Chapter One

1. Introduction

1.1 Background Information

Arbaminch University is one of the major universities in the country. The University gives educational services for regular students as well as extension programs for the people who live in Arbaminch town and neighboring cities. The University is also a research institute for different areas of field of studies.

In the University there are different management activities that are performed. Among those management activities Arbaminch University Students' Dormitory Management is one of the major management activities which are performed to arrange and allocate dorms for students. In this process there is a potential problem associated with the Dormitory Management. So the team initiated this project to identify and analyze those problems and to put possible remedies (solutions).

1.1.1 Strength and weakness of the current system

1.1.1.1 Strength of the current System

- It gives sustainable Dormitory Services for Students
- It provides approximately proportional number of proctors for buildings
- It provides better control of dormitory materials

1.1.1.2 Weakness of the current System

The weaknesses associated with the current system are as follows:

- Manual processing of management activities like: - arranging buildings for the allocation, assigning proctors for

buildings and rearranging students and dorms. Since the total no of students and dormitories available in the campus is very large, managing this huge number manually is very tedious and is prone to many problems.

- All the necessary records of the above management activities are kept manually on papers and stored in file cabinet.
- Due to manual processing, error occur which lead to unnecessary rework.

1.2 Background of the project

1.2.1 How it is initiated?

This project is initiated to develop system, as a final year project for completing a study of B.Sc. Degree in Computer Science and IT. The team is organized to develop a web based Dormitory Management System which will enable the project team to get B.Sc. Degree in Computer Science and IT.

1.2.2 Team composition

The team composition for this project is as follows:

Job position	Name
Project manager	Biniam Tesfaye
System analyst	Tadele Kebebe Frezer Yiheiyis
System designer	Suleyman Yimer
Programmer	Negeri Negassa

1.3 Statement of the problem

The major problems associated with the above mentioned activities are the following: During the arrangement of students for the allocation, list of students is received from student dean and students are classified based on their sex and level of education. During this process some male students are assigned dorm with female students and some female students also assigned with male students. This is a great problem facing each year, and leads to unnecessary rework. The other problem is all the records associated with the overall management process are stored manually on papers and stored in a file cabinet. This makes, managing and manipulation of this data time consuming and has a significant impact on the Dormitory Management System.

1.4 Objectives of the project

➢ General objective

The main objective of this project is to develop a new Web Based Dormitory Management System which solves the above mentioned problems with the existing system. This is achieved by designing a web based application program that will change the actual manual processing to a computerized environment.

➢ Specific objective

ARBAMINCH UNIVERSITY DORMITARY MANEGEMENNT SYSTEM (**AMU-DMS**) has the following specific objectives

- To design a user friendly system

- Designing a user interface for the new Dormitory Management System.

- To develop a database to keep the overall records associated with the management process.

- Enabling students to have all information about their dormitory information before coming to campus, so that students may not face difficulties after coming there.

- Every student's data collected when a student get into dormitory life will be stored in a data base so that there is no loose of student record.

1.5 Feasibility Analysis

1.5.1 Operational feasibility

The entire team member, expect that the system which is in development is to be operational. That is once the system is deployed, it can operate on any of the operating systems which have a .NET framework installed. Therefore, the system will be designed to be operationally feasible that if it is deployed, the system will operate in any kind of platforms without any mal functionalities.

1.5.2 Technical Feasibility

The entire group members are expected the system to be technically feasible. The system is going to be developed by following the Object Oriented System Development technique. And the team has the ability to develop this system without any

difficulty since the team has studied the required methodologies and tools. So the system will be technically feasible.

1.5.3 Economic feasibility

To identify the economic feasibility of the project the team has done cost-benefit analyses which enable the team to specify the benefits and costs associated with the project. The following work sheets specify the costs as well as benefits associated with the project.

Intangible benefits: The following worksheet lists the intangible benefits associated with the project.

INTANGIBLE BENEFITS WORKSHEET AMU Dormitory Management System
1.Increase Employee Morale 2.Reduce Resource Consumption 3.Increase Management flexibility 4.Provides More timely information

Tangible benefits: The team calculated the corresponding tangible benefits based on the technique called the Time Value of Money (TVM).

1. **Cost Reduction and Avoidance**: - To calculate these following things will be considered.

Total Number of proctors in existing system= **27**

Average Salary of each proctor per month = **600.00Birr**

Total money required for payment per year= **27*600*12= 194,400Birr**

Average Number of proctors needed when the new system is deployed= **10**

Average salary of each of them per month = **600.00Birr**

Total money required for payment per year= 10*600*12= **72,000.00Birr**

Difference b/n before and after deployment money required for payment

Cost Reduction and Avoidance= **194,400.00Birr-72,000.00Birr**

= 122,400.00Birr

2 **Error Reduction** :- to calculate the things to be consider are following

Average error occurred = **10%**

Money needed for error payment per year =**194,400.00Birr/10**

=19,440.00Birr

3 **Increase Speed of activity** :- Increased speed calculated as follows

Especially in allocation:-

Average Days required for allocation= **15 days**

Average proctor salary per day=**20.00birr**

Average Days required for allocation in terms of money=27*20*15= **8,100.00Birr**

Average days required for the system= 1 day

Average Days required for allocation in terms of money=10*20*1= **200.00Birr**

Difference (Speed increased by) = 8100.00Birr-200.00Birr= **7,900.00Birr**

TANGIBLE BENEFITS WORKSHEET AMU Dormitory Management System Year 1	
1.Cost Reduction and Avoidance	122,400.00 Birr
2.Error Reduction	19,440.00Birr
3.Increase Speed of activity	7,900.00 Birr
4.Improvement of Management and control	122,400.00 Birr
5.Others	0.00
Total Tangible benefit	**162,140.00 Birr**

One time cost: The following worksheet specifies the One Time cost associated with the project.

ONE-TIME COST WORKSHEET AMU Dormitory Management System	Year 0
1.Development cost	300.00 Birr
2.New Hardware cost	250.00 Birr
3.New (purchased) software	100.00Birr
4.User training	1,400.00 Birr
Total One-time costs	2,050.00 Birr

Recurring cost: the following worksheet specifies the recurring cost of the project.

RECURRING COST WORKSHEET AMU Dormitory Management System Year 1
1.Application Software Maintenance 1,500.00 Birr
2.Incremental Data storage () ---------
3.New hardware/software leases ---------
4.Supplies (papers, pens,....) 100.00 Birr
Total Recurring cost 1600.00 Birr

1.5.4 Schedule feasibility

All the team members expect that the project will be completed with in the time frame stated, so that the system will be feasible regarding schedule.

1.6 Significance of the project

The significances of this project are:

- ✓ Avoiding wastage of students time as well as management time
- ✓ Avoiding data loss because of improper data storage

- ✓ Avoiding improper dormitory allocation
- ✓ Avoiding improper resource consumption

1.7 Beneficiaries of the system

There are different bodies that will be benefited from this system. The main beneficiary of this system is the Dormitory Management in which, first, the environment is changed to a computerized environment, which improves the quality of internal operations as well as services given to students. Secondly the problem associated with manual processing is minimized and the quality of work and services became improved.

The other beneficiaries of the system are **proctors**, **proctor manager** and **students**. Ones the new system is implemented, firstly proctors and proctor managers are benefited from the system in such a way that the quality and performance of their work is improved, the time they spent for manual operation is significantly reduced and their management and control of their job is improved. Secondly **students** are not expected to be in campus to know about their dormitory information. That is, once the allocation report is generated by the system, the system provides an interface which enables the students to know about their dormitory information, about academic colander and some academic announcements and finally they submit their personal information through the Internet.

A **proctor** and a **proctor manager** will also be benefited from the system. These benefits may be in saving time while arranging buildings, Dorms and Students, minimizing errors while allocating students, and avoiding data loss.

1.8 Methodology for the project

1.8.1 Data collection method

To get a precise data from customers the team has used the following fact finding techniques. Those are: -

Interview: - to get the basic information and background information about the existing management system, the team has interviewed the proctors and some students about the services that are given to them, and the problems associated with that environment.

On job observation: - Here the team used to revise some data entry forms and repots associated with the management process.

1.8.2 System Analysis and Design Methodology

In this project the team used Object Oriented System Development methodology (**OOSD**). This has two phases.

Object Oriented Analysis (OOA): During this phase the team used to Model the functions of the system (use case modeling), Find and identify the business objects, Organize the objects and identify the relationship between them and finally model the behavior of the objects.

Object Oriented Design (OOD): During this phase the team used to refine the use case model to reflect the implementation environment, Model object interactions and behaviors that support the use case scenario, and finally update object model to reflect the implementation environment.

1.8.3 Case Tools

In this project the following system development tools are used

- ✓ ***Microsoft visual studio 2008:*** to design the graphical user interface and the whole application.
- ✓ ***Microsoft SQL server 2005***: for designing the database.
- ✓ ***Microsoft VISIO2007:*** for designing UML diagrams associated with the project.
- ✓ ***Microsoft office 2007:*** for documenting the corresponding deliverables associated with the project.
- ✓ ***Macro Media Dreamweaver:*** for designing web interfaces
- ✓ ***Xara Web style 3.***0: To Design web interface of buttons, links and other web controls (interface controls).

1.8.4 System Development Environment

The team used Microsoft Visual studio 2008 for the whole system development environment and SQL server 2005 for designing the database. And the team uses Macro Media Dreamweaver & Xara Webstyle3.0 for designing web interfaces.

1.8.5 Requirements structuring and Data modeling tools

Since the team is being using an Object Oriented System Development methodology, for structuring requirements and for modeling the data the team used a Unified modeling language (**UML**). The team used **UML-** diagrams for requirements structuring as well as data modeling.

1.8.6 Testing procedures

Before directly deploying this system the team will perform different testing for its functionality and meeting customers need. First the team tests each unit at each phase. So, if a problem is encountered it will immediately fixed. Second the team will perform an integration testing to check whether the system meets all the functional requirements. System will be tested using the following system testing procedures.

Alpha testing:-In this testing method, the system will tested by giving the correct input. It is tested by a customer at the developer Site.

Beta testing: **-**In this testing method, team will force the system to be tested for incorrect data input. The System will be tested by the customer at their actual work place.

If any failures occurred while testing the system in all the above testing methods, the team will take immediate correction beginning where this fault occurred before jumping to next work so that it will meet the goal. If all the above testing methods are carried out and find to be valid the system will directly deployed.

1.9 Scope and Limitation of the project

Since Arbaminch University dormitory management performs its basic tasks manually the scope of this project is to develop and implement a new web based Dormitory Management System which will avoid the problems associated with the manual processing.

Scope of the project

The scope of the project is to:

- ✓ Designing and Implementing the Database,
- ✓ Designing and Implementing Graphical User interface including forms and reports.

Limitation Of the project

This project is limited only to those activities and operations related to the dormitory management which the team is intended to deal with. The project is limited to developing the web based dormitory management system.

1.10 Risk Assumption and Constraints

There are two kinds of risk assessment in any software development life cycles. Those are Development process risks and Product risk. While developing this system, the project team may encounter different types of risks like:

- ✓ Requirement changes
- ✓ Technical risks
- ✓ Product risk

✓ **Requirement changes:** since this risk leads to system re work, the team altered participatory type of data modeling to overcome such risks. Since the team discusses with the customer when needed in each phase, the team will handle this risk before it leads to system rework.

✓ **Technical Risks:** These risks may result from excessive constraints, lack of experience, poorly defined parameters, or dependencies on organizations outside the direct control of the project team. In order to mitigate or control this risk the team will perform periodic checks on the work that have been done, and by using continuous advice provided from the project advisers.

✓ **Product risk:** since the system developed through strong participation of customers at each phase such risks may not cause series problems.

Chapter two

2. Description of the existing system

2.1 Players in existing system

An existing system compromises different players to carry out its job. Among those different actors (players), the most commons are **Dean of Student**, this body provides the list of all students who fulfilled every requirement for allocation to proctors, **Students**, they will be placed in their dorm by proctors and assigned for the property they get from the proctor, **Proctors**, They involved strongly in the existing system. Proctors collect students list from student dean. After they get all these information's from this body they will place those students according to their sex, Session, Class year, department and faculty.

The major actors in the existing system are:

- **Dean of Student**
- **Students**
- **Proctors and**
- **Proctor manager**

2.2 Major functions of existing system with clear inputs, processes, and outputs

Even if the existing system is performs its activities manually, it has different major functions.

- **Arranging buildings for the allocation:** here the total no of building is determined with its holding capacity
- **Arranging students for allocation:** here total no of students and their academic information such as department, faculty, class year and session is received from registrar. Students are then arranged based on their sex, class year, session and their department and faculty for dormitory allocation.
- **Dormitory allocation:** based on the arrangement of students dorms are allocated for students along with associated dormitory resources, like lockers, tables, chairs, beds and the like.
- **Generating allocation report:** based the dormitory allocation the allocation report is prepared and posted for student when they arrive at the campus after annual brake.
- **Managing and controlling dormitory materials:** at the beginning and end of each semester, dormitory materials are recorded and controlled whether they are functioning properly or not, then appropriate measure is taken.
- **Controlling student's discipline:** In addition to the above functionalities student's discipline measures are controlled and recorded, whether they use the dormitory materials properly or not, and whether they act and perform things as per the dormitory rules and regulations.

2.3 Business rules

The business rules associated with the existing system are as follows.

- Only one dorm is assigned for one student, and that student should live in the dorm which belongs to him.

- Students should not change their dorm without the permission of the proctor with sufficient reason.
- Students are allocated in such a way that male students are not allocated with female students.
- Proctors should not assign one student in more than one dorm.
- Proctors should not student's personal information for other purposes.
- Buildings should be arranged before the allocation.
- Students should submit their personal information before they are assigned in a dorm.
- After the allocation reports should be prepared by proctors for students.

2.4 Report generated in the existing system

In an existing system there are different reports generated for different purposes. Those reports include Student Dormitory allocation report, Student status report; Resource received report, and clearance report in addition to conditional report such as discipline case report, damaged resource report, and etc.

The dormitory allocation report contains the report related to student's block number and dorm number. Resource received report includes reports of materials that a student has taken from a Proctor when he/she first assigned in to that dorm. The student status report is any report that contains any up-to-date information about a student. Discipline measurement report embraces reports such as does a student contains any discipline record in this campus and what type of discipline measure were taken will be generated in the report. Clearance report is a report which is generated when any student wants to leave a campus because of different reasons. When he/she leave a campus the above reports will be checked by the proctor collectively.

