

# **A Novel Framework for Lab Exam Integrity Using Netbooting**

## **Mini Project Report**

Submitted to the APJ Abdul Kalam Technological University in  
partial fulfilment of the requirements for the award of the Degree of

## **Bachelor of Technology**

in

## **Computer Science and Engineering (Cyber Security)**

By

Abhinav Manoj

PTA22CC003

Alen Mathai

PTA22CC011

Anakha MR

PTA22CC015

Sharon Aliyas Johnson

PTA22CC058

**Under the guidance of**

Ms. Chithra Shaji Thomas

(Assistant Professor, Department of Computer Science and Engineering  
(Cyber Security))



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
(CYBER SECURITY)**

**COLLEGE OF ENGINEERING KALLOOPPARA  
KERALA**

**April 2025**

# DECLARATION

We undersigned hereby declare that the project report **A Novel Framework for Lab Exam Integrity Using Netbooting** submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Ms. Chithra Shaji Thomas, Assistant Professor**. This submission represents our ideas in our own words and ideas or words of others have been included where We have adequately and accurately cited and referenced the original sources. We also declare that We have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the university and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other university.

Place: Kalliooppara

Date: April 2025

Abhinav Manoj  
Alen Mathai  
Anakha MR  
Sharon Aliyas Johnson

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (CYBER SECURITY)

COLLEGE OF ENGINEERING KALLOOPARA  
PATHANAMTHITTA- 689603

## CERTIFICATE



This is to certify that the project design report entitled **A Novel Framework for Lab Exam Integrity Using Netbooting** submitted by **Abhinav Manoj** (PTA22CC003), **Alen Mathai** (PTA22CC011), **Anakha M R** (PTA22CC015), and **Sharon Aliyas Johnson** (PTA22CC058) in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology in Computer Science and Engineering (Cyber Security)** of **APJ Abdul Kalam Technological University** is a bonafide work carried out by them under our guidance and supervision. The report in any form has not been submitted to any other University or Institute for any purpose.

### Coordinator

Ms. AnanthaLakshmi M V

Assistant Professor  
Department of  
Computer Science &  
Engineering  
(Cyber Security)

### Guide

Ms. Chithra Shaji Thomas

Assistant Professor  
Department of  
Computer Science &  
Engineering  
(Cyber Security)

### Head of the Department

Mr. Raj Kumar T

Professor  
Department of  
Computer Science &  
Engineering  
(Cyber Security)

# ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and sincere thanks to everyone who helped us to complete this work successfully. We express our sincere thanks to **Dr. Deepa J**, Principal, for the constant support and help. We also thank **Mr. Raj Kumar T**, Head of Department Computer Science and Engineering (Cyber Security), College of Engineering Kalloopara for providing us with all the necessary facilities and support.

We would like to express our sincere gratitude to the department faculty for their support and cooperation. We would like to place on record our sincere gratitude to our project guide **Ms. Chithra Shaji Thomas**, Assistant Professor, Department of Computer Science and Engineering (Cyber Security), College of Engineering Kalloopara for the guidance and mentorship throughout the course.

Finally, We thank our family, and friends who contributed to the successful fulfilment of this project work.

Abhinav Manoj  
Alen Mathai  
Anakha MR  
Sharon Aliyas Johnson

# Abstract

We present a secure and scalable framework for maintaining integrity in lab-based examinations through the use of network booting (netbooting), addressing the growing need for controlled and tamper-resistant exam environments in academic institutions. Traditional exam setups often face configuration inconsistencies, security vulnerabilities, and administrative overhead. This system leverages netbooting via the Linux Terminal Server Project (LTSP) to deploy uniform, non-persistent operating system environments across all client machines, eliminating the risk of local data manipulation and ensuring standardized configurations. With all systems booting from a centralized image, administrators gain complete control over the environment, while real-time monitoring ensures transparency and accountability throughout the examination session. Secure, encrypted communication between client systems and the server further enhances confidentiality. As educational institutions face increasingly sophisticated threats and logistical challenges in conducting digital assessments, this project offers a practical and future-ready solution that upholds exam integrity, reduces operational complexity, and supports flexible deployment across varied scales.

# Table of Contents

<b>Table of Contents</b> .....	2
<b>List of Figures</b> .....	3
<b>List of Abbreviations</b> .....	4
<b>Chapter 1: Introduction</b> .....	5
1.1 Problem Statement.....	5
1.2 Project Overview.....	5
1.3 System Workflow .....	6
1.4 Target Audience.....	6
1.5 Project Objectives.....	7
1.6 Project Scope .....	7
<b>Chapter 2: Background and Literature Review</b> .....	8
2.1 Introduction .....	8
2.2 Review of Related Literature .....	8
2.2.1 Secure Online and Lab-Based Examination Systems.....	8
2.2.2 Remote Labs and Virtual Infrastructure .....	8
2.2.3 Diskless Workstations and LTSP Implementation .....	8
2.2.4 Lab Management and Monitoring Tools.....	9
2.3 Synthesis and Relevance to the Project .....	9
<b>Chapter 3: Methodology</b> .....	10
3.1 Overview .....	10
3.1.1 PXE Booting .....	10
3.1.2 PXE Boot Process .....	10
3.1.3 iPXE .....	11
3.1.4 PXE Chainloading .....	12
3.1.5 Components Involved in PXE and iPXE Boot Process.....	13
3.1.6 Chainloading Process .....	14
3.2 Linux Terminal Server Project.....	15
3.2.1 Architecture .....	15
3.2.2 Client Boot Process.....	15
3.2.3 Diskless Operation .....	16
3.2.4 Centralized Management .....	16
3.2.5 Resource Sharing .....	16
3.2.6 Customization and Scalability .....	16
3.2.7 Fat Clients .....	16
3.2.8 Enhanced Processing Power .....	16
3.2.9 Network Boot with Local Execution .....	17
3.2.10 Hybrid Approach.....	17
3.2.11 Community and Support.....	17

3.3 Display Servers .....	17
3.3.1 X.Org Server (Xorg).....	17
3.3.2 Wayland .....	19
3.4 DHCP Servers.....	20
3.5 LTSP Networking.....	21
<b>Chapter 4: Implementation</b> .....	23
4.1 Setting Up LTSP .....	23
4.1.1 Installation of LTSP Server Components .....	25
4.1.2 Building Client Images.....	25
4.2 Custom Client Image Creation.....	26
4.3 Multiple DHCP Servers .....	27
4.3.1 On-board Network Stack / NIC .....	28
4.3.2 Chainloaded iPXE Program .....	28
4.3.3 Netbooted Linux Kernel .....	28
4.4 Configuring dnsmasq for LTSP.....	29
4.5 Real-time Monitoring and Logging .....	30
4.6 Web Application Implementation .....	31
4.6.1 Teacher Interface .....	31
4.6.2 Student Interface.....	32
4.6.3 Security and Session Control .....	33
4.7 Deployment .....	33
<b>Chapter 5: Results and Screenshots</b> .....	34
5.1 System Overview .....	34
5.2 Teacher Interface Output .....	34
5.2.1 Login Page .....	34
5.2.2 Exam Dashboard.....	35
5.2.3 Question Creation Interface.....	35
5.3 Student Interface Output .....	36
5.3.1 Login Page .....	36
5.3.2 Exam Interface.....	36
5.4 Monitoring and Control Output.....	36
5.4.1 Static IP Table.....	36
5.4.2 Epopetes Monitoring .....	37
5.5 Summary and Observations.....	38
<b>Chapter 6: Conclusion</b> .....	39
<b>Chapter 7: Future Scope</b> .....	40

# List of Figures

Figure 3.1:	PXE Booting Process .....	11
Figure 3.2:	iPXE Chainloading and Netbooting a Linux kernel .....	13
Figure 3.3:	Architecture of X.Org Server.....	18
Figure 3.4:	Wayland architecture. ....	20
Figure 4.1:	iPXE boot menu .....	27
Figure 4.2:	An isolated network with a secondar DHCP server and LTSP clients ....	28
Figure 4.3:	LTSP server that also act as a DHCP server .....	29
Figure 5.1:	Teacher Login Page.....	34
Figure 5.2:	Teacher Dashboard: Live Monitoring of Exam Status .....	35
Figure 5.3:	Question Entry Interface .....	35
Figure 5.4:	Student Login Page .....	36
Figure 5.5:	Exam Interface: Assigned Question, Timer, and Submission .....	36
Figure 5.6:	Student IP and Submission Status Overview .....	37
Figure 5.7:	Epoptes: Real-Time Classroom Monitoring Tool .....	37



# List of Abbreviations

<b>LTSP</b>	Linux Terminal Server Project
<b>PXE</b>	Preboot Execution Environment
<b>iPXE</b>	Improved Preboot Execution Environment
<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>TFTP</b>	Trivial File Transfer Protocol
<b>GUI</b>	Graphical User Interface
<b>NIC</b>	Network Interface Card
<b>VM</b>	Virtual Machine
<b>OS</b>	Operating System
<b>NFS</b>	Network File System
<b>DNS</b>	Domain Name System
<b>ACL</b>	Access Control List
<b>IP</b>	Internet Protocol
<b>URL</b>	Uniform Resource Locator
<b>HTML</b>	HyperText Markup Language
<b>Flask</b>	A Python-based micro web framework
<b>Epopotes</b>	A classroom management and monitoring tool
<b>GNU</b>	GNU's Not Unix
<b>HTTP</b>	HyperText Transfer Protocol
<b>SSH</b>	Secure Shell

# Chapter 1: Introduction

## 1.1 Problem Statement

As educational institutions increasingly integrate digital technologies into learning environments, the need for secure, scalable, and manageable examination systems becomes more critical than ever. Traditional methods of conducting lab-based exams often rely on manually configured machines, localized operating systems, and minimal supervision. This approach introduces several challenges:

- Inconsistent software environments across systems
- Difficulty in enforcing access restrictions
- High administrative burden in setting up and maintaining exam machines
- Elevated risks of cheating, tampering, or system failure

These limitations are not only impractical at scale, but they also compromise academic integrity and operational efficiency.

## 1.2 Project Overview

This project introduces a unified, secure framework for conducting lab-based examinations using netbooting technologies and centralized control. At the core of the system is a centralized server, powered by the Linux Terminal Server Project (LTSP) and PXE (Preboot Execution Environment), which allows client machines in a lab to boot over the network from a clean and standardized image.

This netbooted approach creates a non-persistent client environment. Each exam session starts fresh, without cached data or prior modifications — effectively eliminating the potential for unauthorized software, saved files, or configuration discrepancies.

The system further integrates a custom-built Flask web application that provides teachers and administrators with a graphical interface to:

- Create and manage exams
- Add questions
- Monitor student activity
- Track submissions

In addition, real-time monitoring capabilities are enabled through Eproptes, allowing invigilators to view all student screens, broadcast messages, or remotely control systems if needed.

