A Project report on

STUDENT RECORD MANAGEMENT SYSTEM – SRMS

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

BACHELOR OF SCIENCE – DATA SCIENCE (M.S. Ds)

By

K.MOUNIKA (197210)

A.KRISHNA DEEPTHI (197265)

M.VIJAYA LAKSHMI (197266)

B.DEVIKA (197270)

Under the esteemed guidance of

MR. K. SUDHEER
Lecturer, Department of Computer Science

DEPARTMENT OF COMPUTER SCIENCE

P. B. SIDDHARTHA COLLEGE OF ARTS AND SCIENCE

VIJAYAWADA – 10

An Autonomous college under the jurisdiction of Krishna University, Machilipatnam.

(A college with potential for excellence; Re – accredited at the level 'A+' by NAAC, Bangalore)
2021 – 2022

P. B. SIDDHARTHA COLLEGE OF ARTS AND SCIENCE

VIJAYAWADA – 10

An Autonomous college under the jurisdiction of Krishna University, Machilipatnam.

(A college with potential for excellence; Re – accredited at the level 'A+' by NAAC, Bangalore)

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the project entitled **"STUDENT RECORD MANAGEMENT SYSTEM – SRMS",** is bonafied work of **K. MOUNIKA (197210), A. KRISHNA DEEPTHI (197265), M. VIJAYA LAKSHMI TIRUPATAMMA (197266), B. DEVIKA (197270)** submitted in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF SCIENCE (B.S.C), DATA SCIENCE** from P. B. Siddhartha College of Arts and Science, Vijayawada.

K. SudheerM.Sc (IT), M.Tech

Internal Guide

K. SridharM.Sc, M.Tech

Head of the Department

Department of Computer Science

ACKNOWLEDGEMENT

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

Our wholehearted thanks to Principal **Dr. M. Ramesh,** PB Siddhartha College of Arts and Science for his encouragement and giving me the opportunity to work out this project

I express my deep gratitude and regards to **Mr. K. Sridhar, M.Sc., M. Tech,** Head of the Department of Computer Science, and project guide, **Mr. K. Sudheer**, Lecturer, Department in Computer Science for their encouragement and valuable guidance in bringing shape to this dissertation

I am thankful to all the **Technical Staff and Non-teaching staff** in the department for their teachings and academic support.

K. MOUNIKA (197210)

A. KRISHNA DEEPTHI (197265)

M. VIJAYA LAKSHMI (197266)

B. DEVIKA (197270)

Contents

ABSTRACT 7

1. INTRODUCTION 7

1.1 PROJECT INTRODUCTION 7

1.2 WHAT IS DATABASE 8

1.3 DATABASE MANAGEMENT SYSTEM 8

2. EXISTING SYSTEM 11

2.1 INTRODUCTION 11

2.2 DEMERITS OF EXISTING SYSTEM 12

3. PROPOSED SYSTEM 13

3.1 INTRODUCTION 13

3.2 MERITS 14

4. SOFTWARE AND HARDWARE REQUIREMENTS 15

4.1 SOFTWARE REQUIREMENTS 15

4.1.1 DJANGO 15

4.1.2 PYTHON 16

4.1.3 HTML 17

4.1.4 CSS 18

4.1.5 JAVASCRIPT 19

4.1.6 BOOTSTRAP 20

4.2 HARDWARE REQUIREMENTS 20

5. DATA DICTIONARY 21

5.1 MODEL - STUDENT 21

5.2 MODEL – TEACHER 22

5.3 MODEL – DEPARTMENT 23

5.4 MODEL – STUDENT ATTENDACE 23

5.5 MODEL – COURSES 23

5.6 MODEL – STUDENT MARKS 24

5.7 MODEL – TECAHER COURSES 24

5.8 MODEL –GROUPS 25

6. REQUIREMENTS SPECIFICATIONS 26

6.1 SRS DOCUMENT 26

6.1.1 INTRODUCTION 26

6.1.2 INTENDED AUDIENCE 26

6.1.3 INTENDED USE 26
6.1.4 SCOPE 27
6.1.4.1 SCOPE DESCRIPTION 27
6.1.4.2 ACCEPTANCE CRITERIA 27
6.1.4.3 EXCLUSIONS 27
6.1.4.4 ASSUMPTIONS 27
6.1.4.5 USER NEEDS 27
6.1.5 FUNCTIONAL REQUIREMENTS 28
6.1.6 NON- FUNCTIONAL REQUIREMENTS 28
7. SYSTEM DESIGN AND DIAGRAMS 29
7.1 USECASE DIAGRAM 29
7.2 SEQUENCE DIAGRAM 30
7.2.1 SEQUENCE DIAGRAM FOR LOGIN 30
....30
7.2.2 SEQUENCE DIAGRAM FOR USERS 31
7.3 ACTIVITY DIAGRAM 32
....32
7.4 CLASS DIAGRAM 33
7.5 COLLABOURATION DIAGRAMS 34
8. SAMPLE SOURCE CODE 36
8.1 VIEWS.PY 36
HTML 69
9. SCREENSHOTS 83
10. TECHNOLOGIES USED 95
10.1 TOOLS 95
10.1.1 SUBLIME TEXT 95
10.1.2 GIT HUB 95
10.1.3 ADOBE XD 95
10.2 TECHNOLOGIES FOR FRONT-END 95
10.3 TECHNOLOGIES FOR BACK-END 95
11. TESTING 96
11.1 TYPES OF TESTING 96
11.2 TEST CASES 97
11.2.1 MOUDULE FACULTY 97
11.2.2 MODULE TEACHER 98

11.2.3 MODULE STUDENT.....100
12. CONCLUSION.....101
13. BIBILIOGRAPHY.....103

STUDENT RECORD MANAGEMENT SYSTEM

ABSTRACT

The motive of our project “STUDENT RECORD MANAGEMENT SYSTEM” is to alter the hand-operated student record system with the computerized program, which enhances the convenience of maintaining student information. The main functionalities comprise of inserting and manipulating the student details along with retrieving the data no matter when. This system also supports in securely storing the non-redundant data for great lengths of time. Student record system assists the user in quick management of the information with a user-friendly interface provided.

A great number of institutes and organisations are adopting these, instead of pen and paper work systems, for the betterment of organised records and command over the data. However, the system developed, mainly focuses on attaining the requirements of our college.

1. INTRODUCTION

1.1 PROJECT INTRODUCTION

Student data are regularly considered as office work produced for the college bureaucracy. However, a well-designed database application, whether or not the use of paper files or automatic structures, yields many benefits. A benefit, specifically with automatic applications, is performance in processing and changing student data in the college. Student record management, thus, play a key function withinside the normal functioning of the college. However greater importantly, they boom a faculty's cap potential to fulfil the wishes of students. The capabilities are encompass producing reports, adding, deleting, converting data, and carrying out analyses. To enhance the performance and value of data, many schools, institutions, and organisations have entered their student data into automatic databases. Automated applications the use of to be had pc era provide excellent blessings over conventional paper work. A large number of institutes and organizations rely on DATABASE-supported software instead of pencil-and-paper systems. The main objective is to improve organized records and data control. Student data and records in an educational institution are no exception. The idea of the student record management system is to keep student records safe. These records include students marks in internal and external examination along with assignment and attendance marks, the record of attendance of each and every student is also maintained. With every record available the reports are generated easily without any additional work

1.2 WHAT IS DATABASE

What is data? In summary, the data will be facts associated with any considered object. as an example, your name, age, height, weight, etc. it's all data associated with you. Images, images, files, pdfs, etc. It also can be counted as data.

what's a database? Database may be a collection of system data. They support electronic storage and manipulation of information. The database makes data management easy. Let's observe an example of a database: a web phone book uses a database to store data about people, phone numbers, and other contact information. Your electricity service provider uses a database to manage billing, customer-related issues, handle failure data, etc. also consider Facebook. you would like to store, manipulate and present data associated with members, their friends, member activities, messages, advertisements, etc. we will provide countless samples of using databases.

1.3 DATABASE MANAGEMENT SYSTEM

Database Management System Database management system is software used to manage databases. For example: MySQL, Oracle, etc. They are a very popular commercial database used in different applications. DBMS provides an interface to perform various operations, such as creating a database, storing data in it, updating data, creating tables in the database, and so on. provides protection and security for the database. For multiple users, it also maintains data consistency. The DBMS allows users to perform the subsequent tasks: Data Definition-used to make, modify, and delete definitions that outline the info organization within the database. Data update: accustomed insert, modify and delete the particular data within the database. Data Retrieval-Used to retrieve data from the database which will be utilized by applications for various purposes. User Management-used to record and monitor users, maintain data integrity, strengthen data security, process concurrency control, monitor performance, and restore information damaged by accidental failures.

It uses a digital repository established on a server to store and manage the data. It can provide a transparent and logical view of the method that manipulates data. DBMS contains automatic backup and recovery procedures. It contains ACID properties which maintain data in a very healthy state just in case of failure. It can reduce the complex relationship between data. It is accustomed support manipulation and processing of information. It is wont to provide security of information. It can view the database from different viewpoints per the necessities of the user.

DBMS allows the following: Controls database redundancy: It can control data redundancy because it stores all the information in one single database file which recorded data is placed within the database.

Data sharing: In DBMS, the authorized users of a company can share the info among multiple users. Easily Maintenance: It is easily maintainable because of the centralized nature of the database system. Reduce time: It reduces development time and maintenance need. Backup: It provides backup and recovery subsystems which create automatic backup of information from hardware and software failures and restores the info if required. multiple user interface: It provides differing kinds of user interfaces like graphical user interfaces, program interfaces.

There are one of a kind fields wherein a database control machine is applied. Following are some programs which make use of the data base management framework – Railway Reservation System – In the rail course reservation framework, the data base is wanted to keep the document or data of price price tag appointments, reputation approximately train's appearance, and flight. Additionally, if trains get late, people grow to be familiar with it thru the data base update. Library Management System – There are hundreds of books withinside the library so; it's miles hard to keep the document of the relative multitude of books in a check in or duplicate. Along those lines, the statistics set management framework (DBMS) is applied to preserve up all of the statistics diagnosed with the call of the book, problem date, accessibility of the book, and its writer. Banking – Database the executive's framework is applied to keep the trade statistics of the consumer withinside the data base. Education Sector – Presently, exams are led on line via way of means of severe faculties and colleges. They address all evaluation data thru the statistics set management framework (DBMS). In spite of that understudy's enlistments subtleties, grades, courses, expense, participation, results, and so on all of the statistics is placed away withinside the data base. Credit card exchanges – The database Management framework is applied for getting on price playing cards and age of month to month proclamations. Social Media Sites – We all usage of on line media web sites to companion with partners and to impart our views to the world. Every day, many human being's institution pursues those on line media bills like Pinterest, Facebook, Twitter, and Google further to. By the usage of the statistics set management framework, all of the statistics of customers are placed away withinside the data base and, we grow to be geared up to interface with others. Broadcast communications – Without DBMS any media transmission organisation can't think. The Database the executive's framework is essential for those agencies to keep the decision subtleties and month to month post-paid payments withinside the data base. Account – The data base management framework is applied for placing away statistics approximately deals, preserving and acquisition of financial instruments, for example, shares and bonds in a statistic set. Online Shopping – These days, internet-primarily based totally purchasing has grown to be a main pattern. Nobody wishes to go to the store and burn thru their time. Everybody wishes to keep thru internet primarily based totally purchasing web sites, (for example,

Amazon, Flipkart, Snapdeal) from home. So, all of the gadgets are offered and brought uniquely with the help of the data base management framework (DBMS). Receipt charges, instalments, purchase statistics those are completed with the help of DBMS. Human Resource Management – Big corporations or agencies have severe experts or representatives operating below them. They keep statistics approximately worker's compensation, evaluation, and paintings with the help of a data base management framework (DBMS). Manufacturing – Manufacturing agencies make numerous forms of gadgets and deal them consistently. To preserve the statistics approximately their gadgets like payments, acquisition of the item, amount, stock community the executives, data base management framework (DBMS) is applied. Airline Reservation System – This framework is equal to the railroad reservation framework. This framework moreover makes use of a data base management framework to keep the facts of flight take off, appearance, and defer reputation.

2. EXISTING SYSTEM

2.1 INTRODUCTION

Currently every single student record is maintained manually. These pencil and paper records are a lot of work. In the case of non-teaching staff, they collect the data of the person who wants to join the institute or organization and stack them in separate files. These implies a great importance for student's privacy. The records are student's personal data consisting of his/her name, personal details, address, contact details, educational qualifications and other information as well as the tenth and intermediate marks memos with the confirmation of transfer certificate and study of conduct. When a student record is needed right away, it takes a long time to find and retrieve it from the stack of files made up of records from all the other students. These files also need a lot of storage space and someone always has to be responsible for their arrangement and order.

While teachers have to mark attendance every day. So, they keep a record consisting of their names put together with the roll numbers. If a student is absent on a specific day, they will be marked as absent. At the end of the semester, the teacher calculates the total number of present days for each student in the class along with the total number of working days allotted for the class. This information is used to identify the student's presence in that particular semester. With a net attendance of less than 75%, zero points are awarded and you are not permitted to take the end semester examinations. If there is more than 75% and less than 80% attendance, two points are awarded. The student receives three points with an attendance rate of more than 80% and less than 85%. If the attendance is more than 85% and less than 90%, the student is awarded four points. The student with an attendance rate of more than 90% receives five points. The attendance grades assigned to each student along with the grades for the assignment are added to their overall internal grades. Given this importance, these attendance records must be collected very carefully and there must be no loss of the records as that have a major impact on the overall grades.

Even with the case of semester examination marks, the records need to be maintained. The examinations conducted are the following: first internal, second internal and semester examinations. The two internals are conducted for 30 marks each and their average is taken into account for 15 marks. These marks are summed up together with attendance and assignment marks. the scholar must score a minimum of 12 marks in both of the internal examination is to be considered as passed or qualified. similarly, the marks of the semester examination also are collected. The qualifying marks for these exams are 35 and if any particular student is absent for the exam, they need to retake the exam along with the students who doesn't seem to be qualified in the tests conducted. The web marks are calculated by the

teaching faculty and allotted to the student by the end of the semester. Most of the faculty uses Excel and other tools to do the process of calculation for attendance marks and examination marks, the faculty has to enter the data which is recorded in the attendance books and other records of the students. This whole process makes things difficulty for the faculty. Whereas, the student's attendance percentages are informed to them via SMS only if their attendance is less than 75%, till then the student has no idea of his presence records.

2.2 DEMERITS OF EXISTING SYSTEM

- A lot of pen and paper work is required, not User-Friendly
- Time Consuming and difficult for report generating.
- Proper Storage Facility is needed.
- The Data is not Accessible Constantly
- Security of the Records.

3. PROPOSED SYSTEM

3.1 INTRODUCTION

The project "Student record management system" is a web-software evolved for the right accessibility of records within no time. It is supplied with the functionalities of inserting, manipulating and retrieving the facts of the student. This application additionally helps in securely storing the constant statistics for greater lengths of time. With the help of a student data management system, it becomes very convenient to provide better management to your facilities as it covers all departments and connects them on a single platform. As mentioned above, it reduces manual data entry when retrieving. upload the data from one entry to the other and in the extended student information system you can create multiple reports with this data, without any effort. Simplify general administrative processes and carry out digital management.

The student record management system helps non-teaching faculty users to easily access student records. Since all data is stored virtually, no space is required to save the recordings. All details are insured as only non-teaching teachers can. Access to student information with a specific valid username and password.

The faculty can note the student's presence in the application, they just have to select the date and enter the attendance, when the attendance is submitted, the program automatically calculates the progress of the student's attendance. The teacher can view a clear report of total class attendance along with the information on the number of students with less than 75% attendance and the total number of working days. At the end of the semester, the teacher can assign marks with just one click. The application also helps with the faculty for teaching with the assignment of marks in internal and semester examinations. Reports are generated for each examination on the basis of the student's performance in that respective topic. Teachers receive a separate assessment to carry out all these tasks with their respective credentials. Students also benefit from the application. The marks assigned by the teacher are shown in their respective profile. The student can also track their attendance progress for that particular class in that particular semester. Students can also view their documents that have been sent to the non-teaching faculty at the beginning of the admission process.

When you implement student management software it allows you to cut back your expenses and increase your efficiency at the identical time. There are such a big amount of benefits of having the ability to take care of various services at a very low cost, reducing the quantity of individuals employed

to manage the institute, automating the complete operation of administration, etc. this suggests that as an academic institution you get plenty of freedom from manual work.

3.2 MERITS

- User-friendly.
- Quick management of work
- Proper Accessibility of data.
- Reports are generated easily.
- Secured data.

4. SOFTWARE AND HARDWARE REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

4.1.1 DJANGO

Django may be a high-level Python web framework that permits rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the effort of web development, so you'll be able to target writing your app while not having to reinvent the wheel. it's free and open source, features a thriving and active community, great documentation, and lots of options at no cost and paid-for support.

Django follows the "Batteries included" philosophy and provides almost everything developers might want to try and do "out of the box". Because everything you would like is an element of the one "product", it all works seamlessly together, follows consistent design principles Django are often (and has been) accustomed build almost any kind of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and might deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). the positioning you're currently reading is constructed with Django!

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the correct things" to guard the web site automatically. as an example, Django provides a secure thanks to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it's vulnerable (instead cookies just contain a key, and also the actual data is stored within the database) or directly storing passwords instead of a password hash. A password hash may be a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However, because of the "one-way" nature of the function, whether or not a stored hash value is compromised it's hard for an attacker to figure out the initial password.

Django uses a component-based “shared-nothing” architecture (each a part of the architecture is independent of the others, and may hence get replaced or changed if needed). Having a transparent separation between the various parts implies that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. a number of the busiest sites have successfully scaled Django to fulfil their demands (e.g. Instagram).

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. particularly, it makes use of the Don't Repeat Yourself (DRY) principle so there's no unnecessary duplication, reducing the number of codes. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

A Django model is that the built-in feature that Django uses to form tables, their fields, and various constraints. In short, Django Models is that the SQL of Database one uses with Django. SQL (Structured Query Language) is complex and involves plenty of various queries for creating, deleting, updating or the other stuff associated with database. Django models simplify the tasks and organize tables into models. Generally, each model maps to one database table. This article revolves about how one can use Django models to store data within the database conveniently. Moreover, we are able to use admin panel of Django to form, update, delete or retrieve fields of a model and various similar operation. Django models provide simplicity, consistency, version control and advanced metadata handling. Basics of a model include – Each model may be a Python class that subclasses django.db.models.Model. Each attribute of the model represents a database field. With all of this, Django gives you an automatically-generated database-access API Django Views are one in all the vital participants of MVT Structure of Django. As per Django Documentation, A view function could be a Python function that takes an online request and returns an online response. This response is the HTML contents of an online page, or a redirect, or a 404 error, or an XML document, or a picture, anything that an internet browser can display. Django views are a part of the computer programme — they sometimes render the HTML/CSS/JavaScript in your Template files into what you see in your browser after you render an online page. (Note that if you've used other frameworks supported the MVC (Model-View-Controller), don't get confused between Django views and views within the MVC paradigm. Django views roughly correspond to controllers in MVC, and Django templates to views in MVC.)

4.1.2 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level inbuilt data structures, combined with dynamic typing and dynamic binding, make it very

attractive for Rapid Application Development, additionally as to be used as a scripting or glue language to attach existing components together. Python's simple, easy to be told syntax emphasizes readability and thus reduces the price of program maintenance. Python supports modules and packages, which inspires program modularity and code reuse. The Python interpreter and therefore the extensive standard library are available in source or binary form for gratis for all major platforms, and may be freely distributed.

Often, programmers fall taken with Python thanks to the increased productivity it provides. Since there's no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers a slip, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the opposite hand, often the quickest thanks to debug a program is to feature some print statements to the source: the fast edit-test-debug cycle makes this easy approach very effective.

