INTRODUCTION TO COMPANY

Mindfire Solutions is One of India's Best Full-spectrum It Services and Industry Solutions Providers with Global Delivery Capabilities. Mimdfire Solutions Focuses on It Consulting and Web Services to Various Companies in Businesses of Financial Services, Telecommunications, Consumer Electronics, Manufacturing, Pharmaceuticals, Healthcare, Automobile, Media, Energy, Education Sectors. With the Best Track Record of Delivering Technology Leadership and Operation Excellence for Its Clients in Various Countries. Headquartered in Bhubaneswar, Noida, USA, Mindfire Solutions Looking Forward to Possesses Nationwide Branches, as Well as an International Footprint in the other States & Countries. Our Cross-globe Network of Talents, Flexible Engagement Models, and Professional Environment Enable Us to Provide Top-quality Cost- effective Services. To attain optimum performance, we create responsive projects fulfilling the customer's attributes. Our IT solutions are credible and trusted offering full customer satisfaction. We ensure our customers of a defined growth in their business.

1. INTRODUCTION

1.1 Project Introduction :-

Blogging has become such a mania that a new blog is being created every second of every minute of every hour of every day. A blog is your best bet for a voice among the online crowd. Blogs were usually the work of a single individual occasionally of a small group, and often covered a single subject. More recently, "multi-author blogs" (MABs) have developed, with posts written by large numbers of authors and professionally edited. MABs from newspapers, other media outlets, universities, think tanks, advocacy groups, and similar institutions account for an increasing quantity of blog traffic. The rise of Twitter and other "microblogging" systems helps integrate MABs and single-author blogs into societal new streams. Blog can also be used as a verb, meaning to maintain or add content to a blog. A novel is a long, fictional narrative which describes intimate human experiences. BlogDen is a combination of both Blog as well as Novels. Blog contain the Information of various things related to Technology, Education, News, International, Business, Sports, Entertainment and ongoing college activities. The main aim of this project is to provide data to students in only one site. Students can gather the information from one site as well as give their feedback and create their own blog. Students can post their views and thought and analyse themselves. Besides all such core functionalities, the application also includes features like FAQ, request, feedback etc. so as to provide a satisfactory user experience.

1.2.1 Existing System :-

Existing system is manual system. It requires a lot of file work to be done. It is a time consuming system. All customer information is maintained manually. Any searching requires so much effort manually.

There is no way of spreading the information so fast and in cheapest manner. In previous system all information does not get in one place. Here people can write whatever they want to write.

1.2.2 Drawbacks of Existing System:-

a) Data redundancy and formatting: The various files are likely to have different formats and therefore lead to redundancy and inconsistency.

b) Maintaining registers is costly: Traditionally documents have been stored in batches and they field in file cabinets and boxes. A numerical system is they assigned. Specifically a consumer number assigned to organize the files.

c) Error prone: Existing systems are error prone, since manual work is required. More time is consumed and errors may propagate due to human mistakes.

d) Low security feature: Due to maintenance's of records manually and shared and could view easily by anyone. Also these could be possible loss of data and confidential information due to some disaster in the form of fire, theft etc.

2. PROPOSED SYSTEM

2.1.1 Why We Choose This Project:-

In recent past time Blogs are store in the paper files and difficult to search or modify any information, for expanding the Blogs infrastructure, Awareness of environmental issues or any other issues related to education, health, digital technology, and search for greater safety give to information to all persons in all age groups and a new role within the education system, I choose this project. As a result of these project initiatives phenomenal growth has taken place in all the activities of blogs and any user can share its information related to any topic to all users.

2.1.2 Benefits of Project

This is a very simple design and implement. It has got following features:

1) Data can be saved safely.
2) No other person cannot view other person's details.
3) Greater efficiency
4) User friendliness
5) Minimum time required
6) Free of cost

2.1.3 Application

BlogDen enables the users to create innovative and attractive information with photos in just few simple steps. The user just needs to upload some images of his choice and can also upload the information or can select from the given category list. This website will provide a personalized environment that would contain the data in motion with images.

2.1.4 Scope of the Project

- To use the personal images in greeting our loved ones.
- To use the music of our choice to greet in the language we want.
- To provide a personalized mp4 video to the customers.
- To satisfy the customers and provide them with the ordered video before time.

2.1.5 Organization of Report:-

Based on the outline design of the system requirements in terms of inputs, output, Procedures, the technical issues raised during technical feasibility include:

1) Does the necessary technology exist to do what is proposed?
2) Does the proposed equipment have the technical capacity to hold the data required to use in the new system?
3) Adequate responses provided by the proposed system?
4) Is the system flexible enough to facilitate expansion?
5) Is there any technical guarantee of accuracy, reliability, ease of access and data security?
6) The system developer's task is to view needed capabilities in light of currently available technology. Our site works hand in hand with high technology. A database has to be maintained in order to update and backup data whenever required. To create databases we use SQL server.
7) For the acceptance within the organization the following points are important and those are explained according to the topics
8) Whether the system provides right information to the right place?
9) In the current system which is the semi computerized system the information may be lost in the process of sending from one place to another. This is mainly due to human interaction in the process of the transferring information from one place to another. Whether the new system affects the current users in the system? The new proposed system will affect the users in the following areas
   - Accuracy
   - Efficiency
   - Productivity

2.2 Operating Environment of The System:

The development of the ASP.NET MVC was done on a system with the following hardware profile:

Processor : Intel Core i5.

RAM : 16 GB DDR2 RAM

Hard-disk : 256 GB

The software packages used, as well as the languages employed in each one of them are as follows:

Front End : HTML, CSS, JQUERY, BOOTSTRAP

Back End : C#

Framework : ASP.NET MVC, Entity framework

Database: SQL Server

Operating System : Windows 10 Professional

Text Editor : Visual Studio 2015

2.3 Methodology Adopted:

Object-oriented Methodology

The project follows object oriented analysis and design approach for development. It exploits UML driven models more specifically use case and sequence diagram. The project covers the entire life cycle (SDLC).

SDLC life cycle

1) Requirement Gathering.
2) Analysis.
3) Design
4) Development
5) Testing

The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process. It is a logical systematic process used to develop software and information system through requirement gathering, analysis, design, development and testing. It is an approach that allows for the system development to be a lot simplified and easy.

The object-oriented approach combines data and processes (called methods) into single entities called objects. Objects usually correspond to the real things a system deals with, such as customers, invoice. Object-oriented models are able to thoroughly represent complex relationships and to represent data and data processing with a reliable notation, which allows an easier mix of analysis and design in a growth process. The aim of the Object-oriented approach is to make system elements more modular, thus improving system quality and the efficiency of the system and design.

3. REQUIREMENT ANALYSIS

Requirements analysis is the process of defining what the user requires from the system and defining the requirements clearly and in an unambiguous state. The outcome of the requirement analysis is the software developing activities. Thus it deals with understanding the problem goals and constraints. This specification part mainly focuses on what had been found during analysis. A requirement is a relatively short and concise piece of information, expressed as a fact. It can be written as a sentence or can be expressed using some kind of diagram. Requirements are divided into two major types functional and non-functional.

This project is overcoming the difficulty of understanding of programming language and other technical subjects by providing a free website to learn and communicate with the experienced people. It provides user friendly interface to share their view on a particular topic.

3.1.1 SRS (Software Requirement Specification)

Functional requirements:

Following is a list of functionalities of the system.

What inputs the system should accept.

What outputs the system should produce.

What data the system must store.

Introduction:

BlogDen is a total management and informative system, which provides the up-to date information of all the articles. BlogDen helps the people to overcome the difficulty in finding knowledge and solutions to their problems. It helps in effective and timely utilization of the hardware and the software resources.

Inputs:

The Administrator handles the entire system. The role of administrator in the system is, to manage all the articles that are posted (like, approve them or decline them), to manage all the users and moderators (Like, whether to rank up a user to make moderator or to rank down a moderator to make normal user). The bloggers contribution for this website is to visit each article and decide their status (like, to approve them or to decline them). Login to the system through the first page of the application

Requirement Specification:

Complete specification of the system (with appropriate assumptions) constitutes this

milestone. A document detailing the same should be written and a presentation on that be made.

Database Creation:

A database should be created, as per the rules for the purpose of maintenance of the records of articles and users.

Implementation of The Front-End:

Implementation of the main screen showing the latest articles posted and also the popular articles. Login option redirects the user to login/signup operations. After that the user can view his/her own profile and can edit it.

Integrating The Front-End with The Database:

The front-end developed in the earlier milestone will now be able to update the database. Other features like mail notification etc. should be functional at this stage. In short, the system should be ready for integration testing.

Processing:

As the system is information-oriented project and there are no certain calculations only database storage and view is provided.

Storage data: In this we store all the details of articles and users

Outputs: The project provides knowledge about technical subjects from experts and experienced people.

Non-Functional Requirements:

Non-functional requirements are the constraints that must be adhered during development. They limit what resources can be used and set bounds on aspects of the software's quality.

User Interfaces:

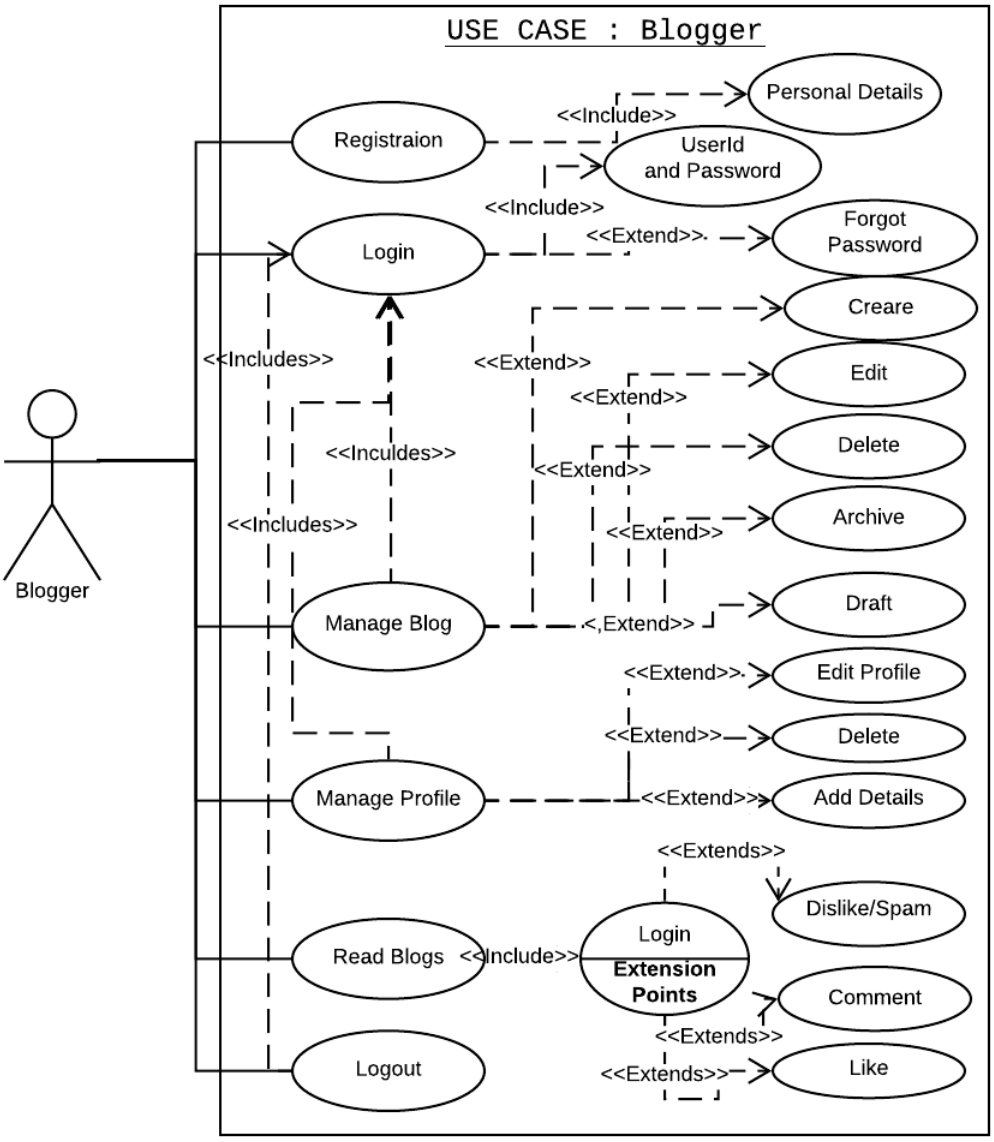
The User Interface is a GUI developed using HTML, CSS, JQuery, Bootstrap

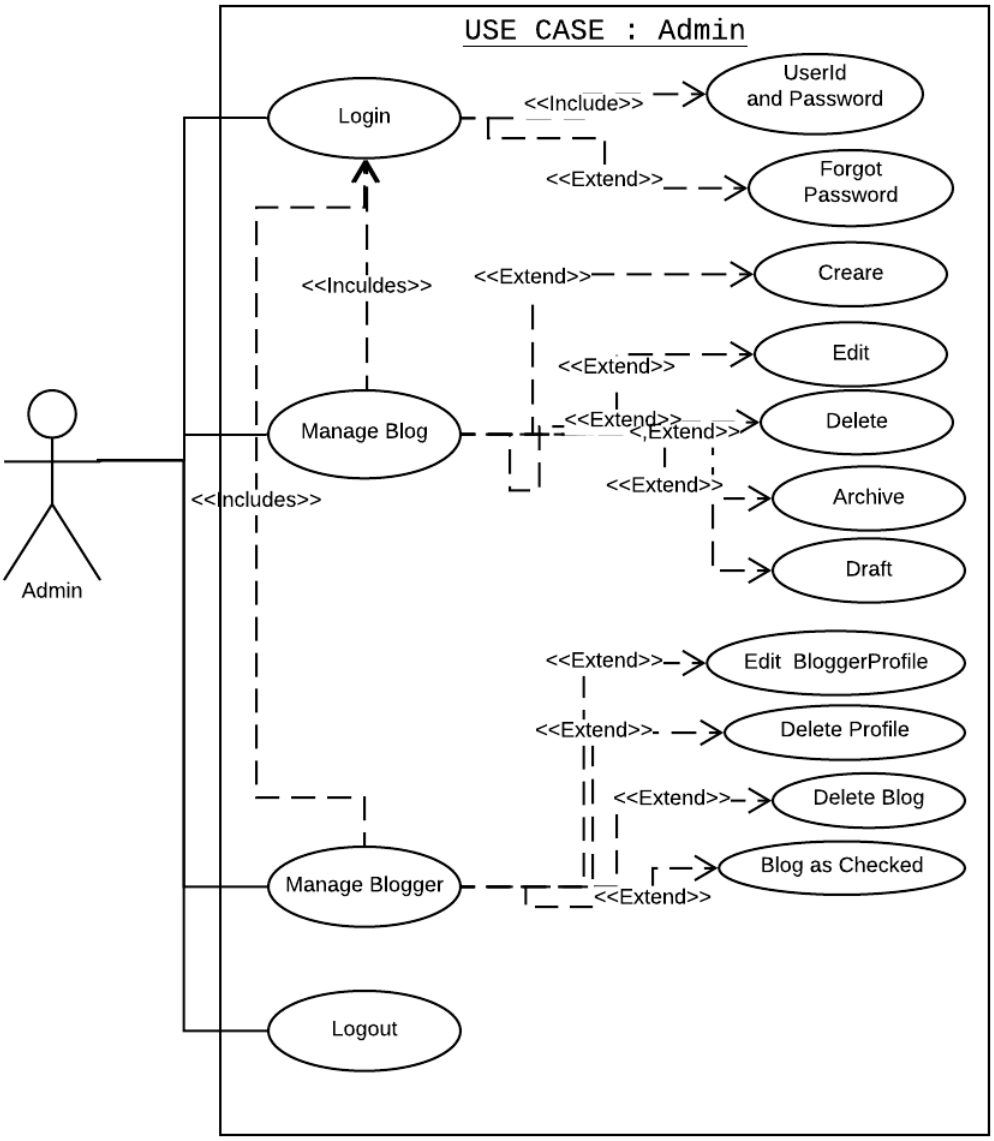
Performance Requirements:

The product performance needs to be assessed on certain characteristics.

Input: The inputs that the user gives i.e., user id and password is very important.

3.1.2 Use Case


USE CASE : Blogger
Blogger
Registraion
Login
Manage Blog
Manage Profile
Read Blogs
Logout
Personal Details
UserId and Password
Forgot Password
Creare
Edit
Delete
Archive
Draft
Edit Profile
Delete
Add Details
Dislike/Spam
Comment
Like
Login
Extension Points
<<Include>>
<<Extend>>
<<Includes>>
<<Inculdes>>
<<Extends>>



Use Cases Description:

1.Login:

Actor: Admin, User.

Description: There are three users for the system, Admin, User. User and Admin have to register first and Login to the system by using their email and password.

2.Create Comment:

Actor: Admin, User.

Description: Admin, and User can comment on any article displayed on the website.

3.View Comment:

Actor: Admin, User.

Description: Admin and User can see all comments posted by other users.

4.Delete Comment:

Actor: Admin, User.

Description: User can delete his/her own articles and also those comment that are posted on his/her own article, but admin and moderator can delete any irrelevant comments posted.

3.2 Functional Requirements :-

3.2.1 Requirments

1) Login for Admin

2) Forgot password for Admin/User

3) Edit Profile for Admin/User

4)Change Password for Admin/User

Functionality:-

1) Dashboard for Admin user
2) Manage Blog
3) Adding New Blog
4) Edit the Exiting Blog
5) View details of the Blog
6) Listing of all Blog

Manage Comment:-

1) Adding New Comment
2) Edit the Exiting Comment
3) View details of the Comment
4) Listing of all Comment

Manage Blog:-

1) Blog Edit the Exiting
2) Create New Blog
3) View details of the Blog

3.2.2 Object Oriented Analysis

Object–Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects. In the system analysis or object-oriented analysis phase of software development, the system requirements are determined, the classes are identified and the relationships among classes are

identified. The three analysis techniques that are used in conjunction with each other for object-oriented analysis are object modelling, dynamic modelling, and functional modelling.

Object Modelling Object modelling develops the static structure of the software system in terms of objects. It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class. The process of object modelling can be visualized in the following steps: • Identify objects and group into classes

• Identify the relationships among classes

• Create user object model diagram

• Define user object attributes

• Define the operations that should be performed on the classes Dynamic Modelling After the static behaviour of the system is analysed, its behaviour with respect to time and external changes needs to be examined. This is the purpose of dynamic modelling.

Dynamic Modelling can be defined as "a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world". The process of dynamic modelling can be visualized in the following steps:

• Identify states of each object

• Identify events and analyse the applicability of actions

• Construct dynamic model diagram, comprising of state transition diagrams • Express each state in terms of object attributes

• Validate the state–transition diagrams drawn Functional Modelling Functional Modelling is the final component of object-oriented analysis.

