Student Name	:	Shamee Mrimmoaee Ahmed
TP Number	:	TP040804
Intake Code	:	UC2F1704SE
Subject	:	Data Structure
Project Title	:	Library Management System
Date Assigned	:	15 October 2017
Date Completed	:	19 January 2018

Table of Contents

1.0 Introduction

The name of the system is Library Management System. This software has been developed to handle the basic function of a library that is it will record all the book detail, register student information and it keeps track of issue or return book record. Beside this, it will help to order new book to the supplier and record all the supplier and their company detail information. This system is only applicable for admin to do their daily task.

This system will provide a user-friendly environment where the result is faster and more satisfied than before. It will minimize the number of book loss and student can only borrow a book from the library if they are registered. Moreover, it will record the quantity of the book available in the library and how many books issued and return by the student. This software is customizable for any library requirement or future use.

2.0 Objectives

This system is applicable to school or university to maintain their daily task efficiently and effectively. This system will **help to record every transaction in a computerized** system that will store at library database of that organization so there will be no chance of losing the record file. Beside that this system will eliminate the paper uses in the library for their daily work. This system has **a user-friendly** graphical interface that can easily learn by the admin of the library. So, there will be no time needed for training to learn how to use the software. This software has the fastest database that adds and update data instantly as well as it has the search option that can be used to find any record with a second. This **system will save the cost and time** of the organization and work with greater access to provide accurate information.

2.1 Diagram

2.1.1Use case diagram

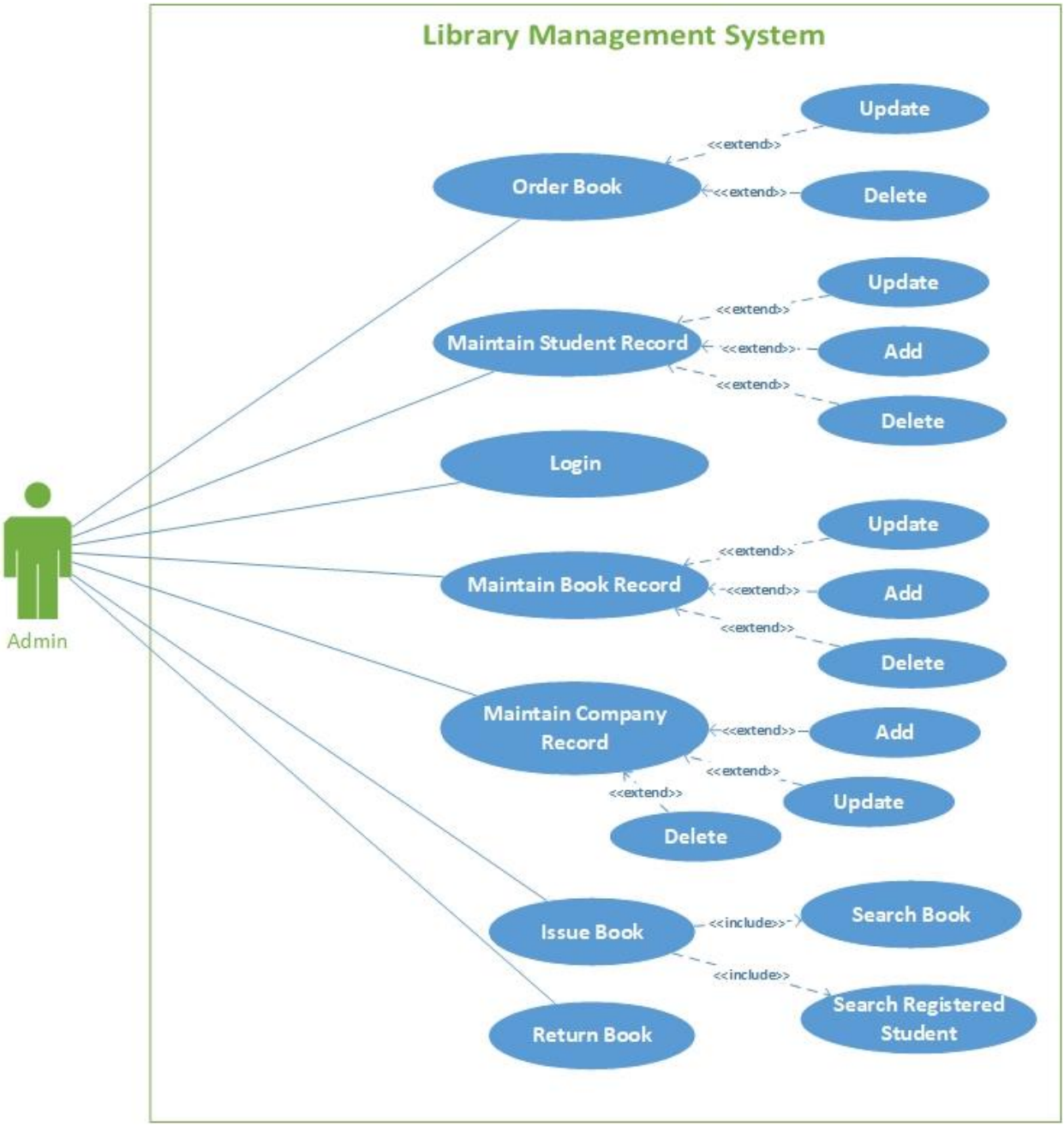

Figure 1: Use Case Diagram

2.1.2 Data dictionary

Table Name	Attribute Name	Description	Data Type	PK or FK
Book_Detail	Book_Name	Name of the book	varchar (50)	PK
	Authur_Name	Authur name of the book	varchar (50)	
	ISBN	ISBN number of the book	varchar (50)	
	Quantity	Quantity of the book	varchar (50)	
	Status	Status of the book	varchar (50)	
Borrow_Book	Student_Name	Name of the student	varchar (50)	PK
	TP_Number	TP number of the student	varchar (50)	
	Book_Name	Name of the book	varchar (50)	
	ISBN	ISBN number of the book	varchar (50)	
	Status	Status of the book	varchar (50)	
	Issue_Date	Date of issue	varchar (50)	
	Return_Date	Date of return	varchar (50)	
Company_Detail	Company_Name	Name of the company	varchar (50)	PK
	Company_Information	Information of the company	varchar (50)	
	Person_Contact	Contact number of that company	varchar (50)	
	Book_Name	Name of the book delivered	varchar (50)	
Login	Username	Username of admin	varchar (10)	
	Password	Password of admin	varchar (10)	
Order_Detail	OrderID	OrderID of the new book	varchar (10)	PK

	Book_Name	Name of the book	varchar (50)	
	Authur_Name	Authur name of the book	varchar (50)	
	Quantity	Quantity of the book	varchar (50)	
	Company_Name	Name of the company	varchar (50)	
	Order_Date	Date of book order	varchar (50)	
	Status	Status of the order	varchar (50)	
Student_Information	Student_Name	Name of the student	varchar (50)	PK
	TP_Number	TP number of the student	varchar (50)	
	Semester	Present semester of the student	varchar (50)	
	Intake_Code	Intake code of the student	varchar (50)	

