**TITLE OF THE PROJECT**

ON-DEMAND

EXAMINATION

SYNOPSIS

**Introduction: -**

This is application based system, main objective of this system is On-Demand Examination is providing effective help and knowledge to fresher and Expert person in the field of Information & Technology to Identify him/her self and improve his/her quality in IT field as well as some other field. To achieve these things we develop an Application named as on-demand examination. Through this Application any person, who want to check his/her quality, he/she gives examination . Our Applications organize more than 100 subject First of all people interact with this Application and create a valid Login id and password. Any valid user have right to select the option for examination like C++, JAVA, etc. The candidate give examination at the minimum interval of 15 days. After complete the examination we provide answer list and certificate to the candidate. The candidate also post question to our question gallery and our experts will provide answer to them. The format of examination will be objective only. Answers are the hidden field on our Application.

**OBJECTIVE: -**The On- Demand Examination is design to provide help for finding quality of student. The main objective of this Application is as follows:

1. The student can register for more than one course at a time.
2. He/she give examination for all courses in a single day.
3. The maximum course selection at a time is three.
4. No any student gives examination before 15 days.
5. The Application support objective type question.
6. The student selects the appropriate answer of the question.
7. The expert checks the answer script.
8. After successful completion of examination the Application generate certificate on-line.

**PROJECT CATEGORY**

RDBMS stands for Relation Database Management System. The RDBMS is responsible for:

1. Maintaining the relationships between data in the database.
2. Ensuring that data is stored correctly- that the rules defining the relationships between data are not violated.
3. Recovering all data to a point of known consistency in the event of a system failure
4. RDBMS also provides referential integrity i.e. Primary Key-Foreign Key and Unique Key-Foreign Key.

SQL Server is designed to be a client server system. Client server systems are constructed so that the database can reside on a control computer known as a **Server** and be shared among several users.

SQL Server can work with thousands of client applications simultaneously.

**Transact-SQL:-**

SQL Server user Transact-SQL as its database query and programming language SQL is a set of commands that allows the user to specify the information to be retrieved or modified with Transact SQL. We can access data and query, update and manage relational database system.

**MODULE DISCRIPTION:-**

The On- Demand Examination have following modules.

1. Student Master
2. Teacher
3. Course Master
4. Question Gallery
5. Question Generator
6. Put Question
7. Answer Master
8. Result Master
9. **Student Master**:- The Student Master module is designed for the purpose of storing information of student. If student is registered then he/she have a valid Id else first of all he/she performs registration then gets Id.
10. **Teacher:**- The Teacher module is designed for the purpose of storing information of teacher who’s registered with our Application. After registration he/she has a valid Id then he/she is our expert.
11. **Course Master**:- Course Master Module is designed for the purpose of the storing information of course like C++, JAVA, .NET, etc.
12. **Question Gallery**:- The Question Generator module is designed for storing Question. The question is generated randomly. This modules have two sub modules, they are:
    1. **Question Generator:-** This module is design for the purpose of storing that type of question which is generated by students. The question is store into database and these database are accesses by teacher. The teacher provide solution and this solution is store into question answer\_Details file. Student access these answer.
    2. **Put Question:**- This module is design for the purpose of storing that type of question which is generated by teacher. This is the Hidden field of our Application. The question is generated randomly at the examination time.
13. Answer Master:- The Answer Master module is designed for the purpose of storing Answer script. This module responsible for tally the answer given by the student to a valid answer script and send to Result Master.
14. Result Master:- The Result Master checks this and generates a result. This result is store into database and also sends to student for valid E-mail Id.

**DATA BASE DESIGN**

1) Student Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | Attribute | Data type | Size |
| 1 | A\_ID | Int |  |
| 2 | St\_ID | Int |  |
| 3 | St\_Name | varchar | 50 |
| 4 | S\_Gnder | varchar | 50 |
| 5 | S\_FirstName | varchar | 50 |
| 6 | S\_MiddleName | varchar | 50 |
| 7 | S\_LastName | varchar | 50 |
| 8 | City | varchar | 50 |
| 9 | Pin | Bigint |  |
| 10 | Telephone No | Bigint |  |
| 11 | Mobil | Bigint |  |
| 12 | B\_ID | Int |  |
| 13 | Enrolment NO | Bigint |  |

2) Teacher Info Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatype | Size |
| 1 | A\_ID | Int |  |
| 2 | T\_ID | Int |  |
| 3 | T\_FirstName | varchar | 50 |
| 4 | T\_MiddleName | varchar | 50 |
| 5 | T\_LastName | varchar | 50 |
| 6 | T\_Address | varchar | 150 |
| 7 | City | varchar | 50 |
| 8 | State | varchar | 50 |
| 9 | Pin | bigint |  |
| 10 | Mobile | bigint |  |
| 11 | T\_D\_Of\_Join | date time |  |
| 12 | Salary | bigint |  |
| 13 | Phone | bigint |  |
| 14 | Y\_Of\_Service | varchar | 20 |
| 15 | Qualification | varchar | 20 |
| 16 | Designation | varchar | 50 |

3) Course Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatype | Size |
| 1 | A\_ID | Int |  |
| 2 | St\_ID | Int |  |
| 3 | Course Method No | Int |  |
| 4 | C\_ID | Int |  |
| 5 | Course Name | varchar | 50 |
| 6 | Course Fee | Int |  |
| 7 | Course time | datetime |  |
| 8 | Courses | varchar | 50 |

4) Question Generator Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatype | Size |
| 1 | A\_ID | Int |  |
| 2 | St\_ID | Int |  |
| 3 | C\_ID | Int |  |
| 4 | Created by | varchar | 50 |
| 5 | D\_Of\_Creation | date |  |
| 6 | Time\_Of\_generet | small Datetime |  |

5) Put Question Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatype | Size |
| 1 | A\_ID | Int |  |
| 2 | St\_ID | Int |  |
| 3 | C\_ID | Int |  |
| 4 | Put\_by | varchar | 50 |
| 5 | D\_Of\_Submit | Datetime |  |
| 6 | Time\_Of\_Submit | Datetime |  |

6) Answer Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatype | Size |
| 1 | A\_ID | Int |  |
| 2 | C\_ID | Int |  |
| 3 | AnsScript No | Int |  |
| 4 | AnsScript\_ID | Int |  |
| 5 | AnsScriptG\_by\_S | varchar | 500 |
| 6 | AnsScriptG\_by\_stu | varchar | 500 |
| 7 | Date\_Of\_Submit | Datetime |  |
| 8 | Time | Datetime |  |
| 9 | FullMarks | Int |  |
| 10 | M\_Obtained | Int |  |
| 11 | S\_ID | Int |  |

7) OPTION DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Data type | Size |
| 1 | A\_ID | Int |  |
| 2 | S\_ID | Int |  |
| 3 | Option1 | varchar | 50 |
| 4 | Option2 | varchar | 50 |
| 5 | Option3 | varchar | 50 |
| 6 | Option4 | varchar | 50 |
| 7 | Answer | varchar | 50 |
| 8 | Question\_ID | Int |  |
| 9 | Create\_date | datetime |  |

8) Result Master Details

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatye | Size |
| 1 | A\_ID | Int |  |
| 2 | T\_ID | Int |  |
| 3 | Checked by | varchar | 50 |
| 4 | Date\_Of\_Checks | Date |  |
| 5 | Attempt Quest | Int |  |
| 6 | Right | Int |  |
| 7 | Wrong | Int |  |
| 8 | Marks\_Obtaind | Int |  |
| 9 | Result | varchar | 20 |
| 10 | D\_Of\_Result | Date |  |
| 11 | S\_ID | Int |  |

9)**Question Answer\_Details**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Attribute | Datatype | Size |
| 1 | A\_ID | Int |  |
| 2 | S\_ID | Int |  |
| 3 | T\_ID | Int |  |
| 4 | Q\_ID | Int |  |
| 5 | Ans\_ID | Int |  |
| 6 | D\_of)Creation | Date |  |
| 7 | Time\_of\_Generation | Datetime |  |

**DATA FLOW DIAGRAM**: -It is a graphical, representation of flow of data through a system. In this diagram, the external entities provide input data for the processing. During the processing some intermediate data is generated. After final processing the final output data is generated.

Rules of making DFD:- There are following seven rules given construction of data flow diagrams.

1. Arrows should not cross each other.
2. Squares, circles and files must bear names.
3. Decomposed data flow must be balanced.
4. No two data flows, squares or circles can have the same name.
5. Draw all data flow around the outside of the diagram.
6. Choose meaningful names for data flows process and data stores.
7. Control information such as record counts passwords and validation requirements are not pertinent to a data-flow diagram.DFD are describe for different level

e.g.:-

1. 0 level
2. 1stlevel
3. 2nd level

The symbols used in DFD are as following:-

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Description |
|  | Data flow |  |
|  | Process |  |
|  | External entity |  |
| Or | Data store |  |

0-LEVEL DFD:-

Student Master

Teacher

Result Master

Answer Master

Course Master

Question Gallery

1st-LEVEL DFD:-

|  |
| --- |
| Student Details |

Put

|  |
| --- |
| Interact |

|  |
| --- |
| Register |

|  |
| --- |
| Access |

|  |
| --- |
| Put Question Detail |

|  |
| --- |
| Question generator details |

|  |
| --- |
| Option Master |

|  |
| --- |
| Put Question Details |

|  |
| --- |
| Answer Details |

|  |
| --- |
| Answer Details |

ss

Student

|  |
| --- |
| Course Details |

|  |
| --- |
| Register  For |

|  |
| --- |
| Store |

|  |
| --- |
| Put question |

|  |
| --- |
| put |

|  |
| --- |
| Access |

|  |
| --- |
| put |

|  |
| --- |
| Generate |

|  |
| --- |
| Give  Answer |

|  |
| --- |
| Access |

|  |
| --- |
| Retrieve |

|  |
| --- |
| access |

2nd –LEVEL DFD:-

Student

Teacher

1. Student Master

|  |
| --- |
| Student Details |

|  |
| --- |
| Retrieve |

Student

Student

Generate a Welcome Message “Hi! You are registered for Test.”

|  |
| --- |
| Course Details |

|  |
| --- |
| Student Details |

|  |
| --- |
| Login /id |

|  |
| --- |
| New |

|  |
| --- |
| check |

|  |
| --- |
| Inform to |

|  |
| --- |
| valid |

|  |
| --- |
| store |

|  |
| --- |
| inform |

|  |
| --- |
| invalid |

|  |
| --- |
| Retrieve |

2) Question Gallery

|  |
| --- |
| Question Generator Details |

Student

|  |
| --- |
| **interact** |

|  |
| --- |
| store |

|  |
| --- |
| retrieve |

Teacher

|  |
| --- |
| Interact |

|  |
| --- |
| Q\_A\_Details |

|  |
| --- |
| Put Question Details |

Teacher

|  |
| --- |
| Give to user |

3) Course Master

Staff

Student

|  |
| --- |
| Interact |

|  |
| --- |
| Student Details |

**ENTITY RELATIONSHIP - DIAGRAM**

An E R diagram is a model that identifies the concept or entities that exist in a system and the relationships between those entities. An ERD is often used as a way to visualize a relational database: each entity represents a database table and the relationship lines represents the key in one table that point to specific records in related tables.

Advantages of ER diagram

* Professional and faster Development.
* Productivity Improvement.
* Fewer Faults in Development.
* Maintenance becomes easy.

Generate

Question Gallery

Access

Put question

Answer Master

Course Master

Registered according to course

Student Master

Access

Result Master

Teacher

**PROCESS LOGIC:**

Process logic for Student Registration:-

Correct

Generate ID and Password and give to user as well as store into Database

Provide necessary details

Student Perform registration

no

yes

Process Logic For Teacher Module

After that Teacher is a our Expert

Generate a valid Teacher Id and Send to teacher as well as store into Database.

correct

Fill Registration form

|  |
| --- |
| yes |

|  |
| --- |
| no |

**TESTING**

Testing is the process to uncover the errors.

Testing objectives

1. Testing is the process of executing the process of executing the program to find error.
2. A group test has a high probability of finding the errors.
3. A successful test uncovers the all errors that have not been found.

Testing principles

1. The test should be according to the customers requirement.
2. There should be a planning for testing before it starts.
3. Testing should begin’ in the small’ and progress toward testing in the large.
4. To be most effective , testing should be conducted by independent third party.

Type of testing is:-

1. White box testing
2. Black box testing
3. White box testing: -It is also called glass box testing. It traces all the path of a program manually to find the error.

Advantage:-

* It guarantees that all independent paths have been checked at least once.
* It checks all logical decisions for true and false.
* Executes all loops at their boundary values.

Reasons for white box testing

* It can find logical errors that can not be found by black box.
* We often believe that a logical path is not likely to be executed when in fact , it may be executed on a regular basis.
* Typographical errors are random. The black box testing can find out typing error but typing errors are rare in the program.

1. Black box testing: -It is also called behavioral testing. The program is directly run by the computer to find the error.

It finds the following types of errors.

* Incorrect or missing function.
* Interface error.
* Performance error.
* Initialization and termination error.
* Error in data structures or database access.

**REPORT GENERATION:-**

This page is tentative , This page is change at the report submission time. The main view is to show the idea about Input / output.

**Registration page:-** This page is design for new user. At the time of registration user fill name, Address, userId, Password, Mobile no etc.

**Login Page**-This page is design for registered user. At the time for Login user fill Login Id and Password and submit it.

**Test Page**: After Login this page is active. User select Course code and submit it.

**Question Paper Page**:- This page Have many question. Question have multiple choice, User select any one.**Tools/ Platform, Hardware and Software Requirement Specification**

**Hardware**

1. Processor – I3 5th Generation
2. RAM - 4GB
3. Hard disk – 500GB
4. Keyboard
5. Mouse
6. Printer
7. Monitor – LED (16 inch)

**Software**

1. Operating System – Windows 10
2. Microsoft Visual Studio 2015
3. SQL Server 2012
4. Microsoft Office 2013

**FUTURE SCOPE**

This page is designed for the purpose of prescribing future scope of the system, in this system there are different point, they are:-

1. The system get examination for different type of course.
2. In future system conduct certain course like MCSC, CCNA,ete.

**REPORT**

**INTRODUCTION: -**

This is application based system, main objective of this system is On-Demand Examination is providing effective help and knowledge to fresher and Expert person in the field of Information & Technology to Identify him/her self and improve his/her quality in IT field as well as some other field. To achieve these things we develop an Application named as on-demand examination. Through this Application any person, who want to check his/her quality, he/she gives examination . Our Applications organize more than 100 subject First of all people interact with this Application and create a valid Login id and password. Any valid user have right to select the option for examination like C++, JAVA, etc. The candidate give examination at the minimum interval of 15 days. After complete the examination we provide answer list and certificate to the candidate. The candidate also post question to our question gallery and our experts will provide answer to them. The format of examination will be objective only. Answers are the hidden field on our Application.

**OBJECTIVE**

The On- Demand Examination is design to provide help for finding quality of student. The main objective of this Application is as follows:

1. The student can register for more than one course at a time.
2. He/she give examination for all courses in a single day.
3. The maximum course selection at a time is three.
4. No any student gives examination before 15 days.
5. The Application support objective type question.
6. The student selects the appropriate answer of the question.
7. The expert checks the answer script.
8. After successful completion of examination the Application generate certificate on-line.

**Tools/ Platform, Hardware and Software Requirement Specification**

**Hardware**

1. Processor – I3 5th Generation
2. RAM - 4GB
3. Hard disk – 500GB
4. Keyboard
5. Mouse
6. Printer
7. Monitor – LED (16 inch)

**Software**

1. Operating System – Windows 10
2. Microsoft Visual Studio 2015
3. SQL Server 2012
4. Microsoft Office 2013

**SYSTEM ANALYSIS**

Systems analysis the process of observing systems for troubleshooting or development purposes. It is applied to information technology, where computer-based systems require defined analysis according to their makeup and design.

systems analysis can include looking at end-user implementation of a software package or product; looking in-depth at source code to define the methodologies used in building software; or taking feasibility studies and other types of research to support the use and production of a software product, among other things.

Software Development Life Cycle

**Software Development Life Cycle (SDLC)** aims to produce a high-quality system that meets or exceeds customer expectations, works effectively and efficiently in the current and planned information technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

SDLC consists of mainly seven steps. These are:

1. Project Identification and Selection
2. Project Initiation and Planning
3. Analysis
4. Logical Design
5. Physical Design
6. Implementation
7. Testing.
8. **Project Identification and Selection**In this phase, the user identifies the need for a new or improved system. Information requirements of the organization as a whole are examined, and projects to meet these requirements are proactively identified. the project identification and selection process is a determination of which systems development projects should be undertaken by the organization at least in terms of an initial study.
9. **Project Initiation and Planning**In second phase the problems that are identified should be investigated and a decision to implement the information system or not for the organization should be taken. A critical step at this point is determining the scope of the proposed system.
10. **Analysis**   
    Analysis is the next phase. During this phase, the analysis has several sub-phases. The first is requirements determination. In this sub-phase, analysts work with users to determine the expectations of users from the proposed system. This sub-phase usually involves a careful study of current systems, manual or computerized that might be replaced or enhanced as part of this project. Next, the requirements are studied and structured in accordance with their inter-relationships and eliminate any redundancies. Third, alternative initial design is generated to match the requirements.
11. **Logical Design**  
    Logical design is not tied to any specific hardware and systems software platform. Theoretically, the system could be implemented on any hardware and systems software. The idea is to make sure that the system functions as intended. Logical design concentrates on the business aspects of the system.
12. **Physical Design**   
    In physical design, the logical design is turned into physical or technical specifications. For example, you must convert diagrams that map the origin, flow, and processing of data in a system into a structured systems design that can then be broken down into smaller and smaller units known as modules for conversion to instruction written in a programming language.
13. **Implementation**   
    During implementation, you turn system specification into working system that is tested and put into use. Implementation includes coding, testing and installation.
14. **Testing**   
    When the software is ready, it is sent to the testing department where Test team tests it thoroughly for different defects. They either test the software manually or using automated testing tools depends on process defined in [STLC (Software Testing Life Cycle)](https://www.softwaretestingmaterial.com/stlc-software-testing-life-cycle/) and ensure that each and every component of the software works fine. Once the QA makes sure that the software is error-free, it goes to the next stage, which is Implementation. The outcome of this phase is the Quality Product and the [Testing Artifacts](https://www.softwaretestingmaterial.com/test-deliverables/).

**FEASIBILITY STUDY**

Feasibility study consists of activities which determine the existence of scope of developing an information system to the organization. Feasibility study starts from the preliminary investigation phase. At this stage, the analyst estimates the urgency of the project and estimates the development cost. The next check point is problem analysis. At this stage, the analyst studies current system. S/he does it to understand the problem in the better way. It helps him/her to make better estimates of development cost, and also to find out the benefits to be obtained from the new system.

There are some feasibility study

1. Technical feasibility
2. Operational feasibility
3. Economic feasibility
4. Legal Feasibility.
5. **Technical feasibility**:-Technical feasibility is concerned with the availability of hardware and software required for the development of the system, to see compatibility and maturity of the technology proposed to be used and to see the availability of the required technical manpower to develop the system. At this stage, the analyst has to see or identify the proposed technology, its maturity, its ability or scope of solving the problem. If the technology is mature, if it has large customer base, it will be preferable to use as large customer base already exists and problems that stem from its usage may be less when compared to other technologies which don’t have a significant customer base. we have to ensure that the required technology is practical, and available.
6. **Operational feasibility:-** This involves undertaking a study to analyze and determine whether your business needs can be fulfilled by using the proposed solution. It also measures how well the proposed system solves problems and takes advantage of the opportunities identified during scope definition. Operational feasibility studies also analyze how the project plan satisfies the requirements identified in the requirements analysis phase of system development. To ensure success, desired operational outcomes must inform and guide design and development. These include such design-dependent parameters such as reliability, maintainability, usability, sustainability, affordability, and others
7. **Economic feasibility:**- It is the measure of cost effectiveness of the project. The economic feasibility is nothing but judging whether the possible benefit of solving the problems is worthwhile or not. At the feasibility study level, it is impossible to estimate the cost because customer’s requirements and alternative solutions have not been identified at this stage. However, when the specific requirements and solutions have been identified, the analyst weighs the cost and benefits of all solutions, this is called “cost benefit analysis”. This is discussed below. A project which is expensive when compared to the savings that can be made from its usage, then this project may be treated as economically infeasible.
8. **Legal Feasibility:-** Legal feasibility studies issues arising out of the need to the development of the system. The possible consideration might include copyright law, labour law, antitrust legislation, foreign trade, regulation, etc. Contractual obligation may include the number of users who will be able to use the software. There may be multiple user’s licences, single user licences, etc..

**PRELIMINARY INVESTIGATION**

The basic purpose behind Preliminary Investigation is to first clarify, understand and evaluate the Project Request**.** Preliminary Investigation basically refers to the collection of information that guides the management of an organization to evaluate the merits and demerits of the project request and make an informed judgment about the feasibility of the proposed system.

This sort of investigation provides us with a through picture of the kind of software and hardware requirements which are most feasible for the system, plus the environment in which the entire project has to be installed and made operational.

**1) Reviewing the Documents provided by the Organization**

They were quite effective in guiding us towards visualizing the features that were needed to be put together in the system and the required output which had to be generated once the system became functional. These specifications provided to us by the organization showed how the new system should look like; it helped us in understanding the basic structure of the application which we were supposed to develop.

**2) On site Observation:**

Another technique utilized by us to gain information about the project was to visit the client site where the system had to be installed. Here a detailed system study was carried out, checking the existing system to replicate it with our system. We also observed the activities of the system directly. During the on-site observation, we saw the office environment, work load of the system and users, method of work, and the facilities provided by the organization. This information helped us to understand how the system should operate. But after interviewing the persons, who is affected by the system, we got more details that further explain the project and shown whether assistance is merited economically, operationally and technically.

**3) Conducting Interviews:**

This method of investigation conducted by us involved questioning the concerned personnel to get the user’s (client) view about the system and the features they desired it to have.

Some of the Questions put forward by our team were:

* The amount of data needed to be stored.
* The number of customers using the system and number of which the application needed to be installed.
* The issue of our application with existing system was widely discussed.
* The level of access given to the customer would depend on his department.

**SCHEDULING**

**Gantt Charts**

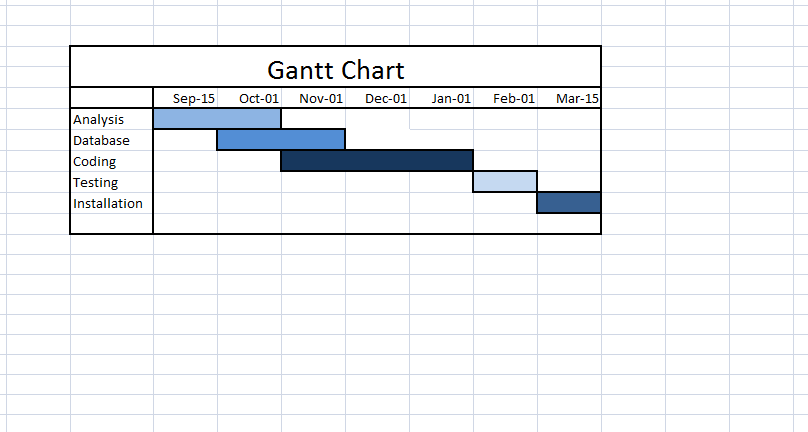
A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

This allows you to see at a glance:

* What the various activities are
* When each activity begins and ends
* How long each activity is scheduled to last
* Where activities overlap with other activities, and by how much
* The start and end date of the whole project

When you set up a Gantt chart, you need to think through all of the tasks involved in your project. As part of this process, you'll work out who will be responsible for each task, how long each task will take, and what problems your team may encounter.

This detailed thinking helps you ensure that the schedule is workable, that the right people are assigned to each task, and that you have workarounds for potential problems before you start.



**PERT Chart**

A PERT chart is a project management tool used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique.

A PERT chart presents a graphic illustration of a project as a network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labeled vectors (directional lines) representing tasks in the project. The direction of the arrows on the lines indicates the sequence of tasks. The tasks between two nodes. And if theyare not dependent on the completion of one to start the other and can be undertaken simultaneously.These tasks are called parallel or concurrent tasks.

**PERT Chart**

Days 9-12

Days 6-9

Days 1-6

Days 12-16

Days 16-23

Days 23-48

Days 48-54

Days 54-56

**Software Specification Requirement**

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform.

The purpose of the SRS is to:

* Establish the basis for agreement between the customers and the suppliers on what the software product is to do. The complete description of the functions to be performed by the software specified in the SRS will assist the potential user to determine if the software specified meets their needs or how the software must be modified to meet their needs.
* Provide a basis for developing the software design. The SRS is the most important document of reference in developing a design.
* Reduce the development effort. The preparation of the SRS forces the various concerned groups in the customer's organisation to thoroughly consider all of the requirements before design work begins. A complete and correct SRS reduces effort wasted on redesign, recoding and retesting. Careful review of the requirements in the SRS can reveal omissions, misunderstandings and inconsistencies early in the development cycle when these problems are easier to correct
* Provide a basis for estimating costs and schedules. The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates
* Provide a baseline for validation and verification. Organisations can develop their test documentation much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured
* Facilitate transfer. The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organisation and suppliers find it easier to transfer it to new customers
* Serve as a basis for enhancement. Because the SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued product evaluation.

**SRS of On-Demand Examination**

**Introduction: -**

This is web based application system, main objective of this system is On-Demand Examination is providing effective help and knowledge to fresher and Expert person in the field of Information & Technology to Identify him/her self and improve his/her quality in IT field as well as some other field. To achieve these things we develop an Application named as on-demand examination. Through this Application any person, who want to check his/her quality, he/she gives examination . Our Applications organize more than 100 subject First of all people interact with this Application and create a valid Login id and password. Any valid user have right to select the option for examination like C++, JAVA, and other courses. The candidate give examination at the minimum interval of 15 days. After complete the examination we provide answer list and certificate to the candidate. The format of examination will be objective only. Answers are the hidden field on our Application.

This website reduces the manual work, maintaining accuracy, increasing efficiency and saving time. Also institutes need not go to develop new software each time, instead they just register and conduct a Exam. For students, it saves time of going to far away centers and also they can view their result then and there.

**Information Description:**

This is web based application system, main objective of this system is On-Demand Examination is providing effective help and knowledge to fresher and Expert person in the field of Information & Technology to Identify him/her self and improve his/her quality in IT field as well as some other field. To achieve these things we develop an Application named as on-demand examination. Through this Application any person, who want to check his/her quality, he/she gives examination . Our Applications organize more than 100 subject First of all people interact with this Application and create a valid Login id and password.

Student first perform Registration, where student fill their information and create a login ID and password. Once student is register he/she can log-in to the website as user. After login user’s home page will open and student has option to apply exam where student can apply up to 3 subject at once. When admin approve the student ‘s requested subject then the exam link will appear in the exam section. When student click on this link the exam will start. After exam result is submitted to the admin and when admin approve the result , result will display in “Result” section of student page. Student can also download the certificate of there result from “Certificate” section.

Student has option to see their profile and edit profile by clicking on “My Profile ”, where student can update their phone no. Email, address, etc.

Admin will login to the website by same login page by “teacherID “ and “password”. The admin home page open when admin logged in. The admin has different link for performing different task. The first link is “MANAGE PROGRAM” where teacher add the program to the database, which helps when student perform registration and select exam, and also needed when teacher/admin add the questions.

The second link is “MANAGE COURSE” to add the course of different program to the database. This will also help student at registration time and applying exam time, and teacher when teacher/admin add the question.

The next link is “MANAGE STATE” and “MANAGE CITY” to add state and their city to the database. This will used when student perform registration and when new teacher is added.

The next link is “MANAGE QUESTION” to add the question to the database, admin/teacher first select the program and their course and then enter the question in the question box and enter the four option in option box and select correct option in drop down list. At this page there is a link “see question” to see the question, after clicking this page new page open where program and course have to select and then all question of that course/subject will displayed.

The next link is “Student Exam Request”. Here all the request is displayed if the requested exam’s question is prepared then admin allow the course by changing 0 to 1. Else admin first prepare or add the question in question and then allow.The “student list ” link is to see all student details.

The next link is “Student Result”. This page display all the result of student and admin allow student to see their result.

The next link is “Exam Setting” to set the exam time and number of question.

The next link is “ADD TEACHER” to add more admin/ teacher of different subject to manage the all activity of this website.

**Module description**

1. Student Master
2. Teacher
3. Course Master
4. Question Gallery
   1. Question Generator
   2. Put Question
5. Answer Master
6. Result Master
7. **Student Master**:- The Student Master module is designed for the purpose of storing information of student. If student is registered then he/she have a valid Id else first of all he/she performs registration then gets Id.
8. **Teacher:**- The Teacher module is designed for the purpose of storing information of teacher who’s registered with our Application. After registration he/she has a valid Id then he/she is our expert.
9. **Course Master**:- Course Master Module is designed for the purpose of the storing information of course like C++, JAVA, .NET, etc.
10. **Question Gallery**:- The Question Generator module is designed for storing Question. The question is generated randomly. This modules have two sub modules, they are:
    1. **Question Generator:-** This module is design for the purpose of storing that type of question which is generated by students. The question is store into database and these database are accesses by teacher. The teacher provide solution and this solution is store into question answer\_Details file. Student access these answer.
    2. **Put Question:**- This module is design for the purpose of storing that type of question which is generated by teacher. This is the Hidden field of our Application. The question is generated randomly at the examination time.
11. **Answer Master:-** The Answer Master module is designed for the purpose of storing Answer script. This module responsible for tally the answer given by the student to a valid answer script and send to Result Master.
12. **Result Master:-** The Result Master checks this and generates a result. This result is store into database and also sends to student for valid E-mail Id.

**Servey of Technology**

**ASP.Net**

ASP.Net is a web development platform provided by Microsoft. It is used for creating web-based applications. ASP.Net was first released in the year 2002.The first version of ASP.Net deployed was 1.0. The most recent version of ASP.Net is version 4.6. ASP.Net is designed to work with the HTTP protocol. This is the standard protocol used across all web applications.ASP.Net applications can also be written in a variety of .Net languages. These include C#, VB.Net, and J#. In this chapter, you will see some basic fundamental of the .Net framework.The full form of ASP is Active Server Pages, and .NET is Network Enabled Technologies.

**Features of .Net Technology:-**

* ASP .NET significantly reduces the amount of code required for building large and complex applications which can increase overall development speed and reduce development costs.
* Just-in-time compilation, smart caching technologies and native optimization dramatically increase overall application performance.
* Framework supports varied languages, it allows you to select the language that better applies to your application.
* It provides ability of cross platform migration.
* It provides simplicity making it easy to perform common tasks including configuration and deployment.
* The features vast class library enclosing a large number of common functions and ready-to-use custom web-controls that allow creating professional applications with no need to develop from scratch.
* It ensured high reliability and security due to built-in Windows authentication and per-application configuration.
* It is regularly updated by Microsoft to meet the most up-to-date technology requirements.

**SQL**

Server is one of the most popular RDBMS of today.Microsoft makes SQL Server available in multiple editions, with different feature sets and targeting different users. These editions are: SQL Server Compact Edition (SQL CE) The compact edition is an embedded database engine. Unlike the other editions of SQL Server, the SQL CE engine is based on SQL Mobile (initially designed for use with hand-held devices) and does not share the same binaries. Due to its small size (1 MB DLL footprint), it has a markedly reduced feature set compared to the other editions. For example, it supports a subset of the standard data types, does not support stored procedures or Views or multiple-statement batches (among other limitations). It is limited to 4 GB maximum database size and cannot be run as a Windows service, Compact Edition must be hosted by the application using it. The 3.5 version includes Considerable work that supports ADO.NET Synchronization Services.

**FEATURES OF SQL 2012 **

* **Security Management:** SQL Server provides a controlled access to data to users by providing a combination of privileges.  Backup and Recovery: SQL Server provided sophisticated security backup and recovery routines. ****
* **Open connectivity:** SQL Server provides open connectivity to and from other vendor’s software such as Microsoft. Also SQL Server database can be access by various front-end software’s such as Microsoft Visual Basic ®, Power Builder etc. 
* **Space Management**: In SQL Server once can flexibly allocate disk spaces for data storage and can control them subsequently. SQL Server 5 is designed with special feature of data warehousing

**Functional Description**

The On- Demand Examination have following modules.

1. Student
2. Teacher
3. Course Master
4. Question Gallery

A. Question Generator

B. Put Question

5. Answer Master

6. Result Master

**1. Student Master**:- The Student Master module is designed for the purpose of storing information of student. If student is registered then he/she have a valid Id else first of all he/she performs registration then gets Id.

**2. Teacher:**- The Teacher module is designed for the purpose of storing information of teacher who’s registered with our Application. After registration he/she has a valid Id then he/she is our expert.

**3. Course Master**:- Course Master Module is designed for the purpose of the storing information of course like C++, JAVA, .NET, etc.

**4. Question Gallery**:- The Question Generator module is designed for storing Question. The question is generated randomly. This modules have two sub modules, they are:

* 1. **Question Generator:-** This module is design for the purpose of storing that type of question which is generated by students. The question is store into database and these database are accesses by teacher. The teacher provide solution and this solution is store into question answer\_Details file. Student access these answer.
  2. **Put Question:**- This module is design for the purpose of storing that type of question which is generated by teacher. This is the Hidden field of our Application. The question is generated randomly at the examination time.

**5. Answer Master:-** The Answer Master module is designed for the purpose of storing Answer script. This module responsible for tally the answer given by the student to a valid answer script and send to Result Master.

**6. Result Master:-** The Result Master checks this and generates a result. This result is store into database and also sends to student for valid E-mail Id.

