Portfolio
2025
Module ICT3715
INFORMATION AND COMMUNICATION TECHNOLOGY PROJECT

Welcome to the Portfolio
This Portfolio consists of 18 pages
The Portfolio is compulsory

Instructions:

1. Complete this Front Page (page 1).
2. Complete the Plagiarism Pledge (page 2). Your Portfolio will not be assessed without this.

STUDENT NUMBER (Student completes)									
6	9	7	2		4	8	0		6

IDENTITY NUMBER (Student completes)												
0	1	0	8	2	3	5	2	0	3	0	8	1

PLAGIARISM PLEDGE BY THE STUDENT

- I have read Unisa's plagiarism policy.
- I understand Unisa's plagiarism policy.
- I agree to abide by Unisa's plagiarism policy.
- I have read the direct copying, plagiarism, and "patch-writing" document.
- I understand what direct copying, plagiarism, and "patch-writing" is.
- I undertake to avoid copying directly, plagiarism and patch writing.
- All academic work, written or otherwise, that I submit is expected to be the result of my own skill and labour.
- I understand that, if I am guilty of the infringement of breach of copyright/plagiarism or unethical practice, I will be subject to the applicable disciplinary code as determined by Unisa.
- The marker has the right to refuse to assess the assignment and the system if plagiarism is detected.

Student name and Surname	Nishay Hira
Student number	69724806
Date	02 October 2025
Student signature	

Additional student instructions:

1. No handwritten Portfolios will be accepted.
2. Incorrect file format Portfolio will not be considered.
3. Make sure that you did complete the instructions on page 1 of this document (template).
4. Remove everything that is placed in brackets […].
5. Make sure that your Table of Content is updated.
6. Save the document as PDF, e.g., 12345678_ICT3715_Portfolio.pdf, (replace 1234568 with your student number).
7. When you are done submit 24 hours before your examination date.
8. Instructions where to submit will follow soon.

Plagiarism is a violation of academic integrity. Unisa has a zero tolerance for plagiarism and/or any other forms of academic dishonesty.

Here you can add your references that you have used e.g., information taken from the Internet, textbooks …

1. .
2. .
3. .
4. .
5. .
6. .
7. .
8. .
9. .
10. .

Table of Contents

1. User Guide

User Guide for Amandla High School Locker Booking System

[October 2025]

Instructions:

- Change everything that is in brackets []
- Add your practical system content to the document.
- Make sure that your Table of Content is updated.
- Update the document revision table.
- Complete all the Process and Process Steps for your practical system in this document

Document Revisions

Date	Version Number	Document Changes
01/10/2025	0.1	Initial Draft
05/10/2025	0.2	Added database structure section
10/10/2025	0.3	Added GUI screenshots and code

Introduction

Scope and Purpose

- **Introduce the practical system and its purpose**.
- Highlighting **key features and benefits**.
- Introduce the **purpose of the user guide**.

This guide introduces the Amandla High School Locker Booking System, a practical digital solution designed to streamline the management and assignment of school lockers.

The system is intended for use by:

- Parents (to register students, book lockers, manage applications, and track payments)
- Administration Staff (to monitor bookings, assign lockers, manage waitlists and track payments)

Key Features and Benefits

- Real-time locker availability by grade
- Parent login with secure authentication
- Admin dashboard with reporting tools
- Automated email confirmations and reminders
- Waitlist and payment tracking functionality
- Simple, mobile-friendly interface

Purpose of This User Guide

This guide will:

- Explain how users interact with the system
- Provide step-by-step instructions for key processes
- Include screenshots and diagrams for visual guidance
- Serve as a reference during system training or onboarding

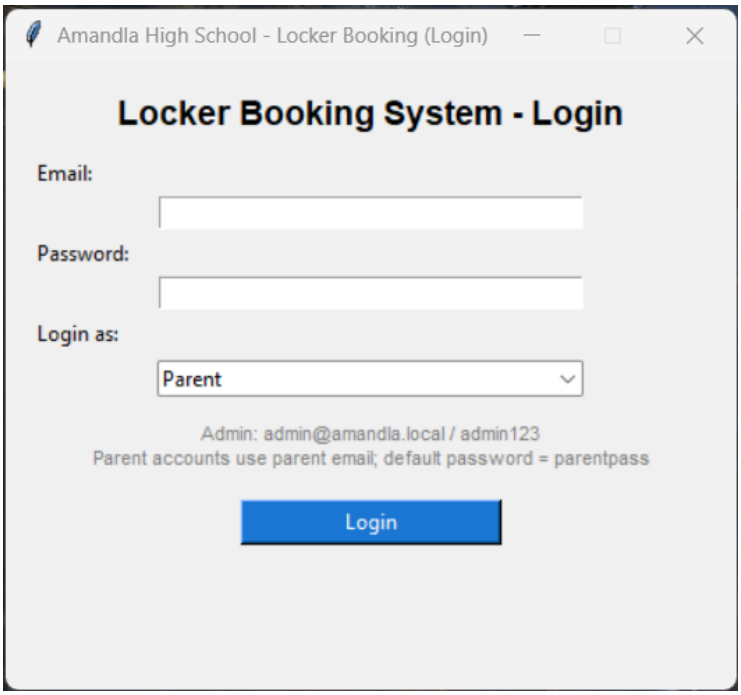
Quick Start Guide

This guide explains how parents and administrators interact with the Amandla High School Locker Booking System, including login, booking, and admin management processes.

Overview of System Functionality

The locker booking system is designed around user roles:

- Parents: Register → Log in → Apply for locker → Receive confirmation
- Admins: Manage lockers → Assign bookings → Monitor payments → Generate reports



Process | Login

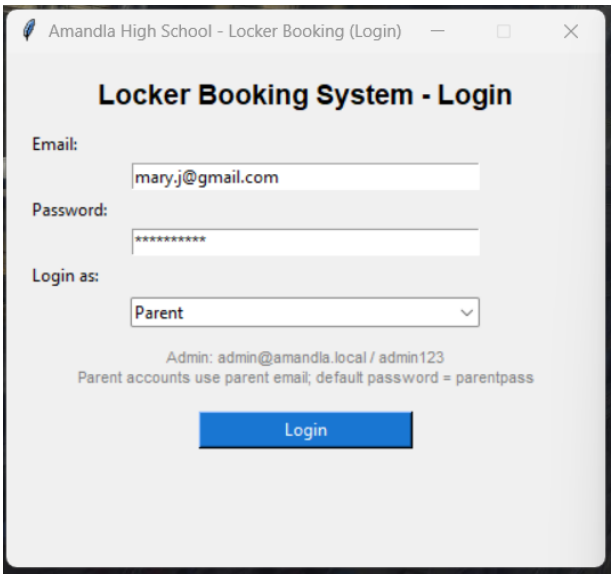
The login process allows both parents and administrators to securely access the Amandla High School Locker Booking System using their unique credentials.

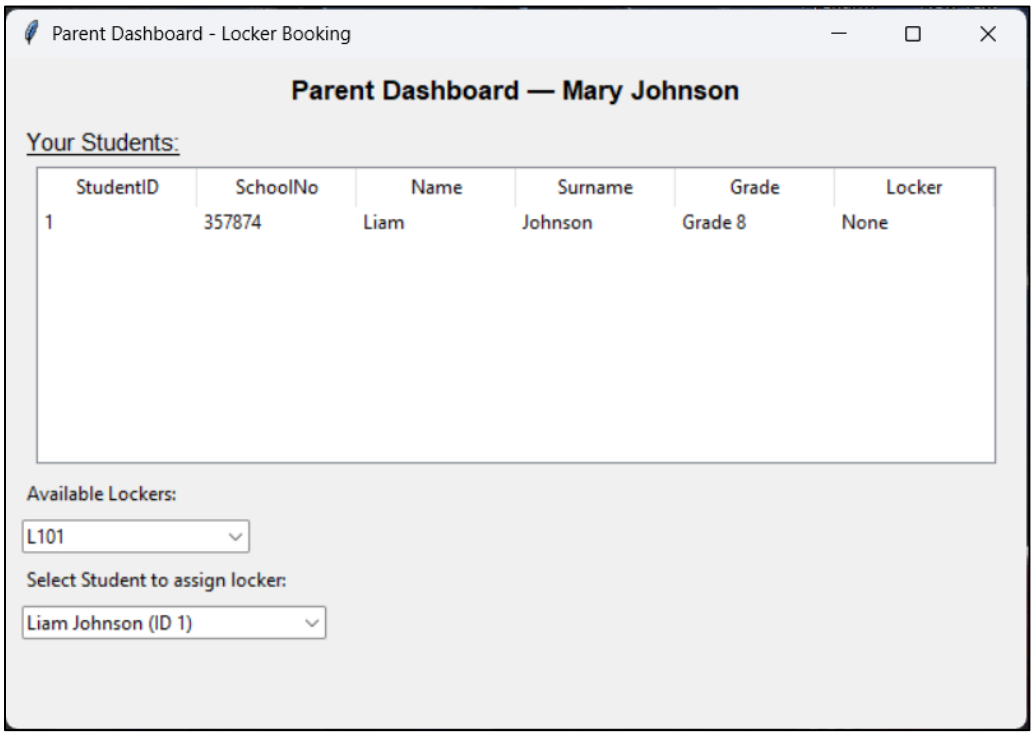
After logging in, users are automatically redirected to their respective dashboards based on their selected role.