## 1.3 System Workflow

The workflow of the examination system is as follows:

1. The LTSP server hosts pre-configured operating system images.
2. When clients are powered on, they PXE-boot from the server.
3. The environment is loaded into memory, ensuring non-persistent sessions.
4. Students log in using their register numbers through the web interface opened in Firefox.
5. The Flask web app fetches the assigned question and starts the timer.
6. Students upload their answers before the timer expires.
7. On the server side, the teacher can monitor status, view answers, and manage the exam in real-time.
8. Eprotes provides screen viewing and system control to uphold discipline during the exam.

The architecture does not rely on browser kiosk modes or third-party lockdown tools. Instead, it leverages OS-level isolation and boot-time integrity to create a secure examination environment.

## 1.4 Target Audience

This solution is designed for:

- **Educational Institutions** — Schools, colleges, and universities conducting lab-based exams in a controlled environment.
- **Teachers and Administrators** — Who require a simplified way to manage exams, create papers, and monitor submissions.
- **System Administrators** — Who want centralized control, minimal maintenance overhead, and scalability without complexity.
- **Students** — Who benefit from a consistent, fair, and distraction-free exam experience.

Whether it's a small classroom or a full-scale exam hall, this system adapts to various deployment scales with consistent performance and strong security measures.

## 1.5 Project Objectives

The core objectives of this project are:

- **Security:** Provide a non-persistent, tamper-proof environment for each exam session.
- **Centralization:** Reduce administrative overhead with a single, manageable server image.
- **Efficiency:** Allow fast setup, reboot, and reset of machines for back-to-back exams.
- **Transparency:** Enable live monitoring and logging of student activity during exams.
- **Scalability:** Handle deployments from small labs to large institutions with minimal performance loss.

## 1.6 Project Scope

The scope of this project encompasses:

- Designing a centralized server environment using LTSP and PXE for non-persistent netbooting
- Building a Flask-based web app for teacher-side exam management
- Implementing real-time monitoring using Epopotes
- Restricting client access to only authorized interfaces and resources
- Managing static IPs and user sessions for network stability
- Providing flexibility in deployment sizes with a focus on ease-of-use and maintainability

Security and performance are primary concerns throughout the architecture. Every aspect—from boot configuration to submission handling—is designed to prevent misuse while maintaining a smooth user experience.

# Chapter 2: Background and Literature Review

## 2.1 Introduction

In recent years, the adoption of centralized and thin client-based computing environments has gained traction in educational institutions as a solution to manage increasing student populations and the need for scalable digital infrastructure. Particularly in the context of examinations, where uniformity, control, and security are critical, solutions like the Linux Terminal Server Project (LTSP) and classroom management tools like Eoptes offer a compelling approach.

This chapter provides the contextual foundation for this project by highlighting key developments and studies in secure examination systems, diskless workstation deployments, and computer lab management methodologies. The focus is on existing technologies and their role in enhancing security, usability, and efficiency in academic environments.

## 2.2 Review of Related Literature

### 2.2.1 Secure Online and Lab-Based Examination Systems

As digital assessments become more widespread, so do concerns about cheating, content leakage, and unauthorized access. Several studies have attempted to address these issues through authentication, system hardening, and proctoring technologies.

Jegatha Deborah and Kannan [1] proposed a mutual authentication model to ensure the legitimacy of both student and server during online assessments. Fluck et al. [2] performed a comparative study across international institutions, focusing on strategies for preserving academic integrity through design choices in the exam environment, including local isolation and network restrictions.

These approaches align with the goals of this project, which emphasizes local network-based exams (lab-based rather than fully online) using non-persistent booting and real-time supervision.

### 2.2.2 Remote Labs and Virtual Infrastructure

With the shift to hybrid learning, many institutions began exploring virtual computer labs. Dashamir Hoxha [3] demonstrated a solution using cloud servers, VPN tunnels, and the Eoptes management suite to recreate a physical lab experience remotely. This setup enabled control over remote machines as if they were local — an idea that closely mirrors the control offered by Eoptes in our LAN-based lab examination system.

Although our approach uses physical clients in a lab, the real-time monitoring and intervention capabilities are shared with these remote lab setups.

### 2.2.3 Diskless Workstations and LTSP Implementation

Diskless workstations reduce maintenance and increase control by booting from a central image. This is especially useful in examination contexts, where a clean, controlled envi-

ronment is essential.

A paper published in the International Journal of Engineering Research and Technology [4] presented an LTSP-based solution where virtualized clients boot from a pre-configured image. This architecture is mirrored in our project setup, where different OS images are maintained centrally and assigned dynamically using PXE boot and a Flask-based control panel.

#### **2.2.4 Lab Management and Monitoring Tools**

Centralized lab monitoring tools like Eproptex enable instructors to supervise students' systems, broadcast screens, and issue system commands remotely. According to a report by Interoperable Europe [5], over 500 Greek schools use Eproptex for real-time classroom management. This shows the tool's reliability and scalability in educational contexts.

Our project leverages Eproptex not only for classroom management but as a core part of the exam integrity system, allowing proctors to observe, assist, or take control of any client during the exam.

### **2.3 Synthesis and Relevance to the Project**

The common thread among the reviewed literature is the emphasis on security, centralization, and usability in educational IT systems. This project combines key ideas from each of the aforementioned domains: leveraging LTSP for non-persistent booting, Eproptex for oversight, and a custom Flask web application for managing exams and static IP configurations.

By situating this work within proven frameworks and open-source solutions, the system benefits from established stability and real-world applicability while also introducing new functionality tailored to lab-based examinations — a niche less explored in contemporary research compared to online examinations.

## Chapter 3: Methodology

### 3.1 Overview

The project proposes a streamlined solution for conducting secure online examinations, utilizing the capabilities of the Linux Terminal Server Project (LTSP) to eliminate the traditional complexities involved in setting up individual operating systems on each client machine. By adopting a centralized netbooting approach, administrators can effortlessly deploy uniform, kiosk-mode-enabled fat clients throughout the lab environment. This significantly reduces setup time and resource requirements while ensuring consistency and standardization across all exam terminals.

The integration of a dedicated management layer, developed using Flask, further simplifies administration by providing an intuitive web-based interface for managing exams, configuring clients, handling user authentication, and allocating system resources. This reduces the need for deep technical expertise and streamlines the management workflow.

In parallel, a real-time monitoring mechanism powered by Eproptics enables proactive oversight of client activity during examinations. This facilitates timely intervention in case of any anomalies or unauthorized behavior, thereby maintaining the integrity and fairness of the examination environment.

By combining the efficiency of LTSP netbooting, a user-friendly administrative interface, and robust monitoring tools, the project aims to deliver a secure, manageable, and scalable solution for online examinations that enhances both administrative control and user confidence.

#### 3.1.1 PXE Booting

The Preboot Execution Environment (PXE) is a widely used standard that enables client computers to boot over a network without relying on local storage devices. In this project, PXE booting acts as the foundation for deploying operating systems to client machines on-demand, ensuring a uniform and tamper-proof environment across all exam terminals. PXE allows the client system's network interface card (NIC), embedded with PXE firmware, to fetch boot-related data such as the kernel and initramfs directly from a server using protocols like DHCP and TFTP. This not only speeds up the deployment process but also eliminates the need to maintain operating systems individually on each machine.

#### 3.1.2 PXE Boot Process

The PXE boot process unfolds in several structured stages that ensure the client system can load an operating system securely from the server:

- **Initialization:** When the client system is powered on, PXE firmware takes control and sends a DHCP broadcast to request a network configuration and boot server information.
- **DHCP + TFTP:** The DHCP server is responsible for providing IP addresses to clients on the network. A TFTP (Trivial File Transfer Protocol) server, on the other hand, is responsible for transferring the necessary boot files between the server and client.

Some tools like dnsmasq can function as both a DHCP and TFTP server, combining the roles into a single lightweight service.

- **PXE Boot File:** Once the DHCP and TFTP communication is established, the PXE boot file—a small program provided by the server—is downloaded and executed by the client. This file is responsible for initiating the next phase of booting by loading the operating system kernel from the network.
- **Kernel Loading:** The bootloader fetched via PXE then loads the Linux kernel and `initramfs` from the server. These components are essential for starting the operating system.
- **Operating System Boot:** The client proceeds to boot into the downloaded environment. In this project, the client boots into a secure, exam-ready interface configured in kiosk mode, ensuring non-persistent and controlled access with no local data storage.

This structured, network-based approach guarantees a consistent environment on every client system, streamlining exam administration and significantly enhancing both security and manageability.

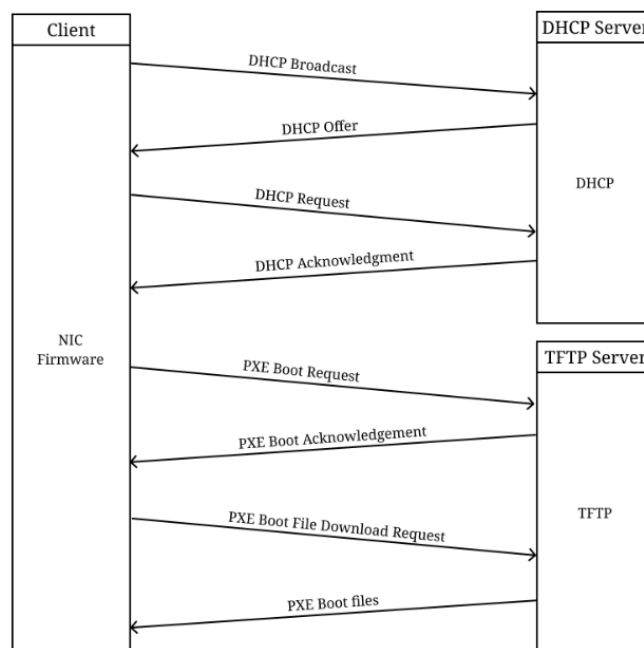


Figure 3.1: PXE Booting Process

### 3.1.3 iPXE

The use of **iPXE**, an open-source network boot firmware, offers a powerful alternative to the standard PXE (Preboot Execution Environment) implementation typically embedded in motherboard firmware. Compared to the traditional PXE, iPXE delivers advanced features, enhanced compatibility, and greater flexibility, making it particularly suited for complex and evolving network boot environments.



