

RESULT BUILDER

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF SCIENCE (COMPUTER SCIENCE)

By

Jagannath B. Patta

Roll Number - 438

Under the esteemed guidance of

Prof. Randeep Singh Ghai

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE

GURU NANAK KHALSA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

(Autonomous)

MATUNGA, MUMBAI – 400 019

MAHARASHTRA

AY 2019 – 2020

APPROVAL OF PROJECT PROPOSAL

PNR No: 2017016402268685

Roll No: 438

1. Name of the Student

Jagannath B. Patta

2. Title of the Project

Result Builder

3. Name of the Project Guide

Prof. Randeep Singh Ghai

4. Teaching experience of the Project Guide - 4 Years

5. Is this your first submission? Yes

Signature of the Student

Date: _____

Signature of the Guide

Date: _____

Signature of the Head of Dept.

Date: _____

Signature of the Examiner

Date: _____

GURU NANAK KHALSA COLLEGE
OF
ARTS, SCIENCE & COMMERCE
(Autonomous)
MATUNGA, MUMBAI, MAHARASHTRA – 400 019

DEPARTMENT OF COMPUTER SCIENCE



CERTIFICATE

This is to certify that the project entitled, " **Result Builder** ", is bonafied work of **JAGANNATH B. PATTA** bearing Seat No: **438** submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE (COMPUTER SCIENCE) from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

Abstract / Synopsis

This project is a website build for generating the results of students, it includes storing data of each and every student according to their stream, department, subjects and teacher. It also allows the feature of generating report card on the basis of the score the teacher will enter. It provides the soft copy of the result properly formatted according to the college report card layout and can be downloaded. The report card generating process is only accessible by the authorised faculty members assigned by the admin of the database.

One of the major focus is on the calculation and converting marks into grades and stored. Teachers only have to enter less input credentials and in return will get lots of extra.

This website comprises of 2 major modules with their sub-modules:

Admin (HOD):

- **Admin Login:**
 - Admin can login to the website using his ID and Password.
- **Assigning Teacher:**
 - Admin can add new teacher details (username / password) into the database.
 -

Teacher:

- **Teacher Login:**
 - Teacher will access the website using his/ her User Name and Password that is stored in the database.
- **Maintain Database:**
 - Adding, Updating, Deleting student details.

ACKNOWLEDGEMENT

It's a great pleasure for me to develop a Web Application that is RESULT BUILDER. I have gathered knowledge and experience during this project. Apart from my efforts, the success of any project depends largely on the encouragement and guidelines of many others. In this endeavour I would like to show my greatest appreciation to our internal project guide Prof. RANDEEP SINGH GHAI of Department of IT/Computer Science, & Mr. ALDRIN JOLLY college Administrator, Guru Nanak Khalsa College of Arts, Science & Commerce, who was abundantly helpful and offered invaluable assistance, support and guidance. His frank suggestions for improvement and innovative ideas have inspired us. Finally, I must acknowledge with due respect the constant support and patience of my Family and my friends who supported me in the project.

DECLARATION

I hereby declare that the project entitled, “**Result Builder**” done at **Guru Nanak Khalsa College**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (COMPUTER SCIENCE)** to be submitted as final semester project as part of our curriculum.

Jagannath B. Patta

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

- 1.1 Objectives**
- 1.2 Scope**
- 1.3 Applicability**
- 1.4 Achievements**

CHAPTER 2: SURVEY OF TECHNOLOGIES

- 2.1 Context**
- 2.2 Existing System**
- 2.3 Proposed System**

CHAPTER 3: REQUIREMENTS AND ANALYSIS

- 3.1 Requirements Specification**
- 3.2 Planning and Scheduling**
 - 3.2.1 GANTT Chart**
 - 3.2.2 WBS Chart**
- 3.3 Software and Hardware Requirements**
- 3.4 Conceptual Models**
 - 3.4.1 ER Diagrams**

CHAPTER 4: SYSTEM DESIGN

- 4.1 Basic Modules**
- 4.2 Data Design**
 - 4.2.1 Schema Design**
 - 4.2.2 Data Integrity and Constraints**

TABLE OF CONTENTS

- 4.3 Procedural Design**
- 4.3.1 Use Case Diagrams**
- 4.4 User interface design**
- 4.5 Security Issues**
- 4.6 Test Cases Design**

CHAPTER 5: IMPLEMENTATION

- 5.1 Backend Coding**
- 5.2 Database Coding**

CHAPTER 6: FINAL REPORT

- 6.1 Future Enhancement**
- 6.2 Conclusion**
- 6.3 References**

Software and Broad Areas of Application

Front End Interaction	Asp.net, CSS(Bootstrap) and HTML
Back End Database	MS SQL (Microsoft Structured Query Language))
Languages used	C#
Tools & Technologies used	Visual Studio 2017 & Microsoft SQL server management Studio.
Areas of Application	Educational

CHAPTER 1: INTRODUCTION

The “Result Builder” has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases reduce the hardships faced by an existing system. The website is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user friendly.

Result Builder, as described above, can lead to error free, secure, reliable and fast. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus, it will help organization in better utilization of resources.

Evey organization, whether big or small, has challenges to overcome and managing the information of Result, Student, Class, Subject, Semester. Every Student Result Management System has different Student needs.

1.1 Objectives

The main objective of the Project on Result Builder is to manage the details of Student, Result, Subject, Class, Semester. It manages all the information about Student, Subject, Semester, Student. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Student, Result, Subject, Subject. It tracks all the details about the Subject, Class, Semester.

1.2 Scope

It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Result Builder. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

Our project aims at Business process automation, i.e. we have tried to computerize various processes of Result Builder.

- 1) In computer system the person has to fill the various forms & number of copies of the forms can be easily generated at a time.
- 2) In computer system, it is not necessary to create the manifest but we can directly print it, which saves our time.
- 3) To assist the staff in capturing the efforts spent on their respective working areas.
- 4) To utilize resources in an efficient manner by increasing their productivity through automation.
- 5) The system generates types of information that can be used for various purposes.
- 6) It satisfies the user requirement.
- 7) Be easy to understand by the user and operator.
- 8) Be easy to operate.
- 9) Have a good user interface.
- 10) Be expandable.
- 11) Delivered on schedule within the budget.

1.3 Applicability

- Provides the searching facilities based on various factors. Such as Student, Subject, Class, Semester.
- Result Builder also manage the Subject details online for Class details, Semester details, Student.
- It tracks all the information of Result, Subject, Class etc.
- Manage the information of Result.
- Shows the information and description of the Student, Result.
- It deals with monitoring the information and transactions of Class.
- Manage the information of Student.
- Editing, adding and updating of Records in improved which results in proper resource management of Student data.
- Manage the information of Class.
- Integration of all records of semester.

1.4 Achievements

I learned a lot of by doing this project

- Web designing :HTML,CSS, Bootstrap
- Language used: C#
- Software tools: Visual studio 2017,Microsoft Sql server management studio 18
- Database : My sql
- Computer Networking Protocols: SMTP , IMAP/POP3

So during this project I learned all above things. Before this project ,I had no idea about Bootstrap, SMPT, POP .Although I had little bit knowledge of C#

Before . but now I learned a lot about C# and got knowledge of using Visual studio for developing web application

CHAPTER 2: SURVEY OF TECHNOLOGIES

2.1 Context

Result Builder is a web-based application designed and engineered for colleges that need to manage results across multiple branches students that need to track, manage and report results. This application can run on any kind of operating system. We download the report card of a student of a particular semester.

2.2 Existing System

There already many systems in the domain used by many institutes for generating results, These systems are basically a combination of three sub systems known as OSA (Overall Assessment System) , RPS (Result Processing System) and RDS (Result Distribution System).

OSA - Overall Assessment of exam paper is been done in this process.

RPS – Result Processing deals with all the calculation of marks.

RDS – Result Distribution is were results get reached to the students.

Result Builder is the combination of RPS & RDS.

2.3 Proposed System

his project is a website build for generating the results of students, it includes storing data of each and every student according to their stream, department, subjects and teacher. It also allows the feature of generating report card on the basis of the score the teacher will enter. It provides the soft copy of the result properly formatted according to the college report card layout and can be downloaded. The report card generating process is only accessible by the authorised faculty members assigned by the admin of the database.

One of the major focus is on the calculation and converting marks into grades and stored. Teachers only have to enter less input credentials and in return will get lots of extra.

Result Builder is the combination of RPS & RDS.

CHAPTER 3: REQUIREMENTS & ANALYSIS

3.1 Requirements Specification

Software requirement specification (SRS) It is the starting point of the software developing activity, as the system grows more complex it become evident that the goal of phase is more. The software project is initiated by the client needs.

The SRS phase consists of two basic activities

- Problem requirement analysis
- Requirement specification

Problem requirement analysis the process is order and more nebulous of the two, ideas with understanding the problem, the goals is to fulfil the constraints posed by the client

Requirement specification Here the focus is on specifying what has been found in the process of analysis such as representation, specification languages and tools and also checking the specification are addressing during the activity. The requirement phase terminates with the production of the validate SRS document, producing SRS document is the basic goals of this phase.

Role of software requirement specification (SRS) The purpose of the software requirement specification is to reduce the communication gap between the clients and developers. Software requirement specification is the medium through which the clients and users' needs are accurately specified. It forms the basis of software development. SRS should specify all the parties and operations involved in the system.

3.2 Planning and Scheduling

Planning and scheduling are the important part of any software development application as this phase decides what the outcome is going to be. Making better software is an important task but completing it on time is even more important for developer. Planning and scheduling is the main key for the software developer. Planning of an application is done in small parts to make the application achieve its goal. Planning is concerned with the development of an application for investment. It identifies and addresses the task required for accomplishment of objectives.

In this phase, the Gantt chart is used for better planning scheduling and understanding.

