**Chapter – 1**

**SYNOPSIS**

**Name of Project: Inter collegiate event management**

**1 Project Description:**

This website is used for arranging online event for various colleges which come under Mysore University. The events are scheduled by the Mysore University and the various colleges which are taking part in the event should register online using this website. After the registration, the colleges should check for the location where the event is taking place. After the competitions is held, the result will be announced. Current Student secretary information’s will be given by the respective colleges. The Mysore University will post the information details for the elections. After this, every college will nominate a candidate for the head of the overall college representative’s post of the Mysore University. Each college secretaries will vote for this post through this website.

**1.1 Present System:**

In Current system all inter college events are handled manually. Also students will find it difficult to get the required information about the events schedule like where exactly the particular event is going on.

**1.2 Proposed System:**

Proposed system will provide all the information about the events at one place, also reduces the travelling between the colleges for inviting the students to the events. This will save students time as well as money.

**1.3 Project Category:**

Web based application

**1.5 Hardware & Software requirements:**

**1.5.1 Hardware requirements:**

1.5.1.1 Processor: Intel dual core or above

1.5.1.2 Processor Speed: 1.0GHZ or above

1.5.1.3 RAM: 1 GB RAM or above

1.5.1.4 Hard Disk: 20 GB hard disk or above

1.5.1.5 Operating system: Windows XP or above

**1.5.2 Software requirements:**

1.5.2.1 ASP.NET Using C#

1.5.2.2 Server: Apache server

1.5.2.3 Database: MS SQL Server 2008

* 1. **Modules:**

**There are mainly 3 modules.**

**1.6.1University Admin**

1.6.1.1 Login

1.6.1.2 College register

1.6.1.3Manage college detail

1.6.1.4 Manage course

1.6.1.5 Update events

1.6.1.6Manage university representative’s post-election

1.6.1.7 Result display

**1.6.2 College Admin Module**

1.6.2.1 Registration

1.6.2.2 Login

1.6.2.3 Manage Student detail

1.6.2.4 Event update

1.6.2.5Nominate Candidates for University [representative’s](https://www.google.co.in/search?biw=1366&bih=662&q=college+representatives&spell=1&sa=X&ved=0ahUKEwikiNmcgN3QAhVJo48KHaiLBfgQvwUIFigA) post

**1.6.3 Student module**

1.6.3.1Login

1.6.3.2View All Event Notice

1.6.3.3 Register for fest.

1.6.3.4Vote for University [representative’s](https://www.google.co.in/search?biw=1366&bih=662&q=college+representatives&spell=1&sa=X&ved=0ahUKEwikiNmcgN3QAhVJo48KHaiLBfgQvwUIFigA) post.

1.6.3.5 View result.

**1.7 Future Scope of the Project:**

The present system is developing as web application. In future we would like to develop it for portable devices like cell phones, wrap or GPRS connections.

**1.8 Conclusion:**

The project work titled “Inter collegiate event management” has been designed using Asp.net and SQL server where in many user friendly form controls and inputs have been added in order to make it a user interactive application. The system is developed in such a way that the user with common knowledge of computers can handle it easily. All information about the inter college events and election of the student security and joint security of the various elections.

This project helps us to improve the knowledge in ASP.net and SQL server. The importance of a good software design was learned during the project.

**1.9 Reference:**

The needed requirement for this project has been obtained from websites like Wikipedia, google.com etc.

**Chapter – 2**

**SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 INTRODUCION:**

The software product is initiated by client’s needs. In beginning, these needs are in the minds of the various people of the client organization. The requirement analyst has to identify the requirements by talking into a formal document. Software Requirements Specification (SRS) documents is a documents that completely describes ‘WHAT’ the software must do without describing how the software will do it. SRS describes the complete external behaviour of the proposed software.

Software requirement specification (SRS) is a document, which describes completely the external behaviour of the software. The first and foremost work a software developer is to study the system to be developed and specify the user requirements before going for the designing phase. This document will let us know how this system behaves and responds.

**2.1.1 PURP0SE:**

The purpose of this document is to serve as a guide to the developers and tester who are responsible for the development of the system. The mission of Inter collegiate event management is to arrange online event college which come under Mysore university. The main purpose of the automation of online inter college is save time and money of travel.

The basic purpose of the SRS is to bridge the gap between the client(user) and developer. Another important purpose of developing the SRS is to help clients to understand their own needs. There are several problems in gathering the requirement. Changing requirements is an irritant task for software developers and may lead to bitterness among client and the developers.

**2.1.2 SCOPE:**

The present system is developing as web application. In future we would like to develop it for portable device like cell phones, WAP or GPRS connections.

This SRS describe the requirements of the system. College under the Mysore university they can send online invitation for fest to other college and student can register online for fest. University can have conducted the election through this website the voting process can done through online. event Result are also shown through this web site.

**2.1.3 DEFINITION, ACRONYMS, AND ABBREVIATION**

DFD: Data Flow Diagram

E-R: Entity Relationship Diagram

SRS: System Requirement Analysis Specification

SQL: Structured Query Language

RAM: Random Access Memory

**2.1.4 DOCUMENT OVERVIEW:**

1. Inter collegiate event management project make easy ways to connect university and various college under it through this website.
2. University can add the college and give the code to each college. After that interested college can register to this website under the Mysore university.
3. The university can conduct the election through this website and all work of this election can be taken place in this website and voting process also can be take place through online and result will display.
4. The event which are going to held under the university and various college can announce through this
5. Website and student can register their name for events and result are also display of various event took place.
6. The college can update and give the comment upon events and college can share their pics of events held on their college.

**2.2 FEASIBILITY STUDY:**

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine whether it would be technically feasible to develop the website. The feasibility study involves activity of the admin student and college admin. Here we get the information about the events and election and result of its.

**Operational:**

Operational feasibility measures how well the solution will work in the organization and how end-user and management will work feel about system. On studying the feasibility of the project, the following conclusion could be delivered.

* Developed system will provide the adequate throughput to end-users.
* It will provide advantage and reliable services.
* If it can be instilled within the suitable environment, system will do operation under environment of limited resources. Thus it is operationally feasible to develop the proposed system.

**Technical:**

Technical feasibility tries to answer the following question to make the software feasible develop.

* The software or tools necessary for building or running the application are easily available or not?
* The compatibility amongst software exist or not
* Are developers aware of these technologies?

**Economic**

While defining the economic feasibility, the following things were taken into the considerations.

* Economically feasible from development point of view? As the company deals with projects in the same date with project in the same database server, the system is feasible from the development point of view also.

**2.3 OVERALL DESCRIPTION:**

**2.3.1 PRODUCT PERSPECTIVE:**

Inter collegiate event management system has three types of users. The different roles can be stated as; university admin, college admin, student. The event is scheduled by the Mysore University are managed by admin and the various college admin which are taking part in the event should register. Every student can see the entire event updated by university admin. After being logged in user can do basic operations according to their permission.

**2.3.2 PRODUCT FUNCTIONS:**

This project includes graphics user interface (GUI)with simple menus, which help the user. Here the admin plays the major roll of controlling this website. It can be managed by university admin, student, college admin for event details and election information etc.

**2.3.3 USER CHARACTERISTICS:**

The system has three user university admin, college admin and student. University Admin has all the privileges like adding course detail, publishing event notice, viewing all student details and also view all repots. Admin has provision to announce university election through this system. Admin can also make any change to the data after the details are saved. The college admin has privilege of adding college information topmost candidate detail for election. Also can manage event scheduled by admin. Student can view the entire event published by admin and student can register to their name for to participate in any inter -college event, student also have permission to vote other candidate. This website is easy to use for admin and students.

**2.3.4 GENERAL CONSTRAINS:**

2.3.4.1 Requires all the mandatory fields to be filled with proper information.

2.3.4.2 The system must be user must friendly.

2.3.4.3 This system must work in pc’s having visual studio 2010 and SQL server 2008.

2.3.4.4 The system should have windows 7 or higher operating system on it.

2.3.4.5 Login and password must be used for authentication of the user.

**2.3.5 ASSUMPTION AND DEPENDENCIES:**

2.3.5.1 User should be familiar to use websites.

2.3.5.2 The system is completely dependent on internet connection.

2.3.5.3 It is assumed that enough that the admin and student will have enough knowledge

About website.

2.3.5.6 It is assumed that browser and operating system will support, NET framework 4.0.

**2.4 SPECIFIC REQUIREMENTS:**

**2.4.1 EXTERNAL INTERFACE REQUIREMENTS:**

It specifies all the interface of the system to people, other system, hardware and other software

**2.4.1.1 USER INTEFACE:**

2.4.1.1.1 The user can access the site through a web browser.

2.4.1.1.2 Home page which has links to other pages.

2.4.1.1.3 User can easily navigate through various contents of menus.

**2.4.1.2 HADRWARE INTERFACE:**

2.4.1.2.1 Any device which has web browser installed in it.

2.4.1.2.2 RAM: 1 GB RAM or above.

2.4.1.2.3 Hard disk: 20 GB hard disk or above.

2.4.1.2.4 Processor: Intel dual core or above.

**2.4.1.3 SOFTWARE INTERFACE:**

2.4.1.3.1 Microsoft visual studio 2010.

2.4.1.3.2 Any web browser.

**2.4.1.4 COMMUNATION INTERFACE:**

2.4.1.4.1 TCP/IP.

**2.4.2 FUNCTIONAL REQUIRMENTS:**

Functional requirements specify which output should be proper for the given inputs. All inputs are entered according to the data type and no blank are allowed for mandatory fields Invalid inputs are not allowed in the system. It prompts to re-enter the data. Appropriate error message is displayed.