Figure 2: Data Dictionary

2.1.3 Sequence diagram

Login

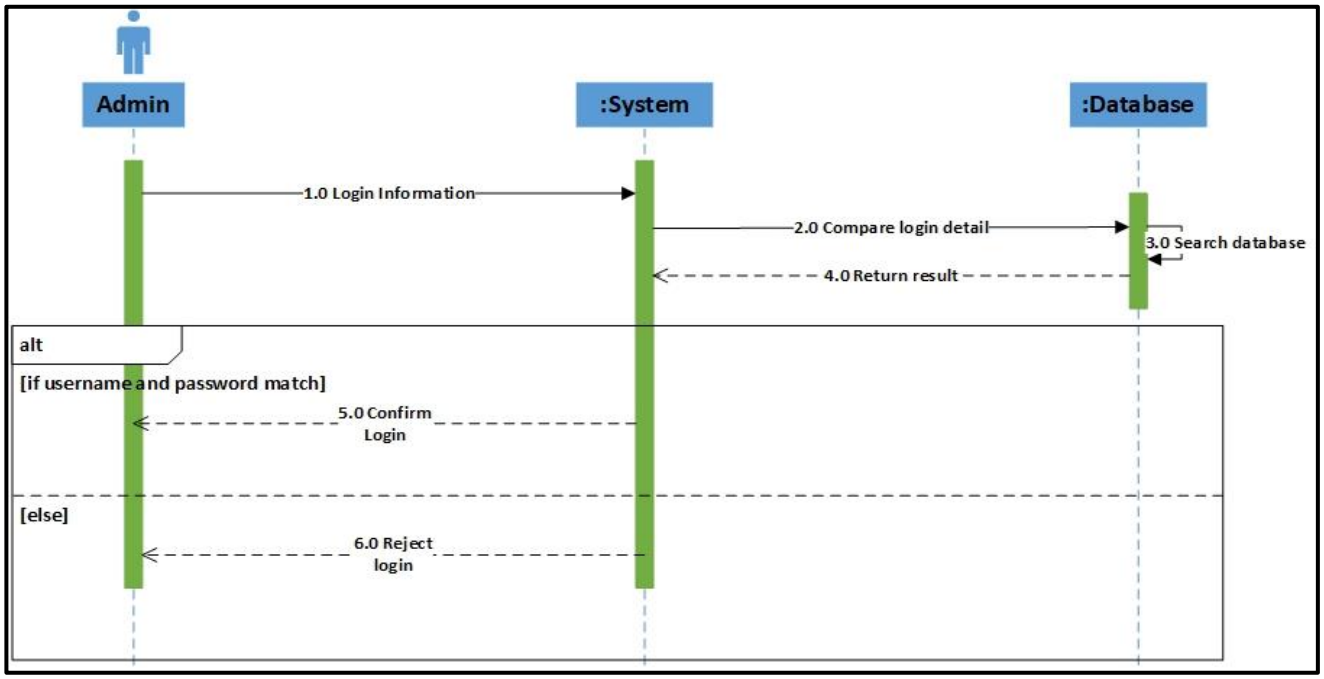

Figure 3: Login

Order Book

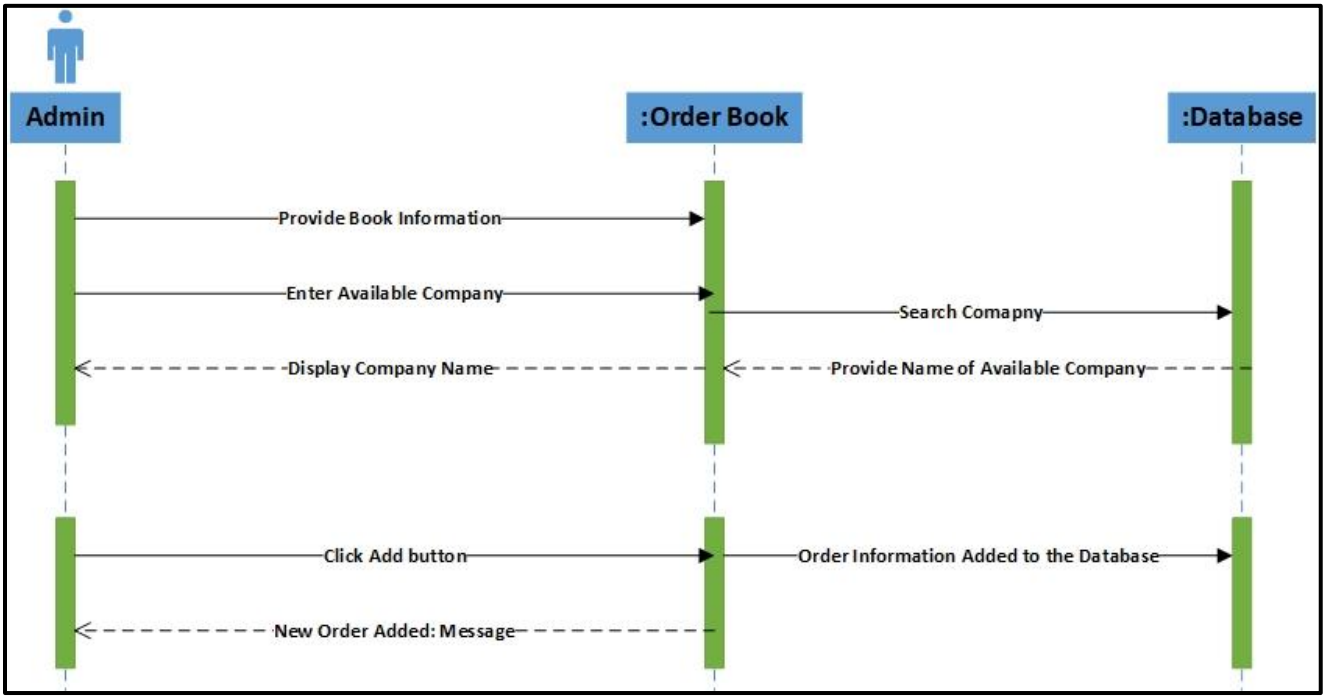

Figure 4: Order Book

Maintain Student Record

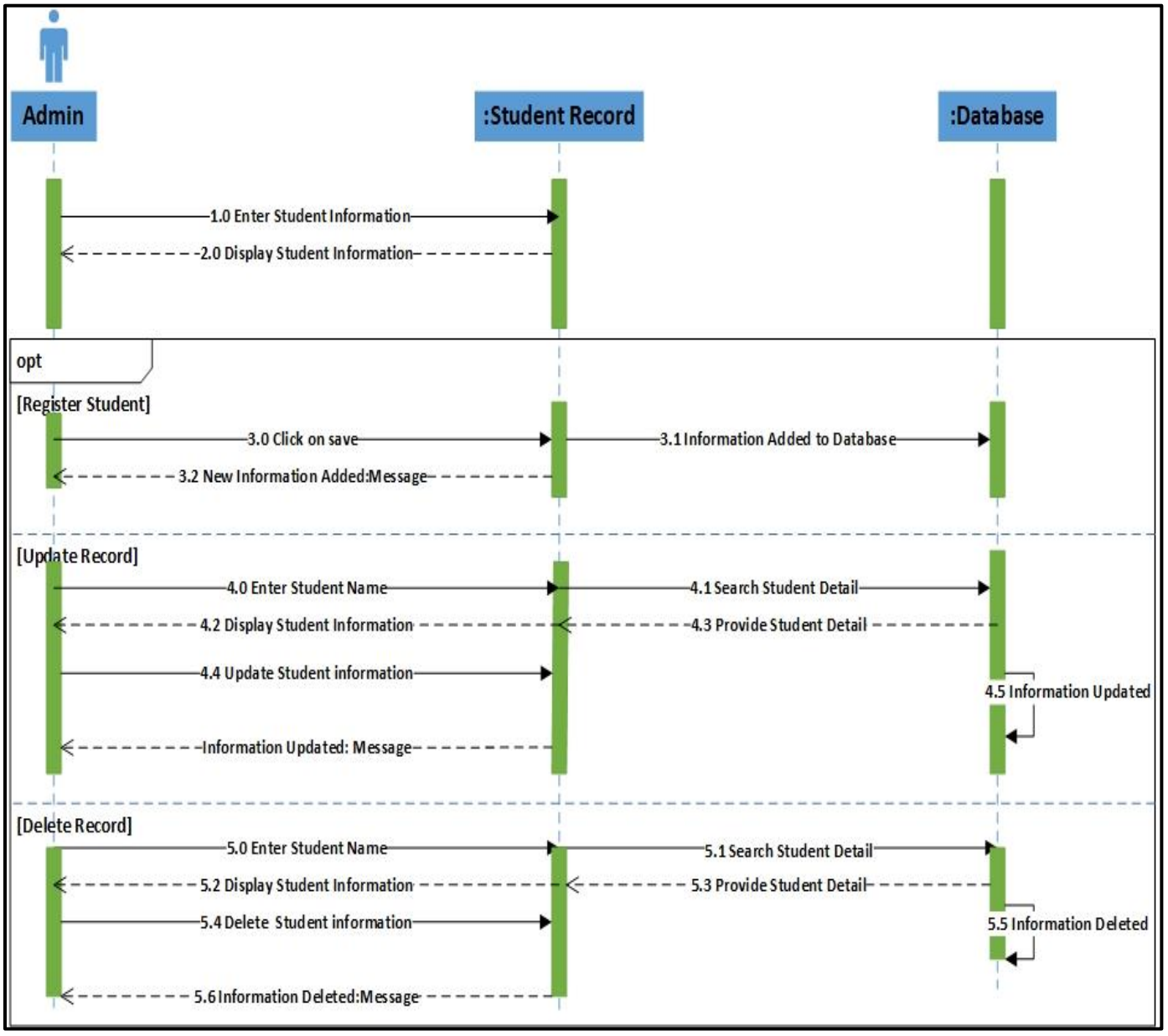

Figure 5: Student Record

Maintain Book record

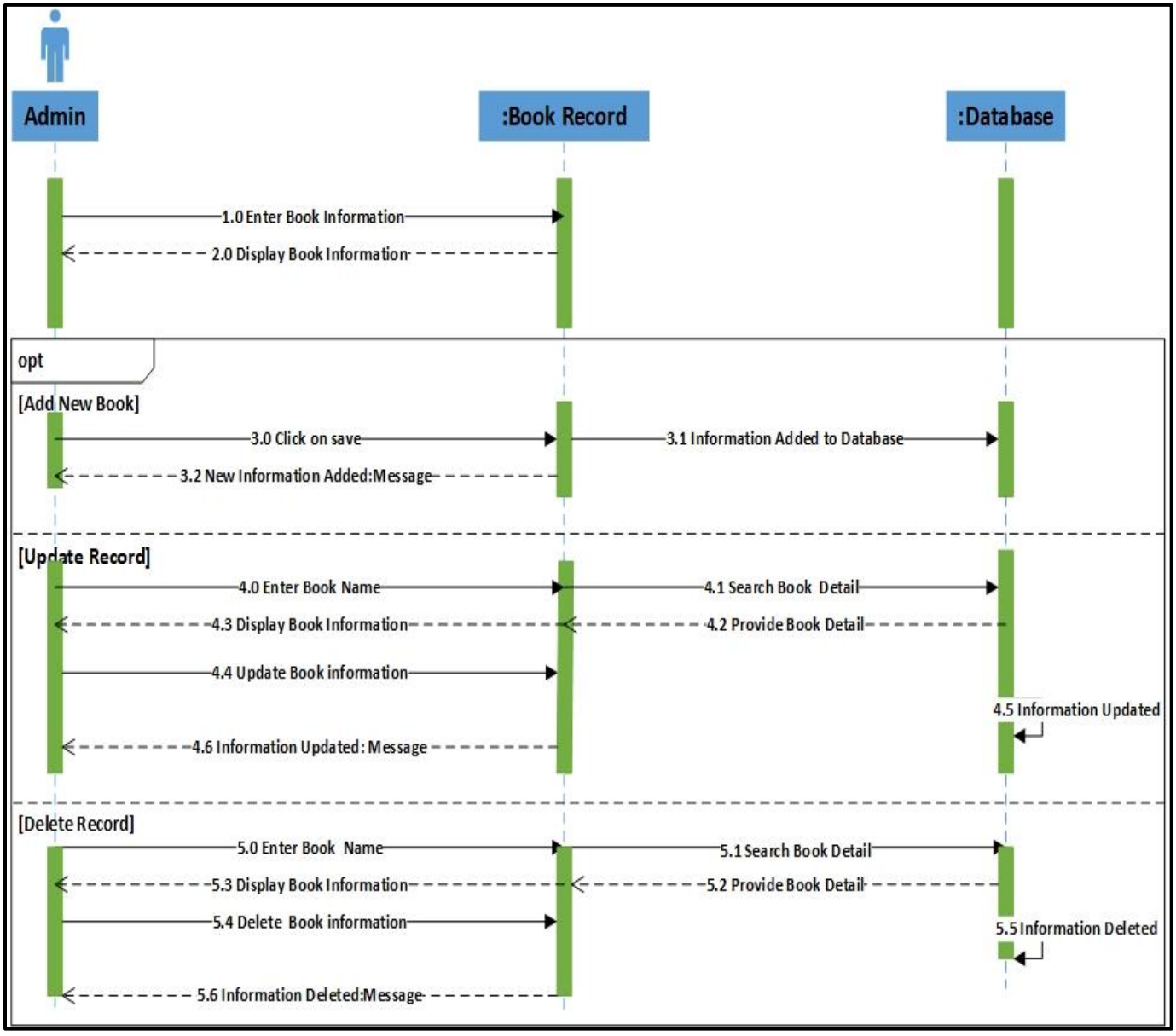

Figure 6: Book Record

Maintain Company Record

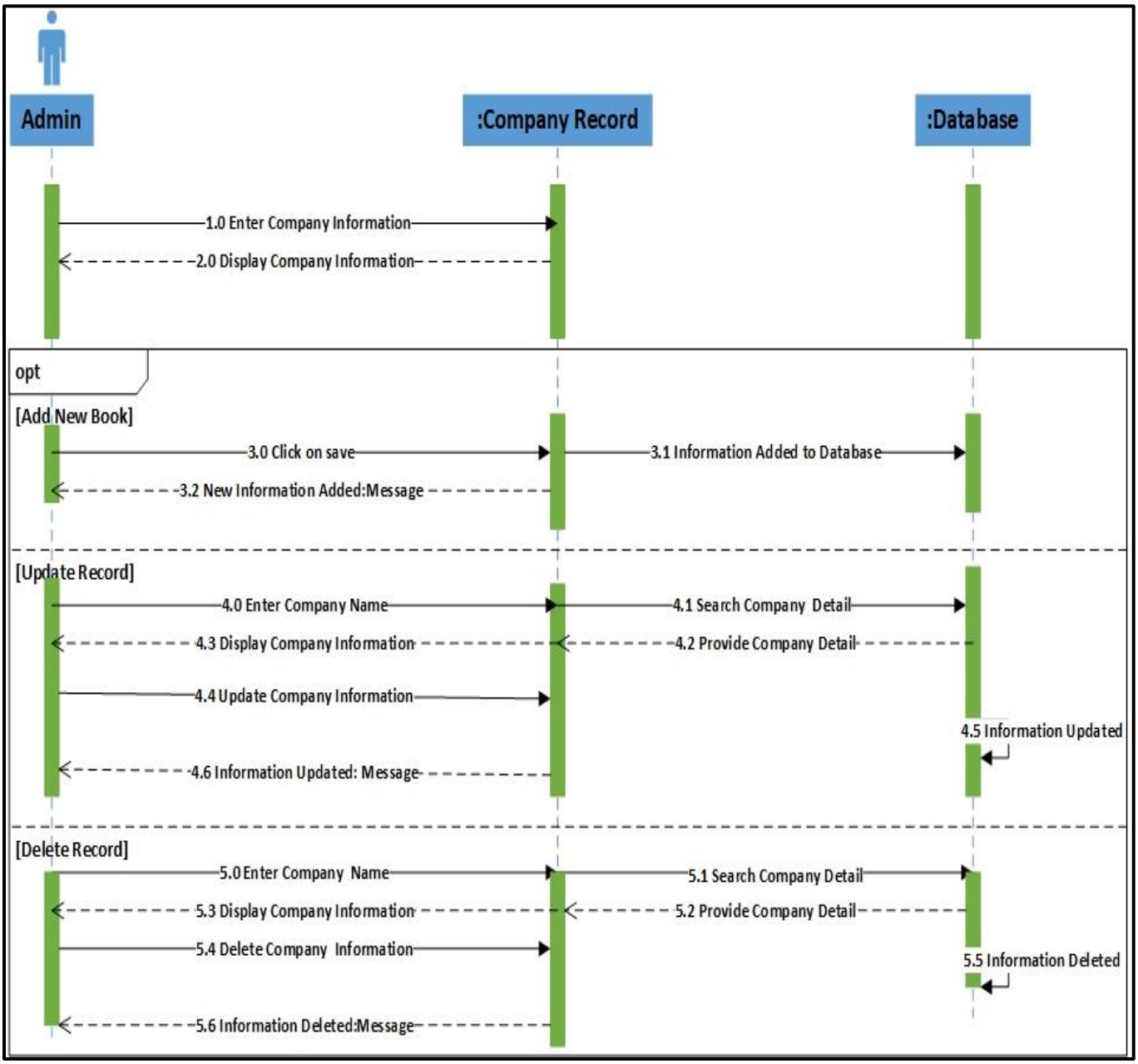

Figure 7: Company Record

Issue Book

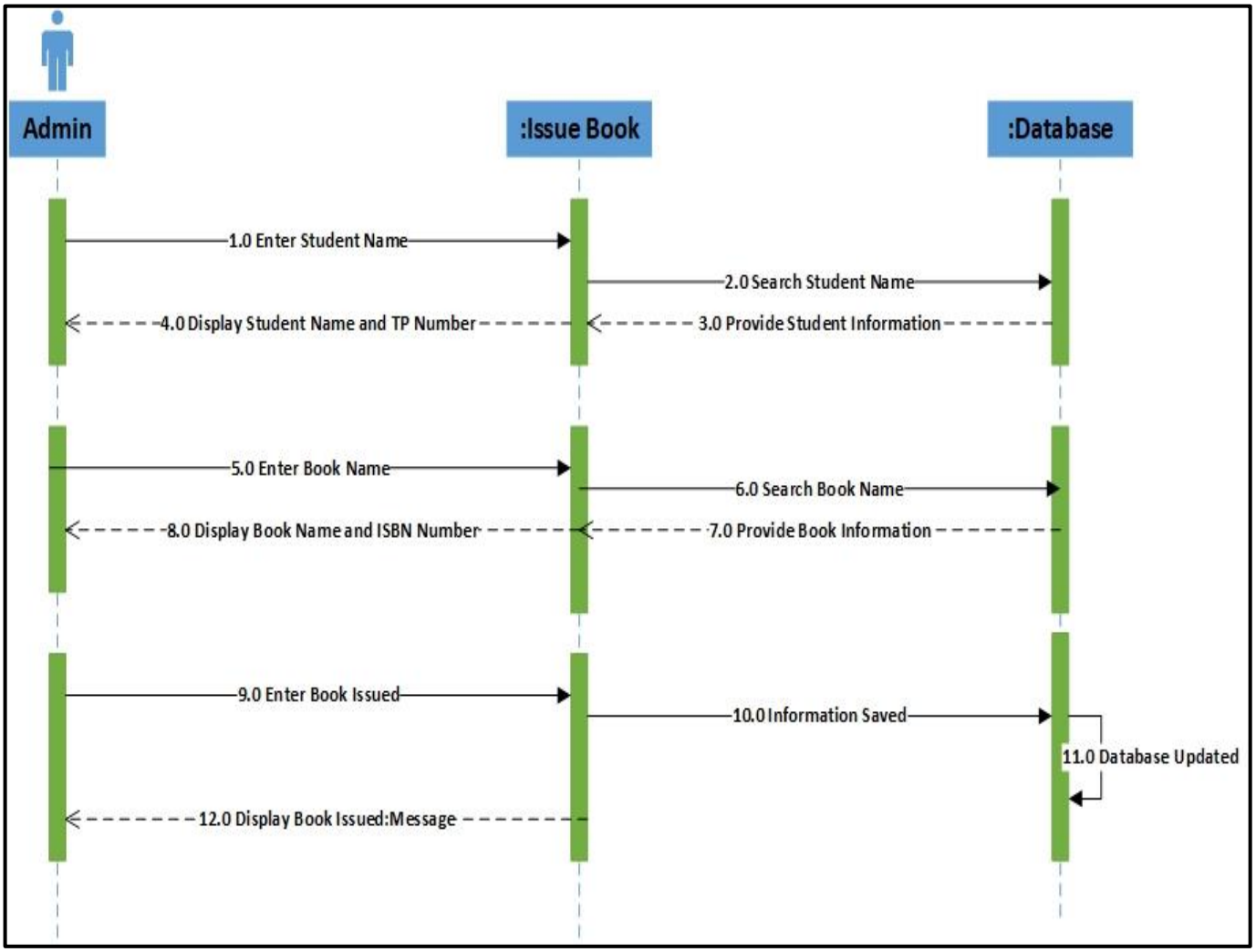

Figure 8: Issue Book

Return Book

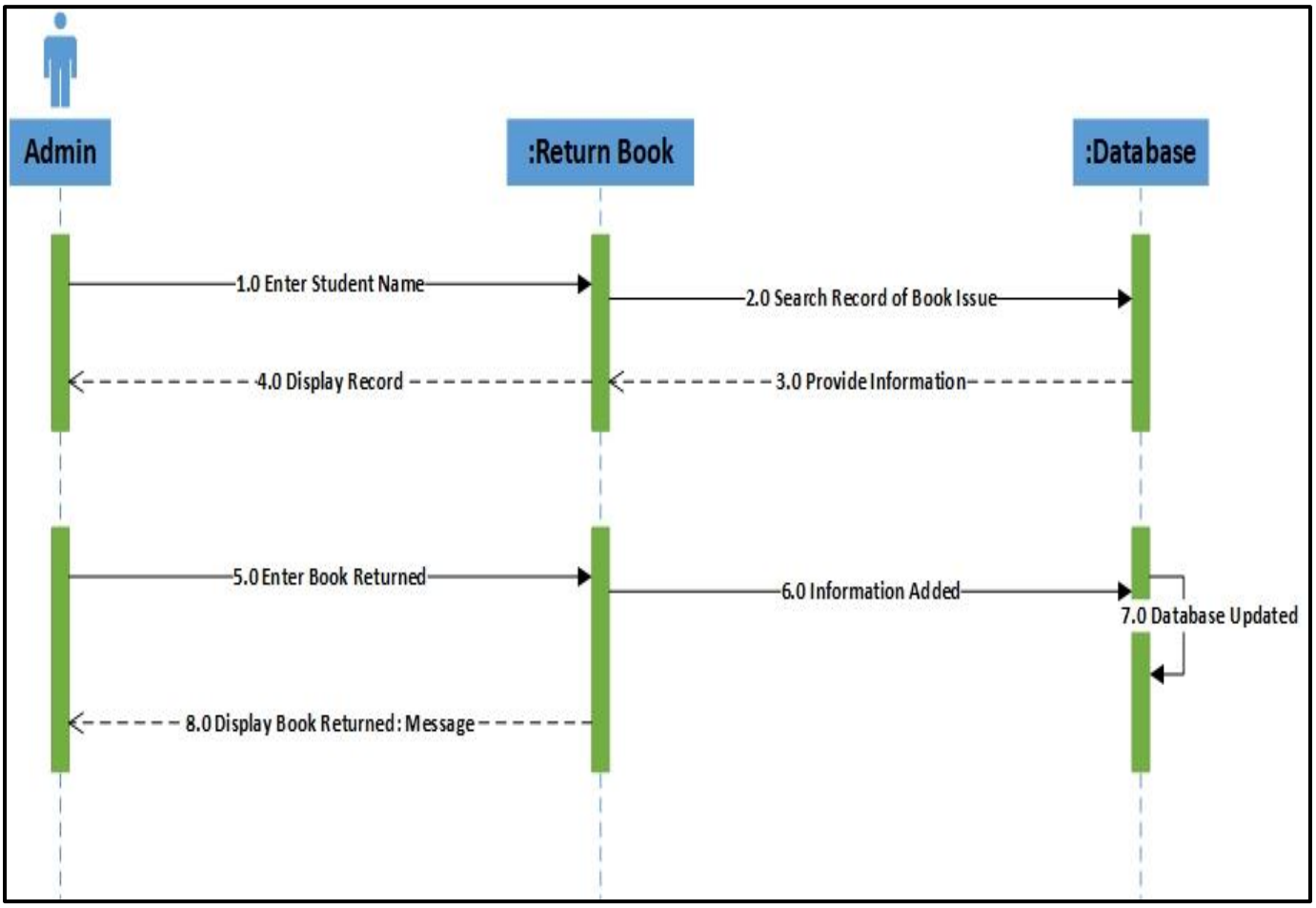

Figure 9: Return Book

2.1.4 Activity diagram

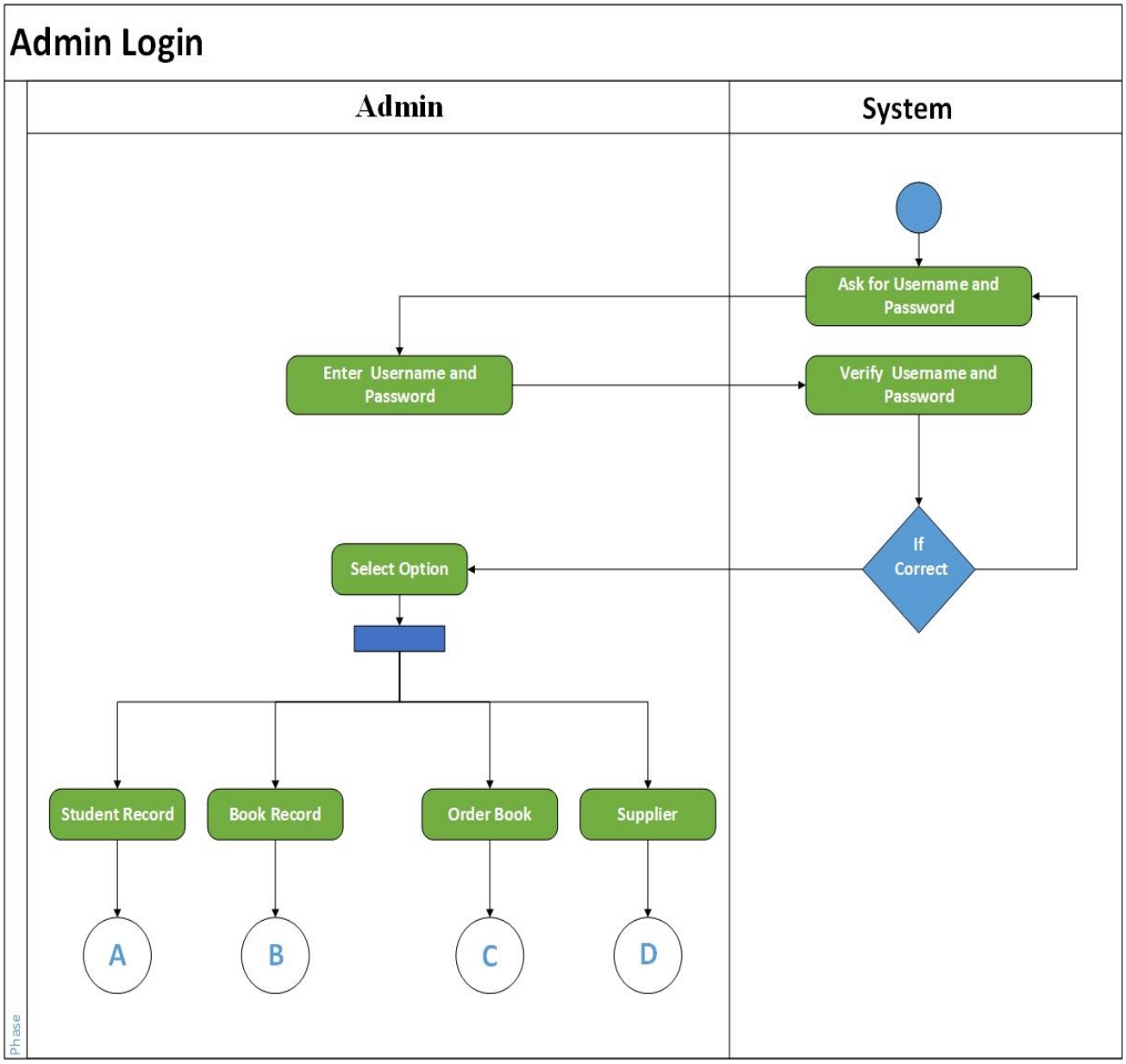

Figure 10: Admin Login

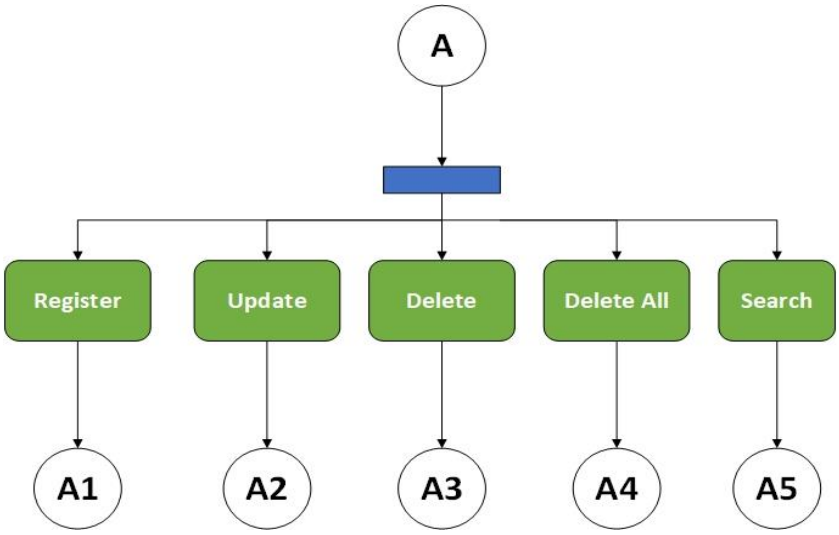

Figure 11: Student Record

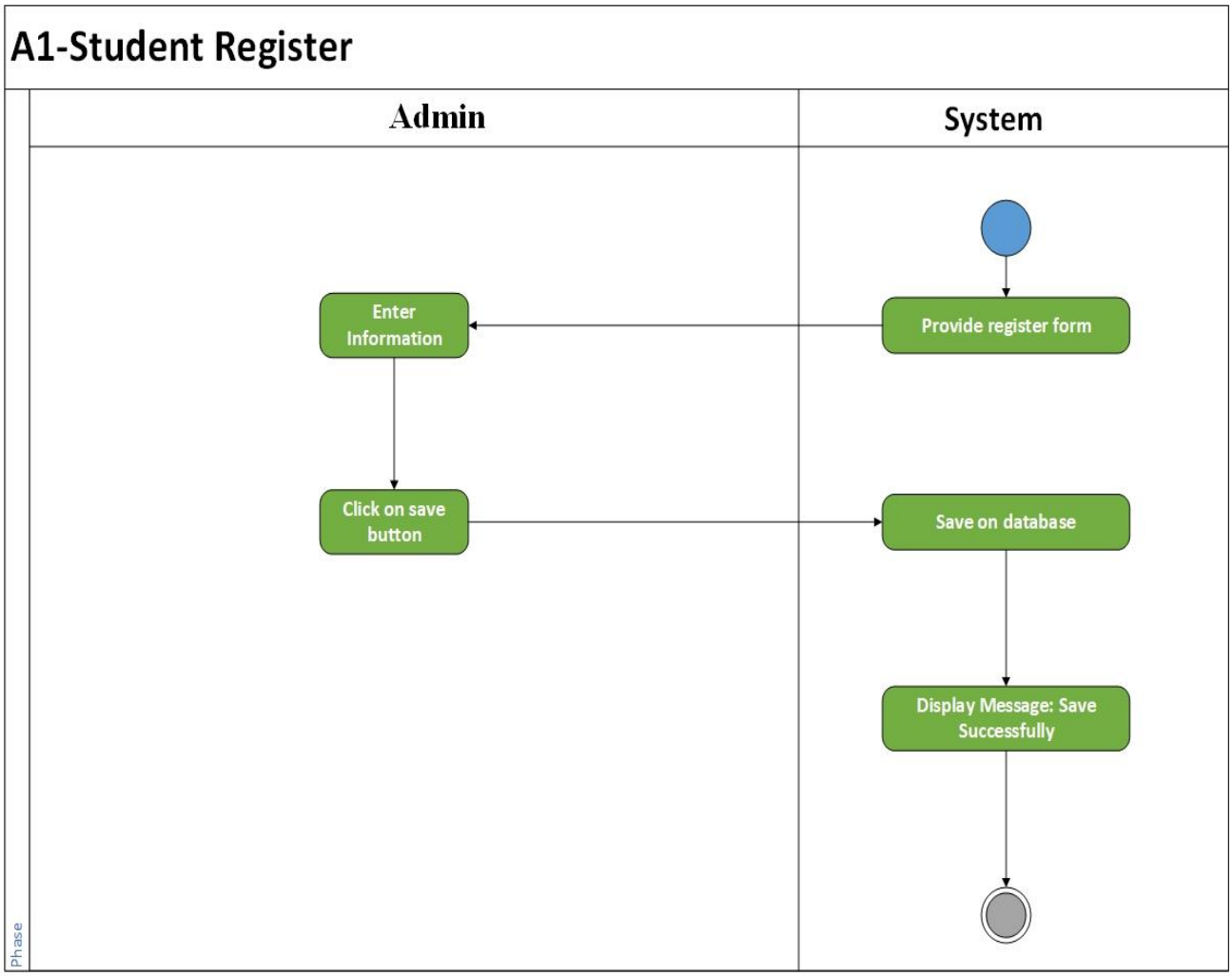

Figure 12: Student Register

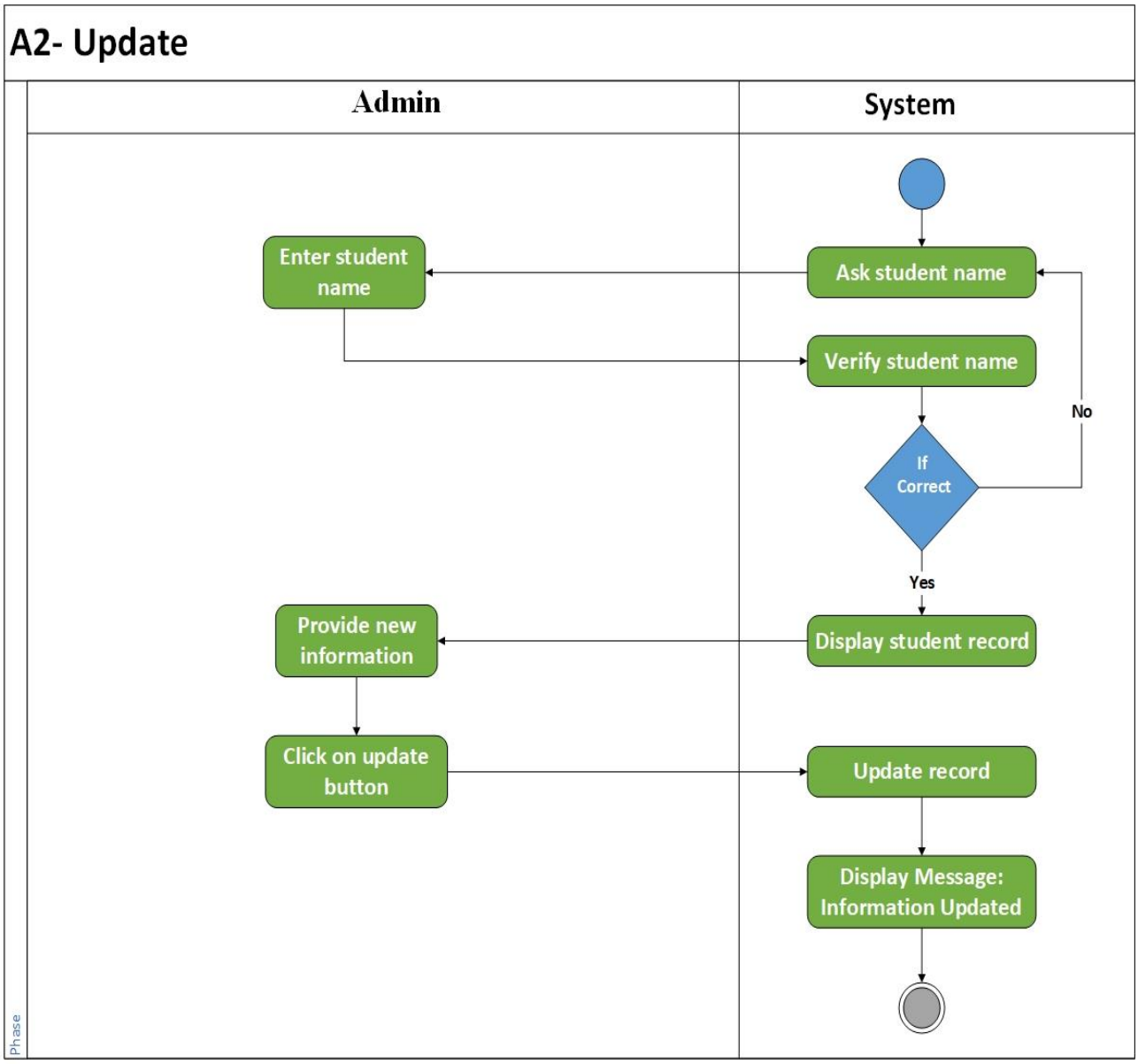

Figure 13: Update

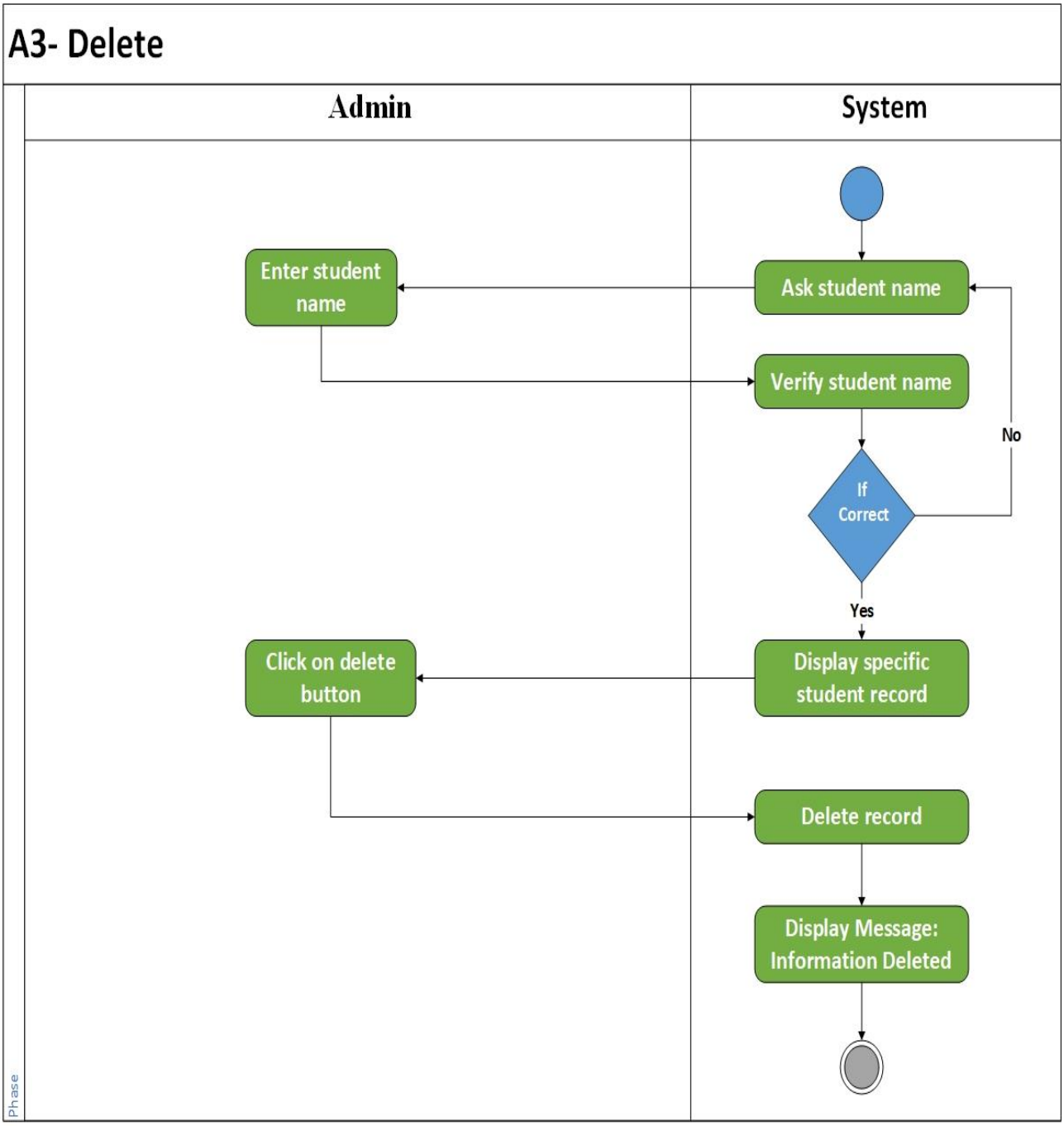

Figure 14: Delete

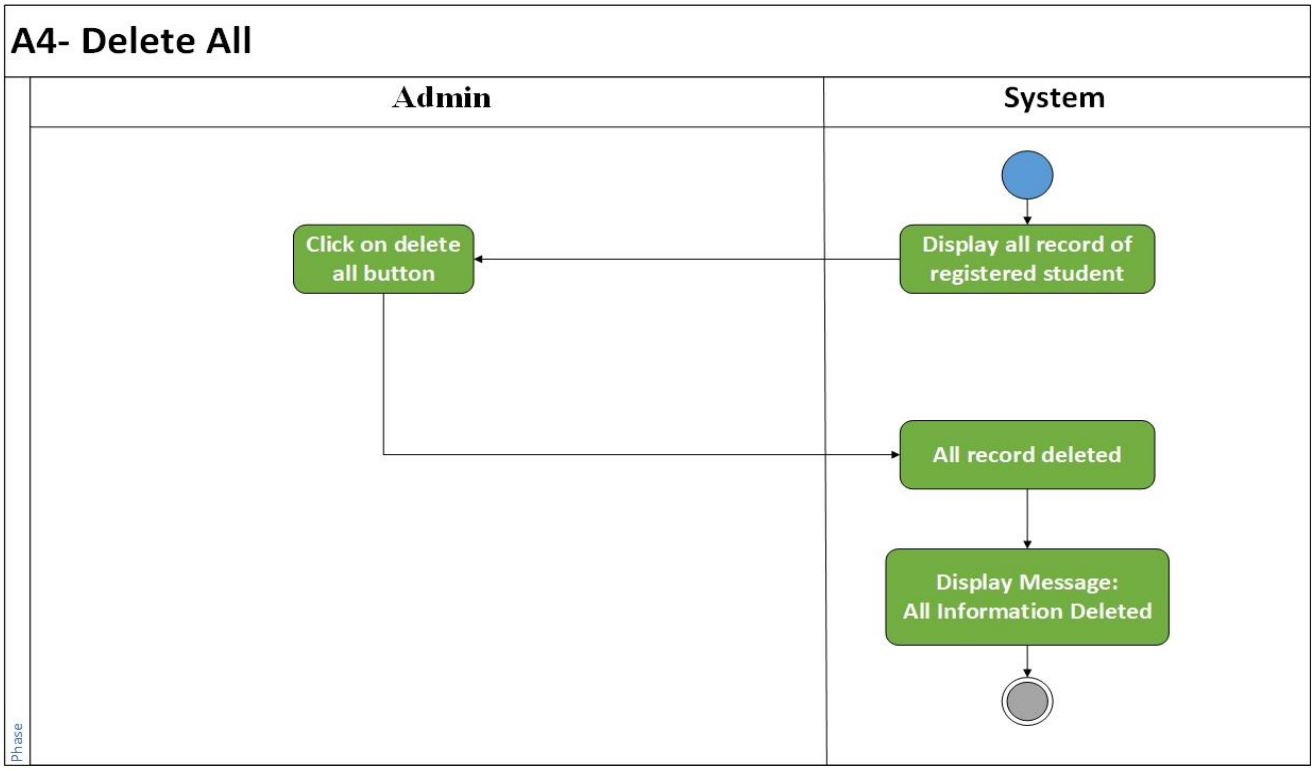

Figure 15: Delete All

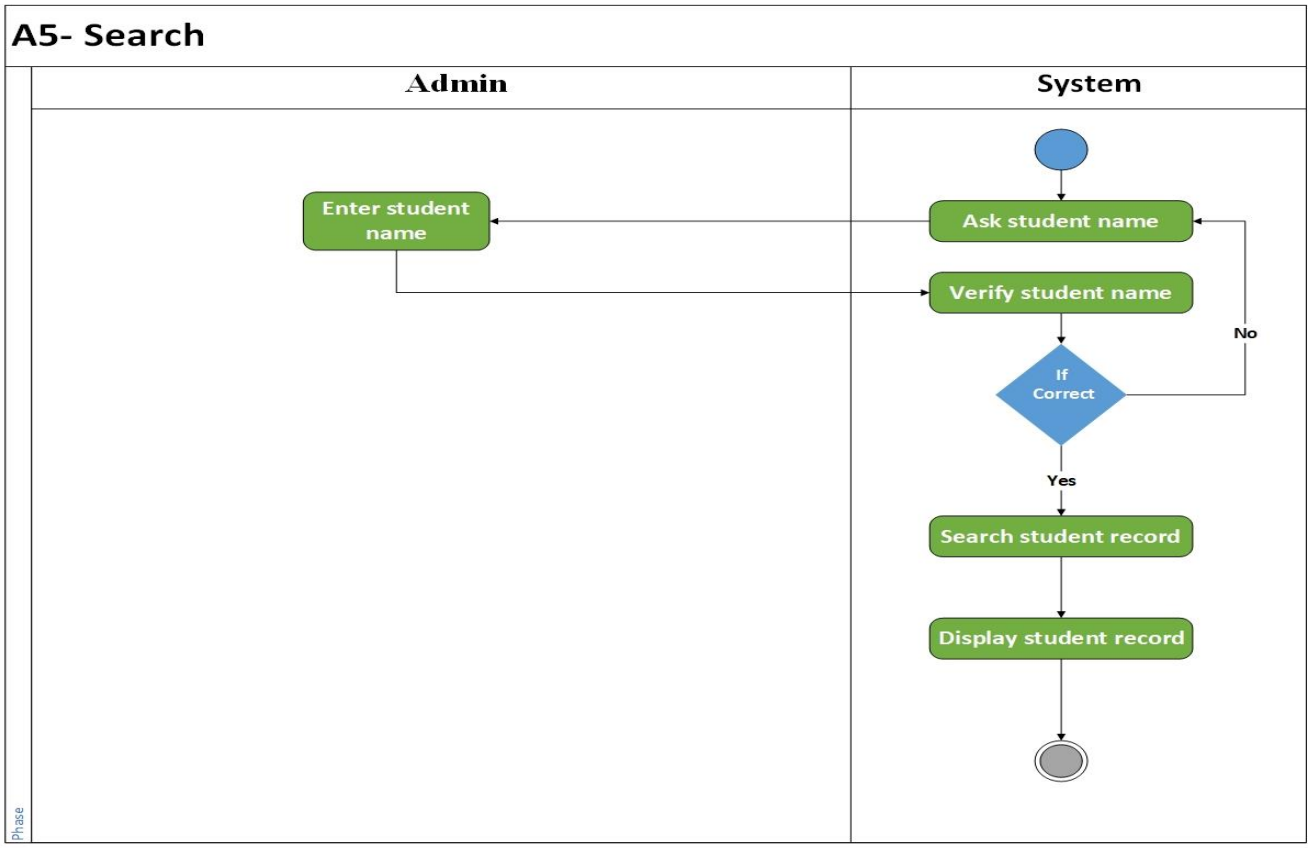

Figure 16: Search

