VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JNANA SANGAMA", BELAGAVI - 590 018

MINI PROJECT REPORT

on

BLOG MANAGEMENT SYSTEM

Submitted by

ANJALI P NAMBIAR	4SF21IS009
VARUN A L	4SF21IS121

In partial fulfillment of the requirements for the V semester

DBMS LABORATORY WITH MINI PROJECT

of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

Under the Guidance of

Mrs. Suketha

Assistant Professor, Department of CSE

at

SAHYADRI

College of Engineering & Management
An Autonomous Institution
MANGALURU
2023 - 24

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575 007

Department of Information Science & Engineering

CERTIFICATE

This is to certify that the **Mini Project** entitled **"Blog Management System"** has been carried out by **Anjali P Nambiar (4SF21IS009)** and **Varun A L (4SF21IS121)**, the bonafide students of Sahyadri College of Engineering Management in partial fulfillment of the requirements for the V semester **DBMS Laboratory with Mini Project (21CSL55)** of **Bachelor of Engineering** in **Information Science & Engineering** of Visvesvaraya Technological University, Belagavi during the year 2023 - 24. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work.

Mrs. Suketha	**Dr. Rithesh Pakkala P**
Assistant Professor	HOD & Associate Professor
Dept. of CSE	Dept. of ISE

External Practical Examination:

Examiner's Name   Signature with Date

1. ....

2. ....

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575 007

Department of Information Science & Engineering

DECLARATION

We hereby declare that the entire work embodied in this Mini Project Report titled **“Blog Management System”** has been carried out by us at Sahyadri College of Engineering and Management, Mangaluru under the supervision of **Mrs. Suketha** as the part of the V semester **DBMS Laboratory with Mini Project (21CSL55)** of **Bachelor of Engineering** in **Information Science & Engineering**. This report has not been submitted to this or any other University. .

Anjali P Nambiar (4SF21IS009)

Varun A L (4SF21IS121)

SCEM, Mangaluru

Abstract

In contemporary times, digital platforms play a pivotal role in facilitating knowledge dissemination, with blogs emerging as indispensable tools for sharing expertise and insights. A blog management system serves as a dynamic platform where users can explore a wide array of articles categorized by topic, fostering an environment conducive to collaborative learning and idea exchange. Users actively engage with content through interactive features such as commenting and sharing, enhancing the community aspect of the platform and encouraging meaningful discussions. Furthermore, the system empowers users to contribute their own articles, promoting inclusivity and enriching the content ecosystem with diverse perspectives. To maintain the quality and relevance of content, administrators play a vital role in moderating the platform, ensuring adherence to community guidelines and standards. Through effective moderation and innovative features, blog management systems facilitate the exchange of ideas and nurture vibrant communities of learners and contributors in the digital landscape.

Acknowledgement

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on **"Blog Management System"**. We have completed it as a part of the V semester **DBMS Laboratory with Mini Project (21CSL55)** of **Bachelor of Engineering** in **Information Science & Engineering** of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guide, **Mrs. Suketha**, Assistant Professor, Department of Computer Science & Engineering for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Rithesh Pakkala P** , Head & Associate Professor, Department of Information Science & Engineering and CSE (Data Science) for his invaluable support and guidance

We sincerely thank **Dr. S. S. Injaganeri**, Principal, Sahyadri College of Engineering Management and **Dr. D. L. Prabhakara**, Director, Sahyadri Educational Institutions,who have always been a great source of inspiration.

Finally, yet importantly, We express our heartfelt thanks to our family friends for their wishes and encouragement throughout the work.

–

Anjali P Nambiar
4SF21IS009
V Sem, B.E., ISE
SCEM, Mangaluru

Varun A L
4SF21IS121
V Sem, B.E., ISE
SCEM, Mangaluru

Table of Contents

Abstract **i**

Acknowledgement **ii**

Table of Contents **iv**

List of Figures **v**

1 Introduction **1**

1.1 Database 1

1.2 Database Management System(DBMS) 1

1.3 Characteristics of the Database Approach 2

1.4 Advantages of using the DBMS Approach 2

1.5 Schemas 3

1.6 DBMS Languages 3

1.7 Motivational Challenges for the Project Work 3

1.8 Application of the Project 4

2 Conceptual Data Modeling **5**

2.1 Entity Relationship (ER) Model 5

2.2 Entities 5

2.3 Attributes 6

2.4 Relationships 7

2.5 Notation for ER Diagrams 7

3 Relational Data Model **9**

3.1 Relational Model Concepts 9

3.2 Relational Model Constraints 9

4 Structured Query Language (SQL) Programming **10**

4.1 SQL Data Definition and Data Types 10

4.2 Assertions 10
4.3 Triggers 11
4.4 Database Programming 11
4.5 Database Connection 12
4.6 Stored Procedures 12

5 Requirements Specification 14
5.1 Hardware Requirements 14
5.2 Software Requirements 14

6 Relational Database Design 15
6.1 Entity Relation(ER) Diagram 15
6.2 Mapping From ER Diagram to Relational Schema Diagram 16
6.3 Relational Schema Diagram 18

7 Implementation 19
7.1 Table Structure 19
7.2 Codes Used For Modules: 22

8 Results and Discussion 25

9 Conclusion and Future work 30

References 32

List of Figures

2.1 Notation for ER Diagrams 8

6.1 ER Diagram 16

6.2 Relational Schema Diagram 18

7.1 User table 19

7.2 Author table 20

7.3 Category table 20

7.4 Comment table 21

7.5 Article table 22

8.1 Home Page 25

8.2 Login Page 26

8.3 Sign-up Page 26

8.4 Article Section 27

8.5 All the articles posted by different authors. 27

8.6 Different Categories of articles 28

8.7 Deatails of the authors 28

8.8 Article posted by an author 29

8.9 Comments on an article 29

Chapter 1

Introduction

1.1 Database

A database is a collection of organized data that allows for efficient storage and retrieval of information. It can be used to store large amounts of data in a structured and organized manner. This makes it an essential tool for businesses and organizations of all sizes. A database has some source from which data is derived, some degree of interaction with events in the real world, and an audience that is actively interested in its contents. The end users of a database may perform business transactions or events may happen that cause the information in the database to change.

1.2 Database Management System(DBMS)

A database management system (DBMS) is a software system that is designed to manage and organize data in a database. It allows for the creation, maintenance, and manipulation of data within a database. This database management system is designed to handle all aspects of the database, from storing and retrieving data, to managing and organizing it, to ensuring data integrity and security.The DBMS is responsible for controlling the data access, data integrity, and data security. It provides a user-friendly interface for creating and modifying the database structure and for inserting, updating, and retrieving data.

1.3 Characteristics of the Database Approach

In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application. The main characteristics of the database approach versus the file-processing approach are the following

- Self-describing nature of a database system.
- Insulation between programs and data, and data abstraction.
- Support of multiple views of the data.
- Sharing of data and multiuser transaction processing.

1.4 Advantages of using the DBMS Approach

1. Controlling Redundancy
2. Restricting Unauthorized Access
3. Providing Persistent Storage for Program Objects
4. Providing Storage Structures and Search Techniques for Efficient Query Processing.
5. Providing Backup and Recovery
6. Providing Multiple User Interfaces.
7. Representing Complex Relationships among Data.
8. Enforcing Integrity Constraints
9. Permitting Inferencing and Actions Using Rules and Triggers.
10. Additional Implications of Using the Database Approach or Potential for Enforcing Standards.
    - Reduced Application Development Time.
    - Flexibility o Availability of Up-to-Date Information.
    - Economies of Scale.

1.5 Schemas

In a data model, it is important to distinguish between the description of the database and the database itself. The description of a database is called the database schema, which is specified during database design and is not expected to change frequently. Most data models have certain conventions for displaying schemas as diagrams. A displayed schema is called a **schema diagram**. A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints.

1.6 DBMS Languages

In many DBMSs where no strict separation of levels is maintained, one language, called the data definition language (DDL), is used by the DBA and by database designers to define both schemas. Another language, the storage definition language (SDL), is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.For a true three-schema architecture, we would need a third language, the view definition language (VDL), to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries.Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the data manipulation language (DML) for these purposes.

1.7 Motivational Challenges for the Project Work

Motivational challenges for the project work for developing Blog Management System can arise from from several factors.

Content Creation Burnout: Creating high-quality content consistently can be challenging, leading to burnout among authors and contributors. Encourage creativity and provide support and recognition for authors to prevent burnout and maintain motivation.

User Engagement and Interaction: Encouraging user engagement and interaction on the platform can be challenging, especially if users feel disconnected or disinterested. Implement features such as comments, likes, shares, and personalized recommendations to enhance user engagement and keep users motivated to participate.

Content Relevance and Freshness: Ensuring that the content remains relevant and fresh over time can be a constant challenge. Regularly update content, curate trending topics, and encourage user-generated content to keep the platform dynamic and engaging.
Competition and Market Saturation: The blogosphere is highly competitive, with numerous blogs covering similar topics. Standing out from the crowd and attracting a loyal audience can be challenging. Focus on niche topics, unique perspectives, and differentiated content to differentiate your blog and attract and retain readers.
Technical Challenges: Managing the technical aspects of the blog management system, such as website maintenance, security, and performance optimization, can be daunting. Invest in robust hosting infrastructure, regularly update software and plugins, and implement security measures to ensure smooth operation and user satisfaction.