**2.4.2.1 LOGIN**

2.4.2.1.1 Admin login.

2.4.2.1.2 College admin login.

2.4.2.1.3 Student login.

**2.4.2.2 University ADMIN**

2.4.2.2.1 Manage events.

2.4.2.2.2 Change password.

2.4.2.2.3 View reports.

2.4.2.2.4 Add view course.

2.4.2.2.5 View university head election.

2.4.2.2.6 View student information.

**2.4.2.3 COLLEGE ADMIN**

2.4.2.3.1 College registration.

2.4.2.3.2 Student registration.

2.4.2.3.3 Manage events.

2.4.2.3.4 Nomination for university held election.

2.4.2.3.5 Update profile.

2.4.2.3.6 Change password

**2.4.2.4 STUDENT**

2.4.2.4.1 Login.

2.4.2.4.2 View events.

2.4.2.4.3 Register name for fest.

2.4.2.4.4. Vote for security candidate

2.4.2.4.5 View event result and election result.

**2.4.3 PERFORMANCE REQUIRMENTS:**

The performance of the overall system should be faster and error free, with built in error checking and correction facilities.

In order to run this application, we require.

2.4.3.1 An internet with minimum 56 kbps bandwidth.

2.4.3.2 To access this page we require IE6 or any version browser.

2.4.3.3 .NET frame work is required.

**2.4.4 DESIGN CONSTRAINS:**

2.4.4.1 Require specifying the information for all the mandatory fields.

2.4.4.2 The application shall have relational database.

2.4.4.3 The application shall be implemented using SQL server 2008 and Microsoft visual studio.

2.4.4.4 The application shall display error message to the user when an error is detected.

**2.4.5 SOFTWARE SYSTEM ATTRIBUTES:**

These sections of the online event management System SRS describe the application attributes and properties.

2.4.5.1 Availability: online retail management system shall be available in internet 24x7 and

Capable of supporting a multiple login.

2.4.5.2 Security: online retail a system shall be managed by the admin.

2.4.5.3 Maintainability: during maintains stage the SRS can be referred for the validation.

2.4.5.4 Portability: since it’s a system, it is portable.

2.4.5.5 Timeliness: the system carries out all the operation with consumption of every less time.

**2.4.6 OTHER REQUIREMENTS:**

**2.4.6.1 NON-FUNCTIONAL REQUIRMENTS:**

Non-functional requirement defines the needs in terms of the performance logical DB

Requirements, design constrains, standards compliances, reliability, availability, security,

Maintainability and portability.

**2.4.6.2 RELIBILITY:**

2.4.6.2.1 Reports generated can be saved for feature reference.

2.4.6.2.2 Save time travelling to the vendor place.

2.4.6.2.3 Reports for the college, student, and university on daily, monthly, yearly bases.

2.4.6.2.4 Good validations for user inputs will be done.

**Chapter-3**

**SYSTEM DESIGN**

**3.1 INTRODUCTION**

The purpose of the design phase is to find a solution to the problem specified in the requirement document. This is the first step in moving from problem domain to solution domain.

The design of a system is perhaps the most critical factor affecting the quality of the software. It has a major impact on the later phases is the design document. This document is similar to a blueprint or a plan for the solution and is used later during implementation testing and maintenance.

The design activity is often divided into separate phases - system design and detailed design. System design is sometimes also called as top level design. This system design aims to identify the modules that should be present inside the system, specifications of these modules, and how they interact with each other to produce the desired output. At the end of the system design all the major data structures, file formats, and the major modules in the system and their specification are decided.

**3.2 OVERVIEW**

This website is used for arranging online event for various colleges which come under Mysore University. The events are scheduled by the Mysore University and the various colleges which are taking part in the event should register online using this website.

The primary purpose of the system is to implement the above stated functionalities. It will have user friendly GUI's that will guide the user to easily achieve the same. The application shall also have some report displaying in the same window.

In addition to this application features to generate different kinds of reports. This application is fully developed under SQL server using ASP.Net.

**3.3 FUNCTIONAL DECOMPOSITION**

Functional decomposition refers broadly the process of resolving a functional relationship into its constituent parts in such a way that the original function can be reconstructed from those parts by function composition. In general, this process decomposition is undertaken either for the purpose of gaining insight into the industry of the constituent components or for the purpose of obtaining a compressed representation of the global function, a task which is feasible, and a task which is feasible when the constituent process has a certain level of modularity.

**3.4 FUNCTIONAL COMPONENT**

**FUNCTIONAL COMPONENT 1: LOGIN MODULE:**

**Introduction: This module allows valid admin or student or college administrator to access the functionalities provided by the website.**

**Input: student or admin or college admin can logon to the website by entering username and password.**

**Output: Login successful and student or admin or college admin is allowed to log in to the website. Otherwise it will display an error message.**

**FUNCTIONAL COMPONENT 2: COLLEGE MODULE:**

**Introduction: This module allows college to register themselves into the website and then get approved by admin.**

**Input: college name, address, contact number, email id, doc proof etc.**

**Output: College information will be added in the database. Otherwise it will display an error message.**

**FUNCTIONAL COMPONENT 3: STUDENT MODULE:**

**Introduction: This module allows Colleges register all the students of their college.**

**Input: Student name, address, reg. number, contact no, etc.**

**Output: student information will be added into database.**

**FUNCTIONAL COMPONENT 4: EVENT MODULE:**

**Introduction: This module allows admin to post all event information.**

**Input: event name, date, event detail, etc.**

**Output: Event information will be updated into respective table.**

**FUNCTIONAL COMPONENT 5: EVENT SCHEDULE MODULE:**

**Introduction: Event details will be updated by respective college admin.**

**Input: event date, location, description etc.**

**Output: Event detail will be added to the database.**

**FUNCTIONAL COMPONENT 6: EVENT RESULT MODULE:**

**Introduction: Event result will be updated by respective college admin.**

**Input: name of the participate, in which he won and which place he taken.**

**Output: Event result will be added to the database**

**FUNCTIONAL COMPONENT 7: ELECTION MODULE:**

**Introduction: inert college Student council election details will be updated using this module by university admin.**

**Input: post detail, election date, result date etc.**

**Output: election details will be added to the database.**

**FUNCTIONAL COMPONENT 8: CANDIDATE MODULE:**

**Introduction: In this module college administrator will Nominate candidate details for election.**

**Input: candidate name, reg. no, contact number, photo, id proof, postid**

**Output: Candidate details will be added to the database.**

**FUNCTIONAL COMPONENT 9: ELECTION RESULT MODULE:**

**Introduction: Election result will be adding by the university admin.**

**Input: name of the candidate who won the election.**

**Output: Election result will be added to the database**

**FUNCTIONAL COMPONENT 10: CHANGE PASSWORD:**

**Introduction: Change password menu will help student/admin/college admin in changing his password.**

**Input: Enter the Current Password, new password, confirm password.**

**Output: Password will be changed.**

**FUNCTIONAL COMPONENT 11: UPDATE PROFILE:**

**Introduction: Student/ College Admin can update his/her profile using this module.**

**Input: name, contact number, photo etc.**

**Output: Student/ College Admin Profile will be updated**

**FUNCTIONAL COMPONENT 12: COURSE MODULE:**

**Introduction: University head or Administrator will update course detail using this module**

**Input: course name.**

**Output: course detail will be updated into database.**

**FUNCTIONAL COMPONENT 13: LOGOUT MODULE:**

**Introduction: This module allows the user to logout from their account.**

**Input: Click logout button.**

**Output: User will be logged out.**

**3.5 DESCRIPTION OF THE COMPONENTS**

**The notations used in the following diagram are:**

**DATAFLOW DIAGRAM NOTATION:**

|  |  |
| --- | --- |
|  | **Function: The bubble represents a process or transformation that is applied to the data which changes in some way. Each bubble is a staff of the information.** |
|  | **File:It represents the repository data that is to be stored for used by one or more process.** |
|  | **Input/output: This rectangle is called an entity, which represent the staff of the information.** |
|  | **Flow: The arrow represents the dataflow. The arrow indicates the direction of the dataflow. All arrows in the dataflow diagram are labeled.** |

**3.5.1 CONTEXT FLOW DIAGRAM (CFD)**

A context flow diagram (CFD) is a general representation of the "flow" of context through an information system. CFD's can also use for the visualization of data processing (Structured Design).

On a CFD data item can flow from an external data source or internal data store to an internal data store, or to an external data sink via an internal process.

A CFD provides no information about the training or ordering of processing, or about whether processes will operate in sequence or in parallel. It is therefore quite different from flowchart, which shows flow of control through an algorithm.

Allowing reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and from the system, or where the data will come from and go to, or where the data will be stored.

**CFD(CONTEXT FLOW DIAGRAM )**

**MANAGE MANAGE**

COLLEGE ADMIN

ADMIN

**REGISTERION REGISTERTION**

**FOR EVENTS AND ELECTION FOR EVENTS**

STUDENT

**3.5.2 DATA FLOW DIAGRAMS**

Data flow diagrams are a graphical representation of the “flow “of data through an information system. Data flow diagrams are commonly used during problem analysis. It views a system as a function that transforms the input into desired output. A DFD shows movement of data through the different transformation or process in the system, with the help of various levels in a crystal clear way.

A DFD representations flow of data through a system. Data flow diagrams are commonly used during problem analysis. It views a system as a function that transforms the input into desired output. A DFD shows movement of data through the different transformation or processes in the system.

