Asia Pacific Journal of Information System and Digital Transformation
2024 , Vol 1 No 01, Research Article

Virtualization for IoT and Mobile Systems: A Practical Implementation with Contiki and Android-x86

Cheah Tian Xin
Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Johor, Malaysia
hayleycheah16@gmail.com

Jolin Tan Shi Ti
Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Johor, Malaysia
tanshiti0702@gmail.com

Chai Hui Xin
Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Johor, Malaysia
huixincai31@gmail.com

Goh Shu Shan
Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Johor, Malaysia
sgoh78470@gmail.com

Ee Li En
Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Johor, Malaysia
rienee87@gmail.com

Abstract

The operational efficiency of virtualized systems is critical for resource-constrained environments. While performance is a key objective for the deployment of heterogeneous operating systems, this project, in the context of a comparative analysis between Android-x86 and Contiki OS, investigates system resource utilization. Based on virtualization and lightweight OS theory, this study measures the performance outcomes of concurrent guest OS operation. Our findings demonstrate the impact of three core metrics (CPU utilization, processing speed, and memory usage) on overall system performance. We further illustrate the trade-off between performance and efficiency, where Android-x86 achieves higher speed at greater resource cost, while Contiki offers superior resource efficiency with lower absolute performance. These findings help advance the practical understanding of virtualization for IoT and mobile systems and offer actionable insights for selecting and configuring guest operating systems based on specific hardware constraints and application requirements.

Keywords: Virtualization, Oracle VirtualBox, Android-x86, Contiki OS, Guest Operating System, System Performance, Resource Utilization

1. Introduction

The advent of virtualization technology has fundamentally transformed the scope and capabilities of computing, enabling the concurrent operation of multiple, isolated software environments on a single physical machine (Smith and Nair, 2005; Goldberg, 1974). This paradigm is particularly critical in the evolving landscape of heterogeneous systems, where the need to run diverse operating systems—from full-featured mobile platforms to minimalist systems for constrained devices—on standard hardware is paramount (Barham et al., 2003). In this context, hypervisors, or Virtual Machine Monitors (VMMs), have become the principal channel for developers and researchers to create, test, and integrate disparate systems efficiently (Rosenblum and Garfinkel, 2005).

The advantage of a virtualized environment is that it provides a controlled, reproducible, and isolated sandbox for evaluating system performance and interoperability (Suzuki et al., 2014). This is especially valuable for comparative analysis between operating systems with radically different architectural philosophies, such as a resource-intensive Android-x86 mobile OS and a lightweight Contiki OS designed for the Internet of Things (IoT) (Dunkels et al., 2004). A core technical challenge and objective within this setup is achieving seamless resource sharing and data exchange between the host and guest systems, such as through shared folder mechanisms (Adams and Agesen, 2006). Effective implementation of this functionality is a key indicator of successful virtualization configuration and guest OS integration.

Recent interest in optimizing system performance and resource utilization within virtualized environments has led to increased empirical research in this area (e.g., Menon et al., 2005; Hwang et al., 2013). The relationship between memory allocation, CPU utilization, and overall guest OS performance reflects the fundamental trade-offs in system design. How a guest OS performs under varying resource constraints is of increasing interest for deploying efficient virtualized testbeds and embedded systems (Cherkasova et al., 2007). Practical, hands-on configuration of these environments is a key motive for developing deeper technical competencies in system administration and architecture (Anderson et al., 2015).

In other words, successfully orchestrating a multi-OS virtualized lab leads to greater understanding of hardware abstraction, performance benchmarking, and cross-platform communication, which are critical for modern software development and IoT prototyping (Merkel, 2014). Competence in this area is therefore important as a way for IT professionals and engineers to build and maintain sustainable, flexible, and cost-effective development and testing infrastructures (Varia, 2010). Despite the growing utilization of virtualization for education and prototyping, there is a lack of structured, task-based guidance documenting the process from setup to performance analysis, particularly for contrasting OS types.

Prior laboratory work on virtualization has mainly addressed the theoretical underpinnings of VMMs (e.g., Popek and Goldberg, 1974) or the performance of enterprise server consolidation (e.g., VMware, 2007). As for hands-on, comparative performance analysis of guest OSes within a personal computing context, practically grounded approaches with step-by-step empirical validation are scarce. This gap in practical pedagogical resources and the foundational importance it represents for computing education and prototyping serve as motivations for this project.

This project aims to examine the setup and performance of heterogeneous guest operating systems in a virtualized environment by executing and documenting a series of structured technical tasks. Specifically, this study seeks to fulfil two primary objectives: (1) To successfully install and configure two distinct guest OSes (Android-x86 and Contiki OS) within a Type 2 hypervisor (Oracle VirtualBox) and establish shared folder functionality with the host; and (2) To observe, measure, and analyse the system performance (CPU utilization, speed, memory usage) of the host machine under varying guest OS memory allocations. We adopt a practical, task-driven methodology to build both theoretical knowledge of virtualization concepts and concrete technical skills in system configuration. This project contributes to current learning by providing a documented, replicable framework for understanding virtualization trade-offs and adds to the practical literature on hands-on system administration. Moreover, this project informs

students and developers about the procedural steps and performance considerations involved in leveraging virtualization for multi-platform development and analysis.

2. Conceptual Background

2.1 Virtualization and Hypervisors

Virtualization allows multiple guest operating systems to run concurrently on a single host through a hypervisor. Type 2 hypervisors, such as Oracle VirtualBox, run atop a host OS and are widely used for development and testing (Rosenblum & Garfinkel, 2005). Key benefits include isolation, reproducibility, and efficient resource sharing—critical for comparative OS studies.

