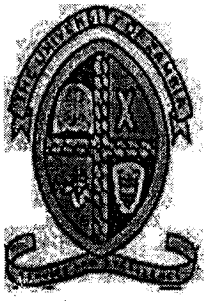
THE UNIVERSITY OF ZAMBIA

SCHOOL OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCES



PROJECT : HOSTEL SPACE(ROOM) ALLOCATION

NAME : OSBECK MANDA

COMPUTER NUMBER : 10052283

TITLE : FINAL YEAR PROJECT

SUPERVISOR : MR DAVID.M.ZULU

COURSE CODE : CSC 4004

DECLARATION

I, The undersigned below declare that the work done is my own and that it according to my knowledge it has not been submitted before for any university degree. All the information in this document they have correct references.

Author:	OSBECK MANDA
Computer Number:	10052283
Date:	.....
Signature:	.....
Supervisor:	.....
Signature	.....

ACKNOWLEDGEMENTS

I would like to thank my project supervisor Mr David Makadani Zulu for being there for me up to the completion of the project. I would also like to thank Dr J Phiri for proposing this project. I would like to further thank the all lecturers at Computer Studies Department for being there for us for the past four years.

ABSTRACT

The Hostel space allocation system is a system developed for use by the Dean of Students. It was developed to be a prototype and was analyzing few major constraints characterizing Students. The system considers males independent from females; allocates bed spaces to students using a number of criteria like students health, the year they are in, the date they applied if they are in first year. The handicapped are given a higher priority than the physically fit. Students in the final year are given a higher priority over the other years because they have to concentrate on their projects. Those who applied early are given a higher priority over those who applied late. The other constraint is school performance, students are grouped into three categories. Those who perform highly or students recommended or those students who have B+'s and above in all the courses. Those who perform good but lower than the commended and those who perform low. These constraints are all added up for each student and the final sum of their weights are ranked from the highest to the lowest. The student having the greatest score gets the room first and so on. The handicapped are allocated on the floor level because of their peculiar health. All these Categories are summed up into:

- Health
- Year of a student
- School Performance
- And Time of Application for first years

TABLE OF CONTENTS

TITLE PAGE

CHAPTER 1 : INTRODUCTION TO RESEARCH..... 6 - 10

1.1 Introduction.....7
1.2 Basic terms.....7
1.3 Problem statement.....8
1.4 Objectives.....8
1.5 Scope.....9
1.6 Expected results.....9
1.7 Requirements.....9

CHAPTER 2 : LITERATURE REVIEW.....11-29

2.1 Introduction to literature review.....11
2.2 Similar works.....12
2.2.1 Abbreviations.....14
2.2.2 CaH1 heuristic.....15
3.2.3 CaH2 heuristic.....16
2.2.4 HaNH heuristic.....19
2.2.5 Genetic Algorithm (GA) (HAGA, FAGA) 20
2.2.6 Chromosome Representation.....22
2.2.7 Fitness Evaluation.....23
2.2.8 GA Operations.....25

CHAPTER 3 : ANALYSIS.....30-38

3.1 Usecases.....30
3.2 Usecase descriptions.....31
3.2.1 Enter student usecase description.....31
3.2.2 Add user usecase description.....32
3.2.3 View user usecase description.....32
3.2.4 Enter free room usecase description.....33
3.2.5 View free room usecase description.....33
3.2.6 View female scores usecase description.....34
3.2.7 View male scores usecase description.....34
3.2.8 View Allocated female scores usecase description.....35
3.2.9 View allocated male scores usecase description.....35
3.2.10 View all students' scores usecase description.....36
3.2.11 View female scores usecase description.....36

3.3 Objects in the domain.....37
3.4 Domain analysis.....38

CHAPTER 4 : DESIGN.....39-40

4.1 Detailed class diagrams.....39
4.2 Database design.....40

CHAPTER 5 : MPLEMENTATION.....41-47

5.1 Brief System description.....41
5.2 System screenshots - Add user page.....44
5.2.2 View empty rooms.....44
5.2.3 Generate female scores.....45
5.2.4 Generate male scores.....45
5.2.5 Allocate Females.....46
5.2.6 Allocate males.....46
5.2.7 Enter student form.....47

CHAPTER 6 : TESTING.....48

6.1 Testing the operations of the system.....48
6.1.1 The students table(on the following page).....48
6.1.2 All the students.....50
6.1.3 System showing rankings of male students' scores.....50
6.1.4 System showing rankings of male students' scores.....51
6.1.5 After entering the empty rooms.....51
6.1.6 After accommodating males.....52
6.1.7 Allocated females.....52
6.1.8 Unallocated.....53
6.1.9 Error messages.....53

CHAPTER 7 : DISCUSSION AND CONCLUSION.....54

REFERENCES.....55

CHAPTER 1 : INTRODUCTION TO RESEARCH

1.1 Introduction

Systems in space allocation have been there from long ago, but the case of hostels space shortage is a problem of developing countries. Therefore , not much have been done in hostel Space allocation. However certain countries in Africa are taking the same step

The Online Space allocation System is a system designed to solve common problems face by School, College and University management in Africa and most developing countries. Since accommodation is scarce and difficult to acquire there came a need to develop a system which was to solve the problem of giving rooms to deservingly students. Corruption and any form of other ways of acquiring rooms was to be eradicated. To smoothly allocate rooms the System uses a number of constraints in assigning hostel space to students like where the student comes from, that is at the country level, Health, Year of a student, time of application, participation in sports, also where the student comes from at district level and academic performance. What the system does is assign priorities to these constraints and come up with a composite or cumulative sum for each student.

1.2 Basic terminology

We provide some basic definitions and notations related to the HSAP.

1. Bed Space: A space created in the hostel to accommodate a single student.
2. Hall (Hostel) (H): A building meant to accommodate a number of students located at various zones within the campus. Each hall has a pre-defined capacity in terms of bed space and consists of set of rooms located at various floors levels and blocks. A general assumption is that halls are located in different parts of the campus vicinity.
3. Capacity: the maximum number of students that can be accommodated in a given entity (e.g. hall, floor, room). This is calculated as the total number of bed spaces in that entity.

4. Category (C): A grouping into which eligible applicants are divided for hostel allocation purpose. Each category has varying number of students with varying degree of allocation priority.

5. Allocation (A): The assignment of students in a given category into an allocation entity (hall, block, floor). For example at the hall level, a_{ij} refers to the number of students in category i allocated to hall j. At the Floor/block level, it refers to the number of students allocated to the floor/block.

1.3 Problem Statement

Currently management allocates hostel space manually and the manual system has brought a lot of inconsistencies in bed space allocation. Students have to use the ordinary and old way of communication, that is sending their application for accommodation using datagrams. This method is inefficient and ineffective. Letters stay for months without delivery and the risk of losing them on the way as they are being transferred. These letters can be lost due to accidents, theft, car catching fire and many other. Therefore, there came a need to develop a system to handle these problems. The hostel space allocation system however has to subsystems : For applying online which is a system to handle online application and the system developed to choose qualified students

1.4 Objectives

To find a way of intelligently allocating students in an orderly manner

Take advantage of different constraints to guide the allocation

Reduce time and difficulties encountered by management to manually allocate hostel space

Employ a strategy to assign different priorities to different categories of students like:

- Final year student
- Peculiar health students
- Foreigners

- First years
- And time of application

1.5 Scope

The Scope of the system is to provide a way for the administrator to enter different names for different students for now since the online application system in place. The System is a standalone developed in java and is to be used from an offline point of view. The system shall also allocate students to free spaces also entered by the administrator. The system is to allocate females independent from males. For now the system considers undergraduate and postgraduates at once

1.6 Expected results

The system is expected to successfully assign female students and male students. The system is expected to rank the students according to their sum of priorities. Constraints(explained in the next section) being the form of metric for this project are expected to satisfy management and students in general. Depending on the number of empty rooms the system shall select students starting with the one having the highest priority followed by the one immediately before the highest and so on. If empty rooms ar used up the that is it, the remaining ones are left out.

1.7 Requirements and constraints

The system considers males independent from females, the system also assumes and considers postgraduates and undergraduate as one. The system allocates bed spaces to students using a number of criteria like students health, the year they are in, the date they applied if they are in first year. The handicapped are given a higher priority than the normal ones. Students in the final year are given a higher priority over the other years because they have to concentrate on their projects. Those who applied early are given a higher priority over those who applied late. The other constraint is school performance, students are grouped into three categories. Those who perform highly or students recommended or those students who have B+'s and above in all the courses. Those who perform good but lower than the commended and those who perform

low. These constraints are all added up for each student and the final sum of their weights are ranked from the highest to the lowest. The student having the greatest score gets the room first and so on. The handicapped are allocated on the floor level because of their peculiar health.

CHAPTER 2: LITERATURE REVIEW AND RELATED WORK

2.1 Introduction

Space allocation problem (SAP) is a combinatorial optimisation problem (COP) with some similarities to the classical knapsack problems. It is also related to some scheduling problems, for example academic timetabling. SAP and the associated resource efficiency issues impact on different institutions ranging from companies, government to educational institutions. It finds relevant application in various areas such as disk storage space allocation and transshipment storage space allocation. The current study focuses on
SAP as it is applied to academic institutions. The distribution of available room space among various entities (staff, students, lectures, laboratories, examination venues, etc.) in academic institutions is a dynamic process that is carried out on regular basis. The scarcity of space makes it necessary to think of evolving an efficient allocation strategy for the limited available one. This kind of efficiency can be said to be achieved when all demanding entities are given the minimum required space possible while observing pre-stated constraints to a high degree of satisfaction. Burke et al. thus advocated the use of appropriate optimization strategy incorporated in an automated system which is believed to save a lot of time and effort in space administration. As with other COPs, exact and heuristic methods can be used to provide solutions to SAP. Exact methods seek to solve problems to guaranteed optimality but execution on large real-world problems usually requires huge computational time.

SAPs are domain-specific decision problem which have attracted attention among researchers working on metaheuristics within the past few years. SAP seeks to reach the best objective under some operational domain constraints. In the context of higher institution, it is defined as the allocation of entities (staff, students, meeting rooms, lectures, special rooms, etc.) to areas of room so as to satisfy as many requirements and constraints as possibleThe problem is highly constrained, has multiple objectives, varies greatly among different institutions and requires frequent modifications due to the addition or removal of entities and/or rooms. It also has direct impacts on the functionality of institutions. Unfortunately, like other difficult tasks in higher institutions, especially in developing countries, space allocation is carried out manually which in

most cases take time to complete. Considering domain specific cases of SAP, the best practice in many institutions in developing countries is reliance on a form of database or spreadsheet that keep record of entities and available spaces without any recourse to any form of algorithm to determine optimal space allocation.

Several domain-specific cases of SAP have been studied in literature including office space allocation, shelf space allocation, and storage spaces allocation for transshipment tasks. Annevelink and Broekmeulen studied the application of GA into space allocation planning in pot-plant nurseries within the field of horticulture. This was a pioneer work that seeks the use and incorporation of GA metaheuristic into decision support system for the complex-natured space allocation planning in the nurseries. Similarly, Dean investigated the use of GA to optimize the staff scheduling within the domain of nurse rostering. His work entails a case of assigning employees to time slots in a way that satisfies given constraints. Similar works on hostel space allocation have been done in Nigeria and South Africa.

2.2 Similar Works

For hostel allocation purpose, students are grouped into eight different categories with different allocation priority. In the order of priority, we have: 1) Final year students (Fy), 2) Scholars - those with cumulative grade point average (CGPA) = 4.20 of 5.00 scaling with those doing master's degree inclusive (Sc), 3) Foreign students (Fo), 4).Health students - physically challenged students or those with peculiar health conditions (Ht), 5) Fresher - first year and direct entry students (Fr), 6) Sports men and women (Sp), 7) Discretionary - based on individual special request (Ds), and 8) Others - students of other levels or years (Ot). Priority is assigned based on certain academic and/or administrative considerations by the institution authority. For example, the need for Fy to be accommodated in order to have full concentration on their studies and project work; and the allocation of all Fo category as they might not have a relative within the country. An optimal category allocation therefore should have a descending allocation curve in the given order of priority.

Level	Constraints		Classification
Category Allocation	a.	All students in Fo, Ht and Sp categories must be allocated	Hard
	b.	As many Fy, Sc, Fr, Ds and Ot as possible should be allocated in this order of priority.	Soft
Hall Allocation	a.	Students in Ht, Sc and Sp must be allocated to designated hostels	Hard
	b.	Allocation for the remaining categories must be allocated in the stated order of priority	Soft
Block/Flow Allocation	a.	Ht category should be allocated to the lowest floor possible in their assigned halls	Soft
	b.	Fy category should be allocated to the highest possible floor	Soft

Table 4.1: Summary of Constraints/Requirements

CATEGORY	SPECIFIED HALLS	
	MALE	FEMALE
Ht	HA1	HA3
Sc	HA2	HA3
Sp	HC1	HC3

Table 4.2: Specified example Halls for certain Categories

HA1, HA2, HC1,HA3, HA4, HC3 are the example 6 Halls given

Given the multi-level nature of the allocation process and the set of constraints in Table 4.1, we propose a hierarchical heuristics to handle the problems (Fig 4.1). Different heuristics handles the allocation at each stage but the result of a succeeding stage depends on that of the previous stage(s). Two distinct algorithms are designed for each of the first two stages. The final distribution into floors is then handled by Genetic algorithm (GA). The main aim is to find an optimal allocation and distribution of students into hostels. This implies optimizing bed space utilization while satisfying specified constraints and requirements. Two classes of allocation

choices are identified at some stages of the allocation process, namely the *fixed-choice* and *free-choice* allocations. At the category level, the fixed-choice allocations of all students in Ht, Fo and Sp categories must be achieved first before the free-choice allocation of the remaining categories according to assigned order of priority. At the hall level, allocation of students in Ht, Sp and Sc is considered fixed-choice while the remaining categories allocation is free-choice.