**There are four kinds of system components:**

1. Processes
2. External entities

Data store

**LEVEL-1(ADMIN AND COLLEGE)**

MANAGELOGIN

ADMIN

LOGIN REGISTRY

USERNAME VALID OR

PASSWORD INVALID

RETRIVES

**RE**

**LEVEL-2(WORKING OF ADMIN)**

**ADD**

TBLADMIN

ADMINID

POSTEVENTID  **ADD** TBLEVENT TBLEVENT

ADMIN

ANNOUNCE

**ADD** TBLELECTION

VREFIETION

COLLEGE ADMIN

REGISTRA ADD TBLCOLLEGE

**LEVEL-3(EVENT ARE TOOK PLACE BY ADMIN AND COLLEGE ADMIN)**

POSTID, YEAR, DETAIL

ADMIN

POSTNAME TBLELECTION

**L** STUID, NAME DETAIL TBLSTUDENT

COLLEGE ADMIN

NOMATION CANDIDATE DETAIL TBLPOSTCANDIDATE

CANDIDATE NAME, PHOTO

EVENTID, ENAME, DESCETION UPDATE

TBLEVENT

ADMIN UPDATE RESULT COLLEGE UPDATE RESULT

ADMIN UPDATE RESULT

OF ELECTION

**LEVEL-4(WORKING OF STUDENT)**

**USERNAME, PASSWORD**

**TBLSTUDENT**

**LOGIN SUCARS**

**VIEW TBLEVENT**

**REG FOR FEST**

STUDENT

**VIEW VIEW TBLCANDIDATE**

**CANDIDATE, POSTID**

**POSTID, STUDID, TBLRESULT**

**VOTE**

**CHAPTER – 4**

**DETAILED DESIGN**

**4.1** **INTRODUCTION**

Detailed design starts after the system design phase and system has been certified through the review. The goal of this phase is to develop the internal logic of each of the modules identified during system design. In the system design, the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for the modules. In other words, in system design attention is on what components are needed, while in the detailed design how the components can be implemented in the software is the issue, the design process for software system has two eves. At the first level focus is on deciding which modules are needed for the system, the specification of these modules and how the modules should be interconnected. This is called system design or top level design. In the specification of the module can be satisfied is decided. This design level is often called detailed design.

Because the detailed design is extension of system design, system design controls the major structural characteristics of the system. The system design has a major impact testability and modifiability of a system and impacts its efficiency much of the design efforts for the designing software are spent creating the system design.

**4.2 APPLICABLE DOCUMENTATION:**

The detailed design refers the system document hence the first applicable document here is system design. Also we are referring the data structure. Hence second applicable document here is database design.

**4.3 DATABASE DESIGN:**

Data Design is used to manage large amount of information. The management of data involves both the definition of structure for storage and provision for the manipulation of information.

**4.3.1 TABLE STRUCTURE:**

Database is a collection of information and data systematically stored in tables the application “Inter collegiate event management “consists of 17 tables. Table is a list of information organized into fields. Usually each field has a field name, data type with fixed length and description.

4.3.3.1 Field name tells the name of the field.

4.3.3.2 Data type’s property tells the type of data stored in that field and the length is

Specified, so that the maximum length is set.

4.3.3.3 Attribute property is mentioned so those mandatory fields are mentioned.

4.3.3.4 Description field tells the detail of field.

**4.3.1.1 Table name: tbl\_admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fileds** | **Data type** | **Constraints** | **Description** |
| **Adminid** | **Int** | **Primary key** | **To store the ID of the college.** |
| **Full name** | **Varchar(20)** | **Not Null** | **To store admin name** |
| **Uname** | **Varchar(20)** | **Not Null** | **To store admin username** |
| **Upass** | **Varchar(20)** | **Not Null** | **To store admin password** |

**4.3.1.2 Table name: tbl\_college**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data type** | **Constraints** | **Description** |
| **Collegeid** | **Int** | **Primary key** | **To store the ID of the college.** |
| **college name** | **Text** | **Not Null** | **To store college name** |
| **College\_code** | **Varchar(20)** | **Not Null** | **To store admin username** |

**4.3.1.3 Table Name: tblcolg\_detail**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **Clinfo** | **int** | **Primary key** | **To store information of the college.** |
| **Colg\_id** | **int** | **Forging key** | **To store colg id of the college.** |
| **Address** | **Varchar(MAX)** | **Not Null** | **To store the address of the college.** |
| **Contact** | **Varchar(20)** | **Not Null** | **To store the contact of the college.** |
| **Contact2** | **Varchar(20)** | **Not Null** | **To store the alternative no.** |
| **Email id** | **Varchar(20** | **Not Null** | **To store the college email id.** |
| **Photo path** | **Image** | **Not Null** | **To store the college image.** |
| **Colg\_password** | **Varchar(20)** | **Not Null** | **To store the college password.** |
| **Status** | **Varchar(20)** | **Not Null** | **To store status of the college the college is active or inactive.** |
| **Colg\_code** | **Varchar(20)** | **Not Null** | **To store the unique code of college.** |

**4.3.1.4 Table Name: tblevent\_notice**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **event\_id** | **Int** | **Primary key** | **To store the ID of the event.** |
| **event\_name** | **Varchar(MAX)** | **Not Null** | **To store the name of the event.** |
| **event\_desc** | **Text** | **Not Null** | **To store the event description.** |
| **event\_date** | **Date** | **Not Null** | **To store the event date.** |
| **event\_file** | **Varchar(20)** | **Not Null** | **To store pdf document of event.** |
| **Clog id** | **Int** | **Not Null** | **To store the college id.** |
| **event\_type** | **Varchar(20)** | **Not Null** | **To store the event type.** |
| **Status** | **Varchar(20)** | **Not Null** | **To store the status of the event.** |

**4.3.1.3.5 Table Name: tblevent\_result**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **Edid** | **Int** | **Primary key** | **To store event id.** |
| **event\_id** | **Int** | **Forging key** | **To store event id.** |
| **event\_schedule** | **Text** | **Not Null** | **To store event schedule.** |
| **event\_result** | **Text** | **Not Null** | **To store event result.** |

**4.3.1.6 Table Name: tblelection**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **Eid** | **Int** | **Primary key** | **To store the ID of the election.** |
| **Epost** | **Varchar(250)** | **Not Null** | **To store the election post.** |
| **Edisc** | **Text** | **Not Null** | **To store the election description.** |
| **e\_date** | **Date** | **Not Null** | **To store the election date.** |
| **result \_date** | **Date** | **Not Null** | **To store the result date.** |
| **Eyear** | **Varchar(20)** | **Not Null** | **To store the election year.** |

**4.3.1.7 Table Name: tblelection\_detail**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **postid** | **int** | **Primary key** | **To store post id.** |
| **Eid** | **Int** | **Forging key** | **To store the election ID.** |
| **post\_name** | **Varchar(150)** | **Not Null** | **To store post name.** |
| **post\_desc** | **Text** | **Not Null** | **To store post description.** |
| **status** | **Varchar(20)** | **Not Null** | **To store status of the election detail.** |

**4.3.1.8 Table Name: tblcandidate**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **candidateid** | **Int** | **Primary key** | **To store candidate id.** |
| **Candidate name** | **Varchar(250)** | **Not Null** | **To store the candidate name.** |
| **studid** | **Varchar(20)** | **Not Null** | **To store studid.** |
| **Candidate\_idcard** | **Varchar(20)** | **Not Null** | **To upload candidate id card.** |
| **clog id** | **Int** | **Forging key** | **To store clog id.** |
| **Post id** | **Int** | **Forging key** | **To store candidate post id.** |

**4.3.1.9 Table Name: tblresult**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **ID** | **Int** | **Primary key** | **To store id.** |
| **Post id** | **Int** | **Not Null** | **To store post id.** |
| **Student id** | **Int** | **Forging key** | **To store student id.** |
| **candidateid** | **int** | **Forging key** | **To store candidate id.** |

**4.3.1.1.0 Table Name: tblcourse**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **Course id** | **Int** | **primary key** | **To store course ID.** |
| **Course name** | **Varchar(250)** | **Not Null** | **To store the course name.** |

**4.3.1.1.1 Table Name: tbl\_student**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **colgid** | **Int** | **Primary key** | **To store colgid.** |
| **studname** | **Varchar(250)** | **Not Null** | **To store the studname name.** |
| **regno** | **Varchar(20)** | **Not Null** | **To store regno.** |
| **gender** | **Varchar(20)** | **Not Null** | **To store gender.** |
| **Admission\_year** | **Int** | **Not Null** | **To store admission year.** |
| **courseid** | **Int** | **Forging key** | **To store courseid.** |
| **Contactno** | **Varchar(20)** | **Not null** | **To store contactno.** |
| **Email\_id** | **Varchar(20)** | **Not null** | **To store email\_id.** |
| **Stud\_pass** | **Varchar(10)** | **Not null** | **To store stu\_password.** |

**4.3.1.1.2 Table Name: tblcertificate**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **fieldid** | **Int** | **Primary key** | **To store id.** |
| **eventid** | **Int** | **Not Null** | **To store post id.** |
| **certificate** | **Varchar(20)** | **Not null** | **To store student id.** |
| **colgid** | **Int** | **Forging key** | **To store colg id.** |

**4.3.1.1.3 Table Name: tblcomment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **fid** | **Int** | **Primary key** | **To store id.** |
| **posteby** | **Int** | **Not Null** | **To store post id.** |
| **posteddon** | **Int** | **Not null** | **To store student id.** |
| **Comment** | **Varchar(30)** | **Not null** | **To store comment.** |
| **eventid** | **Int** | **Forging key** | **To store event id.** |

**4.3.1.1.4 Table Name: tblevent\_reg**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **pid** | **Int** | **Not null** | **To store participated id.** |
| **stuid** | **Int** | **Forging key** | **To store stuid.** |
| **eventid** | **Int** | **Forging key** | **To store eventid.** |
| **colgid** | **Int** | **Forging key** | **To store colgid** |

**4.3.1.1.5 Table Name: tbleventparticipte**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **id** | **Int** | **Primary key** | **To participated id.** |
| **stuid** | **Int** | **Forging key** | **To store student id.** |
| **eventid** | **Int** | **Forging key** | **To store event id.** |

**4.3.1.1.6 Table Name: tblgallery**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **galleryid** | **Int** | **Primary key** | **To storegalleryid.** |
| **photolink** | **Int** | **Not Null** | **To store photo.** |
| **eventid** | **Int** | **Forging key** | **To store event id.** |

**4.3.1.1.7 Table Name: tblresult**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Data Type** | **Constraints** | **Description** |
| **id** | **Int** | **Primary key** | **To store id.** |
| **postid** | **Int** | **Forging key** | **To store post id.** |
| **candidateid** | **Int** | **Forging key** | **To store candidate id.** |
| **stuid** | **Int** | **Forging key** | **To store student id.** |

**4.4. ER DIAGRAM:**

E-R model is a general data model, as is the relational model that underlies SQL. It provides another way of thinking about and organizing data. The E-R model is used to derive an abstract model of the data i.e. the implemented with a set of tables that conform to relational principles. E-R diagram enables to represent the conceptual design of a system’s database.

