should be accessible to a wide web browsers, making it accessible to a wide audience compatible with a wide range of devices, operating systems, and web browsers, to ensure that users can access it from anywhere, at any time.

Internationalization:

The system should support multiple languages, currencies, and time zones, to cater to users from different regions of the world.

Performance efficiency:

The platform should use system resources efficiently and minimize the amount of data and processing required to deliver its services.

Maintainability:

For ease of maintenance and updating, the system should be designed with modularity idea.PayPal typically schedules maintenance activities during off-peak hours to minimize the impact on users.

Availability

The system must be fault tolerant and capable of quickly recovering from any malfunction

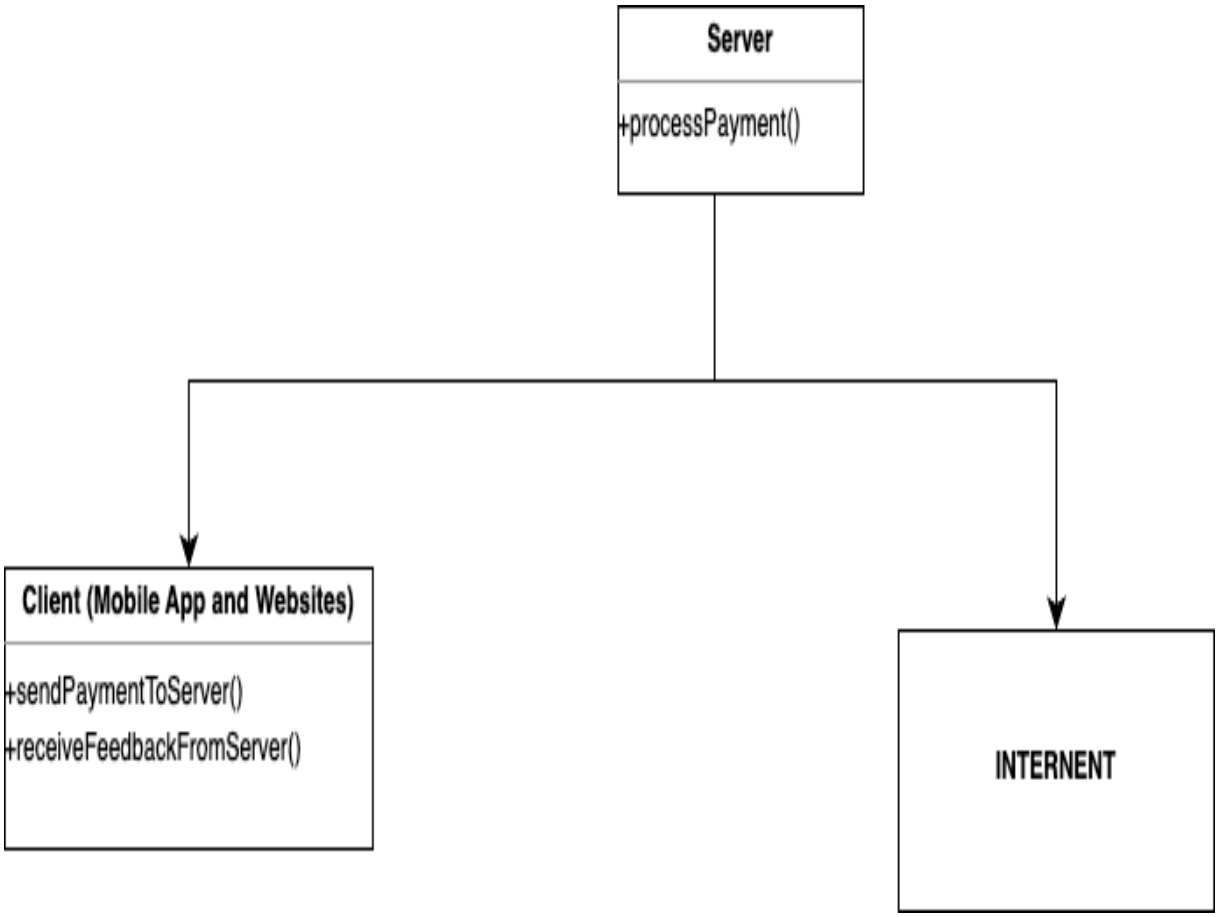
3 Architectural design

The following elements serve as the foundation for the architectural layout of the PayPal system application:

3.1 Architectural patterns

Client-Server Architecture:

The diagram demonstrates the two ways the mobile application and websites can communicate with the server: sendPaymentToServer() and receivepayment-FromServer(), which gets the server's response.



