Xavier University - Ateneo de Cagayan

Clinic Admission

Submitted by:
Gutierrez, Iline
Jaquilmac, Fruee Jane
Navidad, Paul
Toyogon, Carylle Hope

Submitted to:
Mr. Ian Alquitela

Testing and Quality Assurance

a. Deployment of Application on the configured Server

In the previous milestone, we focused on ensuring that the clinic scheduling system is stable, maintainable, and accessible in a simulated production environment. The system was deployed on an **Ubuntu virtual machine** configured with **PHP**, **MySQL**, and **Composer**. Both the Admin and Student Laravel applications were cloned into the server and properly installed using Composer. After configuring the **.env** files with the correct database credentials, the system was successfully migrated and seeded. File permissions were then set for the *storage/* and *bootstrap/cache/* directories to prevent write-related errors, which allowed Laravel to execute properly during runtime.

Initially, the application was run locally using *php artisan serve* to verify that both projects were functioning as expected. To simulate a production-ready environment, Apache was installed on the Ubuntu VM.

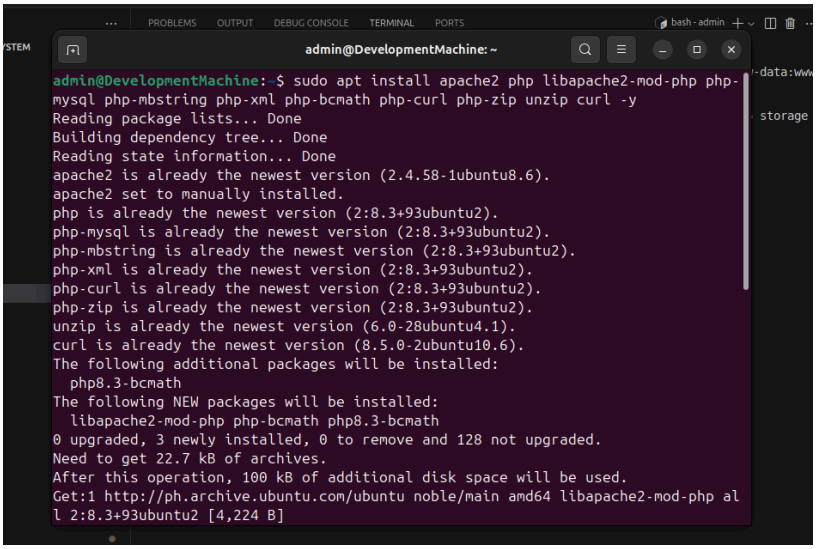


Figure 1. Installation of apache

Virtual host configuration files were created for both admin.local and student.local, each pointing to its respective Laravel project's public/ directory. The default Apache site was disabled to avoid conflicts, and the custom virtual host files were enabled using a2ensite. The /etc/hosts file was updated in both the VM and the host PC, allowing each domain to resolve correctly in the browser. As a result, both Laravel applications were successfully accessed from the host PC using *http://admin.local* and *http://student.local.*