**ELEMENTS OF ER-MODEL:**

**Symbols Used in ER-Diagram**

|  |  |
| --- | --- |
| **Symbol** | **Meaning** |
|  | **Entity** |
|  | **Relationship** |
|  | **Attribute** |
|  | **Key Attribute** |
|  | **Cardinality**  **ratio 1:n for e1:e2**  **in r** |
|  | **cardinality**  **ratio m:n for e1:e2**  **in r** |

**ER-DIGRAMER**

admin

verfition



college admin

student detail

student

college\_name

college\_code

college\_addr



event notice

eid

edate

ename

sname

sreg

year



event result

has



election

annonce



candidate

eid

year

post

cid

pid

cidproof

has



result

votes

view

image

comment

result display

**4.6.2 SYSTEM TECHNICAL ARCHITECTURE.**

IIS SERVER

SQL SERVER

Sql Server

ADMIN

STUDENT

COLLEGE ADMIN

**Add information register voting, participate for fest**

**college**

DATABASE

**view**

**4.6 Detail design**

**4.6.1 FLOW CHART**

Flow chart is a graphic picture of the logical steps and sequence involved in a procedure or a program.

Some of the symbol used in drawing a flowchart-

|  |  |
| --- | --- |
| **Terminator** | **Terminator or oval symbol is used to define the beginning of a flow chart and to show the termination point of flow chart.** |
| **Flow Lines** | **Flow lines connect the other symbol in flow chart. The arrows with the sequence of steps to follow.** |
| **Input/output Box** | **A parallelogram is used represent the Input and Output operation.** |
| **Decisions Box** | **The diamond symbol is used when want the program to decide between alternate courses of action.** |
| **Process Box** | **A rectangle or process symbol is generally used to represent any processing operation.** |

**4.6.1.1. Login**

username and password

error message

validation

**False**

**True**

Logged in successfully

**4.6.1.2. Event detail:**

Event name, event detail,

error message

validation

**False**

**True**

Event result.

**4.6.1.3. Student Registration for fest**

Student name gender, contact etc

error message

validation

False

True

studentregistered successfully

**4.6.1.4. Candidate nominee for election**

Candidate name , contact etc

error message

validation

False

True

Candidate register is successfully

**4.6.1.5 Change password**

new password, confirm password, Login ID

invalid

verify

**False**

**True**

Password will be changed

**4.6.1.6.Update Profile:**

college/ student name, contact number etc.

invalid

verify

False

True

College /student Profile will be updated.

**4.6.2 Pseudo coding(algorithm)**

**4.6.1 LOGIN**

Input username, password,

If user name exits, then

Check for whether password matches

If password matches, then

Insert it to data base then login

Else

display error message

End if

End if

**4.6.2 COLLEGE MODULE**

Input college name, address contact no, email id

If its exits, then

Insert into database

Else display error message

End if

**4.6.3 STUDENT MODULE**

Input student name, address contact no, email id

If its exits, then

Insert into database

Else display error message

End if

**4.6.4 EVENT MODULE**

Input Eventname, event date event detail, email id

If its exits, then

Insert into database

Else display error message

End if

**4.6.2 EVENT SCHEDULE MODULE**

Input Event location, event description, event location

If its exits, then

Insert into database

Else display error message

End if

**Chapter-5**

**Coding**

**5.1 Introduction**

The goal of coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design. The coding phase is not to simplify the job of the programmer. Rather, the goal should be simplifying the job of the tester and maintainer.

There are many different criteria for judging a program, including readability, size of program, execution

time and required memory. Having readability and understanding ass a clear objective of the coding activity can itself help in producing software that is more maintainability.

The coding is done with the following characteristics in mind:

* Ease of design to code translation
* Code efficiency
* Memory efficiency
* Response time
* Maintainability
* Security
* Simple ease to understand code
* Efficient and consistent **logic**

Programming style

It is impossible to provide an exhaustive list of what to do and not to do produce simple readable

Code: Next we will some general rules that usually apply.

**Name**

Selecting module and variable names is often not considered important by invoice programmers. Most variable in a program reflect some entity in the program domain, and the module reflect some process. Variable names should be closely related the entity they represent and module name should be reflected their activity.

**Control constructs**

It is desirable to use a few standard control constructs rather than using a wide variety of constructs, just because their available in the language.

**USER DEFINED TYPES**

Modern languages allow user to defined types like the enumerated type. When such facilities are available, they should be exploited where applicable.

**Module size**

A programmer should examine any routine with very few statements or too many statements. Large modules often will not be functionally cohesive, and too small modules might incur unnecessary overhead.

**Module interface**

A module with a complex interface should be carefully examined. Such modules might not be functionally cohesive and might be implementing multiple functions.

**Robustness**

A program is robust if it does something planned for exceptional conditional. A program might encounter exceptional condition in such from as incorrect input, the correct value of some variable and overflow. A program should try to handle such situation. In general, a program should check for validation, where possible and should check for possible, and should check for possible overflow of data structures.

**5.2 Sample code**

**College register code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Configuration;

using System.Data;

using System.IO;

namespace intercollege

{

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

{

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

private static Random random = new Random();

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

{

}

public static string RandomString(int length)

{

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

return new string(Enumerable.Repeat(chars, length)

.Select(s => s[random.Next(s.Length)]).ToArray());

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

string filename = string.Empty;

string imagename = string.Empty;

string imgFileType = string.Empty;

if (FileUploadimage.HasFile)

{

imagename = DateTime.Now.ToString("MM-dd-yyyy\_HHmmss");

imgFileType = Path.GetExtension(FileUploadimage.FileName).ToString().ToLower();

FileUploadimage.SaveAs(Server.MapPath("~/uploads/colimg/" + imagename + imgFileType));

}

string pass = RandomString(8);

cmd.Parameters.AddWithValue("@clgid", Convert.ToInt32(ddlcollege.SelectedValue.ToString()));

cmd.Parameters.AddWithValue("@addr", txtadd.Text);

cmd.Parameters.AddWithValue("@contact", txtcon.Text);

cmd.Parameters.AddWithValue("@alt", txtalt.Text);

cmd.Parameters.AddWithValue("@email", txtemail.Text);

cmd.Parameters.AddWithValue("@long", txtlon.Text);

cmd.Parameters.AddWithValue("@lat", txtlan.Text);

cmd.Parameters.AddWithValue("@colgcode", txtcolgcode.Text);

cmd.Parameters.AddWithValue("@imagename", "uploads/colimg/" + imagename + imgFileType);

cmd.Parameters.AddWithValue("@status", "Inactive");

cmd.Parameters.AddWithValue("@pass", pass);

cmd.CommandText = "insert into tblcolg\_detail(colg\_id,address,contact,contact2,email\_id,photopath,lattitude,longitude,colg\_pass,status,colg\_code) "

+ "values(@clgid,@addr,@contact,@alt,@email,@imagename,@lat,@long,@pass,@status,@colgcode)";

int i = cmd.ExecuteNonQuery();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Registration success!! Confirmation mail will be sent once univerisity verify the Account');", true);

clearform();

}

}

public void clearform()

{

txtadd.Text = "";

txtcon.Text = "";

txtalt.Text = "";

txtemail.Text = "";

txtlon.Text = "";

txtlan.Text = "";

}

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

{

clearform();

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.CommandText = "Select \* from tblcolg\_detail where email\_id='"+txtemail.Text+"'";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Email Id already Registered');", true);

txtemail.Text = "";

txtemail.Focus();

}

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.CommandText = "Select \* from tblcolg\_detail where colg\_code='" +txtcolgcode.Text + "'";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('code error');", true);

txtcolgcode.Text = "";

txtcolgcode.Focus();

}

}

**Student code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Configuration;

using System.Data;

namespace intercollege

{

public partial class studentreg1 : System.Web.UI.Page

{

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

int studid = 0;

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

{

if (Request.QueryString["stid"] != null)

{

studid = Convert.ToInt32(Request.QueryString["stid"].ToString());

if (!IsPostBack)

{

fillstudentdata(studid);

}

}

}

public void fillstudentdata(int studid)

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@studid", studid);

cmd.CommandText = "Select \* from tbl\_student where studid=@studid";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

txtemail.Text = dt.Rows[0]["email\_id"].ToString();

txtstuname.Text = dt.Rows[0]["studname"].ToString();

textcon.Text = dt.Rows[0]["contactno"].ToString();

txtreg.Text = dt.Rows[0]["regno"].ToString();

lstcousre.SelectedValue = dt.Rows[0]["courseid"].ToString();

lstyear.SelectedValue = dt.Rows[0]["admission\_year"].ToString();

if (dt.Rows[0]["gender"].ToString().Trim() == "Male")

{

radmale.Checked = true;

radfemale.Checked = false;

}

else

{

radfemale.Checked = true;

radmale.Checked = false;

}

btnsubmit.Text = "Update";

}

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

string gender = string.Empty;

if (radmale.Checked)

gender = "male";

else if (radfemale.Checked)

gender = "female";

cmd.Parameters.AddWithValue("@studentname", txtstuname.Text);

cmd.Parameters.AddWithValue("@regno", txtreg.Text);

cmd.Parameters.AddWithValue("@gender", gender);

cmd.Parameters.AddWithValue("@year", lstyear.SelectedValue.ToString());

cmd.Parameters.AddWithValue("@contact", textcon.Text);

cmd.Parameters.AddWithValue("@email", txtemail.Text);

cmd.Parameters.AddWithValue("@course",Convert.ToInt32(lstcousre.SelectedValue.ToString()));

cmd.Parameters.AddWithValue("@clgid", Convert.ToInt32(Session["id"].ToString()));

if (btnsubmit.Text == "Update")