Steps of Login Process:

1. Open the System
    - Double-click the system icon or run the main.py file to launch the application.
2. Enter Your Email and Password
    - Input the credentials that were created or assigned during registration.
    - Default credentials:
        - Administrator: admin@amandla.local / admin123
        - Parent: use any Parent Email from the dataset (password = 'parentpass' for all parents)
3. Select Your Role
    - Choose “Parent” or “Administrator” from the role selection dropdown.
4. Click Login
    - Press the Login button to authenticate your credentials.
5. If Successful:
    - Parents are redirected to the Locker Booking Page where they can view and book available lockers.
    - Administrators are redirected to the Locker Management Dashboard where they can manage locker allocations and payment records.

Screen Capture:





Admin Payment Tracking

The Payment Tracking process enables administrators to monitor and manage the R100 locker fee payments for all students. This functionality ensures financial accountability and fair locker distribution by allowing administrators to update, track, and manage each student's payment status in real time.

Steps of Process:

1. Login as Administrator
    - The admin logs in through the main login screen using admin credentials.
    - They enter their registered email and password, then select the "Administrator" role before clicking Login.
    - Upon successful login, the administrator is directed to the Locker Management & Payment Tracking Dashboard.

2. Access the "Locker Management & Payment Tracking" Page
    - On the admin dashboard, the administrator selects the Payment Tracking section from the navigation menu.
    - The system loads a data table containing all student locker records, including names, grades, locker numbers, and current payment statuses.

3. View Student Records
    - The admin reviews each record to identify students marked as Paid or Unpaid.

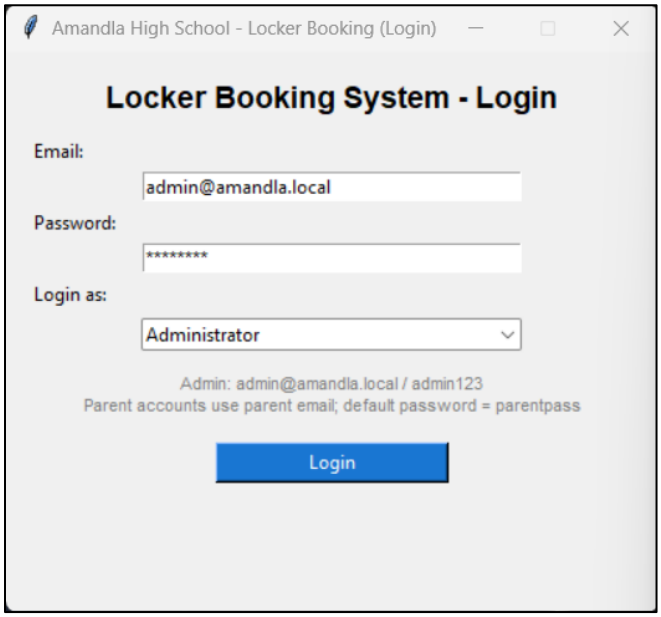
4. Select a Student Record
    - The admin clicks on a student's row to update their payment details.

5. Update Payment Status
    - From the dropdown menu, the admin selects the correct payment status:
        - Paid
        - Unpaid

6. Save the Update
    - The admin clicks "Update Status", and the system immediately updates the table and confirms the change with a message box.

7. Monitor and Manage Outstanding Payments
    - Admins can take further action:
        - Remind parents of unpaid fees.
        - Suspend lockers for students with long overdue payments.

Screen Captures:



1. Business Plan

SOFTWARE DEVELOPMENT PROPOSAL | Business Plan

PREPARED FOR: Amandla High School

PREPARED BY: Nishay Hira 69724806

DATE: 02 October 2025

Dear Amandla High School Representative,

RE: Enclosed Software Development Proposal

I am a student currently pursuing a qualification in Software Development and ICT Systems. I have undertaken a practical project focused on developing a Locker Booking System tailored specifically for school environments such as yours.

This proposal outlines the business rationale, system features, technical approach, and anticipated benefits of implementing the system at Amandla High School. I believe that this solution directly addresses the challenges faced by schools in managing locker allocation, payment tracking, and parent communication.

Throughout my training, I have gained hands-on experience in system analysis, user interface design, coding, testing, and deployment. This project serves both as an academic milestone and a practical demonstration of my capability to deliver real-world software solutions.

I am confident that the proposed system will offer your school:

- Streamlined locker management,
- Transparent payment tracking,
- Improved parent and administrator experience,
- And a digital transformation of a traditionally manual process.

Please find the business plan and system proposal enclosed for your review.
I look forward to your feedback and the opportunity to support your school in adopting this solution.

Kind regards,
Nishay Hira
Student Number: 69724806
Email: 69724806@mylife.unisa.ac.za
Phone: [Optional]

System Overview

The proposed system is a Locker Booking and Management System developed specifically for Amandla High School. The main aim of the system is to digitize and streamline the process of booking, assigning, and managing student lockers, improving the overall efficiency for both administrators and students. Additionally, the system will include payment tracking features, parent notifications, and locker availability status, reducing administrative workload and minimizing manual errors.

System Aims:

- Automate the locker allocation process.
- Allow students (or parents) to book lockers online.
- Provide real-time locker availability and booking status.
- Track payments and generate receipts.
- Notify parents or students via email or SMS after a booking or payment.
- Provide admin with a dashboard for monitoring locker usage and status.

Scope of the System:

- Web-based system accessible via any browser on desktop or mobile.
- Admin dashboard for school staff.
- Booking portal for students and/or parents.
- Simple, intuitive interface designed for ease of use.

Design Process:

The system was designed using an Agile approach, starting with a detailed requirement analysis followed by iterative design and development stages. Regular testing and user feedback ensured that each module met user expectations.
The development followed this lifecycle:

1. Requirement gathering
2. Wireframing and UI/UX design
3. Backend and frontend development
4. Integration and testing
5. Deployment and documentation

Functionality:

- User authentication (admin, student, parent)
- Locker search and booking
- Locker management by admin (add/edit/delete lockers)
- Payment recording system
- Email/SMS notifications
- Reporting tools for admins

Technology Choices:

- Frontend: HTML5, CSS3, JavaScript (React or plain JavaScript)
- Backend: PHP or Node.js
- Database: MySQL
- Hosting: Cloud-based or local school server
- Notification API: Email API, SMS API

Software Distribution:

The software will be deployed as a web application accessible via a school intranet or the public internet depending on the school's preference. It will be hosted on a secure server (cloud or local) with backup and user access control.

Capacity for Users:

- Scalable system designed to handle up to 500 concurrent users.
- Admin dashboard optimized for use by up to 10 admin/staff members.
- Student/Parent portal designed for thousands of user accounts.

Ownership of Intellectual Property:

All intellectual property rights of the software, source code, and related documentation will belong to the developer (Nishay Hira, 69724806: 076 831 6287) unless otherwise agreed upon in writing with Amandla High School.

Testing and Support:

- Functional Testing: Ensures each feature works as intended.
- User Acceptance Testing (UAT): Will be performed with school staff and a sample of student/parent users.
- Bug Tracking and Fixes: A test period of 14 days post-deployment will be offered to resolve any technical issues.
- Support: Basic documentation and training will be provided. Optional support contract can be arranged for ongoing maintenance.

Warranties:

- A 30-day warranty post-deployment will be provided for bug fixes and basic support.
- Additional support and maintenance plans can be discussed separately.

Technology requirements

Hardware

The system will run on all modern desktop and laptop computers with access to a web browser.

Compatible Devices:

- Windows PCs (Windows 8, 10, 11)
- macOS computers
- Android phones/tablets
- iPhones
- iPads

Recommended Specifications:

- Minimum 2GB RAM,
- 1.5GHz processor,
- Internet connection (for web-based access).