1.8 Application of the Project

This project represents a dynamic hub for knowledge exchange and community engagement across diverse subjects. Users can delve into a plethora of articles, sparking lively discussions and fostering a robust sense of community through interactive comments and exchanges. Authors can seize the opportunity to showcase their expertise, positioning themselves as influential voices within their respective domains. With content thoughtfully organized for effortless exploration, users will remain updated on the latest trends, ensuring a continuous flow of fresh insights and perspectives. In essence, the blog management system emerges as an indispensable catalyst for intellectual growth, vibrant interaction, and influential thought leadership.

Chapter 2

Conceptual Data Modeling

2.1 Entity Relationship (ER) Model

The Entity Relationship (ER) model is a data modeling technique used to represent the entities and relationships between them in a database.This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts. This model is used to describe the data elements and their relationships in a clear and concise way. The ER model is made up of entities, which are objects or concepts that are relevant to the system, and relationships, which are the connections between the entities. The ER model is useful for identifying the relationships between the data elements and for creating a logical design for a database.

2.2 Entities

The basic concept that the ER model represents is an entity, which is a thing or object in the real world with an independent existence. An entity may be an object with a physical existence. An entity may be an object with a physical existence or it may be an object with a conceptual existence. Each entity has attributes—the particular properties that describe it. A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.

2.3 Attributes

Several types of attributes occur in the ER model: simple versus composite, singlevalued versus multivalued, and stored versus derived

1. **Composite versus Simple (Atomic) Attributes** : Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. Attributes that are not divisible are called simple or atomic attributes. Composite attributes can form a hierarchy; The value of a composite attribute is the concatenation of the values of its component simple attributes.

2. **Single-Valued versus Multivalued Attributes** : Most attributes have a single value for a particular entity, such attributes are called single-valued.In some cases an attribute can have a set of values for the same entity. A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.

3. **Stored versus Derived Attributes** : Stored attributes and derived attributes are two types of attributes in a database. Stored attributes are attributes that are directly stored in the database and are permanent parts of the record. On the other hand, derived attributes are attributes that are calculated or inferred from other attributes. These attributes can be determined by performing calculations or processing on other attributes.

4. **NULL Values** : NULL values in a database refer to the absence of a value for a specific attribute. In other words, a NULL value indicates that the attribute does not have a known or applicable value for a particular record. NULL values are different from empty or zero values, as they indicate a lack of data rather than a specific value.In a relational database, NULL values are used to represent missing or unknown information.

5. **Complex Attributes** : Complex attributes are attributes that are composed of multiple sub-attributes or components. These attributes are more complex than simple attributes, which are composed of a single data value. Complex attributes can be used to represent more detailed information about an entity in a database.

2.4 Relationships

A relationship type R among n entity types El, E2, ... , En defines a set of associations or a relationship set among entities from these entity types. Similar to the case of entity types and entity sets, a relationship type and its corresponding relationship set are customarily referred to by the same name, R. Mathematically, the relationship set R is a set of relationship instances r1, where each ri associates n individual entities (el, e2, ... ,en), and each entity ej in ri is a member of entity set Ej, 1 jS n. Hence,a relationship set is a mathematical relation on El, E2, . . . , En, alternatively it can be defined as a subset of the Cartesian product of the entity sets El × E2 × . . . SEn.Each of the entity types El, E2, . . ., En is said to participate in the relationship type R. Similarly, each of the individual entities el, e2, . . ., en is said to participate in the relationship instance ri = (el, e2, . . , en).

2.5 Notation for ER Diagrams

It is occasionally difficult to decide whether a particular concept in the miniworld should be modeled as an entity type, an attribute, or a relationship type.In general, the schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.

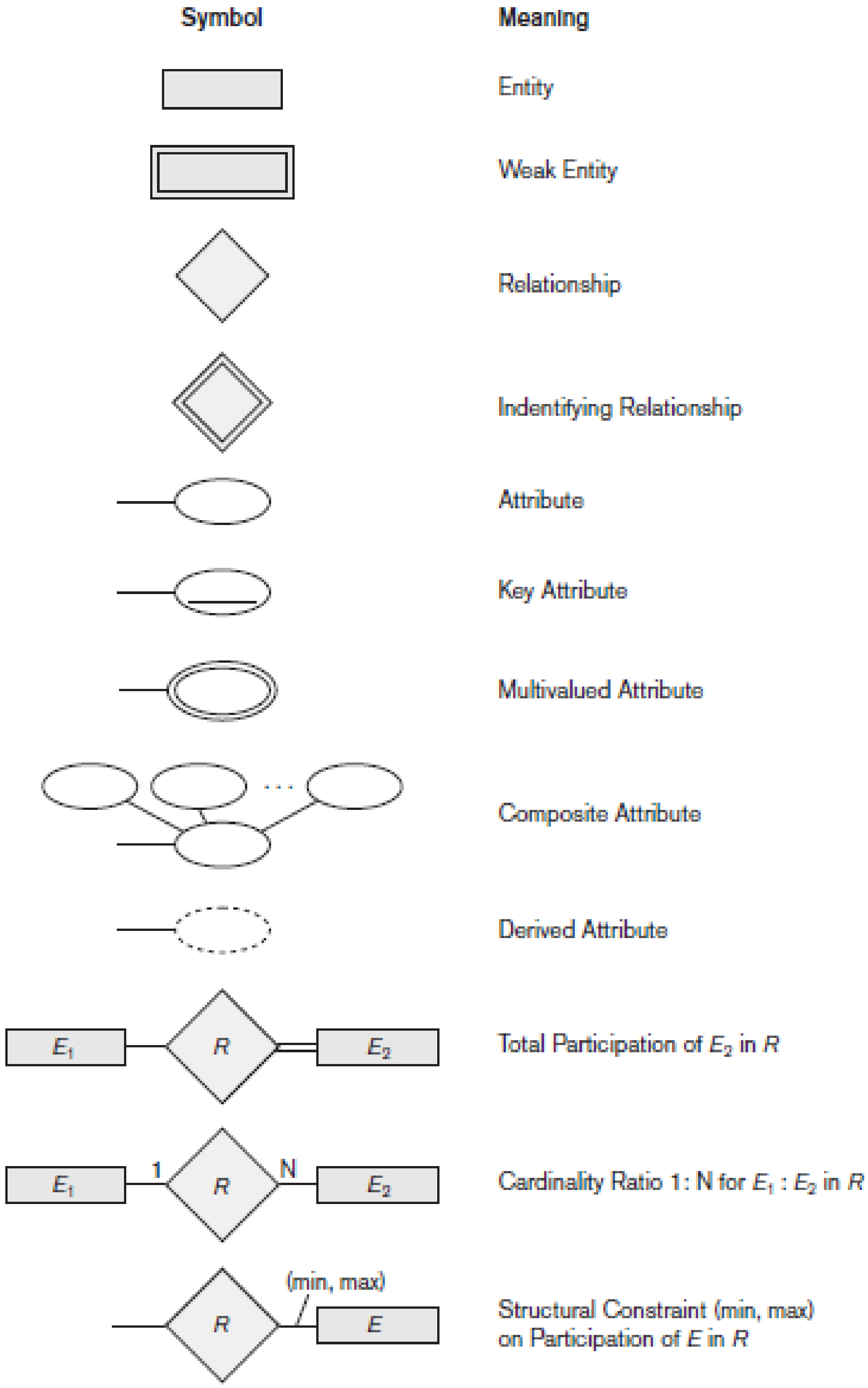

Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
. . .
Composite Attribute
Derived Attribute
E_1
R
E_2
Total Participation of E_2 in R
E_1
1
R
N
E_2
Cardinality Ratio 1: N for E_1 : E_2 in R
R
(min, max)
E
Structural Constraint (min, max) on Participation of E in R

Figure 2.1: Notation for ER Diagrams

Chapter 3

Relational Data Model

3.1 Relational Model Concepts

The relational model represents the database as a collection of relations. Informally, each relation resembles a table of values or, to some extent, a **flat file** of records. It is called a flat file because each record has a simple linear or flat structure. When a relation is thought of as a table of values, each row in the table represents a collection of related data values. A row represents a fact that typically corresponds to a real-world entity or relationship. The table name and column names are used to help to interpret the meaning of the values in each row.

3.2 Relational Model Constraints

Constraints on databases can generally be divided into three main categories:

- Constraints that are inherent in the data model. These inherent model-based constraints or implicit constraints.

- Constraints that can be directly expressed in the schemas of the data model, typically by specifying them in the DDL. These are schema-based constraints or explicit constraints.

- Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs or in some other way. We call these application-based or semantic constraints or business rules.

