User: ok
AI: Hello, I'm your lecturer for this session on system architecture designs. Today, we'll be critically analysing system architecture designs from different domains and providing constructive feedback on them. You'll be taking on the role of a system architecture consultant, reviewing these designs and providing your expert recommendations.

The case studies we'll be discussing are:

1) Uber
2) Netflix
3) Microsoft Office
4) Microsoft Word

Please choose one of these case studies to begin with.
User: 2
AI: Netflix is a global streaming service that offers a variety of TV series, documentaries, and feature films across a wide variety of genres and languages. It's available on a wide range of devices, including smart TVs, game consoles, streaming media players, set-top boxes, smartphones, and tablets.

Netflix's system architecture is designed to support its vast global network, enabling millions of users to stream content simultaneously. This architecture is complex, robust, and highly scalable, ensuring that the service remains reliable and efficient, regardless of the number of users or the amount of data being processed at any given time.

Given this brief overview, what sort of architecture do you think Netflix uses?
User: microservices and cloud
AI: That's correct! Netflix utilises a microservices architecture, which allows for independent deployment, scalability, and maintenance of different components. Each microservice handles a specific function, such as user authentication, content recommendations, and video streaming.

Netflix was also an early adopter of cloud infrastructure and relies heavily on Amazon Web Services (AWS) for a wide range of services, including computing, storage, and content delivery network services.

Now, let's delve a bit deeper. Can you think of how the use of a global content delivery network (CDN) might benefit Netflix's service?
User: faster streaming
AI: Absolutely, the global CDN plays a significant role in ensuring low-latency streaming. By caching and delivering content closer to the users, it allows Netflix to handle massive volumes of traffic and provide high-quality video streaming to users around the world. This is particularly important for a streaming service like Netflix, where user experience is highly dependent on the speed and quality of the video stream.

Now, let's consider the data management aspect. Netflix processes and stores vast amounts of data. Can you think of why they might use a combination of relational databases, NoSQL databases, and data processing frameworks?
User: relational data to allow users to keep track of their watch history, payments etc while noSQL databases to store movies and data processing frameworks to provide realtime analysis

AI: That's a great analysis! Indeed, relational databases are well-suited for managing structured data and handling complex queries, which makes them ideal for managing user data, including watch history and payments.

On the other hand, NoSQL databases are highly scalable and can handle large volumes of data, making them suitable for storing movie data. They also offer flexibility in terms of the data model, which can be beneficial when dealing with diverse content metadata.

Data processing frameworks are used to process large amounts of data in real-time or near-real-time, which is crucial for Netflix's content delivery optimization and recommendation algorithms.

Now, let's move on to the custom technologies developed by Netflix. Can you identify how these technologies contribute to the overall system architecture?
User: what are the custom technologies developd by netflix?
AI: Netflix has developed several in-house technologies to manage its infrastructure, including:

1. Zuul: An edge service proxy that provides dynamic routing, monitoring, resiliency, security, and additional edge functionalities.
2. Eureka: A service registry for resilient mid-tier load balancing and failover.
3. Ribbon: A client-side load balancer that gives you a lot of control over the behaviour of HTTP and TCP clients.
4. Hystrix: A library that provides latency and fault tolerance for distributed systems.
5. Simian Army: A collection of tools that test the resiliency of the system, such as Chaos Monkey, which randomly terminates instances to ensure the system can handle failures.

These technologies play a crucial role in ensuring the robustness, scalability, and reliability of Netflix's system architecture. Can you think of how these technologies might interact with each other and contribute to the overall system?
User: interact using microservice and event driven to provide seamless user experience and streaming
AI: That's a good start! Indeed, these custom technologies are designed to work together in a microservices architecture, each serving a specific function.

Zuul acts as the front door for all requests, handling dynamic routing, security, and other edge functionalities. Eureka and Ribbon work together to handle load balancing and failover, ensuring that the system can distribute load efficiently and recover from failures. Hystrix provides fault tolerance, helping the system to gracefully handle failures and maintain performance.

The Simian Army, including Chaos Monkey, is used to regularly test the resilience of the system. By randomly terminating instances, it ensures that the system can handle unexpected failures without impacting the user experience.

Now let's look at the strengths and weaknesses of this approach. What do you think they might be?
User: high complexity and high maintenance cost for streaming
AI: You're on the right track. Indeed, one of the main challenges of a microservices architecture like Netflix's is its complexity. Managing and coordinating numerous independent services can be complex and requires a robust infrastructure and skilled

engineering team.

On the other hand, microservices offer numerous benefits, including scalability, flexibility, and resilience. Each service can be scaled independently based on demand, which is crucial for a global streaming service like Netflix. If one service fails, it doesn't bring down the entire system, ensuring a consistent user experience.

The use of a global CDN and cloud infrastructure also adds to the cost. However, these are essential for delivering a fast, reliable streaming service to users worldwide.

In terms of maintenance, while there may be higher upfront costs associated with setting up and managing the microservices and cloud infrastructure, these can be offset by the benefits in terms of scalability, performance, and user experience.