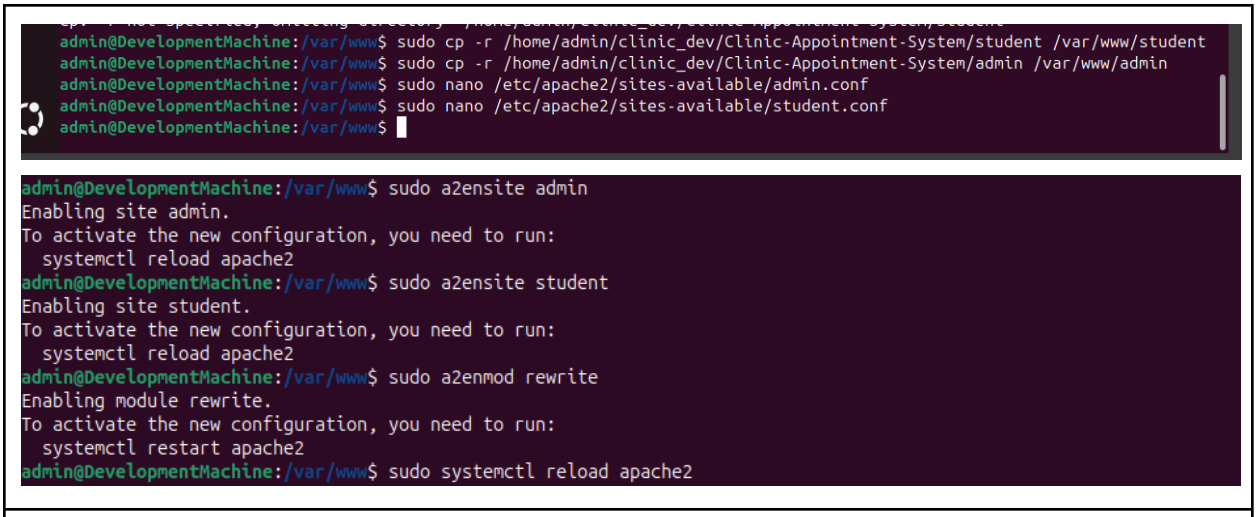


Figure 2. Enabling virtual host config files using a2ensite

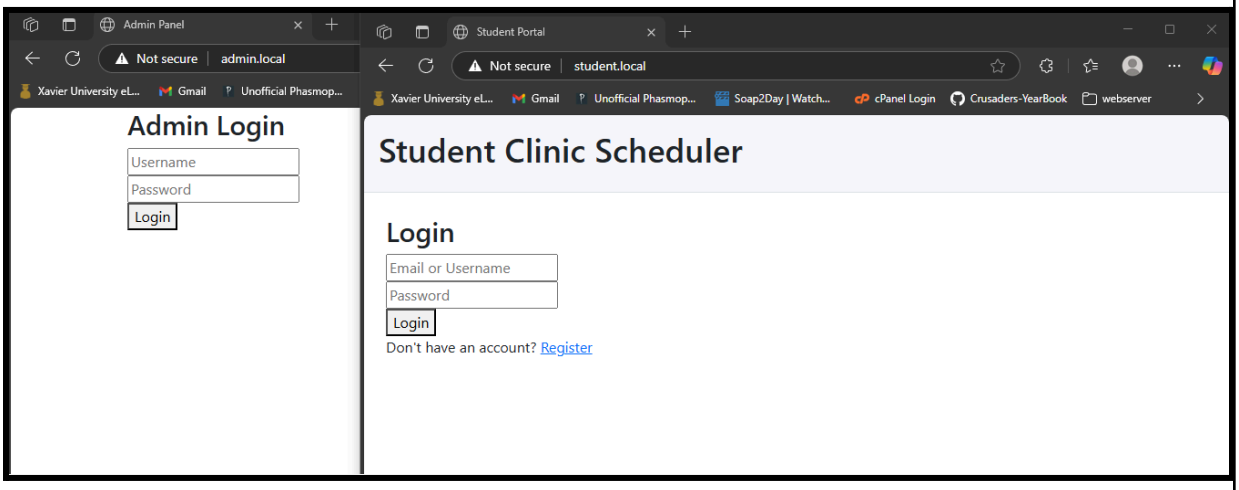


Figure 3. Accessing laravel applications in the host PC

To automate administrative tasks and improve system reliability, we implemented Bash scripts for daily database backups and periodic uptime monitoring. The backup_db.sh script creates a timestamped backup of the MySQL database and logs each backup in a log file located at ~/db_backups/backup.log.

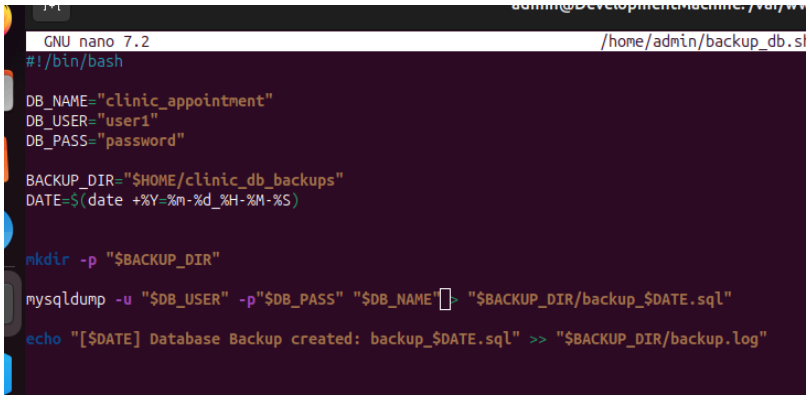


Figure 4. Script file: **backup_db.sh**

This script is scheduled to run automatically every day at 1:00 AM using a cron job. In addition, we created a script named check_web.sh, which checks if the Laravel application is online by sending an HTTP request to the server every 10 minutes. The script logs the status code to ~/server_status.log to help detect downtime or network issues. This script is also scheduled using cron.

Figure 5. Script file: **backup_db.sh**

Monitoring and logging at both the application and system levels has also been implemented, by using Laravel's built-in logging system, actively capturing application-level events and errors using *Log::info()* and *Log::error()* statements integrated into business process controllers. Additionally, System administrators can access and review the most recent log entries from both the Admin and Student systems directly through the web interface. This feature enhances visibility into application behavior and aids in faster debugging and maintenance.

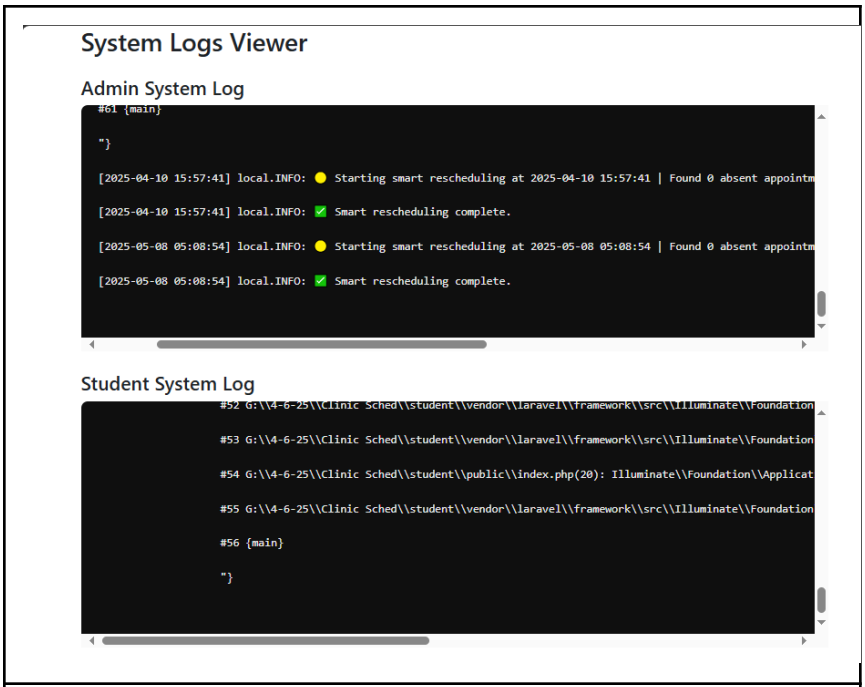


Figure 6. System Logs View Page

```php
protected function handleSmartRescheduling(): void
{
    $now = now();
    $today = $now->toDateString();

    $absentAppointments = Appointment::whereHas(relation: 'schedule', callback: function (Builder<TRelatedModel> $query) use ($now, $today)… {
            $query->whereDate(column: 'date', operator: $today)
                ->where(column: 'end_time', operator: '<', value: $now->format(format: 'H:i'));
        })
        ->where(column: 'is_present', operator: false)
        ->where(column: 'status', operator: 'booked')
        ->with(relations: ['user', 'schedule'])
        ->get();

    Log::info(message: "🟡 Starting smart rescheduling at {$now->toDateTimeString()} | Found {$absentAppointments->count()} absent appointment(s)");

    foreach ($absentAppointments as $appointment) {
        $user = $appointment->user;
        $originalSchedule = $appointment->schedule;

        Log::info(message: "⏱ Checking appointment for user: {$user->name} | Scheduled at: {$originalSchedule->start_time} - {$originalSchedule->end_time}");

        // Look for later available schedule
        $available = Schedule::whereDate(column: 'date', operator: $today)
            ->where(column: 'start_time', operator: '>', value: $originalSchedule->end_time)
            ->withCount(relations: 'appointments')
            ->get()
            ->firstWhere(key: function ($sched): bool {
                return $sched->appointments_count < $sched->slot_limit;
            });
```

