Ahsanullah University of Science & Technology
Department of Computer Science & Engineering

Information Pool System

CSE 3224
Information System Design
&
Software Engineering Lab

Submitted By:

Amreen Tarannum Alam	16.02.04.007
Mayeesha Humaira	16.02.04.008
Shimul Paul	16.02.04.014
Ashna Nawar Ahmed	16.02.04.024

Date of Submission: **15 September, 2019**

Introduction

The information pool is an online site that will provide news and blogs on various topics, alongside ads. Our aim is to analyze and design the ER Diagram, Relational Model, and Basic Front End of our project, Information Pool System.

Names of Entity with Primary Key, Foreign Key or Composite Key

- **User(UserDetails)**: UserId(PK)
- **Blogger**: BloggerId(PK), UserId(FK)
- **Admin**: AdminId(PK)
- **LoginDetails**: LoginId(PK), UserId(FK)
- **Blogs**: BlogId(PK), BloggerId(FK), AdminId(FK)
- **BlogTags**: BlogId(PK)
- **News**: NewsId(PK), AdminId(FK)
- **Ads**: AdsId(PK), UserId(FK), AdminId(FK)
- **Complaint:** ComplaintId(PK), UserId(FK)
- **Complaint_Admins:** ComplaintId+AdminId(PK)
- **Events**: EventId(PK), AdminId(FK)
- **Like_Comment**: UserId+BlogId(PK)

ER Diagram

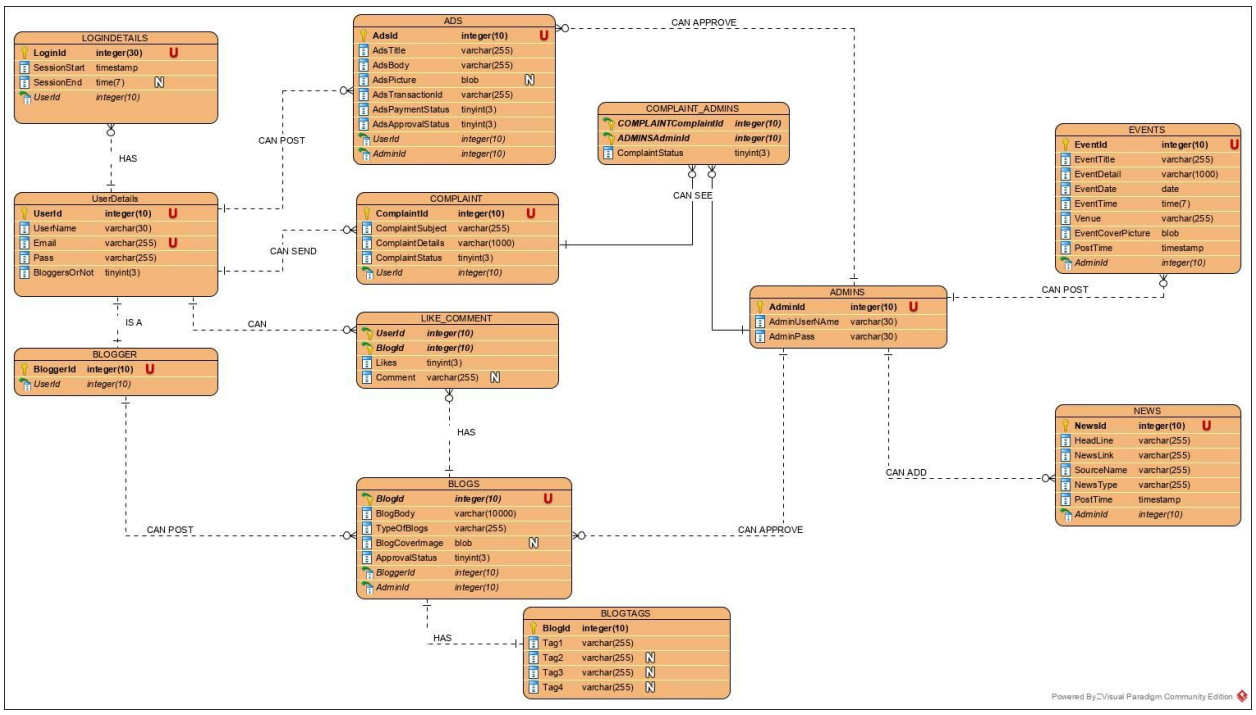


Illustration:

