EX.NO:1	PROBLEM STATEMENT
DATE:05-07-17	

AIM

To write a problem statement for conference management system.

PROBLEM STATEMENT

The conference management system is an online system in which the candidate can register themselves and attend the conference. Candidate should login into the system for viewing the details about the conference; User can select any conference from the listed box. When user selects any conference, the registration form will be displayed. If the user has eligibility to attend that conference, then user must submit confirmation and the payment for attending the conference. The system will send acknowledgement about the confirmation to the user, after these steps the registration process will be done.

RESULT

Thus the problem statement for the conference management system was written successfully.

EX.NO: 2	USE CASE DIAGRAM
DATE:12-07-17	

AIM

To draw use case diagram for the conference management system using Rational Rose.

USE CASE DIAGRAM

A use case diagram is a representation of a user's interaction with the system, and depicts the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. It describes what system does from the standpoint of an external observer. The emphasis is on what a system does rather than how.

NOTATION

USE CASE

Use case is representation as an ellipse with a name inside. It may contain additional responsibility. Use case is used to capture high level functionalities of a system.

Fig2.1. Use case

ACTOR

An actor can be defined as some internal or external entity that interacts with the system. Actor is used in a use case diagram to describe the internal or external entities. A person or organization that is identified by role is called Actor.

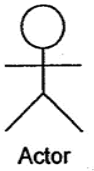


Fig2.2. Actor

There are two kinds of actors

- ✓ Primary actor
- ✓ Supporting actor

PRIMARY ACTORS

Primary actors and their goals invoked in all the use case of recruitment.

SUPPORTING ACTORS

Supporting actors and their goals for all the use cases.

ASSOCIATION

It shows the relation between actor and use case. It consists of extend and include between use case. Extend means is use case optionally requires another use case. Include means same as extend except it is compulsory requires another case.

Fig2.3. Association

DEPENDENCY

A dependency relationship represent that a model element refer an another model element for specification and for implementation.

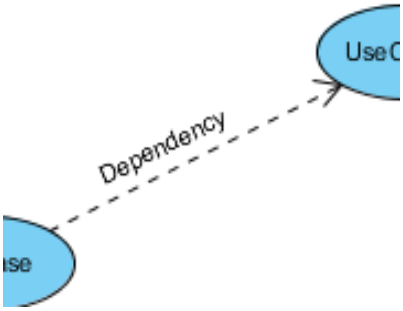


Fig2.4. Dependency

ACTOR'S DOCUMENTATION

- **Candidate:** Login the conference system and submits the paper then do the register process.
- **Admin:** Review the paper, select best candidate and send acknowledgement to them.
- **Registration:** Candidate register their details.
- **Login:** Candidate can login into the system.
- **Conference details:** Admin displays the conference details to the candidate.
- **Availability:** Whether there is a seat is available for the candidate. It there is a availability candidate can apply for conference.
- **Registration:** Candidate registered their conference which they wish to join.
- **Check eligibility:** Admin check the eligibility of the candidate then only allows the candidate.

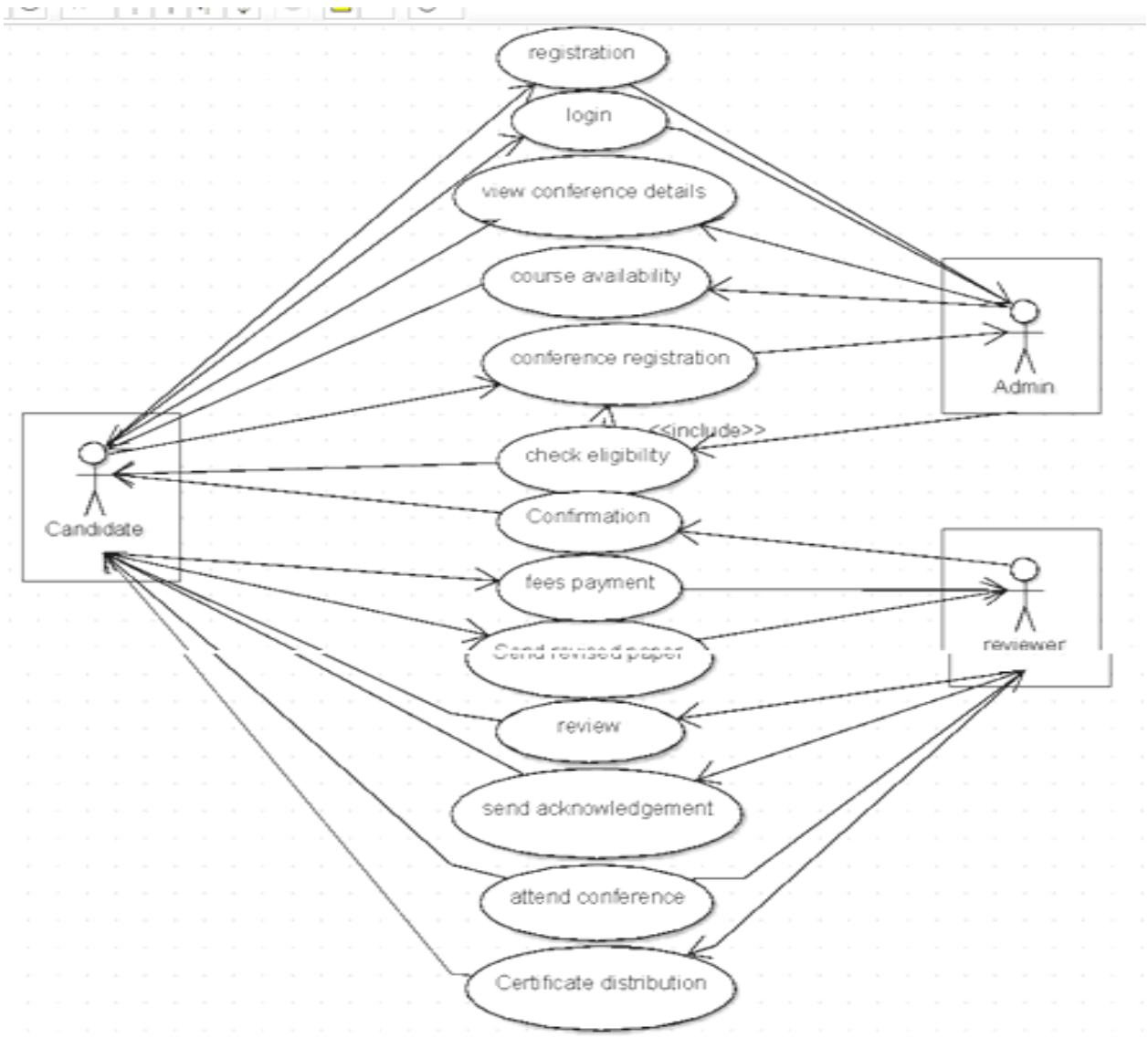


Fig2.5:Use case diagram

- **Fees payment:**Must pay the fees for attending conference.
- **Confirmation:**Admin returns the confirmation to the candidate.
- **Send revised paper:**After the paper is selected and the revised paper should be submitted to the admin by candidate.
- **Review:** Admin reviewed the paper which is submitted by the candidate.
- **Send acknowledgement:**After reviewed the paper, admin send an acknowledgement about the conference.
- **Attend conference:** After these procedure were finished candidate can attend the conference.
- **Certificate distribution:**Admin provides a certificate to the candidates who were attend the conference.

RESULT

Thus the use case diagram for the conference management system was drawn successfully.

EX.NO:3	UML CLASS DIAGRAM
DATE:19-07-17	

AIM

To draw the UML class diagram for conference management system.

UML CLASS DIAGRAM

A class diagram in the unified modeling language (UML) is a type of static structure a diagram that describes the structure of a system by showing the systems classes, their attributes, and the relationships between the classes. It is represented using a rectangle with three compartments. To compartment have the class name, middle compartment the attributes and the bottom compartment with operations. This UML class diagram has three classes' candidate, review and database.

CANDIDATE

Its attribute are name, college name, department, paper title. The operations performed in the candidate class are registration, login, submit the paper, fees payment, and submit revised paper.

ADMIN

Its attributes are name, department, and adminID. The operations performed are view details. Check eligibility, review the paper and send confirmation details and the acknowledgement and a certificate.

