APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 1196-1215
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Smart Library Occupancy Management System using Deep Learning

Gaajendren A/L Viveganathan[1], Rabatul Aduni Sulaiman[1]*

[1] *Faculty of Computer Science and Information Technology,*
Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Johor, MALAYSIA

*Corresponding Author: rabatul@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.064

Article Info

Received: 12 August 2025
Accepted: 20 November 2025
Available online: 30 November 2025

Keywords

Deep Learning, YOLO, YuNet, MobileFaceNet, Person Counting, Face Detection, Face Recognition, Cosine Similarity, Reservation System

Abstract

Libraries play a crucial role in providing a place for students to gain knowledge. However, in today's world, library spaces are often crowded, and managing real-time occupancy is challenging. Smart Library Occupancy Management System is a web-based system which mainly designed for Perpustakaan Tunku Tun Aminah (PTTA) for manage the library's occupancy. The main objective of the system is to detect individuals entering and exiting the library and to provide accurate real-time occupancy data. Additionally, the system includes a room reservation feature, enabling students to check availability and book study rooms online, thus reducing manual work and improving operational efficiency. The system is developed using the Iterative Model, which allows for continuous refinement based on user feedback and testing. It is implemented as a web-based platform using Python (Flask) for backend development and Laravel (PHP) for the reservation module. The system integrates deep learning techniques, utilizing YOLO (You Only Look Once) for person detection, YuNet for face detection, and MobileFaceNet for face recognition. These models work together to identify and count individuals with high accuracy and speed. After testing, the system able to manage library occupancy using deep learning algorithms and future work could be focused on increasing the facial recognition accuracy under different circumstances.

1. Introduction

The growing emphasis on education in today's world has made libraries an essential space for students to acquire knowledge. Libraries have evolved into one of the primary gathering places for students, turning them into significant hotspots that are often crowded [1]. Access to accurate occupancy data plays a crucial role in ensuring the safety and efficient operation of libraries, allowing staff to determine whether any students remain inside [2][3]. By implementing a real-time monitoring system with dual cameras, occupancy levels within libraries can be accurately estimated using neural network models. In addition, a library room reservation system is a computerized platform that enables patrons to book and reserve meeting rooms, study areas, or other library facilities. This system reduces the need for direct staff involvement in managing reservations and allows students to conveniently secure specific rooms for various academic or group activities.

Smart Library Occupancy Management System is a web-based system which mainly designed for Perpustakaan Tunku Tun Aminah (PTTA) which located in UTHM, Parit Raja. Currently, PTTA does not have any system or hardware to detect the number of people inside the library. Furthermore, the process of booking rooms happens manually. In the current system, students should go to the library and ask the staff to check room

availability. If it is available, then they must give at least five matric cards to the librarian for a room reservation. Then, staff will record the details of the reserved room onto the local system manually.

Due to the lack of a system to detect library occupation in real-time, the staff will not have concurrent data to know how many people are left in library on emergencies such as earthquakes. Which makes major issues regarding safety of people. Meanwhile, without having a system that does not allow students to reserve rooms makes time consuming for the student to come over to the library to check the availability.

To address these issues, this project is planned to use two cameras to accurately detect people by using YOLO algorithm and counts the number of people entering and exiting in the library. YOLO is a deep learning algorithm for accurate people counting [4] is used in conjunction with OpenCV, a library of programming function mainly used for image processing [5]. Besides that, this project also includes a face recognition system using YuNet for face detection and MobileFaceNet, which leverages the FaceNet library for face embedding and recognition [6]. YuNet is a method of face detection and alignment based on deep convolution neural network [7]. The integration of face recognition ensures that the system not only counts the number of people entering and exiting but also identifies and verifies individuals accurately. This functionality enhances the system's ability to monitor occupancy and provides more reliable data by associating each person with a unique face embedding.

This data is collected in real-time and shows it on the staff panel website and by aggregating this data, this system provides valuable insights including identifying peak crowded periods, which will be presented in graphical format [8]. Alongside, this project also provides a web system for students to able to reserve rooms in the library. They able to view real-time availability of study room and other library spaces and enable them to reserve that room for study sessions online. This digital solution aims to enhance librarian experience, improve operational efficiency, and enable data-driven decision-making for business growth [9].

The objectives of this project are to design this system with an object-oriented approach, to develop the system as a web-based application, and to test the system thoroughly through system testing and user acceptance testing to ensure its functionality and usability.

This system targets two users, which are the library staff, who are mainly responsible for the occupancy management process and UTHM students who will reserve the library room for educational purposes. There are a total of eight modules which are listed on Table 1.

Table 1 *System Module*

Module	Description
Registration	Allow students to create an account. They are required to verify their newly registered email by clicking the link sent to their email.
Login	Allow staff and students to log in to the system securely.
Account Management	Allow staff to update and delete all the registered accounts, including updating the role of an account while allowing students to update their own profile.
Real-Time Occupancy monitoring	Allow staff to monitor real-time library occupancy, identify individuals through face detection, and manage undetected exits for accurate tracking.
Room Management	Allow staff to create and manage the library room, whereas students can view available rooms.
Reservation Management	Allow staff to manage room reservations, while students can reserve available rooms and track their reservation status.
Occupancy Report	Allow staff to generate occupancy and reservation reports with graphs, downloadable as PDF or Excel.
System Setting	Allow staff to update website styles, people count time limits and adjust detection coordinates for better accuracy.

The rest of the paper will be organized as follows. Section 2 describes the literature review which explains the current system on PTTA, proposed system and compares three existing library systems. In Section 3, the project methodology is discussed. Section 4 describes system analysis and design. Moreover, section 5 explains the system implementation and overall testing results, while in section 6 conclusion and suggestions to improve the system have been discussed.

2. Literature Review

This section discusses the literature review conducted for the Smart Library occupancy management system with using deep learning and the current processes in use by the Perpustakaan Tunku Tun Aminah (PTTA).

2.1 Current PTTA Occupancy Management System

PTTA does not have any technology to count people, which causes the library staff to struggle to ensure that all students have left after closing, and this issue is further complicated during emergency evacuations. The staff will physically patrol the entire library to ensure no students remain after the library closes. For reserve a room, students must go to the library and ask the staff about the room availability. Then, the staff will check the rack that contains many baskets which each of the baskets contains a key to a room. If any baskets contain the key, the staff will ask the student to provide five matrix cards and then place them in the corresponding basket. After that, the key will be handed to the student, and when they return, the key is placed back in the basket, and the matrix cards are returned to the student.

2.2 Smart Library Occupancy Management System

This proposed system will use two cameras at the entrance and exit points of the library with face detection to capture the face of the individual that is entering, and the system automates the process of counting the occupancy rate of the library. This system will show the number of people inside the library alongside the face of that individual that enters the library staff and easily ensure that no student remains in the library after closing, which gives them full control of the library occupancy. It has a web-based system for online room reservation the student can effectively use the occupancy of the space provided by the library for the study purpose.

2.3 Deep Learning

Deep learning (DL) is a part of artificial intelligence (AI) and is widely used in various fields such as automation, healthcare, and surveillance [12]. It mimics the way the human brain works by using artificial neural networks that consist of input, hidden, and output layers [13]. These layers allow the system to process and learn from large datasets without manual programming. Deep learning makes machines able to perform tasks similarly to humans by working with images, text, and audio files. It achieves this by using multi-layered neural networks where the input layer receives raw data, the hidden layers process the data to identify patterns, and the output layer produces the result. As the layers go deeper, they detect increasingly complex features—from simple edges to shapes or entire objects. This process enables DL systems to perform tasks such as face recognition, object detection, and voice analysis with high accuracy. It requires powerful computational resources, such as GPUs, and large datasets to train and operate deep learning models efficiently [14]. In this project, deep learning plays a key role in automating the process of real-time occupancy monitoring and face recognition. The system uses several deep learning algorithms, including YOLO for person detection, YuNet for face detection, and MobileFaceNet for facial recognition. These models are selected due to their speed, accuracy, and ability to run in real-time environments. The integration of these algorithms allows the system to detect and track people entering and exiting the library, recognize faces with high precision, and provide accurate occupancy data with minimal delay.