Software

- Frontend Development: HTML5, CSS3, JavaScript (possibly React)

- Backend Development: PHP 8+ or Node.js
- Database: MySQL 5.7+ or MariaDB
- Hosting: Apache or Nginx web server
- Development Tools: Visual Studio Code, XAMPP/WAMP (for testing), Git
- Notification Services: Email API (SendGrid), optional SMS (Twilio)
- Browser Compatibility: Google Chrome, Firefox, Microsoft Edge, Safari

Obstacles

Potential Risks and Challenges:

1. Internet Dependency
    - Risk: If the system is hosted online, a stable internet connection is required.
    - Solution: Offer an offline/local server version as a backup.
2. User Resistance to Change
    - Risk: Staff or students may be unfamiliar with the new system.
    - Solution: Provide short training sessions and easy-to-read user manuals.
3. Data Loss or Breach
    - Risk: As with any digital system, there is a risk of data breach or accidental loss.
    - Solution: Implement secure login, daily backups, and encrypted data storage.
4. Integration with Existing Systems
    - Risk: Difficulty integrating with existing payment or school admin systems.
    - Solution: Develop flexible APIs for future integration and maintain manual import/export features as a fallback.
5. Budget Constraints
    - Risk: School may have limited funds for extended features or support.
    - Solution: Modular design allows for basic features to be implemented first, with optional add-ons later.

Deployment

Installation

The Locker Booking System will be deployed using one of the following methods, depending on the school's preference:

1. Web-Based (Online) Deployment:
    - The system will be uploaded to a secure web hosting platform
    - Access will be provided via a URL (e.g., https://lockers.amandlahighschool.co.za).
    - Admin login credentials will be provided to designated staff members.
    - Updates and patches will be handled remotely and scheduled outside school hours to avoid disruptions.
2. Local Server (Offline) Deployment:
    - The system will be installed on a Windows-based school server using XAMPP (Apache + MySQL).
    - A step-by-step installation guide will be provided.
    - The software will be accessible on all school computers via the school intranet.
    - Updates will be manually pushed through USB or LAN by the developer or IT staff.

Updates:

- Critical updates (security, bug fixes) will be made available as downloadable .zip packages.
- Optional feature updates can be requested and scheduled.
- Notification of available updates will be emailed to the school's IT coordinator.

Training

Initial Training: A 1-hour on-site training session will be provided to school staff (admin and IT personnel).
User Manuals: A detailed user guide (PDF) will be supplied to all admin users.
Post-Signoff Support:

- A 14-day support period after system sign-off.
- Support will be provided via email and WhatsApp (Monday–Friday, 9am–4pm).
- Additional training and long-term support can be negotiated separately if needed.

Testing

Developer Testing:

- Unit testing and integration testing performed during development.
- All key features (booking, admin login, locker management, notifications, reporting) tested thoroughly.

Client Testing (User Acceptance Testing):

- The client (school) will be asked to test the following:
    - Booking a locker from the front end.
    - Assigning lockers via the admin dashboard.

    - Running a report and exporting data.
- A test user login will be provided to simulate student/parent interaction.
- Feedback will be collected and necessary changes will be made before the final sign-off.

Backup and recovery

Automated Backups:

- Daily backups of the database (locker assignments, user data, payments).
- Backups will be stored securely on the school's server (or hosting platform).
- Retention period: 30 days of backups.

Recovery:

- A built-in recovery module will allow the system admin to restore the last backup manually.
- In case of a complete system failure, a full system reinstall with data restoration can be done within 48 hours.

Documentation

The following documentation will be provided with the system:

Document	**Format**	**Delivery Method**
Installation Guide	PDF	Printed & USB
Admin User Manual	PDF	Printed & Online
Technical System Documentation	PDF	USB (for IT department)
Quick Start Guide	PDF	Printed
Error Handling and Troubleshooting	PDF	Online (linked in system)
Reporting Guide	PDF	Printed & Online

(You will include the final documentation when you submit your system for evaluation – now only required to list what you are going to give.)

Reporting

The Management Information System (MIS) Reporting Module in the Amandla High School Locker Booking System enables administrators to generate reports that support decision-making, accountability, and transparency.

The system produces the following key reports:

1. Payment Status Report

- Displays a summary of students who have paid and students who have not paid the R100 locker fee.
- Provides totals for:
    - Number of paid lockers
    - Number of unpaid lockers

    - Percentage of total payments completed
- Output: Tabular view with export options (CSV or PDF).

2. Locker Allocation Report

- Lists all allocated lockers, showing student names, grades, locker numbers, and current status (Active, Suspended, or Available).
- Helps the administrator monitor usage and identify available lockers.

3. Parent Contact Report

- Extracts all parent contact details for quick communication regarding locker issues or outstanding payments.
- Includes fields such as Parent Name, Email, Phone Number, and Associated Student.

4. Financial Summary Report

- Summarises the total amount collected from locker payments.
- Displays:
    - Number of lockers booked
    - Amount collected (R100 × Paid Lockers)
    - Outstanding balance (R100 × Unpaid Lockers)

5. Suspended Locker Report

- Lists all lockers temporarily suspended due to non-payment.
- Includes reasons for suspension and date of status change.

Each report can be filtered by grade, payment status, or locker number, and exported for record-keeping or presentation to school management.

Evidence

The following graphical user interfaces (GUIs) have been developed to demonstrate progress on the system.
Screenshots of each interface have been included to illustrate the design, functionality, and user experience.

Parent Dashboard (Locker Booking Page)

Purpose:

Allows parents to view available lockers and book one for their child.

Information Displayed:

- Student name and grade
- Locker number and availability
- Booking confirmation message

Actions Available:

- Select an available locker
- Click “Book Locker” to reserve

Administrator Dashboard – Locker Management and Payment Tracking

Purpose

This GUI provides administrators with tools to manage all lockers and monitor student payment statuses.
It helps ensure that lockers are only assigned to students who have paid the required R100 locker fee.

Description:

- Displays a table of students, their grade, assigned locker, and payment status.
- Allows the admin to:
  - View current locker assignments
  - Update a student’s payment status (Paid / Unpaid)
  - Export reports (future enhancement)
- The Update Status button allows immediate updates to selected student entries.
- Logout button safely exits the admin dashboard.

Information Displayed:

- Student Name
- Grade
- Locker Number
- Payment Status (Paid / Unpaid)

Visual Design and Look & Feel

The GUIs follow a simple, modern interface:

- Consistent school theme colors (blue for admin, green for parents)
- Rounded buttons and clear labels
- Easy-to-read font (Helvetica)
- Logical grouping of information
- User-friendly navigation with confirmation popups

This ensures that both parents and administrators can easily interact with the system, even with minimal computer experience.

Evidence

At this stage of development:

- The Login Interface (not shown here per requirements) successfully directs users to their correct dashboards.
- The Parent Dashboard and Admin Dashboard GUIs have been created and are fully functional for demonstration purposes.
- Next steps will include integrating a database for data storage and reporting.

Name: Nishay Hira

Student Number: 69724806

Date: 02 October 2025

Signature: 

2. Assessment 3

Question 1 [20]

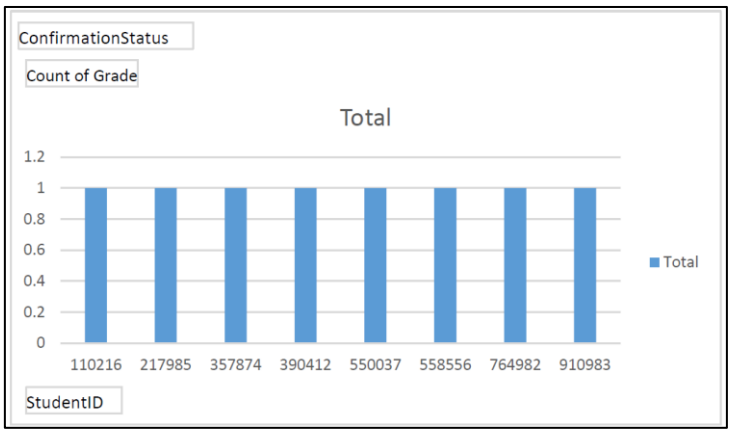
Management Information Systems [MIS] Reports

Create the following MIS Reports
Refer to Task 10.

