**DESIGN AND IMPLEMENTATION OF HOSTEL INFORMATION SYSTEM**

**BY**

**OGBUANU OGOCHUKWU VIVIAN**

**U15/NAS/CSC/098**

**DEPARTMENT OF COMPUTER SCIENCE / MATHEMATICS**

**FACULTY OF NATURAL AND APPLIED SCIENCES**

**GODFREY OKOYE UNIVERSITY, ENUGU**

**IN PARTIAL FULFILLMENT OF THE REQUIERMENT FOR THE DEGREE OF BACHELOR IN SCIENCE**

**SUPERVISOR**

**MR. KANIFE MICHEAL**

**JULY, 2018**

**CERTIFICATION**

I hereby declare that this work is being done by me Ogbuanu Ogochukwu with some research being carried out both in library. Should I be convicted of having cheated in the work, I shall accept verdict of the university.

 ……………………… ………………………….

Ogbuanu Ogochukwu Date

 **APPROVAL**

This is to certify that the research work design and implementation of hostel information system by Ogbuanu Ogochukwu Vivian in the department of Computer Science and Mathematics has been examined by faculty of Natural and Applied Sciences.

…………………………… ………….………………

Mr. Kanife Michael Date

(Supervisor)

……………………….. ………………………..

Dr. Mrs. Agu Monica Date

(Head of Department)

**DEDICATION**

This Project is humbly and sincerely dedicated to the Almighty God, who is the architect of my staying in this school. May the glory honor and adoration be unto Thee. Amen.

**ACKNOWLEDGEMENTS**

I am grateful to my HOD Mrs. Monica Agu, my Supervisor Mr. MC Kanife and all the lecturers in computer science department who contributed to the successful writing of this project work.

I also appreciate all the Friends that I consulted, authors I used their materials.

But pray strongly that Almighty God to guide, protect and reward them maximally.

**ABSTRACT**

The growing number of students in higher institutions poses a lot of accommodation problems for students and School Management. Some of these problems include, few hostel not properly managed, Statistics of rooms required to match the growing number of student are farfetched. Also hostel administrators cannot give accurate information of the occupancy of a particular room. In View of all these, the solution proffered by this project is to have an automated hostel allocation system that not only gives student the opportunity to select their hostel of choice, but also provides the hostel administrators necessary functionalities to adequately manage the accommodation process in the school. To be able to achieve this, proper research was done on the already existing solution in the school, which is manual, and also on automated solutions deployed in other institutions like the University of Nigeria Nsukka. It was realized that for such a system to work effectively there will be need for certain functionalities to be implemented. First, there should be avenue to add new hostel, block, room or bunk space. Second, the system should be able to be fed with the records of students that have duly paid their school fees. Third, students who have been uploaded should be able to get hostel on visiting the system. And lastly, the hostel administrators should be able to download records showing the status of the hostels indicating those that have been occupied and those that have not. Putting all these in consideration the system was designed using the V-Model OOAD methodology and implemented as a web based solution, for ease of access, with web forms using Microsoft ASP.NET C# and SQL Server as the database backend.

**TABLE OF CONTENTS**

Title Page i

Certification ii

Approval iii

Dedication iv

Acknowledgement v

Abstract vi

Table of Contents vii

List of Figures ix

List of Tables x

**Chapter 1: Introduction**

1. Back ground of study 1

 1.1 Statement of the problem 2

 1.2 Objectives 3 1.3 Significance of the Project 3

**Chapter 2: Literature review**

1. Introduction 4

2.1 Theoretical Background 4

2.2 Review of Related Literature 9

2.3 Summary 12

**Chapter 3: System Analysis and Design**

1. Introduction 13

3.1 Methodology 13

3.2 Describe the existing system 14

3.3 Analysis of the proposed system 15

3.4 Design of the proposed system 18 **Chapter 4: System Implementation**

4.0 Introduction 22

4.1 Choice of development Environment 22

4.2 Implementation Architecture 26

4.3 Software Testing 27

4.4 Documentation 31

4.4.1 User Manual 31

4.4.2 Source code listing 32

**Chapter 5: Summary and Conclusion**

5.0 Summary 32

5.1 Conclusion 32

5.2 Recommendations 32

**REFERENCE** 33

 **APPENDIX**  35

**LIST OF FIGURES**

Fig 2.1: Screen short of a form 6

Fig 3.1 V – Model 14

Fig 3.2: Use case Diagram 16

Fig 4.1: Screenshot of SQL Environment 23

Fig 4.2: Screen shot of Visual studio start page 24

Fig 4.3: Screen shot of Visual studio start page 24

Fig 4.4: Screen shot Implementation Environment 25

Fig 4.5: Screen shot Implementation Architecture 26

Fig 4.6: Login Screen Shot 27

Fig 4.7: Admin’s Dashboard Screen Shot 28

Fig 4.8: Add Hostel Screen Shot 28

Fig 4.9: UploadPayment Details Screen Shot 29

Fig 4.10 Get Hostel Screen Shot 29

Fig 4.11: Acknowledgement Slip 30

**LIST OF TABLES**

Table 2.1: Screen short of uploaded students 8

Table 3.1: Hostel Name 18

Table 3.2: Hostel Block 18

Table 3.3: Hostel Room 19

Table 3.4: Hostel Bunk Space 19

Table 3.5: Hostel Payment Details 19

 **CHAPTER ONE**

 **INTRODUCTION**

1. **BACKGROUND OF STUDY**

Computer as indispensible tools in the total life of human endeavor has gained precedence in the world especially in the development growth of the Nigeria society. Computers are super machines that convert physical/mental effort into a mere easy life. The use of computer in our everyday life cannot be over emphasized. This is because of the gradual increase in human population and complexity of life, which has equally brought about increase in volumes of data being generated every day.

The gradual increase in human population has equally affected the education sector. Thus higher institutions in the country today are currently faced with the challenges of hostels/halls allocation to students. The fact remain that the number of students usually out numbers the available accommodation.

It is safe to say that most activities such as hostel allocation that are carried out in most universities in Nigeria are done manually. Therefore, there is a lot of strain on the individuals running the hostel allocation. An E-hostel system is simply software developed for managing most activities that take place in the hostels with the help of a hostel administrator [1].

The goal of this research work is to provide a solution to the problem of hostel management, by designing a computerized system which is user-friendly and GUI-oriented that will be compatible with the existing manual system. The system will solve the problem of hostel management; thus helping to reduce problems associated with the manual hostel management system. The system can only be accessed by students who have paid school fees for the current semester. The project uses Godfrey Okoye University thinker’s corner Enugu as case study.

* 1. **STATEMENT OF THE PROBLEM**

Students at the beginning of each session waste half of the semester looking for accommodation.

1. The few hostels that exist in the universities are not properly managed, since they are using manual system, which is paper-based. The possibility of losing the data is very high, Lots of paper are required to store all the information, data could be misplaced due to human error or in the event of a break in, data could be stolen very easily;
2. It usually takes time to retrieve data or information that has been stored, because the workers need to find the actual folder where the data was stored previously.
3. Most often students pay for hostel fee and end up not getting one due to lack of bed spaces when there are enough space.
4. Hostel administrators cannot give accurate information of the occupancy of a particular room.

