INTELLIGENT TUTORING SYSTEM FOR LEARNING OBJECT ORIENTED PROGRAMMING LANGUAGE

BY

VICTOR ACHOR OKPANACHI

ID: 40574

MASTER of Computer Science

A thesis submitted to the faculty at African University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the

Department of Computer Science

June, 2019.

APPROVAL BY

Supervisor

Surname: Mohamed

First name: Hamada

Signature: 

The Head of Department

Surname: David

First name: Amos

Signature: 

DECLARATION

I, Victor Achor Okpanachi, declare that the work presented in this thesis which is tagged: ***'Intelligent tutoring system for learning Object Oriented programming language(s)'*** submitted to the Department of Computer Science, African University of Science and Technology, Abuja, in accordance to the fulfillment of the requirements for the award of the Master of Science (M.Sc.) in Computer Science. I have neither copied/plagiarized nor submitted the same work for the award of any other degree. In case this undertaking is found incorrect, my degree may be withdrawn unconditionally by the University.

Date: June 03, 2019 Victor Achor Okpanachi

Place: Abuja 40574

ACKNOWLEDGEMENT

I will like to extend my heart-felt gratitude to God Almighty, the Father, Son and Holy Spirit, who has been awesome to me in my academic pursuit.

My sincere appreciation goes to my parents, Mr. & Mrs. Joseph Okpanachi for their love, care, encouragement, and invaluable assistance throughout the period of this program. I deeply recognize and appreciate the exceptional sacrifices they have made to make not just this degree a success, but indeed my whole life. You are the best parents in the world. I also want to appreciate my siblings, Lydia, David, Rejoice and Emma for their constant support and encouragement. You have been my friends and partners in progress, and a blessing to me. I thank God for you, and may God increase you in all spheres.

I will like to at this juncture, appreciate my awesome supervisor, Prof. Hamada. I lack words to appreciate all the love, care and concern you have shown towards me. You made writing this project pleasurable and fun-filled for me, with all your advice, encouragement and how to overcome challenges. Thank you so much sir, and may you continue to prosper in all your deeds.

I will like to extend my heart-felt gratitude to my H.O.D Prof. Amos David and my faculty officer Dr. Rajesh Prasad for their care, concern, love and undivided attention all through this program. Indeed, you both have served as a father figure to me. God bless.

Finally, to all 2019 Computer Science set, I love you all and it is a great delight to have you guys as friends and colleagues. As we all continue in this journey of life and pursuit in all our endeavors, I wish us all great success and to get to fulfill our dreams.

ABSTRACT

Taking Nigeria as a case study, most educational institutions, be it at the primary, secondary or tertiary level are faced with the challenge of over population of student's in a single classroom. Also, the teacher who teaches an over populated class finds it quite difficult to have a one on one interaction or communication with each student in order to learning challenges. As a result of that, most students find it difficult to understand in the classroom.

This research is concerned with the study of Intelligent Tutoring System (ITS) to support smart and adaptive learning.

This research project examines challenges faced by both the students and the teacher in an over populated class that inhibit adequate learning and proposes a solution, not entirely a new solution though, by developing an ITS for learning an Object Oriented Programming (OOP) language, case study, Java programming language.

Keywords: Intelligent Tutoring System, Object Oriented Language, Domain module, Student module, communication module, pedagogical module, cognitive tutors, adaptive learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
ITS	Intelligent Tutoring System
OOP	Object Oriented Programming Language
CT	Cognitive Tutors
ACT-R	Adaptive Control of Thought-Rational
CBM	Constraint Based Models
BITS	Binary Digits
IDE	Interface Development Environment
ERD	Entity-Relationship Diagram
JDBC	Java DataBase Connectivity
PC	Personal Computer
Mac Computer	Macintosh Computer
GB	Giga Byte
RAM	Random Access Memory
HDD	Hard Disk Drive
CPU	Central Processing Unit
HP	Hewllet Packard
GHz	Giga Hertz

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

What is an Intelligent Tutoring System (ITS)?

ITS are programs that possess a wide knowledge on certain matter, and their intention is to transmit this knowledge to the students by means of an interactive individualized process, trying to emulate the form in which a tutor or human teacher would guide the student in his learning process (Millán, Agosta & Pérez, 1999).

ITS systems are emerging styles of teaching which will help boost the learning and understanding of a student in relation to a particular subject matter.

Knowledge of programming is a fundamental component of any Computer Science curriculum. It is also applied and used in other fields such as Business, Finance and Accounting as a result of its widespread application in industry. But then, some beginning students find programming a challenging course. This is shown by the fact that many students either drop out or fail programming courses (Miliszewska & Tan, 2007). Hence, it is crucial to find means of assisting students with programming. A large percentage of the population of students show interest in learning programming. They differ in many aspects such as age, gender, educational level and aptitude for solving logical problems. I once lectured Computer Science students at the polytechnic level and I noticed that it is quite challenging to create a single course that meet the needs of all the different students. A way to address this problem is to provide a face to face single human tutor for a student. But then, the finance to do that can be a problem.

That is why we need an Intelligent Tutoring Systems (ITS).

1.2 Problem Statement

Students are always faced with one or more learning problems or challenges within a traditional teaching classroom. Problems such as over population in classes can make students not to be able to always express themselves in class thus, it limits their understanding in class. The teachers on the other hand mostly do not always know the emotional state of each student, the learning style of each student, the progress rate of each student etc. in order to determine the right teaching strategy i.e. pedagogical strategy to use for teaching each student.

It is imperative to understand that not all students have the same learning style or capability. Some student are fast learners while some are slow learners. Some learn better with pictures / diagrams while some learn better with video tutorials. Also, students' knowledge level with respect to a subject matter differ. Knowledge level could be at the **Beginner level, intermediate level** or **advanced level.**

Haven said all that, it is obviously quite difficult for a teacher teaching in an over populated class to be able to fully understand the learning needs of each student and to provide them with the best teaching strategy all at once. Which means that each student will need an adaptive and personalized form of learning so that they can be more knowledgeable with respect to a particular subject.

1.3 Aim and Objectives

1.3.1 Aim

This project aims to develop a solution, which is an Intelligent Tutoring System (ITS) for learning an Object Oriented Programming language (OOP), case study, Java

programming language, so as to help in enhancing students' learning and provide adaptive learning to each student.

1.3.2 Objectives

The main objectives of this research work are:

i. Give a practical and vivid explanation of the main components or architecture of an ITS system and how it works.
ii. To propose and develop an Intelligent Tutoring System (ITS) for learning an Object Oriented Programming Language (OOP), case study, Java programming language, which will hopefully provide an efficient solution or a better approach towards providing personalized and adaptive learning to students without requiring any human intervention.

1.4 SIGNIFICANCE

Research in Intelligent Tutoring Systems has been growing in momentum over the past few decades. Yet, ITSs are not a concept that is known extensively by educators. One of the main reasons for this is that, although many ITSs have been built, only a few are used in practical teaching situations. This indicates that there is significant room for improvement in the field of ITS. This research attempts to improve on existing ITSs at least to a certain degree. Existing ITSs teach in many different domains, from primary school reading to programming and electronic circuit design.

It is essential that any ITS should be able to guide the student through the nitty-gritty he/she needs to know with the respect to the subject being taught.

This research focuses on an ITS system that can teaches its users object oriented programming. A user is distinguished in to three (3) major levels based on his/her current knowledge level of java. The three (3) different levels are:

Beginner: The user at this level will be taught and worked through on the foundation topics of java programming language. Also, the ITS system provides the curriculum that will be followed to teach the user at this level. There is also a platform to assess the user and another platform to keep track of the user / student performance. Tracking progress is also personalized / customized.

Intermediate: The user at this level will be taught and worked through on the topics that are a little above the foundational topics of java programming language. Also, the ITS system provides the curriculum that will be followed to teach the user at this level. There is also a platform to assess the user and another platform to keep track of the user/student performance. Tracking progress is also personalized / customized.

Advanced: The user at this level will be taught and worked through on the topics that are higher than the Intermediate level topics of java programming language. Also, the ITS system provides the curriculum that will be followed to teach the user at this level. There is also a platform to assess the user and another platform to keep track of the user/student performance. Tracking progress is also personalized / customized.

1.5 Limitation of Study / Scope

The scope of this research study is constrained to proposing a developed software for an Intelligent Tutoring System (ITS) for learning an Object Oriented Programming Language (OOP), case study, Java programming language.

The ITS contains a student module which provides personalized interaction for the current student. To achieve this, the system must have very good knowledge about the student, including features such as his/her names, gender, age, capabilities, emotions and other features. The target of this project is not totally geared towards design of the student module only. Hence, the student module consists of various categories that a student needs for learning in line with the subject matter being taught: student profile, a gallery that consists of video tutorials and slides, an assessment platform and a chatting platform. Another use of many ITS is to provide feedback to the student. The feedback in this system is provided using several levels. The feedback would support better learning if the level of feedback provided was personalized/customized based on the current knowledge level of the student.

1.6 Thesis Outline

The entire research is divided into five chapters. Each chapter highlighted different topics and subtopics as follows:

Chapter 2 discusses the basic concepts and literature review related to Intelligent Tutoring System (ITS).

Chapter 3 explains the material and methods used in our research.

Chapter 4 presents the performance results and discussion.

Finally, Chapter 5 contains summary, conclusion, recommendations, and future work.

CHAPTER TWO

LITERATURE REVIEW

This chapter presents literature reviews that have been done in regards to major aspects where Intelligent Tutoring System (ITS) has been applied.

2.1 Introduction

Adaptive learning and personalised learning represents a key topic with regards to Intelligent Tutoring Systems (ITSs). The ability of ITS to adapt their environment to the needs of the students is often recalled as one of the explanations to support the implementation of ITS within learning projects.

It seems vital to emphasize on student's personalized learning. This is the reason why an ITS requires a good profiling procedure, an activity that leads to the composition of a user profile able to collect and elaborate information that are considered important for the recognition of specific needs.

User profiles are built referring to different models that focus on various characteristics of the individuals. The data considered can vary according to the particular research hypothesis, even according to the specific learning outcomes that are associated with the ITS.

The aim of this paper is to outline a detailed overview on the main progresses made in the field of user modelling and user profiling. We will take into account both the traditional approaches and the recent advantages, in order to highlight new lines of research.

2.2 Different types of ITS from different types of personalization procedures

Different types of ITS can be classified thanks to various characteristics. An alternative way to diversify them is to consider the various methodologies that are the basis of the personalization process.

2.2.1 Cognitive Tutors (CT)

The first family of ITS is that of Cognitive Tutors (CT). They derive from an approach that is grounded in a specific theoretical basis: the ACT-R theory of cognition (Adaptive Control of Thought-Rational), developed by John Anderson (Anderson, 1990; Anderson et al., 1994).

The ACT-R theory formulates a representation of human knowledge, dividing it into *declarative knowledge* and *procedural knowledge*. Declarative knowledge is a factual knowledge, involving facts, images or sounds. It is typically acquired through perception and its elements are usually considered as *chunks* of knowledge (Mitrovic et al., 2003). Procedural knowledge is goal-oriented, because it is related to the comprehension of how to do things. It is typically acquired through practice and its elements are usually conceptualized as *production rules*: they specify the plausible conditions of their application and the consequences, in terms of actions, of their application (Ritter et al., 2007).