1.1 Locker Usage by Grade Report (10)
1.1.1 Locker usage for Grade 8 and Grade 11

```
SELECT
  s.Grade,
  COUNT(b.BookingID) AS LockersBooked,
  (SELECT COUNT(*) FROM Locker l WHERE l.Grade = s.Grade) AS TotalLockers,
  ROUND(COUNT(b.BookingID) * 100.0 / (SELECT COUNT(*) FROM Locker l
WHERE l.Grade = s.Grade), 2) AS PercentageUsed
FROM
  Booking b
JOIN
  Student s ON b.StudentID = s.StudentID
WHERE
  s.Grade IN (8, 11)
  AND b.ConfirmationStatus = 'Confirmed'
GROUP BY
  s.Grade;
```

(Dashboard)



Why this report is useful:

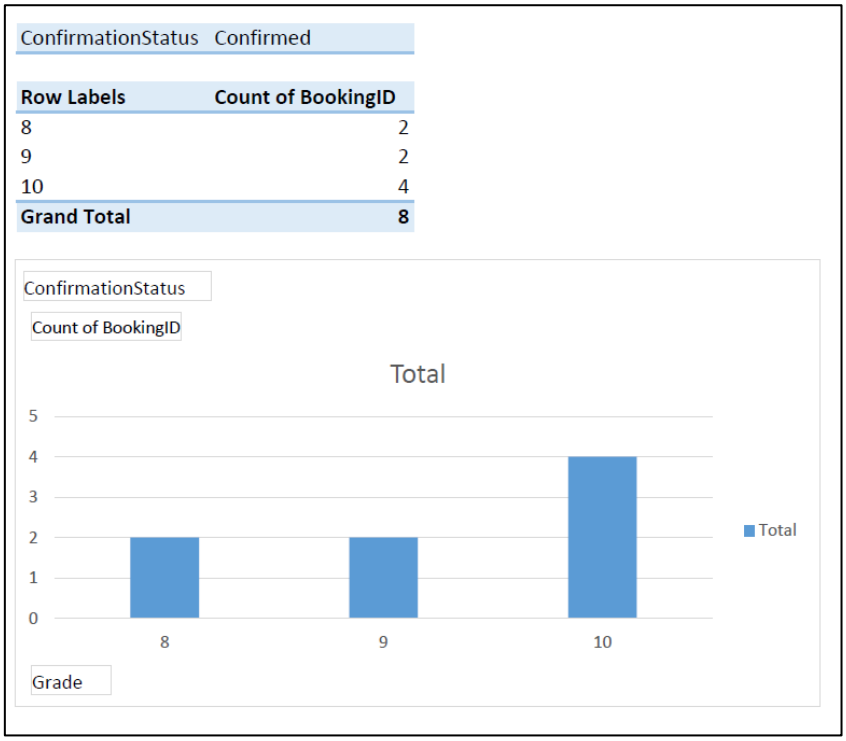
- It helps the school monitor locker allocation efficiency by grade.
- Identifies if any grade is under or over-utilizing lockers.
- Assists in planning for locker capacity needs (e.g., if Grade 8 lockers are fully booked but Grade 12 aren't).
- Helps ensure fair locker distribution among students.

1.2 Locker Booking Summary Report (10)

1.2.1 Summarize the number of lockers booked for period January 2026 to June 2026

```
SELECT
  COUNT(*) AS TotalLockersBooked
FROM
  Booking
WHERE
  BookingDate BETWEEN '2026-01-01' AND '2026-06-30'
  AND ConfirmationStatus = 'Confirmed';
```

Dashboard:



Why this report is useful

- Provides the school with a quick overview of booking activity in the first half of 2026.
- Helps evaluate the demand and booking patterns for lockers.
- Supports budgeting and operational planning, e.g., staffing or locker maintenance.
- Can be used to forecast future locker needs based on historical booking data.

Question 2 [20]

Graphical User Interfaces

Create the following MIS Reports
Refer to Task 11.

It is now time to start with the “coding” and create the front-end Graphical User Interfaces (GUI).

1. Create a Graphical User Interface for the parent, or the administrator to login to the system.
2. When you are done, make a screenshot of the interface, and also provide the code for the GUI.

Code:

```
import tkinter as tk
from tkinter import messagebox, ttk

# --------------------------------
# Function to open Parent Dashboard
# --------------------------------
def create_db_and_import():
    # If DB exists, skip creation step but ensure tables present
    conn = sqlite3.connect(DB_FILE)
    c = conn.cursor()

    # Create tables
    c.execute('''
        CREATE TABLE IF NOT EXISTS parents (
            parent_id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT,
            id_number TEXT,
            first_name TEXT,
            last_name TEXT,
            email TEXT UNIQUE,
            address TEXT,
            phone TEXT,
            password TEXT
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS students (
            student_id INTEGER PRIMARY KEY AUTOINCREMENT,
            school_number TEXT,
            first_name TEXT,
            last_name TEXT,
```

```
            grade TEXT,
            parent_id INTEGER,
            FOREIGN KEY(parent_id) REFERENCES parents(parent_id)
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS lockers (
            locker_id INTEGER PRIMARY KEY AUTOINCREMENT,
            locker_number TEXT UNIQUE,
            status TEXT DEFAULT 'Available',
            assigned_student_id INTEGER
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS bookings (
            booking_id INTEGER PRIMARY KEY AUTOINCREMENT,
            parent_id INTEGER,
            student_id INTEGER,
            locker_id INTEGER,
            booking_date TEXT,
            payment_status TEXT DEFAULT 'Unpaid',
            FOREIGN KEY(parent_id) REFERENCES parents(parent_id),
            FOREIGN KEY(student_id) REFERENCES students(student_id),
            FOREIGN KEY(locker_id) REFERENCES lockers(locker_id)
        )
    ''')

    # Admins (one default admin)
    c.execute('''
        CREATE TABLE IF NOT EXISTS admins (
            admin_id INTEGER PRIMARY KEY AUTOINCREMENT,
            email TEXT UNIQUE,
            password TEXT
        )
    ''')

    # Insert default admin user if not exists
    c.execute("SELECT COUNT(*) FROM admins WHERE email = ?",
("admin@amandla.local",))
    if c.fetchone()[0] == 0:
        c.execute("INSERT INTO admins (email, password) VALUES (?,?)",
("admin@amandla.local", "admin123"))

    # Import data from DATA_TEXT
    reader = csv.DictReader(DATA_TEXT.strip().splitlines(), delimiter='\t')
    parents_count = 0
    students_count = 0
    for row in reader:
```

```
        # Insert parent (password default 'parentpass')
        try:
            c.execute('''
                INSERT OR IGNORE INTO parents (title, id_number, first_name,
last_name, email, address, phone, password)
                VALUES (?,?,?,?,?,?,?,?)
            ''', (
                row['Parent Title'],
                row['Parent ID Number'],
                row['Parent Name'],
                row['Parent Surname'],
                row['Parent Email Address'],
                row['Parent Home Address'],
                row['Parent Phone Number'],
                "parentpass"
            ))
        except Exception as e:
            print("Parent insert error:", e)
        # get parent id
        c.execute("SELECT parent_id FROM parents WHERE email = ?",
(row['Parent Email Address'],))
        parent_id = c.fetchone()[0]

        # Insert student
        try:
            c.execute('''
                INSERT INTO students (school_number, first_name, last_name,
grade, parent_id)
                VALUES (?,?,?,?,?)
            ''', (
                row['Student School Number'],
                row['Student Name'],
                row['Student Surname'],
                row['Student Grade'],
                parent_id
            ))
        except Exception as e:
            print("Student insert error:", e)

    # Pre-create some locker records (L100 - L140)
    for i in range(100, 141):
        locker_number = f"L{i}"
        c.execute("INSERT OR IGNORE INTO lockers (locker_number) VALUES (?)",
(locker_number,))

    conn.commit()
    conn.close()
```

```
# Initialize DB & data
create_db_and_import()

# --- GUI ---
root = tk.Tk()
root.title("Amandla High School - Locker Booking (Login)")
root.geometry("420x360")
root.resizable(False, False)

# Frame layout
frame = tk.Frame(root, padx=15, pady=15)
frame.pack(expand=True, fill="both")

title = tk.Label(frame, text="Locker Booking System - Login",
font=("Helvetica", 14, "bold"))
title.pack(pady=(0,10))

# Email
tk.Label(frame, text="Email:").pack(anchor="w")
email_entry = tk.Entry(frame, width=40)
email_entry.pack(pady=3)

# Password
tk.Label(frame, text="Password:").pack(anchor="w")
password_entry = tk.Entry(frame, width=40, show="*")
password_entry.pack(pady=3)

# Role
tk.Label(frame, text="Login as:").pack(anchor="w")
role_var = tk.StringVar(value="Parent")
role_combo = ttk.Combobox(frame, textvariable=role_var, values=["Parent",
"Administrator"], state="readonly", width=37)
role_combo.pack(pady=5)

info_label = tk.Label(frame, text="Admin: admin@amandla.local /
admin123\nParent accounts use parent email; default password = parentpass",
font=("Helvetica", 8), fg="gray")
info_label.pack(pady=(6,8))

# Login function
def login():
    email = email_entry.get().strip()
    password = password_entry.get().strip()
    role = role_var.get()

    conn = sqlite3.connect(DB_FILE)
    c = conn.cursor()
```

```
    if role == "Administrator":
        c.execute("SELECT password FROM admins WHERE email = ?", (email,))
        r = c.fetchone()
        conn.close()
        if r and r[0] == password:
            messagebox.showinfo("Login", "Administrator login successful")
            open_admin_dashboard()
        else:
            messagebox.showerror("Login Failed", "Invalid admin credentials")
    else:  # Parent
        c.execute("SELECT parent_id, password, first_name FROM parents WHERE
email = ?", (email,))
        r = c.fetchone()
        conn.close()
        if r and r[1] == password:
            messagebox.showinfo("Login", f"Parent login successful. Welcome
{r[2]}!")
            open_parent_dashboard(parent_email=email)
        else:
            messagebox.showerror("Login Failed", "Invalid parent credentials")

login_btn = tk.Button(frame, text="Login", command=login, width=20,
bg="#1976D2", fg="white")
login_btn.pack(pady=6)

# ---------------- Parent Dashboard ----------------
def open_parent_dashboard(parent_email):
    conn = sqlite3.connect(DB_FILE)
    c = conn.cursor()
    c.execute("SELECT parent_id, first_name, last_name FROM parents WHERE
email = ?", (parent_email,))
    parent = c.fetchone()
    if not parent:
        messagebox.showerror("Error", "Parent record not found")
        conn.close()
        return
    parent_id = parent[0]
    parent_name = f"{parent[1]} {parent[2]}"

    pwin = tk.Toplevel(root)
    pwin.title("Parent Dashboard - Locker Booking")
    pwin.geometry("640x420")

    tk.Label(pwin, text=f"Parent Dashboard — {parent_name}",
font=("Helvetica", 13, "bold")).pack(pady=8)
    tk.Label(pwin, text="Your Students:", font=("Helvetica", 11,
"underline")).pack(anchor="w", padx=10)
```

