1. Introduction

News & media portals is a web application, it is a digital version of a traditional print newspaper, which can be accessed via the internet. It typically includes the same types of content as a print newspaper, such as news articles, editorials, and advertisements, but it may also include additional features such as videos, interactive graphics, and social media integration. Online newspapers are often updated throughout the day, providing readers with the latest news and information as it becomes available. They can be accessed on a variety of devices, including computers, smartphones, and tablets, and many offers both free and paid content. Some online newspapers are affiliated with print newspapers, while others are entirely digital publications.

1.1 Advantage

- Users can access the latest news and information instantly, breaking free from traditional time constraints.
- Portals cover a wide range of topics, providing diverse content from politics and current events to entertainment and lifestyle.
- Incorporation of multimedia elements such as videos, images, and audio clips enhance the overall user experience.
- Search functionalities help users find specific articles quickly, while archives provide access to historical news content.
- News portals can adapt to evolving technologies, incorporating new features and ensuring relevance in the digital landscape.

1.2 Objective

- Provide timely and accurate information to a wide audience on variety of topics.
- Increase public awareness by reporting on significant issues and events that impact society.
- Foster user engagement through interactive features such as comments, likes, shares, and participation in forums
- Provide educational content and in-depth analysis to offer users a deeper understanding of complex issues.
- Facilitate easy navigation and information retrieval through effective search functionalities.

1.3 Goals

- Provide unbiased and objective reporting.
- Deliver news in a timely manner.
- Cover a wide range of topics for comprehensive news coverage.
- Present news and information in various formats (articles, videos, images, audio).
- Integrate with social media platforms for wider content distribution.

2. PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million website and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1995, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for Personal Home Page, it now stands for PHP; Hypertext Preprocessor, a recursive acronym. PHP code is interpreted by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an HTML source document. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP is free software released under the PHP License. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

2.1 Php Syntax

```
<? Php

Echo ‘Hello World’;

?>
```

2.2 Why we use PHP?

You have obviously head of a number of programming language out there you may be wondering why we should want to use PHP as our poison for the web programming. Below are some of the compelling reasons.

1. PHP is open source and free.
2. Large community document.
3. It is regular updated to keep abreast with the latest technology trends.

3. HTML

HTML stands for Hypertext Markup Language for creating web pages

- HTML stands for hypertext markup language.
- It describes structure of web pages.
- HTML elements are represented by tags.
- It consists series of elements.

3.1 HTML Example:

```
<html>
  <head>
      <Title>page title</title>
  </head>
      <body>
          <h1>my first Heading</h1>
          <p>my first paragraph</p>
      </body>
  </html>
```

3.2 Advantages of HTML:

1. The first advantage it is widely used.
2. Every browser support HTML language.
3. Easy to learn and use.
4. It is by default in every window so you don't need to purchase extra software.
5. We can integrate HTML with CSS, JavaScript, and Php etc.

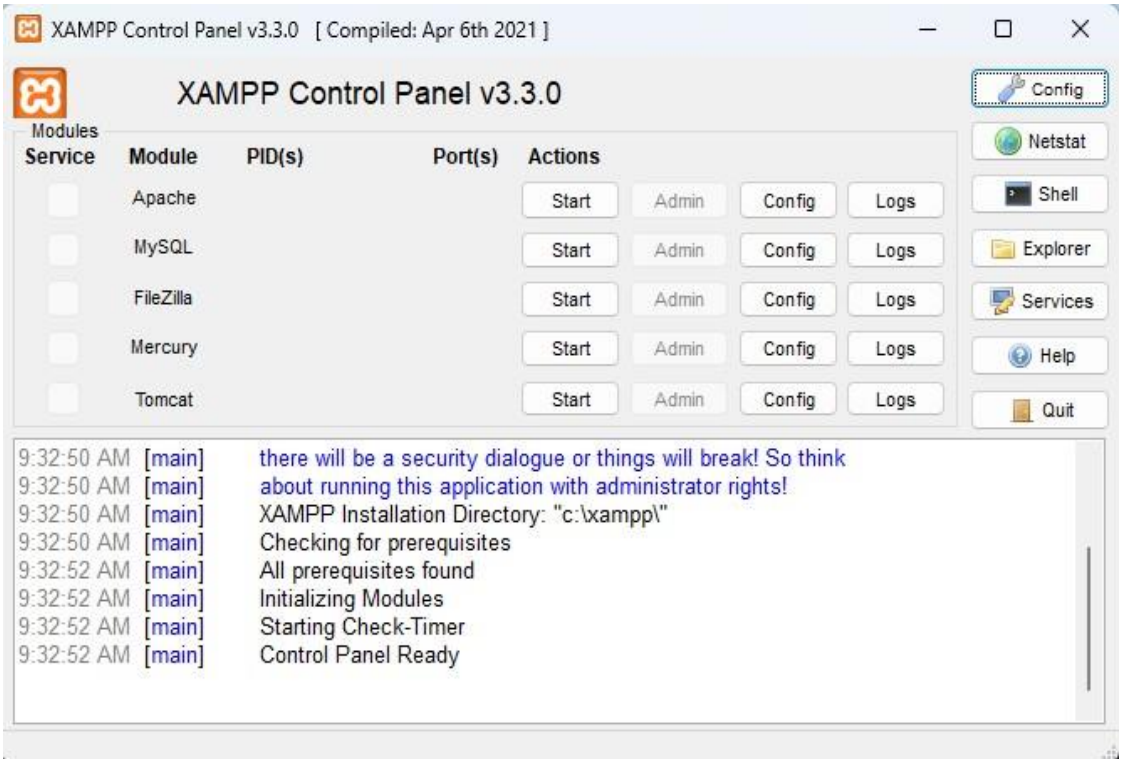
3.3 Disadvantages of HTML:

1. It can create only static and plain pages so if we need dynamic pages then HTML.
2. Need to write a lot of codes for making simple webpage
3. Security features are not good in HTML.
4. If we need to write long code for making a webpage then it produces some complexity.

4. Xampp

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, Maria DB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

XAMPP's ease of deployment means a WAMP or LAMP stack can be installed quickly and simply on an operating system by a developer, with the advantage that common add-in applications such as Word Press and Joomla! Can also be installed with similar ease using Bitnami



I. Xampp

5. System Analysis

5.1 Requirement Specification:

a) Software Requirements:

- Operating System: - Window 8, Window 10, Window 11, Linux etc.
- Framework:- Laravel 5.8*
- Server: - Xampp, Wamp, etc.
- Database:- MySQL(phpmyadmin)
- Text Editor: - Vs. Code, Notepad++ etc.
- Other Software: - Git, Composer etc.

b) Hardware Requirements:

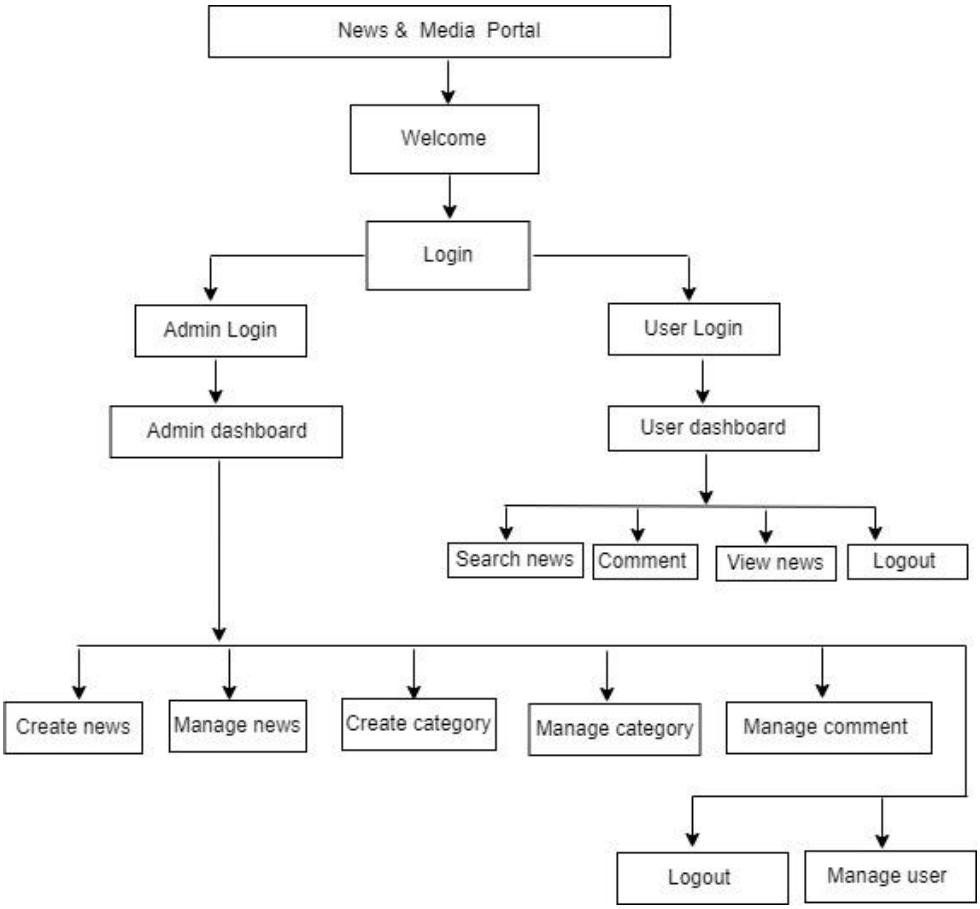
- Processor: - intel i3 or more
- Processor Speed: - 2.30GHz or more
- Hard Disk: - 100 GB or more
- RAM: - 4GB or more
- Other: - Keyboard, Mouse, etc.

.

6. System Design:

Module is a type of diagram which each of a set of standardized parts or independent units that can be used to construct a more complex structure, Such as an item of furniture or a building.

6.1 Module Diagram



I. Fig: Module Diagram

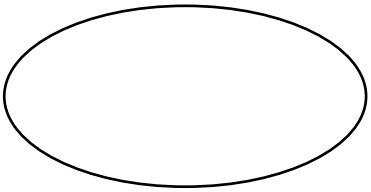
7. ER Diagram:

ER diagram stands for Entity Relationship Diagram. It is a High Level Data Diagram. This Diagram is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER diagram, the database structure is portrayed as a diagram called an entity relationship diagram. The components of ER diagram are explained below: -

a) Entity**:** An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as Rectangles.

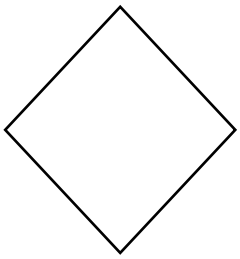
a) **Fig: Rectangle**

b) Attribute: The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.



