Automated Scheduling System for FCSIT

Ooi Ling Sern
(Matric No. : WEK 98084)

Supervisor:
Dr. Syed Malek Fakar Duani

Moderator:
Assoc. Prof. Dr. Roziati Zainuddin

Faculty of Computer Science and
Information Technology
University of Malaya

Session 2001/2002

Abstract

The final year thesis project as entitle, "Automated Scheduling System for FCSIT", is required o build a scheduling system for Faculty of Computer Science and Information Technology (FCSIT).

The objective of this project is to improve the current scheduling process from a manual system to a computerized system. At the end of the project, an automated scheduling system should be produce. This system also should be generic for most timetabling in school or any higher learning institutions.

Acknowledgement

Firstly, I would like to grasp this opportunity to express my deepest gratitude and thank to my supervisor, Dr. Syed Malek Fakar Duani, for his excellent guidance and patience throughout this thesis project. His ideas and advice does contribute a lot to this project.

Next, my sincere appreciation is forwarded to Assoc. Prof. Dr. Roziati Zainuddin for being my moderator.

At the same time, I would like to thank all my friends who have directly or indirectly assisted and guided me during this short period.

List of Figures

Chapter 4

List of Tables

Table of Contents

Chapter 1

Introduction

1.1 Project Definition

Currently, most of the schools or universities still used the traditional method, which is manually to do the scheduling job. This method not only waste time but also may be cannot do it completely. So, with this project, we will save a lot of time and easily do the scheduling job.

This project will assist the schools or universities administration to do the scheduling. This project is called Automated Scheduling System for FCSIT.

1.1.1 Intelligent System

Intelligence is defined as the capacity and ability to think, learn, reason and apply knowledge [5]. It refers to the ability to think and understand instead of doing things by instinct or automatically. **System** is defined as a set of things or ideas working together as a whole [5]. It is a collection of objects and activities, plus a description of the relationships that tie the objects and activities together. Therefore, **intelligent system** refers to a set of activities that utilized the concepts and response rules to reach a final objective.

Intelligent system is fundamentally a stimulus-response system. The stimulus is the communications entering through the needs. The system extracts information from this and represents it as a situation. Then, the system selects a response rule, appropriate to the situation (which allow the system to get nearer to its objective), and performs the response part of this rule. The system makes its selection of response rules from those that it finds stored in its memory that was the accumulated

response rules that it has generated from earlier experiences and from generalizations based on previously elaborated response rules.

1.1.2 Definitions for "Schedule"

There are two definitions for "**schedule**":

> *A **schedule** is a plan that gives a list of evens or tasks together with the times at which each thing should be done [5].*
>
> Or
>
> *A **schedule** or timetable is a placement of a set of meetings in time [10].*

A meeting is a combination of resources such as rooms, people and items of equipment. Some of these resources may specify by the problem, and some must be allocated as part of the solution.

1.1.3 Course Scheduling System

Course scheduling system is the problem of assigning times and places to a many separate courses to satisfy several constraints concerning capacities and locations of available rooms, free-time needs and other such considerations for lecturers and relationships between particular courses. The most prominent overall constraint (central to all timetabling problem) is that there should be no **clashes,** which means that any pair of courses which are expected to share common students or lecturers should not be scheduled simultaneously.

Course Scheduling or timetabling is a complex task. One wishes to schedule courses so that students can take the classes they want, without timing conflicts, so that

faculty desires are accommodated, and so that facilities are effectively utilized. Constraints include room capacity and other attributes, classes that must be scheduled at the same time or on the same day as other classes, faculty that may be available only at limited times, and having a minimum "downtime" between classes.

1.2 The Course Scheduling Problems

Timetabling is a particular form of the general scheduling problem. There are a set of events and a set of processes that need to be carried out for each of those events. In an educational context, scheduling is a problem that must be concern for every teaching institution. Every semester a new schedule must be produced to take into account changes in the staffs, students and courses. This often involves a large amount of administrative work. Due to the difficulty in constructing a schedule, a considerable attention has been devoted to automated scheduling. During the last thirty years, starting in the early 60's with Gotlieb (1963) and others, many researches related to automated scheduling have been done and several applications have been developed and employed successfully.

1.2.1 Course Scheduling at FCSIT

Several problems were faced while generating a course schedule for Faculty of Computer Science and Information Technology (FCSIT). The major problem is the actual size of the problem. The schedule for FCSIT covers more than 50 different courses, 40 lecturers and thousand over of students each semester and these numbers are believed to increase in the coming year.

The next problem is the variety and large number of constraints. These constraints sometimes are non-compatible. Due to the increasing number of courses, lecturers and students, the constraints on a particular schedule are getting more.

1.2.2 Issues in Course Scheduling

A schedule generally has an aim, specific constraints and resources. These must be elicited before the process of constructing a schedule can start. Some of the example issues are givens in figure.

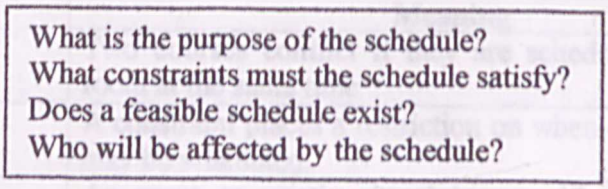


Figure 1.1 Issues in Schedule Construction.

Each of the issues mentioned are relevant to the course schedule and each one may be addresses individually. The purpose of the schedule, for example, is to ensure that all students take every course they are required to, while maintaining a reasonably efficient use of resources. It must satisfy certain constraints, in particular that no student is required to be in more than one course simultaneously.

1.2.3 Scheduling Terms

Scheduling problem has its own associated terminology and the terms associated with course scheduling problem are not specific. Below are the scheduling terms and their meanings.

Terms	Meaning
Conflict/Clash	Two courses conflict if they are scheduled to the same room at the same time.
Constraint	A constraint places a restriction on when or where courses may be scheduled.
Course	An event or meeting involving specific set of lecturers, students, rooms and possibly other resources.
Feasibility	A schedule is feasible if it satisfies all its constraints.
Period	A period is a fixed timeslot in which courses may be scheduled.
Resource	Resources are physical entities that are referred to by the schedule. For example, lecturer, student, room and equipment.
Room	A venue where courses may be held.
Session	All the timeslots that can be used in the schedule.
Schedule	A description of the movement and grouping of resources over time.

Table 1.1 The list of scheduling terms and their meanings.

1.3 Objectives

The main objective of this project is to automate the scheduling system given input parameters such as courses, characteristics of courses, constraint on rooms, time and lecturer availability. This project is very important since the number of courses offer, the lecturers and students are increasing and these make the schedule construction getting more and more complicated and it will be a difficult work to solve the problem manually. The automated scheduling system should:

- speeds up the generation of course schedule
- provides ease of gathering and manipulation of the lecturers, classes, rooms, courses and other records
- has the capabilities of reducing the time and effort for generating the schedule
- produced schedules that are feasible and with sufficient quality to be used
- be flexible enough to handle the variety of constraints encountered in real-life problems
- be user-friendly which means that easy to learn, easy to use and helpful
- be easy to maintain

1.4 Scope

This project typically not only covers on the faculties in University of Malaya but also can be used for the others faculties in all universities. However, the major concern here is to schedule courses that offer by Faculty of Computer Science and Information Technology (FCSIT) in University of Malaya and courses that will be held in this faculty.

Courses that offer by this faculty including the 1st year Degree, 2nd year Degree and 3rd year Degree courses in Computer Science and Information Technology. These courses can be a core subject, an elective subject or a lab course and sometime a course might be a combined course for two or more different year of students.

Chapter 2

Literature Review

2.1 Automated vs. Interactive Scheduling

Most of the people believe that the scheduling problem cannot be done full automatically by the system. There are two reasons why they think like that. The first reason is there are some reasons that make a schedule better than the others' schedules, which cannot easily, determined by an automatic system. The second reason is it is too complicated or too difficult for a system to search for a schedule that can satisfy all the constraints. A human intervention may be needed to produce an ideal schedule, which the system may be not able to find a solution itself. So, most of the scheduling systems enable user to adjust the final schedule manually. These systems are called the interactive or semiautomatic scheduling system.

The size and complexity of the scheduling system have provoked a trend towards more general problem solving algorithms with the **automated** system. A full search of all possible schedules is not acceptable for the users who want to produce the schedule in a short time. The scheduler must able to solve this problem in return for a gain in speed. This could be done by an automated system.

2.2 Solution Approaches

Many different methods have been developed and used successfully to solve real scheduling problems. These methods are used in one particular department or institution only, and there is no benchmark to facilitate which approach is the best solution to scheduling problem.

Later on, researchers started to apply general techniques to the scheduling problem. The approaches taken included integer programming, network flow, graph coloring and others. But, recently, this problem has been tackled with techniques belonging also to artificial intelligence such as logic programming, simulated annealing, tabu search, genetic algorithms and constraints satisfaction.

2.2.1 Heuristics Algorithms

Heuristics is a term derived from a Greek word meaning to discover [14]. A heuristic is a rule of thumb, strategy using some knowledge about particular problem domain, which is used to guide search processes towards acceptable problem solutions. It guides to the lines that have high probability of success while avoiding wasted effort [12].

A variety of heuristic techniques have been devised to improve the efficiency of the search process. These fall into two categories: general-purpose heuristics and domain specific heuristics. General-purpose heuristics are flexible procedures that can be used on a wide variety of problems to prune the search space. Domain specific heuristics are special procedures that apply only to the given problem. One or both types of heuristics may be used depending upon the problem. In both cases,

the goal of the heuristic is to provide knowledge that eliminates the problem of combinatorial explosion and shortens the search time [14].

The scheduling methodology described in Figure 3.1. Algorithm 1 finds a non-conflicting set of courses and Algorithm 2 assigns these selected courses to rooms. The process is circulated for each period until all courses have been scheduled to a period and a room without conflicting. The unselected courses may be taken as a basis for the search of courses for the next period since they will have no conflicts between them, but also will not conflict with any exams from the previous period [7].

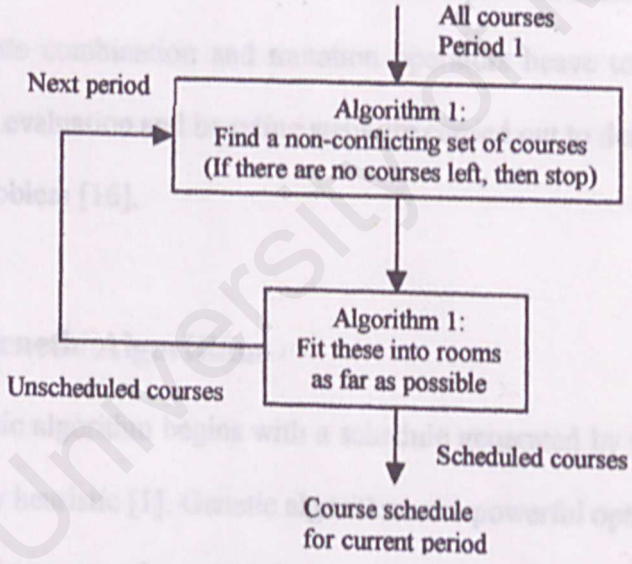


Figure 2.1 A heuristic scheduling framework.

One limitation in this approach is that heuristic assignment algorithms do not easily allow for the possibility of search and strongly limit the type of constraints that may be created for. Some of the constraints were viewed, as hard constraints (cannot be

violated) whereas others may be viewed as soft (may be violated if necessary). It is possible to have a situation where the hard constraints may be ignored, which caused by the large number of soft constraints as the way they are represented [7].

2.2.2 Evolutionary Algorithms

Concurrent with part of the investigation, Lemoine developed a code for the pipeline-scheduling problem, which used evolutionary programming [1]. An evolutionary algorithm works on a set of possible solutions to a problem. Each solution is encoded as a gene (a string of symbol) [6].

Therefore, evolutionary scheduling typically begin with defining the chromosome representation of the event. Next the constraints are translated into a fitness function. Appropriate combination and mutation operators heave to be decided. Then, the selection, evaluation and breeding steps are carried out to determine the best solution for the problem [16].

2.2.3 Genetic Algorithms

The genetic algorithm begins with a schedule generated by the initial application of the greedy heuristic [1]. Genetic algorithms are powerful optimization tools that may be applied to a complex scheduling problem. They often capable to find an optimal solution even in the most complex of search space [8].

The algorithm uses two basic operations, mutation and crossover. Given a schedule, a mutation replaces a random number of randomly selected classes with randomly

selected classes of the corresponding course and shift. A crossover is slightly more complicated and involves combining two schedules [1].

2.2.4 Graph Coloring

In this approach, each event is associated with a vertex in a graph, and there is an edge between each pair of event that cannot be scheduled at the same time. For example, lectures hat share a common lecturer or common students are joined. Unavailability and preassignments are managed imposing some external constraints on the colorability of specific vertices of the graph [3]. Each vertex is then colored such that no pair of vertices connected by an edge has the same color [8].

A coloration of the resulting graph, respecting the external constraints, can be easily turned into a schedule, by assigning a period to each color, and consequently scheduling the events corresponding to a vertex to the period corresponding to its color [3].

2.2.5 Memetic Algorithms

Memetic algorithms are an extention of genetic algorithms, based on a model of how an idea evolved. A **meme** is the basic unit of information that reproduces itself as a result from people exchange idea. A meme differs from gene as it can be improved during their lifetime. Genes usually were passed between individuals unaltered but individuals adapt the meme if it is better [9,10].

The advantage gained with using the Memetic algorithms is that the space of possible solutions is reduced to the subspace of local optima.

Each solution in the population is represented as a number of memes and each of these contains information. Initial population is generated using weighted wheel. Then this population is crossover using a combination of light and heavy random mutation, followed directly by application of hill climbing techniques. After that, evaluation function is applied to it to calculate the fitness. In order to reduce the population, individuals are chosen from the pool with a section function to joint the new population. This is achieving by using the roulette wheel selection to weight the fitness of an individual to its competition [9].

2.2.6 Network Flow Techniques

In this technique, network model was used as the core of the scheduling algorithm. The scheduling problem was reduced to a sequence of network flow problems. A network is created for each event so that the flow in the network identifies constraint and resources given. The construction of the network is repeated for all events eventually. If a solution is found for all the networks, it leads to a complete schedule [3].

But there is one limitation in this approach that there is no backtracking on the events already scheduled, as a result there is no guarantee that the solution is found whenever it exists. Therefore the procedure solves the scheduling problem if it finds a feasible solution, otherwise a human intervention changes some of the constraints manually so the reason of unfeasibility is get rid [3].

2.2.7 Simulated Annealing

Simulated annealing is a probabilistic local search technique for findings optimize solutions to scheduling problem [3]. Simulated annealing is a search strategy, which keep track of one feasible schedule [14].

This technique starts by creating a random initial solution. The main procedure consists of a loop that generates a neighbor of the current solution, slightly altered at random from the current one. This neighbor is accepted as the current schedule if it has a lower penalty. If the neighbor has a higher penalty, it may be accepted according to a probability that related to a control parameter called temperature. The temperature of inferior neighbors is decreasing after a particular number loops. This procedure stop when a solution close to objective is found and no solution that increases the objective function is accepted any more, which the system is frozen.

One limitation with simulated annealing is that the process can take a long time to stop in order to achieve a good solution.

2.2.8 Tabu Search

Tabu Search is a local search technique, family of general-purpose techniques for the solution of optimization problem [3]. The local search techniques start from an initial solution, enter an iteration that navigate the search space, stepping iteratively from one solution to its neighbors [4].

Tabu Search remembers just one current feasible schedule. A Tabu Search maintains a list of tabu moves to prevent recycling. A tabu list represent the schedule which

having visited recently are forbidden in order to prevent the search staying in the same area. This tabu list is usually with a fixed size, with first-in-first-out basis. An aspiration level, which represents the best solution visited, is maintained. If a tabu schedule reaches at the aspiration level, it may be removed from the tabu list [14].

The Tabu Search stops either when the number of iterations reaches a given value or when the value of the objective function in the current solution reaches a given lower bound.

2.2.9 Constraint Logic Programming

Constraint logic programming (CLP) system is a tool for modeling a specific search problem, which provides the ability to declare variables and their domains, and constraint on the problem [3]. CLP generalizes logic programming by replacing unification with constraint solving over a particular domain with a set of discrete variables [11].

In CLP, a labeling strategy dictates the order in which the search space is traversed. There are two orders, which should be take note: the order the variables are instantiated and the order in which the variables' values are assigned [14].

In order t search for a solution, a CLP system generates values for the variables, propagating values through the constraints in order to prune parts of the solution space where inconsistencies are discovered. Therefore, this method is a backtrack search where the constraints allow the system to look ahead to the consequences of decisions and spot failures earlier [6].

CLP is a paradigm coined for solving combinatorial problems. It has been used successfully for a variety of practical applications from scheduling to financial analysis. Constraint handling has been introduced into Logic Programming to simplify the expression of problems and to dramatically improve program efficiency. Essentially the constraints are used to prune the search tree defined by the logic program in which they are embedded [2].

2.3 Case Studies

This section provides some examples of current GA based timetabling systems that are actually being used to provide practical solutions in institutes, giving a flavor of what is currently achievable.

A commercially available evolutionary timetabling systems is ACT (Automated Class Timetabler) which is a state of the art system for university timetabling, running under Windows-95, developed in Korea. The system uses a hybridized algorithm, which incorporates hill climbing, a genetic algorithm and also a best-first heuristic search algorithm. It is able to solve all hard constraints and optimizes soft constraints in a two-phase approach. The system produces class timetables, professor timetables and room timetables. It has been successfully tested in 15 universities in Korea, and can cope with considerably large problems. In the largest university tested, timetables were evolved for 70 departments, with 700 professors, and 4000 subjects, satisfying 5 hard constraints and optimizing up-to 10 soft constraints. Using a Pentium 133Mhx processor, timetables can be produced in around 20 minutes, and hence the system is extremely fast. It contains a straightforward graphical user interface based on MS-Windows for data entry. An added feature is manual scheduling function that enables any lesson attributes to be changed after the automated procedure has finished, with guidance from the system as to the possible changes that can be made.

Several other universities are using implementations of evolutionary timetabling systems to schedule lectures and exams within the university. For example, at Napier University, in Edinburgh, a memetic algorithm based timetabling system has

been developed, known as Neeps *and* Tatties. The approach and representation used in this case is entirely different from that described above. The university to schedule the entire university timetable has successfully used Neeps and Tatties. This is an extremely large problem --- for example, in the first semester of 1997 there were 2067 events to place in 45 time-slots and 183 rooms. These were attended by 669 lecturers and 978 student groups. The timetables produced had to optimize 12 different criteria --- the algorithm was run on a single machine, and takes about an hour to satisfy all hard constraints. From then on, it begins to optimize the soft constraints and can be run indefinitely. In practice, the algorithm was left running for about three days. However, a great advantage of the program is that it can be seeded with previous results, which allows solutions to evolve gradually as the data is gradually entered, or that the program can be seeded with a solution from a previous year to speed things up. A novel feature of this program is the ability to interactively change the weights of the soft constraints as the program runs, and see the results with immediate effect, which is of invaluable help to an administrator trying to find a satisfactory timetable.