**DATABASE DESIGN**

**1. CityDetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | CityID | bigint |  |
| 2 | StateID | varchar | 50 |
| 3 | CityName | varchar | 50 |
| 4 | PinCode | bigint |  |
| 5 | Ts\_Created | datetime |  |
| 6 | Ts\_update | datetime |  |

**2. CourseDeatils**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | CourseID | bigint |  |
| 2 | ProgramName | varchar | 50 |
| 3 | CourseName | varchar | 50 |
| 4 | Ts\_Created | datetime |  |
| 5 | Ts\_update | datetime |  |

**3. ExamRequestDetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | RequestID | bigint |  |
| 2 | UserName | varchar | 50 |
| 3 | CourseName1 | varchar | 50 |
| 4 | CourseName2 | varchar | 50 |
| 5 | CourseName3 | varchar | 50 |
| 6 | ProgramName | varchar | 50 |
| 7 | flag1 | int |  |
| 8 | flag2 | int |  |
| 9 | flag3 | int |  |
| 10 | Ts\_Created | datetime |  |
| 11 | Ts\_update | datetime |  |

1. **ExamSetting**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | ExamId | int |  |
| 2 | TimeLimit | int |  |
| 3 | NoOfQuestion | int |  |
| 4 | Ts\_Created | datetime |  |
| 5 | Ts\_update | datetime |  |

**5. programDetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | ProgramID | bigint |  |
| 2 | ProgramName | varchar | 50 |
| 3 | Duration | varchar | 50 |
| 4 | Ts\_Created | datetime |  |
| 5 | Ts\_update | datetime |  |

1. **QuestionDetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | QuestionID | bigint |  |
| 2 | Program | varchar | 50 |
| 3 | Course | varchar | 50 |
| 4 | Question | varchar | 500 |
| 5 | OptionA | varchar | 150 |
| 6 | OptionB | varchar | 150 |
| 7 | OptionC | varchar | 150 |
| 8 | OptionD | varchar | 150 |
| 9 | CorrectOption | varchar | 10 |
| 10 | Ts\_Created | datetime |  |
| 11 | Ts\_update | datetime |  |

1. **ResultDetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | ResultID | int |  |
| 2 | UserName | varchar | 50 |
| 3 | CourseName | varchar | 50 |
| 4 | ExamDate | datetime |  |
| 5 | NoOfQ | int |  |
| 6 | Marks | int |  |
| 7 | Status | varchar | 50 |
| 8 | flag | int |  |

1. **StateDetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | StateID | bigint |  |
| 2 | StateName | varchar | 50 |
| 3 | Ts\_Created | datetime |  |
| 4 | Ts\_update | datetime |  |

1. **UserDetials**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | UserID | bigint | 50 |
| 2 | FirstName | varchar | 50 |
| 3 | LastName | varchar | 50 |
| 4 | LoginID | varchar | 50 |
| 5 | Password | varchar | 50 |
| 6 | Email | varchar |  |
| 7 | Phone | bigint |  |
| 8 | Gender | varchar | 50 |
| 9 | DOB | varchar | 50 |
| 10 | Program | varchar | 50 |
| 11 | Address | varchar | 50 |
| 12 | StateID | varchar | 50 |
| 13 | cityID | varchar | 50 |
| 14 | PinCoad | bigint |  |
| 15 | Ts\_Created | datetime |  |
| 16 | ts\_update | datetime |  |
| 17 | role | int |  |

10. TeacherDetails

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Attribute** | **Data Type** | **Size** |
| 1 | TeacherID | bigint | 50 |
| 2 | FirstName | varchar | 50 |
| 3 | LastName | varchar | 50 |
| 4 | LoginID | varchar | 50 |
| 5 | Password | varchar | 50 |
| 6 | Email | varchar |  |
| 7 | Phone | bigint |  |
| 8 | Gender | varchar | 50 |
| 9 | DOB | varchar | 50 |
| 10 | Program | varchar | 50 |
| 11 | Address | varchar | 50 |
| 12 | StateID | varchar | 50 |
| 13 | cityID | varchar | 50 |
| 14 | PinCoad | bigint |  |
| 15 | Ts\_Created | datetime |  |
| 16 | ts\_update | datetime |  |
| 17 | role | int |  |

**DFD**(DATA FLOW DIAGRAM)

It is a graphical, representation of flow of data through a system. In this diagram, the external entities provide input data for the processing. During the processing some intermediate data is generated. After final processing the final output data is generated.

Rules of making DFD:- There are following seven rules given construction of data flow diagrams.

1. Arrows should not cross each other.
2. Squares, circles and files must bear names.
3. Decomposed data flow must be balanced.
4. No two data flows, squares or circles can have the same name.
5. Draw all data flow around the outside of the diagram.
6. Choose meaningful names for data flows process and data stores.
7. Control information such as record counts passwords and validation requirements are not pertinent to a data-flow diagram.DFD are describe for different level

e.g.:-

1. 0 level
2. 1stlevel
3. 2nd level

The symbols used in DFD are as following:-

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Description |
|  | Data flow |  |
|  | Process |  |
|  | External entity |  |
| Or | Data store |  |

0-LEVEL DFD:-

Student Master

Teacher

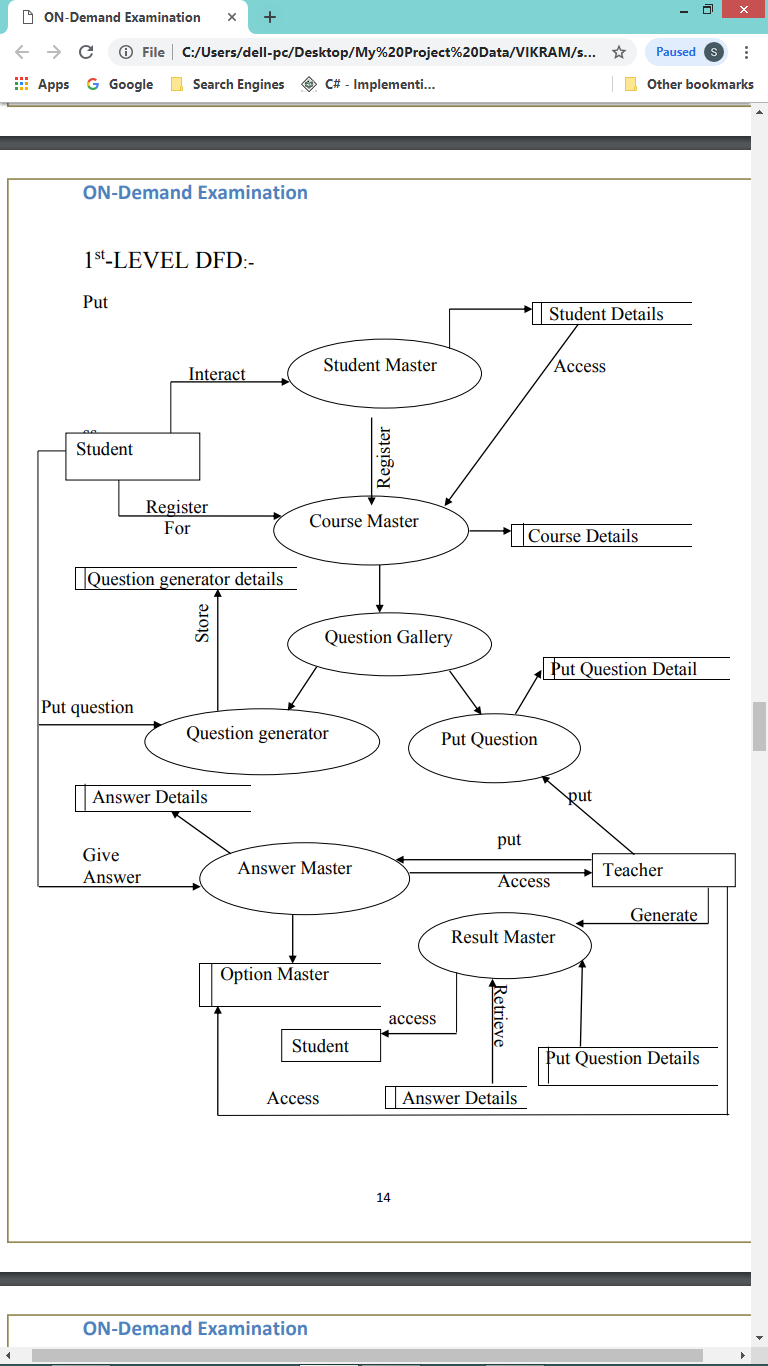
Result Master

Answer Master

Course Master

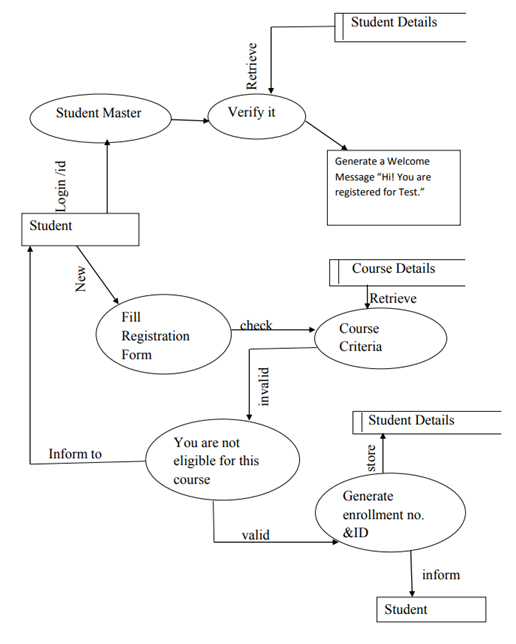
Question Gallery

1st-LEVEL DFD:-



2nd –LEVEL DFD:-

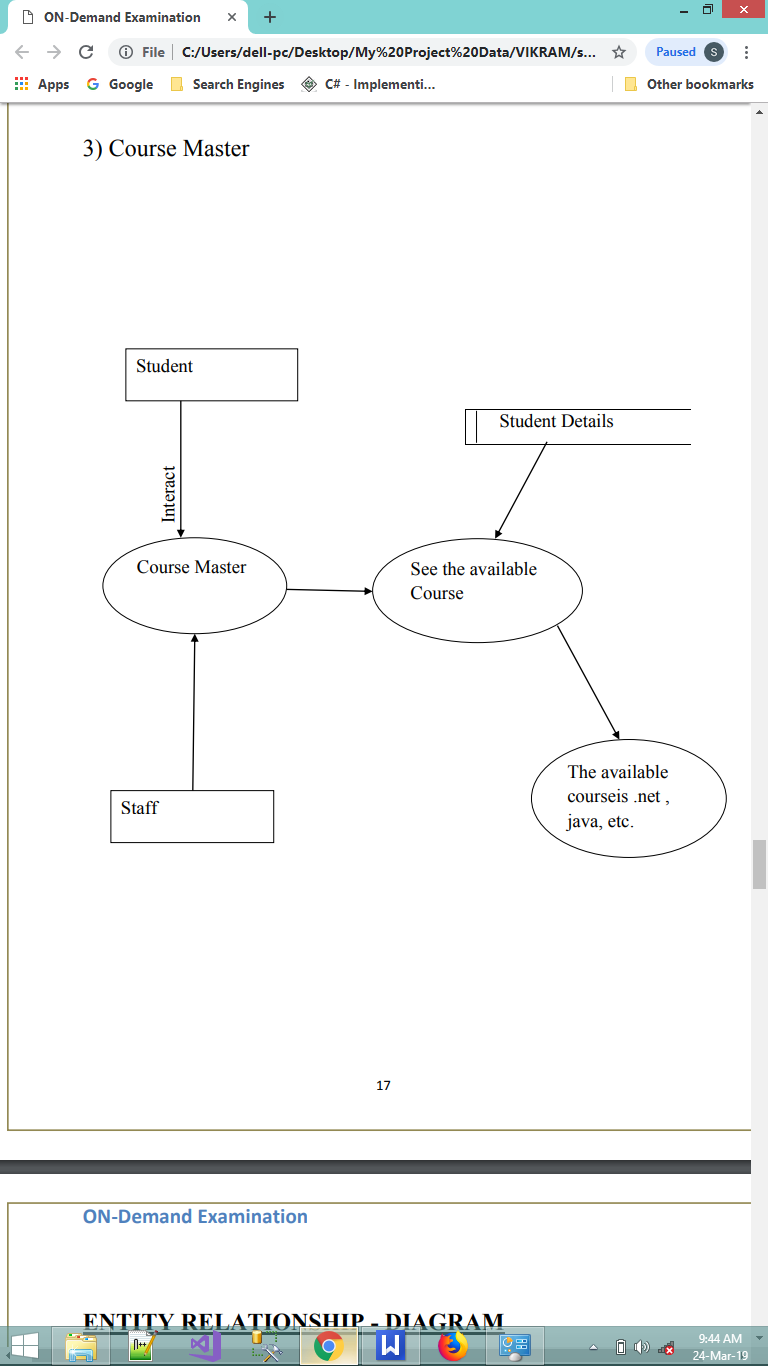
1) Student Master



2) Question Gallery



3) Course Master



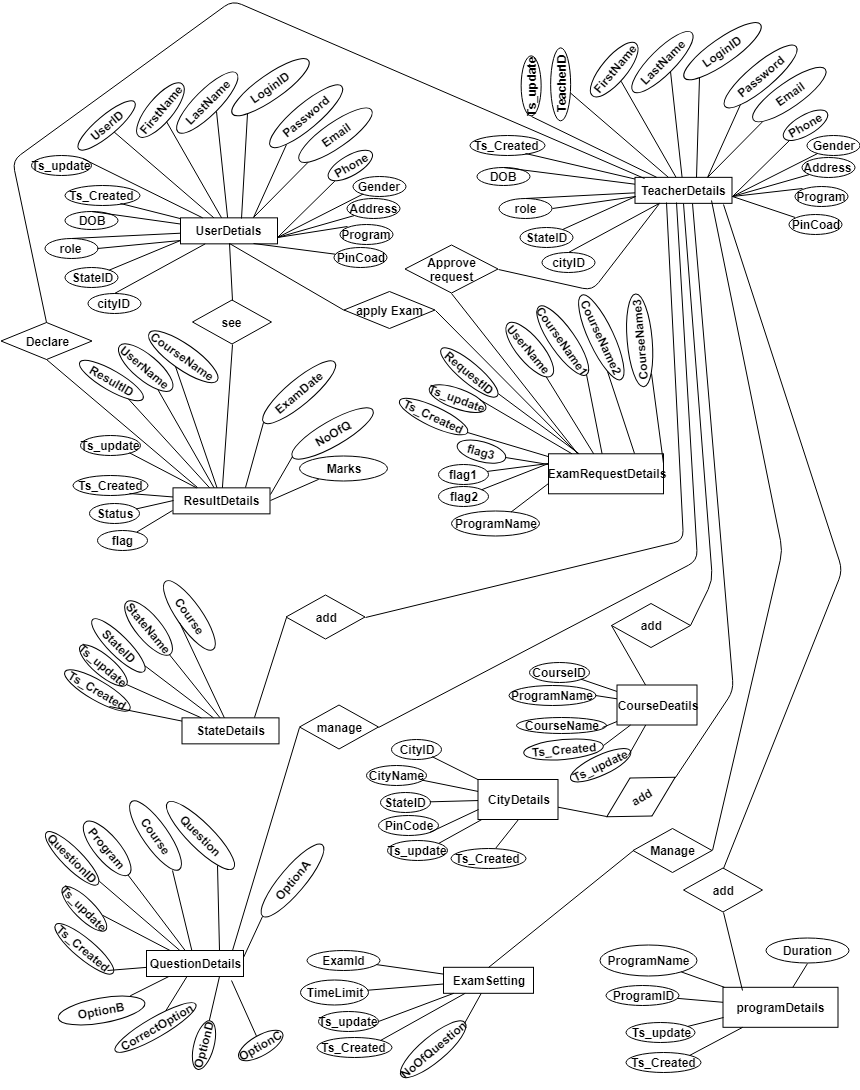
**ENTITY RELATIONSHIP - DIAGRAM**

An E R diagram is a model that identifies the concept or entities that exist in a system and the relationships between those entities. An ERD is often used as a way to visualize a relational database: each entity represents a database table and the relationship lines represents the key in one table that point to specific records in related tables.

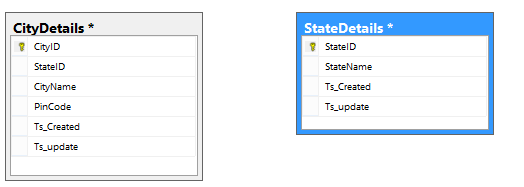
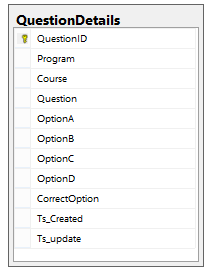
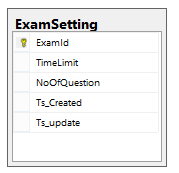
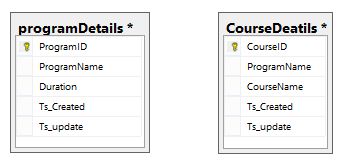
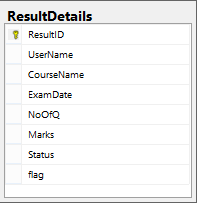
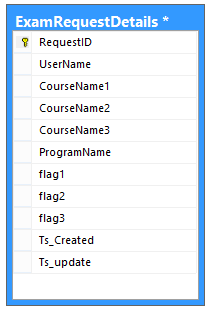
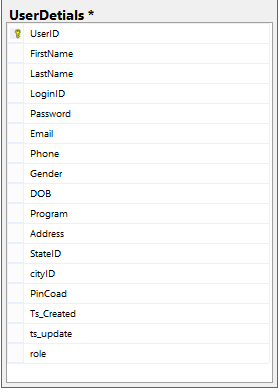
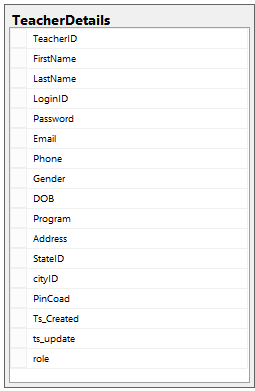
Advantages of ER diagram

* Professional and faster Development.
* Productivity Improvement.
* Fewer Faults in Development.
* Maintenance become easy.

**ERD**

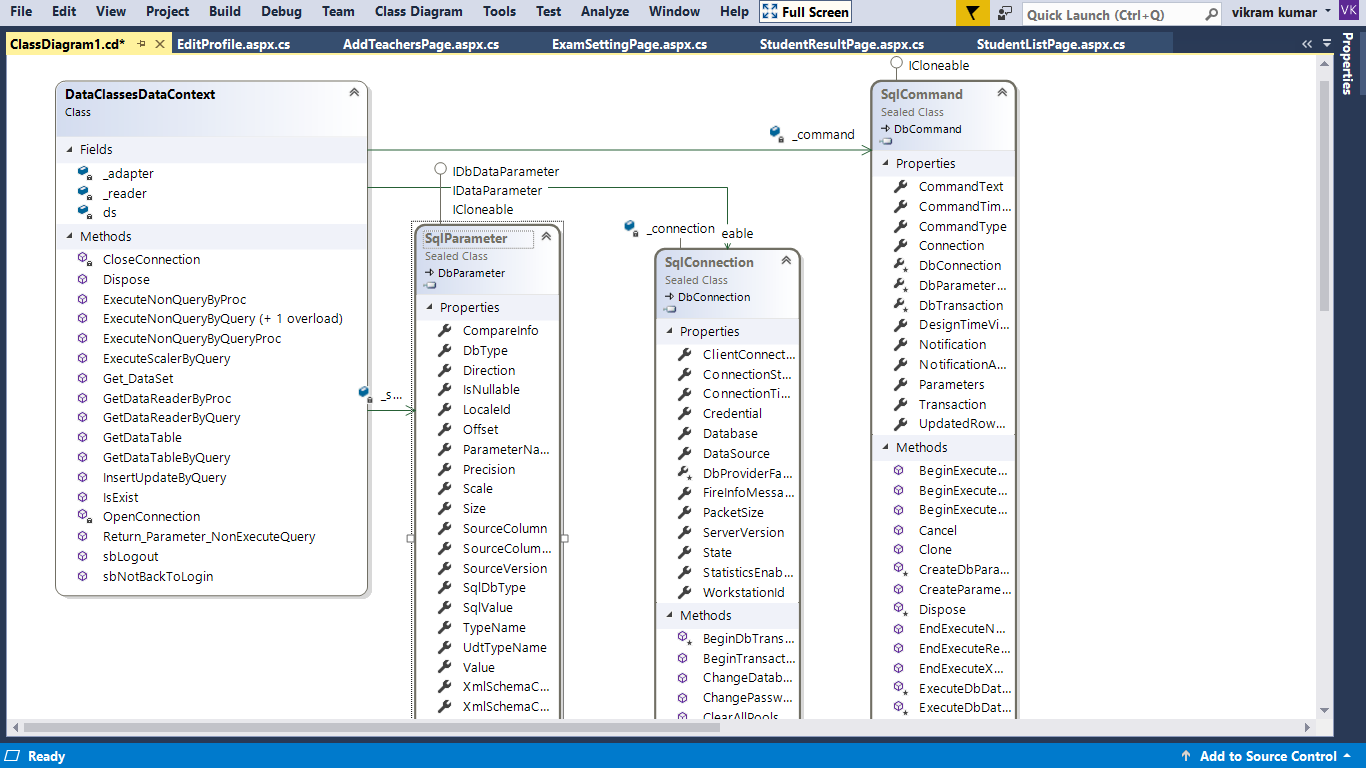


**DATABASE DIAGRAM**



**CLASS DIAGRAM**

**Modularization**Modularity refers to the extent to which a software/web application may be divided into smaller modules software Modularity indicates that the no. Of application modules are capable of serving a specified business domain.



Modularity is successful because developers use pre-written code which save resources overall. Modularity provides great software development manageability.

**Goals of Design**

1. The design of the system should be module based. It means there are modules which together make up the system and the organization of these modules is hierarchical.
2. Each module controls the functions of a suitable number of subordinate modules at the next hierarchical level.
3. One of the important features of good design is that the modules, which make up the system don’t communicate intensively. The communication should be kept at minimum level. The reason for this imposition is that modules should be independent of each other to the maximum extent possible. Independence means, “one module’s functionality should not be dependent on the internal functions of other module”.
4. The size of module should be appropriate as required for the features it should possess like being relatively independent of other modules etc. Basically, no specific size of range of size can be defined on modules though it is done occasionally. The size varies from module to module and from project to project.  
   5. A module should not be assigned the duty of performing more than one function.

6. The coding of modules should be generic. It enables the system to use the module as frequently as possible.

**Guidelines for good design :-**

* A system should be divided into as many relatively independent modules as possible. This is known as factoring.
* A superordinate module should control not more than seven subordinate modules. Of course, this guideline is not strict and varies from system to system.
* The dependency levels between modules should be minimum. This automatically leads to the design of modules, which don’t communicate, frequently with each other. Also, the communication between modules should be through parameters. Of course, Boolean variables or flags can be used for the purpose of communication. This is called coupling.
* Usually, a module if of not more than 100 lines. It may be a minimum of 50 lines. But, these sizes are not to be strictly followed and they may vary from system to system. It is notable here that lesser the lines of code, easier to read.
* A module should not perform more than one function. There should be no line in the code of the module, which is concerned with a function that is not the objective of that particular module. One easy check for this conformance is that the module’s function should be describable easily in a few words. This is called cohesion.

Modules at the lower level of the design are called by more than one superordinate module. It means that multiple superordinate modules use most of the modules at the lower level.

**Coupling:-** it is the interaction between module, it is the measure of interconnection between modules in a software for a good software the coupling should be minimum.

**Types of coupling:-**

1. **Data coupling:-** simple data is passed through one module to another, the data is in form of a list, it is the lowest level of coupling.
2. **Stemp coupling:-** a portion of data structure is passed for example object is passed in object oriented technology.
3. **Control coupling:**- control flag is passed for example passing the true or false value.
4. **External coupling:-** when module interact with the environment externally to the software, for example the software interact with the telecommunication facility.
5. **Common coupling:-** when the module is interact with the globe variable/memory.
6. **Content coupling:-** when the module interact with the data of the other module, for example inheritance.
7. **Type use coupling:-** when one component use a data type defined in another component, for example a class declare the object of another class.
8. **Inclusion or import coupling:**- when one component import a package of other component.

**Cohesion:-** it is defined as the inner strength of a module. It is the extension of information hiding concept. A cohesive module perform a single task.

**Types of cohesion:-**

1. **Coincidental cohesion:-** when one module perform more than one task that are not related, then this type of cohesion is known as coincidental cohesion
2. **Logical cohesion:-** a module perform logically related task.
3. **Procedural cohesion:**- when task of a module are related and must be executed in a specific order.
4. **Communication cohesion:-** when all the task of a module use the same data, that is concentrate on one area of a data structured.
5. **Temporal cohesion:-** when all the related task must be executed in the same time period for example operation performed at the start of the computer.
6. **Functional cohesion:-** when a module perform one and only one computation and then returns the result.
7. **Sequential cohesion:-** components or operation are grouped in a manner that allows the first to provide input to the next output and so on.

**Data integrity and constraints**

Data Integrity refers to validity and consistency of data. Data Integrity means that the data should be accurate and consistent. This is done by providing some checks or constraints. These are consistency rules that the database is not permitted to violate. Constraints may apply to data items within a record or relationships between records. For example, the age of an employee can be between 18 and 70 years only. While entering the data for the age of an employee, the database should check this. However, if Grades of any student are entered, the data can be erroneously entered as Grade C for Grade A. In this case DBMS will not be able to provide any check as both A and C are of the same data type and are valid values.

**Efficient Data Access**

DBMS utilises techniques to store and retrieve the data efficiently at least for unforeseen queries. A complex DBMS should be able to provide services to end users, where they can efficiently retrieve the data almost immediately.

**Multiple User Interfaces**

Since many users having varying levels of technical knowledge use a database, a DBMS should be able to provide a variety of interfaces. This includes ─ a. query language for casual users, b. programming language interfaces for application programmers, c. forms and codes for parametric users, d. menu driven interfaces, and e. natural language interfaces for standalone users, these interfaces are still not available in standard form with commercial database.

**Representing complex relationship among data**

A database may include varieties of data interrelated to each other in many ways. A DBMS must have the capability to represent a variety of relationships among the data as well as to retrieve and update related data easily and efficiently.

**Improved Security**

Data is vital to any organisation and also confidential. In a shared system where multiple users share the data, all information should not be shared by all users. For example, the salary of the employees should not be visible to anyone other than the department dealing in this. Hence, database should be protected from unauthorised users. This is done by Database Administrator (DBA) by providing the usernames and passwords only to authorised users as well as granting privileges or the type of operation allowed. This is done by using security and authorisation subsystem. Only authorised users may use the database and their access types can be restricted to only retrieval, insert, update or delete or any of these. For example, the Branch Manager of any company may have access to all data whereas the Sales Assistant may not have access to salary details.

**Improved Backup and Recovery**

A file-based system may fail to provide measures to protect data from system failures. This lies solely on the user by taking backups periodically. DBMS provides facilities for recovering the hardware and software failures. A backup and recovery subsystem is responsible for this. In case a program fails, it restores the database to a state in which it was before the execution of the program.

**Support for concurrent transactions**

A transaction is defined as the unit of work. For example, a bank may be involved in a transaction where an amount of Rs.5000/- is transferred from account X to account Y. A DBMS also allows multiple transactions to occur simultaneously.

**PROCESS LOGIC:**

**Process logic for Student Registration:**-

Correct

Generate ID and Password and give to user as well as store into Database

Provide necessary details

Student Perform registration

no

yes

**Process Logic For Teacher Module**

After that Teacher is a our Expert

Generate a valid Teacher Id and Send to teacher as well as store into Database.

correct

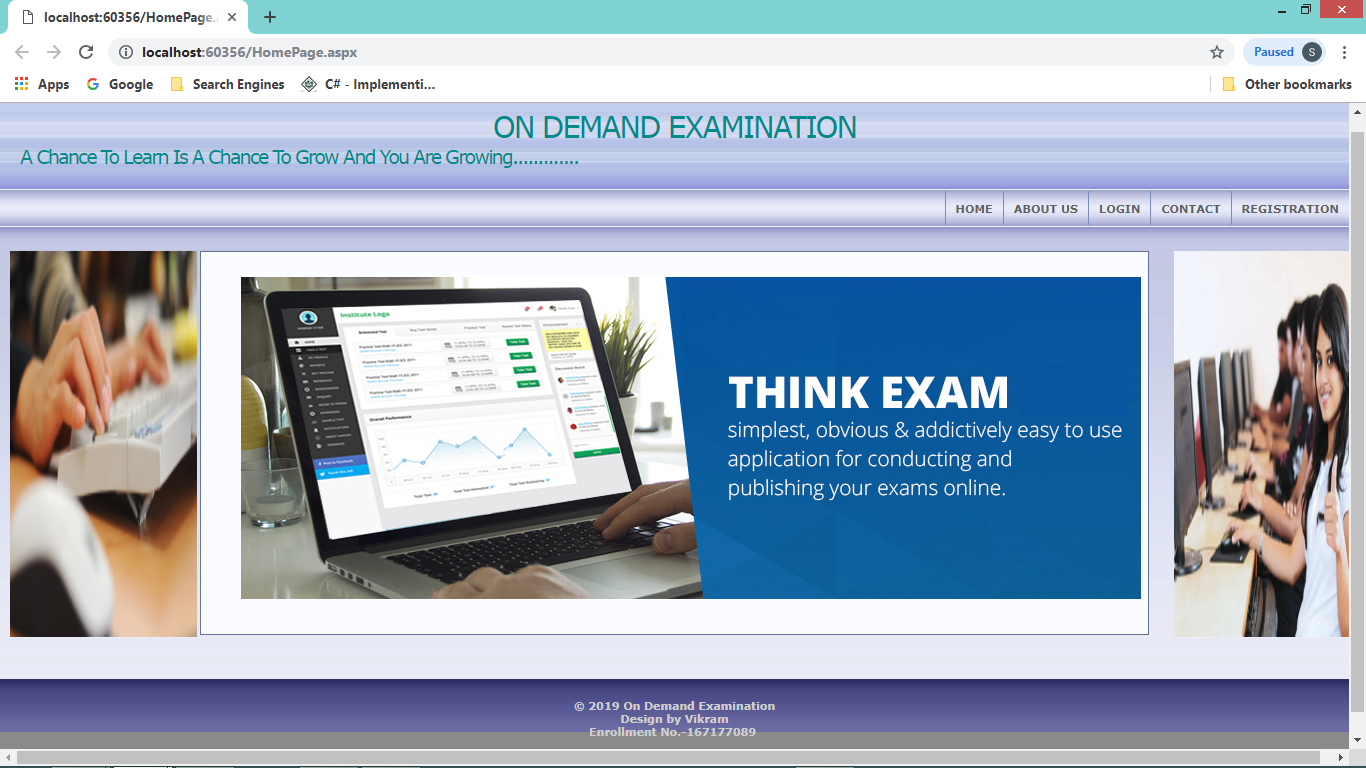
Fill Registration form

|  |
| --- |
| yes |

|  |
| --- |
| no |

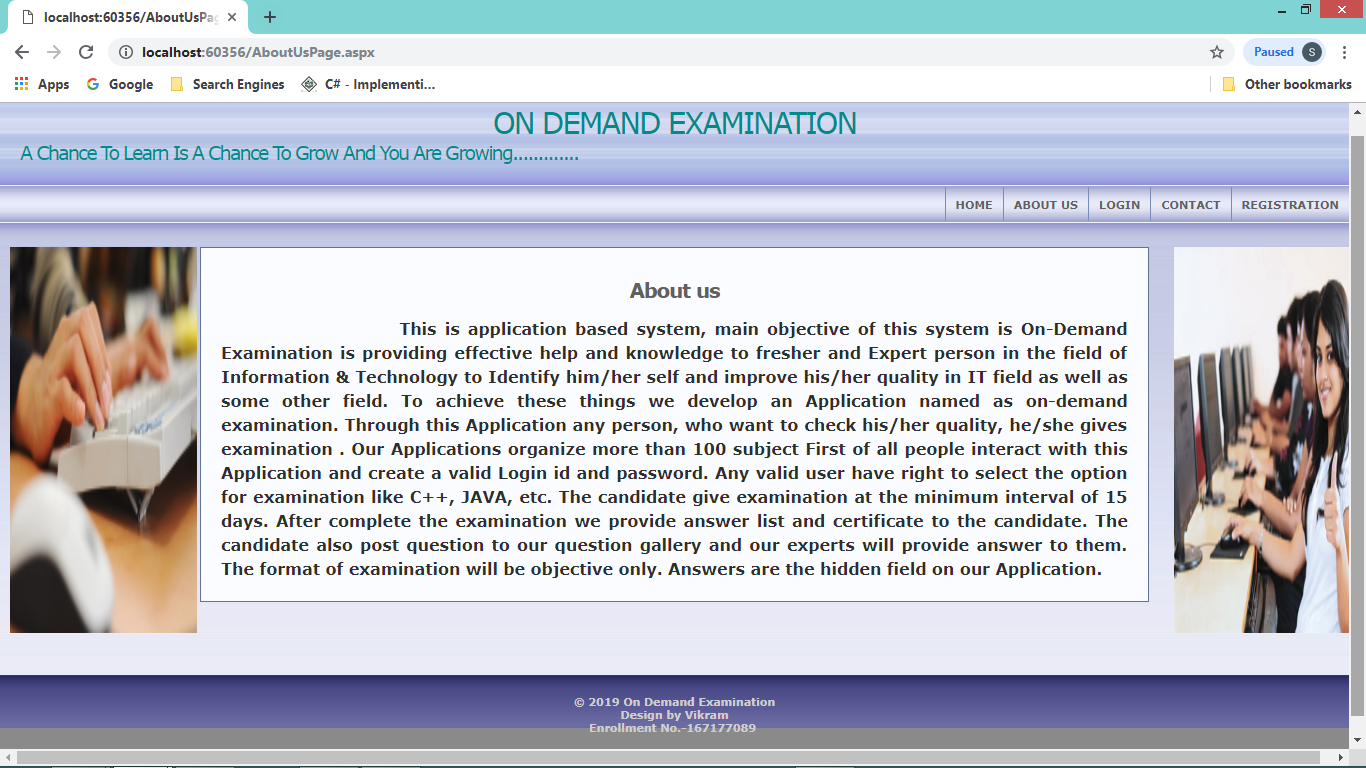
**User Interface:**

**Home Page:-**



The Home Page display to the user at first. Link to the other page are provided to the user at home page.

