Online News Portal with Sentiment Analysis Project Report

By
Sakshi Aurade – AF0481812
Rida Jamadar – AF0484778

Index

Sr.no	Topic	Page no
1	Title of Project	1
2	Acknowledgement	3
3	Abstract	4
4	Introduction	5
5	Objective	6
6	System Analysis	7
7	System Design	22
8	Screenshots	32
9	Coding	39
10	Testing	46
11	Report	47
12	Future Scope	48
13	Conclusion	48
14	Bibliography	49
15	References	49

Acknowledgement

The project **"Online News Portal with Sentiment Analysis"** is the Project work carried out by

Name	Enrollment No
Sakshi Aurade	**AF0481812**
Rida jamadar	**AF0484778**

Under the Guidance.

We are thankful to my project guide for guiding me to complete the Project.

His suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

Abstract

The project called **Online News Portal** is about creating a website to share news quickly, safely, and in an organized way. This system replaces the old manual way of publishing news with a digital platform where administrators can easily add, edit, delete, and manage news articles. It also stores news for a long time, making it easier and faster to handle.

This portal lets users browse news by category, read detailed articles with publication details, and share their thoughts through comments without needing to log in. It also includes a feature to analyze comments, showing whether they are positive, neutral, or negative. This helps administrators understand public feedback better.

The system is simple, easy to use, and works well on mobile devices. It allows real-time updates, comment moderation, and better analysis. It helps organizations save time and resources while reaching more people. This platform focuses on engagement, accessibility, and making decisions based on audience feedback.

1. Introduction

In today's age of Information Communication and Technology, we are surrounded by technology at every moment. From the moment we wake up to the time we go to bed, technology plays a crucial role in our lives. One of the most important aspects of this technological revolution is the development of web applications. These applications enable us to access information from anywhere, anytime, and at a low cost. Information has become one of the most valuable resources in the modern world, and web applications, such as online news portals, provide us with the most up-to-date information.

Our project aims to develop an Online News Portal website that makes the latest news and information accessible to people at any time. The portal will allow users to stay informed about breaking news and other important updates in a timely and dynamic manner. Additionally, sentiment analysis functionality is integrated into the portal, enabling the identification of user feedback trends, such as positive, neutral, or negative sentiments, based on their comments. This adds depth to user interaction and enhances administrative insights for content management.

1.1 Objective of the Present Work

The objectives of this project are as follows:

- To develop a web application for an Online News Portal website that can keep people informed.
- To provide daily news updates.
- To provide breaking news updates in real-time.
- To deliver news in an easy-to-understand, visually appealing, and engaging way.
- To ensure wide coverage and faster dissemination of information through the portal.
- To make it possible for anyone, anywhere, and anytime to access news at a low cost via the internet.
- To create a dynamic platform that allows adding new information easily without any complexity.

System analysis

3.1 PROBLEM DEFINITION

Many news websites do not have an easy way to manage content and understand audience feedback. Users can comment on news articles, but it is hard to know the overall reaction to a story. Also, traditional methods of publishing news take a lot of time and effort.

This project solves these problems by **allowing admins to manage news easily** and **using sentiment analysis to understand reader feedback**. The system makes news publishing faster, helps admins organize content better, and gives users a platform to read and discuss news efficiently.

3.2 Preliminary Investigation

Purpose

The Online News Portal is designed to make news accessible anytime and anywhere through a digital platform. It aims to replace traditional news publishing methods by allowing administrators to manage news articles efficiently while providing users with real-time updates and interactive features like commenting and sentiment analysis.

Benefits

The portal provides several advantages:

- **Instant News Access**: Users can browse news articles categorized by topics such as politics, business, sports, and entertainment.
- **Efficient Content Management**: Admins can easily add, edit, delete, and restore news articles without delays.
- **User Engagement**: Readers can interact by posting comments without login
- **Improved Moderation**: Sentiment analysis helps admins monitor user feedback and moderate discussions effectively.

Proposed System

The proposed system offers a structured, reliable, and automated approach to news publishing:

- Users can browse articles, comment on news stories, and receive instant updates.

- Administrators can moderate comments, manage news categories, and analyze user sentiment.
- Sentiment analysis provides insights into reader reactions, making content management more effective.
- The portal is designed to be simple, responsive, and efficient, ensuring a smooth experience for both users and administrators.

3.3 Feasibility Study

The feasibility study evaluates whether the **Online News Portal with Sentiment Analysis** project is practical, achievable, and beneficial. This assessment ensures the system can be successfully implemented within available resources, technology, and constraints.

Types of Feasibility Analysis

- **Technical Feasibility**
  - ✓ The portal is built using **Python (Django) for backend** and **HTML, CSS, Bootstrap for frontend**, making it compatible with existing web technologies.
  - ✓ The database is managed using **MySQL**, ensuring secure and scalable data storage.
  - ✓ The system is hosted on a **reliable web server**, supporting high user traffic.
  - ✓ Sentiment analysis is integrated using **TextBlob**, a lightweight NLP tool that ensures fast processing of user feedback.
- **Economic Feasibility**
  - ✓ Open-source technologies like Django and MySQL help minimize software costs.
  - ✓ The automated sentiment analysis feature saves administrative effort in assessing user feedback trends.

- **Operational Feasibility**

  - ✓ Users can easily access the portal through any device, ensuring high usability.
  - ✓ News articles are categorized, making navigation simple.
  - ✓ Admins can efficiently moderate comments, ensuring a safe and engaging space for readers.

- **Schedule Feasibility**

  - ✓ The development timeline is realistic, covering design, coding, testing, and deployment within estimated deadlines.

- ✓ Modular development ensures different parts of the system are built incrementally, improving flexibility.

- **Social Feasibility**

  - ✓ The project promotes informed discussions among users by allowing sentiment analysis on comments.
  - ✓ It fosters engagement without misinformation or toxic debates.
  - ✓ The system is accessible to all users, regardless of technical expertise.