Another system was developed at the University of Edinburgh, where it is used each year to schedule examinations. This is known as GATT, standing for Genetic Algorithm Time Tabler. This software was also used by Harvard Business School to schedule their examinations, and was found to outperform a commercially available timetabling system in trials. In a direct head-to-head competition with the commercial (non-GA-based) software, exams were scheduled over 15 slots -- **GATT** found a schedule with 2 students with direct conflicts, involving 2 courses. The commercial software came up with 14 conflicts when running automatically,

while their manual attempt to develop a schedule came up with fewer, with only 5 students with conflicts! **GATT** has also been tested on a real timetabling problem from a secondary school in Belgium, which involved scheduling classes and teachers for the first two-year groups. The problem involved 10 different class groups, 26 different teachers, and 318 different courses. In total, 5320 edge constraints were involved. The constraints included a significant number of specifications and exclusions, where an event was either pre-assigned to a particular slot, or excluded from a range of slots. Defever reports that GATT removed all clash violations. He does not specify however if other constraints, such as room constraints or proximity constraints were considered.

2.4 Programming Language

The programming language that used in this project is Microsoft Visual Basic 6.0.

2.4.1 What is Visual Basic 6.0?

Microsoft Visual Basic 6.0 was mainly used in development and designing of Automated Scheduling System. Although the syntax of the Visual Basic language parallels that of previous versions of BASIC, Visual Basic represents a conceptually different approach to programming, called *event-driven programming*. An application developed with an event-driven model responds to events that happen in the computer environment. Such events include the press of a mouse button or a call from another application running concurrently.

2.4.2 Why Visual Basic 6.0?

Visual Basic 6.0 is a user-friendly programming language. We can create the user interfaces for the applications by directly manipulating on-screen objects such as control buttons and dialog boxes. We assign value for the properties of these objects by selecting various characteristics such as colors and fonts from an extensive list. One of the great triumphs of Visual Basic is that we can develop complete Windows applications with minimal program instructions.

2.4.3 Features of the Software

The software used in this project is the Visual Basic version 6.0.

2.4.3.1 Safe and Fast

Visual Basic meets the professional's demands for speed, performance and safety. It produces compact, fast and highly optimized native machine code, comparable to that of a C compiler.

2.4.3.2 A Wealth of Facilities

All the components that make up the integrated environment can be used in the programs. A wide range of standard predicates provides all the features that expected from a professional programming language.

Chapter 3

Methodology

3.1 Constraints

A constraint is a restriction on resources related to an event [16]. Scheduling constraints are many and varied. Constraints are usually divided into 'Hard' and 'Soft' constraints:

- **Hard constraints**. Hard constraints are a restriction that a schedule must follow. A schedule, which breaks this constraint, is not a feasible solution and must be reconstructed or rejected.
- **Soft constraints**. Soft constraints are less important compared to hard constraints. Usually it is impossible to avoid breaking at least some of these constraints [10].

3.1.1 Constraints on Course Scheduling

3.1.1.1 Lecturer

- No lecturer is schedule to two different courses at the same time.

3.1.1.2 Course

- No two same level and same major courses run simultaneously.

3.1.1.3 Time

- Sessions for Monday to Friday are to hourly intervals from 8.00 a.m. to 6.00 p.m.
- There should not have any classes on Saturday and Sunday.

3.1.1.4 Room

- There must not be more students scheduled to a room that more than the capacity of the room.
- There should not have two classes scheduled in the same time at the same room.
- Lab classes should be schedule to computer lab only.
- Tutorial classes must be scheduled in the tutorial room only.
- Courses with more students must be scheduled in the large rooms.

3.2 Rule-Based System

Rule-based system is defined as a computer program that processes problem specific information contained in the working memory with a set of rules contained in the knowledge base, using an inference engine to infer new information [13].

The rule-based expert system consists of a number of rules or heuristics and conventional recursion to assist in carrying out class assignments. In a rule-based system, the rules are contained in the knowledge base and also the working memory. These rules are combined through the inference engine to infer new information as shown in Figure 3.1.

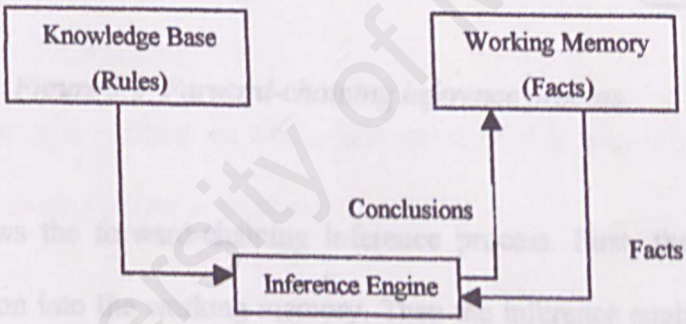


Figure 3.1 Rule-based model.

3.2.1 Forward Chaining

Forward-chaining is defined as "inference strategy that begins with a set of known facts, derives new facts using rules whose premises match the known facts and continues this process until a goal state is reached or until no further rules have premises that match the known or derived facts" [13].

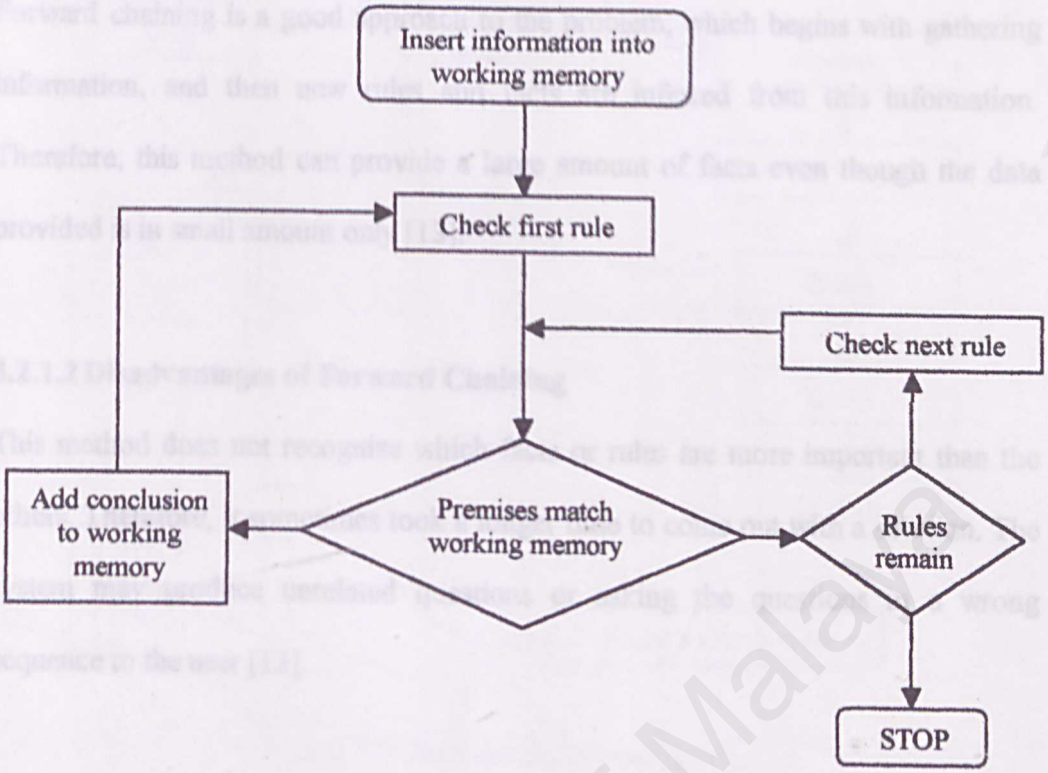


Figure 3.2 Forward-chaining inference process.

Figure 3.2 shows the forward-chaining inference process. First, the system will insert information into the working memory. Then the inference engine checks the rules in the working memory and it will fire new rules to the working memory from its rules' conclusion. The facts in the working memory are continually updated. Rules in the system represent possible actions to take when specified conditions hold on items in the working memory. The conditions are usually patterns that must match items in the working memory, while the actions usually involve adding or deleting items from the working memory.

3.2.1.1 Advantages of Forward Chaining

Forward chaining is a good approach to the problem, which begins with gathering information, and then new rules and facts are inferred from this information. Therefore, this method can provide a large amount of facts even though the data provided is in small amount only [13].

3.2.1.2 Disadvantages of Forward Chaining

This method does not recognize which facts or rules are more important than the others. Therefore, it sometimes took a longer time to come out with a solution. The system may produce unrelated questions or asking the questions in a wrong sequence to the user [13].

3.2.2 Backward Chaining

Backward chaining is defined as "inference strategy that attempts to prove a hypothesis by gathering supporting information" [13].

Backward Chaining often referred to as hypothetical reasoning starts with a specific hypothesis, or set of hypotheses, called the agenda. The agenda structures knowledge and controls backward-chaining event processing by ordering hypotheses or goals, in a numbered, hierarchical outline. The backward-chaining inference engine works backward from the agenda, pursuing a hypothesis via its search order strategies.

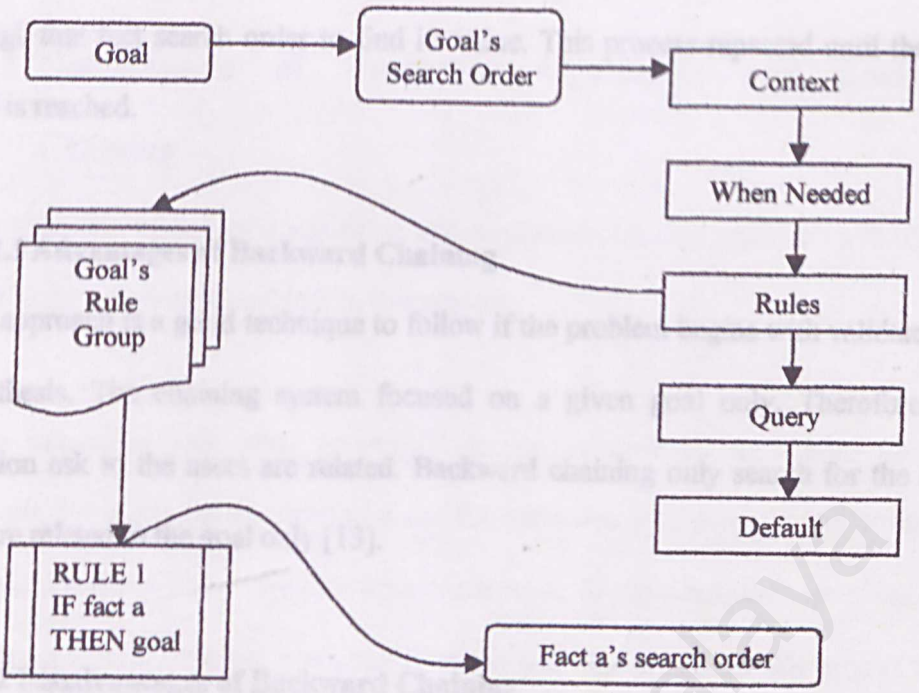


Figure 3.3 Backward-chaining process.

Figure 3.3 shows the backward-chaining process. The backward-chaining inference engine start with the first goal in the knowledge base's agenda. The goal will be the attribute of the domain class or an instance.

Then inference engine backtracks through the goal's search order. This search order provides the backward-chaining inference engine where, and the order to obtain the value needed to prove the goal. Combination of the session context, a when needed method, the knowledge base's rules, and end-user query, or the default values were used to obtain the value.

If the goal can be verify through rules, the inference engine backtracks through the goal's rule group (set of rules that can conclude the goal). If an attribute in the

premises of a rule can conclude the goal, then the inference engine backtracks trough that fact search order to find its value. This process repeated until the final goal is reached.

3.2.2.1 Advantages of Backward Chaining

This approach is a good technique to follow if the problem begins with validates the hypothesis. The chaining system focused on a given goal only. Therefore, the question ask to the users are related. Backward chaining only search for the rules that are related to the goal only [13].

3.2.2.2 Disadvantages of Backward Chaining

The system will continue to follow a line given for reasoning although that particular hypothesis cannot be proved. As a result the system did not stop [13].

3.3 Designing Backward Chaining System

3.3.1 Combination of Heuristics Algorithms and Backward Chaining

A combination of heuristic algorithm and backward-chaining approach will be used to solve the scheduling problem in this project.

Using heuristics will guide to search processes towards acceptable problem solutions. From the expert knowledge and the natural way of doing, it will be easy to produce feasible schedule by following some rules. In the scheduling case, the hard constraints should be given a priority over the soft constraints. In this project there are few heuristic algorithms as below:

1. Find a core course, which is a combined course with a preferred timeslot and fixed this course into the schedule.
2. Find a core course, which is not a combined course with a preferred timeslot and fixed this course into the schedule.
3. Find an elective course, with a preferred timeslot and fixed this course into the schedule.
4. Find a core course, which is a combined course and fixed this course into the schedule.
5. Find a core course, which is not a combined course and fixed this course into the schedule.
6. Find an elective course and fixed this course into the schedule.

Algorithm 1 finds a core course, which is a combined course with a preferred timeslot and fixed this course into the schedule. This algorithm is repeated until all

the core courses, which is a combined course with a preferred timeslot have been scheduled to a period and a room without conflict. Then the system move to Algorithm 2 to finds a core course, which is not a combined course with a preferred timeslot and fixed this course into the schedule. The process is then continues same as previous algorithm.

The backward-chaining methodology approach had been chosen as the basic in this project. This is because the scheduling process always starts with hypothesizing a solution where to fix a time for a particular event and then check whether this is valid without violating any other constraints. At the same time, more data are needed to make conclusions in the scheduling process, which the backward-chaining approach is more suitable here. In addition, the backward-chaining system search for the rules that are relevant to the goal only compare to forward-chaining method that does not recognizes which rules are more important. Therefore, backward chaining will be the more suitable approach to solve the scheduling problem compare to forward-chaining approach that sometimes took a longer time to come out with a solution.

3.3.2 Defining the Problems

The scheduling problem is study detailed here. The major task in this project is to develop an automated scheduling system to assist a scheduler with the production of new schedule.

Information relates to scheduling problems such as courses offer, details about a course, information on lecturers are being collected. Details on a course such as the

code, course's name and credit hours as well as lecturer information such as lecturer code, name and courses taken by the lecturer each new semester are collected too.

3.3.3 Defining the Goals

Coding of a backward-chaining system must begin with defining the system's goal. The automated scheduling system will have two principal goals to achieve:

- Determine the time-slot for a course.
- Determine the venue for a course.

A time-slot is the period where the course can be place on the schedule. The time-slot or period for a particular course must be in between the session determined by the user. The structure of the time-slot for a course is determined from user input to the system too. To determine whether a particular time-slot is available for a course, two situations that must be concern:

- Students availability
- Lecturer availability

There are 3 different goals that this system to pursue. For a course and a particular time,

- All students involved available at the particular time.
- The lecturer involved available at the particular time.
- The particular room available at the time.

A venue for a course is a room where the course will be held in that particular time. To determine whether a venue is available at the time, situation that must be concern:

- Room availability

3.3.4 Defining the Goal Rules

In this project, the system has to fix the course into the schedule, which is the rules' conclusion. Therefore, necessary preconditions to satisfy the rules' conclusion should be determined. The general form for all goal rules will look like following [13]:

IF Precondition_1
AND Precondition_2
.
.
THEN Conclusion

The principal issues considered when making a rule's conclusion in this system:

- Student availability
- Type of course (core subject or elective subject)
- Course structure (ordinary course or combined course)
- Lecturer availability
- Other courses at the same time (yes or no)
- Room availability
- Room size (more than or less than students' size)
- Equipment
- Special room (lab class or not a lab class)

3.3.5 Expanding the System

Then the system is expanded to cover a broader understanding of the problem. The rules are sub divided according to the type or course whether is a lab course or not.

3.3.5.1 Course Rules

IF room available

AND lecturer available

AND student available

THEN time available for course

IF student's size less than room's size

AND no other course at the same time in the room

THEN room available for course

IF lecturer not having other course at the same time

THEN lecturer available for course

IF student not having other conflict's course

THEN student available for course

3.3.5.2 Lab Course Rules

IF lab available

AND student available

THEN time available for lab course

IF student's size less than lab's size

AND no other course at the same time in the lab

THEN lab available for lab course

IF student not having other conflict's course

THEN student available for lab course

3.4 Algorithms for Course Scheduling System

First, the system will find a timeslot for a subject. If the timeslot value is consistent, then an available room that satisfies constraints is selected for that particular timeslot. If the room is not available, then the next timeslot is selected by the random function. Consistency tests will be performed on all new values selected by the random function. The purpose of consistency test is to determine whether values from the domains of related variables are consistent (non-conflicting) or not with respect to constraints. If not consistent, the values will be removed from its domains [17]. The algorithm is showed in Figure 3.4.

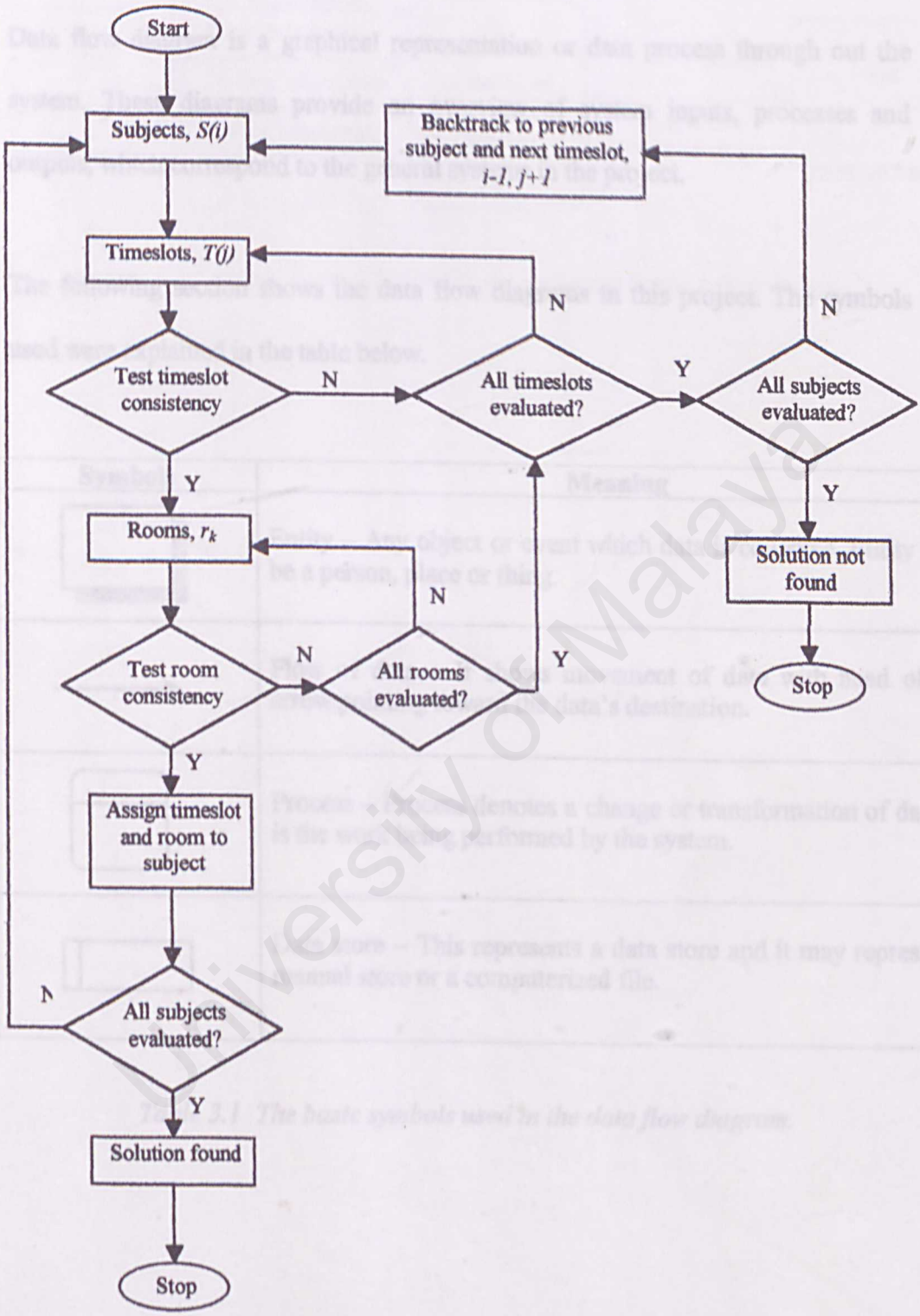


Figure 3.4 The algorithm for course scheduling system.