Chapter 4

Structured Query Language (SQL) Programming

4.1 SQL Data Definition and Data Types

SQL uses the terms **table, row, and column** for the formal relational model terms relation, tuple, and attribute, respectively. We will use the corresponding terms interchangeably. The main SQL command for data definition is the **CREATE** statement, which can be used to create schemas, tables (relations), types, and domains, as well as other constructs such as views, assertions, and triggers.

4.2 Assertions

In SQL, users can specify general constraints via **declarative assertions**, using the **CREATE ASSERTION** statement. Each assertion is given a constraint name and is specified via a condition similar to the WHERE clause of an SQL query. **CHECK** clause and constraint condition can also be used to specify constraints on individual attributes and domains and on individual tuples. A major difference between **CREATE ASSERTION** and the individual domain constraints and tuple constraints is that the **CHECK** clauses on individual attributes, domains, and tuples are checked in SQL only when tuples are inserted or updated in a specific table. the schema designer should use **CREATE ASSERTION** only in cases where it is not possible to use **CHECK** on attributes, domains, or tuples, so that simple checks are implemented more efficiently by the DBMS.

4.3 Triggers

The **CREATE TRIGGER** statement is used to implement such actions in SQL. A typical trigger which is regarded as an ECA (Event, Condition, Action) rule has three components:

- The **event**(s): These are usually database update operations that are explicitly applied to the database.The person who writes the trigger must make sure that all possible events are accounted for. In some cases, it may be necessary to write more than one trigger to cover all possible cases.These events are specified after the keyword **BEFORE** which means that the trigger should be executed before the triggering operation is executed. An alternative is to use the keyword **AFTER**, which specifies that the trigger should be executed after the operation specified in the event is completed.

- The **condition** that determines whether the rule action should be executed: Once the triggering event has occurred, an optional condition may be evaluated.

- The **action** to be taken: The action is usually a sequence of SQL statements, but it could also be a database transaction or an external program that will be automatically executed

4.4 Database Programming

Several techniques exist for including database interactions in application programs. The main approaches for database programming are the following:

1. **Embedding database commands in a general purpose programming language**. In this approach, database statements are **embedded** into the host programming language, but they are identified by a special prefix. For example, the prefix for embedded SQL is the string EXEC SQL, which precedes all SQL commands in a host language program.

2. **Using a library of database functions or classes.A library of functions** is made available to the host programming language for database calls. The actual database query and update commands and any other necessary information are included as parameters in the function calls.

3. **Designing a brand-new language**. A database programming language is designed from scratch to be compatible with the database model and query language. Additional programming structures such as loops and conditional statements are added to the database language to convert it into a full-fledged programming language.

4.5 Database Connection

A database connection refers to the process of connecting to a database management system (DBMS) from a client application, such as a web application or a desktop application. This process involves providing the necessary credentials, such as the username and password, to the DBMS and establishing a communication link between the client and the DBMS.

A database connection refers to the process of connecting to a database management system (DBMS) from a client application, such as a web application or a desktop application. This process involves providing the necessary credentials, such as the username and password, to the DBMS and establishing a communication link between the client and the DBMS.

4.6 Stored Procedures

It is sometimes useful to create database program modules—procedures or functions—that are stored and executed by the DBMS at the database server. These are historically known as database stored procedures, although they can be functions or procedures. The term used in the SQL standard for stored procedures is persistent stored modules because these programs are stored persistently by the DBMS, similarly to the persistent data stored by the DBMS.

Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.

- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.

- These procedures can enhance the modeling power provided by views by allowing more complex types of derived data to be made available to the database users via the stored procedures.

Chapter 5

Requirements Specification

5.1 Hardware Requirements

- Processor : Intel i3 and Above or AMD Ryzen 5 and Above
- RAM : 4GB
- Hard Disk : 500GB
- Input Device : Standard Keyboard and Mouse
- Output Device : Monitor

5.2 Software Requirements

- Database : SQL
- Programming Language : PHP
- IDE : Visual Studio Code
- Operating System : Windows 11

Chapter 6

Relational Database Design

The procedure involves designing an ER or EER conceptual schema and then mapping it to the relational model by a procedure.Here we focus on the logical database design step of database design, which is also known as data model mapping. We present the procedures to create a relational schema from an entity–relationship (ER) or an enhanced ER (EER) schema

6.1 Entity Relation(ER) Diagram

An Entity-Relationship (ER) diagram is a visual representation of the entities and relationships that exist within a database. It is used to model the data and the relationships between different data elements in a clear and concise manner. ER diagrams are widely used in the database design process and they provide a means to understand and communicate the data requirements and the overall architecture of the system. They can be used to validate the data model, identify any inconsistencies or errors, and to make sure that the data model is consistent with the system requirements.

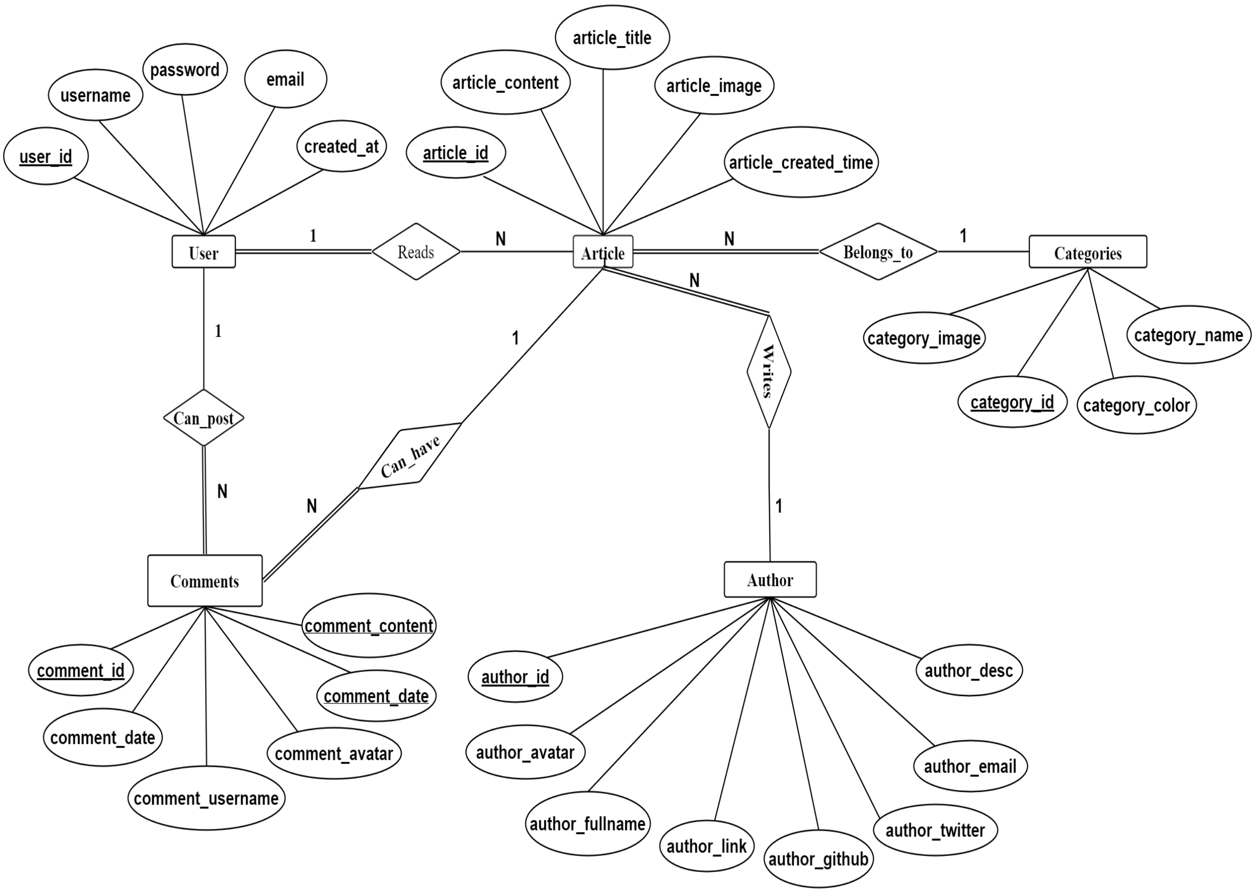


Figure 6.1: ER Diagram

6.2 Mapping From ER Diagram to Relational Schema Diagram

1. **Mapping of Regular Entities**:This step involves mapping all the regular entity types to tabular format by identifying their primary keys.

2. **Mapping of Weak Entity**:When mapping weak entity types along with other attributes the partial key and primary key of parent entity together will form their primary key of the new relation.

3. **Mapping of 1:1 Relation**:In this step foreign keys are assigned using foreign key approach.The primary key of the participating relation R or S is added as primary key to second entity types by looking at the participating constraints.

4. **Mapping of 1:N Relation**:Foreign key approach is used to add one sided primary key to the n sided entity at foreign key.

5. **Mapping of M:N Relation**:Here we use the cross reference approach where the relationship is converted to a new relation within attributes on primary keys of

both participating relation.

6. **Mapping of Multi-valued Relation**:For multi-valued attributes a separate relation has to be created along with primary key of parent relation.