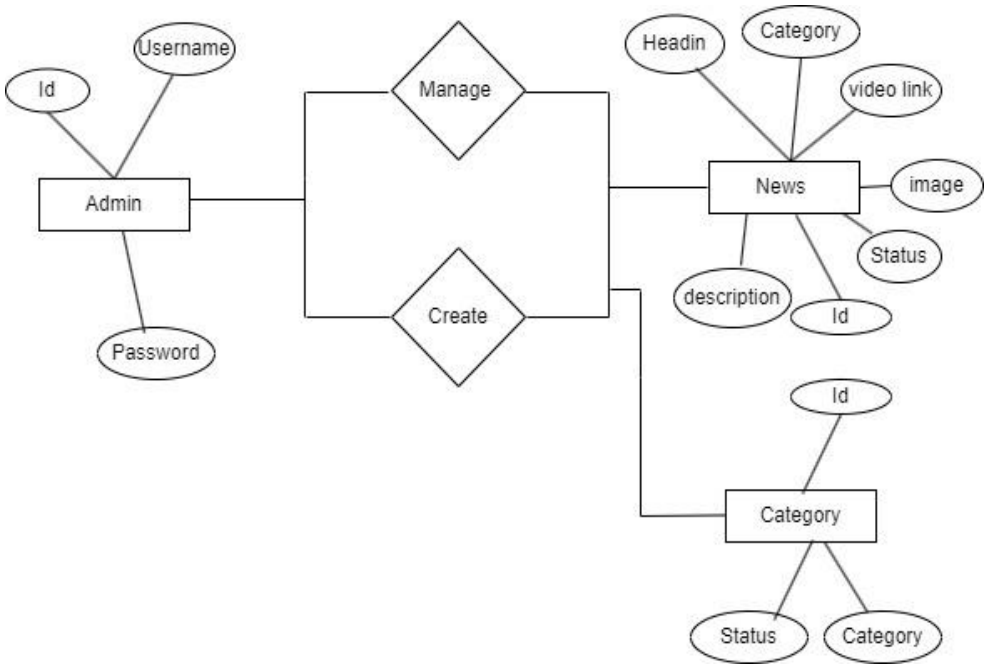
b) Fig: Eclipse

c) Relationship: A Relationship is used to describe the relation between entities. Diamond or Rhombus is used to represent the relationship.



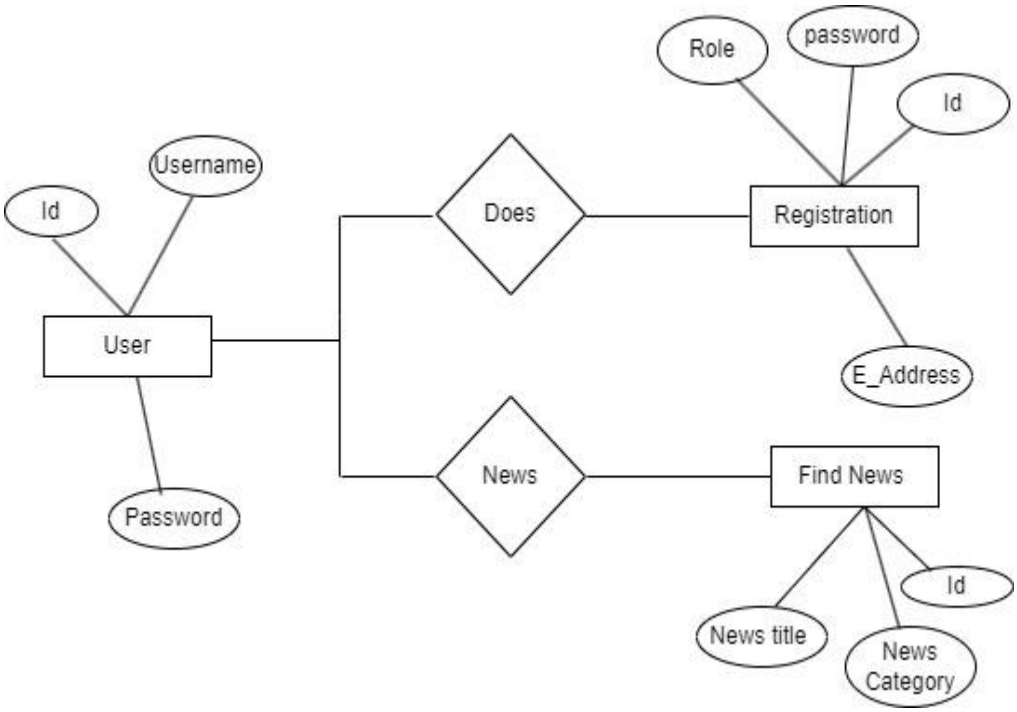
c) Fig: Diamond

7.1 E-R Diagram For Admin



II. Fig: ER-Diagram for Admin

7.2 E-R Diagram For User

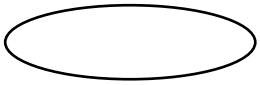


III. Fig: ER-Diagram for User

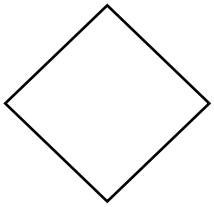
8. Flowchart Diagram:

A Flowchart is a visual representation of a process that makes it easy to understand the process at a glance. Flow Charts depict the nature and flow of steps in a process. Steps and decision points of a process are linked by connecting lines and directional arrows showing process flow direction. This makes it easy for anyone to rationally follow the process from beginning to end. It is important to note that each process step is represented by a different symbol showing different types of actions in a process. There are mainly four types of flowchart symbol which are given below: -

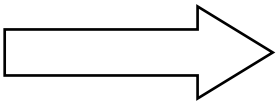
a) **Terminator:** The Terminator symbol represents the starting or ending point of the system.



b) **Decision**: A diamond represents a decision or branching point. Lines coming out from the diamond indicates different possible situations, leading to different sub processes.

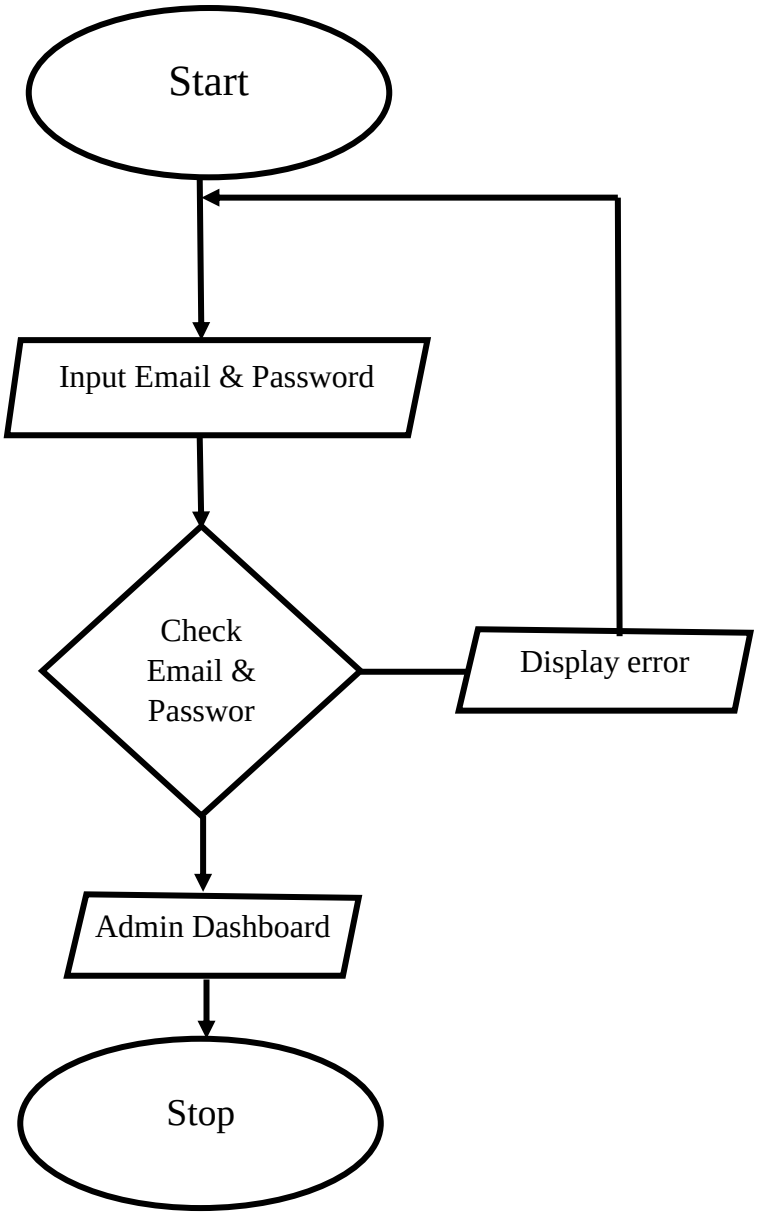


c) **Flow**: Lines represent the flow of the sequence and direction of a process.



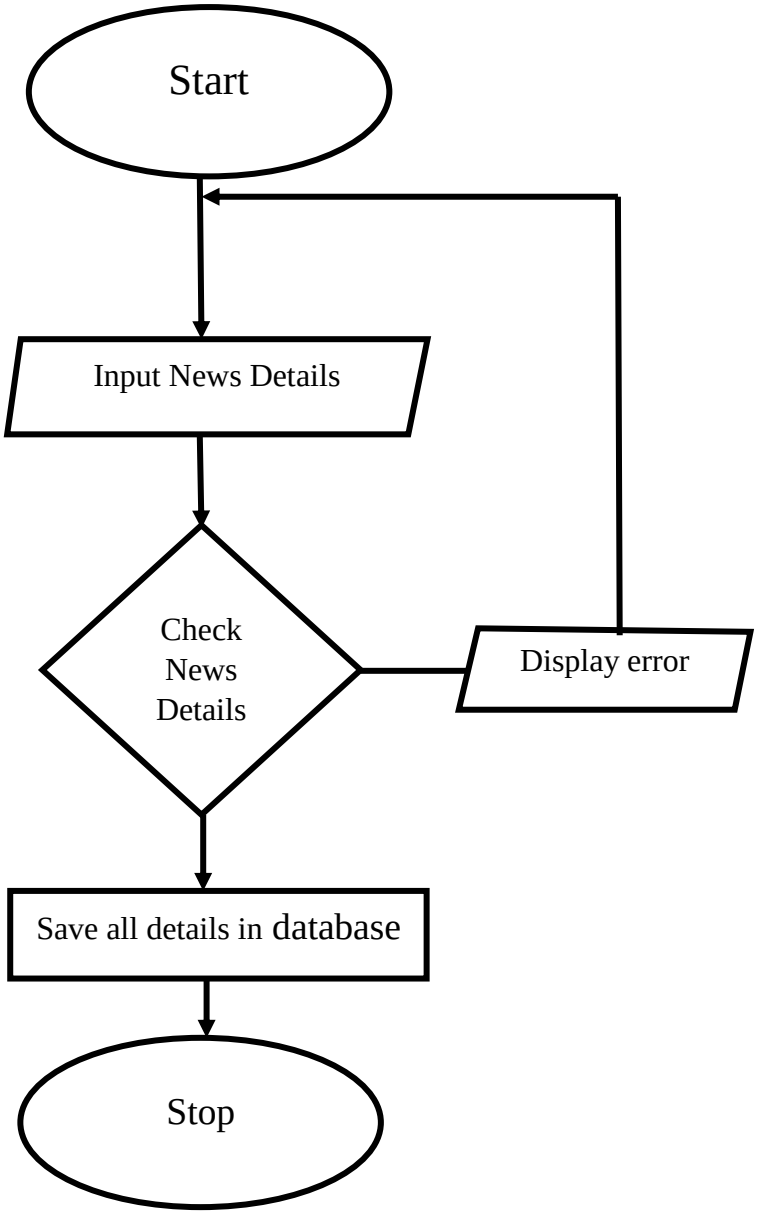
d) **Data**: It represents information entering or leaving the system. An input might be an order forms the customer. Output can be a product to be delivered.