3.5 Processes and Flows in the System

Data flow diagram is a graphical representation or data process through out the system. These diagrams provide an overview of system inputs, processes and outputs, which correspond to the general systems in the project.

The following section shows the data flow diagrams in this project. The symbols used were explained in the table below.

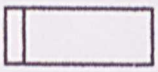
Symbols	Meaning
	Entity – Any object or event which data is collected. Entity may be a person, place or thing.
	Flow of data – It shows movement of data with head of the arrow pointing toward the data's destination.
	Process – Process denotes a change or transformation of data. It is the work being performed by the system.
	Data store – This represents a data store and it may represent a manual store or a computerized file.

Table 3.1 The basic symbols used in the data flow diagram.

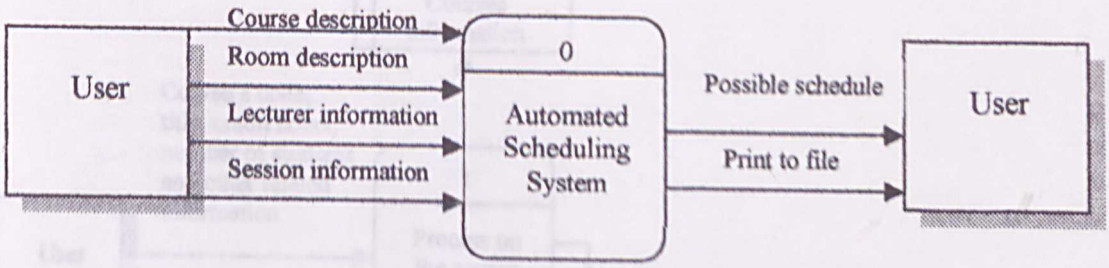


Figure 3.5 Context level diagram for the automated scheduling system.

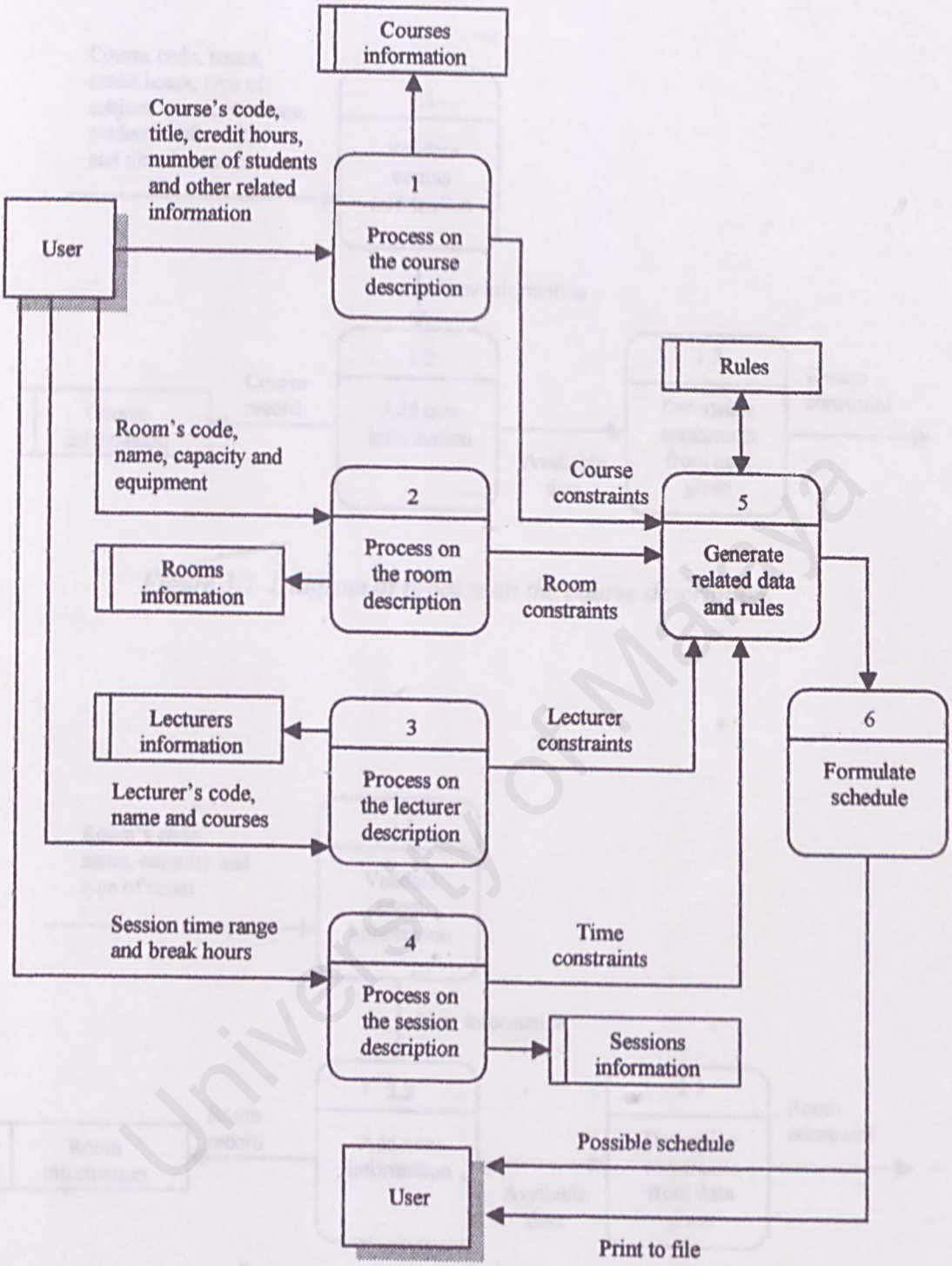


Figure 3.6 Diagram of the automated scheduling system.

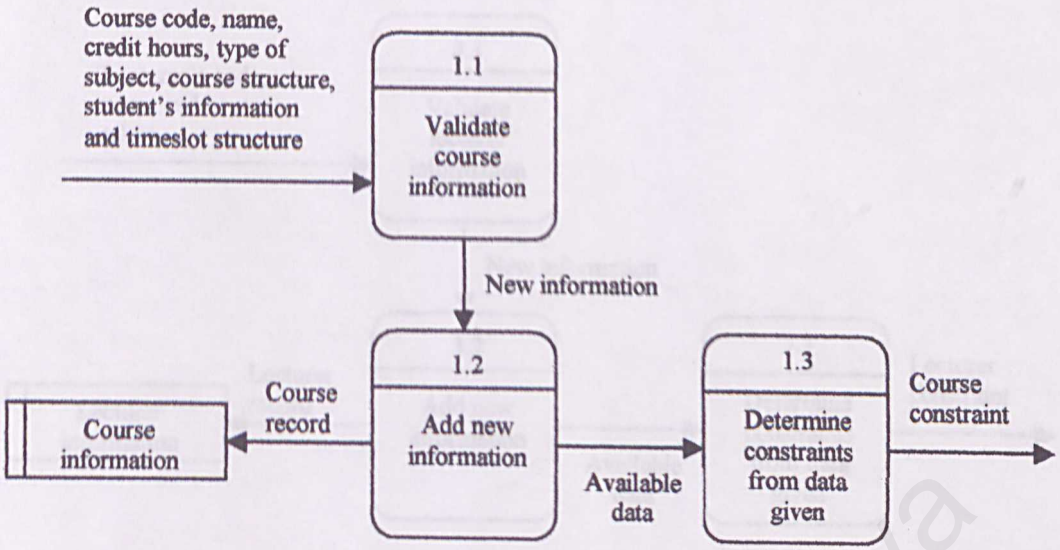


Figure 3.7 Diagram of process on the course description.

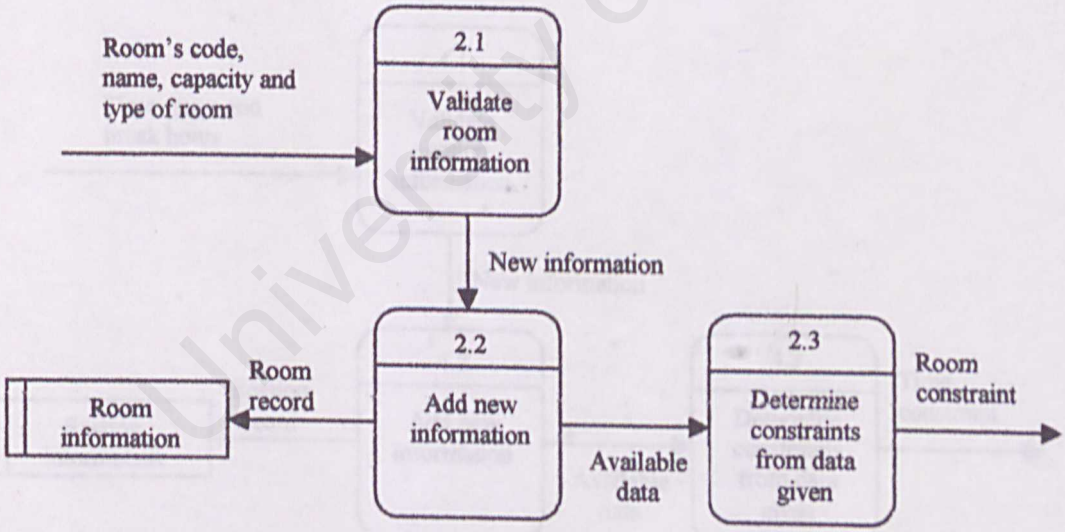


Figure 3.8 Diagram of process on the room description.

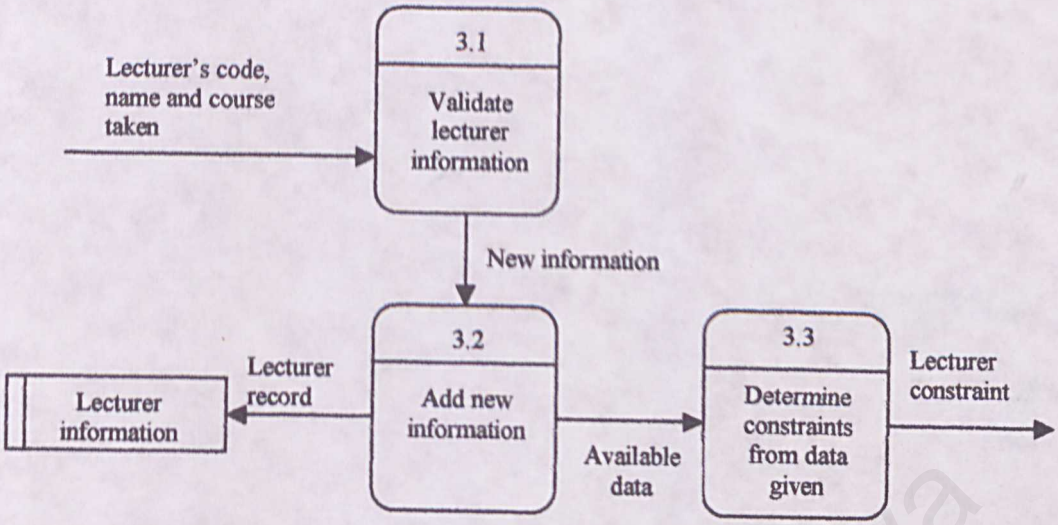


Figure 3.9 Diagram of process on the lecturer description.

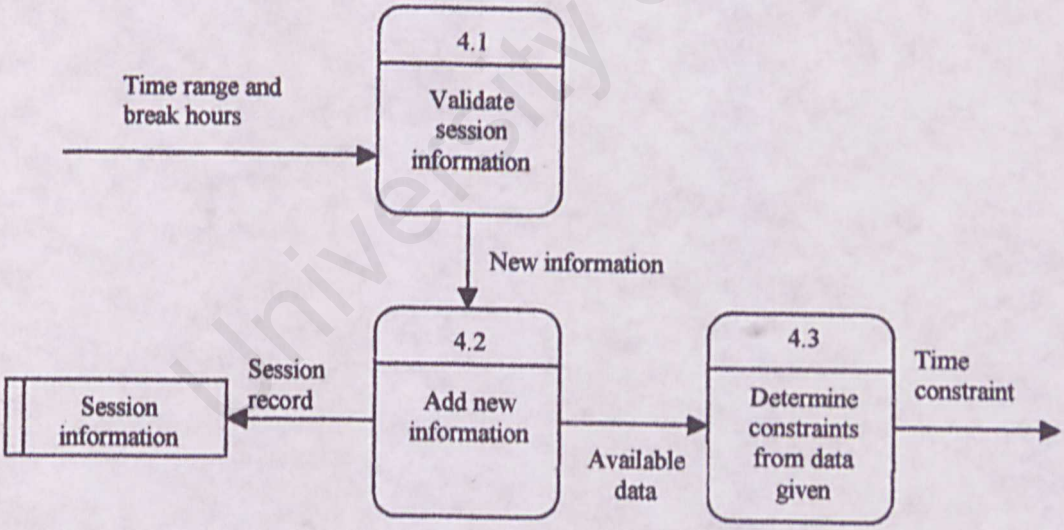


Figure 3.10 Diagram of process on the session description.

Chapter 4

System Design

4.1 Automated Scheduling System Design

This scheduling system design is based on the rule-based system. The system users should be able to generate a schedule from the data provided to the system. These data include course information, lecturer information, room information and schedulable time information as shown in Figure 4.1.

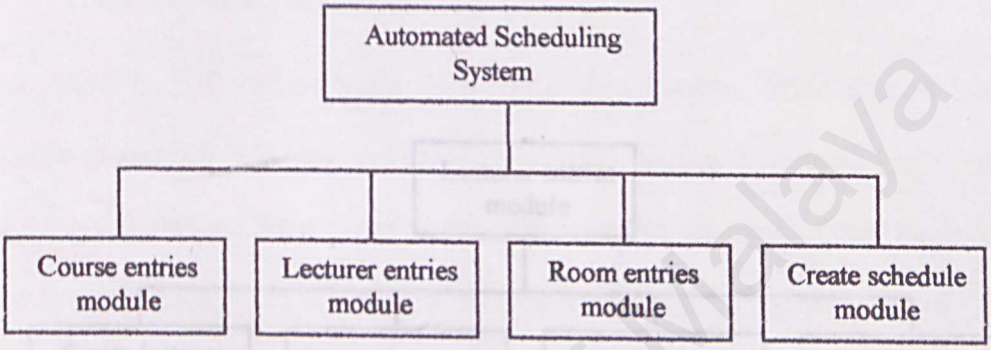


Figure 4.1 Major system design for automated scheduling system.

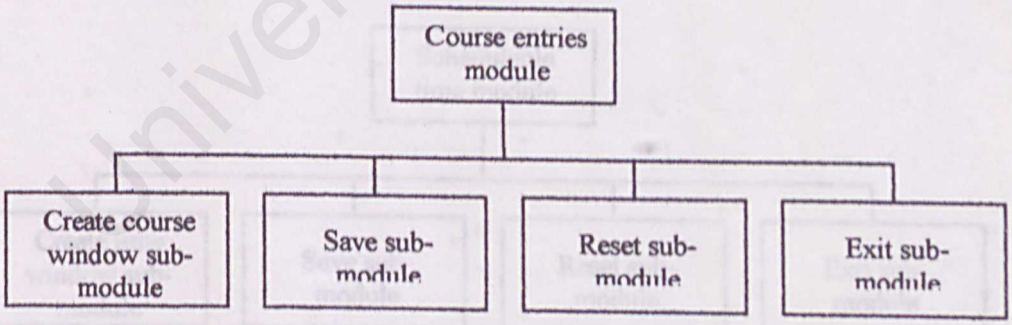


Figure 4.2 Sub-modules in course entries modules.

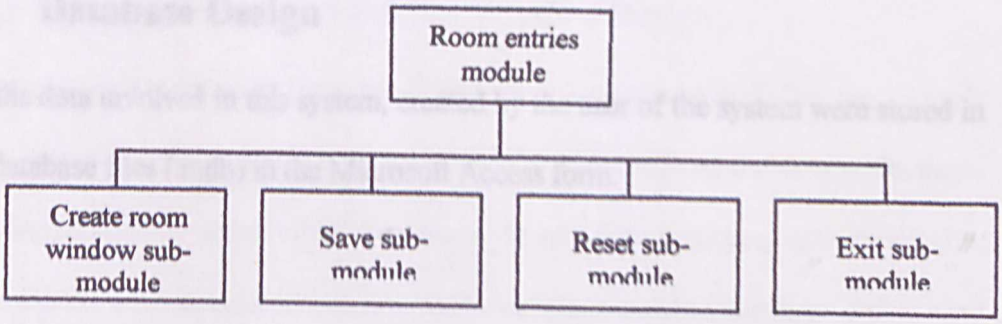


Figure 4.3 Sub-modules in room entries modules.

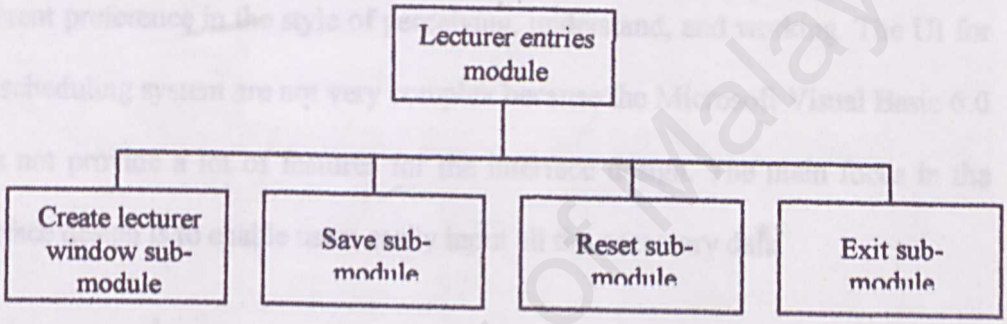


Figure 4.4 Sub-modules in lecturer entries modules.

