ASHESI

ASHESI UNIVERSITY

COMPUTERIZED HOSTEL MANAGEMENT SYSTEM FOR HOSANNA HOSTELS LIMITED

APPLIED PROJECT

B.Sc. Management Information Systems

Benjamin Nai Ako

2021

ASHESI UNIVERSITY

COMPUTERIZED HOSTEL MANAGEMENT SYSTEM FOR HOSANNA HOSTELS LIMITED

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Management Information Systems.

Benjamin Nai Ako

2021

DECLARATION

I hereby declare that this [capstone type] is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

Candidate's Name:

Benjamin Nai Ako

Date:

14th May 2021

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied projects laid down by Ashesi University.

Supervisor's Signature:

Supervisor's Name:

Dr Stephane Nwolley

Date:

14th May 2021

Acknowledgements

I would like to thank God first and foremost for giving me the knowledge and strength to complete this capstone. I would also like to express gratitude to my supervisor Dr Stephan Nwolley for his guidance and support throughout this project. I would also like to thank Papa Darfoor, Kwaku Kwayisi Boohene, Allotei Pappoe and my housemates for their support during the coding process. My final gratitude is to my family for their continuous support and encouragement through my years in Ashesi.

Abstract

An increase in the student population of Ashesi University has led to a greater demand for off-campus accommodation facilities. The limited number of off-campus hostels is unable to adequately meet the current demand, resulting in an annual struggle for accommodation space. Hostel managers are still using old techniques of managing their facilities, and the inefficiency of these techniques negatively impacts the efficient management of the facilities. This project aims to develop a custom computerized Hostel Management System for Hosanna Hostels Limited to streamline processes, including registration & booking and management/administrative processes/tasks. The system will be developed using Php programming language to handle the backend processes, MySQL database for the data storage processes and HTML, JavaScript & CSS for the front-end. The computerized system will eliminate the issues of the old techniques; it will be graphic-user interface oriented, secure, efficient and reliable.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

Since private hostels started operating just outside the Ashesi University campus in 2012, booking and managing these hostels have not gone through much improvement. They still rely on conventional manual methods of managing their facilities and providing services to students. One of these hostels is Hosanna Hostels Limited. Hosanna Hostel has two separate hostel buildings with over 50 rooms and well over 150 student residents. The hostel is currently managed by just one personnel. The entire process of reservation and carrying out some administrative functions are done manually. These processes are cumbersome and time-wasting.

Current Process of Booking

Given the high demand for these hostels, students must book a year in advance. To date, students must go to the hostel premises to inform the hostel managers of their intention to join the hostel as residents. The process usually involves students giving their names, contact numbers and room type choices to the hostel manager, who uses a notebook to keep these records. On the day of reopening, students are required to report to the hostel with a receipt as proof of payment of hostel fees, cross-check if they were allotted the room they asked for and pay cash to get a key to the room.

Challenges of the Conventional Manual Methods

I. **Overbooking of rooms**

Even with booking in advance, some students are still unable to get accommodation because the hostels get overbooked. This overbooking issue primarily occurs because the hostel manager uses a notebook to keep the booking records. Students keep booking for rooms, and the manager makes the entry into the notebook without proper cross-checking to consider the number of rooms available in the hostel. This process is vulnerable to human error, and data recovery is almost impossible. Keeping records of students who are already in the hostel is also difficult.

II. **Improper reporting system**

During their stay, residents must search for the hostel manager to report problems they face and wait for long periods before they get solved because the manager might forget. The manager also goes from room to room to inform students of any announcements and updates concerning the hostel.

1.2 Proposed System

These manual methods of managing the hostel and booking accommodation are inefficient and challenged. This project seeks to solve the challenges faced with the old manual procedures and practices of managing hostels, specifically, Hosanna Hostels Limited, and challenges involved in booking the hostel. The improvement of services for all stakeholders (management and students) is a strong focus of this project. *"Management systems, as the name suggests, are primarily used to manage, oversee or monitor the processes and procedures of a particular task or sequence of activities in an organization*" [13]. The computerized hostel management system will streamline the registration & booking process and automate administrative processes. The system will have a database containing details of all students registered on it, allowing managers to view a profile of each resident in the hostel. Students will access their bills on the system, receive announcements from the manager and report problems in the hostel through the system. Its main sections will include reservation management, inventory management, incident management and other administrative functions. It will also have an integrated payment system to enable students to make online payments and help management keep track of students who have paid and revenue from fees payment. The system will be developed with security, accessibility, availability and reliability in mind to ensure the hostel's private information and data are kept secure, and the system is always available when needed.

1.3 Related Works

a. User-Driven Facility Management System for Universities [7]

This paper explores the challenges of the manual method of facility management and proposes a new design. This design will be based on user requirements. The authors concentrate on using a customer-driven approach that will help them develop an automated system that meets users' needs.

b. Digitized Hostel Room Allotment [12]

This paper addresses issues faced by students in universities when it is time for room allocation. The authors speak on the drawbacks of the conventional process. They go on to propose implementing a digital platform to tackle the current issues. They also discuss improvements in other aspects of university administration that can benefit from digitization.

Chapter 2: Requirements

This chapter focuses on the requirements gathering and analysis needed to begin the development of the proposed solution. The methods of gathering the requirements and analyzing these requirements to produce a requirement statement are highlighted in this chapter.

2.1 Requirements Gathering (Hosanna Hostels Ltd.)

The process of requirement gathering and analysis was done to understand better Hosanna Hostel's operations and how the new system can improve the existing processes and operations. A short interview was carried out. The interview was the method used to collect data from the hostel management, one of the stakeholders. The current process involves students meeting with the hostel's manager to inform him of their interest in living in the hostel in the next academic year. The students write their names, class and contact information (number and email address) in a book provided by the hostel manager. They also indicate what type of room they want and who they want as roommates (roommates usually see the manager at the same time to write their names down). With their details collected, the hostel manager contacts the students through WhatsApp during the vacation to inform them of the prices of rooms for the next academic year. The students have to constantly remind the hostel manager of their intention to live in the hostel the next academic year. It is a first-come, first-serve process, so the students who pay early are given priority. The students must pay for the room and send a copy of the receipt to the hostel manager. On the move-in day, students must present their physical receipts to confirm they can get accommodation. Once this is confirmed, they are given a key and directed to their room. A few students indicate the exact room they want while the others get placed in any room based on the type they paid for (2 per room, 3 per room or 1 per room). A word document is later prepared by the hostel manager containing all the students in the hostel.

Their contact information and how much they have paid are also recorded. If there are any announcements during the semester, the hostel manager prints them out and pastes them on the two notice boards in the hostels.

2.2 Analysis (Hosanna Hostels Ltd.)

Deductions were made from the information gathered through the interview. As a business, the hostel must fill all rooms for the academic year. Due to this, the hostel manager accepts everyone who comes to express an interest. Knowing students can change their minds or even have their decisions changed by their parents, he can't rely on all who expressed interest actually to pay and move in. Students are also made aware of the first-come, first-serve policy, so it depends on them to secure their accommodation in the hostel. Due to the absence of a proper system, the manager has to cross-check payments manually to ensure the hostel is not losing any money or being conned by a student. It would be beneficial for the hostel if they get a proper mode of communication with the residents to improve their experience living in the hostel.

2.3 Requirements Gathering (Ashesi University Students)

Students are also stakeholders in this project, and as such, their views and needs must be included. The new system won't be used by just the hostel administration; students will also be users. A questionnaire containing 11 questions was distributed to students as a method of data collection. The purpose was to find out their experience booking the hostel manually and their experience with other bookings/ reservation systems.

2.4 Analysis (Ashesi University Students)

The questionnaire received 54 responses. The average rate for students' experience with the hostel (concerning booking, reporting problems and general living) was 6.56. The most common response to the challenges they faced from the booking period to the move-in day was constantly reminding the hostel manager of their interest. Others also complained of getting assigned rooms they did not ask for, the hostel manager forgetting demands and not getting enough information about the hostel's booking policies. 100% of the respondents said they had to see the hostel manager to book their rooms. For their ideal booking process, 90% of respondents indicated that an online system would be preferable. In addition to the process being online, they would like to see the rooms available, be informed if the hostel has reached its booking capacity, pay online and receive a confirmation to know their spot has been secured. They indicated the process should also be easy and fast to complete. What stood out for respondents in other booking systems they have used was the ability to do everything on one page, a simple page layout, quick confirmation, knowing whether a room was available or not and the ease of the entire process.

2.5 Functional and Non-Functional Requirements

2.5.1 Functional Requirements

The following requirement statement was formulated based on the information gathered and analysis: A computerized hostel management system including a booking system. Students create accounts and log in to the system. They select the hostel building they want, view the rooms in the hostel, check to see if the room is available or already booked. They choose their preferred room and see the price of the room. They select their payment method and proceed to pay. They receive a confirmation email when payment has been accepted. The hostel manager will create rooms, confirm bookings, monitor payments, manage rooms and post announcements. The manager will also be able to create an inventory of the items provided by the hostel. Students will also be able to report problems through the system.

2.5.2 Non-Functional Requirements

The non-functional requirements of the new system will include:

- Security

  The system will handle sensitive data of students and the hostel; therefore, it must be secure.

- Accessibility

  The system will be built to enable everyone to interact with it.

- Availability

  The system must always be available when needed.

- Reliability

  The system should always perform correctly. It must produce correct output when given input/data

Chapter 3: System Architecture and Design

This chapter details the design & architecture of the system. How users will interact with the system is demonstrated in the chapter with activity and use case diagrams. The technologies used in developing the system will also be highlighted in the chapter.

3.1 System Overview and Architecture

The proposed system will be developed using the Three-Tier Architecture. "*A three-tier architecture is a type of software architecture which is composed of three "tiers" or "layers" of logical computing. The logical processing, access to data and interaction by users are built independently*" [8]. It is a client-server architecture. The 1st tier – Presentation Tier, refers to the system's front-end, which consists of the user interface. The presentation tier will be developed using HTML, JavaScript, CSS and Ajax web technologies. Bootstrap framework will also be utilized in front-end development. The 2nd tier – The application tier refers to the backend of the system, which consists of the functional process logic of the system. It is responsible for the performance of detailed processes and drives the system's core capabilities. PHP will be used to develop this tier. The Data Tier is the 3rd tier. It consists of the database system and the data access layer. This tier handles the data being used by the system and the interface for the database. The application tier accesses data from this tier through API calls. This tier will be developed using MySQL relational database module.

3.2 Database Design

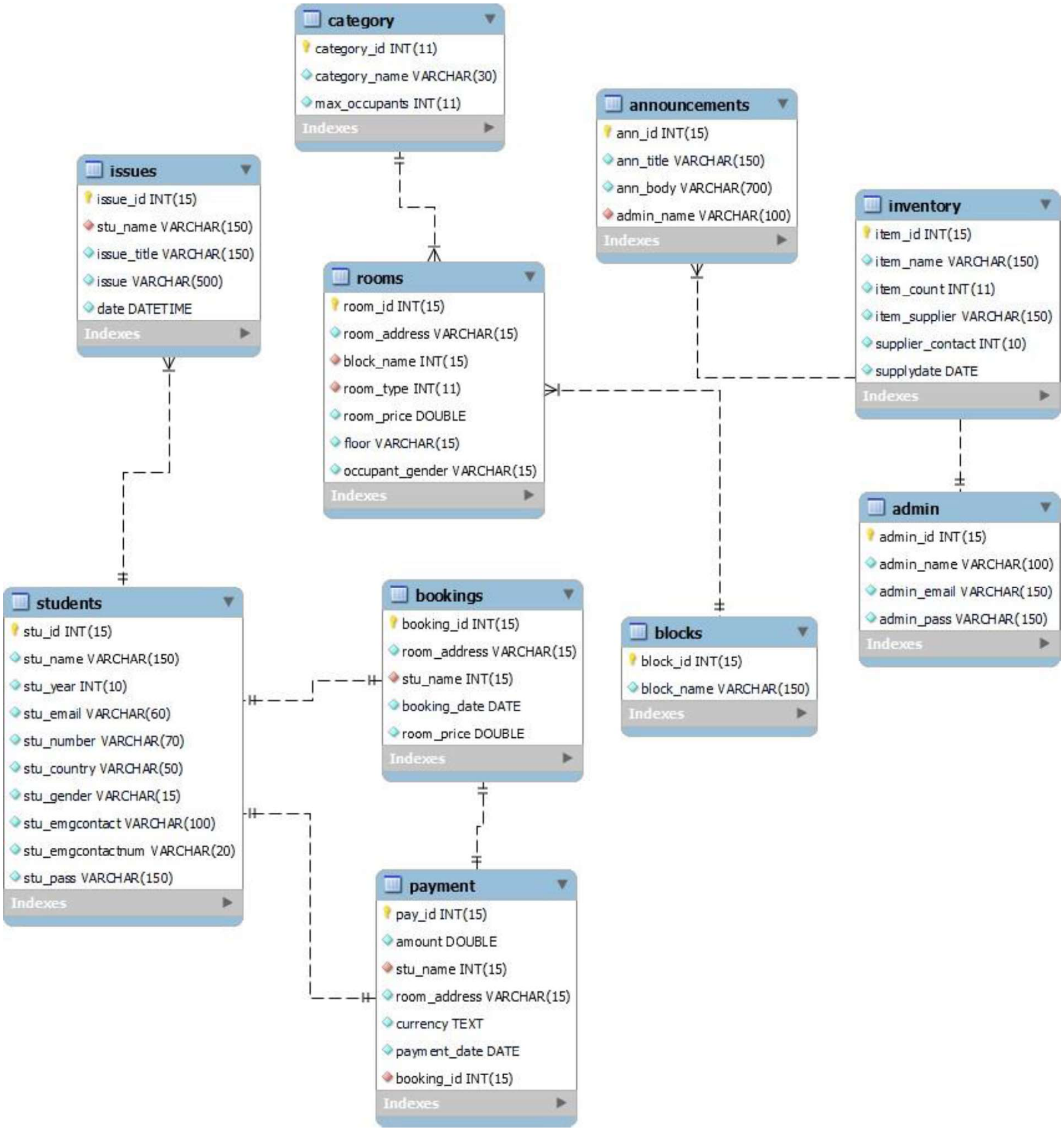


Figure 3.1 Entity Relationship Diagram

3.3 Activity Diagram

The activity diagram gives a visual representation of the steps the student will take to book a room and the actions taken by the administrator to manage the system.

3.3.1 Activity Diagram for Student

This diagram represents the various steps the student will take to book a room on the system. The student will first have to create an account and log in. The student will then select the book a room option from the menu on their page. The student will choose which of the two hostel buildings they want a room in. They will be given a list of rooms based on their gender that matches their preferences above, from which they will select the room they want. They will then move on to make payment using the integrated online payment provider. They will receive an email to confirm their booking, which finalizes the booking process.

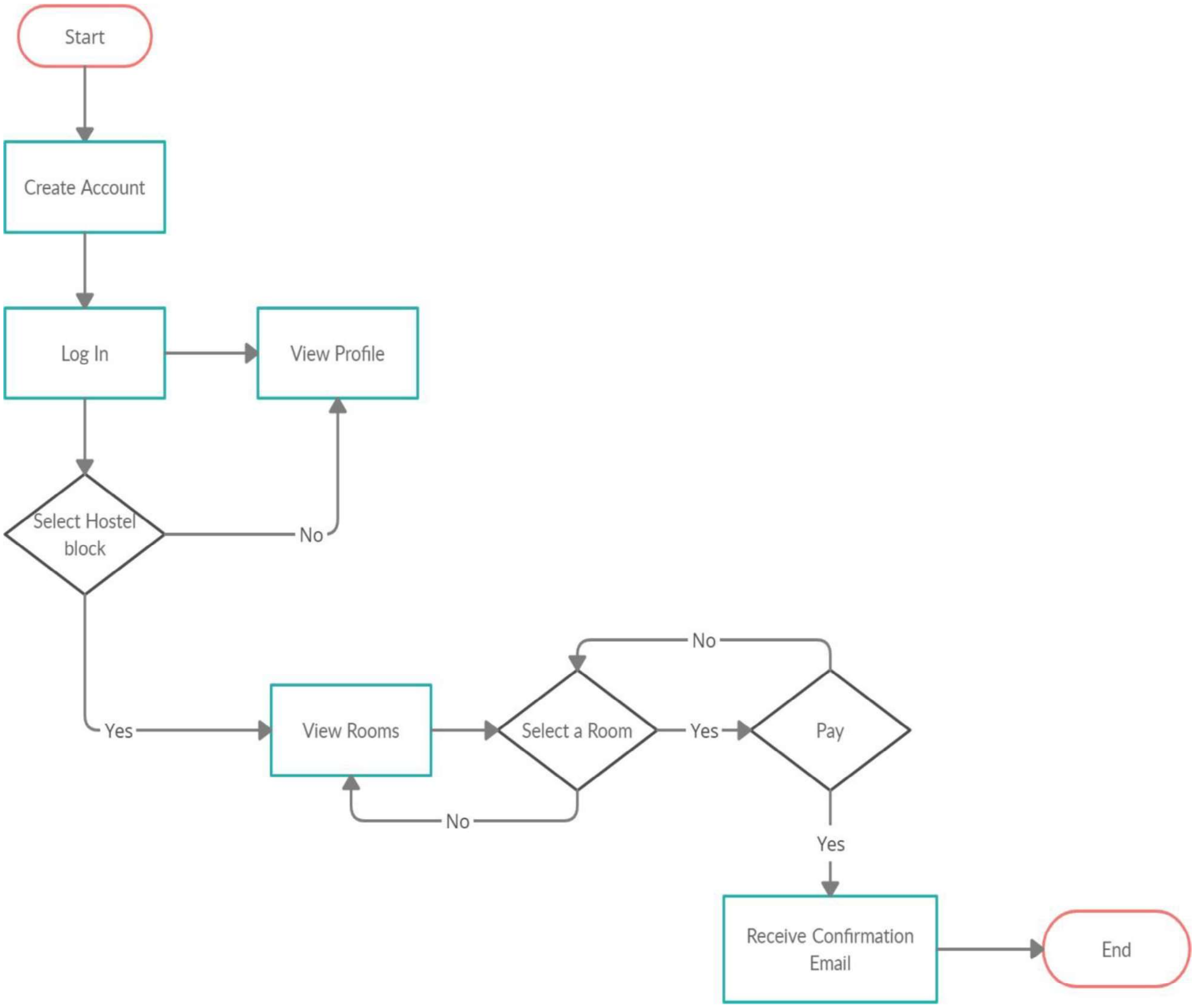


Figure 3.2 Activity diagram showing the activities by the students

3.3.2 Activity Diagram for Administrator

This diagram represents the steps taken by the administrator to manage the system. The administrator will first have to log in. The administrator can then add or remove rooms, monitor payments, post announcements, update inventory and check student profiles.