8.1 **Flowchart Diagram For Admin Login:**



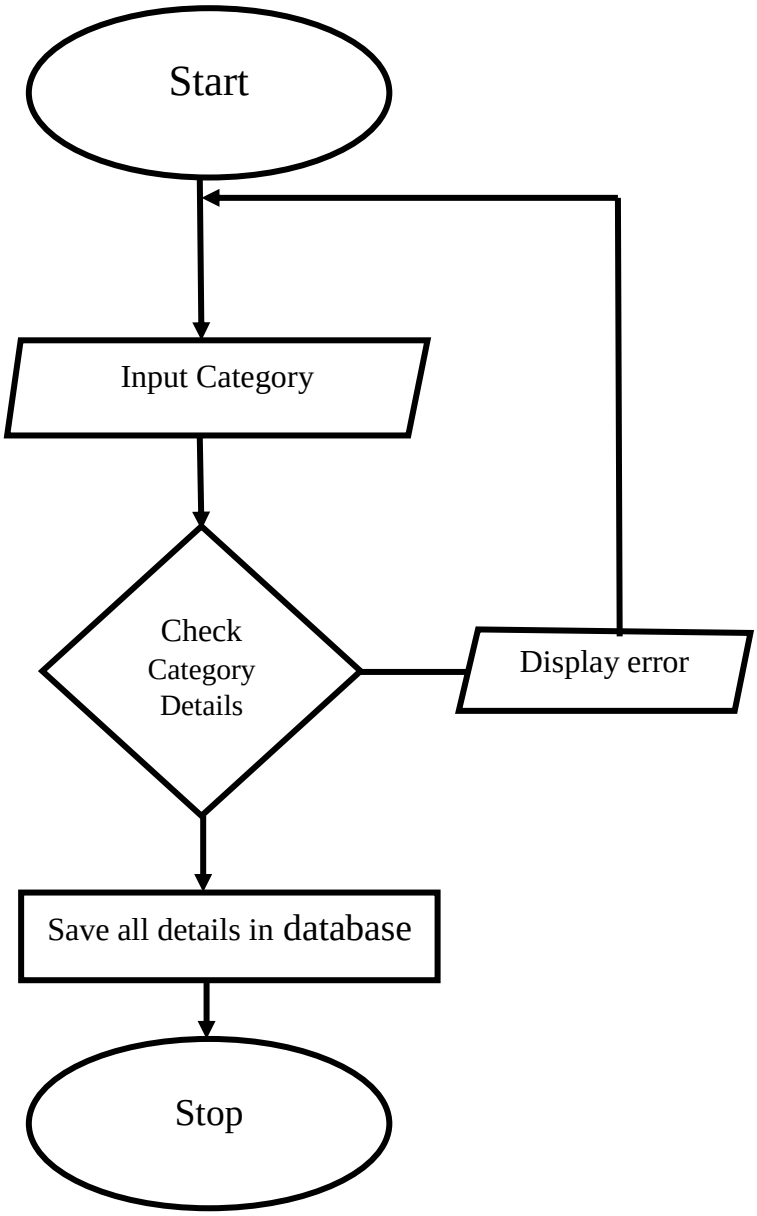
IV. Fig: Flowchart Diagram for Admin Login

8.2 **Flowchart Diagram For News:**



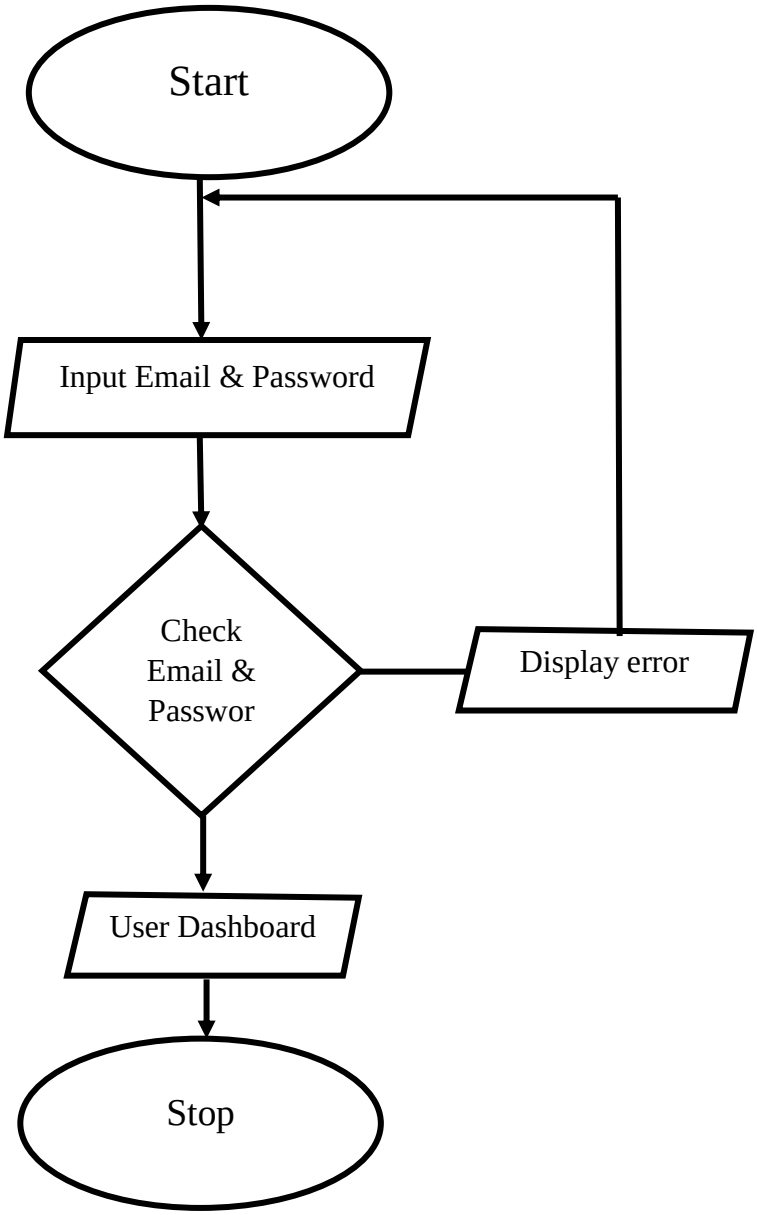
V. Fig: Flowchart Diagram For News

8.3 Flowchart Diagram For Category



VI. Fig: Flowchart Diagram For Category

8.1 Flowchart Diagram For User Login



VII. Fig: Flowchart Diagram For User Login

9. Dataflow Diagram:

Dataflow Diagram is a graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data.

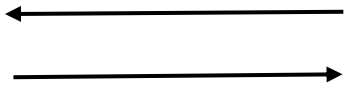
The DFD does not mention anything about how data flows through the system. There is a prominent difference between the DFD and Flowchart. The Flowchart depicts flow of control in program modules. DFD depict flow of data in the system at various levels. DFD does not contain any control or branch elements. The different component of Dataflow Diagram is explained below: -

a) Entities: Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.

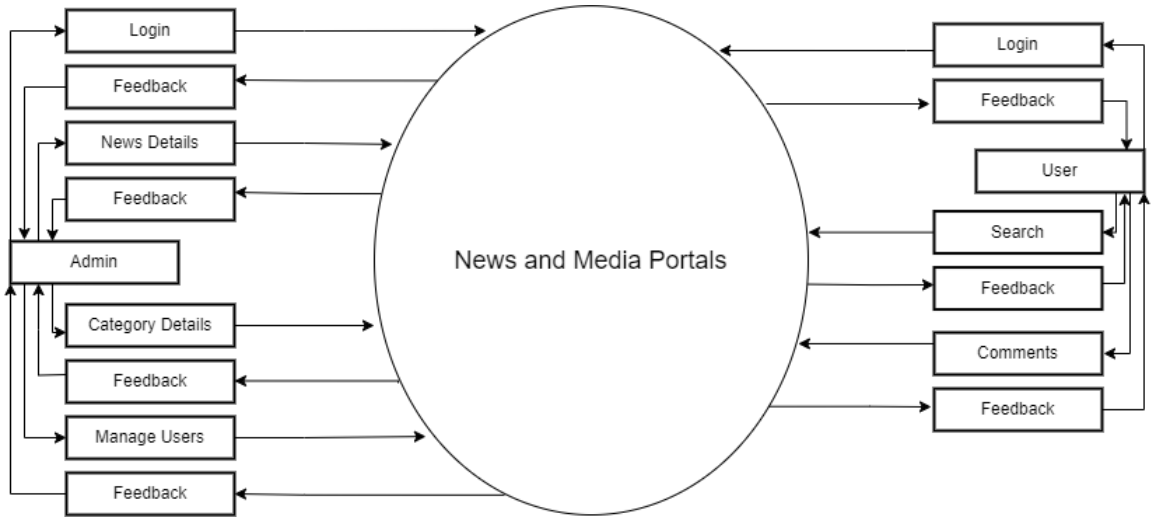
b) Process: Activities action taken on and the data are represented by Circle or oval Shapes.

c) Data Storage: There are two variants of data storage- It can either be represented as a rectangle with absence of both smaller sides or as an open sided rectangle with only one side missing.