2.3.1 You Only Look Once (YOLO)

YOLO (You Only Look Once) is a popular deep learning algorithm based on Convolutional Neural Networks (CNN). It uses a one-stage approach to detect objects by predicting class and location in a single step. The input image is divided into grids, and each grid cell predicts bounding boxes with confidence scores. Boxes with scores above a set threshold are used to find objects in the image [15].

2.3.2 YuNet

YuNet is a face detection model based on convolutional neural networks (CNN). It is developed for fast and efficient face detection in real-time systems. As a single-stage detector, YuNet performs face classification, bounding box prediction, and facial landmark detection in one forward pass. It uses CNN layers to extract features from the input image and accurately detect faces along with key landmarks such as the eyes, nose, and mouth. Because of its lightweight design and accurate face detection, this system uses the YuNet algorithm to detect the faces of people entering and exiting the library in real time.

2.3.3 MobileFaceNet

MobileFaceNet is a popular face recognition algorithm known for its small size and fast speed, making it suitable for use on devices with limited memory like embedded systems [16]. It is a lightweight face recognition model that creates a unique 128-dimensional embedding for each face. This embedding is a set of 128 numerical values (a vector) that represents the key features of a person's face. Instead of storing the full-face image, the system stores this 128-point vector, which makes it faster and more efficient to compare faces. When a new face is detected, the model generates its 128-dimensional vector and compares it with saved vectors to recognize or verify the person.

2.4 A Study on Similar Systems

For the comparison, three universities have been selected to evaluate their library occupancy and room management systems: UTM Library System, Tun Hussein Onn Sunway Library, and Taylor's University Library. These systems are compared with the proposed Smart Library Occupancy Management System using deep learning.

UTM Library System is a web-based platform used for managing room reservations at two libraries under Universiti Teknologi Malaysia. Currently, students need to visit the library to scan a QR code displayed in the facility section, which will navigate them to a Google Form. After selecting the desired library, the form will prompt students to fill in information such as applicant name, event name, status, and booking date. Meanwhile, Tun Hussein Onn Sunway Library offers an online room booking system with a more user-friendly interface. Students start by selecting the booking date and room type, and the system will display a time slot table where red indicates booked and green shows available slots. By clicking on an available slot, users are redirected to a booking form that must be completed within five minutes to secure the reservation. Unlike the other libraries, Taylor's University Library implements face recognition technology at its entrance to monitor student access. A hidden camera captures the student's face, and after a short verification process, the gate opens automatically, allowing the student to enter. While the system focuses on occupancy monitoring, it is not clear whether it supports online room reservation or provides admin control for managing room reservation.

These three systems are compared with the proposed Smart Library Occupancy Management System, which integrates real-time monitoring and room control using deep learning. Table 2 presents the comparison between the existing systems and the proposed system.

Table 2 *System Comparison*

Feature/System	UTM Library	Tun Hussein Onn Sunway Library	Taylor's Library	Proposed System
Login	X	X	√	√
Registration	X	X	√	√
Account Management	X	X	-	√
Real-Time Occupancy monitoring	-	-	√	√
Occupancy Report	-	-	-	√
Room Management	X	√	-	√
Reservation Management	X	√	-	√
Reservation Status	X	X	-	√

The proposed system was compared with three university library systems which are UTM Library System, Tun Hussein Onn Sunway Library, and Taylor's University Library. The proposed system requires user registration and includes account management, unlike the other systems except Taylor's Library. For occupancy monitoring, UTM and Sunway do not allow user access, while Taylor's uses face recognition at the entrance. The proposed system uses a camera to detect students entering and exiting. All systems except UTM support online room management. UTM uses Google Forms, which requires staff to manually update reservations, increasing the chance of human error. The proposed system, like others, allows online reservations without manual staff input. Unlike Sunway, which does not update users on booking status, the proposed system shows booking status and history.

3. Methodology

The chosen methodology to develop this project is Iterative model. The iterative modal was primarily designed to develop a system through repeated cycles (iteration or phases) and in smaller portions at a time [13]. Through this model, full software is not developed at one time, but only the skeleton of whole software is developed and then subsequently requirements are implemented. Each iteration intended to be small and easily manageable and that can be completed within a couple of weeks. Then it will be delivered to the stakeholder and user for review and make improvements based on the feedback. This quickly adapts to requirement changes and will be best if

the functional requirements are unclear in the early phase. There are seven stages on each iteration of an iterative model which are planning, design & analysis, development, testing, review, deploying. Before the seven stages start there is an initial planning phase.

3.1 Iterative Model

The project begins with the initial planning phase, where system requirements are gathered, and project objectives, goals, and stakeholder needs are documented. Meetings were conducted with PTTA staff to discuss workflows, the existing system, problems, and expectations. Interviews were chosen as the primary method for gathering requirements, enabling direct interaction with stakeholders to understand their needs. Questions focused on the current system, existing challenges, and suggestions for the proposed system. A Gantt Chart was created using Microsoft Excel to organize tasks and schedules. Figure 1 shows the overall iterative stage of each iteration.

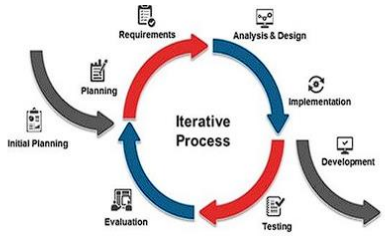


Fig. 1 *Iterative modal stages [11]*

The planning phase involves analyzing detailed requirements for iterative modules or sprints, ensuring the project adapts to stakeholder feedback and prioritizes critical functionalities for each iteration. In the design and analysis phase, the system's structure, functionality, and appearance are planned using UML diagrams, wireframes, and database structures, which are reviewed by stakeholders to validate the system's functionality and ensure clarity. During the development phase, system modules are implemented using programming tools such as Visual Studio Code, PyCharm, and PhpStorm. Code is developed, reviewed, and integrated into a shared repository on a localhost server via Laragon, with a MySQL database connection.

The testing phase focuses on identifying and documenting bugs through system testing and user acceptance testing. Test cases are created based on requirements to verify that the system meets user expectations and functions properly under various conditions. Furthermore, feedback from end users is gathered in the review phase to identify enhancements and prioritize fixes, ensuring the system evolves to meet user needs effectively and remain functional over time. Finaly, the deploy phase involves delivering the system to a secure live server. Activities include setting up the database, migrating files to the server via FTP, and ensuring all modules meet requirements before going live.

3.2 System Development Workflow

Table 3 below shows the system development workflow which describes how each sprint or module progresses through seven iterative phases.

Table 3 *System Development Workflow*

Phase	Process	Output
Initial Planning	• Identify the stakeholders • Initial meeting with stakeholder. • Define scope and objective of the system. • Create Gantt chart by scheduling the task. • Identify the business process and gather user requirements. • Document all details gathered on this process.	• Gantt Chart • Project Proposal
Iteration starts for each module		

Table 3: (*Cont*).

Planning	• Meeting with stakeholders discuss about module. • Identify and gather requirements about the module. • Define business goal and user need on the current iteration module. • Update the document with the current requirement.	• Documented requirement • Functional and non-function requirement
Design & Analysis	• Design and identify database structure and scheme of current module. • Create UML diagram which highlights the current module process workflow. • Design the user interface of the current module • Document all the diagram created and user interface.	• User Interface sketch • Use Case Diagram • Sequence Diagram • Data flow Diagram • Activity Diagram
Develop	• Develop the current module. • Complete all the functions and features of the current module.	• Code folder and file System database
Test	• Create test case. • Test the system by using system testing. • Involves user to perform user acceptance testing. • Document the test result.	• Test reports • Bug logs • Test Case Requirement
Review	• Gather feedback from the user and stakeholder on module that are on live. • Document the feedback and make the changes.	• Feedback result
Iteration End		
Deploy	• Identify a suitable hosting server. • Serve all developed modules.	• Website for all developed modules.

4. System analysis and design

System analysis and design describe the system requirements analysis including the Unified Modelling Language (UML), which represents the requirements analysis of the proposed system through use case diagrams, followed by the class diagram which represents the database conceptual model.