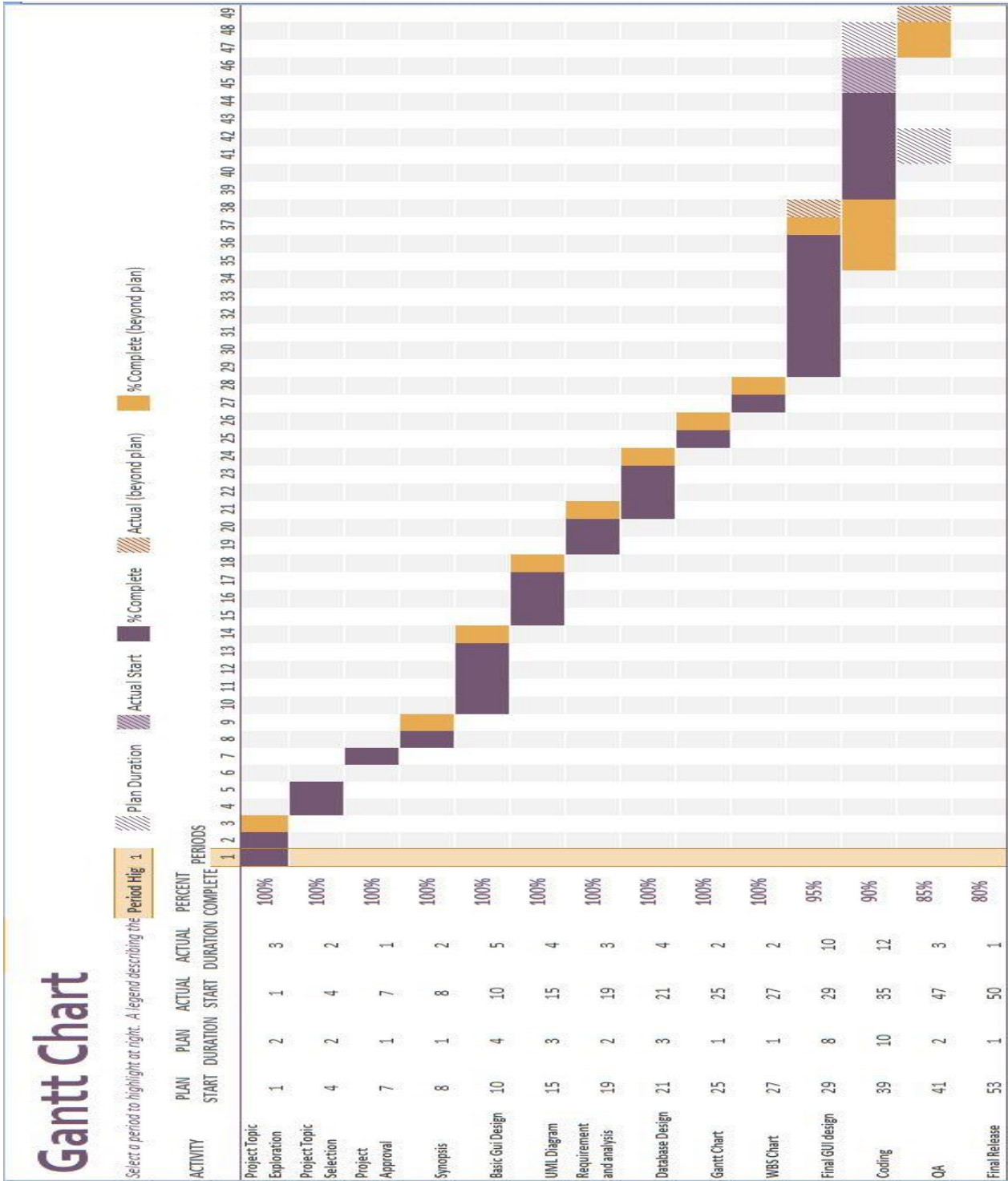
3.2.1 Gantt Chart

It is a graphical representation of graph against the progression of the time. Gantt chart is useful tool for planning and scheduling applications timings. It also schedules the timings of the subparts of the application. They also show the relationship dependency between the activities and the present schedule.

According to this chart, the activities that have to be performed are on the vertical axis and the time intervals are on the horizontal axis.

An elementary Gantt chart or Timeline chart for the development plan is given below. The plan explains the tasks versus the time (in days) they will take to complete.

Fig.1 Gantt Chart



3.2.2 WBS Chart

Work Breakdown structure is a key for a company to divide the application into simpler manageable divisions. It is tree that shows the subparts of effort required to achieve an objective of the application. It is constructed by dividing an application into major components, each of which is further sub-divided into smaller components. The process is continued till a breakdown accomplishes the manageable unit of works for which responsibility can be defined.

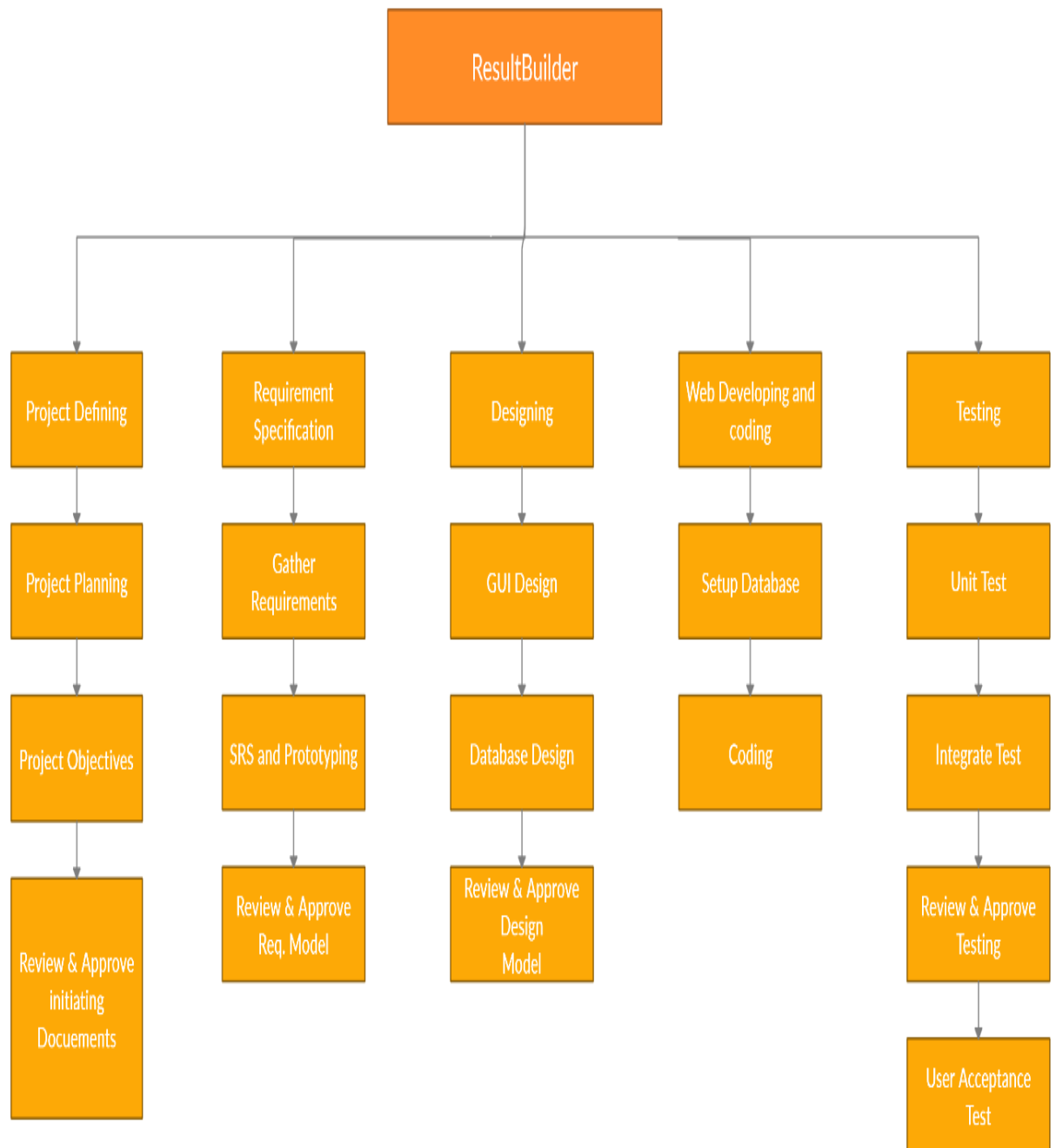


Fig 2 WBS Chart

3.3 Software and Hardware Requirements

Software and hardware requirements are the need of hardware and software for developing Daily Diary Application. Hardware requirements are hardware used by the developer while developing Daily Diary Application.

Hardware Requirements:

- Laptop or Computer
- Processor: Intel core i5
- RAM 8GB

Software Requirements:

- Platform used: Windows 10
- Visual Studio 2017
- Microsoft SQL Server
- Microsoft SQL Server Management Studio
- CSS (Bootstraps files)

3.4 Conceptual Models

3.4.1 ER Diagrams

ER Diagram is the entity-relationship diagram. It is a graphical representation of an information application that depicts the relationships among people, objects, places, concepts or events within that application. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database.

An ERD is often used as a way to visualize a relational database: each entity represents a database table and the relationship lines represent the key in one table that point to specific records in related tables.

