COLLEGE ADMINISTRATION SYSTEM PROJECT REPORT

Abstract

The main aim of this project is to tackle the problem of attendance management, marks management and fee management.We provide a system which simplifies attendance activity of students through graphical user interface which can be used to mark attendance of students rather than regular register based system.Student marks of various semesters are collected at one place for better analysis rather than using regular reports. Fee activity of students are managed by a separate portal associated with the platform in which the balances of students can be seen.In addition to this we provide student accounts for checking attendance and marks on a regular basis. We also provide admin account and college staff accounts.In this way we can be able to manage college administration problems.

Contents **Page No**

List Of Figures

List Of Tables

1.Introduction

1.1 Background

As most of the work is done manually or it is based on paper work such as class attendance , marks entering , fee balance etc. These all process takes time . If we include all the work which will be based on online system , then it can reduce time and work. For example -if any teacher have to query whether all students are present in their class , then he/she have to count manually. But under this system all the details will be processed by a single click on its user screen. A student can analyse his/her marks report of various semesters at one place.

1.2 Problem Statement

The main objective of College Administration System is to automate all functionalities of a college . Using this system you can manage all college management work like fees checking , attendance checking and analysing marks . Using this College Administration System you can view or update data and information about students and staff easily. Admin can also retrieve information of employees and students.

1.3 Objectives

The College Administration System can be used to store student information like attendance, fees, and student result etc. admin can create report regarding any student any time using this system. Using this system you can register new student and their course details. You can submit students fees and can check fees details anytime. You can create exam result and submit in this system. Student can check their result online by logging to the system. You can also add new employee in the system and can check details of the employee easily. Student can also check course detail online from this system.

- College Administration System provides the easiest way to manage all functionalities of a college. This system facilitates colleges to maintain the functionality related to college employees and their students.
- College Administration System can store and manage all data of the various departments of a college like Attendance, Staff details etc. using this system user can retrieve any information related to student, teacher and fees.
- Using this system teacher can check student attendance anytime. This system also help teacher to announce the result. College administration can also manage college work easily.

Chapter 2

SYSTEM ANALYSIS

Requirements Model

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications.

Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

User requirements

1. Execution time should be fast
2. More Accurate
3. User Friendly

2.1 Hardware requirements

1. SYSTEM : Pentium Dual Core
2. HARD DISK : 120 GB
3. MONITOR
4. MOUSE
5. RAM : 2 GB or more

2.2 Software requirements

1. O/S : Windows
2. Language : Python3
3. Django

2.3 Functional Requirements Input:

Functional requirements describe the interaction between the system and its environment independent of its implementation.

Admin: He can maintain the staff and student's records, add/update/delete staff or student records in the database.

Staff: The staff can login and can enter the attendance and marks of all students in the class.

Student: The student can login and can see his attendance , marks of all semesters , fee balance.

Non-Functional Requirements

1. Execution qualities:

a. Efficiency:

The state or quality of being efficient, i.e., able to accomplish something with the least waste of time and effort; competency in performance.

2. Evolution qualities:

a. Testability: The means by which the presence, quality, or genuineness of anything is determined.
b. Extensibility: To enlarge the scope of, or make more comprehensive, as operations, influence etc.
c. Scalability: The ability of something, especially a computer system, to adapt to increased demands.

2.4 UML Diagrams for the project work

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.
m The elements are like components which can be associated in different ways to make a complete UML picture, which is known as diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real-life systems.

Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagrams are not a new concept but it is used widely in different forms in different industries.

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. UMLwas created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 19941996, the UML was developed by three software engineers working at Rational Software.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide

both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency. Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases. Together this use case collection specifies all the ways the system. An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is anassociation. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency. Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases.

By default, the association tool on the toolbox is unidirectional and drawn on a diagram with a single arrow at one end of the association. The end with the arrow indicates who or what is receiving the communication. A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Typically, on class diagrams, a dependency relationship indicates that the operations of the client invoke operations of the supplier. The work flow in this case begins from importing the dataset by the developer and then replacing missing values with mean value of corresponding column ,model building, validating that model by generating a confusion matrix and finally predicting the test sample class label. Transitions are used to show the passing of the flow of control from activity.

The various UML diagrams are

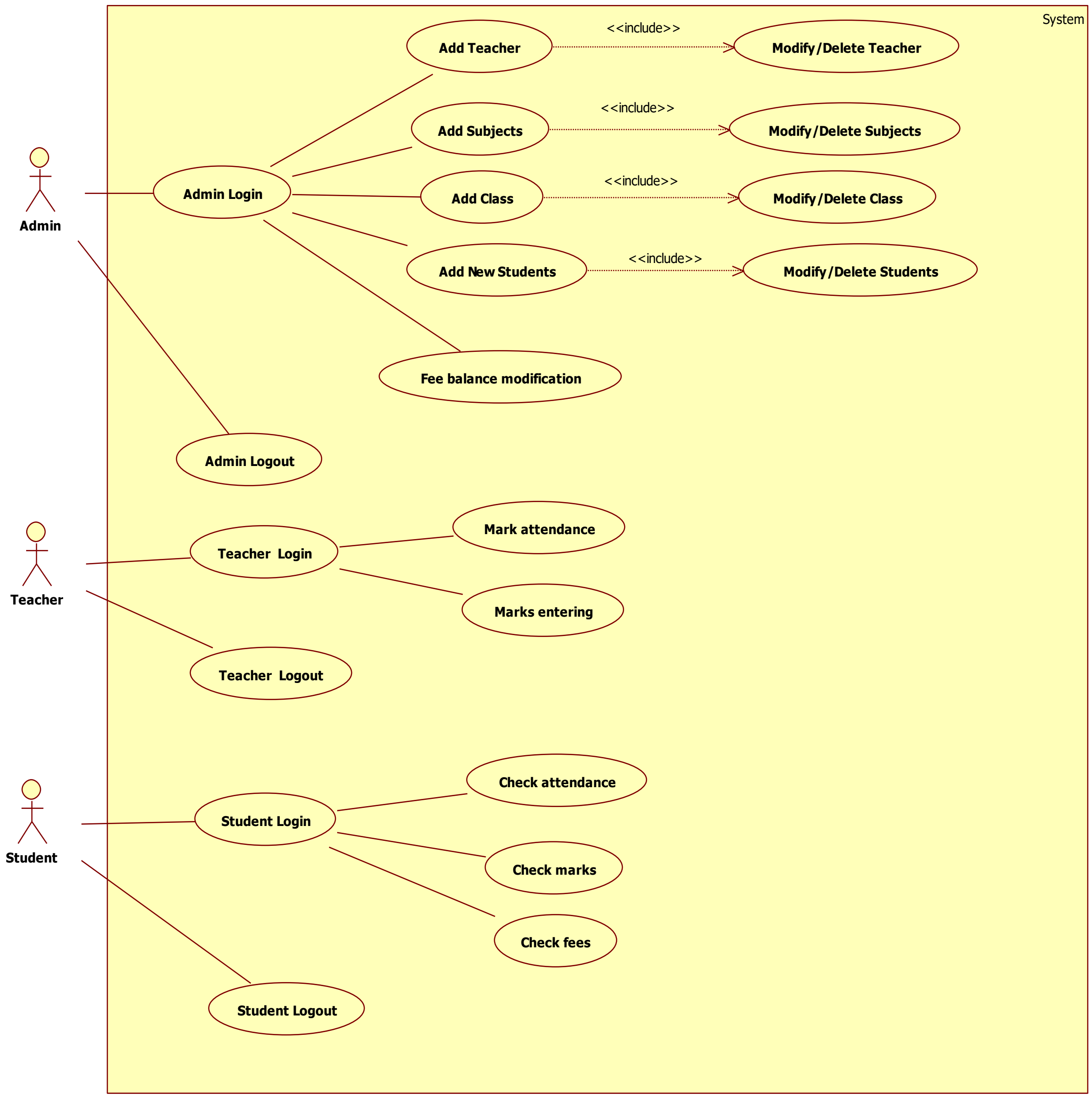
1. Usecase diagram
2. Activity diagram
3. Sequence diagram
4. Colloboration diagram
5. Object diagram
6. State chart diagram
7. Class diagram
8. Component diagram
9. Deployment diagram

Usecase Diagram

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior. Actors are not part of the system. Actors represent anyone or anything that interacts with (input to orreceive output from) the system. Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented. Use case is a sequence of transactions performed by a system that yields a measurable result of values for a particular actor. The use cases are all the ways the system may be used.

Use case is a list of actions or event steps, typically defining the interactions be- tween a role (known as an actor) and a system, to achieve a goal. In case of the use case diagram developer and the end user are the actors. Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases. Together this use case collection specifies.

An include relationship is a stereotyped relationship that connects a base use case to an inclusion use case. An include relationship specifies how behavior in the inclusion use case is used by the base use case. An extend relationship is a stereotyped relationship that specifies how the functionality of one use case can be inserted into the functionality of another use case. Extend relationships between uses cases are modeled as dependencies by using the Extend stereotype. The different use cases are import dataset for importing datasets, preprocessing, model building, validation and prediction.



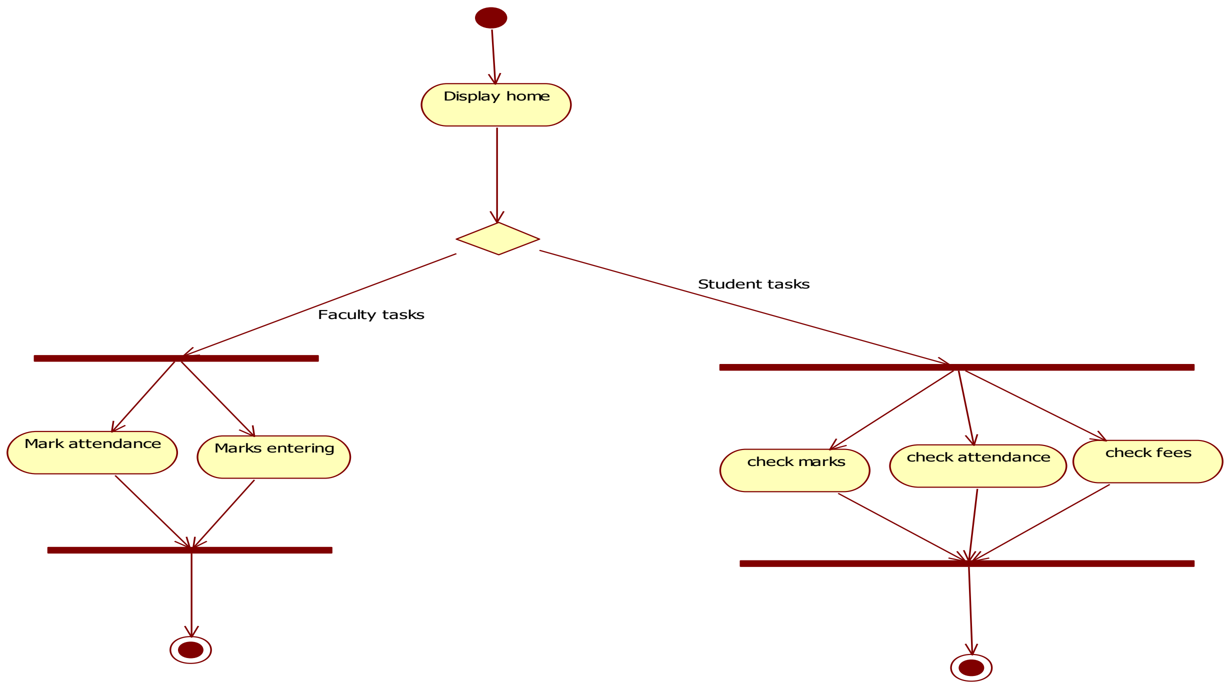
Activity Diagram

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations andthe transitions are triggered by the completion of the operations. The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. Activity diagrams contain activities, transitions between the activities, decision points, and synchronization bars. An activity represents the performance of some behavior in the workflow. In the UML, activities are represented as rectangles with rounded edges, transitions are drawn as directed arrows, decision points are shown as diamonds, and synchronization bars are drawn as thick horizontal or vertical bars as shown in the following. The activity icon appears as a rectangle with rounded ends with a name and a component for actions.

Transition connects activities with other model elements and object flows connect activities with objects. They are typically triggered by the completion of the behavior in the originating activity.