3.4 Project Planning

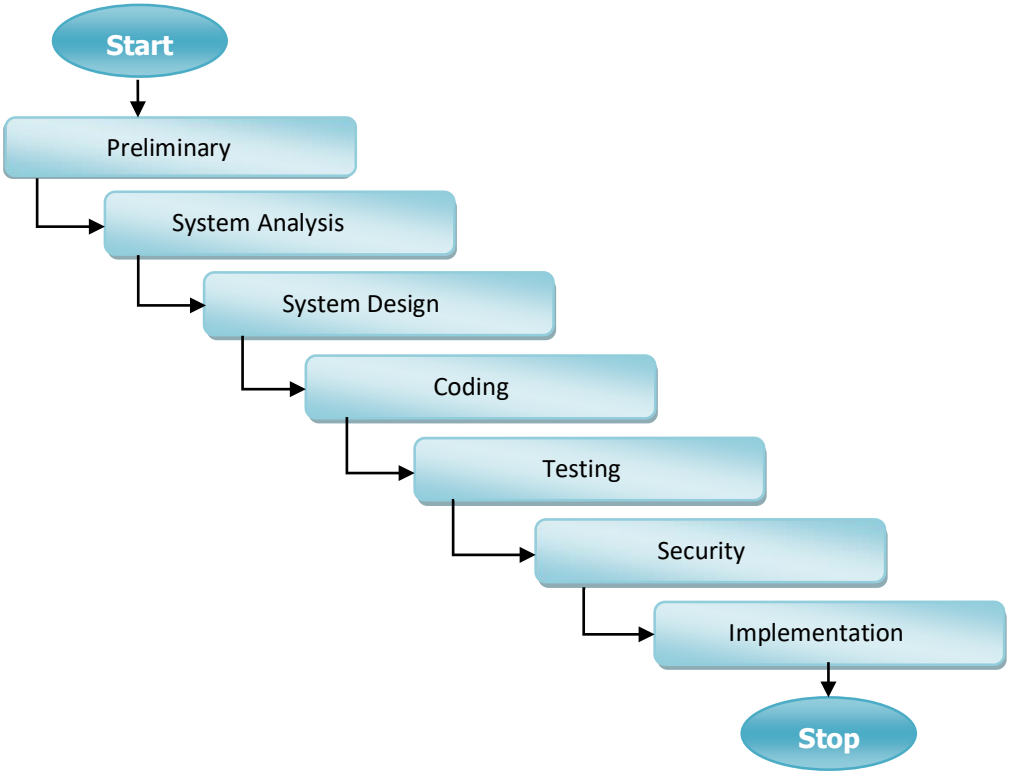
Purpose of Project Planning

Project planning ensures that the development of the **Online News Portal with Sentiment Analysis** follows a structured approach. It helps define the workflow, resource allocation, timelines, and risks involved to deliver the system efficiently.

Phases Covered in the Plan

The planning is divided into different phases to ensure smooth execution:

1. **Preliminary Investigation** – Understanding the project scope and objectives.
2. **System Analysis** – Identifying challenges, gathering requirements, and defining solutions.
3. **System Design** – Structuring modules, database design, and UI development.
4. **Coding** – Developing the portal using Python (Django) and integrating sentiment analysis.
5. **Security** – Implementing authentication, data encryption, and user privacy measures.
6. **Testing** – Performing unit testing, integration testing, and user acceptance testing.
7. **Implementation** – Deploying the final system and ensuring smooth operation.



3.5 Software Requirement Specification (SRS)

The **Software Requirement Specification (SRS)** outlines the fundamental requirements of the **Online News Portal with Sentiment Analysis** to ensure efficient functionality, usability, and maintainability.

System Overview

The Online News Portal allows users to access and interact with news articles across multiple categories. The system is structured into **three modules**:

1. **User Module** – Enables users to browse news, search for articles, and post comments.
2. **Admin Module** – Provides tools for news management, user moderation, and sentiment analysis.
3. **Sub-Admin Module** – Allows sub-admins to manage articles with limited access rights.

Software & Hardware Requirements

Software Requirements

- **Frontend**: HTML, CSS, Bootstrap for responsive UI
- **Backend**: Python (Django) for data handling and security
- **Database**: MySQL for storing articles, comments, and user details
- **Sentiment Analysis**: TextBlob for classifying user comments
- **Web Server**: Apache or Nginx for hosting

Hardware Requirements

- **Processor**: Intel i5 or higher
- **RAM**: Minimum 8GB
- **Storage**: At least 100GB for database and media files
- **Connectivity**: Internet access for real-time updates

3.6 Functional Requirements

1. User Module

Users can:

- Read news articles categorized into Politics, Business, Technology, Sports, Entertainment, etc.
- Search for specific articles using keywords.
- Post comments on news stories without requiring a login.

2. Admin Module

Admins have full control over the platform and can:

- **Secure Login System** – Access the admin dashboard through an authentication system.
- **Dashboard Management** – Control news categories, sub-categories, and published articles.
- **User Moderation** – Approve, unapprove, or delete comments for content control.
- **Sub-Admin Management** – Create and manage sub-admin roles with specific permissions.
- **Sentiment Analysis** – Analyze user comments using **TextBlob** to understand audience reactions.

- **Content Editing** – Add, edit, delete, and restore news posts.
- **Page Management** – Manage static pages like 'About Us' and 'Contact Us'.

3. Sub-Admin Module

Sub-Admins can:

- Perform **news management** (add, edit, delete) but cannot create new sub-admins.
- Access sentiment insights for the articles they manage.
- Moderate comments within assigned categories.

