INSTITUTE MANAGEMENT SYSTEM

Diploma in Software Engineering

Final Project Documentation

National Institute of Business Management
Kandy Regional Center
No: 2, Asgiri Vihara Mawatha,
Kandy

K.D.D. Dissanayake KADSE222F-004
Y.S. Panannala KADSE222F-031
S.A.S. Dhanayake KADSE222F-016
K.V. Maheesha Sulakshana KADSE222F-039

2023

INSTITUTE MANAGEMENT SYSTEM

K.D.D. Dissanayake	KADSE222F-004
Y.S. Panannala	KADSE222F-031
S.A.S. Dhanayake	KADSE222F-016
K.V.M. Sulakshana	KADSE222F-039

Diploma in Software Engineering

Supervisor:

Mr. Sandaruwan Herath

Management Information System Division,

National Institute of Business Management

"The project is submitted in partial fulfilment of the requirement of the Diploma in Software Engineering of the National Institute of Business Management."

December 2023

Declaration

"I certify that this project does not incorporate without acknowledgement, any material previously submitted for a Diploma in any institute and to the best of my knowledge and belief. It does not contain any material previously published or written by another person or by myself except where due reference is made in the text. I also hereby give consent for my project report, if accepted, to be made available for photocopying and interlibrary loans, and for the title and summary to be made available outside organizations."

Student Name	Index No	Signature
K.D.D. Dissanayake	KADSE222F-004	
Y.S. Panannala	KADSE222F-031	
S.A.S. Dhanayake	KADSE222F-016	
K.V.M. Sulakshana	KADSE222F-039	

Certified by

Supervisor: Mr. Sandaruwan Herath

Signature: ……

Date: ……

Course Directors:

Mr. Sandaruwan Herath

Signature: …….

Date: ……

Ms. Inoka Abhayasinghe

Signature: …….

Date: ……

Summary

Mr. Dhammika Thisara Sanchiarrachi who is the manager and sole lecturer at his Institute was facing problems with managing his records. So, to give a solution to his problem we build a computerized system to better manage his record keeping tasks.

All the Standard protocols were followed to build the application. And The Application features an easy to navigate user interface which has built in validations, and helpful features to ensure the validity of the data.

Also, our project features an additional component which is a Web Site for students to use. They can use it to ask questions, view notes, view results, view their schedule, and view answers to their questions.

A Local Database was implemented to store and access these data.

Overall, we believe that we have achieved the goal of our project which was to help Mr. Dhammika to free him from his time-consuming daily record keeping tasks with an automated system.

Acknowledgement

First, we would like to express our heartfelt gratitude to Mr. Dhammika Thisara Sanchiarrachi for giving us the opportunity to build this project.

Also, we would like to extend our thanks to Mr. Sandaruwan Herath, our Project Supervisor, and Ms. Inoka Abhayasinghe, and Mr. Sandaruwan Herath our Course Directors, for providing us with knowledge, skills, guidance, and support.

We would also like to thank all the instructors and staff at NIBM for their assistance in many ways.

Finally, we would like to express our gratitude to our families and friends, who have always supported us in every possible way.

Table of Contents

List of Figures

List of Tables

Contents

Chapter 1: Introduction

1.1 Introduction to the Organization

Lifeway Education Services, located in Nuwara Eliya, has been providing ICT education for children for over a decade. They cover everything from basic computer systems to advanced programming, helping students to navigate and learn everything that they need to know about ICT.

Dhammika Thisara Sanchiarrachi, who manages the institution all by himself, also is the lecturer for his students. Managing both the institution and the lessons has been more difficult than ever, as more and more students enroll in his classes each month. With over 50 students currently enrolled in his lessons, and new groups enrolling each month, his workload has increased to a point where he can't manage it all by himself.

The one thing that takes more time out of his day except teaching, is record-keeping tasks. Manually entering, updating, removing, and searching for records from a long list of students has been more difficult than ever before.

So, to give a solution for this problem we decided to build an application to better manage his day-to-day record-keeping tasks.

1.2 Organizational Structure

In the current organizational structure, Dhammika Thisara Sanchiarrachi manages the entire operation of the Institute all by himself. Including recording data, maintaining them, and everything else.

1.3 Current Operation Method

Currently, all data and information regarding every part of the organization are recorded and handled manually. Such as registering students, marking attendance, viewing student details, recording payments, etc.

1.4 Users and Responsibilities of Organization

Currently, the only users of the organization are the client and the students who come to learn in the institute.

Responsibilities of client

- Maintain all the records of the organization.
- Perform day-to-day checking of the records.

Responsibilities of students

- There are no identified responsibilities of students.

1.5 Problem Definition

Time Consuming manual data manipulation

The main problem that the client faces is to maintain all the records by hand.

Which takes more time than he likes to spend on it, resulting in reduced time that he can use to teach kids.

Error prone to manual mistakes

Very prone to have errors when referring and recording details from one book to another.

Daunting manual search operations

Searching for a record/records has become a difficult task, as he has to look at a long list of records to find results, and also he has to note down the details somewhere else to look at later.

1.6 Project Objectives

The main objective is to replace almost all the manual record-keeping tasks with an automated computerized system.

1.7 Proposed Solution

The proposed solution is to make a User-Friendly GUI application that the client can use to better manage his record-keeping jobs.

To achieve that, we will provide functionality to the application to automatically store, update, delete, and search for records more quickly, making it easier for him to do his work.

1.8 Chapter Summary

In this chapter we got to know about the purpose for the project, who were building the project for, the structure of the organization, its daily tasks, how they are performed, the people who is involved with it, their responsibilities, and the problems that they face.

Also, we discussed how we can provide a better solution for the problems that they're facing by providing a tailor-made application to better manage their day-to-day tasks.

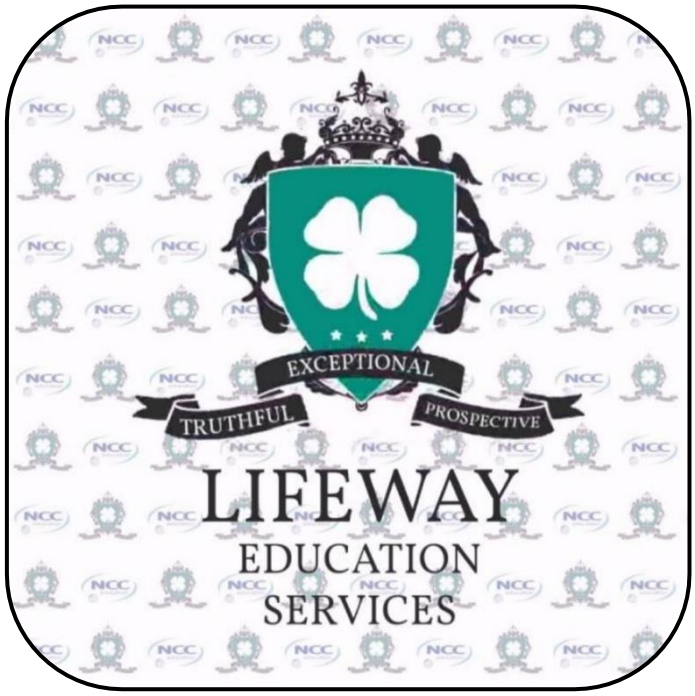

Figure 1: Lifeway Education Services

Chapter 2: Methodology

2.1 Introduction

In this chapter you will get to know about the project plan, data collection methods, software development method, software development tools, testing strategies, and the implementation plan that will be used for the application.

2.2 Project Plan

First, an interview with the client was conducted to understand the needs for the system. After that, a field study was conducted to fine tune his requirements. Next, past records of the institute were reviewed to collect what type of data he wants to be stored. After that, an SRS document was created for review. After it was accepted by the client, we began to build the system.

According to the requirements, designs were created to help us build the system. The components of the system were identified by understanding the organization and its operations. By using various diagrams, we were able to map out the organization clearly.

Then after that, the implementation phase began according to the designs. Various bugs and errors were identified along the way, but we resolved them and moved forward.

Finally, after passing through a thorough testing phase and making sure the application meets all the requirements, the system was successfully implemented in the client's environment.

The below plan was followed to complete the project.

Gathered the user requirements	*1 week*
Conducted a field study and reviewed past records	*1 week*
Prepared the SRS	*1 week*
Designed the System	*2 weeks*
Implemented the System	*2 weeks*
Conducted tests of the System	*½ week*
Prepared the documentation	*2 ½ weeks*
Finalized the system	*2 weeks*

Table 1: Project Timeline

2.3 Data Collection Methods

We gathered all the required information by using,

- Interviews with the client
- Observations of the Organization
- Reviewing Past Records of the organization

2.4 Software Development Method

To make the application as best as we can, we chose to follow the Waterfall model to guide us through the Software Development Life Cycle. We chose this model because it helps us to go through all the SDLC phases, phase by phase without compromising the quality of each stage.

1

Based on that first, an interview and a field study of the organization was conducted to better understand the requirements for the system and to understand what can be converted into an automated system and what cannot.

2

After that, requirements were confirmed with the client to make sure that the proposed system meets all his needs.

3

After that, the System was designed using Flow Charts, DFDs, Use Case Diagrams, Sequence Diagrams, ER Diagrams and more.

4

Next, the implementation begun. While having a strict testing method in place to help us achieve the expected quality of the system.

5

After the testing was completed, and making sure that everything works according to the client's needs and expectations. Working System was handed over to the Client with all the documentation.

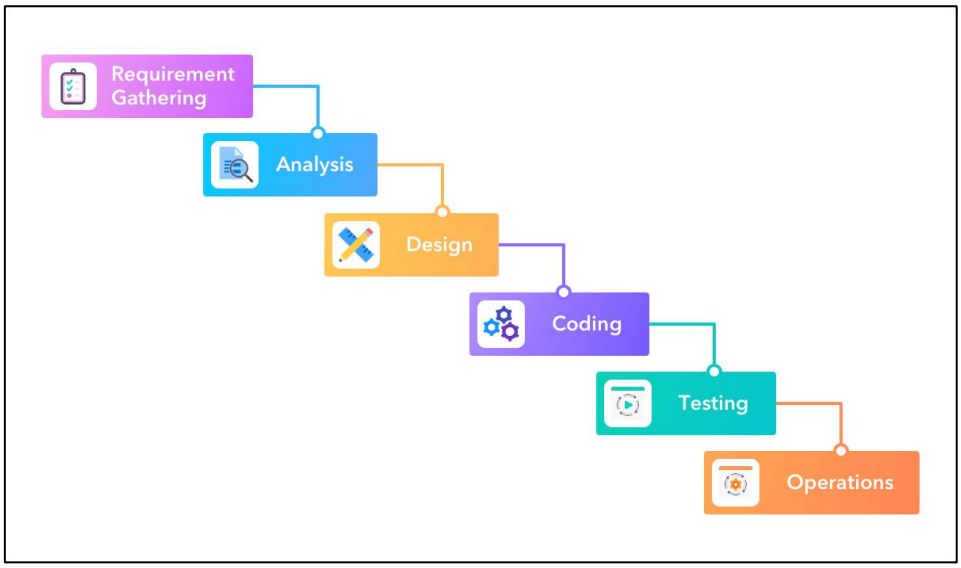

Figure 2: Software Development Method (Waterfall Method)

2.5 Software Development Tools

For our client's application we decided to build it using C# and we used Visual Studio IDE as the development environment.

And for the students' website, we decided to build it using HTML, CSS, and PHP, also we used Visual Studio Code as development environment.

Figure 3: Technologies used.

2.6 Testing Methods

Throughout the implementation process we tested every module, class, and integration to ensure our system performs as expected.

We conducted,

- White Box Testing
- Black Box Testing

for every part of the code.

2.7 Implementation plan

We plan to implement the system in the client's environment by following the parallel method which will allow us to use the current system and the new system at the same time for a planned amount of time which allows us to make sure our application performs exactly as planned. Also, we will use this time to identify any new improvements for the application before we completely implement it in the client's environment.

The advantage of using this method is that, if we encounter a problem in this time, we can quickly refer to the old system. Without having to stop operations immediately.

2.8 Chapter Summary

In this chapter we discussed the project plan, data collection methods, software development method, software development tools, testing strategies, and the implementation plan for the application.

Chapter 3: Analysis

3.1 Introduction

In this chapter includes diagrams which shows the current operational methods of the organization, the proposed operations of the organization, and the blueprint for the new system and for its components.

3.2 Diagrams

Here we will see the diagrams that explain the current system and the proposed system for the client.

Use case diagram for the current system

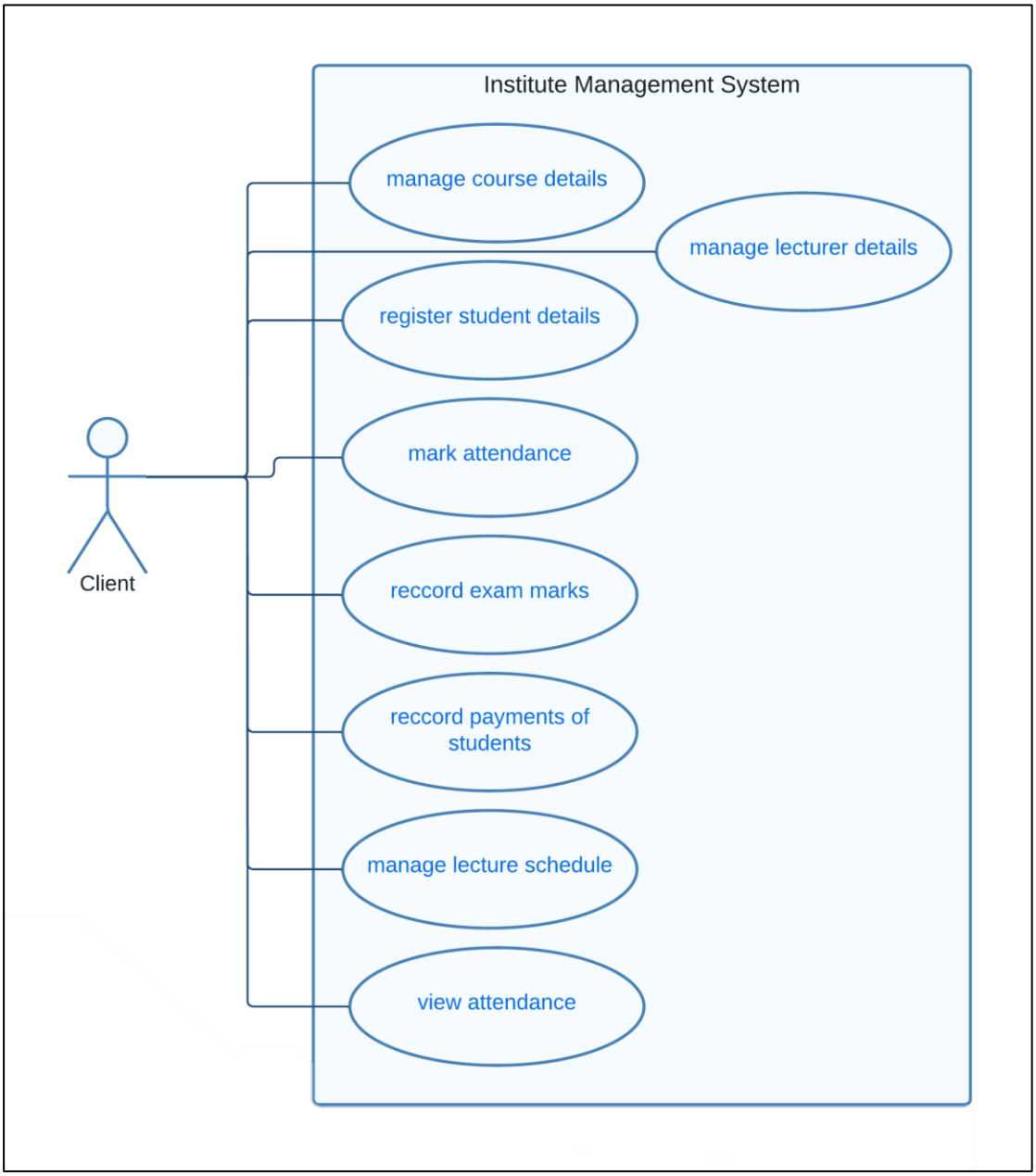

Figure 4: Use Case Diagram for the current system

This diagram shows the current system in place and it's uses

Use Case Diagram for the proposed system.

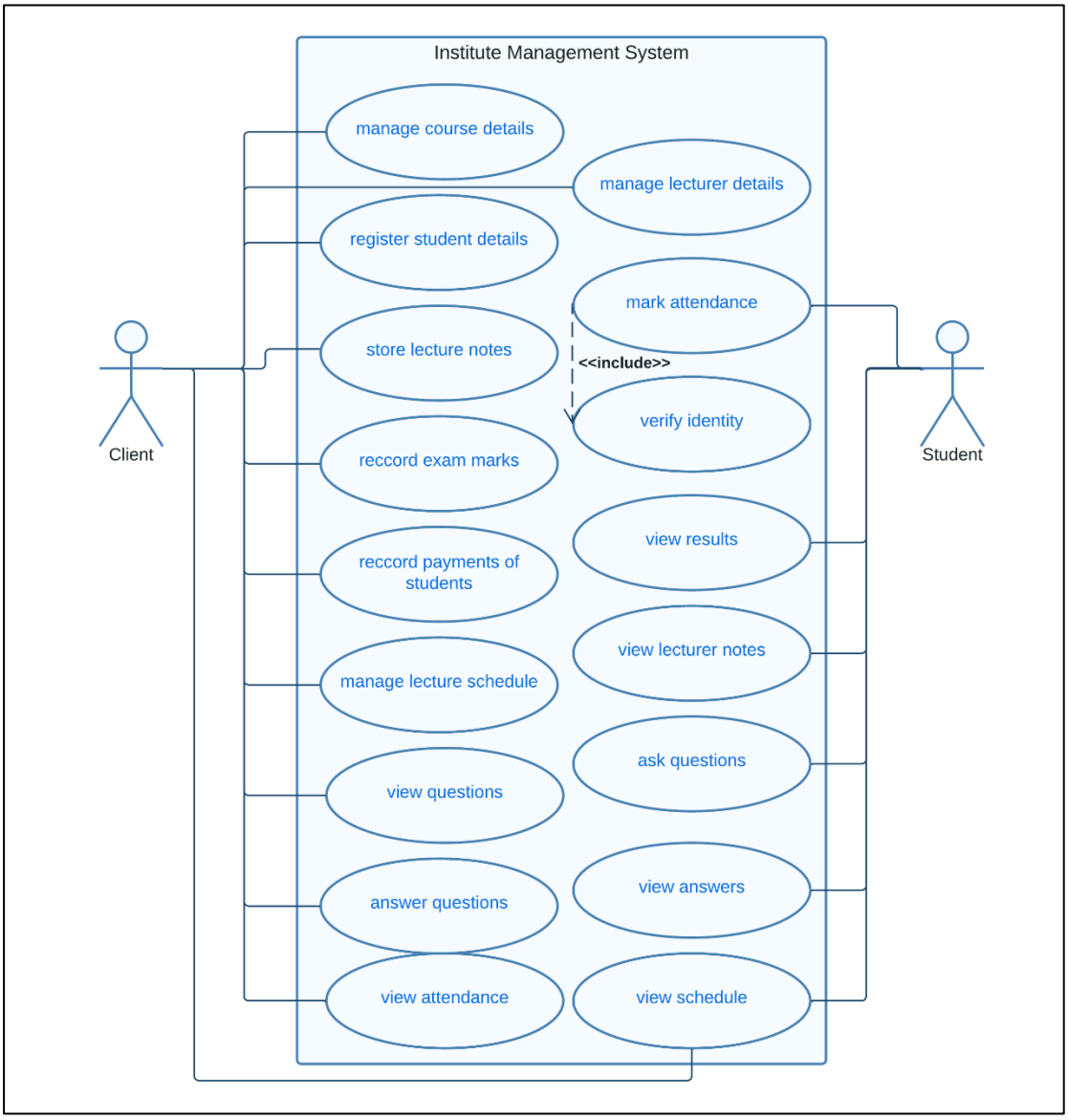

Figure 5: Use Case Diagram for the proposed system.

This diagram shows the proposed system and it's uses

DFD Diagram (Context Level)

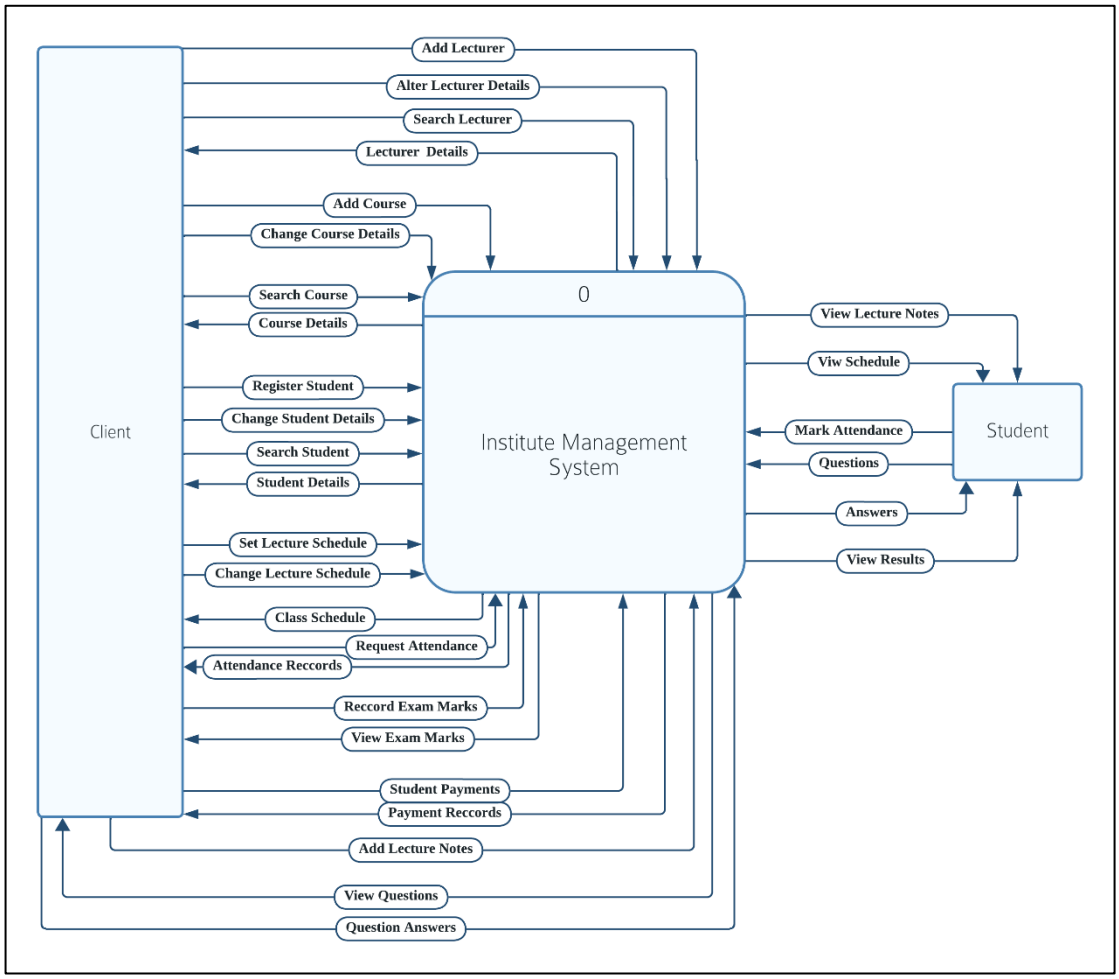

Figure 6: DFD Diagram (Context Level)

This diagram shows the flow of data through the system in a higher level

DFD Diagram (Level 1)