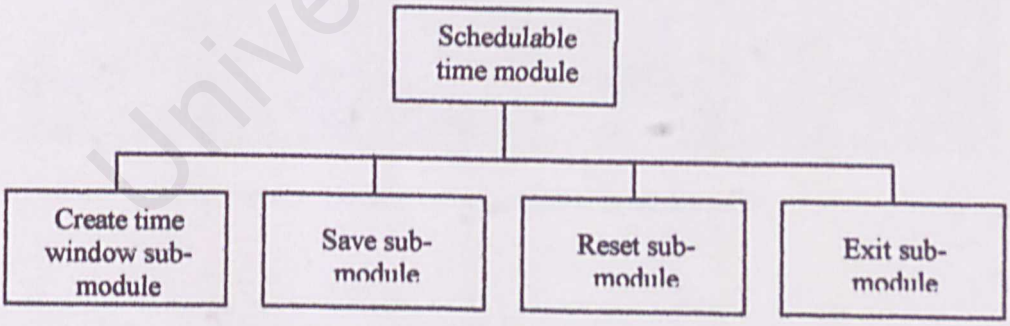


Figure 4.5 Sub-modules in schedulable time modules.

4.2 Database Design

All the data involved in this system, created by the user of the system were stored in the database files (.mdb) in the Microsoft Access form.

4.3 User Interface Design

User interfaces (UI) can be tricky things to design, because different users have different preference in the style of perceiving, understand, and working. The UI for this scheduling system are not very complex because the Microsoft Visual Basic 6.0 does not provide a lot of features for the interface design. The main focus in the interface design is to enable users easily input all the necessary data.

4.3.1 Course Information Entries Window Design

This window is to let users input the information for a course including the course's code, course's name, credit hours, number of students taking the course, lecturers attached, discipline and level of the course. In addition, questions such as whether the course is a lab course, a core or elective course, a combined course also needed by the system.

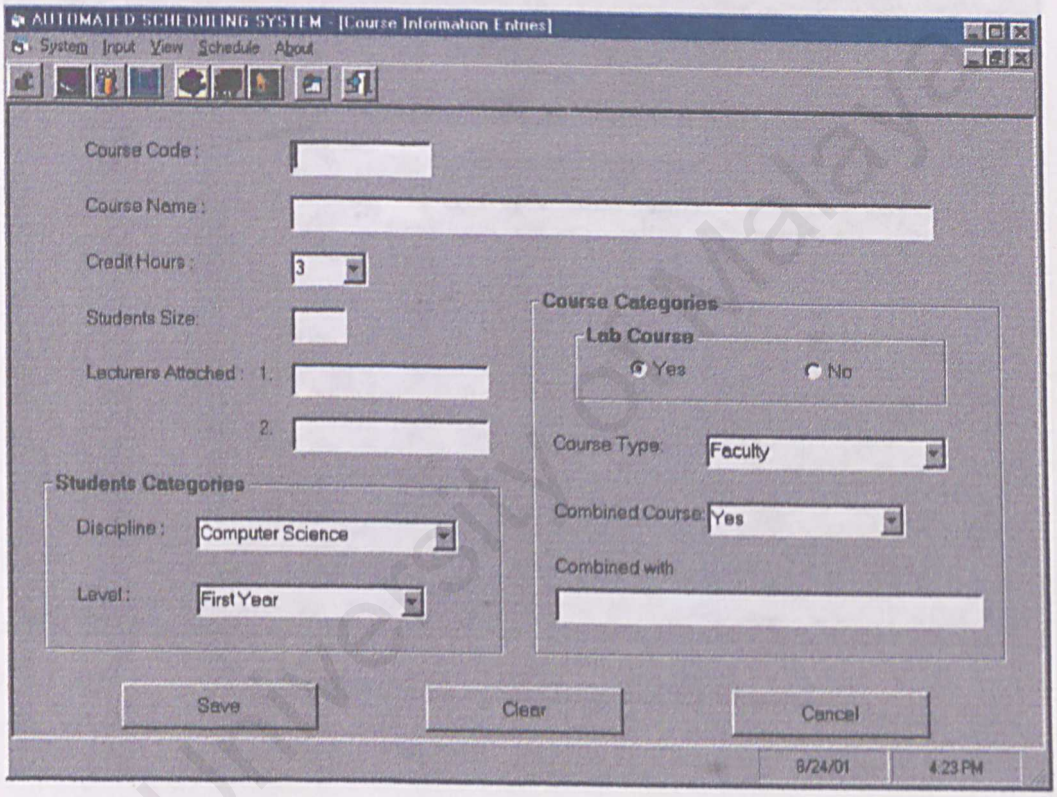


Figure 4.6 Course information entries window.

4.3.2 Room Information Entries Window Design

This window is to let users input the information on the rooms available in scheduling the course. Information such as room's code, name, capacity, and type of room is needed for the rooms.

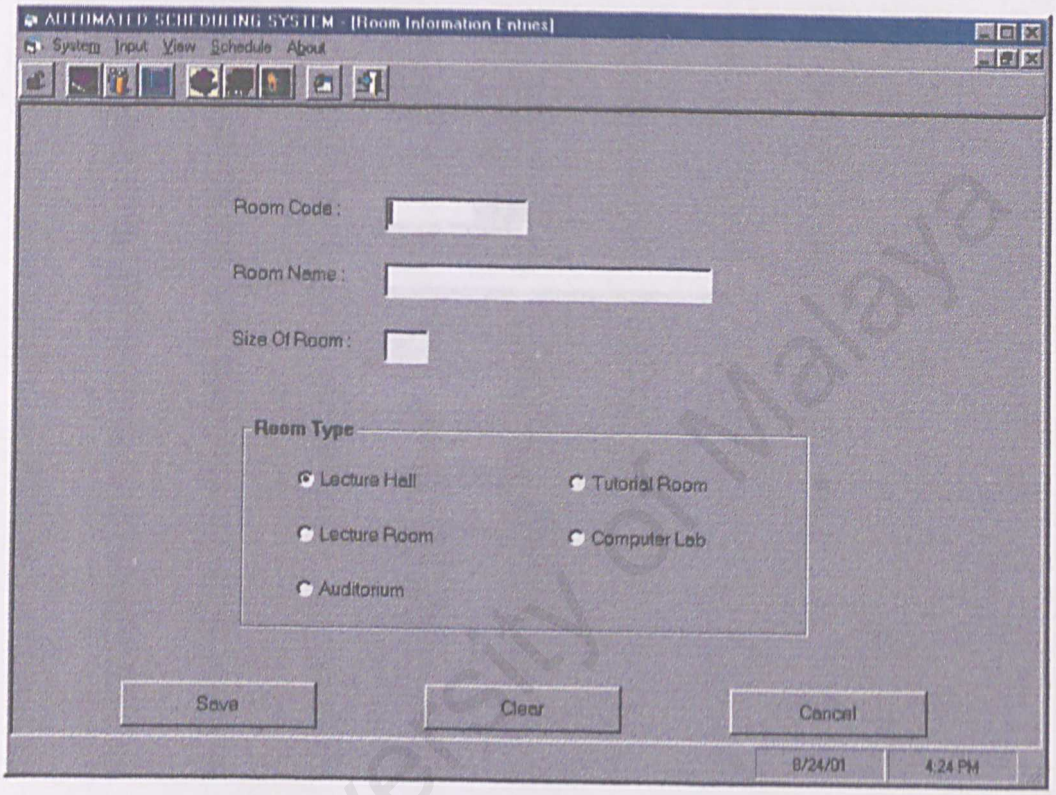


Figure 4.7 Room information entries window.

4.3.3 Lecturer Information Entries Window Design

This window allows users to key in the lecturers' information such as their codes and names. Then the courses lectured by the lecturers in that semester must be specifying if there is any.

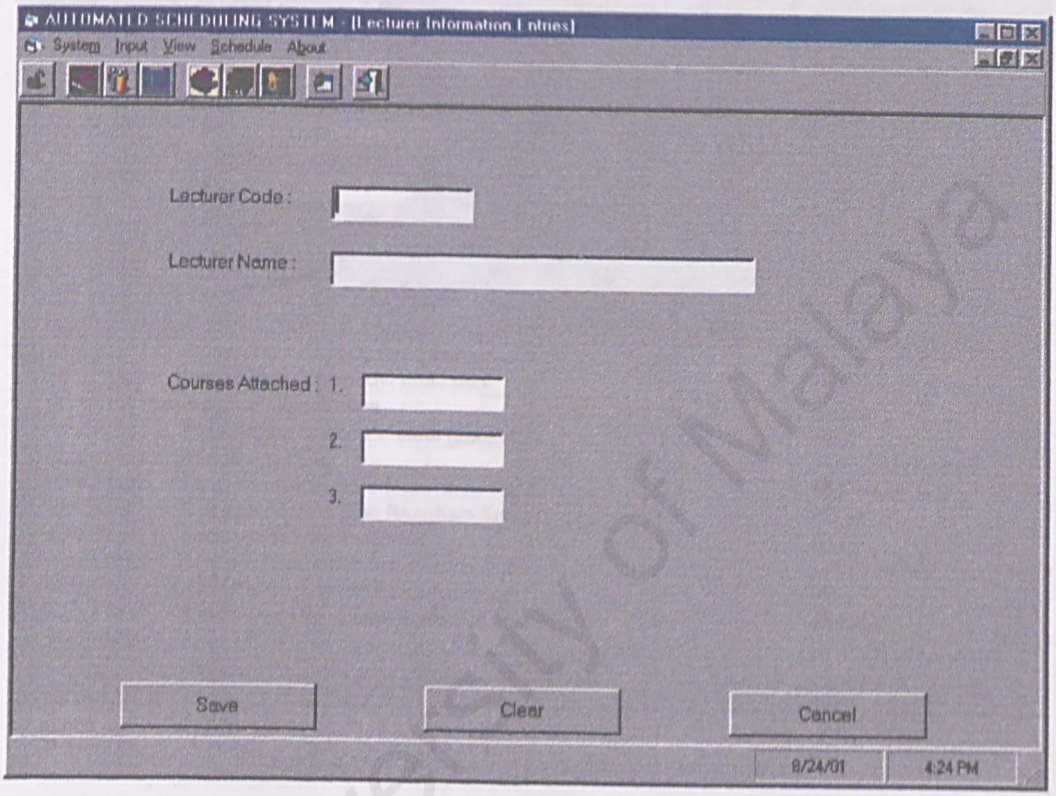


Figure 4.8 Lecturer information entries window.

4.3.4 View Courses Information Window Design

Users can view the available courses by enter data in this form including the discipline and the level of the courses.

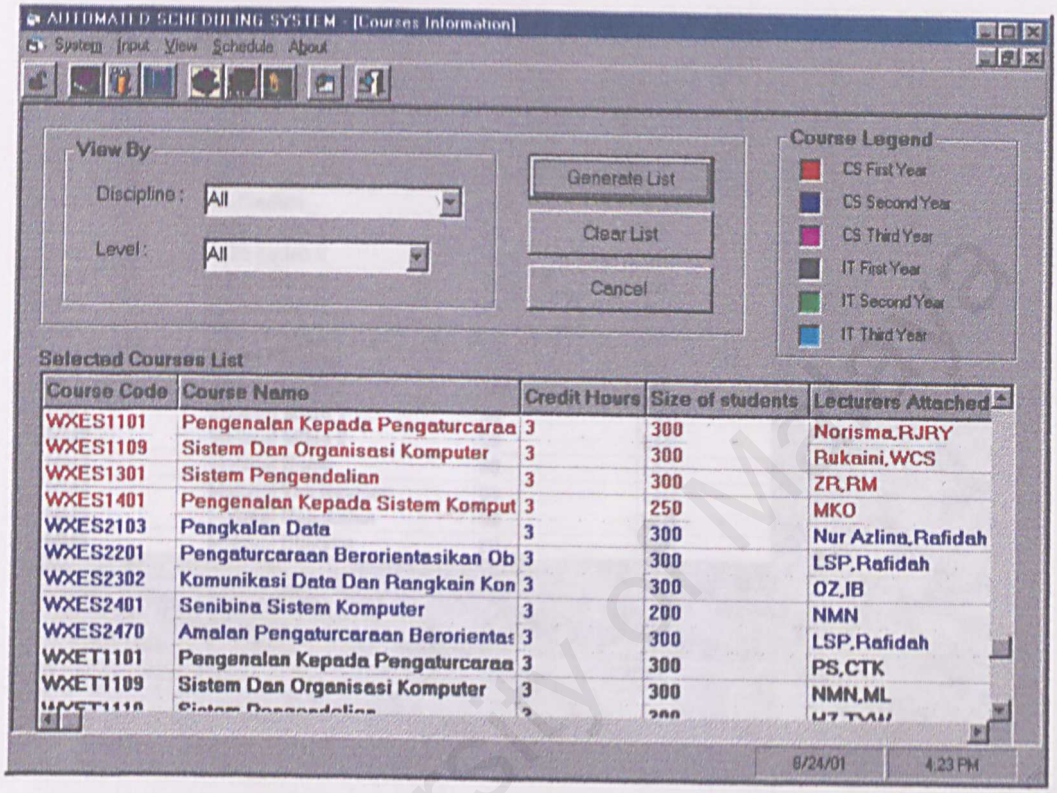


Figure 4.9 View courses information window.

4.3.5 View Rooms Information Window Design

Users can view the available rooms for the uses of lectures or computer labs on that semester.

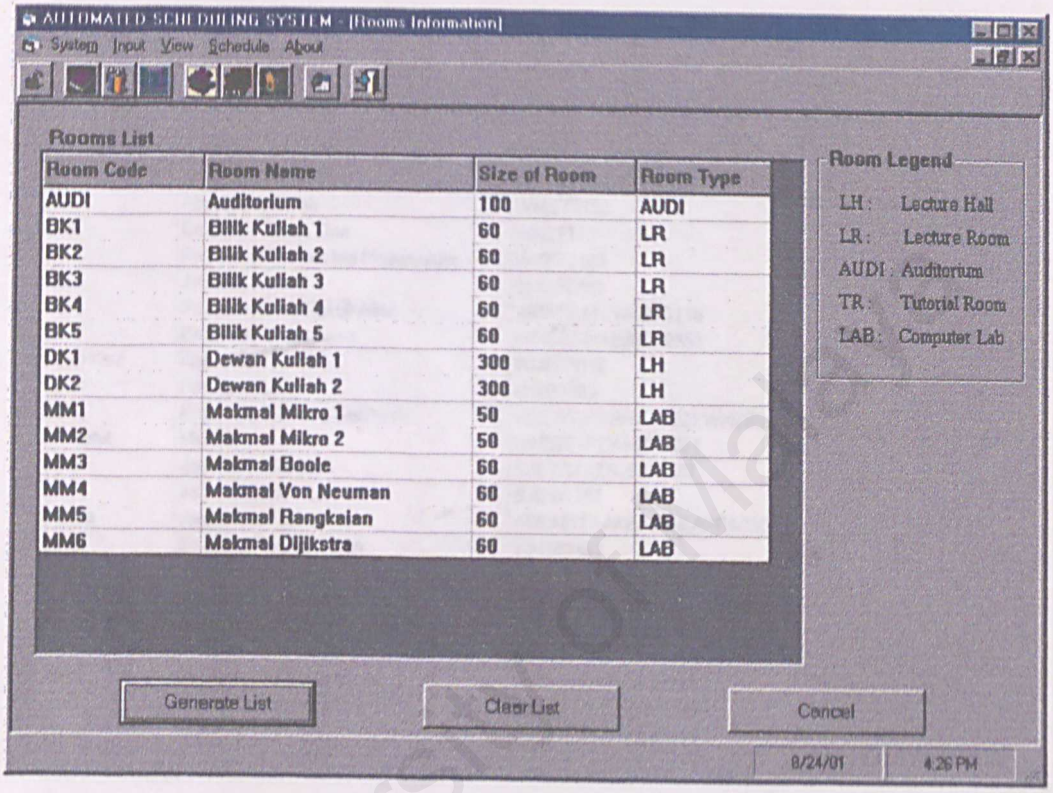


Figure 4.10 View rooms information window.

4.3.6 View Lecturers Information Window Design

Users can view the available lecturers with the courses they are taken in that semester.

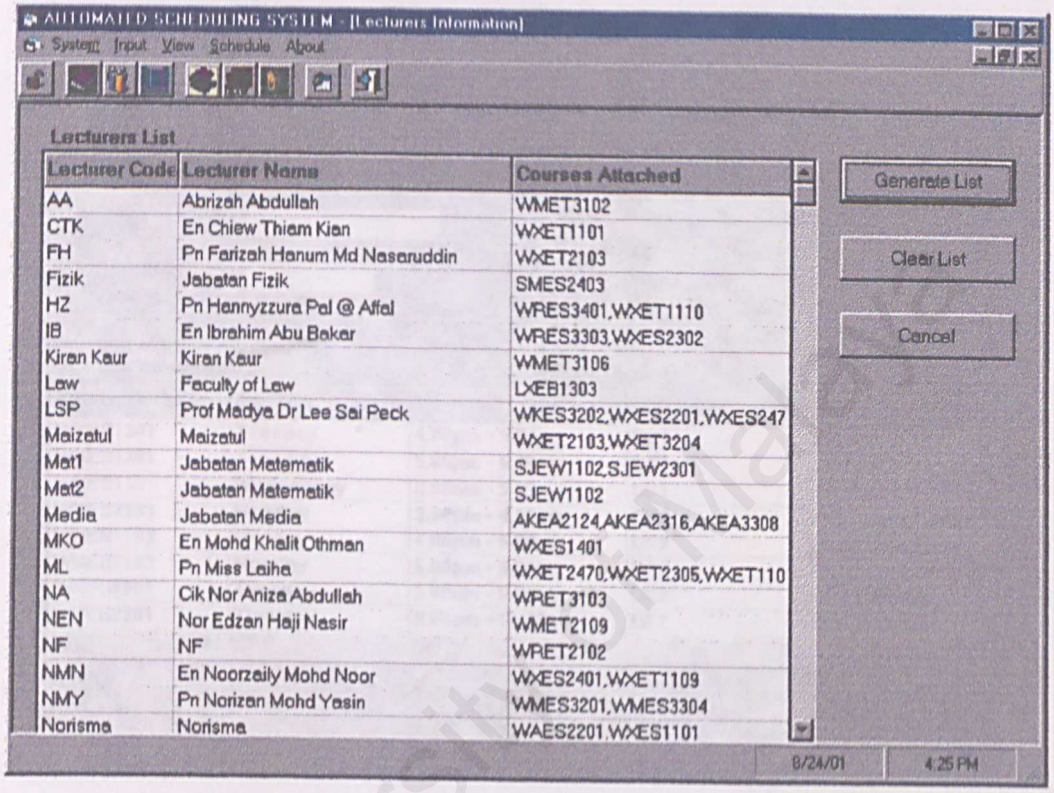


Figure 4.11 View lecturers information window.

