1. Introduction

1.1 Project Overview

The purpose of this project is to create an HMS system for an educational institution “College of Engineering, Vatakara “ where different hostels are running and to manage their day to day work and keep their record and data updated. The motivation behind selection of this project was that the team wished to do something more than instead just receiving the requirements, producing the end product.

All the hostels under this institution at present are managed manually by the hostel office. The Registration form verification to the different data processing are done manually. Thus there are a lot of repetitions which can be easily avoided. And hence there is a lot of strain on the person who are running the hostel and software are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

1.2 Organization Profile

Background of Organization

College of Engineering Vadakara is a new generation Engineering College, and the first Engineering College under the Co-operative Academy of Professional Education (CAPE), Thiruvananthapuram, established by the Govt. of Kerala, with the Chief Minister of Kerala as its Chairman. The college started functioning with all state of- art facilities in June 1999 at Kurunthodi of Maniyoor Grama Panchayath.

The college is affiliated to Cochin University of Science and Technology (CUSAT) and Kerala Technological University (KTU) approved by All India Council for Technical Education (AICTE).

The college have different hostels in different location nearby Vatakara and Kurunthodi. Large no. of student are accommodating hostels ,every academic session hostel accommodation required by students and the hostel management work and storing records are done in manual basis.

Vision

To be a world class technical education institute with the highest quality and standards of excellence to meet the demands of business industry and the community and thereby to contribute to India's socio-economic progress

Mission

To develop high quality technical personnel with a sound footing on basic engineering principles, innovative research capabilities and exemplary professional conduct to lead and use technology for the progress of mankind, adapting themselves to changing technological environment with the highest ethical values as the inner strength.

2. System Analysis

2.1 The Existing System

For the past few years the number of educational institutions are increasing rapidly. Thereby the number of hostels are also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hostel and software are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the following drawbacks of the existing system.

- More human error.
- More strength and strain of manual labour needed
- Repetition of the same procedures.
- Low security
- Data redundancy
- Difficult to handle
- Difficult to update data
- Record keeping is difficult
- Backup data can be easily generated

The existing hostel management system have most of the operations like checking out of rooms, allocating rooms ,calculating hostel bill , vacating rooms, management of notice board details etc. done manually which includes a lot of human error, more data redundancy and wastage of time.

2.2 The Proposed System

The drawback of the existing system is that it is very difficult to retrieve data from records. It is difficult to handle the whole system manually and less accurate to keep the data records for future reference because it may get destroyed.

Moreover it is very difficult to retrieve data. Redundancy of data may occur and this may lead to the inconsistency. The manual system is so time-consuming. The proposed system is very easy to operate. Speed and accuracy are the main advantages of proposed system. There is no redundancy of data. The data are stored in the computer secondary memories like hard disk, etc. it can be easily receive and used at any time. The proposed system will easily handle all the data and the work done by the existing systems. The proposed systems eliminate the drawbacks of the existing system to a great extent and it provides tight security to data.

The new system introduce operation like checking out of rooms, allocating rooms , calculating hostel bill , vacating rooms, management of notice board ,alerts the admin with every new hostel accommodation request. The special features of proposed systems are follows:

- Student registration
- Monitoring Student Bills and Payments
- View and manage hostel activities
- Feedback and Complaints

Module Description

The project have mainly 4 different Modules:

- Student
- Warden
- Other Staff
- Admin

Student Module

The student module consist of the activities related to students. The module contain following operations:

- ✓ Submitting hostel Accommodation form
- ✓ View Bills and Payments history
- ✓ Submitting Complaints

- ✓ Submitting Leave Request
- ✓ Submitting Vacate Request
- ✓ Pay Bills

Warden Module

The Warden module consist of the activities related to warden of the hostel. The module contain following operations:

- ✓ Deal forwarded request from Admin and assign a room for requester
- ✓ Deal student's complaints
- ✓ Approve student's leave request
- ✓ Entry and monitor student In/Out time
- ✓ Deal student's room vacate requests

Other Staff Module

The Other Staff module consist of the activities related to service provider staff like mess staff and laundry man. The module contain following operation:

- ✓ Manage and entry service details
- ✓ Provide services

Admin Module

The Admin module consist of the activities related to administrator. The module contain following operations:

- ✓ Create and Manage User
- ✓ Create and Manage Hostel
- ✓ Verify Student and Visitor's Request
- ✓ Notification entry
- ✓ Deal Student's critical complaints
- ✓ Generate Bill's and manage
- ✓ Receive Payments

2.3 Feasibility Study

Technical Feasibility

This checks that, can the work for the project be done with current equipment and existing software technology and available personnel. Since there is no other technical supports are necessary for the function of the system we can conclude that it is technically feasible. When it is implemented there is only need of a computer (PC) and its other peripherals as usual so it is very satisfied in the matter of technical feasibility.

Economic Feasibility

Economic feasibility analysis is the most frequently used evaluating technique for the effectiveness of the candidate system. This checks, are their sufficient benefits in creating the system to make cost acceptable. In the case of the proposed system it is very necessary to implement in such a firm and when its necessity compared its cost for implementing is very low and it is very acceptable by the users of the system. When its advantages and efficiency evaluated it is economically feasible.

Operational Feasibility

The proposed systems are beneficial only if they can turn into information system that will meet the organization's operating requirements. This test of feasibility asks if the system will work when it is developed and installed. Since this system is developed in PHP and other web resources its operation is very simple, very attractive and user friendly. The software is very much available, and it is the most using package now, and since it is an open source and it provides many tools which helps to make the usage very easier. It is very much acceptable by the users of the system and it is keeping its standard.

Behavioral Feasibility

People are inherently resisted to change and computers have been known to facilitate change. Since the new proposed system is nothing to do with the ordinary customers and worker resistance to the systems are accessing this system through internet and they are computer literate so resistance from this side is also very less.

2.4 SOFTWARE REQUIREMENT SPECIFICATIONS

FUNCTIONAL SPECIFICATION

a) Create user A/c

The HMS shall allow the admin to create and manage the user accounts.

b) Room Allotment

The HMS will allocate a room to student when they submit their accommodation request.

c) Manage Hostels

The HMS shall allow the admin to create new hostel and manage them.

d) Feedback and Complaints

The HMS shall allow the admin and warden to see the student complaints.

e) Manage and provide Services

The HMS shall allow the user to provide mess and other hostel services and keep the record of it.

f) Due and Payment status

The HMS shall allow the admin to generate the bills of the student and the student must be able to see the dues and payment history.

NON FUNCTIONAL SPECIFICATION

a) Access Permission

The HMS shall have several types of access permissions. For instance, the warden is recognized as the system's administrator thus the warden shall be able to perform any type of activities on the system and both the user's and student profiles. At the same time, the other hostel staff members shall have restricted access to both the user's and student profiles. The public in general shall be restricted from accessing any user profile. However, they shall be granted a read access on the student profile.

b) Reliability

The system must be reliable and faster in operation also it must be user friendly and understandable.

c) Availability

The System must be available for all time for all the user.

3. System Design and Development

3.1 Input Design

Input design is a part of overall system design, which requires care full attention. The major objectives of the input design is to make the data entry easier logical and error free. With this objective the screen for the system are developed. The input design requirement such as user friendliness, consistent. Often the collection of input data is the most expensive part of the system, in terms of both the equipment used and people involved. If the data going into the system is incorrect, then the processing and output will magnify the errors.

Input design is the process of converting the user originated inputs to a computer based format. The goal of designing input data is to make the automation is easy and free from errors. The design for handling input specifies how data are accepted for computer processing. Input design is art of overall system design that needs care full attention and if includes specifies the means by which action are taken. A system whether to accept input produce a report or end processing. The collection of input data is considered to be the most expensive part of the system design.

Format and interactive dialog boxes for giving the right message and help for the user at the right time are also considered for the development of the project. The decisions made during the input design are: -

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable to and understood by the user.

3.2 Process Design

Process design plays an important role in project development. In order to understand the working procedure, process design is necessary. Data Flow Diagram and System Flow chart are the tools used for process design.

System Flow Chart is a graphical representation of the system showing the overall flow of control in processing at the job level; specifies what activities must be done to convert from a physical to logical model.

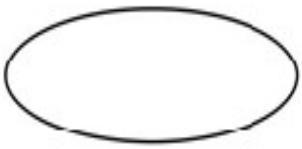
Data Flow Diagram is the logical representation of the data flow of the project. The DFD is drawn using various symbols. It has a source and a destination. The process is represented using circles and source and destination are represented using squares. The data flow is represented using arrows.

One reader can easily get the idea about the project through Data Flow Diagram.

Data Flow Diagrams

A DFD, also known as Bubble Chart, has the purpose of clarifying system requirements and identifies major transformation that will become programs in system design. So it is the starting point of design phase that functionally decomposes the elements specification down to the lowest level of details. A DFD consists of series of bubbles, rectangle and open rectangle joined by Lines, Bubble, Rectangle, Opened Rectangle represents data transformation, sources or destination and data storage respectively. The line represents the data flow in the system.

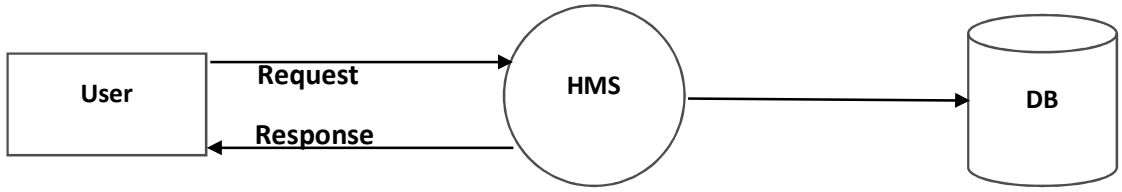
Source or destination



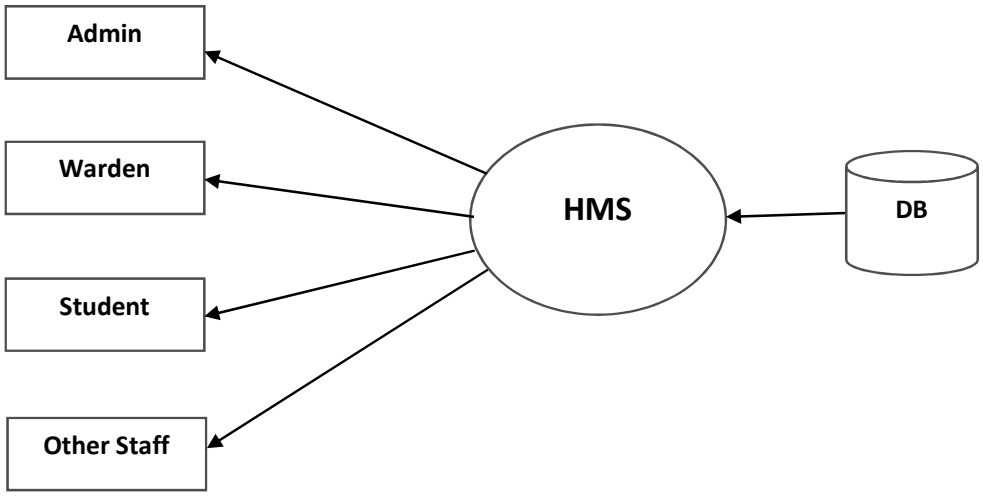
Process that transforms data flow

Data flow

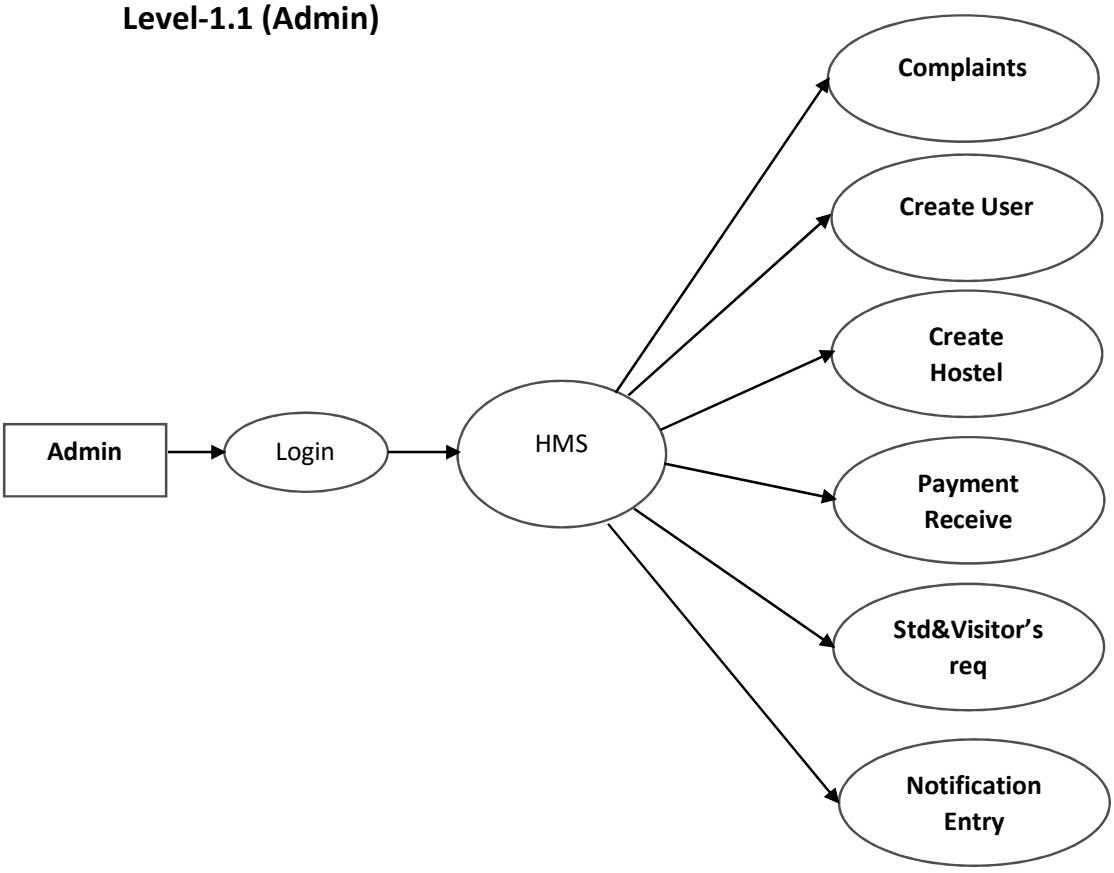
Level-0



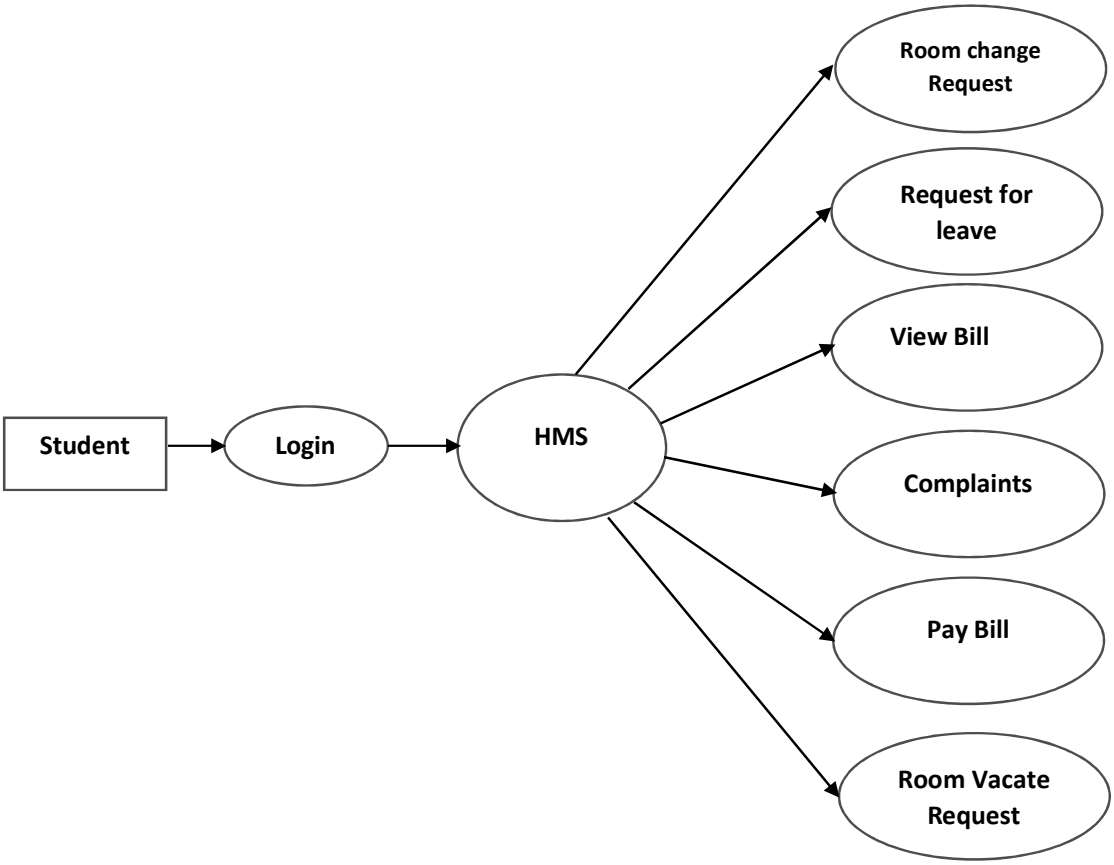
Level-1



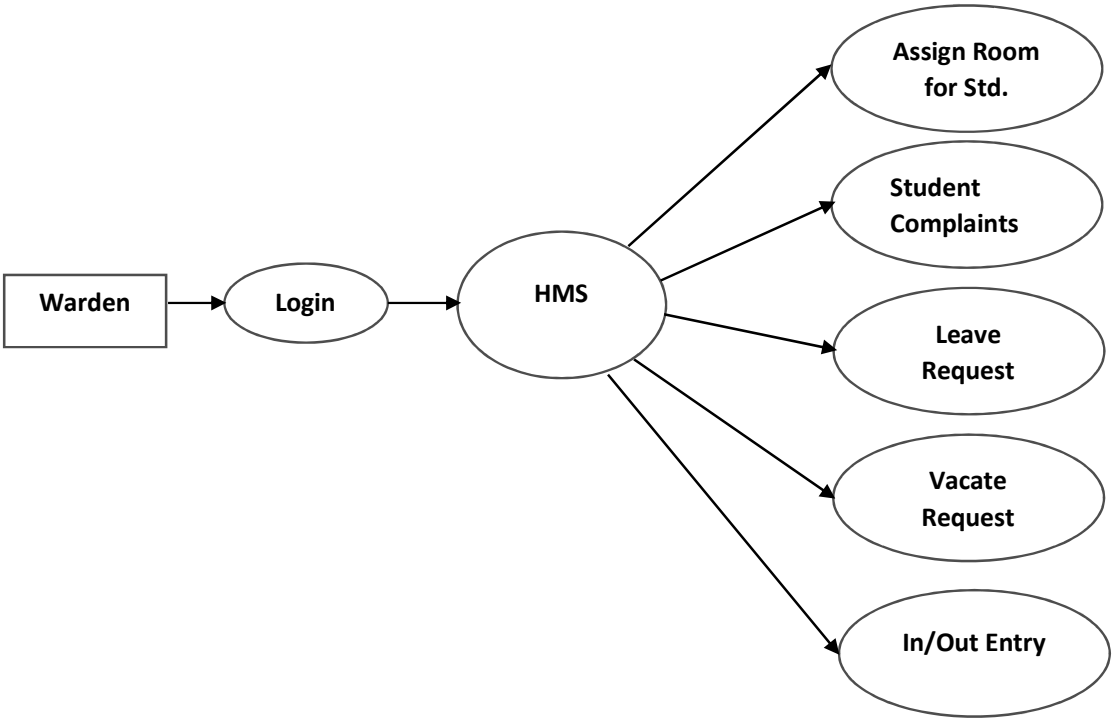
Level-1.1 (Admin)



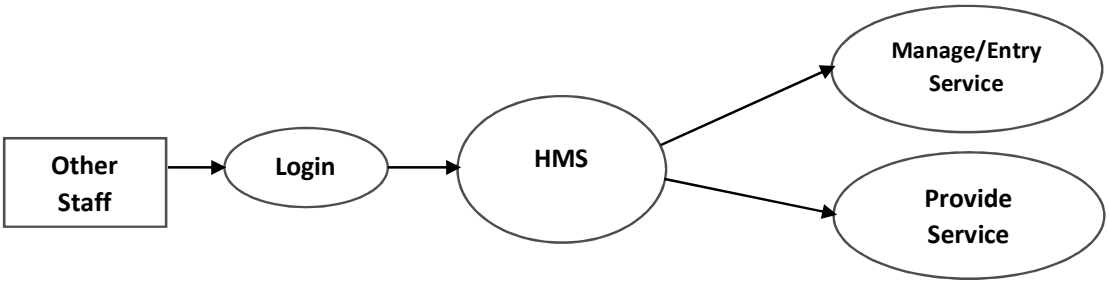
Level-1.2 (Student)