Python is simple to find out we predict this is often the foremost important reason why Python could be a great choice for research. we have seen many researchers with no Python experience gain fluency in a very matter of 1 or two months and successfully build video game experiments. For our customers, the globe of 3D graphics and real-time computer game environments is suddenly cracked up and prepared to be used for research. It gets even more exciting when our customers see how easy Python makes it to gather data from the sensors, reserve it to files, so use Python libraries like NumPy and matplotlib to feature a knowledge analysis and visualization pipeline.

4.1.3 HTML

HTML stands for Hypertext language. it's wont to design websites employing a language. HTML is that the combination of Hypertext and terminology. Hypertext defines the link between the online pages. A nomenclature is employed to define the text document within tag which defines the structure of web content. This language is employed to annotate (make notes for the computer) text in order that a machine can realize it and manipulate text accordingly. Most mark-up languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation should be done on the text. HTML may be a nomenclature utilized by the browser to govern text, images, and other content, so as to

display it within the required format. HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the primary standard version was HTML 2.0, published in 1999.

HTML (Hypertext Mark-up Language) is that the code that's accustomed structure an internet page and its content. as an example, content may be structured within a group of paragraphs, an inventory of bulleted points, or using images and data tables. because the title suggests, this text will provide you with a basic understanding of HTML and its functions.

HTML may be a nomenclature that defines the structure of your content. HTML consists of a series of elements, which you utilize to surround, or wrap, different parts of the content to form it appear a particular way, or act a specific way. The enclosing tags can make a word or image hyperlink to someplace else, can italicize words, can make the font bigger or smaller, and so on. as an example, take the subsequent line of content:

The opening tag: This consists of the name of the element (in this case, p), wrapped in opening and shutting angle brackets. This states where the element begins or starts to require effect — during this case where the paragraph begins. The closing tag: this can be the identical because the opening tag, except that it includes a forward slash before the element name. This states where the element ends — during this case where the paragraph ends. Failing to feature a closing tag is one among the quality beginner errors and might result in strange results. The content: this can be the content of the element, which during this case, is simply text. The element: The opening tag, the closing tag, and also the content together comprises the element.

Attributes contain extra information about the element that you simply don't desire to look within the actual content. Here, class is that the attribute name and editor-note is that the attribute value. The category attribute allows you to allow the element a non-unique identifier that may be accustomed target it (and the other elements with the identical class value) with style information and other things. An attribute should have the following: A space between it and therefore the element name (or the previous attribute, if the element already has one or more attributes). The attribute name followed by an equal sign. The attribute value wrapped by opening and shutting quotation marks.

4.1.4 CSS

Cascading Style Sheets, fondly brought up as CSS, could be a simply designed language intended to simplify the method of creating sites presentable. CSS allows you to use styles to web content. More importantly, CSS enables you to try and do this independent of the HTML that produces up each online

page. CSS is simple to find out and understood, but it provides powerful control over the presentation of an HTML document.

CSS saves time: you'll write CSS once and reuse the identical sheet in multiple HTML pages. Easy Maintenance: to create a world change simply change the fashion, and every one element all told the webpages are going to be updated automatically. Search Engines: CSS is taken into account a clean coding technique, which suggests search engines won't need to struggle to "read" its content. Superior styles to HTML: CSS encompasses a much wider array of attributes than HTML, so you'll be able to provides a much better look to your HTML page compared to HTML attributes. Offline Browsing: CSS can store web applications locally with the assistance of an offline cache. Using this we will view offline websites.

CSS are often used for very basic document text styling — as an example changing the colour and size of headings and links. It is accustomed create layout — for instance turning one column of text into a layout with a main content area and a sidebar for related information. It can even be used for effects like animation. Have a glance at the links during this paragraph for specific examples.

4.1.5 JAVASCRIPT

JavaScript (often shortened to JS) may be a lightweight, interpreted, object-oriented language with first-class functions, and is best referred to as the scripting language for sites, but it's employed in many non-browser environments still. it's a prototype-based, multi-paradigm scripting language that's dynamic, and supports object-oriented, imperative, and functional programming styles. JavaScript runs on the client side of the online, which might be accustomed design / program how the online pages behave on the occurrence of an occasion. JavaScript is a simple to be told and also powerful scripting language, widely used for controlling web content behaviour.

Contrary to popular misconception, JavaScript isn't "Interpreted Java". during a nutshell, JavaScript could be a dynamic scripting language supporting prototype-based object construction. the essential syntax is intentionally kind of like both Java and C++ to cut back the number of recent concepts required to be told the language. Language constructs, like if statements, for and while loops, and switch and check out ... catch blocks function the identical as in these languages (or nearly so). JavaScript can function as both a procedural and an object-oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as against the syntactic class definitions common in compiled languages like C++ and Java. Once an object has been constructed it is used as a blueprint (or prototype) for creating similar objects.

JavaScript could be a lightweight, cross-platform, and interpreted scripting language. it's well-known for the event of sites, many non-browser environments also use it. JavaScript are often used for Client-side developments further as Server-side developments. JavaScript contains a regular library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements.

4.1.6 BOOTSTRAP

Bootstrap could be a free and open-source tool collection for creating responsive websites and web applications. it's the foremost popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. It solves many problems which we had once, one in every of which is that the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All due to Bootstrap developers -Mark Otto and Jacob Thornton of Twitter, though it absolutely was later declared to be an open-source project. Faster and Easier Web Development. It creates Platform-independent web content. It creates Responsive Web-pages. It designed to be alert to mobile devices too. It is Free! Available on www.getbootstrap.com How to use Bootstrap 4 during a webpage: There are two ways to incorporate Bootstrap on the web site. Include Bootstrap from the CDN link. Download Bootstrap from getbootstrap.com and use it. Bootstrap 4 from CDN: This method of putting in Bootstrap is simple. it's highly recommended to follow this method.

4.2 HARDWARE REQUIREMENTS

- 1 GB RAM.
- 1 CPU Core.
- 24 GB SSD Storage.
- 2 TB Transfer.
- 40 Gbit Network In.
- 125 Mbit Network Out.

5. DATA DICTIONARY

5.1 MODEL - STUDENT

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
First_Name	*Varchar*	*30*	*False*	*False*	*False*
Last_Name	*Varchar*	*30*	*False*	*False*	*False*
Class	*Varchar*	*10*	*False*	*False*	*False*
Group	*Varchar*	*10*	*False*	*False*	*False*
Section	*Varchar*	*1*	*False*	*False*	*False*
Admission_Number	*Integer*	*4*	*False*	*False*	***True***
Roll_number	*Integer*	*6*	***True***	*False*	***True***
Date_of_birth	*Date*	*None*	*False*	*False*	*False*
Gender	*Varchar*	*10*	*False*	*False*	*False*
Father_name	*Varchar*	*50*	*False*	*True*	*False*
Mother_name	*Varchar*	*50*	*False*	*True*	*False*
Guardian_name	*Varchar*	*50*	*False*	*True*	*False*
Religion	*Varchar*	*20*	*False*	*False*	*False*
Nationality	*Varchar*	*20*	*False*	*False*	*False*
Aadhaar_number	*Varchar*	*9*	*False*	*False*	***True***
Mobile_number	*Varchar*	*10*	*False*	*False*	***True***

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Alternate Mobile_number	*Varchar*	*10*	*False*	*True*	***True***

Email_id	*Varchar*	*256*	*False*	*False*	***True***
Door_number	*Varchar*	*10*	*False*	*False*	*False*
Premises	*Varchar*	*50*	*False*	*True*	*False*
Street	*Varchar*	*30*	*False*	*True*	*False*
City	*Varchar*	*30*	*False*	*False*	*False*
State	*Varchar*	*30*	*False*	*False*	*False*
Country	*Varchar*	*20*	*False*	*False*	*False*
Pin code	*Varchar*	*10*	*False*	*False*	*False*
Student_Image	*Image*	*Less than 300 kb*	*False*	*False*	***True***
Study_Of_Conduct	*Image*	*Less than 300 kb*	*False*	*False*	***True***
Transfer_Certificate	*Image*	*Less than 300 kb*	*False*	*False*	***True***
Tenth_Marks_Long _Memo	*Image*	*Less than 300 kb*	*False*	*False*	***True***
Intermediate_Marks _Short_Memo	*Image*	*Less than 300 kb*	*False*	*False*	***True***
Student_Signature	*Image*	*Less than 300 kb*	*False*	*False*	***True***

5.2 MODEL – TEACHER

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Name	*Varchar*	*30*	*False*	*False*	*False*
Teacher_id	*Integer*	*6*	***True***	*True*	***True***
Mobile_number	*Varchar*	*10*	*False*	*False*	***True***

Email_id	*Varchar*	*256*	*False*	*False*	***True***
Department_id	*Integer*	*4*	*False*	*False*	*False*

5.3 MODEL – DEPARTMENT

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Department_Id	*Varchar*	*7*	***True***	*False*	***True***
Department	*Varchar*	*10*	*False*	*False*	***True***

5.4 MODEL – STUDENT ATTENDACE

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Date_and_Time	*Datetime*	*None*	*False*	*False*	*False*
Subject	*Varchar*	*15*	*False*	*False*	*False*
Semester	*Integer*	*1*	*False*	*False*	*False*
Roll_Number	*Integer*	*6*	***False (Foreign Key)***	*False*	*False*
Teacher_id	*Integer*	*6*	***False (Foreign Key)***	*False*	*False*
Course_Code	*Varchar*	*10*	***False (Foreign Key***	*False*	*False*
Attendance	*Varcher*	*10*	*False*	*False*	*False*

5.5 MODEL – COURSES

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***

Course_code	*Varchar*	*7*	***True***	*False*	***True***
Subject	*Varchar*	*10*	*False*	*False*	*False*
Details	*Varchar*	*50*	*False*	*False*	***True***

5.6 MODEL – STUDENT MARKS

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Date_and_Time	*Datetime*	*None*	*False*	*False*	*False*
Semester	*Integer*	*1*	*False*	*False*	*False*
Course_code	*Varchar*	*7*	***False (Foreign Key)***	*False*	
Exam_Type	*Varchar*	*10*	*False*	*False*	*False*
Subject	*Varchar*	*10*	*False*	*False*	*False*
Roll_Number	*Integer*	*6*	***False (Foreign Key)***	*False*	*False*
Teacher_id	*Integer*	*6*	***False (Foreign Key)***	*False*	*False*
Marks	*Float*	*None*	*False*	***True***	*False*

5.7 MODEL – TECAHER COURSES

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Group	*Varchar*	10	***False***	*False*	*False*

			(Foreign Key)		
Teacher_Id	*Varchar*	6	***False*** ***(Foreign Key)***	*False*	*False*
Course_Code	*Varchar*	*7*	***False*** ***(Foreign Key)***	*False*	*False*
Semester_Teacher	*Varchar*	*1*	*False*	*False*	*False*
Year_Of_Admission	*Interger*	*10*	*False*	*False*	*False*

5.8 MODEL –GROUPS

Attributes	***Datatype***	***length***	***Primary key***	***nullable***	***Unique***
Group	*Varchar*	10	***True***	*False*	*True*
Class	*Varchar*	10	*False*	*False*	*False*
Groups_Type	*Varchar*	10	*False*	*False*	*False*

6. REQUIREMENTS SPECIFICATIONS

6.1 SRS DOCUMENT

6.1.1 INTRODUCTION

The "Student File Management System" project is web software that was developed for the correct accessibility of files in the shortest possible time. It is supplied with the functionality of inserting, editing and retrieving student data. This application also helps to safely store constant statistics. for a longer period of time. load the data from one input to the other and in the extended student information system you can easily create multiple reports with this data.

When you implement student management software it allows you to cut back your expenses and increase your efficiency at the identical time. There are such a big amount of benefits of having the ability to take care of various services at a very low cost, reducing the quantity of individuals employed to manage the institute, automating the complete operation of administration, etc. this suggests that as an academic institution you get plenty of freedom from manual work.

6.1.2 INTENDED AUDIENCE

- Non-Teaching Faculty
- Teaching Faculty
- Student

6.1.3 INTENDED USE

The main purpose is to maintain the data related to the students of the college in an organized and efficient way to retrieve and manipulate them.

The Non-Teaching faculty can have the access to manipulate the records of the students submitted during the time of admission process. The operation performed by the faculty includes adding a new student, updating a student and removing a student.

The Teaching faculty with a proper login details can put the attendance and submit the student’s marks. The application also assists in report generation and easy access of the results. The students can view their details, marks along with attendance allotted by the teacher.

6.1.4 SCOPE

6.1.4.1 SCOPE DESCRIPTION

The idea of Student record management system is to safely secure the student records. The main objective is for the betterment of organised records and command over the data.

6.1.4.2 ACCEPTANCE CRITERIA

- Inserting, viewing, updating and removing the records of students
- Every user has their individual access to the system
- The teacher cam marks the attendance and assign the marks of the students
- The students having the access of their respective info along with attendance and marks allotted

6.1.4.3 EXCLUSIONS

- The fees record of the student
- Any other activities apart from marks and attendance of the student are not included (part 4)

6.1.4.4 ASSUMPTIONS

- Teacher adding the details of new paper included in the curriculum.
- Student access to update their login credentials.
- The user can take print of documents
- Giving multiple login accesses to the faculty
- Admin approval for teacher registration

6.1.4.5 USER NEEDS

- Managing the records of the student which includes,

1) personal information
2) contact details
3) address info
4) Student image
5) 10^{th} and intermediate marks memo along with study of conduct and transfer certificate.

- The teaching faculty having the access to,

I. insert

II. update

III. remove

the marks scored by the student in internal and external examinations as well as Attendance and Assignment marks.

- Providing a clear report for the student, examination scores and number of attended days in total working days of the semester

6.1.5 FUNCTIONAL REQUIREMENTS

1) Users accessing the system with a proper valid login detail.
2) Submission of student details and certificates.
3) Submission of marks and attendance by the teacher.
4) Report generation for attendance and marks.
5) Student viewing the details, marks, attendance and report.
6) Teacher accessing and updating the details of marks and attendance
7) Non-teaching faculty accessing the details of student and certificates

6.1.6 NON- FUNCTIONAL REQUIREMENTS

A. Security

Secure school administration software often offers password protection for each module. Administrators can configure the software to vary usage access for different users who use the software. a teacher using the system may need access to various modules including attendance management, exam management, etc. This can be well supported by the use of a secure school administration system.

B. User-friendly

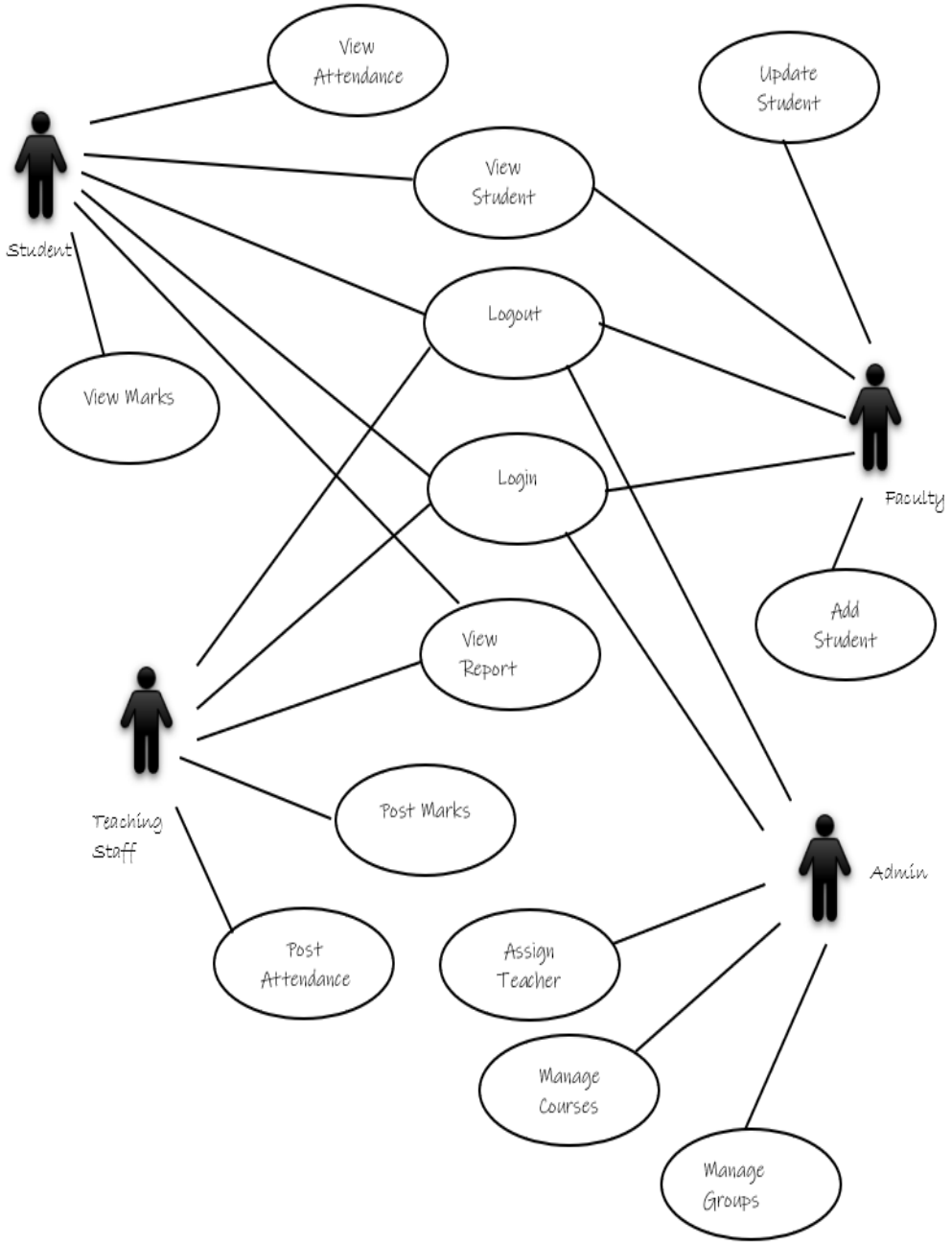
The application has a user-friendly interface which is built with the web development software that includes HTML, CSS, JavaScript etc. The user can retrieve the information needed anytime with one click. Report are automatically generated especially for the teaching faculty.