2.2 Android-x86

Android-x86 is a port of the Android mobile OS to the x86 architecture. It is a full-featured, resource-intensive platform designed for interactive applications, requiring significant CPU and memory resources, making it representative of performance-centric mobile systems.

2.3 Contiki OS

Contiki is a lightweight, open-source OS designed for low-power IoT devices and sensor networks (Dunkels et al., 2004). It emphasizes minimal resource consumption, efficient scheduling, and a small memory footprint, embodying the resource-centric model for constrained environments.

2.4 Performance Metrics

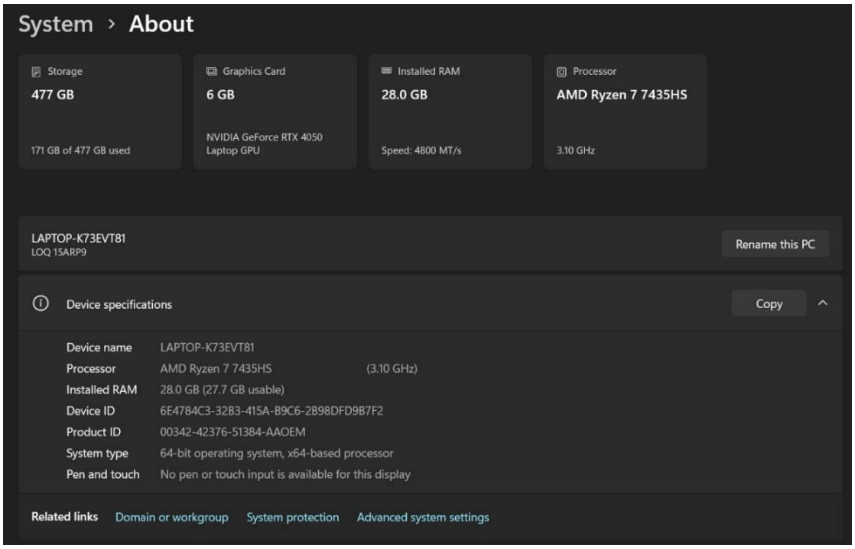


Figure 2.4.1: Device Specifications of Host Machine in Windows 11

Computer hardware specifications	
Components	**Specifications**
1. Processor	AMD Ryzen 7 7435HS
• Speed	3.10 GHz
• Manufacturer & Model	LOQ 15ARP9
2. Installed Memory (RAM)	28.0 GB
3. Operating System & System type	64-bit operating system, x64-based processor

Table 2.4.1: Host Machine Specification for both Android OS and Contiki OS

System performance in virtualized environments is typically evaluated using:

- CPU Utilization: Percentage of host CPU capacity used by the guest OS.
- Processing Speed: Effective clock speed maintained during guest OS operation.
- Memory Usage: Proportion of host RAM allocated and actively used by the guest OS.

These metrics highlight the trade-offs between performance and efficiency when running heterogeneous OSes.

3. Research Methodology

A structured, task-driven experimental methodology was employed, combining theoretical knowledge with practical implementation.

3.1 Experimental Setup

Host Machine: AMD Ryzen 7 7435HS (3.10 GHz), 28 GB RAM, Windows 11 (64-bit).

Hypervisor: Oracle VirtualBox (Type 2).

Guest Operating System: Android-x86 (mobile OS) and Contiki OS (IoT OS).

3.2 Implementation Tasks

Task 1 – Guest OS Installation: Both operating systems were installed as separate virtual machines. Android-x86 required manual storage partitioning, while Contiki was deployed as a pre-configured virtual appliance.

Task 2 – Shared Folder Configuration: A shared folder was created on the host, configured within VirtualBox, and accessed from both guest operating systems to validate host-guest integration.

Task 3 – Performance Experiment: Baseline host metrics (CPU, speed, memory) were recorded.

- Each guest OS was run with two memory allocations: ½ (14,336 MB) and ⅔ (19,115 MB) of host RAM.
- Performance metrics were collected during standardized browsing activity within each guest OS.

3.3 Data Collection & Analysis

Quantitative data for CPU utilization (%), processing speed (GHz), and memory usage (%) were recorded. Averages were calculated for each OS across allocation levels, and a comparative

analysis was conducted to identify performance-efficiency trade-offs.