The learning process arises from two phases. It starts accumulating declarative knowledge, while in a second phase it is converted in declarative knowledge. A complex task involves a set of cognitive skills to be resolved. Such skills can be decomposed and represented as production rules. The ability to solve a given task, therefore, lies in the control of those elements of knowledge.

CT promises personalized support while students are engaged in problem solving activities. The system observes each learner's behaviour to identify the most suitable problems to assign and the most suitable feedback to provide. They can achieve this goal using a cognitive model that represents the description of the skills and strategies, expressed as a set of production rules, required to solve a task in a particular knowledge domain. Creating such a model is a complex challenge, because it must include all the knowledge components that are considered essential within a certain domain, the analysis of human behaviour while solving a particular problem, all the possible paths that we can follow to solve it. (Ritter et al, 2007).

A CT constantly monitors its users, collecting information about their behaviour in a profile. Then, it compares it with the elements stored in the cognitive model in order to assess if a student needs help. This process is called *model tracing*. If an inappropriate action is detected the tutor reports it to the student and gives him/her suitable hints and feedback. Another important process is that of *knowledge tracing*: each action of the student is related to one of the skills provided in the cognitive model (Aleven & Koedinger, 2002). Then, the tutor makes a prediction about the probability that this skill is correctly mastered by the learner. The tutor uses this information to propose activities and exercises that focus the attention on those skills that must be strengthened (Ritter at al., 2007).

2.2.2 Constraint Based Models (CBM)

The second family of ITS is usually defined Constraints Based Models (CBM). They are projected to overcome some limitations of CT.

To design cognitive models, we need a long and very complex process. Referring to some particular domains, such as Human Sciences, the process can result impossible because problems can have multiple solutions and, furthermore, they can be solved following a wide variety of possible paths. Even referring to those domains, such as Mathematical Sciences, in which the process is possible, CT present a second type of problems. The main purpose of the tutor is to understand when and why a certain student makes an error. The tutor can understand it only if it is able to replicate that action, even if it is incorrect. As a consequence, it becomes essential to incorporate within its cognitive model also the so-called *buggy rules*, that are all those choices that lead to a wrong result (Mitrovic, 2012).

CBM represent a new methodology for building models, which are no longer based on production rules, but on the constraints (Ohlsson, 1992). This approach derives from a theoretical basis: the theory of learning from performance errors (Ohlsson, 1996). According to this theory, knowledge is divided into declarative knowledge and procedural knowledge, in the same way we discussed above referring to ACT-R theory. While procedural knowledge is crucial to generate actions, declarative knowledge is crucial to evaluate the consequences of our actions. Errors derive from missing or faulty procedural knowledge. Learning from errors follows two phases: *error detection* and *error correction* (Mitrovic, 2012). The ability to detect an error depends on declarative knowledge. If it presents some gaps, a certain person won't be able to detect an error individually. This means that he/she needs some help. This support can come from an intelligent tutor that needs a model no longer based on production rules, but on constraints.

Constraints are the basic principles of a domain knowledge. Once established these principles, it becomes possible to determine the correctness or incorrectness of any solution given by students. To be defined correct a solution must respect those principles. If it violates even one of them, it is incorrect (Mitrovic et al., 2003). This reasoning can be applied to any answers elaborated, regardless of the strategy used to achieve it. The solution path is not considered important even for the generation of feedback, because it is directly linked to the violated constraint.

This new type of ITS leaves much space for creativity (Mitrovic, 2012). Every solution and every strategy can be defined correct if they respect the domain constraints, even if they were not inserted into the model by designers.

2.2.3 Curriculum sequencing

The third family of ITS follows a different approach, usually remembered as curriculum sequencing. It aims to generate personalized learning paths adapting, in a dynamic way, the didactic content of the course according to the student's objectives, to his/her previous knowledge and to his/her success in acquiring new knowledge (Brusilovsky, 2001).

The system monitors users' behaviour in order to select the most appropriate *teaching operations* for each individual, such as different sets of materials, examples, questions or problems, in order to support the achievement of personal learning goals (Brusilovsky & Vassileva, 2003). All teaching operations are stored in a database. To select the most suitable for a certain person the system requires at least two models: a model for representing a specific domain knowledge and a model for representing the learner's current state of knowledge (Heller et al., 2006).

A theoretical support for this purpose could come from the Knowledge Space theory (Falmagne et al., 1990). It assumes that a domain knowledge can be represented as a network of concepts, that are thought as questions or problems. The knowledge state of a learner in that domain can be considered as a subset composed by all the questions that he/she is able to solve individually. The problems within a domain are linked by mutual dependencies. For this reason, not all the possible knowledge states can be plausible. The group of plausible knowledge states, referring to a certain domain, is called a knowledge space. We can also assert that a knowledge space determines the structure of prerequisites among concepts within a domain (Desmarais et al., 2006).

Comparing the three types of ITS that we are proposing, we can highlight that CT and CBM share some purposes. This is the reason why Brusilovsky unifies them into one category: *problem solving and solution analysis tutors* (Brusilovsky & Peylo, 2003). CT and CBM share the main aim to provide students with just-in-time, specific and effective feedback while they are engaged in problem solving activities. The main aim in curriculum sequencing is to elaborate an assessment of a larger set of skills in order to adapt learning content in general. Due to the number of skills to take into consideration the tutor needs a model able to establish network of skills, to make the process sustainable Desmarais & Baker, 2012).

All the traditional approaches presented are based on the idea that student's profiles must deal with skills and levels of knowledge. More recent researches assume that student profiling can take into account a larger range of elements, as we will discuss in the next session.

2.3 New challenges for personalization

Over the past 15 years, the topic of personalization has received renewed interest from the scientific community that deals with ITS. Starting from the consideration that the learning process is affected by a whole series of factors, the fundamental question becomes whether and how these factors play a crucial role even when the process is computer-mediated.

2.3.1 Emotions, motivation and disengagement

Emotion, motivation and disengagement are three correlated aspects that play an important action in affecting the learning process. It is extremely important to understand how ITS can detect them and how they can provide consequent feedback. This is a very complex task, because when learning process is computer-mediated we don't have an easy access to a vast amount of information, such us facial cues, postures or gestures, that can help us to draw a picture of a learner's current state.

Conati and Maclaren develop a method to reveal a vast range of students' emotions while they are interacting with an educational game (Conati & Maclaren, 2009). To record these emotions, they use four non-intrusive biometric sensors to measure skin-conductivity, electromyography of some facial muscles, blood-volume pressure and respiration.

Mota and Picard monitor children while they are solving a task using a computer to recognize different affective states related to different levels of interest displayed (Mota & Picard, 2003). They analyse different postures that are gathered using two matrices of pressure sensors mounted on the seat and back of a chair.

Chaouachi and Frasson measure the electrical activity of the human brain (Chaouachi & Frasson, 2010). Using 8 biosensors and two video cameras, they demonstrate that different emotions affect students' performance, engagement and response time.

Forbes and Litman work with an ITS with natural language dialogues to create an automatic method to predict students' emotions (Forbes & Litman, 2004; 2006). They use as reference points acoustic and prosodic features.

D'Mello and colleagues investigate the transitions between affective states (D'Mello et al., 2007; 2008). They use videos of participants' faces and recorded interaction histories to map possible paths to identify four specific emotions: boredom, flow, confusion, and frustration.

Some of the researches presented share a possible limitation: the use of sensors. Arroyo and his team create a relatively cheaper and more comfortable suite of sensors to use at school (Arroyo et al., 2009). Conati and colleagues highlight that children who wear them do not perceive the condition as intrusive, but the issue of their possible application to a wide population remain opened (Conati & Maclaren, 2009).

The emotional state of a student is often related to his/her motivational state. De Vicente and Pain conducted an empirical study to analyse some diagnostic procedures to infer it (De Vicente & Pain, 2002). Analysing the students' interactions with an ITS for learning Japanese numbers, researches create various rules to diagnose a motivational state that are later incorporated in a model within an ITS. These rules are often related to the quality of students' performance, such as duration of the interaction, mouse movements or the order of exercises followed by an individual.

Rebolledo and colleagues add a different motivational model into an existing ITS, that is able to provide consequent scaffolding (Rebolledo et al., 2006). This model is based on three main elements: effort, independence and confidence. Some years later Rebolledo tries to create a new model where motivation is also linked to the student's level of attention (Rebolledo et al., 2010).

One of the problem in ITS is students' disengaged behaviour, often associated with negative learning outcomes. It is important for an intelligent tutor to understand what type of behaviours could imply this particular situation. Baker and colleagues study a particular attitude that can be considered interesting in this direction: gaming the system (Baker et al., 2006). They describe it as: «attempting to succeed in an educational environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly» (Baker et al., 2006, p. 101). As they demonstrate that this attitude can negatively affect learning, they develop a method to detect this strategy and to generate the most suitable remedial. Arroyo and colleagues assert that gaming the system can also derive from a poor usage of meta-cognitive resources (Arroyo et al., 2007). For this reason, they try to create a different type of support, including the production of meta-cognitive feedback.

2.4 ARCHITECTURE OF A TYPICAL ITS SYSTEM

A typical ITS, has the following four basic components.

2.4.1. The Domain model:

The domain model (also known as the cognitive model/expert knowledge model) consists of the concepts, facts, rules, and problem-solving strategies of the domain in context. It

serves as a source of expert knowledge, a standard for evaluation of the student's performance and diagnosis of errors.

2.4.2. The Student model:

The student model is an overlay on the domain model. It emphasizes cognitive and affective states of the student in relation to their evolution as the learning process advances. As the student works step-by-step through their problem solving process, the system engages itself in model tracing process. Anytime there is any deviation from the predefined model, the system flags it as an error.

2.4.3. The Tutoring model:

The tutor model (also called teaching strategy or pedagogic module) accepts information from the domain and student models and devices tutoring strategies with actions. This model regulates instructional interactions with student. It is closely linked to the student model, makes use of knowledge about the student and its own tutorial goal structure, to devise the pedagogic activity to be presented. It tracks the learner's progress, builds a profile of strengths and weaknesses relative to the production rules (termed as "knowledge-tracing").

2.4.4. User Interface model:

This is the interacting front-end of the ITS. It integrates all types of information needed to interact with learner, through graphics, text, multi-media, key-board, mouse-driven menus, etc. Prime factors for user-acceptance are user-friendliness and presentation. Figure 1 presents a typical ITS architecture.