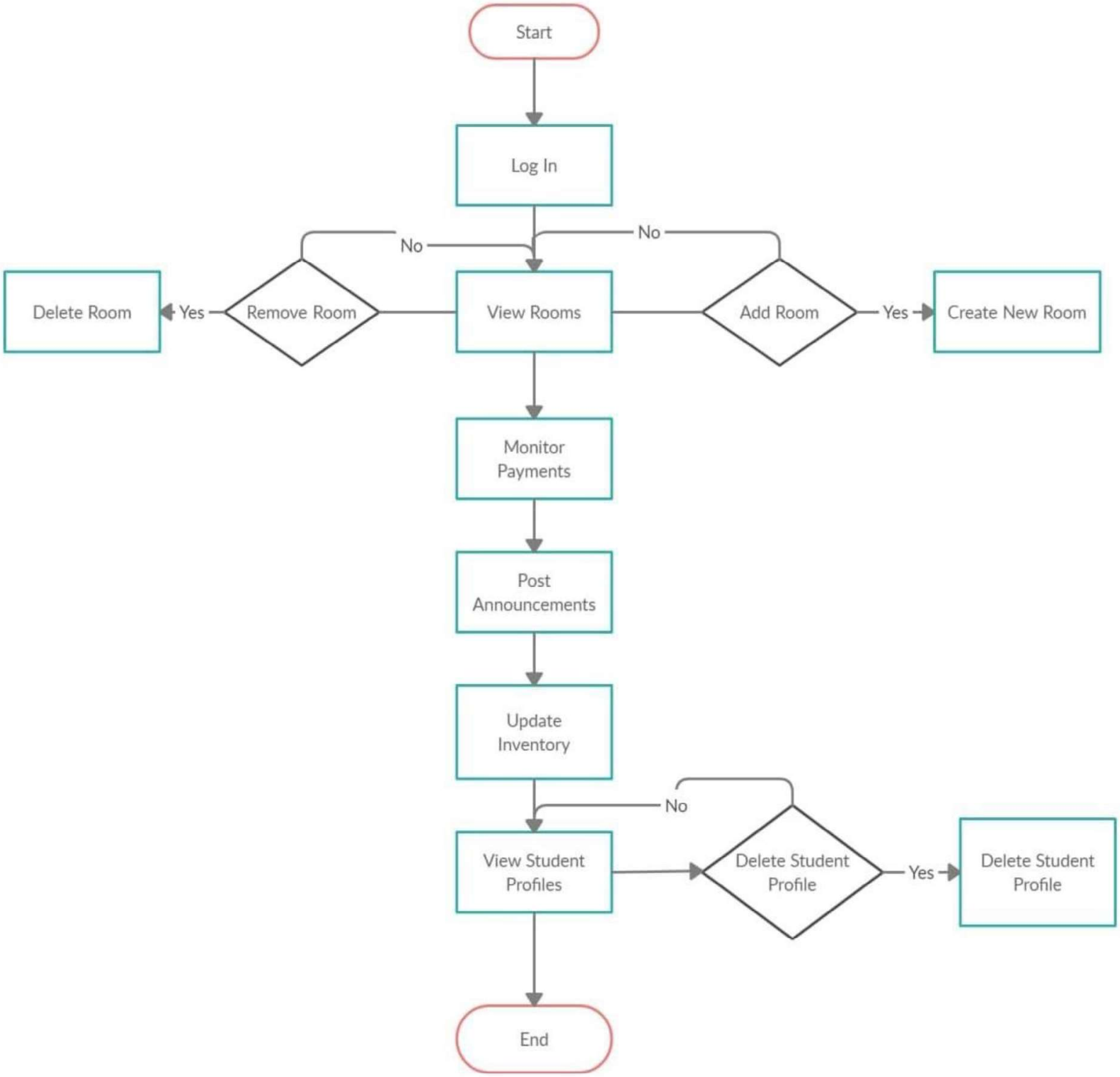


Figure 3.3 Activity diagram showing the activities by the administrator

3.4 Use Case Diagram

"*The purpose of a use case diagram in Unified Modelling Language (UML) is to demonstrate the different ways that a user might interact with a system*" [11].

3.4.1 Use Case Diagram for Student

This diagram represents how the student interacts with the system. The student creates an account and logs in to the system. The student then selects a hostel, an accommodation type, a room and makes payment. The student is then sent an email to confirm their booking.

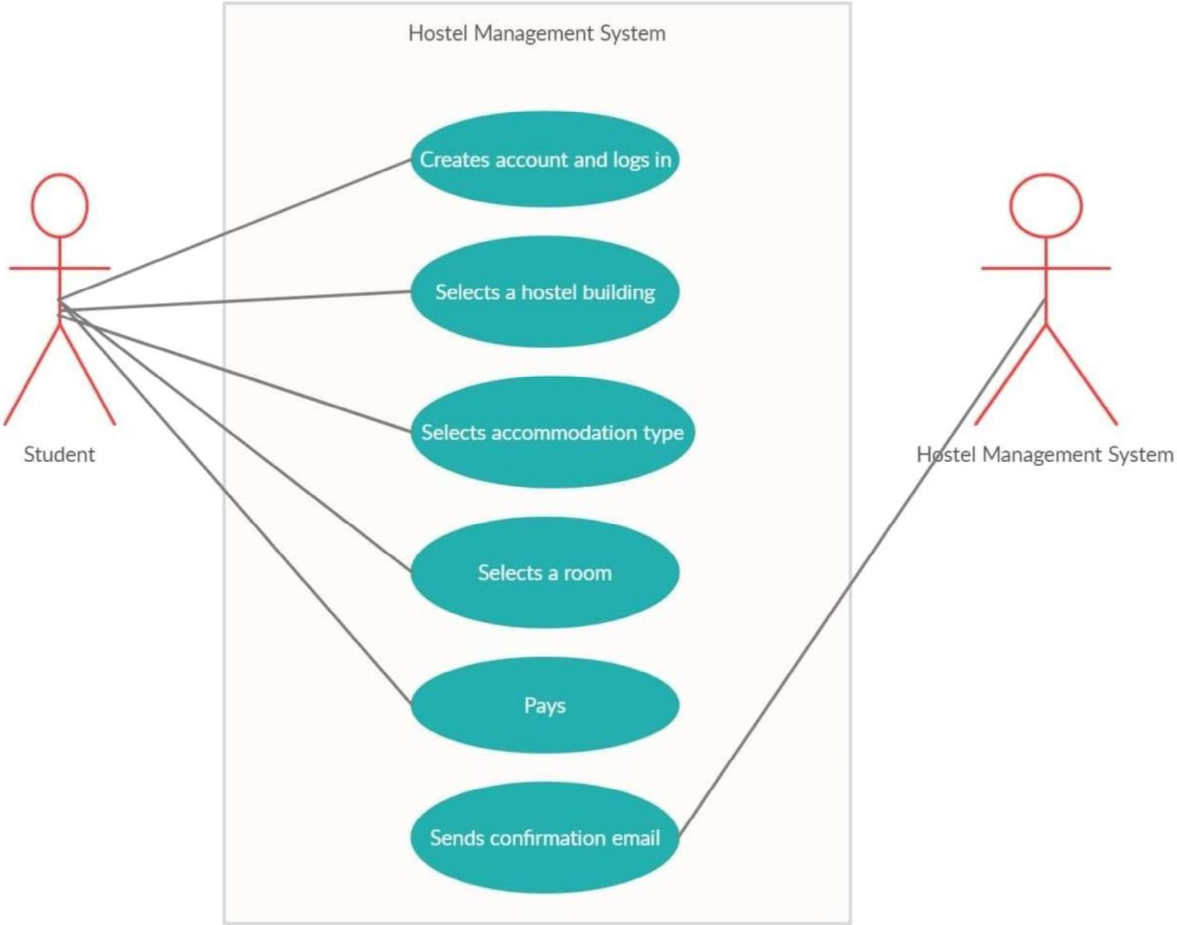


Figure 3.4 Use Case diagram for the student

3.4.2 Use Case Diagram for Administrator

This diagram represents how the administrator interacts with the system. The administrator logs in to the system. He can manage rooms, monitor payments, post announcements, update inventory or check student profiles.

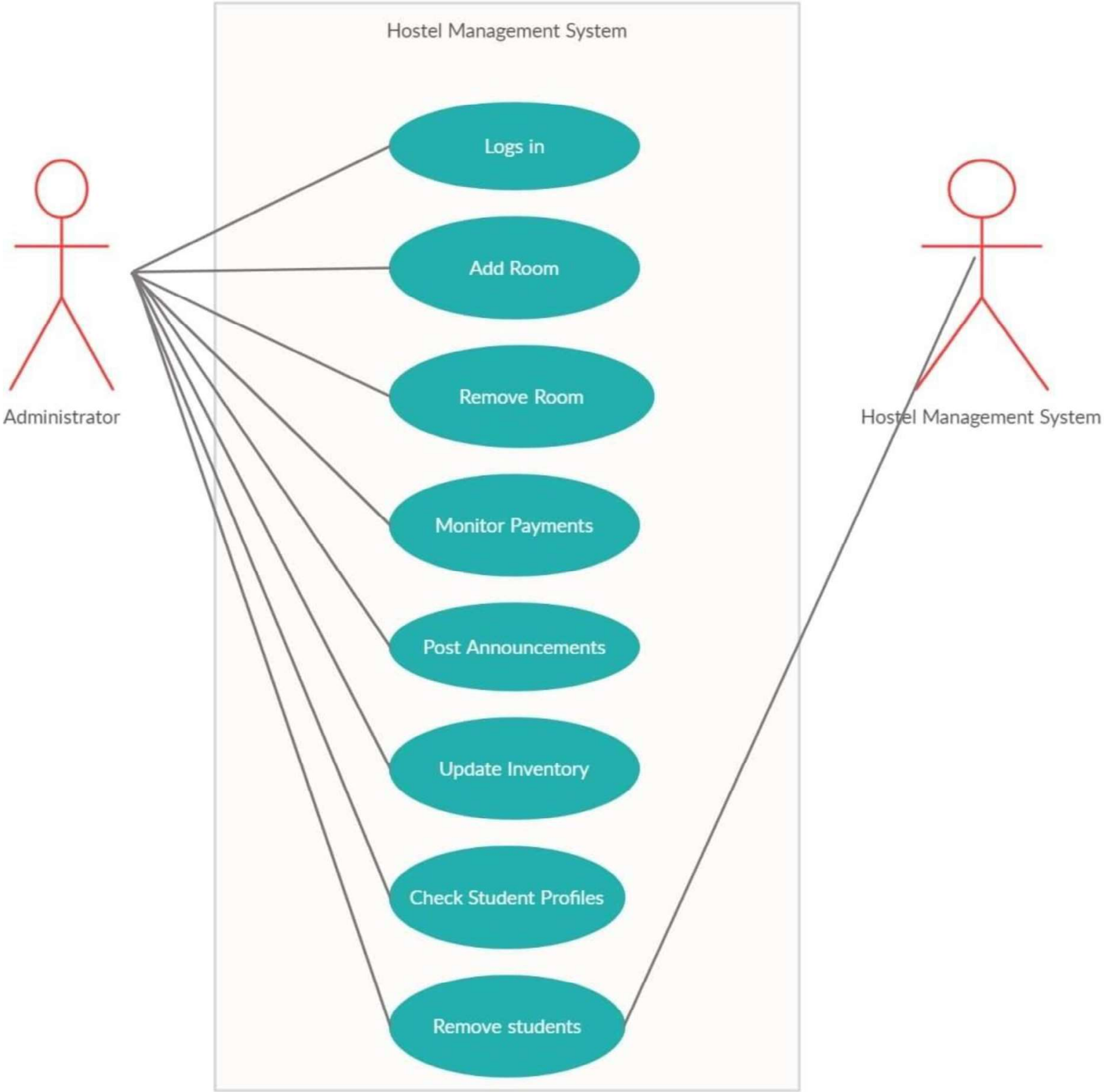


Figure 3.5 Use Case diagram for the administrator

3.5 Technologies Used

3.5.1 Integrated Development Environment

Sublime Text

"*An Integrated Development Environment (IDE) refers to software that enables programmers to consolidate the different aspects of writing a computer program*" [1]. The IDE used to build this system is Sublime Text. Sublime Text is a powerful and easy to use cross-platform editor. It has features such as auto-completion, split editing, multiple selection, dark mode and syntax highlighting. These features make it easy to use for any level of programming. It supports HTML, PHP, SQL, JavaScript, Python, C++, C# and many other programming languages.

3.5.2 Programming Languages

- HTML

HTML stands for Hypertext Markup Language. "*Hypertext Markup Language is a standard markup language for creating content on pages of websites*" [9]. HTML is utilized in front-end development.

- CSS

CSS stands for Cascading Style Sheet. CSS is used for designing documents written in a markup language (e.g. HTML). CSS determines how the web pages will be presented. Colours, fonts and layouts are included on a web page using CSS. CSS is also utilized in front-end development.

- JavaScript

"*JavaScript is a programming language that adds interactivity to your website*" [10]. JavaScript handles elements like button response, alerts, and form validations.

- PHP

PHP stands for Hypertext Preprocessor. It is a server-side programming language that handles interaction with the database, mainly inserting and retrieving information. It is used in backend development.

- SQL

SQL stands for Structured Query Language. It is a programming language used in managing the data in relational database management systems.

3.5.3 Application Programming Interface (APIs)

"*APIs define the interaction between multiple systems. They allow applications to interact with and access data from external systems*" [15].

- PHPMailer

"*It is used for sending emails from PHP. It has integrated SMTP support which allows for emails to be sent without a local mail server*" [14].

- PayStack API

This is an API provided by PayStack Company Limited to handle online payments. It will be the integrated payment API for the system

Chapter 4: Implementation

This chapter describes the implementation process. It highlights the various techniques, frameworks, APIs, modules and components of the system, how they were implemented and their relevance to the system. It also includes screenshots of the pages of the developed system and snippets of the code responsible for those pages.

4.1 Implementation Techniques and Constraints

The hostel management system was developed as a web-based system. Since it has a booking feature, it needed to be web-based to enable students to access it from wherever they are provided they are connected to the internet. Other reasons for developing the web-based system include:

- Accessibility across devices

Web-based systems can be used on various browsers and work on every operating system. Users (administrators and students) will be able to use the system regardless of the type of devices they are using. The only requirement is that the device is connected to the internet.

- Ease of customization

Web-based systems are dynamic and can be easily modified to meet the organization's changing needs.

- Increased efficiency

Web-based systems enable businesses to streamline their processes. Conventional methods get replaced with workflow-based solutions and lead to improvements in business processes.

This particular system will allow the hostel to streamline its booking, payment and incident management processes.

- Availability of Information

The web-based system will enable the administrators to get access to real-time information.

The dashboard feature provides them with information necessary for decision making.

4.1.1 Implementation Techniques

The development of the system was done using the incremental model. The requirements for the system were broken down into multiple modules and implemented sequentially. Getting an ideal template to use for the interface was the first objective. The initial plan was to have one template, but I ended up having to use two. The need for a landing page for the hostel led to the use of two templates; one for the landing page and the other for the management system. The landing page was, however, not part of the scope of the project. It contains information about the hostel and essentially acts as a gateway to the management system. The creation of a landing page was one of much iteration during the development process. These iterations, however, did not interrupt the development process because they followed the agile methodology. This made it possible to make changes to make the system better. The programming for the system was done using the object-oriented programming style. Using classes and objects ensured reusability of code and easy implementation of changes. To make the development process less cumbersome, frameworks, APIs and libraries were also used. Features such as payment, automatic emailing and dashboards were implemented using these libraries, frameworks and APIs.

4.1.2 Implementation Constraints

One major constraint was getting the requirements for getting a payment API. The payment platform I initially wanted to use was PayLink, but I had to shift to PaySwitch. PayLink requires a business certificate and some other business documents which were not readily available. PaySwitch, however, did not require these things and also provided a testing platform. This enabled me to test the payment API with my system during the various stages of development. Aside from that, integrating the APIs was also generally quite challenging. It took a lot of hours and contributions from multiple sources to successfully implement the APIs.

The database design was another constraint faced during the development stage. Given the changes that needed to be made from time to time, problems were frequently encountered with the database. The database had to be restructured almost every time till the end of the project.

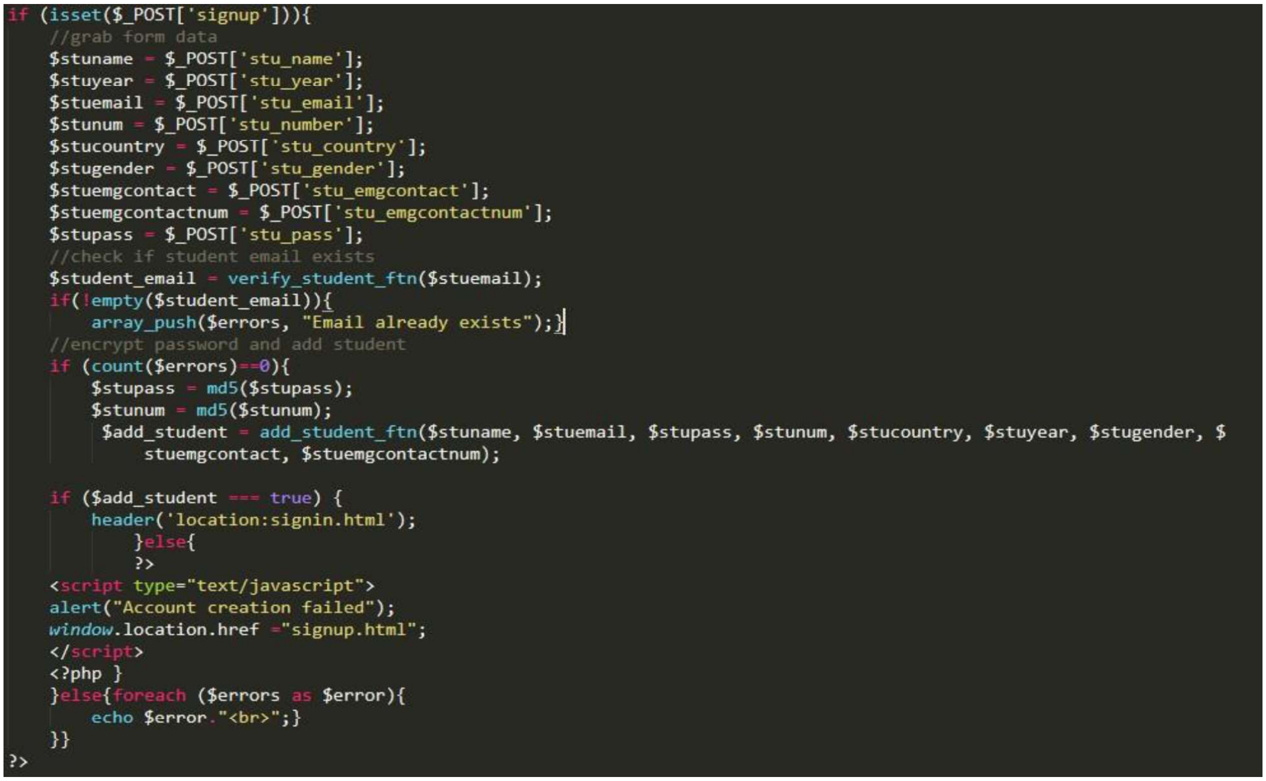
Due to the global pandemic, contact with other humans has been limited. This affected my project because it was difficult getting responses from stakeholders. I had to constantly remind people to answer the questionnaires I sent them, and this delayed progress of the project.

