UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS

SYSTEM: WEB ENABLED AGENT-BASED UNIVERSITY TIMETABLING ASSISTANT

BY: STEPHANIE GAKU

This project has been submitted as part of the fulfilment of the degree of Bachelor of Science of the University of Nairobi.

JUNE 2012

DECLARATION

The project presented in this report is my original work and has been done with full support of my supervisor as part of the fulfilment for Bachelor of Science in Computer Science.

Signed ....

Date ....

Gaku Stephanie – P15/23197/2008

This project has been submitted as part of the fulfilment for the Bachelor of Science in Computer Science with my approval as a lecturer in the University of Nairobi, School of Computing and Informatics and as the Project Supervisor.

Signed ....

Date ....

Mr. Elisha Opiyo

School of Computing and Informatics

University of Nairobi.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for giving me the wisdom and strength to develop this project to completion. I'd also want to thank my supervisor, Mr Elisha Opiyo, for being available and resourceful at all times in the development phases of this project. His support and ideas were major driving factor for this project.

I also thank my fellow classmates for their support and conducive environment they offered for the period I undertook this project. I also thank School of Computing and Informatics for convenience in opening the library and the computer laboratory which were imperative in research done for the project.

And last but not least, I would like to show great appreciation to my family for being there at all times and for the unconditional support they have given me throughout this period.

Thank you all.

Stephanie Gaku.

ABSRACT

The university course timetabling problem is a difficult problem that most universities face on a regular basis. It is considered as an iterative and time consuming process, for it involves a lot of negotiating and compromising of resources in order to resolve any conflicts that may arise during the process. Currently the generation of these timetables is done manually and the results are often not satisfactory as they do not fully optimize the institutions' resources. Some educational institutes have tried solving this problem by the automation of this process whereby mathematical models are used to deal with the scheduling problems. Such an approach requires the generation of multiple mathematical models so as to satisfy the requirements of different educational institutes. Therefore, a web based multi-agent timetabling assistant, is proposed to solve the timetabling problem. In this case,an agent is a computer system that is situated in some environment and is capable of independent action so as to meet its design objectives. The proposed system will use agents to compete for a number of resources on a given number of time slots whereby negotiation will be used as a means for reaching compromises among agents with conflicting interests. Such a system will significantly reduce the amount of time and energy spent ingenerating timetables and at the same time, optimize on all the available resources as a result of using agents. Consequently, the utilization of agents makes the system flexible whereby it is able to adapt to different scenarios and behave in an appropriate manner and also the fact that it's web-based makes it easily accessible to its users.

Table of Contents

LIST OF ABBREVIATIONS

NP	Non-Deterministic Polynomial
MAS	Multi-Agent System
CHC	Common Hard Constraints
HC	Hard Constraints
SC	Soft Constraints
AMAS	Adaptive Multi-Agent Systems
NCS	Non Cooperative Situations

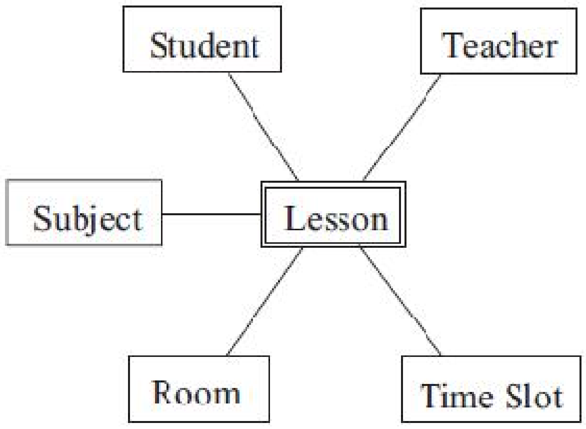
CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

1.1.1 General Timetabling

According to Collins Concise Dictionary (4th Edition) a *timetable* is a *table of events arranged according to the time when they take place*. The events are usually meetings between people at a particular location. Consequently, a timetable specifies which people meet at which location and at what time. A timetable must meet a number of requirements and should satisfy the desires of all people involved simultaneously in the best possible manner. The timing of events must be such that nobody has more than one event at the same time.

With respect to the above considerations,the process of coming up with a timetable for any learning institution is considered as a time consuming and interactive process that involves a lot of compromising on both the human and technical resources offered by the university. The technical resources would be the class rooms and the human resources would be the lecturers and the students.Hence, university course timetabling problems are combinational problems which consist of scheduling a set of courses within a given number of class rooms and time periods. The diagram below illustrates the relation between concepts of timetabling.



In order to come up with a feasible timetable, it is important to compromise between all lecturers' time preferences and all also keeping in mind availability of classrooms and students, number of students and curricula.

1.1.2 Timetabling in the University

The university comprises of a number of schools, all located in different campuses. Each and every school has its own timetable and over the years, the university has been generating these timetables manually. This process is time consuming and the end result i.e. the timetables generated, may or may not be satisfactory.

With respect to the project scope, it would be ideal to zero in to the College of Physical and Biological Sciences that comprises of various schools that individually deal with their own majors. For every major there is a set of associated courses, which can either be compulsory or elective. The lecturers are shared between schools depending on the courses they teach and there are incidents whereby students from different schools need to undertake a common course that will require students to occupy shared room resources.

The timetable has a span of five days, starting from Monday to Friday for the undergraduate students and each day is divided into various time slots whereby the time slot values varies from school to school. The events that can take place in a time slot can either be a lecture or a practical session and is given by either a lecturer or a teaching assistant. The lecturers take place in a classrooms or lecture halls and the practical sessions take place in the labs. In summary, an event is given by a lecturer or an assistant during a time slot in a day to a specific group using a specific room resource. This relationship is represented by a timetable for all events, provided that hard constraints are not violated. Hard constraints are constraints that cannot be violated and are considered to be of great necessity to the university operation. These constraints can be also added by some lecturers like part timers or visiting professors. The timetable tries to satisfy other constraints which are not very important or critical. Such constraints are known as soft constraints that should be satisfied but maybe violated.

Lecturers can be assigned more than one courses and furthermore, some constraints can be taken into consideration to improve the quality of education e.g. No lectures should be scheduled in the last slot of any day or a certain day should have an activity slot for students, and a slot where all university academics can meet. For those slots no sessions should be scheduled

In summary the timetabling problem is too complex since a lot of constraints should be taken into account when scheduling a session. Due to the complexity of the problem, a good model is needed to solve the problem and this can be done through the help of a multi-agents that are able to compromise conflicting interests to come up with a feasible timetable. This would in turn save weeks of manual work.

1.2 PROBLEM DEFINITION

In the old days, schools taught a limited number of compulsory subjects. The number of students as well as teachers was small and the facilities within a school environment were limited. In contrast to old days, the modern world has advanced at a great pace in terms of population, facilities and technology. With the advancement of the mentioned areas, the number of students and teachers has increased, schools have expanded, more subjects are introduced and more facilities are required to accommodate the modern education.

Therefore, there is a need to come up with a method of using every teacher, class and facility effectively while maintaining the educational standards and this can only be done using timetables.

Currently, the university generates its timetables manually and this process takes a period of 3-4 weeks. This process has proven to be difficult over the years since it's generated by a group of lecturers who need to go back and forth so as to come up with a feasible timetable.

Some of the issues that arise from generating the timetable manually is that a lot of effort is used in creating new instances of a timetable whenever a conflict arises and this in turn wastes a lot of time and energy. Another issue that arises is coming up with a feasible timetable that will result in utilizing the highest productivity of teachers, optimum use of facilities and classrooms, while maintaining the required number of timeslots for the respectful subjects. Often the timetable generated, prove not to be satisfactory hence not feasible.

One of the objectives is to take advantage of the latest technological advancements to come up with a timetabling system that is able to tackle the above issues by using multi-agents that are able to compromise conflicting interests to come up with a feasible timetable in the shortest time possible.

1.3 OBJECTIVES OF THE STUDY

The main goal of the system is to

- Provide the university with a web based time tabling assistant that is able to generate a timetable from scratch given a number of constraints such as time, number of students, classrooms capacity lecturers availability etc.

In order to effectively achieve the requirements of the time tabling assistant the following objectives have to be met

- Undertaking an understudy of the university scenario so as to come up with clear timetable requirements.
- Designing a system that proves to be flexible and scalable with increased collaborated use of assignments.
- Implementing the timetabling system and ensure that its able to
    - Maximize on the schools human and technical resources
    - Reduce the number of conflicts that may arise when developing the timetable
    - Ensure fair re-assignments of resources where the demand of utilities is matched to the supply.
- Testing the functionality of the system by conducting interviews so as to get people's views and opinions about the system.
- Report and document the users' feedback.

1.4 SYSTEM JUSTIFICATION

The web enabled agent-based university timetabling assistant will largely contribute to the generation of feasible timetables by using negotiating agents that are able to compromise conflicting interests in the shortest time possible. This will in turn save weeks of manual work and effort and also improve the quality of education since the universities facilities will be fully optimized.

1.5 PROJECT SCOPE

This system will focus on the collection of data from the School of Computing and Informatics and the School of Mathematics. The information collected will then be used to form constraints for the system so as to make the system as practical as possible.

The system will have the following modules:-

- The course booking module- this module will enable lecturers to book whatever course they'd like to teach.
- The time slot booking module- this module will enable the lecturers to book time slots of their preference depending on the availability of a particular time slot.
- The negotiating module- this module enables lecturers to negotiate time slots in case they are not satisfied with the current time slot they possess.
- The timetable generation module- this module is able to generate the class timetable, the course timetable and the lecturers own schedule.
- The administrator module- this module enables an administrator to add, edit, delete and search for various components of the system. This module also supports room allocation to various student groups.
- The student module- this module enables a student to register for courses and also view their respective timetables.
- The visualization tools- this module helps visualize the availability of timeslots and also their negotiating state.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

Generating a timetable from scratch consists of scheduling a set of courses, within a given number of rooms and time slots and due to the wide number of constraints, the generic solutions for such problems are not always 100% compatible for the university. According to Abdennadher and Mohamed (2007), the number of courses, academic resources, room resources, and university specific requirements are all factors which make the problem variant and hard. Hence the attempt made by universities to generate timetables that best suits their needs is becomes difficult and complex as the number of constrains increase. It goes without say that the generation of functional timetables requires a lot of time and effort. In addition, manually generated schedules usually fail to satisfy all essential constraints which are critical for the operation of the university. Thus, automating the generation of course timetables is still a challenging task according to McCollum (2006).

2.2 *NP*-COMPLETE PROBLEMS

According to Agerbeck and Hansen (2008), Problems that can be solved by algorithms in polynomial time are considered to be easy problems. For a problem of size n the time needed to find a solution is a polynomial function of n.
Harder problems require an exponential function of n, which of course means that the execution time grows much faster than for an easy problem, when the size of the problem increases.
NP-complete problems are hard problems to solve. They belong to a class of computational problems, for which no deterministic polynomial algorithm has been found.
NP-complete problems are a subset of the class *NP* (Non-Deterministic Polynomial). A Non-deterministic algorithm is able to find a correct solution, but it is not always guaranteed. The solution is found by making a series of guesses, and the algorithm will only arrive at a correct solution, if the right guesses are made along the way. A problem is called *NP*, if its solution can be found and verified by a non-deterministic algorithm in polynomial time

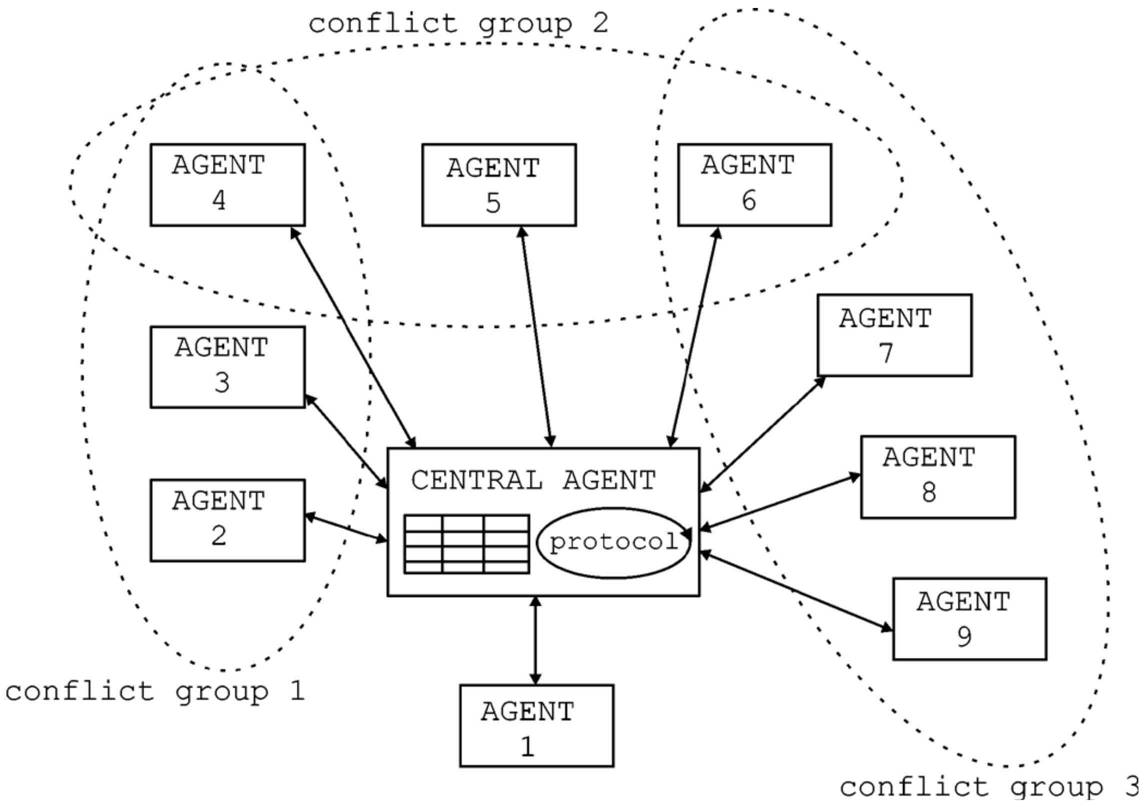
In this case, the NP complete problem is scheduling the timetable and the idea of using a multi-agent system (MAS) can be used as an intuitive approach. The system will simulate the way humans think and interact, to construct algorithms and heuristics that can solve the *NP*-complete problem i.e. timetable scheduling.

2.3 MULTI-AGENTS SYSTEMS (MAS)

According to Opreal(2006), Multi agent systems are concerned with coordinating behavior among a collection of autonomous intelligent agents that work in an environment. Sometimes, the intelligent agents are designed to reconcile their own interests with the constraints implied by other agents or they can opt to become selfish and reconcile their own interest with no regards to the other agents in the environment.

The complexity of multi-agent systems is generally higher than that corresponding to conventional software systems and their success rely on properly designed and well tested subsystems. Also, in the particular case of timetable scheduling, the MAS could find an optimal or a sub-optimal solution using mainly inter-agent communication (with minimal message passing).

The scheme of the scheduling MAS



2.4 AN AGENT SOLUTION MODEL

According to Yan, Raman and Luigi (2006), all course timetabling problems have certain common hard constraints i.e.

1) No student has more than one course at the same time;

2) Only one course is scheduled in a classroom at any one time;

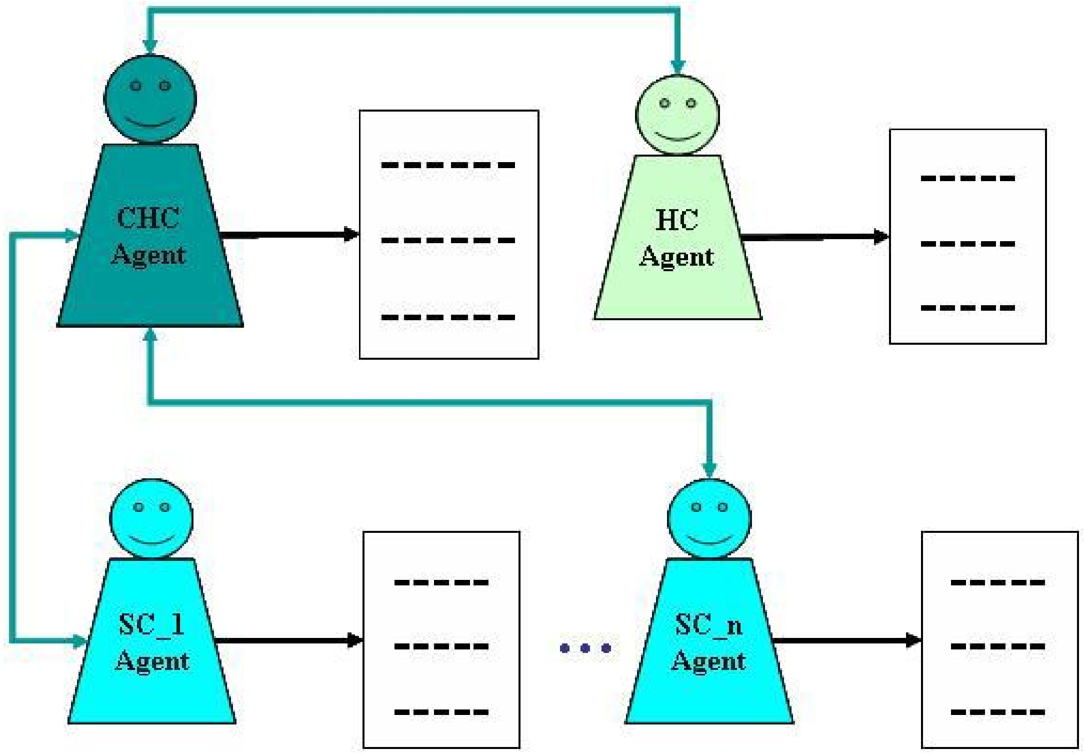
3) The classroom assigned to any course is large enough to hold all enrolled students.

By identifying the common hard constraints provides the basis to establish a general solution model for the course timetabling problem using agent technology.
A course is an event involving course name, class time, classroom and people, i.e. students, lecturers and teaching assistants. There are a set of constraints which govern the relationships between these entities.
In the proposed agent solution model, each agent represents a constraint in the course timetabling problem. The model includes the following agents:

- ✓ The **CHC** Agent which represents the **C**ommon **H**ard **C**onstraints (identified above);
- ✓ The **HC** Agents which represent other additional **H**ard **C**onstraints (such as no class scheduled during the lunch break time);
- ✓ The **SC** Agents which represent the **S**oft **C**onstraints (such as no student assigned more than two classes consecutively).

There is only one CHC Agent in the system. The number of HC and SC Agents depends on the specific course timetabling problem. The constraint agents are independent of one another. The system design is presented in three different modes of operation: the passive mode, the active mode and the parallel-active mode.