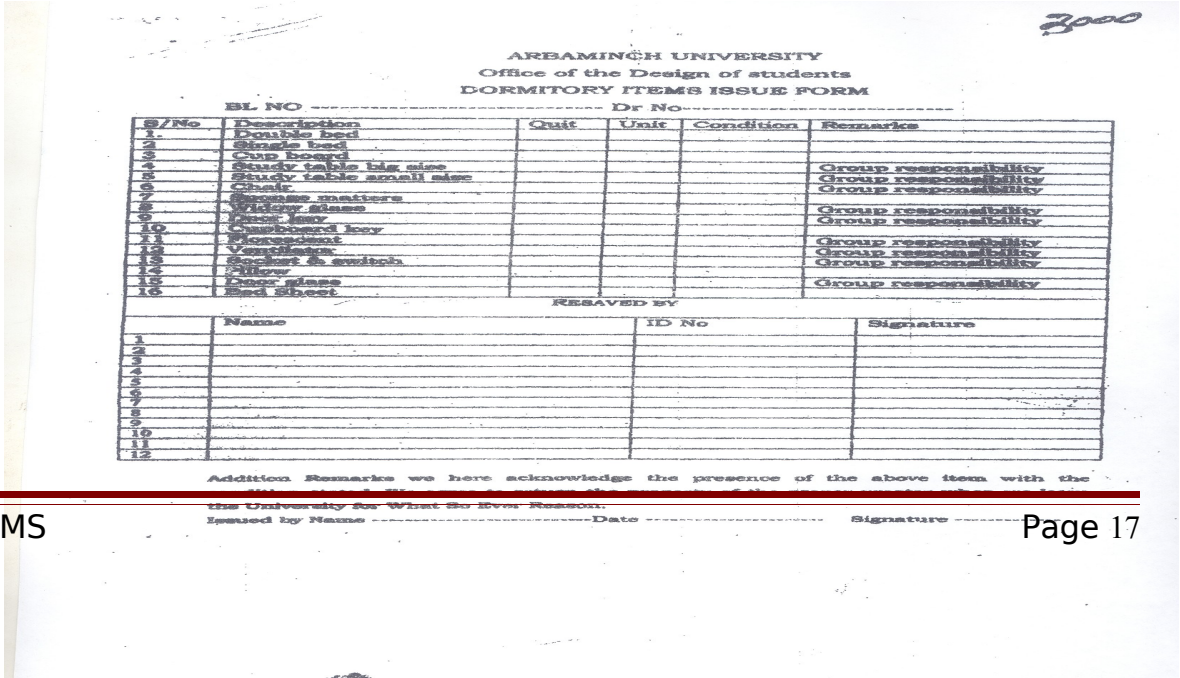
Those all reports were checked to clarify a student whether he/she returned all resources that he/she used, is he/she free of discipline measures? After checking those reports a proctor will clear the student that ensures that the student is free of any resources while he/she was in dorm.

2.5 Forms, reports and documents etc used in existing system

In the current system, they use different forms and reports to manipulate different records associated with the different activities. From those forms some are **student's personal information form**, **dormitory materials property form**, **clearance form**, and others. And the reports associated with the existing system are **dormitory allocation report**, **student status report**, and other reports associated with the system.

Some **sample forms** associated with the current system are shown below

- **Dormitory Item Issue Form:**



ARBAMINCH UNIVERSITY
Office of the Design of students
DORMITORY ITEMS ISSUE FORM

BL NO ---------- Dr No----------

S/No	Description	Quit	Unit	Condition	Remarks
1	Double bed				
2	Single bed				
3	Cup board				
4	Study table big size				Group responsibility
5	Study table small size				Group responsibility
6	Chair				Group responsibility
7	Bathroom matters				
8	Widow glass				Group responsibility
9	Door key				Group responsibility
10	Padlock and key				
11	Floorpedent				Group responsibility
12	Ventilator				Group responsibility
13	Socket & switch				Group responsibility
14	Pillar				
15	Door glass				Group responsibility
16	Bed Sheet				

RESAVED BY

	Name	ID No	Signature
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Addition Remarks we here acknowledge the presence of the above item with the [illegible] the University for What So Ever Reason.

Issued by Name ----------Date ---------- Signature ----------

- **Student Personal Information Record**

የአርባ ምንጭ ዩኒቨርሲቲ
የተማሪዎች አገልግሎት የተማሪዎች
የግል ሁኔታ የመመዝገቢያ ቅጽ

ተ.ቁ	የተማሪው ስም Name of the student	ዲርግ.ተ.ሪ ብሉክ Blak	የዲርጎም ቁጥር	ዲፓርት መ.ን Dep	የትምህርት ደረጃ	የመታወቂያ ቁጥር ID .No	የግል ሁኔታዎች / Personal Data የአደጋ ጊዜ ተጠሪ ስም	የስል ክ ቁጥር	ክልል	ከተማ	ተባሉ ከፍ.ተኛ

Form

2.6 Problems of existing system (using PIECES frame work)

The manual dormitory management system is prone to various problems. These problems can be seen from the following perspectives like performance, information, economic, control, efficiency and services given by the existing system to the users, by using the PIECES framework as follows.

The **performance** of any system is required to exhibit to meet the needs of users of that system. The current system's performance is weak. This is due to the following reasons: - first the acceptable throughput rate is relatively high i.e. the time required from initiation to completion of a particular task is relatively high. For example during arrangement of buildings for the allocation it may take a week or more due to its manual operation. Second is the acceptable response time for a particular task is large.

Information- the main input for the current system is student record and records of different dormitory materials which enable the system to rearrange students and buildings for the allocation. Based on this the system rearranges and allocates dorms for students at the beginning each academic year and generates the allocation report which may be viewed by the students as well as the management. After this the students are required to submit their personal information which will be stored on papers to be viewed latter if it is necessary. The other data that is stored is record of materials associated with the dormitory. The system manipulates and manages all of these and other records manually on papers.

Economical- AMU DMS performs all of its tasks manually which requires much of the work to be done by a man power, and it a huge amount of papers for the manual storage of data on papers, which lead the manual system to spend much money for human resource and for purchasing papers and other materials.

Controlling- since all the records associated with the manual system are recorded and stored manually the security

that the system provide for the privacy of this records is not good. The system shouldn't provide sufficient protection for access and manipulation of the records associated with the system.

Efficiency- due to the manual operation most of the activities are prone to wastage of resources like papers, man power, time etc. to produce the corresponding outputs. This makes the current system inefficient while utilizing resources. There should be a mechanism that reduce wastage of resources and that make the system to be efficient.

Services- the main users of the current system are students and the management itself. The services given to users are not flexible, reliable and expandable i.e. the users must there in the campus to get the services given by the system. Those services given by the system are limited to a particular area.

2.7 Practices to be preserved from existing system

Here most of the main activities that are performed in the manual management system will be preserved by designing the corresponding simulation of those activities. That is each activity that is pertinent to the system are designed and automated to achieve the best functionality. The team may also add additional features to the existing system. These additional features will add other additional functionalities to the current system that will change the way the current system operates, and services given to its users.

2.8 Alternative options to address Problems of existing system

To address the problem of the existing system the team has studied and analyzed the problems of the existing system. To come up with the possible solutions associated with those problems a group will just follow the natural system development methodologies as the only option for addressing and solving those problems associated with the current system.

2.9 Options analysis and the proposed new system

Even though an existing system provides different functions that are stated above, it is not to mean that the functions are satisfactory. This is because all the processes (actions) are performed manually. To overcome or improve this manual operation the team comes up with a new Dormitory Management System entitled AMU-Dormitory Management System. This new system is a Web based application that enables the users to access the services given by the system through the Internet.

The proposed new system operates in the following manner. First it accepts all inputs from a body which it concerns. For example in case of new student (first year) it takes input from dean of students that is students list, incase of other students it take form dean of students and will be feed to the system by proctors. This feeding of data will be performed based on their year, department, faculty and gender. After all data were collected and given to the system, it will rearrange students for the allocation. After doing this the system will generate the allocation report which contains dormitory information like student's name, id number, dorm number, and block number. This report will be released online for the student so that they can access this information by entering his name and registration number on the webpage provided by the system just by sitting where ever they are.

Once students get their dormitory information they will be expected to fill their personal information on the form provided feed this data to the system. As they arrived, students will be expected to fill the property form which specifies list of dormitory materials that the students will use. All the corresponding records of the above activities other are recorded and stored in the database.

So now everything is recorded and performed. The next thing to be performed is the management of the property. Here a

proctor will perform periodic checking for the dormitory materials. If a proctor found any property crashed/damaged he will immediately record that material, a person who did so by his name, id, dorm number, and block number. So the system having this information will generate a report about a person's status. In case a student wants any clearance and contact the proctor, a proctor will recall the report that is generated above and forces a student to charge what he crashed. The same but different approach will be performed in case of discipline case report.

2.10 System requirements of the new system

The following are Functional and Nonfunctional requirements of the proposed new system that a group member have identified from requirement use cases associated with each actor and use case interaction.

a. Functional requirements

The following are the functional requirements of the new system.

- ✓ The system should rearrange the buildings for the allocation.
- ✓ The system should rearrange students for the allocation.
- ✓ The system should assign dorms for students.
- ✓ The system should assign proctors for buildings.

- ✓ The system should generate timely report about the allocation.
- ✓ The system should store all the data related with all the tasks performed into a database.

b. Non-functional requirements

The following are the non functional requirements associated with the new system.

- ✓ The system must be error free while operating with a huge set of data.
- ✓ The system must be user friendly
- ✓ The system must be able to communicate users at different location.
- ✓ The system must recover immediately when a user enters erroneous data.
- ✓ The system must have a good response time.
- ✓ The system must be compatible with any environment.

CHAPTER THREE
Modeling (of the existing and proposed system using the chosen methodology)

Introduction

As mentioned above, in this project, the team used an object oriented system development methodology which incorporates two principal phases. In this chapter, what the team will do is the **object oriented analysis (OOA)**.

During Object Oriented Analysis the following major activities are performed.

- ❖ ***Modeling the Functions of the system (Use Case Modeling)***

The main activities that are performed in this part are:

- Identifying if there is any additional actors and use cases,
- Constructing a use case model, and
- Documenting the use case course of events.

❖ **Finding and identifying potential business objects**

❖ **Organizing the Objects and Identifying the Association between them**

3.1 Use Case Modeling (*Modeling the Functions of the system*)

The **first step** is to identify Actors and use cases associated with the system. The following table specifies the actors and use cases that a group member have identified with in the proposed new system. The table also describes use case descriptions associated with the corresponding use cases.

Table1. List of **Actors** and **Use Cases** associated with the new system.

Actor	Use case	Use case description
Student Dean	**Submit student list**	Submits list of students to be allocated to dormitory management.
Proctor Manager	**Manage proctors**	Manages and controls proctors.
Proctor Manager	**Arrange buildings**	Rearranges buildings for the allocation.
Proctor	**Arrange students**	Rearranges students to be allocated.
Proctor	**Material control**	Controls the dormitory materials.
Proctor	**Allocate dorms**	Allocate dorms for the students
Student	**Submit personal information**	Submits his/her personal information.
Time	**Generate report**	Generate a timely report.

The **second step** is to construct the **use case model** which graphically depicts the interaction of the system with the external environment. The following figure specifies the use case model of the system.

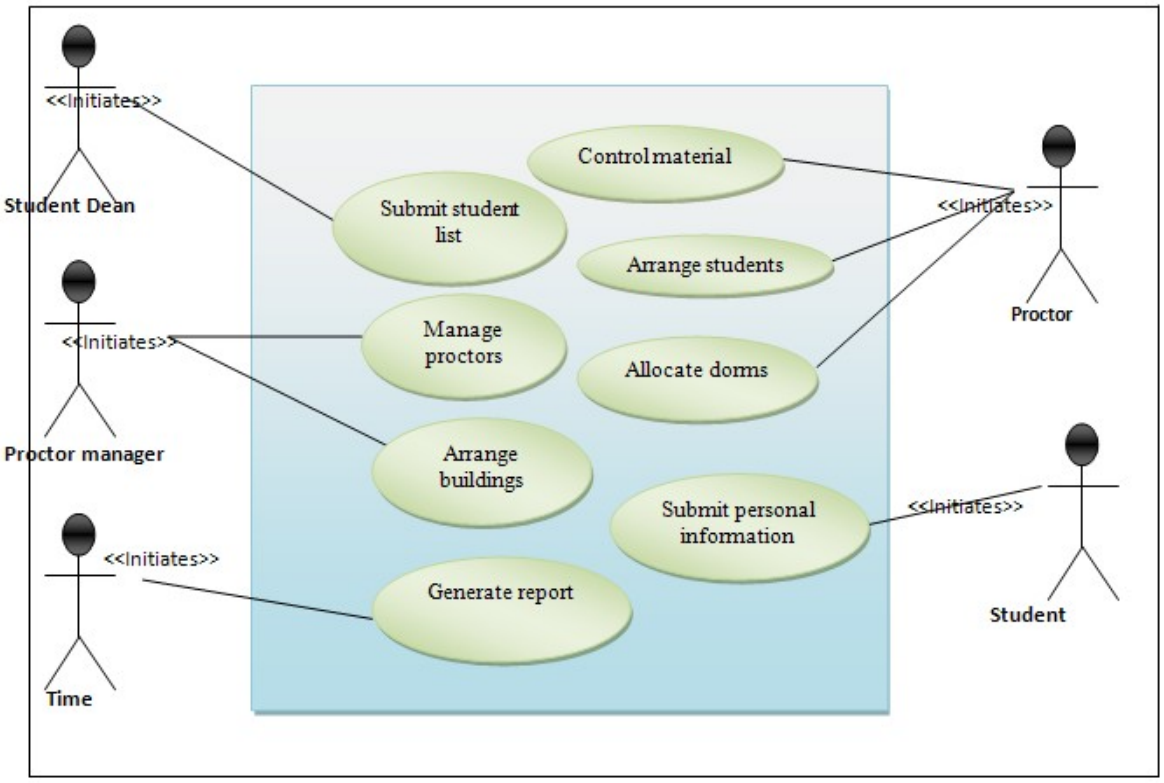


Figure1: use case diagram of *AMU-DMS* system.

The **third step** is to document each of the above use case courses of events to determine the requirement use cases as described in the following section.

3.2 Use Case Documentation of each Use Case

The following consecutive tables show the use case documentation for each of the use cases that has identified in the above use case diagram. Each table contains the use case name, the actor which initiates and interacts with the use case, description of the use case and typical course of events that show the interaction between the actor and the use case which enable the team to easily depict the functions of the proposed system.

•Use case documentation **for "Submit Student List"** use case.

Use case Name:	**Submit Student List**	
Actor(s):	dean of students	
Description:	This use case describes the process of a registrar submitting list of students to the dormitory management. On completion, the registrar will be sent a notification that list was accepted.	
Typical course of events:	**Actor Action** **Step1:** this use case is imitated when a registrar submits a student list to be processed. **Step5:** this use case concludes when the registrar receives the conformation notice.	**System Response** **Step2:** The number of students on the list such as number of male, female students validated against what is currently on file. **Step3:** Validated student list will be recorded to the data base. **Step4:** Invoke a use case arrange students to arrange students based on list.
Alternate courses:	—	
Precondition:	List of Students only submitted by Student Dean to proctors.	
Post condition:	List of Students has been recorded and list of students is routed to proctor.	

•Use case documentation for **“Submit Personal Information”** use case.

Use case Name:	**Submit Personal Information**	
Actor(s):	**Student**	
Description:	The use case describes the process of a student submitting personal information to the dormitory management system. On completion, the student will be sent a notification that the information accepted.	
Typical case of events:	**Actor Action** **Step1:** this use case is initiated when a student submits his personal information. **Step6:** This use case concludes when the student receives allocation confirmation notice.	**System Response** **Step2:** The student personal information such as his addresses and phone no is validated. **Step3:** The student personal information is checked and recorded to make sure that invalid information is not Recorded. **Step4:** Invoke the material control use case to allocate Dormitory materials. **Step5:** Generate the status confirmation notice indicating the status of allocation and send it to the Student.
Alternate courses:	**Step2:** If student submits invalid information such as incorrect Id number, send a notification to the student to submit valid information.	
Pre condition :	Student personal information is only submitted by a student.	
Post condition:	Students personal; information has been recorded and material control use case will be processed.	