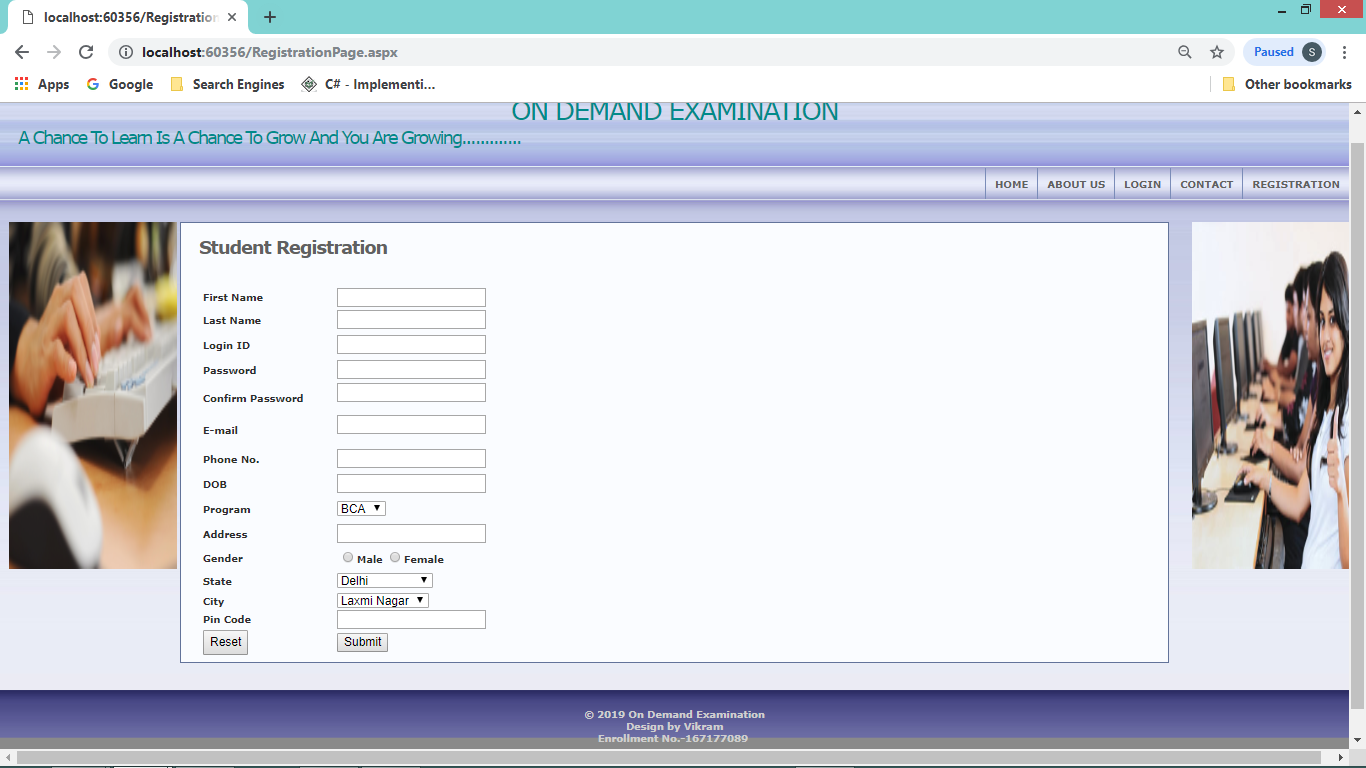
**About Us page:-**



The about us page design for give the all information about the system to the user.**Contact Us page:-**

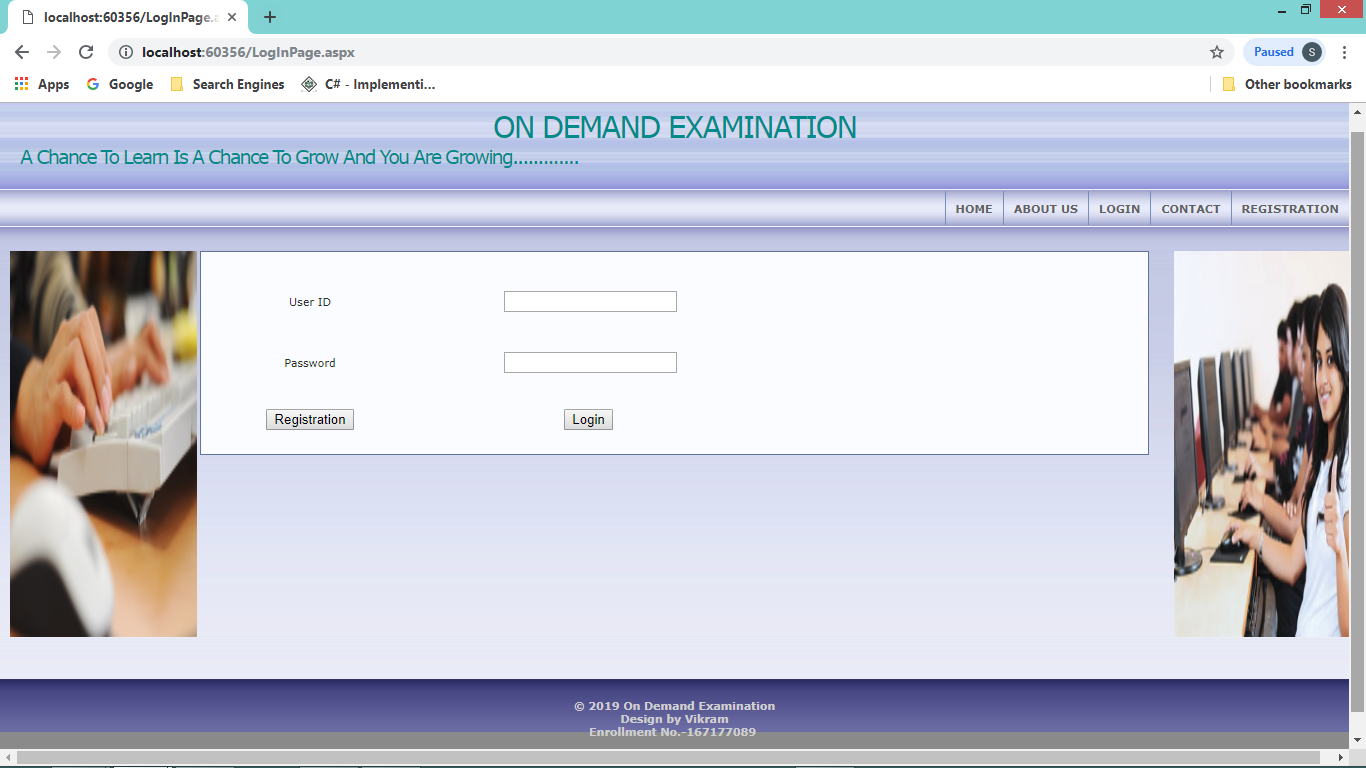


This page is design for contact, so that user can solve there query.**Registration page:-**



The student who want to take online exam can register here first. **USER**

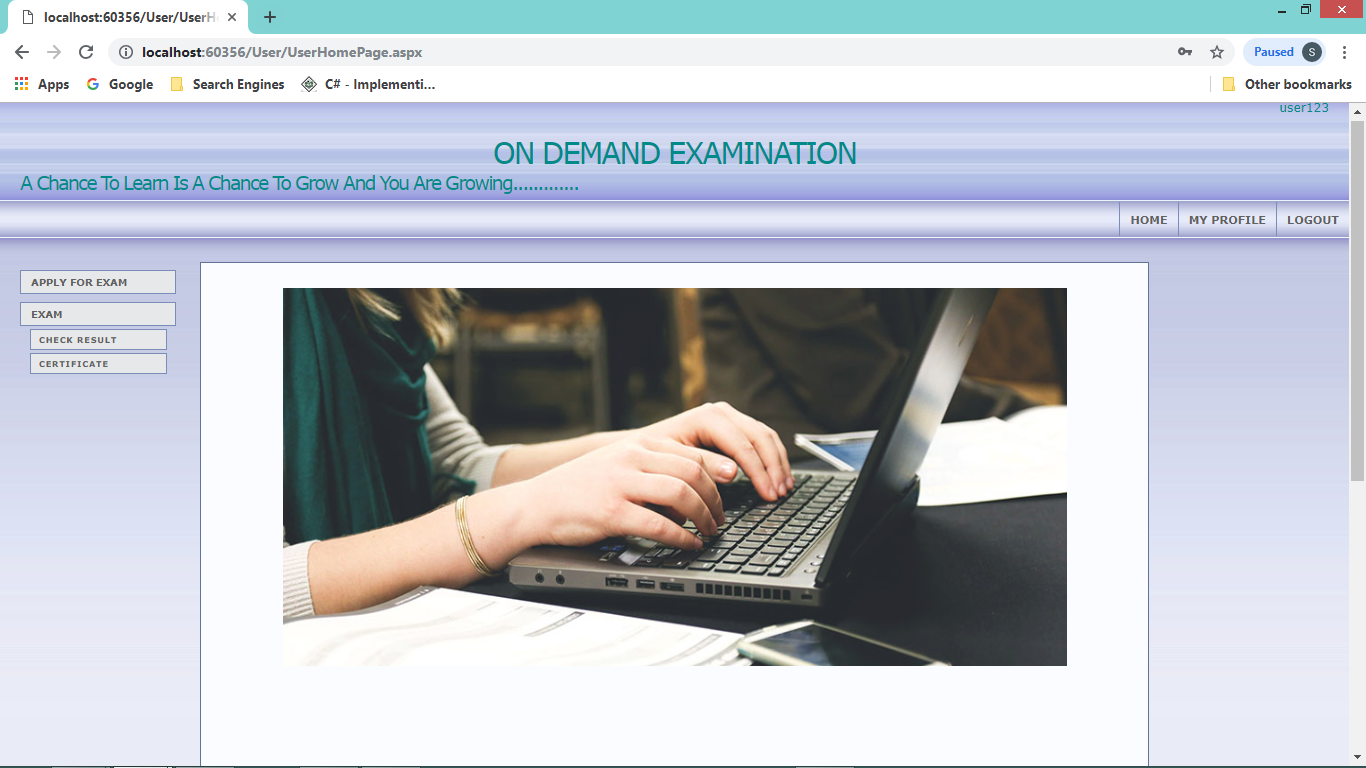
**Log-in Page:-**



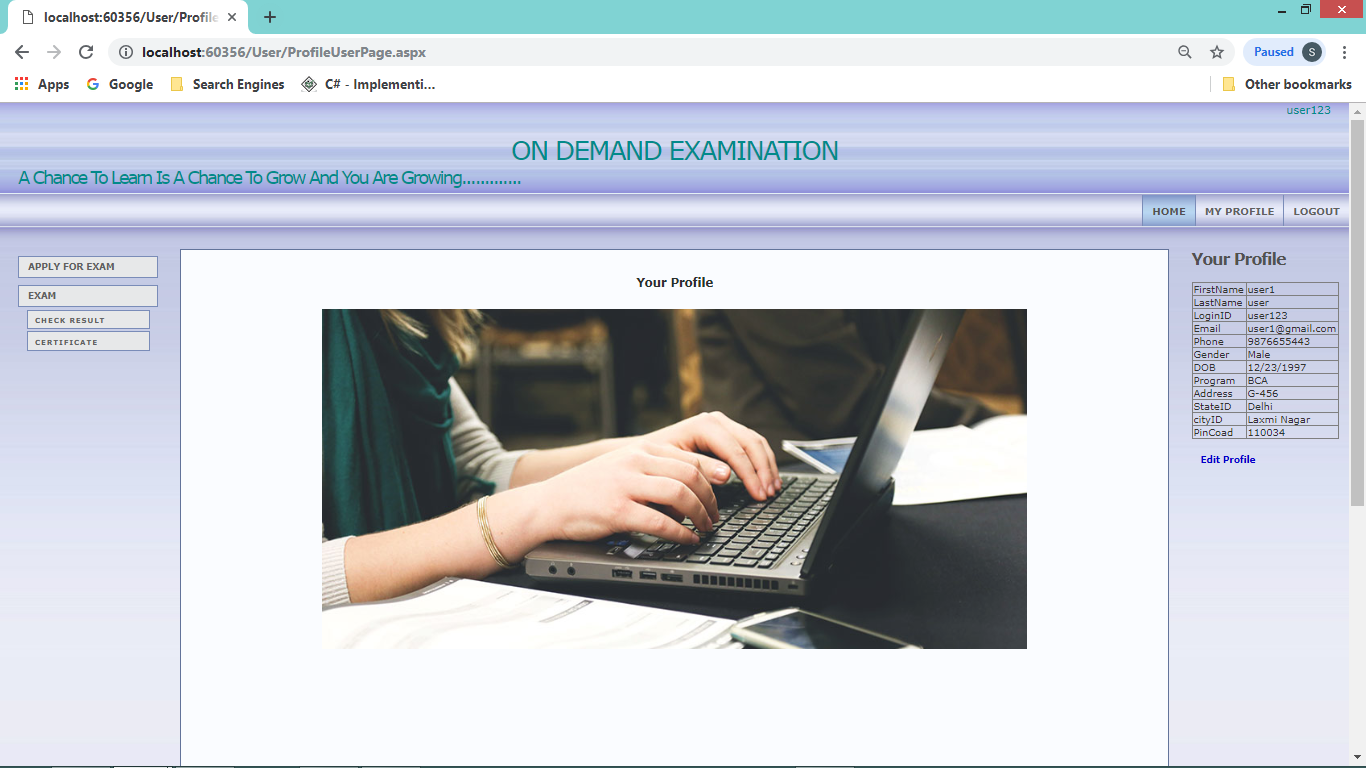
After registration student can login

**User**

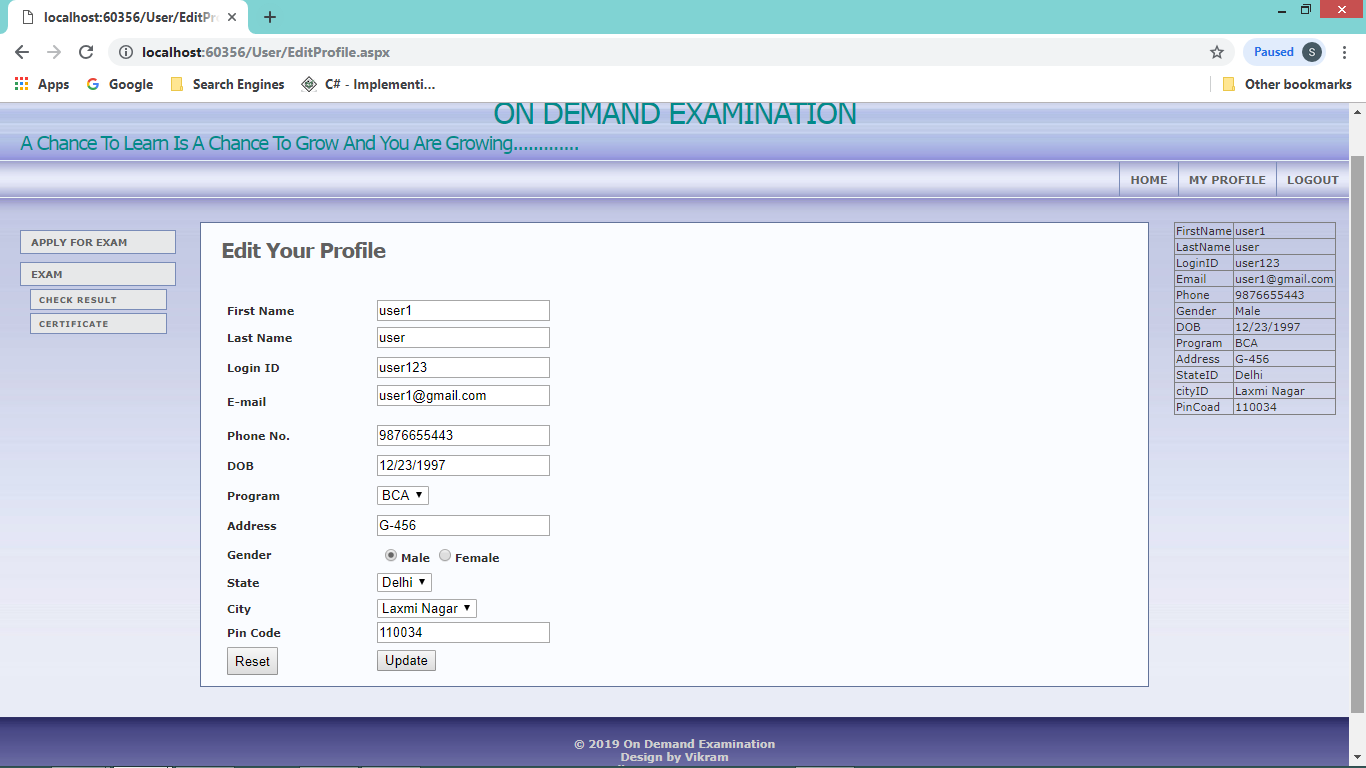
**Home Page:**



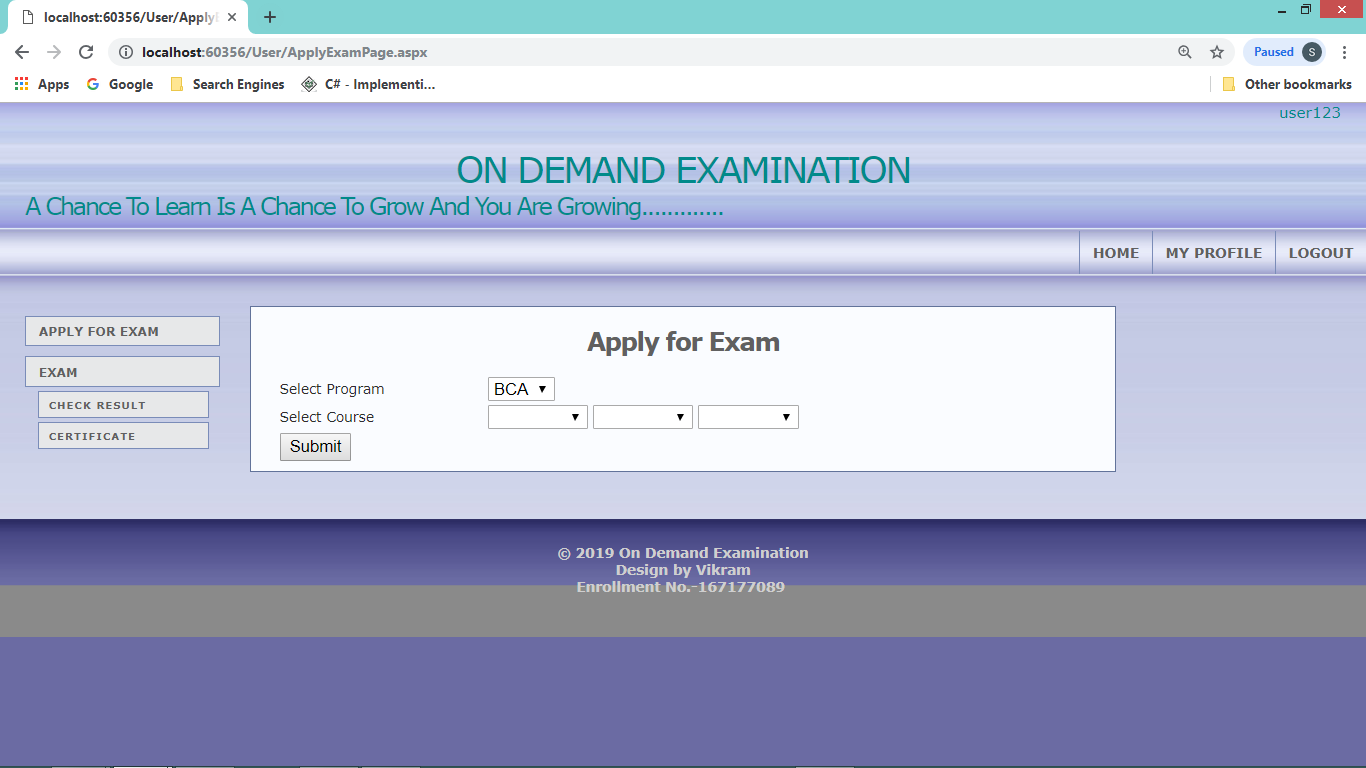
The is user Home page after login**My Profile Page:-**



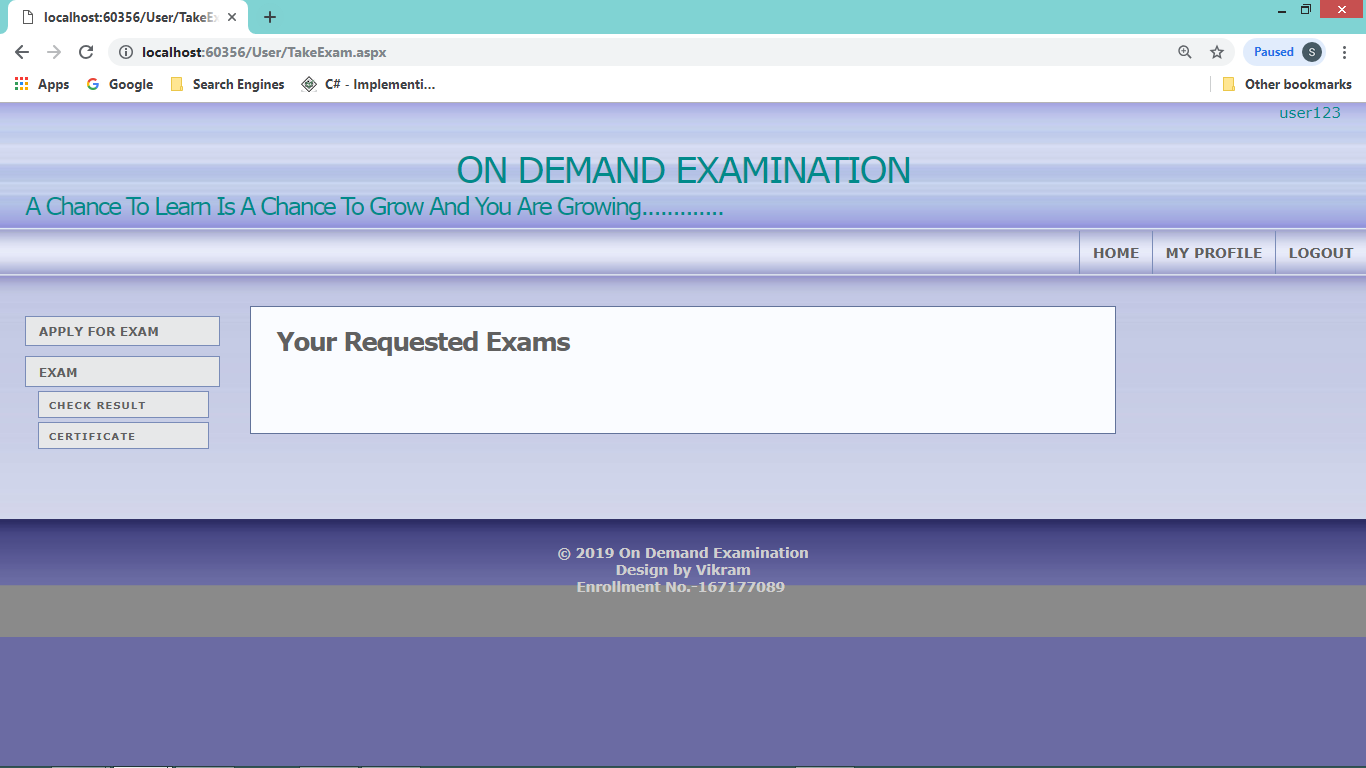
In profile section student can see their registration details.**Edit Profile Page-**



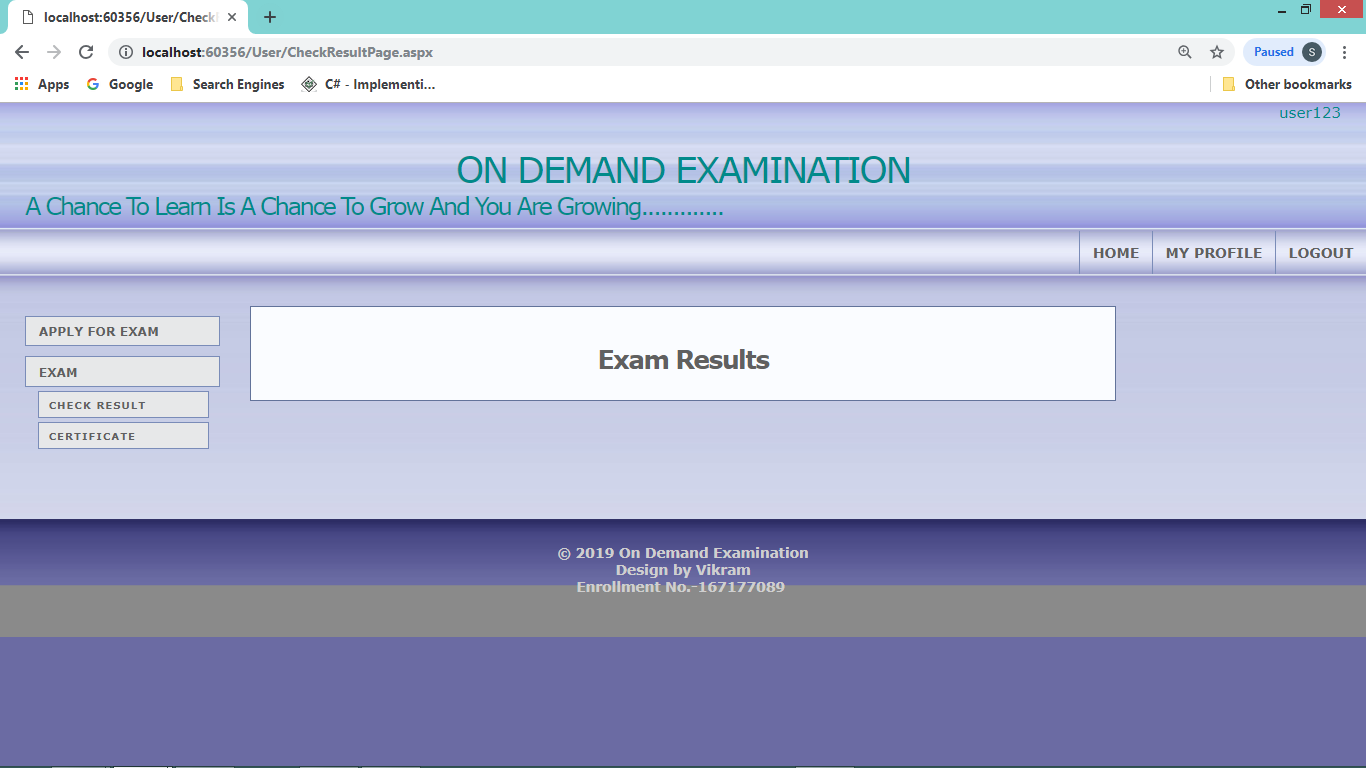
Edit profile page allow student to edit there profile.**Apply Exam page:-**



For apply Exam student have to select course and click on submit button. Student can apply only 3 course at a time.**Exam Page:-**

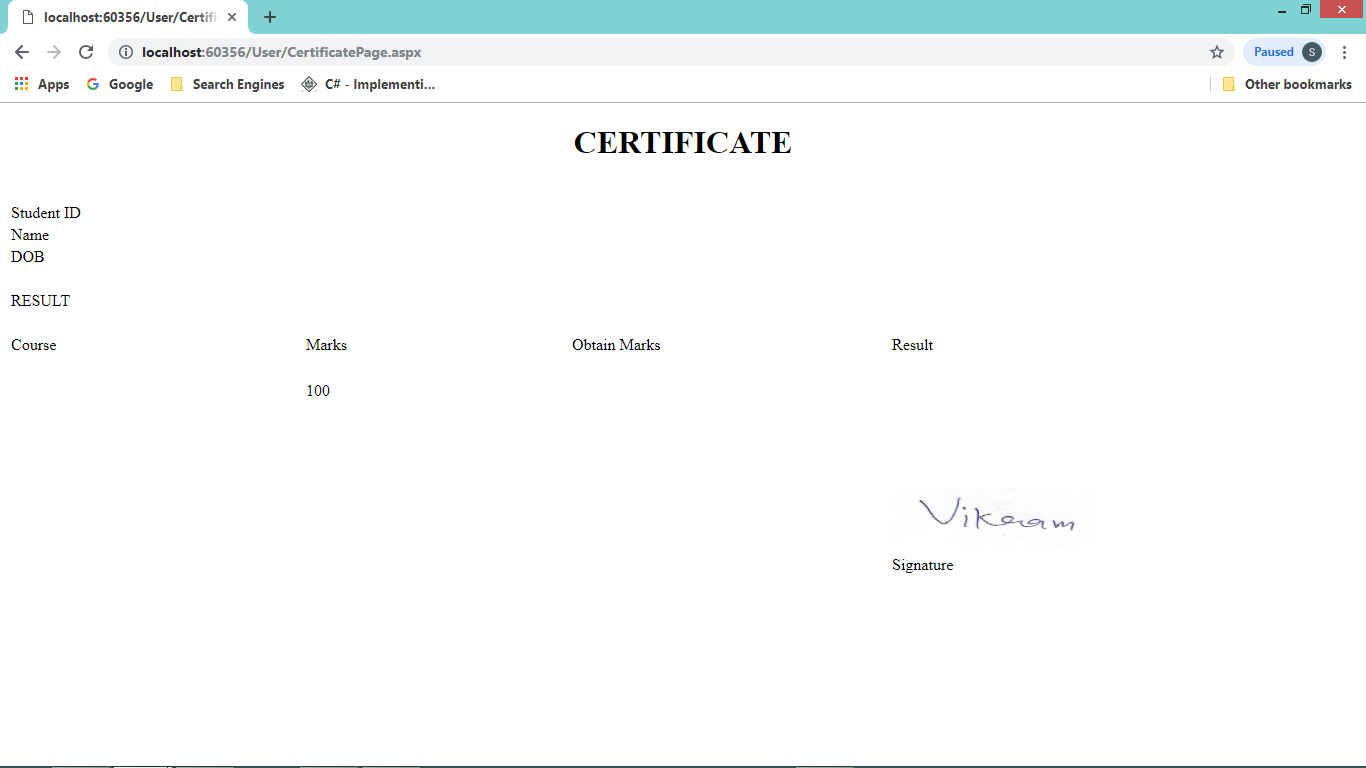


This page display the course that student apply and if request is approved then exam option is display to the user.**Result Page:**



Result page display all the exam that student have done.

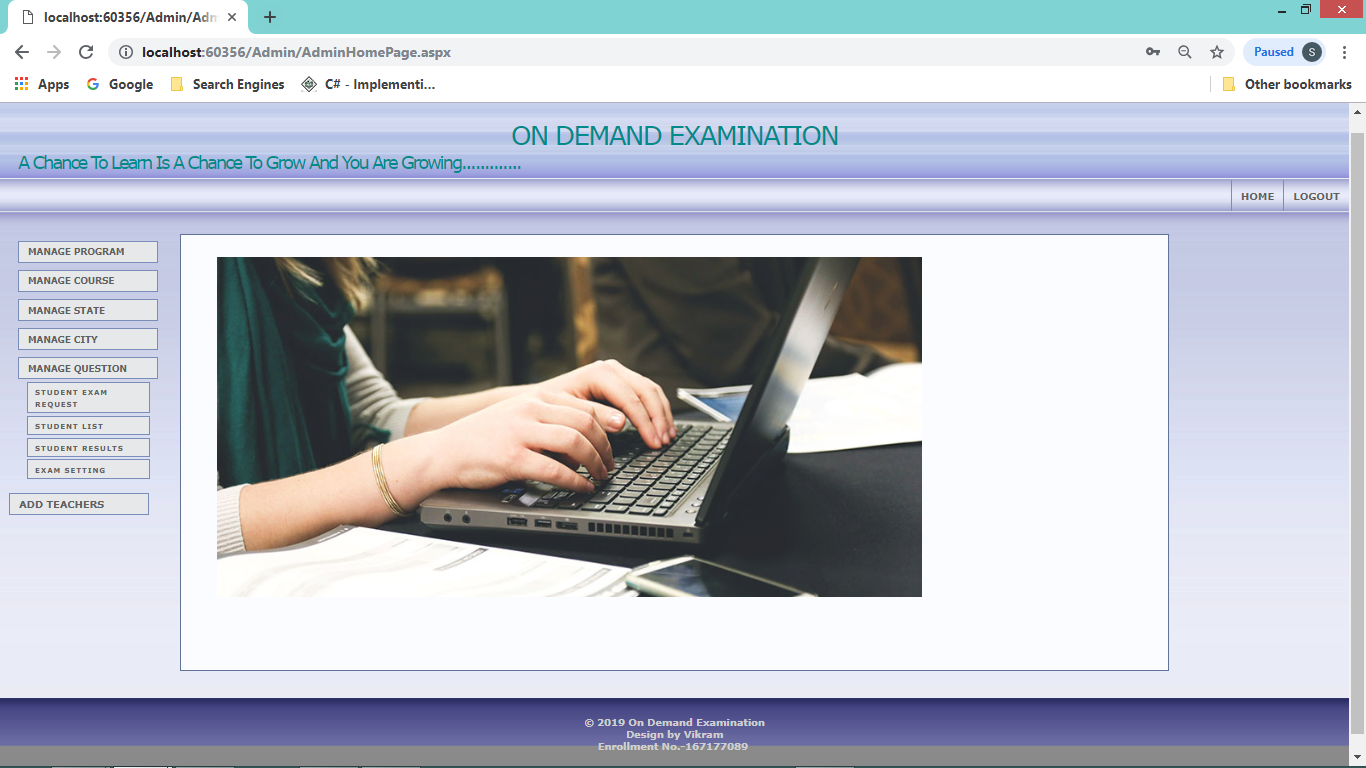
Certificate Page:



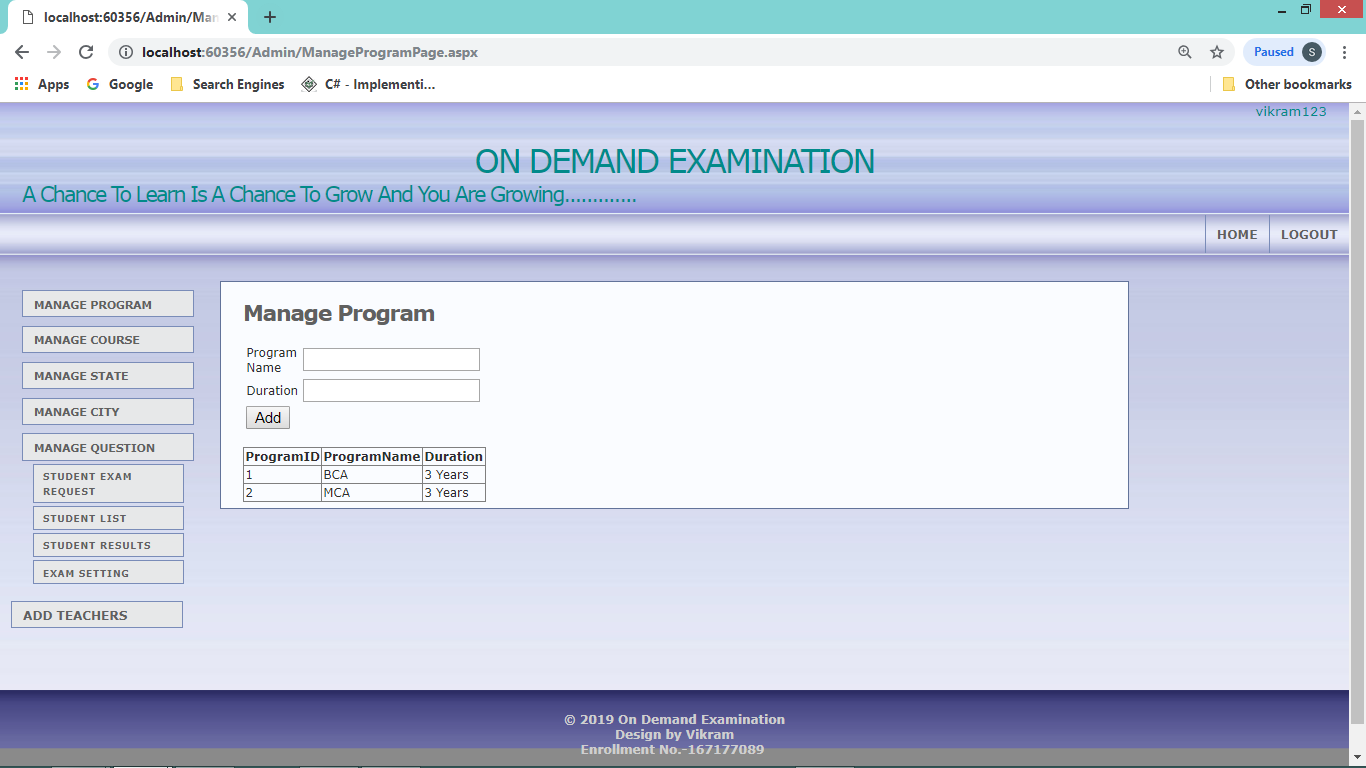
Student can download there exam certificate.