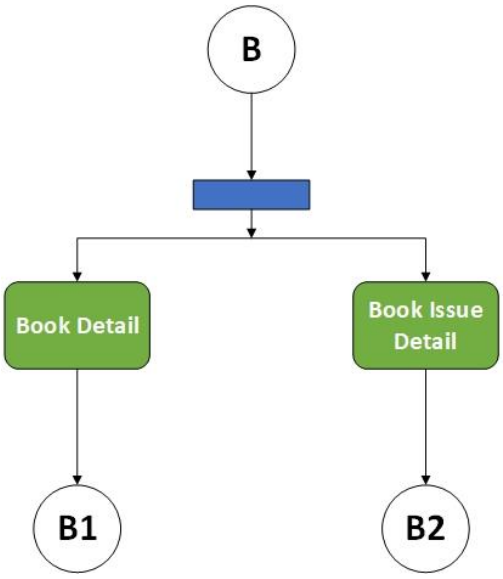

Figure 17: Book

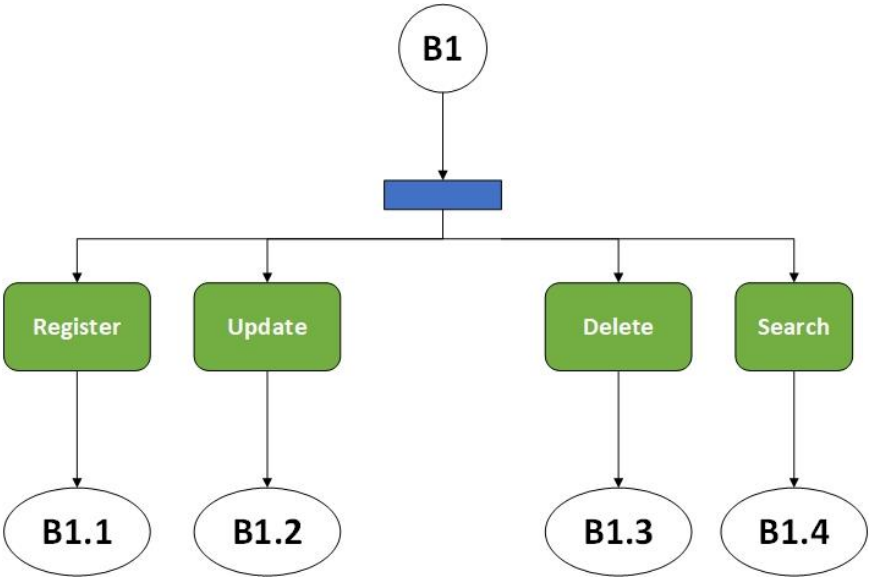

Figure 18: Book Detail

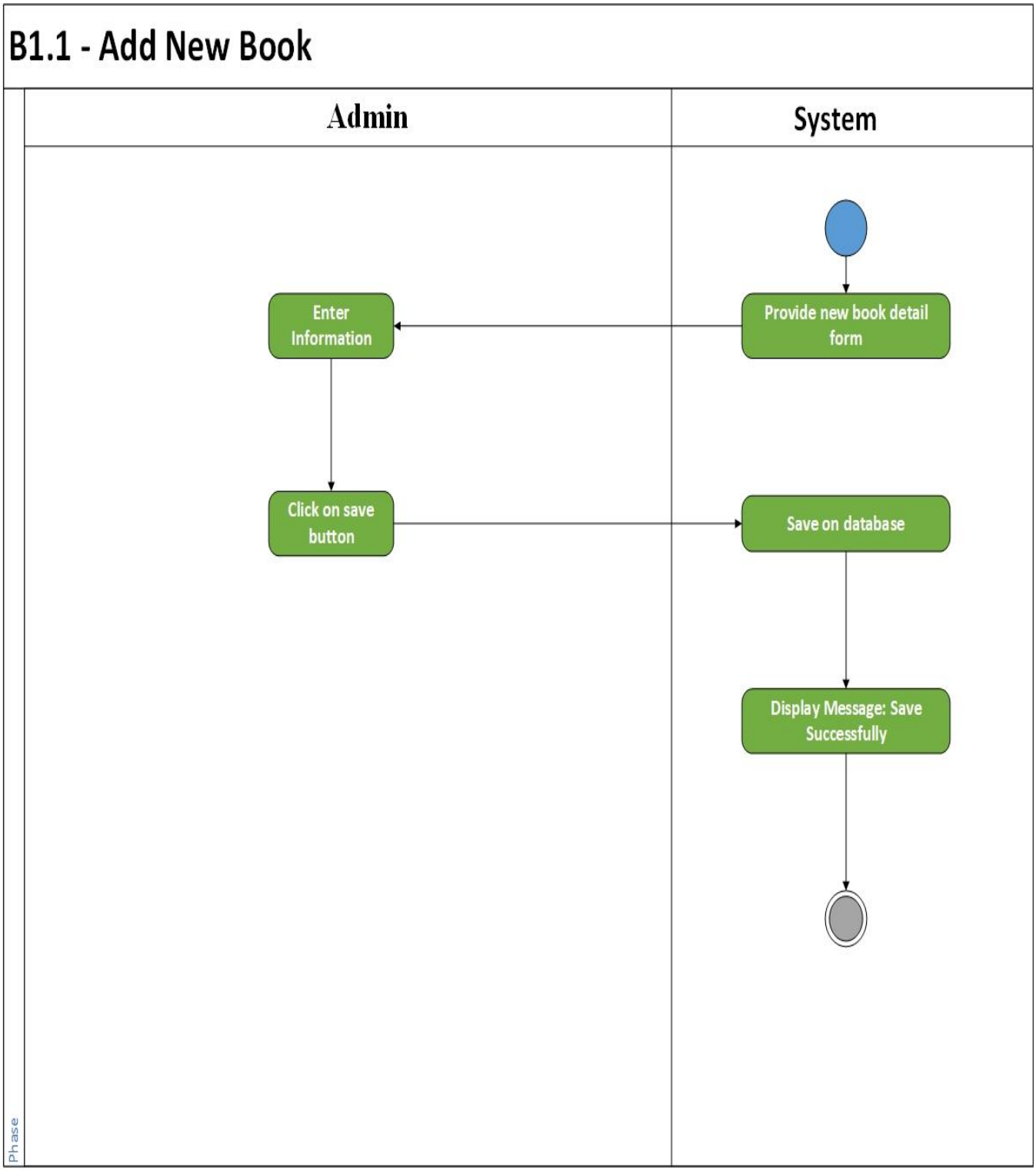

Figure 19: Add New Book

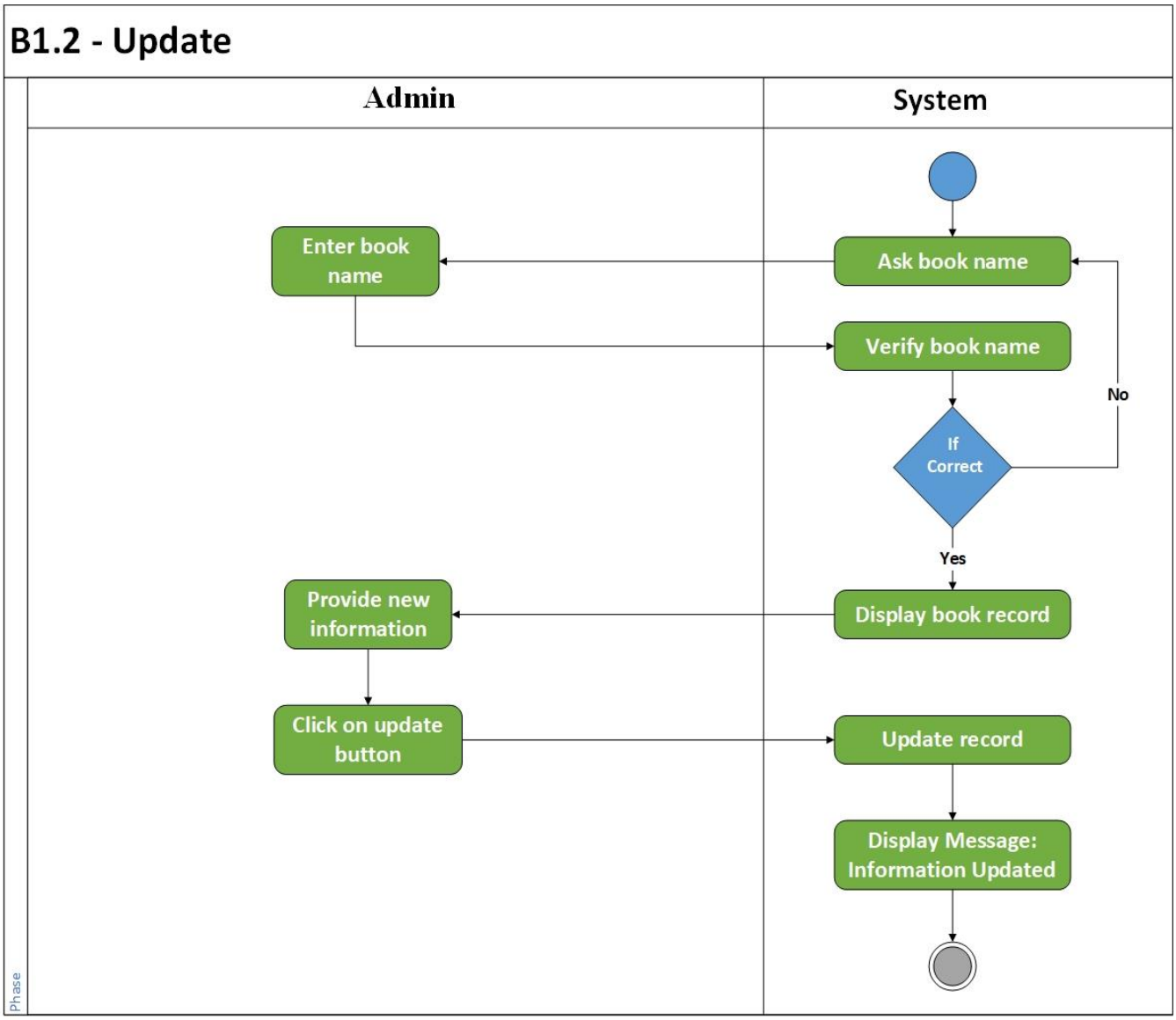

Figure 20: Update

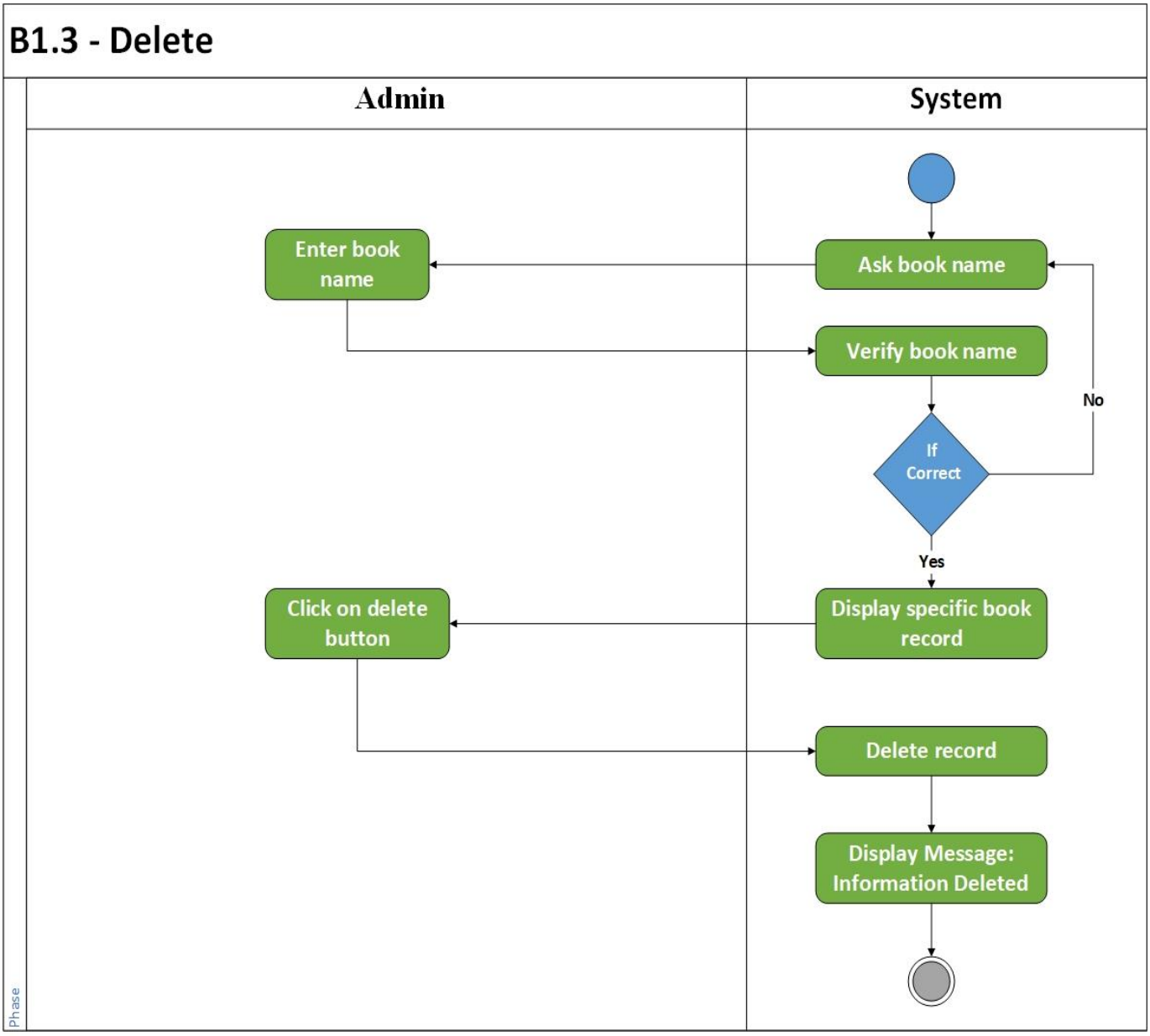

Figure 21: Delete

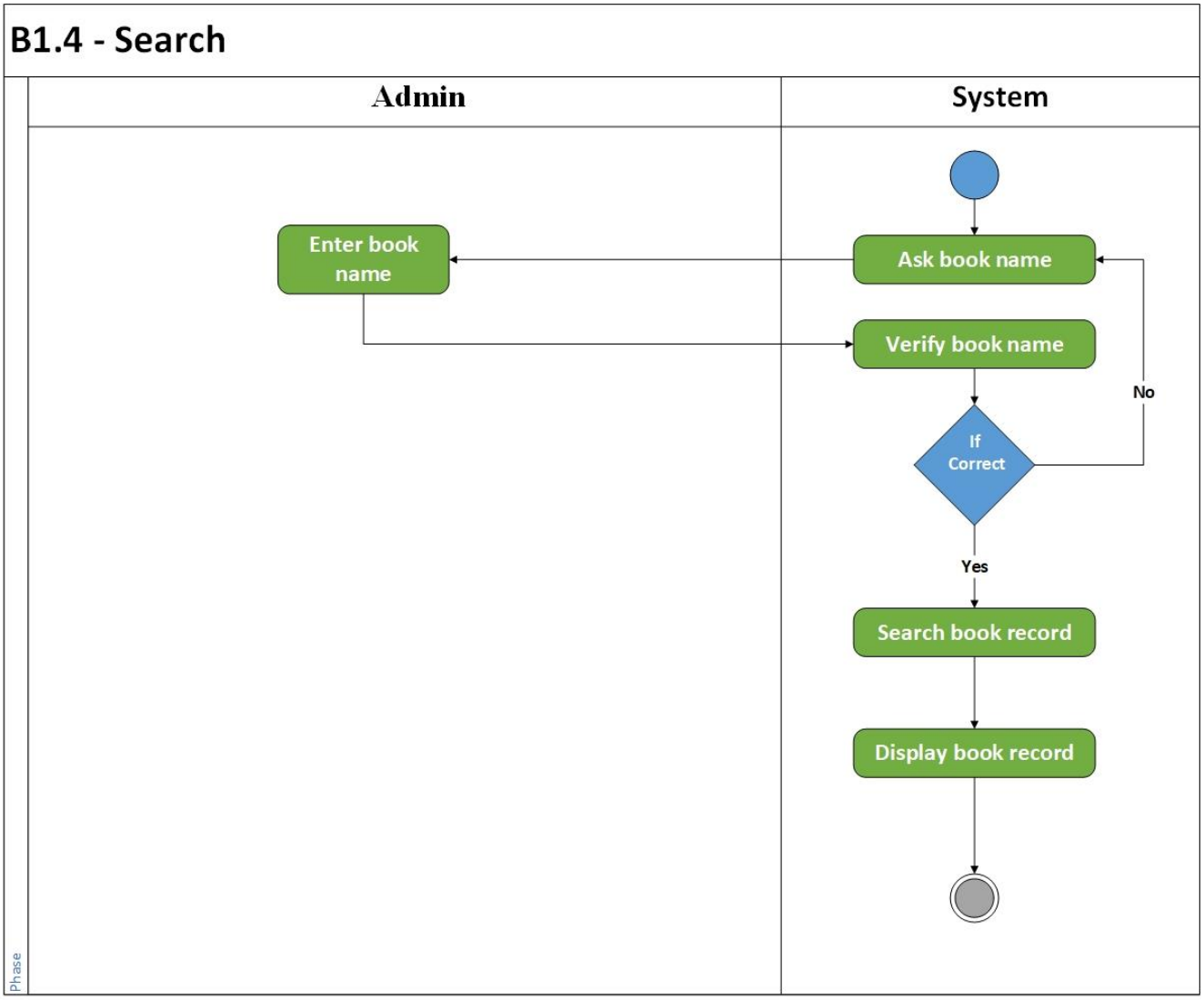

Figure 22: Search

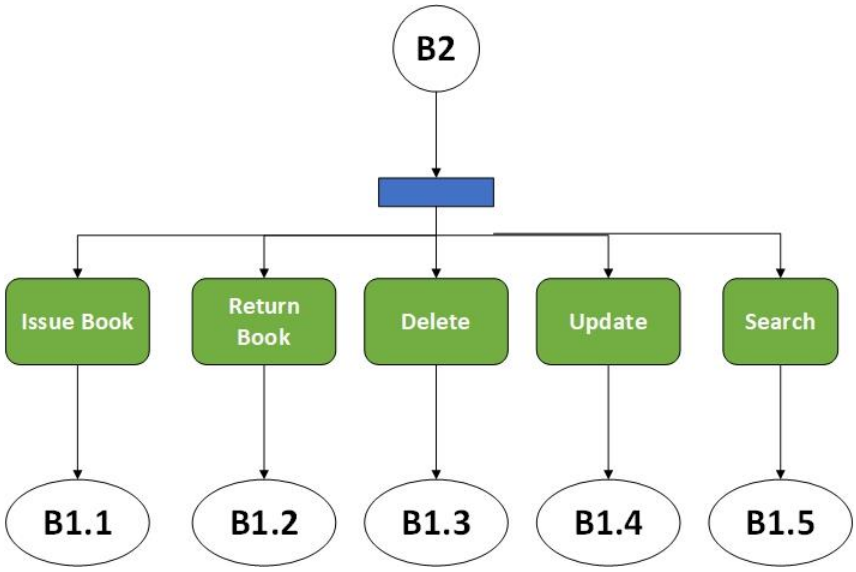

Figure 23: Book Issue Detail

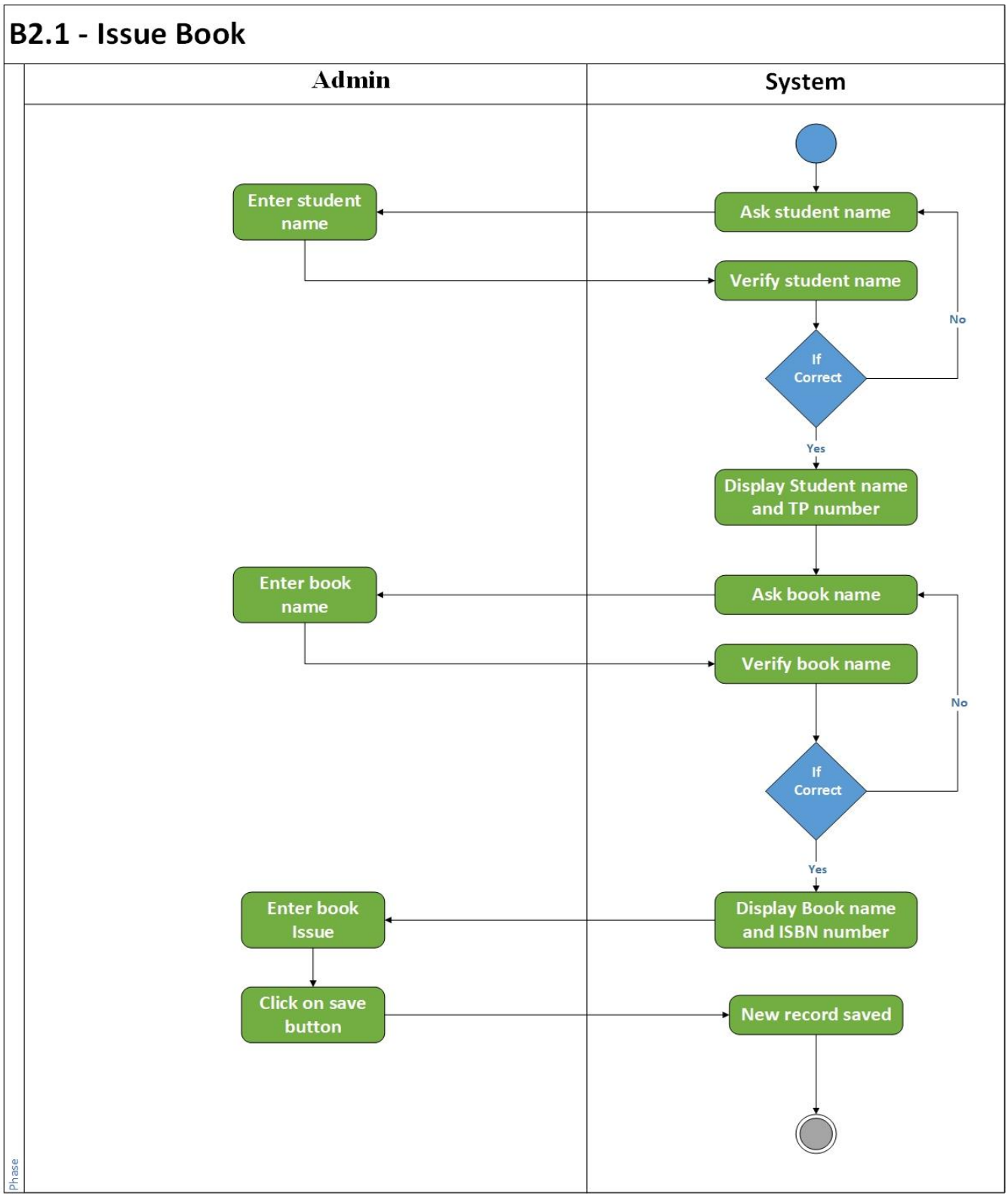

Figure 24: Issue Book

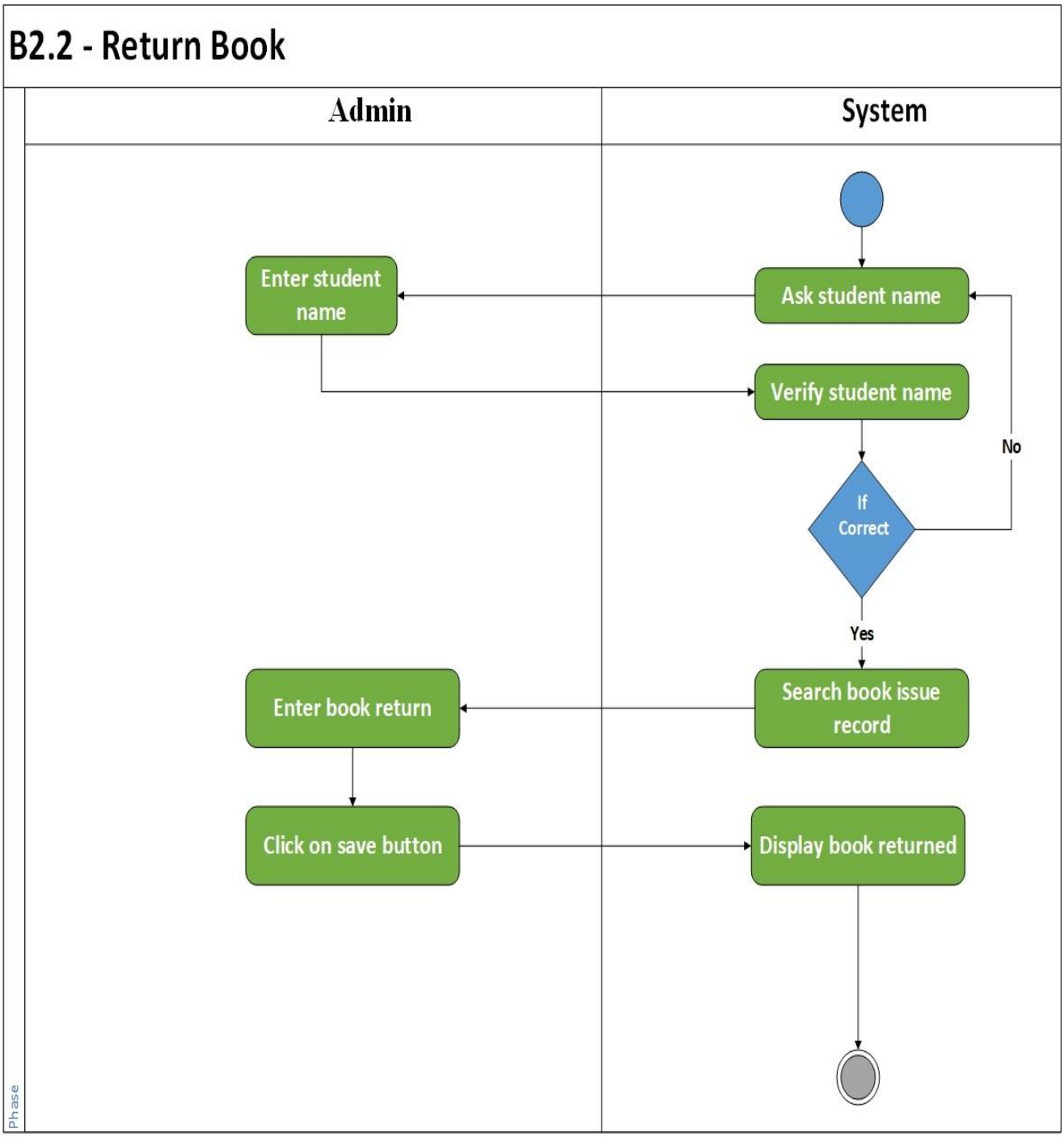

Figure 25: Return Book

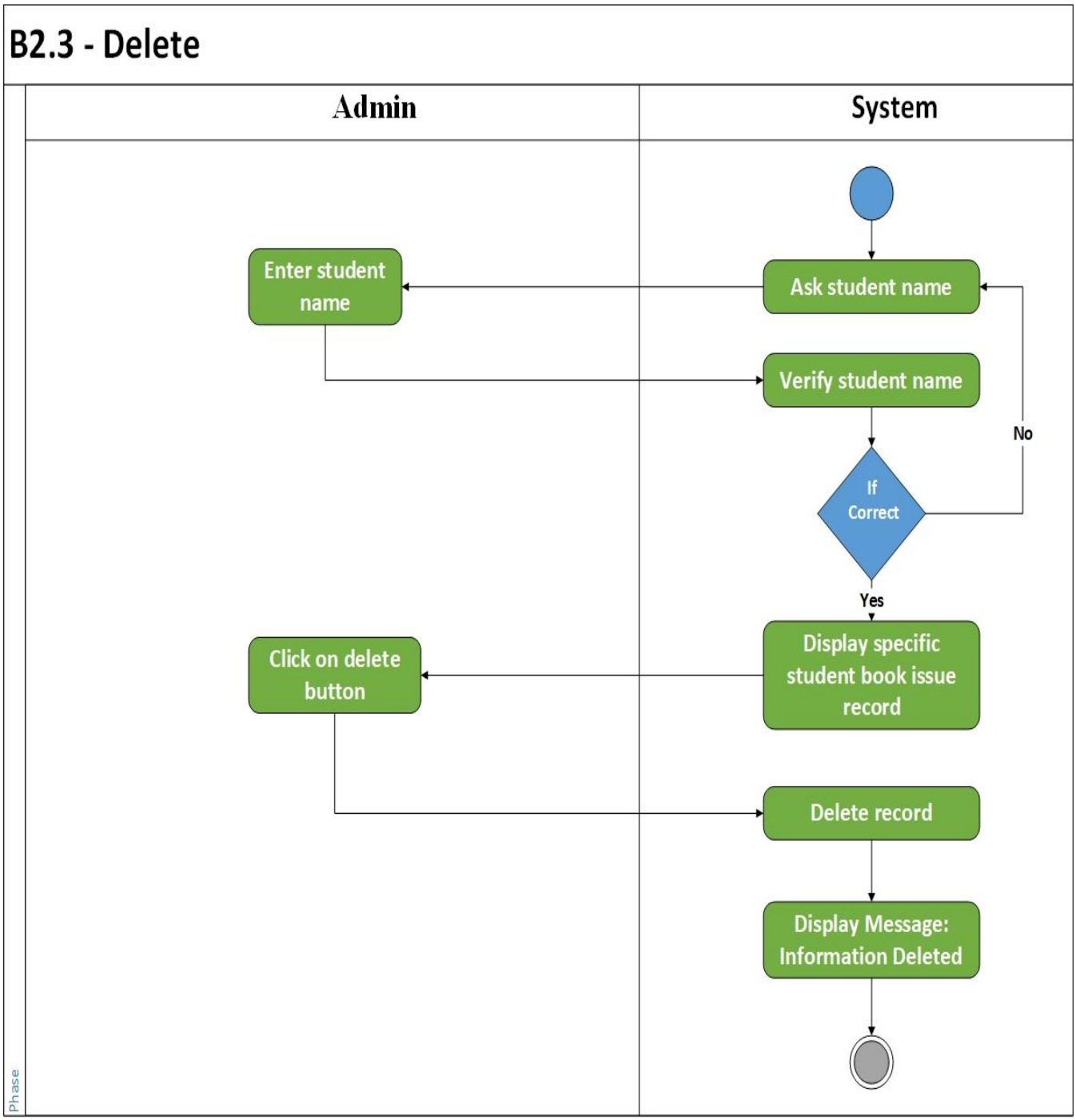

Figure 26: Delete

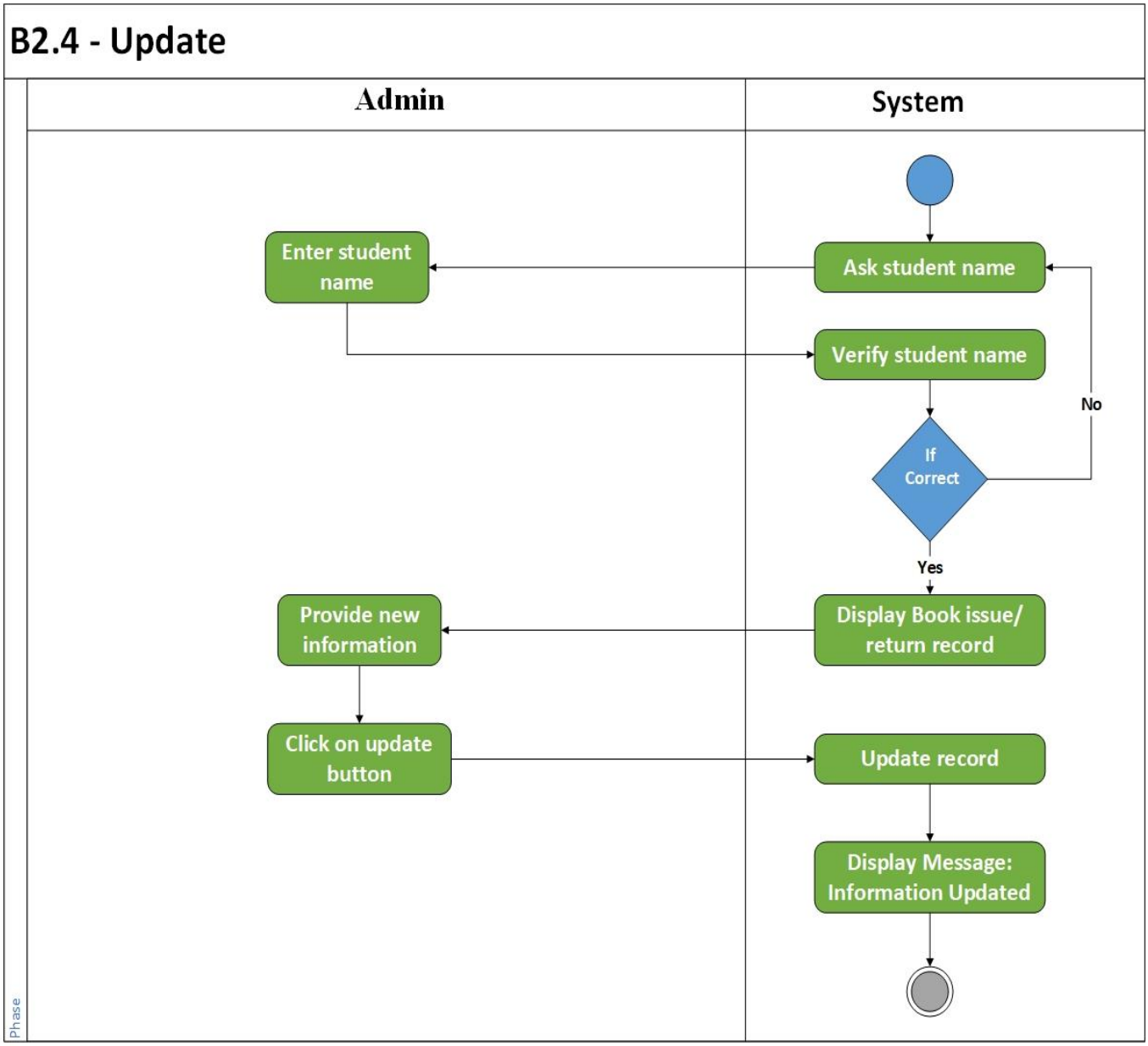

Figure 27: Update

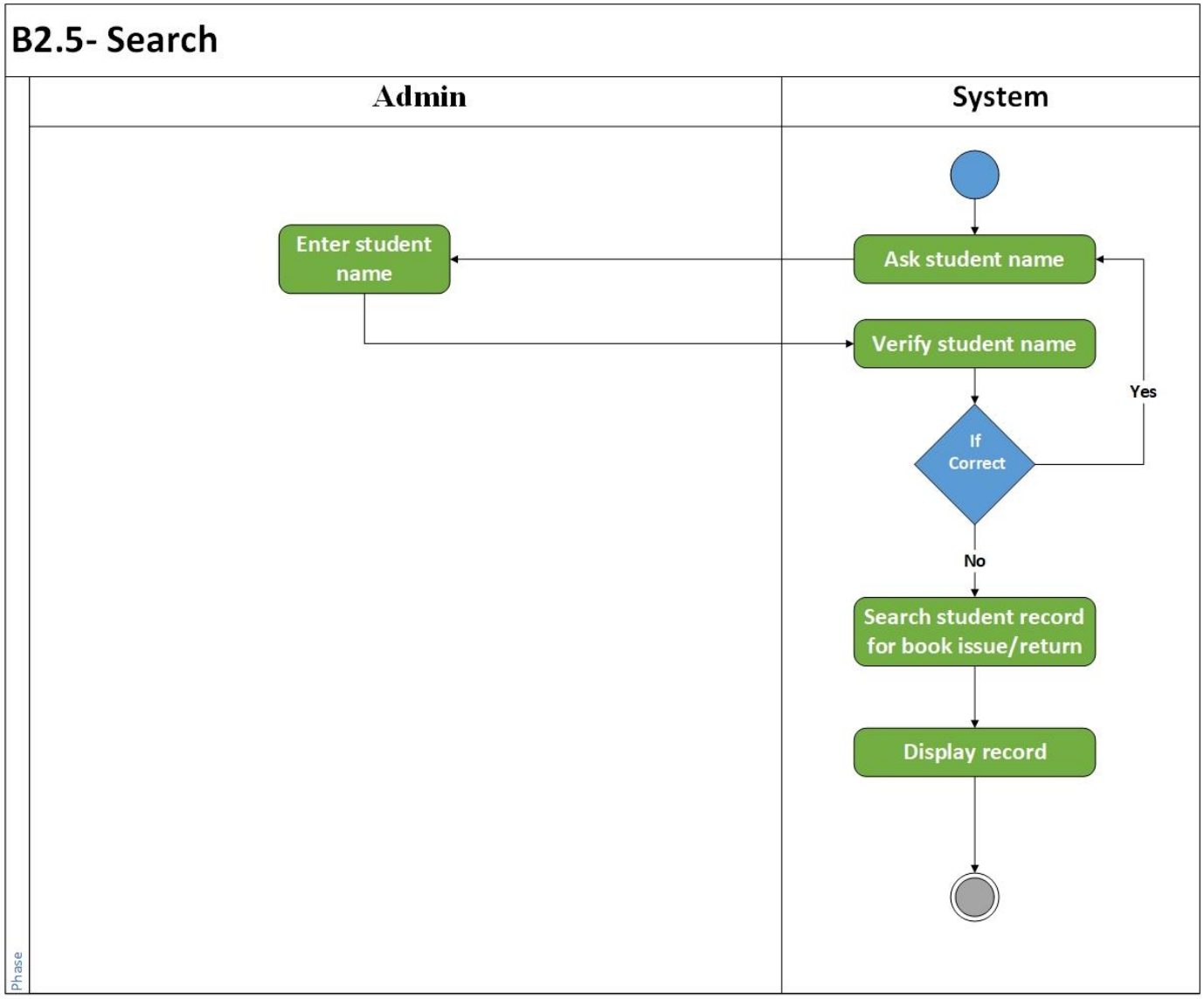

Figure 28: Search

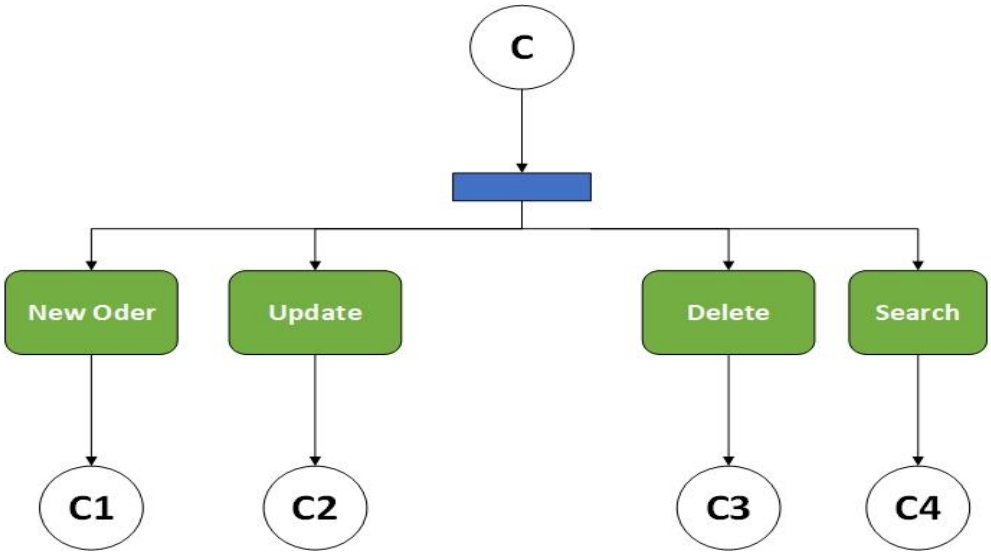

Figure 29: Order Book

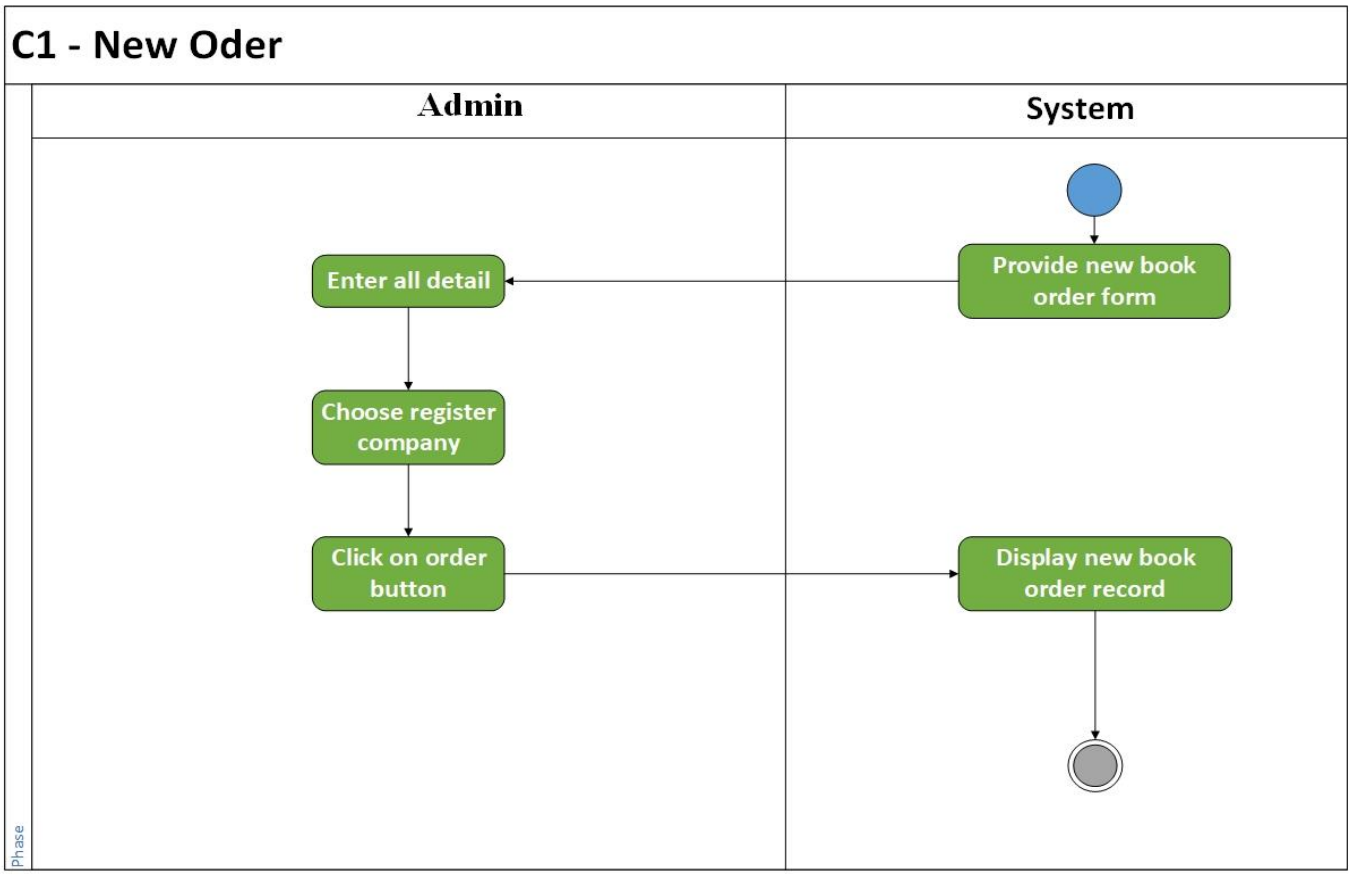

Figure 30: New Order

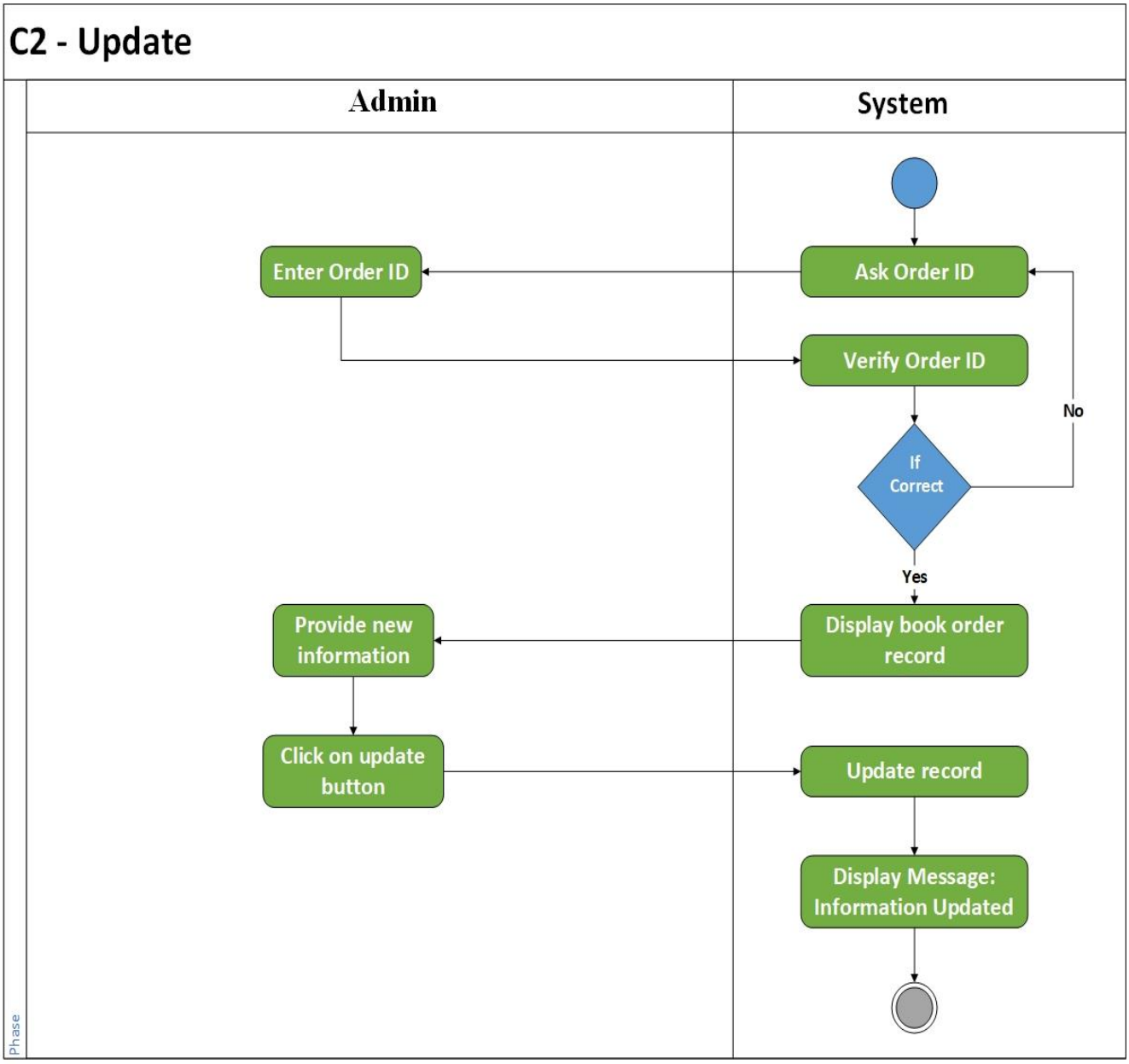

Figure 31: Update

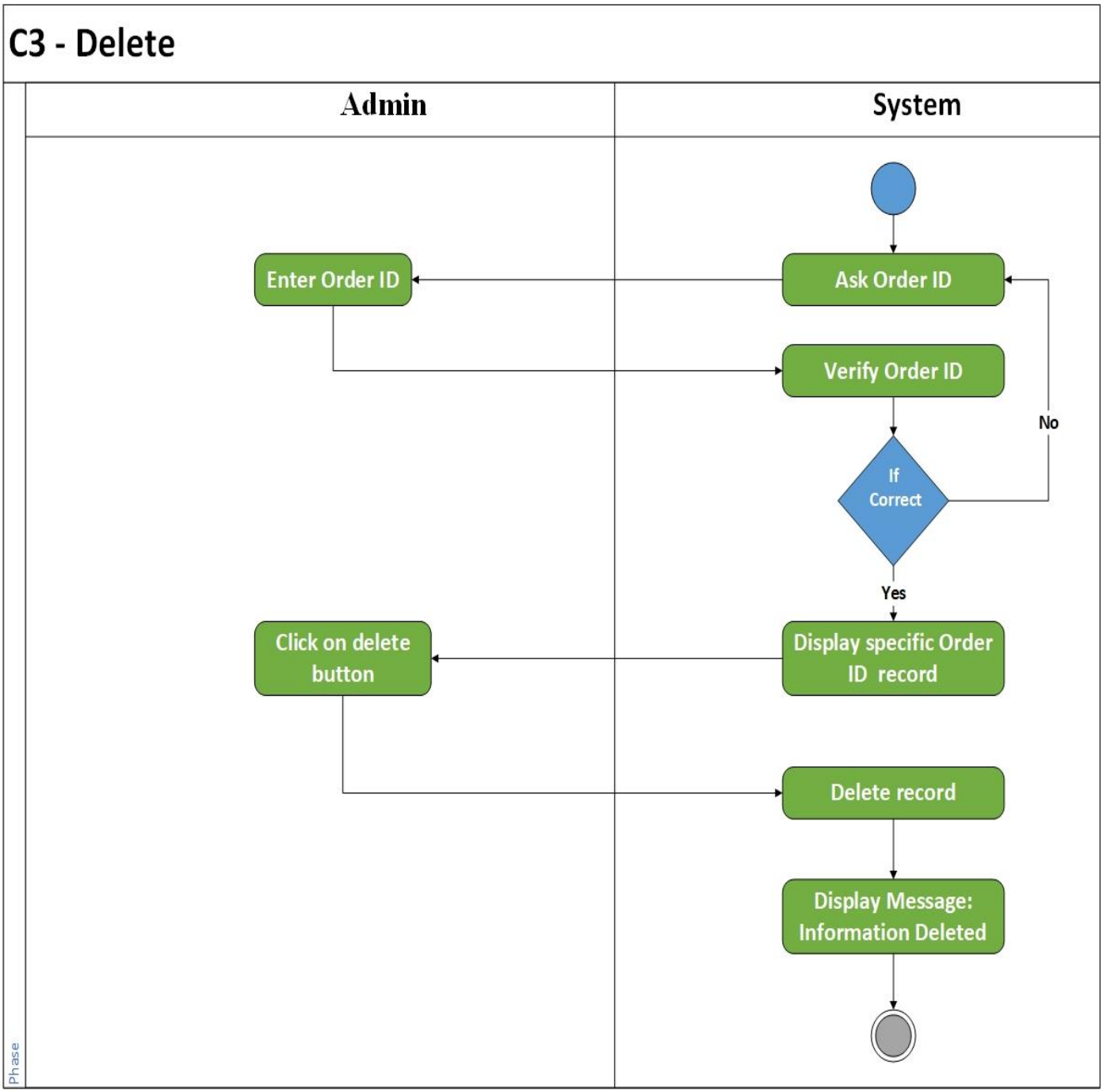

Figure 32: Delete

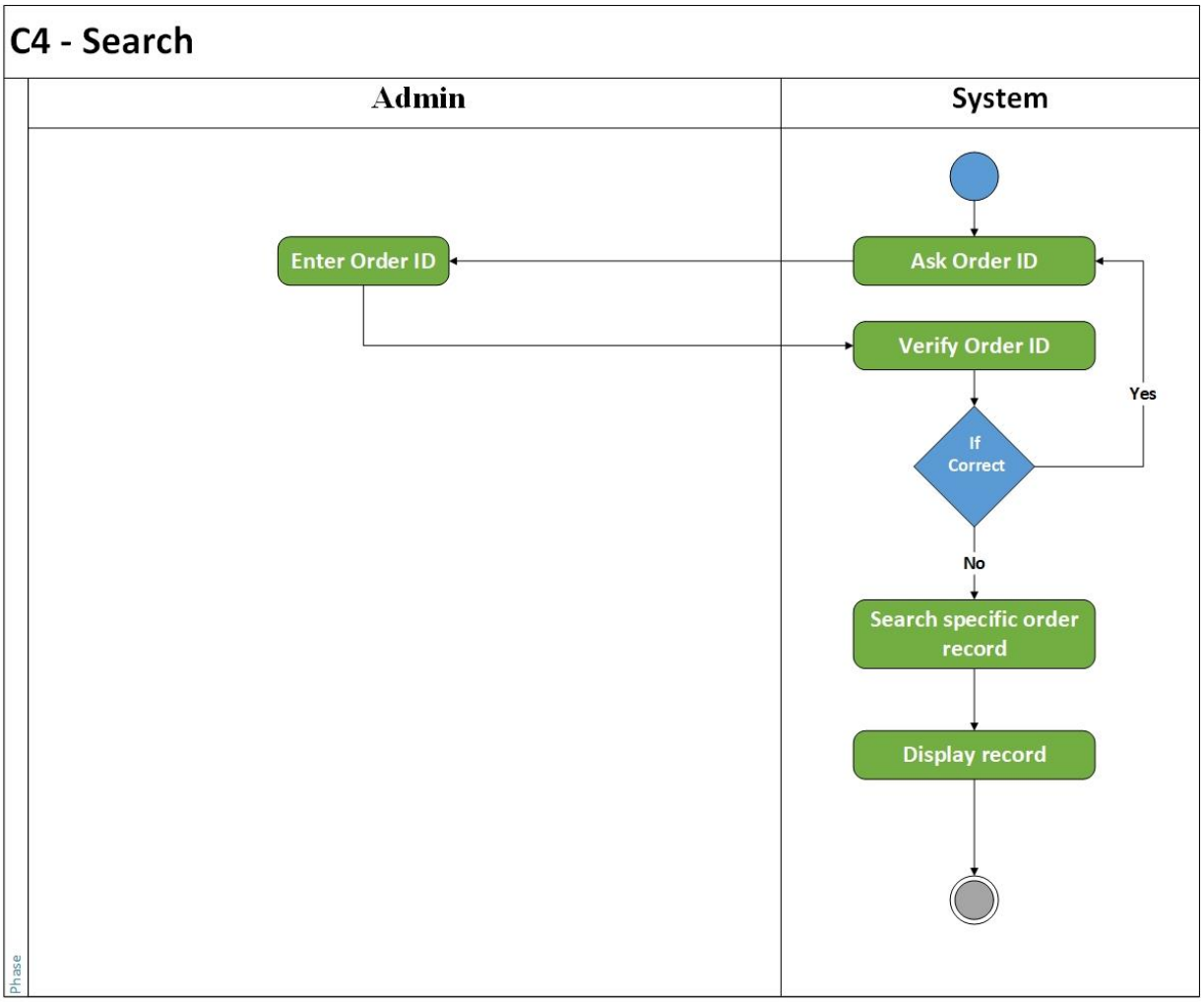

Figure 33: Search

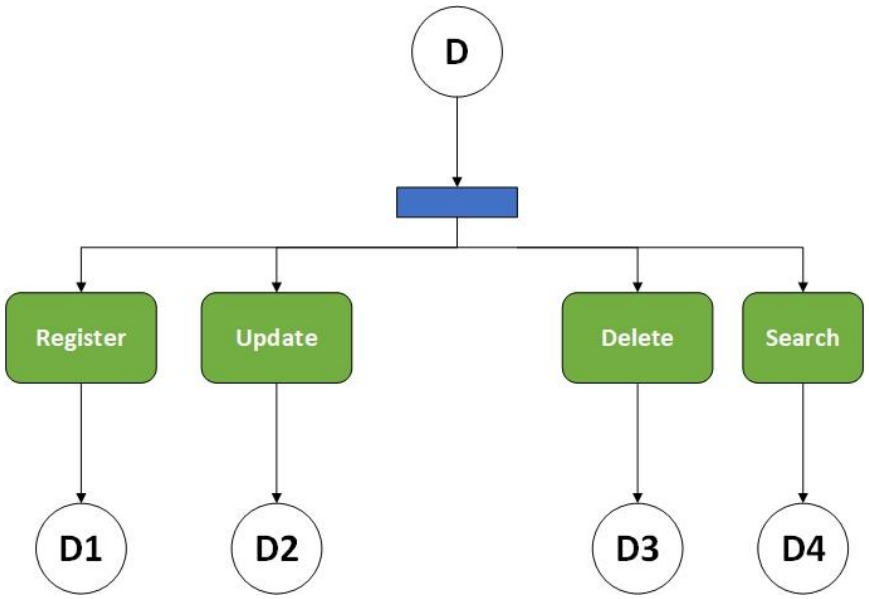