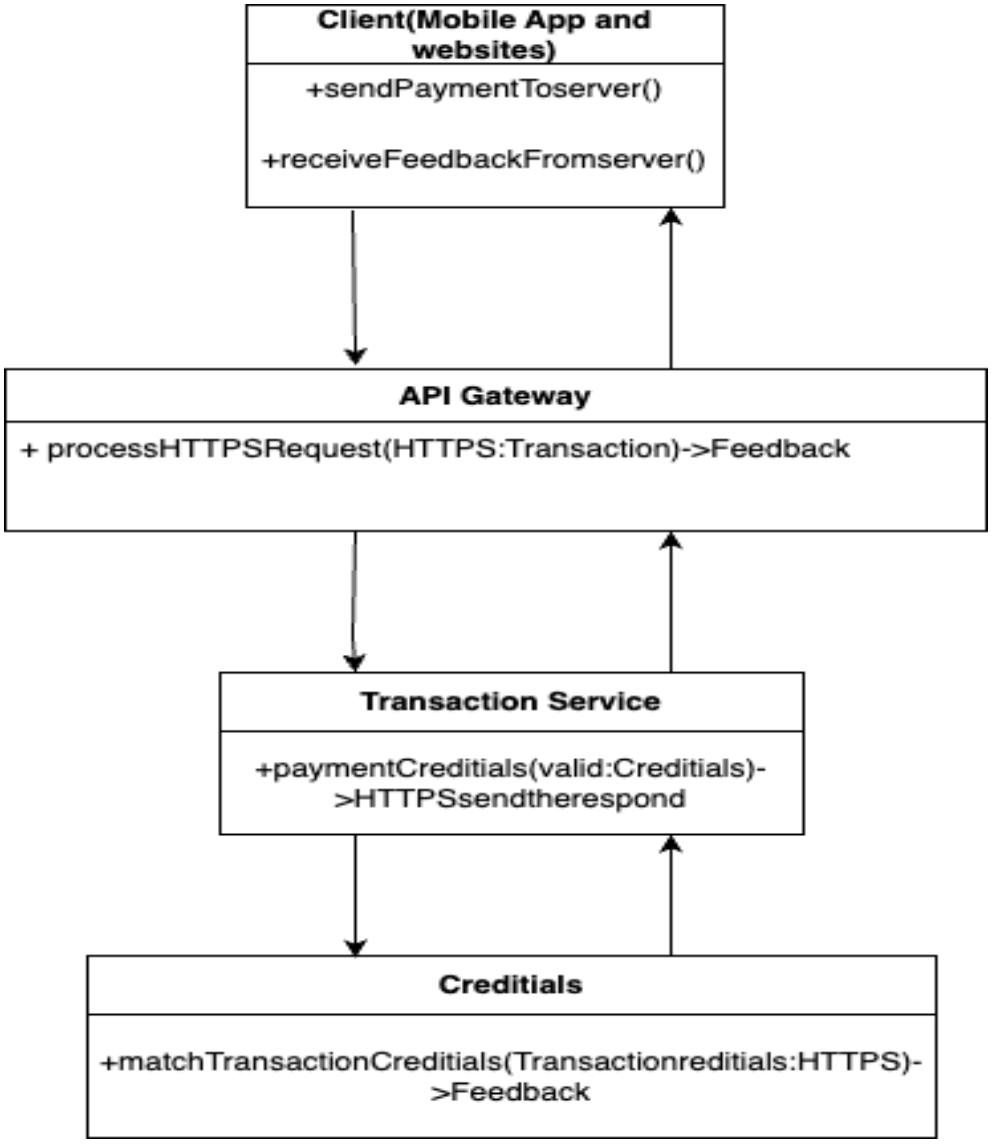
-sendPaymentToServer() method:

When a client application, like a web browser or mobile app, starts a transaction with PayPal, it sends an HTTPS request to PayPal's servers. Information regarding the transaction, including the amount, receiver, and method of payment, is included in the request.