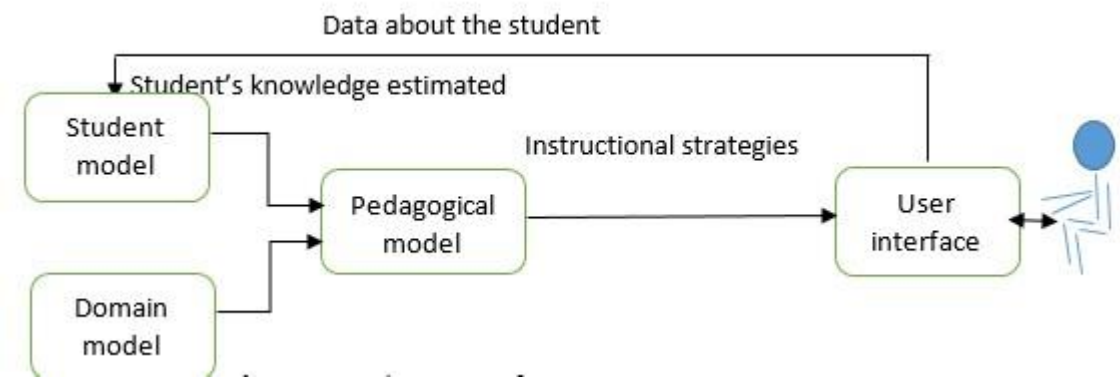


Fig.1 Architecture of an Intelligent Tutoring System

2.5 BAYESIAN APPROACH FOR ITS SYSTEMS

Andes (Conati et al, 2002; Gertner & VanLehn, 2000) is an ITS which was developed to teach physics for the students in Naval Academy. Bayesian networks were primarily used in Andes for decision making.

The major foci of the system are

1. Select the most suitable strategy for the student
2. Predict Student's actions
3. Perform a long term assessment of the student's domain knowledge.

Andes is a domain dependent ITS. Each problem in the system was broken into some steps and Bayesian network was formed using those steps as nodes. So, the problems were represented in the system as Bayesian networks. The Bayesian network would predict the most probable path for the student during a course. Each student could have different approaches to a problem, the network would be adjusted accordingly (the probabilities would change) and finally for a new problem it would predict the best strategy for the student.

ViSMod is another ITS which used Bayesian network (Zapata-Rivera et al, 2004). In the system the Bayesian network was divided into three levels. At the top most level the concepts (to be taught) were represented in a hierarchical manner. After that in the second level student's performance and behavior were described. Finally, the third level nodes represented some analysis on the student's performance. Only the first level is domain dependent, whereas other two levels would remain same over different domains. Again student can observe only the top two levels of the Bayesian net. The third level is only visible to the teachers. During a course the probabilities in the second and third level of the Bayesian network changed according to the student's performance.

BITS, a web based Intelligent tutoring system for Computer Programming, uses Bayesian Networks for making the decisions. Using the Bayesian Network, the prerequisite relationships among the concepts are represented directly and clearly. In BITS, there are two methods of obtaining the evidence required to update the Bayesian Network:

A student's direct reply to a BITS query if this student knows a particular concept.

A sample quiz result for the corresponding concept to determine whether the student has understood a particular concept or not.

2.5.1 Case Study on Data Structures- Graphs

Graph is denoted by {V,E} where V is the set of vertices and E is the set of edges connecting the vertices. The topics related to Graph can be represented by a Knowledge Dependency Graph in the Knowledge Base Model as:

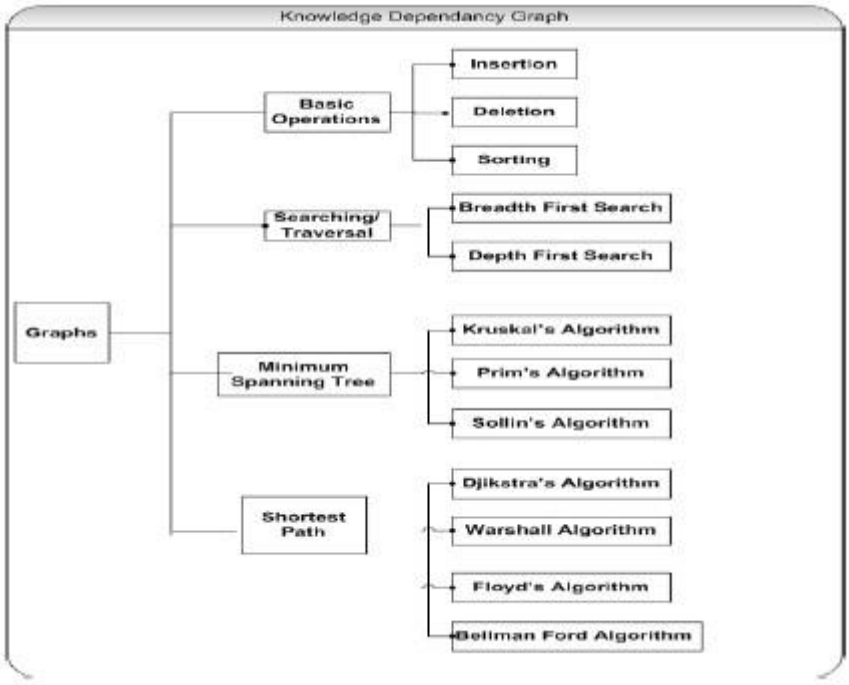


Fig. 2 Knowledge Dependency Graph

As seen in Figure 2, the prerequisite for studying the various concepts can be stored in the Knowledge Base model to facilitate easy understanding to the learners.

CHAPTER THREE

METHODS AND DESIGN

3.1 INTRODUCTION

In this chapter, we describe the methods used in carrying out the study and accumulating data for analysis and design. Before the development of an efficient Intelligent Tutoring System, hereinafter termed ITS, it is important to undertake a careful and analytical study also known as System analysis. The reason is to identify the inconsistency prone areas of ITS operations as well as the flaws affecting accuracy, reliability, data integrity, speed of operation, and overall efficiency of the system. This chapter encapsulates the following: Research design, Research intervention, Research instrument, Research Processes, Data Analysis and Interpretation. It also includes System architecture, Program design, File organization, System models, Programming Language, Interaction models, Database used, System control and System requirements.

3.2 Methodology

The method adopted here for program design is the structured software development approach using Object-Oriented Analysis, Design and Implementation. This is carefully done to ensure that the software can be read, understood and built upon in the future.

As it has been stated in Chapter 1, the problem that this research work tends to address is to enhance learning in a more efficient way. Traditionally, classrooms can be over-populated with students. In that kind of a condition, learning may not be efficiently administered thus, limiting understanding of most students.

3.3 Research Design

In this research work, we will make use of both the traditional waterfall approach to software development and incremental build model as the work is designed in a sequence from start to finish. (Ukem and Ofoegbu, 2012) referred to the Waterfall software life-cycle model as a traditional model because it was the first widely used software development life cycle. They stated that it is part of Structured Software Engineering or the Structured Paradigm, which is the older of the two earliest approaches to formal software engineering.

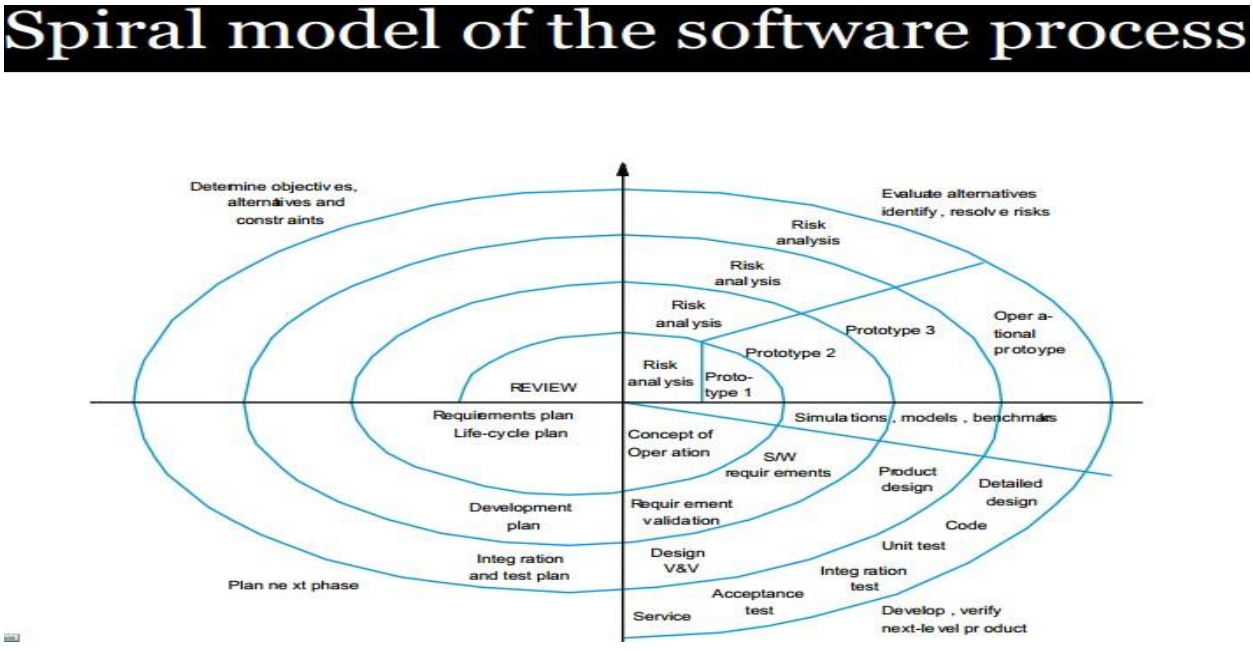


Fig. 3 Spiral model software development.

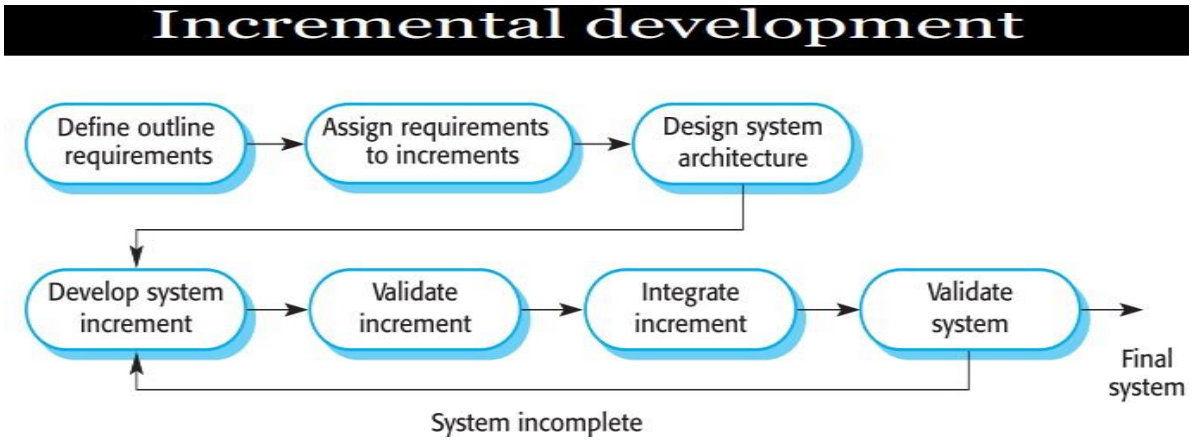


Fig. 4 incremental model for software development

Also, at every stage of the development, improvements were made based upon the tests that was carried out and further suggestions that was made by my supervisor – Prof. Hamada on the application being developed, until the required outcome was obtained. This research is about an ITS which was developed using the Netbeans IDE (Interface Development Environment). The goal of ITS is to provide customized and personalized efficient learning among students or learners without requiring human intervention.