{

cmd.Parameters.AddWithValue("@studid", studid);

cmd.CommandText = "update tbl\_student set studname=@studentname,regno=@regno,gender=@gender,admission\_year=@year,courseid=@course,contactno=@contact,email\_id=@email where studid=@studid";

int i = cmd.ExecuteNonQuery();

con.Close();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('success!! Record Updated.');window.location='view\_stud.aspx'", true);

}

}

else

{

cmd.CommandText = "insert into tbl\_student(colgid,studname,regno,gender,admission\_year,courseid,contactno,email\_id)" +

"values(@clgid,@studentname,@regno,@gender,@year,@course,@contact,@email)";

int i = cmd.ExecuteNonQuery();

con.Close();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('success!! Record Added.');window.location='view\_stud.aspx'", true);

}

}

}

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

{

txtemail.Text = "";

txtreg.Text = "";

txtstuname.Text = "";

textcon.Text = "";

lstcousre.SelectedIndex = 0;

lstyear.SelectedIndex = 0;

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@regno", txtreg.Text);

cmd.CommandText = "Select \* from tbl\_student where regno=@regno";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Registration No. Already Exist');", true);

txtreg.Text = "";

txtreg.Focus();

}

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.CommandText = "Select \* from tbl\_student where email\_id='" + txtemail.Text + "'";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Email Id already Registered');", true);

txtemail.Text = "";

txtemail.Focus();

}

}

}

}

**Event detail code**

sing System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Configuration;

using System.Data;

using System.Globalization;

namespace intercollege

{

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

{

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

int eventid = 0;

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

{

if (string.IsNullOrEmpty(Session["type"] as string))

{

Response.Redirect("index.aspx");

}

if (!string.IsNullOrEmpty(Request.QueryString["event"] as string))

{

eventid = Convert.ToInt32(Request.QueryString["event"].ToString());

if (!IsPostBack)

{

filleventdata(eventid);

}

}

}

public void filleventdata(int eventid)

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@eventid", eventid);

cmd.CommandText = "Select \* from tblevent\_notice where event\_id=@eventid";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

txtevent.Text = dt.Rows[0]["event\_name"].ToString();

txtdes.Text = dt.Rows[0]["event\_desc"].ToString();

txtevedate.Text = Convert.ToDateTime(dt.Rows[0]["event\_date"]).ToShortDateString();

ddlcolg.SelectedValue = dt.Rows[0]["colgid"].ToString();

if (dt.Rows[0]["event\_type"].ToString().Trim() == "Sports")

{

rbsports.Checked = true;

rbculture.Checked = false;

rbothers.Checked = false;

}

else if (dt.Rows[0]["event\_type"].ToString().Trim() == "Culture")

{

rbsports.Checked = false;

rbculture.Checked = true;

rbothers.Checked = false;

}

else

{

rbsports.Checked = false;

rbculture.Checked = false;

rbothers.Checked = true;

}

btnsubmit.Text = "Update";

}

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

string event\_type = string.Empty;

if (rbculture.Checked)

event\_type = "Culture";

else if (rbsports.Checked)

event\_type = "Sports";

else

event\_type = "Other";

string filename = string.Empty;

if (up\_file.HasFile)

{

filename = up\_file.FileName;

up\_file.SaveAs(Server.MapPath("~/uploads/" + filename));

}

else

{

if (btnsubmit.Text == "Update")

{

filename = filecopy.HRef;

}

}

cmd.Parameters.AddWithValue("@evenname", txtevent.Text);

cmd.Parameters.AddWithValue("@descrition", txtdes.Text);

cmd.Parameters.AddWithValue("@eventdate", DateTime.ParseExact(txtevedate.Text,"dd/MM/yyyy",CultureInfo.InvariantCulture));

cmd.Parameters.AddWithValue("@eventupload", "uploads/" + filename);

cmd.Parameters.AddWithValue("@colname", Convert.ToInt32(ddlcolg.SelectedValue.ToString()));

cmd.Parameters.AddWithValue("@event\_type", event\_type);

cmd.Parameters.AddWithValue("@status", "Active");

if (btnsubmit.Text == "Update")

{

cmd.Parameters.AddWithValue("@event", eventid);

cmd.CommandText = "update tblevent\_notice set event\_name=@evenname,event\_desc=@descrition,event\_date=@eventdate,event\_file=@eventupload,colgid=@colname,event\_type=@event\_type where event\_id=@event";

int i = cmd.ExecuteNonQuery();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('success!! Record Updated.');window.location='ManageEvents.aspx';", true);

}

}

else

{

cmd.CommandText = "insert into tblevent\_notice(event\_name,event\_desc,event\_date,event\_file,colgid,status,event\_type) values(@evenname,@descrition,@eventdate,@eventupload,@colname,@status,@event\_type)";

int i = cmd.ExecuteNonQuery();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('success!! Record Added.')", true);

clearform();

}

}

}

public void clearform()

{

txtdes.Text = "";

txtevedate.Text = "";

txtevent.Text = "";

ddlcolg.SelectedIndex = 0;

}

}

}

**Election detail code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Globalization;

using System.Data.SqlClient;

using System.Configuration;

namespace intercollege

{

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

{

DataTable dt = new DataTable();

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

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

{

if (!IsPostBack)

{

dt.Columns.Add("post", typeof(string));

dt.Columns.Add("descr", typeof(string));

Session["dtobj"] = dt;

}

}

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

{

string post = txtpost.Text;

string descr = txtdescr.Text;

dt = (DataTable)Session["dtobj"];

dt.Rows.Add(post, descr);

Session["dtobj"] = dt;

gvepost.DataSource = dt;

gvepost.DataBind();

btnnext.Enabled = true;

txtpost.Text = "";

txtdescr.Text = "";

}

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

{

dt = (DataTable)Session["dtobj"];

DateTime ele\_date = DateTime.ParseExact(txtele.Text, "yyyy/MM/dd", CultureInfo.InvariantCulture);

DateTime res\_date = DateTime.ParseExact(txtresl.Text, "yyyy/MM/dd", CultureInfo.InvariantCulture);

string title = txttitle.Text;

string desc = txtdescr.Text;

string ayear = lstyear.SelectedValue.ToString();

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@ele\_date", ele\_date);

cmd.Parameters.AddWithValue("@res\_date", res\_date);

cmd.Parameters.AddWithValue("@title", title);

cmd.Parameters.AddWithValue("@desc", desc);

cmd.Parameters.AddWithValue("@ayear", ayear);

cmd.Parameters.AddWithValue("@status", "Active");

cmd.CommandText = "insert into tbl\_election(epost,edisc,e\_date,result\_date,status,eyear) output inserted.eid values(@title,@desc,@ele\_date,@res\_date,@status,@ayear)";

int eid = Convert.ToInt32(cmd.ExecuteScalar().ToString());

if (eid > 0)

{

for (int i = 0; i < dt.Rows.Count; i++)

{

SqlCommand cmd1 = new SqlCommand();

cmd1.Connection = con;

cmd1.Parameters.AddWithValue("@post", dt.Rows[i][0].ToString());

cmd1.Parameters.AddWithValue("@descr", dt.Rows[i][1].ToString());

cmd1.Parameters.AddWithValue("@eid", eid);

cmd1.Parameters.AddWithValue("@estatus", "Active");

cmd1.CommandText = "insert into tblelection\_detail(eid,post,post\_desc,status) values(@eid,@post,@descr,@estatus)";

int j = cmd1.ExecuteNonQuery();

}

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('success!! Record Added.')", true);

clearform();

}

}

public void clearform()

{

txtdescr.Text = "";

txttitle.Text = "";

txtpost.Text = "";

lstyear.SelectedIndex = 0;

txtresl.Text = "";

txtele.Text = "";

txtdes.Text = "";

dt = new DataTable();

gvepost.DataSource = dt;

}

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

{

clearform();

}

protected void gvepost\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName == "del")

{

dt = (DataTable)Session["dtobj"];

for (int i = dt.Rows.Count - 1; i >= 0; i--)

{

DataRow dr = dt.Rows[i];

if (dr["post"].ToString() == e.CommandArgument.ToString())

dr.Delete();

}

Session["dtobj"] = dt;

gvepost.DataSource = dt;

gvepost.DataBind();

}

}

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

{

try

{

DateTime electiondate = DateTime.ParseExact(txtele.Text, "yyyy/MM/dd", CultureInfo.InstalledUICulture);

DateTime result = DateTime.ParseExact(txtresl.Text, "yyyy/MM/dd", CultureInfo.InstalledUICulture);

if (DateTime.Compare(result, electiondate) < 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Result date should be later than the Election Date')", true);

txtresl.Text = "";

}

else if (DateTime.Compare(result, electiondate) == 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Result date should be later than the Election Date')", true);

txtresl.Text = "";

}

}

catch (Exception ex)

{

}

}

}

}

**Election participate code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Data;

using System.Configuration;

using System.IO;

namespace intercollege

{

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

{

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

int candidateid = 0;

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

{

if (Session["type"] == null)

{

Response.Redirect("index.aspx");

}

if (!IsPostBack)

{

ddlpost.Items.Add(new ListItem("--Select Post--", "0"));

}

if (!string.IsNullOrEmpty(Request.QueryString["id"] as string))

{

candidateid = Convert.ToInt32(Request.QueryString["id"].ToString());

if (!IsPostBack)

{

Fillcandidate(candidateid);

}

}

}

public void Fillcandidate(int cid)

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@cid", cid);

cmd.CommandText = "SELECT tbl\_candidate.candidateid, tbl\_candidate.studid, tbl\_candidate.candidate\_idcrad, tbl\_candidate.postid, tbl\_election.eid, tbl\_candidate.colgid FROM tbl\_candidate INNER JOIN " +

" tblelection\_detail ON tbl\_candidate.postid = tblelection\_detail.postid INNER JOIN tbl\_election ON tblelection\_detail.eid = tbl\_election.eid where candidateid = @cid";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

ddlcandidate.SelectedValue = dt.Rows[0]["studid"].ToString();

ddlelection.SelectedValue = dt.Rows[0]["eid"].ToString();

fillpost(Convert.ToInt32(dt.Rows[0]["eid"].ToString()));

ddlpost.SelectedValue = dt.Rows[0]["postid"].ToString();