**Admin:**

**Admin home page:**

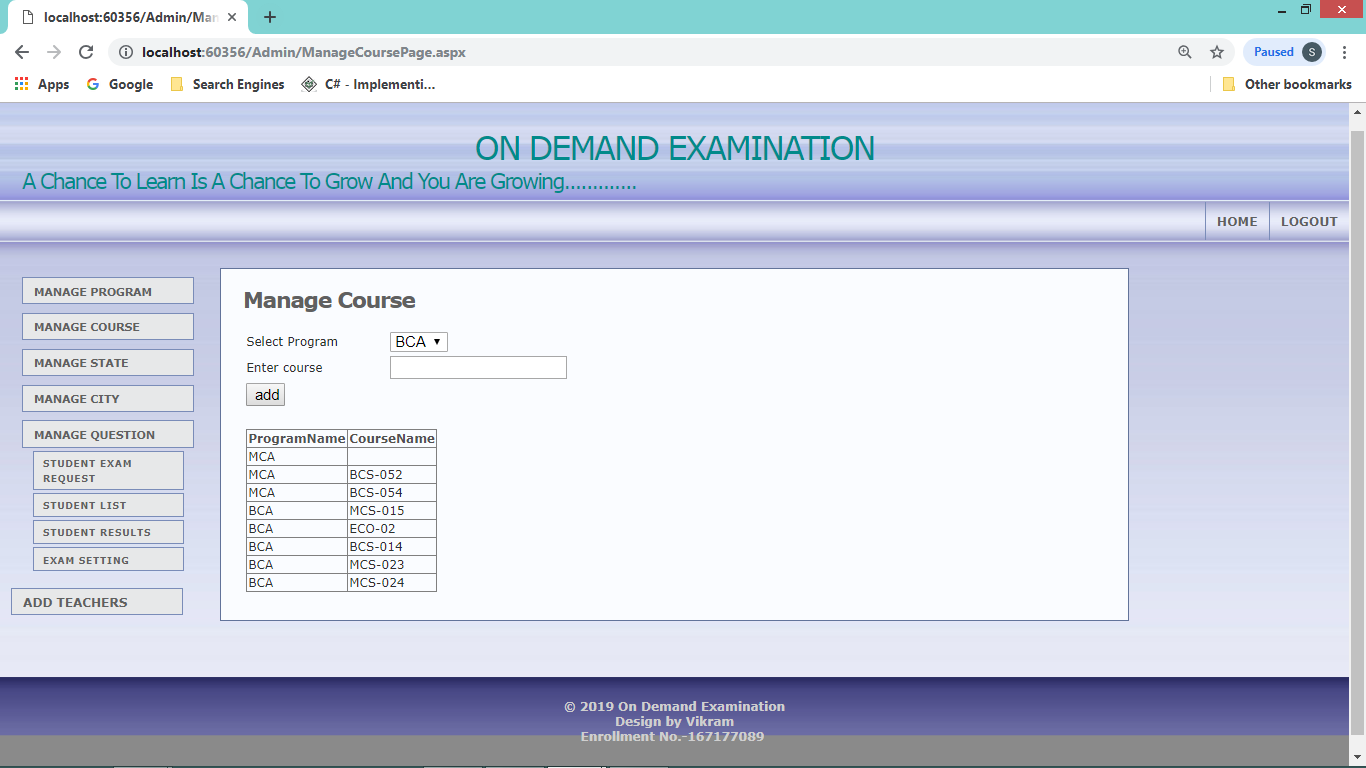


This is admin page where all the option for performing the task are display.**Manage Program page:**

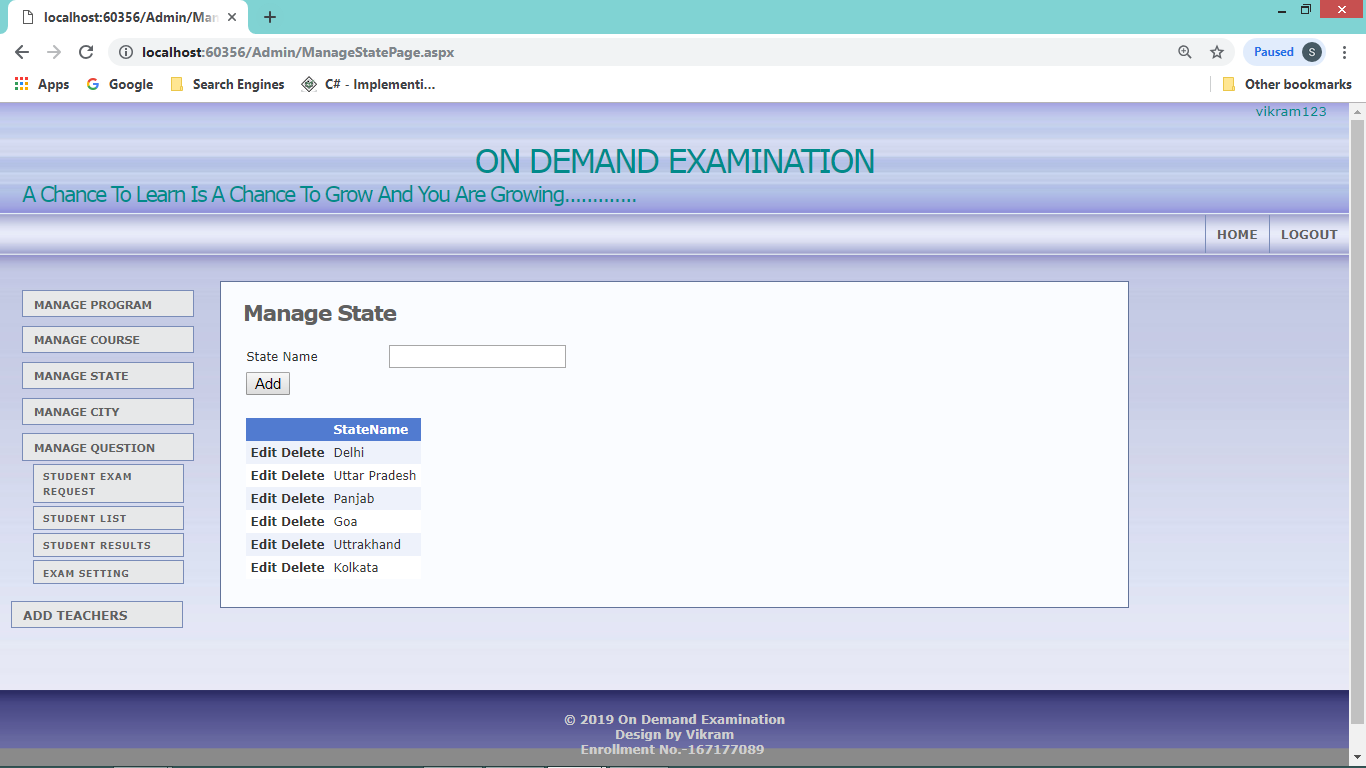


At this page admin can add program or there details.

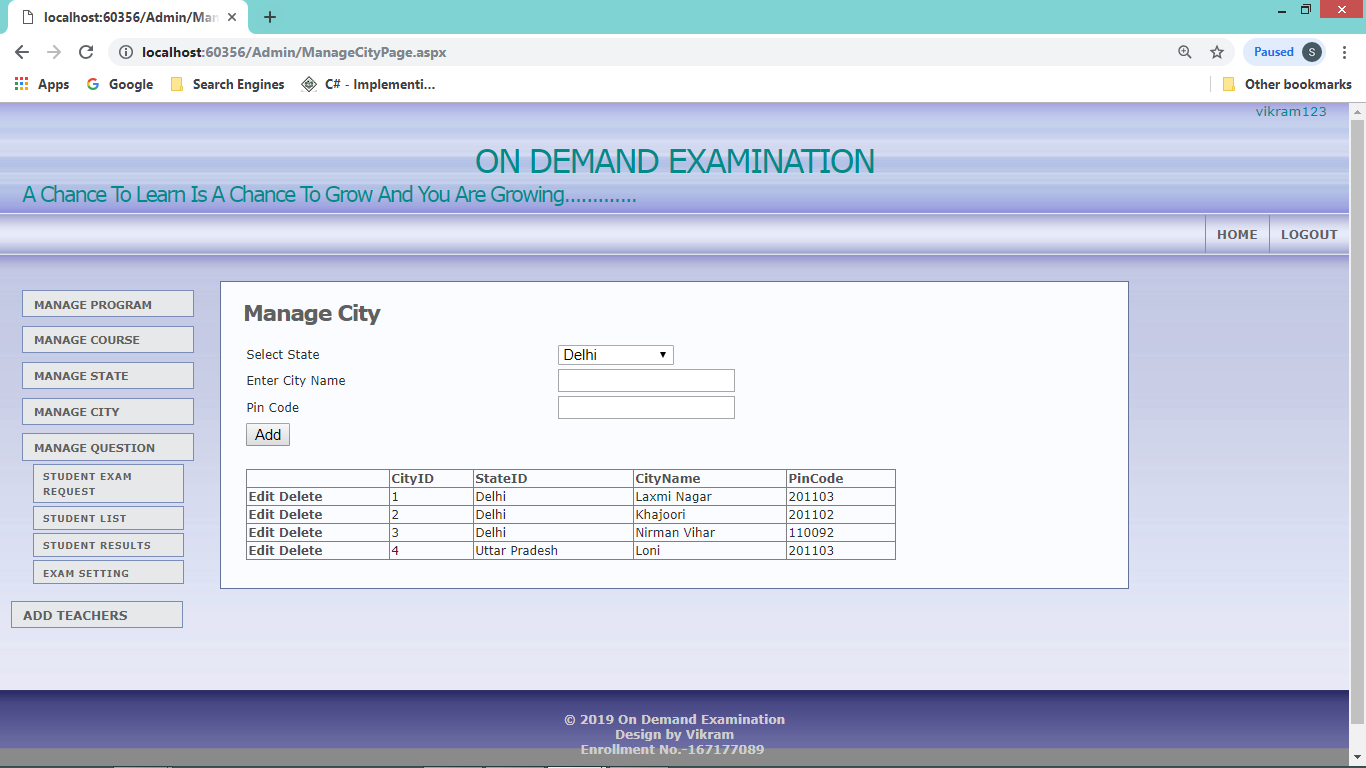
**Manage Course Page:**



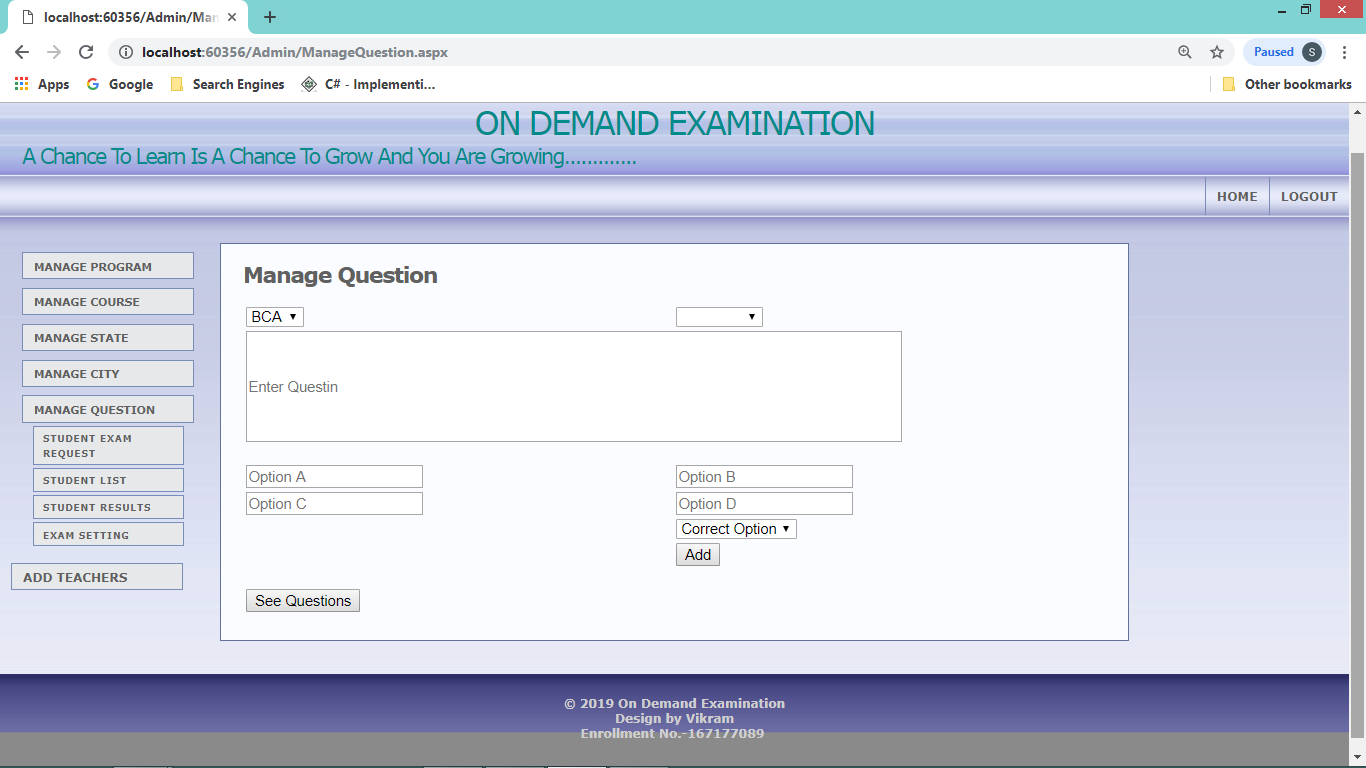
Here admin can add course. First admin have to select program and enter the course then click add button to add course.**Manage State Page:**



For adding state **Manage City Page:**

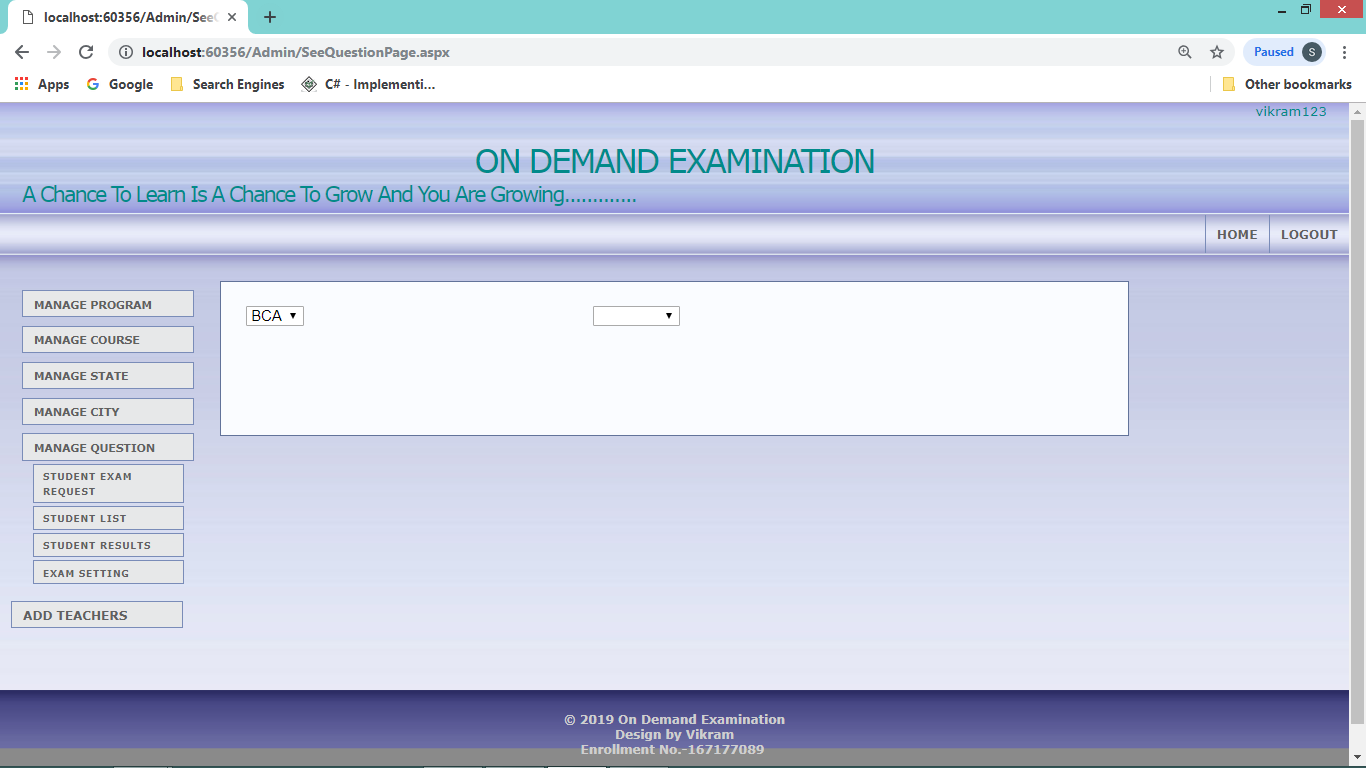


For adding city and pin-code corresponding to state. **Manage Question Page:**

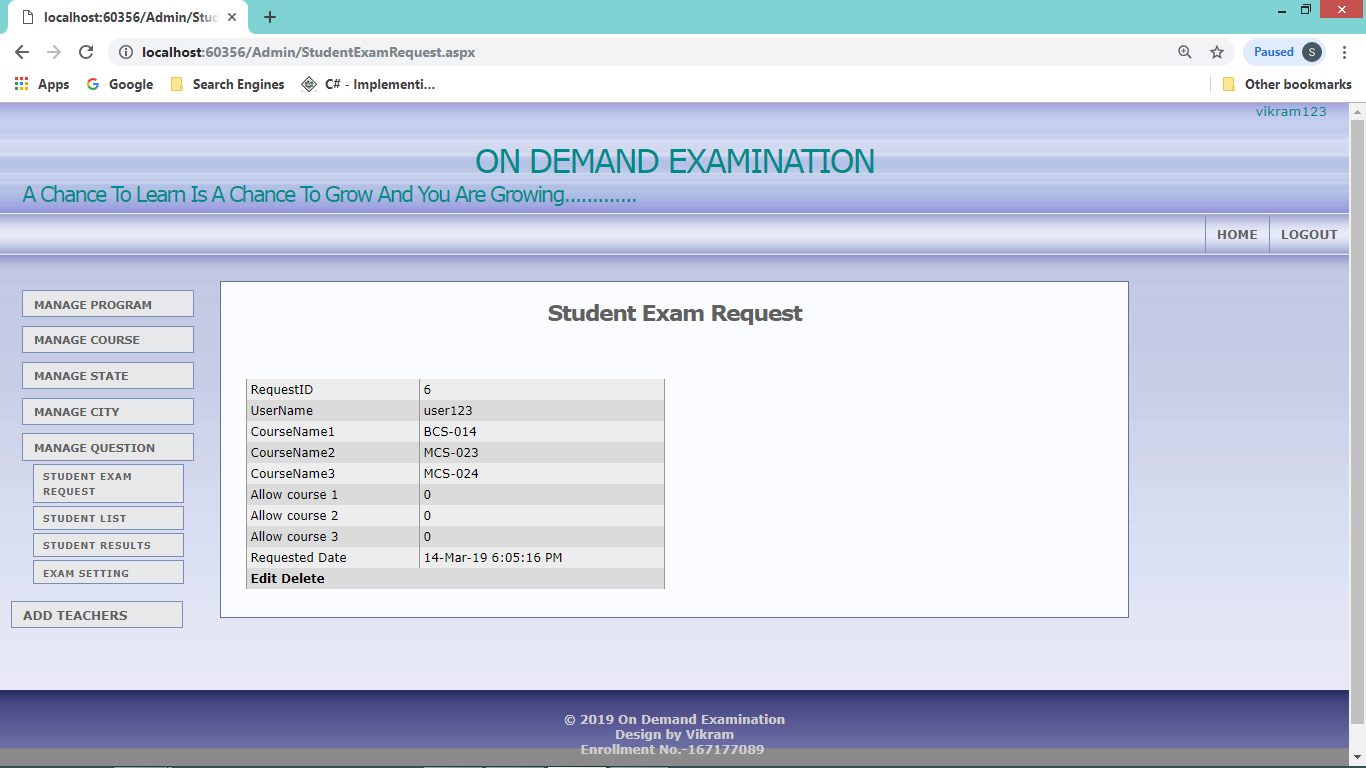


For storing question manage page is designed. Here admin select program and there course and then enter question and their options, correct option is choosing from drop down list.

**See Question Page**



This page display the question and answer. First select the program from drop down and then course from drop down to see the question, options and correct option.**Student Exam Request page:**



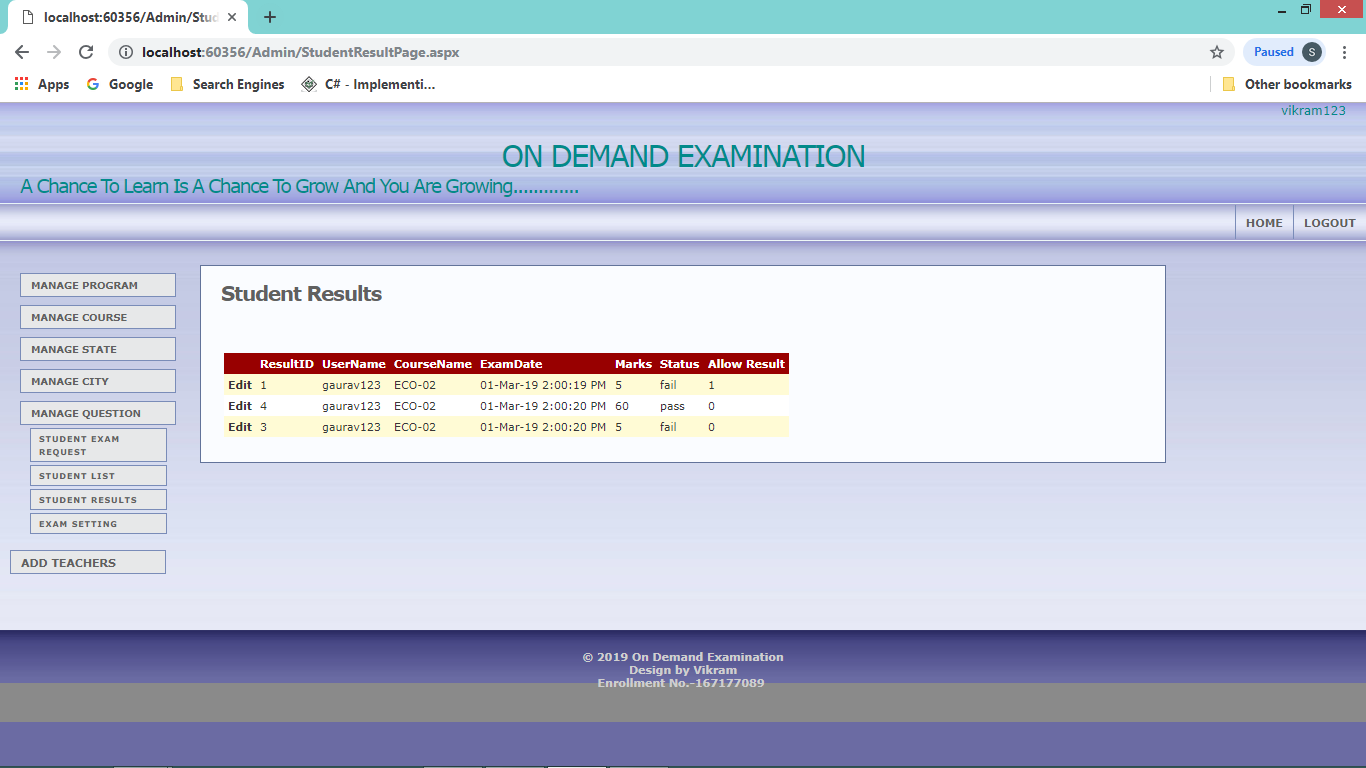
The “student exam request page” display the requested exam/course by student if questions for that paper is available for that course admin allow the exam by edit “Allow Course” “0 “ to “1”. Or if question for that paper is not added then admin add the question. From ‘manage question page”.

**Student list page:**

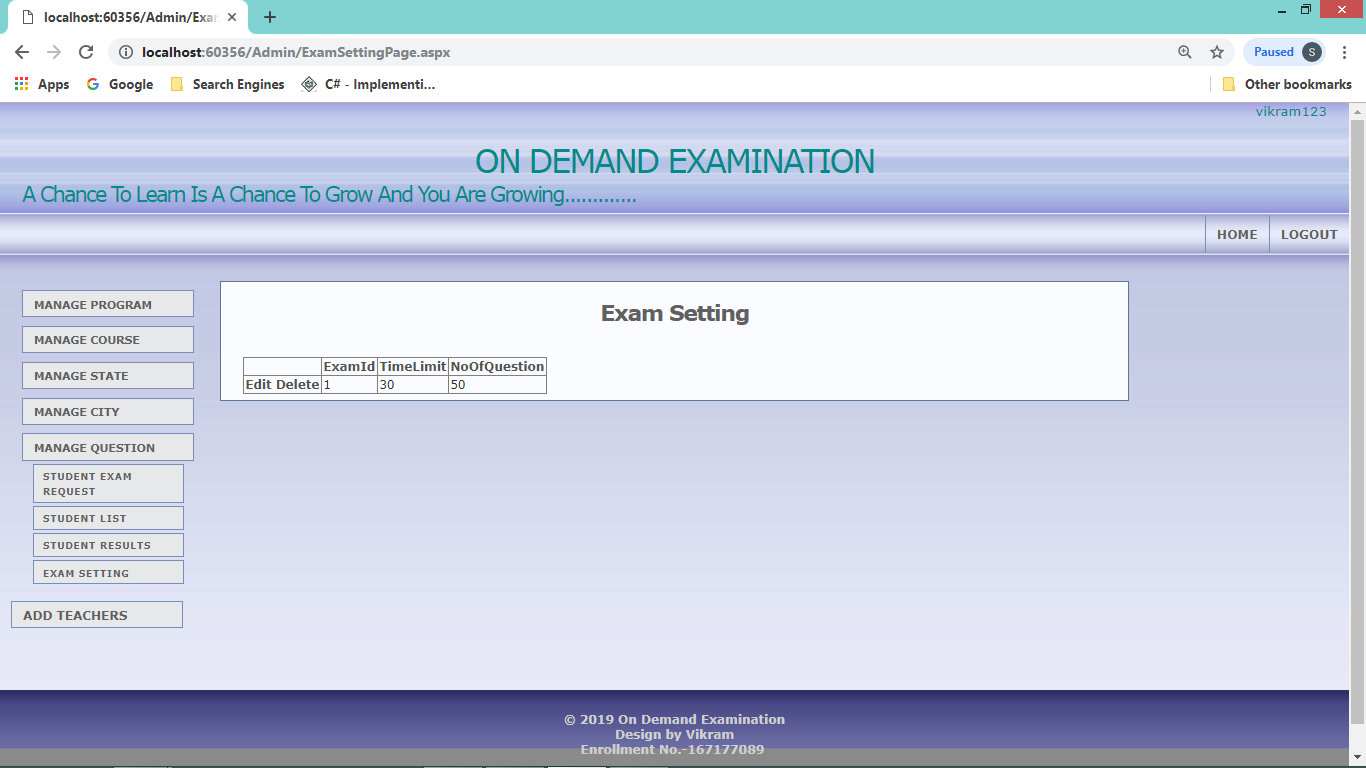


This page display all the registered student.

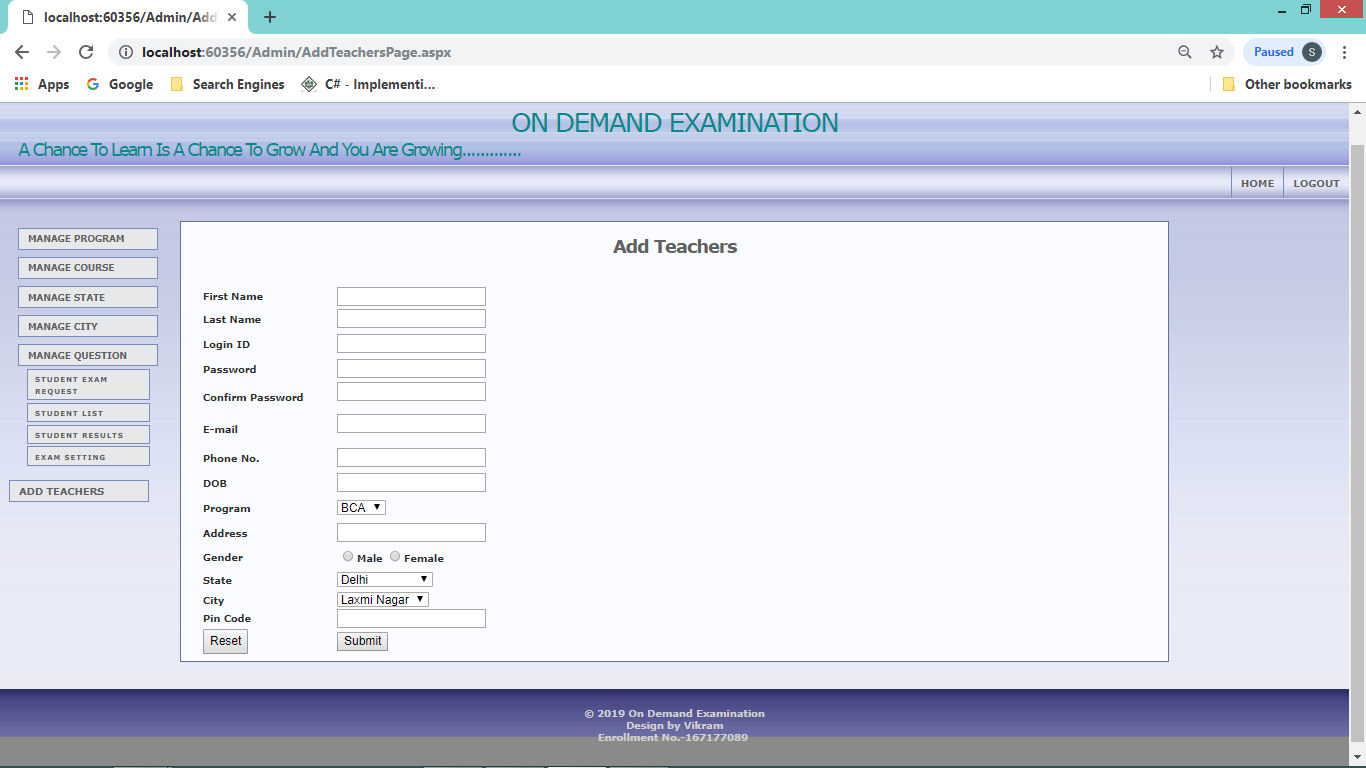
**Student Result Page:**



This page display the result of the student for declare result admin allow the result by changing “Allow Result” “0” to ”1”.**Exam Setting page**



This page is design for set exam time and no. Of question in exam**Add Teachers Page**



This page is design for adding more teachers or admin.**Program Coding**

**Web.config**

<?xml version="1.0"?>

<configuration>

<connectionStrings>

<add name ="connectionDb" connectionString="Data Source=DELL\SQLEXPRESS;Initial Catalog=OnDemandExam;Integrated Security=True" providerName="system.Data.SqlClient"/>

</connectionStrings>

<appSettings/>

<system.web>

<compilation debug="true" targetFramework="4.5"/>

<authentication mode="Windows"/>

<pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID"/>

</system.web>

</configuration>

**MasterPage.Master**

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="MasterPage.master.cs" Inherits="OnDemandExamination.MasterPage" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title></title>

<link rel="stylesheet" href="style.css" type="text/css" media="screen" />

<asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">

</asp:ContentPlaceHolder>

<style type="text/css">

.auto-style1 {

margin-left: 0px;

}

</style>

</head>

<body>

<form id="form1" runat="server">

<div>

<div id="container">

<div id="sitename" style="color:darkcyan">

<h1 class="center">ON DEMAND EXAMINATION</h1>

<h2>A Chance To Learn Is A Chance To Grow And You Are Growing.............</h2>

</div>

<div id="mainmenu">

<ul>

<li><a href="HomePage.aspx">Home</a></li>

<li><a href="AboutUsPage.aspx">About Us</a></li>

<li><a href="LogInPage.aspx">Login</a></li>

<li><a href="ContactUsPage.aspx">Contact</a></li>

<li><a href="RegistrationPage.aspx">Registration</a></li>

</ul>

</div>

<div id="wrap">

<div id="leftside">

<asp:Image ID="Image1" runat="server" Height="386px" ImageUrl="~/img/exam3.png" Width="187px" />

</div>

<div id="rightside">

<asp:Image ID="Image2" runat="server" CssClass="auto-style1" Height="386px" ImageUrl="~/img/exam.jpg" Width="187px" />

</div>

<div id="content">

<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

</asp:ContentPlaceHolder>

</div>

<div class="clearingdiv">&nbsp;</div>

</div>

</div>

<div id="footer">© 2019 On Demand Examination<br />

Design by Vikram<br />

Enrollment No.-167177089

<br />

&nbsp;&nbsp;

</div>

</div>

</form>

</body>

</html>

**MasterPage.Master.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination

{

public partial class MasterPage : System.Web.UI.MasterPage

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**style.css**

body {

background:#6B6BA5;

color:#303030;

font:76% Verdana,Tahoma,Arial,sans-serif;

margin:0;

padding:0;

text-align:center;

}

a {

color:#505050;

font-weight:bold;

text-decoration:none;

}

a:hover {

color:#808080;

text-decoration:underline;

}

p {

line-height:1.5em;

margin:0 0 15px;

}

#container {

background:#f0f0f0 url(img/bodybg.jpg) repeat-x;

color:#303030;

margin:0;

min-width:770px;

padding:0;

text-align:left;

width:100%;

}

#sitename {

color:#ffffff;

height:92px;

margin:0 20px 10px;

text-align:left;

}

#sitename h1,#sitename h2 {

font-weight:400;

letter-spacing:-2px;

margin:0;

padding:0;

}

#sitename h1 {

font-size:2.4em;

padding-top:20px;

}

#sitename h2 {

font-size:1.6em;

}

#mainmenu {

float:right;

margin-bottom:25px;

}

#mainmenu ul {

font-size:16px;

margin:0;

padding:0;

}

#mainmenu li {

float:left;

height:35px;

list-style:none;

margin:0;

padding:0;

}

#mainmenu a {

border-left:1px solid #7B8DBA;

color:#606060;

display:block;

font-size:0.7em;

padding:11px 10px 10px;

text-transform:uppercase;

}

#mainmenu a:hover {

background:#f0f0f0 url(img/menuhover.jpg) top left repeat-x;

color:#505050;

text-decoration:none;

}

#mainmenu a.current {

background:#f0f0f0 url(img/menuhover.jpg) top left repeat-x;

color:#505050;

text-decoration:none;

}

#wrap {

clear:both;

font-size:0.9em;

padding:0;

}

#leftside,#rightside {

margin:0;

padding:0 10px 10px;

width:165px;

}

#leftside {

float:left;

margin-right:10px;

}

#rightside {

float:right;

margin-left:10px;

}

#leftside h1,#rightside h1 {

color:#505050;

font-size:1.6em;

font-weight:bold;

letter-spacing:-1px;

margin:0 0 12px;

}

#leftside h2,#rightside h2 {

font-size:1.1em;

margin:0 0 20px;

}

#leftside p,#rightside p {

font-size:0.9em;

line-height:1.4em;

margin:0 0 16px 10px;

}

.linklist {

list-style:none;

margin:0 0 16px 10px;

padding:0;

}

.linklist li {margin-bottom:0.7em;}

.nav {

background:#e8e9ea;

border:1px solid #7B8DBA;

color:#606060;

display:block;

margin-top:8px;

padding:5px 4px 4px 10px;

position:relative;

text-transform:uppercase;

width:140px;

}

.nav:hover,.active {

background:#f8f9fa;

border:1px solid #62739B;

color:#303030;

text-decoration:none;

}

.sub {

font-size:0.8em;

letter-spacing:1px;

margin:3px 0 2px 10px;

padding:4px 2px 2px 8px;

width:125px;

}

#content,#contentalt {

background-color:#fafcff;

border:1px solid #62739B;

color:#2a2a2a;

padding:15px 20px 5px;

}

#content {margin:0 200px;}

#contentalt {margin:0 200px 0 20px;}

#content h1,#content h2,#contentalt h1,#contentalt h2 {

background-color:inherit;

color:#606060;

font-size:1.8em;

font-weight:bold;

letter-spacing:-1px;

margin:0 0 15px;

padding:0;