Figure 34: Supplier

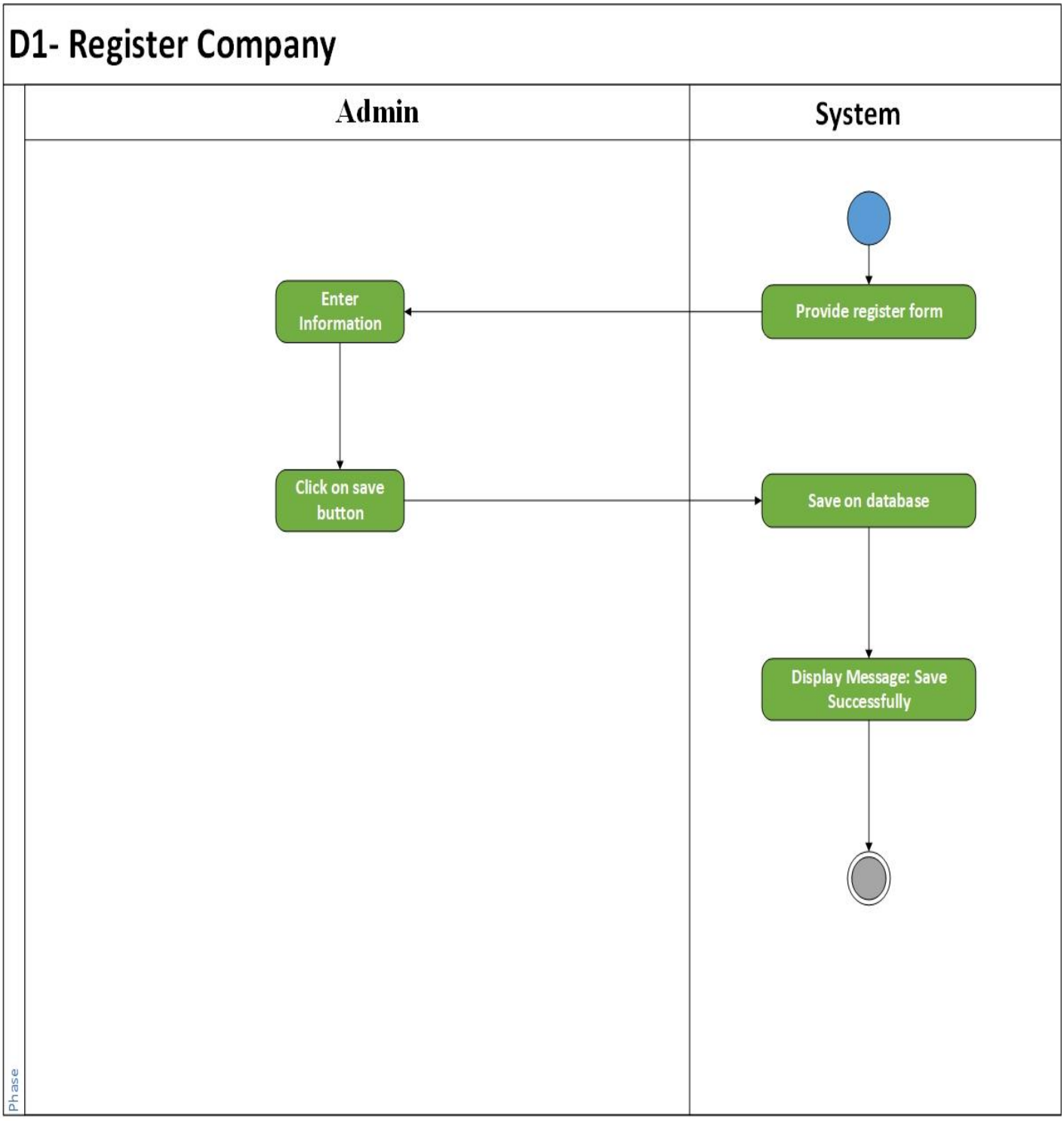

Figure 35: Register Company

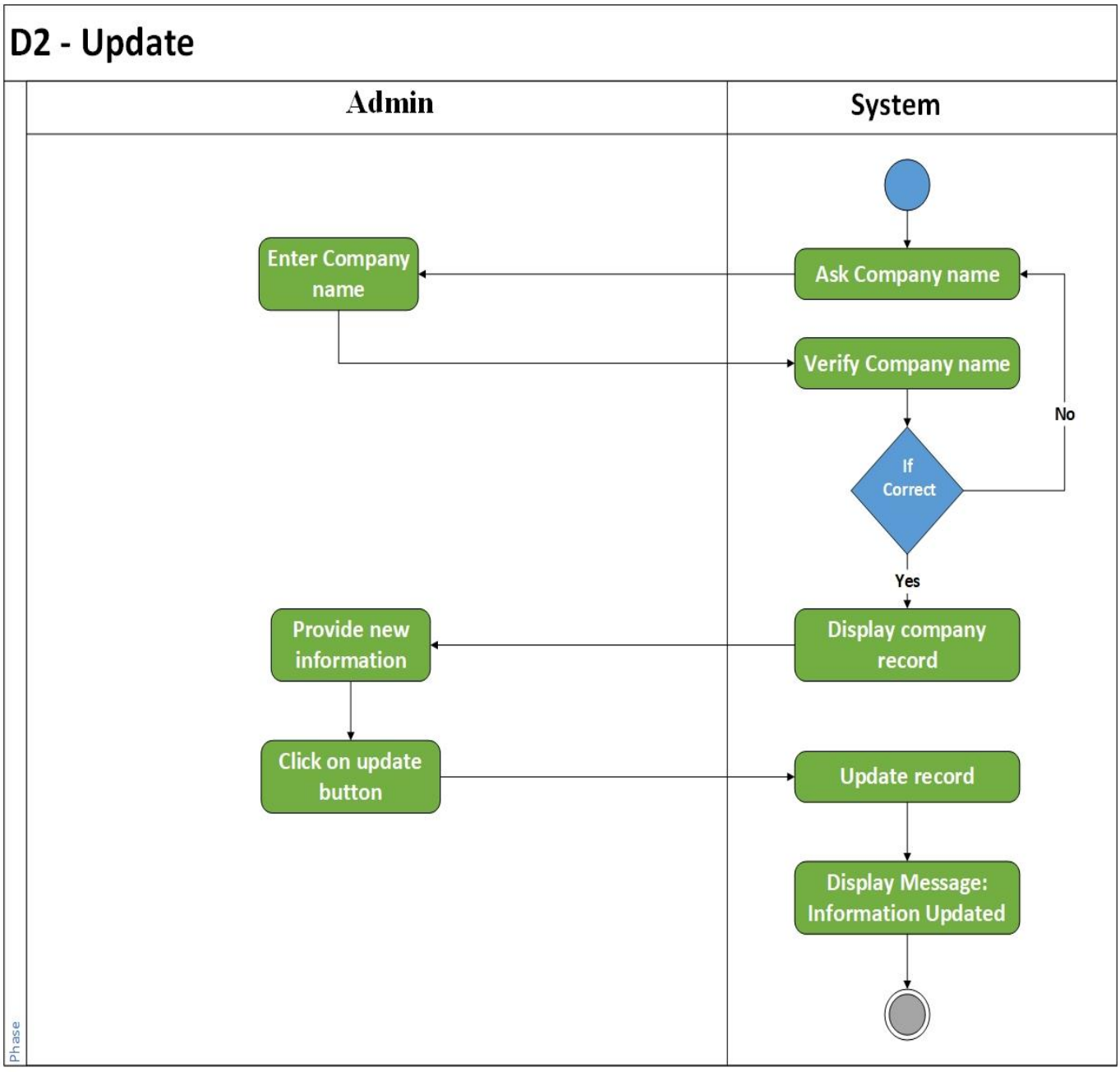

Figure 36: Update

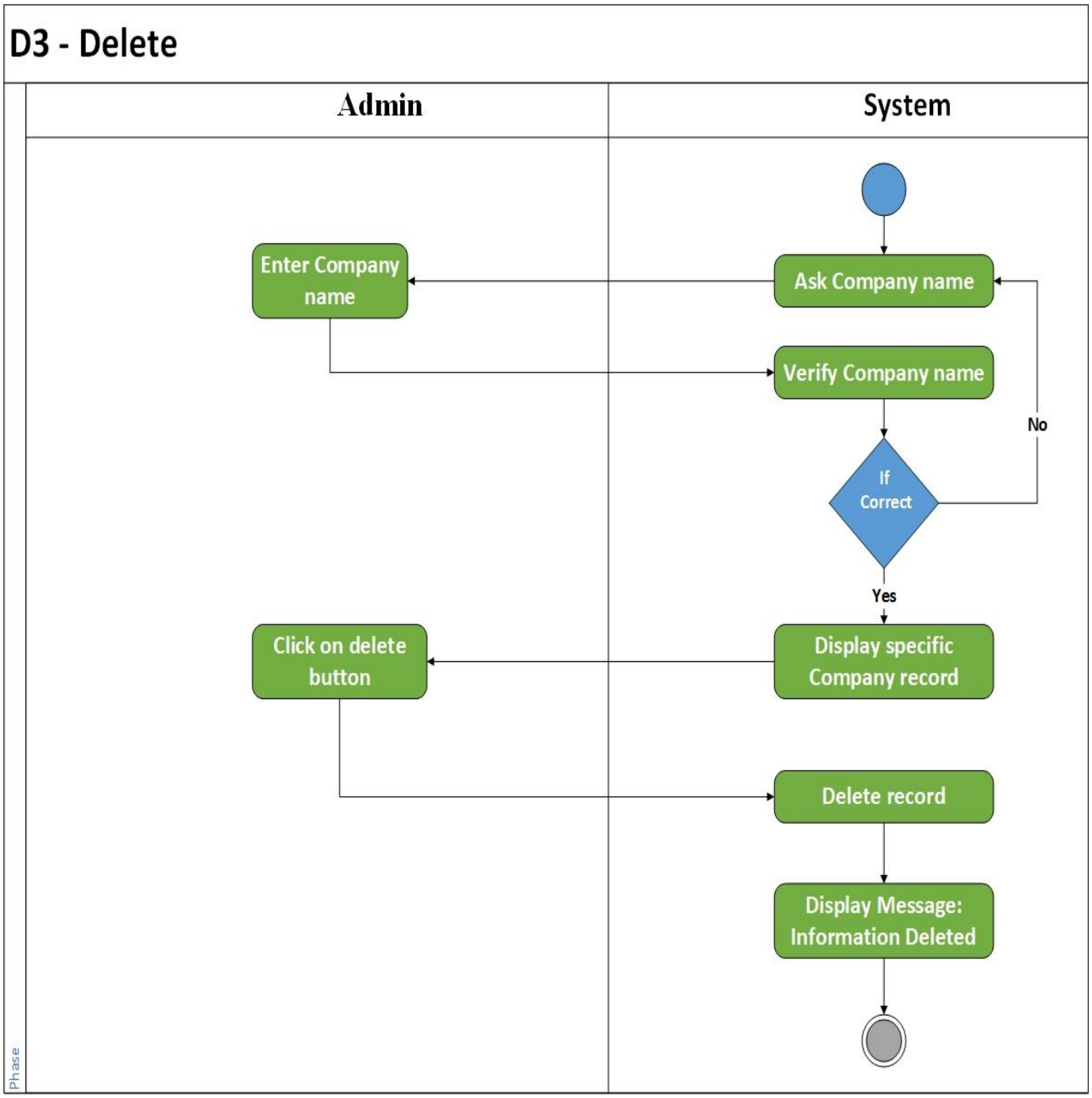

Figure 37: Delete

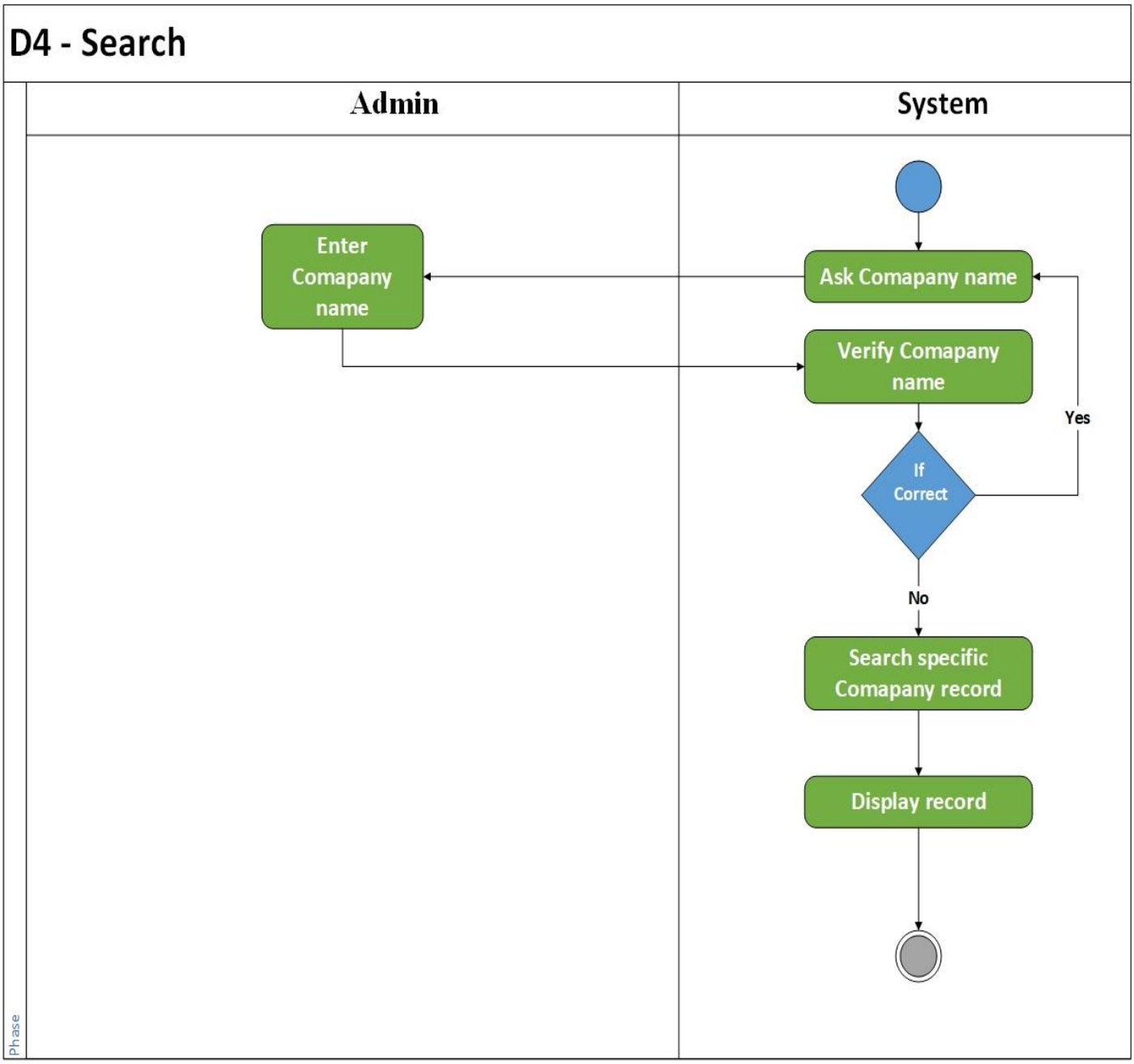

Figure 38: Search

3.0 Functionalities

3.1 Description of Interface

3.1.1 Login Interface

Login screen is the first screen of the system. Admin needs to provide the registered username and password to login into the system. If the provided username and password is not corrected then this system will give a message that username and password are not correct otherwise if it is correct then the user will access into the system and new screen Main Menu will appear.

3.1.2 Main Menu Interface

Main Menu Screen will come after login Screen. This screen is the menu of the system, here the screen will show four option to choose that is Book, Order, Student, Supplier. From this screen, admin can choose different task based on his working.

3.1.3 Student Interface