2.2.1 Abbreviations

CaH1=Category allocation Heuristic (algorithm) 1,
CaH2=Category allocation Heuristic (algorithm) 2
HaGA=Hall allocation Genetic Algorithm

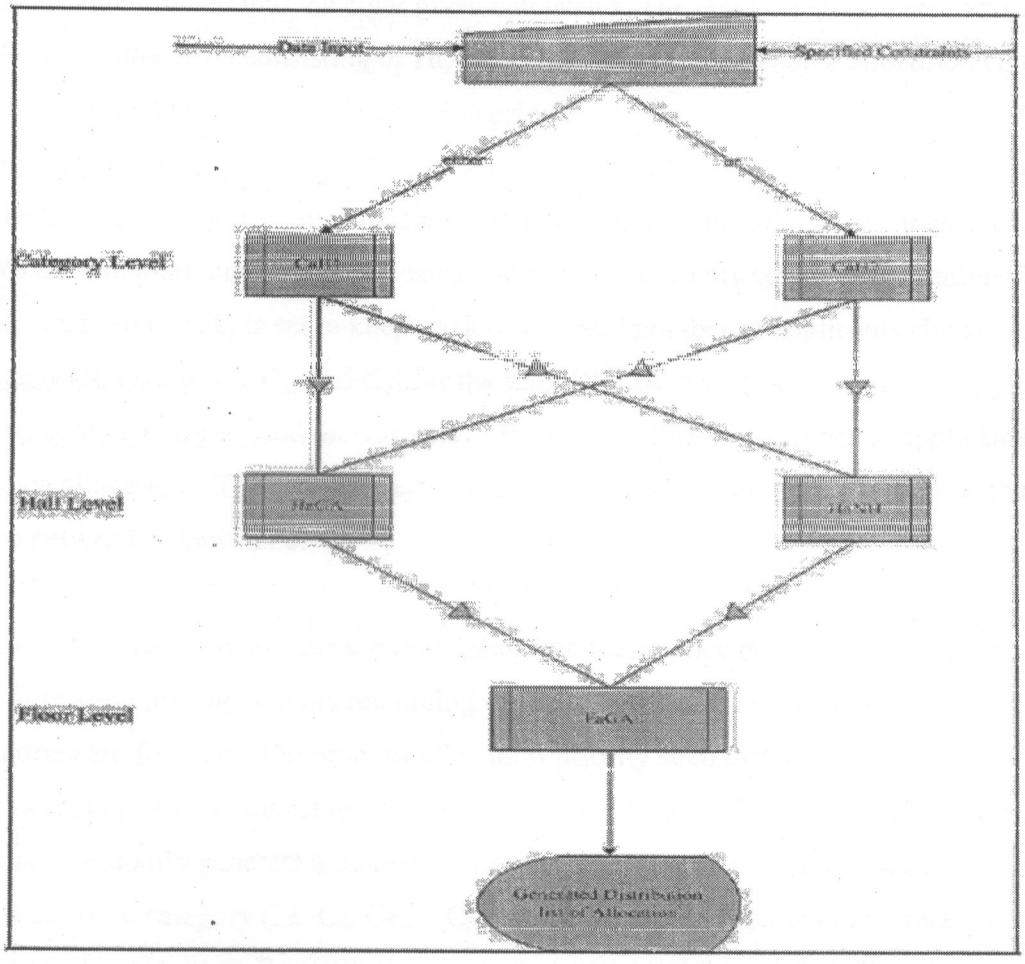


Fig 4.1: The structure of the hierarchical heuristics

With respect to Fig 4.1, necessary input such as the capacities for halls, number of floor for each hall, and number of applicants for each category are supplied to the system together with necessary specified constraints. The category level is implemented via CaH1 or CaH2 heuristics. Results obtained can then be processed at the hall level via either HaGA or HaNH.
Finally, the floor allocation is driven by GA via FaGA to give the distribution of students into various blocks and floors within each hall of residence. The HaGA and FaGA presented in Fig 4.1 are both GA metaheuristics with similar structures used at the hall and floor allocations respectively.

2.2.2 CAH1 heuristic

The applicants are divided into the eight given categories. The eight categories are grouped into the *fixed-choice allocation* consisting of Ht, Fo and Sp and the *free-choice allocation* consisting of Fy, Sc, Fr, Ds and Ot, in order of allocation priority.
CaH1 employs a greedy-like approach that first ensures that the fixed-choice categories are fully allocated while observing the hall capacity constraint. Initially, the allocation for each category is set to zero and the total number of applicants and the total capacity for all halls combined, TH, are noted. A variable, TFc, is set to keep track of the total number of applicants allocated so far. The category allocation (C_1, C_2, and C_3) for the fixed-choice is set to the number of applicants for each respective category and the variable, TFc, is set to the total number of applicants for all fixed-choice allocations. The total capacity of space remaining in all the halls, rem, is thus the difference between TH and TFc.

The heuristic then seeks to allocate the remaining five free-choice categories while observing the assigned allocation priority and the remaining capacity constraint for the halls.
The categories are first sorted in order of allocation priority such that $C_4 > C_5 > \ldots > C_8$. A Boolean variable is used to detect and prevent negative allocation into any of the category.
The heuristic randomly generate a number, r, between 0 and remaining hall capacity (rem). And set each successive category (i.e. C_4, C_5,..., C_8), as the minimum between r and total number of applicants for that category. For analysis purpose, we compute the number of unallocated

applicants for each category as the difference between the number of applicants and the number allocated for the category.

The pseudo-code for CaH1 is given below:

---

i. Initialize: *Set the total capacity of all Halls to TH, Ci = 0 for all categories,*

Appls[i] = Total number of applicants for category i

ii. Allocate Fixed Choice: *Set Ci = Appls[i], i = 1,2,3*

Sum up the students in Fo, Ht and Sp given TFc and subtract from TII.

iii. Allocate Free Choice:

Initial: *Set rem = TH - TFc and Bool Ok... FALSE;*

Prioritize: *Set free choice categories Ci, i = 4,..,,8 in order of priority such that*

C4 > C5> ... > C8.

while *(NOT Ok)*

rem = TH – TFc

int remNew = rem;

Allocate: *Set Ci = Min{remNew, Appls[i]},*

Set remNew = remNew - Ci, i = 4,..,8 in order of priority

CheckOk()

End while

iv. Calculate Unallocated: *Unallocated[i] = Appls[i] - Ci, for i = 4 to 8*

v. CheckOk: *If Ci >= 0,for all i = 4,..,8 Set Ok = TRUE*

2.2.3 CAH2 heuristic

Similar to CaH1, applicants are divided into both fixed-choice and free-choice with the allocation of the fixed choice categories proceeding in the same manner. CaH2 then allocates the remaining five free-choice categories based on the assigned allocation priority and the remaining capacity constraint for the halls. However, instead of a greedy-like approach of

CaH1, CaH2 employs a percentage range allocation distribution strategy which is set based on the allocation priority. The goal is to ensure that there is at least some allocation for each category of the free-choice assigned to some hall of residence.

The free-choice allocation categories are first sorted in order of allocation priority such that $C_4 > C_5 > \ldots > C_8$. A percentage range in $[Minp_i, Maxp_i]$, i=4,..,8, is assigned to all the categories such that the maximum percentage for the highest priority category is less than 100 and the minimum percentage for a higher category is greater than or equal to the maximum percentage for the subsequent category. Having assigned the percentage range for all categories, an allocation range $[MinA_i, MaxA_i]$, i=4,..,8, is computed for each category. A randomly number, x_i in $[MinA_i, MaxA_i]$, i=4,..,8, is generated and summed up to give, sum, for all the free-choice categories. This number x_i determines the number of applicants to allocate for the i^{th} category. However, to prevent space overuse, we need to balance up this sum with the remaining hall capacity constraint, rem. This is done by allocating exactly only rem/sum fraction of xi for each category C_i, i=4,..,8. We assign

$$GO_i = \frac{rem}{sum} x_i$$

Where

$$sum = \sum_{4}^{8} x_i$$

Clearly,

$$\sum_{4}^{8} GO_i = rem$$

Note that even though xi is random, making it possible for sum to have different values for different execution of the algorithm, the balancing we did ensures that the total number of students allocated is exactly the total number of space left to be filled in all the halls. Thus, the

random nature of xi will not have any negative effect on the GA used at the floor allocation level.

Similar to CaH1, the number of unallocated applicants for each category computed as the difference between the number of applicants and the number allocated for the category.

The pseudo-code for the CaH2 is now given:

---

i. Initialize: *Set the total capacity of all Halls to TH, $C_i = 0$ for all categories, Appls[i] = Total number of applicants for category i*

ii. Allocate Fixed Choice: *Set C_i = Appls[i], i = 1,2,3*

Sum up the students in Fo, Ht and Sp given TFc and subtract from TH

iii. Allocate Free Choice:

Initial: Set variable sum = 0; Set rem = TH - TFc

Prioritize: *Set free choice categories C_i, i = 4... 8 in order of priority such that C4 > C5> ... > C8.*

Assign Percent Range: *For each C_i, assign a percentage range [Maxpi, Minpi] such that $Maxp_4 < 100\%$ and $Minp_i >= Maxp_{i+1}$.*

Calculate: *Calculate Allocation Range [$MaxA_i$, $MinA_i$] based on [$Maxp_i$, $Minp_i$]*

Select: *For each C_i, randomly generate a number xi in [$MaxA_i$, $MinA_i$] and add to sum*

Balance: *For each generated random xi, calculate new $\mathfrak{O}_i = x_i$ * rem/sum*

iv. Allocate: *For each C_i, allocate exactly the new xi, that is, $C_i = x_i$.*

v. Calculate Unallocated: *Unallocated[i] = Appls[i] – C_i, for i = 4 to 8*

2.2.4 HANH heuristic

At the hall level, the HaNH (see Fig 4.1) aims at optimizing the spread of allocated category of students into available hostels. It starts by first allocating thc fixed-choice categories. The heuristic then seeks to distribute the free-choice categories such that there will be maximal spread across all the hostels that are yet to be filled to capacity. The process is described as follows:

The category allocation obtained from either CaH1 or CaH2 is taken as input with the hall capacity constraints. The eight categories are divided into fixed-choice categories of Ht, Sp and Sc and the free-choice categories of Fy, Fo, Fr, Ds, and Ot. No order of priority is considered at this stage. The total capacity of thc halls is computed as TII. The first part of the heuristic allocates the fixed-choice categories into designated halls, say, Ha, Hb, and Hc and calculates the total allocations as TFc. This is donc by assigning number of students in each of these three categories to the respective designated hall. The capacity for each of Ha, Hb and Hc is thereafter decreased by the allocations for the category allocated to it rcspcctively. We then set the space left to be allocated in each hall as H_j, j=1...n, where n is the number of halls. The total space left to be filled is thus:

THNew = TH – TFc. S1njH

In order to ensure optimal spread in the distribution of the free-choice categories into available halls of residence, we compute a real-value hall ratio to drive the distribution. The hall ratio, HR_j, j=1,...,n, is computed as the ratio of the capacity of a hall to the total number space left to be allocated for all the halls (THNew), i.e. $HR_j = H_j/THNew$. Therefore, for each category in the free-choice and for each hall with a space left to be filled, the hall allocation for category i in hall j, a_{ij} is computed as the product of the hall ratio, HR_j, and the number of students in category i, C_i.

The pseudo-code for the HaNH is given as:

---

i. Input: *C_i, i=1..m from Category allocation, Hall capacities for all hostels*

ii. Initialize: *Set the total capacity of all Halls to TH.*

iii. Allocate Fixed Choice: *Sum up the students in Ht, Sc and Sp given TFc*

Assign CHt, CSc and CSp to designated Halls and compute space left for the halls

Let Hj, j=1..n be the space left to be fill for all hostels

Calculate total space left THNew = TH TFc

iv. Allocate Free Choice:

Hall Ratio: *Compute HR_j, j = 1..n as $HR_j = Hj/THNew$*

Reorder: *Set free choice categories Ci, i = 1...5 (in any order)*

Allocate: *For each category in the Free choice*

For each hall, Hj, j=1..n

Assign $a_{ij} = HR_j * C_i$

2.2.5 genetic algorithm (HAGA, FAGA)

GA [21] is a probabilistic search algorithm that iteratively transforms a population set of mathematical objects, each having an associated fitness value, into a new population of offspring objects based on the Darwinian principle of natural selection and use of operations that are patterned after naturally occurring genetic operations of crossover and mutation [22].
GA metaheuristic seeks to evolve towards an optimum (global best) solution to a given problem. Possible solutions (individuals) are set of values represented as chromosomes. Set of solutions (population) are generated from an initial population of solution and these evolve until terminating conditions are met, for example, the optimal solution is found or a pre-defined maximum number of iterations (generations) has been performed. Evolution of generations involves selection of individuals (otherwise known as parents) to breed based on a pre-defined fitness function and production of offspring through the applications of genetic operators of crossover and mutation. More description on the nature and application of GA can be found in

[21, 23, 24, 25]. We apply GA at both hall allocation (as an alternative heuristic) and floor level (as the final heuristic). Fig 4.5.1 gives an overview of the GA.