d) Data Flow: Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

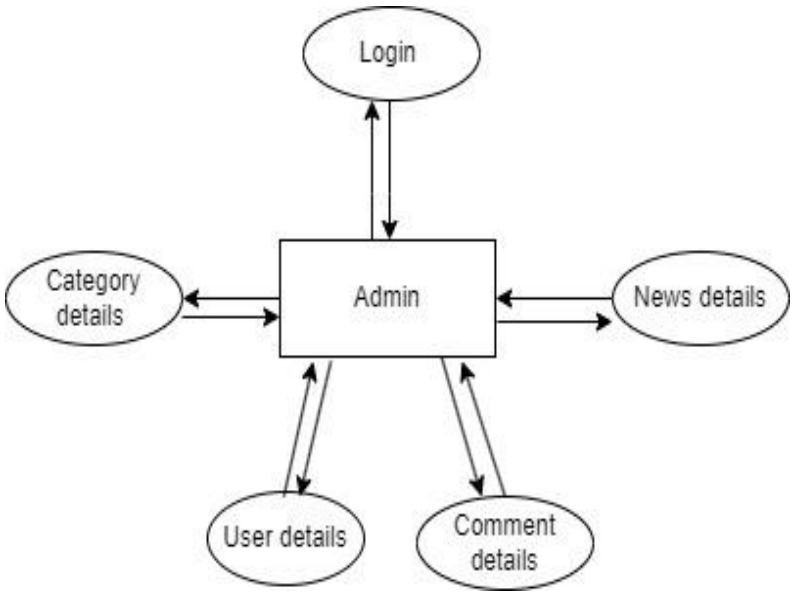


9.1 Context Level Diagram



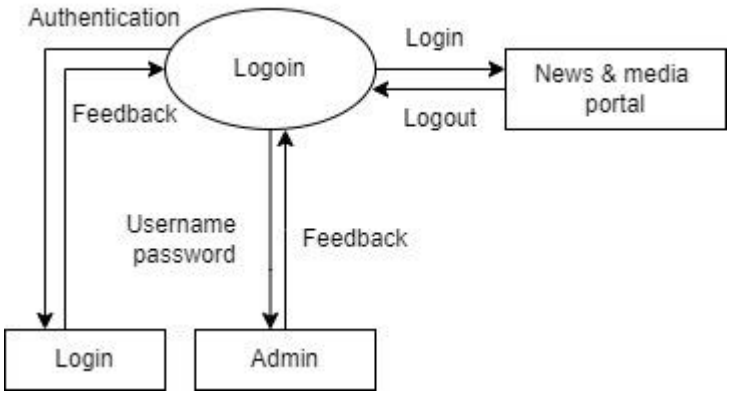
VIII. DFD Level 0: Context Diagram

9.2 DFD Application process: Admin



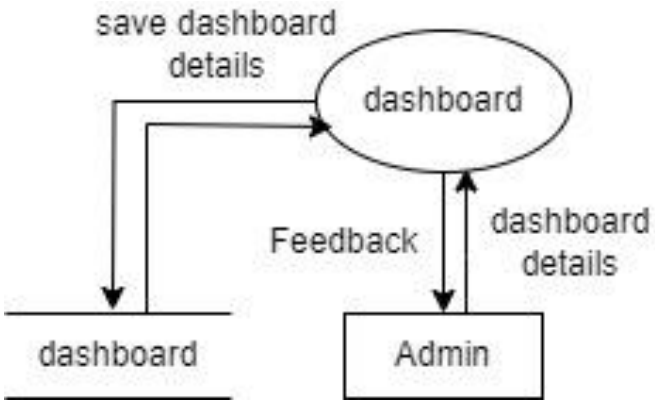
IX. Fig: DFD Level 1.0: Application process: Admin

9.3 DFD Function process: Admin Login



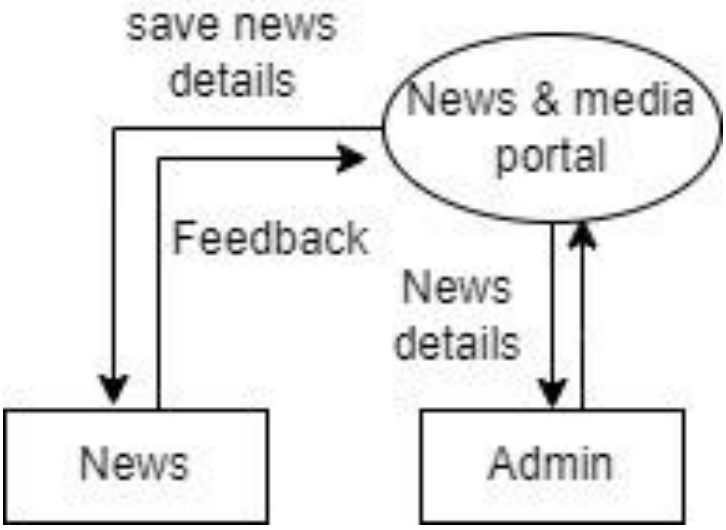
X. Fig: DFD Function process: Login

9.1 DFD Function process: Dashboard



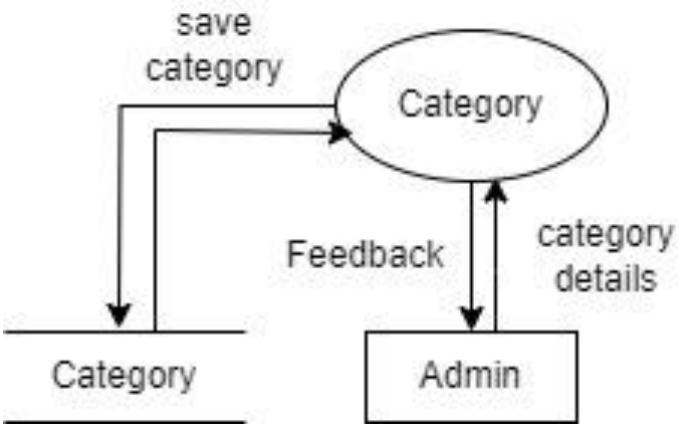
XI. Fig: DFD Function process: Dashboard

9.1 DFD Function Process: News



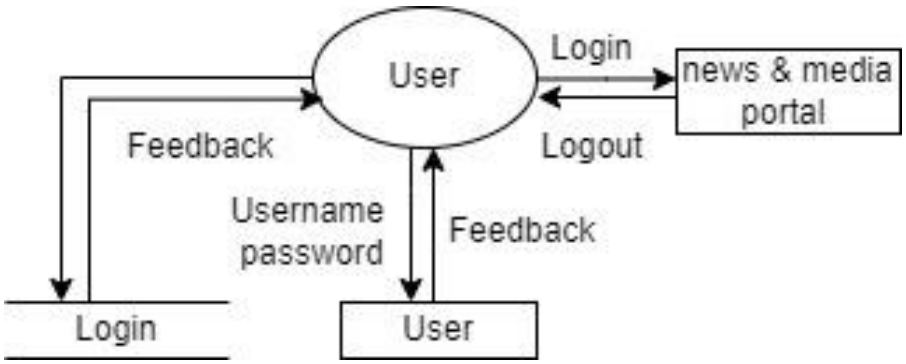
XII. Fig: DFD Function Process: News

9.1 DFD Function process: Category



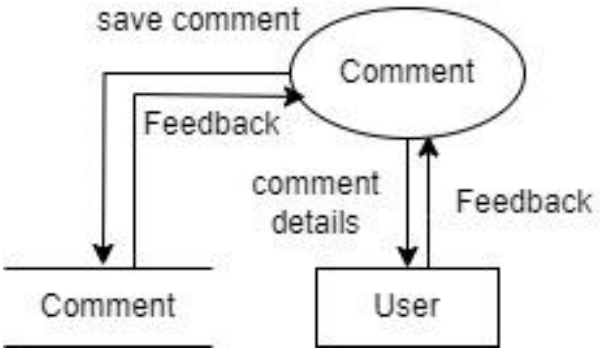
XIII. Fig: DFD Function process: Category

9.1 DFD Function Process: User Login



XIV. Fig: DFD Function process: User Login

9.1 DFD Function Process: Comment



XV. Fig: DFD Function process: Comments

10. Database Diagram

Database diagram is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relation among them are associated. Its formula all the constraints that are to be applied on the data. We can use some datatypes to store data in the database.

10.1 Login Table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
3	email	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	email_verified_at	timestamp			Yes	NULL			Change Drop More
5	password	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
6	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
7	created_at	timestamp			Yes	current_timestamp()			Change Drop More
8	updated_at	timestamp			Yes	current_timestamp()			Change Drop More
9	role	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
10	profile	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More

XVI. Fig: Login Table

10.2 News Table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	category_id	bigint(20)		UNSIGNED	No	None			Change Drop More
3	heading	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	description	longtext	utf8mb4_unicode_ci		No	None			Change Drop More
5	status	tinyint(4)			No	None			Change Drop More
6	image	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
7	link	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
8	created_at	timestamp			Yes	current_timestamp()			Change Drop More
9	updated_at	timestamp			Yes	current_timestamp()			Change Drop More
10	subheading	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More

XVII. Fig: News Table

10.3 Comment Table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
☐	2	category_id	bigint(20)		UNSIGNED	No	None			Change Drop More
☐	3	user_id	bigint(20)		UNSIGNED	No	None			Change Drop More
☐	4	comment	longtext	utf8mb4_unicode_ci		No	None			Change Drop More
☐	5	created_at	timestamp			Yes	current_timestamp()			Change Drop More
☐	6	updated_at	timestamp			Yes	current_timestamp()			Change Drop More
☐	7	news_id	bigint(20)		UNSIGNED	No	None			Change Drop More

XVIII. Fig Comment Table

10.4 Category Table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
☐	3	status	tinyint(4)			No	1			Change Drop More
☐	4	created_at	timestamp			Yes	NULL			Change Drop More
☐	5	updated_at	timestamp			Yes	NULL			Change Drop More

XIX. Fig: Category Table

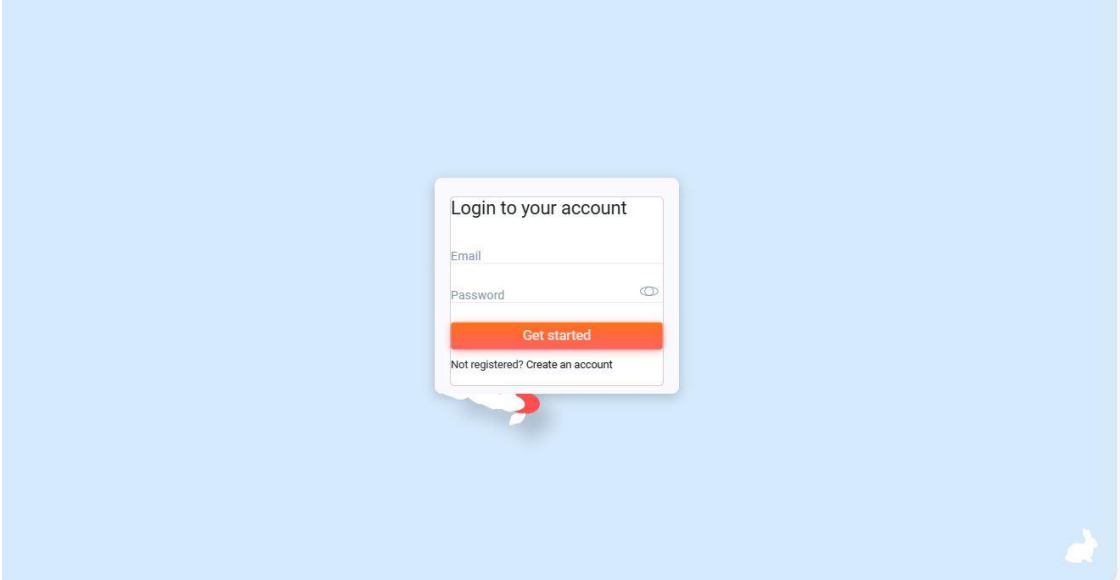
10.5 System Table

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	3	email	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	4	date	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	5	phone	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	6	address	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	7	slogan	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	8	logo	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
☐	9	created_at	timestamp			Yes	NULL			Change Drop More
☐	10	updated_at	timestamp			Yes	NULL			Change Drop More