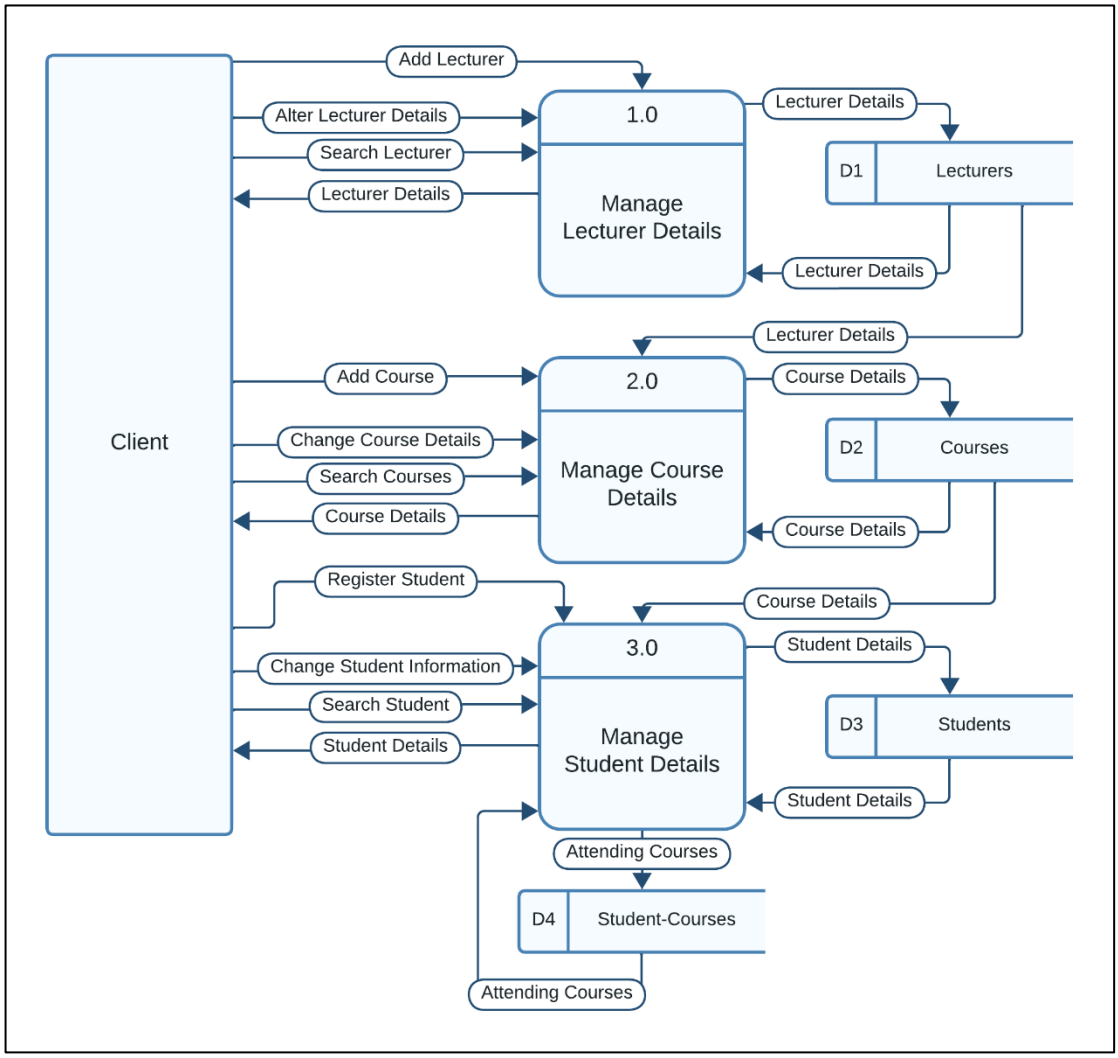

Figure 7: DFD Diagram (Level 1)

This diagram shows the flow of data through when managing lecturer details, course details, student details

DFD Diagram (Level 1)

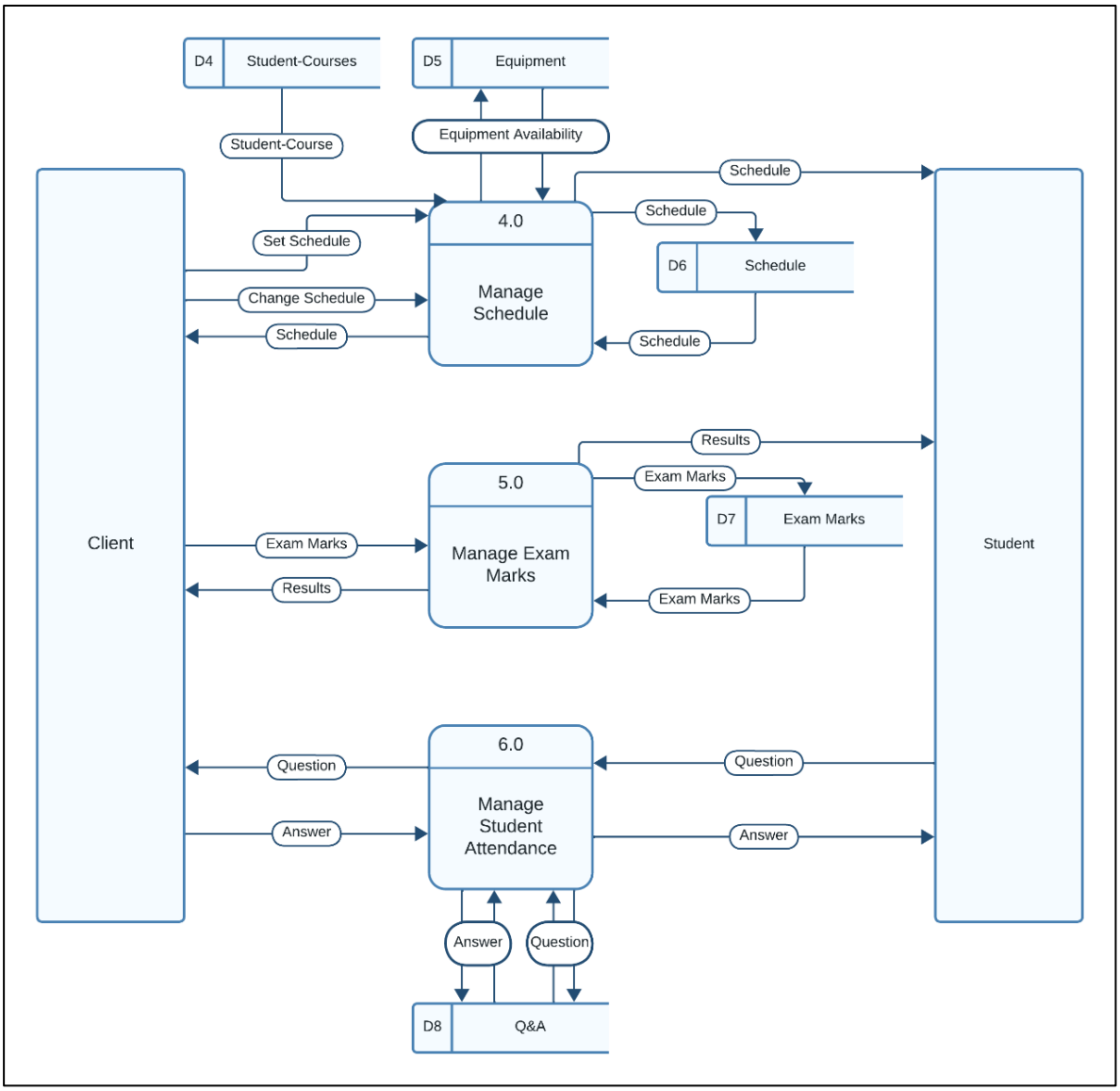

Figure 8: DFD Diagram (Level 1)

This diagram shows the flow of data through when managing schedule details, exam details, student questions

DFD Diagram (Level 1)

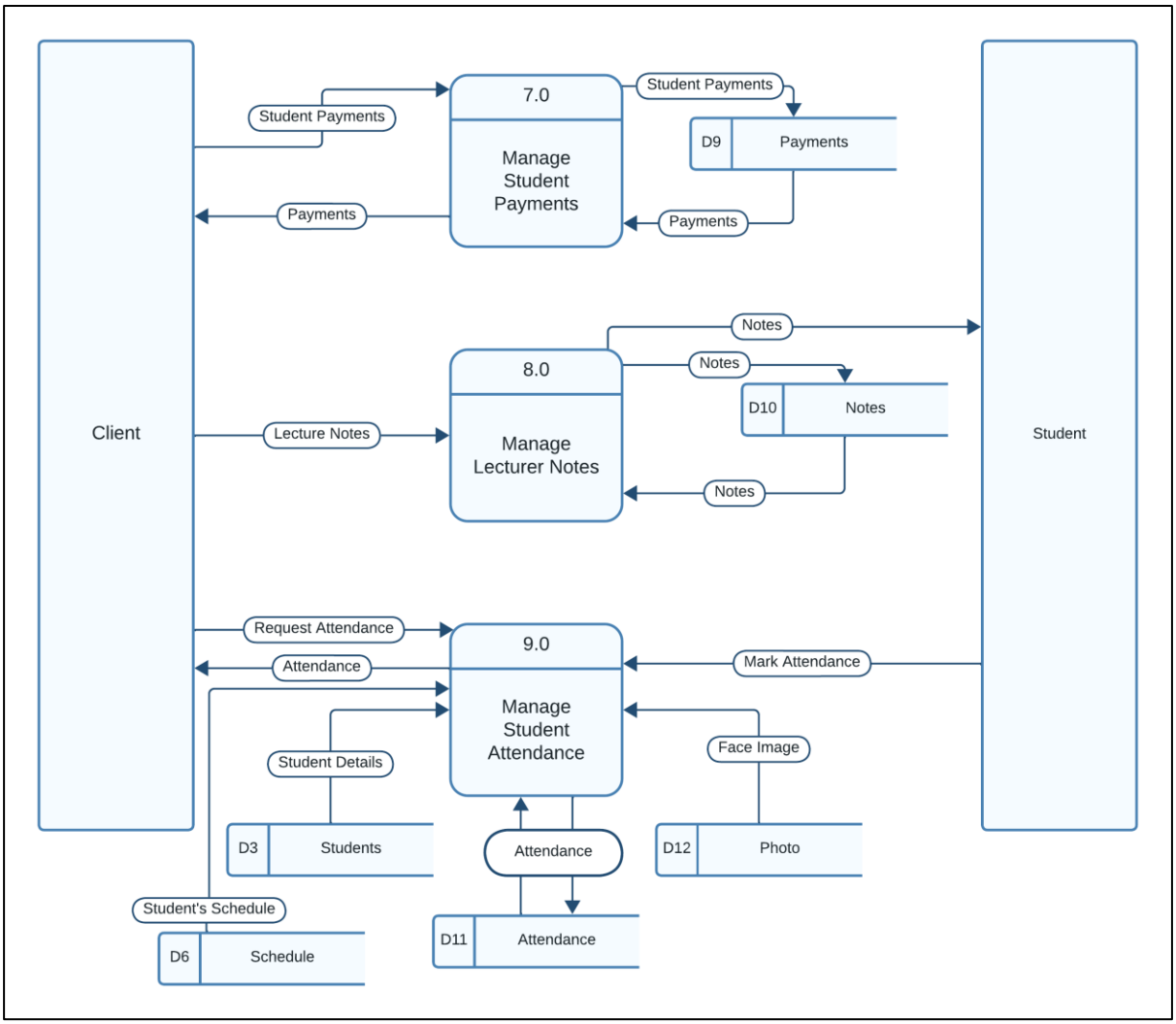

Figure 9: DFD Diagram (Level 1)

This diagram shows the flow of data through when managing payment details, lecture notes , student attendance

Class Diagram

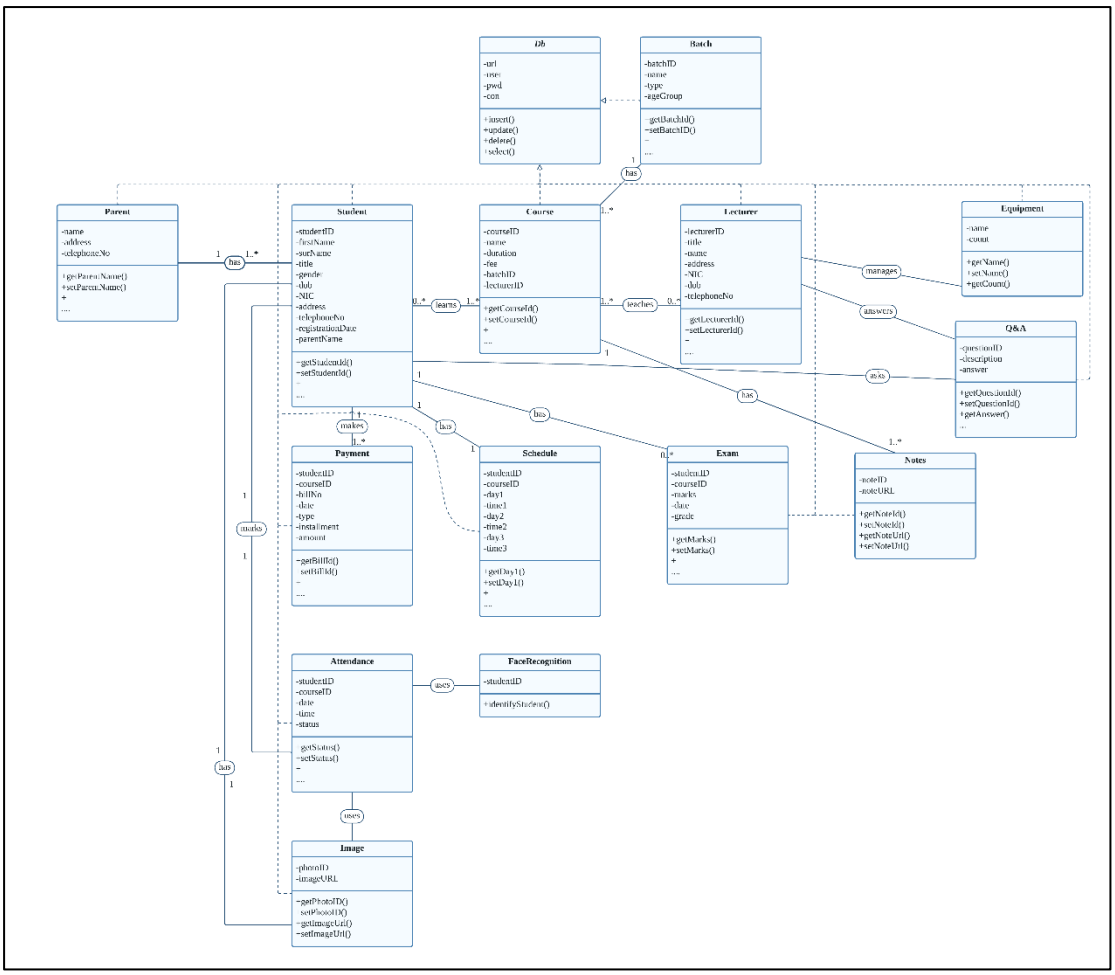

Figure 10: Class Diagram

This diagram shows the classes that is implemented in the system

Class Diagram

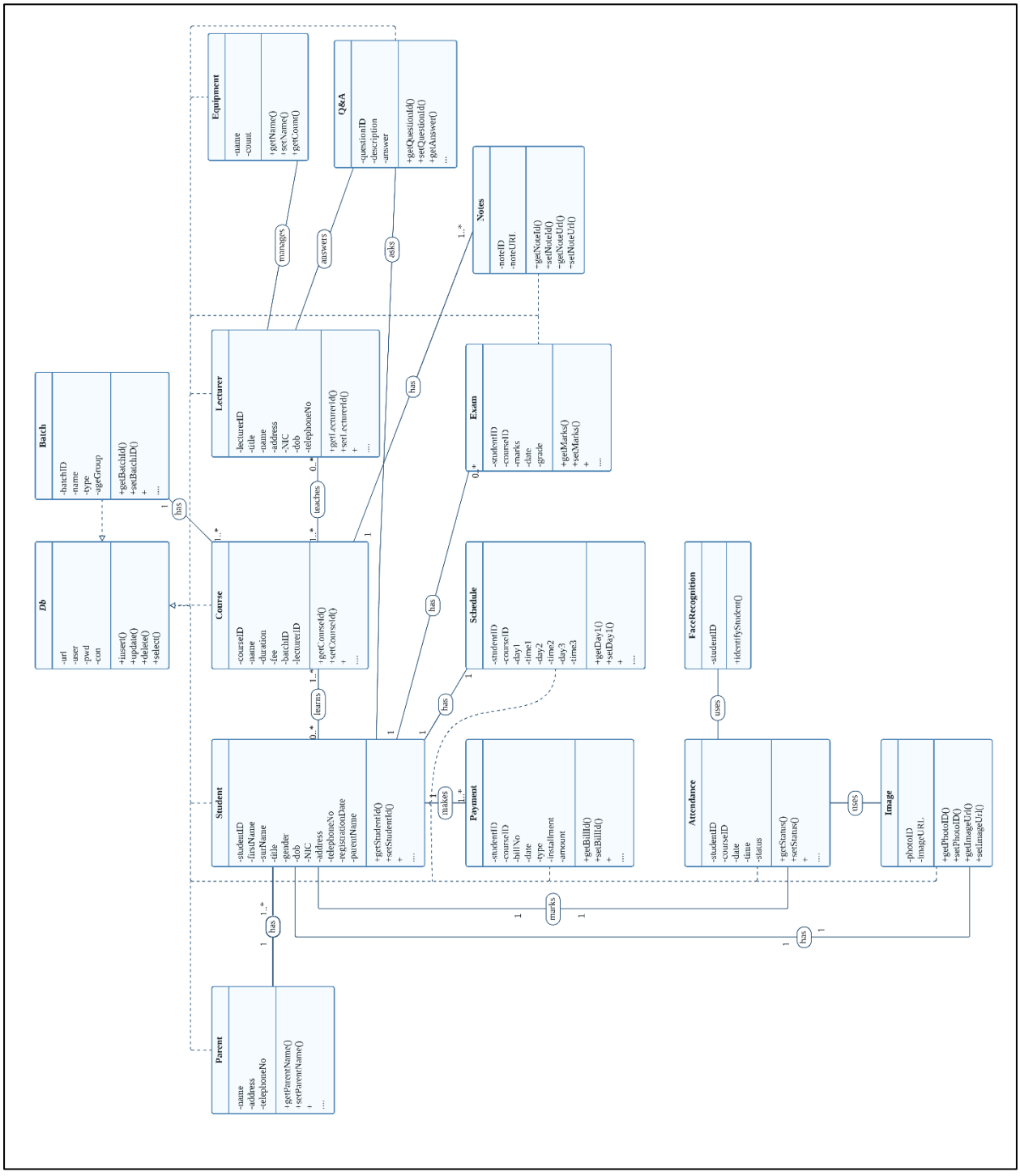

Figure 11: Class Diagram

This diagram shows the classes that is implemented in the system

ER Diagram

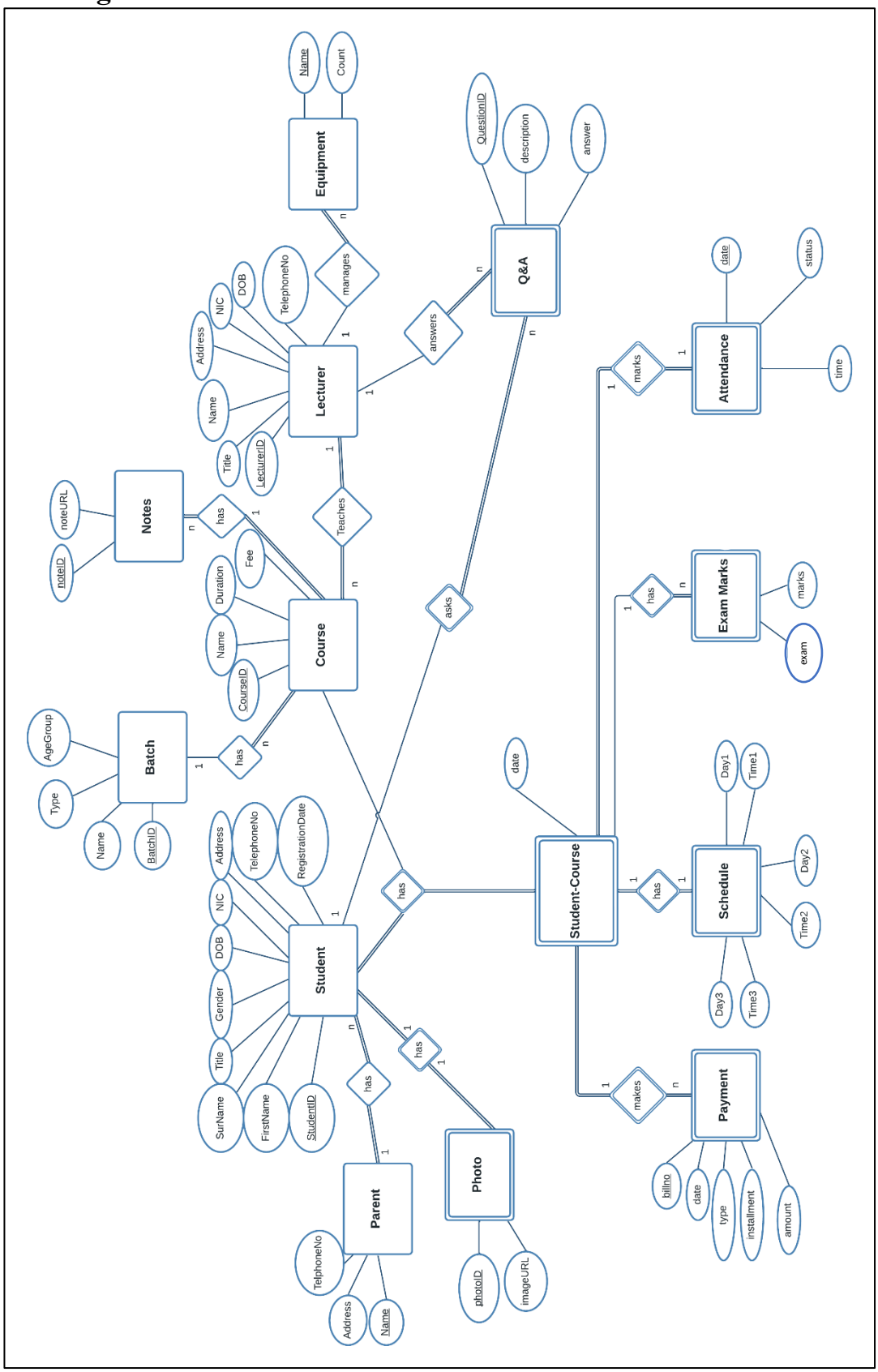

Figure 12: ER Diagram

This diagram shows the entities in the system and their relationships

Sequence Diagram for Manage Course Details

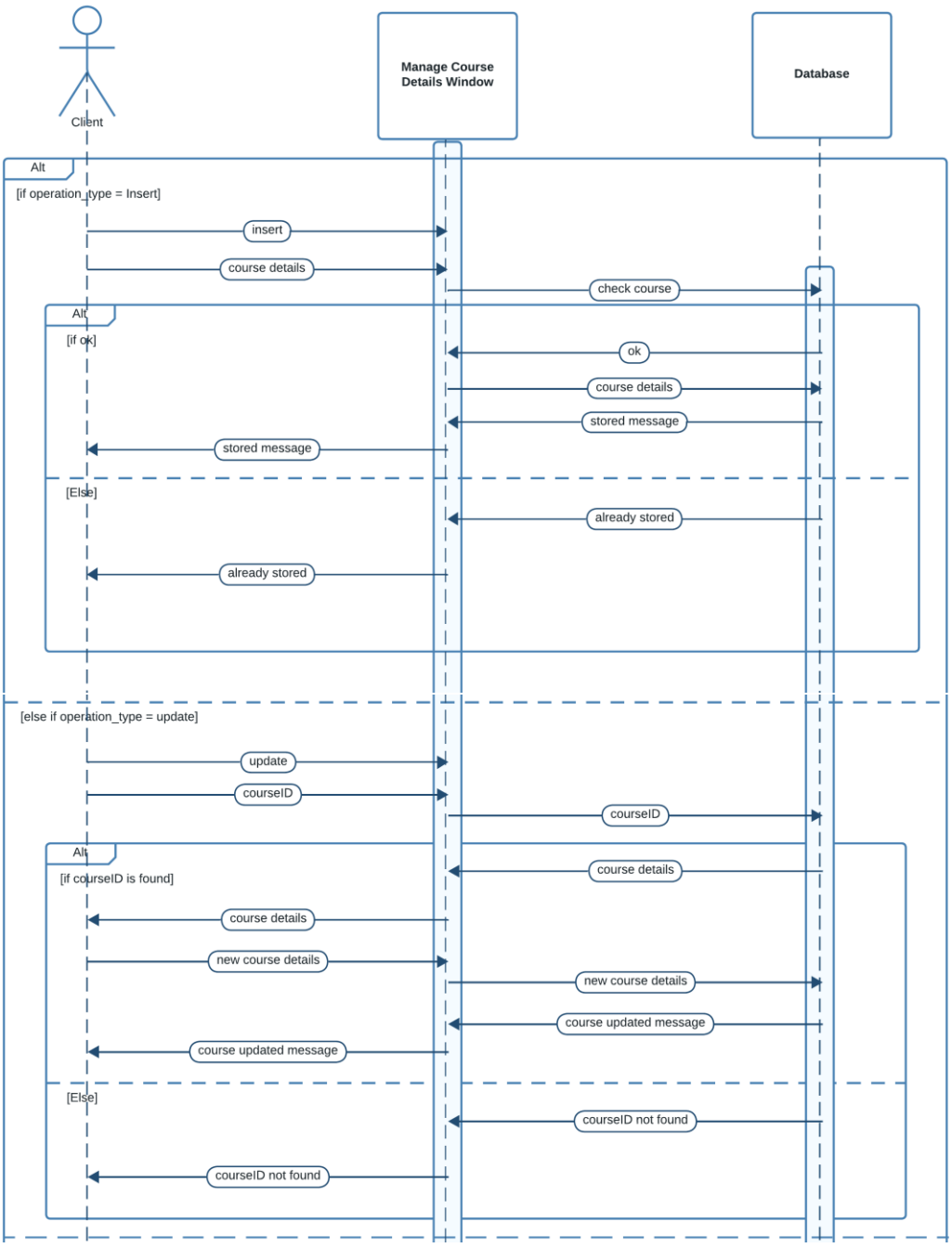

Figure 13: Sequence Diagram (Manage Course Details)