- **Login details**
  Each user has zero or many login details
- **Ads**
  Each user can post zero or many ads
  Each admin can approve zero or many ads
- **Complaints**
  Each user can send zero or many complaints
- **Like_comment**
  Each user can give zero or many likes and comments
  Each blog can have zero or many likes and comments
- **Blogger**
  Each user can be assigned as a blogger
- **Blogs**
  Each blogger can post zero or many blogs
  Each admin can approve zero or many blogs
- **BlogTags**
  Each blog can have one blogtag
- **News**
  Each admin can add zero or many news
- **Events**
  Each admin can post zero or many events
- **Complaint_Admins**
  Each admin can see zero or many complaint_admins
  ;t Each complaint can see zero or many complaint_admins

Relational Model

➢ SQL Commands

```
CREATE DATABASE ISDFINAL;
USE ISDFINAL;
/*UserDetails TABLE*/
CREATE TABLE UserDetails (
UserId INT IDENTITY(1,1) UNIQUE ,
UserName VARCHAR (30) NOT NULL,
Email VARCHAR (255) NOT NULL,
Pass VARCHAR (255) NOT NULL,
BloggerOrNot TINYINT NOT NULL DEFAULT 0,

PRIMARY KEY (USERID)
);
```

```
/*LOGINDETAILS TABLE*/
CREATE TABLE LOGINDETAILS (
LoginId INT IDENTITY(1,1) UNIQUE,
SessionStart TIMESTAMP NOT NULL,
SessionEnd TIME NULL,
UserId INT NOT NULL,

PRIMARY KEY(LoginId),
FOREIGN KEY(UserId) REFERENCES  UserDetails (UserId)
);

/*BLOGGER TABLE*/
CREATE TABLE BLOGGER(
BloggerId INT IDENTITY(1,1) UNIQUE,
UserId INT NOT NULL

PRIMARY KEY(BloggerId),
FOREIGN KEY (UserId) REFERENCES UserDetails(UserId)
);

/*ADMINS TABLE*/
CREATE TABLE ADMINS(
AdminId INT IDENTITY(1,1) UNIQUE,
AdminUserName VARCHAR(30) NOT NULL,
AdminPass VARCHAR(30) NOT NULL,

PRIMARY KEY (AdminId)
);

/*BLOGS TABLE*/
CREATE TABLE BLOGS(
BlogId INT IDENTITY(1,1) UNIQUE,
BlogBody VARCHAR(1000) NOT NULL,
ApprovalStatus TINYINT NOT NULL DEFAULT 0,
TypeOfBlog VARCHAR(255) NOT NULL,
BlogCoverImage IMAGE,
BlogerId INT NOT NULL,
AdminId INT NULL,

PRIMARY KEY (BlogId),
FOREIGN KEY (BlogerId) REFERENCES Blogger(BloggerId),
FOREIGN KEY (AdminId) REFERENCES ADMINS(AdminId)
);
```

/*BLOGTAGS TABLE*/

```
CREATE TABLE BLOGTAGS(
BlogId INT ,
Tag1 VARCHAR(255) NOT  NULL,
Tag2 VARCHAR(255) NULL,
Tag3 VARCHAR(255) NULL,
Tag4 VARCHAR(255) NULL,

PRIMARY KEY (BlogId),
FOREIGN KEY (BlogId) REFERENCES BLOGS(BlogId)
);
```

/*COMPLAINT TABLE*/

```
CREATE TABLE COMPLAINT(
ComplaintId INT IDENTITY(1,1),
ComplaintSubject VARCHAR(255) NOT NULL,
ComplaintDetails VARCHAR(1000) NOT NULL,
ComplaintStatus TINYINT DEFAULT 0,
UserId INT NOT NULL,

PRIMARY KEY (ComplaintId),
FOREIGN KEY (UserId) REFERENCES UserDetails(UserId)
);
```

