USING ANALOGUES TO PREDICT STUDENT'S PERFORMANCE AND GENERATE ADVICE

By

CHONG LEE KIAN

WEK 990223

A THESIS PRESENTED TO THE FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY OF UNIVERSITY MALAYA IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR HE DEGREE OF

BACHELOR OF COMPUTER SCIENCE

UNIVERSITY MALAYA

JANUARY 2002

ABSTRACT

This report provides an overview and technical report of what I did in my thesis. The main objective of this project is to develop a system to predict student's performance and generate advice for all the computer science students in University Malaya.

Nowadays, student advising become more important because of the increasingly amount of the computer science student problem in the Faculty Computer Science & Information Technology (FCSIT). Students receive counseling help in planning their academic schedules and in clarifying career goals throughout their few years in University Malaya. Academic advisor and students work together to review academic success to verify completion of study requirements, and to discuss student's interest. Therefore, it is essential for the academic advisor to have a system to assist them in the administration work. The system will enhance the quality of work and reduce the workload of the academic advisor, also as a lecturer.

The effectiveness of the advising task may be enhanced with the use of procedures or tools that forecast the student's future class performance. This project presents a prototype intelligent system that uses case-based reasoning in order to forecast a student's performance. The system draws conclusion on the basis of similarities between a student's current class performance and the performance of other students that attended the same class. Presented here are the problem domain, my approach for an operational system, and the results achieved by the developed prototype system. Findings of the prototype implementation indicate that the potential utility of this predictive approach is high. Any educator may develop a similar system that is customized to the structure of his/her own classes and be capable of assisting in advising students on their class progress way before it is too late for the student.

ACKNOWLEDGEMENTS

The development of the student performance prediction and advisory system was carried out with the advice, assistance and contributions from many individuals.

I would like to thank to my supervisor, **Dr. Syed Malek Fakar Duani** (Deputy Dean (Academic)), for spending his valuable time to give guidance and advice to me in developing the project. I would also like to convey my special thanks to my moderator, **Dr. Roziati Zainuddin** (Head of AI Department) for taking precious time to listen my presentation during VIVA and giving me some suggestions and guidelines about the project.

Finally, many thanks to my fellow course mates, who were giving me an utmost support during my final report development.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

This introductory chapter gives a description or purpose of the project and problem to be solved. It introduces the project in a general view and the rationale of the project.

This chapter uncovers:

- Project Definition,
- Project Objectives,
- Project Scope,
- Expected Outcome,
- Project Development Life Cycle
- Project Schedule.

1.1 Project Definition

Academic advising is a developmental process, which assist students in the clarification of their life or career goals and in the development of educational plans for the realization of these goals.

Advising students on their class performance and motivating them in order to continue or improve their performance is an integral part of every instruction. The advising tasks may be enhanced with the use of methodologies or tools that forecast the student's future class performance. Predicting future class performance is a difficult process and every attempt to automate this task must overcome a number of challenges.

To address these challenges, it is essential to provide a computerized system for the advisors in predicting student's performance and generating academic advice to student.

Enhanced computer technology has made it possible to systematically store and retrieve large amounts of student's information. This technology has changed the task of academic advising by making transcript and course requirement information readily available to faculty and staff who advise students. Use of the computer eliminates much of the clerical burden once held by advisors who had to transfer information to files by hand. Computers also provide the opportunity to perform more complex tracking of student progress and outcomes. The computer assisted advising practices outline the types of data collected and how they are used, including the use of tracking to plan interventions for at-risk students.

The Student Performance Predictor uses case-based techniques. Case-based Reasoning (CBR) systems adapt old solutions to meet new demands, using old cases to explain new situations, using old cases to critique new solutions, or reasoning from precedents to interpret a new situation or create an equitable solution to a new problem.

Case-based reasoning (CBR) systems rely on various "knowledge containers," such as the case-base and similarity criteria, that affect how well a system performs. Explicit or implicit changes in the reasoning environment, task focus, and user base may influence the fit of current knowledge state to task context, which can affect the quality and efficiency of reasoning results. Over time, the knowledge containers may need to be updated in order to maintain or improve performance in response to changes in task or environment.

1.2 Project Objectives

The project objectives are divided into two main parts, the purpose of student's performance assessment and the aims and goals of academic advising.

1.2.1 Purposes of Student's Performance Assessment

The student performance prediction system identifies five purposes. These five purposes include:

- ***Monitoring student progress*** toward desired outcomes is a feature of most assessments developed at any level of initiation.

- ***Holding schools and teachers accountable*** for student achievement — either formally, through a system of rewards and sanctions, or informally, through such mechanisms as reporting school and district performance averages in the media — is a purpose shared by several of the performance assessment systems included in the study.

- ***Certifying student skills and capabilities*** is a purpose of some performance assessment systems.

- ***Achieving better alignment of curriculum, instruction, and assessment*** is the focus of some national reform efforts and is sometimes an implicit goal of state-initiated performance assessment systems as well.

- ***Informing and influencing curriculum and instructional practice*** is the most frequently cited purpose underlying assessment reform; it is assumed that the use of performance assessments will necessarily promote shifts in pedagogy to emphasize higher-order thinking and problem-solving skills.

1.2.2 Aims and Goals of Academic Advising

The faculty has approved a system for academic advising and articulated the following aims and goals of effective academic counseling:

For students:

i. Enable students to take greater responsibility for designing their programs of study.

ii. Increase the degree of motivation with which students approach their academic works.

iii. Encourage and assist the student to explore and articulate interests or career goals.

iv. Encourage the student to take a "reasoned, contemplative approach" to the problem of designing a program of study.

v. Assist the student in designing a program within the liberal arts framework that is coherent and purposeful and is clearly related to interests or career goals.

vi. Ensure that the student has been fully informed about all available options and has been encouraged to examine all these options, and that the course of study is designed to meet the student's individual goals.

vii. Provide the student every reasonable opportunity to have an adviser who takes an empathetic interest in him or her as an individual.

For academic advisers:

i. To provide advisers who are willing not merely to monitor the student's academic program, but also to speak personally with the student and explore his or her changing interests and goals.

ii. To motivate faculty members to give high priority to advising.

iii. To provide a framework for discussion and exchange among advisers so that they may learn and profit from each other's experience.

iv. To motivate advisers to improve their advising and help them find concrete ways to do so.

v. To ensure that advisers have current and detailed information about course offerings and are aware of the full variety of options offered to students.

vi. To facilitate the accumulation of a body of collective interpretation to help advisers make judgments in cases where administrative policy is not clear.

vii. To provide for evaluation of the performance of individual advisers and for concrete suggestions about ways in which performance may be improved.

viii. To provide for evaluation of the advising system and for the discovery of modifications that might improve its usefulness.

1.3 Project Scope

The function of this project is to develop a system to predict student's performance based on few factors such as frequencies of attending classes, commitment, class participation, homework assignments, a series of laboratory assignments, projects, midterm tests and final examination.

This system typically was developed for the Department of Artificial Intelligence, Faculty of Computer Science & Information Technology (FCSIT). The major concern is to predict the computer science students' performance and generate academic advices according to their performance.

The problem domain presented here are students' performance. The system draws conclusions on the basis of similarities between a student's current class performance and the performance of other students that taken the same subject course.

Basically the system has several scopes:

- Student information management
- Counseling record management
- Generation of various type of report and analysis for counselor's reference
- Generation of academic advice on student's performance

1.4 Expected Outcome

Education both enriches our lives and empowers us with the knowledge and skills we needed to succeed. Professional counselors play a critical role in maximizing educational opportunities. Consequently, it is a great expectation that this project will produce a complete prediction and advisory system for the Faculty of Computer Science & Information Technology, University Malaya.

The outcome of this project is to efficiently help students to realize their potential academically, personally and socially. Therefore, the outputs of this project are listed below:

- Various type of information which is needed by computer science students
- Progress report for use by counselor or the students themselves
- Better student and counselor information management
- More effective and efficient work

1.5 Facilities Required

Hardware requirement:

The development of Student Performance Predictor requires a machine with

- GenuineIntel Pentium(r) III 500 MHz Processor
- 256MB RAM
- 11 Gigabyte hard drive storage
- Mouse and keyboard

Software requirement:

Development of Student Performance Predictor is performed on the following platforms:

- Windows 98 operating system
- Microsoft Visual Basic 6.0
- Microsoft Notepad

1.6 Project Development Life Cycle

A development life cycle as shown in Figure 1.1, is used to develop an advisory system that fulfills the needs of end users.

The development life cycle consists of six major phases, which are:

- Assessment
- Knowledge Acquisition
- Design
- Test
- Documentation
- Maintenance

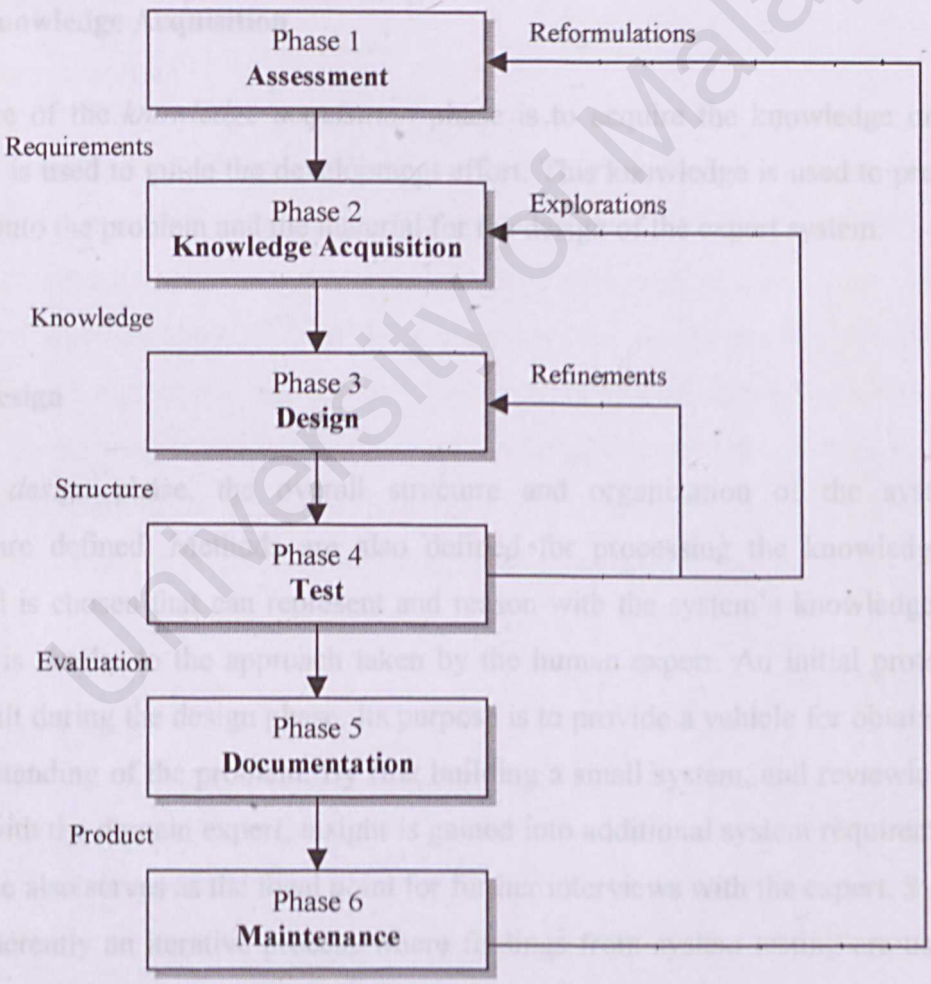


Figure 1.1: Phases in System Development Life Cycle

Phase 1 – Assessment

During the *assessment* phase, studies are conducted to determine the feasibility and justification of the candidate problem. Following this study, the problem is further examined to define the overall goals of the project. This effort specifies the important features and scope of the project, and also establishes the needed resources including project personnel. Sources of needed knowledge, including experts and various reports, are also identified. After this initial phase of the project, the principal project requirements are defined.

Phase 2 – Knowledge Acquisition

The objective of the *knowledge acquisition* phase is to acquire the knowledge on the problem that is used to guide the development effort. This knowledge is used to provide both insight into the problem and the material for the design of the expert system.

Phase 3 – Design

During the *design* phase, the overall structure and organization of the system's knowledge are defined. Methods are also defined for processing the knowledge. A software tool is chosen that can represent and reason with the system's knowledge in a manner that is similar to the approach taken by the human expert. An initial prototype system is built during the design phase. Its purpose is to provide a vehicle for obtaining a better understanding of the problem. By first building a small system, and reviewing the test results with the domain expert, insight is gained into additional system requirements. The prototype also serves as the focal point for further interviews with the expert. System design is inherently an iterative process where findings from system testing are used to refine the system's knowledge and structure.

Phase 4 – Testing

The *testing* phase is not a separate task, but rather a continual process throughout the project. Following each interview with the domain expert, new knowledge is added to the system. This is followed by additional testing where again the system's knowledge may be modified. The major objective of testing is to validate the overall structure of the system and its knowledge. In addition, this phase studies the acceptability of the system by the end-user. Throughout the testing, the designer works closely with both the domain expert who serves to guide the growth of the knowledge and the end-user who provides guidance to the development of the system's interface.

Phase 5 – Documentation

The *documentation* phase addresses the need to compile all the project's information into a document that can meet the requirements of both the user and developer of the expert system. Accommodating the user requires that the documentation meets requirements found in most software projects. That is, it explains how to operate the system and possibly provides a tutorial that step through the major operational features of the system. In particular, the documentation must contain a knowledge directory that provides a well-organized presentation of the system's knowledge and problem solving procedures. It is augmented throughout the project as new knowledge is obtained.

Phase 6 – Maintenance

After the system is deployed in the work environment, it will need to be periodically *maintained*. The system's knowledge may need to be refined or updated to meet current needs. Major system requirement changes may also occur that would require a reformulation of system specifications. Therefore, it is important that an effective maintenance program be established for an expert system project.

1.7 Project Schedule

To achieve the project objectives, a project schedule was planned to manage the time and task that must be accomplished within the development phases. The project schedule are divided into nine major activities which are listed as below:

i. Research and literature review
ii. Identifying problems, opportunities and objectives
iii. Determining information requirements
iv. Analyzing system requirements
v. Designing the recommended system
vi. Developing and documentation application
vii. Testing and maintaining system
viii. Implementing and evaluating system
ix. Writing report

Research and literature review on the prediction and advisory system is carried out. Resources for the literature review included reference book, journals, Internet and etc.

In ***identifying problems, opportunities and objectives*** phase, problems in this project were identify, opportunities where the situation that could be improved were recognized. The objectives of the system were also determined. This phase was carried out together with the literature review.

Determine information requirement is the next phase where the requirements of users were determined. Several techniques were used to define information requirements including interviewing. This phase was carried out wit the literature review.

Analyzing system needs phase will be carried out after the information requirements are determined. One of the main activities of the prediction and advisory system is the data flow through the system, so data flow diagram will be used.

During ***designing the recommended system***, collected information will be used in order to accomplish the logical design of the system. Part of the logical design of the system is designing the user interface. The design phase also includes designing the database.

In the phase of ***developing and documenting application***, student performance prediction application and advisory application were developed. These application programs were developed using Microsoft Visual Basic 6.0. The modular development was applied where programming is broken into logical and manageable portions. During this phase, documentation for application, including procedure manual was also developed.

In ***testing and maintaining system*** phase, the system will be tested. Testing will be conducted includes unit testing, module testing, sub-system testing and system testing.

Implementing and evaluating system is the final phase to be carried out where the users will be trained to use the system and the system will be evaluated.

Table 1.1 on the next page illustrates the project schedule from July 2001 to January 2002.

Activity	July	Aug	Sept	Oct	Nov	Dec	Jan
Research and Literature Review							
Identifying Problems, Opportunities and Objectives							
Determining Information Requirements							
Analyzing System Requirements							
Designing the Recommended System							
Developing and Documentation Application							
Testing and Maintaining System							
Implementing and Evaluating System							
Writing Report							

Table 1.1: Project Schedule

CHAPTER 2

LITERATURE REVIEW

This chapter will discuss the meaning and overview of some areas covered in this project literally. The major area will be covered are:

- The Concept of Advising,
- Artificial Intelligence
- Overview of Case-based Reasoning.

2.1 The Concept of Advising

"*Academic advising is a developmental process which assists students in the clarification of their life/career goals and in the development of educational plans for the realization of these goals. The advisor serves as a facilitator of communication, a coordinator of learning experiences through course and career planning and academic progress review, and an agent of referral to other campus agencies as necessary.*" (American College Testing Program, 1984) [23]

Ideally, advising is first a means of exploring careers and majors and then a method for selecting courses and arranging schedules. As buddies in the process, students can learn to discover options, frame questions, gather information, and make decisions, which can increase their involvement in university and encourage them to persist to graduation.

Research on university students suggests that activities like advising could increase students' involvement in their college or university experiences. This report focuses on outcomes of advising in the context of research on contact between faculty and students, students' involvement, and persistence. [4]

2.1.1 Advisory System

Each student is assigned to a faculty adviser who has been chosen especially for this task and who works with his or her advisee until the end of the sophomore year. Juniors and seniors are assigned either to their major department chairs or to other members of the department, if the chair determines such an assignment appropriate. Students should be particularly careful when arranging their academic programs, for they must comply with all graduation requirements and fulfill specific prerequisites. Faculty advisers are not infallible, and students must remember that the final responsibility for meeting all of the academic requirements rests with the individual student. [24]

Both adviser and student have a responsibility in counseling. It is essential that both take the matter seriously if the student is to achieve a meaningful and successful program of study. In the dialogue between adviser and student, the adviser serves in two capacities: to interpret the university and its goals to the student, and to encourage the advisee to gain understanding of his or her potential and how it may be developed. In a very practical way, the adviser is a source of information for the advisee, explaining campus rules and customs, giving clarification about special programs and requirements, and more. The adviser will make this step as easy as possible for the student. The adviser does, however, encourage student initiative as an important aspect of the individual's self- development.

In most elementary schools, students spend the better part of their day with one teacher. That person gets to know the students extremely well, and usually gets to know the students' parents fairly well, too. In this type of setting, the art or physical education teacher naturally tells the 2nd grade teacher about an unusual incident in her class. When a 2nd grader misbehaves on the playground or wins a music award, the regular classroom teacher is informed. In secondary schools, most students have 4 to 8 teachers each day. Guidance counselors typically have 125 to 350 students assigned to them and little sustained contact with most students. [21]

In an advisory system, the goal is to have one adult in a secondary school who sees each student every day during a time similar to a traditional homeroom time. The adult is also that student's advisor—an element that differentiates advisory time from homeroom time. Such issues as the role of the guidance counselor and the method by which a reasonable ratio of students to staff can be achieved are addressed in later chapters. For now, what's important is that an advisory system guarantees certain benefits to students in a secondary school [21]:

- Each student is known well by one staff member.
- The staff member receives all important information on the student.
- The staff member knows the student's parents or guardians.
- Each student has one advocate in the school.

No student can get lost because she is quiet, or he doesn't stand out in any obvious way—or because no particular adult thinks having a strong relationship with a student is her responsibility. All secondary schools need an advisory system, and the larger the school, the greater the need.

2.1.2 Measurement and Evaluation

Measurement and evaluation are important in student's performance prediction and advising. The meaning of the two terms is explained below.

Measurement

Measurement as used by teacher is a process of collecting information about the performance of a student or a class. It is a descriptive process, which describes student performance or characteristic. Measurement often includes the assignment of a number to express in quantitative terms the degree to which a student possesses a given characteristic. For instance, a student's ability is measured to write a program using C++, and then his or her score, let say 80 of 100 points on a scoring sheet will be recorded. In that case, a student's performance has been measured and reported in numerical or descriptive terms. Such quantification tends to increase objectivity of the description so that it will have the same meaning from time to time and from person to person.

Measurement is not an end in itself. It does not imply judgments concerning the worth or value of the behavior being measured.

One of the most common tools of measurement used by teachers is the paper and pencil test. It measures many kinds of performance well. It is obviously not the only tool, however. Scales, cameras, tally sheets, anecdotal records and many more tools are used to collect information about (measure) student performance.

Evaluation

When a teacher makes value judgments about students' performance, then she is doing more than measuring. He is using measurement data to evaluate. Evaluation takes place when a teacher determines which students have satisfactory completed a laboratory project and which ones have not. Evaluation occurs when teacher compares a student's potential with his or her performance.

There was a comparison made in each example of evaluation above. The performance of students in the computer science course was measured. That performance was then compared to the minimum requirements for passing the class: those who met or exceeded the requirements passed. Student's qualifications or behavior was compared with the requirements, and some students were found eligible to participate in interschool competition; the child's performance was measured and then compared with his potential. Evaluation, then, is a process of comparing student's performance or characteristics against a standard.

A student's performance may be compared with the performance of other students (normative evaluation); or a student's performance may be compared with a predetermined standard (criterion evaluation) as in the case of determining which students are eligible for interschool competition. Deciding that a student's spelling score of 70% earns him or her an A (any score of 65 to 80 is an A in this teacher's class) is another example of criterion evaluation because the teacher compared the student's score with the pre-set standard she had set for A's, B's, C's, etc.

Although evaluations in education do not necessarily involve measurements, the usual purpose of measuring is to provide data that may be used in the evaluative process.

2.1.3 Why Evaluate Student?

Lecturer has many reasons for evaluating students. They are classified as either primary or secondary reasons.

Primary Reasons:

Primary reasons for evaluating pupils are those reasons, which are an essential part of a lecturer's main responsibility--helping students, improve in knowledge and skills, feelings and attitudes; helping student learn.

1. *Improving instructional materials*

   Lecturer need information regarding how effective teaching procedures, activities, the textbook, and other materials are in teaching what needs to be learned. Evaluation can provide this. If the teacher has the information and updates it frequently then he can modify and plan instruction, which will be best for the students.

2 *Improving student learning*

   Both lecturers and students need to know how students are doing. First, of course, they should know what the goal is toward which the students are studying--what they eventually need to know or to be able to do. If through evaluation lecturers and students get feedback as to what students already know, have learned, or don't know yet, then lecturers can direct students' study appropriately to learn the remaining material.

3 *Determining content mastery*

   Lecturer evaluates students to determine if and when they have mastered the subject matter.

4 *Establishing criteria or standards of performance for the course*

Through evaluation a lecturer can better decide how much of the material to be taught can be learned in the time available by the kind of students who usually enroll in the class. With this information the lecturer can establish realistic criteria or standards for the class.

5 *Teaching*

Evaluation activities, if appropriately planned and used, can be powerful learning activities. Self-tests, for example, can communicate to students what the lecturer thinks is important and can give students valuable practice in doing whatever they are learning to do.

Secondary Reasons:

Secondary reasons for evaluating students are those reasons which are not central to the lecturer's responsibility to help students learn but which are often met through evaluation. The needs of others involved in education--parents of the students, administrators, taxpayers, etc.--are met through evaluation, but for this course, these are secondary.

1 *Grading students*

Parents, administrators, universities, and sometimes employees need evidence of pupil progress. Whether progress is reported as a ranking in the class or as a score, which represents how much of the subject has been acquired by the student, evaluation provides the data for the report.

2 *Placing students in special groups or ranking students for special purposes*

Sometimes lecturers choose to group students according to their ability. Students are selected for special experiences or honors, or contests. Evaluation is used to help teachers make the decisions.

3 *Conducting research on teaching methods or curriculum*

Researchers often measure students' ability or growth or needs in order to make decisions regarding which method is effective or which subjects should be taught.

2.2 Artificial Intelligence

Artificial Intelligence (AI) is a field of study in computer science that pursues the goal of making a computer reason in a manner similar to humans. The main goal of AI is to make computers smarter by creating software that will allow a computer to mimic some of the functions of the human brain in selected application [19].

Artificial intelligence gives computers added computing capability, allowing them to exhibit more intelligent behavior. Intelligence, the ability of a human being to acquire knowledge and apply it, means the capability of thinking and reasoning. To a limited degree, artificial intelligence permits computes to accept knowledge from human input, and then use that knowledge through simulated thought and reasoning process to solve problems.

A key part of any AI application is knowledge, an understanding of some subject are obtained through education and experience. A computer can acquire knowledge given to it by human experts. The knowledge consists of facts, concepts, theories, procedures, and relationships. Knowledge is also information that has been organized and analyzed to make it understandable and applicable to problem solving or decision making. Most knowledge bases are limited in that they typically focus on some specific subject area or domain.

Once a knowledge base is built, AI techniques are used to give the computer thought and reasoning capability. The computer will be able to think reason, and make inference and judgments based on the facts and relationships contained in the knowledge base. It will be able to look through the knowledge base and reach conclusion based on the content.

With a knowledge base and the ability to draw inference from it, the computer can now be put to some practical use as a problem solver and decision maker. Figure 2.1 illustrates the concept of a computer using AI in an application. By searching the knowledge base for relevant facts and relationships, the computer can reach one or more alternative

solutions to the given problem. The computer' knowledge base and inferencing capability augment those of the user.

