Case Study Question Answers:

01) Scalability Issues: A two-tier architecture cannot scale as readily to accommodate growing users.

Single Point of Failure: The master server is a single point of failure. When it crashes, the entire platform is down, leading to uneven availability for users in different time zones.

Performance Bottlenecks: There is only one server serving all the requests and therefore experiences performance bottlenecks.

Limited Fault Tolerance: There is no redundancy in the two-tiered design.

Geographical Latency: The users at different time zones would experience high latency since the server is located at a single location

02) Scalability: The separation of the application into three tiers allows each tier to scale independently
Load Distribution: The workload is shared among several servers, reducing the workload on any single server and improving performance.

Fault Tolerance: Redundancy can be introduced at each layer such that the system is still operational even if one component fails.

Geographical Distribution: Servers can be distributed across different regions to reduce latency for users in different time zones.

Modularity: Each tier can be constructed, operated, and scaled independently, making the system more modular and simpler to manage.

03) Video Streaming: Video streaming can be done over P2P, where video chunks are exchanged by students amongst themselves, offloading the load from the central server and improving the streaming performance.

File Sharing for Assignments: Assignment files can be exchanged directly peer-to-peer by students without centralized storage, reducing the uploads/downloads.
Live Chats: decentralized real-time chatting for live discussions

04) Security Concerns -

Data Privacy: Private user data must be protected from unauthorized access.

Authentication and Authorization: Ensuring that only authorized users can access particular resources.

Data Integrity: Preventing data from being altered while in transit or storage.

Malicious Peers in P2P: In a P2P system, malicious peers may attempt to propagate malware or infect data.

Distributed Denial-of-Service (DDoS) Attacks: The system can be attacked by DDoS, which would overwhelm the servers.

Solutions -

Encryption: Encrypt data in transit and at rest.

Access Control: Implement strong authentication and role-based access control (RBAC) to restrict resource access.

Data Integrity Checks: Use cryptographic hashes to verify the integrity of data.

Peer Reputation Systems: Employ reputation systems to identify and exclude malicious peers in P2P systems.

DDoS Mitigation: Leverage firewalls, load balancers, and distributed traffic filtering to protect against DDoS attacks.

05) Recommended Architecture: Event-Driven Architecture.

Real-Time Communication: Event-driven architectures are ideally suited for real-time communication since they can handle asynchronous events well.

Scalability: Events can be passed on various servers, allowing easy horizontal scaling as more users join.

Low Latency: Event-driven systems can handle and send messages efficiently in real-time, ensuring interactive conversations have low latency.

Decoupling: The pieces are able to function independently, so the system is more modular and simpler to manage.