7. **Mapping of N-ary Relation**:For mapping N array relationship we create a new relation with a relationship name in its attribute and primary keys of all participating entity types.

In our database we have the following mappings:

Step 1 : Mapping of Regular Entities :

From the ER diagram we identify all the strong entities E and create a relation R that includes all it's simple attributes and primary keys.

The following are the strong entities from our schema diagram :

1.USERS(user_id,username,password,created_at)

2.AUTHOR(author_Id,author_fullname,author_desc,author_email,author_twitter, author_github,author_link,author_avatar)

3.ARTICLE(article_id,article_title,article_content,article_image,article_created_time)

4.CATEGORY(category_id,category_name,category_image,category_color)

5.COMMENT(comment_id,comment_username,comment_avatar,comment_content, comment_date)

Step 2 : Mapping of Weak Entities :

From the ER diagram we identify that there is no weak entity.

Step 3 : Mapping of binary 1:1 Relation Types :

From the ER diagram we identify that there is no weak entity.

Step 4 : Mapping of binary 1:N Relation Types :

The User and Article entities are participating in the 1:N relation type.Since Article is on the nth side of the relation, the primary key of User entity, that is user_id is added as the Foreign key in Article entity.

The Article and Category entities are participating in the 1:N relation type.Since Article is on the nth side of the relation, the primary key of Category entity, that is category_id is added as the Foreign key in Article entity.

The Article and the Author entities are participating in the 1:N relation type.Since Article is on the nth side of the relation, the primary key of Author entity, that is article_id is added as the Foreign key in Author entity.

The User and the Comments entities are participating in the 1:N relation type.Since

Comments is on the nth side of the relation, the primary key of User entity, that is user_id is added as the Foreign key in Author entity.

The Article and the Comments entities are participating in the 1:N relation type.Since Comments is on the nth side of the relation, the primary key of Article entity, that is Article_id is added as the Foreign key in Author entity.

Step 5: Mapping of binary M:N Relation Types :

From the ER diagram we identify that there are no M:N Relation.

Step 6: Mapping of Multivalued Relation Types :

From the ER diagram we identify that there are no multivalued attributes.

6.3 Relational Schema Diagram

A Schema is a pictorial representation of the relationship between the database tables in the database that is created. The database schema of a database system is its structure described in a formal language sup ported by the database management system (DBMS). The term ”schema” refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

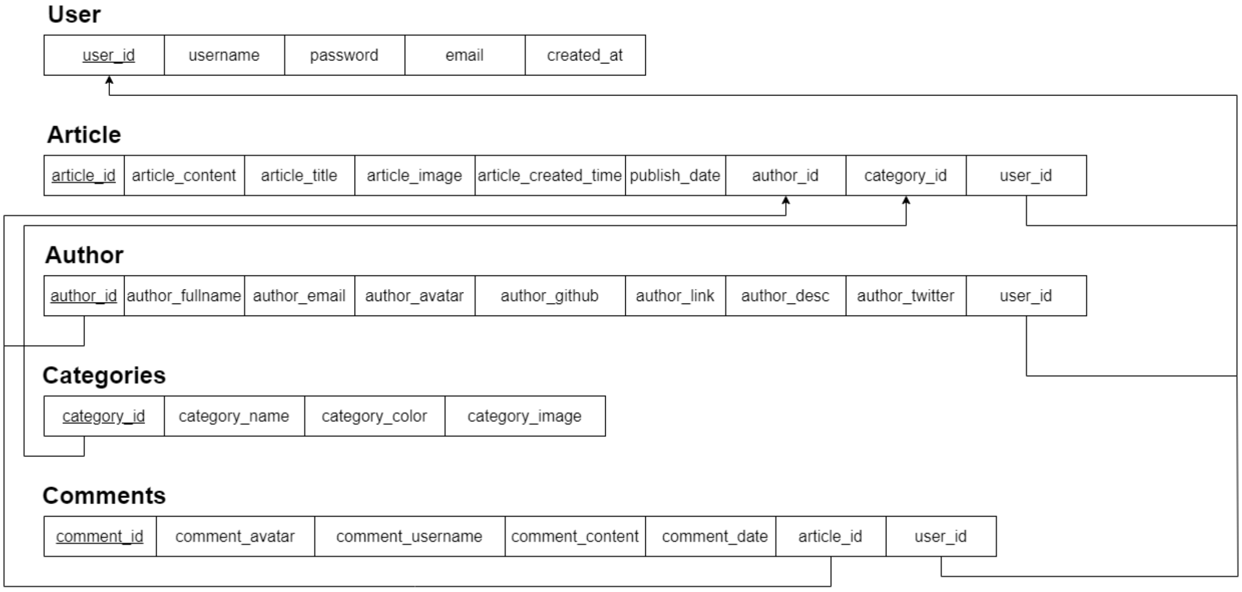

User
user_id username password email created_at
Article
article_id article_content article_title article_image article_created_time publish_date author_id category_id user_id
Author
author_id author_fullname author_email author_avatar author_github author_link author_desc author_twitter user_id
Categories
category_id category_name category_color category_image
Comments
comment_id comment_avatar comment_username comment_content comment_date article_id user_id

Figure 6.2: Relational Schema Diagram

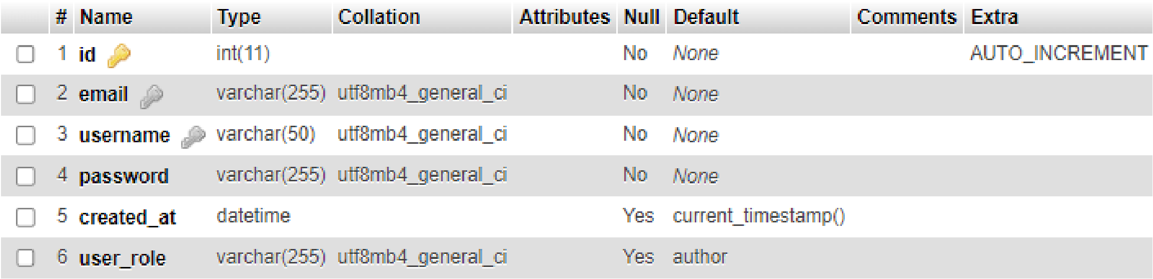
Chapter 7

Implementation

7.1 Table Structure

Users

```
CREATE TABLE users (
id int(11) NOT NULL,
email varchar(255) NOT NULL,
username varchar(50) NOT NULL,
password varchar(255) NOT NULL,
created_at datetime DEFAULT current_timestamp(),
user_role varchar(255) DEFAULT ’author’
);
```

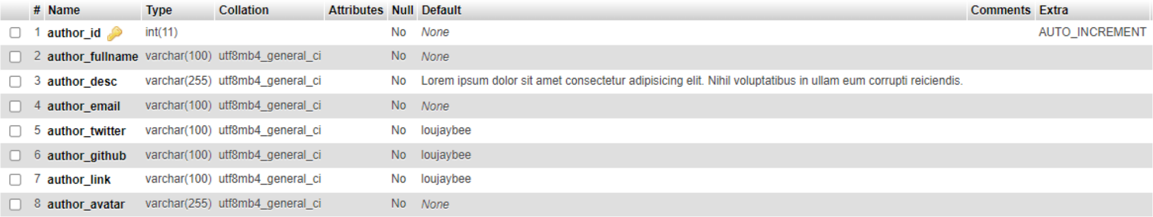


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	id 🔑	int(11)			No	*None*		AUTO_INCREMENT
□	2	email 🔑	varchar(255)	utf8mb4_general_ci		No	*None*		
□	3	username 🔑	varchar(50)	utf8mb4_general_ci		No	*None*		
□	4	password	varchar(255)	utf8mb4_general_ci		No	*None*		
□	5	created_at	datetime			Yes	current_timestamp()		
□	6	user_role	varchar(255)	utf8mb4_general_ci		Yes	author		

Figure 7.1: User table

Author

```
CREATE TABLE author (
author_id int(11) NOT NULL,
author_fullname varchar(100) NOT NULL,
author_desc varchar(255) NOT NULL ,
author_email varchar(100) NOT NULL,
author_twitter varchar(100) NOT NULL ,
author_github varchar(100) NOT NULL ,
author_link varchar(100) NOT NULL ,
author_avatar varchar(255) NOT NULL
);
```