XX. Fig: System Table

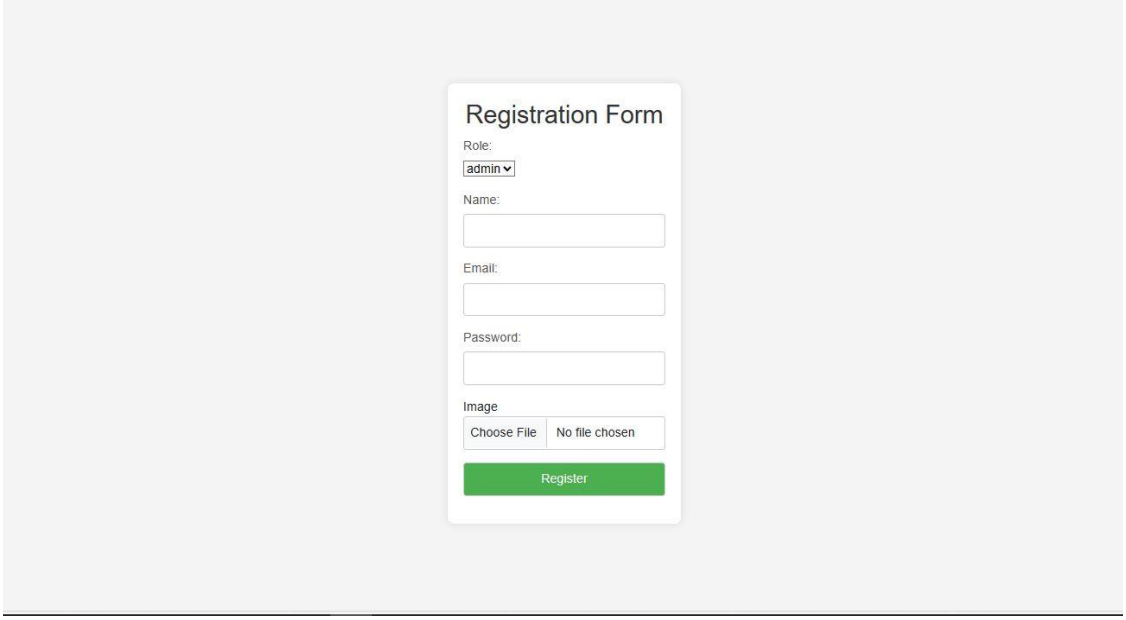
11 Admin Page

11.1 Admin Login Page



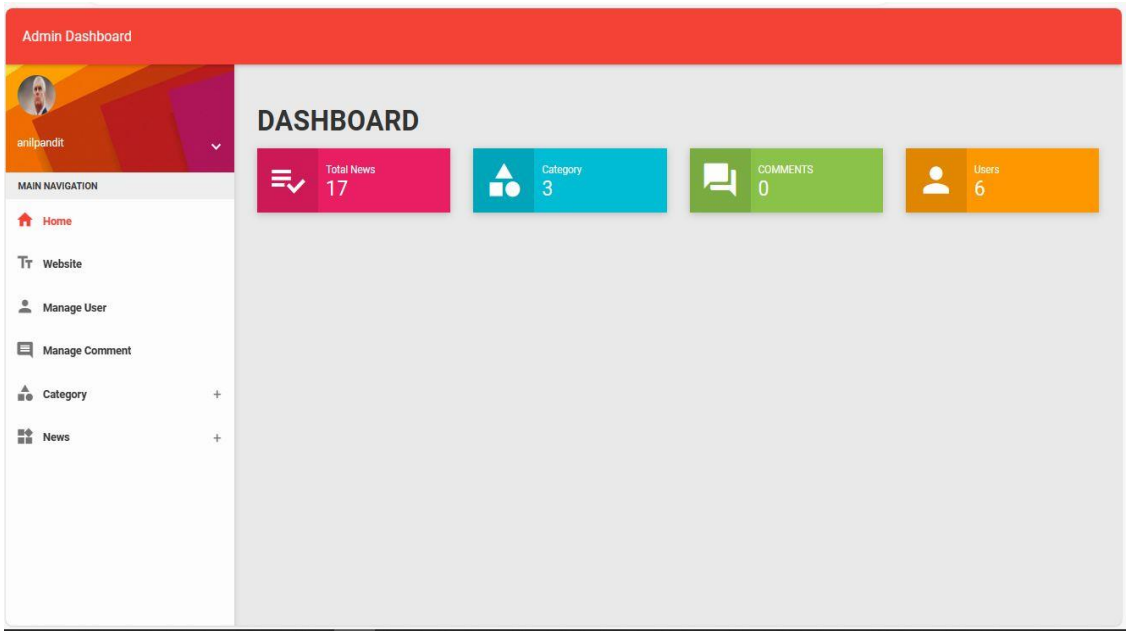
XXI. Fig: Admin Login Page

11.2 Admin Register Page



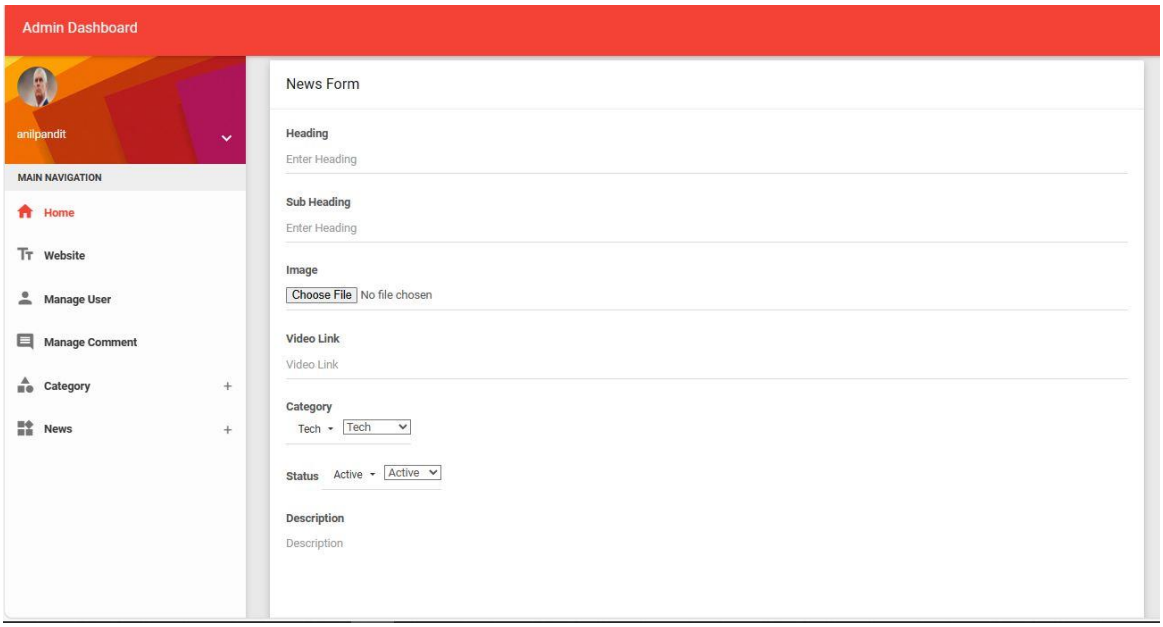
XXII. Fig: Admin Register Page

11.3 Admin Home Page



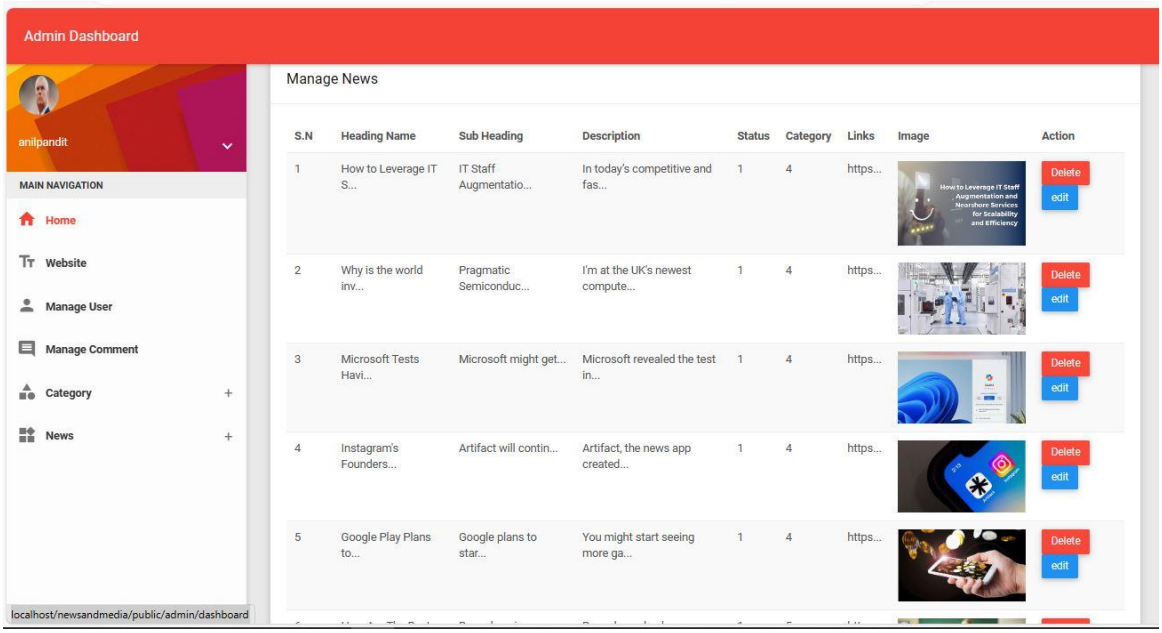
XXIII. Fig Admin Home Page

23.4 News Create Page



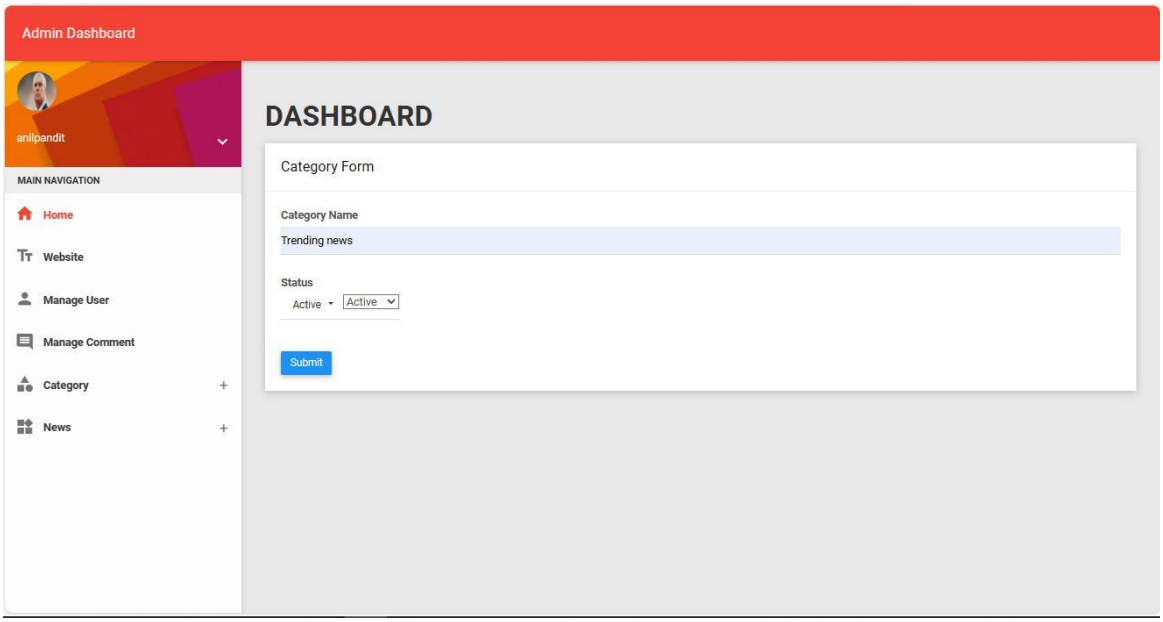
XXIV. Fig: News Create Page

11.5 News Manage Page



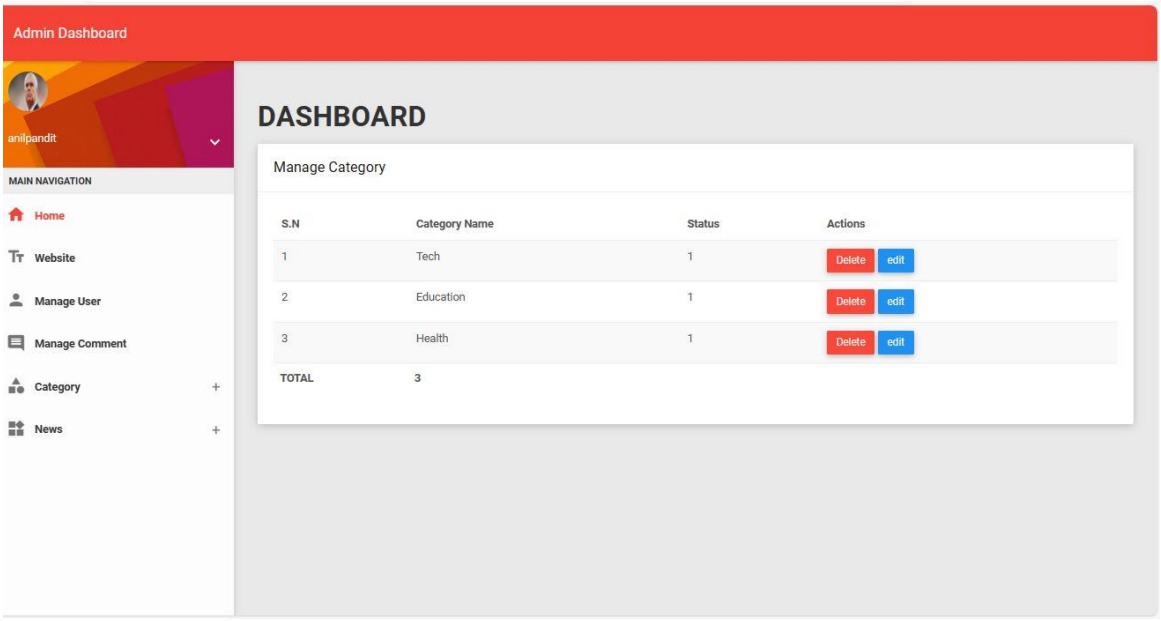
XXV. Fig: News Manage Page

11.6 Category Create Page



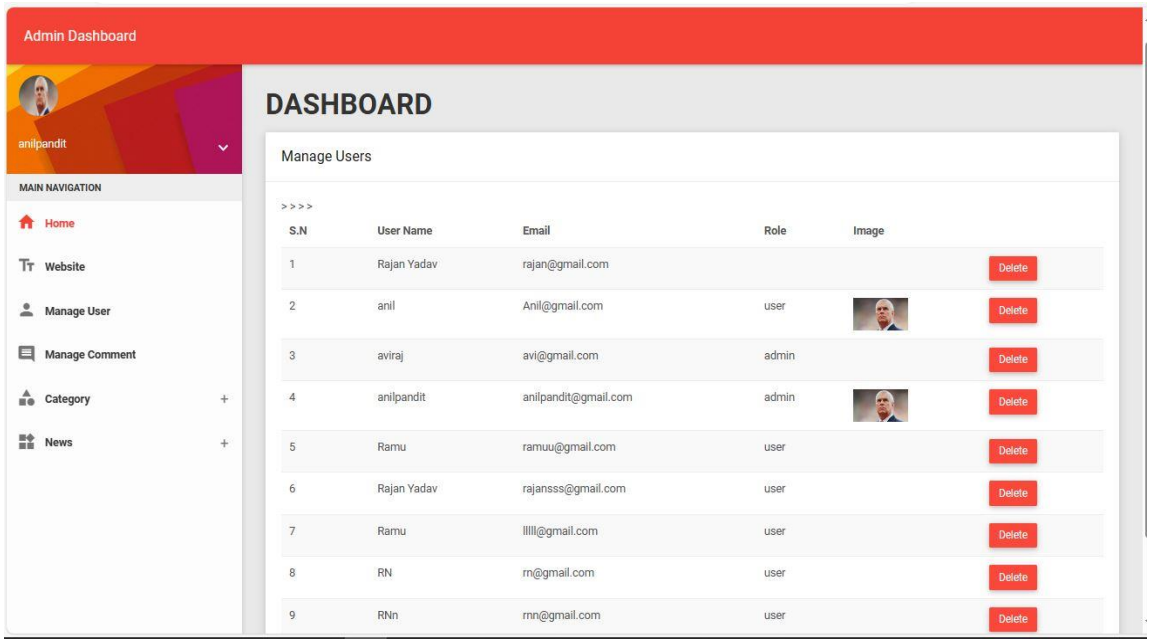
XXVI. Fig: Category Create Page

11.7 Category Manage Page



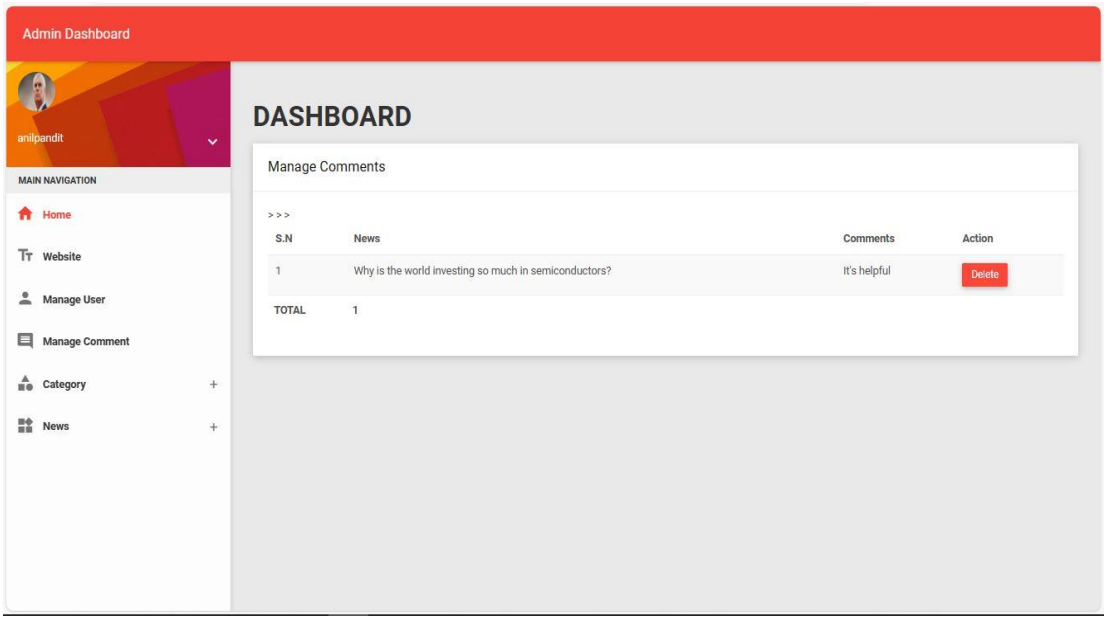
XXVII. Fig: Category Manage Page

11.8 User Manage Page



XXVIII. Fig: User Manage Page

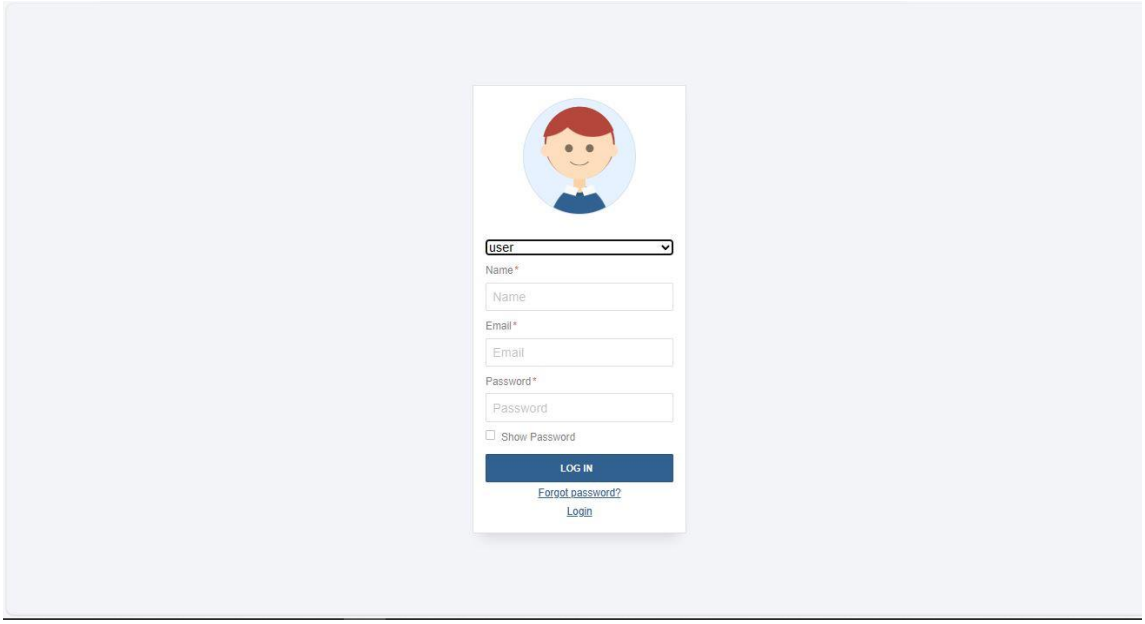
11.8 COMMENTS MANAGE PAGE