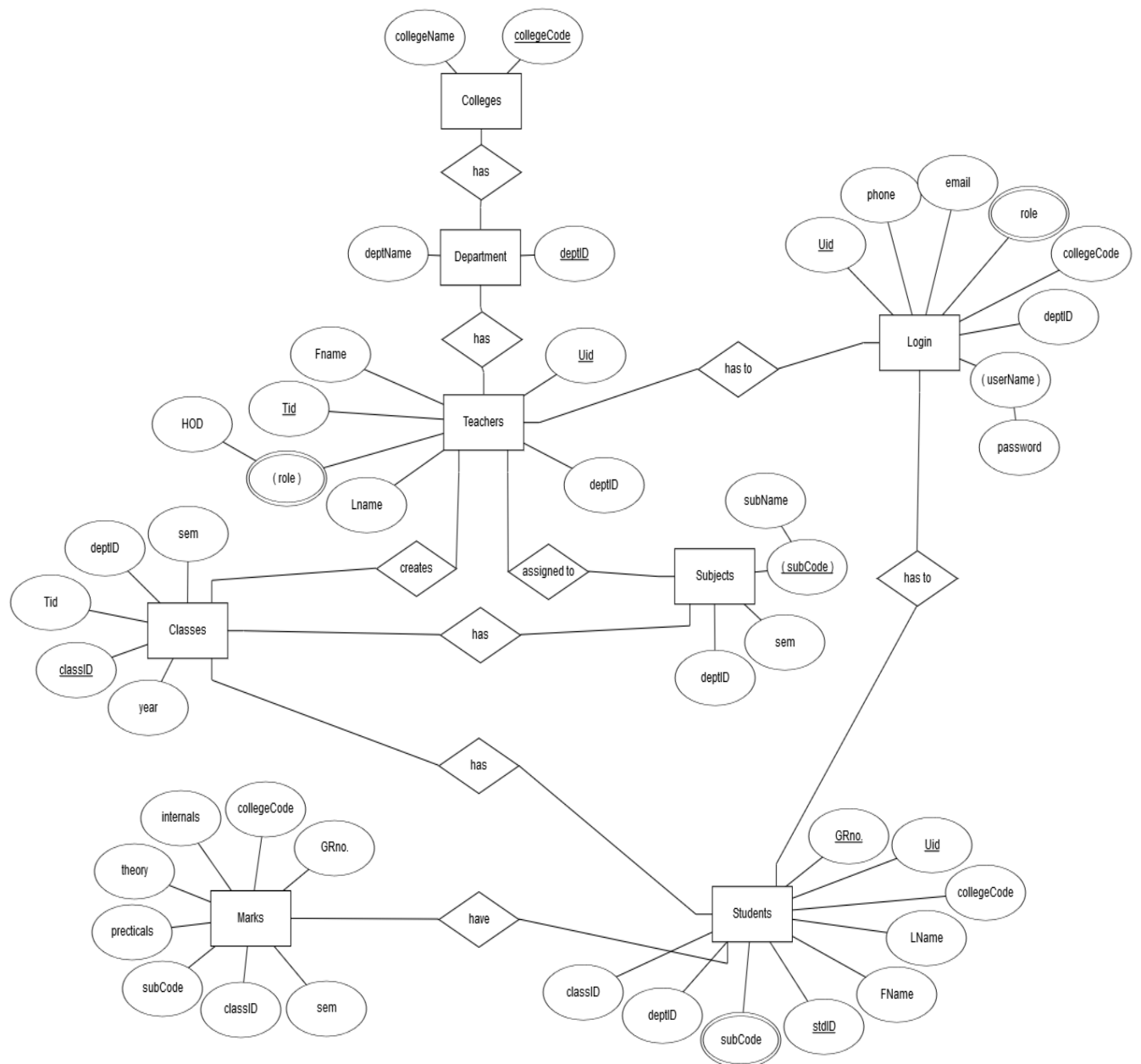


Fig 3. ER Diagram

ER Diagram Explanation

An ERD is a data modelling technique that can help define business processes and be used as the foundation for a relational database.

ER diagram of Result Builder represents an entity and relations between them to show the dependency. In the above figure there are Seven entities which are connected to each other in different types of entity relationships.

There is a college entity connected to the department entity, department entity connected to teacher entity then with the teacher entity Classes and Subjects entities were connected. There is a connection between the classes and the student entity and student and marks entity.

CHAPTER 4: SYSTEM DESIGN

4.1 Basic Modules

There are three modules in this project,

First one is the HOD module.

- HOD Logins with the username and password that the Admin assigned at the time of adding this HOD into the database, at server side.
- HOD creates new classes with subjects and assigns the teacher to class.
- HOD adds, deletes and updates the Teacher to their specific department.
- HOD adds, deletes and updates the Students to their specific department.

Second one is the Teacher module.

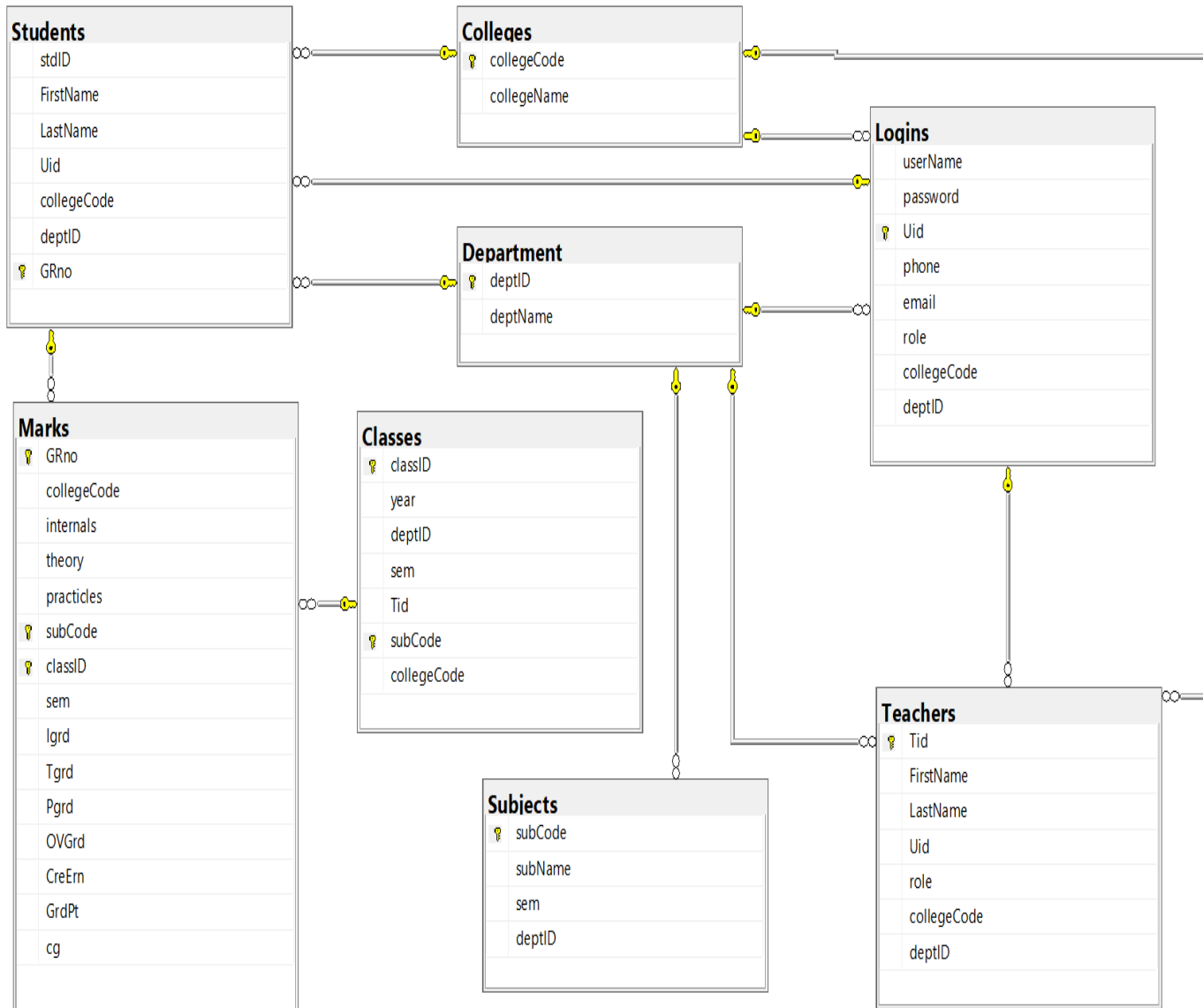
- Teacher Logins with the username and password that the HOD assigned at the time of adding this Teacher into the department.
- Teacher has access to only those classes to which this teacher is been assigned.
- Teacher adds Students in the class from the list of all the students in the department.
- Teacher adds Marks for every student in that class.

Lastly the Student module.

- Student Logins with the username and password that the HOD or Teacher assigned at the time of adding this Student into the department.
- Student chooses the Semester to view the result of that semester.
- Student module is still under development.

4.2 Data Design

4.2.1 Schema Design



Database Diagram

Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the GRno. of a student makes him/her identifiable among students.

- **Super Key** – A set of attributes (one or more) that collectively identifies an entity in an entity set.

- **Candidate Key** – A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** – A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

4.2.2 Data Integrity and Constraints

A data-integrity mechanism is the parent-and-child relationship of related records. If a parent record owns one or more related child records all of the referential integrity processes are handled by the database itself, which automatically ensures the accuracy and integrity of the data so that no child record can exist without a parent (also called being orphaned) and that no parent loses their child records. It also ensures that no parent record can be deleted while the parent record owns any child records. All of this is handled at the database level and does not require coding integrity checks into each application.

4.3 Procedural Design

4.3.1 Use Case Diagrams

Use case diagrams are behavior diagrams used to describe a set of actions that some application should or can perform in collaboration with one or more external users of the application (Actors). Each use case should provide some observable and valuable result to the actor or other stakeholders of the application. The purpose of the use case diagram is to identify the "uses", or use cases in order to identify how the system is used. It is a convenient way to document the functions that the system must support. Sometimes a single and comprehensive use case diagram is used to describe the entire system. At other times, a set of smaller use case diagrams make up the use case model.