- Use case documentation for **“Arrange buildings”** use case.

Use case Name:	**Arrange buildings**	
Actor(s):	**Proctor-manager**	
Description:	This use case describes the process of a proctor-manager arranging the buildings for the allocation. On compilation, the proctor will be sent a notification that the building was arranged.	
Typical case event:	**Actor Action** **Step1:** this use case is initiated when a proctor-manager select the building number (block no) to be arranged. **Step6:** this use case concludes when the proctor manager receive a building status confirmation notice.	**System Response** **Step2:** for each building selected validate the building number. **Step3:** Display building status, total number of dorms capacity etc. **Step4:** Invoke the extension use case assign proctor. **Step5:** Generate a conformation notice indicating the status of a building and send it to proctor manager.

Alternate courses:	**Step2:**If a proctor manager enter an invalid block no, send a notification to the proctor manager to enter a valid block no.
Precondition:	A building can only be arranged by a proctor manager.
Post condition:	A building status has been recorded.

- Use case documentation for **"Arrange student"** use case.

Use case name:	**Arrange student**	
Actor(s):	**Proctor**	
Description:	This use case describes the process of the proctor arranging the student for the allocation. On completion, the proctor will be sent a notification that students are arranged.	
Typical course events:	**Actor Action** **Step1:** this use case is initiated when the proctor need to arrange students by using their faculty. **Step6:** this use case concludes when the	**System Response** **Step2:** retrieve list of students with in the corresponding faculty. **Step3:** categorize the list of gender, department, year, and section. **Step4:** Invoke extension use case allocate dorms. **Step5:** generate confirmation notice, and Send it to proctor.

	proctor receive the confirmation notice.	
Alternate courses:	___	
Pre condition:	The students are not arranged	
Post condition:	The arrangement has been recorded.	

- Use case documentation for **“Allocate Dorms”** use case.

Use case Name:	**Allocate Dorms**	
Actor(s):	**Proctor**	
Description:	This use case describes the process of a proctor allocating dorms for the students. On completion, the proctor will be sent a notification that the allocation is performed.	
Typical course event:	**Actor Action** **Step1:** this use case is initiated when the proctor assigns dorm for students by selecting a block by block no. **Step6:** this use case concludes when the proctor receives confirmation notice.	**System Response** **Step2:** for each block selected validate the block no. **Step3:** display building status such as, total no of dorms, dorm capacity, etc. **Step4:** assign students into each dorm. **Step5:** generate a confirmation notice indicating status of allocation and send it to the proctor.
Alternate courses:	**Step2:** If block no is not invalid, send a notification to proctor.	
Pre condition:	Dorms are not allocated (assigned).	
Post condition:	The allocation will be recorded and allocation report will be generated.	

- Use case documentation for **“Control material”** use case.

Use case Name	**Control material**	
Actor(s):	**Proctor**	
Description:	This use case Describes the process of a proctor controlling dormitory materials found in the dorm. On completion, the status of all the materials in the dorm is recorded for further controlling.	
Typical course event:	**Actor action** **Step1:** This use case is initiated when a proctor performs a periodic checking for the dormitory materials by entering the dorm number. **Step5:** this use case concludes when the proctor receives confirmation notice indicating the status of the material is updated	**System Response** **Step2:** Display dormitory materials found in that specific dorm. **Step3:** Display previous status of that material. **Step4:** Update the status of the dormitory material.
Alternate courses:	**Step2:** If Dorm number is invalid, send a notification to proctor.	
Pre	The proctor doesn’t know about the dorm materials	
Post	The Damaged material will be recorded	

- Use case documentation for **“Manage Proctor”** use case.

Use case Name	**Manage Proctor**	
Actor(s):	**Proctor Manager**	
Description:	This Use case describes the proctor-manager managing proctors by assigning proctor for a building and recruitment of a proctor.	
Typical course event:	**Actor action** **Step1:** This use case is initiated when a proctor- manager selects a proctor by entering proctor-ID. **Step3:** select a building for a proctor to be assigned. **Step5:** Assign a proctor for the building being selected. **Step7:** This use case concludes when the Proctor Manager receives confirmation notice.	**System Response** **Step2:** Display the status of a proctor being selected. **Step4:** Display status of the building. **Step6:** Generate a confirmation notice indicating status of allocation and send to the Proctor-manager.
Alternate courses:	**Step2:** If proctor-ID is invalid, send a notification to proctor-manager.	
Pre condition:	The proctors are not assigned to the blocks.	
Post condition:	Proctor status will be recorded.	

- Use case documentation for **“Generate Report”** use case.

Usecase Name	**Generate Report**	
Actor(s):	**Time**	
Description:	This use case describes the processes of Generating status report based on the time and the type of report needed.	
Typical course event:	**Actor action** **Step1:** This use case is initiated when a time for generating a report is reached. **Step5:** This usecase concludes when student and proctor get reported.	**System Response** **Step2:** Generate Allocation report **Step3:**Generate discipline measure report **Step4:** Generate student status report.
Alternate courses:	____	
Pre condition:	Reports are only generated when the time of generation is reached.	
Post condition:	Report regarding a particular task is generated.	

3.3 Sequence Diagram

The following figure depicts the high level sequence diagram of the system. The figure depicts the high level interaction of the actors with the system that specifies the work flow the system.

Figure 2: High level Sequence diagram of the system.

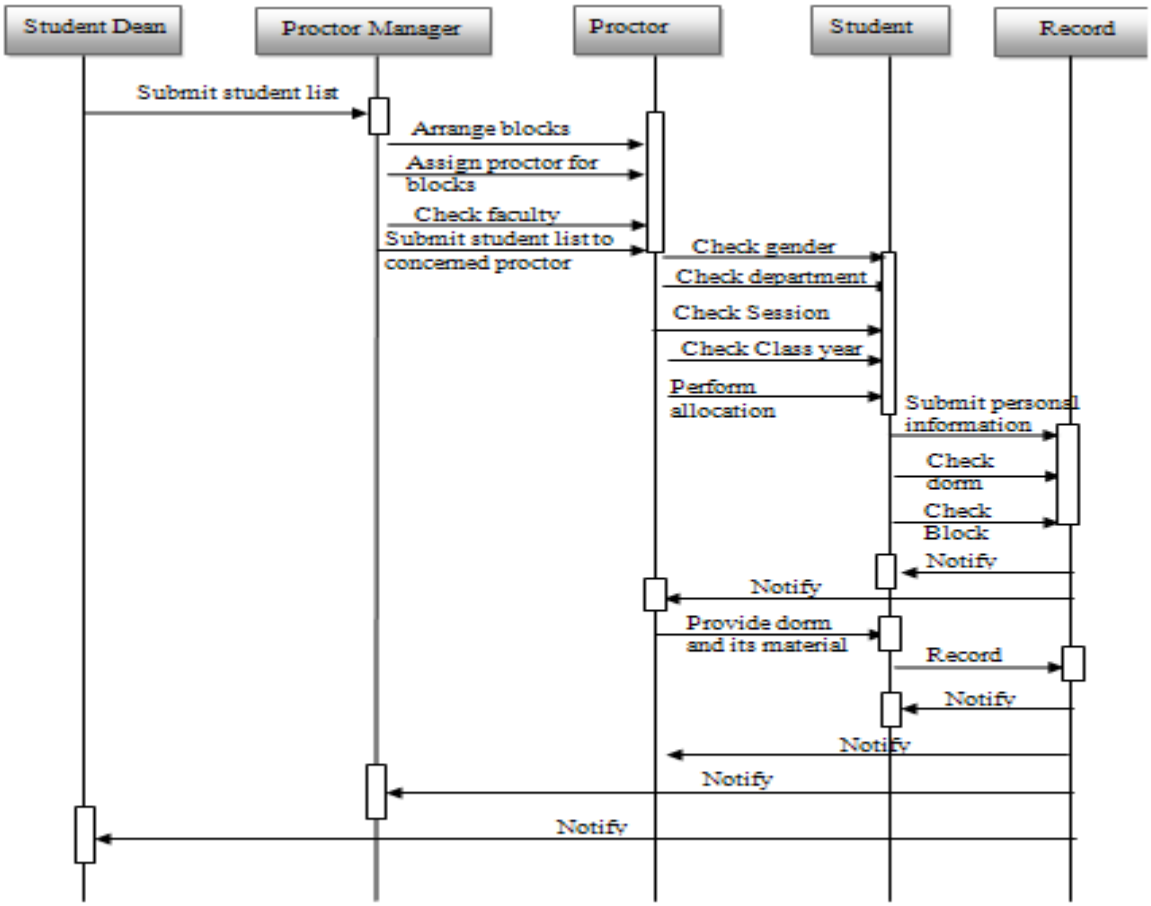


Figure 2: High level Sequence diagram of the system.