/*ADS TABLE*/

```
CREATE TABLE ADS(
AdsId INT IDENTITY(1,1) UNIQUE,
AdsTitle VARCHAR(255) NOT NULL,
AdsBody VARCHAR(255) NOT NULL,
AdsTransactionId VARCHAR(255) NOT NULL,
AdsPaymentStatus TINYINT NOT NULL DEFAULT 0,
AdsAprovalStatus TINYINT NOT NULL DEFAULT 0,
UserId INT NOT NULL,
AdminId INT NULL,

PRIMARY KEY (AdsId),
FOREIGN KEY (UserId) REFERENCES UserDetails (UserId),
FOREIGN KEY (AdminId) REFERENCES ADMINS(AdminId)
);
```

```
/*NEWS TABLE*/
CREATE TABLE NEWS(
NewsId INT IDENTITY(1,1) UNIQUE,
Headline VARCHAR(255) NOT NULL,
NewsLink VARCHAR(255) NOT NULL,
SourceName VARCHAR(255) NOT NULL,
NewsType VARCHAR(255) NOT NULL,
PostTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
AdminId INT NOT NULL,

PRIMARY KEY(NewsId),
FOREIGN KEY (AdminId) REFERENCES ADMINS (AdminId)
);

/*EVENTS TABLE*/
CREATE TABLE EVENTS(
EventId INT IDENTITY(1,1) UNIQUE,
EventTitle VARCHAR(255) NOT NULL,
EventDetail VARCHAR(1000) NOT NULL,
EventDate DATE NOT NULL,
EventTime TIME NOT NULL,
Venue VARCHAR(255) NOT NULL,
EventCoverPicture IMAGE NOT NULL,
PostTime DATETIME NOT NULL
        DEFAULT CURRENT_TIMESTAMP,
AdminId INT NOT NULL,

PRIMARY KEY(EventId),
FOREIGN KEY (AdminId) REFERENCES ADMINS (AdminId)
);

/*LIKE_COMMENT TABLE*/
CREATE TABLE LIKE_COMMENT(
UserId INT ,
BlogId INT ,
Likes TINYINT NOT NULL DEFAULT 0,
Comment VARCHAR(255) NULL

PRIMARY KEY(UserId,BlogId),
FOREIGN KEY (UserId) REFERENCES UserDetails(UserId),
FOREIGN KEY (BlogId) REFERENCES BLOGS(BlogId)
);
```