These and many more form the statement of the problem that necessitated this research work.

* 1. **OBJECTIVES OF THE STUDY**

The project is designed to help in the management of hostel allocation in the University. The main objective of the system includes:

1. To create and develop a central database system that will contain information on all the available rooms in the hostel.
2. To add new existing hostels and rooms to the system.
3. To print out the evidence of hostel space being allocated to a student.
4. To generate reports on hostel occupancy.
5. To ensure that any registered and eligible students are allocated into hostel.

**1.3 SIGNIFICANCE OF THE PROJECT**

The new system designed for computer driven student’s hostel allocation will among other things:

1. The system will enable the students get their hostel room before the lectures will commence every semester.
2. The system helps to know the particular student occupying bunk space each semester.
3. The system will enable to improve in the condition of how the hostel is being managed.
4. It gives the actual number of the students in the hostel statistically.
5. Store all student information and enables the management team to retrieve back the information that have been stored and to maintain all activities in managing a hostel.

 **CHAPTER TWO**

 **LITERATURE REVIEW**

**2.0 INTRODUCTION**

Thischapter covers the theoretical background of the work and the technologies used in the development of the web-based Hostel Allocation software with these programming languages C#, HTML, JavaScript, CSS and SQL Server.

**2.1 Theoretical Background**

This project was implemented as a web based solution to be accessed by any valid web client, for examples a browser. The technology stack used is html, css and JavaScript for the front end and ASP .Net web forms for the backend. The database system was implemented using Microsoft SQL Server. For a quick and efficient design process Microsoft Visual Studio was used as an IDE.

**HTML elements** are written with a **start** tag, an **end** tag, and the **content** in between for instance; in student end Get Hostel Room is written in html tag

**<h3>Get Hostel Room</h3>**

The HTML **element** is everything from the start tag to the end tag.

<H1>means start tag and </H1>closing tag.

Some major HTML features here are the form and CSS.

**CSS**, meaning Cascading Style Sheet, is a style sheet language that describes the presentation of an HTML. It also describes how elements must be rendered on screen, on paper, or in other media. It is a cornerstone technology for World Wide Web, alongside HTML and JavaScript. Each CSS declarations end with semi-colon, the property and the values are separated with colon. CSS is used in beautification of this application by giving them color, size of the text and so. The application of CSS to the HTML document depends on the particular tag selected to style. Below is the sample of CSS rule:

 Selector Declaration Declaration Declaration

{Font-size: 20px; color: orange; background-color: brown ;}

body

 Property value Property value Property value

And this

body {
    background-color: #d0e4fe;
}

h1 {
    color: orange;
    text-align: center;
}

p {

Font-family: "Times New Roman";

Font-size: 20px;
}

The above CSS sample shows how HTML Selectors are being style.

HTML Form is used to collect information in a web application for example registration number and receipt number for students or username and password for staff then login that will take your input to the system for eligibility and accessibility.

Form is rendered on browser as below



**FIG 2.1: Screen short of a form**

 ASP. Net web form is a web framework that enhances the development of web applications. It abstracts out the difficult part of building an application from scratch there by leaving the developer to only worry about the business logic of the application. Every web forms application has a view file, ending with ".aspx", that defines the user interface (UI) of the application and either have a code file, ending in ".aspx.cs", or has the code embedded into the view file. When the code file is separate it is called the "code behind". This contains all the logics of your web forms application.

A web form has the concept of controls. These are objects that may or may not have visual representation whose purpose is to act as the logical building block for the overall application. Each control in the application can be referenced in the code-behind by the IDs given to them at the time of creation. With visual studio it is easy to drag and drop controls on to a page there by facilitating rapid application development.

 For controls with visual representation in like <asp: Button/> the application renders them in the front end with the equivalent html tags that make them up. For example a web forms button control, <asp: Button/>, will render as <input type="submit"/>.

**SQL** Structured Query Language is a standard language for accessing and manipulation of data in our databases. SQL Server is used here to insert and retrieve data from database.

 In **SQL** server, information are fetched from the login page into the database to check if the person is actual a student whether they have paid school fees or Administrator whether you have access to the database.

As a student your details should be uploaded into database in excel sheet format like;



 **Table 2.1: Screen short of uploaded students**

When you login as a student in the aboveform in figure 2.1 with REG\_NO and RECIEPT\_NO it will direct you to GET HOTEL, block, room and bunk space at the end of process you get your acknowledgement slip.

**2.2 Review of the Related Literature**

Hostelling has come a long way since the German school teacher Richard Schirrmann began the youth hostel movement in 1909. He saw the need for overnight accommodation that would allow school children the chance to travel and experience other parts of the country safely and affordably. As a result, the world’s first Jugendherberge (youth hostel) was opened in 1912 in the beautiful Altena Castle, located on the Lenne river valley, western Germany.

The aim of Richard in that year was to promote the education of all young people of all nations, but especially young people who do or don’t have particular ambition, this help them to merge with other people to get inspiration and to focus with their career. It also encourages in them a greater knowledge that will see them in future, and also an appreciation of the cultural values of the towns and cities in all parts of the world [2].

The development of hostel management system is to give the particular organization a well improved system that would make it possible for them to confidently save and keep information safe and secure. Anderson claims that a Management Information System (MIS) is a system of processing data in an organization to collect information, which is later communicated to various departments to facilitate solid decision making. Since the organization is still using the manual paper-based system, it is very good plan to the organization to change from manual paper-based system to computerized system. The advantage of use computerized or automated systems is user have the ability to make quick, well-timed and informed decisions.

Therefore, the main goal of creating this hostel management system is to facilitate both student and management team for the dealing process. In order to meet my goal [3].

A **hostel** is a budget-oriented, shared-room ("dormitory") accommodation that accepts individual travelers or groups for short-term stays, and that provides common areas and communal facilities. To be considered a **hostel**, the property must provide short-term, shared (dormitory-style) accommodation for individual travelers, though many hostels also provide private rooms. The word "dormitory" refers to a room where travelers independently book individual beds in a shared room as opposed to booking entire rooms like in a hotel or guesthouse. In some countries in the world **hostel** can also refer to student accommodation or long-term accommodation for drug addicts or the homeless. [4]

Hostel allocation is a software solution used in schools and higher institution who want to have a huge database available where you can add and access any hostel of your choice, in this software tables were created that contain all the hostels available in the institution with their bunk spaces.

According to University of Nigeria update,

* The University has approximately 5645 bed spaces in total to accommodate over 15,000 students.
* It is practically impossible to accommodate every student desiring accommodation on campus.
* The acute accommodation problem is made worse by the total collapse of Zik’s Flats Hostels which ordinarily will accommodate all the first year students and Mbanefo hostel that would have taken up to 1,064 students.
* The available bed spaces have been rationed as follows:-

 1st Year students     –           40%

 Final year student   –           25%

 Others years            –           35%

* All bed space allocation is strictly **ON-LINE** and on First come First Serve basis.
* Issues resulting from defaulting students (those who did not pay within the 3 days window after generation of invoice) are being resolved [5].