XXIX. Fig: Comments Manage Page

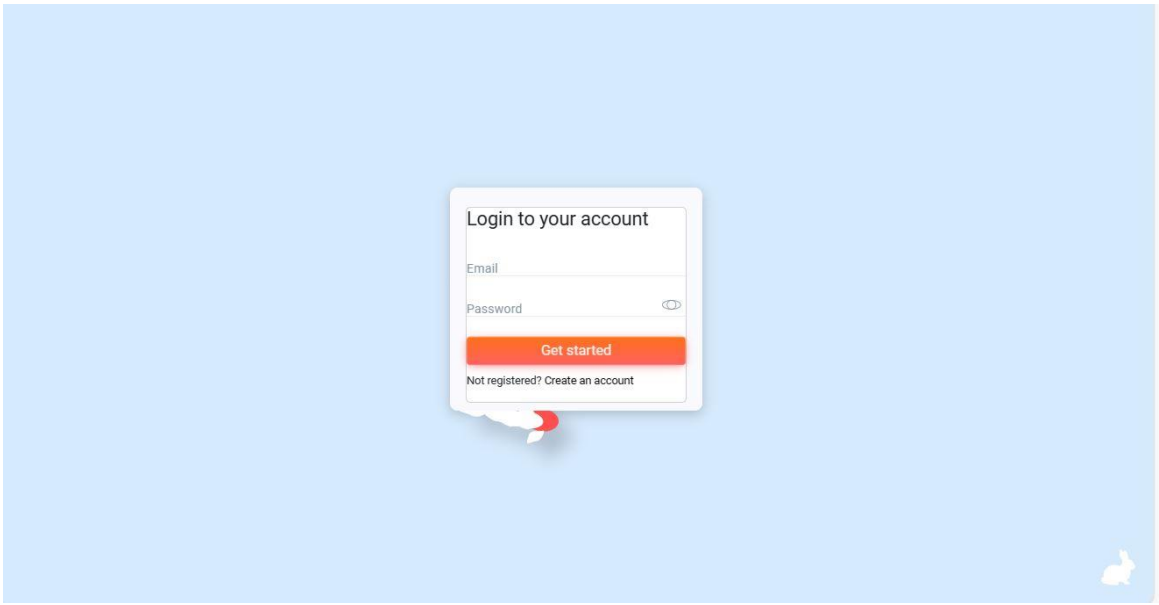
12 User Pages

12.1 Register User Page



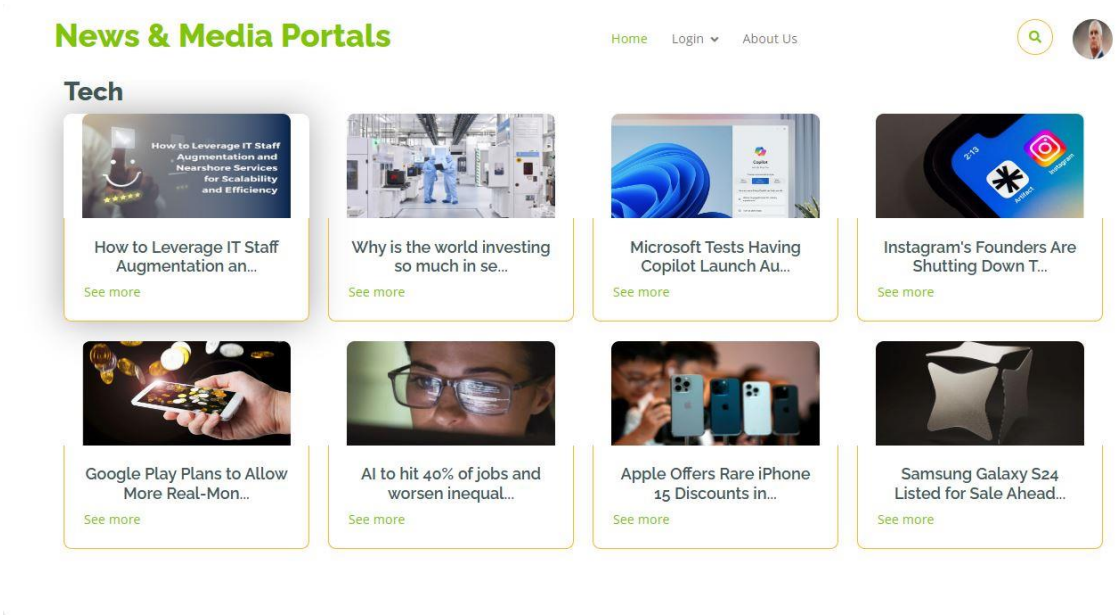
XXX. Fig: Register User Page

12.2 Login User Page



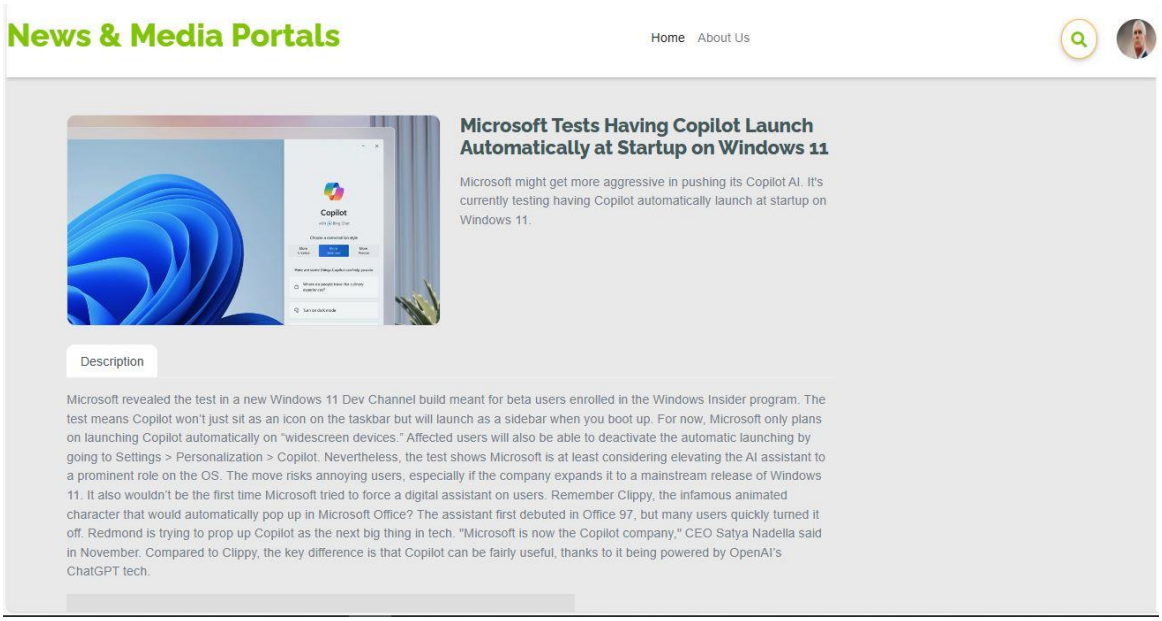
XXXI. Fig: User Login Page

12.3 User Dashboard



XXXII. Fig: User Dashboard

12.4 News Details Page



XXXIII. Fig: News Details Page

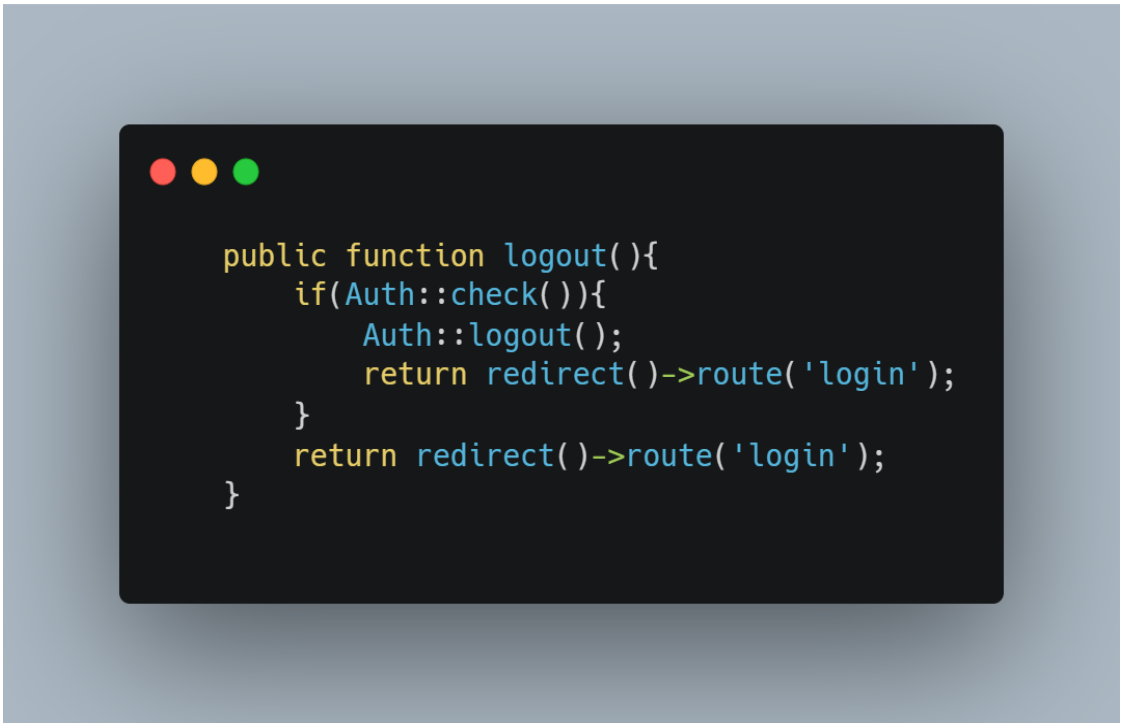
12.1 Code For Register User

```php
public function signup (Request $request){

        $rules = [
            'name' => 'required|string|max:255',
            'email' => ['required','string','email','max:255',Rule::unique('users')],
            'password' => 'required|string|min:8',
            'role' => 'required',

        ];
              // Validate the request
              $validator = Validator::make($request->all(), $rules);
              // If validation fails, return the errors
              if ($validator->fails()) {
                return redirect()->back()->withErrors($validator)->withInput();
              }
              $image_url = '';
              if($request->has('image') && $request->file('image')){
                  $file  = $request->file('image');
                  $name  = time().'.'.$file->getClientOriginalExtension();
                  $path = public_path().'/newsimg'.'/';
                  $file->move($path,$name);
                  $image_url = asset('/newsimg').'/'.$name;
              }

        $data=[
            'name'=>$request->name,
            'email'=>$request->email,
            'password'=>bcrypt($request->password),
            'role'=> $request->role,
            'profile'=> $image_url
        ];

        user::insert($data);
        return redirect()->back()->with('success', 'User registered successfully!');
       }
```