DATABASE

The operations performed are storing candidate details and verifying login.

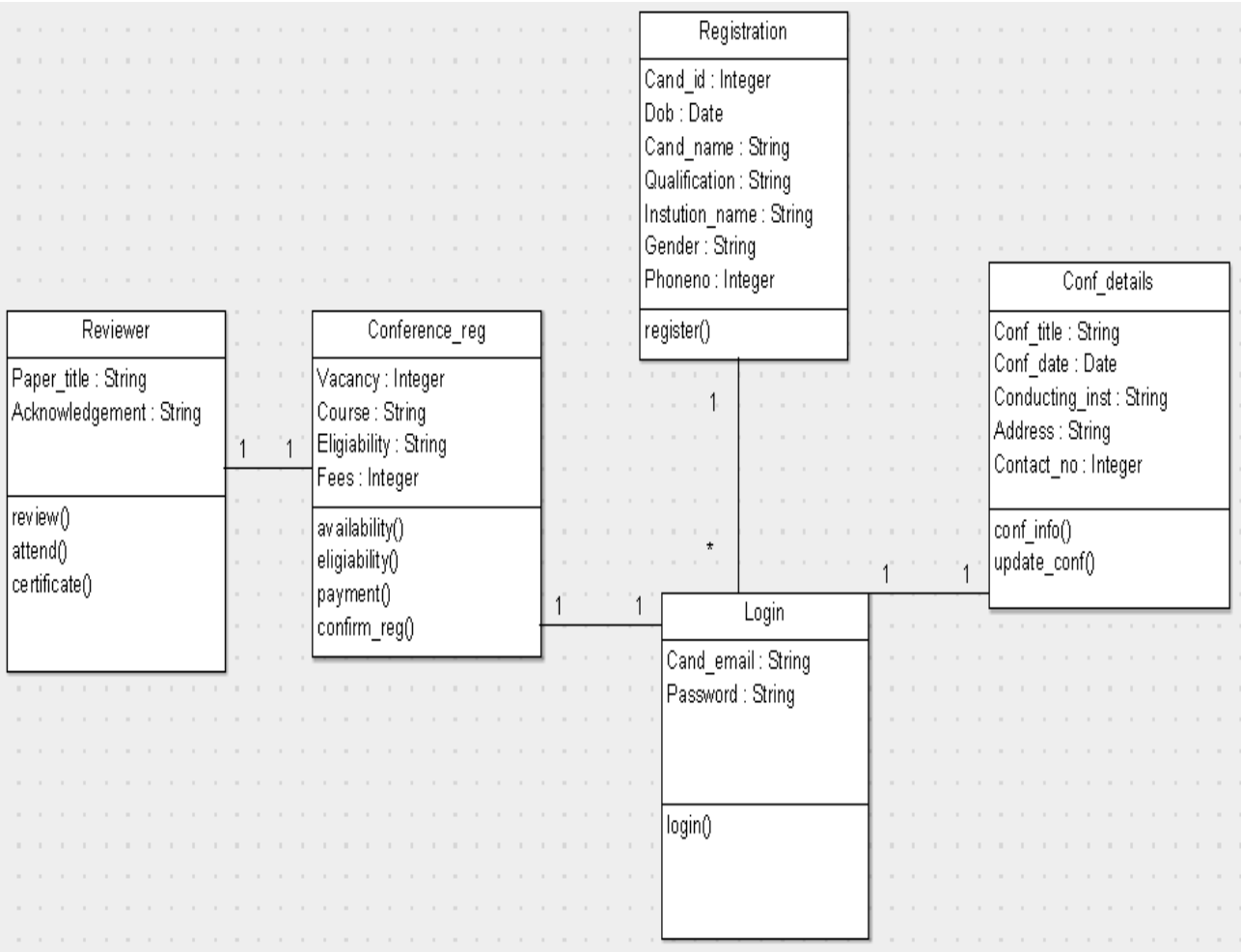


Fig3.1.UML class diagram

RESULT

Thus the UML class diagram for the conference management system was drawn successfully.

EX.NO:4	UML SEQUENCE DIAGRAM
DATE:26-07-17	

AIM

To draw a UML sequence diagram for conference management system.

UML SEQUENCE DIAGRAM

A sequence diagram is the most common kind of interaction diagram, which focus on the message interchange between a numbers of lifelines.

Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged along with their corresponding occurrence specifications on the lifelines.

A sequence diagram illustrates a kind of format in which each object interacts via message. It is generalize between two or more specialized diagram.

OBJECT

This box shape represents a class or object, in UML. They demonstrate how an object will behave in the context of the system, class attributes should not be listed in this shape.

ACTIVTION BONES

Symbolized by rectangle shape. It represents the time needed for an object to complete a table.

ACTORS

It represented by a stick figure, actors are entities that are both interactive with and external to the system.

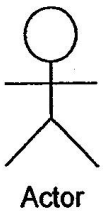


Fig4.1. Actors

PACKAGE

It is also known as frame, it contain interactive elements of the diagram, the shape has is small inner rectangle for labeling the diagram.

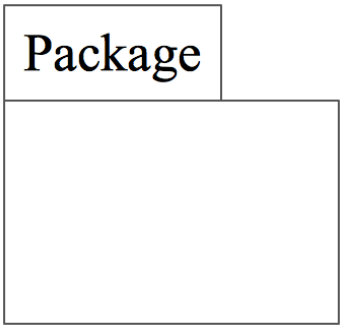


Fig4.2. Package

LIFELINES

A dashed vertical line that represents the passage of time as it extends downed. Along with time they represents the sequential events that occur to an object during the charted process. It may begin with a labeled rectangle shape or an actor symbol.

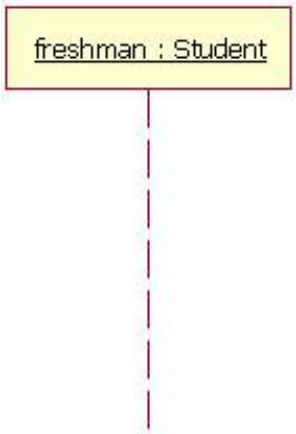


Fig4.3. Life Line

MESSAGE

- **Synchronous messages:** The diagram should show both call and reply.
- **Asynchronous messages:** The diagram should show only call.
- **Asynchronous returned messages:** It represented by a dashed line with a linked arrowhead.
- **Create message:** These messages are sent to lifelines in order to create themselves.
- **Reply messages:** There are replies to call.
- **Deleted messages:** This indicates the destruction of an object and is placed in its path on the lifelines.

EXECUTION

It is interaction fragment which represents a period in the participant's lifetime when it is executing unit of behavior or action with in the lifelines.

