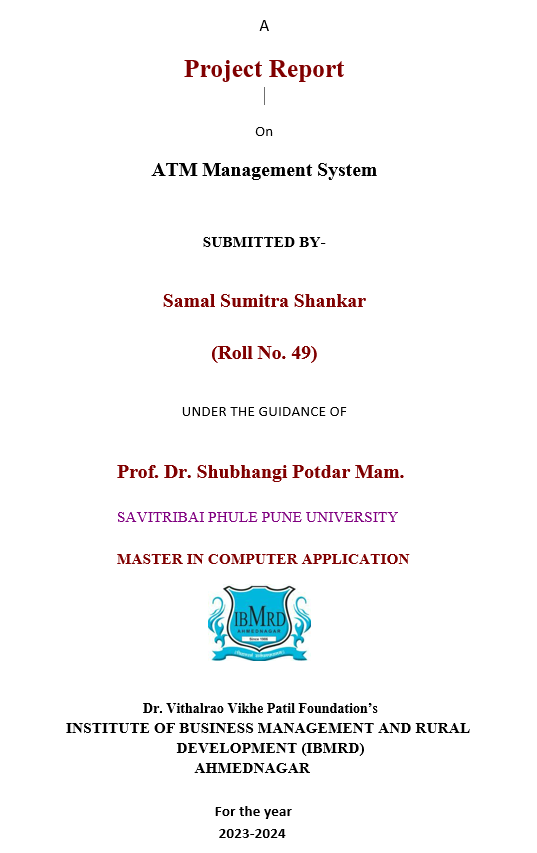
****



**CERTIFICATE Date:**

This is to certify **Miss. Sumitra Shankar Samal** student of Institute of Business Management and Rural Development have successfully completed the Mini Project work prescribed by the Savitribai Phule Pune University, Pune in the partial fulfillment of the requirement of First Year, Master of Computer Application Program for the Academic Year 2022-23.

The Project Work titled as **“Online Student Reports Maintaining System”**

Dr. Shubhangi Potdar Dr. Shubhangi Potdar

**Project Guide Head of Dept.**

**External Examiner Internal Examiner**

**ACKNOWLEDGEMENT**

At every outset we express my gratitude to almighty lord for showering his grace and blessings upon me to complete this project.

Although our name appears on the cover of this book, many people had contributed in some form or the other form to this project Development. We could not do this project without the assistance or support of each of the following we thank you all.

We wish to place on my record my deep sense of gratitude to my project guide, for his constant motivation and valuable help through the project work. Express my gratitude to **Dr. Sanjay Dharmadhikari (Director)** and **Dr. Potdar Shubhangi (H.O.D)** of Institute of Business Management And Rural Development for her valuable suggestions and advices throughout the **MCA** course. We also extend my thanks to other faculties for their Cooperation during my Course.

Finally, we would like to thank my friends for their co-operation to complete this project.

**Miss. Sumitra Shankar Samal**

**DECLARATION**

We hereby declare that the project work entitled,

**“Online Student Reports Maintaining System”** submitted under the guidance of **Prof. Dr. Shubhangi Potdar** is our original work completed under the four walls of our institute.

The Report submitted is our own work and has not been duplicated from any other source. We shall be responsible for any unpleasure moment/situation.

**Miss. Sumitra Shankar Samal**

**TITLE**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Topic** | **Page.No.** |
| **1.** | **Introduction** | **7** |
|  | **1.1 Existing System and Need for System** | **8** |
|  | **1.2 Scope of work** | **8** |
|  | **1.3 Operating Environment – Hardware and Software** | **9** |
|  | **1.4 Details Description of Technology used** | **9** |
| **2.** | **Proposed System** |  |
|  | **2.1 Proposed System** | **11** |
|  | **2.2 Objectives of System** | **11** |
|  | **2.3 User Requirements** | **12** |
| **3.** | **Analysis And Design** |  |
|  | **3.1 Use Case Diagram** | **14** |
|  | **3.2 Activity Diagram** | **15-21** |
|  | **3.3 Sequence Diagram** | **22-25** |
|  | **3.4 Collaboration Diagram** | **26** |
|  | **3.5 Class Diagram** | **27** |
|  | **3.6 Component Diagram** | **28** |
|  | **3.7 Deployment Diagram** | **29** |
|  | **3.8 State Transition Diagram** | **30** |
|  | **User Interface Designs** | **31-42** |
|  | **Sample Code** | **43-48** |
|  | **Table Specification** | **49-50** |
|  | **Drawbacks and Limitations** | **51** |
|  | **Proposed Enhancement** | **51** |

**Chapter: 1**

**INTRODUCTION**

1. **Existing System and Need for System**

* A student record management system enables you to track and maintain several records. It completely automates the process of data tracking and record-keeping in institutes collected from various sources such as admission, attendance, performance, documents, fees, and behavioral records.
* A student record management software is a tool that tracks and records regular activities of the students in institutes including attendance, exam performance, and their behavior.
* The software can be accessed by students, teachers, admin and parents with a role-based login.
* It collects all the student data along with their personal information which can be easily searched and retrieved later on. The software stores years of data online on a cloud platform.
* A records management system is an tool used to track, store, and organize records in the institutes. It enables easy data management activities from its creation to disposal.
* This system helps to assist the management faculty in regulatory compliance, information governance, and risk management.

**2. Scope of the System-**

* A student record management software is a tool that tracks and records regular activities of the students in institutes including attendance, exam performance, and their behavior.
* The Student Result Management System is user-friendly interface enables users to store, manage, and analyze student data with ease.
* Using paper-based systems to manage student data can often be inefficient and prone to errors, making computer- based systems a better alternative.
* An online platform called a Student Result Management System was created to make it easier to manage student scholastic records, grades, and other related data.
* For students, teachers, and administrators to view and handle academic records, the system offers a user-friendly interface.
* The system is intended to automate and simplify the process of managing student data, including keeping track of grades, attendance, and other data, producing reports, and giving students and parents academic input. The whole world and the administrators of educational institutions in our country care about the accuracy of student results.

1. **Operating Environment -**
   * **Hardware Requirements:**

* 8GBRAM required.
* Processor : IntelProcessori7.
* Hard disk storage : 80MB.
* Internet or LAN Connection.
  + **Software Requirements:**
* Programming Language used : Python.
* Software required : Sqlite3, MySQL.
* Tools or IDE : Visual Studio.

1. **Detail Description of Technology used:**
   * In this System, I have used Visual Studio to write python code and I have used MySQL and sqlite3 Database as backend.
   * By using this python technology, I have created different python files which gives very simple and attractive User interface to the user.
   * I am storing all the data on the localhost server.
   * This system is most secure means no one can steal the data which is present in the database.
   * After completing this all operations, I am getting good knowledge and experience about how to make a project that will be useful for the users.

# Chapter: 2

**PROPOSED SYSTEM**

# Proposed System –

# We have successfully proposed the “Online Student Reports maintaining System” for replacing the manual work of the administration.

# By this application Student can easily access the modules like students results and courses other information required to student.

# This application is flexible and can easily access by the student.

# So the time taken for getting the information will be reduced.

1. **Objective of the System-**
   * A Student Result Management System enhances communication between teachers, parents, and students by providing real-time access to performance data. Parents can easily monitor their child's progress through secure portals, enabling them to stay actively engaged in their education.
   * Student information system is a software that helps the institutes to store, maintain, process and compile student data, keep track of their regular activities, attendance and performance and offer required guidance and feedback to them.
   * A student record management system enables you to track and maintain several records. It completely automates the process of data tracking and record-keeping in institutes collected from various sources such as admission, attendance, performance, documents, fees, and behavioral records.

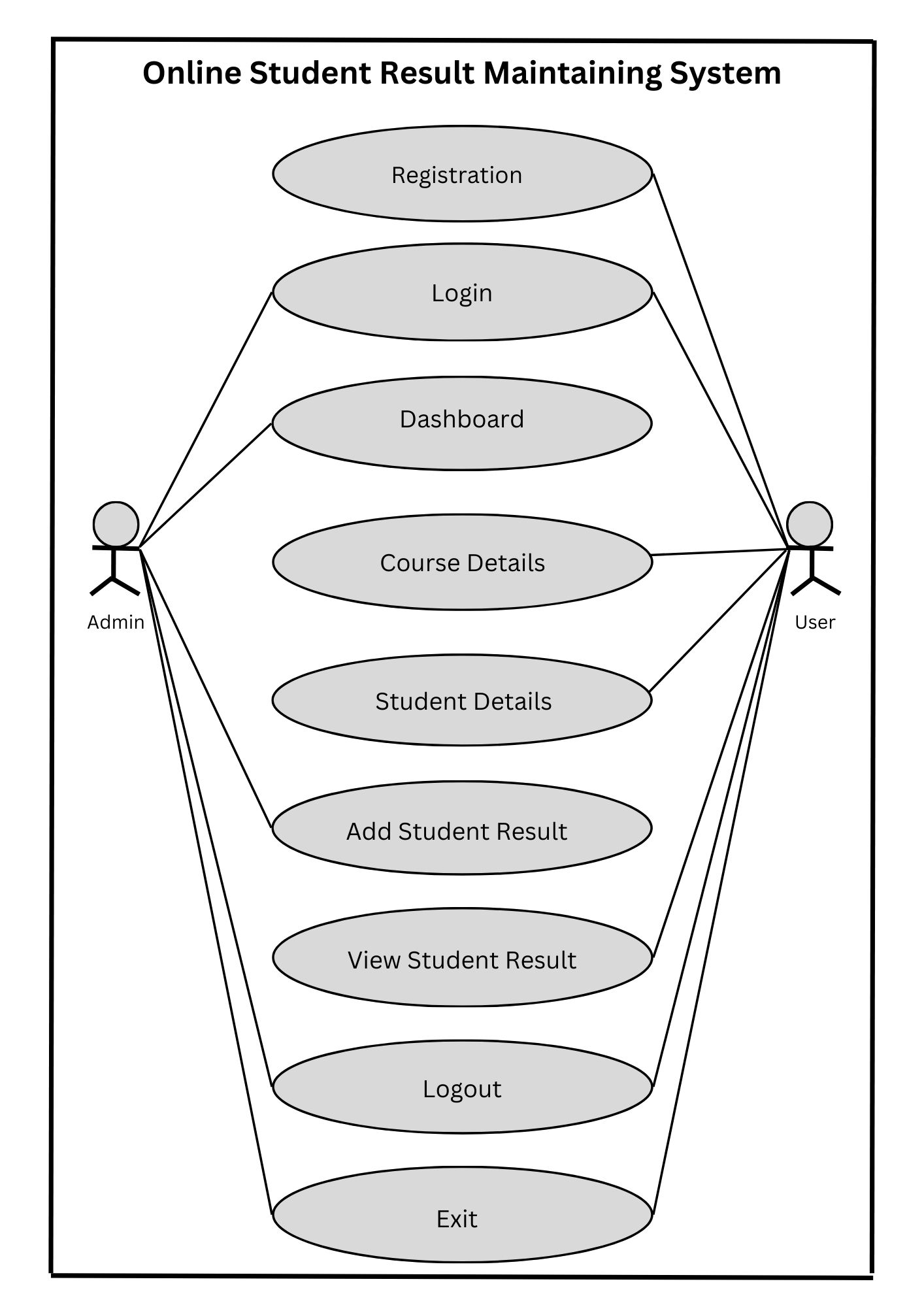
**User Requirements-**

* Depending upon the user role you will be able to access only the specific modules of the system.
* Registration and Login facility for enabling only authorized access to the system.
* User will be able to modify/Update/Delete information about different students that are enrolled for the course in different years.
* User will be able to add/Update/Delete information regarding marks obtained by different students in different semesters.
* User will be able to reset the system leading to deletion of all existing information from the backend database. They will be able to Create/modify/Delete existing user accounts.

