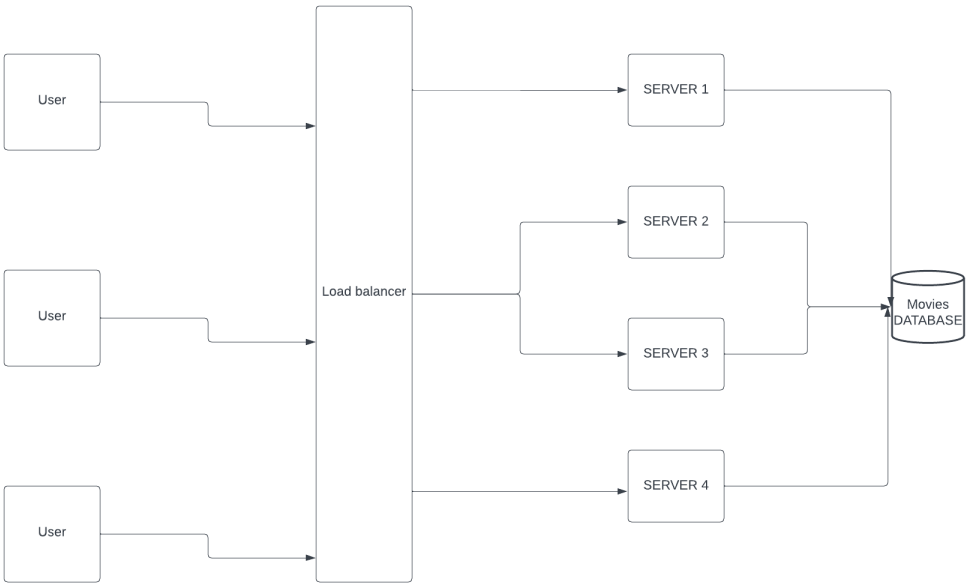
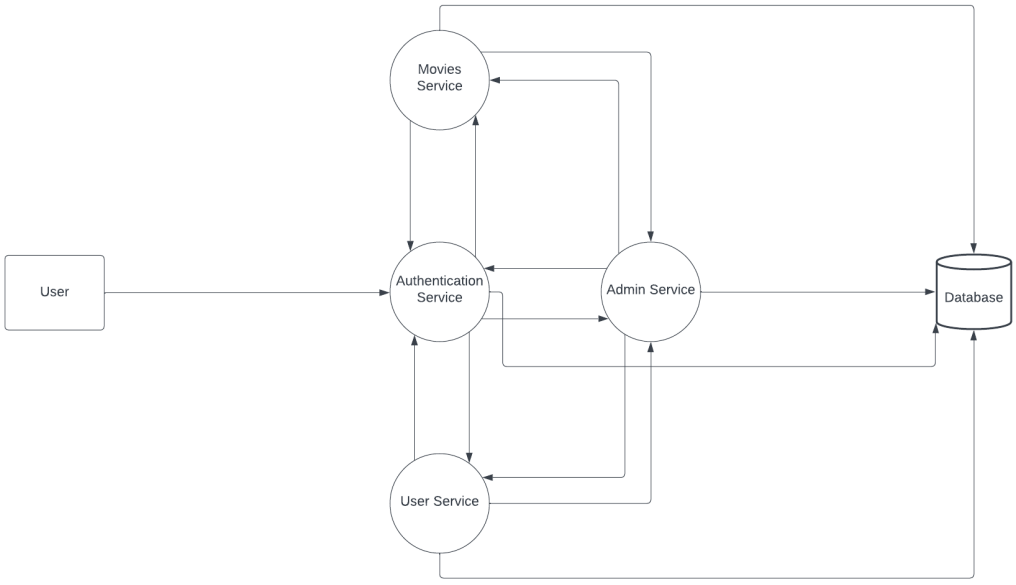
Scaling Movies API

- **Load Balancing**: The API servers can be horizontally scaled by using multiple servers and load balancers can be used to equally distribute the incoming request to multiple servers. This will eventually reduce the load on one server.



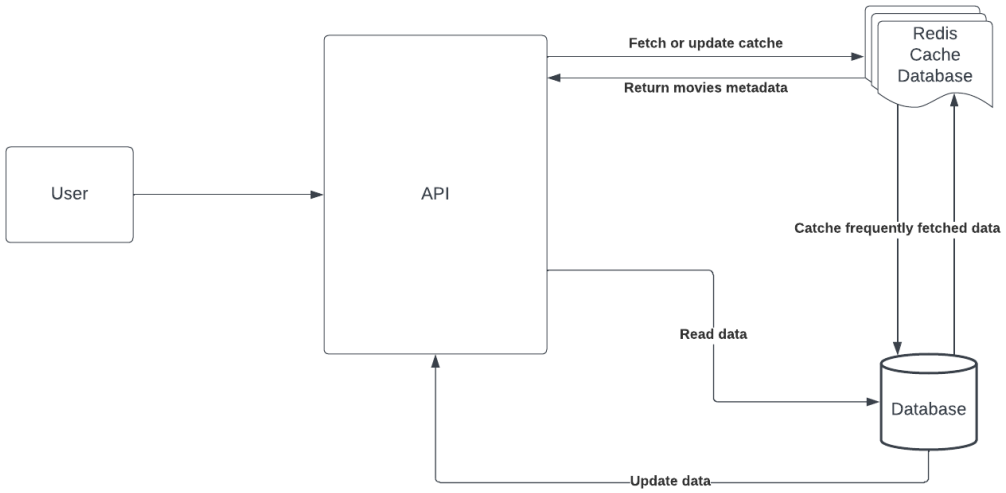
- **Microservices Architecture:** The whole application can be broken down into sub systems or microservices.
    - **Movies:** This will only serve movies metadata
    - **Authentication:** This will handle authentication
    - **User Services:** This subsystem will only handle requests made by users
    - **Admin Services:** This can also be treated as a separate service



Each of these services can be scaled individually based on the workload on each microservice. This architecture also solves the server breakdown problem as if one microservice breaks the rest of the application is not affected.

- **Content Delivery Network:** The static content can be served through CDNs hence further reducing the load on the server for sending static content.

- **Caching and Database scaling:** The movie metadata can be distributed across multiple database instances using. By using services like AWS S3 bucket. We can create instances of the data based on criteria, say grouping by genres. This will reduce the load on the database serves as well. We can also implement caching, by this we can cache the frequently requested movies metadata and serve it directly through the cache. We can use tools like Redis for this.



- **Rate Limiting:** We can implement rate limiting to prevent API abuse. This can be implemented by adding limits, say 500 API requests per day for each JWT authentication token. This will prevent DDOS attacks and also reduce the overall load on the API server.

- **Asynchronous Programming:** Python allows us to write asynchronous code with async/await. This will move all the CPU intensive tasks to the event loop and will be only called when the server is free to process tasks. By using asynchronous programming, we make sure every request is served equally and no other tasks are blocking the main thread.