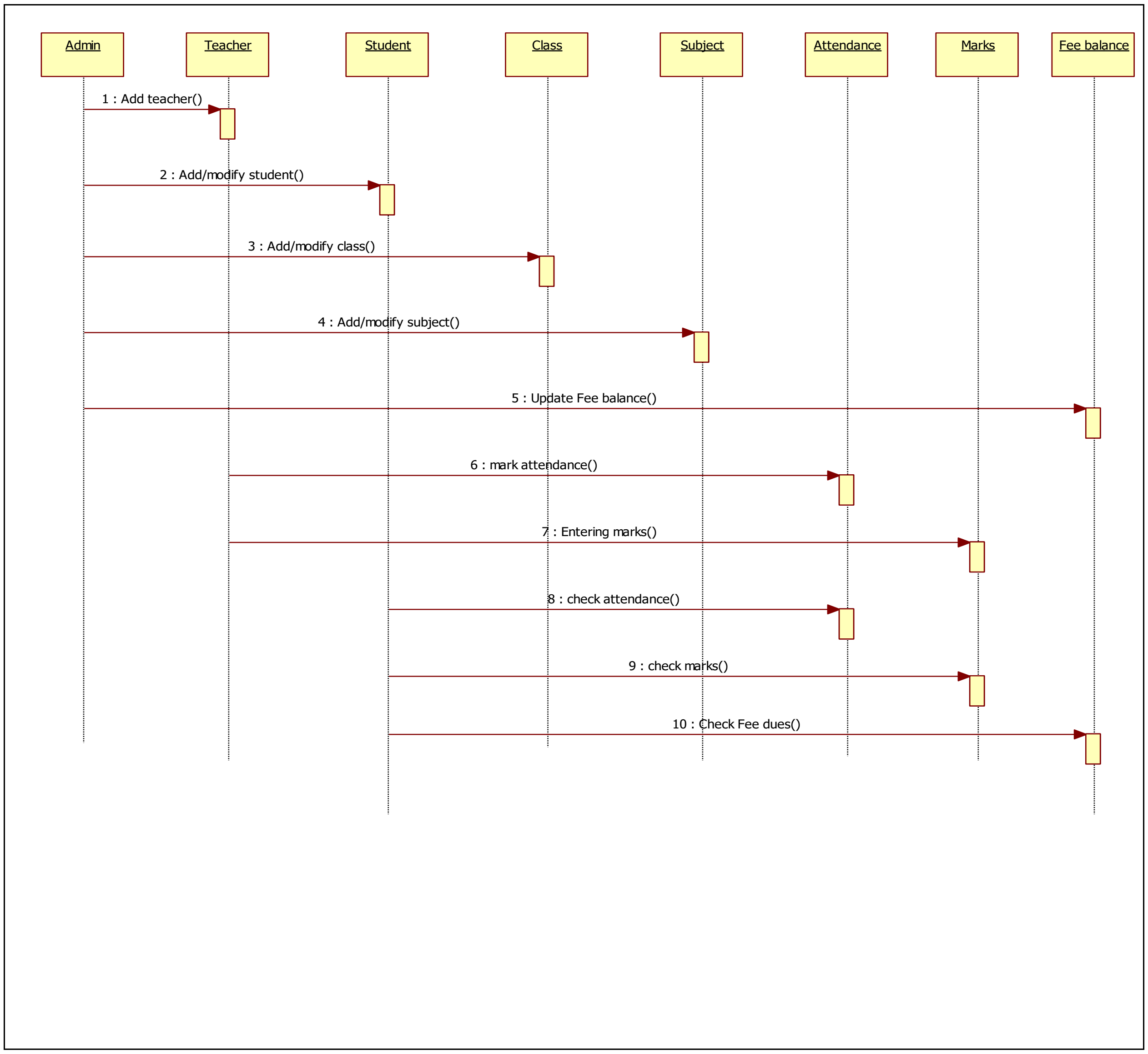
Swim lanes may be used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activities contained in the swim lane. Swim lanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swim lanes are very similar to an object because they provide a way to tell who is performing a certain role. Swim lanes only appear on activity diagrams. When a swim lane is

dragged onto an activity diagram, it becomes a swim lane view. Swim lanes ap- pear as small icons in the browse while a swim lane views appear between the thin, vertical lines with a header that can be renamed and relocated. An activity represents the performance of some behavior in the work flow. In the UML, activities are rep- resented as rectangles with rounded edges, transitions are drawn as directed arrows, decision 17 points are shown as diamonds, and synchronization bars are drawn as thick horizontal or vertical bars as shown in the following. The activity icon appears as a rectangle with rounded ends with a name and a component for actions.



Sequence Diagram

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called as event diagrams.



Class Diagram

Class diagrams contain icons representing classes, interfaces, and the irrelationships. You can create one or more class diagrams to represent the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model. You can also create one or more class diagrams to represent classes contained by each package in your model; such class diagrams are themselves contained by the package enclosing the classes they represent; the icons representing logical packages and classes in class diagrams.

1. Class diagrams are created to provide a picture or view of some or all of the classes in the model.

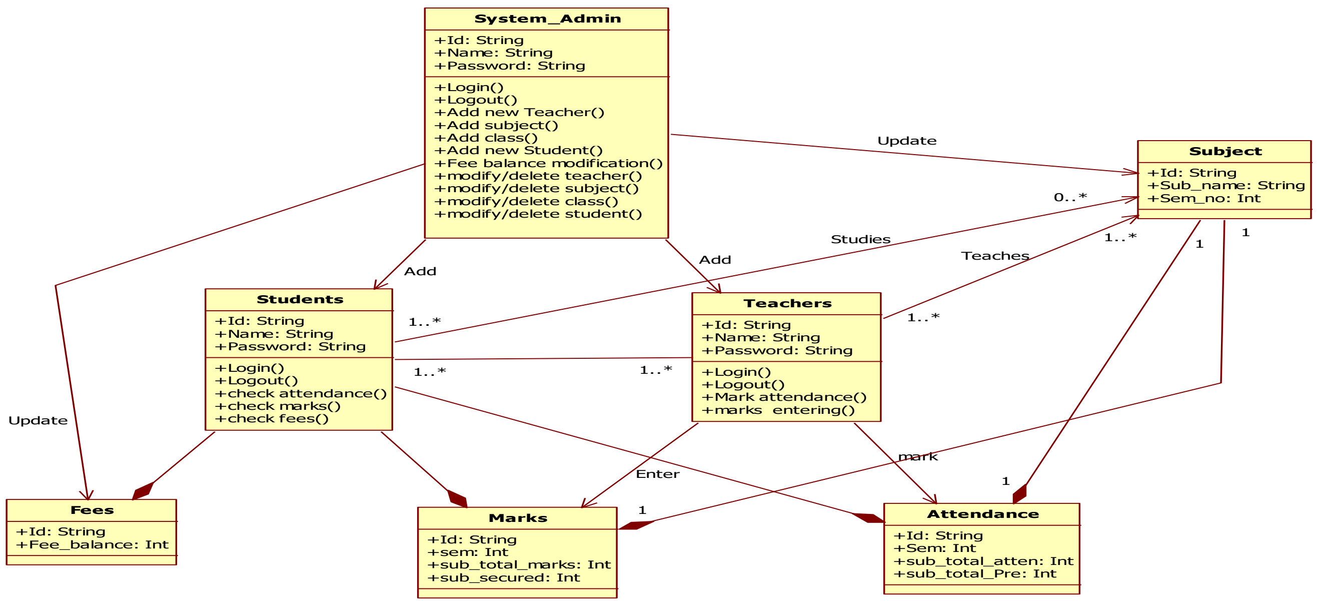
The main class diagram in the logical view of the model is typically a picture of the packages in the' system. Each package also has its own main class diagram, which typically displays the "public" classes of the package.

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models.

A Class is a description of a group of objects with common properties (attrib-utes)common behavior(operations),common relationships to other objects,and com-mon semantics. Thus, a class is a template to create objects. Each object is

an instance of some class and objects cannot be instances of more than oneclass. In the UML, classes are represented as compartmentalized rectangles.

1. The top compartment contains the name of the class.
2. The middle compartment contains the structure of the class(attributes).
3.The bottom compartment contains the behavior of the class(operations).



State Chart Diagram

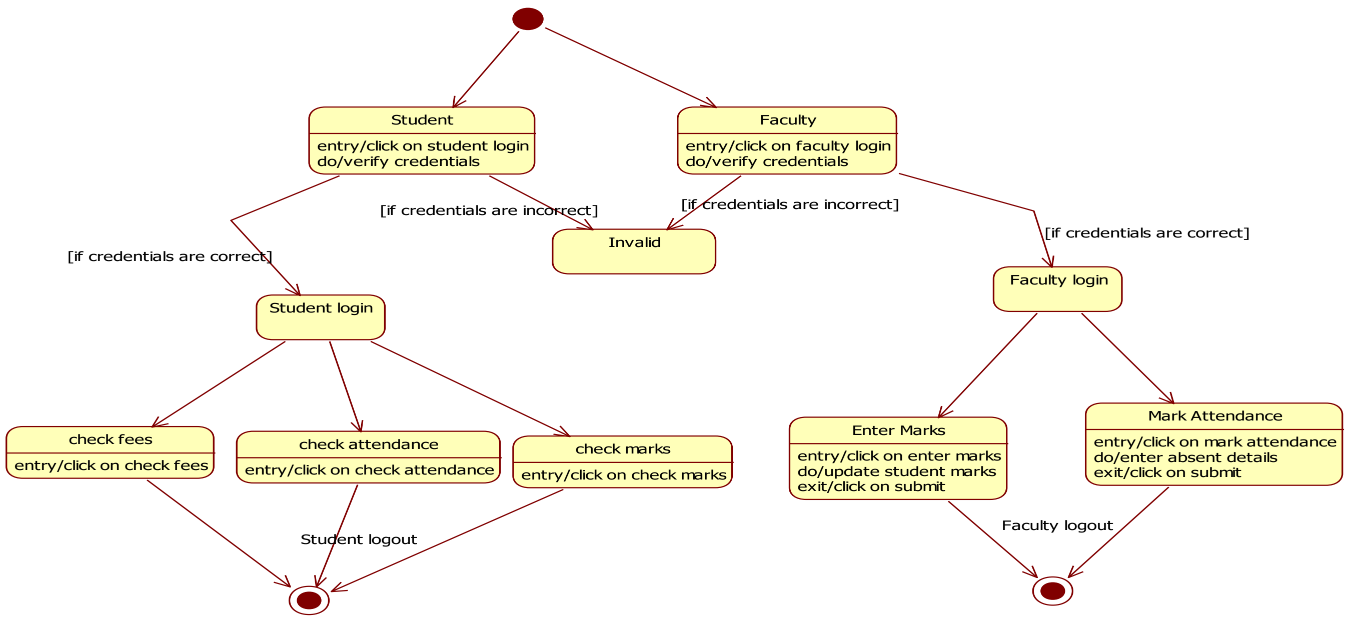
Use cases and scenarios provide a way to describe system behavior; in the form of interaction between objects in the system.Sometime it is necessary to consider inside behavior of an object. A state chart diagram show s the states of a single objects, the events or messages that cause a transition from one state to an other and the actions that result from a state change.As activity diagram,state chart diagram also contains special symbols for start state and stop state.

State chart diagram cannot be created for every class in the system,it is only for those lass objects with significant behavior.

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied.A state transition is a relationship between two states, two activities, orbetween an activity and a state.

We can show one or more state transitions from a state as long as each transition is unique.Transitions originating from a state cannot have the same event, unless there are conditions on the event.

Provide a label for each state transition with the name of at least one eventthat causes the state transition. You do not have to use unique labels for state transitions because the same event can cause a transition to many different states or activities.


Student
entry/click on student login
do/verify credentials
Faculty
entry/click on faculty login
do/verify credentials
[if credentials are incorrect]
[if credentials are incorrect]
Invalid
[if credentials are correct]
[if credentials are correct]
Student login
Faculty login
check fees
entry/click on check fees
check attendance
entry/click on check attendance
check marks
entry/click on check marks
Enter Marks
entry/click on enter marks
do/update student marks
exit/click on submit
Mark Attendance
entry/click on mark attendance
do/enter absent details
exit/click on submit
Student logout
Faculty logout

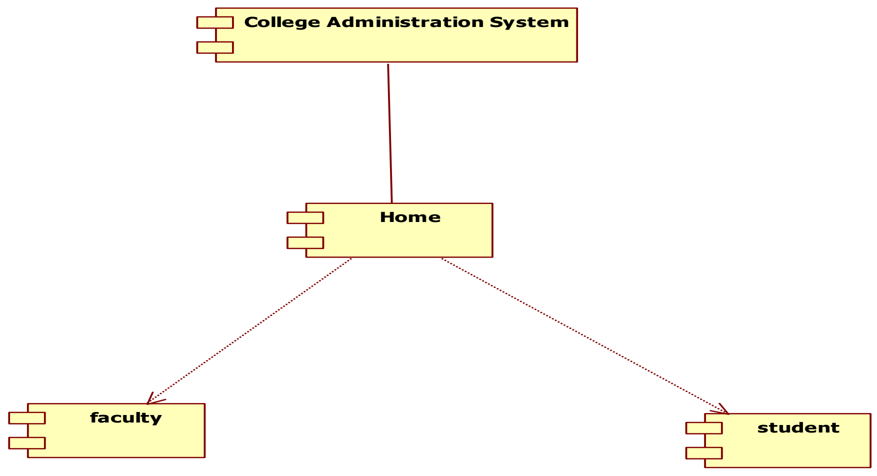
Component Diagram

Component Diagrams show the dependencies between software components in the system. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time or atruntime.

In a large project there will be many files that makeup the system.These files will have dependencies on one another.The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time. There are also dependencies between source code

files and the executable files or bytecode files that are derived from them by com- pilation. Component diagrams are one of the two types of implementation diagram in UML. Component diagrams show these dependencies between software compo- nents in the system. Stereotypes can be used to show dependencies that are specific to particular languages also.