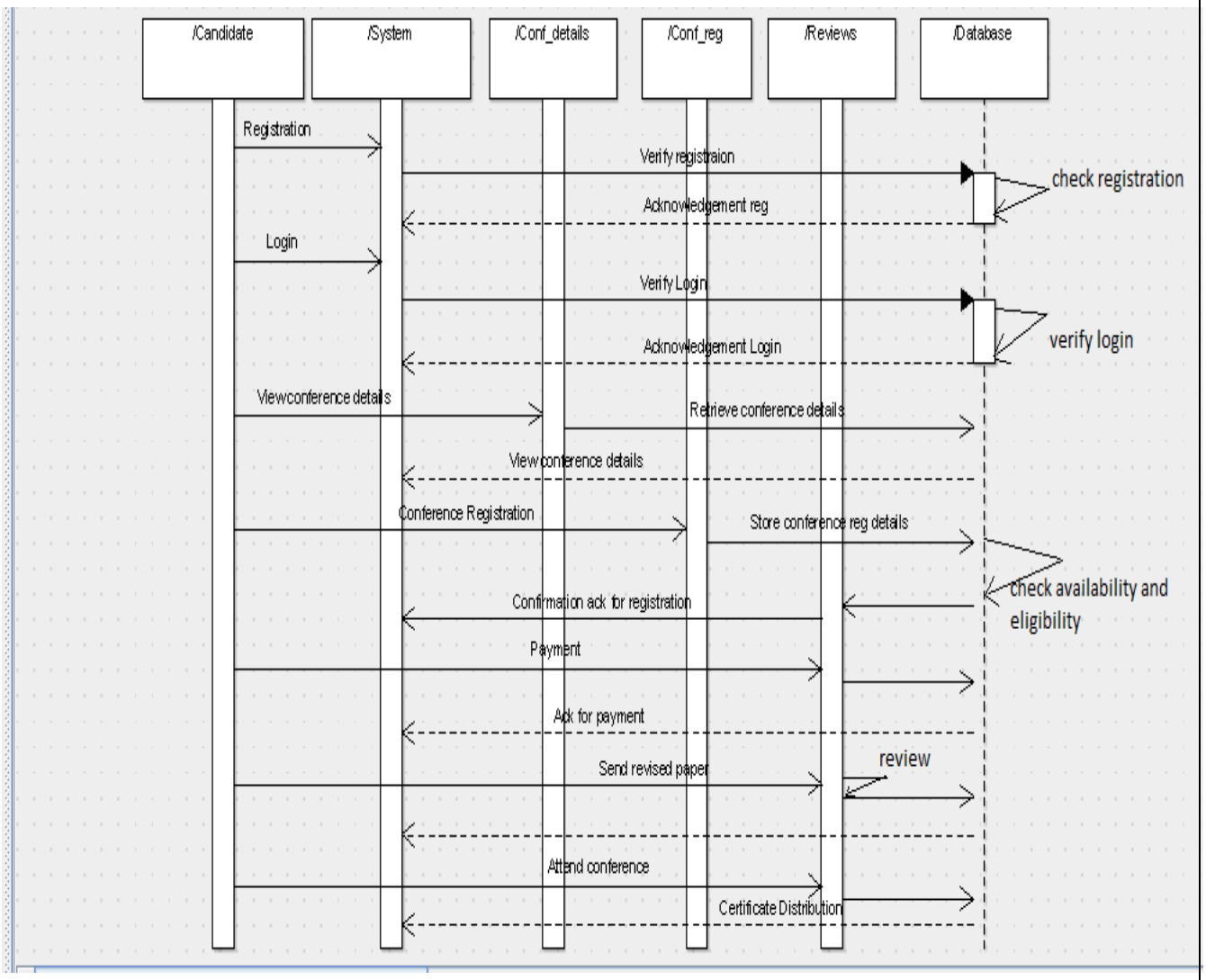


Fig4.4. UML sequence diagram

RESULT

Thus the sequence diagram for conference management system was drawn successfully.

EX.NO:5	UML STATE CHART DIAGRAM
DATE:02-08-17	

AIM

To draw state chart for conference management system.

STATE CHART DIAGRAM

State chart diagram is used to model dynamic nature of a system. They define different state of an object during its lifetime, and their states are changed by events. So it is useful to model reactive systems.

MAIN PURPOSE OF USING STATE CHART DIAGRAM

I. To model dynamic aspect of a system.
II. To model lifetime of a reactive system.
III. To describe different states of an object during its life time.
IV. Define a state machine to model states of an object.

COMPONENTS

STATES

A state is a condition during the life of an object or an interaction during which it satisfy some condition, performs some action or waits for some events.

Fig5.1. States

SUBMACHINE STATE

It is a syntactical convenience that facilities reuse and modularity. It is shorthand that implies a macro like expansion by another state machine and is semantically equivalent to a composite state.

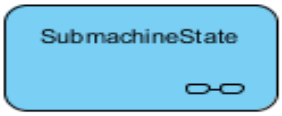


Fig5.2. Submachine State

INITIAL STATE

It is a kind of pseudo state that represents the starting point in a region of a state machine. It has a single enclosing region, and has no incoming transitions.

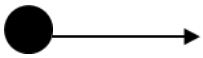


Fig5.3. Initial State

FINAL STATE

It represents “last” state of enclosing composite state. When a final state is reached and there are no other enclosing states it means that the entire state machine has completed its transitions.

Fig5.4. Final State

TRANSITION

It is a directed relationship between a source state vertex and a target state vertex. It takes state machine from one state configuration to another, representing the complete response of the state machine to particular event instance.

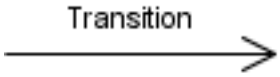


Fig5.5. Transition

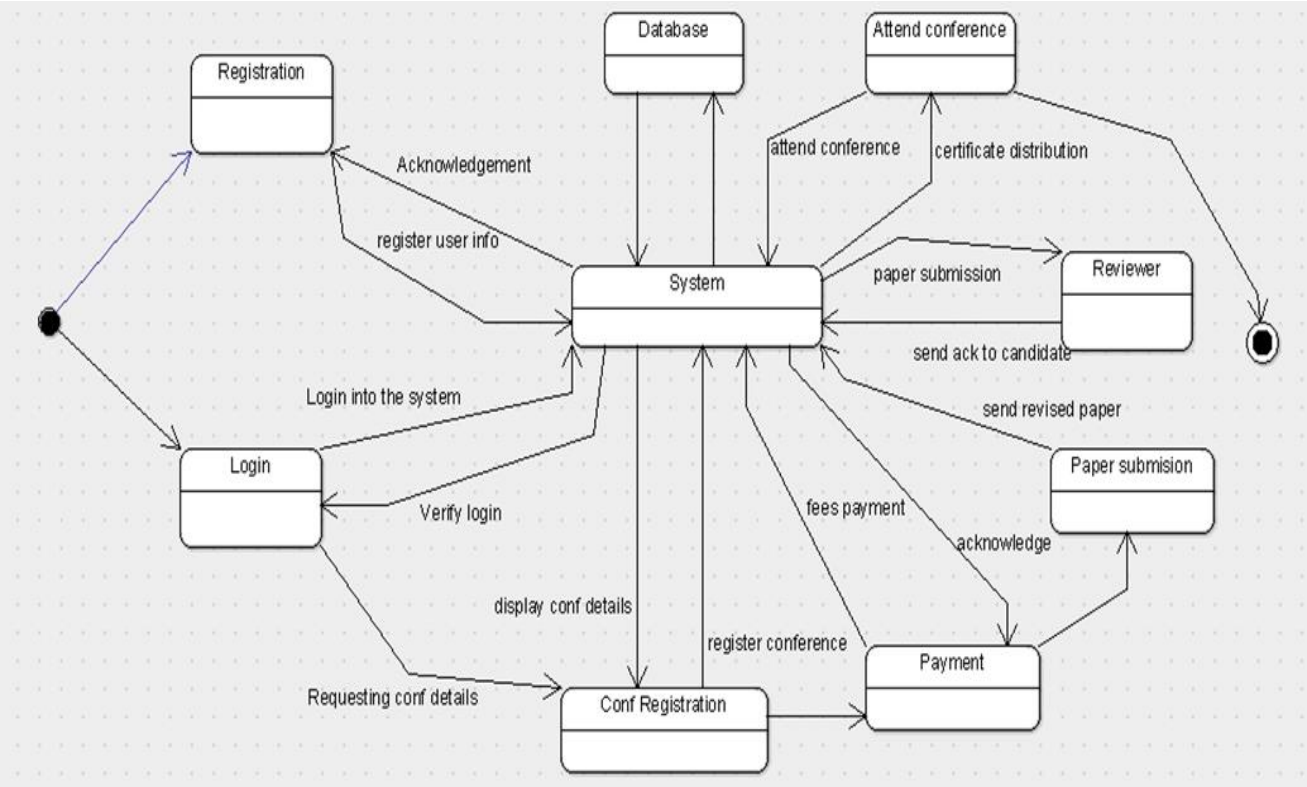


Fig5.6.UML state chart and activity diagram

STATE CHART DIAGRAM EXPLANATION

In conference management system candidate register and login into the system to view the conference details. If they want to register, they can register using course registration. If the seat vacancy is available, then the admin send confirmation to the candidate otherwise admin reject the registration.

Candidate send a revised paper to the reviewer. The reviewer reviews the paper, if it is accepted then the send acknowledgement otherwise the paper will be rejected.

After receiving acknowledgement candidate must pay the amount and then he is allowed to attend the conference.

RESULT

Thus the state chart diagram for the conference management system was drawn successfully.

EX.NO:6	UML ACTIVITY DIAGRAM
DATE:02-08-17	

AIM

To draw activity for conference management system.