C. Manageability

D. Availability

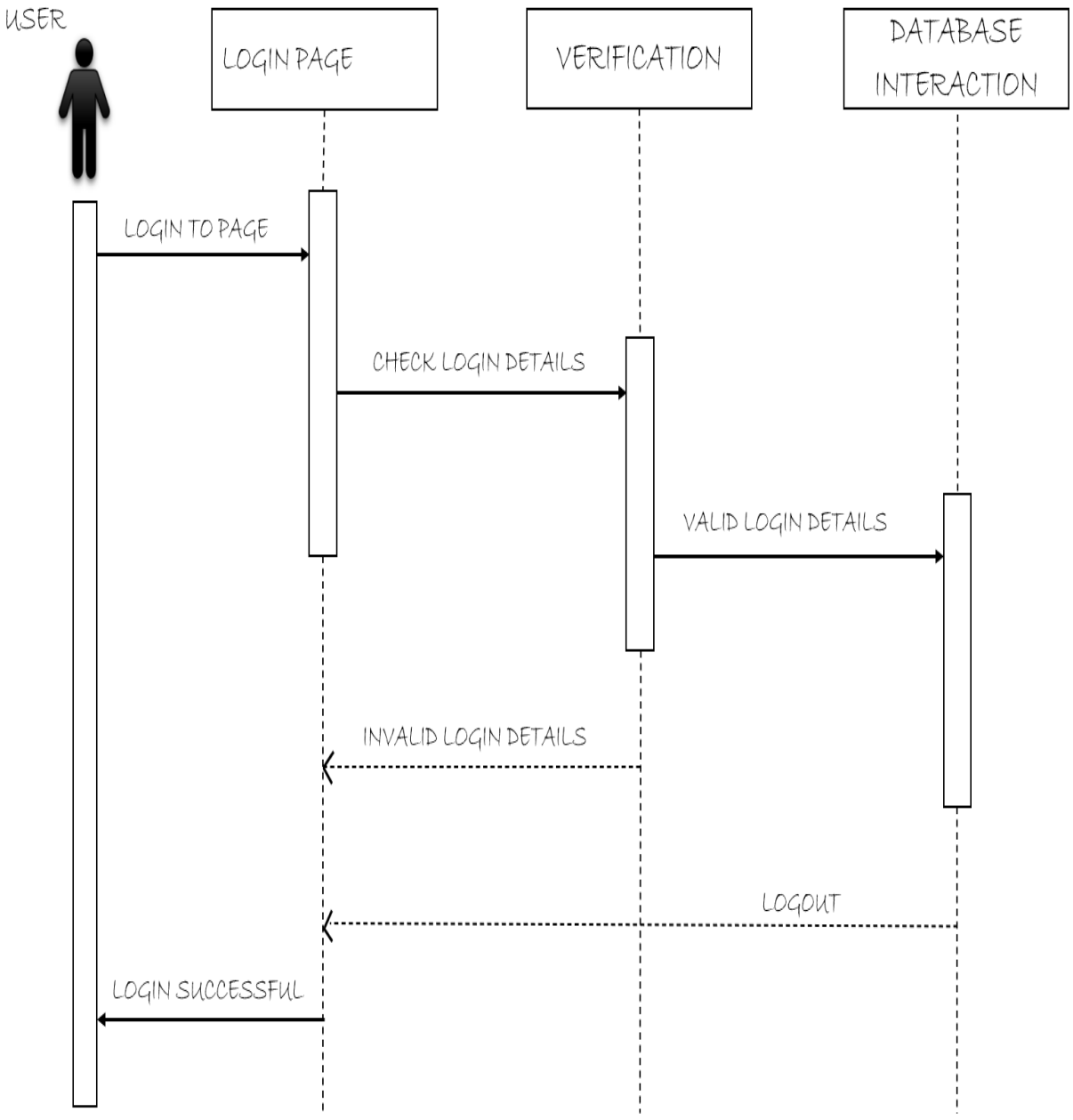
7. SYSTEM DESIGN AND DIAGRAMS

7.1 USECASE DIAGRAM

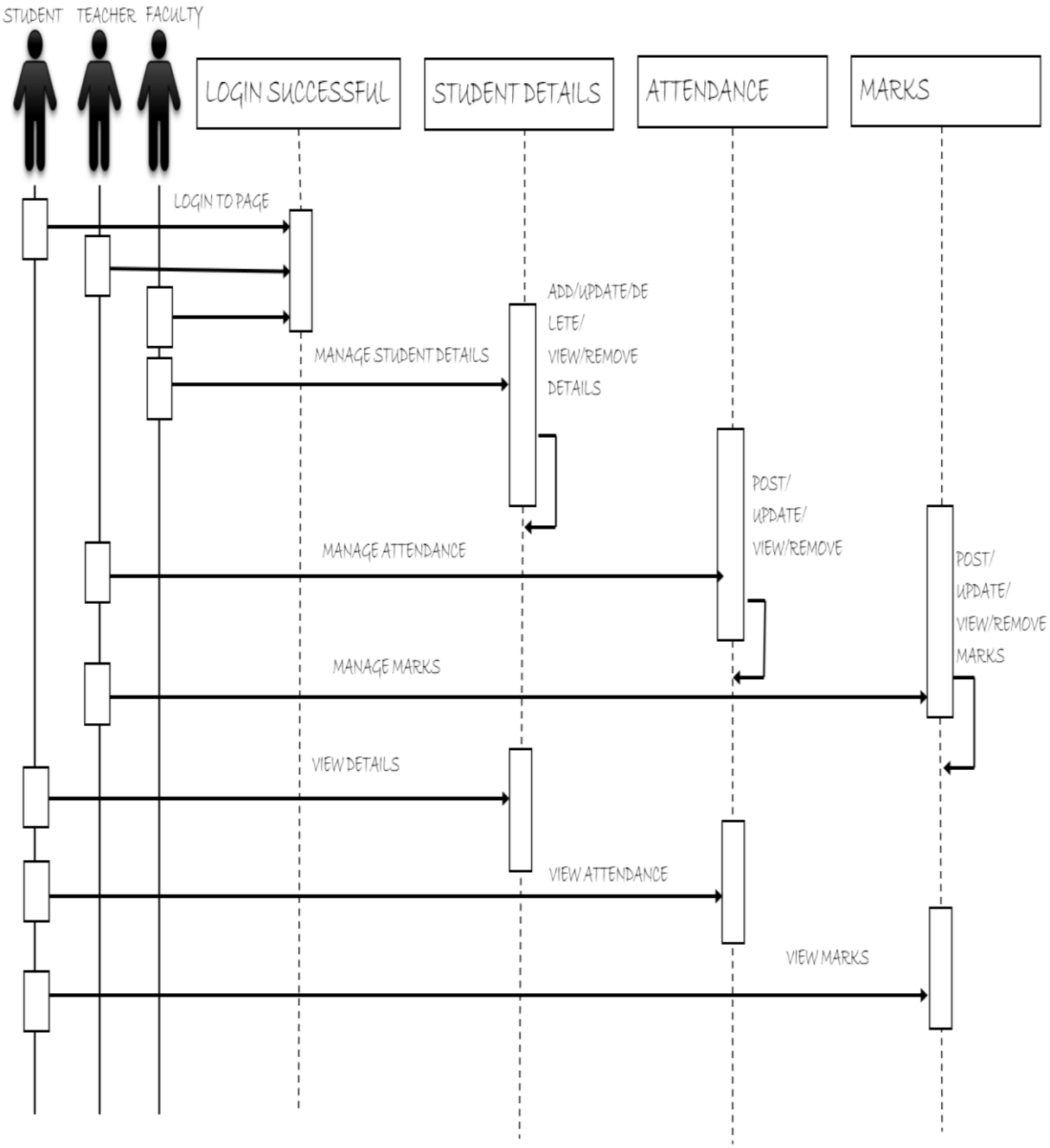


7.2 SEQUENCE DIAGRAM

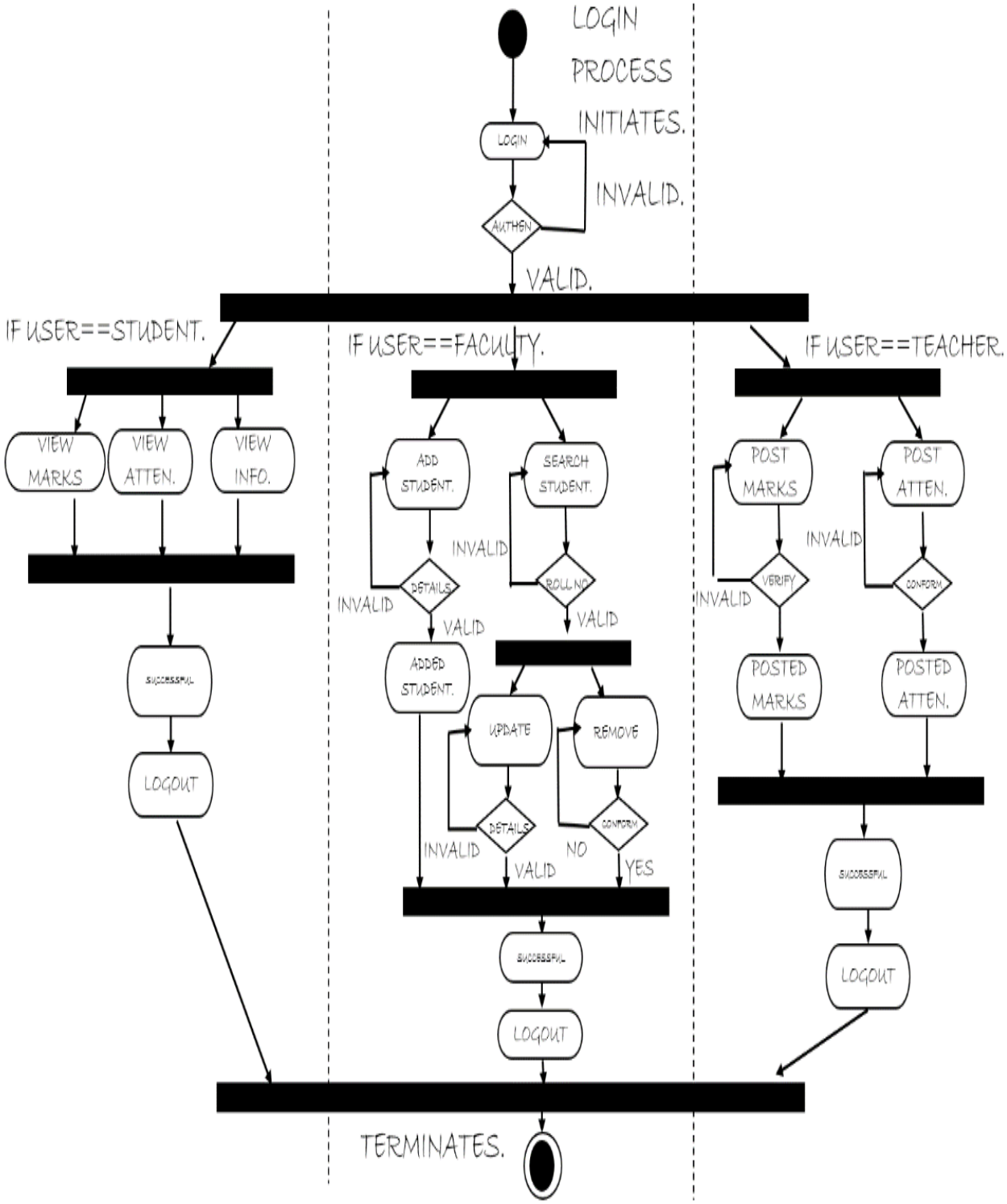
7.2.1 SEQUENCE DIAGRAM FOR LOGIN



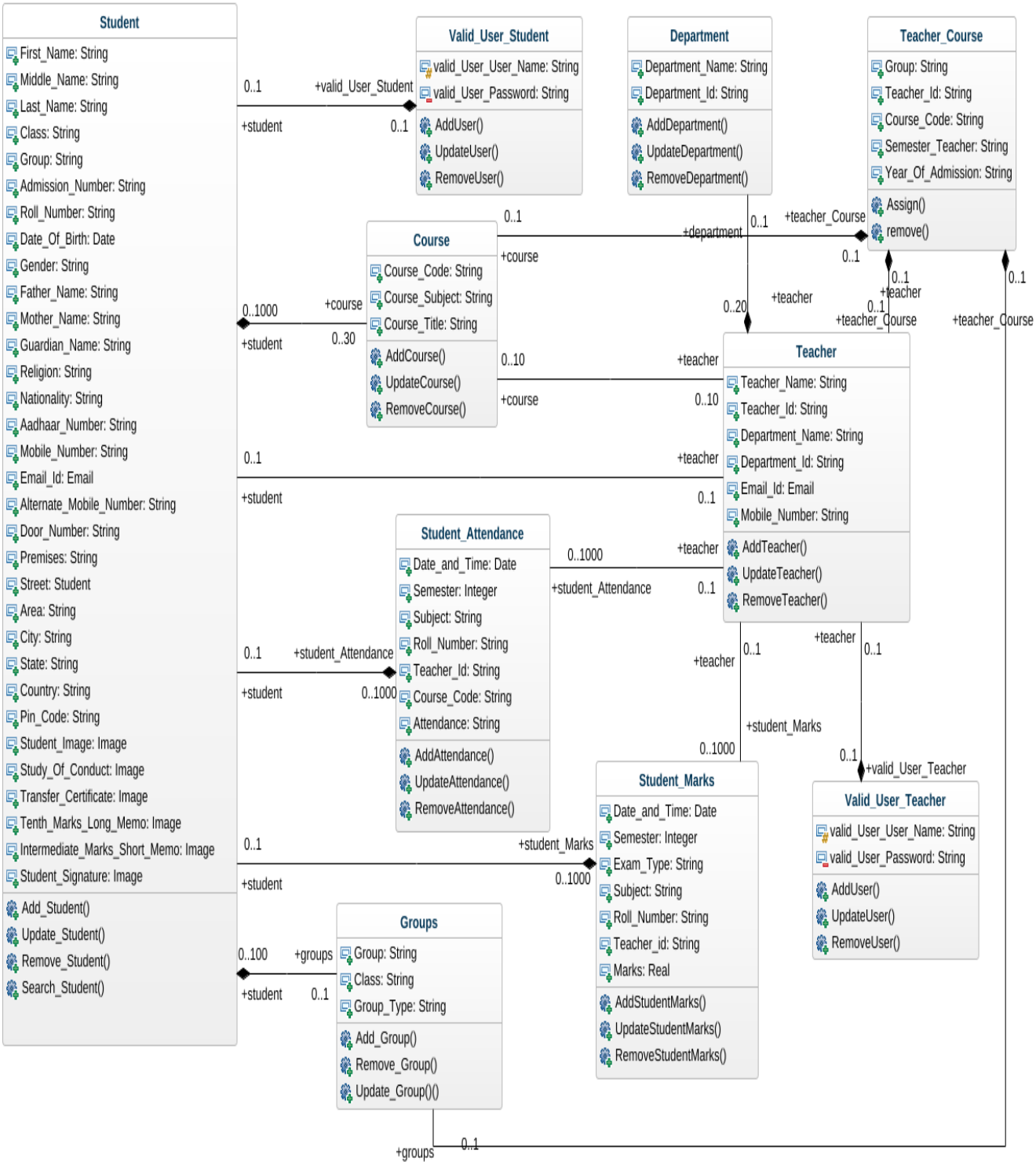
7.2.2 SEQUENCE DIAGRAM FOR USERS



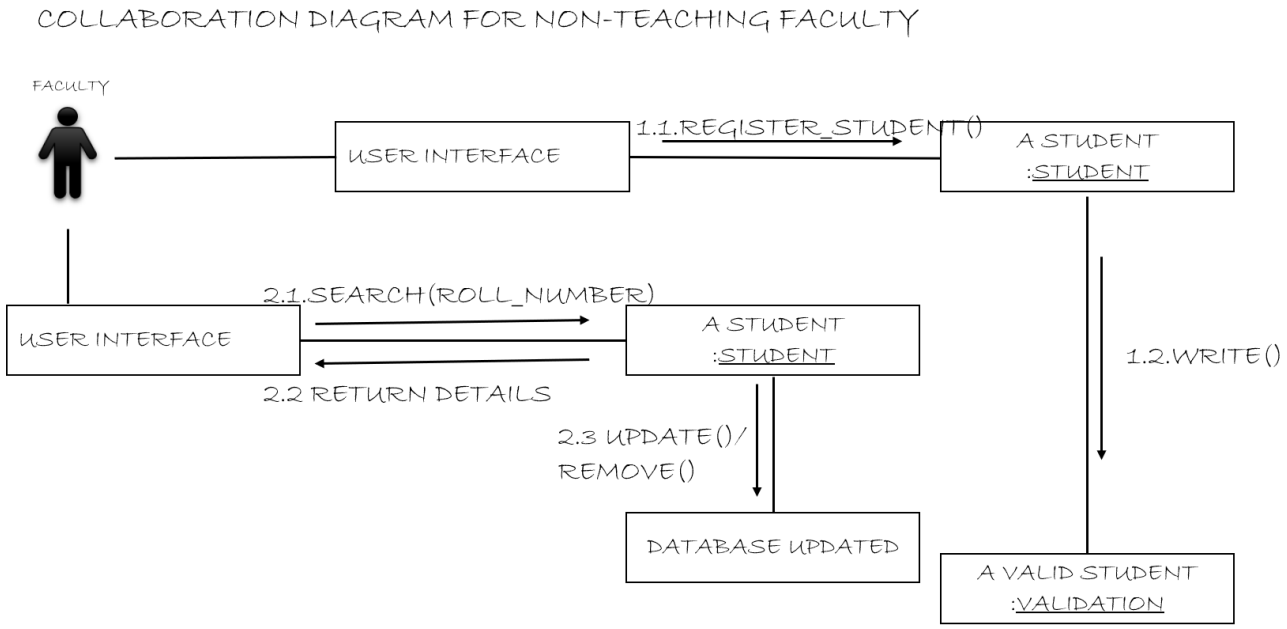
7.3 ACTIVITY DIAGRAM

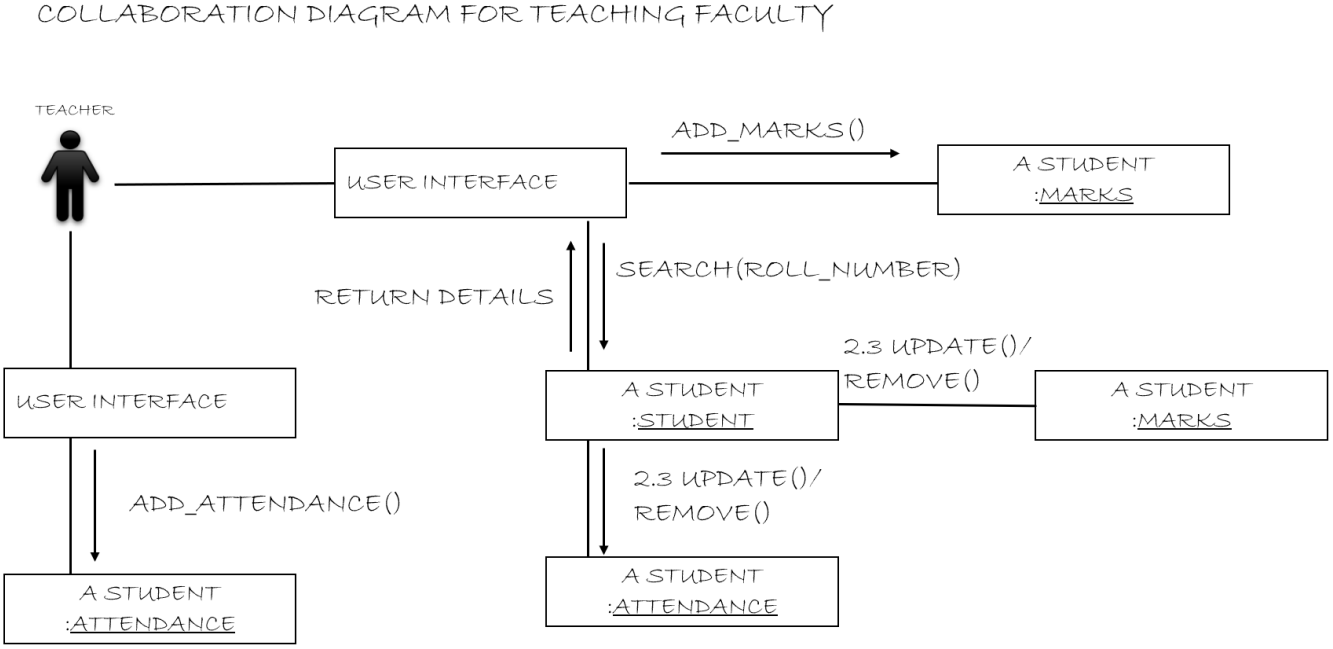


7.4 CLASS DIAGRAM

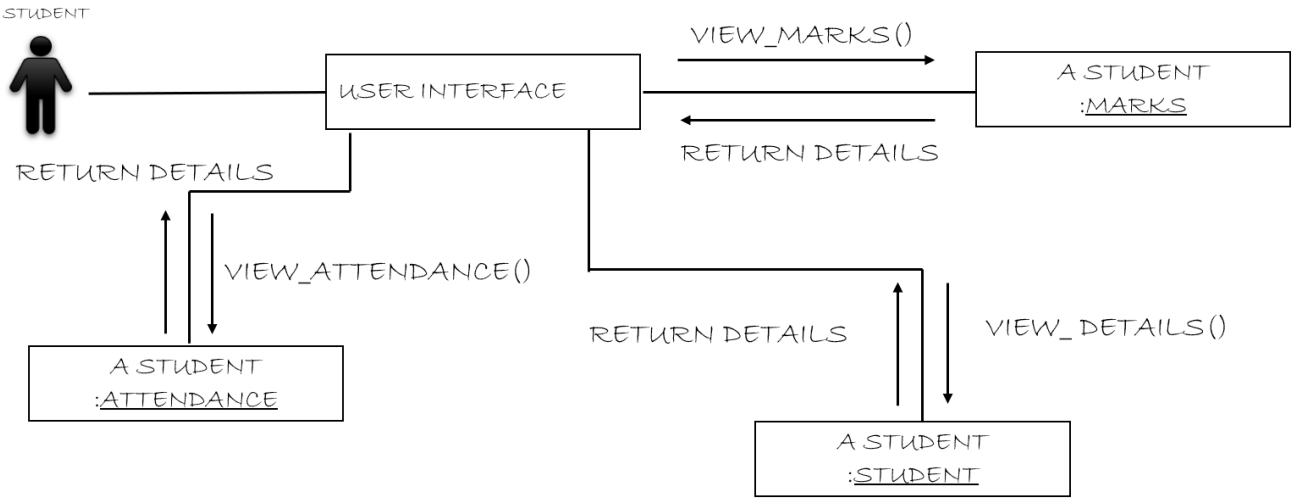


7.5 COLLABOURATION DIAGRAMS





COLLABORATION DIAGRAM FOR STUDENT



8. SAMPLE SOURCE CODE

8.1 VIEWS.PY

```
from django.shortcuts import render,get_object_or_404,HttpResponse
from student.models import *
from student.logic import *
from student.libraries import *
from student.modelforms import *
# Create your views here.

def Selection(request):
        return render(request,'temp/selection.html')

def Login_Faculty(request):
        return render(request,'temp/Facultylogin.html')

def Login_Teacher(request):
        return render(request,'temp/Teacherlogin.html')

def Login_Student(request):
        return render(request,'temp/Studentlogin.html')

def faculty_second(request,Year,Group_Given):
```

```
        Student_Filter_Year_Of_Admission = Student.objects.filter(Year_Of_Admission = Year, Group = Group_Given).order_by('-Group','Year_Of_Admission')
        content = {
                        'Student_Filter' : Student_Filter_Year_Of_Admission
        }
        return render(request, 'temp/class.html',content)

def faculty_third(request,Year,First_Name,Roll_Number):
        Student_Details = Student.objects.filter(Roll_Number = Roll_Number)
        content = {
                        'Student_Details':Student_Details
        }
        return render(request,'temp/student.html',content)

def Add_Student(request):
                form = Student_ModelForm()
                content = {
                                'form':form
                }
                return render(request,"temp/addstudent.html",content)

def Edit_Student(request,Year,Group,Roll_Number):
                student = get_object_or_404(Student, Roll_Number = Roll_Number)
                form = Student_ModelForm(instance  = student)
```

```
        details = Student.objects.filter(Roll_Number = Roll_Number)
        content = {
                'form':form,
                'details':details
        }
        return render(request,"temp/editstudent.html",content)

def Delete_Student(request,Year,Group,Roll_Number):
    Student_Details = Student.objects.get(Roll_Number = Roll_Number)
    Student_Details.delete()
    content = {
                'Student_Filter' : Student_Filter_Year_Of_Admission
    }
    return render (request, 'temp/class.html',content)

def Certificate_Student(request,Year,Group,Roll_Number):
    Student_Details = Student.objects.filter(Roll_Number = Roll_Number)
    content = {
            'Details': Student_Details
    }
    return render(request,'temp/certificates.html',content)

def Teacher_second(request,Id,Group,Year,Semester,Course):
```

```
    Class_Students = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
    content = {
                'Id' : Id,
                'class_student' : Class_Students,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Course
    }
    return render(request, 'temp/Teachersecond.html',content)


def Teacher_third(request,Id,Year,Group,Semester,Course):
    Rolls = Student.objects.filter(Year_Of_Admission = Year).filter(Group = Group).values_list('Roll_Number')
    Total_strength = len(Rolls)
    Dates = {}
    for p in Previous:
        Date = p.Date_Time_Original.date()
        Dates[Date] = len(Previous.filter(Date_Time_Original = p.Date_Time_Original).filter(Attendance_Date = 'PRESENT'))
    content = {
                'Semester' : Semester,
                'Course' : Course
```

```
    }
    return render(request, 'temp/Teacherthird.html',content)


def Teacher_fourth(request,Id,Year,Group,Semester,Course):
    if request.POST['SUBMIT'] == 'Mark':
        Attendance = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
        Roll =Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number').values_list('Roll_Number')
        content = {
                'Id' : Id,
                'Attendance':Attendance,
                'Roll':Rolls,
                }
        return render(request,'temp/Teacherfourth.html',content)
    elif request.POST['SUBMIT'] == 'Delete':
        values = list(request.POST.keys())
        for i in range(2,len(values)):
            if request.POST[values[i]] == 'on':
                Date = values[i]
                Month = Find_Month(Date[0:3])
        for i in Values:
            i.delete()
        content = {
```

```
                'Id' : Id,
                'T' : Total_strength,
                'Dates' : Dates,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Course
            }
        return render(request, 'temp/Teacherthird.html',content)
    elif request.POST['SUBMIT'] == 'Report':
        studentgraph = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
        for i in range(len(studentgraph)):
            datas.append(list(dataframe2.loc[i]))
        content = {
            'Id' : Id,
            'Year' : Year,
            'Group' : Group,
            'Semester' : Semester,
            'Course' : Course,
            'student' : studentgraph,
        }
        return render(request, 'temp/TeacherReport.html',content)
```

```
def Teacher_fifth(request,Id,Year,Group,Semester,Course,Date):
    Month = Find_Month(Date[0:3])
    Values = Student_Attendance.objects.filter(Teacher_Id_Attendance = Id).filter(Course_Code = Course).filter(Semester = Semester).filter(Date_Time_Original__date = datetime.date(int(Date[9:13]),Month,int(Date[5:7])))
    content = {
            'Date' : Date[5:7],
            'Month' : Date[0:3],
            'Semester' : Semester,
            'Course' : Course
    }
    return render(request, 'temp/Teacherfifth.html',content)


def Teacher_sixth(request,Id,Year,Group,Semester,Course):
    Rolls = Student.objects.filter(Year_Of_Admission = Year).filter(Group = Group).values_list('Roll_Number')
    Previous_Date = set()
    for key in Attendance:
        Key = Student.objects.get(Roll_Number = key)
        if request.POST.get('Date') not in Previous_Date:
            Instance = Student_Attendance(Date_Time_Original = request.POST.get('Date'),Semester = Semester,
            Attendance_Date = Attendance[key])
            Instance.save()
    content = {
```

```
                'Id' : Id_original,
                'T' : Total_strength,
                'Dates' : Dates,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Course_original
        }
        return render(request, 'temp/Teacherthird.html', content)


def Teacher_seventh(request,Id,Year,Group,Semester,Course,Date):
        Month = Find_Month(Date[0:3])
        value = list(request.POST)
        li = value[3:len(value)]
        Attendance = {}
        for i in range(0,len(li)):
                Attendance[li[i]] = request.POST.get(li[i])
        for i in range(len(Values)):
                if list(Attendance.keys())[i] == Values[i].Roll_Number_Attendance.Roll_Number and
Attendance[list(Attendance.keys())[i]] != Values[i].Attendance_Date:
                        instance = Values[i]
                        instance.Attendance_Date = Attendance[list(Attendance.keys())[i]]
                        instance.save()
        Previous_Date = Previous.values_list('Date_Time_Original')
```

```
    Total_strength = len(Rolls)
    Dates = {}
    for p in Previous:
        Date = p.Date_Time_Original.date()
        Dates[Date] = len(Previous.filter(Date_Time_Original = p.Date_Time_Original).filter(Attendance_Date = 'PRESENT'))
    content = {
            'Id' : Id,
            'T' : Total_strength,
            'Dates' : Dates,
    }
    return render(request, 'temp/Teacherthird.html',content)


def Teacher_eight(request,Id,Year,Group,Semester,Course):
    Student_List = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
    Types = {'I INTERNAL','II INTERNAL','ASSIGNMENT','SEMESTER'}
    content = {
            'Attendance' : Student_List,
            'Types' : Types,
            'Id' : Id,
            'Year' : Year,
            'Group' : Group,
            'Semester' : Semester,
```

```
                'Course' : Course
        }
        return render(request,'temp/Teachereight.html',content)


def Teacher_ninth(request,Id,Year,Group,Semester,Course):
        Rolls = Student.objects.filter(Year_Of_Admission = Year).filter(Group = Group).values_list('Roll_Number')
        for i in Internal_1:
                if i.Marks_Alloted == None or i.Marks_Alloted < 12:
                        Total_Fail += 1
                else:
                        Total_Pass += 1
        if len(Internal_1) != 0:
                content = {
                                'T_S' : len(Rolls),
                                'average': marks_details('I Internal',Internal_1),
                                'labels' :['Total Students Passed','Total Students Failed'],
                                'data' : [Total_Pass,Total_Fail],
                                'Attendance' : Internal_1,
                                'Absent' : Total_Absent,
                        }
                return render(request, 'temp/Teacherninehalf.html',content)
        else:
```

```
        Internal_2 = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
        content = {
                'Attendance' : Internal_2,
                'Id' : Id,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Course
        }
        return render(request, 'temp/Teacherninth.html',content)


def Teacher_tenth(request,Id,Year,Group,Semester,Course):
    Rolls = Student.objects.filter(Year_Of_Admission = Year).filter(Group = Group).values_list('Roll_Number')


    Total_Absent = len(Absent)
    Total_Present = Total_Strength - Total_Absent
    Total_Fail = 0
    Total_Pass = 0
    for i in Internal_2:
        if i.Marks_Alloted == None or i.Marks_Alloted < 12:
            Total_Fail += 1
        else:
```

```
                Total_Pass += 1
        if len(Internal_2) != 0:
                content = {
                        'T_S' : len(Rolls),
                        'average': marks_details('II Internal',Internal_2),
                        'labels' :['Total Students Passed','Total Students Failed'],
                        'data' : [Total_Pass,Total_Fail],
                        'Attendance' : Internal_2,
                        'Id' : Id,
                    }
                return render(request, 'temp/Teachertenhalf.html',content)
        else:
                Internal_2 = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
                content = {
                        'Attendance' : Internal_2,
                        'Id' : Id,
                        'Year' : Year,
                        'Group' : Group,
                        'Semester' : Semester,
                        'Course' : Course
                }
                return render(request, 'temp/Teachertenth.html',content)
```

```
def Teacher_eleven(request,Id,Year,Group,Semester,Course):
    if len(Assignment) != 0:
        content = {
                'Attendance' : Assignment,
                'Id' : Id,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Course
            }
        return render(request, 'temp/Teacherelevenhalf.html',content)
    else:
        Assignment = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
        content = {
                'Attendance' : Assignment,
                'Id' : Id,
                'Year' : Year,
        }
        return render(request, 'temp/Teachereleven.html',content)


def Teacher_twelve(request,Id,Year,Group,Semester,Course):
    Rolls = Student.objects.filter(Year_Of_Admission = Year).filter(Group = Group).values_list('Roll_Number')
```

```
        Semesters = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Marks_Type = 'Semester').filter(Roll_Number_Marks__in = Rolls)
        Total_Pass = 0
        for i in Semesters:
            if i.Marks_Alloted == None or i.Marks_Alloted < 35:
                Total_Fail += 1
            else:
                Total_Pass += 1
        if len(Semesters) != 0:
            content = {
                        'Year' : Year,
                        'Group' : Group,
                        'Semester' : Semester,
                        'Course' : Course
                    }
            return render(request, 'temp/Teachertwelvehalf.html',content)
        else:
            Semesters = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
            content = {
                        'Attendance' : Semesters,
                        'Id' : Id,
                        'Year' : Year,
                        'Group' : Group,
```

```
                'Semester' : Semester,
                'Course' : Course
        }
        return render(request, 'temp/Teachertwelve.html',content)


def Teacher_thirteen(request,Id,Year,Group,Semester,Course):
    Cours = Course
    Ids = Id
    A = request.POST
    B = list(A.keys())[3:len(A)]
    Course = Courses.objects.get(Course_Code = Course)
    Id = Teacher.objects.get(Teacher_Id = Id)
    Rolls = Student.objects.filter(Year_Of_Admission = Year).filter(Group = Group).values_list('Roll_Number')
    Previous_Internal_1 = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls)
        Previous_Internal_1 = Previous_Internal_1.filter(Marks_Type = 'I Internal')
        Previous_1 = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'I Internal')
        Total_Strength = len(Previous_1)
        Absent = Previous_1.filter(Marks_Alloted = None)
        Total_Absent = len(Absent)
        Total_Present = Total_Strength - Total_Absent
        Total_Fail = 0
```

```
            Total_Pass = 0
            for i in Previous_1:
                if i.Marks_Alloted == None or i.Marks_Alloted < 12:
                    Total_Fail += 1
                else:
                    Total_Pass += 1
            content = {
                'T_S' : len(Rolls),
                'average': marks_details('I Internal',Previous_1),
                'labels' :['Total Students Passed','Total Students Failed'],
                'data' : [Total_Pass,Total_Fail],
                'Attendance' : Previous_Internal_1,
                'Absent' : Total_Absent,
            }
            return render(request,'temp/Teacherninehalf.html',content)
        elif request.POST['SUBMIT'] == 'updatepage':
            Internal_1 = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Marks_Type = 'I Internal').filter(Roll_Number_Marks__in = Rolls)
            content = {
                    'Attendance' : Internal_1,
                    'Id' : Ids,
                    'Year' : Year,
                    'Group' : Group,
                    'Semester' : Semester,
```

```
                'Course' : Cours
            }
        return render(request, 'temp/Teacherninehalfupdate.html',content)
elif request.POST['SUBMIT'] == 'updatemarks':
        Total_Strength = len(Previous_1)
        Absent = Previous_1.filter(Marks_Alloted = None)
        Total_Absent = len(Absent)
        Total_Present = Total_Strength - Total_Absent
        Total_Fail = 0
        Total_Pass = 0
        for i in Previous_1:
            if i.Marks_Alloted == None or i.Marks_Alloted < 12:
                Total_Fail += 1
            else:
                Total_Pass += 1
        content = {
            'Absent' : Total_Absent,
            'Present': Total_Present,
            'T_S' : len(Rolls),