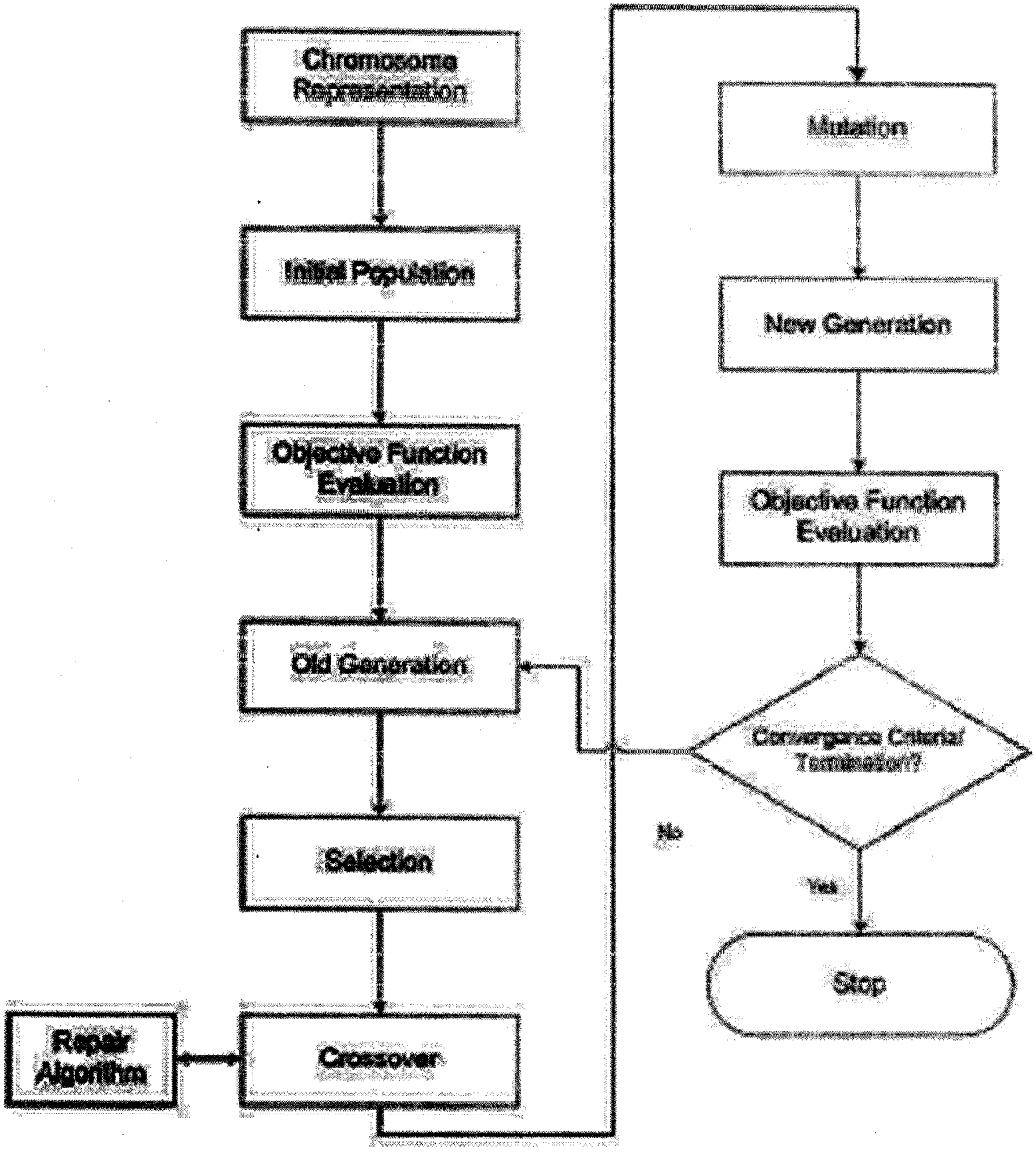


Fig 4.5.1: Graphical Illustration of the GA steps

2.2.6 Chromosome representation

We design similar but different chromosome representations based on a multi-dimensional array structure for both hall and block/floor allocations (Figs. 4.6.1&4.6.2). This type of representation has proved to be effective in handling similar problems with large data set and constraints [16]. At the hall level, the chromosome has the structure (c_1 (a_{11} ... a_{1n}), c_2 (a_{21} ... a_{2n}) ... c_m (a_{m1} ... a_{mn})), where c_i represents category i and a_{ij} represents the allocation of category i in hall j, m is the number of categories and n is the number of halls (Fig. 4.6.1). The population for the hall allocation thus consists of Ψ individuals, where Ψ is the population size. This is represented as a 3-dimensional array of integers for simulation purpose. An integer in the array with subscripts p, i, and j indicates the number of students in category i allocated to hall j in the solution represented by the individual p.

The chromosome at the block/floor level has the structure: (c_1 (b_{11} (a_{111} ... a_{11x}) ... b_{1w} (a_{1w1} ... a_{1wx})) ... c_m (b_{m1} (a_{m11} ... a_{m1y}) ... b_{mw} (a_{mw1} ... a_{mwy}))) where ci represents category i, b_{il} represents block l in category i's allocation, a_{ilk} represents category i's allocation on floor k of block l, m is the number of categories w is the number of blocks and x and y represent the varying number of floors in the respective blocks. Fig. 4.6.2 presents four sections (one for each block), each with 8 rows representing the eight categories and the blocks having 3, 1, 2 and 1 floors respectively. The population consists of Ψ individuals. For implementation purpose, the population is represented as a 4-dimensional array of integers with subscripts p, i, l and k representing the number of students in category i allocated to floor k in block l in the solution of individual p.

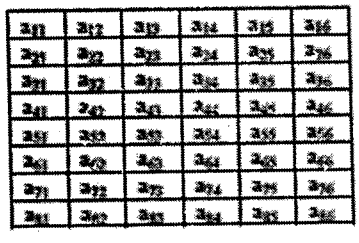


Fig 4.6.1: A chromosome representation for hall allocation with m=8 and n=6.

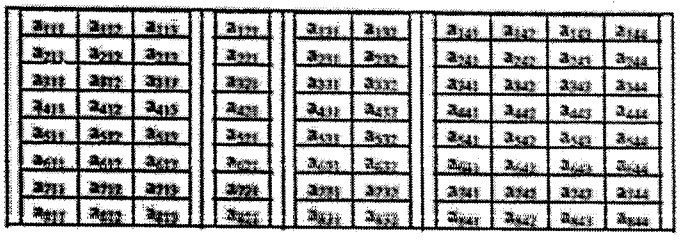


Fig 4.6.2: A chromosome representation for block and floor allocation with m=8, and w=4.

2.2.7 Fitness evaluation

Given the diversity in the criteria imposed on and by various categories in the study, it apparently become difficult to design an evaluation function that incorporates all the criteria with adequate weighting. However, the possibility of breaking down the constraints into the various levels makes the goal achievable. The fitness of an individual in a given population is computed as a measure of the degree of satisfaction of given constraints that influence the allocation. To this end, an assignment of weights accordingly to the various constraints at both the hall and block/floor levels. This guides the algorithm during the allocation process to ensure satisfaction of relevant constraints. The fitness evaluation function and the weights are combined to give a set of real number fitness values in [0, 1]. A fitness value of 0 indicates non-satisfaction of given all constraints for the allocation; while a value of 1 indicates total satisfaction of constraints. These fitness values are thus determined using the combination of the space utilization factor and assigned weight for respective constraints. Based on the constraints stated in Tables 4.1 and 4. 2, the fitness of an individual in a population at the hall level is computed as:

$$f=\sum_{q} w_q u_q \varepsilon[0,1]$$

where U is the utilization factor $= \dfrac{\text{Allocated Space (A)}}{\text{Capacity}}$

$$U_q = \frac{a_{ij}}{s_i},$$

w_q = weight assigned to constraint q, q 1, 2, 3 representing constraints presented in Table 4.2 for the three fixed-allocation categories, a_{ij} = number of allocations of category i to hall j, s_i = total number of students in category i where i,j are fixed (Table 4.2). The computation is repeated for all individuals in the population and the computed values used to determine the average fitness for the generation. We take note of the best fitness values over the generations. The three constraints at this level are assigned weights as follows: w1 – 0.4; w2 – 0.3; w3 – 0.3, where the subscript 1 represented Ht category, subscript 2 represents Sc and subscript 3 represents Sp.

Considering the two constraints at the block/floor level (Table 4.1), the fitness of an individual is computed thus:

$$f = \sum_q w_q u_q \varepsilon [0,1]$$

where u is the utilization factor at the floor allocation level given as:

$$u_q = \frac{T_i}{s_{ij}},$$

and

$$T_i = \sum_{k=1}^{p} a_{ik},$$

W_q is the weight assigned to constraint q, q = 1,2 representing floor level soft constraints (highest and lowest floor possible) as presented in Table 4.1, a_{ik} represents the allocation of category i to floor k, p = an integer representing the highest (or lowest) floor and s_{ij} = total number of students in category i allocated to hall j.

There are only two major constraints that affect allocation at this level, namely, for Ht (allocation to the lowest floors possible in hall) and Fy (allocation to the highest possible floors in the halls). We assign weight as follows: where the hall has an Ht allocation and no Fy allocation: w_1 – 1.0; w_2 – 0.0 with subscript 1 representing Ht category and subscript 2 representing the Fy category.

Where the hall has Fy allocation and no Ht (typically hall is not a specified hall for Ht): $w_1 - 0.0$; $w_2 - 1.0$; and where the hall has a both Ht and Fy allocations, $w_1 - 0.7$; $w_2 - 0.3$. Thus Ht students are given higher priority because of their peculiar health status and thus more stringent hard constraints associated with the category at all levels (Table 4.1).

2.2.8 Genetic algorithm (GA) operations

We apply GA in a similar but mutually exclusive fashion at the hall and floor levels. At both levels, a repair algorithm is incorporated to prevent space overuse. The pseudo-algorithm of the GA structure is depicted as follows:

---

i. Initialize: *Create initial population, NewPopulation, random*

ii. Evaluate: *Calculate_Fitness (NewPopulation)*

iii. Set: *Set Current Population = Initial Population*

iv. While NOT *(Terminal conditions)*

v. For *counter = 0 to PopulationSize do*

vi. Selection:

Parent1 = Heuristic_Select (Current Population)

Parent2 = Heuristic_Select (Current Population)

vii. Crossover:

Heuristic_Cross (Parent1, Parent2, NewPopulation)

viii. Repair:

Heuristic_Repair (NewPopulation)

ix. Mutation:

Mutate_Population (NewPopulation)

x. Evaluate: *Calculate_Fitness (NewPopulation)*

xi. Replace:

Replace_Population (Current Population, New Population)

xii. endwhile

xiii. Display Output

Initial population of solutions is generated stochastically for both HaGA and FaGA while observing stated constraints. At the hall level, a randomly generated number of students between 1 and allocation for the current category is generated and allocated into a randomly generated hall. The resulting population is evaluated, and if found okay, set as the current population. During each successive epoch, a proportion of the existing population is selected to breed a new generation of population using the roulette wheel selection. We have established in an earlier study that roulette wheel gives good results for the case under study in comparison with other selection strategies. Individual solutions are selected through a fitness-based process, with better solutions are typically more likely to be selected.

The algorithm however allows small proportion of less fit solutions to be selected sometimes. This helps to keep the diversity of the population large and prevents premature convergence on poor solutions. The relative fitness of each individual in a population is computed by dividing the cumulative fitness of the individual by the total fitness of the population. The first individual whose cumulative fitness is greater than a randomly generated number, r ε is selected for recombination. For each category, a single-point crossover strategy is used to generate a new offspring from selected parents based on a pre-defined crossover rate and a randomly generated number, r in [0,1]. If r is less than 0.5, the first offspring inherits the allocation for the category from the first parent while the second offspring inherits from the second parent otherwise the reverse is performed. The single-point crossover has also been compared with double-point crossover in an earlier study and proved to be effective.

The repair algorithm is invoked to ensure that generated chromosomes do not exceed the capacity constraint at the level under consideration. A similar process is carried out for the floor allocation while substitution floors for halls appropriately.

The repair algorithm, for example at the hall level, ensures that for each hall, the allocation for all categories does not exceed the capacity of the hall. A boolean variable is set to track all possible overflows. If there is an overflow for any, the difference between the total allocation and the capacity is computed and redistributed into other halls stochastically.
To achieve the redistribution, a category with non-zero allocation as well as a hall with non-zero allocation that has no space overuse is selected at random. We then compute the minimum, m1, between the overflow to be redistributed and the space remaining in the selected hall. Furthermore, the minimum, m2, between m1 and total allocation for the hall with overflow is computed. A random number, r ε [1, m2] is generated. The allocation for the hall with overflow is then decreased by r while that of the selected hall is increased by r.
This is repeated until all capacity constraints are satisfied.

Mutation is carried out based on pre-defined mutation rate and a randomly generated variable. At the hall level, two halls and two allocated categories are selected randomly. A random number r ε [1, m3] is generated where m3 is the smaller between the allocations for the first category in the first hall and the second category in the second hall. The allocation of the first category in the first hall is them increased by r while that of the second category in the same hall is decreased by r. The reverse is performed for the second hall to ensure that the hall capacities are not exceeded. The same mutation process is used for the block/ floor allocation with floors substituted for halls.

Both HaGA and FaGA are terminated if 1) an optimal solution is found ((i.e. one with $\sum w_q u_q = 1$ for all the entities considered), 2) for fifty consecutive generations, the average fitness of the current population is at least 95% that of the previous average fitness and 80% of the best fitness found throughout execution, 3) execution has been carried out over a specified number of generation without (1) or (2) being satisfied.