ACTIVITY DIAGRAM

Activity diagram is basically a flow chart to represent the flow from one Activity to another. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential branched or concurrent.

CONTENTS

Initial/final state, activating, fork and join, branch, swim lanes.

ACTION STATE

It represents the execution of an atomic action, typically the invocation of an operation.

Fig6.1. Action State

SUB ACTIVITY STATE

It represents the execution of a non-atomic sequence of steps that has some duration. It consists of a set of actions and possibly waiting for events. It is hierarchical action, where an associated sub activity graph is executed.

INITIAL STATE

It has a single outgoing transition to default state of an enclosing region and has no end coming transitions.

Fig6.2. Initial Sate

FINAL STATE

It represents the last state of the enclosing composite state. There may be more than one final state at any level signifying that the composite state can end in different ways of conditions.

Fig6.3. Final Sate

FORK

It represents the splitting of a single flow of control into two or more concurrent flow of control. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control.

JOIN

A join represents the synchronization of two or more concurrent flows of control. If join may have two or more incoming and one outgoing transitions.

BRANCHING

If specifies alternative paths takes based or some Boolean. Expression branch is represented by diamond. Branch may have one incoming transition and two or more outgoing one on each outgoing transition.

SWIM LANE

Swim lanes are useful when we model workflows of business processes to partition the activity states on an activity diagram into groups.

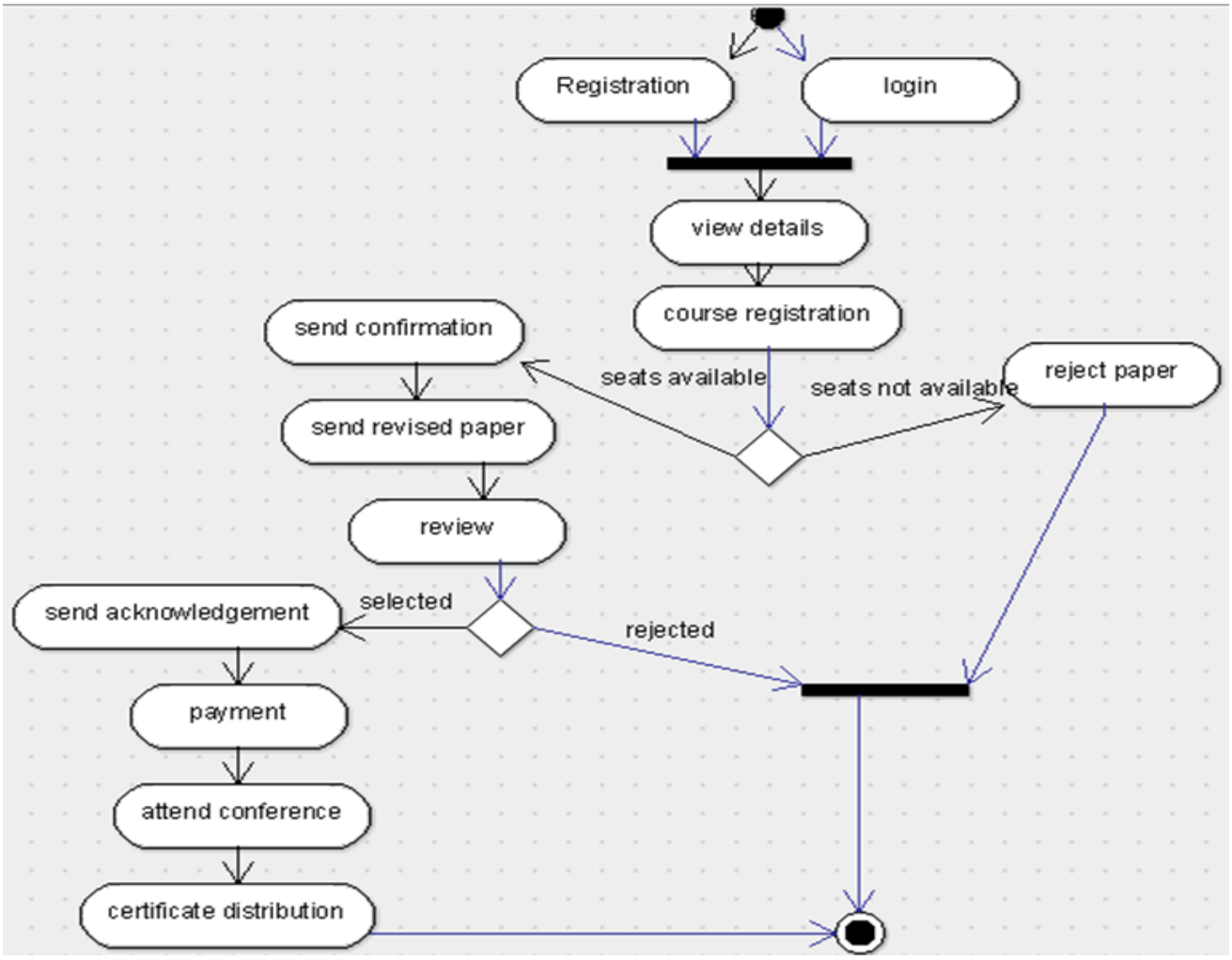


Fig6.4.UML state chart and activity diagram

ACTIVITY DIAGRAM EXPLANATION

In conference management system registration, login, course registration are the action states. Send confirmation, send revised paper, review, and payment, sending acknowledgement are the sub activity state. If seat is available, then we can register otherwise it will be rejected. When all the functions were performed then it reaches a final state.

RESULT

Thus the activity diagram for the conference management system was drawn successfully.

EX.NO:7	UML PACKAGE DIAGRAM
DATE:23-08-17	

AIM

To identify the user interface, domain objects and technical services. Draw the partial layered logical architecture diagram with UML package diagram.

PACKAGE DIAGRAM

Package diagram organize the elements of a system into related groups to minimize dependencies among them. The entire system can be thought of as a single high-level package, with all the UML diagrams organized within it. A package may contain both subordinate package and ordinary model elements. All UML models and diagrams are organized into package. Packages appear as rectangles with small tabs at the top. The package name is on the tab or inside the rectangle. The dotted arrows are dependencies. One package depends on another if changes in the other cloud possibly force changes in the first. There are three types of layers,

- ✓ User interface layer
- ✓ Domain layer
- ✓ Technical services layer

USER INTERFACE LAYER

This layer provides the user interface (UI) within a composite application.

DOMAIN LAYER

A domain layer also knows as the business logic layer (BLL) is a software engineering practice of compact metalizing. It separates the business logic from other models, such as the data access layer and user interface.

TECHNICAL SERVICES LAYER

Technical services subsystems that support low-level functions, such as interfacing with a database, external hardware or error logging. In strict layering, a layer only calls upon services in the layer directly below it.

The figure in the next page represents the partial layered architecture diagram for the conference management system.

- ✓ **User interface:** It has login pages, etc.
- ✓ **Domain objects:** It has all the objects of the system such as Admin. Reviewer and candidate etc.
- ✓ **Technical services:** It has the database of the conference management system.