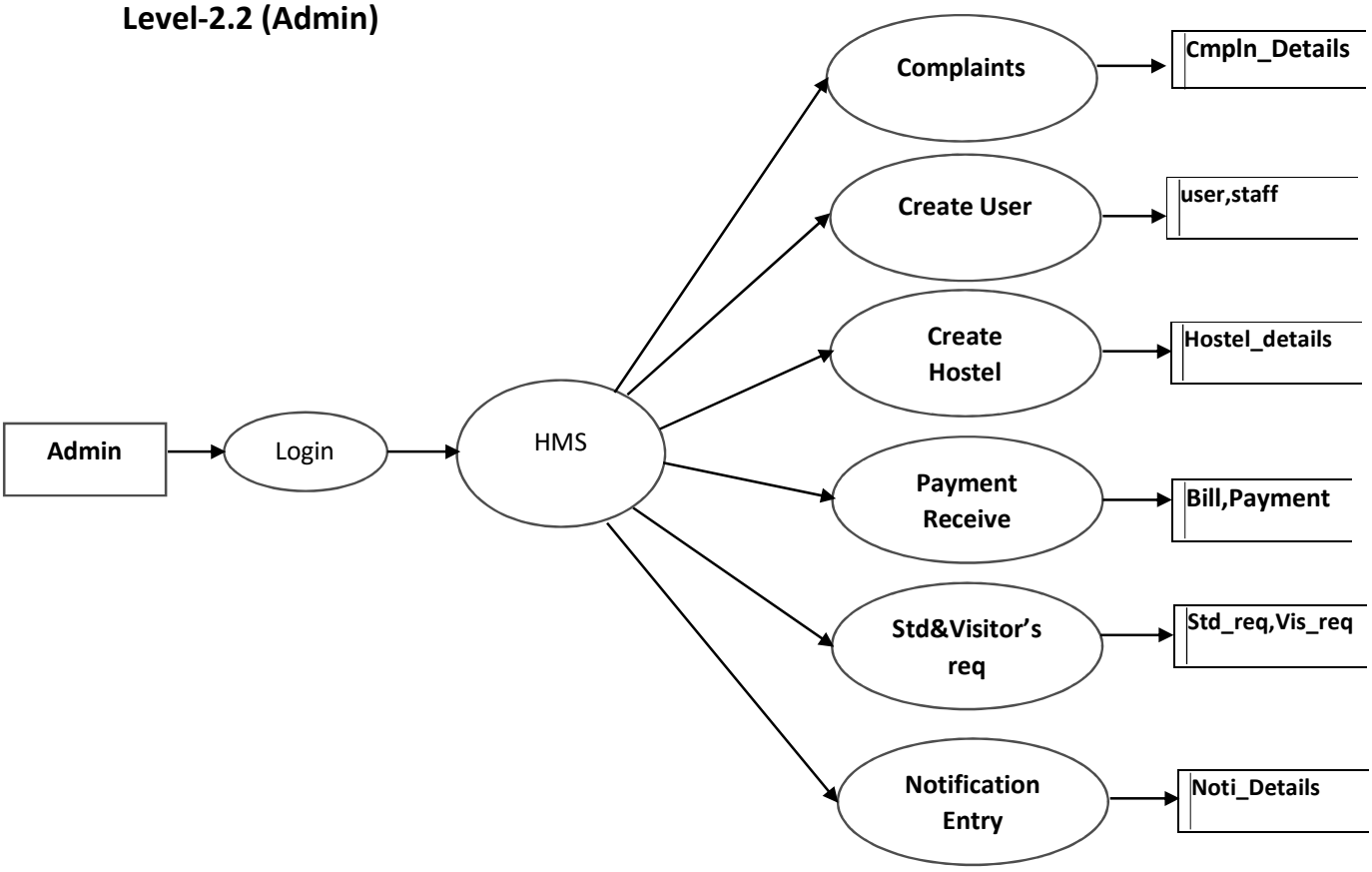
Level-1.3 (Warden)



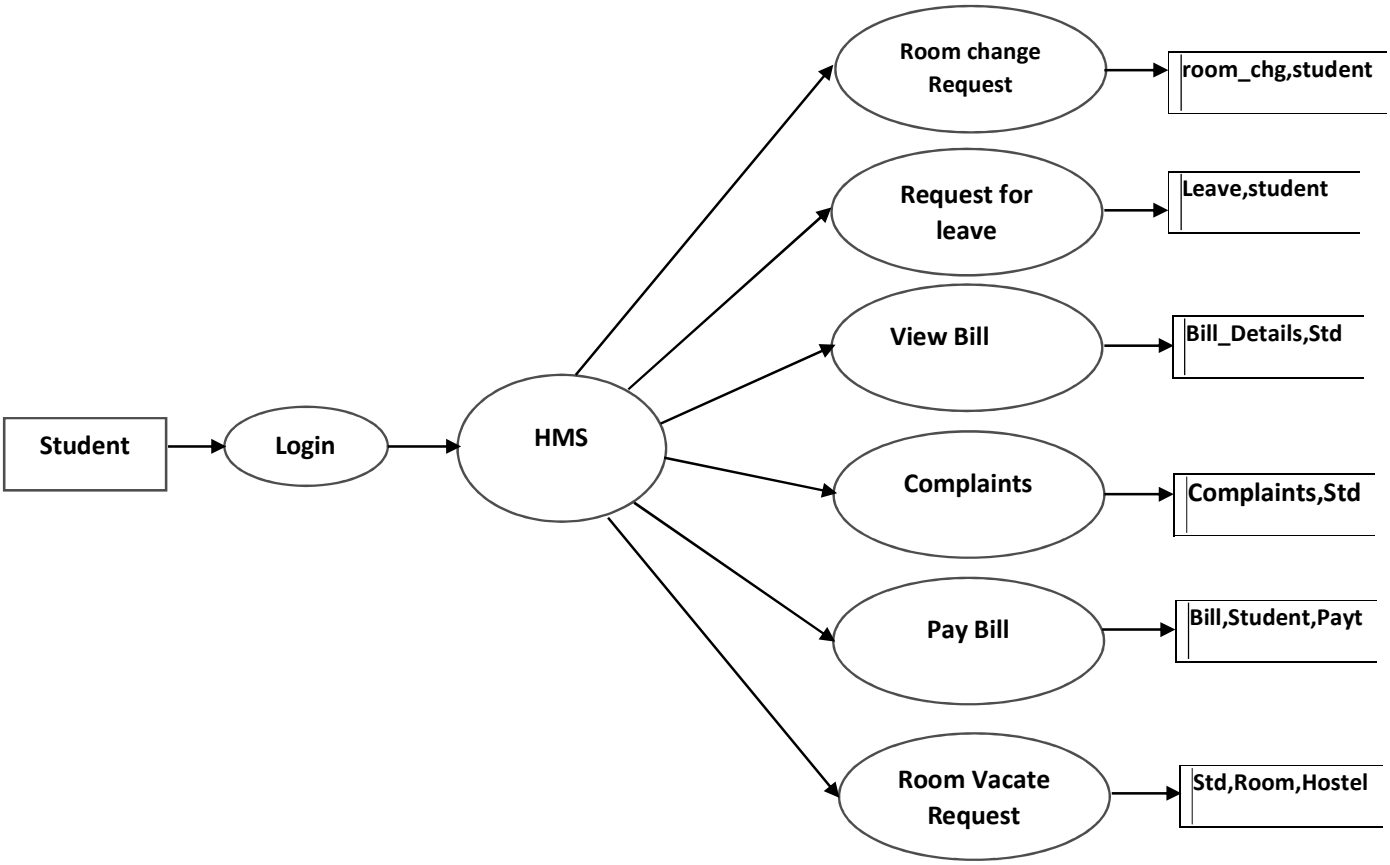
Level-1.4 (Other Staff)



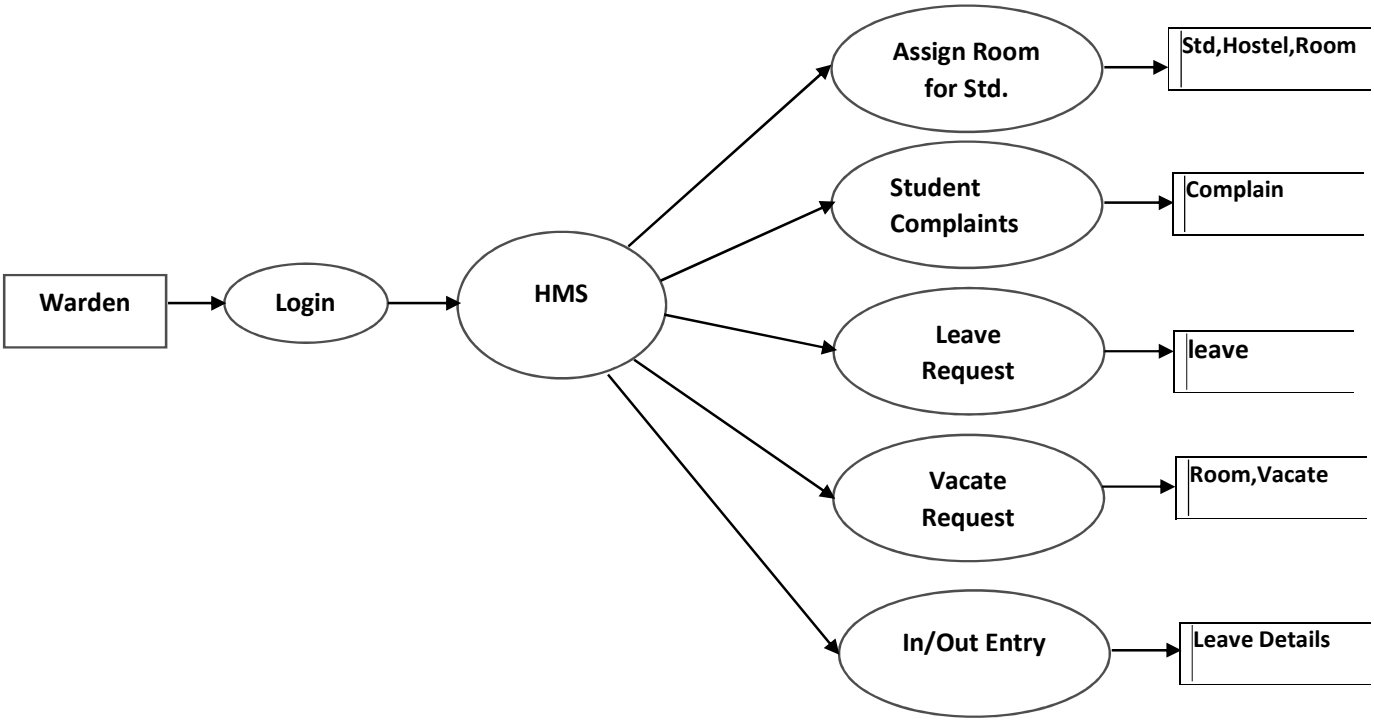
Level-2.2 (Admin)



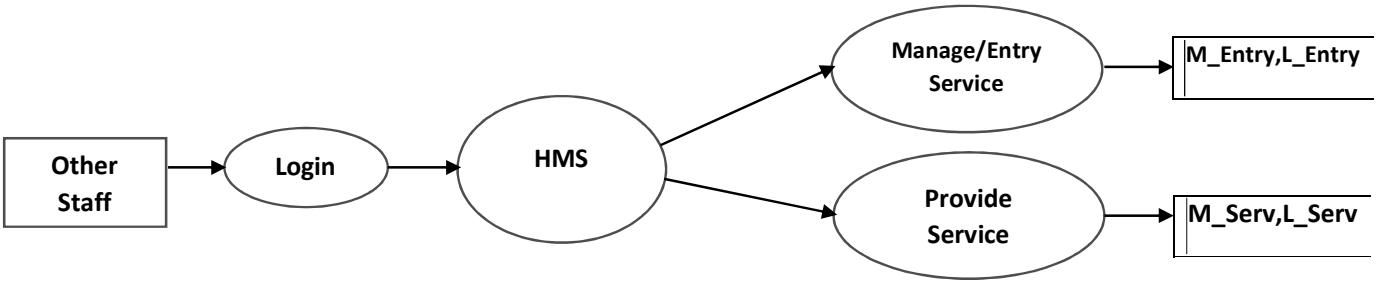
Level-2.2 (Student)



Level-2.3 (Warden)



Level-2.4 (Other Staff)



3.3 Database Design

The overall objective in the development of database technology has been to treat data as an organizational resource and as an integrated whole. Database Management System allows data to be protected and organized separately from other resources. Database is an integrated collection of data. This is the difference between logical and physical data.

The organization of data in the database aims to achieve three major objectives:

- Data integration
- Data integrity
- Data independence.

Normalization

The process of normalization is concerned with the transformation of the conceptual schema to a computer representable form. Normalization reduces the redundancies and anomalies. Redundant data wastes disk space and creates maintenance space. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. There are a few rules for database normalization. It is process that helps analysts or database designers to design table structures for an application. The focus of normalization is to reduce table data be very minimum. By this process RDBMS schema designers try their best to reduce table data to the very minimum. It is essential to remember that redundant data cannot be reduced to zero in any database management system. Each rule is called a "Normal Form".

The First Normal Form:

First normal form does not allow multivalued and composite valued attributes. It states that the domain of an attribute must include only atomic values and that value of any attribute in a tuple must be single value from the domain of that attribute.

The Second Normal Form:

In second normal form, for relations where primary key contains multiple attributes, non-key attributes should not be functionally dependent on a part of the primary key.

TABLES

Table Name: **bill_factor**

Purpose: **Storing Billing parameter of different hostels**

Field Name	Data Type	Constrains
id	int(30)	Primary Key
hostelid	bigint(30)	FK from hostel_details
roomrent	bigint(30)	NOT NULL
watercharge	bigint(30)	NOT NULL
electricitycharge	bigint(30)	NOT NULL
maintinancecharge	bigint(30)	NOT NULL
misccharge	bigint(30)	NOT NULL

Table Name: **contact_us**

Purpose: **Storing details of public contact who want to contact to administration**

Field Name	Data Type	Constrains
messageid	bigint(20)	Primary Key
sendername	varchar(50)	NOT NULL
senderemailid	varchar(50)	NOT NULL
contactno	varchar(10)	NOT NULL
subject	varchar(255)	NOT NULL
message	mediumtext	NOT NULL
status	varchar(20)	NOT NULL

Table Name: **hostel_details**
Purpose: **Storing all the hostels under college**

Field Name	Data Type	Constrains
hostel_id	bigint(30)	Primary Key
Hostel_Name	varchar(255)	NOT NULL
Hostel_AddressLine1	varchar(255)	NOT NULL
Hostel_AddressLine2	varchar(255)	NOT NULL
District	varchar(50)	NOT NULL
State	varchar(50)	NOT NULL
Pincode	int(6)	NOT NULL
Contact No	bigint(10)	NOT NULL
No_of_room	int(10)	NOT NULL
hosteltype	varchar(50)	NOT NULL

Table Name: **laundaryentry**
Purpose: **Storing details charges of different kind of laundry**

Field Name	Data Type	Constrains
laundaryid	bigint(10)	Primary Key
laundaryitem	varchar(50)	NOT NULL
price	bigint(0)	NOT NULL
description	varchar(50)	NOT NULL

Table Name: **laundary_service**
Purpose: **Storing the details of service by laundary man**

Field Name	Data Type	Constrains
serviceid	bigint(30)	Primary Key
partyid	bigint(30)	FK from student and Visitor
laundary_item	varchar(50)	FK from laundaryentry
service_date	Date	NOT NULL
exp_delidate	Date	NOT NULL
act_delidate	Date	NOT NULL
status	varchar(50)	NOT NULL

Table Name: **Mess_Entry**
Purpose: **Storing details of available food in mess and their price**

Field Name	Data Type	Constrains
foodid	bigint(10)	Primary Key
fooditem	varchar(30)	NOT NULL
price	bigint(10)	NOT NULL
description	varchar(50)	NOT NULL

Table Name: **Mess_Service**
Purpose: **Storing mess service details**

Field Name	Data Type	Constrains
mess_serviceid	bigint(20)	Primary Key
party_Id	varchar(30)	FK from student
fooditem	varchar(30)	Fk from Mess_Entry
servicedate	Date	NOT NULL

Table Name: **parents_details**
Purpose: **Storing parents details of student**

Field Name	Data Type	Constrains
parents_id	bigint(10)	Primary Key
fname	varchar(30)	NOT NULL
lname	varchar(30)	NOT NULL
emailid	varchar(30)	NOT NULL
contactno	varchar(10)	NOT NULL
Student_id	varchar(30)	FK from Student

Table Name: **publicfeed**
Purpose: **Storing public feedback from the public section**

Field Name	Data Type	Constrains
feedbackid	bigint(10)	Primary Key
feed_concern	varchar(30)	NOT NULL
hostel	varchar(30)	NULL
fname	varchar(30)	NOT NULL
lname	varchar(30)	NOT NULL
addline1	varchar(30)	NOT NULL
addline2	varchar(30)	NOT NULL
addline3	varchar(30)	NOT NULL
emailid	varchar(30)	NOT NULL
contactno	varchar(30)	NOT NULL
complain	mediumtext	NOT NULL
complain_date	date	NOT NULL

Table Name: **room_allotment**
Purpose: **Storing room allotment details of students**

Field Name	Data Type	Constrains
id	int(10)	Primary Key
hostel_id	bigint(0)	FK from hostel_details
allocatee_id	bigint(0)	FK from student
room_no	int(10)	NOT NULL
from_date	Date	NOT NULL
to_date	Date	NOT NULL
status	varchar(50)	NOT NULL

Table Name: **staff_details**
Purpose: **Storing Staff details under hostels**

Field Name	Data Type	Constrains
Staff_id	bigint(30)	Primary Key
fname	varchar(50)	NOT NULL
lname	varchar(50)	NOT NULL
addressline1	varchar(50)	NOT NULL
addressline2	varchar(50)	NOT NULL
district	varchar(50)	NOT NULL
state	varchar(50)	NOT NULL
pincode	int(6)	NOT NULL
mobileno	bigint(10)	NOT NULL
emailid	varchar(50)	NOT NULL
stafftype	varchar(15)	NOT NULL
hostelid	bigint(30)	FK from hostel_details

Table Name: **stdrequest**
Purpose: **Storing student accommodation request**

Field Name	Data Type	Constrains
requestno	varchar(30)	Primary Key
stdudent_id	bigint(30)	FK from Student
req_date	Date	NOT NULL
req_status	varchar(30)	NOT NULL
hostelid	varchar(30)	FK from hostel_details
rejection	varchar(255)	NOT NULL

Table Name: **Student**
Purpose: **Storing student details**

Field Name	Data Type	Constrains
Student_id	bigint(30)	Primary Key
admnregno	varchar(15)	NOT NULL
admnregdate	Date	NOT NULL
branch	varchar(10)	NOT NULL
semester	varchar(5)	NOT NULL
fname	varchar(30)	NOT NULL
lname	varchar(30)	NOT NULL
dob	Date	NOT NULL
gender	varchar(30)	NOT NULL
emailid	varchar(30)	NOT NULL
mobno	varchar(10)	NOT NULL
addressline1	varchar(30)	NOT NULL
addressline2	varchar(30)	NOT NULL
state	varchar(30)	NOT NULL
district	varchar(30)	NOT NULL
pincode	varchar(6)	NOT NULL
caste	varchar(30)	NOT NULL
religon	varchar(30)	NOT NULL

Table Name: **std_inout_entry**
Purpose: **Storing student In/Out time by warden**

Field Name	Data Type	Constrains
id	int(10)	Primary Key
student_id	bigint(30)	FK from student
out_date	date	NOT NULL
out_time	varchar(20)	NOT NULL
in_date	date	NOT NULL
in_time	varchar(20)	NOT NULL
status	varchar(30)	NOT NULL

Table Name: **student_complaints**

Purpose: **Storing student complaints which will be sent to warden and admin**

Field Name	Data Type	Constrains
complaints_no	bigint(30)	Primary Key
complaint_date	date	NOT NULL
complaints_concern	varchar(255)	NOT NULL
hostel_id	bigint(30)	FK from hostel_details
roomno	varchar(50)	NOT NULL
description	varchar(255)	NOT NULL
complaint_by	bigint(30)	FK from student
complaint_status	varchar(50)	NOT NULL
attended_date	date	NULL
action_taken	Mediumtext	NULL

Table Name: **student_hstlvacate_request**

Purpose: **Storing student hostel vacate request which will be sent to warden**

Field Name	Data Type	Constrains
requestno	bigint(30)	Primary Key
request_date	date	NOT NULL
student_id	bigint(30)	FK from student
hostel_id	bigint(30)	FK from hostel_details
roomno	int(20)	NOT NULL
vacatedate	date	NOT NULL
status	varchar(50)	NOT NULL
req_granted_date	Date	NULL

Table Name: **student_leave_request**

Purpose: **Storing student leave request which will be sent to warden**

Field Name	Data Type	Constrains
requestno	bigint(30)	Primary Key
request_date	date	NOT NULL
student_id	bigint(30)	FK from student
hostel_id	bigint(30)	FK from hostel_details
from_date	date	NOT NULL
to_date	date	NOT NULL
reason	varchar(255)	NOT NULL
status	varchar(50)	NOT NULL
req_granted_date	Date	NULL

Table Name: **student_room_change_request**
Purpose: **Storing student room change request which will be sent to warden**

Field Name	Data Type	Constrains
requestno	bigint(30)	Primary Key
request_date	date	NOT NULL
student_id	bigint(30)	FK from student
hostel_id	bigint(30)	FK from hostel_details
currentroomno	varchar(20)	NOT NULL
willingroomno	varchar(20)	NOT NULL
reason	varchar(255)	NOT NULL
status	varchar(20)	NOT NULL
req_granted_date	date	NULL

Table Name: **userdetails**
Purpose: **Storing user details of differen type for login into HMS System**

Field Name	Data Type	Constrains
userid	bigint(30)	Primary Key
username	varchar(30)	NOT NULL
password	varchar(15)	NOT NULL
user_flag	varchar(5)	NOT NULL
student_staff_id	bigint(30)	FK from student and staff_details

Table Name: **visitor_details**
Purpose: **Stroring visitors details**

Field Name	Data Type	Constrains
visitorid	bigint(10)	Primary Key
purpose	varchar(30)	NOT NULL
wardid	varchar(30)	FK from Student
fname	varchar(30)	NOT NULL
lname	varchar(30)	NOT NULL
dob	date	NOT NULL
gender	varchar(30)	NOT NULL
email	varchar(50)	NOT NULL
contactno	varchar(10)	NOT NULL
addressline1	varchar(50)	NOT NULL
addressline2	varchar(50)	NOT NULL
state	varchar(50)	NOT NULL
district	varchar(50)	NOT NULL
pincode	varchar(6)	NOT NULL

Table Name: **visitor_room**
Purpose: **Stroring visitor room allotment details**

Field Name	Data Type	Constrains
id	int(10)	Primary Key
hostelid	bigint(30)	FK from hostel_details
allocatee_id	bigint(10)	FK from visitors_details
room_no	int(10)	NOT NULL
reqgrantdate	date	NOT NULL
checkindate	date	NOT NULL
checkoutdate	date	NOT NULL
status	varchar(30)	NOT NULL

Table Name: **vstrrequest**
Purpose: **Storing visitor accomodation request**

Field Name	Data Type	Constrains
vstreqno	varchar(30)	Primary Key
visitorid	bigint(10)	FK from visitors_details
vstrreqdate	date	NOT NULL
checkindate	date	NOT NULL
checkoutdate	date	NOT NULL
vstreqstatus	varchar(30)	NOT NULL
hostelid	bigint(30)	FK from hostel_details
rejection	varchar(255)	NULL

Table Name: **vstr_inout_entry**
Purpose: **Storing visitor In/Out time who come to meet their wards**

Field Name	Data Type	Constrains
token_no	bigint(30)	Primary Key
full_name	varchar(50)	NOT NULL
student_id	bigint(30)	FK from student
in_date	date	NOT NULL
in_time	varchar(20)	NOT NULL
out_date	date	NULL
out_time	varchar(20)	NULL
status	varchar(30)	NOT NULL

3.4 Output Design

Output design generally refers to the results and information that are generated by the system. For many end-users, Output is the main reason for developing the system and the basis on which they evaluate the usefulness of application.