3.7 Software Engineering Paradigm

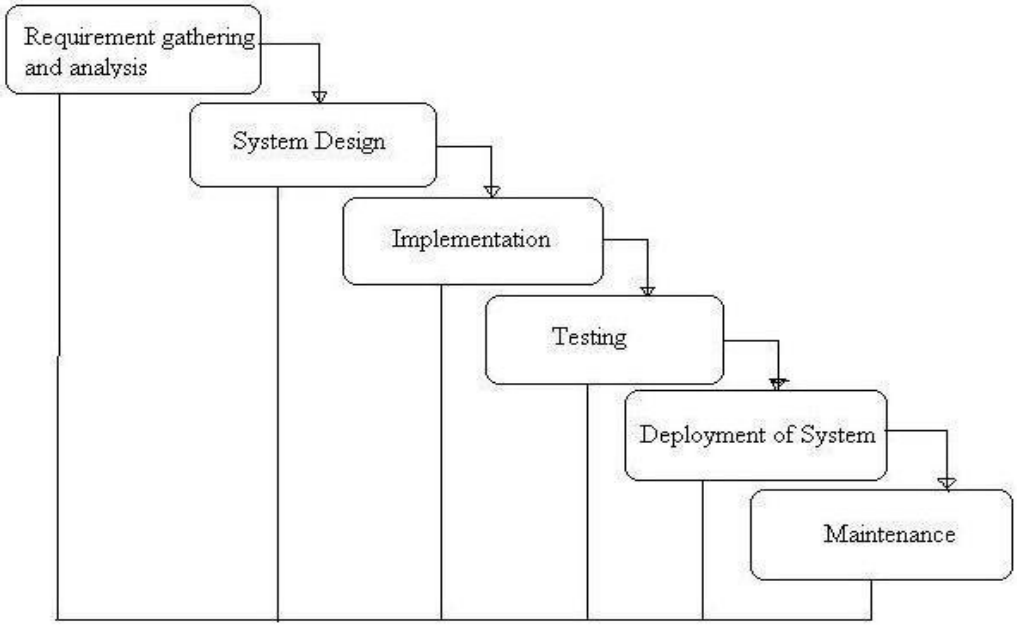
The development of the **Online News Portal with Sentiment Analysis** follows a structured approach to ensure efficiency, reliability, and maintainability. The chosen paradigm helps streamline the project by defining clear phases while allowing iterative improvements.

Development Model: Adapted Waterfall Model

The **Waterfall Model** is traditionally a linear approach, but for this project, an **iterative feedback mechanism** is incorporated. This helps refine earlier phases based on insights gathered during implementation.

Key Adaptations in the Waterfall Model:

1. **Structured Phase Progression** – Each stage follows a defined sequence, ensuring clarity in execution.
2. **Iterative Refinements** – Feedback loops allow adjustments, especially between testing and coding.
3. **Defined Milestones** – Each stage reaches completion before moving to the next phase.
4. **Flexible Adjustments** – Overlapping is permitted when necessary to enhance efficiency.



Phases of Development

1. **Requirement Analysis & System Study**
   - Identifying project goals, challenges, and functional specifications.
   - Gathering stakeholder requirements and defining core functionalities.
2. **System Design**
   - Structuring the **database, modules, and architecture**.
   - Designing **user interfaces** for optimal accessibility.
3. **Implementation (Coding)**
   - Backend development using **Python (Django)**.
   - Frontend design using **HTML, CSS, Bootstrap**.
   - Database integration with **MySQL**.
   - Sentiment analysis integration with **TextBlob**.
4. **Testing & Debugging**
   - Unit testing, integration testing, and usability checks.
   - Debugging for performance improvements.
5. **Deployment & Maintenance**
   - Hosting on a scalable environment.
   - Continuous updates for feature enhancements.

3.8 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

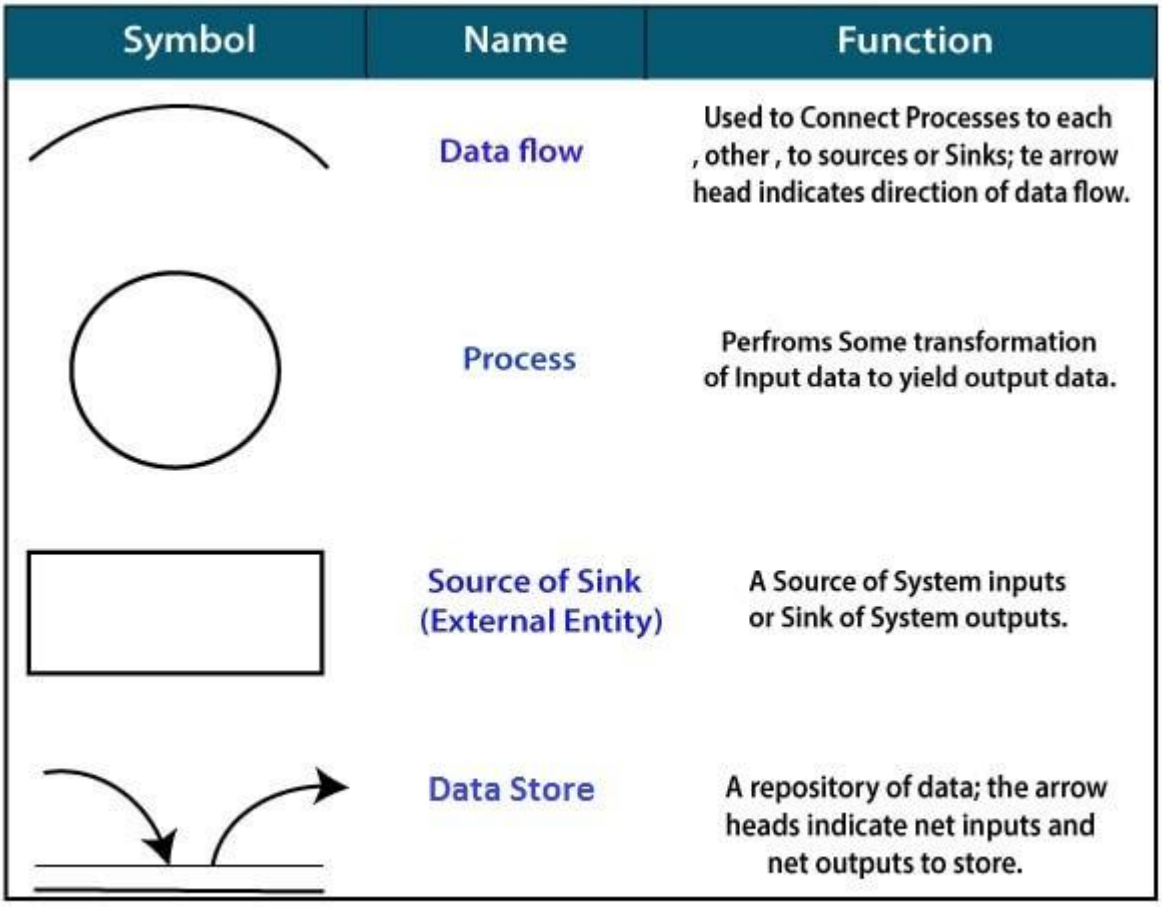
It shows how data enter and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:



Symbols for Data Flow Diagrams

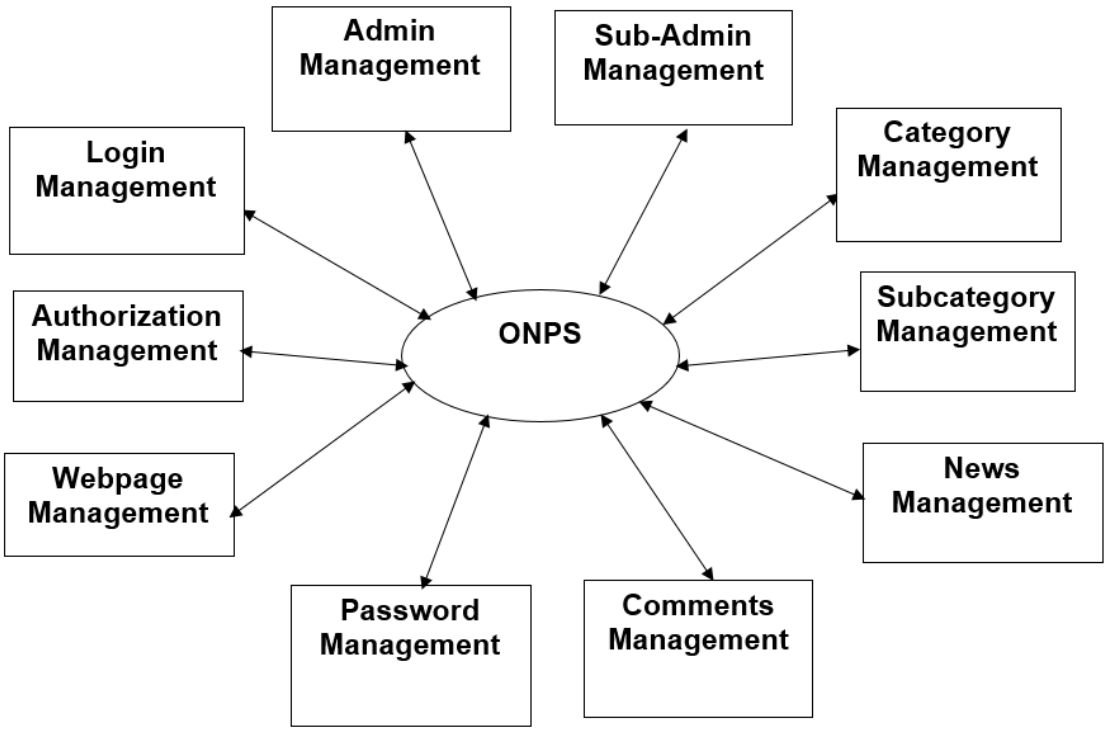
Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

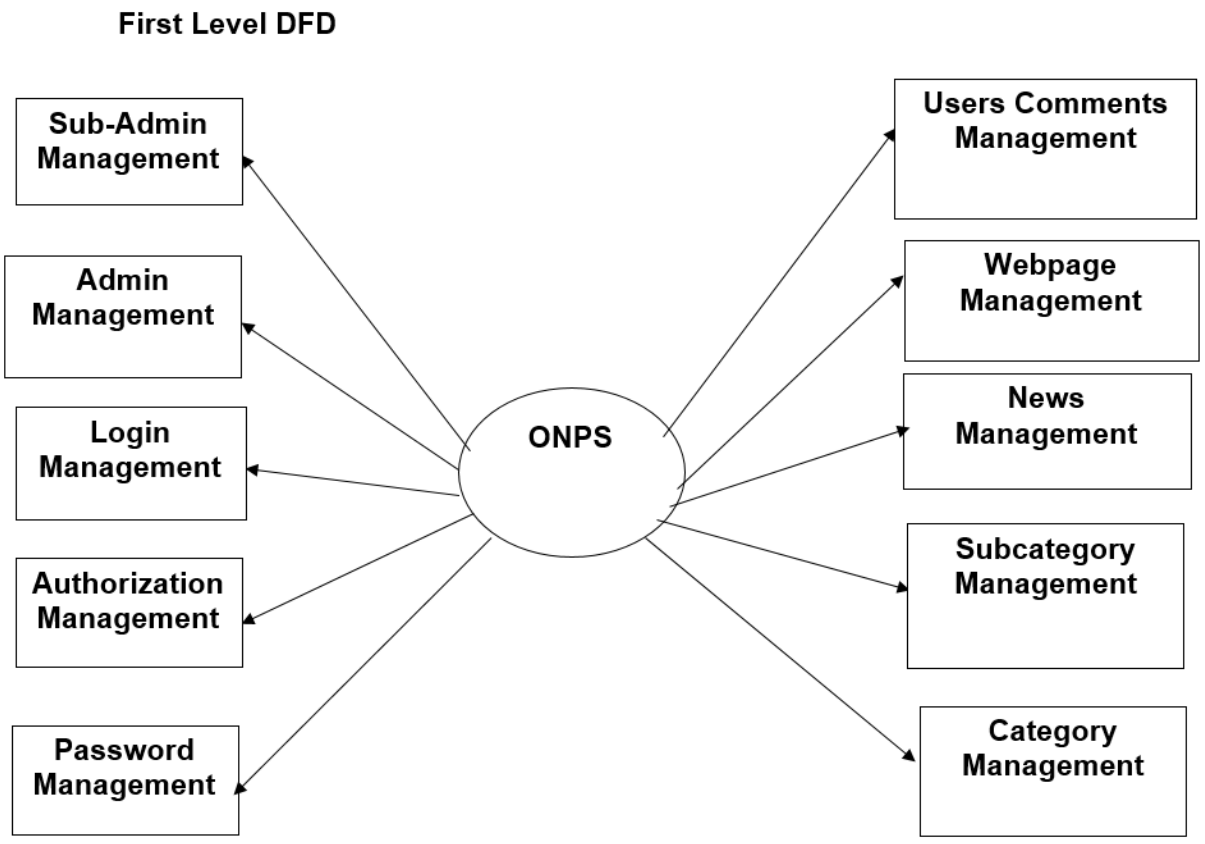
Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