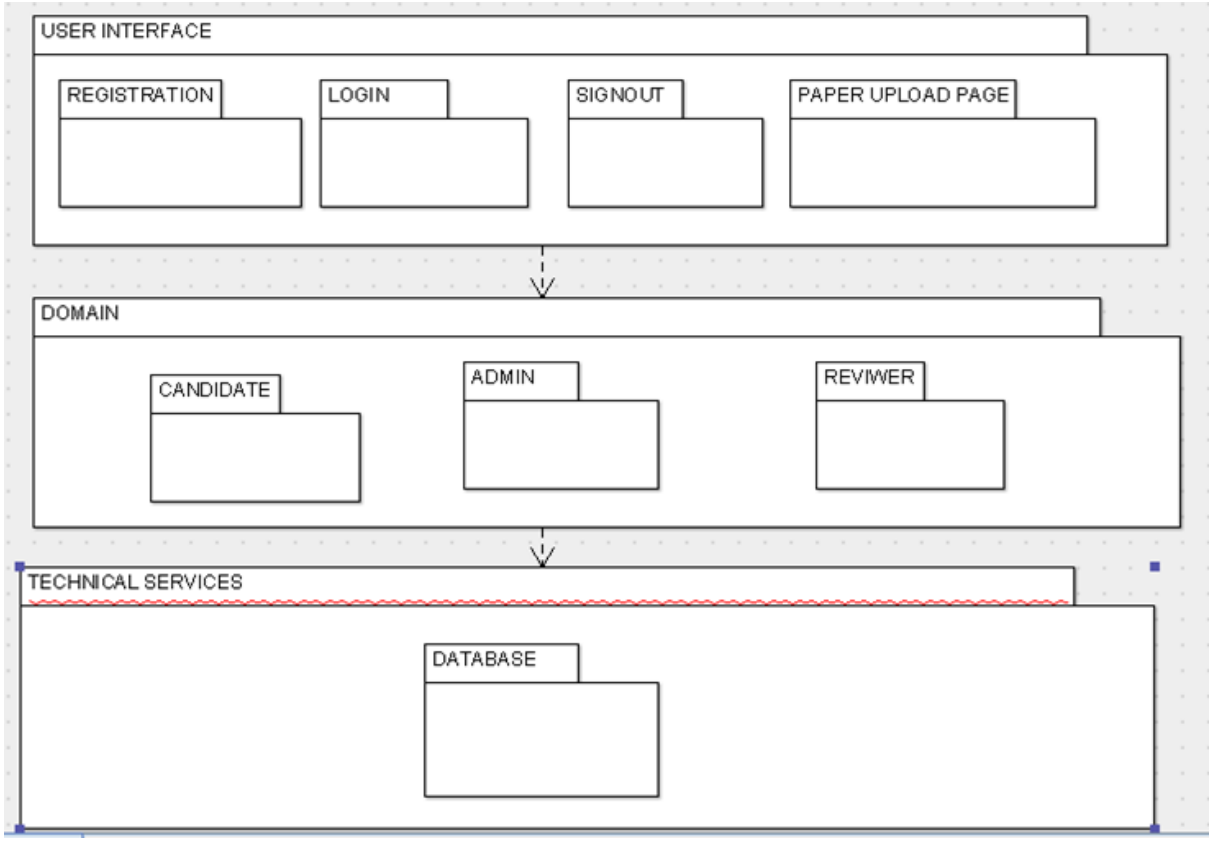


Fig7.1. Package diagram

RESULT

Thus the user interface, domain objects and technical services were identified and partial layered, logical architecture diagram with UML package diagram notation was drawn.

EX.NO:8	USER INTERFACE LAYER
DATE:30-08-17	

AIM

To implement User Interface Layer for Conference Management System.

USER INTERFACE MODELING

User interface modeling is a development technique used by computer application programmers. Today's user interfaces (UIs) are complex software components, which play an essential role in the usability of an application. The development of UIs requires therefore, not only guidelines and best practice reports, but also a development process including the elaboration of visual models and a standardized notation for this visualization.

The term user interface modeling is mostly used in an information technology context. A user interface model is a representation of how the end user(s) interact with a computer program or another device and also how the system responds. The modeling task is then to show all the "directly experienced aspects of a thing or device".

Modeling user interfaces is a well-established discipline in its own right. For example, modeling techniques can describe interaction objects, tasks, and lower-level dialogs in user interfaces. Using models as part of user interface development can help capture user requirements, avoid premature commitment to specific layouts and widgets, and make the relationships between an interface's different parts and their roles explicit.

USER INTERFACE ANALYSIS

- Create user interface realization
- Identify candidate UI elements
- Model role-boundary interaction
- Re-factor UI responsibilities
- Model UI navigation

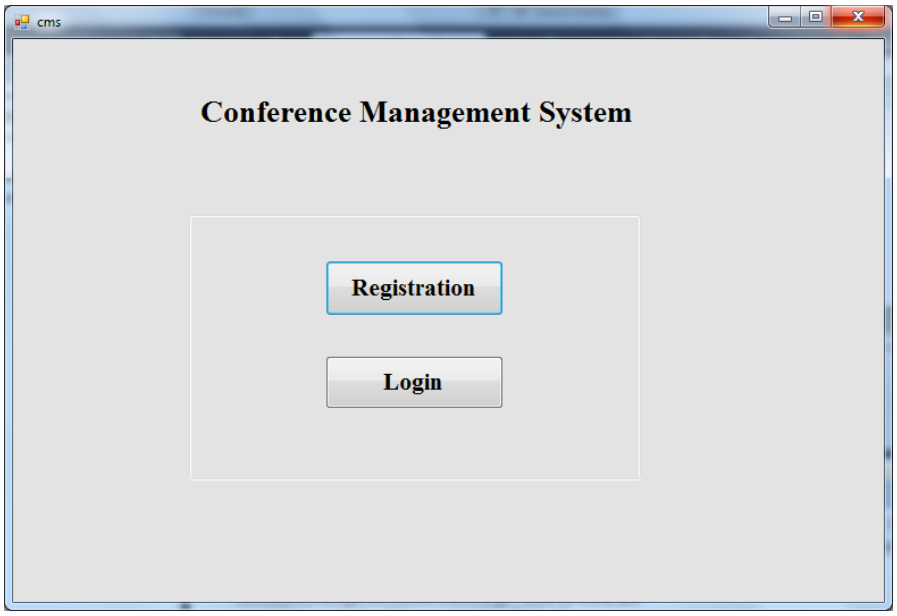
USER INTERFACE DESIGN

- Platform-specific model
- Influenced by UI architecture, standards and conventions, and platform constraints
- Many possible design models for one analysis model

USER INTERFACES FOR CONFERENCE MANAGEMENT SYSTEM:

FORMS:

CMS FORM

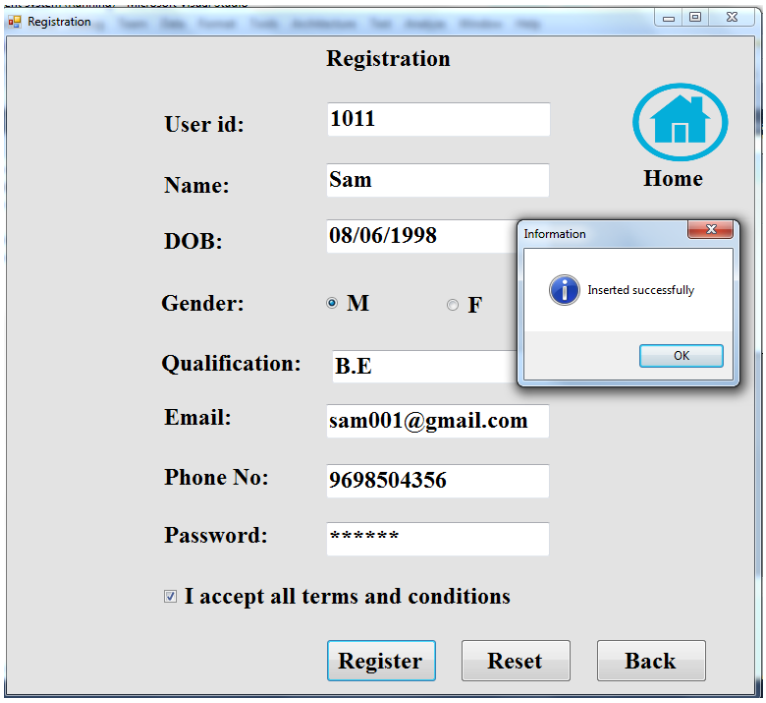


CODING

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

namespace Conference_management_system
{
   public partial class CMS : Form
   {
      public CMS()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Registration r = new Registration();
         r.Show();
         this.Hide();

      }

      private void button2_Click(object sender, EventArgs e)
```

```
        {
            Logintypes lt = new Logintypes();
            lt.Show();
            this.Hide();
        }