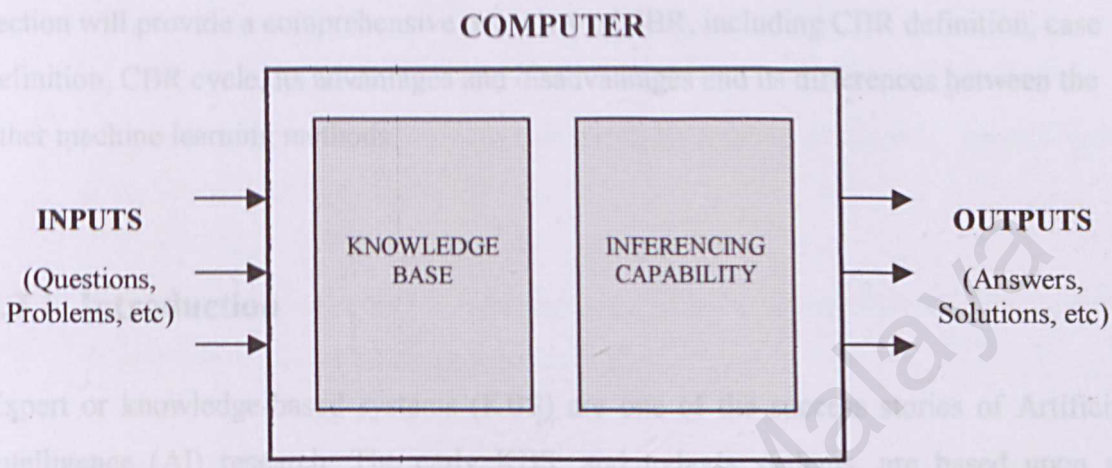


Figure 2.1: Applying AI Concepts with a Computer

2.3 Overview of Case-Based Reasoning

Case-based Reasoning (CBR) is a relatively problem solving technique that is attracting increasing attention. CBR is used for the system to predict student's performance. This section will provide a comprehensive overview of CBR, including CBR definition, case definition, CBR cycle, its advantages and disadvantages and its differences between the other machine learning methods.

2.3.1 Introduction

Expert or knowledge-based systems (KBS) are one of the success stories of Artificial Intelligence (AI) research. The early KBS, and today's systems, are based upon an explicit model of the knowledge required to solve a problem - so called second-generation systems using a deep causal model that enables a system to reason using first principles. But whether the knowledge is shallow or deep an explicit model of the domain must still be elicited and implemented often in the form of rules or perhaps more recently as object models. However, despite the undoubted success of model-based KBS in many sectors developers of these systems have met several problems [5]:

- Knowledge elicitation is a difficult process, often being referred to as the knowledge elicitation bottleneck;
- Implementing KBS is a difficult process requiring special skills and often taking many man years;
- Once implemented model-based KBS are often slow and are unable to access or manage large volumes of information; and
- Once implemented they are difficult to maintain.

However, over the last few years an alternative reasoning paradigm and computational problem solving method has increasingly attracted more and more attention. Case-based reasoning (CBR) solves new problems by adapting previously successful solutions to similar problems. CBR is attracting attention because it seems to directly address the problems outlined above that model-based KBS is facing. Namely:

- CBR does not require an explicit domain model and so elicitation becomes a task of gathering case histories;
- Implementation is reduced to identifying significant features that describe a case, an easier task than creating an explicit model;
- By applying database techniques largely volumes of information can be managed;
- CBR systems can learn by acquiring new knowledge as cases thus making maintenance easier.

2.3.2 What Is CBR?

As the name implies, it is Reasoning, Based on Cases.

From Webster's Dictionary [22]:

- Reasoning - The drawing of inferences or conclusions through the use of facts or other intelligible information.
- Based - Grounded in known theory, knowledge or information.
- Case - Similar set of related facts or information.

Case-based reasoning (CBR) is the act of developing solutions to unsolved problems based on pre-existing solutions of a similar nature. In other words, CBR is an approach to problem solving in which past solutions to problems are retrieved and adapted to solve new problems. CBR system solve new problems by finding solved problems similar to the current problem and adapting their solutions to the current problem, taking into consideration any differences between the current and previously solved situations. Because CBR system associates features of a problem, they are classified as associational reasoning system [15].

CBR class systems use various techniques to match a problem description to a database of previously experienced problems and known solutions. One may think of case-based retrieval as a precedence-based or experience-based diagnosis – it assumes that the reported problem occurred in the past, and that the solution to the problem has been documented. From this perspective, a CBR system is a classification system – it classifies the new problem in order to match it to existing cases. This is in contrast to technologies such as model-based reasoning where an expert system can provide troubleshooting guidance to problems that never experienced before [3].

2.3.3 What Is A Case?

Cases, which represent specific knowledge tied to specific situations, represent knowledge at an operational level; that is, they make explicit how a task was carried out or how a piece of knowledge was applied or what particular strategies for accomplishing a goal were used. In addition, they capture knowledge that might be too hard to capture in a general model, allowing reasoning from specifics when general knowledge is not available. Another advantage of cases is that they chunk together knowledge that belongs together. A reasoner that uses cases is saved from having to compose a lot of decontextualized pieces of knowledge with each other to solve a problem. The case caches compositions of knowledge that have been made already.

Cases can come in many different shapes and sizes. They may cover a situation that evolves over time, they may represent a snapshot, or they may cover any size time slice in between those extremes. They may represent a problem-solving episode, associate a situation description with an outcome, or do some combination.

A case records experiences that are different from what is expected. Not all differences are important to record, however. Cases worthy of recording as cases teach a useful lesson. Useful lesson are those that have the potential to help a reasoner achieve a goal or set of goals more easily in the future or that warn about the possibility of a failure or point out an unforeseen problem.

As a conclusion, a case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner. There are two major parts to a case: *the lesson(s) it teaches* and *the context in which it can teach its lesson(s)*. The lessons it teaches comprise the case's content; the contexts in which it can teach those lessons are case's indexes. Indexes record under what circumstances it is appropriate to retrieve the case. [16]

2.3.4 A History of CBR

CBR has grown, in part, out of the more general field of artificial intelligence. A.I. is distinct from general computing due to its base premise of attempting to solve a general-purpose problem. Most computers and application code are designed to move and manipulate numbers, "number crunchers". On the other hand, the ultimate expression of artificial intelligence is to develop computer code that mimics and can implement the general mechanisms underlying human intelligence. In other words, develop a computer program that generates solution(s) to new problems based on first principles of logic. First principles are a logical discourse on topic matter that leads to a solution of the problem, given in terms a knowledgeable human can understand. No a-priori knowledge of the problem domain or other solutions of similar problems is required. [22]

During research into the human ability to solve problems, researchers realized that most people derive solutions based on previous experience(s) with similar situations. It has been observed that people even discuss problems and solutions in terms of previous experiences. Thus, it appears obvious that, complete solutions derived solely from first principles is fairly rare. Instead, most problem solvers approach new problems and their associated solutions by relating both the problem and the solution to previous experiences. Thus, they build a new solution from information gained from previous experiences, coupled with some reasoning from first principles.

Expert Systems or Knowledge Based Systems (KBS) are a subset of CBR, and are based on a more limited problem domain (domain knowledge). This has evolved in this manner largely because a general problem solver was too broad based of a task to be accomplished.

2.3.4 A History of CBR

CBR has grown, in part, out of the more general field of artificial intelligence. A.I. is distinct from general computing due to its base premise of attempting to solve a general-purpose problem. Most computers and application code are designed to move and manipulate numbers, "number crunchers". On the other hand, the ultimate expression of artificial intelligence is to develop computer code that mimics and can implement the general mechanisms underlying human intelligence. In other words, develop a computer program that generates solution(s) to new problems based on first principles of logic. First principles are a logical discourse on topic matter that leads to a solution of the problem, given in terms a knowledgeable human can understand. No a-priori knowledge of the problem domain or other solutions of similar problems is required. [22]

During research into the human ability to solve problems, researchers realized that most people derive solutions based on previous experience(s) with similar situations. It has been observed that people even discuss problems and solutions in terms of previous experiences. Thus, it appears obvious that, complete solutions derived solely from first principles is fairly rare. Instead, most problem solvers approach new problems and their associated solutions by relating both the problem and the solution to previous experiences. Thus, they build a new solution from information gained from previous experiences, coupled with some reasoning from first principles.

Expert Systems or Knowledge Based Systems (KBS) are a subset of CBR, and are based on a more limited problem domain (domain knowledge). This has evolved in this manner largely because a general problem solver was too broad based of a task to be accomplished.

2.3.5 CBR Cycle

The basic characteristic of a CBR system is its ability to represent and utilize a library of cases that at least coarsely cover the problems that come up in a particular domain. The processes involved in CBR can be represented by a schematic cycle in Figure 2.

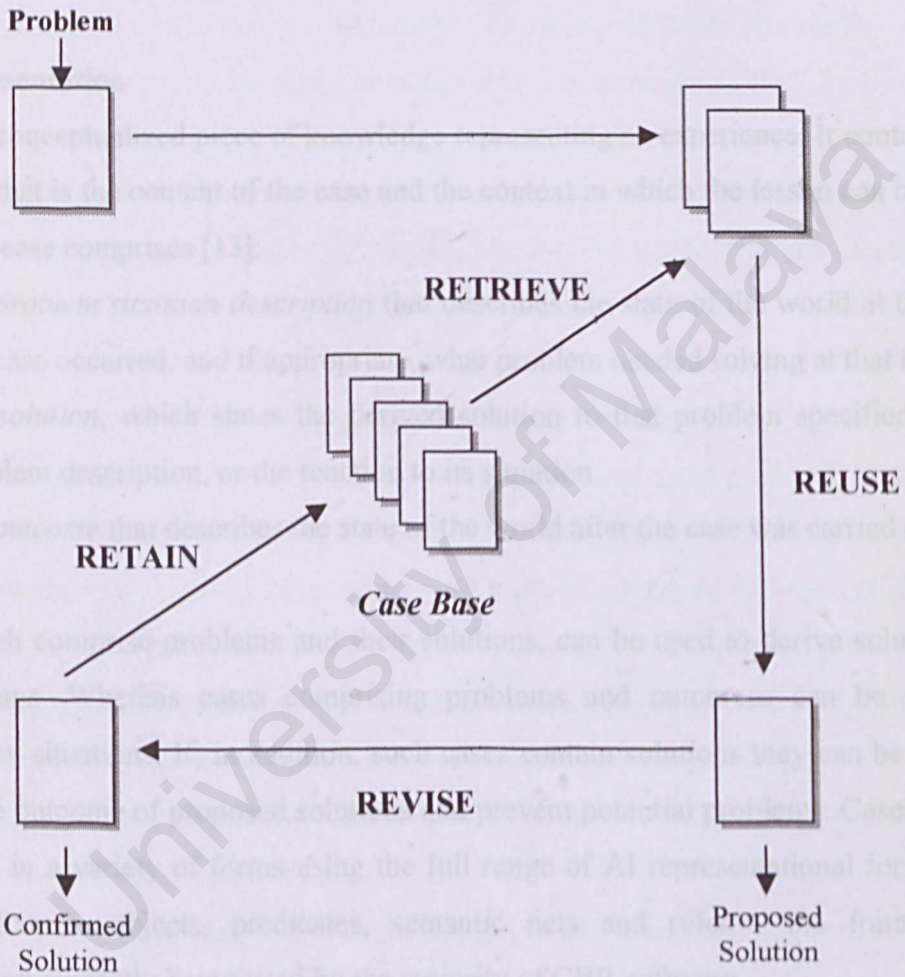


Figure 2.2: The Case-Based Reasoning Cycle

CBR is described as a cyclical process comprising the four REs [1]:

- RETRIEVE the most similar case(s) comparing the case to the library of past cases;
- REUSE the retrieved case(s) to attempt to solve the current problem;
- REVISE the proposed solution if necessary, and
- RETAIN the final solution as a part of a new case.

A new problem is matched against cases in the case base and one or more similar cases are *retrieved*. A solution suggested by the matching cases is then *reused* and tested for success. Unless the retrieved case is a close match the solution will probably have to be *revised* producing a new case that can be *retained*. There are several different methods for organizing, retrieving, adapting, utilizing and indexing the knowledge retained in past cases. The following sections will outline how each process in the cycle can be handled.

Case Representation

A case is a conceptualized piece of knowledge representing an experience. It contains the past lesson that is the content of the case and the context in which the lesson can be used. Typically a case comprises [13]:

- the *problem/situation description* that describes the state of the world at the time the case occurred, and if appropriate, what problem needed solving at that time.
- the *solution*, which states the derived solution to that problem specified in the problem description, or the reaction to its situation.
- the *outcome* that describes the state of the world after the case was carried out.

Cases, which comprise problems and their solutions, can be used to derive solutions to new problems. Whereas cases comprising problems and outcomes can be used to evaluate new situations. If, in addition, such cases contain solutions they can be used to evaluate the outcome of proposed solutions and prevent potential problems. Cases can be represented in a variety of forms using the full range of AI representational formalisms including frames, objects, predicates, semantic nets and rules - the frame/object representation currently being used by the majority of CBR software.

Indexing

Case indexing involves assigning indexes to cases to facilitate their retrieval. Several guidelines for choosing indexes for particular cases have been proposed by the CBR researchers [9]. Indexes should:

- be predictive
- address the purposes the case will be used for
- be abstract enough to make a case useful in a variety of future situations
- be concrete enough to be easily recognized in future situations

Both manual and automated methods have been used to select indexes. Choosing indexes manually involves deciding a case's purpose with respect to the aims of the reasoner and deciding under what circumstances the case will be useful.

There are some different automated indexing methods including:

- *Checklist-based indexing* – Indexing cases by features and dimensions that tend to be predictive across the entire domain i.e., descriptors of the case which are responsible for solving it or which influence its outcome. In this method the domain is analyzed and the dimensions that tend to be important are computed. These are put in a checklist and all cases are indexed by their values along these dimensions. For example, MEDIATOR uses this method by indexing on type and function of disputed objects and relationship between disputants, whilst CHEF indexes on texture and taste. [10]
- *Difference-based indexing* – differentiates cases from one another so that at retrieval time, retrieval algorithms can choose best-matching cases from the case library. During this process the system discovers which features of a case differentiate it from other similar cases, choosing as indexes those features that differentiate cases best. [11]
- *Similarity and explanation-based generalization methods* – produces an appropriate set of indexes for abstract cases created from cases that share some common set of features, whilst the unshared features are used as indexes to the original cases.

- *Explanation-based indexing* – determines relevant features for each case. This method analyses each case to find which of their features are predictive ones. Cases are then indexed by those features. [12]

Storage

Case storage is an important aspect in designing efficient CBR systems in that, it should reflect the conceptual view of what is represented in the case and take into account the indexes that characterize the case. The case-base should be organized into a manageable structure that supports efficient search and retrieval methods. A balance has to be found between storing methods that preserve the semantic richness of cases and their indexes and methods that simplify the access and retrieval of relevant cases. These methods are usually referred to as *case memory models*. The most influential case memory models are the *dynamic memory model* of Schank and Kolodner. The premise is that remembering, understanding, experiencing, and learning cannot be separated from each other. [14]

Retrieval

Given a description of a problem, a retrieval algorithm, using the indexes in the case-memory, should retrieve the most similar cases to the current problem or situation. The retrieval algorithm relies on the indexes and the organization of the memory to direct the search to potentially useful cases.

The issue of choosing the best matching case has been addressed by research into analogy. This approach involves using heuristics to constrain and direct the search. Several algorithms have been implemented to retrieve appropriate cases. They can be serial search, hierarchical search and simulated parallel search.

Case-based reasoning will be ready for large-scale problems only when retrieval algorithms are efficient at handling thousands of cases. Unlike database searches that target a specific value in a record, retrieval of cases from the case-base must be equipped with heuristics that perform partial matches, since in general there is no existing case that exactly matches the new case.

Adaptation

Once a matching case is retrieved a CBR system should adapt the solution stored in the retrieved case to the needs of the current case. Adaptation looks for prominent differences between the retrieved case and the current case and then applies formulae or rules that take those differences into account when suggesting a solution. In general, there are two kinds of adaptation in CBR [7]:

- *Structural adaptation*, in which adaptation rules are, applied directly to the solution stored in cases.

- *Derivational adaptation*, that reuses the algorithms, methods or rules that generated the original solution to produce a new solution to the current problem. In this method the planning sequence that constructed that original solution must be stored in memory along with the solution as in MEDIATOR. Derivational adaptation, sometimes referred to a *reinstantiation*, can only be used for cases that are well understood.

An ideal set of adaptation rules must be strong enough to generate complete solutions from scratch, and an efficient CBR system may need both structural adaptation rules to adapt poorly understood solutions and derivational mechanisms to adapt solutions of cases that are well understood.

Several techniques, ranging from simple to complex, have been used in CBR for adaptation. These include:

- *Null adaptation,* a direct simple technique that applies whatever solution is retrieved to the current problem without adapting it. Null adaptation is useful for problems involving complex reasoning but with a simple solution. For example, when someone applies for a bank loan, after answering numerous questions the final answer is very simple: grant the loan, reject the loan, or refer the application.

- *Parameter adjustment,* a structural adaptation technique that compares specified parameters of the retrieved and current case to modify the solution in an appropriate direction. A specialized adjustment heuristics are applied to the old solution to create a new one. JUDGE (Bain 1986) uses this technique to adapt an old sentence for a crime to a new situation.

- *Abstraction and respecialization,* a general structural adaptation technique that is used in a basic way to achieve simple adaptations and in a complex way to generate novel, creative solutions. The PLEXUS planning system uses this technique.

- *Critic-based adaptation,* in which a critic looks for combinations of features that can cause a problem in a solution. Importantly, the critic is aware of repairs for these problems. PERSUADER is a system, which uses all the techniques of adaptation discussed above.

- *Reinstantiation* is used to instantiate features of an old solution with new features. For example, CHEF can reinstantiate *chicken* and *snow peas* in a Chinese recipe with *beef* and *broccoli* thereby creating a new recipe.

- *Derivational replay* is the process of using the method of deriving an old solution or solution piece to derive a solution in the new situation. For example, BOGART, which replays stored design, plans to solve problems.

- *Model-guided repair* uses a causal model to guide adaptation as in CELIA, which is used for diagnosis and learning in auto mechanics, and KRITIK used in the design of physical devices.

- *Case-based substitution* uses cases to suggest solution adaptation as in ACBARR a system for robot navigation.

2.3.6 Advantages and Disadvantages of CBR

Case-based reasoning is applicable to a wide range of real-world situations, ranging from knowledge-rich situations in which construction of solutions is complex to knowledge-poor situations in which cases provide the only available knowledge.

Case-based reasoning has several advantages [8,17]:

- *Case-based reasoning allows the reasoner to propose solutions to problem quickly, avoiding the time necessary to derive those answers from scratch.*
  Although the CBR has to evaluate proposed solutions, as any reasoner does, it gets a head start on solving problems because it can generate proposals easily. There is considerable advantage in not having to redo time-consuming computations and inferences. This advantage is helpful for almost all reasoning tasks, including problem solving, planning, explanation and diagnosis.

- *A case-based system can be easily be made to learn.*
  In CBR, problem-solving efforts are cached to save future work. Learning is, in effect, a natural consequence of problem-solving efforts. Case-based systems can be engineered to add to their capabilities or adapt to small changes in their environments by continuing to collect cases and insert them into the case library after the system is fielded.

- *When using case-based reasoning to solve problems, solutions can be justified by the cases they are derived from.*
  The cases used to solve a problem provide grist for both justifying derived solutions and analyzing their probable outcomes. In a domain where it is difficult to evaluate solutions objectively, CBR has the advantage of providing illustrations of the effects of particular solutions and kinds of solutions.

- *Case-based reasoners can easily be designed to anticipate potential problems as a natural part of their reasoning.*
  Unsuccessful experiences with past solutions can be used in case-based systems to anticipate possible problems that might result from solving a problem a certain way. In general, this capability adds efficiency, allowing solutions to be partially debugged before they are carried out. In some domains, anticipating problems that might arise when carrying out a solution plan is critical.

- *Cases help a reasoner to focus its reasoning on important parts of a problem by pointing out what features of a problem are the important ones.*
  What was important in previous situations will tend to be important in new ones. Thus, if in a previous case, some set of features was implicated in a failure, the reasoner focuses on those features to ensure that the failure will not be repeated. Similarly, if some features are implicated in a success, the reasoner knows to focus on those features. Such focus plays a role in both problem-solving and interpretive CBR.

- *Case-based reasoning provides a way for humans and computers to interact to solve problems.*
  CBR is inspired by human behavior. However, when we look at the processes involved in CBR, we see that some are easier for people, while others are easier or more appropriate for computer. People, for example, are good at creative adaptation but poor at remembering the full range of applicable cases, either because they tend to be biased in their remembering or because, as novices, they have not yet had the experiences they need to solve some problem. Computers can augment the memory limitations of people, providing for them the cases they would otherwise not remember. CBR provides a way of using the best qualities of both human and computer for solving problems.

- *Knowledge acquisition for a case-based system is natural.*
  Communication between system and domain experts can use concrete examples rather than piecemeal rules. Experts find it difficult to report the knowledge they use to solve problems. They are, however, quite at home reporting their experiences and discussing the ways in which cases are different from one another. Their experiences can be coded as cases. The differences they talk about help with both indexing cases and recording knowledge that adaptation heuristic can use.

There are also disadvantages in using cases to reason:

- A case-based reasoner might be tempted to use old cases blindly, relying too heavily on previous experience without validating it in the new situation.
- A case-based reasoner might allow cases to bias him or her or it too much in solving a new problem.
- Often people, especially novices, are not reminded of the most appropriate sets of cases when they are reasoning (Gick and Holyoak 1980; Gentner 1989).

2.3.7 Comparisons and Differences with CBR

Table 2.1 depicts a comparison between CBR and the machine learning methods. [2]

Machine Learning Method	Resistance Against Data Outliners	Explains Output	Suitable For Small Datasets	Reasoning Capability	Adaptive
Case-Based Reasoning	Yes	Yes	Partially	Yes	Yes
Neural Nets	No	No	No	No	No
Fuzzy Logic	Partially	Yes	Yes	Yes	No
Analogy	Yes	Yes	Partially	Yes	Yes
Rule-Based	Yes	Yes	Yes	Partially	No
Regression Tree	Yes	Yes	Partially	Partially	No
Hybrid System	Partially	No	Partially	Partially	No

Table 2.1: Comparisons Between CBR and Other Machine Learning Methods

The Differences Between CBR and Knowledge-Based System (KBS)

Knowledge-based systems use rules to guide their decision processes. Typically a knowledge engineer works with a domain expert to derive the heuristics that, the expert uses when solving a problem. Whereas, case based reasoning 'looks' for similarities between the current needs and previous examples of similar problems and their attendant solutions. Rule base programming is currently very popular and well developed. Most 'experts' will expound on the rules they use to solve either everyday or very difficult and detailed problems. However research into human problem solving has determined that in almost all cases the 'rules' used by experts have been in part derived from a cause and effect relationship derived from previous experiences - cases. [22]

In short, the most significant difference between CBR and KBS problem solving techniques is that in the KBS paradigm, the rules are more concrete and tangible. Whereas in CBR the solution methodology is a process of comparison and evaluation of current needs with existing situations.

2.3.7 Comparisons and Differences with CBR

Table 2.1 depicts a comparison between CBR and the machine learning methods. [2]

Machine Learning Method	Resistance Against Data Outliners	Explains Output	Suitable For Small Datasets	Reasoning Capability	Adaptive
Case-Based Reasoning	Yes	Yes	Partially	Yes	Yes
Neural Nets	No	No	No	No	No
Fuzzy Logic	Partially	Yes	Yes	Yes	No
Analogy	Yes	Yes	Partially	Yes	Yes
Rule-Based	Yes	Yes	Yes	Partially	No
Regression Tree	Yes	Yes	Partially	Partially	No
Hybrid System	Partially	No	Partially	Partially	No

Table 2.1: Comparisons Between CBR and Other Machine Learning Methods

The Differences Between CBR and Knowledge-Based System (KBS)

Knowledge-based systems use rules to guide their decision processes. Typically a knowledge engineer works with a domain expert to derive the heuristics that, the expert uses when solving a problem. Whereas, case based reasoning 'looks' for similarities between the current needs and previous examples of similar problems and their attendant solutions. Rule base programming is currently very popular and well developed. Most 'experts' will expound on the rules they use to solve either everyday or very difficult and detailed problems. However research into human problem solving has determined that in almost all cases the 'rules' used by experts have been in part derived from a cause and effect relationship derived from previous experiences - cases. [22]

In short, the most significant difference between CBR and KBS problem solving techniques is that in the KBS paradigm, the rules are more concrete and tangible. Whereas in CBR the solution methodology is a process of comparison and evaluation of current needs with existing situations.