4.1 System Requirement Analysis

System requirements analysis is for identifying the overall needs and objectives of the system. Table 4 below shows the proposed system requirements.

Table 4 *System Requirements of the proposed system*

Modules	Functionalities	User
Registration	• Allow students to register and create an account to access the system. • Allow students to verify their account using the link sent to their registered email.	Student
Login	• Allow students to log in using a registered account. • Allow staff to log in using accounts either given by the developer or created by another staff.	Student, Staff
Account Management	• Allow students to view and edit profiles. • Allow staff to modify user account roles and details.	Student, Staff

Table 4: (*Cont*).

Real-time occupancy management	• Allow staff to view the number of people in real-time. • Allow staff to view the occupancy details of the library. • Allow staff to view the faces of individuals who have entered, are excited, and are currently inside the library.	Staff
Room Management	• Allow staff to view, update, create, and delete the room and its details. • Allow students to view all the rooms created by staff and their details.	Student, Staff
Reservation Management	• Allow staff to view, update the status, create, and delete the reservation and its details. • Allow students to reserve the room and view the status of their current reservation. • Allow students to check past room reservations.	Student, Staff
Occupancy Report	• Allow staff to view the occupancy rate and reservation count in a graphical format, such as a bar graph. • Allow staff to filter data based on specific criteria to meet their requirements. • Allow staff to download the graph in PDF format.	Staff
System Setting	• Allow staff to update the operating time for person counting. • Allow staff to update the coordinate lines for the system to detect individuals entering or exiting the library. • Allow staff to update the student panel website title name, logo, and background color.	Staff

4.2 General System Architecture

This system has two servers built using the Laravel and Flask frameworks, respectively. The Laravel server is responsible for managing all student tasks, requests, and staff except occupancy management. The occupancy management module is fully managed by the Flask server. Figure 2 shows the system architecture of the Smart Library Occupancy Management System.

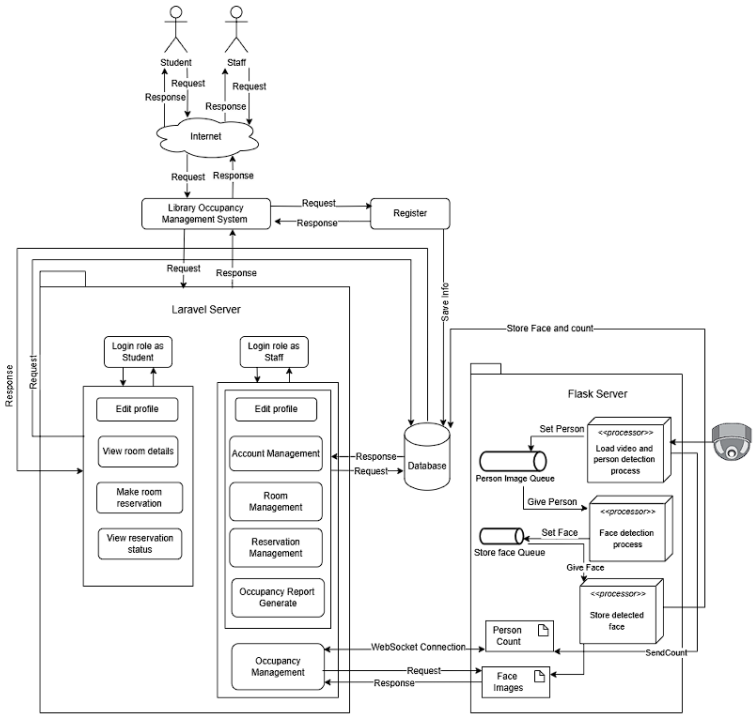


Fig. 2 *System Design Diagram of the Proposed System*

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1196-1215

4.3 Unified Modelling Language (UML)

UML diagram is a standardized visual language used to model and design the structure, behavior, and interactions of a system.

4.3.1 Use Case Diagram

This proposed system has two types of users, which are PTTA library staff and students. For registration and login, two users need to register an account and will be set to default roles which are student. Only other admins can change account roles from student to staff. Staff can manage accounts, library rooms, room reservations and manage occupancy. They can also view reports about how full the library is and change settings like the system name and logo. Meanwhile, students can view available rooms, make reservations online, and track their booking status. They able to print the ticket for their reservation and cancel the reservation under some specific condition. All this user interaction into the system is being shown in use case diagram which in Figure 3.

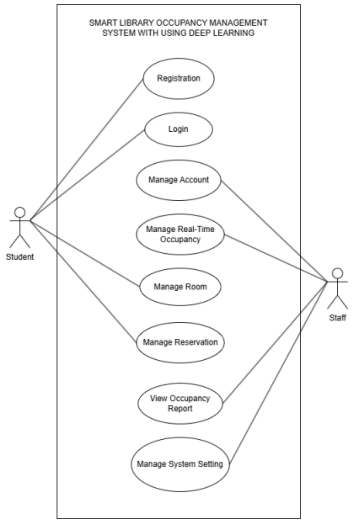


Fig. 3 *Use Case Diagram*

4.3.2 Activity Diagram

This section explains the activities involved in real-time occupancy monitoring, room reservation, and report generation. Staff can monitor and manage the number of people currently in the library. Real-time updates display occupancy rates and details of individuals who have entered, exited, or are still inside. The facial recognition feature helps identify and match unknown individuals with their images. In addition, staff can manage room reservations by viewing, creating, and updating booking statuses. Students are able to make reservations and view both current and past bookings. The system also provides an occupancy report feature that allows staff to view data related to occupancy and reservations in graphical format and generate downloadable PDF reports. Figures 4(a), 4(b), and 5 show the activity diagrams for real-time occupancy management, reservation management, and report generation, respectively.

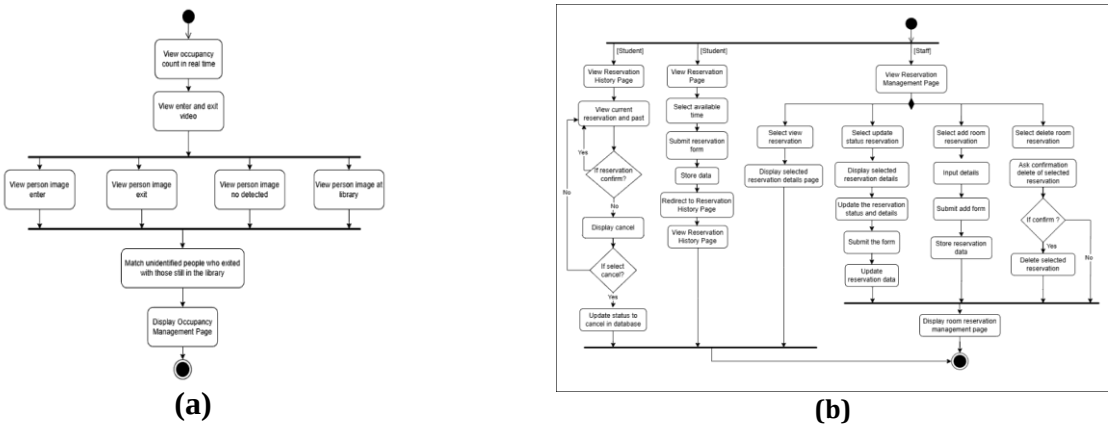


Fig. 4 (a) *Activity Diagram of the real-time occupancy management; (b) Activity Diagram of the reservation management.*

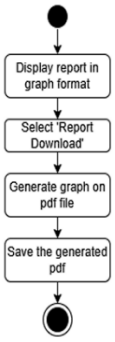


Fig. 5 *Activity diagram of the report module.*

4.4 Class Diagram

The class diagram is a graphical representation of the database conceptual model. It highlights the core entities and their relationship with the proposed system. It involves mainly student and staff classes, which are inherited from the user class. Figure 7 shows the class diagram of the proposed system.