The common symbols used in a use case diagram are:

- A simple Stick figure is used to represent an actor. The Stick figure is given a name that characterizes the role being played by the actor.
- The use case itself is symbolized by an oval with the name of the use case inside.
- The lines between the actors and the use cases indicate which actors participate with which use case.

Actor (HOD)

- HOD Logs in with the username and password that the Admin assigned at the time of adding this HOD into the database, at server side.
- HOD creates new classes with subjects and assigns the teacher to class.
- HOD adds, deletes and updates the Teacher to their specific department.
- HOD adds, deletes and updates the Students to their specific department.

Actor (Teacher)

- Teacher Logs in with the username and password that the HOD assigned at the time of adding this Teacher into the department.
- Teacher has access to only those classes to which this teacher is been assigned.
- Teacher adds Students in the class from the list of all the students in the department.
- Teacher add, update and delete Marks for every student in that class.

Actor (Student)

- Student Logins with the username and password that the HOD or Teacher assigned at the time of adding this Student into the department.
- Student chooses the Semester to view the result of that semester.
- Student module is still under development.



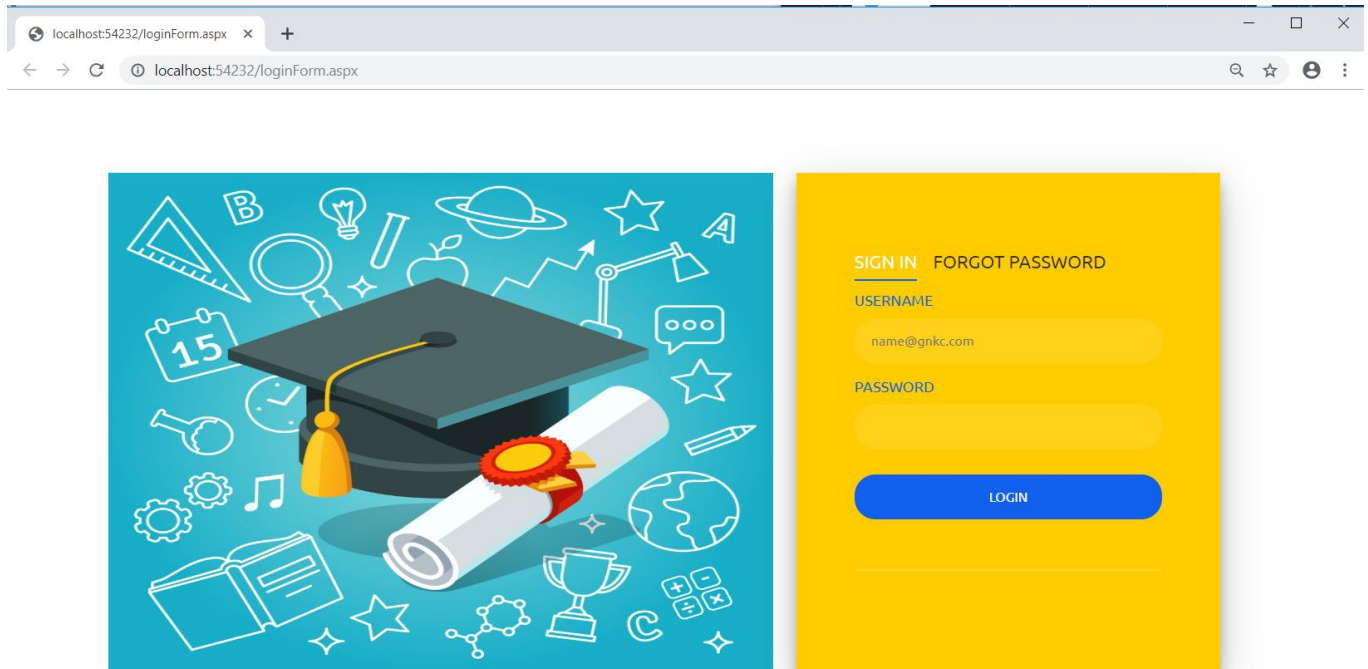
UseCase Diagram

4.4 User interface design

Following screenshots represents the architecture of the proposed System.

loginForm.aspx

There is a single login page for very user type login.



HOD login


There three important pages.

- HodClasses.aspx were HOD creates new classes with subjects and assigns the teacher to class and view every record of the table on the page.
- teachers.aspx were HOD adds, deletes and updates the Teacher to their specific department view every record of the table on the page.
- Students.aspx were HOD adds, deletes and updates the Students to their specific department view every record of the table on the page.

HodClasses.aspx

HOD Home Page

localhost:54232/HodClasses.aspx



Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

Home Classes Teachers Students New Item Logout

List of Classes

ClassID	Academic Year	Sem	SubCode	Sub Name	Teacher	
1018	FY 2019 -	1	USCS101	Computer Organization and Design	Teacher1CS	View
1019	FY 2019 -	1	USCS102	Programmingwith Python-I	Teacher2CS	View
1020	FY 2019 -	1	USCS103	Free and Open Source Software	Teacher3CS	View
1021	FY 2019 -	1	USCS104	Database Systems	Teacher4CS	View
1022	FY 2019 -	1	USCS105	Discrete Mathematics	Teacher1CS	View
1023	FY 2019 -	1	USCS106	Descriptive StatisticsandIntroduction to Probability	Teacher2CS	View
1024	FY 2019 -	1	USCS107	Soft Skills Development	Teacher3CS	View

Create Classes & Assign Teachers

Class ID

Select Class

Select Year

Academic Year

Enter ClassId

FY

2019

Enter AcademicYear

Select Semester

SEM 6

Check the subjects you want to assign Teachers.

Select the Teacher

HOD1

Save

Delete

Clear

Wireless Sensor Networks and Mobile Communication

Cloud Computing

Cyber Forensics

Information Retrieval

Digital Image Processing

Data Science

Ethical Hacking


Copyright 2019 © Guru Nanak Khalsa College, Mumbai - All Rights Reserved
Maintained by Jagannath Patta

Activate Windows
Go to Settings to activate Windows.

teachers.aspx

HOD Home Page

localhost:54232/teachers.aspx



Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

Home Classes Teachers Students New Item Logout

List of all Teachers

Teacher ID	First Name	Role	Aadhar No.	Email ID	Contact No.	
227	HOD1	HOD	1234567891234567	jaggup0@gmail.com	9967824801	View
229	Teacher1CS	Teacher	5236987415236987	teacher1@gmail.com	9513578245	View
230	Teacher2CS	Teacher	5236987415236984	teacher2@gmail.com	9874563210	View
231	Teacher3CS	Teacher	5236987415236981	teacher3@gmail.com	9756312481	View
232	Teacher4CS	Teacher	5236987415236980	teacher4@gmail.com	9463215870	View

Registration Form

First Name

Enter First Name

Last Name

Enter Last Name

Select Role

Teacher

Aadhar No.

Enter Aadhar No.

Phone No.

Enter Phone No.

Email address

Enter email

We'll never share your email with anyone else.

UserName

Enter UserName

Activate Windows
Go to Settings to activate Windows.

students.aspx

HOD Home Page x +

localhost:54232/students.aspx

Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

Home Classes Teachers Students New Item Logout

To Add the Students of Whole class Upload the Excel File Here.

Download a excel templet file to upload data.[Click Here](#)

Upload Excel Fiel: No file chosen

List of all Students

GR No.	Roll No.	First Name	Last Name	Aadhar No.	Email ID	Contact No.	
12345	1	Student1CS		5214789635214789	student1cs@gmail.com	8745631290	View
12346	2	Student2CS		5214789635214786	student2cs@gmail.com	8796325410	View
12347	3	Student3CS		5214789635214783	student3cs@gmail.com	8136547920	View
12348	4	Student4CS		5214789635214782	student4cs@gmail.com	8632145970	View

To Add a Single Student fill the Student Details Here.

Student Details

GR No.

Roll No.

First Name

Last Name

Aadhar No.

Phone No.

Activate Windows
Go to Settings to activate Windows.

Forms on both teacher.aspx and students.aspx pages are created with all proper validation rules.

Student Details

GR No.

Roll No.

First Name

Last Name

Aadhar No.

*Enter valid Aadhar number

Phone No.

*Enter valid Phone number

Email address

We'll never share your email with anyone else.

UserName

UserName should be your First name and clg name e.g "jagannath@khalsa.com"

Password

Confirm Password

*Passwords Does not match

Registration Form

First Name

Last Name

Select Role

Aadhar No.

*Enter valid Aadhar number

Phone No.

*Enter valid Phone number

Email address

We'll never share your email with anyone else.

UserName

UserName should be your First name and clg name e.g "jagannath@khalsa.com"

Password

Confirm Password


*Passwords Does not match

Teacher login

There Four important pages.

- teacherClasses.aspx were Teacher has access to only those classes to which this teacher is been assigned and after click on a view button teacher get redirected to AddStudent.aspx page.
- AddStudent.aspx were Teacher adds Students in the class from the list of all the students in the department by clicking on add button and can remover student from class by clicking remove button.
- Then after clicking on the Add Marks button the teacher gets redirected to EnterMarks.aspx page.
- EnterMarks.aspx were Teacher add, update and delete Marks for every student in that class, and at the back end the marks get converted into the grads and both Marks and grades are stored into the database.

teacherClasses.aspx



Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

[Home](#) [Classes](#) [Subjects](#) [Students](#) [New Item](#) [Logout](#)

List of Classes

Class Id	Acedemic Year	Sem	Subject Code	Subject Name	
1018	FY 2019 -	1	USCS101	Computer Organization and Design	View
1019	FY 2019 -	1	USCS102	Programmingwith Python-I	View
1020	FY 2019 -	1	USCS103	Free andOpen Source Software	View
1021	FY 2019 -	1	USCS104	Database Systems	View
1022	FY 2019 -	1	USCS105	Discrete Mathematics	View
1023	FY 2019 -	1	USCS106	Descriptive StatisticsandIntroduction to Probability	View
1024	FY 2019 -	1	USCS107	Soft Skills Development	View

Copyright 2019 © Guru Nanak Khalsa College, Mumbai - All Rights Reserved
Maintained by Jagannath Patta


Activate Windows
Go to Settings to activate Windows.