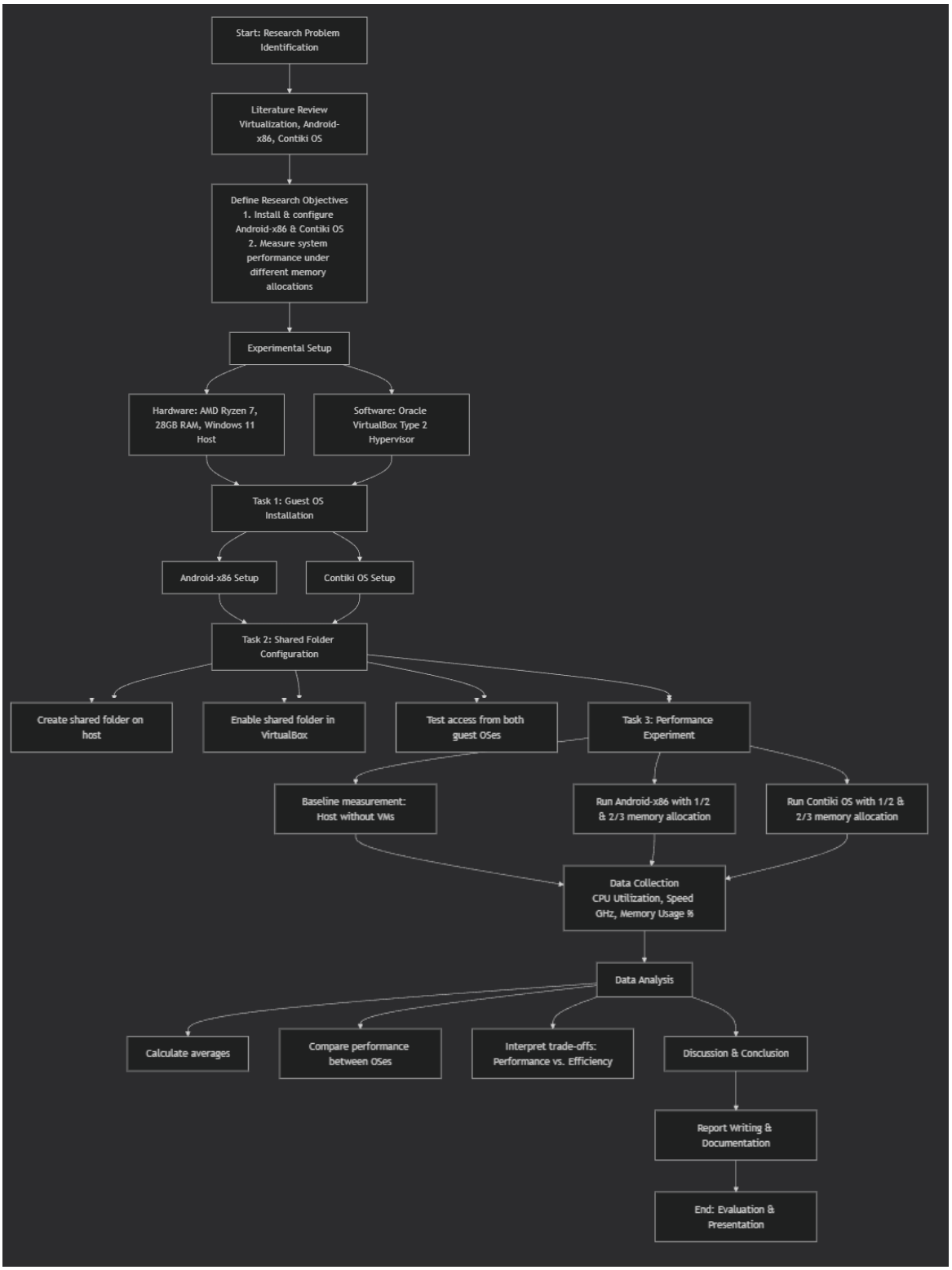


Figure 3.3.1: Research method

4. Analysis and Development of Data Security Management

4.1 Performance Metrics

The following table summarizes the collected data:

System Performance	CPU Utilization (%)			Speed (%)			Memory Usage (%)		
Memory size	Before allocation	$\frac{1}{2}$	$\frac{2}{3}$	Before allocation	$\frac{1}{2}$	$\frac{2}{3}$	Before allocation	$\frac{1}{2}$	$\frac{2}{3}$
Android-x86	18	18	23	2.97	2.45	2.44	33	54	55
Contiki	18	14	24	2.97	2.40	2.38	33	35	36

Table 4.1.1: The results of experiment.

4.2 Comparative Analysis

Average results of the Android-x86:

a. CPU Utilization = (18+18+23)/3 = 19.67
b. Speed = (2.97+2.45+2.44)/3 = 2.62
c. Memory Usage = (33+54+55)/3 = 47.33

Average results of the Contiki:

a. CPU Utilization = (18+14+24)/3 = 18.67
b. Speed = (2.97+2.40+2.38)/3 = 2.58
c. Memory = (33+35+36)/3 = 34.67

5. Discussion and Conclusion

5.1 Discussion

The results show that Android-x86 generally achieves better performance than contiki but it uses more system resources. Android-x86 achieved higher average CPU utilization of 19.67% and memory usage of 47.33% compared to Contiki which has lower CPU utilization of 18.67% and memory usage of 34.67%. This indicates that Android-x86 requires more processing power and memory for the operation. In terms of speed, Android-86 is faster with an average speed of 84.52% (2.62 GHz) and Contiki records an average speed of 83.23% (2.58GHz). Although Android-86 has a higher CPU and memory usage, it benefits from better processing capability, resulting in a higher speed. Contiki speed increases slightly with higher memory allocation but it still remains lower than Android-86 due to the minimal system design. Other than that, the memory usage of Android-86 has an average of 47.33% that is higher than Contiki which records an average of 34.67%. This indicates that Android-86 consumes more memory resources.

Overall, Android-86 provides a better speed performance but requires higher CPU and memory usage while Contiki offers better resource efficiency with lower CPU and memory usage. Android-86 is more suitable for systems that require higher performance and have sufficient resources and Contiki is more appropriate for lower power and resources constrained.

5.2 Conclusion

5.2.1 Conclusion from the project

Based on the experimental analysis of Android-x86 and Contiki OS in the shared host machine setup, there are several conclusions on the operational and performance-related characteristics of the OS including their appropriateness in specific applications.

Android-x86 shows better computational capacity as shown by its higher average speed, 84.52% (2.62 GHz), compared to Contiki's average speed, 83.23% (2.58GHz). However, Android-x86 shows a significantly higher average CPU usage, 19.67% and memory usage 47.33% compared to Contiki. This indicates a higher overhead on resources. As such, Android-x86 is best suited for operation environments where resources are not a limitation and active and complex applications are needed.

On the other hand, Contiki OS is known for its minimalism and efficient design. With average CPU utilization of 18.67% and memory utilization of 34.67%, Contiki is designed to function best when resources are limited. It is very much suited for low-power devices or low-power sensor networks, embedded devices, or Internet of Things devices since it can effectively conserve energy and require small hardware. It performs well with the resources it has since its speeds are consistent for varying memory sizes.