-
-
-
-
-
-
-

➢ Sequence Diagram for some processes performed by the actors are listed below

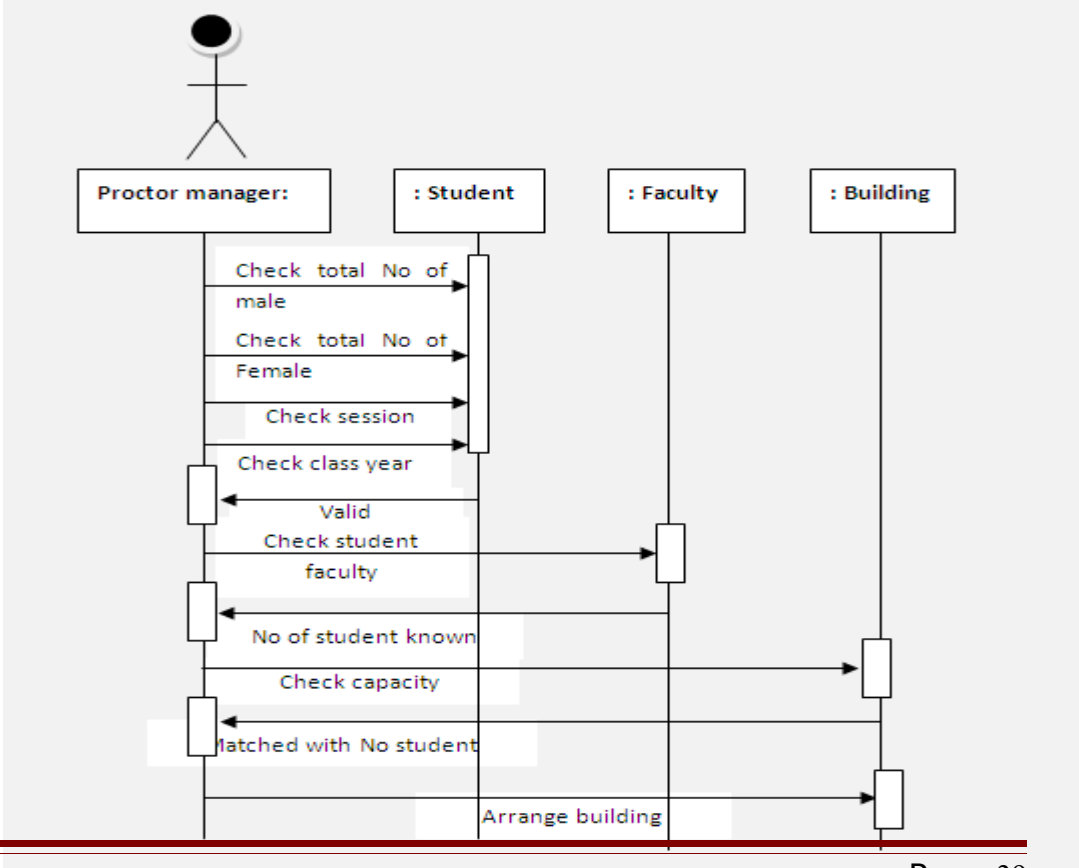


Fig: Sequence Diagram for Building Arrangement

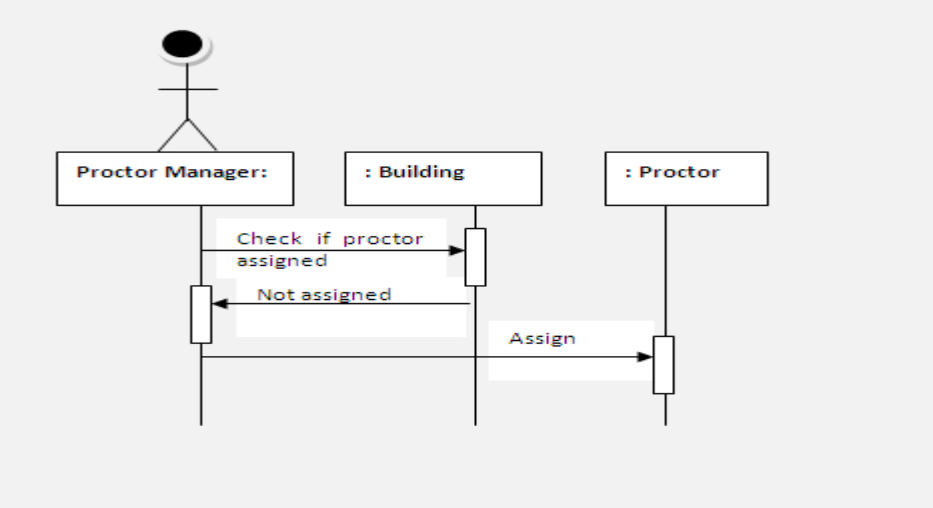


Fig: Sequence Diagram for Assigning Proctor

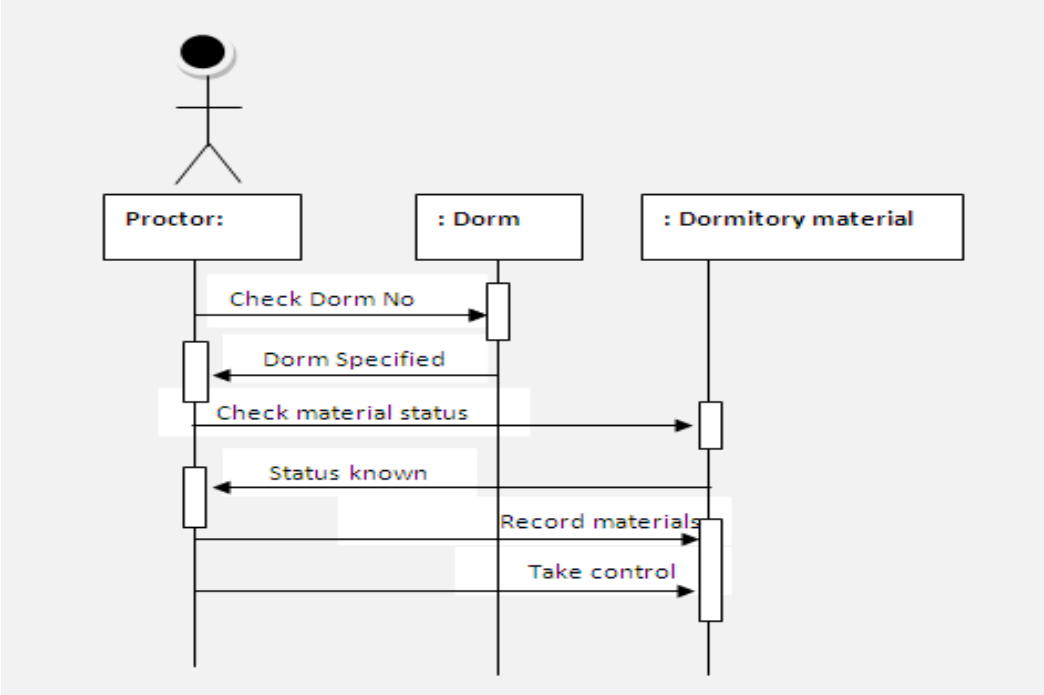


Fig: Sequence Diagram for Controlling Dormitory material

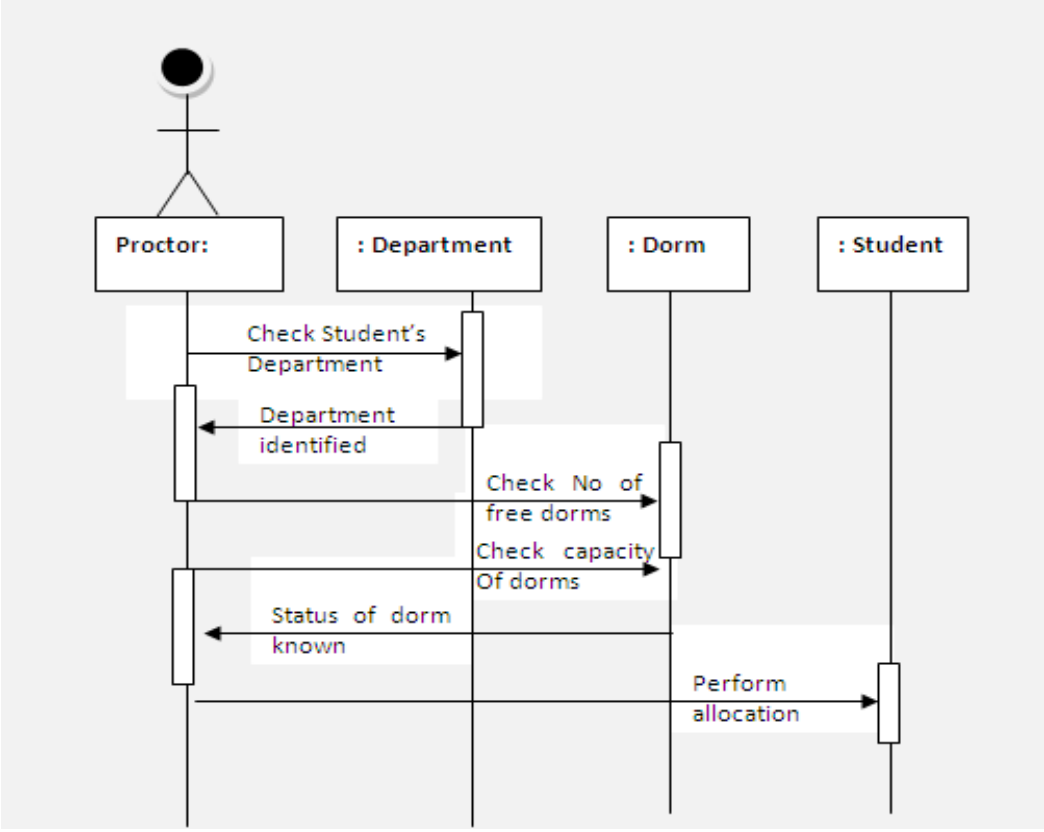


Fig: Sequence Diagram for Allocating Student

Finding and identifying potential business objects

The main activity that is performed here is to identify the potential business objects associated with the above use case scenarios. This is done by selecting each **noun** or **noun phrases** in the use case documentation and checking whether they describe an object or entity in the scope of the problem domain. The following are the main potential business objects associated with the above use case scenarios. They are: -

Building **Dorm**
Proctor **Student**
Dormitory-material **Faculty**
Department

Organizing the Objects and Identifying Association between them

In this step the team used to identify and establish the association between the proposed business objects. The team used the following decision table to perfectly identify the relation

ship between the objects. The table also depicts the multiplicity of the association between each object.

Table2. A decision table to identify r/ship b/n the objects

	Faculty	**Department**	**Student**	**Proctor**	**Building**	**Dorm**	**Dormitory material**
Faculty	Is	1-contains-(1..n)	1-enrols-*	X	X	X	X
Department	(1..n)-in-1	Is	1-enrols-*	X	X	X	X
Student	*-enrols-1	*-are in-1	Is	*-assigned by-1	*-live in-1	(1..n)-live-1	X
Proctor	X	X	1-assigns-*	Is	1-manages-(1…n)	1-allocates-1,*	1-controls-*
Building	X	X	X	(1..n)-managed by-1	Is	1-contains-..n	X
Dorm	X	X	1-have-(1..n)	*-allocated by-1	n-contained by-1	Is	1-has-*
Dormitory material	X	X	X	*-Controlled by-1	X	*-occupied by -1	Is

X: indicates that the objects have no direct relation with each other.

3.4 Analysis level class diagram (conceptual modeling)

Once the team has identified the association and the corresponding multiplicity of association of the objects of the proposed new system, the next step will be to draw the class diagram. The next sub topic shows the class diagram of the new system.

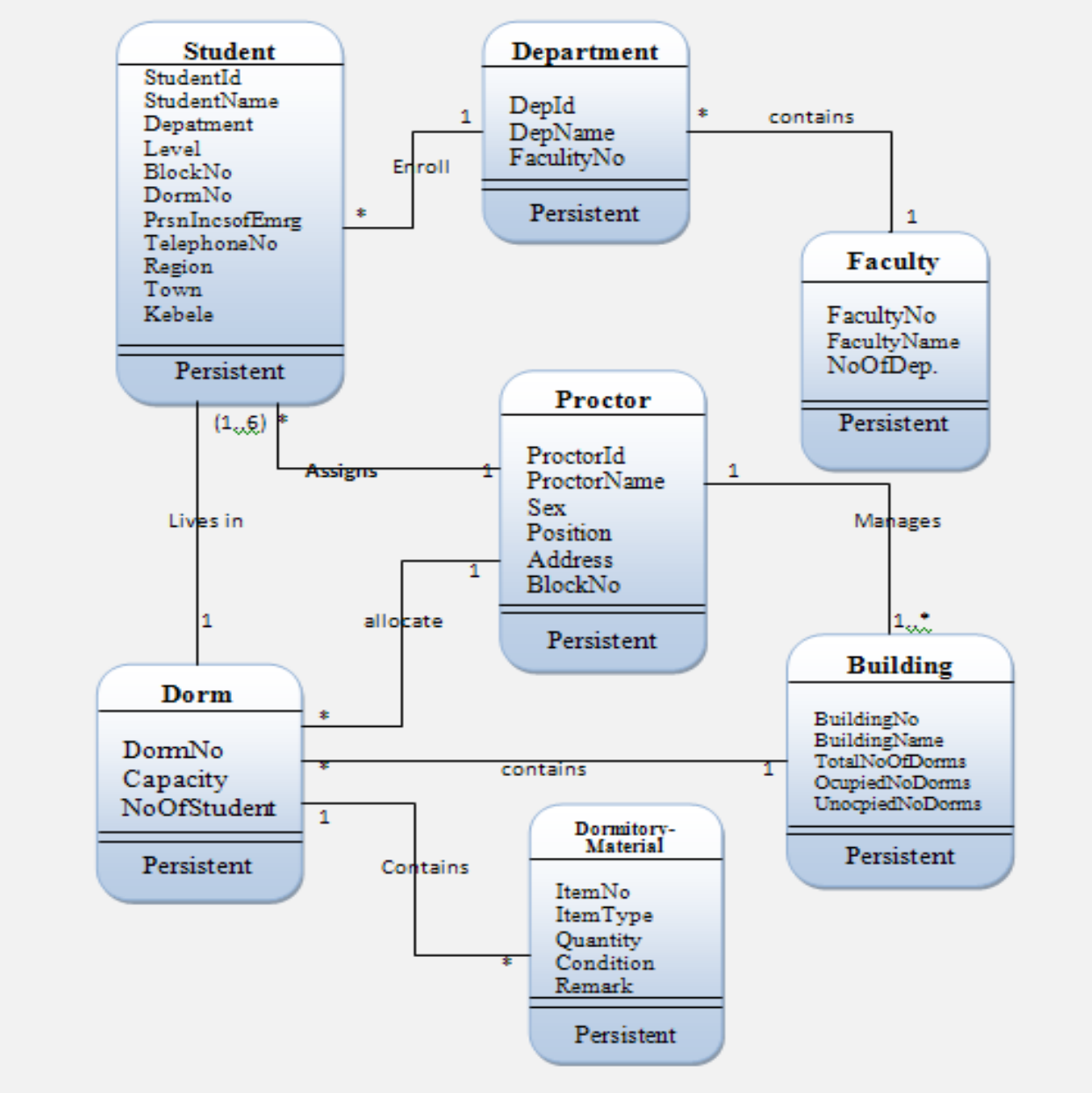


Figure 3: Class Diagram of the new system**.**

3.5 User Interface Prototyping

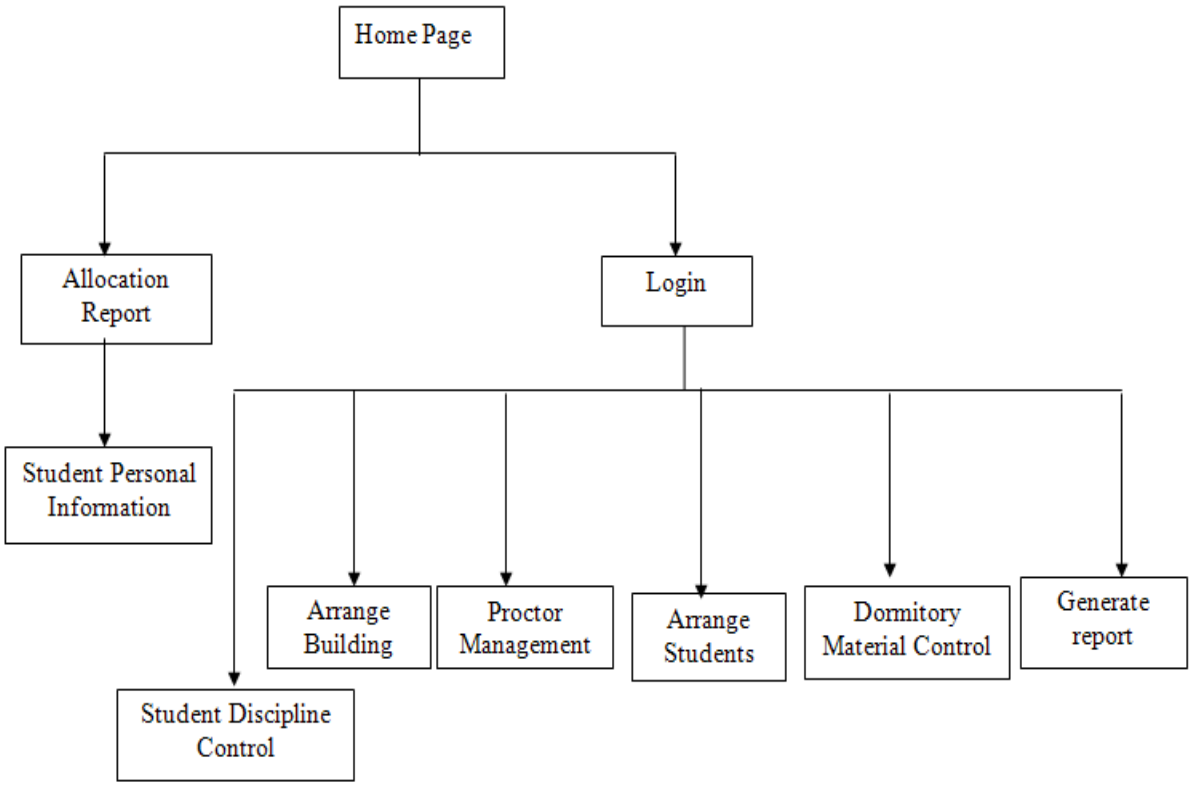


Fig4: User Interface Prototyping of the new system

3.6 Supplementary specifications

Business Rules

The business rules is a principle or a policy in which the proposed system operates accordingly. It deals with access control issue. Some of the business rules that have included in this project are the following:

- Any student should submit his personal information after watching the allocation report.
- Any student should not be assigned a dorm more than ones.
- Only a proctor manager should arrange buildings for the allocation.

- Only a proctor manager should assign proctors for the building.

Chapter Four

Design

4.1 Introduction

As mentioned above, in this project, group members used an object oriented system development methodology which incorporates two principal phases. In this chapter, what the team will do is the **object oriented design (OOD)**.

During Object Oriented Design the following major activities are performed.

- **Refine the use case model to reflect the implementation environment,**
- **Model object interactions and behaviors that support the use case scenario,**

4.2 Class Type Architecture

In this project the team uses a **three tier architecture** which has three layers. These three layers are the **Application or Presentation layer**, the **business layer** and the **data access layer**. **Application or presentation layer** is the form which provides the user interface to either programmer or end user. The **business layer** is the class which the team uses to write the function which works as a mediator to transfer data from application layer or presentation layer to data layer. This layer also has a **property layer** which is a class where variables are declared corresponding to the fields of the database which can be required for the application and make the properties so that the team can get or set the data using these properties into the variables. The third tire is the **data access layer** which is also a class to get or set data to the database queries back and forth.

This layer only interacts with the database. The database queries or stored procedures will be written here to access the data from the database or to perform any operation to the database.

The following diagram shows the class type architecture of the system.