Category	*App[i]*	*rem*	*Min[App[i], rem]*	C_i
		1000		
C4	300	700	300	300
C5	450	250	450	450
C6	600	0	250	250
C7	100	0	0	0
C8	230	0	0	0
Total→1680			**Sum→1000**	

Table 6.1.1 Sample Illustration of the CaH1 Heuristic

Category	*App[i]*	Percentage Range [$Minp_i$,$Maxp_i$]	Allocation Range [$MinA_i$,$MaxA_i$]	Random x_i in [$MinA_i$,$MaxA_i$]	x_i→GD$_i$	C_i
C4	300	90%-95%	[270,285]	278	213	213
C5	450	80%-88%	[360,396]	395	302	302
C6	600	70%-80%	[420,480]	436	333	333
C7	100	60%-70%	[60,70]	66	50	50
C8	230	50%-60%	[115,138]	134	102	102
Total→1680			Sum→1309		Total Allocation→1000	

Table 6.1.2 Sample Illustration of the CaH2 Heuristic

	HA1	HA2	HB1	HB5	HC1	HC4	Total
Fo	2	6	4	0	0	8	20
Ht	70	0	0	0	0	0	70
Sp	0	0	0	0	400	0	400
Fy	531	6	473	139	17	74	1240
Fr	31	13	171	692	83	342	1332
Sc	0	400	0	0	0	0	400
Ds	8	14	61	0	16	1	100
Ot	18	5	91	137	10	87	348
Total	660	444	800	968	526	512	3910

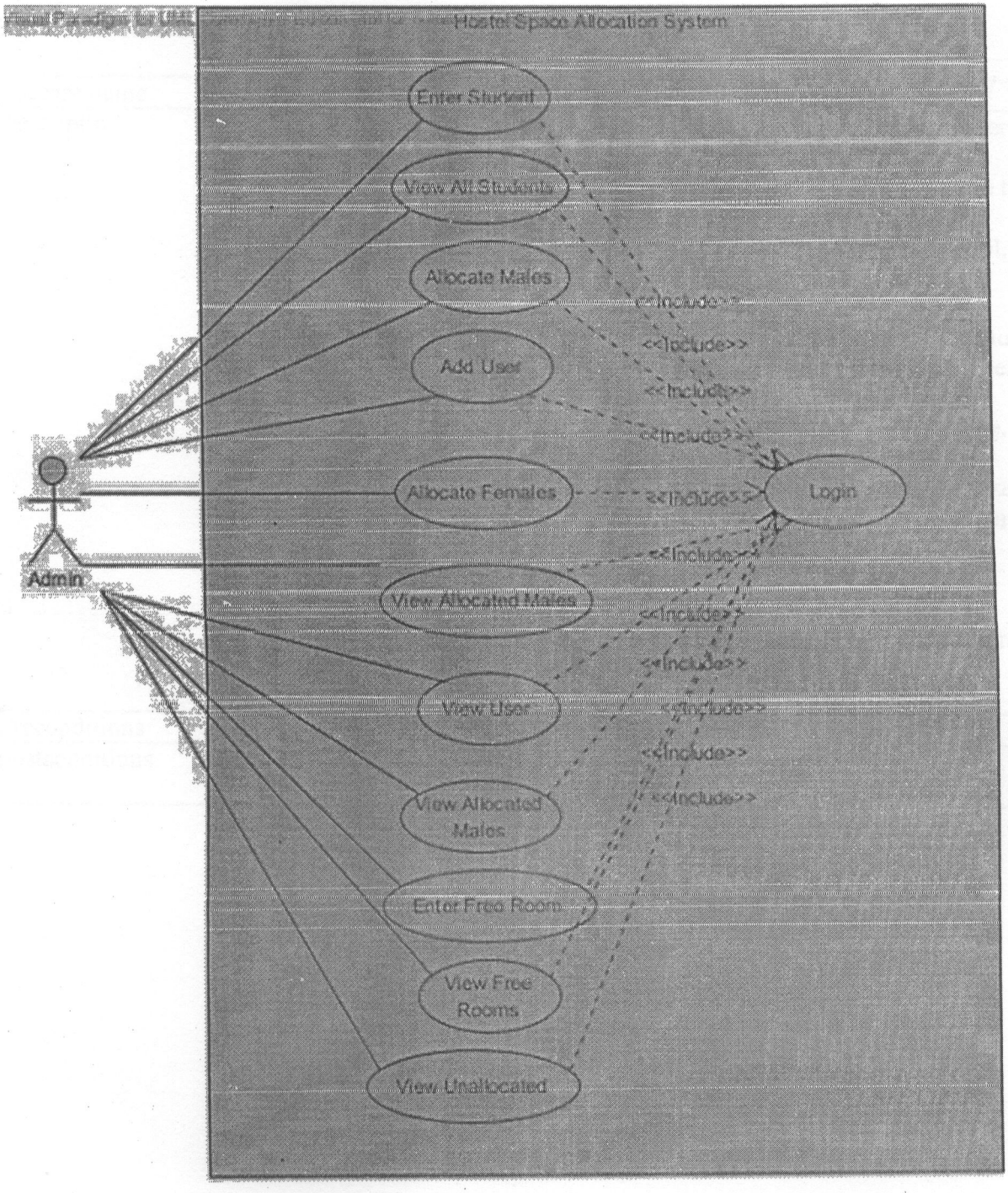
Table 6.1.3 Sample Illustration of the HaGA

	HA1	HA2	HB1	HB5	HC1	HC4	Total
Fo	4	0	5	6	1	3	20
Ht	70	0	0	0	0	0	70
Sp	0	0	0	0	400	0	400
Fy	241	18	326	395	51	209	1240
Fr	259	19	351	424	55	224	1332
Sc	0	400	0	0	0	0	400
Ds	19	1	26	32	4	17	100
Ot	68	5	92	111	14	59	348
Total	660	444	800	968	526	512	3910

Table 6.1.4 Sample Illustration of the HaNH

CHAPTER 3 : ANALYSIS

3.1 Usecases



3.2 Usecase descriptions

3.2.1 Enter student usecase description

Usecase name	Enter student
Description	1. The administrator clicks on the 'Enter student' button. 2. The System displays the form where to input the following information: Comp#, first name, last name, school, program, year, program, address, town/city, country, status (undergraduate or postgraduate), status of program (full time or part time), sex, marital status, time of application, health status and school performance. 3. Then the administrator clicks enter 4. The System checks for errors like, user entering character strings in a number field, out of bounds errors and incomplete Or unfilled forms. 4.1 The System displays the errors. 5. The user corrects the mistakes if any and clicks enter again.
Preconditions	The administrator must have logged in
postconditions	The system a message saying "Data successfully updated into the database".

3.2.2 Add user usecase description

Usecase name	Add user(administrator)
Description	1. The administrator clicks the "add user" button. 2. The system displays the form for entering the administrator with the following information: First name, last name, sex, user name and password. 3. Then the user(administrator) clicks enter. 4. The System checks for errors like, user entering character strings in a number field, out of bounds errors and incomplete Or unfilled forms. 4.1 The System displays the errors. 5. The user corrects the mistakes and clicks enter.
Preconditions	The user(administrator) must have logged in.
Postconditions	The system informs the user(administrator) that the update was successful by display the "User entered" message.

3.2.3 View user usecase description

Usecase name	View user(administrator)
Description	1. The administrator clicks the "View user" button. 2. The system asks for password. 3. The user enters the password.
Preconditions	The user(administrator) must have logged in
Postconditions	1. If user/password in database 1.1 System displays names 1.2 Else deny access

3.2.4 Enter free room usecase description

Usecase name	Enter free rooms
Description	1. The administrator clicks the "enter free room" button. 2. The system displays the form for entering the unoccupied room with the following information: Hostel name, hostel number, sex, floor number and room number. 3. Then the user(administrator) clicks enter. 4. The System checks for errors like, user entering character strings in a number field, out of bounds errors and incomplete Or unfilled forms. 4.1 The System displays the errors. 5. The user corrects the mistakes and clicks enter.
Preconditions	The user(administrator) must have logged in.
Postconditions	The system informs the user(administrator) that the update was successful by display the "Room entered" message.

3.2.5 View free room usecase description

Usecase name	View free room
Description	1. The administrator clicks the "View free room" button.
Preconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing unoccupied rooms with the following information: Hostel name, hostel number, sex, floor number and room number.

3.2.6 View female scores usecase description

Usecase name	View female scores
Description	1. The administrator clicks the "View female scores" button.
Preconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing rankings of female students in descending order with the following information: Comp#, first name, last name, year and the.

3.2.7 View male scores usecase description

Usecase name	View male scores
Description	1. The administrator clicks the "View male scores" button.
Preconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing rankings of male students in descending order with the following information: Comp#, first name, last name, year and the.

3.2.8 View Allocated female scores usecase description

Usecase name	View allocated females
Description	1. The administrator clicks the "View allocated females" button.
Prcconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing the allocated female students starting with one having the highest rank, followed by the second and so on

3.2.9 View allocated male scores usecase description

Usecase name	View allocated males
Description	1. The administrator clicks the "View allocated males" button.
Preconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing the allocated male students starting with one having the highest rank, followed by the second and so on

3.2.10 View all students' scores usecase description

Usecase name	View all students
Description	1. The administrator clicks the "View all students" button.
Preconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing all the students in the System prior to allocation.

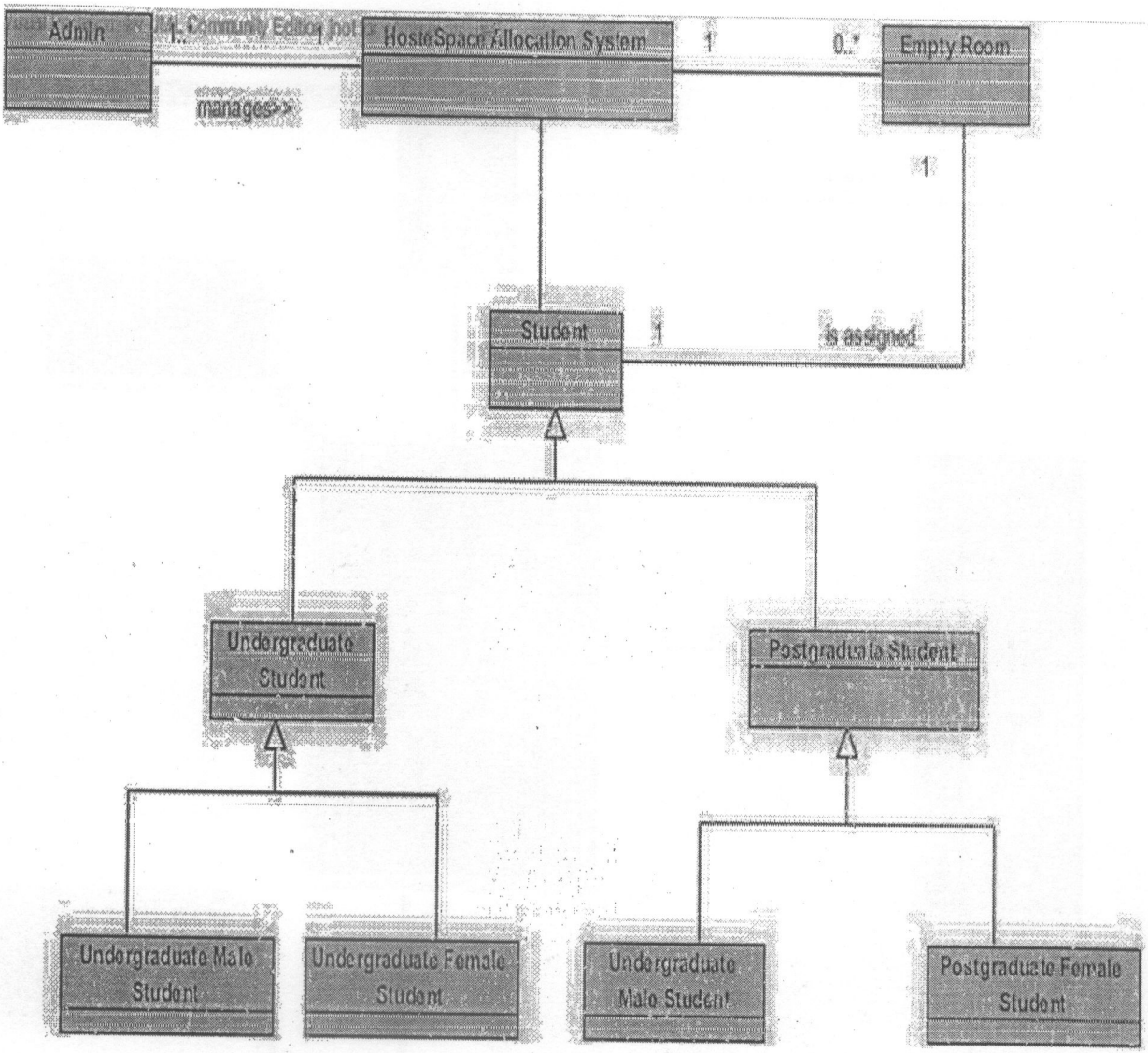
3.2.11 View female scores usecase description

Usecase name	View Unallocated students
Description	1. The administrator clicks the "View Unallocated students" button.
Preconditions	The user(administrator) must have logged in
Postconditions	The system displays a table showing all the unallocated students in the System.