The functional model shows the processes that are performed within an object and how the data changes as it moves between methods. It specifies the meaning of the operations of object modelling and the actions of dynamic modelling. The functional model corresponds to the data flow diagram of traditional structured analysis. The process of functional modelling can be visualized in the following steps:

• Identify all the inputs and outputs

• Construct data flow diagrams showing functional dependencies

• State the purpose of each function

• Identify constraints

• Specify optimization criteria

3.2.3 Object Oriented Models:

Blog Comment

```
[Key]
public int CommentId { get; set; }
[MaxLength]
public string Comments { get; set; }
[MaxLength(40)]
public string CreatedBy { get; set; }
[DataType(DataType.Date)]
public DateTime CreationTime { get; set; }
[MaxLength(40)]
public string EditedBy { get; set; }
[DataType(DataType.Date)]
public DateTime LastEditTime { get; set; }
public int? ParentId { get; set; }
[ForeignKey("ParentId")]
public virtual List<BlogComment> SubComment { get; set; }
[ForeignKey("Blog")]
public int BlogId { get; set; }
public Blog Blog { get; set; }
```

Blog Image

```
[Key]
public int BlogImageId { get; set; }
[MaxLength]
public string ImgageUrl { get; set; }
[ForeignKey("Blog")]
public int BlogId { get; set; }
public Blog Blog { get; set; }
```

BlogStatusCount

```
[Key]
public int BlogCountId { get; set; }
public int LikesCount { get; set; }
public int DislikesCount { get; set; }
public int SpamCount { get; set; }
public int CommentsCount { get; set; }
[ForeignKey("GetBlogs")]
public int BlogId { get; set; }
public Blog GetBlogs { get; set; }
public int UserId { get; set; }
```

BlogTopic

```
[Key]
public int TopicId { get; set; }
[MaxLength]
public string TopicName { get; set; }
public int? ParentId { get; set; }
[ForeignKey("ParentId")]
public virtual List<BlogTopic> SubTopic { get; set; }
```

Message

```
[Key]
public int MessageId { get; set; }
[Required]
[MaxLength(30)]
public string Name { get; set; }
[EmailAddress]
[Required]
public string Email { get; set; }
[Required]
[MaxLength]
public string Comment { get; set; }
[DataType(DataType.Date)]
public DateTime CreationTime { get; set; }
```

Response

```
[Key]
public int ResponseId { get; set; }
[ForeignKey("Status")]
public int StatusId { get; set; }
public Status Status { get; set; }
[ForeignKey("Blog")]
public int? BlogId { get; set; }
public virtual Blog Blog { get; set; }
[ForeignKey("User")]
public int? UserId { get; set; }
public virtual User User { get; set; }
```

Status

```
[Key]
public int StausId { get; set; }
[MaxLength(20)]
public string StatusName { get; set; }
```

Tag

```
[Key]
public int TagId { get; set; }
[MaxLength]
public string TagTitle { get; set; }
[ForeignKey("Blog")]
public int? BlogId { get; set; }
public virtual Blog Blog { get; set; }
[ForeignKey("BlogTopic")]
public int? TopicId { get; set; }
public virtual BlogTopic BlogTopic { get; set; }
```

User

```
[Key]
public int UserId { get; set; }
[Required]
[MaxLength(30)]
public string Name { get; set; }
[Required]
[EmailAddress]
public string Email { get; set; }
[Required]
[MaxLength(60)]
public string Password { get; set; }
[MaxLength(15)]
public string Mobile { get; set; }
[DataType(DataType.Date)]
public DateTime CreationTime { get; set; }

public DateTime DateOfBirth { get; set; }
[MaxLength]
public string ImageUrl { get; set; }
[MaxLength(15)]
public string Gender { get; set; }
[MaxLength(80)]
public string Description { get; set; }
public string Rank { get; set; }
```

UserImage

```
[Key]
public int UserImageId { get; set; }
[MaxLength(60)]
public string Title { get; set; }
[MaxLength]
public string ImageURL { get; set; }
[DataType(DataType.Date)]
public DateTime CreationTime { get; set; }
[ForeignKey("User")]
public int UserId { get; set; }
public User User { get; set; }
```

UserSkill

```
[Key]
public int UserSkillId { get; set; }
[MaxLength(60)]
public string Company { get; set; }
[MaxLength(90)]
public string skill { get; set; }
[ForeignKey("GetUserSkills")]
public int UserId { get; set; }
public User GetUserSkills { get; set; }
```

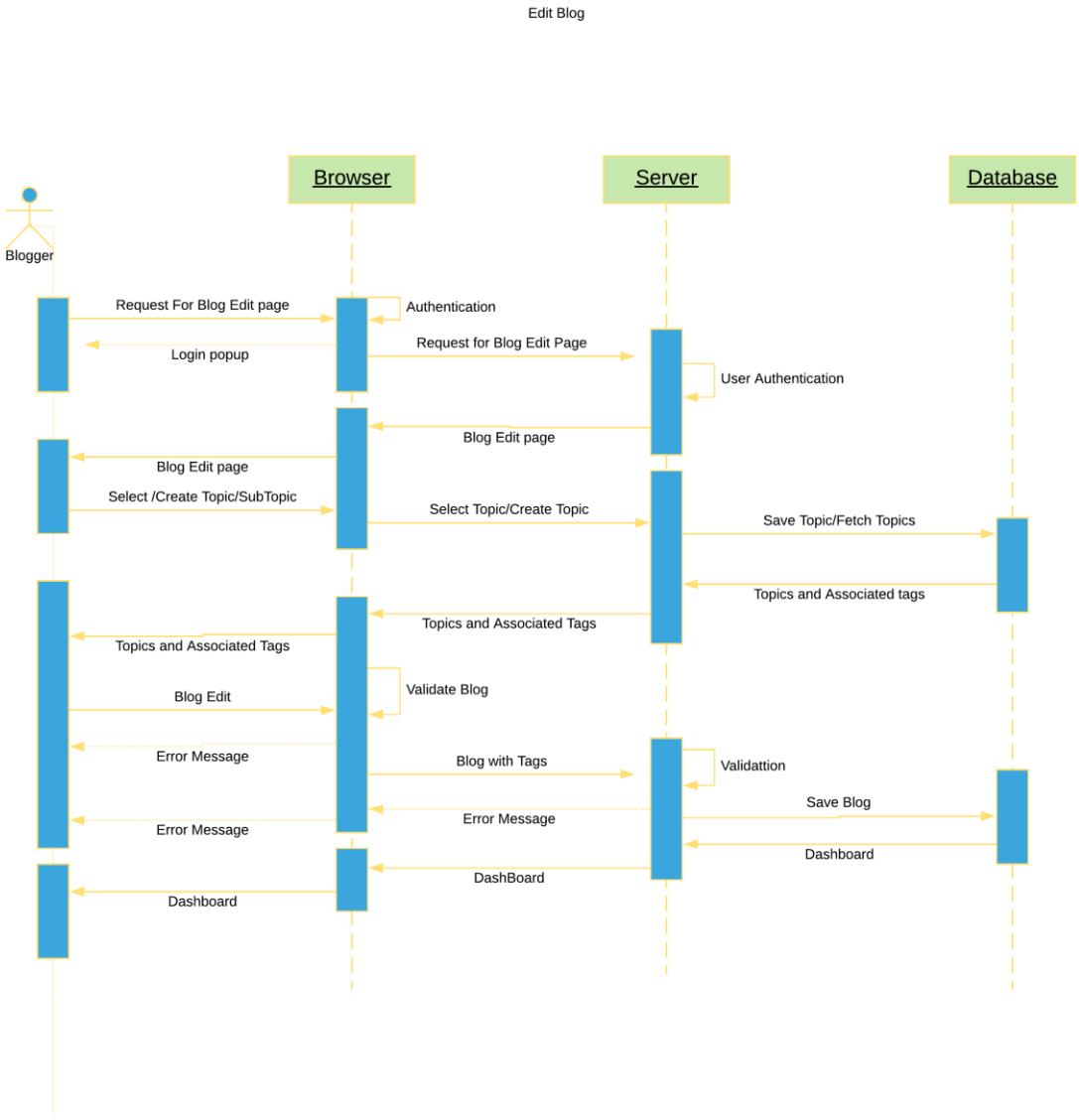
Admin

```
[Key]
public int Id { get; set; }
[MaxLength(25)]
public string Username { get; set; }
[MaxLength(60)]
public string Password { get; set; }
```

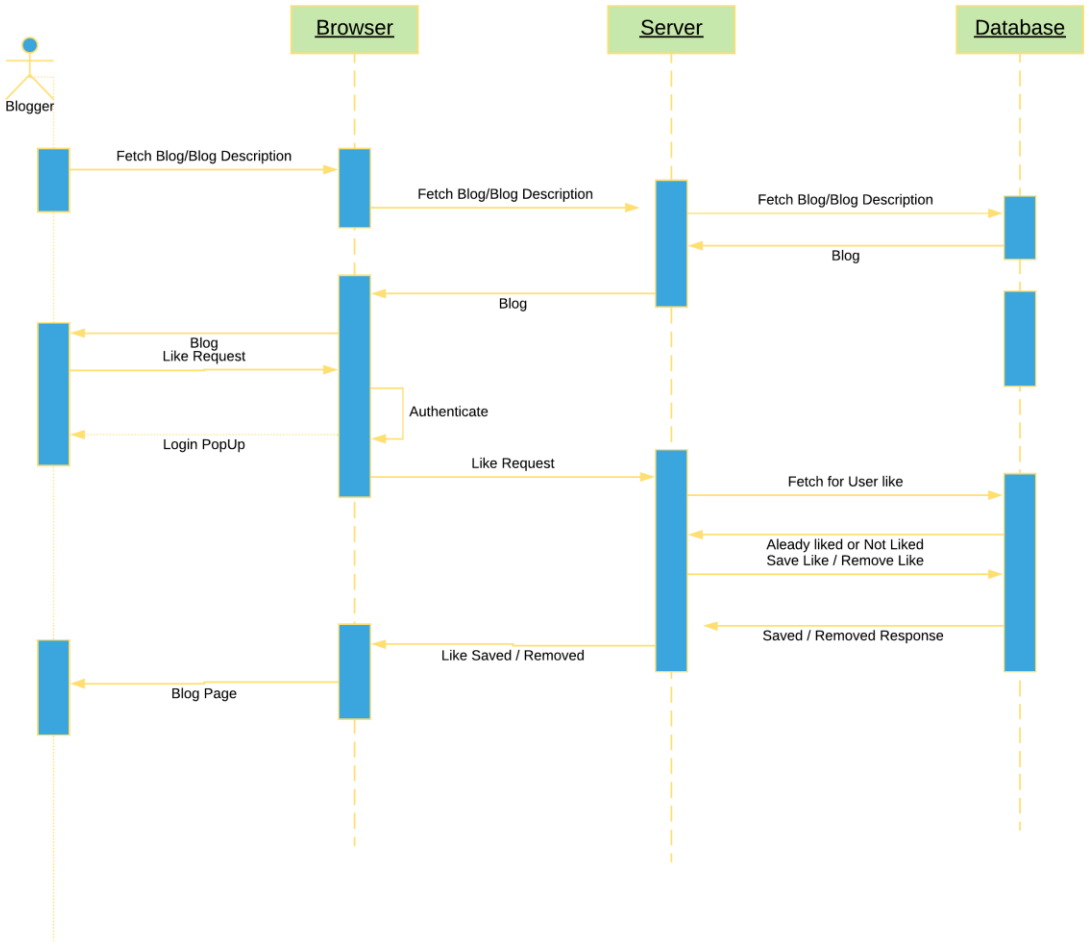
Blog

```
[Key]
public int BlogId { get; set; }
[MaxLength]
public string Title { get; set; }
[MaxLength]
public string Description { get; set; }
[MaxLength(40)]
public string CreatedBy { get; set; }
[DataType(DataType.Date)]
public DateTime CreationTime { get; set; }
[MaxLength(40)]
public string EditedBy { get; set; }
[DataType(DataType.Date)]
public DateTime LastEditTime { get; set; }
public int BlogStatus { get; set; }
[ForeignKey("User")]
public int UserId { get; set; }
public User User { get; set; }
[ForeignKey("BlogTopic")]
public int TopicId { get; set; }
public BlogTopic BlogTopic { get; set; }
```

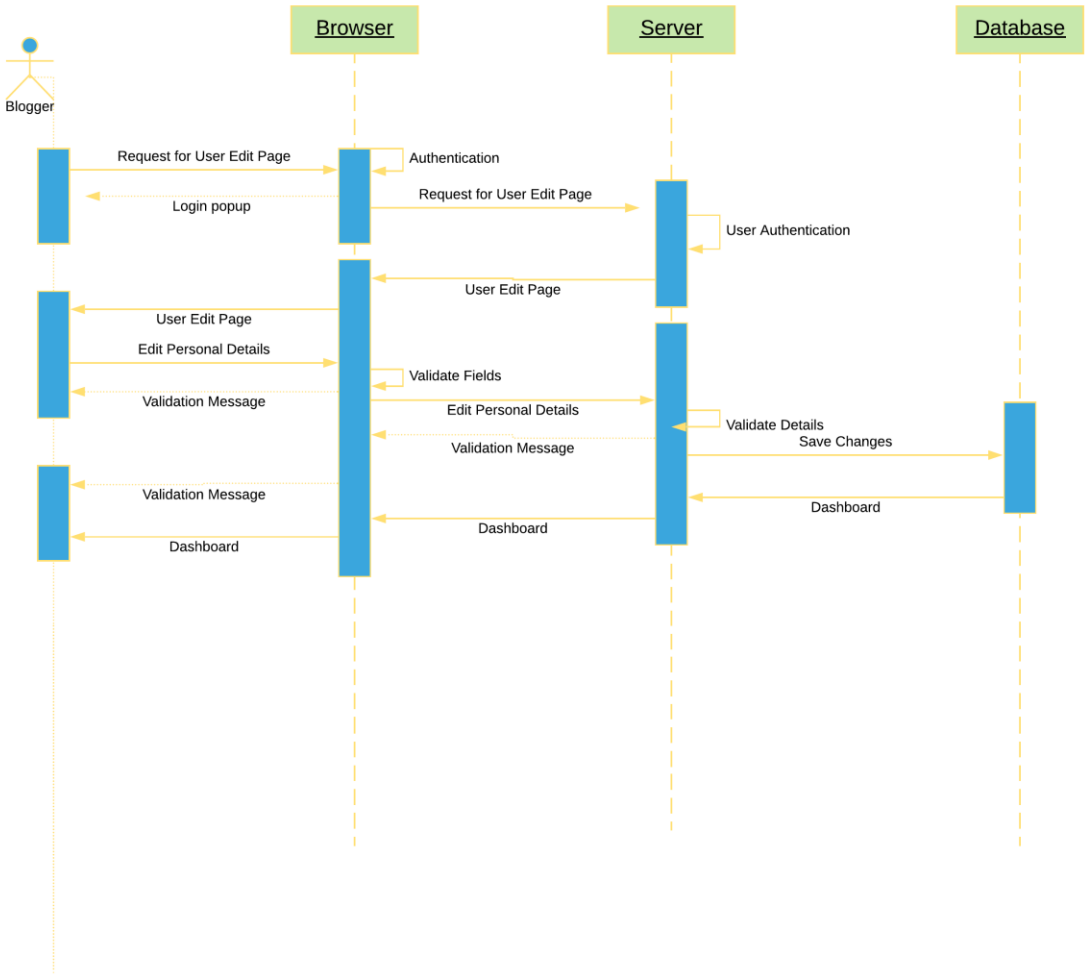
3.2.3 Sequence Diagram:



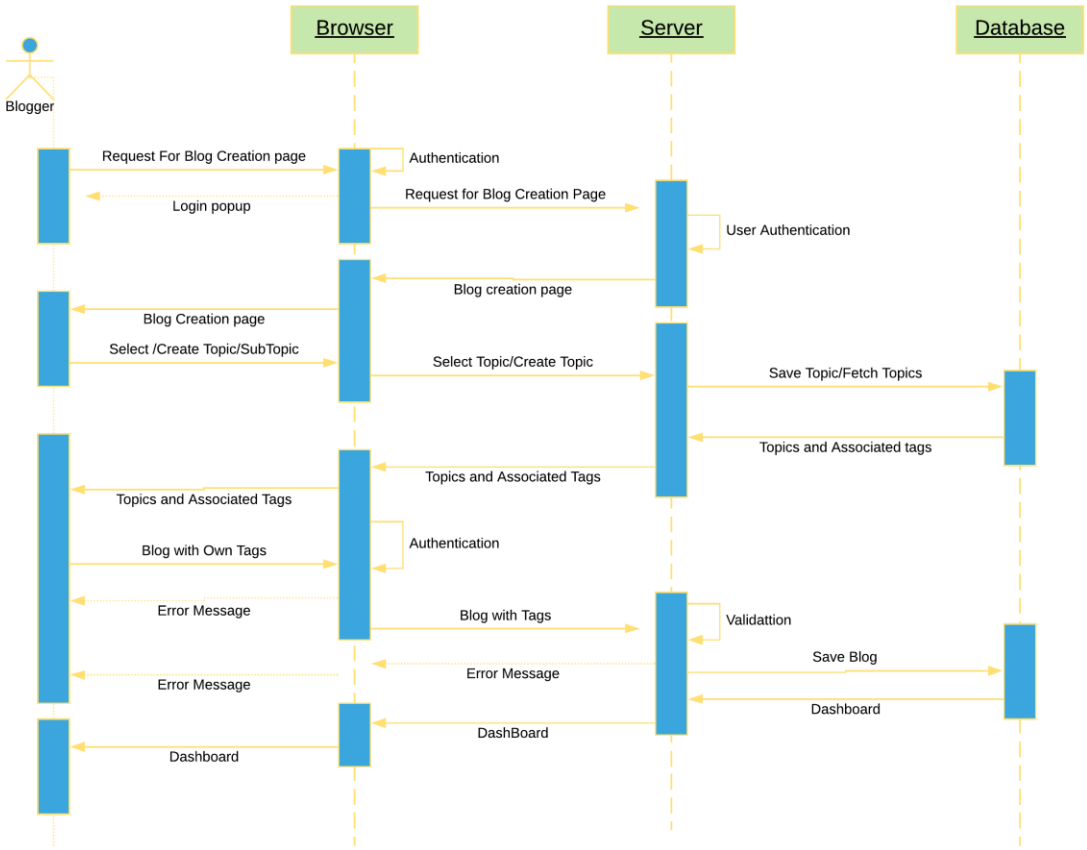
Like


Browser
Server
Database
Blogger
Fetch Blog/Blog Description
Fetch Blog/Blog Description
Fetch Blog/Blog Description
Blog
Blog
Blog
Like Request
Authenticate
Login PopUp
Like Request
Fetch for User like
Aleady liked or Not Liked
Save Like / Remove Like
Saved / Removed Response
Like Saved / Removed
Blog Page