-receiveFeedbackFromServer()method:

After receiving the request, PayPal's servers authenticate the client device by employing a variety of security steps, including confirming digital certificates and encrypting important information. The transaction is processed by PayPal's servers after the client device has been authorized, and a response is then sent back to the client device over HTTPS.

Microservices Architecture:



This diagram demonstrates the microservices architecture used by the PayPal

system ,where each system component is divided into smaller, independent services. These are the services:

-Client:

This is the user-facing portion of the program that communicates with the system by sending HTTPS protocol requests and receiving responses from the system.

-API Gateway:

This part is in charge of taking client requests and directing them to the proper microservice.

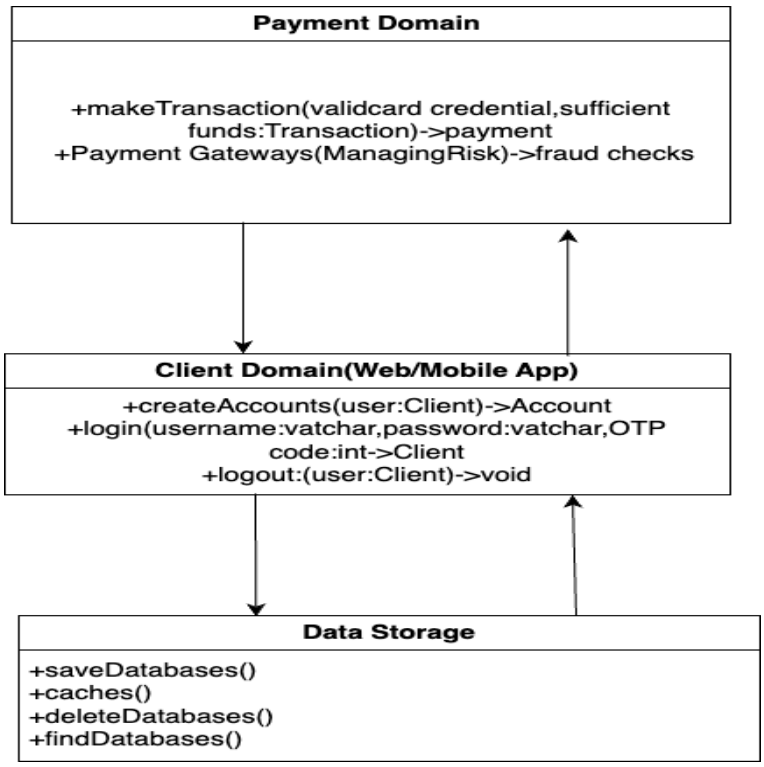
-Transaction Service:

This service is in charge of processing transactions by validating bank credits.

-Creditials Service:

This service's job is to match HTTPS requests with credit cards.

Domain-Driven Design (DDD) Architecture:



-Payment Domain:

This domain is responsible is responsible for processing and managing payments which includes facilitating transactions between buyers and sellers, authorizing and verifying payments,protecting buyers and sellers from fraud and Providing financial reporting.

-Client Domain:

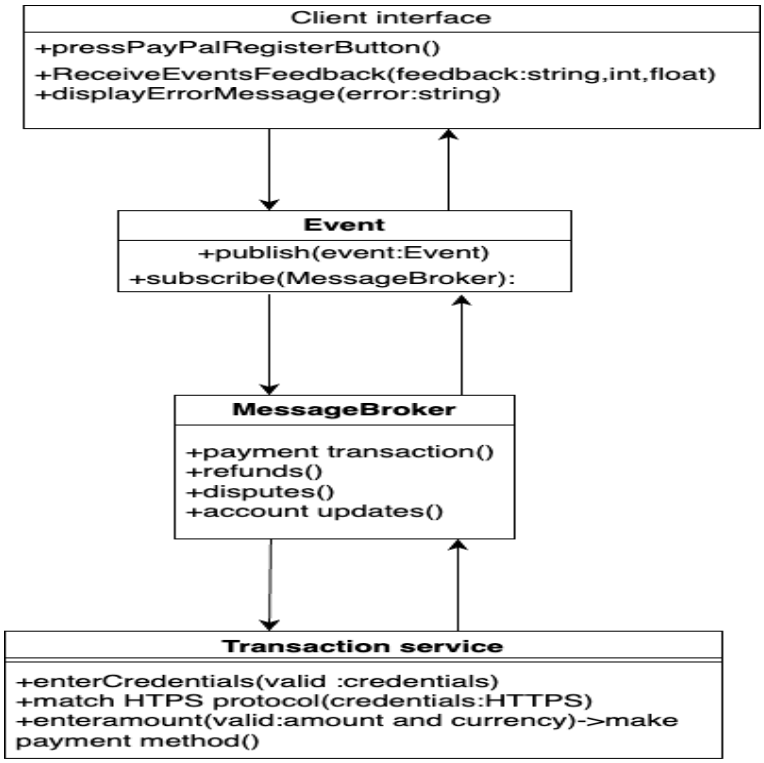
This domain is responsible for managing user accounts and authentication.

