1199 CRM
Non-Functional Requirement

Digital Service Development Division,

Department of Digital Transformation,

GovTech

Table of Contents

1. Development Platform and Technology

1.1. Database System

- ❖ Recommendation: The database for the application is recommended to be implemented using an **open-source** database management system.
  - ➢ PostgreSQL
  - ➢ *Required latest stable version*.
- ❖ Rationale: These databases are chosen for their reliability, scalability, and community support. Load testing will ensure they meet performance requirements under expected loads.

1.2. Development Language and Framework

- ❖ Development Language Options:
  - ➢ Backend: Python/Java*
  - ➢ Frontend: Next.js
  - ➢ *Required latest stable version.*
- ❖ Framework Recommendation: The development framework should be selected based on a thorough understanding of the system requirements and expected load. The firm must propose a framework known for its robustness and scalability.
  - ➢ Python Frameworks: Django, FastAPI, Spring boot
- ❖ Rationale: These frameworks are well-supported, offer extensive libraries, and are capable of handling enterprise-level applications.

**Python framework if web-based application and Java framework if software based application.*

1.3. Database Design and Modelling

- ❖ Normalization and Denormalization
  - ➢ Normalization: Start with a normalized schema to eliminate redundancy and ensure data integrity.
  - ➢ Denormalization: Denormalize selectively to improve read performance, especially for frequently accessed data.
- ❖ Indexing

    - ➢ Use indexes to speed up query performance. Be mindful of the trade-offs, as excessive indexing can slow down write operations.
- ❖ Partitioning
    - ➢ Partition large tables to improve performance and manageability. Choose the right partitioning strategy (e.g., range, list, hash) based on your data access patterns.
- ❖ Data Modeling Tools
    - ➢ Use data modeling tools to visualize and design the database schema. Tools like MySQL Workbench, and Microsoft Visio can be helpful.
- ❖ Regular Audits and Monitoring
    - ➢ Regularly audit the database for performance issues and security vulnerabilities. Use monitoring tools to track database performance and health.
- ❖ Compliance
    - ➢ Ensure the database design complies with relevant regulations and standards, such as (General Data Protection Regulation) and GDPR for data protection.

1.4. System Architecture

- ❖ Architecture Style:
    - ➢ Monolithic Architecture
        - ■ Segregation: Functional and non-functional modules should be segregated.
        - ■ Scalability: High-load monolithic applications should be auto-scalable using orchestration tools or other solutions.
    - ➢ Three-Tier System Architecture
        - ■ UI/UX Tier: Handles the presentation layer
        - ■ Logical Tier: Manages application logic and data processing
        - ■ Database Tier: Handles data storage and management

- ❖ Rationale: Three-tier architecture ensures clear separation of concerns and efficient system management.

1.5. DevOps Principles and Tools

- ❖ Design Principles: The system should be developed using DevOps principles to ensure continuous integration and delivery.
- ❖ Tools:
  - ➢ Version Control: Gitlab should be used
  - ➢ Unit Testing: Frameworks specific to the chosen development language (e.g., JUnit/TestNG for Java, Pytest for Python)
  - ➢ CI/CD: Tools like Jenkins, GitLab CI/CD
  - ➢ Containerization: Docker
    - ■ Every software shall be fully compatible with the containerization
  - ➢ Cluster Management: Kubernetes
  - ➢ Deployment: Docker Compose, Dockerfile

  - ❖ DevOps implementation should be as per the **GovTech Standard (DSOM)**.
  - ❖ Backup and restore should be as per the **Govtech Standard (DSOM).**

- ❖ Rationale: Adopting DevOps principles and tools will enhance development efficiency, improve code quality, and facilitate automated deployments.

1.6. Hosting and Infrastructure

- ❖ Infrastructure Provider: The Procuring Agency in the Government data center will provide the necessary infrastructure for hosting applications, gateways, databases, and platforms in accordance with the **Cloud Division Standard, GovTech.**
- ❖ Rationale: Leveraging government-provided infrastructure ensures compliance with security policies and reduces costs associated with third-party hosting services.

2. Performance Requirements

2.1. Load Testing

- ❖ Requirement: The system should undergo load balancer testing to ensure it can handle concurrent users effectively.
- ❖ Rationale: Load testing is essential to identify performance bottlenecks and ensure the system can scale to meet user demand without degradation in performance.

2.2. Performance Metrics

- ❖ Concurrent Users: The system should be tested for performance under the expected number of concurrent users (10 current users *).

  ** Users are ISC and call-center agent which will keep on increasing as per increase in center*
- ❖ Response Time: Ensure acceptable response (2 seconds) times under peak load conditions.
- ❖ Resource Utilization: Monitor CPU, memory, and network usage during load testing to ensure optimal resource utilization.
- ❖ **In the event of system downtime:** Implement a standalone CRM system on a local PC, which allows the 1199 call center to continue receiving and logging calls during a system outage. The local CRM stores interaction data in offline mode using local memory, ensuring no customer interactions are lost. Once the live CRM system is back online, the data is automatically synchronised with the central CRM, ensuring all updates are reflected seamlessly. This approach ensures business continuity during downtime while maintaining data integrity and minimizing disruption to service.