2.4.1 Passive Mode



Notepad=Course timetable

⟷ = Communication between agents

The passive mode general solution model is composed of one CHC Agent, one HC Agent and a number of SC Agents. The hard constraints can be represented by 1 HC agent or a couple just as the Soft constraints (SC) but in order to simplify the illustration 1 HC is used. Generally, there are more than one soft constraint in the course timetabling problem, so the soft constraint agents are represented by the SC_1 Agent through SC n Agent where the value of n depends on the number of soft constraints.

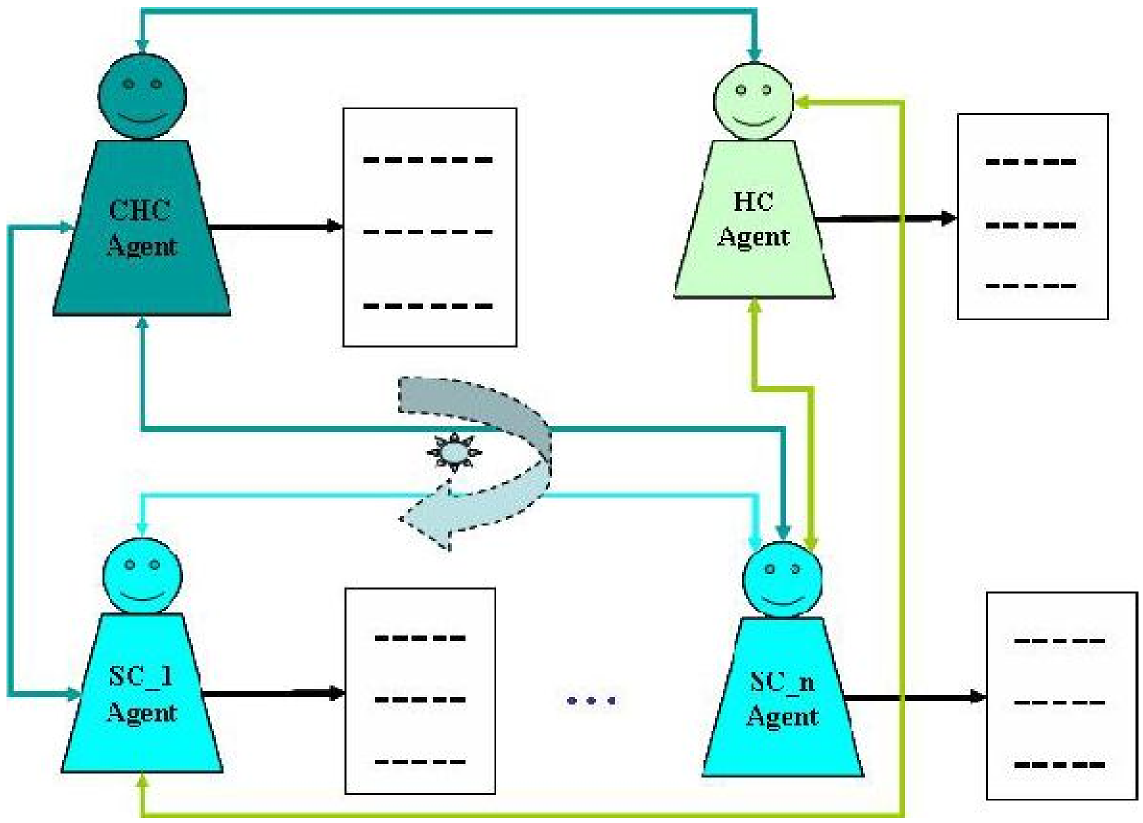
During the course timetabling process, the CHC Agent first produces a basic feasible course timetable consistent with the common hard constraints. Thereafter it sends a copy of the basic feasible timetable to each of other constraint agents. Each constraint agent evaluates the timetable based on its own constraint and sets penalty points on the number of constraint violations. These evaluations are sent back to the CHC Agent and it develops an overall evaluation result on the basic feasible timetable.

Following this, the CHC Agent creates a small disturbance to the timetable and guarantees that the disturbance does not violate the common hard constraints. The small disturbance can be realized by either moving one course to another available position in the timetable or exchanging the positions between two or three courses. As a timetable improvement proposal, the disturbance is sent to the constraint agents for evaluation. The constraint

agents evaluate the proposal and provide feedback to the CHC Agent. After developing a new overall evaluation result on the modified timetable, the CHC Agent decides if the modified timetable represents an improved schedule. If the modified timetable is accepted, the CHC Agent will adopt it as the current best timetable and inform other constraint agents to make the same change. In this way, the system can guarantee that at any moment, the course timetable carried by each agent is exactly the same. If the modified timetable is rejected, the CHC Agent will discard it. The CHC Agent then continues looking for a new proposal attempting to further improve the timetable. This process is repeated until all the constraints violations are eliminated and a final course timetable satisfying all the constraints is found. On the other hand, if the constraints violation cannot be reduced further, the course timetabling process will go to the active mode described in the next subsection.
In this timetabling mode, only CHC Agent makes proposals and all the other constraint agents passively receive and evaluate proposals. For this reason we call this the passive mode.

2.4.2 Active Mode

In the active each constraint agent has connections to all of others.
In addition, there is a token which is passed around between the constraint agents. Only the agent owning the token has the right to make proposals to try to improve the course timetable and the proposals are guaranteed not to violate its own constraint. If a proposal violates the common hard constraints, it will be rejected because all timetables must always be basic feasible course timetable, even if they do not reduce other constraint violations. Otherwise, the timetabling process is similar to that of passive mode.
In the active mode, all the agents equally contribute to improve the course timetable based on their own constraints. This allows the constraint violations to be reduced more effectively and efficiently. That is also the reason it is called the active mode. If a solution cannot be found after the token has been passed for a certain number of rounds among all the agents, the timetabling process is ended hence can’t guarantee an optimal solution satisfying all the constraints.



2.4.3 Parallel-active Mode

The active mode can be further modified to make the token unnecessary allowing all the constraint agents to work in parallel.
In this situation, the CHC Agent acts as a mediator to judge if the proposals are valid or not. Furthermore, each agent can adopt a different algorithm to create a hybrid and parallel course timetabling solution model. In such an operation mode, the system control would be much more complex, but it could be expected that the problem may be solved more quickly and more effectively.

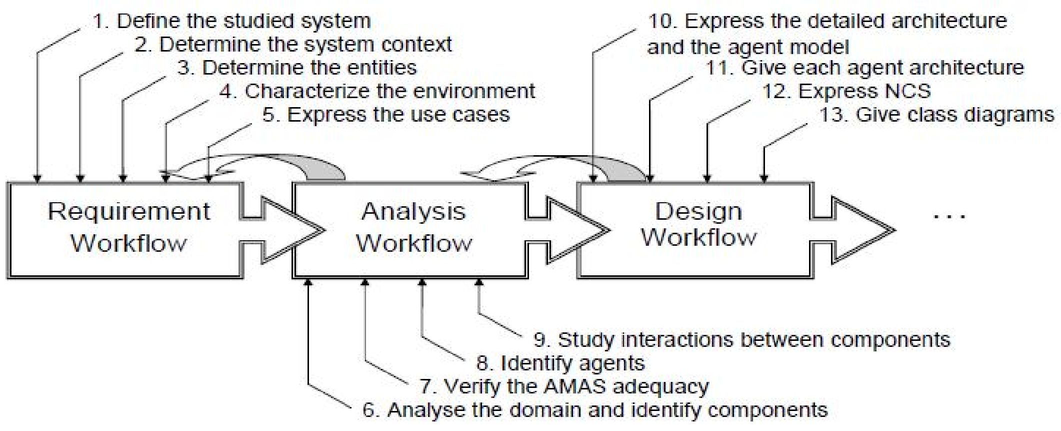
2.5 THE AMAS(*ADAPTIVE MULTI-AGENT SYSTEMS)* THEORY

The AMAS theory provides a solution to build complex systems for which classical algorithmic solutions cannot be applied. The concerned systems are open and complex. All the interactions the system may have with its environment cannot be exhaustively enumerated; also unpredictable interactions can occur during the system functioning and the system must adapt itself to these unpredictable events.
The solution provided by the AMAS theory is then to rid ourselves of the global searched goal by building artificial systems for which the observed collective activity is not described in any agent composing it .i.e. Each agent only pursues an individual objective and interacts with agents it knows by respecting cooperative techniques which lead to avoid unpredictable situations (like conflict, concurrency, etc.), called Non Cooperative Situations

(NCS). Faced with a NCS, a cooperative agent acts to reach a new cooperative state and permanently adapts itself to unpredictable situations while learning on others. Interactions between agents depend on their local view and on their ability to cooperate with each other. Changing these local interactions reorganizes the system and thus changes its global behavior.
Applying the AMAS theory consists in enumerating, according to the current problem to solve, all the cooperative failures that can appear during the system functioning and then defining the actions the system must apply to come back to a cooperative state.

2.6 ADELFE METHODOLOGY

The following are the activities that are involved when using the ADELFE methodology to develop a multi-agent system


1. Define the studied system
2. Determine the system context
3. Determine the entities
4. Characterize the environment
5. Express the use cases
10. Express the detailed architecture and the agent model
11. Give each agent architecture
12. Express NCS
13. Give class diagrams
Requirement Workflow
Analysis Workflow
Design Workflow
...
9. Study interactions between components
8. Identify agents
7. Verify the AMAS adequacy
6. Analyse the domain and identify components

Designing agents

In the ADELFE methodology, nine stereotypes have been defined to express how an agent is formed and/or how its behavior may be expressed:

«Co-operative agent»- A cooperative agent is an agent that possesses a cooperative social attitude.

As an object, a cooperative agent must have a run method which simulates its lifecycle which consists in having perceptions, taking decisions and then doing actions (perceive-decide-act).

To ensure that this method does exist, an agent inherits from a super-class called Cooperative Agent.

Rule:

- An agent-stereotyped class inherits from the Cooperative Agent class.

«Characteristic»- A characteristic is an intrinsic or physical property of an agent.

An attribute which is stereotyped with "characteristic" expresses the value of such a property. For example, the size of an agent, the number of legs of an ant.

A characteristic-stereotyped method is a means to modify or update an agent property.

For example, a method that enables to modify the number of legs of an ant.

A characteristic can be called during any phase of the lifecycle of the agent. A characteristic can be accessed or not by other agents.

«Perception» A perception is a means to receive information from the physical or social (other agents) environment.

An attribute "perception" represents a datum coming from the environment and is necessarily private.

For instance, the number of agents seen by an agent at a given time, a messages list.

A method stereotyped with "perception" is a means to modify or update attributes that are stereotyped "perception". This kind of method is not necessarily private because other agents can call it (to send a message, for example).

For example, a method that enables to put a message in an agent mailbox.

Rule:

- An attribute stereotyped with “perception " is necessarily private.

«Action» - An action is a means to act on the environment during the action phase of an agent. An attribute stereotyped with "action" stands for a parameter of an action.

For example, a move length, the maximum size for a message.

A method stereotyped with "action" represents a possible action for an agent.

For instance, move, send a message.

An agent is the only one which can use its actions, they are then private.

Rules:

- an attribute which is stereotyped with "action" is private,
- an action-stereotyped method is private,
- a method which is stereotyped using "action" can only be called during the action phase of an agent.

«Skill» -A skill-stereotyped attribute represents a datum which is useful to act on the world, or a parameter that is linked to a skill-stereotyped method. For instance, an integer value represents a minimal distance a robot has to respect to avoid obstacles.

A skill-stereotyped method represents a reasoning that the agent can only make during its decision phase. For example, a method that describes the logical reasoning to avoid obstacles.

A skill-stereotyped attribute or method can only be accessed/affected or called by the agent itself to express its autonomy of decision.

Skills can be represented by a multi-agent system when they need to evolve.

Rules:

- an attribute or a method that is stereotyped with "skill" is necessarily private,
- a skill-stereotyped attribute can only be used by a skill-stereotyped method,
- a skill-stereotyped method can only be called during the decision phase of an agent.

«Aptitude»-Aptitudes show the ability of an agent to reason both about knowledge and beliefs it owns.

An aptitude-stereotyped attribute represents a functioning datum or parameter of a reasoning made by an aptitude-stereotyped method. For example, an integer value represents the exploration depth of a planning tree.

A method stereotyped with "aptitude" expresses a reasoning which an agent is able to do using its perceptions and its world representations. This method uses one or several parameters that are attributes also stereotyped with "aptitude". For example, a method allowing actions planning or a decision mechanism.

A method or an attribute which is stereotyped with "aptitude" can only be accessed/affected or called by the agent itself, to express its autonomy of decision.

Rules:

- an attribute or a method that is stereotyped with "aptitude" is necessarily private,
- an "aptitude" attribute can only be used by a method that is also stereotyped with "aptitude",
- a method which is stereotyped with "aptitude" can only be called during the decision phase of the agent,

- an aptitude-stereotyped method can only call methods or attributes that are stereotyped with perception, representation or " interaction (for instance, for messages).

«Representation» -World representations of an agent, or beliefs, concern other agents, the physical environment or the agent itself. These representations are used by the agent to determine its behavior.

An attribute which is stereotyped with "representation" is a means to express a knowledge unit that describes an agent. For example, the number of agents known by an agent is knowledge.

A method which is stereotyped with "representation" corresponds to a handling of a representation: access, alteration... For example, a method which allows alteration of a user profile.

An attribute or a method which is stereotyped with "representation" can only be accessed/affected by the agent itself.

Rules:

- An attribute or a method which is stereotyped with "representation" is necessarily private,
- A representation-stereotyped attribute can only be used by a method which is stereotyped with "representation" or aptitude,
- A representation-stereotyped method can only be called during the decision phase of the agent.

«Interaction»-The interaction language represents tools that enable an agent to communicate directly or not with others or with its environment.

A method stereotyped with "interaction" expresses the ability an agent has to interact with other agents or with its environment in an indirect manner.

An attribute stereotyped with "interaction" represents a functioning datum or a parameter of an interaction made via an interaction-stereotyped method.

Rule:

- A method stereotyped with "interaction" can only call methods stereotyped with skill or interaction.

«Co-operation»-The social attitude of agents is implemented using rules allowing Non Cooperative Situations (NCS) solving.

An agent must have a set of rules (predicates) that allow it to detect NCS. It must also have a method to enable it to solve NCS, this method associates actions with situations in order to process them.

A method that is stereotyped with "cooperation" is always private and can be of two kinds:

- a method that returns a boolean result to detect a situation, its parameters are perceptions, representations or skills,
- a solving method (a priori, one per agent) that allows the association of one or several solving actions with each situation.

These methods are called only during the decision phase of an agent.

Rule:

- A cooperation-stereotyped method is private.

2.7 STATISTICS

The multi-agent approach to the university course timetabling problem has proven successful in offline experiments. We believe that its major advantages in comparison to the other approaches of timetabling are:

- ✓ It is aimed directly towards the resolution of the conflict spots in the timetable, assuring a relatively good efficiency.
- ✓ It allows different strategies and preferences to be employed within a well-defined negotiation framework.
- ✓ It is a natural computational model for some types of scheduling problems that can be projected onto a multi-agent system in a straightforward manner.

2.8RELATED WORK

Monash University Timetable System

Monash University Timetable System is a web based timetabling system that is able to generate timetables from scratch using different constraints such as user defined preferences. The system then outputs the generated timetable with reference to the users' preferences. The system has been used for quite some while and has significantly improved the general performance of the schools' management.Below is a link to the Monash Timetabling System.

https://mutts.monash.edu.au/mutts/timetable/bymodule.aspx?splus_year=2012

Untis Express

Untis Express is a timetabling assistant that enables one to enter basic data about the school, define exactly, which teacher is available at what time, avoid the scheduling of lessons at a particular time in is able to fact in all constraints and come up with an efficient timetable using a self-engineered algorithm. The algorithm of Untis Express does not calculate just one, but many different results then gives you the option to choose the one which you like most by defining exactly which criteria are the most important ones and which ones can be broken if necessary.

FET

FET is open source free software for automatically scheduling the timetable of a school, high-school or university. It uses a fast and efficient timetabling algorithm. The fully automatic generation algorithm, allows also semi-automatic or manual allocation

Usually, FET is able to solve a complicated timetable in maximum 5-20 minutes. For simpler timetables, it may take a shorter time, under 5 minutes (in some cases, a matter of seconds). For extremely difficult timetables, it may take a longer time, a matter of hours.

Mimosa Scheduling Software

Mimosa is a Finnish scheduling and course planning software for use in any kind of school and university of varying type and size. Besides academic timetabling, Mimosa is used to schedule conferences and in business and organisations. Mimosa has a very rich set of efficient optimisation tools and interactive timetabling selections even for the most demanding scheduling challenges. Mimosa also uses *Multi*-sensory approach *using* the Microsoft *Agent* character Peedy.

2.9 CHAPTER SUMMARY

In conclusion, the generation of a feasible timetable in any educational institutions is critical for the proper management and utilization of school resources. Therefore the automation of the timetabling process is necessary so as to ensure the generation of feasible timetables and one way of doing this is by using the agent based approach whereby a multi-agent system (MAS) is used. The use of MAS would be ideal due to the fact that the course timetabling problem is considered to be NP complete problem and the idea of using a (MAS) can be used as an intuitive approach. The MAS uses negotiating agents also known as multi agents to reach compromises between conflicting interests by using the active mode principles.

CHAPTER 3: METHODOLOGY

3.1 PLANNING AND SELECTION

The feasibility analysis carried out for the project is as discussed below.

***Economic feasibility*: In the initial stages, the system was economically viable to create for it only requires the programming tools. Once the system was created, it will be taken out for testing.

Technical feasibility: The system was deployed using a web platform which makes it easy to access and readily available to the public to use.

Operational feasibility: it will require adequate skills in web browsing so as to experience the efficiency and the effectiveness of the system.

Scope: The scope of the system will extend to data collection, analysis and representation of data via the website. The system for now will focus on two schools, i.e. School of Computing and Informatics and School of Mathematics. Maybe with time the system would be Inco-operated to the university as a whole.

3.2 DATA COLLECTION

The primary means of data collection that was used so as to acquire the users' needs was the use of questionnaires and interviews. The interviews that were carried out were used to acquire the techniques used to come up with a timetable manually. The logic behind coming up with the timetable manually would then be expressed into program logic and later be used to automate the whole timetabling system.

This information was then generalized to come up with an online system that would be able to address majority of the users' needs and generate a feasible timetables.

3.3 METHODOLOGY

The methodology that was incorporated in the system development was**the ADELFE methodology.** ADELFE is the French acronym for "toolkit to develop software with emergent functionality" (Atelier pour le DEveloppement de Logiciels à FonctionnalitéEmergente)

3.3.1*ADELFE Overview*

The ADELFE process consists of six work definitions:
Preliminary Requirements-The preliminary requirements work definition mainly deals with the description of the system and the environment in which the system will be deployed. It consists of defining the most appropriate system for end users.

Final Requirements- The aim of the final requirements is to transform this view in a use-case model, and to organize and to manage the requirements (functional or not) and their priorities.

Analysis- this involves the identification of the agents. The analysis work definition has to develop an understanding of the system, its structure in terms of components and to know whether or not the AMAS theory is required.