A row has broken out over hostel allocation at the federal University of Agriculture in Makurdi, the Benue state capital. Some students are accusing Bursary Department workers of hoarding the allocation forms. The workers deny the charge, saying the bed spaces were allocated first-come-first-serve basis. Some had complained that they have stood on the queue for four hours and one Paul from the University argue with the management let them emulate Ahumadu Bello University in Zaria, where hostel accommodation is processed online because the process lacks transparency here[6].

 A student pleaded with the National University Commission (NUC), Minister of Education and whoever else that have some authority to restore sanity of the system by developing a computerized system where student will get their hostel accommodation online. Noting that this will benefit especially people in final year that has lots of work to do. Most of them don’t find off-campus attractive. Even some of them who had lived off-campus tend to seek hostel accommodation once they get into their final year. They need all the time they have for their academic work which might be affected by the several distractions that go with living in the town. Effort should be made to first accommodate female students among them without delay. Many of them are going about every evening to find a place to lay their heads and this exposes them to dangers [7].

Students on campuses of tertiary institutions are facing certain level of challenges that draw them not to come back to get their accommodation on time. Some of these students may be physically challenged or due to illness [8].

**2.3 Summary**

It is obvious that the benefits of having hostels should not be lost to the irregularities encountered in hostel allocation. Equally it is clear that there is no way to achieve a hitch free hostel allocation the manual way without having it automated.

With the world's daily processes going automated through the use of computers, it would be highly beneficial to also have hostel allocation tow this line to ease the administrative overhead encountered by the respective authorities in charge of hostel for example the student affairs department.

**CHAPTER THREE**

 **System Analysis and Design**

**3.0 INTRODUCTION**

This system was designed as a web application. It was made to have two roles for hostel admin and another for students. The general concept is that the admin can create hostels and rooms and can then upload records of those that have paid school fees. Students can then go ahead to obtain hostel from the system. Having the system as a web based solution would make it easy for the admin and students to easily access the application anywhere they are as long as they have internet access.

**3.1 METHODOLOGY**

The new system was designed using the OOAD methodology V-Model and is a unique, linear development methodology used during a software development life cycle (SDLC). The V-model focuses on a fairly typical waterfall- esque method. The V-Shape of the V-Model method represents the various stages that will be passed through during the implementation of Hostel allocation to the students. Beginning at the top-left stage and working, overtime, toward the top-right tip, the stages represent a linear progression of development similar to the waterfall model [10].

CO

**Operation and Maintenance**

**Concept of Operation**

**Verification and Validation**

**System Verification and Validation**

**Requirements and Architecture**

**Project Test and Integration**

**Integration, Test and Verification**

**Detailed Design**

**Project**

**Definition**

 **Test and**

**Implementation**

 **Time**

**FIG3.1: V-Model of (SDLC)**

**3.2 Description of the existing system**

 In the existing system of manual hostel allocation to students, in the beginning of each semester includes;

* Student goes to bank to pay school fees.
* Students obtain their school fees receipt being in a very long queue in bursary office.
* Students are orally request for hostel from bursary.
* A card is being issued to each student with names, registration number and hostel name.
* Students proceed to hostel and register with students affairs.

**3.3 Analysis of the proposed system**

In this project work called Design and implementation of hostel allocation to the student, there are two phases one for students and the other for admin. The hostel allocation link is placed on school website in student services and it gives login provision for both admin and students of the University. During the design it is assume that hostel allocation starts immediately when the semester start.

**Phase 1: Admin**

* Login with Username and Password
* Add Hostel, Block, Room and Bunk Space.
* Upload payment details.
* Create login details.
* Generation of hostel report.

**Phase 2: Students**

* Login with Registration Number and Receipt number
* Get Hostel Space.
* Printout acknowledgement Slip.

**During analysis of the work above the following problems were encountered.**

* Insufficient network connection
* Inability to login when the details are not created for both staff and students.
* Student’s inability to get hostel when the payment details are not uploaded and save by staff or Hostel administrator.
* Student inability to be accommodated in hostel if the acknowledgement slip is not presented to the students affairs

**UNIFIED MODEL LANGUAGE**

**Use case diagram**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different **use** cases in which the user is involved [11].

The large rectangle box is called Customer Relationship Management (CRM); it contains different activities taking place in hostel allocation. Roles individual play to complete hostel allocation to each student and the images in human form are the actors that play those roles.

 **SYSTEM CRM**

 **Fig 3.1: Use case Diagram**

 **3.3 Design of the proposed system**

For the database backend of this project, Microsoft SQL Server was used. The database was modeled in such a way as to properly represent the key entities and concepts used presented in the solution. To that end the following tables where created.

* hostels
* blocks
* rooms
* bunk\_space
* payment\_details



Table 3.1: Hostel Name



Table 3.2: Hostel Block



**Table 3.3: Hostel Room**



**Table 3.4: Hostel Bunk space**



**Table 3.5: Hostel Payment Details**

The overall structure of the system is to make provision for only students whose details have been uploaded into the database be able to get a hostel and to get it only ones.

Sequel to that the system is designed to contain the available room spaces and also the records of the qualified students. Provision is equally made for the admin to add more hostels, rooms and bunk spaces.

For a student to get a room the interface presents only available room spaces for him/her to select based on choice. When a room is gotten the students registration number is logged against the record of the room and an acknowledgement page is to be displayed. With this, record of the room spaces can be generated showing occupied and empty rooms.

**System Architecture**

The overall structure of the system is to make provision for only students whose details have been uploaded into the database be able to get a hostel and to get it only ones.

Sequel to that the system is designed to contain the available room spaces and also the records of the qualified students.

For a student to get a room the interface presents only available room spaces for him/her to select based on choice. When a room is gotten the students registration number is logged against the record of the room and an acknowledgement page is to be displayed. With this record of the room spaces can be generated showing occupied and empty rooms.

**INPUT DESIGN**

Input design is the process of converting user originated input to computer based on the format.

The inputdesign of the proposed system can be grouped in this manner;

* Login input
* Add hostel
* Add block
* Add room
* Upload payment detail

**OUTPUT DESIGN**

* Get hostel
* Print hostel report
* Print acknowledgement slip.

**CHAPTER FOUR**

 **SYSTEM IMPLEMENTATION**

**4.0 INTRODUCTION**

This chapter contain system implementation such as Choice of development Environment, Implementation Architecture, Software testing, Documentation, User manual and source code listing. The above mentioned features show the real form of this project being implemented from scratch.

**4.1 CHOICE OF DEVELOPMENT ENVIRONMENT**

Being a web based project, the design and implementation of this project was executed using Microsoft development stack. For this project this stack consists of C# programming language using the .NET framework, ASP.NET web forms, Microsoft SQL Server and Visual Studio for IDE

ASP.Net Web forms was used as the server framework due to its ease of us in creating medium to complex application and also the vast number of components and libraries that exists for it. It has also been known to be stable in functionality with a very good support backing from Microsoft. Equally large number of cooperate organizations like banks and government agencies have deployed solutions built entirely with ASP.NET web forms.