Figure 7. Controller functions containing *Log::* statements

b. Testing

Student Side

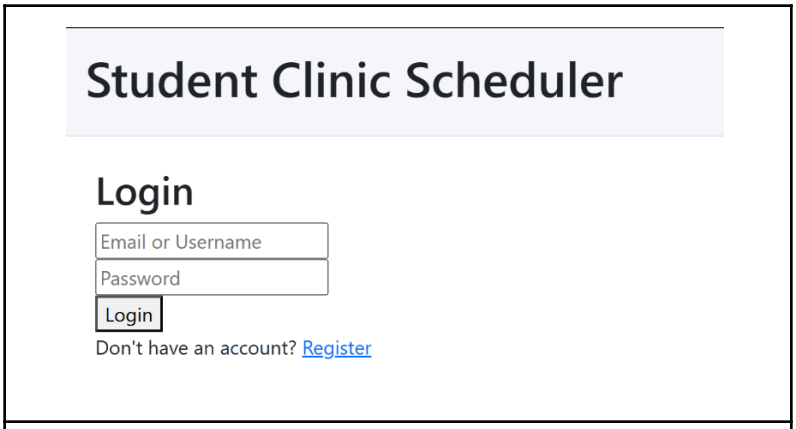


Figure 8. Student Login

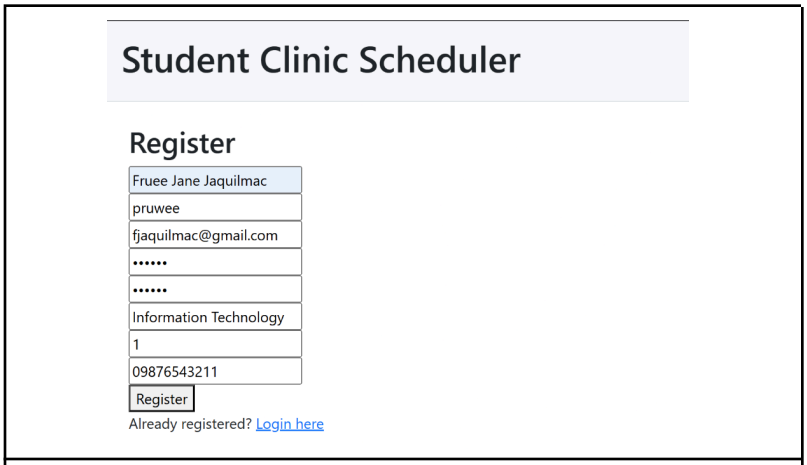


Figure 9. Student Register

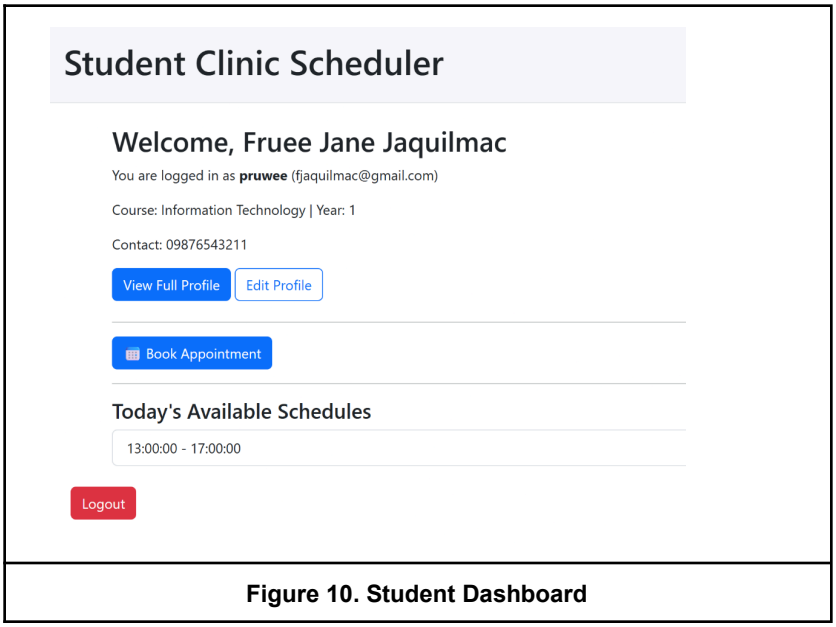
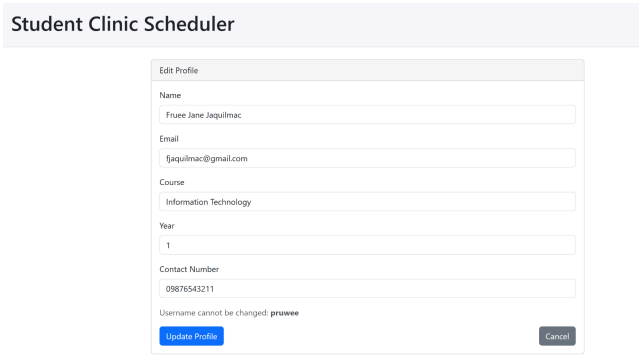
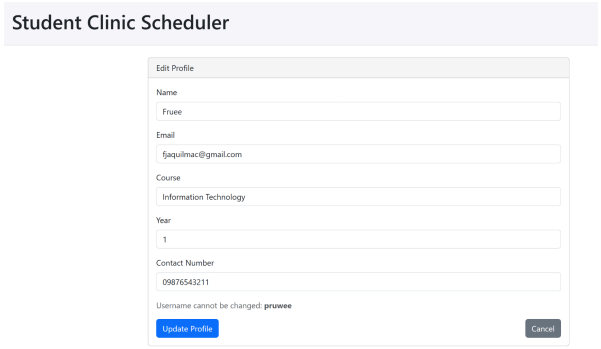


