# CHAPTER III: SYSTEM DESIGN

## 3.1. System Architecture and Overview

System design is the process of defining the architecture components, modules, interface and data for a system to satisfy specified requirements. It involves specifying how the system will accomplish its objectives and meet the needs of users, considering factors such as efficiency, scalability and maintainability. It is a crucial phase in the software development life cycle (SDLC) and involves transforming the requirements gathered during the analysis phase into a blueprint for the actual system. We have developed "CONTENT MANAGEMENT SYSTEM (GYANSHRISTI)" which required MongoDB for its functioning. To use this system, we need a web browser, internet, laptop/ desktop and the users.

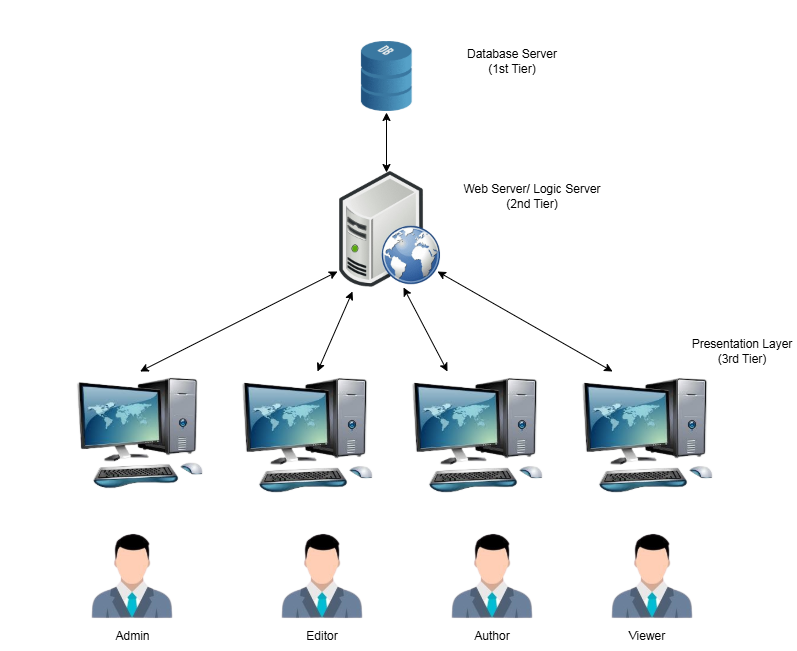


Figure 3.1: System Architecture

## 3.2. System Design

System design is the process of planning and defining the structure, components, interfaces, and data of a system to fulfill specified requirements. It involves creating a detailed blueprint that guides the implementation of the system, covering aspects such as architecture, components, data organization, interfaces, procedures, and security.