}

#content h2,#contentalt h2 {

font-size:1.6em;

margin-bottom:10px;

}

#content img,#contentalt img {

border:1px solid #7B8DBA;

margin:5px 15px 6px;

padding:5px;

}

#footer {

background:#8b8b8b url(img/footerbg.jpg) top left repeat-x;

clear:both;

color:#d0d0d0;

font-size:0.9em;

font-weight:bold;

margin:0;

padding:20px 0;

text-align:center;

width:100%;

}

#footer a {

color:#d0d0d0;

font-weight:bold;

}

.thumbnail {

background:#fafbfc;

border:1px solid #7B8DBA;

margin:0 0 10px 10px;

padding:5px;

}

.searchform {margin:0;}

.searchbox {

background:#f0f0f0;

border:1px solid #7B8DBA;

margin:0 4px 0 0;

width:100px;

}

.searchbutton {

background:#f0f0f0;

border:1px solid #7B8DBA;

}

.clearingdiv {

clear:both;

height:30px;

width:1px;

}

.hide {display:none;}

.intro {

font-size:1.1em;

font-weight:bold;

letter-spacing:-1px;

}

.small {font-size:0.8em;}

.large {font-size:1.4em;}

.center {text-align:center;}

.right {text-align:right;}

**HomePage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true" CodeBehind="HomePage.aspx.cs" Inherits="OnDemandExamination.HomePage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p>

<asp:Image ID="Image1" runat="server" ImageUrl="~/img/exam2.jpg" />

</p>

</asp:Content>

HomePage.aspx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination

{

public partial class HomePage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**AboutUsPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true" CodeBehind="AboutUsPage.aspx.cs" Inherits="OnDemandExamination.AboutUsPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p><h1 class="center">About us</h1></p>

<p class="auto-style2"> <strong>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; This is application based system, main objective of this system is On-Demand Examination is providing effective help and knowledge to fresher and Expert person in the field of Information & Technology to Identify him/her self and improve his/her quality in IT field as well as some other field. To achieve these things we develop an Application named as on-demand examination. Through this Application any person, who want to check his/her quality, he/she gives examination . Our Applications organize more than 100 subject First of all people interact with this Application and create a valid Login id and password. Any valid user have right to select the option for examination like C++, JAVA, etc. The candidate give examination at the minimum interval of 15 days. After complete the examination we provide answer list and certificate to the candidate. The candidate also post question to our question gallery and our experts will provide answer to them. The format of examination will be objective only. Answers are the hidden field on our Application.</strong></p>

</asp:Content>

<asp:Content ID="Content2" runat="server" contentplaceholderid="ContentPlaceHolder2">

<style type="text/css">

.auto-style2 {

text-align: justify;

font-size: medium;

}

</style>

</asp:Content>

**AboutUsPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination

{

public partial class AboutUsPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**ContactUsPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true" CodeBehind="ContactUsPage.aspx.cs" Inherits="OnDemandExamination.ContactUsPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<br />

<h1> contact No.:- 8750284362</h1><br />

<h1> Email :- <a href="mailto:vikram2050@gmail.com">vikram205051@gmail.com</a> </h1>

<p> &nbsp;</p>

<h1> Enrollment No. :- 167177089 </h1>

</asp:Content>

**ContactUsPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination

{

public partial class ContactUsPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**LogInPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true" CodeBehind="LogInPage.aspx.cs" Inherits="OnDemandExamination.LogInPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<table style="width: 563px" class="center">

<tr>

<td style="width: 112px; height: 63px;" class="center">User ID</td>

<td style="width: 248px; height: 63px;">

<asp:TextBox ID="TextBoxUserId" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 112px; height: 52px;" class="center">Password</td>

<td style="width: 248px; height: 52px;">

<asp:TextBox ID="TextBoxPassword" runat="server" TextMode="Password"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 112px; height: 53px;">

<asp:Button ID="ButtonLogin0" runat="server" Text="Registration" OnClick="ButtonLogin0\_Click" />

</td>

<td style="width: 248px; height: 53px;" class="center">

<asp:Button ID="ButtonLogin" runat="server" Text="Login" OnClick="ButtonLogin\_Click" />

<asp:Label ID="LabelMessage" runat="server"></asp:Label>

</td>

</tr>

</table>

</asp:Content>

**LogInPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Configuration;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination

{

public partial class LogInPage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void ButtonLogin\_Click(object sender, EventArgs e)

{

try

{

string \_ProcName = "loginCheck";

SqlParameter[] \_parameter = {

new SqlParameter("@LoginID",TextBoxUserId.Text),

new SqlParameter("@Password",TextBoxPassword.Text)

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

if (dr["role"].ToString().Equals("1"))

{

Session["admin"] = TextBoxUserId.Text;

Response.Redirect("~\\Admin\\AdminHomePage.aspx");

}

else if (dr["role"].ToString().Equals("0"))

{

Session["user"] = TextBoxUserId.Text;

Response.Redirect("~\\User\\UserHomePage.aspx");

}

}

}

catch (Exception ex)

{

LabelMessage.Text = ex.Message;

}

}

protected void ButtonLogin0\_Click(object sender, EventArgs e)

{

Response.Redirect("~/registrationpage.aspx");

}

}

}

**RegistrationPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true" CodeBehind="RegistrationPage.aspx.cs" Inherits="OnDemandExamination.RegistrationPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Student Registration</h1>

<table style="border-width: thick; border-color: #000000; width:100%; padding: 3px; font-weight: bold;" id="RegstrationTable">

<tr>

<td style="width: 146px">&nbsp;</td>

</tr>

<tr>

<td style="width: 146px">First Name</td>

<td>

<asp:TextBox ID="FirstName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="FirstName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Last Name</td>

<td style="height: 26px">

<asp:TextBox ID="LastName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="LastName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Login ID</td>

<td style="height: 26px">

<asp:ScriptManager runat="server"></asp:ScriptManager>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

<ContentTemplate>

<asp:TextBox ID="LoginId" runat="server" OnTextChanged="LoginId\_TextChanged" AutoPostBack="True" AutoCompleteType="Disabled"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator7" runat="server" ControlToValidate="LoginId" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</ContentTemplate>

</asp:UpdatePanel>

<asp:Label ID="lblUserAvailabel" runat="server"></asp:Label>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Password</td>

<td style="height: 26px">

<asp:TextBox ID="Password" runat="server" TextMode="Password" AutoCompleteType="Disabled"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator8" runat="server" ControlToValidate="Password" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Confirm Password</td>

<td style="height: 26px">

<asp:TextBox ID="confirmPassword" runat="server" TextMode="Password"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator9" runat="server" ControlToValidate="FirstName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

<br />

<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="Password" ControlToValidate="confirmPassword" ErrorMessage="password does not match" ForeColor="#CC0066" SetFocusOnError="True"></asp:CompareValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">E-mail</td>

<td style="height: 26px">

<asp:TextBox ID="Email" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="Email" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

<br />

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="Email" ErrorMessage="Email should be in correct formate" ForeColor="Red" SetFocusOnError="True" ValidationExpression="\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Phone No.</td>

<td style="height: 26px">

<asp:TextBox ID="phone" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator3" runat="server" ControlToValidate="phone" ErrorMessage="Please enter valid Phone No." ForeColor="#FF3399" SetFocusOnError="True" ValidationExpression="\d{10}"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">DOB</td>

<td style="height: 26px">

<asp:TextBox ID="dob" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ControlToValidate="dob" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Program</td>

<td style="height: 26px">

<asp:DropDownList ID="DropDownListProgram" runat="server" DataSourceID="SqlDataSource3" DataTextField="ProgramName" DataValueField="ProgramName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT DISTINCT [ProgramName] FROM [programDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Address</td>

<td style="height: 26px">

<asp:TextBox ID="address" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server" ControlToValidate="address" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 24px;">Gender</td>

<td style="height: 24px">

<asp:RadioButtonList ID="RadioButtonListGender" runat="server" RepeatDirection="Horizontal">

<asp:ListItem>Male</asp:ListItem>

<asp:ListItem>Female</asp:ListItem>

</asp:RadioButtonList>

</td>

</tr>

<tr>

<td style="width: 146px; height: 24px;">State</td>

<td style="height: 24px">

<asp:DropDownList ID="DropDownListState" runat="server" DataSourceID="SqlDataSource1" DataTextField="StateName" DataValueField="StateName" AutoPostBack="True">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [StateName] FROM [StateDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px">City</td>

<td>

<asp:DropDownList ID="DropDownListCity" runat="server" DataSourceID="SqlDataSource2" DataTextField="CityName" DataValueField="CityName">

<asp:ListItem Value="0">Select City</asp:ListItem>

<asp:ListItem Value="1">Laxmi Nagar</asp:ListItem>

<asp:ListItem Value="2">Nirman Vihar</asp:ListItem>

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [CityName] FROM [CityDetails] WHERE ([StateID] = @StateID)">

<SelectParameters>

<asp:ControlParameter ControlID="DropDownListState" Name="StateID" PropertyName="SelectedValue" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px">Pin Code</td>

<td>

<asp:TextBox ID="PinCode" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ControlToValidate="PinCode" ErrorMessage="please enter valid pin code" ForeColor="#FF3399" SetFocusOnError="True" ValidationExpression="\d{6}"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px">

<asp:Button ID="ButtonReset" runat="server" Text="Reset" Height="28px" OnClick="ButtonReset\_Click" />

</td>

<td>

<asp:Button ID="ButtonSubmit" runat="server" Text="Submit" OnClick="ButtonSubmit\_Click" />

&nbsp;

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

</table>

</asp:Content>

**RegistrationPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination

{

public partial class RegistrationPage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void ButtonReset\_Click(object sender, EventArgs e)

{

Response.Redirect("RegistrationPage.aspx");

}

protected void ButtonSubmit\_Click(object sender, EventArgs e)

{

if (CheckUserName())

{

return;

}

try

{

DateTime now = DateTime.Now;

string date = now.ToString();

string \_ProcName = "registration";

SqlParameter[] \_parameter = {

new SqlParameter("@FirstName",FirstName.Text),

new SqlParameter("@LastName",LastName.Text),

new SqlParameter("@LoginID", LoginId.Text),

new SqlParameter("@Password",Password.Text),

new SqlParameter("@Email",Email.Text),

new SqlParameter("@Phone",Convert.ToInt64(phone.Text)),

new SqlParameter("@Gender",RadioButtonListGender.SelectedValue),

new SqlParameter("@DOB",dob.Text),

new SqlParameter("@Program",DropDownListProgram.SelectedValue.ToString()),

new SqlParameter("@Address",address.Text),

new SqlParameter("@StateID",DropDownListState.SelectedValue.ToString()),

new SqlParameter("@cityID",DropDownListCity.SelectedValue.ToString()),

new SqlParameter("@PinCoad",Convert.ToInt64(PinCode.Text))

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

Response.Redirect("RegistrationConfirmPage.aspx");

}

private bool CheckUserName()

{

bool flag = false;

try

{

string \_ProcName = "CheckUsername";

SqlParameter[] \_parameter = {

new SqlParameter("@LoginId",LoginId.Text)

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

return flag = true;

}

dr.Close();

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

return flag;

}

protected void LoginId\_TextChanged(object sender, EventArgs e)

{

if (CheckUserName())

{

lblUserAvailabel.ForeColor = System.Drawing.Color.Red;

lblUserAvailabel.Text = "Username already exist";

}

else

{

lblUserAvailabel.ForeColor = System.Drawing.Color.Green;

lblUserAvailabel.Text = "Username Availabel";

}

}

}

}

**RegistrationConfirmPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.Master" AutoEventWireup="true" CodeBehind="RegistrationConfirmPage.aspx.cs" Inherits="OnDemandExamination.RegistrationConfirmPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p>

<br />

<br />

<br />

<br />

<br />

<br />

<br />

<br />

<h1 class="center"> Registration Complete</h1>

<p class="center">

<asp:LinkButton ID="LinkButton1" runat="server" ForeColor="Blue" OnClick="LinkButton1\_Click">Log in now</asp:LinkButton>

</p>

<br />

<br />

<br />

<br /><br /><br /><br /><br /><br />

</p>

</asp:Content>

**RegistrationConfirmPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination

{

public partial class RegistrationConfirmPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void LinkButton1\_Click(object sender, EventArgs e)

{

Response.Redirect("LogInPage.aspx");

}

}

}

**User Coding**

**UserMasterPage.Master**

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="UserMasterPage.master.cs" Inherits="OnDemandExamination.User.UserMasterPage" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title></title>

<link rel="stylesheet" href="style.css" type="text/css" media="screen" />

<asp:ContentPlaceHolder ID="ContentPlaceHolder3" runat="server">

</asp:ContentPlaceHolder>

</head>

<body>

<form id="form1" runat="server">

<div>

<div id="container">

<div id="sitename" style="color:darkcyan">

<div class="right">

<asp:Label ID="lbluserName" runat="server"></asp:Label>

</div>

<h1 class="center">ON DEMAND EXAMINATION</h1>

<h2>A Chance To Learn Is A Chance To Grow And You Are Growing.............</h2>

</div>

<div id="mainmenu">

<ul>

<li><a href="UserHomePage.aspx">Home</a></li>

<li><a href="ProfileUserPage.aspx">My Profile</a></li>

<li> <asp:LinkButton ID="LinkButton1" runat="server" OnClick="LinkButton1\_Click">Logout</asp:LinkButton></li>

</ul>

</div>

<div id="wrap">

<div id="leftside" >

<p>

<span class="hide"> &nbsp;| </span>

<a class="nav" href="ApplyExamPage.aspx">Apply for Exam</a><span class="hide"> | </span>

<a class="nav" href="TakeExam.aspx">Exam</a><span class="hide"> | </span>

<a class="nav sub" href="CheckResultPage.aspx">Check Result</a><span class="hide"> | </span>

<a class="nav sub" href="CertificatePage.aspx">Certificate</a><span class="hide"> | </span>

</div>

<div id="rightside">

<asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">

</asp:ContentPlaceHolder>

</div>

<div id="content">

<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

<p>

&nbsp;</p>

</asp:ContentPlaceHolder>

</div>

<div class="clearingdiv">&nbsp;</div>

</div>

</div>

<div id="footer">© 2019 On Demand Examination<br />

Design by Vikram<br />

Enrollment No.-167177089

<br />

&nbsp;&nbsp;

</div>

</form>

</body>

</html>

**UserMasterPage.Master.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.User

{

public partial class UserMasterPage : System.Web.UI.MasterPage

{

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["user"] != null)

{

lbluserName.Text = Session["user"].ToString();

}

}

protected void LinkButton1\_Click(object sender, EventArgs e)

{

Session["user"] = null;

Session.Abandon();

Response.Redirect("~/LogInPage.aspx");

}

}

}

**UserHomePage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="UserHomePage.aspx.cs" Inherits="OnDemandExamination.User.AdminHomePage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p style="margin: 0 0 450px;" class="center">

<asp:Image ID="Image1" runat="server" Height="378px" ImageUrl="~/User/img/exam5.jpg" Width="784px" />

</p>

</asp:Content>

<asp:Content ID="Content2" runat="server" contentplaceholderid="ContentPlaceHolder2">

<p>

</p>

</asp:Content>

**UserHomePage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.User

{

public partial class AdminHomePage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**ProfileUserPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="ProfileUserPage.aspx.cs" Inherits="OnDemandExamination.User.ProfileUserPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

<h1>Your Profile</h1>

<p>

<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False" DataSourceID="SqlDataSource1" Height="50px" Width="125px">

<Fields>

<asp:BoundField DataField="FirstName" HeaderText="FirstName" SortExpression="FirstName" />

<asp:BoundField DataField="LastName" HeaderText="LastName" SortExpression="LastName" />

<asp:BoundField DataField="LoginID" HeaderText="LoginID" SortExpression="LoginID" />

<asp:BoundField DataField="Email" HeaderText="Email" SortExpression="Email" />

<asp:BoundField DataField="Phone" HeaderText="Phone" SortExpression="Phone" />

<asp:BoundField DataField="Gender" HeaderText="Gender" SortExpression="Gender" />

<asp:BoundField DataField="DOB" HeaderText="DOB" SortExpression="DOB" />

<asp:BoundField DataField="Program" HeaderText="Program" SortExpression="Program" />

<asp:BoundField DataField="Address" HeaderText="Address" SortExpression="Address" />

<asp:BoundField DataField="StateID" HeaderText="StateID" SortExpression="StateID" />

<asp:BoundField DataField="cityID" HeaderText="cityID" SortExpression="cityID" />

<asp:BoundField DataField="PinCoad" HeaderText="PinCoad" SortExpression="PinCoad" />

</Fields>

</asp:DetailsView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [FirstName], [LastName], [LoginID], [Email], [Phone], [Gender], [DOB], [Program], [Address], [StateID], [cityID], [PinCoad] FROM [UserDetials] WHERE ([LoginID] = @LoginID)">

<SelectParameters>

<asp:SessionParameter Name="LoginID" SessionField="user" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</p>

<p>

<asp:LinkButton ID="LinkButtonEditProfile" runat="server" ForeColor="#0000CC" OnClick="LinkButtonEditProfile\_Click">Edit Profile</asp:LinkButton>

</p>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h3 class="center">Your Profile</h3>

<p style="margin: 0 0 450px;" class="center">

<asp:Image ID="Image1" runat="server" Height="378px" ImageUrl="~/User/img/exam5.jpg" Width="784px" />

</p>

</asp:Content>

**ProfileUserPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.User

{

public partial class ProfileUserPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void LinkButtonEditProfile\_Click(object sender, EventArgs e)

{

Response.Redirect("EditProfile.aspx");

}

}

}

**EditProfile.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="EditProfile.aspx.cs" Inherits="OnDemandExamination.User.EditProfile" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False" DataSourceID="SqlDataSource4" Height="50px" Width="125px">

<Fields>

<asp:BoundField DataField="FirstName" HeaderText="FirstName" SortExpression="FirstName" />

<asp:BoundField DataField="LastName" HeaderText="LastName" SortExpression="LastName" />

<asp:BoundField DataField="LoginID" HeaderText="LoginID" SortExpression="LoginID" />

<asp:BoundField DataField="Email" HeaderText="Email" SortExpression="Email" />

<asp:BoundField DataField="Phone" HeaderText="Phone" SortExpression="Phone" />

<asp:BoundField DataField="Gender" HeaderText="Gender" SortExpression="Gender" />

<asp:BoundField DataField="DOB" HeaderText="DOB" SortExpression="DOB" />

<asp:BoundField DataField="Program" HeaderText="Program" SortExpression="Program" />

<asp:BoundField DataField="Address" HeaderText="Address" SortExpression="Address" />

<asp:BoundField DataField="StateID" HeaderText="StateID" SortExpression="StateID" />

<asp:BoundField DataField="cityID" HeaderText="cityID" SortExpression="cityID" />

<asp:BoundField DataField="PinCoad" HeaderText="PinCoad" SortExpression="PinCoad" />

</Fields>

</asp:DetailsView>

<asp:SqlDataSource ID="SqlDataSource4" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [FirstName], [LastName], [LoginID], [Email], [Phone], [Gender], [DOB], [Program], [Address], [StateID], [cityID], [PinCoad] FROM [UserDetials] WHERE ([LoginID] = @LoginID)">

<SelectParameters>

<asp:SessionParameter Name="LoginID" SessionField="user" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Edit Your Profile</h1>

<table id="RegstrationTable" style="border-width: thick; border-color: #000000; width:100%; padding: 3px; font-weight: bold;">

<tr>

<td style="width: 146px">&nbsp;</td>

</tr>

<tr>

<td style="width: 146px">First Name</td>

<td>

<asp:TextBox ID="FirstName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="FirstName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Last Name</td>

<td style="height: 26px">

<asp:TextBox ID="LastName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="LastName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Login ID</td>

<td style="height: 26px">

<asp:TextBox ID="LoginId" runat="server" OnTextChanged="LoginId\_TextChanged"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">E-mail <td style="height: 26px">

<asp:TextBox ID="Email" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="Email" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

<br />

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="Email" ErrorMessage="Email should be in correct formate" ForeColor="Red" SetFocusOnError="True" ValidationExpression="\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Phone No.</td>

<td style="height: 26px">

<asp:TextBox ID="phone" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator3" runat="server" ControlToValidate="phone" ErrorMessage="Please enter valid Phone No." ForeColor="#FF3399" SetFocusOnError="True" ValidationExpression="\d{10}"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">DOB</td>

<td style="height: 26px">

<asp:TextBox ID="dob" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ControlToValidate="dob" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Program</td>

<td style="height: 26px">

<asp:DropDownList ID="DropDownListProgram" runat="server" DataSourceID="SqlDataSource3" DataTextField="Program" DataValueField="Program">

<asp:ListItem>select</asp:ListItem>

<asp:ListItem>MCA</asp:ListItem>

<asp:ListItem>MBA</asp:ListItem>

<asp:ListItem>BA</asp:ListItem>

<asp:ListItem Value="BCA">BCA</asp:ListItem>

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [Program] FROM [UserDetials] WHERE ([LoginID] = @LoginID)">

<SelectParameters>

<asp:SessionParameter Name="LoginID" SessionField="user" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Address</td>

<td style="height: 26px">

<asp:TextBox ID="address" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server" ControlToValidate="address" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 24px;">Gender</td>

<td style="height: 24px">

<asp:RadioButtonList ID="RadioButtonListGender" runat="server" RepeatDirection="Horizontal">

<asp:ListItem>Male</asp:ListItem>

<asp:ListItem>Female</asp:ListItem>

</asp:RadioButtonList>

</td>

</tr>

<tr>

<td style="width: 146px; height: 24px;">State</td>

<td style="height: 24px">

<asp:DropDownList ID="DropDownListState" runat="server" DataSourceID="SqlDataSource1" DataTextField="StateID" DataValueField="StateID">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>"

ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>"

SelectCommand="SELECT [StateID] FROM [UserDetials] WHERE ([LoginID] = @LoginID)">

<SelectParameters>

<asp:SessionParameter Name="LoginID" SessionField="user" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px">City</td>

<td>

<asp:DropDownList ID="DropDownListCity" runat="server" DataSourceID="SqlDataSource2" DataTextField="CityID" DataValueField="CityID">

<asp:ListItem Value="0">Select City</asp:ListItem>

<asp:ListItem Value="1">Laxmi Nagar</asp:ListItem>

<asp:ListItem Value="2">Nirman Vihar</asp:ListItem>

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [cityID] FROM [UserDetials] WHERE ([LoginID] = @LoginID)">

<SelectParameters>

<asp:SessionParameter Name="LoginID" SessionField="user" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px">Pin Code</td>

<td>

<asp:TextBox ID="PinCode" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ControlToValidate="PinCode" ErrorMessage="please enter valid pin code" ForeColor="#FF3399" SetFocusOnError="True" ValidationExpression="\d{6}"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px">

<asp:Button ID="ButtonReset" runat="server" Height="28px" OnClick="ButtonReset\_Click" Text="Reset" />

</td>

<td>

<asp:Button ID="ButtonSubmit" runat="server" OnClick="ButtonSubmit\_Click" Text="Update" />

&nbsp;

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

</table>

</asp:Content>

**EditProfile.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.User

{

public partial class EditProfile : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

CustomerProfile();

}

}

protected void ButtonSubmit\_Click(object sender, EventArgs e)

{

try

{

string \_ProcName = "updateUserDetails";

SqlParameter[] \_parameter = {

new SqlParameter("@FirstName",FirstName.Text),

new SqlParameter("@LastName",LastName.Text),

new SqlParameter("@LoginID",Session["user"].ToString()),

new SqlParameter("@Email",Email.Text),

new SqlParameter("@Phone",Convert.ToInt64(phone.Text)),

new SqlParameter("@Gender",RadioButtonListGender.SelectedValue),

new SqlParameter("@DOB",dob.Text),

new SqlParameter("@Program",DropDownListProgram.SelectedValue.ToString()),

new SqlParameter("@Address",address.Text),

new SqlParameter("@StateID",DropDownListState.SelectedValue.ToString()),

new SqlParameter("@cityID",DropDownListCity.SelectedValue.ToString()),

new SqlParameter("@PinCoad",Convert.ToInt64(PinCode.Text))

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

public void CustomerProfile()

{

try

{

string \_ProcName = "getUserDeatils";

SqlParameter[] \_parameter = {

new SqlParameter("@LoginID",Session["user"].ToString())

};

DataTable dbtable = new DataTable();

dbtable = db.GetDataTable(\_ProcName, \_parameter);

if (dbtable.Rows.Count > 0)

{

FirstName.Text = dbtable.Rows[0]["FirstName"].ToString();

LastName.Text = dbtable.Rows[0]["LastName"].ToString();

LoginId.Text = dbtable.Rows[0]["LoginID"].ToString();

Email.Text = dbtable.Rows[0]["Email"].ToString();

phone.Text = dbtable.Rows[0]["phone"].ToString();

RadioButtonListGender.SelectedValue = dbtable.Rows[0]["Gender"].ToString();

dob.Text = dbtable.Rows[0]["DOB"].ToString();

address.Text = dbtable.Rows[0]["address"].ToString();

PinCode.Text = dbtable.Rows[0]["PinCoad"].ToString();

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

protected void ButtonReset\_Click(object sender, EventArgs e)

{

}

protected void LoginId\_TextChanged(object sender, EventArgs e)

{

}

}

}

**ApplyExamPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="ApplyExamPage.aspx.cs" Inherits="OnDemandExamination.User.ApplyExamPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1 class="center">Apply for Exam</h1>

<table style="width: 100%">

<tr>

<td>Select Program</td>

<td>

<asp:DropDownList ID="DropDownListProgram" runat="server" AutoPostBack="True" DataSourceID="SqlDataSource1" DataTextField="ProgramName" DataValueField="ProgramName">

<asp:ListItem Selected="True">Select</asp:ListItem>

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ProgramName] FROM [programDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td>Select Course</td>

<td>

<asp:DropDownList ID="DropDownListCourse" runat="server" DataSourceID="SqlDataSource2" DataTextField="CourseName" DataValueField="CourseName">

<asp:ListItem Selected="True">Select</asp:ListItem>

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [CourseName] FROM [CourseDeatils] WHERE ([ProgramName] = @ProgramName)">

<SelectParameters>

<asp:ControlParameter ControlID="DropDownListProgram" Name="ProgramName" PropertyName="SelectedValue" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

<asp:DropDownList ID="DropDownListCourse2" runat="server" DataSourceID="SqlDataSource2" DataTextField="CourseName" DataValueField="CourseName">

<asp:ListItem Selected="True">Select</asp:ListItem>

</asp:DropDownList>

<asp:DropDownList ID="DropDownListCourse3" runat="server" DataSourceID="SqlDataSource2" DataTextField="CourseName" DataValueField="CourseName">

</asp:DropDownList>

</td>

</tr>

<tr>

<td>

<asp:Button ID="Submit" runat="server" OnClick="Submit\_Click" Text="Submit" />

</td>

<td>

<asp:Label ID="LabelErrorMessage" runat="server"></asp:Label>

</td>

</tr>

</table>

</asp:Content>

**ApplyExamPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.User

{

public partial class ApplyExamPage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Submit\_Click(object sender, EventArgs e)

{

try

{

string \_ProcName = "ApplyExam";

SqlParameter[] \_parameter = {

new SqlParameter("@UserName",Session["user"].ToString()),

new SqlParameter("@ProgramName",DropDownListProgram.SelectedItem.ToString()),

new SqlParameter("@CourseName1",DropDownListCourse.SelectedValue.ToString()),

new SqlParameter("@CourseName2",DropDownListCourse2.SelectedValue.ToString()),

new SqlParameter("@CourseName3",DropDownListCourse3.SelectedValue.ToString())

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "successfull";

Response.Redirect("ExamRequestConfirmPage.aspx");

}

else

{

LabelErrorMessage.Text = "Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

}

}

**ExamRequestConfirmPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="ExamRequestConfirmPage.aspx.cs" Inherits="OnDemandExamination.User.ExamRequestConfirmPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder3" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p>

Request sent</p>

</asp:Content>

**ExamRequestConfirmPage.aspx**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.User

{

public partial class ExamRequestConfirmPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**TakeExam.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="TakeExam.aspx.cs" Inherits="OnDemandExamination.User.TakeExam" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder3" runat="server">

</style>

<style type="text/css">

.auto-style1 {

width: 100%;

}

.auto-style2 {

width: 139px;

}

</style>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div>

<h1> Your Requested Exams</h1>

<br />

<table class="auto-style1">

<tr>

<td class="auto-style2">

<asp:Label ID="Label1" runat="server"></asp:Label>

</td>

<td>

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click1" Text="Start Exam" />

</td>

</tr>

<tr>

<td class="auto-style2">

<asp:Label ID="Label2" runat="server"></asp:Label>

</td>

<td>

<asp:Button ID="Button2" runat="server" OnClick="Button2\_Click" Text="Start Exam" />

</td>

</tr>

<tr>

<td class="auto-style2">

<asp:Label ID="Label3" runat="server"></asp:Label>

</td>

<td>

<asp:Button ID="Button3" runat="server" OnClick="Button3\_Click" Text="Start Exam" />

</td>

</tr>

<tr>

<td class="auto-style2">&nbsp;</td>

<td>

<asp:Label ID="LabelErrorMessage" runat="server"></asp:Label>

</td>

</tr>

</table>

</div>

</asp:Content>

**TakeExam.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.User

{

public partial class TakeExam : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

int flag1,flag2,flag3;

protected void Page\_Load(object sender, EventArgs e)

{

Button1.Visible =false;

Button2.Visible = false;

Button3.Visible = false;

call();

checkExam();

}

public void call()

{

try

{

string \_ProcName = "getExamCourseDetails";

SqlParameter[] \_parameter = {

new SqlParameter("@LoginID",Session["user"].ToString())

};

DataTable dbtable = new DataTable();

dbtable = db.GetDataTable(\_ProcName, \_parameter);

if (dbtable.Rows.Count > 0)

{

Label1.Text = dbtable.Rows[0]["CourseName1"].ToString();

Label2.Text = dbtable.Rows[0]["CourseName2"].ToString();

Label3.Text = dbtable.Rows[0]["CourseName3"].ToString();

flag1=Convert.ToInt16(dbtable.Rows[0]["flag1"]);

flag2=Convert.ToInt16(dbtable.Rows[0]["flag2"]);

flag3=Convert.ToInt16(dbtable.Rows[0]["flag3"]);

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

protected void checkExam()

{

if(flag1==1)

{

Button1.Visible = true;

}

if (flag2 == 1)

{

Button1.Visible = true;

}

if (flag3 == 1)

{

Button1.Visible = true;

}

}

protected void Button1\_Click1(object sender, EventArgs e)

{

Session["course"] = Label1.Text;

Response.Redirect("EXAM.aspx");

}

protected void Button2\_Click(object sender, EventArgs e)

{

Session["course"] = Label2.Text;

Response.Redirect("EXAM.aspx");

}

protected void Button3\_Click(object sender, EventArgs e)

{

Session["course"] = Label3.Text;

Response.Redirect("EXAM.aspx");

}

}

}

**EXAM.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="EXAM.aspx.cs" Inherits="OnDemandExamination.User.EXAM" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<style type="text/css">

.style20

{

width: 802px;

}

.style21

{

width: 339px;

}

.style19

{

width: 400px;

}

.style15

{

width: 100%;

}

.style16

{

width: 69px;

}

.style17

{

width: 206px;

}

.auto-style1 {

width: 339px;

height: 17px;

}

.auto-style2 {

width: 400px;

height: 17px;

}

</style>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:ScriptManager ID="ScriptManager1" runat="server">

</asp:ScriptManager>

<asp:UpdatePanel ID="TimedPanel" runat="server" updatemode="Conditional">

<ContentTemplate>

<asp:Timer ID="Timer1" runat="server" Enabled="False" Interval="1000"

ontick="Timer1\_Tick">

</asp:Timer>

<table class="style20">

<tr>

<td align="left" class="auto-style1">

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:Label ID="lblnoofquestions" runat="server"></asp:Label>

</td>

<td align="left" class="auto-style2">

<asp:Label ID="clock" runat="server" Text=""></asp:Label>

</td>

</tr>

</table>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<asp:Panel ID="pnlQuest" runat="server"

Height="316px" style="margin-left: 40px;" Width="688px">

<asp:Label ID="lblQuestion" runat="server" ForeColor="#0066FF"></asp:Label>

<br />

<br />

<asp:RadioButton ID="RadioButton1" runat="server" GroupName="ans" />

<asp:CheckBox ID="CheckBox1" runat="server" />

<asp:Label ID="Option1" runat="server" Text=""></asp:Label>

<br />

<br />

<asp:RadioButton ID="RadioButton2" runat="server" GroupName="ans" />

<asp:CheckBox ID="CheckBox2" runat="server" />

<asp:Label ID="Option2" runat="server" Text=""></asp:Label>

<br />

<br />

<asp:RadioButton ID="RadioButton3" runat="server" GroupName="ans" />

<asp:CheckBox ID="CheckBox3" runat="server" />

<asp:Label ID="Option3" runat="server" Text=""></asp:Label>

<br />

<br />

<asp:RadioButton ID="RadioButton4" runat="server" GroupName="ans" />

<asp:CheckBox ID="CheckBox4" runat="server" />

<asp:Label ID="Option4" runat="server" Text=""></asp:Label>

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<br />

<table class="style15">

<tr>

<td class="style16">

<asp:Button ID="ButtonNext" runat="server" OnClick="ButtonNext\_Click" Text="NEXT" />

</td>

<td class="style17">

<asp:Button ID="ButtonStart" runat="server" OnClick="ButtonStart\_Click" Text="START" />

</td>

</tr>

<tr>

<td class="style16">

&nbsp;</td>

<td align="left" class="style17">

&nbsp;</td>

</tr>

</table>

<br />

<asp:SqlDataSource ID="SqlDataSource2" runat="server"

ConnectionString="Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\DbOnlineTest.mdf;Integrated Security=True;User Instance=True"

ProviderName="System.Data.SqlClient"

SelectCommand="SELECT \* FROM [tblQuestions]"></asp:SqlDataSource>

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<br />

<br />

</asp:Panel>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

</ContentTemplate>

<Triggers>

<asp:AsyncPostBackTrigger controlid="Timer1" eventname="Tick" />

</Triggers>

</asp:UpdatePanel>

</div>

</form>

</body>

</html>

**EXAM.aspx.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.User

{

public partial class EXAM : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

int TotQus = 0;

static int count = 0;

static int viewradio = 0;

static float marks = 0.0f, correct = 0.0f;

int totalSeconds = 0, seconds = 60, minutes = 60;

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["connectionDb"].ConnectionString);

string selected, subject\_name, Result = null, UserName = null;

static String CrtAns = null;

string con\_str = @"Data Source=DELL\SQLEXPRESS;Initial Catalog=OnDemandExam;Integrated Security=True";

DataTable dt;

DataSet ds;

SqlDataAdapter ad;

static int rowindex = -1;

protected void Page\_Load(object sender, EventArgs e)

{

subject\_name = Session["course"].ToString();

UserName = Session["user"].ToString();

string noofqns = "SELECT [NoOfQuestion] FROM [ExamSetting]";

SqlCommand cmd = new SqlCommand(noofqns, con);

con.Open();

TotQus = (int)cmd.ExecuteScalar();

String cmd11 = "SELECT TOP " + TotQus + " [QuestionID], [Course], [Question], [OptionA], [OptionB], [OptionC], [OptionD], [CorrectOption] FROM [QuestionDetails] where [Course]='" + subject\_name + "'order by newid()";

ds = new DataSet();

ad = new SqlDataAdapter(cmd11, con);

ad.Fill(ds, "QuestionDetails");

dt = ds.Tables["QuestionDetails"];

con.Close();

if (viewradio == 0)