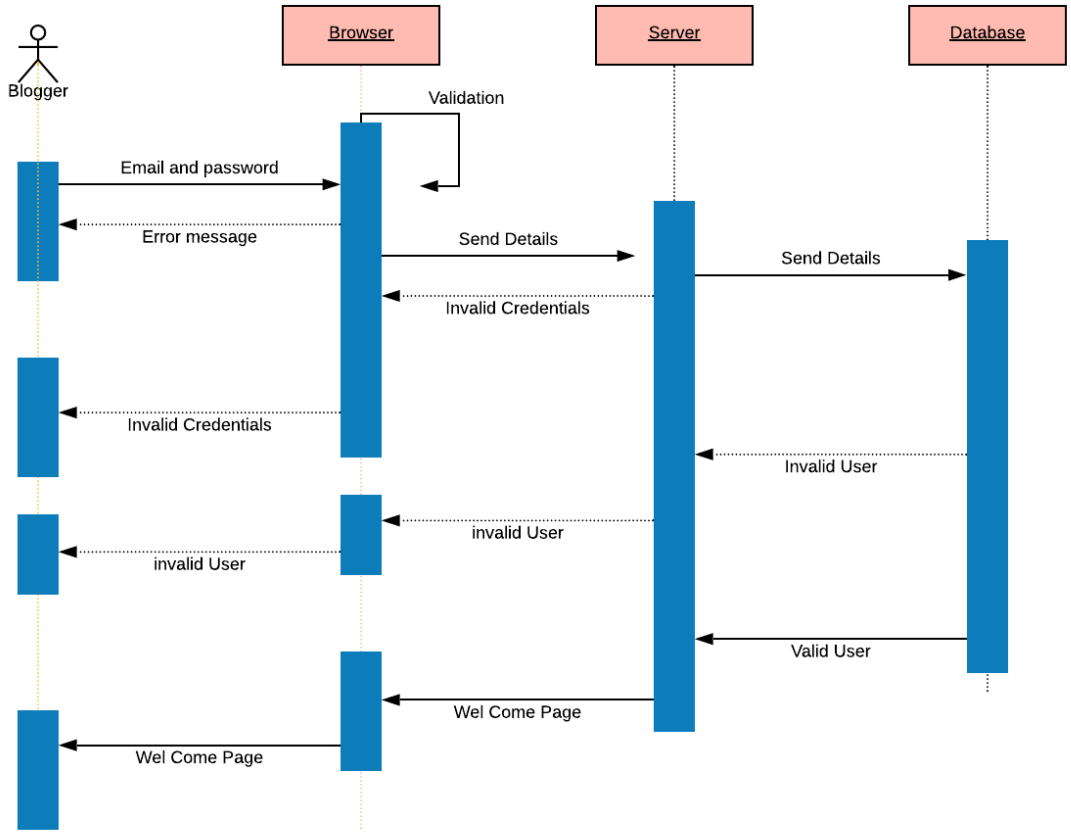
Edit USER details



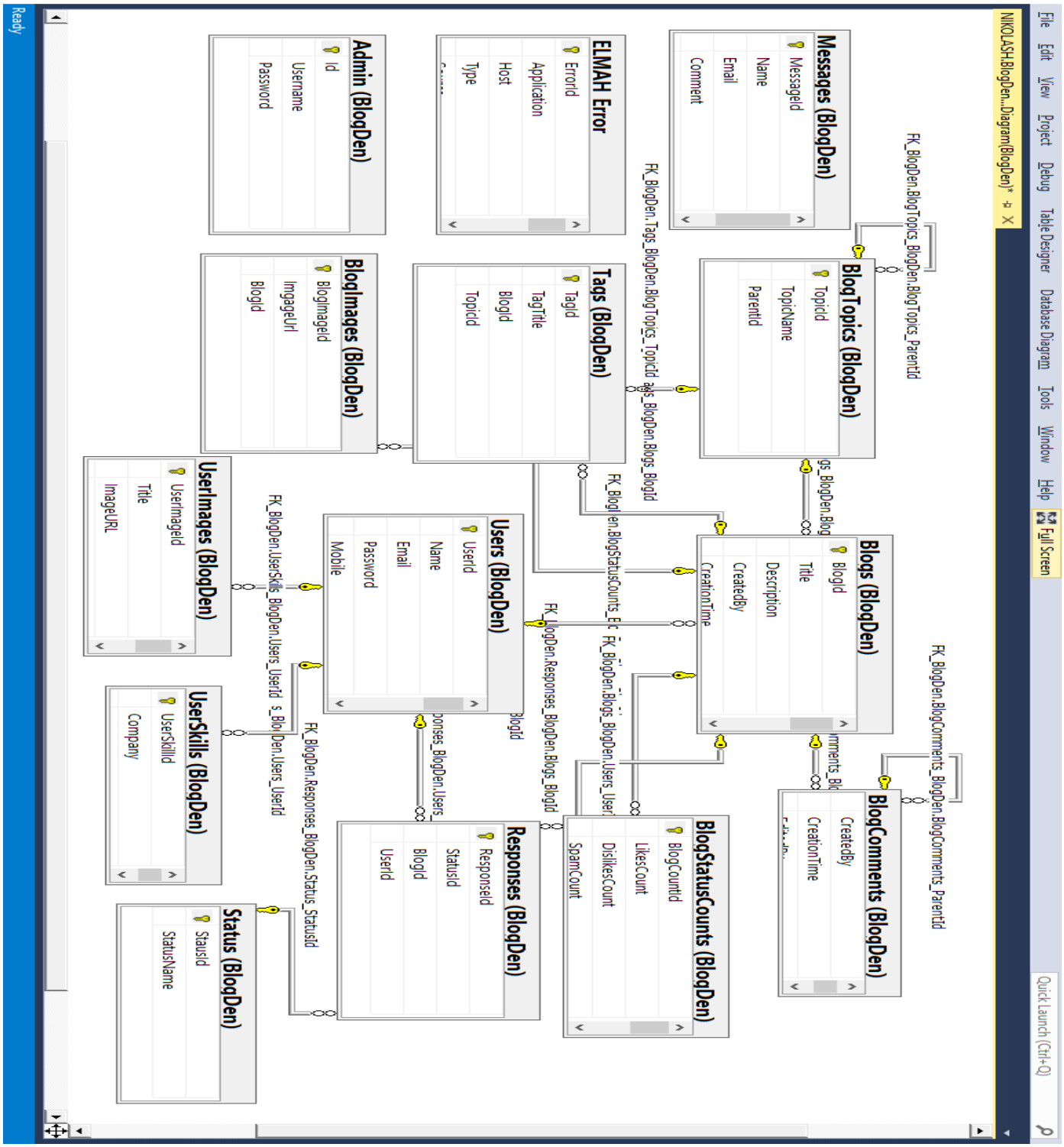
Create Blog

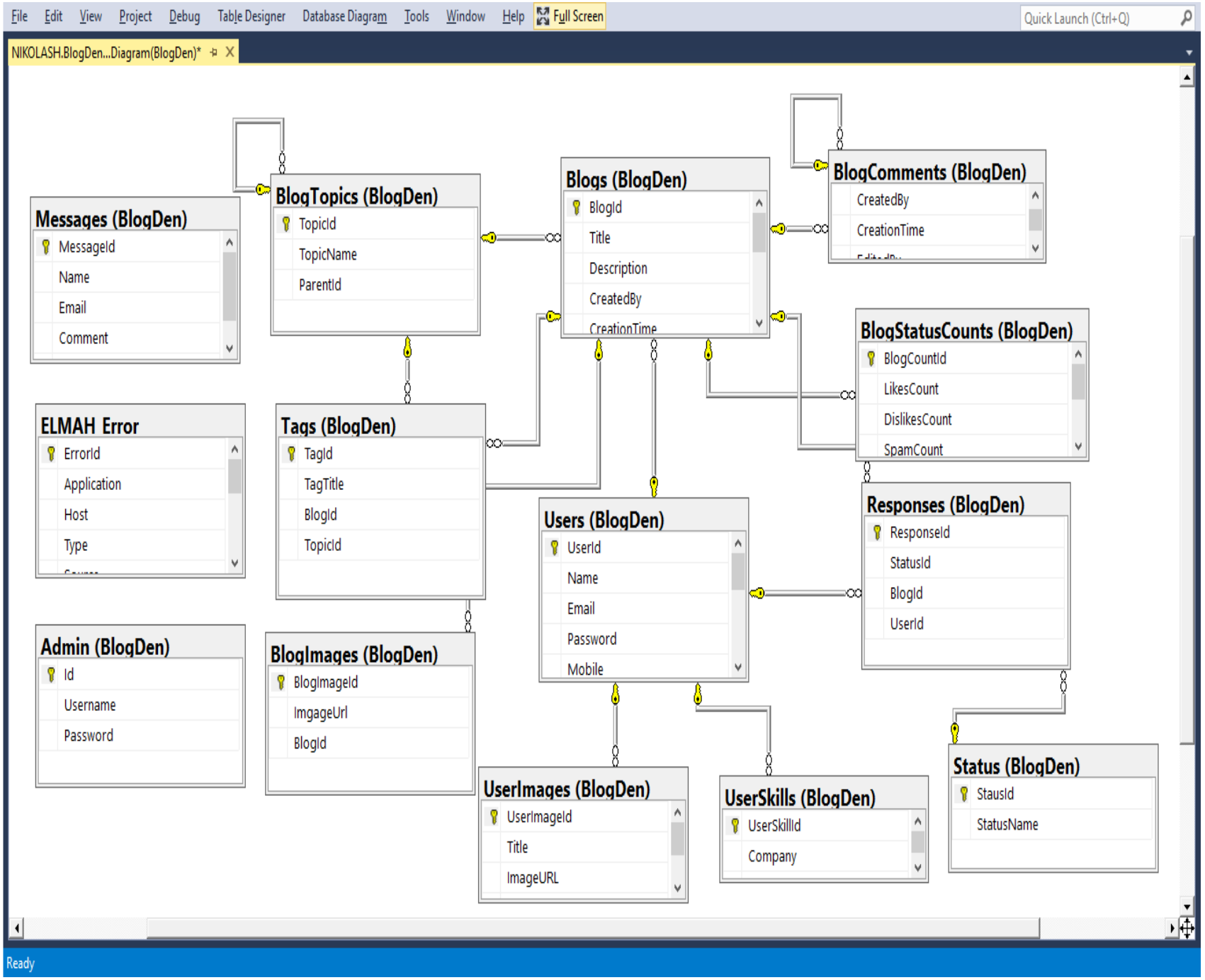

Browser
Server
Database
Blogger
Request For Blog Creation page
Authentication
Login popup
Request for Blog Creation Page
User Authentication
Blog creation page
Blog Creation page
Select /Create Topic/SubTopic
Select Topic/Create Topic
Save Topic/Fetch Topics
Topics and Associated tags
Topics and Associated Tags
Topics and Associated Tags
Authentication
Blog with Own Tags
Error Message
Blog with Tags
Validattion
Save Blog
Error Message
Error Message
Dashboard
DashBoard
Dashboard

Sequence Diagram for Login



3.3 ER-Model


File Edit View Project Debug Table Designer Database Diagram Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
NIKOLASH.BlogDen...Diagram.BlogDen)*
Messages (BlogDen)
MessageId
Name
Email
Comment
BlogTopics (BlogDen)
TopicId
TopicName
ParentId
Blogs (BlogDen)
BlogId
Title
Description
CreatedBy
CreationTime
BlogComments (BlogDen)
CreatedBy
CreationTime
BlogStatusCounts (BlogDen)
BlogCountId
LikesCount
DislikesCount
SpamCount
ELMAH Error
ErrorId
Application
Host
Type
Tags (BlogDen)
TagId
TagTitle
BlogId
TopicId
Admin (BlogDen)
Id
Username
Password
BlogImages (BlogDen)
BlogImageId
ImgageUrl
BlogId
Users (BlogDen)
UserId
Name
Email
Password
Mobile
Responses (BlogDen)
ResponseId
StatusId
BlogId
UserId
UserImages (BlogDen)
UserImageId
Title
ImageURL
UserSkills (BlogDen)
UserSkillId
Company
Status (BlogDen)
StatusId
StatusName
FK_BlogDen.BlogTopics_BlogDen.BlogTopics_ParentId
FK_BlogDen.Tags_BlogDen.BlogTopics_TopicId
FK_BlogDen.BlogComments_BlogDen.BlogComments_ParentId
FK_BlogDen.Responses_BlogDen.Blogs_BlogId
FK_BlogDen.Responses_BlogDen.Status_StatusId
FK_BlogDen.UserSkills_BlogDen.Users_UserId
Ready


File Edit View Project Debug Table Designer Database Diagram Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
NIKOLASH.BlogDen...Diagram(BlogDen)*
Messages (BlogDen)
MessageId
Name
Email
Comment
BlogTopics (BlogDen)
TopicId
TopicName
ParentId
Blogs (BlogDen)
BlogId
Title
Description
CreatedBy
CreationTime
BlogComments (BlogDen)
CreatedBy
CreationTime
BlogStatusCounts (BlogDen)
BlogCountId
LikesCount
DislikesCount
SpamCount
ELMAH Error
ErrorId
Application
Host
Type
Tags (BlogDen)
TagId
TagTitle
BlogId
TopicId
Users (BlogDen)
UserId
Name
Email
Password
Mobile
Responses (BlogDen)
ResponseId
StatusId
BlogId
UserId
Admin (BlogDen)
Id
Username
Password
BlogImages (BlogDen)
BlogImageId
ImgageUrl
BlogId
UserImages (BlogDen)
UserImageId
Title
ImageURL
UserSkills (BlogDen)
UserSkillId
Company
Status (BlogDen)
StausId
StatusName
Ready

4. DESIGN:

4.1 System Architecture:

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A model View Controller pattern is made up of the following components

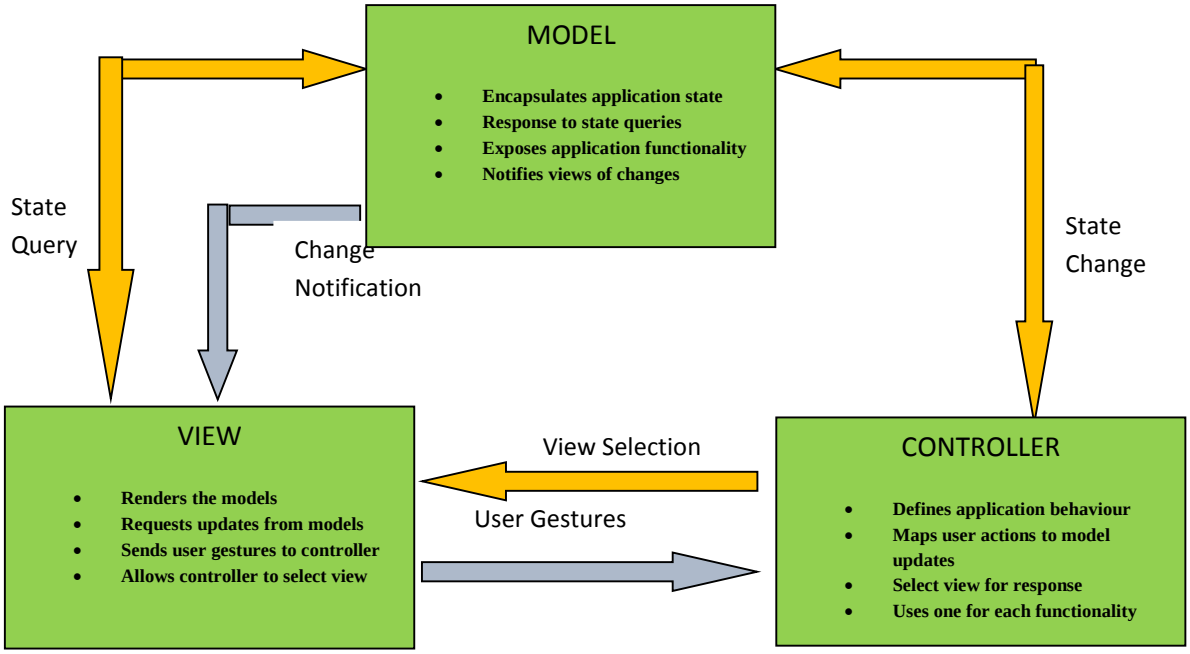
Model – The lowest level of the pattern which is responsible for maintaining data

View – This is responsible for displaying all or a portion of the data to the user.

Controller – Software Code that controllers the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the controller receives all requests for the application adn then works with the Model to prepare any data needed by the View. The View then uss the data prepared by the controller to generate a final presentable response, The MVC abstraction can be graphically represented as follows.

MVC (Model View Controller) Diagram



Method Invocations

Events

Role of Users in the system:

User/Blogger:

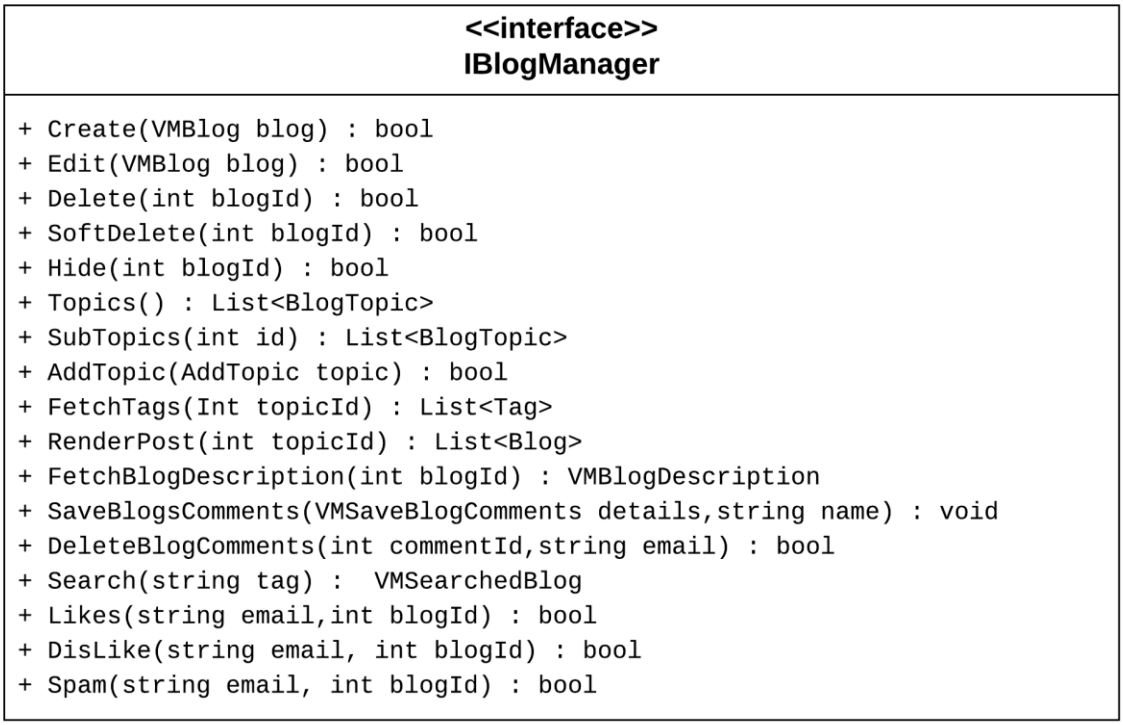
A registered User can create article and add tags to it. A non-registered User and a Registered User can view all the approved articles in the main page, and only a registered user can comment on the article. A user can able to view the author's Profile Details. The moderators can review the articles and that articles will be shown to the admin for further approval. Moderator can also create an article. A moderator can comment on the articles.

