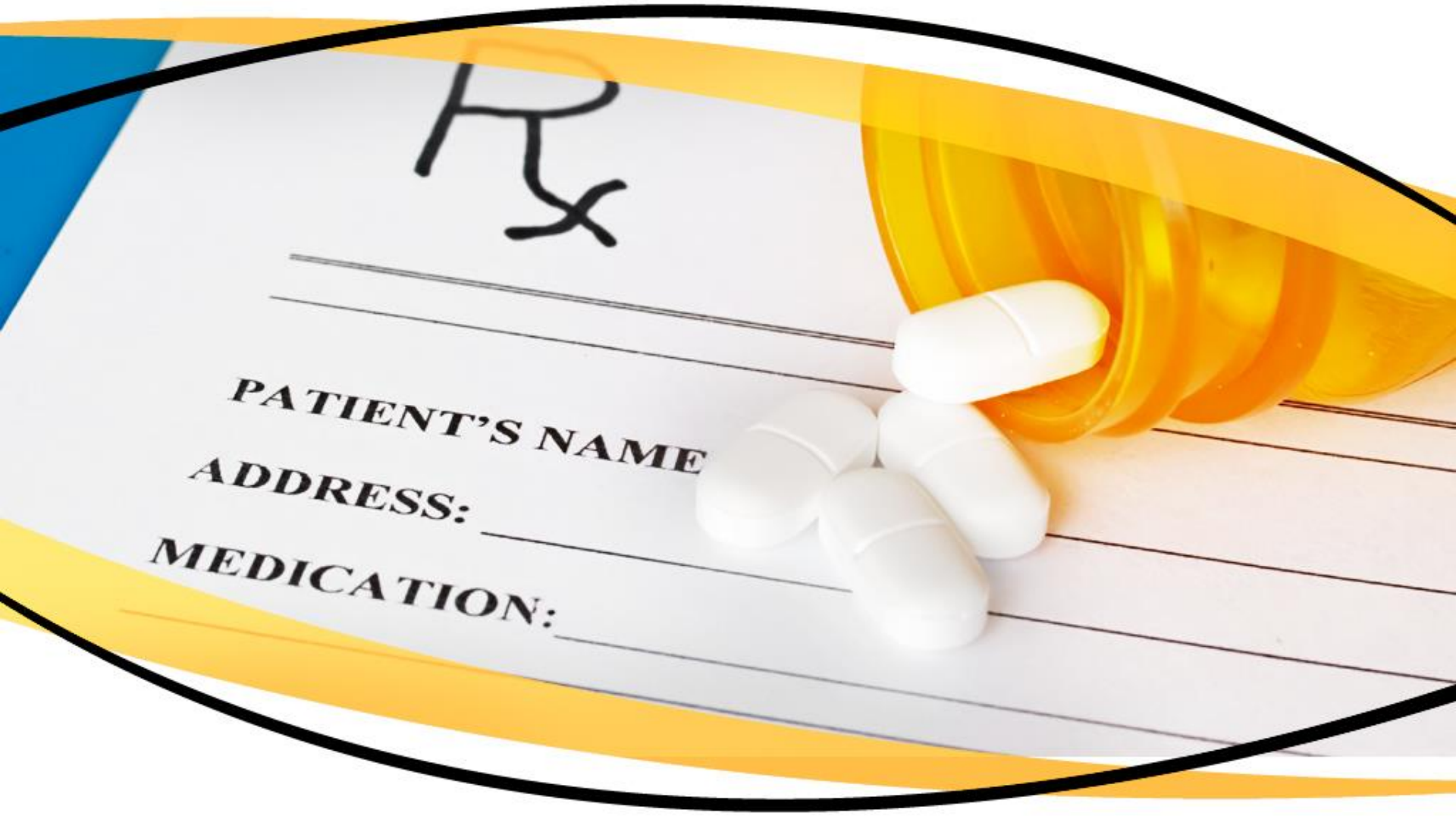
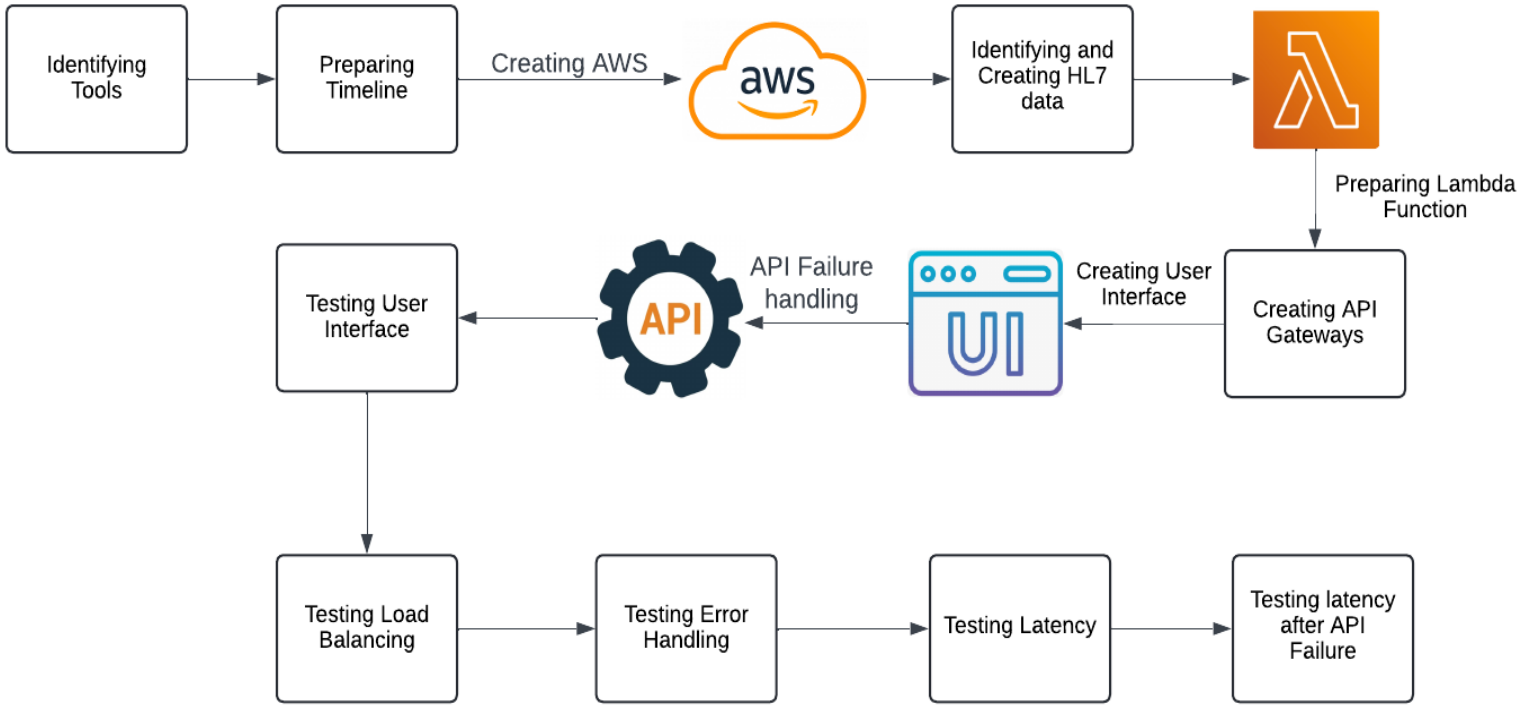