{

RadioInvisible();

}

else

{

RadioVisible();

}

viewradio += 1;

CheckBoxInvisible();

}

protected void Timer1\_Tick(object sender, EventArgs e)

{

Session["time"] = Convert.ToInt16(Session["time"]) - 1;

if (Convert.ToInt16(Session["time"]) <= 0)

{

clock.Text = "TimeOut!";

ButtonNext.Visible = false;

RadioInvisible();

}

else

{

totalSeconds = Convert.ToInt16(Session["time"]);

seconds = totalSeconds % 60;

minutes = totalSeconds / 60;

clock.Text = minutes + ":" + seconds;

}

}

protected void ButtonNext\_Click(object sender, EventArgs e)

{

call();

}

protected void call()

{

rowindex = rowindex + 1;

selected = SelectedAnswer();

if (CrtAns == selected)

{

correct = correct + 1;

}

RadioUncheck();

CheckBoxUnchek();

DisplayQuestion();

}

protected void DisplayQuestion()

{

try

{

if (rowindex < TotQus)

{

count += 1;

lblnoofquestions.Text = count.ToString() + " of " + TotQus;

lblQuestion.Text = dt.Rows[rowindex]["Question"].ToString();

Option1.Text = dt.Rows[rowindex]["OptionA"].ToString();

Option2.Text = dt.Rows[rowindex]["OptionB"].ToString();

Option3.Text = dt.Rows[rowindex]["OptionC"].ToString();

Option4.Text = dt.Rows[rowindex]["OptionD"].ToString();

CrtAns = dt.Rows[rowindex]["CorrectOption"].ToString();

RadioVisible();

CheckBoxInvisible();

}

else

{

ButtonNext.Visible = false;

RadioInvisible();

CheckBoxInvisible();

marks = (float)(((float)(correct) / (float)(TotQus)) \* 100);

if (marks >= 50)

{

Result = "pass";

}

else

{

Result = "fail";

}

Report();

Response.Redirect("ExamFinishMessagePage.aspx");

}

}

catch (IndexOutOfRangeException)

{

Response.Write("Questions Over");

}

}

protected void ButtonStart\_Click(object sender, EventArgs e)

{

string query = "SELECT [TimeLimit] FROM [ExamSetting]";

SqlCommand cmd = new SqlCommand(query, con);

con.Open();

int timeget = (int)cmd.ExecuteScalar();

totalSeconds = timeget \* 60;

Timer1.Enabled = true;

Session["time"] = totalSeconds;

rowindex = rowindex + 1;

DisplayQuestion();

ButtonStart.Visible = false;

ButtonNext.Visible = true;

con.Close();

}

protected void Report()

{

try

{

string \_ProcName = "ResultSet";

SqlParameter[] \_parameter = {

new SqlParameter("@UserName",UserName),

new SqlParameter("@CourseName", subject\_name),

new SqlParameter("@NoOfQ",TotQus),

new SqlParameter("@Marks",marks),

new SqlParameter("@Status",Result)

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

}

catch (Exception ex)

{

throw;

Response.Write("Try Again");

}

}

public string SelectedAnswer()

{

if (RadioButton1.Checked)

{

return "A";

}

else if (RadioButton2.Checked)

{

return "B";

}

else if (RadioButton3.Checked)

{

return "C";

}

else if (RadioButton4.Checked)

{

return "D";

}

else

{

return "nothing";

}

}

protected string ChoosedAnswer()

{

string choose = null;

if (CheckBox1.Checked)

{

choose = "A,";

}

if (CheckBox2.Checked)

{

choose += "B,";

}

if (CheckBox3.Checked)

{

choose += "C,";

}

if (CheckBox4.Checked)

{

choose += "D,";

}

return choose;

}

protected void RadioUncheck()

{

RadioButton1.Checked = false;

RadioButton2.Checked = false;

RadioButton3.Checked = false;

RadioButton4.Checked = false;

}

protected void CheckBoxUnchek()

{

CheckBox1.Checked = false;

CheckBox2.Checked = false;

CheckBox3.Checked = false;

CheckBox4.Checked = false;

}

protected void RadioInvisible()

{

RadioButton1.Visible = false;

RadioButton2.Visible = false;

RadioButton3.Visible = false;

RadioButton4.Visible = false;

}

protected void RadioVisible()

{

RadioButton1.Visible = true;

RadioButton2.Visible = true;

RadioButton3.Visible = true;

RadioButton4.Visible = true;

}

protected void CheckBoxInvisible()

{

CheckBox1.Visible = false;

CheckBox2.Visible = false;

CheckBox3.Visible = false;

CheckBox4.Visible = false;

}

protected void CheckBoxVisible()

{

CheckBox1.Visible = true;

CheckBox2.Visible = true;

CheckBox3.Visible = true;

CheckBox4.Visible = true;

}

}

}

**ExamFinishMessagePage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="ExamFinishMessagePage.aspx.cs" Inherits="OnDemandExamination.User.ExamFinishMessagePage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder3" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p>

<h1 class="center"> Exam Finished</h1></p>

</asp:Content>

**ExamFinishMessagePage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.User

{

public partial class ExamFinishMessagePage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**CheckResultPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/User/UserMasterPage.Master" AutoEventWireup="true" CodeBehind="CheckResultPage.aspx.cs" Inherits="OnDemandExamination.User.CheckResultPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder3" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p>

<h1 class="center">Exam Results</h1>

</p>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" CellPadding="4" DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None" Height="114px" Width="519px">

<AlternatingRowStyle BackColor="White" />

<Columns>

<asp:BoundField DataField="ExamDate" HeaderText="ExamDate" SortExpression="ExamDate" />

<asp:BoundField DataField="CourseName" HeaderText="CourseName" SortExpression="CourseName" />

<asp:BoundField DataField="Marks" HeaderText="Marks" SortExpression="Marks" />

<asp:BoundField DataField="Status" HeaderText="Status" SortExpression="Status" />

</Columns>

<EditRowStyle BackColor="#2461BF" />

<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />

<RowStyle BackColor="#EFF3FB" />

<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

<SortedAscendingCellStyle BackColor="#F5F7FB" />

<SortedAscendingHeaderStyle BackColor="#6D95E1" />

<SortedDescendingCellStyle BackColor="#E9EBEF" />

<SortedDescendingHeaderStyle BackColor="#4870BE" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ExamDate], [CourseName], [Marks], [Status] FROM [ResultDetails] WHERE (([UserName] = @UserName) AND ([flag] = @flag))">

<SelectParameters>

<asp:SessionParameter Name="UserName" SessionField="user" Type="String" />

<asp:Parameter DefaultValue="1" Name="flag" Type="Int32" />

</SelectParameters>

</asp:SqlDataSource>

</asp:Content>

**CheckResultPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.User

{

public partial class CheckResultPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**CertificatePage.aspx**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="CertificatePage.aspx.cs" Inherits="OnDemandExamination.User.CertificatePage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<style type="text/css">

.auto-style1 {

text-align: center;

}

.auto-style2 {

width: 100%;

}

.auto-style3 {

width: 395px;

}

.auto-style6 {

width: 170px;

}

.auto-style9 {

width: 170px;

height: 23px;

}

.auto-style10 {

height: 23px;

width: 249px;

}

.auto-style11 {

width: 249px;

}

.auto-style14 {

width: 115px;

}

.auto-style15 {

height: 23px;

width: 115px;

}

.auto-style16 {

width: 103px;

}

.auto-style17 {

height: 23px;

width: 103px;

}

</style>

</head>

<body>

<form id="form1" runat="server">

<div>

<h1 class="auto-style1"> CERTIFICATE</h1><br />

<table class="auto-style2">

<tr>

<td class="auto-style14">Student ID</td>

<td class="auto-style16">

<asp:Label ID="labelStudentID" runat="server"></asp:Label>

</td>

<td colspan="2">&nbsp;</td>

</tr>

<tr>

<td class="auto-style14">Name</td>

<td class="auto-style16">

<asp:Label ID="labelName" runat="server"></asp:Label>

</td>

<td colspan="2">&nbsp;</td>

</tr>

<tr>

<td class="auto-style14">DOB</td>

<td class="auto-style16">

<asp:Label ID="labelDOB" runat="server"></asp:Label>

</td>

<td colspan="2">&nbsp;</td>

</tr>

<tr>

<td class="auto-style14">&nbsp;</td>

<td class="auto-style16">&nbsp;</td>

<td colspan="2">&nbsp;</td>

</tr>

<tr>

<td class="auto-style3" colspan="2">RESULT </td>

<td colspan="2">&nbsp;</td>

</tr>

<tr>

<td class="auto-style3" colspan="2">&nbsp;</td>

<td colspan="2">&nbsp;</td>

</tr>

<tr>

<td class="auto-style14">Course</td>

<td class="auto-style16">Marks</td>

<td class="auto-style6">Obtain Marks</td>

<td class="auto-style11">Result</td>

</tr>

<tr>

<td class="auto-style14">&nbsp;</td>

<td class="auto-style16">&nbsp;</td>

<td class="auto-style6">&nbsp;</td>

<td class="auto-style11">&nbsp;</td>

</tr>

<tr>

<td class="auto-style15">

<asp:Label ID="labelCourse" runat="server"></asp:Label>

</td>

<td class="auto-style17">100</td>

<td class="auto-style9">

<asp:Label ID="labelmarksObtain" runat="server"></asp:Label>

</td>

<td class="auto-style10">

<asp:Label ID="labelResult" runat="server"></asp:Label>

</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

&nbsp;</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

&nbsp;</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

&nbsp;</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

<asp:Image ID="Image1" runat="server" Height="58px" ImageUrl="~/User/img/signature.jpg" Width="201px" />

</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

Signature</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

&nbsp;</td>

</tr>

<tr>

<td class="auto-style15">

&nbsp;</td>

<td class="auto-style17">&nbsp;</td>

<td class="auto-style9">

&nbsp;</td>

<td class="auto-style10">

&nbsp;</td>

</tr>

</table>

</div>

</form>

</body>

</html>

**CertificatePage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.User

{

public partial class CertificatePage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

getDetails();

}

protected void getDetails()

{

try

{

string \_ProcName = "getResultDetails";

SqlParameter[] \_parameter = {

new SqlParameter("@LoginID",Session["user"].ToString())

};

DataTable dbtable = new DataTable();

dbtable = db.GetDataTable(\_ProcName, \_parameter);

if (dbtable.Rows.Count > 0)

{

labelCourse.Text = dbtable.Rows[0]["CourseName"].ToString();

labelmarksObtain.Text = dbtable.Rows[0]["Marks"].ToString();

labelResult.Text = dbtable.Rows[0]["Status"].ToString();

labelStudentID.Text = dbtable.Rows[0]["UserName"].ToString();

labelName.Text = (dbtable.Rows[0]["FirstName"].ToString())+" "+( dbtable.Rows[0]["LastName"].ToString());

labelDOB.Text = dbtable.Rows[0]["DOB"].ToString();

}

}

catch (Exception)

{

}

}

}

}

**Admin Coding**

**AdminMasterPage.Master**

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="AdminMasterPage.master.cs" Inherits="OnDemandExamination.Admin.AdminMasterPage" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title></title>

<link rel="stylesheet" href="style.css" type="text/css" media="screen" />

<asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server"></asp:ContentPlaceHolder>

</head>

<body>

<form id="form1" runat="server">

<div>

<div id="container">

<div id="sitename" style="color:darkcyan">

<div class="right">

<asp:Label ID="lbluserName" runat="server"></asp:Label>

</div>

<h1 class="center">ON DEMAND EXAMINATION</h1>

<h2>A Chance To Learn Is A Chance To Grow And You Are Growing.............</h2>

</div>

<div id="mainmenu">

<ul>

<li><a href="AdminHomePage.aspx">Home</a></li>

<li>

<asp:LinkButton ID="LinkButton1" runat="server" OnClick="LinkButton1\_Click">Logout</asp:LinkButton></li>

</ul>

</div>

<div id="wrap">

<div id="leftside">

<p><a class="nav" href="ManageProgramPage.aspx">Manage Program</a><span class="hide"> | </span>

<a class="nav" href="ManageCoursePage.aspx">Manage Course</a><span class="hide"> | </span>

<a class="nav" href="ManageStatePage.aspx">Manage State</a><span class="hide"> | </span>

<a class="nav" href="ManageCityPage.aspx">Manage City</a><span class="hide"> | </span>

<a class="nav" href="ManageQuestion.aspx">Manage Question</a><span class="hide"> | </span>

<a class="nav sub" href="StudentExamRequest.aspx">Student Exam Request</a><span class="hide"> | </span>

<a class="nav sub" href="StudentListPage.aspx">Student List</a><span class="hide"> | </span>

<a class="nav sub" href="StudentResultPage.aspx">Student Results</a><span class="hide"> | </span>

<a class="nav sub" href="ExamSettingPage.aspx">Exam Setting</a></p>

<a class="nav" href="AddTeachersPage.aspx">ADD TEACHERS</a><span class="hide"> | </span>

<ul class="linklist">

</ul>

</div>

<div id="rightside">

</div>

<div id="content">

<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

</asp:ContentPlaceHolder>

</div>

<div class="clearingdiv">&nbsp;</div>

</div>

</div>

<div id="footer">© 2019 On Demand Examination<br />

Design by Vikram<br />

Enrollment No.-167177089

<br />

&nbsp;&nbsp;

</div>

</form>

</body>

</html>

**AdminMasterPage.Master.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class AdminMasterPage : System.Web.UI.MasterPage

{

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["admin"] != null)

{

lbluserName.Text = Session["admin"].ToString();

}

}

protected void LinkButton1\_Click(object sender, EventArgs e)

{

Session["admin"] = null;

Session.Abandon();

Response.Redirect("~/LogInPage.aspx");

}

}

}

**AdminHomePage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="AdminHomePage.aspx.cs" Inherits="OnDemandExamination.Admin.AdminHomePage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p style="margin: 0 0 450px;" class="center">

<asp:Image ID="Image1" runat="server" Height="378px" ImageUrl="~/Admin/img/exam5.jpg" Width="784px" />

</p>

&nbsp;

</asp:Content>

**AdminHomePage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class AdminHomePage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**ManageProgramPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="ManageProgramPage.aspx.cs" Inherits="OnDemandExamination.Admin.ManageProgramPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Manage Program</h1>

<table style="width: 23%; height: 59px;">

<tr>

<td style="width: 229px">Program Name</td>

<td style="width: 313px">

<asp:TextBox ID="textBoxProgramName" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 229px">Duration</td>

<td style="width: 313px">

<asp:TextBox ID="textBoxDuration" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 229px">

<asp:Button ID="buttonAdd" runat="server" OnClick="buttonAdd\_Click" Text="Add" />

</td>

<td style="width: 313px">

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="Red"></asp:Label>

</td>

</tr>

</table>

<br />

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="ProgramID" DataSourceID="SqlDataSource1">

<Columns>

<asp:BoundField DataField="ProgramID" HeaderText="ProgramID" InsertVisible="False" ReadOnly="True" SortExpression="ProgramID" />

<asp:BoundField DataField="ProgramName" HeaderText="ProgramName" SortExpression="ProgramName" />

<asp:BoundField DataField="Duration" HeaderText="Duration" SortExpression="Duration" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ProgramID], [ProgramName], [Duration] FROM [programDetails]"></asp:SqlDataSource>

</asp:Content>

**ManageProgramPage.aspx**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.Admin

{

public partial class ManageProgramPage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void buttonAdd\_Click(object sender, EventArgs e)

{

if (CheckProgram())

{

LabelErrorMessage.Text = ("already exit");

return;

}

try

{

string \_ProcName = "addProgram";

SqlParameter[] \_parameter = {

new SqlParameter("@ProgramName",textBoxProgramName.Text),

new SqlParameter("@Duration",textBoxDuration.Text)

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

private bool CheckProgram()

{

bool flag = false;

try

{

string \_ProcName = "checkProgram";

SqlParameter[] \_parameter = {

new SqlParameter("@ProgramName",textBoxProgramName.Text)

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

return flag = true;

}

dr.Close();

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

return flag;

}

}

}

**ManageCoursePage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="ManageCoursePage.aspx.cs" Inherits="OnDemandExamination.Admin.ManageCoursePage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Manage Course</h1>

<table style="width: 100%">

<tr>

<td style="width: 128px">Select Program</td>

<td>

<asp:DropDownList ID="DropDownListProgram" runat="server" DataSourceID="SqlDataSource1" DataTextField="ProgramName" DataValueField="ProgramName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ProgramName] FROM [programDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 128px">Enter course</td>

<td>

<asp:TextBox ID="textBoxCourse" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 128px">

<asp:Button ID="buttonAdd" runat="server" OnClick="Add\_Click" Text="add" style="width: 35px" />

</td>

<td>

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

<tr>

<td style="width: 128px">

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" DataSourceID="SqlDataSource2">

<Columns>

<asp:BoundField DataField="ProgramName" HeaderText="ProgramName" SortExpression="ProgramName" />

<asp:BoundField DataField="CourseName" HeaderText="CourseName" SortExpression="CourseName" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ProgramName], [CourseName] FROM [CourseDeatils]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 128px">

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

</table>

</asp:Content>

**ManageCoursePage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.Admin

{

public partial class ManageCoursePage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Add\_Click(object sender, EventArgs e)

{

if (CheckProgram())

{

LabelErrorMessage.Text = ("already exit");

return;

}

try

{

string \_ProcName = "addCourse";

SqlParameter[] \_parameter = {

new SqlParameter("@ProgramName",DropDownListProgram.SelectedItem.ToString()),

new SqlParameter("@CourseName",textBoxCourse.Text)

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

private bool CheckProgram()

{

bool flag = false;

try

{

string \_ProcName = "checkCourse";

SqlParameter[] \_parameter = {

new SqlParameter("@CourseName",textBoxCourse.Text),

new SqlParameter("@ProgramName",DropDownListProgram.SelectedValue.ToString())

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

return flag = true;

}

dr.Close();

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

return flag;

}

}

}

**ManageStatePage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="ManageStatePage.aspx.cs" Inherits="OnDemandExamination.Admin.ManageStatePage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Manage State</h1>

<table style="width: 60%">

<tr>

<td style="width: 127px">State Name</td>

<td>

<asp:TextBox ID="textBoxStateName" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td style="width: 127px">

<asp:Button ID="buttonAdd" runat="server" Text="Add" OnClick="buttonAdd\_Click" />

</td>

<td>

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

<tr>

<td style="width: 127px">

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" CellPadding="4" DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None">

<AlternatingRowStyle BackColor="White" />

<Columns>

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />

<asp:BoundField DataField="StateName" HeaderText="StateName" SortExpression="StateName" />

</Columns>

<EditRowStyle BackColor="#2461BF" />

<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />

<RowStyle BackColor="#EFF3FB" />

<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

<SortedAscendingCellStyle BackColor="#F5F7FB" />

<SortedAscendingHeaderStyle BackColor="#6D95E1" />

<SortedDescendingCellStyle BackColor="#E9EBEF" />

<SortedDescendingHeaderStyle BackColor="#4870BE" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" DeleteCommand="DELETE FROM [StateDetails] WHERE [StateID] = @StateID" InsertCommand="INSERT INTO [StateDetails] ([StateName]) VALUES (@StateName)" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [StateName], [StateID] FROM [StateDetails]" UpdateCommand="UPDATE [StateDetails] SET [StateName] = @StateName WHERE [StateID] = @StateID">

<DeleteParameters>

<asp:Parameter Name="StateID" Type="Int64" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="StateName" Type="String" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="StateName" Type="String" />

<asp:Parameter Name="StateID" Type="Int64" />

</UpdateParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 127px">

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

</table>

</asp:Content>

**ManageStatePage.aspx**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.Admin

{

public partial class ManageStatePage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void buttonAdd\_Click(object sender, EventArgs e)

{

if (checkState())

{

LabelErrorMessage.Text = ("already exit");

return;

}

try

{

string \_ProcName = "addState";

SqlParameter[] \_parameter = {

new SqlParameter("@StateName",textBoxStateName.Text)

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

private bool checkState()

{

bool flag = false;

try

{

string \_ProcName = "checkState";

SqlParameter[] \_parameter = {

new SqlParameter("@StateName",textBoxStateName.Text)

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

return flag = true;

}

dr.Close();

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

return flag;

}

}

}

**ManageCityPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="ManageCityPage.aspx.cs" Inherits="OnDemandExamination.Admin.ManageCityPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Manage City</h1>

<table style="width: 100%">

<tr>

<td>Select State</td>

<td>

<asp:DropDownList ID="DropDownListState" runat="server" DataSourceID="SqlDataSource1" DataTextField="StateName" DataValueField="StateName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [StateName], [StateID] FROM [StateDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td>Enter City Name</td>

<td>

<asp:TextBox ID="textBoxCityName" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>Pin Code</td>

<td>

<asp:TextBox ID="TextBoxPinCode" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>

<asp:Button ID="Add" runat="server" OnClick="Add\_Click" Text="Add" />

</td>

<td>

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

<tr>

<td>

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="CityID" DataSourceID="SqlDataSource2" Width="591px">

<Columns>

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />

<asp:BoundField DataField="CityID" HeaderText="CityID" InsertVisible="False" ReadOnly="True" SortExpression="CityID" />

<asp:BoundField DataField="StateID" HeaderText="StateID" SortExpression="StateID" />

<asp:BoundField DataField="CityName" HeaderText="CityName" SortExpression="CityName" />

<asp:BoundField DataField="PinCode" HeaderText="PinCode" SortExpression="PinCode" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConflictDetection="CompareAllValues" ConnectionString="<%$ ConnectionStrings:connectionDb %>" DeleteCommand="DELETE FROM [CityDetails] WHERE [CityID] = @original\_CityID AND (([StateID] = @original\_StateID) OR ([StateID] IS NULL AND @original\_StateID IS NULL)) AND (([CityName] = @original\_CityName) OR ([CityName] IS NULL AND @original\_CityName IS NULL)) AND (([PinCode] = @original\_PinCode) OR ([PinCode] IS NULL AND @original\_PinCode IS NULL))" InsertCommand="INSERT INTO [CityDetails] ([StateID], [CityName], [PinCode]) VALUES (@StateID, @CityName, @PinCode)" OldValuesParameterFormatString="original\_{0}" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [CityID], [StateID], [CityName], [PinCode] FROM [CityDetails]" UpdateCommand="UPDATE [CityDetails] SET [StateID] = @StateID, [CityName] = @CityName, [PinCode] = @PinCode WHERE [CityID] = @original\_CityID AND (([StateID] = @original\_StateID) OR ([StateID] IS NULL AND @original\_StateID IS NULL)) AND (([CityName] = @original\_CityName) OR ([CityName] IS NULL AND @original\_CityName IS NULL)) AND (([PinCode] = @original\_PinCode) OR ([PinCode] IS NULL AND @original\_PinCode IS NULL))">

<DeleteParameters>

<asp:Parameter Name="original\_CityID" Type="Int64" />

<asp:Parameter Name="original\_StateID" Type="String" />

<asp:Parameter Name="original\_CityName" Type="String" />

<asp:Parameter Name="original\_PinCode" Type="Int64" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="StateID" Type="String" />

<asp:Parameter Name="CityName" Type="String" />

<asp:Parameter Name="PinCode" Type="Int64" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="StateID" Type="String" />

<asp:Parameter Name="CityName" Type="String" />

<asp:Parameter Name="PinCode" Type="Int64" />

<asp:Parameter Name="original\_CityID" Type="Int64" />

<asp:Parameter Name="original\_StateID" Type="String" />

<asp:Parameter Name="original\_CityName" Type="String" />

<asp:Parameter Name="original\_PinCode" Type="Int64" />

</UpdateParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td>

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

</table>

</asp:Content>

**ManageCityPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.Admin

{

public partial class ManageCityPage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Add\_Click(object sender, EventArgs e)

{

if (CheckProgram())

{

LabelErrorMessage.Text = ("already exit");

return;

}

try

{

string \_ProcName = "addCity";

SqlParameter[] \_parameter = {

new SqlParameter("@StateName",DropDownListState.SelectedValue.ToString()),

new SqlParameter("@CityName",textBoxCityName.Text),

new SqlParameter("@PinCode",Convert.ToInt64(TextBoxPinCode.Text))

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

private bool CheckProgram()

{

bool flag = false;

try

{

string \_ProcName = "checkCity";

SqlParameter[] \_parameter = {

new SqlParameter("@CityName",textBoxCityName.Text)

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

return flag = true;

}

dr.Close();

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

return flag;

}

}

}

**ManageQuestionPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="ManageQuestion.aspx.cs" Inherits="OnDemandExamination.Admin.ManageQuestion" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1>Manage Question</h1>

<table style="width: 100%">

<tr>

<td>

<asp:DropDownList ID="DropDownListProgram" runat="server" DataSourceID="SqlDataSource1" DataTextField="ProgramName" DataValueField="ProgramName" AutoPostBack="True">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ProgramName] FROM [programDetails]"></asp:SqlDataSource>

</td>

<td>

<asp:DropDownList ID="DropDownListCourse" runat="server" DataSourceID="SqlDataSource2" DataTextField="CourseName" DataValueField="CourseName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [CourseName] FROM [CourseDeatils] WHERE ([ProgramName] = @ProgramName2)">

<SelectParameters>

<asp:ControlParameter ControlID="DropDownListProgram" Name="ProgramName2" PropertyName="SelectedValue" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<td colspan="2">

<asp:TextBox ID="TextBoxQuestion" runat="server" placeholder="Enter Questin" Height="95px" Width="592px"></asp:TextBox>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr>

<td>

<asp:TextBox ID="TextBoxOptionA" placeholder="Option A" runat="server"></asp:TextBox>

</td>

<td>

<asp:TextBox ID="TextBoxOptionB" placeholder="Option B" runat="server" OnTextChanged="TextBox4\_TextChanged"></asp:TextBox>

</td>

</tr>

<tr>

<td>

<asp:TextBox ID="TextBoxOptionC" placeholder="Option C" runat="server"></asp:TextBox>

</td>

<td>

<asp:TextBox ID="TextBoxOptionD" placeholder="Option D" runat="server"></asp:TextBox>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

<asp:DropDownList ID="DropDownListCorrectOption" runat="server">

<asp:ListItem Selected="True" Value="0">Correct Option</asp:ListItem>

<asp:ListItem>A</asp:ListItem>

<asp:ListItem>B</asp:ListItem>

<asp:ListItem>C</asp:ListItem>

<asp:ListItem>D</asp:ListItem>

</asp:DropDownList>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

<asp:Button ID="buttonAdd" runat="server" OnClick="buttonAdd\_Click" Text="Add" />

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="See Questions" />

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>

&nbsp;</td>

</tr>

</table>

</asp:Content>

**ManageQuestionPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.Admin

{

public partial class ManageQuestion : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void TextBox4\_TextChanged(object sender, EventArgs e)

{

}

protected void clear()

{

TextBoxQuestion.Text = "";

TextBoxOptionA.Text = "";

TextBoxOptionB.Text = "";

TextBoxOptionC.Text = "";

TextBoxOptionD.Text = "";

}

protected void buttonAdd\_Click(object sender, EventArgs e)

{

try

{

string \_ProcName = "addQuestion";

SqlParameter[] \_parameter = {

new SqlParameter("@Program",DropDownListProgram.SelectedValue),

new SqlParameter("@Course",DropDownListCourse.SelectedValue),

new SqlParameter("@Question",TextBoxQuestion.Text),

new SqlParameter("@OptionA",TextBoxOptionA.Text),

new SqlParameter("@OptionB",TextBoxOptionB.Text),

new SqlParameter("@OptionC",TextBoxOptionC.Text),

new SqlParameter("@OptionD",TextBoxOptionD.Text),

new SqlParameter("@CorrectOption",DropDownListCorrectOption.SelectedValue)

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

clear();

}

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("SeeQuestionPage.aspx");

}

}

}

**SeeQuestionPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="SeeQuestionPage.aspx.cs" Inherits="OnDemandExamination.Admin.SeeQuestionPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

<style type="text/css">

.auto-style1 {

width: 100%;

}

.auto-style2 {

height: 26px;

}

</style>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<table class="auto-style1">

<tr>

<td class="auto-style2">

<asp:DropDownList ID="DropDownProgram" runat="server" AutoPostBack="True" DataSourceID="SqlDataSource2" DataTextField="ProgramName" DataValueField="ProgramName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ProgramName] FROM [programDetails]"></asp:SqlDataSource>

</td>

<td class="auto-style2">

<asp:DropDownList ID="DropDownCourse" runat="server" AutoPostBack="True" DataSourceID="SqlDataSource3" DataTextField="CourseName" DataValueField="CourseName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [CourseName] FROM [CourseDeatils] WHERE ([ProgramName] = @ProgramName)">

<SelectParameters>

<asp:ControlParameter ControlID="DropDownProgram" Name="ProgramName" PropertyName="SelectedValue" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:DetailsView ID="DetailsView1" runat="server" AllowPaging="True" AutoGenerateRows="False" DataKeyNames="QuestionID" DataSourceID="SqlDataSource1" Height="50px" Width="608px">

<Fields>

<asp:BoundField DataField="QuestionID" HeaderText="QuestionID" InsertVisible="False" ReadOnly="True" SortExpression="QuestionID" />

<asp:BoundField DataField="Question" HeaderText="Question" SortExpression="Question" />

<asp:BoundField DataField="OptionA" HeaderText="OptionA" SortExpression="OptionA" />

<asp:BoundField DataField="OptionB" HeaderText="OptionB" SortExpression="OptionB" />

<asp:BoundField DataField="OptionC" HeaderText="OptionC" SortExpression="OptionC" />

<asp:BoundField DataField="OptionD" HeaderText="OptionD" SortExpression="OptionD" />

<asp:BoundField DataField="CorrectOption" HeaderText="CorrectOption" SortExpression="CorrectOption" />

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />

</Fields>

</asp:DetailsView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConflictDetection="CompareAllValues" ConnectionString="<%$ ConnectionStrings:connectionDb %>" DeleteCommand="DELETE FROM [QuestionDetails] WHERE [QuestionID] = @original\_QuestionID AND (([Question] = @original\_Question) OR ([Question] IS NULL AND @original\_Question IS NULL)) AND (([OptionA] = @original\_OptionA) OR ([OptionA] IS NULL AND @original\_OptionA IS NULL)) AND (([OptionB] = @original\_OptionB) OR ([OptionB] IS NULL AND @original\_OptionB IS NULL)) AND (([OptionC] = @original\_OptionC) OR ([OptionC] IS NULL AND @original\_OptionC IS NULL)) AND (([OptionD] = @original\_OptionD) OR ([OptionD] IS NULL AND @original\_OptionD IS NULL)) AND (([CorrectOption] = @original\_CorrectOption) OR ([CorrectOption] IS NULL AND @original\_CorrectOption IS NULL))" InsertCommand="INSERT INTO [QuestionDetails] ([Question], [OptionA], [OptionB], [OptionC], [OptionD], [CorrectOption]) VALUES (@Question, @OptionA, @OptionB, @OptionC, @OptionD, @CorrectOption)" OldValuesParameterFormatString="original\_{0}" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [QuestionID], [Question], [OptionA], [OptionB], [OptionC], [OptionD], [CorrectOption] FROM [QuestionDetails] WHERE ([Course] = @Course)" UpdateCommand="UPDATE [QuestionDetails] SET [Question] = @Question, [OptionA] = @OptionA, [OptionB] = @OptionB, [OptionC] = @OptionC, [OptionD] = @OptionD, [CorrectOption] = @CorrectOption WHERE [QuestionID] = @original\_QuestionID AND (([Question] = @original\_Question) OR ([Question] IS NULL AND @original\_Question IS NULL)) AND (([OptionA] = @original\_OptionA) OR ([OptionA] IS NULL AND @original\_OptionA IS NULL)) AND (([OptionB] = @original\_OptionB) OR ([OptionB] IS NULL AND @original\_OptionB IS NULL)) AND (([OptionC] = @original\_OptionC) OR ([OptionC] IS NULL AND @original\_OptionC IS NULL)) AND (([OptionD] = @original\_OptionD) OR ([OptionD] IS NULL AND @original\_OptionD IS NULL)) AND (([CorrectOption] = @original\_CorrectOption) OR ([CorrectOption] IS NULL AND @original\_CorrectOption IS NULL))">

<DeleteParameters>

<asp:Parameter Name="original\_QuestionID" Type="Int64" />

<asp:Parameter Name="original\_Question" Type="String" />

<asp:Parameter Name="original\_OptionA" Type="String" />

<asp:Parameter Name="original\_OptionB" Type="String" />

<asp:Parameter Name="original\_OptionC" Type="String" />

<asp:Parameter Name="original\_OptionD" Type="String" />

<asp:Parameter Name="original\_CorrectOption" Type="String" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="Question" Type="String" />

<asp:Parameter Name="OptionA" Type="String" />

<asp:Parameter Name="OptionB" Type="String" />

<asp:Parameter Name="OptionC" Type="String" />

<asp:Parameter Name="OptionD" Type="String" />

<asp:Parameter Name="CorrectOption" Type="String" />

</InsertParameters>

<SelectParameters>

<asp:ControlParameter ControlID="DropDownCourse" Name="Course" PropertyName="SelectedValue" Type="String" />

</SelectParameters>

<UpdateParameters>

<asp:Parameter Name="Question" Type="String" />

<asp:Parameter Name="OptionA" Type="String" />

<asp:Parameter Name="OptionB" Type="String" />

<asp:Parameter Name="OptionC" Type="String" />

<asp:Parameter Name="OptionD" Type="String" />

<asp:Parameter Name="CorrectOption" Type="String" />

<asp:Parameter Name="original\_QuestionID" Type="Int64" />

<asp:Parameter Name="original\_Question" Type="String" />

<asp:Parameter Name="original\_OptionA" Type="String" />

<asp:Parameter Name="original\_OptionB" Type="String" />

<asp:Parameter Name="original\_OptionC" Type="String" />

<asp:Parameter Name="original\_OptionD" Type="String" />

<asp:Parameter Name="original\_CorrectOption" Type="String" />

</UpdateParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

</table>

</asp:Content>

**SeeQuestionPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class SeeQuestionPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**StudentExamRequest.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="StudentExamRequest.aspx.cs" Inherits="OnDemandExamination.Admin.StudentExamRequest" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

<style type="text/css">

.auto-style1 {

width: 100%;

}

</style>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div>

<h1 class="center">Student Exam Request</h1>

<br />

<table class="auto-style1">

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:DetailsView ID="DetailsView1" runat="server" AllowPaging="True" AutoGenerateRows="False" BackColor="White" BorderColor="#999999" BorderStyle="None" BorderWidth="1px" CellPadding="3" DataKeyNames="RequestID" DataSourceID="SqlDataSource1" GridLines="Vertical" Height="50px" Width="381px">

<AlternatingRowStyle BackColor="#DCDCDC" />

<EditRowStyle BackColor="#008A8C" Font-Bold="True" ForeColor="White" />

<Fields>

<asp:BoundField DataField="RequestID" HeaderText="RequestID" InsertVisible="False" ReadOnly="True" SortExpression="RequestID" />

<asp:BoundField DataField="UserName" HeaderText="UserName" SortExpression="UserName" />

<asp:BoundField DataField="CourseName1" HeaderText="CourseName1" SortExpression="CourseName1" />

<asp:BoundField DataField="CourseName2" HeaderText="CourseName2" SortExpression="CourseName2" />