Key advantages of using iPXE include:

1. **Enhanced Functionality:** iPXE expands on standard PXE capabilities by supporting advanced boot methods such as HTTP, iSCSI, and FTP booting. It also enables the use of scripting to automate and customize the boot process, thereby increasing operational efficiency.
2. **Improved Compatibility:** Unlike manufacturer-provided PXE firmware, which may exhibit limitations with certain network or hardware configurations, iPXE is built to be widely compatible across various setups. This makes it more reliable in heterogeneous network environments.
3. **Greater Customization:** With support for powerful scripting, iPXE allows administrators to tailor the booting logic to specific needs, integrate dynamic configurations, and automate repetitive deployment tasks. This provides a fine degree of control over the booting workflow.
4. **Ease of Maintenance and Updates:** As an actively maintained open-source project, iPXE benefits from regular updates, security patches, and feature additions. Furthermore, it can be updated independently of the motherboard's firmware, simplifying version control and system maintenance.
5. **Support for Modern Technologies:** iPXE includes support for UEFI booting, ensuring compatibility with newer systems and firmware standards. This makes it adaptable to both legacy BIOS and modern hardware platforms.
6. **Performance Optimization:** By utilizing faster protocols such as HTTP for image delivery, iPXE can significantly reduce boot times and improve bandwidth usage—especially critical in large-scale exam environments with multiple concurrent clients.

In summary, iPXE offers a robust, customizable, and scalable booting mechanism, perfectly aligned with the requirements of a modern secure online examination system. Its flexibility and feature-rich environment empower administrators to build highly efficient and secure deployment infrastructures with minimal friction.

### 3.1.4 PXE Chainloading

PXE Chainloading refers to the technique of transitioning control from one bootloader to another during the network boot process. Within the context of PXE (Preboot Execution Environment) and iPXE, chainloading is commonly used to allow **iPXE**—a more advanced bootloader—to load and execute a secondary PXE bootloader such as `pxelinux.0` or similar components.

This method enables administrators to leverage the enhanced capabilities of iPXE, including scripting, advanced networking features, and support for a broader range of protocols (e.g., HTTP, iSCSI), while still utilizing the familiar PXE boot infrastructure. It provides a hybrid boot approach that enhances flexibility without requiring complete replacement of existing systems.

To avoid unintended infinite loops during the chainloading process, a detection safeguard is commonly implemented. This mechanism checks if iPXE has already been loaded. If it is detected that the current bootloader is iPXE, the system bypasses another chainload and directly proceeds to load the desired kernel or operating system image. This check ensures seamless and efficient continuation of the boot sequence, eliminating redundancy and preserving system performance.

In essence, PXE chainloading combines the power of iPXE with existing PXE workflows, enabling a more dynamic and robust booting experience while maintaining compatibility with legacy deployment strategies.

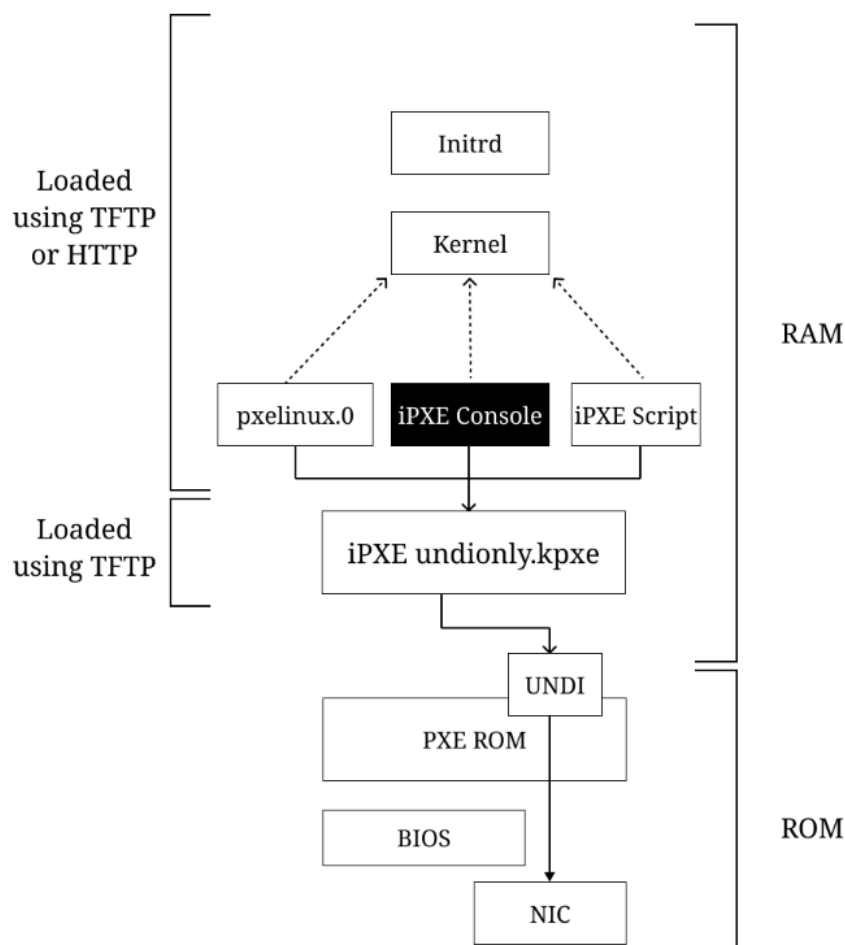


Figure 3.2: iPXE Chainloading and Netbooting a Linux kernel

### 3.1.5 Components Involved in PXE and iPXE Boot Process

#### 1. NIC (Network Interface Card)

The NIC initiates the boot process via network. It sends out a DHCP request to locate a PXE server.

## **2. BIOS**

The Basic Input/Output System initializes hardware components and passes control to the PXE ROM.

## **3. PXE ROM**

Firmware embedded in the NIC that understands how to perform a PXE boot.

## **4. UNDI (Universal Network Device Interface)**

A software layer used by PXE to abstract hardware-level communication.

## **5. iPXE undionly.kpxe**

A network bootloader loaded via TFTP which provides advanced network booting capabilities like HTTP boot and scripting.

## **6. iPXE Console**

The interface provided by iPXE to interact with boot scripts, PXE configurations, or manual commands.

## **7. pxelinux.0**

A secondary PXE bootloader often used to boot Linux systems, typically called by iPXE via chainloading.

## **8. iPXE Script**

Custom boot scripts that define the boot flow, such as loading a kernel and initrd via HTTP or TFTP.

## **9. Kernel**

The core of the operating system to be loaded into memory.

## **10. Initrd (Initial RAM Disk)**

A temporary root filesystem used during the system's initial boot phase.

### **3.1.6 Chainloading Process**

The chainloading process enables transitioning from PXE to iPXE, allowing for advanced boot features while maintaining compatibility with existing infrastructure. Below is a step-by-step breakdown of the process:

1. The computer powers on, and the BIOS is loaded from ROM.
2. The BIOS initializes hardware components, including the Network Interface Card (NIC).

3. The PXE ROM embedded in the NIC begins execution and searches for a TFTP server on the network.
4. Upon discovery, the TFTP server transmits the `undionly.kpxe` iPXE boot image to the client system.
5. The iPXE image is loaded into memory and begins execution.
6. An iPXE script is fetched and executed, which contains instructions for the boot process, such as specifying the kernel, `initrd`, and additional parameters.
7. The kernel and `initrd` are downloaded from the network server using supported protocols like TFTP or HTTP.
8. The kernel is loaded into memory and starts execution.
9. The `initrd` is mounted as a temporary root file system.
10. The kernel uses the contents of the `initrd` to initialize hardware and boot the complete operating system.

In addition to TFTP and HTTP protocols, iPXE also supports booting from a Network File System (NFS) share. This allows the kernel to be fetched directly from the NFS or even mount an NFS folder as its root filesystem, offering flexibility and scalability in deployment.

## 3.2 Linux Terminal Server Project

The Linux Terminal Server Project (LTSP) is an open-source solution that facilitates the deployment of thin client computing environments using Linux-based systems. LTSP enables multiple client machines to boot from a centralized server, granting users access to shared resources and applications over the network.

### 3.2.1 Architecture

LTSP follows a client-server model in which one or more central servers host the necessary services, applications, and user data. Client terminals, also referred to as thin or fat clients depending on configuration, connect to these servers via the network. The LTSP server typically runs a Linux distribution configured with LTSP services, while clients boot a minimal environment provided entirely by the server.

### 3.2.2 Client Boot Process

When an LTSP client is powered on, it begins a PXE (Preboot Execution Environment) boot sequence. During this process, the client acquires an IP address from a DHCP server and downloads required boot files from the LTSP server using the Trivial File Transfer Protocol (TFTP). The client then loads a lightweight Linux kernel and an initial RAM disk (`initrd`), which provides basic network connectivity and supports diskless operation.

### **3.2.3 Diskless Operation**

One of LTSP's standout features is diskless operation. Client terminals do not require local storage devices such as hard drives or SSDs. Instead, they rely entirely on the LTSP server for all file system resources and applications. This design minimizes hardware costs, reduces maintenance complexity, and improves security by eliminating the risk of data loss or unauthorized access through local storage.

### **3.2.4 Centralized Management**

LTSP simplifies system administration by offering centralized control over all connected client terminals. System administrators can configure client settings, deploy updates, manage user sessions, and monitor performance—all from the LTSP server. This centralized approach streamlines maintenance and minimizes administrative overhead.

### **3.2.5 Resource Sharing**

Through LTSP, applications and storage are hosted centrally, enabling multiple users to access the same resources concurrently. This model maximizes resource efficiency, promotes collaboration, and ensures consistency across client sessions. Centralized backups and storage also facilitate simplified data management and streamlined disaster recovery procedures.

### **3.2.6 Customization and Scalability**

LTSP is highly customizable and can be adapted to suit diverse environments. Administrators can configure the system to support different types of hardware, network configurations, and user needs. Moreover, LTSP is scalable—from small-scale classroom setups to large enterprise deployments—making it suitable for both modest and complex infrastructures.

### **3.2.7 Fat Clients**

Unlike thin clients, fat clients are conventional desktop systems or workstations equipped with local storage and computing capabilities. However, in the LTSP environment, fat clients also boot from the network but utilize their local hardware more extensively for application execution and system processes.

### **3.2.8 Enhanced Processing Power**

Fat clients typically feature more powerful CPUs, increased RAM, and greater storage capacity compared to thin clients. This allows them to run resource-intensive applications locally, offloading computational tasks from the LTSP server and improving performance for demanding use cases. Additionally, they are well-suited for scenarios where access to local peripherals or high-performance processing is required.

### 3.2.9 Network Boot with Local Execution

In an LTSP setup, fat clients initiate their boot process over the network in a manner similar to thin clients. They obtain boot files and the operating system kernel from the LTSP server. However, post-boot, fat clients execute applications and processes locally using their internal hardware, reducing their dependency on the server during runtime and lowering network load.

### 3.2.10 Hybrid Approach

The LTSP model supports a hybrid architecture, allowing system administrators to deploy a mix of thin and fat clients within the same network. This flexibility enables organizations to tailor their infrastructure to the varying performance needs of different users. While thin clients are ideal for general-purpose or lightweight tasks, fat clients cater to users requiring enhanced performance or offline functionality.

