The National University of Lesotho

Department of Mathematics and Computer Science

Faculty of Science and technology

CS4430: Distributed database

Task: System Design

Due: 23 April 2023

Team members:

	Student Number	Surname, initials	Year of study	Major
1.	201901913	Tautona, T.M	IV	CS and Stats
2.	201903761	Moketa, L.N	IV	B. Eng
3.	201903874	Libabe, M.J	IV	CS
4.	201903645	Pholoana, T.I	IV	CS
5.	201901706	Lefaphana, B.L	IV	CS

SYSTEM DESIGN

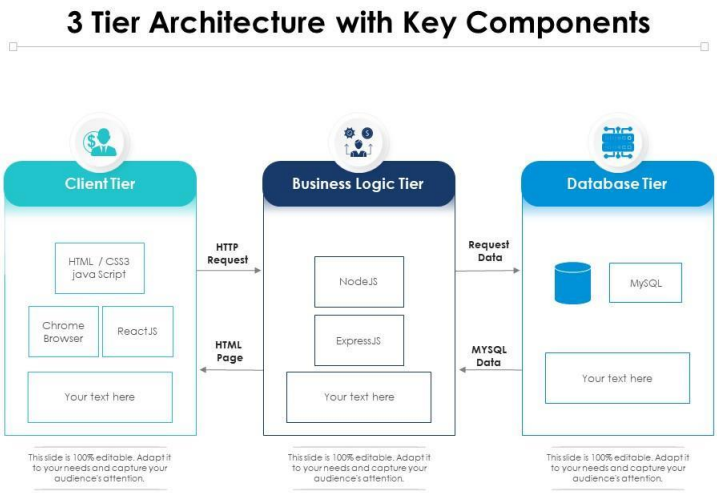
For a drug inventory distributed database system, we would adopt a three-tier architecture, which consists of a presentation layer, application layer, and database layer. This is because with this architecture, each tier runs on its own infrastructure, therefore each
can be updated or scaled as needed without impacting the other tiers.

Hardware Architecture:
For the hardware architecture, we would choose a combination of on-premise and cloud-based servers. The on-premise server would host the database and application servers, while the cloud server would be used for backup and disaster recovery purposes.

Networking Architecture:
For the networking architecture, we would use a combination of local site and remote networking. The local site would be used for accessing the system within the same network, while remote networking would be used for accessing the system from other locations.

Software Architecture:
For the software architecture, we would use a platform-independent approach, which means that the system should be accessible from any device or operating system. I would also use a web-based application, which allows for easy access from any device with a web browser.

The following diagram illustrates the proposed architecture:



The presentation layer consists of the user interface, which is accessible via a web browser. The user interface allows users to interact with the system by viewing, adding, or modifying drug inventory data. The application layer consists of the application server, which handles user requests and performs data processing, and the database server, which stores and manages the drug inventory data.

The rationale for this design is that it allows for easy scalability and maintenance of the system. Using a three-tier architecture allows for separation of concerns, which means that each layer can be upgraded or modified independently without affecting the other layers. The use of both on-premise and cloud-based servers provides a reliable backup and disaster recovery solution, ensuring that the

system remains accessible and operational even in the event of a hardware failure. The use of a web-based application makes the system easily accessible from any device or operating system, without requiring any additional software installation.

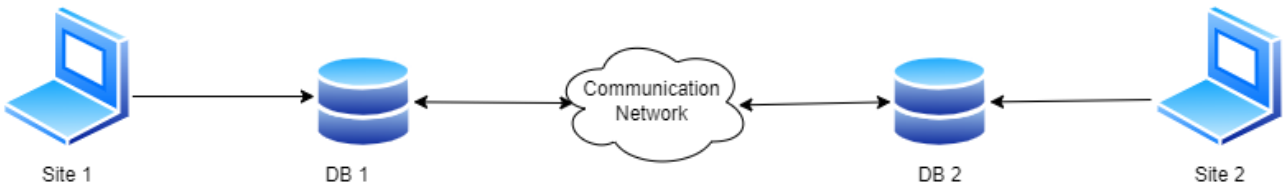
Drug Inventory Architecture:

Hardware Architecture:

Two servers: both remote servers
Each server should have a minimum of 4GB RAM, a dual-core CPU, and a 250GB hard drive

Networking Architecture:

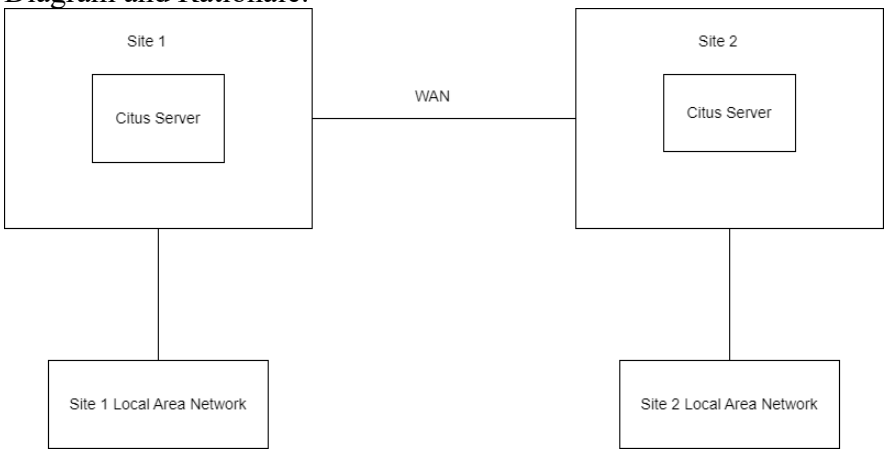
The drug inventory distributed database system will consist of two sites, each with a Citus cluster running on a laptop.
The system has two locations, each with a laptop acting as a server. These laptops are connected to the local area network (LAN) at each location, which allows them to communicate with other devices on that network.
Each laptop server would be connected to the local area network (LAN) at its respective site, allowing it to communicate with other devices on the LAN. The WAN connection between the two sites would provide connectivity between the laptop servers, allowing them to transfer data and execute queries across the distributed database.
To ensure secure communication between the two sites, a VPN or other secure communication protocol over the WAN should be used.



Software Architecture:

Platform: Linux Ubuntu Server OS.
Application: The drug inventory system will be built using a web-based platform with HTML, CSS, JavaScript, and PHP. The system will use a MySQL database for data storage. It will also use Citus, PostgreSQL and maybe a load balancer such as HAProxy or Nginx.

Diagram and Rationale:



Rationale:

We opted for Ubuntu Server OS as it's open source and widely used for server systems.
Web-based platform with HTML, CSS, JavaScript, and PHP was chosen as it's easily accessible through a web browser and can be accessed from any device.
We picked MySQL as it's a robust and widely used database system. We also chose PostgreSQL because it provides advanced features for data management and security, and is widely used in enterprise systems. By building the system on top of PostgreSQL, it can benefit from its advanced features and capabilities.
The VPN connection between the local and remote servers ensures secure and encrypted communication.

Detailed Distributed Database Design

Designing a distributed database system for drug inventory requires careful consideration of various factors such as data consistency, availability, fault tolerance, and scalability. Below we provide a detailed design that takes into account these factors.