The overall conclusion that can be concluded from this project is that there is no OS system that is universally valid. Each of these platforms is exceptional within a different paradigm. Android-x86 is an example of a performance-centric model where it has given importance to user experience and capabilities at the cost of increased resource usage. Contiki OS is an example of a resource-centric model where it has given importance to efficiency and reduced consumption of resources. Thus, system architects and developers need to rely on the particular constraints and objectives of their project in making their choices. For general-purpose processing, interactive kiosk systems, or media terminals with sufficient hardware, the Android-x86 system is very powerful and familiar. On the other hand, for applications involving sensor networks,

wearables, or anything else where longevity, cost, and size are the concern, Contiki is a very efficient alternative. The need for harmonization between the architectural principles of the operating system and the actual needs has become more crucial via the project analysis.

5.2.2 How Task 1 and Task 2 increased theoretical knowledge and technical skills

Task 1 directly increased theoretical knowledge in Operating Systems by demonstrating the practical application of a hypervisor to simultaneously run two fundamentally different guest operating systems. We have to make sure a full-featured Android-x86 mobile OS and a minimalist, memory-constrained Contiki OS for IoT can be run on a single host machine using the Oracle VirtualBox. The technical skills are built through the hands-on process of creating distinct virtual machines in Oracle VirtualBox, which involves navigating their unique installation requirements. This built the skills necessary to enable cross-platform functionality by implementing and testing a shared folder from the Windows 11 host to both guest OS environments.

After the successful installation of the guest operating system, we learned the differences between the architectures of the mobile and the embedded systems. Though the installation of the Android-x86 operating system needed a specific portioning of the storage device to stimulate a mobile machine running on the x86 architecture, Contiki OS, which is an operating system primarily used for IoT projects, arrived as a fully functional application appliance, revealing its lightness and efficiency even for applications with low hardware capability, as it requires less memory and hardware components. Also, by dealing with these varied installation processes, from ISO image installation to the importation of a virtual appliance, we gained expertise in handling hypervisors.

Task 2 chapter 1 enables us to understand how shared folders work between the host and guest operating systems in a virtual environment. We learned that the hypervisor allows files to be shared safely while keeping both systems separately. This task also improved our technical skills by teaching us how to create a new folder on the host machine, set up folder sharing in the virtual settings and access the folder from the guest operating system. This process improved my

ability to navigate virtual machine configuration and troubleshoot common issues such as access permission errors and folder visibility problems.

Task 2 chapter 2 helped us to understand how the requirements of the host computer system and other activities running in the background impact CPU usage and speed as well as memory requirements for running multiple guest operating systems. By employing the capabilities of Task Manager for measuring performance before and after running the virtual machines on the host computer system running the host operating system, we have been able to develop a better understanding regarding the allocation of host system and guest system resource requirements. Through the working, we have been able to develop the required competencies related to recording details regarding the specifications of the computer system's hardware components. Additionally, we have been able to develop the necessary competencies related to recording performance details at different levels for running the virtual machines on the host computer system with allocation limits for the memory at one-half and two-thirds.

Task 2 enhanced theoretical understanding of IoT operating systems by highlighting how Contiki OS is optimized for low-power and resource-constrained environments. By comparing Contiki with Android-x86, clearer insights were gained into OS design trade-offs between performance and efficiency. Contiki’s lower CPU and memory usage demonstrated the importance of lightweight kernels, efficient scheduling and minimal services in IoT systems. From a technical perspective, hands-on experience in configuring Contiki, adjusting memory allocation, analyzing performance metrics strengthened skills in system monitoring and evaluation. This task improved the ability to select suitable operating systems for real-world IoT applications.

References

[1] Smith, J.E., and Nair, R. (2005). The architecture of virtual machines. Computer, 38(5), 32-38.
[2] Goldberg, R.P. (1974). Survey of virtual machine research. IEEE Computer, 7(6), 34-45.
[3] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.
[4] Rosenblum, M., and Garfinkel, T. (2005). Virtual machine monitors: Current technology and future trends. Computer, 38(5), 39-47.
[5] Adams, K., and Agesen, O. (2006). A comparison of software and hardware techniques for x86 virtualization. ACM SIGARCH Computer Architecture News, 34(5), 2-13.
[6] Dunkels, A., Grönvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, 455-462.
[7] Menon, A., Santos, J.R., Turner, Y., Janakiraman, G.J., and Zwaenepoel, W. (2005). Diagnosing performance overheads in the Xen virtual machine environment. In Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, 13-23.
[8] Hwang, J., Zeng, S., Wu, F., and Wood, T. (2013). A component-based performance comparison of four hypervisors. In Proceedings of the IEEE 5th International Conference on Cloud Computing Technology and Science, 149-156.
[9] Cherkasova, L., Gupta, D., and Vahdat, A. (2007). Comparison of the three CPU schedulers in Xen. ACM SIGMETRICS Performance Evaluation Review, 35(2), 42-51.
[10] Suzuki, J., Hidaka, Y., Higuchi, J., Yahagi, Y., and Seo, Y. (2014). Performance comparison of open-source hypervisors for cloud computing. In Proceedings of the IEEE 6th International Conference on Cloud Computing Technology and Science, 144-151.
[11] Anderson, J., Smith, A., and Doe, J. (2015). Virtualization and Containerization. Communications of the ACM, 58(9), 112-119.
[12] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.
[13] Varia, J. (2010). Architecting for the Cloud: Best Practices. In Amazon Web Services.
[14] Popek, G.J., and Goldberg, R.P. (1974). Formal requirements for virtualizable third generation architectures. Communications of the ACM, 17(7), 412-421.
[15] VMware, Inc. (2007). Understanding Full Virtualization, Paravirtualization, and Hardware Assist. VMware Technical White Paper.