This diagram shows the sequences when managing course details

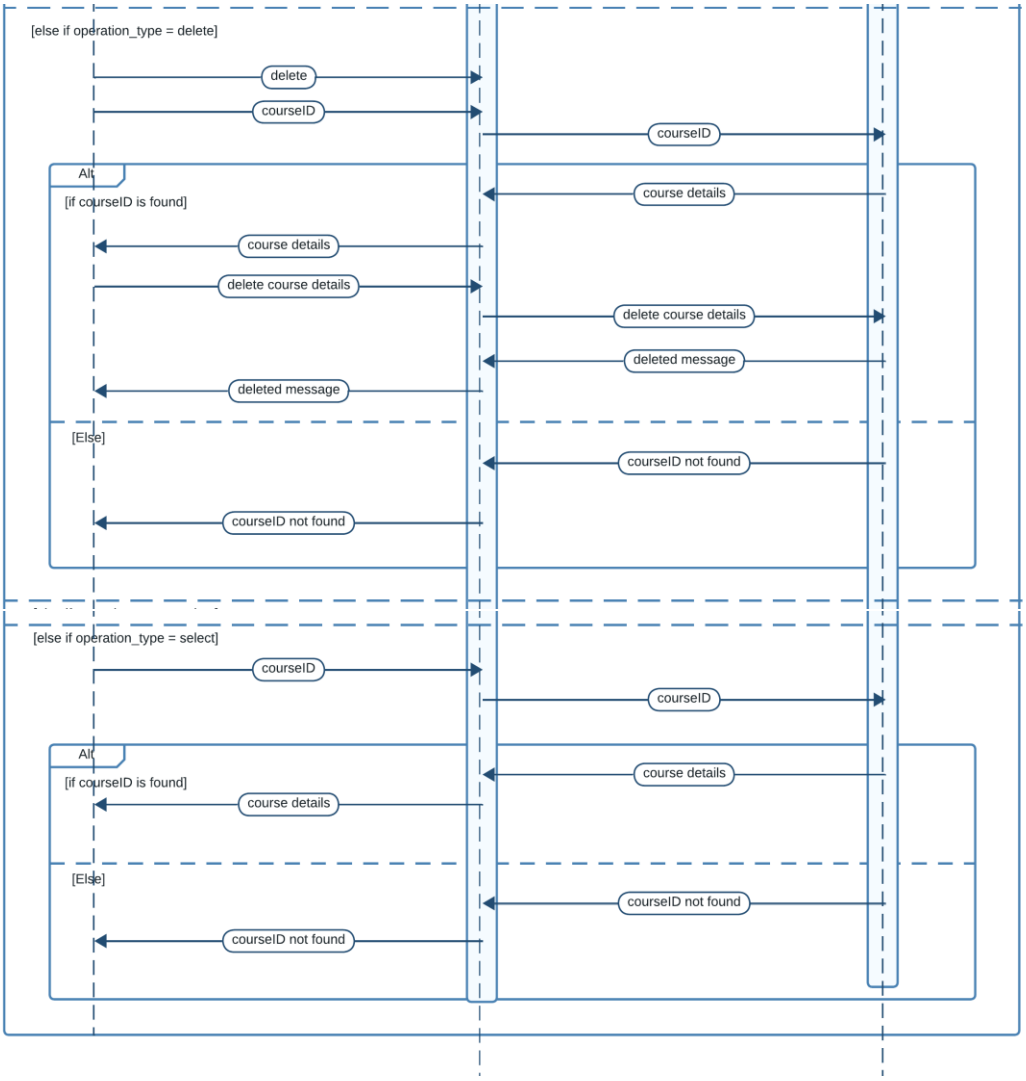

Figure 14: Sequence Diagram (Manage Course Details)

This diagram shows the sequences when managing course details

Sequence Diagram for Manage Student

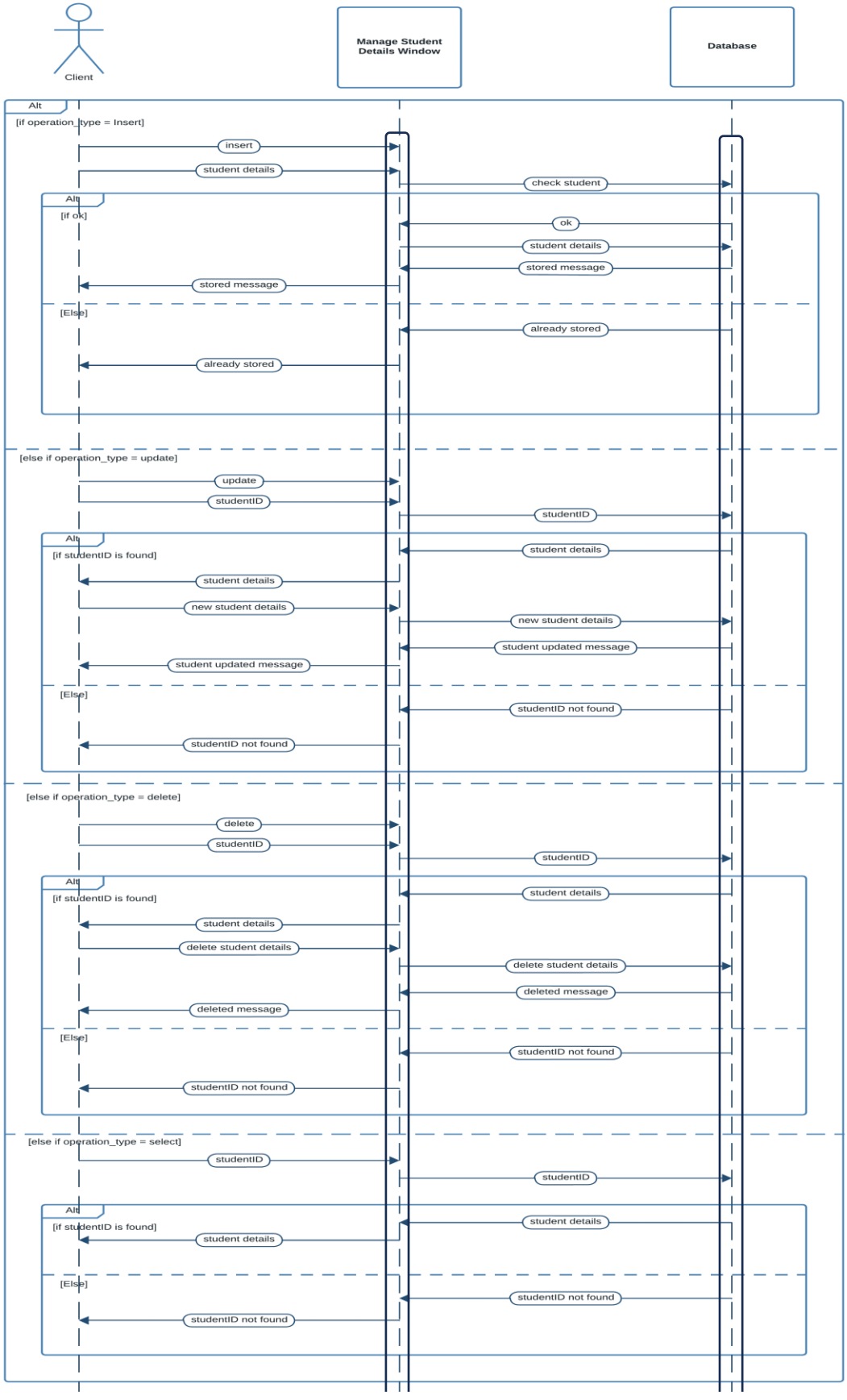

Figure 15: Sequence Diagram (Manage Student Details)

This diagram shows the sequences when managing student details

Sequence Diagram for Manage Lecturer

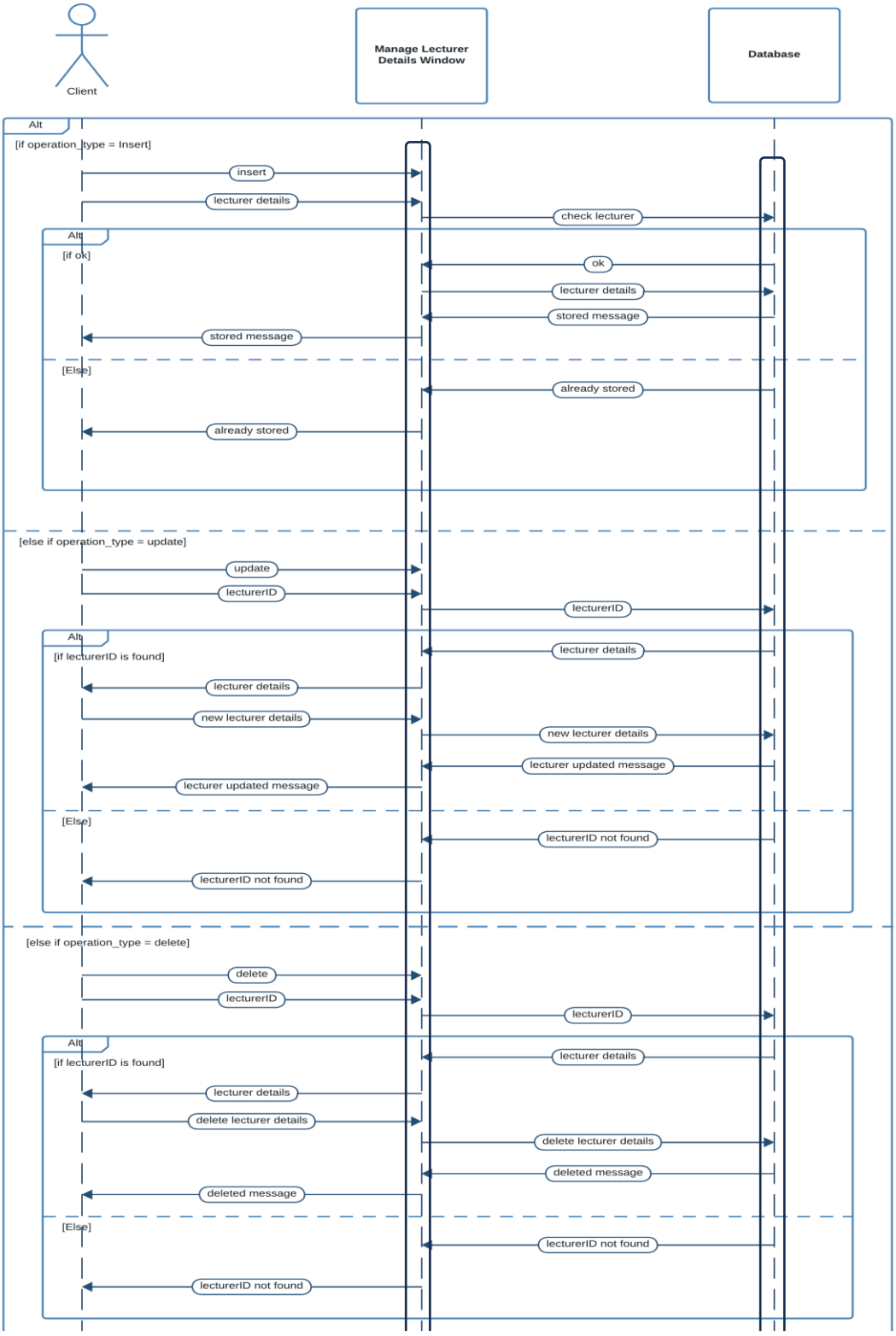

Figure 16: Sequence Diagram (Manage Lecturer Details)

This diagram shows the sequences when managing lecturer details

Sequence Diagram for Manage Notes

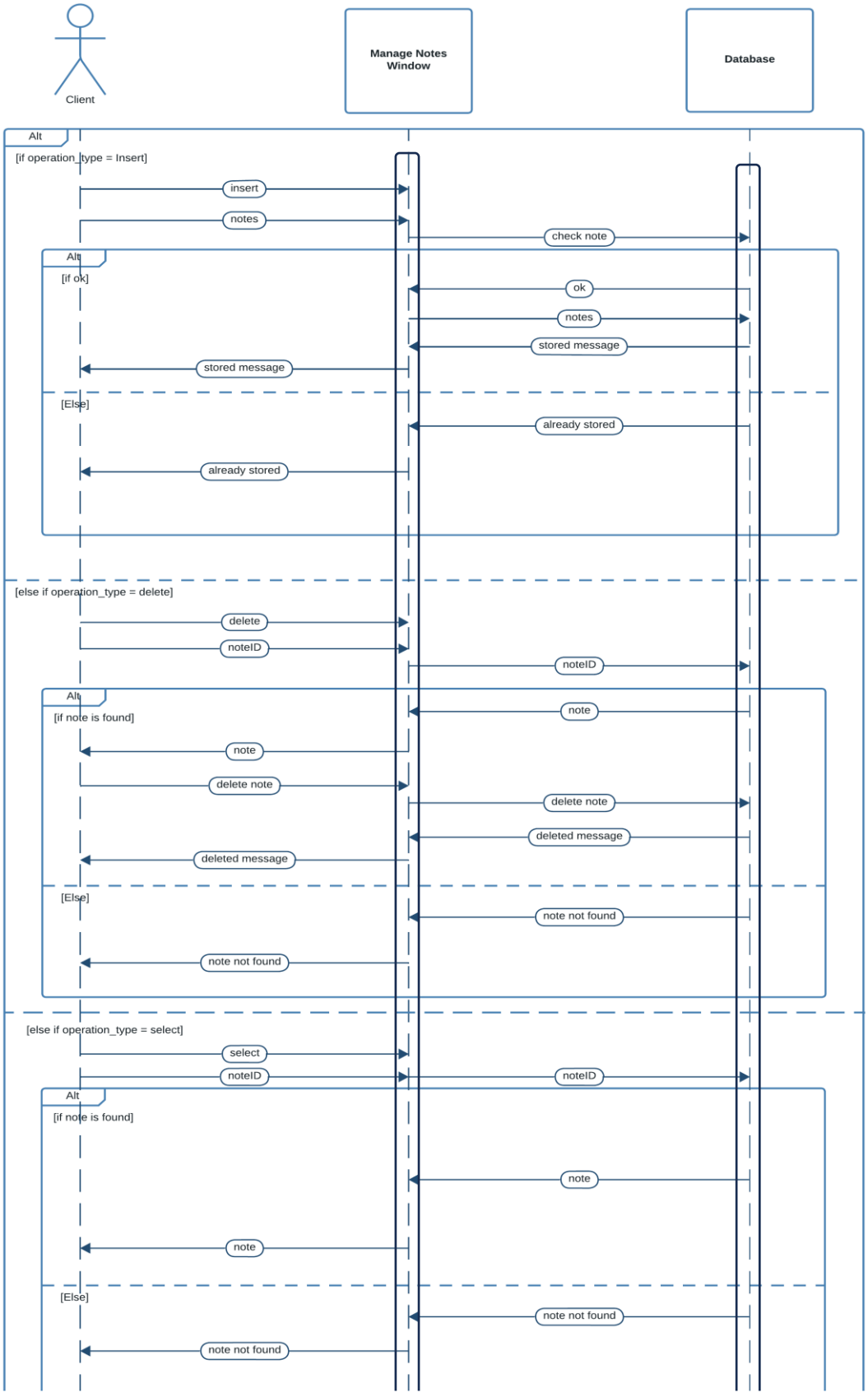

Figure 17: Sequence Diagram (Store Notes)

This diagram shows the sequences when managing notes like storing and retrieving them

Sequence Diagram for Manage Exam Marks

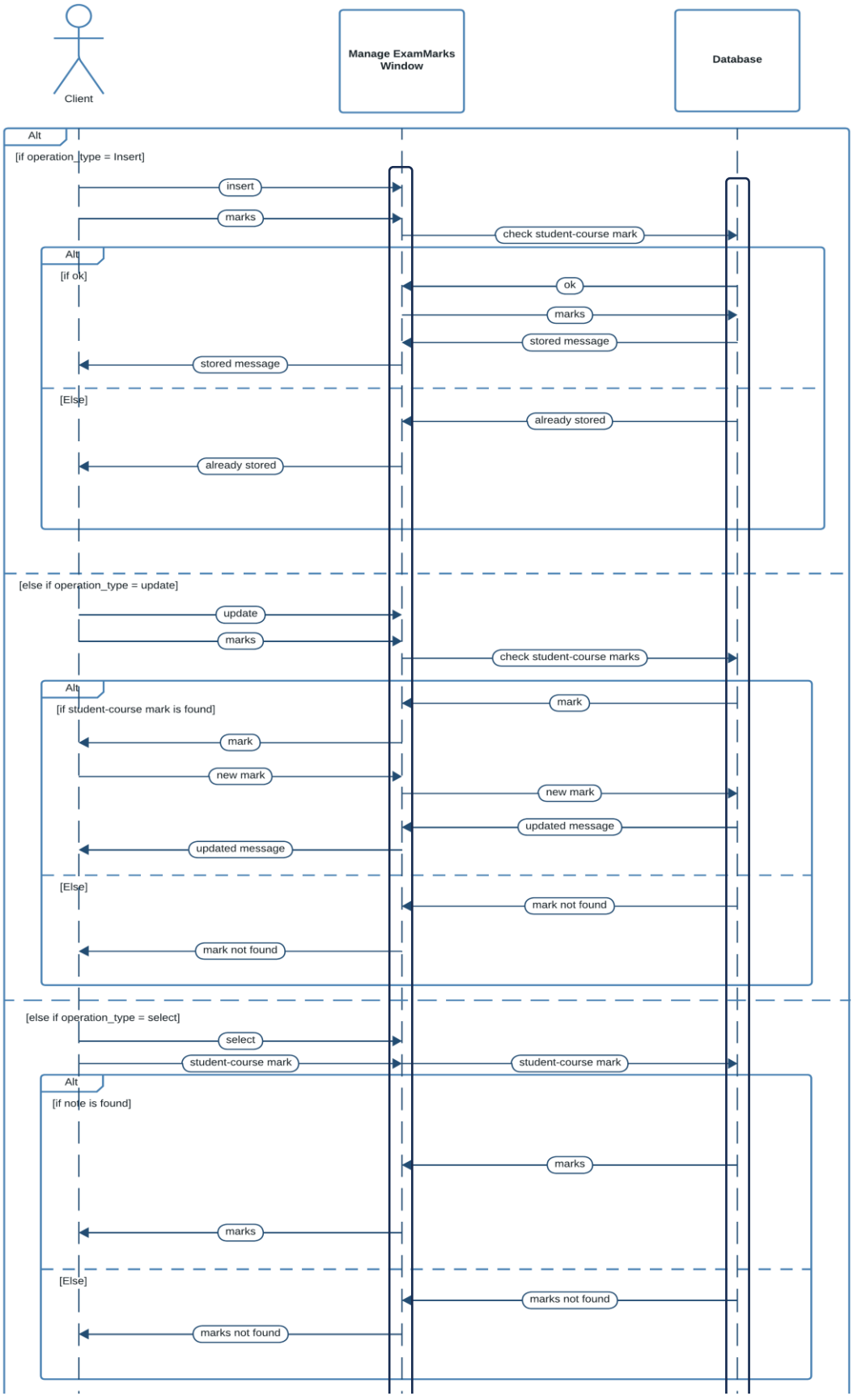

Figure 18: Sequence Diagram (Store Exam Marks)