-Data Storage:

This service is responsible for managing and storing the vast amount of data generated by the platform such saving,delete data and managing data backup and caches. The diagram shows that the Payment Domain is responsible for pro-

cessing and managing payments which includes facilitating transactions between buyers and sellers,which allows good interacting between buyers and sellers.The Client Domain provides methods for creating user accounts, logging in, and logging out. The Data Storage service is responsible for saving, deleting, and finding entities in the database.

Event-Driven Architecture(EDA):



This diagram illustrates that the PayPal system uses an event-driven architecture,where events trigger certain actions within the system. The components are as follows:

-Client Interface:

This is the portion of the application that users see and interact with, displaying outcomes and error warnings while enabling them to subscribe for content by clicking the PayPal register button.

-Event :

The proper handler is triggered when an event is published, and this component is also in charge of subscribing to events.

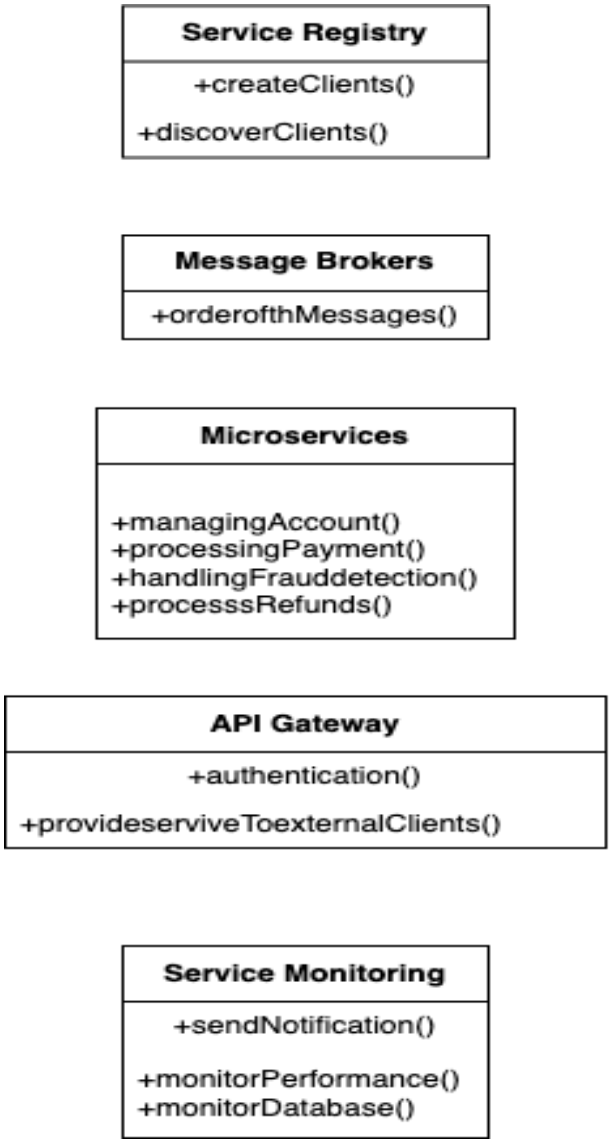
-Message Broker:

The message broker receives events produced by different sources, including payment transactions, refunds, disputes, and account updates.