This screen is uses for providing all the information about a student to register into the system. Admin can update, delete or search a specific detail about a student. Besides that, it has a delete all button that can delete all the record of the system about the student, if needed. Furthermore, a table is provided so that user can view when new data added, deleted or updated from the system instantly

3.1.4 Order Interface

This screen is uses for ordering a specific book to the supplier. Admin needs to fill the order form and choose a company that is already registered then need to press save button to order. Here, admin can update, delete and search a specific book order detail. Besides that, a table is provided so that user can view when new data added, deleted or updated from the system instantly.

3.1.5 Supplier Interface

This screen is uses for recording all the information about the book supplier. Here the user can register a new company, delete or update. There is another option to search for a specific company by providing the name of the company. Moreover, a table is given to view all the information about the company.

3.1.6 Book Detail Interface

This screen has a number of option to do work that is admin can fill the book detail form to add a new book to the system or he can update and delete the book from the system. Besides that, a table is given in this screen so that user can view instantly what data is added, updated or deleted. Moreover, a search option is also provided for searching specific book that is already registered in the system.

3.1.7 Book Borrow Interface

This screen is used for issue or return book record from the student. Here for new book issue, admin needs to search student name from the combo box, if the student is registered then only their name will appear in the combo box. So, after choosing a selected name their TP number will automatically come and then admin need to search that book and fill up the other form to save the record. Beside that when a student wants to return a book that user can search by their name and all record about the issue book will come then the user need to update from issued to return and save the record. Moreover, the user can search, delete and update a specific record.