Figure 10. Student Dashboard





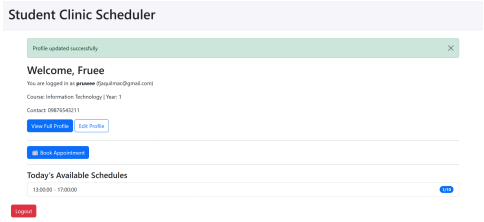


Figure 11. Student Edit Profile

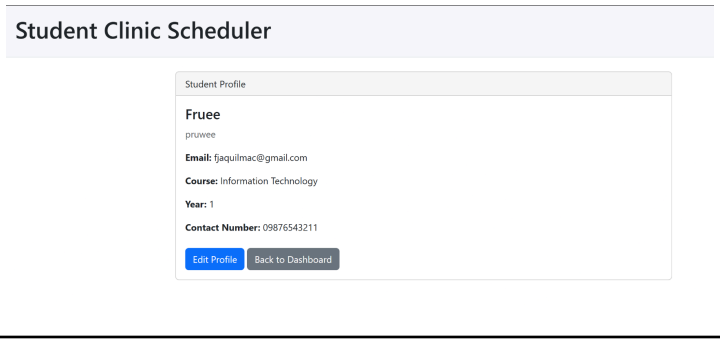
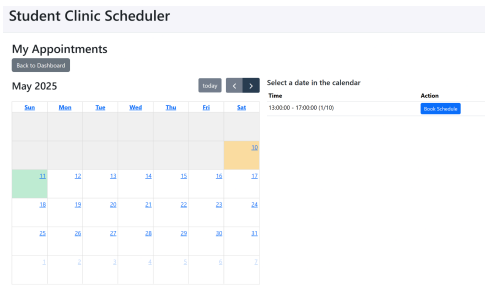
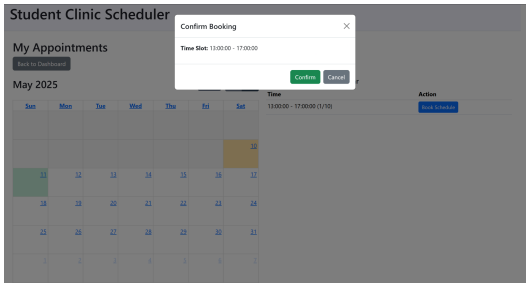


Figure 12. View Full Profile





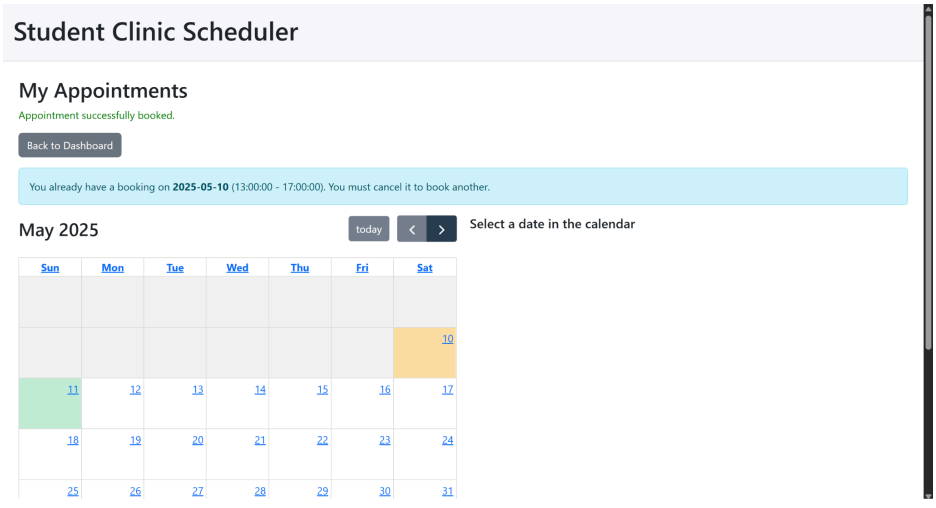
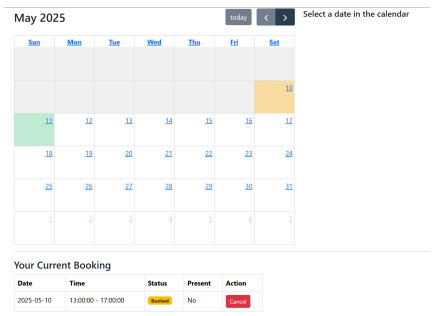
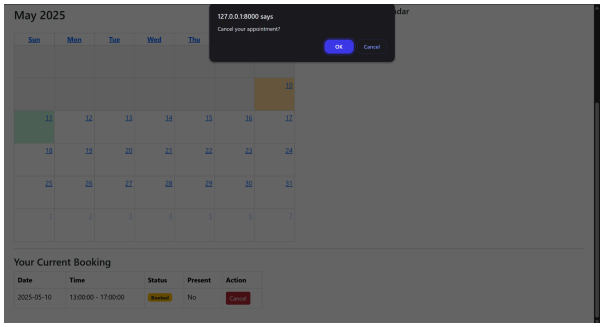