3.3 Objects in the domain

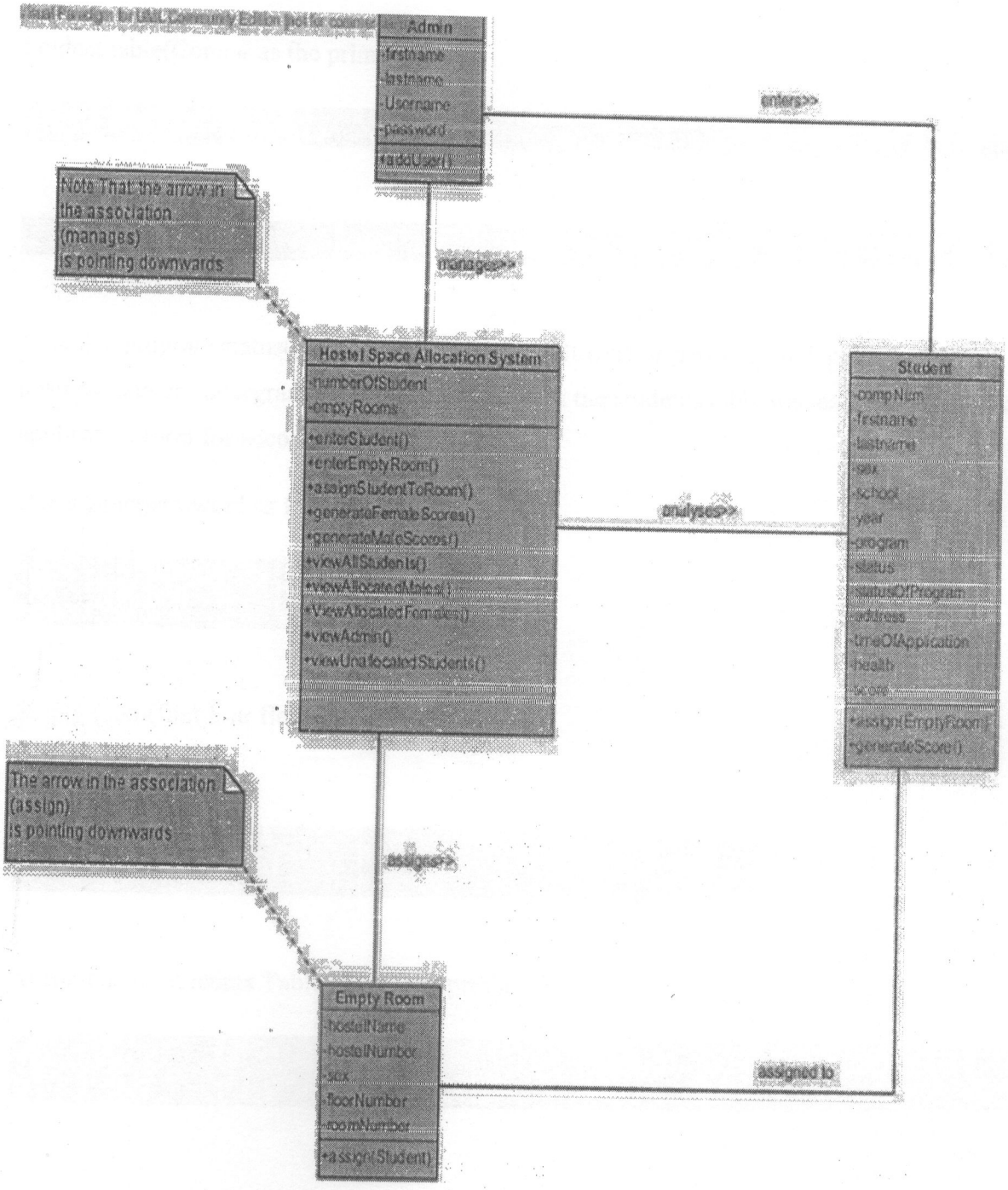
- User(Administrator)
- Female Student
- Male Student
- Bed Space
- Allocated Room
- Empty Room
- Hostel Allocation System

3.4 Domain analysis



CHAPTER 4 : DESIGN

4.1 Detailed class diagrams



4.2 Database design

The following are the table tables used by the system

Student table(Comp# as the primary key)

occupied	occup_id	school	program	year	health_st	marital	nationalit	res_addr	fname	lname

sex	prog_status	status	year	health	marital_st	address	school_per	date_applied

Note that program status(prog_status) indicates full-time or part-time and status indicates postgraduate or undergraduate. This information in the students table was extracted from the application form for accommodation.

Users table(password as Primary Key)

fname	lname	sex	username	password

Empty room(first four fields as primary key)

hostel_nam	hostel_numbr	floor_number	room_number	sex

Accomodated students Table (comp# as primary key)

comp#	fname	lname	hostel_nam	hostel_numbr	floor_number	room_number	sex

CHAPTER 5: EMPLEMENTATION

5.1 Brief system description

The hostel space allocation was developed using java and mysql database. The system allocates bed spaces to students using a number of contraints. These constraints are assigned priorities(weights) in form of numbers. Then these weights are added for each student then ranks them in descending order. The student with the highest score gets the room first, followed by the second and so on. The code snippet doing these computations is as shown below

```
else if(str6.equalsIgnoreCase("full-
time")&&str7.equalsIgnoreCase("undergraduate")&&str15.equalsIgnoreCase("F")&&(str8.equal
sIgnoreCase("2")||str8.equalsIgnoreCase("3")))
                    {

    undergraduate_female_full_time_mid_year.add(rs.getString(1));/*A list saving full-time
female students who are in neither first year nor final year*/
                        double cummulativeWeight=0;
                        cummulativeWeight+=0.1;
                        if(str9.equalsIgnoreCase("handicapped"))
                        {
                            cummulativeWeight+=0.3;
                        }
                        else
                        {
                            cummulativeWeight+=0.1;
                        }
```

```
if(str14.equalsIgnoreCase("zambian"))
{
        cummulativeWeight+=0.1;
}
else
{
        cummulativeWeight+=0.2;
}

if(dateOfApplication.equalsIgnoreCase("early"))
{
        cummulativeWeight+=0.2;
}
else if(dateOfApplication.equalsIgnoreCase("late"))
{
        cummulativeWeight+=0.1;
}

if(str16.equalsIgnoreCase("low"))
{
```

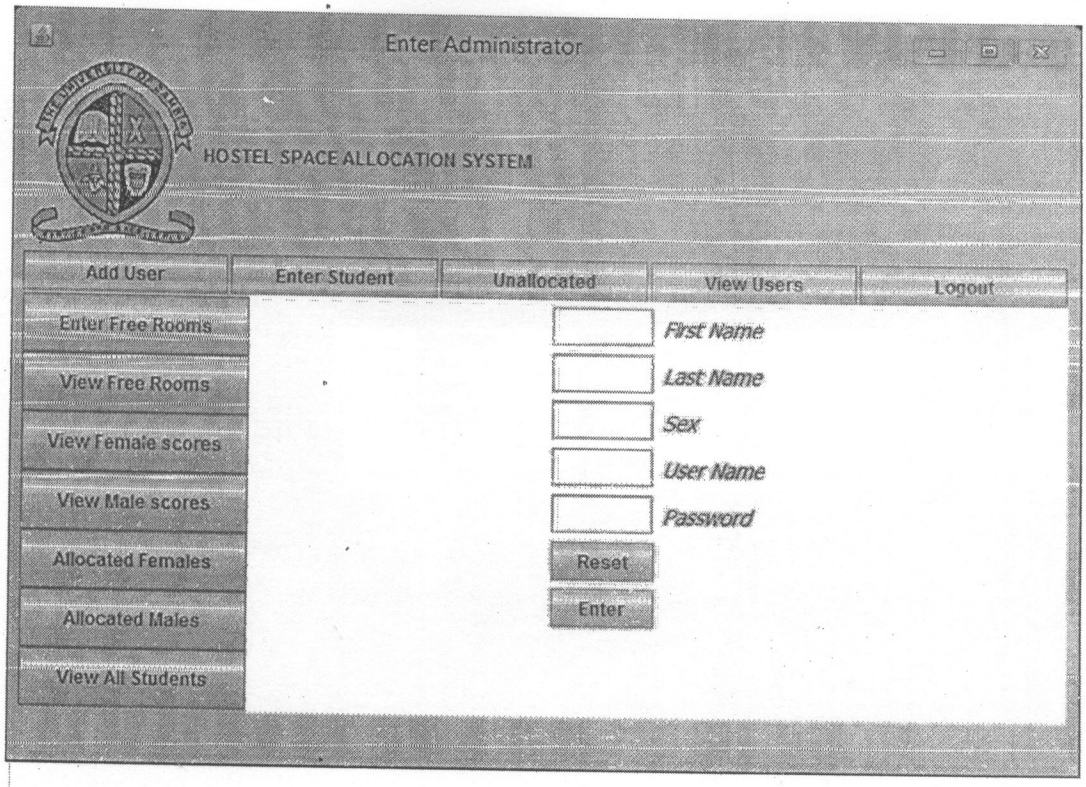
```
                cummulativeWeight+=0.1;
            }
            else if(str16.equalsIgnoreCase("medium"))
            {
                cummulativeWeight+=0.2;
            }
            else if(str16.equalsIgnoreCase("high"))
            {
                cummulativeWeight+=0.3;
            }
```

The priorities are assigned as follow:

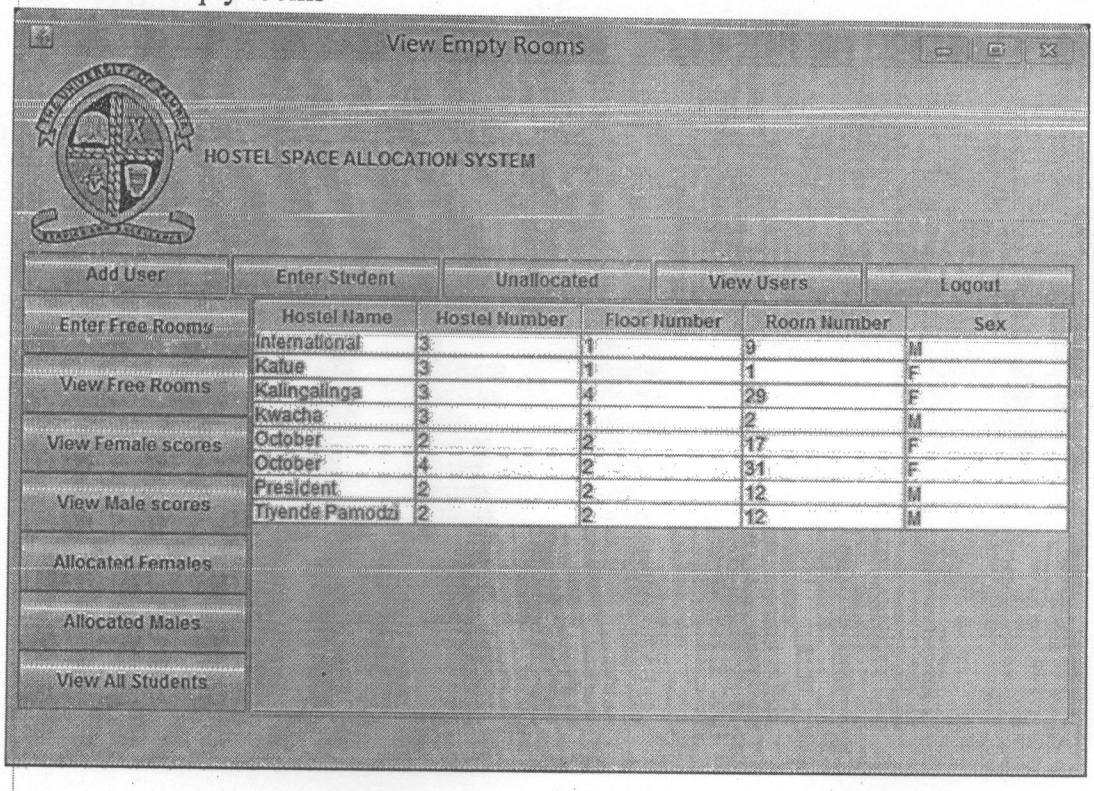
1. Year has a maximum of 0.3 broken down into final year students are given 0.3 cause their year is crucial; they have to concentrate on projects, first years has 0.1 and mid years have a slight advantage having 0.2;
2. School performance has also 0.3 only applies to returning, Students commended by the school gets the highest of 0.3. That is students having B+ and higher in all the courses for a particular year. Students having atleast 3 B+ are given a middle priority of 0.2 and the rest are given 0.1.
3. Time of application also applies only to first years, those who applies early are given a priority of 0.2 and those late are given 0.1
4. To encourage more foreigners and as a source of revenue for the University they are given 0.2 and Zambians are given 0.1.
5. The handicapped also are given a priority of 0.3 and the normal are given 0.1

If you add up all these priorities, the maximum an individual can go is 1.3 points. For the full implementation of this system refer to the disk attached to this Document.