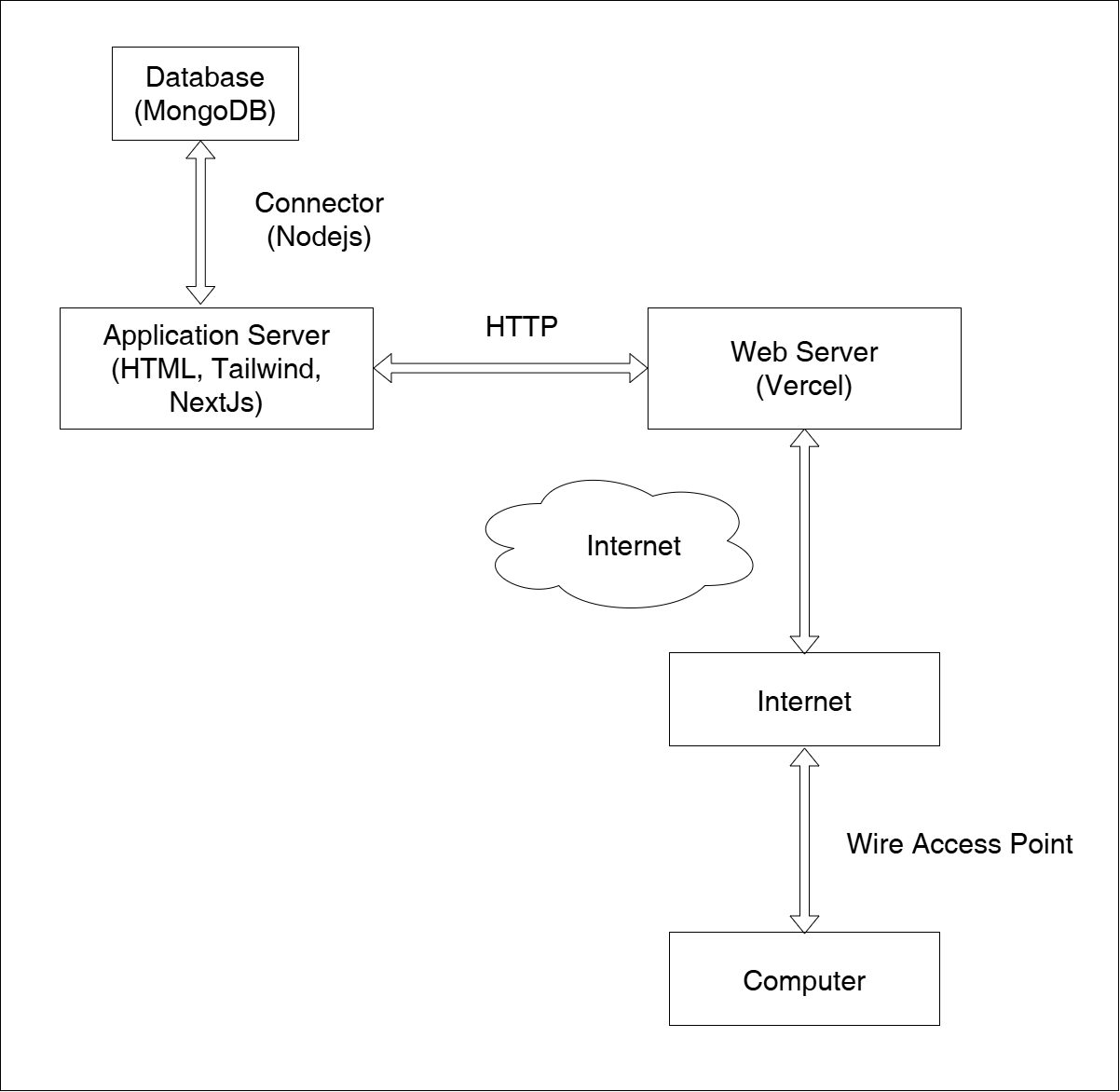


Figure 3.2: Implementation architecture of Content Management System

## 3.3. Interface Design

Interface design is the process of creating visual and interactive elements for computer systems, software, websites, or applications to make them user-friendly and aesthetically pleasing. It involves designing the layout, appearance, and functionality of on-screen elements such as buttons, menus, and navigation, with the aim of providing a positive and efficient user experience.