Admin:

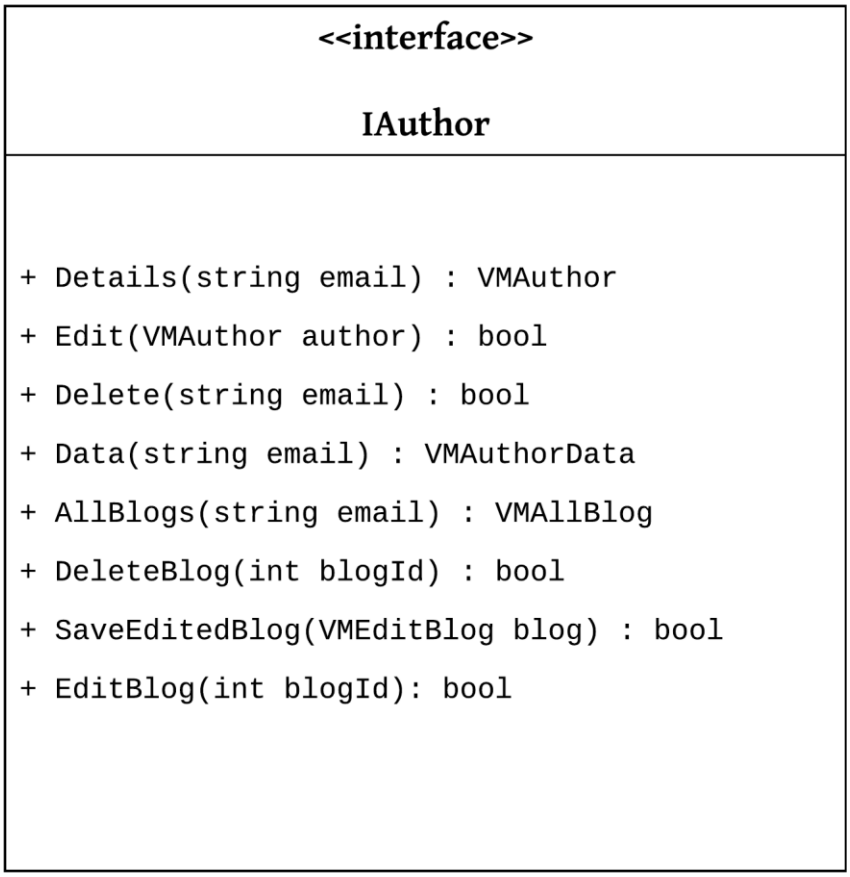
Admin can login into his account and he will view the approved articles. S/he select the eligible users to be moderators according to his/her articles and the rates on those articles. Admin also has the authorization to make a moderator a normal user if the moderator posts some irrelevant comments or articles.