4.2 System Modules

The following modules were developed according to the requirements of the system:

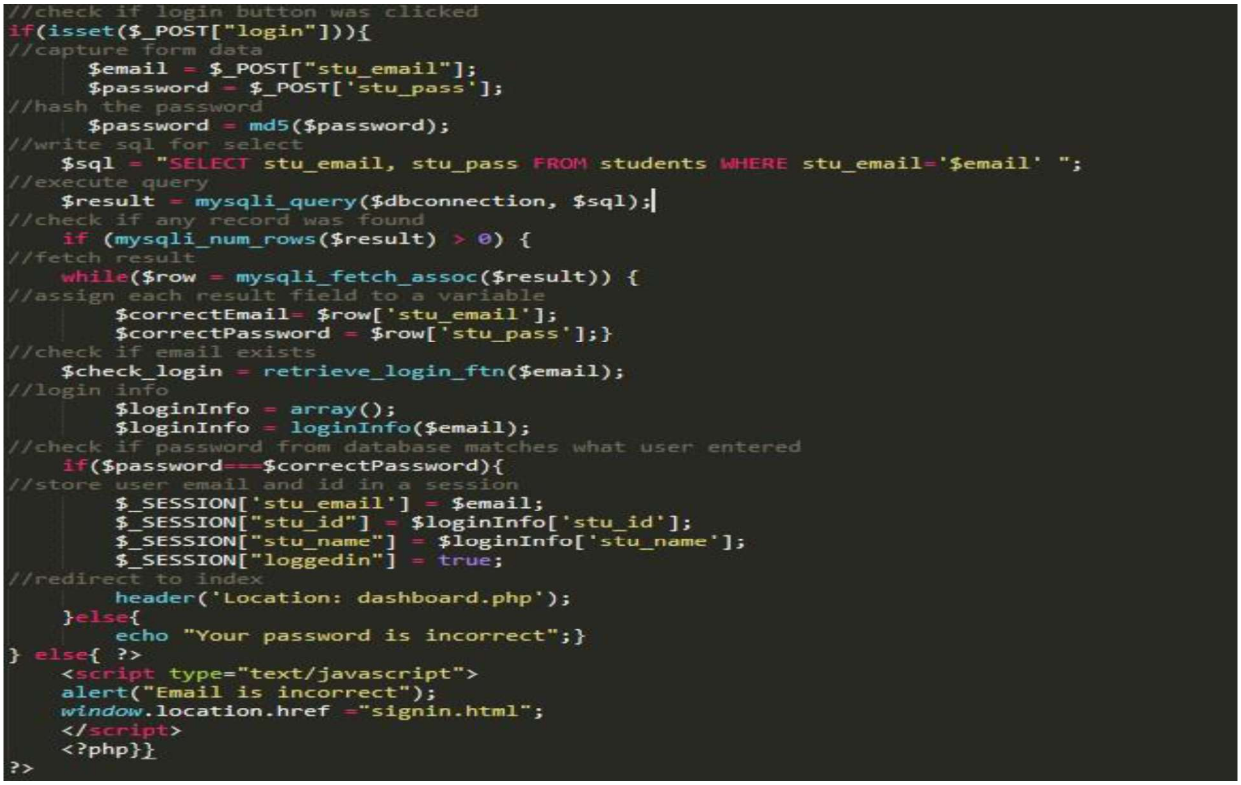
- **Account creation and login**

  This module enabled users to create accounts and log in to the system to use it for its intended purposes.



```
if (isset($_POST['signup'])){
    //grab form data
    $stuname = $_POST['stu_name'];
    $stuyear = $_POST['stu_year'];
    $stuemail = $_POST['stu_email'];
    $stunum = $_POST['stu_number'];
    $stucountry = $_POST['stu_country'];
    $stugender = $_POST['stu_gender'];
    $stuemgcontact = $_POST['stu_emgcontact'];
    $stuemgcontactnum = $_POST['stu_emgcontactnum'];
    $stupass = $_POST['stu_pass'];
    //check if student email exists
    $student_email = verify_student_ftn($stuemail);
    if(!empty($student_email)){
        array_push($errors, "Email already exists");}
    //encrypt password and add student
    if (count($errors)==0){
        $stupass = md5($stupass);
        $stunum = md5($stunum);
         $add_student = add_student_ftn($stuname, $stuemail, $stupass, $stunum, $stucountry, $stuyear, $stugender, $
             stuemgcontact, $stuemgcontactnum);

    if ($add_student === true) {
        header('location:signin.html');
            }else{
            ?>
    <script type="text/javascript">
    alert("Account creation failed");
    window.location.href ="signup.html";
    </script>
    <?php }
    }else{foreach ($errors as $error){
        echo $error."<br>";}
    }}
?>
```

Figure 4.1 Snippet of signup process code



```
//check if login button was clicked
if(isset($_POST["login"])){
//capture form data
      $email = $_POST["stu_email"];
      $password = $_POST['stu_pass'];
//hash the password
      $password = md5($password);
//write sql for select
    $sql = "SELECT stu_email, stu_pass FROM students WHERE stu_email='$email' ";
//execute query
    $result = mysqli_query($dbconnection, $sql);
//check if any record was found
    if (mysqli_num_rows($result) > 0) {
//fetch result
    while($row = mysqli_fetch_assoc($result)) {
//assign each result field to a variable
        $correctEmail= $row['stu_email'];
        $correctPassword = $row['stu_pass'];}
//check if email exists
    $check_login = retrieve_login_ftn($email);
//login info
        $loginInfo = array();
        $loginInfo = loginInfo($email);
//check if password from database matches what user entered
    if($password===$correctPassword){
//store user email and id in a session
        $_SESSION['stu_email'] = $email;
        $_SESSION["stu_id"] = $loginInfo['stu_id'];
        $_SESSION["stu_name"] = $loginInfo['stu_name'];
        $_SESSION["loggedin"] = true;
//redirect to index
        header('Location: dashboard.php');
    }else{
        echo "Your password is incorrect";}
} else{ ?>
    <script type="text/javascript">
    alert("Email is incorrect");
    window.location.href ="signin.html";
    </script>
    <?php}}
?>
```

Figure 4.2 Snippet of login process code

- **Display**

The display module was implemented to enable users of the system to view rooms, profiles, announcements and reports etc.

  - **Rooms Page**

    This page displays all the rooms created by the administrator.

  - **Profile Page**

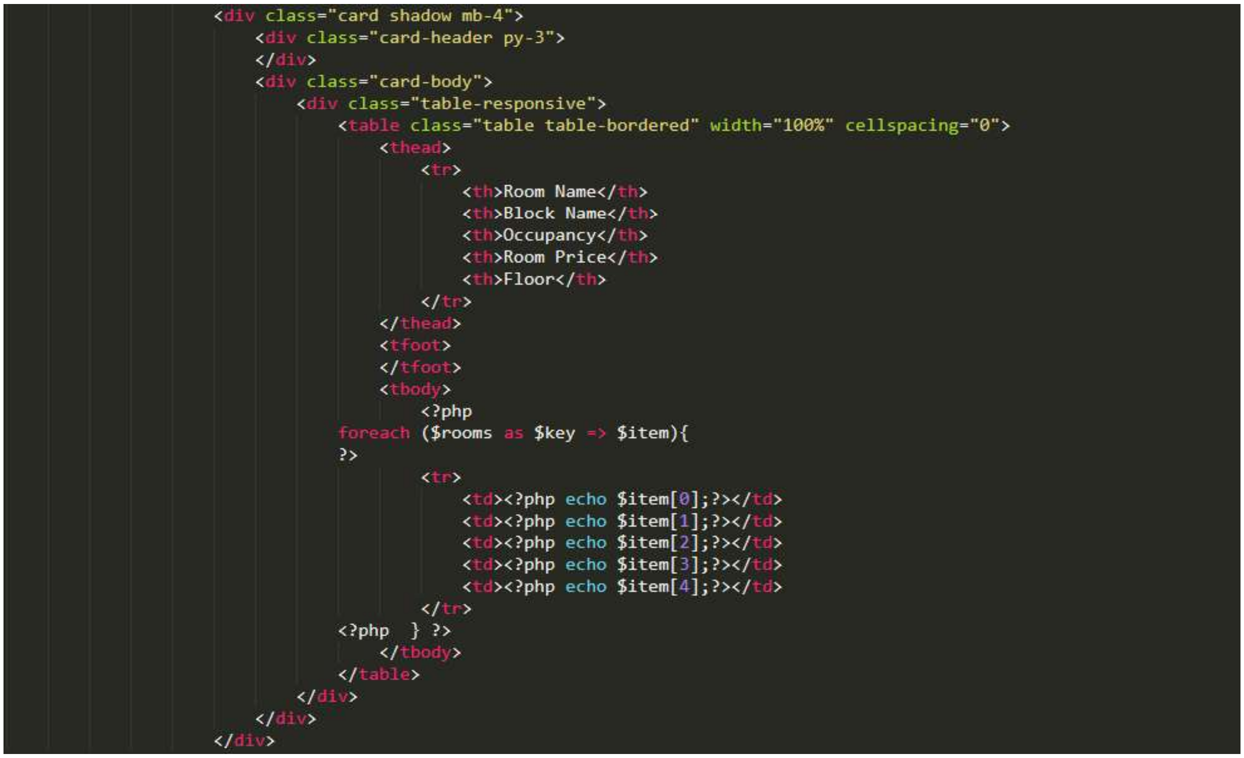
    This page displays the details of a selected user.

  - **Announcements Page**

    This page displays all announcements made by the administrator

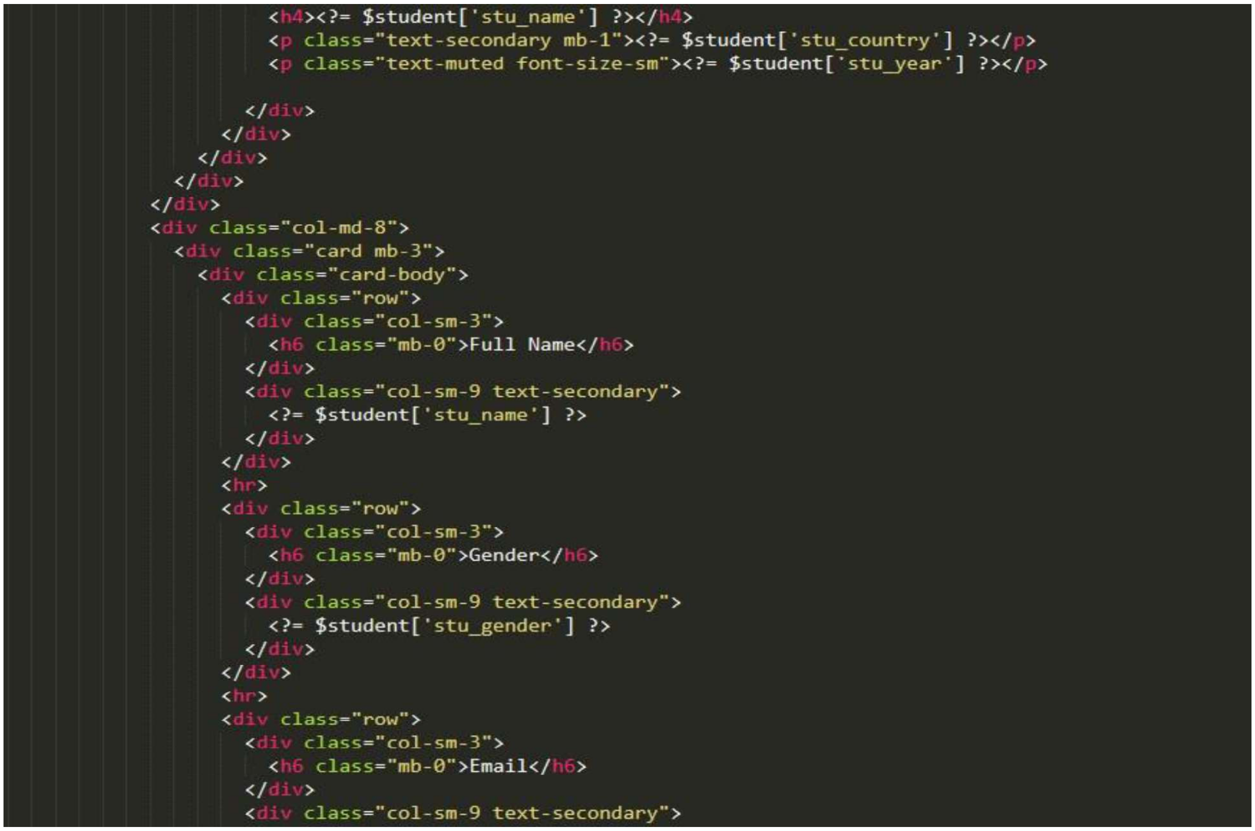
  - **Reports Page**

    This page displays all the issues that students have reported to the administrator.



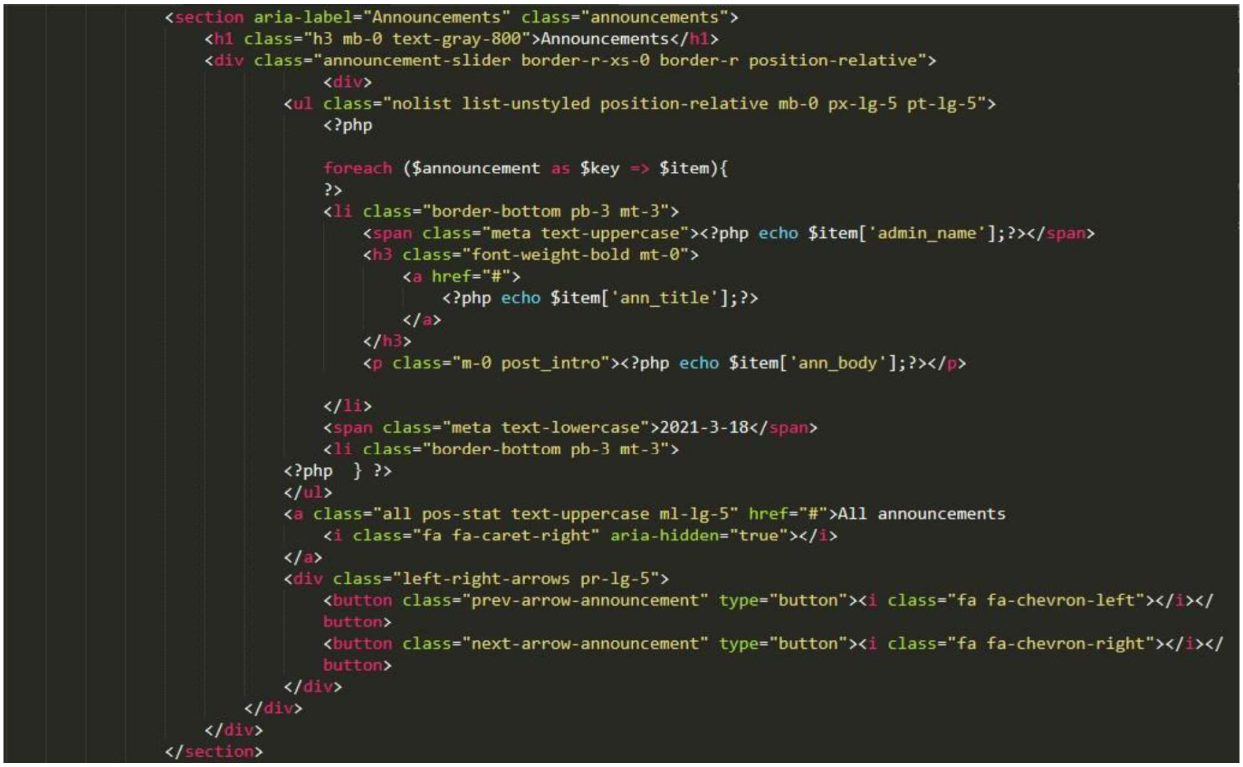
```
<div class="card shadow mb-4">
    <div class="card-header py-3">
    </div>
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-bordered" width="100%" cellspacing="0">
                <thead>
                    <tr>
                        <th>Room Name</th>
                        <th>Block Name</th>
                        <th>Occupancy</th>
                        <th>Room Price</th>
                        <th>Floor</th>
                    </tr>
                </thead>
                <tfoot>
                </tfoot>
                <tbody>
                    <?php
            foreach ($rooms as $key => $item){
            ?>
                    <tr>
                        <td><?php echo $item[0];?></td>
                        <td><?php echo $item[1];?></td>
                        <td><?php echo $item[2];?></td>
                        <td><?php echo $item[3];?></td>
                        <td><?php echo $item[4];?></td>
                    </tr>
            <?php  } ?>
                </tbody>
            </table>
        </div>
    </div>
</div>
```