Design -The design work definition aims to formulate models that focus on non-functional requirements and the solution domain and that prepare for the implementation and test of the system. In ADELFE, agents being identified and their relationships being studied, designers have now to study the way in which the agents are going to interact

After design, follows the implementation of the system where the various models of the system are developed.

Finally, the system is tested in various capacities for any errors that could have been made during the implementation stage. Unit testing will be done, whereby; each component of the system is tested separately, then slowly integrated to see whether the system is able to operate as a single unit.

Reasons for choosing this methodology:

- ✓ This methodology was specifically designed foradaptive multi-agent systems based on the AMAS theory. ADELFE provides a new methodology to design a society of agents exhibiting a coherent activity.
- ✓ ADELFE has been used or is used in several case studies: an intranet system design, a timetabling problem, a flood forecast system (in progress), a mechanical design system (in progress) and a bioinformatics system (in progress) hence it passes of as a best practice.
- ✓ ADELFE is aimed to be a development toolkit of software with emerging functionalities and not only a "mere" methodology. Therefore, ADELFE will be able to provide some tools and libraries to ease the design and development of AMAS systems.
- ✓ ADELFE uses UML and AUML notations which makes it easier to understand the system and one is able to quickly identify all application deliverables therefore the system developers can significantly improve project plans and track real progress.

CHAPTER 4: ANALYSIS AND DESIGN

4.1 PRELIMINARY ANALYSIS

The means to which preliminary requirements were gathered was through interviews. The interviews were carried out with the people in charge of creating timetables in the school. Through these interviews, the key requirements of the system and also techniques that would deem key to the development of this system were attained.

The interviews were scheduled based on the interviewees' availability. A set of questions was used to guide the interview and are available in APPENDIX I.

4.1.1 Results of the interview

The following was noted after carrying out the interview

<u>The current system</u>

The university over the years has been generating timetables manually. This is done in meetings whereby lecturers within a school sit down and distribute various courses to different lecturers. Once that is done each lecturer is given a time slot in which they are comfortable teaching at that time. In case of conflicts, there is need for negotiation between lecturers. Conflicts take a substantial amount of time to resolve depending on how flexible the lecturers schedule is.

The key stakeholders for the course timetabling problem are

i. The lecturers
   Lecturers have some constraints about their availability whereby each of them have different time preferences in which the can teach a particular lecture.
   Another constraint would be the lecturers' capability to teach a certain course whereby a lecturer is assigned a course in which he/she is comfortable and passionate about teaching.
ii. The students groups
   A student group needs to take up a certain number of courses in a particular semester whereby each course is allocated a number of time slots in the timetable to complete the syllabus. (X time slots for a course 1, Y time slots for a course 2, etc.).
iii. The lecture rooms
   Lecturer rooms can either be equipped or not equipped.
   The lecture rooms that are not equipped are normally used to carry out the theoretical part of a course. The lecture rooms that are equipped have tools necessary to undertake a practical and these rooms can only be available at a given time slot or on a certain day because some of these facilities are shared among different student groups

The success of this system will depend on how well the system prioritizes and executes the constraints.

4.1.2 User requirements

The system will be deployed in two schools within the university, namely, the School of Computing and Informatics and School of Mathematics. The following will be the users of the system

- The lecturers
- The student groups
- Course manger
- Rooms manager

4.1.2.1 Functional Requirements

The following are the functional user requirements

- ✓ The lecturers will need to declare their time preferences and also state their back up options in case that particular time slot is already taken.
- ✓ The students from various student groups will need to view the end result timetable.
- ✓ The course manager will need to allocate different courses to the lecturers, depending on their capability to teach that course.
- ✓ The room manager will need to allocate classrooms to student groups depending on the capacity of that particular student group and the nature of that course (practical lesson or theoretical lesson)
- ✓ The system should provide a timetable that is able to meet the users' needs and preferences.

4.1.2.2 Non-functional requirements

Usability - The system should provide a friendly graphical interface to ensure ease of use when the end users utilize system functionality. The interface to the system should be inherently fail-safe, full-proof, and overly cautious in defending against accidental and intentional misuse.

Efficiency - The system should support a response time for addressing severe issues in less than 1 minute.

Availability - The system should be protected against both accidental and malicious denials of service, and is available for use whenever it is expected to be operational.

Confidentiality - Only legitimate officials have access to the users' confidential files and data.

Integrity - The system should be tamperproof whereby system changes are only executed by system administrators

Security - Only legitimate officials have access to the timetable database and this would be achieved by giving administrative passwords and usernames.

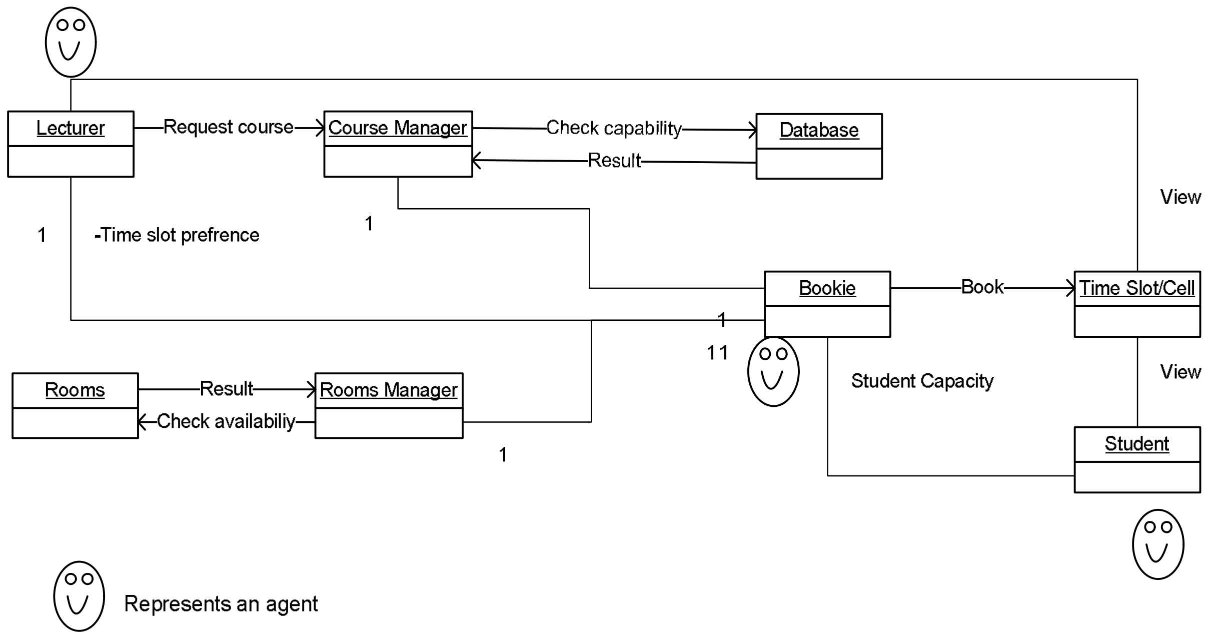
4.1.3 The basic concept of the system architecture

The system is web based hence making it readily available to its end-users via the internet. Each lecturer will need access to the system using unique usernames and passwords. Thereafter they will be assigned courses to teach using the course manager depending on their capability. The lecturers will then be able to choose a time slot that they prefer teaching based on their availability and this would be done through the help of the bookie.

The room manager will then need to allocate rooms to student groups based on the capacity of the student group.

This information will then be stored in the systems database and then used to come up with a suitable timetable that satisfies the users' needs and preferences.

Once a suitable timetable is generated, both the lecturers and students need to view it using their unique usernames and passwords to gain system access.



4.1.4 The demands imposed by the facilities and human resources

There are a number if constraints that need to be addressed in the development for this system.

- The lecturers' availability in terms of time
- The lectures' capability to teach a certain course
- The rooms' capacity
- The rooms' state (equipped/not equipped)

4.2 SYSTEM REQUIREMENTS

4.2.1 Characterization of the environment

The environment in which the system runs is said to be

- *Dynamic*: the evolution of the active entities does not depend on the system; they are unpredictable from the point of view of the system;
- *Accessible*: the environment can obtain information on the state of the environment;
- *Non-deterministic*: the system is not able to know what could be the effects of its actions on active entities;
- *Continuous*: the number of interactions between system and entities are infinite.

4.2.2System entities

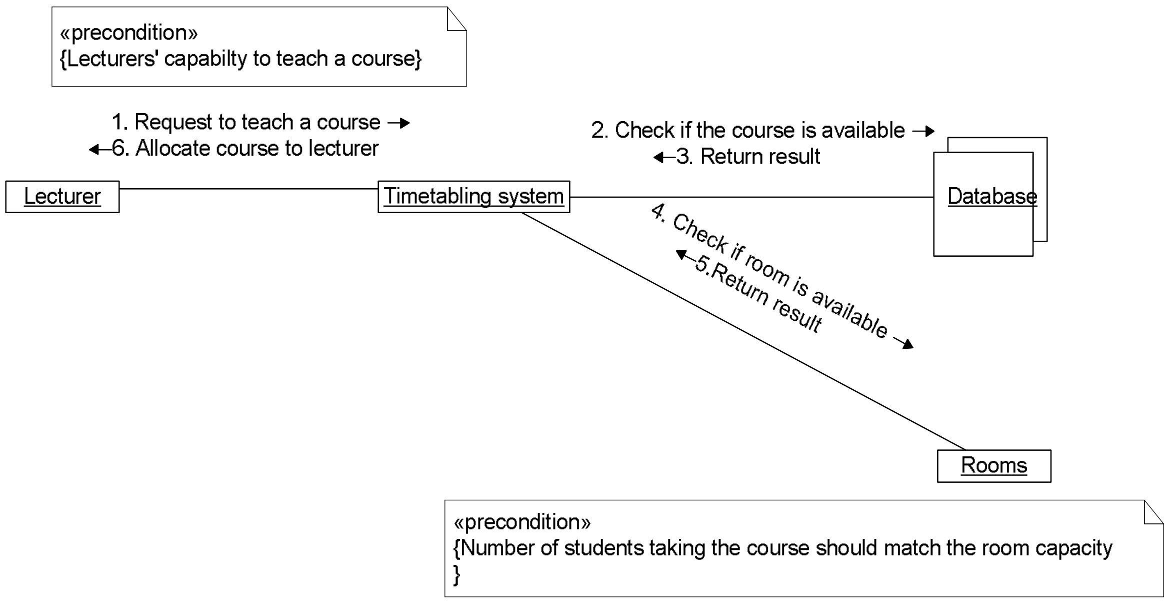
There are two types of entities within the system.

a. Active entities- they are entities that are able to change by themselves, change their own constraints or they can interact with the system. In this case they would be lecturers, student groups, the course manager and the room manager.

b. Passive entities- they are entities that represent resources and their character cannot change on their own. In this case it would be rooms and the database that contain information about courses (number of sessions per week, duration of each time slots, number of courses, earliest session, latest session, free periods etc.)

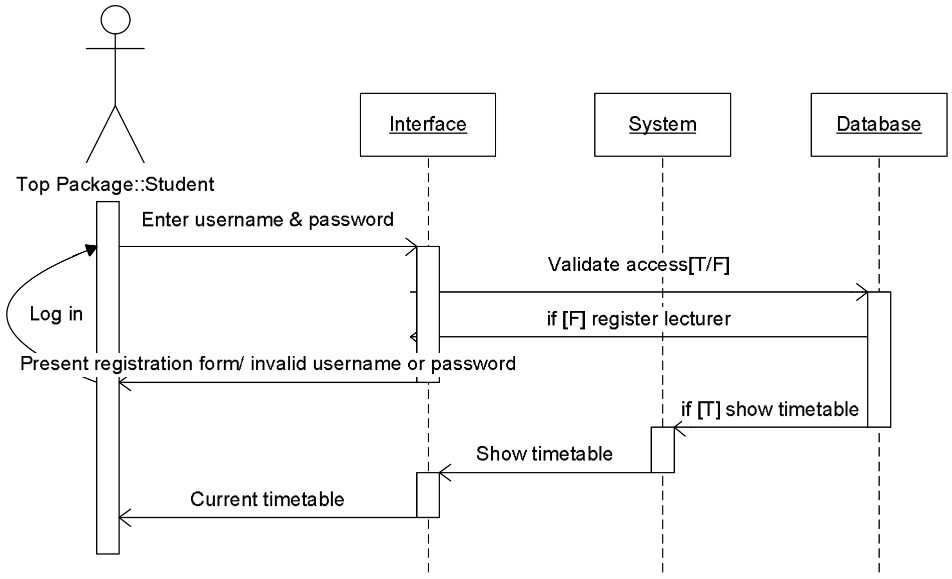
4.2.3Context definition

The collaboration diagram between the passive entities and the system

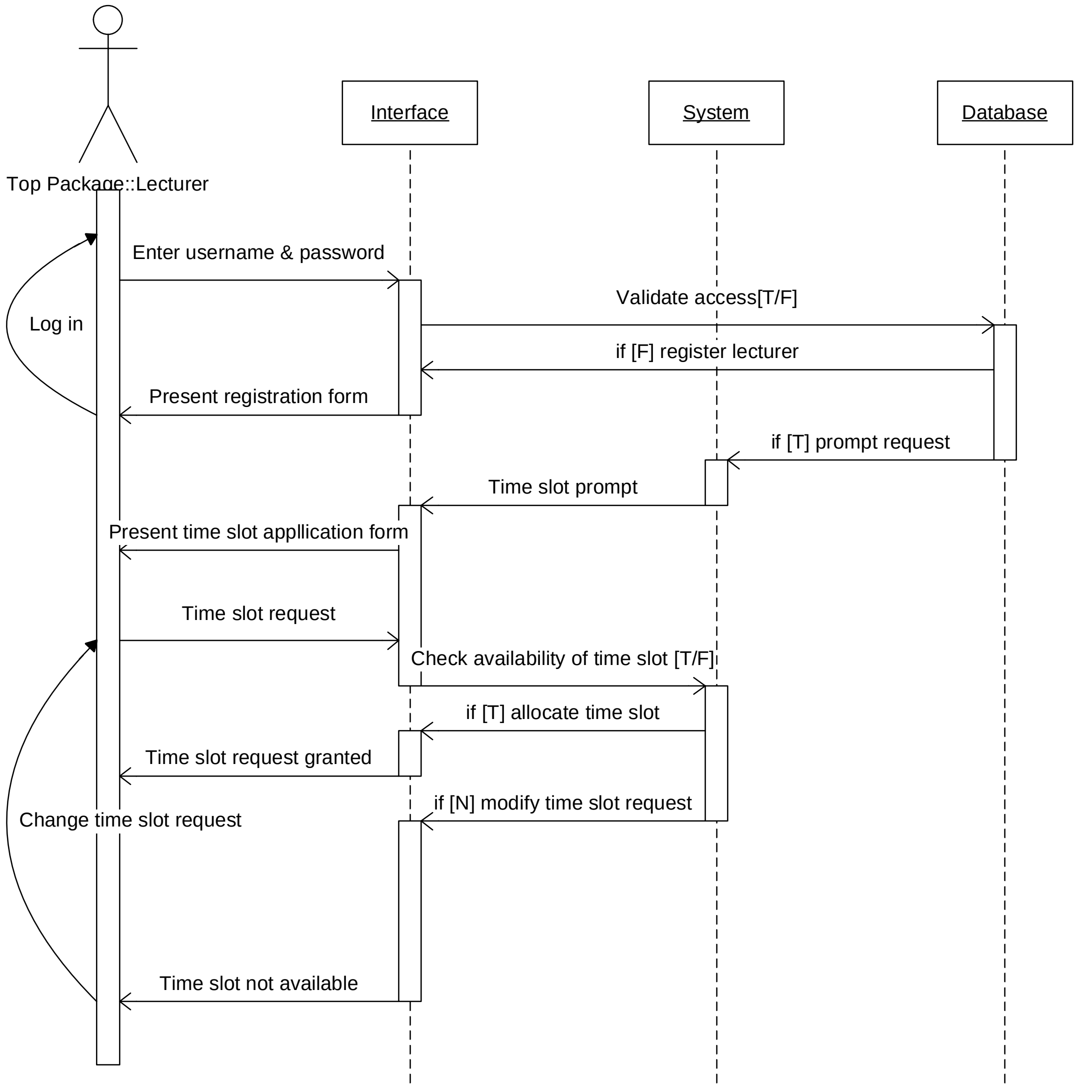

«precondition»
{Lecturers' capabilty to teach a course}
1. Request to teach a course →
← 6. Allocate course to lecturer
2. Check if the course is available →
← 3. Return result
Lecturer
Timetabling system
Database
4. Check if room is available →
← 5.Return result
Rooms
«precondition»
{Number of students taking the course should match the room capacity
}

Sequence diagram between the system and the active entities

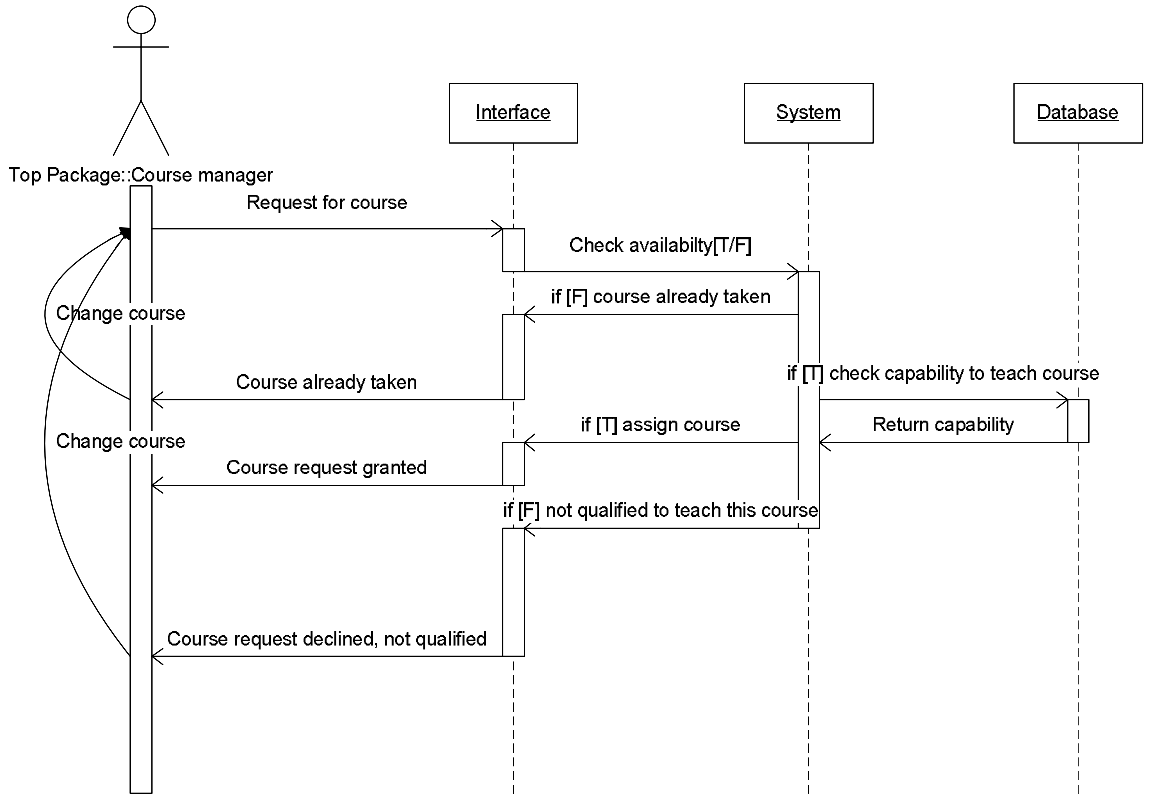
Student entity interaction with the system