As a component of the dot net framework, asp.net web forms can be programmed either in C# or VB.Net. However for this project C# was used due to its matured syntax and the expressive ability it gives developers to easily abstract out and solves problems using object oriented programming construct.

Having chosen dot net framework as the solution platform for this projects it goes natural to use Microsoft SQL Server for the database backend. It will be good to note that any capable database system can be used for this project like oracle, mysqlDB, mariaDB, postgreDB and others, however due to the level of dot net interoperability functionality build into SQL Server it was chosen as the database system for this project.



**Fig 4.1: Screenshot of SQL Environment**

Lastly Visual Studio was employed as the Integrated Development Environment for the project. Visual Studio is arguably one of the best IDEs for development on Microsoft Windows. It comes packed with components and functions that make programming easy and enjoyable such as intellisense, syntax highlighting, code-folding, code refactoring, code navigations, integrated build system (debugger, compiler etc) and so many other tools and plugins. Visual studio makes it easy for one to manage his/her project workspace, organizing files and project assets in neat and understandable manner.

For example to create a c# web form application in visual studio, on the IDE menu click File -> New Project or web site...

****

**Fig 4.2: Screen shot of Visual studio start page**

From the popup, select Visual C# under Templates (in the right column) and select ASP.NET Web form Site Application from the middle column.

**Fig 4.3: Screen shot of Visual studio start page**

In the name section, give any appropriate project name, location where you want to create all the project files and solution name.

Click OK to create the webform project. Program.cs will be created as default .cs file in Visual Studio where you can write your C# code in Program class as shown below.



**Fig 4.4: Screen shot Implementation Environment**

Thus, you can create a web form application with C# using Visual Studio.

**4.2 IMPLEMENTATION ARCHITECTURE**

The overview architecture of the implemented system consists of three parts: the front end, the server backend and the database end. Functionally the system is implemented as shown below

**INPUT**

**Username &**

**Password**

**INPUT**

**Reg number &**

**Receipt Number**

**Get Hostel**

**Add Hostel**

**Block**

**Bunk**

**Room**

**Payment**

**Report**

 **LOGOUT**

**PRINT**

**Acknowledgement Slip**

**LOGIN**

**Students**

**Admin**

**Hostel Manager**

**Admin**

**Student**

**Fig 4.5: Screen shot Implementation Architecture**

**4.3 SOFTWARE TESTING**

Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product: Meets the business and technical requirements that guided its design and development [13]

This shows steps to be taken to access hostel allocation from login to add hostel, get hostel, and print acknowledgement slip.

****

**Fig 4.6: Login Screen Shot**

**Fig 4.7: Admin’s Dashboard Screen Shot**

 ****

**Fig 4.8: Add Hostel Screen Shot**

****

**Fig 4.9: Upload Payment Details Screen Shot**

****

**Fig 4.10 Get Hostel Screen Shot**

****

**Fig 4.11: Acknowledgement Slip**

**4.4 Documentation**

Documentation is the process of writing the written text or illustrations that accompany computer software. It explains how the software is being used from the input design to the output design.

**4.4.1 User manual**

* Open a browser either chromes, fire fox, explorer and so on
* Visit [www.hostelManager.com](http://www.hostelManager.com)
* Click on the login form
* Fill your detail correctly
* Login to Hostel Manager Dashboard
* As an admin Username and password

 Add hostel

 Add block

 Add room

 Add bunk space

 Upload Payment detail (It must be in excel format when uploading in the process every upload must be save to ensure successfully upload)

 Download and print hostel report

* As a student Reg No and Receipt No

Get hostel

Print acknowledgement slip

* Logout

**CHAPTER FIVE SUMMARY**

**5.0 CONCLUSION**

Based on the information being carried out, the following conclusion is being drawn;

1. Hostel allocation done by students using this system consumes less time and fastened the settling down for academic session.
2. It also eliminates queuing of students during resumption.
3. It helps captures the number of students in the hostel as they are being assigned.

**5.1 RECOMMENDATIONS**

Having observed that most students in the University encounter one problem or the other during hostel allocation the following recommendation are proffered;

1. The management should endeavor to upload new hostels, renew the accommodation platform and place the advert on website in time before the semester runs off.
2. The management should allow students to select the particular hostel they want with this software.
3. All the students should endeavor to have their accommodation slip immediately after hostel is being allocated to you online.
4. All students should make sure that the occupant of the particular bunk report to hostel at least before the end of three days where ever you are.

**REFERENCES**

#### [1] **Joe Project Store,** Design and implementation of an Electronic Hostel management system [Online] Available:[https://iproject.com.ng/computer-science/design-and-implementation- of-hostel-allocation-system/index.html](https://iproject.com.ng/computer-science/design-and-implementation-%20of-hostel-allocation-system/index.html)

[2] Stephen, (2011, May 16) World’s First Youth Hostel [Online] Available: <http://blog.hihostels.com/2011/05/brief-history-of-hostelling/>

[3] UniAssignment Center, () Rationale Enquiries (00)35722768633 [Online]Available:<https://www.uniassignment.com/essay-samples/information-technology/the-history-of-the-hostel-management-systems-information-technology-essay.php>

[4] (2018 July 9) The World’s Large Community of Hotel Professional [Online] Available: <https://hostelmanagement.com/glossary/hostel.html>

[5] **Prof. Peter A,** (2017/2018) **University Hostel-Accommodation** [Online] Available:<http://www.unn.edu.ng/update-ii-university-hostel-accommodation/>

[6] Amos Abba and Damsa Ahangba, (July 2015) controversy over hostel allocation. [Online] Available: www.Thenationonlineng.net/

[7] I.B.Nga, (2016, October 2) Accommodation Racket at UNN [Online] Available:https://www.google.com.ng/search?q=Accommodation+Racket+at+UNN&oq=Accommodation+Racket+at+UNN&aqs=chrome..69i57.523j0j7&sourceid=chrome&ie=UTF-8

[8] Ekeh Ikenta, (2017 July) Development of Virtual Class Room

#  [9] Alice Keeler, (2018 July 9) C# tutorial [Online] Available:

[www.tutorialsteacher/www.tutorialsteacher/www.tutorialsteacher.com/csharp/csharp-tutorials.html](http://www.tutorialsteacher/www.tutorialsteacher/www.tutorialsteacher.com/csharp/csharp-tutorials.html)

[10] Andrew Powell-Morse (2016 December 26) the process of v model [Online] Available: <https://airbrake.io/blog.sdlc/v-model>

[11] McLaughlin et al (2018, July 22) Retrieved Use case diagram [Online] Available: https://en.wikipedia.org/wiki/use\_case\_diagram

 **APPENDIX**

**4.4.2 Source code listing (.aspx and .aspx.cs)**

**LOGIN.aspx.cs**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class Login : System.Web.UI.Page

{

 protected void Page\_Load(object sender, EventArgs e)