Figure 4.3 Snippet of rooms display code



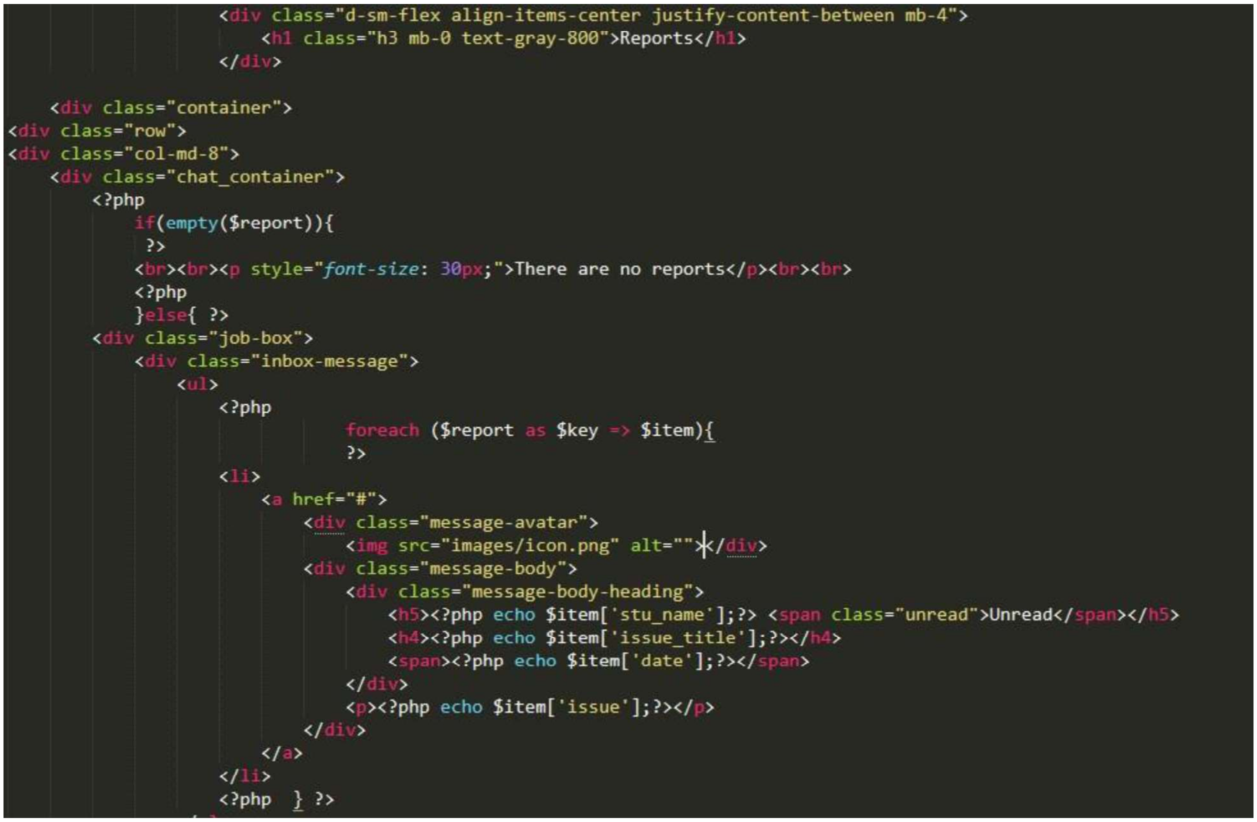
```
          <h4><?= $student['stu_name'] ?></h4>
          <p class="text-secondary mb-1"><?= $student['stu_country'] ?></p>
          <p class="text-muted font-size-sm"><?= $student['stu_year'] ?></p>

        </div>
      </div>
    </div>
  </div>
</div>
<div class="col-md-8">
  <div class="card mb-3">
    <div class="card-body">
      <div class="row">
        <div class="col-sm-3">
          <h6 class="mb-0">Full Name</h6>
        </div>
        <div class="col-sm-9 text-secondary">
          <?= $student['stu_name'] ?>
        </div>
      </div>
      <hr>
      <div class="row">
        <div class="col-sm-3">
          <h6 class="mb-0">Gender</h6>
        </div>
        <div class="col-sm-9 text-secondary">
          <?= $student['stu_gender'] ?>
        </div>
      </div>
      <hr>
      <div class="row">
        <div class="col-sm-3">
          <h6 class="mb-0">Email</h6>
        </div>
        <div class="col-sm-9 text-secondary">
```

Figure 4.4 Snippet of profile page code



```
<section aria-label="Announcements" class="announcements">
    <h1 class="h3 mb-0 text-gray-800">Announcements</h1>
    <div class="announcement-slider border-r-xs-0 border-r position-relative">
                <div>
            <ul class="nolist list-unstyled position-relative mb-0 px-lg-5 pt-lg-5">
                <?php

                foreach ($announcement as $key => $item){
                ?>
                <li class="border-bottom pb-3 mt-3">
                    <span class="meta text-uppercase"><?php echo $item['admin_name'];?></span>
                    <h3 class="font-weight-bold mt-0">
                        <a href="#">
                            <?php echo $item['ann_title'];?>
                        </a>
                    </h3>
                    <p class="m-0 post_intro"><?php echo $item['ann_body'];?></p>

                </li>
                <span class="meta text-lowercase">2021-3-18</span>
                <li class="border-bottom pb-3 mt-3">
            <?php  } ?>
            </ul>
            <a class="all pos-stat text-uppercase ml-lg-5" href="#">All announcements
                <i class="fa fa-caret-right" aria-hidden="true"></i>
            </a>
            <div class="left-right-arrows pr-lg-5">
                <button class="prev-arrow-announcement" type="button"><i class="fa fa-chevron-left"></i></
                button>
                <button class="next-arrow-announcement" type="button"><i class="fa fa-chevron-right"></i></
                button>
            </div>
        </div>
    </div>
</section>
```

Figure 4.5 Snippet of announcements display code



```
                    <div class="d-sm-flex align-items-center justify-content-between mb-4">
                        <h1 class="h3 mb-0 text-gray-800">Reports</h1>
                    </div>

    <div class="container">
<div class="row">
<div class="col-md-8">
    <div class="chat_container">
        <?php
            if(empty($report)){
             ?>
            <br><br><p style="font-size: 30px;">There are no reports</p><br><br>
            <?php
            }else{ ?>
        <div class="job-box">
            <div class="inbox-message">
                <ul>
                    <?php
                                foreach ($report as $key => $item){
                                ?>
                    <li>
                        <a href="#">
                            <div class="message-avatar">
                                <img src="images/icon.png" alt=""></div>
                            <div class="message-body">
                                <div class="message-body-heading">
                                    <h5><?php echo $item['stu_name'];?> <span class="unread">Unread</span></h5>
                                    <h4><?php echo $item['issue_title'];?></h4>
                                    <span><?php echo $item['date'];?></span>
                                </div>
                                <p><?php echo $item['issue'];?></p>
                            </div>
                        </a>
                    </li>
                    <?php  } ?>
```

Figure 4.6 Snippet of reports display code

- **Booking**

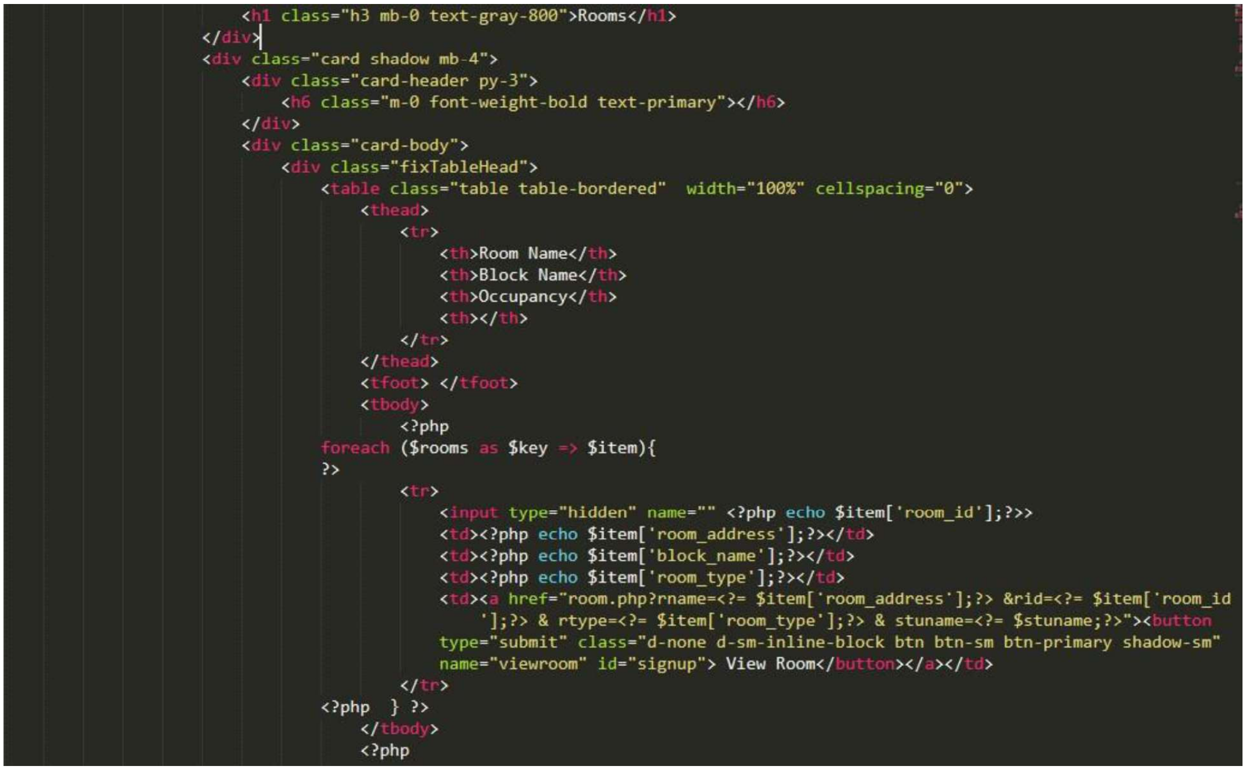
  This module handles what the user sees and does to book a room on the system successfully. Its sub-modules include:

  - **Room Selection**

    The user can see all rooms based on their gender, available in the hostel because of this module.

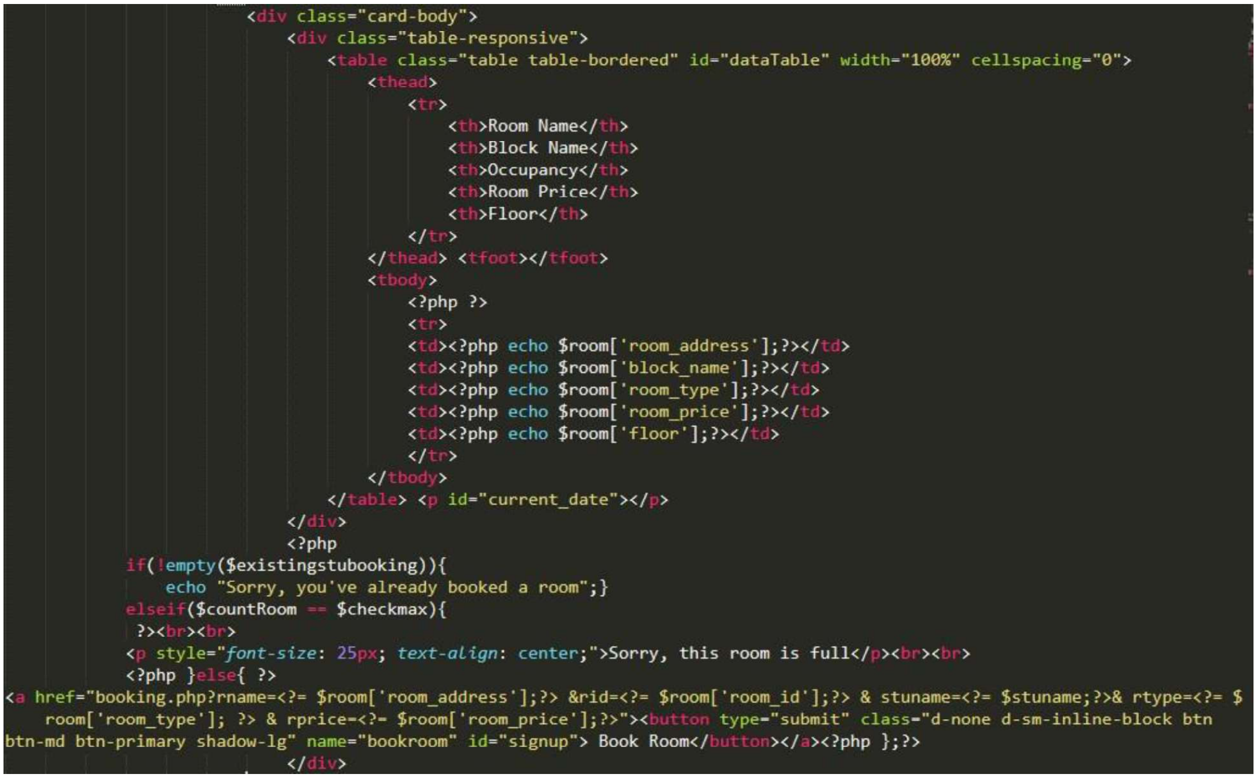
  - **Bill Viewing**

    This module enables the user to see the details of the booking they have just made before paying.



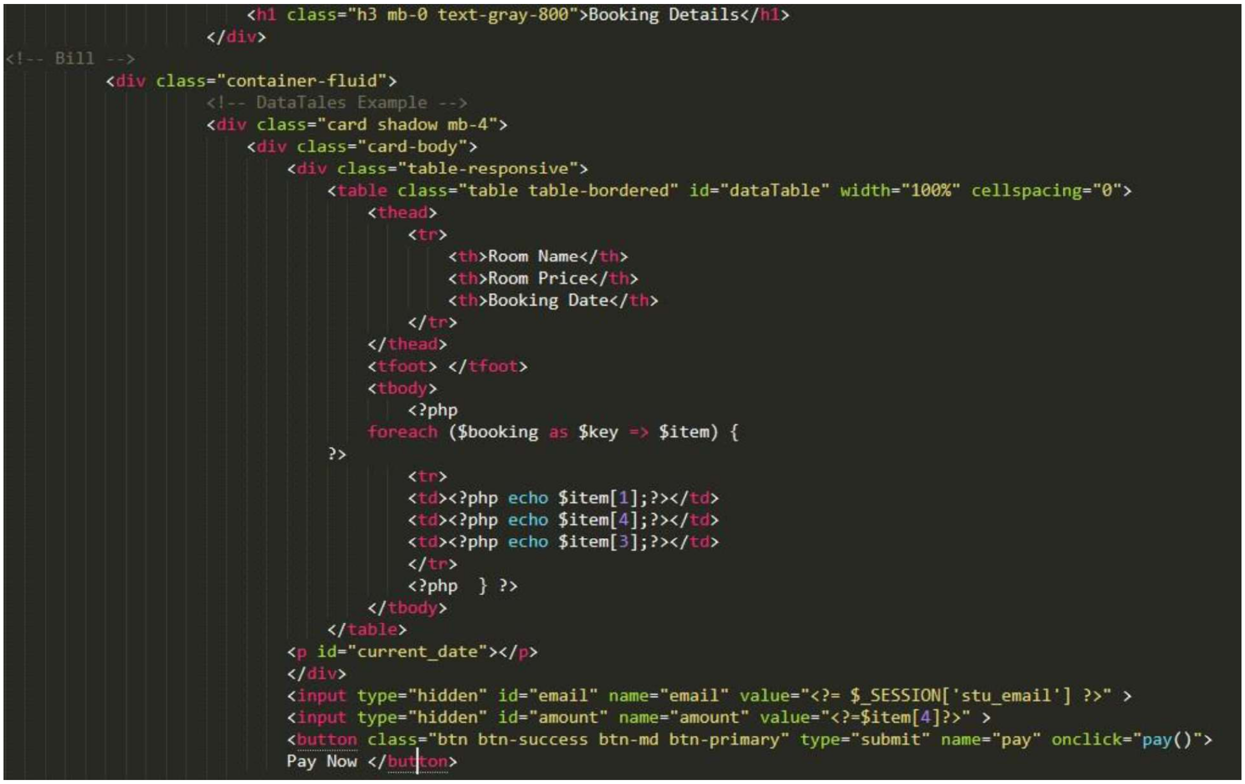
```
    <h1 class="h3 mb-0 text-gray-800">Rooms</h1>
</div>
<div class="card shadow mb-4">
    <div class="card-header py-3">
        <h6 class="m-0 font-weight-bold text-primary"></h6>
    </div>
    <div class="card-body">
        <div class="fixTableHead">
            <table class="table table-bordered"  width="100%" cellspacing="0">
                <thead>
                    <tr>
                        <th>Room Name</th>
                        <th>Block Name</th>
                        <th>Occupancy</th>
                        <th></th>
                    </tr>
                </thead>
                <tfoot> </tfoot>
                <tbody>
                    <?php
            foreach ($rooms as $key => $item){
            ?>
                    <tr>
                        <input type="hidden" name="" <?php echo $item['room_id'];?>>
                        <td><?php echo $item['room_address'];?></td>
                        <td><?php echo $item['block_name'];?></td>
                        <td><?php echo $item['room_type'];?></td>
                        <td><a href="room.php?rname=<?= $item['room_address'];?> &rid=<?= $item['room_id
                            '];?> & rtype=<?= $item['room_type'];?> & stuname=<?= $stuname;?>"><button
                        type="submit" class="d-none d-sm-inline-block btn btn-sm btn-primary shadow-sm"
                        name="viewroom" id="signup"> View Room</button></a></td>
                    </tr>
            <?php  } ?>
                </tbody>
                <?php
```

Figure 4.7 Snippet of room selection code



```
                        <div class="card-body">
                            <div class="table-responsive">
                                <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
                                    <thead>
                                        <tr>
                                            <th>Room Name</th>
                                            <th>Block Name</th>
                                            <th>Occupancy</th>
                                            <th>Room Price</th>
                                            <th>Floor</th>
                                        </tr>
                                    </thead> <tfoot></tfoot>
                                    <tbody>
                                        <?php ?>
                                        <tr>
                                        <td><?php echo $room['room_address'];?></td>
                                        <td><?php echo $room['block_name'];?></td>
                                        <td><?php echo $room['room_type'];?></td>
                                        <td><?php echo $room['room_price'];?></td>
                                        <td><?php echo $room['floor'];?></td>
                                        </tr>
                                    </tbody>
                                </table> <p id="current_date"></p>
                            </div>
                            <?php
            if(!empty($existingstubooking)){
                echo "Sorry, you've already booked a room";}
            elseif($countRoom == $checkmax){
             ?><br><br>
            <p style="font-size: 25px; text-align: center;">Sorry, this room is full</p><br><br>
            <?php }else{ ?>
<a href="booking.php?rname=<?= $room['room_address'];?> &rid=<?= $room['room_id'];?> & stuname=<?= $stuname;?>& rtype=<?= $
    room['room_type']; ?> & rprice=<?= $room['room_price'];?>"><button type="submit" class="d-none d-sm-inline-block btn
btn-md btn-primary shadow-lg" name="bookroom" id="signup"> Book Room</button></a><?php };?>
                            </div>
```

Figure 4.8 Snippet of room selection code (Single room)