Figure 13. Student Book Appointment





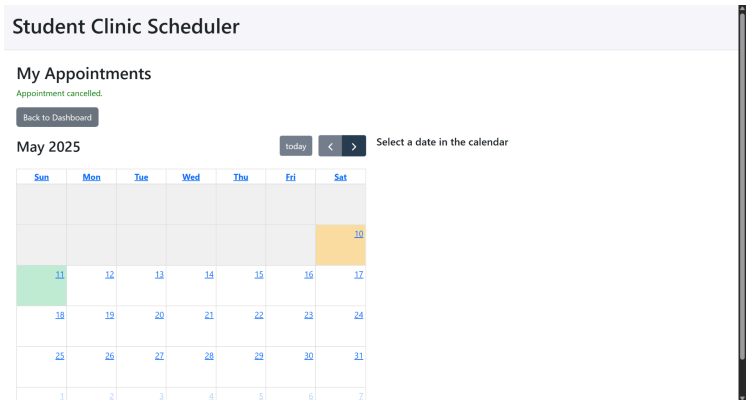


Figure 14. Student Booking Cancellation

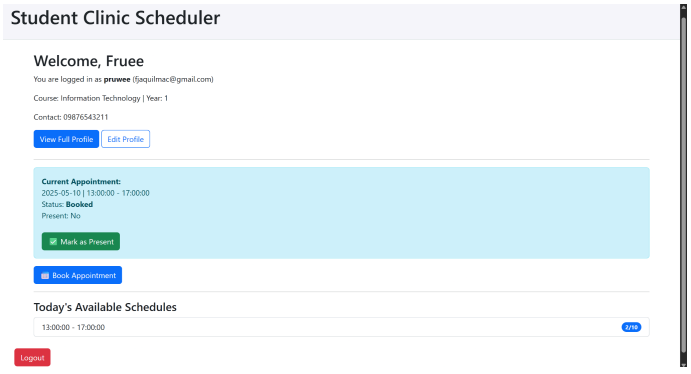


Figure 15. STUDENT DASHBOARD (booking again)

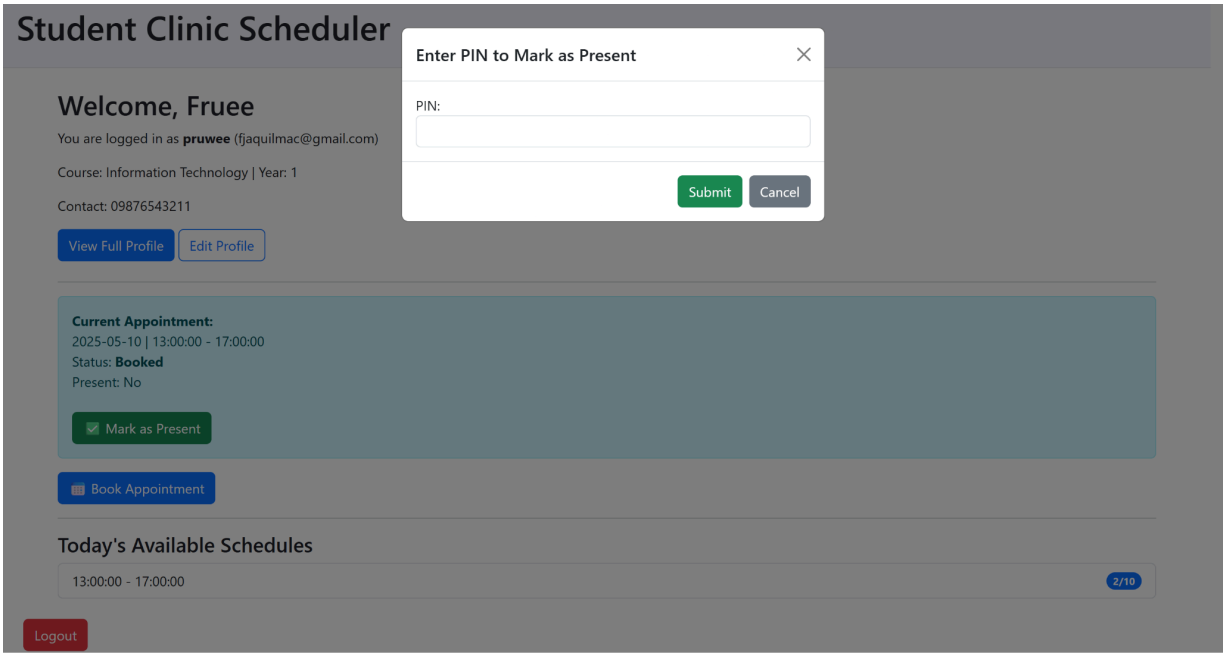


Figure 16. Student Side Marking as Present

The “Mark as Present” button will show a pop-up. You can obtain the PIN from the clinic itself when you show up from your appointment.