Global Conceptual Schema:

The global conceptual schema for the drug inventory distributed database system can be designed as follows:

Drug (drug_id, name, description, category, price, quantity, manufacturer)

Inventory (inventory_id, drug_id, location, quantity)

Order (order_id, drug_id, quantity, date, status)

Supplier (supplier_id, name, contact_details, address)

DRUG

Drug_ID	Name	Description	Category	Price	Location
STP123	Penicillin	Treat bacterial infections	Antibiotics	M99	Botha Bothe
POP234	Albumin	Hydration	Intravenous fluids	M45	Hlotse
SOC763	Paroxetine	Treat depression	antidepressants	M72	Botha Bothe
FIL567	Heparin	Reduce Blood clots	anticoagulants	M37	Roma
EXO765	Morphine	Relieve Pain	analgesics	M92	Mafeteng

INVENTORY

INVENTORY_ID	DRUG_ID	QUANTITY	MANUFACTURER
3	STP123	10	Bilong_pty ltd
1	STP123	15	Acrileg_pty ltd

4	POP234	35	Man_city ltd
7	FIL567	60	Bilong_pty ltd
5	EXO765	10	Man_city ltd

ORDER

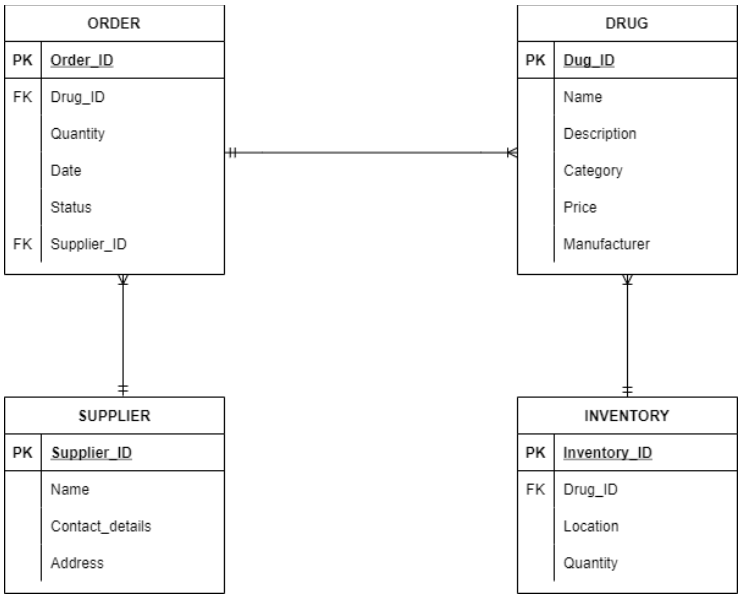
0RDER_ID	DRUG_ID	SUPPLIER_ID	QUANTITY	STATUS	DATE
242	STP123	A1	10	Pending	20 March
652	STP123	A4	15	Delivered	31 March
743	POP234	A3	35	Delivered	20 March
234	FIL567	A2	5	Processing	24 April
432	EXO765	A1	10	Pending	24 April

SUPPLIER

Supplier_ID	Name	Contact_details	Address
A1	Thabo Bester	2665657663	Khapung
A2	Warren Masemola	2665648363	Manonyane
A3	Lucas Ntuli	2665847393	Kamoho
A4	Soki Sebotsa	2663437393	Tlokoeng
A5	Alfred Maimane	2667383739	Mononts'a

The Drug entity stores information about the drugs, including their unique identifier, name, description, category, price, quantity, and manufacturer. The Inventory entity stores information about the inventory levels of drugs at different locations, including the inventory's unique identifier, the drug's unique identifier, the location, and the quantity. The Order entity stores information about the orders placed for drugs, including the order's unique identifier, the drug's unique identifier, the quantity, the order date, and the status. The Supplier entity stores information about the suppliers of the drugs including their unique identifier, name, contact details and address.

Entity-Relationship diagram



Data Partitioning:

To ensure efficient data retrieval and minimize the network traffic, we can partition the data based on the drug's unique identifier. One approach to partitioning the data is to use a range-based partitioning scheme. In this scheme, we can divide the range of drug IDs into multiple partitions, and each partition can be assigned to a different node in the distributed database system. For example, if we have 100 drugs with unique identifiers ranging from 1 to 100, we can divide the range into 10 partitions, each containing 10 unique drug identifiers.