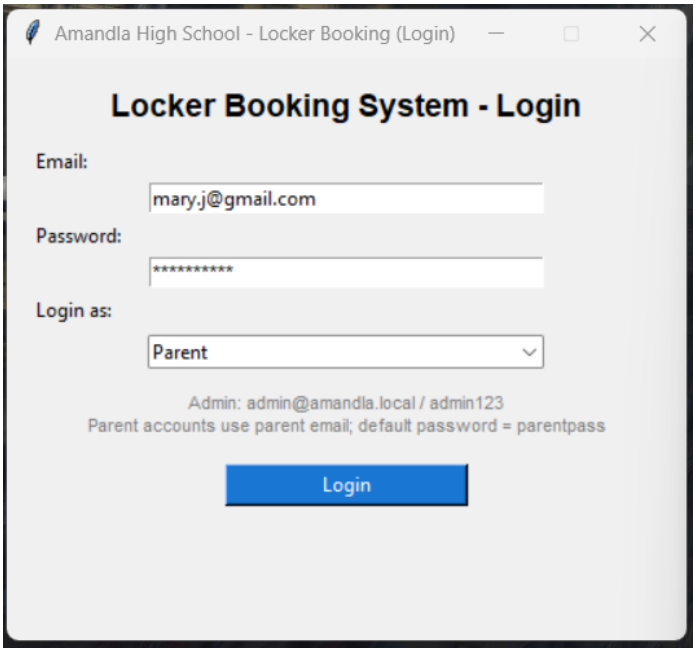
```
    # Student list
    cols = ("StudentID", "SchoolNo", "Name", "Surname", "Grade", "Locker")
    tree = ttk.Treeview(pwin, columns=cols, show="headings", height=8)
    for col in cols:
        tree.heading(col, text=col)
        tree.column(col, width=100)
    tree.pack(padx=10, pady=(4,8))

    c.execute("SELECT s.student_id, s.school_number, s.first_name,
s.last_name, s.grade, l.locker_number FROM students s LEFT JOIN bookings b ON
s.student_id=b.student_id LEFT JOIN lockers l ON b.locker_id=l.locker_id WHERE
s.parent_id = ?", (parent_id,))
    rows = c.fetchall()
    for r in rows:
        tree.insert("", "end", values=r)

    # Locker selection
    tk.Label(pwin, text="Available Lockers:").pack(anchor="w", padx=10)
    c.execute("SELECT locker_id, locker_number FROM lockers WHERE
status='Available' LIMIT 50")
    lockers = c.fetchall()
    locker_map = {f"{ln}": lid for lid, ln in lockers}
    locker_vals = list(locker_map.keys())

    locker_var = tk.StringVar()
    locker_combo = ttk.Combobox(pwin, textvariable=locker_var,
values=locker_vals, width=20)
    locker_combo.pack(padx=10, pady=6, anchor="w")

    # Student select
    tk.Label(pwin, text="Select Student to assign locker:").pack(anchor="w",
padx=10)
    student_var = tk.StringVar()
    student_combo = ttk.Combobox(pwin, textvariable=student_var,
values=[f"{r[2]} {r[3]} (ID {r[0]})" for r in rows], width=28)
    student_combo.pack(padx=10, pady=6, anchor="w")

    def confirm_booking():
        sel_locker = locker_var.get()
        sel_student = student_var.get()
        if not sel_locker or not sel_student:
            messagebox.showerror("Missing", "Select a locker and a student")
            return
        # parse student id
        sid = int(sel_student.split("ID")[-1].strip(" )"))
        lid = locker_map.get(sel_locker)
        if not lid:
```

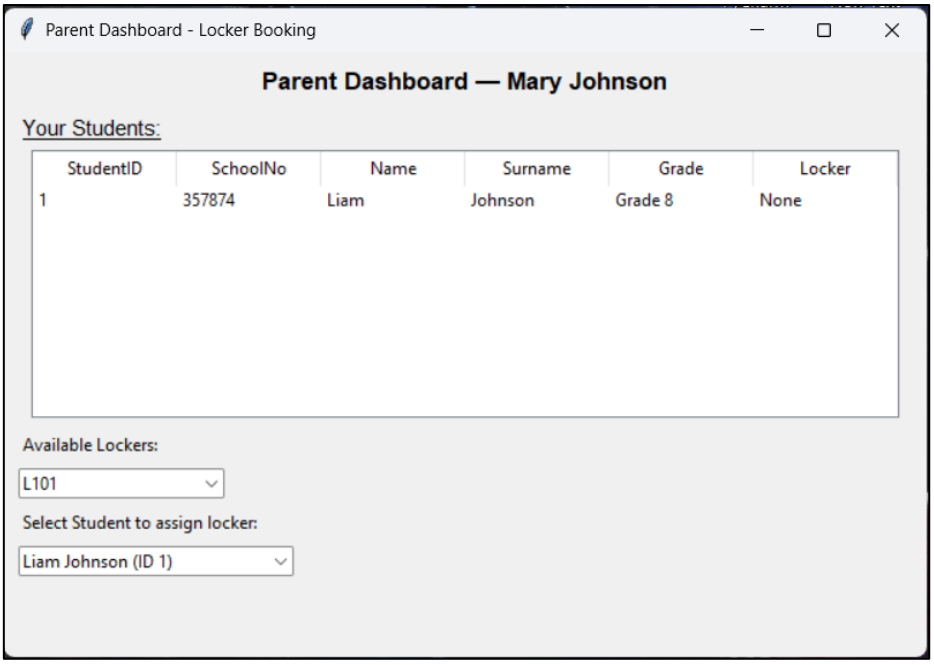
```
            messagebox.showerror("Error", "Locker selection invalid or taken.
Refresh and try again.")
            return
        conn2 = sqlite3.connect(DB_FILE)
        c2 = conn2.cursor()
        # create booking
        c2.execute("INSERT INTO bookings (parent_id, student_id, locker_id,
booking_date, payment_status) VALUES (?,?,?,?,?)",
                   (parent_id, sid, lid, date.today().isoformat(), "Unpaid"))
        # mark locker
        c2.execute("UPDATE lockers SET status=?, assigned_student_id=? WHERE
locker_id=?", ("Booked", sid, lid))
        conn2.commit()
        conn2.close()
        messagebox.showinfo("Booked", f"{sel_locker} booked for student.")
        pwin.destroy()

    tk.Button(pwin, text="Confirm Booking", command=confirm_booking,
bg="#2E7D2", fg="white").pack(pady=10, anchor="w", padx=10)

    tk.Button(pwin, text="Logout", command=pwin.destroy, bg="#C62828",
fg="white").pack(side="bottom", pady=10)

    conn.close()

# ---------------- Admin Dashboard ----------------
def open_admin_dashboard():
    awin = tk.Toplevel(root)
    awin.title("Admin Dashboard - Locker Management & Payments")
    awin.geometry("900x520")

    tk.Label(awin, text="Administrator Dashboard", font=("Helvetica", 14,
"bold")).pack(pady=8)

    # Table of all students and payment status
    cols = ("StudentID", "Student", "Grade", "Parent", "Locker", "Payment")
    tree = ttk.Treeview(awin, columns=cols, show="headings", height=18)
    for col in cols:
        tree.heading(col, text=col)
        tree.column(col, width=140 if col!="StudentID" else 80)
    tree.pack(padx=10, pady=6)

    conn = sqlite3.connect(DB_FILE)
    c = conn.cursor()
    c.execute('''
        SELECT s.student_id, s.first_name || ' ' || s.last_name, s.grade,
               p.first_name || ' ' || p.last_name, l.locker_number,
               COALESCE(b.payment_status, 'Unpaid')
```

