##### **DECLARATION**

I hereby certify that the work which is being presented by Aman Kaushik and Anchit Jatana in partial fulfillment of requirements for the award of degree of B.E. in Computer Science & Engineering submitted at Chandigarh College of Engineering and Technology under PUNJAB UNIVERSITY, CHANDIGARH is an authentic record of my own work carried out under the supervision of Mr. Bhushan Garg.

The matter presented in this report has not been submitted by me in any other University / Institute for the award of B.Tech Degree.

Anchit Jatana (CO10307)

CSE 5th Semester

**ACKNOWLEDGEMENT**

It has been a great honor and privilege to undergo training at Techno Campus, Chandigarh.

I am very much thankful to Mr. Bhushan Garg for providing all facilities and support to meet my project requirements. We would like to take opportunity to express our humble gratitude to him**,** under whom we executed this project. His constant guidance and willingness to share his vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks.

A note of thanks to respected Dr. M.S. Gujral, Principal, CCET, Lect. Pooja Singla , HOD CSE, Training Incharge of CSE Dept. Lect. Jagreet Kaur who has always motivated and given expert guidance to us.

I wish to express our sincere thanks to our college teachers, parents and friends who always inspired and cooperated with us, which helped in the successful completion of the training program.

Although there may be many who remain unacknowledged in this humble note of gratitude, there are none who remain unappreciated

Anchit Jatana (CO10307)

CSE 5th Semester

**ABSTRACT**

Title of the project : User Adaptive Examination System

Duration: 6 Weeks

Project Category: Java Based Application

The project named User Adaptive Examination System is developed using JAVA and SQL during the industrial training. This project has been developed in partial fulfillment for the degree of B.TECH. (CSE) , from Chandigarh College of Engineering & Technology, Chandigarh.

User Adaptive Examination System allows a user to conduct a computer adaptive test. To appear in the test the candidates will have to register themselves by filling up an online registration form. After completion of registration a hall ticket card will be generated along with the a unique hall ticket number. The registration process will continue for a given period of time and after that the registration window will shut down. To begin the test, the candidate has to login with his name, date of birth and hall ticket number. The test duration is 30 minutes in which the candidate has to answer 30 questions with varying difficulty level. A question cannot be skipped and the candidate cannot refer to a previous question. At the end of the 30 minutes a score card will be generated displaying the total marks obtained, however a user can also end the test prior to the end of 30 minutes.

Technically earlier my knowledge about JAVA was limited but now I am very comfortable in this language. My knowledge about the various processes involved in the field of software has also improved considerably.

Handling this project has given us an ample amount of exposure, confidence and sense of responsibility.

**PREFACE**

As a part of 45 days training, we have made a Java based application.

The main objective of this project is to provide an innovative examination system which one of its own kind.

This report provides the information regarding my project.

This project is a complete software solution ,catering to all the requirements of various examination systems:

1. Registration of candidates for the Exam(For Eligible Candidates).
2. Question database building facility(For the Examiner).
3. Login and Exam taking facility(For the Registered Candidates).

**INTRODUCTION**

User Adaptive Examination System is a complete software solution ,catering to all the requirements of various examination systems. To appear in the test the candidates will have to register themselves by filling up an online registration form. After completion of registration a hall ticket card will be generated along with the a unique hall ticket number. The registration process will continue for a given period of time and after that the registration window will shut down.

At the time of the test, the user will have to enter his hall ticket number, name and date of birth to begin the test.

The overview of the test is as follows:

* Test duration will be of 30 minutes.
* The questions will be selected from three subjects namely : Quantative Analysis, Verbal Ability, and Logical Reasoning.
* There will be 30 questions in the test, 10 from each subject, with easy, medium and hard difficulty level.
* The first question in the test will be of average difficulty. If the answer is right, the next question will be slightly difficult. If the answer is wrong, the next question will be slightly easier.
* Marks distribution is as follows:

1. Hard Difficulty Level Questions = 6 marks
2. Medium Difficulty Level Questions = 4 marks
3. Easy Difficulty Level Questions = 3 marks
4. For each wrong answer, 1 mark will be deducted from the total score.

* A question cannot be skipped and the candidate cannot refer to a previous question.
* The candidate can end the test prior to 30 minutes, otherwise the test will end automatically at the end of 30 minutes.

After the test is over a report card mentioning the total marks scored by the candidate will be generated. The result of the candidate will be stored in the centrally located database.

**TOOLS**

* Hardware & Software Requirements:
  + Processor: Pentium IV/equivalent or Higher.

****

* + MY SQL Database (at the Back-End)

**INTRODUCTION TO JAVA**

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, Mike Sheridan at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called “Oak” but was renamed “Java” in 1995. Between the initial implementation of Oak in the fall of 1992 and the public announcement of Java in the spring of 1995, many more people contributed to the design and evolution of the language. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin, and Tim Lindholm were key contributors to the maturing of the original prototype. Somewhat surprisingly, the original impetus for Java was not the Internet! Instead, the primary motivation was the need for a platform-independent (that architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls. As you can probably guess, many different types of CPUs are used as controllers. The trouble with C and C++ (and most other languages) is that they are designed to be compiled for a specific target. Although it is possible to compile a C++ program for just about any type of CPU, to do so requires a full C++ compiler targeted for that CPU. The problem is that compilers are expensive and time-consuming to create. An easier— and more cost-efficient—solution was needed. In an attempt to find such a solution, Gosling and others began work on a portable, platform-independent language that could be used to produce code that would run on a variety of CPUs under differing environments. This effort ultimately led to the creation of Java. About the time that the details of Java were being worked out, a second, and ultimately more important, factor was emerging that would play a crucial role in the future of Java. This second force was, of course, the World Wide Web. Had the Web not taken shape at about the same time that Java was being implemented, Java might have remained a useful but obscure language for programming consumer electronics. However, with the emergence of the World Wide Web, Java was propelled to the forefront of computer language design, because the Web, too, demanded portable programs.

Most programmers learn early in their careers that portable programs are as elusive as they are desirable. While the quest for a way to create efficient, portable (platform-independent) programs is nearly as old as the discipline of programming itself, it had taken a back seat to other, more pressing problems. Further, because much of the computer world had divided itself into the three competing camps of Intel, Macintosh, and UNIX, most programmers stayed within their fortified boundaries, and the urgent need for portable code was reduced. However, with the advent of the Internet and the Web, the old problem of portability returned with a vengeance. After all, the Internet consists of a diverse, distributed universe populated with many types of computers, operating systems, and CPUs. Even though many types of platforms are attached to the Internet, users would like them all to be able to run the same program. What was once an irritating but low-priority problem had become a high-profile necessity.