### 3.2.11 Community and Support

LTSP is backed by a strong and active open-source community that contributes to its ongoing development and provides extensive support. Users benefit from access to comprehensive documentation, online forums, and collaborative community resources. This ecosystem encourages knowledge sharing, facilitates problem-solving, and supports the continued evolution of LTSP.

## 3.3 Display Servers

Display servers are essential components of the Linux graphics infrastructure, responsible for handling graphical output, input devices, and window management. Two major display server technologies in use today are **X.Org Server (Xorg)** and **Wayland**. In this project, GNOME is used as the window manager, which by default operates atop the Wayland protocol in most modern Linux distributions.

### 3.3.1 X.Org Server (Xorg)

X.Org Server, commonly referred to as Xorg, has historically been the standard display server for Unix-like operating systems. It implements a client-server architecture where applications (clients) communicate with the X server to render graphics, manage input devices (keyboard, mouse, etc.), and control window interactions.

A key feature of Xorg is its **network transparency**, allowing applications to run on remote systems while rendering their display locally. This has made Xorg a preferred choice for distributed computing environments for many years.

However, Xorg's aging architecture and complex codebase introduce performance bottlenecks and security challenges. These limitations have catalyzed the development of more modern solutions like Wayland.

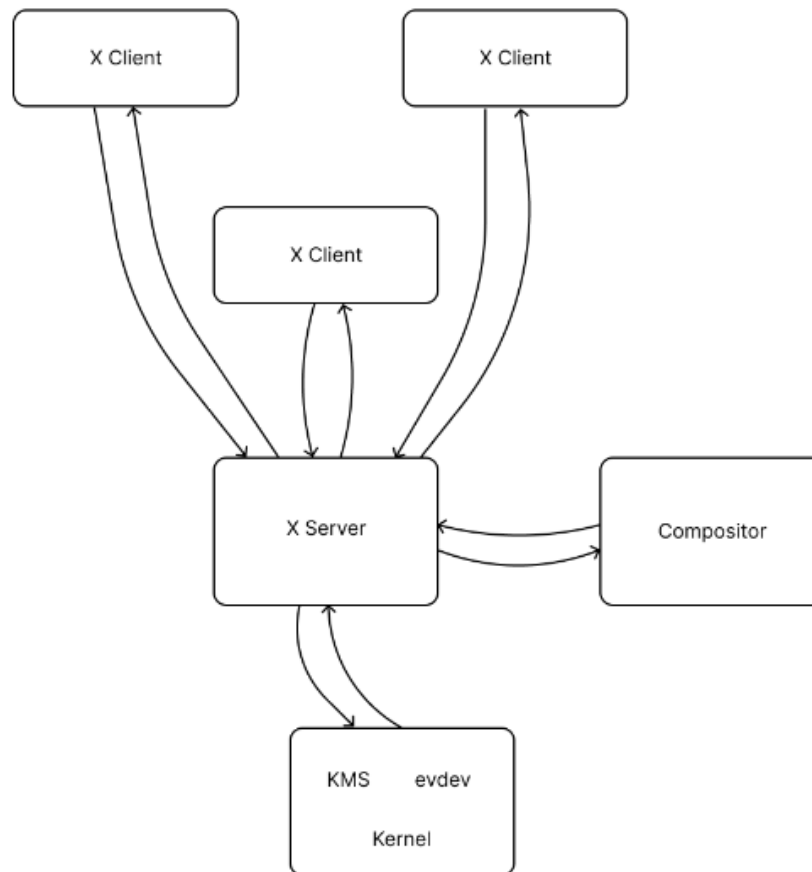


Figure 3.3: Architecture of X.Org Server.

### **Xorg Event and Rendering Workflow:**

1. The kernel receives an input event (e.g., keyboard or mouse) and forwards it via the evdev input driver to the X server.
2. The X server determines which application window should receive the event and dispatches it accordingly.
3. The X server lacks awareness of window transformations (e.g., scaling, rotation) handled by the compositor.
4. The client interprets the input event and updates its user interface as needed.
5. The client sends rendering instructions back to the X server.
6. The X server passes these instructions to the graphics driver for hardware rendering.
7. The X server calculates the region that has changed and notifies the compositor via a *damage event*.
8. The compositor recomposites only the modified screen regions.

9. Despite managing screen content, the compositor still relies on the X server for rendering.
10. The X server handles front/back buffer management or performs a page flip as requested by the compositor.
11. Context switches between overlapping windows may occur, which are unnecessary in full-screen compositing scenarios.

### 3.3.2 Wayland

Wayland is a modern display server protocol designed to replace the traditional Xorg system by simplifying the graphics stack. Unlike Xorg's client-server approach, Wayland allows clients to communicate directly with the compositor, eliminating the need for a separate server process.

This streamlined architecture improves performance, reduces graphical latency, and delivers smoother animations. It also introduces a more secure environment by enforcing strict sandboxing, preventing applications from interfering with each other's input or display buffers.

#### **Advantages of Wayland:**

- **Improved performance:** Reduced overhead and more direct rendering paths enhance graphical responsiveness.
- **Better security:** Applications are isolated, preventing unauthorized access to input/output streams.
- **Modern protocol:** Wayland is designed with contemporary computing needs in mind, offering a cleaner and more maintainable codebase.

While Wayland has seen growing adoption, especially in environments like GNOME (used in this project), some legacy applications and graphics drivers may still face compatibility issues. However, the ecosystem continues to mature, with ongoing development focused on improving support across a wider range of software and hardware.

*Note:* GNOME, the window manager used in this project, defaults to using Wayland for its graphical session, leveraging its advantages in performance, security, and user experience.



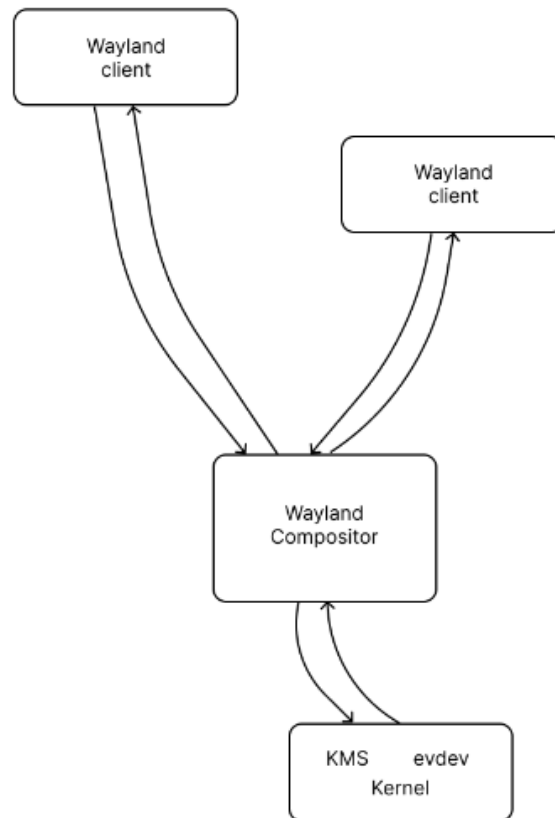


Figure 3.4: Wayland architecture.

### 3.4 DHCP Servers

Dynamic Host Configuration Protocol (DHCP) servers play a fundamental role in network management by automating the assignment of IP addresses and network configuration parameters to client devices. DHCP servers provide the following features:

#### 1. IP Address Assignment

- (a) DHCP servers dynamically allocate IP addresses to client devices within a network, eliminating the need for manual configuration.
- (b) When a client device connects to the network, it sends a DHCP request broadcast message to discover available DHCP servers.
- (c) The DHCP server responds with an offer, including an IP address lease and other configuration parameters.

#### 2. Lease Management

- (a) DHCP servers lease IP addresses to client devices for a specific duration, known as the lease time.
- (b) Client devices periodically renew their leases with the DHCP server to maintain network connectivity.

- (c) Lease management allows DHCP servers to reclaim and reassign IP addresses that are no longer in use, optimizing address utilization within the network.

### **3. Configuration Parameters**

- (a) In addition to IP addresses, DHCP servers provide clients with essential network configuration parameters, such as subnet mask, default gateway, and DNS server addresses.
- (b) These parameters enable clients to properly configure their network interfaces and communicate with other devices within the network and the internet.

### **4. Scalability and Centralized Management**

- (a) DHCP servers facilitate the scalability of network deployments by centralizing IP address management.
- (b) Administrators can easily deploy and manage large networks without the need for manual IP address assignment on individual devices.
- (c) Centralized management simplifies network administration tasks, such as IP address allocation, troubleshooting, and network reconfiguration.

### **5. Redundancy and High Availability**

- (a) DHCP servers support redundancy and high availability configurations to ensure uninterrupted service delivery.
- (b) Redundant DHCP servers operate in failover pairs, where one server assumes the DHCP duties if the primary server becomes unavailable.
- (c) High availability configurations prevent network disruptions and downtime, enhancing the reliability of DHCP services.

### **6. Security Considerations**

- (a) DHCP servers incorporate security features to mitigate potential risks, such as unauthorized IP address assignment or network attacks.
- (b) Authentication mechanisms, access control lists (ACLs), and DHCP snooping are commonly employed to secure DHCP transactions and prevent malicious activities.

DHCP servers streamline network configuration and management by automating the assignment of IP addresses and network parameters to client devices. Their scalability, centralized management capabilities, and support for redundancy contribute to the efficient operation and reliability of modern networks.

## **3.5 LTSP Networking**

LTSP networking involves the configuration of a centralized server that delivers computing resources to multiple thin or fat clients over a local network. This architecture is highly cost-effective, energy-efficient, and easy to manage, making it an ideal solution for environments such as schools, libraries, and offices where users do not require dedicated high-performance computing hardware.

There are two commonly used approaches for setting up LTSP networking:

1. **Single NIC with External DHCP Server:** In this setup, the LTSP server uses a single network interface card (NIC) and relies on an external DHCP server—such as a router, pfSense box, or a Windows server—to provide IP addresses and network configuration to clients. This is a plug-and-play approach with minimal configuration and works well in environments where a DHCP infrastructure is already available.
2. **Dual NIC Configuration:** This method involves configuring the LTSP server with two NICs. One NIC connects to the institution's regular network (which includes internet access), and the second NIC is dedicated to a private network connected to a switch that serves only the LTSP clients. This setup enables complete isolation and dedicated control over the LTSP subnet and DHCP service.

In our implementation, we adopt a **single NIC configuration** with the added flexibility of toggling between two modes:

- **DHCP Proxy Mode:** Where the LTSP server simply forwards the DHCP requests from clients to an external DHCP server.
- **Internal DHCP Mode:** Where the LTSP server runs its own DHCP server to manage IP address assignment within an isolated or controlled network environment.

This hybrid approach allows for adaptable deployment depending on network infrastructure, offering the convenience of integration with existing systems while retaining the ability to operate in self-contained setups when needed.