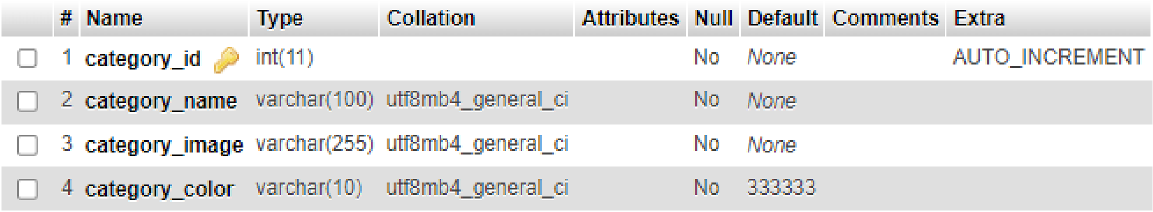


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	**author_id**	int(11)			No	*None*		AUTO_INCREMENT
□	2	**author_fullname**	varchar(100)	utf8mb4_general_ci		No	*None*		
□	3	**author_desc**	varchar(255)	utf8mb4_general_ci		No	Lorem ipsum dolor sit amet consectetur adipisicing elit. Nihil voluptatibus in ullam eum corrupti reiciendis.		
□	4	**author_email**	varchar(100)	utf8mb4_general_ci		No	*None*		
□	5	**author_twitter**	varchar(100)	utf8mb4_general_ci		No	loujaybee		
□	6	**author_github**	varchar(100)	utf8mb4_general_ci		No	loujaybee		
□	7	**author_link**	varchar(100)	utf8mb4_general_ci		No	loujaybee		
□	8	**author_avatar**	varchar(255)	utf8mb4_general_ci		No	*None*		

Figure 7.2: Author table

Category

```
CREATE TABLE category (
category_id int(11) NOT NULL,
category_name varchar(100) NOT NULL,
category_image varchar(255) NOT NULL,
category_color varchar(10) NOT NULL DEFAULT,
);
```



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	**category_id**	int(11)			No	*None*		AUTO_INCREMENT
□	2	**category_name**	varchar(100)	utf8mb4_general_ci		No	*None*		
□	3	**category_image**	varchar(255)	utf8mb4_general_ci		No	*None*		
□	4	**category_color**	varchar(10)	utf8mb4_general_ci		No	333333		

Figure 7.3: Category table

Comment

CREATE TABLE comment (

comment_id int(11) NOT NULL,

comment_username varchar(100) NOT NULL,

comment_avatar varchar(255) NOT NULL DEFAULT,

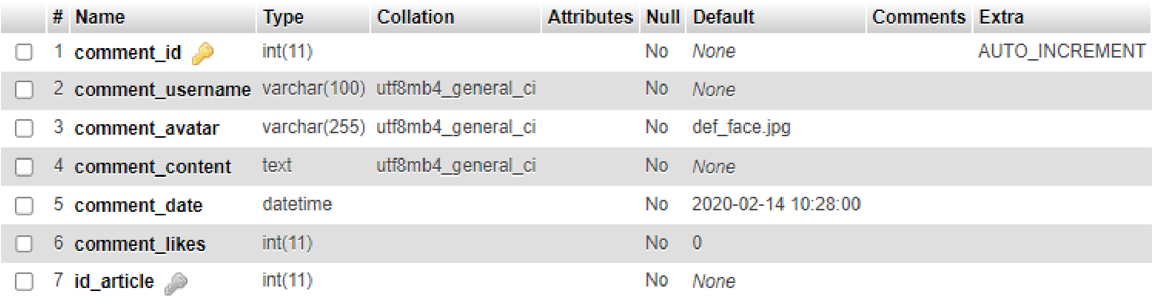
comment_content text NOT NULL,

comment_date datetime NOT NULL DEFAULT,

comment_likes int(11) NOT NULL DEFAULT 0,

id_article int(11) NOT NULL

);



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	comment_id	int(11)			No	*None*		AUTO_INCREMENT
□	2	comment_username	varchar(100)	utf8mb4_general_ci		No	*None*		
□	3	comment_avatar	varchar(255)	utf8mb4_general_ci		No	def_face.jpg		
□	4	comment_content	text	utf8mb4_general_ci		No	*None*		
□	5	comment_date	datetime			No	2020-02-14 10:28:00		
□	6	comment_likes	int(11)			No	0		
□	7	id_article	int(11)			No	*None*		

Figure 7.4: Comment table

Article

CREATE TABLE article (

article_id int(11) NOT NULL,

article_title varchar(255) NOT NULL,

article_content text NOT NULL,

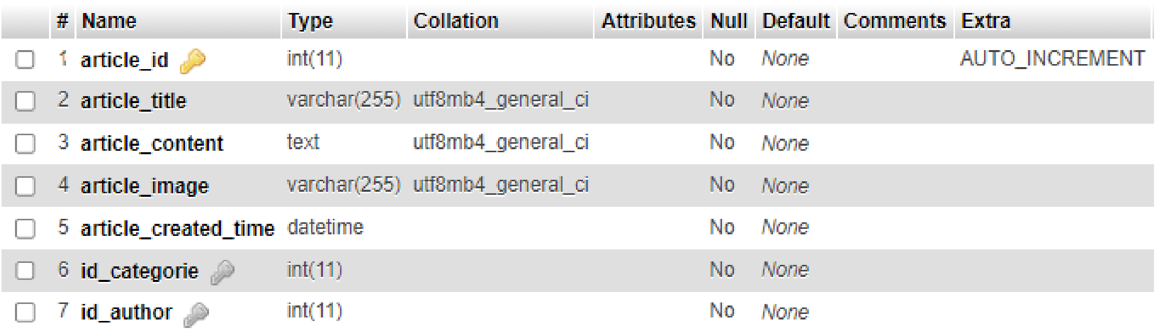
article_image varchar(255) NOT NULL,

article_created_time datetime NOT NULL,

id_categorie int(11) NOT NULL,

id_author int(11) NOT NULL

);



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	article_id 🔑	int(11)			No	*None*		AUTO_INCREMENT
□	2	article_title	varchar(255)	utf8mb4_general_ci		No	*None*		
□	3	article_content	text	utf8mb4_general_ci		No	*None*		
□	4	article_image	varchar(255)	utf8mb4_general_ci		No	*None*		
□	5	article_created_time	datetime			No	*None*		
□	6	id_categorie 🔑	int(11)			No	*None*		
□	7	id_author 🔑	int(11)			No	*None*		

Figure 7.5: Article table

7.2 Codes Used For Modules:

Trigger:

```
CREATE TRIGGER delete_article_on_user_del
AFTER DELETE ON author FOR EACH
ROW DELETE FROM article WHERE id_author = OLD.author_id;
```

Stored Procedure:

```
CREATE PROCEDURE GetAllArticles () SELECT
article.article_id,
article.article_title,
article.article_content,
article.article_image,
article.article_created_time,
category.category_name,
author.author_fullname
FROM
article
JOIN category ON article.id_categorie = category.category_id
JOIN author ON article.id_author = author.author_id
ORDER BY article.article_id;

CREATE PROCEDURE GetAuthArticles (IN authorID INT) SELECT
article.article_id,
article.article_title,
article.article_content,
article.article_image,
article.article_created_time,
category.category_name,
author.author_fullname
FROM
article
JOIN category ON article.id_categorie = category.category_id
JOIN author ON article.id_author = author.author_id
WHERE author.author_id = authorID
ORDER BY article.article_id;
```

Alter:

```
ALTER TABLE article
ADD PRIMARY KEY (article_id),
ADD KEY article_category (id_categorie),
ADD KEY article_author (id_author);

ALTER TABLE author
ADD PRIMARY KEY (author_id);

ALTER TABLE category
ADD PRIMARY KEY (category_id);

ALTER TABLE comment
ADD PRIMARY KEY (comment_id),
ADD KEY comment_article (id_article);

ALTER TABLE users
ADD PRIMARY KEY (id),
ADD UNIQUE KEY username (username),
ADD UNIQUE KEY email (email);

ALTER TABLE article
MODIFY article_id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=49;
```