 {

 if(!Page.IsPostBack)

 {

 if(Request.QueryString["auth"] != null && Request.QueryString["auth"] == "1")

 {

 ProcessLogin();

 }

 }

 }

 protected void btnLogin\_Click(object sender, EventArgs e)

 {

 try

 {

 string username = txtUsername.Text;

 string Password = txtPassword.Text;

 //first check if it is student

 if (IsStudent(username))

 {

 //check if receipt no is valid

 if (!IsReceiptValid(username, Password))

 {

 ShowMessage("Wrong Receipt Number");

 return;

 }

 //check if student has already gotten hostel and move him over to the acknowledgment slip

 if (HasGottenHostel(username))

 {

 Response.Redirect(String.Format("~/Modules/Hostel/HostelAcknowledgement.aspx?sent={0}", Encryptor.UrlEncrypt(username)));

 }

 //if student has not gotten hostel before move him/her to Get hostel module

 Response.Redirect(String.Format("~/Modules/Hostel/StudentGetHostel.aspx?regno={0}&receiptno={1}", Encryptor.UrlEncrypt(username), Encryptor.UrlEncrypt(Password)));

 }

 if (!DB.DoesUsernameExists(username))

 {

 ShowMessage("The username [" + username + "] does not exists !");

 return;

 }

 if (!DB.DoesUsernamePasswordExists(username, Password))

 {

 ShowMessage("Invalid Password !");

 return;

 }

 //if (Membership.GetUser(username).IsOnline)

 //{

 // ShowMessage("The User is Currently Logged in !");

 // DB.SignUserOut(username);

 FormsAuthentication.SetAuthCookie(username, true);

 // return;

 //}

 Membership.GetUser(username).UnlockUser();

 if (DB.IsAccountLockedOut(username))

 {

 DB.UnLockUserByName(username);

 }

 FormsAuthentication.RedirectFromLoginPage(username, false);

 Response.Redirect("~/login.aspx?auth=1");

 ProcessLogin();

 }

 catch (Exception ex)

 {

 Response.Write(ex.Message);

 }

 }

 private bool IsStudent(string regno)

 {

 DataSet ds = DB.LoadDatasource("SELECT ID FROM PAYMENT\_DETAILS WHERE REG\_NO = {0}", regno);

 if(ds != null && ds.Tables[0].Rows.Count > 0)

 {

 return true;

 }

 return false;

 }

 private bool IsReceiptValid(string regNo, string receiptNo)

 {

 DataSet ds = DB.LoadDatasource("SELECT ID FROM PAYMENT\_DETAILS WHERE REG\_NO = {0} AND RECEIPT\_NO = {1}", regNo, receiptNo);

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 return true;

 }

 return false;

 }

 private bool HasGottenHostel(string regno)

 {

 DataSet ds = DB.LoadDatasource("SELECT REG\_NO FROM vw\_hostel\_allocation WHERE REG\_NO = {0}", regno);

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 return true;

 }

 return false;

 }

 private void ShowMessage(string strMsg)

 {

 string csName = "popupScript";

 Type csType = this.GetType();

 ClientScriptManager cs = Page.ClientScript;

 if (!cs.IsStartupScriptRegistered(csType, csName))

 {

 String cstext = "alert('" + strMsg + "');";

 cs.RegisterStartupScript(csType, csName, cstext, true);

 }

 }

 private void ProcessLogin()

 {

 if (HttpContext.Current.User.Identity.IsAuthenticated)

 {

 if (Session["Username"] != null)

 {

 if (Convert.ToString(Session["Username"]) != HttpContext.Current.User.Identity.Name.ToString())

 {

 Response.Redirect("~/Logout.aspx");

 }

 }

 String[] rolenames = new String[5];

 String rolename;

 String role;

 String Sessionid = Session.SessionID;

 SqlConnection con = new SqlConnection();

 SqlCommand cmdRole = new SqlCommand();

 rolenames = Roles.GetRolesForUser(HttpContext.Current.User.Identity.Name);

 rolename = (String)rolenames.GetValue(0);

 con.ConnectionString = ConfigurationManager.AppSettings["ConnStr"];

 cmdRole.CommandType = CommandType.StoredProcedure;

 cmdRole.CommandText = "dbo.STP\_GET\_ROLE\_ID";

 cmdRole.Connection = con;

 cmdRole.Parameters.Add(new SqlParameter("@RoleName", SqlDbType.VarChar, 50));

 cmdRole.Parameters["@RoleName"].Value = rolename;

 cmdRole.Parameters.Add(new SqlParameter("@RoleID", SqlDbType.VarChar, 100));

 cmdRole.Parameters["@RoleID"].Direction = ParameterDirection.Output;

 try

 {

 con.Open();

 cmdRole.ExecuteNonQuery();

 role = (String)cmdRole.Parameters["@RoleID"].Value;

 Session["role"] = (String)cmdRole.Parameters["@RoleID"].Value;

 Response.Cookies["User"].Value = "";

 Response.Cookies["RoleId"].Value = "";

 Response.Cookies["pur"].Expires = DateTime.Now.AddDays(1);

 Response.Cookies["RoleId"].Expires = DateTime.Now.AddDays(1);

 Response.Cookies["User"].Expires = DateTime.Now.AddDays(1);

 Response.Cookies["RoleId"].Value = role; //assig

 Response.Cookies["User"].Value = HttpContext.Current.User.Identity.Name;

 Session["Username"] = HttpContext.Current.User.Identity.Name;

 }

 catch (Exception ex)

 {

 Response.Redirect("login.aspx?msg=" + ex.Message, true);

 //con.Close;

 }

 finally

 {

 if (con != null)

 {

 con.Close();

 }

 }

 string userid = DB.GetASPNetUserId(HttpContext.Current.User.Identity.Name);

 DB.LockUser(userid);

 Response.Redirect("~/dashboard.aspx", true);

 }

 else

 {

 Response.Redirect("~/login.aspx?err=2", true);

 }

 }

}

**AddHostel.aspx.cs**

public partial class Modules\_Hostel\_AddHostel : System.Web.UI.Page

{

 protected void Page\_Load(object sender, EventArgs e)

 {

 lblError.Text = "";

 }

 protected void btnView\_Click(object sender, EventArgs e)

 {

 ShowHostels();

 }

 private void ShowHostels()

 {

 txtHostelName.Text = "";

 ddlGender.SelectedIndex = 0;

 DataSet ds = DB.LoadDatasource("Select \* from Hostel");

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 DataTable dt = new DataTable();

 dt.Columns.Add("SN");

 dt.Columns.Add("HOSTEL\_NAME");

 dt.Columns.Add("GENDER");

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 string name = Convert.ToString(ds.Tables[0].Rows[i]["HOSTEL\_NAME"]);

 string gender = Convert.ToString(ds.Tables[0].Rows[i]["GENDER"]);

 dt.Rows.Add((i + 1).ToString(), name, gender);

 }

 gvHostel.DataSource = dt;

 gvHostel.DataBind();

 pnlView.Enabled = false;

 mView.ActiveViewIndex = 0;