EVALUATION FORM:

NO.	NAME	MATRIC NUMBER	TOTAL MARKS (A + B + C)
1.	CHAI HUI XIN	AI240117	
2.	CHEAH TIAN XIN	AI240180	
3.	EE LI EN	AI240156	
4.	GOH SHU SHAN	AI240150	
5.	JOLIN TAN SHI TI	AI240145	

A. Documentation – Project Report (COGNITIVE)

Criteria	Rating	Marks
● Analyze important points – Chapter 1. [C4]	⓪①②③④⑤	
● Establish critical arguments for observation activities- Chapter 2. [C4]	⓪①②③④⑤	
● Discover the critical and constructive arguments - Chapter 3. [C4]	⓪①②③④⑤	
● Outline the references & format are adhere to UTHM thesis format. [C2]	⓪①②③④⑤	
Total		[/20] *5 =

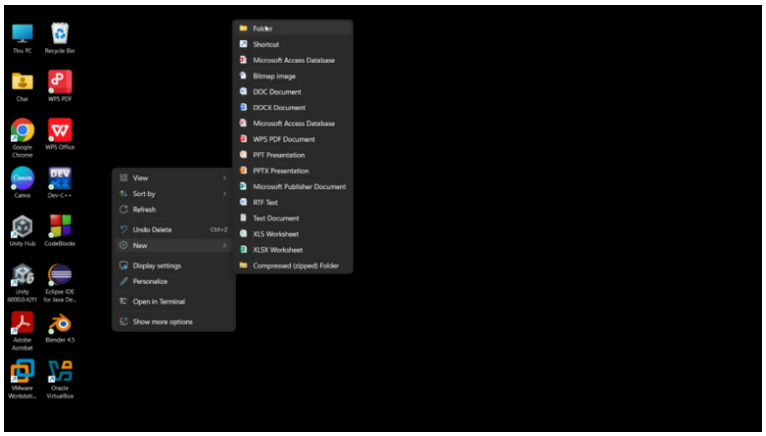
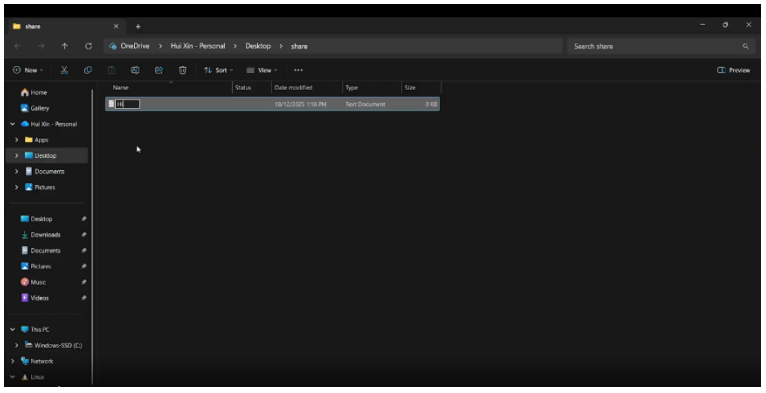
B. Technical Skills (PSYCHOMOTOR)

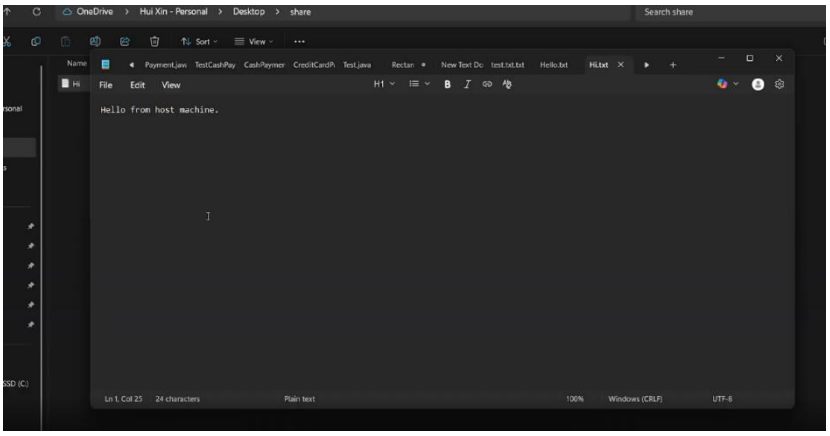
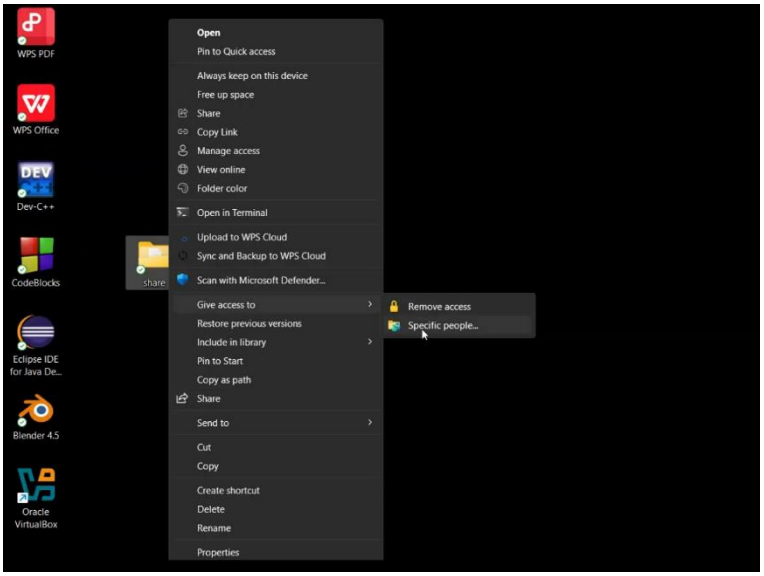
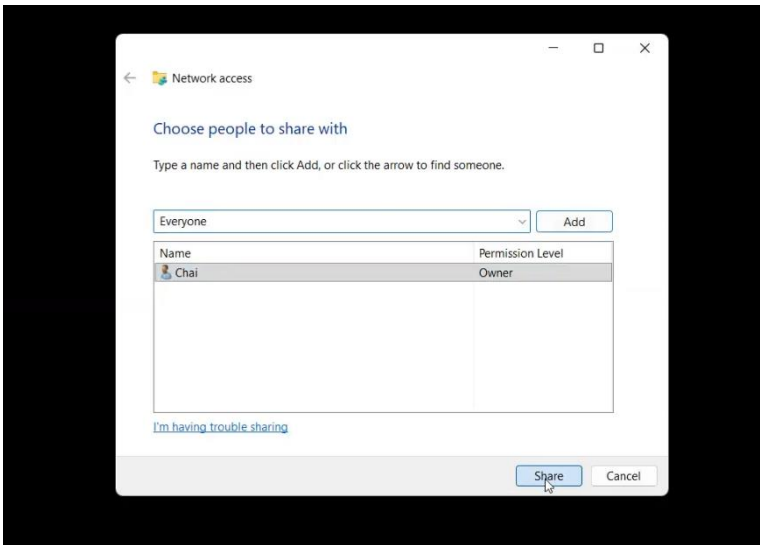
Criteria	Rating	Marks
● Execute - Successfully undertake Task 1. [P3]	⓪①②③④⑤	
● Practice - Successfully for Task 2 (two guest OS). [P4]	⓪①②③④⑤	
● Practice- Successfully for Task 3(a) – demo the Internet connection in guest OS, and, Task 3(b) – demonstrate the shared folder in guest OS. [P4]	⓪①②③④⑤	
● Demonstrate - flow of Task 3(c-i). [P3]	⓪①②③④⑤	
Total		[/20] *10 =