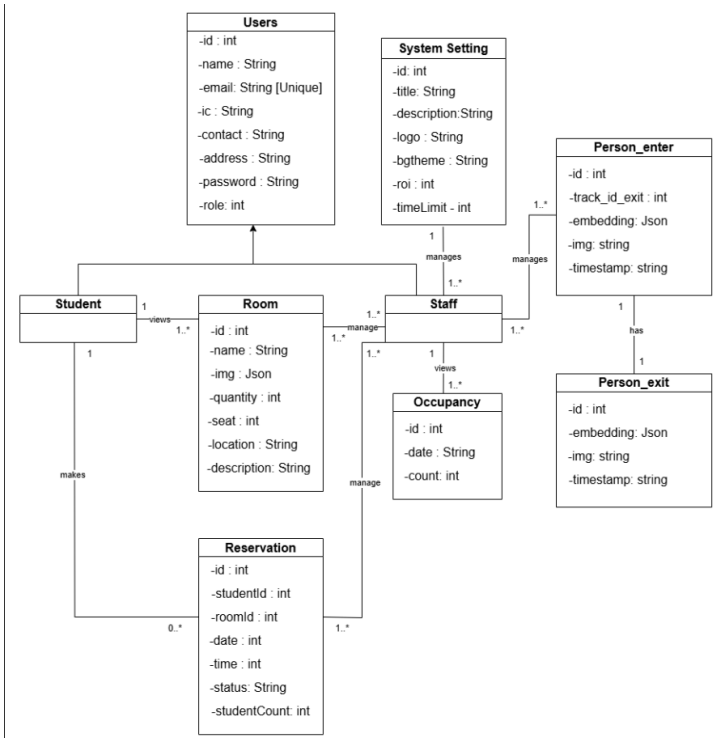


Fig. 7 *The class diagram of the proposed system*

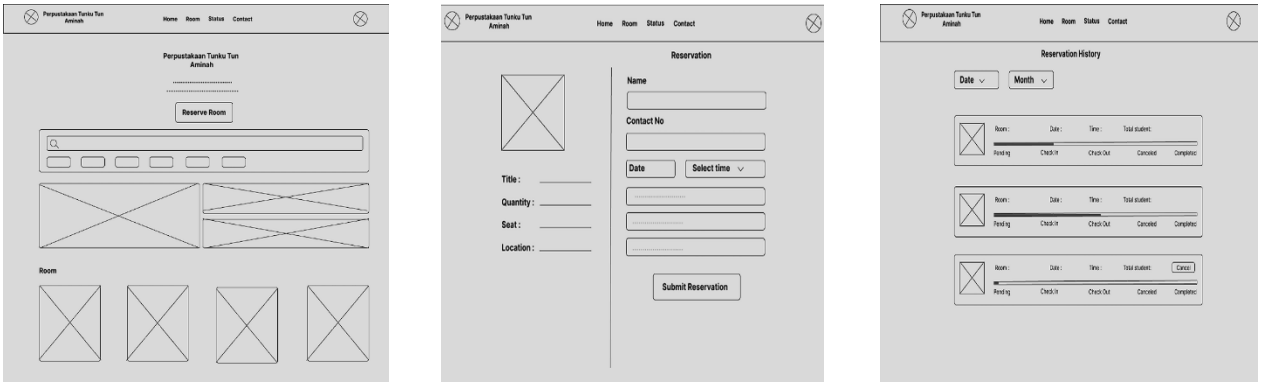
4.5 System Design

System design involves creating a visual blueprint of the system's structure and functionality, including user interfaces, workflows, and data flows. Wireframes of the system's UI are created using Figma to outline the layout, features, and interactions, ensuring clarity and alignment with user requirements.

4.5.1 Student Interface Design

Students are users that are responsible for viewing and reserve the room, include viewing the status of the reservation. Figure 8(a) shows the interface of the student dashboard and Figure 8(b) shows the interface of the student make reservation page. Besides that, Figure 8(c) shows the interface of the student reservation history page.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1196-1215



(a) **(b)** **(c)**

Fig. 8 *(a) Student dashboard page; (b) Student make reservation page; (c) Student reservation history page*

4.5.2 Staff Interface Design

Staff are responsible for managing the occupancy and the reservation. report. Figure 9(a) shows the interface of Real-Time Occupancy Management and Figure 9(b) shows the interface of the staff index reservation page.

(a) **(b)**

Fig. 9 *(a) Interface of the occupancy management page; (b) Interface of the staff index reservation page;*

5. Implementation

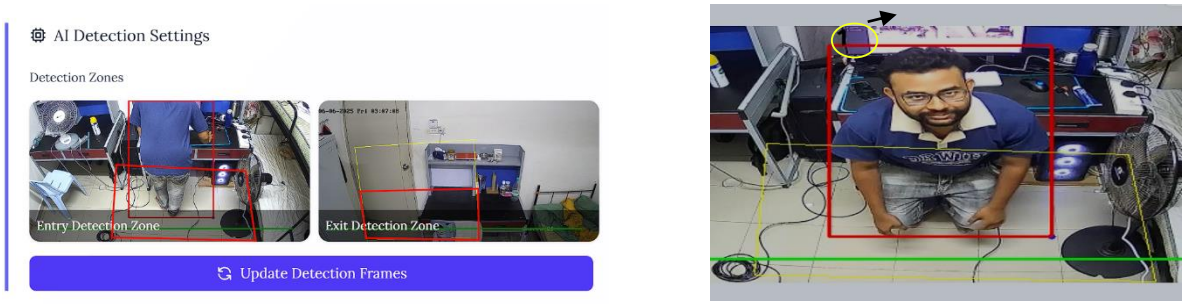
This section explains the user interface and code for the core module of the system which are reservation and real-time occupancy management.

5.1 Real-Time Occupancy Management Module

This module is responsible to show count of occupancy rate in the library at real-time and show face of the person that enter, exit and still at library. This module uses deep learning algorithms which are YOLO, YuNet and MobileFaceNet. All these models are run in the ONNX format to improve performance and ensure smooth integration across different platforms.

5.1.1 Person Counting (YOLO)

This system uses You Only Look Once (YOLO) deep learning algorithm for detect person that entering and exiting the library. That algorithm will give bounding box of the detected person which then will be used to count the number of persons enter and exit the library for get current occupancy count of that library. It takes input images from the camera using the OpenCV videoCapture function. There will be a region of area (ROI) in that image and if a person crosses that specific region, then the YOLO algorithm will send the coordinate of the person. The ROI can be dynamically change on system setting which shown in Figure 10(a). After that, the detected person coordinate will pass to SORT algorithm which are responsible for setting a unique track id for that person across multiple frames. Figure 10(b) shows the bounding box of person detection by YOLO and the tracking number on top of that. The code segments of the person detection are shown in Figure 11.



(a) **(b)**

Fig. 10 *(a) The ROI in red colour line; (b) The red colour line bounding box and tracking number;*

```
results = modal(resized_frame, verbose=False, stream=True, classes=[0])
detections = np.empty((0, 5))

for r in results:
    for box in r.boxes:
        x1, y1, x2, y2 = box.xyxy[0]
        x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
        conf = math.ceil((box.conf[0] * 100)) / 100

        if cv2.pointPolygonTest(np.array(r_io), pt: (x2, y2), measureDist: False) < 0:
            continue

        currentArray = np_array([x1, y1, x2, y2, conf])
        detections = np.vstack((detections, currentArray))

        cv2.circle(resized_frame, center: (x2, y2), radius: 3, color: (255, 0, 0), thickness=-1)
        cv2.rectangle(resized_frame, (x1, y1), (x2, y2), (0, 0, 255), thickness=2)
```