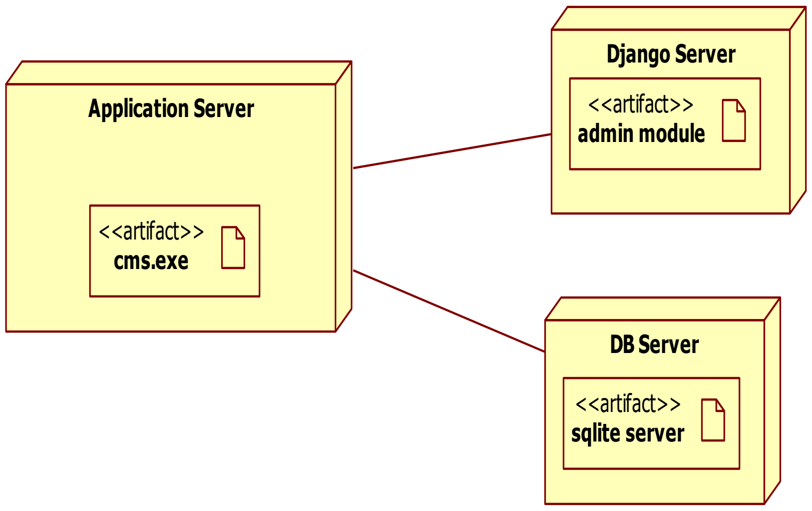
A component diagram shows the allocation of classes and objects to components in the physical design of a system. A component diagram may represent all or part of the component architecture of a system along with dependency relationships. The dependency relationship indicates that one entity in a component diagram uses the services or facilities of another.



Deployment Diagram

The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time processing elements and the software components and processes that are located on them. Deployment diagrams are made up of nodes and communication associations.

Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show other processing resources such as people or mechanical resources. Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes connected by communication associations.



3.System Design

3.1 Architecture

It is a three-tier architecture. A three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

Three-tier architecture is a software design pattern and a well-established software architecture. Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently.

The user interface is implemented on any platform such as a desktop PC, smartphone or tablet as a native application, web app, mobile app, voice interface, etc. It uses a standard graphical user interface with different modules running on the application server.

The relational database management system on the database server contains the computer data storage logic. The middle tiers are usually multitiered. Since the three are not physical but logical in nature, they may run in different servers both in on-premises based solutions, as well as in software-as-a-service (SaaS). it provides great freedom to development teams who can independently update or replace only specific parts of the application without affecting the product as a whole. The application can be scaled up and out rather easily by detaching the front-end application from the databases that are selected according to the individual needs of the customer. New hardware, such as new servers, can also be added at a later time to deal with massive amounts of data or particularly demanding services.

A three-tier-architecture also provides a higher degree of flexibility to enterprises who may want to adopt a new technology as soon as it becomes available.

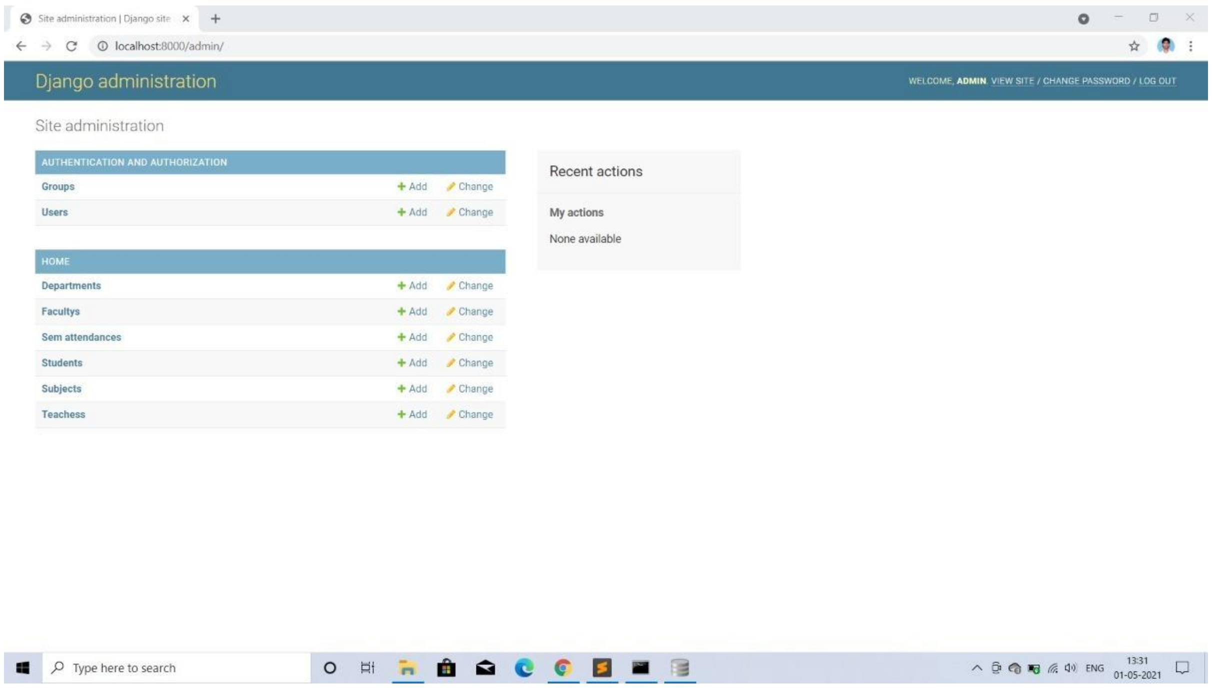
3.2 Modules

Admin module :-

A web-based administrative backend is a standard feature of modern websites. The administrative interface, or admin for short, allows trusted site administrators to create, edit and publish content, manage site users, and perform other administrative tasks.

Django comes with a built-in admin interface. With Django's admin you can authenticate users, display and handle forms, and validate input; all automatically. Django also provides a convenient interface to manage model data.

In this chapter, we will explore the basics of the Django admin—create a superuser login, register models with the admin, customize how our models are viewed in the admin, add and edit model data, and learn how to manage users in the admin.



Site administration | Django site

localhost:8000/admin/

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

HOME		
Departments	+ Add	Change
Facultys	+ Add	Change
Sem attendances	+ Add	Change
Students	+ Add	Change
Subjects	+ Add	Change
Teachess	+ Add	Change

Recent actions

My actions

None available

Type here to search

ENG 13:31 01-05-2021

Student module :-

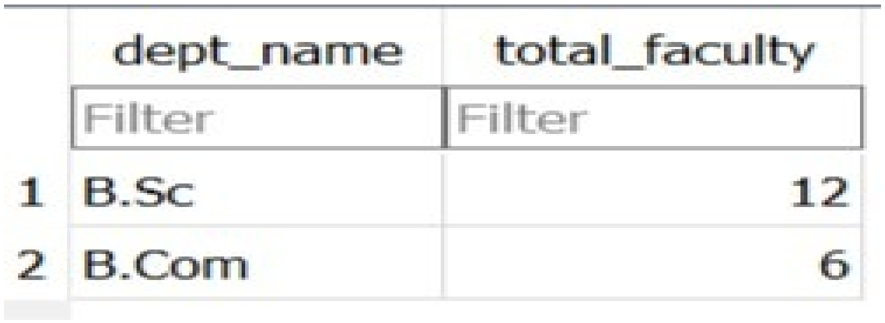
In this , the student can login and he/she can able to check their attendance , analyse their marks , check their fee dues of the semester.

Teacher module :-

In this , the teacher can login and can able to mark the attendance of the students and can enter the semester marks of respective student.

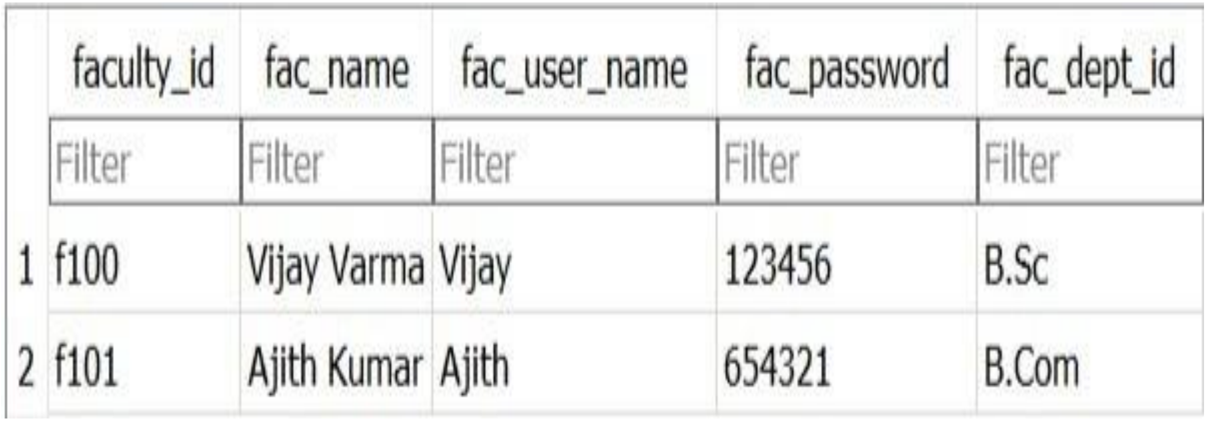
3.3 Database design tables

Department Table



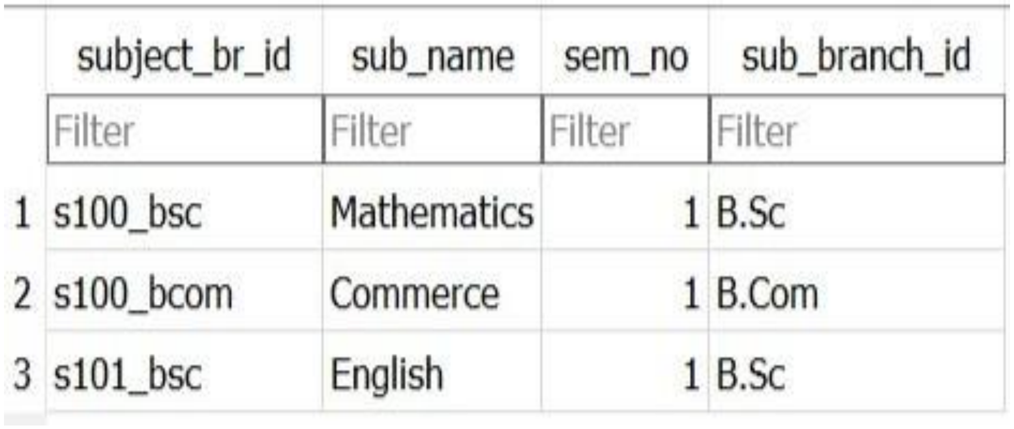
	dept_name	total_faculty
	Filter	Filter
1	B.Sc	12
2	B.Com	6

Faculty Table



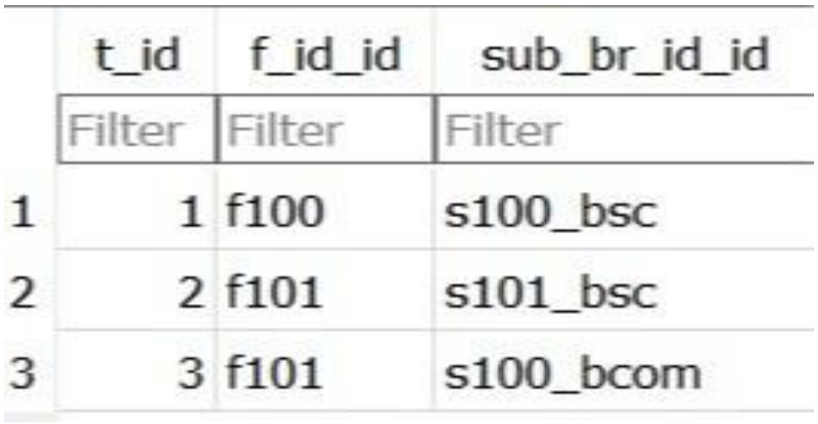
	faculty_id	fac_name	fac_user_name	fac_password	fac_dept_id
	Filter	Filter	Filter	Filter	Filter
1	f100	Vijay Varma	Vijay	123456	B.Sc
2	f101	Ajith Kumar	Ajith	654321	B.Com