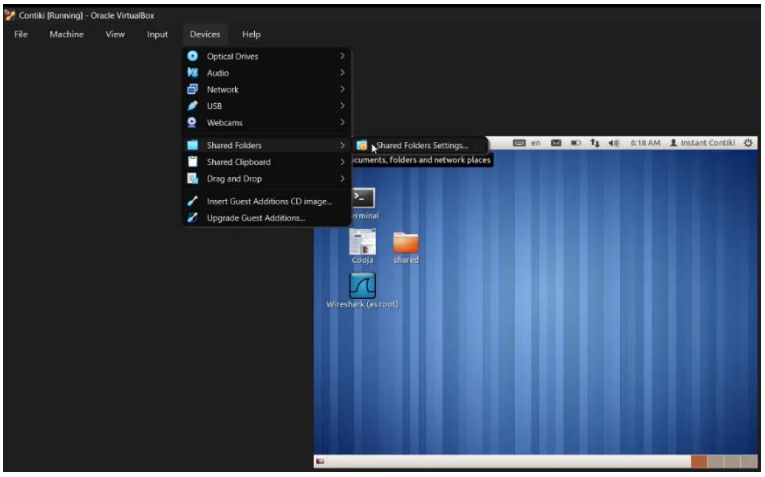
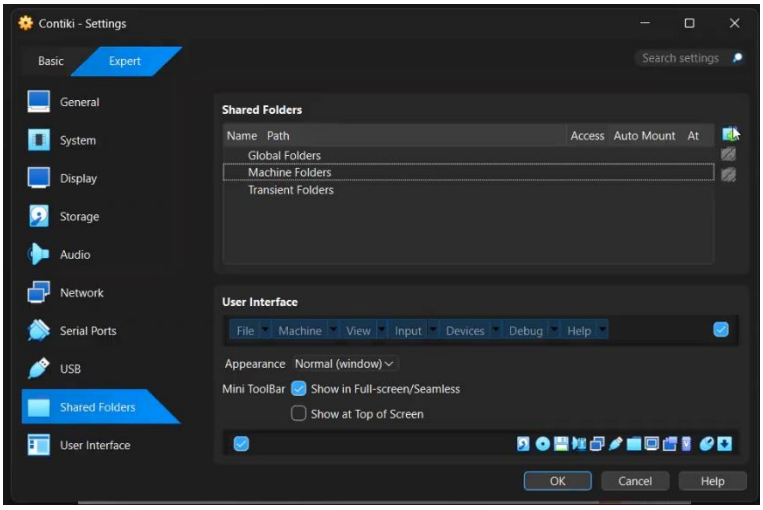
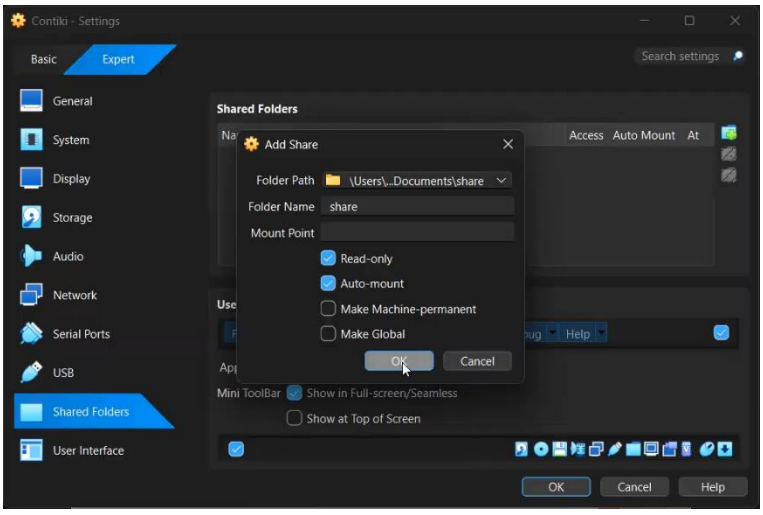
C. Presentation (AFFECTIVE)