CHAPTER 3

METHODOLOGY

This chapter will cover the following section:

- The System and User Requirement
- Waterfall Model with Prototyping
- Case-Based Reasoning Approach
- Development Environment

3.1 System and User Requirements

The system requirement needs to be drawn out to provide a guideline when developing a system. A requirement is a feature of the system or a description of the system is capable of doing in order to fulfill the system purpose.

There are two types of requirement:

- Functional requirement
- Non-functional requirement

3.1.1 Functional Requirements

Functional requirement is a description of an interaction between the system and its environment. It also describes how the system should behave when given a certain stimulus. Functional requirements are frequently identified in terms of inputs, outputs, processes, and stored data that are needed to satisfy the system improvement objectives.

Functional requirements must be analyze such that they can be verified and communicated to both students and lecturer or academic adviser or counselor.

The functional requirements for the proposed system are stated below:

i. General User Section

This section is responsible to communicate with user in getting information from user to progress the student's performance prediction, and displays the outcome to the user.

- Display module

  This is a front-end design, which is responsible to the interaction between the user and the system. It consists the following forms:

  ✓ Input Form

  The purpose of the input form is to enable the user to input the student data such as student name, matric number, IC number, class attendance, assignments, and etc. These data will be combined to enable the agent section for prediction and generate advices.

  ✓ Output Page

  This page shows the prediction result about the student performance.

  ✓ Advice Page

  This page generates some useful advice to the student based on his or her academic performance.

ii. Agent Section

There are two important parts in the agent section, which play major role in back-end of the system.

- Prediction Module

  This module purposely assist user in decision-making by predicting the student performance where student information and related data are required.

- Advice Generation Module

  This module generates advices about the student performance based on the factors that determine the student's performance.

3.1.2 Non-Functional Requirements

A non-functional requirement is a description of other features, characteristics, and attributes of the system as well as any constraints on the system that may limit the boundaries of the proposed solution to the problem. These constraints usually narrow the selection of language, platform or implementation technique and tool.

Below are the non-functional requirements of the system:

i. ***Maintainability***

Maintainability is the degree to which the system can be cost-effectively made to perform its functions in a possibly changing operating environment. The system are easy to modify and test in updating process to meet the new request, correcting errors, or move to a different computer system.

ii. ***Reliability***

The degree in which the system operates in a user-acceptable manner when used in the environment for which it was designed, which does not produce dangerous, costly failure or destructive error when it is applied in a reasonable manner.

iii. ***Efficiency***

Implementation of the system corresponds to the most cost-effective computing resource utilization, where process that can be called or accessed in an unlimited number of times to produce similar outcomes at a creditable pace or speed.

iv. ***Control and Security***

Control requirements represent the environment in which the system must operate, as well as the type and degree of security that must be provided.

v. ***User-friendly Environment***

The design of the system and its interface should be user friendly and easy understanding by all level of the users. The users may be non-technical personnel who would not be able to comprehend complex system. So, the system should use the graphical user interface (GUI) approach in order to provide a better understanding of how to use the system and better communication between the system and users.

Generally, the design of all the interfaces should conform to the following criterions:

- Consistent, in terms of screen design and error messages displayed.
- High degree of understandability and avoid memorization of events and commands.

vi. ***Simplicity***

Forms and screens are kept properly uncluttered in a manner that focuses the user attention.

vii. ***Attractive Interface***

It is important to design an attractive user interface that will appeal to the users. Using images, highlights and variation of colours can make a whole lot of difference in making the system more fun to use.

viii. ***Understandability***

Coding method used, allow other programmer to understand the logic of the program flow.

ix. ***Expandability***

The system should be able to be extended to accommodate more functionality in the future.

3.2 Waterfall Model with Prototyping

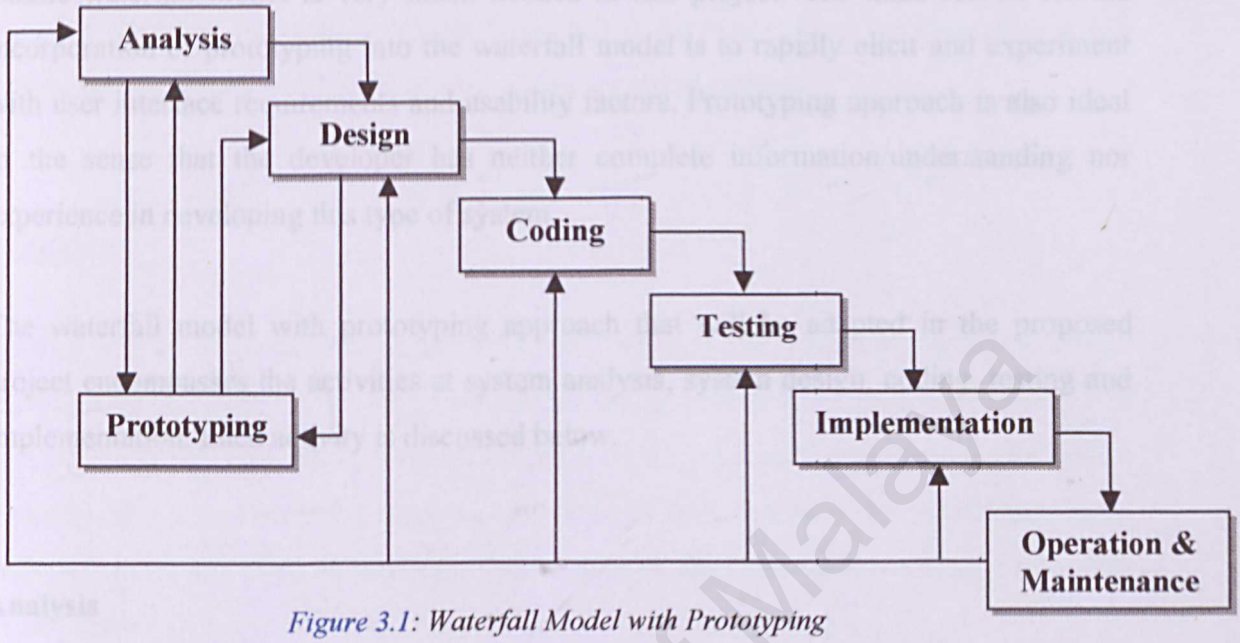


Figure 3.1: Waterfall Model with Prototyping

Combination of the Waterfall model and prototype approach [20] will be used to develop this project. The Waterfall model with prototyping is chosen because the strengths of each can be achieved on a single project. This model is actually the classic waterfall model combined with the prototyping approach in its early stages as shown in Figure 3.1 above.

In the proposed development strategy, the waterfall model will serve as the base for the development because the steps of this model are very similar to the generic steps of software development process that are applicable to all software engineering paradigms. It also provides a template into which methods for analysis, design, coding, testing and maintenance can be placed. Prototyping will be involved in the early stages of the waterfall model where there is a need for experimentation and learning before commitment of any resources to develop the full-scale system. Prototyping will not be involved in the later stages of the development because its major drawback in increasing the opportunities to produce negative effects on structural factors such as performance, design quality and maintainability if not carry out properly.

In the case of the proposed project, an idea solution might be one that combined rapid results (from prototyping) with stability (from the waterfall model). The stability of the classic waterfall model is very much needed in this project. The main reason for the incorporation of prototyping into the waterfall model is to rapidly elicit and experiment with user interface requirements and usability factors. Prototyping approach is also ideal in the sense that the developer has neither complete information/understanding nor experience in developing this type of system.

The waterfall model with prototyping approach that will be adapted in the proposed project encompasses the activities at system analysis, system design, coding, testing and implementation. Each activity is discussed below.

Analysis

This is the phase where the study of the current system is done and the definition of requirements for the new system is made. The system analysis phase is concerned with the data gathering and data analysis. Data will be collected form the system user. Data Flow Diagram (DFD) is used to analyze the collected data because it enables the information domain and functional domain to be modeled at the same time. DFD also be used to graphically show the data flow through the system. The most important outcome from this phase will be an accurate system requirement specification.

Design

In the system design phase, the requirements that were produced in the previous phase are translated into a representation of the system. This phase will be concerned with the user interface design and system design. The user interface will be built using Visual Basic 6.0. In system design, structure chart will be involved in structuring the system's modules and flow chart might be used to depict the design of procedural details.

Coding

The coding phase translates and implements the detail design representation of the system into programming realization. Visual Basic 6.0 is used in coding the information and functional domain as well as the control of the proposed system.

Testing

Testing will be a critical step in assuring the quality of the developed system and will represent the ultimate review of specification, design and coding. First, unit testing will be performed to verify each program module. Next, integration testing is performed to integrate unit-tested program modules and conduct tests that uncover errors associated with the interfacing of those modules. Validation test succeeds when the system functions in the manner that is reasonably expected.

Implementation

The finally stage of the development will be system implementation. The system will be implemented on its target software and hardware requirement. The whole system will be revise to uncover the necessity to add further enhancements.

Operation and Maintenance

Maintenance process should be an ongoing activity in real development. Monitoring a necessary adjustment continue so that the system produces the expected results. However, system enhancements and maintenance will only be carried out in the proposed project if time constraint allowed.

3.3 Case-Based Reasoning Approach

3.3.1 Method Description

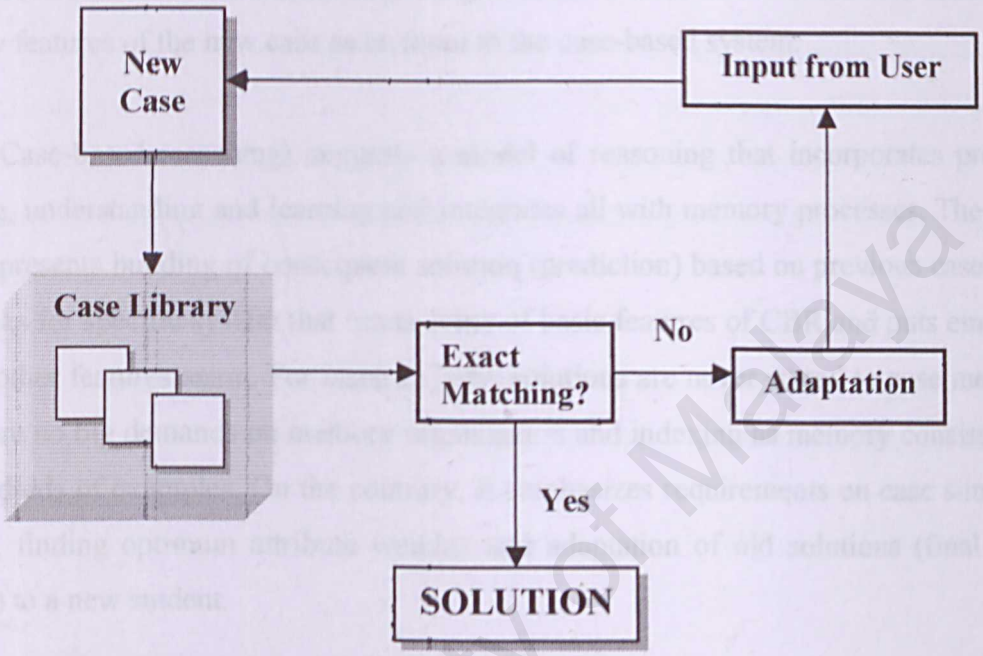


Figure 3.2: Case-Based Reasoning System

CBR (Case-Based Reasoning) is a retrospective system, which replicates the natural way of solving problems by humans. Several past cases of student's performance from case library are retrospected and compared in order to find an identical case.

If a matching case is found then the suggested solution will be used for the new solution to the current student's performance. This means when the new student's performance is presented to the system, the reasoner searches its case library, looking for the library student that his or her class performance is most similar with the class performance of the new student, and finally predicts that the final grade of the given to the student in case library will be the final grade to be assigned to the new student after the completion of the class.

If the matching is not exact but partially matched, then the solution can be obtained by adjusting the old solution. It needs to be emphasized here that the reasoning process to be modeled is to help users to predict the most appropriate final grade according to the class performance of the student. Figure 3.2 depicts the case-based system. The values of the new case are instantiated interactively from the user's local information. The user enters the key features of the new case as an input to the case-based system.

CBR (Case-based reasoning) suggests a model of reasoning that incorporates problem solving, understanding and learning and integrates all with memory processes. The given task represents building of consequent solution (prediction) based on previous cases. The task asks for specific system that omits some of basic features of CBR and puts emphasis on its other features again. For instance, new solutions are never stored to case memory, there are no big demands on memory organization and indexing as memory consists only of hundreds of examples. On the contrary, it emphasizes requirements on case similarity metric, finding optimum attribute weights and adaptation of old solutions (final grade values) to a new student.

3.3.2 Feature-Based Retrieval

Feature-based retrieval is used to retrieve cases from the case base using heuristics that rely on the distribution of cases.

Feature-based retrieval does not rely on the textual content of the cases. Instead, each case is associated with qualitative and quantitative parameters called features or attributes. Features define the important facts about each case and the specific feature-values are used in the classification and the computation of similarity between the case base and the reported problem.

There are two methods used in feature-based retrieval that are *nearest neighbor method* and *induction method*. These two methods are helpful in prediction of student performance.

3.3.2.1 Nearest Neighbor Method

Nearest neighbor class algorithms were used in CBR from its inception. The algorithm computes the similarity between the features of new case and all the cases to determine the cases that resemble the problem based on its known features. As in hand-guided classifications, the system uses questions to narrow the list of possible solutions based on their discriminating capabilities.

However, the internal representation and the authoring process are different. Cases are displayed based on the feature-values of the reported symptom. Questions provide information about features and are selected based on their ability to query the user for the most discriminating features, thereby reducing the number of candidate cases most efficiently. In theory, this approach increases the reusability of questions and lowers maintenance because the number of discriminating features is significantly smaller than the number of cases.

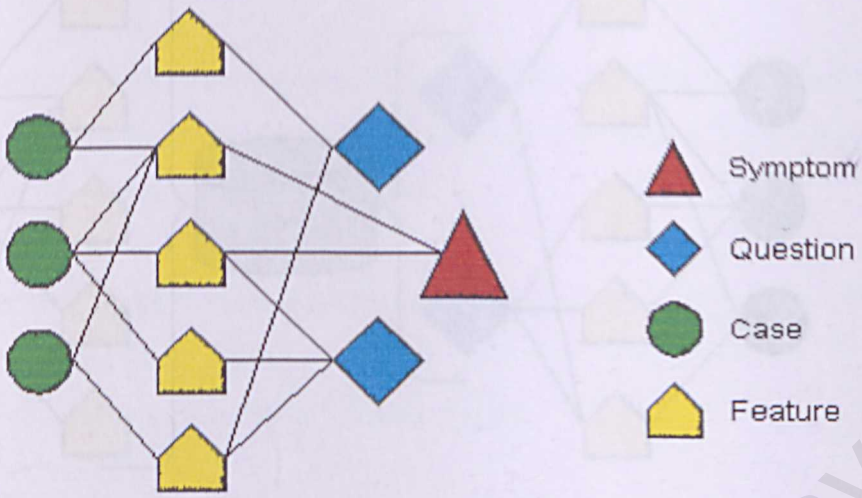


Figure 3.3: Nearest Neighbor Method in Feature-Based Retrieval

Adding a new case requires only definition of its known feature-values, as relevancy to the symptom is determined by similarity calculations. When new questions are authored, it is only required to define the feature(s) they are acquiring from the user.

3.3.2.2 Induction Method

Induction is a technology that generalizes from a collection of sample cases to solve new problems. These cases, often referred to as training cases, can be actual historical data or synthetic cases built for this purpose.

Induction, like nearest neighbor, uses case features. It uses a "greedy search" strategy based on information gain calculations to choose the most discriminating features and devise an effective decision tree. In fact, pure induction methods use the training cases to formulate the search tree and "forget" about the case base at runtime. It uses the decision tree to query the user for features and uses the case base only to propose possible solutions.

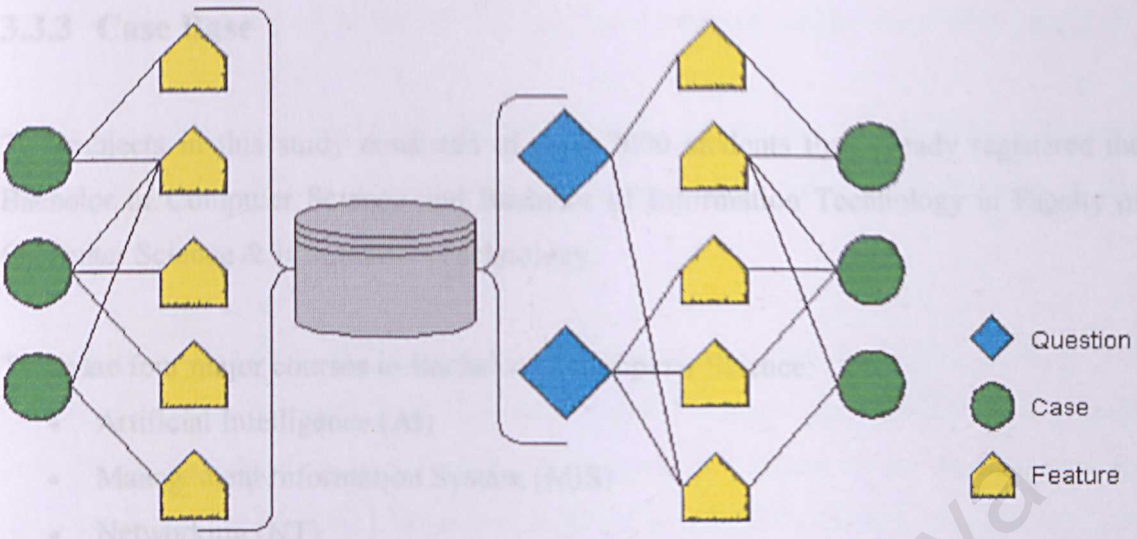


Figure 3.4: Induction Method in Feature-Based Retrieval

While in many ways a nearest neighbor and an induction system appear very similar, differences exist in several areas. Most importantly, inductive systems are most efficient in presenting the best questions and candidate cases to the user, potentially reaching their goal in the least number of steps. For similar reasons, maintenance of large case bases is easier.

In general, inductive systems tend to be faster, especially in large case bases, because of the off-line classification that was completed prior to the runtime stage. On the other hand, nearest-neighbor methods have higher immunity to "noisy" and erroneous data and can handle missing values better.

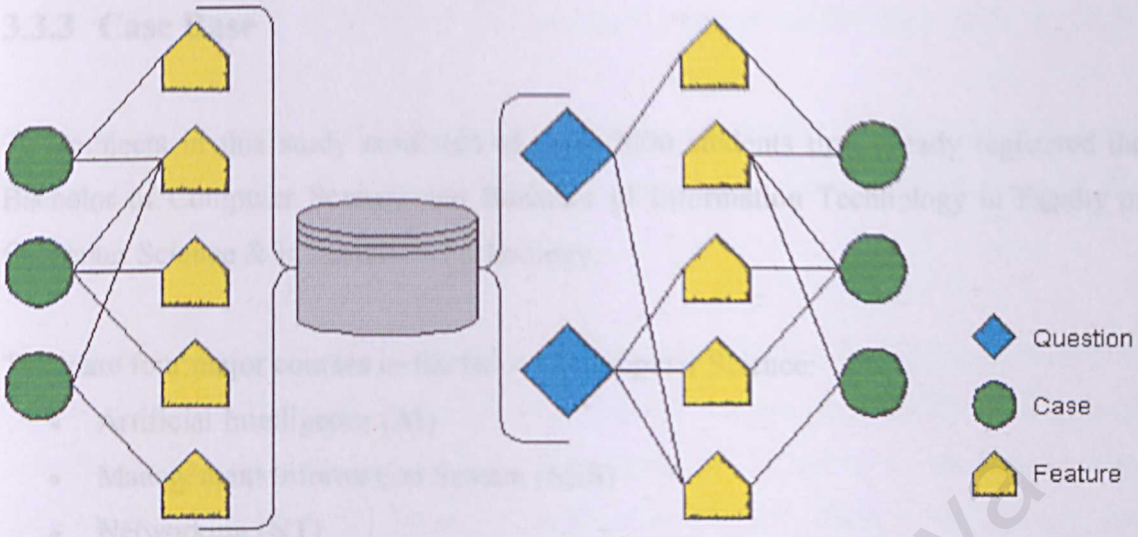


Figure 3.4: Induction Method in Feature-Based Retrieval

While in many ways a nearest neighbor and an induction system appear very similar, differences exist in several areas. Most importantly, inductive systems are most efficient in presenting the best questions and candidate cases to the user, potentially reaching their goal in the least number of steps. For similar reasons, maintenance of large case bases is easier.

In general, inductive systems tend to be faster, especially in large case bases, because of the off-line classification that was completed prior to the runtime stage. On the other hand, nearest-neighbor methods have higher immunity to "noisy" and erroneous data and can handle missing values better.

3.3.3 Case Base

The subjects in this study consisted of over 2000 students that already registered the Bachelor of Computer Science and Bachelor of Information Technology in Faculty of Computer Science & Information Technology.

There are four major courses in Bachelor of Computer Science:

- Artificial Intelligence (AI)
- Management Information System (MIS)
- Networking (NT)
- Software Engineering (SE)

For the Bachelor of Information Technology, there are three major courses that are:

- Information System (IS)
- Management (M)
- Multimedia (MM)

The case base consists of students' performance and his or her final grade for the specific subject. Each student constituted a case that contained the following attributes:

- Matric number
- Student name
- IC number
- Session
- Semester
- Subject
- Class attendance
- Eight tutorials
- Two assignments
- Two midterm test
- Four survey answers
- Student's final grade for the subject

All cases that included all of the student information comprised the space of library cases.

This system task is to search an appropriate value for the final grade field of an input case. Therefore this field is considered the solution data for a particular case in this domain.

The possible data values for the solution data are the characters A, A-, B+, B, B-, C+, C, C-, D+, D, F that represent grade levels. Table 3.1 depicts grade levels and the grade point for each grade. A is the highest possible grade. D, D+ and C- are the failure grades but it can be redeemed if the student has achieved at least 2.0 CGPA (Comparative Grade Point Average) for the semester. F is the failure grade that cannot be redeemed. The other grade levels such as I, K, P, PA, NP, S, U, R and W are not taken in GPA (Grade Point Average) calculation.

Grade	Grade Point
A	4.0
A-	3.7
B+	3.3
B	3.0
B-	2.7
C+	2.3
C	2.0
C-	1.7
D+	1.5
D	1.0
F	0

Pass Grade

Fail Grade that can be redeemed if CGPA>2.0 for the semester

Fail Grade that cannot be redeemed

Table 3.1: Grade and Grade Point

The possible values for the midterm test fields are integers that range between 0 and 10. The possible value for each assignment field is an integer that ranges between 0 and 10, and the possible value for the tutorial is an integer that ranges between 0 and 10. The possible value for the class attendance is a character string, which is excellent, good, fair or poor.

The possible values for the student attributes that determine the final grade for the semester are displayed in Table 3.2.

Attributes	Value Type	Value Range
Class Attendance	String	Excellent; Good; Fair; Poor.
Tutorial	Integer	0 – 10 (%)
Assignment 1 (by group)	Integer	0 – 10 (%)
Assignment 2 (by individual)	Integer	0 – 10 (%)
Midterm Test 1	Integer	0 – 10 (%)
Midterm Test 2	Integer	0 – 10 (%)
Revision (hour per week)	String	Less than 1 hour, 1-2 hours, 2-3 hours, 3-4 hours, More than 4 hours
Understanding in Lecture	String	Very poor; Poor; Fairly well; Well; Very well.
Understanding of Content	String	Very poor; Poor; Fairly well; Well; Very well.
Study Group, Discussion	String	No; Yes, 1-2 hours; Yes, 2-3 hours; Yes, More than 3 hours.

Table 3.2: Student Attributes and Their Possible Values

3.3.4 Matching

There are two matching methodologies were used to test the system's predictive power.

Based on the first method in order to establish the similarity between a certain input case and a library case, the prediction system compared corresponding features one at a time. For example, the first laboratory assignment of the input case with the first laboratory assignment of the library case, the second laboratory assignment of the input case with the second one of the library and so on.

Since each laboratory assignment was equally weighted, instead of comparing corresponding assignment values, the second matching mechanism compared the mean assignment values between a certain input case and a library case.

This system was evaluated for predictability and matching confidence. Predictability is defined as the measure of prediction system's ability to correctly predict the solution data in a set of input cases. A correct prediction is a letter grade that agrees with the grade that eventually was assigned to the student by his or her instructor at the end of the semester. For example, for a given input case where the correct solution data is the letter grade "A" if the prediction system retrieved five library cases as being most similar cases and if four of them indicate that the final grade is an "A" and the final grade of the other one is a "B", then the system' prediction rate is 80%.

Matching confidence is a measure that is directly related to the degree of similarity between an input case and the most similar library case. It represents the degree of similarity as a percentage after taking in consideration that each index feature is weighted in terms of how important the feature is in establishing similarity between two cases. In general, the matching confidence indicates how "certain" is the prediction system in offering a prediction for a particular input case.

