VIRTUAL DESKTOPS AND REMOTE WORK

PHASE 3: SOLUTION DEVELOPMENT AND TESTING

College Name: Shree Devi Institute of Technology, Mangalore

Group Members:

- **Name:** Abdulla Shafaz K U
  CAN ID: CAN_34027192

  CONTRIBUTION: Select tools/Performance suitable for the project type

- **Name:** Abhishek Kulal
  CAN ID: CAN_34029363

  CONTRIBUTION: Develop the product using cloud tools and services

- **Name:** Dhanya Ashok Kamat
  CAN ID: CAN_34030860

  CONTRIBUTION: Ensure proper integration of API's and SDK's

- **Name:** Shrutika Shetty
  CAN ID: CAN_34030246

  CONTRIBUTION: Test the Solution/Product to ensure and fix bugs

1. Select Tools/Platforms Suitable for the Project Type

To ensure the successful deployment of Remote desktop for remote access, the following tools and platforms are selected:

Oracle Virtual Instance

- Provides the foundational infrastructure for hosting the deployment.
- Offers scalability and flexibility to meet the project's resource requirements.

Apache Guacamole

- A clientless remote desktop gateway that enables secure access to remote systems via a web browser.
- Supports multiple protocols, including VNC, RDP, and SSH.

VNC Server

- Provides remote desktop sessions that Guacamole connects to using the VNC protocol.
- Ensures compatibility with various operating systems and desktop environments.

MariaDB

- A robust, open-source relational database used to store Guacamole's user accounts, connections, and configurations.
- Ensures data integrity and secure authentication.

Tomcat9

- A lightweight and efficient servlet container for running the Guacamole web application.
- Simplifies deployment and management of the web application.

Security Tools

- **Firewall Configuration**: Restrict access to necessary ports (e.g., 8080 for Tomcat9, 5901 for VNC Server).

2. Develop the Product Using Cloud Tools and Services

The development process involves the following stages:

Step 1: Provisioning Resources

- Provision an Oracle Virtual Instance with the required specifications (e.g., Ubuntu 20.04, 2 vCPUs, 4 GB RAM).
- Configure network settings to allow access to necessary ports (8080, 3306, 5901).

Step 2: Integration of Tools

- Install and configure Apache Guacamole, VNC Server, MariaDB, and Tomcat9 on the Oracle Virtual Instance.
- Integrate MariaDB with Guacamole for user authentication and connection management.
- Configure the VNC Server to provide remote desktop sessions.

Step 3: Security Measures

- Configure firewall rules to restrict access to the Oracle Virtual Instance.
- Use strong passwords and access controls for MariaDB and VNC Server.

3. Ensure Proper Integration of APIs or SDKs

Efficient integration of APIs and SDKs is critical for operational reliability.

Key steps include:

Guacamole API Integration

- Use Guacamole's REST API to manage connections, users, and permissions programmatically.
- Integrate the API with existing IT management tools for streamlined administration.

Database Integration

- Use MariaDB's native connectors to integrate with Guacamole for secure data storage and retrieval.
- Ensure proper error handling and retry mechanisms for database operations.

Documentation

- Maintain detailed documentation of integration processes, including API endpoints, authentication mechanisms, and configuration steps.

4. Test the Solution/Product to Ensure and Fix Bugs

Testing ensures the reliability and performance of the deployment. Key testing steps include:

Functional Testing

- Verify that users can access the Guacamole web interface and log in successfully.
- Test remote desktop connections via the VNC protocol to ensure seamless access.

Stress Testing

- Simulate multiple concurrent users accessing the Guacamole web interface to validate performance under load.
- Monitor system resources (CPU, memory, network) during stress testing.

Error Simulation

- Create controlled failure scenarios, such as database downtime or VNC Server unavailability, to test system resilience.
- Verify that the system can recover gracefully from failures.

Bug Resolution

- Track and resolve bugs using issue management tools (e.g., Jira, GitHub Issues).
- Conduct iterative testing cycles to ensure all issues are resolved.

5. Measure Performance Using Monitoring Tools

Performance measurement helps optimize the system's efficiency and reliability. Key practices include:

Monitoring Tools

- Use Oracle Cloud Monitoring to track system performance metrics, such as CPU usage, memory consumption, and network traffic.
- Set up alerts for predefined thresholds (e.g., high CPU usage, low disk space).

Analytics Dashboards

- Create visual dashboards to monitor key performance indicators (KPIs) in real-time.
- Use historical data to identify trends and optimize system performance.

Continuous Improvement

- Regularly review performance data to identify bottlenecks and optimize resource allocation.
- Experiment with different configurations (e.g., VNC Server settings, Tomcat9 tuning) to improve performance.