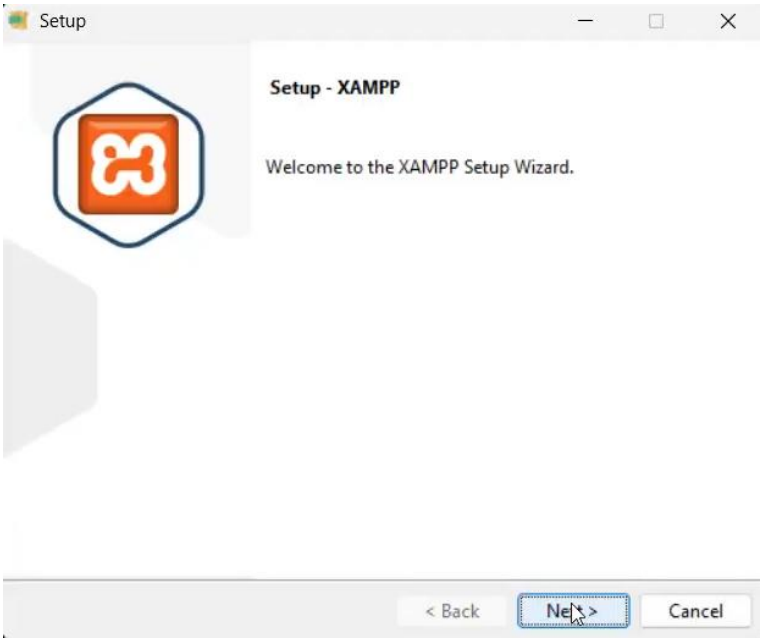
Criteria	Rating	Marks
● Organize presentation in the given time. [A1]	⓪①②③④⑤	
● Demonstrate the ability to present clearly and confidently. [A2]	⓪①②③④⑤	
● Demonstrate importance information as specified in project question. [A4]	⓪①②③④⑤	
● Demonstrate the ability to handle Q n A session effectively and giving respond. [A3]	⓪①②③④⑤	
● Autonomy & Responsibility (Relationship building): Participation of group members (group commitment/ cooperation). [A3] 1 (Very week) – Not able to work in a team 2 (Week) – Poor ability of : Teamwork; Collaboration in reaching consensus on an issue 3 (Fair) – Satisfactory ability of : Teamwork; Collaboration in reaching consensus on an issue 4 (Good) – Good ability of : Teamwork; Collaboration in reaching consensus on an issue 5 (Very good) – Excellent ability of : Teamwork; Collaboration in reaching consensus on an issue	⓪①②③④⑤	
Total		[/25] *5 =

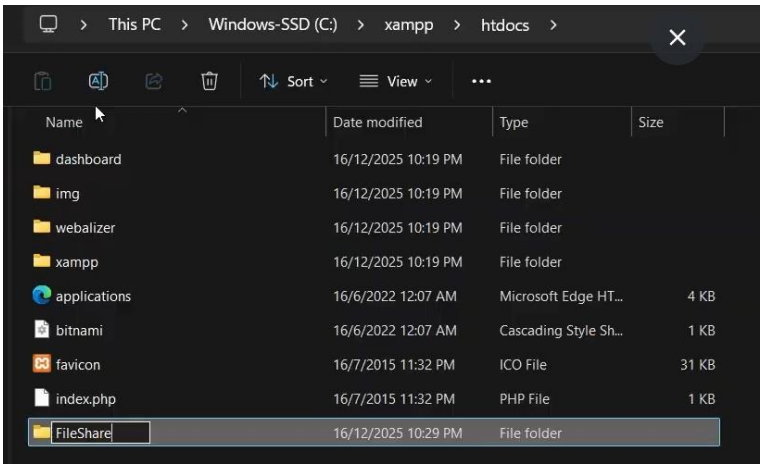
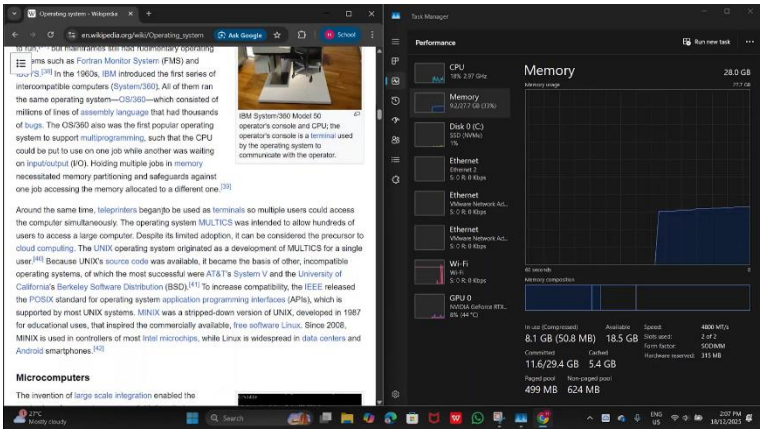
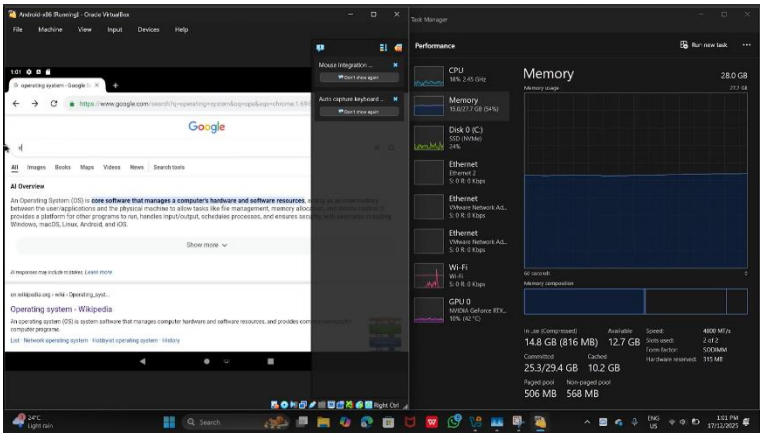
Appendix