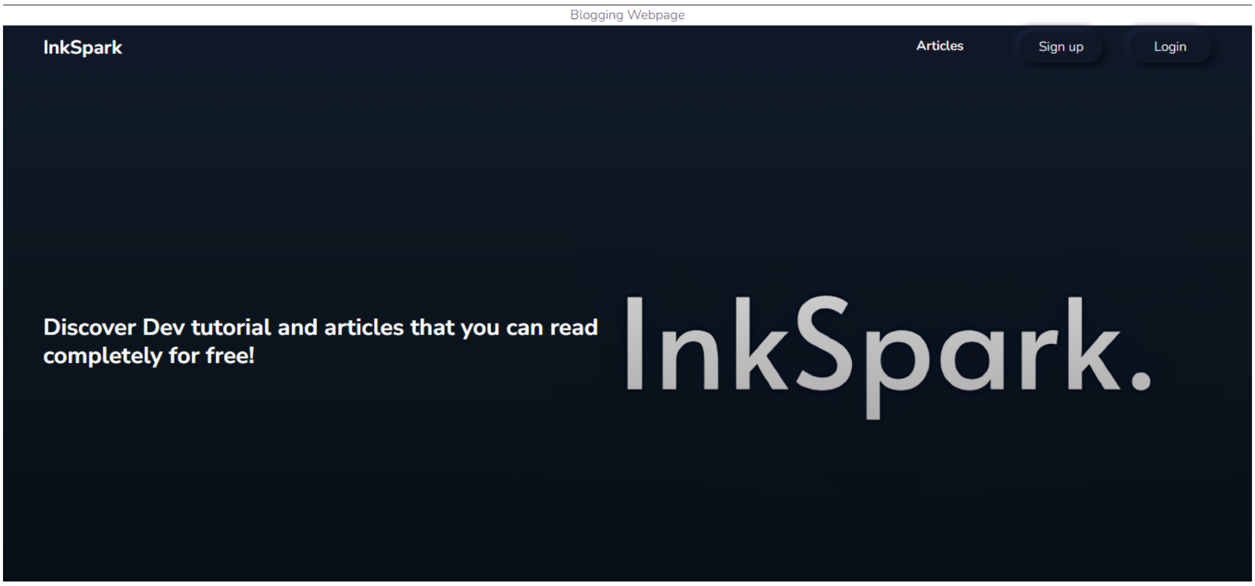
```
ALTER TABLE author
MODIFY author_id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

ALTER TABLE category
MODIFY category_id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;

ALTER TABLE comment
MODIFY comment_id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=23;

ALTER TABLE users
MODIFY id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=14;

ALTER TABLE article
ADD CONSTRAINT article_author FOREIGN KEY (id_author) REFERENCES author
(author_id) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT article_category FOREIGN KEY (id_categorie) REFERENCES
category (category_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE comment
ADD CONSTRAINT comment_article FOREIGN KEY (id_article) REFERENCES article
(article_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

Chapter 8

Results and Discussion

The implementation of the blog management system has been successful, providing a functional platform for users to share and explore articles on various topics. Key features such as user registration, article publishing, category organization, commenting, and administrative controls have been implemented and tested. Users can easily navigate through the platform, read articles, comment on them, and interact with other users. Authors can publish articles and manage their content efficiently. Administrators have the necessary tools to moderate the platform and ensure content quality and adherence to guidelines.

Home Page:
This is the blogging platform's homepage. Users can read articles, comment, and sign up to post their own.



Discover Dev tutorial and articles that you can read completely for free!

Figure 8.1: Home Page

Login Page:

Existing users can log in from this page to access article posting functionality

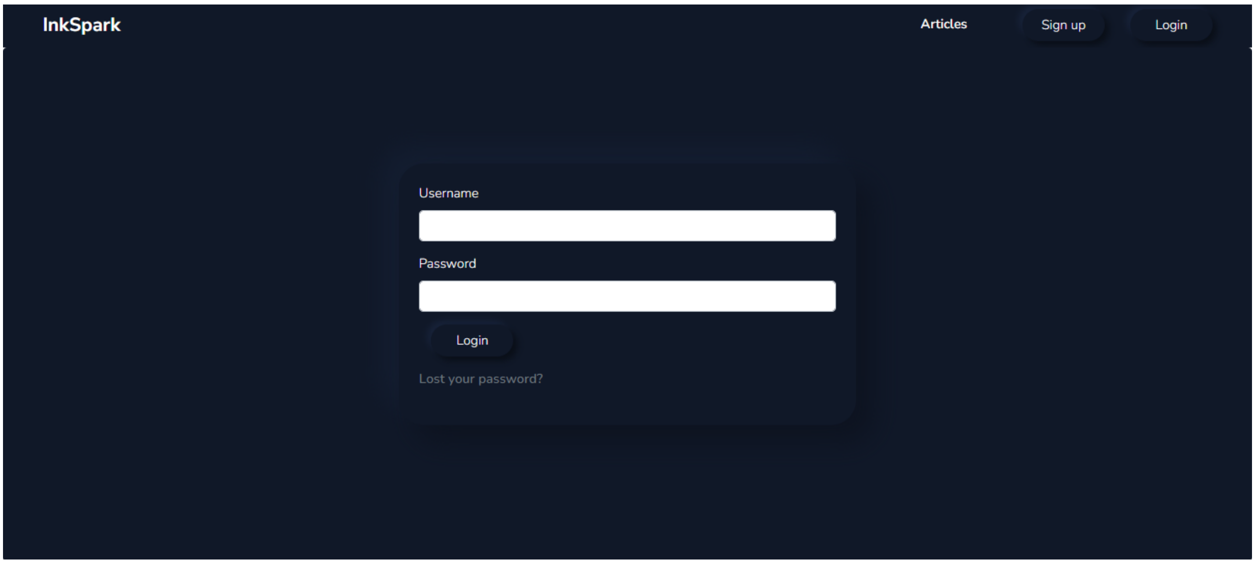


Figure 8.2: Login Page

Sign-up Page:

This page facilitates new user sign-ups, enabling them to become authors and publish their articles on the platform.

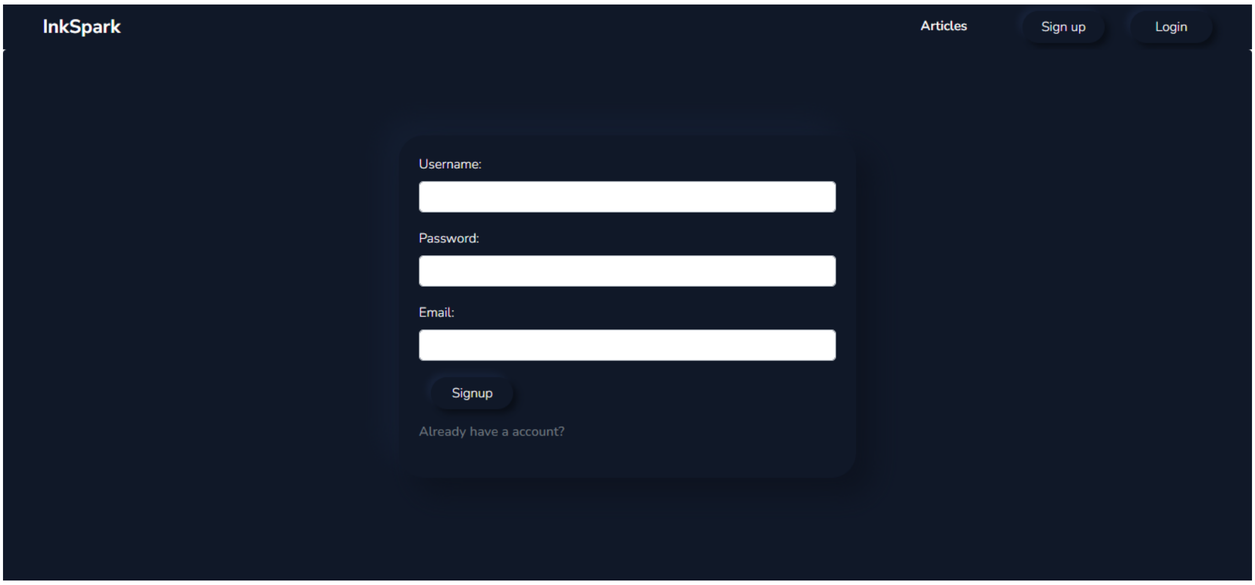
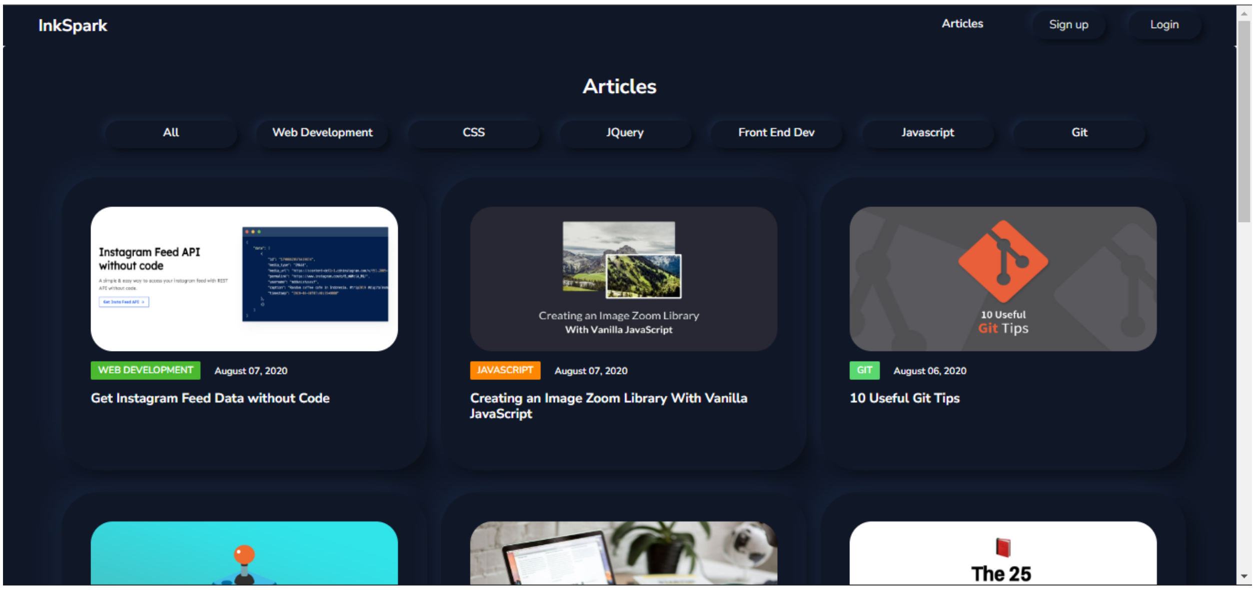


Figure 8.3: Sign-up Page

Article Section:

This page displays all articles contributed by various authors, granting access to all users visiting the site.



WEB DEVELOPMENT August 07, 2020
Get Instagram Feed Data without Code

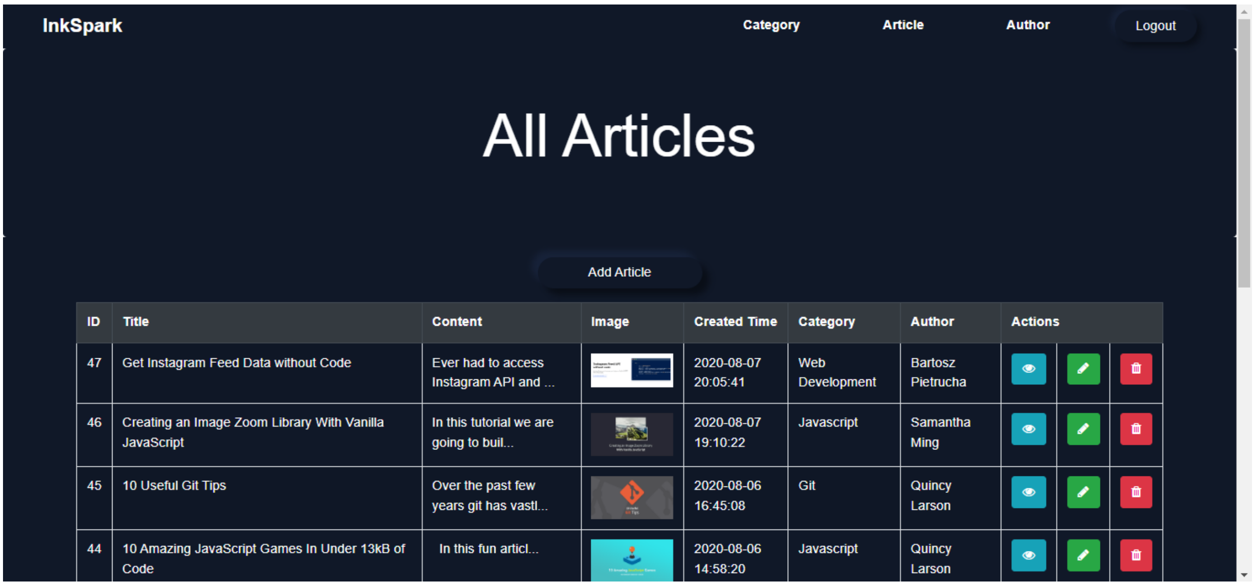
JAVASCRIPT August 07, 2020
Creating an Image Zoom Library With Vanilla JavaScript