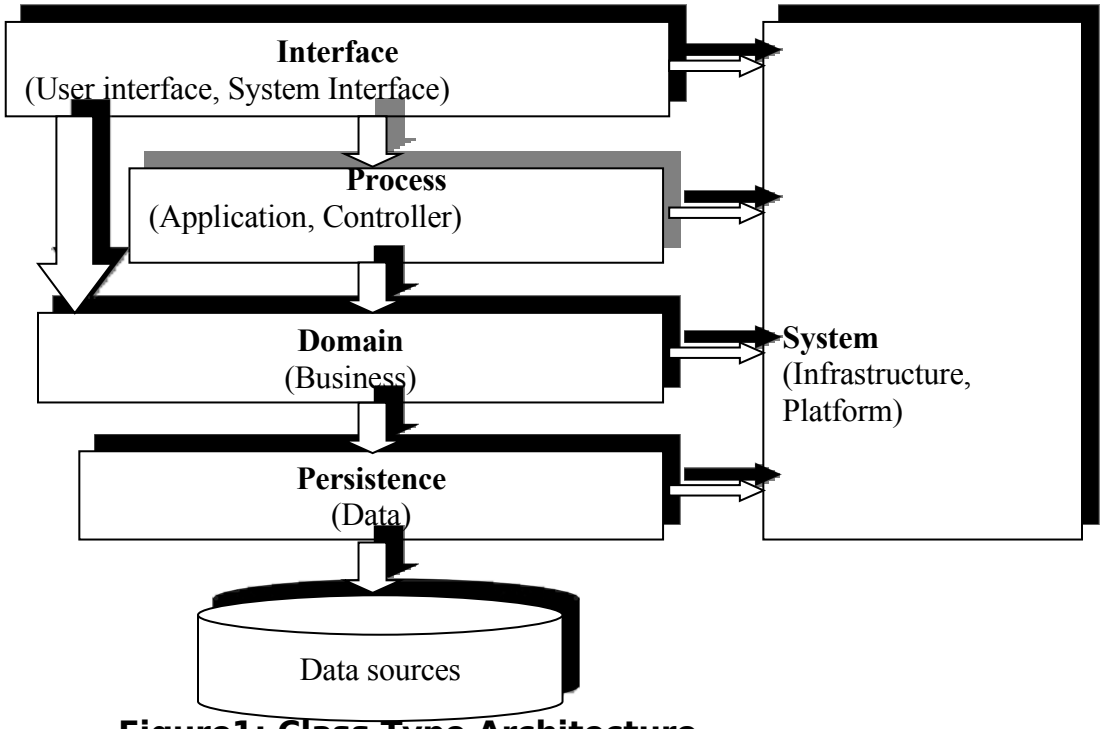


Figure1: Class Type Architecture

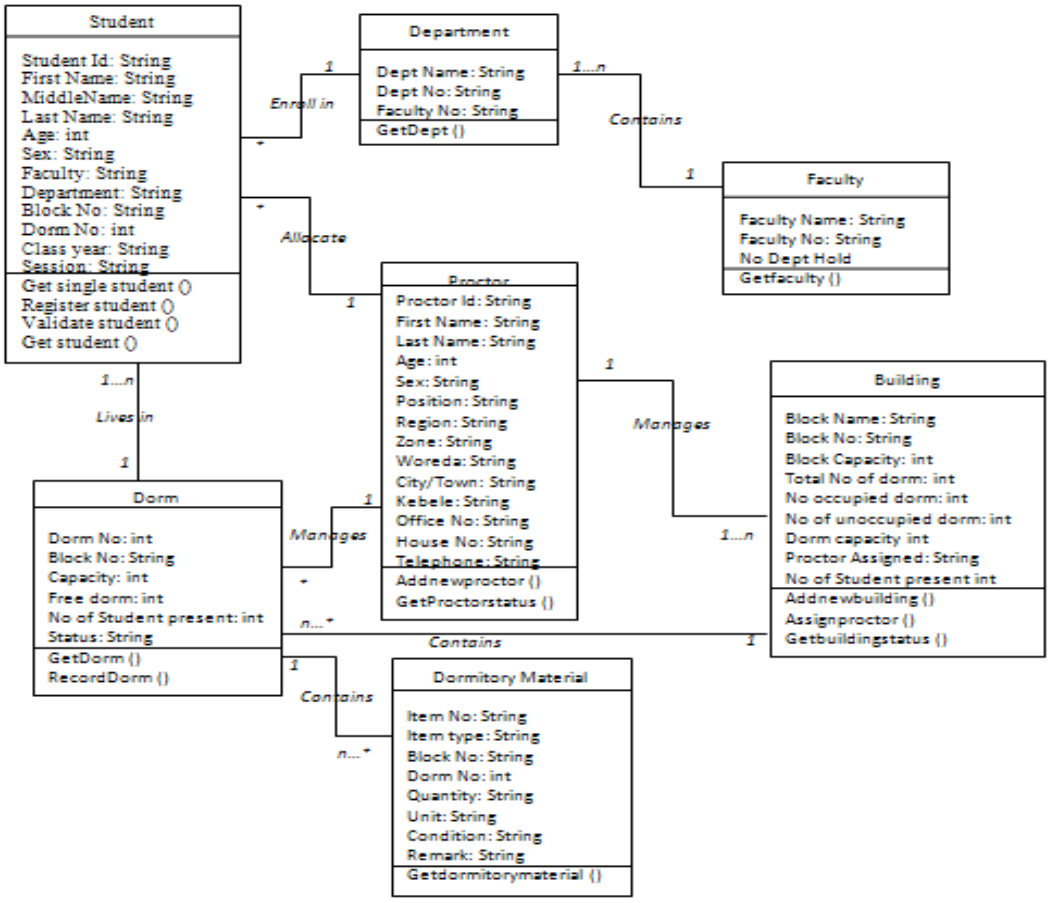
4.3 Class Modeling

During object oriented design in order to model the classes a group member has followed the following steps.

- **Refine the use case model to reflect the implementation environment**

This step is used to transform “Analysis use case” into the “Design use case” and update the use case diagram and other documentation to reflect the new use case functionalities. Then a class diagram will be constructed.

Figure2: Class diagram of the new system.



-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

- **Model object interactions and behaviors that support the use case scenario**

This step is used to identify and classify the use case design objects. There are three types of design objects. These are the

interface, ***control*** and ***entity*** **objects**. The following table lists the newly found objects.

Table3: list of design objects of the proposed new system

Interface Object	Control Object	Entity Object
AMU-DMS main window	**Allocation processor**	**Proctor**
Assign proctor window	**Report generator**	**Department**
Student personal information window		**Student**
Allocation report window		**Building**
Allocate student main window		**Dorm**

The above table lists the design objects of the system. The next step is to identify the attributes of the objects and model the high level interaction of the use cases. The following UML diagram depicts the high level object interactions of the objects in the "**Allocate students**" use case to show the functionalities of the system.

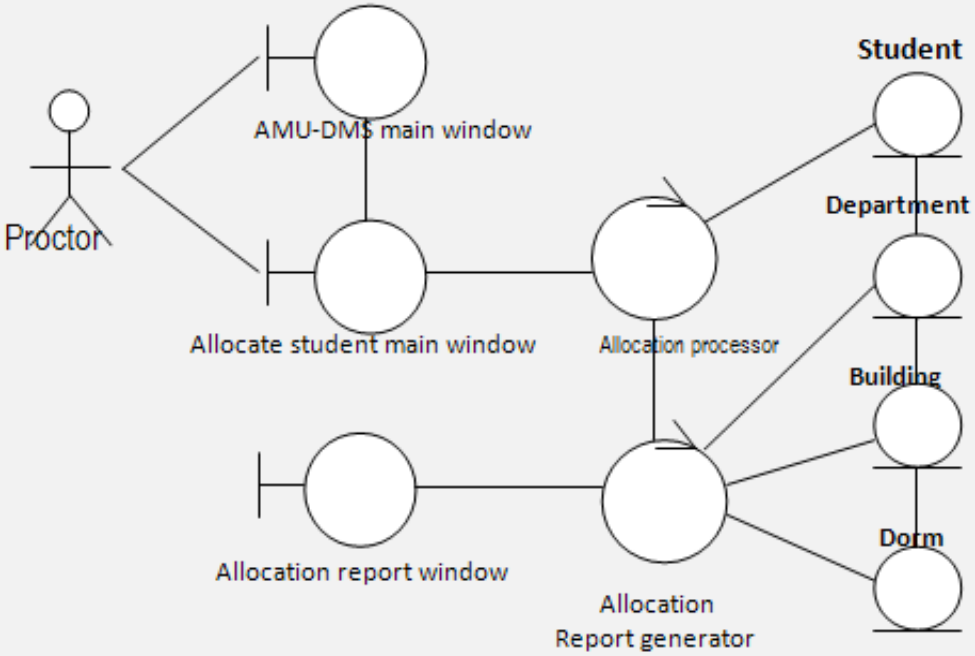


Figure2: Partial ideal object diagram for "Allocate students" use case

4.4 State Chart Modeling

In this part the team used to model the behaviors of the objects by drawing the state diagram. The state diagram depicts the state of objects as their attributes change from one state to the other state. State chart modeling is used to show the sequence of states that an object goes through, the events that cause the transition from one state to the other and the actions that result from a state change. The following figure shows the state of the objects of the corresponding use cases.

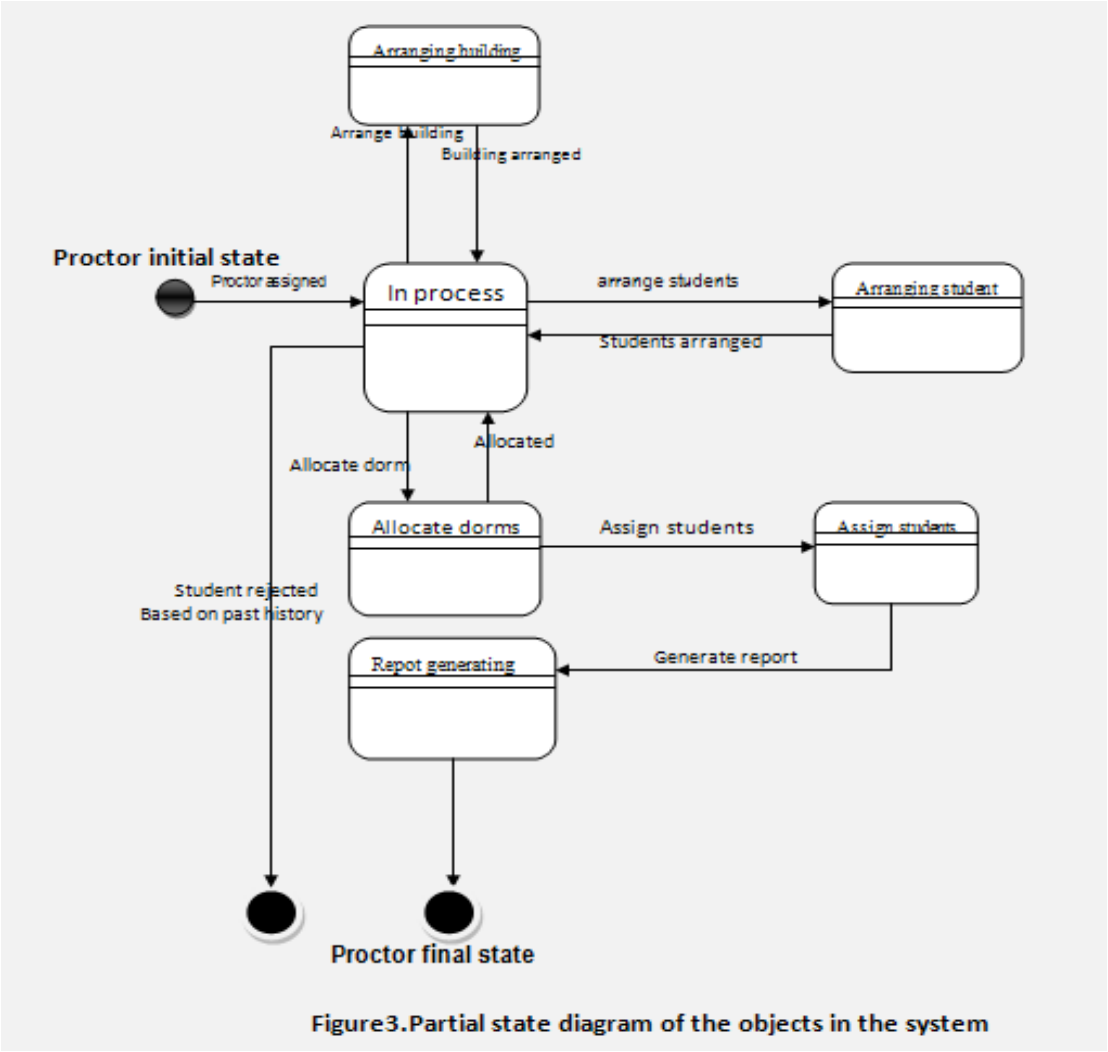


Figure3.Partial state diagram of the objects in the system

4.5

4.6

4.5

4.6

4.7

4.5 Collaboration Diagram

This is used to show some data flows between objects and the interaction caused between them. Some of the data flows among objects were explained below.

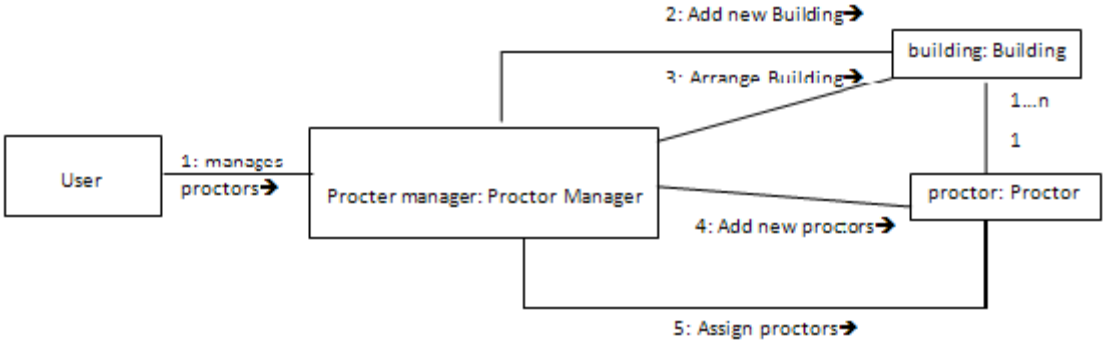


Figure: Collaboration diagram for the "Proctor Management" flow

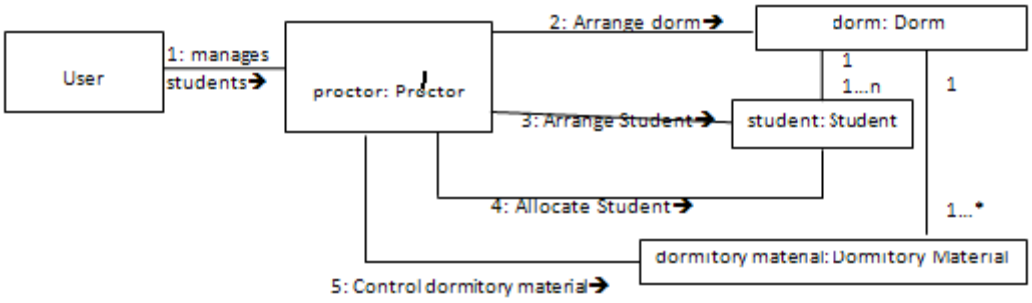


Figure: Collaboration diagram for the "Proctor" flow

4.6 Component Diagram

In this Diagram components of the system will be wired showing that there is relation among components, management of the system, database and operations performed on databases such security issue. This in some extent shows which component or objects will be accessed by whom and what type of security infrastructures it is using. The diagram is simulated below.