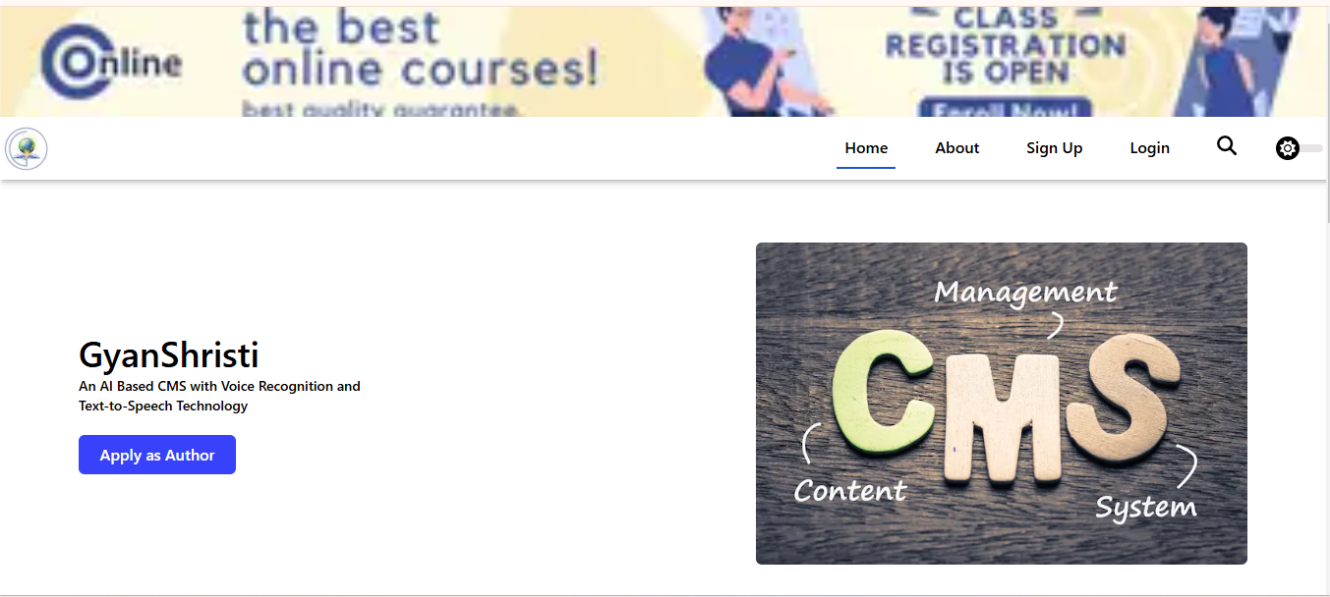


Figure 3.3: Homepage of Viewer

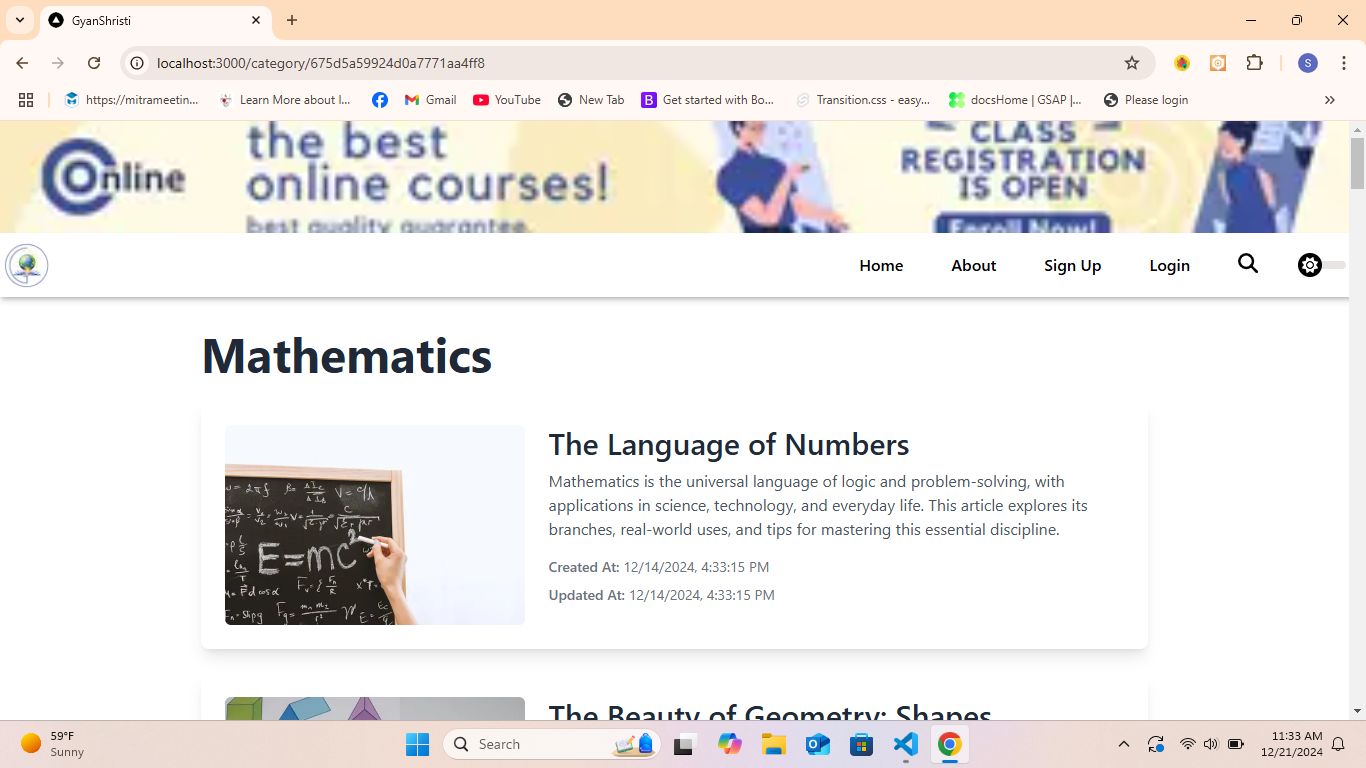


Figure 3.4: Category Wise Content of Viewer

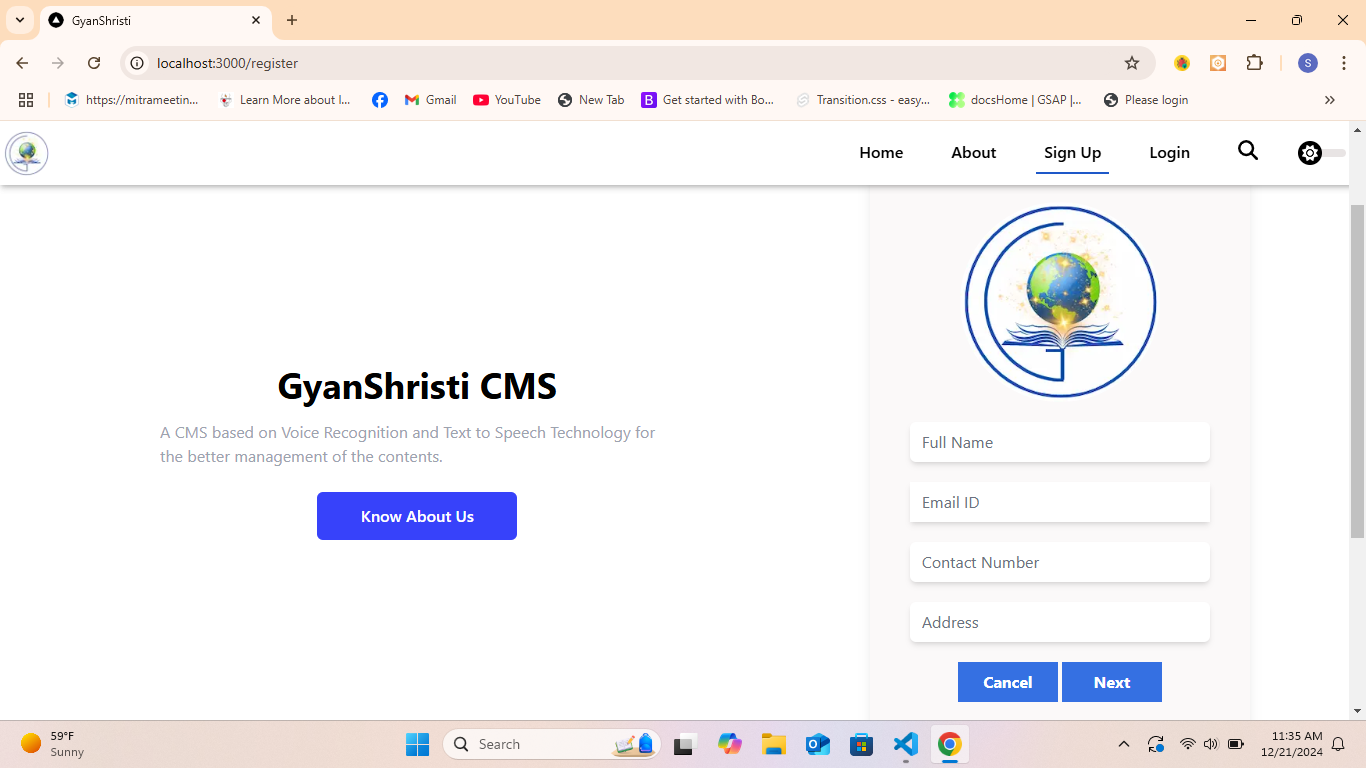


Figure 3.5: Sign up Page of Non-registered User

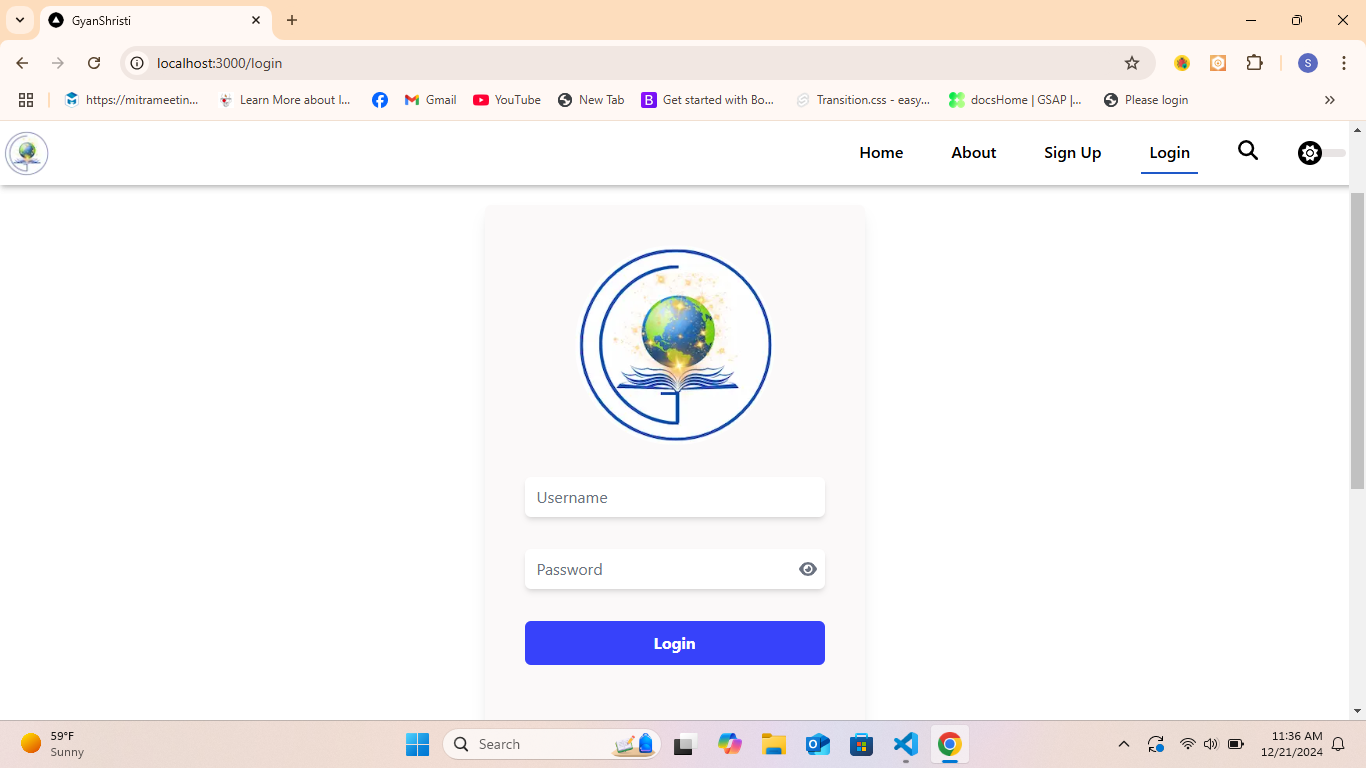


Figure 3.6: Login Page

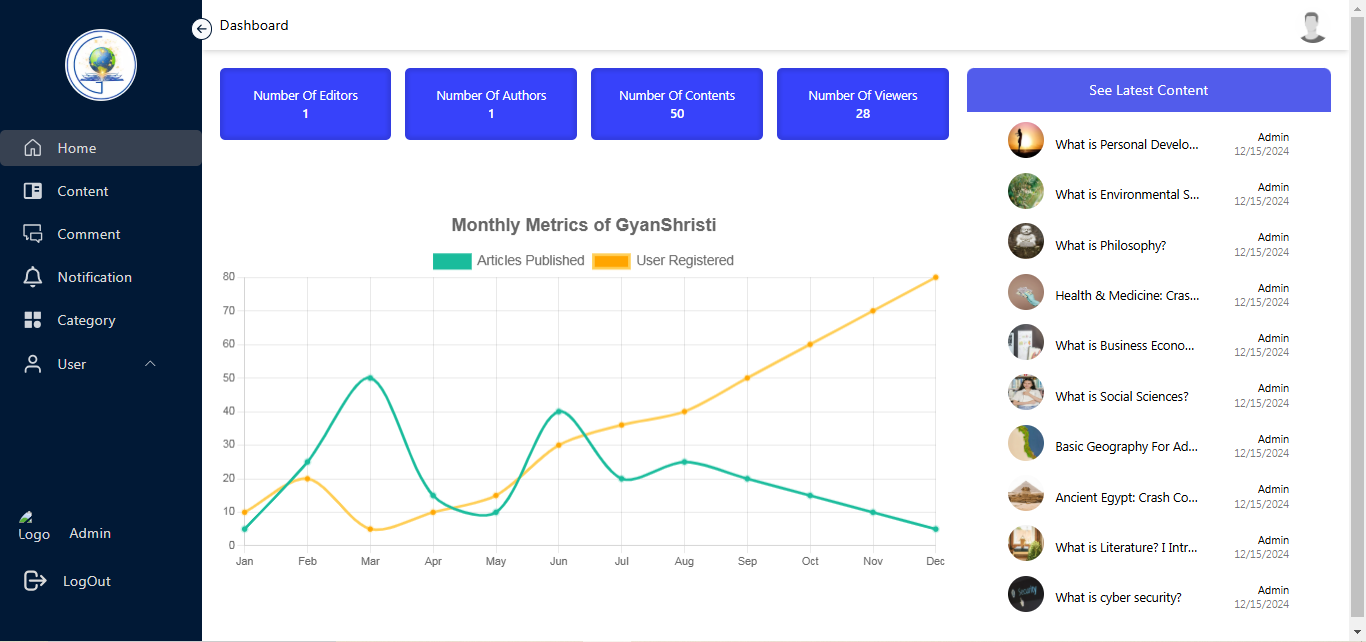


Figure 3.7: Dashboard Page of Admin

## 3.4. Database Schema

A database schema in MongoDB defines the structure of data in collections and documents, allowing flexibility with varying fields, while tools like Mongoose can enforce consistent patterns and validation.

**3.4.1 UserSchema**

const UserSchema = new mongoose.Schema({  
name: {  
type: String,  
required: true,  
},  
username: {  
type: String,  