4.3.7 View Students' Schedule Window Design

Users can view the students' schedule filtered by students' discipline and level in that semester.

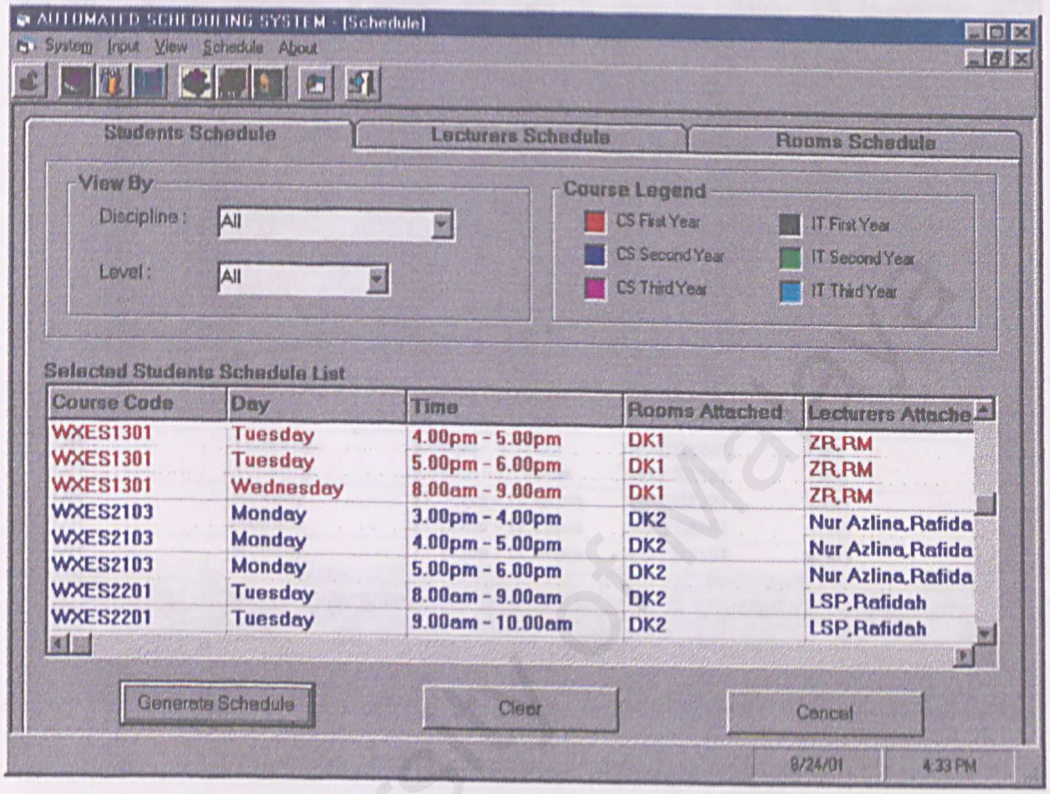


Figure 4.12 View students' schedule window.

4.3.8 View Lecturers' Schedule Window Design

Users can view the lecturers' schedule filtered lecturer code in that semester.

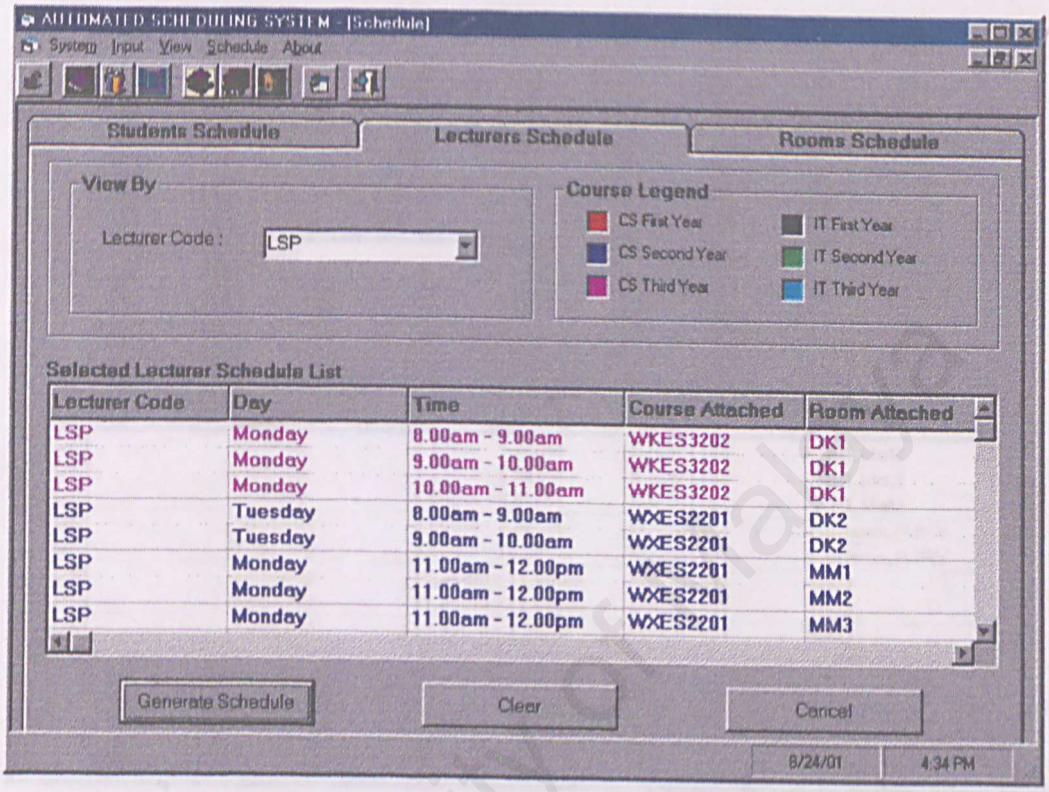


Figure 4.13 View lecturers' schedule window.

4.3.9 View Rooms' Schedule Window Design

Users can view the rooms' schedule filtered by room code in that semester.

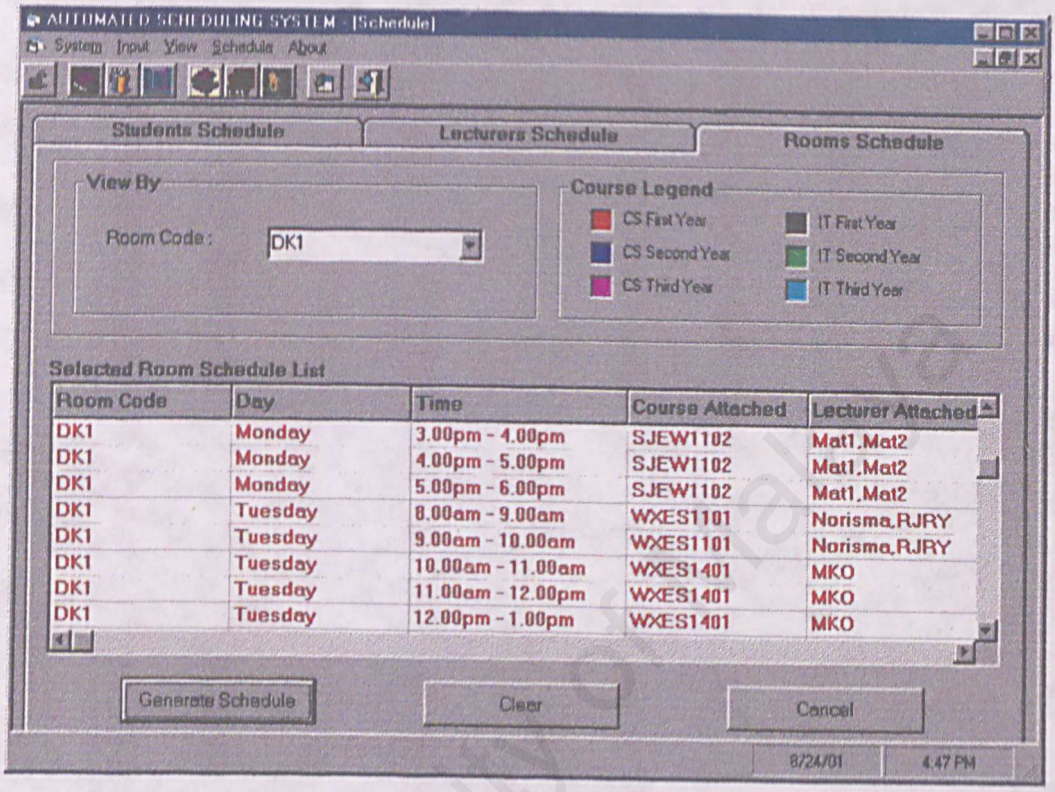


Figure 4.14 View rooms' schedule window.

Chapter 5

System Implementation and Testing

5.1 Developing Environment

Developing environment may have a certain impact on the development of a system. In order to speed up the system development, the suitable hardware and software should be used.

5.1.1 Hardware

Hardware used for developing this automated scheduling system are listed as below:

- 350 MHz Pentium II Processor
- 64 MB RAM
- 3.2 GB Hard Disk Drive
- 14" color SVGA monitor
- CD-ROM Drive
- Floppy Drive
- Mouse
- Standard keyboard
- Canon Bubble Jet Printer 210-S

5.1.2 Software

Table 5.1 shows the software that have used in this project.

Software	Remarks
Visual Basic Version 6.0	This software was used in developing the system. All coding of the program involved is done with this software.
Microsoft Office	This software was used to prepare the proposal and report. It was also used to draw the entire diagram found in this report.
Capture Express 2000 Version 1.1	This software was used to capture the image of the screen of the application for the use of this report.

Table 5.1 Software used and descriptions.

5.2 System Development

Languages used to develop the automated scheduling system for FCSIT is Visual Basic 6.0. Visual Basic 6.0 is a user-friendly programming language. It is used to create all the user interfaces in the system.

In this system, Visual Basic is used to define the relationship between the objects involve in scheduling such as courses, students, lecturers, rooms, time and many others. Each of these objects may have a very complicated relationship between each other.

5.3 Database Development

Microsoft Access is used as the databases for the system. The database in the system is in mdb files form. All data can be added to the database directly as the application system is running or through Microsoft Access.

5.4 Testing

Software testing refers to verification and validation of the program coded to solve the problems. Verification involves ensuring that the characteristics of a good design are incorporated into the program and the system is actually operates the way it was expected to be. Validation refers to execution of the program and system meets the requirements.

Therefore, the major focus in testing is to find the faults that might occurs in the application program. A test is successful only when a fault is discovered or when a failure has been come across. Testing actually involves the iteration of the process of fault identification and fault correction or removal.

In developing large system, testing usually involves several stages. These stages are module testing, integration testing, function testing, performance testing, acceptance testing and installation testing.

5.4.1 Module Testing

Module testing or unit testing is to verify that the small unit functions properly with the types of input expected from the design. Unit testing has been carried out under a controlled environment where a predetermined set of data has been provided to the modules. Unit testing has been carried out to observe what input and output actions and the data produced [18].

In module testing, each of the sub-modules in the course entries modules, room entries modules, lecturers entries modules, schedulable time modules, and create

schedule modules are tested separately. Ten each of the modules is tested for the creation of the user interface, the input data handling and output to data files, reset and exit from the module to make sure those modules does not actually has been design. Test cases have been developed to show that the input is properly converted to the desired output.

5.4.2 Integration Testing

After the collections of the modules have been unit-tested, the next is to ensure that the interfaces among the components are defined and handled properly. Integration testing is the process of verifying that the system modules work together as describes in the system and program specification [18].

In this stage, all the individual sub-modules and modules are integrated and tested to ensure that the interfaces between the sub-modules and modules, modules and the main program are handled properly. Here all the small modules that are tested isolated before they are combined into a big program in the intelligent system and tested together.

The testing approach has been apply in the integration testing is the bottom-up integration. Each component at the lowest level of the system hierarchy is tested individually first. For the intelligent system, each sub-module is tested individually first, then the modules are tested. Finally, after the integration into a big program, the main program is test to ensure that system works correctly.

5.4.3 Function Testing

After the integration test, function test is carried out to assure that the system has the desired functionality. Function test will evaluate the system to determine if the functions described by the requirement specifications are actually performed by the integrated system [18].

The intelligent system in this stage is test to determine that it will schedule the courses and try to fix the courses to the schedule with the time and venue. Before, the system do so, it should be able to check all the constraints to make sure there is not any conflict to the schedule.

5.4.4 Other Testing

The other testing should be carrying out after the functional testing are performance test, acceptance test and installation test.

Performance test compares the integrated components with the nonfunctional system requirements such as security, accuracy, speed and reliability. The users of the system to assure them that the system they need was the system that was built for them run acceptance test. Installation test allows users to exercise the system functions and document additional problems that result from being at the actual site [18].

These tests are not actually tested for the automated scheduling system for FCSIT because these tests should be carry out by the users of the system which may be the administration in the office.

Chapter 6

System Evaluation and Conclusion

6.1 System Evaluation

The system should be evaluated in order to know their effectiveness and efficiency of the system implementation. With system evaluation, the system can be improving by looking at the system limitations.

6.1.1 System Strength

- **Simple interface**

  The user interfaces in this system are very simple. It is very easy to train the users on how to used the system.

- **Save time**

  The system has the capabilities of reducing the time and effort for generating the schedule.

- **Feasible schedule**

  The system produced schedule that was feasible and with sufficient quality to be used.

- **Speed**

  The system speeds up the generation of course schedule. The generation of schedule with the system is very fast and users do not need to spend much time on clerical work involve in scheduling. The schedule can be easily got from the data files that stored the schedule.

- **Dynamic and interactive content**

  Because the users provide all the data during the execution of the application, the content of the data are dynamic and interactive where the users can have their own codes and others for their data.

- **Automated scheduling process**

  The scheduling process is automated given the necessary input parameters such as the courses, lecturers, time and rooms.

6.1.2 System Limitations

- **No error checking**

  There are no validations on user inputs into the data files. User may enter something that may affect the output of the system.

- **Limitation to the input**

  Input in the system should not include any characters that are not alphanumeric. Course structure can only take in one number.

- **Constraints integrated in the coding**

  The constraints on the course scheduling are integrated into the coding of the program and this limit the program ability to handle other constraints.

6.2 Future Enhancements

Due to the limitations found in system, there is many improvements can be done as future enhancements on this system.

- **Web based**

  The application can be turn into a web based application in order to enable user to access the application from anywhere and the user only need a browser.

- **Updateable record**

  The record in the data files should be reusable and user can pick from the old data if they want to. This safe the users time from key in all the data all over again each time he uses the system.

- **Additional query functions**

  Application should provide other query functions than scheduling only. These functions include listing out the courses, lecturers, venues, certain type of courses and many more.

- **Apply the algorithms**

  The AI approach such as Genetic Algorithms, Memetic Algorithms, Simulated Annealing, Tabu Search, or Constraint Logic Programming can be integrated into the application to improve the scheduling process.

- **Flexible constraints**

  The users can provide any constraint to the application and the system should be able to process these constraints without integrated them into the codes for the system.

- **Object oriented**

  The concept of object oriented and reusable code should be given attention because this concept is very important to a better programming approach and the object-oriented concept is developing in the AI area.

6.3 Problem Encountered

There are many problems encounter during the system development. Most of these problems are due to the limitation in the language itself. Visual Basic is a very primitive language and therefore it cannot support certain type of functions or it needs to have longer code to execute the function.

Time given to develop the system is not long enough and yet many further improvements on the system cannot be done.

Debugging tool in Visual Basic is not powerful enough where the changes of the term and variables cannot be seen clearly like other language. This made the testing of the Visual Basic program time consuming.

6.4 Knowledge Gained

I'm provided with the opportunity to use the Visual Basic language to develop an AI application with this thesis. I had an opportunity to expose myself to this programming language.

In the process of completing this thesis, I learned more about the AI concepts such as rule-based, forward chaining, backward chaining, heuristic approach and also a little bit on Genetic Algorithms, Memetic Algorithms, Tabu Search and others.

6.5 Conclusions

The objective of the automated scheduling system is to provide an application that automated the process of scheduling in FCSIT is partially achieved with this application system.

The system actually can generate the schedule for the users but the schedule created is still not the best schedule and users are not given choice to choose from schedules. In addition to that, there are still many limitations found this intelligent system need to be improved in the future.

References

[1] A. Apte, A. Jayasuriya, J. Kennington, I. Krass, R. Mohamed, S. Sorensen and J. Whitler, "Class Scheduling Algorithms for Navy Training Schools", Naval Research Logistics: An International Journal, Vol. 45 No. 6, September 1998, 533-551.

[2] A. Bundy, "Artificial Intelligence Techniques: A Comprehensive Catalogue", 4th Revised Ed., Germany, Springer-Verlag Berlin Heidelberg, 1997.

[3] A. Schaerf, "A Survey of Automated Timetabling", Centrum voor Wiskund en Infirmatica (CWI) report CS-R9567, The Netherlands, 1995.

[4] A. Schaerf, "Tabu Search Techniques for Large High-School Timetabling Problems", Centrum voor Wiskund en Infirmatica (CWI) report CS-R9611, The Netherlands, 1996.

[5] A. Wevell, "Word Power Dictionary", 1st Ed, The Reader's Digest Association South Africa (Pty) Ltd, 1996.

[6] D. Corne, P. Ross and H.L. Fang, "Evolutionary Timetabling: Practice, Prospects and Work in Progress", UK Planning & Scheduling SIG Workshop, Strathclyde, September 1994.

[7] E.K. Burke, D.G. Elliman and R.F. Weare, "A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems", The 6th International Conference on Genetic Algorithms, Pittsburgh USA, 15th-19th July 1995, 605-610.

[8] E.K. Burke, D.G. Elliman and R.F. Weare, "The Automation of the Timetabling Process in Higher Education", Journal of Educational Technology Systems, Vol. 23(4), 1995, 257-266.

[9] E.K. Burke, J.P. Newall and R.F. Weare, "A Memetic Algorithm for University Exam Timetabling", 1st International Conference on the Practice and Theory of Automated Timetabling, Napier University, Edinburgh, August 29th-September 1st 1995.

[10] E.K. Burke, K.S. Jackson, J.H. Kingston and R.F. Weare, "Automated University Timetabling: The State of the Art", The Computer Journal, London, Vol. 40 No. 9, 1997, 565-571.

[11] G. Lajos, "Complete University Modular Timetabling using Constraints Logic Programming", 1995.

[12] G.F. Luger and W.A. Stubblefield, "Artificial Intelligence Structure and Strategies for Complex Problem Solving", 2nd Ed, California, The Benjamin/Cummings Publishing Company, 1993.

[13] J. Durkin, "Expert System Design and Development", New York, Macmillan Publishing Company, 1994.

[14] L.E. Frenzer, Jr., "Crash Course in Artificial Intelligence and Expert Systems", 1st Ed., USA, Howard W. Sams & Co..

[15] P. Lucas and V.D.G. Linda, "Principles of Expert Systems", 1st printed, England, Addison-Wesley Publishing Company, 1991.