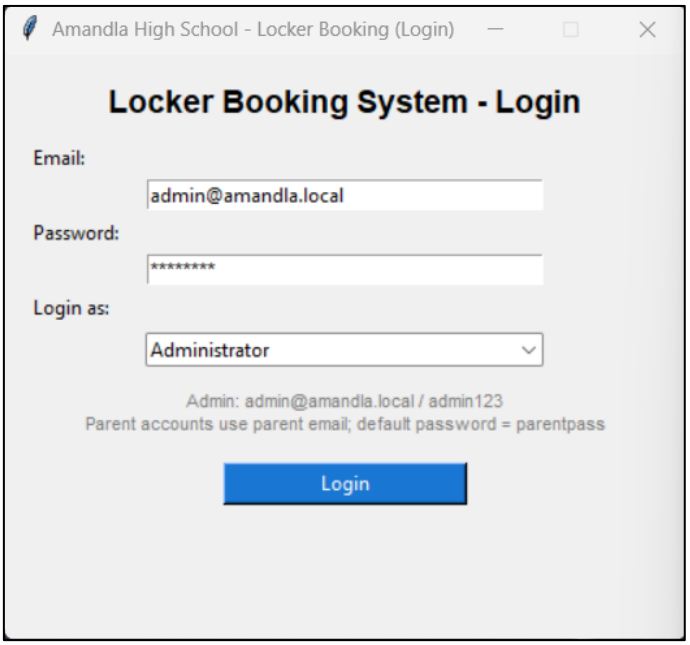
```
        FROM students s
        LEFT JOIN parents p ON s.parent_id = p.parent_id
        LEFT JOIN bookings b ON s.student_id = b.student_id
        LEFT JOIN lockers l ON b.locker_id = l.locker_id
        ORDER BY s.student_id
    ''')
    for row in c.fetchall():
        tree.insert("", "end", values=row)
    conn.close()

    # payment update controls
    ctrl_frame = tk.Frame(awin)
    ctrl_frame.pack(pady=8)

    tk.Label(ctrl_frame, text="New Payment Status:").grid(row=0, column=0,
padx=6)
    status_var = tk.StringVar(value="Paid")
    status_combo = ttk.Combobox(ctrl_frame, textvariable=status_var,
values=["Paid", "Unpaid"], state="readonly", width=12)
    status_combo.grid(row=0, column=1, padx=6)

    def update_payment_status():
        sel = tree.selection()
        if not sel:
            messagebox.showerror("Select", "Please select a student row to
update")
            return
        new_status = status_var.get()
        item = tree.item(sel[0])['values']
        student_id = item[0]
        # find booking for that student
        conn2 = sqlite3.connect(DB_FILE)
        c2 = conn2.cursor()
        c2.execute("SELECT booking_id FROM bookings WHERE student_id = ?",
(student_id,))
        r = c2.fetchone()
        if r:
            booking_id = r[0]
            c2.execute("UPDATE bookings SET payment_status=? WHERE
booking_id=?", (new_status, booking_id))
            conn2.commit()
            messagebox.showinfo("Updated", "Payment status updated")
        else:
            messagebox.showwarning("No booking", "No booking record for this
student. Create a booking first if appropriate.")
        conn2.close()
        # refresh table
        awin.destroy()
```

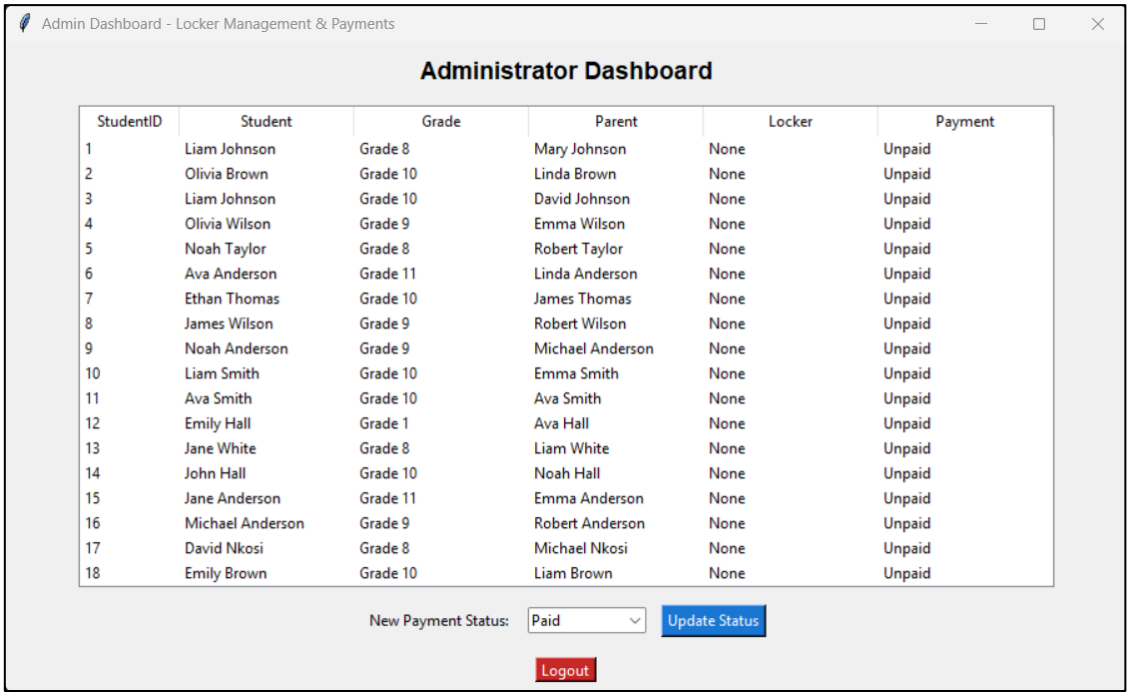
```
        open_admin_dashboard()

    tk.Button(ctrl_frame, text="Update Status", command=update_payment_status,
bg="#1976D2", fg="white").grid(row=0, column=2, padx=6)
    tk.Button(awin, text="Logout", command=awin.destroy, bg="#C62828",
fg="white").pack(side="bottom", pady=8)

# Start mainloop
root.mainloop()
```

Screenshots:






Amandla High School - Locker Booking (Login)
Locker Booking System - Login
Email:
admin@amandla.local
Password:
****
Login as:
Administrator
Admin: admin@amandla.local / admin123
Parent accounts use parent email; default password = parentpass
Login


Admin Dashboard - Locker Management & Payments
Administrator Dashboard
StudentID Student Grade Parent Locker Payment
1 Liam Johnson Grade 8 Mary Johnson None Unpaid
2 Olivia Brown Grade 10 Linda Brown None Unpaid
3 Liam Johnson Grade 10 David Johnson None Unpaid
4 Olivia Wilson Grade 9 Emma Wilson None Unpaid
5 Noah Taylor Grade 8 Robert Taylor None Unpaid
6 Ava Anderson Grade 11 Linda Anderson None Unpaid
7 Ethan Thomas Grade 10 James Thomas None Unpaid
8 James Wilson Grade 9 Robert Wilson None Unpaid
9 Noah Anderson Grade 9 Michael Anderson None Unpaid
10 Liam Smith Grade 10 Emma Smith None Unpaid
11 Ava Smith Grade 10 Ava Smith None Unpaid
12 Emily Hall Grade 1 Ava Hall None Unpaid
13 Jane White Grade 8 Liam White None Unpaid
14 John Hall Grade 10 Noah Hall None Unpaid
15 Jane Anderson Grade 11 Emma Anderson None Unpaid
16 Michael Anderson Grade 9 Robert Anderson None Unpaid
17 David Nkosi Grade 8 Michael Nkosi None Unpaid
18 Emily Brown Grade 10 Liam Brown None Unpaid
New Payment Status: Paid Update Status
Logout

3. Assessment 2

Question 1 [10]
Cleaning the data
1.1 Download the data file (Parent_Student_Data) from Additional Resources on the myModules 2025 Site.
Refer to Task 8 regarding cleaning data.
1.2 Follow the process to clean the data. Attached the clean data file to the end of your assessment.