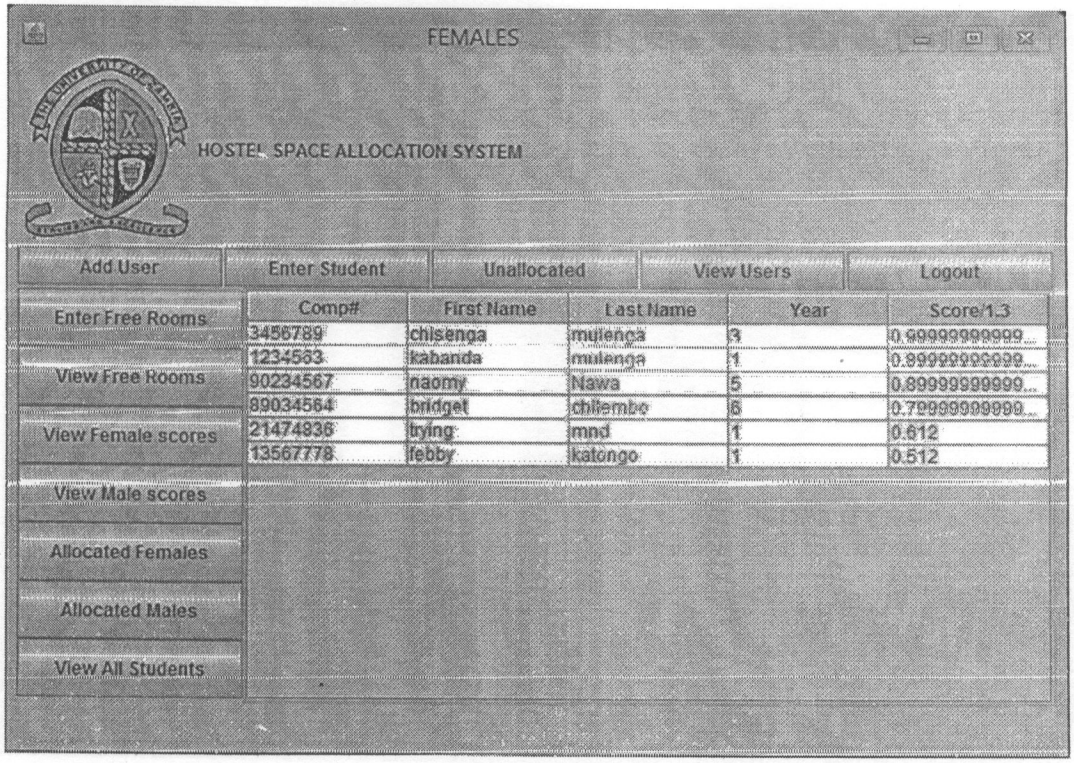
5.2 System screenshots - Add user page



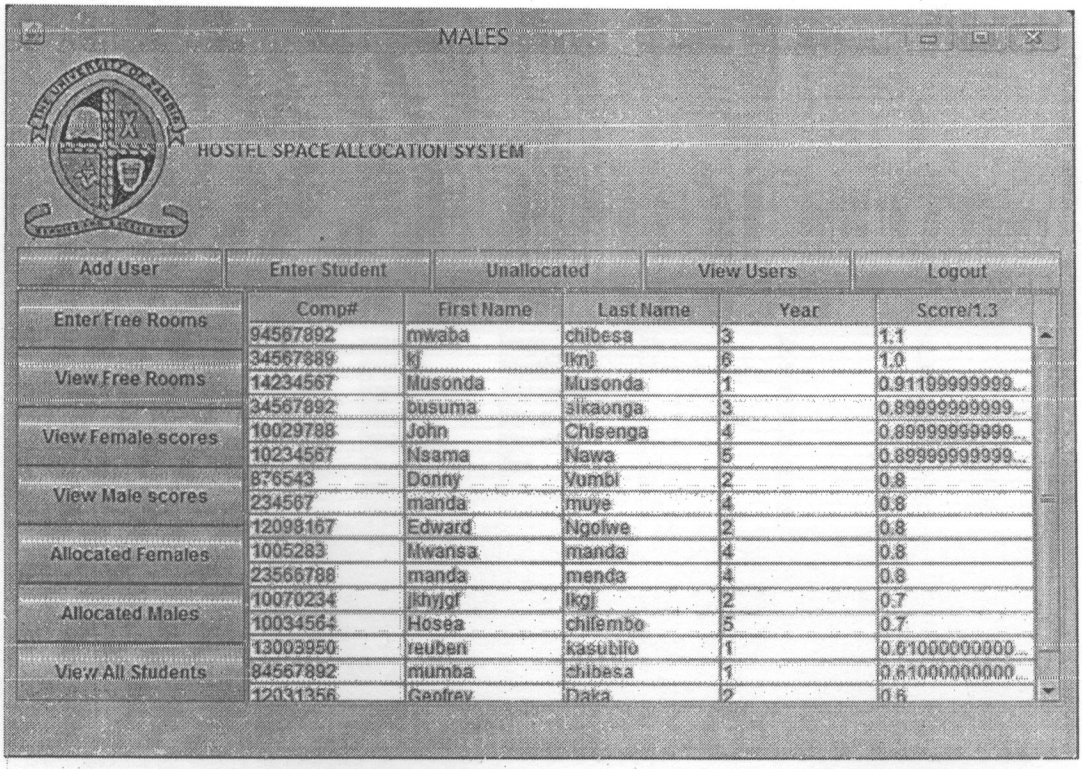
5.2.2 View empty rooms



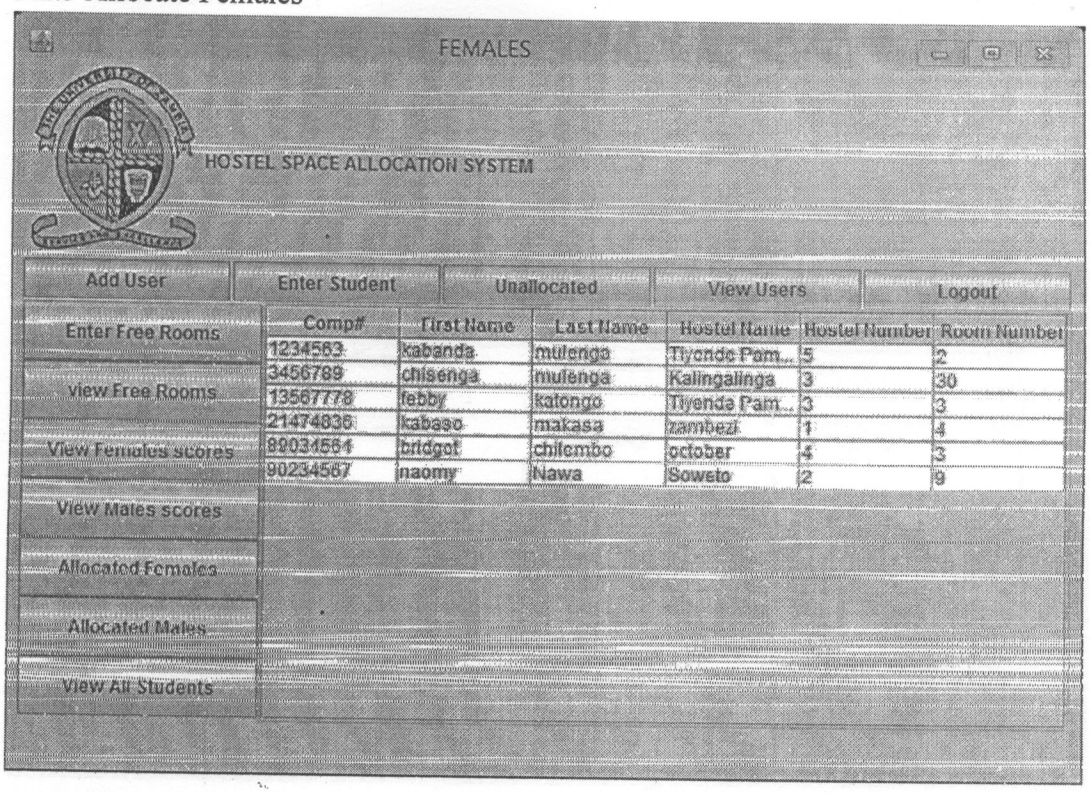
5.2.3 Generate female scores



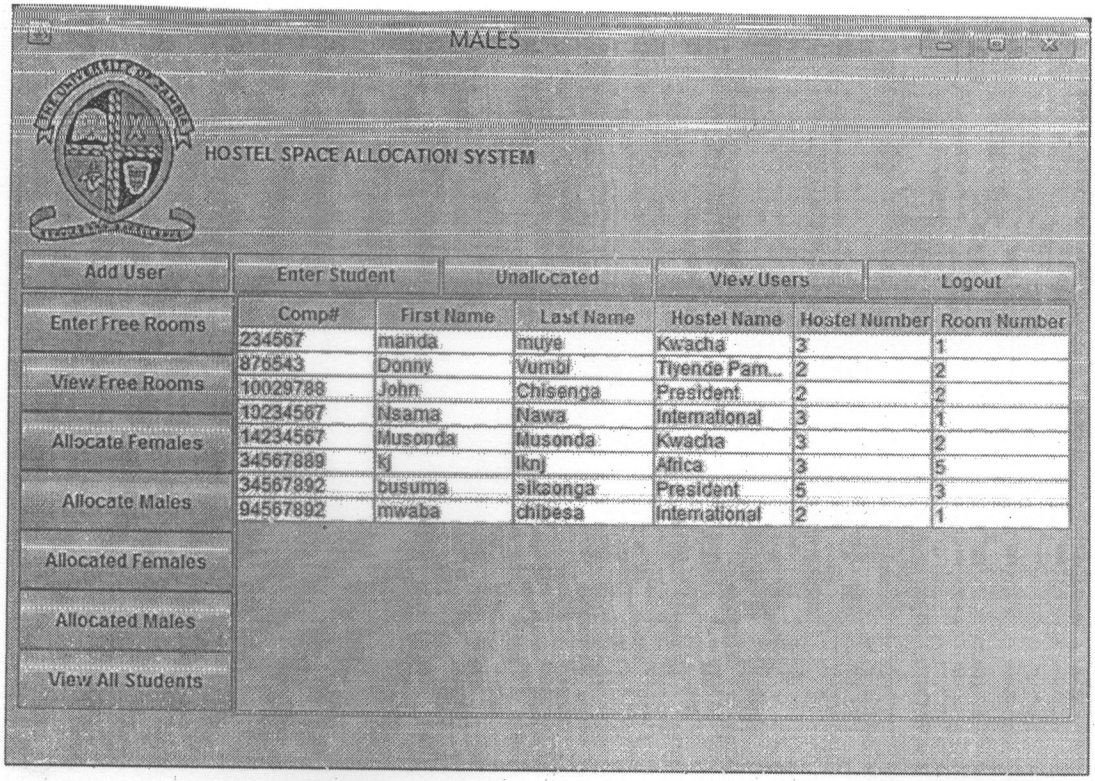
5.2.4 Generate male scores



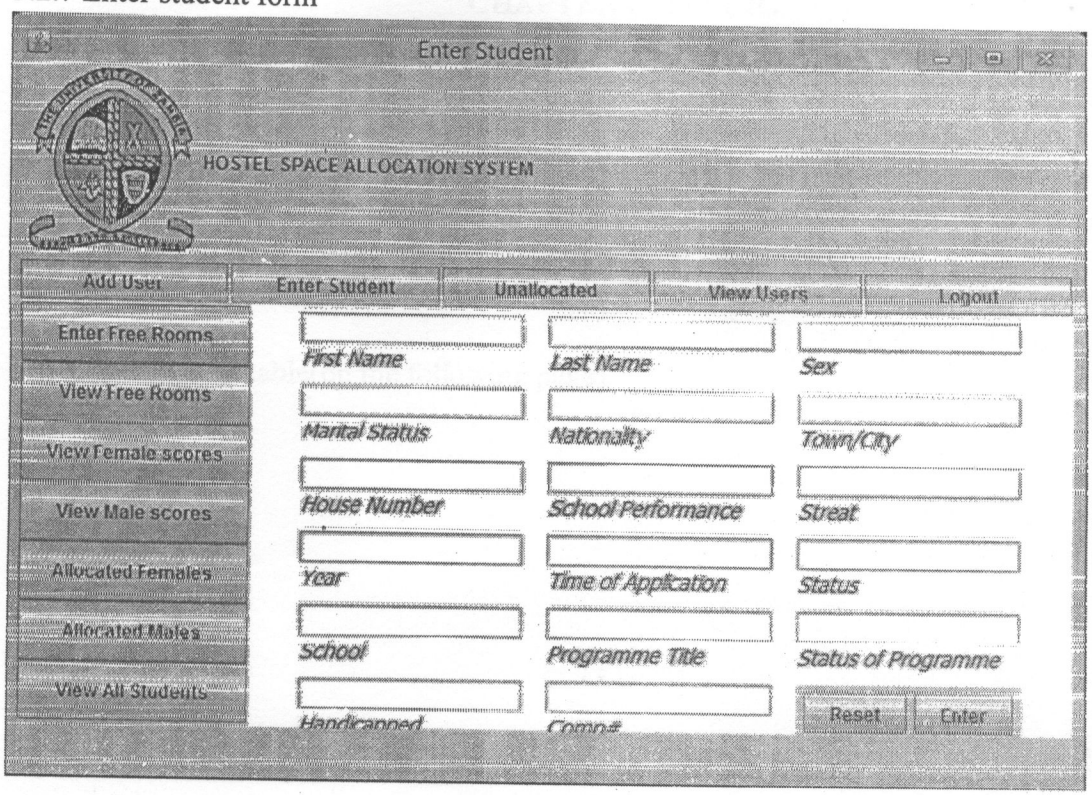
5.2.5 Allocate Females



5.2.6 Allocate males



5.2.7 Enter student form



CHAPTER 6: TESING

Every page of the system was tested and the result of each test is as shown below:

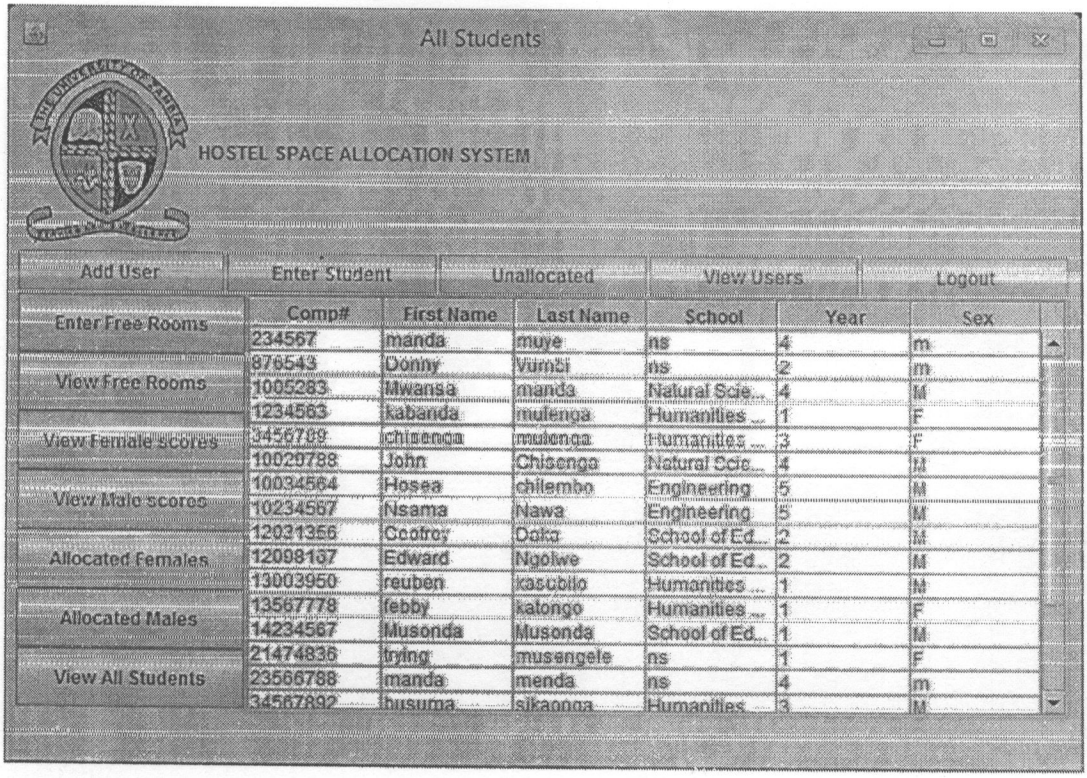
We will demonstrate the algorithm by feeding the System with the following information from the database for different Students shown below(The information is in the students database):