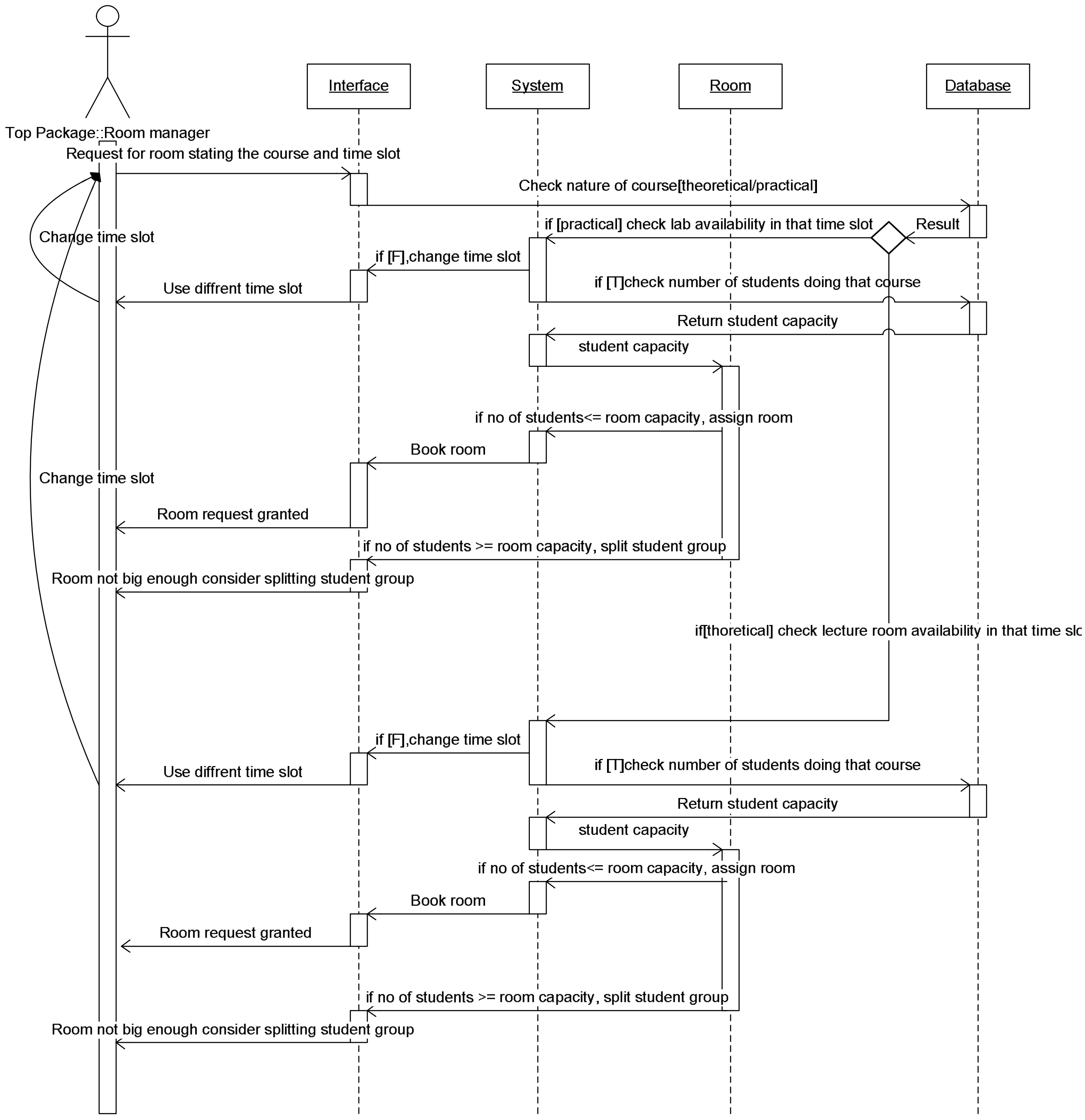
Lecturer entity interaction with the system


Interface
System
Database
Top Package::Lecturer
Enter username & password
Log in
Validate access[T/F]
if [F] register lecturer
Present registration form
if [T] prompt request
Time slot prompt
Present time slot appllication form
Time slot request
Check availability of time slot [T/F]
if [T] allocate time slot
Time slot request granted
if [N] modify time slot request
Change time slot request
Time slot not available

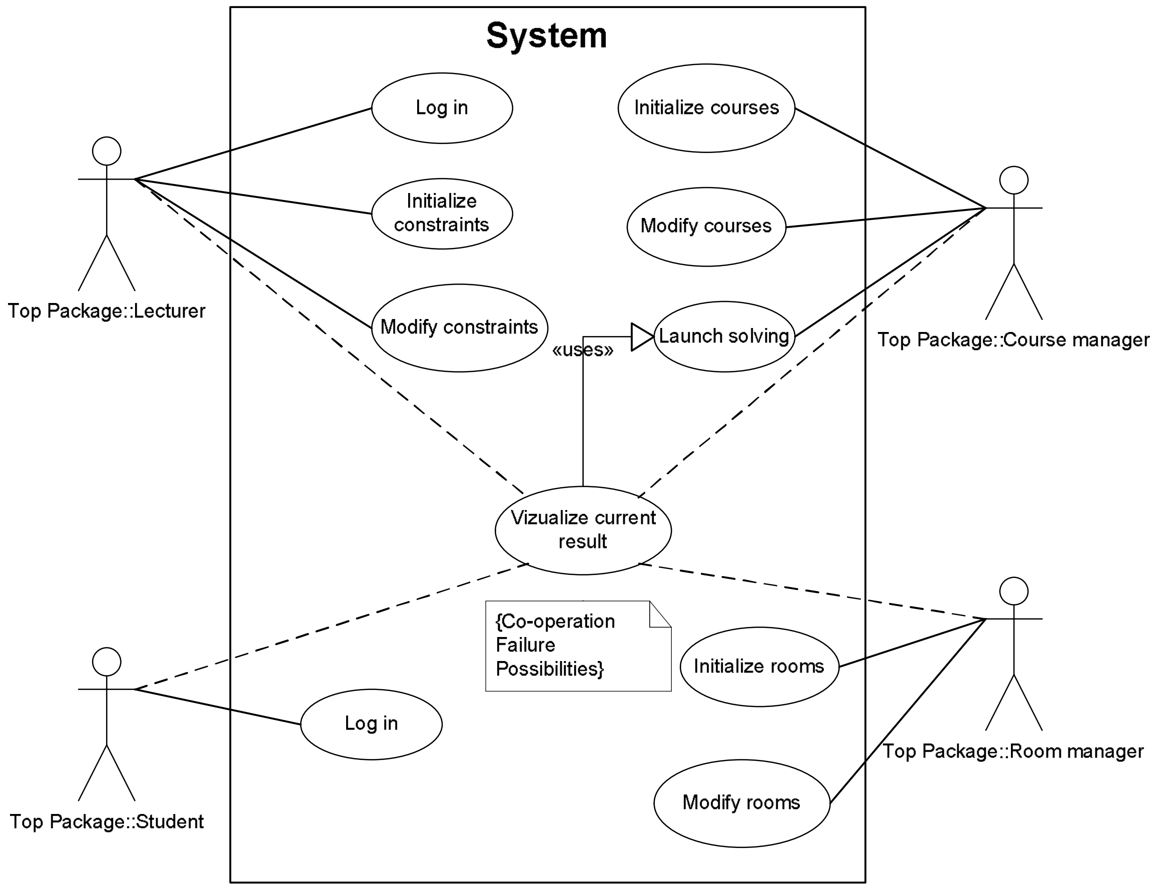
Course manager entity interaction with the system


Top Package::Course manager
Interface
System
Database
Request for course
Check availabilty[T/F]
if [F] course already taken
Change course
Course already taken
if [T] check capability to teach course
Return capability
if [T] assign course
Change course
Course request granted
if [F] not qualified to teach this course
Course request declined, not qualified

Room manager entity interaction with the system


Interface
System
Room
Database
Top Package::Room manager
Request for room stating the course and time slot
Check nature of course[theoretical/practical]
if [practical] check lab availability in that time slot
Result
Change time slot
if [F],change time slot
Use diffrent time slot
if [T]check number of students doing that course
Return student capacity
student capacity
if no of students<= room capacity, assign room
Book room
Change time slot
Room request granted
if no of students >= room capacity, split student group
Room not big enough consider splitting student group
if[thoretical] check lecture room availability in that time slc
if [F],change time slot
Use diffrent time slot
if [T]check number of students doing that course
Return student capacity
student capacity
if no of students<= room capacity, assign room
Book room
Room request granted
if no of students >= room capacity, split student group
Room not big enough consider splitting student group

4.2.4 Determination of the use cases



4.2.5 Identification of co-operation failures

The associations to '*Visualize current result case' are* potentially non cooperative: the result of the time tabling resolution is the only cause of cooperation failure between the users and the system, in the sense that users expect the system to satisfy their constraints.

Here are some of the situations that may result to non-co-operation

- The lecturer is not able to get a time slot that they desire.
- The lecturer not able to teach a course that they desire.
- Some courses not getting lecturers to teach them cause the remaining lectures are not qualified to teach them and the ones capable have taken up other courses.
- A student group not able to get a room within a given time slot.
- A room is not big enough to accommodate a certain student group.

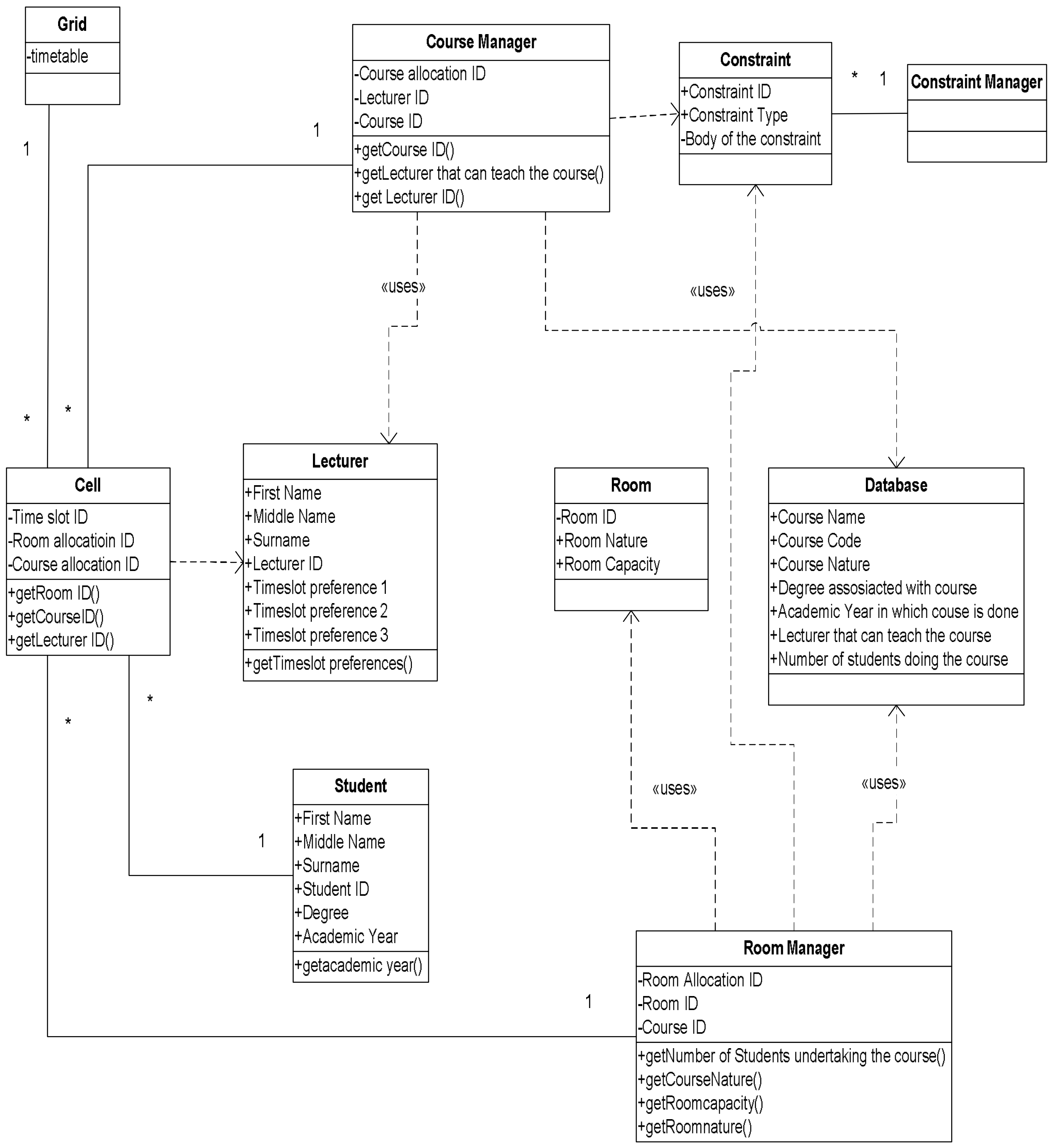
4.3 DESIGN

4.3.1 Class identification

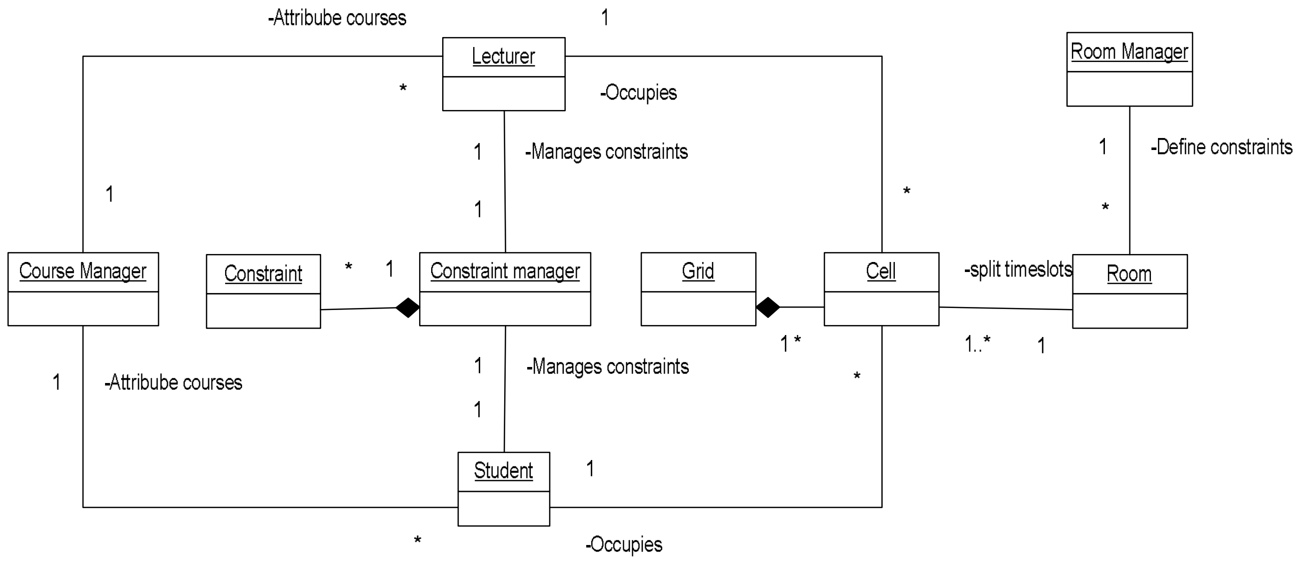
The classes that are present within the system would be

a) The lecturer class: the class represents all the lecturers within the university and their time preferences in terms of their availability.
b) The student class: the class represents all students groups within the university based on their major.
c) The room manager class: the class represents the allocation of courses to various rooms based on the room capacity and the number of students undertaking that course.
d) The course manager class: the class represents the assignment of course to lecturers based on their capability to teach.
e) The room class: this class represents the all the rooms within the university and their corresponding capacities and sole purpose (lab/ lecture room).
f) The database class: this class represents all the information one would want to know regarding a certain course.
g) The constraint class: this class represents the constraints that are declared by each entity.
h) The constraint manager class: it represents the manner to which the constraints will be handled and executed.
i) The cell class: this class represents the various intersections in terms of time slots.
j) The grid class: this class is used to visualize the current timetable.

4.3.2 The interclass relationships



4.3.3 The preliminary class diagram



4.3.4 Verification of the global level AMAS adequacy

At the global level we need to answer the question “is a system implementation using AMAS needed?”

For the case study, the decision tool clearly suggests to use the AMAS to design the global level. Moreover, the tool indicates that some entities could be decomposed as AMAS. So, once the agents are identified, there is need to reuse the method on them.

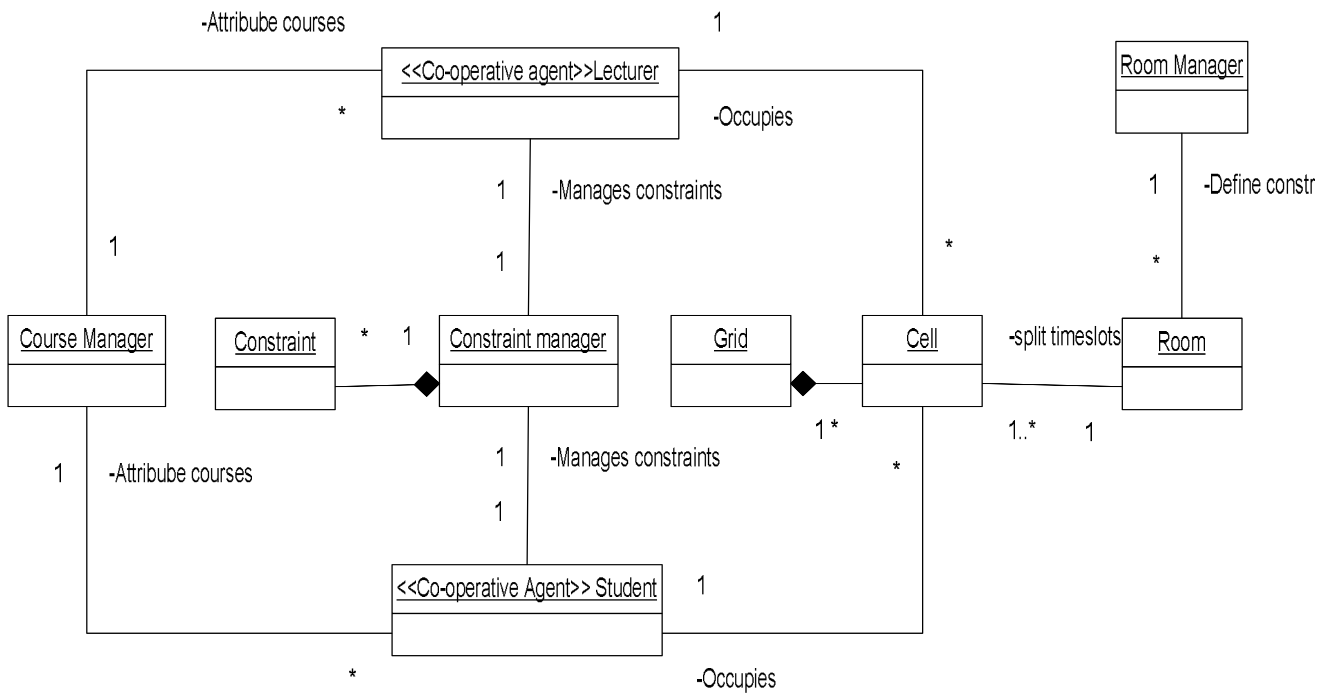
4.3.5 Identification of agents

In this phrase of the ADELFE methodology, we need to identify the co-operative agents in the system. These agents are simply entities that ignore the global function of the system such that they pursue an individual objective and try to be permanently cooperative with other agents involved in the system.