Reviewed
Parent_Student_Data.x

Question 2 [20]
Create the database
2.1 Create the database structure and database tables for your system according to your ERD diagram.

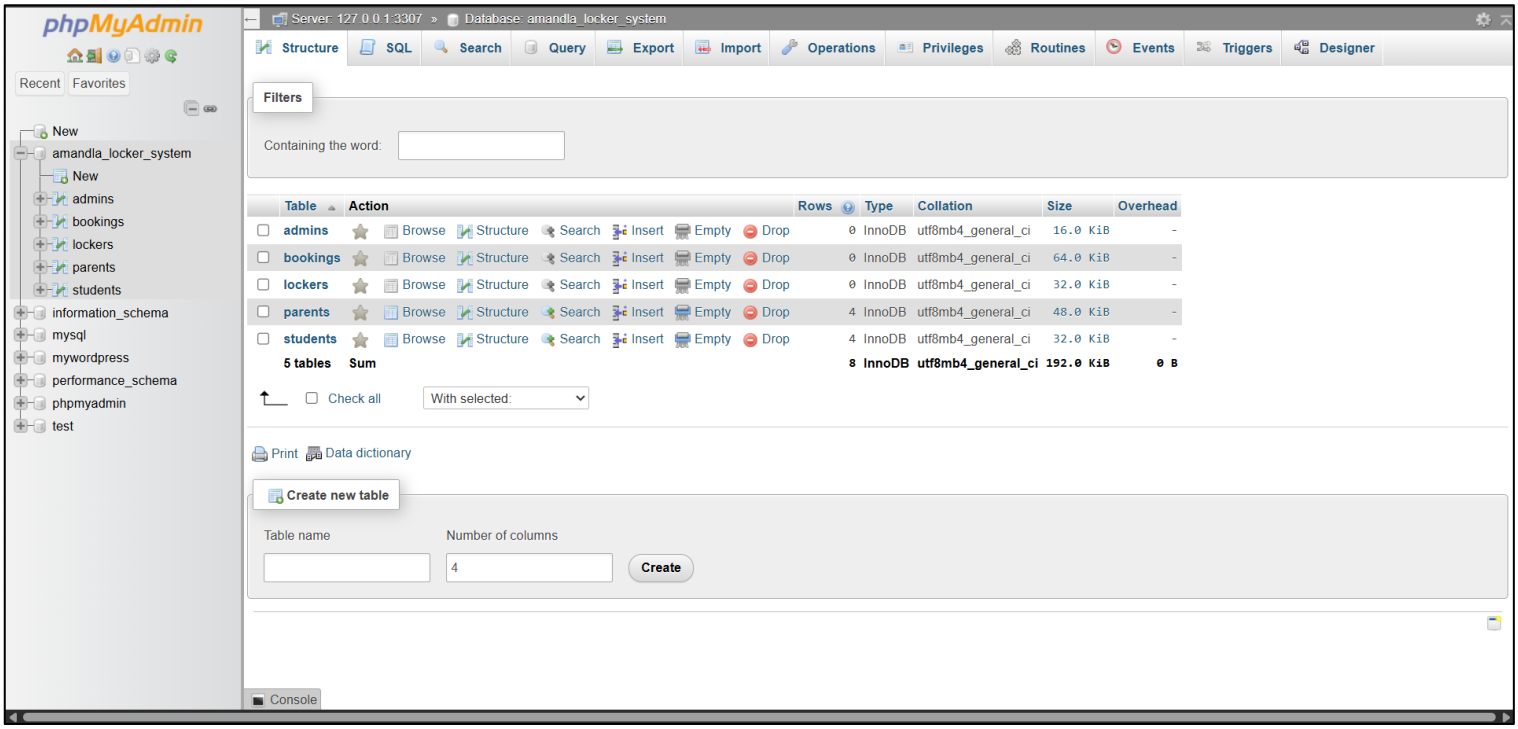
2.2 Import the data from the data file into your database.

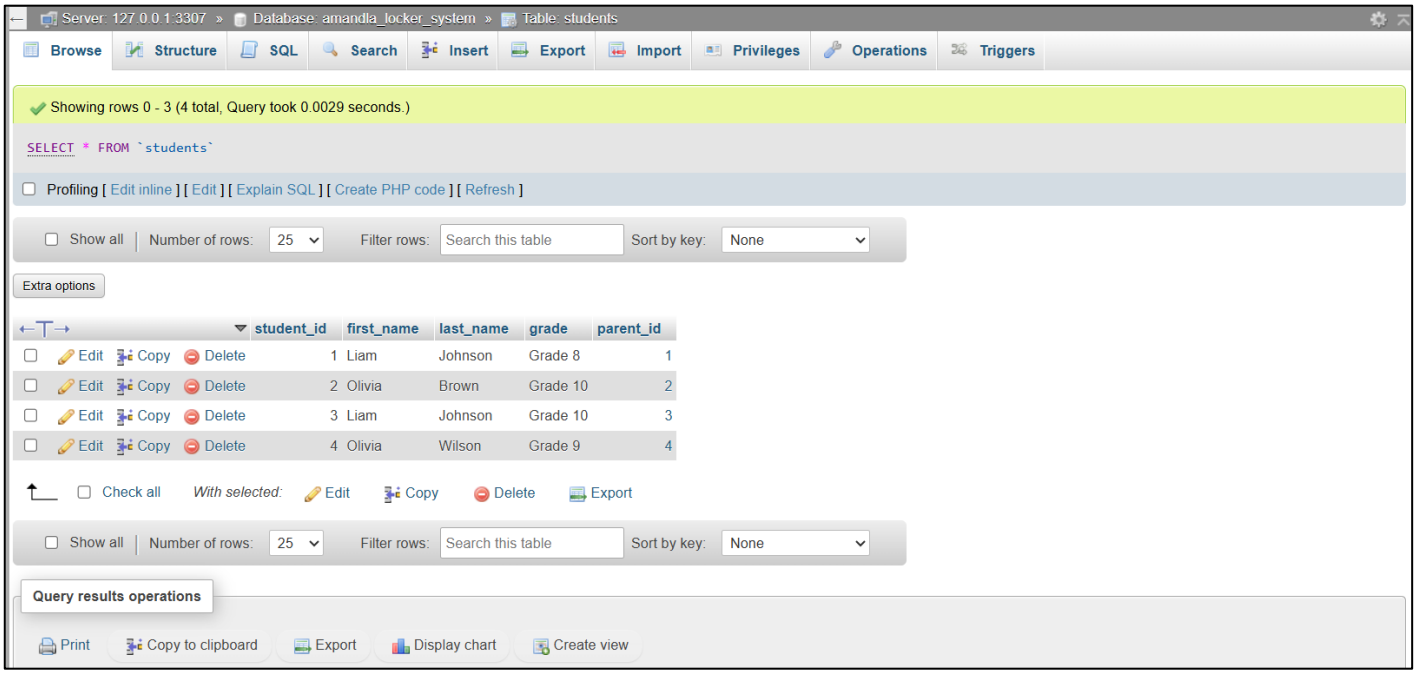
Code:

```
-- Create the database
CREATE DATABASE amandla_locker_system;
USE amandla_locker_system;

-- Parent table
CREATE TABLE parents (
    parent_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(10),
    id_number VARCHAR(20) UNIQUE,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE,
    address VARCHAR(255),
    phone_number VARCHAR(20)
);

-- Student table
CREATE TABLE students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    grade VARCHAR(20),
    parent_id INT,
    FOREIGN KEY (parent_id) REFERENCES parents(parent_id)
);

-- Locker table
CREATE TABLE lockers (
    locker_id INT AUTO_INCREMENT PRIMARY KEY,
    locker_number VARCHAR(10) UNIQUE,
    status ENUM('Available', 'Booked', 'Suspended') DEFAULT 'Available',
    location VARCHAR(50)
);

-- Booking table
CREATE TABLE bookings (
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    parent_id INT,
    student_id INT,
    locker_id INT,
    booking_date DATE,
    payment_status ENUM('Paid', 'Unpaid') DEFAULT 'Unpaid',
    FOREIGN KEY (parent_id) REFERENCES parents(parent_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (locker_id) REFERENCES lockers(locker_id)
);

-- Admin table
CREATE TABLE admins (
    admin_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(100)
);
```

Screenshots:





4. Assessment 1

Question 1 [4]
Programming Languages (3)
Identified the programming languages that you will use to develop your system.