The objective of a system finds its shape in terms of the output. The analysis of the objective of a system leads to determination of outputs. Outputs of a system can take various forms. The most common are reports, screens displays, printed form, graphical drawing etc. the output also vary in their terms of their contents, frequency, timing and format. The users of the output, its purpose and sequence of details to be printed are all considered. The output from a system is the justification for its existence. If the outputs are inadequate in any way, the system itself is inadequate.

Output design phase of the system is concerned with the convergence of information to the end user-friendly manner. The output design should be efficient, intelligible so that the system relationship with the end user is improved and they're by enhancing the process of decision-making.

The basic requirements of output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose. Hence it is necessary to design output so that the objectives of the system are met in the best possible manner. The outputs are in the form of reports.

Designing computer output should proceed in an organized, well throughout manner; the right output element is designed so that people will find the system whether or executed. When we design an output we must identify the specific output that is needed to meet the system. The usefulness of the new system is evaluated on the basis of their output.

Once the output requirements are determined, the system designer can decide what to include in the system and how to structure it so that they require output can be produced. For the proposed software, it is necessary that the output reports be compatible in format with the existing reports. The output must be concerned to the overall performance and the system's working, as it should. It consists of developing specifications and procedures for data preparation, those steps necessary to put the inputs and the desired output, ie maximum user friendly. Proper messages and appropriate directions can control errors committed by users.

The output design is the key to the success of any system. Output is the key between the user and the sensor. The output must be concerned to the system's working, as it should.

Output design consists of displaying specifications and procedures as data presentation. User never left with the confusion as to what is happening without

appropriate error and acknowledges message being received. Even an unknown person can operate the system without knowing anything about the system.

4. System Implementation and Testing

4.1 System Implementation

4.1.1 Coding standards used

a) PHP

The Hypertext Pre-processor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases PHP started out as a small open source project that evolved as more and more people found out how useful it was. Ramus Lerdorf unleashed the first version of PHP way back in 1994.

PHP is a recursive acronym for "PHP: Hypertext Pre-processor".

- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, oracle, Sybase, Informix, and Microsoft SQL server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the UNIX side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for java and distributed object architectures (COM and COBRA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP syntax is same as C.

Common uses of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can be handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, and modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP:

Five important characteristics make PHP's practical nature possible:

- Simplicity.
- Efficiency.
- Security.
- Flexibility.
- Familiarity.

b) MYSQL

The MySQL relational database server was born almost 15 years ago out of an internal company project by employees of the Sweden-based TcX DataKonsult AB (AB is an abbreviation for Aktiebolag, which is the Swedish term for corporation). Their project, dubbed MySQL, was first released to the general public at the end of 1996. The software proved so popular that in 2001 they founded a company based entirely around MySQL-specific service and product offering, calling it MySQL AB. Profitable since its inception, MySQL AB grew by leaps and bounds, establishing offices in several countries, attaching substantial venture capital funding, and announcing numerous high-profile partnerships with an array of corporate heavyweights, including Red Hat, Veritas, Novell, and Rack space. This growth

culminated in the company's 2008 acquisition by sun microsystems, which was in turn purchased by Oracle Corporation in early 2009.

From the first public release, MySQL's developers placed particular emphasis on software performance and scalability. The result was a highly optimized product that was lacking in many features considered standard for enterprise database products: stored procedures, triggers, and transactions, for example. Yet the product caught the attention of a vast number of users who were more interested in speed and scalability that in capabilities that would, in many cases, often go unused anyway. Subsequent versions added these features anyway, which attracted even more users. MySQL is a relational database server that offers the same features found in competing proprietary products.

The MySQL allows the user to specify the database that contains:

- Tables
- Diagrams Views etc.

- Tables allow the user to create the table in a design view where he has to specify the attribute name, type, size, allow null or not.
- Diagrams allow the user to specify the foreign key constraints and also to place the relationship between the 2 tables.
- Views are virtual tables derived from another tables or from a predefined view.

The user can insert the data into the table at both the design time and through application programs. The user enters data into tables through the user interface. The MySQL Enterprise manager helps us to create and maintain tables of data.

4.1.2 Environment used

a) WAMP

Stands for "Windows, Apache, MySQL, and PHP." WAMP is a variation of LAMP for Windows systems and is often installed as a software bundle (Apache, MySQL, and PHP). It is often used for web development and internal testing, but may also be used to serve live websites.

The most important part of the WAMP package is Apache (or "Apache HTTP Server") which is used run the web server within Windows. By running a local

Apache web server on a Windows machine, a web developer can test webpages in a web browser without publishing them live on the Internet.

WAMP also includes MySQL and PHP, which are two of the most common technologies used for creating dynamic websites. MySQL is a high-speed database, while PHP is a scripting language that can be used to access data from the database. By installing these two components locally, a developer can build and test a dynamic website before publishing it to a public web server.

While Apache, MySQL, and PHP are open source components that can be installed individually, they are usually installed together. One popular package is called "WampServer," which provides a user-friendly way to install and configure the "AMP" components on Windows.

b) DREAMWEAVER

Adobe Dreamweaver is a software program for designing web pages, essentially a more fully featured HTML web and programming editor. The program provides a what-you-see-is-what-you-get (WYSIWYG) interface for users to create and edit web pages in a more user-friendly environment. Dreamweaver supports multiple mark-up languages including HTML and Extensible Mark-up Language (XML), style sheet languages like Cascading Style Sheets (CSS), and programming languages including JavaScript, C#, Visual Basic (VB), Active Server Pages (ASP), and others. The program is also available in a number of languages, including English, Spanish, French, German, Japanese, Chinese (both Simplified and Traditional), Italian, Russian, and more.

Dreamweaver was originally developed and published by Macromedia in 1997. Adobe purchased Macromedia (which included the rights to Dreamweaver) in 2005 and continued the development of the program. The many features of Dreamweaver make it a versatile web editing tool, where it be for creating complex or very simple's sites.

Features

Adobe Dreamweaver CC is a web design and development application that combines a visual design surface known as Live View and a code editor with standard features such as syntax highlighting, code completion, and code collapsing as well as more sophisticated features such as real-time syntax checking and code introspection for generating code hints to assist the user in writing code. Combined with an array of site management tools, Dreamweaver lets its users design, code and manage websites as well as mobile content. Dreamweaver is positioned as a versatile web design and development tool that enables visualization of web content while coding.

Dreamweaver, like other HTML editors, edits files locally then uploads them to the remote web server using FTP, SFTP, or WebDAV. Dreamweaver CS4 now supports the Subversion (SVN) version control system.

4.2.3 Hardware/software used for implementation

SOFTWARE SPECIFICATION

Front End

Operating system : Windows 7

IDE : Dreamweaver

Language : PHP

Back End

Database : My SQL

HARDWARE SPECIFICATION

Processor	:	Pentium IV or Above.
Main Memory	:	1 GB RAM.
Cache Memory	:	512 KB.
CPU Speed	:	800 MHz
Hard Disk Capacity	:	20 GB or Above.
CD ROM Drive	:	52 X
Mouse	:	Standard (Microsoft compactable)
Monitor	:	Display Panel (1024*764)

4.2 System Testing

Testing Objectives

Software testing is an important element of Software Quality Assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and the costs associated with a software failure are motivating forces for well planned, through testing.

There are several rules that can serve as testing objectives,

1. Testing is a process of executing a program with the intent of finding errors.
2. A good test case is the one that has a high probability of finding an undiscovered error.
3. A successful test is the one that uncovers an undiscovered error.

System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computing based system. Although each test has a different purpose, all verify that all system elements have been properly integrated and perform allocated functions.

During testing, we tried to make sure that the product does exactly what is supposed to do. Testing is the final verification and validation activity within the

organization itself. In the testing stage, i tried to achieve the following goals; to affirm the quality of the product, to find and eliminate any residual errors from previous stages, to validate the software as a solution to the original problem, to demonstrate the presence of all specified functionality in the product, to estimate the operational reliability of the system. During testing the major activities are concentrated on the examination and modification of the source code.

Testing Methodologies

The following are the testing methodologies:

1) **Unit testing**

Unit testing begins at the vortex of the spiral and concentrates on each unit (e.g. component, class or web app content object) of the software as implemented in source code. Unit testing focuses verification effort on the smallest unit of software design- the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of a component.

2) **Integration testing**

Integration testing is a systematic technique for constructing the software architecture while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit-tested components and build a program structure that has been dictated by design.

The following are the types of integration testing:

Top-Down Integration

Top-Down integration testing is an incremental approach to construction of the software architecture. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main program). Modules

subordinate (and ultimately subordinate) to the main control module are incorporated into the structure in either a depth - first or breadth - first manner.

The integration process is performed in a series of 5 steps:

- The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.
- Depending on the integration approach selected (i.e. depth or breadth first), subordinate steps are replaced one at a time with actual components.
- Tests are conducted as each component is integrated.
- On completion of each set of tests, another stub is replaced with the real component.
- Regression testing (discussed late in this section) may be conducted to ensure that new errors have not been introduced.

The process continues from step 2 until the entire program structure is built.

Bottom-up integration

Bottom-up integration testing, as its name implies, begin construction and testing with automatic modules (i.e., components at the lowest levels in the program structure).Because components are integrated from the bottom up, the functionality provided by components subordinate to a given level is always available and the need for stubs is eliminated. A Bottom-up integration strategy may be implemented with following steps:

- Low-level components are combined in to clusters(sometimes called builders) that perform a specific software sub function
- A driver(a control program for testing) is written to co-ordinate test case input and output
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure.

3) User Acceptance Testing

Acceptance testing that is conducted by the customer in an effort to exercise all required features and functions. When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements.

4) Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. The outputs generated or displayed by the system under consideration are tested by asking the users about the format required by them. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

5) Validation Testing

Validation checks are performed on the following fields.

Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested. A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

The above testing is done by taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

1. Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have they entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

2. Using Artificial Test Data

Artificial test data created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

5. SYSTEM SECURITY MEASURES

- Error Messages:

A measure to control error messages so that they don't give up any sensitive information. Provides minimal errors to the user to ensure no secrets in the server are leaked.

- Server side Validation or Form Validation:

Browser catches simple failures like mandatory field empty and type mismatching. Server does deeper validation to detect and prevent undesirable results in the website and malicious codes being inserted to the database.

- Passwords:

Using strong password for the server and website admin area, also insist on good password practice for the users to protect the security of their account.

- File Uploads:

Measures are taken to ensure the uploaded files are not executed in the server. They include renaming the file to ensure the extensions since image extension are not executed on servers, and changing the file permissions. Ultimately the direct access to the uploaded files is restricted. Additionally firewall setup blocks all non-essential ports.

6. Conclusion

To conclude the description about the project: The project, developed using PHP and MySQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement.

The expanded functionality of today's software requires an appropriate approach towards software development. This hostel management software is designed for people who want to manage various activities in the hostel. For the past few years the number of educational institutions are increasing rapidly. Thereby the number of hostels are also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hostel and software's are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

7. Scope for Future Enhancement

Upon evaluation of the project with the customer representative, the team saw the potential that this product could bring in terms of improving business revenues and heightening the strategic position it had in the market. Centralized processing of hostel activities under HMS is key to timely information reporting and informed decision making in CEV Hostels. The objective of HMS is to have in place state-of-the-art IT enabled resource Information Management System, which will meet the requirement of College of Engineering, Vatakara Hostel Department. In due course, HMS shall also facilitate accrual based information regarding hostels and students residing under College of Engineering, Vatakara.

The Hostel Management System allows the client to automate calculations and improve the quality of services. It is developed in such a way that there is flexibility to add programs to overcome the limitations such as to increase the number of outputs or variable information. The file accessing methods can be made faster and efficient.

Although time constraint places a boundary on the functionality of the final product to focus of meeting the original needs of the intention of this system, the team uncovered many potential upgrades to the system to assist and make their work easiest and efficient. Example of such includes:

- ✓ To introduce online banking to pay bills in the next version.
- ✓ Full fledge management of account of the hostel.
- ✓ Automated SMS and Gmail notification.
- ✓ Generation of specific report based on the requirement of the hostels

Bibliography

BOOKS:-

[1] Roger S. Pressman. —Software Engineering A Practitioner‘s Approach‖ 6th Edition, McGraw Hill, 2005

[2] Rasmus Leadoff, Kevin Tat roe, Peter Macintyre. —Programming PHP‖ 2nd Edition, O‘REILLY 2006

[3] Kevin Yank. —Build Your Own Database Driven Website Using PHP & MySQL‖3rd Edition Paperback, 2001

[4] Rajib Mall, Fundamentals of Software Engineering‖ Eastern Econ omy Edition, 1998

[5] Pankaj Jalote, —An Integrated Approach to Software Engineering‖, 2nd Edition, Narosa Publishing House, 1995

[6] Christian Darie, Bogdan Brinzarea, —Building Responsive Web Applications‖ First Edition, PACKT Publishing,2008

[7] Chris Charuhas, —Managing Web Projects‖ 2003 Edition, Firewall media.

Web Resources:-

[1] http://www.stripesframework.org

[2] http://dev.mysql.com/doc/refman/5.1/en

[3] http://www.w3schools.com/PHP

[4] mailto: http://in.php.net/tut.PHP

[5] mailto: http://www.freewebmasterhelp.com/PHP

[6] mailto:http://www.w3schools.com/JavaScript

Appendix

Use Case Diagram:

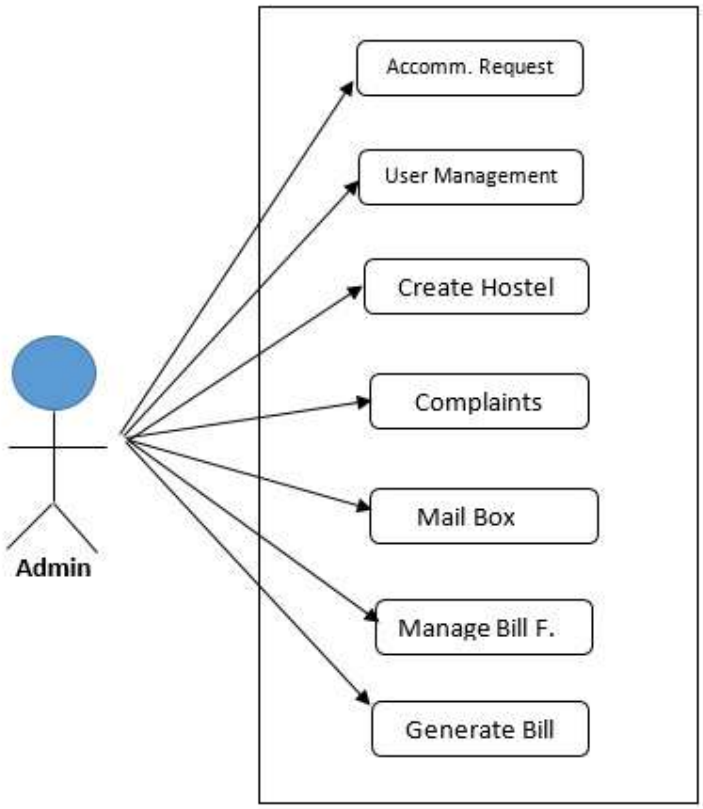


Fig: Use Case Diagram for Admin

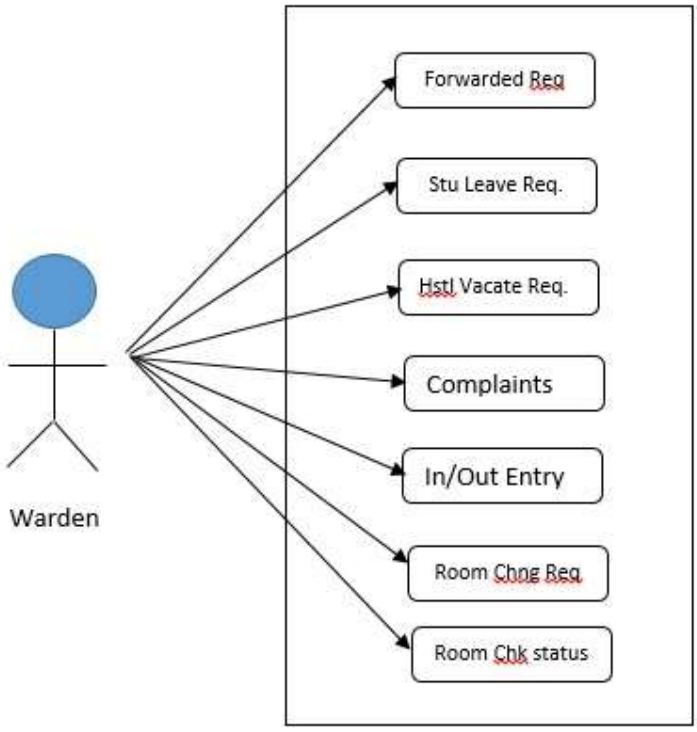


Fig: Use Case Diagram for Warden

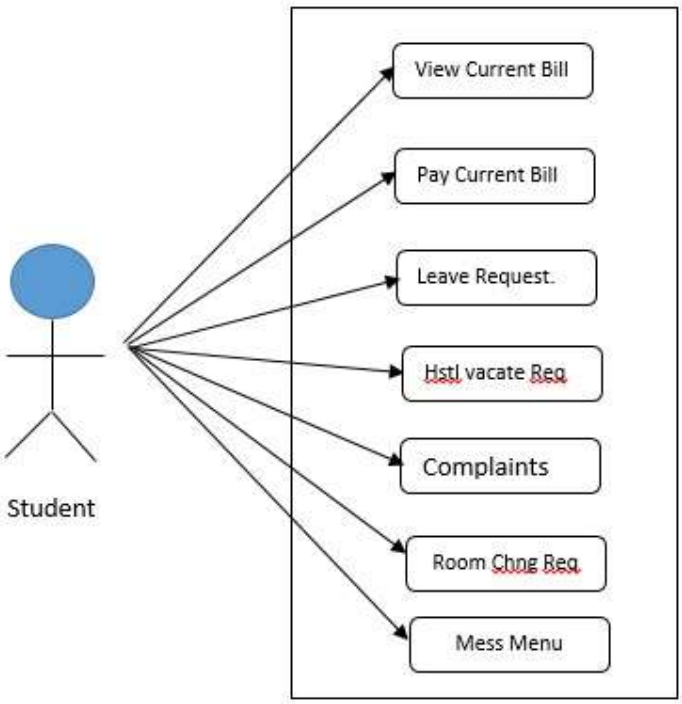


Fig: Use Case Diagram for Student

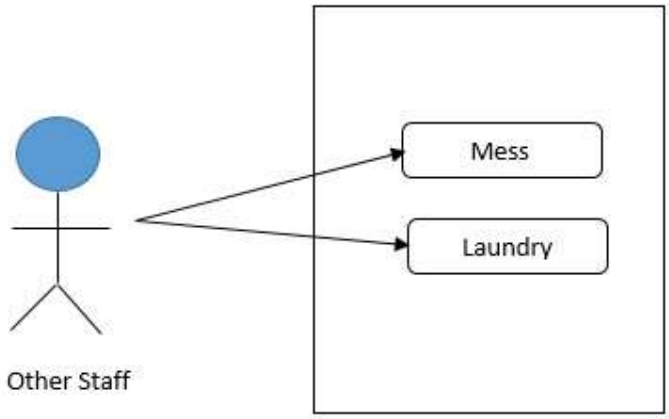
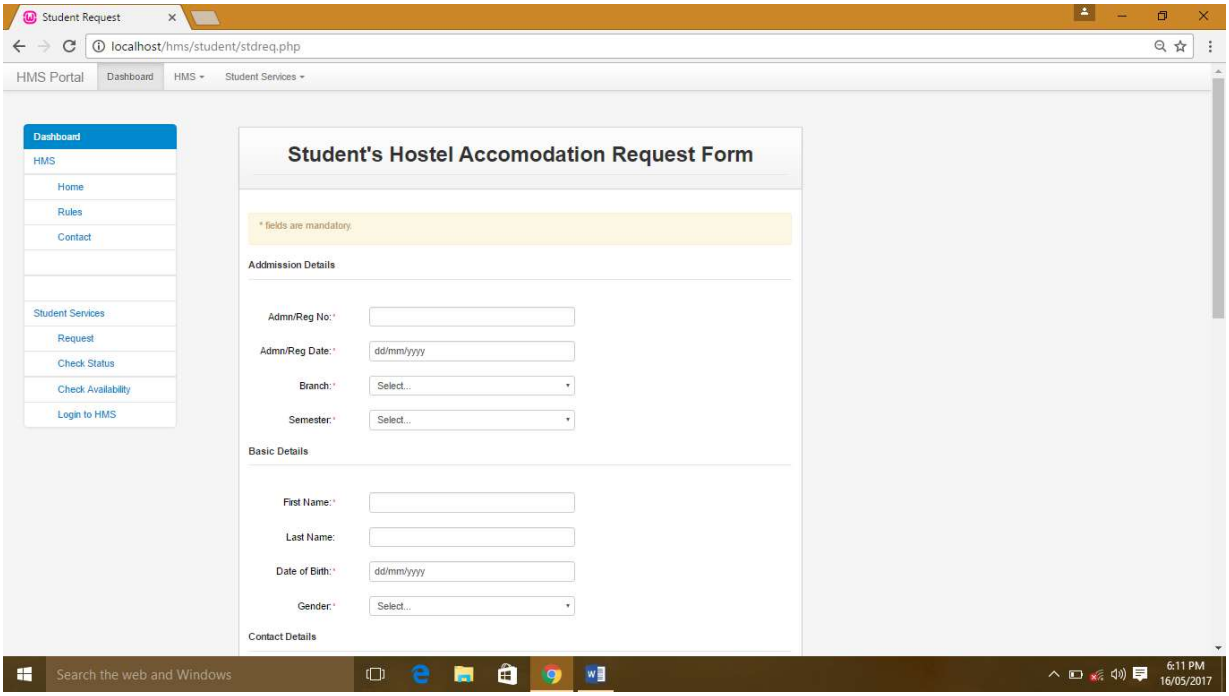
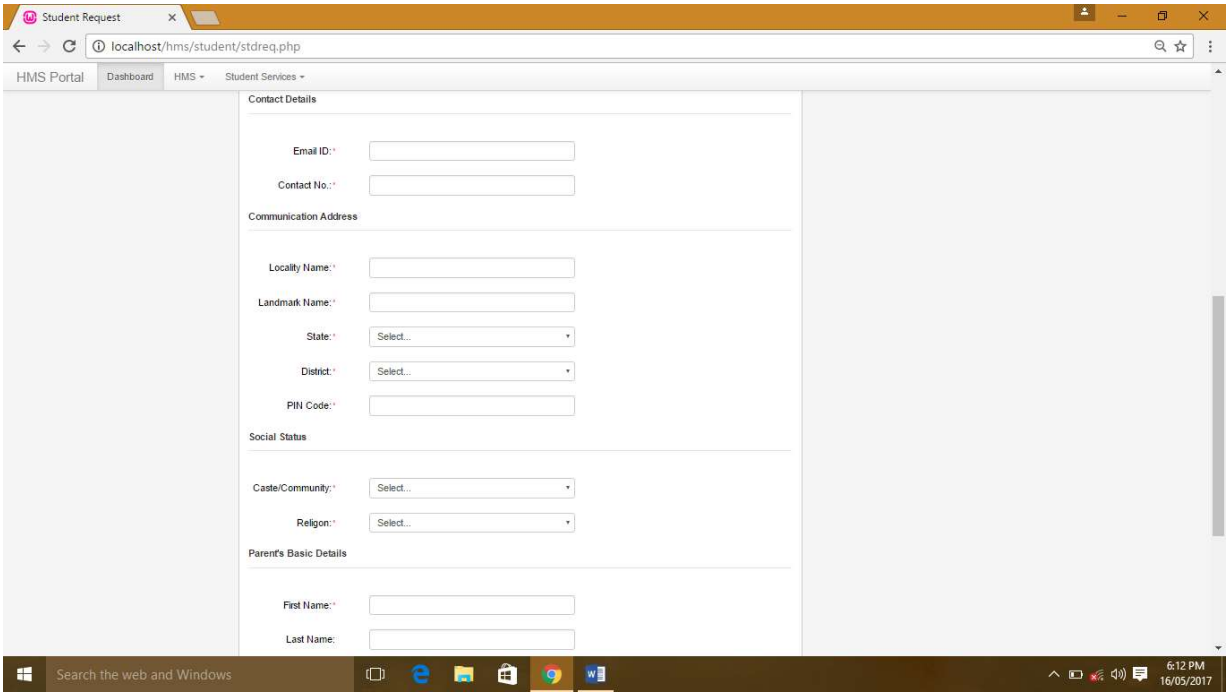