Subjects Table



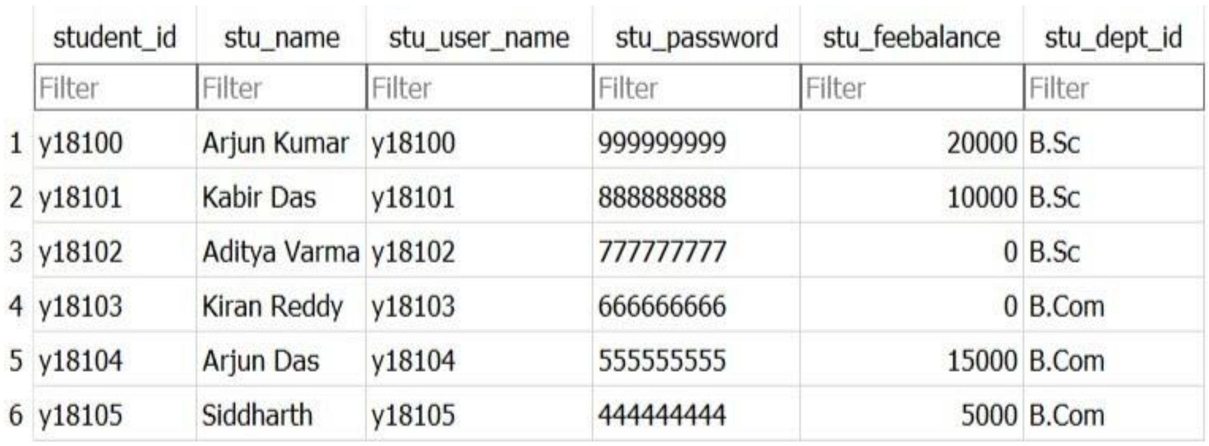
	subject_br_id	sub_name	sem_no	sub_branch_id
	Filter	Filter	Filter	Filter
1	s100_bsc	Mathematics	1	B.Sc
2	s100_bcom	Commerce	1	B.Com
3	s101_bsc	English	1	B.Sc

Teachers Table



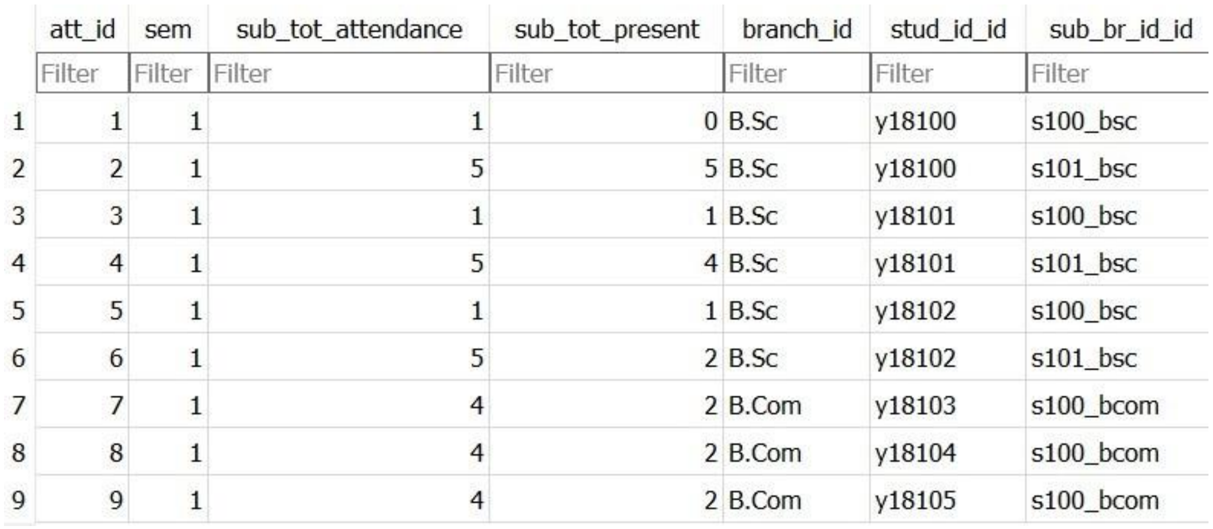
	t_id	f_id_id	sub_br_id_id
	Filter	Filter	Filter
1	1	f100	s100_bsc
2	2	f101	s101_bsc
3	3	f101	s100_bcom

Students Table



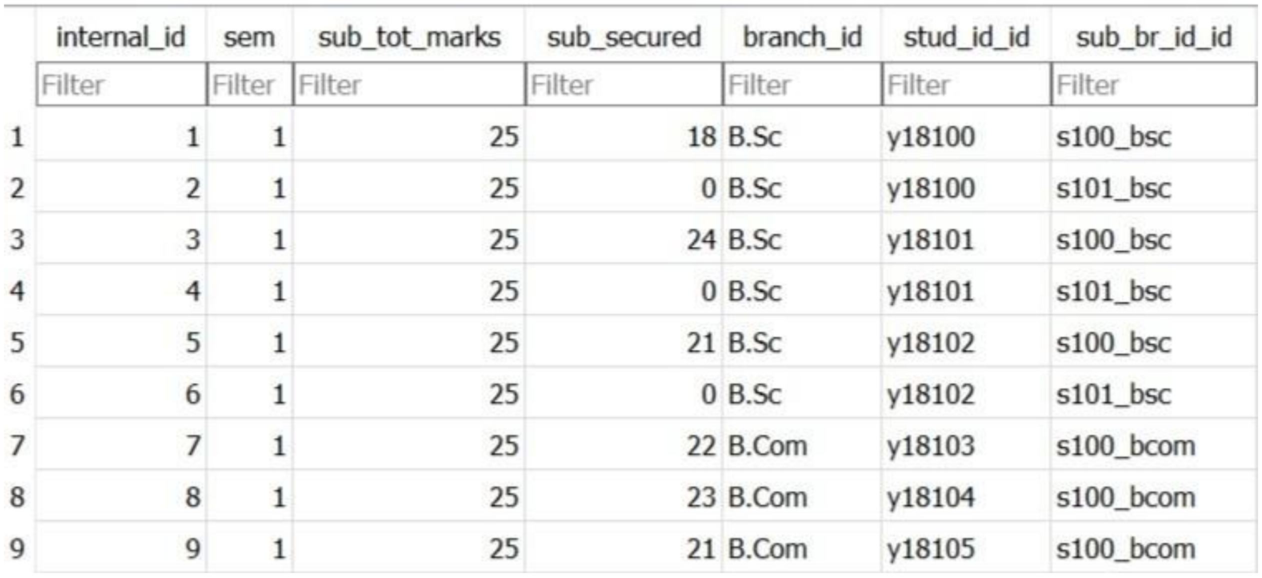
	student_id	stu_name	stu_user_name	stu_password	stu_feebalance	stu_dept_id
	Filter	Filter	Filter	Filter	Filter	Filter
1	y18100	Arjun Kumar	y18100	999999999	20000	B.Sc
2	y18101	Kabir Das	y18101	888888888	10000	B.Sc
3	y18102	Aditya Varma	y18102	777777777	0	B.Sc
4	y18103	Kiran Reddy	y18103	666666666	0	B.Com
5	y18104	Arjun Das	y18104	555555555	15000	B.Com
6	y18105	Siddharth	y18105	444444444	5000	B.Com

Sem Attendance Table



	att_id	sem	sub_tot_attendance	sub_tot_present	branch_id	stud_id_id	sub_br_id_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	0	B.Sc	y18100	s100_bsc
2	2	1	5	5	B.Sc	y18100	s101_bsc
3	3	1	1	1	B.Sc	y18101	s100_bsc
4	4	1	5	4	B.Sc	y18101	s101_bsc
5	5	1	1	1	B.Sc	y18102	s100_bsc
6	6	1	5	2	B.Sc	y18102	s101_bsc
7	7	1	4	2	B.Com	y18103	s100_bcom
8	8	1	4	2	B.Com	y18104	s100_bcom
9	9	1	4	2	B.Com	y18105	s100_bcom

Internals Table



	internal_id	sem	sub_tot_marks	sub_secured	branch_id	stud_id_id	sub_br_id_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	25	18	B.Sc	y18100	s100_bsc
2	2	1	25	0	B.Sc	y18100	s101_bsc
3	3	1	25	24	B.Sc	y18101	s100_bsc
4	4	1	25	0	B.Sc	y18101	s101_bsc
5	5	1	25	21	B.Sc	y18102	s100_bsc
6	6	1	25	0	B.Sc	y18102	s101_bsc
7	7	1	25	22	B.Com	y18103	s100_bcom
8	8	1	25	23	B.Com	y18104	s100_bcom
9	9	1	25	21	B.Com	y18105	s100_bcom

4.Implementation

4.1 Code

```
home.html
-----------------------
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Home</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  <style type="text/css">
        footer {
          position: fixed;
          left: 0;
     bottom: 0;
     width: 100%;
     background-color: #fedcba;
     padding: 25px;
    }
     .jumbotron {
     background-color: #abcdef;
     margin-bottom: 50px;
    }
     .row.content {height: 450px}
    @media screen and (max-width: 767px) {
     .row.content {height:auto;}
    }
     .button {
     background:#fdb531;
     box-shadow:20px 20px 0 rgba(0,0,0,.5);
     display:inline-block;
     font-size:2em;
     padding:.5em 2em;
     text-decoration:none;
     width:100%;
     padding-top: 150px;
     padding-bottom: 150px;
     border-top: 50 px;
    }

    .button:hover {
     box-shadow: 0 0 0 ;
    }

    .parallelogram{
     transform: skew(-20deg);
    }

  </style>
</head>
<body>
        <div class="jumbotron">
     <div class="container text-center">
       <h1>College Name</h1>
       <p>Mission, Vission & Values</p>
     </div>
    </div>
    <div class="container-fluid text-center">
      <div class="row content">
```

```
<div class="col-sm-2">
        </div>
        <div class="col-sm-4 text-center">
          <div class="container-fluid">
                      <a class="button parallelogram" href='{% url "faclog" %}'>
                                  Faculty Login
           </a>
          </div>
        </div>
        <div class="col-sm-4 text-center">
          <div class="container-fluid">
                       <a class="button parallelogram" href='{% url "studlog" %}'>
                                  Student Login
            </a>
          </div>
        </div>
        <div class="col-sm-2">
        </div>
      </div>
    </div>

    <footer class="container-fluid text-center">
      Contact Us: 999999999
    </footer>

</body>
</html>


faclog.html
------------
<!DOCTYPE html>
<html>
<head>
         {% block title%}
         <title>Faculty Login</title>
         {% endblock %}
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

```
<style type="text/css">
        footer {
          position: fixed;
          left: 0;
     bottom: 0;
     width: 100%;
     background-color: #fedcba;
     padding: 25px;
   }
    .jumbotron {
     background-color: #abcdef;
     margin-bottom: 50px;
   }
   body {font-family: Arial, Helvetica, sans-serif;}
   form {border: 3px solid #f1f1f1;}

   input[type=text], input[type=password] {
    width: 100%;
    padding: 12px 20px;
    display: inline-block;
    border: 1px solid #ccc;
    box-sizing: border-box;
   }
   button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
   }

   button:hover {
    opacity: 0.8;
   }
   .container {
    padding: 16px;
   }

   </style>
</head>
<body>
   <div class="jumbotron">
```

```
     <div class="container text-center">
        <h1>College Name</h1>
        <p>Mission, Vission & Values</p>
      </div>
    </div>
    {%  block content %}
    <h1 class="text-center">Faculty Login</h1>
    <form action="{% url 'faclog' %}" method="POST">
    {% endblock %}
        {% csrf_token %}
          <div class="container">
         <label for="uname"><b>Username</b></label>
         <input type="text" placeholder="Enter Username" name="uname" required>

         <label for="psw"><b>Password</b></label>
         <input type="password" placeholder="Enter Password" name="psw" required>

         <button type="submit">Login</button>

        </div>
    </form>

    <footer class="container-fluid text-center">
        Contact Us: 999999999
    </footer>
  </body>
  </html>

  studlog.html
  -------------
  {% extends 'faclog.html' %}
  {% block title %}
  <title>Student Login</title>
  {% endblock %}
  {% block content %}
  <h1 class="text-center">Student Login</h1>
  <form action="{% url 'studlog' %}" method="POST">
{% endblock %}

  facultyduties.html
  -------------------
  <!DOCTYPE html>
  <html lang="en">
  <head>
```

```
<title>Faculty Duties</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  <style>
     body {font-family: Arial, Helvetica, sans-serif;
           background-color: lightblue;}
     p {
        font-size:50px;
        color: red;
       }
     h1{ font-family: American Typewriter, serif }

     u{ color: blue;}
     .row.content {height: 450px}
    @media screen and (max-width: 767px) {
      .row.content {height:auto;}
    }
     .button {
      background:#fdb531;
      box-shadow:20px 20px 0 rgba(0,0,0,2);
      display:inline-block;
      font-size:2em;
      padding:.5em 2em;
      text-decoration:none;
      width:100%;
      padding-top: 150px;
      padding-bottom: 150px;
      border-top: 50 px;
      font-family: URW Chancery L, cursive;
    }

    .button:hover {
      box-shadow: 0 0 0 ;
    }

    .ellipse{
     background: #9fb325;
     border-radius: 50%;
    }</style>
```

```
</head>
<body>
  <p class="text-center"> Welcome {{data.fac_user_name}}</p><br>
  <div class="container">
  <a class="btn navbar-btn btn-danger navbar-right pull-right" role="button" href="{% url
'center' %}">Logout</a>
  </div>
  <h1 class="text-center"> <u>Duties</u> </h1>
  <div class="container-fluid text-center">
      <div class="row content">
        <div class="col-sm-2">
        </div>
        <div class="col-sm-4 text-center">
          <div class="container-fluid">
                      <a class="button ellipse" href="{% url 'choose' data.faculty_id %}">
                                    <i>Attendance Marking</i>
           </a>
          </div>
        </div>
        <div class="col-sm-4 text-center">
          <div class="container-fluid">
                       <a class="button ellipse" href="{% url 'choose1' data.faculty_id %}">
                                    <i>Marks Entering</i>
            </a>
          </div>
        </div>
        <div class="col-sm-2">
        </div>
      </div>
    </div>
</body>
</html>
```

choose.html