By 1993, it became obvious to members of the Java design team that the problems of portability frequently encountered when creating code for embedded controllers are also found when attempting to create code for the Internet. In fact, the same problem that Java was initially designed to solve on a small scale could also be applied to the

Internet on a large scale. This realization caused the focus of Java to switch from consumer electronics to Internet programming. So, while the desire for an architecture neutral programming language provided the initial spark, the Internet ultimately led to Java’s large-scale success.

As mentioned earlier, Java derives much of its character from C and C++. This is by intent. The Java designers knew that using the familiar syntax of C and echoing the object-oriented features of C++ would make their language appealing to the legions of experienced C/C++ programmers. In addition to the surface similarities, Java shares some of the other attributes that helped make C and C++ successful. First, Java was designed, tested, and refined by real, working programmers. It is a language grounded in the needs and experiences of the people who devised it. Thus, Java is also a programmer’s language. Second, Java is cohesive and logically consistent. Third, except for those constraints imposed by the Internet environment, Java gives you, the programmer, full control. If you program well, your programs reflect it. If you program poorly, your programs reflect that, too. Put differently, Java is not a language with training wheels. It is a language for professional programmers. Because of the similarities between Java and C++,

**Java Database Connectivity (JDBC)**

From the start, the developers of the Java technology at Sun were aware of the potential that Java showed for working with databases. Starting in 1995, they began working on extending the standard Java library to deal with SQL access to databases. What they first hoped to do was to extend Java so that it could talk to any random database, using only "pure" Java. It didn't take them very long to realize that this is an impossible task: there are simply too many Some database vendors now build a Java Virtual Machine into the database itself so that you can write stored procedures in Java. This technology is part of the SQLJ specification. For more information on SQLJ, Over the years, many technologies were invented to make database access more efficient and fail-safe. Standard databases support indexes, triggers, stored procedures, and transaction management. JDBC supports all these features, but we do not discuss them in detail in this chapter. One could write an entire book on advanced database programming for the Java platform, and many such books have been written. What all the database vendors and tool vendors *did* agree on was that it would be useful if Sun provided a pure Java API for SQL access *along* with a driver manager to allow third-party drivers to connect to specific databases. Database vendors could provide their own drivers to plug into the driver manager. There would then be a simple mechanism for registering third party drivers with the driver manager—the point being that all the drivers needed to do was follow the requirements laid out in the driver manager API. After a fairly long period of public discussion, the API for database access became the JDBC API, and the rules for writing drivers were encapsulated in the JDBC service provider interface, or SPI. (The SPI is of interest only to database vendors and database tool providers; we don't cover it here.) This protocol follows the very successful model of Microsoft's ODBC, which provided a C programming language interface for database access. Both JDBC and ODBC are based on the same idea: Programs written according to the JDBC API would talk to the JDBC driver manager, which, in turn, would use the drivers that were plugged into it at that moment to talk to the actual database. More precisely, the JDBC consists of two layers. The top layer is the JDBC API. This API communicates with the JDBC manager driver API, sending it the various SQL statements. The manager should (transparently to the programmer) communicate with the various third-party drivers that actually connect to the database and return the information from the query or perform the action specified by the query. All this means the Java/JDBC layer is all that most programmers will ever have to deal with. A list of JDBC drivers currently available can be found at the web site

The JDBC specification will actually allow you to pass any string to the underlying driver. The driver can pass this string to the database. This feature allows you to use specialized versions of SQL that may be supported by the driver and its associated database. JDBC drivers are classified into the following *types:* A *type 1 driver* translates JDBC to ODBC and relies on an ODBC driver to communicate with the database. Sun includes one such driver, the *JDBC/ODBC bridge,* with the JDK. However, the bridge requires deployment and proper configuration of an ODBC driver. When JDBC was first released, the bridge was handy for testing, but it was never intended for production use. At this point, there are plenty of better drivers available, and we advise against using the JDBC/ODBC bridge. A *type 2 driver* is a driver, written partly in the Java programming language and partly in native code, that communicates with the client API of a database. When you use such a driver, you must install some platform-specific code in addition to a Java library. A *type 3 driver* is a pure Java client library that uses a database-independent protocol to communicate database requests to a server component, which then translates the requests into a database-specific protocol. The client library is independent of the actual database, thus simplifying deployment. A *type 4 driver* is a pure Java library that translates JDBC requests directly to a database-specific protocol. Most database vendors supply either a type 3 or type 4 driver with their database. Furthermore, a number of third-party companies specialize in producing drivers with better standards conformance, support for more platforms, better performance, or, in some cases, simply better reliability than the drivers that are provided by the database vendors. In summary, the ultimate goal of the JDBC is to make possible the following: Programmers can write applications in the Java programming language to access any database, using standard SQL statements—or even specialized extensions of SQL— while still following Java language conventions. Database vendors and database tool vendors can supply the low-level drivers. Thus, they can optimize their drivers for their specific products.

**Structured Query Language (SQL)**

JDBC is an interface to SQL, which is the interface to essentially all modern relational databases. Desktop databases usually have a graphical user interfaces that lets users manipulate the data directly, but server-based databases are accessed purely through SQL. Most desktop databases have a SQL interface as well, but it often does not support the full range of ANSI SQL92 features, the current standard for SQL. The JDBC package can be thought of as nothing more than an application programming interface (API) for communicating SQL statements to databases. We will give a short introduction to SQL in this section. If you have never seen SQL before, you may not find this material sufficient. If so, you should turn to one of the many books on the topic. We recommend *Client/Server Databases* by James Martin and Joe Leben [Prentice-Hall 1998] orthe venerable and opinionated book. Each column has a *column name.* The rows contain the actual data. These are sometimes called *records.*

As the example database for this book, we use a set of tables that describe a collection of books on HTML. with a table of publishers. Both the Books and the Publishers table contain a numerical code for the publisher. When we join both tables on the publisher code, we obtain a *query result* made up of values from the joined tables. Each row in the result contains the information about a book, together with the publisher name and Web page URL. Note that the publisher names and URLs are duplicated across several rows since we have several rows with the same publisher.

The Authors table

The Books table

The Books Authors table

Table 4-4. The Publishers table

Two tables joined together

The benefit of joining tables is to avoid unnecessary duplication of data in the database tables. For example, a naive database design might have had columns for the publisher name and URL right in the Books table. But then the database itself, and not just the query result, would have many duplicates of these entries. If a publisher's Web address changed, *all* entries would need to be updated. Clearly, this is somewhat error prone. In the relational model, we distribute data into multiple tables such that no information is ever unnecessarily duplicated.

SELECT Books.ISBN, Books.Price, Books.Title,

Books.Publisher\_Id, Publishers.Name, Publishers.URL

FROM Books, Publishers

WHERE Books.Publisher\_Id = Publishers.Publisher\_Id

In the remainder of this section, you will learn how to write such queries. If you are already familiar with SQL, just skip this section. By convention, SQL keywords are written in all caps, although this is not necessary. The SELECT operation is quite flexible. You can simply select all elements in the Books

table with the following query:

SELECT \* FROM Books

The FROM statement is required in every SQL SELECT statement. The FROM clause tells the

database which tables to examine to find the data.