3. Integration

- ❖ Real Time
  - ➢ Real-time interaction between systems may be required to optimize business processes. This might include bidirectional access through API’s, such as:
    - ➢ Web services

  - ➢ RESTful Services
  - ➢ NDI Integration
  - ➢ Other API interfaces
- ❖ All API integration should route through National Data Exchange (NDE).

4. Batch processing

- ❖ Batch data import and export is required in the system. Please provide information on how your solution allows for these interactions, including:
  - ➢ Acceptable formats (e.g., Excel,CSV, pipe-delimited, etc…)
  - ➢ Scheduling options
  - ➢ Secure transfer capabilities (e.g., SFTP)

5. Usability

- ❖ Ease of Use:

  Prioritize UX/UI Design: Implement a user-friendly interface with intuitive navigation and clear, consistent workflows. Conduct usability testing to gather feedback and refine the design to meet users' needs effectively.

- ❖ Device, Browser, and OS Support:

  Multi-Platform Compatibility: Ensure the solution is compatible with major operating systems, including Windows, macOS, and Linux.

  Browser Support: The application should work across all major browsers (Chrome, Firefox, Safari, Edge), with a focus on the latest stable versions and support for at least the two most recent versions of each browser.

  Device Responsiveness: Optimize the solution for desktop, tablet, and mobile devices. Use responsive design principles to ensure the application is accessible and functional on different screen sizes and orientations.

- ❖ Administrative Access: Minimize the need for administrative rights for end-users to operate the software. Design the application to function efficiently with standard user permissions to enhance security and ease of deployment.

6. Code Quality Standard (Clean Code)

Comply with the clean code standard of GovTech Standard Annexure I.