3.3.4.1 Assessing Similarity Between Cases

At the heart of a cased-based reasoning (CBR) system is the computation similarity between a new case - the user's input - and previous cases stored in a case library. Cases are associated with qualitative and quantitative parameters called features or attributes. The CBR algorithm calculates the similarity between cases based on feature-value pairs between the new input case and each historical case.

A similarity measure should have the following attributes:

- Reflective: a case is similar to itself
- Symmetric: If A is similar to B, then B is similar to A

A similarity measure is not always transitive: If A is similar to B and B is similar to C, it cannot be asserted that A is similar to C. This is because the features defining the similarities between A and B can be different from these in B and C. For example, a white Ford Escort is similar to a white Dakota truck (they are both white vehicles), and a white Dakota is similar to a red Dakota (they are both Dakota trucks). However, a Ford Escort is not similar to a Dakota.

3.3.4.2 Handling Missing Values

This CBR system able to retrieve cases effectively when some of the feature values are missing as a result of values that was not collected in the past or when the user does not supply them during consultation.

Consider two library cases $Case_1$ and $Case_2$, and a new problem description $Case_n$ that input from the user, which is represented in Table 3.3.

$Case_1$	$Case_2$	$Case_n$
Feature-values	Feature-values	Feature-values
Class attendance = Excellent	Class attendance = Good	Class attendance = Excellent
Homework assignment 1 = 18	Homework assignment 1 = 18	Homework assignment 1 = 18
Homework assignment 2 = 19	Laboratory assignment 2 = 7	Laboratory assignment 2 = 7
Laboratory assignment 1 = 8	Project = 26	Laboratory assignment 4 = 10
Laboratory assignment 2 = 7	Midterm Test = 17	Midterm Test = 17
Laboratory assignment 3 = 10		
Laboratory assignment 4 = 10		
Laboratory assignment 5 = 9		
Project = 26		
Midterm Test = 17		

Table 3.3: Comparing Between Library Cases and Input Case

From Table 3.3, although $Case_1$ matches all the features of the new case, $Case_n$, whereas $Case_2$ matches only three of them, system will identify $Case_2$ as a better match, because they calculate similarity based primarily on the percentage of matching features. $Case_n$ matches 4 of 10 features of $Case_1$, receiving a score of 0.40 (4/10), whereas the match with $Case_2$ will receive a higher score of 0.60 (3/5). In a way, $Case_2$ is preferred because of the small number questions answered by the user. The algorithm must include provisions to compensate for absent values and artificially lower the score of cases with a higher number of absent values.

3.3.5 Suitability

Some of the characteristics of a domain that indicate that a CBR approach might be suitable for this system include:

- records of previously solved problems exist;
- historical cases are viewed as an asset which ought to be preserved;
- remembering previous experiences is useful;
- experience is at least as valuable as textbook knowledge.

When using case-based reasoning, the need for knowledge acquisition can be limited to establishing how to characterise cases. Case-based reasoning allows the case-base to be developed incrementally, while maintenance of the case library is relatively easy and can be carried out by domain experts.

Mentioned in Chapter 2, knowledge-based systems (KBS) use rules in problem solving and decision process. However KBS seem not a suitable method for this system. This approach has several major problems:

- *Knowledge elicitation is difficult*
  This problem was recognized as soon as KBS were built and was often attributed to the knowledge elicitation bottleneck.

- *KBS can be very complex and can take long time to develop*
  This problem is familiar to any KBS developer and has partially been responsible for the increasing interest in KBS development methodologies and of knowledge modeling languages and ontologies.

- *KBS are frequently slow*

- *KBS are often poor at managing large volumes of information*

- *Once developed they are difficult to maintain.*

Hence, there is a strong case for CBR since it has several potential advantages over KBS:

- CBR systems can be built without passing through the knowledge elicitation bottleneck since elicitation becomes a simpler task of acquiring past cases.

- CBR systems can be built where a model does not exist.

- Implementation becomes a simpler task of identifying relevant case features, and moreover a system can be rolled out with only a partial case-base. Indeed, using CBR a system need never be complete since it will be continually growing. This removes one of the bugbears of KBS - how to tell when a knowledge base is complete.

- CBR systems can propose a solution quickly by avoiding the need to infer an answer from first principles each time.

- Individual or generalized cases can be used to provide explanation that are perhaps more satisfactory than explanations generated by chains of rules, important in many domains with legal implications.

- CBR systems can learn by acquiring new cases making maintenance easier.

- Finally, by acquiring new episodic cases CBR systems can grow to reflect their organization's experience. If a rule-based KBS were delivered to six companies and used for six months, after that time each system would be identical, assuming no maintenance had taken place. If six identical CBR systems were used in a similar way after six months there could be six different systems as each could have acquired different episodic cases.

3.4 Development Environment

The development environment of the system consists of two major parts that are hardware and software requirement of the system.

3.4.1 Hardware Requirements

The following hardware requirements are needed for the system operation:

- GenuineIntel Pentium(r) III 500 MHz Processor
- 256 MB RAM
- 11 GB hard drive storage
- Keyboard and mouse

3.4.2 Software Requirements

A few software products will be used to develop the system. These software products are divided into two categories, which are Operating System and System Application Program.

3.4.2.1 Operating System

Microsoft Windows 98

Microsoft Windows 98 is used for the proposed system. Windows 98 is a powerful operating system that all of the users are very familiar with it. Windows is easier to use, more reliable, and faster than ever.

3.4.2.2 System Application Programming Language

Microsoft Visual Basic 6.0

Microsoft Visual Basic 6.0 is used as a programming language in this system.

Visual Basic is a powerful programming language. The Visual Basic language facilitates a structured and disciplined approach to computer design. This language is an extremely rich programming environment. It is not just a language. It is an Integrated Development Environment (IDE) in which you can develop, run, test and debug your application.

The programmer has the ability to create Graphical User Interfaces (GUIs) by pointing and clicking with the mouse. Visual programming eliminates the need for the programmer to write code that generates a form, code to create a control on a form, code that set the property settings for the selected control, code to change foreground and background colours, etc.

All these codes are provided as part of the project. The programmer does not need to be an expert Windows programmer to create functional Windows programs. The programmer creates a GUI and writes code to describe what happens when the user interacts (click, press a key, double-click, etc.) with the GUI. These notifications are called events, which are passed into the program by Microsoft Windows operating system. So the programmer just needs to know the basic principles of developing applications with visual tools and event programming.

CHAPTER 4

SYSTEM DESIGN

System design is a transformation of the specification of requirements, first into a detailed logical/conceptual specification and secondly into a detailed physical/technical specification.

Design is the creative process of transforming the problem into a solution. Conceptual design tells the user exactly what the system will do. Technical design allows system builder to understand the actual hardware and software needed to solve the user's problem.

A design specification describes the features of the system, the components or elements of the system and their appearance to users.

The system design of the system of predicting and advising on student's performance has considered the following design issues:

i. System Functionality Design
ii. Effective Output Design
iii. Effective Input Design
iv. User Interface Design

4.1 System Functionality Design

System functionality design is based on the system requirements stated in Chapter 3. It translates the system requirements into system functionality. The design focuses on the system structure design and data flow design.

Data Flow Diagram (DFD) is a graphical representation of data process throughout the system. The diagrams provide an overview of system inputs, processes and outputs, which correspond to the general system in the project [6].

The following section shows the data flow diagrams in this project. The symbols used were explained in the Table 4.1.

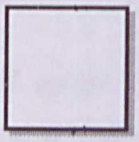
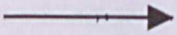
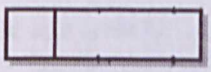
Symbols	Meaning
	**Entity** Any object or event which data is collected. Entity may be a person, place or thing.
	**Data flow** It shows movement of data with head of the arrow pointing toward the data's destination.
	**Process** It denotes a change or transformation of data. It is the work being performed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

Table 4.1: Description of Symbols Used in DFD

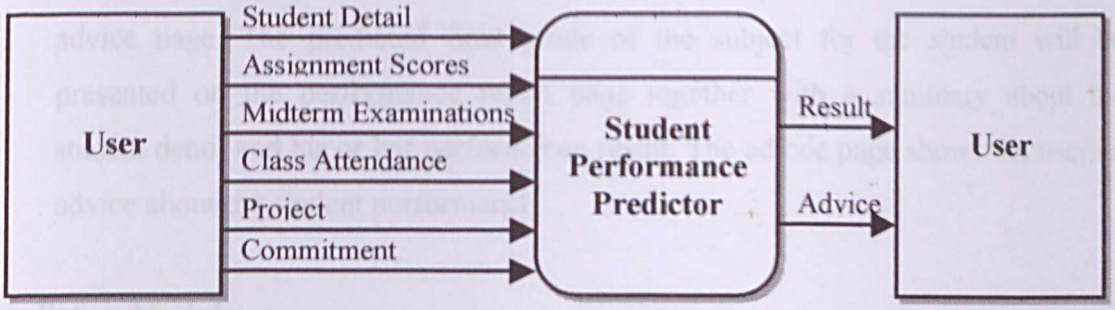


Figure 4.1: Context Level Diagram For Student Performance Predictor

Figure 4.1 represents the context level diagram for the system. User is asked to input the student detail, assignment scores, midterm examinations, project, class attendance and commitment into the system. After the preprocessing of the data input, the systems then produces the student performance result and generate advices to the user.

The proposed system is developed using top down modular approach. This system is divided into three major modules, which are

- Display Module
- Prediction Module
- Advice Generation Module

Display Module

This module is a user interface that consist of two main screen:

i. *User input screen*

This input screen let the user to input student information and details about the student class attendance, homework assignments, a series of laboratory assignments, project, midterm examination, final examination and commitment.

ii. Output screen

There are two pages for the output screen that are performance result page and advice page. The predicted final grade of the subject for the student will be presented on the performance result page together with a summary about the student detail and his or her performance result. The advice page shows the user an advice about the student performance.

Prediction Module

The function of this module is to predict the final grade for the student, which are determined through homework assignments, project, midterm examinations, a series of laboratory assignments, class attendance, commitment and a final examination. The prediction process is performed using case-based techniques, which is stated in chapter 3.

Advice Generation Module

This module will generate advices about the student performance based on the factors that determine the student's performance.

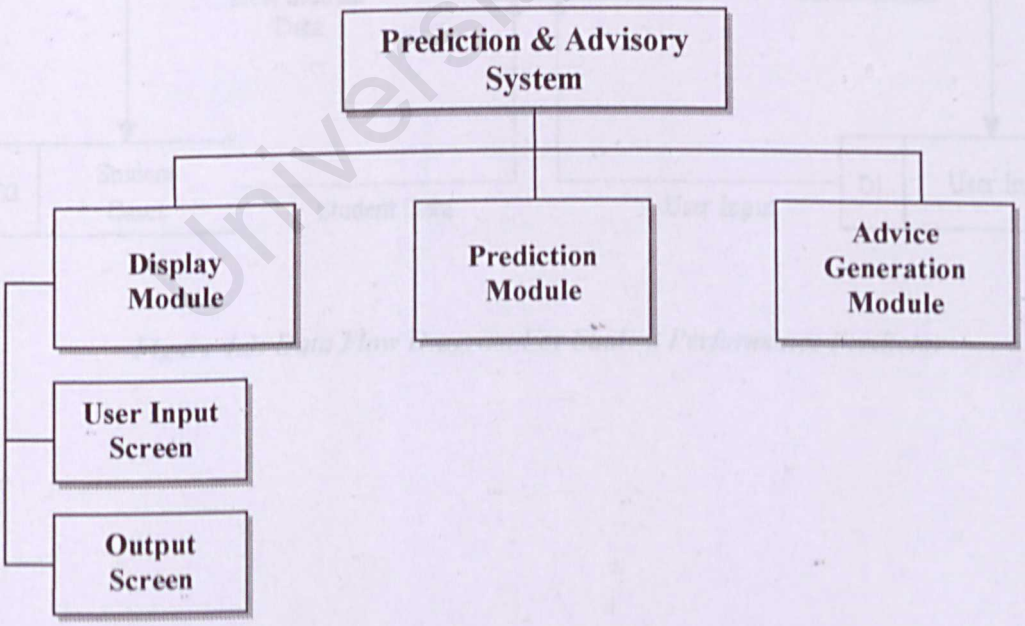


Figure 4.2: Structure Chart of Student Performance Predictor

Data flow diagram for the whole system is illustrated in Figure 4.3 and data flow diagram for the prediction module is shown in Figure 4.4.

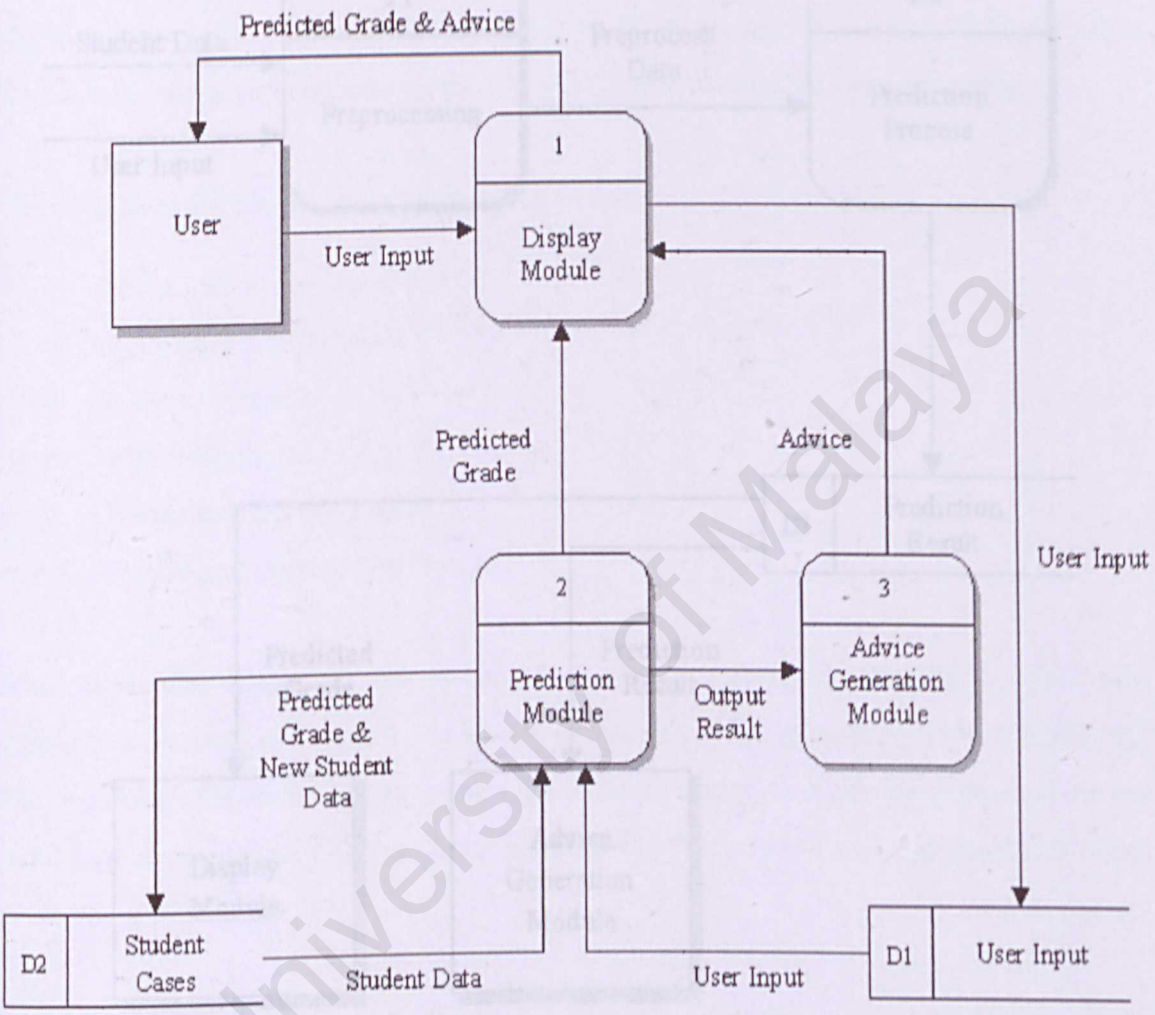


Figure 4.3: Data Flow Diagram For Student Performance Predictor

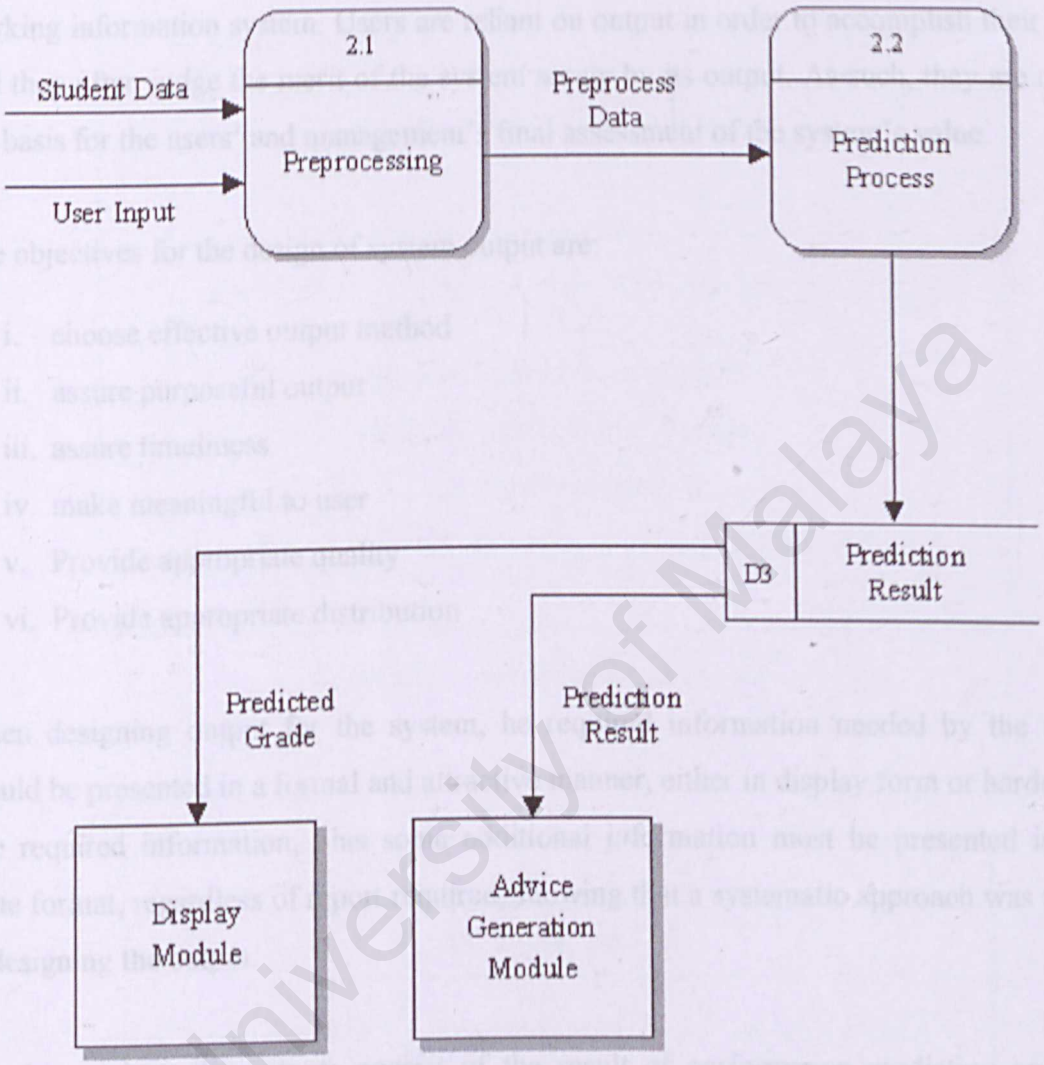


Figure 4.4: Data Flow Diagram For the Prediction Module

4.2 Effective Output Design

Outputs present information to system users. Outputs are the most visible component of a working information system. Users are reliant on output in order to accomplish their task and they often judge the merit of the system sorely by its output. As such, they are often the basis for the users' and management's final assessment of the system's value.

The objectives for the design of system output are:

i. choose effective output method
ii. assure purposeful output
iii. assure timeliness
iv. make meaningful to user
v. Provide appropriate quality
vi. Provide appropriate distribution

When designing output for the system, he required information needed by the users should be presented in a formal and attractive manner, either in display form or hardcopy. The required information, plus some additional information must be presented in the same format, regardless of report required, showing that a systematic approach was taken in designing the output.

For this project, the outputs consist of the result of performance prediction and the academic advices page of the student. Figure 4.5 and Figure 4.6 illustrate the prototype of performance result page and the prototype of advice page.

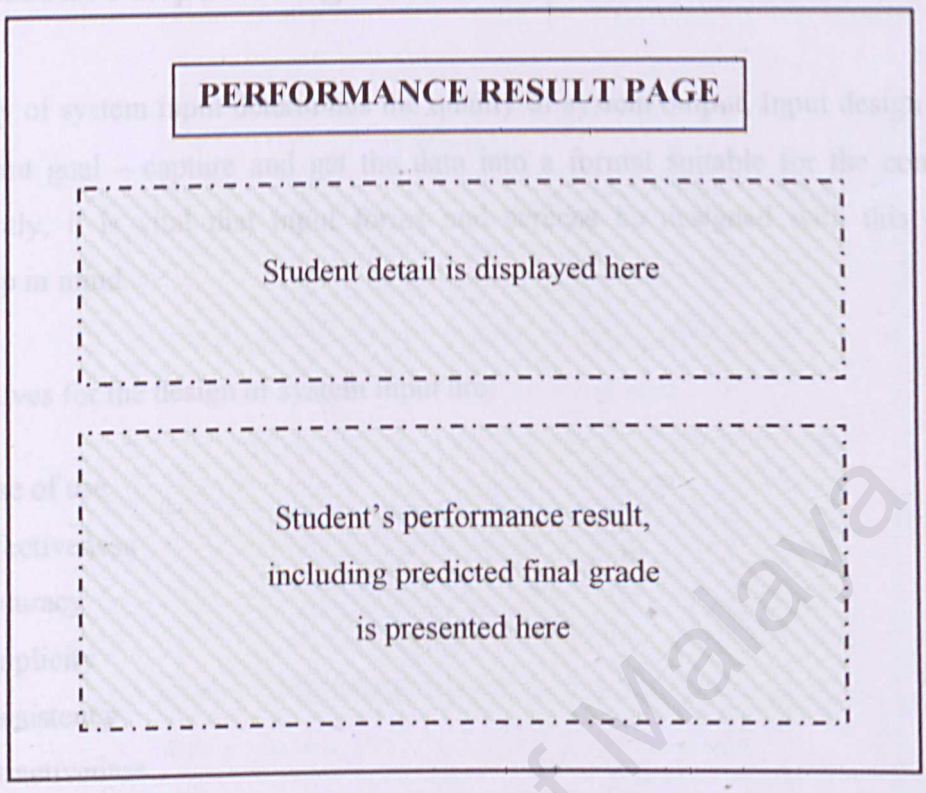


Figure 4.5: Performance Result Output

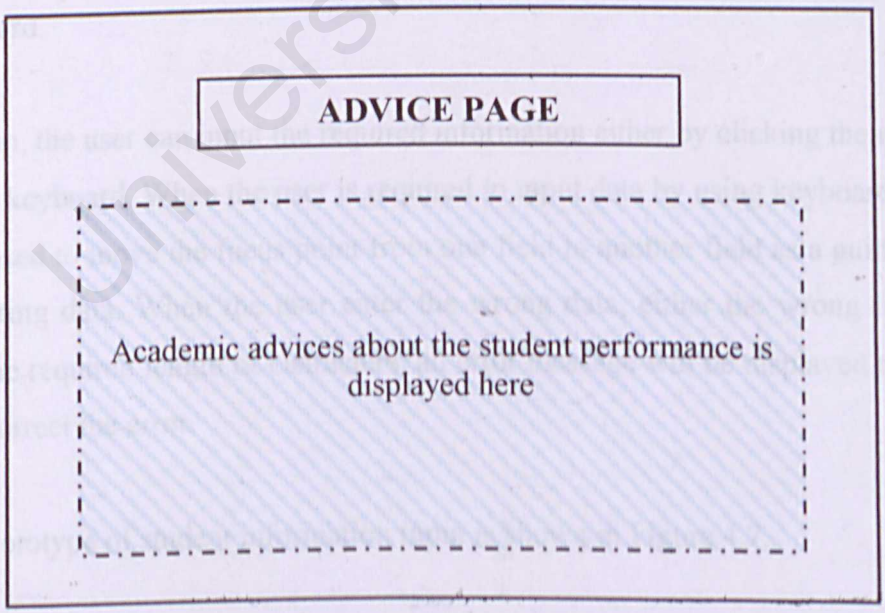


Figure 4.6: Advice Page Output

4.3 Effective Input Design

The quality of system input determines the quality of system output. Input design serves an important goal – capture and get the data into a format suitable for the computer. Consequently, it is vital that input forms and screens be designed with this critical relationship in mind.