You can choose the columns that you want.

SELECT ISBN, Price, Title

FROM Books

You can restrict the rows in the answer with the WHERE clause.

**Text Fields**

The Swing text field is encapsulated by the JTextComponentclass, which extends JComponent. It provides functionality that is common to Swing text components. One of its subclasses is JTextField, which allows you to edit one line of text. Some of its constructors are shown here:

JTextField( )

JTextField(int*cols*)

JTextField(String *s*, int*cols*)

JTextField(String *s*)

Here, *s* is the string to be presented, and *cols* is the number of columns in the text field. The following example illustrates how to create a text field. The applet begins by getting its content pane, and then a flow layout is assigned as its layout manager. Next, a JTextFieldobject is created and is added to the content pane.

**Buttons**

Swing buttons provide features that are not found in the Button class defined by the AWT. For example, you can associate an icon with a Swing button. Swing buttons are subclasses of the AbstractButton class, which extends JComponent. AbstractButton contains many methods that allow you to control the behavior of buttons, check boxes, and radio buttons. For example, you can define different icons that are displayed for the component when it is disabled, pressed, or selected. Another icon can be used as a *rollover* icon, which is displayed when the mouse is positioned over that component.

The following are the methods that control this behavior:

voidsetDisabledIcon(Icon *di*)

voidsetPressedIcon(Icon *pi*)

voidsetSelectedIcon(Icon *si*)

voidsetRolloverIcon(Icon *ri*)

Here, *di*, *pi*, *si*, and *ri*are the icons to be used for these different conditions.

The text associated with a button can be read and written via the following methods:

String getText( )

voidsetText(String *s*)

Here, *s* is the text to be associated with the button.

Concrete subclasses of AbstractButtongenerate action events when they are

pressed. Listeners register and unregister for these events via the methods shown here:

voidaddActionListener(ActionListener*al*)

voidremoveActionListener(ActionListener*al*)

Here, *al* is the action listener. AbstractButton is a superclass for push buttons, check boxes, and radio buttons.

Each is examined next.

**The JButton Class**

The JButtonclass provides the functionality of a push button. JButton allows an icon, a string, or both to be associated with the push button. Some of its constructors are shown here:

JButton(Icon *i*)

JButton(String *s*)

JButton(String *s*, Icon *i*)

Here, *s* and *i* are the string and icon used for the button.

The following example displays four push buttons and a text field. Each button

displays an icon that represents the flag of a country. When a button is pressed, the name of that country is displayed in the text field. The applet begins by getting its content pane and setting the layout manager of that pane. Four image buttons are created and added to the content pane. Next, the applet is registered to receive action events that are generated by the buttons. A text field is then created and added to the applet. Finally, a handler for action events displays the command string that is associated with the button.

**Check Boxes**

The JCheckBoxclass, which provides the functionality of a check box, is a concrete implementation of AbstractButton. Its immediate superclass is JToggleButton, which provides support for two-state buttons. Some of its constructors are shown here:

JCheckBox(Icon *i*)

JCheckBox(Icon *i*, boolean*state*)

JCheckBox(String *s*)

JCheckBox(String *s*, boolean*state*)

JCheckBox(String *s*, Icon *i*)

JCheckBox(String *s*, Icon *i*, boolean*state*)

Here, *i* is the icon for the button. The text is specified by *s*. If *state* is true, the check box is initially selected. Otherwise, it is not.

The state of the check box can be changed via the following method:

voidsetSelected(boolean*state*)

Here, *state* is true if the check box should be checked.

The following example illustrates how to create an applet that displays four check

boxes and a text field. When a check box is pressed, its text is displayed in the text field. The content pane for the JApplet object is obtained, and a flow layout is assigned as its layout manager. Next, four check boxes are added to the content pane, and icons are assigned for the normal, rollover, and selected states. The applet is then registered to receive item events. Finally, a text field is added to the content pane. When a check box is selected or deselected, an item event is generated. This is handled by itemStateChanged( ). Inside itemStateChanged( ), the getItem( ) method gets the JCheckBoxobject that generated the event. The getText( ) method gets the text

for that check box and uses it to set the text inside the text field.

**Radio Buttons**

Radio buttons are supported by the JRadioButtonclass, which is a concrete

implementation of AbstractButton. Its immediate superclass is JToggleButton, which provides support for two-state buttons. Some of its constructors are shown here:

JRadioButton(Icon *i*)

JRadioButton(Icon *i*, boolean*state*)

JRadioButton(String *s*)

JRadioButton(String *s*, boolean*state*)

JRadioButton(String *s*, Icon *i*)

JRadioButton(String *s*, Icon *i*, boolean*state*)

Here, *i* is the icon for the button. The text is specified by *s*. If *state* is true, the button is initially selected. Otherwise, it is not. Radio buttons must be configured into a group. Only one of the buttons in that group can be selected at any time. For example, if a user presses a radio button that is in a group, any previously selected button in that group is automatically deselected. The ButtonGroupclass is instantiated to create a button group. Its default constructor is invoked for this purpose. Elements are then added to the button group via the following method:

void add(AbstractButton *ab*)

Here, *ab* is a reference to the button to be added to the group.

The following example illustrates how to use radio buttons. Three radio buttons

and one text field are created. When a radio button is pressed, its text is displayed in the text field. First, the content pane for the JAppletobject is obtained and a flow layout is assigned as its layout manager. Next, three radio buttons are added to the content pane. Then, a button group is defined and the buttons are added to it. Finally, a text field is added to the content pane.

Radio button presses generate action events that are handled by actionPerformed( ). The getActionCommand( ) method gets the text that is associated with a radio button and uses it to set the text field.

**Combo Boxes**

Swing provides a *combo box* (a combination of a text field and a drop-down list)

through the JComboBoxclass, which extends JComponent. A combo box normally displays one entry. However, it can also display a drop-down list that allows a user to select a different entry.

You can also type your selection into the text field. Two of JComboBox’s constructors are shown here:

JComboBox( )

JComboBox(Vector *v*)

Here, *v* is a vector that initializes the combo box.