Sequence Diagram for Manage Schedule

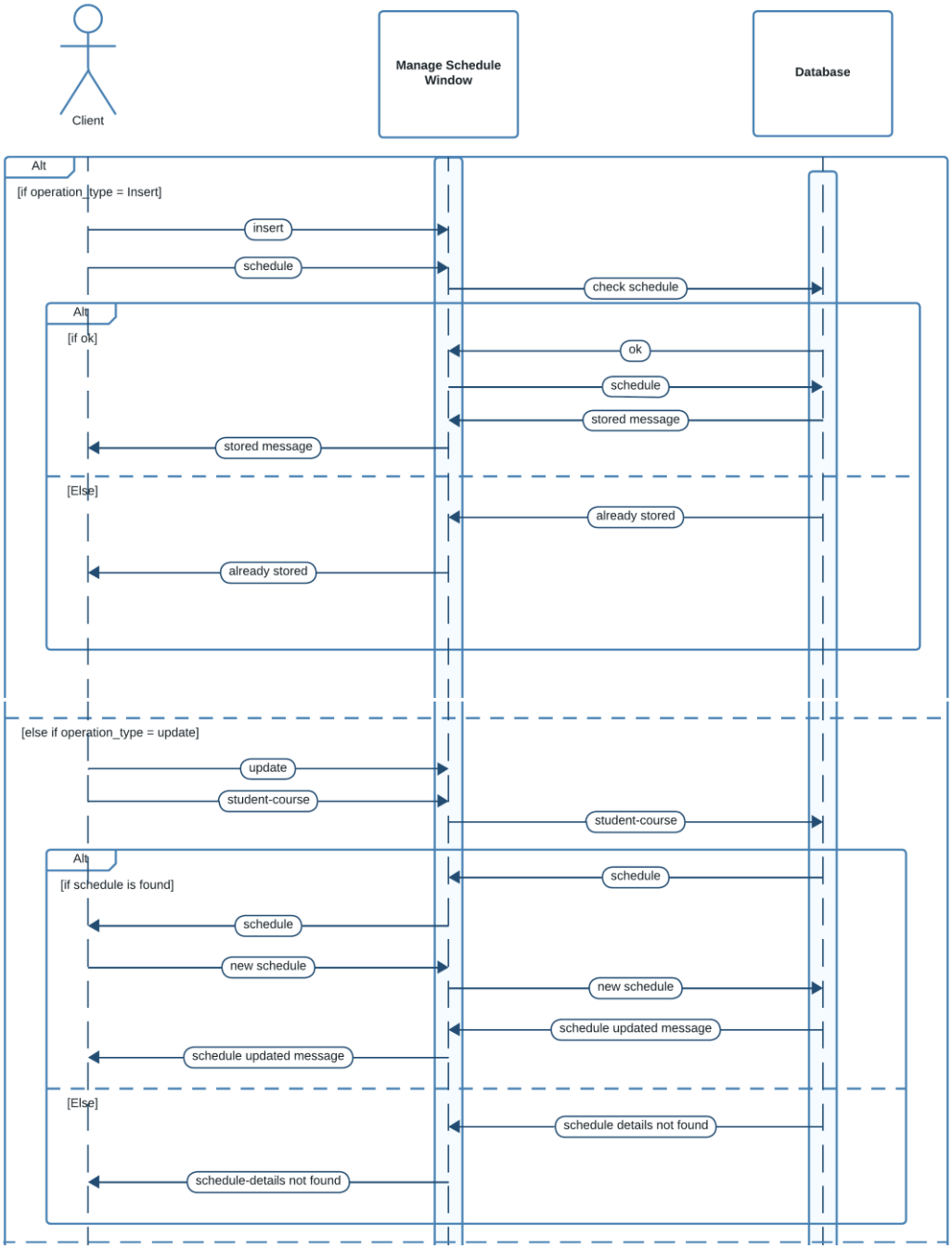

Figure 19: Sequence Diagram for Manage Schedule

This diagram shows the sequences when managing schedule

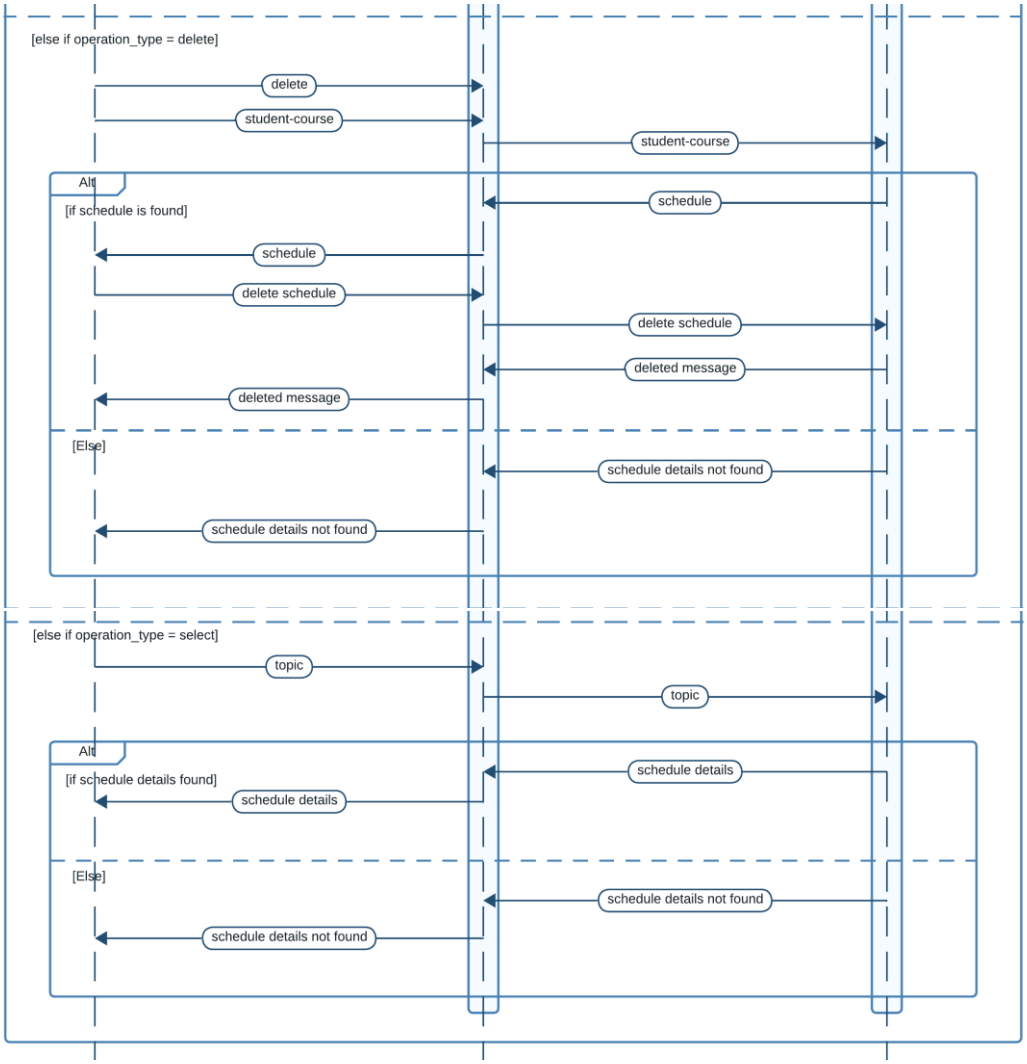

Figure 20: Sequence Diagram for Manage Schedule

This diagram shows the sequences when managing schedule

Sequence Diagram for Manage Payments

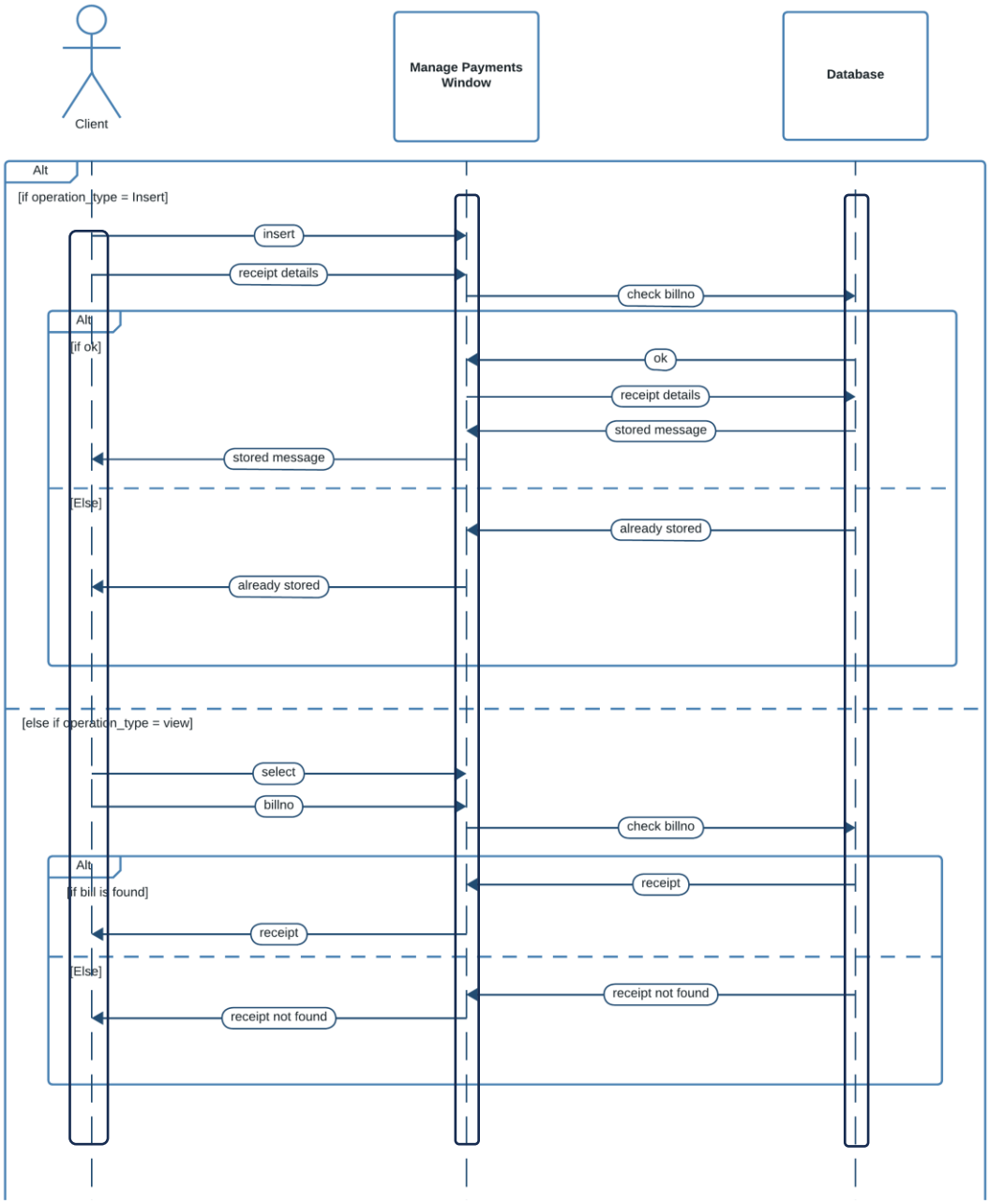

Figure 21: Sequence Diagram (Store payments)

This diagram shows the sequences when storing payments

Sequence Diagram for View Attendance

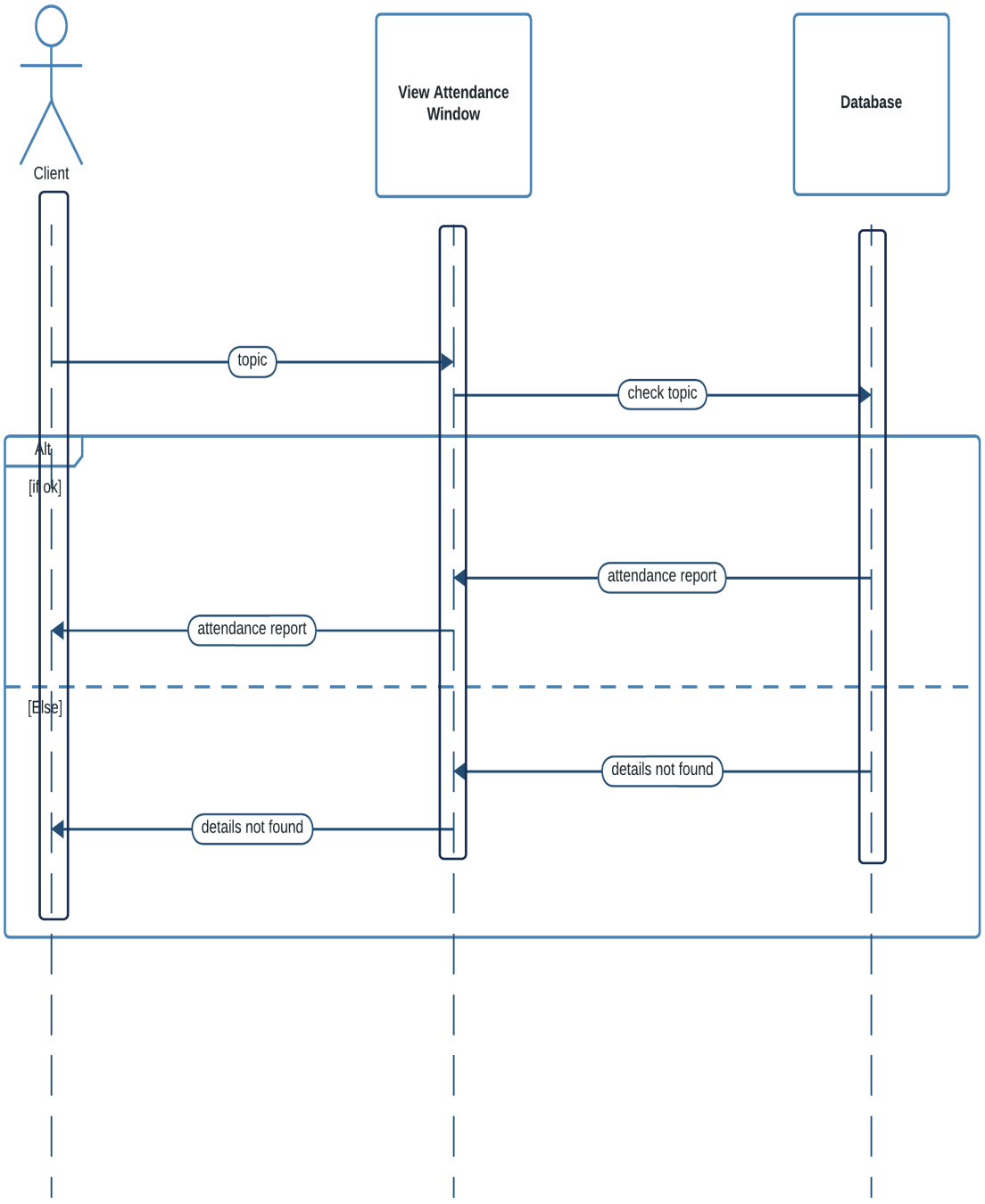

Figure 22: Sequence Diagram (View Attendance)

This diagram shows the sequences when viewing student attendance

Sequence Diagram for View Questions

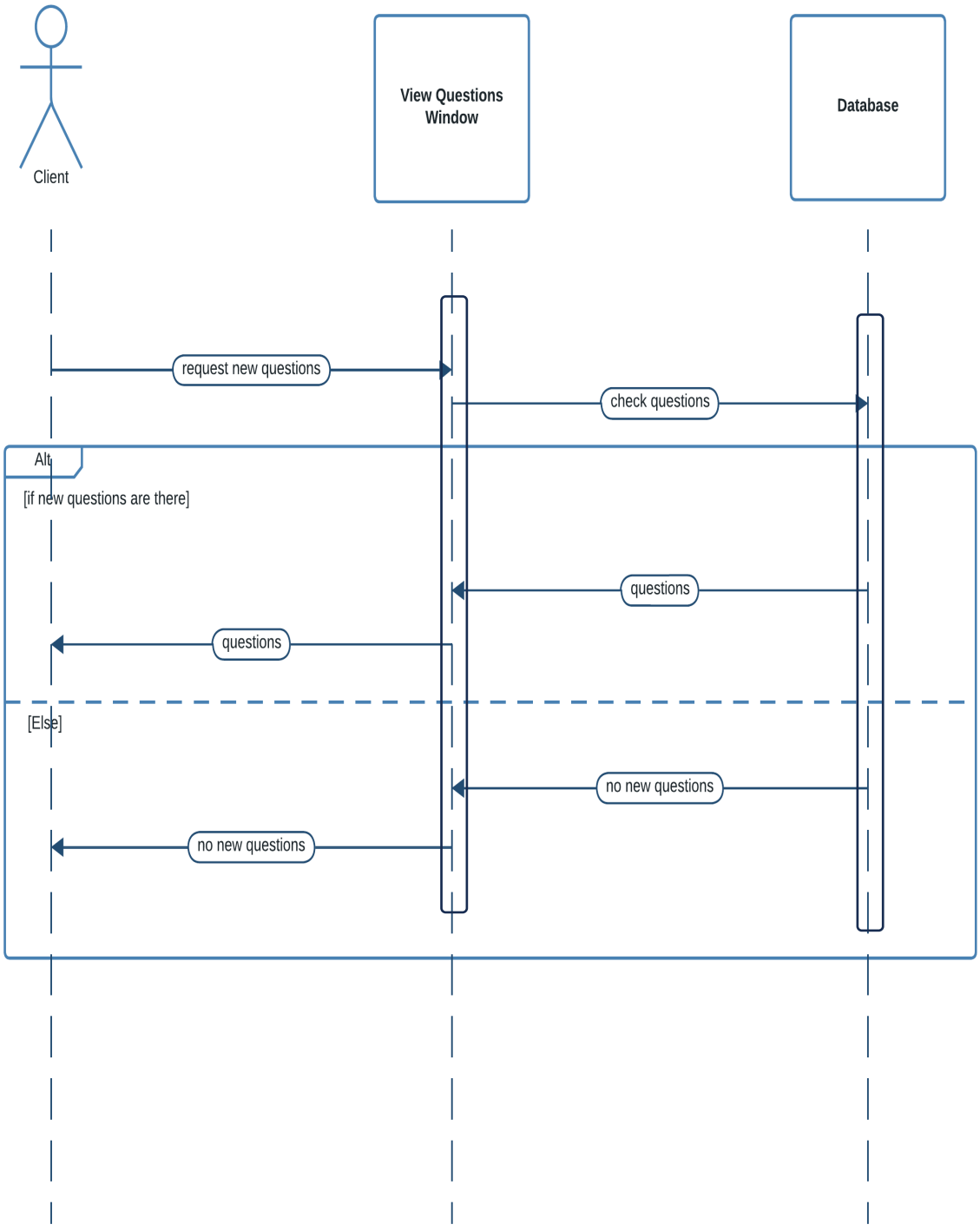

Figure 23: Sequence Diagram (View Questions)

This diagram shows the sequences when viewing questions

Sequence Diagram for Answer Questions

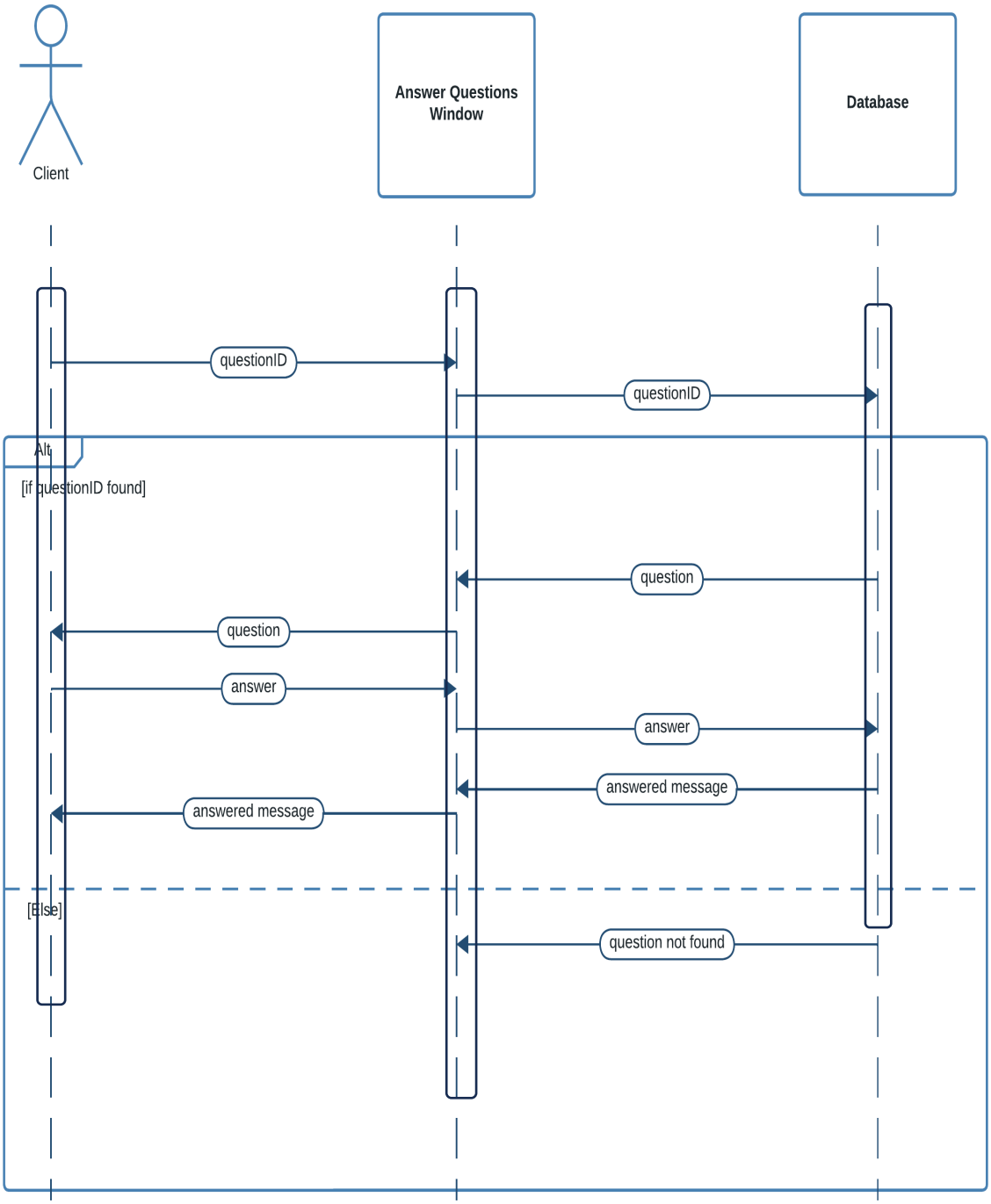

Figure 24: Sequence Diagram (Answer Questions)