4.2 Interface Design:

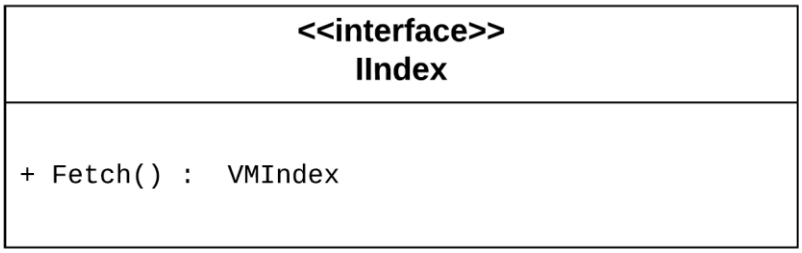
4.2.1 Interface for Blog



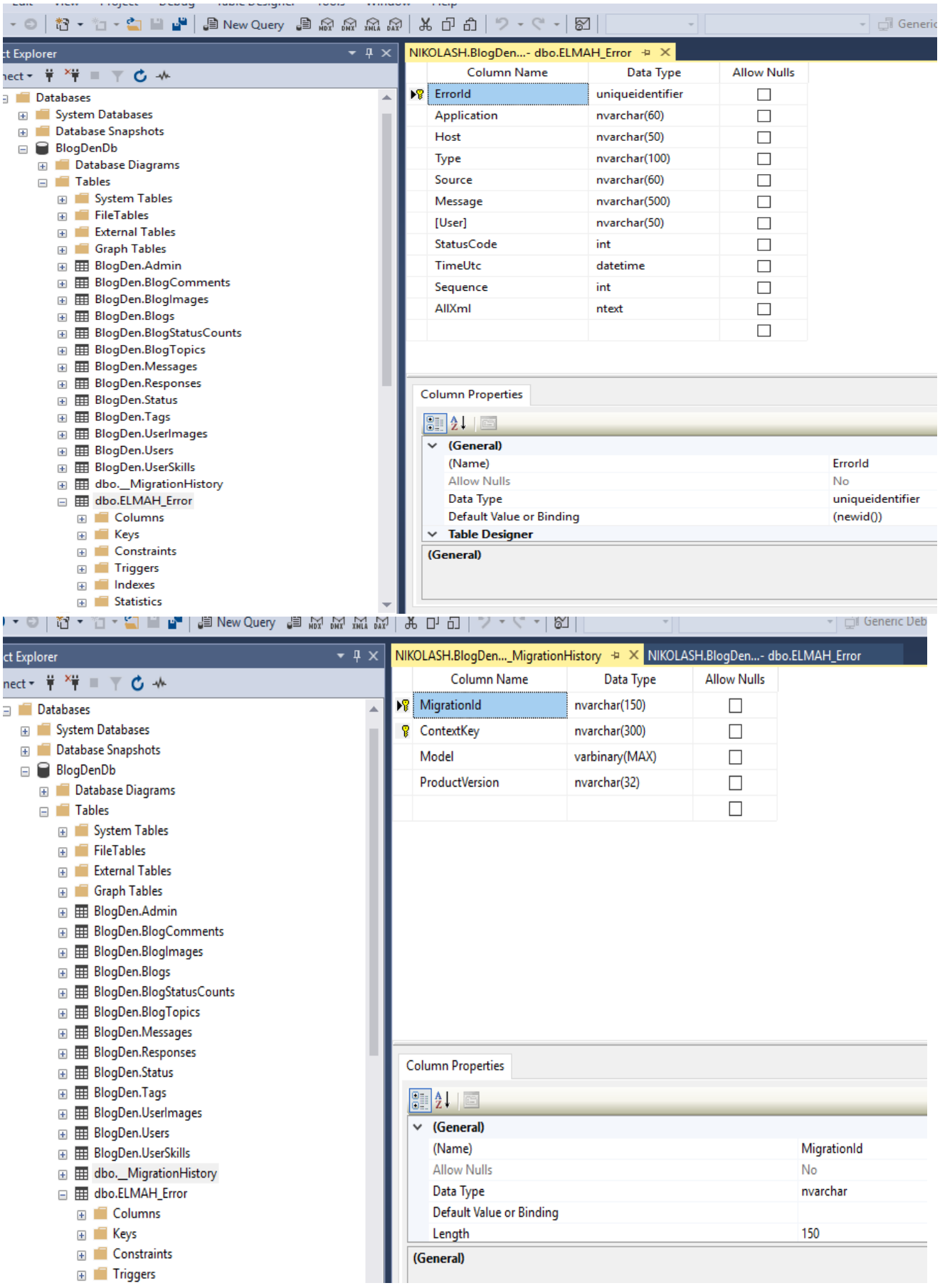
4.2.2 Interface for Author/Blogger

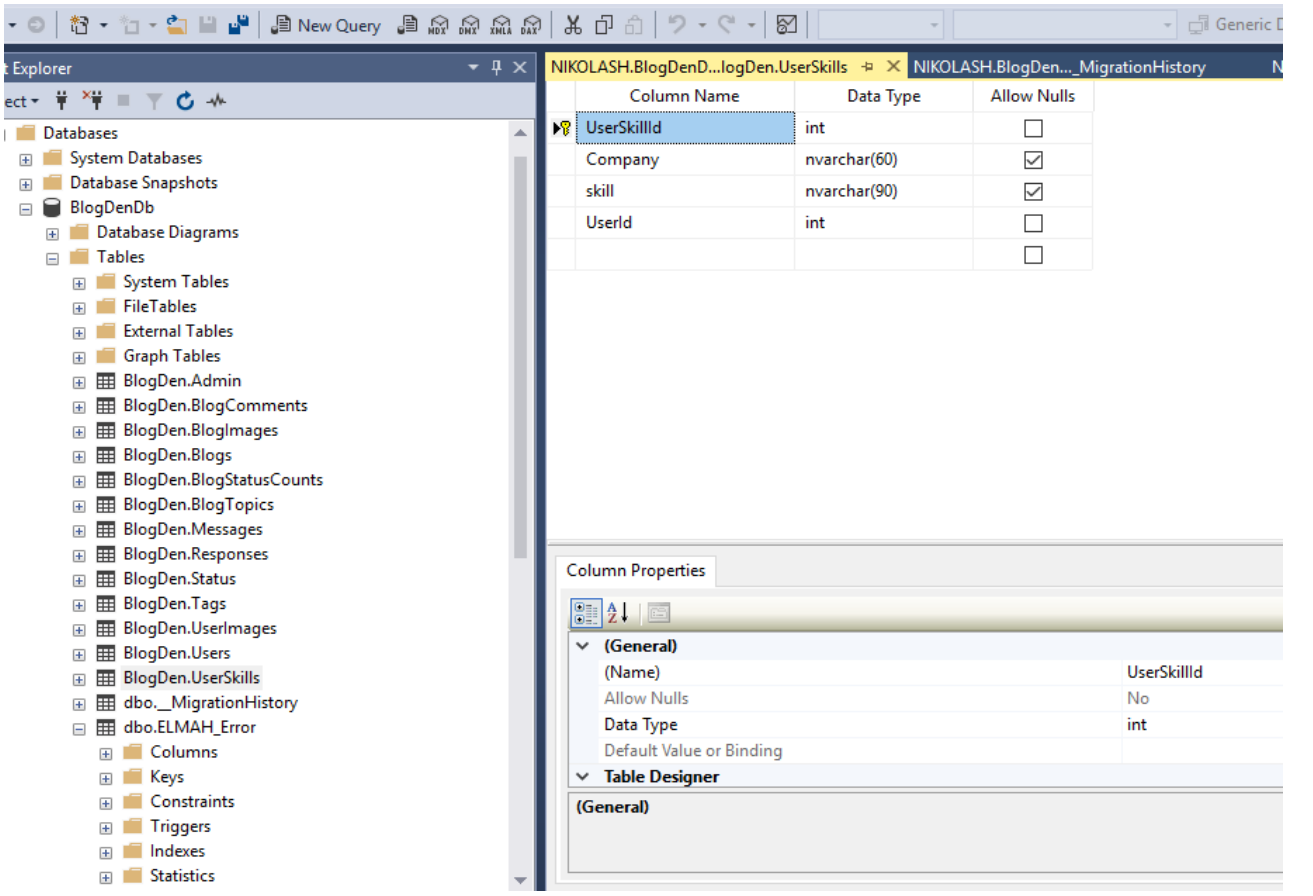


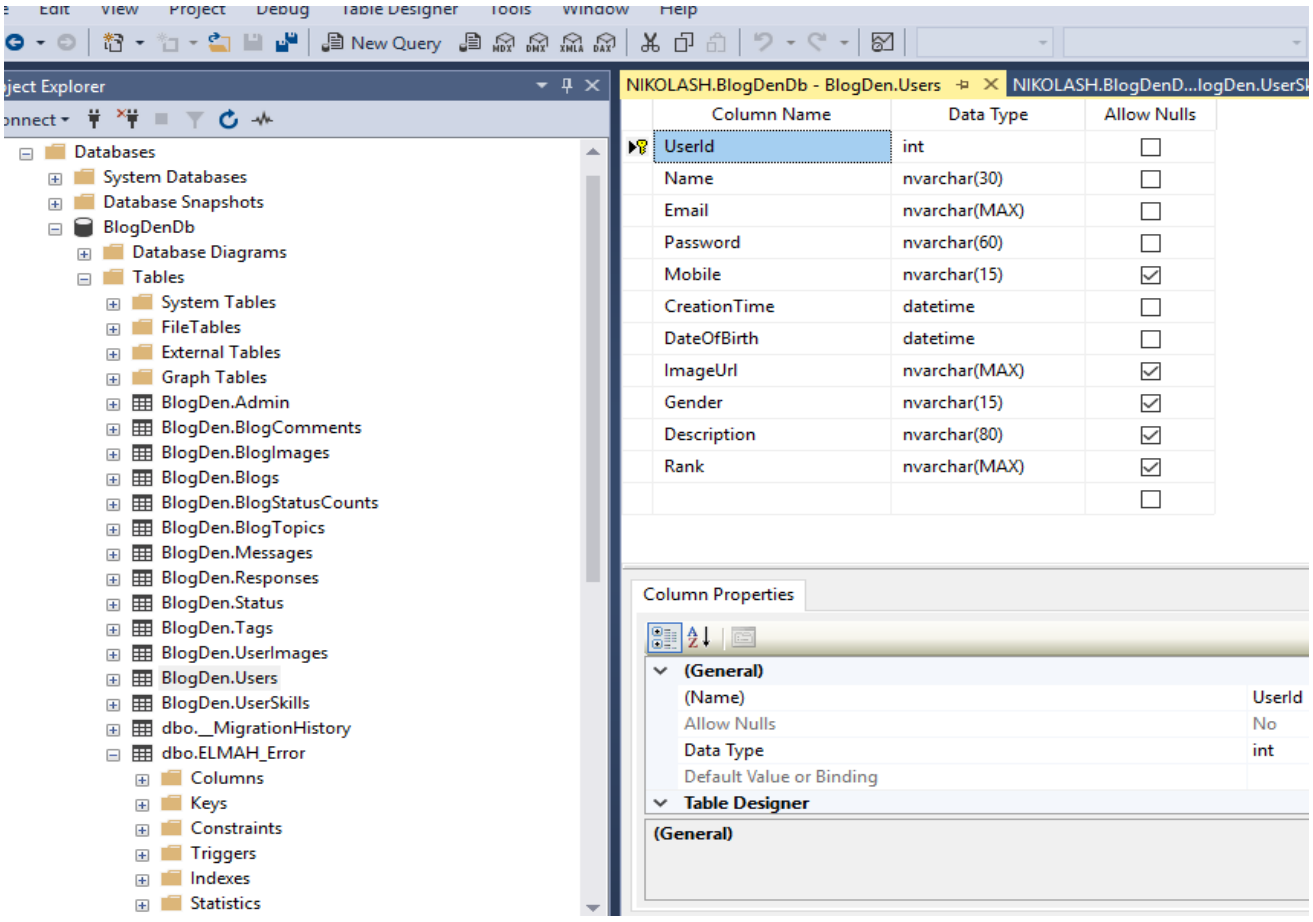
4.2.3InterfaceforContact,IndexPage

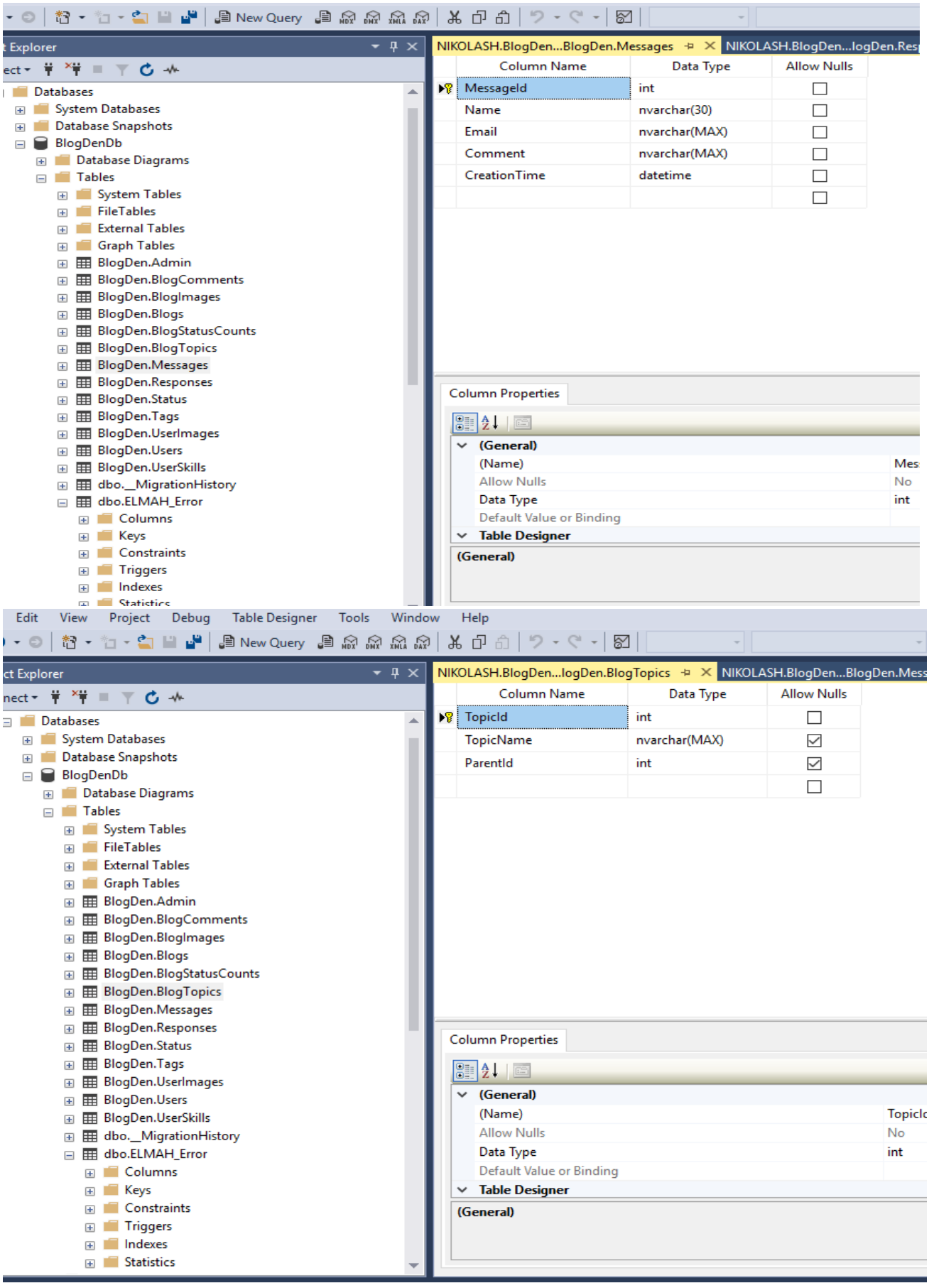


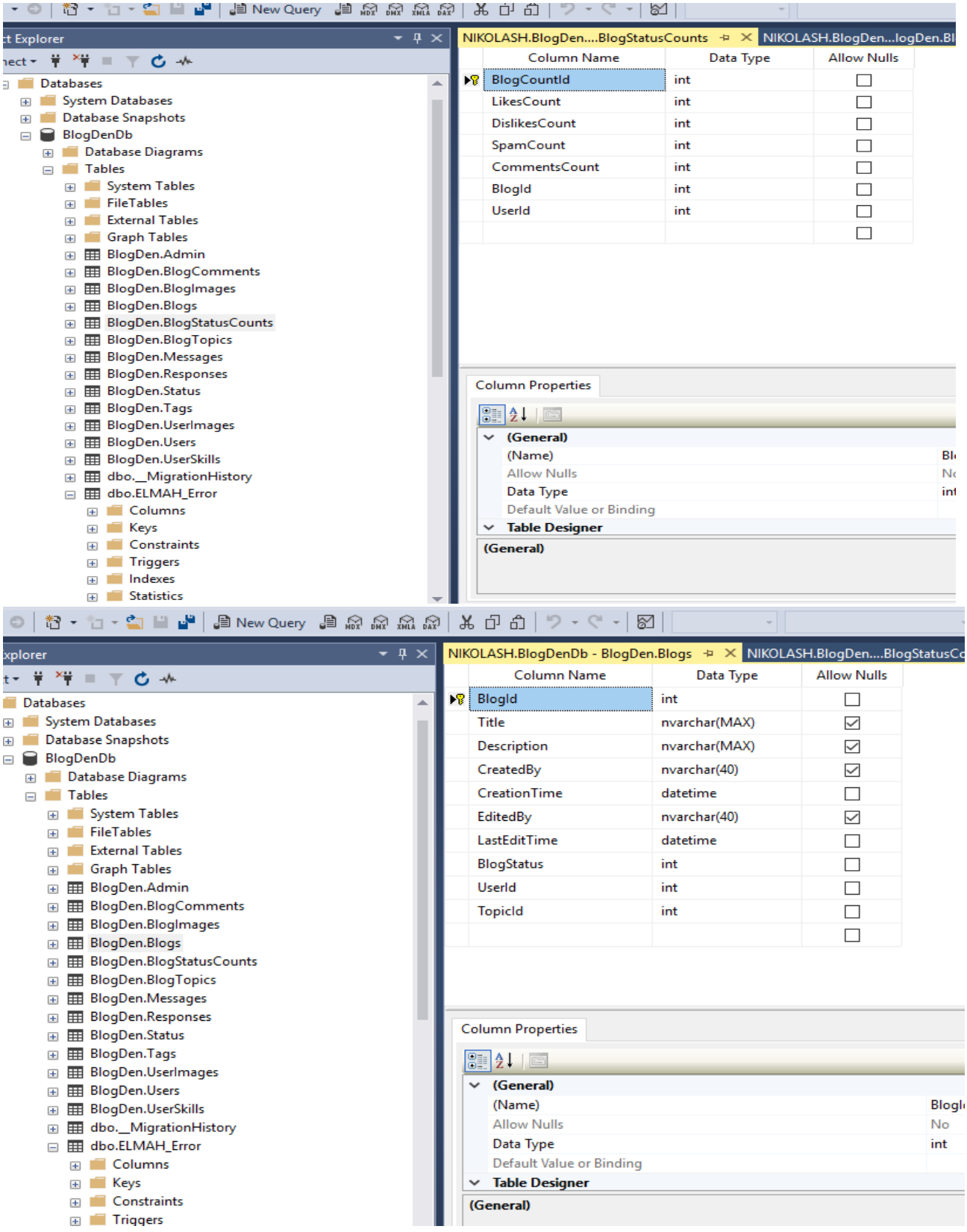
4.3 Database Table Design:




New Query
Explorer
Databases
System Databases
Database Snapshots
BlogDenDb
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
BlogDen.Admin
BlogDen.BlogComments
BlogDen.BlogImages
BlogDen.Blogs
BlogDen.BlogStatusCounts
BlogDen.BlogTopics
BlogDen.Messages
BlogDen.Responses
BlogDen.Status
BlogDen.Tags
BlogDen.UserImages
BlogDen.Users
BlogDen.UserSkills
dbo.__MigrationHistory
dbo.ELMAH_Error
Columns
Keys
Constraints
Triggers
Indexes
Statistics
NIKOLASH.BlogDenD...logDen.UserSkills
NIKOLASH.BlogDen..._MigrationHistory
Column Name
Data Type
Allow Nulls
UserSkillId
int
Company
nvarchar(60)
skill
nvarchar(90)
UserId
int
Column Properties
(General)
(Name)
UserSkillId
Allow Nulls
No
Data Type
int
Default Value or Binding
Table Designer
(General)


Edit
View
Project
Debug
Table Designer
Tools
Window
Help
New Query
Object Explorer
Databases
System Databases
Database Snapshots
BlogDenDb
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
BlogDen.Admin
BlogDen.BlogComments
BlogDen.BlogImages
BlogDen.Blogs
BlogDen.BlogStatusCounts
BlogDen.BlogTopics
BlogDen.Messages
BlogDen.Responses
BlogDen.Status
BlogDen.Tags
BlogDen.UserImages
BlogDen.Users
BlogDen.UserSkills
dbo.__MigrationHistory
dbo.ELMAH_Error
Columns
Keys
Constraints
Triggers
Indexes
Statistics
NIKOLASH.BlogDenDb - BlogDen.Users
NIKOLASH.BlogDenD...logDen.UserSk
Column Name
Data Type
Allow Nulls
UserId
int
Name
nvarchar(30)
Email
nvarchar(MAX)
Password
nvarchar(60)
Mobile
nvarchar(15)
CreationTime
datetime
DateOfBirth
datetime
ImageUrl
nvarchar(MAX)
Gender
nvarchar(15)
Description
nvarchar(80)
Rank
nvarchar(MAX)
Column Properties
(General)
(Name)
UserId
Allow Nulls
No
Data Type
int
Default Value or Binding
Table Designer
(General)


New Query
Explorer
Databases
System Databases
Database Snapshots
BlogDenDb
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
BlogDen.Admin
BlogDen.BlogComments
BlogDen.BlogImages
BlogDen.Blogs
BlogDen.BlogStatusCounts
BlogDen.BlogTopics
BlogDen.Messages
BlogDen.Responses
BlogDen.Status
BlogDen.Tags
BlogDen.UserImages
BlogDen.Users
BlogDen.UserSkills
dbo.__MigrationHistory
dbo.ELMAH_Error
Columns
Keys
Constraints
Triggers
Indexes
Statistics
NIKOLASH.BlogDen...BlogDen.Messages
Column Name
Data Type
Allow Nulls
MessageId
int
Name
nvarchar(30)
Email
nvarchar(MAX)
Comment
nvarchar(MAX)
CreationTime
datetime
Column Properties
(General)
(Name)
Allow Nulls
Data Type
Default Value or Binding
Table Designer
(General)
Edit
View
Project
Debug
Table Designer
Tools
Window
Help
NIKOLASH.BlogDen...logDen.BlogTopics
TopicId
int
TopicName
nvarchar(MAX)
ParentId
int


New Query
Object Explorer
Databases
System Databases
Database Snapshots
BlogDenDb
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
BlogDen.Admin
BlogDen.BlogComments
BlogDen.BlogImages
BlogDen.Blogs
BlogDen.BlogStatusCounts
BlogDen.BlogTopics
BlogDen.Messages
BlogDen.Responses
BlogDen.Status
BlogDen.Tags
BlogDen.UserImages
BlogDen.Users
BlogDen.UserSkills
dbo.__MigrationHistory
dbo.ELMAH_Error
Columns
Keys
Constraints
Triggers
Indexes
Statistics
NIKOLASH.BlogDen....BlogStatusCounts
Column Name
Data Type
Allow Nulls
BlogCountId int
LikesCount int
DislikesCount int
SpamCount int
CommentsCount int
BlogId int
UserId int
Column Properties
(General)
(Name)
Allow Nulls
Data Type
Default Value or Binding
Table Designer
(General)
New Query
Object Explorer
Databases
System Databases
Database Snapshots
BlogDenDb
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
BlogDen.Admin
BlogDen.BlogComments
BlogDen.BlogImages
BlogDen.Blogs
BlogDen.BlogStatusCounts
BlogDen.BlogTopics
BlogDen.Messages
BlogDen.Responses
BlogDen.Status
BlogDen.Tags
BlogDen.UserImages
BlogDen.Users
BlogDen.UserSkills
dbo.__MigrationHistory
dbo.ELMAH_Error
Columns
Keys
Constraints
Triggers
NIKOLASH.BlogDenDb - BlogDen.Blogs
Column Name
Data Type
Allow Nulls
BlogId int
Title nvarchar(MAX)
Description nvarchar(MAX)
CreatedBy nvarchar(40)
CreationTime datetime
EditedBy nvarchar(40)
LastEditTime datetime
BlogStatus int
UserId int
TopicId int
Column Properties
(General)
(Name)
Allow Nulls
Data Type
Default Value or Binding
Table Designer
(General)

5. SYSTEM IMPLEMENTATION

5.1.1 Implementation

Implementation is the stage in the project where the theoretical design is turned into the working system and is giving confidence to the new system for the users i.e. will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of method to achieve the changeover, an evaluation, of change over methods. A part from planning major task of preparing the implementation is education of users. The more complex system is implemented, the more involved will be the system analysis and design effort required just for implementation. An implementation coordinating committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation for the system. According to this plan, the activities are to be carried out, discussions may regard the equipment has to be acquired to implement the new system Implementation is the final and important phase. The most critical stage is in achieving a successful new system and in giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transaction while using the new system. The major elements of implementation plan are test plan, training plan, equipment installation plan, and a conversion plan.

There are three types of implementation:

1) Implementation of a computer system to replace a manual system.
2) Implementation of a new computer system to replace an existing system.
3) Implementation of a modified application to replace an existing one, using the same computer.

Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it. It has been observed that even the best system cannot show good result if the analysts managing the implementation do not attend to every important detail. This is an area where the systems analysts need to work with utmost care.

5.1.2 Implementation Phase

1. Training personnel

2. Conversion Procedures

3. Post-implementation review

Training of Personal Involved with System Even well designed system can succeed or fail because of the way they are operated and used. Therefore, the quality of training received by the personal involved with the system in various capacities helps or hinders and may even prevent the successful implementation of management information system. System Operators Training Running of the system successfully depend on the personnel working in the Computer Centre. They are Responsible for providing the necessary

support. Their training must ensure that they are able to handle all possible operations, both routine and extra-ordinary in nature. If the system calls for the installation of new equipment, such as new computer system, special terminals or different data entry machines, the operators training should include such fundamentals as how to turn the equipment on and use it, how to power off and a knowledge of what constitutes normal operations. The operators should also be trained on different type of malfunctioning, how to recognize them and what steps should also be taken whenever they arise. User Training User may be trained on use equipment, particularly in the case where, e.g. a microcomputer is in use and individual involved is both operator and user. In such cases, user must be given training on how to operate and user. In such cases, user must be given training on how to operator the system also. Questions that may be trivial to the analyst, such as how to turn on a terminal, how to insert a diskette into a micro-computer or when it is safe to turn off equipment without danger of data loss are significant problems to new users who are not familiar. In most of the cases user training deals with the operation of the system itself, with proper attention given to data handling techniques. It is imperative that users be properly trained in methods of entering transaction, editing data, formulating inquiries, deleting and inserting of records. No training is complete without familiarizing users with simple systems maintenance activities. Weakness in any aspect of training may lead of awkward situation that creates user frustration and error.

Conversion Methods A conversion is the process of changing from the old system to the new one. It must be properly planned and executed. Four methods are common in use. They are Parallel Systems, Direct Conversion, Pilot System and Phase in method. Each method should be considered in the light of the opportunities that it offers and problems that it may create. In general, system conversion should be accomplished in shortest possible time. Long conversion periods create problems for all persons involved including both analysts and users. Parallel System The most secure method of converting from an old to new system is to run both systems in parallel. This method is safest one because it ensures that in case of any problem in using new system, the organization can still fall back to the old system without the loss of time and money. The Disadvantages of Parallel Systems Approach Are:

1. It doubles operating costs.

2. The new system may not get fair trial.

Direct Conversion This method converts from the old system to new system abruptly, sometimes over a weekend or even overnight. The old system is used until a planned conversion day, when it is replaced by the new system. Pilot System Pilot approach is often preferred in the case of the new system which involves new techniques or some drastic changes in organization performance. In this method, a working version of the system is implemented in one part of the organization, such as a single work area or department.

Phase –In- Method This method is used when it is not possible to install a new system throughout an organization all at once. The conversion of files, training of personnel or arrival of equipment may force the staging of the implementation over a period of time, ranging from weeks to months. Post Implementation Review After the system is implemented and conversion is complete, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. A post implementation review measures the systems performance against predefined requirement. It determines how well the system continues to meet the performance specifications.

5.2.1 System Testing

System Testing System testing is a critical element of quality assurance and represents the ultimate review of analysis, design and coding. Test case design focuses on a set of techniques for the creation of test because that meet overall testing objective. When a system is developed it is hoped that it performs properly. The main purpose of testing an information system is to find the errors and correct them. The scope of system testing should include both manual and computerized operations. System testing is comprehensive evaluation of the programs, manual procedures, computer operations and controls. System testing is the process of checking whether the developed system is working according to the objective and requirement. All testing is to be conducted in accordance to the test conditions specified earlier. This will ensure that the test coverage meets the requirements and that testing is done in a systematic manner.

TEST CHARACTERS :

1. A good test has a high probability of finding an error.

2. A good test is not redundant.

3. A good test should be “best of breed”.

5.2.2 Levels of Testing :-

BLACK BOX TESTING:

The method of Black Box Testing is used by the software engineer to derive the required results of the test cases:

1. Black Box Testing alludes to test that are conducted at the software interface. Client Needs Acceptance Testing Requirements System Testing Design Integration Testing Code Unit Testing

2. A Black Box Test examines some fundamental aspect of a system with little regard for the internal logic structure of the software.

3. A limited number of important logical paths can be selected and exercised.

Black box testing was performed to find errors in the following categories :-

- Incorrect or missing functions
- Graphics error.
- Errors in data in binary format.
- Error in data in integer format.
- File error.
- Pointer error
- Variable error

WHITE BOX TESTING:

White Box Testing is sometimes called Glass Box Testing.

Using White Box Testing method's the software engineer can derive the following test cases:

1. Guarantee that all independent paths within a module have been exercised at least once.

2. Exercise all logical decisions on their true and false sides.

3. Execute all loops at their boundaries and within their operational bounds.

In White Box Testing efforts were made to handle the following:-

• Number of input parameters equal to number of arguments.

•Parameters and arguments attributes match.

• Number of arguments transmitted is called modules equal to attributes of parameters.

•Unit system of argument transmitted is called modules equal unit system of parameter.

• Number of attributes and order of arguments to build in functions correct.

•Any references to parameters not associated to build in functions correct.

•Input only arguments altered.

•Global variable definition consistent across module.

•Files attributes correct.

Unit Testing

Code testing was carried out to see the correctness of the logic involved and the correctness of the modules. Tests were conducted based upon sample and live data as well. All the modules are checked separately for assuming correctness and accuracy in all the calculations. Specification Testing It examines the specification stating about what program should do and how it performs under various conditions. This testing strategy is better strategy since it focuses on the way the software is expected to work. Integration Testing The next level testing that was performed is often referred to as integration testing. During this phase many unit test modules were combined into subsystems, which were then tested. The goal here was to see if modules can be integrated properly. Here the emphasis was on testing interfaces between different constituent modules of system. Functionality Testing Here the entire software system was tested. The reference document for this process is the requirements document, and the goal was to see if software solution meets its requirements. This level of testing is essentially a validation exercise, and in many situations it is the only validation activity. Stress Testing Proxy server developed for the specified purpose was testing under heavy load, i.e. a large no. of clients had made to sit in lab and were asked to send requests for logging in and then were asked to request for text on internet. System responded to request as desired. Acceptance Testing Acceptance was performed in the real environment with realistic data of the client to demonstrate if the software developed is working satisfactorily. Here the main focus was on the external behaviour of the system; the internal logic of the program was not emphasized. Test Data and Test Cases The primary objective of test case design is to derive a set of tests that have the highest likelihood of uncovering errors in software. The test case specification is the major activity in the testing process. Careful selection of test cases that satisfy the criterion on approach specified is essential for proper testing. Various characteristics of test cases that are required for portal are:

1) A good test has a high probability of finding an error.
2) A good test is not redundant.
3) A good test should be "Best of Breed".

4) A good test should be neither too simple not too complex.

Overview of Testing

1. Testing: Testing involves executing the program (or part of it) using sample data and inferring from the output whether the software performs correctly or not. This can be done either during module development (unit testing) or when several modules are combined (system testing).

2. Direct Testing: Defect testing is testing for situation where the program does not meet its functional specification. Performance testing tests a system's performance or reliability under realistic loads. This may go some way to ensuring that the program meets its non- functional requirements.

Typical levels of testing:

Acceptance testing - whole system with real data (involve customer, user, etc.) .

Alpha testing is acceptance testing with a single client (common for bespoke systems).

Beta testing involves distributing system to potential customers to use and provide feedback. In, this project, Beta testing has been followed. This exposes system to situations and errors that might not be anticipated by us.

5.2.3 Debugging

Debugging is a cycle of detection, location, repair and test. Debugging is a hypothesis testing process. When a bug is detected, the tester must form a hypothesis about the cause and location of the bug. Further examination of the execution of the program (possible including many returns of it) will usually take place to confirm the hypothesis. If the hypothesis is demonstrated to be incorrect, a new hypothesis must be formed. Debugging tools that show the state of the program are useful for this, but inserting print statements is often the only approach. Experienced debuggers use their knowledge of common and/or obscure bugs to facilitate the hypothesis testing process. After fixing a bug, the system must be reset to ensure that the fix has worked and that no other bugs have been introduced. This is called regression testing. In principle, all tests should be performed again but this is often too expensive to do. Test Planning Testing needs to be planned, to be cost and time effective. Planning is setting out standards for tests. Test plans set out the context in which individual engineers can place their own work. Typical test plan contains: Interface Testing: Usually done at integration stage when modules or sub-systems are combined. Objective is to detect errors or invalid assumptions about interfaces between modules. Reason these are not shown in unit testing is that test case may perpetuate same incorrect assumptions made by module designer. Particularly important when OO development has been used.

Four Types of Interface

1. Parameter: data (or occasionally function references) passed from one unit to another.

2. Shared memory: block of memory shared between units (e.g. global variable). One places data there and the other retrieves it.

3. Procedural: Object-Oriented or abstract data type form of interface, encapsulating several procedures.

4. Message passing: one sub-system requests a service by passing a message. Client-server interface also used by some OO architectures.