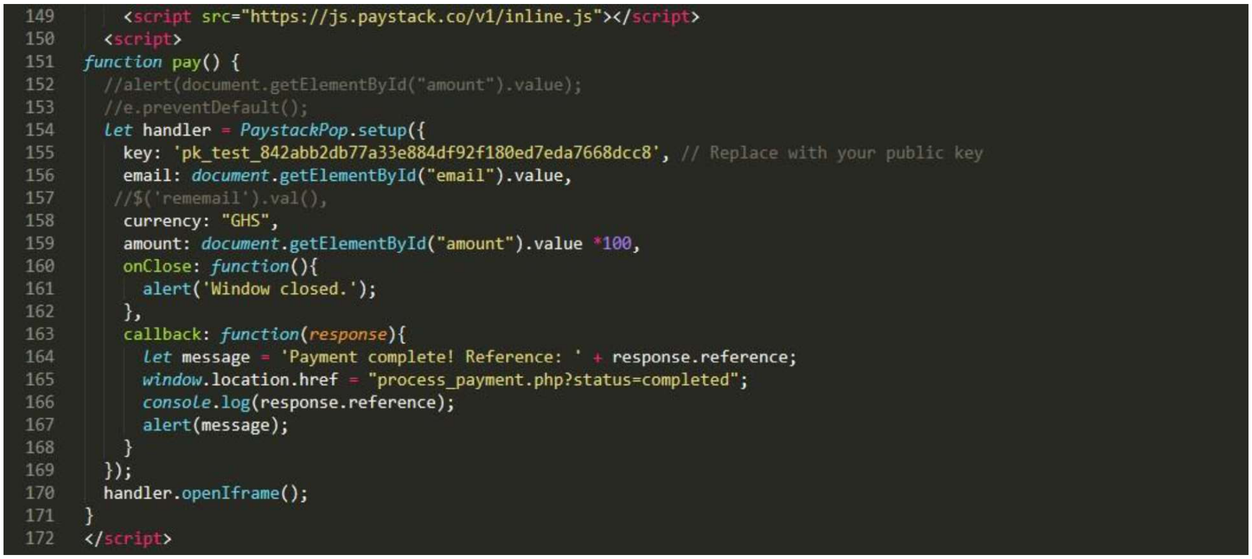


```
                        <h1 class="h3 mb-0 text-gray-800">Booking Details</h1>
                    </div>
<!-- Bill -->
          <div class="container-fluid">
                    <!-- DataTales Example -->
                    <div class="card shadow mb-4">
                        <div class="card-body">
                            <div class="table-responsive">
                                <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
                                    <thead>
                                        <tr>
                                            <th>Room Name</th>
                                            <th>Room Price</th>
                                            <th>Booking Date</th>
                                        </tr>
                                    </thead>
                                    <tfoot> </tfoot>
                                    <tbody>
                                        <?php
                                    foreach ($booking as $key => $item) {
                                ?>
                                        <tr>
                                        <td><?php echo $item[1];?></td>
                                        <td><?php echo $item[4];?></td>
                                        <td><?php echo $item[3];?></td>
                                        </tr>
                                        <?php  } ?>
                                    </tbody>
                                </table>
                            <p id="current_date"></p>
                            </div>
                            <input type="hidden" id="email" name="email" value="<?= $_SESSION['stu_email'] ?>" >
                            <input type="hidden" id="amount" name="amount" value="<?=$item[4]?>" >
                            <button class="btn btn-success btn-md btn-primary" type="submit" name="pay" onclick="pay()">
                            Pay Now </button>
```

Figure 4.9 Snippet of bill viewing code

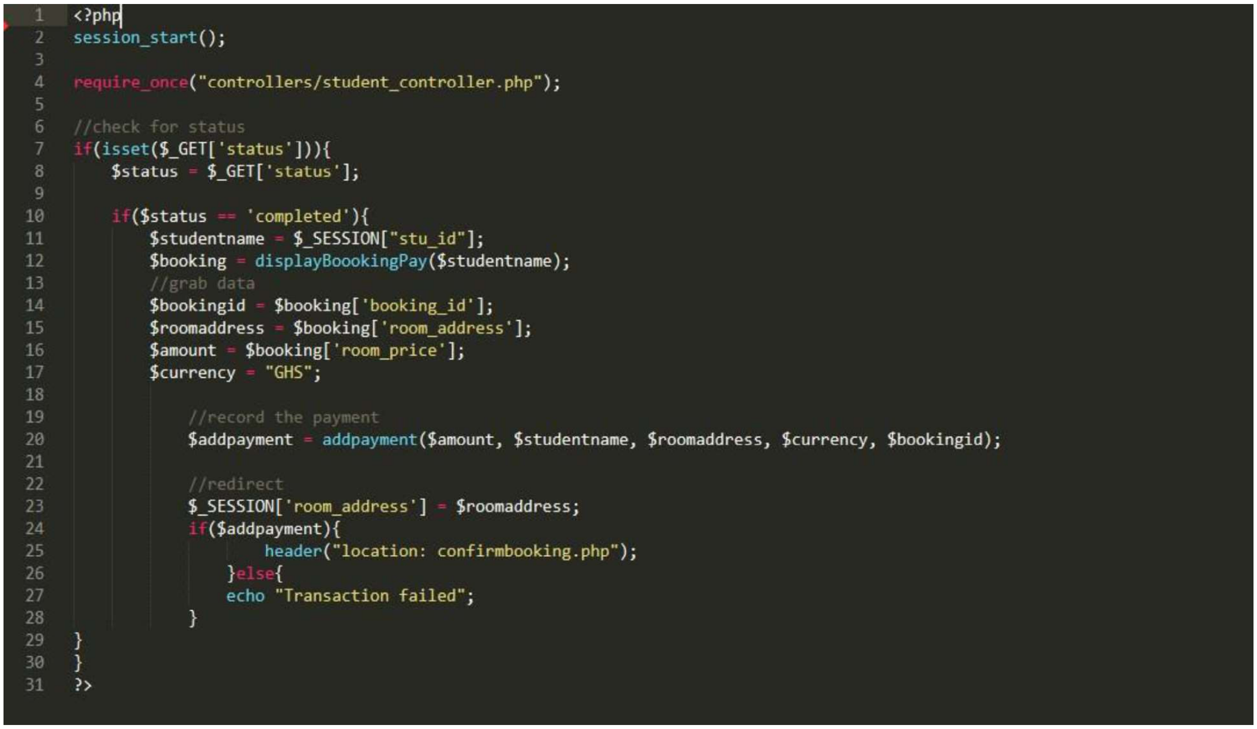
- **Payment**

  This module enables users to make payment for the rooms they have booked. PayStack's API was integrated to make this module successful.



```
    <script src="https://js.paystack.co/v1/inline.js"></script>
  <script>
function pay() {
  //alert(document.getElementById("amount").value);
  //e.preventDefault();
  let handler = PaystackPop.setup({
    key: 'pk_test_842abb2db77a33e884df92f180ed7eda7668dcc8', // Replace with your public key
    email: document.getElementById("email").value,
   //$('rememail').val(),
    currency: "GHS",
    amount: document.getElementById("amount").value *100,
    onClose: function(){
      alert('Window closed.');
    },
    callback: function(response){
      let message = 'Payment complete! Reference: ' + response.reference;
      window.location.href = "process_payment.php?status=completed";
      console.log(response.reference);
      alert(message);
    }
  });
  handler.openIframe();
}
</script>
```

Figure 4.10 Snippet of payment code

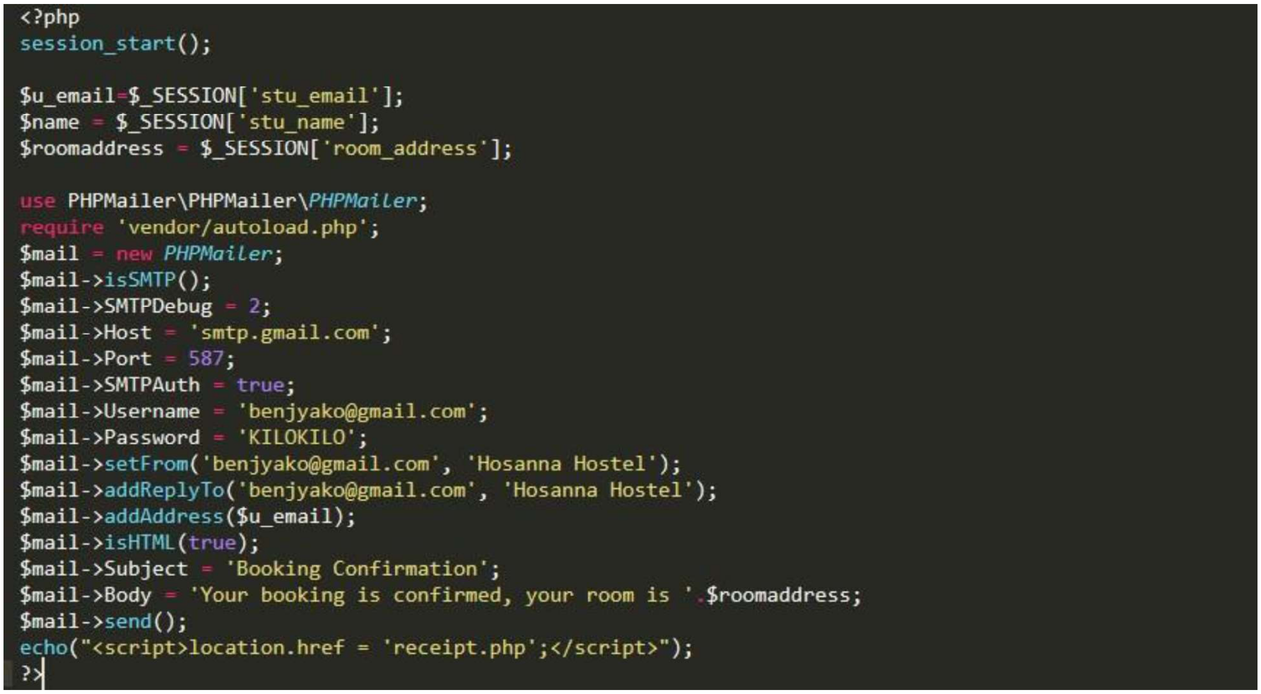


```
<?php
session_start();

require_once("controllers/student_controller.php");

//check for status
if(isset($_GET['status'])){
    $status = $_GET['status'];

    if($status == 'completed'){
        $studentname = $_SESSION["stu_id"];
        $booking = displayBoookingPay($studentname);
        //grab data
        $bookingid = $booking['booking_id'];
        $roomaddress = $booking['room_address'];
        $amount = $booking['room_price'];
        $currency = "GHS";

            //record the payment
            $addpayment = addpayment($amount, $studentname, $roomaddress, $currency, $bookingid);

            //redirect
            $_SESSION['room_address'] = $roomaddress;
            if($addpayment){
                    header("location: confirmbooking.php");
                }else{
                echo "Transaction failed";
            }
}
}
?>
```

Figure 4.11 Snippet of payment process code

- **Confirmation**

PHP Mailer was the API used for this module. It sends an email to users once they have paid for their room to confirm their booking.

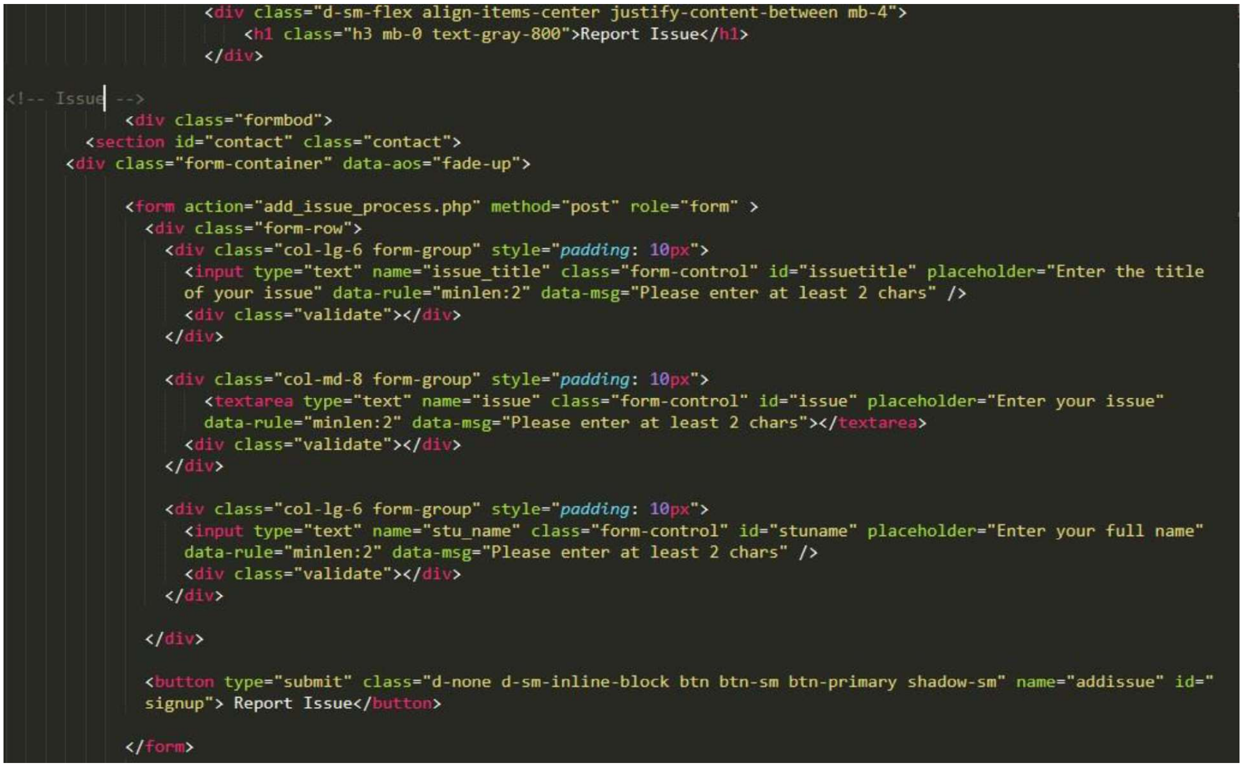


```
<?php
session_start();

$u_email=$_SESSION['stu_email'];
$name = $_SESSION['stu_name'];
$roomaddress = $_SESSION['room_address'];

use PHPMailer\PHPMailer\PHPMailer;
require 'vendor/autoload.php';
$mail = new PHPMailer;
$mail->isSMTP();
$mail->SMTPDebug = 2;
$mail->Host = 'smtp.gmail.com';
$mail->Port = 587;
$mail->SMTPAuth = true;
$mail->Username = 'benjyako@gmail.com';
$mail->Password = 'KILOKILO';
$mail->setFrom('benjyako@gmail.com', 'Hosanna Hostel');
$mail->addReplyTo('benjyako@gmail.com', 'Hosanna Hostel');
$mail->addAddress($u_email);
$mail->isHTML(true);
$mail->Subject = 'Booking Confirmation';
$mail->Body = 'Your booking is confirmed, your room is '.$roomaddress;
$mail->send();
echo("<script>location.href = 'receipt.php';</script>");
?>
```

Figure 4.12 Snippet of booking confirmation with PHP mailer code

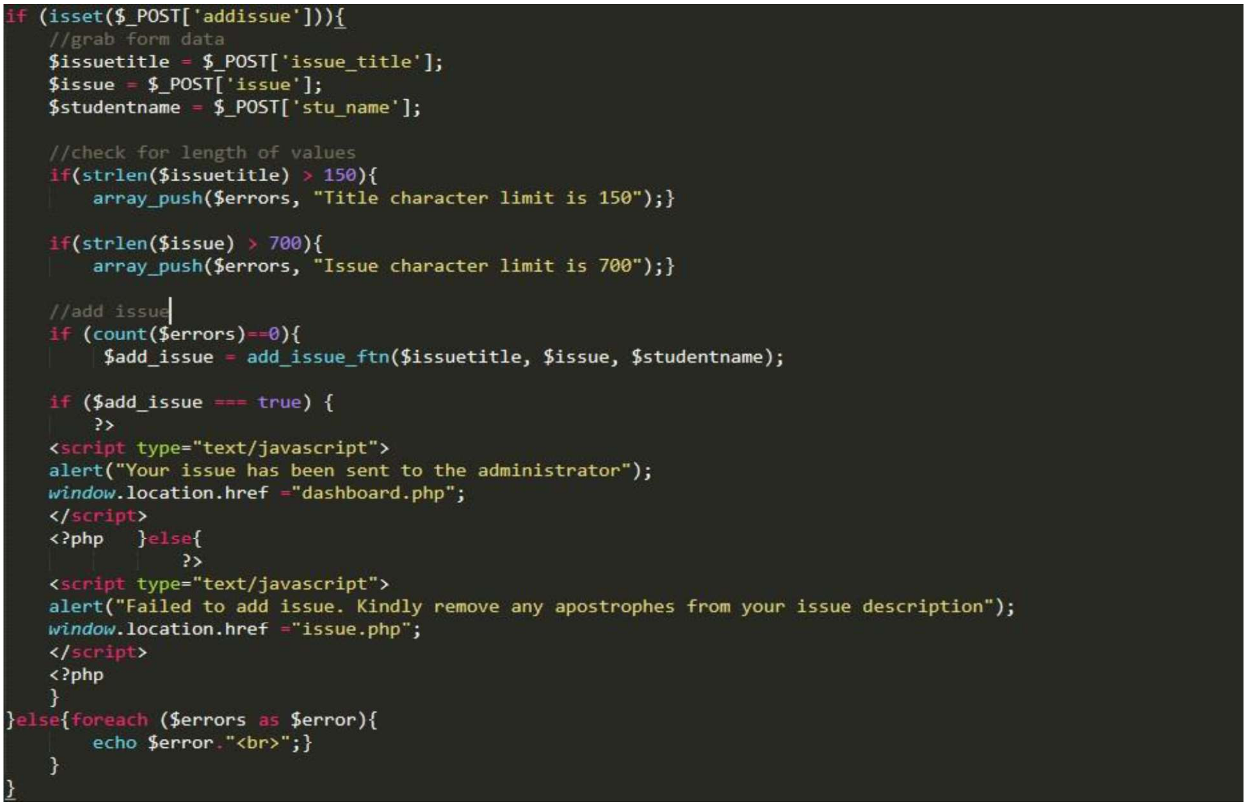
- **Issue Reporting**

This module allows students to report problems they have in the hostel to the administrator.