This diagram shows the sequences when answering questions

Sequence Diagram for View Results

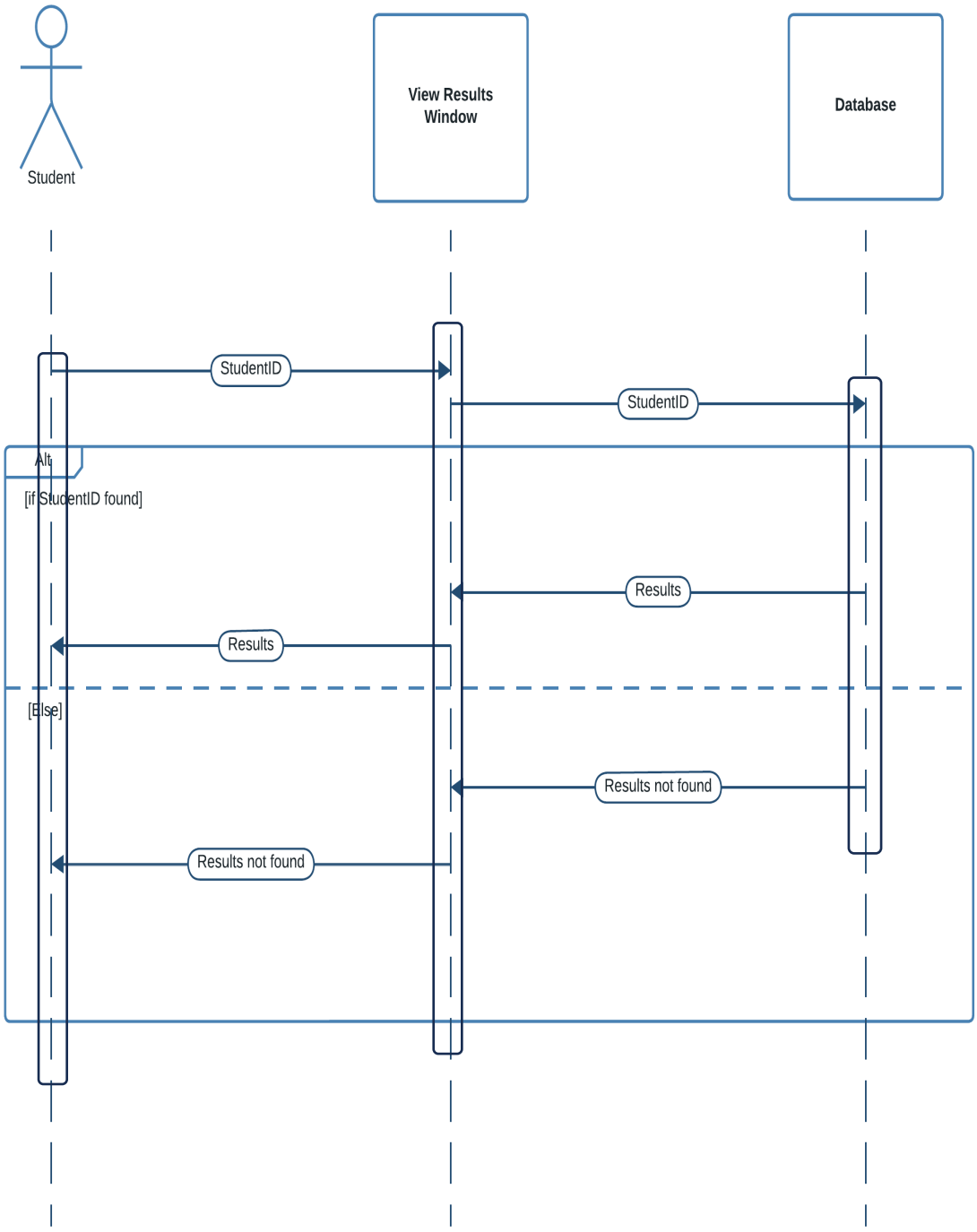

Figure 25: Sequence Diagram for View Results

This diagram shows the sequences when viewing exam results

Sequence Diagram for View Schedule

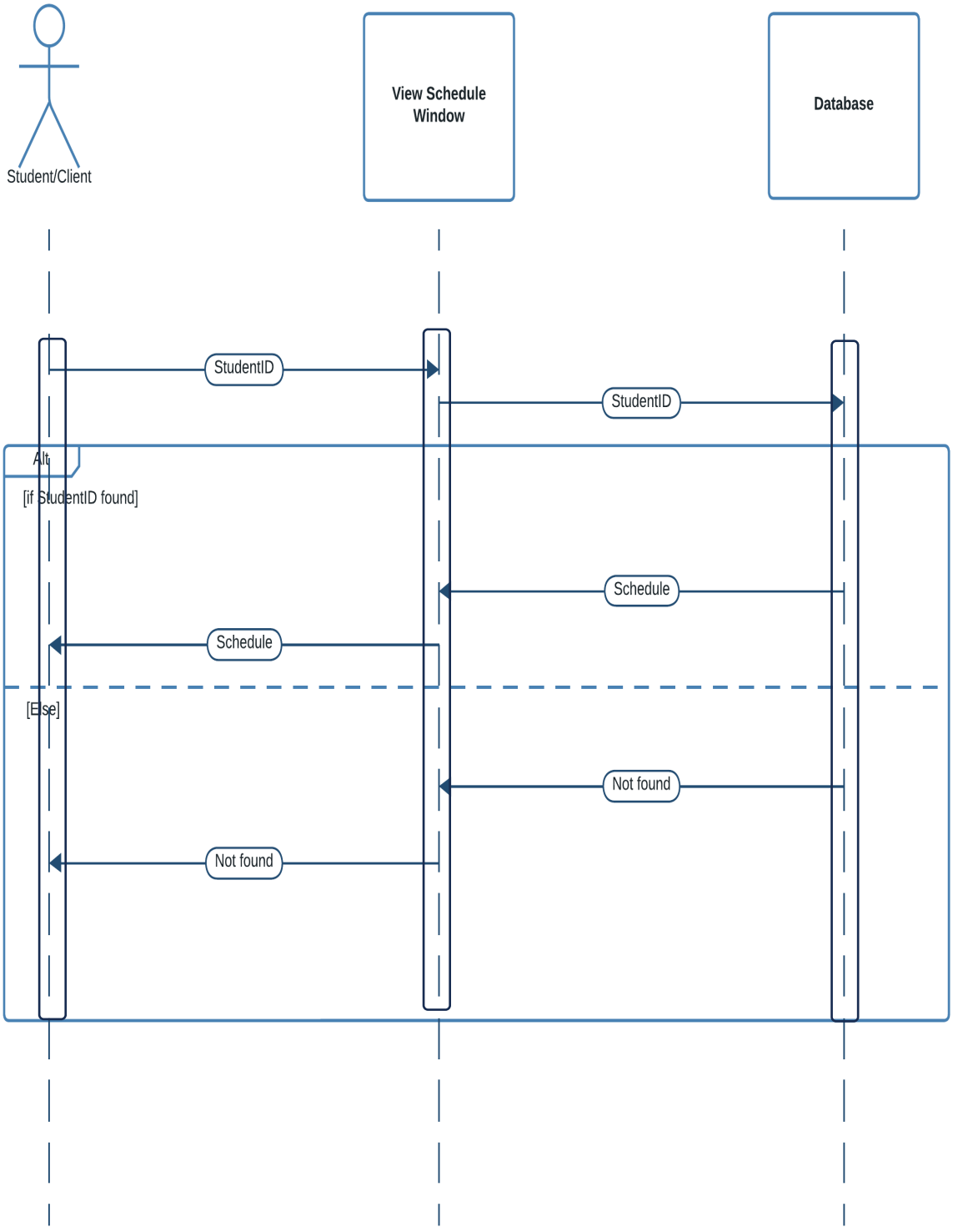

Figure 26: Sequence Diagram for View Schedule

This diagram shows the sequences when viewing schedule

Sequence Diagram for Ask Questions

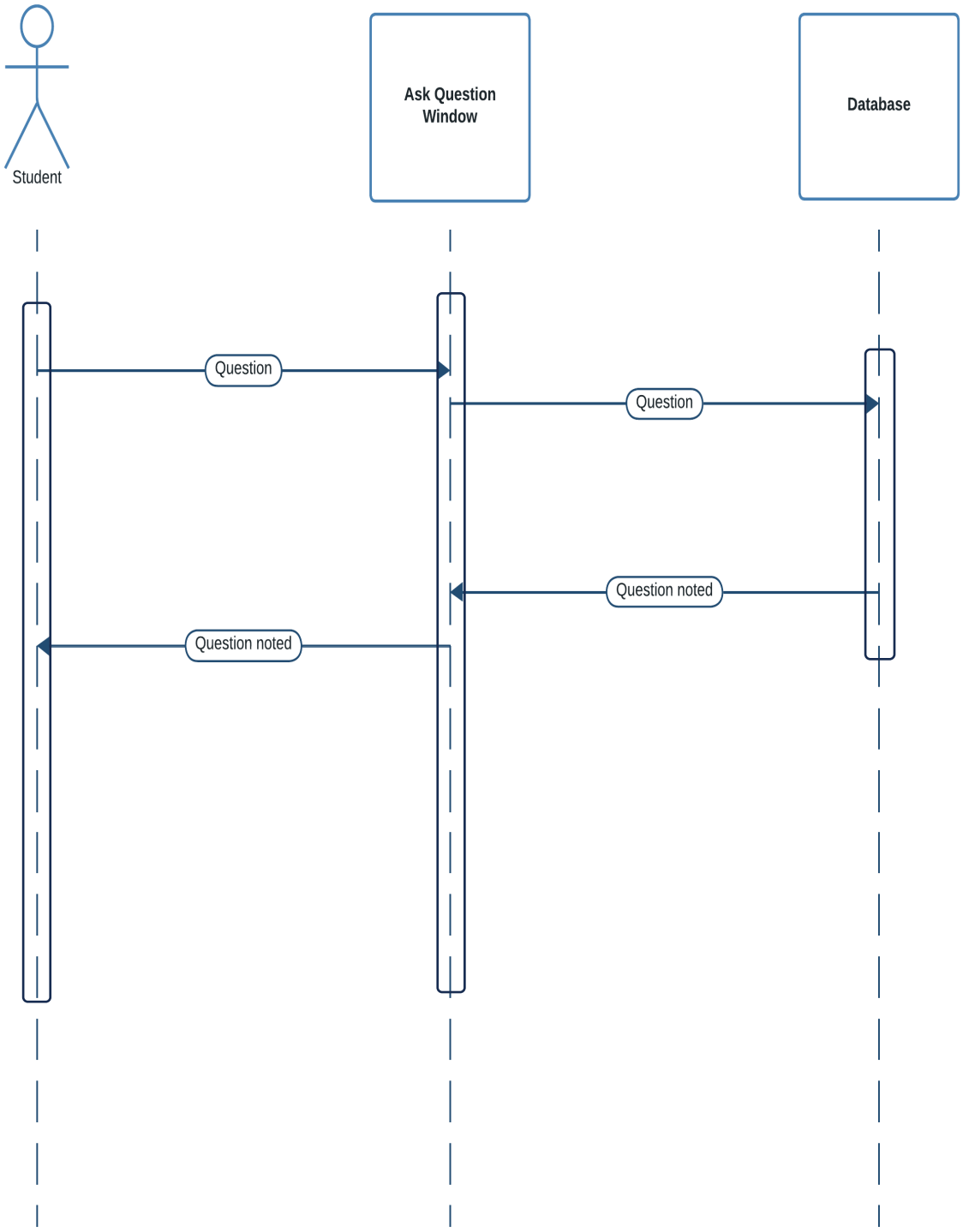

Figure 27: Sequence Diagram for Ask Questions

This diagram shows the sequences when students are asking questions

Sequence Diagram for View Answers

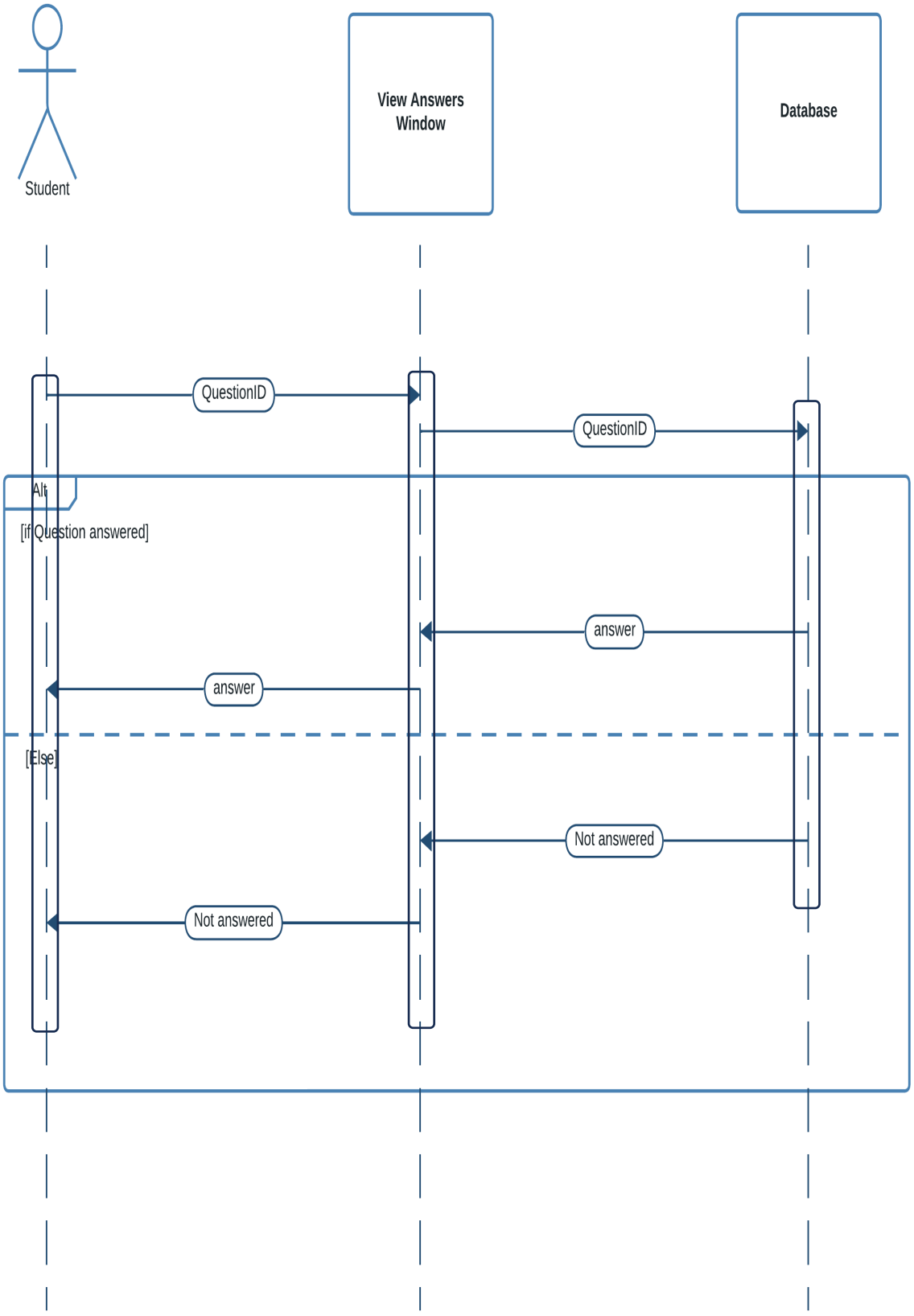

Figure 28: Sequence Diagram for View Results

Sequence Diagram for View Lecture Notes

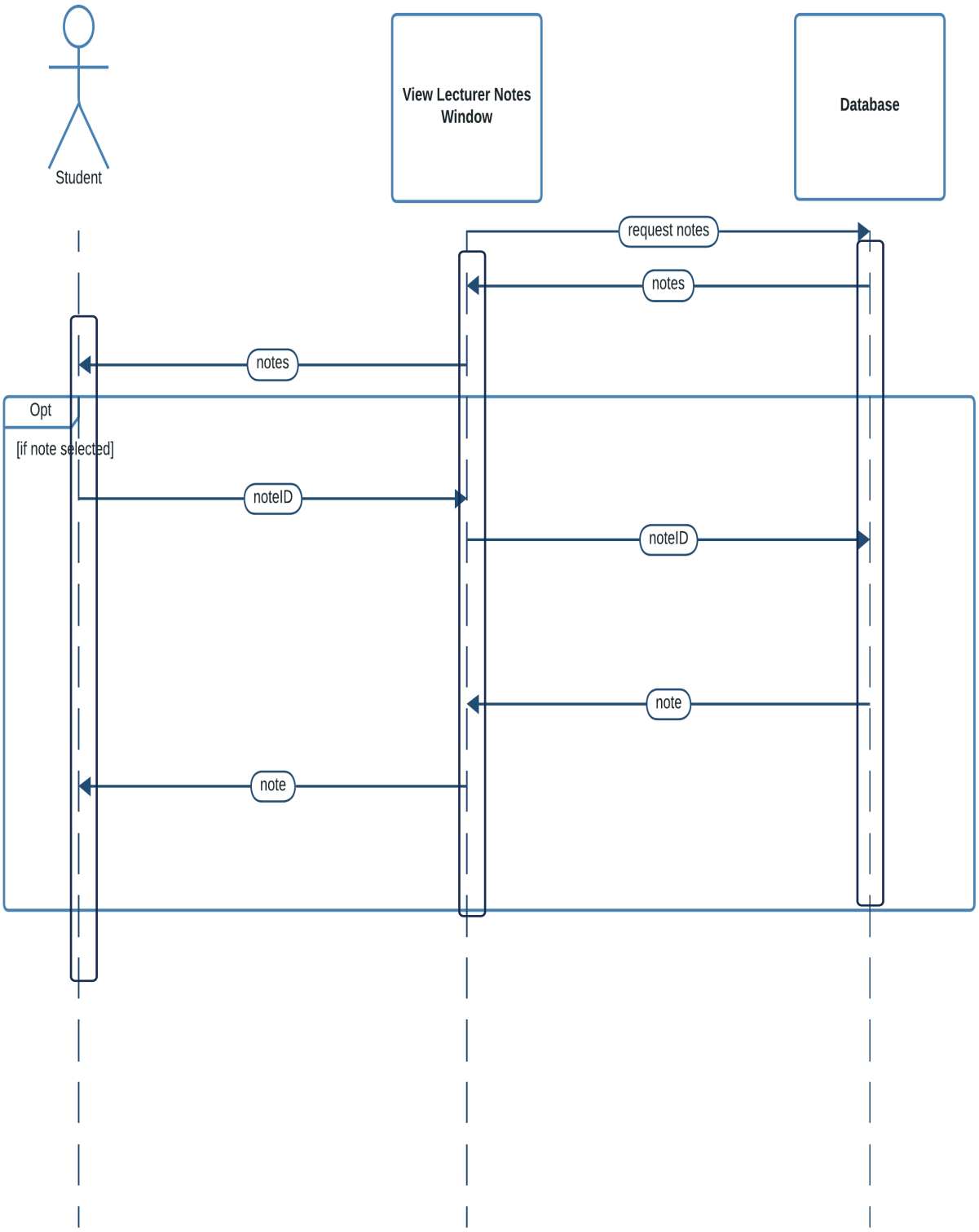

Figure 29: Sequence Diagram for View Lecture Notes

3.3 Chapter Summary

In this chapter we looked at the designs of the system which helped us to create the application. The diagrams made it easy to implement the program because we knew what to implement and how to implement it exactly.

Chapter 4: Solution Design

4.1 Introduction

In this section we will be looking at the Interface Designs, and Database Schema Design of the System

4.2 Interface Design

Interface number	Interface name
1	Login Screen
2	Course Details
3	Student Details
4	Lecturer Details
5	Payment Details
6	Set Schedule
7	Store Lecture Note
8	Store Exam Marks
9	Ask Questions
10	View Results
11	View Notes
12	View Questions
13	View Attendance
14	View Schedule
15	Home Screen
16	Parent Details

Table 2: Table of Interfaces

Interface No: 01

Interface Name: Login Screen

Description: Used to login to the system

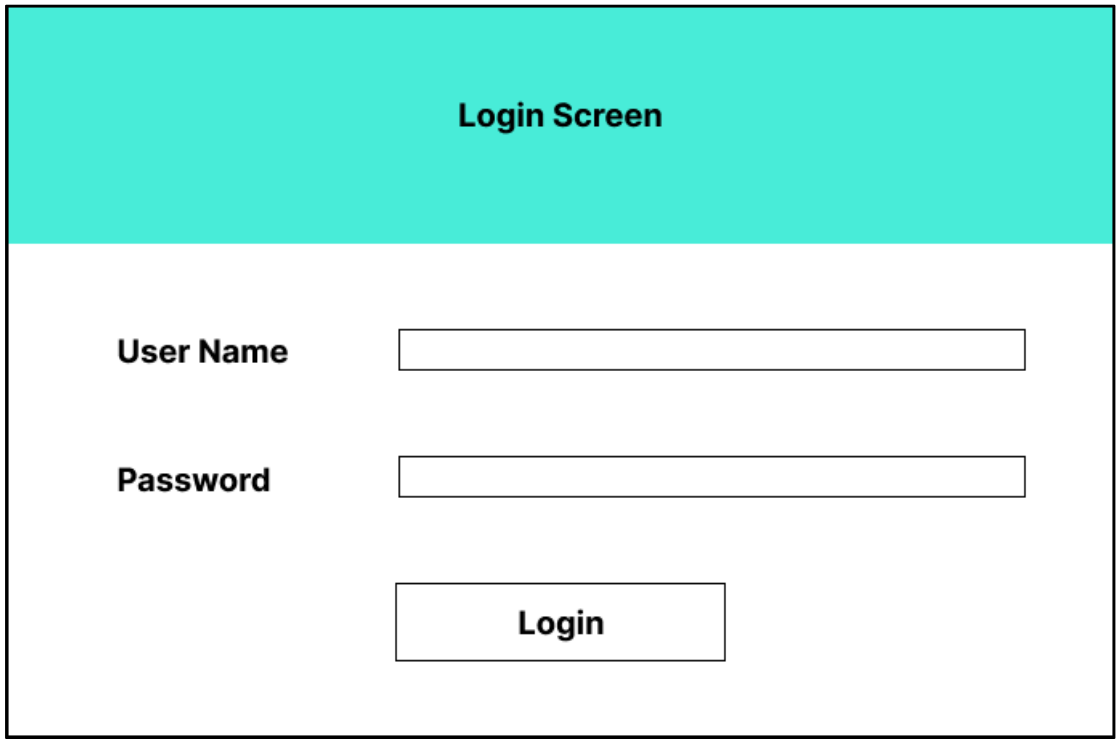

Figure 30: Login Page

Pseudo Code

```
START
        //VALIDATIONS
        INPUT USERNAME
        INPUT PASSWORD

        IF USERNAME == “DHAMMIKA” && PASSWORD == “1234”
        THEN
                OUTPUT LOGIN SUCCESSFUL
                SHOW HOME PAGE
                ELSE
                OUTPUT USERNAME OR PASSWORD IS INVALID
        END IF
END
```

Figure 31: Pseudo code for login

Interface No: 02

Interface Name: Course Details

Description: Used to handle all the Course Details related operations

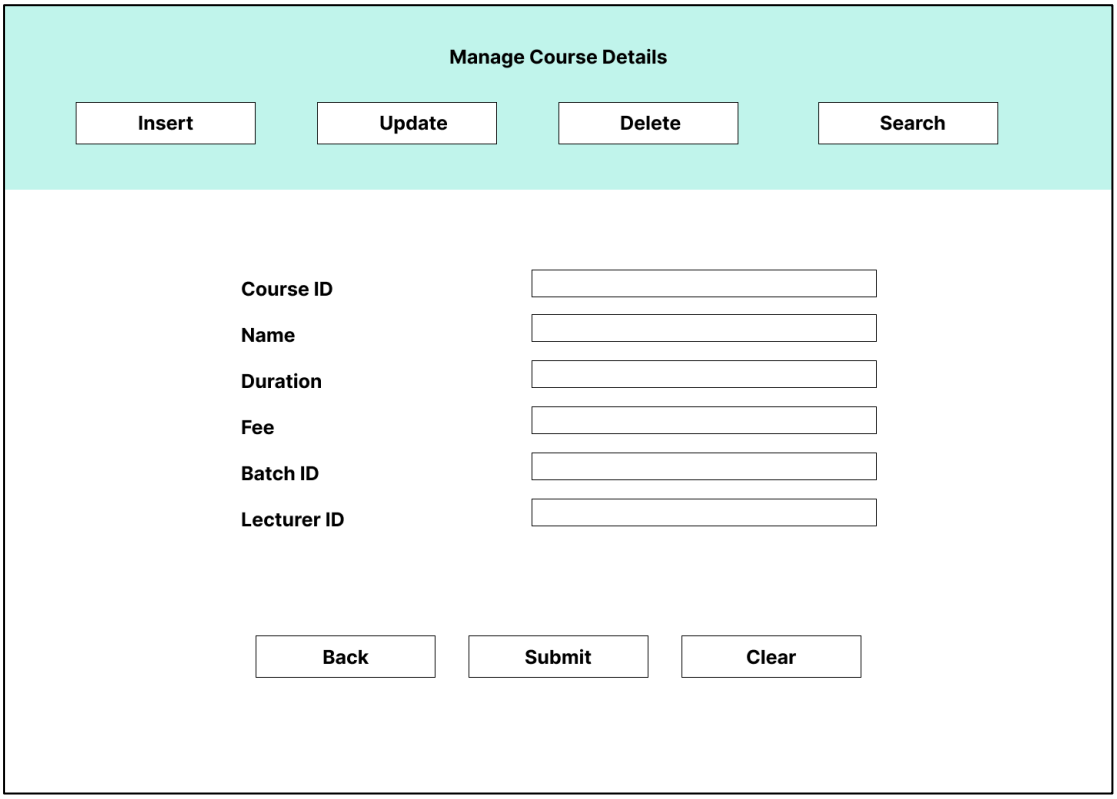

Figure 32: Manage Course Details Screen

Pseudo Code

```
START
        //VALIDATIONS


        OPEN DBCON
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO COURSE (“COURSEID”,…))
        ELSE
        IF UPDATE_CLICKED
         THEN
                executeQuery(UPDATE COURSE SET FEE = “2000.00” WHERE COURSEID =”C0001”)
        ELSE
        IF DELETE_CLICKED
         THEN
                executeQuery(DELETE FROM COURSE WHERE COURSEID=”C0001”)
        ELSE
        IF SELECT_CLICKED
         THEN
                Result = executeQuery(SELECT * FROM COURSE)
                OUTPUT Result
        END IF

        CLOSE DBCON
END
```

Figure 33: Pseudo code for course management

Interface No: 03

Interface Name: Student Details

Description: Used to handle all the Student Details related operations

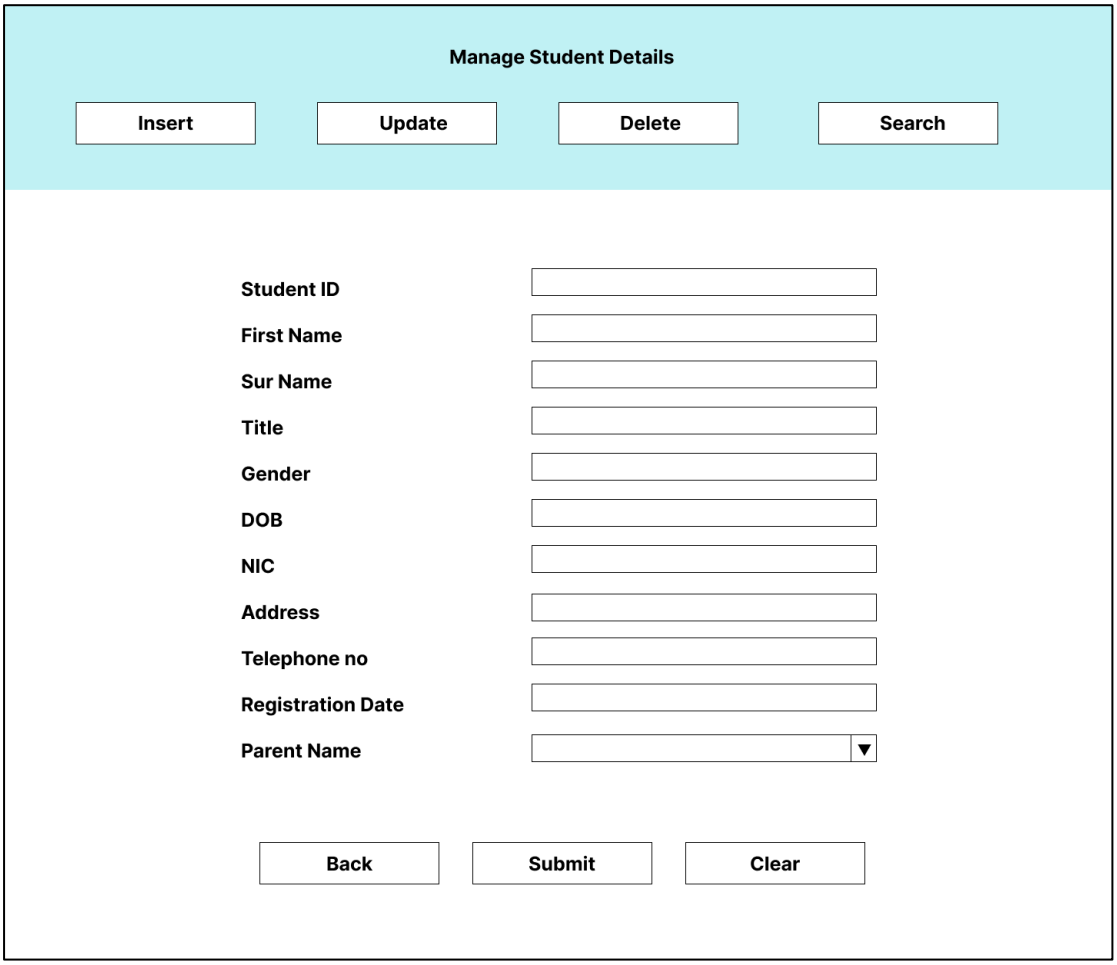

Figure 34: Manage Student Details Screen

Pseudo Code

```
START
        //VALIDATIONS
        INCLUDE DbCon
        OPEN DBCON
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO STUDENTS (“STUDENTID”,…))
        ELSE
        IF UPDATE_CLICKED
         THEN
                executeQuery(UPDATE STUDENTS SET ADDRESS=”” WHERE STUDENTID = “S0001”)
        ELSE
        IF DELETE_CLICKED
         THEN
                executeQuery(DELETE FROM STUDENTS WHERE STUDENTID = “S0001”)
        ELSE
        IF SELECT_CLICKED
         THEN
                Result = executeQuery(SELECT * FROM STUDENTS)
                OUTPUT Result
        END IF
        CLOSE DBCON
END
```

Figure 35: Pseudo code for student management

Interface No: 04

Interface Name: Lecturer Details

Description: Used to handle all the Lecturer Details related operations

Figure 36: Manage Lecturer Details Screen

Pseudo Code

```
START
        INCLUDE DbCon
        //VALIDATIONS
        OPEN DBCON
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO LECTURERS (“LECTURERID”, …))
        ELSE
        IF UPDATE_CLICKED
         THEN
                executeQuery(UPDATE LECTURER SET ADDRESS=”” WHERE LECTURERID = “L0001”)
        ELSE
        IF DELETE_CLICKED
         THEN
                executeQuery(DELETE FROM LECTURER WHERE LECTURERID = “L0001”)
        ELSE
        IF SELECT_CLICKED
         THEN
                Result = executeQuery(SELECT * FROM LECTURER)
                OUTPUT Result
        END IF
        CLOSE DBCON
END
```