-Transaction Service:

This service is responsible for processing the payments and matching the Hypertext Transfer Protocol Secure(HTTPS) protocol with correct and valid card credentials. The diagram demonstrates how pressing the PayPal register button causes an event to be published to the Event from MessageBroker, which receives events created by various sources such as payment transactions, refunds, disputes, and account modifications. The process for validating credentials is started by the Transaction Service, which has subscribed to this event. A specific amount in a supported currency can then be processed once the credentials have been verified. This event is also subscribed to by the Transaction Service, which initiates the match Hypertext Transfer Protocol Secure(HTTPS) protocol to match particular amount requests with valid card credentials.Last but not least, the Transaction Service publishes a result event to the Event, which activates the proper handler in the Client Interface to display the result or error message.With a more loosely linked system and greater extensibility due to the event-driven architecture, the system can be improved upon. Furthermore, since events may be simulated and verified separately, testing and debugging are made simpler.

Service-Oriented Architecture:



This diagram demonstrates the service-oriented architecture used by the PayPal system, in which many services interact with one another to provide the required functionality. The parts consist of the following:

-Service Registry:

This service is responsible for helping different services to discover and communicate with each other, it isalso in charge of generating and locating users.

-Message Brokers:

This service makes sure that messages in PayPal system are delivered reliably and can be processed in the correct order.

-Microservices:

Each microservice within this service has a specific function, such as managing accounts, processing payments, or handling fraud detection.

-API Gateway:

The role of API Gateway is to expose its services to external clients while providing services such as authentication, rate limiting, and API documentation.

-Service Monitoring:

This service is responsible for monitoring the PayPal system performance using tools such as metrics collectors, log analyzers, and distributed tracing systems and also plays a role in sending notifications.

3.2 Architectural Styles

Architectural Styles for Performance:

Concurrency:

PayPal is a large-scale payment system that processes a large volume of transactions every day. To handle this scale, PayPal uses concurrency techniques to maximize performance and efficiency. This could be achieved through the use of multi-threading,connection pooling, distributed systems and caching. or parallel processing techniques.

Batch Processing:

PayPal can use batch processing to improve its efficiency and reduce the processing time of certain operations. This would allow the system to process multiple requests at once, improving overall efficiency especially a batch of transactions.

Scheduling:

PayPal can use scheduling to improve its efficiency and automate repetitive tasks such as Batch processing: PayPal can use scheduling to automate the batch processing of transactions, data processing, and report generation. This would help to reduce system overload and improve overall performance.

Caching:

PayPal can use caching to improve its performance and reduce the response time of frequently accessed data such as tore user session data, such as user credentials, shopping cart data, and payment information. This would allow the system to access the data more quickly, reducing processing time and improving overall performance.

Load balancing:

PayPal can use load balancing to improve its performance and ensure high availability by distributing the incoming traffic across multiple servers or resources.This legacy software may also use load balancing to distribute incoming requests across multiple servers, improving the system's ability to handle high traffic volumes and improve response times.

Optimization of database queries:

Ensuring that database queries are optimized is critical for improving the performance of a PayPal system. Indexing, caching, and other database optimizations can help speed up queries and reduce the time it takes to retrieve data.

Use of Content Delivery Network(CDN):

A CDN can help improve the performance of a PayPal system by caching and serving static content from servers located closer to the end user, reducing the time it takes to load content.

Monitoring and optimization:

Regular monitoring of system performance and optimization of bottlenecks can help identify and fix issues that can affect system performance.

in-memory technologies:

In-memory technologies such as Redis and Memcached can help improve performance of PayPal by reducing the time it takes to access and retrieve data.

Architectural Styles for Security:

Authentication:

PayPal could use authentication to ensure that only authorized users are able to access the system. This could involve the use of username and password authentication, two-factor authentication, or other authentication methods.

Multi-factor authentication:

Multi-factor authentication adds an additional layer of security by requiring users to provide more than one form of identification before accessing their account. This can prevent unauthorized access, even if a user's password is compromised.

Encryption:

Encryption can be used to protect sensitive data such as credit card numbers and other financial information. Encryption can prevent data from being intercepted or viewed by unauthorized users.

Regular updates and patches:

Regular updates and patches can help fix security vulnerabilities and ensure that the system is up-to-date with the latest security measures.

Access controls:

Access controls can be used to limit access to sensitive data and features within the system. This can prevent unauthorized users from accessing sensitive information or performing unauthorized actions.