I made use of two levels of system design models. The first model is the logical model and the second model is the physical model. The logical model is about the database tables, data within these tables and the relationship between the tables that make up the application. They logical model identifies entities (i.e. data), and what these entities are, can be shown in a normalized and fully attributed entity-relationship diagram (ERD) representing the data model. An entity-relationship diagram (ERD) is a data modeling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system, (Searchcrm, 2014). The entities identified in this project design are: domain, students, tutor, student's registration and user

interface. We will be going through these in details in chapter four of this thesis. The second model, physical model, defines how the system is physically and technically implemented. The output of the physical modeling is the database model. The database model contains the database table which holds the data as described in the logical model, it also contains the physical data files in which the database tables together with their data will be stored. It is recommended that both the application server and database server be on separate computers, but because of the cost implication we will run both applications on the same computer, which is also fine. Students / users will be authenticated at the login page before they can interact with the application and database, therefore usernames and passwords will be required from users at the point of login, once successfully authenticated by the system at login, users are re-directed to the appropriate page.

3.4 Research intervention/ Proposed System

The ITS system used in this research was developed using the Netbeans IDE (Interface Development Environment) for learning an Object Oriented Programming Language (OOP), case study, Java programming language, which will hopefully provide an efficient solution or a better approach towards providing personalized and adaptive learning to students without requiring any human intervention.

It has the following benefits:

1. **Improvement in academic performance:**

   Learners / students who oblige in learning with an ITS system showed a boost in student's marks. Research has shown that students score high percentage in their

final exam in computer programming when using ITS than their counterparts in a conventional learning environment where human tutor is available (Ong & Ramachandran, 2000). Also, Ross and Casey (1994) discovered that students involved in ITS programs developed optimal "problem solving capabilities.

2. **Applicable to other fields and sectors:**

ITS are applied in a various of learning environments. Security agencies have adopted ITS systems to provide operational skills and tactical training. ITS have also been successfully applied in aviation sector for in flight simulations, in fire services for fire training, in secondary schools for mathematics and physics courses and in the health care profession for nurses' training. ITS are also being used in basic skills training to teach employees about selling, negotiating and working collaboratively (Ong & Ramachandran, 2000; Leddo & Kolodziej, 1997).

3. **Efficiency in cost:**

ITS is practically less expensive than funding in conventional classroom where each human-tutor has to be remunerated for his / her services. The only cost incurred in ITS is that of hardware or software purchase. Due to ITS cost efficiency, it has been widely used in various occupations and classrooms. (edited by Steffanie Reid, 2008).

4. **Speed in learning:**

In contrast to their traditional or conventional learning environment, ITS helps to enhance student’s cognition skills and also assists them work through their assignments at a faster pace.

5. **Tutoring advantage:**

   Providing a human educational tutor for each student in a traditional classroom is almost not feasible due to the financial constraint attached to it. ITS provide the same learning experience just like the ones provided by a human tutor but at a much lower cost. (edited by Steffanie Reid, 2008).

The diagram bellow is a flowchart representation of an Intelligent Tutoring System.

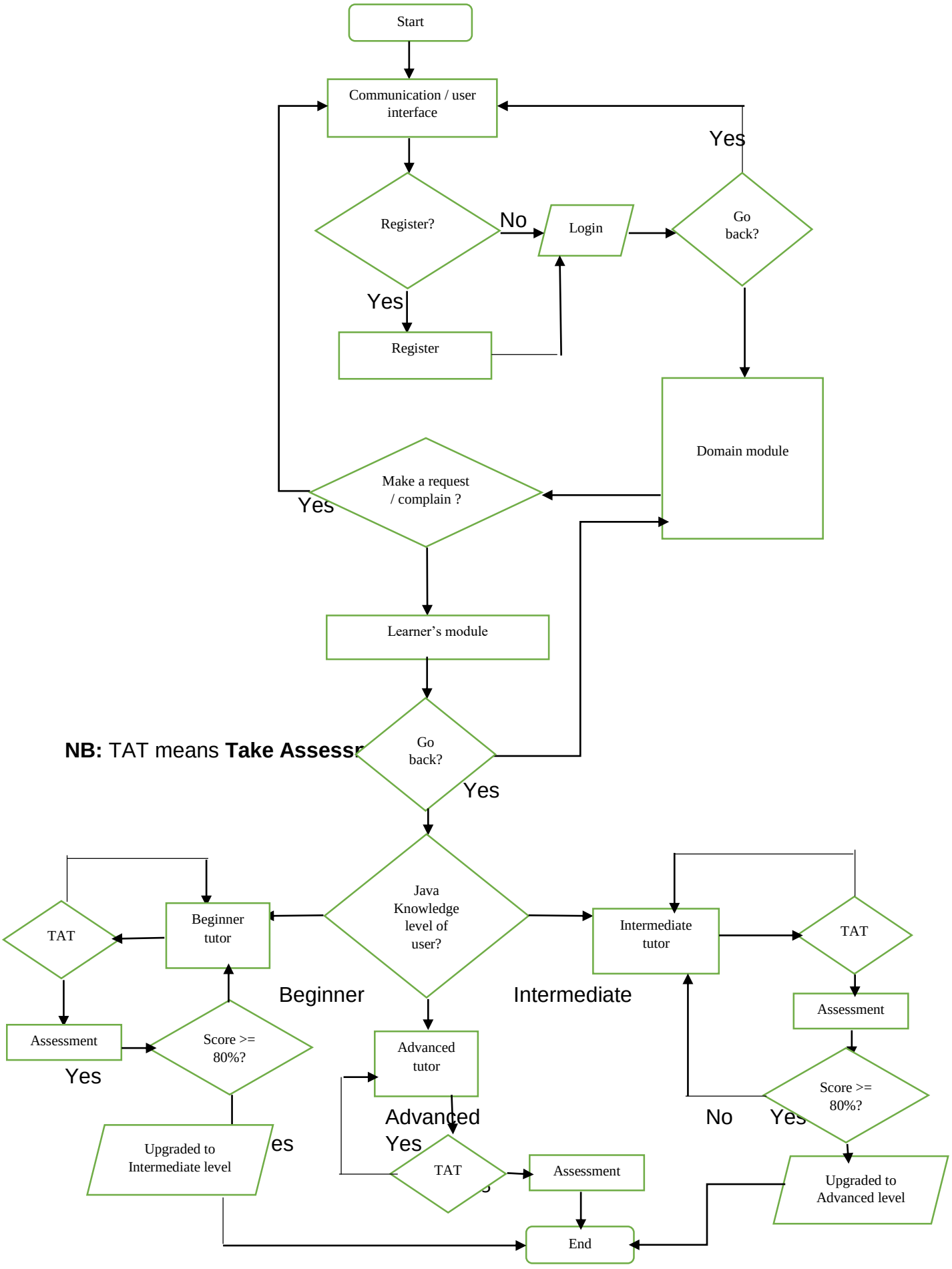

Start
Communication / user interface
Register?
No
Login
Go back?
Yes
Yes
Register
Domain module
Make a request / complain ?
Yes
Learner's module
Go back?
Yes
NB: TAT means Take Assessr
Java Knowledge level of user?
Beginner
Intermediate
Beginner tutor
TAT
Assessment
Score >= 80%?
Yes
Upgraded to Intermediate level
es
Intermediate tutor
TAT
Assessment
Score >= 80%?
No
Yes
Upgraded to Advanced level
Advanced tutor
Advanced
Yes
TAT
Assessment
End

Fig. 5 Intelligent Tutoring System Flow chart

3.5 Requirements

3.5.1 Software Requirement

The framework is implemented in Java programming language. It comes along with needed libraries such JDBC (Java DataBase Connectivity). This language was chosen over other because of it is user-friendly, and flexibility yet given good output during the preprocessing stages.

3.5.2 Hardware Requirement

The minimum hardware requirements include the following:

1. Windows 8 or 10, 64 bits (PC or Mac computers).
2. All CPU (Intel family or Xeon).
3. 4 GB RAM or above, 20GB HDD Free Space

CHAPTER FOUR

IMPLEMENTATION AND RESULTS

This chapter discusses about the results and discussions. Screenshot is used in the presentation of our findings. This chapter includes implementation which comprises of the programming language used, text input, preprocessing, matching, and evaluation of the performance.

4.1 Presentation of Results

The solution was implemented in the Java programming language (Netbeans IDE 8.2) and run on a Hewllet Packard (HP 630) laptop @2.00GHz with 6GB RAM on Windows 10. Following on the development methodology, this chapter describes design implementation in terms of Program coding, Program user interface, Database design and code logic for the entire system.

4.2 Implementation of the system Application

The development of the Senate format system involves the following process:

4.2.1 Coding of the Program

The coding of the program was done using Object-oriented, modular approach to programming. Part of the program codes are shown in **Appendix I.**

4.2.2 Processor Module Object Classes

All the Processor Objects have their source from the one class definition. The code snippet below shows the main class construction, moreover, any added attribute or method depends on the type of object being created.

4.2.3 Program Interface Design Screen Shots

1. **COMMUNICATION / USER INTERFACE:** This interface will be responsible for introducing and welcoming the user to the ITS platform. Via this same interface, the user can find the link to the registration interface, login interface or can make a request or lay a complain.