Items are added to the list of choices via the addItem( **)** method, whose signature is

shown here:

voidaddItem(Object *obj*)

Here, *obj*is the object to be added to the combo box. The following example contains a combo box and a label. The label displays an icon. The combo box contains entries for “France”, “Germany”, “Italy”, and “Japan”. When a country is selected, the label is updated to display the flag for that country.

importjava.awt.\*;

importjava.awt.event.\*;

importjavax.swing.\*;

/\*

<applet code="JComboBoxDemo" width=300 height=100>

</applet>

\*/

public class JComboBoxDemo extends JApplet

implementsItemListener {

JLabeljl;

ImageIconfrance, germany, italy, japan;

public void init() {

Container contentPane = getContentPane();

contentPane.setLayout(new FlowLayout());

JComboBoxjc = new JComboBox();

jc.addItem("France");

jc.addItem("Germany");

jc.addItem("Italy");

jc.addItem("Japan");

jc.addItemListener(this);

contentPane.add(jc);

jl = new JLabel(new ImageIcon("france.gif"));

contentPane.add(jl);

}

public void itemStateChanged(ItemEventie) {

String s = (String)ie.getItem();

jl.setIcon(new ImageIcon(s + ".gif"));

}

}

**Scroll Panes**

A *scroll pane* is a component that presents a rectangular area in which a component may be viewed. Horizontal and/or vertical scroll bars may be provided if necessary.

Scroll panes are implemented in Swing by the JScrollPaneclass, which extends

JComponent. Some of its constructors are shown here:

JScrollPane(Component *comp*)

JScrollPane(int*vsb*, int*hsb*)

JScrollPane(Component *comp*, int*vsb*, int*hsb*)

Here, *comp* is the component to be added to the scroll pane. *vsb*and *hsb*are int

constants that define when vertical and horizontal scroll bars for this scroll pane are shown. These constants are defined by the ScrollPaneConstantsinterface. Some examples of these constants are described as follows:

Constant Description

HORIZONTAL\_SCROLLBAR\_ALWAYS Always provide horizontal scroll bar

HORIZONTAL\_SCROLLBAR\_AS\_NEEDED Provide horizontal scroll bar,

if needed VERTICAL\_SCROLLBAR\_ALWAYS Always provide vertical

scroll bar VERTICAL\_SCROLLBAR\_AS\_NEEDED Provide vertical scroll bar,

if needed

Here are the steps that you should follow to use a scroll pane in an applet:

1. Create a JComponentobject.

2. Create a JScrollPaneobject. (The arguments to the constructor specify the

component and the policies for vertical and horizontal scroll bars.)

3. Add the scroll pane to the content pane of the applet.

**FEASIBILITY STUDY**

The feasibility study is used to determine if the project should get the go-ahead. If the project is to proceed, the feasibility study will produce a project plan and budget estimates for the future stages of development. Feasibility is the determination of whether or not a project is worth doing the process followed in making this determination is called a feasibility study. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternativ*e.* Feasibility study is carried out to select the best system that meets system performance requirements**.**

**Different types of feasibility study:**

* Technical feasibility
* Operational feasibility
* Economic feasibility

**1. TECHNICAL FEASIBILITY:**

This is concerned with the specifying equipment and software that will successfully satisfy the requirements. The proposed system is technically feasible as it can be developed easily with the help of available technology. The proposed system requires MYSQL which is used as back-end and Java as front end.

**In the technical needs of the system these points are considered.**

* The facility to produce in given time.
* Response time under conditions.
* Availability to process a certain volume of transaction at a particular speed.

**2. ECONOMICAL FEASIBILITY:**

Economic analysis is the most frequently used technique for evaluating the effectiveness of the proposed system. The producer is to determine the benefits and the saving that are expected from a proposed system and compare them with the proposed system.

The only tangible benefits proposed that the manual work and burden is reduced maximum as possible, resulting the reduction in manpower requirement and cost incurred on manpower as well. The system provides many benefits that can’t be measured in terms of money for e.g. user friendliness, more efficient user response, maintenance of database etc.

**3. OPERATIONAL FEASIBILITY:**

The proposed system is highly user friendly and it is much easily to interact with the user. Therefore, the user will easily accept the system as data entry system and queries can be easily solved. Initial stages of the system might face some resistance but once complete automation is achieved and operators are trained. The system will provide maximum easiness.

It is mainly related to human organizational and political aspects in it we consider:

* What changes will be brought with the system?
* What organizational structures are distributed?
* What new skill will be required?

**SYSTEM ANALYSIS**

The Analysis model :-The analysis model must achieve three primary objectives:

* To establish the basis for the enhancement of a software design.
* To define a set of requirements that can be validated once
* To describe what the customer requires. The software is completely enhanced.

System analysis is an explicit formal inquiry carried out to help someone identify a better [course of action](http://pespmc1.vub.ac.be/ASC/COURSE_ACTIO.html) and make a better decision than he might otherwise have made. The characteristic attributes of a problem situation where systems analysis is called upon are complexity of the issue and [uncertainty](http://pespmc1.vub.ac.be/ASC/UNCERTAINTY.html) of the outcome of any course of action that might reasonably be taken. Systems analysis usually has some combination of the following: identification and re-identification) of [objectives,](http://pespmc1.vub.ac.be/ASC/OBJECTIVE.html) [constraint](http://pespmc1.vub.ac.be/ASC/CONSTRAINT.html)s, and alternative courses of action; examination of the probable [consequences](http://pespmc1.vub.ac.be/ASC/CONSEQUENCE.html) of the alternatives in terms of costs, benefits, and [risks;](http://pespmc1.vub.ac.be/ASC/RISK.html) presentation of the results in a comparative framework so that the decision maker can make an informed choice from among the alternatives.

There are several specific kinds or focuses of systems analysis for which different terms are used: A systems analysis related to public decisions is often referred to as a POLICY ANALYSIS. A systems analysis that concentrates on comparison and ranking of alternatives

on basis of their known characteristics is referred to as DECISION ANALYSIS**.**

**SYSTEM DESIGN**

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to script programs during the development phase. Program designs are constructed in various ways. Using a top-down approach, designers first identify and link major program components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections. Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections.  
Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design. Designers should carefully document completed designs. Detailed documentation enhances a programmer’s ability to develop programs and modify them after they are placed in production. The documentation also helps management ensure final programs are consistent with original goals and specifications. Organizations should create initial testing, conversion, implementation, and training plans during the design phase. Additionally, they should draft user, operator, and maintenance manuals.

**IMPLEMENTATION**

The implementation phase involves installing approved applications into production environments. Primary tasks include announcing the implementation schedule, training end users, and installing the product. Additionally, organizations should input and verify data, configure and test system and security parameters, and conduct post-implementation reviews. Management should circulate implementation schedules to all affected parties and should notify users of any implementation responsibilities. After organizations install a product, pre-existing data is manually input or electronically transferred to a new system. Verifying the accuracy of the input data and security configurations is a critical part of the implementation process. System design is the creative act of invention, developing new inputs, a database, offline files, procedures and output for processing business to meet an organization objective. System design builds information gathered during the system analysis.

## The designing phase of any software development is carried out in the following stages:

## Architectural design (high level design)

## Detailed design(low level design or physical design)

**INDUSTRY**

Techno Campus is a respected learning solutions provider which is dedicated to create success stories for their customers. They have unique integrated learning solutionswhich is a proven approach and can be customized to the individual training needs in the area of Technical Training, Desktop Applications Training, Professional Development Training.