            'average': marks_details('I Internal',Previous_1),
        }
        return render(request,'temp/Teacherninehalf.html',content)
elif request.POST['SUBMIT'] == 'delete':
```

```
        Previous_1 = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'I Internal')
        for i in Previous_1:
            i.delete()
        previous_marks = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
        content = {
            'Attendance' : previous_marks,
            'Id' : Ids,
        }
        return render(request,'temp/Teacherninth.html',content)


def Teacher_fourteen(request,Id,Year,Group,Semester,Course):
    Cours = Course
    Ids = Id
    A = request.POST
    B = list(A.keys())[3:len(A)]
    Course = Courses.objects.get(Course_Code = Course)
    Id = Teacher.objects.get(Teacher_Id = Id)
    if request.POST['SUBMIT'] == 'submit' and "II Internal" not in  previous_marks_type:
        for i in B:
            if len(A[i]) != 0:
```

```
                    Instance = Student_Marks(Semester = Semester,Marks_Type = 'II Internal',Course_Code = Course,Roll_Number_Marks = Student.objects.get(Roll_Number= i),Teacher_Id_Marks = Id,Marks_Total = A['Total Marks'],Marks_Alloted = float(A[i]))
                    Instance.save()
                else:
                    Instance = Student_Marks(Semester = Semester,Marks_Type = 'II Internal',Course_Code = Course,Roll_Number_Marks = Student.objects.get(Roll_Number= i),Teacher_Id_Marks = Id,Marks_Total = A['Total Marks'])
                    Instance.save()
            Total_Absent = len(Absent)
            Total_Present = Total_Strength - Total_Absent
            Total_Fail = 0
            Total_Pass = 0
            for i in Previous_1:
                if i.Marks_Alloted == None or i.Marks_Alloted < 12:
                    Total_Fail += 1
                else:
                    Total_Pass += 1
            content = {
                'T_S' : len(Rolls),
                'average': marks_details('II Internal',Previous_1),
                'labels' :['Total Students Passed','Total Students Failed'],
            }
            return render(request,'temp/Teachertenhalf.html',content)
        elif request.POST['SUBMIT'] == 'updatepage':
```

```
                Internal_2 = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Marks_Type = 'II Internal').filter(Roll_Number_Marks__in = Rolls)
                content = {
                        'Attendance' : Internal_2,
                        'Id' : Ids,
                        'Year' : Year,
                        'Group' : Group,
                        'Semester' : Semester,
                        'Course' : Cours
                    }
                return render(request, 'temp/Teachertenhalfupdate.html',content)
            elif request.POST['SUBMIT'] == 'updatemarks':
                for student in Previous_Internal_2:
                    for s in B:
                        if student.Roll_Number_Marks.Roll_Number == s and str(student.Marks_Alloted) != A[s] and len(A[s]) != 0:
                            student.Marks_Alloted = float(A[s])
                            student.save()
                        elif student.Roll_Number_Marks.Roll_Number == s and str(student.Marks_Alloted) != A[s] and len(A[s]) == 0:
                            student.Marks_Alloted = None
                            student.save()
                for i in Previous_1:
                    if i.Marks_Alloted == None or i.Marks_Alloted < 12:
```

```
                Total_Fail += 1
            else:
                Total_Pass += 1
        content = {
            'Absent' : Total_Absent,
            'Present': Total_Present,
            'T_S' : len(Rolls),
            'average': marks_details('II Internal',Previous_1),
            'labels' :['Total Students Passed','Total Students Failed'],
            'data' : [Total_Pass,Total_Fail],
            'Attendance' : Previous_1,
            'Id' : Ids,
            'Year' : Year,
            'Group' : Group,
            'Semester' : Semester,
            'Course' : Cours
        }
        return render(request,'temp/Teachertenhalf.html',content)
elif request.POST['SUBMIT'] == 'delete':
        content = {
            'Attendance' : previous_marks,
            'Id' : Ids,
            'Year' : Year,
            'Group' : Group,
```

```
                'Semester' : Semester,
                'Course' : Cours
            }
            return render(request,'temp/Teachertenth.html',content)

def Teacher_fifteen(request,Id,Year,Group,Semester,Course):
    Cours = Course
    Ids = Id
    A = request.POST
    B = list(A.keys())[3:len(A)]
    Course = Courses.objects.get(Course_Code = Course)
    Id = Teacher.objects.get(Teacher_Id = Id)
            Previous_Assignment = Previous_Assignment.filter(Marks_Type = 'Assignment')
            content = {
                'Attendance' : Previous_Assignment,
                'Id' : Ids,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Cours
            }
            return render(request,'temp/Teacherelevenhalf.html',content)
    elif request.POST['SUBMIT'] == 'updatepage':
```

```
            Assignment = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Marks_Type = 'Assignment').filter(Roll_Number_Marks__in = Rolls)
            content = {
                        'Attendance' : Assignment,
                        'Id' : Ids,
                        'Year' : Year,
                    }
            return render(request, 'temp/Teacherelevenhalfupdate.html',content)
        elif request.POST['SUBMIT'] == 'updatemarks':
            Previous_Assignment = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'Assignment')
            for student in Previous_Assignment:
                for s in B:
                    if student.Roll_Number_Marks.Roll_Number == s and str(student.Marks_Alloted) != A[s]:
                                student.Marks_Alloted = float(A[s])
            content = {
                'Attendance' : Previous_Assignment_2,
                'Id' : Ids,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Cours
            }
```

```
            return render(request,'temp/Teacherelevenhalf.html',content)
        elif request.POST['SUBMIT'] == 'delete':
            Previous_Assignment = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'Assignment')
            content = {
                'Attendance' : previous_marks,
                'Id' : Ids,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Cours
            }
            return render(request,'temp/Teachereleven.html',content)

def Teacher_sixteen(request,Id,Year,Group,Semester,Course):
    Cours = Course
    Ids = Id
    A = request.POST
    B = list(A.keys())[3:len(A)]
    Course = Courses.objects.get(Course_Code = Course)
    Id = Teacher.objects.get(Teacher_Id = Id)
    previous_marks_type = set()
    for i in Previous_Semester.values_list('Marks_Type'):
```

```
            previous_marks_type.add(i[0])
        if request.POST['SUBMIT'] == 'submit' and "Semester" not in  previous_marks_type:
            Previous_Internal_1 = Previous_Semester.filter(Marks_Type = 'Semester')
            Previous_1 = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'Semester')
            content = {
                'T_S' : len(Rolls),
                'average': marks_details('Semester',Previous_1),
                'labels' :['Total Students Passed','Total Students Failed'],
                'data' : [Total_Pass,Total_Fail],
                'Course' : Cours
            }
            return render(request,'temp/Teachertwelvehalf.html',content)
        elif request.POST['SUBMIT'] == 'updatepage':
            Semesters = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Marks_Type = 'Semester').filter(Roll_Number_Marks__in = Rolls)
            content = {
                    'Attendance' : Semesters,
                    'Id' : Ids,
                }
            return render(request, 'temp/Teachertwelvehalfupdate.html',content)
        elif request.POST['SUBMIT'] == 'updatemarks':
```

```
            Previous_Semester = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'Semester')
            content = {
                'Absent' : Total_Absent,
                'Present': Total_Present,
                'T_S' : len(Rolls),
                'average': marks_details('Semester',Previous_1),
            }
            return render(request,'temp/Teachertwelvehalf.html',content)
        elif request.POST['SUBMIT'] == 'delete':
            Previous_Semester = Student_Marks.objects.filter(Semester = Semester).filter(Course_Code = Course).filter(Teacher_Id_Marks = Id).filter(Roll_Number_Marks__in = Rolls).filter(Marks_Type = 'Semester')
            for i in Previous_Semester:
                i.delete()
            previous_marks = Student.objects.filter(Group = Group).filter(Year_Of_Admission = Year).order_by('Roll_Number')
            content = {
                'Attendance' : previous_marks,
                'Id' : Ids,
                'Year' : Year,
                'Group' : Group,
                'Semester' : Semester,
                'Course' : Cours
```

```
        }
        return render(request,'temp/Teachertwelve.html',content)


def Teacher_seventeen(request,Id,Year,Semester,Course,Group):
    A = request.POST
    B = list(A.keys())[2:len(A)]
        for student in Previous_Attendance:
            for j in B:
                if student.Roll_Number_Marks.Roll_Number == j and student.Marks_Alloted != float(A[j]):
                    student.Marks_Alloted = float(A[j])
                    student.save()
    datas = list()
    for i in range(len(studentgraph)):
        datas.append(list(dataframe2.loc[i]))
    content = {
        'Id' : Id,
        'Year' : Year,
        'course' : Course,
        'Cos' : cos[0],
        'T_W' : Bargraph(Id,Year,Group,Semester,Course)[2],
        'len' : len(studentgraph),
        'p' : round(Bargraph(Id,Year,Group,Semester,Course)[1],2),
        'below' : list(dataframe['index'])[0]
```

```
        }
        return render(request, 'temp/TeacherReport.html',content)


#STUDENT ATTENDANCE
def Student_second(request,Roll_Number,Group):
        Courses_Involved = Teacher_Course.objects.filter(Group = Group).filter(Semester_Teacher = "1")
        if len(Courses_Involved) != 0:
                content = {
                #'Attendance' : courses_attendance
                }
                return render(request,'temp/Studentsecond.html',content)
        else:
                content = {
                'Semester' : 'Semester 1',
                "no": "NO ATTENDANCE POSTED YET",
                }
                return render(request,'temp/Studentthird.html',content)
#STUDENT MARKS
def Student_third(request,Roll_Number,Group):
        Courses_Involved = Teacher_Course.objects.filter(Group = Group).filter(Semester_Teacher = "2")
        if len(Courses_Involved) != 0:
                content = {
```

```
                'Semester' : 'Semester 2',
                'Roll_Number': Roll_Number,
                'Attendance' : Courses_Involved,
                'Group' : Group,
                'sem' : "2"
                }
                return render(request,'temp/Studentsecond.html',content)
        else:
                'Attendance' : Courses_Involved,
                'Group' : Group
                }
                return render(request,'temp/Studentthird.html',content)
def Student_fourth(request,Roll_Number,Group):
                content = {
                'Semester' : 'Semester 3',
                "no": "NO ATTENDANCE POSTED YET",
                'Roll_Number': Roll_Number,
                'Attendance' : Courses_Involved,
                'Group' : Group
                }
                return render(request,'temp/Studentthird.html',content)
def Student_fifth(request,Roll_Number,Group):
        Courses_Involved = Teacher_Course.objects.filter(Group = Group).filter(Semester_Teacher =
"4")
```

```
    if len(Courses_Involved) != 0:
            content = {
            'Semester' : 'Semester 4',
            'Roll_Number': Roll_Number,
            'Attendance' : Courses_Involved,
            'Group' : Group,
            "sem" : "4"
            }
            return render(request,'temp/Studentsecond.html',content)
    else:
            content = {
            'Semester' : 'Semester 4',
            "no": "NO ATTENDANCE POSTED YET",
            'Roll_Number': Roll_Number,
            'Attendance' : Courses_Involved,
            'Group' : Group
            }
            return render(request,'temp/Studentthird.html',content)
def Student_sixth(request,Roll_Number,Group):
    Courses_Involved = Teacher_Course.objects.filter(Group = Group).filter(Semester_Teacher = "5")
    if len(Courses_Involved) != 0:
            'Group' : Group,
            "sem" : "5"
```

```
            }
            return render(request,'temp/Studentsecond.html',content)
        else:
            content = {
            'Semester' : 'Semester 5',
            "no": "NO ATTENDANCE POSTED YET",
            'Roll_Number': Roll_Number,
            'Attendance' : Courses_Involved,
            'Group' : Group
            }
            return render(request,'temp/Studentthird.html',content)
def Student_seventh(request,Roll_Number,Group):
    Courses_Involved = Teacher_Course.objects.filter(Group = Group).filter(Semester_Teacher = "6")
    if len(Courses_Involved) != 0:
            content = {
            'Semester' : 'Semester 6',
            'Roll_Number': Roll_Number,
            'Attendance' : Courses_Involved,
            'Group' : Group,
            "sem" : "6"
            }
            return render(request,'temp/Studentsecond.html',content)
    else:
```

```
            content = {
            'Group' : Group
            }
            return render(request,'temp/Studentthird.html',content)

def Student_eighth(request,Roll_Number,Group,Id,Course,Semester):
            Teacher_Id_Attendance = Id).filter(Course_Code = Course).filter(Semester = Semester).filter(Attendance_Date = 'PRESENT'))
      Total_Absent = Total_working - Total_Present
      if Total_working != 0:
            content = {
                  'Attendance' : Student_attendance,
                  'T_W' : Total_working,
                  'T_P' : Total_Present,
                  'T_A' : Total_Absent
            }
            return render(request,'temp/Studentfourth.html',content)
      else:
            return HttpResponse('No Attendane posted yet')

def Student_ninth(request,Roll_Number,Group):
      Internal_1_marks = Student_Marks.objects.filter(Roll_Number_Marks = Roll_Number).filter(Marks_Type = 'I Internal').filter(Semester = "1")
      content ={
```

```
                'Student' : Internal_1_marks,
                'Roll_Number' : Roll_Number,
                'Group' : Group,
                'Attendance' : Semesters,
                'Marks' : Marks
        }
        return render(request,'temp/Studentfifth.html',content)

def Student_tenth(request,Roll_Number,Group,Semester,Exam):
        Internal_1_marks = Student_Marks.objects.filter(Roll_Number_Marks = Roll_Number).filter(Marks_Type = Marks_Rank[Exam]).filter(Semester = Semester_Rank[Semester])
        if len(Internal_1_marks) != 0:
                content ={
                        'Student' : Internal_1_marks,
                        'Roll_Number' : Roll_Number,
                }
                return render(request,'temp/Studentfifth.html',content)
        else:
                Text = 'No Marks Posted Yet'
                content ={
                        'Student' : Internal_1_marks,
                        'Roll_Number' : Roll_Number,
                        'Group' : Group,
                }
```