The objectives for the design of system input are:

i. Ease of use
ii. Effectiveness
iii. Accuracy
iv. Simplicity
v. Consistency
vi. Attractiveness

The design of inputs for the system usually should be simple for the user to understand. The interaction between the user and the system should be kept simple, easy and straightforward.

In this system, the user can input the required information either by clicking the mouse or by using the keyboard. When the user is required to input data by using keyboard, the tab function is used to move the focus point from one field to another field as a guide for the user in entering data. When the user enter the wrong data, either the wrong format or exceeding the required length of characters, an error message will be displayed to inform the user to correct the error.

The input prototype of student information input is shown in Figure 4.7.

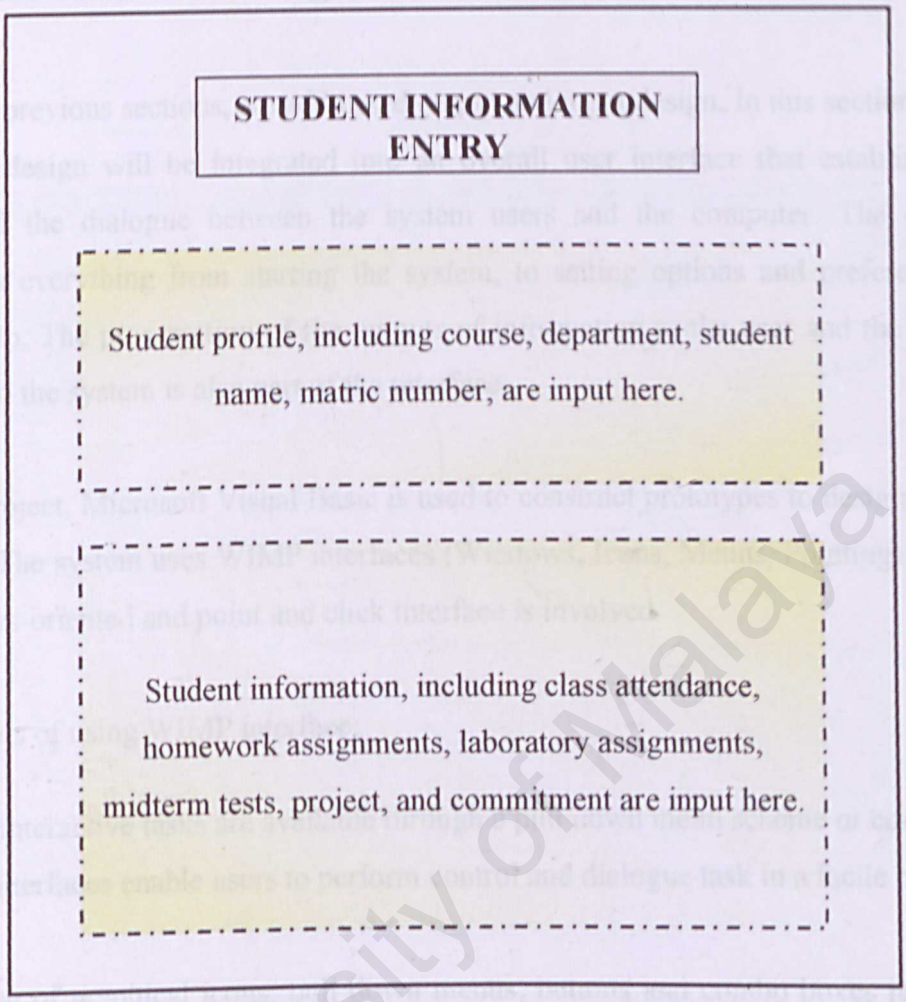


Figure 4.7: Input Prototype For Student Information

4.4 User Interface Design

In the two previous sections, we addressed output and input design. In this section, output and input design will be integrated into an overall user interface that establishes the layout and the dialogue between the system users and the computer. The dialogue determines everything from starting the system, to setting options and preferences, to getting help. The presentation of the outputs of information to the user and the input of new data to the system is also part of the interface.

For this project, Microsoft Visual Basic is used to construct prototypes to design the user interface. The system uses WIMP interfaces (Windows, Icons, Menus, Pointing) where it is Windows-oriented and point and click interface is involved.

The benefits of using WIMP interface:

- Many interactive tasks are available through a pull down menu scheme or combo box. Such interfaces enable users to perform control and dialogue task in a facile manner;

- The use of graphical icons, pull down menus, buttons and combo boxes reduce the amount of typing.

Figure 4.8 illustrates the main menu screen of the system. This main menu screen is basic constructed by the window. A window is a rectangular, boarder area. A title is displayed at the top of the window. There are some user interface controls within the window. Label 1 shows the faculty name, label 2 shows a statement to welcome the user and ask the user to select the options. Image 1 and image 2 show the UM logo and FSKTM logo respectively. There are three buttons to allow the user to click by using mouse.

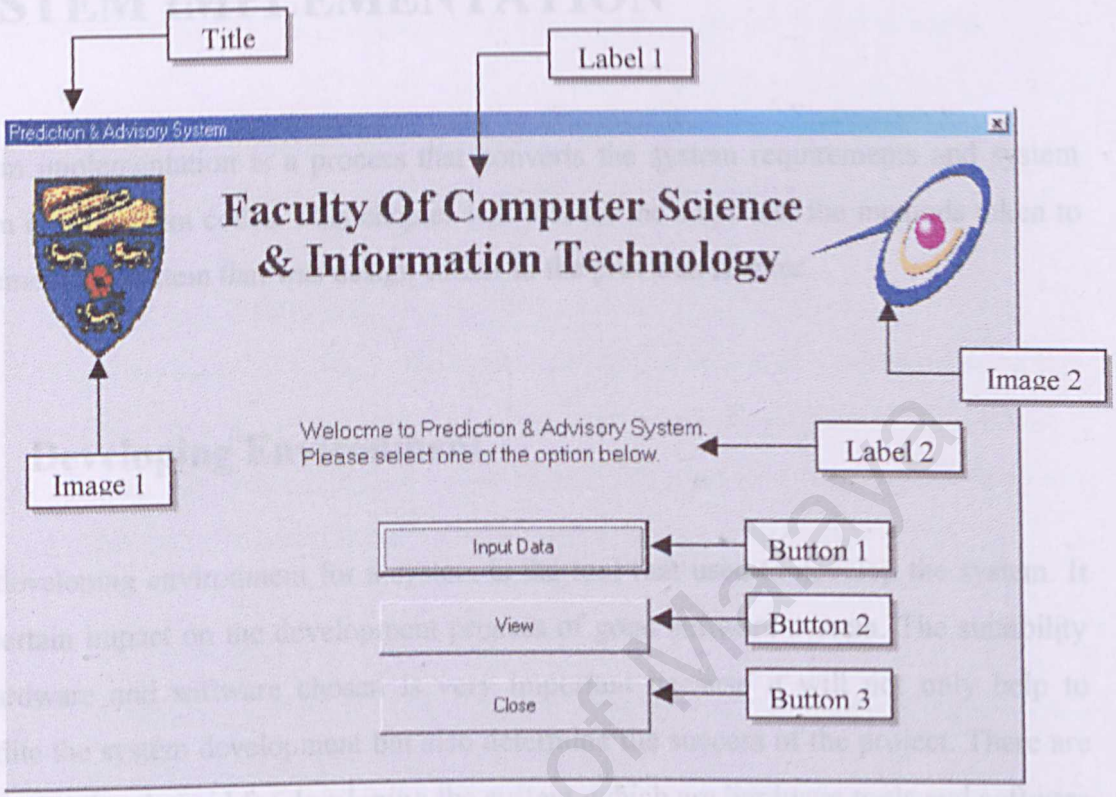


Figure 4.8: Main Menu Screen Design

CHAPTER 5

SYSTEM IMPLEMENTATION

System implementation is a process that converts the system requirements and system design into program codes. This chapter will discuss the steps and the methods taken to implement the system that was design earlier in the previous chapter.

5.1 Developing Environment

The developing environment for a system is the tool that used to develop the system. It has certain impact on the development process of good software system. The suitability of hardware and software chosen is very important because it will not only help to expedite the system development but also determine the success of the project. There are two types of tools used for developing the system, which are hardware tools and software tools that listed in the following part.

5.1.1 Hardware Tools

The hardware tools that used to develop the system are listed as below:

- Pentium III processor 500 MHz
- 256 MB RAM
- 11 GB Hard Drive Storage
- Mouse
- Keyboard

This configuration is enough to run the system.

5.1.2 Software Tools

The software that has been implemented in the standalone system are stated below:

- *Windows 98 or Windows 2000*

  This operating system is the system requirement that needed to run the program.

- *Microsoft Visual Basic 6.0*

  This is the main software for the system development and the interface design. It is used to code the system and design the interface of the program. Visual Basic offers an easy-to-use interface and language, therefore allowing the most novices of users to easily create programs to run on Windows platforms.

- *Microsoft Notepad*

  This is required to store the data of every single student record including student information, class performance and final grade.

5.2 System Implementation

System implementation is a process that comprises of system design structure to a computer readable system. The system will be evolved from scratch design to a run able application.

The implementation of this system includes three types of implementation:

(i) The first type is the *interface implementation*, which is the implementation of the front-end of the system.

(ii) Meanwhile, the second type of implementation is the *prediction implementation* in the system, which operates as the back-end of the system.

(iii) The third type is the *result and advice generation implementation*, which generates some advice about the partial student performance that user has inputted.

These implementations will be explained in the following sections.

5.3 Interface Implementation

A few criteria have been taken into account to develop a good interface. Some of the criteria are simple, user-friendly, interactive and so on. The interface that was created has fulfilled these criteria. The screen shots of the real interfaces of Student Performance Predictor can be found in Appendix A. The method for implementing the interface is described as below:

5.3.1 User Input Form

The user input form is to let the user or student to input their related information and their past performance to enable the system to predict the final grade of the subject and generate advices. There are three sections need the user to input their information, which are *student information form*, *subject form* and *student performance form*.

Student Information Form

In this form, all the student information like student name, register number, IC number and gender are entered.

Subject Form

In this form, the user is allow to select which subject does he or she want to predict.

Student Performance Form

In this form, the student performance is entered. The student performance includes tutorial 1-8, assignment 1, assignment 2, midterm test 1, midterm test 2 and class attendance. Additionally, four questions also will be asked to evaluate the performance of the student before the final exam.

5.3.2 User Output Form

The user output form display the predicted final grade and some advices about the past performance that user has inputted. This form consists of *result form* and *advice form*.

Result Form

The result form display the predicted final grade of the subject and the number of cases that compared with the new case. It also displays the three cases that most similar with the new case that the student input. The similarity degree for each case is displayed to show the similarity between the library case and the new case.

Advice Form

The academic advice is generated and display in the advice form. This advice is a guide for the student before going for the final exam.

5.4 Prediction Implementation

Sequential file processing is used in the prediction implementation. Files are used for permanent retention of large amounts of data. Data in files is said to be persistent.

Each file presents a student record, which called as a case. Each case contained 16 features, divided by two types of features sets, which are indexing features and non-indexing features. Indexing feature is an index of a case that acts like index to a book in a library. A case's index is combination of its important descriptors, the ones that distinguish it from another cases. The first three features are used to identify the student information and the following two features are used to identify the particular subject course that a certain student took. These five features are non-indexing features. The remaining eleven features denote the partially known partial class performance of the same student. Two tables below show the non-indexing features and indexing features.

Non-Indexing Features
Matric Number
Student Name
IC Number
Session
Semester

Indexing Features
Class Attendance
Tutorial
Assignment 1 (by group)
Assignment 2 (by individual)
Midterm Test 1
Midterm Test 2
Revision (hour per week)
Understanding in Lecture
Understanding of Content
Discussion Made (hour per week)
Grade

Table 5.1: Case's non-indexing features and indexing features

Given the partially known partial class performance, i.e., given a test case a test case, Student Performance Predictor's task is to predict the student's final grade for the particular subject that the student took. This was done by using the case library that is already exist in order to search the cases that are most similar to the test case. Similarity was determined by comparison of corresponding indexing features. Corresponding indexing features with identical numerical values receive a similarity count of 1 while corresponding features that the absolutely value of their difference is greater than or equal to 15% receive a similarity count of 0. If the difference is less than 15% then the similarity count is a numerical value between 0 and 1. For the corresponding features with identical non-numerical (string) values, it receives a similarity count of 1, other than that receives a similarity count of 0. The sum of the similarity counts for each feature constitutes the degree of similarity between two cases. Therefore the maximum possible match value between two cases is equal to the number of case features that is 10.0 (100.0%).

After comparisons between test case and all library cases, Student Performance Predictor will select the three library cases that most similar with the test case. It is done by selecting the three library cases with the highest similarity degree. Then an average of final grade is calculated by taking the average of the final grades of the three library cases. The average of final grade then put into the test case as a predicted final grade.

5.5 Advice Generation Implementation

In this section, advice will be generated to the user or student after the result of the prediction has been produced. The advice will be generated according to the past performance of the student and the predicted final grade.

5.6 Internal Structure

The following sections show the internal structures of some important forms in the system code. Several main procedures will be discussed for each form.

MDIMain MDI Form

Procedures	Actions
MainToolbar_ButtonClick	Do the particular event when a button on toolbar is clicked.
mnuAbout_Click	Display program info, version number and copyright.
mnuExit_Click	Terminate the program.
mnuPrediction_Click	Start student performance prediction.
mnuShowWizard_Click	Show or unload wizard form at startup.
mnuToolBar_Click	Show or not show the main toolbar.
mnuViewRecord_Click	Open an existing student record.
mnuWizard_Click	Show wizard.
Timer_Timer	Display the splash screen at regular interval.

frmAbout Form

Procedures	Actions
cmdSysInfo_Click	Try to get system info and program path\name from registry.
cmdOK_Click	Close About form.

frmEditPerformanceII Form

Procedures	Actions
cmdUpdate_Click	Update the survey answer.

frmInputPerformance Form

Procedures	Actions
cmdBack_Click	Go back to the previous form.
cmdCancel_Click	Cancel prediction session.
cmdNext_Click	Continue the performance prediction.
cmdOK_Click	Update the record.

frmInputStudentInfo Form

Procedures	Actions
cmdCancel_Click	Cancel prediction session.
cmdNext_Click	Continue the performance prediction.
cmdOK_Click	Update the record.
cmdSearch_Click	Search the student record.
mskMatricNo_LostFocus	Convert the contents of the matric number to uppercase letter.
txtStudentName_LostFocus	Convert the contents of the student name to uppercase letter.

frmInputSubject Form

Procedures	Actions
cboSubjectCode_Click	Select a subject.
cmdBack_Click	Go back to the previous form.
cmdCancel_Click	Cancel prediction session.
cmdNext_Click	Continue the performance prediction.
cmdOK_Click	Update the record.

frmLogin Form

Procedures	Actions
cmdCancel_Click	Cancel the login to view record.
cmdOK_Click	Enter and view record if the specific record exists.

frmResult Form

Procedures	Actions
form_Load	Display three most similar cases and calculate the average of final grade.
cmdViewAdvice_Click	View an advice about the student performance.

frmSummary Form

Procedures	Actions
cmdCancel_Click	Cancel current prediction session.
cmdEditPerformance1_Click	Edit student performance.
cmdEditPerformance2_Click	Edit student performance.
cmdEditStudentInfo_Click	Edit student information.
cmdEditSubject_Click	Edit subject criteria.
cmdFinish_Click	Process the prediction and save the student record.
lstQuestion_Click	Display the specific survey question and answer.
lstSummary_Click	Display the specific criteria of the student.

frmSurvey Form

Procedures	Actions
cmdCancel_Click	Cancel current prediction session.
cmdNext_Click	Continue the performance prediction.

frmViewStudentRecord Form

Procedures	Actions
cmdClose_Click	Close the form.
lstSubjectCode_Click	Display the performance and final grade of the selected subject.

frmWizard Form

Procedures	Actions
chkLoadWizard_Click	Save whether or not this form should be displayed at startup.
cmdAbout_Click	Display program info, version number and copyright.
cmdCloseWizard_Click	Close wizard.
cmdExit_Click	Terminate the program.
cmdStartPrediction_Click	Start student performance prediction.
cmdView_Click	Open an existing student record.

CHAPTER 6

SYSTEM TESTING

Software testing refers to verification and validation of the program to solve the problem. Verification involves ensuring that the characteristics of a good design are incorporated into the program and the system is actually operates in the way it is expected to be. On the other hand, validation is used to test the execution of the program and system meets the requirements.

The major focus of testing is to find the faults within the program that are not realized at the time of coding. A test is successful only when a fault is discovered or when a failure has been encountered. 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 or unit testing, integration testing, function testing, performance testing, acceptance testing and installation testing. Each of these stages will be discussed individually.

6.1 Module/Unit Testing

The module testing is a way to verify that the small unit function properly with the types of input expected from the design. It has been carried out under a controlled environment where a predetermined set of data has been provided to the modules. In other words, this kind of testing is used to observe what input and its related output actions as well as the data produced.

In module testing, each of the sub-modules is tested separately. After that, each of the modules will be tested in turn for the creation of the user interface, the input data handling and output to data files, reset and exit from the modules to make sure those modules do exactly what it were designed to perform. Test cases have been developed to show that the input is properly converted to the desired output.

6.2 Integration Testing

After the collections of modules have been tested, the next step is to ensure that the interface among the components are defined and handled properly. Thus, integration testing will be performed to achieve this. Integration testing is the process of verifying that the system modules work together as described in the system and program specification.

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 are isolated first before they are combined into a functional program in the system and tested together.

For this system, the testing approach that has been applied in the integration testing is the bottom-up integration. Each component at the lowest level of the system hierarchy is tested individually first. Here, each of the sub-modules is tested individually first, then the modules in which comprise of sub-modules are tested in turn. Finally, after the integration of the modules into a functional program, the main program is tested to ensure that the system performs its works correctly.

6.3 Function Testing

After the integration testing is conducted, the function testing should be carried out to assure that the system has the desired functionality. The function test will evaluate the system to determine if the functions described in the requirement specification are actually performed by the integrated system.

6.4 Other Testing

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

Performance test compares the integrated components with the non-functional system requirements such as security, accuracy, speed and reliability. Acceptance test is ran by the users of the system to assure them that the system they wanted was the system that was built for them. The installation test allows users to exercise the system functions and document additional problem that result from being at the actual site.

CHAPTER 7

SYSTEM EVALUATION AND CONCLUSION

In this final chapter, the whole system will be evaluated in a few areas. It covers the system strength and the limitation of the system. A few of suggestion will be also stated to enhance the system in the future. There is a thorough discussion of the problems encountered and knowledge gained during developing the project. Finally, the conclusion of the system will be made.

7.1 System Strength

The advantages and value-added functionalities of Student Performance Predictor are listed as below:

i. ***Simple Interface***

The interfaces of this system are quite simple in design and also user-friendly to the user. This simple design let the user to input their data more easily.

ii. ***Transparent to user***

The system is transparent by enabling the user to see the working memory and its prediction output as the prediction session progresses. This provides reliable to the user.

iii. ***Intelligent Searching***

This system perform an intelligent search in the case library to select the most similar cases with the new case that user entered. After the searching, the system will display the data of the most similar cases to the user in order the user can see the similarity between the new case and the cases that displayed.

7.2 System Limitations

System limitations are the shortcomings of Student Performance Predictor. The following lists the limitations of the system:

i. ***Generated advice not strong enough***

Does not provide further and strong advice and explanation about the student performance. The advice that generated by Student Performance Predictor is fully dependent of the partial performance of the student and the predicted final grade.

ii. ***Limited scope***

The scope of Student Performance Predictor is limited to the AI student. The system cannot be used to predict final grades of other subjects.

iii. ***Inflexible prediction***

The prediction of a final grade is fully depending to the past cases in the case library and no adaptation is made during prediction progress. This will produce some "incorrect" output.

iv. ***Lack of control in Visual Basic***

Visual Basic is a revolutionary software development tool that offers an easy-to use interface and language, therefore allowing the most novices of users to easily cerate programs to run a Windows platform. These features also translate to making Visual Basic a quick prototyping tool for more experienced programmers. The drawbacks to this situation are that Microsoft had to restrict access to lower level functions of the windows kernel (as compared to other visual studio tools such as Visual C++).

Many of the required functions of this program had to written explicitly as declared functions and calls to the windows kernel library to access lower level functions because they were not provided in Visual Basic. This was especially true when the program had to capture specific keystrokes and key combinations on the MDI form.

7.3 Future Enhancement

Student Performance Predictor would be a complete system if the limitations mentioned earlier were to be rectified. As such, future candidate enhancements for the system include:

- ***Strong advice generation***
  Generation of an advice should be dependent on the relation between the retrieved and current case, that is from the past performance and the current performance generate a good advice.

- ***Expand to all the computer science and IT students***
  This system can be expanded to all computer science and IT students. A powerful database is needed to store all the subjects, student and lecturer information. In addition, this kind of database is easier to manipulate existing data such as adding, deleting and modifying.

- ***Enhance the prediction by implement adaptation***
  A structural adaptation technique is needed for the prediction. It compares specified parameters of the retrieved and current case to modify the solution in an appropriate direction.

7.4 Problems Encountered And Solutions Taken

There are a lot of problems encountered during the development of the entire course of the project, either non-technical problem or technical problem. Some of the problems encountered have the solutions. The following parts are the problems encountered and the solution during development of the project.

- ***Problem:*** Unsure of the type of interface design of Student Performance Predictor
  - ***Solution:*** Observed other system such as Adobe Photoshop 6.0 to get an idea of interface design

- ***Problem:*** Lack of experience in the programming language Visual Basic.
  - ***Solution:*** Do some trials and errors on the coding to make it work and referred to some reference book.

- ***Problem:*** Difficulty in evaluating student performance in Student Performance Predictor.
  - ***Solution:*** Observed websites to obtain some information about student performance evaluation and assessment.

- ***Problem:*** Confusion during coding because of complexity of the inference engine of Student Performance Predictor.
  - ***Solution:*** Proper documentation and comments in the code were added. Use of pseudo-codes.

- ***Problem:*** System failures encountered during incremental testing.
  - ***Solution:*** Extensive debugging process and immediate rectification.

7.5 Knowledge Gained

I have gained a lot of useful knowledge as well as experience in various aspects during developing the project. The knowledge gained is listed as below:

- ***Knowledge of using Microsoft Visual Basic 6.0.***
  It brings me a lot of improvement in programming skill, especially in Visual Basic. I discovered that to develop a systematic and organized program, some practices have to be considered. Firstly, design and draw a program structure before doing the actual coding. It makes the programming task easier and systematic. Secondly, design a good and attractive interface that makes the program be user-friendly to the users. Thirdly, list down all the relevant procedures of the program, then code the program systematically. Lastly, document the program and prepare a user manual to help users in using the program.

- ***Knowledge of Case-based Reasoning***
  Case-based reasoning is a intelligent technique to solve new problem by finding solved problems similar to the current problem and adapting their solutions to the current problem, taking into consideration any differences between the current and previously solved situations.

- ***Knowledge of applying sequential file processing***
  Sequential file processing is one of the most important methods to develop Student Performance Predictor. It introduces me the data hierarchy from bits, to bytes, to fields, to records, to files. I learned how to store data sequentially in a file and how to read data sequentially from a file.

7.6 Conclusion

Throughout the whole development of ***Student Performance Predictor***, from its initial requirements gathering phase to the final system-testing phase, no major obstacles were encountered.

Based on the belief that analogues may provide a way to predict results grounded on what has been true in the past, I develop a prototype intelligent system that uses case-based reasoning in order to forecast a student's performance in academic. Case-based reasoning technique provides a faster solution in the prediction of student performance. The system draws conclusions on the basis of similarities between a student's current partial class performance and the performance of other students that took the same subject course. The reasoner maintains a library of cases and operates within the traditional case-based reasoning phrases of input, matching, retrieval and storage.

As a conclusion, *Student Performance Predictor* was successfully developed according to the initial system scope and requirements definition. It is not only that all requirements and functionalities were implemented into the system; other value-added features were also included to make the system more usable, transparent and efficient.