HiddenField1.Value = dt.Rows[0]["candidate\_idcrad"].ToString();

btnsubmit.Text = "Update";

}

}

public void fillpost(int eid)

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@eid", eid);

cmd.CommandText = "select \* from tblelection\_detail where eid = @eid";

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

con.Close();

ddlpost.Items.Clear();

ddlpost.DataSource = dt;

ddlpost.DataTextField = "post";

ddlpost.DataValueField = "postid";

ddlpost.DataBind();

ddlpost.Items.Insert(0, new ListItem("-- Select Post --", ""));

}

protected void ddlelection\_SelectedIndexChanged(object sender, EventArgs e)

{

int eid = Convert.ToInt32(ddlelection.SelectedValue.ToString());

fillpost(eid);

}

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

{

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

string filename = string.Empty;

string strFileType = "";

string idproof = string.Empty;

if (FileUploadidproof.HasFile)

{

filename = DateTime.Now.ToString("MM-dd-yyyy\_HHmmss");

strFileType = Path.GetExtension(FileUploadidproof.FileName).ToString().ToLower();

FileUploadidproof.SaveAs(Server.MapPath("~/uploads/doc/" + filename + strFileType));

idproof = "uploads/doc/" + filename + strFileType;

}

else

{

idproof = HiddenField1.Value;

}

cmd.Parameters.AddWithValue("@eid", Convert.ToInt32(ddlelection.SelectedValue.ToString()));

cmd.Parameters.AddWithValue("@postid", Convert.ToInt32(ddlpost.SelectedValue.ToString()));

cmd.Parameters.AddWithValue("@colgid", Convert.ToInt32(Session["id"].ToString()));

cmd.Parameters.AddWithValue("@cdname", Convert.ToInt32(ddlcandidate.SelectedValue.ToString()));

cmd.Parameters.AddWithValue("@idproof", idproof);

if (btnsubmit.Text == "Update")

{

cmd.Parameters.AddWithValue("@cid", candidateid);

cmd.CommandText = "update tbl\_candidate set candidate\_idcrad=@idproof,colgid=@colgid,postid=@postid,studid=@cdname where candidateid=@cid";

}

else

{

cmd.CommandText = "insert into tbl\_candidate(candidate\_idcrad,colgid,postid,studid)" + "values(@idproof,@colgid,@postid,@cdname)";

}

int i = cmd.ExecuteNonQuery();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('success!!');", true);

clearform();

gvcandidate.DataBind();

}

}

public void clearform()

{

ddlelection.SelectedIndex = 0;

ddlpost.Items.Clear();

//ddlpost.SelectedIndex = 0;

}

protected void gvcandidate\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName == "edit")

{

int candidate = Convert.ToInt32(e.CommandArgument.ToString());

Response.Redirect("postcandidate.aspx?id=" + candidate);

}

else if (e.CommandName == "del")

{

int candidate = Convert.ToInt32(e.CommandArgument.ToString());

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@candidate", candidate);

cmd.CommandText = "delete from dbo.tbl\_candidate where candidateid=@candidate";

int i = cmd.ExecuteNonQuery();

con.Close();

if (i > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Record Deleted...');", true);

clearform();

gvcandidate.DataBind();

}

}

}

protected void ddlpost\_SelectedIndexChanged(object sender, EventArgs e)

{

int postid = Convert.ToInt32(ddlpost.SelectedValue.ToString());

int collgid = Convert.ToInt32(Session["id"].ToString());

if (con.State != ConnectionState.Open)

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.Parameters.AddWithValue("@postid", postid);

cmd.Parameters.AddWithValue("@collgid", collgid);

cmd.CommandText = "select \* from tbl\_candidate where colgid=@collgid and postid=@postid";

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

con.Close();

if (dt.Rows.Count > 0)

{

ScriptManager.RegisterClientScriptBlock(this, GetType(), "info", "alert('Candidate Already selected for this post...');window.location='postcandidate.aspx';", true);

}

}

}

}

**Chapter-6**

**Testing**

**6.1 Introduction:**

Testing is a process used to help identify the correctness, completeness and quality of developed computer software. With that in mind establish the correctness of computer software.

**6.2 Testing objectives include:**

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is the one that uncovers as yet undiscovered error.
3. A successful test is the one that uncovers as yet undiscovered errors.

Testing should systematically uncover different classes of errors in a minimum amount time and with a minimum amount of effort. A secondary benefit testing is that it demonstrates that the software appears to be working as stated in the specification.

Test methodology:

Testing can be done in any one of the following levels:

* Unit testing test the minimal software component or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented.
* Integration testing exposes defects in the interfaces and interfaces and interaction between the integrated components (modules).
* System testing tests a completely integrated system to verify that in meets its requirements.
* System integration testing verifies that a system is integrated to any external or third party system defined in the system requirements.

**Test Cases for Login page:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **TEST CONDITION** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | When the admin/user clicks on “login” button with a valid username and password. | Logs to the home page | Successful |
| 2 | When the admin/user clicks on “login” button with an invalid username and password. | Msgbox appears with a message “invalid username/password” | Successful |
| 3 | When the admin/User clicks on the login button without entering any of the fields | Msgbox appears with a message “Enter username/password” | Successful |

**Test Cases for Registration Page:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **TEST CONDITION** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | When the user clicks on “register” button with a valid details. | Logs to the home page | Successful |
| 2 | When the user clicks on “register ” button with invalid details. | Asks to enter the fields correctly,doesn’t redirect to next page | Successful |
| 3 | When the user clicks on “cancel” button after entering the field details. | Fields details entered are cleared | Successful |

**Test Cases for Student Registration Page:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **TEST CONDITION** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | When the student clicks on “register” button with a valid details. | Logs to the home page | Successful |
| 2 | When the student clicks on “register ” button with invalid details. | Asks to enter the fields correctly,doesn’t redirect to next page | Successful |
| 3 | When the student clicks on “cancel” button after entering the field details. | Fields details entered are cleared | Successful |

**Test Cases for Forgot password:**

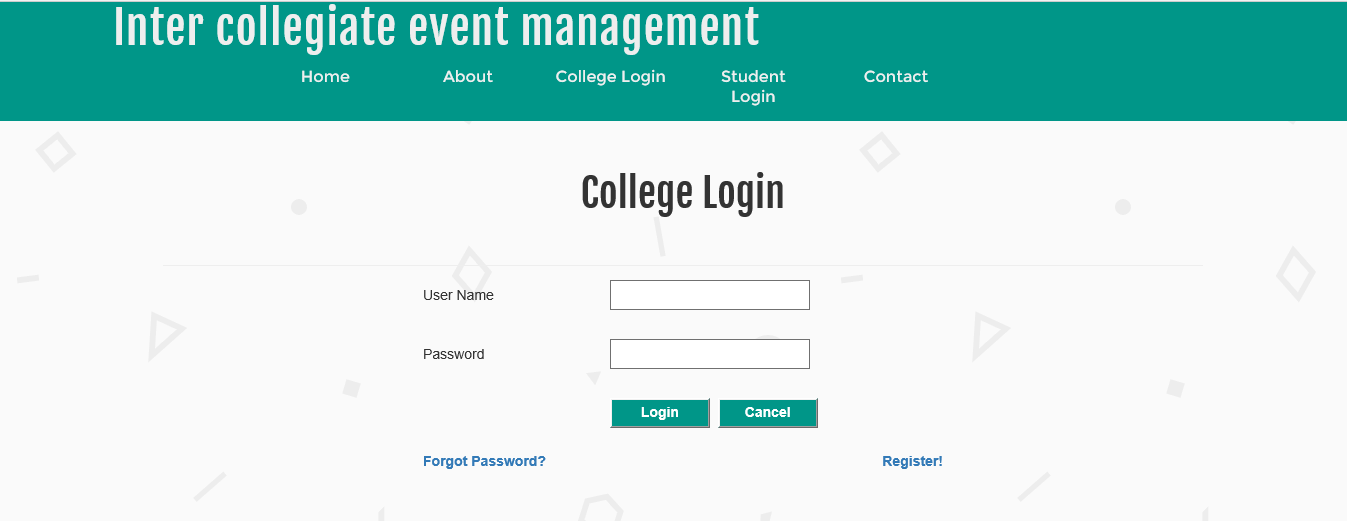
|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **TEST CONDITION** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | When the user enters the valid email id and enters submit button | Msgbox appears with a message “success your new password is sent to your mail” | Successful |
| 2 | When the user enters the invalid /unregisted emailed and enters submit button | Msgbox appears with a message”Emailid not registered” | Successful |
| 3 | When user doesn’t enter the email id and clicks sumit button | Msgbox appears with the message “Enter email id” | Successful |