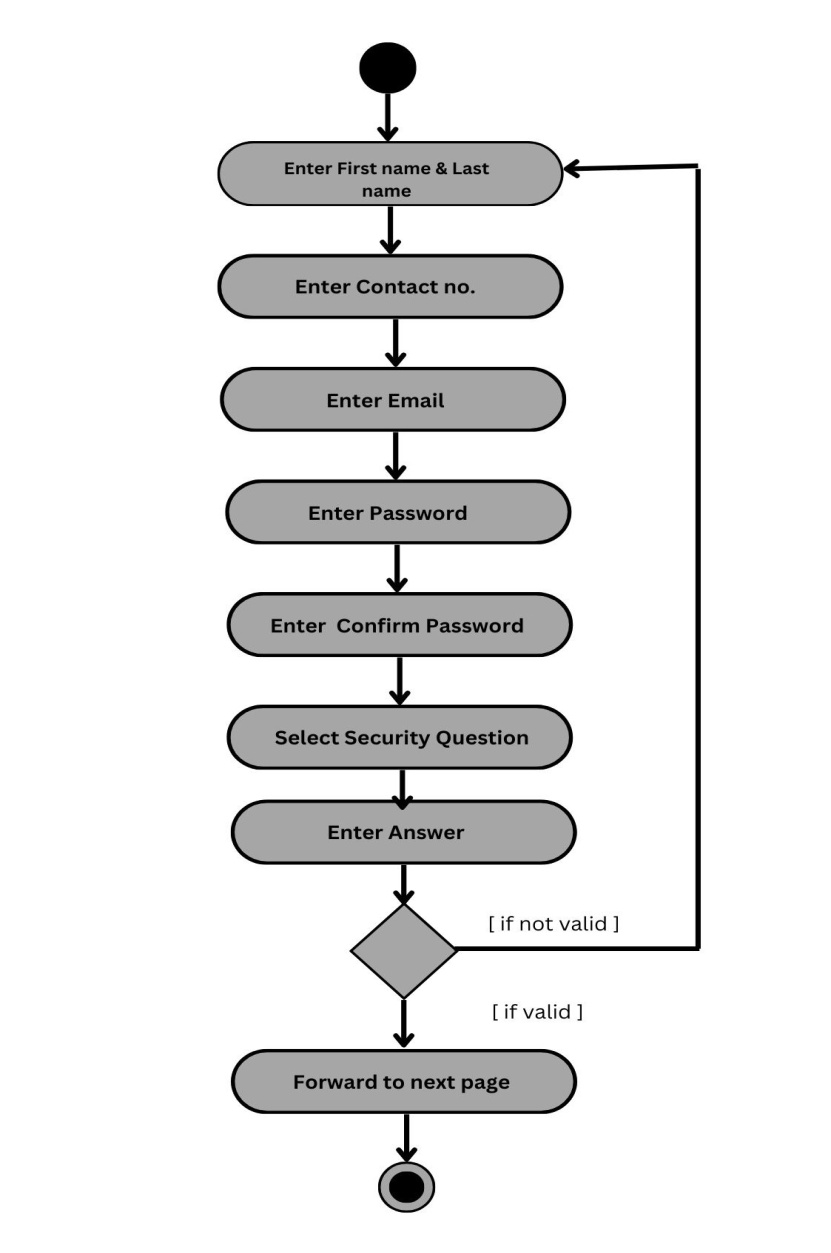
**Chapter: 3**

# ANALYSIS& DESIGN

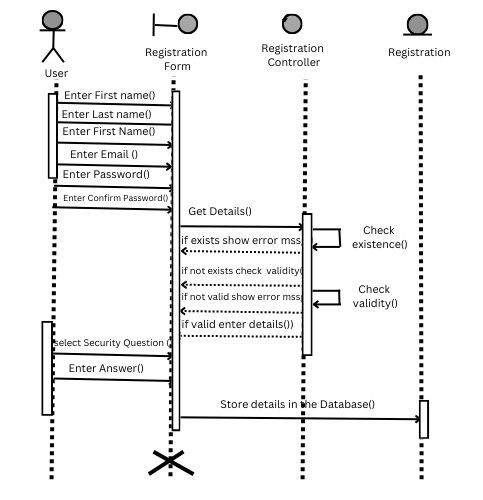
1. **UseCase Diagram**



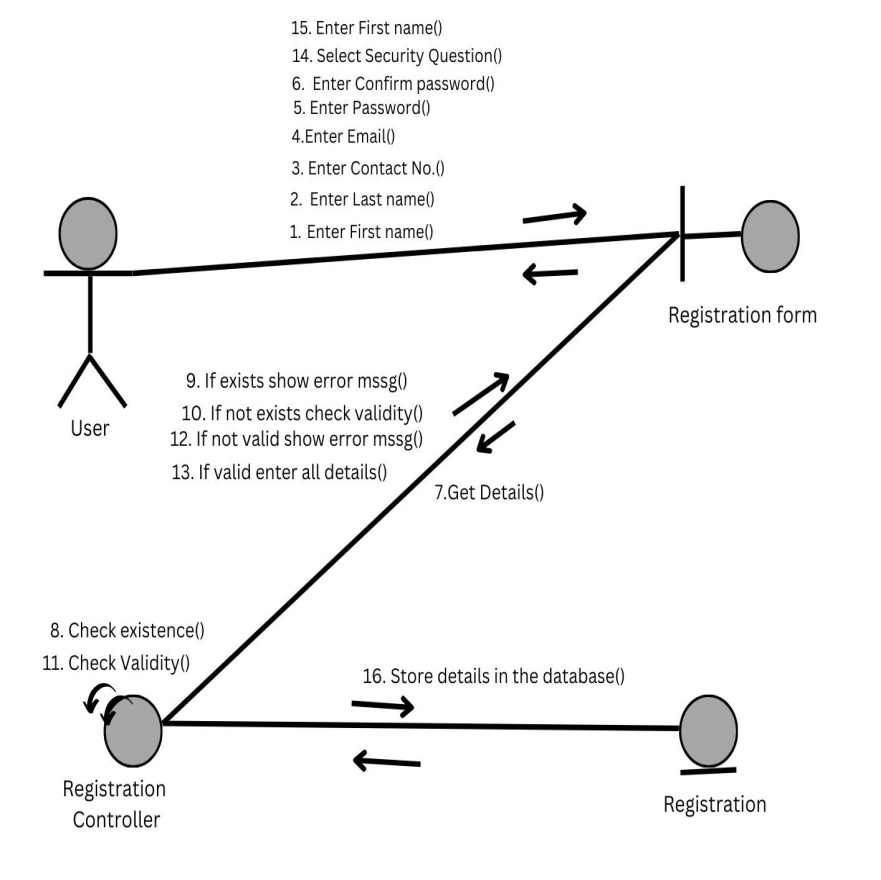
1. **Activity Diagram for Registration**

****

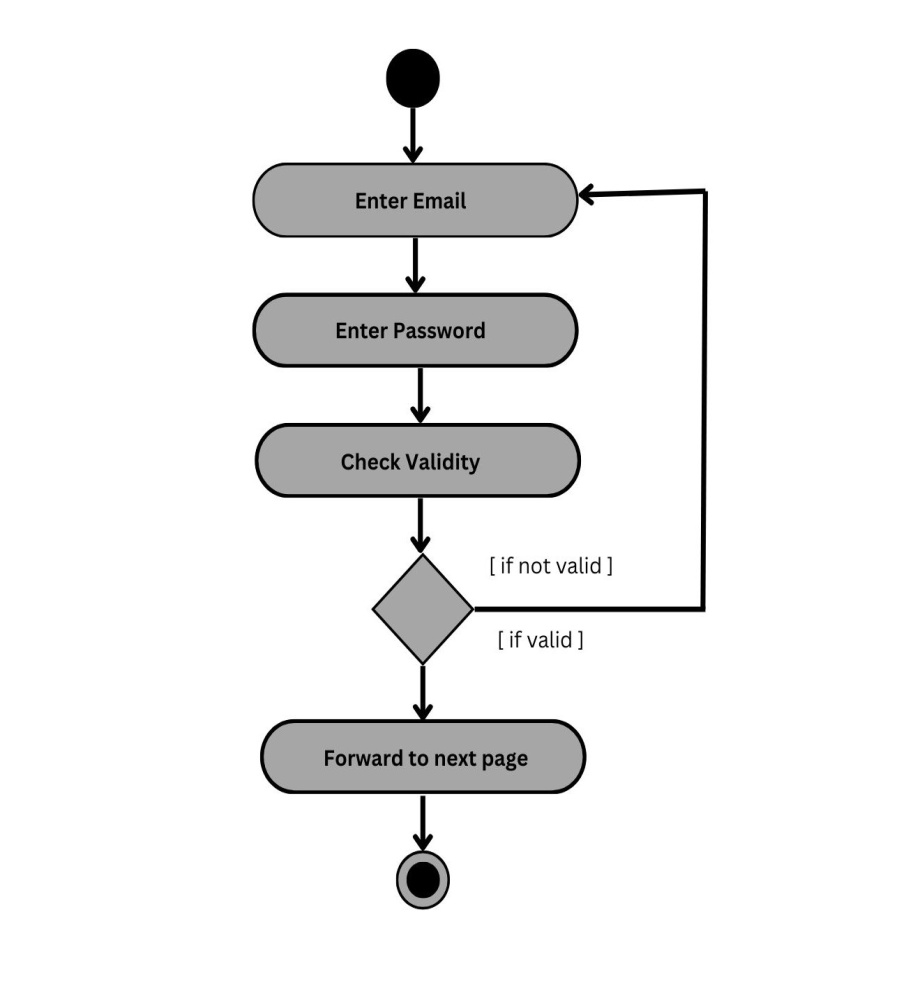
1. **Sequence Diagram for Registration**