    }
}
```

REGISTRATION FORM



CODING

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

namespace Conference_management_system
{
    public partial class Registration : Form
    {
```

```
        OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\VELMURUGAN\Pictures\cmsdb.accdb");
        string gen;
        public Registration()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (checkBox1.Checked)
            {

                if (radioButton1.Checked == true)
                {
                    gen = "Male";

                }
                else if (radioButton2.Checked == true)
                {
                    gen = "Female";

                }
                con.Open();

                OleDbCommand cmd = new OleDbCommand();
                cmd.CommandType = CommandType.Text;
                cmd.Connection = con;
                cmd.CommandText = "insert into registration values('" + textBox1.Text + "','" +
textBox2.Text + "','" + textBox3.Text + "','" + gen + "','" + comboBox1.Text + "','" +
textBox4.Text + "','" + textBox5.Text + "','" + textBox6.Text + "') ";
                cmd.ExecuteNonQuery();

                con.Close();
                MessageBox.Show("Inserted
successfully","Information",MessageBoxButtons.OK,MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("Pls Accept T&C", "Alert", MessageBoxButtons.RetryCancel,
MessageBoxIcon.Error);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
```

```
            gen = "";
            textBox4.Text = "";
            comboBox1.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
        }

        private void button3_Click_1(object sender, EventArgs e)
        {
            CMS c = new CMS();
            c.Show();
            this.Hide();
        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {
            CMS c = new CMS();
            c.Show();
            this.Hide();
        }

    }
}
```

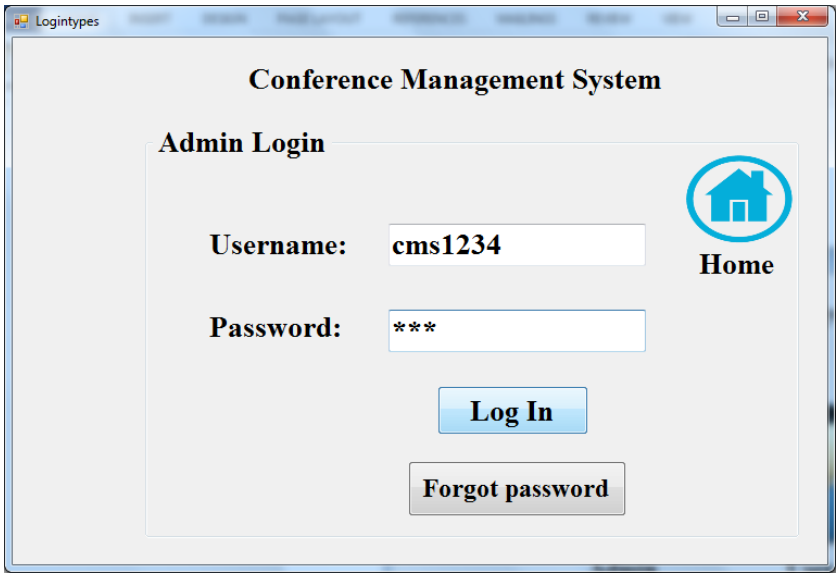
LOGINTYPES FORM

CODING

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

```
using System.Data.OleDb;

namespace Conference_management_system
{
   public partial class Logintypes : Form
   {
      public Logintypes()
      {
         InitializeComponent();
      }

      private void pictureBox1_Click(object sender, EventArgs e)
      {
        Adminlogin al = new Adminlogin();
        al.Show();
        this.Hide();
      }

      private void pictureBox2_Click(object sender, EventArgs e)
      {
         Login l = new Login();
         l.Show();
         this.Hide();
      }

      private void pictureBox3_Click(object sender, EventArgs e)
      {
         CMS c = new CMS();
         c.Show();
         this.Hide();
      }
   }
}
```

ADMIN LOGIN FORM

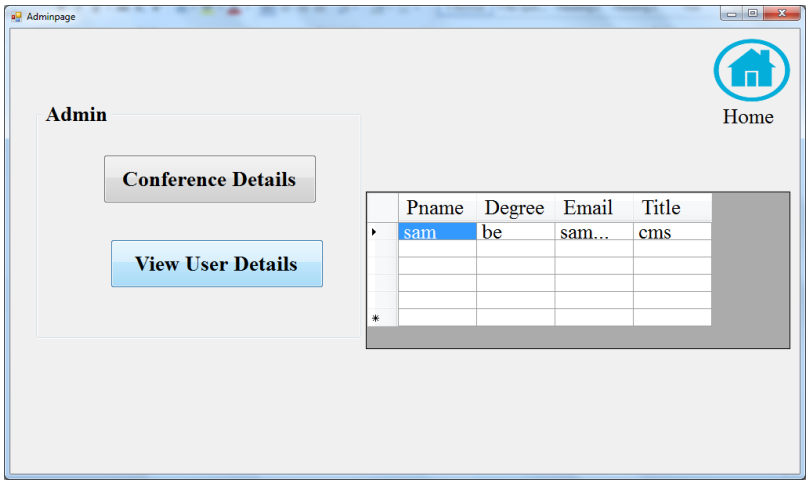


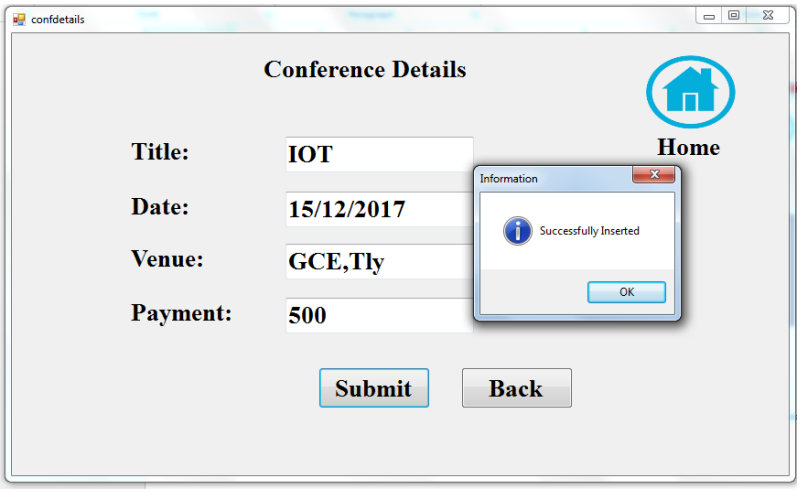
CODING

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

namespace Conference_management_system
{
   public partial class Adminlogin : Form
   {
      public static string Username;
      public Adminlogin()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\VELMURUGAN\Pictures\cmsdb.accdb");
         con.Open();

         try
         {
            OleDbCommand cmd = new OleDbCommand("select * from admin where User =
@Username", con);
            cmd.Parameters.Add(new OleDbParameter("@Username", textBox1.Text));

            OleDbDataReader dr = cmd.ExecuteReader();
            dr.Read();
            string pass = textBox2.Text;
            string passdb = dr[1].ToString();
            Username = textBox1.Text;
            if (pass == passdb)
            {
               Adminpage ad = new Adminpage();
               ad.Show();
               this.Hide();
            }
            else
            {
               MessageBox.Show("Invalid password");
               textBox2.Focus();
            }
```

```
        }
        catch (Exception)
        {
            MessageBox.Show("Invalid Username");
            textBox1.Focus();
        }
        con.Close();
    }
  private void pictureBox3_Click(object sender, EventArgs e)
    {
        CMS c = new CMS();
        c.Show();
        this.Hide();
    }