 }

 }

 protected void btnAdd\_Click(object sender, EventArgs e)

 {

 if(txtHostelName.Text.Trim() == "")

 {

 lblError.Text = "Enter Hostel Name";

 return;

 }

 if(ddlGender.SelectedIndex < 1)

 {

 lblError.Text = "Select Gender";

 return;

 }

 string hostelName = txtHostelName.Text;

 string gender = ddlGender.SelectedValue;

 Int64 ret = InsertHostel(hostelName, gender);

 if(ret > 0)

 {

 lblError.Text = "Hostel Successfully Added";

 }

 else if(ret == -200)

 {

 lblError.Text = "Duplicate hostel name not allowed";

 }

 else if(ret != -100)

 {

 lblError.Text = "Operation Failed";

 }

 ShowHostels();

 }

 private Int64 InsertHostel(string HostelName, string Gender)

 {

 string constr = ConfigurationManager.AppSettings["ConnStr"];

 SqlConnection conn = new SqlConnection(constr);

 conn.Open();

 SqlCommand cmd = new SqlCommand("STP\_ADD\_HOSTEL", conn);

 cmd.CommandType = CommandType.StoredProcedure;

 Int64 ret = -100;

 try

 {

 // output param section

 SqlParameter paramId = new SqlParameter("@paramId", SqlDbType.Int);

 paramId.Direction = ParameterDirection.Output;

 cmd.Parameters.Add(paramId);

 //input param section

 cmd.Parameters.Add("@paramHostelName", SqlDbType.VarChar, 500).Value = HostelName;

 cmd.Parameters.Add("@paramGender", SqlDbType.VarChar, 2).Value = Gender;

 cmd.ExecuteNonQuery();

 // output param return value section

 ret = Convert.ToInt64(cmd.Parameters["@paramId"].Value);

 }

 catch (Exception ex)

 {

 //add implementation

 lblError.Text = ex.Message;

 }

 finally

 {

 conn.Close();

 }

 return ret;

 }

}

**Addblock.aspx.cs**

public partial class Modules\_Hostel\_AddBlock : System.Web.UI.Page

{

 protected void Page\_Load(object sender, EventArgs e)

 {

 lblError.Text = "";

 if(!Page.IsPostBack)

 {

 PopulateHostel();

 }

 }

 private void PopulateHostel()

 {

 try

 {

 string hostel = "";

 string hostelId = "";

 ddlHostel.Items.Clear();

 ddlHostel.Items.Add("Select Hostel");

 DataSet dsState = DB.LoadDatasource("Select \* from hostel Order by hostel\_name ");

 if (dsState != null && dsState.Tables[0].Rows.Count > 0)

 {

 for (int i = 0; i < dsState.Tables[0].Rows.Count; i++)

 {

 hostel = Convert.ToString(dsState.Tables[0].Rows[i]["hostel\_name"]);

 hostelId = Convert.ToString(dsState.Tables[0].Rows[i]["hostel\_id"]).Trim();

 ddlHostel.Items.Add(new ListItem(hostel, hostelId));

 }

 }

 }

 catch (Exception ex)

 {

 lblError.Text = "Error : " + ex.Message;

 }

 }

 protected void btnView\_Click(object sender, EventArgs e)

 {

 if(ddlHostel.SelectedIndex < 1)

 {

 lblError.Text = "Select Hostel";

 return;

 }

 string hostelId = ddlHostel.SelectedValue;

 ShowBlocks(hostelId);

 }

 private void ShowBlocks( string hostelId)

 {

 txtBlockName.Text = "";

 DataSet ds = DB.LoadDatasource("Select \* from blocks where hostel\_id = " + hostelId);

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 DataTable dt = new DataTable();

 dt.Columns.Add("SN");

 dt.Columns.Add("BLOCK\_NAME");

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 string name = Convert.ToString(ds.Tables[0].Rows[i]["BLOCK\_NAME"]);

 dt.Rows.Add((i + 1).ToString(), name);

 }

 gvHostel.DataSource = dt;

 gvHostel.DataBind();

 }

 pnlView.Enabled = false;

 mView.ActiveViewIndex = 0;

 }

 protected void btnAdd\_Click(object sender, EventArgs e)

 {

 if (txtBlockName.Text.Trim() == "")

 {

 lblError.Text = "Enter Block Name";

 return;

 }

 string hostelId = ddlHostel.SelectedValue;

 string blockName = txtBlockName.Text;

 Int64 ret = InsertBlock(hostelId, blockName);

 if (ret > 0)

 {

 lblError.Text = "Block Successfully Added";

 }

 else if (ret == -200)

 {

 lblError.Text = "Duplicate Block Name in a hostel is not allowed";

 }

 else if (ret != -100)

 {

 lblError.Text = "Operation Failed";

 }

 ShowBlocks(hostelId);

 }

 private Int64 InsertBlock(string HostelId, string BlockName)

 {

 string constr = ConfigurationManager.AppSettings["ConnStr"];

 SqlConnection conn = new SqlConnection(constr);

 conn.Open();

 SqlCommand cmd = new SqlCommand("STP\_ADD\_BLOCK", conn);

 cmd.CommandType = CommandType.StoredProcedure;

 Int64 ret = -100;

 try

 {

 // output param section

 SqlParameter paramId = new SqlParameter("@paramId", SqlDbType.Int);

 paramId.Direction = ParameterDirection.Output;

 cmd.Parameters.Add(paramId);

 //input param section

 cmd.Parameters.Add("@paramHostelId", SqlDbType.Int).Value = HostelId;

 cmd.Parameters.Add("@paramBlockName", SqlDbType.VarChar, 500).Value = BlockName;

 cmd.ExecuteNonQuery();

 // output param return value section

 ret = Convert.ToInt64(cmd.Parameters["@paramId"].Value);

 }

 catch (Exception ex)

 {

 //add implementation

 lblError.Text = ex.Message;

 }

 finally

 {

 conn.Close();

 }

 return ret;

 }

 protected void btnBack\_Click(object sender, EventArgs e)

 {

 ddlHostel.SelectedIndex = 0;

 pnlView.Enabled = true;

 mView.ActiveViewIndex = -1;

 }

}

**AddRoom.aspx.cs**

public partial class Modules\_Hostel\_AddRoom : System.Web.UI.Page

{

 protected void Page\_Load(object sender, EventArgs e)

 {

 lblError.Text = "";

 if (!Page.IsPostBack)

 {

 PopulateHostel();

 }

 }

 private void PopulateHostel()

 {

 try

 {

 string hostel = "";

 string hostelId = "";

 ddlHostel.Items.Clear();

 ddlHostel.Items.Add("Select Hostel");

 DataSet dsState = DB.LoadDatasource("Select \* from hostel Order by hostel\_name ");

 if (dsState != null && dsState.Tables[0].Rows.Count > 0)

 {

 for (int i = 0; i < dsState.Tables[0].Rows.Count; i++)