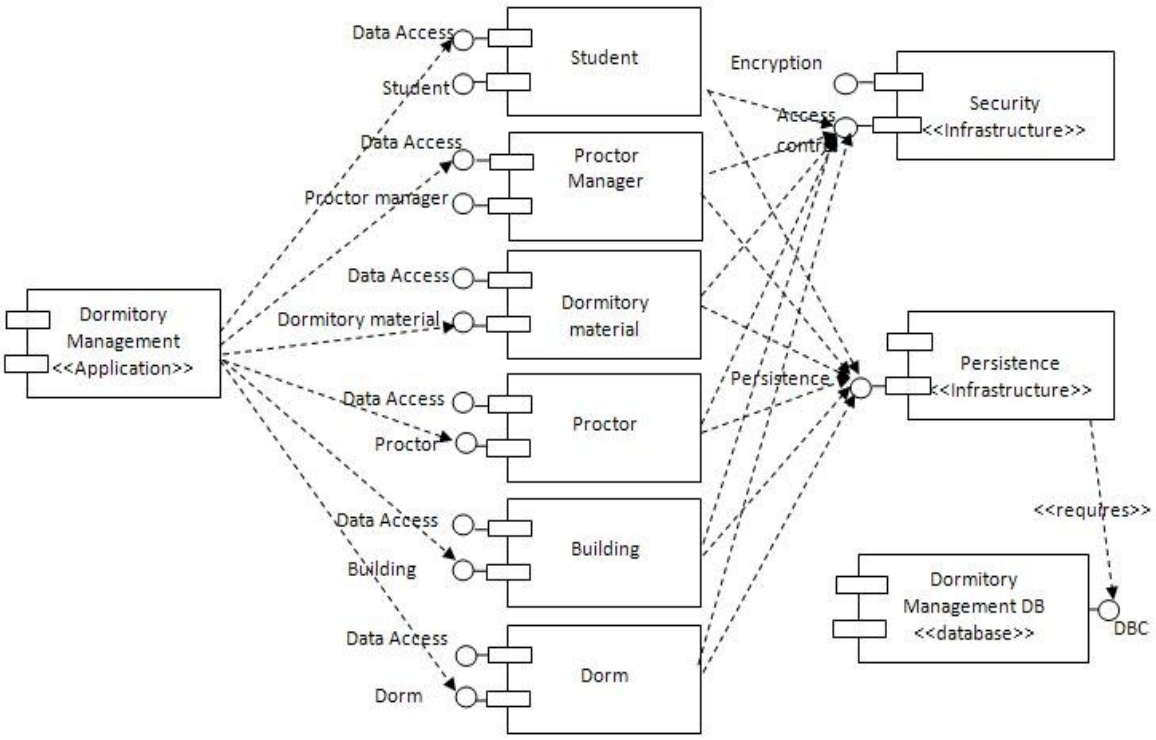


Fig: Component Diagram

4.7 Deployment Modeling

Deployment modeling is used to show the hardware of the system, the software that is installed in the hardware and also the middleware that is used to connect the disparate machines to one and other. It also shows how the software and the hardware components work together.

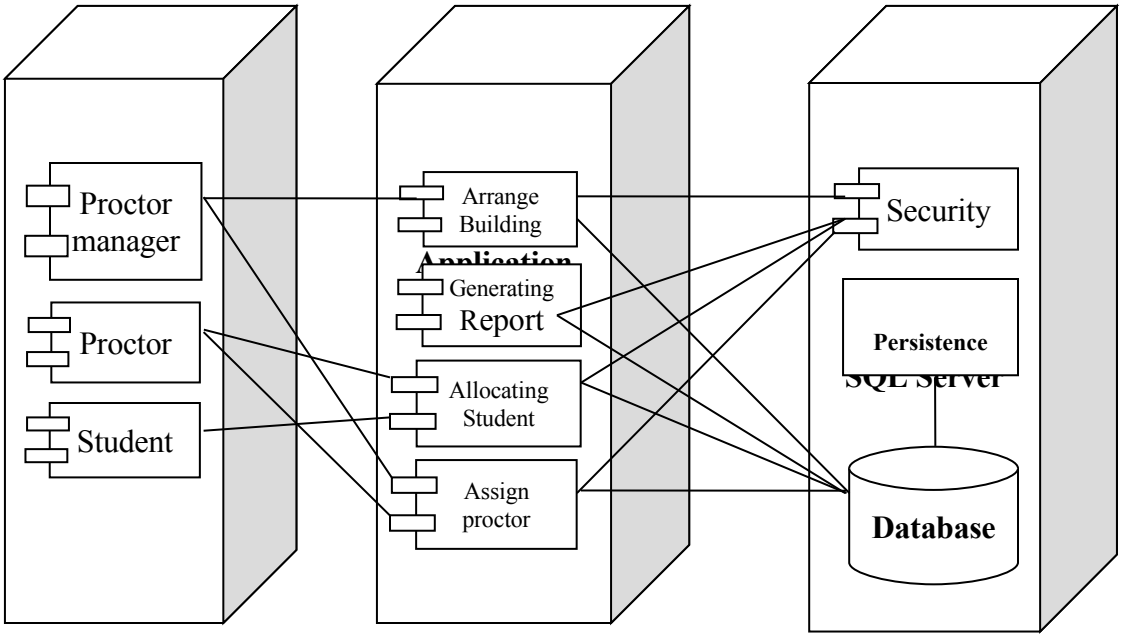


Figure4: deployment modeling

4.8 Persistence Modeling

Persistence modeling is used to communicate the design of the database, usually the data base to both the users and the developers. It is also used to describe the persistence data aspect of the system. The following diagram indicate the persistence diagram of the system

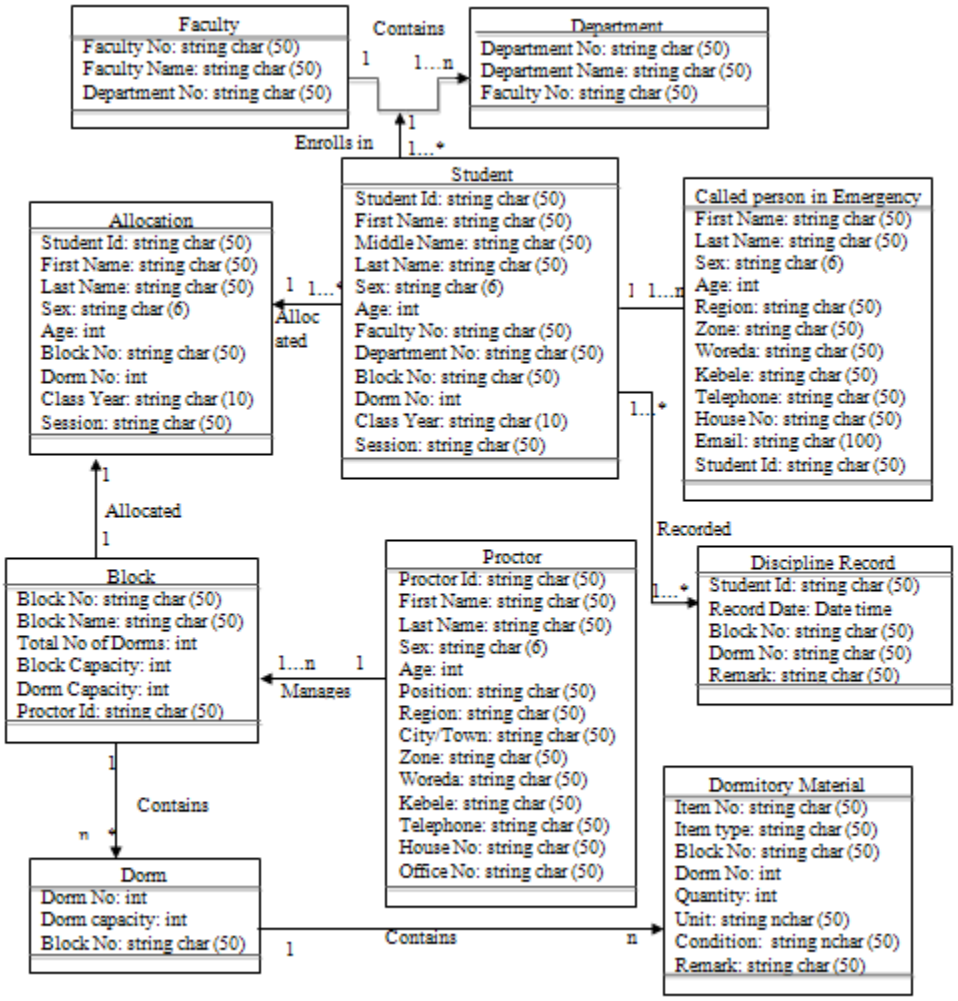
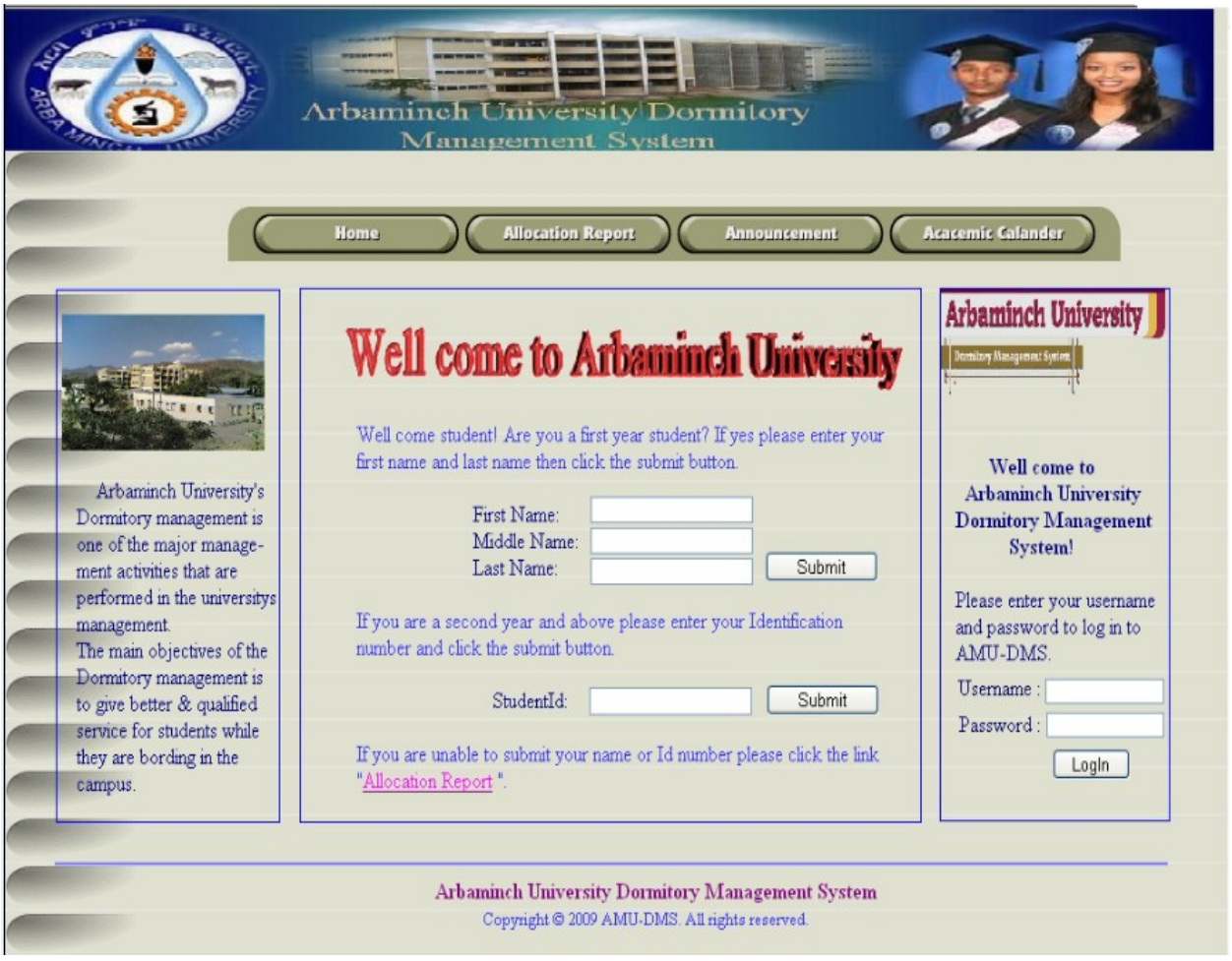


Figure5: Persistence Modeling Diagram

4.9 User Interface Design

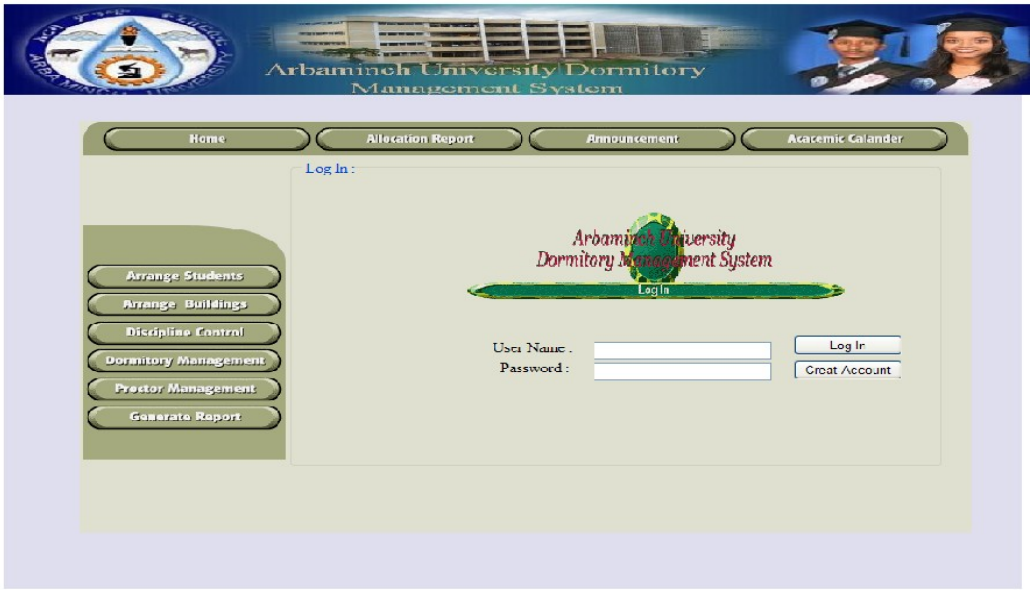
In this system users will communicate with it through the following user interfaces.

I. **Home Page:** This form contains some links which lead it to the concerned page, and if the user has an account he/she will directly go to concerned page by entering their username and



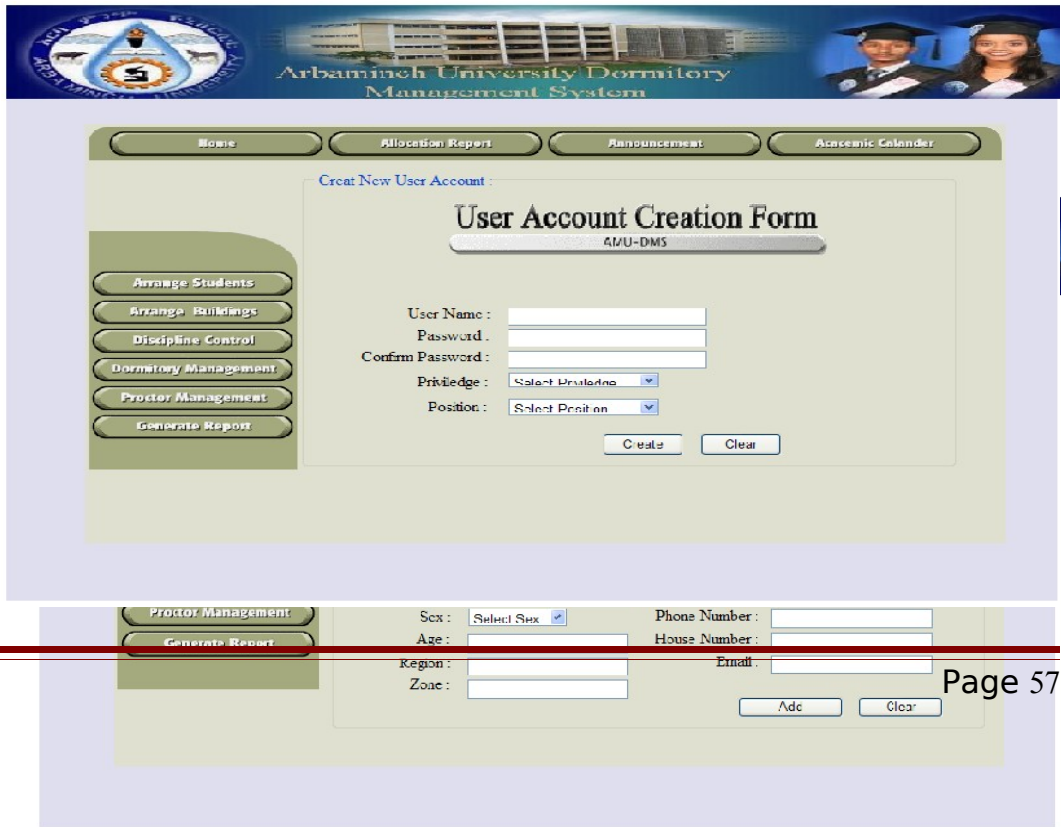
password.