GIT August 06, 2020
10 Useful Git Tips

Figure 8.4: Article Section

All Article Page:

Exclusively for admins, this page showcases articles authored by different contributors. Admins retain the ability to edit or remove articles, ensuring content relevance and quality control.

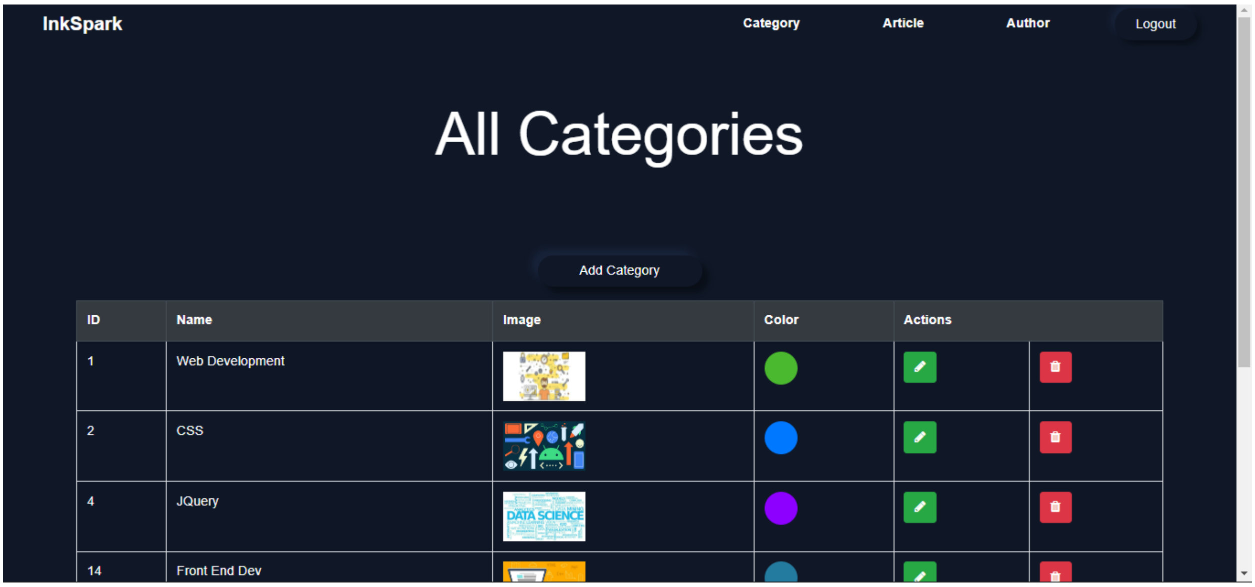


ID	Title	Content	Image	Created Time	Category	Author	Actions
47	Get Instagram Feed Data without Code	Ever had to access Instagram API and ...		2020-08-07 20:05:41	Web Development	Bartosz Pietrucha	👁 ✎ 🗑
46	Creating an Image Zoom Library With Vanilla JavaScript	In this tutorial we are going to buil...		2020-08-07 19:10:22	Javascript	Samantha Ming	👁 ✎ 🗑
45	10 Useful Git Tips	Over the past few years git has vastl...		2020-08-06 16:45:08	Git	Quincy Larson	👁 ✎ 🗑
44	10 Amazing JavaScript Games In Under 13kB of Code	In this fun articl...		2020-08-06 14:58:20	Javascript	Quincy Larson	👁 ✎ 🗑

Figure 8.5: All the articles posted by different authors.

All Categories Page:

Accessible solely to admins, this page showcases article categories. Admins have full control to adjust and maintain the content's thematic focus as needed.

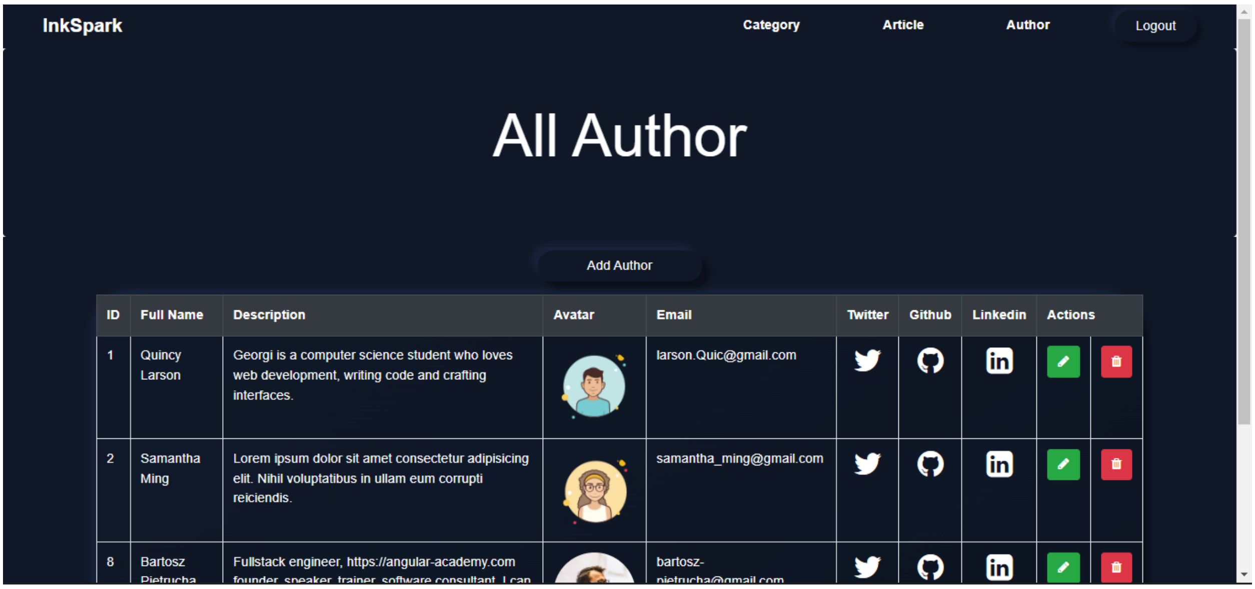


ID	Name	Image	Color	Actions
1	Web Development		●	
2	CSS		●	
4	JQuery		●	
14	Front End Dev		●	

Figure 8.6: Different Categories of articles

All Authors:

Exclusive to admins, this page displays author details. Admins hold the power to delete authors based on content relevance and rectify any inaccuracies in author descriptions.

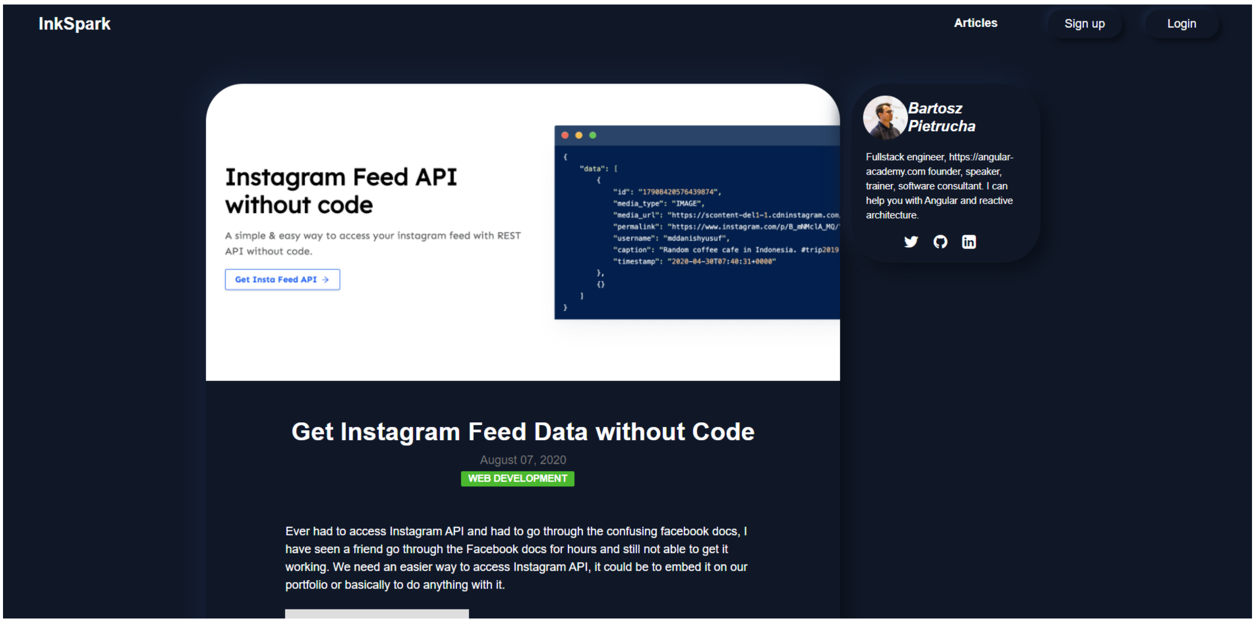


ID	Full Name	Description	Avatar	Email	Twitter	Github	Linkedin	Actions
1	Quincy Larson	Georgi is a computer science student who loves web development, writing code and crafting interfaces.		larson.Quic@gmail.com				
2	Samantha Ming	Lorem ipsum dolor sit amet consectetur adipisicing elit. Nihil voluptatibus in ullam eum corrupti reiciendis.		samantha_ming@gmail.com				
8	Bartosz Pietrucha	Fullstack engineer, https://angular-academy.com founder, speaker, trainer, software consultant. I can		bartosz-pietrucha@gmail.com				

Figure 8.7: Deatails of the authors

Article Page:

Upon opening an article, users are presented with the article content alongside the author's description and comments pertaining to the post.



Instagram Feed API without code

A simple & easy way to access your instagram feed with REST API without code.