Fig: Use Case Diagram for Other Staff

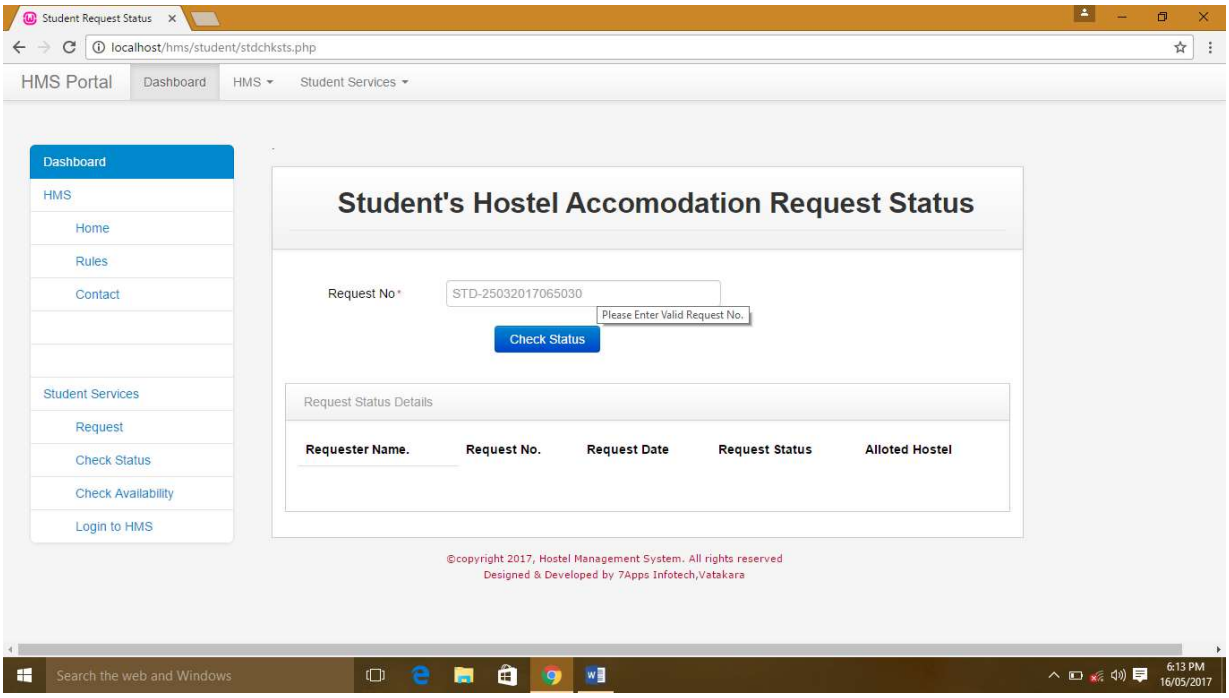
Screenshots:

Student Hostel Accommodation Form:

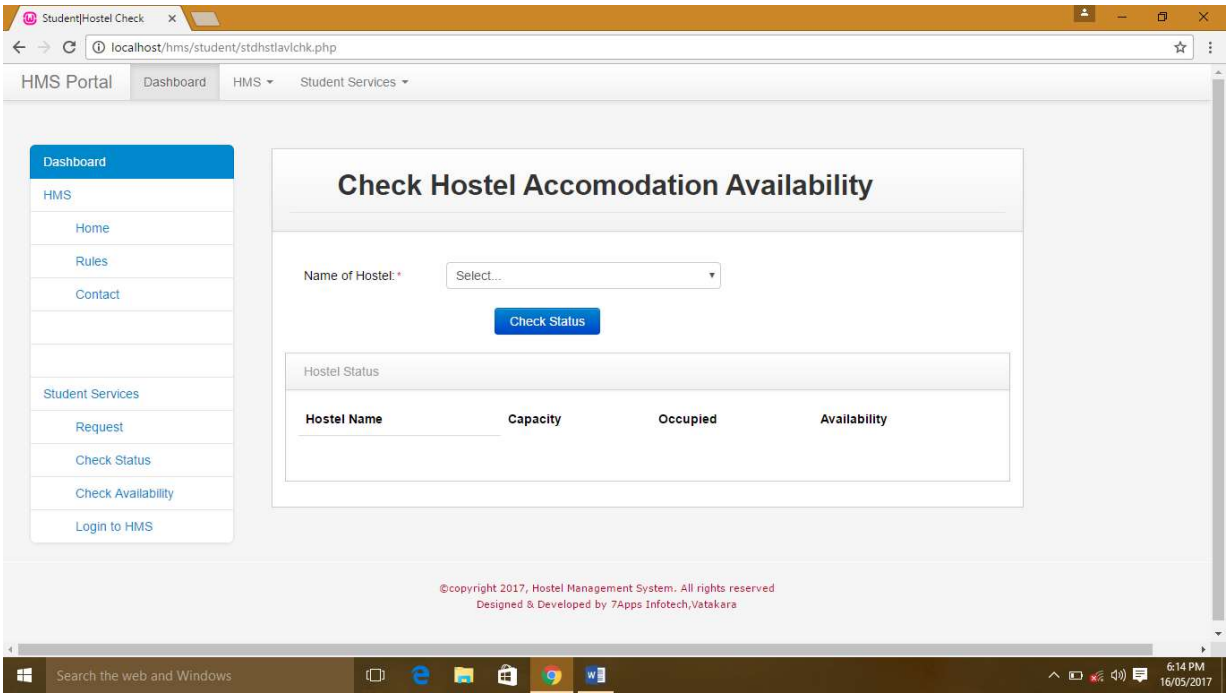




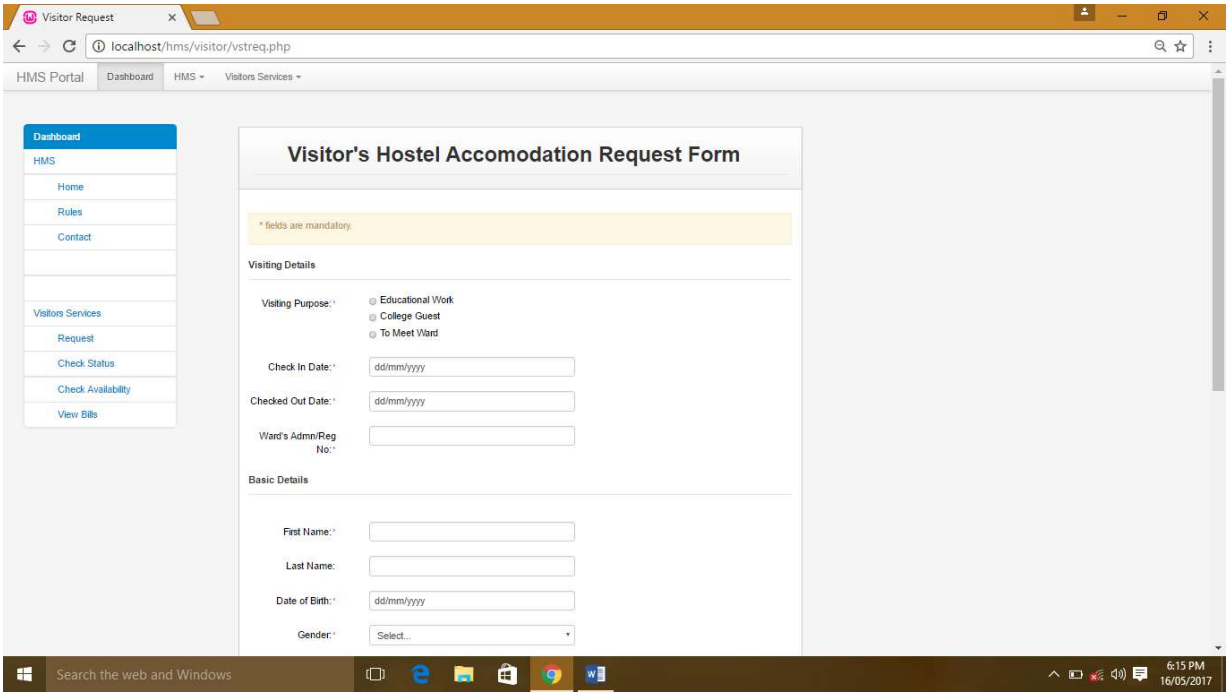
Student Hostel Accommodation Request Check Status:

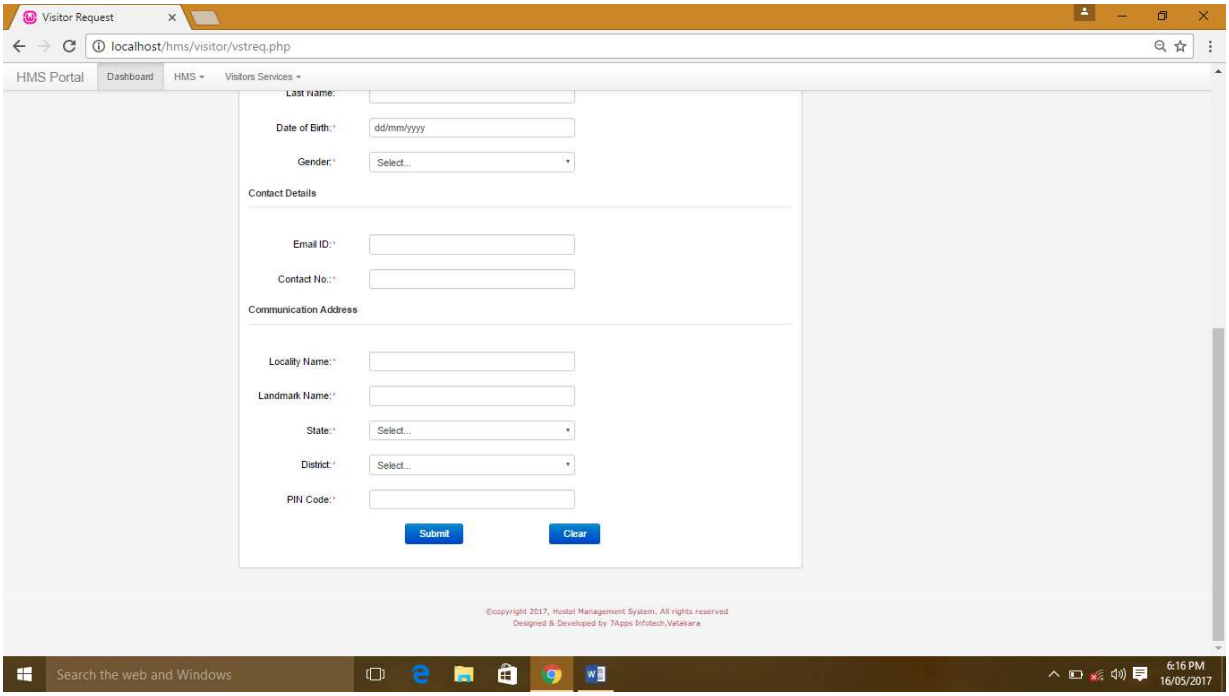


Check Hostel Availability:

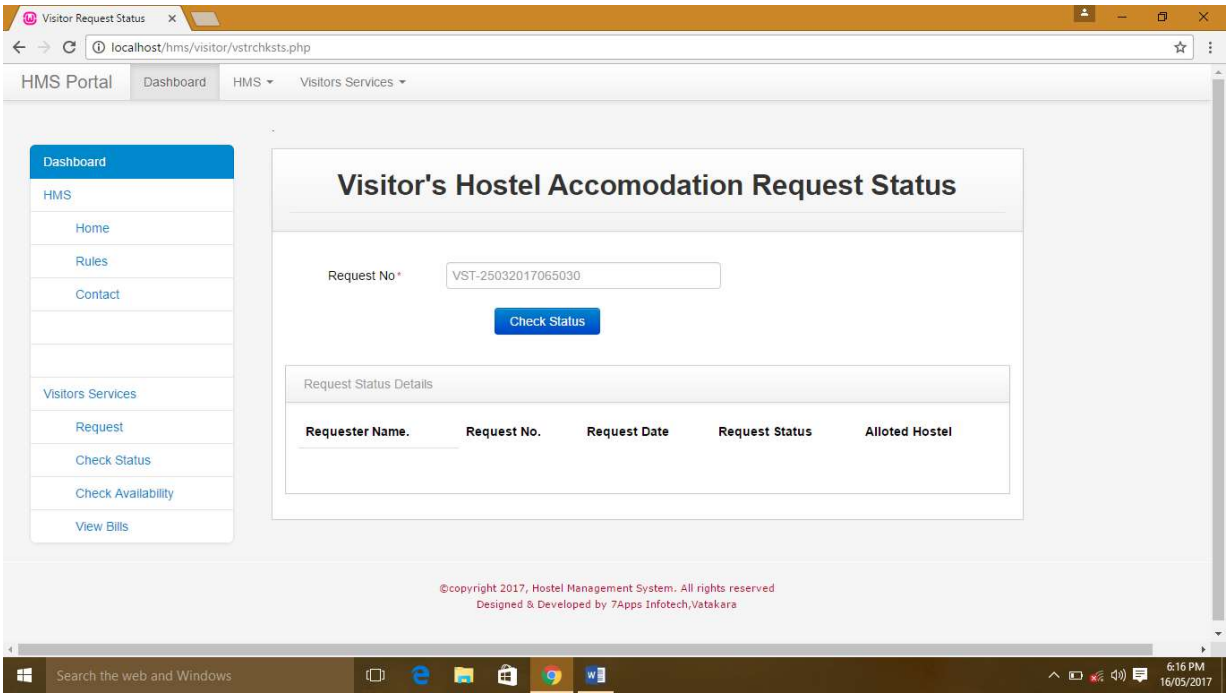


Visitor's Hostel Accommodation Request Form:

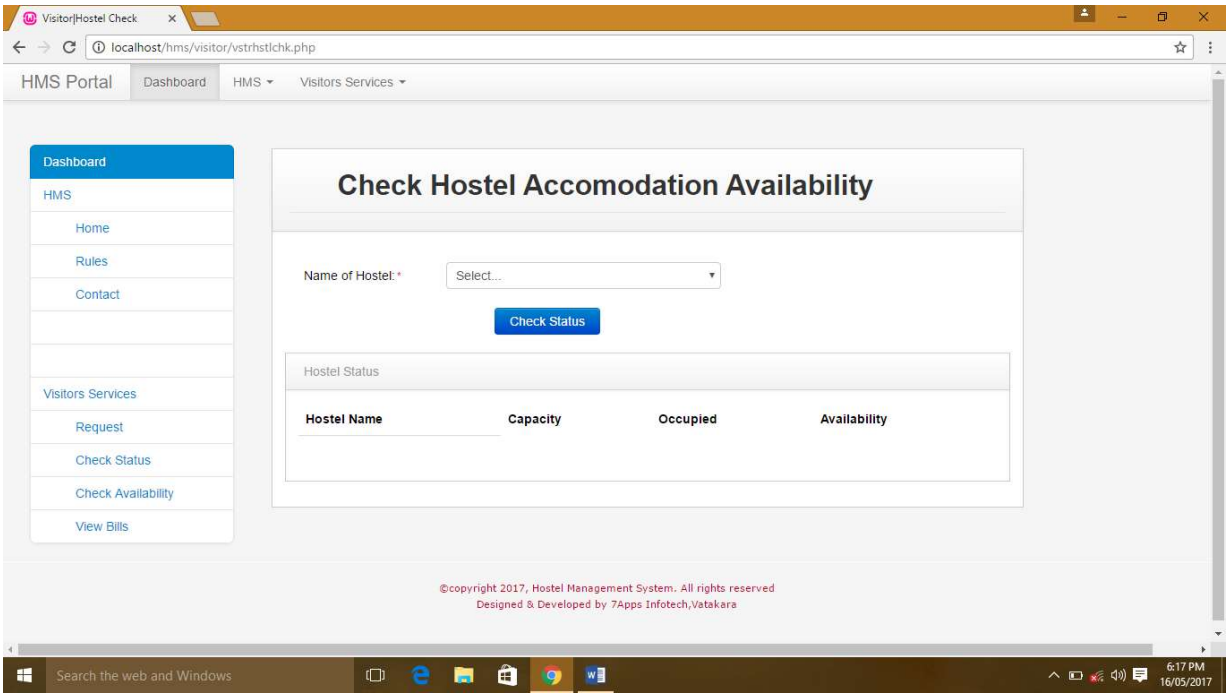




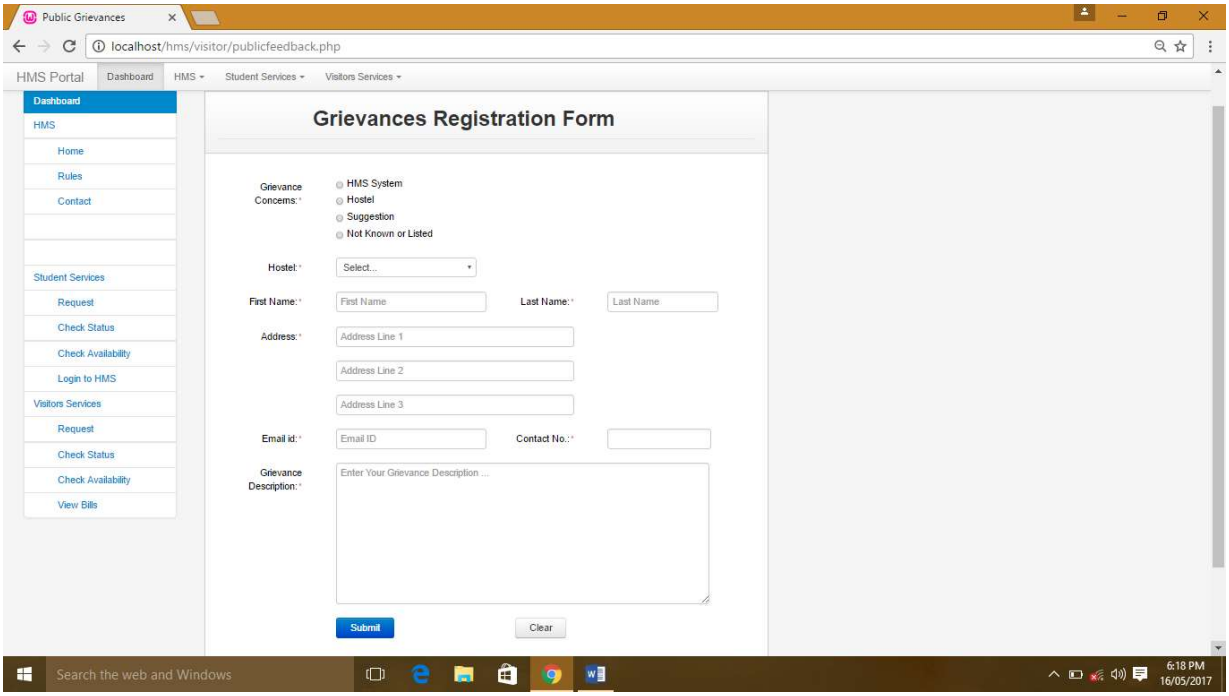
Visitor's Accommodation Request Check Status:



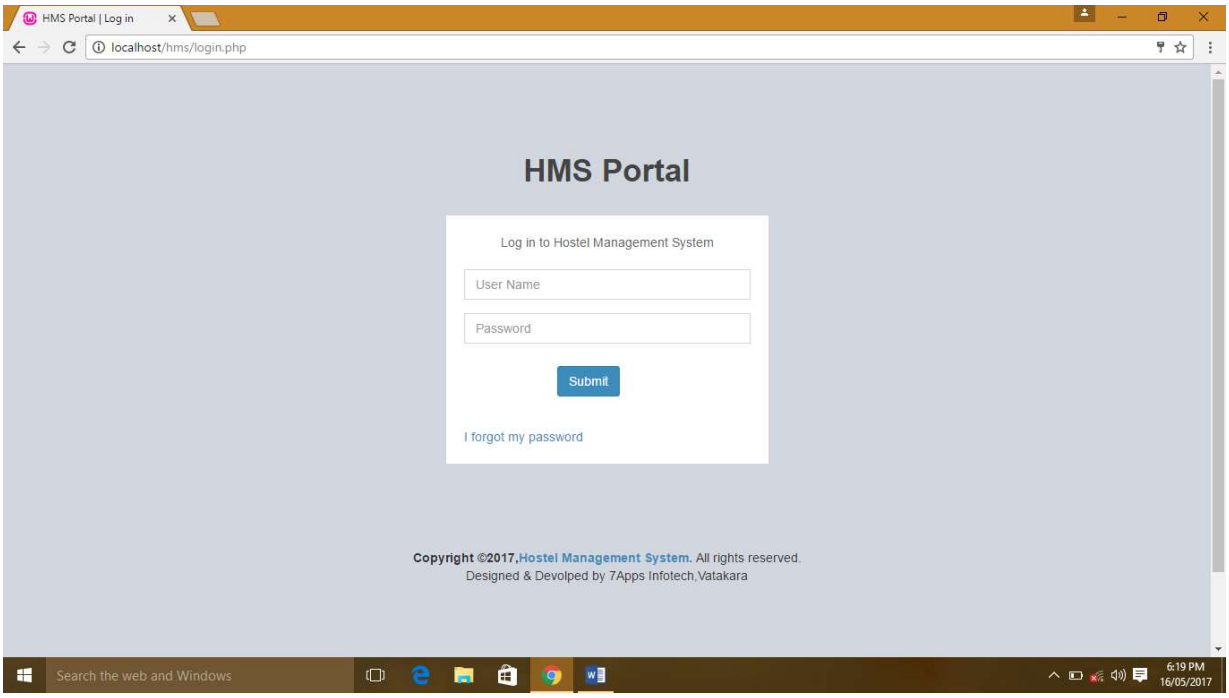
Visitor's Check Hostel Availability:



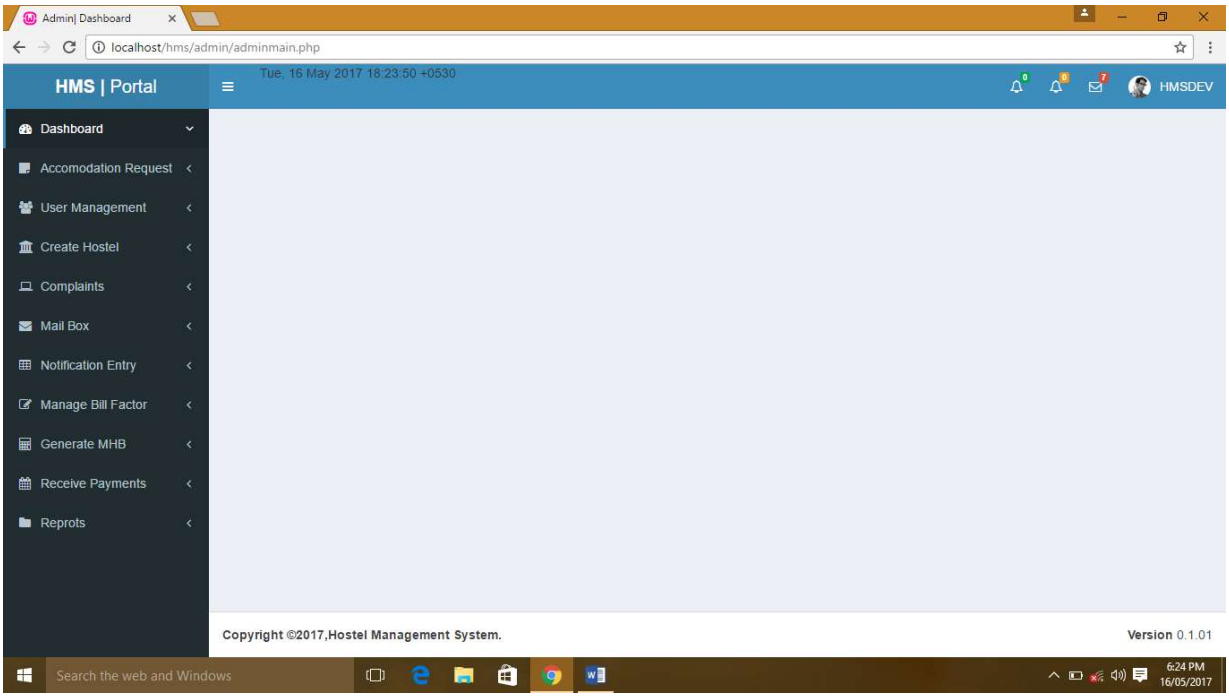
Public Feedback/Grievances Form:



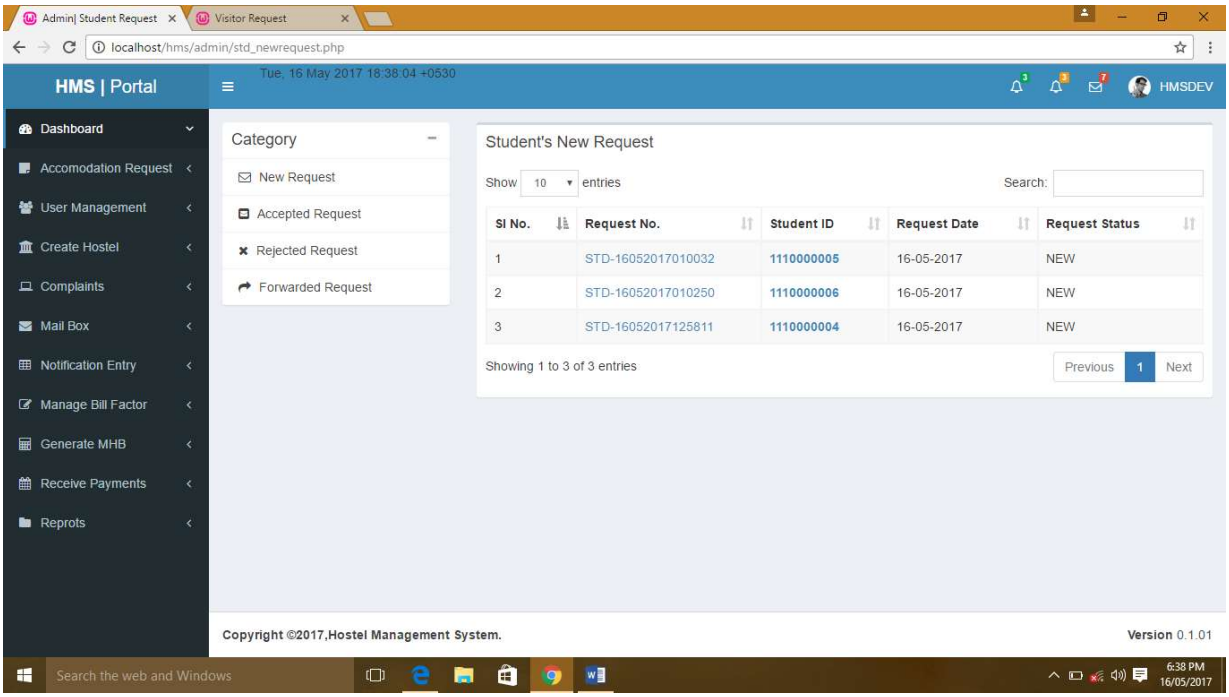
User Login:



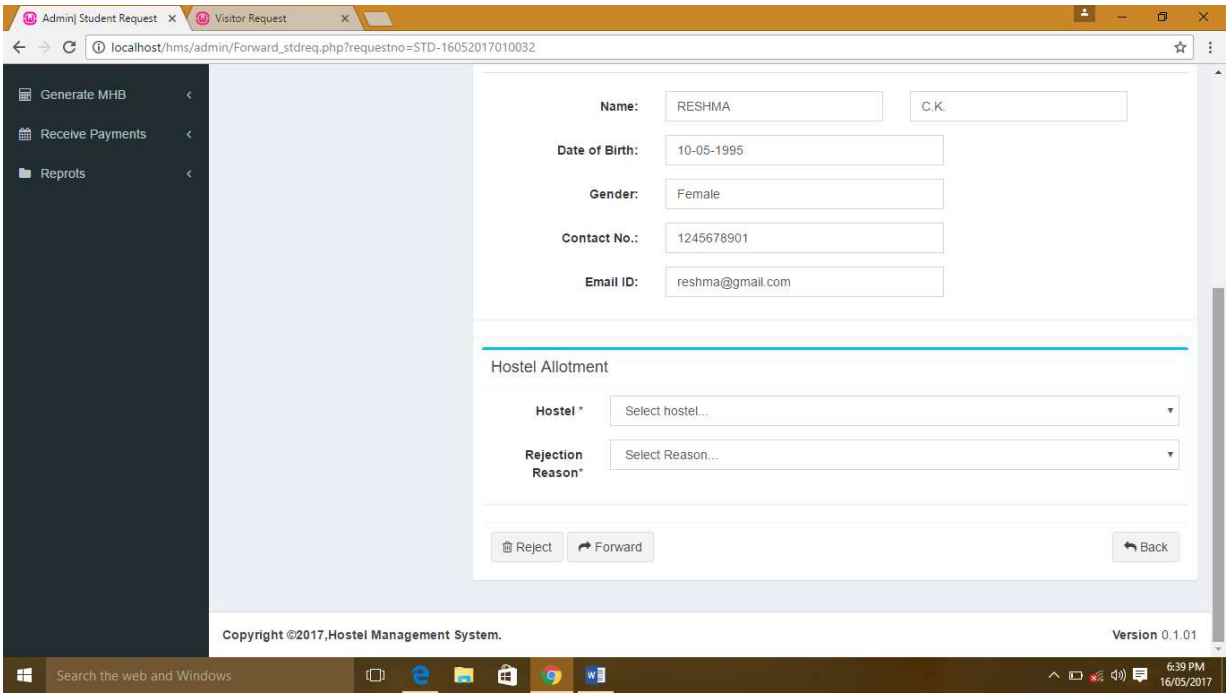
Admin Dashboard



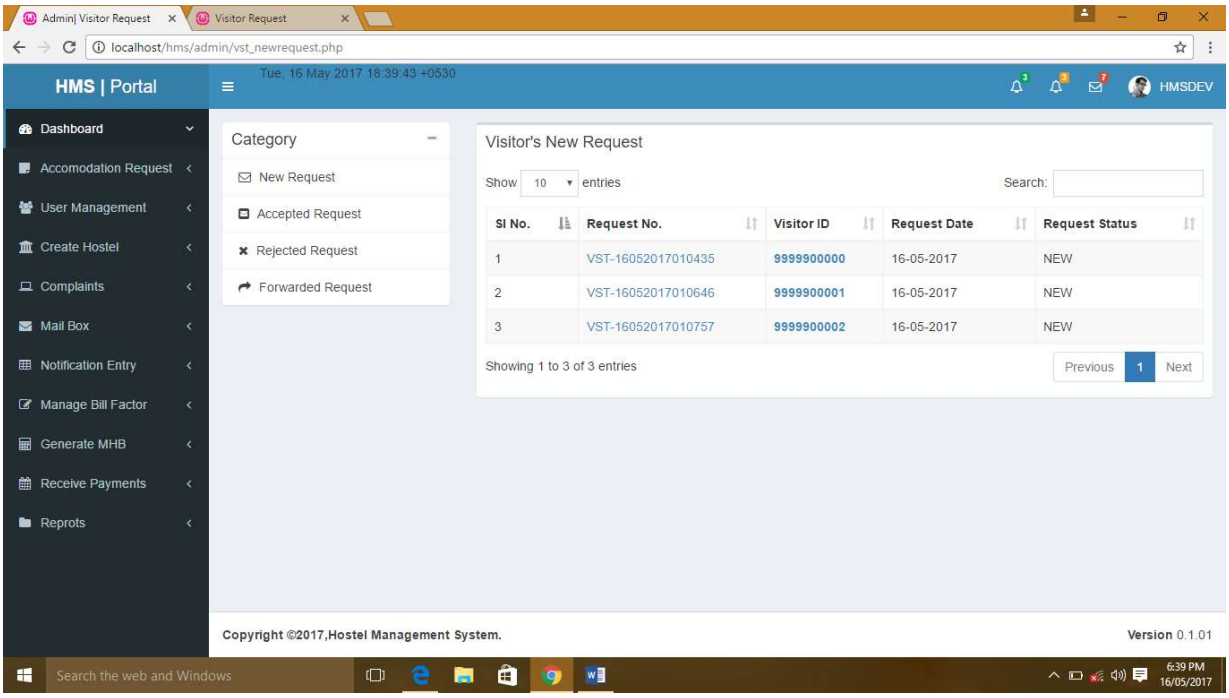
Student New Request



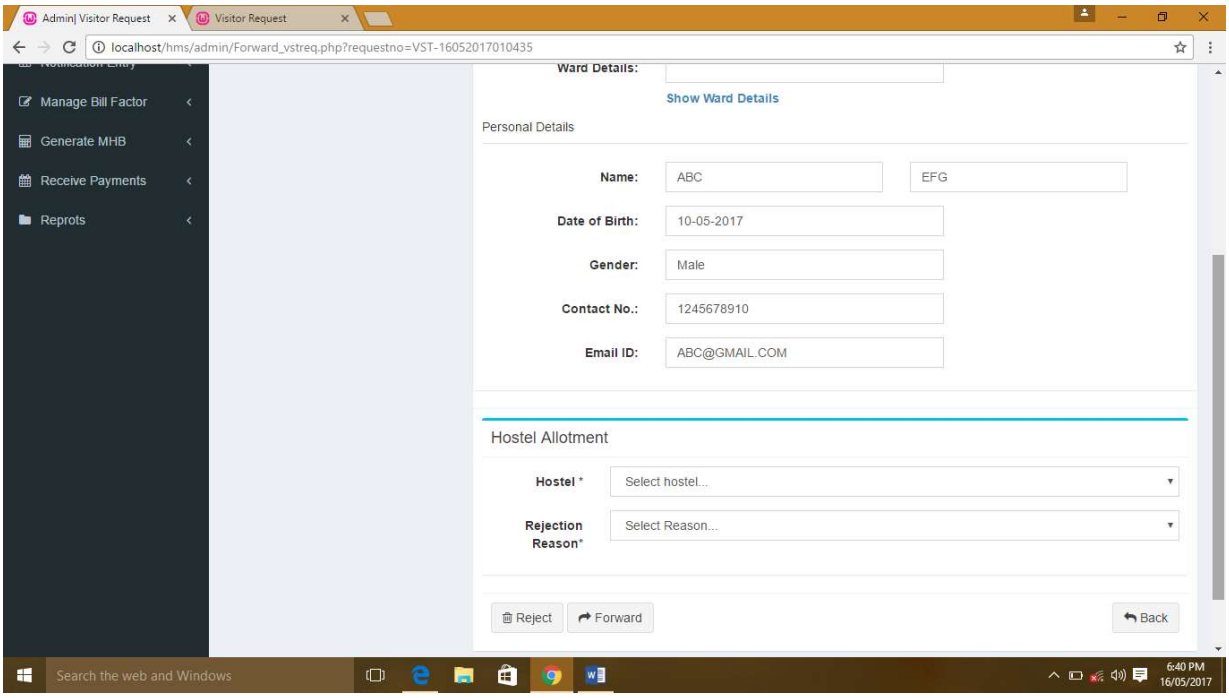
Forward Request to Warden



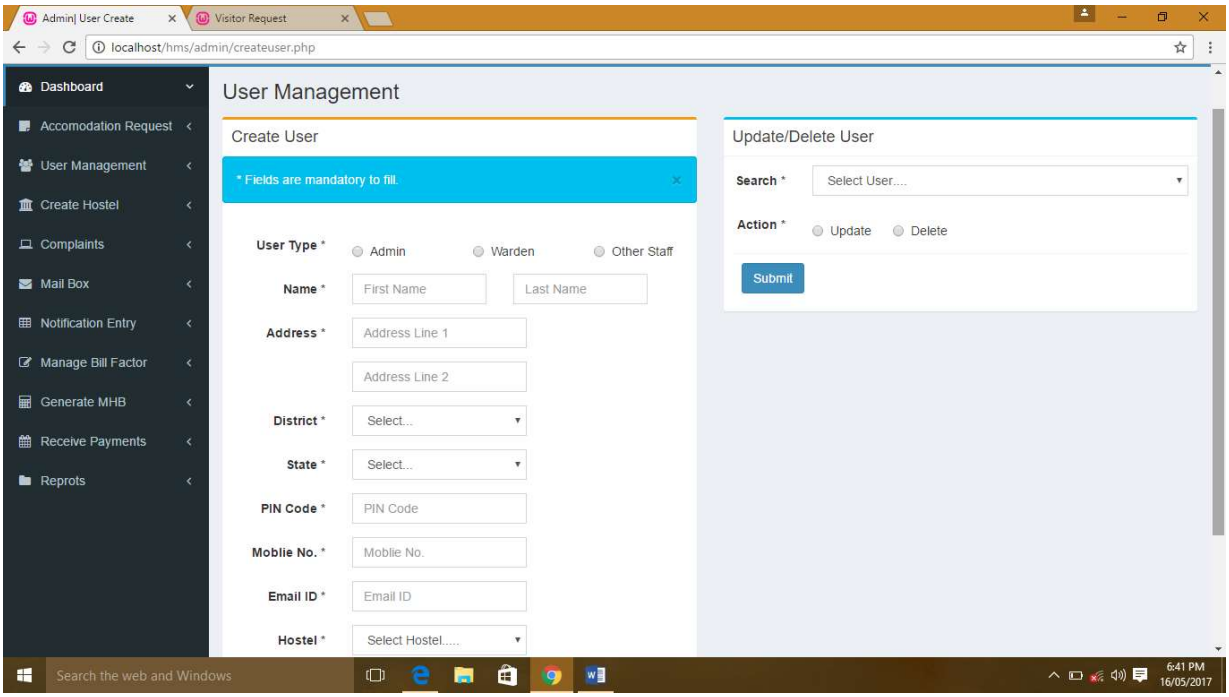
Visitor New Request



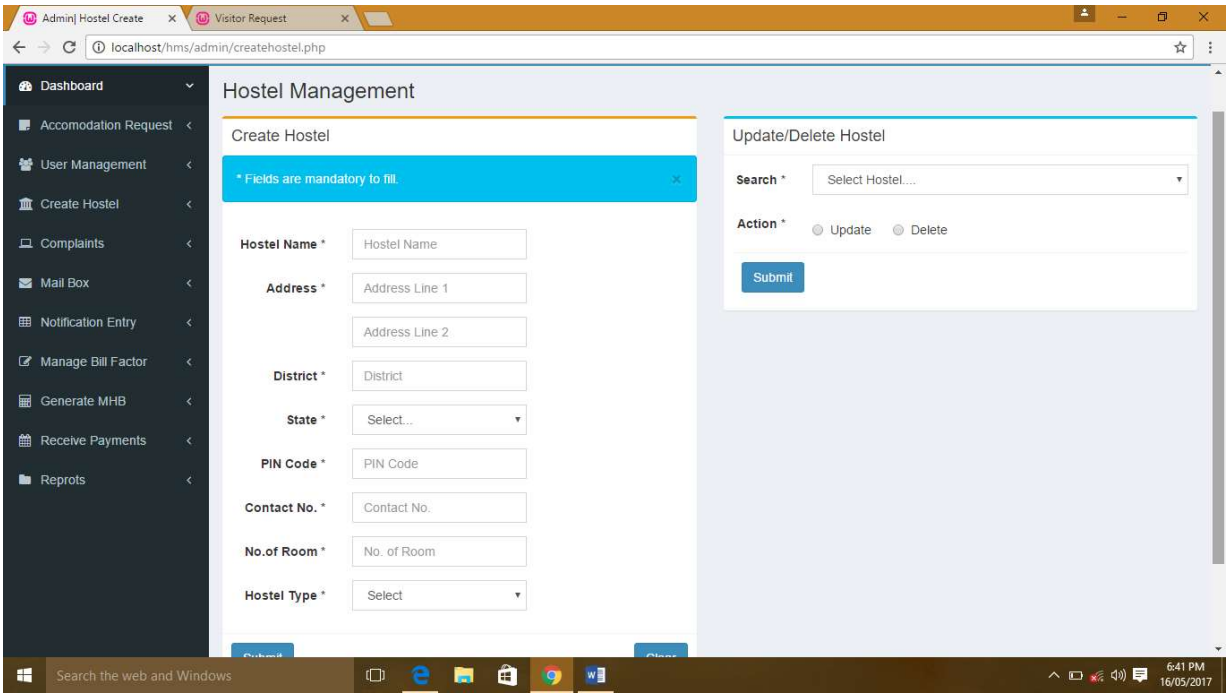
Forward Request to Warden



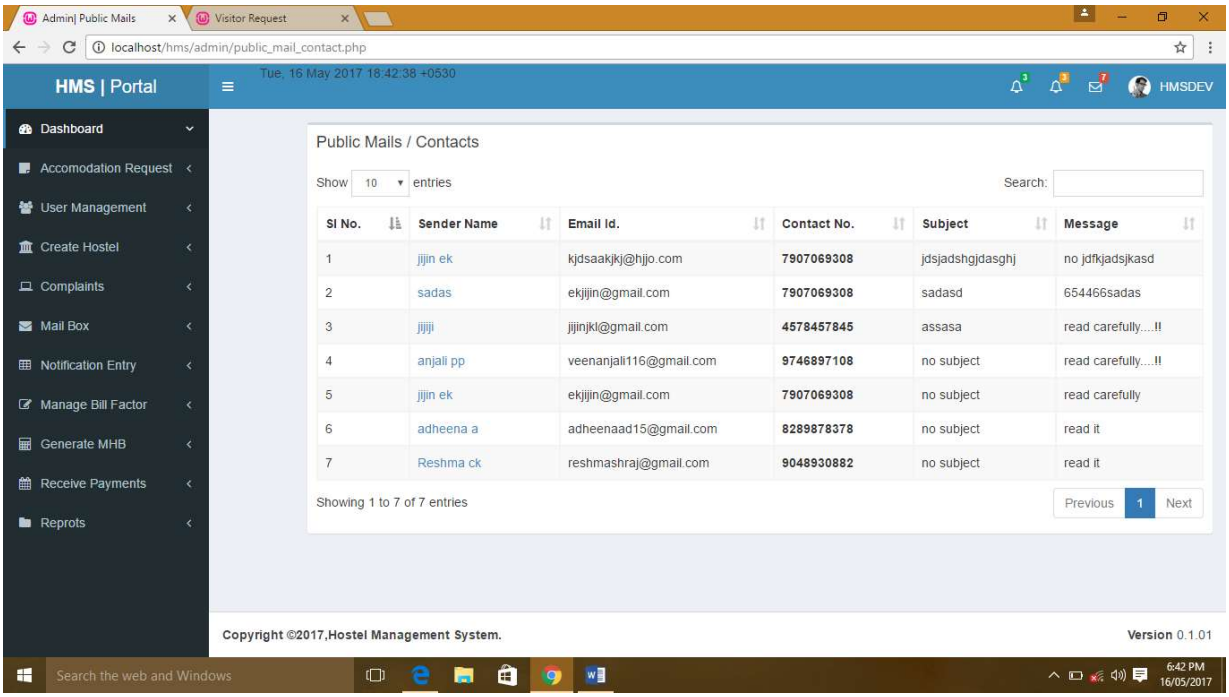
Create user/Staff



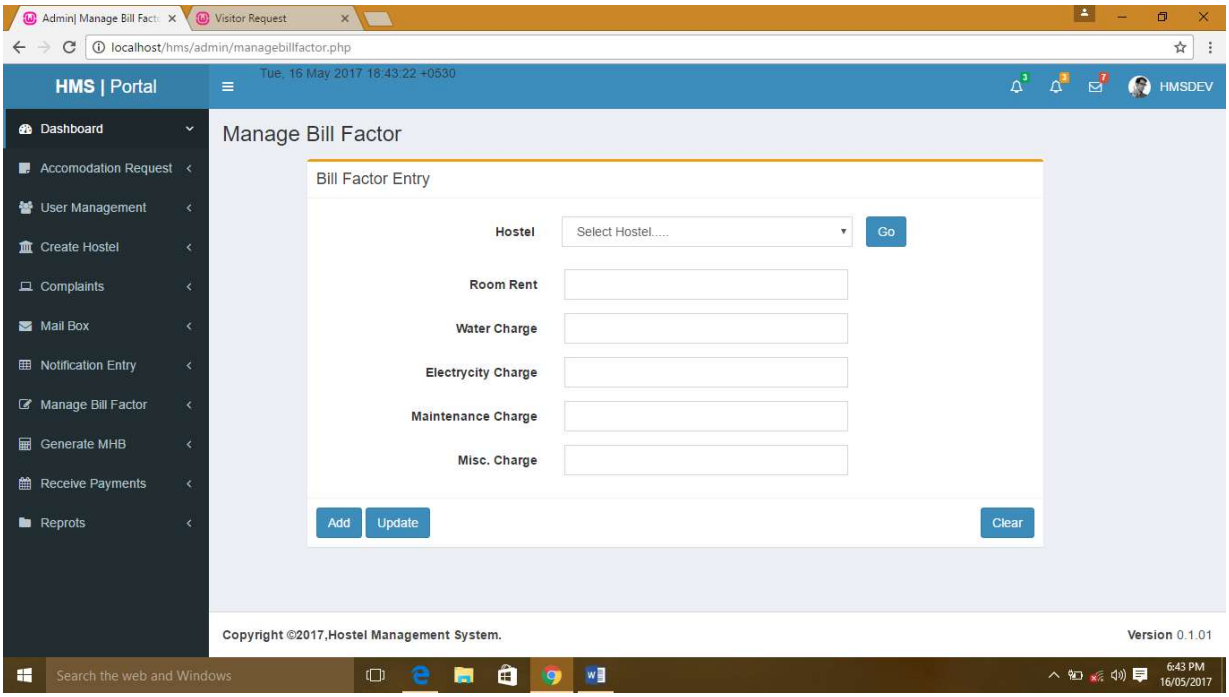
Create/Add hostel



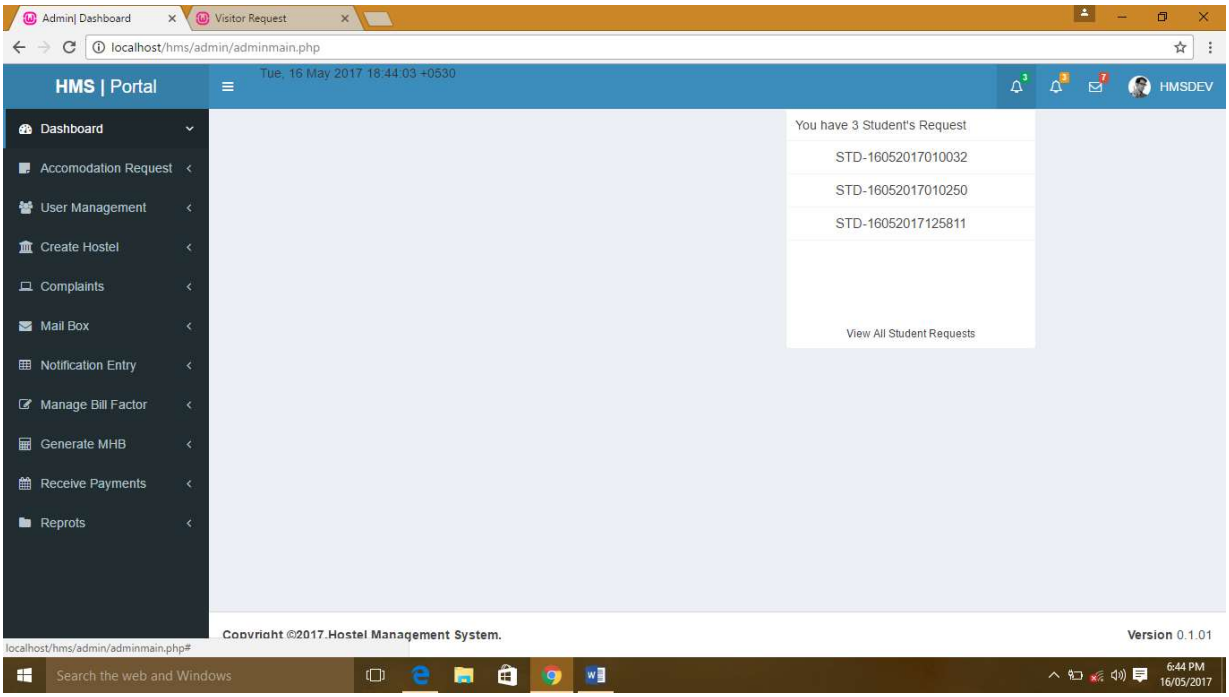
Public Contact



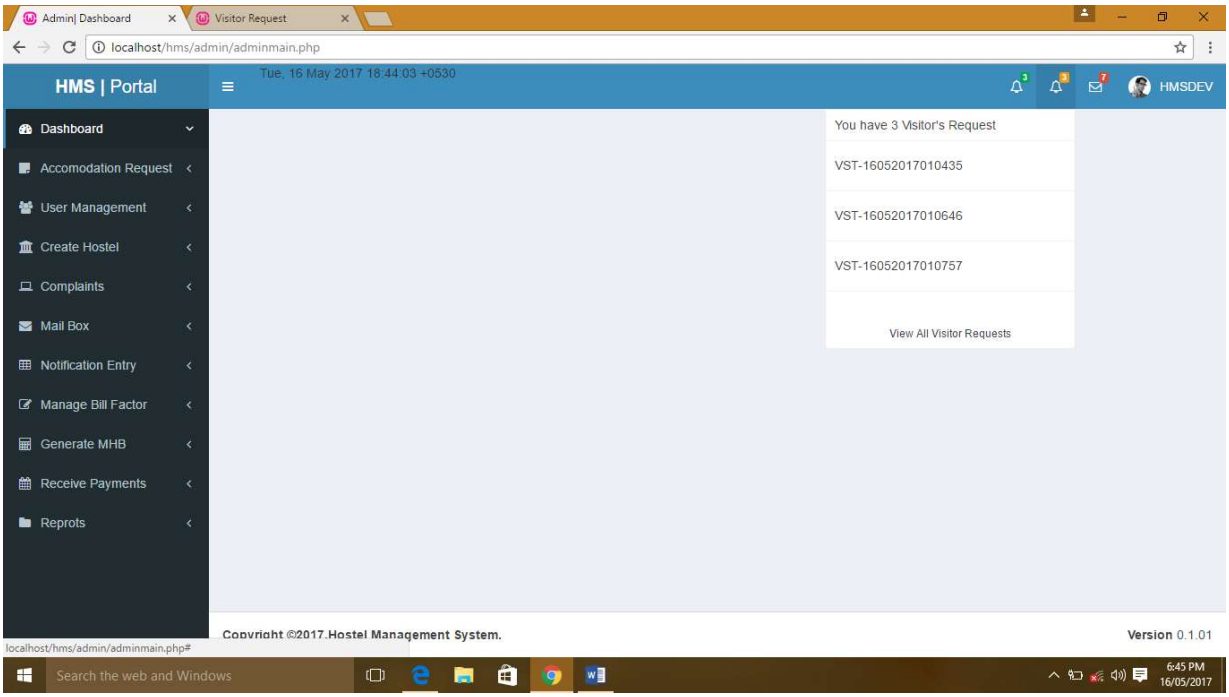
Manage Bill Parameter for Hostels



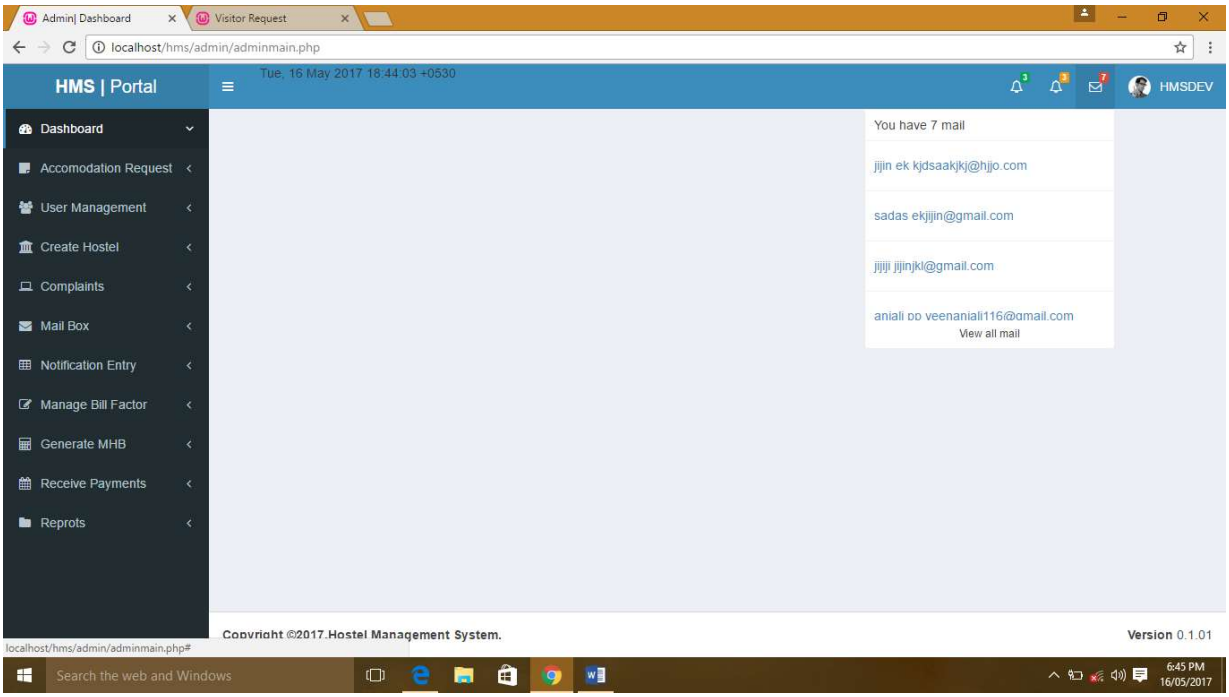
Student request Notification



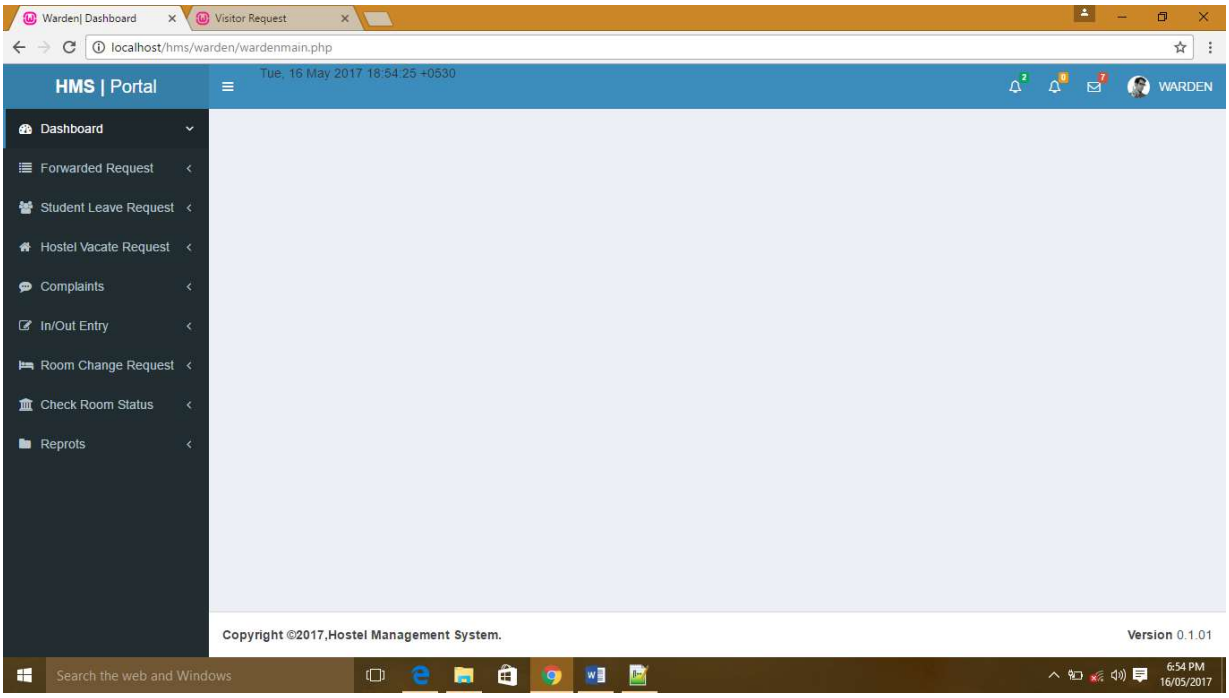
Visitor's Request Notifications