AddStudent.aspx

Teacher Home Page

localhost:54232/AddStudent.aspx

☆ ⓘ ⋮



Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

HomeClassesSubjectsStudentsNew Item

Logout

FY 2019 - Computer Organization and Design

GRno	stdID	FirstName	LastName	
12345	1	Student1CS		Add
12346	2	Student2CS		Add
12347	3	Student3CS		Add
12348	4	Student4CS		Add

Students Already In Class

GRno	FirstName	LastName	
12345	Student1CS		Remove
12346	Student2CS		Remove
12347	Student3CS		Remove

Add Marks


Copyright 2019 © Guru Nanak Khalsa College, Mumbai - All Rights Reserved
Maintained by Jagannath Patta

EnterMarks.aspx

Teacher Home Page

localhost:54232/EnterMarks.aspx

☆ ⓘ ⋮



Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

HomeClassesSubjectsStudentsNew Item

Logout

FY 2019 - Computer Organization and Design

GR No.	Roll No.	First Name	Last Name	Internals	Theory	Practical		
12345	1	Student1CS					Edit	Delete
12346	2	Student2CS		Marks out of 25	Marks out of 75	Marks out of 50	Update	Cancel
12347	3	Student3CS		19	49	39	Edit	Delete

student Successfully Updated

Copyright 2019 © Guru Nanak Khalsa College, Mumbai - All Rights Reserved
Maintained by Jagannath Patta

teacherStudents.aspx

The screenshot shows a web browser window with the URL `localhost:54232/teacherStudents.aspx`. The page has a dark blue header with the logo of Shiromani Gurdwara Parbandhak Committee's Guru Nanak Khalsa College of Arts, Science & Commerce (AUTONOMOUS) and its address in Matunga, Mumbai. Below the header is a yellow navigation bar with links: Home, Classes, Subjects, Students, and New Item, along with a Logout button. The main content area is white and contains two sections. The left section, titled 'To Add the Students of Whole class Upload the Excel File Here.', includes a link to download an Excel template, an 'Upload Excel File' button (labeled 'Choose File'), and an 'Upload' button. Below this is a table titled 'List of all Students' with columns for GR No., Roll No., First Name, Last Name, Aadhar No., Email ID, Contact No., and a View button. The table contains four rows of student data. The right section, titled 'To Add a Single Student fill the Student Details Here.', has a 'Student Details' heading and form fields for GR No., Roll No., First Name, and Last Name, each with a 'View' button. A Windows watermark is visible on the right side of the page.

Shiromani Gurdwara Parbandhak Committee's
Guru Nanak Khalsa College of Arts, Science & Commerce
(AUTONOMOUS)
Matunga, Mumbai - 400 019
Accredited by NAAC with 'A' Grade with a CGPA of 3.54

Home Classes Subjects Students New Item Logout

To Add the Students of Whole class Upload the Excel File Here.

Download a excel templet file to upload data.[Click Here](#)

Upload Excel Fiel : No file chosen

List of all Students

GR No.	Roll No.	First Name	Last Name	Aadhar No.	Email ID	Contact No.	
12345	1	Student1CS		5214789635214789	student1cs@gmail.com	8745631290	View
12346	2	Student2CS		5214789635214786	student2cs@gmail.com	8796325410	View
12347	3	Student3CS		5214789635214783	student3cs@gmail.com	8136547920	View
12348	4	Student4CS		5214789635214782	student4cs@gmail.com	8632145970	View

To Add a Single Student fill the Student Details Here.

Student Details

GR No.

Roll No.

First Name

Last Name

Activate Windows
Go to Settings to activate Windows.

Student login

There two important pages.

- StudentHome.aspx were Student chooses the Semester to view the result of that semester and then clicks the click here link button and redirected to Reportcard.aspx page.
- Reportcard.aspx were properly aligned and formatted report card is displayed on the screen with a print button to download as pdf.

StudentHome.aspx

The screenshot shows a web browser window with the URL `localhost:54232/StudentHome.aspx`. The page has a light gray background. On the left, it says 'HI Student1CS' followed by a row of radio buttons for Sem 1, Sem 2, Sem 3, Sem 4, Sem 5, and Sem 6. Below this is a link 'Click Here View Your Result.' On the right, there is a red 'Logout' button.

HI Student1CS

Sem 1 Sem 2 Sem 3 Sem 4 Sem 5 Sem 6

[Click Here View Your Result.](#)

Logout

Reportcard.aspx

localhost:54232/Reportcard.aspx

localhost:54232/Reportcard.aspx

Print

		Grade Card									
Programme :Bachelor of Science (B.Sc.) in Computer Science				SEMESTER : 1							
PRN / Reg.No. : 12345		Examination Seat No. : 38		Month & Year of Examination :							
Name of the Candidate : Jagannath Patta											
Course Code	Course Title	Course Credits	TH	IA	OVERALL	Credits Earned (C)	Grade Points (G)	C x G	SGPI= ECG/EC		
USCS101	Computer Organization and Design	2	B+	A	B+	2	7				
USCS102	Programmingwith Python-I	2	A	O	A+	2	9				
USCS103	Free andOpen Source Software	2	B	O	A	2	8				
USCS104	Database Systems	2	C	O	B+	2	7				
USCS105	Discrete Mathematics	2	D	A+	B	2	6				
USCS106	Descriptive StatisticsandIntroduction to Probability	2	A	A	A	2	8				
USCS107	Soft Skills Development	2	A	O	A+	2	9				
Total											
Remarks : Not Successful		Credit Earned :		SGPI :		Overall Grade:					
Sem 1: Credit Earned :-		SGPI :-	Sem 2: Credit Earned :		SGPI :	Sem 3: Credit Earned :		SGPI :	Sem 4: Credit Earned :		SGPI :
Place : Mumbai		Checked By:		Chairperson Examination Committee		PRINCIPAL					
Date :											

Activate Windows
Go to Settings to activate Windows.

4.5 Security Issues

Security is an essential part for any system. The user will lose his/her faith in the system if the security of the system is compromised. Following are the security approaches that are implemented related to the security for the project

1. Authorization

Authorization is a process by which a server determines if the client has permission to use a resource or access a file.

2. Authentication

Authentication is used by a server when the server needs to know exactly who is accessing their information or site. Authentication is used by a client when the client needs to know that the server is the system it claims to be. In authentication, the user or computer has to prove its identity to the server or client.

3. Confidentiality

The information of the user is not accessible by any unauthorized person.

4. Information Integrity

Only the authorized user has the right for modification of the information. The users only have the right for modification of any information provided by the system.

5. Encryption

Encryption involves the process of transforming data so that it is unreadable by anyone who does not have a decryption key.

6. Secure the Mail Server Application

Organizations should install the minimal mail server services required and eliminate any known vulnerabilities through patches, configurations, or upgrades. If the installation program installs unnecessary applications, services, or scripts, these should be removed immediately after the installation process is complete.

4.6 Test Cases Design

TESTING OBJECTIVES

To ensure that during operation the system will perform as per specification.

- TO make sure that system meets the user requirements during operation
- A good test case is one that has a high probability of finding an as yet undiscovered error
- To see that when correct inputs are fed to the system the outputs are correct
- To make sure that during the operation, incorrect input, processing and output will be detected
- Testing is a process of executing a program with the intent of finding an error
- To verify that the controls incorporated in the same system as intended

Unit Testing

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

Integration Testing

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRE CONDITION	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTE D BY	EXECUTE DATE	COMMENTS
Ts_01	User login	Tc_login_01	check the login with valid username and password	1.Open the url	valid url Test Data	http://localhost/loginForm.asp x HOD1@gnkc.com	user should be redirected to HOD home page	user is redirected to HOD home page	pass	Jagannath Patta	9/25/2019	no comments
				2.Enter valid username								
				3.Enter valid password		123						
				5.Click on Login button								
Ts_01	User login	Tc_login_02	check the login with invalid username and invalid password	1.Open the url	valid url Test Data	http://localhost/loginForm.asp x	user should get an error message i.e Invalid Credentials	user is getting an error message of Invalid Credentials	pass	Jagannath Patta	9/25/2019	no comments
				2.Enter invalid username		abcdefgh						
				3.Enter invalid password		123456						
				5.Click on Login button								
Ts_01	User login	Tc_login_03	check the login with Teacher username and password	1.Open the url	valid url Test Data	http://localhost/loginForm.asp x teacher1cs@gnkc.com	user should be redirected to Teacher home page	user is redirected to Teacher home page	pass	Jagannath Patta	9/25/2019	no comments
				2.Enter Teacher username								
				3.Enter valid password		123						
				4.Click on Login button								
Ts_01	User login	Tc_login_04	check the login with Student username and password	1.Open the url	valid url Test Data	http://localhost/loginForm.asp x studen1cs@gnkc.com	user should be redirected to Student home page	user is redirected to Student home page	pass	Jagannath Patta	9/25/2019	no comments
				2.Enter valid username								
				3.Enter valid password		123						
				4.Click on Login button								

TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRE CONDITION	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTED BY	EXECUTE DATE	COMMENTS
Ts_05	Add Teacher	Tc_addTeacher_01	check if the HoD is able to Add Teacher	1.Open the url 2.Enter First Name 3.Enter Last Name 4.Select Role 5.Enter Aadhar No. 6. Enter Phone No. 7. Enter Email Address 8. Enter Username 9. Enter Password 7.Click on Submit button	valid url Test Data	http://localhost/teachers.aspx Teacher5cs Teacher 123654789654 9874563210 teacher5@gmail.com teacher5cs@gnkc.com 123	Teacher should be Added & Viewed in Table	Teacher is Added & listed on the table	pass	Jagannath Patta	9/25/2019	no comments
Ts_06	Update Teacher	Tc_updateTeacher_01	check if the HoD is able to update Teacher details	1.Open the url 2.Click on View button 3.Enter Last Name 4.Select Role 5. Enter Phone No. 7. Enter Email Address 7.Click on Update button	valid url Test Data	http://localhost/teachers.aspx Some thing Teacher 9874563299 teacher55@gmail.com	Teacher Details should be Updated & Viewed in Table	Teacher Details are updated & listed on the table	pass	Jagannath Patta	9/25/2019	no comments
Ts_07	Delete Teacher	Tc_deleteTeacher_01	check if the HoD is able to delete Teacher	1.Open the url 2.Click on View button 7.Click on Delete button	valid url Test Data	http://localhost/teachers.aspx	Teacher Details should be Deleted Table	Teacher Details are Deleted from the table	pass	Jagannath Patta	9/25/2019	no comments

TEST SCENARIO ID	TEST SCENARIO DESCRIPTION	TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	PRE CONDITION	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS	EXECUTE D BY	EXECUTE DATE	COMMENTS
Ts_08	Add Student	Tc_addStudent_01	check if the HoD is able to Add Student	1.Open the url 2.Enter GR no. 3.Enter Roll no. 4.Enter First Name 5.Enter Last Name 6.Enter Aadhar No. 7. Enter Phone No. 8. Enter Email Address 9. Enter Username 10. Enter Password 11.Click on Submit button	valid url Test Data	http://localhost/student.aspx 123459 6 Student5cs 123654789654 9874563210 studentcs5@gmail.com student5cs@gnkc.com 123	Student should be Added & Viewed in Table	Student is Added & listed on the table	pass	Jagannath Patta	9/25/2019	no comments
Ts_06	Update Student	Tc_updateStudent_01	check if the HoD is able to update Student details	1.Open the url 2.Click on View button 3.Enter Last Name 4. Enter Phone No. 5. Enter Email Address 6.Click on Update button	valid url Test Data	http://localhost/student.aspx Some thing 9874563299 Student55@gmail.com	Student Details should be Updated & Viewed in Table	Student Details are updated & listed on the table	pass	Jagannath Patta	9/25/2019	no comments
Ts_07	Delete Student	Tc_deleteStudent_01	check if the HoD is able to delete Student	1.Open the url 2.Click on View button 7.Click on Delete button	valid url Test Data	http://localhost/student.aspx	Student Details should be Deleted Table	Student Details are Deleted from the table	pass	Jagannath Patta	9/25/2019	no comments

CHAPTER 5: IMPLEMENTATION

5.1 Backend Coding

loginForm.aspx.cs

```
using System.Data.SqlClient;
using System.Data;
using System.Data;
using System.Data.SqlClient;
using System.Net.Mail;
using System.Net;

namespace RB
{
    public partial class loginForm : System.Web.UI.Page
    {
        protected void btn_login_Click(object sender, EventArgs e)
        {
            if (txt_Password.Text.Equals("") || txt_UserName.Text.Equals(""))
            {
                status.Text = "*Invalid Credentials";
            }
            else
            {
                SqlConnection connection = new SqlConnection("Data Source=localhost;Initial
Catalog=ResultsBuilder;Integrated Security=True");
                connection.Open();

                string query = "SELECT COUNT(1) FROM Logins WHERE userName=@un
AND password=@p";

                SqlCommand sqlcmd = new SqlCommand(query, connection);
                sqlcmd.Parameters.AddWithValue("@un", txt_UserName.Text.Trim());
                sqlcmd.Parameters.AddWithValue("@p", txt_Password.Text.Trim());
                int count = Convert.ToInt32(sqlcmd.ExecuteScalar());
                if (count == 1){
```

```
string query2 = "SELECT * FROM Logins WHERE userName=@un AND  
password=@p";
```

```
SqlCommand sqlcmd2 = new SqlCommand(query2, connection);  
sqlcmd2.Parameters.AddWithValue("@un", txt_UserName.Text.Trim());  
sqlcmd2.Parameters.AddWithValue("@p", txt_Password.Text.Trim());  
SqlDataReader dataReader = sqlcmd2.ExecuteReader();  
dataReader.Read();
```

```
string role = dataReader[5].ToString();  
Session["uid"] = Convert.ToDouble(dataReader[2]);  
Session["email"] = dataReader[4].ToString();  
Session["clgCode"] = Convert.ToInt32(dataReader[6]);  
Session["depId"] = Convert.ToInt32(dataReader[7]);  
Console.WriteLine(role);
```

```
if (role == "HOD"){  
    Session["Username"] = txt_UserName.Text.Trim();  
    Server.Transfer("~/hodHome.aspx", false);  
}
```

```
else if (role == "Teacher"){  
    Session["Username"] = txt_UserName.Text.Trim();  
    Response.Redirect("~/teacherHome.aspx", false);  
}
```

```
else if (role == "Student"){  
    Session["Username"] = txt_UserName.Text.Trim();  
    Response.Redirect("~/StudentHome.aspx", false);  
}
```

```
}  
else{    status.Text = "*Invalid Credentials";    }
```

```
}  
}
```

```
protected void btnsnpd_Click(object sender, EventArgs e)
```

```

{
    if (TextBox1.Text.Equals("") || TextBox2.Text.Equals(""))
        Label8.Text = "*Invalid Credentials";
    else{
        String password;

        String myquery = "Select * from Logins where userName='" + TextBox1.Text + "'
and email='" + TextBox2.Text + "'";

        SqlConnection con = new SqlConnection("Data Source=localhost;Initial
Catalog=ResultsBuilder;Integrated Security=True");

        SqlCommand cmd = new SqlCommand();

        cmd.CommandText = myquery;

        cmd.Connection = con;

        SqlDataAdapter da = new SqlDataAdapter();

        da.SelectCommand = cmd;

        DataSet ds = new DataSet();

        da.Fill(ds);

        if (ds.Tables[0].Rows.Count > 0){
            password = ds.Tables[0].Rows[0]["password"].ToString();

            sendpassword(password, TextBox2.Text);

            Label8.Text = "Your Password Has Been Sent to Registered Email Address.
Check Your Mail Inbox";
        }
        else {
            Label8.Text = "Your Username is Not Valid or Email Not Registered";
        }

        con.Close();
    }
}

private void sendpassword(String password, String email)
{
    SmtplibClient smtp = new SmtplibClient();

```

```

smtp.Host = "smtp.gmail.com";
smtp.Port = 587;
smtp.Credentials = new System.Net.NetworkCredential\("dummy@gmail.com",
"password");
smtp.EnableSsl = true;
MailMessage msg = new MailMessage();
msg.Subject = "Recovered Password ( ResultBuilder )";
msg.Body = "Dear " + TextBox1.Text + ", Your Password is " + password +
"\n\nThanks & Regards\nResult Builder Team";
string toaddress = TextBox2.Text;
msg.To.Add(toaddress);
string fromaddress = "ResultBuilder <rajbatra080@gmail.com>";
msg.From = new MailAddress(fromaddress);
try {
    smtp.Send(msg);
} catch
{
    throw;
}
}

```

HodClasses.aspx

```

using System.Data.SqlClient;
using System.Data;
namespace RB
{
    public partial class WebForm4 : System.Web.UI.Page
    {
        SqlConnection connection = new SqlConnection("Data Source=localhost;Initial
Catalog=ResultsBuilder;Integrated Security=True");
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["Username"] != null)
            {
                if (!IsPostBack)
                {

```



```

        int year = DateTime.Now.Year;
        for (int i = year; i <= year + 4; i++)
        {
            ListItem li = new ListItem(i.ToString());
            yr.Items.Add(li);
        }

        btnDelete.Enabled = false;
        fillGridView();
    }
}

protected void btnSave_Click(object sender, EventArgs e)
{
    if (connection.State == ConnectionState.Closed)
        connection.Open();

    int count= 1;
    if (btnSave.Text.Equals("Save"))
    {
        string query = "SELECT COUNT(1) FROM Classes WHERE year=@year AND
subCode =@sc AND collegeCode=@clgcode";

        SqlCommand sqlcmd = new SqlCommand(query, connection);
        sqlcmd.Parameters.AddWithValue("@year", txtAY.Text.Trim());
        sqlcmd.Parameters.AddWithValue("@sc", RadioButtonList1.SelectedValue);
        sqlcmd.Parameters.AddWithValue("@clgcode", Session["clgCode"]);
        count = Convert.ToInt32(sqlcmd.ExecuteScalar());
    }

    if(count == 0){
        adddata(RadioButtonList1.SelectedValue.ToString());
        clear();
        susTxt.Text = "Save sucessfully";
    }
}

```

```

    }
    else if (count == 1)
    {
        adddata(txtsubcode.Text);

        clear();

        Page.Response.Redirect(Page.Request.Url.ToString(), true);

        susTxt.Text = "Updated Sucessfully";
    }
    else
    {
        susTxt.Text = "";

        failTxt.Text = "Teacher to this subject is already assigned.";
    }

    fillGridView();

    Page.Response.Redirect(Page.Request.Url.ToString(), true);
}