**Test cases for Add Event:**

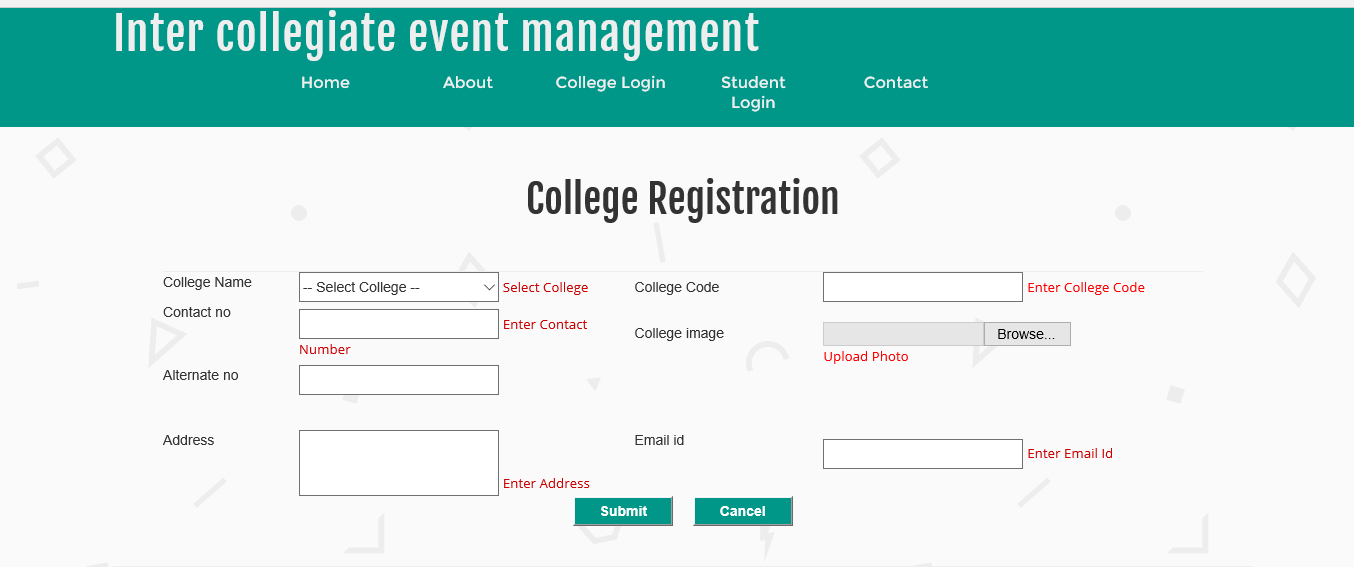
|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO** | **TEST CONDITION** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | When admin enters event and clicks on add button | Event is added | **Successful** |
| 2 | When admin doesn’t enter anything and clicks on add button | Msgbox appers with a message “Please enter the event detail” | Successful |
| 3 | When admin enters previously added event | Msgbox appears with a message “event detail already exists”. | Successful |
| 4 | When admin click delete button placed next event | Event detail is deleted | Successful |

**Chapter-7**

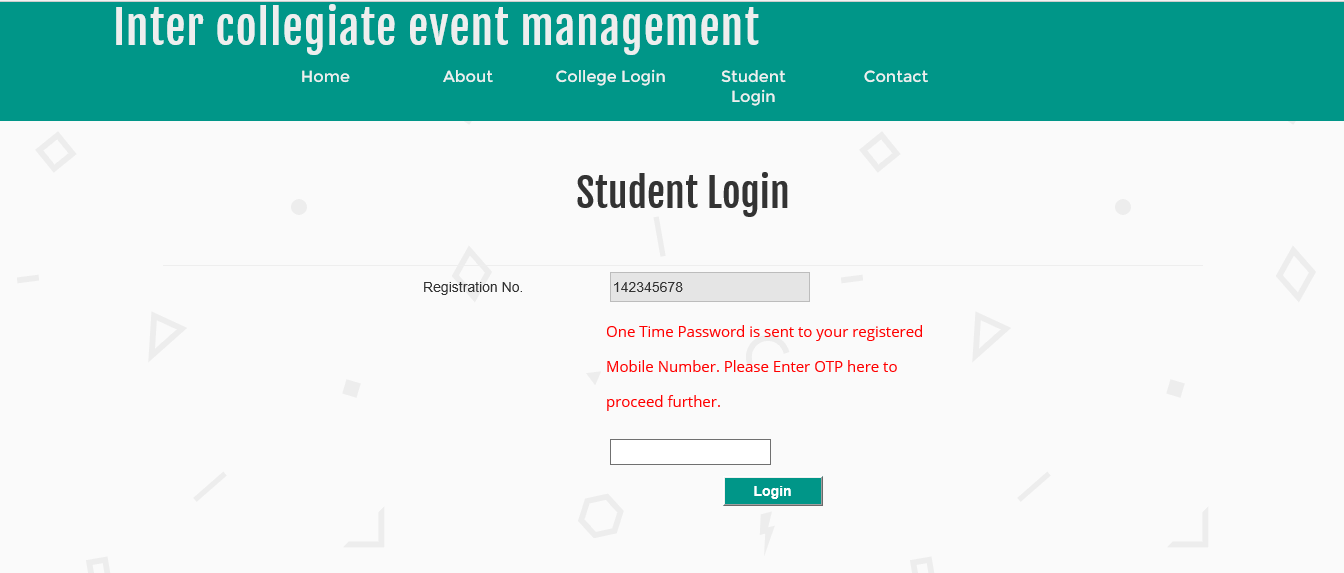
**Screen shots**

****

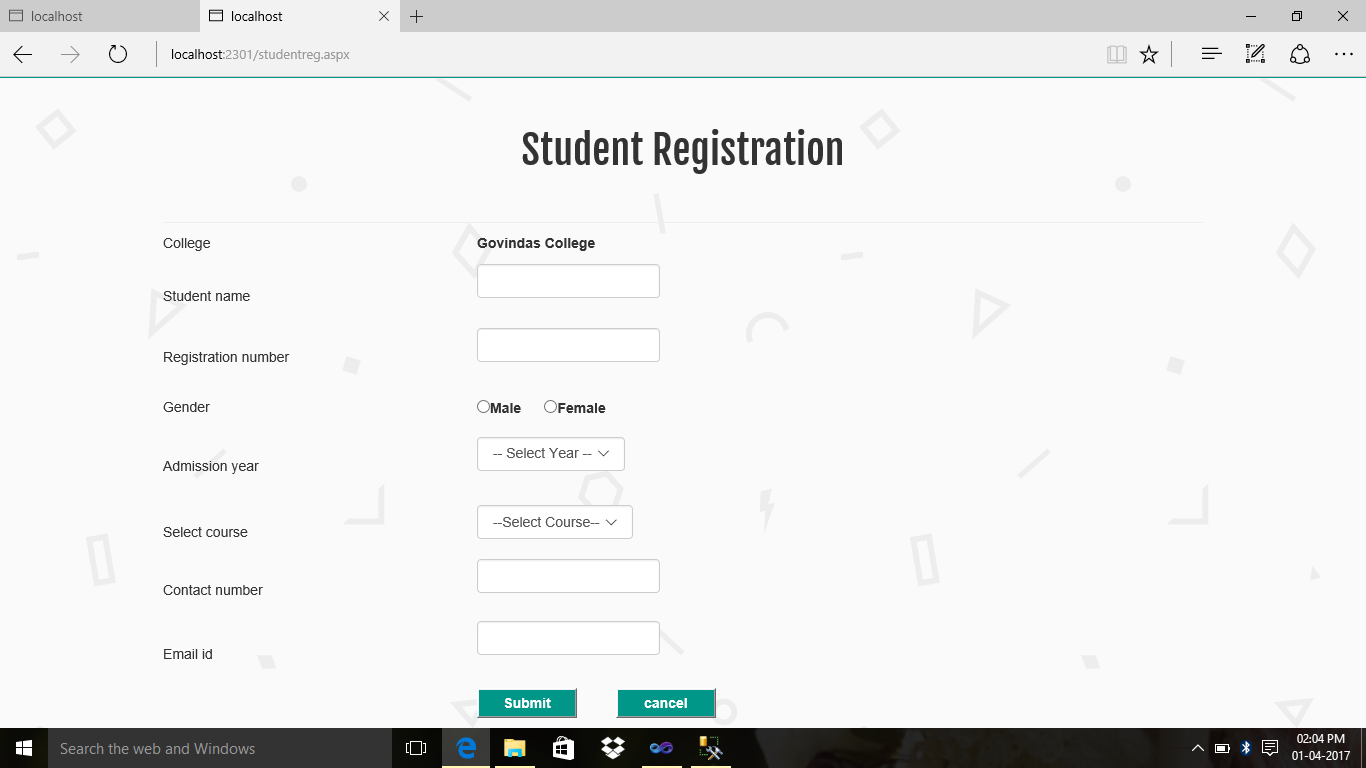
This page will be used for college login

****

This page will have used to college register

****

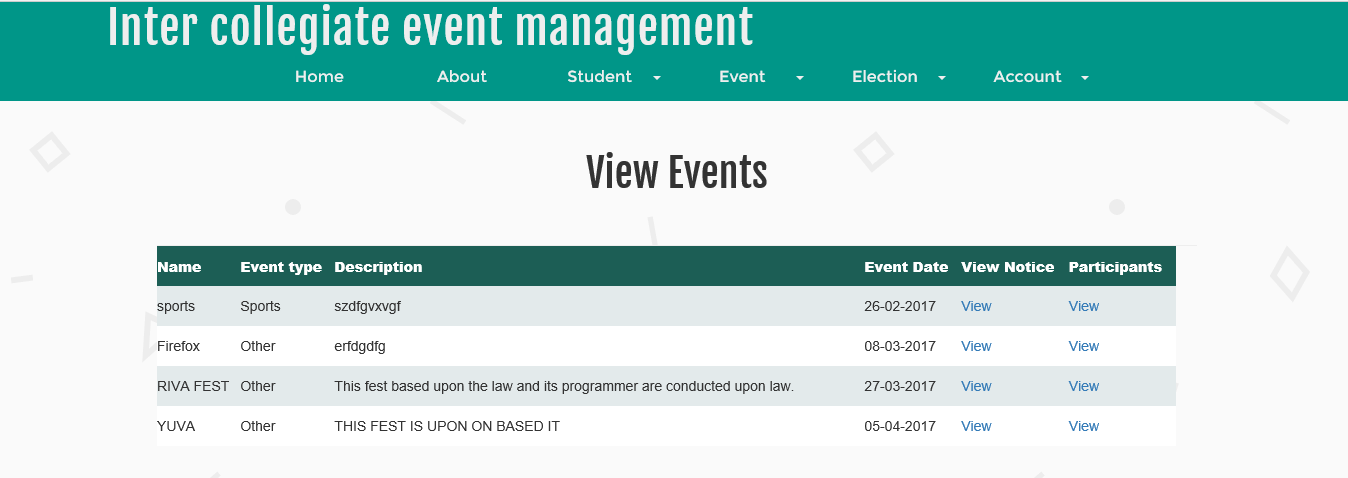
This page will have used to student login