FRAGMENTING THE INVENTORY RELATION

The simple predicates that would be used to fragment according to the first application are:
p1: MANUFACTURER = “Bilong_pty ltd”
p2: MANUFACTURER = “Acrilec_pty ltd”
p3: MANUFACTURER = “Man_city ltd”

The simple predicates that would be used to fragment according to the second application are:
p4: QUANTITY ≤ 15
p5: QUANTITY > 15

Now, the following six minterm predicates that form M can be defined:
m1: (MANUFACTURER = “Bilong_pty ltd l”) ^ (QUANTITY ≤ 15)
m2: (MANUFACTURER = “Bilong_pty ltd l”) ^ (QUANTITY > 15)
m3: (MANUFACTURER = “Acrilec_pty ltd”) ^ (QUANTITY ≤ 15)
m4: (MANUFACTURER = “Acrilec_pty ltd”) ^ (QUANTITY > 15)
m5: (MANUFACTURER = “Man_city ltd”) ^ (QUANTITY ≤ 15)
m6: (MANUFACTURER = “Man_city ltd”) ^ (QUANTITY > 15)

The resulting horizontal fragmentation of INVENTORY forms six fragments $F_{INVENTORY} = \{INVENTORY_1, INVENTORY_2, INVENTORY_3, INVENTORY_4, INVENTORY_5, INVENTORY_6\}$. Since fragment $INVENTORY_4$ is empty, it is excluded.

$INVENTORY_1$

INVENTORY_ID	DRUG_ID	QUANTITY	MANUFACTURER
3	STP123	10	Bilong_pty ltd

$INVENTORY_2$

INVENTORY_ID	DRUG_ID	QUANTITY	MANUFACTURER
7	FIL567	60	Bilong_pty ltd

$INVENTORY_3$

INVENTORY_ID	DRUG_ID	QUANTITY	MANUFACTURER
1	STP123	15	Acrileg_pty ltd

$INVENTORY_5$

INVENTORY_ID	DRUG_ID	QUANTITY	MANUFACTURER
5	EXO765	10	Man_city ltd

$INVENTORY_6$

INVENTORY_ID	DRUG_ID	QUANTITY	MANUFACTURER
4	POP234	35	Man_city ltd

FRAGMENTING THE ORDER RELATION:

The simple predicates that would be used to fragment according to the first application are:
p1: DATE = “20 March”
p2: DATE = “31 March”
p3: DATE = “24 April”

The simple predicates that would be used to fragment according to the second application are:
p4: QUANTITY ≤ 15
p5: QUANTITY > 15

Now, the following six minterm predicates that form M can be defined:
m1: (DATE = “20 March”) ^ (QUANTITY ≤ 15)
m2: (DATE = “20 March”) ^ (QUANTITY > 15)
m3: (DATE = “31 March”) ^ (QUANTITY ≤ 15)
m4: (DATE = “31 March”) ^ (QUANTITY > 15)
m5: (DATE = “24 April”) ^ (QUANTITY ≤ 15)
m6: (DATE = “24 April”) ^ (QUANTITY > 15)

The resulting horizontal fragmentation of ORDER forms six fragments $F_{ORDER} = \{ORDER_1, ORDER_2, ORDER_3, ORDER_4, ORDER_5, ORDER_6\}$.Since fragments $ORDER_4$ and $ORDER_6$ are empty, they are excluded.