<asp:BoundField DataField="CourseName3" HeaderText="CourseName3" SortExpression="CourseName3" />

<asp:BoundField DataField="flag1" HeaderText="Allow course 1" SortExpression="flag1" />

<asp:BoundField DataField="flag2" HeaderText="Allow course 2" SortExpression="flag2" />

<asp:BoundField DataField="flag3" HeaderText="Allow course 3" SortExpression="flag3" />

<asp:BoundField DataField="Ts\_Created" HeaderText="Requested Date" SortExpression="Ts\_Created" />

<asp:CommandField ShowEditButton="True" ShowDeleteButton="True" />

</Fields>

<FooterStyle BackColor="#CCCCCC" ForeColor="Black" />

<HeaderStyle BackColor="#000084" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#999999" ForeColor="Black" HorizontalAlign="Center" />

<RowStyle BackColor="#EEEEEE" ForeColor="Black" />

</asp:DetailsView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConflictDetection="CompareAllValues"

ConnectionString="<%$ ConnectionStrings:connectionDb %>"

DeleteCommand="DELETE FROM [ExamRequestDetails] WHERE [RequestID] = @original\_RequestID"

InsertCommand="INSERT INTO [ExamRequestDetails] ([UserName], [CourseName1], [CourseName2], [CourseName3], [flag1], [flag2], [flag3], [Ts\_Created]) VALUES (@UserName, @CourseName1, @CourseName2, @CourseName3, @flag1, @flag2, @flag3, @Ts\_Created)"

OldValuesParameterFormatString="original\_{0}" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>"

SelectCommand="SELECT [RequestID], [UserName], [CourseName1], [CourseName2], [CourseName3], [flag1], [flag2], [flag3], [Ts\_Created] FROM [ExamRequestDetails]"

UpdateCommand="UPDATE [ExamRequestDetails] SET [UserName] = @UserName, [CourseName1] = @CourseName1, [CourseName2] = @CourseName2, [CourseName3] = @CourseName3, [flag1] = @flag1, [flag2] = @flag2, [flag3] = @flag3, [Ts\_Created] = @Ts\_Created WHERE [RequestID] = @original\_RequestID">

<DeleteParameters>

<asp:Parameter Name="original\_RequestID" Type="Int64" />

<asp:Parameter Name="original\_UserName" Type="String" />

<asp:Parameter Name="original\_CourseName1" Type="String" />

<asp:Parameter Name="original\_CourseName2" Type="String" />

<asp:Parameter Name="original\_CourseName3" Type="String" />

<asp:Parameter Name="original\_flag1" Type="Int32" />

<asp:Parameter Name="original\_flag2" Type="Int32" />

<asp:Parameter Name="original\_flag3" Type="Int32" />

<asp:Parameter Name="original\_Ts\_Created" Type="DateTime" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="UserName" Type="String" />

<asp:Parameter Name="CourseName1" Type="String" />

<asp:Parameter Name="CourseName2" Type="String" />

<asp:Parameter Name="CourseName3" Type="String" />

<asp:Parameter Name="flag1" Type="Int32" />

<asp:Parameter Name="flag2" Type="Int32" />

<asp:Parameter Name="flag3" Type="Int32" />

<asp:Parameter Name="Ts\_Created" Type="DateTime" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="UserName" Type="String" />

<asp:Parameter Name="CourseName1" Type="String" />

<asp:Parameter Name="CourseName2" Type="String" />

<asp:Parameter Name="CourseName3" Type="String" />

<asp:Parameter Name="flag1" Type="Int32" />

<asp:Parameter Name="flag2" Type="Int32" />

<asp:Parameter Name="flag3" Type="Int32" />

<asp:Parameter Name="Ts\_Created" Type="DateTime" />

<asp:Parameter Name="original\_RequestID" Type="Int64" />

<asp:Parameter Name="original\_UserName" Type="String" />

<asp:Parameter Name="original\_CourseName1" Type="String" />

<asp:Parameter Name="original\_CourseName2" Type="String" />

<asp:Parameter Name="original\_CourseName3" Type="String" />

<asp:Parameter Name="original\_flag1" Type="Int32" />

<asp:Parameter Name="original\_flag2" Type="Int32" />

<asp:Parameter Name="original\_flag3" Type="Int32" />

<asp:Parameter Name="original\_Ts\_Created" Type="DateTime" />

</UpdateParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

</table>

</div>

</asp:Content>

**StudentExamRequest.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class StudentExamRequest : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**StudentListPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="StudentListPage.aspx.cs" Inherits="OnDemandExamination.Admin.StudentListPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<p>

<br />

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="UserID" DataSourceID="SqlDataSource1">

<Columns>

<asp:BoundField DataField="UserID" HeaderText="UserID" InsertVisible="False" ReadOnly="True" SortExpression="UserID" />

<asp:BoundField DataField="FirstName" HeaderText="FirstName" SortExpression="FirstName" />

<asp:BoundField DataField="LastName" HeaderText="LastName" SortExpression="LastName" />

<asp:BoundField DataField="LoginID" HeaderText="LoginID" SortExpression="LoginID" />

<asp:BoundField DataField="Email" HeaderText="Email" SortExpression="Email" />

<asp:BoundField DataField="Phone" HeaderText="Phone" SortExpression="Phone" />

<asp:BoundField DataField="Program" HeaderText="Program" SortExpression="Program" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConflictDetection="CompareAllValues" ConnectionString="<%$ ConnectionStrings:connectionDb %>" DeleteCommand="DELETE FROM [UserDetials] WHERE [UserID] = @original\_UserID AND (([FirstName] = @original\_FirstName) OR ([FirstName] IS NULL AND @original\_FirstName IS NULL)) AND (([LastName] = @original\_LastName) OR ([LastName] IS NULL AND @original\_LastName IS NULL)) AND (([LoginID] = @original\_LoginID) OR ([LoginID] IS NULL AND @original\_LoginID IS NULL)) AND (([Email] = @original\_Email) OR ([Email] IS NULL AND @original\_Email IS NULL)) AND (([Phone] = @original\_Phone) OR ([Phone] IS NULL AND @original\_Phone IS NULL)) AND (([Program] = @original\_Program) OR ([Program] IS NULL AND @original\_Program IS NULL))" InsertCommand="INSERT INTO [UserDetials] ([FirstName], [LastName], [LoginID], [Email], [Phone], [Program]) VALUES (@FirstName, @LastName, @LoginID, @Email, @Phone, @Program)" OldValuesParameterFormatString="original\_{0}" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [UserID], [FirstName], [LastName], [LoginID], [Email], [Phone], [Program] FROM [UserDetials]" UpdateCommand="UPDATE [UserDetials] SET [FirstName] = @FirstName, [LastName] = @LastName, [LoginID] = @LoginID, [Email] = @Email, [Phone] = @Phone, [Program] = @Program WHERE [UserID] = @original\_UserID AND (([FirstName] = @original\_FirstName) OR ([FirstName] IS NULL AND @original\_FirstName IS NULL)) AND (([LastName] = @original\_LastName) OR ([LastName] IS NULL AND @original\_LastName IS NULL)) AND (([LoginID] = @original\_LoginID) OR ([LoginID] IS NULL AND @original\_LoginID IS NULL)) AND (([Email] = @original\_Email) OR ([Email] IS NULL AND @original\_Email IS NULL)) AND (([Phone] = @original\_Phone) OR ([Phone] IS NULL AND @original\_Phone IS NULL)) AND (([Program] = @original\_Program) OR ([Program] IS NULL AND @original\_Program IS NULL))">

<DeleteParameters>

<asp:Parameter Name="original\_UserID" Type="Int64" />

<asp:Parameter Name="original\_FirstName" Type="String" />

<asp:Parameter Name="original\_LastName" Type="String" />

<asp:Parameter Name="original\_LoginID" Type="String" />

<asp:Parameter Name="original\_Email" Type="String" />

<asp:Parameter Name="original\_Phone" Type="Int64" />

<asp:Parameter Name="original\_Program" Type="String" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="FirstName" Type="String" />

<asp:Parameter Name="LastName" Type="String" />

<asp:Parameter Name="LoginID" Type="String" />

<asp:Parameter Name="Email" Type="String" />

<asp:Parameter Name="Phone" Type="Int64" />

<asp:Parameter Name="Program" Type="String" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="FirstName" Type="String" />

<asp:Parameter Name="LastName" Type="String" />

<asp:Parameter Name="LoginID" Type="String" />

<asp:Parameter Name="Email" Type="String" />

<asp:Parameter Name="Phone" Type="Int64" />

<asp:Parameter Name="Program" Type="String" />

<asp:Parameter Name="original\_UserID" Type="Int64" />

<asp:Parameter Name="original\_FirstName" Type="String" />

<asp:Parameter Name="original\_LastName" Type="String" />

<asp:Parameter Name="original\_LoginID" Type="String" />

<asp:Parameter Name="original\_Email" Type="String" />

<asp:Parameter Name="original\_Phone" Type="Int64" />

<asp:Parameter Name="original\_Program" Type="String" />

</UpdateParameters>

</asp:SqlDataSource>

</p>

</asp:Content>

**StudentListPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class StudentListPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**StudentResultPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="StudentResultPage.aspx.cs" Inherits="OnDemandExamination.Admin.StudentResultPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

<style type="text/css">

.auto-style1 {

width: 100%;

}

.auto-style2 {

width: 382px;

}

</style>

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div>

<h1>Student Results</h1>

<br />

<table class="auto-style1">

<tr>

<td class="auto-style2">&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr>

<td colspan="2">

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" CellPadding="4" DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None" AllowPaging="True" AllowSorting="True" DataKeyNames="ResultID">

<AlternatingRowStyle BackColor="White" />

<Columns>

<asp:CommandField ShowEditButton="True" />

<asp:BoundField DataField="ResultID" HeaderText="ResultID" SortExpression="ResultID" InsertVisible="False" ReadOnly="True" />

<asp:BoundField DataField="UserName" HeaderText="UserName" SortExpression="UserName" />

<asp:BoundField DataField="CourseName" HeaderText="CourseName" SortExpression="CourseName" />

<asp:BoundField DataField="ExamDate" HeaderText="ExamDate" SortExpression="ExamDate" />

<asp:BoundField DataField="Marks" HeaderText="Marks" SortExpression="Marks" />

<asp:BoundField DataField="Status" HeaderText="Status" SortExpression="Status" />

<asp:BoundField DataField="flag" HeaderText="Allow Result" SortExpression="flag" />

</Columns>

<FooterStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#FFCC66" ForeColor="#333333" HorizontalAlign="Center" />

<RowStyle BackColor="#FFFBD6" ForeColor="#333333" />

<SelectedRowStyle BackColor="#FFCC66" Font-Bold="True" ForeColor="Navy" />

<SortedAscendingCellStyle BackColor="#FDF5AC" />

<SortedAscendingHeaderStyle BackColor="#4D0000" />

<SortedDescendingCellStyle BackColor="#FCF6C0" />

<SortedDescendingHeaderStyle BackColor="#820000" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>"

SelectCommand="SELECT [ResultID], [UserName], [CourseName], [ExamDate], [Marks], [Status], [flag] FROM [ResultDetails] ORDER BY [ExamDate]"

ConflictDetection="CompareAllValues"

DeleteCommand="DELETE FROM [ResultDetails] WHERE [ResultID] = @original\_ResultID"

InsertCommand="INSERT INTO [ResultDetails] ([UserName], [CourseName], [ExamDate], [Marks], [Status], [flag]) VALUES (@UserName, @CourseName, @ExamDate, @Marks, @Status, @flag)"

OldValuesParameterFormatString="original\_{0}"

UpdateCommand="UPDATE [ResultDetails] SET [UserName] = @UserName, [CourseName] = @CourseName, [ExamDate] = @ExamDate, [Marks] = @Marks, [Status] = @Status, [flag] = @flag WHERE [ResultID] = @original\_ResultID">

<DeleteParameters>

<asp:Parameter Name="original\_ResultID" Type="Int32" />

<asp:Parameter Name="original\_UserName" Type="String" />

<asp:Parameter Name="original\_CourseName" Type="String" />

<asp:Parameter Name="original\_ExamDate" Type="DateTime" />

<asp:Parameter Name="original\_Marks" Type="Int32" />

<asp:Parameter Name="original\_Status" Type="String" />

<asp:Parameter Name="original\_flag" Type="Int32" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="UserName" Type="String" />

<asp:Parameter Name="CourseName" Type="String" />

<asp:Parameter Name="ExamDate" Type="DateTime" />

<asp:Parameter Name="Marks" Type="Int32" />

<asp:Parameter Name="Status" Type="String" />

<asp:Parameter Name="flag" Type="Int32" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="UserName" Type="String" />

<asp:Parameter Name="CourseName" Type="String" />

<asp:Parameter Name="ExamDate" Type="DateTime" />

<asp:Parameter Name="Marks" Type="Int32" />

<asp:Parameter Name="Status" Type="String" />

<asp:Parameter Name="flag" Type="Int32" />

<asp:Parameter Name="original\_ResultID" Type="Int32" />

<asp:Parameter Name="original\_UserName" Type="String" />

<asp:Parameter Name="original\_CourseName" Type="String" />

<asp:Parameter Name="original\_ExamDate" Type="DateTime" />

<asp:Parameter Name="original\_Marks" Type="Int32" />

<asp:Parameter Name="original\_Status" Type="String" />

<asp:Parameter Name="original\_flag" Type="Int32" />

</UpdateParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td class="auto-style2">&nbsp;</td>

<td>&nbsp;</td>

</tr>

</table>

</div>

</asp:Content>

**StudentResultPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class StudentResultPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**ExamSettingPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="ExamSettingPage.aspx.cs" Inherits="OnDemandExamination.Admin.ExamSettingPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1 class="center">Exam Setting</h1>

<br />

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" DataSourceID="SqlDataSource1">

<Columns>

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />

<asp:BoundField DataField="ExamId" HeaderText="ExamId" SortExpression="ExamId" />

<asp:BoundField DataField="TimeLimit" HeaderText="TimeLimit" SortExpression="TimeLimit" />

<asp:BoundField DataField="NoOfQuestion" HeaderText="NoOfQuestion" SortExpression="NoOfQuestion" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [ExamId], [TimeLimit], [NoOfQuestion] FROM [ExamSetting]" ConflictDetection="CompareAllValues" DeleteCommand="DELETE FROM [ExamSetting] WHERE [ExamId] = @original\_ExamId AND (([TimeLimit] = @original\_TimeLimit) OR ([TimeLimit] IS NULL AND @original\_TimeLimit IS NULL)) AND (([NoOfQuestion] = @original\_NoOfQuestion) OR ([NoOfQuestion] IS NULL AND @original\_NoOfQuestion IS NULL))" InsertCommand="INSERT INTO [ExamSetting] ([TimeLimit], [NoOfQuestion]) VALUES (@TimeLimit, @NoOfQuestion)" OldValuesParameterFormatString="original\_{0}" UpdateCommand="UPDATE [ExamSetting] SET [TimeLimit] = @TimeLimit, [NoOfQuestion] = @NoOfQuestion WHERE [ExamId] = @original\_ExamId AND (([TimeLimit] = @original\_TimeLimit) OR ([TimeLimit] IS NULL AND @original\_TimeLimit IS NULL)) AND (([NoOfQuestion] = @original\_NoOfQuestion) OR ([NoOfQuestion] IS NULL AND @original\_NoOfQuestion IS NULL))">

<DeleteParameters>

<asp:Parameter Name="original\_ExamId" Type="Int32" />

<asp:Parameter Name="original\_TimeLimit" Type="Int32" />

<asp:Parameter Name="original\_NoOfQuestion" Type="Int32" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="TimeLimit" Type="Int32" />

<asp:Parameter Name="NoOfQuestion" Type="Int32" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="TimeLimit" Type="Int32" />

<asp:Parameter Name="NoOfQuestion" Type="Int32" />

<asp:Parameter Name="original\_ExamId" Type="Int32" />

<asp:Parameter Name="original\_TimeLimit" Type="Int32" />

<asp:Parameter Name="original\_NoOfQuestion" Type="Int32" />

</UpdateParameters>

</asp:SqlDataSource>

</asp:Content>

**ExamSettingPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace OnDemandExamination.Admin

{

public partial class ExamSettingPage : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

}

}

**AddTeachersPage.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Admin/AdminMasterPage.Master" AutoEventWireup="true" CodeBehind="AddTeachersPage.aspx.cs" Inherits="OnDemandExamination.Admin.AddTeachersPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<h1 class="center">Add Teachers</h1>

<table style="border-width: thick; border-color: #000000; width:100%; padding: 3px; font-weight: bold;" id="RegstrationTable">

<tr>

<td style="width: 146px">&nbsp;</td>

</tr>

<tr>

<td style="width: 146px">First Name</td>

<td>

<asp:TextBox ID="FirstName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="FirstName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Last Name</td>

<td style="height: 26px">

<asp:TextBox ID="LastName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="LastName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Login ID</td>

<td style="height: 26px">

<asp:ScriptManager runat="server"></asp:ScriptManager>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

<ContentTemplate>

<asp:TextBox ID="LoginId" runat="server" OnTextChanged="LoginId\_TextChanged" AutoPostBack="True" AutoCompleteType="Disabled"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator7" runat="server" ControlToValidate="LoginId" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</ContentTemplate>

</asp:UpdatePanel>

<asp:Label ID="lblUserAvailabel" runat="server"></asp:Label>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Password</td>

<td style="height: 26px">

<asp:TextBox ID="Password" runat="server" TextMode="Password" AutoCompleteType="Disabled"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator8" runat="server" ControlToValidate="Password" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Confirm Password</td>

<td style="height: 26px">

<asp:TextBox ID="confirmPassword" runat="server" TextMode="Password"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator9" runat="server" ControlToValidate="FirstName" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

<br />

<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="Password" ControlToValidate="confirmPassword" ErrorMessage="password does not match" ForeColor="#CC0066" SetFocusOnError="True"></asp:CompareValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">E-mail</td>

<td style="height: 26px">

<asp:TextBox ID="Email" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="Email" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

<br />

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="Email" ErrorMessage="Email should be in correct formate" ForeColor="Red" SetFocusOnError="True" ValidationExpression="\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Phone No.</td>

<td style="height: 26px">

<asp:TextBox ID="phone" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator3" runat="server" ControlToValidate="phone" ErrorMessage="Please enter valid Phone No." ForeColor="#FF3399" SetFocusOnError="True" ValidationExpression="\d{10}"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">DOB</td>

<td style="height: 26px">

<asp:TextBox ID="dob" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ControlToValidate="dob" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Program</td>

<td style="height: 26px">

<asp:DropDownList ID="DropDownListProgram" runat="server" DataSourceID="SqlDataSource3" DataTextField="ProgramName" DataValueField="ProgramName">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT DISTINCT [ProgramName] FROM [programDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px; height: 26px;">Address</td>

<td style="height: 26px">

<asp:TextBox ID="address" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server" ControlToValidate="address" ErrorMessage="this field can not be empty" ForeColor="#CC0000" SetFocusOnError="True"></asp:RequiredFieldValidator>

</td>

</tr>

<tr>

<td style="width: 146px; height: 24px;">Gender</td>

<td style="height: 24px">

<asp:RadioButtonList ID="RadioButtonListGender" runat="server" RepeatDirection="Horizontal">

<asp:ListItem>Male</asp:ListItem>

<asp:ListItem>Female</asp:ListItem>

</asp:RadioButtonList>

</td>

</tr>

<tr>

<td style="width: 146px; height: 24px;">State</td>

<td style="height: 24px">

<asp:DropDownList ID="DropDownListState" runat="server" DataSourceID="SqlDataSource1" DataTextField="StateName" DataValueField="StateName" AutoPostBack="True">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [StateName] FROM [StateDetails]"></asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px">City</td>

<td>

<asp:DropDownList ID="DropDownListCity" runat="server" DataSourceID="SqlDataSource2" DataTextField="CityName" DataValueField="CityName">

<asp:ListItem Value="0">Select City</asp:ListItem>

<asp:ListItem Value="1">Laxmi Nagar</asp:ListItem>

<asp:ListItem Value="2">Nirman Vihar</asp:ListItem>

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:connectionDb %>" ProviderName="<%$ ConnectionStrings:connectionDb.ProviderName %>" SelectCommand="SELECT [CityName] FROM [CityDetails] WHERE ([StateID] = @StateID)">

<SelectParameters>

<asp:ControlParameter ControlID="DropDownListState" Name="StateID" PropertyName="SelectedValue" Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</td>

</tr>

<tr>

<td style="width: 146px">Pin Code</td>

<td>

<asp:TextBox ID="PinCode" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ControlToValidate="PinCode" ErrorMessage="please enter valid pin code" ForeColor="#FF3399" SetFocusOnError="True" ValidationExpression="\d{6}"></asp:RegularExpressionValidator>

</td>

</tr>

<tr>

<td style="width: 146px">

<asp:Button ID="ButtonReset" runat="server" Text="Reset" Height="28px" OnClick="ButtonReset\_Click" />

</td>

<td>

<asp:Button ID="ButtonSubmit" runat="server" Text="Submit" OnClick="ButtonSubmit\_Click" />

&nbsp;

<asp:Label ID="LabelErrorMessage" runat="server" ForeColor="#CC0000"></asp:Label>

</td>

</tr>

</table>

</asp:Content>

**AddTeachersPage.aspx.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using OnDemandExamination.App\_Code;

namespace OnDemandExamination.Admin

{

public partial class AddTeachersPage : System.Web.UI.Page

{

DataClassesDataContext db = new DataClassesDataContext();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void ButtonSubmit\_Click(object sender, EventArgs e)

{

if (CheckUserName())

{

return;

}

try

{

DateTime now = DateTime.Now;

string date = now.ToString();

string \_ProcName = "AddTeacher";

SqlParameter[] \_parameter = {

new SqlParameter("@FirstName",FirstName.Text),

new SqlParameter("@LastName",LastName.Text),

new SqlParameter("@LoginID", LoginId.Text),

new SqlParameter("@Password",Password.Text),

new SqlParameter("@Email",Email.Text),

new SqlParameter("@Phone",Convert.ToInt64(phone.Text)),

new SqlParameter("@Gender",RadioButtonListGender.SelectedValue),

new SqlParameter("@DOB",dob.Text),

new SqlParameter("@Program",DropDownListProgram.SelectedValue.ToString()),

new SqlParameter("@Address",address.Text),

new SqlParameter("@StateID",DropDownListState.SelectedValue.ToString()),

new SqlParameter("@cityID",DropDownListCity.SelectedValue.ToString()),

new SqlParameter("@PinCoad",Convert.ToInt64(PinCode.Text))

};

int index = db.ExecuteNonQueryByQueryProc(\_parameter, \_ProcName);

if (index > 0)

{

LabelErrorMessage.Text = "Insert successfull";

}

else

{

LabelErrorMessage.Text = "Insert Failed";

}

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

}

protected void ButtonReset\_Click(object sender, EventArgs e)

{

Response.Redirect("RegistrationPage.aspx");

}

private bool CheckUserName()

{

bool flag = false;

try

{

string \_ProcName = "CheckUsername";

SqlParameter[] \_parameter = {

new SqlParameter("@LoginId",LoginId.Text)

};

SqlDataReader dr = db.GetDataReaderByProc(\_ProcName, \_parameter);

dr.Read();

if (dr.HasRows)

{

return flag = true;

}

dr.Close();

}

catch (Exception ex)

{

LabelErrorMessage.Text = ex.Message;

}

return flag;

}

protected void LoginId\_TextChanged(object sender, EventArgs e)

{

if (CheckUserName())

{

lblUserAvailabel.ForeColor = System.Drawing.Color.Red;

lblUserAvailabel.Text = "Username already exist";

}

else

{

lblUserAvailabel.ForeColor = System.Drawing.Color.Green;

lblUserAvailabel.Text = "Username Availabel";

}

}

}

}

**ERROR HANDLING:**

An Exception occurs when a program encounter any unexpected problems. Such as running out of memory or attempting to read from a file that no longer exists. These problems are not necessarily caused by a programming error but they mainly occur because of violation of assumption that you might have made about the execution environment. When a program encounters an exception the default behavior is to throw the exception which generally translates to abruptly, terminating the program after displaying an error message. But this is not a characteristic of a robust application. But the best way is to bindle the exception situations if possible, gracefully recover from them. This is called “exception handling”. I used try, catch, finally and throw in my project to handle the exception.

**The Try Block**: Place the code that might cause exception in a try block. A typical try block looks like this

Try

{

//Code that may cause exception

}

A try block can have another try block inside when an exception occurs at any point rather than executing any further lines of code, the CLR (Common Language Runtime) Secures for the nearest try block that enclosure this code. . The control is then passed to a matching catch block if any and then to the kindly block associated with this try block.

**Catch Block:** There can be no of catch blocks immediately following a try block. Each catch block handles an exception of a particular type. When an exception occurs in a statement placed inside the try block the CLR looks for a mainly block that is capable of handling the type of exception.

**Throw block**: A throw statement explicitly generates an exception in code. You can throw when a particular path in code results in an anomalous situation.

**Finally Block:** The finally block contains the code that always executes whenever or not any exception occurs.

**Parameter Passing**

Passing parameters from one page to another is a very common task in Web development. There are still many situations in which you need to pass data from one Web page to another. One of the simplest and most efficient ways of passing parameters among pages is to use the query string. Unfortunately, packing data into the query string via string manipulations can quickly lead to cumbersome and often difficult to maintain code, especially as the parameter list grows. To overcome this problem, I’ve used Session in my project.

**Query String some of them are described below **

* Query String is client side. But Session is server side. 
* The information or data stored in Query String is visible to everyone. But in Session it is hidden and can’t be viewed easily. 
* Query String can store only a piece of information but in Session we can store the more and more data. 
* The Query String speed never falls as the load increase because it stores a piece of information. But on the other hand Session increase congestion as the loads increase.

**Validation Checks**

**(1) Date Validation**:The validation on date data type has been specified to be of the format DD/MM/YY. Any other format is unacceptable.

**(2) Time validation:** The validation on time data type has been specified to be of the format hours-minutes-seconds. Any other format is unacceptable.

**(3) Number field validation:** The field specified with number as then their data- type will not accept character.

**(4) User Authentication:** When a Customer/user logs on to the system to access data from tables and database, the Id & password needs to be checked.

**(5) Password change Validation:** Only authorized users are allowed to change the password and the process requires asking the old password before changing it to the new one.

**TESTING**

## Test Case Execution

The workflow diagram below depicts the high level steps necessary to follow in order to set up and execute test based on the Test Case Template.



**Legend**

1 - BII WG4 Test Case Template.doc. The **Test Case template** used to define and set up the **Test Case Description**.

2 – The **test object specification** provides a reference to the object subject to test or if required, enter a copy of the object description excerpted from the object description for the test object. When referenced, the reference should include at least :

Testing is the process to uncover the errors.

**Objectives of Testing: -**

There are following objectives:

1. Testing is the process of executing the program to find error.
2. A group test has a high probability of finding the errors.
3. A successful test uncovers the all errors that have not been found.

**Testing Principle: -**

There are following objectives:

1. The test should be according to the customer’s requirement.
2. There should be a planning for testing before it starts.
3. Poreto principle implies that 80 percent of all errors uncover during testing will likely be traceable to 20 percent of all program components.
4. Testing should begin ‘in the small’ and progress toward testing ‘in the large’.

There are two types of testing: -

* 1. Black Box Testing
  2. White Box Testing

**1. Black Box Testing: -** It is also called behavioral testing. The program is directly run by the computer to find the errors.

**Objective of the Black Box Testing**

1. Incorrect or missing function.
2. Interface error.
3. Errors in data structures on database access.
4. Performance error.
5. Initialization and termination error.

**2. White Box Testing: -** It is also called glass box testing. It traces all the paths of a program manually to find the errors.

**Advantage of White Box Testing**

1. It guarantees that all independent paths have been checked at least once.
2. It chicks all logical decisions for true and false.
3. Executes all loops at their boundary values.
4. Checks internal data structures.

**Reasons for White Box Testing**

1. It can find logical errors, which cannot be found by ‘black box’.
2. We often believe that a logical path is not likely to be executed when, in fact, it may be executed on a regular basis.
3. Typographical errors are random. The block box testing can find out typing error but typing error but typing error are in the program.

**Unit Testing**

It is a technique of testing individual module at a time. The important control paths are tested to find the errors within the boundary of the module.

The interface is tested to check that input and output the module are correct. The data structure is tested to check that the data flow from input to output is correct. Boundary conditions are tested to check that the module works correct at boundary. The independent paths are tested to check that each part is executed t least once then all error paths are checked.

Driver

Module to be Tested

Stub

Stub

Test Case

**Integration Test**

It is the technique of testing after integrating the module. It finds the error related to the interface. There are two method of integration test:

1. Top Down Test
2. Bottom Up Test
3. **Top Down Test: -** The modules are integrated by moving down from top to bottom. The modules can be integrated by either using depth-first integrated or breadth-first integration. In DFS modules all integrated from top to down on individual path. For e.g. M1, M2, M5, & M8 INTEGRATED FIRST. In BFS integration is level by level. For e.g. Modules M1, M2, M3 and M4 all are integrated first.

M2

M5

M8

M6

M1

M3

M7

M4

The steps of top-down test:-

1. The main control module is used as a test driver and stubs are substituted for all components directly sub ordinate to the main control module.
2. Depending on the integration approach selected subordinate stubs are replaced one at a time with actual components.
3. Tests are conducted as each component is integrated.
4. One completion of each set of tests, another stub is replaced with the real component.
5. Regression testing may be conducted to ensure that new errors have not been introduced.

**B. Bottom up Integration: -** Integrates the modules from bottom to up. It has following steps:

1. Low level components are combined into clusters.
2. Drivers are developed for clusters.
3. Cluster is tested.
4. Drivers are removed and Chester is combined moving upward.

M2

D1

D2

M1

M3

Cluster 1 Cluster 2

**System Testing**

It is a series of different tests to test the overall system. Although each test has a different purpose, all work to verity those elements have been properly integrated and preformed allocated functions. The type of system tests are following:

1. **Recovery Test: -** It is performed to ensure that the data can be recovered and system can be restarted even if the system failure. It is a system test that forces the software to fail in a variety of ways and verified that recovery is performed properly. If recovery is automatic re-initialization, checkpoint mechanism data recovery and restart are evaluated for corrections. If recovery requires human intervention, the mean-time-to-repair is evaluated to determine whether it is within acceptable limits.
2. **Security Test: -** It is performing to verify that protection mechanisms built into a system are perfect or not i.e., can it protect system from improper penetration attempt. The system should be tested for any type of for query attempt. During security testing the tester play the role of individual who desired to penetrate the system. The faster may attempt to get password, may attack the system to break down the security may purposely cause system errors, may browse through unsecure data.  
   The good security testing will penetrate the system and break the security. So, the role of the system designer is to make penetration cost more than the value that will be obtained by breaking the system security.
3. **Stress Test: -** It is performed to test the abnormal situation i.e. how high we can crank the system before it fails. Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency or volume. For example
4. Special test may be design that generates ten interrupts per second, when one or two is the average rate.
5. Input data rates may be increased by an order of magnitude to determine how input function will respond.
6. Test cases that require maximum memory or other resource are executed.
7. Test cases that may cause thrashing in a virtual operating system are designed.
8. Test cases that may cause excessive hunting for disk resident data are created.  
   Essentially, the tester attempts to break the program.
9. **Performance Test: -** It is used to test the run time performance of the system. It occurs throughout all the steps in the testing process. The performance of an individual module may be checked using white box method. It is necessary to measure the resource utilization.

**Test case:**

A test case has components that describe an input, action or event and an expected response, to determine if a feature of an application is working correctly.”

 A test case is also defined as a sequence of steps to test the correct behavior of a functionality/feature of an application

A test case is a list of the conditions or issues of what the tester want to test in software. Test case helps to come up with test data. A test case has an input description, Test sequence and an expected behavior**.**

The characteristics of a test case are that there is a *known input* and an *expected output*, which is worked out *before* the test. The known input should test a pre-condition and the expected output should test a post-condition

**How to write test cases?**Here is a simple test case format

**Fields in test cases:**

**Test case id:  
Unit to test:** What to be verified?  
**Assumptions:  
Test data:** Variables and their values  
**Steps to be executed:  
Expected result:  
Actual result:  
Pass/Fail:  
Comments:**

**Registration Page**

**Fields in test cases: Email id**

**Test case id: 01  
Unit to test:** Email id format.  
**Assumptions: User can fill this field without email id format for that time the system generate an error massage “formet is wrong”.  
Test data:** Varchar (50) and ‘vikram’  
**Steps to be executed: 01  
Expected result: Show Error  
Actual result: Generate error “Please fill correct email id”  
Pass/Fail: Pass  
Comments: This validation is working properly.**

**Fields in test cases: Email id**

**Test case id: 01  
Unit to test:** Email id format.  
**Assumptions: User can fill this field without email id format for that time the system generate an error massage “wrong formet”.  
Test data:** Varchar (50) and ‘vikram205051@gmail.com’  
**Steps to be executed: 02  
Expected result: No Error  
Actual result: No any Error**

**Pass/Fail: Pass  
Comments: This validation is working properly.**

**Login page**

**Fields in test cases: Email id**

**Test case id: 01  
Unit to test:** Email id format.  
**Assumptions: User can fill this field without email id format for that time the system generate an error massage “wrong formet”.  
Test data:** Varchar (50) and ‘vikram205051@gmail.com’  
**Steps to be executed: 02  
Expected result: No Error  
Actual result: No any Error**

**Pass/Fail: Pass  
Comments: This validation is working properly.**

**Fields in test cases: user id and passward**

**Test case id: 01  
Unit to test:**fields are empty.  
**Assumptions:user can not fill user id and passward for that time the system generate an error message”fild cannot be empty”.  
Test data:** Varchar (50) and ‘dota2’  
**Steps to be executed: 02  
Expected result: No Error  
Actual result: No any Error**

**Pass/Fail: Pass  
Comments: This validation is working properly.**

**Debugging And Code Improvement**

In ideal worlds, all programmers would be so skilled and attentive to detail that they would write bug-free code. Unfortunately, we do not live in an ideal world. As such, debugging, or tracking down the source of errors and erroneous result, is an important task that all developers need to perform before they allow end-user to use their applications. We will discuss some techniques for reducing the number of bugs in code up front.

**There are three categories of bugs**

**Syntax error:**

These errors occur when code breaks the rule of the language, such as visual Basic sub statement without a closing End sub, or a forgotten closing curly braces ({}) in c#. Theses error the easiest to locate. The language complier **or** integrated development environment (IDE) will alert you to them and will not allow you to compile your program until you correct them.

**Semantic error:**

These errors occur in code that is correct according to rules of the compiler, but that causes unexpected problems such as crashes or hanging on execution. A good example is code that execute in a loop but never exists the loop, either because the loop depends on the variable whose values was expected to be something different than it actually was or because the programmer forget to increment the loop counter. Another category of errors in this area includes requesting a field from a dataset, there is no way to tell if the field actually exists at compile time. These bugs are harder to detect and are one type of running error.

**Logic error:**

Logic errors are like semantic errors, logic errors are runtime error. That is, they occur while the program is running. But unlike semantic errors, logic errors do not cause the application to crash or hang. Logic error results in unexpected values or output. This can be a result of something as simple as a mistyped variables name that happens to match another declared variable in the program. This type of error can be extremely difficult to track down to eliminate.