****

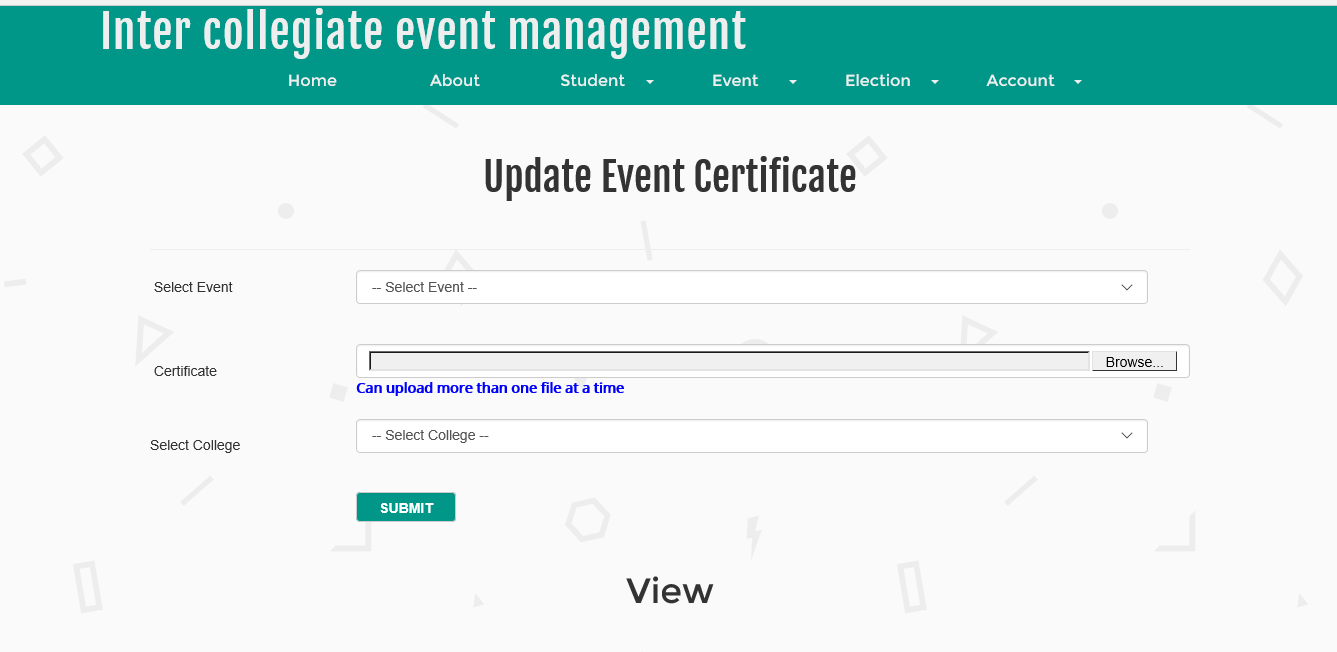
**This page will have used to student registration**

****

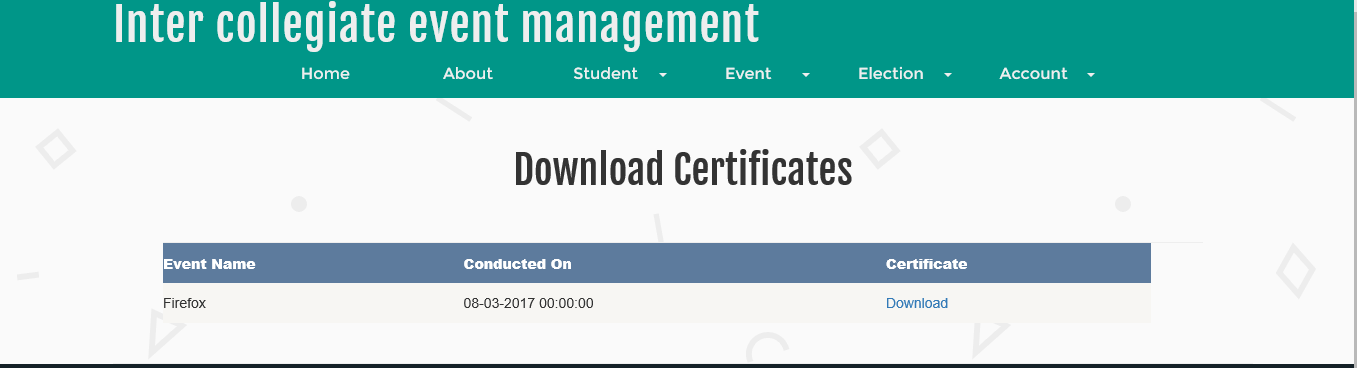
This page used to view the election detail

****

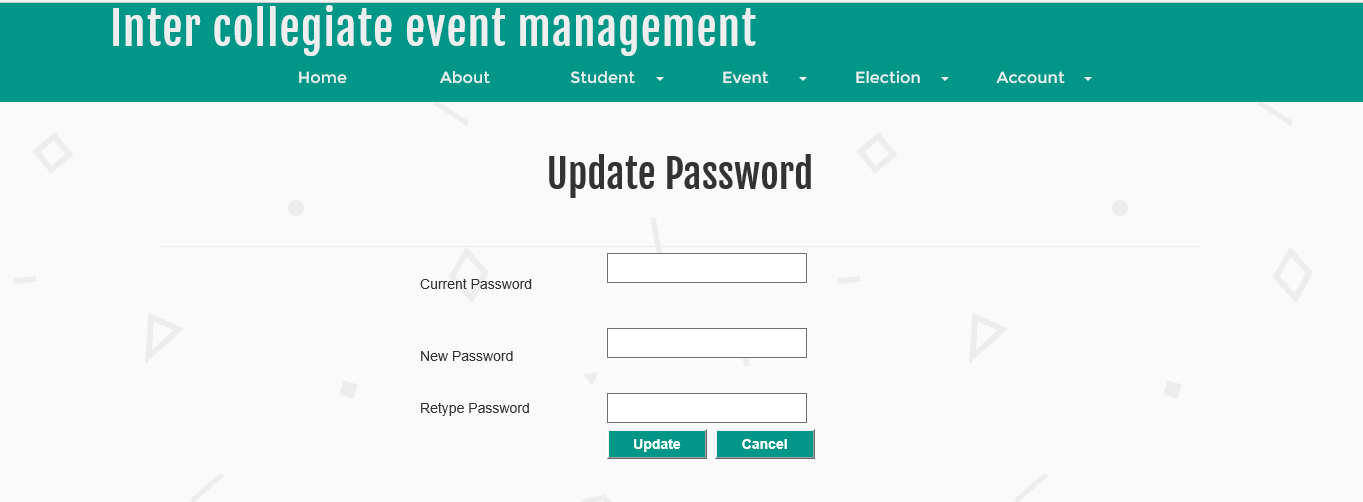
To view the event details

****

To update the event certificate

****

To download certificate



To update password

**CHAPTER -8**

**USER MANUAL**

**8.1. INTRODUCTION:**

The main admin is university admin.The university Admin, college admin and student are the three main access the program and views and operating unlimited access to all functionality of the software like registering, approving, registering the event, inviting the various college for the various event thought this website view and result are also displayed thought this website and also can comment and share. The election are also took place thought this website .The student can vote their candidate through the help of this website.

**8.2. DOCUMENT OVERVIEW:**

The document provides detailed information about the project “INTER COLLEGIATE EVENT MANAGEMENT” begin with contents synopsis, SRS, system design, detailed design, coding, testing, screen shots, user manual, conclusion and bibliography details.

**8.3. APPLICATION DOCUMENT:**

The application document contains the information needed for the using mechanism of this application

* The admin use to login first and the college use to get register.
* Then the university admin use to verify the college and then that college get approve.
* Then the various college admin input all the student detail of their college.
* After getting register the college can update their event detail.
* The university election is also take place the university use to update the election detail and the candidate are update by the college admin.
* The student can take part in event and for voting prepare through opt code.
* And the both the result (event, election) are also displayed in this website.

And student can update the images and can comment upon it.

**8.4. FEATURES**

* The system allows three types of users to use the system namely
* University Admin
* College admin
* Student
* The university admin is the main admin which control upon the various events and detail.
* College register themselves and approved by the university admin.
* Approved college can be details can be viewed.
* College add there college student and there various event which is going to conduct.
* The college can also display their result and also update their images of the event which took place and the student can comment.
* The election also can take place through this website and the candidate from various colleges can be added for the election.
* The results are update through this website.

**CHAPTER-9**

**CONCLUSION AND FUTURE ENCHANCEMENT**

**9 Conclusion and future enhancement**

This website is used for arranging online event for various colleges which come under Mysore University and Mysore University. The event is scheduled by the Mysore University and various colleges which college is taking part in the event are taking place. After the competition is held, the result will be announced. The current student secretary information will be given by the respective colleges. The Mysore University will post the information detail for elections. After every college will nominate a candidate for the year of overall college representative post of the Mysore University. Each college secretaries will vote for thus website.

**Project purpose**

* It takes place through the website.
* No need of going to each college and spend the money and time we can save our time and money thought this concept.
* The college can invite through the help of this website.
* Through this website the student can knew the detail about event and the election event also.
* The student can elect their candidate through this website.
* And can also participant in various events through this website and also can view result of various event and election.
* And student can comment and share the image.

**9.1Conclusion:**

The project work titled inter college event calendar has been designed using asp.net server where in many users friendly from control and has been added in order to make it interactive application. The system in developed in such a way that the user with common knowledge of computer can handle it easily. The entire document required for operation and admittance of the website had been provided. The system forms a general platform for building most advance shopping system.

The project helps us the knowledge in asp.net and SQL server. The importance of a good software design was learned during the project.

**CHAPTER-10**

**BIBLIOGRAPHY**

* **Books Referred:**

Pankaj Jalote

An Integrated Approach to Software Engineering, 3rd Edition.

* **Deekshith Rajkumar**

Introduction to Systems Analysis and Design , 4th Edition

* Web References:

[www.google.com](http://www.google.com)

[www.codeproject.com](http://www.codeproject.com)

[www.stackoverflow.com](http://www.stackoverflow.com)