Centralized Sharing of Medical Records Using Cloud Platform

Research Overview

- This project aims to propose and assess the effectiveness of a **cloud-based medical records sharing system** that was built with **AWS Lambda, AWS API Gateway and Python**.
- The system is designed to provide the user with a fast and secure way of accessing patient's data through a server.
- **Aim:** To develop a centralized medical records sharing system using a cloud platform in the UK healthcare sector, facilitating seamless data transfer by effectively approaching API endpoints with load balancing techniques and Error Handling.
- **Research Question:** Is there a direct impact on the latency difference while handling load balancing techniques and during API failures?

Workflow of the Project



Outcomes

Response times of two API endpoints

Test	Virginia (Server 1)	Ohio (Server 2)
Test 1	0.84	0.81
Test 2	0.82	0.83
Test 3	0.77	0.85
Test 4	0.81	0.86
Test 5	0.82	0.82
Test 6	0.9	0.89
Test 7	0.89	0.82
Test 8	0.87	0.8
Test 9	0.86	0.85
Test 10	0.86	0.84

Latency of API Failure

Test	Virginia (Server 1) in milliseconds	Ohio (Server 1) in milliseconds
Test 1	4.72	4.92
Test 2	4.1	4.67
Test 3	4.2	4.4
Test 4	4.7	4.7
Test 5	4.66	4.6
Test 6	4.6	4.3
Test 7	4.6	4.6
Test 8	4.82	4.9
Test 9	4.1	4.8
Test 10	4.1	4.4

Main Findings

- The **load balancing technique** used in the application which is round-robin was able to **spread the API requests** across several endpoints.
- When the Ohio API endpoint was deliberately stopped, the **application worked smoothly as the requests were forwarded to the Virginia API**.
- The latency test showed that **response time for the Virginia and Ohio servers was relatively similar** in the normal scenario with slight fluctuations between the two.
- When Ohio (Server 2) became unresponsive, the **latency for the Virginia (Server 1) endpoint saw a slight rise from 4. 1 to 4. 82 milliseconds**. Likewise, when Virginia was tested after its failure, latency results were between 4. 3 to 4. 92 milliseconds.
- The **User Interface (UI)** to get and present patient records was **effective in the testing phase**.

Challenges Faced

- ***Inability of the API*** *to call data from the Lambda function at the beginning.*

Overcome by *checking on the Lambda function settings and permissions, and made sure that it has the ability to read from the sources.*

- **Usage of EC2 instances**

Overcome by *integration of the Lambda functions facilitated the deployment process*

- **Errors in API failures**

Overcome by *adding if-else conditions to handle the failover to ensure that the system remains stable*

Future Scope

✓ *Extend the system to* ***accommodate multiple locations****.*

✓ ***Management of Large Database*** *Medical Records to enhance the data storage to cater for the large volumes of data.*

✓ ***Integrating AWS CloudWatch*** *for monitoring and performance metrics gives a very important factor in the system's performance.*

Thank You