Now, let's move on to the next case study. Please choose from the remaining options:

1) Uber
3) Microsoft Office
4) Microsoft Word
User: 1
AI: Uber is a multinational ride-hailing company offering services that include peer-to-peer ridesharing, ride service hailing, food delivery (Uber Eats), and a micro-mobility system with electric bikes and scooters. The company's platform can be accessed via its websites and mobile apps.

Uber's system architecture is designed to handle a large number of concurrent users, process a high volume of transactions, and provide real-time updates to users and drivers. This architecture is crucial to Uber's ability to deliver a reliable and efficient service, regardless of the number of users or the amount of data being processed at any given time.

Based on this brief overview, what sort of architecture do you think Uber uses?
User: cloud and gps, and microservice
AI: That's correct! Uber employs a microservices architecture, where each service is designed to perform a specific function. This allows for independent deployment, scaling, and maintenance of different components. Some of the key microservices include User Services, Driver Services, and Trip Services.

Uber also relies heavily on cloud services for scalability and reliability. They use a combination of proprietary and open-source technologies, including containerization and serverless computing.

The use of GPS (Global Positioning System) is crucial for Uber's real-time location tracking and routing services. This is an integral part of Uber's Maps and Routing Services.

Now, let's delve a bit deeper. Can you think of how Uber's real-time data processing might be beneficial for its service?
User: users are able to track where their things or rides are with ease
AI: Absolutely, real-time data processing is crucial for Uber's operations. It allows the system to provide real-time updates to both drivers and riders about their trip status. This includes tracking the driver's location, estimating arrival times, and calculating the best routes based on current traffic conditions.

Moreover, real-time data processing also supports dynamic pricing (surge pricing) by analysing supply (number of drivers) and demand (number of riders) in real-time. It also aids in detecting fraudulent activities promptly, enhancing the security of the platform.

Now, let's consider the backend infrastructure. Uber uses a combination of proprietary and open-source technologies. Can you identify how these technologies contribute to the overall system architecture?
User: can you explain?
AI: Of course! Uber's backend infrastructure is built on a combination of proprietary and open-source technologies. They use cloud services for scalability and reliability, which is crucial for handling the high volume of transactions and concurrent users.

Key components of Uber's backend infrastructure include:

1. Containerization: Uber uses Docker and Kubernetes for container orchestration. This allows them to package their application with all its dependencies into a "container" and run it anywhere, ensuring consistency across different environments.

2. Serverless Computing: Uber utilises AWS Lambda for serverless computing. This allows them to run their code without provisioning or managing servers, which can scale automatically with the size of the workload.

3. Database Management: Uber employs a mix of relational databases like PostgreSQL and NoSQL databases like Cassandra. This allows them to handle a wide variety of data types and use cases, from user profiles and trip histories to geospatial data for maps and routing.

Now, let's move on to the frontend architecture. The frontend of Uber's app is designed for a seamless user experience across multiple platforms. Can you think of how this might be achieved?
User: using frameworks that supports cross platform development
AI: That's right! Uber uses frameworks like React Native for building their mobile app. React Native allows developers to write code in JavaScript, which is then rendered as native platform-specific code. This means they can have a single codebase that works across multiple platforms (iOS, Android, Web), which can significantly reduce development time and effort.

In addition to React Native, Uber also uses Node.js for backend services that require quick response times. They use an API Gateway to manage and secure access to the backend services, ensuring that requests are handled efficiently and securely.

Now, let's consider the security and compliance aspect. Uber's architecture includes robust security measures to protect user data and ensure compliance with various regulations. Can you elaborate on why this might be important?
User: it is crucial to protect user data to ensure user statisfaction
AI: Absolutely, protecting user data is not only a legal requirement but also crucial for maintaining user trust and satisfaction. Uber handles sensitive information, including user profiles, payment details, and location data. Therefore, robust security measures are integral to their system architecture.

These measures include encryption for data at rest and in transit, authentication and authorization using OAuth and JWT tokens, and adherence to data privacy regulations like GDPR and CCPA. These measures help prevent unauthorized access, protect user privacy, and ensure compliance with legal requirements.

In addition to these, Uber also uses comprehensive monitoring and analytics tools to maintain the quality of service and quickly address any issues. These tools track system performance, user behaviour, and operational metrics, enabling Uber to continuously improve its service based on data-driven insights.

Now let's look at the strengths and weaknesses of this approach. What do you think they might be?
User: able to address or monitor easily however not every user would want their data to be tracked or monitored
AI: That's a good point. Indeed, the ability to monitor and analyse data in real-time is a significant strength of Uber's architecture. It allows for prompt issue detection and resolution, dynamic pricing, and enhanced user experience through personalised services.

However, this also raises privacy concerns. While data tracking and monitoring are necessary for Uber to provide its services, it's crucial that they do so in a way that respects user privacy and complies with data protection laws. Transparency about data usage and providing users with control over their data are important aspects of addressing these concerns.

Another potential weakness could be the complexity of managing a distributed system with numerous microservices and real-time data processing. This requires a robust infrastructure and skilled engineering team.