Fig. 11 *Code Segment for person detection*

5.1.2 Face Detection (YuNet)

Face detection is to identify faces on given images for the system by using YuNet deep learning algorithm. It starts with YOLO where it detects the person in the image/frame, then by using OpenCV crop function to isolate and extract the detected person from the rest of the frame. Then, the cropped image of the detected person are send into the YuNet algorithm which responsible to identify the face on the image and send the coordinate of the face on that image/frame. Figure 12 shows the code segments for the face detection.

```
face_data = face_queue.get(block=True)
track_id, face_image, way = face_data

print(f"Processing face data for track_id {track_id} in {way} direction...")

if (way == 'enter' and person_enter[track_id]["checked"]) or (
        way == 'exit' and person_exit[track_id]["checked"]):
    with counter_lock:

        del (active_face_tracks_enter if way == 'enter' else active_face_tracks_exit)[track_id]
        face_queue.task_done()
    continue #check here

h, w = face_image.shape[:2]
detector.setInputSize((w, h))

faces = detector.infer(face_image)

for face in faces:
    confidence = face[-1]
    if not (person_enter[track_id]['face'] if way == 'enter' else person_exit[track_id]['face']):
        confidence = 0.9 #recheck
    if confidence >= 0.9: #make it
        x1, y1, w, h = [int(v) for v in face[:4]]
        cropped_face = face_image[y1:y1 + h, x1:x1 + w]
```

Fig. 12 *Code Segment for face detection by YuNet deep learning algorithm*

5.1.3 Face Recognition (MobileFaceNet)

Face Recognition is to recognize whose face it belongs to. For that on this system are being used MobileFaceNet deep learning algorithm which is responsible to generate a 128 vector point of embedding. This embedding is a numerical representation based on the unique structure of the person's facial landmarks. Figure 13 shows the code segments for the generate the embedding. For comparison, the system uses Cosine Similarity calculation. Cosine similarity measures how similar two embedding are by calculating the cosine of the angle between them, ranging from -1 to 1. The code for this calculation is being shown in Figure 14.

```python
def get_mobileFacenet_embedding(face_img):  1 usage  gaajendren
    try:

        face_img = cv2.cvtColor(face_img, cv2.COLOR_BGR2RGB)
        face_img = cv2.resize(face_img,  dsize: (112, 112))
        face_img = (face_img.astype(np.float32) - 127.5) / 128.0

        face_img = np.transpose(face_img,  axes: (2, 0, 1))
        face_img = np.expand_dims(face_img, axis=0)

        embedding = face_recognizer.run( output_names: None,  input_feed: {'x': face_img})[0][0]
        return embedding
    except Exception as e:
        print(f"Embedding failed: {str(e)}")
        return None
```

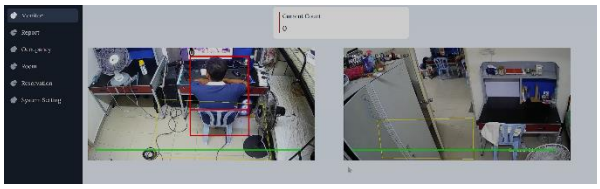
Fig. 13 *Code Segment of the embedding generate*

```python
def batch_compare(current_embedding, threshold=0.4):  1 usage  gaajendren
    with counter_lock:
        similarities = np.dot(enter_embeddings, current_embedding)
        best_idx = np.argmax(similarities)
        best_sim = similarities[best_idx]

        # Threshold check
        if best_sim >= threshold:
            print(f"Best match: Person {best_idx} with similarity {best_sim:.4f}")
            return enter_ids[best_idx], best_sim
        return -1, 0.0
```

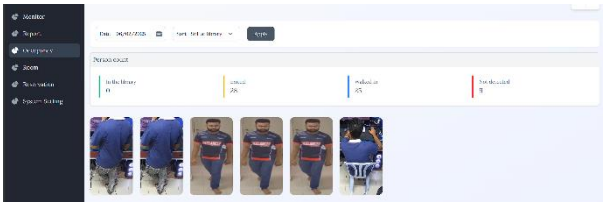
Fig. 14 *Code Segments of embedding comparison using cosine similarity*

5.1.4 Occupancy Management Interface

For the interface, the system uses Socket.IO to enable real-time communication between the flask server and client through a WebSocket connection. This allows instant data changes of occupancy count in real-time, without needing to refresh the page. This system makes staff able to view the occupancy count and video feed from the camera enter and exit. Then, the staff are also able to view the face of the person enter, exit and still at library. Figure 15(a) shows the interface of page viewing the occupancy count and Figure 15(b) shows The interface of faces that enter the library, following Figure 15(c) shows code segments of the Socket.IO connection that responsible to show occupancy count.



(a)



(b)

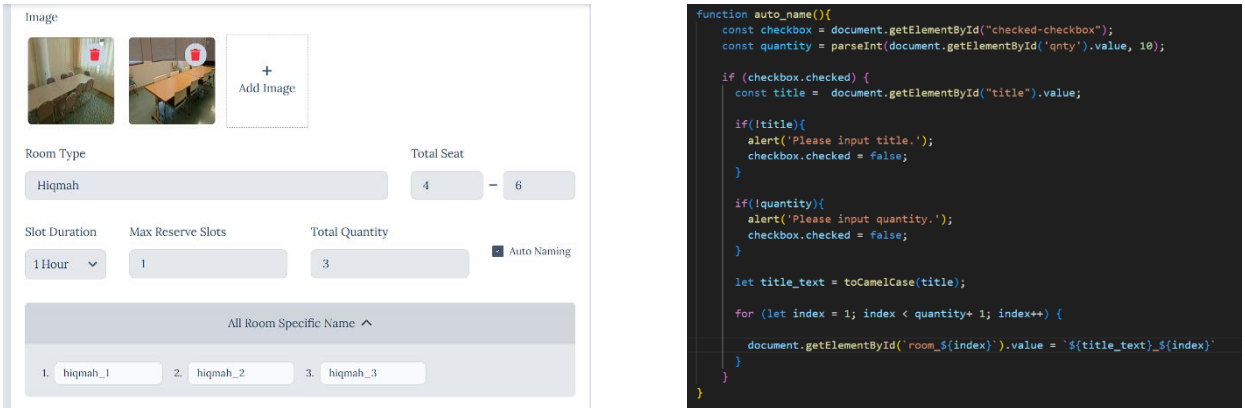
```javascript
const socket = io('http://127.0.0.1:5000', {transports: ['websocket', 'polling', 'flashsocket']});

socket.on('person_count', function(data) {
    console.log(data)
    document.getElementById('person-count').innerText = data.count;
});

function setupStream(way) {
    const container = document.getElementById(`video-${way}`);
    const img = new Image();
    container.appendChild(img);

    let lastUpdate = 0;
    const targetFps = 60;

    socket.on(`video_frame_${way}`, (data) => {
        const now = performance.now();
        if (now - lastUpdate >= 1000/targetFps) {
            img.src = `${data.frame}`;
            lastUpdate = now;
        }
    });
}

setupStream('enter');
setupStream('exit');
socket.emit('request_video_feeds', {ways: ['enter', 'exit']});
```

(c)

Fig. 15 *(a) Interface of occupancy count and video feed; (b) The interface of faces that enter; (c) Code Segments of Socket.IO fetching occupancy count;*

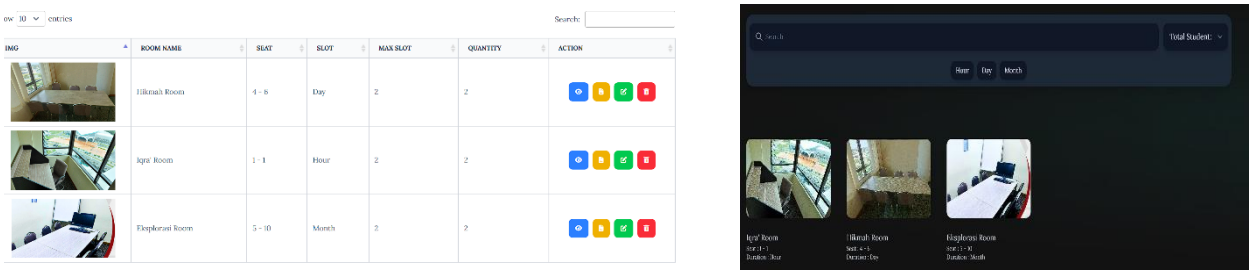
5.2 Room Management Module

On this module, staff can create different room types by filling out a form. They can set how many rooms are available for each type and choose the maximum number of slots a student can book. Staff can also pick the slot type whether it's by hour, day, or month. When they set the room quantity, the system will automatically generate names for the rooms based on that number, but staff can still change the names manually if needed. Figure 16(a) shows the interface of creating room types and its room and Figure 16(b) shows the automatic room name assign code segments. After filling up the room type, the system will save it in database and will show in index pages which are in Figure 17(a). The student also can view the room type that has been created on the dashboard and by clicking it will navigate to the room details page where student book that room. Figure 17(b) shows the interface of room type list for the student to view.