return render(request,'temp/Studentsixth.html',content)

HTML

```
{% load static %}

<!DOCTYPE html>

<html>

    <head>

        <title>STUDENT RECORD MANAGEMENT SYSTEM</title>

        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

        <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>

        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>

    </head>

    <body>

        <div class="container">

            {% block content %}
```

```
                {% endblock %}
            <!-- Content here -->
            </div>
        </body>
</html>
<!---style = "background-repeat: no-repeat;
 background-attachment: fixed; align-self: center;"--->

{% extends "main.html" %}
{% block content %}
{%load static%}
<head>
  <link rel="stylesheet" type="text/css" href="css/studentstyle.css">
  <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap" rel="stylesheet">
  <script src="https://kit.fontawesome.com/a81368914c.js"></script>
</head>
<style>

.title{
  margin-left:20%;
  width:10%
}
.v1 {
```

```
  border-left: 2px solid #00868B;
  height: 500px;
}
*{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

body{
    font-family:  Garamond, serif;
    font-weight: normal;
    overflow: hidden;
}

.clouds{
  position: fixed;
  bottom: 0;
  left: 0;
  height: 100%;
  width: 2000px;
  z-index: -1;
}
```

```
.img{
  display: flex;
  justify-content: flex-end;
  align-items: center;

}

.login-content{
  display: flex;
  justify-content: center;
  align-items: center;
  text-align: center;
  width:420px;
  height:490px;
    box-sizing: border-box;
    border:1px solid #62021E ;
    border-radius: 40px;
    background-color: e2eafc;
    margin-top: 130px;

}



}
```

```
.img{
  width: 500px;
}

form{
  width: 360px;
}

.login-content img{
    height: 100px;
    width: 100px;
}

.login-content h2{
  margin: 15px 0;
  color: #333;
  text-transform: uppercase;
  font-size: 2.9rem;
}

.login-content .input-div{
  position: relative;
    display: grid;
```

```
    grid-template-columns: 7% 93%;
    margin: 25px 0;
    padding: 5px 0;
    border-bottom: 2px solid #d9d9d9;
}

.login-content .input-div.one{
  margin-top: 0;
}

.i{
  color: #d9d9d9;
  display: flex;
  justify-content: center;
  align-items: center;
  border-radius: 50px;
}

.i{
  transition: .3s;
   border-radius: 50px;

}
```

```
.input-div > div{
    position: relative;
  height: 45px;
}

.input-div > div > h5{
  position: absolute;
  left: 10px;
  top: 50%;
  transform: translateY(-50%);
  color: #999;
  font-size: 18px;
  transition: .3s;
}

.input-div:before, .input-div:after{
  content: '';
  position: absolute;
  bottom: -2px;
  width: 0%;
  height: 2px;
  background-color: #38d39f;
  transition: .4s;
}
```

```
.input-div:before{
  right: 50%;
}

.input-div:after{
  left: 50%;
}

.input-div.focus:before, .input-div.focus:after{
  width: 50%;
}

.input-div.focus > div > h5{
  top: -5px;
  font-size: 15px;
}

.input-div.focus > .i > i{
  color: #38d39f;
}

.input-div > div > input{
  position: absolute;
```

```
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  border: none;
  outline: none;
  background: none;
  padding: 0.5rem 0.7rem;
  font-size: 1.2rem;
  color: #555;
  font-family: 'poppins', sans-serif;
}

.input-div.pass{
  margin-bottom: 4px;

}

a{
  display: block;
  text-align: right;
  text-decoration: none;
  color: #999;
  font-size: 0.9rem;
```

```
  transition: .3s;
}

a:hover{
  color: #38d39f;
}

.btn{
  display: block;
  width: 100%;
  height: 50px;
  border-radius: 25px;
  outline: none;
  border: none;
  background-image: linear-gradient(to right,#f79d6d,#f5ae4b,#f36f29);
  background-size: 200%;
  font-size: 1.2rem;
  color: #fff;
  font-family: 'Poppins', sans-serif;
  text-transform: uppercase;
  margin: 1rem 0;
  cursor: pointer;
  transition: .5s;
}
```

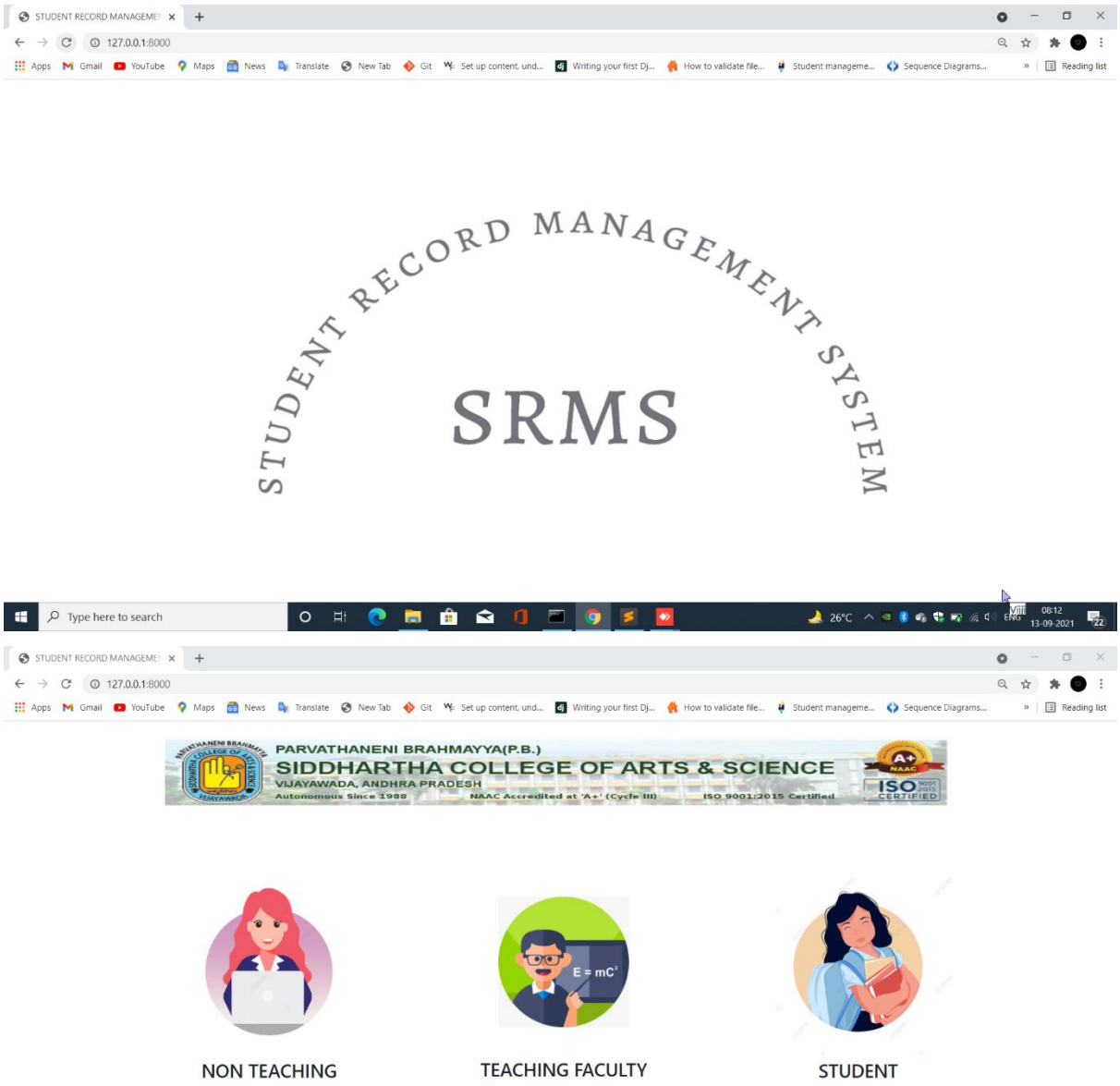
```
.btn:hover{
  background-position: right;
}
.fg{
  font-size: 20px;
}

@media screen and (max-width: 1050px){
  .container{
    grid-gap: 5rem;
  }
}

@media screen and (max-width: 1000px){
  form{
    width: 290px;
  }
  .img img{
    width: 400px;
  }
}

@media screen and (max-width: 900px){
  .container{
```

```
    grid-template-columns: 1fr;
  }

  .img{
    display: none;
  }

  .clouds{
    display: none;
  }

  .login-content{
    justify-content: center;
  }
}
</style><br><br>
<div class='a'>
<div class="container1">
  <br><br><br>
  <div class="row" >
    {% load static %}
    <div class="col">
      <img src="{% static '/Image/facultyside3.jpg' %}" style="height:500px;margin-top:20px; margin-left:-30px;">
```

```
</div>
<div class="col" style="margin-top: -650px;margin-left: 600px;" >
  <center>
   <div class="login-content">
  <form action='/Faculty/Homepage/Welcome/' method="POST">
    {% csrf_token %}
    <img src="{% static '/Image/nonteachingicon.jpeg' %}">

    <h3 class="title">FACULTY LOGIN</h3>
          <div class="input-div one">
            <div class="i">
              <i class="fas fa-user"></i>
            </div>
            <div class="div">
              <!--<h5  style="font-family: Cursive;">Username</h5>-->
              <input type="text" class="input" name = "Username" placeholder="Username" required="" autocomplete="off">
            </div>
          </div>
          <div class="input-div pass">
            <div class="i">
              <i class="fas fa-lock"></i>
            </div>
            <div class="div">
```

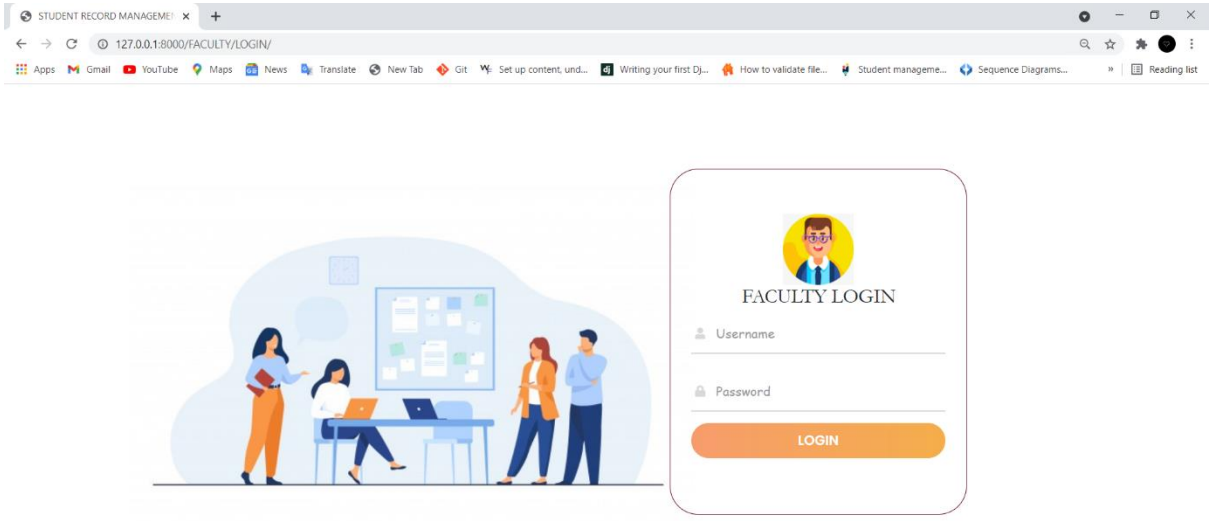
```
                <!--<h5  style="font-family: Cursive;">Password</h5>-->
                <input type="password" class="input" name="Password" placeholder="Password" required="" autocomplete="off">
              </div>
            </div>
            <input type="submit" class="btn" value="Login">
          </form>
        </div>
      </center>
    </div>
    <script type="text/javascript" src="js/main.js"></script>
</div>
{% endblock %}
```