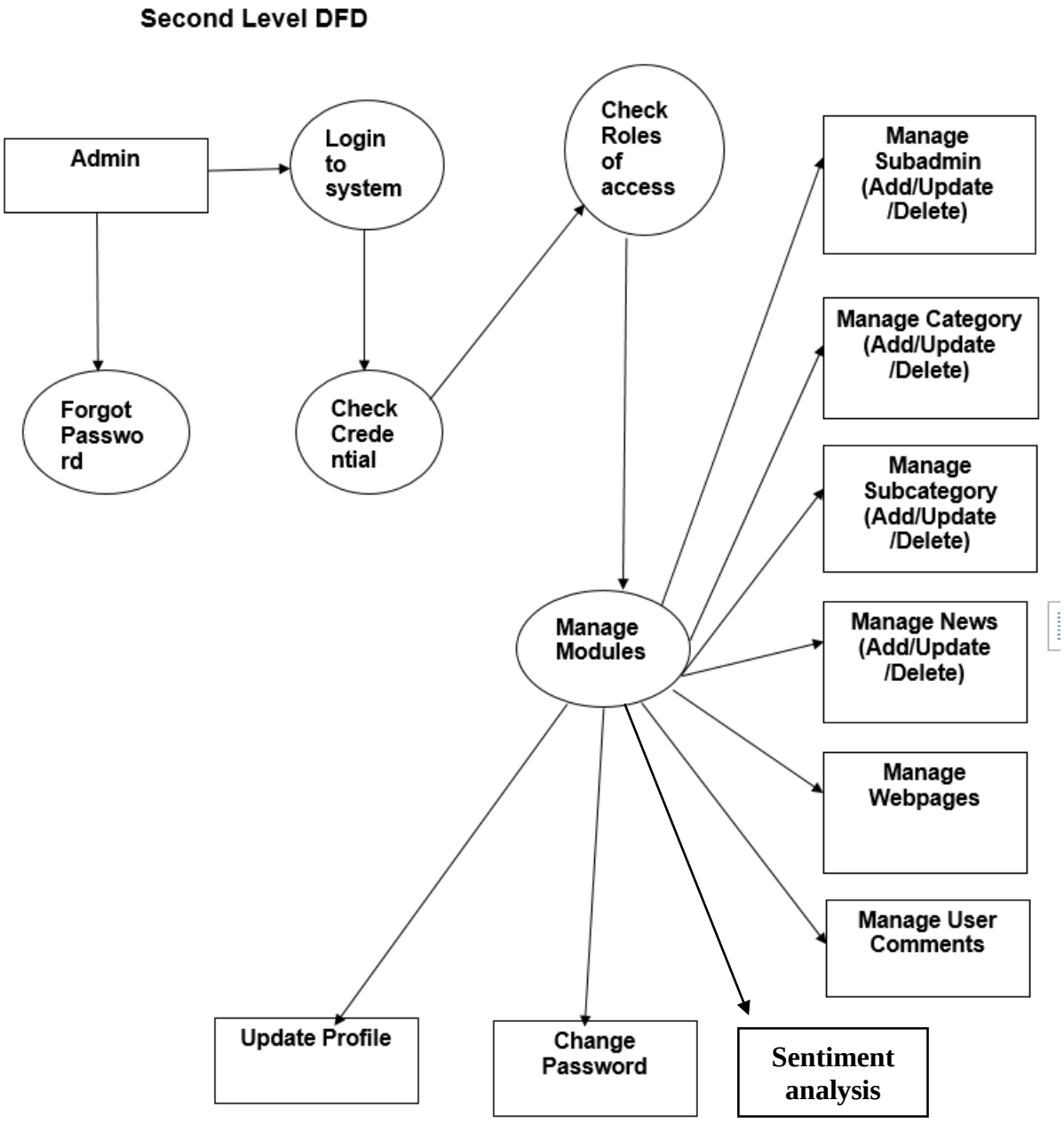
Zero Level DFD



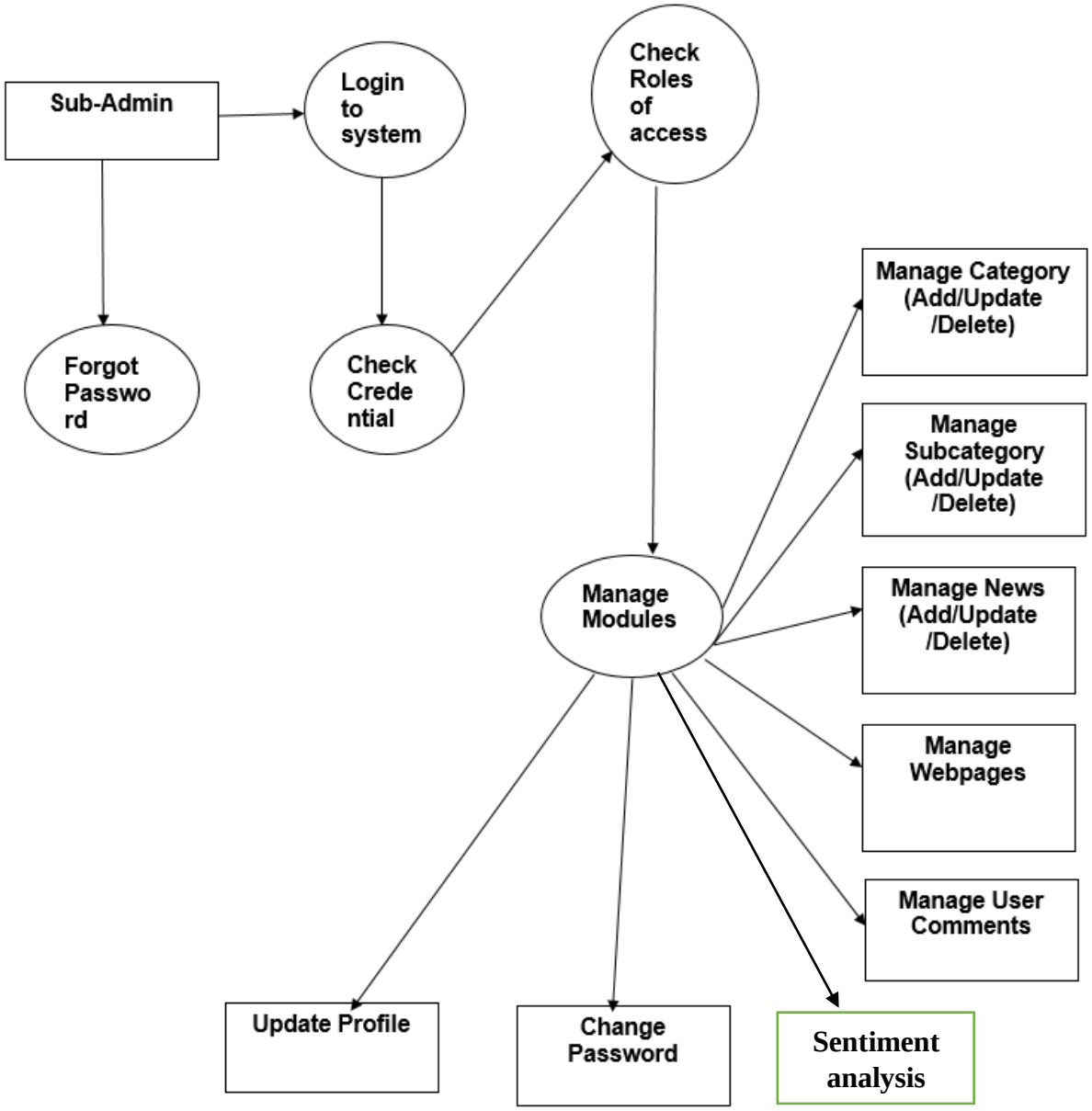
The **Zero Level DFD** of the Online News Portal System (ONPS) shows the main processes interacting with the system, such as Admin, Sub-Admin, Category, Subcategory, News, Comments, Login, Password, Authorization, and Webpage Management. It highlights the flow of data between the central system (ONPS) and these