## Chapter 4: Implementation

In developing our secure lab-based examination environment, our approach centers around the use of **LTSP (Linux Terminal Server Project)** to ensure uniform, controlled, and non-persistent system behavior across all client terminals. By designing custom netboot images, we create a standardized environment that eliminates inconsistencies, local storage vulnerabilities, and unauthorized software access. Each client boots directly from the server, guaranteeing that the system starts in a clean state for every exam session.

Unlike traditional online exam systems that rely on web-based kiosk modes and internet connectivity, our solution is specifically engineered for lab-based deployment, where all components operate within a local, isolated network. The absence of browser-based kiosk configurations allows for deeper system-level control, enabling us to fine-tune the environment to suit academic use cases while maintaining a minimal, distraction-free interface.

A critical component of our setup is the integration of a custom-built **Flask** web application that serves as the management interface for teachers and administrators. This interface facilitates exam creation, question uploads, student login management, and submission tracking, all within a user-friendly framework. Additionally, the application allows for the assignment of static IPs to netbooted clients, enhancing network stability and simplifying identification during exams.

To maintain oversight and ensure fairness, the system is integrated with **Eproptes**, a powerful classroom monitoring tool that enables real-time supervision of all clients, screen broadcasting, and session control. Through Eproptes, administrators can observe, assist, or intervene in ongoing sessions without disrupting the entire lab.

Security, performance, and maintainability have been our guiding principles throughout the implementation. From extensive compatibility testing of netboot images across varied hardware, to securing the management layer with proper access controls and encryption protocols, each component has been built with precision and purpose. This collaborative effort between system developers, administrators, and institutional stakeholders aims to deliver a highly secure, scalable, and efficient examination framework tailored specifically for controlled lab environments.

### 4.1 Setting Up LTSP

LTSP offers several methods to install and configure a server capable of providing computing environments to client machines. These methods cater to varying deployment requirements and the preferences of system administrators. The commonly used methods include:

#### 1. Creating a Client Image from the Server OS

- This approach uses the server's existing operating system as the base template for client machines.
- It is simple and efficient, as it replicates the server's configuration and installed packages.

- It ensures consistency across all clients, which run the same OS and software stack as the server.
- The administrator maintains and configures a single OS for both server and clients.

## 2. Creating a Virtual Machine for Client Image

- In this method, a virtual machine (VM) is created on the server to simulate the desired client environment.
- The VM acts as a sandbox for testing, configuring, and customizing client images before deployment.
- It provides flexibility, allowing easy modification and cloning of the client environment.
- However, managing both the server and VM OS adds an extra layer of complexity.

## 3. Using a Chroot Jail

- This technique sets up a chroot jail within the server, isolating a directory structure that becomes the root filesystem for client systems.
- The chrooted environment is configured independently, giving administrators granular control over what is included in the client image.
- It offers a high degree of flexibility and isolation, while still using the server's resources.
- This method requires careful configuration to ensure stability, compatibility, and security.

Each of these methods has its own trade-offs and is selected based on factors such as deployment scale, customization requirements, and available resources. In our project, we adopt a hybrid approach that combines the strengths of both the **Virtual Machine** and **chroot jail** methods.

We begin by creating a custom operating system image inside a virtual machine, allowing us to tailor the environment with specific applications, features, and configurations based on exam requirements. Once the image is finalized, it is exported and transformed into a chroot jail on the LTSP server. This chroot environment then serves as the base image from which clients boot via PXE. The image is added to the LTSP iPXE boot menu, enabling network booting across all lab systems.

This approach allows us to efficiently manage and update multiple client OS images with varying configurations. By modifying the original VM templates and regenerating the chroot environments, we can push updates or changes to clients with minimal overhead. The method ensures flexibility, consistency, and scalability—making it ideal for maintaining secure and standardized lab environments for examinations.

### 4.1.1 Installation of LTSP Server Components

Installing LTSP begins with selecting the appropriate Linux distribution that supports the required tools and system structure. Since LTSP is packaged exclusively in the .deb format, it is essential to use a Debian-based distribution. Additionally, the system must support systemd as its init system, as LTSP relies heavily on systemd units for managing core services such as dnsmasq in the background.

For this project, we selected **Debian 12 Bookworm** with the **XFCE** desktop environment as our LTSP server OS. Debian 12, being a stable and widely supported .deb-based distribution, aligns perfectly with LTSP's packaging format. The choice of XFCE provides a lightweight yet fully functional desktop interface, ideal for server environments where performance and resource efficiency are critical.

Systemd integration allows seamless management of LTSP services, enabling automatic startup and streamlined control over networking, DHCP, TFTP, and boot image provisioning. This tight coupling ensures a stable, reliable platform that simplifies both day-to-day operations and long-term maintenance of the LTSP server.

The installation and configuration steps for LTSP are well-documented in the official LTSP wiki, offering guidance on package installation, image creation, PXE boot setup, and service management. Following these guidelines, we implemented a **single-NIC** setup in which DHCP requests from clients are **proxied to an external DHCP server** (e.g., router or firewall appliance).

To enhance flexibility, we integrated a custom feature into our management layer that allows administrators to **toggle between DHCP proxy mode and internal DHCP service**. This means the same LTSP setup can either forward DHCP requests to an external service or independently assign IP addresses within an isolated network—depending on the exam scenario or network policy.

This dual-mode capability is built into our Flask-based management layer and enhances adaptability during lab examinations. The result is a versatile and maintainable LTSP server setup, powered by a well-optimized OS base, and designed for the demands of secure, scalable, and centrally managed lab environments.

### 4.1.2 Building Client Images

In an LTSP-based lab examination environment, building customized client images is essential to ensure that each terminal boots into a secure, consistent, and purpose-built operating system. One of the key components enabling this flexibility is the **iPXE boot menu**, which allows multiple operating system images to be hosted and selected dynamically during the boot process.

Using iPXE, administrators can maintain and manage a variety of netboot images tailored to different use cases—whether for different exam sessions, departments, or testing tools. Each client system, upon booting, is presented with a selection menu from which the appropriate OS image can be chosen.

To build these images, we utilize **VirtualBox**, a powerful and open-source virtualization tool that supports x86 and AMD64/Intel64 architectures. VirtualBox provides an efficient,

isolated environment for generating clean and minimal OS installations. For this project, we use it to create **bare-bones Debian-based virtual machines**, which serve as templates for client OS images.

Each VM is configured with only the necessary software and features required for the lab exam environment. This includes system utilities, exam submission tools, and performance-optimized settings. Once the VM setup is complete, the image is exported and converted into a chroot-compatible format for LTSP integration.

These chroot environments are then linked to LTSP and made available through the iPXE bootloader. This setup gives system administrators fine-grained control over what each client boots into, while ensuring a lightweight and standardized experience across all lab machines.

This modular and scalable approach simplifies the deployment of multiple exam environments and allows for rapid updates or rollbacks. By combining VirtualBox's flexibility with LTSP's centralized architecture, the system remains easy to maintain, secure, and adaptable to institutional needs.

## 4.2 Custom Client Image Creation

In this project, the client image used for lab examinations was created using a method that combines the flexibility of a full OS environment with the centralized control offered by LTSP. Instead of building the image from scratch or relying solely on virtual machines, we adopted a practical and efficient approach that starts by cloning the server's own operating system.

The process begins with the creation of a complete system image from the server OS. Since the server is already configured with all necessary packages, drivers, and settings, it serves as a reliable and consistent base for generating the client environment. This approach ensures that the client image inherits the stability and compatibility of the host system.

Once the server OS image is generated, it is converted into a modifiable format by extracting its contents using the `unsquashfs` tool. This process decompresses the image into a chroot directory structure, which acts as a standalone Linux filesystem. The chroot environment allows for safe and isolated modifications, enabling the addition or removal of applications, configuration files, scripts, and exam-specific tools without affecting the live server.

Inside the chroot, we make the necessary adjustments to align the environment with the requirements of the exam—such as removing unnecessary system services, disabling access to external applications, and preparing the system for secure student login. Any sensitive data, unwanted software, or graphical bloat is stripped out to keep the image lightweight and focused.

After the modifications are complete, the chroot environment is registered with LTSP and linked to the iPXE boot menu. During the boot process, LTSP serves this customized image to clients over the network. As each client boots into this image, it inherits all the configured restrictions and tools, maintaining a consistent and secure experience for all examinees.

This method provides a fast and efficient workflow for managing client environments.

Changes can be rolled out by simply updating the chroot contents and rebuilding the image, avoiding the need to recreate or reconfigure clients individually. The result is a scalable, reliable, and easily maintainable system tailored specifically for controlled lab examinations.

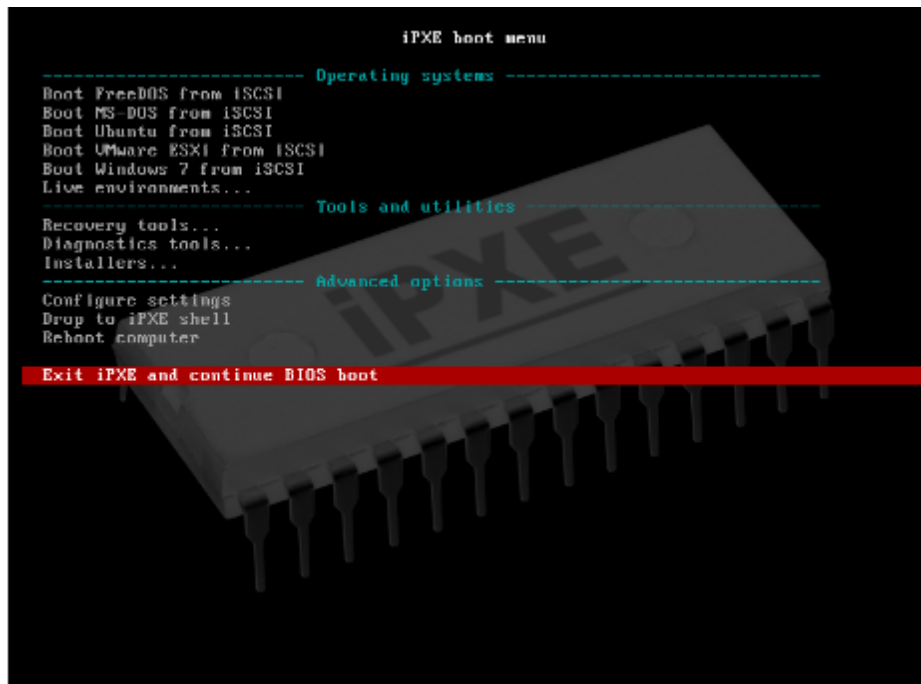


Figure 4.1: iPXE boot menu

## 4.3 Multiple DHCP Servers

When implementing an internal DHCP server on an LTSP setup, it is critical to ensure that no other DHCP servers are active on the same network segment. The presence of multiple, uncoordinated DHCP servers can lead to IP address conflicts, unpredictable client behavior, and failure of the LTSP boot process. For stable and reliable operation, there must be a clear, singular point of DHCP control.