II. **Log In form:-**this form found immediately following the home page. Home page appears as the site on which the system is deployed is opened. Only proctor and proctor manager will have their own password. Those forms appeared using password and user name will not accessible by other persons



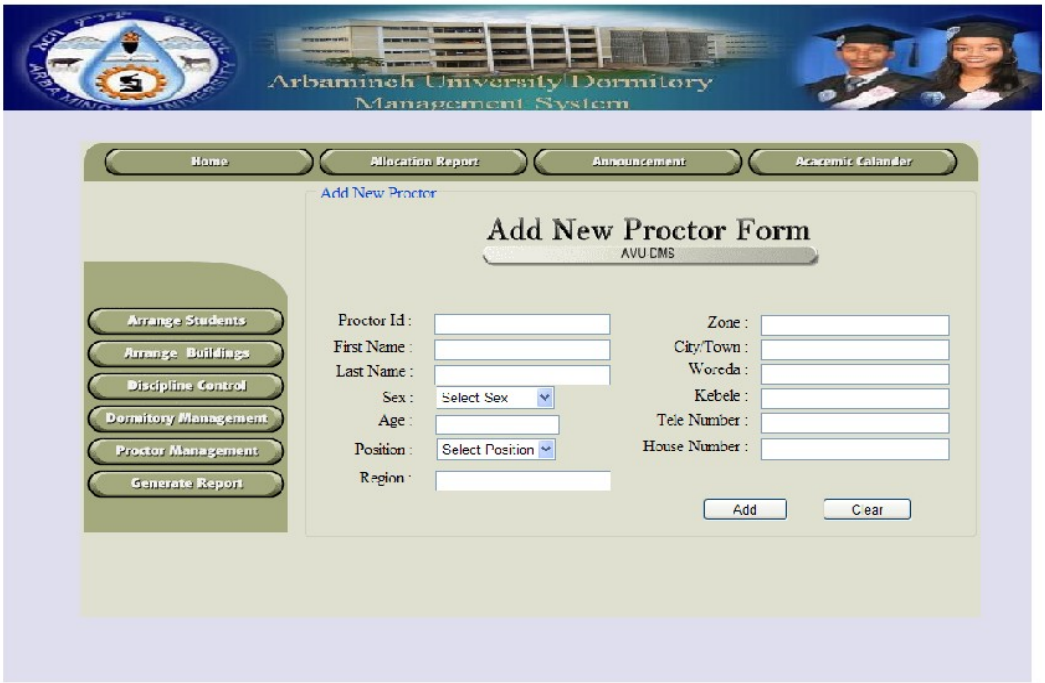
except for those who have privilege.

III. **User Account Creation form:** In this form Users granted to have an account in a system will create his/her account. This account is granted by a body having privilege.

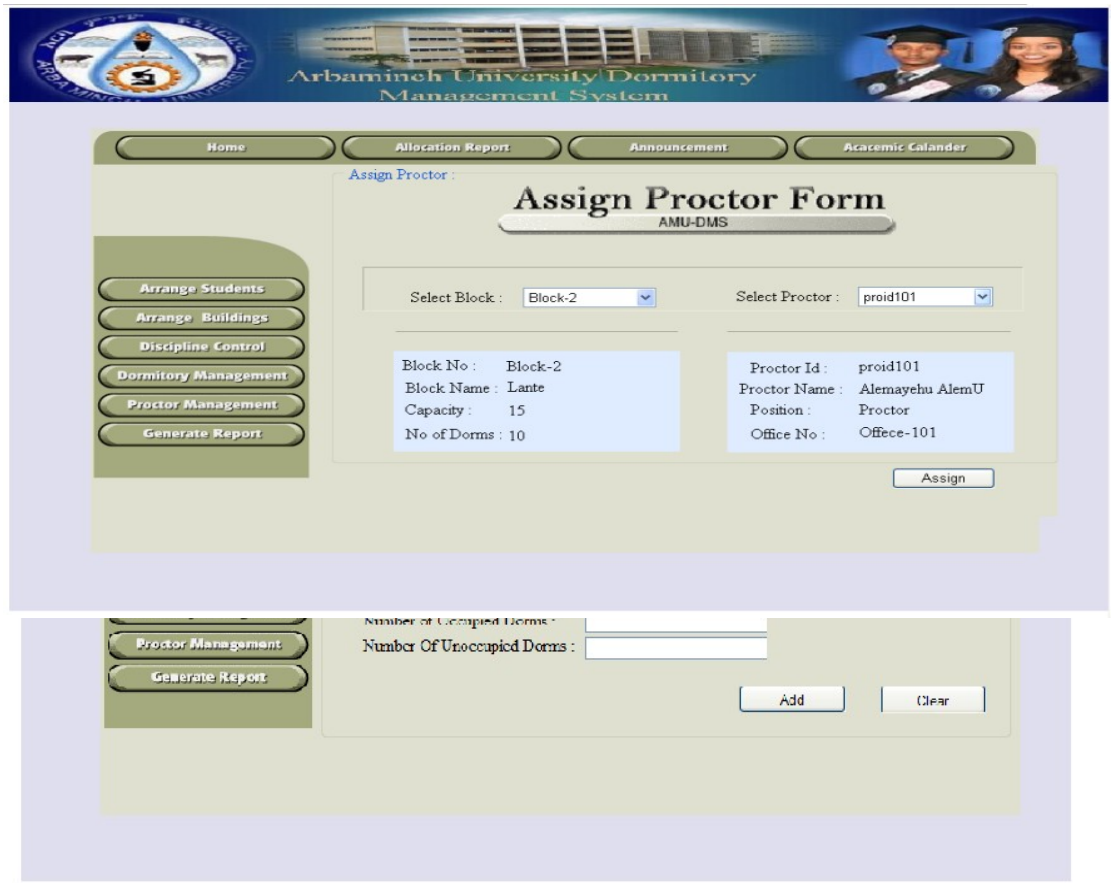


V. **Student Registration Form: -** As a proctor gets data form concerned body, he will register this student and also he will recall or use it while allocation.

VI. **Add New proctor Form: -** This form is also a secured form only accessible by a proctor manager when a new proctor is hired. I.e. the data of a proctor will be entered and saved through this

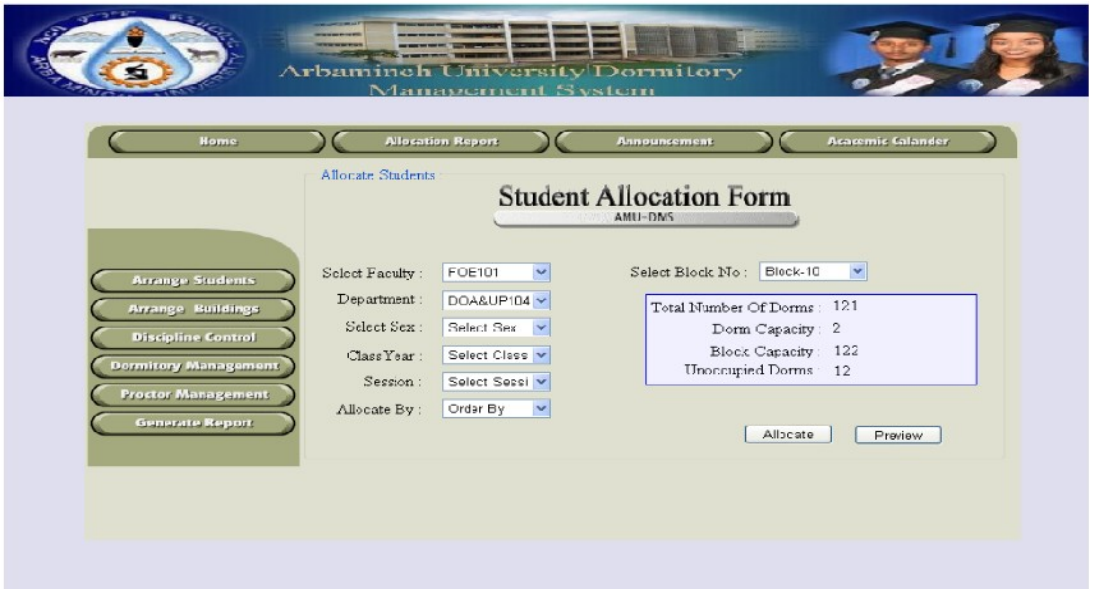


VII. **Assign Proctor Form: -** This form is used when a proctor manager wants to assign a proctor to a block which has not assigned yet. It is also used only by proctor manager.



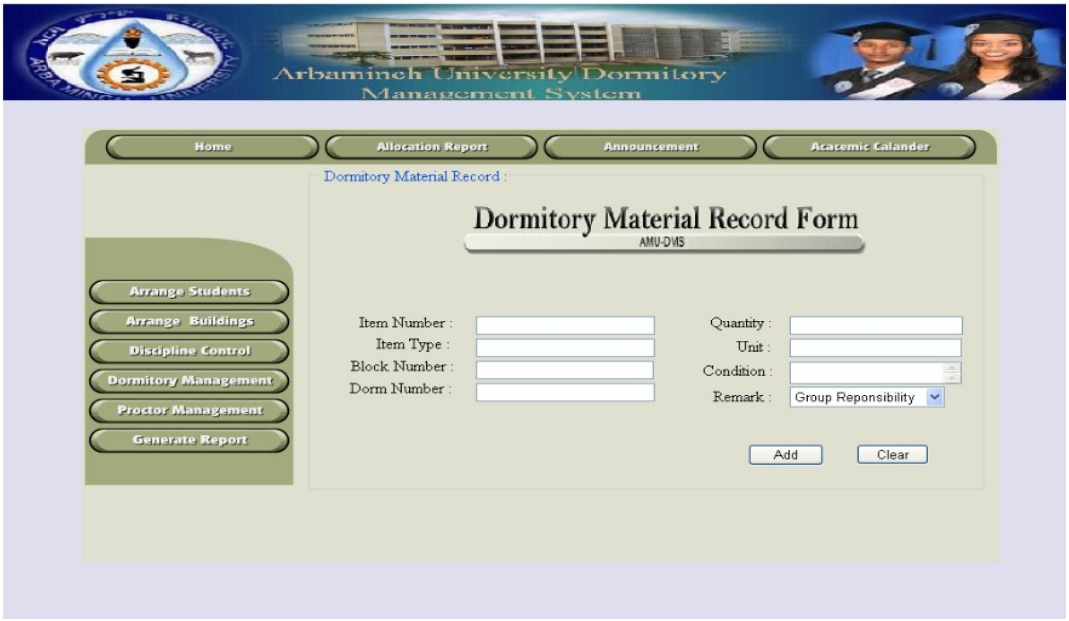
VIII. **Add New Building Form: -** When a new building is constructed or released from another service for a dormitory purpose, it will be recorded through this form. This form also contain number of dorm it contain, possible number of students per dorm (dorm capacity), and block capacity.

IX. **Student Allocation form:** This form bound to Database take dorm number, block number, faculty, and department from different tables and take input from users and perform



allocation accordingly.

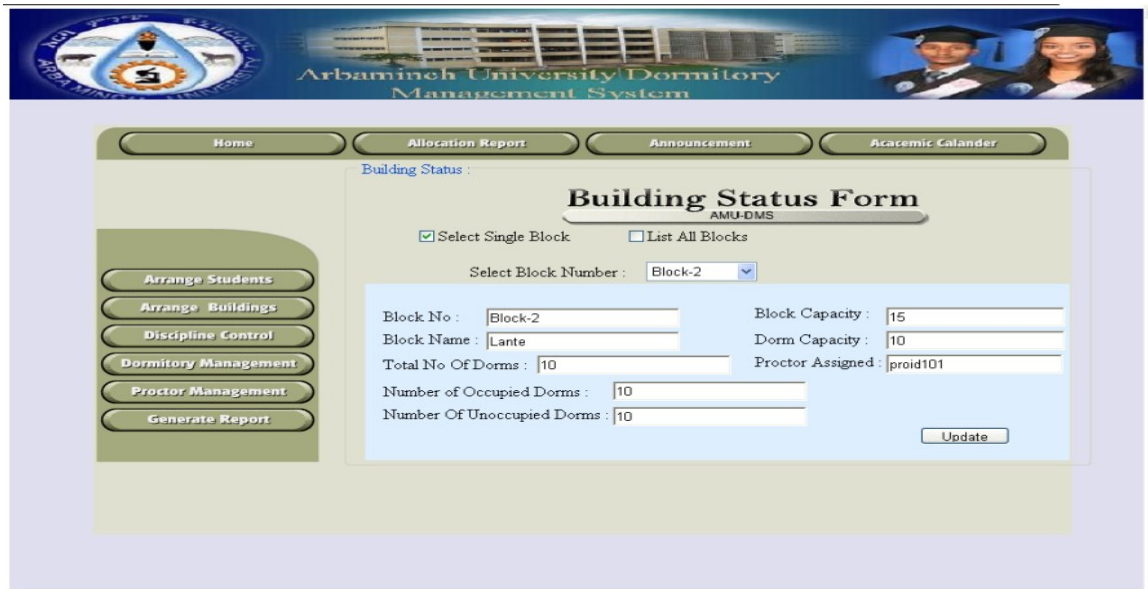
X. **Dormitory material Record Form: -** Materials given to a student together with a dorm need to be registered to simplify controlling of it. This is done through this form.