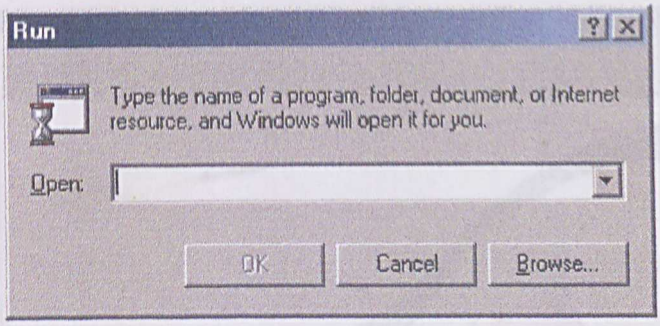
[16] P. Ross, D. Corne and H.L. Fang, "Successful Lecture Timetabling with Evolutionary Algorithms", ECAI'94 Workshop W17: Applied Genetic and other Evolutionary Algorithms, 1994.

[17] S. Deris, S.Omatu, H. Ohta and P. Saad, "Incorporating constraints propagation in genetic algorithm for university timetable planning", Engineering Application of Artificial Intelligence: The International Journal of Intelligent Real-time Automation, Vol. 12 No. 3, June 1999, 241-253.

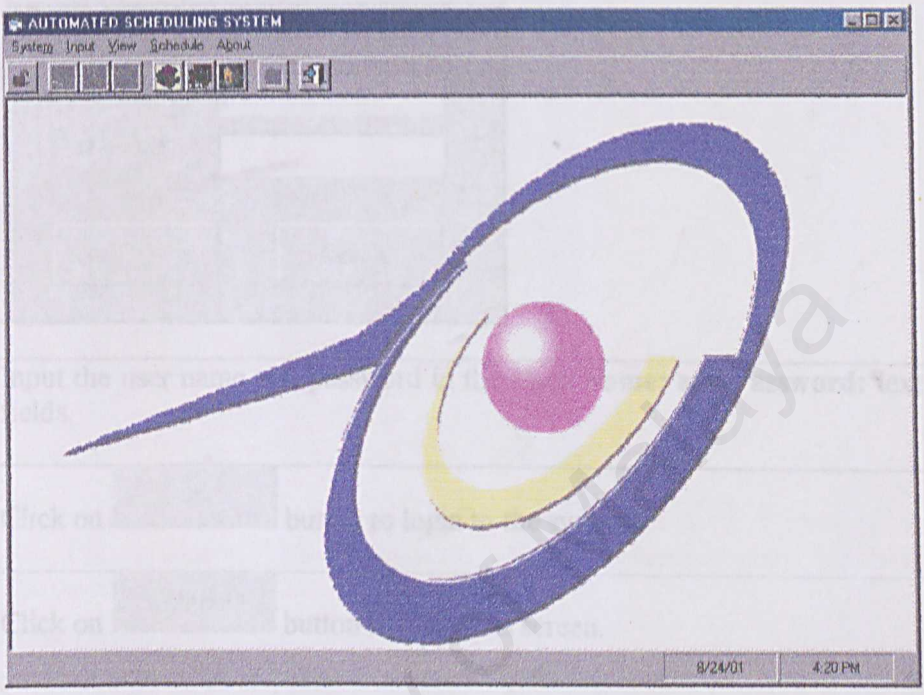
[18] S.L. Pfleeger, "Software Engineering Theory and Practice", Washington, Prentice-Hall International, 1998.

User Manual

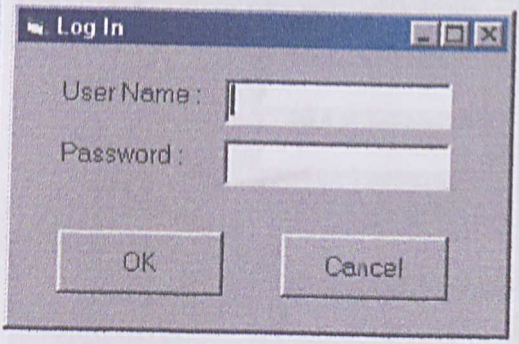
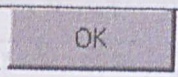
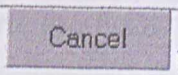
Installing Visual Basic 6.0

Step	Action
1	Insert the Visual Basic Version 6.0 CD into your CD-ROM drive.
2	Click on **Start | Run** on the Start Menu. The following window appears. 
3	Type **x:\setup.exe** (where x is your CD-Rom drive).
4	Following the step with clicking on **Next** to install Visual Basic.
5	Click **Finish** to complete the installation and restart your computer.

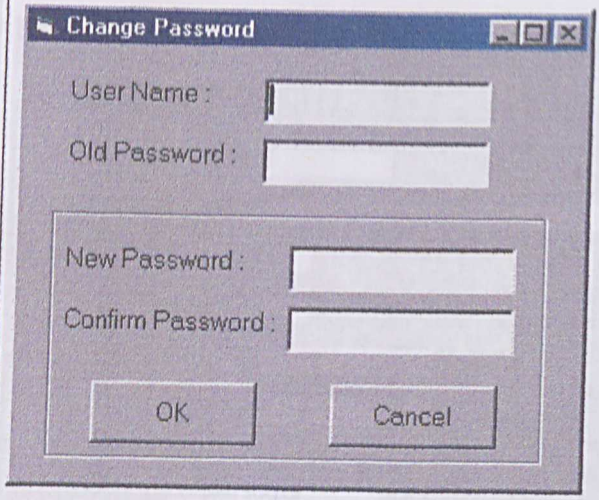
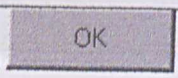
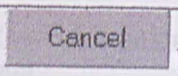
Running Automated Scheduling System for FCSIT

Step	Action
1	Double click on  icon on the desktop. The following window appears. 

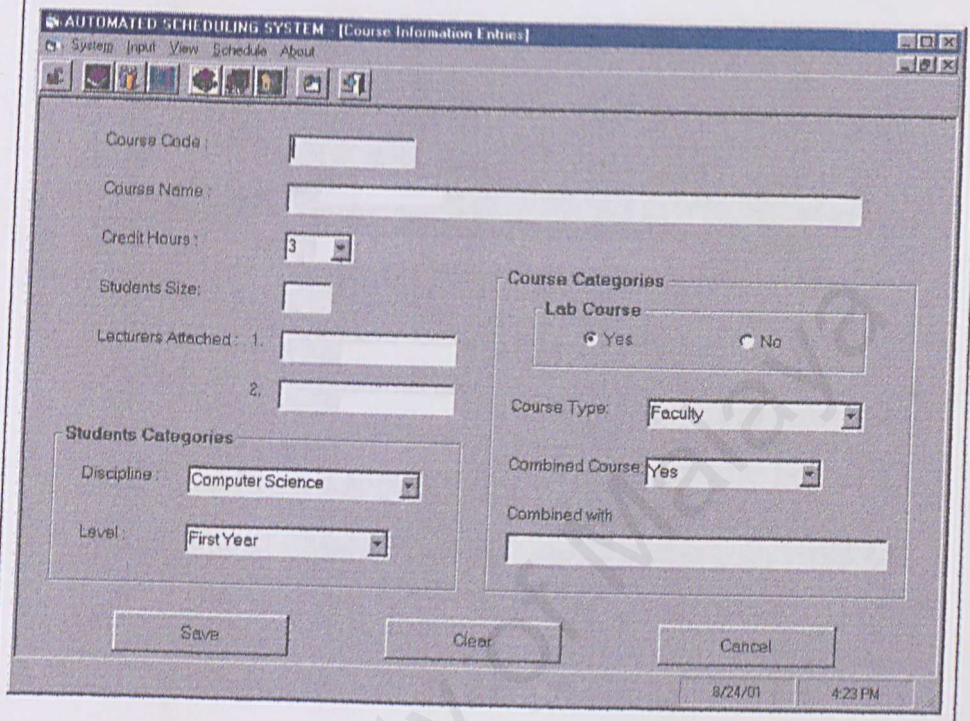
Login Window

Step	Action
1	Click on **System | Login** or  button on the toolbar. The following window appears. 
2	Input the user name and password in the **User Name:** and **Password:** text fields.
3	Click on  button to login to the system.
4	Click on  button to close this screen.

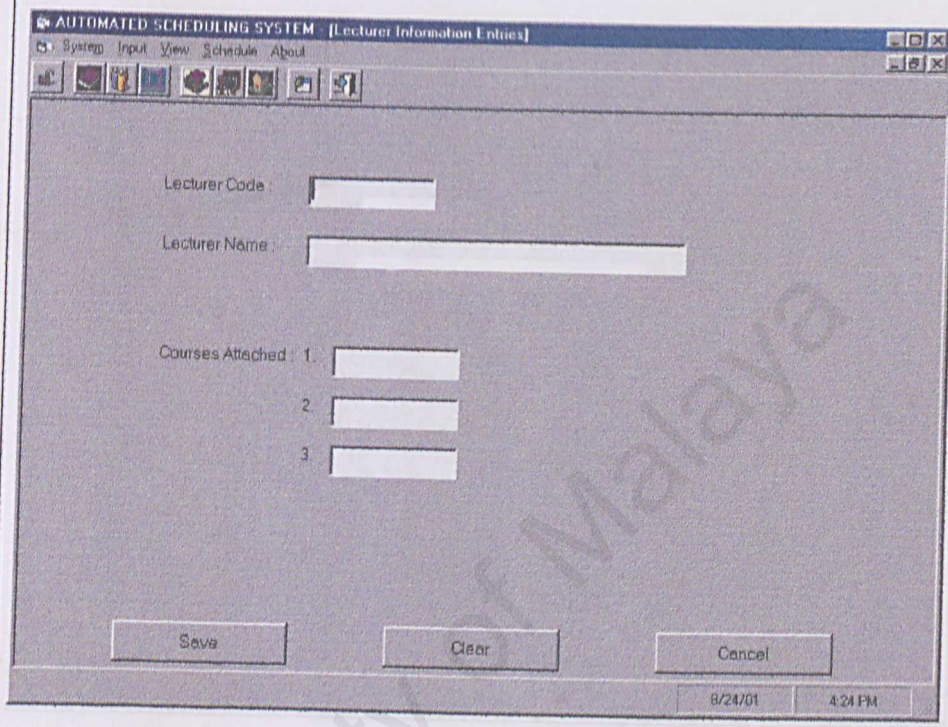
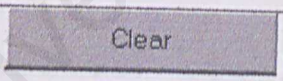
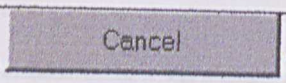
Change Password Window

Step	Action
1	Click on **System | Change Password** on the toolbar. The following window appears. 
2	Input the user name, old password, new password and confirm password in the **User Name:**, **Old Password:**, **New Password:** and **Confirm Password:** text fields.
3	Click on  button to change your password.
4	Click on  button to close this screen.

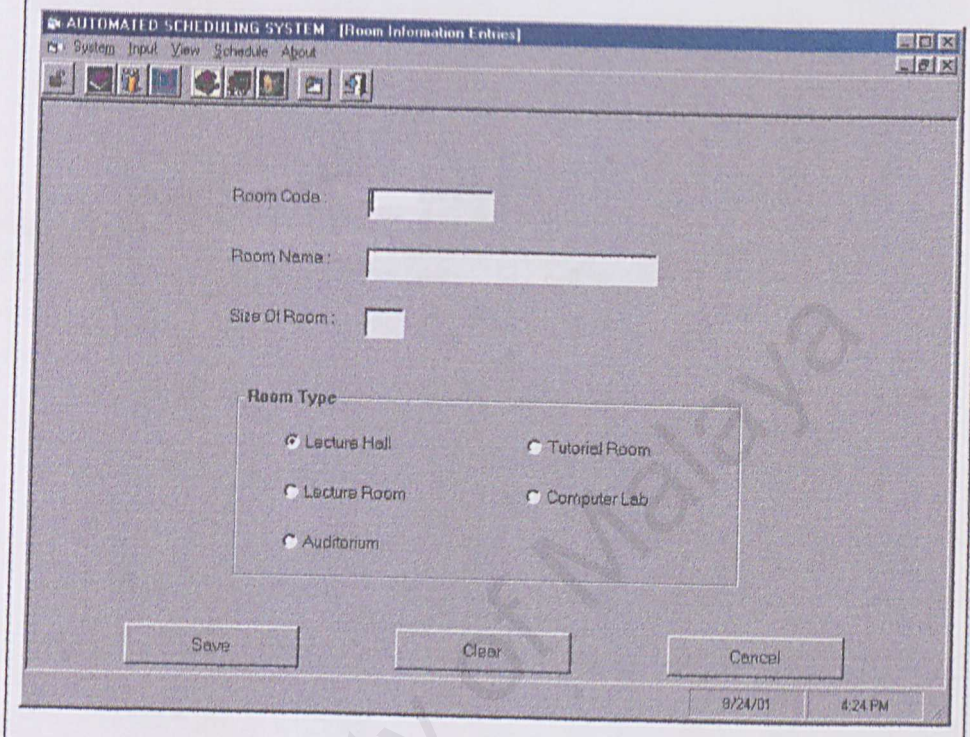
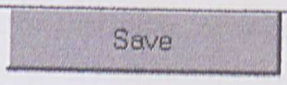
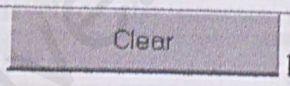
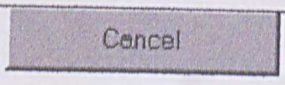
Course Information Entries Window

Step	Action
1	Click on **Input | Course Information** or the  button on the toolbar. The following window appears. 
2	Input all the information and click on the Save button to save the data.
3	Click on the Clear button to clear all the information on the current window.
4	Click on the Cancel button to exit from this screen.

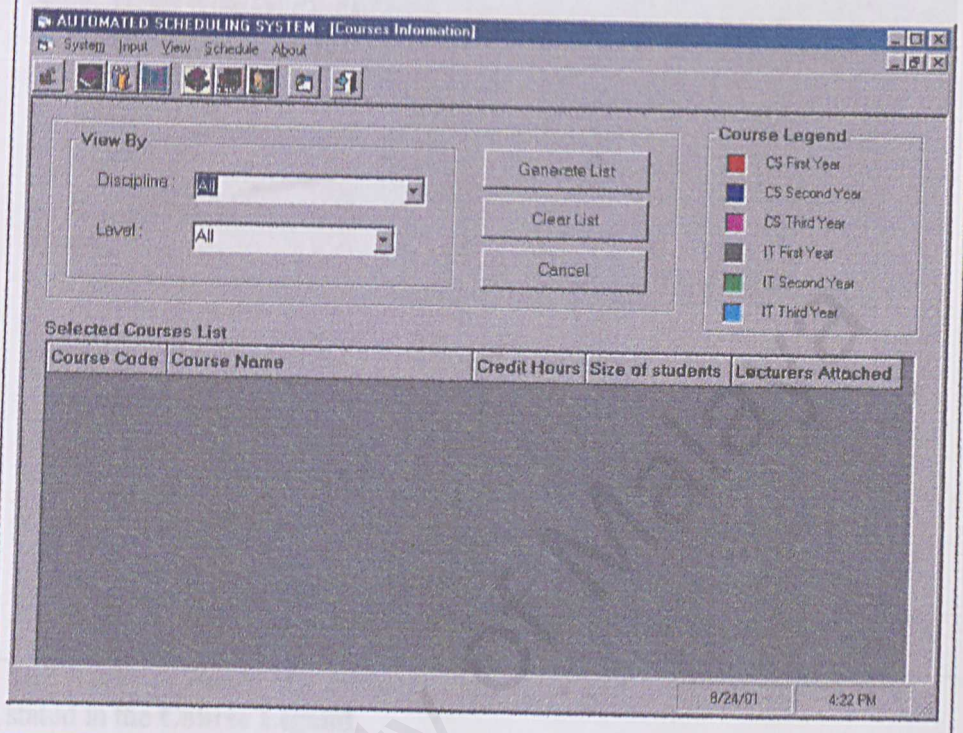
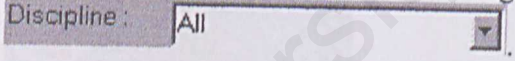
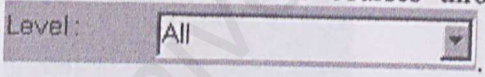
Lecturer Information Entries Window

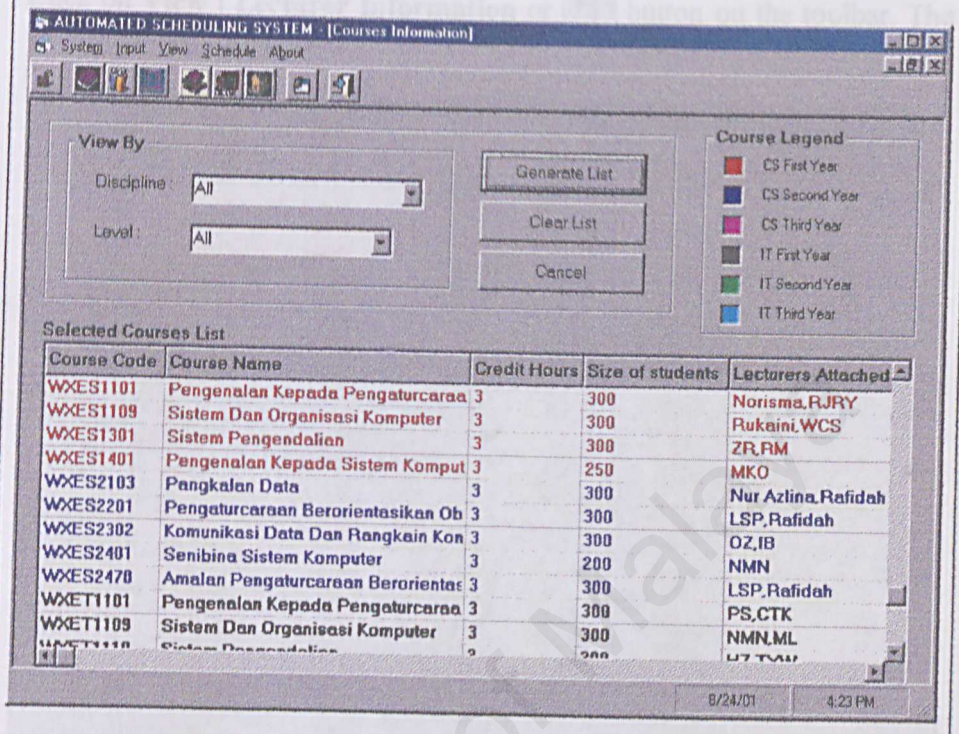
Step	Action
1	Click on **Input | Lecturer Information** or the  button on the toolbar. The following window appears. 
2	Input all the information and click on the  button to save the data.
3	Click on the  button to clear all the information on the current window.
4	Click on the  button to exit from this screen.

Room Information Entries Window

Step	Action
1	Click on **Input | Room Information** or the  button on the toolbar. The following window appears. 
2	Input all the information and click on the  button to save the data.
3	Click on the  button to clear all the information on the current window.
4	Click on the  button to exit from this screen.