APPENDIX A: SCREEN SHOTS

Shown below are several screen shots of actual interface of *Student Performance Predictor.*

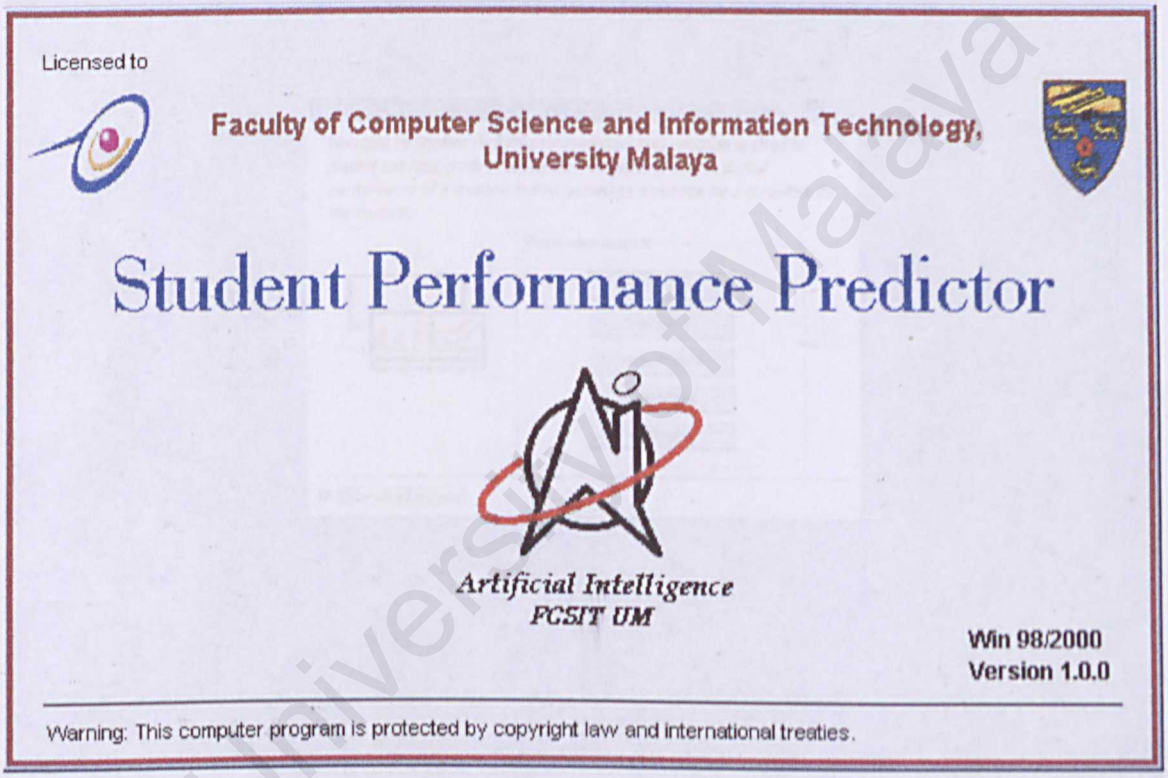


Figure A1: Splash Screen

APPENDIX A

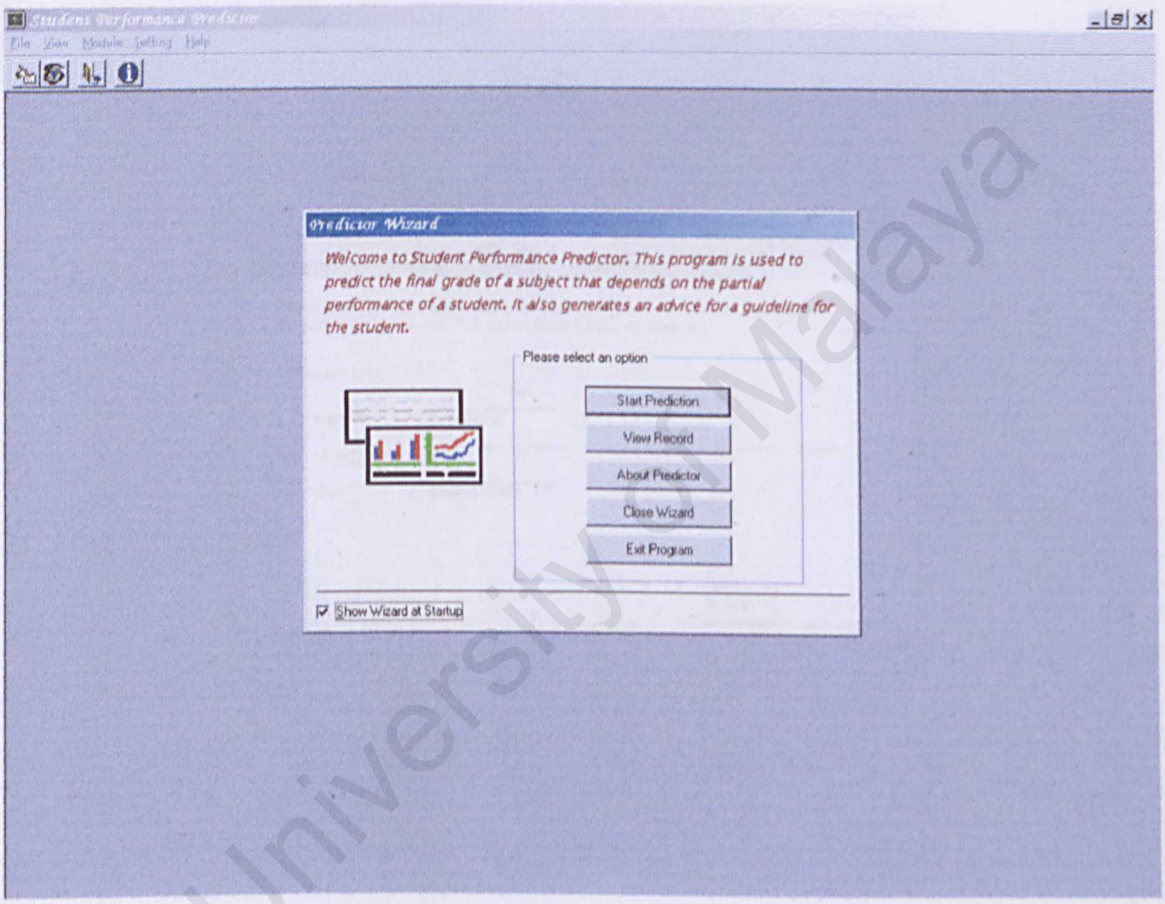


Figure A2: Student Performance Predictor Wizard

APPENDIX A

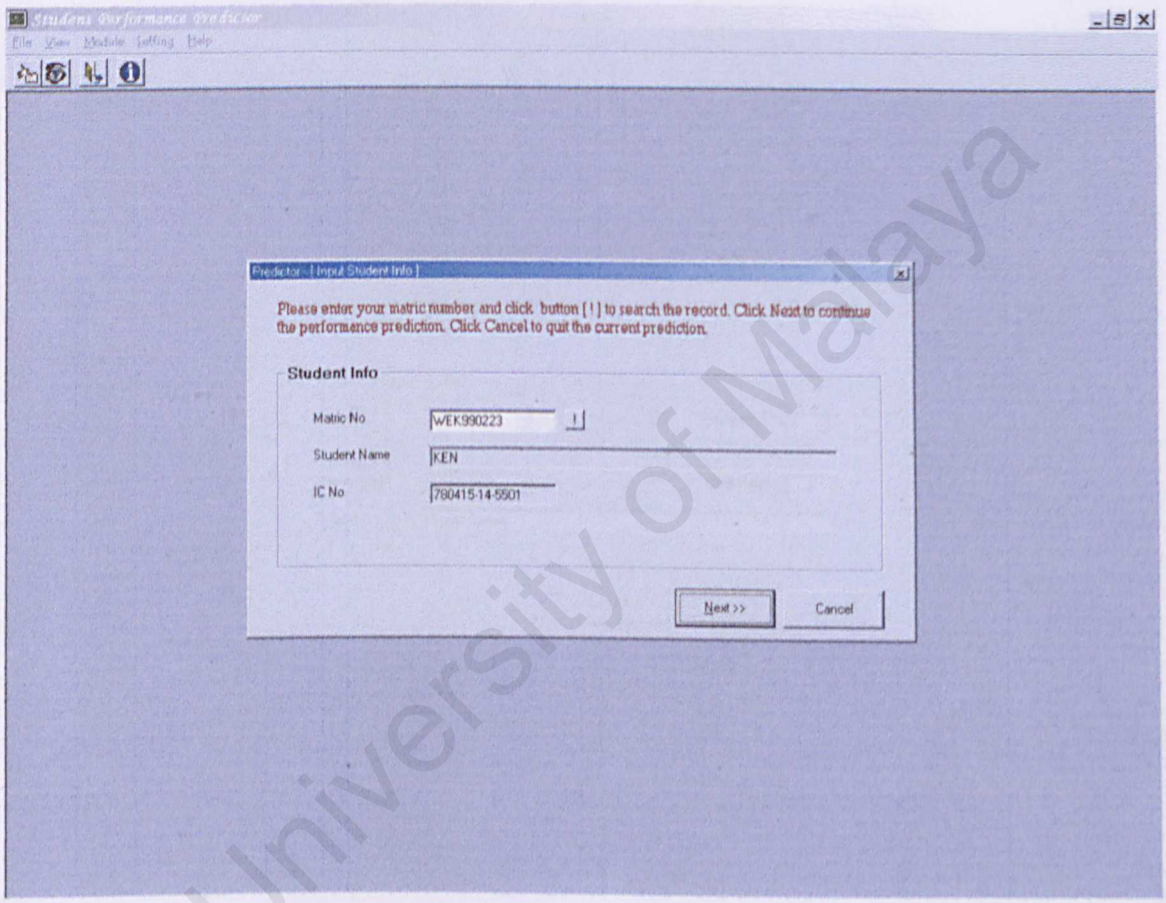


Figure A3: Input Student Info

APPENDIX A

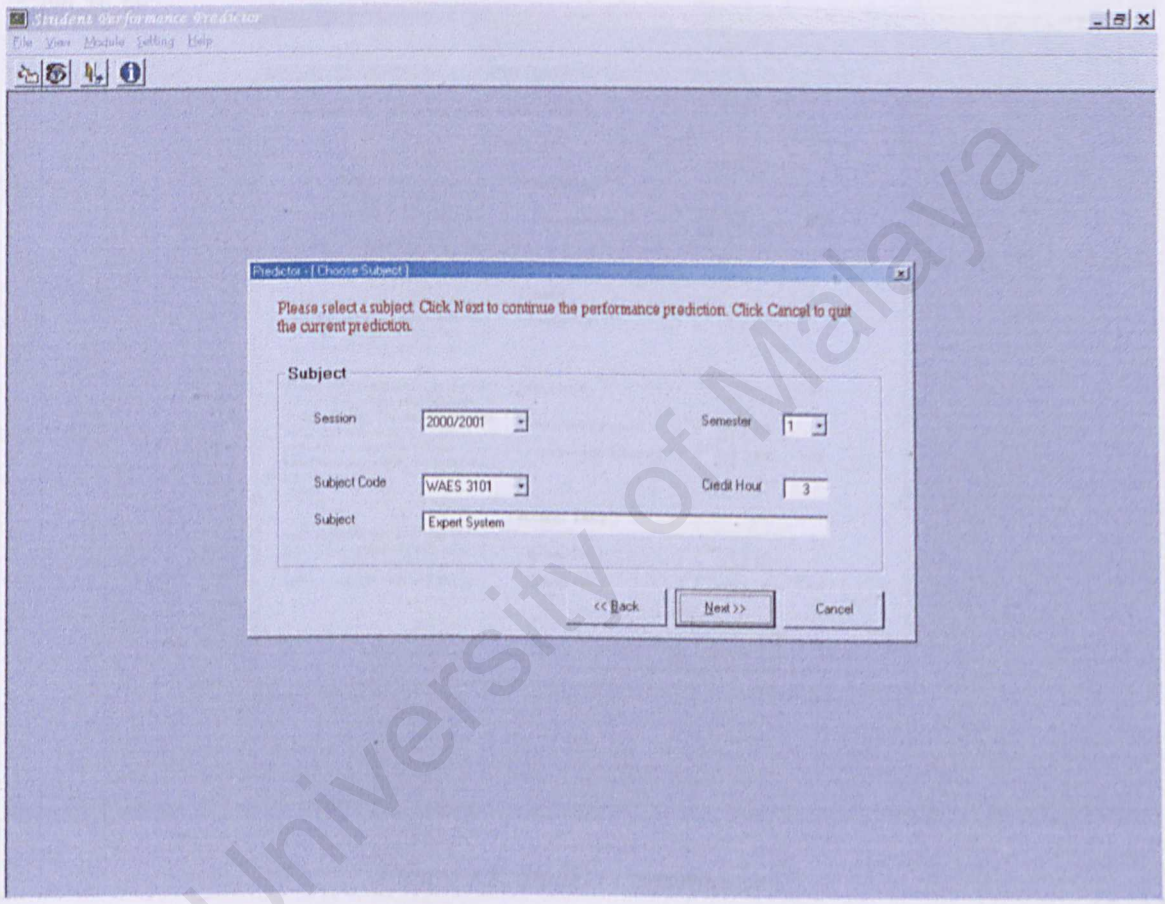


Figure A4: Input Subject

APPENDIX A

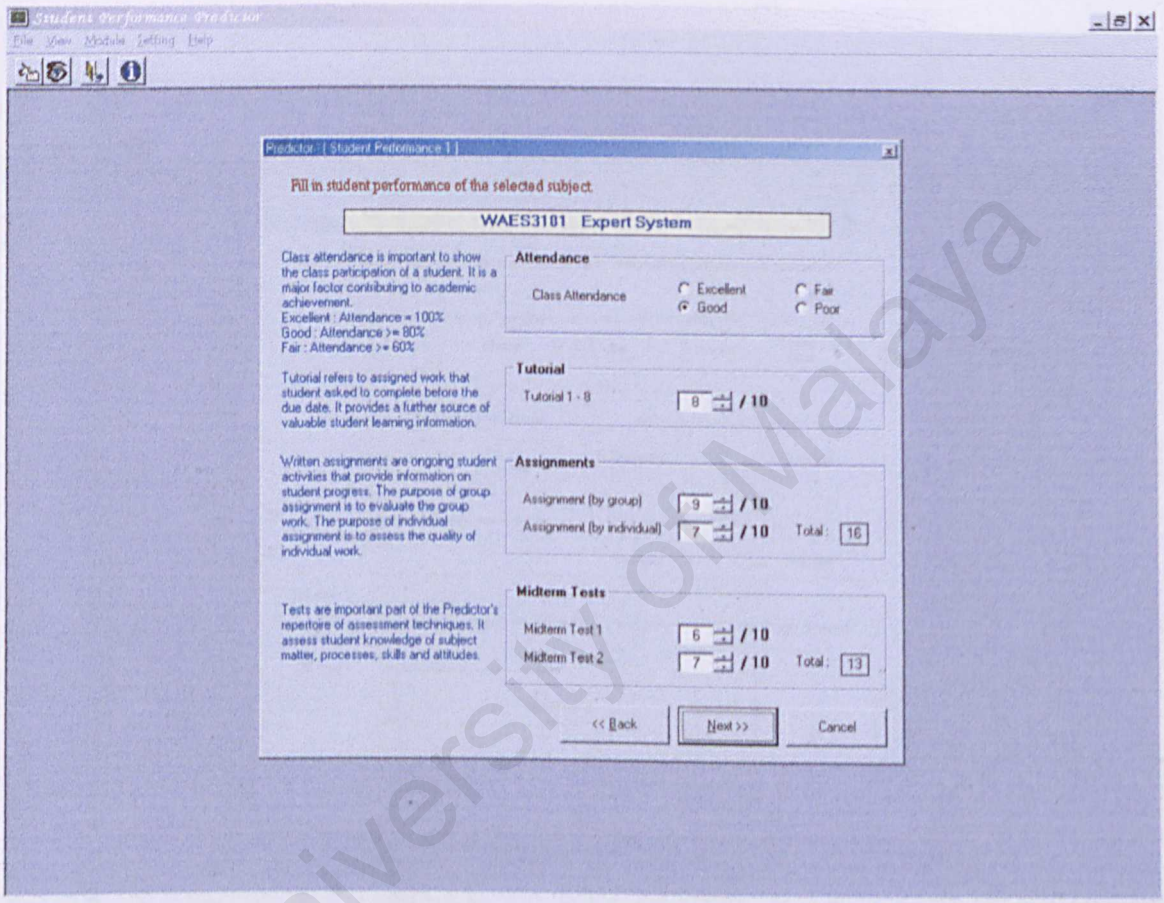


Figure A5: Student Performance 1

APPENDIX A

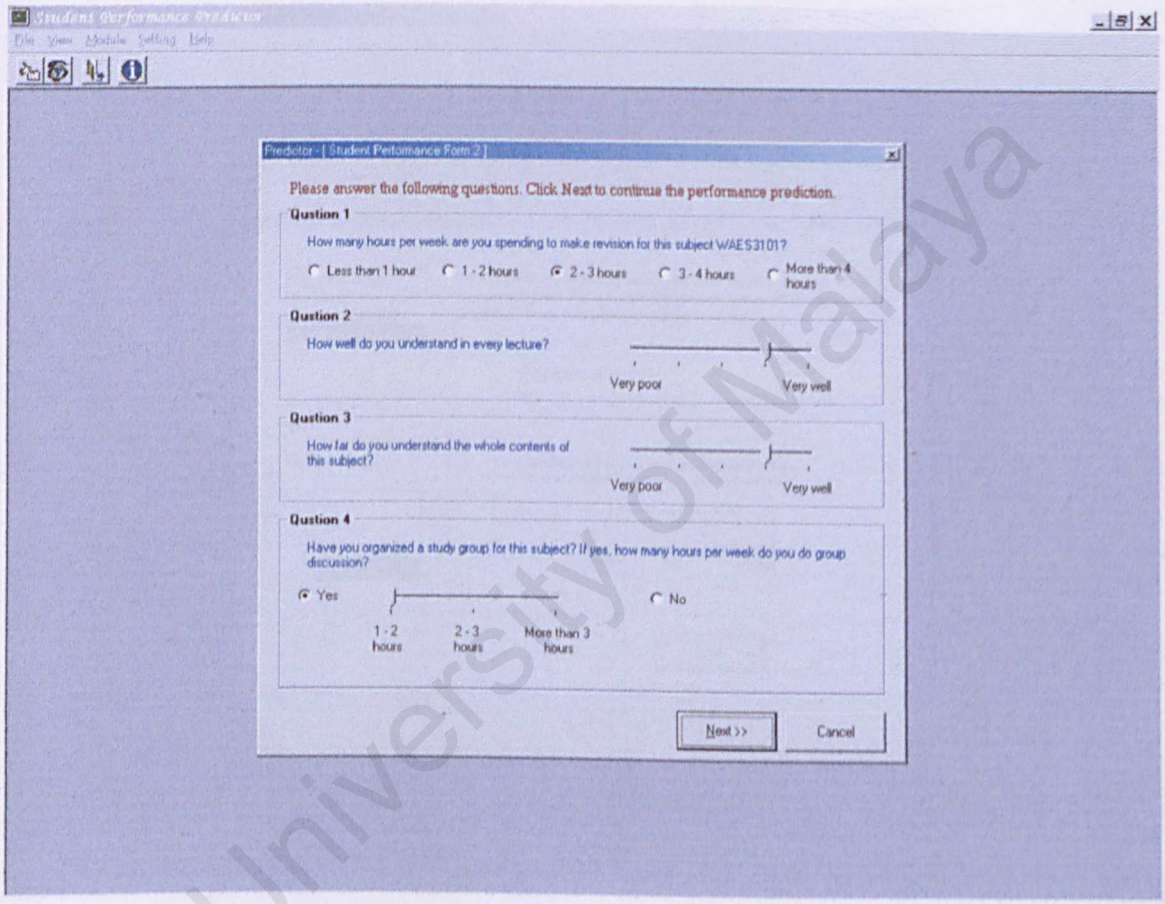


Figure A6: Student Performance 2

APPENDIX A

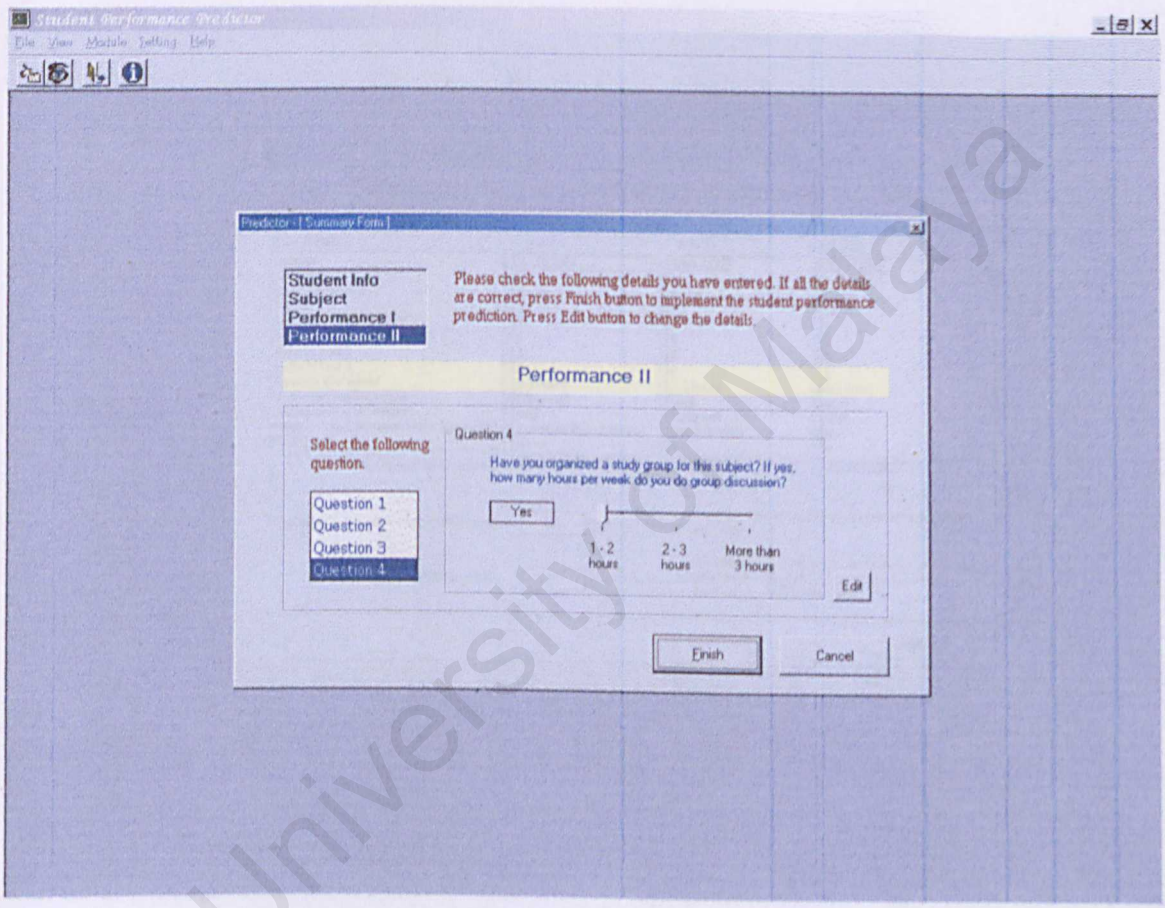


Figure A7: Summary of User Record

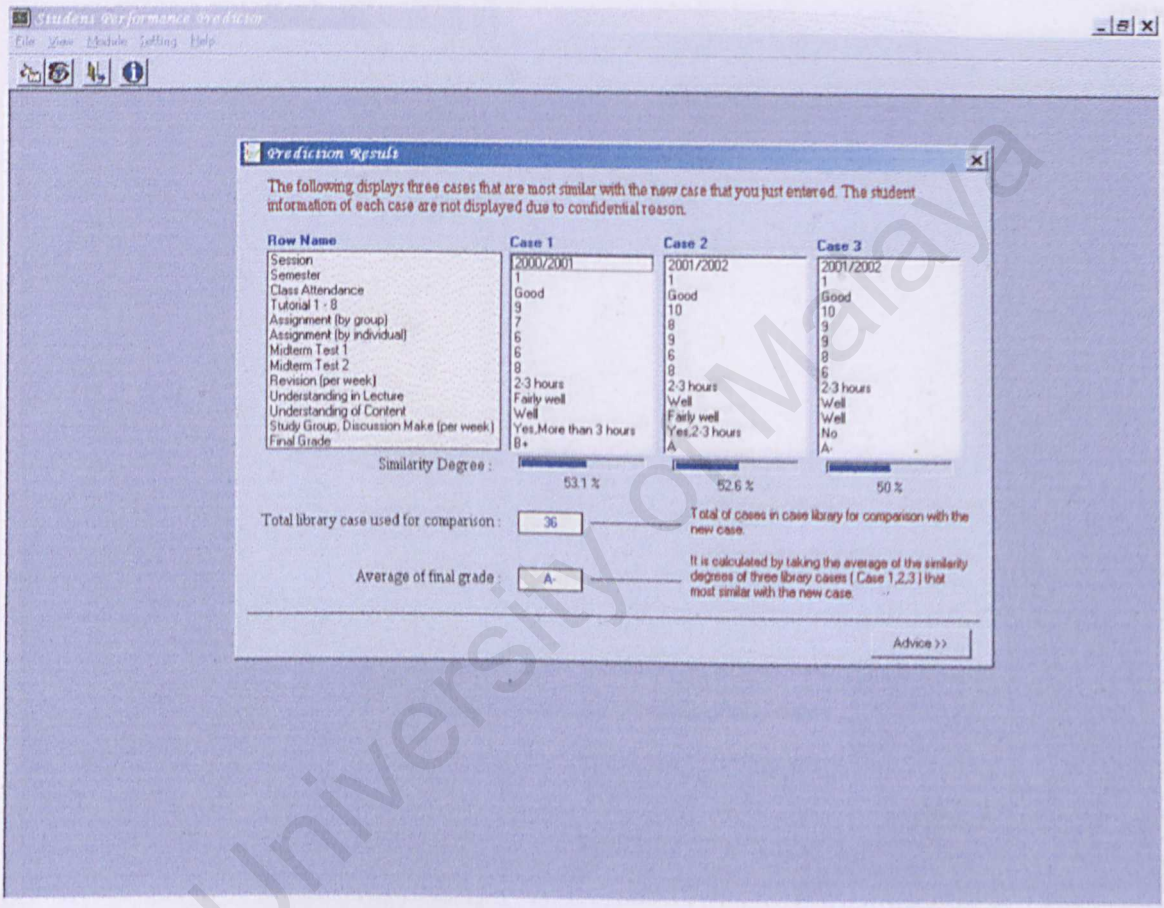


Figure A8: Prediction Result

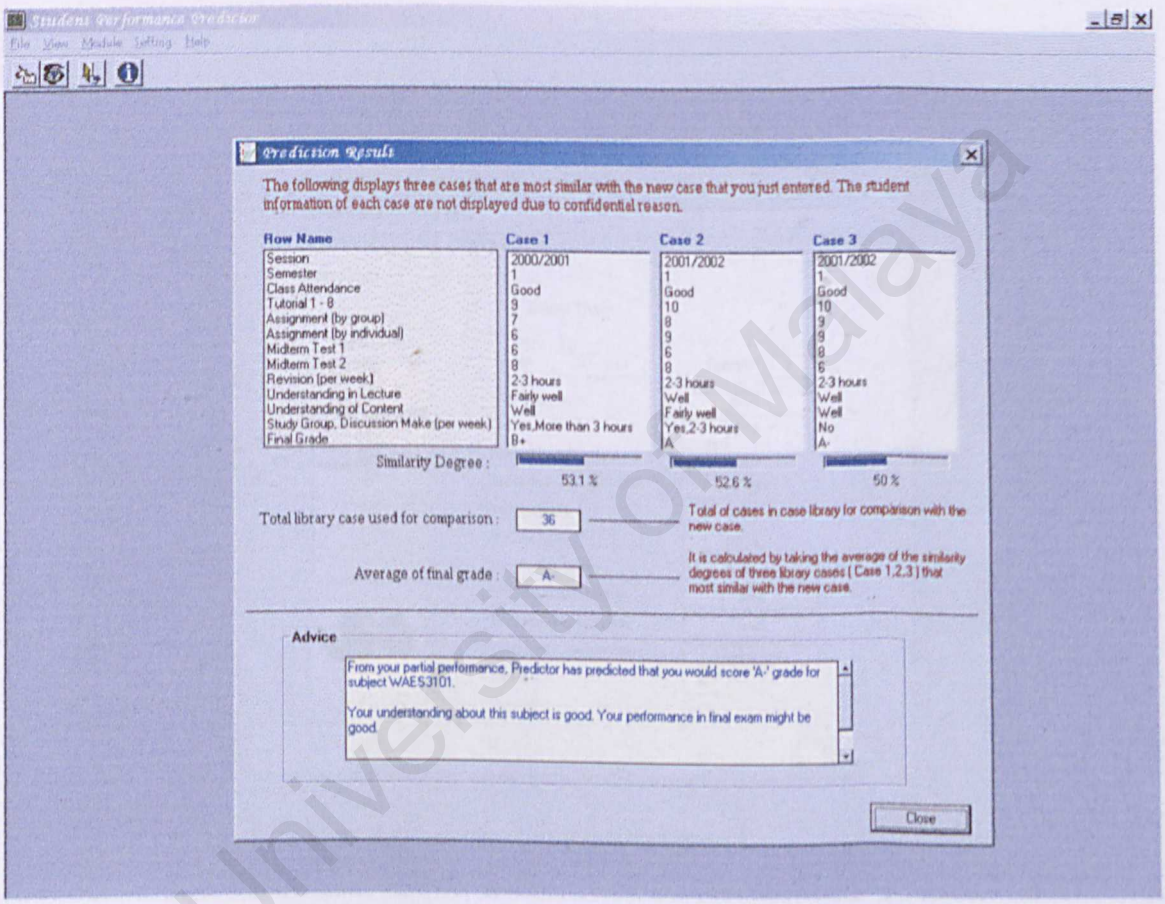


Figure A9: Prediction Result and Advice

APPENDIX A

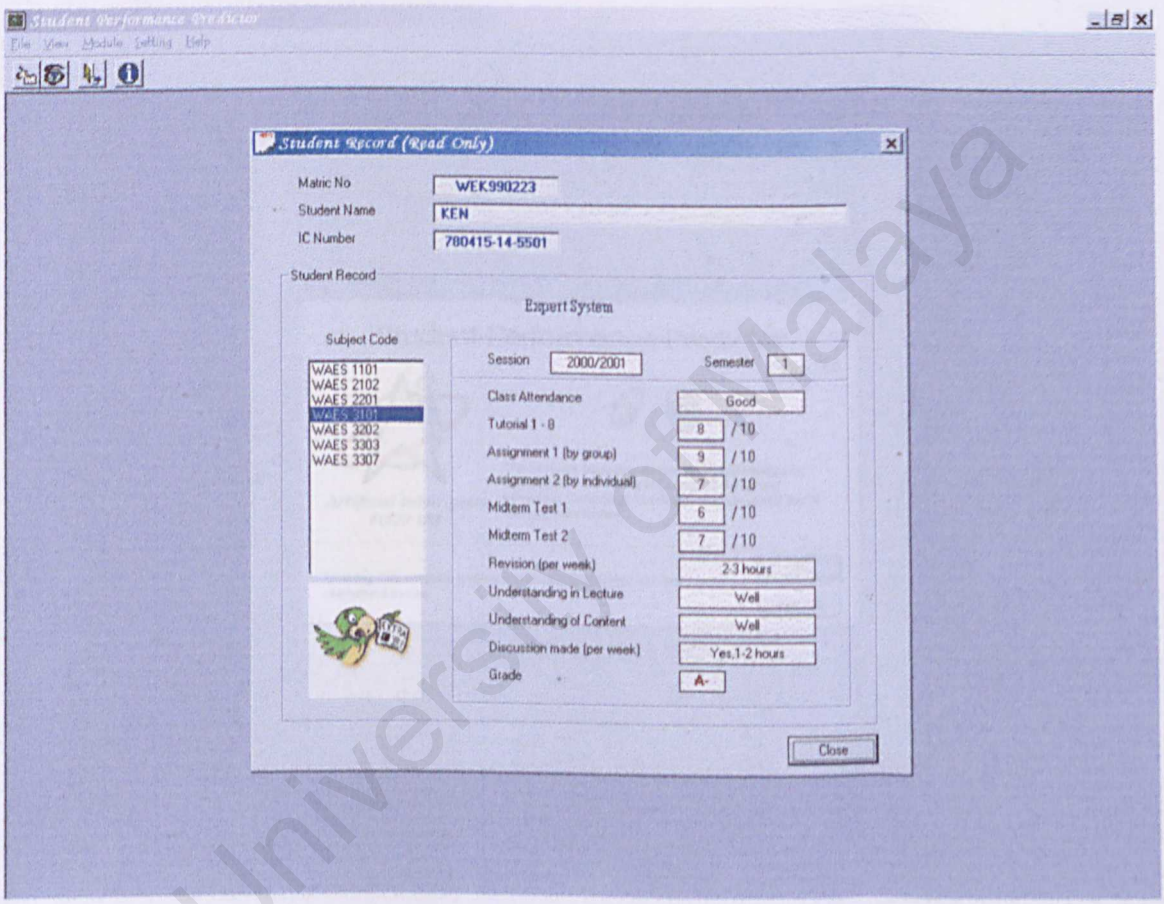


Figure A10: Student Record (Read only)

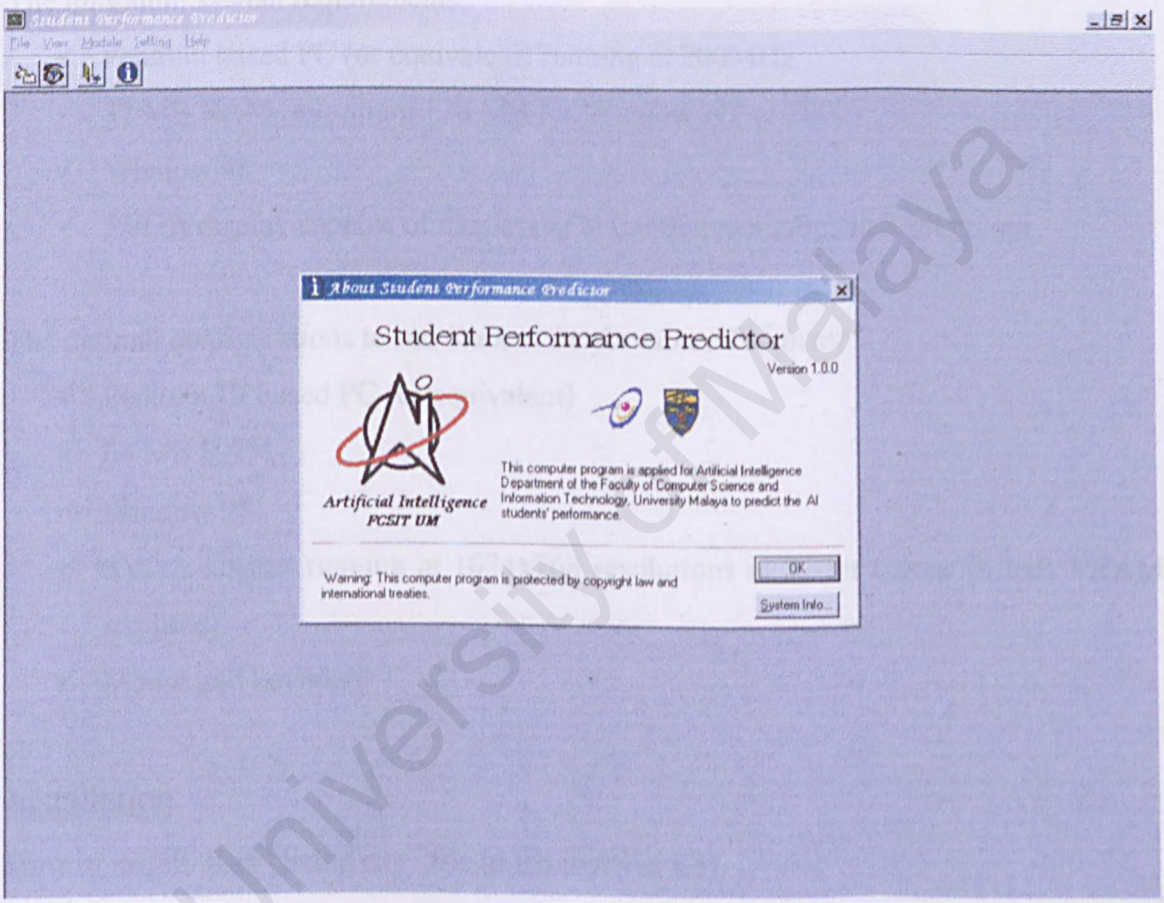


Figure A11: About Student Performance Predictor

APPENDIX B: USER MANUAL

System Requirements

The minimum system requirements:

- ✓ Pentium based PC (or equivalent) running at 200MHz
- ✓ 32 MB RAM (minimum 128 MB for Window NT or 2000)
- ✓ Window 98
- ✓ SVGA display capable of displaying 800x600 resolutions in 8-bit colour

The optimal configurations to run *Student Performance Predictor*:

- ✓ Pentium III based PC (or equivalent)
- ✓ 64 MB RAM
- ✓ Window 98
- ✓ SVGA display running at 1024x768 resolutions in 32-bit colour (8 MB VRAM required)
- ✓ Mouse and keyboard

Installation

How to install: Run "Setup.exe" file in the installer CD.

Uninstalling

To uninstall *Student Performance Predictor* follow these steps:

1. Click on the Start menu
2. Choose Programs
3. Select *Student Performance Predictor*
4. Click on the Uninstall Icon

Splash Screen

This is the first screen when the user starts the program *Student Performance Predictor*. This splash screen displays the faculty, university, program title, version number, logo and copyright. It appears for a few seconds before a desktop of the program turns up.

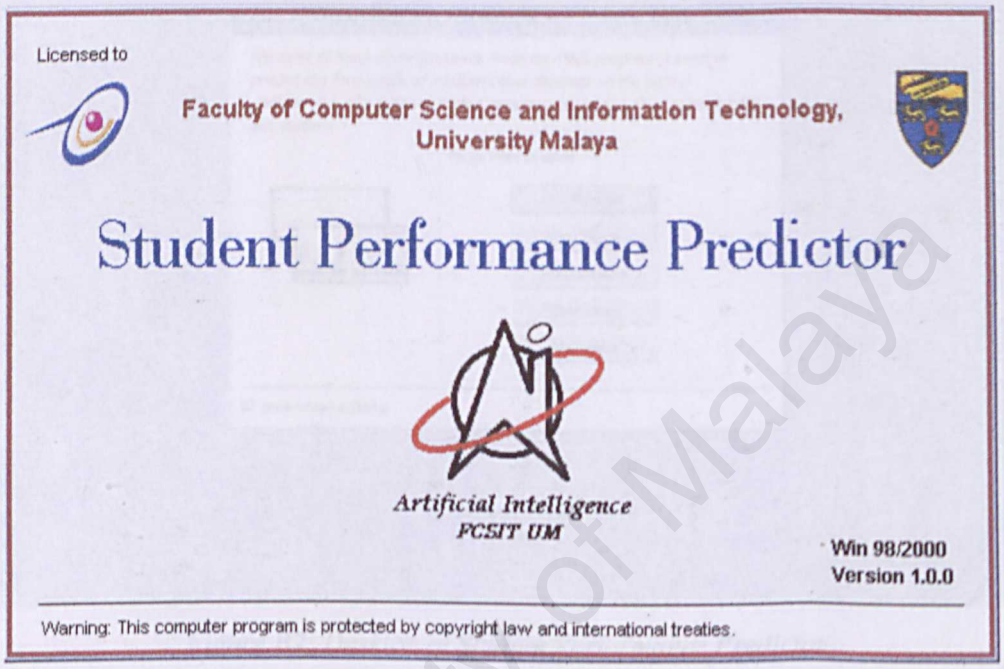


Figure B1: Splash Screen

Desktop