It is certified from Aptech which was established in 1986 and is a pioneer in IT education which so has trained more than 6.4 million students.

Techno Campus having development office in Chandigarh and Office in Jalandhar, Punjab, is one of the leading Software Development company with extensive experience in designing and development cutting edge software solutions. We are a group of highly motivated IT professionals with an ability to innovate a strong desire to excel.

Techno Campus is a customer focused company working to provide software solutions, which are delivered with Reliability, Timeliness, Flexibility and low cost customer satisfaction as the primary aim.

The designing solutions are purely based on business and technical requirements of the project in hand. Software Development Life Cycle is strictly followed to ensure the timely delivery of the project without compromising on quality.

Techno Campus provides a broad range of offshore outsourcing services including Application and Systems Level Programming on Leading Technologies. It also provides wide range of Networking Solutions for Corporates, Industries and Software Development Firms.

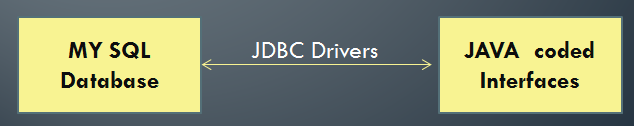
Techno Campus has a niche in providing training to Engineering & MCA Students. Thousands of Engineering students from Chandigarh, Punjab, Haryana, Rajasthan andHimachal have successfully completed their Industrial and Summer Training.

The focus of the Industrial Training is to facilitate students with Live Projects on latesttechnologies like VB.NET, ASP.NET, ORACLE DBA/DEVELOPER, J2EE, JDO, STRUTS, LINUX, Networking under the strict guidance of Industry Experts.

**WORK**

The basic layout for the User Adaptive Examination System is:

* On the User-End we have JAVA coded interfaces.
* At the Back-End we have MY SQL Database.
* Linking these two ends is a “Linking Channel” or the drivers called the JDBC(JAVA Database Connectivity) Drivers.



This project is a complete software solution ,catering to all the requirements of various examination systems:

1. Registration of candidates for the Exam(For Eligible Candidates).
2. Question database building facility(For the Examiner).
3. Login and Exam taking facility(For the Registered Candidates).

To solve each of the Purpose mentioned above, there is a set of 7 Modules(coded in JAVA)

* Module #1 & #2 –For Registration process.
* Module #3,#4,#5,#6 –For Exam Taking.
* Module # 7 –Question Database building.

These modules are linked with the “MY SQL database” at the back-end , which offers a purpose to the operational functionality to each of these

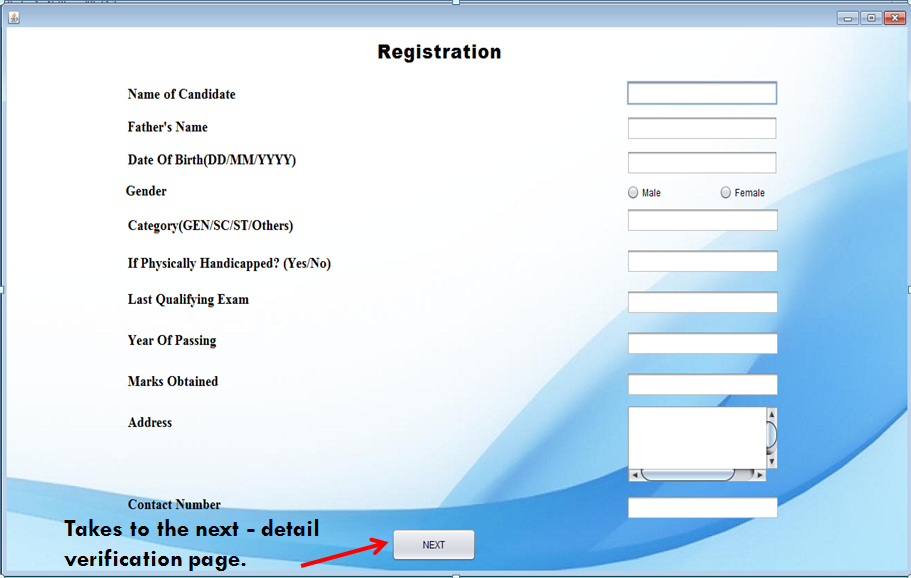
**Various Interfaces of the project :**

1. Registration interface.
2. Detail checking page (checking and submission

of details entered).

1. Login interface.
2. Rules and regulations of the examination.
3. Test taking interface.
4. Result displaying interface.
5. Question database building interface.