  }}
```

ADMIN PAGE



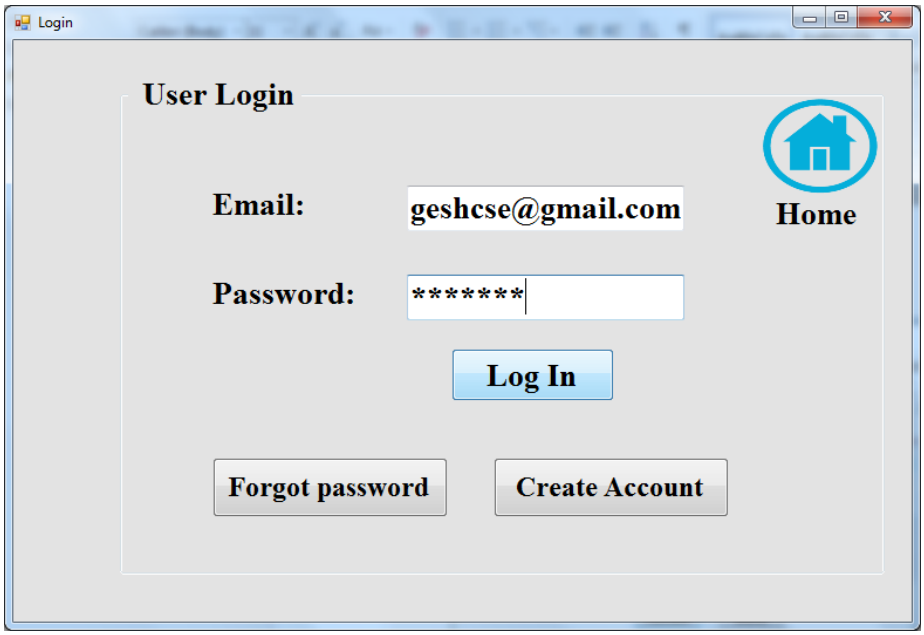


CODING

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

namespace Conference_management_system
{
   public partial class Adminpage : Form
   {
      OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\VELMURUGAN\Pictures\cmsdb.accdb");
      public Adminpage()
      {
         InitializeComponent();
      }

      private void button2_Click(object sender, EventArgs e)
      {
         con.Open();
         OleDbCommand cmd = con.CreateCommand();
         cmd.CommandType = CommandType.Text;
         cmd.CommandText = " select Pname,Degree,Email,Title from papers  ";
         cmd.ExecuteNonQuery();
         OleDbDataAdapter da = new OleDbDataAdapter(cmd);
         DataTable dt = new DataTable();
         da.Fill(dt);

         dataGridView1.DataSource = dt;

         con.Close();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         confdetails cd = new confdetails();
         cd.Show();
         this.Hide();
      }

      private void pictureBox3_Click(object sender, EventArgs e)
      {
         CMS c = new CMS();
         c.Show();
```

```
            this.Hide();
        }
    }
}
```

USER LOGIN

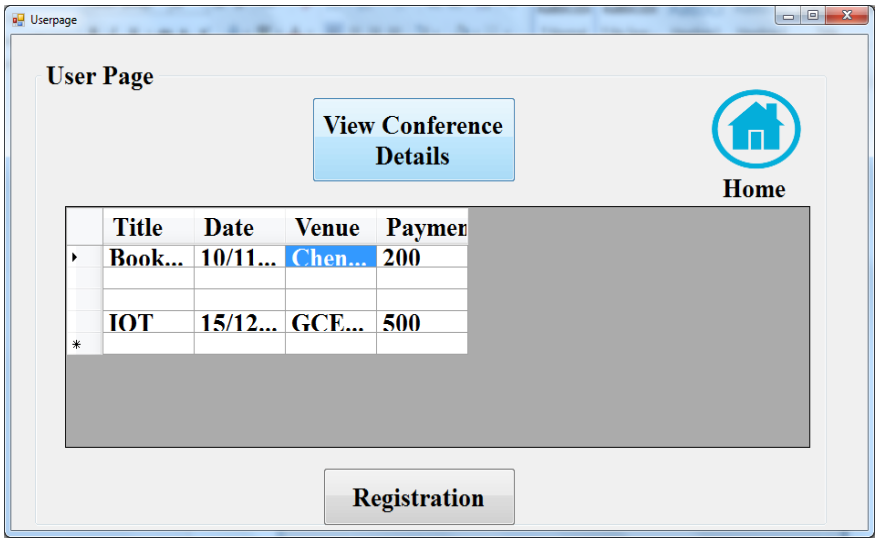


CODING

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

namespace Conference_management_system
{
    public partial class Login : Form
    {

        public static string email;
        public Login()
        {
            InitializeComponent();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Registration r = new Registration();
```

```
            r.Show();
            this.Hide();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\VELMURUGAN\Pictures\cmsdb.accdb");
            con.Open();

            try
            {
                OleDbCommand cmd = new OleDbCommand("select * from registration where
Email = @email",con);
                cmd.Parameters.Add(new OleDbParameter("@email", textBox1.Text));

                OleDbDataReader dr = cmd.ExecuteReader();
                dr.Read();
                string pass = textBox2.Text;
                string passdb = dr[7].ToString();
                email = textBox1.Text;
                if (pass == passdb)
                {
                    Userpage up = new Userpage();
                    up.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Invalid password");
                    textBox2.Focus();
                }

            }
            catch (Exception)
            {
                MessageBox.Show("Invalid email id");
                textBox1.Focus();
            }
            con.Close();
        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {
            CMS c = new CMS();
            c.Show();
            this.Hide();
        }
    }}
```

USER PAGE FORM

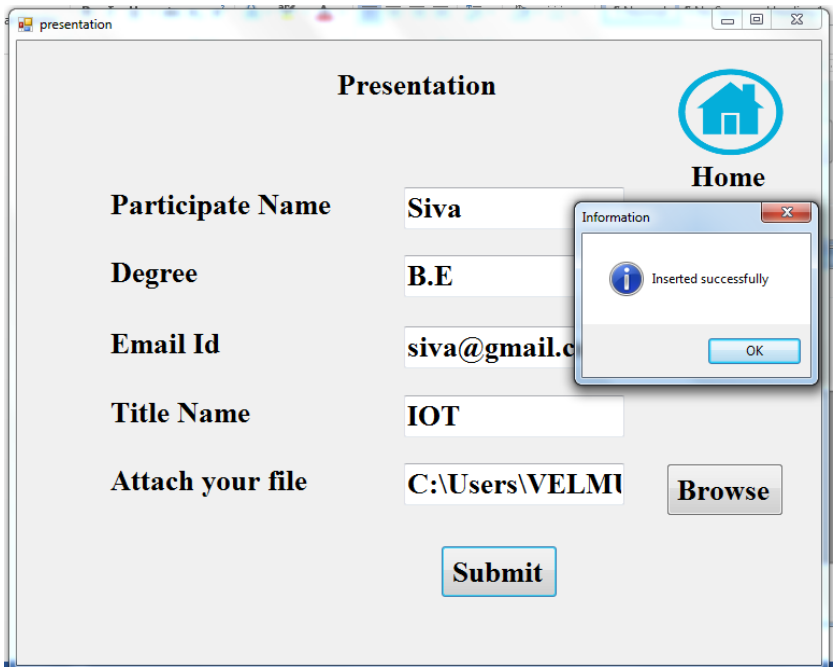


CODING

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

namespace Conference_management_system
{
   public partial class Userpage : Form
   {
      OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\VELMURUGAN\Pictures\cmsdb.accdb");
      public Userpage()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         con.Open();
         OleDbCommand cmd = con.CreateCommand();
         cmd.CommandType = CommandType.Text;
         cmd.CommandText = "select * from confdetails";
         cmd.ExecuteNonQuery();
         OleDbDataAdapter da = new OleDbDataAdapter(cmd);
         DataTable dt = new DataTable();
```

```
            da.Fill(dt);

            dataGridView1.DataSource = dt;

            con.Close();

        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {
            CMS c = new CMS();
            c.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            presentation p = new presentation();
            p.Show();
            this.Hide();
        }

    }
}
```

PRESENTATION FORM



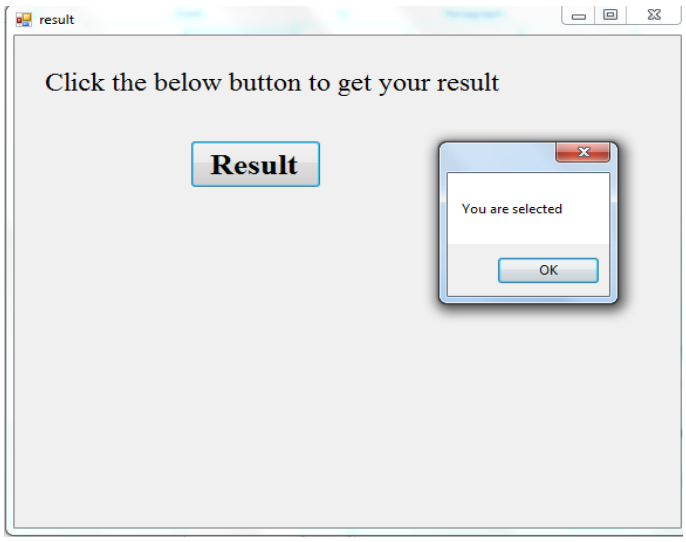
CODING

```
using System;
```

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

namespace Conference_management_system
{
   public partial class presentation : Form
   {
      OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\VELMURUGAN\Pictures\cmsdb.accdb");
      public presentation()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {

         con.Open();