At this stage, we identify lecturers and students groups as being cooperative agents. All other entities are considered as objects.

Teachers and students are autonomous, have local views, are plunged in the rooms and have to negotiate to find partners and to resolve resource problems.

4.3.6 Agent Interaction with other entities.



Three classes of agents appear in the above diagram:

The StudentsGroup, the Lecturer and the BookingAgent. The two firsts are interface agents between the system and the users.

Booking Agents aim to reserve time slots (Cell of a Grid) and Rooms for Teachers or StudentsGroups in terms of their Constraints.

4.3.7 Verification of the local level AMAS adequacy

If the first step of adequacy to the AMAS theory indicates a possible decomposition, each agent has to be analyzed as a system. The goals of an agent, Lecturer or Student, are to find different places and partners to follow or to give each course. These goals raise the problem of ubiquity. Agents cannot be at different places at different moments. Therefore, we propose to create one agent per course for each teacher or student group. Two agent levels are distinguished:

- RepresentativeAgent (RA): at the highest level, it represents a teacher or a student group within the system;
- BookingAgent (BA): at the lowest level, it is responsible for finding partners to negotiate with in terms of time slots and booking rooms for a RA. There are as many BA as the number of courses a teacher has to give or a student group has to follow.

4.3.7 Detailed Architecture and Agent Model.

This phrase of the ADELFE involves identification of software components and their descriptions in terms of blocks, classes, agents and interactions thus making the architecture of the system. The agent model is also included in this architecture. In this case, the design pattern would be named co-operative agent architecture and it has the following four packages.

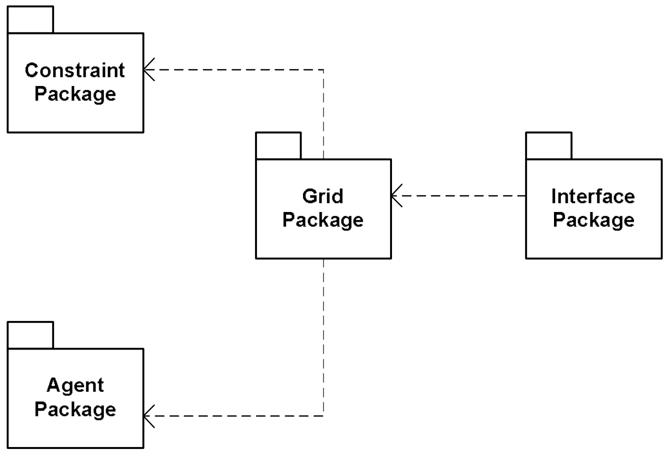
· Agent package, to manage BAs and RAs;

· Grid package, to manage the different dimensions of a grid and its cells;

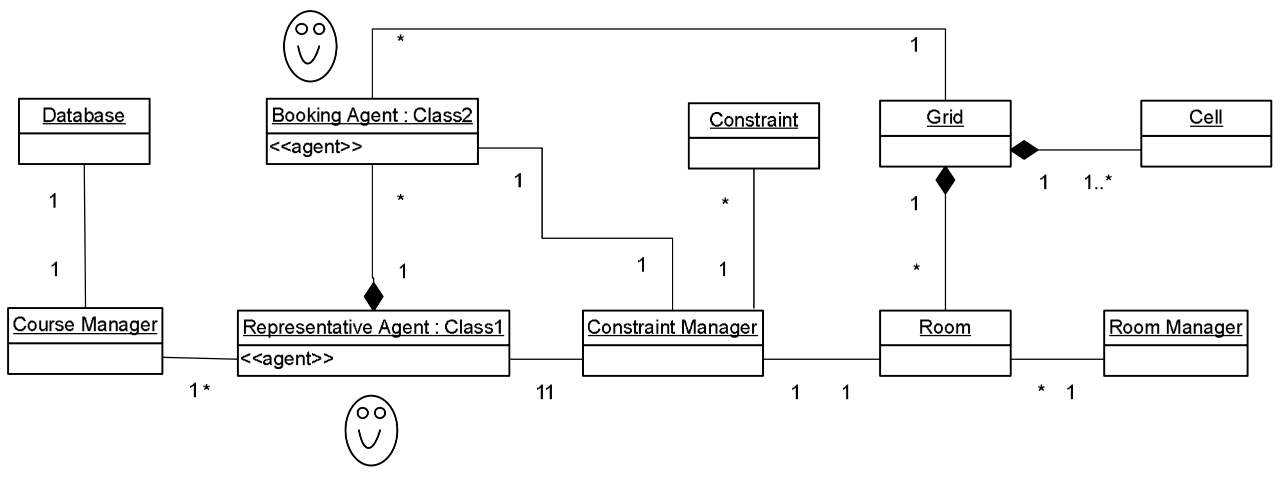
· Constraint package, which has to be accessible to rooms and agents;

· Interface package, to enable a user to interact with the system.

4.3.7.1 Package diagram



4.3.7.2 Agent Model Diagram



☺ Represents an agent

4.3.8 Agent Architecture

The BookingAgent specification:

- **Skills.**
- **Aptitudes and interaction languages are methods.**
- **Representations are attributes.**

<u>Booking Agent</u>	
Representation	
• Constraints • bookState • negotiatingState	• negotiating partners • currentCell
Skills	
• moveInGrid • manageConstraints	• manageNegotiation • manageBooking
Amplitude	
• bookATimeslot • cancelBooking • negotiateTimeslotBooking	• CancelPartnership • negotiatePartnership • establishNegotiationPartnership

Non Cooperative Situation Model.

These are the different situations an agent might come across that may lead the agent to a Non-Cooperative State and the corresponding actions they should take to come out of the NCS. In this case the NCS for a BookingAgent are:

· *Partnership incompetence*: the BA meets another BA that may be an uninteresting partner;

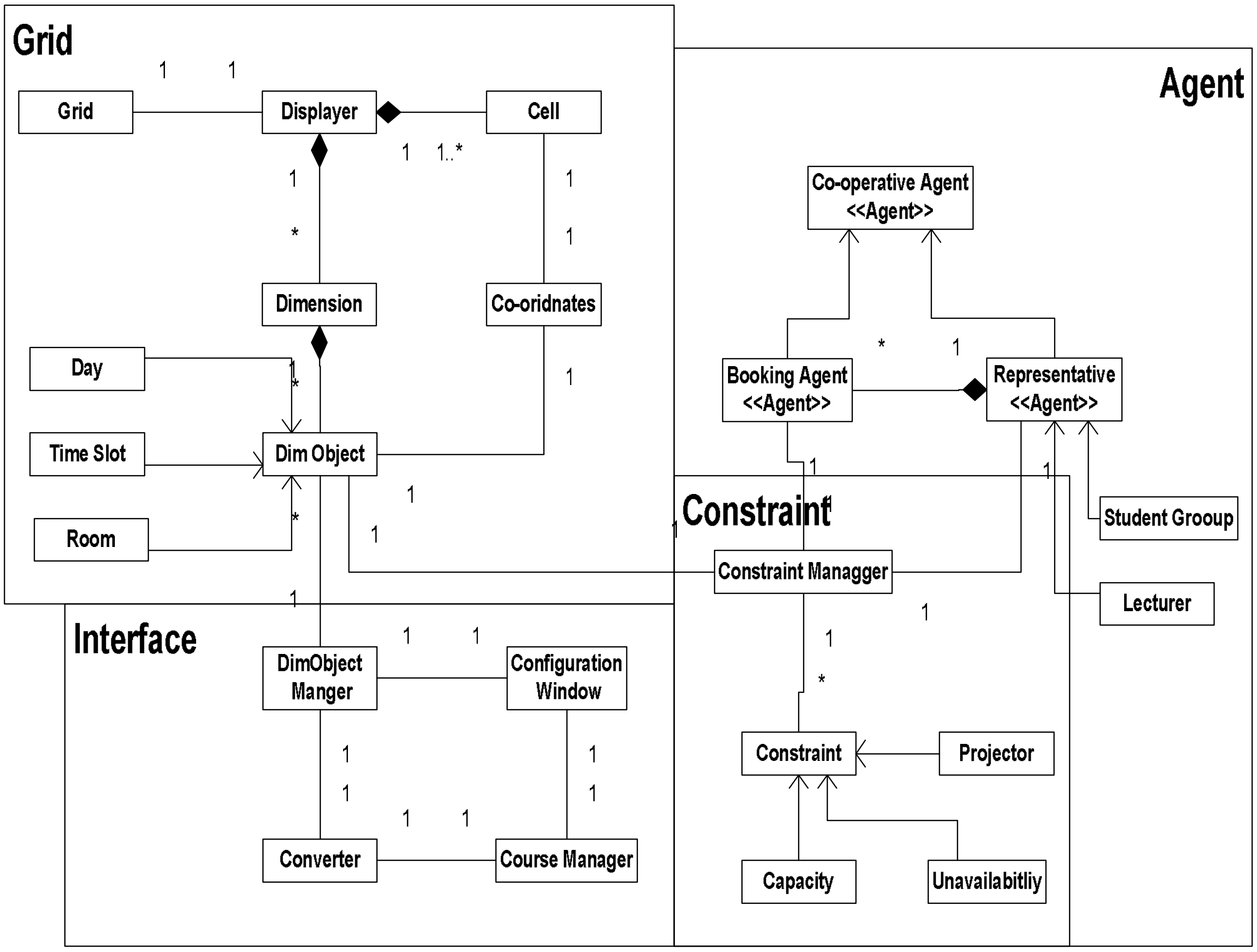
· *Booking incompetence*: the BA is in a cell that is uninteresting to book;

· *Partnership conflict*: the BA meets another BA that is interesting, but this other BA has already a partner;

· *Booking conflict*: the BA is in a cell that is interesting to book but this cell is already booked

· *Booking uselessness*: the BA meets its partner: they must separate to explore more efficiently the grid.

4.3.9 Complete design diagram



CHAPTER 5: IMPLEMNTATION AND RESULTS

5.1 Hardware Platform used

- Personal computers and laptops running windows operating system.
- Web Servers and backup servers running on Apache Processor
- Intel ® Celeron ® CPU, 2.19 GHz
- RAM: 1.87GB
- System Type: 64 bit Operating System
- 150 GB Hard Disk

5.2 Software Platform used

- Adobe Dreamweaver CS3.
- Macromedia Fireworks.
- Xampp Server
- MySql for the database.
- Language used - JavaScript, PHP has an active community support on the Web.
- Visio Professional.
- Visual Paradigm.
- Microsoft Word 2007.
- Microsoft PowerPoint 2007.
- Operating System: Windows 7.

5.3 Network Platform used

Timetabling assistant system is a web platform and thus requires the following network configuration of facilities available:

a) WAP/EDGE/3G/3.5G for mobile devices to access the site.

b) Any internet connection to access the PC website.

5.4 Choice of programming tools

Web programming technologies, tools and languages were used to implement the platform. The following gives a description and basis for selecting the web tools used in this project.

5.4.1 PHP

PHP is officially known as PHP: Hypertext Pre-processor. It is a server-side scripting language that is often written in a HTML context. PHP code that has embedded HTML fragments in it is not sent to the client but is interpreted on the server side and the resultant HTML is combined with initial HTML fragments and then sent to the client.

PHP was selected for the following reasons:

a) It is an excellent tool for server-side scripting and allows for easier automation that was traditionally handled by Perl or Shell scripting.

b) It addresses many of the web programming issues that programmers have faced while using languages such as C, and Perl. PHP has a rich library of functions to handle almost everything when developing for the web.

c) PHP performs well since a programmer need not rely on other utility libraries that often slow things down.

d) PHP also supports object-oriented programming which makes the final code easier to debug, allows code reuse during development and lends itself well to the MVC (Model-View-Controller) architecture. This shortens the development time considerably.

e) PHP is also very easy to configure on a web server and most all if not all web browsers support it. PHP is well documented and user forums are widely available on the internet.

f) PHP does not put strain on servers. It uses its own inbuilt memory space that decreases the workload from the servers and the processing speed automatically enhances. Its script is optimized to make the server's job easier, thus nowadays the uses of PHP is being popular among the programmers

5.4.2JAVASCRIPT

JavaScript is a scripting language developed to modify web pages. In this project, JavaScript was particularly used for the PC web interface to increase usability of the web based timetabling assistant system in terms of the look and feel of the web interface.

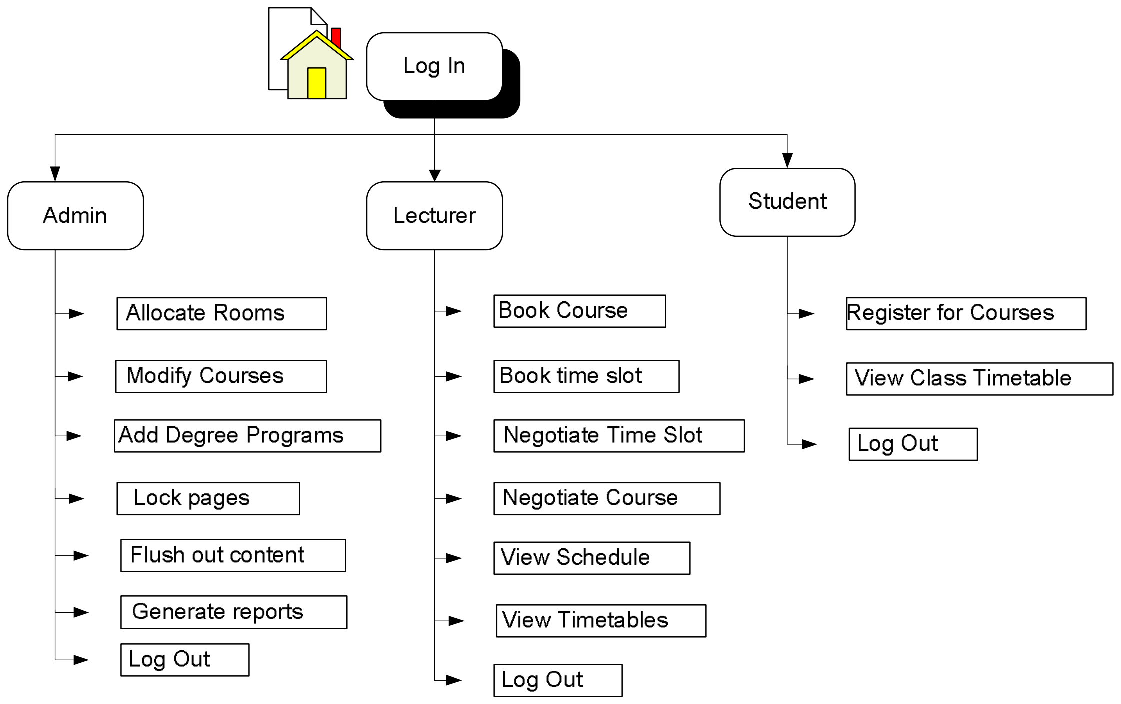
5.4.3 HTML

Hypertext Markup Language (HTML) enables you to markup text so that it can function as hypertext on the web (that is text with hyperlinks that allows for navigation). All web browsers are based on HTML.

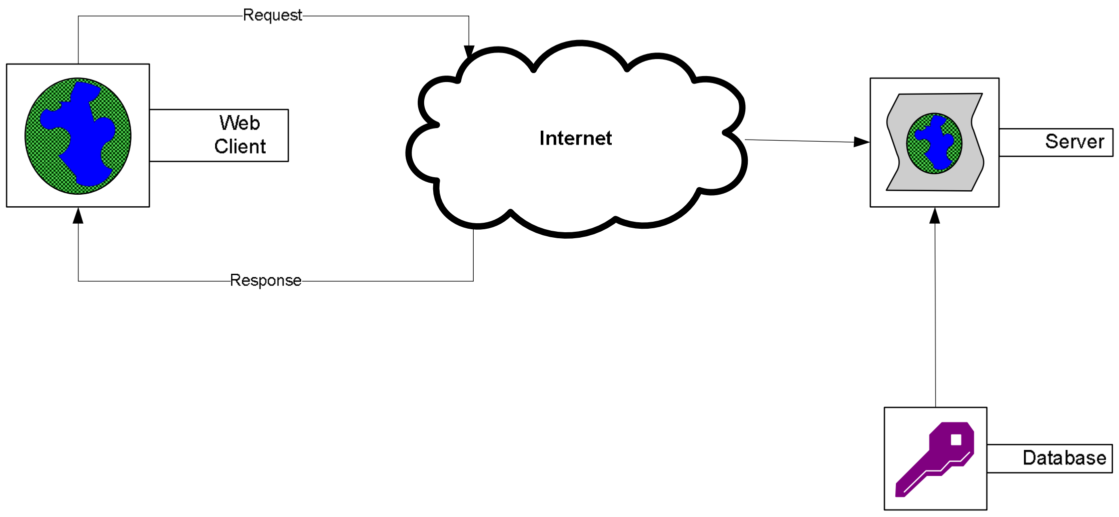
5.5 Coding

The coding effort involved a structured and modular approach in which functions and procedures needed were identified and written individually. All functions were written individually and stored in separate files. The following diagram shows a design of how the Web module was implemented.

Website Map



<u>Overall design Architecture</u>



5.6 Testing

Testing on the system was carried out using regression testing. This is done by testing each module independently and then integrating the modules to ensure that they work properly together and cohesively. The testing was iteratively, as each component was added.