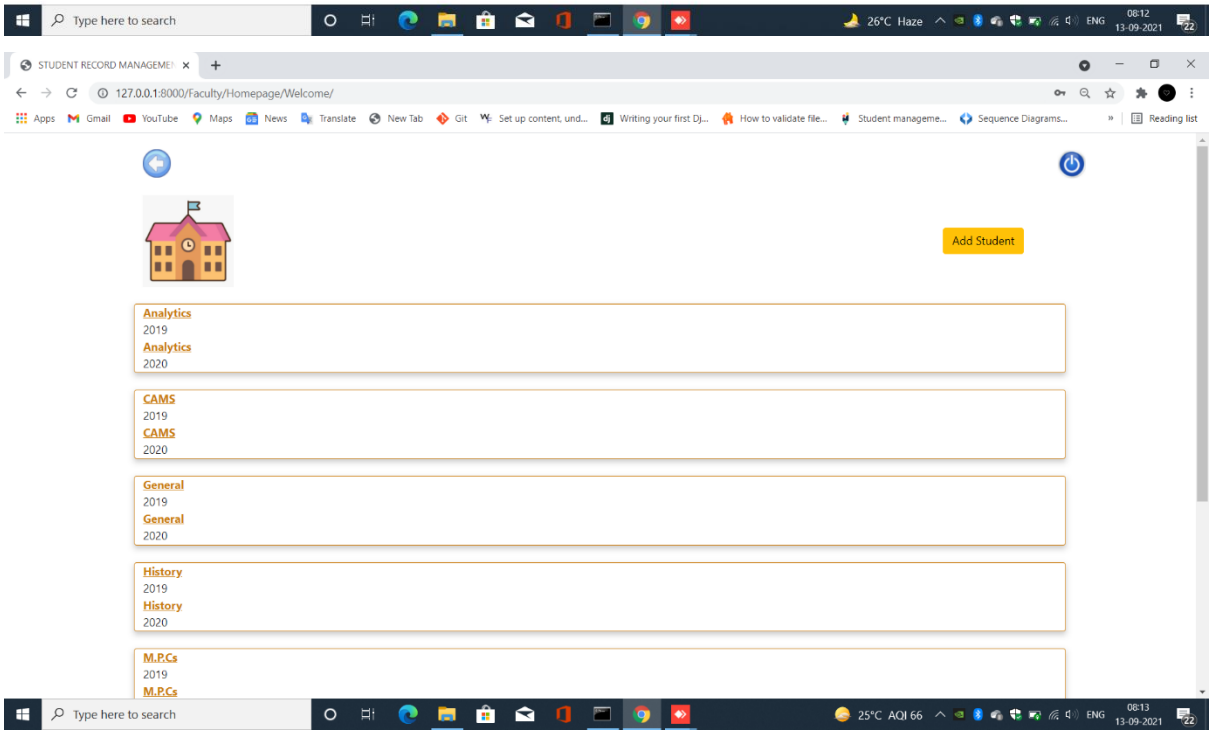
9. SCREENSHOTS



SRMS

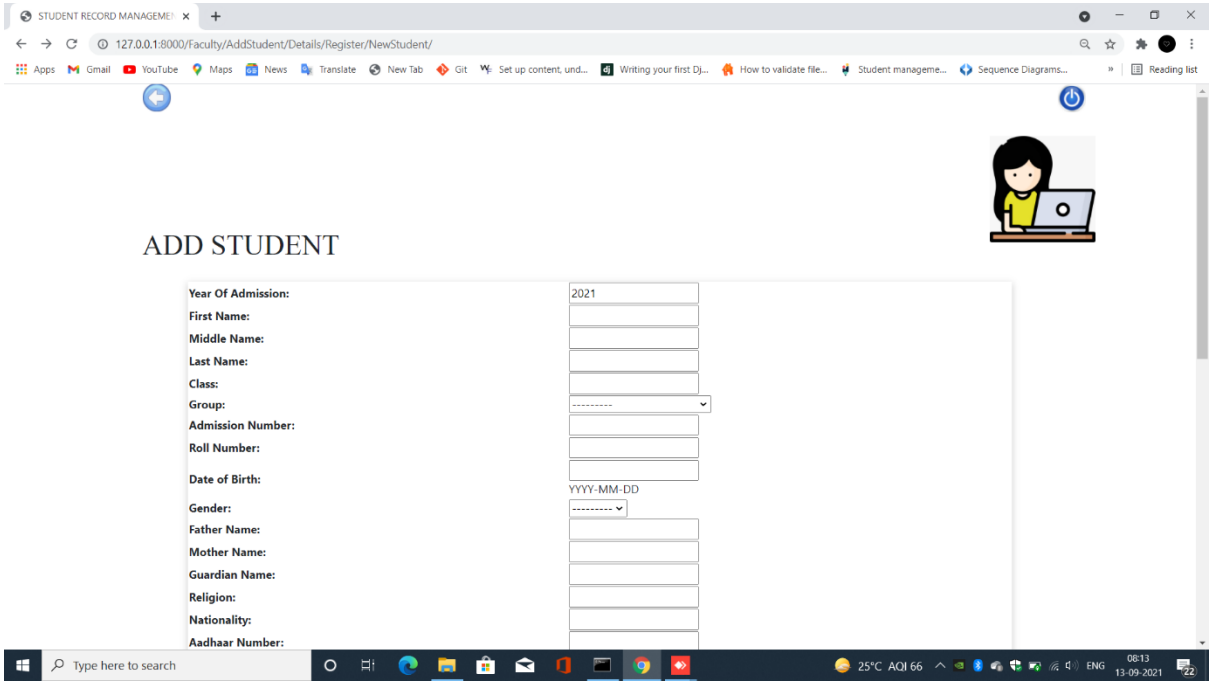
BEGINNING PAGE AND SELECTION PAGE

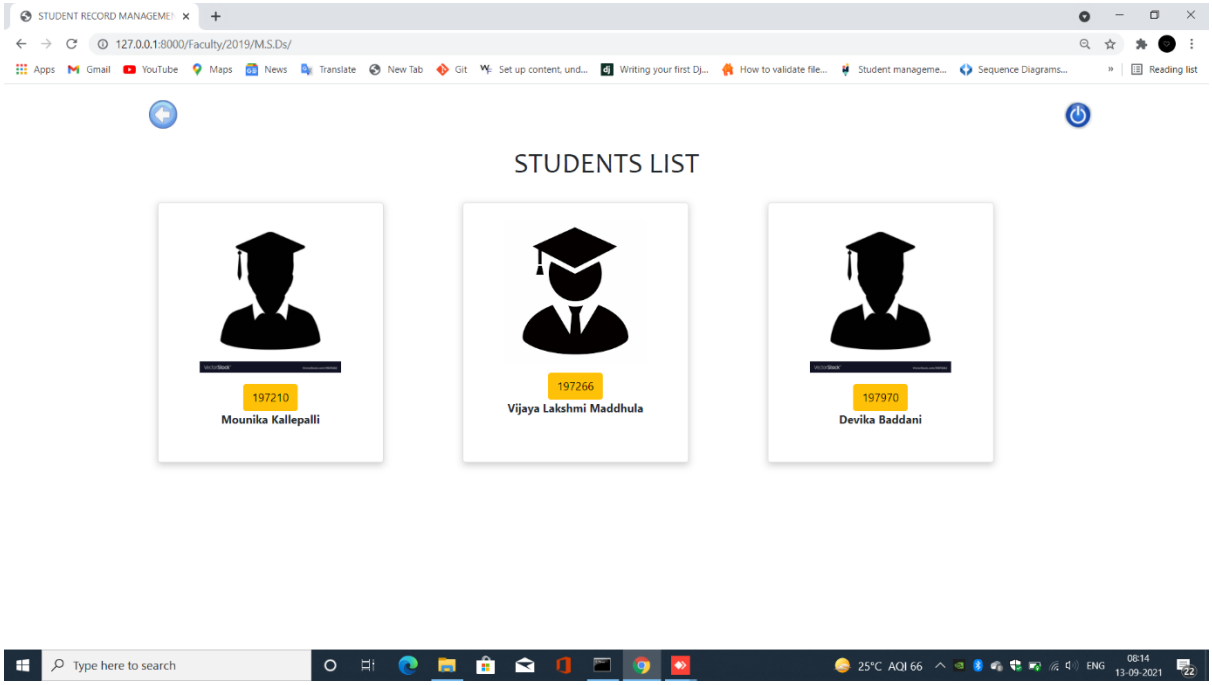




FACULTY

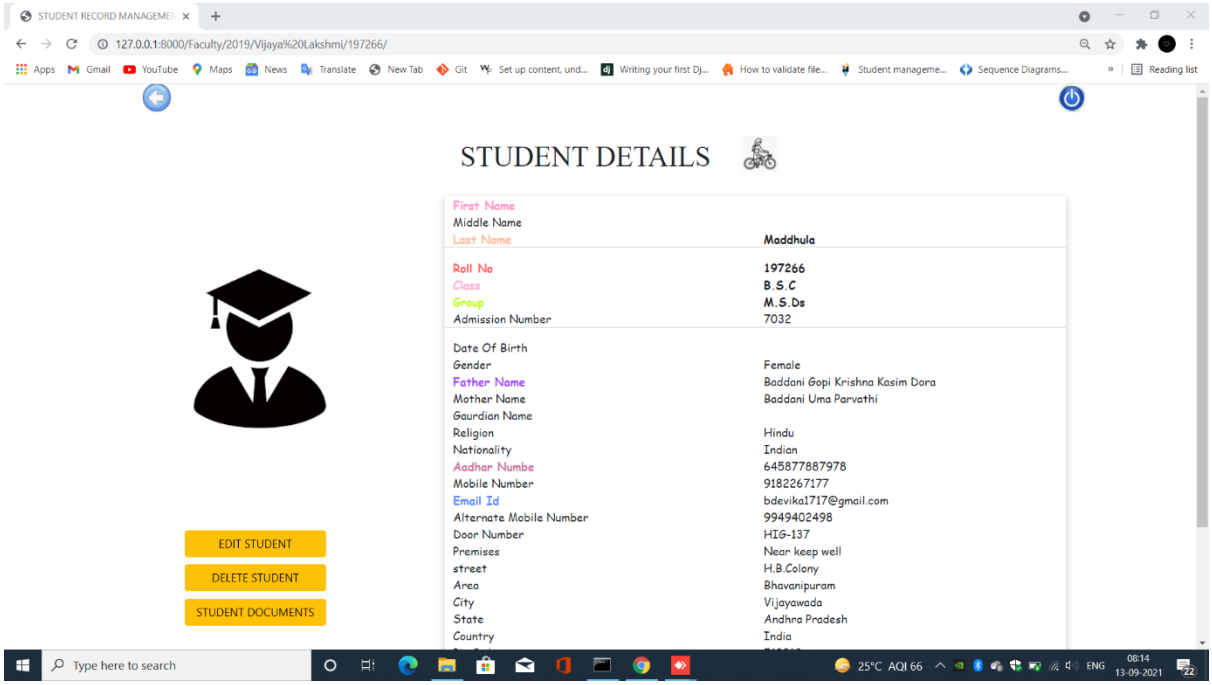
FACULTY LOGIN AND COURSES LIST PAGE

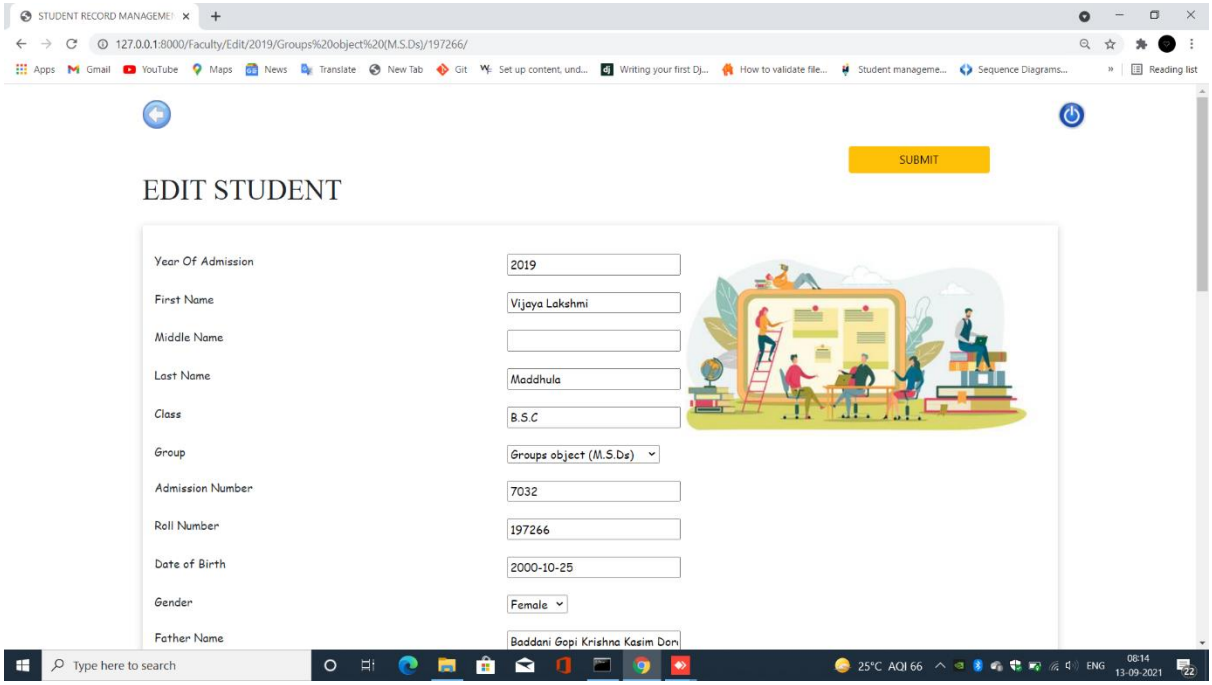




FACULTY

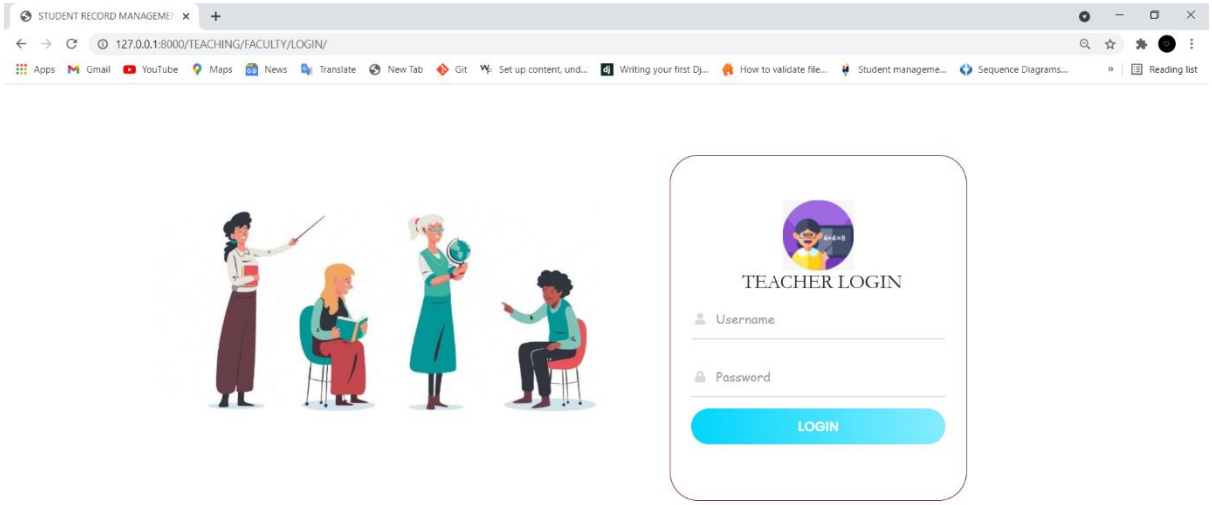
ADD STUDENT AND CLASS LIST PAGES

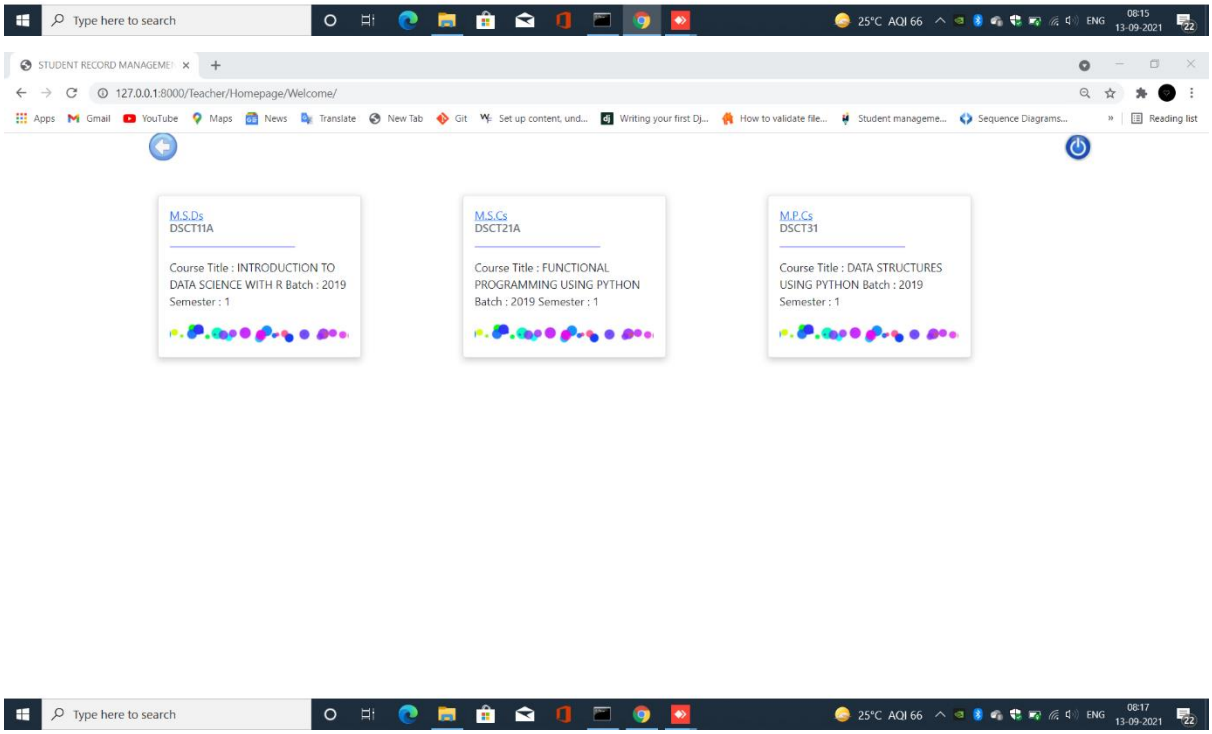




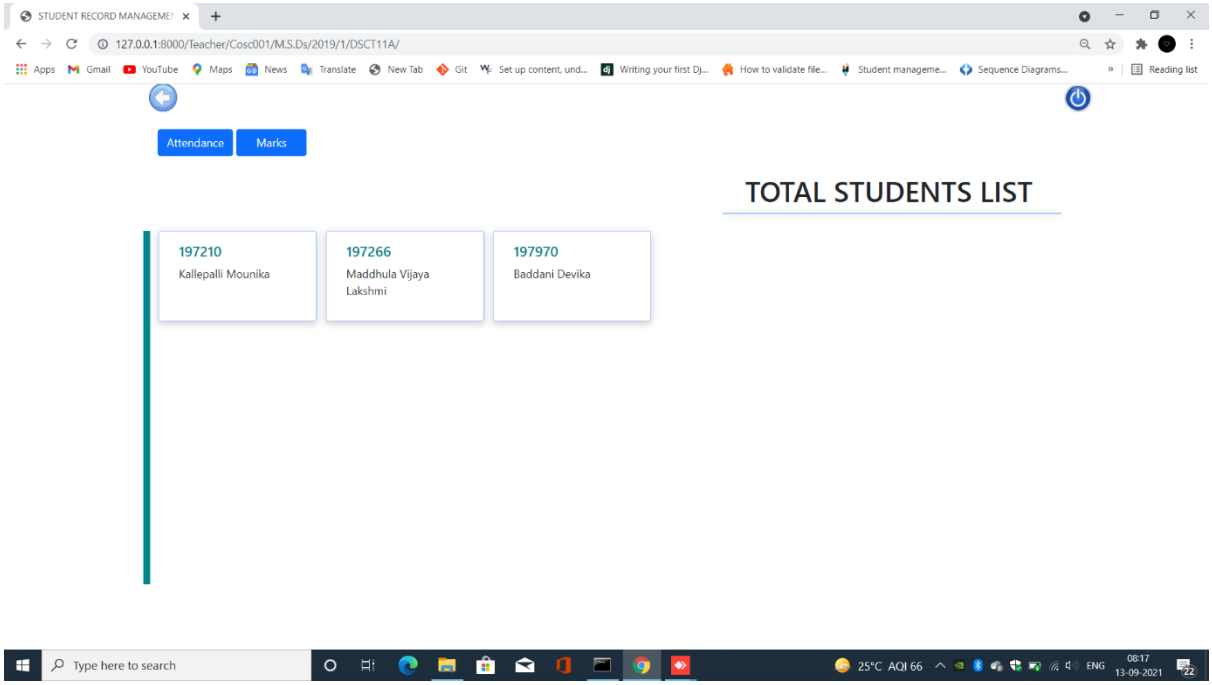
FACULTY

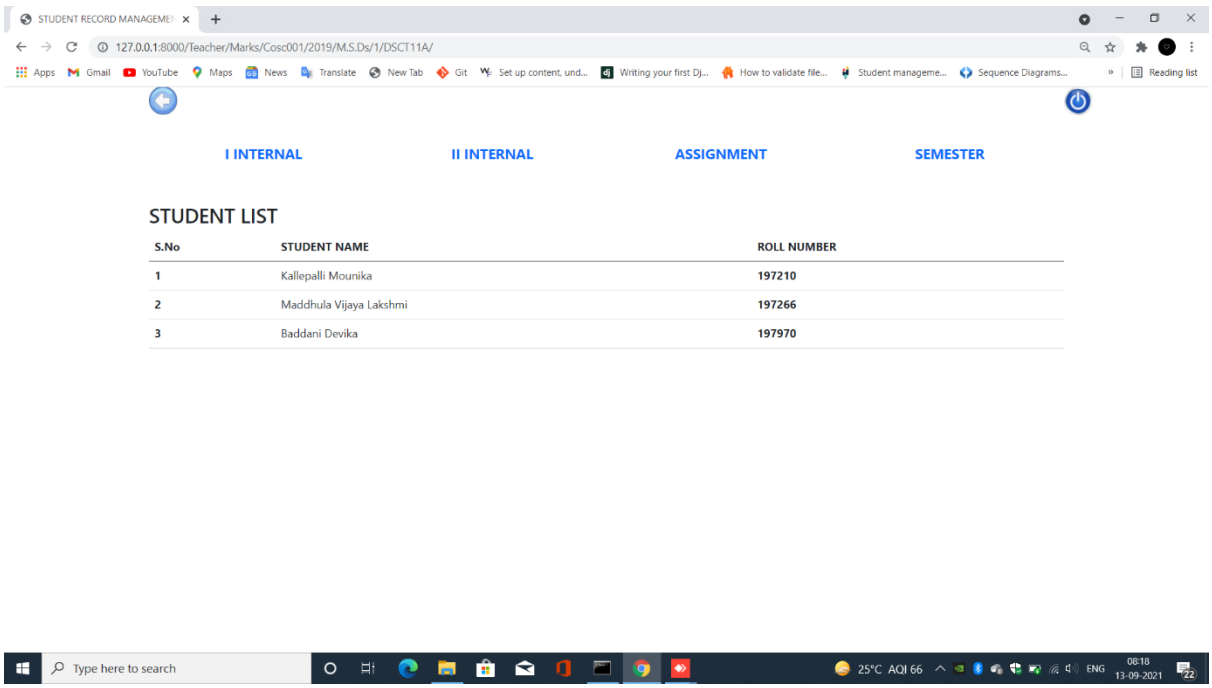
STUDENT DETAILS AND STUDENT EDIT PAGES





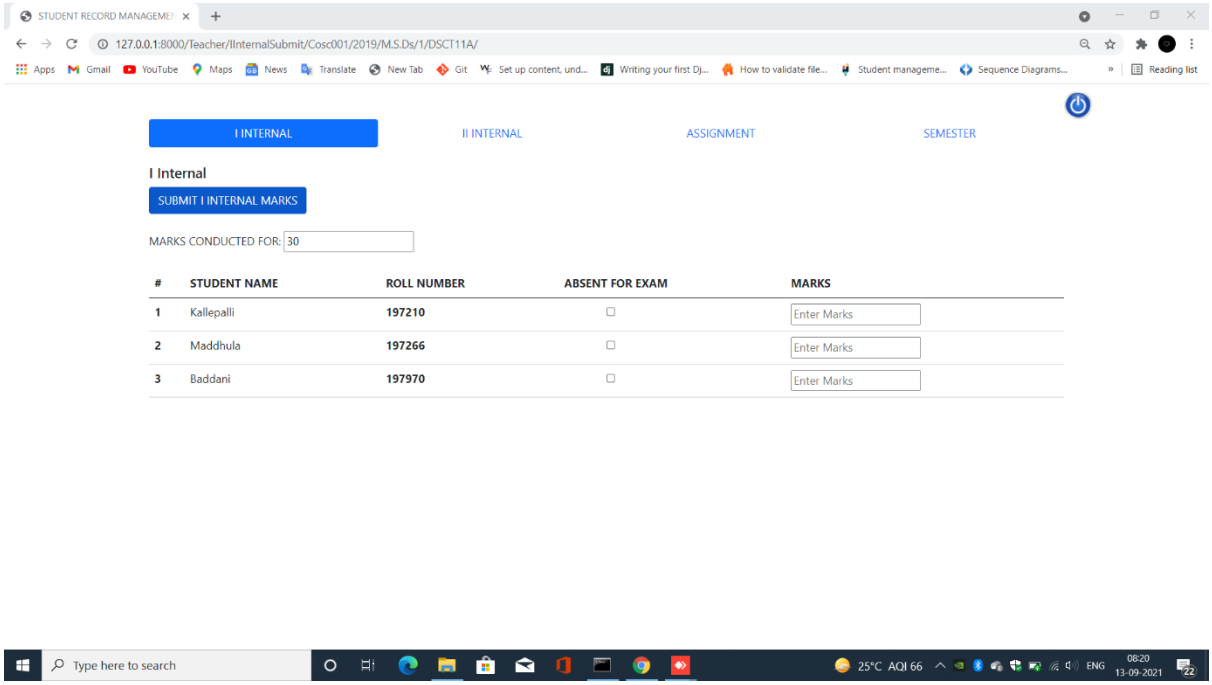
TEACHER LOGIN AND COURSES LIST PAGE

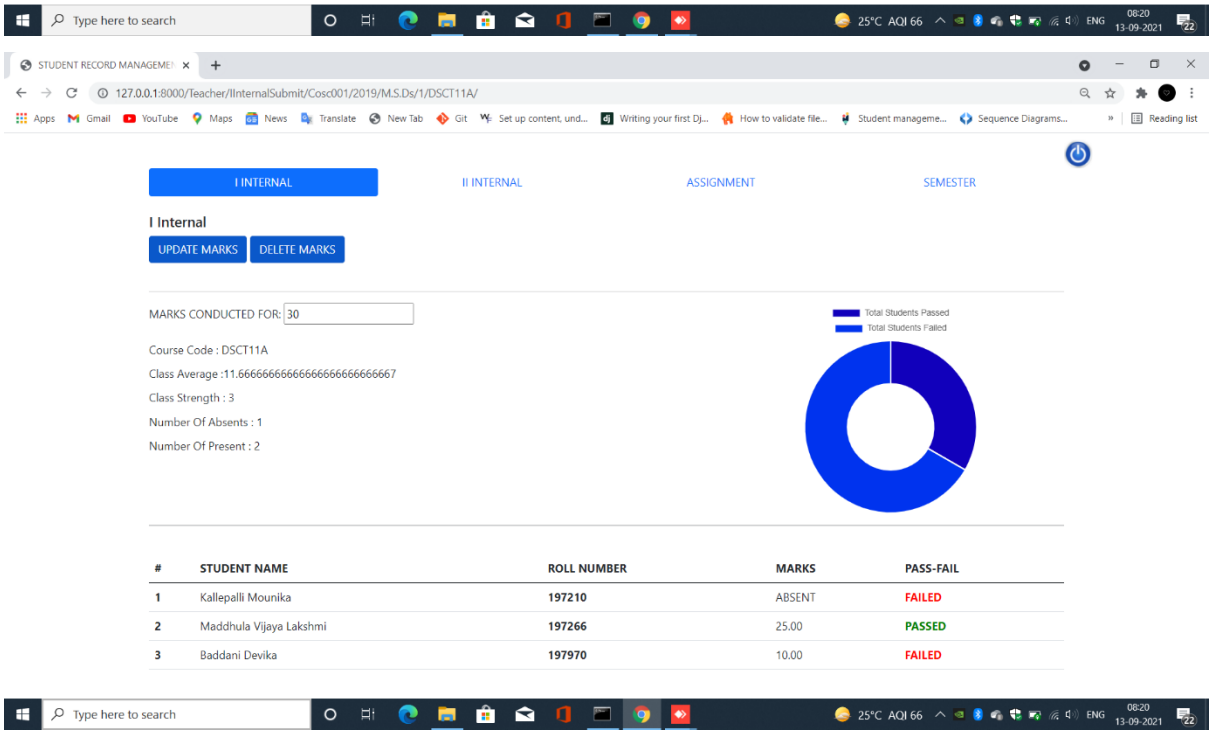




TEACHER

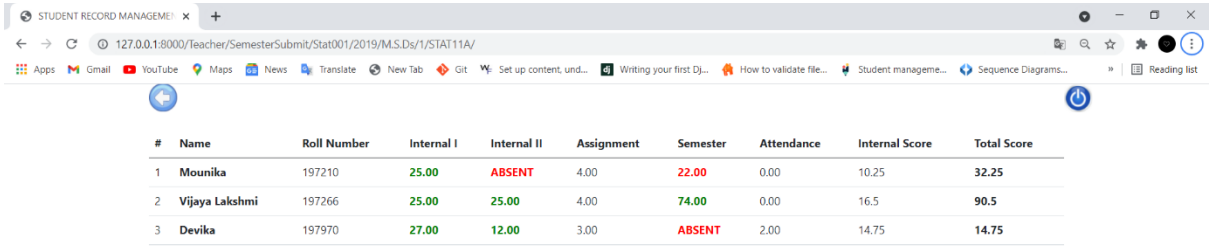
STUDENT LIST AND MARKS PAGE



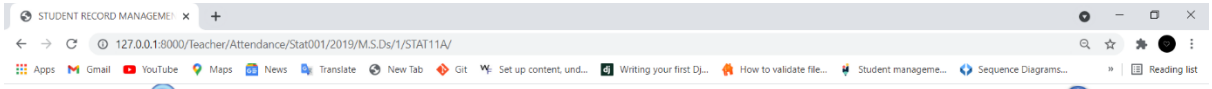