         OleDbCommand cmd = new OleDbCommand();
         cmd.CommandType = CommandType.Text;
         cmd.Connection = con;
         cmd.CommandText = "INSERT INTO papers values('" + textBox1.Text + "','" +
textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text + "','" + textBox5.Text + "') ";
         cmd.ExecuteNonQuery();


         con.Close();
         MessageBox.Show("Inserted successfully", "Information", MessageBoxButtons.OK,
MessageBoxIcon.Information);



         MessageBox.Show("Successfully Upload");
         result r = new result();
         r.Show();
         this.Hide();
      }

      private void button3_Click(object sender, EventArgs e)
      {

         OpenFileDialog openFileDialog1 = new OpenFileDialog();
```

```
            openFileDialog1.InitialDirectory = @"C:\";
            openFileDialog1.Title = "Browse Text Files";

            openFileDialog1.CheckFileExists = true;
            openFileDialog1.CheckPathExists = true;

            openFileDialog1.DefaultExt = "txt";
            openFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 2;
            openFileDialog1.RestoreDirectory = true;

            openFileDialog1.ReadOnlyChecked = true;
            openFileDialog1.ShowReadOnly = true;



            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                textBox5.Text = openFileDialog1.FileName;


            }
        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {
            CMS c = new CMS();
            c.Show();
            this.Hide();
        } }
}
```

RESULT FORM



CODING

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

namespace Conference_management_system
{
   public partial class result : Form
   {
      public result()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         MessageBox.Show("You are selected");
      }
   }
}
```

RESULT

Thus the user interface for the conference management system was constructed successfully.

EX.NO:9	DOMAIN LAYER
DATE:06-09-17	

AIM

To implement Domain objects layer for the Conference Management System.

DOMAIN LAYER

A **domain model** in problem solving and software engineering is a conceptual model of all the topics related to a specific problem. It describes the various entities, their attributes, roles, and relationships, plus the constraints that govern the problem domain.

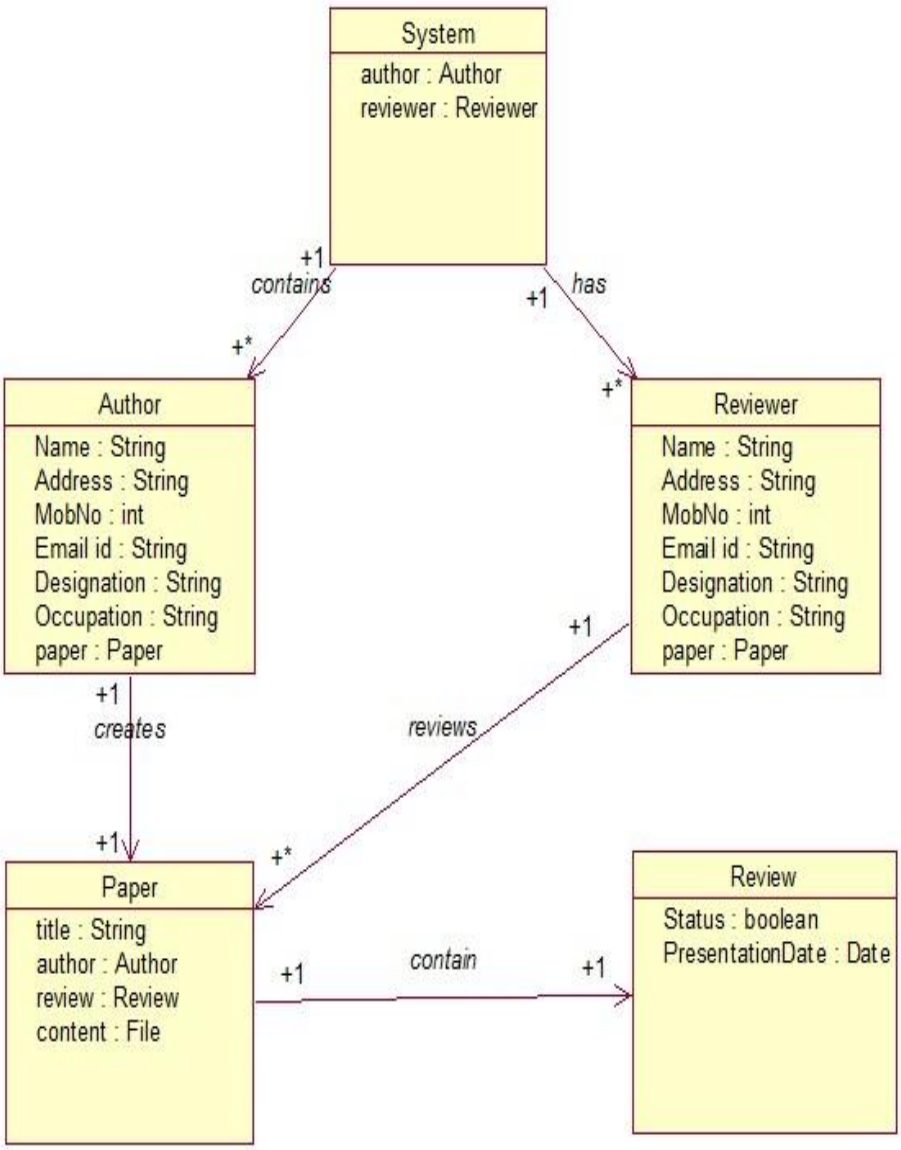
OVERVIEW

The domain model is created in order to represent the vocabulary and key concepts of the problem domain. The domain model also identifies the relationships among all the entities within the scope of the problem domain, and commonly identifies their attributes. A domain model that encapsulates methods within the entities is more properly associated with object oriented models. The domain model provides a structural view of the domain that can be complemented by other dynamic views, such as use case models.

An important advantage of a domain model is that it describes and constrains the scope of the problem domain. The domain model can be effectively used to verify and validate the understanding of the problem domain among various stakeholders. It defines a vocabulary and is helpful as a communication tool. It can add precision and focus to discussion among the business team as well as between the technical and business teams

DRAWING DOMAIN MODEL

In domain-driven design, the Domain Model (domain entities and actors) covers all layers involved in modelling a business domain, including (but not limited to) Service Layer, Business Layer, and Data Access Layer thus ensuring effective communication at all levels of engineering. It is considered an effective tool for software development, especially when domain knowledge is iteratively provided by domain experts (such as Business Analysts, Subject Matter Experts and Product Owners.) A domain is a collection of related concepts, relationships, and workflows.



RESULT

Thus the domain model for the conference management system was constructed successfully.

EX.NO:10	TECHNICAL SERVICE LAYER	35
DATE:13-09-17		

AIM

To implement the Technical Services Layer for the Conference Management System.

DATABASE

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker.

- **Data definition** – Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- **Update** – Inserting, modifying, and deleting data.
- **Retrieval** – Obtaining information either for end-user queries and reports or for processing by applications.
- **Administration** – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control.

REGISTRATION

confdetails | papers | registration | admin

ID	Name	DOB	Gender	Qualificatio	Email	Phone no	Password
1031	ravi	7/11/1997	Male	Diploma	rainaravi@gma	9876543210	jfjfjd
1044	sam	9/9/1998	Male	B.Sc	sam009@gmail	9698785500	sami123
1026	manikandan	8/8/1998	Male	Diploma	maninm@gmai	9876543456	maninm
31	Mugesh	4/6/1998	Male	B.E	tmugeshcse@g	9698503234	tmugesh
1000	dnnd	9/9/1999	Male	B.Sc	dhdhhd	8888888	dddd
1011	Sam	8/6/1998	Male	B.E	sam001@gmail	9698504356	Mugesh
0							

CONFERENCE DETAILS

confdetails | papers | registration | admin

Title	Date	Venue	Payment
Book bank	10/11/2017	Chennai	200
IOT	15/12/2017	GCE,Tly	500

PAPER DETAILS

confdetails | papers | registration | admin

Pname	Degree	Email	Title	File
sam	be	sam@gmail.co	cms	ong binary data
Siva	B.E	siva@gmail.co	IOT	ong binary data

RESULT

Thus the technical service layer for conference management system was constructed successfully.