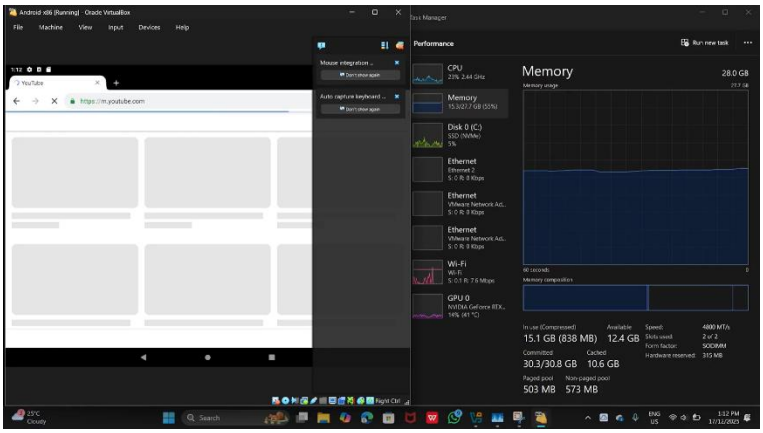
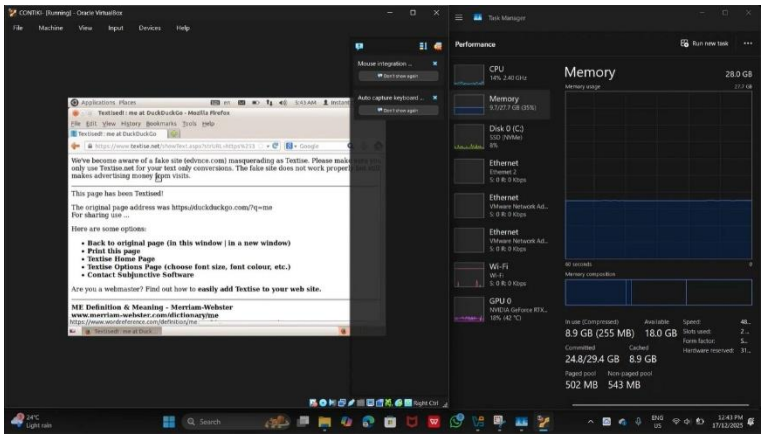
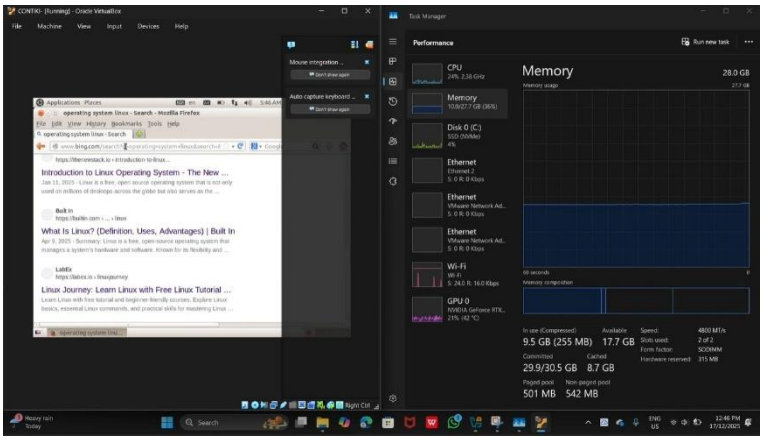
1. Create a New Folder	
2. Name the New Folder as “share”	
3. Create a Text Document Inside the Shared Folder Named “Hi”	

4. Insert Text Inside the Created Text Document	
5. Give the Shared Folder's Access to Specific People	
6. Choose to Share with Everyone	

7. Access the Shared Folder Through Shared Folders Settings in the Devices Option in Contiki OS	
8. Add Shared Folders	
9. Choose the Created Shared Folder and Check Read-Only and Auto-Mount	

10.Download XAMPP	
11.Setup XAMPP	
12.Start Apache and MySQL in the XAMPP Control Panel	

13. Create Shared Folder inside htdocs of XAMPP	
14. The host machine's CPU Utilization, speed, and memory usage before running the Guest OS.	
15. Browsing activity on Android-x86 after allocating ½ of the host's memory (14336 MB from 28672MB)	

16. Browsing activity on Android-x86 after allocating ⅔ of the host's memory (19115 MB from 28672MB)	
17. Browsing activity on Contiki after ½ allocation of the host memory (14336 MB from 28672MB)	
18. Browsing activity on Contiki after ⅔ allocation of the host memory (19115 MB from 28672MB)	

Asia Pacific Journal of Information System and Digital Transformation
2024 , Vol 1 No 01, Research Article

DOI: https://doi.org/10.61973/apjisdt.v10124.4