```
------------
<!DOCTYPE html>
<html>
<head>
<title>Choose</title>
<meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
   <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
   <style>
   body {font-family: Arial, Helvetica, sans-serif;
        background-color: #09febd;}
   p {
       font-size:50px;
       color: red;
      }
      .cj{
      text-align: center;
      font-size:50px;
      }
      .kl{
      font-size:30px;
      background: black;
      color: white;
      }
      .vk{
      font-size:25px;
      }
   </style>
</head>
<body>
{% block cd %}
<p class="text-center">Choose the subject to mark the attendance</p><br>
{% endblock %}
<div class="container">
<div class="dropdown cj">
  <button class="btn btn-default dropdown-toggle kl" type="button" id="menu1" data-
toggle="dropdown">Subjects
  <span class="caret"></span></button>
  <ul class="dropdown-menu" role="menu" aria-labelledby="menu1">
  {% for i in data %}
   {% block content %}
   <li class="vk" role="presentation"><a role="menuitem" href="{% url 'markattend'
i.sub_br_id_id %}">{{i.sub_br_id_id}}</a></li>
   {% endblock %}
   {% endfor %}
  </ul>
</div>
</div>
</body></html>
```

Markattend.html

```
<!DOCTYPE html>
<html>
<head>
  {% block title %}
        <title>Mark Attendance</title>
  {% endblock %}
        <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <style>
      body {font-family: Arial, Helvetica, sans-serif;
           background-color: lightyellow;}
      p {
        font-size:50px;
        color: red;
        }
      button {
     background-color: #4CAF50;
     color: white;
     padding: 14px 20px;
     margin: 8px 0;
     border: none;
     cursor: pointer;
     width: 100%;
    }
    .dark{
          font-size:20px;
          background:black;
          color:white;
        }
    .cd{
          font-size: 20px;
          color: red;
        }

    </style>
</head>
<body>
```

```
{% block content %}
<p class="text-center">Mark Attendance</p><br>
<div class="container">
<form action="{% url 'markattend' data.1.sub_br_id_id %}" method="POST">
{% csrf_token %}
<table class="table table-bordered table-striped">
  <thead class="dark">
    <tr>
      <th scope="col">Student_Id</th>
      <th scope="col">Present</th>
      <th scope="col">Absent</th>
    </tr>
  </thead>
  <tbody class='cd'>
        {% for i in data %}
    <tr>
      <th scope="row">{{i.stud_id_id}}</th>
      <td>
        <label class="radio-inline">
        <input type="radio" name={{i.stud_id_id}} value="Present" checked>
        Present
        </label>
      </td>
      <td>
        <label class="radio-inline">
        <input type="radio" name={{i.stud_id_id}} value="Absent">
        Absent
        </label>
      </td>
    </tr>
    {% endfor %}
  </tbody>
</table>
{% endblock %}
<button type="submit">Submit</button>
</form>
</div>
</body>
</html>
```

choose1.html

```
-----------------------
{% extends 'choose.html' %}
{% block cd %}
```

```
<p class="text-center">Choose the subject to enter the marks</p><br>
{% endblock %}
{% block content %}
<li class="vk" role="presentation"><a role="menuitem" href="{% url 'entermarks' i.sub_br_id_id
%}">{{i.sub_br_id_id}}</a></li>
{% endblock %}

  entermarks.html
  ---------------------------
  {% extends 'markattend.html' %}
  {% block title %}
  <title>Enter Marks</title>
  {% endblock %}
  {% block content %}
  <p class="text-center">Enter Marks of {{data.1.sub_br_id_id}} </p><br>
  <div class="container">
  <form action="{% url 'entermarks' data.1.sub_br_id_id %}" method="POST">
  {% csrf_token %}
  <table class="table table-bordered table-striped">
    <thead class="dark">
      <tr>
        <th scope="col">Student_Id</th>
        <th scope="col">Marks Gained</th>
      </tr>
    </thead>
    <tbody class='cd'>
          {% for i in data %}
      <tr>
        <th scope="row">{{i.stud_id_id}}</th>
        <td>
          <input type="number" min="0" max={{i.sub_tot_marks}} step="1" name={{i.stud_id_id}}
  value={{i.sub_secured}} />
        </td>
      </tr>
      {% endfor %}</tbody></table>
  {% endblock %}

   studentservices.html
  ------------------------------
  <!DOCTYPE html>
  <html lang="en">
  <head>
    <title>Student Services</title>
    <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  <style>
    body {font-family: Arial, Helvetica, sans-serif;
         background-color: lightblue;}
    p {
       font-size:50px;
       color: red;
       }
    h1{ font-family: American Typewriter, serif }

    u{ color: blue;}
    .row.content {height: 450px}
   @media screen and (max-width: 767px) {
     .row.content {height:auto;}
   }
     .button {
     background:#fdb531;
     box-shadow:20px 20px 0 rgba(0,0,0,2);
     display:inline-block;
     font-size:2em;
     padding:.5em 2em;
     text-decoration:none;
     width:100%;
     padding-top: 150px;
     padding-bottom: 150px;
     border-top: 50 px;
     font-family: URW Chancery L, cursive;
   }

   .button:hover {
     box-shadow: 0 0 0 ;
   }

   .ellipse{
    background: #9fb325;
    border-radius: 50%;
   }
  </style>
</head>
```

```
<body>
  <p class="text-center"> Welcome {{data.stu_user_name}}</p><br>
  <div class="container">
  <a class="btn navbar-btn btn-danger navbar-right pull-right" role="button" href="{% url
'center' %}">Logout</a>
  </div>
  <h1 class="text-center"> <u>Services</u> </h1>
  <div class="container-fluid text-center">
       <div class="row content">
         <div class="col-sm-2">
         </div>
         <div class="col-sm-4 text-center">
           <div class="container-fluid">
                     <a class="button ellipse" href="{% url 'attendancecheck' data.student_id %}">
                               <i>Attendance Checking</i>
            </a>
           </div>
         </div>
         <div class="col-sm-4 text-center">
           <div class="container-fluid">
                      <a class="button ellipse" href="{% url 'markscheck' data.student_id %}">
                               <i>Marks Checking</i>
             </a>
           </div>
         </div>
         <div class="col-sm-2">
         </div>
       </div>
    </div>
    <div class="container-fluid text-center">
       <div class="row content">
         <div class="col-sm-4">
         </div>
         <div class="col-sm-4 text-center">
           <div class="container-fluid">
                     <a class="button ellipse" href="{% url 'feecheck' data.student_id %}">
                               <i>Fee Balance</i>
              </a>
           </div>
         </div>
         <div class="col-sm-4">
         </div>
       </div>
    </div></body></html>
```

feecheck.html

```
-------------------
<!DOCTYPE html>
<html>
<head>
  {% block title %}
        <title>Fee Check</title>
  {% endblock %}
        <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <style>
          body {font-family: Arial, Helvetica, sans-serif;
           background-color: #08675f;}
          p {
         font-size:50px;
         color: red;
        }
        .dark{
          font-size:20px;
          background:black;
          color:white;
        }
        .cd{
          font-size: 20px;
          color: red;
        }
    </style>
</head>
<body>
{% block content %}
<p class="text-center"> Fee Balance</p><br>
<div class="container">
<table class="table table-bordered">
  <thead class="dark">
    <tr>
      <th scope="col">Name</th>
      <th scope="col">Fee Balance</th>
    </tr>
  </thead>
```

```
<tbody class='cd'>
    <tr>
      <th scope="row">{{data.stu_name}}</th>
      <td>{{data.stu_feebalance}}</td>
    </tr>
  </tbody>
</table>
</div>
{% endblock %}
</body>
</html>
```

attendancecheck.html

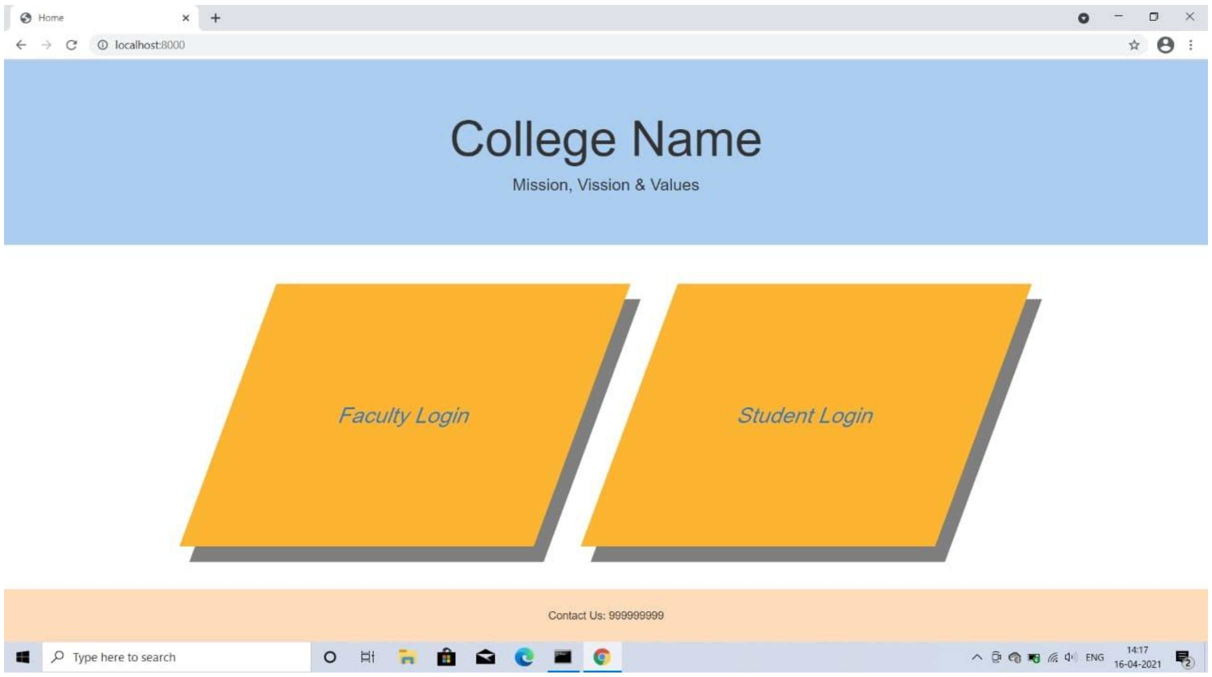
```
--------------------------------
{% extends 'feecheck.html' %}
{% block title %}
<title>Attendance Check</title>
{% endblock %}
{% block content %}
<p class="text-center">Attendance</p><br>
<div class="container">
<table class="table table-bordered">
  <thead class="dark">
    <tr>
      <th scope="col">Subject_Id</th>
      <th scope="col">Total_Periods</th>
      <th scope="col">No. of_Periods_Present</th>
    </tr>
  </thead>
  <tbody class='cd'>
        {% for i in data %}
    <tr>
      <th scope="row">{{i.sub_br_id_id}}</th>
      <td>{{i.sub_tot_attendance}}</td>
      <td>{{i.sub_tot_present}}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>
</div>
{% endblock %}
```