4.0 Interfaces

4.1 Interface Screenshot

4.1.1 Login

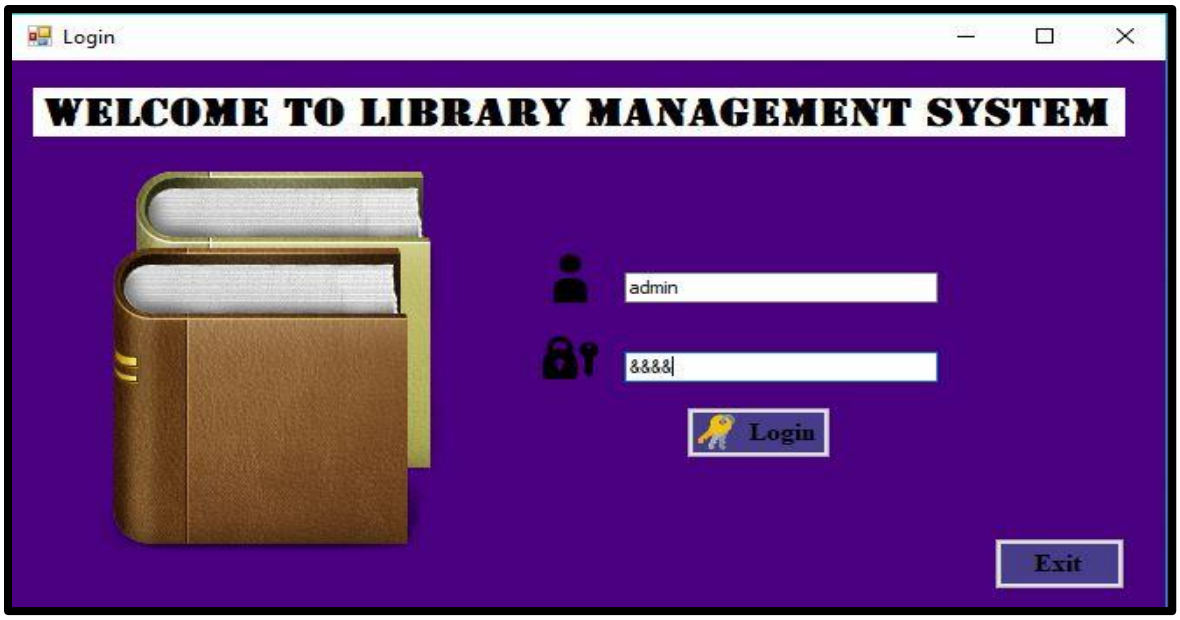

Figure 39: Login Screen

4.1.2 Main Menu

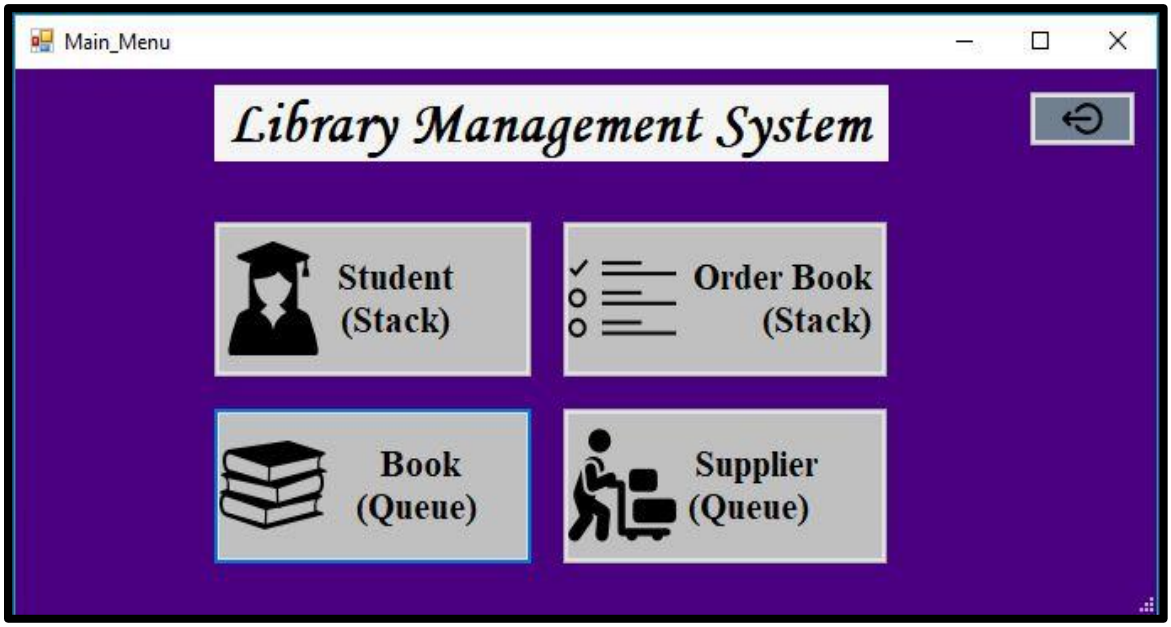

Figure 40 : Main Menu

4.1.3 Student Information

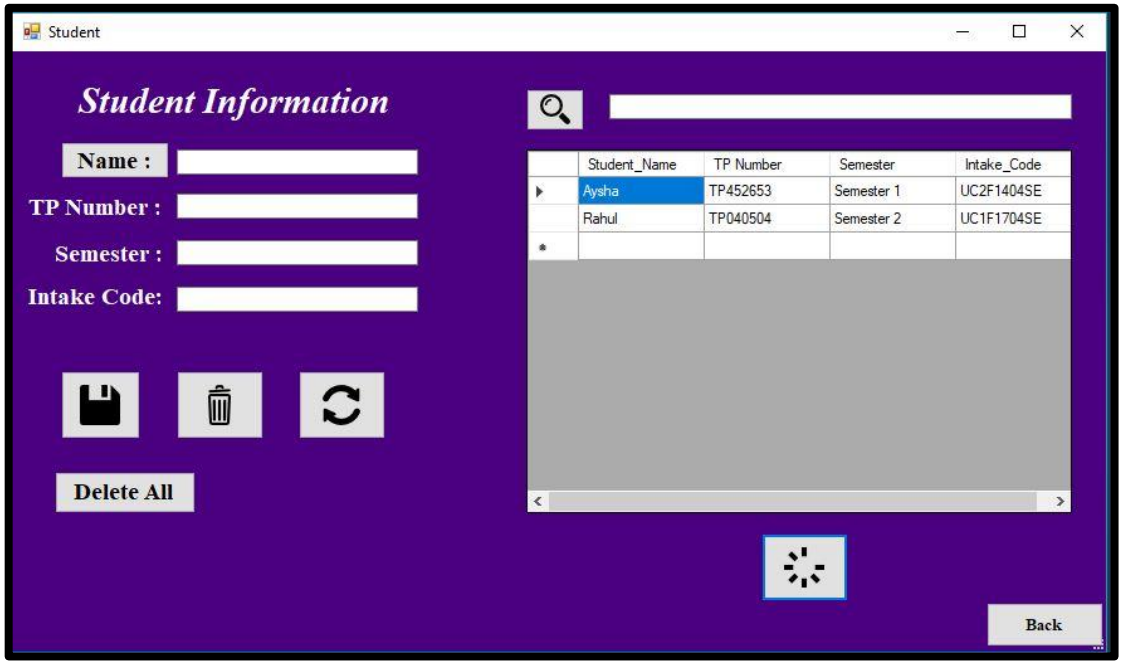

Figure 41: Student Information

4.1.4 Book Order

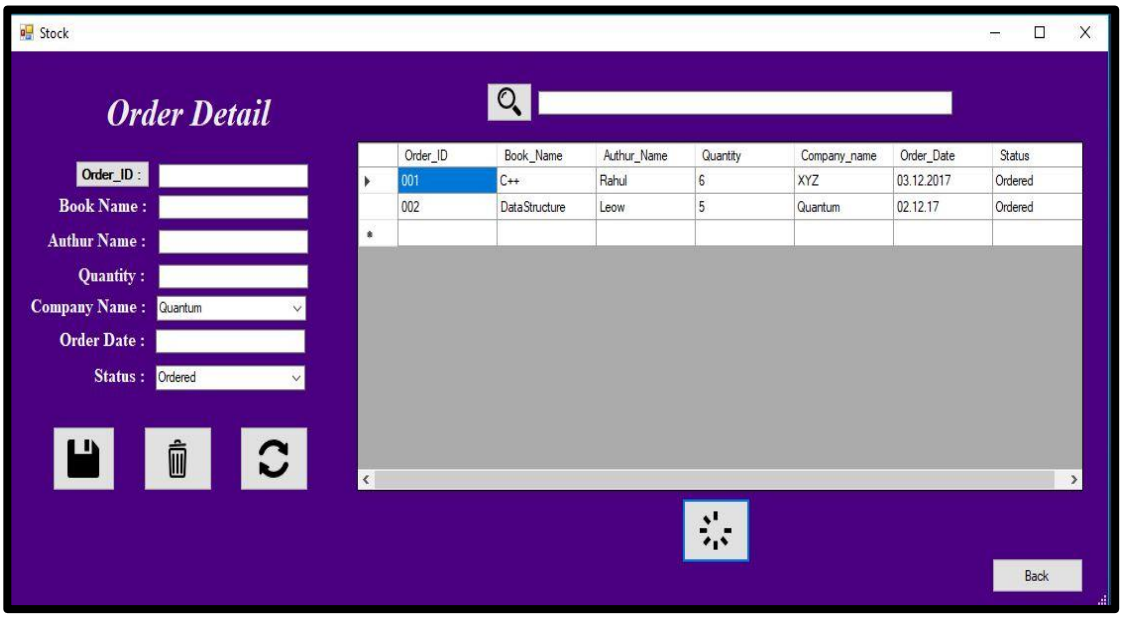

Figure 42: Order Detail

4.1.5 Company Information

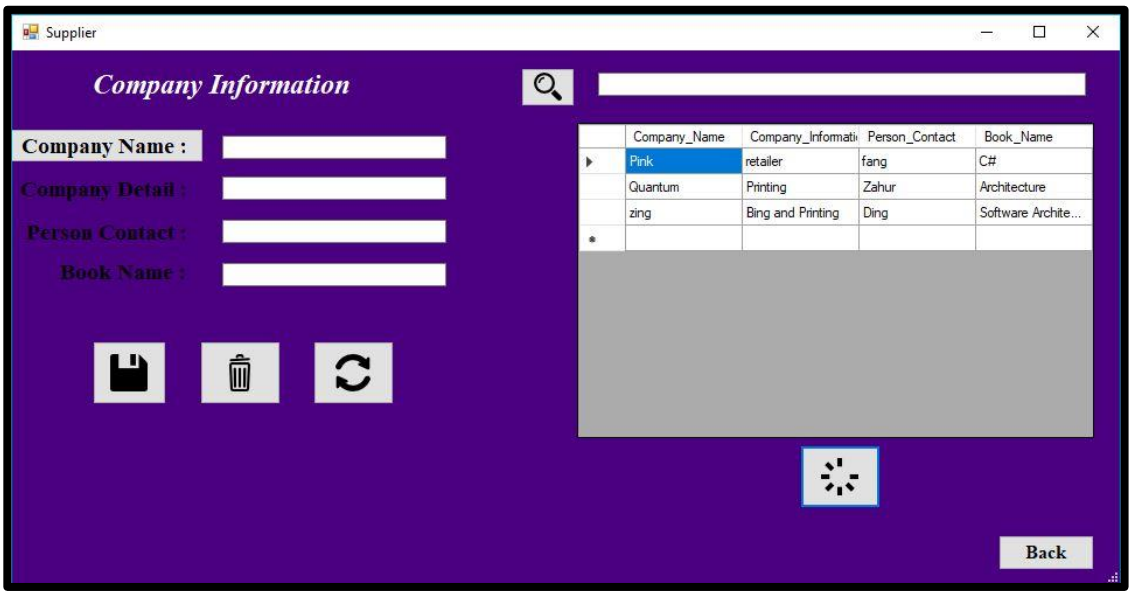

Figure 43: Company Information

4.1.6 Book Detail

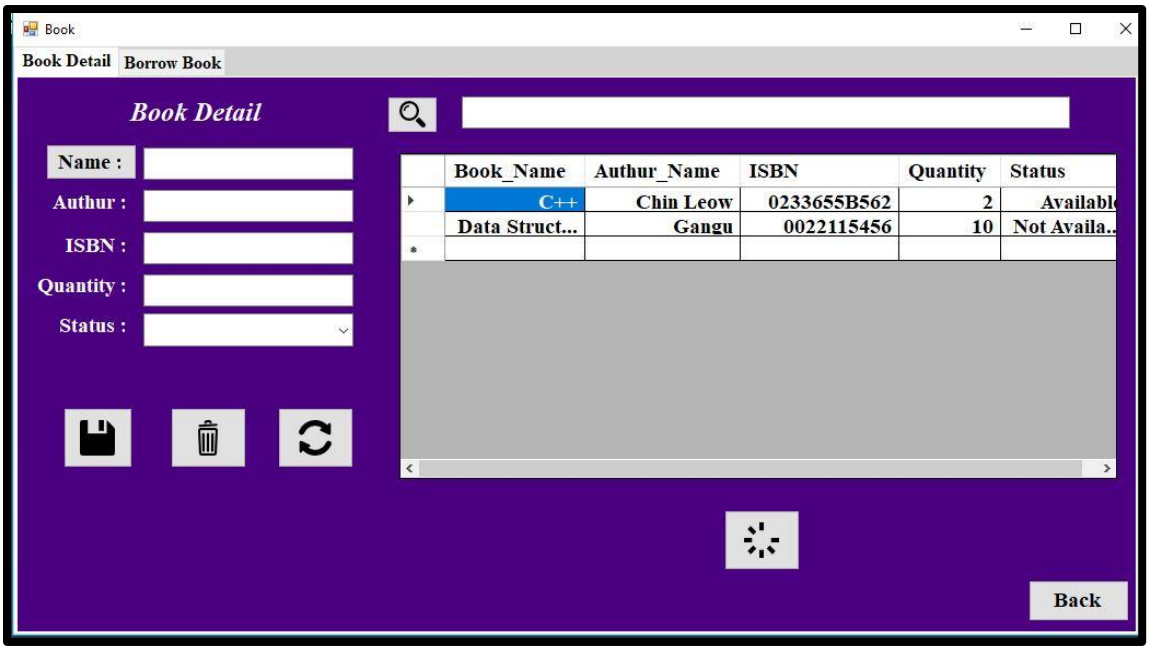

Figure 44: Book Detail

4.1.7 Book Borrow Information

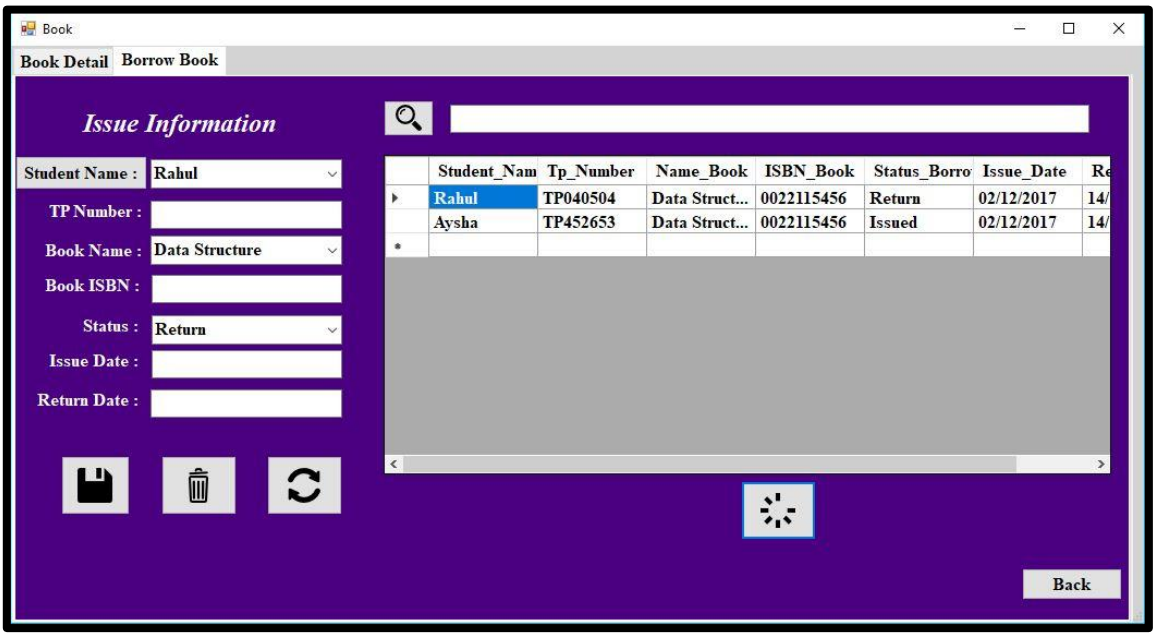

Figure 45: Book Issue Information

4.2 Program Code

4.2.1 Login

```
#pragma endregion
   private: System::Void btnLogin_Click(System::Object^  sender,
     System::EventArgs^  e) {
       String^ Username;
       String^ Password;
       //connection to database
       SqlConnection^ connection = gcnew SqlConnection();
       connection->ConnectionString = "Data Source=.\
         \SQLEXPRESS;Database=LibraryManagement;Integrated Security=true";

       try
       {

           SqlCommand^ query = gcnew SqlCommand();
           query->Connection = connection;
           //Select from database table
           query->CommandText = "SELECT * FROM [dbo].[Login] WHERE [Username]
              = '" + this->txtUsername->Text + "' AND [Password] ='" + this-
             >txtPassword->Text + "'", connection;
           connection->Open();
           SqlDataReader^ reader = query->ExecuteReader();
           while (reader->Read()) {
               Username = reader->GetString(0);
               Password = reader->GetString(1);
           }
           //Identify that username and password are correct
           if (this->txtUsername->Text == Username&&this->txtPassword->Text
             == Password)
           {
               //MessageBox::Show("Welcome");
                       this->Hide();
                       Main_Menu ^obj1 = gcnew Main_Menu(this);
                       obj1->ShowDialog();
           }
           else
           {
               MessageBox::Show("Please enter correct Username and
                 Password!");
           }
       }
       catch (Exception^ ex)
       {
           MessageBox::Show("Error");
           MessageBox::Show(ex->Message);
       }
       finally
       {
           connection->Close();
       }
```

Figure 45: Login Screen Code

4.2.2 Main menu

```
#pragma endregion
    private: System::Void btnLogout_Click(System::Object^  sender,
      System::EventArgs^  e) {
        this->Hide();
        obj->Show();
    }
private: System::Void btnBook_Click(System::Object^  sender,
  System::EventArgs^  e) {
    this->Hide();
    Book ^frm2 = gcnew Book(this);
    frm2->ShowDialog();
}
private: System::Void btnStudent_Click(System::Object^  sender,
  System::EventArgs^  e) {
    this->Hide();
    Student ^frm1 = gcnew Student(this);
    frm1->ShowDialog();
}
private: System::Void btnStock_Click(System::Object^  sender,
  System::EventArgs^  e) {
    this->Hide();
    Stock ^frm3 = gcnew Stock(this);
    frm3->ShowDialog();
}
private: System::Void btnSupplier_Click(System::Object^  sender,
  System::EventArgs^  e) {
    this->Hide();
    Supplier ^frm4 = gcnew Supplier(this);
    frm4->ShowDialog();
}
};
}
```

Figure 47: Main Menu Code

4.2.3 Save stack

```
private: System::Void btnAdd_Click(System::Object^  sender, System::EventArgs^
    e) {
    //connection to database
    String^ constring = "Data Source=.\
      \SQLEXPRESS;Database=LibraryManagement;Integrated Security=true";
    SqlConnection^ conDataBase = gcnew SqlConnection(constring);
    //Table of database and insert the value inside database
    SqlCommand^ cmdDataBase = gcnew SqlCommand("INSERT INTO [dbo].
      [Order_Detail]([OrderID],[Book_Name], [Aurthur_Name], [Quantity],
      [Company_Name], [Order_Date], [Status])VALUES ('" + this->txtOrderid-
      >Text + "' ,'" + this->txtBookname->Text + "' ,'" + this->txtAuthurname-
      >Text + "' , '" + this->txtQuantity->Text + "' , '" + this-
      >combCompanyname->Text + "' , '" + this->txtOrderdate->Text + "' , '" +
      this->combStatus->Text + "');", conDataBase);

    //Reader
    SqlDataReader^ myReader();
    try {

        //open connection
        conDataBase->Open();


        SqlDataReader^ myReader = cmdDataBase->ExecuteReader();
        while (myReader->Read()) {


        }

        MessageBox::Show("New Order Added");
    }
    catch (Exception^ ex) {
        MessageBox::Show(ex->Message);
    }

   // Textbox clear;
    txtOrderid->Text = ("");
    txtBookname->Text = ("");
    txtAuthurname->Text = ("");
    txtQuantity->Text = ("");
    txtOrderdate->Text = ("");

}
```

Figure 48: Save Stack Code

4.2.4 Delete stack

```
private: System::Void btnDelete_Click(System::Object^  sender, System::EventArgs^  e) {
    SQLinkedList s1; // object

    for (int i = 0; i < Student_Information->Rows->Count - 1; i++)
    {
        s1.push(Student_Information->Rows[i]->Cells["Student_Name"]->Value->ToString());
    }
    String^ theprevioustop = s1.getTop(); //get the data that comes first
    s1.pop();  //delete the data that recently added
    Student_Information->Rows->Clear();

    //Sql connection
    SqlConnection^ connection = gcnew SqlConnection("Data Source=.\\SQLEXPRESS;Database=LibraryManagement;Integrated Security=true");

    try {
        connection->Open();

        //delete from database
        SqlCommand^ cmd = gcnew SqlCommand("DELETE FROM [dbo].[Student_Information] WHERE Student_Name ='" + theprevioustop + "'", connection);
        cmd->ExecuteNonQuery();
        MessageBox::Show("Successfully Deleted");

        SqlCommand^ command = gcnew SqlCommand("SELECT * FROM  [dbo].[Student_Information]", connection);

        SqlDataReader^ reader2 = command->ExecuteReader();
        while (reader2->Read())
        {
            Student_Information->Rows->Add(reader2[0]->ToString(), reader2[1]->ToString(), reader2[2]->ToString(), reader2[3]->ToString());
        }
        reader2->Close();

    }
    catch (SqlException^ e) {
        MessageBox::Show(e->Message);
    }
```

Figure 49: Delete Stack Code

4.2.5 Save Queue

```
private: System::Void btnAdd_Click(System::Object^  sender, System::EventArgs^
    e) {
    SQLinkedList s1;

    String^ thepreviousstop = s1.getBottom();

    String^ constring = "Data Source=.\
      \SQLEXPRESS;Database=LibraryManagement;Integrated Security=true";
    SqlConnection^ conDataBase = gcnew SqlConnection(constring);
    SqlCommand^ cmdDataBase = gcnew SqlCommand("INSERT INTO [dbo].
      [Company_Detail]([Company_Name], [Company_Information],
      [Person_Contact], [Book_Name])VALUES ('" + this->txtName->Text + "' ,'"
      + this->txtCompanydetail->Text + "' ,'" + this->txtPerson->Text + "' ,
      '" + this->txtBookname->Text + "');", conDataBase);
    SqlCommand^ cmd = gcnew SqlCommand("INSERT INTO [dbo].[Company_Detail]
      WHERE [Company_Name] ='" + thepreviousstop + "'", conDataBase);
    //INSERT INTO [dbo].[Company_Detail] WHERE [Company_Name]
    SqlDataReader^ myReader();
    try {
        conDataBase->Open();


        SqlDataReader^ myReader = cmdDataBase->ExecuteReader();
        while (myReader->Read()) {


        }

        MessageBox::Show("New Information Saved");
    }
    catch (Exception^ ex) {
        MessageBox::Show(ex->Message);
    }
```

Figure 50: Save Queue Code

4.2.6 Delete Queue

```
private: System::Void btnDelete_Click(System::Object^  sender,
  System::EventArgs^  e) {
    SQLinkedList s1;

    for (int i = 0; i < Supplier_info->Rows->Count - 1; i++)
    {
        s1.enqueue(Supplier_info->Rows[i]->Cells["Company_Name"]->Value-
          >ToString());
    }
    String^ thebottomtop = s1.getTop();

    s1.dequeue();
    Supplier_info->Rows->Clear();

    SqlConnection^ connection = gcnew SqlConnection("Data Source=.\
      \SQLEXPRESS;Database=LibraryManagement;Integrated Security=true");

    try {
        connection->Open();


        SqlCommand^ cmd = gcnew SqlCommand("DELETE FROM [dbo].[Company_Detail]
           WHERE Company_Name='" + thebottomtop + "'", connection);
        cmd->ExecuteNonQuery();
        MessageBox::Show("Successfully Deleted");

        SqlCommand^ command = gcnew SqlCommand("SELECT * FROM [dbo].
          [Company_Detail]", connection);

        SqlDataReader^ reader2 = command->ExecuteReader();
        while (reader2->Read())
        {
            Supplier_info->Rows->Add(reader2[0]->ToString(), reader2[1]-
              >ToString(), reader2[2]->ToString(), reader2[3]->ToString());
        }
        reader2->Close();


    }
    catch (SqlException^ e) {
        MessageBox::Show(e->Message);
    }
    finally{
        connection->Close();
    }
```

Figure 51: Delete Queue Code

4.2.7 Delete All

```
private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e) {

    try
    {
        //Sql connection
        SqlConnection^ connection = gcnew SqlConnection();
        connection->ConnectionString = "Data Source=.\\SQLEXPRESS;Database=LibraryManagement;Integrated Security=true";
        connection->Open(); //connection open

        //Delete all data from the database table
        SqlCommand^ query = gcnew SqlCommand("Delete From Student_Information", connection);
        query->ExecuteNonQuery();
        connection->Close(); //connection close
        MessageBox::Show("Deleted all data successfully");


    }
    catch (Exception ^ e)
    {
        MessageBox::Show(e->Message);
    }
```

Figure 52: Delete All Code

4.2.8 Search

```
private: System::Void button3_Click(System::Object^  sender,
  System::EventArgs^  e) {
    Supplier_info->Rows->Clear(); //clear table row

    try
    {
        //connect to database
        SqlConnection^ connection = gcnew SqlConnection();
        connection->ConnectionString = "Data Source=.\
          \SQLEXPRESS;Database=LibraryManagement;Integrated Security=true";
        connection->Open(); //connection open

        SqlCommand^ query = gcnew SqlCommand();
        query->Connection = connection;
        String^ tp = txtSearch->Text; //text box using for search

        //select from database table with the first name
        query->CommandText = "SELECT * FROM [dbo].[Company_Detail] WHERE
          Company_Name='" + tp + "'", connection;


        SqlDataReader^ reader = query->ExecuteReader();

        Boolean exist;

        while (reader->Read() == true)
        {
            String^ id = reader->GetString(0);
            if (tp == id) //match the name

            {
                exist = true;
                Supplier_info->Rows->Add(tp, reader->GetString(1), reader-
                  >GetString(2), reader->GetString(3));
            }
            else
            {
                exist = false;
            }
        }
        reader->Close();
        if (exist == true)
        {
            return;
        }
        else
        {
            MessageBox::Show("Company name does not exist");
        }
```

Figure 53: Search Code

4.2.9 Update

```
    try {
        SqlConnection^ connection = gcnew SqlConnection("Data Source=.\
          \SQLEXPRESS;Database=LibraryManagement;Integrated Security=true");
        connection->Open();
        String^ name = txtName->Text;
        String^ tpnumber = txtTPnumber->Text;
        String^ semester = txtSemester->Text;
        String^ intakecode = txtIntakeCode->Text;

        SqlCommand^ query = gcnew SqlCommand();
        query->Connection = connection;
        query->CommandText = "UPDATE [dbo].[Student_Information] SET TP_Number
           = '" + tpnumber + "', [Semester] = '" + semester + "',
          [Intake_Code] = '" + intakecode + "' WHERE Student_Name = '"+name
          +"'", connection;
        SqlDataReader^ reader = query->ExecuteReader();
        MessageBox::Show("Update Successful!");
        connection->Close();
        reader->Close();
    }
    catch (Exception^ e)
    {
        MessageBox::Show(e->Message);
    }

    //Text box clear;
    txtName->Text = ("");
    txtTPnumber->Text = ("");
    txtSemester->Text = ("");
    txtIntakeCode->Text = ("");

}
private: System::Void button3_Click(System::Object^  sender,
  System::EventArgs^  e) {
    Student_Information->Rows->Clear();

    try
    {
        SqlConnection^ connection = gcnew SqlConnection();
        connection->ConnectionString = "Data Source=.\
```