12.2 Code For Login User

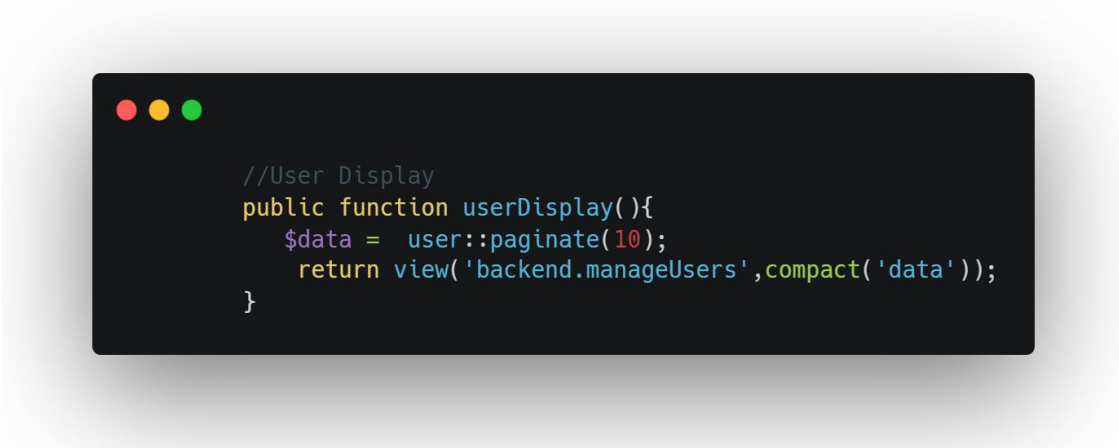
```
public function signin(Request $request){

        $rules=[
            'email' => 'required|email',
            'password' =>'required|min:6'
        ];

        // Validate the request
        $validator = Validator::make($request->all(), $rules);

        // If validation fails, return the errors
        if ($validator->fails()) {
        return redirect()->back()->withErrors($validator)->withInput();
        }

        // $name= $request->name;
        $email = $request->email;
        $password = $request->password;
        $user = user::where('email' ,$email)->first();
        if($user){

        if($user->password){
            if (Hash::check($password,$user->password)){
                    Auth::login($user);
                if($user->role =='admin'){
                return redirect()->route('dashboard');
             }
            else if ($user->role =='user'){
            return redirect()->route('userdashboard');
            }
            else{
            return redirect()->back()->with('error', 'Check Email , Password!');
             }
            return redirect()->back()->with('error', 'Check Email , Password!');
            }
        }
        }
        }
```

12.3 Code For Logout User



```
public function logout(){
    if(Auth::check()){
        Auth::logout();
        return redirect()->route('login');
    }
    return redirect()->route('login');
}
```

12.4 Code For Display Users



```
//User Display
public function userDisplay(){
   $data =  user::paginate(10);
    return view('backend.manageUsers',compact('data'));
}
```

12.5 Code For Delete User

```
//Delete User Record
        public function deleteUser($id){
            if(!$id){
                return redirect()->back();
            }

             $users= user::find($id);
             if($users){
            $users->delete();
             }
            return redirect()->back();
        }
```

12.6 Code For Create News

```php
//Data Insert
    public function newscreate(Request $request){
      $rules = [
        'heading' => 'required',
        'subheading'=>'required',
        'description'=>'required',
        'category_id'=>'required',
        'status'=>'required',
        'image'=>'required',

    ];
          // Validate the request
          $validator = Validator::make($request->all(), $rules);

          // If validation fails, return the errors
          if ($validator->fails()) {
            return redirect()->back()->withErrors($validator)->withInput();
          }

        $image_url = '';
        if($request->has('image') && $request->file('image')){
            $file  = $request->file('image');
            $name  = time().'.'.$file->getClientOriginalExtension();
            $path = public_path().'/newsimg'.'/';
            $file->move($path,$name);
            $image_url = asset('/newsimg').'/'.$name;
        }
        $data = [
            'heading'=>$request->heading,
            'subheading'=>$request->subheading,
            'description'=>$request->description,
            'status'=> $request->status,
            'image'=>$image_url,
            'link'=>$request->link,
            'category_id'=>$request->category_id
       ];

       news::insert($data);
       return redirect()->back()->with('success', 'Inserted successfully!');
    }
```

12.7 Code For Update News

```php
//Update Data
          public function updatenews(Request $request,$id)
          {
           if(!$id){
               return redirect()->back();
           }
           $cat_data= news::find($id);
          if($cat_data){
           $image_url = '';
           if($request->has('image') && $request->file('image')){
               $file  = $request->file('image');
               $name  = time().'.'.$file->getClientOriginalExtension();
               $path = public_path().'/newsimg'.'/';
               $file->move($path,$name);
               $image_url = asset('/newsimg').'/'.$name;
           }
           $data = [
               'heading'=>$request->heading,
               'subheading'=>$request->subheading,
               'description'=>$request->description,
               'status'=> $request->status,
               'image'=>$image_url,
               'link'=>$request->link,
               'category_id'=>$request->category_id
          ];

          $cat_data->update($data);
         return redirect()->route('display.news');

          }
          return redirect()->back();
        }
```

12.8 Code For Edit News

```
public function editnews($id)
         {
          if(!$id){
              return redirect()->back();
          }
          $category=category::all();
          $cat_data= news::find($id);
         if($cat_data){
          return view('backend.news.edit',compact('cat_data','category'));
         }
         return redirect()->back();
         }
```

12.9 Code For Delete News

```
    // Delete Record
    public function deletenews($id){
        if(!$id){
            return redirect()->back();
        }

       $cat_data= news::find($id);
       if($cat_data){
        $cat_data->delete();
       }
       return redirect()->back();
       }
```

12.10 Code For Display News

```php
//News Display
 public function displaynews(){
    $data= news::paginate(10);
     return view('backend.news.display',compact('data'));
 }
```

12.11 Code For Display Category

```php
//Category Display
public function displayData(){
   $data['categories']= category::where('status',1)->get();
    return view('backend.news.create',$data);
}
```

12.12 Code For Search News

```php
public function search(Request $request)
{
  $searchterm =$request->search;
  $query=news::query();
  $data['systemdata'] = system::find(1);
  $_SESSION['setting'] = $data['systemdata'];

  if($searchterm){
    $query->where('heading','LIKE','%'.$searchterm.'%')
          ->orWhere('description','LIKE','%'.$searchterm.'%')
          ->orWhere('subheading','LIKE','%'.$searchterm.'%');
          $data['newss']=$query->get();
          return view('frontend.result',$data);
  }
  else{
    return redirect()->back();
  }
}
```

12.13 Code For News Details

```php
//News Details
        public function newsdetails($id){

            if(!$id){
              return redirect()->back();
            }
            $data['news'] =  news::with('comments')->find($id);
            $data['system'] = system::find(1);
            $_SESSION['setting'] = $data['system'];
              $news = $data['news'];
            if($data['news']){
              $data['related'] = news::whereHas('category', function($query) use ($news) {

                $query->where('id','!=', $news->category->id);
              })->take(4)->get();
              return view('frontend.newsdetails', $data);
            }

            if(!$id){
              session()->flash('error','news Not Found!');
              return redirect()->back();
            }

        }
```

12.14 Code For Create Category

```php
    // Insert Data
    public function categorydata(Request $request){
        $data = [
            'name' =>$request->catename,
            'status' => $request->status,
       ];

       category::insert($data);
       return redirect()->route('admin.display');
    }
```

12.15 Code For Update Category

```php
//Update Data
       public function update(Request $request,$id)
       {
        if(!$id){
            return redirect()->back();
        }
        $cat_data= category::find($id);
       if($cat_data){
        $data=[
            'name' =>$request->catename,
            'status' => $request->status,
        ];
       $cat_data->update($data);
      return redirect()->route('admin.display');

       }
       return redirect()->back();
    }
```

12.16 Code For Delete Category

```php
    //Delete Record
    public function delete($id){
        if(!$id){
            return redirect()->back();
        }

       $cat_data= category::find($id);
       if($cat_data){
        $cat_data->delete();
       }
       return redirect()->back();
       }
```

12.17 Code For Create Comments

```php
public function commentinsert(Request $request){


 $data = [
     'category_id'=>$request->category_id,
     'user_id'=>$request->user_id,
     'comment'=>$request->comment,
     'news_id'=>$request->news_id,
];

comment::insert($data);
return redirect()->back();
}
```

12.18 Code For Display Comments

```php
//Comment Display
public function commentdisp(){
 $data['news'] =  news::all();

 $data= comment::paginate(10);
  return view('backend.manageComment',compact('data'));
}
```

12.19 Code For Delete Comments

```
//Delete Record
public function deleteComment($id){
    if(!$id){
        return redirect()->back();
    }

   $comments= comment::find($id);
   if($comments){
    $comments->delete();
   }
   return redirect()->back();
   }
```

12.20 Code For Create System Details

```php
public function systemdata (Request $request){

      $logo = '';

      if($request->has('logo') &&  $request->file('logo')){
          $setting  = system::find(1);
          if($setting && $setting->logo){
               if(file_exists(public_path("/uploads/$setting->logo"))){
                  unlink(public_path("uploads/$setting->logo"));
               }
          }

       $file  = $request->file('logo');
       $newName = time().'-'. rand(10,9999999999999).'-'.$file->getClientOriginalName();
       $newPath = public_path()."/uploads/";
       $file->move($newPath, $newName);
       $logo = $newName;
      }


      $record = system::find(1);
      if ($record) {
        // If the record exists, update it
        $record->update([
          'name'=>$request->systemname,
          'email'=>$request->email,
          'phone'=>$request->phone,
          'address'=>$request->address,
          'date'=>$request->date,
          'slogan'=>$request->slogan,
          'logo'=>$logo,
        ]);
        return redirect()->back()->with('success', 'Record updated successfully.');
    }

    else {
      system::create([
          'name'=>$request->systemname,
          'email'=>$request->email,
          'phone'=>$request->phone,
          'address'=>$request->address,
          'date'=>$request->date,
          'slogan'=>$request->slogan,
          'logo'=>$logo,
      ]);
      return redirect()->back()->with('success', 'Record updated successfully.');
    }
     }
```

13 Conclusion

The News and Media Portals project has successfully achieved its primary objectives of creating a dynamic and user-friendly platform for accessing and disseminating news. The project focused on enhancing user experience, providing reliable information, and fostering community engagement. With a modern and responsive design, the portal ensures accessibility across various devices, meeting the evolving needs of our audience.

Key features such as personalized content recommendations, real-time updates, and interactive user forums have contributed to increased user satisfaction and retention. The integration of multimedia elements, including video and interactive graphics, has enriched the overall storytelling experience.

14 REFERENCE

- www.google.com
- www.youtube.com
- www.wikipedia.com
- www.w3school.com
- www.javapoint.com
- www.tutorialpoint.com
- www.carbon.now.sh
- Previous project