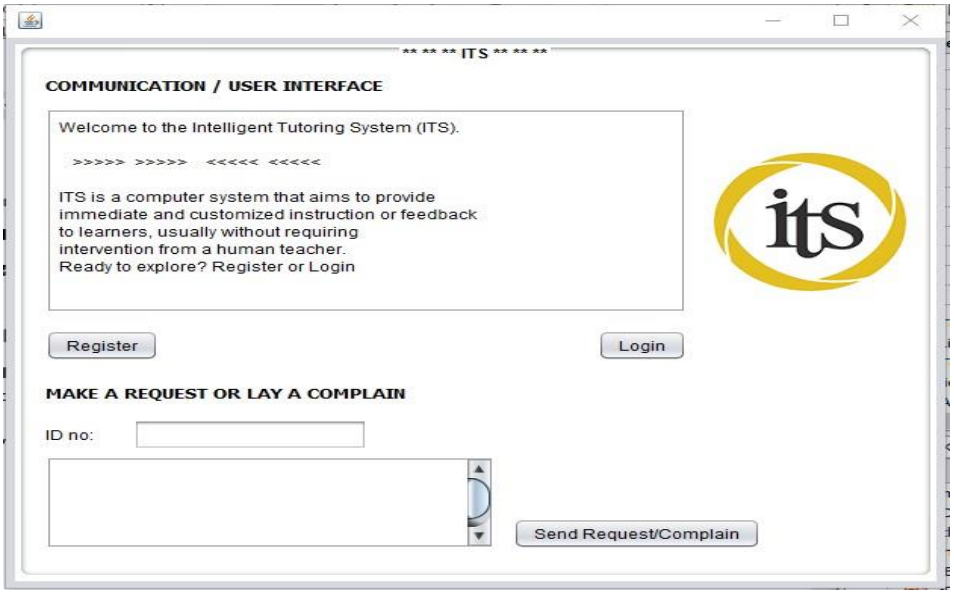


Fig. 6 Communication / user interface

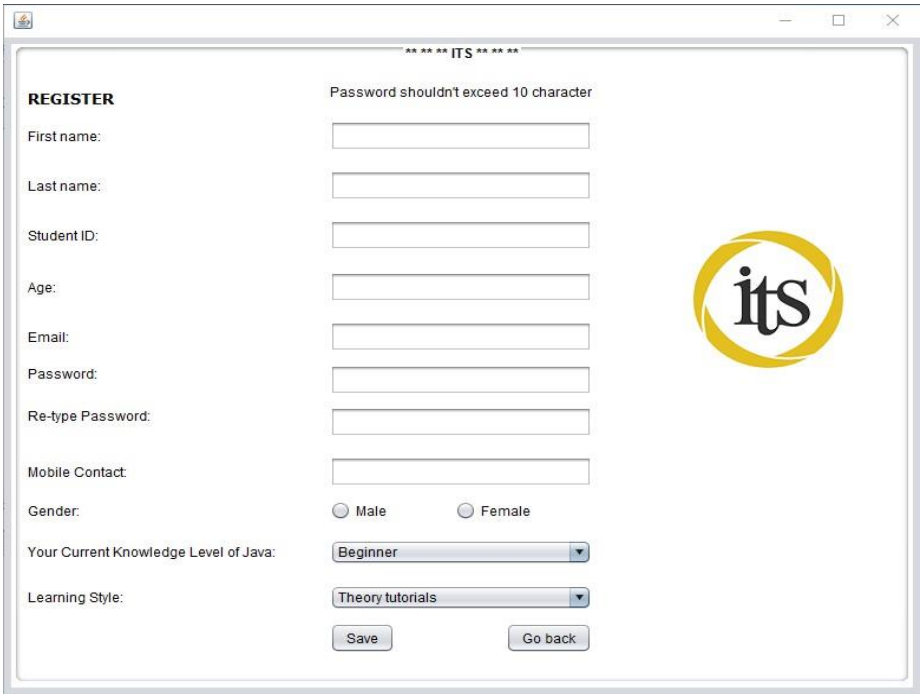


Fig. 7 Learner's registration interface

NB: The database I used is WampServer 64 bits version 3.17 which comes with MySQL and phpMyAdmin.

Fig. 8 Login Interface

2. **LEARNER'S / STUDENT MODULE:** This interface is responsible for creating the student profile which will show details about the registered user such as names, email, gender, knowledge-level, etc.

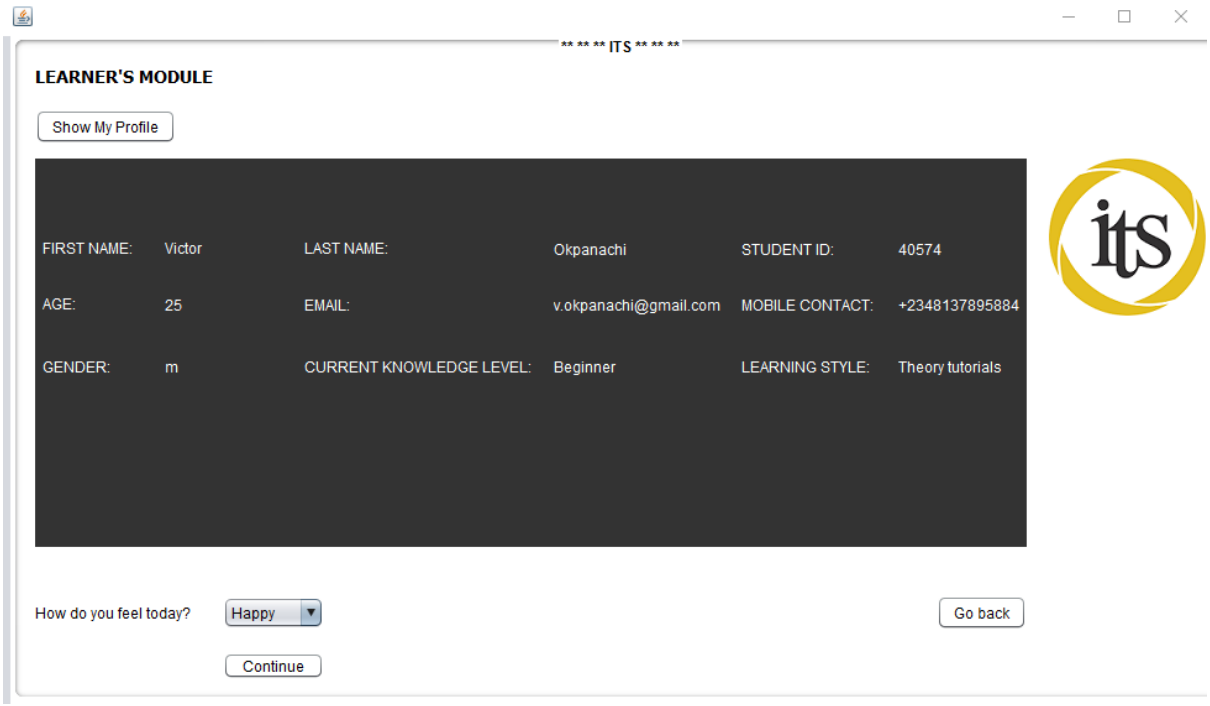


Fig. 9 Learner's module

3. **DOMAIN MODULE:** This module is responsible for telling its user what the ITS system intends to do. Also, this module gives a brief description of the three (3) pedagogical/teaching strategies that will be used in this system. They are:
    a. Beginner's pedagogical interface
    b. Intermediate pedagogical interface
    c. Advanced pedagogical interface

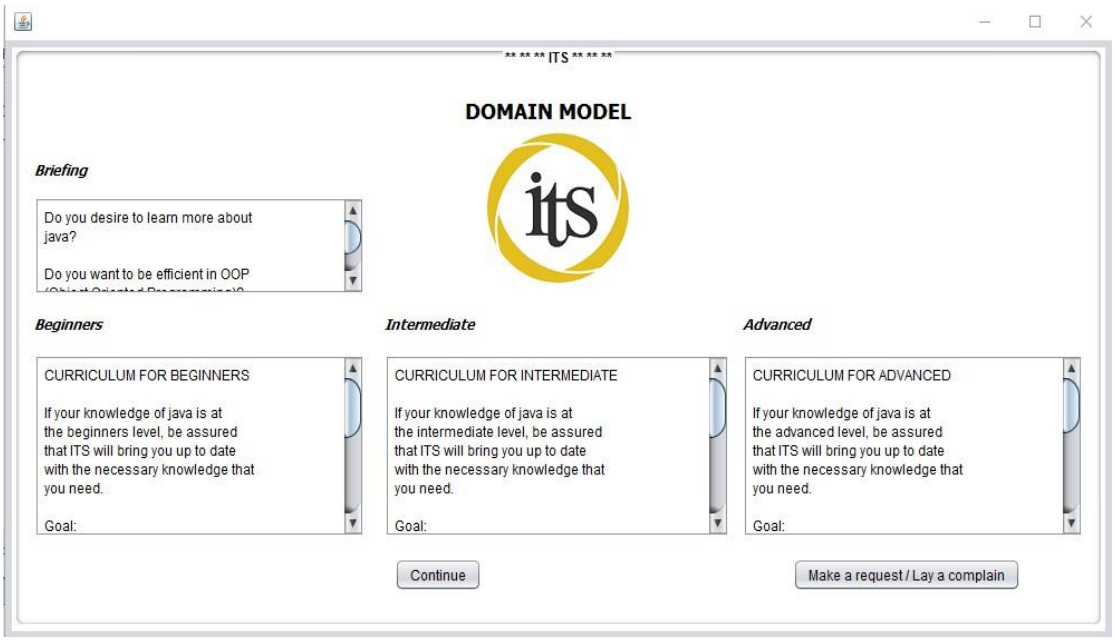


Fig. 10 Domain module

4. **PEDAGOGICAL / TUTOR / TEEACHING MODULE:** The pedagogical interfaces will be administered to learner's depending on their current knowledge level of java. Current Knowledge levels is divided into the three (3) pedagogical strategies mentioned above.