```
markscheck.html
-----------------------
{% extends 'feecheck.html' %}
{% block title %}
<title>Marks Check</title>
{% endblock %}
{% block content %}
<p class="text-center">Internal Marks</p><br>
<div class="container">
<table class="table table-bordered">
  <thead class="dark">
    <tr>
      <th scope="col">Subject_Id</th>
      <th scope="col">Marks_Secured</th>
    </tr>
  </thead>
  <tbody class='cd'>
        {% for i in data %}
    <tr>
      <th scope="row">{{i.sub_br_id_id}}</th>
      <td>{{i.sub_secured}}</td>
    {% endfor %}
  </tbody></table></div>
{% endblock %}

success.html
------------------
<!DOCTYPE html>
<html>
<head>
        <title>Success</title>
        <style>
                a{
                        color: #ab90fe;
                        text-align: center;
                        font-size: 50px;
                }
        </style>
}
</head>
<body>
        <center>
        <a href="{% url 'center' %}">Database Updated</a>
    </center></body></html>
```

4.2 Screen shots

Home page

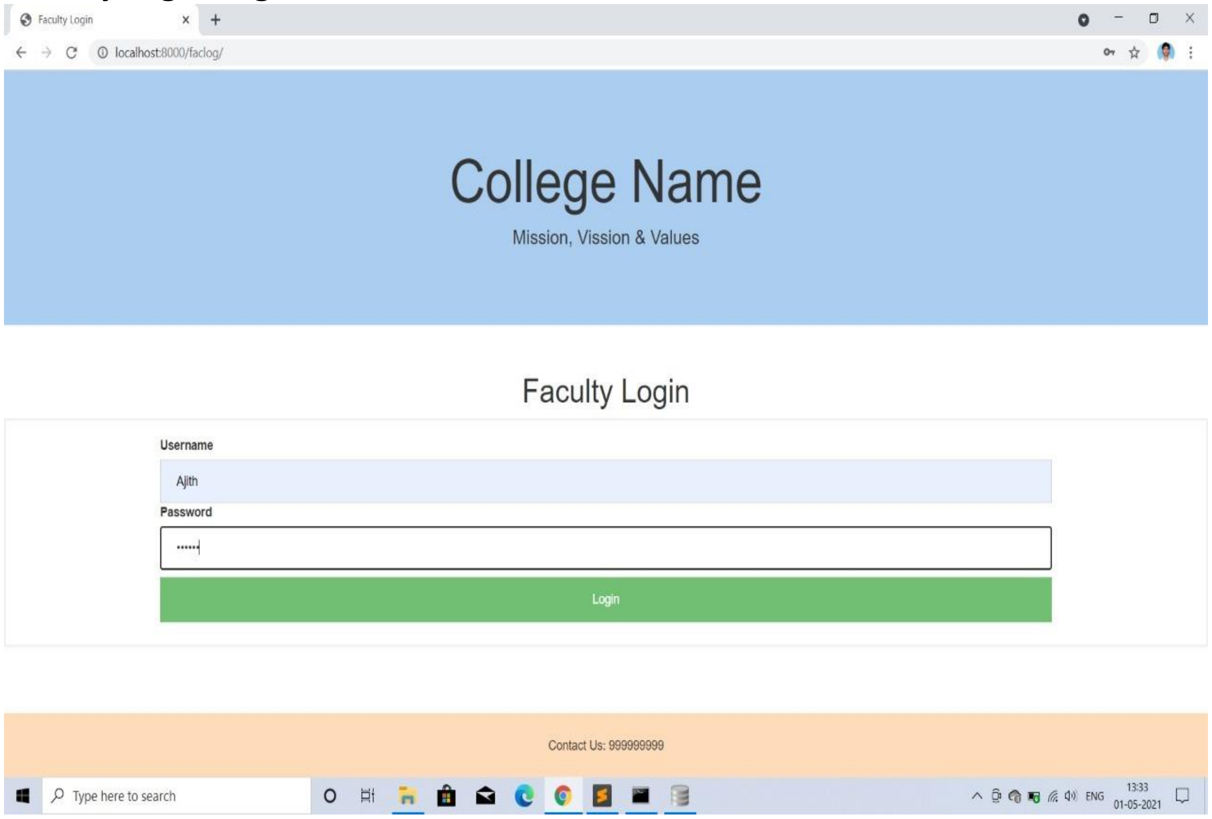


College Name

Mission, Vission & Values

Contact Us: 999999999

Faculty Login Page



College Name

Mission, Vission & Values

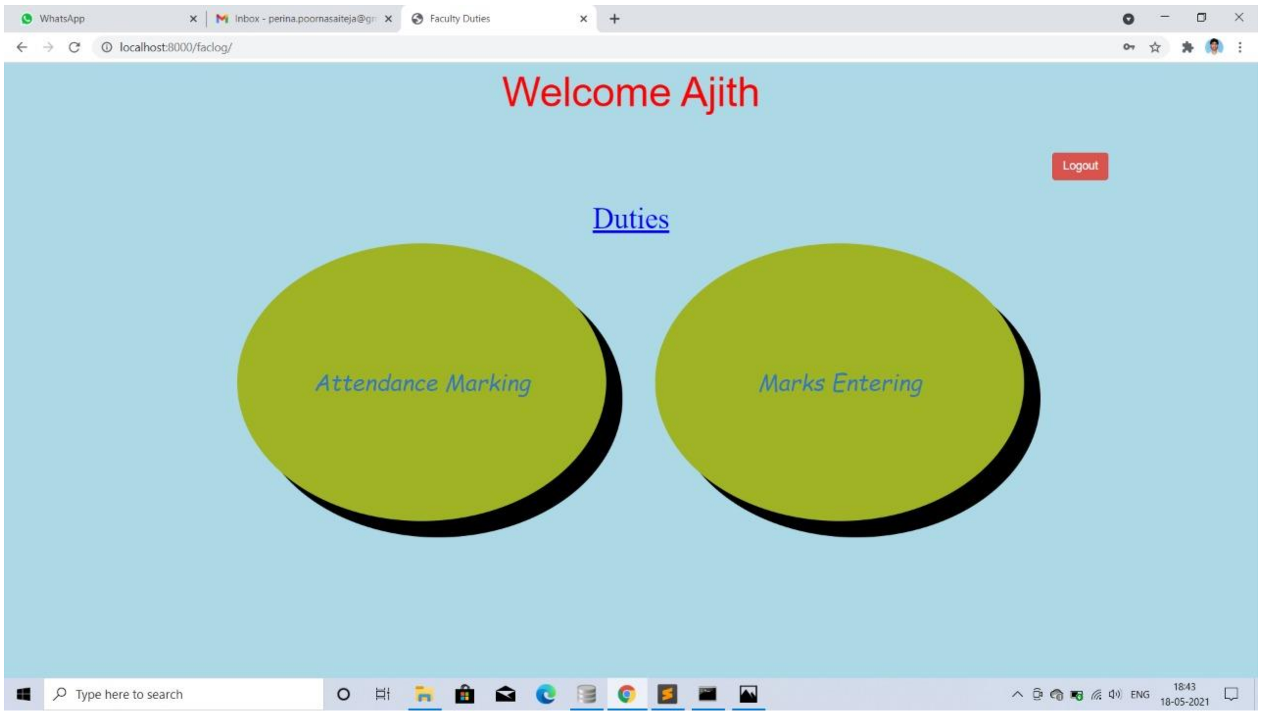
Faculty Login

Username: Ajith

Password: ••••••

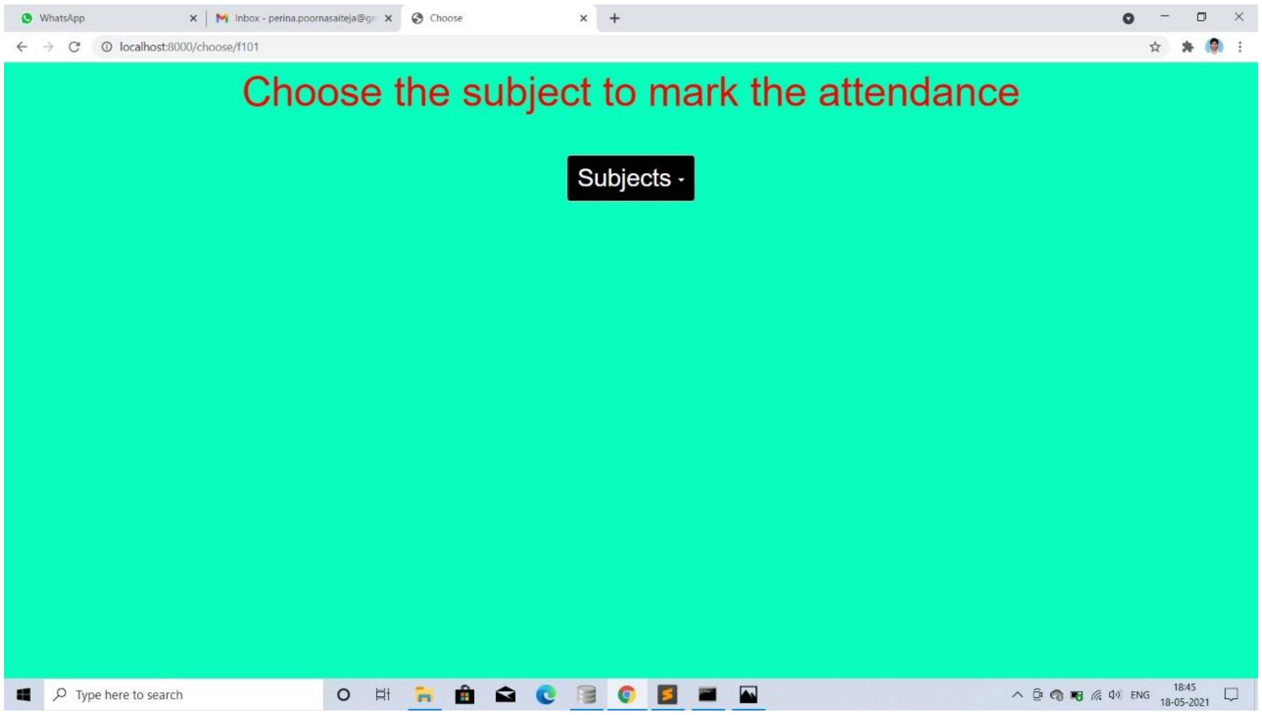
Contact Us: 999999999

Faculty Page

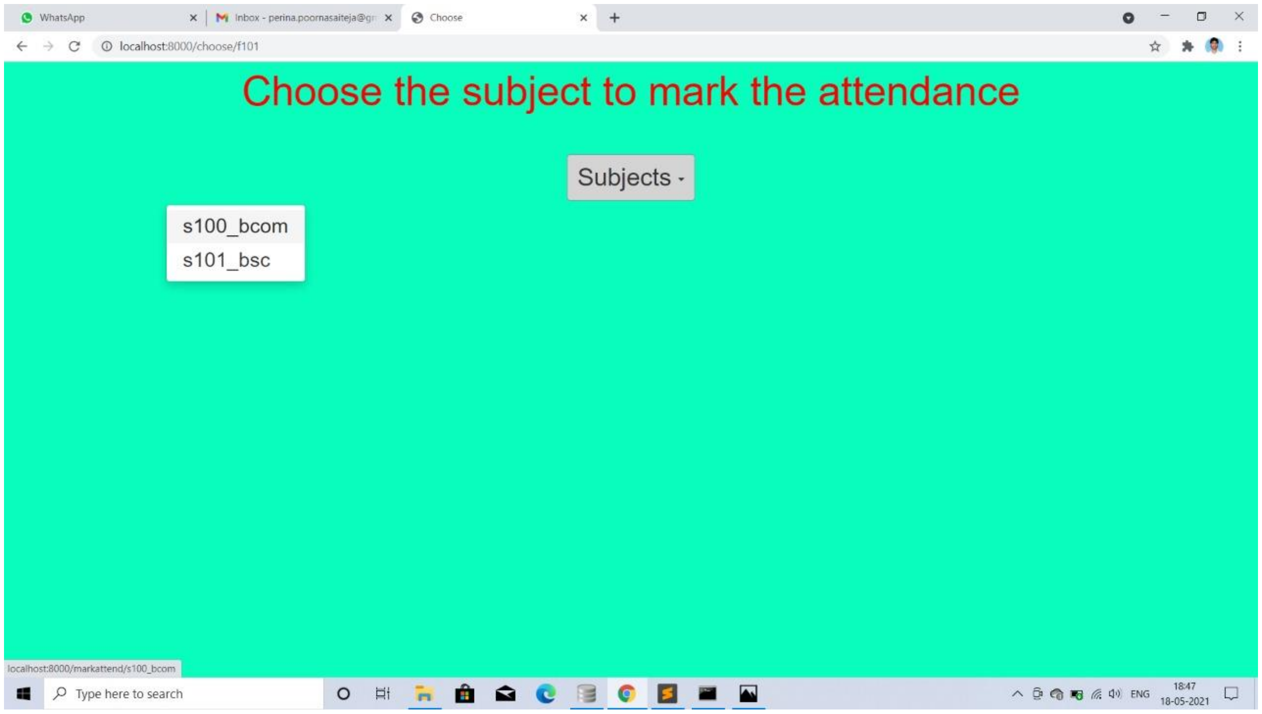


This page appears when the faculty successfully login to his/her account.

Attendance Marking Page



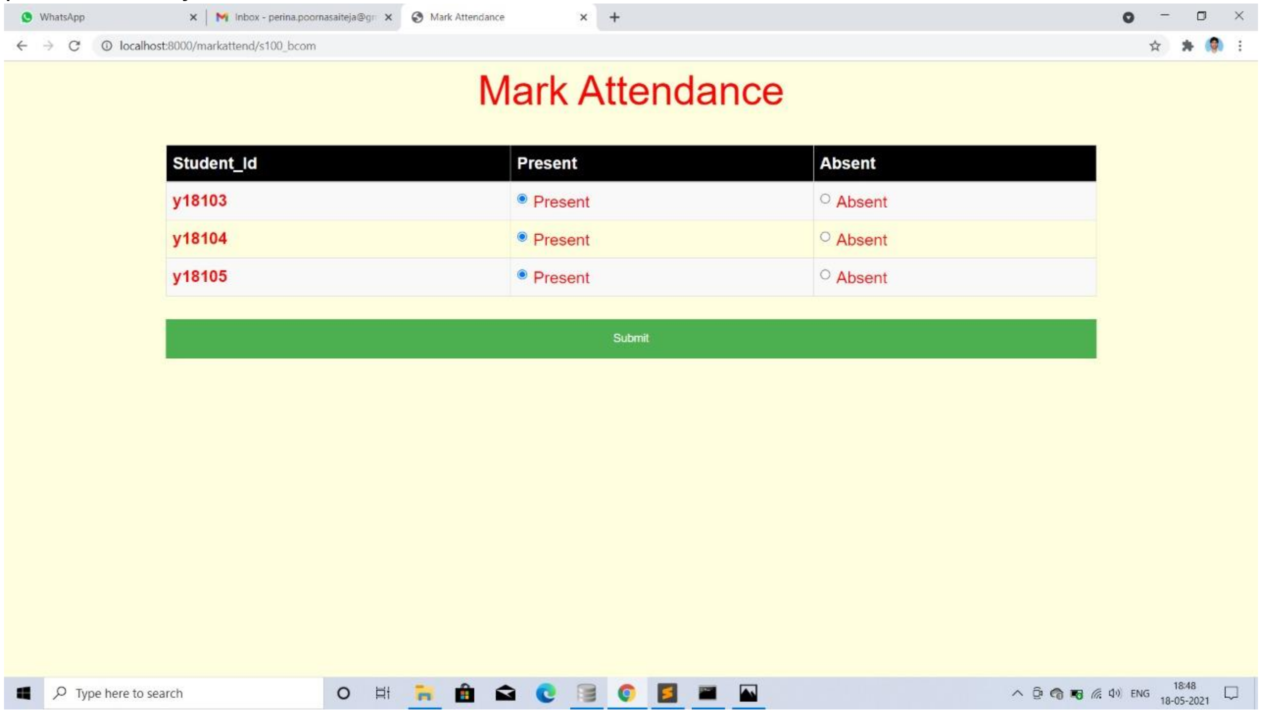
When the teacher selects attendance marking option then this page will appear.


WhatsApp
Inbox - perina.poornasaiteja@gm
Choose
localhost:8000/choose/f101
Subjects
localhost:8000/markattend/s100_bcom
Type here to search
ENG
18:47
18-05-2021

Choose the subject to mark the attendance

s100_bcom
s101_bsc

This page appears when the teacher his selects the subjects list.In this he has to choose a particular subject to mark attendance.

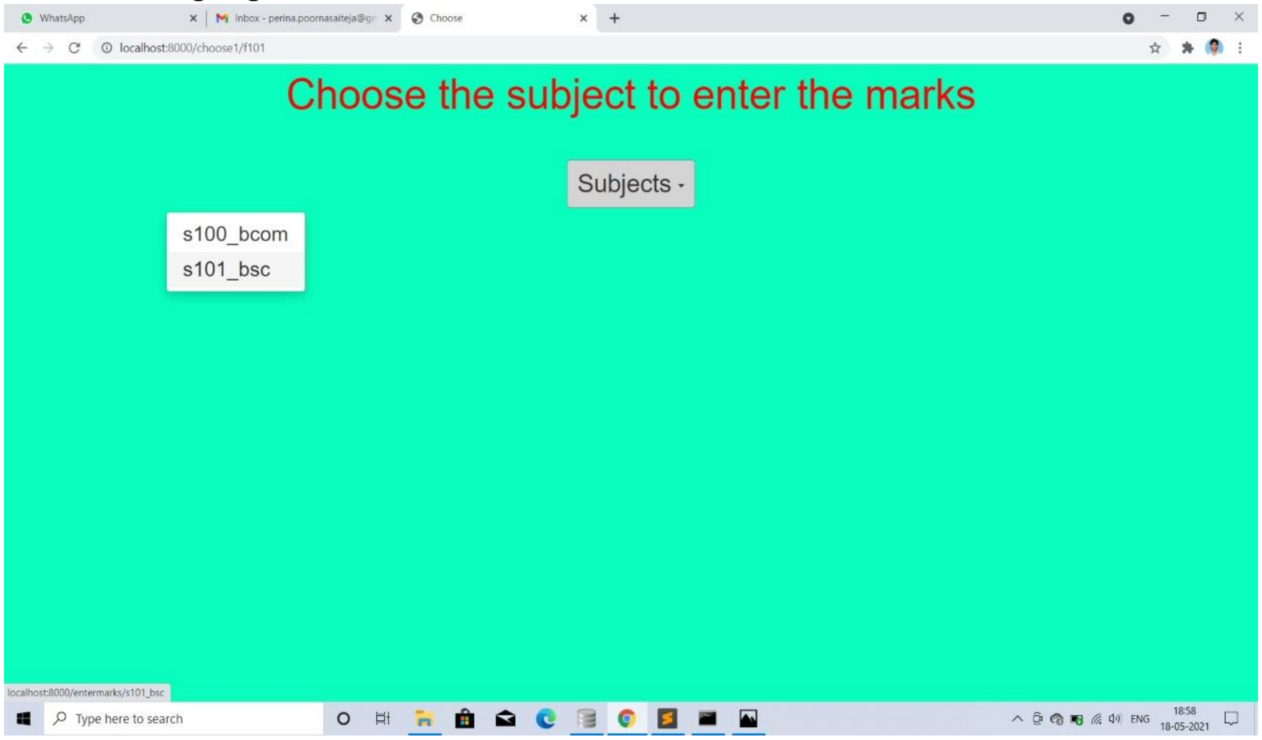

WhatsApp
Inbox - perina.poornasaiteja@gm
Mark Attendance
localhost:8000/markattend/s100_bcom
Submit
Type here to search
ENG
18:48
18-05-2021

Mark Attendance

Student_Id	Present	Absent
y18103	● Present	○ Absent