****

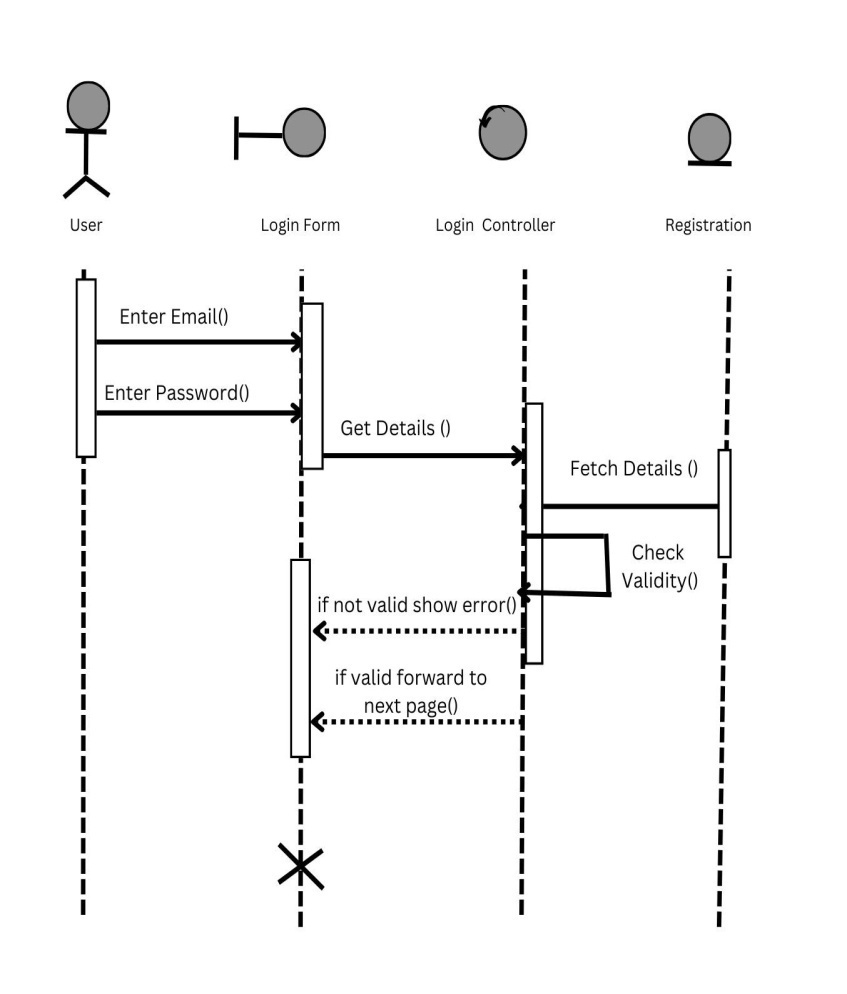
1. **Collaboration Diagram for Registration**

****

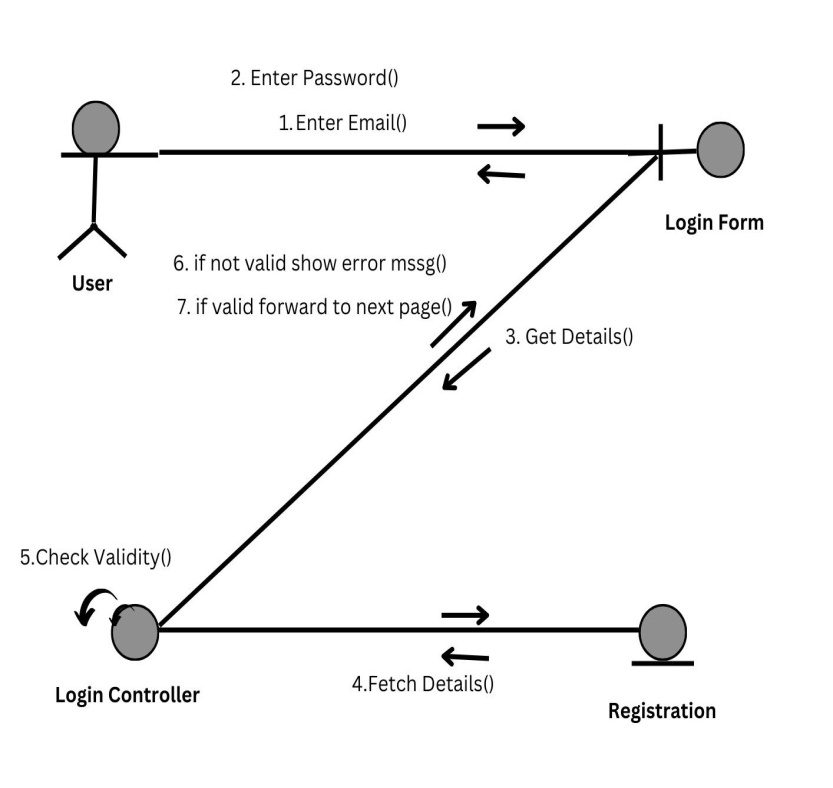
1. **Activity Diagram for Login**

****

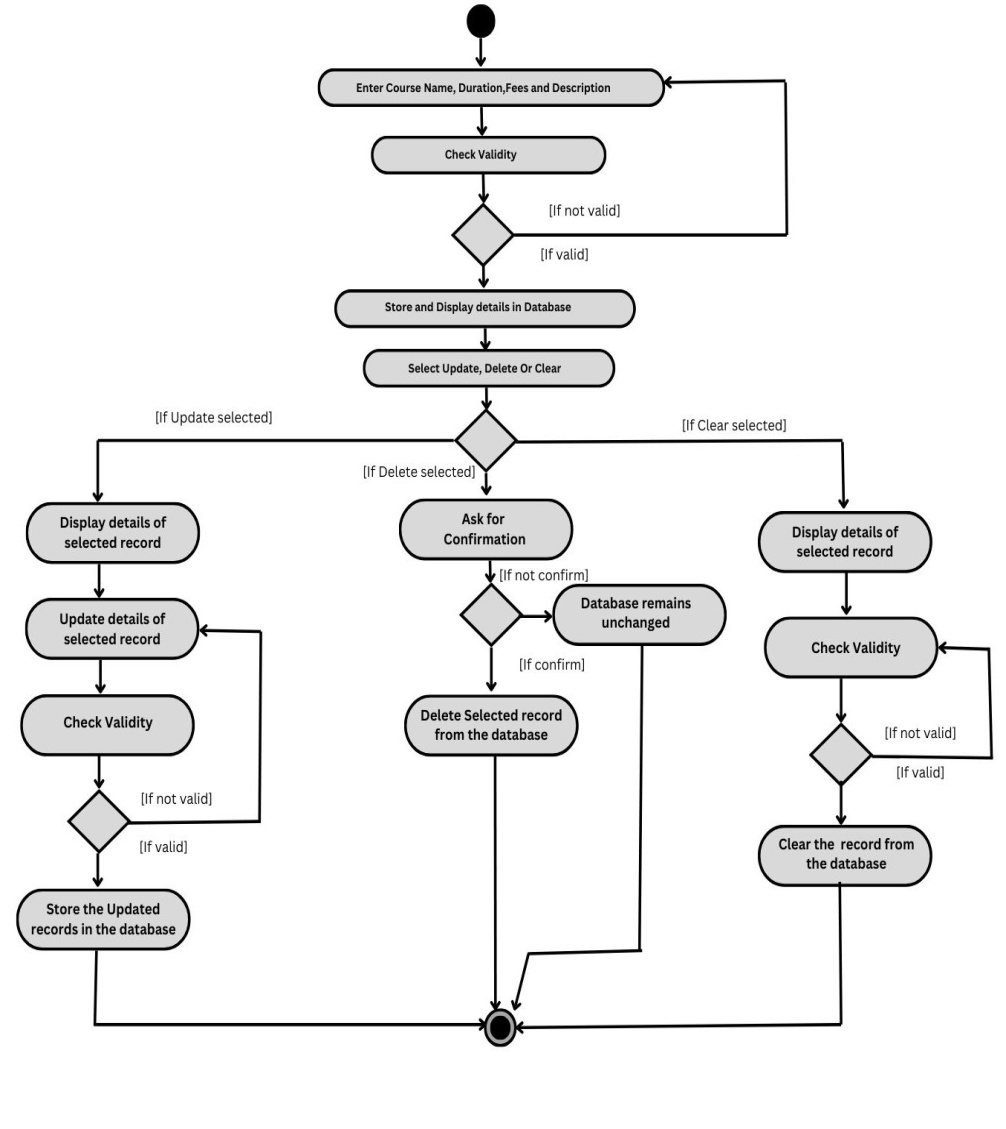
1. **Sequence Diagram for Login**

****

1. **Collaboration Diagram for Login**

****

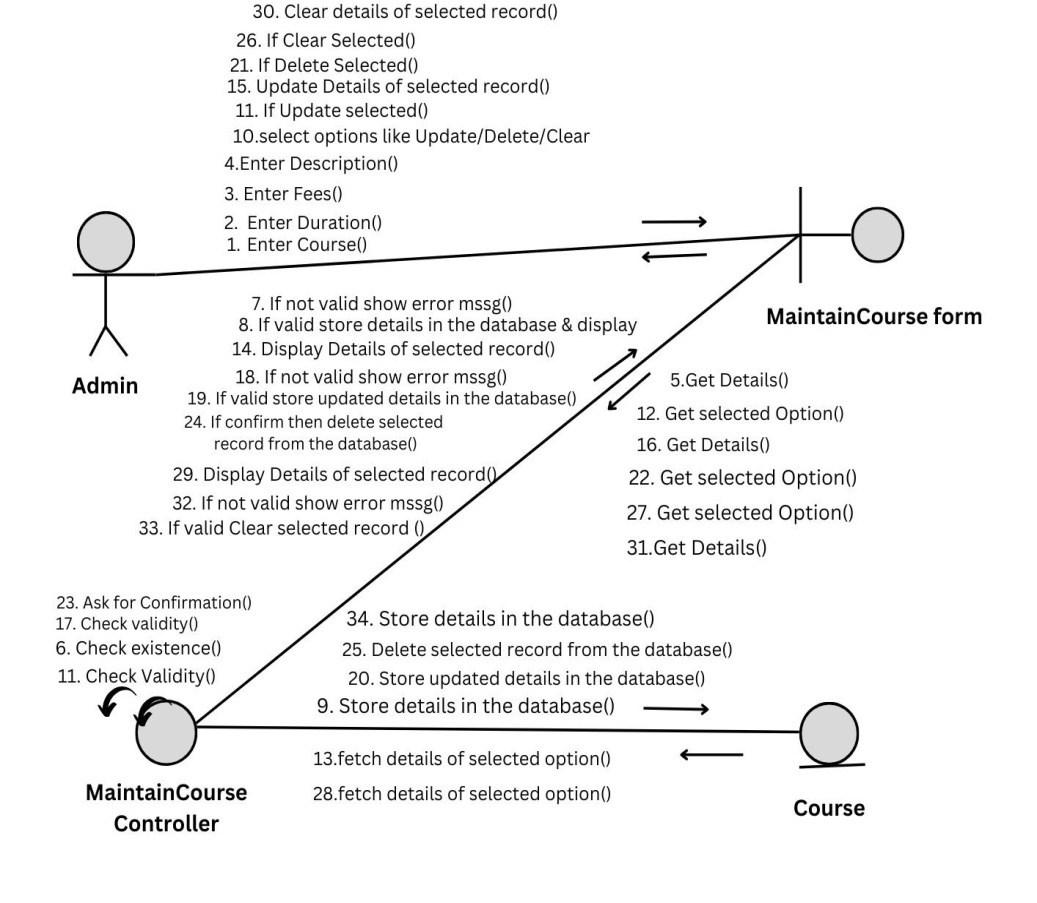
1. **Activity Diagram for MaintainCourse**