Fig. 11 Beginners pedagogical interface

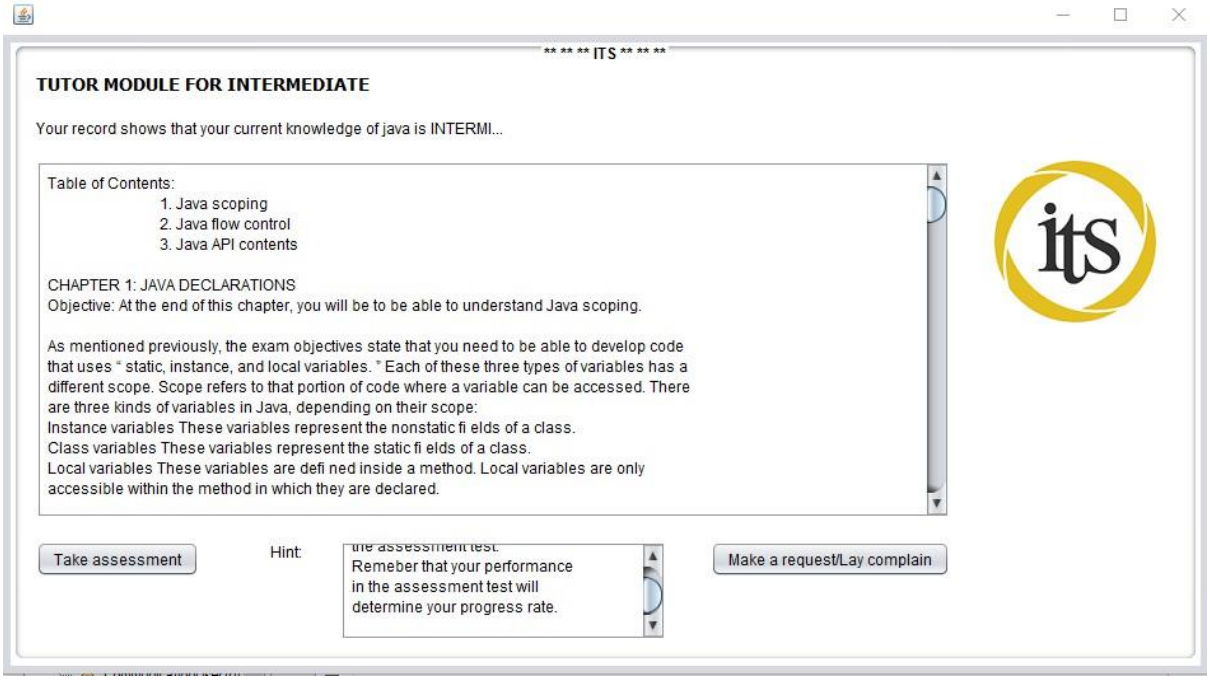


Fig. 12 Intermediate pedagogical interface

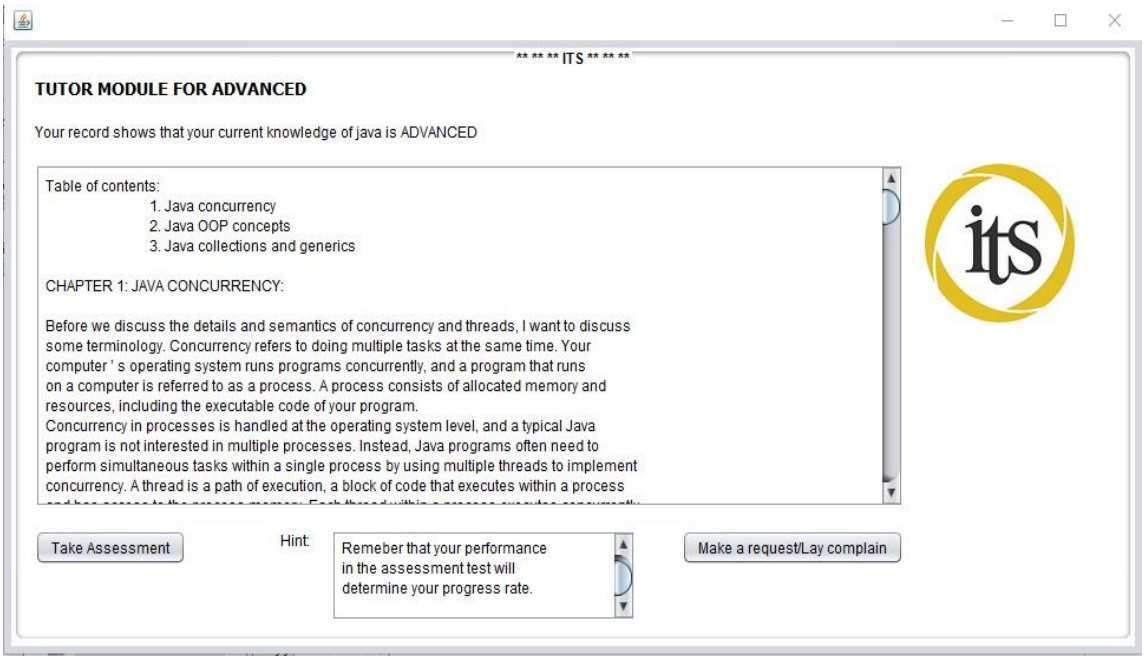


Fig. 13 Advance pedagogical interface

5. **ADMIN MODULE:** The Admin module has the ability of viewing enrolled students, communicate / chat with students and send feedback / provide advice in time to student.

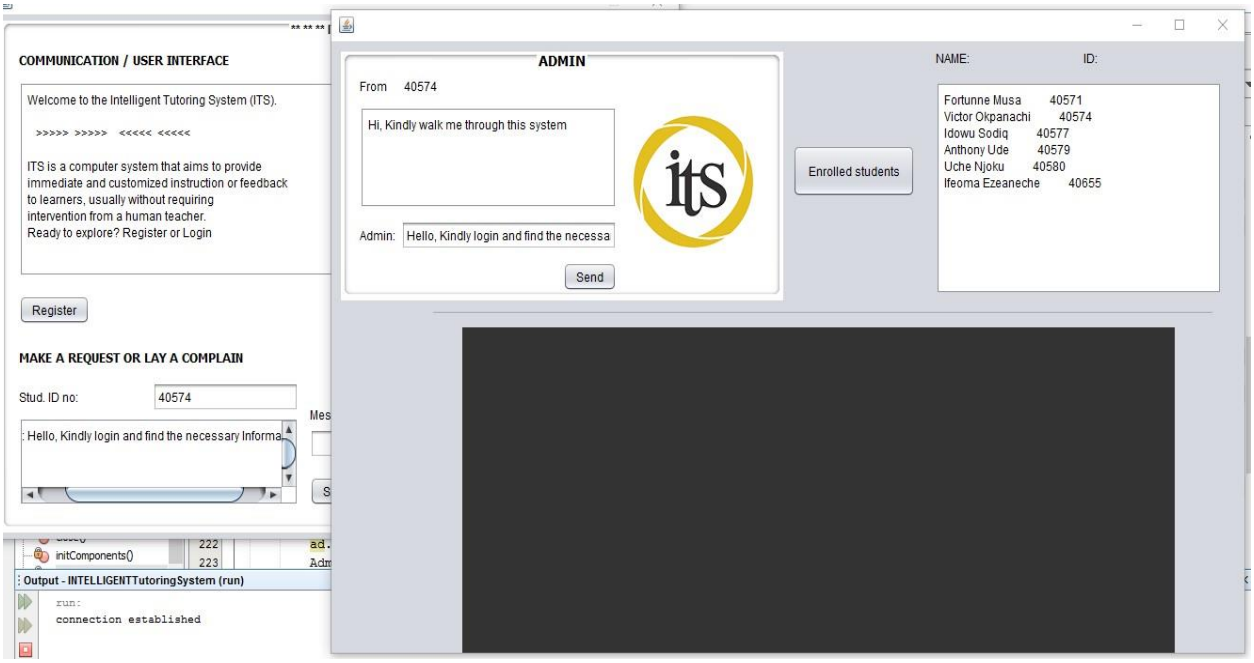


Fig. 14 Admin Module

4.3.1 Class: Domain module:

```
package intelligenttutoringsystem;

import java.awt.Toolkit;
import java.awt.event.WindowEvent;
/**
 *
 * @author Victor
 */
public class DomainModule extends javax.swing.JFrame {
    public DomainModule() {
        initComponents();
    }
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        close();
        CommunicationUserInterface cui = new CommunicationUserInterface();
```

```
        cui.setVisible(true);
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        close();
        LearnerInterface li = new LearnerInterface();
        li.setVisible(true);
    }
public static void main(String args[]) {
java.awt.EventQueue.invokeLater(() -> {
            new DomainModule().setVisible(true);
        });
    }
public void close(){
        WindowEvent winClosingEvent = new WindowEvent(this,
WindowEvent.WINDOW_CLOSING);
        Toolkit.getDefaultToolkit().getSystemEventQueue().postEvent(winClosingEvent);
    }
}
```

4.3.2 Class: Admin module:

```
package intelligenttutoringsystem;

import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author Victor
 */
public class Admin extends javax.swing.JFrame {
    public Admin() {
        initComponents();
```

```
    }
private void jBAdminSendMsgActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        String adMsg = "Admin: " + jTAdminTypeMsg.getText();
        CommunicationUserInterface.jTxtACommSeeMsg.setText(adMsg);
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jLNames.setText("NAME:");
        jLIds.setText("ID:");
        try{

                    Class.forName("com.mysql.jdbc.Driver");
      //connect to DataBase with your username and password
                            System.out.println("connection established");
           //return conn;
                String sql = "select firstName, lastName, studentID from
studentregistertable";
                Statement st = conn.createStatement();
                ResultSet rs = st.executeQuery(sql);


                while(rs.next()){
                 String stud = rs.getString("firstName") +" "+ rs.getString("lastName") +"
"+ rs.getString("studentID");
                 enrolledStud.append(stud +"\n");


                }
                st.close();

              }catch(SQLException | HeadlessException e){
                JOptionPane.showMessageDialog(null, e);
              } catch (ClassNotFoundException ex) {
            Logger.getLogger(Admin.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
    public static void main(String args[]) {
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Admin().setVisible(true);
            }
        });
    }
```

Please refer to **Appendix I** for some other codes that make up the complete **Intelligent Tutoring System** program.

CHAPTER FIVE

SUMMARY, CONCLUSION, RECOMMENDATIONS, AND FUTURE WORK

This chapter discusses the complete thesis report, which highlights the importance of our findings in summary, conclusion, recommendations, and the future work.

5.1 Summary and Conclusion

A computer system that aims to provide immediate and customized instruction or feedback to learners, usually without requiring intervention from a human teacher is known as an **Intelligent Tutoring System.**

ITS have four (4) main components as stated below:

I. Communication module
II. Domain module
III. Learner's module
IV. Pedagogical module

It is imperative to mention that an adaptive ITS needs to select the best instructional strategies at the right time, depending on the learner's profile in certain conditions and the learning process in general. Selections of teaching strategies should be done in order to maximize in-depth learning and motivation so as to reduce costs and training time.

Table 5.1 Difference between domain expertise and domain pedagogy

Domain expertise	Domain pedagogy
It addresses issues of tasks the expert is supposed to do.	It addresses issues of teaching strategies.
What characteristics does the expert look out for?	What actions should the ITS look out for?
What does the expert do to influence transition?	What measures should be used to enhance learning?
What tasks and outcomes measure performance?	What are the learning objectives?

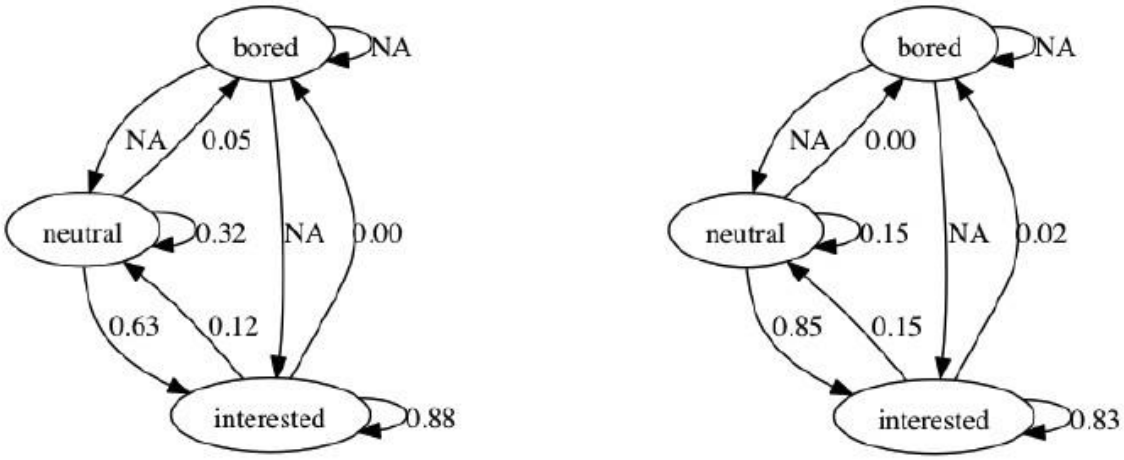
Importance of intelligent tutoring system

1. ITS provides personalized or customized one-to-one learning.
2. ITS is handy in environment where expert human-tutor is/are unavailable.
3. ITS provides best/optimal teaching strategies to learner.
4. ITS accommodates all levels of participants (both young and old).
5. ITS is always adaptive to learner's current state.

Emotional Intelligence in ITS

Studies have shown that learners who are bored, angry, moody or destabilized will not be in the right state of mind to learn. Thus the need for emotional intelligence in ITS which will help manage and determine the emotional state of the learner before determining the right teaching tactic to administer to the learner.

Fig. 15 **Markov chain analysis, showing learner's transition from one emotional state to another**



D1. **D2.**

D1 is a diagram, which analyse students who have high access to an ITS (Student performance page). According to Markov, learner's in this category tend to experience higher enjoyment and interest to learn. Thus, there is a zero probability of getting bored when a student is interested in a topic.

D2 on the other hand, shows students who have less access to an ITS (Student performance page). According to Markov, there is a level of probability that students in this group lose interest in a topic, thereby transiting into the bored-emotional state.

5.2 Recommendations

i. Understanding learner's cognitive ability is very important in any ITS system. Thus, more instructional mediums such as images, videos, Illustrations, etc. can be added to the ITS system.

ii. A chatting system can be added to the ITS system where learners can chat with other learners, share and discuss their learning tasks together. Hence, they will be

an admin who will monitor the chats of the learners and the learners can further relay their queries to the admin.

iii. Providing adaptive learning to users is key in an ITS system. I suggest incorporating ways of determining user's emotional status into the ITS system.