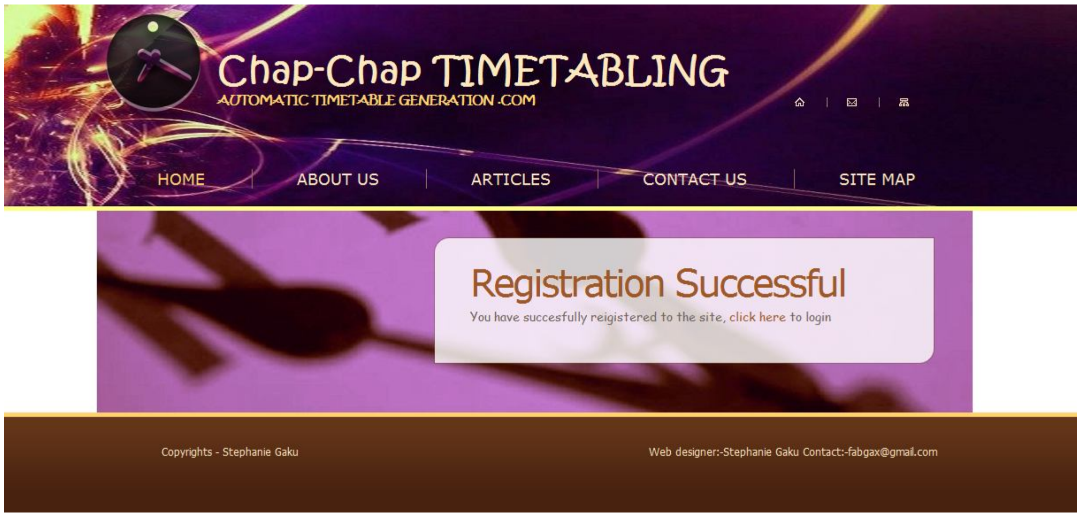
To test the web system, different values were entered to ensure that the system has been properly validated and the user cannot enter values that aren't correct. Appropriate error responses were also created so that the user can know that the transaction performed on the system wasn't successful, and whether they should try again later, or use contact the administrator for help.

Debugging of the application was done so that the system does not crash when it incurs an error.

5.6.1 Function Testing

Functions and procedures were tested one at a time. Their outputs were displayed to verify they were working in the intended manner. Examples are shown below.

This message is displayed when a user has successfully registered to the system.



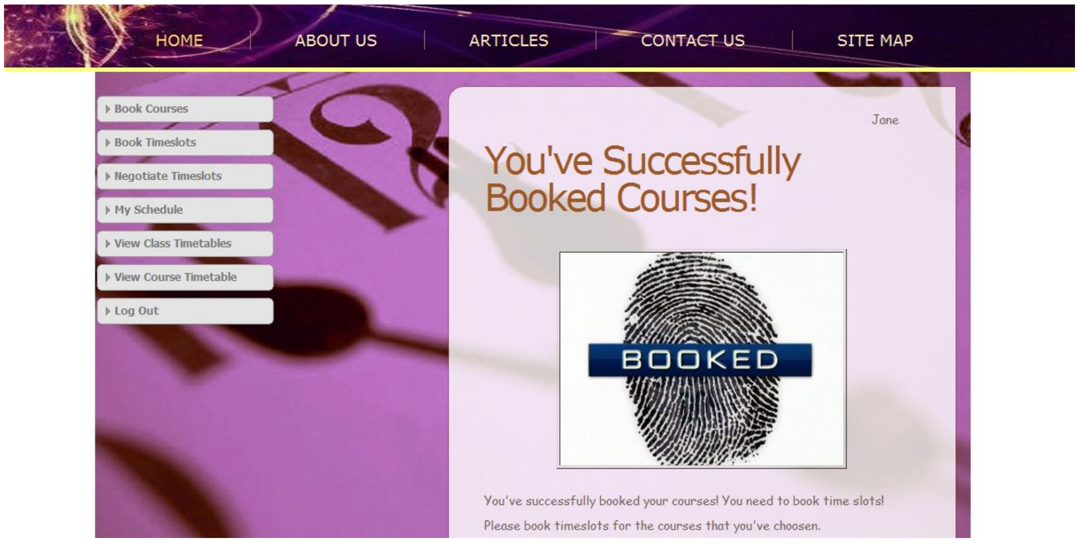
Registration Successful

You have succesfully reigistered to the site, click here to login

Copyrights - Stephanie Gaku

Web designer:-Stephanie Gaku Contact:-fabgax@gmail.com

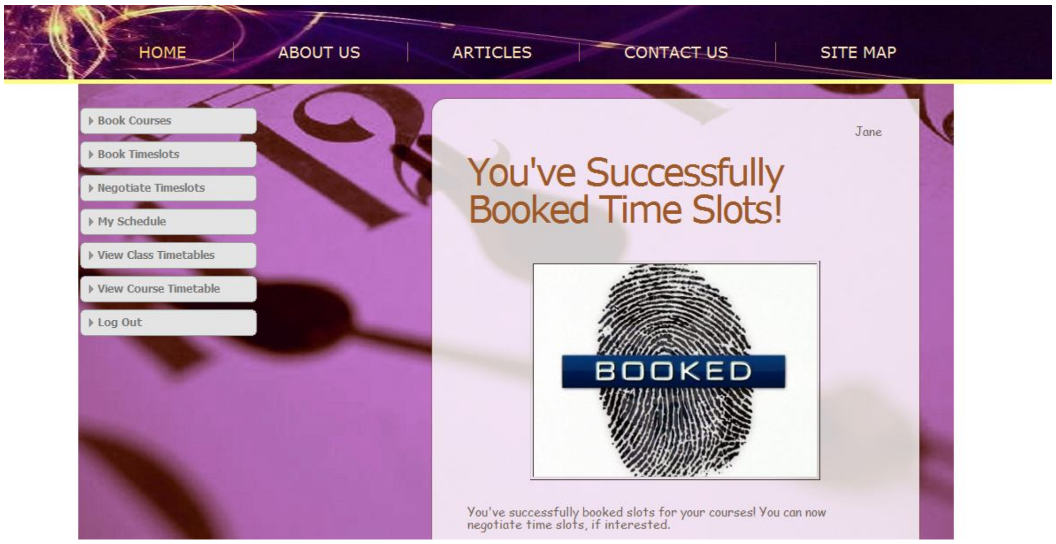
This message is displayed when a lecturer has successfully booked courses of their preference and based on their availability.



You've Successfully Booked Courses!

You've successfully booked your courses! You need to book time slots! Please book timeslots for the courses that you've choosen.

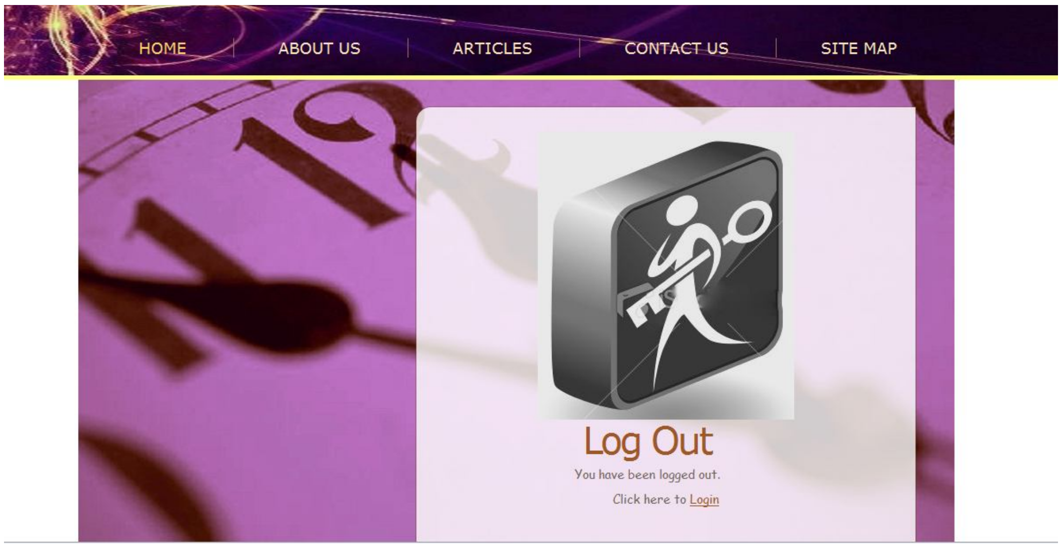
The following message is displayed when a lecturer has successfully booked timeslots for their respective courses.



You've Successfully Booked Time Slots!

You've successfully booked slots for your courses! You can now negotiate time slots, if interested.

The following message is displayed when a system user has successfully logged out of the system.



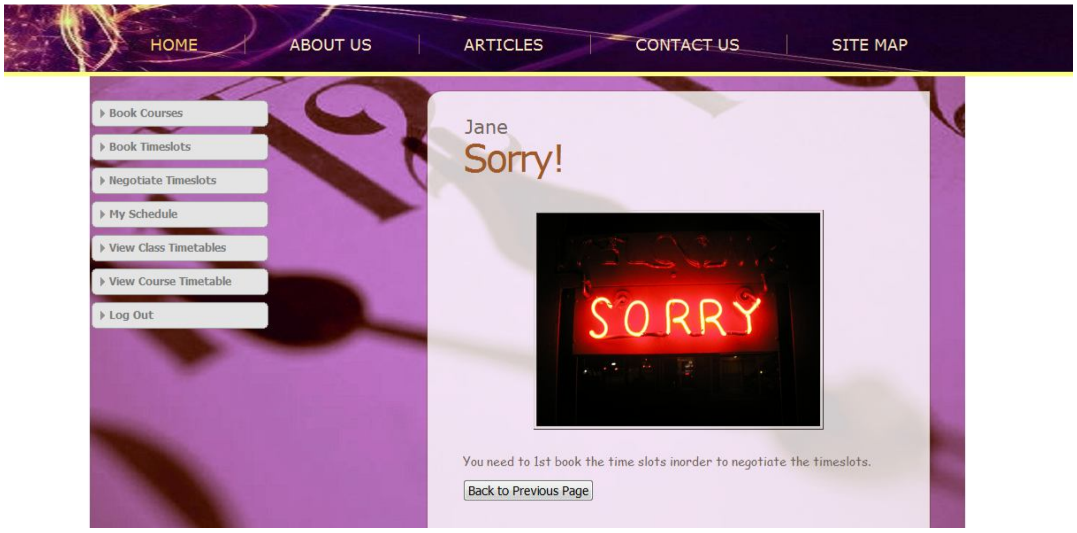
Log Out

You have been logged out.

Click here to Login

5.6.2 Integration Testing

Results of functions were passed to each other where applicable. For example, in order for you to negotiate for a time slot you first need to book a timeslot. This is because the lecturer user needs to trade in slots so as to participate in the negotiation process. If a user tries to negotiate for timeslots before booking the time slots, an error message is displayed, instructing the user on what to do. Below is an illustration of this function.

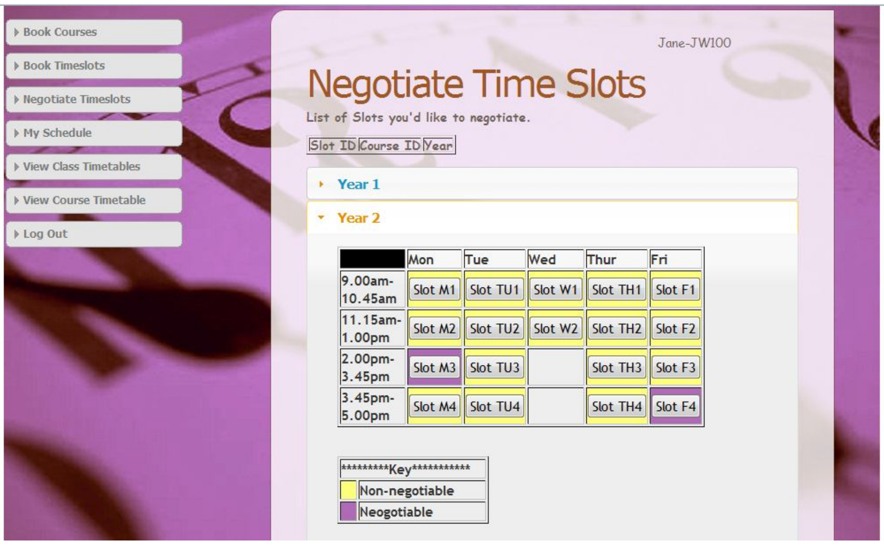


Jane

Sorry!

You need to 1st book the time slots inorder to negotiate the timeslots.

If the user books the slots and later requests to negotiate the time slots, permission is granted to them. Below is an illustration of this process.



Jane-JW100

Negotiate Time Slots

List of Slots you'd like to negotiate.

Slot ID	Course ID	Year

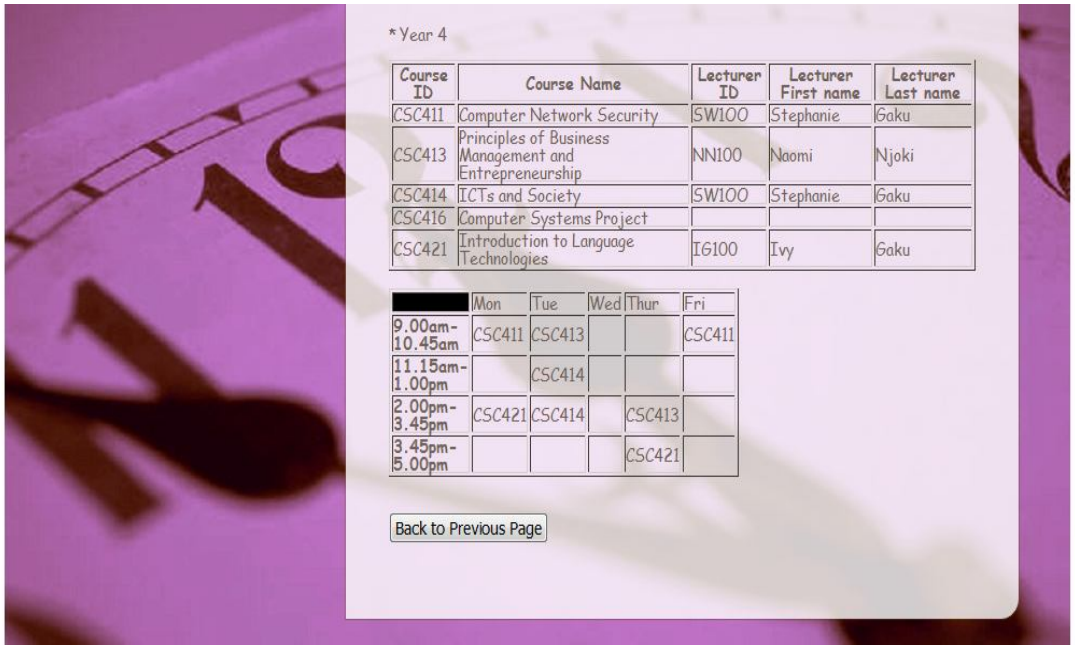
	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am	Slot M1	Slot TU1	Slot W1	Slot TH1	Slot F1
11.15am-1.00pm	Slot M2	Slot TU2	Slot W2	Slot TH2	Slot F2
2.00pm-3.45pm	Slot M3	Slot TU3		Slot TH3	Slot F3
3.45pm-5.00pm	Slot M4	Slot TU4		Slot TH4	Slot F4

*********Key***********	
🟨	Non-negotiable
🟪	Neogotiable

5.6.3 System Testing

The overall function of the system was examined and tested. The results were examined to determine if they were correct.

The correctness of the system was tested by comparing the end result timetables with previously manually generated. Based on this comparison, it was concluded that the automated system saves a lot of time and energy spent on the generation of timetables manually. The system was also found to be flexible for it has a negotiation platform.Below are some examples of the automatically generated timetables based on the environment constraints.



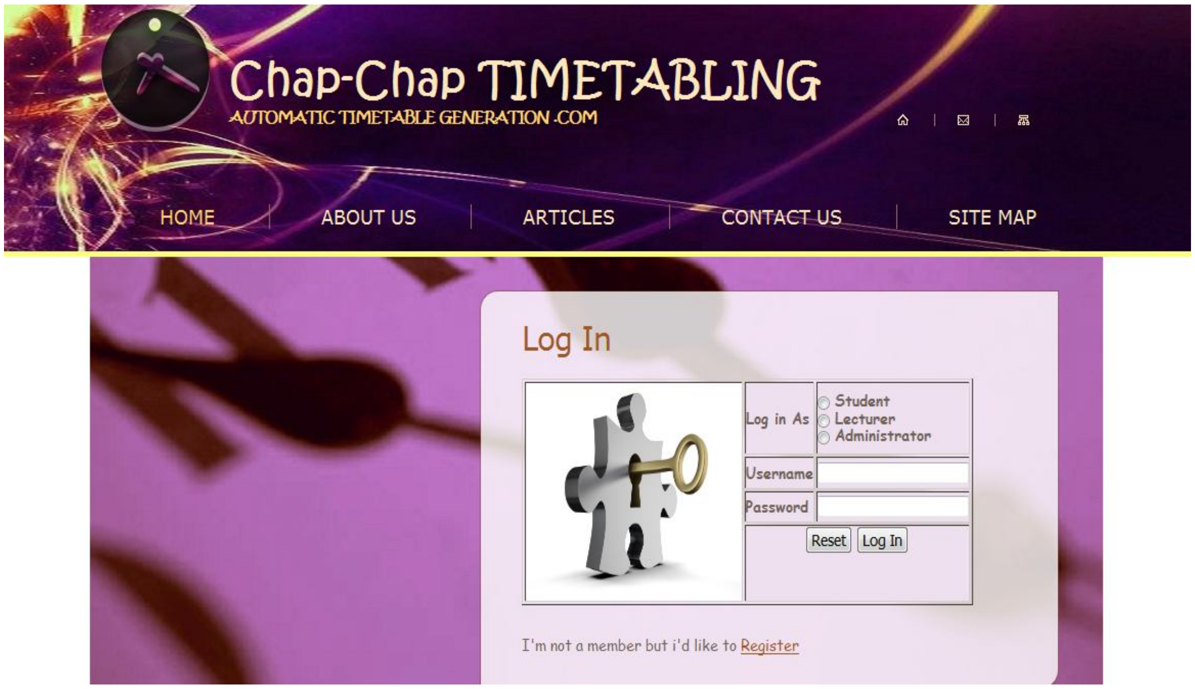
*Year 4

Course ID	Course Name	Lecturer ID	Lecturer First name	Lecturer Last name
CSC411	Computer Network Security	SW100	Stephanie	Gaku
CSC413	Principles of Business Management and Entrepreneurship	NN100	Naomi	Njoki
CSC414	ICTs and Society	SW100	Stephanie	Gaku
CSC416	Computer Systems Project			
CSC421	Introduction to Language Technologies	IG100	Ivy	Gaku

	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am	CSC411	CSC413			CSC411
11.15am-1.00pm		CSC414			
2.00pm-3.45pm	CSC421	CSC414		CSC413	
3.45pm-5.00pm				CSC421	

5.7 Security

All users who access this system must be authenticated first. The authentication method used in this system is the use of a username and a password. A log in page was created for registered users i.e. lecturers, students and administrators to access the system. A sample of this is shown.