y18104	● Present	○ Absent
y18105	● Present	○ Absent

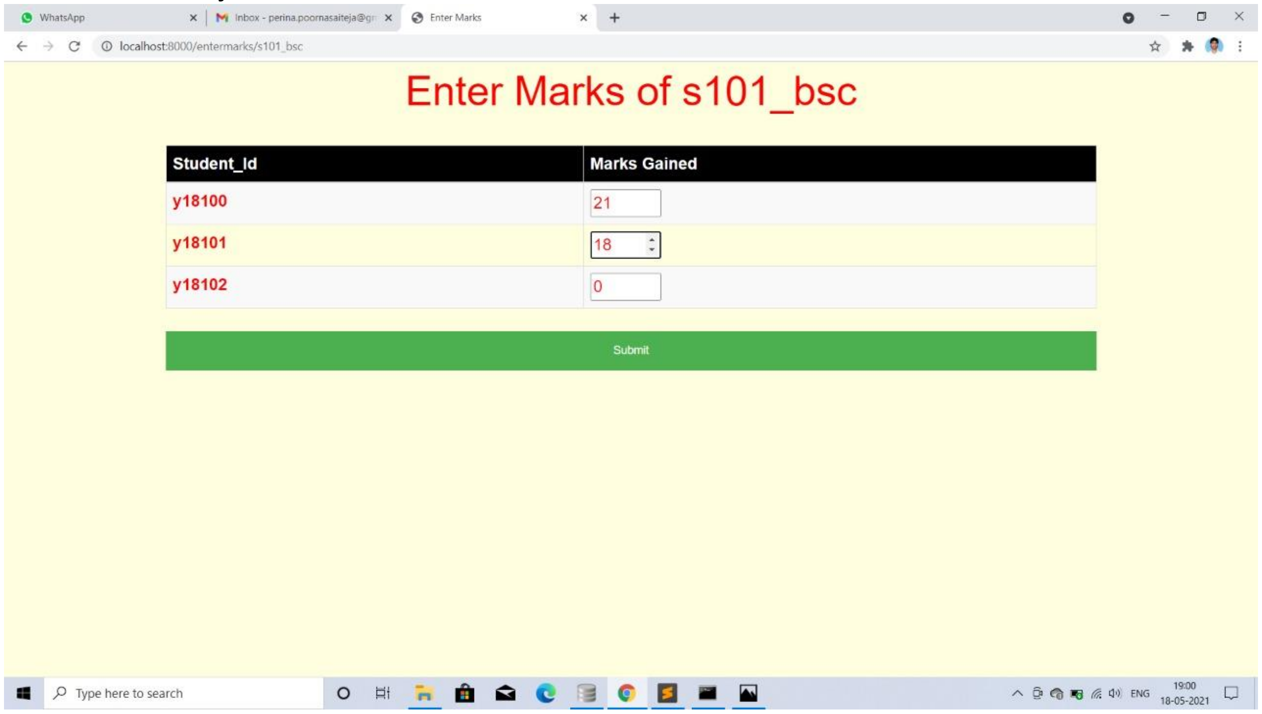
In this page the teacher will mark attendance to students by simply selecting radio button option.

Marks Entering Page:-


WhatsApp
Inbox - perina.poornasaiteja@gr
Choose
localhost:8000/choose1/f101
Subjects
s100_bcom
s101_bsc
localhost:8000/entermarks/s101_bsc
Type here to search

Choose the subject to enter the marks

In this page the teacher will enter the marks of student of particular subject. The teacher wants to select the subjects in which he want to enter the marks.

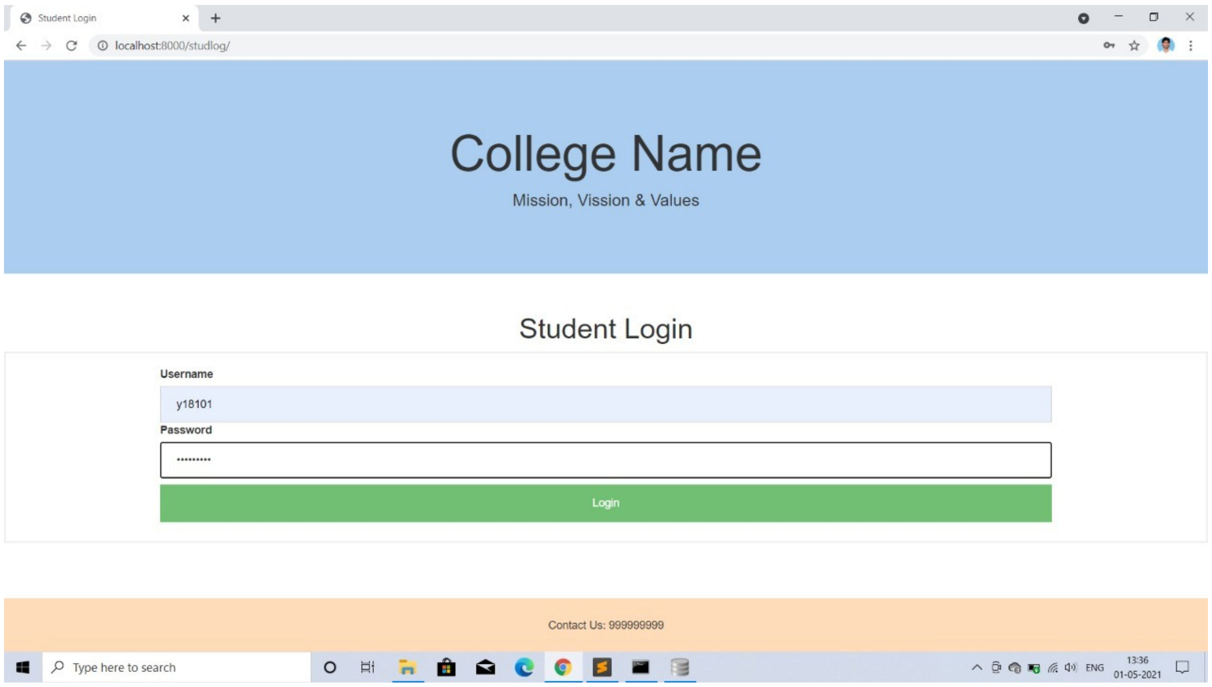

WhatsApp
Inbox - perina.poornasaiteja@gr
Enter Marks
localhost:8000/entermarks/s101_bsc
Submit
Type here to search

Enter Marks of s101_bsc

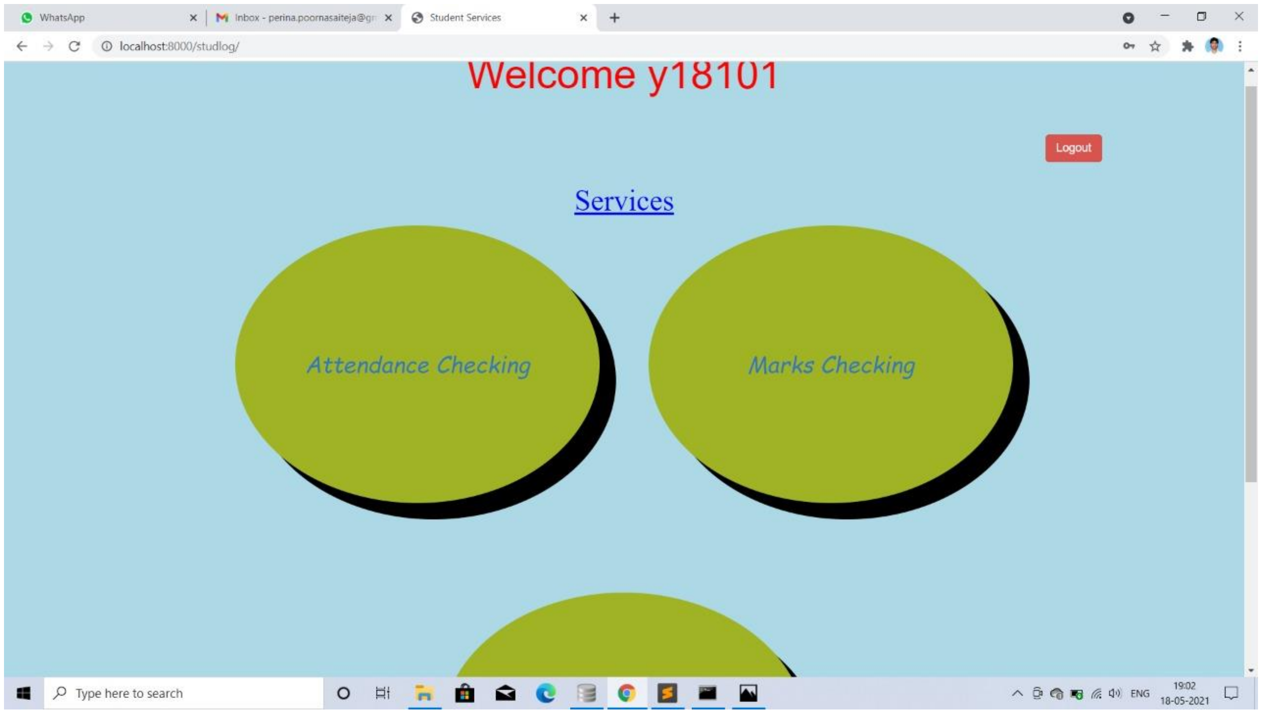
Student_Id	Marks Gained
y18100	21
y18101	18
y18102	0

Here after selection of subject , the teacher can enter the marks of all students at one place.

Student Login Page

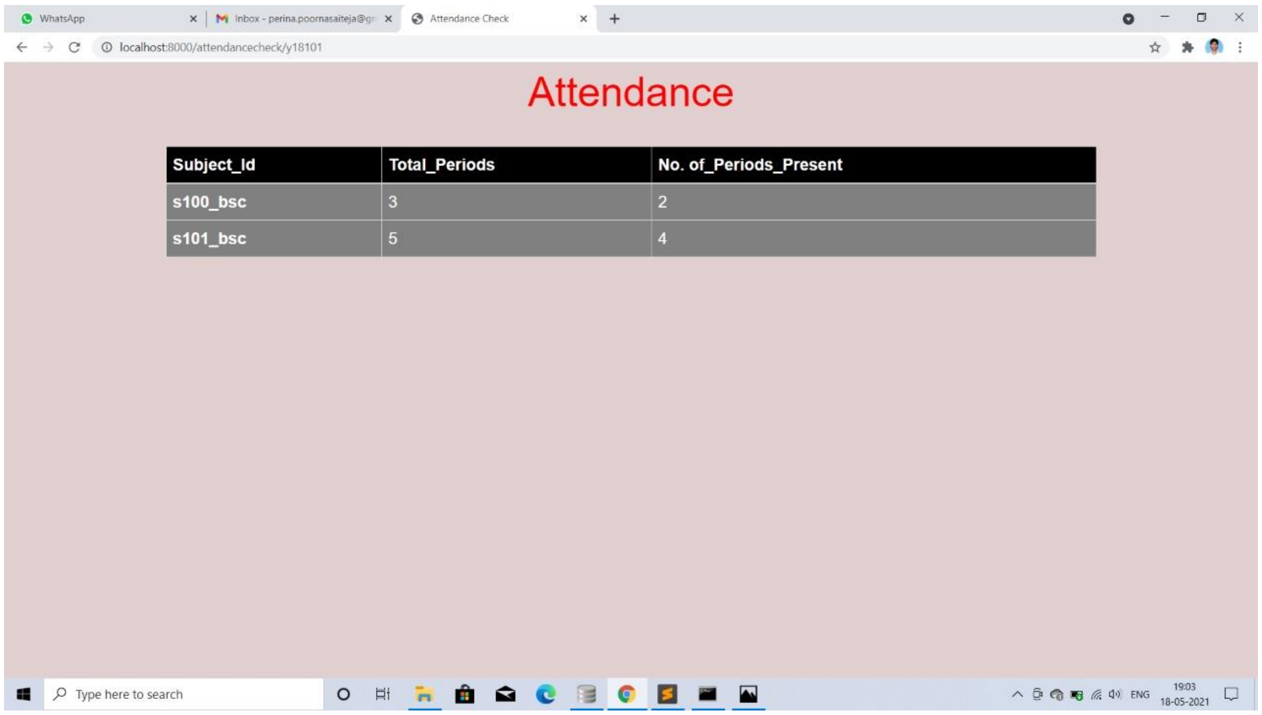


Student Page



After successful student login the student can select any of the listed options.

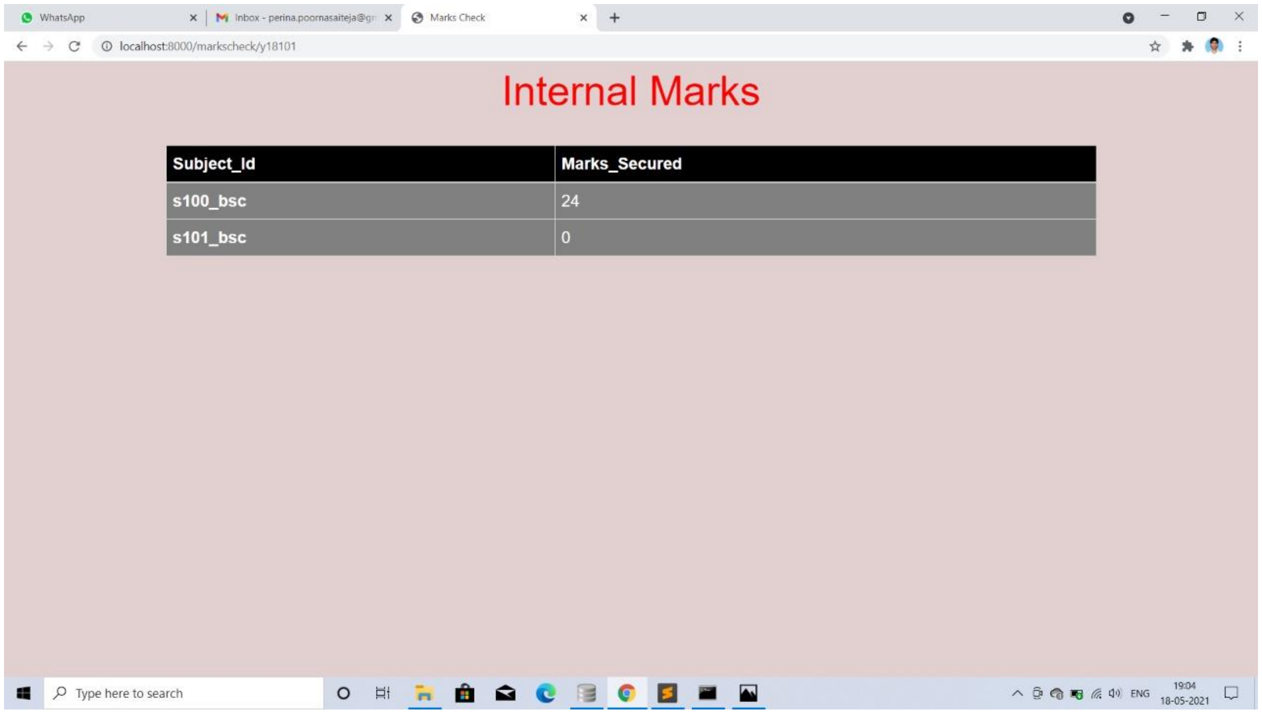
Attendance Checking Page


WhatsApp
Inbox - perina.poornasaiteja@gr
Attendance Check
localhost:8000/attendancecheck/y18101
Type here to search
19:03
18-05-2021

Attendance

Subject_Id	Total_Periods	No. of_Periods_Present
s100_bsc	3	2
s101_bsc	5	4

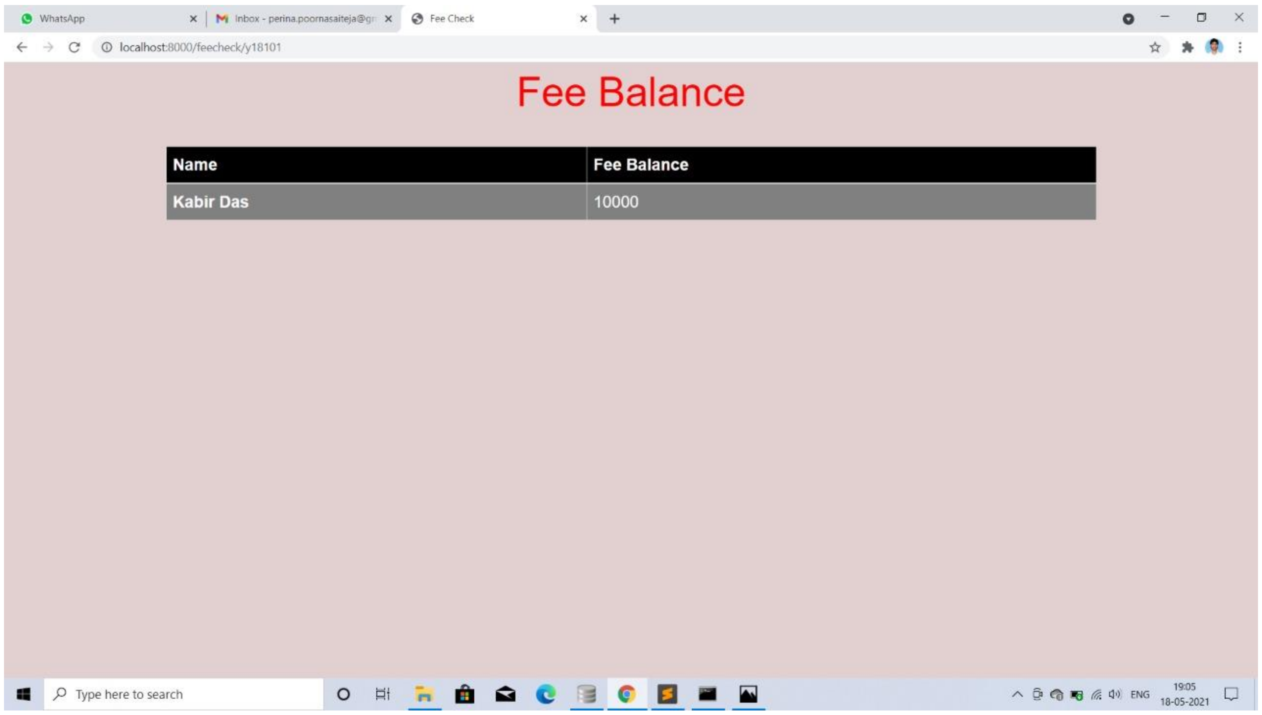
Marks checking page


WhatsApp
Inbox - perina.poornasaiteja@gr
Marks Check
localhost:8000/markscheck/y18101
Type here to search
19:04
18-05-2021

Internal Marks

Subject_Id	Marks_Secured
s100_bsc	24
s101_bsc	0

Fees Checking Page


WhatsApp
Inbox - perina.poornasaiteja@g
Fee Check
localhost:8000/feecheck/y18101
Type here to search
ENG
19:05
18-05-2021

Fee Balance

Name	Fee Balance
Kabir Das	10000

5.Testing

5.1 Introduction to Testing

Testing is a fault detection technique that tries to create failure and erroneous states in a planned way. This allows the developer to detect failures in the system before it is released to the customer.

Note that this definition of testing implies that a successful test is test that iden- tifies faults.We will use this definition throughout the definition phase.Another often used definition of testing is that it demonstrates that faults are not present. Testing can be done in two ways:

1. Top down approach

2.Bottom up approach

1. Top down approach:

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written.

2. **Bottom up Approach:** Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short

program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. In this project, bottom up approach is used where the lower level modules are tested first and the next ones having much data in them. Testing Methodologies The following are the Testing Methodologies:

1. Unit Testing.
2. Integration Testing.
3. User Acceptance Testing.
4. Output Testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a modules control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module inter- faces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main

objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design. The following are the types of Integration Testing:

1. Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner. In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. **Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always

available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

1. The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.

2. A driver (i.e.) the control program for testing is written to coordinate test case input and output.

3. The cluster is tested. d. Drivers are removed and clusters are combined moving upward in the program structure. The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways one is on screen and another in printed format.

5.2 Test Cases

Test Case Name	**Description**	**Step**	**Expected**	**Actual**	**Test Status(P/F)**
Staff Login	Username and Password	Check valid login or not	Matching username password of staff	Result	P
Student Login	Username and Password	Check valid login or not	Matching username password of student	Result	P

6.Conclusion

- ➢ The main agenda of this project is to build effective system which can maintain attendance , marks of students so that it can incorporate any future enhancements.
- ➢ Students and Staff can easily identify their requirements.
- ➢ Human prone errors will be minimised.