The desktop of *Student Performance Predictor* mimics an actual desktop where work is done. All modules in *Student Performance Predictor* are loaded into the desktop using either the buttons on the wizard, the desktop own menu bar or the icon buttons on the toolbar.

Toolbar

 Start performance prediction.

 Open an existing student record.

 Exit the program.

 Display program information, version number and copyright.

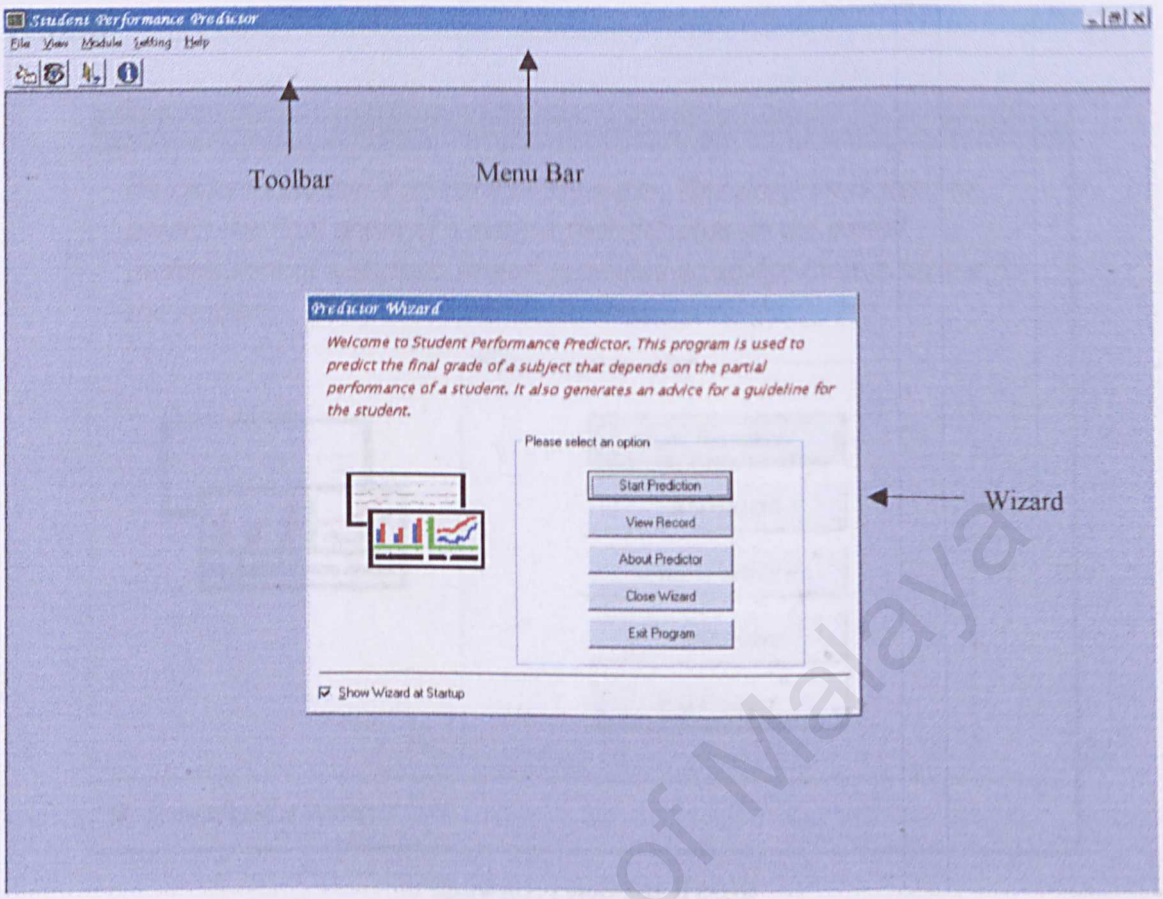
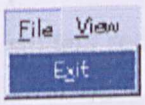
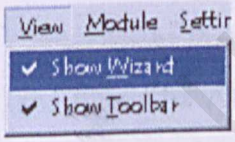


Figure B2: Desktop of Student Performance Predictor

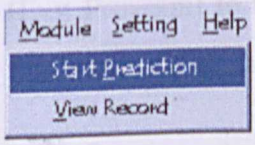
Menu Bar



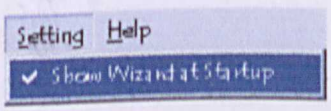
Click [Exit] to exit the program.



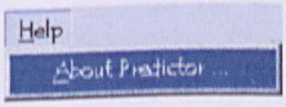
Tick [Show Wizard] to display wizard on the desktop.
Tick [Show Toolbar] to show toolbar on the desktop.



Click [Start Prediction] to Start performance prediction.
Click [View Record] to open an existing student record.



Tick [Show Wizard at Startup] to show Predictor wizard at startup of program.



Tick [About Predictor…] to display the about form.

Wizard

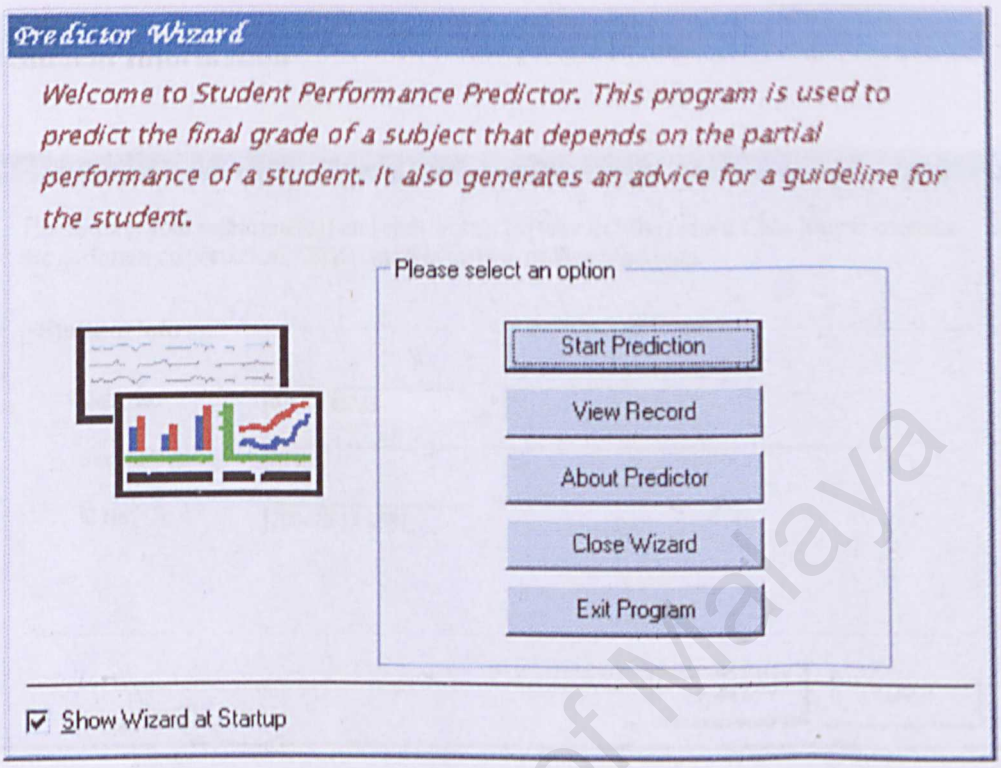


Figure B3: Predictor Wizard

There are five buttons on the Predictor wizard. If you want to start your performance prediction, click [Start Prediction]. You can click [View Record] to open an existing student record.

Click [About Predictor] to show the program information, version number and copyright (see Appendix A). You can close the Predictor wizard by clicking [Close Wizard]. Click [Exit Program] to terminate the program. Tick the check box on the left bottom corner will show the Predictor wizard at startup of the program.

Start Student Performance Prediction Module

Input Student Information

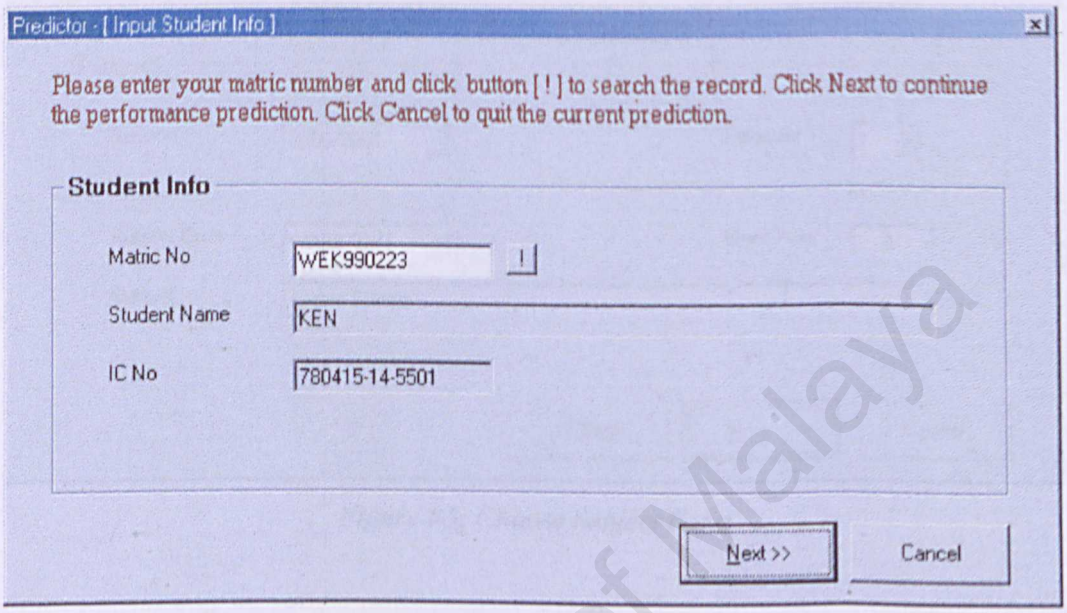


Figure B4: Input Student Information Form

When you start performance prediction, an Input Student Info form will be displayed. You are asked to input your matric number, student name and IC number.

You can click button [!] to search your existing information. If no information available, then you have to input your name and IC number. Then click [Next] to continue student performance prediction. Click [Cancel] if you want to quit the current prediction session.