$ORDER_1$

0RDER_ID	DRUG_ID	SUPPLIER_ID	QUANTITY	STATUS	DATE
242	STP123	A1	10		20 March

$ORDER_2$

0RDER_ID	DRUG_ID	SUPPLIER_ID	QUANTITY	STATUS	DATE
743	POP234	A3	35		20 March

$ORDER_3$

0RDER_ID	DRUG_ID	SUPPLIER_ID	QUANTITY	STATUS	DATE
652	STP123	A4	15		31 March

$ORDER_5$

0RDER_ID	DRUG_ID	SUPPLIER_ID	QUANTITY	STATUS	DATE
234	FIL567	A2	5		24 April
432	EXO765	A1	10		24 April

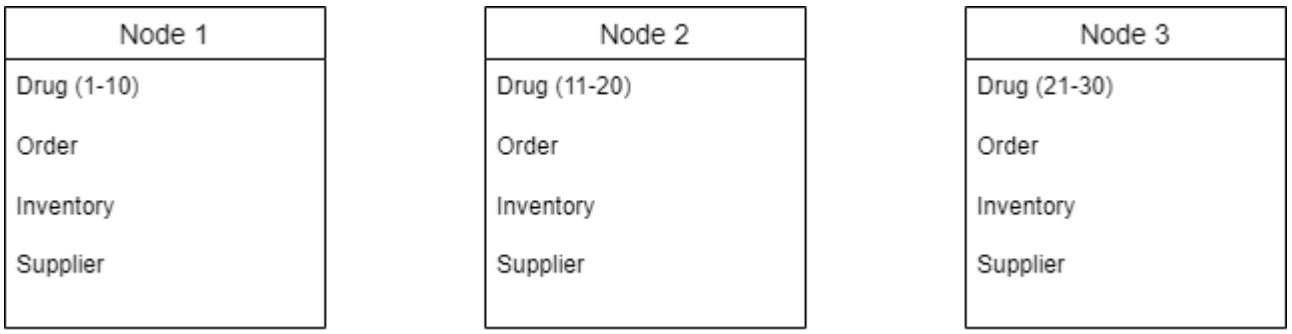
Data Allocation and Replication:

To ensure fault tolerance and availability, we can replicate the data across multiple nodes in the distributed database system. One approach to data replication is to use a master-slave replication scheme. In this scheme, one node can act as the master node, responsible for handling all write requests. The remaining nodes can act as slave nodes, responsible for handling read requests.

To allocate the data, we can use a consistent hashing algorithm that maps each drug's unique identifier to a specific node in the distributed database system. This approach ensures that each drug's data is stored on a single node and enables us to distribute the load evenly across the nodes.

Diagram and Rationale:

The following diagram summarizes the design of the drug inventory distributed database system:



In this design, we have three nodes, and each node stores the data for a specific range of drug identifiers. Each node also replicates its data to other nodes in the distributed database system, providing fault tolerance and availability.

By partitioning the data based on the drug's unique identifier, we can minimize network traffic and ensure efficient data retrieval. Using a master-slave replication scheme ensures that write requests are handled by a single node, while the remaining nodes will be responsible for handling read requests.

Detailed Test Plan:

To evaluate the quality of the drug inventory distributed database system implementation, we can perform several tests. Here's a detailed outline of the tests we can run:

Data consistency test: This test checks whether the data stored in the database is consistent across all nodes. We can randomly select a few drugs, check their inventory levels, and verify that they are the same across all nodes.

Data partitioning test: This test checks whether the data partitioning is done correctly and evenly distributes the load across all nodes. We can insert a large number of drugs into the database and check whether each node has an equal number of drugs.

Data allocation test: This test checks whether the data allocation algorithm is working correctly and distributing the data evenly across all nodes. We can insert a large number of drugs into the database and check which nodes they are stored on to ensure that the data is evenly distributed.

Data replication test: This test checks whether the data replication is working correctly and whether all nodes have the same data. We can update the inventory levels of a drug and check whether the update is reflected in all nodes.

Fault tolerance test: This test checks whether the system can handle failures gracefully. We can simulate a node failure and check whether the system can still function correctly and maintain data consistency.

Scalability test: This test checks whether the system can handle an increasing load of requests. We can gradually increase the number of requests made to the system and monitor its performance to ensure that it can handle the increased load.