 {

 hostel = Convert.ToString(dsState.Tables[0].Rows[i]["hostel\_name"]);

 hostelId = Convert.ToString(dsState.Tables[0].Rows[i]["hostel\_id"]).Trim();

 ddlHostel.Items.Add(new ListItem(hostel, hostelId));

 }

 }

 }

 catch (Exception ex)

 {

 lblError.Text = "Error : " + ex.Message;

 }

 }

 protected void btnView\_Click(object sender, EventArgs e)

 {

 if (ddlHostel.SelectedIndex < 1)

 {

 lblError.Text = "Select Hostel";

 return;

 }

 if (ddlBlock.SelectedIndex < 1)

 {

 lblError.Text = "Select Block";

 return;

 }

 string blockId = ddlBlock.SelectedValue;

 ShowRooms(blockId);

 }

 private void ShowRooms(string blockId)

 {

 txtRoomName.Text = "";

 DataSet ds = DB.LoadDatasource("Select \* from room where block\_id = " + blockId);

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 DataTable dt = new DataTable();

 dt.Columns.Add("SN");

 dt.Columns.Add("ROOM\_NAME");

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 string name = Convert.ToString(ds.Tables[0].Rows[i]["ROOM\_NAME"]);

 dt.Rows.Add((i + 1).ToString(), name);

 }

 gvBlock.DataSource = dt;

 gvBlock.DataBind();

 }

 pnlView.Enabled = false;

 mView.ActiveViewIndex = 0;

 }

 protected void btnAdd\_Click(object sender, EventArgs e)

 {

 if (txtRoomName.Text.Trim() == "")

 {

 lblError.Text = "Enter Room Name";

 return;

 }

 string blockId = ddlBlock.SelectedValue;

 string roomName = txtRoomName.Text;

 Int64 ret = InsertRoom(blockId, roomName);

 if (ret > 0)

 {

 lblError.Text = "Room Successfully Added";

 }

 else if (ret == -200)

 {

 lblError.Text = "Duplicate Room Name in a Block is not allowed";

 }

 else if (ret != -100)

 {

 lblError.Text = "Operation Failed";

 }

 ShowRooms(blockId);

 }

 private Int64 InsertRoom(string blockId, string roomName)

 {

 string constr = ConfigurationManager.AppSettings["ConnStr"];

 SqlConnection conn = new SqlConnection(constr);

 conn.Open();

 SqlCommand cmd = new SqlCommand("STP\_ADD\_ROOM", conn);

 cmd.CommandType = CommandType.StoredProcedure;

 Int64 ret = -100;

 try

 {

 // output param section

 SqlParameter paramId = new SqlParameter("@paramId", SqlDbType.Int);

 paramId.Direction = ParameterDirection.Output;

 cmd.Parameters.Add(paramId);

 //input param section

 cmd.Parameters.Add("@paramBlockId", SqlDbType.Int).Value = blockId;

 cmd.Parameters.Add("@paramRoomName", SqlDbType.VarChar, 500).Value = roomName;

 cmd.ExecuteNonQuery();

 // output param return value section

 ret = Convert.ToInt64(cmd.Parameters["@paramId"].Value);

 }

 catch (Exception ex)

 {

 //add implementation

 lblError.Text = ex.Message;

 }

 finally

 {

 conn.Close();

 }

 return ret;

 }

 protected void btnBack\_Click(object sender, EventArgs e)

 {

 pnlView.Enabled = true;

 mView.ActiveViewIndex = -1;

 }

 protected void ddlHostel\_SelectedIndexChanged(object sender, EventArgs e)

 {

 if(ddlHostel.SelectedIndex < 1)

 {

 return;

 }

 string hostelId = ddlHostel.SelectedValue;

 PopulateBlock(hostelId);

 }

 private void PopulateBlock(string hostelId)

 {

 try

 {

 string blockName = "";

 string blockId = "";

 ddlBlock.Items.Clear();

 ddlBlock.Items.Add("Select Block");

 DataSet ds = DB.LoadDatasource("Select \* from blocks where hostel\_id = {0} Order by block\_name ", hostelId);

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 blockName = Convert.ToString(ds.Tables[0].Rows[i]["block\_name"]);

 blockId = Convert.ToString(ds.Tables[0].Rows[i]["block\_id"]).Trim();

 ddlBlock.Items.Add(new ListItem(blockName, blockId));

 }

 }

 }

 catch (Exception ex)

 {

 lblError.Text = "Error : " + ex.Message;

 }

 }

}

**Addbunkspace.aspx.cs**

public partial class Modules\_Hostel\_AddBunkSpace : System.Web.UI.Page

{

 protected void Page\_Load(object sender, EventArgs e)

 {

 lblError.Text = "";

 if (!Page.IsPostBack)

 {

 PopulateHostel();

 }

 }

 private void PopulateHostel()

 {

 try

 {

 string hostel = "";

 string hostelId = "";

 ddlHostel.Items.Clear();

 ddlHostel.Items.Add("Select Hostel");

 DataSet dsState = DB.LoadDatasource("Select \* from hostel Order by hostel\_name ");

 if (dsState != null && dsState.Tables[0].Rows.Count > 0)

 {

 for (int i = 0; i < dsState.Tables[0].Rows.Count; i++)

 {

 hostel = Convert.ToString(dsState.Tables[0].Rows[i]["hostel\_name"]);

 hostelId = Convert.ToString(dsState.Tables[0].Rows[i]["hostel\_id"]).Trim();

 ddlHostel.Items.Add(new ListItem(hostel, hostelId));

 }

 }

 }

 catch (Exception ex)

 {

 lblError.Text = "Error : " + ex.Message;

 }

 }

 protected void btnView\_Click(object sender, EventArgs e)

 {

 if (ddlHostel.SelectedIndex < 1)

 {

 lblError.Text = "Select Hostel";

 return;

 }

 if (ddlBlock.SelectedIndex < 1)

 {

 lblError.Text = "Select Block";

 return;

 }

 if (ddlRoom.SelectedIndex < 1)

 {

 lblError.Text = "Select Room";

 return;

 }

 string roomId = ddlRoom.SelectedValue;

 ShowBunkSpace(roomId);

 }

 private void ShowBunkSpace(string roomId)

 {

 txtBunkSpace.Text = "";

 DataSet ds = DB.LoadDatasource("Select \* from bunk\_space where room\_id = " + roomId);

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 DataTable dt = new DataTable();

 dt.Columns.Add("SN");

 dt.Columns.Add("BUNK\_SPACE");

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 string name = Convert.ToString(ds.Tables[0].Rows[i]["BUNK\_SPACE"]);

 dt.Rows.Add((i + 1).ToString(), name);

 }

 gvBunk.DataSource = dt;

 gvBunk.DataBind();

 }

 pnlView.Enabled = false;

 mView.ActiveViewIndex = 0;

 }

 protected void btnAdd\_Click(object sender, EventArgs e)

 {

 if (txtBunkSpace.Text.Trim() == "")

 {

 lblError.Text = "Enter Bunk Space";

 return;