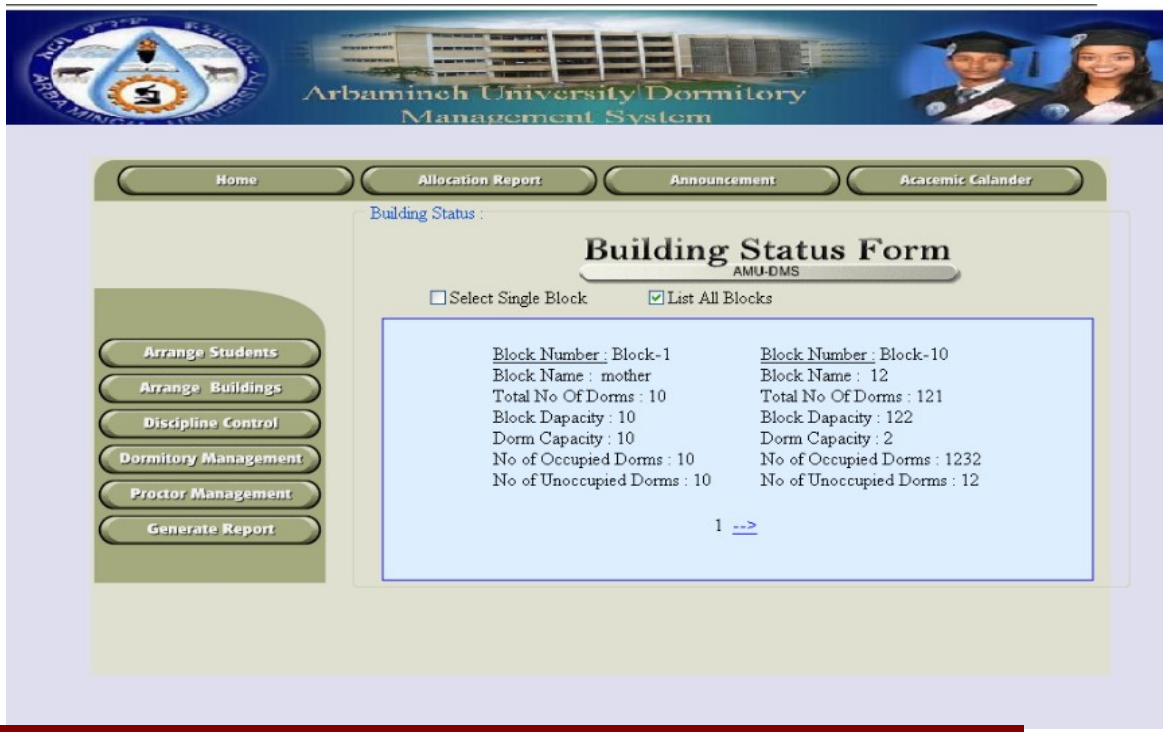
XI. **New Dorm Record Form:** In this form recording of new dorm when a new building is added will be recorded. It embraces all data related with new dorm.

XII. **Student Discipline Record Form:** - When a student violets rule, if it is a first time it will be recorded through this form. And appropriate measure according to the rule will be taken up on the student.

XIII. **Building Status Form (**when single block check box checked**)**



XIV. **Building Status** (when list all blocks check box checked)



All forms stated above are the main forms the team is working on. The remaining forms will be the reports that will be generated from all the above forms. Beside those forms, there may be other forms used as links, status viewer, reports etc. But the above are the main forms that user will directly contact (interact) with.

Chapter Five

Implementation

5.1 Introduction

Implementation refers to the Coding of the all documents gathered starting from requirement analysis to Design phase. So now the team is in a position of converting all documents gathered and designed into the code so that the system will be implemented for the user to be used for the purpose it developed. To implement it the user must have a server on which the system will be hosted.

5.2 Algorithm Design and testing

In this system the team has used a so called three layer (tier) architecture approach. These three layers are described as follows:

a) **Business layer:** This layer has **two** classes**, Entity or property** class and **Business** class.

Entity Classes: The categories of this class specify the data members of the objects and the properties to get

Entity

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace AMUDMS.BLL
{
    public class clsStudent
    {
        public string  StudentId { get; set; }
        public string  FirstName { get; set; }
        public string  MiddleName { get; set; }
        public string  LastName { get; set; }
        public int Age { get; set; }
        public string  Sex { get; set; }
        public string  FacultyNumber { get; set; }
        public string  DepartmentNumber { get; set; }
        public string  BlockNumber { get; set; }
        public int DormNumber { get; set; }
        public string  ClassYear { get; set; }
        public string  Session { get; set; }

    }
    public class clsDecipline
    {

        public int Id { get; set; }
        public string  StudentId { get; set; }
        public string FirstName { get; set; }
        public string  LastName { get; set; }
        public string  Faculty { get; set; }
        public string  Department { get; set; }
        public DateTime  Date { get; set; }
        public int BlockNumber { get; set; }
        public int  DormNumber { get; set; }
        public string  Remark { get; set; }
    }

    public class clsAllocationProcessor
    {
        public string  allocationFacultyNo { get; set; }
        public string  allocationDepartmentNo { get; set; }
        public string  allocationAllocationSex { get; set; }
        public string  allocationClassYear { get; set; }
        public string  allocationSession { get; set; }
        public string  allocationBlockNo { get; set; }
        public string  allocationOrderedBy { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using AMUDMS.DAL;

namespace AMUDMS.BLL
{
    public class cbsStudent
    {

        public static void RegisterStudent(clsStudent student)
        {
            string sqlStatement = "INSERT INTO Student(StudentId,
            FirstName, MiddleName, LastName, Age, Sex, FacultyNumber,
            DepartmentNumber, BlockNumber, DormNumber, ClassYear,
            Session)VALUES('"+ student.StudentId
            +"','"+student.FirstName +"','"+student.MiddleName
            +"','"+student.LastName
            +"','"+student.Age+"','"+student.Sex"','"
            +student.FacultyNumber +
             "','"+student.DepartmentNumber +"','"+student.BlockNumber
            +"','"+student.DormNumber +"','"+student.ClassYear
            +"','"+student.Session +"')";
            clsDAL dal = clsDAL.GetDAL();
            dal.ExcuteNonQuery(sqlStatement);

        }

        public static bool ValidateStudent(string studentId)
        {
            bool studentExist = false;
            clsDAL dal = clsDAL.GetDAL();
            string sqlStatement = "SELECT StudentId FROM Student WHERE
            StudentId='" + studentId + "'";
            DataTable dt = new DataTable();
            dt = dal.ExcuteQuery(sqlStatement);
            if (dt.Rows.Count > 0)
            {
                foreach (DataRow dr in dt.Rows)
                {
                    if (dr["StudentId"].ToString() == studentId)
                    {
                        studentExist = true;
                    }
                }
            }
            return studentExist;
        }
    }
      //Continued…
```

B

```
public static clsStudent GetSingleStudent(clsStudent student)
        {
            clsDAL dal = clsDAL.GetDAL();
            DataTable dt = new DataTable();
            if (student.StudentId != "")
            {
                  string sqlStatement = "SELECT * FROM Student WHERE
                  StudentId='" + student.StudentId  + "' ORDER BY
                  FirstName";
                  dt = dal.ExcuteQuery(sqlStatement);
            }
            else
            {
                  string sqlStatement = "SELECT * FROM Student WHERE
                  FirstName='" + student.FirstName + "' AND
                  MiddleName='" + student.MiddleName + "'AND
                  LastName='" + student.LastName + "'";
                  dt = dal.ExcuteQuery(sqlStatement);
            }

            if (dt.Rows.Count > 0)
            {
               foreach (DataRow dr in dt.Rows)
                {
                    student.StudentId= dr["StudentId"].ToString();
                    student.FirstName = dr["FirstName"].ToString() ;
                    student.MiddleName = dr["MiddleName"].ToString() ;
                    student.LastName = dr["LastName"].ToString();
                    student.Age = Convert.ToInt32(dr["Age"]) ;
                    student.Sex = dr["Sex"].ToString();
                    student.BlockNumber =
                    dr["BlockNumber"].ToString();
                    student.DormNumber =
                    Convert.ToInt32(dr["DormNumber"]);
                    student.ClassYear = dr["ClassYear"].ToString();
                    student.Session = dr["Session"].ToString();
                }
            }
            return student;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data;
using System.Configuration;

namespace AMUDMS.DAL
{
      public class clsDAL
      {
            private static clsDAL newInstance = null;
            private static SqlConnection newConnection = null;
            public clsDAL(){
                  newConnection = new SqlConnection();
                  newConnection.ConnectionString =
                  ConnectionManager.GetConnectionString;
            }
            public static clsDAL GetDAL(){
                  if (newInstance == null)
                newInstance = new clsDAL();
            return newInstance;
            }
        public DataTable ExcuteQuery(string sqlstmt){
                  DataSet ds = new DataSet();
            try{  newConnection.Open();
                  SqlCommand com = new SqlCommand(sqlstmt,
                  newConnection);
                  com.CommandType = CommandType.Text;
                  SqlDataAdapter da = new SqlDataAdapter(com);
                  da.Fill(ds);
                  newConnection.Close();}
            catch { }
            return ds.Tables[0];
         }

      public void ExcuteNonQuery(string sqlstmt){
            try{  newConnection.Open();
                  SqlCommand com = new SqlCommand(sqlstmt,
                  newConnection);
                  com.CommandType = CommandType.Text;
                  com.ExecuteNonQuery();
                  newConnection.Close();
                  }
      class ConnectionManager{
        public static string GetConnectionString{
            get{
                  return ConfigurationManager.ConnectionStrings
                  ["AMUDMSConnectionString"].ConnectionString ;
            }
        }
    }
```

access pr

```
using System;
using AMUDMS.BLL;

namespace AMU_Dormitory_Management_System
{
    public partial class WebForm4 : System.Web.UI.Page
    {
        protected void btnAdd_Click(object sender, EventArgs e)
        {
            clsStudent student = new clsStudent();
            student.StudentId = txtStudentId.Text;
            bool studentAlreadyExists=
            cbsStudent.ValidateStudent(student.StudentId);
            if (studentAlreadyExists)
            {
                lblMessage.Text = "***  Student Already Exist!";
            }
            else
            {
                student.FirstName = txtfirstname.Text;
                student.MiddleName = txtmiddlename.Text;
                student.LastName = txtlastname.Text;
                student.Age = Convert.ToInt32 (txtage.Text) ;
                student.Sex = ddlSex.SelectedValue.ToString();
                  student.FacultyNumber =
                  ddlFacultyNo.SelectedValue.ToString();
                  student.DepartmentNumber =
                  ddlDepartNo.SelectedValue.ToString();
                  student.BlockNumber =
                  ddlBlockNo.SelectedValue.ToString();
                  student.DormNumber = Convert.ToInt32(txtdormno.Text );
                  student.ClassYear =
                  ddlClassYear.SelectedValue.ToString();
                  student.Session = ddlSession.SelectedValue.ToString();
                  cbsStudent.RegisterStudent(student);
                  lblMessage.Text = "***  Student Registered
                  Successfully!";
            }


        }

}
```

```
public partial class WebForm2 : System.Web.UI.Page
    {

      protected void ddlBlockNo_SelectedIndexChanged(object sender,
      EventArgs e)
        {string blockNo = ddlBlockNo.SelectedValue.ToString();
            clsBuilding buiding = cbsBuilding.getBuildingStatus
(blockNo);
            txtBuidNO.Text = buiding.blockNo.ToString();
            txtBlockName.Text = buiding.blockName.ToString();
            txtTotalNoOfDorms.Text = buiding.totalNoOfDorms.ToString();
            txtBlockCapacity.Text = buiding.blockCapacity.ToString();
            txtDormCapacity.Text = buiding.dormCapacity.ToString();
            txtNoOfOccupiedDorms.Text =
            buiding.noOfOccupiedDorms.ToString();
            txtNoUnOccupiedDorms.Text =
            buiding.noOfUnnoccupiedDorms.ToString();
            txtProctorAssigned.Text =
buiding.proctorAssigned.ToString();
            pnlListSingleBlock.Visible = true;
            btnUpdate.Visible = true ;
        }
      protected void chkListAllBocks_CheckedChanged(object sender,
      EventArgs e)
        {
            if (chkListAllBocks.Checked == true)
            {
            lblSelectBlockNo.Visible = false;
                ddlBlockNo.Visible = false;
                pnlListSingleBlock.Visible = false;
                chkSelectSingleBlock.Checked = false;
                pnlListAllBlocks.Visible = true;
                btnUpdate.Visible = false ;
                  }
            else
            {
                pnlListAllBlocks.Visible = false;
                btnUpdate.Visible = false;
            }
        }
```

5.2 Test procedures

To the test whole system the team follow the following procedures.

Unit testing: -Every module of the System is separately tested. I.e. the team tests every module by applying some selection mechanism. Through this mechanism every modules gets tested. If an error occurs correction will be taken with out affecting another module.

Integrated testing: - In this testing part, all the modules will be combined together and tested it for its fitness with each other and with the systems functionality. If error occurs in combining them, the module with problem will be identified and re combined.

Both **units testing** and **integrated testing** are performed by all team members at the work place. Here there I no involvement of user.

System testing: - In this testing, the team performs over all functional testing by checking whether it meets the required target.

User Acceptance testing: - Under this testing there are another sub testing which would be performed: **alpha testing and Beta testing**.

- **Alpha testing: -** representative of the user will come to us and test the system by him self whether it meets their need or not.
- **Beta testing: -** The system will be tested by the users at their own working place whether it meets their needs or not.

5.3 Hardware and Software acquisitions

For the project implementation; the following Software and hardware are used.

Hardware

- Printer: For printing Documentation
- Server: for connection to the client computer(to host the system)
- Computers
- Network connection

Software

For the System implementation the following software's are used.

- **Microsoft SQL server 2005**
- **Microsoft Windows Last XP professional edition**
- **Microsoft Windows Server 2003**
- **Microsoft visual Studio 2008**
- **Internet Information Services 6 (IIS 6)**

5.4 User manual preparation

Since the system is web based every thing important for the user will be explained and implemented while giving short training when the system is deployed. Ruther there is no need of preparing full user manual because it is only deployed (hosted) on a single machine that is server.

5.5 Training

During the deployment of the system, the project group members will give short time training for the system administrators explaining how the system works and in what way they can manage their system.

5.6 Installation

Since the project is a web based System, there is no need to install it on a particular machine rather it will be hosted on a server.

5.7 Start-up Strategy

Once the system is hosted, it has two parts: One which needs password and username that is for proctor and proctor manager. To access those parts one has to have password and user name so that he/she can enter into it and use it. This

accessibility has also two parts, one which is restricted for proctor managers and the other for proctor and both.

The other part is those which do not need pass word and username so they can be viewed by any body.

Chapter six

Conclusions and Recommendations

6.1 Conclusions

Arbaminch University Dormitory management System is one of the main Management System found in the Universities Management. This system is a web based application to serve students as well as the working group of the system in different direction. Specially:-1) Students now made possible to know their dorm on line which overcomes extra expenditure of students time and resource, 2) saving proctors time lost for assigning dorm for students, preparing report while student leave from campus, etc

Through various challenging, now the team is coming to the end of this project. Those different challenges made possible by the cooperation of all the group members. In developing this project all group members contributed their full capability with maximum interest and all group members get ways toward developing a project.

6.2 Recommendations

While doing this system the team has faced different challenges. But by the cooperation of all the group members and an advisor the team is now able to reach to the final result. I.e. all the group members strongly fought these challenge and take the turn to the front.

So now all the group members strongly recommend the department that for the coming students, it has to provide them with better service than the present in better hard ware, guaranteed software’s, giving orientations how to proceed, offering guest to provide them with more experienced work,

support morally, manually, forming good relation with students, giving students description of each phases and so on. So that it will get what it expects from its students and satisfy with them.

Appendix

References

To do this system starting from the requirement analysis to the implementation the team used the following materials

Books

- Essentials of System analysis and design(in analysis and design phase)
- System analysis and Design methods(in analysis and design phase)
- Microsoft C# Programming for the Absolute Beginner(in implementation phase)
- AS P.NET Web Developer's Guide(in implementation phase)
- ASP.NET video tutorial with c#(in implementation phase)

Websites

- **www.Google.com**
- **www.codeproject.com**
- **www.learnvisualstudio.net**
- **Other occasional sites.**

Glossary

UML: Unified Modeling language

AMU-DMS: Arbaminch University Dormitory management System

ASP: Active server page

SQL –Structural Query Language

TVM: Time Value of Money

OOA: Object Oriented Analysis

OOD: Object Oriented Design