modules, representing how each core function is managed within the portal.



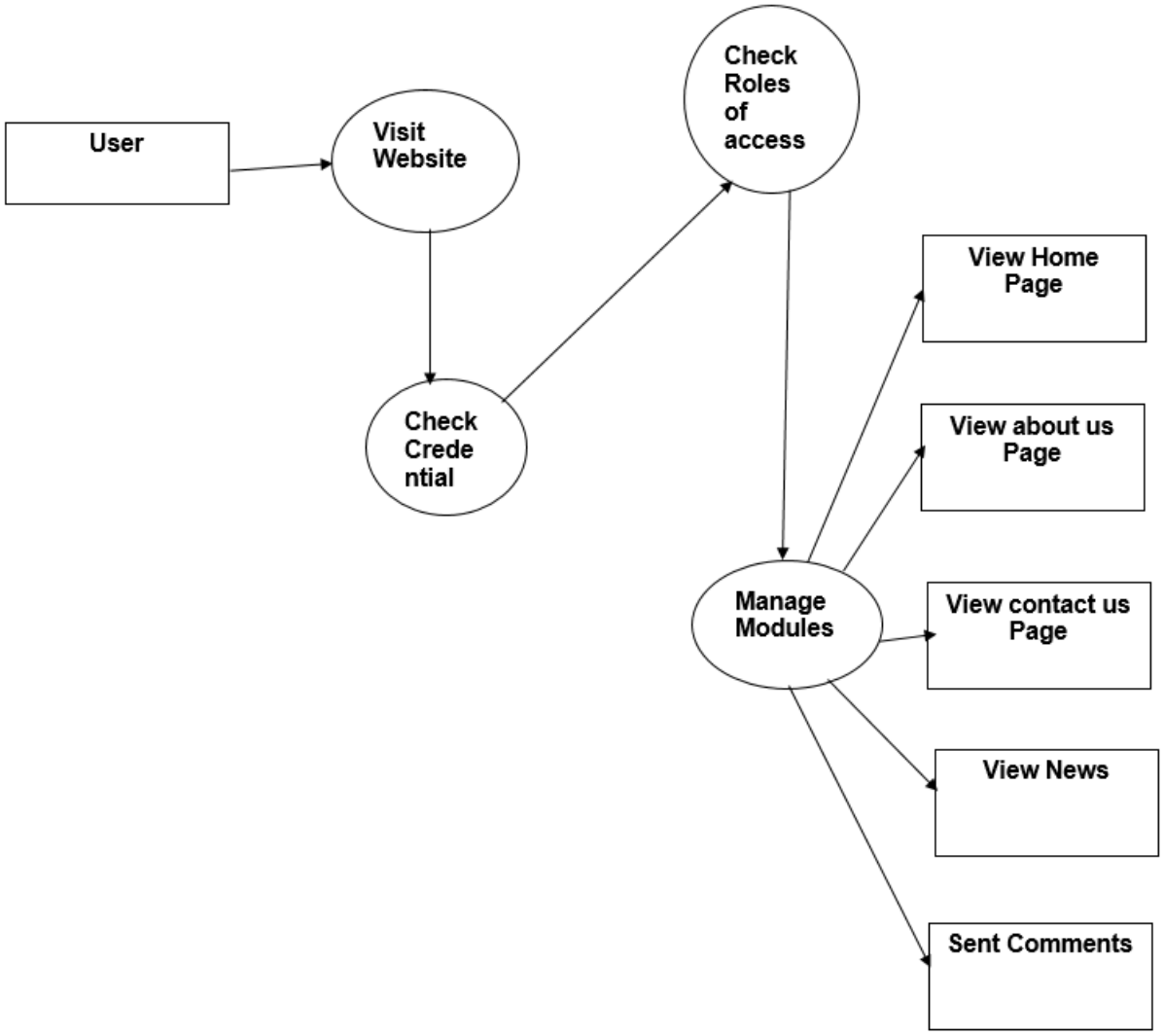
The **First-Level DFD** of ONPS shows how the system handles key functions like login, admin, sub-admin, authorization, and password management, and connects them to content-related modules such as news, category, subcategory, webpage, and user comments management for smooth portal operations.