Admin Side

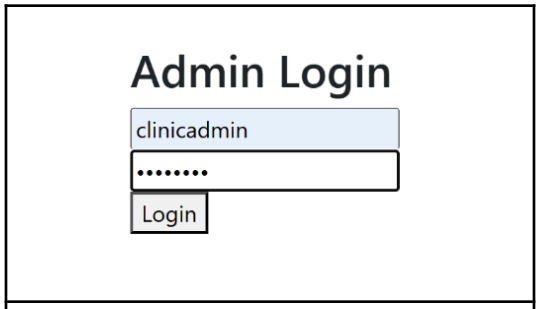


Figure 17. Admin Login Page

ADMIN DASHBOARD: You can see from the screenshot the appointment made by Fruee

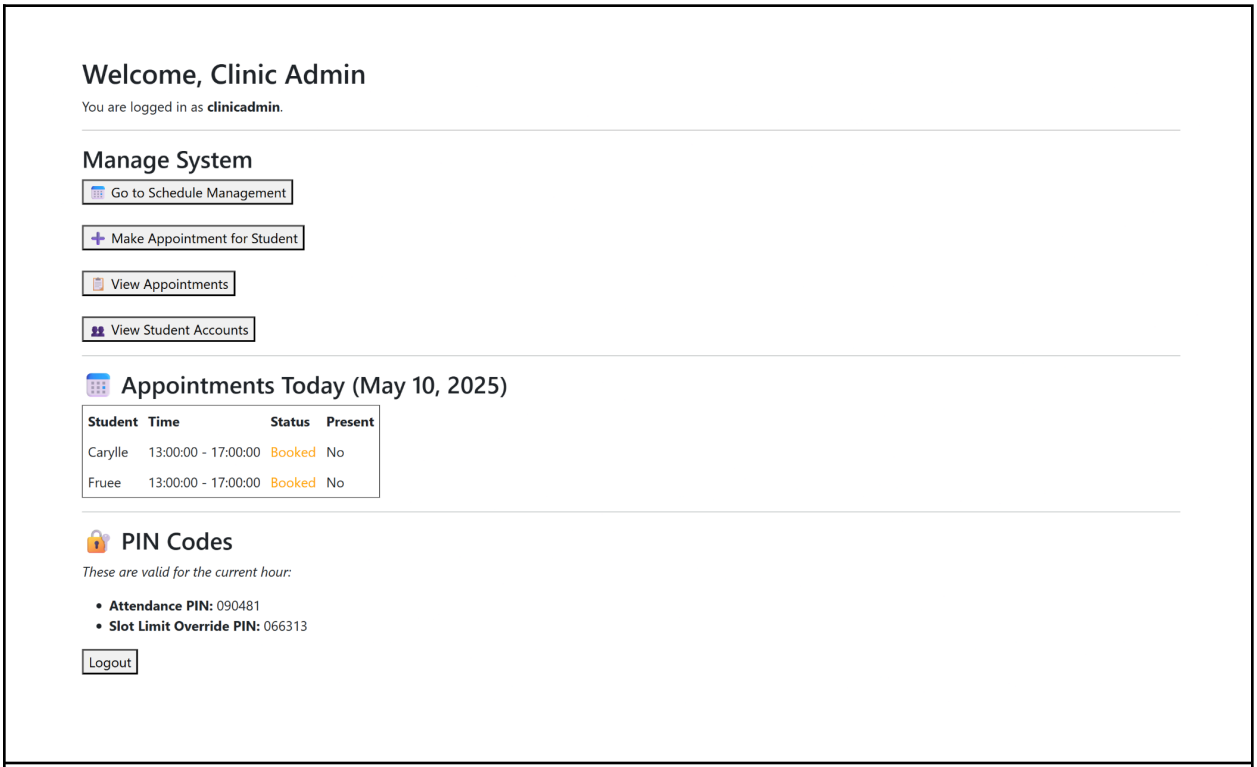


Figure 18. Admin Dashboard

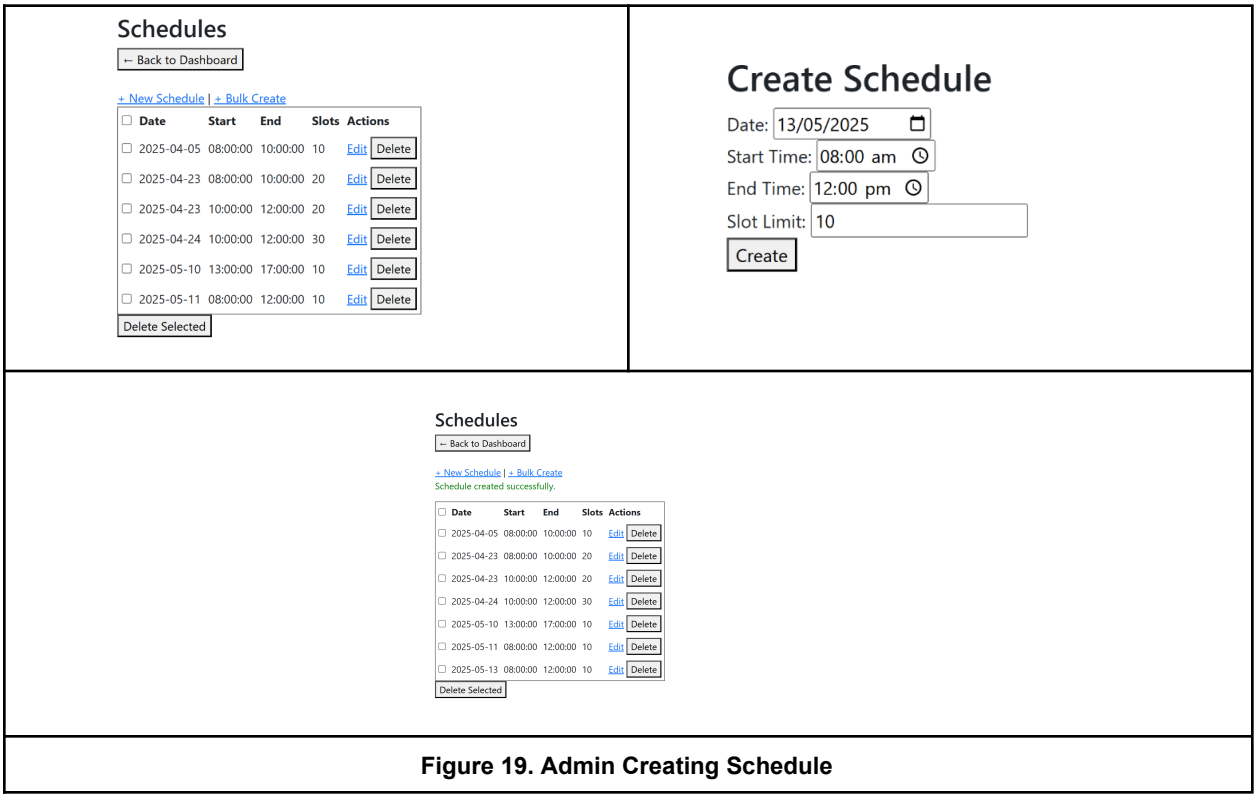


Figure 19. Admin Creating Schedule