**Preventing Debug Write readable code:**

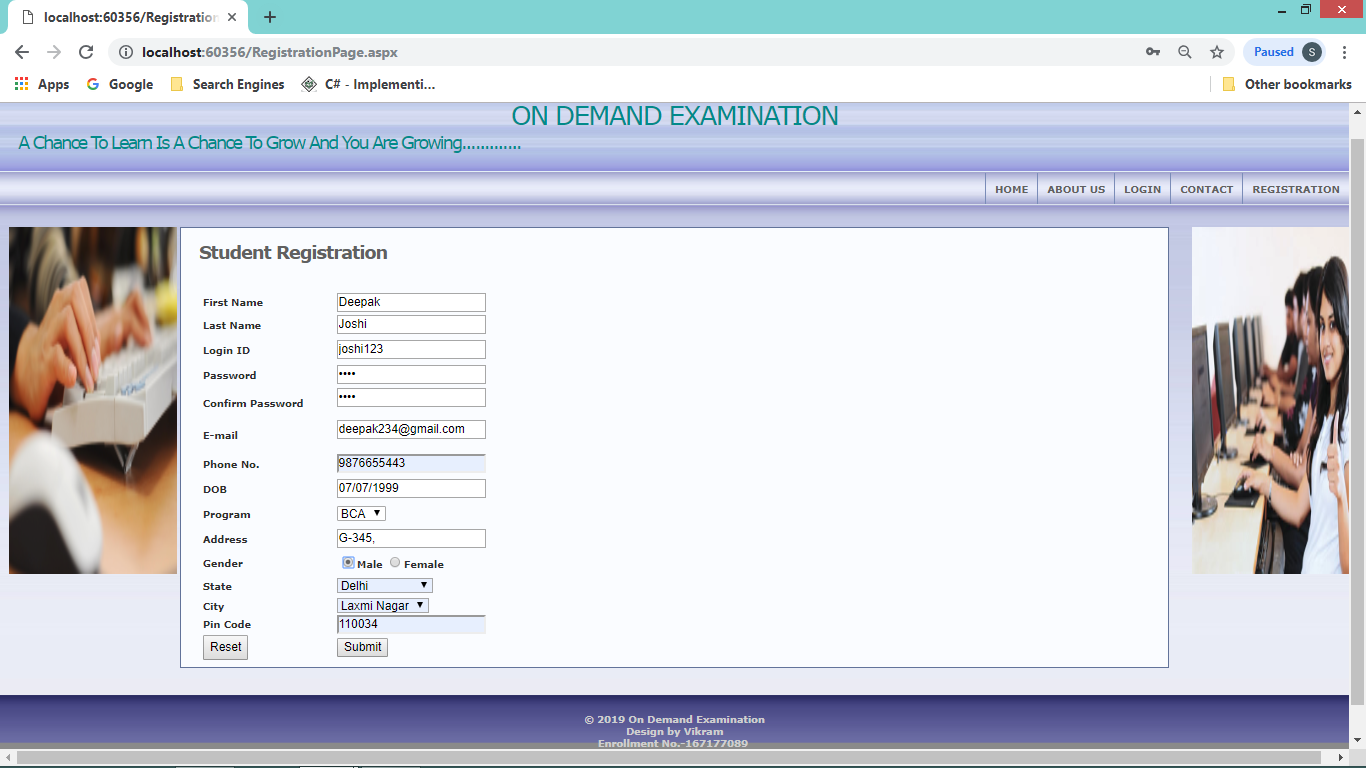
Develop and make consistent use of naming and coding standards. It not that important which standard we use, such as Hungarian notation or Pascal, Casing (First Name) or other naming conventions, as long as we use one. We should also strive for consistency in our comments and encourage liberal commenting code.

**Create effective test plan:**

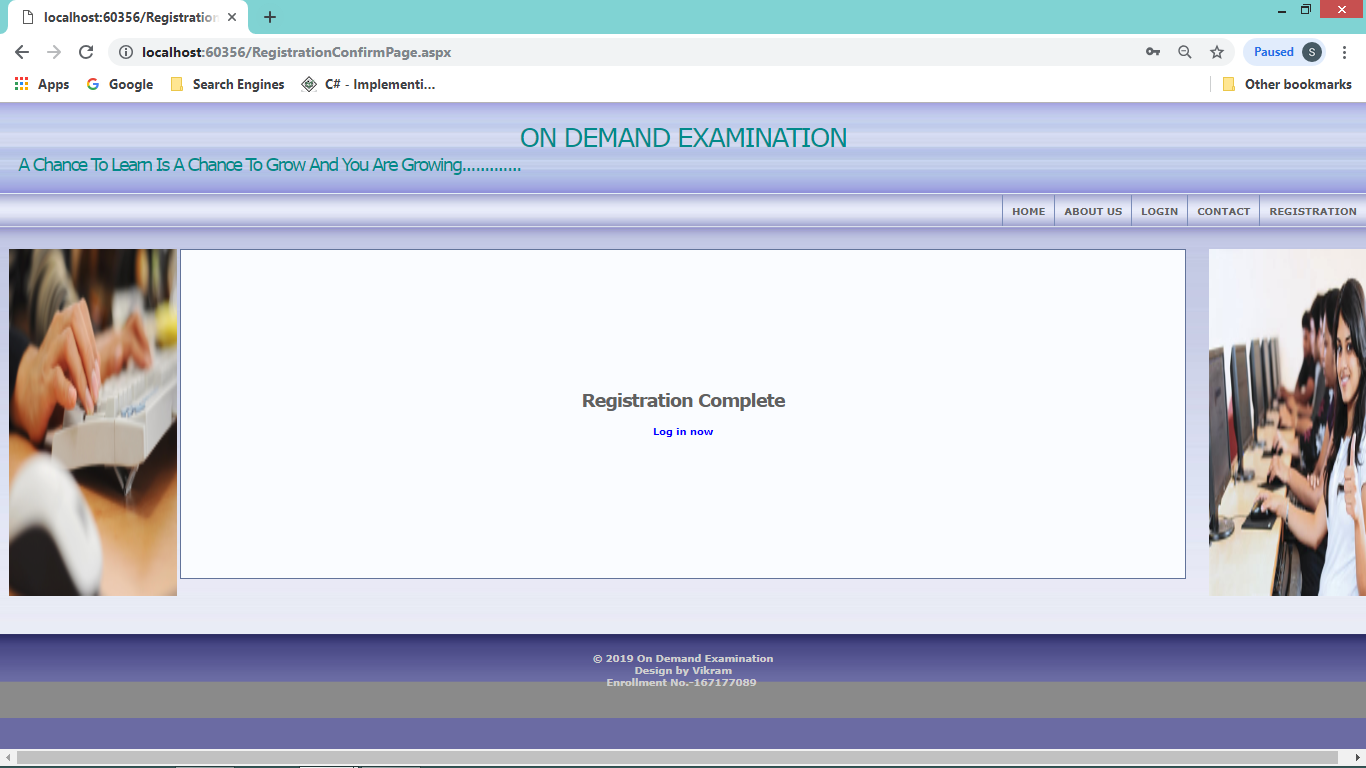
The only effective way to eliminate logic error is to test very path of your application with every possible data values that a user could enter.

This is difficult to manage without effective planning. We should create our test plan at the same time we are designing the application, and we should update these plans as you modify the application design.**Input/Output**

**Registration page Input**



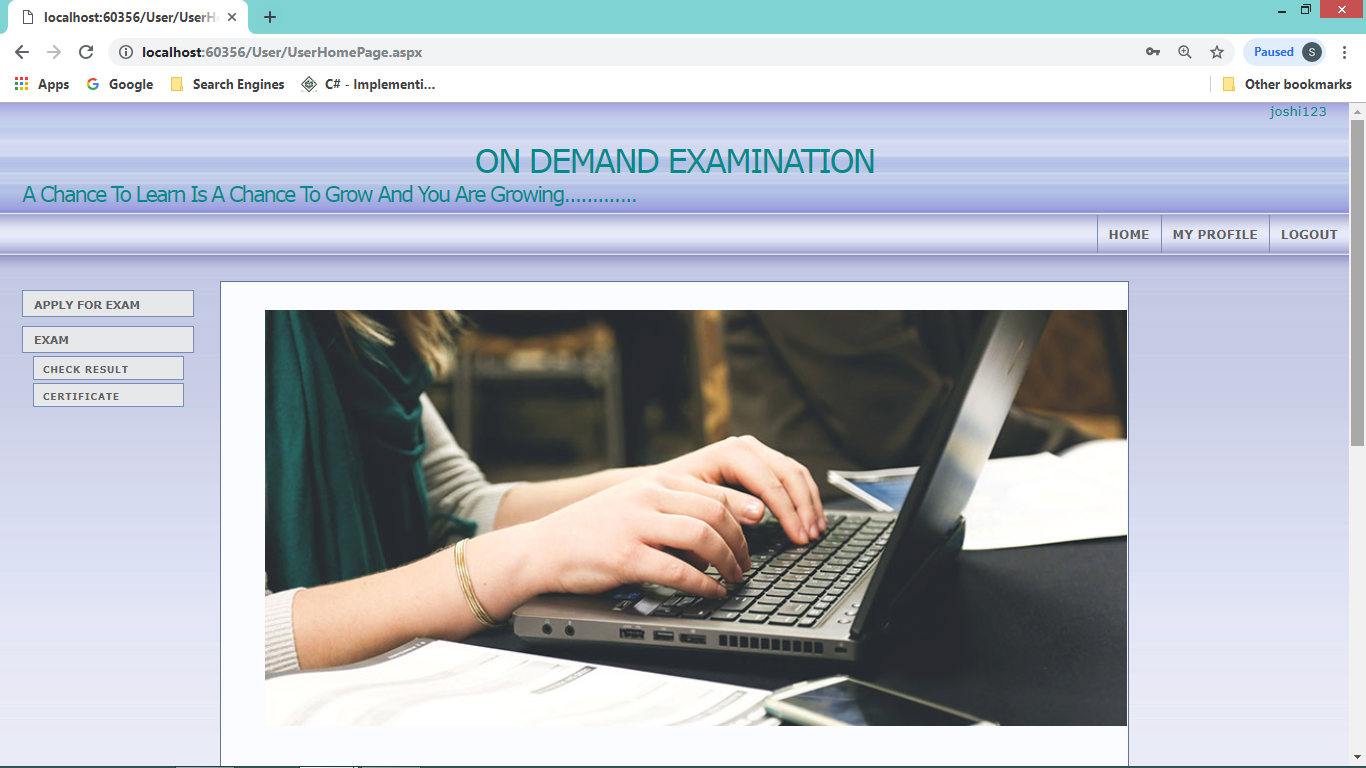
**Registration page Output**



**Log in Page Input**



**Log in Page Output**

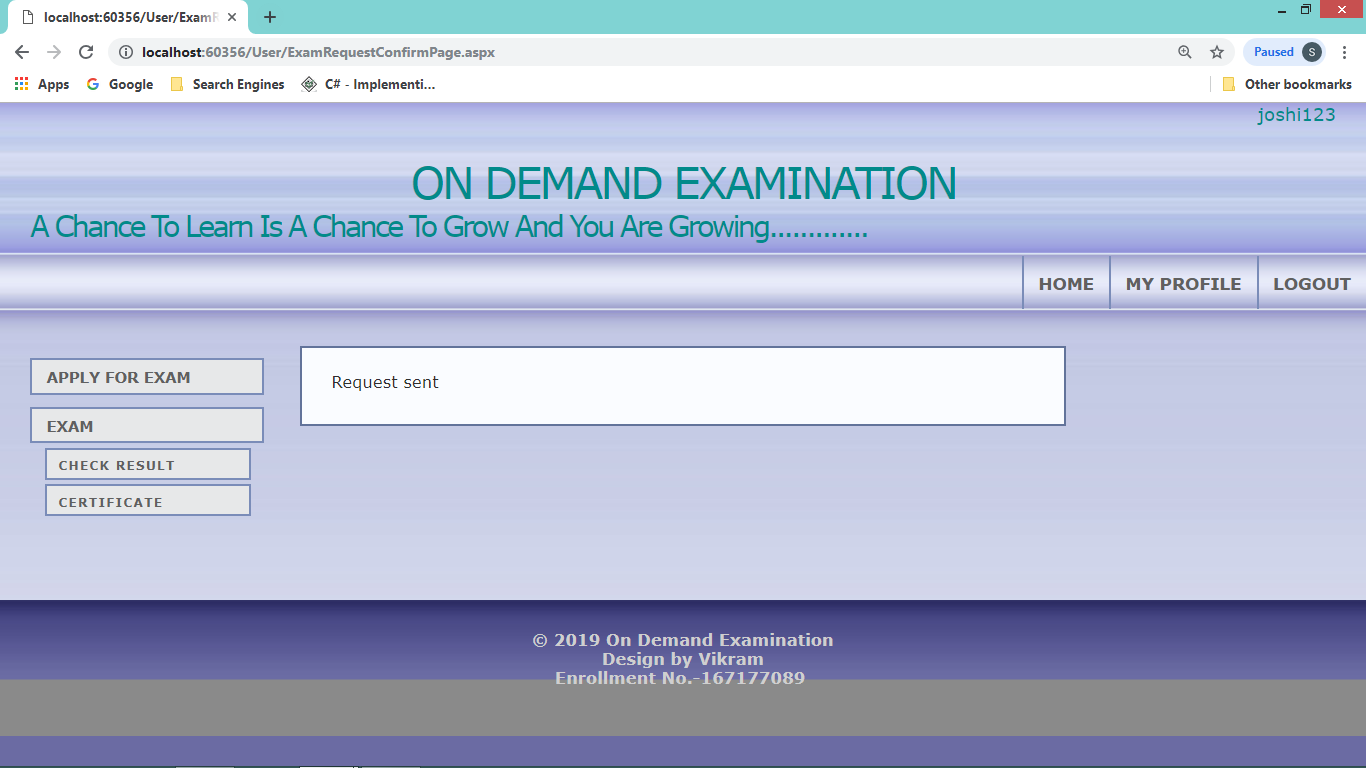


**User Input/output**

**Apply Exam Page Input**



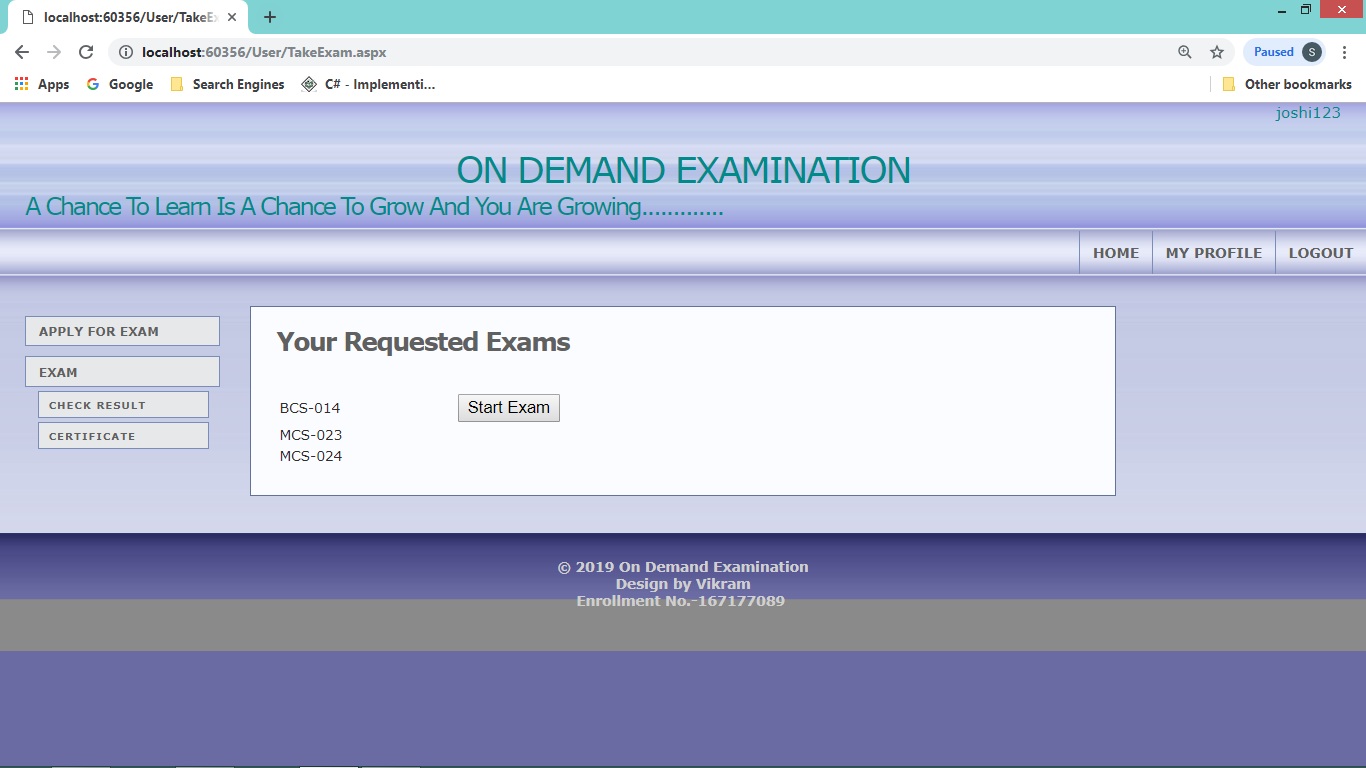
**Apply Exam Page Output**



**Take Exam Page :**

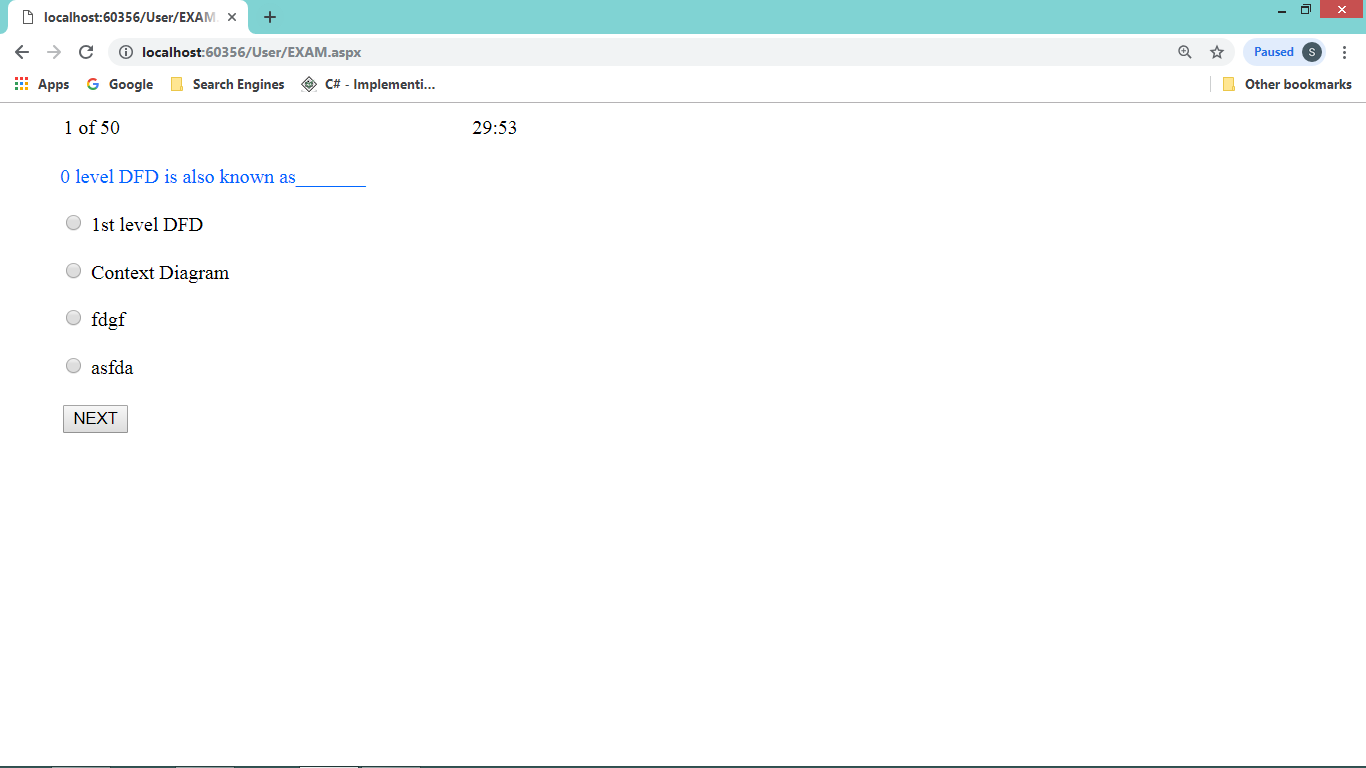
**Exam input**

**When student click on start Exam button**



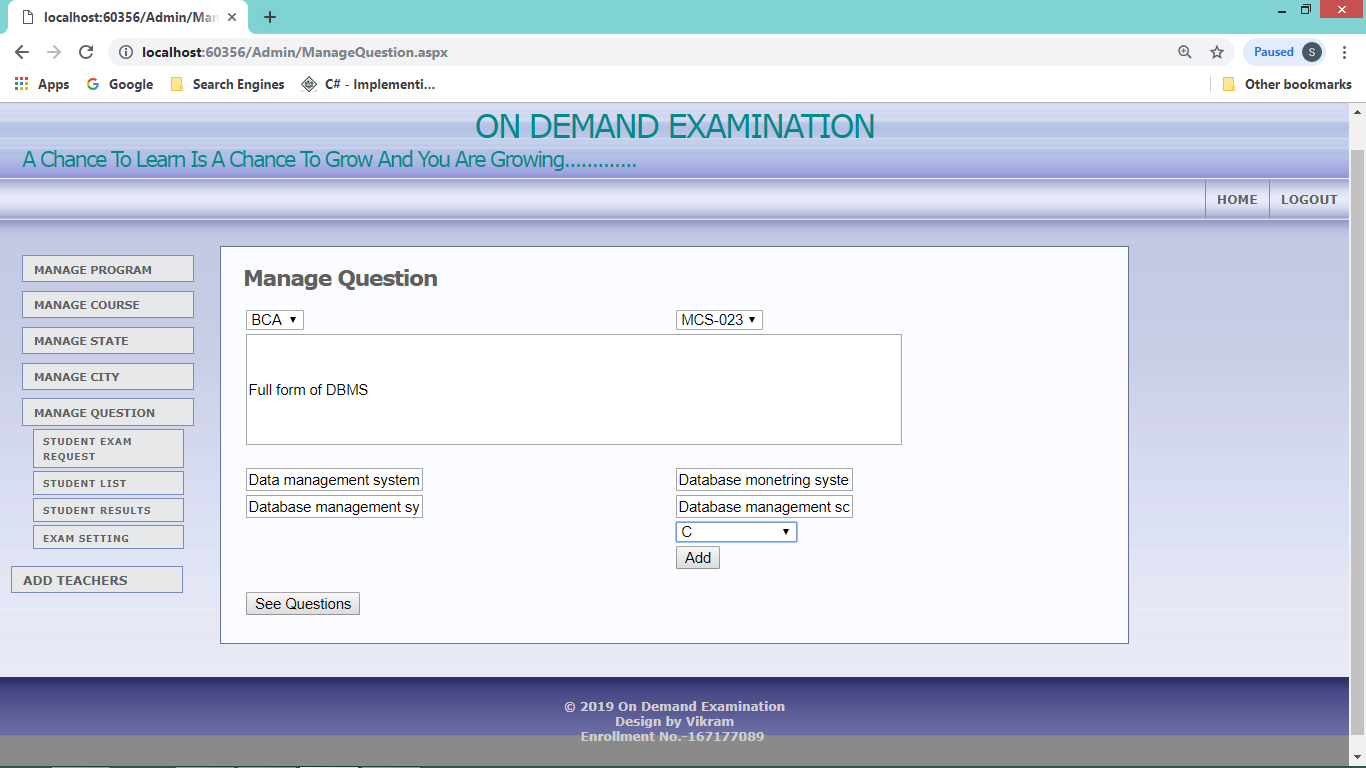
**Start Exam output**

**The start button opens Exam window**

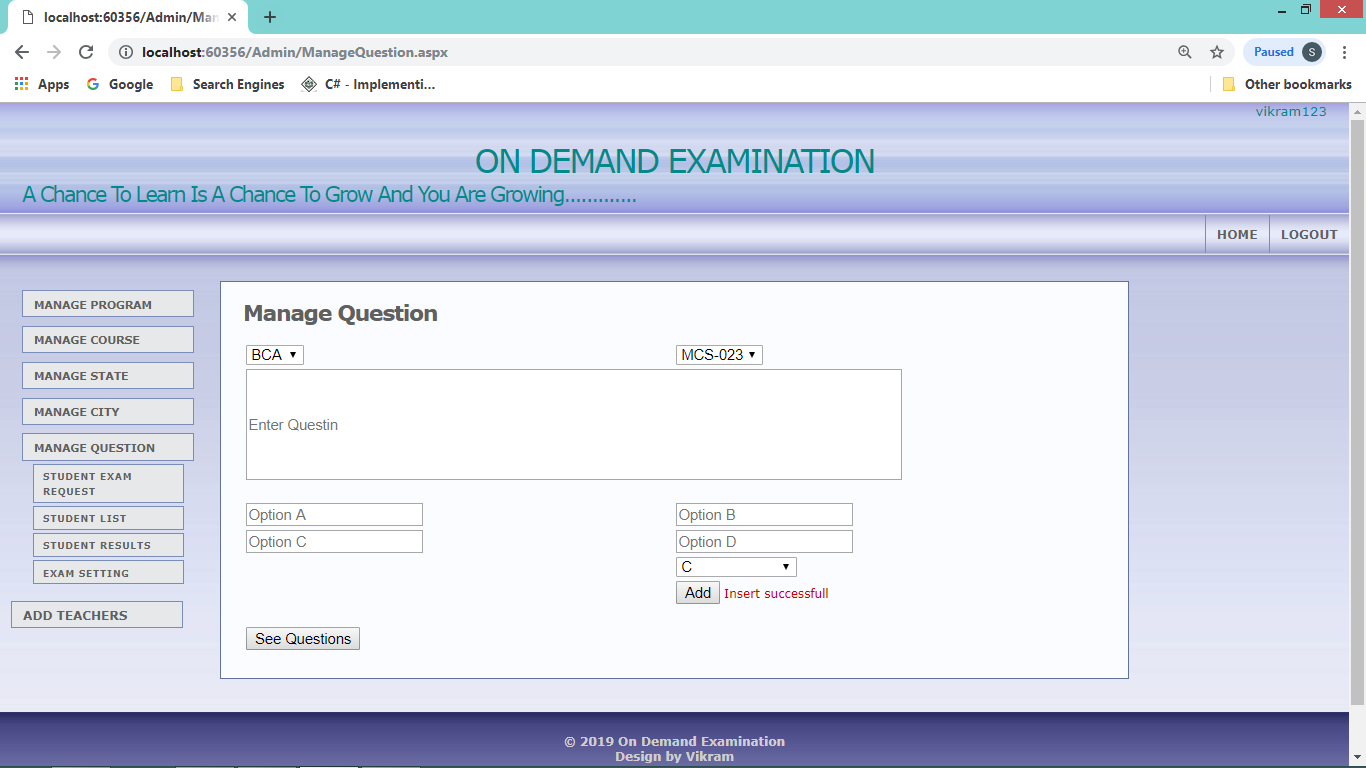


**Admin Page Input/Output**

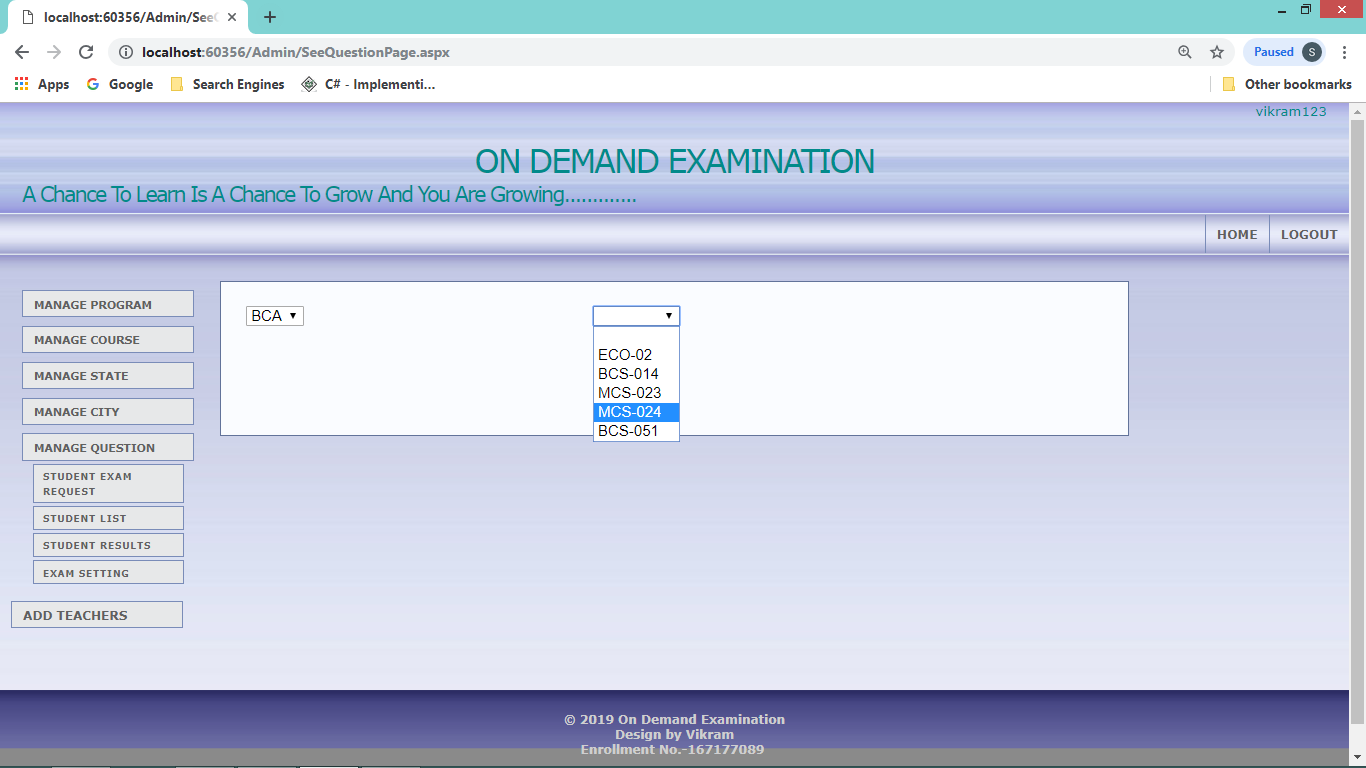
**Manage Question Input**



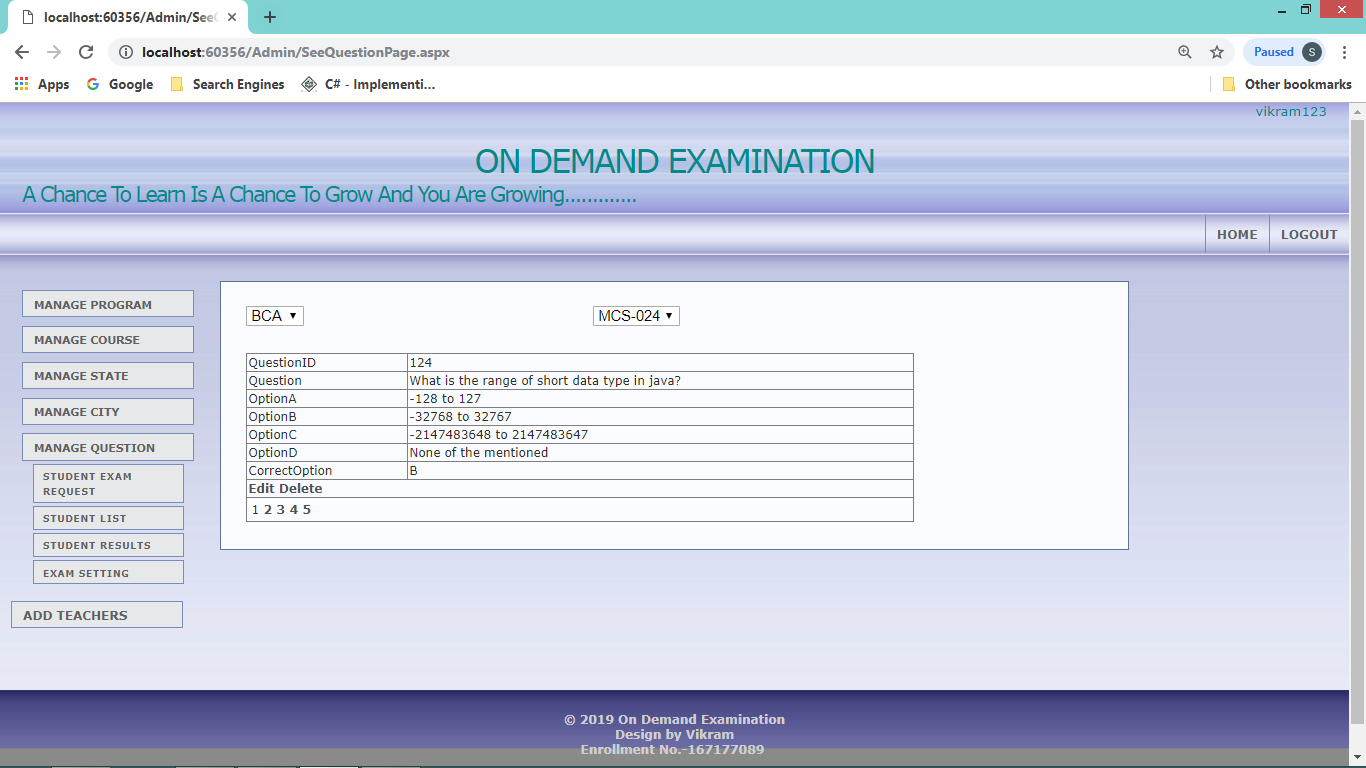
**Manage Question Output**



**See question Input**



**See question Output**



**Limitation**



* Only the student who are registered can use this website. 
* Student and teacher/admin have to login before using this website. 
* Users can not use the others information. 
* Login ID should be unique or must for sing in.

**Future Scope:**

The project has a very wide scope in future. In future the project can be implemented on internet. As the requirement arises so that project can be updated in future . the user is able to easily manage and hence run the entire work in a much better, accurate and error free manner. The following are the future scope for the project.

* The number of levels that the software is handling can be made unlimited in future from the current status of handling up to N levels as currently laid down by the software.
* Efficiency can be further enhanced and boosted up to a great extent by normalizing and de-normalizing the database tables used in the project as well as taking the kind of the alternative set of data structures and advanced calculation algorithms available.
* We can in future generalize the application from its current customized status wherein other vendors developing and working on similar applications can utilize this software and make changes to it according to their business needs. 
* Faster processing of information as compared to the current system with high accuracy and reliability. 
* Automatic and error free report generation as per the specified format with ease. 
* Automatic calculation and generation of correct and precise Bills thus reducing much of the workload on the accounting staff and the errors arising due to manual calculations .With a fully automated solution, lesser staff, better space utilization and peaceful work environment, the company is bound to experience high turnover.

**Bibliography:**

1. **An Integrated Approach to Software Engineering**: PANKAJ JALOTE
2. **Microsoft visual studio 2015** Unleashed Publication(Third Edition)
3. **HTML: *Sams* Teach Your Self**: Dick Oliver
4. **SQL Server 2012**, **Database Design Study Guide**: Kevin Hough
5. Structured Analysis and Design (**MCS-14**) of IGNOU
6. http://webdev.wrox.co.uk