(a) **(b)**

Fig. 16 *(a) Interface of the add room type and room; (b) Code Segment of room auto naming;*

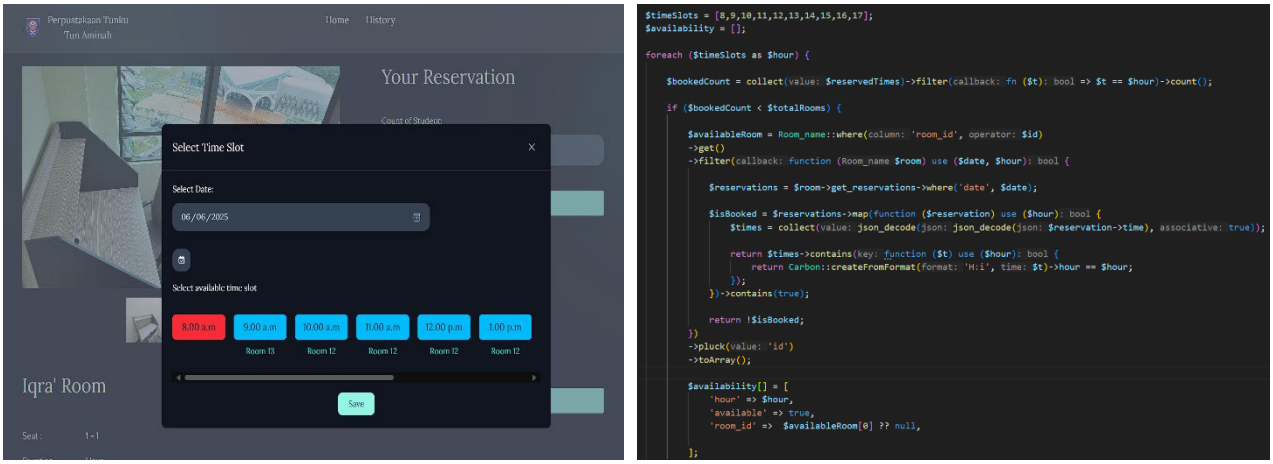


(a) **(b)**

Fig. 17 *(a) Interface of Room Index page; (b) Interface of room list page;*

5.3 Reservation Management Module

In this module there are a total of two users involved, which are students and staff. Firstly, students are responsible for reserving a library room depending on the slot duration, whether it is hour, day or month based. They need to choose date, room type and the system will provide an available slot and also will automatically assign the room based on availability. Figure 18(a) will show the interface design of the selecting available slot and Figure 18(b) shows the code segments of automatically assign available room on each time slot.




```php
$timeSlots = [8,9,10,11,12,13,14,15,16,17];
$availability = [];

foreach ($timeSlots as $hour) {

    $bookedCount = collect(value: $reservedTimes)->filter(callback: fn ($t): bool => $t == $hour)->count();

    if ($bookedCount < $totalRooms) {

        $availableRoom = Room_name::where(column: 'room_id', operator: $id)
        ->get()
        ->filter(callback: function (Room_name $room) use ($date, $hour): bool {

            $reservations = $room->get_reservations->where('date', $date);

            $isBooked = $reservations->map(function ($reservation) use ($hour): bool {
                $times = collect(value: json_decode(json: json_decode(json: $reservation->time), associative: true));

                return $times->contains(key: function ($t) use ($hour): bool {
                    return Carbon::createFromFormat(format: 'H:i', time: $t)->hour == $hour;
                });
            })->contains(true);

            return !$isBooked;
        })
        ->pluck(value: 'id')
        ->toArray();

        $availability[] = [
            'hour' => $hour,
            'available' => true,
            'room_id' =>  $availableRoom[0] ?? null,

        ];
```

(a) **(b)**

Fig. 18 (a) Interface of the selecting available slot; (b) Code Segments for getting available time slots and auto assigns an available room to each time slot.

On the other hand, staff can manage the reservation by updating the status of the reservation. They view all the reservations in table form and are able to decide the status of the reservation. Figure 19 shows the interface of reservation index page where all reservation listed and following figure 30 shows the code segments for get all reservation and update status.

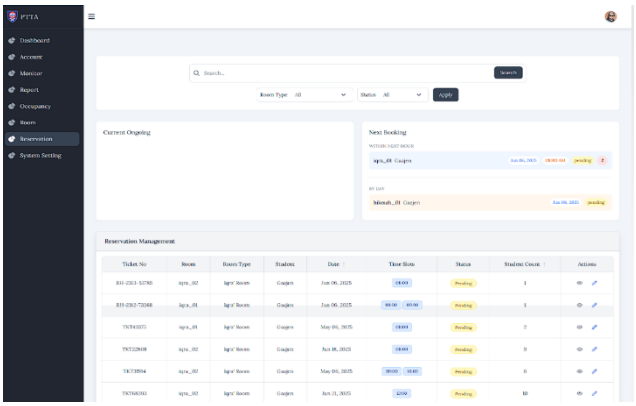


Fig. 19 *Interface of reservation index page*

```php
public function index(): View
{
    $reservations = Reservation::orderBy(column: 'created_at', direction: 'desc')->paginate(perPage: 15);
    $room_types = Room::all();

    return view(view: 'staff.reservation_management.index')->with(key: 'reservations', value: $reservations)
    ->with(key: 'room_types', value: $room_types);
}

public function update(Request $request, $id): RedirectResponse
{
    $input = $request->all();

    $rules =[ 'status' => 'required'];
    $messages = ['status.required' => 'Status is required.',];

    $request->validate(rules: $rules, params: $messages);

    try{

      $reservation =  Reservation::find(id: $id);
      $reservation->update($input);

      return redirect()->route(route: 'staff.reservation')
      ->with(key: 'success', value: 'Status has successfully updated!!!');

    }catch(\Exception $e){
        return redirect()->route(route: 'staff.reservation')->with(key: 'error', value: 'Try again later');
    }
}
```

Fig. 20 *Code Segment of getting all reservation and update reservation status*

Moreover, the system also sends email notifications about the reservation made by the student. The student will receive the ticket in their mail, which consists of the reservation details the interface of the email shown in Figure 21, followed by Figure 22 which shows the code segments for email notification.

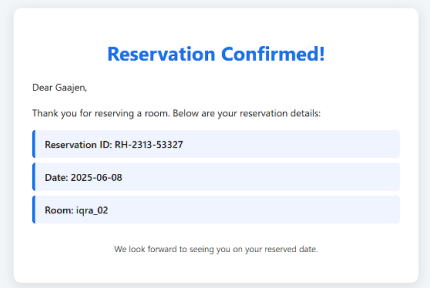
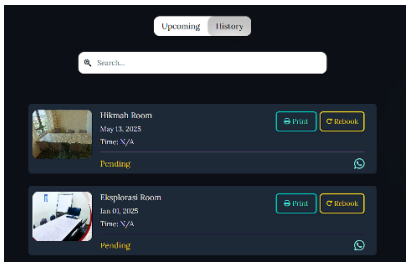


Fig. 21 *Reservation Ticket send in email*

```
0 references | 0 overrides
public function handle(ReservationNotification $event): void
{
    Log::info(message: 'SendReservationTicket Listener Triggered for Reservation ID: '
    . $event->reservation->id);
    Mail::to(users: $event->reservation->get_student->email)
    ->send(mailable: new ReservationTicket(reservation: $event->reservation));
}
```

Fig. 22 *Code Segments of email notification sending*

Besides that, the system also allows students to view upcoming and past reservations made by them. The reservation can be cancelled if status pending can student also able to click WhatsApp icon for ask any further enquiries to the library staff. Figure 23(a) shows the interface of the reservation history page and Figure 23(b) shows the code segments of getting the reservation.