****

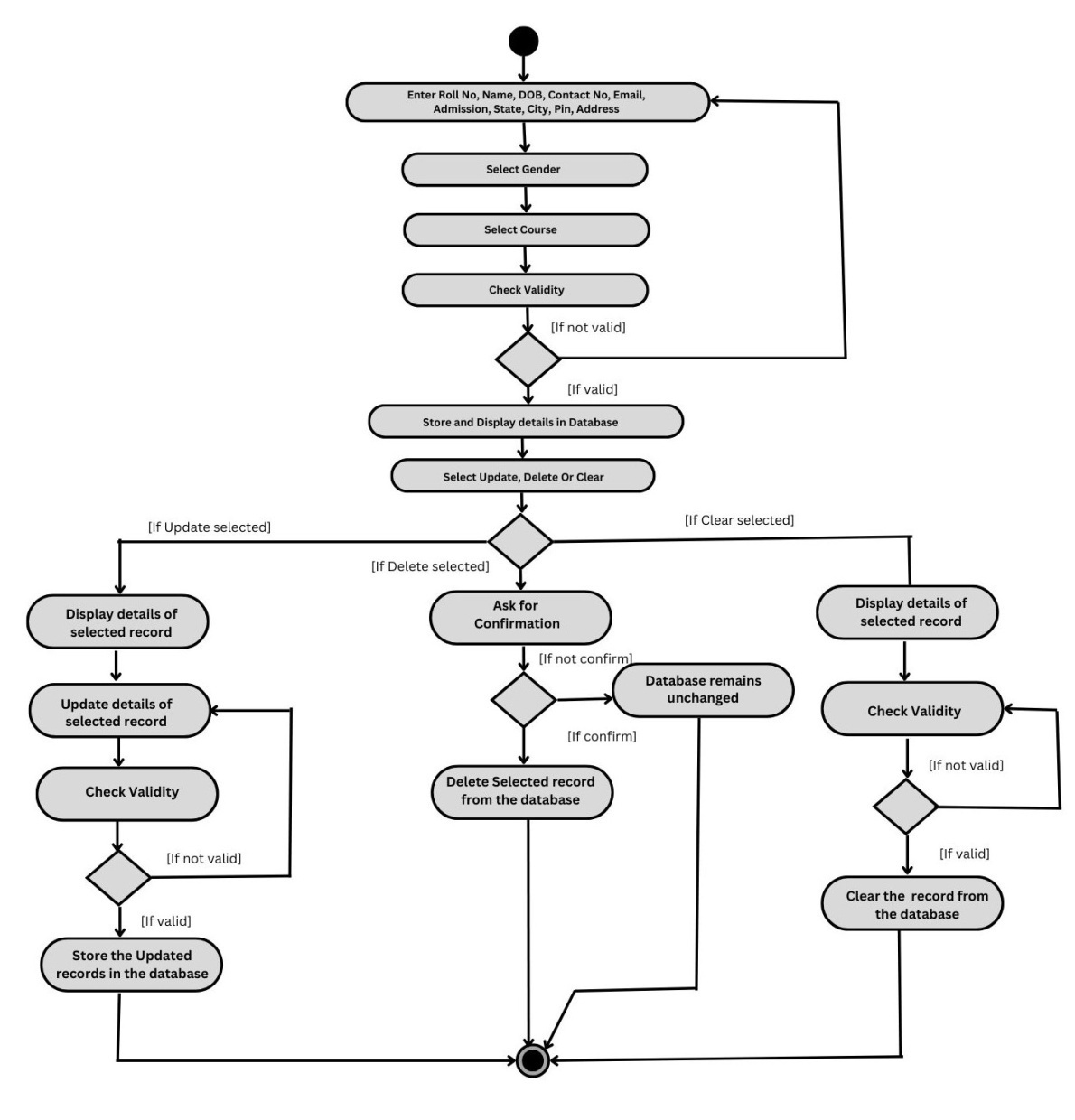
1. **Sequence Diagram for MaintainCourse**



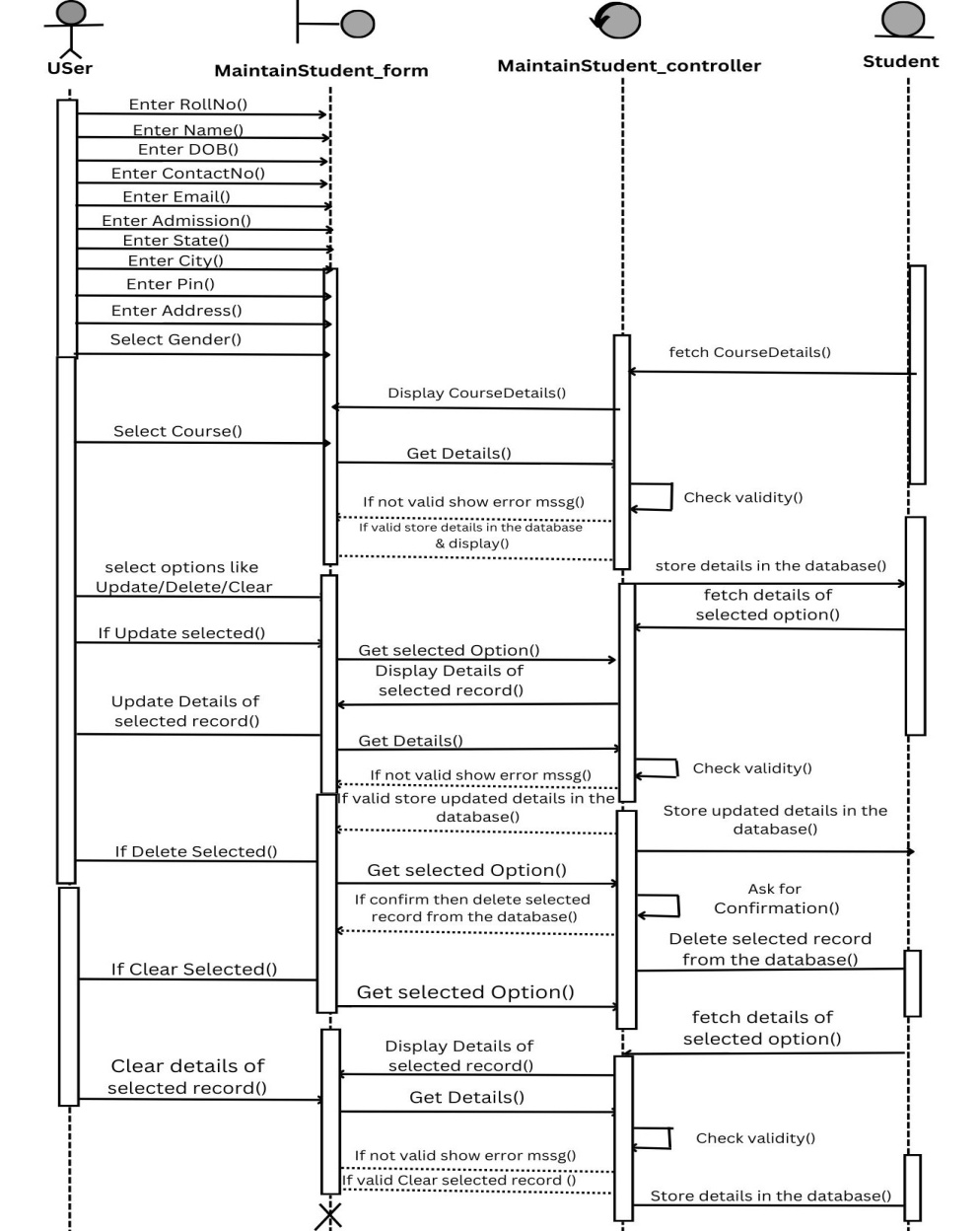
1. **Collaboration Diagram for MaintainCourse**



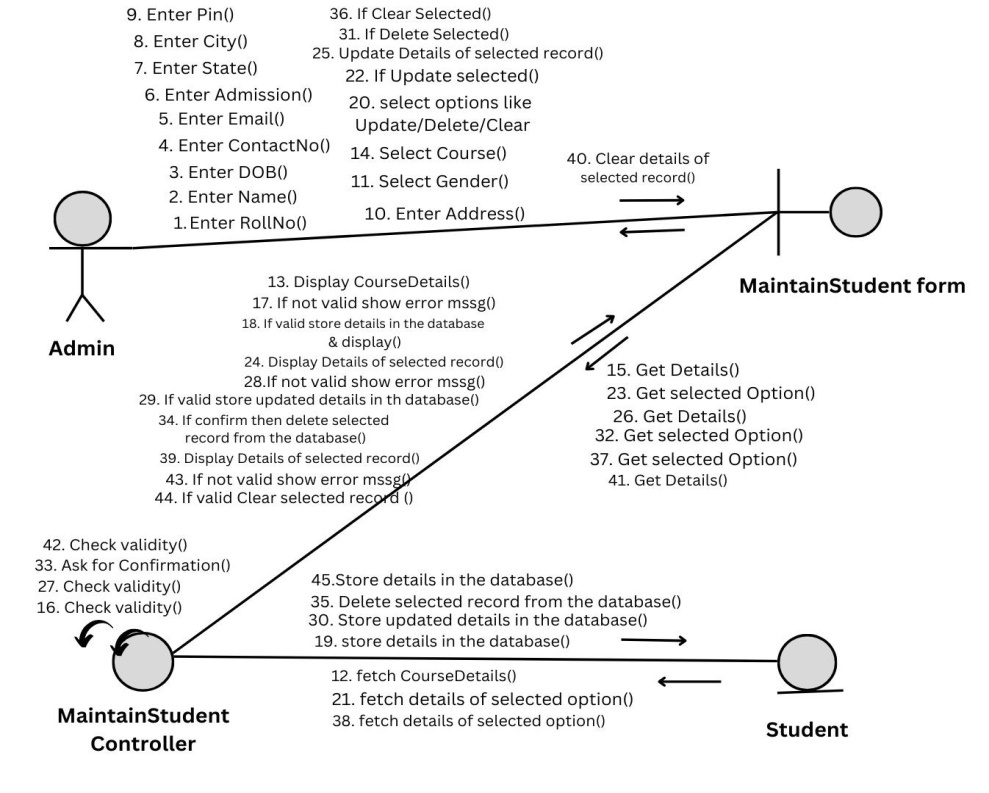
1. **Activity Diagram for MaintainStudent**