➢ Highlight the Primary Key and Foreign Key

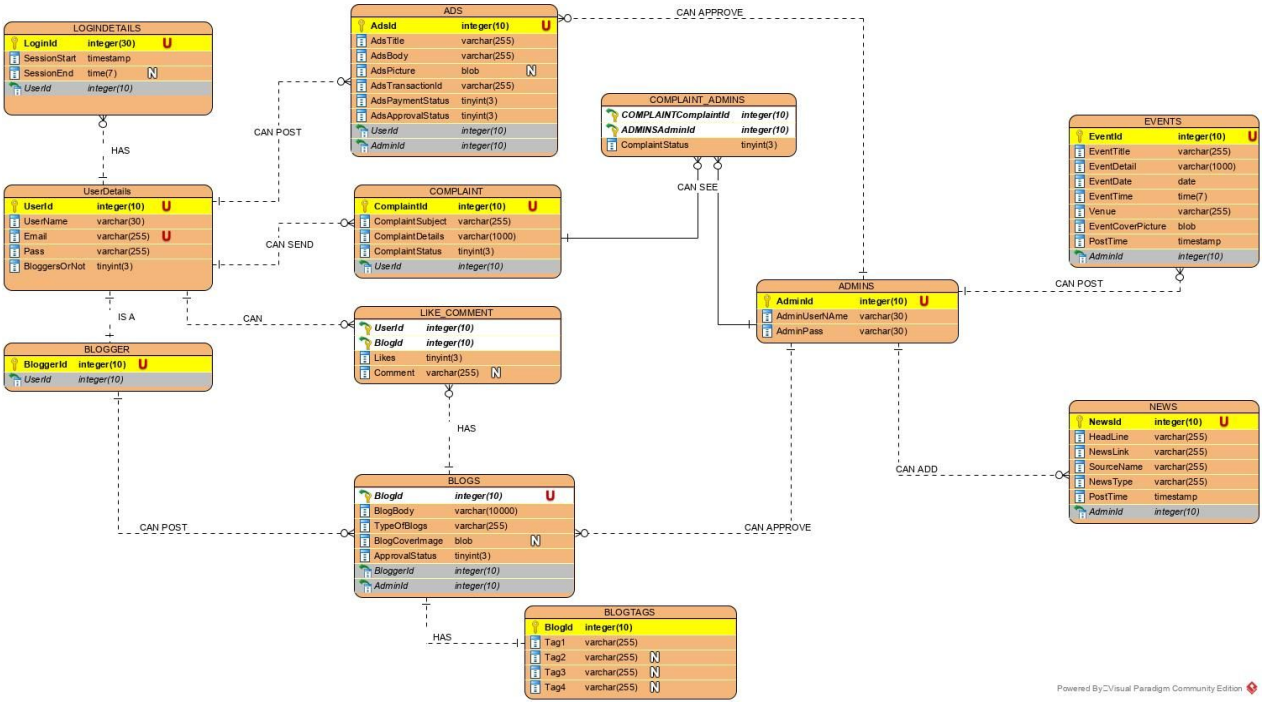
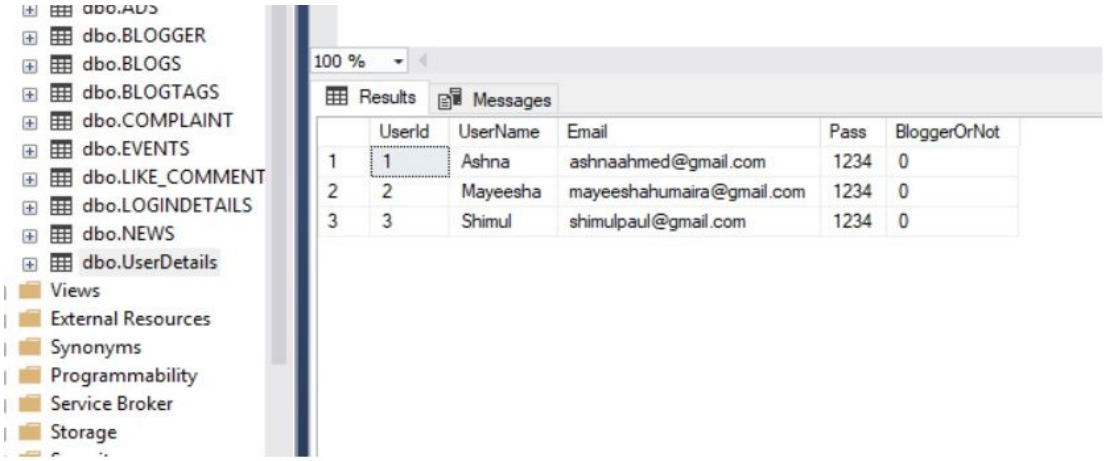


Illustration:

- ➢ Yellow colour represents the primary key of an entity
- ➢ Grey colour represents foreign key
- ➢ White colour represents composite key

➢Insert some dummy data to the table and justify

- **User Details Table:**

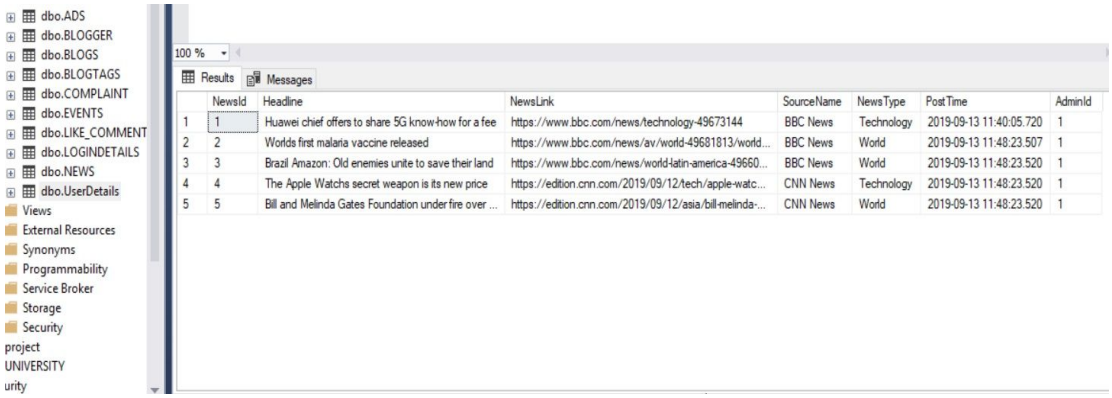


	UserId	UserName	Email	Pass	BloggerOrNot
1	1	Ashna	ashnaahmed@gmail.com	1234	0
2	2	Mayeesha	mayeeshahumaira@gmail.com	1234	0
3	3	Shimul	shimulpaul@gmail.com	1234	0