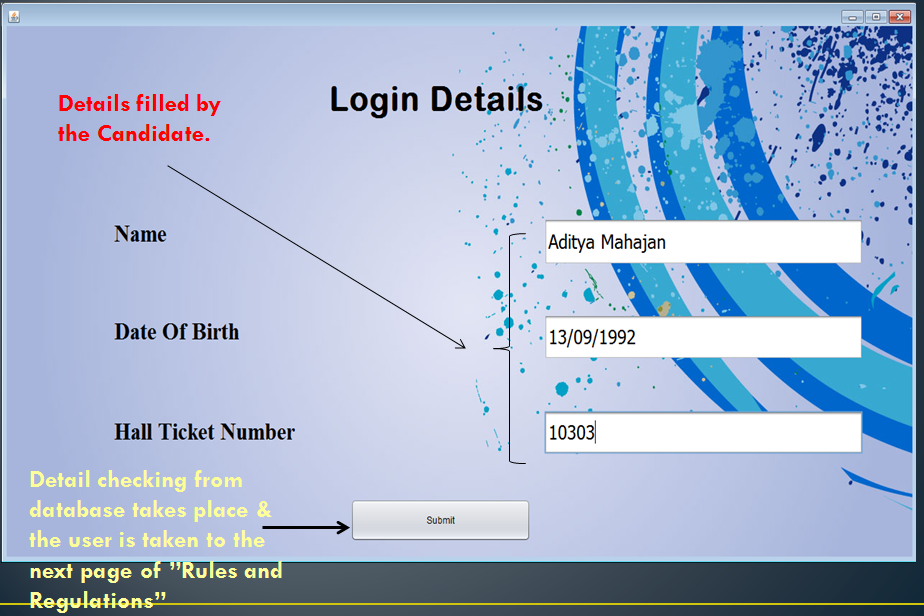
**Registration Interface**



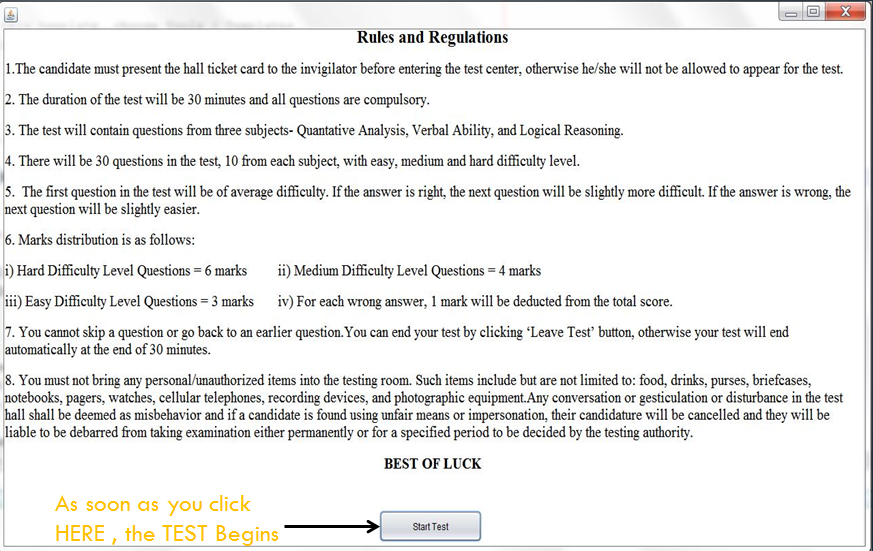
**Submission Page**



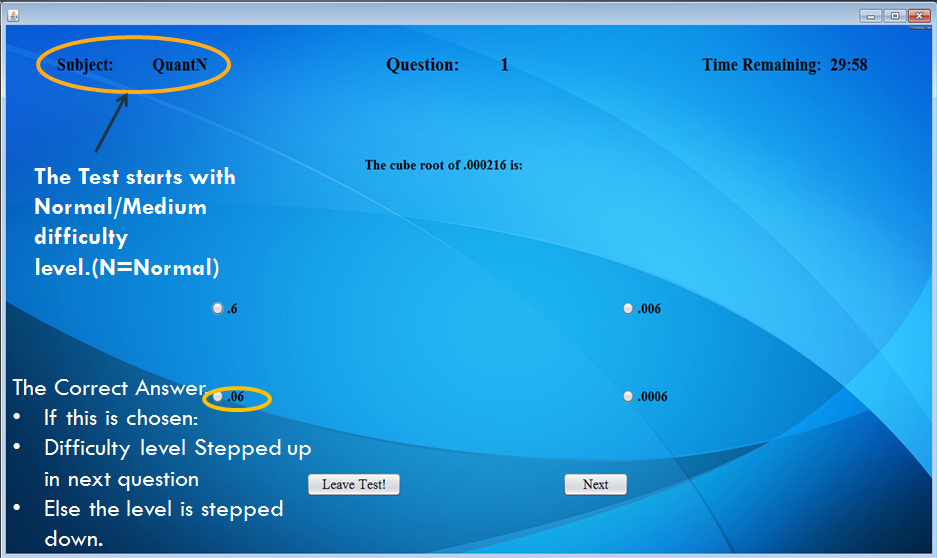
**Login Interface**

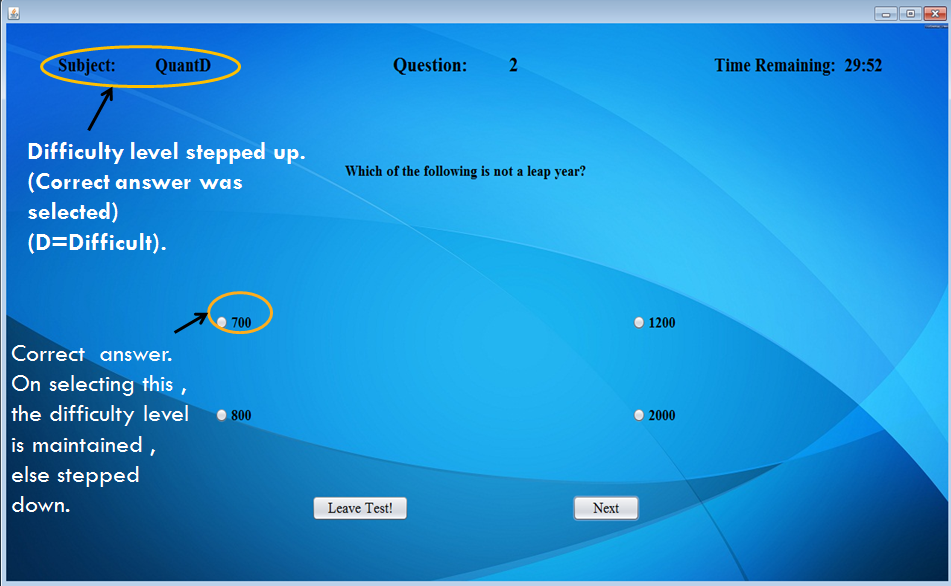


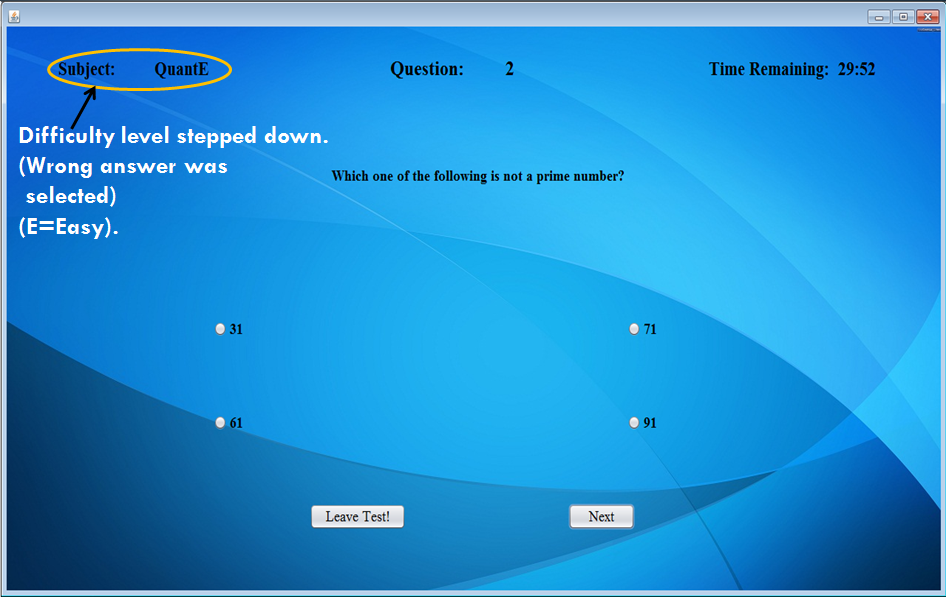
**Rules and Regulations Page**



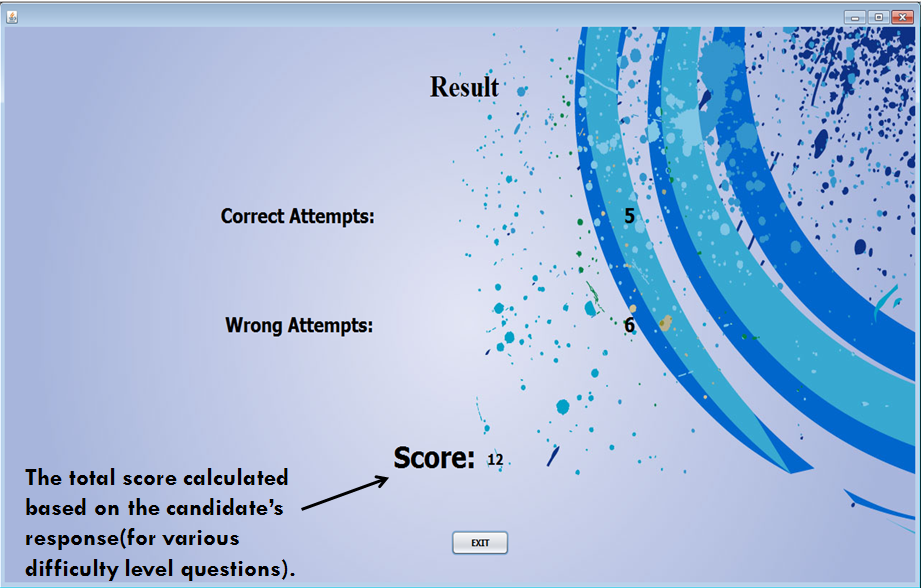
**Test Interfaces**



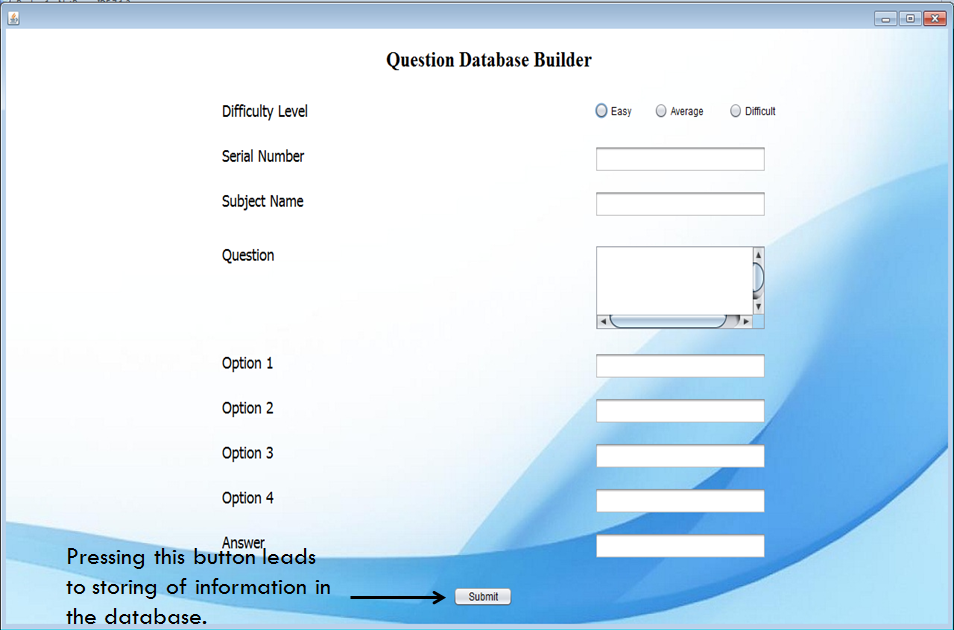




**Result Interface**

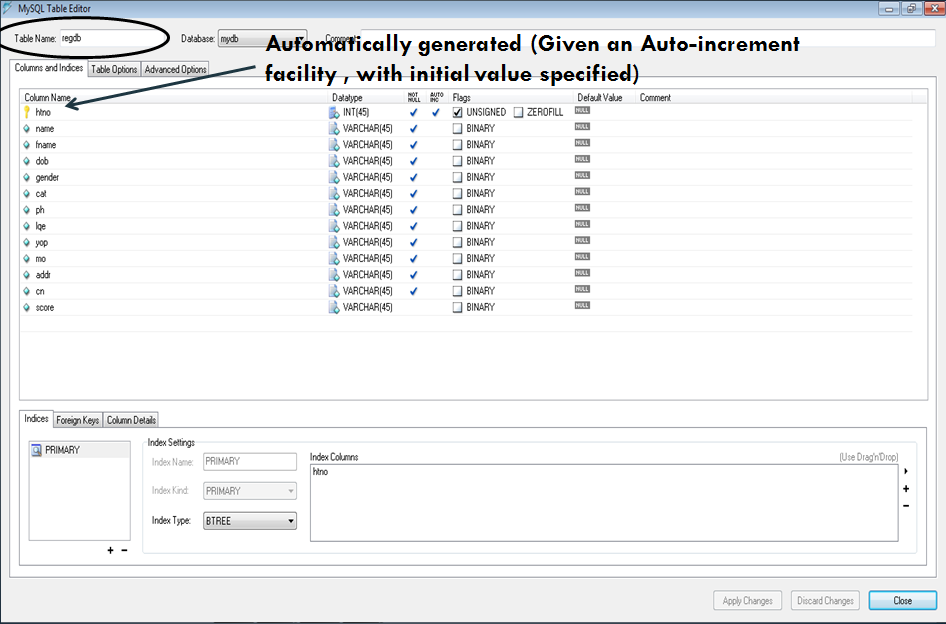


**Question Bank Builder**

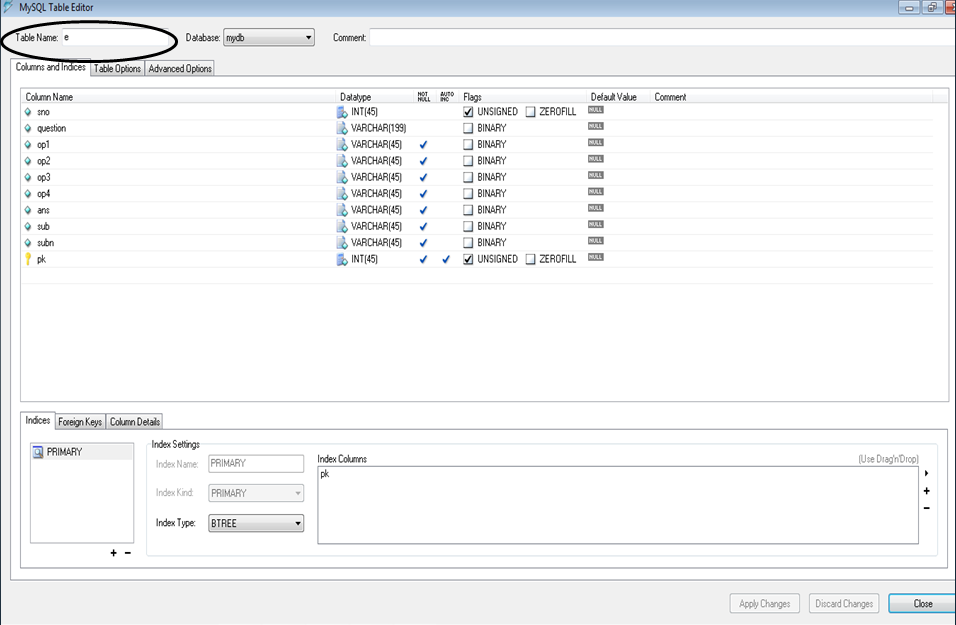


**Databases Used**

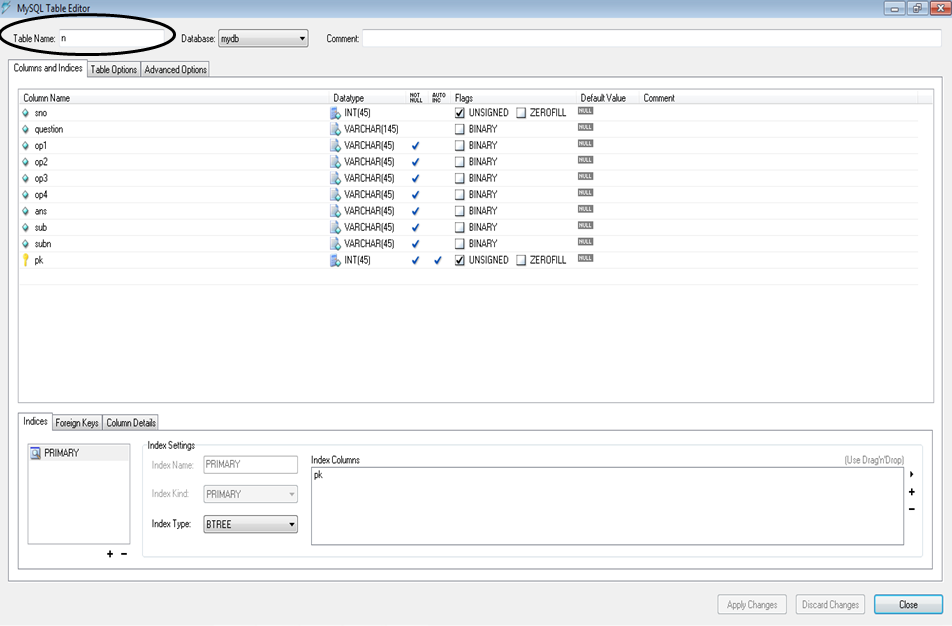
1. Registration Database – regdb



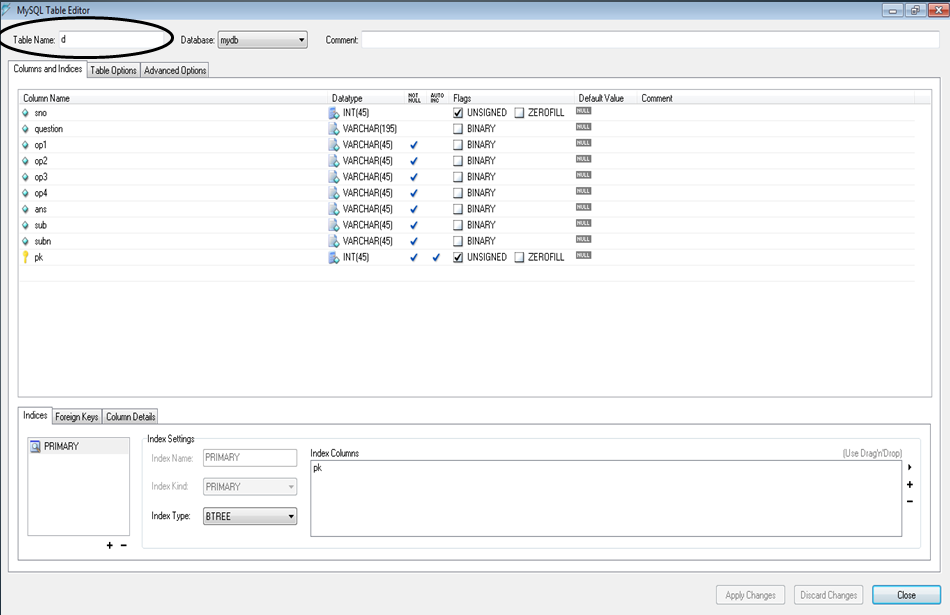
2.Questions Database - Easy



3.Questions Database – Medium



1. Questions Database – Difficult



**SCHEDULE OF ACTIVITIES/WEEKLY AGENDA OF THE PROJECT**

I took 6 weeks of training in Core JAVA from Techno Campus, Chandigarh under the guidance of Mr Bhushan Garg. These 6 weeks were exhaustively used to the best of time usage:

1. Four weeks of Teaching Sessions.

2. Two weeks of Project work.

I strictly followed an Iterative approach towards the development of the project-“User Adaptive examination System”.