CHAPTER 6: DISCUSSION

6.1 Achievements

Several achievements were made as a result of undertaking this project. These include:

- Development of a system that will help in the automatic generation of feasible timetables for the university
- Appreciation of web based applications solutions
- A better understanding of software development techniques and design and a wider knowledge of computer technologies
- Development of a system that has a significant impact on real world application
- A basis for further software development work

6.2 Constraints

The main challenges encountered were:

- Limited Information about multi- agent systems.
- Limited examples of multi-agent systems.
- Program incompatibility with agent based programs in terms of web hosting.

6.3 Recommendations

The following are additions that can be made to this project:

- Development of the system as a mobile application.
- Incorporating the system into the university as a whole so as to incorporate all the schools within the university.
- Send alerts via sms to users when any changes are made on the timetable.
- Allow prioritization of users, such that high level users are given first priority in booking courses and timeslots.

6.4 Suggestions for Further Work

One way of improving the web application, is to incorporate an emailing trigger whereby the timetables could be emailed to the users with respect to their associatively. This would ensure that every user gets a copy of the timetable.

Another way of improving the system is by creating another negotiating platform for courses. This would allow lecturers to exchange courses, if the choice that they got was not satisfactory for them.

Another way of improving the system is to incorporate a prioritization mechanism whereby high level users are given first priority to access and use the system.

6.5 Conclusion

Using the web application developed, the school is able to generate timetables with the greatest of ease. The automation of this system has greatly reduced cases of conflict and is able to create timetables from scratch according to user preference and provides a negotiating platform in case a user is not satisfied with the time slot presented to them.

For the potential users of the system, providing access to such a facility saves a lot of time spent in the generation of timetables. Also, the system is able to come up with feasible timetables on time using the least amount of energy whereas significantly reducing the number of conflicts.

Implementing such a facility in the university will prove to be an added advantage for it will save up a lot of time and significantly cut down on any expenses used during the generation of timetables. Since the system is web based, it will be accessible to everyone and also the expenses on web-hosting are significantly low hence more reason to implement the system.

APPENDIX I

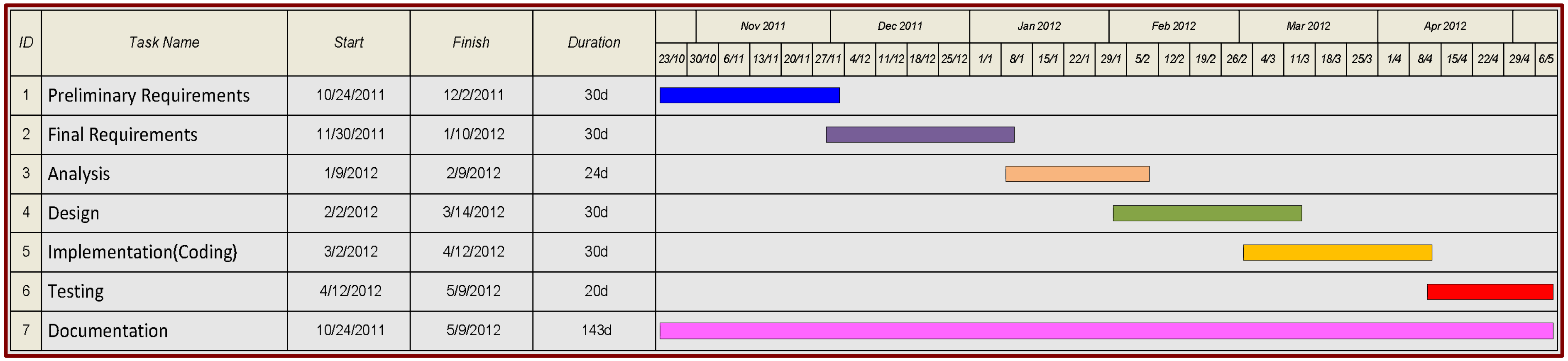
Automatic Timetabling System Interview Questions

1. How does the university currently handle the generation of timetables?(Manual/Semi - automatic / automatic)
2. How long has the university been using this method to generate timetables?
3. How long does it take to come up with a timetable manually?
4. What are the things you put into consideration when coming up with a time table?
5. What would be the hard constraints when coming up with timetables?
6. How exactly do you come up with a timetable? The procedures and steps you follow so as to come up with a timetable?
7. What are the problems that you encounter when coming up with a timetables?
8. What recommendations would you have for developing such a system?

APPENDIX II

PROJECT PLANNING AND RESOURCES

PROJECT PLAN



ID	Task Name	Start	Finish	Duration
1	Preliminary Requirements	10/24/2011	12/2/2011	30d
2	Final Requirements	11/30/2011	1/10/2012	30d
3	Analysis	1/9/2012	2/9/2012	24d
4	Design	2/2/2012	3/14/2012	30d
5	Implementation(Coding)	3/2/2012	4/12/2012	30d
6	Testing	4/12/2012	5/9/2012	20d
7	Documentation	10/24/2011	5/9/2012	143d

PROJECT RESULTS AND PRODUCTS

- An online system that will be able to generate feasible timetables with respect to various constraints
- A multi-agent system for scheduling sessions
- A user friendly interface
- A user guide on how to operate the website.

RESOURCES REQUIRED

Expendable resources: Printing paper

- Hardware
- Internet access
- Software for web hosting such as apache web server and dream weaver for developing the website.
- Java Script and PHP to manage the constraints.
- Informants for the university timetabling process
- Previously published research
- Previous timetables that were manually schedules

BUDGET

	ITEM	QUANTITY	PRICE in KSHS	TOTAL
HARDWARE	PC/ LAPTOP	1	35,000	35,000.00
	Patch cable	1	500	500.00
	Flash disk	1	1,500	1,500.00
	Printer cartridge	2	3,000	6,000.00
	Rim of paper	1	525	525.00
Software	Ms office(Students version)	Whole suite	1,000	1,000.00
	Antivirus	1 copy	1,200	1,200.00
Travel cost	Within Nairobi	N/A	50 per km	N/A
Communication	Local calls	N/A	4 per Min	N/A
Training	Users	N/A	0(Free)	0
Miscellaneous		N/A	10% of total	4,572.50
Total				50,297.50

CONCLUSION

This project will be able to demonstrate how the course timetabling problem at the university can be modeled as a constraint satisfaction problem. The use of agents suggests that the system is highly flexible and can be easily maintained due to the highly adaptive nature of multi-agent systems MAS. These qualities prove to be which is crucial for any university.

APPENDIX III

Data Source Sample

Log In Execution Code

```
<?php
	//Start session
	session_start();

	//Include database connection details
	require_once('config.php');

	//Array to store validation errors
	$errmsg_arr = array();

	//Validation error flag
	$errflag = false;

	//Connect to mysql server
	$link = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD);
	if(!$link) {
		die('Failed to connect to server: ' . mysql_error());
	}

	//Select database
	$db = mysql_select_db(DB_DATABASE);
```

```
if(!$db) {
	die("Unable to select database");
}

//Function to sanitize values received from the form. Prevents SQL injection
function clean($str) {
	$str = @trim($str);
	if(get_magic_quotes_gpc()) {
		$str = stripslashes($str);
	}
	returnmysql_real_escape_string($str);
}

//Sanitize the POST values
$mode = clean($_POST['login']);
$username = clean($_POST['username']);
$password = clean($_POST['password']);

//Input Validations
if($mode == '') {
	$errmsg_arr[] = 'Please choose who your logging in as';
	$errflag = true;
}
if($username == '') {
	$errmsg_arr[] = 'Login ID missing';
	$errflag = true;
}
```

```
	if($password == '') {
		$errmsg_arr[] = 'Password missing';
		$errflag = true;
	}

	//If there are input validations, redirect back to the login form

	if($errflag) {
		$_SESSION['ERRMSG_ARR'] = $errmsg_arr;
		session_write_close();
		header("location: login.php");
		exit();
	}
	//Check whether the query was successful or not
	if($mode == 'student'){
	//Create query
	$qry="SELECT * FROM studusers WHERE username='$username'AND
password='".md5($_POST['password'])."'";
	$result=mysql_query($qry);

	if($result) {
		if(mysql_num_rows($result) == 1) {
			//Login Successful
			session_regenerate_id();
			$member = mysql_fetch_assoc($result);
			$_SESSION['SESS_MEMBER_ID'] = $member['studmemID'];
			$_SESSION['SESS_FIRST_NAME'] = $member['fn'];
```

```
                    $_SESSION['SESS_LAST_NAME'] = $member['ln'];

                    $_SESSION['studid'] = $member['studid'];


                    session_write_close();

                    header("location: member-index1.php");

                    exit();

          }else {

                    //Login failed

                    header("location: login-failed.php");

                    exit();

          }

     }else {

          die("Query failed");

     }

     }


     if($mode == 'lecturer'){

     //Create query

     $qry="SELECT * FROM lectusers WHERE username='$username'AND
password='".md5($_POST['password'])."'";

     $result=mysql_query($qry);


     if($result) {

          if(mysql_num_rows($result) == 1) {

                    //Login Successful

                    session_regenerate_id();

                    $member = mysql_fetch_assoc($result);
```

```
                $_SESSION['SESS_MEMBER_ID'] = $member['lectmemID'];

                $_SESSION['SESS_FIRST_NAME'] = $member['fn'];

                $_SESSION['lectid'] = $member['lectid'];

                session_write_close();

                header("location: member-index2.php");

                exit();

            }else {

                //Login failed

                header("location: login-failed.php");

                exit();

            }

        }else {

            die("Query failed");

        }

        }


        if($mode == 'admin'){

        //Create query

        $qry="SELECT * FROM admin WHERE username='$username'AND
password='".md5($_POST['password'])."'";

        $result=mysql_query($qry);


        if($result) {

            if(mysql_num_rows($result) == 1) {

                //Login Successful

                session_regenerate_id();

                $member = mysql_fetch_assoc($result);
```

```
                        $_SESSION['SESS_MEMBER_ID'] = $member['adminId'];

                        $_SESSION['SESS_FIRST_NAME'] = $member['username'];

                        session_write_close();

                        header("location: member-index3.php");

                        exit();

                }else {

                        //Login failed

                        header("location: login-failed.php");

                        exit();

                }

        }else {

                die("Query failed");

        }

        }

?>
```

User Manual

This section gives a description of the procedures to be followed when using the Chap-Chap Timetabling System.

One needs to have installed Xampp server on their computer.

To launch the web-based application, first start the Xampp server. You may find the start-up icon on your desktop, or you can go to 'Programs' and start it up from there.

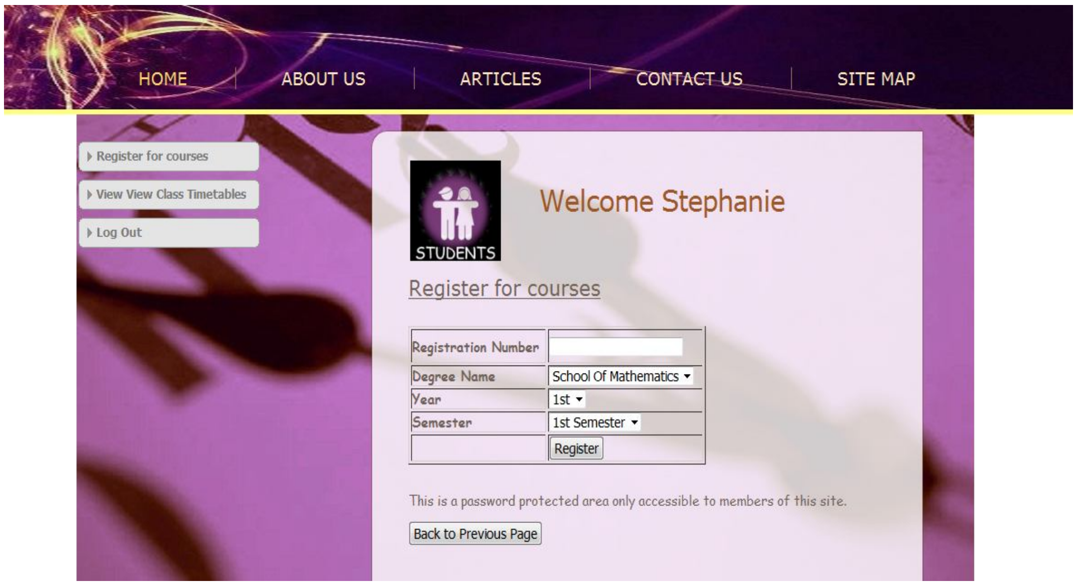
Open the browser of your choice and specify the local host address which directs you to where your application is stored. A user has to present his/her identification details together with a password in order to be authorized to access and utilize the timetabling system.

There are various links on the home page that allow a user to perform various tasks. Such tasks and access to information depends on the administration rights a user has. Information is limited to ordinary system users, while administrative user has rights to access all the information in the system.

There are three different user levels. The student, lecturer and administrator user.

The student user login, leads to a group of pages that allow the user to register and view the timetables.

<u>Registration of courses</u>



Welcome Stephanie

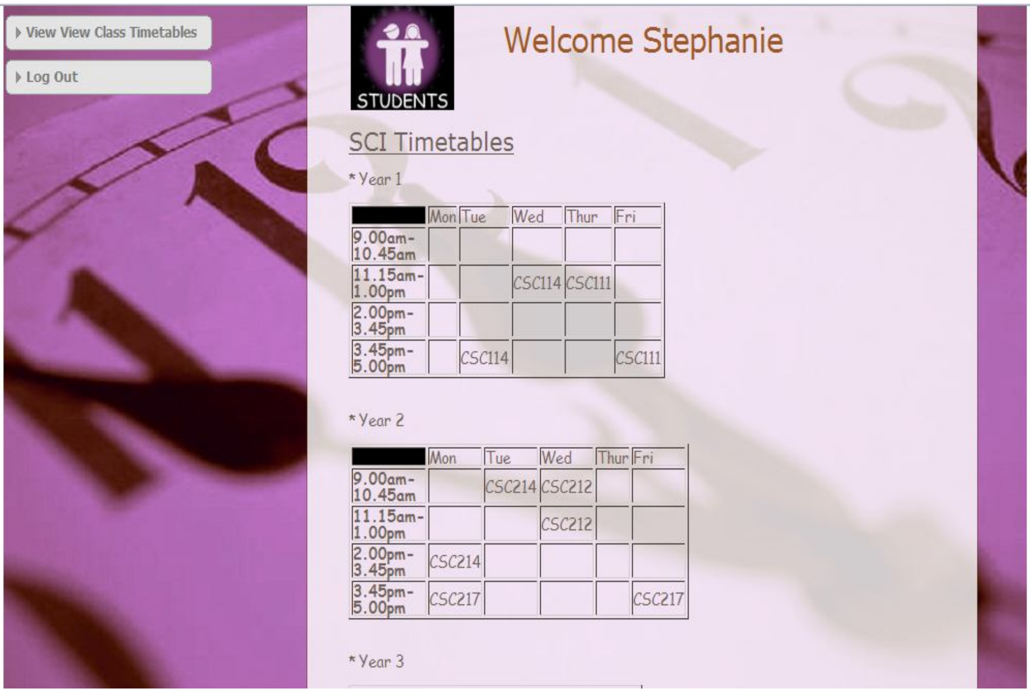
Register for courses

Registration Number	
Degree Name	School Of Mathematics
Year	1st
Semester	1st Semester
	Register

This is a password protected area only accessible to members of this site.

Back to Previous Page

View timetables



Welcome Stephanie

SCI Timetables

*Year 1

	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am					
11.15am-1.00pm			CSC114	CSC111	
2.00pm-3.45pm					
3.45pm-5.00pm		CSC114			CSC111

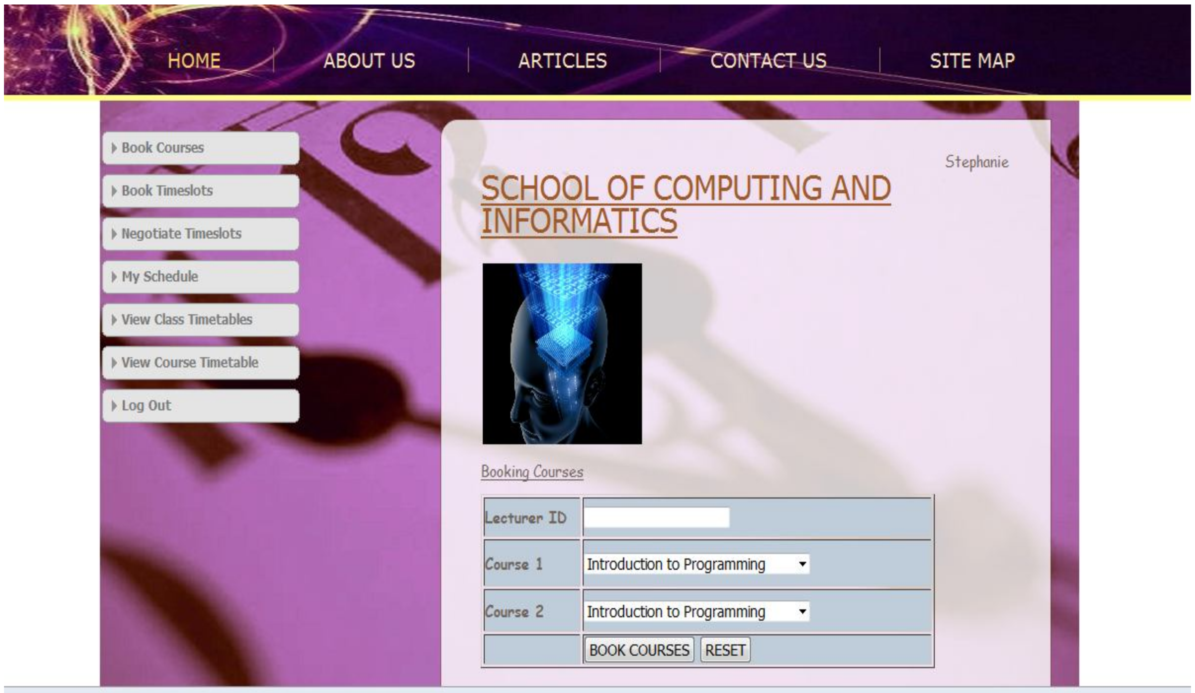
*Year 2

	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am		CSC214	CSC212		
11.15am-1.00pm			CSC212		
2.00pm-3.45pm	CSC214				
3.45pm-5.00pm	CSC217				CSC217

*Year 3

The lecturer login leads you to a group of pages that allow the lecturer to book courses and time slots, negotiate timeslots, view their schedule and other timetables as well.