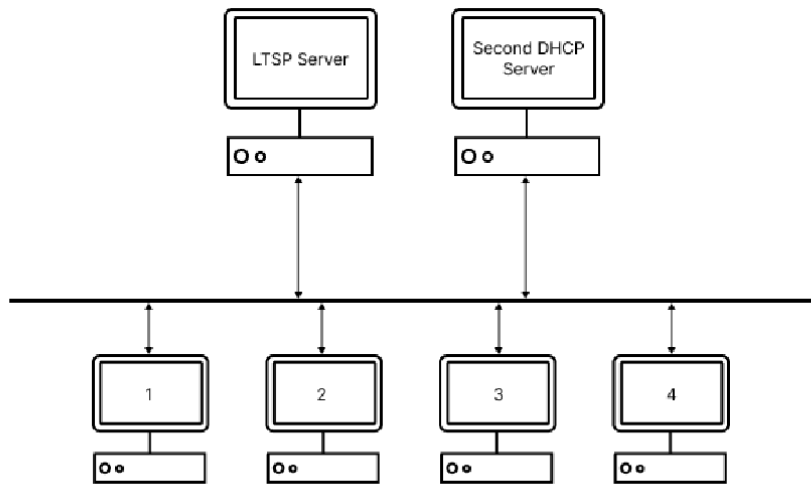


Figure 4.2: An isolated network with a secundar DHCP server and LTSP clients

During the boot sequence of an LTSP client, multiple DHCP requests may be broadcast as the system progresses through different stages of initialization. These requests occur sequentially, and understanding them is key to managing network behavior effectively:

### 4.3.1 On-board Network Stack / NIC

The first DHCP request originates from the client's hardware network interface card (NIC) during power-up. This is handled by the firmware's built-in PXE stack and is the initial step in acquiring network configuration before the bootloader is even loaded.

### 4.3.2 Chainloaded iPXE Program

After the client receives basic boot instructions from the network, it loads the iPXE boot-loader—either from the network or local media. iPXE itself may issue a second DHCP request to confirm or refresh its network configuration before proceeding to load the Linux kernel.

### 4.3.3 Netbooted Linux Kernel

The third and final DHCP request is sent by the netbooted Linux kernel. This request is typically made once the kernel is loaded into memory and begins initializing network interfaces for full system operation.

Each of these three requests plays a distinct role in the boot process, and all rely on the correct DHCP server being present to supply not just an IP address, but also critical boot parameters like the location of the TFTP server and boot file name.

However, if another DHCP server—such as one from a router or another unconfigured system—is active on the same network, it may respond to one of these requests, bypassing the LTSP server entirely. In such a case, the client may receive an IP address without the necessary boot directives, resulting in failure to download and execute the required kernel and initrd files.

**Therefore, to ensure proper LTSP operation, it is essential to eliminate or isolate any secondary DHCP servers** on the network unless they are explicitly coordinated as part of a failover pair. In our setup, we implemented additional flexibility within the management layer by allowing administrators to toggle between DHCP proxy mode (for environments with external DHCP servers) and a fully internal DHCP service managed by the LTSP server itself.

This safeguard preserves the integrity of the boot process, ensures client systems receive the correct configuration, and maintains the overall stability of the LTSP-based examination environment.

## 4.4 Configuring dnsmasq for LTSP

When deploying LTSP, administrators must configure network services that enable clients to boot over the network. This includes DHCP, TFTP, and PXE boot protocols. To streamline this setup, LTSP uses dnsmasq, a lightweight and multifunctional network tool that combines DNS, DHCP, TFTP, and PXE boot services into a single, easy-to-configure daemon.

dnsmasq simplifies the LTSP environment by avoiding the need for separate services such as `isc-dhcp-server`, `tftpd-hpa`, and others. It provides a minimal and efficient solution that is well-suited for lab environments where resource usage, reliability, and ease of configuration are essential.

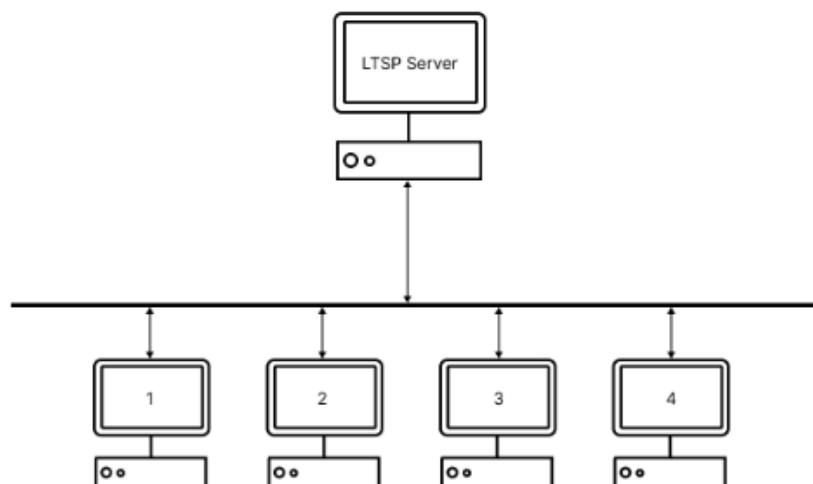


Figure 4.3: LTSP server that also act as a DHCP server

In a typical LTSP installation, dnsmasq is configured to:

- Provide dynamic IP address allocation to LTSP clients.
- Serve boot files (kernel and initrd) over TFTP.
- Enable PXE boot functionality for network booting.

- Point clients to the appropriate bootloader using DHCP options.

The configuration file is usually located at `/etc/ltsp/dnsmasq.conf`, and includes options to define the TFTP root, IP range, PXE boot menu, and other essentials. Once set up, LTSP's systemd services ensure that dnsmasq starts automatically with the correct configuration.

In our project, we used the standard LTSP installation workflow to configure dnsmasq in its default DHCP and TFTP mode. No additional customization was required beyond the documented steps in the official LTSP guides. This setup proved effective in reliably serving client systems across the lab network during exam sessions.

## 4.5 Real-time Monitoring and Logging

Real-time monitoring and logging are essential features in our LTSP-based examination environment, providing administrators with comprehensive oversight and control capabilities. This functionality is implemented using a dedicated application called **Epoptes**, developed by the same core team behind LTSP.

Epoptes enables administrators to monitor the activity of each client machine connected to the LTSP server, view real-time display output, and perform various administrative actions remotely. It simplifies classroom and lab management by combining monitoring, control, and command execution within a single graphical interface.

Epoptes offers the following core features:

### 1. Real-time Monitoring

- Epoptes provides administrators with live monitoring of all connected client machines.
- Administrators can view the display output of each client to observe user activity, assist with troubleshooting, or track examination progress.
- This enhances visibility and enables proactive intervention in case of suspicious behavior or technical issues.

### 2. Remote Command Execution

- Epoptes allows administrators to execute remote commands from the LTSP server.
- Common commands include shutdown, reboot, logoff, and message broadcasting, which can be applied to individual or multiple clients simultaneously.
- This streamlines system administration, especially in large-scale environments where manual interaction with each client is impractical.

### 3. Centralized Control and Management

- Epoptes provides a unified control interface, combining monitoring and management tools in one application.

- Administrators can access Epoptes from the LTSP server or any authorized workstation on the same network.
- Centralization simplifies management workflows and increases administrative efficiency.

#### 4. User Authentication and Access Control

- Access to Epoptes is restricted to authorized personnel through authentication mechanisms.
- Only authenticated users can utilize the monitoring and control features, protecting the system from unauthorized access or misuse.

#### 5. Logging and Auditing

- Epoptes maintains logs of all administrative actions and monitoring events.
- These logs provide an audit trail useful for troubleshooting, compliance verification, and incident response.
- Logging contributes to maintaining exam integrity and accountability in the administration process.

By integrating Epoptes into our LTSP environment, we ensure effective real-time control, proactive monitoring, and centralized management of lab-based exam systems. It plays a critical role in upholding the security, integrity, and smooth execution of examinations.

## 4.6 Web Application Implementation

To manage and secure the examination workflow, we developed a custom-built web application using the **Flask** microframework. The application provides distinct interfaces for teachers and students, running on the LTSP server and accessed through web browsers. The web app supports core functionality such as exam creation, real-time monitoring, secure student login, question distribution, timed exams, and answer submission. It is lightweight, modular, and designed to run efficiently alongside the LTSP environment.

### 4.6.1 Teacher Interface

The teacher interface is accessible from the LTSP server through a browser. It requires authentication to prevent unauthorized access. Once logged in, teachers are presented with a centralized dashboard to manage exams and monitor students.

Teachers can perform the following actions:

- **Create Exam:** Define exam title, time duration, and specify the roll number range of participating students.
- **Add Questions:** Input individual or multiple questions to associate with the exam.
- **Start Exam:** Launch the session, allowing students to log in and begin.

After the exam is started, the dashboard displays a live monitoring table showing:

- Student registration number
- Assigned IP address
- Assigned question
- Submission status (Submitted/Not Submitted)
- View/download button for submitted answers

### **Question Assignment Logic**

To reduce the likelihood of cheating, questions are distributed intelligently using a round-robin algorithm. Students are assigned questions based on their registration number or IP address in such a way that:

- Adjacent students (physically or digitally) are less likely to receive the same question.
- Questions are evenly distributed among all students.
- No bias exists in question assignment, ensuring exam fairness.

### **4.6.2 Student Interface**

The student interface is automatically opened in Firefox on the LTSP client system upon boot. The browser runs in full-screen mode with restricted navigation and no access to the internet or external resources.

Students are required to:

- Enter their registration number
- Select the active exam

After logging in, the system:

- Assigns a unique question to the student
- Starts a countdown timer based on the exam duration
- Displays a file upload section for answer submission

The system restricts actions to ensure integrity:

- Internet access is disabled.
- System applications and shortcuts are blocked.
- The student's IP is logged and bound to their session to prevent impersonation or switching systems.

### 4.6.3 Security and Session Control

- Teachers can view submitted files during or after the exam via the dashboard.
- Once the exam timer expires, the upload interface locks automatically.
- All student actions are tracked via IP and timestamp to support traceability.

This web application forms the backbone of our lab-based examination system, seamlessly integrating with LTSP, and providing a smooth, controlled experience for both administrators and students.

## 4.7 Deployment

The deployment process of the system is streamlined through a custom shell script named `setup.sh`. This script automates the installation and configuration of necessary components on a clean **Debian 12 Bookworm** system, including LTSP and the Flask web application backend.