Choose Subject

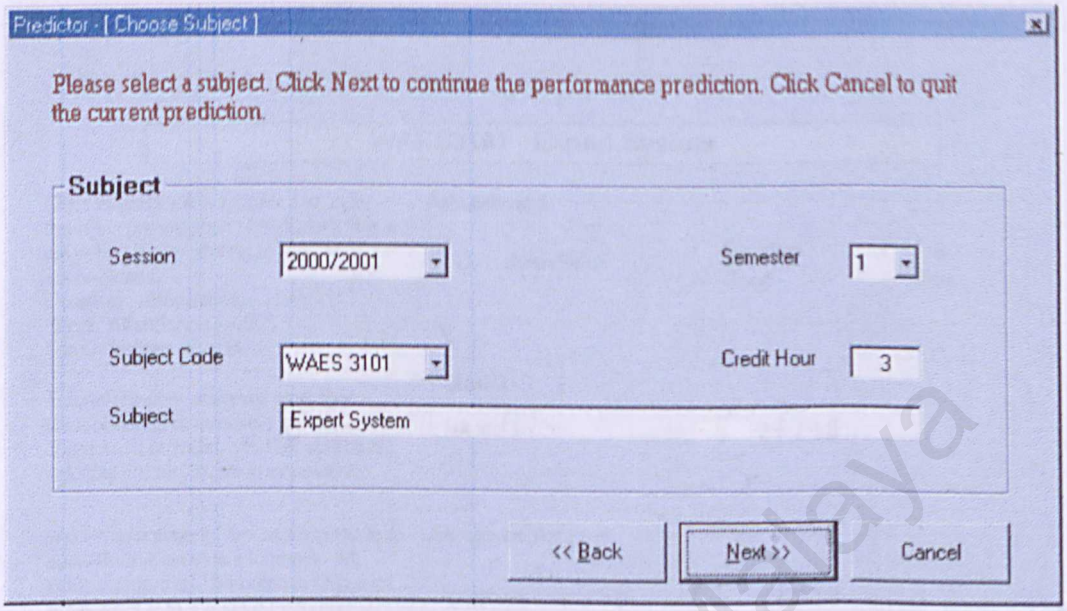


Figure B5: Choose Subject Form

After entering student information, a Choose Subject form will be shown to ask you to choose a subject you want the system predict.

Select session and semester that the subject you took. Click [Next] to continue student performance prediction. You can click [Back] to reenter your information in the previous form. Click [Cancel] to quit the current prediction session.

Enter Student Performance (Part 1)

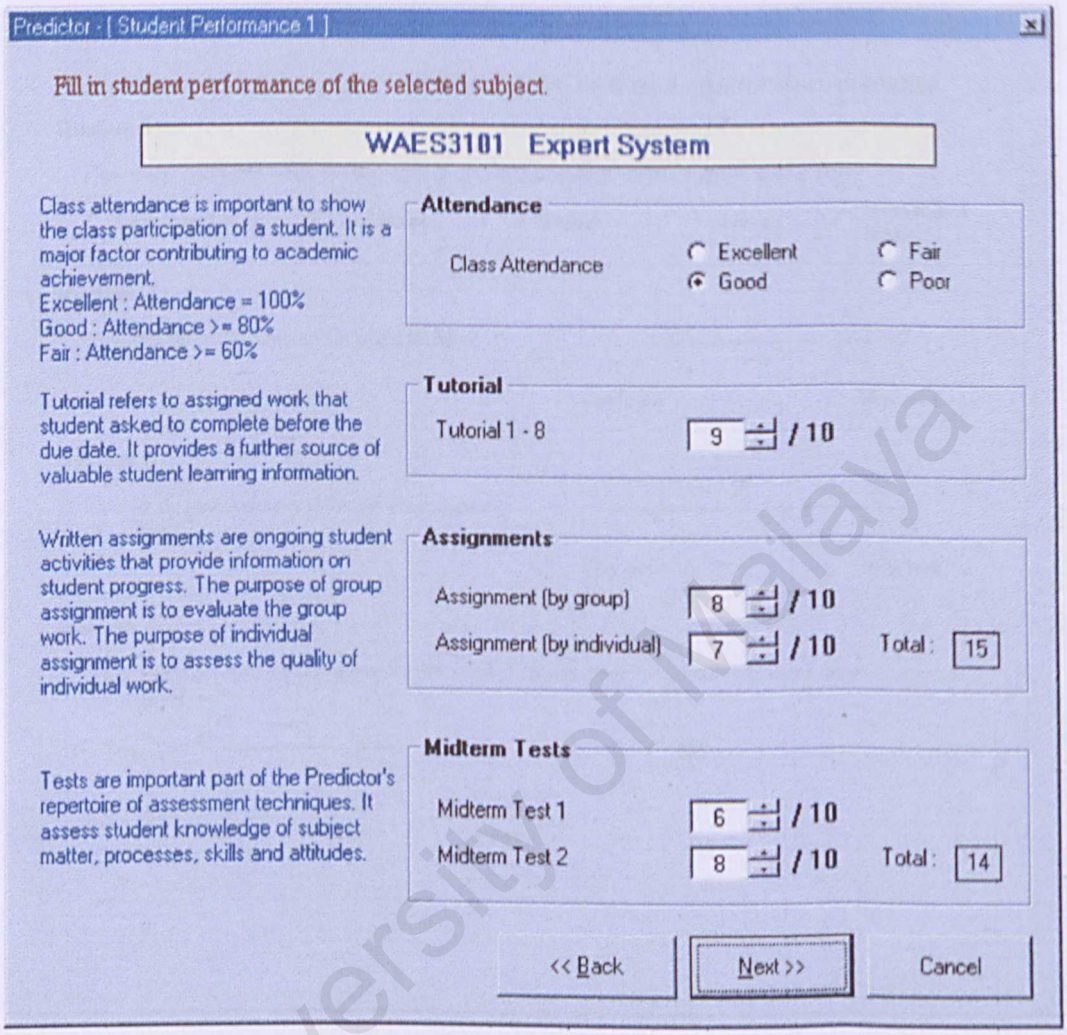


Figure B6: Student Performance (Part 1)

A Student Performance (Part 1) form will be pop up after choosing subject. This form allows you to enter your partial performance in the subject course you just selected. There is an explanation for each criterion of performance. After finish entering, click [Next] to continue student performance prediction. Click [Back] to reselect subject in the previous form and click [Cancel] if you want to quit the current prediction session.

Enter Student Performance (Part 2)

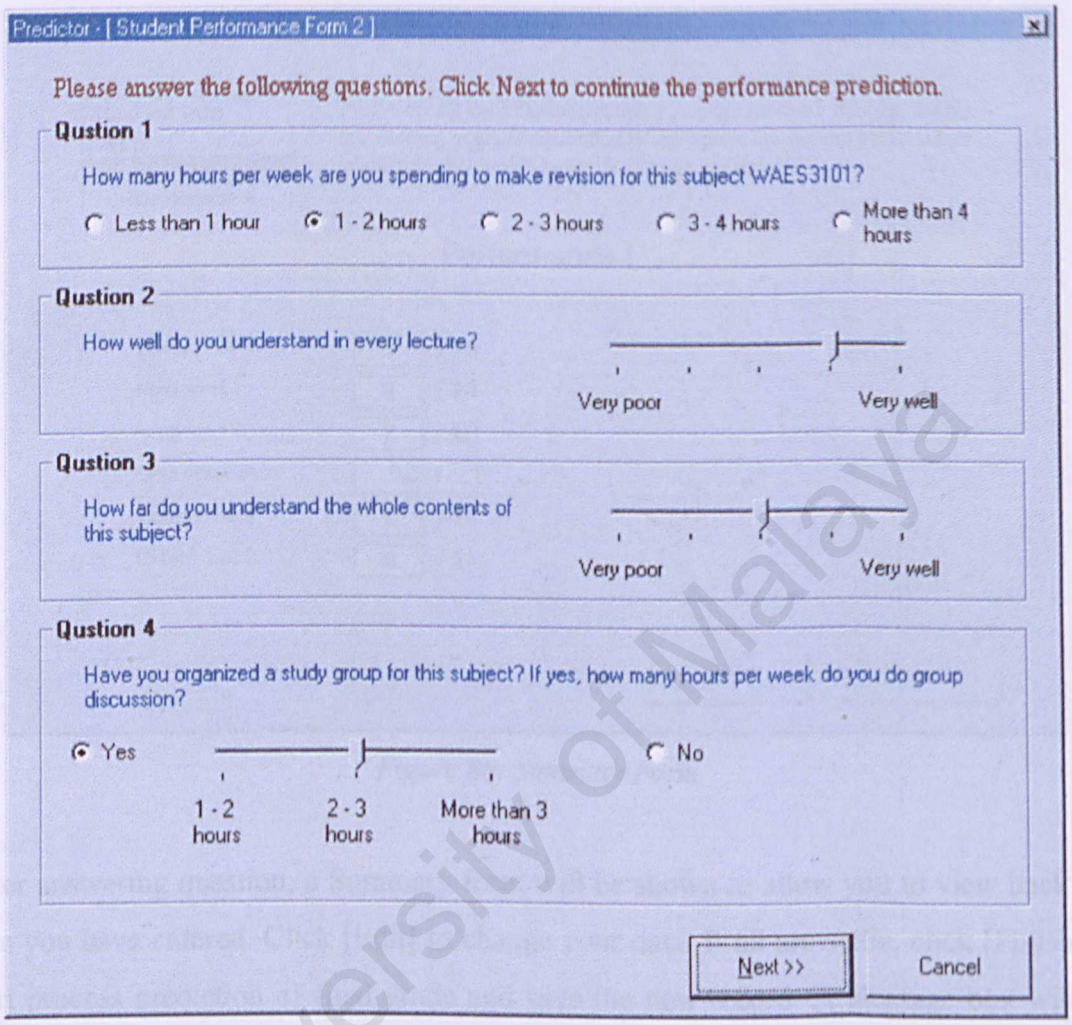


Figure B7: Student Performance (Part 2)

A Student Performance (Part 2) form will be pop up after Student Performance (Part 1). This form asks you to answer four simple questions that related to your performance before the final exam. In question 1, select how many hour per week you make revision. In question 2 and question 3, select understanding degree by dragging the slider to a specific value. In question 4, select Yes if you have organized a study group for the subject course, or else select No. After answering all the questions, click [Next] to continue or click [Cancel] to quit the current prediction session.

Summary of record

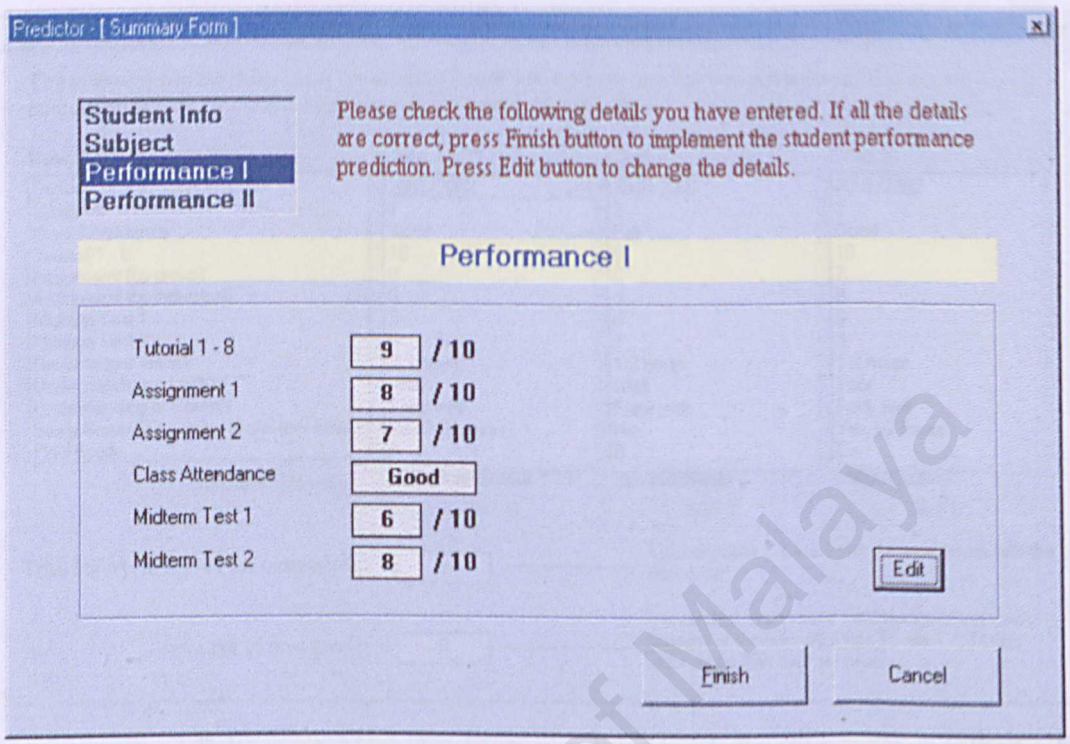


Figure B8: Summary Form

After answering question, a Summary form will be shown to allow you to view back the data you have entered. Click [Edit] to change your data. If all are settle, click [Finish] to start process prediction of final grade and save the new record. A message box will be pop up to inform you that the new record has been saved.

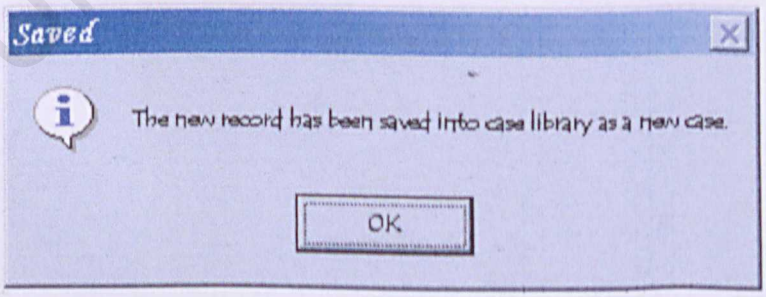


Figure B9: Message Box show the new record has been saved

Prediction Result

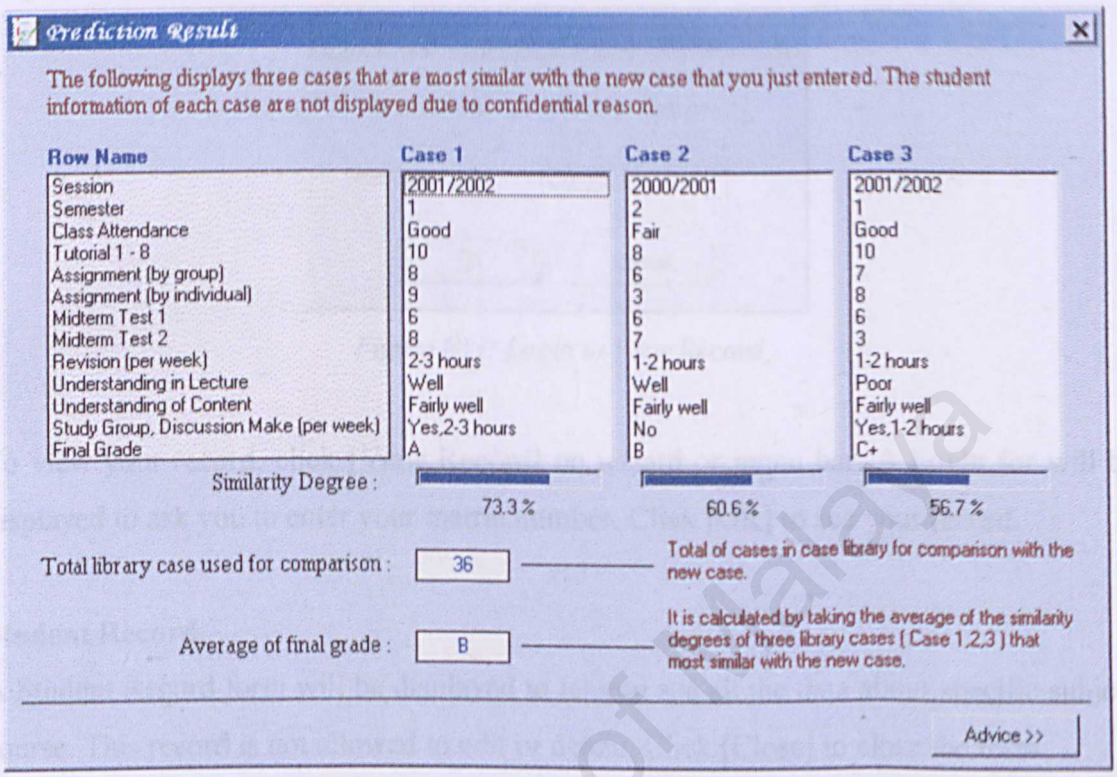


Figure B10: Prediction Result

Finally, a Prediction Result form will be shown to display the result of prediction. The data of three most similar cases are displayed and the similarity degree shows the similarity with the your record (as a new case). Click [Advice>>] to see advice about your performance.

View Student Record

Login

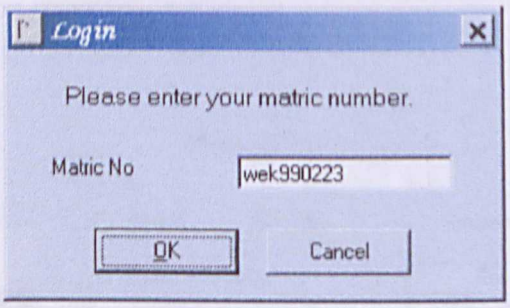


Figure B11: Login to View Record

To view your record, click [View Record] on wizard or menu bar. A Login for will be displayed to ask you to enter your matric number. Click [OK] to see your record.

Student Record

A Student Record form will be displayed to let you see all the data about specific subject course. This record is not allowed to edit or delete. Click [Close] to close the form.

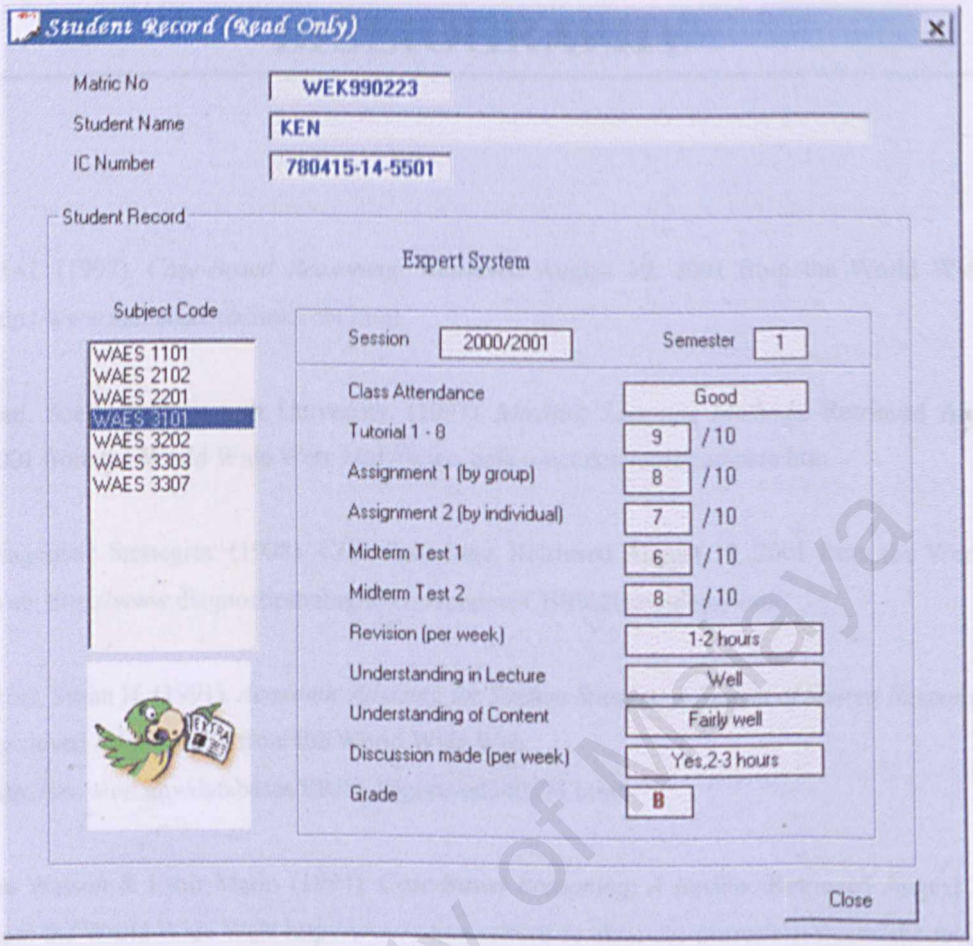


Figure B12: Student Record (Read Only)

BIBLIOGRAPHY

[1] AIAI. (1997). *Case-Based Reasoning*. Retrieved August 19, 2001 from the World Wide Web: http://www.aiai.ed.ac.uk/links/cbr.html

[2] Dan. Snell, Bournemouth University. (1997). *Machine Learning Methods*. Retrieved August 19, 2001 from the World Wide Web: http://www.ecfc.u-net.com/cost/compare.htm

[3] Diagnostic Strategies. (1998). *CBR Taxonomy*. Retrieved August 19, 2001 from the World Wide Web: http://www.diagnosticstrategies.com/papers/CBR%20taxonomy.htm

[4] Frost, Susan H. (1991). *Academic Advising for Student Success: A System of Shared Responsibility*. Retrieved July 25, 2001from the World Wide Web: http://www.ed.gov/databases/ERIC_Digests/ed340274.html

[5] Ian Watson & Fahir Marir. (1994). *Case-Based Reasoning: A Review*. Retrieved August 20, 2001 from the World Wide Web: http://www.scpm.salford.ac.uk/ai-cbr-mirror/classroom/cbr-review.htm

[6] K.E. Kendall and J.E. Kendal. (1995). *System Analysis and Design* 3rd ed. (pp. 230-265, 580). London: Prentice Hall.

[7] Kolodner, J.L. (1993). Adaptation Methods and Strategies. *In Case-Based Reasoning* (pp. 393-437). Morgan Kaufmann.

[8] Kolodner, J.L. (1993). Building A Case-Based Reasoner. *In Case-Based Reasoning* (pp. 529-537). Morgan Kaufmann.

[9] Kolodner, J.L. (1993). Indexing Vocabulary. *In Case-Based Reasoning* (pp. 197-202). Morgan Kaufmann.

[10] Kolodner, J.L. (1993). Methods For Index Selection. *In Case-Based Reasoning* (pp. 257-258). Morgan Kaufmann.

[11] Kolodner, J.L. (1993). Methods For Index Selection. *In Case-Based Reasoning* (pp. 266). Morgan Kaufmann.

[12] Kolodner, J.L. (1993). Methods For Index Selection. *In Case-Based Reasoning* (pp. 268-270). Morgan Kaufmann.

[13] Kolodner, J.L. (1993). Representing Cases. *In Case-Based Reasoning* (pp. 146-160). Morgan Kaufmann.

[14] Kolodner, J.L. (1993). The Cognitive Model. *In Case-Based Reasoning* (pp. 99-105). Morgan Kaufmann.

[15] Kolodner, J.L. (1993). What Is Case-Based Reasoning?. *In Case-Based Reasoning* (pp. 3-6). Morgan Kaufmann.

[16] Kolodner, J.L. (1993). What Is Case-Based Reasoning?. *In Case-Based Reasoning* (pp. 8-14). Morgan Kaufmann.

[17] Kolodner, J.L. (1993). What Is Case-Based Reasoning?. *In Case-Based Reasoning* (pp. 25-27). Morgan Kaufmann.

[18] Kuan Lee Huat. (1999/2000). *School Counseling Unit Administration System (COUNSAS).* Unpublished degree's thesis, University of Malaya, Kuala Lumpur.

[19] Louis E. Frenzel, Jr. (1987). Making Computers More Useful by Making Them Smarter. *In Crash Course in Artificial Intelligence and Expert Systems* (pp. 1-3). USA: Howard W. Sams & Co.

[20] Pfleeger, S.L., "Software Engineering Theory and Practice", Washington, Prentice-Hall International, 1998, page 48-51, 57, 135-144, 192-195, 244-254, 401-408.

[21] Mark F. Goldberg. (1998). *How to Design an Advisory System for a Secondary School.* Retrieved July 25, 2001 from the World Wide Web: http://www.ascd.org/readingroom/books/goldberg98book.html

[22] Mark McVea. (1994). *CBR - Case Based Reasoning.* Retrieved August 20, 2001 from the World Wide Web: http://abe.www.ecn.purdue.edu/~engelb/abe565/cbr.htm

[23] Railsback, Gary & Colby, Anita. (1988). Improving Academic at the Community College. Retrieved July 25, 2001 from the World Wide Web: http://www.ed.gov/databases/ERIC_Digests/ed320647.html

[24] Washington College. *The Advisory System*. Retrieved July 25, 2001 from the World Wide Web: http://advising.washcoll.edu/body/sys_intro/stud_peer.html