Figure 54: Update Code

4.3 Stack and Queue

4.3.1Stack

Save

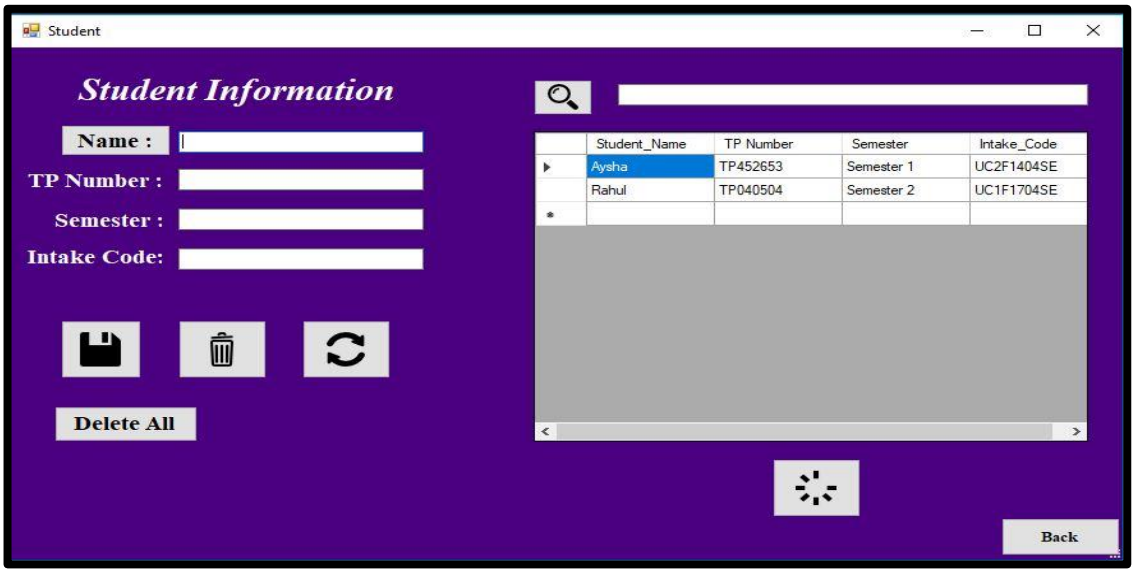

Figure 55: Before Save Data

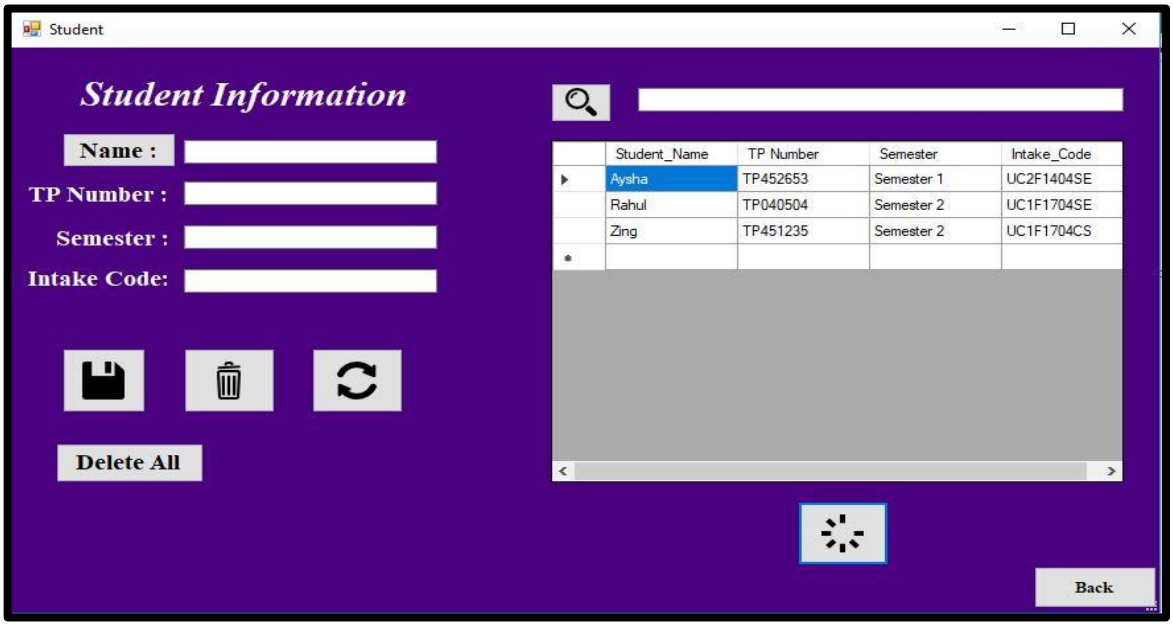

Figure 56: After Save Data

Delete

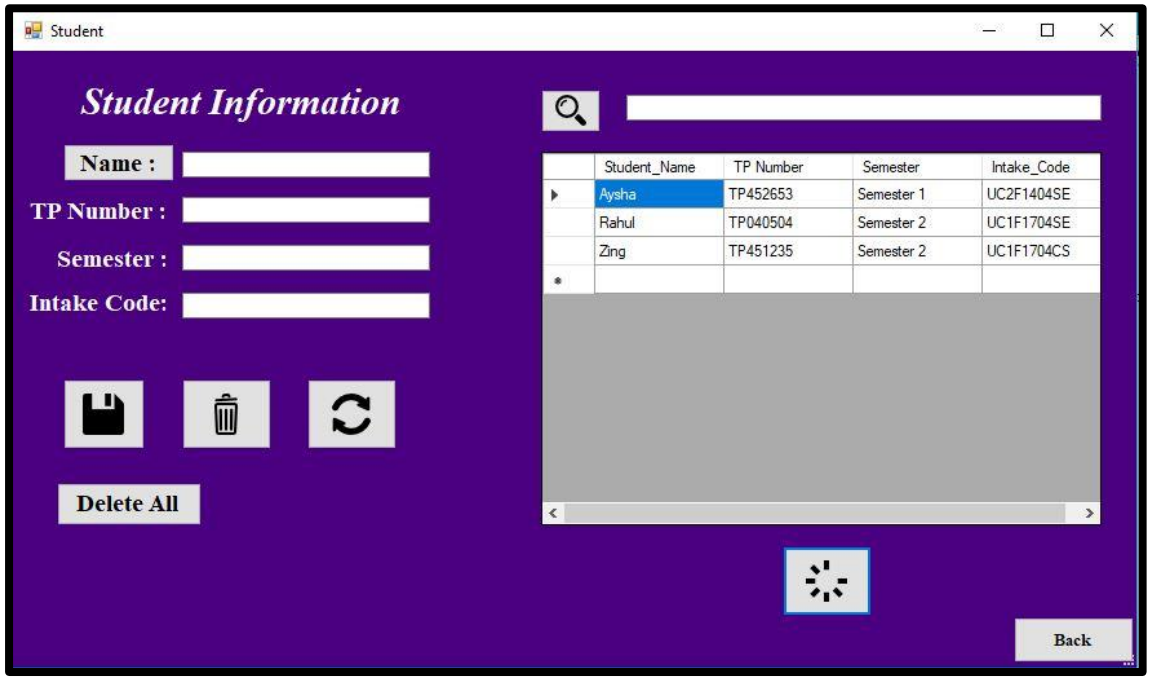

Figure 57: Before Delete Data

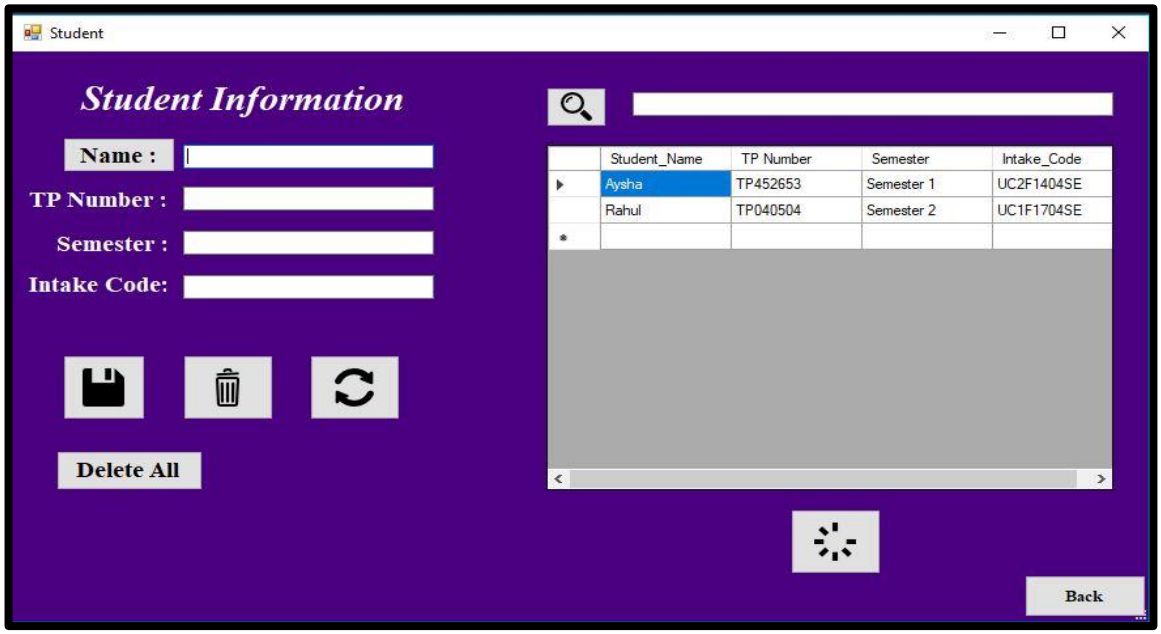

Figure 58: After Delete Data

4.3.2 Queue

Save

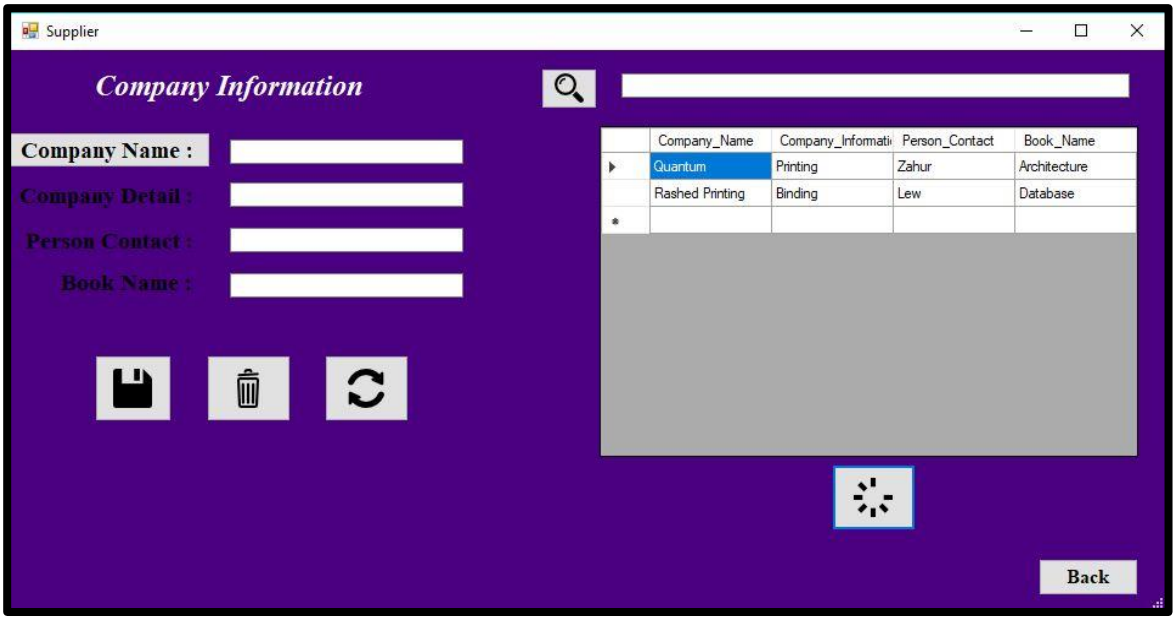

Figure 59: Before Save Data

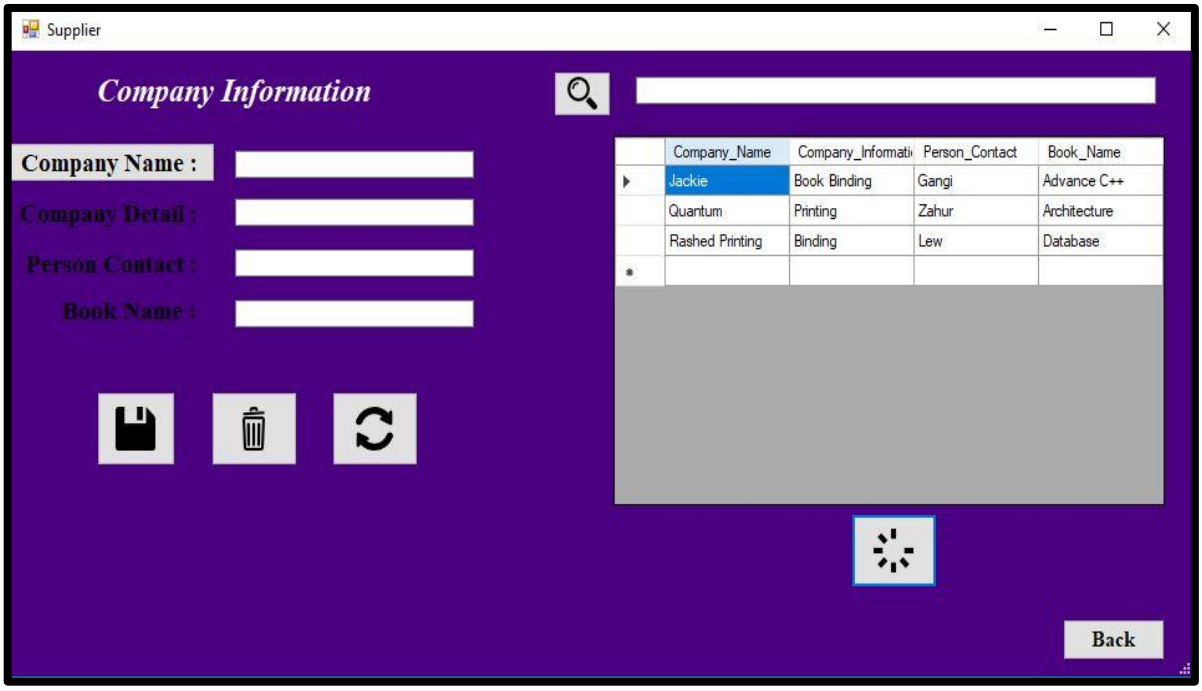

Figure 60: After Save Data

Delete

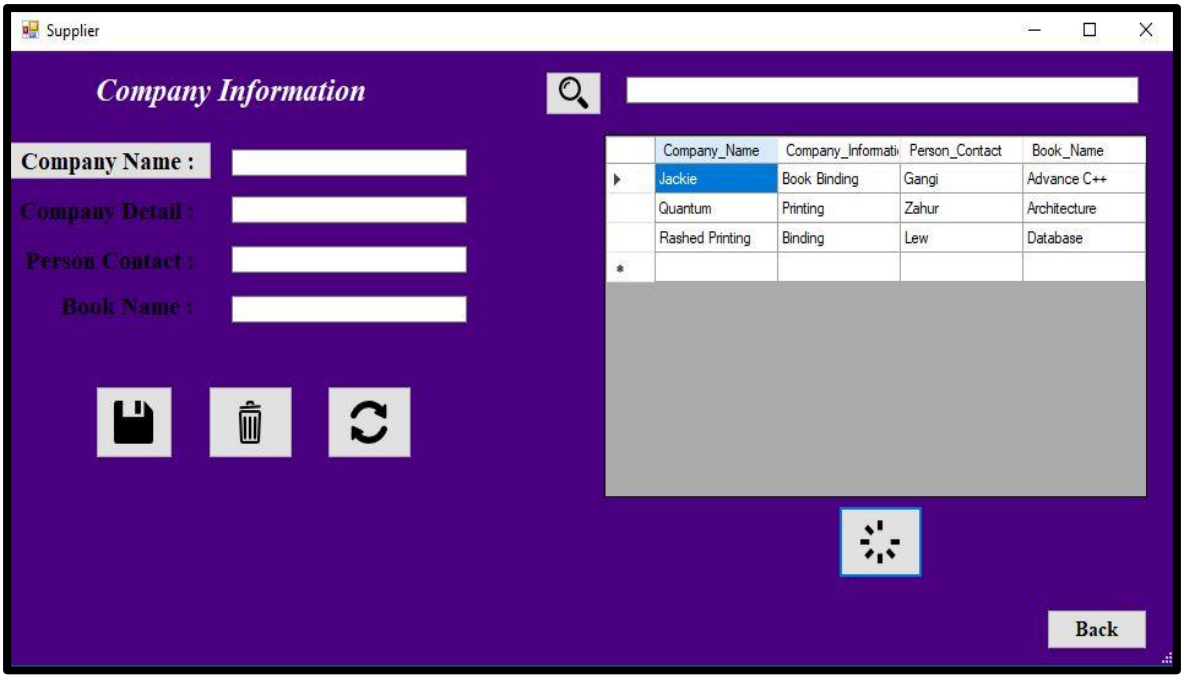

Figure 61: Before Delete Data

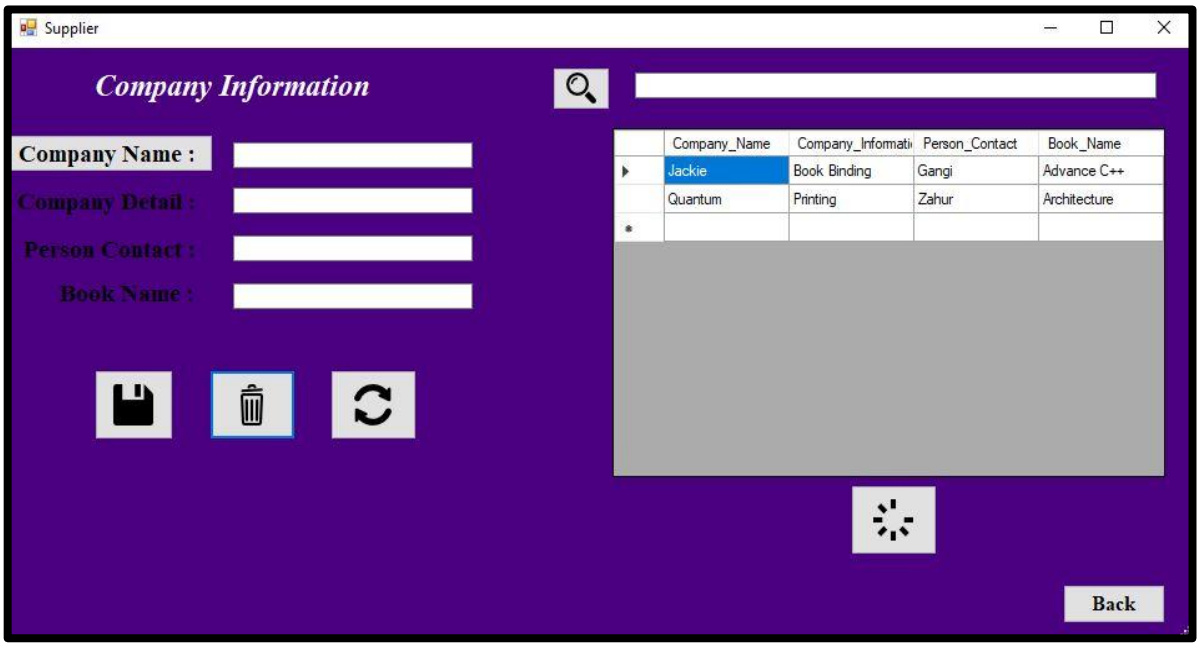

Figure 62: After Delete Data

4.4 Database Table

4.4.1 Login Table

SH\SQLEXPRESS.Lib...ement - dbo.Login

	Username	Password
▶	admin	1212
*	NULL	NULL

Figure 63: Login Table

4.4.2 Book Detail Table

SH\SQLEXPRESS.Libr...- dbo.Book_Detail

	Book_Name	Aurthur_Name	ISBN	Quantity	Status
▶	C++	Hangi	2356986532	7	Available
	Data Structure	Gangu	0022115456	10	Not Available
	Database	Teong	12354688978	8	Available
*	NULL	NULL	NULL	NULL	NULL

Figure 64: Book Detail Table

4.4.3 Borrow Book Table

SH\SQLEXPRESS.Li...- dbo.Borrow_Book

Student_Name	TP_Number	Book_Name	ISBN	Status	Issue_Date	Return_Date
Aysha	TP452653	Data Structure	0022115456	Issued	02/12/2017	14/12/2017
Kashish	TP213654	C++	2356986532	Issued	02/12/2017	14/12/2017
Rahul	TP040504	Data Structure	0022115456	Return	02/12/2017	14/12/2017
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Figure 65: Borrow Book Table

4.4.4 Company Detail Table

SH\SQLEXPRESS.Li...bo.Company_Detail

Company_Na...	Company_Info...	Person_Contact	Book_Name
Jackie	Book Binding	Gangi	Advance C++
Quantum	Printing	Zahur	Architecture
Super Printing	Printing	Harry	Mathmatices
NULL	*NULL*	*NULL*	*NULL*

Figure 66: Company Detail Table

4.4.5 Student Information Table

SH\SQLEXPRESS.Lib...udent_Information

Student_Name	TP_Number	Semester	Intake_Code
Aysha	TP452653	Semester 1	UC2F1404SE
Bikash	TP565604	Semester 2	UC1F1704IT
Kashish	TP213654	Semester 3	UC2F1704AI
Rahul	TP040504	Semester 2	UC1F1704SE
NULL	*NULL*	*NULL*	*NULL*

Figure 66: Student Information Table

4.4.6 Order Detail Table

SH\SQLEXPRESS.Libr...- dbo.Order_Detail

	OrderID	Book_Name	Aurthur_Name	Quantity	Company_Na...	Order_Date	Status
▶	001	C++	Rahul	6	XYZ	03.12.2017	Ordered
	002	DataStructure	Leow	5	Quantum	02.12.17	Ordered
	003	Database	Lew chang	6	Jackie	02/2/2017	Delivered
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 67: Order Detail Table

5.0 Conclusion

In conclusion, the system "Library Management System" has all the functionality that need to do daily operation in the library. It allows the work with error free and no chance to lose data about the book and registered student, company, order detail of book. It has a large capacity database that can store huge number of data and beside that the user of this system can delete the unnecessary data that will no need in future. Overall this system will reduce the effort of the staff.

This system is developed by using C++ programming language and Microsoft SQL Server as a database. Hence, data structure technique that is stack and queue with linked list is implemented into this system to work much more efficient. Several testing's is carried out to find the error of this system and after a series of testing this system is now error free and fully functional. Thus, the project "Library Management System" is portable and flexible for further enhancement in future requirement of the organization.

6.0 References

1.Cs.cmu.edu. (2017). *LinkedLists*. [online] Available at: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html [Accessed 29 Dec. 2017].

2.Learn C++. (2017). *Learn C++*. [online] Available at: http://www.learncpp.com/ [Accessed 29 Dec. 2017].

3.MCQs, C., Lists, L., patidar, k. and pane, e. (2017). *C++ program to implement Stack using Linked List - Pro Programming*. [online] Pro Programming. Available at: https://proprogramming.org/c-program-to-implement-stack-using-linked-list/ [Accessed 29 Dec. 2017].

4.Sanfoundry. (2017). *C++ Program to Implement Queue using Linked List - Sanfoundry*. [online] Available at: http://www.sanfoundry.com/cpp-program-implement-queue-linked-list/ [Accessed 29 Dec. 2017].

5.Sanfoundry. (2017). *C++ Program to Implement Stack using Linked List - Sanfoundry*. [online] Available at: http://www.sanfoundry.com/cpp-program-implement-stack-linked-list/ [Accessed 29 Dec. 2017].

6.Btechsmartclass.com. (2017). *Stack using Linked List - Data Structures*. [online] Available at: http://btechsmartclass.com/DS/U2_T3.html [Accessed 29 Dec. 2017].

7.Sanfoundry. (2017). *C Program to Implement a Stack using Linked List - Sanfoundry*. [online] Available at: http://www.sanfoundry.com/c-program-stack-using-linked-list/ [Accessed 29 Dec. 2017].

8.GeeksforGeeks. (2017). *Stack Data Structure (Introduction and Program) - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/stack-data-structure-introduction-program/ [Accessed 29 Dec. 2017].

9.Gist. (2017). *Linked List implementation of Queue.*. [online] Available at: https://gist.github.com/mycodeschool/7510222 [Accessed 29 Dec. 2017].

10.GeeksforGeeks. (2017). *Linked List | Set 1 (Introduction) - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/linked-list-set-1-introduction/ [Accessed 29 Dec. 2017].