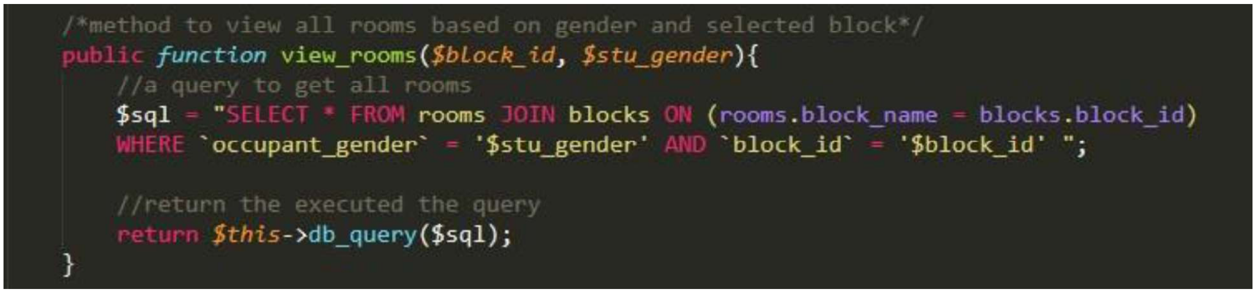
```
                    <div class="d-sm-flex align-items-center justify-content-between mb-4">
                        <h1 class="h3 mb-0 text-gray-800">Report Issue</h1>
                    </div>

<!-- Issue -->
            <div class="formbod">
        <section id="contact" class="contact">
      <div class="form-container" data-aos="fade-up">

            <form action="add_issue_process.php" method="post" role="form" >
              <div class="form-row">
                <div class="col-lg-6 form-group" style="padding: 10px">
                  <input type="text" name="issue_title" class="form-control" id="issuetitle" placeholder="Enter the title
                  of your issue" data-rule="minlen:2" data-msg="Please enter at least 2 chars" />
                  <div class="validate"></div>
                </div>

                <div class="col-md-8 form-group" style="padding: 10px">
                    <textarea type="text" name="issue" class="form-control" id="issue" placeholder="Enter your issue"
                    data-rule="minlen:2" data-msg="Please enter at least 2 chars"></textarea>
                  <div class="validate"></div>
                </div>

                <div class="col-lg-6 form-group" style="padding: 10px">
                  <input type="text" name="stu_name" class="form-control" id="stuname" placeholder="Enter your full name"
                  data-rule="minlen:2" data-msg="Please enter at least 2 chars" />
                  <div class="validate"></div>
                </div>

              </div>

              <button type="submit" class="d-none d-sm-inline-block btn btn-sm btn-primary shadow-sm" name="addissue" id="
              signup"> Report Issue</button>

            </form>
```

Figure 4.13 Snippet of issue reporting code



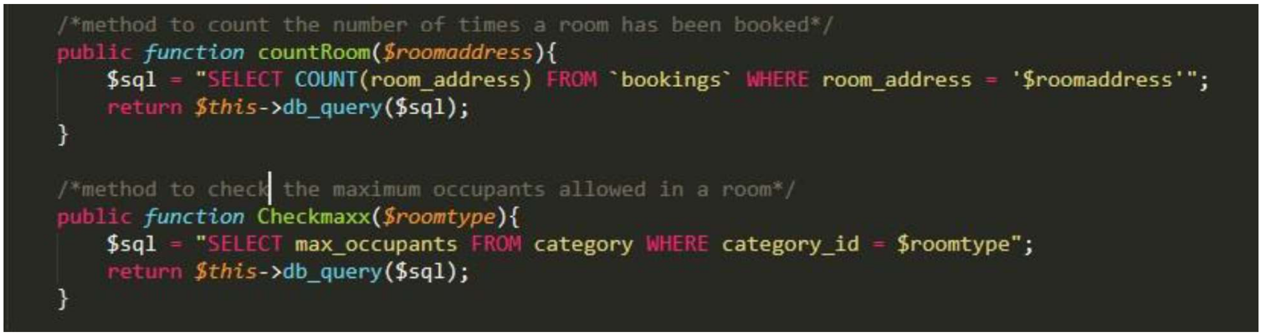
```
if (isset($_POST['addissue'])){
    //grab form data
    $issuetitle = $_POST['issue_title'];
    $issue = $_POST['issue'];
    $studentname = $_POST['stu_name'];

    //check for length of values
    if(strlen($issuetitle) > 150){
        array_push($errors, "Title character limit is 150");}

    if(strlen($issue) > 700){
        array_push($errors, "Issue character limit is 700");}

    //add issue
    if (count($errors)==0){
         $add_issue = add_issue_ftn($issuetitle, $issue, $studentname);

    if ($add_issue === true) {
        ?>
    <script type="text/javascript">
    alert("Your issue has been sent to the administrator");
    window.location.href ="dashboard.php";
    </script>
    <?php   }else{
                ?>
    <script type="text/javascript">
    alert("Failed to add issue. Kindly remove any apostrophes from your issue description");
    window.location.href ="issue.php";
    </script>
    <?php
    }
}else{foreach ($errors as $error){
        echo $error."<br>";}
    }
}
```

Figure 4.14 Snippet of issue report processing code



```
/*method to view all rooms based on gender and selected block*/
public function view_rooms($block_id, $stu_gender){
    //a query to get all rooms
    $sql = "SELECT * FROM rooms JOIN blocks ON (rooms.block_name = blocks.block_id)
    WHERE `occupant_gender` = '$stu_gender' AND `block_id` = '$block_id' ";

    //return the executed the query
    return $this->db_query($sql);
}
```

Figure 4.15 Query to prevent same room booking by different genders



```
/*method to count the number of times a room has been booked*/
public function countRoom($roomaddress){
    $sql = "SELECT COUNT(room_address) FROM `bookings` WHERE room_address = '$roomaddress'";
    return $this->db_query($sql);
}

/*method to check the maximum occupants allowed in a room*/
public function Checkmaxx($roomtype){
    $sql = "SELECT max_occupants FROM category WHERE category_id = $roomtype";
    return $this->db_query($sql);
}
```

Figure 4.16 Queries to avoid overbooking of rooms

- **Database**

The database stores all the information for the system. Since all the modules are connected, the database is of the relational form. The database has ten tables and was programmed using SQL.

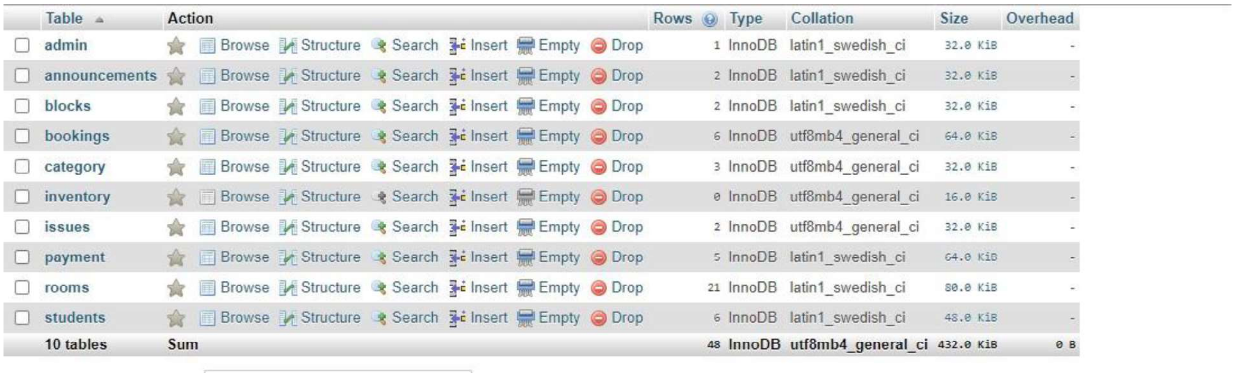
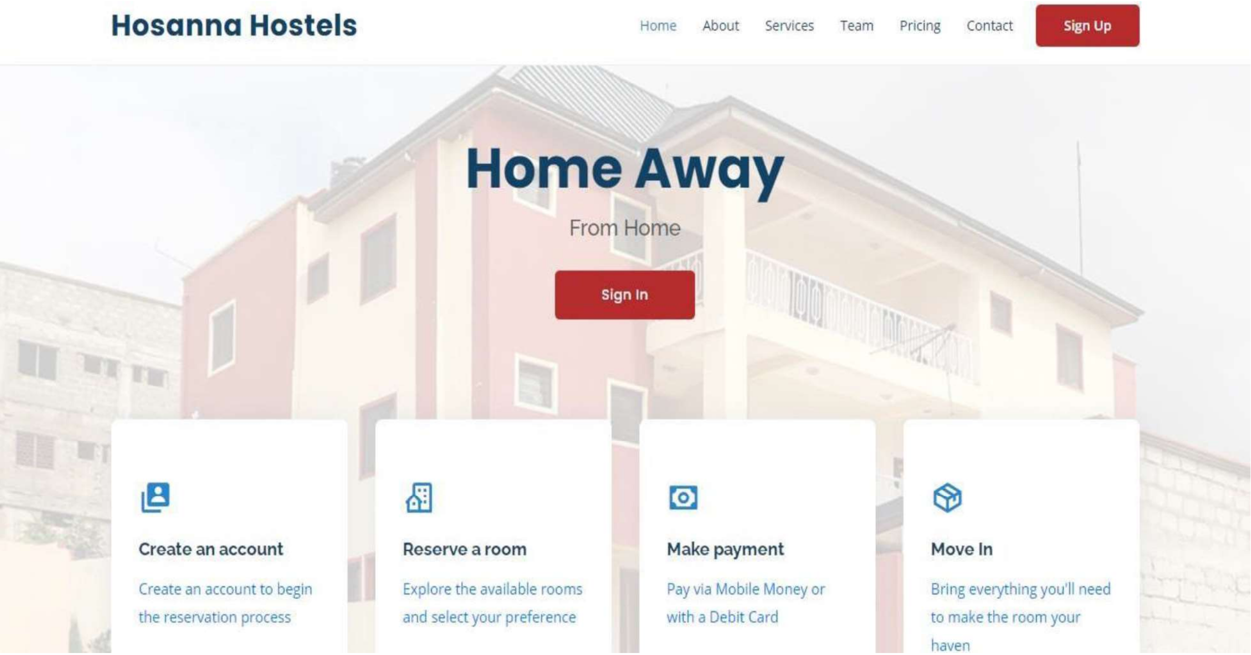


	Table	Action	Rows	Type	Collation	Size	Overhead
☐	admin	Browse Structure Search Insert Empty Drop	1	InnoDB	latin1_swedish_ci	32.0 KiB	-
☐	announcements	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32.0 KiB	-
☐	blocks	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32.0 KiB	-
☐	bookings	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	64.0 KiB	-
☐	category	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	inventory	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	issues	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	payment	Browse Structure Search Insert Empty Drop	5	InnoDB	latin1_swedish_ci	64.0 KiB	-
☐	rooms	Browse Structure Search Insert Empty Drop	21	InnoDB	latin1_swedish_ci	80.0 KiB	-
☐	students	Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	48.0 KiB	-
	10 tables	Sum	48	InnoDB	utf8mb4_general_ci	432.0 KiB	0 B

Figure 4.17 Database View

4.3 Evidence of Implementation

The system has two main sections; the administrator section and the student section.



Home Away

From Home

Create an account

Create an account to begin the reservation process

Reserve a room

Explore the available rooms and select your preference

Make payment

Pay via Mobile Money or with a Debit Card

Move In

Bring everything you'll need to make the room your haven

Figure 4.18 Index page

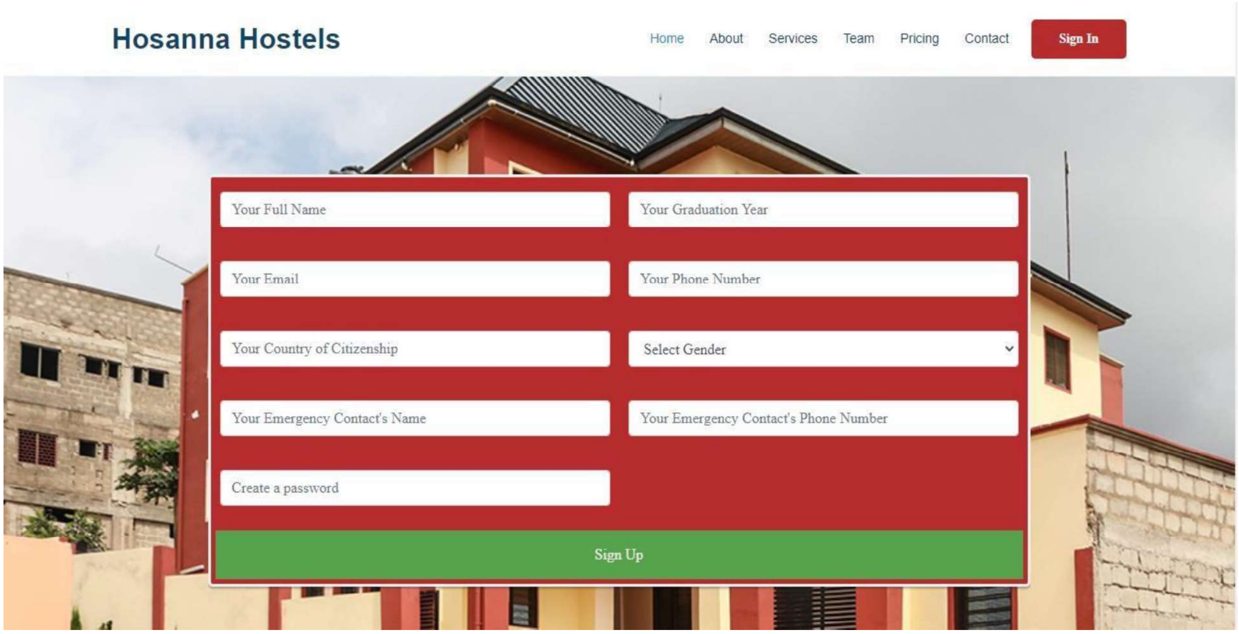


Figure 4.19 Sign Up page

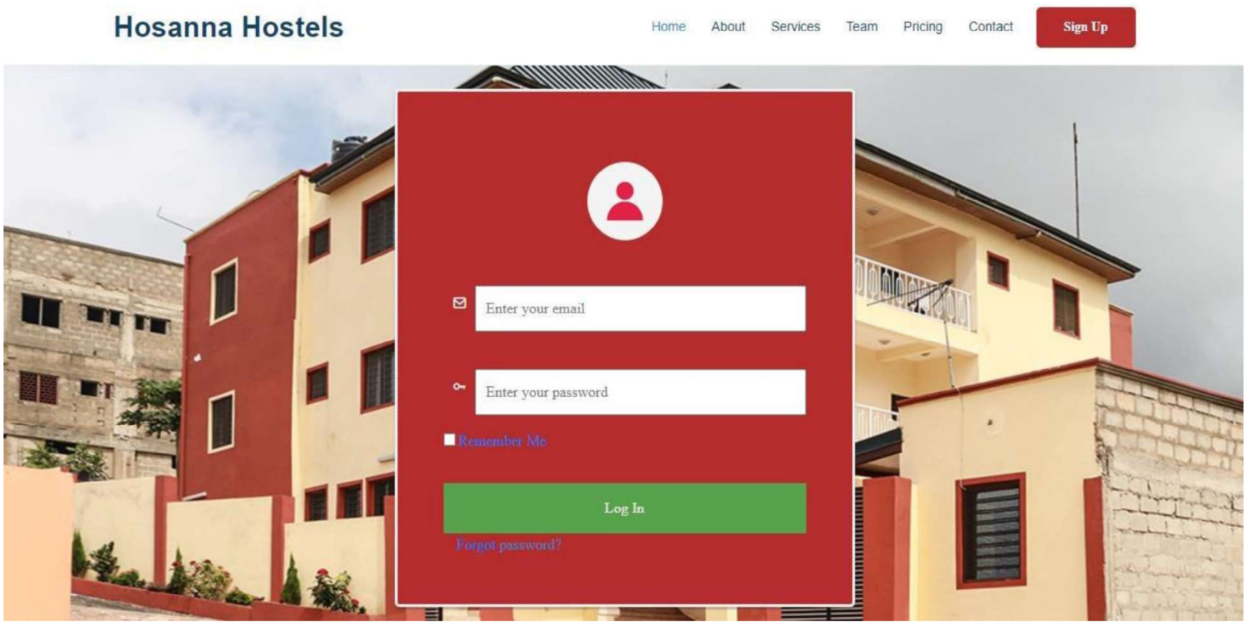
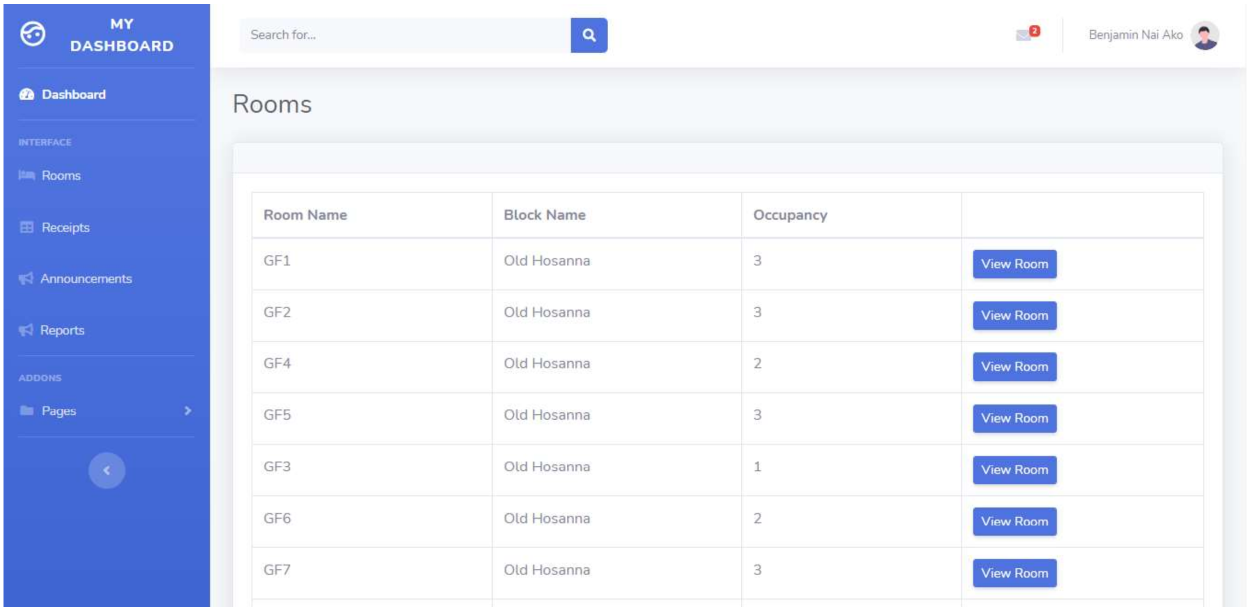


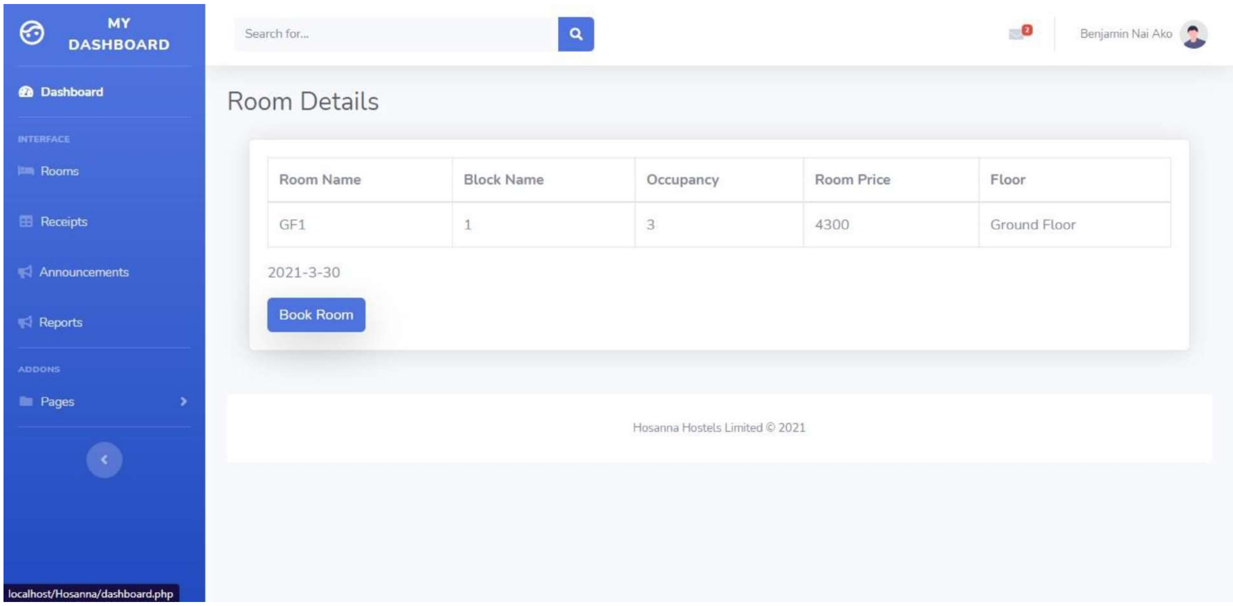
Figure 4.20 Login page



Rooms

Room Name	Block Name	Occupancy	
GF1	Old Hosanna	3	View Room
GF2	Old Hosanna	3	View Room
GF4	Old Hosanna	2	View Room
GF5	Old Hosanna	3	View Room
GF3	Old Hosanna	1	View Room
GF6	Old Hosanna	2	View Room
GF7	Old Hosanna	3	View Room