```

```

void adddata(String Subcode)
{
    if (connection.State == ConnectionState.Closed)
        connection.Open();

    SqlCommand sqlCmd = new SqlCommand("classCreateOrUpdate", connection);
    sqlCmd.CommandType = CommandType.StoredProcedure;

    sqlCmd.Parameters.AddWithValue("@classID", (HiddenField1.Value == "" ? 0 :
Convert.ToInt32(HiddenField1.Value)));

    sqlCmd.Parameters.AddWithValue("@year", txtAY.Text);
    sqlCmd.Parameters.AddWithValue("@deptID", Session["depId"]);
    sqlCmd.Parameters.AddWithValue("@sem", sem.SelectedValues);

    sqlCmd.Parameters.AddWithValue("@Tid", Convert.ToInt32(
DropDownList1.SelectedValue));

```

```

sqlCmd.Parameters.AddWithValue("@subCode", Subcode);
sqlCmd.Parameters.AddWithValue("@clgcode", Session["clgCode"]);
sqlCmd.ExecuteNonQuery();
connection.Close();
clear();
}

protected void grid_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    grid.PageIndex = e.NewPageIndex;
    fillGridView();
}

protected void btnDelete_Click(object sender, EventArgs e)
{
    try
    {
        if (connection.State == ConnectionState.Closed)
            connection.Open();

        SqlCommand sqlcmd = new SqlCommand("classDeleteById", connection);
        sqlcmd.CommandType = CommandType.StoredProcedure;
        sqlcmd.Parameters.AddWithValue("@classID",
Convert.ToInt32(HiddenField1.Value));

        sqlcmd.Parameters.AddWithValue("@subCode", grid.SelectedRow.Cells[3].Text);
        sqlcmd.ExecuteNonQuery();
        connection.Close();
        clear();
        fillGridView();
        susTxt.Text = "Deleted Sucessfully";
    } catch (Exception ex)
    {
        failTxt.Text = ex.Message.ToString();
    }

    Page.Response.Redirect(Page.Request.Url.ToString(), true);
}

```

```

    }

    protected void grid_SelectedIndexChanged(object sender, EventArgs e)
    {
        GridViewRow gr = grid.SelectedRow;

        txtCid.Text = gr.Cells[0].Text;

        txtAY.Text = gr.Cells[1].Text;

        HiddenField1.Value = gr.Cells[0].Text;

        sem.SelectedValue = gr.Cells[2].Text;

        clsyr.Enabled = yr.Enabled = Panel1.Visible = false;

        Panel2.Visible = true;

        txtsubcode.Text = gr.Cells[3].Text;

        txtsub.Text = gr.Cells[4].Text;

        DropDownList1.SelectedIndex = -1;

        DropDownList1.Items.FindByText(gr.Cells[5].Text).Selected = true;

        // DropDownList1.SelectedItem.Text = gr.Cells[5].Text;

        btnSave.Text = "Update";

        btnDelete.Enabled = true;
    }
}

```

TeacherClasses.aspx.cs

```

void fillGridView()
{
    if (connection.State == ConnectionState.Closed)
        connection.Open();

    SqlDataAdapter sqlCmd = new SqlDataAdapter("ViewClassesToTeacher",
connection);

    sqlCmd.SelectCommand.CommandType = CommandType.StoredProcedure;

    sqlCmd.SelectCommand.Parameters.AddWithValue("@depId", Session["depId"]);

    sqlCmd.SelectCommand.Parameters.AddWithValue("@tid", Session["tid"]);

    DataTable dtb1 = new DataTable();
}

```

```

sqlCmd.Fill(dtb1);

connection.Close();

GridView1.DataSource = dtb1;

GridView1.DataBind();
}

protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{
    GridViewRow gr = GridView1.SelectedRow;

    Session["clsid"] = Convert.ToInt32( gr.Cells[0].Text);

    Session["year"] = gr.Cells[1].Text;

    Session["sem"] = Convert.ToInt32(gr.Cells[2].Text);

    Session["subcode"] = gr.Cells[3].Text;

    Session["subname"] = gr.Cells[4].Text;

    Response.Redirect("AddStudent.aspx");
}

```

AddStudent.aspx.cs

```

int clsid, sem;

String year, subcode, subname;

SqlConnection connection = new SqlConnection("Data Source=localhost;Initial
Catalog=ResultsBuilder;Integrated Security=True");

protected void GridView2_SelectedIndexChanged(object sender, EventArgs e)
{
    GridViewRow gr = GridView2.SelectedRow;

    try{
        if (connection.State == ConnectionState.Closed)
            connection.Open();

        SqlCommand sqlCmd = new SqlCommand("DeleteFromMarks", connection);

        sqlCmd.CommandType = CommandType.StoredProcedure;

        sqlCmd.Parameters.AddWithValue("@grno", Convert.ToInt32(gr.Cells[0].Text));
    }
}

```

```

        sqlCmd.Parameters.AddWithValue("@clgcode", Session["clgCode"]);
        sqlCmd.Parameters.AddWithValue("@subcode", Session["subcode"]);
        sqlCmd.Parameters.AddWithValue("@clsid", Session["clsid"]);
        sqlCmd.Parameters.AddWithValue("@sem", Session["sem"]);
        sqlCmd.ExecuteNonQuery();
        connection.Close();

        Page.Response.Redirect(Page.Request.Url.ToString(), true);
    }catch (Exception ex)
    {
        Panel2.Visible = true;
        lbl.Text = ex.Message.ToString();
    }
}

protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{
    GridViewRow gr = GridView1.SelectedRow;
    try{
        if (connection.State == ConnectionState.Closed)
            connection.Open();

        SqlCommand sqlCmd = new SqlCommand("InsertIntoMarks", connection);
        sqlCmd.CommandType = CommandType.StoredProcedure;
        sqlCmd.Parameters.AddWithValue("@grno", Convert.ToInt32(gr.Cells[0].Text));
        sqlCmd.Parameters.AddWithValue("@clgcode", Session["clgCode"]);
        sqlCmd.Parameters.AddWithValue("@subcode", Session["subcode"]);
        sqlCmd.Parameters.AddWithValue("@clsid", Session["clsid"]);
        sqlCmd.Parameters.AddWithValue("@sem", Session["sem"]);
        sqlCmd.ExecuteNonQuery();
        connection.Close();
    }
}

```

```

        Page.Response.Redirect(Page.Request.Url.ToString(), true);
    }catch(Exception ex)
    {
        Panel2.Visible = true;
        flbl.Text = ex.Message.ToString();
    }
}

```

Reportcard.aspx.cs

protected void **Page_Load**(object sender, EventArgs e)

```

{
    fillGridView();
    if (Convert.ToInt32(Session["depId"]) == 120)
    {
        lbprg.Text = "Bachelor of Science (B.Sc.) in Computer Science";
    }
    else if (Convert.ToInt32(Session["depId"]) == 140)
    {
        lbprg.Text = "Bachelor of Science (B.Sc.) in Information Technology";
    }
    lbsem.Text = Session["sem"].ToString();
    lbprn.Text = Session["grno"].ToString();
    lbstno.Text = Session["stdid"].ToString();
    lbname.Text = Session["Fname"].ToString() + " " + Session["lname"].ToString();
}

```

void **fillGridView**()

```

{
    if (connection.State == ConnectionState.Closed)
        connection.Open();

    SqlDataAdapter sqlCmd = new SqlDataAdapter("GenerateResult", connection);
    sqlCmd.SelectCommand.CommandType = CommandType.StoredProcedure;
    sqlCmd.SelectCommand.Parameters.AddWithValue("@grno", Session["grno"]);
    sqlCmd.SelectCommand.Parameters.AddWithValue("@sem", Session["sem"]);
    DataTable dtb1 = new DataTable();
}

```

```
sqlCmd.Fill(dtb1);  
connection.Close();  
GridView1.DataSource = dtb1;  
GridView1.DataBind();  
}
```

```
protected void print_Click(object sender, EventArgs e)
```

```
{          exportpdf();    }
```

```
private void exportpdf()
```

```
{
```

```
    Response.ContentType = "application/pdf";
```

```
    Response.AddHeader("content-disposition", "attachment;filename=reportcard.pdf");
```

```
    Response.Cache.SetCacheability(HttpCacheability.NoCache);
```

```
    StringWriter sw = new StringWriter();
```

```
    HtmlTextWriter hw = new HtmlTextWriter(sw);
```

```
    Panel1.RenderControl(hw);
```

```
    StringReader sr = new StringReader(sw.ToString());
```

```
    Document pdfDoc = new Document(PageSize.A4.Rotate(), 10f, 10f, 100f, 0f);
```

```
    HTMLWorker htmlparser = new HTMLWorker(pdfDoc);
```

```
    PdfWriter.GetInstance(pdfDoc, Response.OutputStream);
```

```
    pdfDoc.Open();
```

```
    htmlparser.Parse(sr);
```

```
    pdfDoc.Close();
```

```
    Response.Write(pdfDoc);
```

```
    Response.End();
```

```
}
```


5.2 Database Coding

/****** Object: StoredProcedure [dbo].[classCreateOrUpdate] *****/

CREATE PROC [dbo].[classCreateOrUpdate]

@classID int, @year nchar(10), @deptID int, @sem int, @Tid smallint, @subCode
nchar(10), @clgcode int

AS

BEGIN

IF (@classID=0)

BEGIN

INSERT INTO Classes(year, deptID, sem, Tid,subCode,collegeCode)VALUES (@year,
@deptID, @sem,@ Tid,@subCode,@clgcode)

END

ELSE

BEGIN

UPDATE Classes SET Tid = @Tid WHERE classID=@classID AND subCode=@subCode

END

END

/****** Object: StoredProcedure [dbo].[classDeleteById] *****/

CREATE PROC [dbo].[classDeleteById]

@classID int, @subCode nchar(10)

AS

BEGIN

DELETE FROM Classes WHERE classID =@classID AND subCode = @subCode

END

/****** Object: StoredProcedure [dbo].[classViewAll] *****/

CREATE PROC [dbo].[classViewAll]