Figure 37: Pseudo code for lecturer management

Interface No: 05

Interface Name: Payment Details

Description: Used to handle all the Payment Details related operations

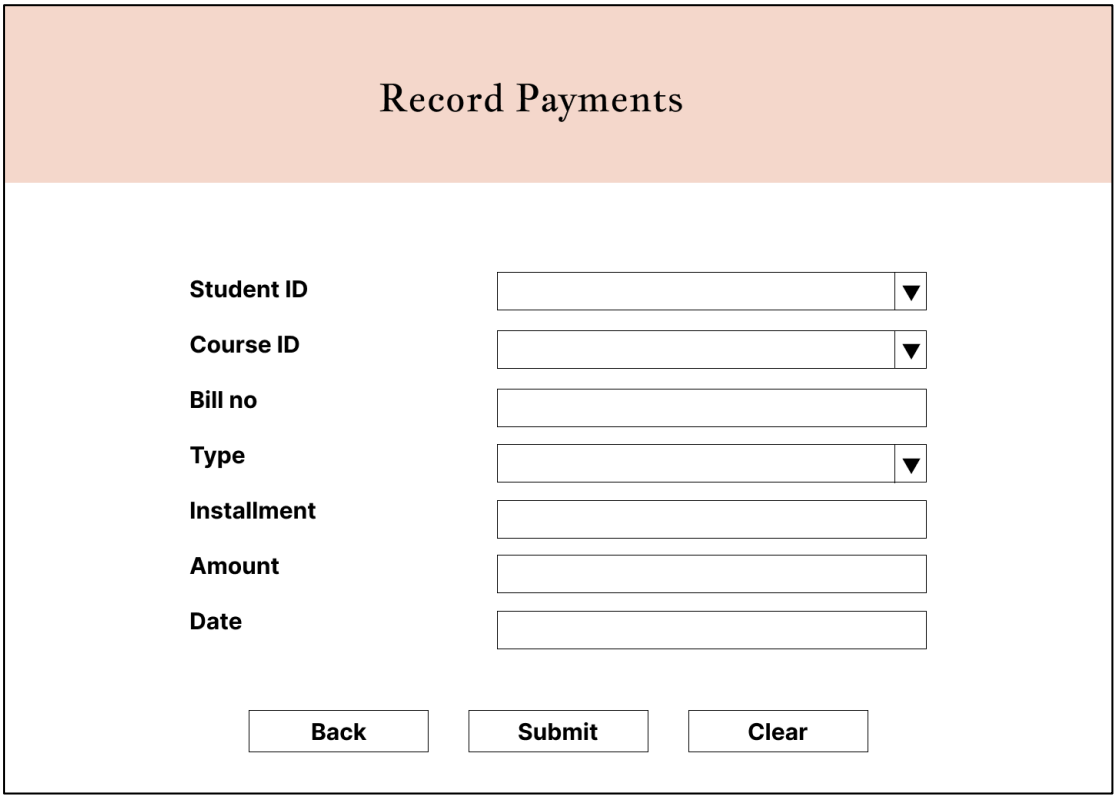

Figure 38: Record Payment Screen

Pseudo Code

```
START
        INCLUDE DbCon
        //VALIDATIONS
        OPEN DBCON
        IF INSERT_CLICKED
         THEN
        executeQuery(INSERT INTO PAYMENTS (“STUDENTID”, “COURSEID”, “TYPE”, “AMOUNT”, “INSTALLMENT”, ‘BILLNO’);
        END IF
        CLOSE DBCON
END
```

Figure 39: Pseudo code for manage payments

Interface No: 06

Interface Name: Set Schedule

Description: Used to set schedule of the institute

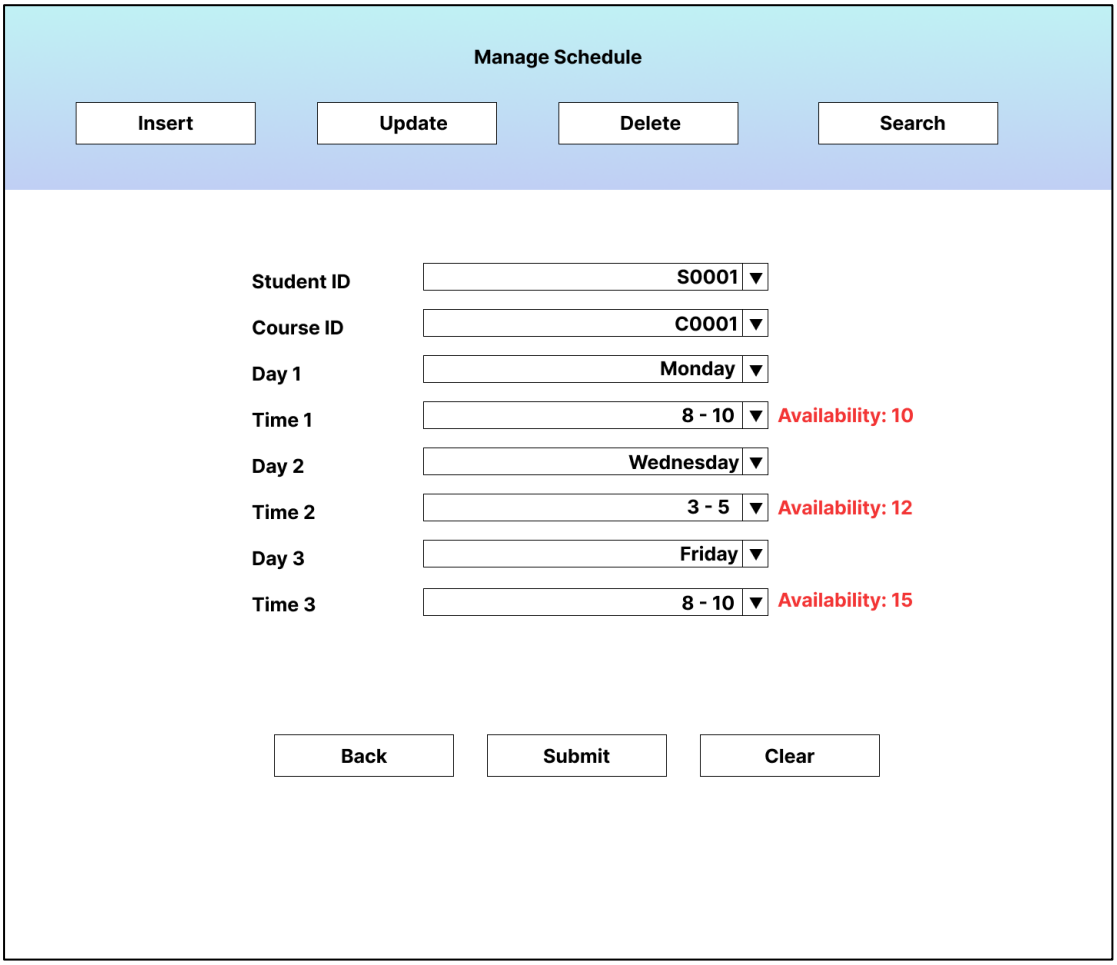

Figure 40: Manage Schedule Screen

Pseudo Code

```
START
        INCLUDE DbCon
        //VALIDATIONS
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO SCHEDULE (“STUDENTID”, ..))
        ELSE
        IF UPDATE_CLICKED
         THEN
                executeQuery(UPDATE SCHEDULE SET TIME=”10 - 12” WHERE BILLNO = “S0001”)
        ELSE
        IF DELETE_CLICKED
         THEN
                executeQuery(DELETE FROM SCHEDULE WHERE STUDENTID = “S0001”)
        ELSE
        IF SELECT_CLICKED
         THEN
                Result = executeQuery(SELECT * FROM SCHEDULE)
                OUTPUT Result
        END IF
END
```

Figure 41: Pseudo code for schedule management

Interface No: 07

Interface Name: Store Lecture Note

Description: Used to store lecture notes to the system

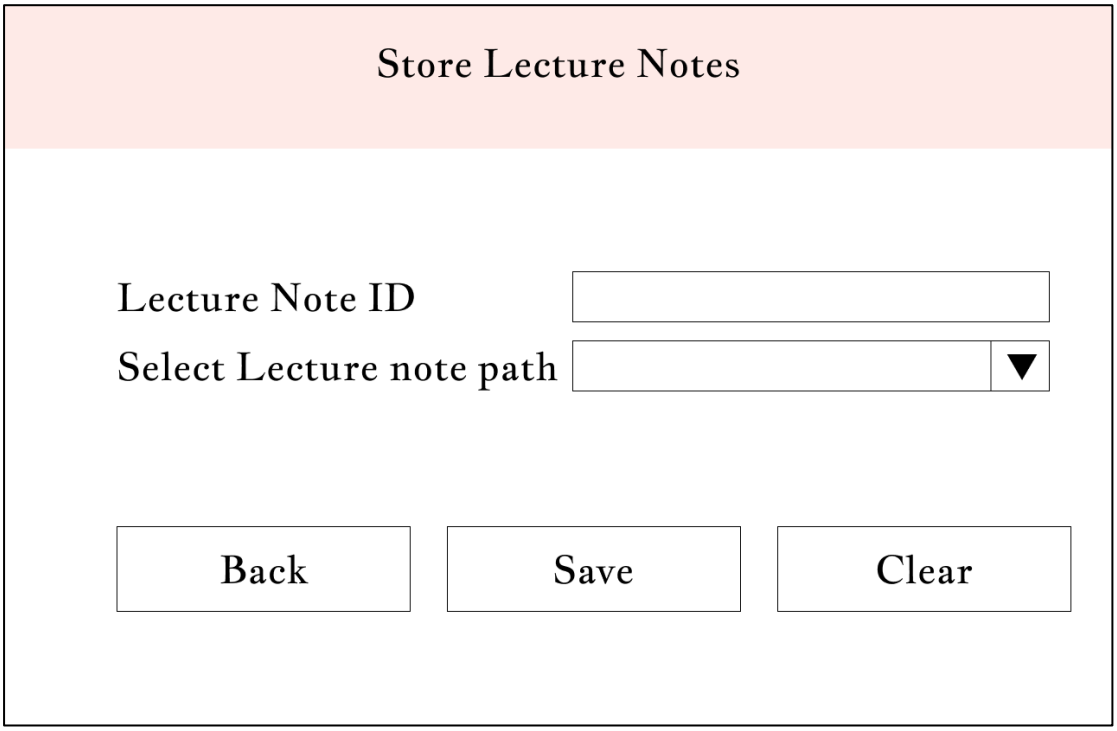

Figure 42: Store Lecture Notes Screen

Pseudo Code

```
START
        INCLUDE DbCon
        //VALIDATIONS
        OPEN DBCON
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO NOTES (“NOTEID”, “NOTEPATH”))
        END IF
        CLOSE DBCON
END
```

Figure 43: Pseudo code for store lecture notes

Interface No: 08

Interface Name: Store Exam Marks

Description: Used to store Exam Marks

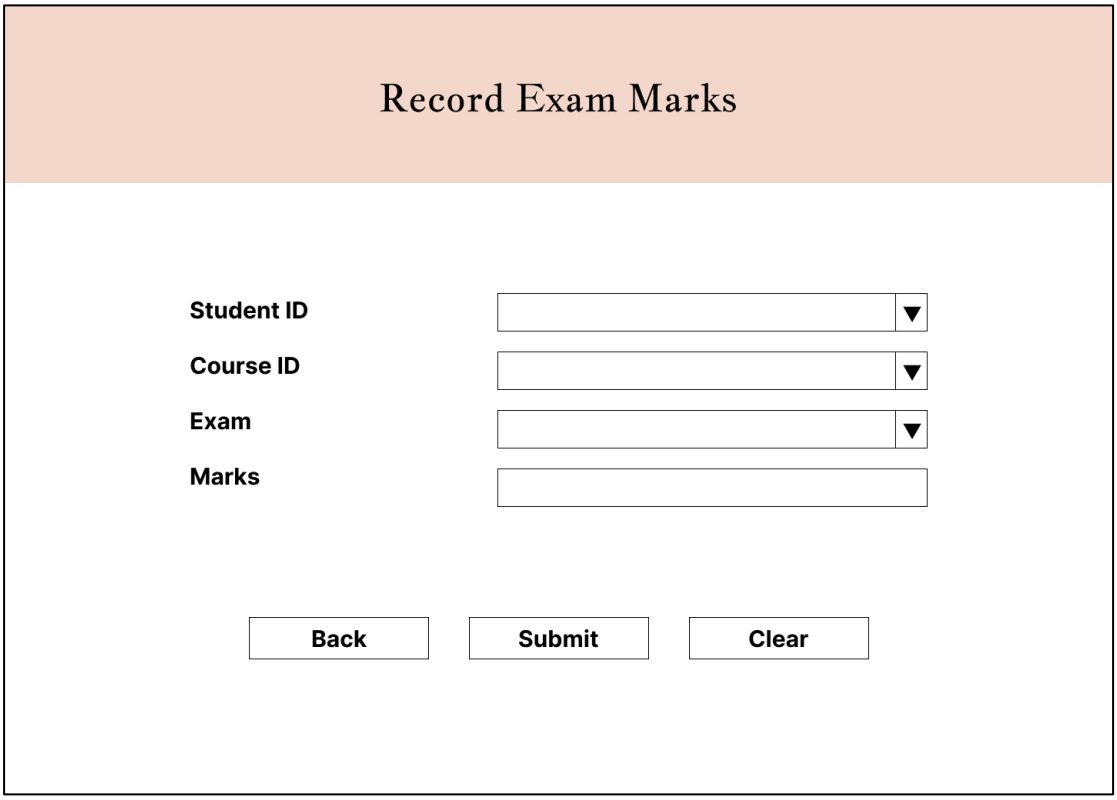

Figure 44: Store Exam Marks Screen

Pseudo Code

```
START
        INCLUDE DbCon
        //VALIDATIONS
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO MARKS (“STUDENTID”, “COURSEID”, “EXAM”,
                                                “MARKS”))
        END IF
END
```

Figure 45: Pseudo code to store exam marks

Interface No: 09

Interface Name: Ask Questions

Description: Used to ask questions from the lecturer

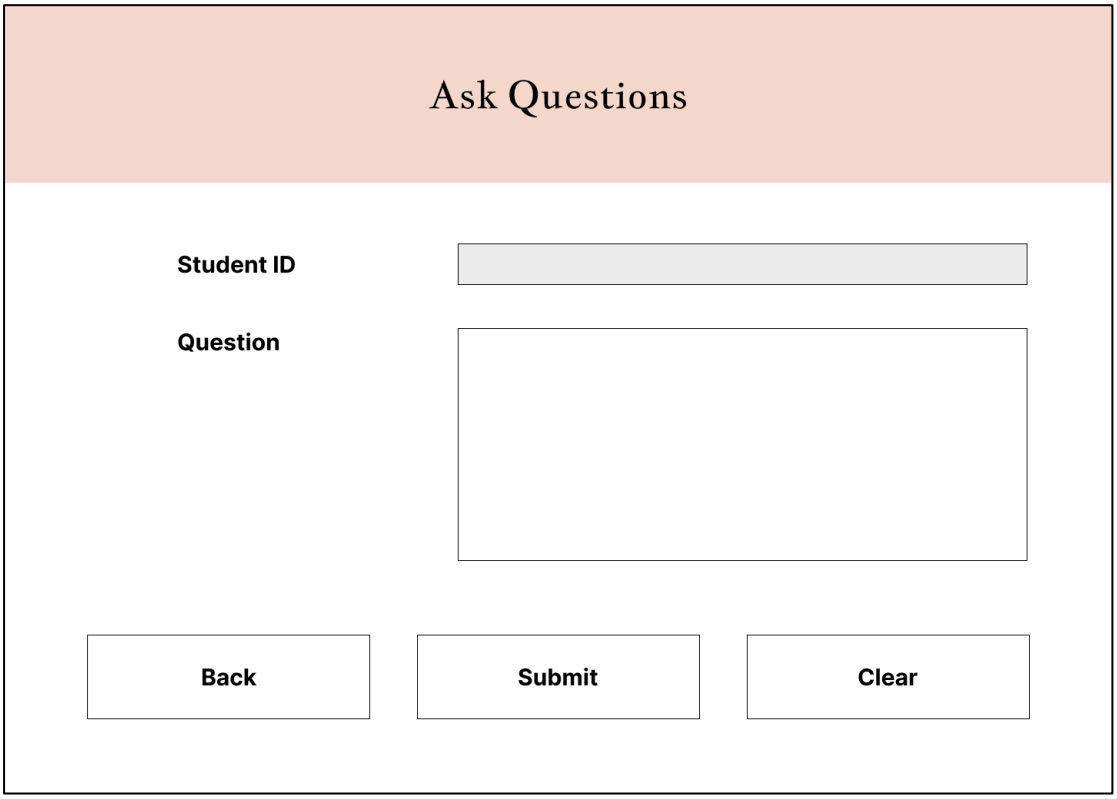

Figure 46: Ask Questions screen

Pseudo Code

```
START
        INCLUDE DbCon
        INPUT STUDENTID
        INPUT QUESTION
        OPEN DBCON
        IF INSERT_CLICKED
         THEN
                executeQuery(INSERT INTO QUESTIONS (STUDENTID, QUESTION))
        END IF
        CLOSE DBCON
END
```

Figure 47: Pseudo code to ask questions

Interface No: 10

Interface Name: View Results

Description: Used to view results of exams

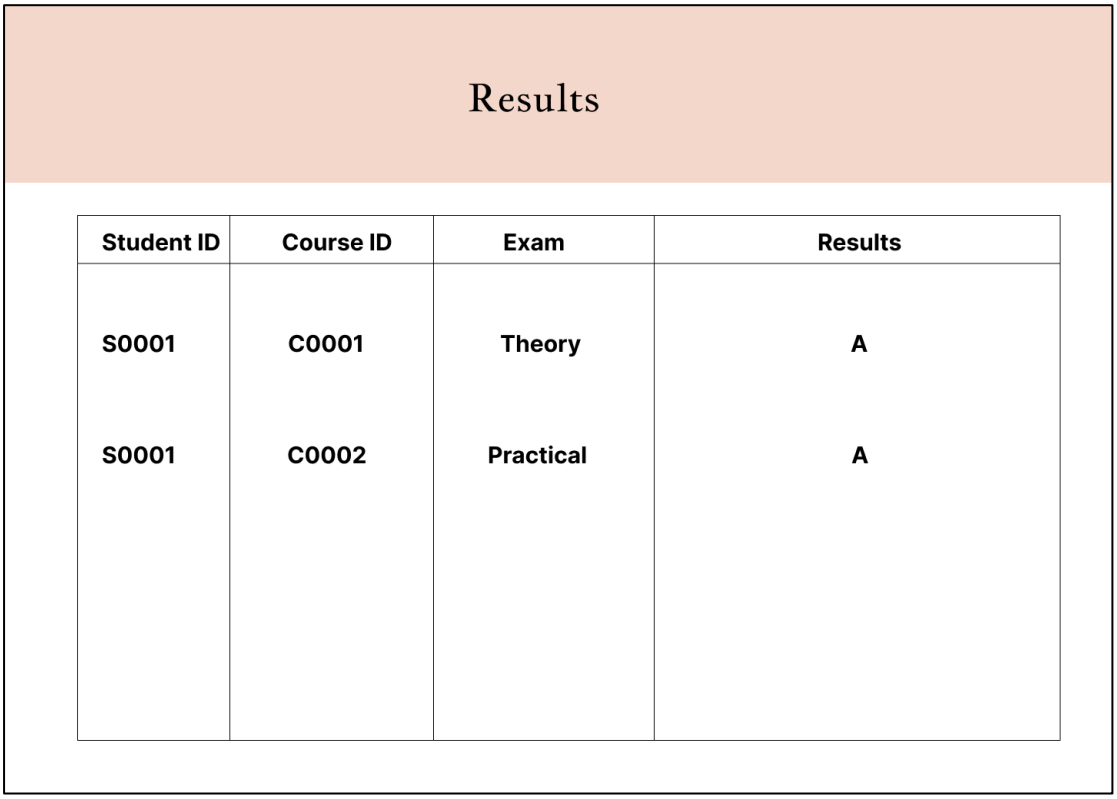

Figure 48: View Results Screen

Pseudo Code

```
START
        INCLUDE DbCon
        OPEN DBCON
        ON LOAD
                Result = executeQuery(SELECT * FROM RESULTS)
                OUTPUT Result
        END
        CLOSE DBCON
END
```

Figure 49: Pseudo code for view exam results

Interface No: 11

Interface Name: View Notes

Description: Used to view and select lecturer notes

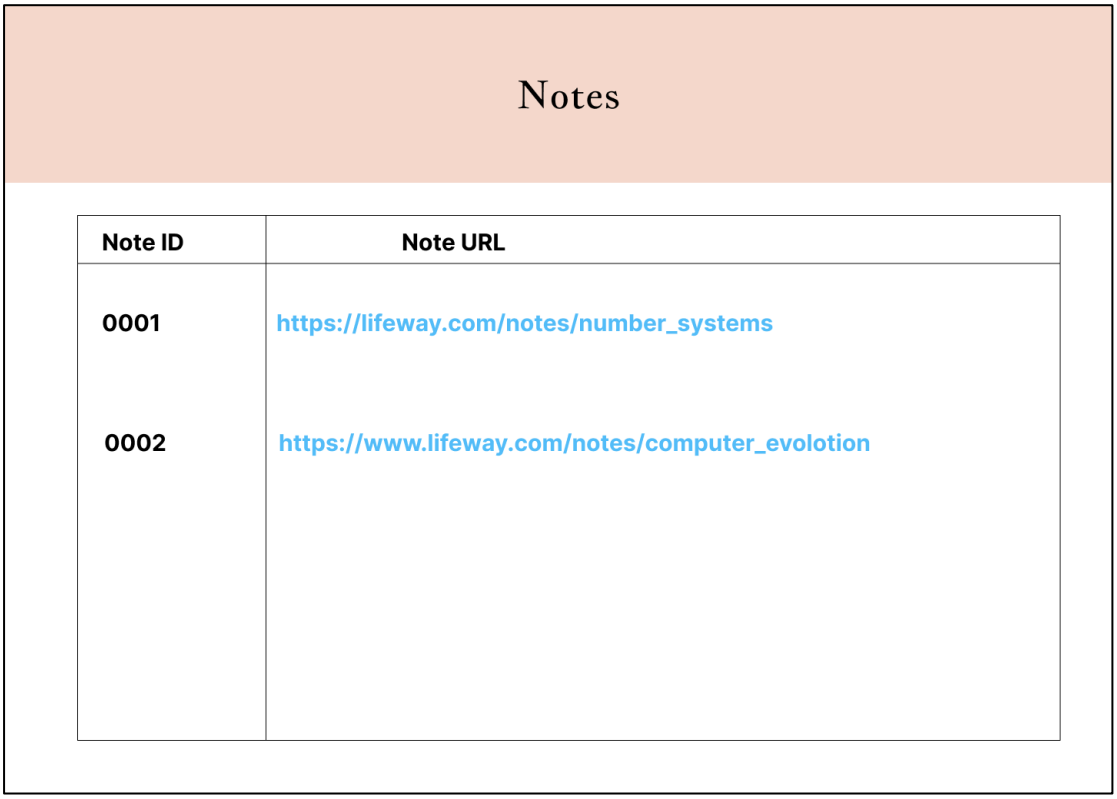

Figure 50: View Notes Screen

Pseudo Code

```
START
        INCLUDE DbCon
        OPEN DBCON
        ON LOAD
                Result = executeQuery(SELECT * FROM NOTES)
                OUTPUT Result
        END
END
```

Figure 51: Pseudo code for view notes

Interface No: 12

Interface Name: View Questions

Description: Used to view Questions that the students have sent and select and answer them

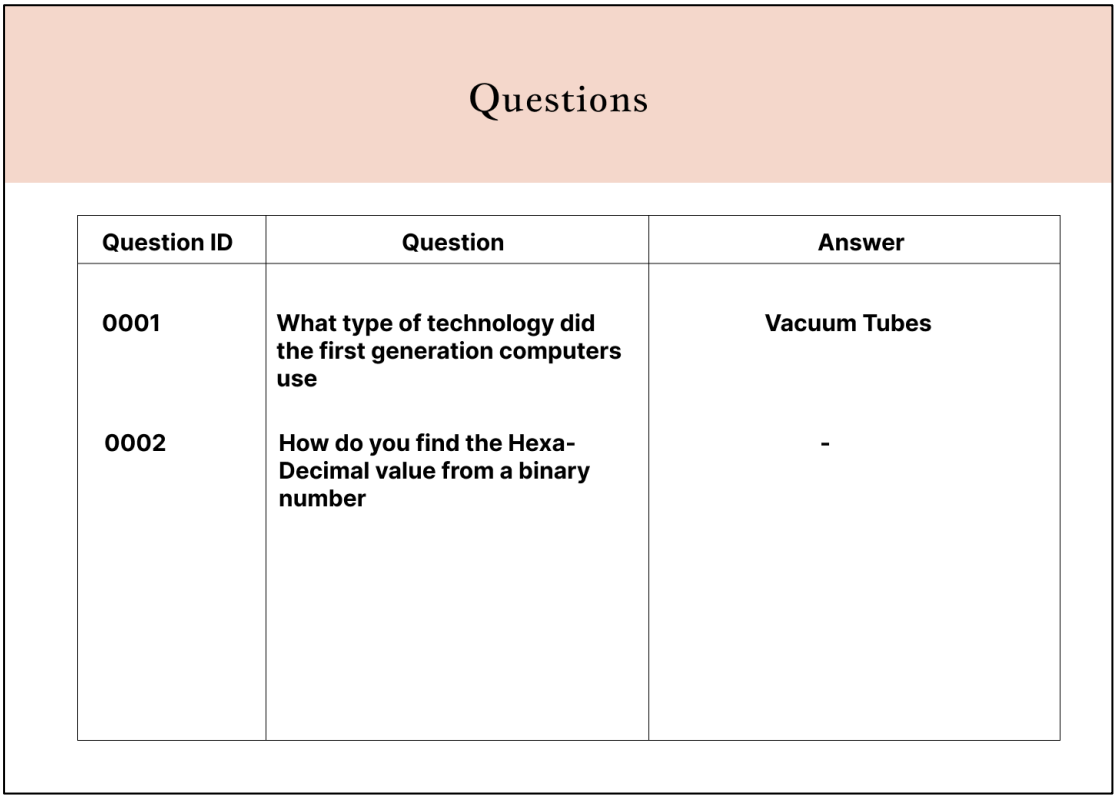

Figure 52: View Questions Screen

Pseudo Code

```
START
        INCLUDE DbCon
        ON LOAD
                Result = executeQuery(SELECT * FROM QUESTIONS)
                OUTPUT Result
        END
END
```