Illustration

The above picture shows the table of user details that contains user id, user name, email, password and blogger or not. Data inserted in the table are dummy data. Three users' information has been stored.

- **News Table:**



	NewsId	Headline	NewsLink	SourceName	NewsType	PostTime	AdminId
1	1	Huawei chief offers to share 5G know-how for a fee	https://www.bbc.com/news/technology-49673144	BBC News	Technology	2019-09-13 11:40:05.720	1
2	2	Worlds first malaria vaccine released	https://www.bbc.com/news/av/world-49681813/world...	BBC News	World	2019-09-13 11:48:23.507	1
3	3	Brazil Amazon: Old enemies unite to save their land	https://www.bbc.com/news/world-latin-america-49660...	BBC News	World	2019-09-13 11:48:23.520	1
4	4	The Apple Watchs secret weapon is its new price	https://edition.cnn.com/2019/09/12/tech/apple-watc...	CNN News	Technology	2019-09-13 11:48:23.520	1
5	5	Bill and Melinda Gates Foundation under fire over ...	https://edition.cnn.com/2019/09/12/asia/bill-melinda-...	CNN News	World	2019-09-13 11:48:23.520	1

Illustration:

The above picture shows the news table that has news id, headline, news link, source name, news type, post time and admin id as columns. Data given in the table are dummy data.

- **Admin Table:**

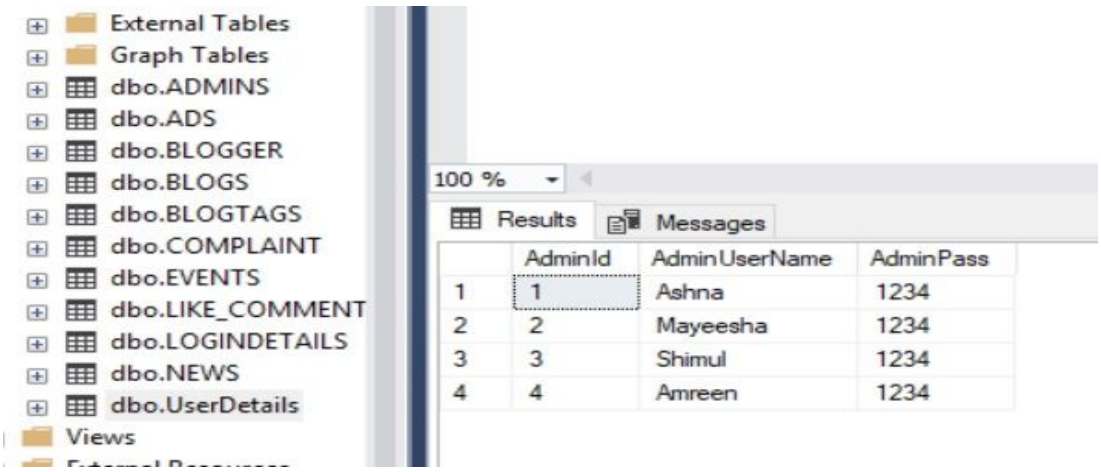


Illustration:

The above picture shows the table for admins. It consists of admin id, admin User name and admin pass as a column. Data inserted are some dummy data.