The deployment steps are as follows:

1. **Root Privilege Check:** Confirms the script is run with administrative rights.
2. **Distribution Verification:** Ensures the OS is compatible (Debian-based).
3. **System Updates:** Installs package updates using `apt`.
4. **LTSP Installation:** Installs LTSP and all required services including `dnsmasq`, `nfs-kernel-server` and `openssh-server`.
5. **Virtual Environment Setup:** Creates and activates a Python virtual environment, installs dependencies from `requirements.txt`.
6. **Web Server Launch:** Starts the Flask web application via Gunicorn, then opens Firefox to access the admin dashboard.

Once the web server is up, the browser automatically opens the admin interface where the teacher can proceed with exam creation and setup (see Section 4.6). From this point, the LTSP clients can be powered on.

Each client system boots into a controlled environment, where it fetches its image over the network and launches Firefox to access the student portal. Static IP assignment is handled automatically through LTSP configuration and logged at the time of student login.

After the exam, the administrator can use **Epopetes** to trigger a bulk restart of all clients, ensuring a clean exit and preparing the systems for future sessions.

*For details regarding the functionality and flow of the web application, refer to Section 3.6: Web Application Implementation.*

## Chapter 5: Results and Screenshots

This chapter showcases the final outcome of the secure lab-based examination system implemented using LTSP, a custom Flask web application, and real-time monitoring tools. The screenshots included here demonstrate the functionality of both teacher and student interfaces, the effectiveness of the question distribution algorithm, and the robustness of the system's network control and security enforcement.

### 5.1 System Overview

The developed system successfully integrates:

- Centralized control over lab client machines via LTSP
- A fully functional web interface for teachers and students
- Dynamic question assignment with fairness in distribution
- Time-bound submissions with enforced client restrictions
- Real-time monitoring and reboot control via Eptotes

The outcome reflects a scalable, reliable, and exam-focused lab environment where academic integrity and operational control are prioritized.

### 5.2 Teacher Interface Output

#### 5.2.1 Login Page

The entry point for administrative access. Only authorized teachers can log in and manage exam sessions.

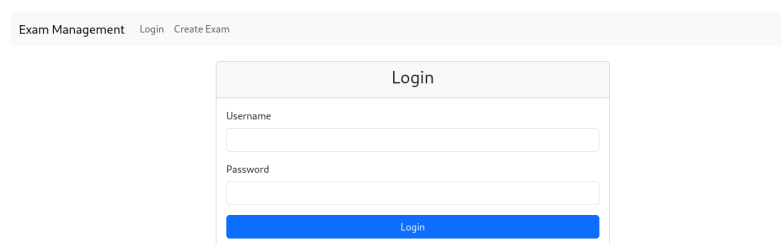
The screenshot shows a web interface for a login page. At the top, there is a navigation bar with three links: "Exam Management", "Login", and "Create Exam". Below this, the main content area is titled "Login". It contains two input fields: "Username" and "Password". Below the password field is a blue button labeled "Login".

Figure 5.1: Teacher Login Page

### 5.2.2 Exam Dashboard

After login, teachers are greeted with a dashboard to create and control exams. Once an exam is launched, a live table updates in real-time with student activity.

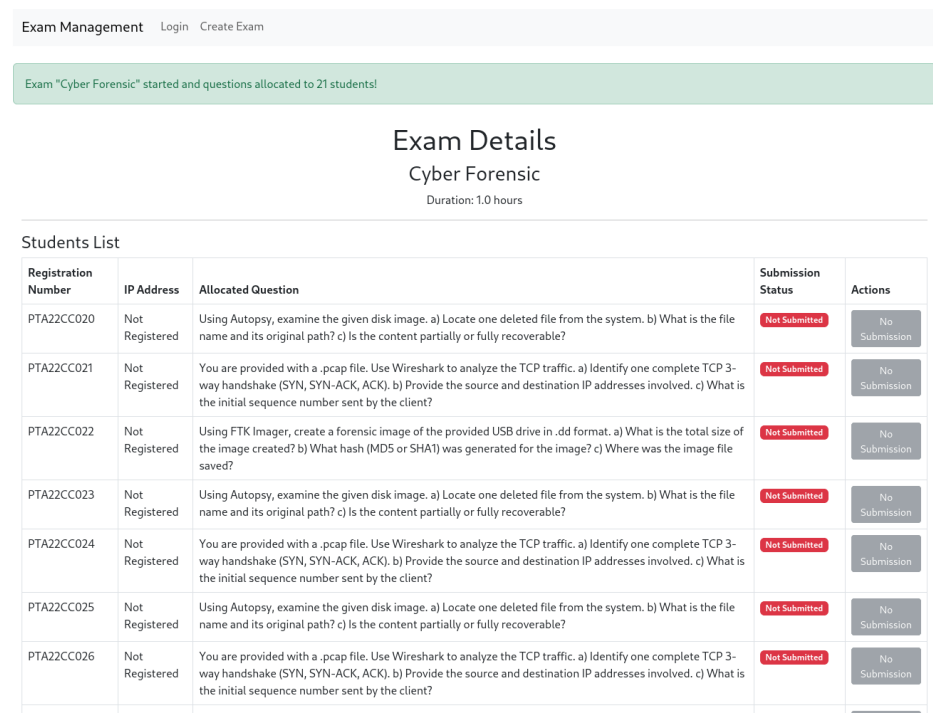


Figure 5.2: Teacher Dashboard: Live Monitoring of Exam Status

### 5.2.3 Question Creation Interface

Teachers can add questions and link them to exams. The system handles distribution automatically upon student login.

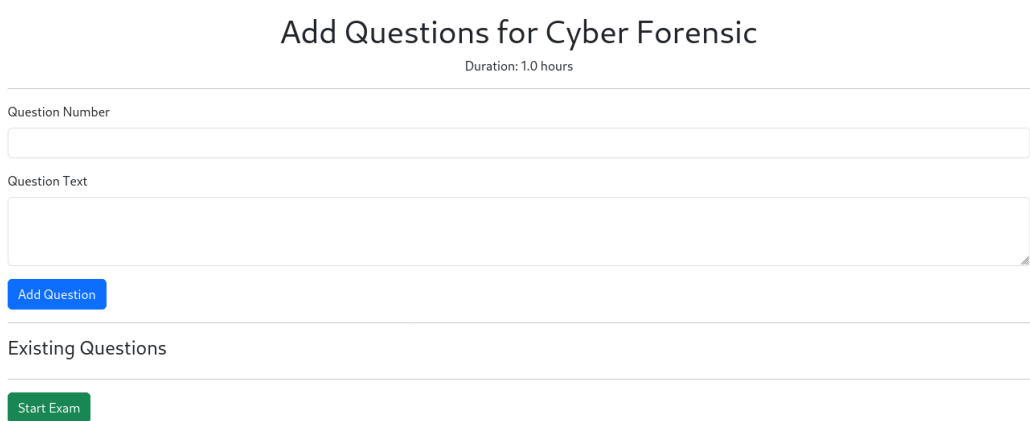


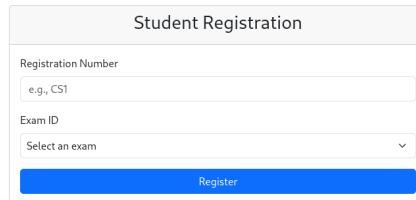
Figure 5.3: Question Entry Interface



## 5.3 Student Interface Output

### 5.3.1 Login Page

Clients boot into Firefox with full-screen lockdown. Students must enter their registration number and select the exam.

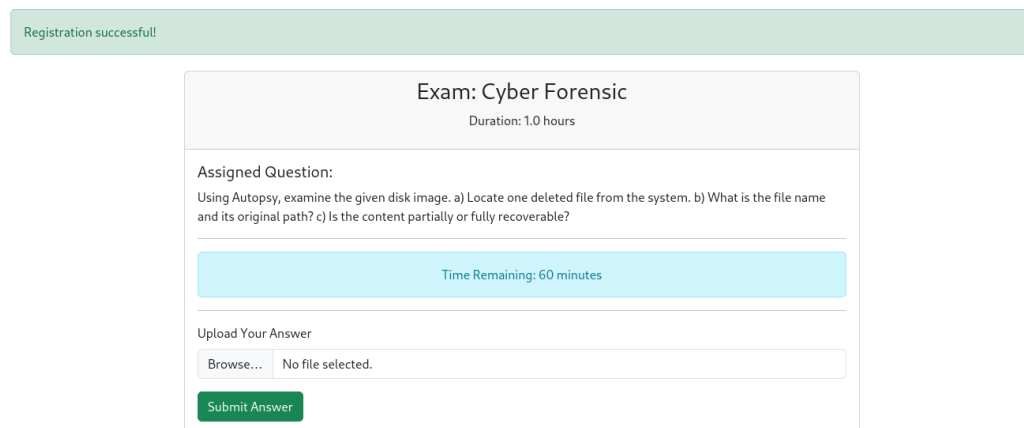


The image shows a 'Student Registration' form. It has a title bar 'Student Registration'. Below it, there is a text input field for 'Registration Number' with a placeholder 'e.g., CS1'. Below that is a dropdown menu for 'Exam ID' with the text 'Select an exam' and a downward arrow. At the bottom is a blue button labeled 'Register'.

Figure 5.4: Student Login Page

### 5.3.2 Exam Interface

Upon successful login, students receive a question, a countdown timer, and an upload field. The interface locks after the timer expires.



The image shows the 'Exam Interface'. At the top, there is a green notification bar that says 'Registration successful!'. Below it is the main exam interface. The title is 'Exam: Cyber Forensic' with a subtitle 'Duration: 1.0 hours'. The 'Assigned Question:' section contains the text: 'Using Autopsy, examine the given disk image. a) Locate one deleted file from the system. b) What is the file name and its original path? c) Is the content partially or fully recoverable?'. Below the question is a light blue bar that says 'Time Remaining: 60 minutes'. Underneath is the 'Upload Your Answer' section, which includes a 'Browse...' button, a text field showing 'No file selected.', and a green 'Submit Answer' button.

Figure 5.5: Exam Interface: Assigned Question, Timer, and Submission

## 5.4 Monitoring and Control Output

### 5.4.1 Static IP Table

The server maps each student to a unique IP address upon login, preventing identity switching. This IP is tracked in the teacher dashboard.

Exam Management Login Create Exam

Exam "Cyber Forensic" started and questions allocated to 21 students!