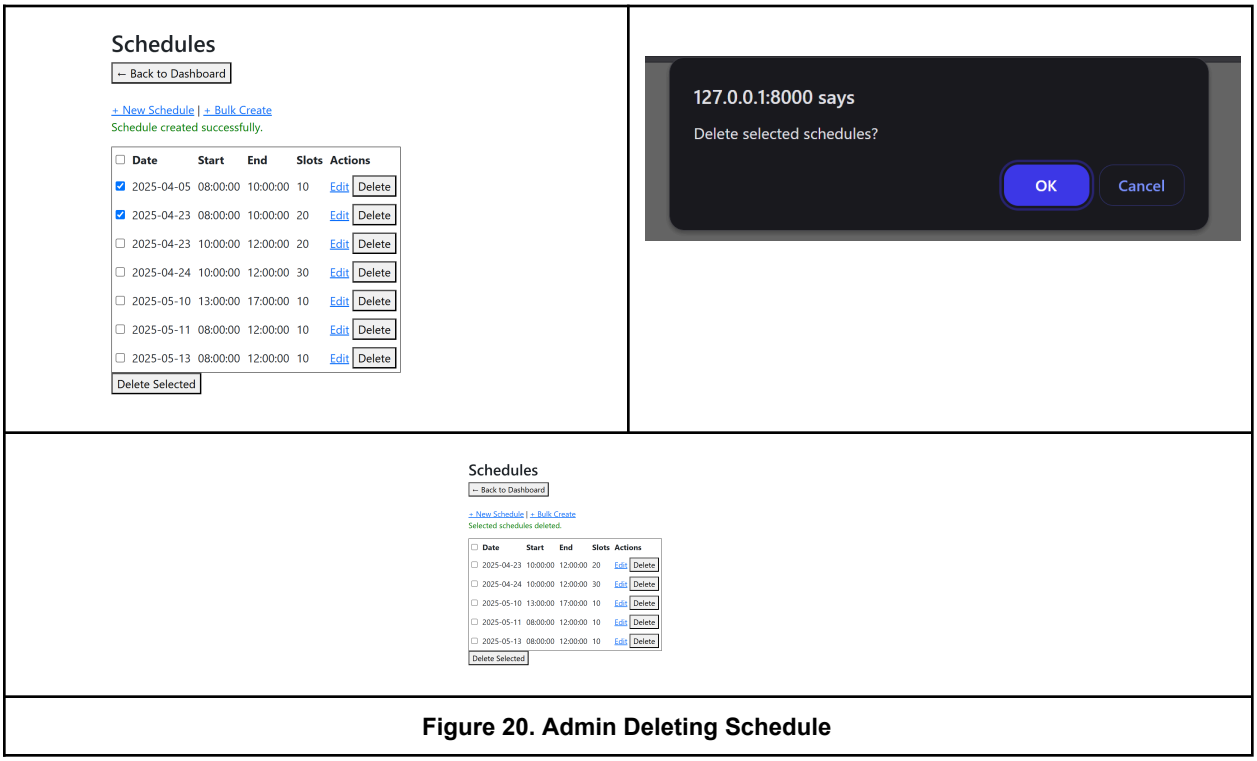


Figure 20. Admin Deleting Schedule

You can search or you can click on the “Search” button to view full list of all students who booked for an appointment

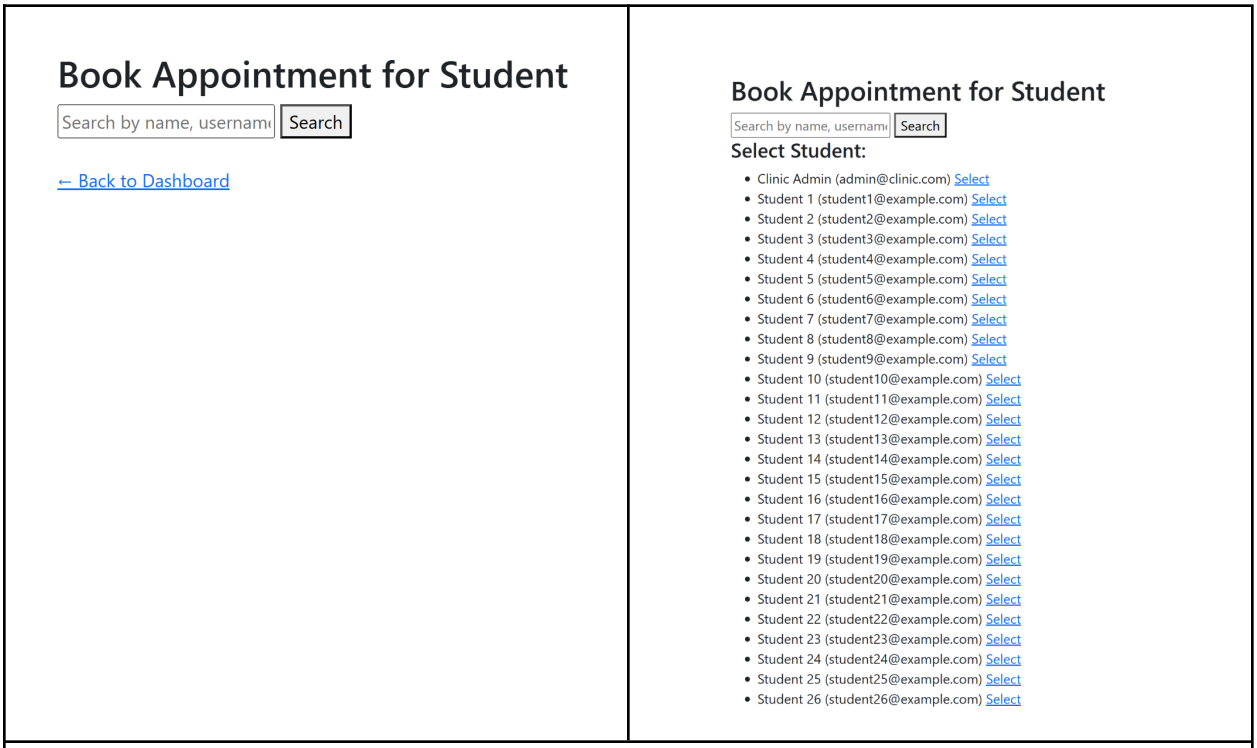
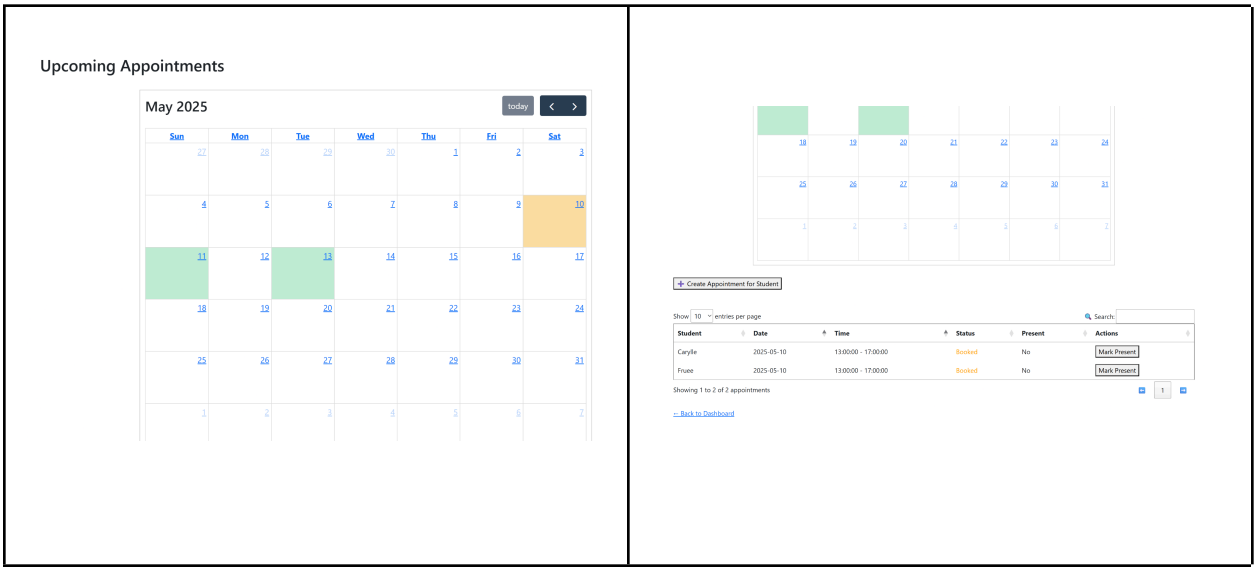


