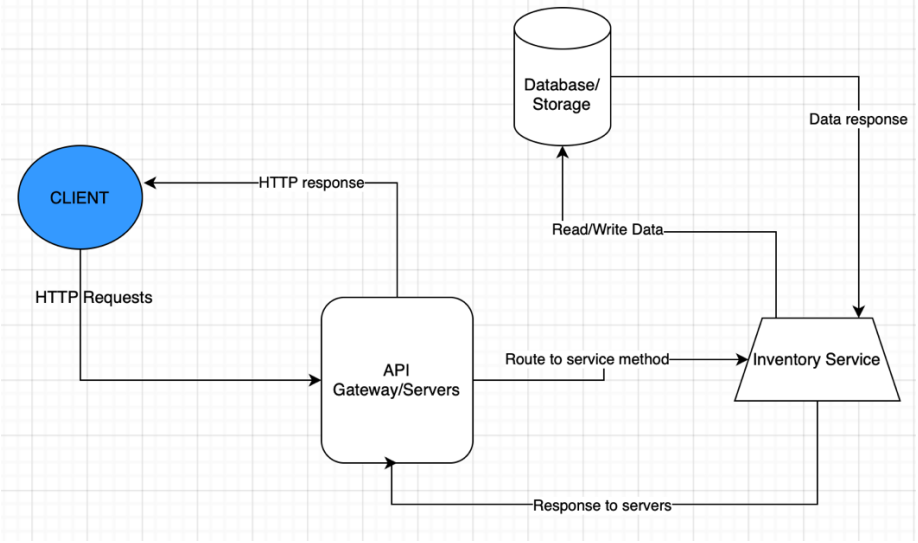
DESIGN DOCUMENT

Illustration of your Design:

Client that communicates with the system through a server that serves as an API gateway by making HTTP requests to it.The Inventory Service, which has the central logic for inventory management, is contacted by the server after it has processed these requests. Item definitions, additions, deletions, and backup operations-including the storing and loading of snapshots—are all managed by this service. Ultimately, the Database/Storage layer ensures that the system can effectively restore earlier states by preserving backup snapshots and inventory data. Separation of concerns, maintainability, and scalability are guaranteed by this modular design.



Benefits of the design:

To facilitate independent development and testing, the client, server, and inventory service components are divided into separate components in a modular and scalable design. Future feature additions are made possible by this framework, which also improves maintainability. The system is made more resilient by having snapshot-based backup technology, which allows it to recover from errors and restore data. Additionally, the system can easily support additional load and be expanded with new capabilities thanks to the straightforward API design.

Value of the benefits:

The ability to improve the efficiency and dependability of the system. Modularity reduces development time and potential issues while adding new features by improving code maintainability. Critical data recovery capabilities are provided by the snapshot-based backup capability, which reduces downtime in the event of a system breakdown and is particularly useful in real-world applications. The system's long-term viability and adaptability are ensured by the transparent API structure, which makes it simple to integrate with other services and scale in the future.

Drawbacks of the design:

The additional complexity brought about by the addition of backup and restore features is one possible flaw in the design. This may result in higher memory consumption and possible performance snags while handling substantial amounts of inventory data. Furthermore, especially in high-frequency transaction situations, depending too much on snapshot-based backups may lead to some data discrepancy between the most recent saved snapshot and the present state. To make sure that the backup and restore procedures are solid and dependable, the design also makes extensive testing and error management more necessary.

Impact of the drawbacks:

The disadvantages could result in poorer user experience if backup or restore activities take longer than expected, particularly when dealing with huge datasets. Inaccurate inventory monitoring may arise from data discrepancies between saved snapshots and the present condition. Furthermore, if edge cases in the backup/restore procedure are poorly handled, the added complexity may present maintenance issues and debugging tasks.

Value Analysis:

The advantages of having a modular design and backup features exceed the disadvantages of having more complexity and possible performance problems. Reduced downtime and greater maintainability are a result of enhanced dependability and data integrity. The whole investment promotes long-term operating efficiency and customer happiness, despite complexity management problems, which makes the design an important asset for the inventory service.

LINK TO THE FINAL CODE:

Final Code commit:

https://github.com/SaiAkashNekkanty/InventoryServer-Activity1/commit/6087fb4eee08c9b8e62d1728ebd49319511d6bb5