required: true,  
},  
password: {  
type: String,  
required: true,  
},  
email: {  
type: String,  
required: true,  
},  
address: {  
type : String,  
required : true,  
},  
profile\_pic: {  
type: String,  
},  
phone\_number: {  
type: Number,  
required: true,  
},  
role: {  
type: String,  
enum: ["admin", "editor", "author", "viewer"],  
default: "viewer",  
},  
status: {  
type: String,  
enum: ["unrequested", "pending", "approved","rejected"],  
},  
otp:{  
type: String  
},  
created\_at: {  
type: Date,  
default: Date.now,  
},  
});

**3.4.2 ContentSchema**

const ContentSchema = new mongoose.Schema({  
title: {  
type: String,  
required: true,  
},  
description: {  
type: String,  
required: true,  
},  
content\_type: {  
type: String,  
enum: ["pdf", "video", "post"],  
required: true,  
},  
location: {  
type: String,  
},  
blog: {  
type: String,  
},  
thumbnail: {  
type: String,  
default:  
"https://images.pexels.com/photos/1323550/pexels-photo-1323550.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1",  
},  
status: {  
type: String,  
enum: ["Pending", "Uploaded", "Rejected"],  
default: "Pending",  
},  
user\_id: {  
type: mongoose.Schema.Types.ObjectId,  
ref: "User",  
},  
category\_id: {  
type: mongoose.Schema.Types.ObjectId,  
ref: "Category",  
default: null,  
},  
created\_at: {  
type: Date,  
default: Date.now,  
},  
updated\_at: {  
type: Date,  
default: Date.now,  
},  
});

**3.4.3 CommentSchema**

const CommentSchema = new mongoose.Schema({  
description : {  
type : String,  
required : true  
},  
user\_id : {  
type : mongoose.Schema.Types.ObjectId,  
ref : 'User'  
},  
content\_id : {  
type : mongoose.Schema.Types.ObjectId,  
ref : 'Content'  
},  
parent\_comment\_id :[{  
type : mongoose.Schema.Types.ObjectId,  
ref : 'Comment'  
}],  
created\_at : {  
type : Date,  
default : Date.now  
}  
});

**3.4.4 CategorySchema**

const CategorySchema = new mongoose.Schema({  
title: {  
type: String,  
required: true,  
},  
user\_id: {  
type: mongoose.Schema.Types.ObjectId,  
ref : "User"  
}  
});

**3.4.5 LoginSchema**

const LoginSchema = new mongoose.Schema({  
user\_id: {  
type: mongoose.Schema.Types.ObjectId,  
required: true,  
},  
created\_at: {  
type: Date,  
default: Date.now,  
},  
});