Design

➢ Home Page

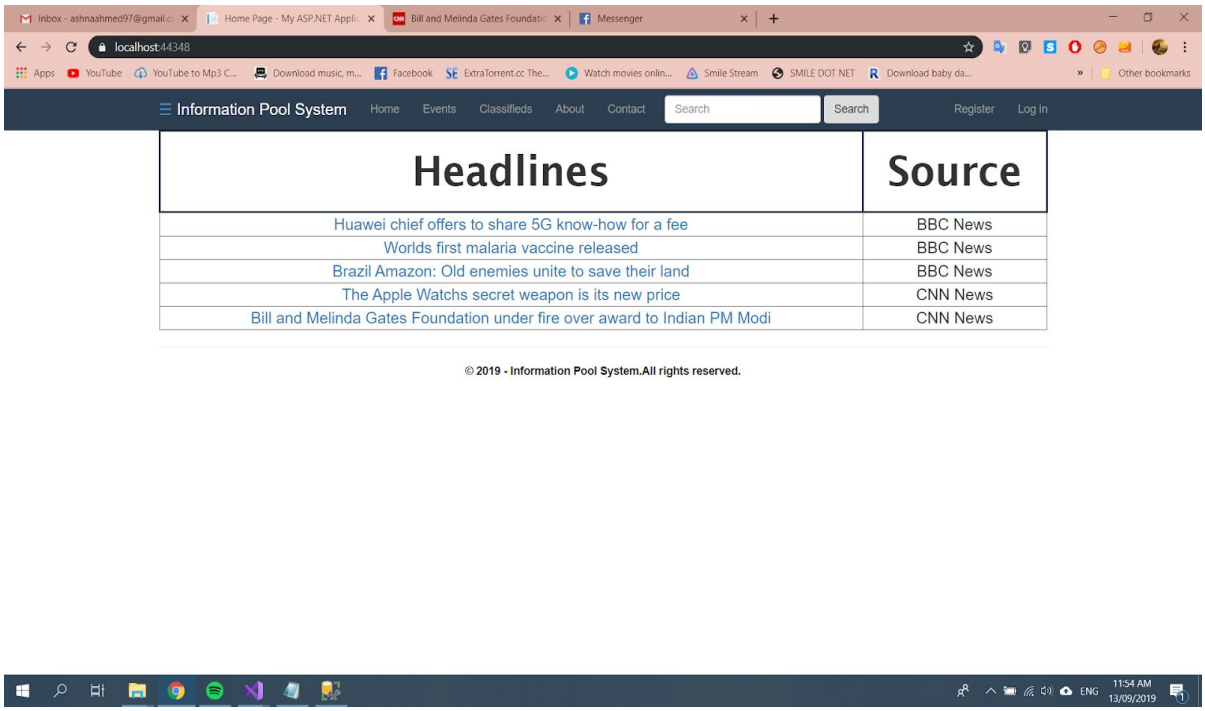


Illustration:

The above picture shows the home page of the Information Pool System, where a user can see different headlines links of the various newspaper.