Figure 21. Admin Appointment for Student (in-case walkins)



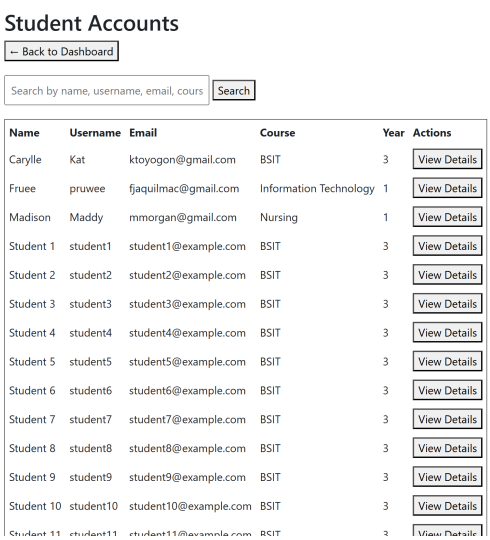


Figure 22. Admin View Appointments

c. Maintenance

We have successfully expanded our system-level performance monitoring by implementing an additional Bash script that records **RAM usage, disk space utilization, and the status of the Apache service** on an hourly basis, which then automatically logs these metrics into a dedicated log file, which are then integrated into the system's admin interface, allowing administrators to conveniently monitor performance and identify potential issues early on.

```
  GNU nano 7.2                          /home/admin/health_check.sh
#!/bin/bash

DATE=$(date +'%Y-%m-%d %H:%M:%S')
LOG_FILE="/home/admin/clinic_dev/Clinic-Appointment-System/admin/storage/logs/system_health.log"

MEMORY=$(free -h | grep Mem | awk '{print $3 " used / " $2 " total"}')
DISK=$(df -h / | tail -1 | awk '{print $5 " used"}')

APACHE_STATUS=$(systemctl is-active apache2)

echo "[$DATE] Memory: $MEMORY | Disk: $DISK | Apache: $APACHE_STATUS" >>  $LOG_FILE
```

Figure 23. Script file: health_check.sh

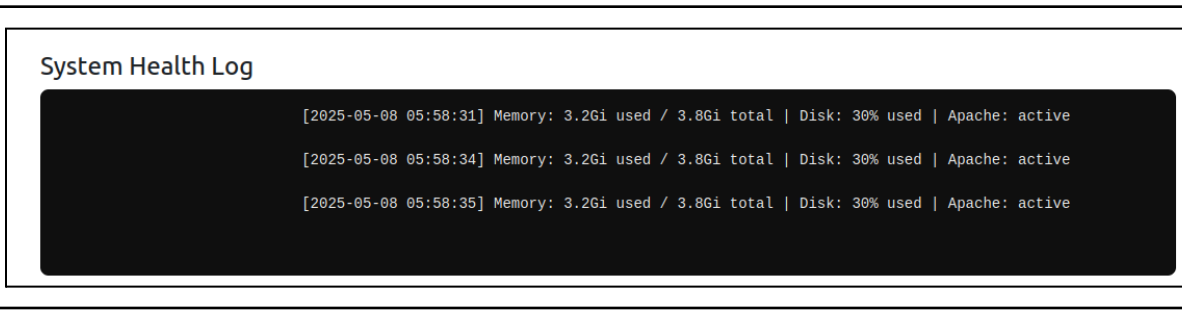


Figure 24. System health logs viewer