Figure 53: Pseudo code for View Questions

Interface No: 13

Interface Name: View Attendance

Description: Used to view the attendance of students

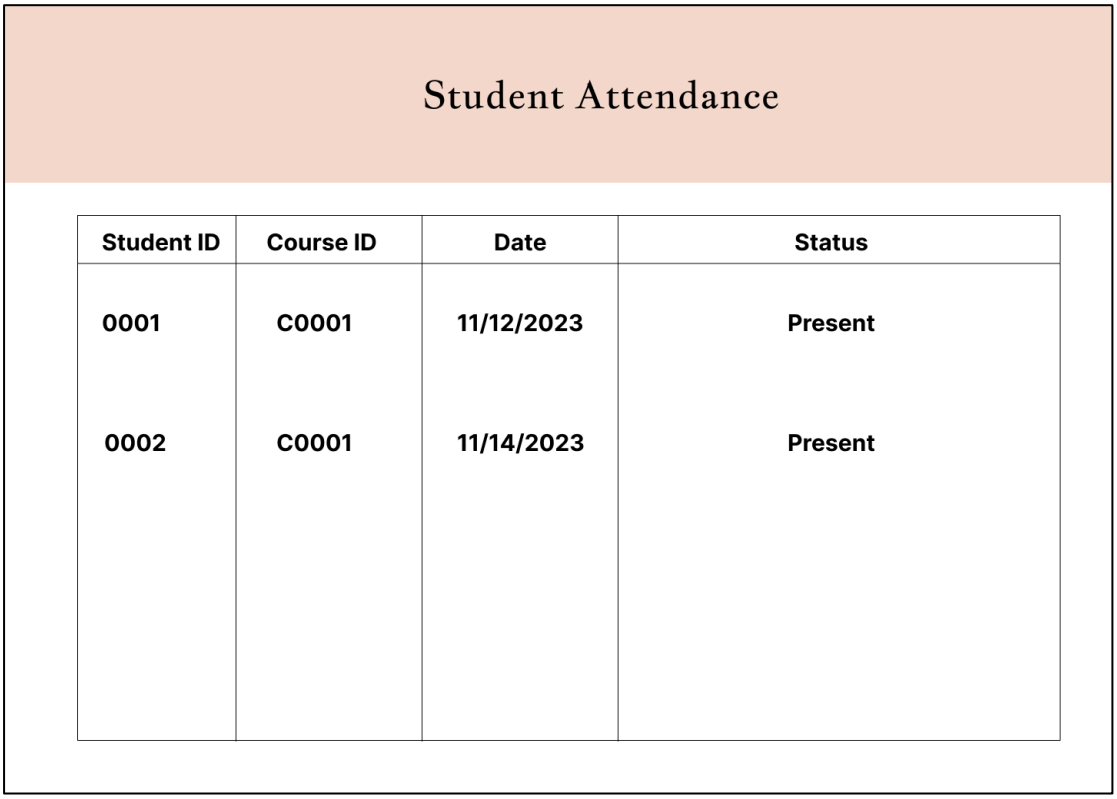

Figure 54: View Student Attendance Screen

Pseudo Code

```
START
        INCLUDE DbCon
        OPEN DBCON
        ON LOAD
                Result = executeQuery(SELECT * FROM ATTENDANCE)
                OUTPUT Result
        END
        CLOSE DBCON
END
```

Figure 55: Pseudo code for view student attendance

Interface No: 14

Interface Name: View Schedule

Description: Used to view the schedule

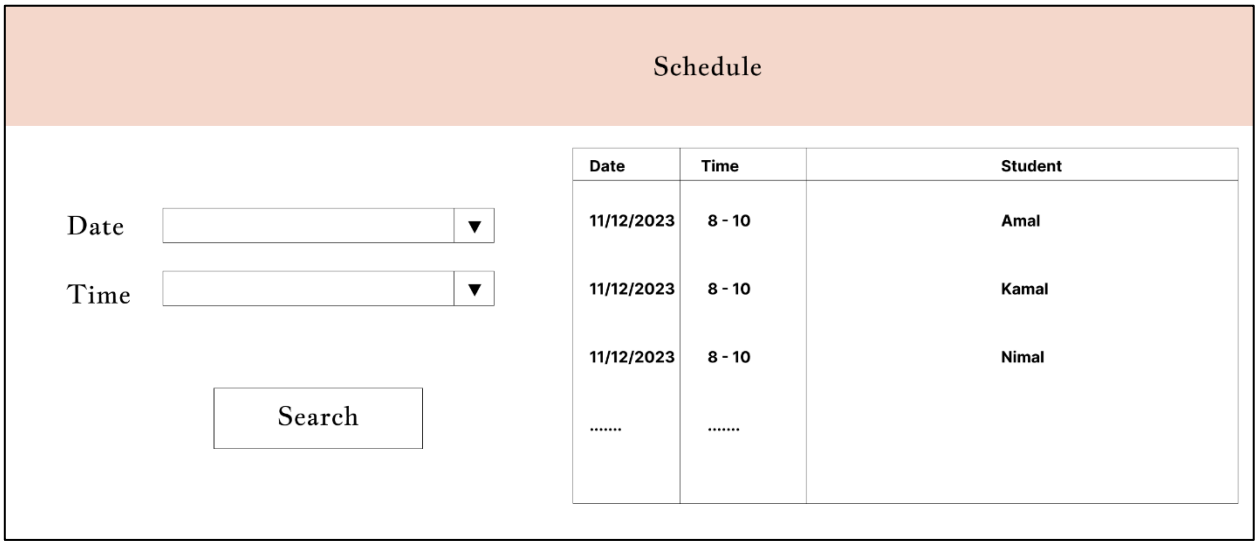

Figure 56: View Schedule Screen

Pseudo Code

```
START
        INCLUDE DbCon
        INPUT DATE
        INPUT TIME
        OPEN DBCON
        IF SEARCH_CLICKED
        THEN
                Result = executeQuery(SELECT * FROM QUESTIONS WHERE DATE="11/12/2023" AND
                TIME="8-10")
                OUTPUT Result
        END
        CLOSE DBCON
END
```

Figure 57: Pseudo code to view schedule

Interface No: 15

Interface Name: Home Screen

Description: Used to navigate through the system by clicking buttons

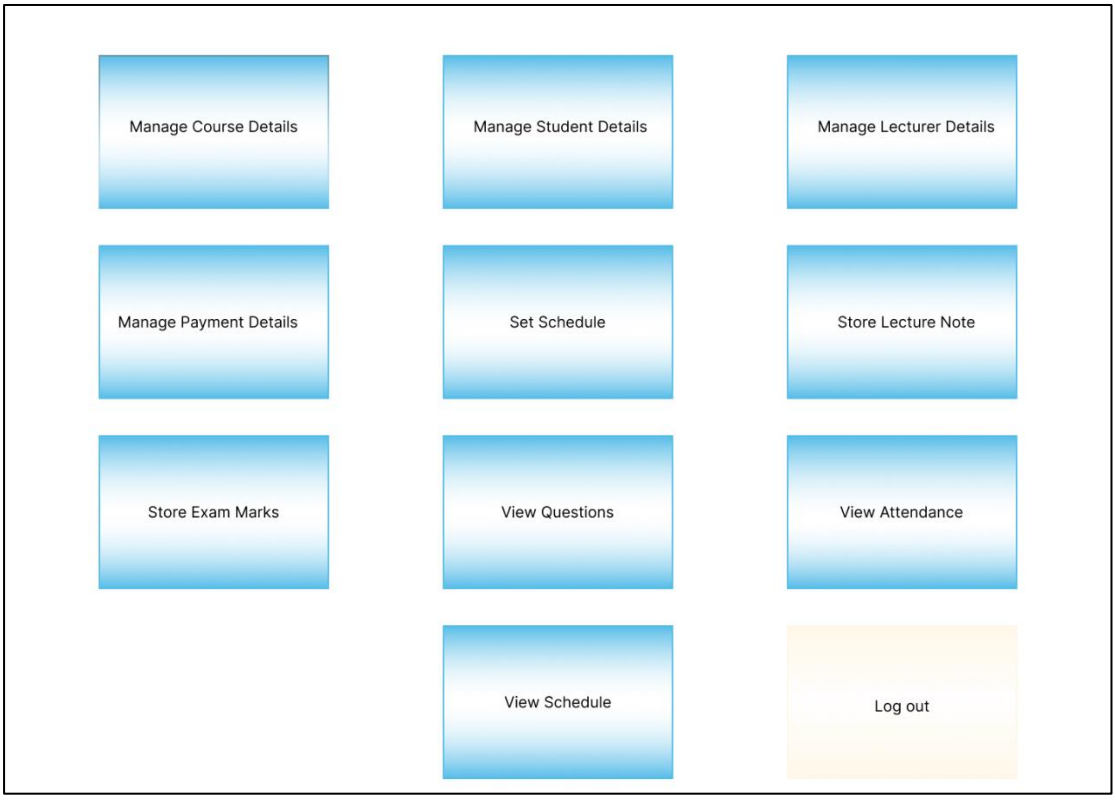

Figure 58: Home Screen

Pseudo Code

```
START
        IF COURSE_CLICKED
        THEN
                SHOW COURSE DETAILS WINDOW
        ELSE IF STUDENTS_CLICKED
        THEN
                SHOW STUDENTS DETAILS WINDOW
        ELSE IF LECTURER_CLCKED
        THEN
                SHOW LECTURER DETAILS WINDOW
        ELSE IF SCHEDULE_CLICKED
        THEN
                SHOW SCHEDULE WINDOW
        ELSE IF LECTURE NOTES_CLICKED
        THEN
                SHOW NOTES WINDOW
        IF LOGOUT_CLICKED
        THEN
                LOGOUT SESSION
                CLEAR DATA
                TERMINATE APPLICATION
        END IF

END
```

Figure 59: Pseudo code for home screen

Interface No: 16

Interface Name: Parent Details

Description: Used to store parent details of a student

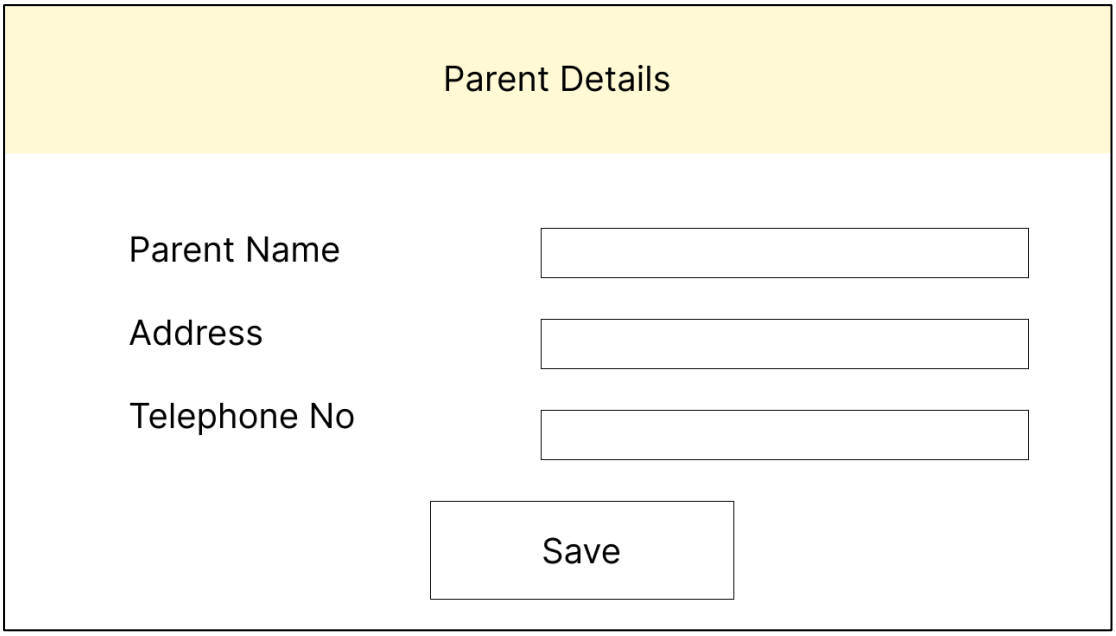

Figure 60: Parent Details Screen

Pseudo Code

```
START
        INCLUDE DbCon
        INPUT PARENT_NAME
        INPUT ADDRESS
        INPUT TELEPHONENO
        IF SAVE_CLICKED
        THEN
                ExecuteQuery(INSERT INTO PARENTS (PARENT_NAME, ADDRESS,
                                TELEPHONENO))
                OUTPIT SAVED
        END IF

END
```

Figure 61: Pseudo code to maintain Parent Details

4.3 Database Schema

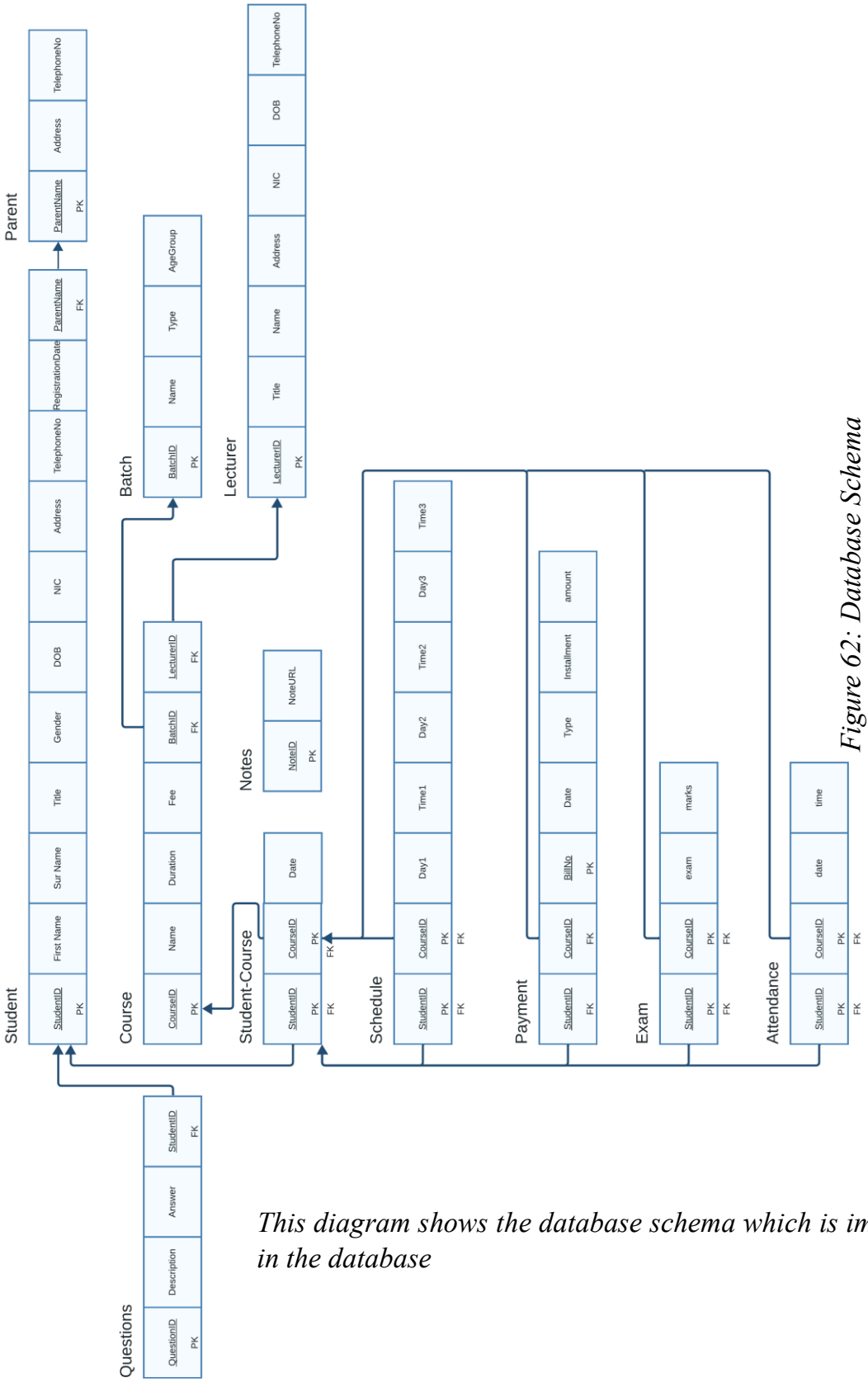

Figure 62: Database Schema

This diagram shows the database schema which is implemented in the database

4.4 System Requirements

Hardware

- Core i3 Processor
- 2GB RAM
- 1GB Hard Disc Space
- Monitor
- Webcam
- Keyboard and Mouse

Software

- Windows 7
- Windows 8
- Windows 10

(With .Net framework 3.5 above)

Chapter 5: Conclusion

Project Conclusion

Time, we all try to spend our valuable time as effectively as possible. For Mr. Dhammika this became a particularly hard thing to do. As he was playing both the manager and the sole lecturer at his institute. Because he was one the bet teachers in his area, more and more students enrolled in his classes. Although he loved teaching, he didn't like spending much time in record keeping. As his student count group grew day-by-day it came to a point where he couldn't manage the record-keeping work manually anymore. As one of our group members was one of his students, he knew exactly what Mr. Dhammika was going through.

So, we came together to build an application for him, to take him out of this record-keeping task. To do that first we went to his institute, we observed, and gathered all the needed information needed. Next, we discussed with Mr. Dhammika to get a clear idea of what he wants. And after confirming what his expectations are we started to build the application.

We followed all the standard practices that are involved in creating a good application. And we think that our application satisfies our client's expectations.

We worked hard and created a feature rich application for him to use which has all the functionality that he needed. Which were managing all the records related to his institute more easily, searching for records more easily, and overall, not being worried about human mistakes.

Also, we went beyond what he needed and created an application for his students to use as well. Where the students can ask questions, review notes, view exam results, view their schedule and much more.

Having an application that does the things that you were manually doing before saves you much more time and effort.

So, for the Achievements of the project

- Implemented a new record keeping system that is more efficient.
- Which saves time and effort needed to maintain data within the organization.
- Gives the ability to quickly search for something easily.
- Allows students to be involved in their studies than ever before.
- Allows students record their attendance more easily.

And, for the Weaknesses of the project

- Students will mark their attendance manually.

But overall, we think that we have achieved what this project meant to our client.

And it wouldn’t be possible if we didn’t have the support and guidance from our Supervisor, Course Directors, Lecturers, our family, and our friends. So, for that we would like to convey our heartfelt gratitude to every who helped us.

References

(2023) ‘Learn C# Programming’, Available at: https://www.programiz.com/csharp-programming (Accessed: 20 December 2023)

(2023) ‘C# Tutorial (C Sharp)’, Available at: https://www.w3schools.com/cs/index.php. (Accessed: 20 December 2023)

(2023) ‘C# documentation’, Available at: https://learn.microsoft.com/en-us/dotnet/csharp/ (Accessed: 20 December 2023)

Appendices

Work Breakdown Chart

The below chart shows how we managed the given time to complete our project.

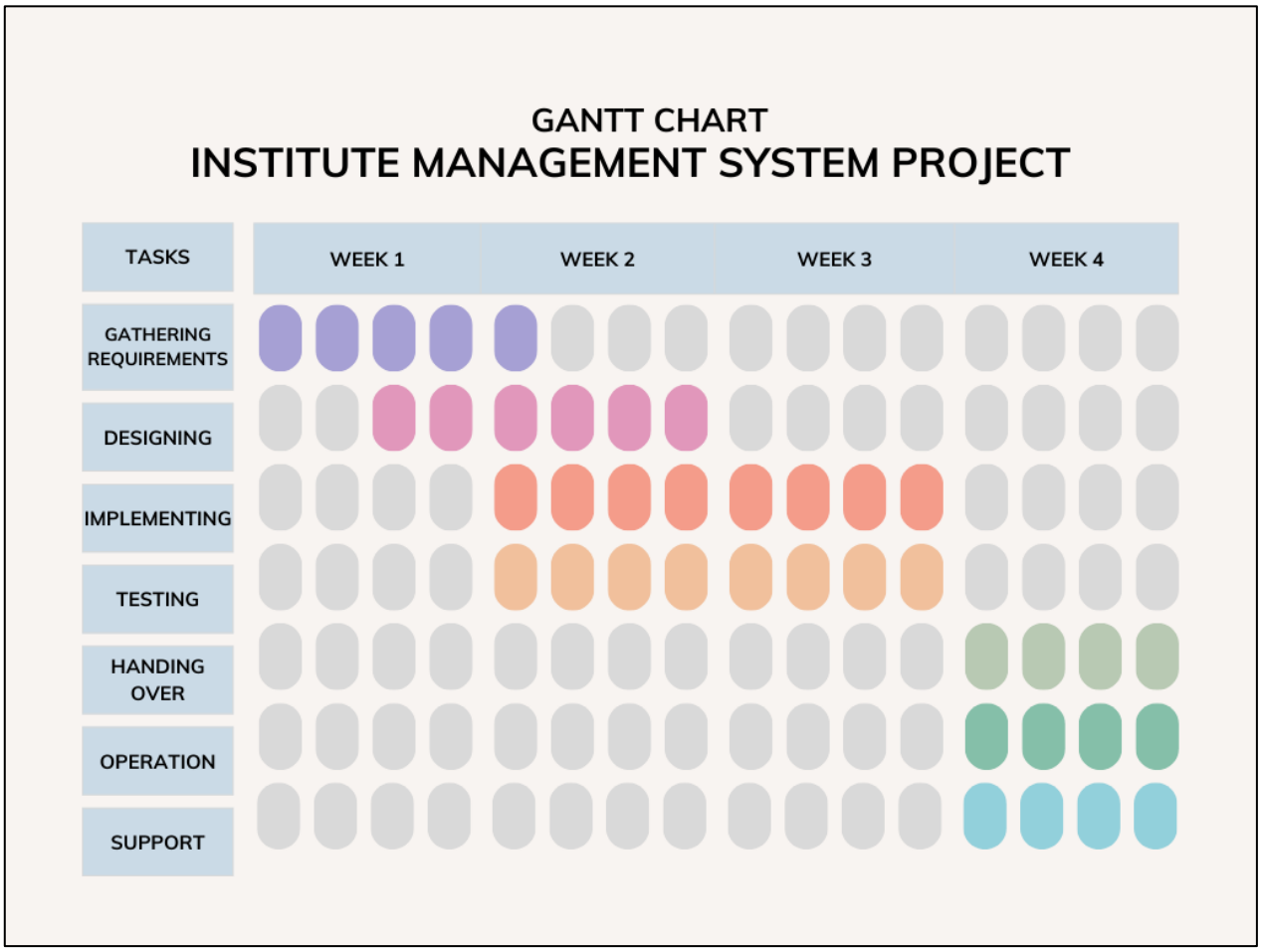

Figure 63: Work Breakdown Sheet

Future Enhancements

As for the future enhancements, we are looking forward to improving the student attendance marking system by introducing a mobile-based application.

And to provide more functionality to the students.

Also, we will help Mr. Dhammika with any additional features that he might want in the future.