The **Second-Level DFD** shows the internal working of the admin in the ONPS system. It includes processes like login, credential checking, role-based access, and module management. The admin can manage subadmins, categories, subcategories, news, webpages, and user comments, along with profile updates, password changes, and performing sentiment analysis.



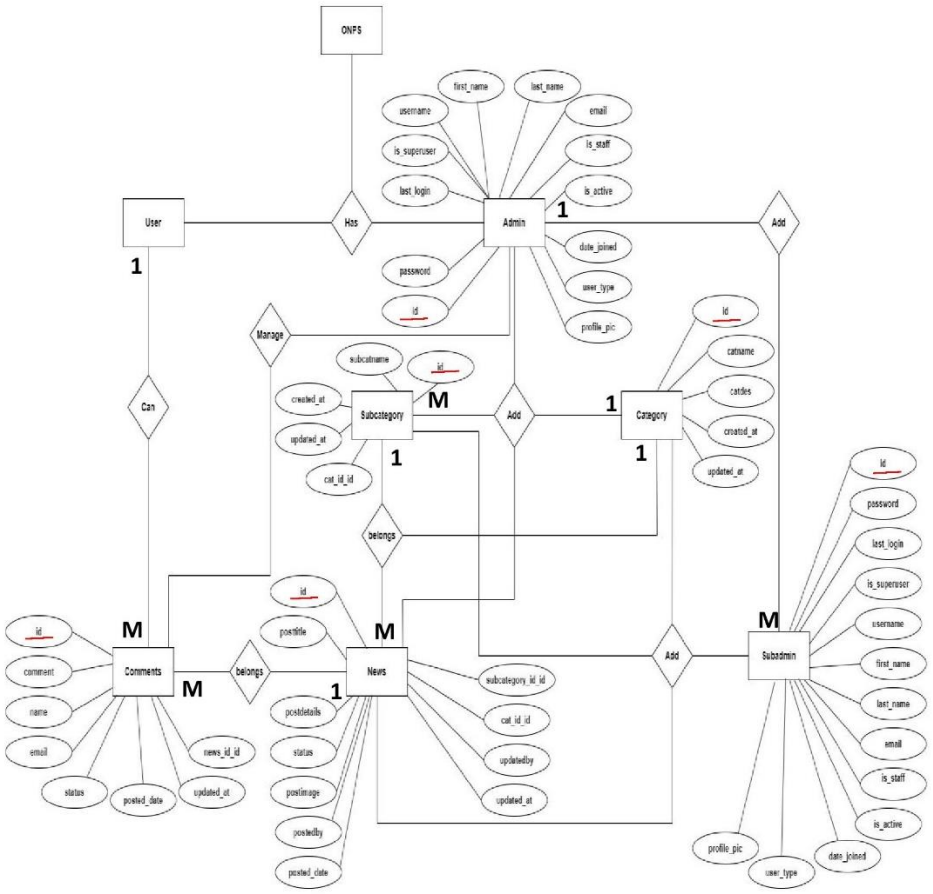
This **Second-Level DFD** represents the internal workflow for the Sub-Admin in the ONPS system. After logging in and verifying credentials, the system checks the Sub-Admin's access role and allows them to manage specific modules such as categories, subcategories, news, webpages, and user comments. Additionally, the Sub-Admin can update their profile, change their password, and perform sentiment analysis.



This **Second-Level DFD** illustrates the **User's interaction** with the ONPS system. When a user visits the website users can interact with various modules.

ER diagram

The **ER Diagram** of the **Online News Portal System** shows key relationships between **Users, Admins, News Articles, Categories, Comments, and Sentiment Analysis**, ensuring structured data management and insightful moderation.



4. System design

4.1. Module

Users can:

- Read news articles categorized into Politics, Business, Technology, Sports, Entertainment, etc.
- Search for specific articles using keywords.
- Post comments on news stories without requiring a login.

2. Admin Module

Admins have full control over the platform and can:

- **Secure Login System** – Access the admin dashboard through an authentication system.
- **Dashboard Management** – Control news categories, sub-categories, and published articles.
- **User Moderation** – Approve, unapprove, or delete comments for content control.
- **Sub-Admin Management** – Create and manage sub-admin roles with specific permissions.
- **Sentiment Analysis** – Analyze user comments using **TextBlob** to understand audience reactions.
- **Content Editing** – Add, edit, delete, and restore news posts.
- **Page Management** – Manage static pages like 'About Us' and 'Contact Us'.

3. Sub-Admin Module

Sub-Admins can:

- Perform **news management** (add, edit, delete) but cannot create new sub-admins.
- Access sentiment insights for the articles they manage.
- Moderate comments within assigned categories.

4.2 DATA STRUCTURE OF ALL MODULES:

We have organized one database **news portal** for system design. It can be accessed directly or sequentially by registered. The database determines files, record, fields, and characters. It can be easily controlled and updated. **"Online News Portal System" contains 15 MySQL tables(In this MySQL 6 table is customized and 9 table made by default in django) :**

Customized Tables Details

Admin and Sub-Admin Table:(newsapp_customuser)
This store admin personal and login details.

```
mysql> DESCRIBE newsapp_customuser;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| password     | varchar(128) | NO   |     | NULL    |                |
| last_login   | datetime(6)  | YES  |     | NULL    |                |
| is_superuser | tinyint(1)   | NO   |     | NULL    |                |
| username     | varchar(150) | NO   | UNI | NULL    |                |
| first_name   | varchar(150) | NO   |     | NULL    |                |
| last_name    | varchar(150) | NO   |     | NULL    |                |
| email        | varchar(254) | NO   |     | NULL    |                |
| is_staff     | tinyint(1)   | NO   |     | NULL    |                |
| is_active    | tinyint(1)   | NO   |     | NULL    |                |
| date_joined  | datetime(6)  | NO   |     | NULL    |                |
| user_type    | varchar(50)  | NO   |     | NULL    |                |
| profile_pic  | varchar(100) | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

Category Table: Table name(newsapp_category)
This table store the details of category of news.