HTML, CSS and PHP

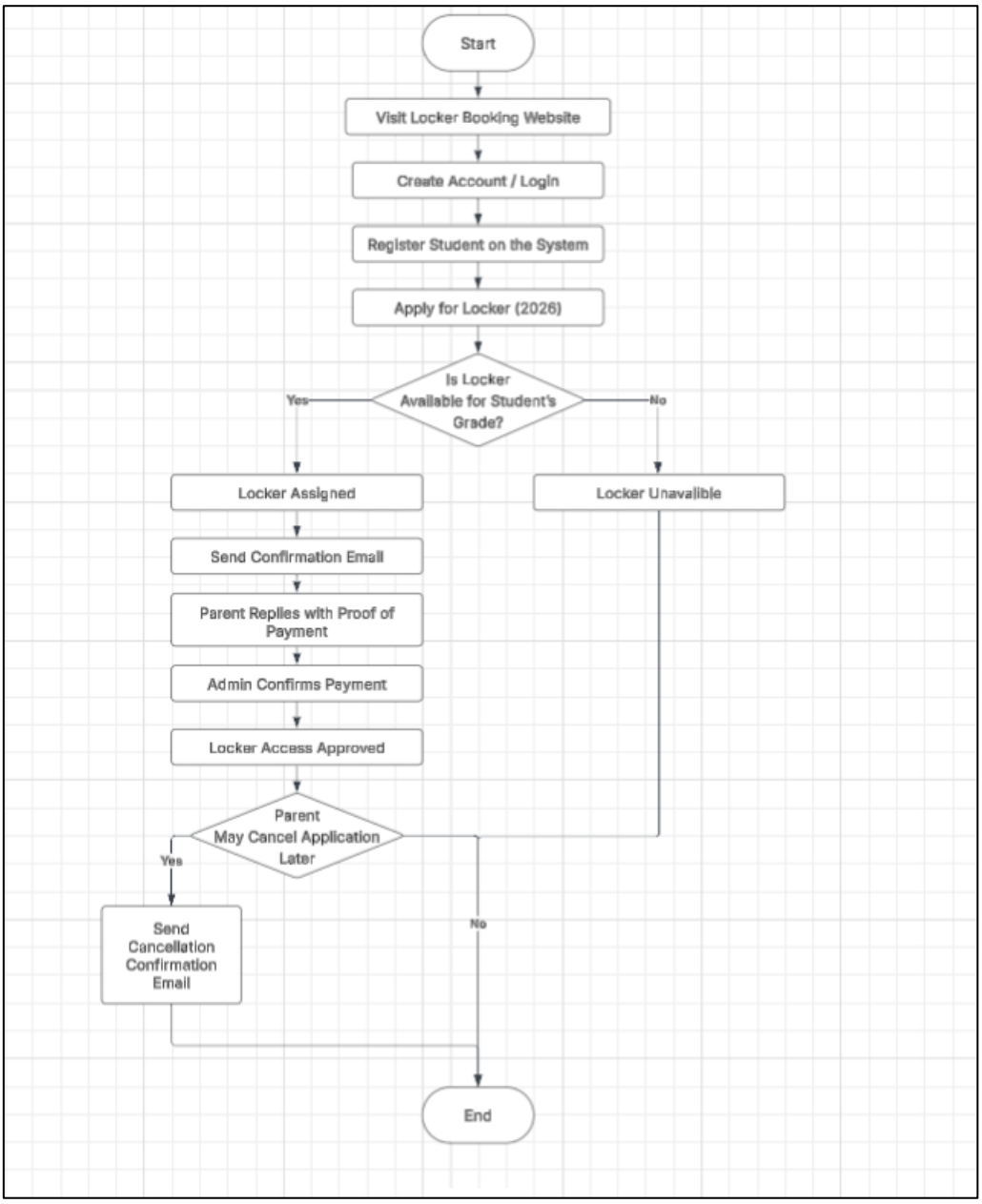
Database Software (1)
Identified the database software that you will use to develop your database.

MySQL and using phpMyAdmin for easy DB management.

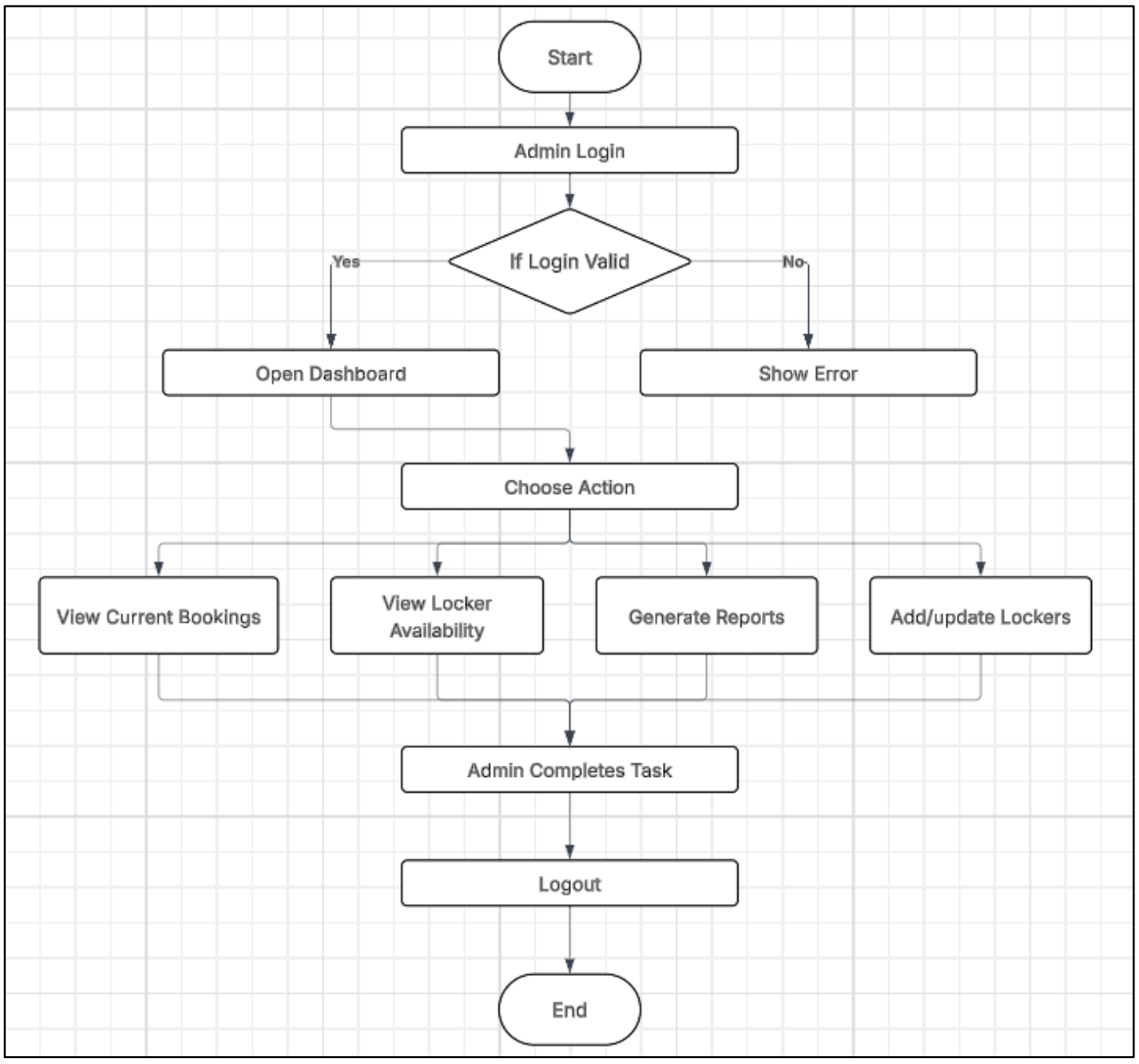
Question 2 [20]

Start by designing your system by completing an Activity Diagram for your system.

Parent Flowchart



Admin Flowchart



Question 3 [20]

Start by designing your database by completing an ERD Diagram for your database.