5.3. Sample Code:

5.3.1 Controllers

```
namespace MindfireSolutions.Controllers
{
   [Authorize]
   public class BlogController : Controller
   {
      /// <summary>
      /// Reference fot Blogmanager
      /// </summary>
      private readonly IBlogManager _blogManager;

      /// <summary>
      /// Constructor of BlogConrtoller
      /// </summary>
      /// <param name="blogManager"></param>
      public BlogController(IBlogManager blogManager)
      {
         _blogManager = blogManager;
      }

      /// <summary>
      /// Method for View of Blog Creation page with Blog Creation Fields.
      /// </summary>
      /// <returns>View for Blog Creation.</returns>
      [HttpGet]
      public ActionResult PostBlog()
      {
         var email = FormsAuthentication.Decrypt(Request.Cookies[FormsAuthentication.FormsCookieName].Value).Name;
         DAL db = new DAL();
         var userData = db.Users.Where(m => m.Email == email).FirstOrDefault();
         var data = new VMBlog()
```

```
    {
        UserId = userData.UserId,
        UserImage = userData.ImageUrl,
        Name = userData.Name,
        Email=userData.Email
    };
    return View(data);
}



/// <summary>
/// Method For adding Blog To the Database
/// </summary>
/// <param name="blog"></param>
/// <returns>Blog View to Add more Blogs.</returns>
[HttpPost]
[ValidateInput(false)]
[ValidateAntiForgeryToken]
public ActionResult PostBlog(VMBlog blog)
{
    var blogCreated = _blogManager.Create(blog);
    if (blogCreated)
    {
        return RedirectToAction("Dashboard", "Author");
    }
    return RedirectToAction("PostBlog", "Blog");
}


/// <summary>
/// Method to Render Post on Clicking 0f Submenu on Index page
/// </summary>
/// <param name="topicId"></param>
```

```
/// <returns></returns>
[AllowAnonymous]
public ActionResult RenderPost(int topicId)
{
   var blogResult = _blogManager.RenderPost(topicId);
   if (blogResult != null)
   {
      return View(blogResult);
   }
   return View();
}



public ActionResult BlogTopic()
{
   var data = _blogManager.Topics();
   return PartialView("_BlogTopic", data);
}

[HttpPost]
public JsonResult GetMembers(int topicId)
{
   var data = _blogManager.SubTopics(topicId);
   return Json(data);
}

[ValidateAntiForgeryToken]
[HttpPost]
public ActionResult AddTopic(AddTopic topic)
{
   var topicAdded = _blogManager.AddTopic(topic);
   return Json(new { success = topicAdded }, JsonRequestBehavior.AllowGet);
}
```

```
/// <summary>
///
/// </summary>
/// <param name="topicId"></param>
/// <returns></returns>
[HttpPost]
public JsonResult FetchTags(int topicId)
{
   var data = _blogManager.FetchTags(topicId);
   if (data != null)
   {
      return Json(data);
   }
   return Json(null);
}

/// <summary>
/// This function gets called from Render post view to get details of blog clicked
/// </summary>
/// <param name="blogId"></param>
/// <returns></returns>
[AllowAnonymous]
public ActionResult GetBlogDescription(int blogId)
{
   var blogDescription = _blogManager.FetchBlogDescription(blogId);
   return View(blogDescription);
}

/// <summary>
/// This is to save comments on blogs by authenticated user only
/// </summary>
/// <param name="details"></param>
```

```
/// <returns></returns>
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult SaveBlogComments(VMSaveBlogComments details)
{
   if(details!=null)
   {
      string email= FormsAuthentication.Decrypt(Request.Cookies[FormsAuthentication.FormsCookieName].Value).Name;
      _blogManager.SaveBlogsComments(details,email);
   }
   return RedirectToAction("GetBlogDescription", "Blog", new { details.BlogId });
}

/// <summary>
/// This is to Delete comments on blogs by User whose had commented
/// </summary>
/// <param name="commentId"></param>
/// <returns></returns>
public JsonResult DeleteVisitorComments(int commentId,string email)
{
   var userEmail = (FormsAuthentication.Decrypt(Request.Cookies[FormsAuthentication.FormsCookieName].Value).Name);
   if (email==userEmail)
   {
      if (_blogManager.DeleteBlogComments(commentId, userEmail))
      {
         return Json(new { success = true, }, JsonRequestBehavior.AllowGet);
      }
      return Json(new { success = false, }, JsonRequestBehavior.AllowGet);
   }
   return Json(new { success = false, }, JsonRequestBehavior.AllowGet);

}
[AllowAnonymous]
```

```
[HttpGet]
public ActionResult Search(string SearchTag)
{
   var data = _blogManager.Search(SearchTag);
   ViewBag.Message = SearchTag;
   return View(data);
}

[HttpPost]
public ActionResult Like(int blogId)
{
   string email = FormsAuthentication.Decrypt(Request.Cookies[FormsAuthentication.FormsCookieName].Value).Name;
   var flag = _blogManager.Likes(email,blogId);
   if (flag)
   {
      return Json(true);
   }
   return Json(false);

}
[HttpPost]
public ActionResult Dislike(int blogId)
{
   string email = FormsAuthentication.Decrypt(Request.Cookies[FormsAuthentication.FormsCookieName].Value).Name;
   var flag = _blogManager.DisLike(email, blogId);
   if (flag)
   {
      return Json(true);
   }
   return Json(false);

}
[HttpPost]
```

```
    public ActionResult Spam(int blogId)
    {
        string email = FormsAuthentication.Decrypt(Request.Cookies[FormsAuthentication.FormsCookieName].Value).Name;
        var flag = _blogManager.Spam(email, blogId);
        if (flag)
        {
            return Json(true);
        }
        return Json(false);

    }

  }
}
```

5.3.2 Views

```
@model IEnumerable<MindfireSolutions.Models.Blog>
@{
  ViewBag.Title = "RenderPost";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="container-fluid">
  <div>
    @if (Model.Count() > 0)
    {
      foreach (var item in Model)
      {
        <div class="row margin-top20">
          <div class="col-xs-8">
            <h2 class="topic-name">Result For @item.BlogTopic.TopicName...</h2>
          </div>
          <div class="col-xs-4">
```

```
            @using (Html.BeginForm("Search", "Blog", FormMethod.Get, new { @enctype = "multipart/form-data", @id =
"RegistrationForm" }))

            {

              <div class="right-inner-addon">

                <button type="submit" class="astext"><i class="fa fa-search text-primary"></i></button>

                <input id="SearchTag" name="SearchTag" type="text" class="form-control text-center" placeholder="Search Tags Topics or
your favourite Bloggers" autocomplete="off" />

              </div>

            }



          </div>

        </div>

        break;

      }

      <div class="container">

        <div class="row">

          <div class="col-sm-8  blog-align">

            @foreach (var item in Model)

            {

              <p class="text-info margin-top2"><a href="/Blog/GetBlogDescription?blogId=@item.BlogId" class="font15">
@item.Title</a></p>

              <div class="minimize-text"><a
href="/Blog/GetBlogDescription?blogId=@item.BlogId">@Html.Raw(System.Web.HttpUtility.HtmlDecode(item.Description))</a></div>

                <div class="col-sm-offset-8">

                  <ul class="blog-display">

                    Blogger Name : <a href="/Author/Dashboard"><span class="decorate">@item.User.Name</span></a>

                    Created On :<span>@item.CreationTime</span>



                </div>

                <hr />

            }

          </div>

          <div class="col-sm-4 margin-top20">

            <h3 class="h3 text-primary text-center">Related Topics</h3>

            <hr/>
```

```
            </div>
          </div>
        </div>
    }
    else
    {
        <div class="row">
          <div class="col-xs-8">
            <h2 class="topic-name">No Results Found ...</h2>
          </div>
          <div class="col-xs-4 margin-top20 margin-bottom20">
            @using (Html.BeginForm("Search", "Blog", FormMethod.Get, new { @enctype = "multipart/form-data", @id = "RegistrationForm" }))
            {
              <div class="right-inner-addon">
                <button type="submit" class="astext"><i class="fa fa-search text-primary"></i></button>
                <input id="SearchTag" name="SearchTag" type="text" class="form-control text-center" placeholder="Search Tags Topics or your favourite Bloggers" autocomplete="off" />
              </div>
            }
          </div>
        </div>
    }
  </div>
</div>
```

5.3.3 Layout

```
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title</title>
    <link rel="icon" href="~/Images/ContentImages/TitleIcon.jpg">
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
    <link href="~/Content/Header.css" rel="stylesheet" />
    <link href="~/Content/bootstrap.css" rel="stylesheet" />
    <link href="~/Content/FooterStyle.css" rel="stylesheet" />
    <link href="~/Content/ModalStyle.css" rel="stylesheet" />
    <link href="~/Content/DashBoard.css" rel="stylesheet" />
    <link href="~/Content/Sidebar.css" rel="stylesheet" />
    <link href="~/Content/Profile.css" rel="stylesheet" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
    <script src="https://cdn.quilljs.com/1.3.6/quill.js"></script>
    <script src="https://cdn.quilljs.com/1.3.6/quill.min.js"></script>
    <link href="https://cdn.quilljs.com/1.3.6/quill.snow.css" rel="stylesheet">
    <link href="https://cdn.quilljs.com/1.3.6/quill.bubble.css" rel="stylesheet">
    <link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet" integrity="sha384-
wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN" crossorigin="anonymous">
</head>
<body>
    <nav class="navbar navbar-inverse" id="Top">
        <div class="container-fluid">
            <div class="container">
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#MyNavbar">
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    <a class="navbar-brand" href="/Home/Index">Blog<span class="text-brand">Den</span></a>
                </div>
                <div class="collapse navbar-collapse" id="MyNavbar">
```

```
            <ul class="nav navbar-nav">
              <a href="/Home/Index">Home</a>
              <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown" href="#">Topics<span class="caret"></span></a>
                <ul class="dropdown-menu">
                  @{Html.RenderAction("DynamicMenu", "Home");}


              <a href="#">About Us</a>
              <a href="#" data-toggle="modal" data-target="#UserComment" id="ContactAdmin">Contact</a>

            @Html.Partial("_LoginPartial")
          </div>
        </div>

    </div>
  </nav>
  <div>
    @RenderBody()

    <footer>
      <section id="Footer" class="section footer">
        <div class="container">
          <div class="row mar-bottom20 margin-top" data-andown="fadeIn" data-animation="animation">
            <div class="col-sm-12 text-center">
              <ul class="social-network social-circle">
                <a href="http://www.mindfiresolutions.com/pressreleases-archives.htm" target="_blank" class="icoRss" title="Rss"><i class="fa fa-rss"></i></a>
                <a href="https://www.facebook.com/Mindfire-Solutions-559849804098764/" target="_blank" class="icoFacebook" title="Facebook"><i class="fa fa-facebook"></i></a>
                <a href="https://twitter.com/mindfires?lang=en" class="icoTwitter" target="_blank" title="Twitter"><i class="fa fa-twitter"></i></a>
                <a href="http://www.mindfiresolutions.com/google-api-integration-maps-api-integration.htm" target="_blank" class="icoGoogle" title="Google"><i class="fa fa-google"></i></a>
```

```
                    <a href="https://in.linkedin.com/company/mindfire-solutions" class="icoLinkedin" target="_blank" title="Linkedin"><i class="fa fa-linkedin"></i></a>


                </div>
            </div>
            <div class="row text-center mar-bottom20">
                <ul class="footer-menu">
                    @Html.ActionLink("Home","Index","Home")
                    <a href="#">About us</a>
                    <a href="#">Privacy policy</a>
                    <a href="#" data-toggle="modal" data-target="#UserComment" id="ContactAdmin">Get In Touch</a>

            </div>
            <div class="row align-center">
                <div class="col-sm-12">
                    <p class="container text-center text-warning">&copy; @DateTime.Now.Year - BlogDen</p>
                </div>
            </div>
            <div class="credits">
                <a href="#"> Designed by MindfireSolutions</a>
            </div>
        </div>
    </section>
    <a href="#Top" class="scrollup"><i class="fa fa-chevron-up"></i></a>
</footer>
</div>


@*Modal PopUp for User Login*@
<div class="modal fade" id="UserLogin" tabindex="-1" role="dialog" aria-labelledby="ModalTitle" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
            <div class="modal-header">
```

```
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
            <p class="modal-title h3 text-center" id="ModalTitle">Login</p>
        </div>
        @using (Html.BeginForm(null, null, FormMethod.Post, new { @enctype = "multipart/form-data", @id = "Login" }))
        {
            @Html.AntiForgeryToken()
            <div class="modal-body">
                @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                <div class="row form-group">
                    <div class="col-xs-12">
                        <label class="control-label" for="EmailId">Email<span class="error">*</span></label>
                    </div>
                    <div class="col-xs-12">
                        <input type="email" class="form-control modal-form-control" id="EmailId" name="EmailId" value="" placeholder="Enter your Email"/>
                    </div>
                </div>
                <div class="row form-group">
                    <div class="col-xs-12">
                        <label class="control-label" for="PasswordValue">Password<span class="error">*</span></label>
                    </div>
                    <div class="col-xs-12">
                        <input type="password" class="form-control modal-form-control" id="PasswordValue" name="PasswordValue" value="" placeholder="Enter your Password" autocomplete="off" />
                    </div>
                </div>
            </div>
            <div class="modal-footer">
                <button class="btn btn-primary" data-dismiss="modal">Close</button>
                <button type="submit" class="btn btn-danger" id="LoginUser">Login</button>
            </div>
        }
```

```
        </div>
      </div>
    </div>



    @*Modal PopUp for Visitor Comments*@
    <div class="modal fade" id="UserComment" tabindex="-1" role="dialog" aria-labelledby="ModalTitle" aria-hidden="true">
      <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
            <p class="modal-title h3 text-center" id="ModalCommentTitle">Message</p>
          </div>
          @using (Html.BeginForm(null, null, FormMethod.Post, new { @enctype = "multipart/form-data", @id = "Comment" }))
          {
            @Html.AntiForgeryToken()
            <div class="modal-body">
              @Html.ValidationSummary(true, "", new { @class = "text-danger" })
              <div class="row form-group">
                <div class="col-xs-12">
                  <label class="control-label" for="VisitorName">Name<span class="error">*</span></label>
                </div>
                <div class="col-xs-12">
                  <input type="text" class="form-control modal-form-control" id="VisitorName" name="VisitorName" value=""
placeholder="Enter your Name" autocomplete="off" />
                </div>
              </div>
              <div class="row form-group">
                <div class="col-xs-12">
                  <label class="control-label" for="VisitorEmail">Email<span class="error">*</span></label>
                </div>
```

```
                    <div class="col-xs-12">
                        <input type="email" class="form-control modal-form-control" id="VisitorEmail" name="VisitorEmail" value="" placeholder="Enter your Email" autocomplete="off" />
                    </div>
                </div>
                <div class="row form-group">
                    <div class="col-xs-12">
                        <label class="control-label" for="VisitorComment">Comments<span class="error">*</span></label>
                    </div>
                    <div class="col-xs-12">
                        <textarea class="comment-control" id="VisitorComment" rows="4" cols="50" required name="VisitorComment" placeholder="Enter Your Comments" autocomplete="off"></textarea>
                    </div>
                </div>
            </div>
            <div class="modal-footer">
                <button class="btn btn-primary" data-dismiss="modal">Close</button>
                <button type="submit" class="btn btn-danger" id="SaveComments">Post</button>
            </div>
        }
    </div>
  </div>
</div>

<script src="~/scripts/jquery-1.10.2.js"></script>
<script src="~/scripts/bootstrap.js"></script>
<script src="~/scripts/Script.js"></script>
</body>
</html>
```

5.3.4 Partial View

```
@if (Request.IsAuthenticated)
{
```

```
   <ul class="nav navbar-nav navbar-right">


        @Html.ActionLink("Hello " + User.Identity.Name + "!", "Dashboard", "Author",new { email=User.Identity.Name }, htmlAttributes: new { title = "Manage" })


      <a href="/Register/UserLogout"><span class="glyphicon glyphicon-log-out"></span>Logout</a>



   <input type="hidden" value="@User.Identity.Name" id="GetEmail" />

}

else

{

   <ul class="nav navbar-nav navbar-right">

      <a href="/Register/UserRegistration"><span class="glyphicon glyphicon-user"></span> Sign Up</a>

      <a href="#" data-toggle="modal" data-target="#UserLogin"><span class="glyphicon glyphicon-log-in"></span> Login</a>


}


<nav class="navbar navbar-default sidebar" role="navigation" id="SideNavHeight">

   <div class="container-fluid">

      <ul class="nav navbar-nav">

         <li class="active"><a class="nav-icon-user" href="#"><img src="@Model.UserImage" alt="img" id="DashboardImage"/> <label> @Model.Name</label></a>

         <li class="divider">

         <a href="/Author/Dashboard">Dashboard<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-home"></span></a>

         <li class="divider">

         <a href="/Author/AuthorProfile">Profile<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-user"></span></a>

         <li class="divider"><li class="divider">

         <a href="/Blog/PostBlog">Create Blog<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-edit"></span></a>

         <li class="divider">

         <a href="/Author/EditProfile">Edit Profile<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-edit"></span></a>
```

```
        <li class="divider">
        <a href="#">Add Personal Details<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-home"></span></a>
        <li class="divider">
        <a href="/Auther/Delete"onclick="return confirm('Are you sure want to Delete your Account ?')">Delete Account<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-trash"></span></a>
        <li class="divider">
        <a href="#">Your Blogs<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-check"></span></a>
        <li class="divider">
        <a href="#">Archived Blogs<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-download-alt"></span></a>
        <li class="divider">
        <a href="#">Drafted Contents<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-bookmark"></span></a>
        <li class="divider">
        <a href="#">Settings<span style="font-size:16px;" class="pull-right showopacity glyphicon glyphicon-cog"></span></a>

    </div>
</nav>
```

5.3.5 View Models

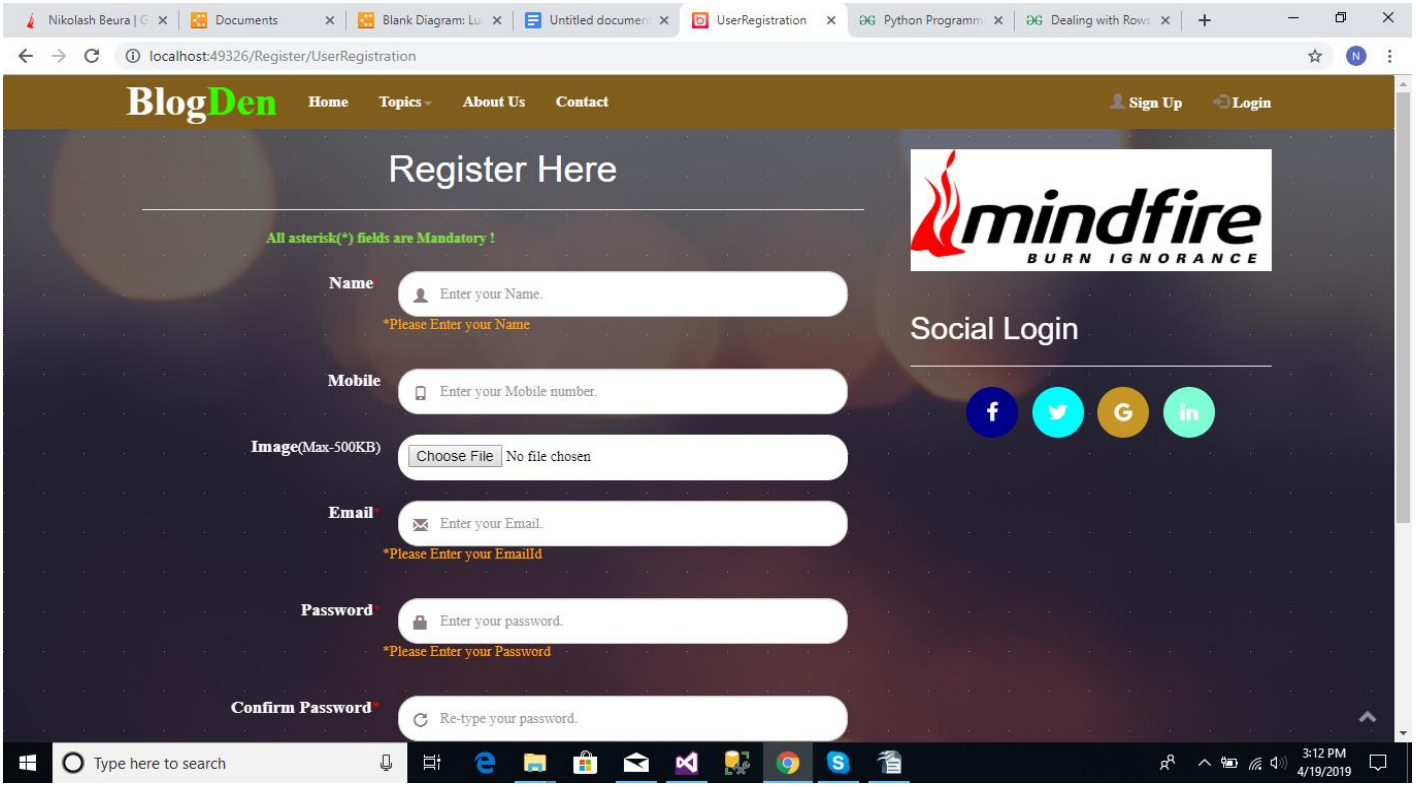
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace MindfireSolutions.ViewModel
{
    public class VMUserLogin
    {
        [Required(ErrorMessage = "*Please Enter your Email-Id ")]
```