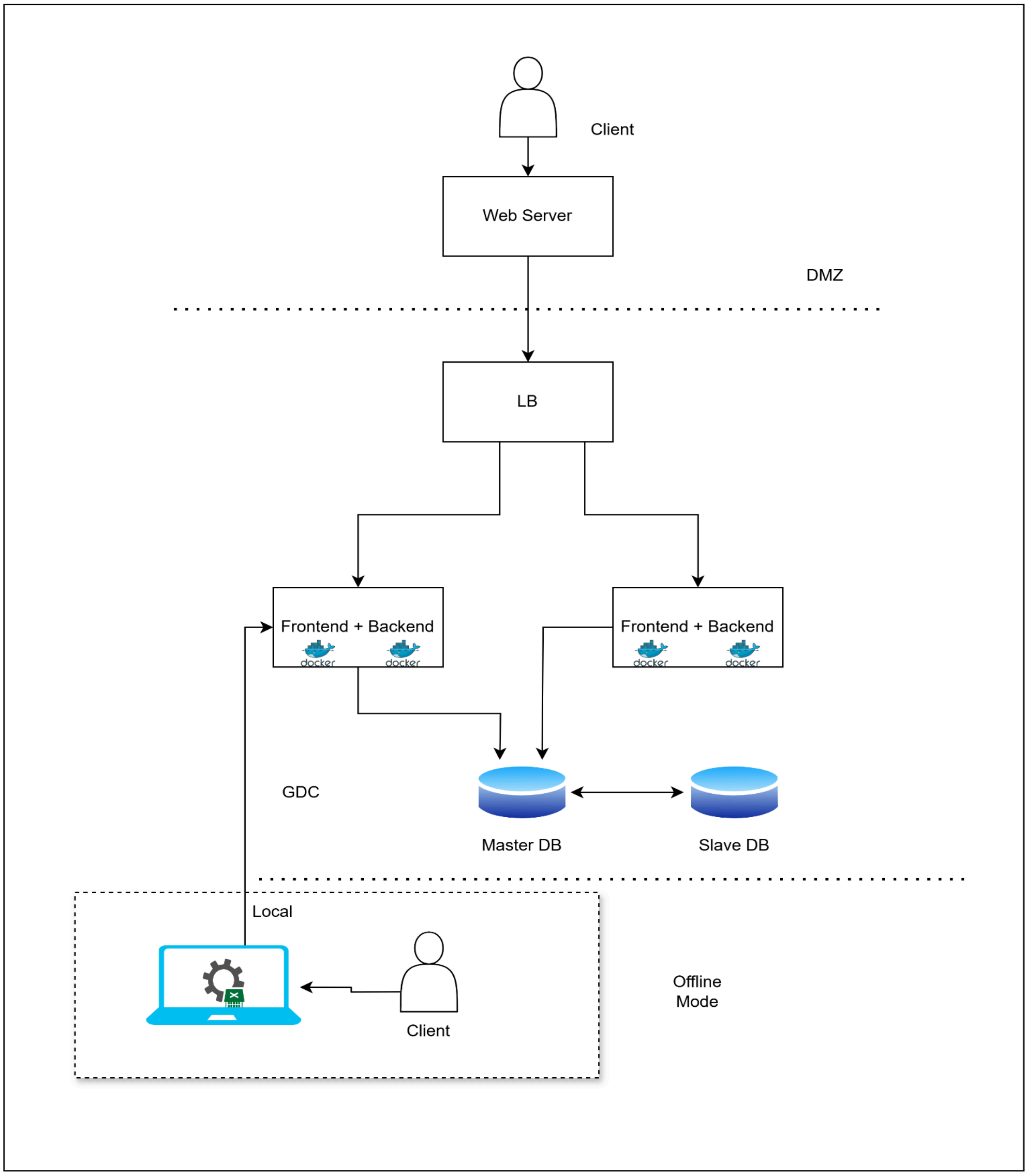
7. Table: Non-Functional Requirements

Sl.No	**Non-functional Requirement**	**Level**	**Reason**
1.	Scalability	Medium	The system should handle growth from 5 to 50 concurrent users. While scaling is necessary, the number of users is relatively small, so medium priority is given to ensure the system can support the increase.
2.	Availability	High	High availability is critical, as the call center needs to be accessible at all times. Downtime can severely impact customer service, making this a top priority.
3.	Reliability	High	The system needs to be reliable to avoid interruptions in service, especially during customer calls. A reliable system is crucial to maintaining uninterrupted operations.
4.	Maintainability	High	Since the system will likely evolve as more users join, maintainability is crucial for easy updates, bug fixes, and overall system health. Ongoing support is necessary for long-term success.
5.	Usability	Medium	The system should be user-friendly, especially for

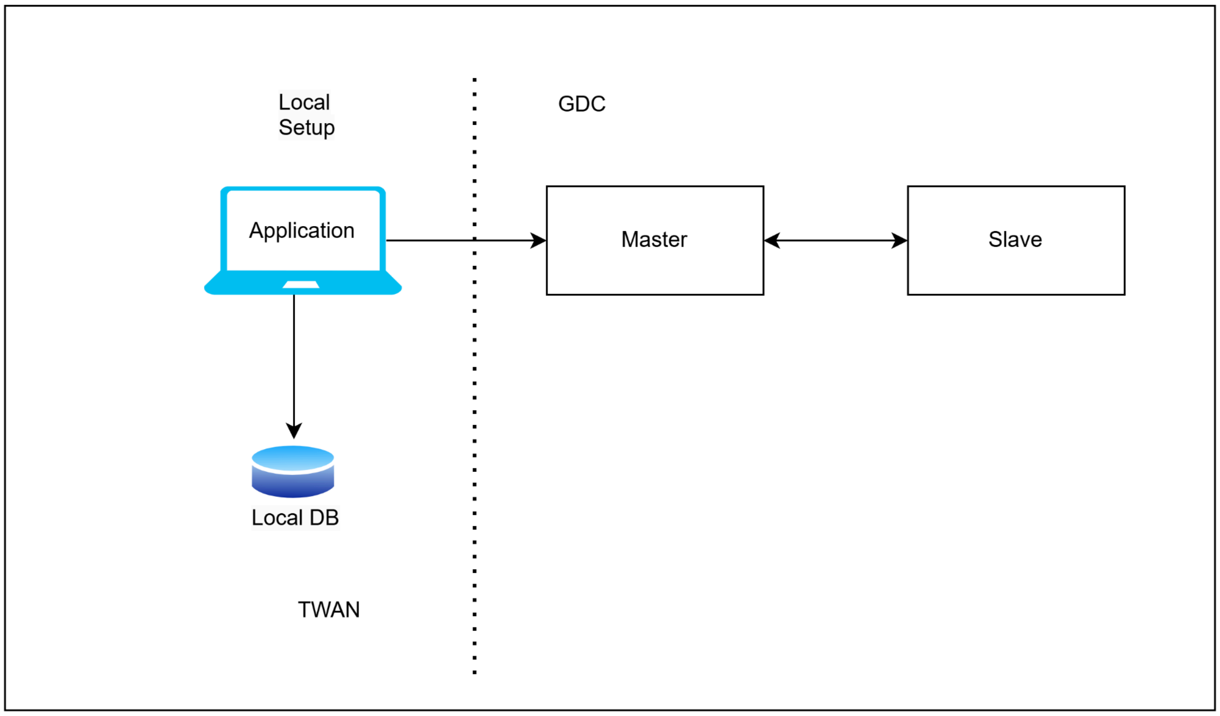
			agents interacting with it frequently. However, usability is secondary to ensuring reliability and availability, so it's given medium priority.
6.	Efficiency	High	Efficiency is important for optimizing agent productivity and minimizing response time. A call center system must handle transactions swiftly to ensure high customer satisfaction.
7.	Performance	High	High performance ensures smooth operation even when the number of concurrent users increases. The system should handle real-time customer interactions without lag or delays.

8. Proposed Solution Architecture

Web Application



Software


Local Setup
GDC
Application
Master
Slave
Local DB
TWAN