(a)

```
$type = $request->query(key: 'type');
$studentId = auth()->user()->id;
$today = Carbon::today();

$reservations = Reservation::where(column: 'studentId', operator: $studentId)
->when(value: $type === 'upcoming', callback: function (Builder $query) use ($today): Builder {
    return $query->where(column: 'date', operator: '>=', value: $today);
}, default: function (Builder $query) use ($today): Builder {
    return $query->where(column: 'date', operator: '<', value: $today);
})
->with(relations: 'get_roomType')->with(relations: 'get_room')
->get();


return response()->json(data: $reservations);
```

(b)

Fig. 23 *(a) Reservation History Page; (b) Code segment of getting reservation history;*

5.4 Occupancy Report Module

This system also gives a graphical report of the occupancy data. It consists of past 6 hours live occupancy rate, peak hour time period, average occupancy counts by hour, day and month based. It involves different types of charts which are line, bar and radar chart. Figure 24 shows the interface of the graph and Figure 25 shows the code segments for generating the chart.

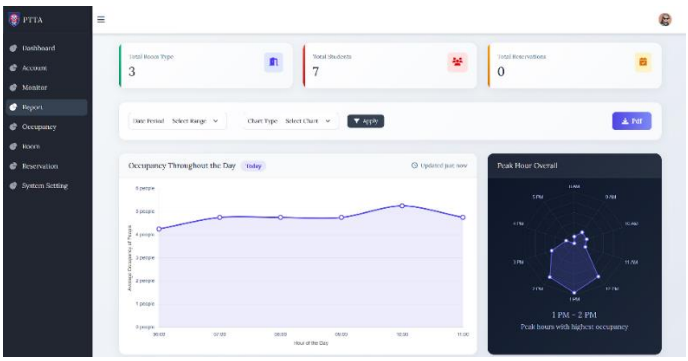


Fig. 24 *Interface of the occupancy report chart*

```
function peakHourOverallChart(){
        axios.get('/peak_occupancy').then((res)=>{

            const formattedTimes = [];

            res.data.hours.forEach(hour => {
                formattedTimes.push(convertTo12Hour(hour, 'no_add'));

            });

            document.getElementById('peak_hour').textContent = convertTo12Hour(res.data.peak_hours.toStri

            const peakHourOverallCtx = document.getElementById('peakHourOverallChart').getContext('2d');
            new Chart(peakHourOverallCtx, {
                type: 'radar',
                data: {
                    labels: formattedTimes,
                    datasets: [{
                        label: 'Average Occupancy',
                        data: res.data.averages,
                        backgroundColor: 'rgba(99, 102, 241, 0.2)',
                        borderColor: '#6366f1',
                        borderWidth: 2,
                        pointBackgroundColor: 'white',
                        pointBorderWidth: 2,
                        pointRadius: 4
                    }]
                },
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                    scales: {
                        r: {
                            angleLines: {
                                color: 'rgba(255, 255, 255, 0.1)'
                            },
                            grid: {
                                color: 'rgba(255, 255, 255, 0.1)'
                            },
                            pointLabels: {
                                color: '#c7d2fe',
                                font: {
                                    size: 11
                                }
                            },
                            ticks: {
                                display: false,
                                stepSize: 20
                            },

                        }
                    },
```

Fig. 25 *Code Segment of the peak hour chart*

5.5 System Setting Module

On this module, this system allows staff to change the user interface of the student dashboard's appearance. The logo of the system, title, description and banner can be changed easily by staff. Figure 26 shows the interface of the system setting page and its code segment present on Figure 27.

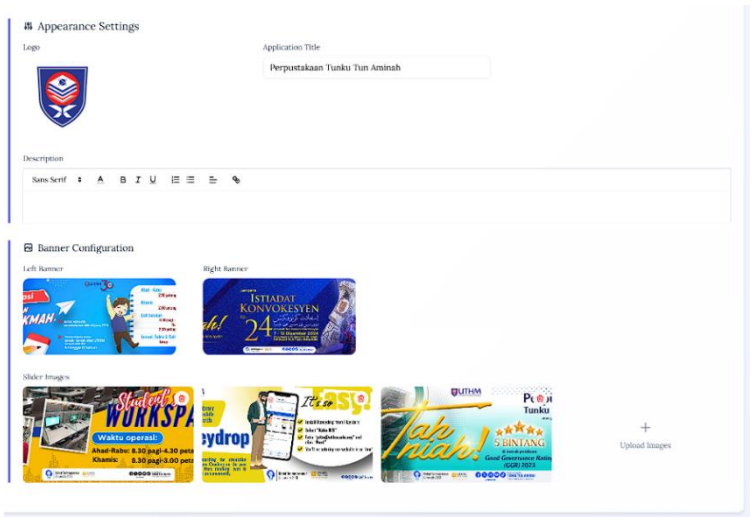


Fig. 26 *System setting interface*

```
$setting->title = $request->title;
$setting->description = $request->description;
$setting->start_time = $request->start_time;
$setting->end_time = $request->end_time;
$setting->is_manual = $request->is_manual;

if ($request->hasFile(key: 'logo')) {
    $image = $request->file(key: 'logo');
    $imageName = time() . '.' . $image->getClientOriginalExtension();
    $image->move(directory: public_path(path: 'img'), name: $imageName);
    $setting->logo = 'img/' . $imageName;
}

try{
    $filenames = [];
    $banner = json_decode(json: $setting->banner, associative: true);
    $filenames = $banner['slider_image'];

    if ($request->hasFile(key: 'img') ) {

        foreach ($request->file(key: 'img') as $image) {
            $filename = $image->getClientOriginalName();
            $uniqueFilename = time() . '_' . $filename;
            $image->move(directory: public_path(path: 'img'), name: $uniqueFilename);
            $filenames[] = $uniqueFilename;
        }
    }

    if($request->hasFile(key: 'banner_3')){
        $sideBannerName_3 = $request->file(key: 'banner_3');
        $BannerName_3 = time() . '.' . $sideBannerName_3->getClientOriginalExtension();
        $sideBannerName_3->move(directory: public_path(path: 'img'), name: $BannerName_3);
    } else {
        $BannerName_3 = $banner['banner_3'];
    }
```

Fig. 27 *System setting update code segment*

5.6 Testing

This system has been tested using two different test types which are system testing and user acceptance testing. System testing was done to ensure that all modules and features function correctly as a complete system. User acceptance testing (UAT) was then carried out to get the user feedback which was used to improve the user interface.

5.6.1 System Testing

System testing is done to make sure the system works as expected and meets both functional and non-functional needs. The testing is considered complete when all modules work correctly according to the requirements. Table 5 shows the overall result of the test cases conducted.

Table 5 Overall Result of test cases

Test Case Modules	Test Case Id	Total Test Case	Total Success	Total Failed
Registration	TC_100	8	8	0
Login	TC_200	10	10	0
Account Management	TC_300	10	10	0
Real-Time Occupancy Management	TC_400	4	4	0
Room Management	TC_500	12	12	0
Reservation Management	TC_600	20	20	0
Occupancy Report	TC_700	7	7	0
System Setting	TC_800	5	5	0

There is total sixty-six test cases have been conducted, and the result turn out all test cases has been success without have any failed.

5.6.2 User Acceptance Testing

User Acceptance Testing (UAT) was carried out to evaluate the performance, usability, and overall satisfaction of the developed Smart Library Occupancy Management System. The UAT was designed and created by the author using Google Forms and was distributed to selected students who had hands-on experience using the system. The test aimed to collect user feedback regarding the system's core functionalities, including real-time occupancy monitoring, face recognition accuracy, and room reservation features. The form was divided into three sections which are user functionality, system interface and suggestions for improvement. Each section included multiple rating-based and open-ended questions to capture both quantitative and qualitative feedback. Participants were instructed to use the system beforehand, then respond based on their interaction and experience. Figure 28 shows the result of the UAT section for user functionality.