```
        public string EmailId { get; set; }


        [Required(ErrorMessage = "*Please Enter your Password ")]
        public string PasswordValue { get; set; }
    }
}
using MindfireSolutions.CustomAttribute;
using System.ComponentModel.DataAnnotations;
using System.Web;

namespace MindfireSolutions.ViewModel
{
    public class VMUser
    {
        [Required(ErrorMessage = "*Please Enter your Name")]
        [MinLength(2, ErrorMessage = "*Minimum length should be 2")]
        [MaxLength(30, ErrorMessage = "*Maxlength should be 30")]
        [RegularExpression("^[a-zA-Z ]+$", ErrorMessage = "*Only Alphabets are allowed")]
        public string Name { get; set; }


        [Required(ErrorMessage = "*Please Enter your EmailId")]
        [EmailAddress]
        public string Email { get; set; }


        [Required(ErrorMessage = "*Please Enter your Password ")]
        [RegularExpression("^(?=.*[0-9])(?=.*[A-Z])(?=.*[!@#$%^&*])[a-zA-Z0-9!@#$%^&*]{8,}$", ErrorMessage = "Password should contain
atleast one uppercase+lowercase+digit+special character respectively")]
        public string Password { get; set; }


        [Required(ErrorMessage = "*Please Enter your Password ")]
        [Compare("Password", ErrorMessage = "*Passwords do not match.")]
        public string ConfirmPassword { get; set; }
```

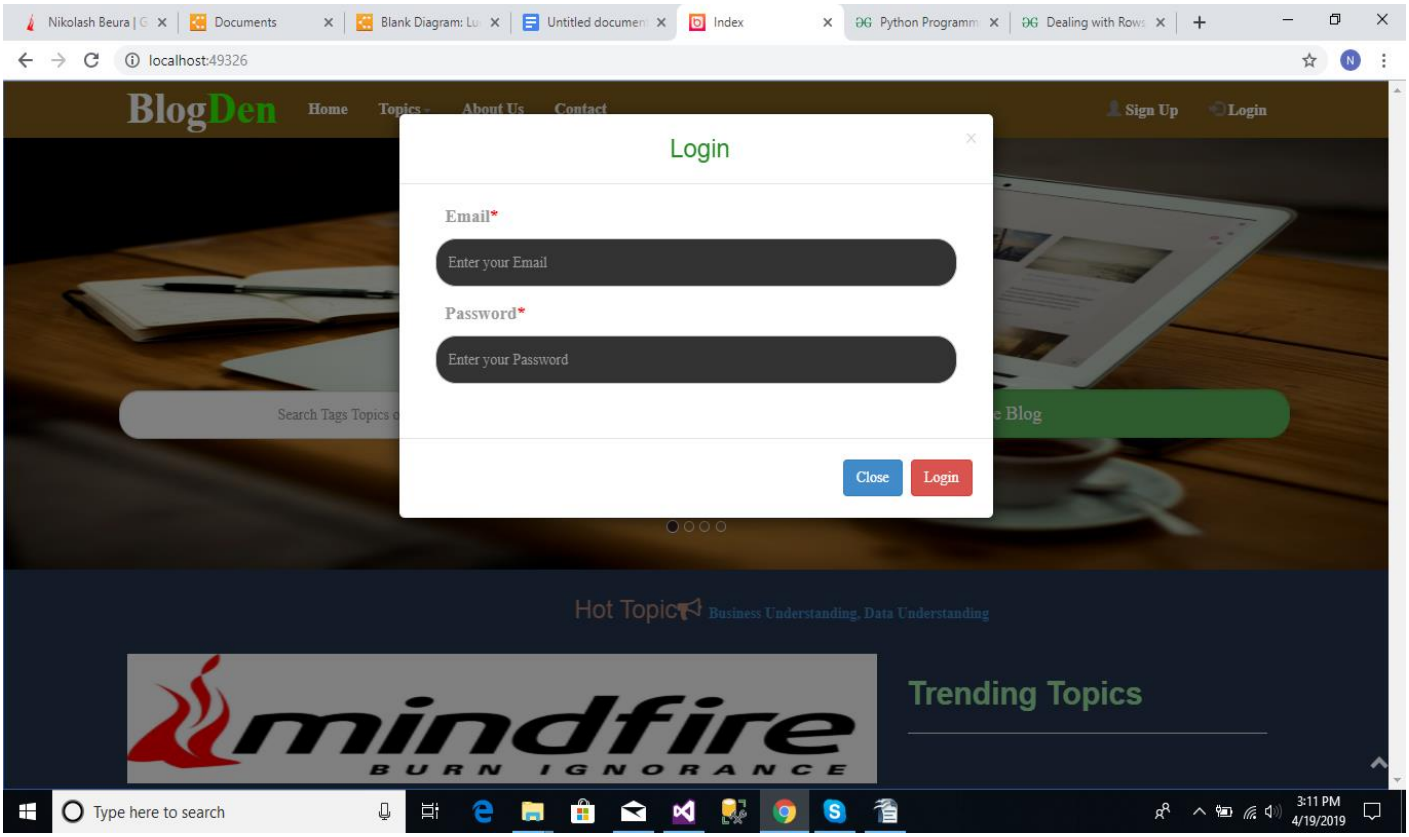
```
        [FileFormat]
        [FileSize(512000)]
        public HttpPostedFileBase ImageUpload { get; set; }


        [MaxLength(15, ErrorMessage = "*Maxlength should be 15")]
        [RegularExpression("^[0-9]+$", ErrorMessage = "Not a valid phone number")]
        public string Mobile { get; set; }