@clgcode int

AS

BEGIN

SELECT C.classID , C.year , C.sem , C.subCode , S.subName , T.FirstName

```
FROM Classes C LEFT JOIN Subjects AS S ON C.subCode = S.subCode AND C.deptID = S.deptID
```

```
LEFT JOIN Teachers AS T On C.deptID = T.deptID And C.Tid = T.Tid And C.collegeCode = T.collegeCode
```

```
WHERE C.deptID= S.deptID AND C.collegeCode = @clgcode
```

```
END
```

```
/****** Object: StoredProcedure [dbo].[DeleteFromMarks] *****/
```

```
create proc [dbo].[DeleteFromMarks]
```

```
@grno int, @clgcode int, @subcode nchar(10), @clsid int, @sem int
```

```
As
```

```
Begin Delete from Marks where GRno = @grno And collegeCode=@clgcode And subCode=@subcode And sem=@sem And classID=@clsid
```

```
End
```

```
/****** Object: StoredProcedure [dbo].[GenerateResult] *****/
```

```
CREATE proc [dbo].[GenerateResult]
```

```
@grno int, @sem int
```

```
as
```

```
Begin
```

```
SELECT M.subCode , S.subName ,M.internals ,M.theory , M.practicles , M.Tgrd , M.Igrd , M.Pgrd ,M.OVGrd , M.CreErn ,M.GrdPt , M.cg
```

```
FROM Marks M LEFT JOIN Subjects AS S ON M.subCode = S.subCode
```

```
WHERE M.GRno = @grno And M.sem = @sem
```

```
End
```

```
/****** Object: StoredProcedure [dbo].[StudentCreateOrUpdate] *****/
```

```
CREATE PROC [dbo].[StudentCreateOrUpdate]
```

```
@Uid bigint, @role varchar(10), @collegeCode int, @deptID int, @userName varchar(20),  
@password varchar(10), @phone bigint, @email varchar(25), @stdID int, @FirstName  
varchar(25), @LastName varchar(10), @GRno int
```

```
AS
```

```
BEGIN
```

```
IF NOT EXISTS (SELECT * FROM Students WHERE GRno = @GRno AND Uid =  
@Uid)
```

```

BEGIN

INSERT INTO Logins (userName,password,Uid,phone,email,role,collegeCode,deptID)
      VALUES( @userName,@password,@Uid,@phone,@email,@role,@collegeCode,@deptID)

INSERT INTO Students ( stdID, FirstName,LastName,Uid,collegeCode,deptID ,GRno)
      VALUES( @stdID,
      @FirstName,@LastName,@Uid,@collegeCode,@deptID,@GRno)

END

ELSE

BEGIN

      UPDATE Students SET stdID = @stdID, FirstName = @FirstName, LastName =
      @LastName , collegeCode = @collegeCode, deptID = @deptID

      where GRno =@GRno AND Uid =@Uid

      UPDATE Logins SET userName =@userName, password = @password, phone = @phone,
      email =@email, role = @role, collegeCode = @collegeCode, deptID = @deptID

      where Uid = @Uid

END

END

/***** Object: StoredProcedure [dbo].[StudentsAlreadyInClass] *****/

Create proc [dbo].[StudentsAlreadyInClass]

@clsid int

As

Begin

SELECT M.GRno , S.FirstName ,S.LastName

FROM Marks M LEFT JOIN Students AS S ON M.GRno = S.GRno

WHERE M.classID = @clsid

End

/***** Object: StoredProcedure [dbo].[StudentViewAll] *****/

CREATE PROC [dbo].[StudentViewAll]

@deptID int, @collegeCode int

AS

BEGIN

```

```
SELECT S.GRno , S.stdID , S.FirstName ,S.LastName , S.Uid ,L.email , L.phone  
FROM Students S    LEFT JOIN Logins AS L ON S.deptID = L.deptID AND  
S.collegeCode = L.collegeCode AND S.Uid = L.Uid
```

```
WHERE S.collegeCode=@collegeCode AND S.deptID =@deptID
```

```
END
```

```
/****** Object: StoredProcedure [dbo].[StudentViewById] *****/
```

```
CREATE proc [dbo].[StudentViewById]
```

```
@GRno int, @Uid bigint
```

```
As
```

```
BEGIN
```

```
SELECT S.GRno , S.stdID , S.FirstName ,S.LastName , S.Uid , L.userName , L.password  
,L.email , L.phone , S.collegeCode , S.deptID , L.role
```

```
FROM Students S    LEFT JOIN Logins AS L ON S.deptID = L.deptID AND  
S.collegeCode = L.collegeCode AND S.Uid = L.Uid
```

```
WHERE S.GRno=@GRno AND S.Uid =@Uid
```

```
END
```

```
/****** Object: StoredProcedure [dbo].[TeacherCreateOrUpdate] *****/
```

```
CREATE PROC [dbo].[TeacherCreateOrUpdate]
```

```
@Tid smallint, @FirstName varchar(10), @LastName Varchar(10), @Uid bigint, @role  
varchar(10), @collegeCode int, @deptID int, @userName varchar(20), @password  
varchar(10), @phone bigint, @email varchar(25)
```

```
AS
```

```
BEGIN
```

```
IF (@Tid = 0)
```

```
BEGIN
```

```
INSERT INTO Logins (userName,password,Uid,phone,email,role,collegeCode,deptID)  
VALUES(@userName,@password,@Uid,@phone,@email,@role,@collegeCode,@d  
eptID)INSERT INTO Teachers (FirstName,LastName,Uid,role,collegeCode,deptID)  
VALUES(@FirstName,@LastName,@Uid,@role,@collegeCode,@deptID)
```

```
END
```

```
ELSE
```

```
BEGIN
```

```
UPDATE Logins SET userName =@userName,password = @password,phone = @phone,  
email =@email,role = @role, collegeCode = @collegeCode,deptID = @deptID  
where Uid =@Uid
```

```
UPDATE Teachers SET FirstName = @FirstName, LastName = @LastName ,role  
=@role, collegeCode = @collegeCode, deptID = @deptID
```

```
where Tid =@Tid And Uid =@Uid
```

```
END
```

```
END
```

```
/****** Object: StoredProcedure [dbo].[TeachersViewAll] *****/
```

```
CREATE PROC [dbo].[TeachersViewAll]
```

```
@deptID int, @collegeCode int
```

```
AS
```

```
BEGIN
```

```
SELECT T.Tid , T.FirstName , T.role , T.Uid , L.email , L.phone
```

```
FROM Teachers T LEFT JOIN Logins AS L ON T.deptID = L.deptID AND  
T.collegeCode = L.collegeCode AND T.Uid = L.Uid
```

```
WHERE T.collegeCode=@collegeCode AND T.deptID =@deptID
```

```
END
```

```
/****** Object: StoredProcedure [dbo].[TeacherViewById] *****/
```

```
CREATE PROC [dbo].[TeacherViewById]
```

```
@Tid int, @Uid bigint
```

```
AS
```

```
BEGIN
```

```
SELECT T.Tid ,T.FirstName , T.LastName , T.role , T.Uid , L.phone , L.email , L.userName  
, L.password ,T.collegeCode , T.deptID FROM Teachers T Left join Logins as L on T.Uid =  
L.Uid
```

```
WHERE T.Tid =@Tid AND T.Uid = @Uid
```

```
END
```

```
/* ***Object: StoredProcedure [dbo].[UpdateIntoMarks] ***
```

```
CREATE proc [dbo].[UpdateIntoMarks]
```

```
@grno int, @subcode nchar(10),@internal smallint, @theory smallint, @pract smallint, @ig  
nchar, @tg nchar, @pg nchar, @og nchar, @ce int, @gp int
```

As

begin

Update Marks set internals = @internal,theory = @theory, practicles = @pract, Igrd =
@ig,Tgrd = @tg,Pgrd = @pg,OVGrd = @og,CreErn =@ce,GrdPt = @gp, cg = @ce * @gp

where GRno = @grno And subCode=@subcode

End

/* Object: StoredProcedure [dbo].[ViewClassesToTeacher] */

CREATE proc [dbo].[ViewClassesToTeacher]

@depid int, @tid int

As

BEGIN

SELECT C.classID , C.year , C.sem , C.subCode , S.subName

FROM Classes C LEFT JOIN Subjects AS S ON C.subCode = S.subCode

WHERE C.Tid = @tid

End

/** Object: StoredProcedure [dbo].[ViewMarks] Script Date: 9/26/2019 9:56:46 PM **/

create proc [dbo].[ViewMarks]

@subcode nchar(10), @clsid int

As

Begin

Select M.GRno ,S.stdID, S.FirstName ,S.LastName,M.internals,M.theory,M.practicles

FROM Marks M LEFT JOIN Students AS S ON M.GRno = S.GRno

where M.classID=@clsid AND M.subCode = @subcode

End

CHAPTER 6: FINAL REPORT

6.1 Future Enhancement

- Because of the time limits I couldn't complete some of the tasks which I assure you to complete in the future.
- The Student module is still under work I assure you to complete.
- Provision of the above mentioned limitations will improve the quality and working of the system.

6.2 Conclusion

- This project is basically built to reduce the manual work done in the process of generating the results.
- Using this project this whole process of generating the result becomes so fast.
- So that the students get there results on time.
- And the results will be stored into the database for long-long years.
- Due to time constraints it was not possible to incorporate all the concepts related to the topic.
- So the program created is just an instance of the Combination of Online Result Processing Syste & Result Distribution System.

6.3 References

- <https://bootswatch.com/united/>
- YouTube channel: Hariti Study Hub – Easy Learn
https://www.youtube.com/channel/UCXU84JJI_wBcPICovba415w
- Co - Tester: Manpreet Singh Saini (TYCS - 449)
- Error Solutions : <https://stackoverflow.com/>