Our basic ideology rested on the fact that we intended to develop a running project as soon as possible so as to allow a future scope of adding further features to the project.

I followed a three day iterative cycle i.e. after a cycle of three days i had a running project and every subsequent cycle ended up in a better featured model.

I based the development after setting certain weekly agenda’s like :

• First week-Designing of all the frame(for all the modules) having basic linked & running functionality.

• Second week- Constructing/Designing a reliable link between the java code and the database.

**REVIEW AND FUTURE SCOPE OF WORK**

This project, User Adaptive Examination System, is a complete software solution ,catering to all the requirements of various examination systems like Registration of candidates for the Exam(For Eligible Candidates), Question database building facility(For the Examiner), and Login and Exam taking facility(For the Registered Candidates).

* This project i.e. “User adaptive Examination System” is purely a new concept in the examination arena.
* It is one of its own kind
* Its functionality is not predefined , but based upon the users response during the test.
* It has real time decision making ability and real time program flow control features.
* It judges the candidates responses and thereafter provides a difficulty level to the candidate.

For further development of the project:

* Registration process can be set time – bounded, so that after a predefined period of time, the candidate cannot register for the examination.
* More subjects and further more questions can be added in the questions database.
* A print button can be added to the ‘Confirm Registration’ and ‘Result Card’ to generate a hardcopy

**LIMITATIONS OF THE PROJECT**

My project in itself is a self – sufficient and a reliable software, though it exhibit a distinctive dependency on a database.

Along with the numerous demarcated features it offer, like every other software project it has several limitations which I feel are not restricting productive usage of the software but are in the form of future improvements/scope. Every project, in order to increase it’s longevity in usage and sustainability in market requires new features to be added time and again so as to compete with the latest softwares. Similarly, our software innovatively puts forward the idea of ‘User Adaptability’ which may be enhanced in the further iterative attempts.

Being an open source project, I have left a scope of further enhancing ideas and features which could help the project to maintain its integrity.