Exam Details

Cyber Forensic

Duration: 1.0 hours

Students List

Registration Number	IP Address	Allocated Question	Submission Status	Actions
PTA22CC020	Not Registered	Using Autopsy, examine the given disk image. a) Locate one deleted file from the system. b) What is the file name and its original path? c) Is the content partially or fully recoverable?	Not Submitted	No Submission
PTA22CC021	Not Registered	You are provided with a .pcap file. Use Wireshark to analyze the TCP traffic. a) Identify one complete TCP 3-way handshake (SYN, SYN-ACK, ACK). b) Provide the source and destination IP addresses involved. c) What is the initial sequence number sent by the client?	Not Submitted	No Submission
PTA22CC022	Not Registered	Using FTK Imager, create a forensic image of the provided USB drive in .dd format. a) What is the total size of the image created? b) What hash (MD5 or SHA1) was generated for the image? c) Where was the image file saved?	Not Submitted	No Submission
PTA22CC023	Not Registered	Using Autopsy, examine the given disk image. a) Locate one deleted file from the system. b) What is the file name and its original path? c) Is the content partially or fully recoverable?	Not Submitted	No Submission
PTA22CC024	Not Registered	You are provided with a .pcap file. Use Wireshark to analyze the TCP traffic. a) Identify one complete TCP 3-way handshake (SYN, SYN-ACK, ACK). b) Provide the source and destination IP addresses involved. c) What is the initial sequence number sent by the client?	Not Submitted	No Submission
PTA22CC025	Not Registered	Using Autopsy, examine the given disk image. a) Locate one deleted file from the system. b) What is the file name and its original path? c) Is the content partially or fully recoverable?	Not Submitted	No Submission
PTA22CC026	Not Registered	You are provided with a .pcap file. Use Wireshark to analyze the TCP traffic. a) Identify one complete TCP 3-way handshake (SYN, SYN-ACK, ACK). b) Provide the source and destination IP addresses involved. c) What is the initial sequence number sent by the client?	Not Submitted	No Submission

Figure 5.6: Student IP and Submission Status Overview

5.4.2 Eproptes Monitoring

Eproptes gives administrators a real-time window into all client screens. It supports broad-casting, messaging, rebooting, and full control—all from one place.

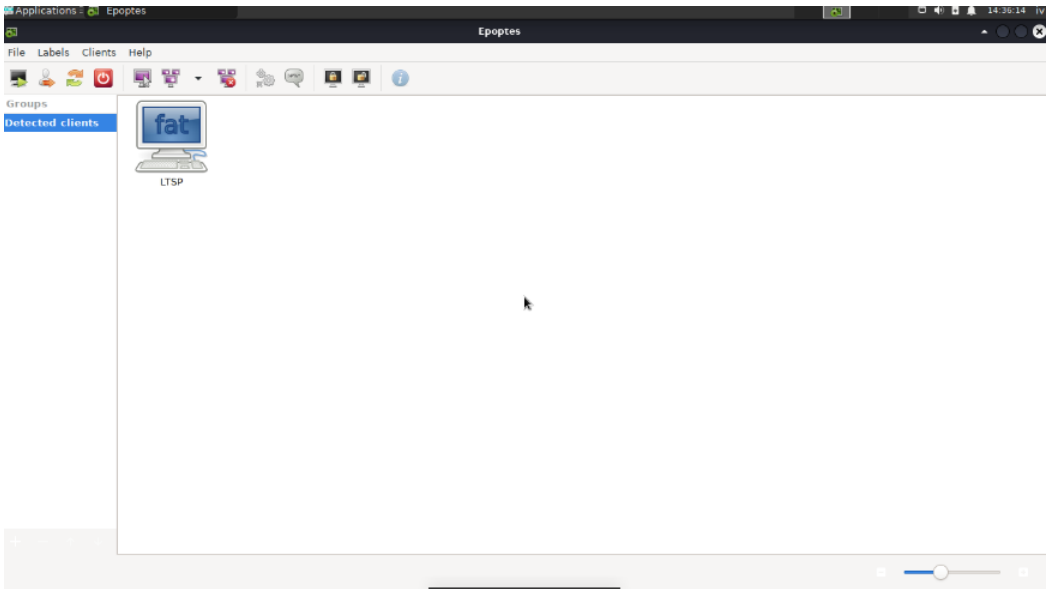


Figure 5.7: Eproptes: Real-Time Classroom Monitoring Tool

## 5.5 Summary and Observations

The system was tested in a lab setup mimicking a real exam environment. Results observed:

- Question assignment worked accurately with zero repetition between adjacent students.
- Clients were successfully isolated from external network access.
- Exam sessions respected time limits and automatically disabled uploads post-deadline.
- Eproctes monitoring enabled seamless classroom control without physical intervention.
- The static IP locking mechanism effectively prevented students from switching systems.

Overall, the system delivered a secure, stable, and fair exam experience—achieving the core goals of integrity, centralization, and ease of administration in a lab-based environment.

## Chapter 6: Conclusion

The implementation of a secure lab-based examination environment marks a significant advancement in modern educational assessment systems. This project successfully brings together technologies such as **LTSP** for centralized netbooting and **Eproptes** for real-time classroom monitoring, to deliver a platform that is secure, scalable, and streamlined for both administrators and students.

By eliminating traditional dependencies on browser-based kiosk modes and instead focusing on a controlled, boot-level architecture, we ensured that each student system boots into a clean, non-persistent, exam-ready state. Our approach not only enhances system uniformity but also effectively minimizes opportunities for tampering, unauthorized access, or academic dishonesty.

The inclusion of a custom-built **Flask web application** served as the heart of the management system, enabling teachers to easily create exams, assign questions, monitor submissions, and oversee the entire assessment process from a single dashboard. On the student side, a clean, focused interface provided a distraction-free experience with built-in safeguards such as time-bound submission, IP tracking, and question randomization.

The automated workflow—from static IP assignment to question distribution and final answer submission—dramatically reduced the administrative burden typically associated with lab-based exams. Additionally, real-time visibility through Eproptes empowered administrators to take instant action when needed, reinforcing exam integrity and system reliability.

This project directly addresses the critical needs of educational institutions looking to modernize their examination processes. It creates a secure, centralized environment that can scale efficiently for use in schools, colleges, certification centers, and professional testing environments.

**Looking ahead**, this system lays the groundwork for future enhancements such as biometric authentication, offline grading modules, automated result processing, and deep integration with learning management systems (LMS). As educational models evolve in a post-digital world, the adaptability and foundation of this platform make it an ideal candidate for continued innovation.

With this project, we demonstrate not just a solution—but a vision. A vision of secure, fair, and future-ready education systems, where technology serves both academic rigor and administrative ease. This is a step toward empowering educators and learners alike, fostering trust in digital assessment methods, and shaping a more resilient educational future.

## Chapter 7: Future Scope

The secure lab-based examination environment developed in this project serves as a solid foundation for conducting controlled, high-integrity assessments within institutional networks. While the current system fulfills its intended purpose effectively, there are several areas of expansion and enhancement that can be pursued to make the platform even more versatile, efficient, and adaptable to emerging needs in the educational technology space.

### 1. Migration to Lightweight Init Systems

The present implementation relies on `systemd` for service and session management. In the future, transitioning to a more lightweight init system such as `runit` or `s6` could offer performance benefits, especially on older or low-spec client hardware. A minimalist init system would streamline the boot process, reduce system overhead, and potentially lead to faster session startup times—making the platform more responsive and resource-efficient.

This migration would align with the core principles of the project: minimalism, security, and scalability. It would also allow broader deployment across environments where system resources are limited.

### 2. Cross-Distribution Packaging

Currently, the system is optimized for Debian-based distributions. To widen its accessibility and adoption, future work could include creating distribution-specific packages and installation scripts for popular Linux environments such as:

- Ubuntu
- Fedora
- Arch Linux
- CentOS / Rocky Linux

Developing cross-platform compatibility would allow institutions with different infrastructure preferences to integrate the system seamlessly, without being tied to a specific base OS. Packaging the project as a `.deb` or `.rpm`, along with proper dependency management, would greatly simplify deployment and encourage broader adoption.

### 3. AI-Driven Proctoring and Monitoring

One of the most promising directions for the project lies in the integration of AI-based proctoring capabilities. As online and lab-based exams scale up, manual monitoring becomes increasingly difficult to manage. AI can assist by:

- Detecting behavioral anomalies during the exam session
- Identifying screen activity patterns suggestive of malpractice
- Analyzing answer submission patterns to flag irregularities or potential collusion

Rather than replacing human supervision, AI can act as a supplementary layer of intelligence, offering real-time alerts and post-exam analysis to support administrators in maintaining exam integrity at scale.

## **4. Greater Network Flexibility and Hybrid Models**

Future iterations could allow the system to support more flexible deployment models, including hybrid setups where some students are on-premise (LTSP-based) and others are connected remotely through VPN-based isolation. With proper segmentation and monitoring, this approach could extend the use of the platform beyond just physical labs, while still preserving its core values of control and standardization.

## **5. Custom Reporting and Analytics**

Currently, exam results are viewed manually. Future enhancements could include:

- Auto-generated reports summarizing exam submissions
- Student-wise performance metrics
- Session logs with timestamps and IP mapping

Such analytics would help educators gain quick insights into participation and behavior, while also serving as valuable audit trails for post-exam verification.

In conclusion, the future of this project lies not only in refining what already exists but in exploring meaningful enhancements that increase usability, portability, and intelligence—without compromising the integrity and simplicity that form the core of the platform.

# Bibliography

- [1] L. Jegatha Deborah and A. Kannan, “Secure online examination system for e-learning,” *ResearchGate*, 2018.
- [2] A. Fluck, D. Pullen, and C. Harper, “Secure e-examination systems compared: Case studies from two continents,” *Journal of Information Technology Education: Innovations in Practice*, vol. 16, pp. 107–125, 2017.
- [3] D. Hoxha, “Virtual computer lab,” *ResearchGate*, 2020.
- [4] I. J. of Engineering Research and Technology, “Advanced client management tool for diskless workstations using ltsp,” 2013.
- [5] I. Europe, “Computer lab management tool adopted by over 500 schools in greece,” *Interoperable Europe Portal*, 2014.
- [6] die.net, *dnsmasq(8) - Linux man page*, n.d. Available online: <https://linux.die.net/man/8/dnsmasq>.
- [7] die.net, *agetty(8) - Linux man page*, n.d. Available online: <https://linux.die.net/man/8/agetty>.
- [8] die.net, *xdpyinfo(1) - Linux man page*, n.d. Available online: <https://linux.die.net/man/1/xdpyinfo>.
- [9] LTSP Community, “Linux terminal server project.” <https://ltsp.org/>. Accessed: 2025-04-08.
- [10] LTSP Documentation, “Server installation - ltsp documentation.” <https://ltsp.org/docs/installation/#maintaining-a-client-image>. Accessed: 2025-04-08.
- [11] Oracle Corporation, “Oracle virtualbox.” <https://www.virtualbox.org/>. Accessed: 2025-04-08.
- [12] ArchWiki Contributors, “Systemd drop-in files.” [https://wiki.archlinux.org/title/systemd#Drop-in\\_files](https://wiki.archlinux.org/title/systemd#Drop-in_files). Accessed: 2025-04-08.
- [13] ArchWiki Contributors, “Xinit - archwiki.” <https://wiki.archlinux.org/title/xinit>. Accessed: 2025-04-08.
- [14] Epoptes Developers, “Epoptes: Computer lab management and monitoring tool.” <https://epoptes.org/>. Accessed: 2025-04-08.