```
{
    "data": [
        {
            "id": "17908420576439874",
            "media_type": "IMAGE",
            "media_url": "https://scontent-del1-1.cdninstagram.com
            "permalink": "https://www.instagram.com/p/B_mNMclA_MQ/
            "username": "mddanishyusuf",
            "caption": "Random coffee cafe in Indonesia. #trip2019
            "timestamp": "2020-04-30T07:40:31+0000"
        },
        {}
    ]
}
```

Bartosz Pietrucha

Fullstack engineer, https://angular-academy.com founder, speaker, trainer, software consultant. I can help you with Angular and reactive architecture.

Get Instagram Feed Data without Code

August 07, 2020

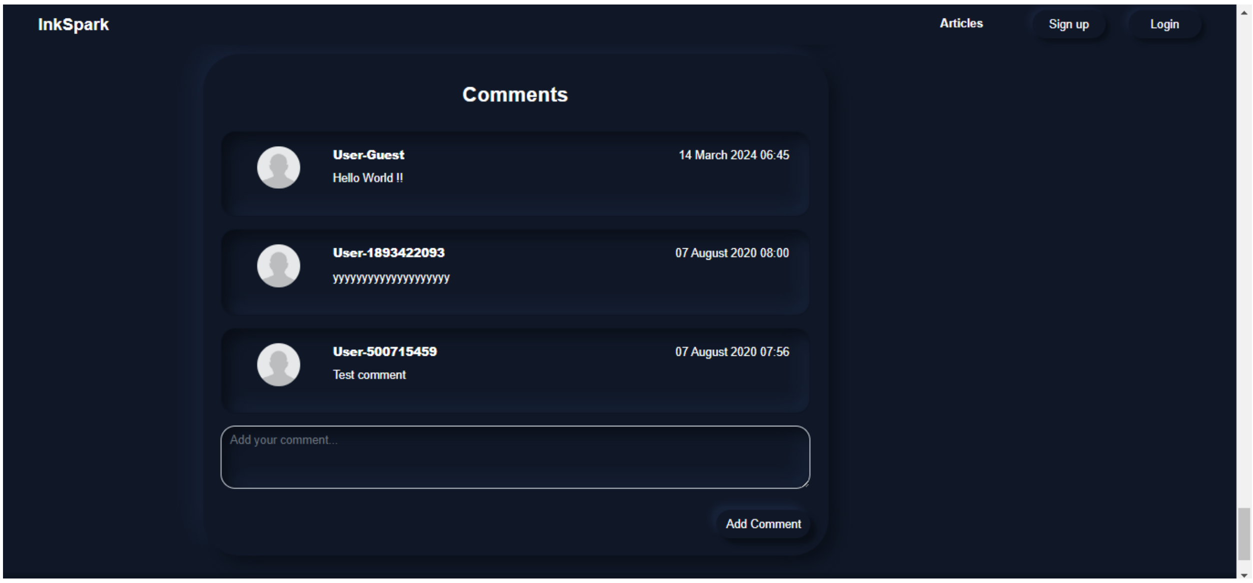
WEB DEVELOPMENT

Ever had to access Instagram API and had to go through the confusing facebook docs, I have seen a friend go through the Facebook docs for hours and still not able to get it working. We need an easier way to access Instagram API, it could be to embed it on our portfolio or basically to do anything with it.

Figure 8.8: Article posted by an author

Comment Section:

This section displays comments from various users on the article, offering users the ability to contribute their own comments as well.



Comments

User-Guest 14 March 2024 06:45
Hello World !!

User-1893422093 07 August 2020 08:00
yyyyyyyyyyyyyyyyyy

User-500715459 07 August 2020 07:56
Test comment

Figure 8.9: Comments on an article

Chapter 9

Conclusion and Future work

The development of the blog management system has provided a robust platform for users to share knowledge and insights through articles. The system facilitates seamless interaction between users, authors, and administrators, fostering a collaborative environment for learning and exchange of ideas. By categorizing articles based on content and providing features for commenting and engagement, the system promotes active participation and meaningful discussions within the community. Administrators play a crucial role in maintaining the integrity and quality of content, ensuring adherence to guidelines and standards. Overall, the blog management system serves as a dynamic and inclusive platform for knowledge sharing, contributing to the growth and enrichment of the user community.

Moving forward, several enhancements and features can be implemented to further enhance the functionality and user experience of the blog management system.

Some potential areas for future work include:

- Reporting and Analytics: Implementing reporting and analytics features to provide insights into user engagement, article performance, and trends. This can help identify popular topics, user preferences, and areas for improvement.
- Enhanced User Profiles: Enhancing user profiles with additional features such as profile customization, user badges or achievements, and social integration to promote user engagement and recognition.
- Content Recommendations: Implementing personalized content recommendations based on user preferences, browsing history, and interactions to improve content discoverability and user engagement.

- Gamification Elements: Introducing gamification elements such as leaderboards, badges, or rewards to incentivize user participation, encourage contributions, and foster a sense of community.
- Integration with Social Media: Integrating with social media platforms to enable seamless sharing of articles, cross-platform engagement, and broader reach for content published on the platform.
- Monetization Options: Exploring monetization options such as advertising, premium content subscriptions, or sponsored content partnerships to generate revenue and sustain the platform's growth.

By incorporating these enhancements and features, the blog management system can evolve into a more dynamic, engaging, and valuable platform for users, authors, and administrators alike, catering to the diverse needs and preferences of its user community.

References

[1] Database systems Models, Languages, Design and Application Programming, Ramez Elmasri and Shamkant B. Navathe, 6th Edition, Pearson.

[2] Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.

[3] Silberschatz Korth and Sudharshan: Database System Concepts, 6th Edition, McGraw Hill, 2013.

[4] Coronel, Morris, and Rob, Database Principles Fundamentals of Design, Implementation and Management, Cengage Learning 2012