6.1 Testing the operations of the system

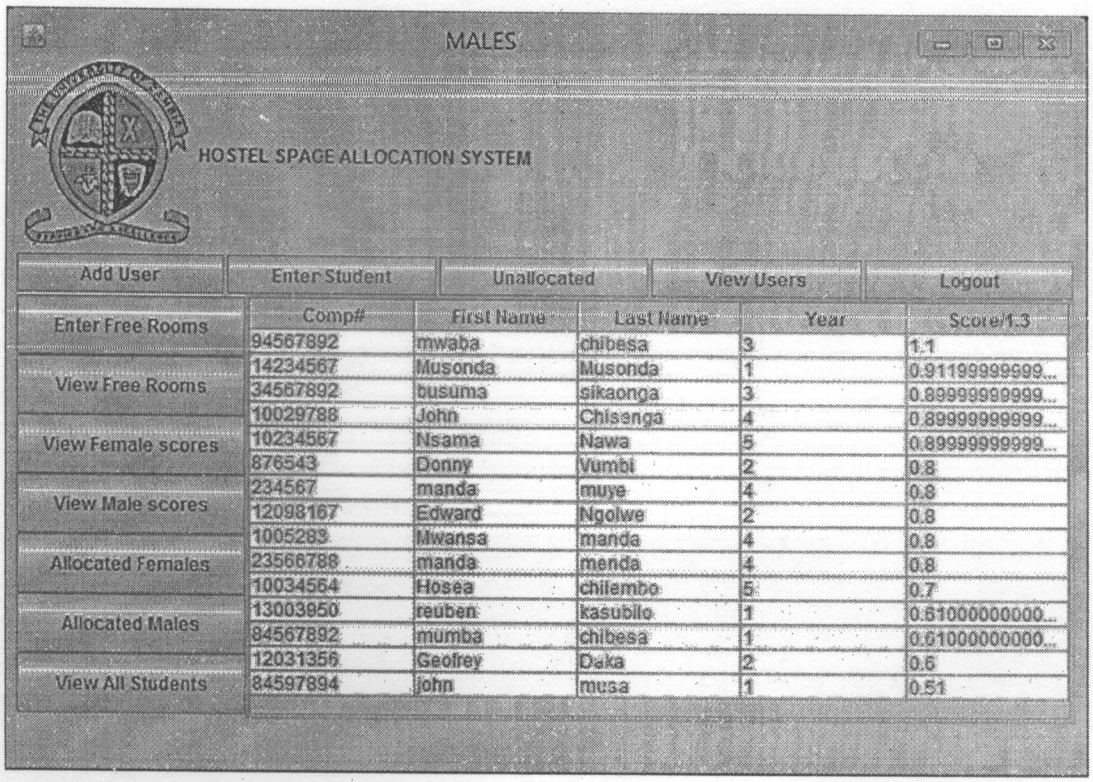
6.1.1 The students table(on the following page)

occup_id	fname	lname	school	program	program_status	undergraduate_or_postgraduate	year	health_status	marital_status	house_number	street	town	nationality	sex	performance	date_of_application
234987	manda	muye	ns	ms	full-time	undergraduate	4	normal	single	jh4	sd	mufulira	zambian	m	medium	late
876642	Donny	Vumbi	ns	as	full-time	postgraduate	2	normal	single	ghjk	bbb	Lusaka	zambian	m	medium	early
1005283	Mwansa	manda	Natural Sciences	Bsc Comp Scie	Full-Time	undergraduate	4	normal	single	v145	Kamuchanga	Mufulira	Zambian	M	medium	late
1234563	kabanda	mulenga	Humanities and Social Sciences		parallel	undergraduate	1	handicapped	married	qw 34	jumbo	Harare	Zimbabwean	F	low	early
3456789	chisenga	mulenga	Humanities and Social Sciences	BA LLIS	parallel	postgraduate	3	handicapped	married	qw 34	kisomo	Lilongwe	Malawian	F	low	early
10025788	John	Chisenga	Natural Sciences	Bsc Comp Scie	Full-Time	undergraduate	4	normal	single	g 45	buyantanshi	Kalulushi	Zambian	M	high	late
10034564	Hosea	chilembo	Engineering	Mechanical engineering	Full-Time	undergraduate	5	normal	single	h145	suburbs	Mufulira	Zambian	M	medium	
10234567	Nsama	Nawa	Engineering	Civil Engineering	Full-Time	undergraduate	5	handicapped	single	123 n	kabulonga	Lusaka	Zambian	M	married	early
12031368	Geofrey	Daka	School of Education	BaEd EED	Full-Time	undergraduate	2	normal	single	B3/25	Kalingalinga	Lusaka	Zambian	M	low	late
13098167	Edward	Ngoma	School of Education	BaEd EED	Full-Time	undergraduate	2	normal	single	10 KTC	Olympia	Lusaka	Zambian	M	medium	early
13008960	reuben	kasublo	Humanities and Social Sciences	Bachelor of arts	Full-Time	undergraduate	1	normal	single	vm 145	Kabangwe	Lusaka	Zambian	M	high	early
13687778	febby	katongo	Humanities and Social Sciences	social work	Full-Time	undergraduate	1	normal	single	V145	Kamuchanga	Mufulira	Zambian	F	low	late
14234567	Musonda	Musonda	School of Education	Civic Education	Full-Time	undergraduate	1	handicapped	single	123 ty	kabulonga	Mongwe	Kenyan	M	married	early
21474838	tryng	musengele	ns	ed	full-time	undergraduate	1	normal	married	34	gf	ufhy	ox	F	good	late
23686788	manda	menda	ns	bsc comp s	parallel	undergraduate	4	normal	single	d23	kimoto	mufulira	zambian	m	high	late
34567892	busuma	sikaonga	Humanities and Social Sciences	BA LLIS	Full-Time	postgraduate	3	handicapped	devorced	bn 567	vvn	Lilongwe	Malawian	M	low	late
84567892	mumba	chibesa	Humanities and Social Sciences	Economics	parallel	undergraduate	1	normal	single	r66	busakile	Kitwe	Zambian	M	low	early
84567894	john	musa	School of Education	physics	parallel	undergraduate	1	normal	single	s123	chifubu	ndola	Zambian	M	low	late
89084684	bridget	chilembo	VET	Mechanical engineering	Full-Time	undergraduate	6	normal	single	h145	suburbs	Mufulira	Zambian	F	low	early
90234567	nawny	Nawa	Engineering	Civil Engineering	Full-Time	undergraduate	5	handicapped	single	123 n	kabulonga	Lusaka	Zambian	F	married	early
94567892	mwaba	chibesa	Humanities and Social Sciences	Economics	parallel	postgraduate	3	handicapped	single	r66	busakile	Kitwe	Zambian	M	high	early

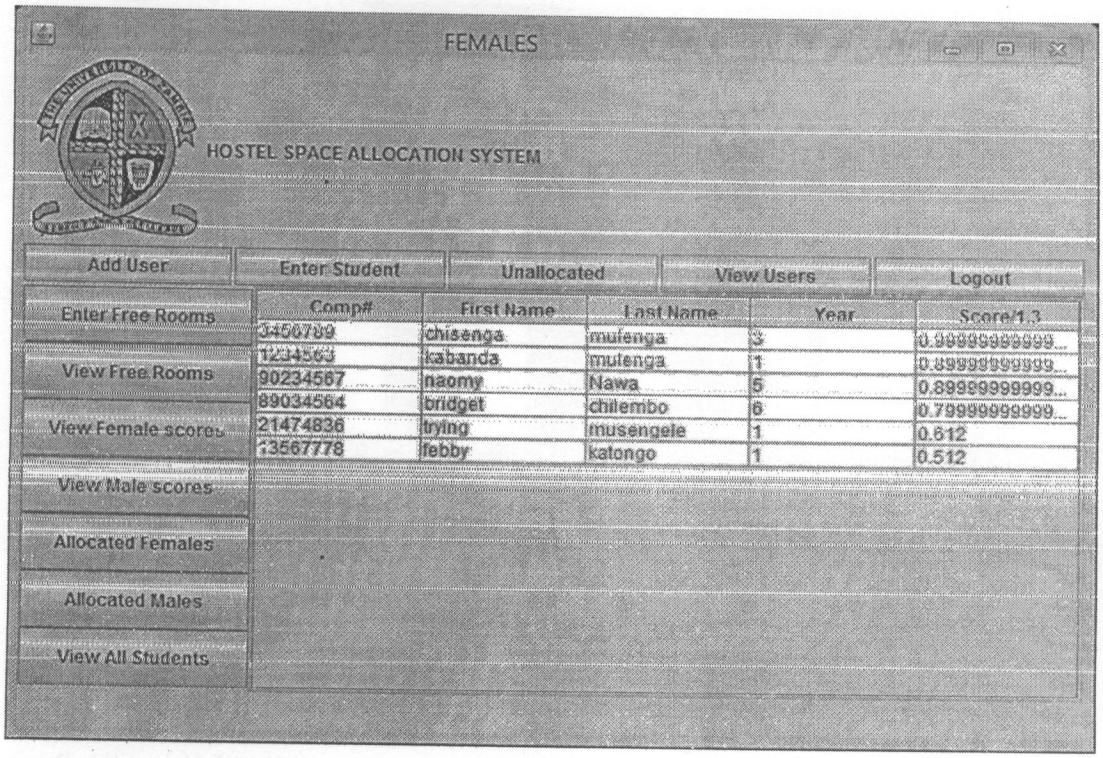
6.1.2 All the students



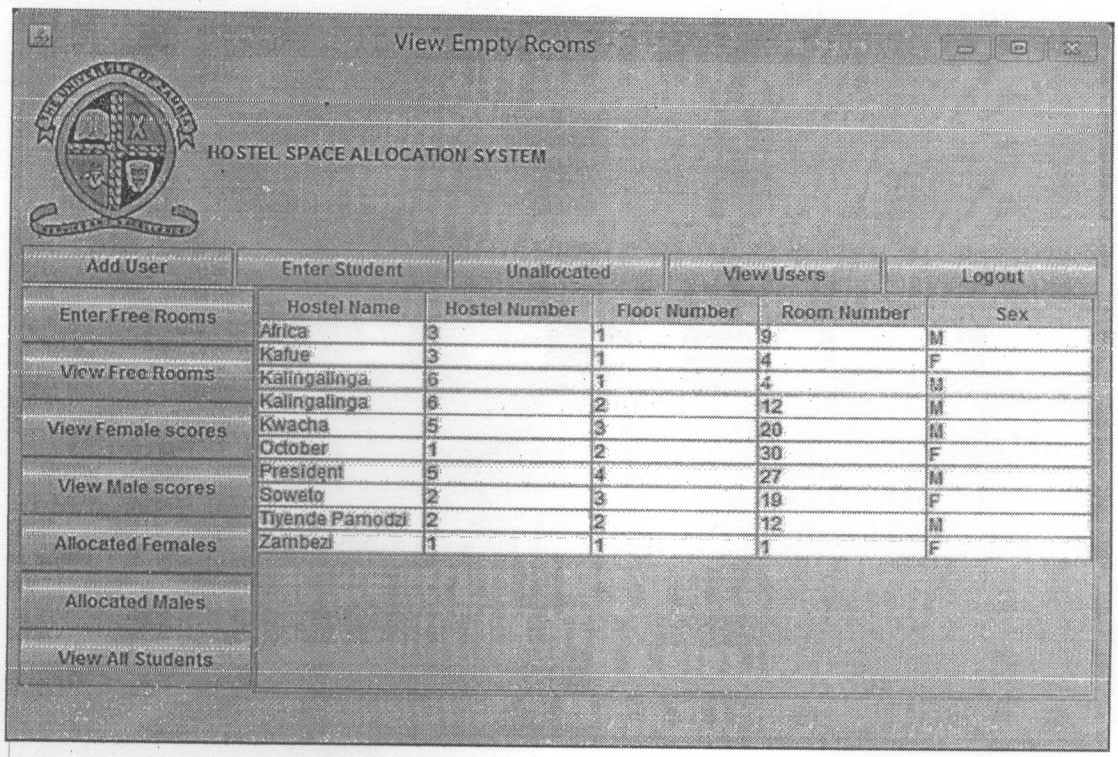
6.1.3 System showing rankings of male students' scores