    }
}
```

5.4 Screenshots:

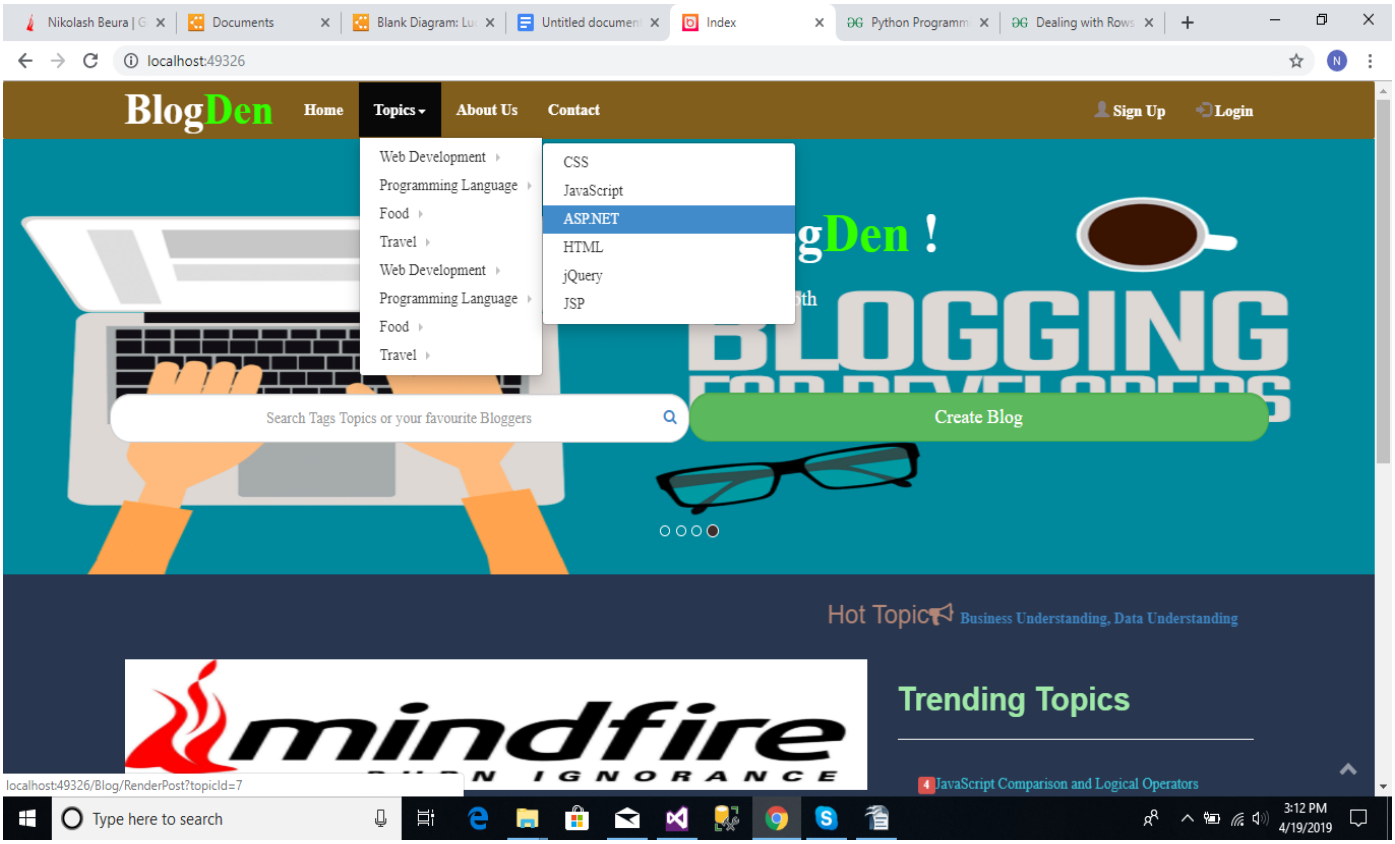
5.4.1 Registration



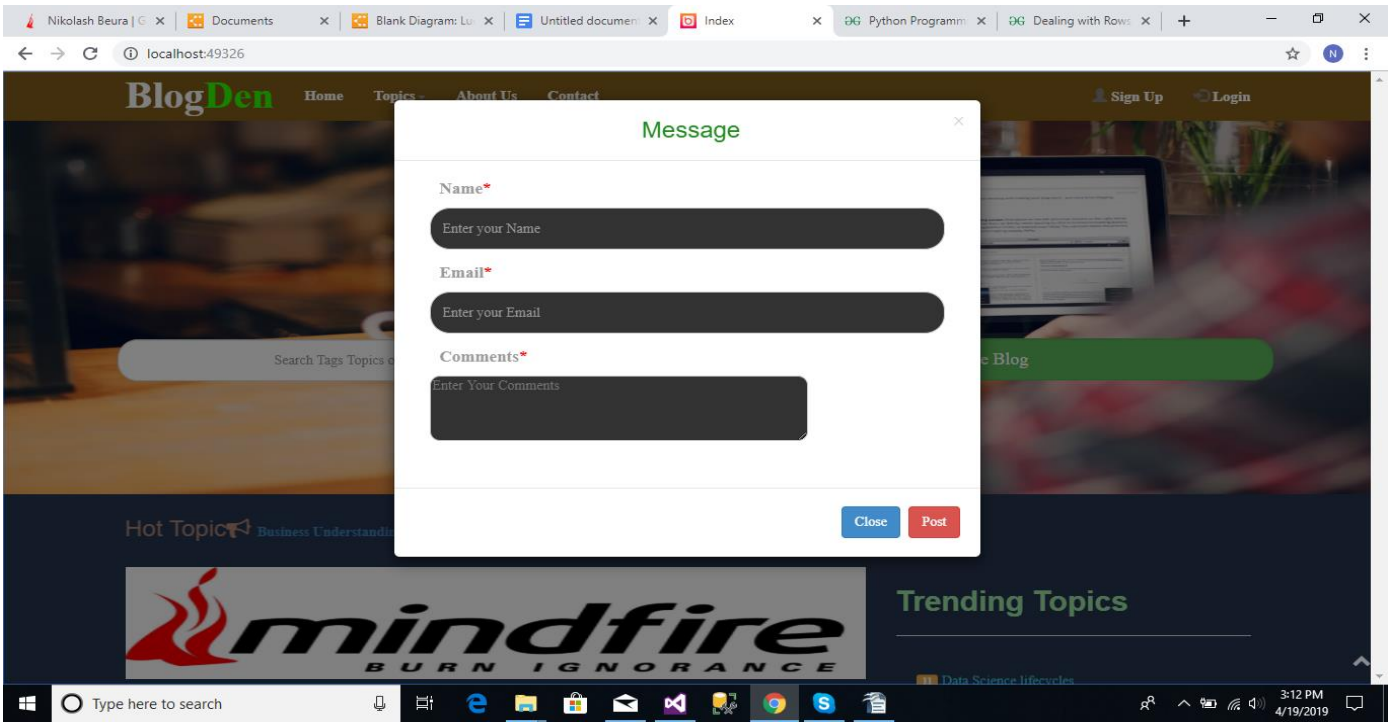
5.4.2 Login Popup



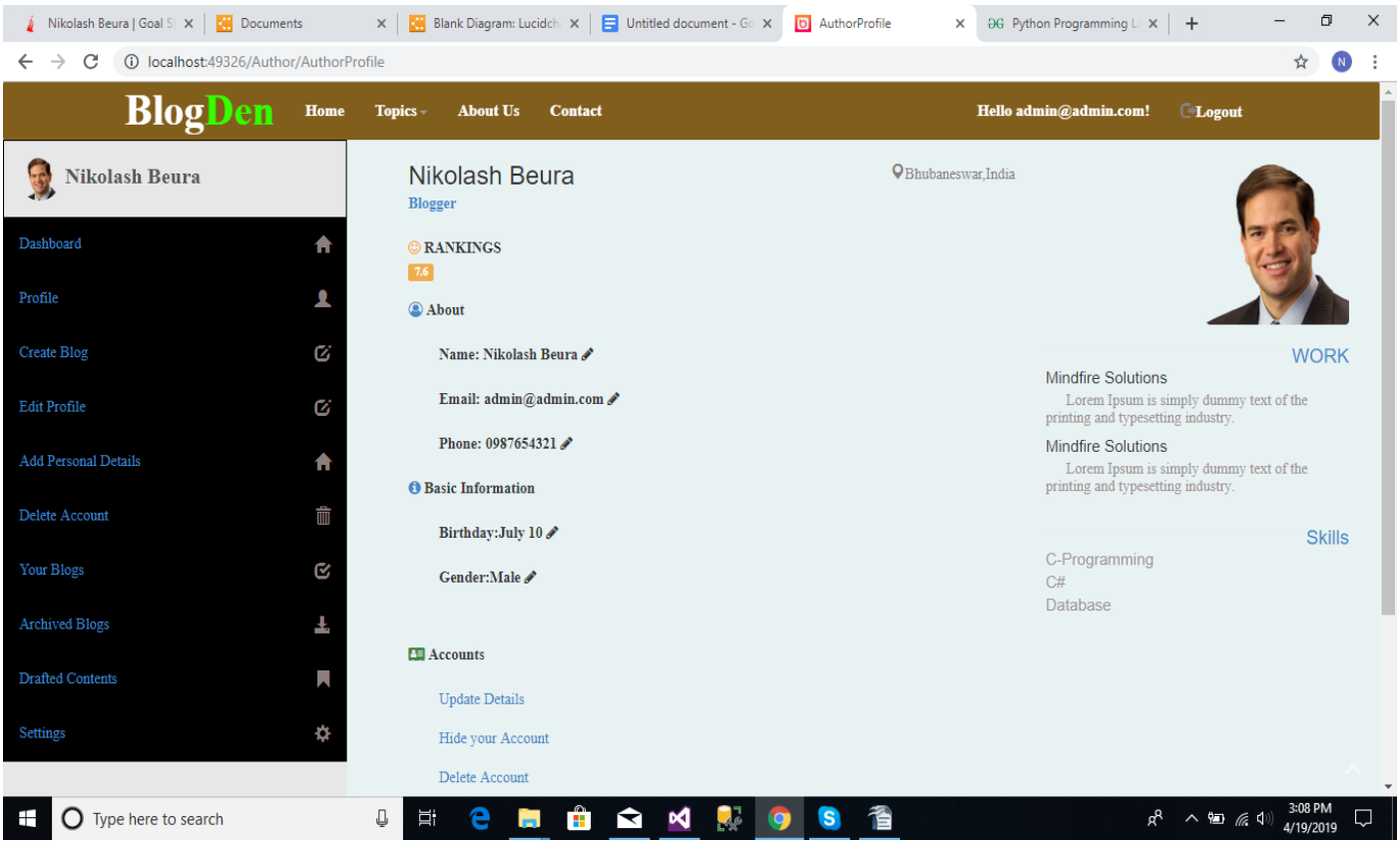
5.4.3 Index Page



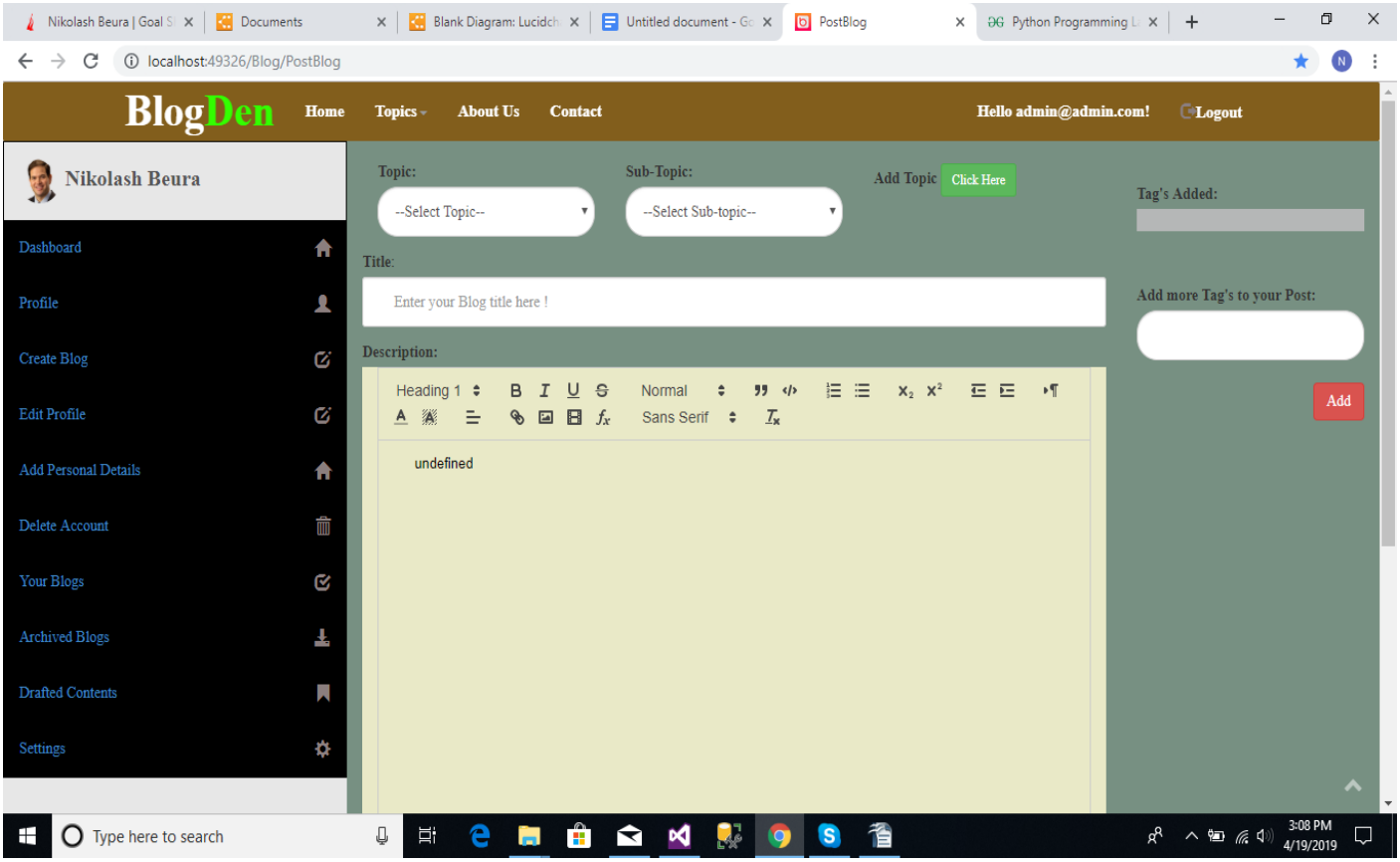
5.4.4 Contact



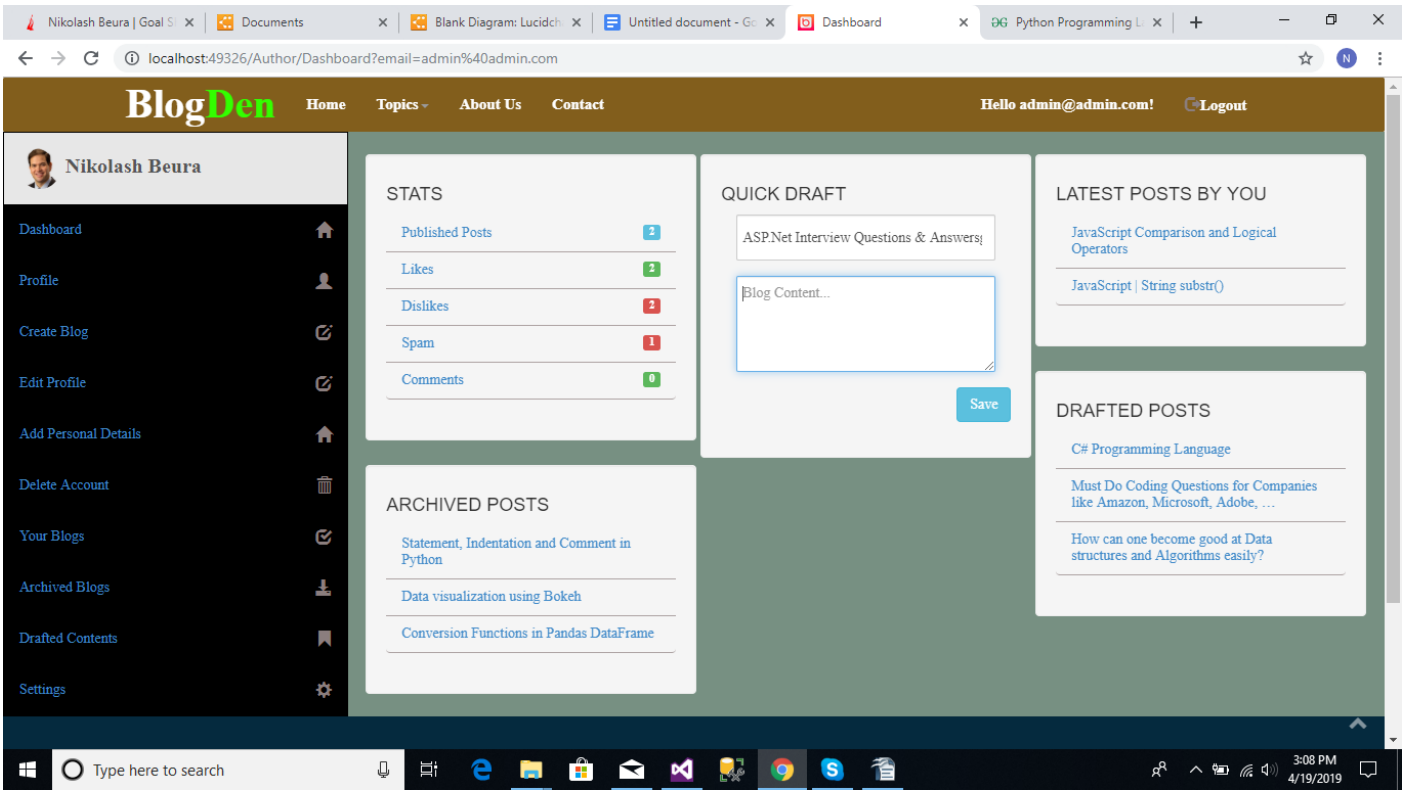
5.4.5 Profile



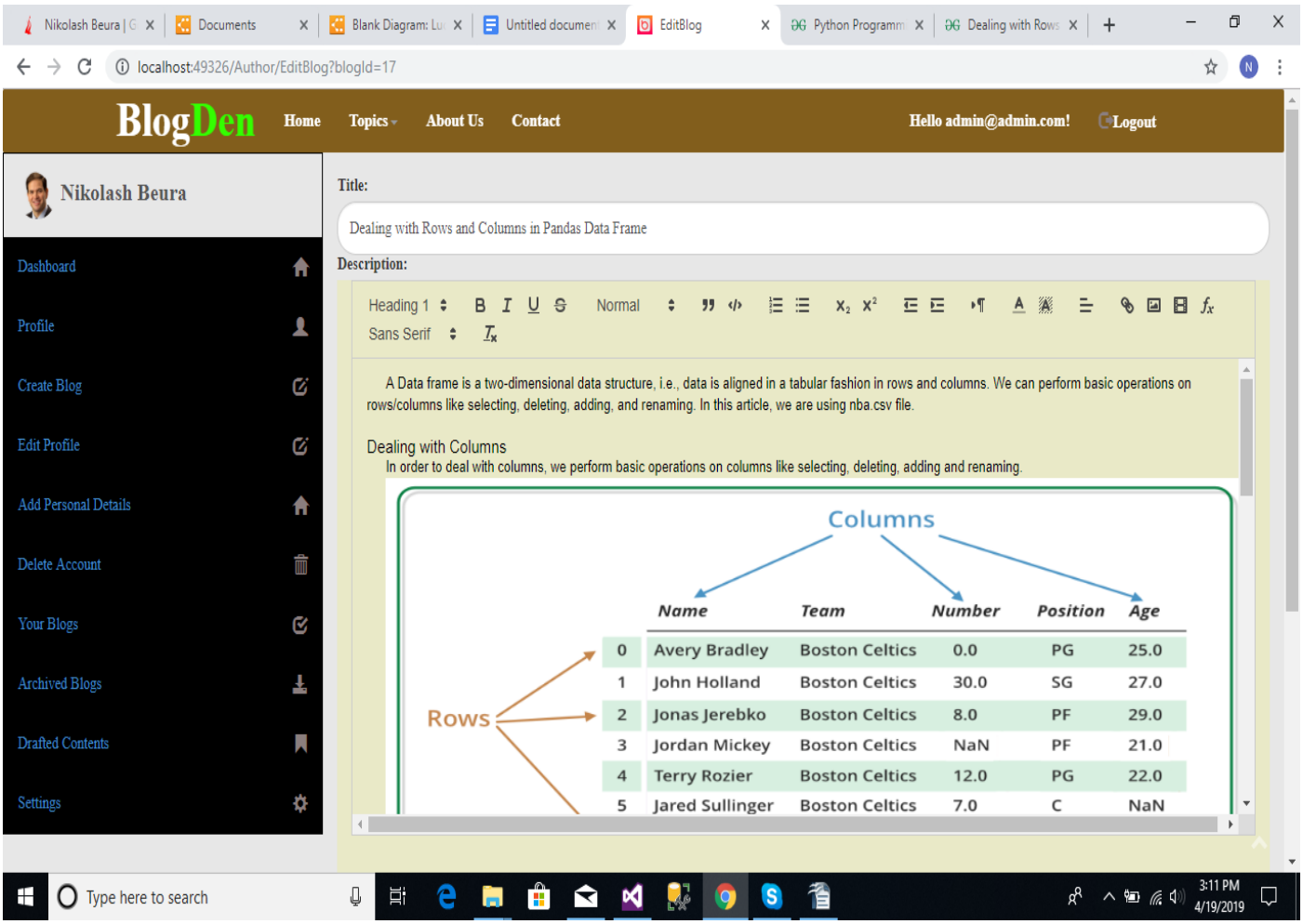
5.4.6 Create Blog



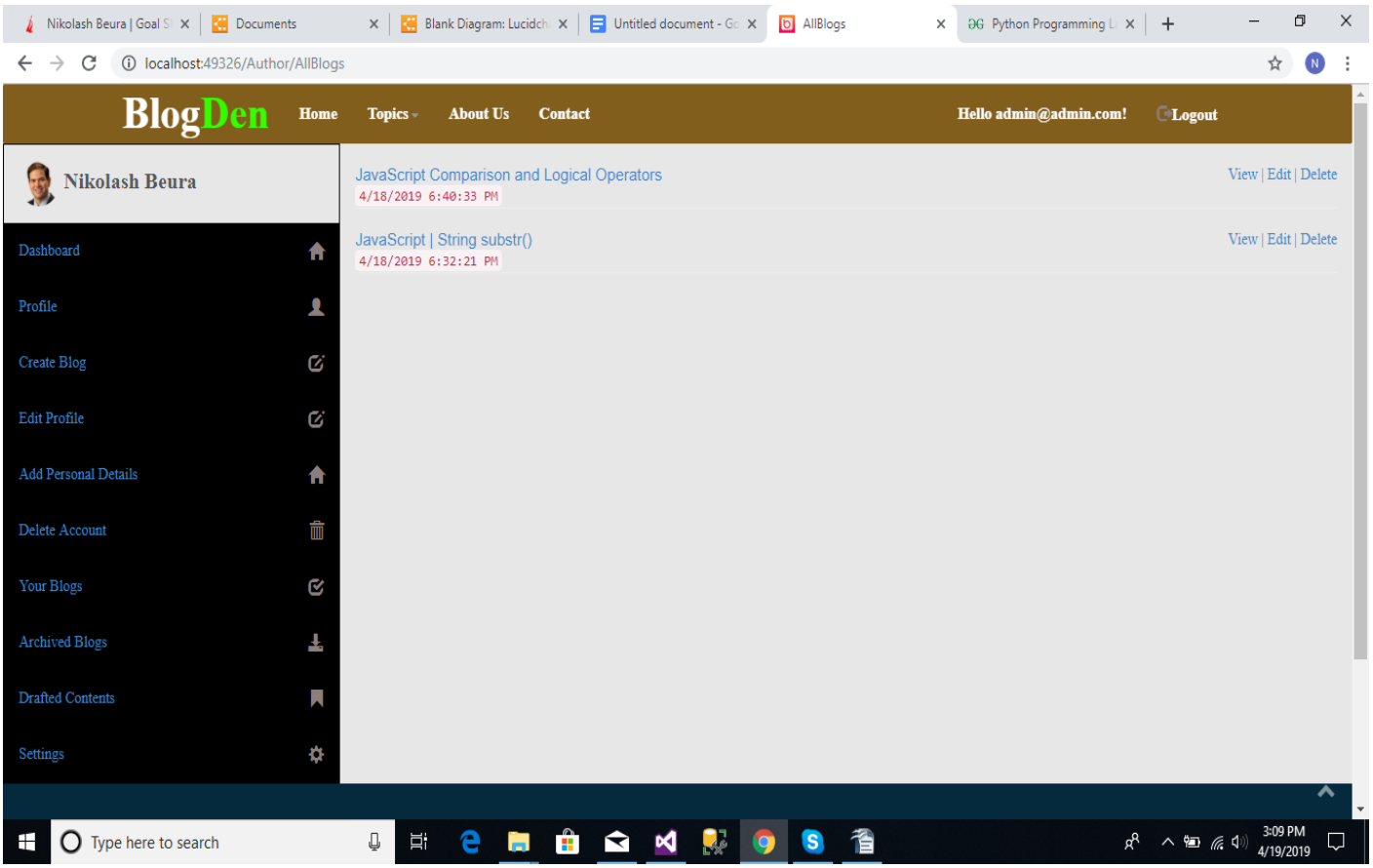
5.5.7 Dashboard



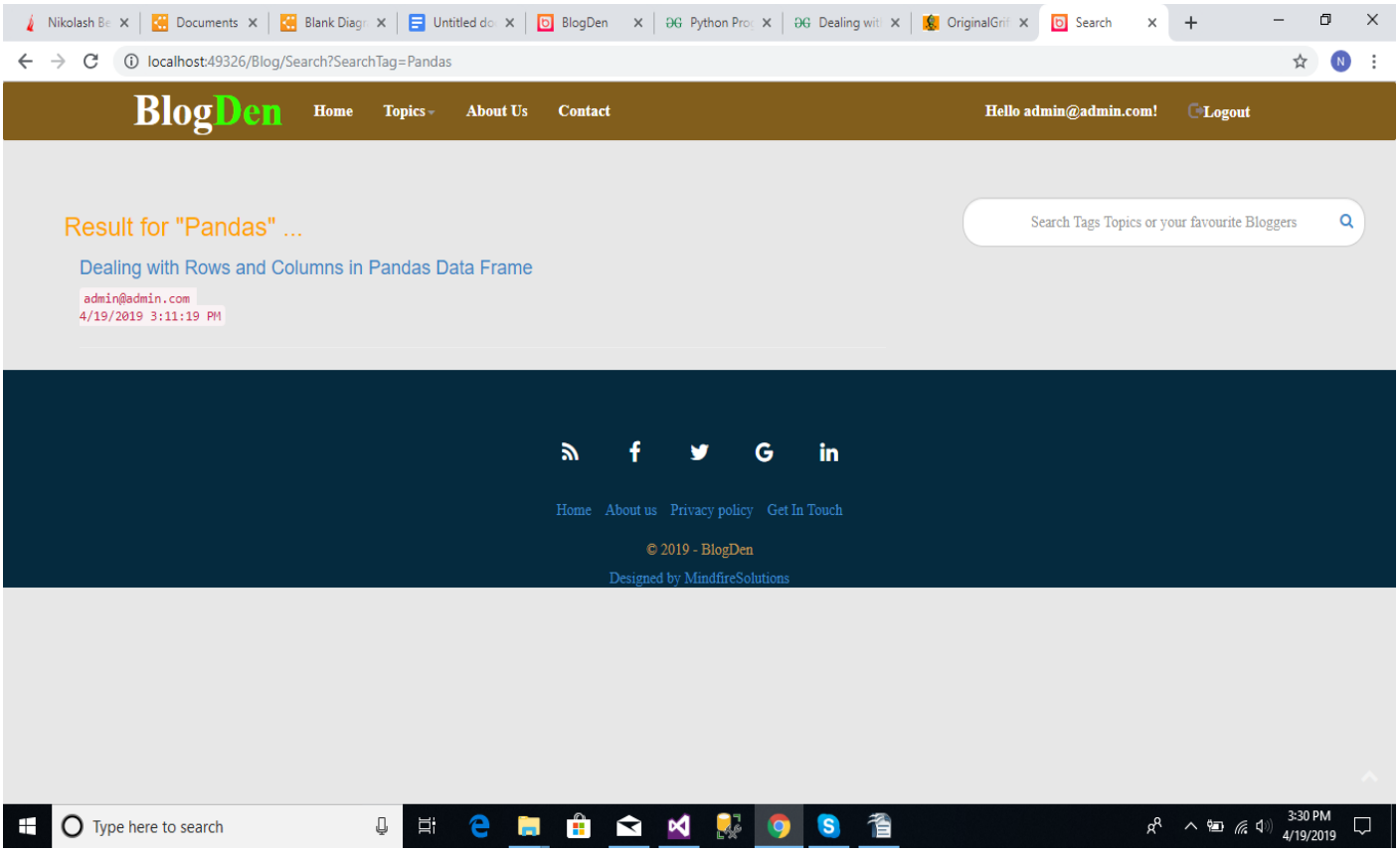
5.4.8 Edit



5.4.9 All Blogs



5.4.10 Search Result



6. SYSTEM SECURITY

6.1. Introduction

The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural

Disaster is known as System Security.

SYSTEM SECURITY refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

DATA SECURITY is the protection of data from loss, disclosure, modification and destruction.

SYSTEM INTEGRITY refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

PRIVACY defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

CONFIDENTIALITY is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

6.2 Security in Software

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls:

CLIENT SIDE VALIDATION

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

VBScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.

Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.

Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

SERVER SIDE VALIDATION

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.

User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.

Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User- name, passwords and permissions are controlled o the server side.

Using server side validation, constraints on several restricted operations are imposed.

7. CONCLUSION

This project developed, incorporated all the activities involved in the Publishing of Blogs.

It provides all necessary information to the management as well as the customer with the use of this system; the user can simply sit in front of the system and monitor all the information around the world without any physical movement

The system provides quickly and valuable information. These modules have been integrated for effective use of the management for future forecasting and for the current need.

SCOPE FOR FURTHER DEVELOPMENT

Search Engine Optimization for better search result

Machine Learning Implementation for better User interactive search

Android application can be produced for this webpage so the website can be accessed remotely without visiting the website. Which will provide nitrification when latest blogs are posted on the system

8. BIBLIOGRAPHY

Reference Websites

www.getbootstrap.com

www.w3schools.com

http://www.tutorialspoint.com/javascript/index.htm

LINQ by tutorial teacher

C# by tutorial teacher

MVC by tutorial point

Microsoft Documentation

jQuery Documentation

git Documentation

Reference Books

The Black Book

Database by Jeffrey D. Ullman

Master in HTML & CSS

Cyber Security by Zach Wabber