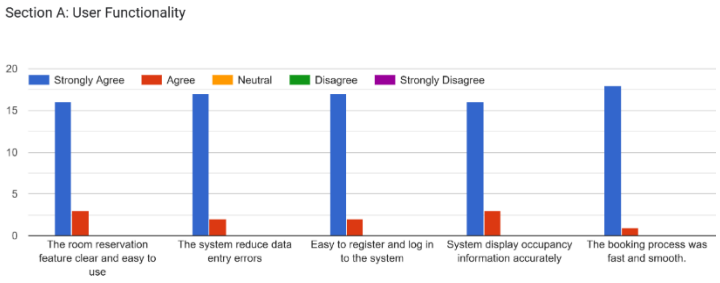


Fig. 28 *User functionalities result in bar chart*

The result from the user functionalities shows strong positive feedback from the user and most users strongly agree that the room reservation is clear and easy to use, the system reduce data entry errors, it easy to register and log into the system, the system display occupancy information accurately and the booking process was fast and smooth. Besides that, for section system interface design also many users strongly agree that the system interface is user-friendly and easy to navigate, the user interface of the system attractive, the layout of the system is well-organized, and the text size easily can be read. Figure 29 shows the result of the section System Interface Design in chart format.

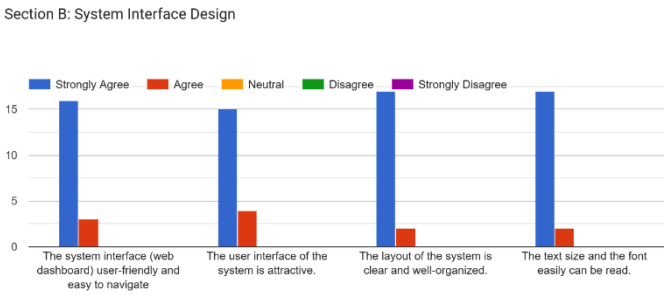


Fig. 29 *System Interface Design result in chart*

Meanwhile, in section four recommendation there are users that give suggestion to improve the system which are to support for Malay and English language options, to have a simple help page or chatbot that explains how to book a room and allow students to give feedback on room usage, cleanliness, or system issues and more. The comments for suggestion have shown in Figure 30.

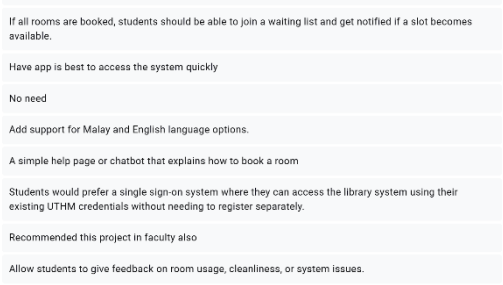


Fig. 30 *Comments list for system improvement*

6. Conclusion

The Smart Library Occupancy Management System using Deep Learning is designed to help staff manage library occupancy and handle room reservations more easily. It also allows students to book rooms and check their reservation status without any extra effort.

The system offers several advantages. It provides secure authentication through email verification and password hashing. Students can easily reserve library rooms and track their reservation status. Staff can efficiently manage all reservations and use the system as a storage platform for future reference. The system also helps staff monitor the current occupancy count, which is especially useful during emergencies. Additionally, it enables staff to understand peak occupancy times, allowing them to manage library space more effectively. The system is also future proof, allowing staff to update the website's logo and title through a simple settings form.

Despite its benefits, the system has some drawbacks. It requires a browser and a stable internet connection to access. Running the deep learning algorithms smoothly also demands a high-performance server. The system may run slowly on servers that only use CPUs or AMD graphics cards, as they do not support CUDA technology. Furthermore, the system depends on high-quality cameras, which can be expensive.

Future improvements can be implemented by developing a mobile application version of the system. This would allow students and staff to access the system more easily and receive real-time notifications on their devices. Staff also easily can view the current occupancy rate without having open any browser. In addition, using a powerful server or setting up a local high-performance server can help the system run more smoothly. The server needs an NVIDIA graphics card that supports CUDA technology, which later needs some turn over the code to support the CUDA acceleration which able to speed up the run time of the code execution. Lastly, this system needs a very quality camera cause the face detection algorithm heavily depends on the quality of the video. The recommendation is to use a 4k resolution camera that has a good RTSP connection. A cheaper camera can reduce accuracy of face and person detection, and the camera connection may not be stable.

Acknowledgement

The author extends sincere gratitude to Dr. Rabatul Aduni Binti Sulaiman for her guidance. Appreciation is also given to author parents, Viveganthan A/L Sundarajoo and Vikneswary A/P N.Rajoo, for their unwavering support and encouragement. Special thanks to friends for their assistance in completing this project successfully.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] Ullah, A., & Usman, M. (2023). Role of libraries in ensuring quality education at higher education institutions: A perspective of Pakistan. *International Journal of Social Sciences*, 2(4), 0-152. https://doi.org/10.1022/ijss.v2i4.57

[2] Halsted, D., & Clifton, S. (2014). *Library as safe haven: Disaster planning, response, and recovery - A how-to-do-it manual for librarians* (Vol. 8). American Library Association. https://alastore.ala.org/sites/default/files/LibraryAsSafeHavenPDF.pdf

[3] Khoche, S., Chandrasekhar, K. V., Sasirekha, G. V. K., Bapat, J., & Das, D. (2021). Occupancy detection for emergency management of smart buildings based on indoor localization: A feasibility study. *SN Computer Science*, 2(6). https://doi.org/10.1007/s42979-021-00812-4

[4] Ren, P., Fang, W., & Djahel, S. (2017). A novel YOLO-based real-time people counting approach. https://doi.org/10.1109/ISC2.2017.8090864

[5] Mahamkali, N., & Ayyasamy, V. (2015). *OpenCV for computer vision applications*.

[6] Lu, Z., Zhou, C., & Wang, H. (2024). A face detection and recognition method built on the improved MobileFaceNet. *International Journal of Sensor Networks*, *45*(3), 166-176.Ku, H., & Dong, W. (2020). Face recognition based on MTCNN and convolutional neural network. *Frontiers in Signal Processing*, 4(1), 37–42.

[7] Wu, W., Peng, H., & Yu, S. (2023). Yunet: A tiny millisecond-level face detector. *Machine Intelligence Research*, *20*(5), 656-665.

[8] Perera, D., Hilley, W., & Nykolaiszyn, J. M. (2024). Enhancing study room reservations at the Oklahoma State University Libraries: A case study on user interface development. *Journal of Access Services*, 21(1). https://doi.org/10.1080/15367967.2024.2316709

[9] Yaseen, M., Ibrahim, N., & Mustapha, A. (2019). Requirements prioritization and using the iteration model for successful implementation of requirements. *International Journal of Advanced Computer Science and Applications*, 10(1).
[10] Miraz, D., & Ali, M. (2020). Blockchain Enabled Smart Contract Based Applications: Deficiencies with the Software Development Life Cycle Models. *Baltica*, *33*, 101–116.
[11] Almatrooshi, Fatima & Akour, Iman & Alhammadi, Sumayya & Shaalan, Khaled & Salloum, Said. (2020). A Recommendation System for Diabetes Detection and Treatment. 10.1109/CCCI49893.2020.9256676.
[12] Srivastava, S., Divekar, A. V., Anilkumar, C., Naik, I., Kulkarni, V., & Pattabiraman, V. (2021). Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, *8*(1), 66. https://doi.org/10.1186/s40537-021-00434-w
[13] Hassaballah, M., & Awad, A. I. (2020). Deep learning in computer vision: principles and applications. CRC Press.
[14] Abdel-Jaber, H., Devassy, D., Al Salam, A., Hidaytallah, L., & EL-Amir, M. (2022). A Review of Deep Learning Algorithms and Their Applications in Healthcare. *Algorithms*, *15*(2). https://doi.org/10.3390/a15020071
[15] Sultana, F., Sufian, A., & Dutta, P. (2019). *A Review of Object Detection Models based on Convolutional Neural Network*. https://doi.org/10.1007/978-981-15-4288-6_1
[16] Jingwei Li, Yipei Ding, Zhiyu Shao, Wei Jiang, *Face recognition method based on fusion of improved MobileFaceNet and adaptive Gamma algorithm*, *Journal of the Franklin Institute*, *Vol. 361, No. 17*, 2024, Article 107306, ISSN 0016-0032, https://doi.org/10.1016/j.jfranklin.2024.107306