TEACHER

MARKS SUBMISSION AND MARKS REPORT PAGE



#	Name	Roll Number	Internal I	Internal II	Assignment	Semester	Attendance	Internal Score	Total Score
1	Mounika	197210	25.00	ABSENT	4.00	22.00	0.00	10.25	32.25
2	Vijaya Lakshmi	197266	25.00	25.00	4.00	74.00	0.00	16.5	90.5
3	Devika	197970	27.00	12.00	3.00	ABSENT	2.00	14.75	14.75

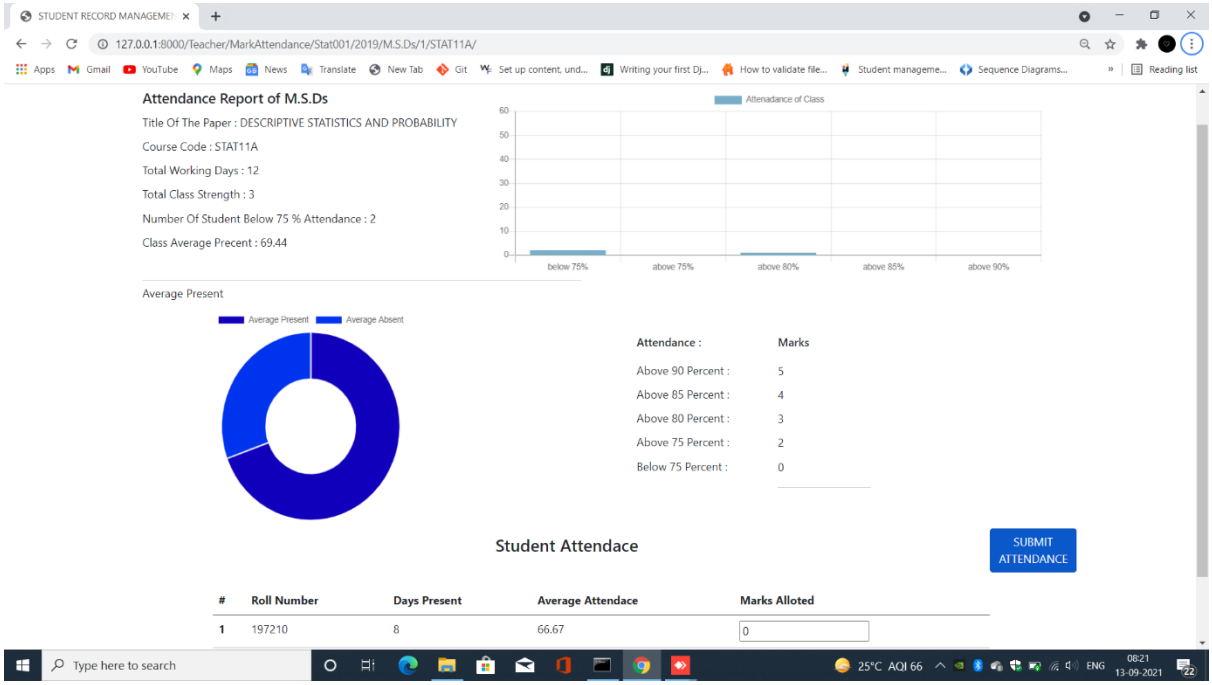


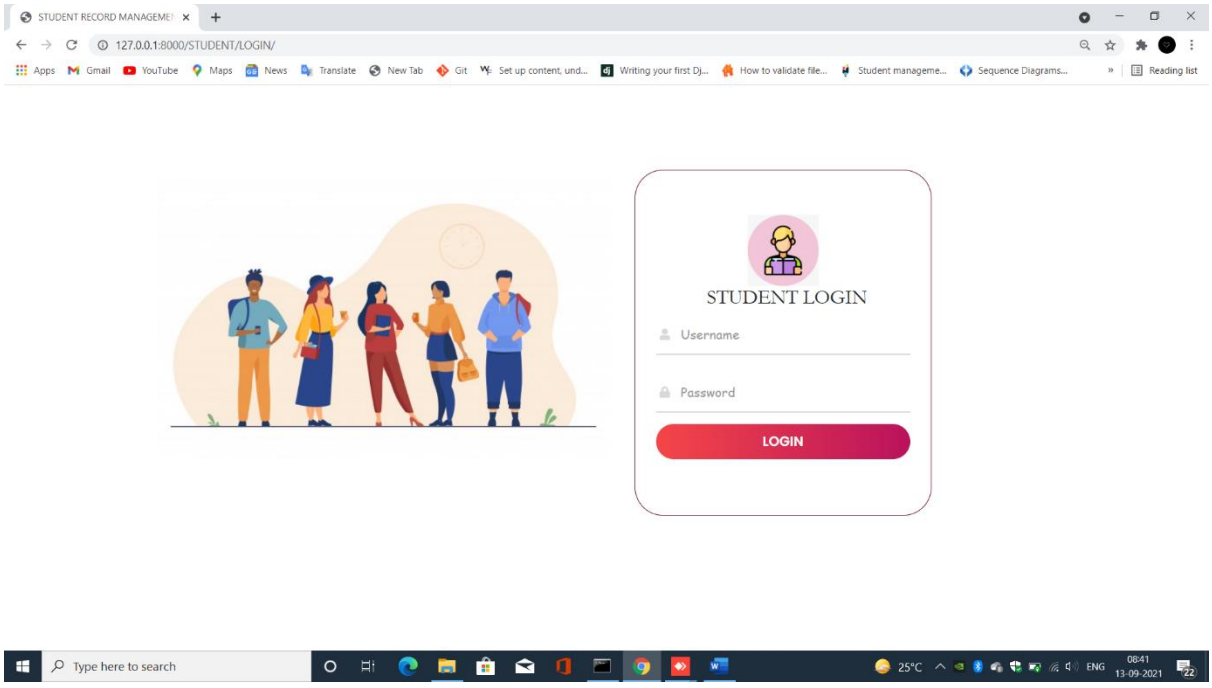
MARK ATTENDANCE DELETE ATTENDANCE REPORT GENERATED

checkbox	#	Date of the Attendance	Number Of Absent	Number Of Present
☐	1	Aug. 13, 2021	2	1
☐	2	Aug. 14, 2021	1	2
☐	3	Aug. 20, 2021	2	1
☐	4	Aug. 31, 2021	1	2
☐	5	Sept. 1, 2021	0	3
☐	6	Sept. 2, 2021	0	3
☐	7	Sept. 3, 2021	1	2
☐	8	Sept. 5, 2021	2	1
☐	9	Sept. 6, 2021	0	3
☐	10	Sept. 7, 2021	0	3
☐	11	Sept. 8, 2021	1	2
☐	12	Sept. 9, 2021	1	2