Booking Courses



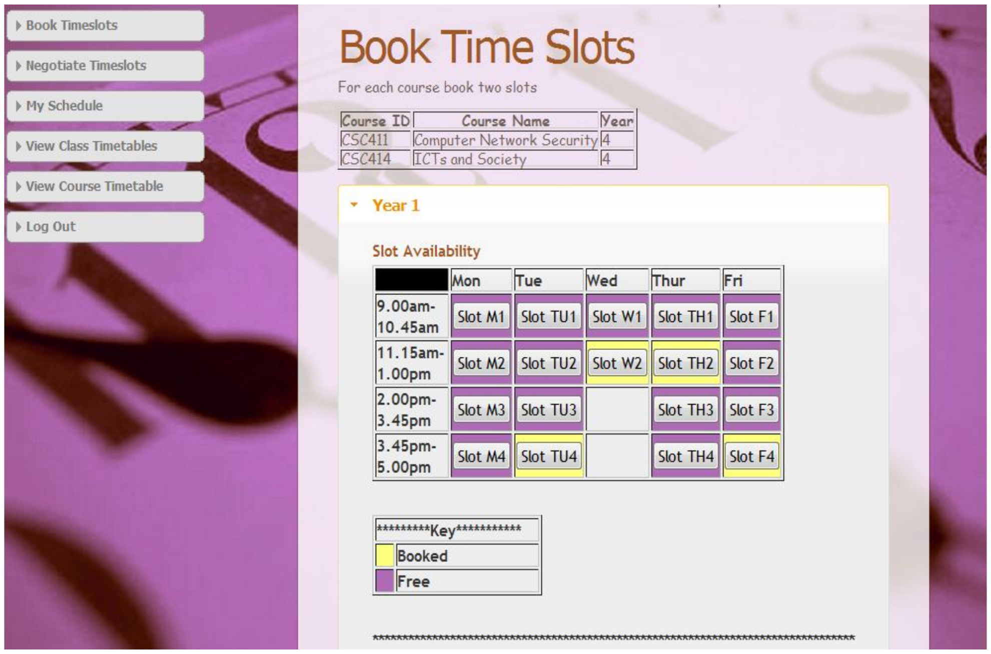
Stephanie

SCHOOL OF COMPUTING AND INFORMATICS

Booking Courses

Lecturer ID	
Course 1	Introduction to Programming
Course 2	Introduction to Programming
	BOOK COURSES RESET

Booking Timeslots



Book Time Slots

For each course book two slots

Course ID	Course Name	Year
CSC411	Computer Network Security	4
CSC414	ICTs and Society	4

Year 1

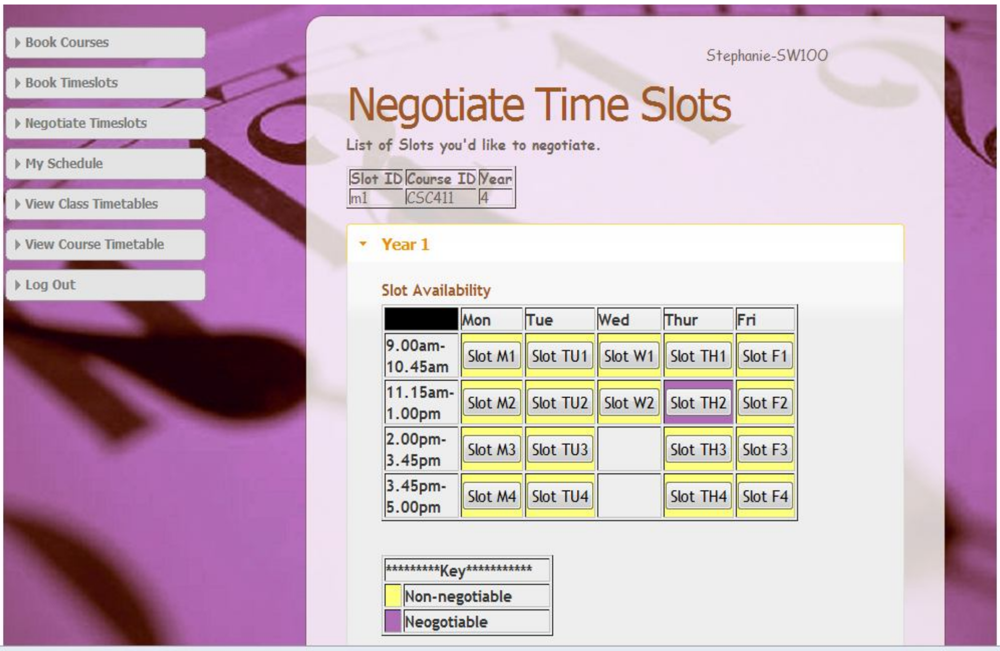
Slot Availability

	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am	Slot M1	Slot TU1	Slot W1	Slot TH1	Slot F1
11.15am-1.00pm	Slot M2	Slot TU2	Slot W2	Slot TH2	Slot F2
2.00pm-3.45pm	Slot M3	Slot TU3		Slot TH3	Slot F3
3.45pm-5.00pm	Slot M4	Slot TU4		Slot TH4	Slot F4

*********Key***********	
🟨	Booked
🟪	Free

****

Negotiating Timeslots



Stephanie-SW100

Negotiate Time Slots

List of Slots you'd like to negotiate.

Slot ID	Course ID	Year
m1	CSC411	4

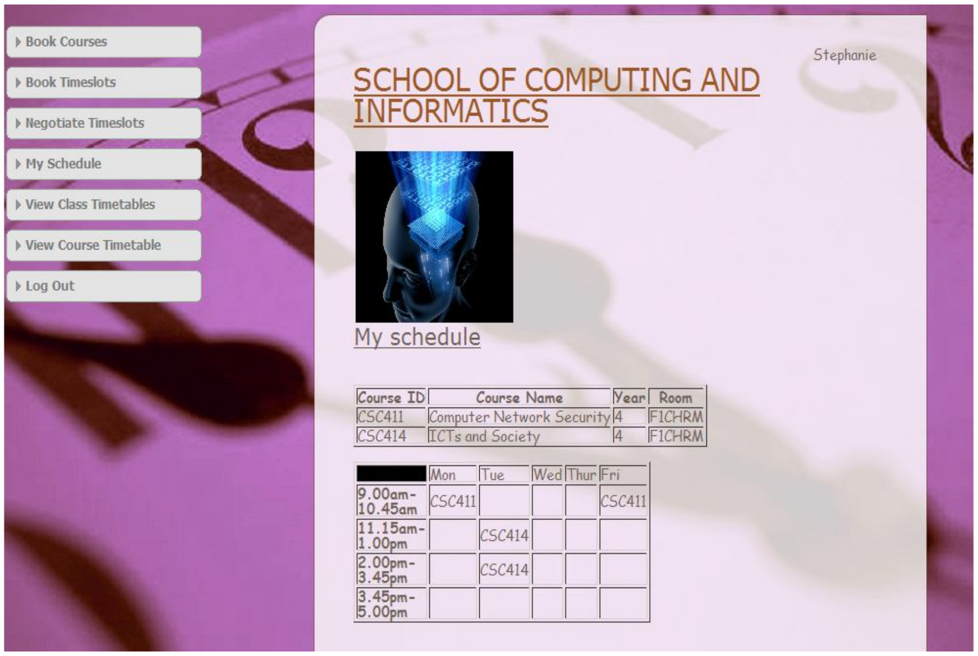
Year 1

Slot Availability

	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am	Slot M1	Slot TU1	Slot W1	Slot TH1	Slot F1
11.15am-1.00pm	Slot M2	Slot TU2	Slot W2	Slot TH2	Slot F2
2.00pm-3.45pm	Slot M3	Slot TU3		Slot TH3	Slot F3
3.45pm-5.00pm	Slot M4	Slot TU4		Slot TH4	Slot F4

*********Key***********	
🟨	Non-negotiable
🟪	Neogotiable

View Schedule



SCHOOL OF COMPUTING AND INFORMATICS

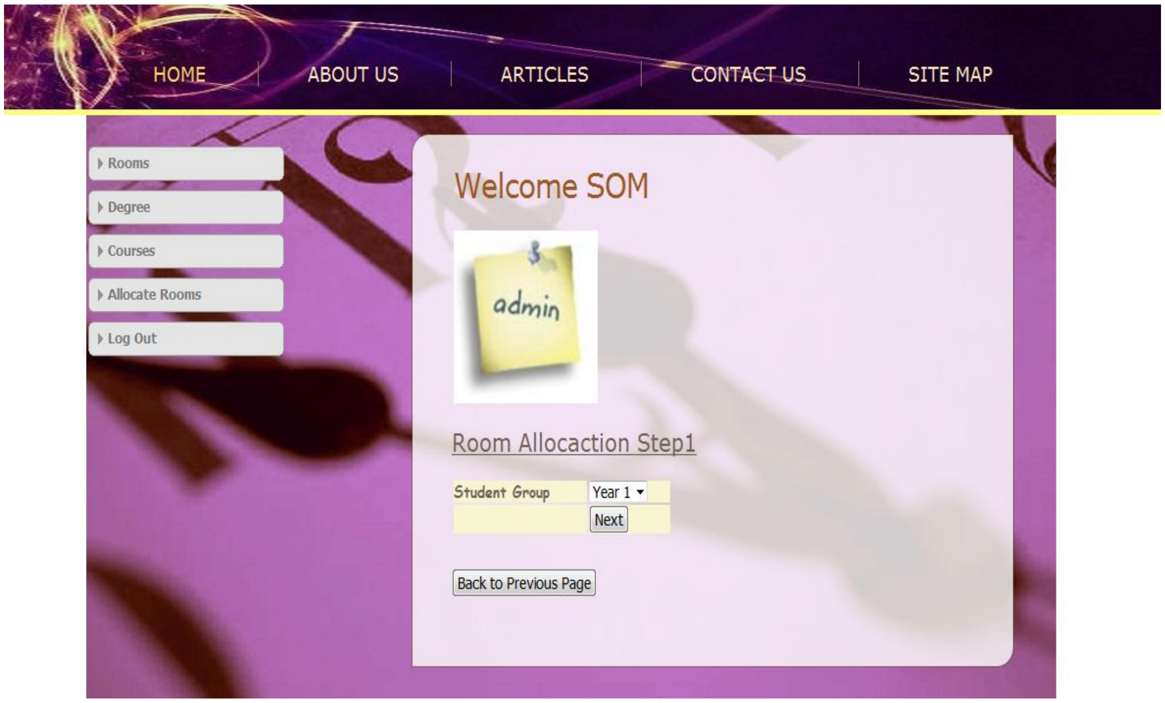
My schedule

Course ID	Course Name	Year	Room
CSC411	Computer Network Security	4	F1CHRM
CSC414	ICTs and Society	4	F1CHRM

	Mon	Tue	Wed	Thur	Fri
9.00am-10.45am	CSC411				CSC411
11.15am-1.00pm		CSC414			
2.00pm-3.45pm		CSC414			
3.45pm-5.00pm					

The administrator login allows the administrator to allocate rooms and edit other system information.

Allocation of rooms step 1



Welcome SOM

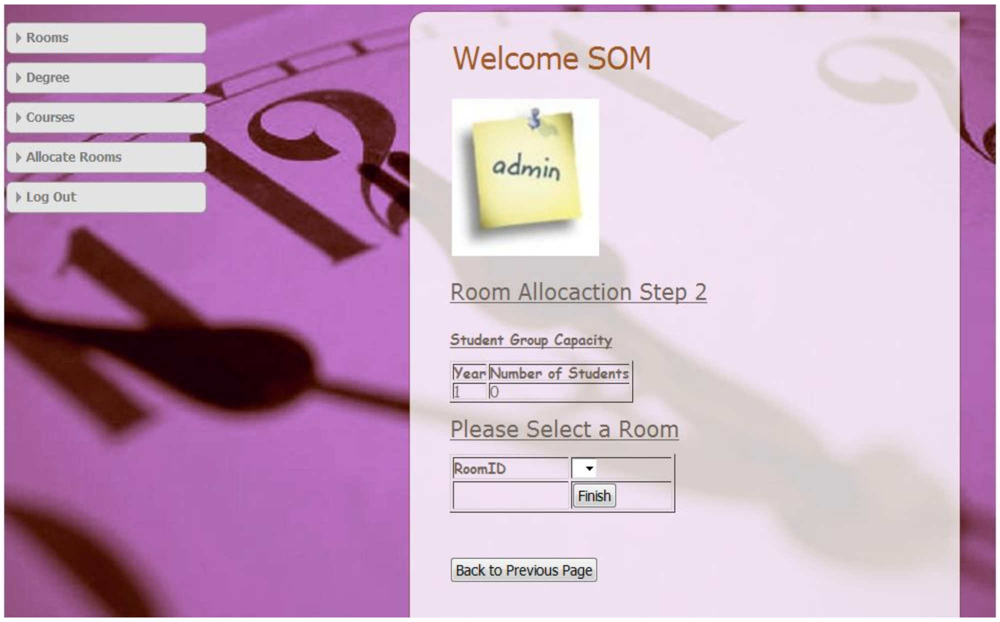
Room Allocaction Step1

Student Group: Year 1

Next

Back to Previous Page

Allocation of rooms step 2



Welcome SOM

Room Allocaction Step 2

Student Group Capacity

Year	Number of Students
1	0

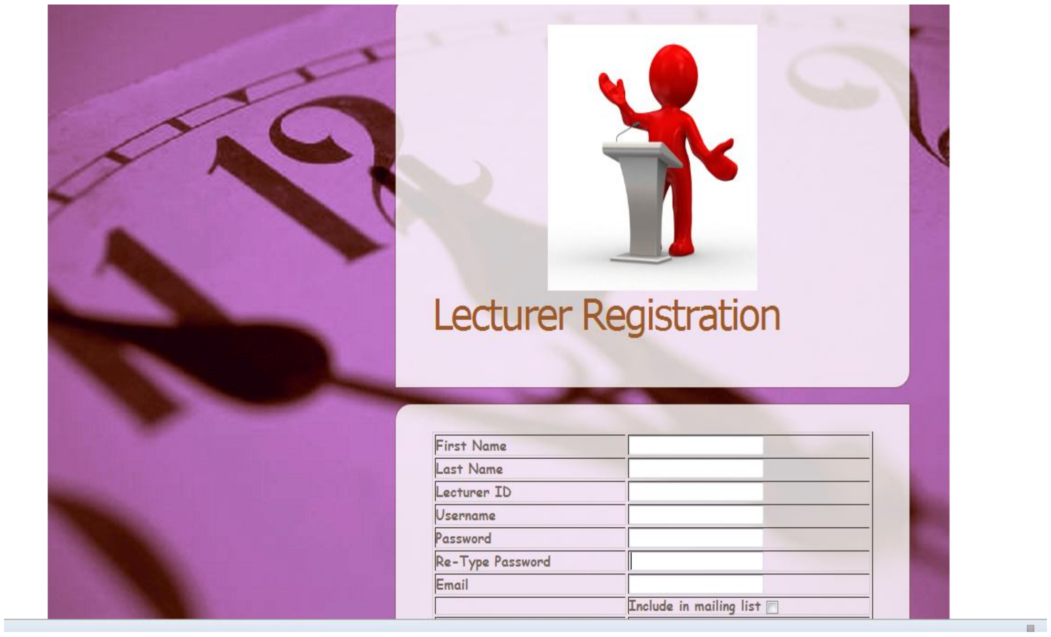
Please Select a Room

RoomID	

The users are able to navigate through these pages and perform these tasks using the side bar menu.

The system also provides users with an interface that allows users to enter data from remote areas. There are various forms to be filled in this system, these forms are processed and validated and processed on the client side and then this information is sent to the server. Examples are shown below.

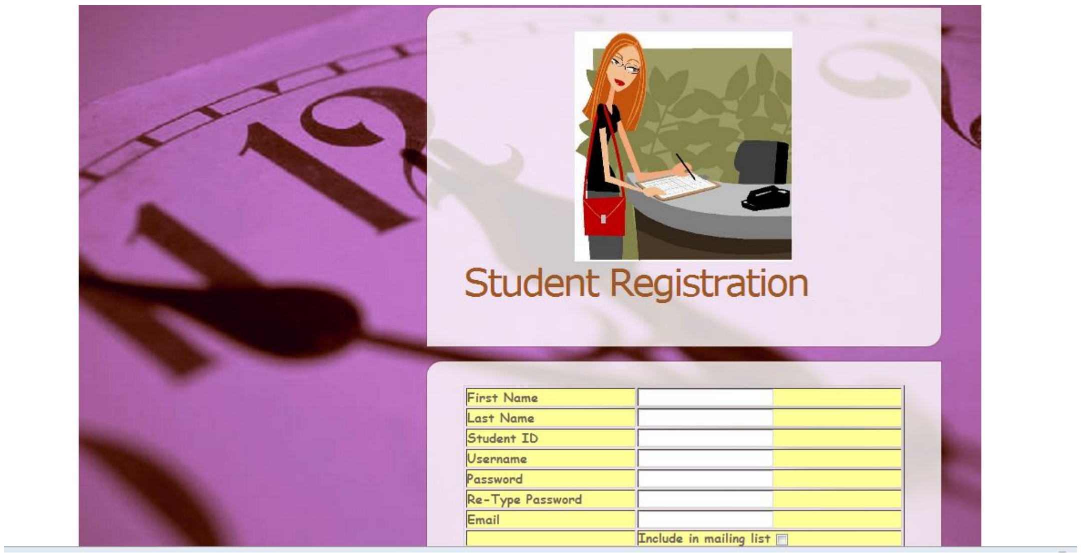
Lecturer registration form



Lecturer Registration

First Name	
Last Name	
Lecturer ID	
Username	
Password	
Re-Type Password	
Email	
	Include in mailing list ☐

<u>Student Registration Form</u>



Student Registration

First Name	
Last Name	
Student ID	
Username	
Password	
Re-Type Password	
Email	
	Include in mailing list ☐

REFERENCES

1. Slim Abdennadher, Mohamed Aly (2007).Constraint-Based University Timetabling for the German University in Cairo.http://met.guc.edu.eg/Repository/Faculty/Publications /67/Paper.pdf(accessed on 8th September 2011).

2. Yan Yang, Raman Paranjape, LuigiBenedicenti(2006). An Agent Based General Solution Model For the Course Timetabling Problem.http://jmvidal.cse.sc.edu/library/AAMAS-06/docs/P3fs405.pdf(accessed 19th October 2011).

3. C. Bernon, M. Gleizes, S. Peyruqueou, G. Picard (2007). ADELFE a Methodology for Adaptive Multi-Agent Systems Engineering. http://www.agent.ai/doc/upload/200307/bern02_1.pdf(accessed on 25th July 2011).

4. McCollum, B., Ireland, N.(2006). University timetabling: Bridging the gap between re-search and practice. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling.

5. C. Agerbeck, M. O. Hansen (2008).A Multi-Agent Approach to Solving *NP*-Complete Problems.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.4736& rep=rep1&type=pdf (accessed on 5th January 2011).

6. M.Oprea1(2006).Multi-Agent System for University Course Timetable Scheduling.http://fmi.unibuc.ro/cniv/2006/disc/icvl/documente/pdf/tech/2_oprea.pdf(accessed on 27th September 2011).