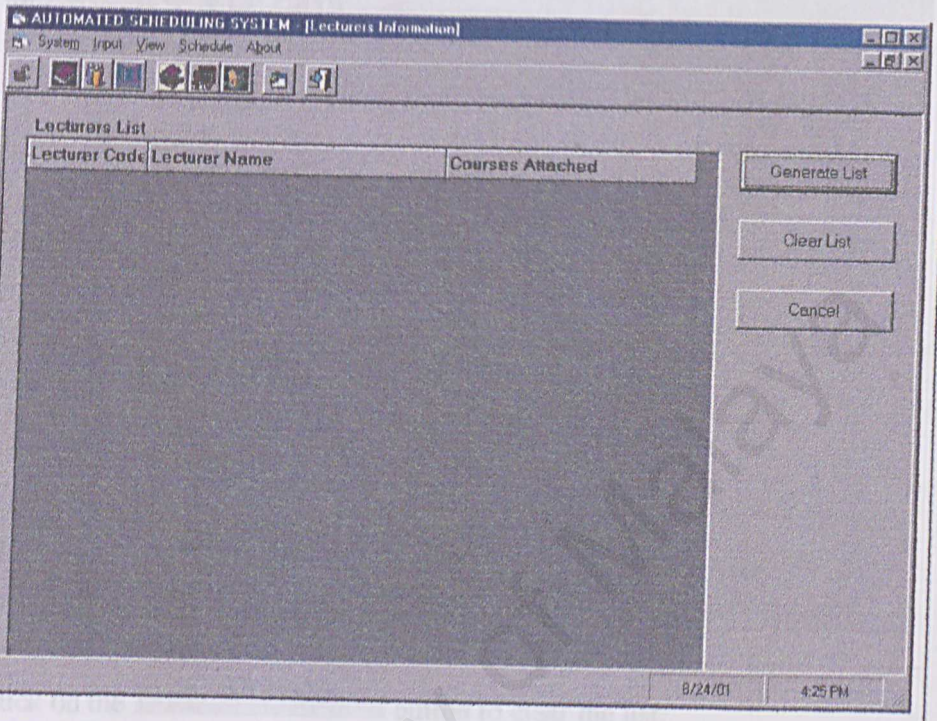
View Courses Information Window

Step	Action
1	Click on **View | Course Information** or  button on the toolbar. The following window appears. 
2	Select the discipline of the courses through the **Discipline:** drop down menu 
3	Select the level of the courses through the **Level:** drop down menu 

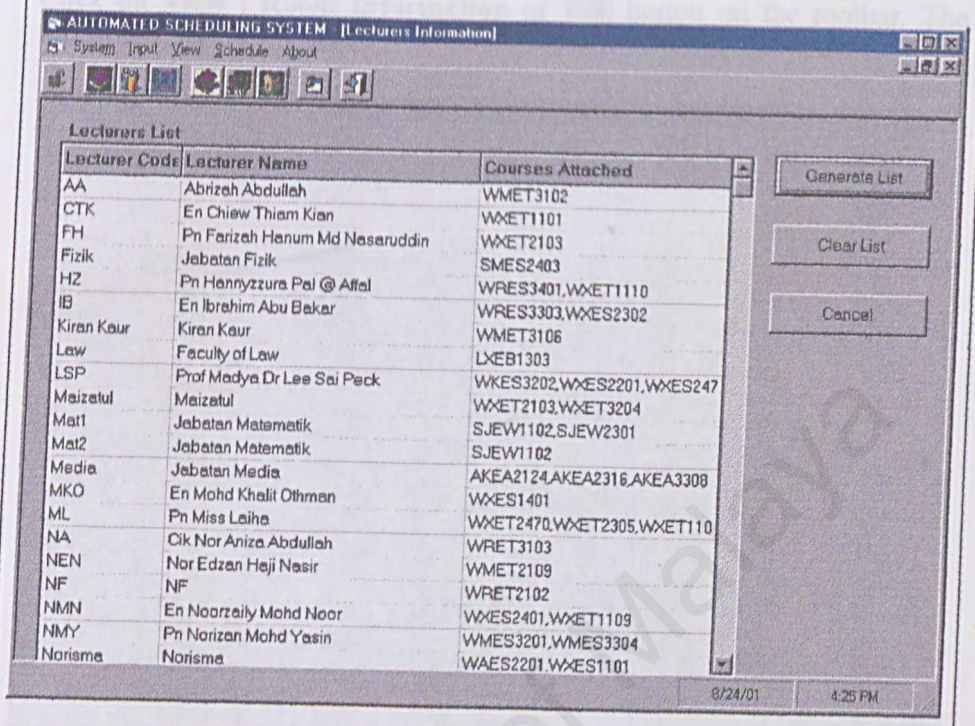
Step	Action
4	Click on the Generate List button to generate a selected courses list. The following screen appears. 
5	The colors represent the different disciplines and levels of the courses as stated in the **Course Legend**. 
6	Click on the Clear List button to clear the selected list.
7	Click on the Cancel button to exit from this screen.

View Lecturers Information Window

Step	Action
1	Click on **View | Lecturer Information** or  button on the toolbar. The following window appears.

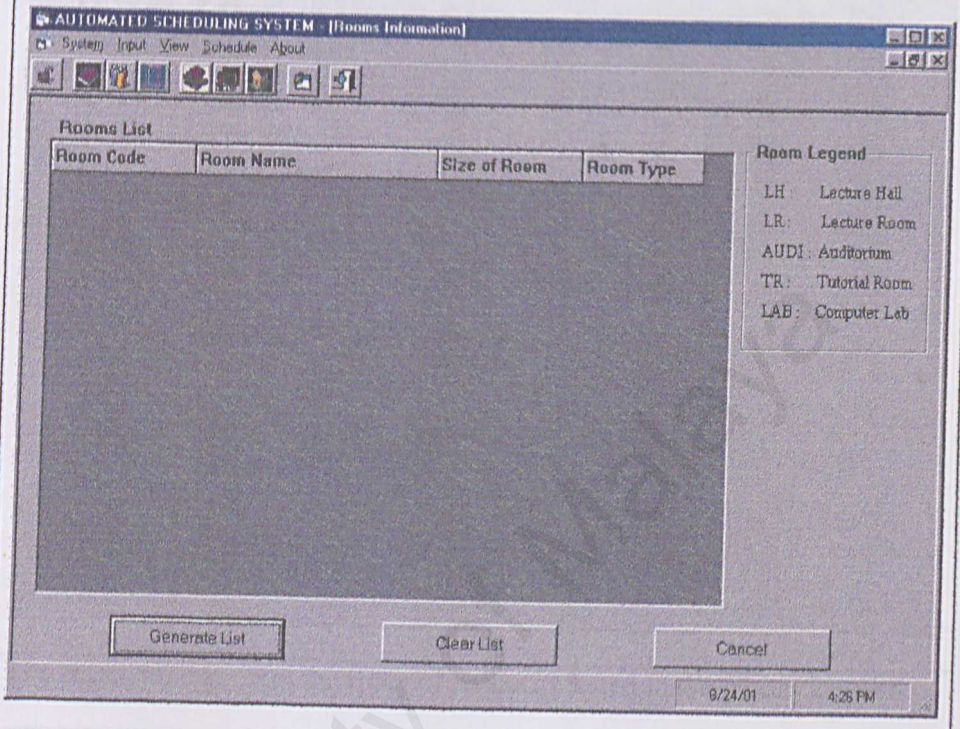


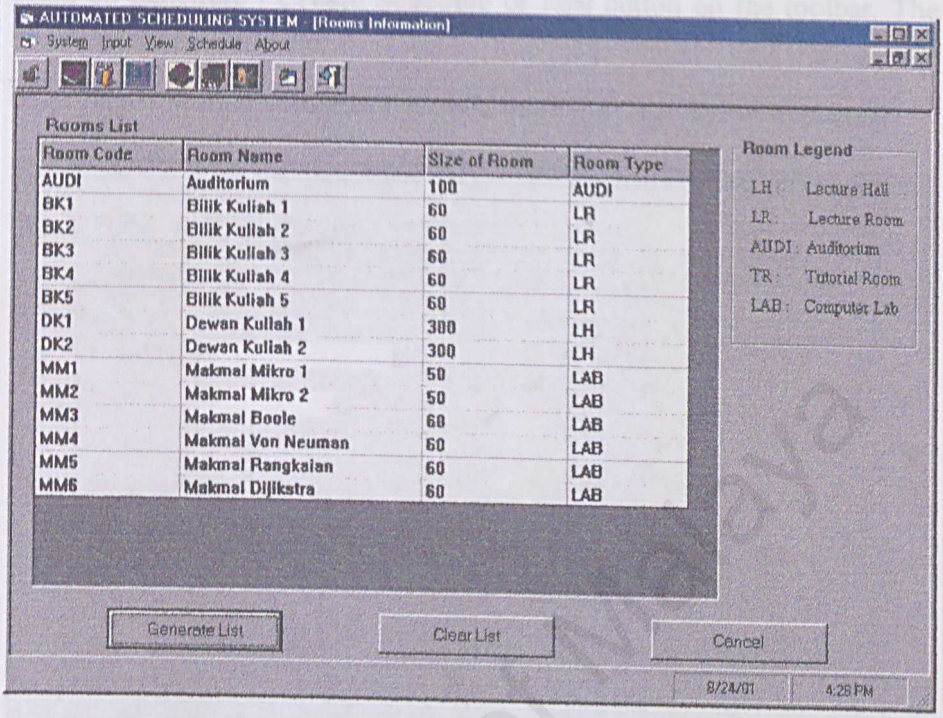
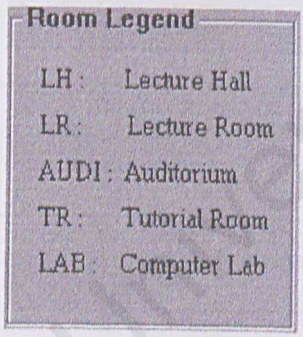
Step	Action
2	Click on the Generate List button to generate the lecturers' list. The following screen appears.
3	Click on the Clear List button to clear the list.
4	Click on the Cancel button to exit from this screen.



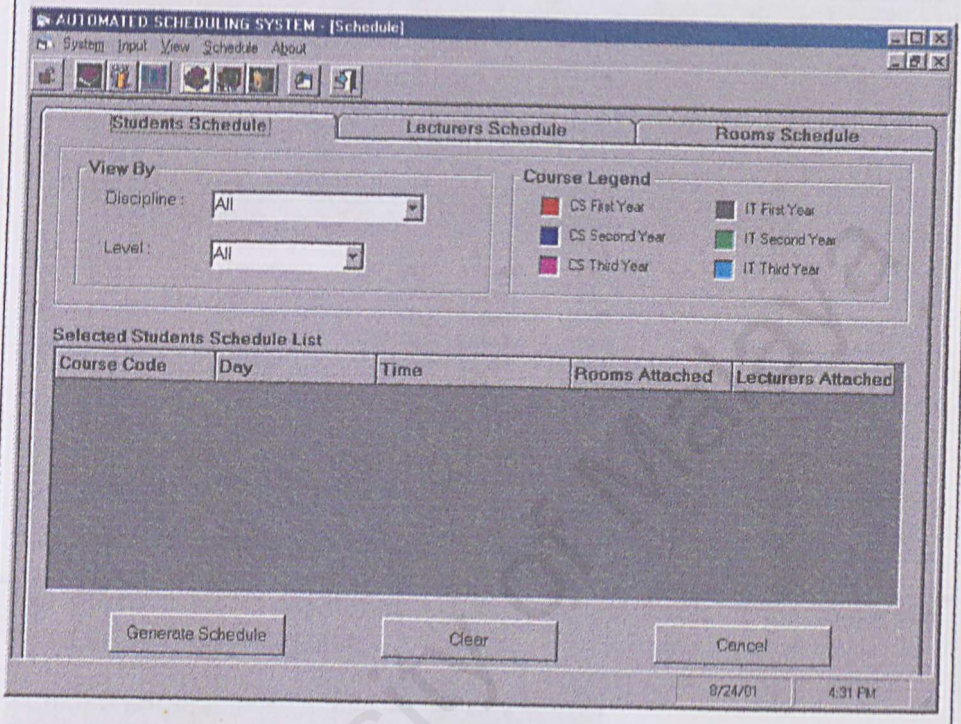
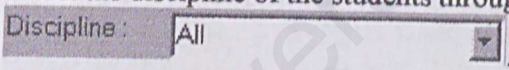
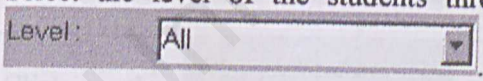
View Rooms Information Window

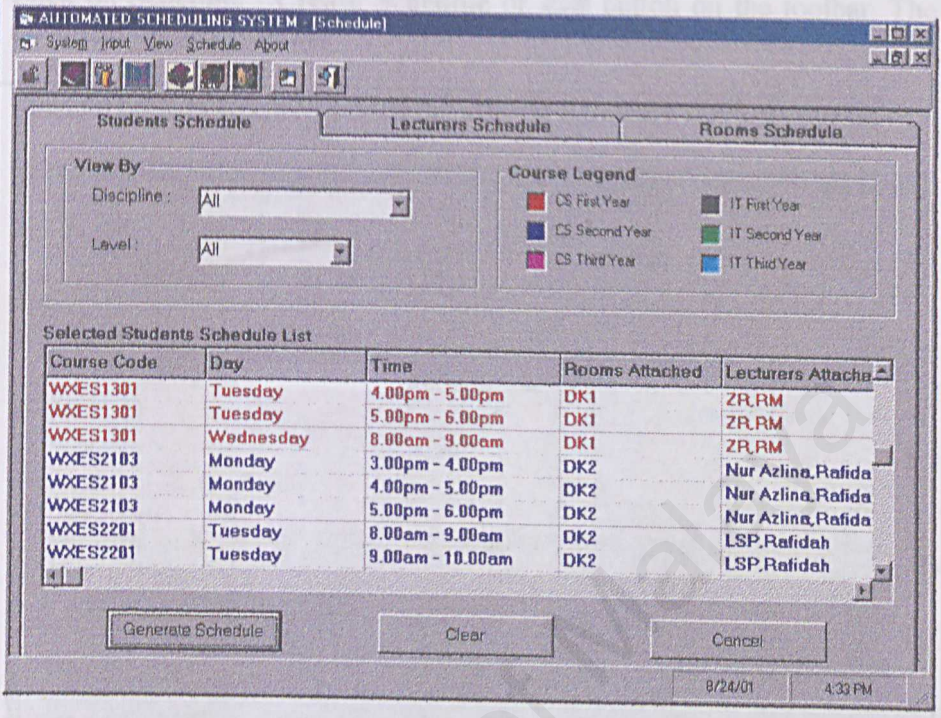
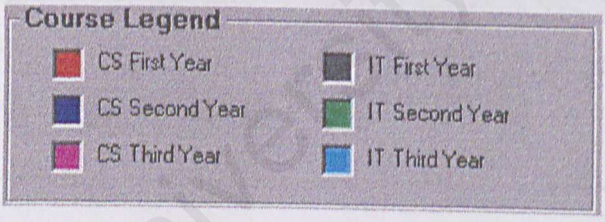
Step	Action
1	Click on **View | Room Information** or  button on the toolbar. The following window appears.



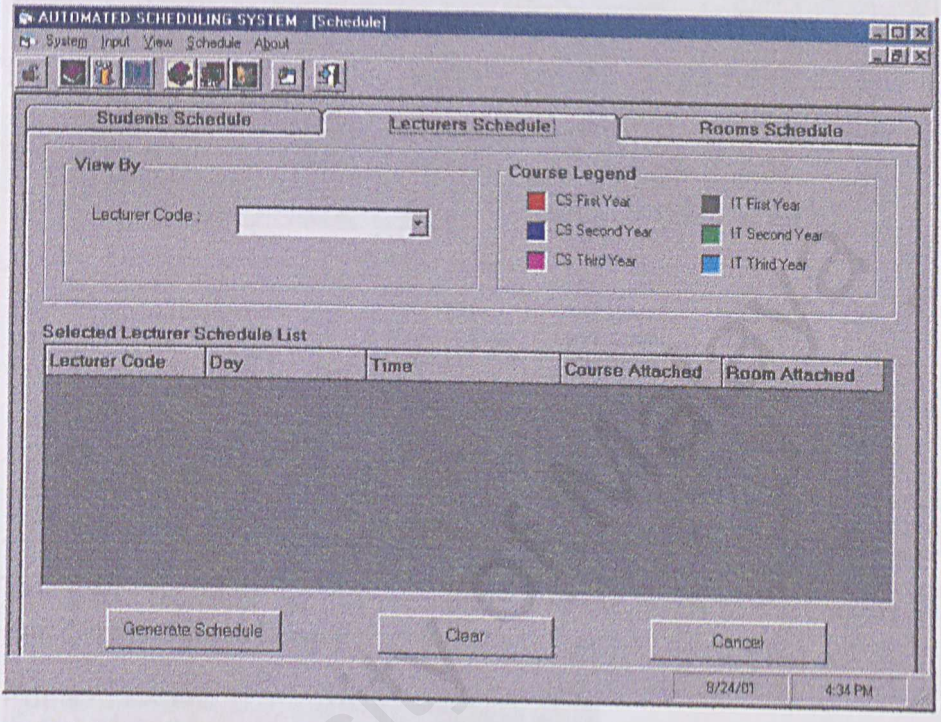
Step	Action
2	Click on the Generate List button to generate the rooms' list. The following screen appears. 
3	The acronyms represent the types of rooms as stated in the **Room Legend**. 
4	Click on the Clear List button to clear the list.
5	Click on the Cancel button to exit from this screen.