➢ Login Page

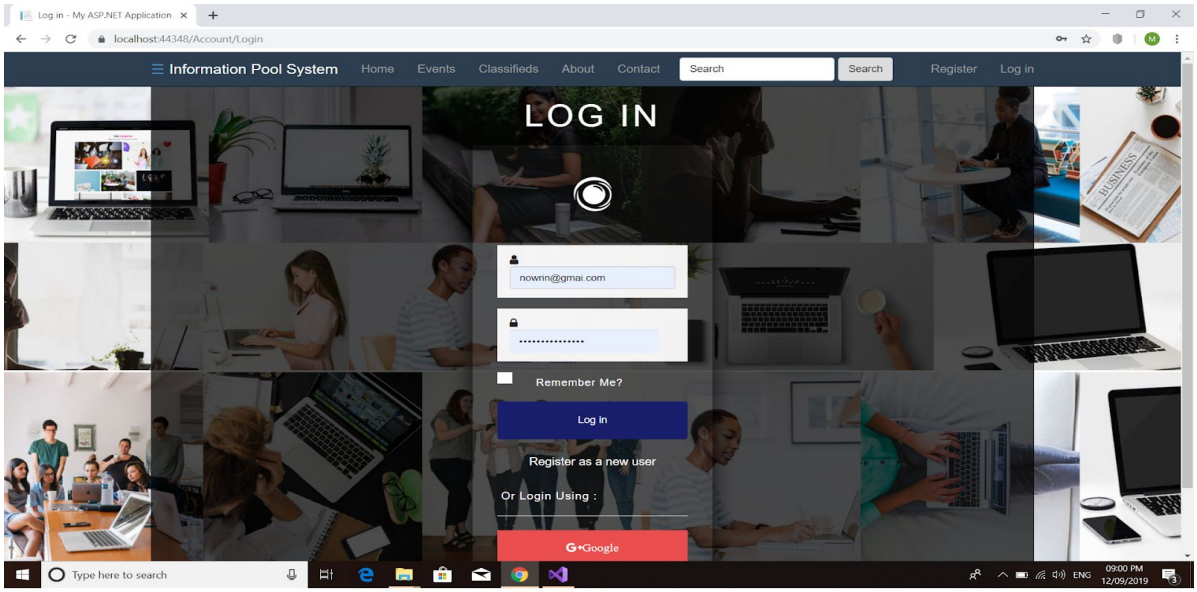


Illustration:

The above picture shows the login page of Information Pool System, where the user needs to write email and password for login or they can also log in through G+ Google. If any user is not registered yet then he can go to the registration page through 'Register as a new user'.

➢ Registration Page

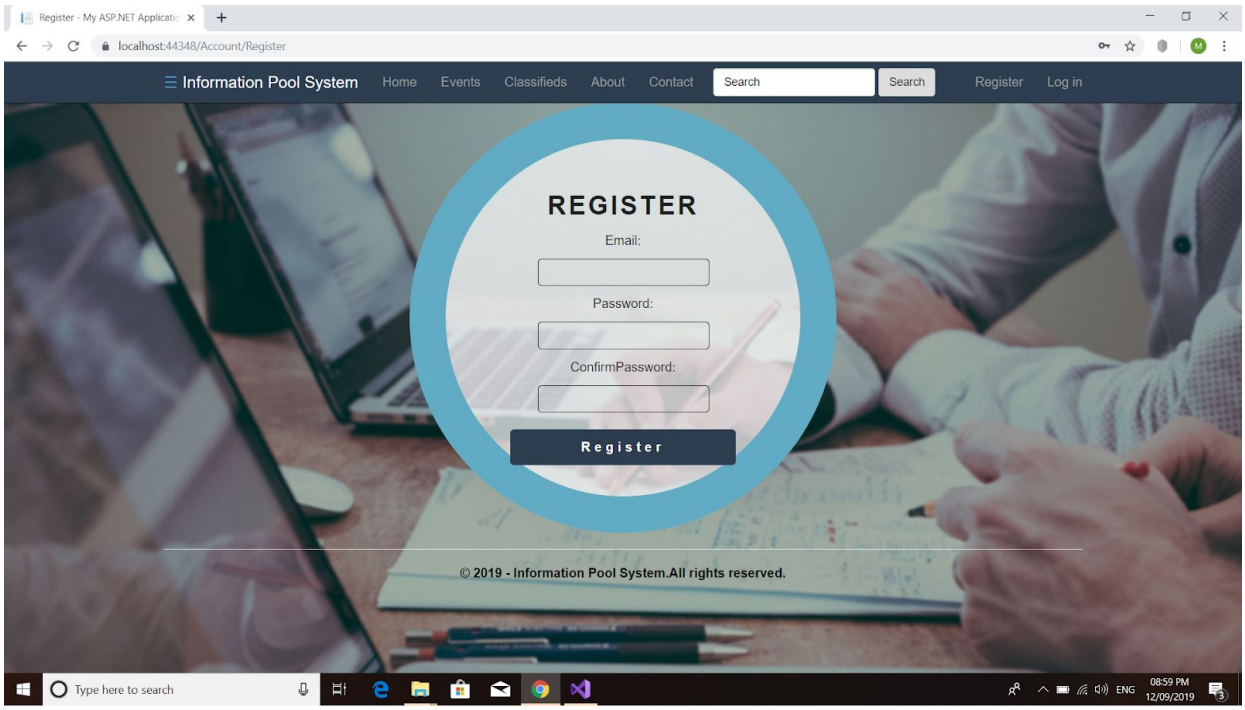


Illustration:

The above picture shows the registration page which consists of email, password and confirmation password with a registration button.

Conclusion

From the above discussion, we visualised the Information Pool System using ER Diagram, Relational Model, and Basic Front End. Using the ER Diagram Analyst can produce a good database structure so that the data can be stored and retrieved in a most efficient manner. The relational model for database management is an approach for managing data. These diagrams will be helpful in the future to build the back end of our project.