****

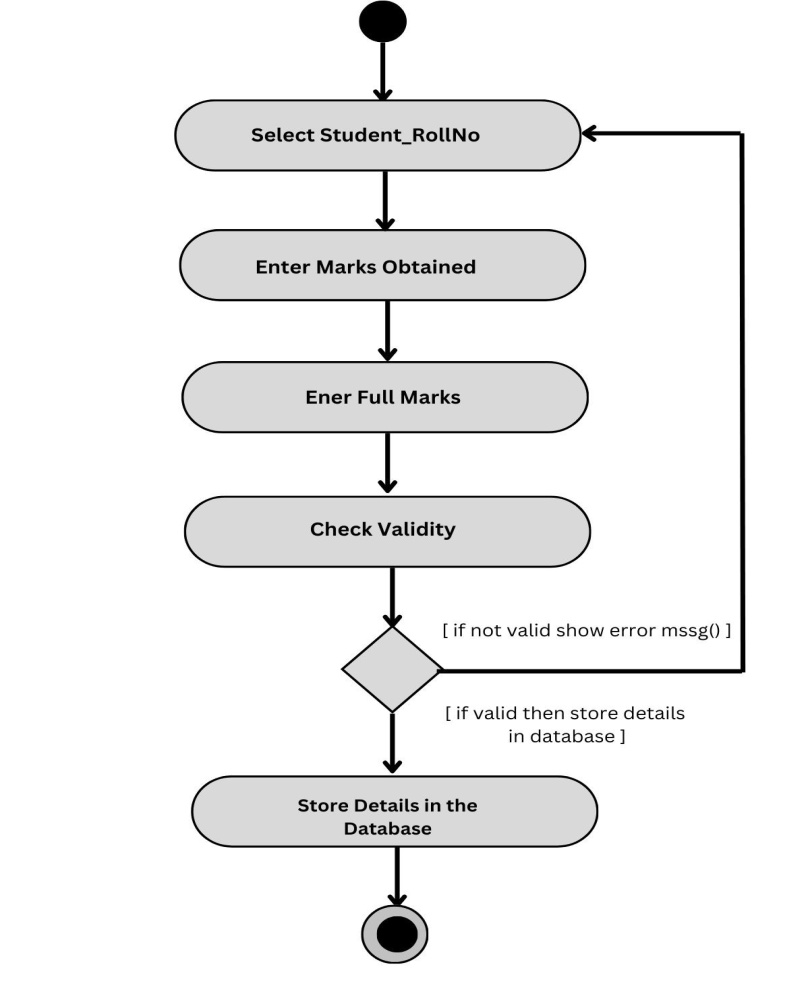
1. **Sequence Diagram for MaintainStudent**

****

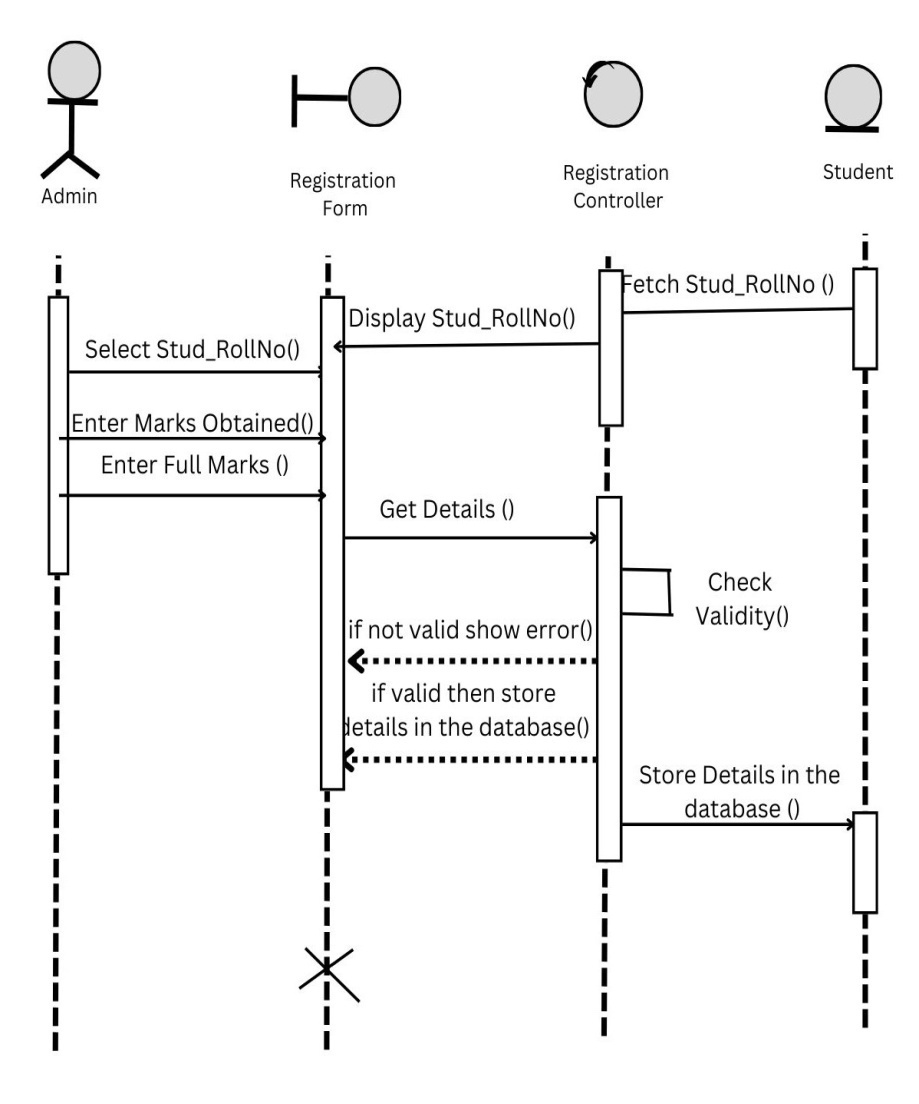
1. **Collaboration Diagram for MaintainStudent**