Figure 4.21 Rooms display page



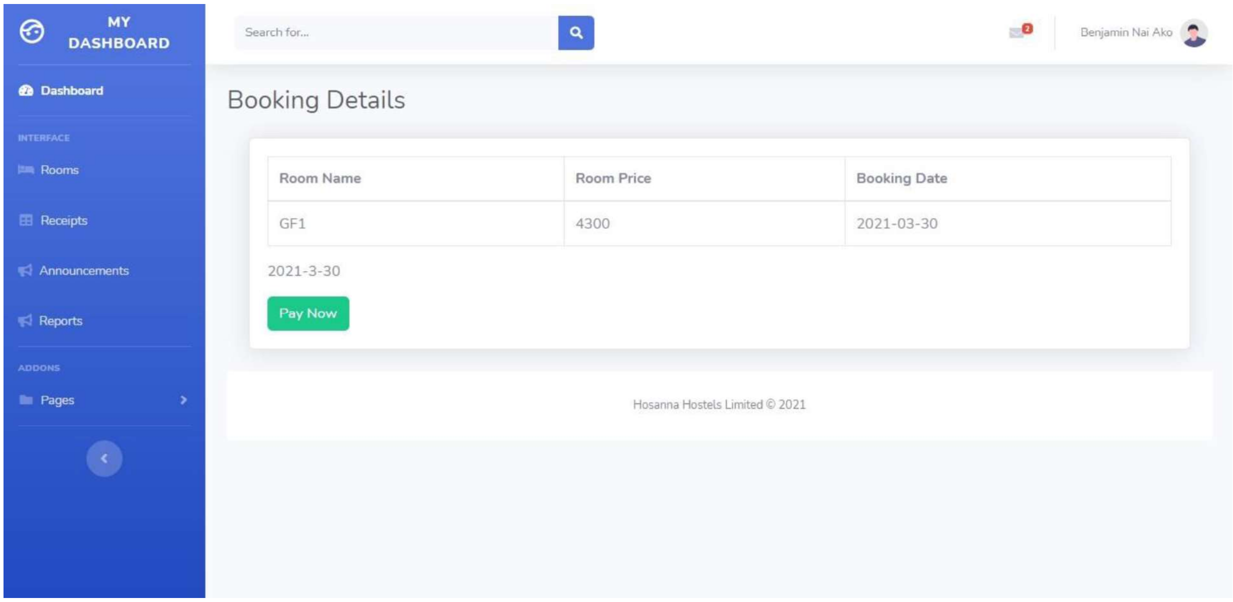
Room Details

Room Name	Block Name	Occupancy	Room Price	Floor
GF1	1	3	4300	Ground Floor

2021-3-30

Hosanna Hostels Limited © 2021

Figure 4.22 Room details page



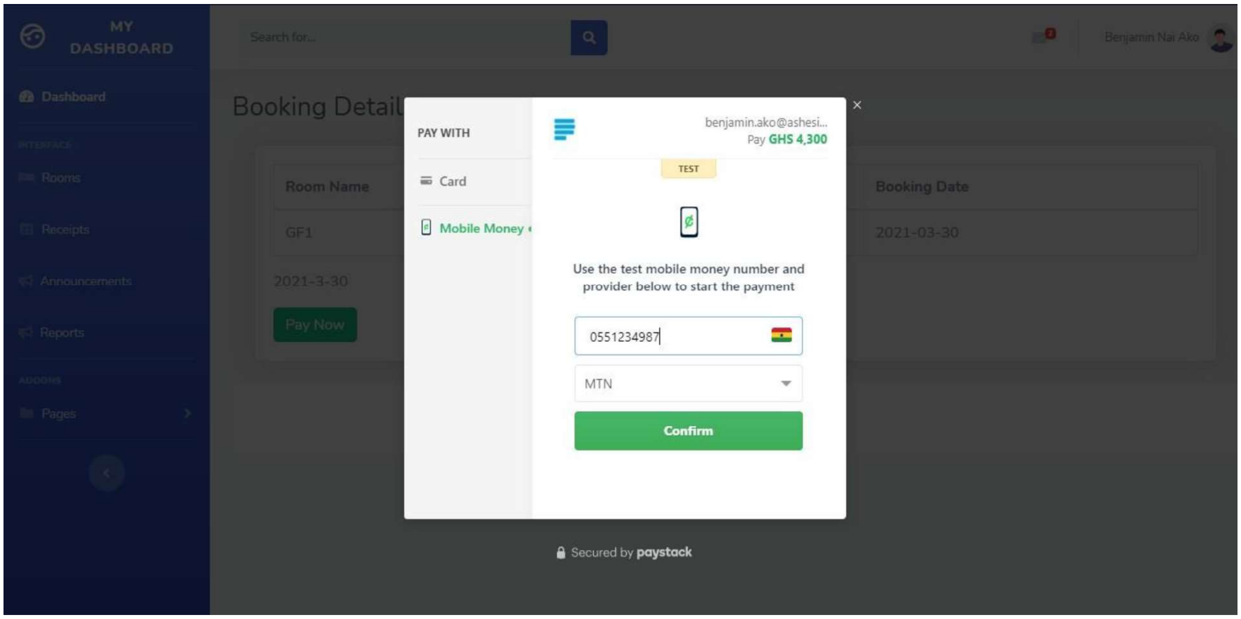
Booking Details

Room Name	Room Price	Booking Date
GF1	4300	2021-03-30

2021-3-30

Hosanna Hostels Limited © 2021

Figure 4.23 Bill page



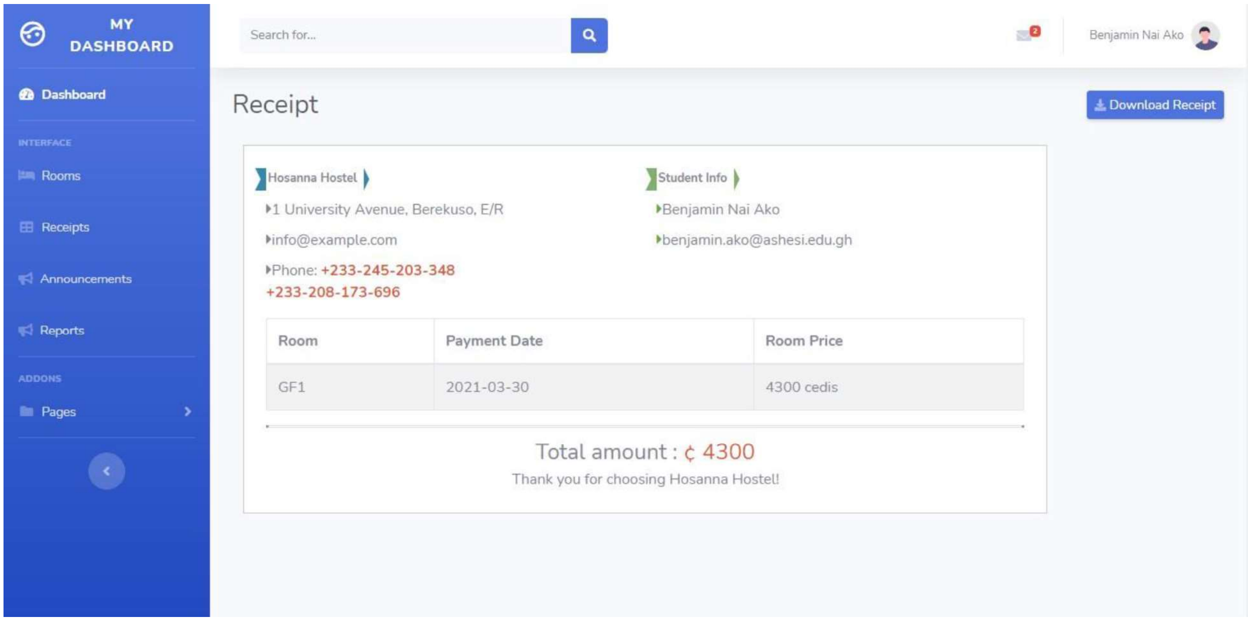
benjamin.ako@ashesi...
Pay GHS 4,300

Use the test mobile money number and provider below to start the payment

0551234987

MTN

Figure 4.24 Payment view



Receipt

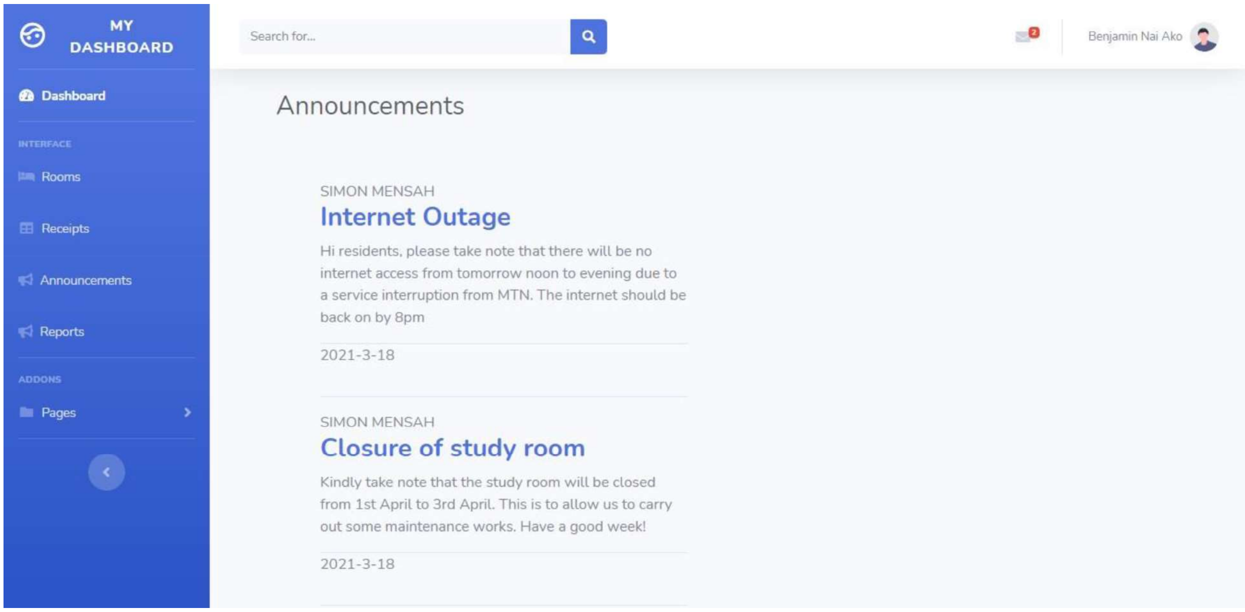
Hosanna Hostel
1 University Avenue, Berekuso, E/R
info@example.com
Phone: +233-245-203-348
+233-208-173-696

Student Info
Benjamin Nai Ako
benjamin.ako@ashesi.edu.gh

Room	Payment Date	Room Price
GF1	2021-03-30	4300 cedis

Total amount : ₵ 4300
Thank you for choosing Hosanna Hostel!

Figure 4.25 Receipt page



Announcements

SIMON MENSAH

Internet Outage

Hi residents, please take note that there will be no internet access from tomorrow noon to evening due to a service interruption from MTN. The internet should be back on by 8pm

2021-3-18

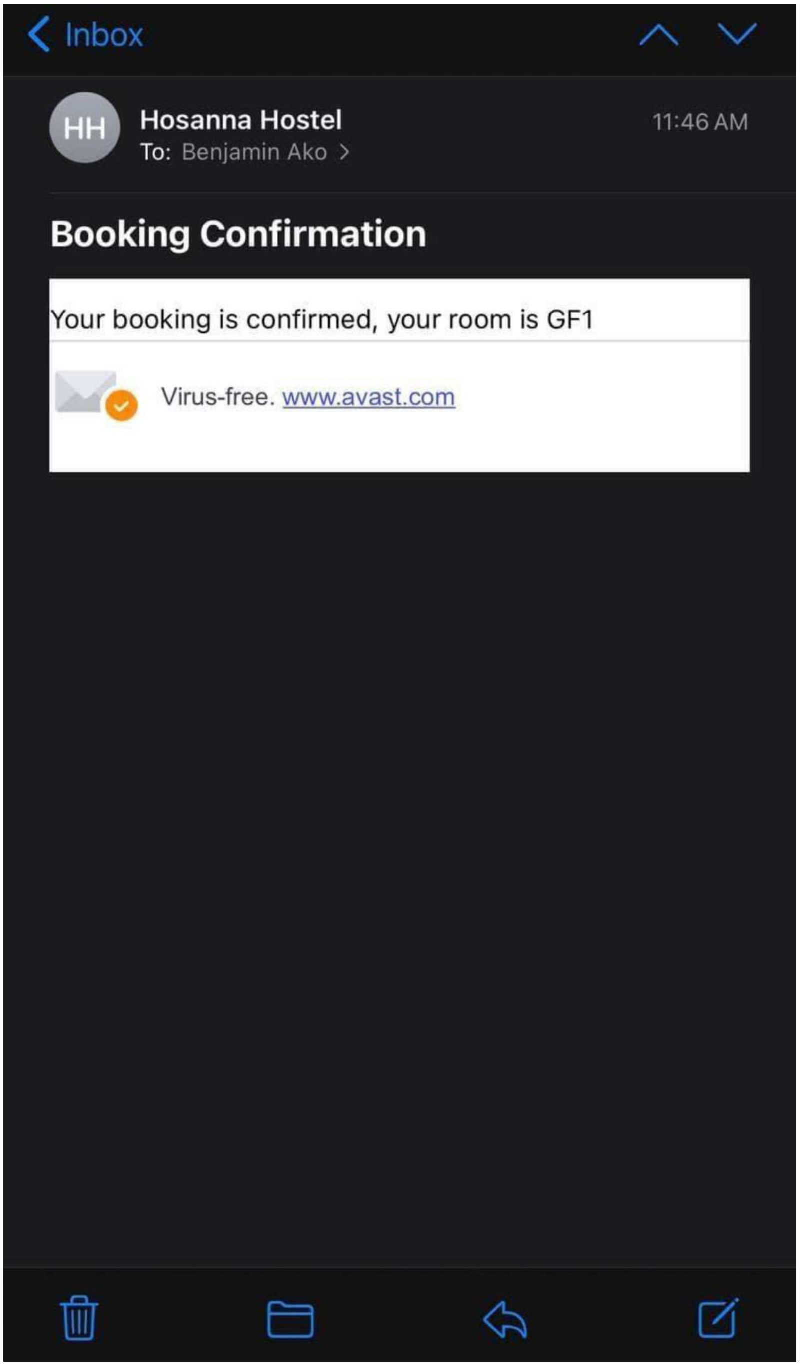
SIMON MENSAH

Closure of study room

Kindly take note that the study room will be closed from 1st April to 3rd April. This is to allow us to carry out some maintenance works. Have a good week!