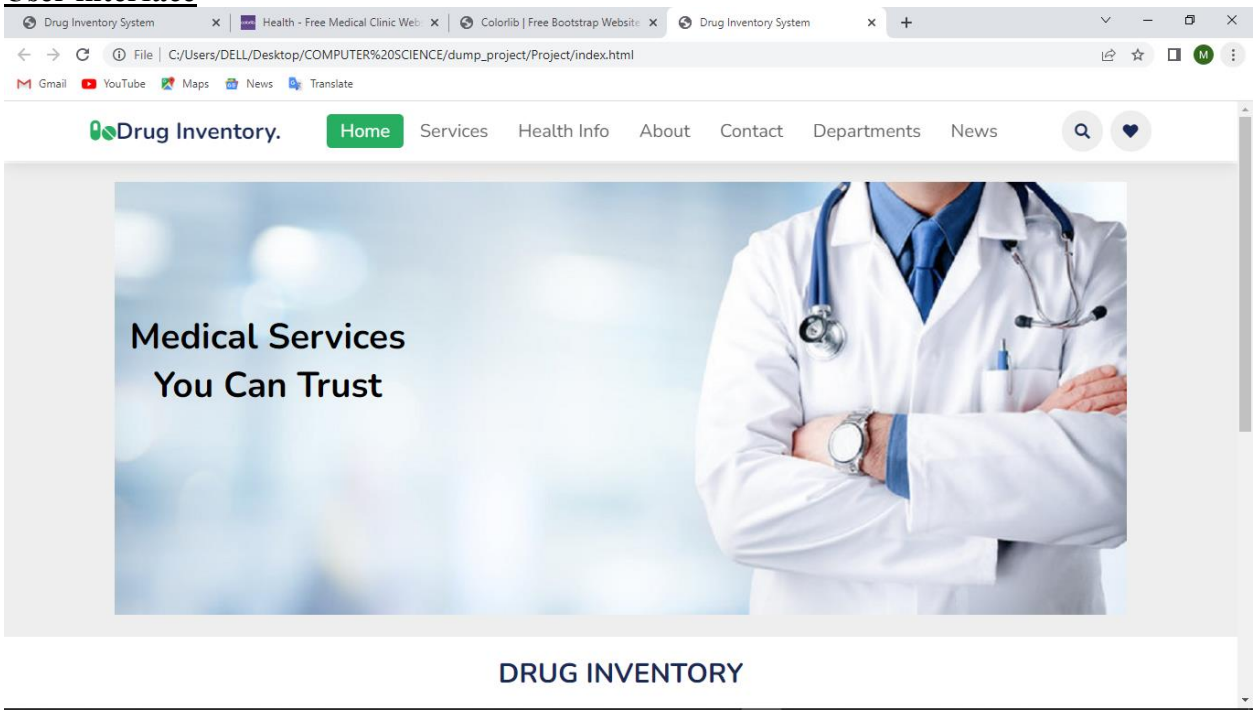
Here's a summary of the tests in a table:

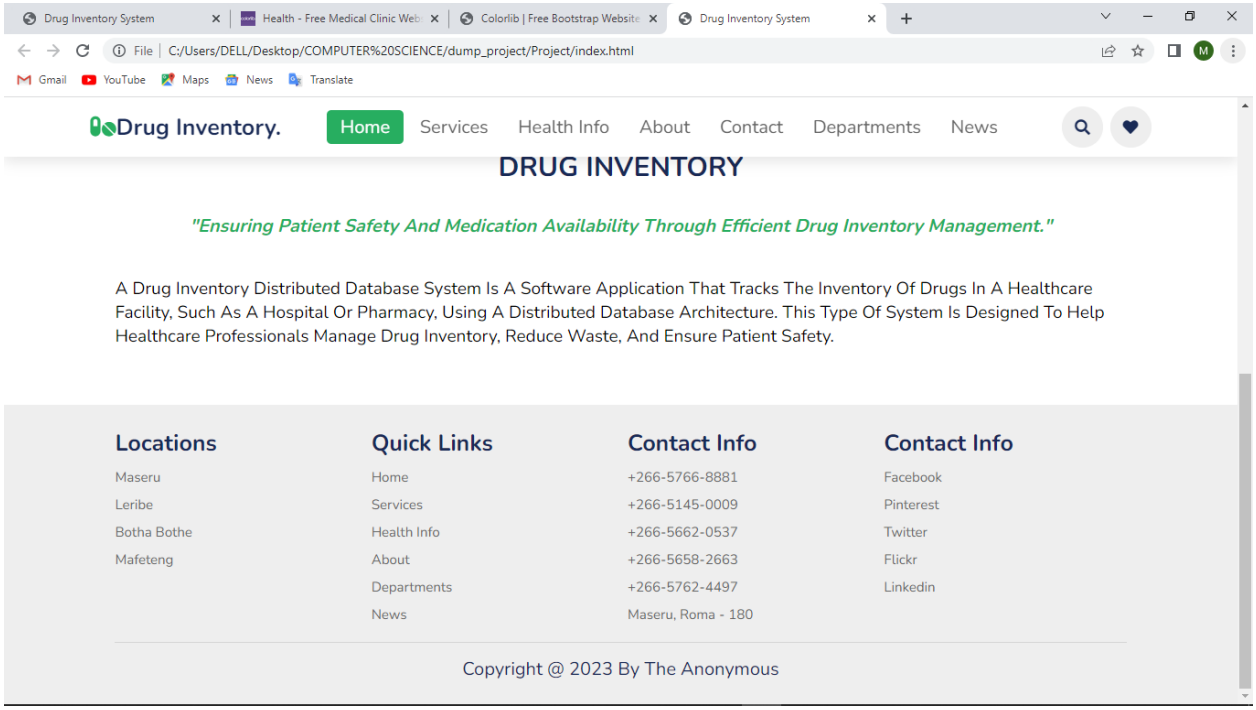
Test Name	**Description**

Data consistency	Check whether the data stored in the database is consistent across all nodes.
Data partitioning	Check whether the data partitioning is done correctly and evenly distributes the load across all nodes.
Data allocation	Check whether the data allocation algorithm is working correctly and distributing the data evenly across all nodes.
Data replication	Check whether the data replication is working correctly and whether all nodes have the same data.
Fault tolerance	Simulate a node failure and check whether the system can still function correctly and maintain data consistency.
Scalability	Gradually increase the number of requests made to the system and monitor its performance to ensure that it can handle the increased load.

By performing these tests, we can ensure that the drug inventory distributed database system implementation is of high quality and meets our requirements for data consistency, fault tolerance, scalability, and performance.

User interface




Drug Inventory System
Health - Free Medical Clinic Web
Colorlib | Free Bootstrap Website
Drug Inventory System
File | C:/Users/DELL/Desktop/COMPUTER%20SCIENCE/dump_project/Project/index.html
Gmail
YouTube
Maps
News
Translate
Drug Inventory.
Home
Services
Health Info
About
Contact
Departments
News
DRUG INVENTORY
"Ensuring Patient Safety And Medication Availability Through Efficient Drug Inventory Management."
A Drug Inventory Distributed Database System Is A Software Application That Tracks The Inventory Of Drugs In A Healthcare Facility, Such As A Hospital Or Pharmacy, Using A Distributed Database Architecture. This Type Of System Is Designed To Help Healthcare Professionals Manage Drug Inventory, Reduce Waste, And Ensure Patient Safety.
Locations
Maseru
Leribe
Botha Bothe
Mafeteng
Quick Links
Home
Services
Health Info
About
Departments
News
Contact Info
+266-5766-8881
+266-5145-0009
+266-5662-0537
+266-5658-2663
+266-5762-4497
Maseru, Roma - 180
Contact Info
Facebook
Pinterest
Twitter
Flickr
Linkedin
Copyright @ 2023 By The Anonymous