Monitoring and logging:

Regular monitoring and logging of system activity can help detect and respond to security threats. This can include monitoring for unusual login activity, access attempts, and suspicious behavior.

Regular security assessments and audits:

Regular security assessments and audits can help identify potential vulnerabilities and areas for improvement in the system's security.

Use of firewalls and intrusion detection/prevention systems:

Firewalls and intrusion detection/prevention systems can be used to monitor network traffic and detect and prevent unauthorized access.

3.3 Architectural constraints

Security:

Security is critical for a PayPal system to protect sensitive information such as financial and personal data. The system must comply with security standards and regulations such as PCI DSS, GDPR, and other data protection laws.

Scalability:

The PayPal system must be designed to handle a large volume of transactions and users. The system must be able to scale up or down to meet demand, handle peak loads, and ensure high availability.

Performance:

The system must be designed to provide fast and responsive user experiences. The system must be optimized for performance and be able to handle large volumes of data without impacting performance.

Availability:

The system must be highly available, with minimal downtime or service interruptions. The system must have redundancy and failover mechanisms in place to ensure continuity of service.

Compliance:

The PayPal system must comply with various regulatory and legal requirements, such as anti-money laundering laws, tax regulations, and financial regulations.

Interoperability:

The PayPal system must be able to integrate with other systems, applications, and services. The system must use standard protocols and APIs to ensure interoperability and data exchange.

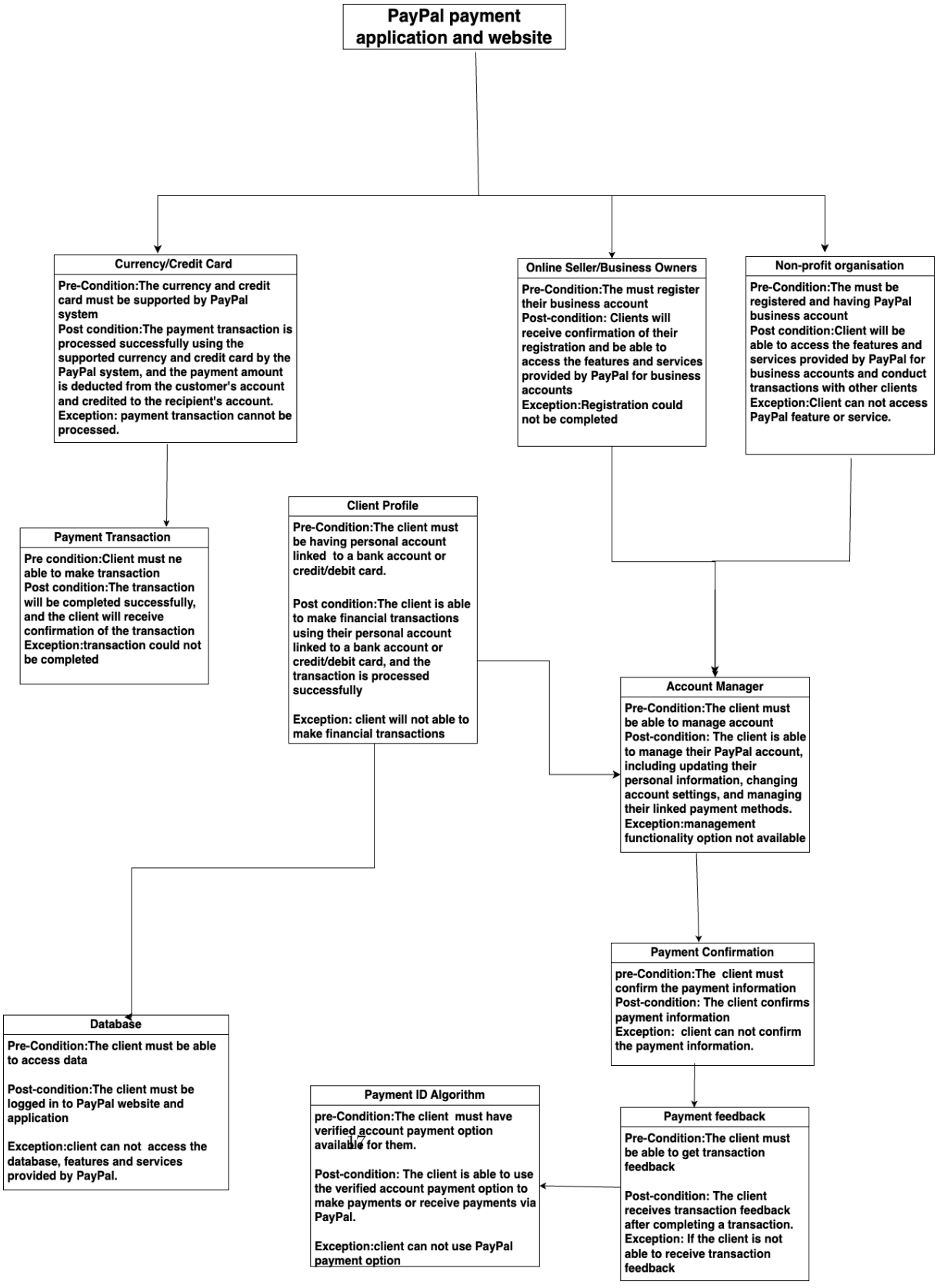
Maintainability:

The system must be easy to maintain, update, and manage. The system must have a well-defined architecture, modular design, and clear documentation to facilitate maintenance and upgrades.

Cost-effectiveness:

The PayPal system must be designed to be cost-effective, with efficient use of resources and infrastructure. The system must balance performance, scalability, and availability with cost considerations.

3.4 Actor-system interaction



3.5 Deployment model

The PayPay system is deployed on cloud-based infrastructure using the following:

Configuration Management Tools:

Configuration management tools like Ansible, Chef, and Puppet are commonly used to automate the deployment and configuration of software and infrastructure.

Containerization Technologies:

Containerization technologies like Docker and Kubernetes can be used to package and deploy software applications in a lightweight, portable, and scalable way.

Continuous Integration and Continuous Deployment (CI/CD) Tools:

CI/CD tools like Jenkins, GitLab CI/CD, and CircleCI are used to automate the building, testing, and deployment of software applications.

Cloud Services:

Cloud services like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) are commonly used to deploy and manage applications in the cloud.

Monitoring and Logging Tools:

Monitoring and logging tools like New Relic, Splunk, and ELK Stack are used to monitor the health and performance of applications and infrastructure, and to troubleshoot issues as they arise.

Terraform:

Terraform is an infrastructure-as-code tool that can be used to manage and provision infrastructure resources, such as virtual machines, cloud services, and networking components.

HashiCorp Vault:

HashiCorp Vault is a tool for managing secrets and sensitive data, such as passwords, API keys, and other credentials. It can be used to securely store and manage access to secrets used by the PayPal system.

Apache Kafka:

Apache Kafka is a distributed streaming platform that can be used to handle large volumes of data and events. It can be used as a messaging system for communication between different components of the PayPal system.

Redis:

Redis is an in-memory data store that can be used for caching and as a message broker. It can be used to improve the performance and scalability of the PayPal system.

NGINX:

NGINX is a web server and reverse proxy that can be used to improve the performance and security of web applications. It can be used to manage incoming traffic to the PayPal system and to load balance requests between different servers.

3.6 Technical requirements(technology)

PayPal has a complex technical architecture that includes multiple technologies to support its operations.The followings are technical requirements and technologies used by PayPal:

Programming Languages:

Java, JavaScript, Python, and Go are programming languages used in PayPal system to create its software applications.

Web Development:

PayPal uses web development technologies such as HTML, CSS, and JavaScript to create its web-based interfaces.

Databases:

PayPal uses both SQL and NoSQL databases such as MySQL, Oracle, and Cassandra to store and manage its data.

Cloud Computing:

PayPal's system runs on cloud computing infrastructure provided by Amazon Web Services (AWS) and Google Cloud Platform (GCP) for its hosting and computing needs.

Microservices:

PayPal uses a microservices architecture to create smaller, more manageable applications that can be deployed and scaled independently.

Application Programming Interfaces(APIs):

PayPal provides APIs (Application Programming Interfaces) to enable developers to integrate with the platform, allowing for customization and third-party integrations.

Machine Learning:

PayPal uses machine learning algorithms to detect and prevent fraudulent transactions, improve customer experience, and provide data-driven insights.

Mobile Development:

PayPal has mobile applications for both iOS and Android platforms, built using native technologies such as Swift and Kotlin.