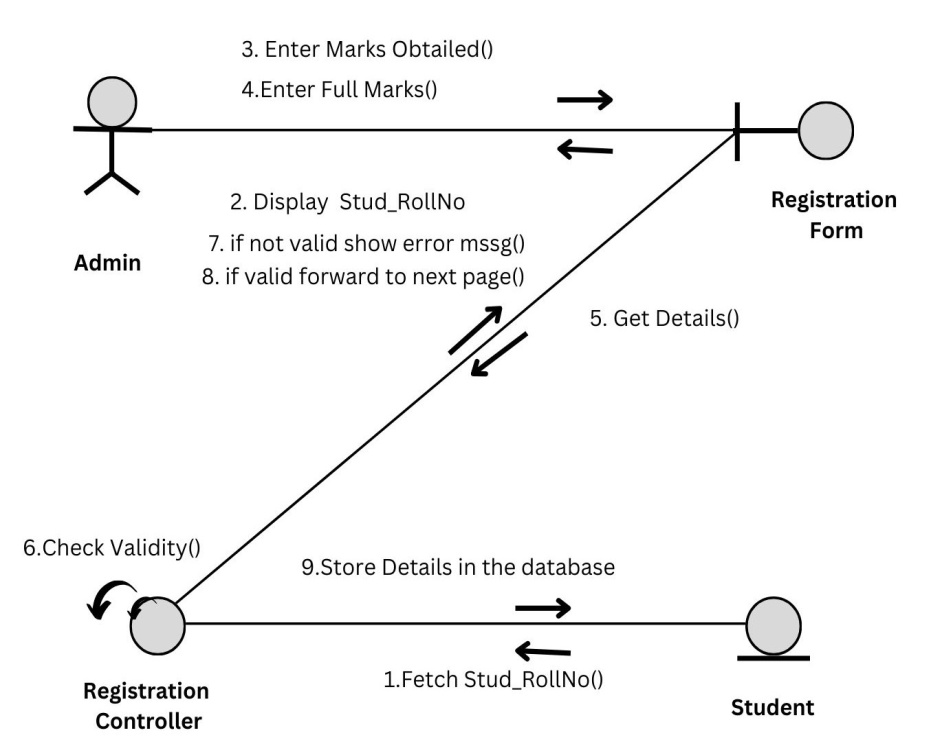
1. **Activity Diagram for AddResult**

****

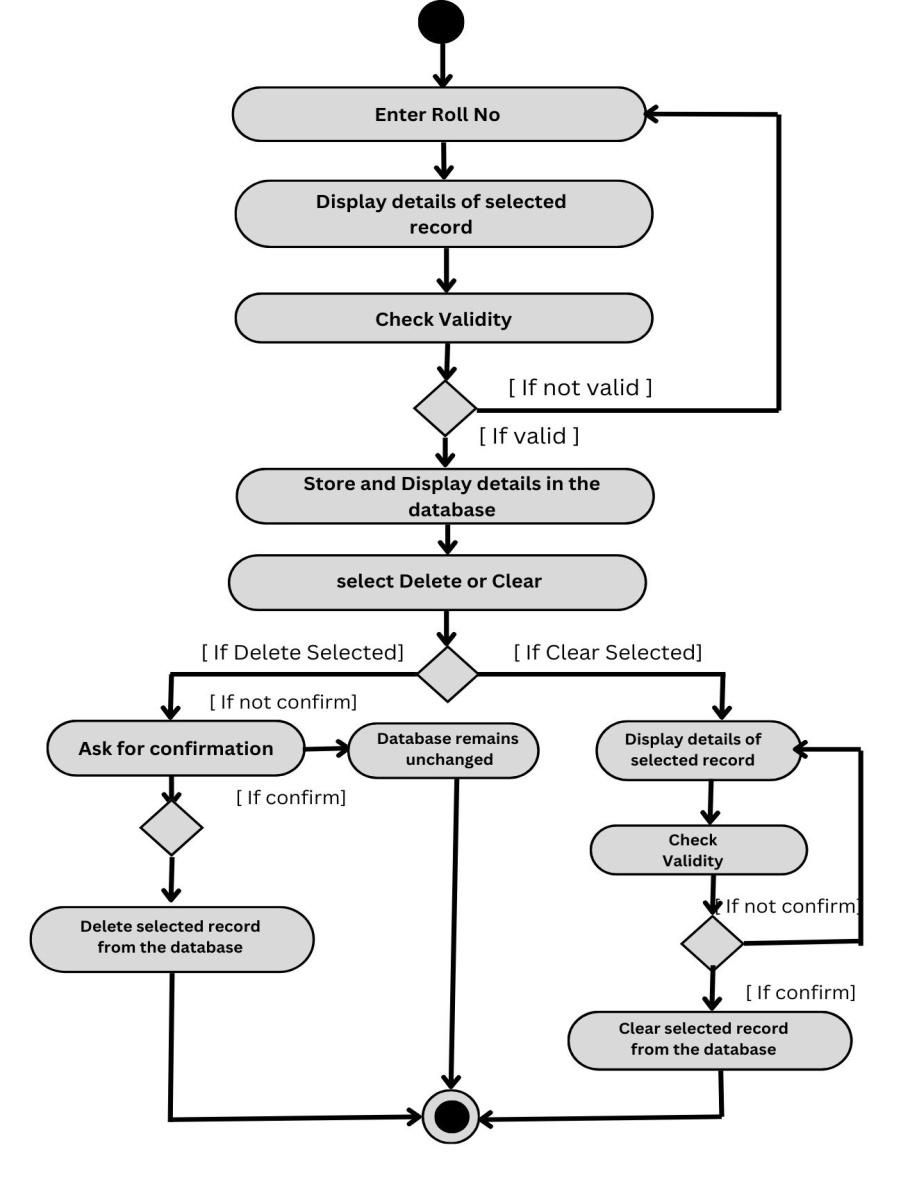
1. **Sequence Diagram for AddResult**

****

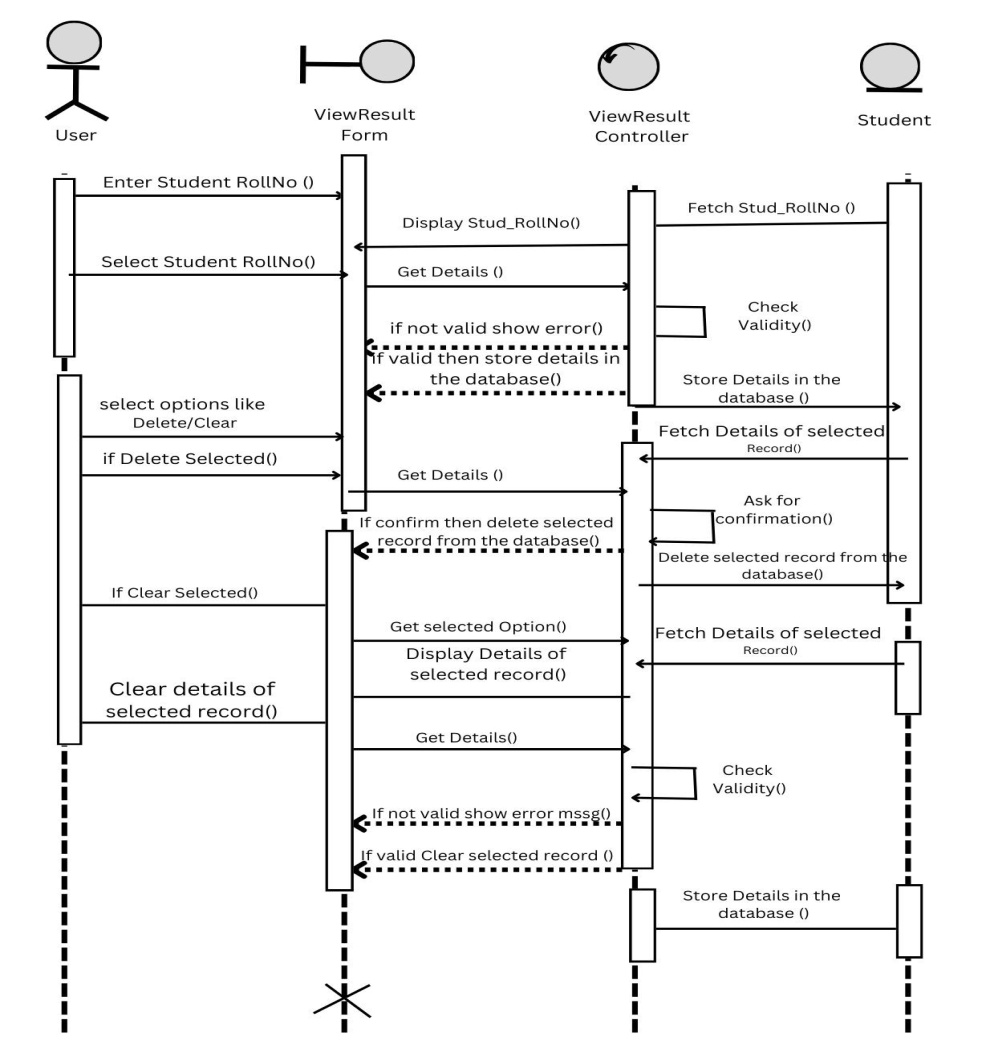
1. **Collaboration Diagram for AddResult**

****

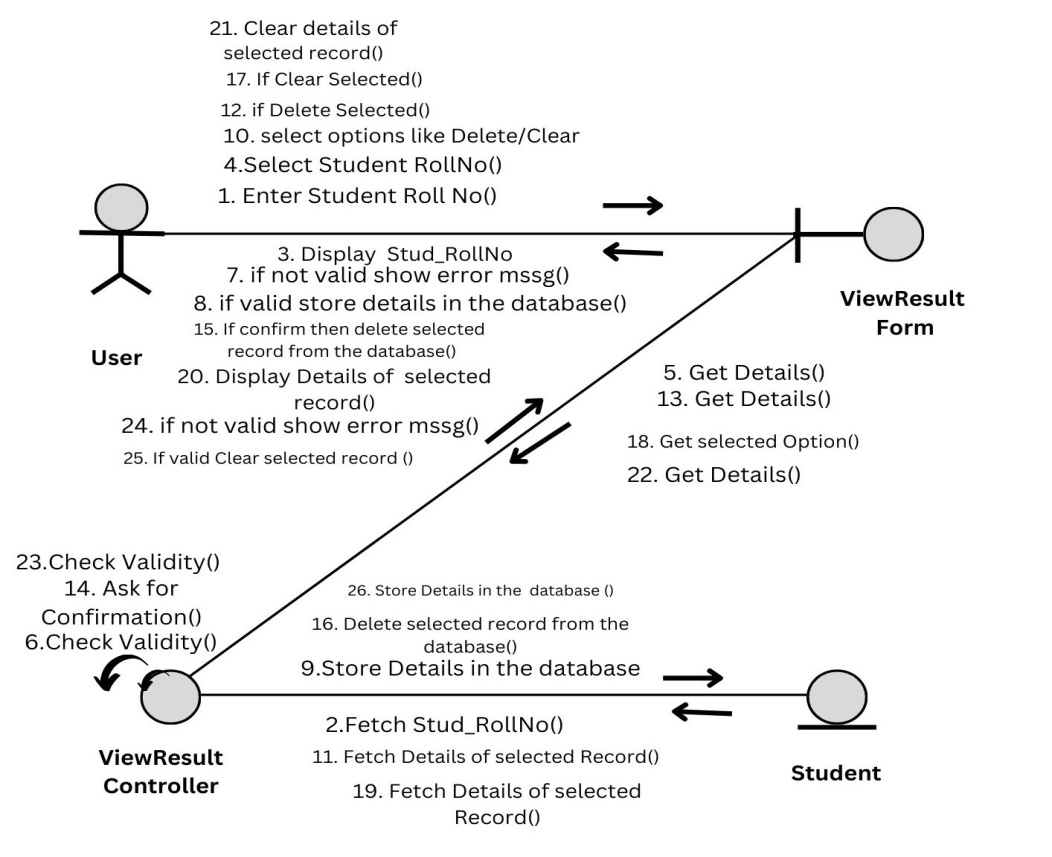
1. **Activity Diagram for ViewResult**



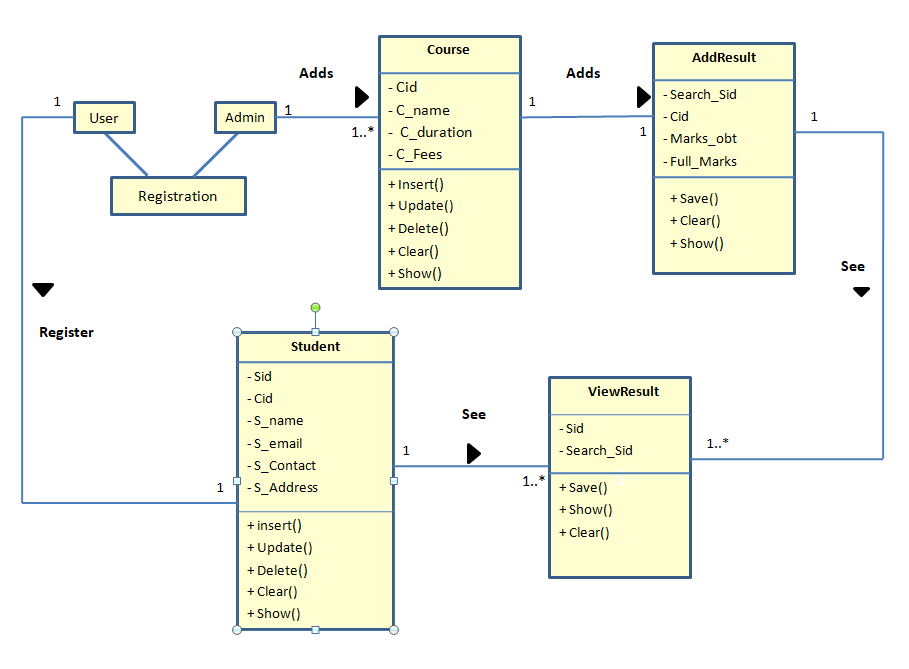
1. **Sequence Diagram for ViewResult**

****

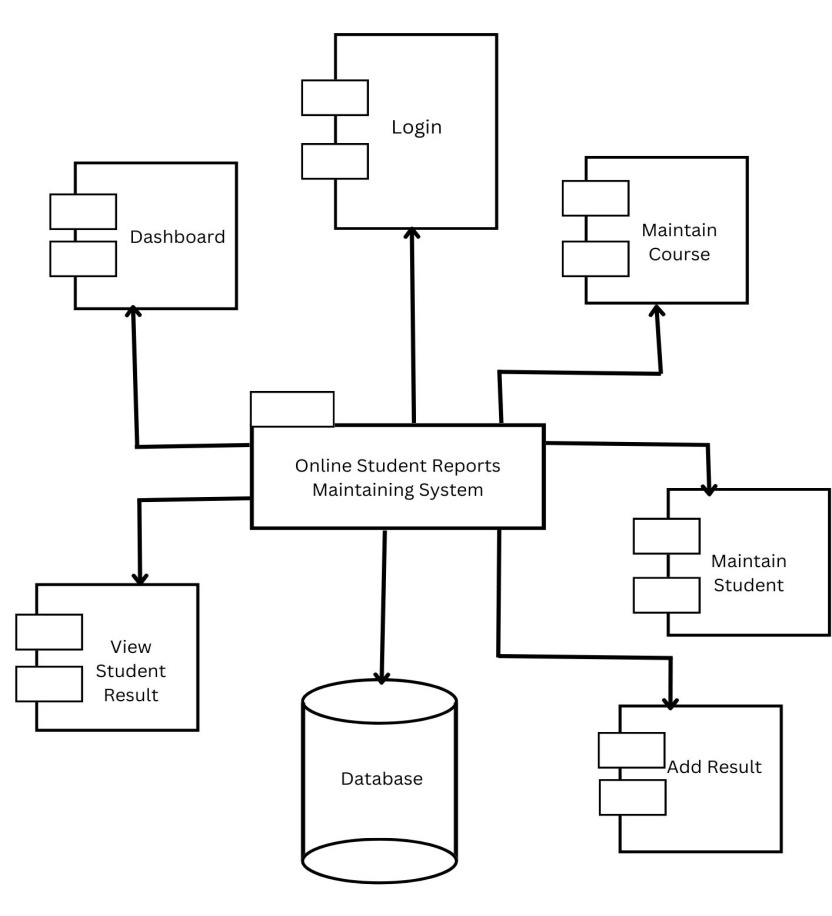
1. **Collaboration Diagram for ViewResult**