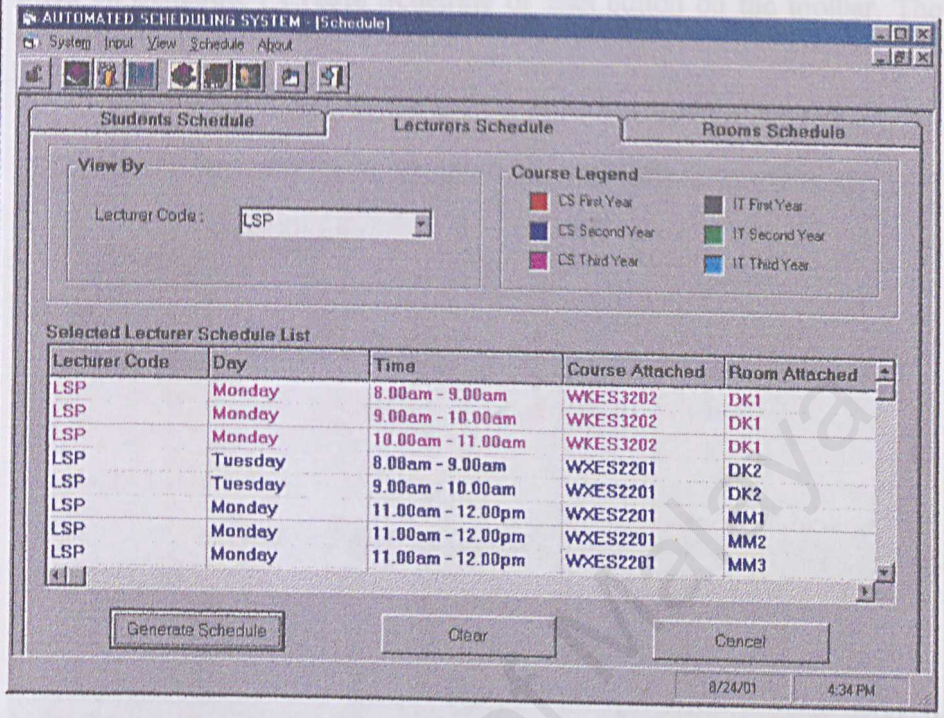
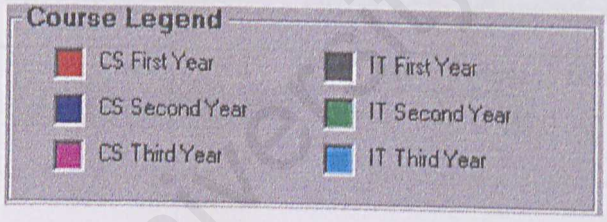
Students Schedule Window

Step	Action
1	Click on **Schedule | Create Schedule** or  button on the toolbar. The schedule will be generated.
2	Click on the **Students Schedule** tab and the following window appears. 
3	Select the discipline of the students through the **Discipline:** drop down menu 
4	Select the level of the students through the **Level:** drop down menu 

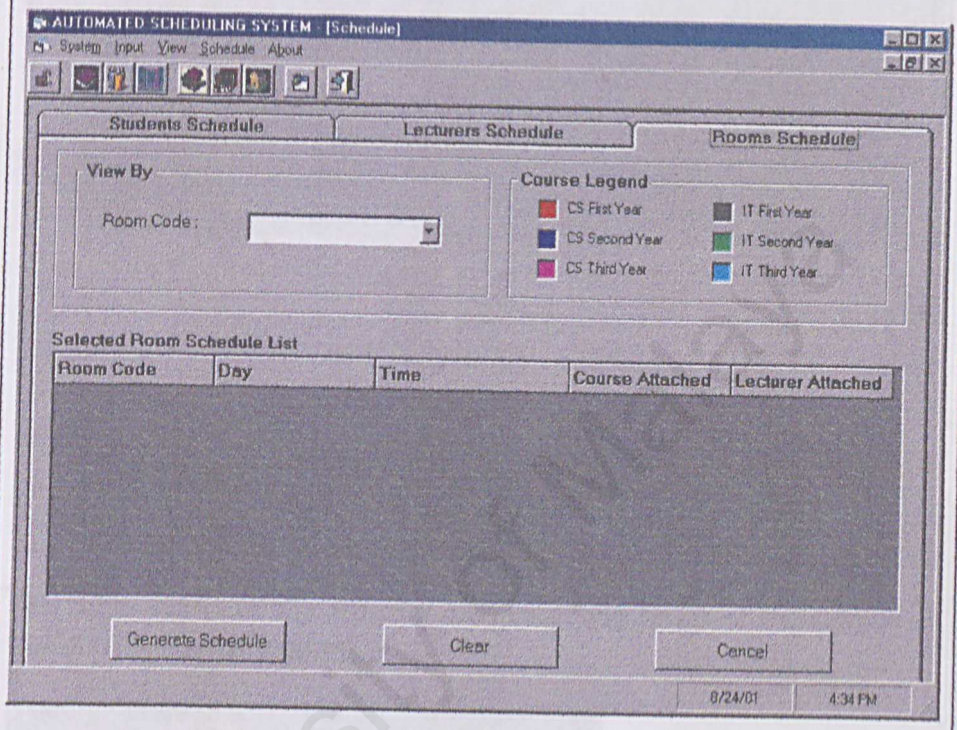
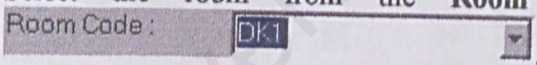
Step	Action
5	Click on the Generate Schedule button to generate the schedule of the selected students. The following screen appears. 
6	The colors represent the different disciplines and levels of the courses as stated in the **Course Legend**. 
7	Click on the Clear button to clear the selected students schedule list.
8	Click on the Cancel button to exit from this screen.

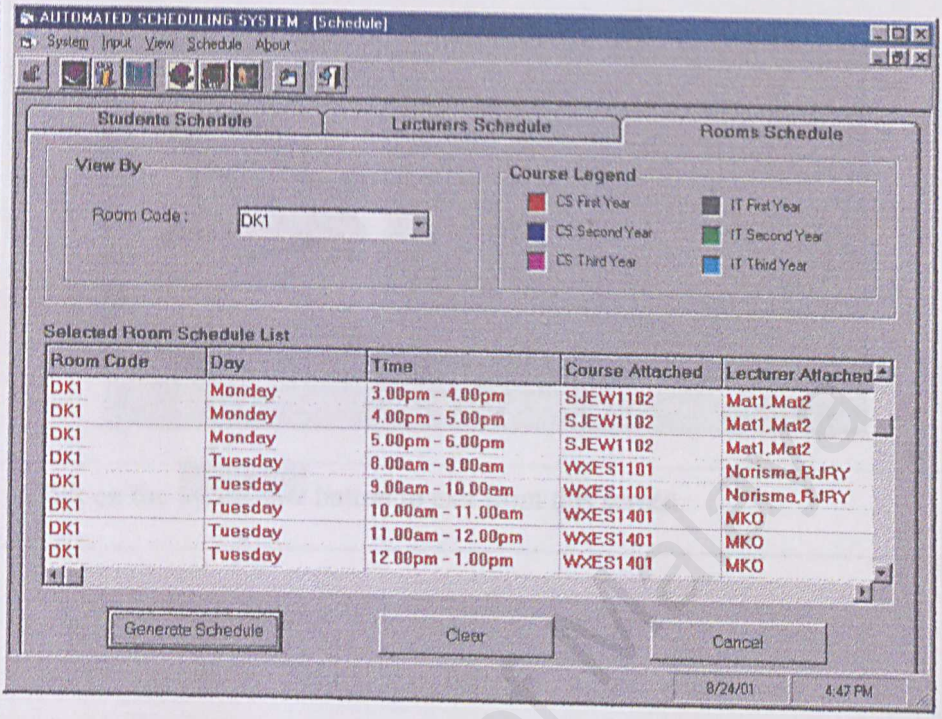
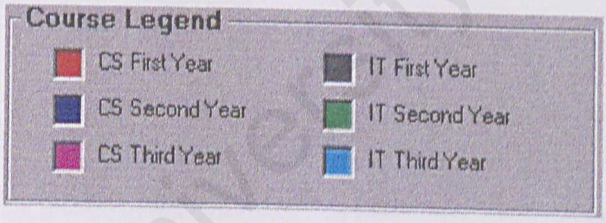
Lecturers Schedule Window

Step	Action
1	Click on **Schedule | Create Schedule** or  button on the toolbar. The schedule will be generated.
2	Click on the **Lecturers Schedule** tab and the following window appears. 
3	Select the lecturer from the **Lecturer Code:** drop down menu 

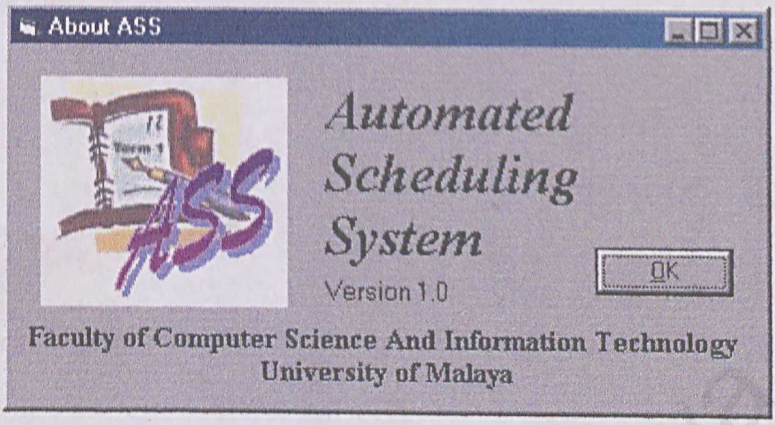
Step	Action
4	Click on the Generate Schedule button to generate the schedule of the selected lecturer. The following screen appears. 
5	The colors represent the different disciplines and levels of the courses as stated in the **Course Legend**. 
6	Click on the Clear button to clear the selected lecturer schedule list.
7	Click on the Cancel button to exit from this screen.

Rooms Schedule Window

Step	Action
1	Click on **Schedule | Create Schedule** or  button on the toolbar. The schedule will be generated.
2	Click on the **Rooms Schedule** tab and the following window appears. 
3	Select the room from the **Room Code:** drop down menu 

Step	Action
4	Click on the Generate Schedule button to generate the schedule of the selected room. The following screen appears. 
5	The colors represent the different disciplines and levels of the courses as stated in the **Course Legend**. 
6	Click on the Clear button to clear the selected room schedule list.
7	Click on the Cancel button to exit from this screen.

About ASS Window

Step	Action
1	Click on **About | ASS** on the toolbar. The following window appears. 
2	Click on the OK button to exit from this screen.

Appendix

Coding Sample

```
Option Explicit
Public dbLect As Database
Public rsLect As Recordset
Public dbRoom As Database
Public rsRoom As Recordset
Dim rows As Integer
Dim j As Integer
Dim z As Integer
Dim countRow As Integer
Dim pos As Integer
Dim lect As String
Dim lect1 As String
Dim lect2 As String
```

```
Private Sub cmdClear1_Click()
   cboDiscipline.Clear
   cboLevel.Clear
   flxStuSche.Clear
   flxStuSche.rows = 1
   DisplayStuSche
   showcboDiscipline
   showcboLevel
   cboDiscipline.SetFocus
End Sub
```

```
Private Sub cmdCancel_Click(Index As Integer)
   Unload frmSchedule
End Sub
```

```
Private Sub DisplayStuSche()
Dim i%

   flxStuSche.Clear
   flxStuSche.rows = 1

   flxStuSche.FormatString = "Course Code" + vbTab + "Day" + vbTab + "Time" + vbTab + _
      "Rooms Attached" + vbTab + "Lecturers Attached"

   flxStuSche.Redraw = True
   flxStuSche.ColWidth(0) = 2000
   flxStuSche.ColWidth(1) = 2000
   flxStuSche.ColWidth(2) = 2400
   flxStuSche.ColWidth(3) = 2000
   flxStuSche.ColWidth(4) = 2000

   flxStuSche.Row = 0
   For i = 0 To flxStuSche.Cols - 1
      flxStuSche.Col = i
      flxStuSche.CellFontSize = 10
      flxStuSche.CellFontBold = True
      flxStuSche.ColAlignment(i) = 1
      flxStuSche.RowHeight(0) = 380
   Next i
End Sub
```

```
Private Sub cmdGenerate1_Click()
Dim displayMsg As String

   DisplayStuSche
   j = 0

   displayMsg = "Application is processing data " + vbCrLf + "This will take a few seconds. " +
vbCrLf + "Please wait ..... "
   Call showProgress(30, 0, Int((10 * Rnd) + 60), displayMsg)
   Me.MousePointer = 11

   If cboDiscipline.Text = "All" Then
      If cboLevel.Text = "All" Then
         For z = 0 To countRow - 1
            ShowCourse
            If flx1.TextMatrix(z, 5) = "CS" And flx1.TextMatrix(z, 6) = "1" Then
               cColorLoop (vbRed)
            ElseIf flx1.TextMatrix(z, 5) = "CS" And flx1.TextMatrix(z, 6) = "2" Then
               cColorLoop (vbBlue)
            ElseIf flx1.TextMatrix(z, 5) = "CS" And flx1.TextMatrix(z, 6) = "3" Then
               cColorLoop (vbMagenta)
            ElseIf flx1.TextMatrix(z, 5) = "IT" And flx1.TextMatrix(z, 6) = "1" Then
               cColorLoop (vbBlack)
            ElseIf flx1.TextMatrix(z, 5) = "IT" And flx1.TextMatrix(z, 6) = "2" Then
               cColorLoop (vbGreen)
            ElseIf flx1.TextMatrix(z, 5) = "IT" And flx1.TextMatrix(z, 6) = "3" Then
               cColorLoop (vbCyan)
            End If
         Next z

      ElseIf cboLevel.Text = "First Year" Then
         For z = 0 To countRow - 1
            If flx1.TextMatrix(z, 6) = "1" Then
               ShowCourse
               If flx1.TextMatrix(z, 5) = "CS" Then
                  cColorLoop (vbRed)
               ElseIf flx1.TextMatrix(z, 5) = "IT" Then
                  cColorLoop (vbBlack)
               End If
            End If
         Next z

      ElseIf cboLevel.Text = "Second Year" Then
         For z = 0 To countRow - 1
            If flx1.TextMatrix(z, 6) = "2" Then
               ShowCourse
               If flx1.TextMatrix(z, 5) = "CS" Then
                  cColorLoop (vbBlue)
               ElseIf flx1.TextMatrix(z, 5) = "IT" Then
                  cColorLoop (vbGreen)
               End If
            End If
         Next z

      ElseIf cboLevel.Text = "Third Year" Then
         For z = 0 To countRow - 1
            If flx1.TextMatrix(z, 6) = "3" Then
               ShowCourse
               If flx1.TextMatrix(z, 5) = "CS" Then
                  cColorLoop (vbMagenta)
```

```
            ElseIf flx1.TextMatrix(z, 5) = "IT" Then
              cColorLoop (vbCyan)
            End If
          End If
        Next z
      End If

    ElseIf cboDiscipline.Text = "Computer Science" Then
      If cboLevel.Text = "All" Then
        For z = 0 To countRow - 1
          If flx1.TextMatrix(z, 5) = "CS" Then
            ShowCourse
            If flx1.TextMatrix(z, 6) = "1" Then
              cColorLoop (vbRed)
            ElseIf flx1.TextMatrix(z, 6) = "2" Then
              cColorLoop (vbBlue)
            ElseIf flx1.TextMatrix(z, 6) = "3" Then
              cColorLoop (vbMagenta)
            End If
          End If
        Next z

      ElseIf cboLevel.Text = "First Year" Then
        For z = 0 To countRow - 1
          If flx1.TextMatrix(z, 5) = "CS" And flx1.TextMatrix(z, 6) = "1" Then
            ShowCourse
            cColorLoop (vbRed)
          End If
        Next z

      ElseIf cboLevel.Text = "Second Year" Then
        For z = 0 To countRow - 1
          If flx1.TextMatrix(z, 5) = "CS" And flx1.TextMatrix(z, 6) = "2" Then
            ShowCourse
            cColorLoop (vbBlue)
          End If
        Next z

      ElseIf cboLevel.Text = "Third Year" Then
        For z = 0 To countRow - 1
          If flx1.TextMatrix(z, 5) = "CS" And flx1.TextMatrix(z, 6) = "3" Then
            ShowCourse
            cColorLoop (vbMagenta)
          End If
        Next z
      End If

ElseIf cboDiscipline.Text = "Information Technology" Then
  If cboLevel.Text = "All" Then
    For z = 0 To countRow - 1
      If flx1.TextMatrix(z, 5) = "IT" Then
        ShowCourse
        If flx1.TextMatrix(z, 6) = "1" Then
          cColorLoop (vbBlack)
        ElseIf flx1.TextMatrix(z, 6) = "2" Then
          cColorLoop (vbGreen)
        ElseIf flx1.TextMatrix(z, 6) = "3" Then
          cColorLoop (vbCyan)
        End If
      End If
```

```
            Next z

        ElseIf cboLevel.Text = "First Year" Then
            For z = 0 To countRow - 1
              If flx1.TextMatrix(z, 5) = "IT" And flx1.TextMatrix(z, 6) = "1" Then
                ShowCourse
                cColorLoop (vbBlack)
              End If
            Next z

        ElseIf cboLevel.Text = "Second Year" Then
            For z = 0 To countRow - 1
              If flx1.TextMatrix(z, 5) = "IT" And flx1.TextMatrix(z, 6) = "2" Then
                ShowCourse
                cColorLoop (vbGreen)
              End If
            Next z

        ElseIf cboLevel.Text = "Third Year" Then
            For z = 0 To countRow - 1
              If flx1.TextMatrix(z, 5) = "IT" And flx1.TextMatrix(z, 6) = "3" Then
                ShowCourse
                cColorLoop (vbCyan)
              End If
            Next z
          End If
      End If

      Me.MousePointer = 0
      displayMsg = "Process will be completed soon." + vbCrLf + "Please wait ..... "
      Call showProgress(20, 68, 100, displayMsg)

  End Sub
  ------------------------------------------------------------------------------------------

  Private Sub Form_Load()
  Dim displayMsg As String
  Dim message As String
  Dim title As String

     displayMsg = "Application is processing data " + vbCrLf + "This will take a few seconds. " +
  vbCrLf + "Please wait ..... "
     Call showProgress(30, 0, Int((10 * Rnd) + 20), displayMsg)
     Me.MousePointer = 11

     Call Schedule
      Me.MousePointer = 0
     displayMsg = "Process will be completed soon." + vbCrLf + "Please wait ..... "
     Call showProgress(20, 68, 100, displayMsg)

    showcboDiscipline
    showcboLevel
    showcboLectCode
    showcboRoomCode
    DisplayStuSche
    DisplayLectSche
    DisplayRoomSche

End Sub
```

```
Private Sub showcboLectCode()
Dim i As Integer
Dim LectCode As String

   Set dbLect = OpenDatabase(App.Path + "\databases\lect.mdb")
   Set rsLect = dbLect.OpenRecordset("Lecturers", dbOpenTable)

   rows = 0
   rows = rsLect.RecordCount
   cboLectCode.Clear
   For i = 1 To rows
      LectCode = rsLect.Fields("Lecturer Code").Value
      cboLectCode.AddItem LectCode
      rsLect.MoveNext
   Next i

End Sub
```

```
Private Sub ShowCourse()
Dim cDay As String
Dim cTime As String
Dim cCou As String
Dim cRoom As String
Dim cLect As String
Dim s As String

   cDay = flx1.TextMatrix(z, 0)
   cTime = flx1.TextMatrix(z, 1)
   cCou = flx1.TextMatrix(z, 2)
   cRoom = flx1.TextMatrix(z, 3)
   cLect = flx1.TextMatrix(z, 4)

   s = cCou + vbTab + cDay + vbTab + cTime + vbTab + cRoom + vbTab + cLect
   flxStuSche.AddItem s

   j = j + 1

End Sub
```

```
Private Sub cColorLoop(colorCode)
Dim i As Integer
Dim k As Integer

   flxStuSche.Row = 1
   For i = 1 To flxStuSche.rows - 1
      If i = j Then
         For k = 0 To flxStuSche.Cols - 1
            flxStuSche.Col = k
            flxStuSche.CellForeColor = colorCode
            flxStuSche.CellFontBold = True
         Next k
      Else
         flxStuSche.Row = flxStuSche.Row + 1
      End If
   Next i
End Sub
```