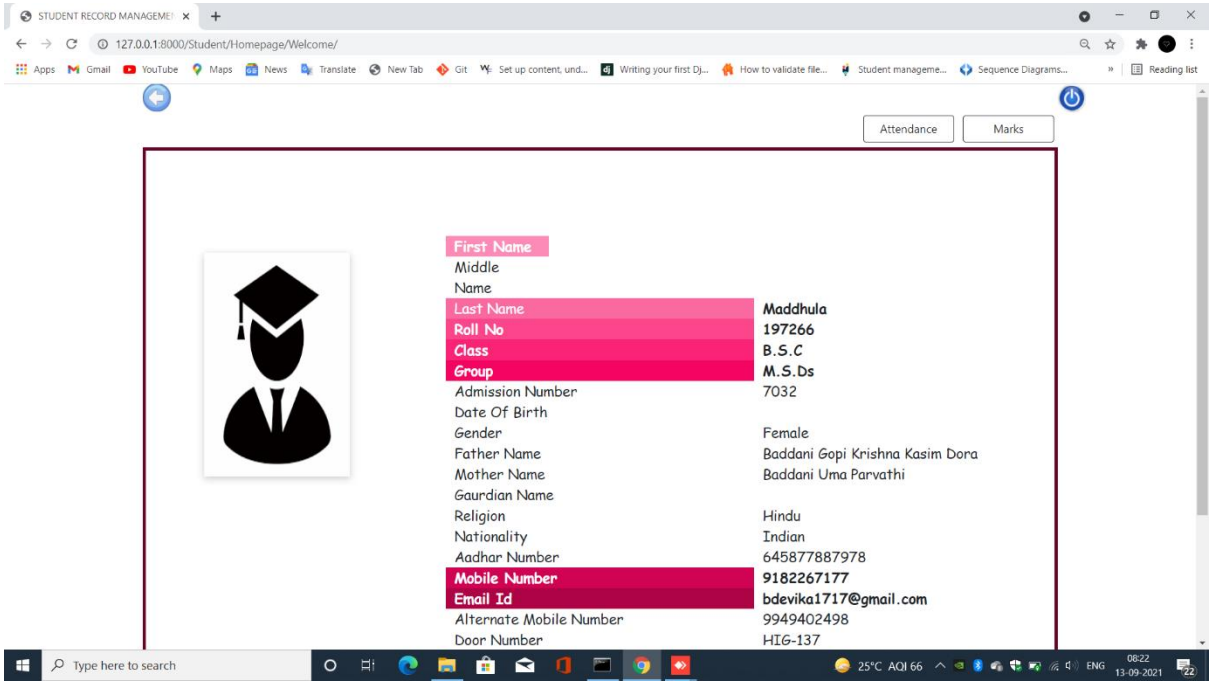
TEACHER

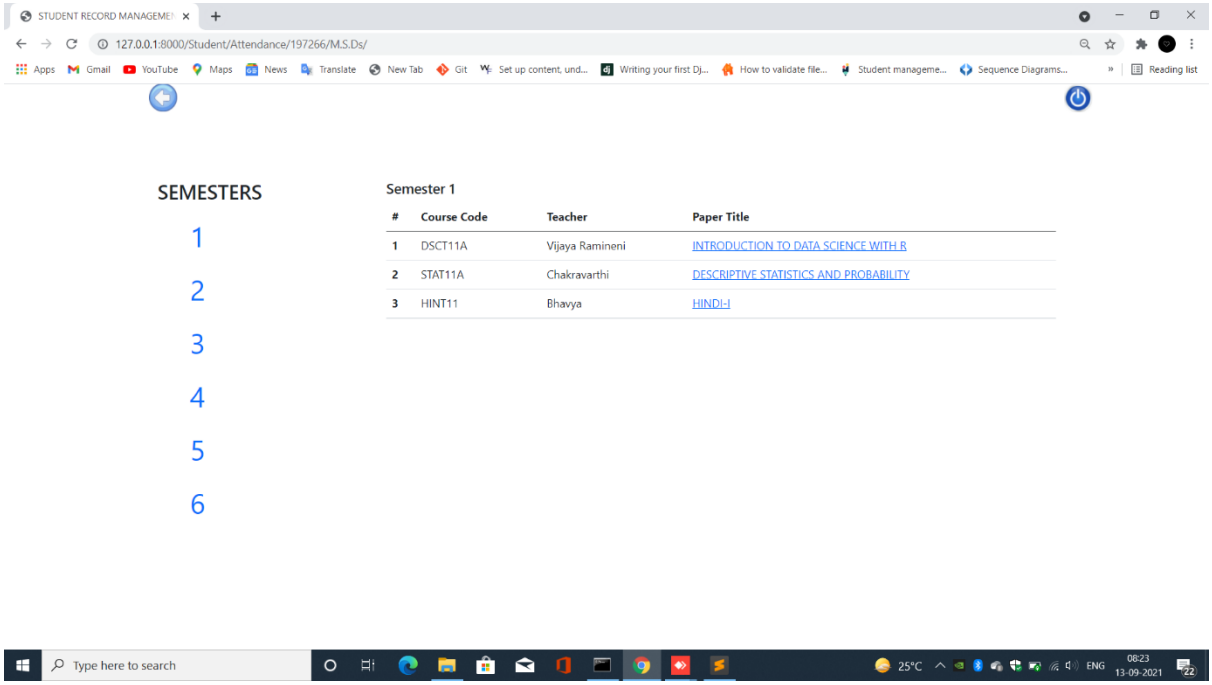
TOTAL MARKS REPORT AND ATTENDANCE LIST PAGES





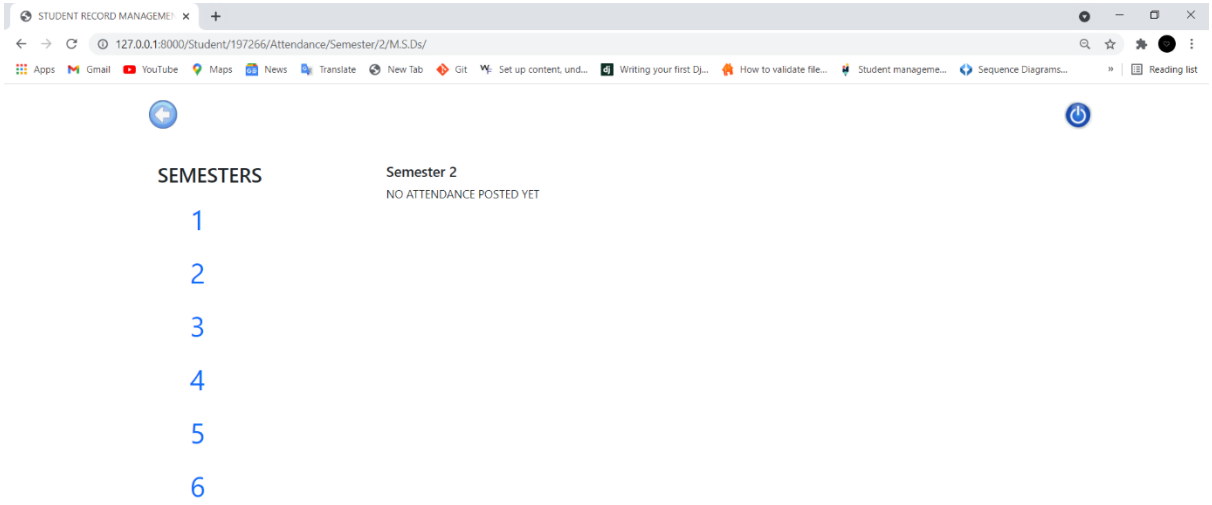
ATTENDANCE REPORT AND STUDENT LOGIN PAGE

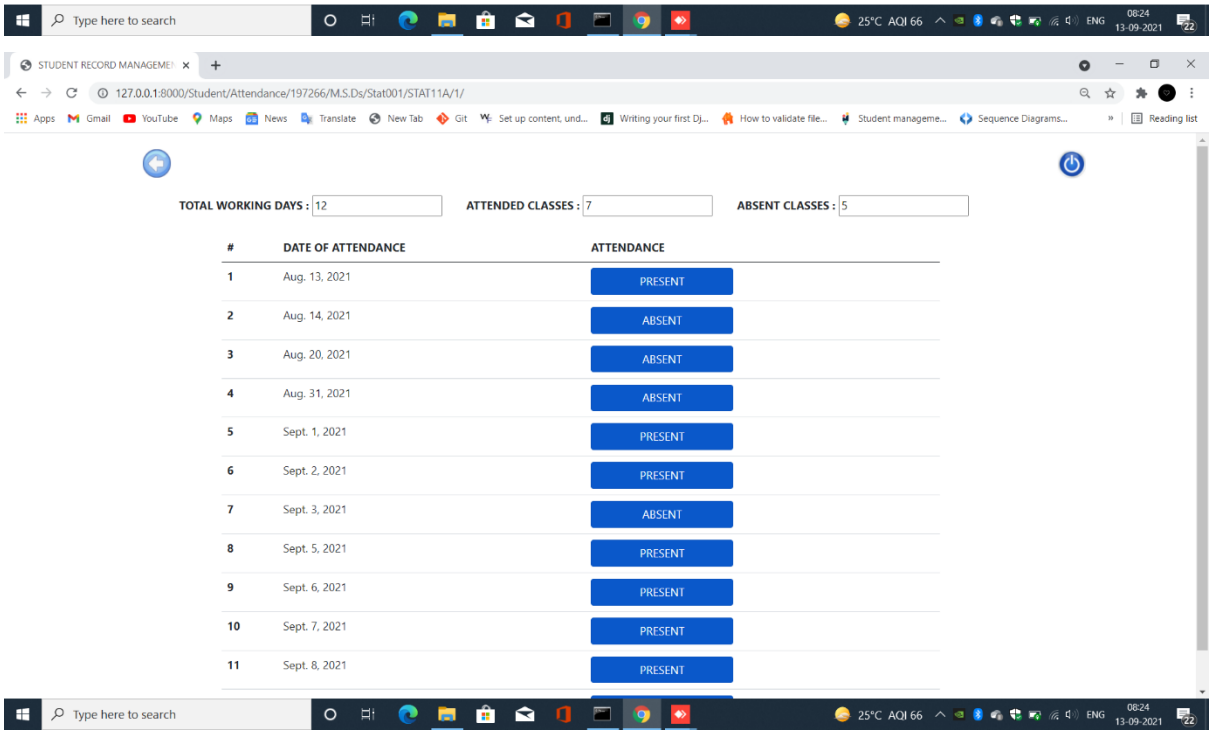




STUDENT

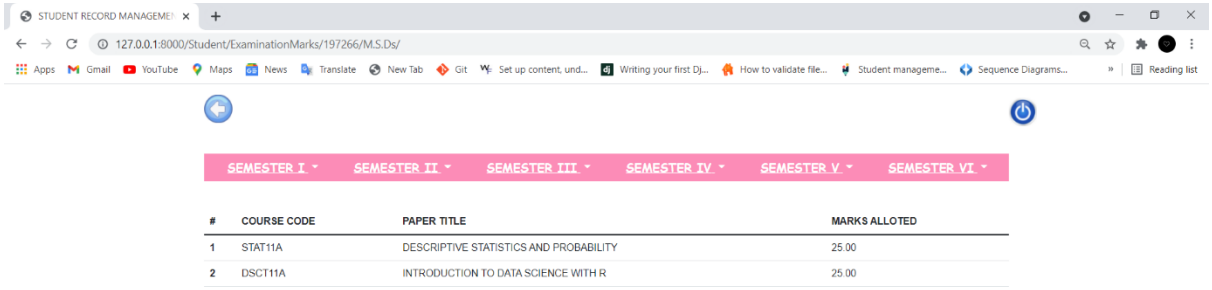
STUDENT DETAILS AND ATTENDANCE CHECK PAGE

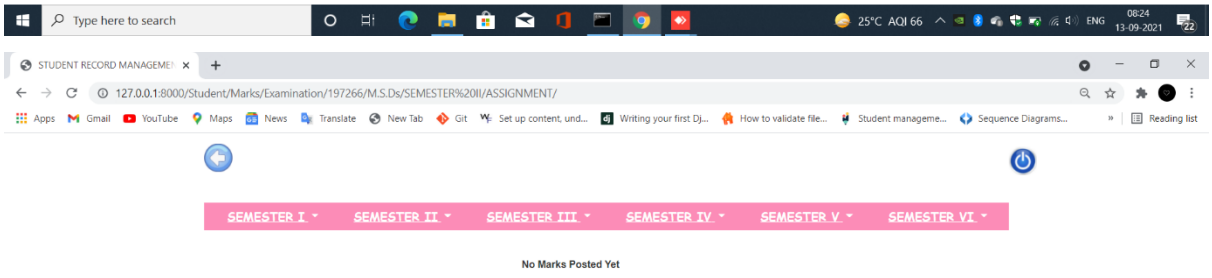




STUDENT

EMPTY ATTENDANCE AND SUBJECT ATTENDACE PAGE





SEMESTER MARKS PAGE

10. TECHNOLOGIES USED

10.1 TOOLS

10.1.1 SUBLIME TEXT

SUBLIME TEXT is a commercial source code editor. Natively there are too many programming languages and labelling languages. Users can expand their functionality with the add-ons, usually supplemented and kept licenses in free software. To facilitate the add-ons, sublime text properties.

10.1.2 GIT HUB

The main goal of GitHub.com is to make the version control and issue tracking aspects of software development easier. Tags, milestones, responsibilities and a search engine are available for problem tracking. For version control, Git (and, as an extension, GitHub.com) allows pull requests to suggest changes to the source code. Users with the ability to review proposed changes can see a difference to the requested changes and approve them. In Git terminology, this action is called a "commit" and one instance of it is "commit". A history of all confirmations is kept and can be viewed later.

10.1.3 ADOBE XD

Adobe XD is a vector-based user experience design tool developed and published by Adobe Inc. for web and mobile applications. It's available for macOS and Windows, but there are also iOS and Android versions that let you preview your work right on mobile devices. You can use Adobe XD to design websites and prototype clicks.

10.2 TECHNOLOGIES FOR FRONT-END

- HTML
- CSS
- JAVASCRIPT
- BOOTSTRAP

10.3 TECHNOLOGIES FOR BACK-END

- PYTHON
- DJANGO

11. TESTING

11.1 TYPES OF TESTING

1. Unit Testing It fixates on the most diminutive unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by utilizing sample input and visually examining its corresponding outputs.

2. Integration Testing The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is coalesced to engender output. Integration testing is of four types: (i) Top-down (ii) Bottom-up (iii) Sandwich (iv) Sizably voluminousBang Example: (a) Ebony Box testing: - It is utilized for validation. In this we ignore internal working mechanism and focuses on what is the output? (b) White Box testing: - It is utilized for verification. In this we fixate on internal mechanism i.e. how the output is achieved? 3. Regression Testing Every time an incipient module is integrated leads to vicissitudes in the program. This type of testing ascertains that the whole component works congruously even after integrating components to the consummate program.

4. Smoke Testing This test is done to ascertain that software under testing is yare or stable for further testing It is called a smoke test as the testing an initial pass is done to check if it did not catch the fire or smoke in the initial switch on.

5. Alpha Testing This is a type of validation testing. It is a type of acceptance testing which is done afore the product is relinquished to customers. It is typically done by QA people.

6. Beta Testing The beta test is conducted at one or more customer sites by the cessation-utilizer of the software. This version is relinquished for a circumscribed number of users for testing in an authentic-time environment

7. System Testing This software is tested such that it works fine for the different operating systems. It is covered under the ebony box testing technique. In this, we just fixate on the required input and output without fixating on internal working. In this, we have security testing, instauration testing, stress testing, and performance testing

8. Stress Testing In this, we give inauspicious conditions to the system and check how they perform in those conditions.

9. Performance Testing It is designed to test the run-time performance of software within the context of an integrated system. It is utilized to test the haste and efficacy of the program. It is withal called load testing. In it we check, what is the performance of the system in the given load.

10. Object-Oriented Testing This testing is a coalescence of sundry testing techniques that avail to verify and validate object-oriented software. This testing is done in the following manner: Testing of Requirements, Design and Analysis of Testing, Testing of Code, Integration testing, System testing, User Testing

11.2 TEST CASES

11.2.1 MOUDULE FACULTY

S.NO	*TITLE*	*EXPECTED OUTPUT*	*ACQUIRED OUTPUT*	*RESULT*
1	Faculty login	Should not allow user with incorrect login details	The user with invalid login credentials is not allowed	PASS
2	Add Student	The student must be added with unique Roll Numbers	Students with the same roll numbers are not allowed	PASS
3	Student Image	The student image size must less than 300kb	The images with size more than 300kb are not allowed	PASS
4	Edit Student	All the validations must satisfy	The student details with wrong input are not updated	PASS
5	Delete student	The Student must be	The student is deleted and the	PASS

		deleted on clicking delete	page is redirected to class page	
6	Certificates	All the student certificates must be displayed in certificates page	The student documents are displayed	PASS

11.2.2 MODULE TEACHER

S.NO	*TITLE*	*EXPECTED OUTPUT*	*ACQUIRED OUTPUT*	*RESULT*
1	Teacher Login	The user with invalid login details is not allowed	Invalid login details page is displayed	PASS
2	Teacher class page	All the class list should be displayed	The class list is displayed	PASS
3	Attendance	The Attendance should not submit without proper date input	The Attendance is not submitted	PASS
4	Update Attendance	The page should have the previous values displayed	The attendance updated is submitted	PASS

5	Delete Attendance	All the attendance on that particular date should be removed	The attendance is deleted	PASS
6	Attendance Report	The Attendance report should be displayed along with student attendance rate	The Attendance report is displayed	PASS
7	Attendance marks submission	The attendance marks should be submitted via report page	The attendance is submitted	PASS
8	Update attendance marks	The marks must be updated every time the attendance marks are submitted	The attendance marks are updates	PASS
9	Examination marks	The student marks should be null when he/she is absent for the exam	The student marks are null	PASS
10	Update marks	The marks must be updated via update pages	The marks are updated	PASS

		with proper details		

11.2.3 MODULE STUDENT

S.NO	*TITLE*	*EXPECTED OUTPUT*	*ACQUIRED OUTPUT*	*RESULT*
1	Student Login	The user with invalid login details is not allowed	Invalid login details page is displayed	PASS

12. CONCLUSION

The "STUDENT RECORD MANAGEMENT SYSTEM" helps college personnel to modify the hand-worked understudy record framework with the mechanized program, which works on the accommodation of keeping up with understudy data. The primary functionalities contain embeddings and controlling the understudy subtleties alongside recovering the information regardless of when. This framework additionally upholds in safely putting away the non-excess information for extraordinary periods of time. Understudy record framework helps the client in fast administration of the data with an easy to understand interface gave. An extraordinary number of foundations and associations are embracing these, rather than pen and desk work frameworks, for the improvement of coordinated records and order over the information. Nonetheless, the framework grew, predominantly centres around accomplishing the necessities of our school.

MODULES COMPLETED

- Access to non-teaching faculty
- Adding a new student
- Deleting an existing student
- Updating student details
- Display of students via classes
- Access to teaching faculty
- Viewing the students list
- Marking attendance
- Updating and deleting existing attendance
- Allotment of marks
- Updating and deleting marks
- Report for attendance
- Report for marks
- Access to student
- Student details
- Student checking marks and attendance report
- Sending e mail after student registration

FUTURE SCOPE

- The examinations schedules can be changed depending on the needs.
- The student documents can be printed.
- The users can change their login credentials.
- The teacher can choose the classes for the semester irrespective of the admin permission.
- The student can submit their details in the process of registration.
- Part 4 details can also be included.

13. BIBILIOGRAPHY

[1] R Ramakrishnan, J Gehrke, J Gehrke, “Database management systems” - 2003

[2] M. Atkinson, "The Object-Oriented Database System Manifesto", Proc. First Int'l Conf. Deductive and Object-Oriented Databases, pp. 40-57, 1989.

[3] T. Andrews and C. Harris, "Combining Language and Database Advances in Object-Oriented Development Environment", Proc. Object-Oriented Programming Systems Languages and Applications, pp. 430-440, 1987.

[4] “Web Based Students Record Management System for Tertiary Institutions” Department of Computer Science, Imo State University, Owerri, Nigeria Vol. 6, Issue 6, June 2019.

[5] Zhibing Liu, H.W., HuiZan, “Design and implementation of student information management system”, International symposium on intelligence information processing and trusted computing, 2010.

[6] Zhi-gang YUE, You-wei JIN, “The development and design of the student management system based on the network environment”,2010 International Conference on Multimedia Communications, 978-0-7695-4136-5/10 2010 IEEE.

[7] J. M. Hellerstein, M. Stonebraker, J. Hamilton, “Architecture of a Database System,” Journal Foundations and Trends in Databases, USA, Vol 1, Issue 2, pp 141-259, 2007

[8] Almahdi Alshareef, Ahmed Alkilany "Toward a Student Information System for Sebha University, Libya",Fifth international conference on Innovative Computing Technology (INTECH 2015)p 34

[9] Prabhu 39 T Kannan, Srividya K Bansal,"Unimate: A Student Information System",2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)p12511256

[10] System and method for communicating student information among student, parents’ guardians and educators. (US 20060127870 A1)

[11] TANG Yufang, ZHANG Yongbased on the web services”. Natural Science Foundation of Shandong Province(Y2008G22), 978 n Estimation, sheng,“Design and implementation of college student information manageme142443930nt system 0/09 2009 IEEE.

[12] Documentation on developing Django web applications.

[13] Python Programming |A modular approach | First Edition | By Pearson by Taneja Sheetal and Kumar Naveen

[14] International Journal of Advanced Research in Science, Engineering and Technology Vol. 6, Issue 6 , June 2019 Web Based Students' Record Management