2021-3-18

Figure 4.26 Announcements page



HH

Hosanna Hostel

To: Benjamin Ako

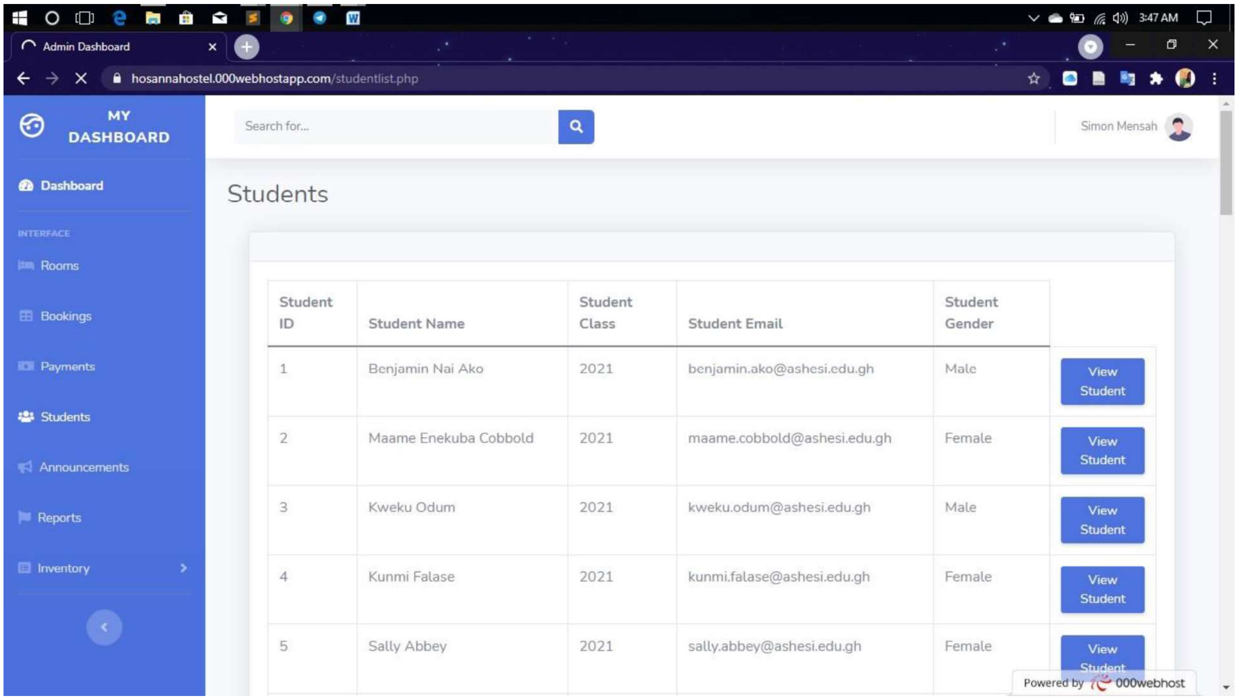
11:46 AM

Booking Confirmation

Your booking is confirmed, your room is GF1

Virus-free. www.avast.com

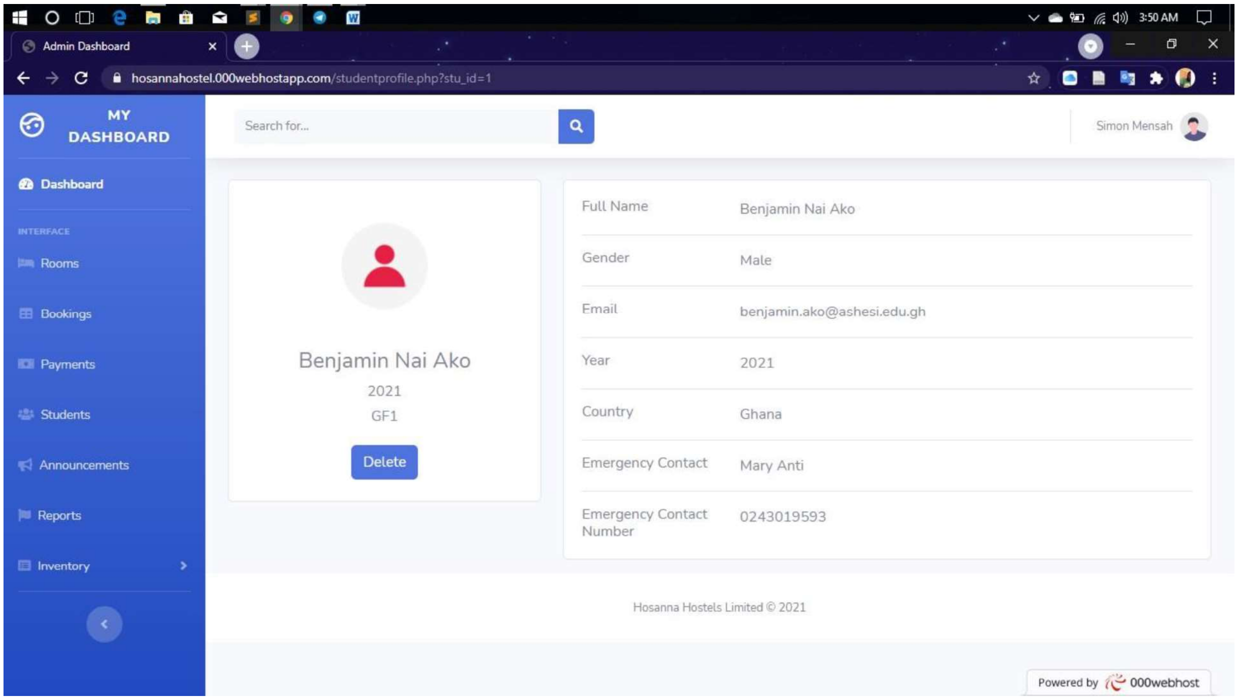
Figure 4.27 Booking confirmation



Students

Student ID	Student Name	Student Class	Student Email	Student Gender	
1	Benjamin Nai Ako	2021	benjamin.ako@ashesi.edu.gh	Male	View Student
2	Maame Enekuba Cobbold	2021	maame.cobbold@ashesi.edu.gh	Female	View Student
3	Kweku Odum	2021	kweku.odum@ashesi.edu.gh	Male	View Student
4	Kunmi Falase	2021	kunmi.falase@ashesi.edu.gh	Female	View Student
5	Sally Abbey	2021	sally.abbey@ashesi.edu.gh	Female	View Student

Figure 4.28 Students list

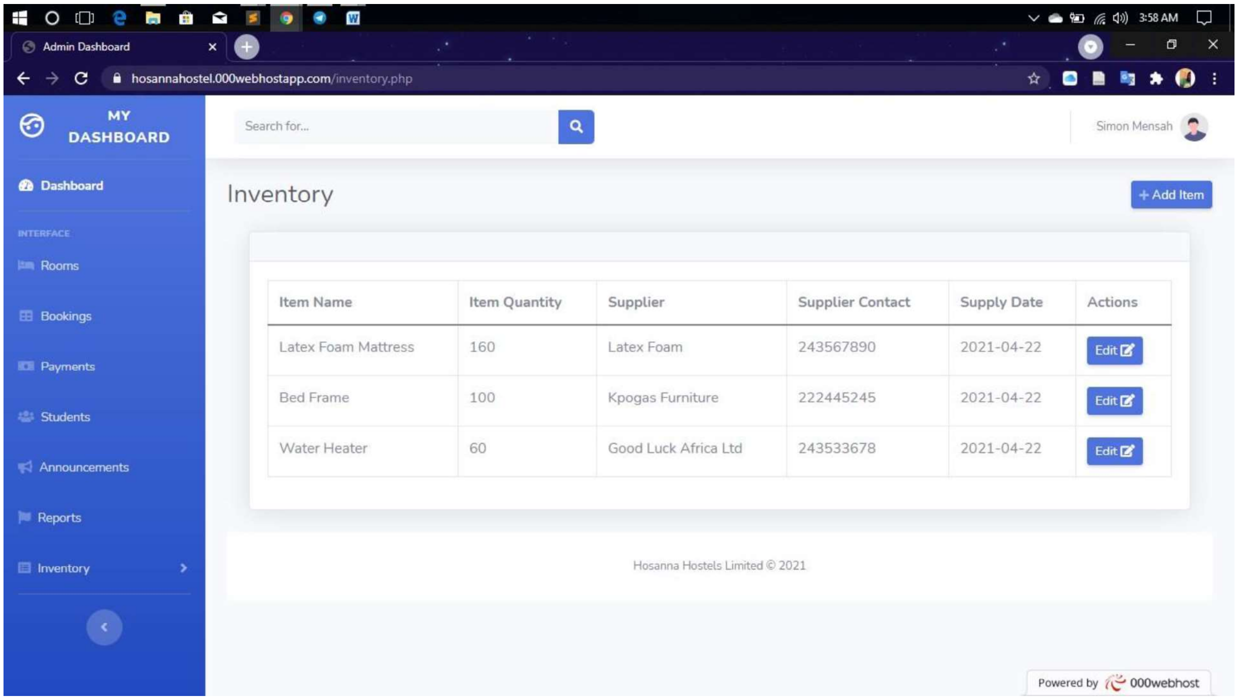


Benjamin Nai Ako
2021
GF1

Full Name	Benjamin Nai Ako
Gender	Male
Email	benjamin.ako@ashesi.edu.gh
Year	2021
Country	Ghana
Emergency Contact	Mary Anti
Emergency Contact Number	0243019593

Hosanna Hostels Limited © 2021

Figure 4.29 Single student view



Inventory

Item Name	Item Quantity	Supplier	Supplier Contact	Supply Date	Actions
Latex Foam Mattress	160	Latex Foam	243567890	2021-04-22	Edit
Bed Frame	100	Kpogas Furniture	222445245	2021-04-22	Edit
Water Heater	60	Good Luck Africa Ltd	243533678	2021-04-22	Edit

Figure 4.30 Inventory page

Chapter 5: Testing and Results

This chapter discusses the various types of testing done, how and why the system was tested, and the tools and techniques used for the testing.

5.1 User Testing

The final stage of the development process was testing. Users tested the system to find their level of satisfaction with the system and if it meets all the requirements (functional and non-functional). To facilitate the user testing, I hosted the system on a live server. A link to the site was then shared with some Ashesi students to interact with the system, try booking rooms, and use other functionalities. A short questionnaire was also shared to get students' feedback when they're done going through the site. A total of 73 students tested the system. The number of respondents was not very much because not everyone who got the link and questionnaire participated in the testing. Getting feedback from students, as mentioned earlier, was difficult because of the global pandemic limiting human interaction.

After testing the system, 96% of the users (70 out of 73) found the system to be easy to use and gave it an overall rating of 8.3 out of 10. 64% of the users responded that they would choose to move to Hosanna Hostel because of the new booking system. There were suggestions to include certain features to the system such as making students see rooms based on their gender, view pictures of the room, view a floor plan to see where exactly their rooms are, and see the names of their roommates and half payment of the hostel fees. Based on this feedback, I implemented some of these suggestions and forwarded the one that was beyond my control to the hostel administration (half payment of hostel fees). The hostel administrator was also sent a link to test the system. He was impressed with the system and also suggested including pictures of the rooms. Generally, all stakeholders were happy with the system and gave good constructive feedback.

5.2 Unit Testing

"Unit testing is carried out to determine if the individual units of the system work as they are expected to" [2]. This testing was done in the early phases of the development process.

- **Account creation**

  Pre-condition: The form takes the user's data

  Expected Results: Form successfully grabs user data and inserts it into the database

Table 5.1 Test for account creation unit

Valid Input	**Result**
• Submit account creation data	Data is inserted into the database, and the user is redirected to the login page
Invalid Input	**Result**
• Data already exists in the database	The user is alerted and sent back to the signup page

- **Logging In**

  Pre-condition: The form takes the user's login details

  Expected Results: Form successfully grabs user's login details

Table 5.2 Test for login unit

Valid Input	Result
- Submit login details	The user gets access into the system and is sent to their dashboard. Successful
Invalid Input	**Result**
- Submit wrong details	The user is alerted of the error and sent back to the login page. Unsuccessful

- **Selecting a room**

  Pre-condition: Some room details are collected

  Expected results: Details are collected and displayed to the user

Table 5.3 Test for room selection unit

Valid Input	Result
- Select room	Full room details are displayed to the user. Successful
Invalid Input	**Result**
- No room is selected	No details are displayed. Unsuccessful

- **Posting announcements**

  Pre-condition: Announcement details are collected

  Expected results: Details are inserted into the database and displayed to students

Table 5.4 Test for announcements unit

Valid Input	Result
• Submit announcement details	Details are inserted into the database and displayed to students on the announcements page. Successful
Invalid Input	**Result**
• Submit announcement form with empty fields	Admin is alerted and redirected to the announcement form. Unsuccessful

- **Creating rooms**

  Pre-condition: Fill the form to collect room details

  Expected result: Room details are inserted into the database

Table 5.5 Test for room creation unit

Valid Input	Result
• Submit new room details	Details are inserted into the database, and a new room is created.

Invalid Input	Result
• Submit details of an already existing room	The administrator is alerted that the room already exists and is redirected to the form.

- **Reporting Issues**

  Pre-condition: Details of the issue-reporting form are collected

  Expected result: Details are inserted into the database and displayed to the administrator

Table 5.6 Test for issue reporting unit

Valid Input	Result
• Submit details of the issue	The new issue is inserted into the database and displayed to the administrator. Successful
Invalid Input	**Result**
• Submit the form with empty fields	The student is alerted of the empty fields and redirected to the form. Unsuccessful

5.3 Component Testing

Component testing is carried out to determine if the various units can work together as they should. The multiple units combine to form the components/modules of the system.

- **Room Booking**

The booking component includes room viewing, room selection and booking.

Pre-condition: Details of the room selected is displayed to the user, and the details are inserted into the database along with the user's id and date.

Expected Results: The booking details are inserted into the database and displayed to the user.

Table 5.7 Test for room booking module

Valid Input	**Result**
• Rooms are viewed • Preferred room is selected	Room details, user id and date are inserted into the booking table. Successful
Invalid Input	**Result**
• A filled room is selected	The user is informed of the status of the room, and no details are entered into the database. Unsuccessful

- **Payment**

  The payment component includes viewing details of the room booked for, payment and viewing receipt.

  Pre-condition: Details about the booked room is displayed to the user, they are directed to the payment page, and the payment details are inserted into the database.

Expected Results: Payment details are inserted into the database, and a receipt is displayed to the user.

Table 5.8 Test for payment module

Valid Input	**Result**
• Submit payment details	Details are inserted into the database, and a receipt is displayed to the user. Successful
Invalid Input	**Result**
• Wrong details submitted	Details are not inserted into the database, and the user is informed of a failed transaction. Unsuccessful

5.4 System Testing

"System testing is carried out to check if all the components integrate successfully to work as a single functional system" [3]. The objectives of this testing include component integrations, thorough testing of each input and checking the user's overall experience with the system. The techniques used to accomplish this testing include usability, performance, functional and load testing.

- **Usability Testing**

  This testing is done to check the ease of use of the system. "*Usability testing focuses on the system's ability to meet its objectives and the users' ease to use the system*" [4]. Ease of use is one of the system's non-functional requirements and thus required this testing.

Labels were used very well in the interface to enable user's know what they are clicking on. A side and top navigation bar were also included in the interface to navigate the pages easily. A link to the system was shared with some Ashesi students; 70 out of the 73 testers found the system easy to use.

- **Performance Testing**

  This testing is carried out to ensure the system satisfies all the non-functional requirements; security, availability, accessibility and reliability. Testers' passwords were hashed, payments were made securely, the site was accessible on different screen types, the site showed up whenever its URL was entered and always displayed the correct data to testers.

- **Functional Testing**

  "*Functional testing is done to check if there are any missing functionalities. It is also carried out to check if the system's functionalities work as they are supposed to*" [5]. All the system's functionalities worked as expected when tested by testers. The functionalities that depended on APIs also worked perfectly; the payment portal loaded and handled transactions without error, and mails were sent immediately after booking a room.

- **Load Testing**

  "*Load testing is done to check if the system will be able to perform under real-life loads and pressure*" [6]. Ten students tested the system at the same time. They created accounts, booked rooms, made payment and reported issues to the hostel administrator. None of them complained of the system responding slowly or having any challenges.

Chapter 6: Conclusions and Recommendations

This chapter includes an overview of the system, how the system met requirements, recommendations, limitations and a conclusion.

6.1 Summary of Project

Hosanna hostel's inefficient booking process and other administrative problems necessitated the development of this management system. The system is made up of a booking system, incident and inventory management system. The requirements to guide the development process were divided into functional and non-functional. Ease of use, accessibility, availability and reliability are the non-functional requirements of the system. The main modules of the system include account creation & login, booking, payment and issue reporting. The system was tested using unit testing, component testing and system testing. APIs were included in the system to handle certain features such as payment and emailing. PayStack's API was integrated to handle the payment, and PHP Mailer was integrated to handle confirmation emails. Given the requirements, the system solves the problems of the hostel.

6.2 Recommendations

- **Mobile Application**

  A mobile application will be a great alternative to this developed system. The system is a web application that makes it accessible on different platforms, provided there is internet connectivity. All aspects of the system are handled online. To improve the usability and accessibility of the system, a mobile application version can be developed. Ideally, the mobile application will be hybrid.

- **Inclusion of Other Hostels**

Only Hosanna Hostel does not face the problem being solved by the system but all the off-campus hostels. The scope of the application can be revised to include making the system such that it caters for all off-campus hostels. The system will then be the main point of booking for all off-campus hostels for Ashesi students, and these hostels will use the other features to manage their administrative processes.

6.3 Limitations

- One major limitation was integrating APIs during the development process. This was my first time working with some of the APIs, and they posed quite a challenge. It took various hours and contributions from other people to get some of the APIs functional.
- Another limitation was getting the payment API I wanted to use initially. The provider requested certain documents that I did not have access to. I thus had to look for another payment service and learn how to use their API.
- Getting feedback from stakeholders was another limitation. Because I could not engage with stakeholders physically, getting them to answer questionnaires and give feedback was problematic. This also affected user testing because very few people were using the links I sent them to test the system and provide me with feedback.

6.4 Conclusion

The hostel management system provides a solution to Hosanna Hostels inefficient booking, inventory and incident management processes. It was developed with input from stakeholders (Ashesi students and management of Hosanna Hostel). Requirements were gathered from stakeholders and analyzed to produce a requirement statement, functional and non-functional requirements. The requirements guided the development, and the solution was tested to ensure it meets all the requirements. The system includes a booking feature, incident reporting feature and an inventory management feature. Students no longer have to face difficulties booking a room in Hosanna Hostel, and the management can execute their duties effectively due to the computerized hostel management system.

References

[1] Code Academy. What is an IDE? Retrieved November 17, 2020, from https://www.codeacademy.com/articles/what-is-an-ide

[2] Unit Testing Tutorial: What is, Types, Tools, and Example. Retrieved April 10, 2021, from https://www.guru99.com/unit-testing-guide.html

[3] System Testing Tutorial: Types & Definitions with Example. Retrieved April 10, 2021, from https://www.guru99.com/systemtesting.html

[4] Usability Testing Tutorial: What is Usability Testing? (UX) Testing Examples Retrieved April 10, 2021, from https://www.guru99.com/usability-testing-tutorial.html

[5] Functional Testing Tutorial: What is Functional Testing? Types & Examples (Complete Tutorial) Retrieved April 10, 2021, from https://www.guru99.com/functional-testing.html

[6] Load Testing Tutorial: What is? How To? (With Examples) Retrieved April 10, 2021, from https://www.guru99.com/load-testing-tutorial.html

[7] Grace L. Intal, Rex A. Robielos and Alysia G. Ortega. 2018. Design of User-Driven Facility Management System for Universities. *International Conference on Information and Education Technology* (January 2018), 170 - 176. DOI: https://doi.org/10.1145/3178158.3178211

[8] 3-Tier Architecture: A Complete Overview. JReport. Retrieved October 8 2020, from https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-completeoverview/

[9] Introduction to HTML. Received November 18, 2020, from https://www.w3schools.com/html/html_intro.asp

[10] JavaScript basics. MDN Web Docs. Retrieved November 18, 2020, from https:// developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web /JavaScript_basics

[11] UML Use Case Diagram Tutorial. Lucidchart. Received November 20, 2020, from https://www.lucidchart.com/pages/uml-use-case-diagram

[12] Tanmay Nandanwar, Priyanka Bahutule and Rajevita Buddala. 2020. A Study on Shift Towards Digitization of Hostel Room Allotment for a University. *In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, February 24 – 25, 2020, Vellore, India. https://doi.org/10.1109/ic-ETITE47903.2020.117

[13] Philip Owusu-Afriyie. 2020. Capstone Management System. (October 2020). Retrieved from https://air.ashesi.edu.gh/bitstream/handle/20.500.11988/595

[14] PHPMailer/PHPMailer. *GitHub*. Retrieved November 18, 2020, from https://github.com/PHPMailer/PHPMailer

[15] Matt Wyatt. What is an API? A Digestible Definition with API Examples for Ecommerce Owners. Retrieved November 18, 2020, from https://www.bigcommerce.com/blog/what-is-an-api/#what-is-an-api

Appendix

Questions used in requirements gathering.

1. What year group are you in?
2. During the semester, do you live on-campus or off-campus?
3. Is there a particular reason for your choice of accommodation? What is it?
4. If you live off-campus, what hostel do you live in?
5. Have you ever lived in Hosanna Hostel?
6. If yes, how did you book the room?
7. What challenges did you face from the booking period to the move-in day?
8. How will you rate your experience with Hosanna Hostel (with respect to booking, reporting problems and general living)? (out of 10)
9. How will you describe your ideal booking process?
10. Have you used an online booking system before?
11. If yes, what features stood out for you in the system?

Questions used in user/usability testing.

1. How easy was it to use the system?
2. What features do you think can be included to make the system and booking process better?
3. How will you rate the system? (out of 10)
4. If this system were implemented, will it affect your decision on what hostel to stay in next academic year? (Will you choose to stay in Hosanna instead of another hostel because of this booking system?)