****

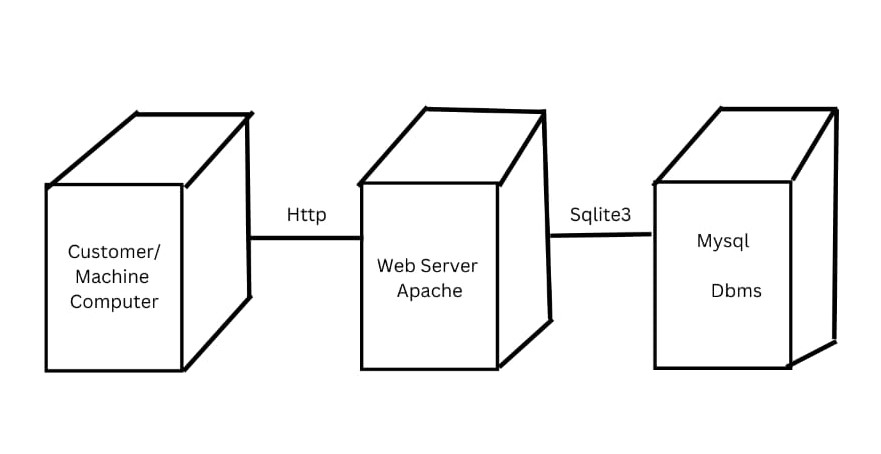
**19. Class Diagram**

****

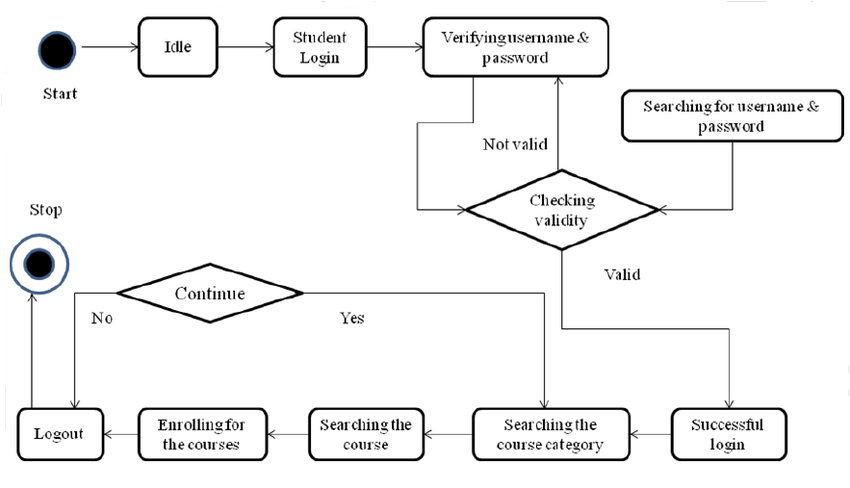
**20. Component Diagram**

****

**21. Deployment Diagram**

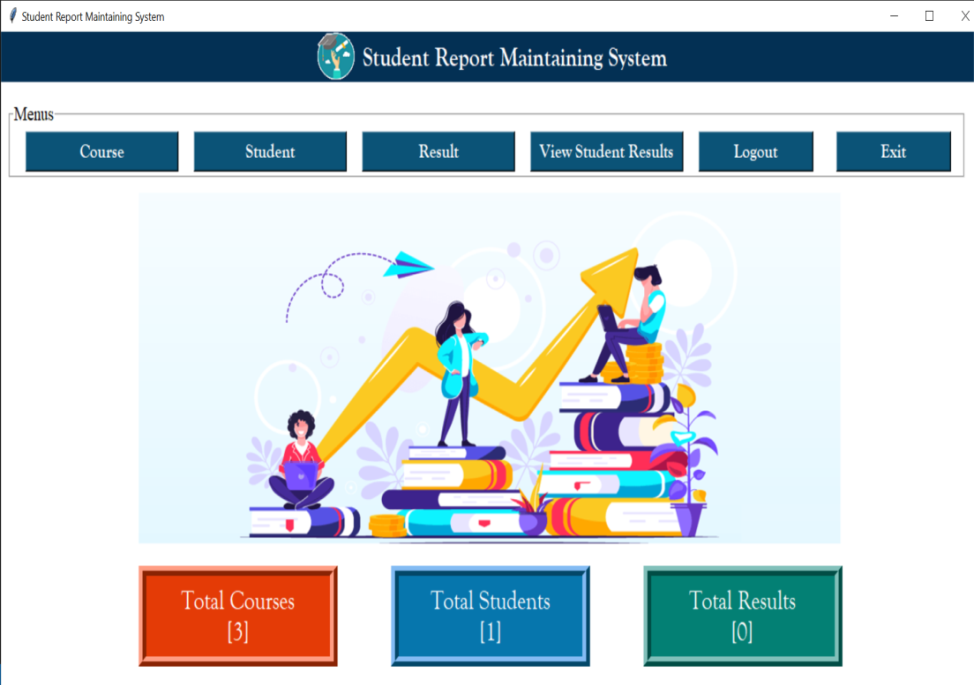
****

**22. State Transition Diagram**

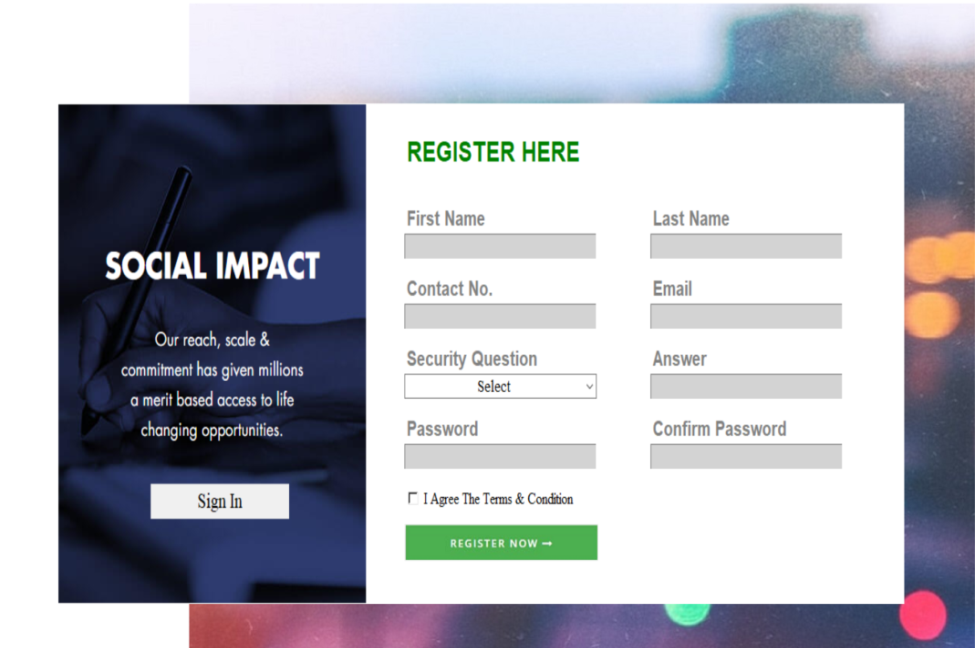
****

**User Interface Designs–**

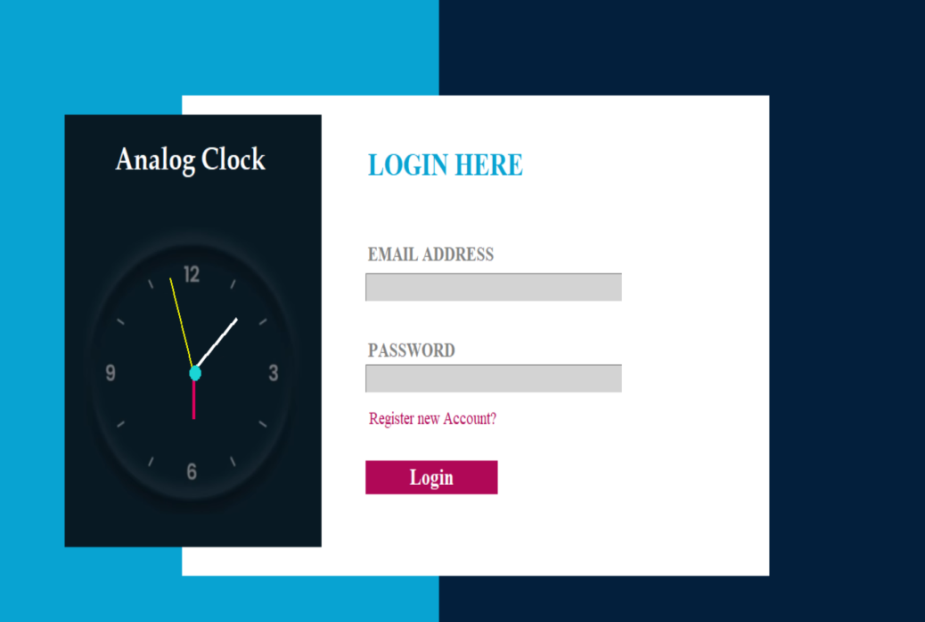
1. **Dashboard**

****

1. **Registration Form**

****

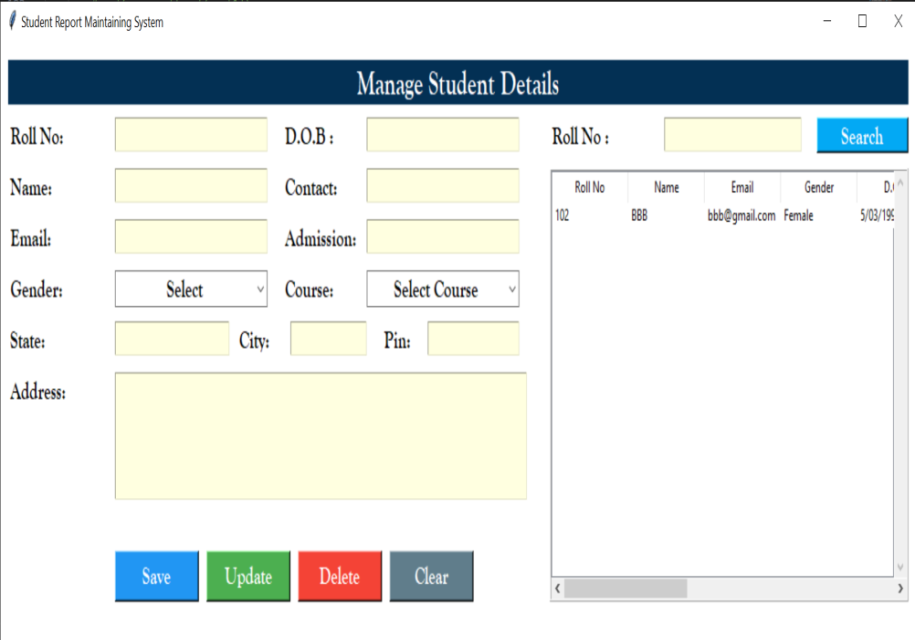
1. **Login Form**

****

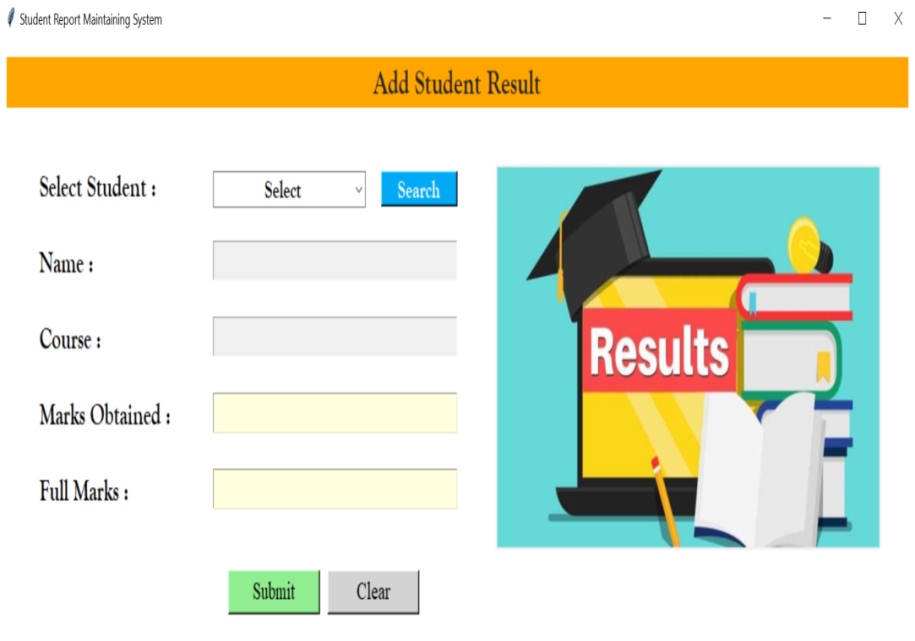
1. **MaintainCourse Form**

****

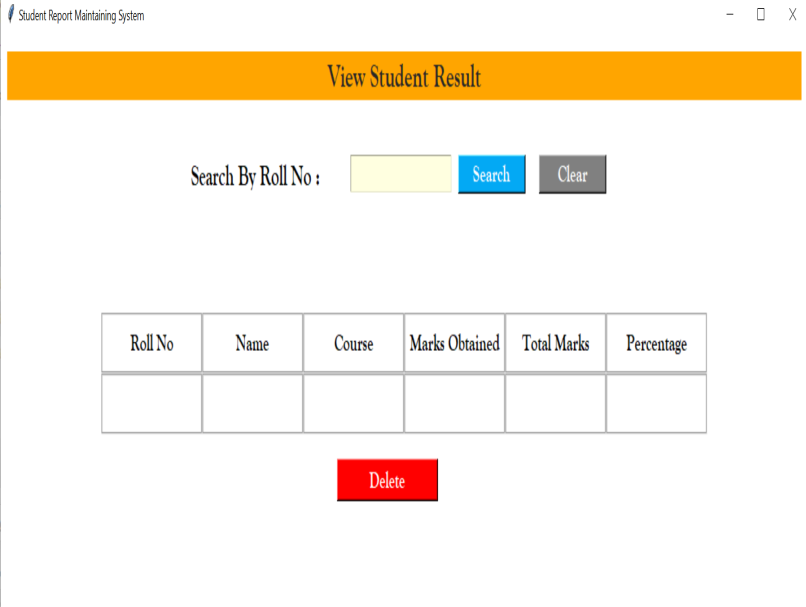
1. **MaintainStudent Form**

****

1. **AddResult Form**

****

1. **ViewResult Form**

****

**Sample Code –**

1. **Course.py**

 def add(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var\_course.get()=="":

                messagebox.showerror("Error","Course name should be required",parent=self.root)

            else:

                cur.execute("select \* from course where name=?",(self.var\_course.get(),))

                row=cur.fetchone()

                if row!=None:

                    messagebox.showerror("Error","Course name already present",parent=self.root)

                else:

                    cur.execute("insert into course (name,duration,charges,description) values(?,?,?,?)",(

                        self.var\_course.get(),

                        self.var\_duration.get(),

                        self.var\_charges.get(),

                        self.txt\_description.get("1.0",END)

                    ))

                    con.commit()

                    messagebox.showinfo("Success","Course Added Successfully !!",parent=self.root)

                    self.show()

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def update(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var\_course.get()=="":

                messagebox.showerror("Error","Course name should be required",parent=self.root)

            else:

                cur.execute("select \* from course where name=?",(self.var\_course.get(),))

                row=cur.fetchone()

                if row==None:

                    messagebox.showerror("Error","Select Course from list",parent=self.root)

                else:

                    cur.execute("update  course set duration=?,charges=?,description=? where name=?",(

                        self.var\_duration.get(),

                        self.var\_charges.get(),

                        self.txt\_description.get("1.0",END),

                        self.var\_course.get()

                    ))

                    con.commit()

                    messagebox.showinfo("Success","Course Updated Successfully !!",parent=self.root)

                    self.show()

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to {str(ex)}")

1. **Student.py**

def add(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var\_roll.get()=="":

                messagebox.showerror("Error","Roll No. should be required",parent=self.root)

            else:

                cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

                row=cur.fetchone()

                if row!=None:

                    messagebox.showerror("Error","Roll No. already present",parent=self.root)

                else:

                    cur.execute("insert into student(roll,name,email,gender,dob,contact,admission,course,state,city,pin,address) values(?,?,?,?,?,?,?,?,?,?,?,?)",(

                        self.var\_roll.get(),

                        self.var\_name.get(),

                        self.var\_email.get(),

                        self.var\_gender.get(),

                        self.var\_dob.get(),

                        self.var\_contact.get(),

                        self.var\_a\_date.get(),

                        self.var\_course.get(),

                        self.var\_state.get(),

                        self.var\_city.get(),

                        self.var\_pin.get(),

                        self.txt\_address.get("1.0",END)

                    ))

                    con.commit()

                    messagebox.showinfo("Success","Student Added Successfully !!",parent=self.root)

                    self.show()

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def update(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var\_roll.get()=="":

                messagebox.showerror("Error","Roll No. should be required",parent=self.root)

            else:

                cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

                row=cur.fetchone()

                if row==None:

                    messagebox.showerror("Error","Select student from list",parent=self.root)

                else:

                    cur.execute("update student set name=?,email=?,gender=?,dob=?,contact=?,admission=?,course=?,state=?,city=?,pin=?,address=? where roll=?",(

                        self.var\_name.get(),

                        self.var\_email.get(),

                        self.var\_gender.get(),

                        self.var\_dob.get(),

                        self.var\_contact.get(),

                        self.var\_a\_date.get(),

                        self.var\_course.get(),

                        self.var\_state.get(),

                        self.var\_city.get(),

                        self.var\_pin.get(),

                        self.txt\_address.get("1.0",END),

                        self.var\_roll.get(),

                    ))

                    con.commit()

                    messagebox.showinfo("Success","Student Updated Successfully !!",parent=self.root)

                    self.show()

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to {str(ex)}")

def delete(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var\_roll.get()=="":

                messagebox.showerror("Error","Roll No. should be required",parent=self.root)

            else:

                cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

                row=cur.fetchone()

                if row==None:

                    messagebox.showerror("Error","Please select student from the list first!! ",parent=self.root)

                else:

                   op=messagebox.askyesno("Confirm","Do you really want to delete?",parent=self.root)

                   if op==True:

                       cur.execute("Delete from student where roll=?",(self.var\_roll.get(),))

                       con.commit()

                       messagebox.showinfo("Delete","Student Deleted Successfully!!",parent=self.root)

                       self.deleted()

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to {str(ex)}")

1. **Result.py**

  def add(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var\_name.get()=="":

                messagebox.showerror("Error","Please first search student record !!",parent=self.root)

            else:

                cur.execute("select \* from result where roll=? and course=?",(self.var\_roll.get(),self.var\_course.get()))

                row=cur.fetchone()

                if row!=None:

                    messagebox.showerror("Error","Result already present",parent=self.root)

                else:

                    per=(int(self.var\_marks.get())\*100)/int(self.var\_full\_marks.get())

                    cur.execute("insert into result (roll,name,course,marks\_ob,full\_marks,per) values(?,?,?,?,?,?)",(

                        self.var\_roll.get(),

                        self.var\_name.get(),

                        self.var\_course.get(),

                        self.var\_marks.get(),

                        self.var\_full\_marks.get(),

                        str(per)

                    ))

                    con.commit()

                    messagebox.showinfo("Success","Result Added Successfully !!",parent=self.root)

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to {str(ex)}")

**Table Specifications –**

A Table Specification consists of formal specifications of all data types used, objects declared, and operations over database objects defined. The database specification is stored in the database and can be used for the specification of an application program operating with it.

1. **Registration Table –**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Rid(Primary key)** | **FirstName**  **(Varchar)** | **LastName**  **(Varchar)** | **Contact**  **(int)** | **Email**  **(Varchar)** | **question**  **(Varchar)** | **Answer**  **(Varchar)** | **password**  **(Varchar)** |
| 1 | ABC | EEE | 1234567873 | abc@gmail.com | Your First Pet Name | xyz | abc@123 |
| 2 | Sumitra | Samal | 8484885450 | sumitrasaml99@gmail.com | Your Best Friend Name | Sumitra | Shva@2003 |
| 3 | Bhagyashree | Samal | 9021183411 | bhagya05@gmail.com | Your First Pet Name | Cat | Bhagya |
| 4 | qer | rer | 2345670997 | asdf@gmail.com | Your Birth Place | nagar | 123 |
| 5 | Sharyu | Kale | 1234567890 | sharyu@gmail.com | Your Best Friend Name | Prasanna | 000 |

1. **Course Table –**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cid** | **Name** | **Duration** | **Charges** | **Description** |
| 4 | Java | 4 months | 6000 | Simple |
| 5 | Python | 8months | 10000 | Easy |
| 8 | C | 3 months | 3000 | It is easy to learn and understand. |

1. **Student Table –**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Roll** | **Name** | **email** | **gender** | **DOB** | **Contact** | **Admission** | **Course** | **State** |
| 102 | BBB | bbb@gmail.com | Female | 5/03/1990 | 9876543213 | 8/02/2005 | C | sdted |
| 103 | PDB | pallavi@gmail.com | Female | 15/01/2005 | 9812345678 | 12/04/2023 | C | Maharashtra |

|  |  |  |
| --- | --- | --- |
| **City** | **Pin** | **Address** |
| eref | 414003 | shreyas colony, a.nagar |
| Ahmednagar | 414003 | Shramik nagar. |

1. **Result Table –**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rid** | **Roll** | **name** | **course** | **Marks\_ob** | **Full\_marks** | **Per** |
| 4 | 102 | BBB | C | 78 | 100 | 78.0 |
| 6 | 103 | PDB | C | 85 | 100 | 85.0 |

**Drawbacks and Limitations –**

* The student result management system is prone to hacks.
* Administration cannot edit or modify scores after the deadline.
* Extensive modules and features make it difficult for a user to utilize the application.
* Minor technical glitches and issues.
* Calculate scores, percentages and grades.
* Record and search details of every exam/test, student wise/course wise.
* Add & manage students and Declare Results.
* Records marks and result on a single database.

**Proposed Enhancement –**

* The first step is to identify the user requirements for the system. This includes identifying the stakeholders (e.g., administrators, teachers, students) and their needs. For example, administrators may need to manage student records, while teachers may need to enter grades and attendance.
* The next step is to develop a database schema that represents the various entities in the system, such as students, courses, grades, and attendance. This can be done using a database management system such as MySQL or sqlite3.
* The user interface can be developed using python. The interface should allow users to access the various functions of the system, such as entering grades, viewing attendance records, and generating reports.
* The backend should include functions for adding and retrieving data from the database, as well as functions for performing calculations, such as calculating grades.
* To ensure data privacy and security, the system should implement measures such as user authentication and authorization, encryption of sensitive data, and secure connections.
* The system should be tested thoroughly to ensure that it meets the user requirements and functions correctly. Once the system has passed testing, it can be deployed on a server or cloud platform for use by the stakeholders.