5.3 Future Work

As a future work, this research study can further be improved upon by the addition of facial and body **emotional analysis** to the ITS System.

It involves the following steps as can be seen in the diagram bellow:

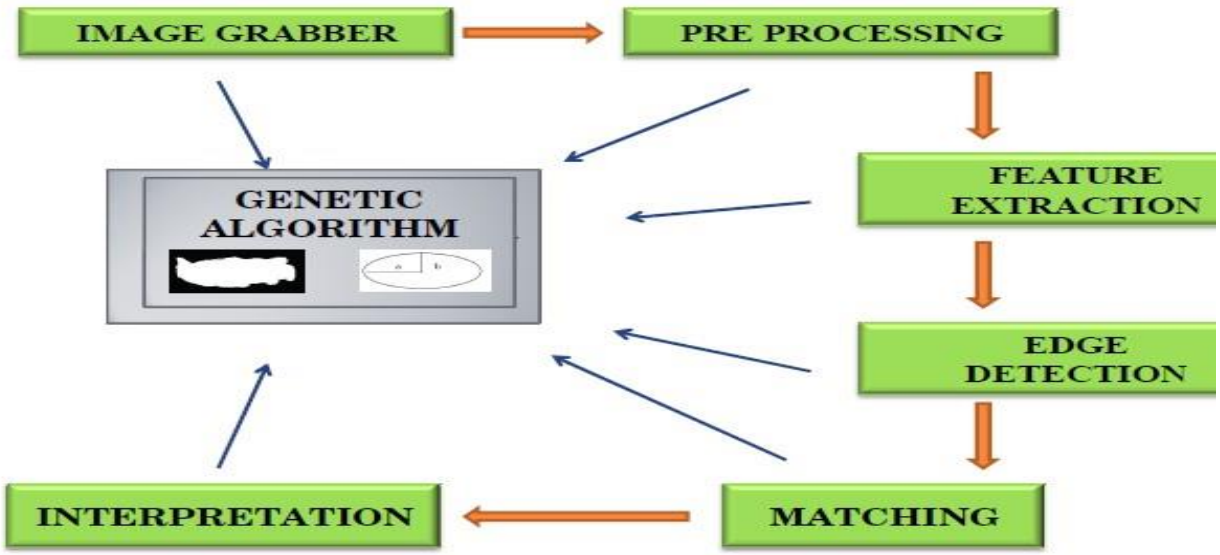


Fig. 16: ITS Emotional and Body analysis System.

REFERENCES

Aleven, V. & Koedinger, K. R. (2000). Limitations of Student Control: Do Students Know when they need help? In *Intelligent Tutoring Systems: 5th International Conference, ITS 2000,* Montreal, Canada, June 19th-23rd, 2000, pp. 292-303.

Aleven, V., McLaren, B., Roll, I., Koedinger, K. R. (2006). Toward Meta-cognitive Tutoring: A Model of Help-Seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education 16* (2), pp. 101-130.

Aleven, V. & Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science 26*, pp. 147-179.

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Lebière, C., Qin, Y. (2004). An integrated theory of the mind. *Psycological review 111*, pp. 1036-1060.

Arroyo, I., Ferguson K., Johns J., Dragon T., Meheranian H., Fisher D., Barto, A., Mahadevan, S., Woolf, B. P. (2007). Repairing Disengagement With Non-Invasive Interventions. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007*, Los Angeles, Usa, July 9th-13th, 2007, pp. 195-202.

Arroyo, I., Cooper, D. G., Burleson, W., Woolf, B. P., Muldner, K., Christopherson, R. (2009). Emotion Sensors Go To School. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education, AIED 2009*, Brighton, UK, July 6th-10th, 2009, pp. 17-24.

Baker, R. S. J., Corbett, C., Koedinger K. R., Evenson, S., Roll I., Wagner A. Z., Naim, M., Raspat, J., Baker, D. J., Beck J. E. (2006). Adapting to When Students Game an Intelligent Tutoring System. In *Intelligent Tutoring Systems: 8th International Conference, ITS 2006*, Jhongli, Taiwan, June 26th-30th, 2006, pp. 392-401.

Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction 11* (1-2), pp. 87-110.

Brusilovsky, P. & Peylo, C. (2003). Adaptive and Intelligent Web-based Educational Systems. *International Journal of Artificial Intelligence in Education 13* (2), pp. 156-169.

Brusilovsky, P. & Vassileva, J., (2003). Course sequencing techniques for large-scale web-based education. *Engineering and Lifelong Learning 13* (1-2), pp. 75-94.

Chaouachi, M. & Frasson, C. (2010). Exploring the Relationship between Learner EEG Mental Engagement and Affect. In *Intelligent Tutoring Systems: 10th International Conference, ITS 2010*, Pittsburgh, USA, June 14th-18th, 2010, Part II, pp. 291-293.

Conati, C. & Maclaren, H. (2009). Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction 19* (3), pp. 267-303.

Biswas, G., Jeong H., Kinnebrew, J. S., Sulcer. B., Roscoe, R. (2010). Measuring Self-Regulated Learning Skills through Social Interactions in a Teacher Agent Environment. *Research and Practice in Technology-Enhanced Learning 5* (2), pp. 123-152.

Desmarais, M. C., Meshkinfam, P., Gagnon; M. (2006). Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction 16* (5), pp. 403-434.

Desmarais, M. C & Baker, R. S. J. d. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction 22* (1-2), pp. 9-38.

D'Mello, S. K., Taylor R. S., Graesser, A. (2007). Monitoring Affective Trajectories during Complex Learning. In *Proceedings of the 29th Annual Cognitive Science Society*, Nashville, USA, August 1st-4th, 2007, pp. 203-208.

D'Mello, S. K , Graig, S. D., Witherspoon, A. M., Graesser, A. (2008). Automatic Detection of Learner's Affect from Conversational Cues. *User Modeling and User-Adapted Interaction 18* (1-2), pp. 45-80.

De Vicente, A. & Pain, H. (2002). Informing the Detection of the Students' Motivational State: an Empirical Study. In *Intelligent Tutoring Systems: 6th International Conference, ITS 2002*, Biarritz, France and San Sebastian, Spain, June 2nd -7th , 2002, pp. 933-943.

Falmagne, J. C., Mathiue, K., Villano, M., Doignon, J. P., Johannesen, L. (1990). Introduction to knowledge spaces: How to build, test, and search them. *Psychological Review 97* (2), pp. 201-224.

Forbes-Riley, K. & Litman D. J. (2004). Predicting Emotion in Spoken Dialogue from Multiple Knowledge Sources. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT/NAACL 2004*, Boston, USA, May 2nd-7th, 2004, pp. 201-208.

Forbes-Riley, K. & Litman D. J. (2006). Modelling User Satisfaction and Student Learning in a Spoken Dialogue Tutoring System with Generic, Tutoring, and User Affect Parameters. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, New York, USA, June 5th-12th, 2006, pp. 264–271.

Fournier-Viger, P., Nkambou, R., Nguifo, E. M. (2010). Building Intelligent Tutoring Systems for Ill-Defined Domains. *Studies in Computational Intelligence, Advances in Intelligent Tutoring Systems 308*, pp. 81-101.

Heller, J., Mayer, B., Hockemeyer, C., Dietrich, A. (2006). Competence-based Knowledge Structures for Personalized Learning: Distributed Resources and Virtual Experiments. *International Journal on E-Learning 5* (1), pp. 75-88.

Kumar, R., Rosé, C. P., Wang Y. C, Joshi, M., Robinson A. (2007). Tutorial Dialogue as Adaptive Collaborative Learning Support. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007*, Los Angeles, USA, July 9th-13th, 2007, pp. 383-390.

Mitrovic, A., Koedinger, K. R., Martin, B. (2003). A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling. *User Modeling 2003: 9th International Conference, UM 2003*, Johnstown, USA, June 22nd-26th, 2003, pp. 313-322.

Mitrovic, A. (2012) Fifteen years of constraints-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction 22* (1-2), pp. 39-72.

Montalvo, O., BakeR, R. S. J. d., Sao Pedro, M. A., Nakama, A., Gobert, J. D. (2010). Identifying Students' Inquiry Planning Using Machine Learning. *In 3rd International Conference on Educational Data Mining*, Pittsburgh, USA, June 11th-13th, 2010, pp. 141-150.

Mota, S. & Picard, R. W. (2003). Automated Posture Analysis for Detecting Learner's Interest Level. In *Computer Vision and Pattern Recognition Workshop, CVPRW 2003*, Madison, USA, June 16th-22nd, 2003, pp.49-55.

Ohlsson, S. (1992). Constraint-based student modeling. *Artificial Intelligence in Education 3* (4), pp. 429–447.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review* 103 (2), pp. 241-262.

Prata, D. N., Baker, R. S. J. d., Costa, E. d. B., Rosé, C. P., Cui, Y., De Carvalho, A. M. J. B. (2009). Detecting and Understanding the Impact of Cognitive and Interpersonal Conflict in Computer Supported Collaborative Learning Envirnment. In *2nd International Conference on Educational Data Mining*, Cordoba, Spain, July 1st-3rd, 2009, pp. 131-140.

Rebolledo-Mendez, G., Du Boulay, B., Luckin, R. (2006). Motivating the learner: an empirical study. In *Intelligent Tutoring Systems: 8th International Conference, ITS 2006*, Jhongli, Taiwan, June 26th-30th, 2006, pp. 545-55.

Rebolledo-Mendez, G., De Freitas., S., Rojano-Caceres, J. R., Garcia-Gaona, A. R. (2010). An Empirical Examination of the Relation Between Attention and Motivation in Computer-Based Education: A Modelling Approach. In *Proceedings of the 23rdInternational Florida Artificial Intelligence Research Society Conference, FLAIRS 2010*, Daytona Beach, USA, May 19th-21st, 2010, pp. 74- 79.

Ritter, S., Anderson, J. R., Koedinger, K. R., Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bullettin & Review 14* (2), pp. 249-255.

Roll, I., Aleven, V., McLaren, B. M., Koedinger, K. R. (2007). Can Help Seeking Be Tutored? Searching for the Secret Sauce of Metacognitive Tutoring. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education, AIED 2007*, Los Angeles, USA, July 9th-13th, 2007, pp. 203-210.

Shih, B., Koedinger, K. R., Scheines, R. (2008). A Response Time Model for Bottom-Out Hints as Worked Examples. In *First International Conference on Educational Data Mining*, Montreal, Canada, June 20th-21th, 2008, pp. 117-126.

Vassileva J., McCalla, G., Greer, J. (2003). Multi-Agent Multi-User Modeling in I-Help. *User Modeling and User-Adapted Interaction 13* (1-2), pp. 179-210.

Walker, E., Walker, S., Rummel, N., Koedinger, K. R. (2010). Using Problem-Solving Context to Access Help Quality in Computer-Mediated Peer Tutoring. In *Intelligent Tutoring Systems: 10th International Conference, ITS 2010*, Pittsburgh, USA, June 14th-18th, 2010, Part I, pp. 145-155.

Alawar, M. W., & Abu Naser, S. S. (2017). CSS-Tutor: *An intelligent tutoring system for CSS and HTML*. International Journal of Academic Research and Development, 2(1), 94-98.