 }

 string roomId = ddlRoom.SelectedValue;

 string bunkSpaceName = txtBunkSpace.Text;

 Int64 ret = InsertBunkSpace(roomId, bunkSpaceName);

 if (ret > 0)

 {

 lblError.Text = "Bunk Space Successfully Added";

 }

 else if (ret == -200)

 {

 lblError.Text = "Duplicate Bunk Space in a Room is not allowed";

 }

 else if (ret != -100)

 {

 lblError.Text = "Operation Failed";

 }

 ShowBunkSpace(roomId);

 }

 private Int64 InsertBunkSpace(string roomId, string bunkSpaceName)

 {

 string constr = ConfigurationManager.AppSettings["ConnStr"];

 SqlConnection conn = new SqlConnection(constr);

 conn.Open();

 SqlCommand cmd = new SqlCommand("STP\_ADD\_BUNK\_SPACE", conn);

 cmd.CommandType = CommandType.StoredProcedure;

 Int64 ret = -100;

 try

 {

 // output param section

 SqlParameter paramId = new SqlParameter("@paramId", SqlDbType.Int);

 paramId.Direction = ParameterDirection.Output;

 cmd.Parameters.Add(paramId);

 //input param section

 cmd.Parameters.Add("@paramRoomId", SqlDbType.Int).Value = roomId;

 cmd.Parameters.Add("@paramBunkSpace", SqlDbType.VarChar, 500).Value = bunkSpaceName;

 cmd.ExecuteNonQuery();

 // output param return value section

 ret = Convert.ToInt64(cmd.Parameters["@paramId"].Value);

 }

 catch (Exception ex)

 {

 //add implementation

 lblError.Text = ex.Message;

 }

 finally

 {

 conn.Close();

 }

 return ret;

 }

 protected void btnBack\_Click(object sender, EventArgs e)

 {

 pnlView.Enabled = true;

 mView.ActiveViewIndex = -1;

 }

 protected void ddlHostel\_SelectedIndexChanged(object sender, EventArgs e)

 {

 if (ddlHostel.SelectedIndex < 1)

 {

 return;

 }

 string hostelId = ddlHostel.SelectedValue;

 PopulateBlock(hostelId);

 }

 protected void ddlBlock\_SelectedIndexChanged(object sender, EventArgs e)

 {

 if (ddlBlock.SelectedIndex < 1)

 {

 return;

 }

 string blockId = ddlBlock.SelectedValue;

 PopulateRoom(blockId);

 }

 private void PopulateBlock(string hostelId)

 {

 try

 {

 string blockName = "";

 string blockId = "";

 ddlBlock.Items.Clear();

 ddlBlock.Items.Add("Select Block");

 DataSet ds = DB.LoadDatasource("Select \* from blocks WHERE hostel\_id = " + hostelId + " Order by block\_name ");

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 blockName = Convert.ToString(ds.Tables[0].Rows[i]["block\_name"]);

 blockId = Convert.ToString(ds.Tables[0].Rows[i]["block\_id"]).Trim();

 ddlBlock.Items.Add(new ListItem(blockName, blockId));

 }

 }

 }

 catch (Exception ex)

 {

 lblError.Text = "Error : " + ex.Message;

 }

 }

 private void PopulateRoom(string blockId)

 {

 try

 {

 string roomName = "";

 string roomId = "";

 ddlRoom.Items.Clear();

 ddlRoom.Items.Add("Select Room");

 DataSet ds = DB.LoadDatasource("Select \* from room where block\_id = " + blockId + " order by room\_name ");

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

 {

 roomName = Convert.ToString(ds.Tables[0].Rows[i]["room\_name"]);

 roomId = Convert.ToString(ds.Tables[0].Rows[i]["room\_id"]).Trim();

 ddlRoom.Items.Add(new ListItem(roomName, roomId));

 }

 }

 }

 catch (Exception ex)

 {

 lblError.Text = "Error : " + ex.Message;

 }

 }

}

**GetHostel.aspx.cs**

public partial class Modules\_Hostel\_GetHostel : System.Web.UI.Page

{

 public class Student

 {

 public Student(string firstname, string lastname, int id, List<int> scores)

 {

 First = firstname;

 Last = lastname;

 ID = id;

 Scores = scores;

 }

 public string First { get; set; }

 public string Last { get; set; }

 public int ID { get; set; }

 public List<int> Scores;

 }

 protected void Page\_Load(object sender, EventArgs e)

 {

 lblError.Text = "";

 }

 protected void btnSee\_Click(object sender, EventArgs e)

 {

 DataSet ds = ConnectDB.LoadDatasource("SELECT \* FROM vw\_hostel\_allocation");

 if(ds != null && ds.Tables[0].Rows.Count > 0)

 {

 var query =

 from hostel in ds.Tables[0].AsEnumerable()

 select new

 {

 BunkSpace = hostel.Field<String>("bunk\_space"),

 HostelName = hostel.Field<String>("hostel\_name")

 };

 foreach (var record in query)

 {

 Response.Write(String.Format("Bunk Space: {0} Is in hostel: {1} <br/>",

 record.BunkSpace,

 record.HostelName));

 }

 }

 }

 protected void btnGroup\_Click(object sender, EventArgs e)

 {

 DataSet ds = ConnectDB.LoadDatasource("SELECT \* FROM vw\_hostel\_allocation");

 if (ds != null && ds.Tables[0].Rows.Count > 0)

 {

 var query =

 from hostel in ds.Tables[0].AsEnumerable()

 group hostel by hostel.Field<string>("room\_name") into RoomName

 select new

 {

 RoomName = RoomName.Key

 };

 DataTable dt = new DataTable();

 dt.Columns.Add("SN");

 dt.Columns.Add("ROOM\_NAME");

 int count = 0;

 foreach (var record in query)

 {

 count += 1;

 dt.Rows.Add(count.ToString(), record.RoomName);

 }

 gvReport.DataSource = dt;

 gvReport.DataBind();

 for(int i=0; i < gvReport.Rows.Count; i++)

 {

 GridViewRow row = gvReport.Rows[i];

 Label lblRoomName = row.FindControl("lblRoomName") as Label;

 DataList dlRoomList = row.FindControl("dlRoomList") as DataList;

 string roomName = lblRoomName.Text.Trim();

 var roomQuery =

 from hostel in ds.Tables[0].AsEnumerable()

 where hostel.Field<string>("room\_name") == roomName

 select hostel;

 DataTable dtBunk = new DataTable();

 dtBunk.Columns.Add("BUNK\_SPACE");

 dtBunk.Columns.Add("DATA");

 foreach (var r in roomQuery)

 {

 string bunkSpace = r.Field<string>("bunk\_space");

 string data = roomName + "#" + bunkSpace;

 dtBunk.Rows.Add(bunkSpace, data);

 }

 dlRoomList.DataSource = dtBunk;

 dlRoomList.DataBind();

 }

 }

 }

 protected void dlRoomList\_ItemCommand(object source, DataListCommandEventArgs e)

 {

 string data = e.CommandArgument.ToString();

 Response.Write(data);

 }

}