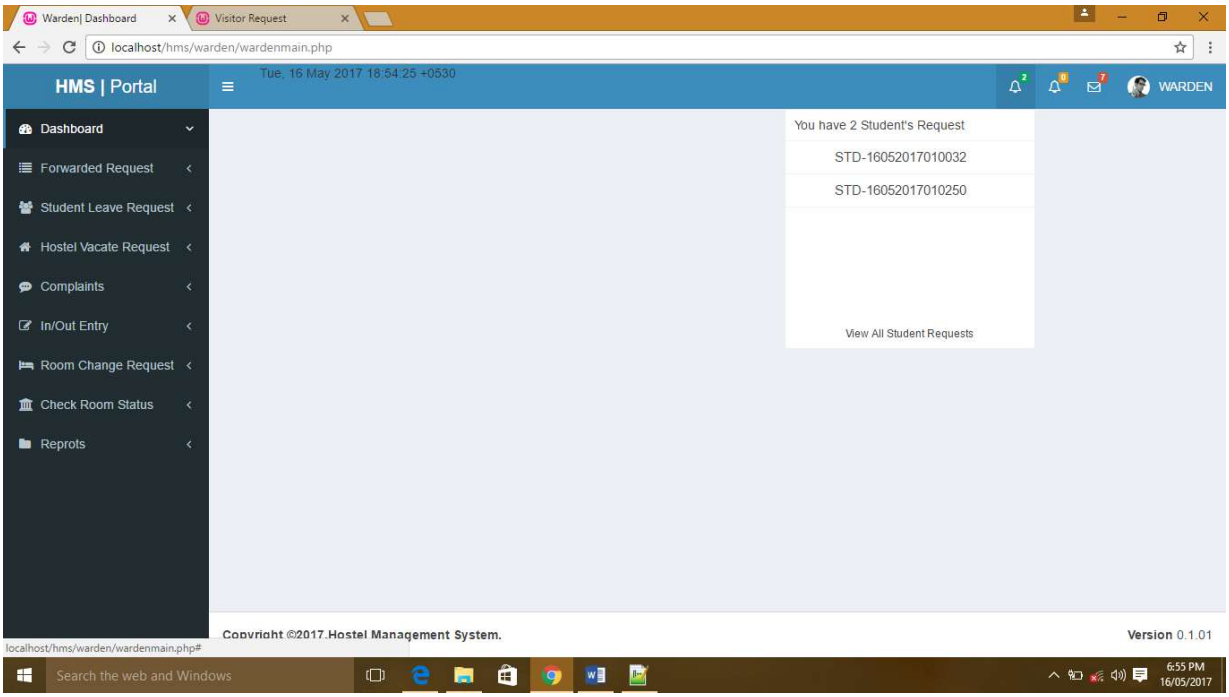
Notification for Public Contact



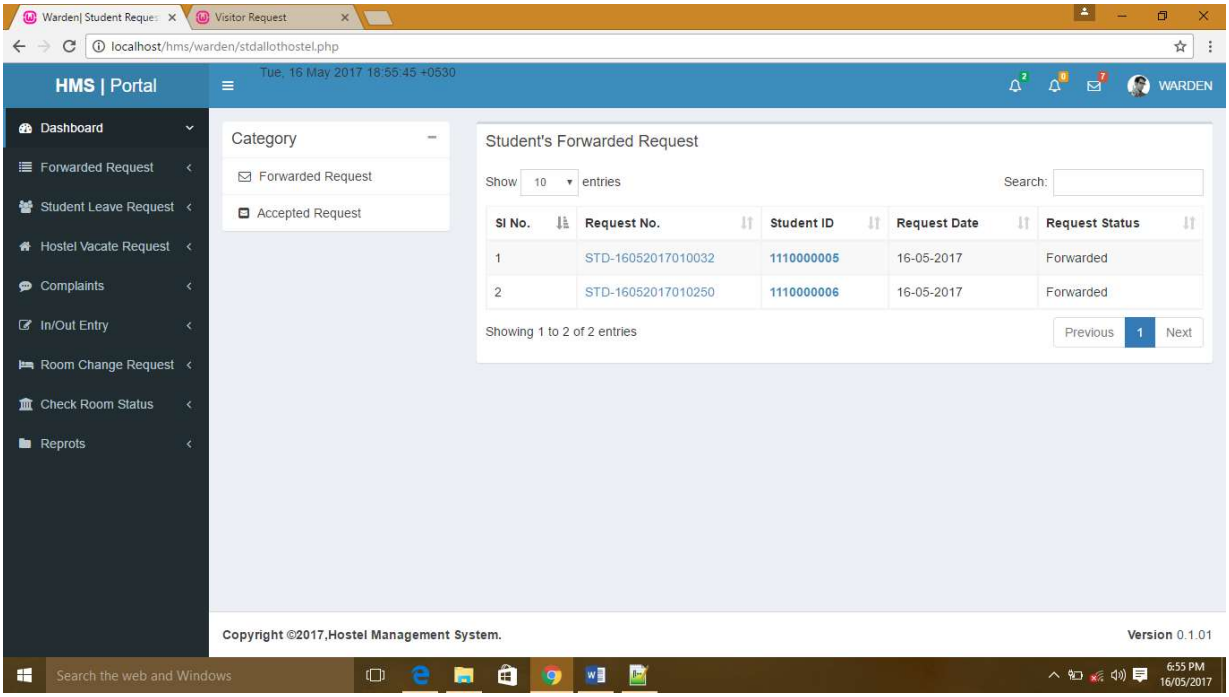
Warden Dashboard



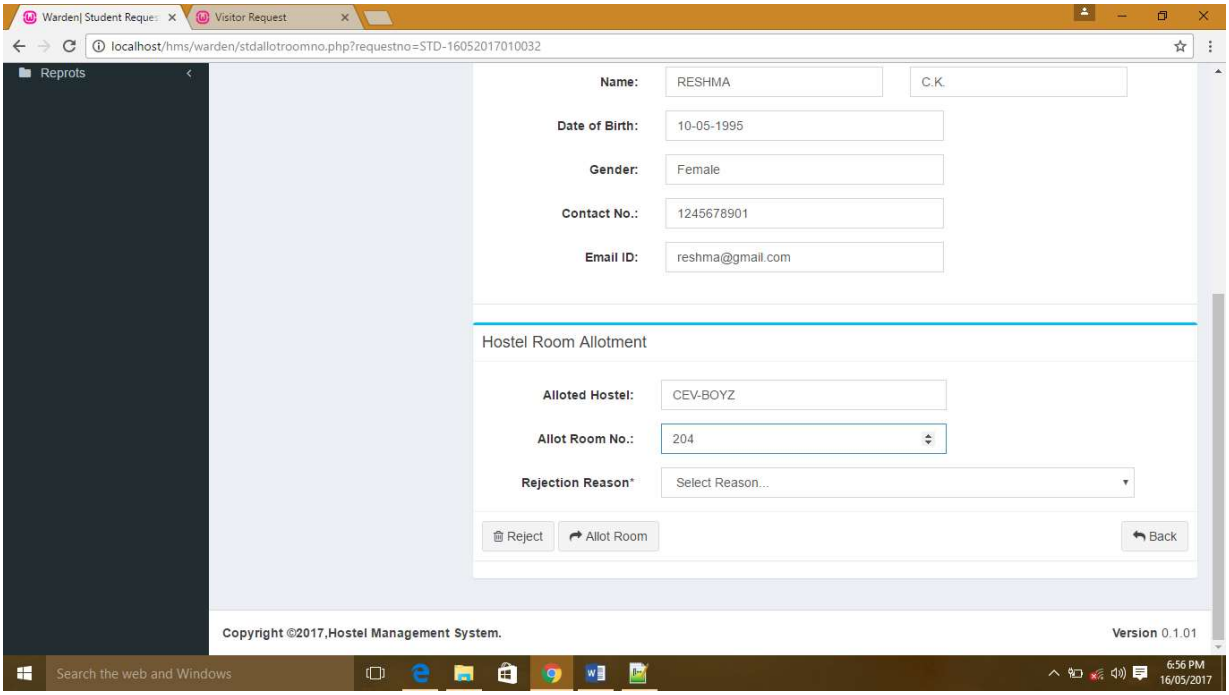
Student Forwarded Request Notification



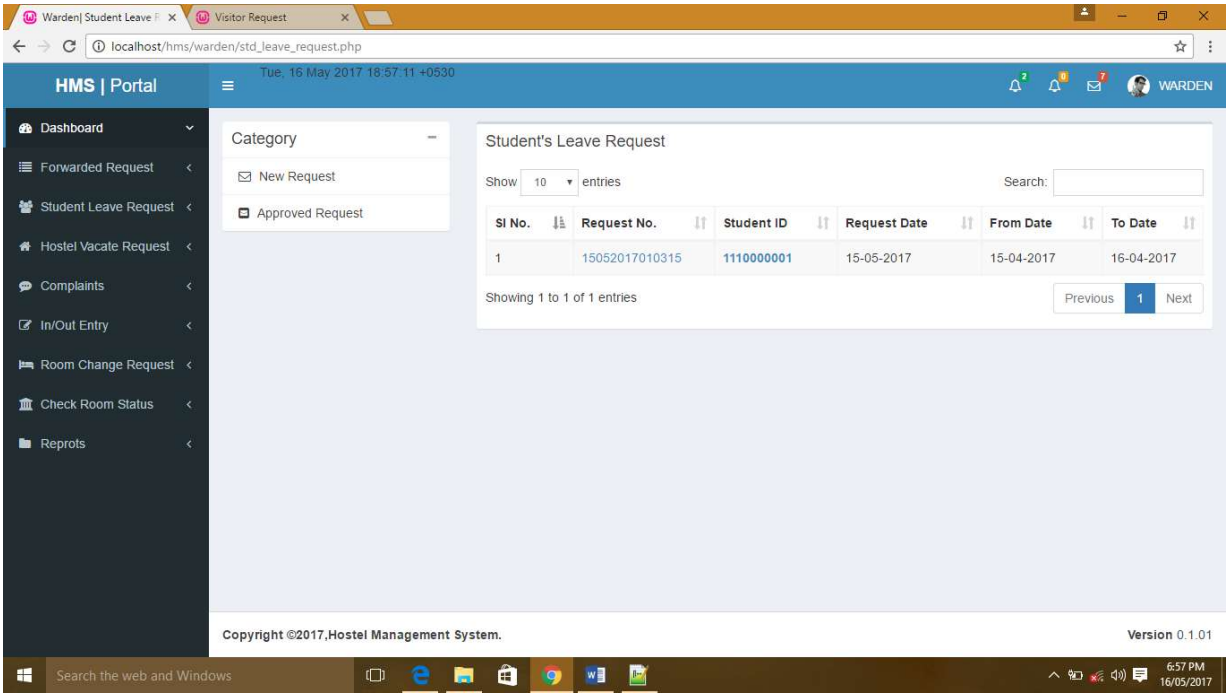
Student Forwarded Request



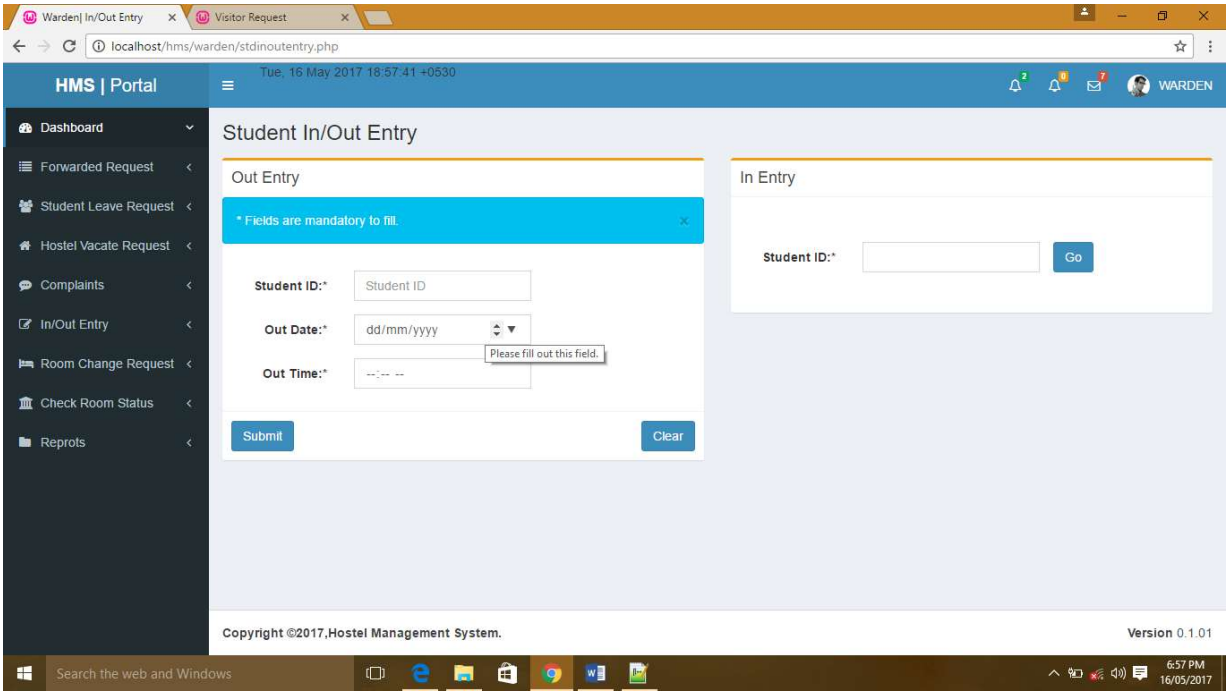
Allot Room for Student



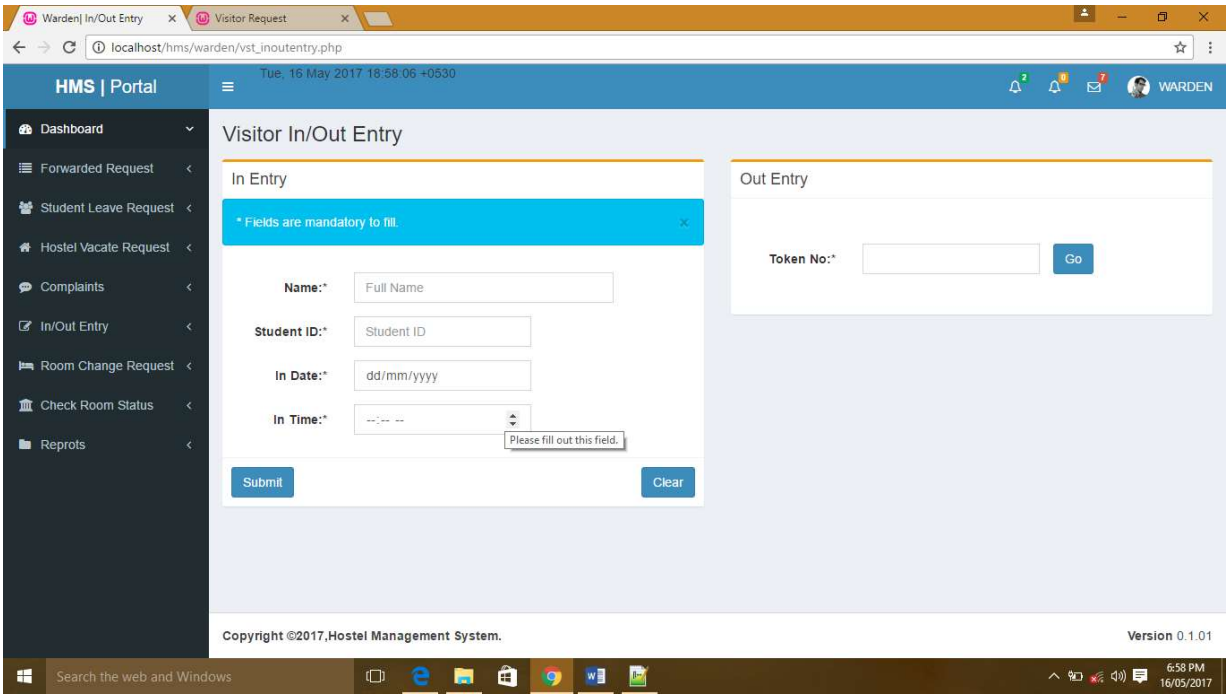
Student Leave Request



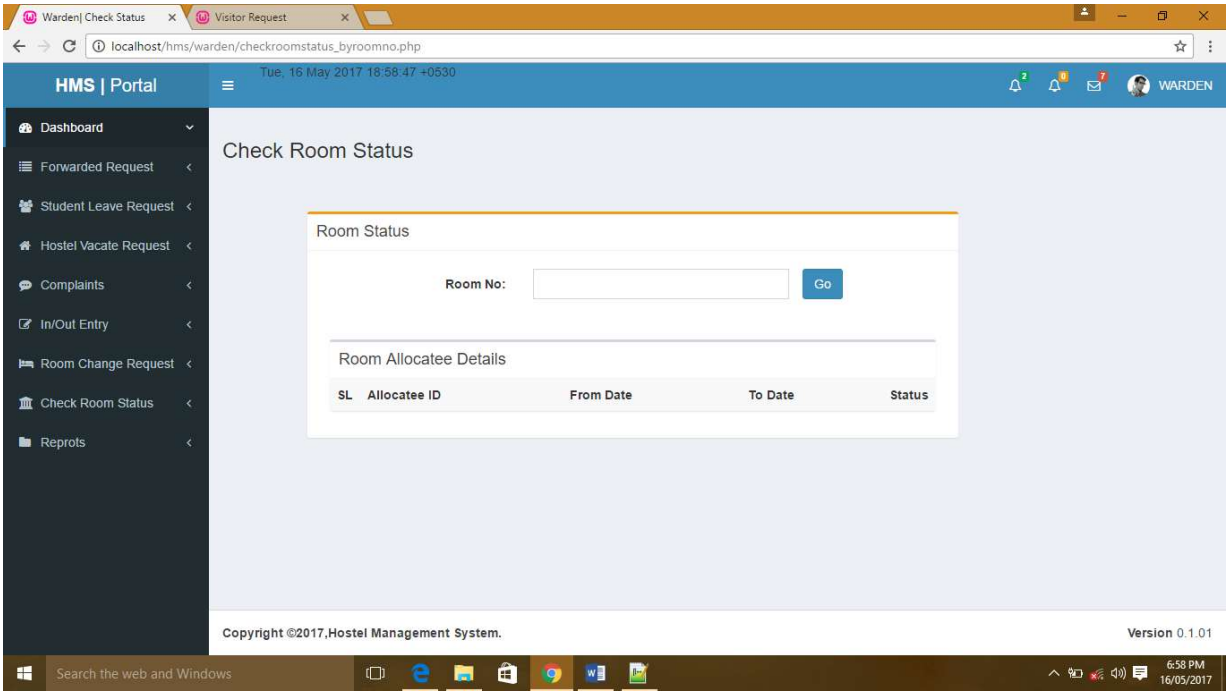
Student In/Out Entry

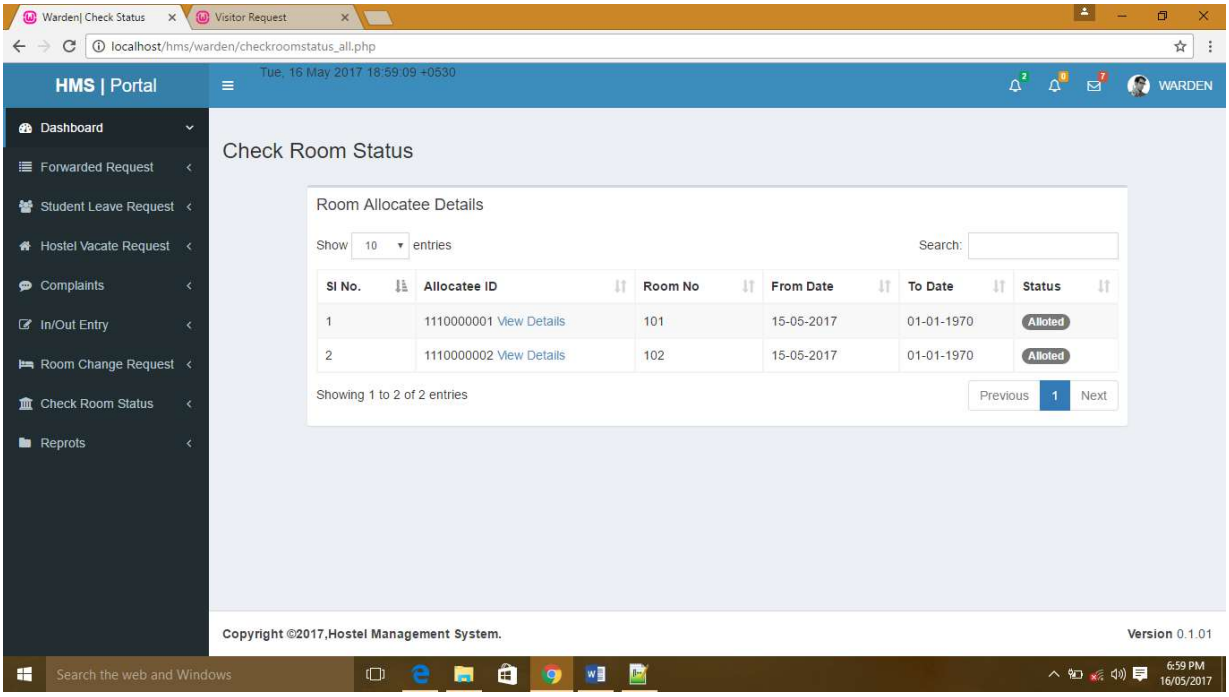


Visitor In/Out Entry

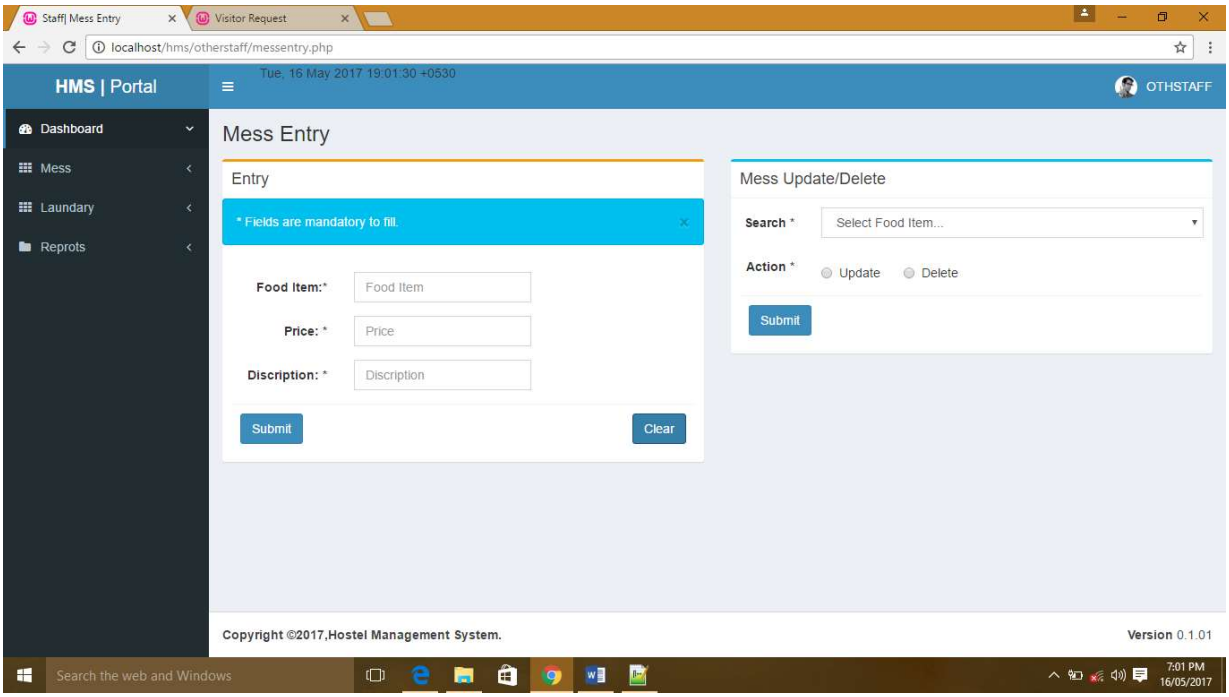


Check Room Status

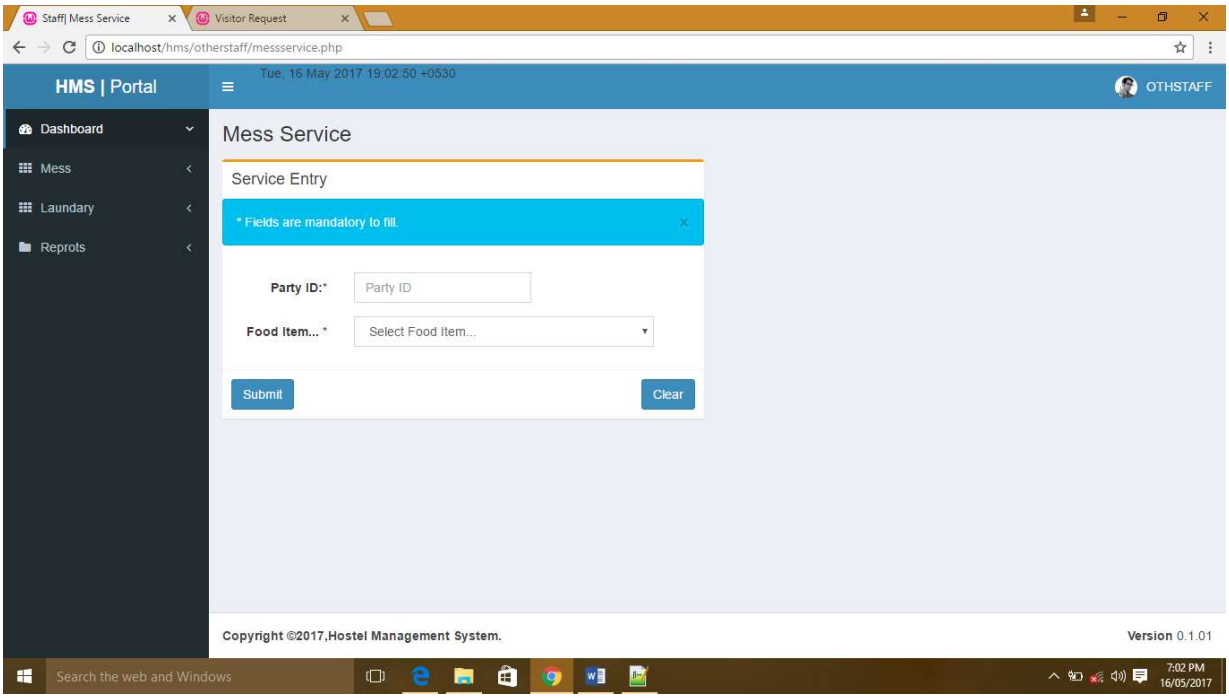




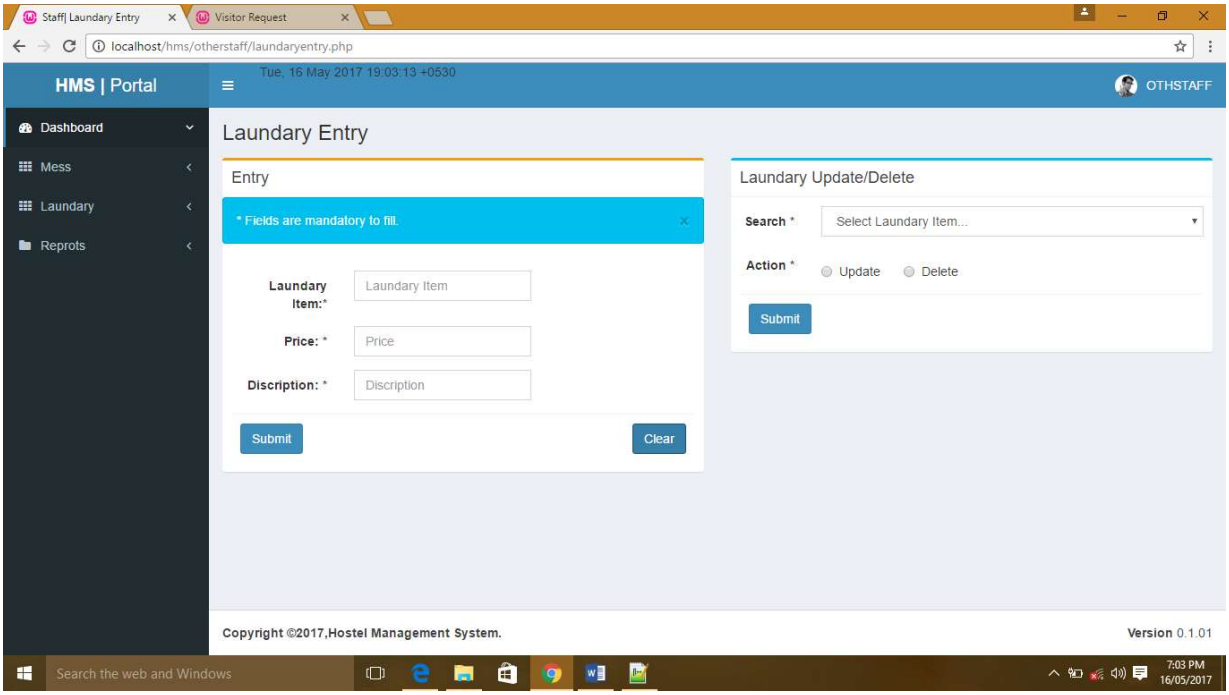
Mess Staff Food Item Entry:



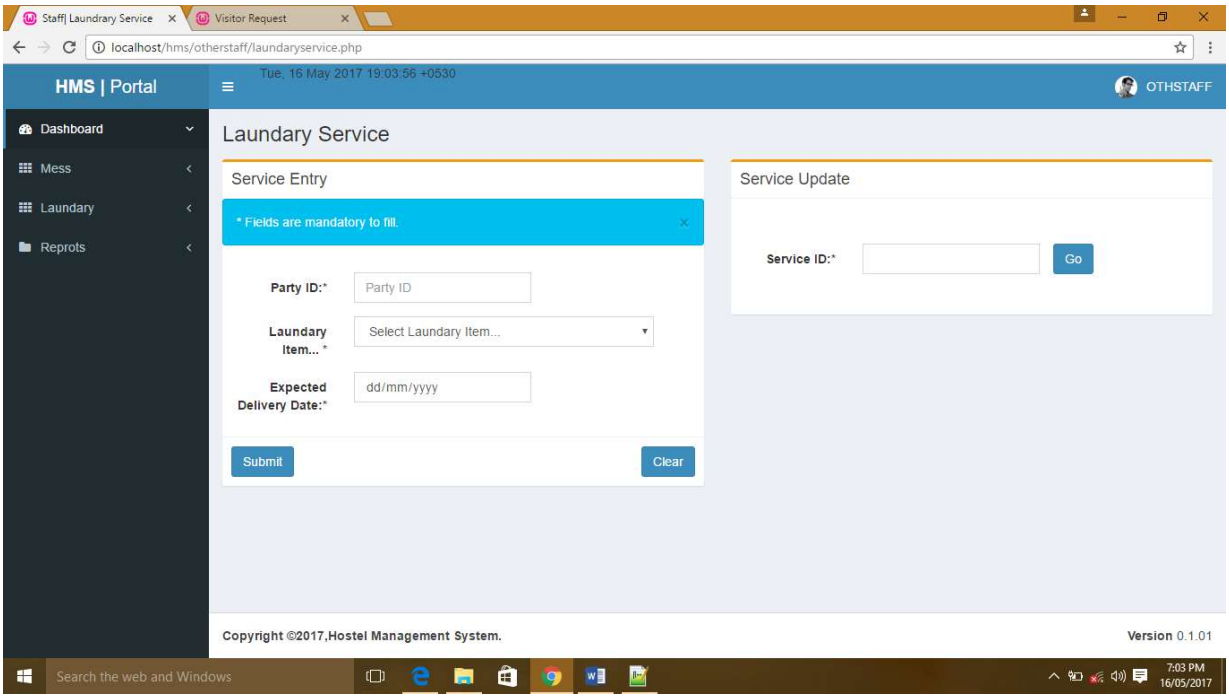
Mess Service



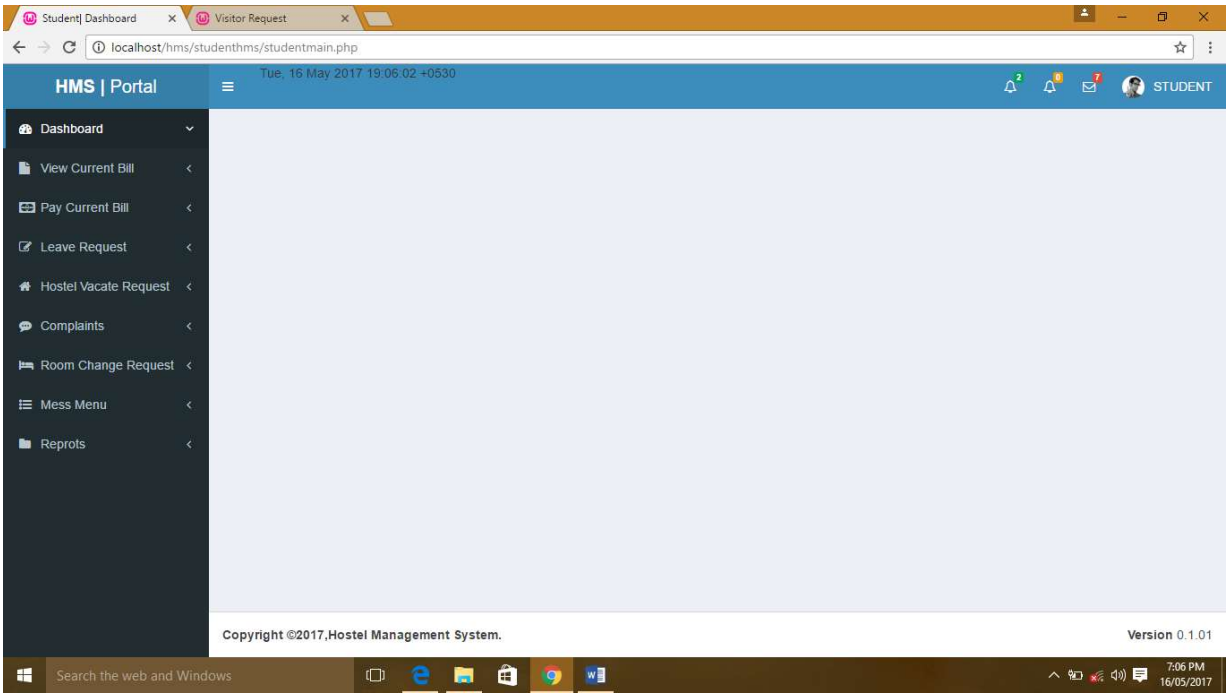
Laundry Entry



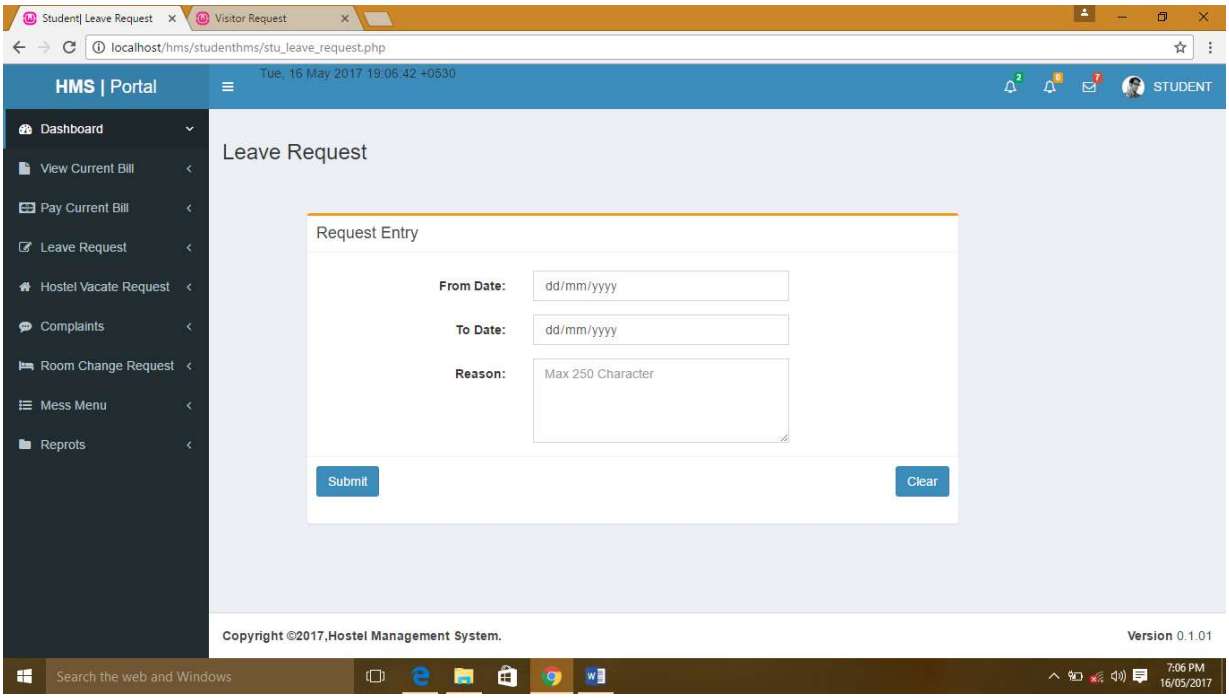
Laundry Service



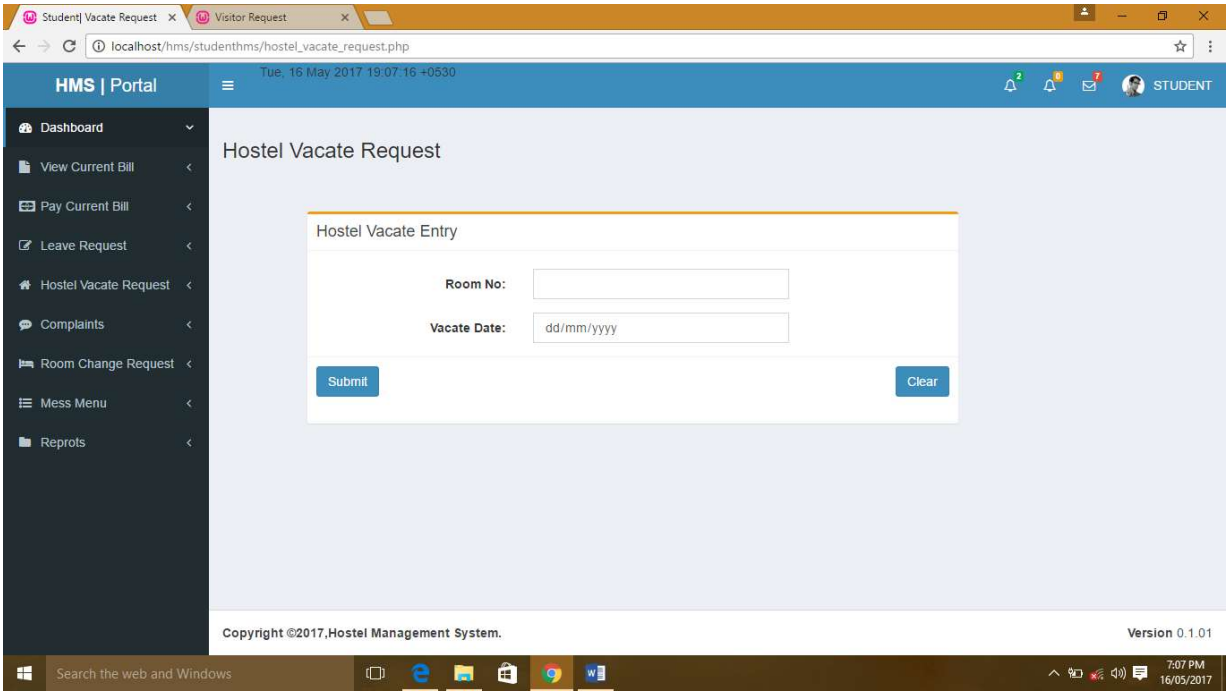
Student Dashboard



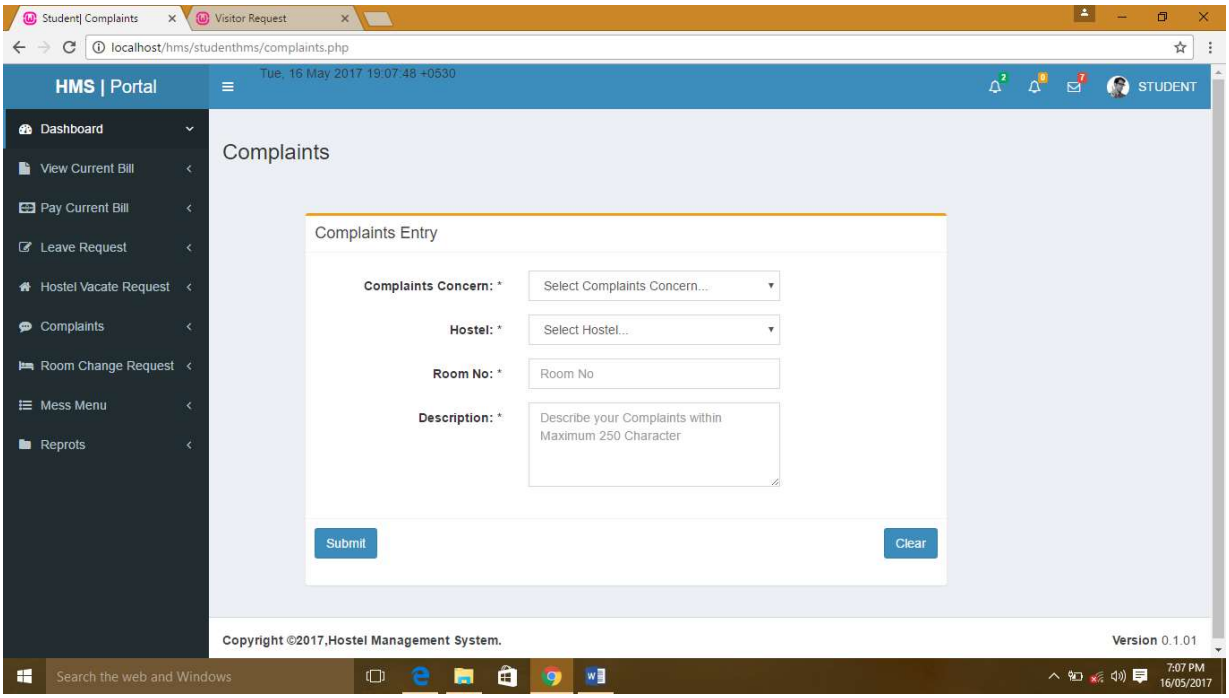
Leave Request



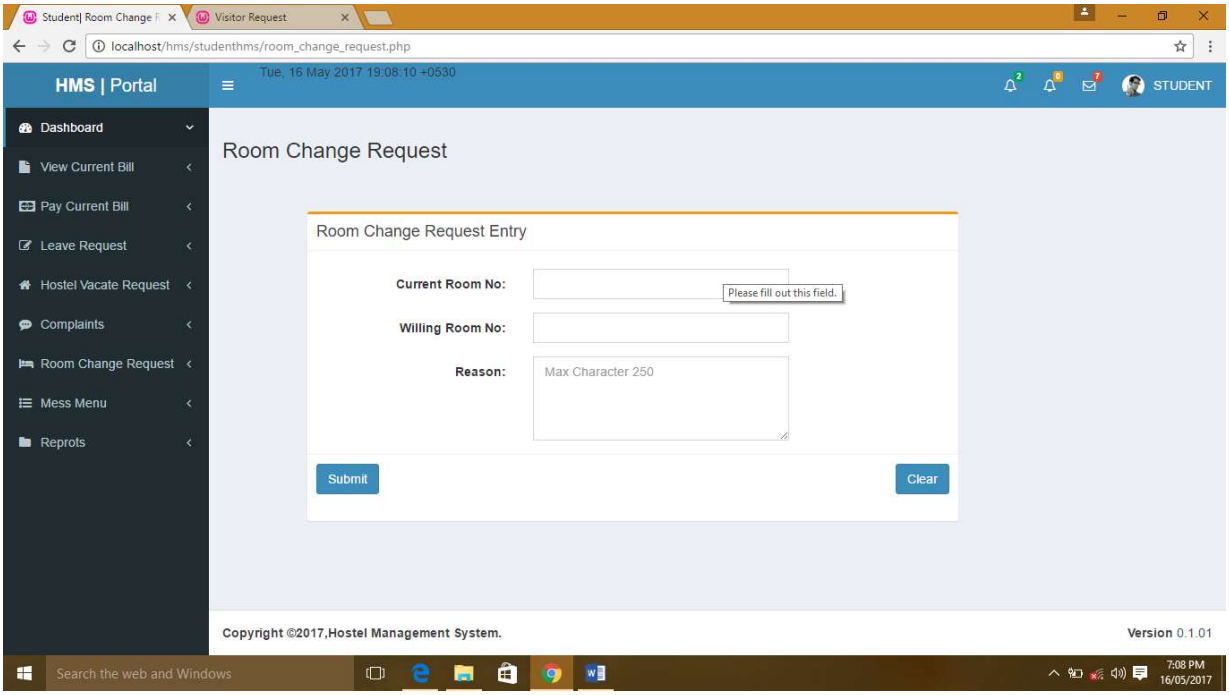
Hostel Vacate Request



Student Complaints



Room Change Request



Sample Code

Login Form

```
<?php
  include("database.php");
        session_start();
        $access = isset($_SESSION['access']) ? $_SESSION['access']: null;
        if(isset($_POST['loginbtn']))
        {
                        $myusername=$_POST['username'];
                        $mypassword=$_POST['password'];
                        $sql="SELECT * FROM `userdetails` WHERE username='$myusername' and
                                password='$mypassword'";
                        $result=mysqli_query($con,$sql);
                        $data=mysqli_fetch_array($result,MYSQLI_ASSOC);
                                if($data['username']!=$myusername and
                                        $data['password']!=$mypassword)
                                $error=1;
                                else if ($data['username']=$myusername and
                                        $data['password']=$mypassword)
                                        {
                                                //session_register("myusername");
                                                $_SESSION['login_user']=$myusername;
                                                if ($data['user_flag']=="DEV")
                                                        header("location:admin/adminmain.php");
                                                else if($data['user_flag']=="ADMIN")
                                                        header("location:admin/adminmain.php");
                                                else if($data['user_flag']=="OTHS")
                                                        header("location:otherstaff/osmain.php");
                                                else if($data['user_flag']=="STDU")
                        header("location:studenthms/studentmain.php");
                                                else if($data['user_flag']=="WRDN")
```

```
                                        header("location:warden/wardenmain.php");
                                    else
                                        header("location:welcome.php");
                                }
        }
    ?>
```

Student Request Form

```
<?php
session_start();
try{
    $con=new PDO ("mysql:host=localhost;dbname=hmsportal","root","");
    $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $con->beginTransaction();
    if(isset($_POST['stdreqsubmitbtn']))
      {
if($_POST['admnregno']==""||$_POST['admnregdate']==""||$_POST['branch']=='Select...'||$_POST
['semester']=='Select...'||$_POST['stdfirstname']==""||$_POST['dateofbirth']==""||

$_POST['gender']=='Select...'||$_POST['stdemail']==""||$_POST['stdconnumber']==""||$_POST['ad
dressline1']==""||$_POST['state']=='Select...'||
$_POST['district']==""||$_POST['socialstatus']=='Select...'||$_POST['religon']=='Select...'||$_POST['
parentfirstsname']==""||$_POST['parentssecname']==""||
                $_POST['parentsemail']==""||$_POST['parentsconnumber']=="")
                  $error;
            else{
                $admnregno=$_POST['admnregno'];
                $regdate=$_POST['admnregdate'];
                $admnregdate=Date("Y-m-d",strtotime($regdate));
                $branch=$_POST['branch'];
                $semester=$_POST['semester'];
                $stdfirstname=$_POST['stdfirstname'];
                $stdlastname=$_POST['stdlastname'];
```

```
                $birthdate=$_POST['dateofbirth'];
                $dateofbirth=Date("Y-m-d",strtotime($birthdate));
                $gender=$_POST['gender'];
                $stdemail=$_POST['stdemail'];
                $stdconnumber=$_POST['stdconnumber'];
                $addressline1=$_POST['addressline1'];
                $addressline2=$_POST['addressline2'];
                $state=$_POST['state'];
                $district=$_POST['district'];
                $pincode=$_POST['pincode'];
                $socialstatus=$_POST['socialstatus'];
                $religon=$_POST['religon'];
                $parentfirstsname=$_POST['parentfirstsname'];
                $parentssecname=$_POST['parentssecname'];
                $parentsemail=$_POST['parentsemail'];
                $parentsconnumber=$_POST['parentsconnumber'];
            /* inserting data into student table */
            $sql=("INSERT INTO
student(admnregno,admnregdate,branch,semester,fname,lname,dob,gender,emailid,mobno,addre
ssline1,addressline2,state,district,pincode,caste,religon)

        VALUES(:admnregno,:admnregdate,:branch,:semester,:stdfirstname,:stdlastname,:dateofbir
th,:gender,:stdemail,:stdconnumber,:addressline1,:addressline2,:state,:district,:pincode,:socialstatu
s,:religon)");
                $query=$con->prepare($sql);
                $query->execute(array (
              ':admnregno'=>$admnregno,
                  ':admnregdate'=>$admnregdate,
              ':branch'=>$branch,
              ':semester'=>$semester,
              ':stdfirstname'=>$stdfirstname,
              ':stdlastname'=>$stdlastname,
              ':dateofbirth'=>$dateofbirth,
```

```
':gender'=>$gender,
':stdemail'=>$stdemail,
':stdconnumber'=>$stdconnumber,
':addressline1'=>$addressline1,
':addressline2'=>$addressline2,
':state'=>$state,
':district'=>$district,
':pincode'=>$pincode,
':socialstatus'=>$socialstatus,
':religon'=>$religon));
/* end of inserting data into student table */
$stdfkid=$con->lastInsertId();  /* returning last inserted stdid from student table */
/* to use it as FK in Parents_details table and stdrequest  table */
/* inserting parents details in parents_details table */
/* inserting data into parents_details table start here */
$sql=("INSERT INTO parents_details(fname,lname,emailid,contactno,student_id)
VALUES(:parentfirstsname,:parentssecname,:parentsemail,:parentsconnumber,:stdfkid)");
$query=$con->prepare($sql);
$query->execute(array (
':parentfirstsname'=>$parentfirstsname,
':parentssecname'=>$parentssecname,
':parentsemail'=>$parentsemail,
':parentsconnumber'=>$parentsconnumber,
':stdfkid'=>$stdfkid));
/* end of inserting data into parents table */
/* primary key for student request table */
$date =Date("dmYhis");
$request_date=Date("Y-m-d");
$reqsts='NEW';
$code='STD-';
$reqpk=$code.$date;  /* final pk */
```

```
                /* inserting data into stdrequest table start here */
                $sql=("INSERT INTO stdrequest(requestno,stdudent_id,req_date,req_status)
                    VALUES(:reqpk,:stdfkid,:request_date,:reqsts)");
                $query=$con->prepare($sql);
                $query->execute(array (
                 ':reqpk'=>$reqpk,
                 ':stdfkid'=>$stdfkid,
                 ':request_date'=>$request_date,
                 ':reqsts'=>$reqsts));
                 /* end of data inserting into stdrequest table */
      $request_no=$reqpk;
    }
          $con->commit();
                }
  }
 catch(PDOException $e){
          $con->rollback();
  echo "Error !".$e->getMessage()."<br/>";
          die();
 }
 $con=null;
?>
```