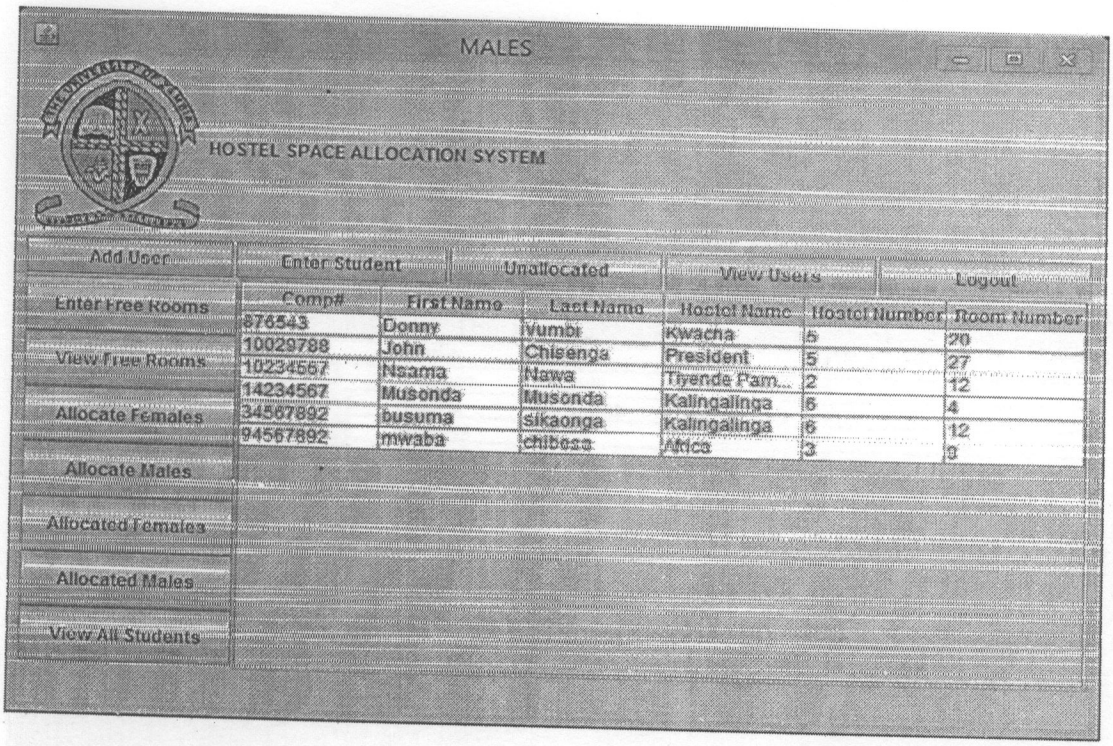
6.1.4 System showing rankings of male students' scores



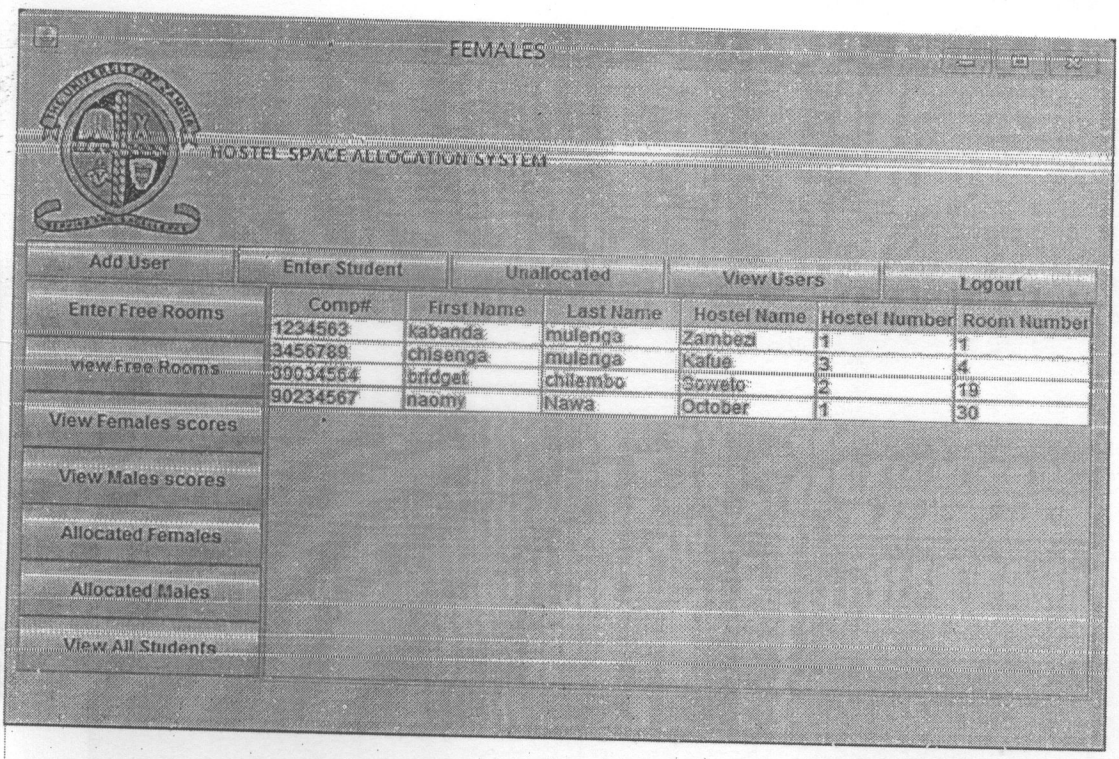
6.1.5 After entering the empty rooms



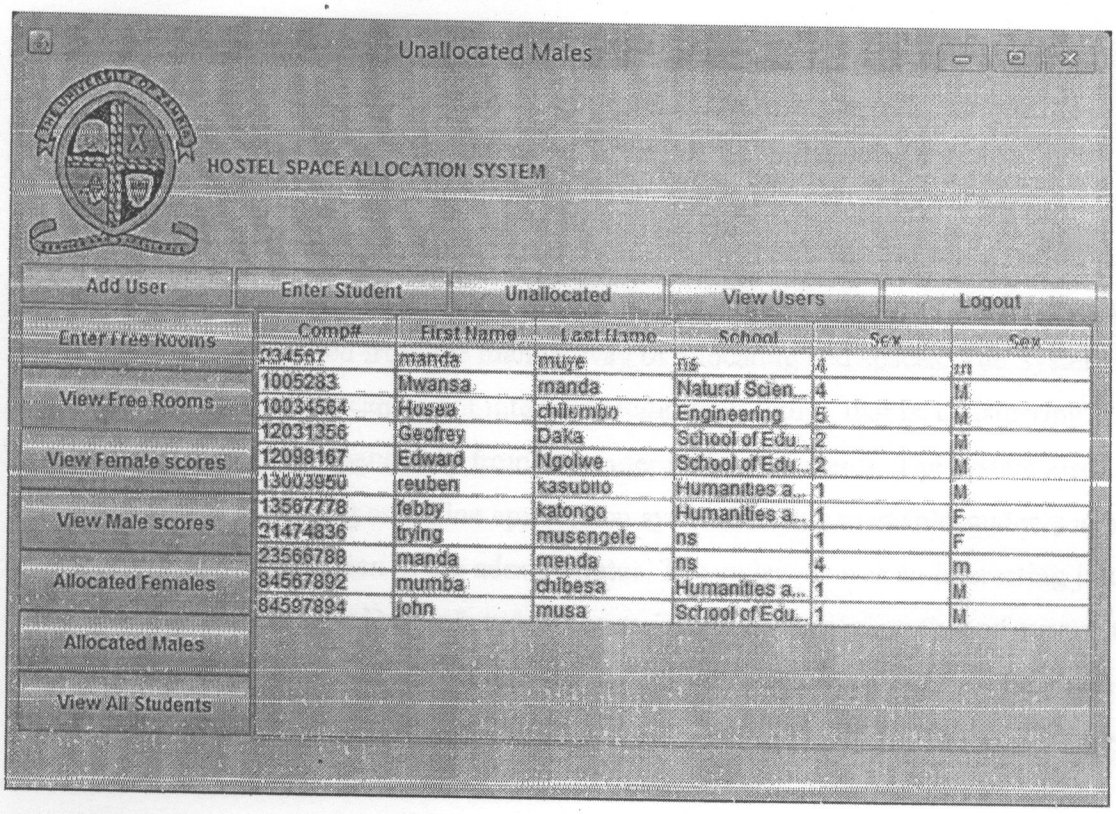
6.1.6 After accommodating males



6.1.7 Allocated females



6.1.8 Unallocated



Input	Result	Actual
NumberFormat Exception (entering a string in a number input)	Appropriate error message	√
Required but unfilled fields	Enter all the stated must fill fields	√
Wrong username/password	Error message: wrong username/password combination	√

6.1.9 Error messages

CHAPTER 7 : DISCUSSION CONCLUSION

The project was carried out to see if applying a number of constraints, that is properties characterizing students can yield good results in allocating bed spaces to students. According to this survey the results were overwhelming and can greatly and instantly have a positive impact in the way bed spaces are allocated. The System can greatly help the Dean of Students to help them reduce the pressure resulted from the manual way of allocating bed spaces. The System has some limitations however which can be put into consideration in future, that is, considering postgraduate students independently from the undergraduate students. The system also has to be developed further by adding an online application system module to enable students apply online and therefore less intervention of the administrator. The endeavor was an interesting journey and if implemented can help the Dean of students to assign rooms transparently and every Student be happy at the end. Therefore, it is my hope that the system shall continue developing and help other institutions in the time to come.

5. REFERENCES

[1] R. Bai, *An Investigation of Novel Approaches for Optimizing Retail Shelf Space Allocation.* PhD Thesis. School of Computer Science and Information Technology, University of Nottingham, UK, 2005.

[2] J.D. Landa-Silva, *Metaheuristics and Multi-objective Approaches for Space Allocation.* PhD Thesis. School of Computer Science and Information Technology, University of Nottingham, UK, 2003.

[3] A.O. Adewumi, N, Ihemedu and J.O.A. Ayeni, *An Evolutionary-based Approach to University Course Time-Tabling Problem. First Annual Research Conference and Fair,* University of Lagos, October 26, (2005) No. 9.11

[4] H.L. Morgan, *Optimal Space Allocation on Disk Storage Devices. Communications of the* ACM, ACM New York, NY, USA, 17 (3) (1974) 139-142.

[5] D.C. Mattfeld, *Allocation of Storage Space.* In. D.C. Mattfeld, The Management of Transshipment Terminals, Operations Research/Computer Science Interfaces Book Scries 35 (2006) 83-108.

[6] E.K. Burke and D.B. Varley, *Space Allocation: An Analysis of Higher Education Requirements.* In: E.K. Burke and M.W. Carter (eds.) *The Practice and Theory of Automated Timetabling II*: Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling PATAT 1997, Lecture Notes in Computer Science, Springer, 1408 (1998) 20-33.

[7] E.K. Burke, P. Cowling, J.D. Landa-Silva, B. McCollum, *Three Methods to Automate The Space Allocation Process in UK Universities,* Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000, Konstanz, Germany, (2000) 374-393.

[8] Wikipedia. *Heuristic algorithm. Accessed via http://en.wikipedia.org/wiki/Heuristic algorithm* in October 2009.

[9] F. Glover, *Future paths for integer programming and links to artificial intelligence.* Computer & Operations Research, 13 (5) (1986) 533–549.

[10] M. Gendreau, *An Introduction to Tabu Search.* In. F. Glover and G.A. Kochenberger (eds.). Metaheuristic Handbook, Kluwer Academic Publishers, Boston, MA, (2003) 37-54

[11] A. Petrovski, A. Wilson and J. McCall. *Statistical Identification and Optimization of Significant GA Factors.* Technical paper, The Robert Gordon University, School of Computer and Mathematical Sciences, Scotland, 2007

[12] A.O. Adewumi, B.A. Sawyerr and M.M. Ali, *A Heuristics Approach to the University Timetabling Problems.* Engineering Computations, Emerald Publishers, UK (2009) To appear.

[13] E.K. Burke and D.B. Varley. *Automating Space Allocation in Higher Education.* Selected Papers from the 2nd Asia Pacific Conference on Simulated Evolution and Learning SEAL 98, Lectures Notes in Artificial Intelligence, Springer, 1585 (1998) 66-73.

[14] M. Yang and W. Chen, *A Study of Shelf Space Allocation and Management.* International Journal of Production Economics, Elsevier, 60-61 (1999), 309-317.

[15] E. Annevelink and R.A.C.M. Broekmeulen, *The Genetic Algorithm applied to Space Allocation Planning in Pot-Plant Nurseries.* XII International Symposium on Horticultural Economics Acta Hort. (ISHS) 340 (1995) 141-148.

[16] J.S. Dean, *Staff Scheduling by a Genetic Algorithm with a Two-Dimensional Chromosome,* In E.K. Burke, M. Gendreau, (Eds.), Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling – PATAT 2008, Montreal, Canada, August 18-22, 2008, 15 pgs

[17] Futminna, *Student Affairs Department. Federal University of Technology,* Minna, www.futminna.org/currentstudents/student_affairs.htm. Retrieved in January, 2009.

[18] M&G. *Varsities run out of housing.* Mail & Guardians (South African) Newspaper article on higher learning, September 23, 2009 pg 31.

[19] D. Pisinger. *An exact algorithm for large multiple knapsack problems.* European Journal of Operations Research. 144 (1999) 528–541.

[20] A. Federgruen, G.V. Ryzin. *Probabilistic Analysis of a Generalized Bin Packing Problem and Applications.* Operations Research. 45(4) (1997) 596-609.

[21] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[22] J.R. Koza, 2007. *Genetic Algorithms and Genetic Programming. Presentation.* Department of Electrical Engineering, School of Engineering, Stanford University, Stanford, California. http://www.smi.stanford.edu/people/koza/, Retrieved in August.

[23] D.E. Goldberg, Genetic Algorithms in Search, optimization and Machine Learning. Addison Wesley, 1989

[24] M. Mitchell, *An Introduction to Genetic Algorithms.* The MIT Press, 1998.

[25] K.A. Dowsland. *Genetic Algorithms - a tool for OR?* Journal of the Operational Research Society 44 (1996), 550-561 [ISSN 0160-5682]

[26] A.O. Adewumi. *Some Improved Genetic-Algorithms Based Heuristics for Global Optimization with Innovative Applications.* PhD thesis to be submitted in 2009, School of Computational and Applied Mathematics, University of Witwatersrand, South Africa.

[27] A.O. Adewumi and M.M. Ali. *A Multi-level Genetic Algorithm for a Multi-Stage Space Allocation Problem.* Mathem