Program Code

Desktop Application Code

Student Entity Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lifeway_Institute_Management_System
{
    public class Student
    {
        private String studentID;
        private String fname;
        private String surname;
        private String title;
        private String gender;
        private String dob;
        private int nic;
        private String address;
        private int tel;
        private String registrationDate;
        private String parent;

        public Student()
        {
            studentID = "";
            fname = "";
            surname = "";
            title = "";
            gender = "";
            dob = "";
            nic = 0;
            address = "";
            tel = 0;
            registrationDate = "";
            parent = "";
        }

        public String StudentID
        {
            get { return studentID; }
            set { studentID = value; }
        }

        public String Fname
        {
            get { return fname; }
```

```
    set { fname = value; }
}

public String Surname
{
    get { return surname; }
    set { surname = value; }
}

public String Title
{
    get { return title; }
    set { title = value; }
}

public String Gender
{
    get { return gender; }
    set { gender = value; }
}

public String DOB
{
    get { return dob; }
    set { dob = value; }
}

public int NIC
{
    get { return nic; }
    set { nic = value; }
}

public String Address
{
    get { return address; }
    set { address = value; }
}

public int Tel
{
    get { return tel; }
    set { tel = value; }
}

public String Registrationdate
{
    get { return registrationDate; }
    set { registrationDate = value; }
}

public String Parent
{
    get { return parent; }
```

```
    set { parent = value; }
}
```

Student Database Class

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lifeway_Institute_Management_System
{
    internal class StudentDb : DbConnection, DbOperations
    {
        Student student = new Student();
        MySqlConnection con;
        MySqlCommand com;

        public StudentDb()
        {

        }
        public StudentDb(Student student)
        {
            this.student = student;
        }

        public void insert()
        {
            con = getConnection();
            String query = "Insert into Students(Fname, Surname,
Title, Gender, DOB, NIC, Address, Tel, RegistrationDate, Parent) " +

"values( '"+student.Fname+"'," +

"'"+student.Surname+"'," +

"'"+student.Title+"'," +

"'"+student.Gender+"'," +

"'"+student.DOB+"'," +

""+student.NIC+"," +

"'"+student.Address+"'," +

""+student.Tel+"," +

"'"+student.Registrationdate+"'," +

"'"+student.Parent+"')";
            com = new MySqlCommand(query, con);

            con.Open();
            com.ExecuteNonQuery();
```

```
            con.Close();
        }

        public void update()
        {
            con = getConnection();
            String query = "Update Students set Address = '" +
student.Address + "', Tel = " + student.Tel + " " +
                                                    "where StudentID =
"+student.StudentID+"";
            com = new MySqlCommand(query, con);

            con.Open();
            com.ExecuteNonQuery();
            con.Close();
        }

        public void delete()
        {
            con = getConnection();
            String query = "Delete from students where StudentID =
'" + student.StudentID + "'";
            com = new MySqlCommand(query, con);

            con.Open();
            com.ExecuteNonQuery();
            con.Close();
        }

        //gets the most recent studentID
        public int getID()
        {
            try
            {
                int ID = 0;

                con = getConnection();
                String query = "Select max(StudentID) from
Students";
                com = new MySqlCommand(query, con);

                con.Open();
                MySqlDataReader dr = com.ExecuteReader();

                while (dr.Read())
                {
                    ID = dr.GetInt32(0);
                }

                con.Close();
                return ID;
            }
            catch (Exception ex)
            {
```

```
            showErrorMessage(ex, this);
            con.Close();
            return 0;
        }
    }

    //gets a list of stored IDs
    public List<int> getIDs()
    {
        con = getConnection();
        String query = "Select StudentID from Students";
        com = new MySqlCommand(query, con);

        con.Open();
        List<int> IDs = new List<int>();

        MySqlDataReader dr = com.ExecuteReader();

        while (dr.Read())
        {
            IDs.Add(dr.GetInt32(0));
        }
        con.Close();
        return IDs;
    }

    //gets a list of stored First Names
    public List<String> getFNames()
    {
        con = getConnection();
        String query = "Select Fname from Students";
        com = new MySqlCommand(query, con);

        con.Open();
        MySqlDataReader dr = com.ExecuteReader();
        List<String> names = new List<String>();

        while (dr.Read())
        {
            names.Add(dr.GetString(0));
        }

        con.Close();
        return names;
    }

    //gets a list of stored Last Names
    public List<String> getSNames()
    {
        con = getConnection();
        String query = "Select Surname from Students";
        com = new MySqlCommand(query, con);

        con.Open();
```

```
            MySqlDataReader dr = com.ExecuteReader();
            List<String> names = new List<String>();

            while (dr.Read())
            {
                names.Add(dr.GetString(0));
            }

            con.Close();
            return names;
        }

        //gets details of student whose ID is passed
        public Student getStudent(int ID)
        {
            Student student = new Student();

            con = getConnection();
            String query = "Select * from Students where StudentID =
" + ID + "";
            com = new MySqlCommand(query, con);

            con.Open();
            MySqlDataReader dr = com.ExecuteReader();

            dr.Read();

            student.StudentID = dr.GetString(0);
            student.Fname = dr.GetString(1);
            student.Surname = dr.GetString(2);
            student.Title = dr.GetString(3);
            student.Gender = dr.GetString(4);
            student.DOB = dr.GetString(5);
            student.NIC = dr.GetInt32(6);
            student.Address = dr.GetString(7);
            student.Tel = dr.GetInt32(8);
            student.Registrationdate = dr.GetString(9);
            student.Parent = dr.GetString(10);

            con.Close();

            return student;
        }

        //gets detials of student whose first name and last name is
passed
        public Student getStudent(String fname, String sname)
        {
            Student student = new Student();

            con = getConnection();
            String query = "Select * from Students where Fname =
'"+fname+"' and Surname = '"+sname+"'";
            com = new MySqlCommand(query, con);
```

```
            con.Open();
            MySqlDataReader dr = com.ExecuteReader();

            dr.Read();

            student.StudentID = dr.GetString(0);
            student.Fname = dr.GetString(1);
            student.Surname = dr.GetString(2);
            student.Title = dr.GetString(3);
            student.Gender = dr.GetString(4);
            student.DOB = dr.GetString(5);
            student.NIC = dr.GetInt32(6);
            student.Address = dr.GetString(7);
            student.Tel = dr.GetInt32(8);
            student.Registrationdate = dr.GetString(9);
            student.Parent = dr.GetString(10);

            con.Close();

            return student;
        }
    }
}
```

Manage Student Details Form

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lifeway_Institute_Management_System
{
    public partial class frmStudent : Form
    {
        String type = "";
        Student student = new Student();
        StudentDb studentdb = new StudentDb();
        frmHome frmHome = new frmHome();
        public frmStudent()
        {
            InitializeComponent();
        }

        public frmStudent(frmHome frmHome)
        {
            InitializeComponent();
            this.frmHome = frmHome;
        }

        //clears the text fields
        public void clear()
```

```
{
    cboID.Text = "";
    cbo_firstname.Text = "";
    cbo_firstname.Items.Clear();
    cbo_surname.Text = "";
    cbo_surname.Items.Clear();
    txttitle.Clear();
    cbogender.SelectedIndex = -1;
    txtdob.Clear();
    txt_NIC.Clear();
    txt_address.Clear();
    txt_tel.Clear();
    txtRegistrationDate.Clear();
    cbo_Parent.SelectedIndex = -1;
    cbo_Parent.Text = "";

    if (type == "insert")
    {
        setInsertEnvironment();
    }
    else if (type == "update")
    {
        setUpdateEnvironment();
    }
    else if (type == "delete")
    {
        setDeleteEnvironment();
    }
    else if (type == "select")
    {
        setSelectEnvironment();
    }
```

```
}
public void populateIDs()
{
    cboID.Items.Clear();
    List<int> IDs = studentdb.getIDs();

    foreach (int ID in IDs)
    {
        cboID.Items.Add(ID);
    }
}

public void populateNames()
{
    cbo_firstname.Items.Clear();
    cbo_surname.Items.Clear();

    List<String> fnames = new List<String>();
    List<String> snames = new List<String>();

    fnames = studentdb.getFNames();
    snames = studentdb.getSNames();

    foreach (String name in fnames)
    {
        cbo_firstname.Items.Add(name);
    }

    foreach (String name in snames)
    {
        cbo_surname.Items.Add(name);
    }
```

```
}

public void populateParentNames()
{
    cbo_Parent.Items.Clear();

    ParentDb parentdb = new ParentDb();

    List<String> names = parentdb.getNames();

    foreach (String name in names)
    {
        cbo_Parent.Items.Add(name);
    }

    cbo_Parent.Items.Add("Add New...");
}

public void setInsertEnvironment()
{
    lblID.Visible = false;
    cboID.Visible = false;
    cbo_firstname.Enabled = true;
    cbo_surname.Enabled = true;
    txttitle.Enabled = true;
    cbogender.Enabled = true;
    txtdob.Enabled = true;
    txt_address.Enabled = true;
    txt_NIC.Enabled = true;
    txt_tel.Enabled = true;
    txtRegistrationDate.Enabled = true;
    cbo_Parent.Enabled = true;
```

```
            btnGetDetails.Visible = false;
            btnSubmit.Visible = true;
            cbo_firstname.Items.Clear();
            cbo_surname.Items.Clear();
            txtRegistrationDate.Text                                    =
System.DateTime.Now.Date.ToShortDateString();
            populateParentNames();
        }

        public void setUpdateEnvironment()
        {
            lblID.Visible = true;
            cboID.Visible = true;
            cboID.Enabled = true;
            cbo_firstname.Enabled = false;
            cbo_surname.Enabled = false;
            txttitle.Enabled = false;
            cbogender.Enabled = false;
            txtdob.Enabled = false;
            txt_address.Enabled = false;
            txt_NIC.Enabled = false;
            txt_tel.Enabled = false;
            txtRegistrationDate.Enabled = false;
            cbo_Parent.Enabled = false;

            cbo_firstname.Items.Clear();
            cbo_surname.Items.Clear();
            btnGetDetails.Visible = false;
            btnSubmit.Visible = false;
            populateIDs();
        }
```

```
public void setDeleteEnvironment()
{
    lblID.Visible = false;
    cboID.Visible = false;
    cbo_firstname.Enabled = true;
    cbo_surname.Enabled = true;
    txttitle.Enabled = false;
    cbogender.Enabled = false;
    txtdob.Enabled = false;
    txt_address.Enabled = false;
    txt_NIC.Enabled = false;
    txt_tel.Enabled = false;
    txtRegistrationDate.Enabled = false;
    cbo_Parent.Enabled = false;

    btnSubmit.Visible = false;
    btnGetDetails.Visible = false;
    populateNames();
}

public void setSelectEnvironment()
{
    lblID.Visible = false;
    cboID.Visible = false;
    cbo_firstname.Enabled = true;
    cbo_surname.Enabled = true;
    txttitle.Enabled = false;
    cbogender.Enabled = false;
    txtdob.Enabled = false;
    txt_address.Enabled = false;
    txt_NIC.Enabled = false;
    txt_tel.Enabled = false;
```

```
    txtRegistrationDate.Enabled = false;
    cbo_Parent.Enabled = false;

    btnSubmit.Visible = false;
    btnGetDetails.Visible = false;
    populateNames();
}

private void btnInsert_Click(object sender, EventArgs e)
{
    type = "insert";
    clear();
}

private void btnSubmit_Click(object sender, EventArgs e)
{
    if (type == "insert")
    {
        //validations
        if (cbo_firstname.Text == "")
        {
            MessageBox.Show("Must Enter Student First Name");
            return;
        }

        if (cbo_surname.Text == "")
        {
            MessageBox.Show("Must Enter Student Sur Name");
            return;
        }

        if (cbogender.SelectedIndex == -1)
```

```
                {
                    MessageBox.Show("Must select gender");
                    return;
                }

                if (txtdob.Text == "")
                {
                    MessageBox.Show("Must  select  student  Date  of
Birth");
                    return;
                }

                if (txt_NIC.Text == "")
                {
                    MessageBox.Show("Must enter student NIC");
                    return;
                }

                if (txt_NIC.Text.Length != 10)
                {
                    MessageBox.Show("Must enter valid NIC number");
                    return;
                }

                if (txt_address.Text == "")
                {
                    MessageBox.Show("Must enter address of student");
                    return;
                }

                if (txt_tel.Text == "")
                {
```

```
                MessageBox.Show("Must enter student contact
number");
                return;
            }

            if (txt_tel.Text.Length < 10)
            {
                MessageBox.Show("Must enter a valid Telephone
No");
                return;
            }

            if (cbo_Parent.SelectedIndex == -1)
            {
                MessageBox.Show("Must select Parent Name");
                return;
            }

            Student student = new Student();

            student.Fname = cbo_firstname.Text;
            student.Surname = cbo_surname.Text;
            student.Title = txttitle.Text;
            student.Gender = cbogender.SelectedItem.ToString();
            student.DOB = txtdob.Text;
            student.NIC = Convert.ToInt32(txt_NIC.Text);
            student.Address = txt_address.Text;
            student.Tel = Convert.ToInt32(txt_tel.Text);
            student.Registrationdate = txtRegistrationDate.Text;
            student.Parent = cbo_Parent.SelectedItem.ToString();

            studentdb = new StudentDb(student);
```

```
                studentdb.insert();


                int studentID = studentdb.getID();


                if (studentID != 0)

                {

                    MessageBox.Show("Student             successfully
registered\nStudentID: " + studentID);

                }


                clear();


                if (MessageBox.Show("Do you want to continue to
register the student with a course?", "", MessageBoxButtons.YesNo) ==
DialogResult.Yes)

                {

                    this.Close();

                    frmStudentCourse    frmstudentcourse    =    new
frmStudentCourse(frmHome, studentID);

                    frmstudentcourse.Show();

                }

            }

            else if (type == "update")

            {

                student.Address = txt_address.Text;

                student.Tel = Convert.ToInt32(txt_tel.Text);


                if (MessageBox.Show("Do you really want to update the
details  of  "  +  student.Fname  +  "  "  +  student.Surname,  "Update
Confirmation            Dialog",            MessageBoxButtons.YesNo,
MessageBoxIcon.Information) == DialogResult.Yes)

                {

                    studentdb = new StudentDb(student);

                    studentdb.update();
```

```
                    MessageBox.Show("Reccord  successfully  updated",
"Updated Message");
                    clear();
                }
                else
                {
                    MessageBox.Show("Student Details not updated");
                }
            }
            else if (type == "delete")
            {
                if (MessageBox.Show("Do you really want to delete the
details  of  "  +  student.Fname  +  "  "  +  student.Surname,  "Delete
Confirmation            Dialog",            MessageBoxButtons.YesNo,
MessageBoxIcon.Information) == DialogResult.Yes)
                {
                    studentdb = new StudentDb(student);
                    studentdb.delete();
                    MessageBox.Show("Reccord  successfully  deleted",
"Deleted Message");
                    clear();
                }
                else
                {
                    MessageBox.Show("Student Details not deleted");
                }
            }
        }


        private  void  cbogender_SelectedIndexChanged(object  sender,
EventArgs e)
        {
            if (cbogender.SelectedIndex != -1)
            {
```

```
            if (cbogender.SelectedItem.ToString() == "Male")
            {
                txttitle.Text = "Mr.";
            }
            else
            {
                txttitle.Text = "Miss.";
            }
        }
    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        clear();
    }

    private   void   cboID_SelectedIndexChanged(object   sender,
EventArgs e)
    {
        if (cboID.SelectedIndex != -1)
        {
            btnGetDetails.Visible = true;
        }
    }

    private void btnGetDetails_Click(object sender, EventArgs e)
    {
        if (type == "update")
        {
            int                         ID                         =
Convert.ToInt32(cboID.SelectedItem.ToString());

            student = studentdb.getStudent(ID);
```

```
    cboID.Enabled = false;
    txt_address.Enabled = true;
    txt_tel.Enabled = true;
    btnSubmit.Visible = true;
}
else if (type == "delete" || type == "select")
{
    String fname = cbo_firstname.SelectedItem.ToString();
    String sname = cbo_surname.SelectedItem.ToString();

    student = studentdb.getStudent(fname, sname);

    cbo_firstname.Enabled = false;
    cbo_surname.Enabled = false;
}

if (type == "delete")
{
    btnSubmit.Visible = true;
}

cbo_firstname.Text = student.Fname;
cbo_surname.Text = student.Surname;
txttitle.Text = student.Title;
cbogender.Text = student.Gender;
txtdob.Text = student.DOB;
txt_NIC.Text = student.NIC.ToString();
txt_address.Text = student.Address;
txt_tel.Text = student.Tel.ToString();
txtRegistrationDate.Text = student.Registrationdate;
cbo_Parent.Text = student.Parent;
```

```
            btnGetDetails.Visible = false;
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            type = "update";
            clear();
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            type = "delete";
            clear();
        }

        private    void    cbo_firstname_SelectedIndexChanged(object
sender, EventArgs e)
        {
            if (type == "delete" || type == "select")
            {
                if    (cbo_firstname.SelectedIndex    !=    -1    &&
cbo_surname.SelectedIndex != -1)
                {
                    btnGetDetails.Visible = true;
                }
            }
        }

        private void cbo_surname_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (type == "delete" || type == "select")
```

```
            {
                if    (cbo_firstname.SelectedIndex    !=    -1    &&
cbo_surname.SelectedIndex != -1)
                {
                    btnGetDetails.Visible = true;
                }
            }
        }

        private void btnSelect_Click(object sender, EventArgs e)
        {
            type = "select";
            clear();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private  void  cbo_Parent_SelectedIndexChanged(object  sender,
EventArgs e)
        {
            if (cbo_Parent.Text != "")
            {
                if   (cbo_Parent.SelectedItem.ToString()   ==   "Add
New...")
                {
                    cbo_Parent.SelectedIndex = -1;
                    this.Hide();
                    btnrefresh.Visible = true;

                    frmParent frmParent = new frmParent(this);
```

```
                frmParent.Show();
            }
        }
    }

    private void btnrefresh_Click(object sender, EventArgs e)
    {

        populateParentNames();
        btnrefresh.Visible = false;
    }

    private     void     frmStudent_FormClosed(object     sender,
FormClosedEventArgs e)
    {
        frmHome.Show();
    }
}
}
```

DbConnection Class

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using MySql.Data.MySqlClient;

namespace Lifeway_Institute_Management_System
{
    public class DbConnection
    {
        private MySqlConnection connection;

        private String host = "localhost";
        private String username = "root";
        private String pwd = "";
        private String database = "Lifeway";
        private String str;

        public DbConnection() {
            //sets the connection string to connect with the
database
            this.str = "server=" + host + ";user=" + username +
";pwd=" + pwd + ";database=" + database + "";

            //create the connection to the database
            try
            {
                connection = new MySqlConnection(str);
            }
            catch (Exception ex)
            {
                showErrorMessage(ex, this);
            }
        }
        public void showErrorMessage(Exception ex, Object from)
        {
            MessageBox.Show("Something went wrong please contact
developer \nError Details\nError generated : " +from.GetType()+
"\nError code: " +ex.Message);
        }

        public MySqlConnection getConnection()
        {
            return this.connection;
        }
    }
}
```

DbOperations Interface

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lifeway_Institute_Management_System
{
    public interface DbOperations
    {
        public void insert();

        public void update();

        public void delete();
    }
}
```

Home Window

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lifeway_Institute_Management_System
{
    public partial class frmHome : Form
    {
        frmLogin frmlogin;
        public frmHome()
        {
            InitializeComponent();
        }

        public frmHome(frmLogin frmlogin)
        {
            InitializeComponent();
            this.frmlogin = frmlogin;
        }

        private void btnManageStudents_Click(object sender,
EventArgs e)
        {
            this.Hide();
            frmStudent frmStudent = new frmStudent(this);
            frmStudent.Show();
        }

        private void btnBack_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnManageLecturer_Click(object sender,
EventArgs e)
        {
            this.Hide();
            frmLecturer frmLecturer = new frmLecturer(this);
            frmLecturer.Show();
        }

        private void frmHome_FormClosed(object sender,
FormClosedEventArgs e)
        {
            frmlogin.Show();
        }
```

```
        private void btnManageCourses_Click(object sender, EventArgs
e)
        {
            this.Hide();
            frmCourse frmcourse = new frmCourse(this);
            frmcourse.Show();
        }

        private void btnManageBatches_Click(object sender, EventArgs
e)
        {
            this.Hide();
            frmBatch frmbatch = new frmBatch(this);
            frmbatch.Show();
        }

        private void btnPayments_Click(object sender, EventArgs e)
        {
            this.Hide();
            frmPayment frmpayment = new frmPayment(this);
            frmpayment.Show();
        }

        private void btnStudentCourse_Click(object sender, EventArgs
e)
        {
            this.Hide();
            frmStudentCourse frmstudentcourse = new
frmStudentCourse(this);
            frmstudentcourse.Show();
        }

        private void btnNotes_Click(object sender, EventArgs e)
        {
            this.Hide();
            frmNotes frmnotes = new frmNotes(this);
            frmnotes.Show();
        }

        private void btnManageSchedule_Click(object sender,
EventArgs e)
        {
            this.Hide();
            frmSchedule frmSchedule = new frmSchedule(this);
            frmSchedule.Show();
        }

        private void btnAnswerQuestions_Click(object sender,
EventArgs e)
        {
            this.Hide();
            frmAnswers frmAnswers = new frmAnswers(this);
            frmAnswers.Show();
```

```
        }

        private void btnExamMarks_Click(object sender, EventArgs e)
        {
            this.Hide();
            frmexamMarks frmexammarks = new frmexamMarks(this);
            frmexammarks.Show();
        }

        private void btnViewSchedule_Click(object sender, EventArgs
e)
        {
            this.Hide();
            frmViewSchedule frmViewSchedule = new
frmViewSchedule(this);
            frmViewSchedule.Show();
        }

        private void btnViewAttendance_Click(object sender,
EventArgs e)
        {
            this.Hide();
            frmViewAttendance frmviewAttendance = new
frmViewAttendance(this);
            frmviewAttendance.Show();
        }
    }
}
```

Student Website Application Code

Ask Questions Web Page

```
<?php
    include_once 'DbConnection.php';
    $query = "Select studentid from students";
    $res = mysqli_query($con, $query);
    $no_rows = mysqli_num_rows($res);

    $c = 1;
?>

<html>
    <head>
        <Title>Ask Questions</Title>

        <link rel="stylesheet" href = "stylings.css"/>
    </head>

    <body style = "display: none;">
        <h1 id = "heading">Feel free to ask any Question you have
regarding you're studies</h1>
        <table cellspacing = "0px" cellpadding = "5px" align =
"center">

            <tr>
                <td id = "Title">Ask Questions</td>
            </tr>

            <!--Question form-->
            <tr>
                <td>
                    <form action="save-questions.php" method =
"POST">
```

```
                        <!--Label and ComboBox to select StudentID
to submit question-->
                        <label for="studentID">Student ID</label>
                        <select required name = "studentID">
                            <option value="">Please Select
StudentID</option>
                            <?php while ( $c <= $no_rows ) {$row =
mysqli_fetch_assoc($res);?>
                                <option value = '<?php echo
$row['studentid'];?>'>
                                    <?php echo $row['studentid'];?>
                                </option>
                            <?php $c = $c + 1; } ?>
                        </select> <br> <br>


                        <!--Label and TextArea to note the question-
->
                        <label for="question">Question</label> <br>
                        <textarea required rows = "10" cols = "40"
name = "question"></textarea> <br><br>

                        <!--Button to submit question-->
                        <button type = "Submit" name = "submit" id =
"submit">
                            Submit
                        </button>
                    </form>
                </td>
            </tr>

        </table>

        <script src = "JQuery.js"></script>
```

```
        <script>
            $(function(){
                $('body').fadeIn(1000);
            });
        </script>
    </body>
</html>
```

View Results Web Page

```
<?php
    include_once "DbConnection.php";
    $studentID = $_POST['studentID'];
    $query = "Select * from exam_marks where studentID =
".$studentID."";
    $res = mysqli_query($con, $query);
    $no_of_rows = mysqli_num_rows($res);
    $c = 0;
?>

<html>
    <head>
        <title>Student Results</title>

        <style>
            div{
                font-size: 40px;
                font-weight: bold;;
                width: 100%;
                height: 100px;
                background-color:cornsilk;
                text-align: center;
            }

            table{
                position: relative;
                top: 125px;
                font-size: 40px;
            }
        </style>
    </head>

    <body>
        <div>
```

```
            Student Results
        </div>

        <table border = "1" cellspacing = "0px" cellpadding = "5px"
align = "center">
            <tr>
                <th>Student ID</th>
                <th>Course ID</th>
                <th>Exam Type</th>
                <th>Marks</th>
            </tr>

            <?php while ($c < $no_of_rows){
                $row = mysqli_fetch_assoc($res);
                $c = $c + 1; ?>
                <tr>
                    <td> <?php echo $row['StudentID'] ?> </td>
                    <td> <?php echo $row['CourseID'] ?> </td>
                    <td> <?php echo $row['Exam'] ?> </td>
                    <td> <?php echo $row['Marks'] ?> </td>
                </tr>
            <?php } ?>
        </table>
    </body>
</html>
```

View Notes Web Page

```
<?php
    include_once 'DbConnection.php';
    $query = "Select * from Notes";
    $res = mysqli_query($con, $query);
    $no_of_rows = mysqli_num_rows($res);

    $c = 1;
?>

<html>
    <head>
        <title>View Notes</title>

        <style>
            table{
                table-layout: fixed;
                position: relative;
                top: 10px;
            }

            td{
                word-wrap: break-word;
            }
        </style>
    </head>

    <body>
        <table border = "1" width = "80%" cellpadding = "5px"
cellspacing = "0px" align = "center">

            <tr>
                <th width = "20%">Note</th>
                <th width = "80%">Resource</th>
            </tr>
```

```
            <?php while ($c <= $no_of_rows){
                $row = mysqli_fetch_assoc($res); $c = $c + 1;
            ?>

                <tr>
                    <td> <?php echo $row['name']; ?> </td>
                    <td> <a href = "<?php echo $row['destination']
?>" target = "_blank"> Click here to view note </a></td>
                </tr>

            <?php } ?>

        </table>
    </body>
</html>
```