C.J. Butz, S. Hua, R.B. Maguire Department of Computer Science University of Regina Regina, Saskatchewan, Canada S4S 0A2 Email: {butz, huash111, rbm}@cs.uregina.ca (2010). *AWeb-based Bayesian Intelligent Tutoring System for Computer Programming*

“Andes Physics Tutoring System” (2005). *International Journal of Artificial Intelligence in Education.*

Anohina, A, (2007). Advances in Intelligent Tutoring Systems: *Problem-solving Modes and Model of Hints*. International Journal of Computers, Communications & Control, v.II, No.1, pp. 48-55

Baker, R.S., Corbett, A.T. & Koedinger, K.R., *Detecting Student Misuse of Intelligent Tutoring Systems*. Retrieved February 15, 2007, from http://www.psychology.nottingham.ac.uk/staff/lpzrsb/BCK2004MLFinal.pdf

Davidovic, A. (2001). *Learning benefits of structural example-based adaptive tutoring systems*. Phd. University of South Australia, School of Computer and Information

Science. Retrieved February 19, 2007, from http:\\www.ariic.libary.unsw.edu.au/unisa.

Leddo, J. & Kolodziej, J. (1997). *Distributed interactive intelligent tutoring simulation*. ERIC Document Access Code ED416319.

Ong, J. & Ramachandran, S. (2000). *Intelligent tutoring systems: the what and the how*. Retrieved February 14, 2007 from http://www.learningcircuits.org/2000/feb2000/ong.htm.

Ross, S. & Casey, J. (1994). *Using interactive software to develop students' problem solving skills: evaluation of the "intelligent physics tutor."* ERIC Document Access Code ED373754

APPENDIX I

LEARNERS CODE

```
package intelligenttutoringsystem;


import java.awt.Toolkit;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowEvent;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;


/**
 *
 * @author Victor
 */
public class LearnerInterface extends javax.swing.JFrame {


    Connection conn;
    ResultSet rs;
    PreparedStatement pst;
//    private int emotionalCounter = 0;
    public static String knowledgeLvl;
```

```
    public LearnerInterface() {
        initComponents();
    }
private void showMyProfileBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        fnameLBL.setText("FIRST NAME:");
        lnameLBL.setText("LAST NAME:");
        studidLBL.setText("STUDENT ID:");
        ageLBL.setText("AGE:");
        emailLBL.setText("EMAIL:");
        mobileConLBL.setText("MOBILE CONTACT:");
        genderLBL.setText("GENDER:");
        cklLBL.setText("CURRENT KNOWLEDGE LEVEL:");
        lStylLBL.setText("LEARNING STYLE:");

            //Code to read data from wamp/phpMyAdmin database and copy into labels
            Function f = new Function();
            rs = f.find(StudentLogin.myStr);
             try {
               while(rs.next()){
                   fNameLbl.setText(rs.getString("firstName"));
                   lNamelbl.setText(rs.getString("lastName"));
                   studIdLbl.setText(rs.getString("studentID"));
                   ageLbl.setText(rs.getString("age"));
                   emailLbl.setText(rs.getString("email"));
                   mobileContLbl.setText(rs.getString("mobileContact"));
                   genderLbl.setText(rs.getString("gender"));
                   currentKnowLbl.setText(rs.getString("currentKnowLvl"));
                   learnigStylLbl.setText(rs.getString("learningStyle"));

                   }
```

```
            } catch (SQLException ex) {
                Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
            }
        continueBtn.setText("More materials & Assessment");
        knowledgeLvl = currentKnowLbl.getText();
        liveChatBtn.setText("Chat");
        videoTutorials(knowledgeLvl);
        slides(knowledgeLvl);
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    close();
    DomainModule dm = new DomainModule();
    dm.setVisible(true);
}

private void continueBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    if(continueBtn.getLabel().equals("More materials & Assessment")){
        if(knowledgeLvl.equals("Beginner")){
            close();
            BeginnerPedagogicalModule bpm = new BeginnerPedagogicalModule();
            bpm.setVisible(true);
        }
        if(knowledgeLvl.equals("Intermediate")){
            close();
            IntermediatePedagogicalModule ipm = new IntermediatePedagogicalModule();
            ipm.setVisible(true);
        }
        if(knowledgeLvl.equals("Advanced")){
            close();
```

```
                AdvancedPedagogicalModule apm = new AdvancedPedagogicalModule();
                apm.setVisible(true);
            }
        }
    }
    private void liveChatBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        close();
        Server s = new Server();
        s.setVisible(true);
        Client c = new Client();
        c.setVisible(true);
    }
    public static void main(String args[]) {
java.awt.EventQueue.invokeLater(() -> {
            new LearnerInterface().setVisible(true);
        });
    }
public void close(){
        WindowEvent winClosingEvent = new WindowEvent(this, WindowEvent.WINDOW_CLOSING);
        Toolkit.getDefaultToolkit().getSystemEventQueue().postEvent(winClosingEvent);
    }

public class Function{

    Connection c = null;
    ResultSet r = null;
    PreparedStatement p = null;

    public ResultSet find(String s){
        try {
```

```
            c = DriverManager.getConnection("jdbc:mysql://localhost:3306/studentregisterdatabase", "root",
"");
            p = c.prepareStatement("select * from studentregistertable where studentID = ?");
            p.setString(1, s);
            r = p.executeQuery();
        } catch (SQLException ex) {
            Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
        }
        return r;
}
}
private void videoTutorials(String javaKnowLevel) {

//**** BEGINNER VIDEOS ****
//video tutorial 1
        if(javaKnowLevel.equals("Beginner")){
        jLvidTut1.addMouseListener(new MouseListener(){
            @Override
            public void mouseClicked(MouseEvent e) {
                try {
                    //throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.

                    Runtime.getRuntime().exec("rundll32                    url.dll,FileProtocolHandler
"+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\beginners video\\Java Tutorial For Beginners 1
- Introduction and Installing the java (JDK) Step by Step Tutorial.mp4");
                } catch (IOException ex) {
                    Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
                }

            }

            @Override
```

```
        public void mousePressed(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }

        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

    });
  }


  //video tutorial 2
  if(javaKnowLevel.equals("Beginner")){
    jLvidTut2.addMouseListener(new MouseListener(){
        @Override
        public void mouseClicked(MouseEvent e) {
```

```
            try {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

                Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\beginners video\\Java Tutorial For Beginners 6 - Math and Arithmetic Operators in Java.mp4");
            } catch (IOException ex) {
                Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
            }

        }

        @Override
        public void mousePressed(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }

        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseExited(MouseEvent e) {
```

```
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }


    });
  }


   //video tutorial 3
  if(javaKnowLevel.equals("Beginner")){
    jLvidTut3.addMouseListener(new MouseListener(){
        @Override
        public void mouseClicked(MouseEvent e) {
            try {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

                Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\beginners video\\Java Tutorial For Beginners 5 - Getting User Input using Java.mp4");
            } catch (IOException ex) {
                Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
            }


        }


        @Override
        public void mousePressed(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }


        @Override
        public void mouseReleased(MouseEvent e) {
```

```
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.


        }


        @Override

        public void mouseEntered(MouseEvent e) {

            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }


        @Override

        public void mouseExited(MouseEvent e) {

            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }


    });

  }



  //video tutorial 4

  if(javaKnowLevel.equals("Beginner")){

    jLvidTut4.addMouseListener(new MouseListener(){

        @Override

        public void mouseClicked(MouseEvent e) {

            try {

                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.


                Runtime.getRuntime().exec("rundll32                                    url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\beginners video\\Java Tutorial For Beginners 9 - Logical Operators in Java.mp4");

            } catch (IOException ex) {
```

```
                Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
            }


        }

        @Override
        public void mousePressed(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }

        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

    });
  }
  //End of Beginner video tutorial
```

```
   //**** INTERMEDIATE VIDEOS ****
//video tutorial 1
        if(javaKnowLevel.equals("Intermediate")){
        jLvidTut1.addMouseListener(new MouseListener(){
            @Override
            public void mouseClicked(MouseEvent e) {
                try {
                    //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.


                    Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\Intermediate\\Java Tutorial For Beginners 13 - Arrays in Java.mp4");
                } catch (IOException ex) {
                    Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
                }


            }


            @Override
            public void mousePressed(MouseEvent e) {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
            }


            @Override
            public void mouseReleased(MouseEvent e) {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
```

```
        }

        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

    });
  }

  //video tutorial 2
  if(javaKnowLevel.equals("Intermediate")){
    jLvidTut2.addMouseListener(new MouseListener(){
        @Override
        public void mouseClicked(MouseEvent e) {
            try {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

                Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\Intermediate\\Java Tutorial For Beginners 17 - Parameter passing and Returning a Value from a Method.mp4");
            } catch (IOException ex) {
                Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
            }
```

```
        }

        @Override
        public void mousePressed(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }

        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

    });
  }

   //video tutorial 3
  if(javaKnowLevel.equals("Intermediate")){
     jLvidTut3.addMouseListener(new MouseListener(){
```

```
    @Override
    public void mouseClicked(MouseEvent e) {
        try {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

            Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\Intermediate\\Java Tutorial For Beginners 20 - Method Overloading in Java.mp4");
        } catch (IOException ex) {
            Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    @Override
    public void mousePressed(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

    }

    @Override
    public void mouseEntered(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
```

```
        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

    });
  }

  //video tutorial 4
  if(javaKnowLevel.equals("Intermediate")){
    jLvidTut4.addMouseListener(new MouseListener(){
        @Override
        public void mouseClicked(MouseEvent e) {
            try {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

                Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\Intermediate\\Java Tutorial For Beginners 20 - Method Overloading in Java.mp4");
            } catch (IOException ex) {
                Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
            }

        }

        @Override
        public void mousePressed(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }
```

```
        @Override
        public void mouseReleased(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

        }

        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }

    });
  }
  //End of Intermediate video tutorial


  //**** ADVANCED VIDEOS ****
//video tutorial 1
    if(javaKnowLevel.equals("Advanced")){
    jLvidTut1.addMouseListener(new MouseListener(){
        @Override
        public void mouseClicked(MouseEvent e) {
            try {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
```

```
            Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\Advanced\\Java Threads Tutorial 2 - How to Create Threads in Java by Extending Thread Class.mp4");
        } catch (IOException ex) {
            Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    @Override
    public void mousePressed(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

    }

    @Override
    public void mouseEntered(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public void mouseExited(MouseEvent e) {
        //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
```

```
        });
    }


    //video tutorial 2
    if(javaKnowLevel.equals("Advanced")){
        jLvidTut2.addMouseListener(new MouseListener(){
            @Override
            public void mouseClicked(MouseEvent e) {
                try {
                    //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.


                    Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "+"C:\\Users\\Victor\\Desktop\\Prev Files\\MScThesis Files\\Advanced\\Java Tutorial For Beginners 18 - Classes and Objects in Java.mp4");
                } catch (IOException ex) {
                    Logger.getLogger(LearnerInterface.class.getName()).log(Level.SEVERE, null, ex);
                }


            }


            @Override
            public void mousePressed(MouseEvent e) {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
            }


            @Override
            public void mouseReleased(MouseEvent e) {
                //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
```

```
        }


        @Override
        public void mouseEntered(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }


        @Override
        public void mouseExited(MouseEvent e) {
            //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
        }


    });
  }
```