```
mysql> DESCRIBE newsapp_category;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint       | NO   | PRI | NULL    | auto_increment |
| catname    | varchar(200) | NO   |     | NULL    |                |
| catdes     | longtext     | NO   |     | NULL    |                |
| created_at | datetime(6)  | NO   |     | NULL    |                |
| updated_at | datetime(6)  | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
5 rows in set (0.04 sec)
```

Subcategory Table: Table name(newsapp_subcategory)

This table store sub-category of news.

```
mysql> DESCRIBE newsapp_subcategory;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint       | NO   | PRI | NULL    | auto_increment |
| subcatname | varchar(200) | NO   |     | NULL    |                |
| created_at | datetime(6)  | NO   |     | NULL    |                |
| updated_at | datetime(6)  | NO   |     | NULL    |                |
| cat_id_id  | bigint       | NO   | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Comments Table: Table name(newsapp_comments)

This table store the comments details which is given by news reader.

```
mysql> DESCRIBE newsapp_comments;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | bigint       | NO   | PRI | NULL    | auto_increment |
| comment     | longtext     | NO   |     | NULL    |                |
| name        | varchar(250) | NO   |     | NULL    |                |
| email       | varchar(250) | NO   |     | NULL    |                |
| status      | varchar(250) | NO   |     | NULL    |                |
| posted_date | datetime(6)  | NO   |     | NULL    |                |
| updated_at  | datetime(6)  | NO   |     | NULL    |                |
| news_id_id  | bigint       | NO   | MUL | NULL    |                |
| sentiment   | varchar(20)  | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Website Setting Table: Table name(newsapp_page)

This table store the website details which is manage by admin.

```
mysql> DESCRIBE newsapp_page;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint       | NO   | PRI | NULL    | auto_increment |
| pagetitle    | varchar(250) | NO   |     | NULL    |                |
| address      | varchar(250) | NO   |     | NULL    |                |
| aboutus      | longtext     | NO   |     | NULL    |                |
| email        | varchar(200) | NO   |     | NULL    |                |
| mobilenumber | int          | NO   |     | NULL    |                |
| created_at   | datetime(6)  | NO   |     | NULL    |                |
| updated_at   | datetime(6)  | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

News Table: Table name(newsapp_news)

This table store art products details.

```
mysql> DESCRIBE newsapp_news;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | bigint       | NO   | PRI | NULL    | auto_increment |
| posttitle         | longtext     | NO   |     | NULL    |                |
| postdetails       | longtext     | NO   |     | NULL    |                |
| status            | varchar(50)  | NO   |     | NULL    |                |
| postimage         | varchar(100) | NO   |     | NULL    |                |
| postedby          | varchar(50)  | NO   |     | NULL    |                |
| posted_date       | datetime(6)  | NO   |     | NULL    |                |
| updated_at        | datetime(6)  | NO   |     | NULL    |                |
| updatedby         | varchar(50)  | NO   |     | NULL    |                |
| cat_id_id         | bigint       | NO   | MUL | NULL    |                |
| subcategory_id_id | bigint       | NO   | MUL | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

Default Tables Details

auth_group table Structure : This table in Django is part of the Django authentication system and is used to represent groups of users.

```
mysql> DESCRIBE auth_group;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(150) | NO   | UNI | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

auth_group_permissions table Structure : This table in Django is a part of the permission system and is used to manage the relationship between user groups and permissions.

```
ysql> DESCRIBE auth_group_permissions;
+---------------+--------+------+-----+---------+----------------+
| Field         | Type   | Null | Key | Default | Extra          |
+---------------+--------+------+-----+---------+----------------+
| id            | bigint | NO   | PRI | NULL    | auto_increment |
| group_id      | int    | NO   | MUL | NULL    |                |
| permission_id | int    | NO   | MUL | NULL    |                |
+---------------+--------+------+-----+---------+----------------+
rows in set (0.01 sec)
```

auth_permission table Structure : This table in Django is used to store information about the permissions defined.

```
mysql> DESCRIBE auth_permission;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| id              | int          | NO   | PRI | NULL    | auto_increment |
| name            | varchar(255) | NO   |     | NULL    |                |
| content_type_id | int          | NO   | MUL | NULL    |                |
| codename        | varchar(100) | NO   |     | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

django_admin_log table Structure : This table is used to store records of actions taken by administrators or users through the Django admin interface.

```
mysql> DESCRIBE django_admin_log;
+-----------------+-------------------+------+-----+---------+----------------+
| Field           | Type              | Null | Key | Default | Extra          |
+-----------------+-------------------+------+-----+---------+----------------+
| id              | int               | NO   | PRI | NULL    | auto_increment |
| action_time     | datetime(6)       | NO   |     | NULL    |                |
| object_id       | longtext          | YES  |     | NULL    |                |
| object_repr     | varchar(200)      | NO   |     | NULL    |                |
| action_flag     | smallint unsigned | NO   |     | NULL    |                |
| change_message  | longtext          | NO   |     | NULL    |                |
| content_type_id | int               | YES  | MUL | NULL    |                |
| user_id         | bigint            | NO   | MUL | NULL    |                |
+-----------------+-------------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

<u>django_content_type table Structure</u> : This table in Django is a system table that is used to store information about each model (database table) in your Django project.

```
mysql> DESCRIBE django_content_type;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int          | NO   | PRI | NULL    | auto_increment |
| app_label | varchar(100) | NO   | MUL | NULL    |                |
| model     | varchar(100) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

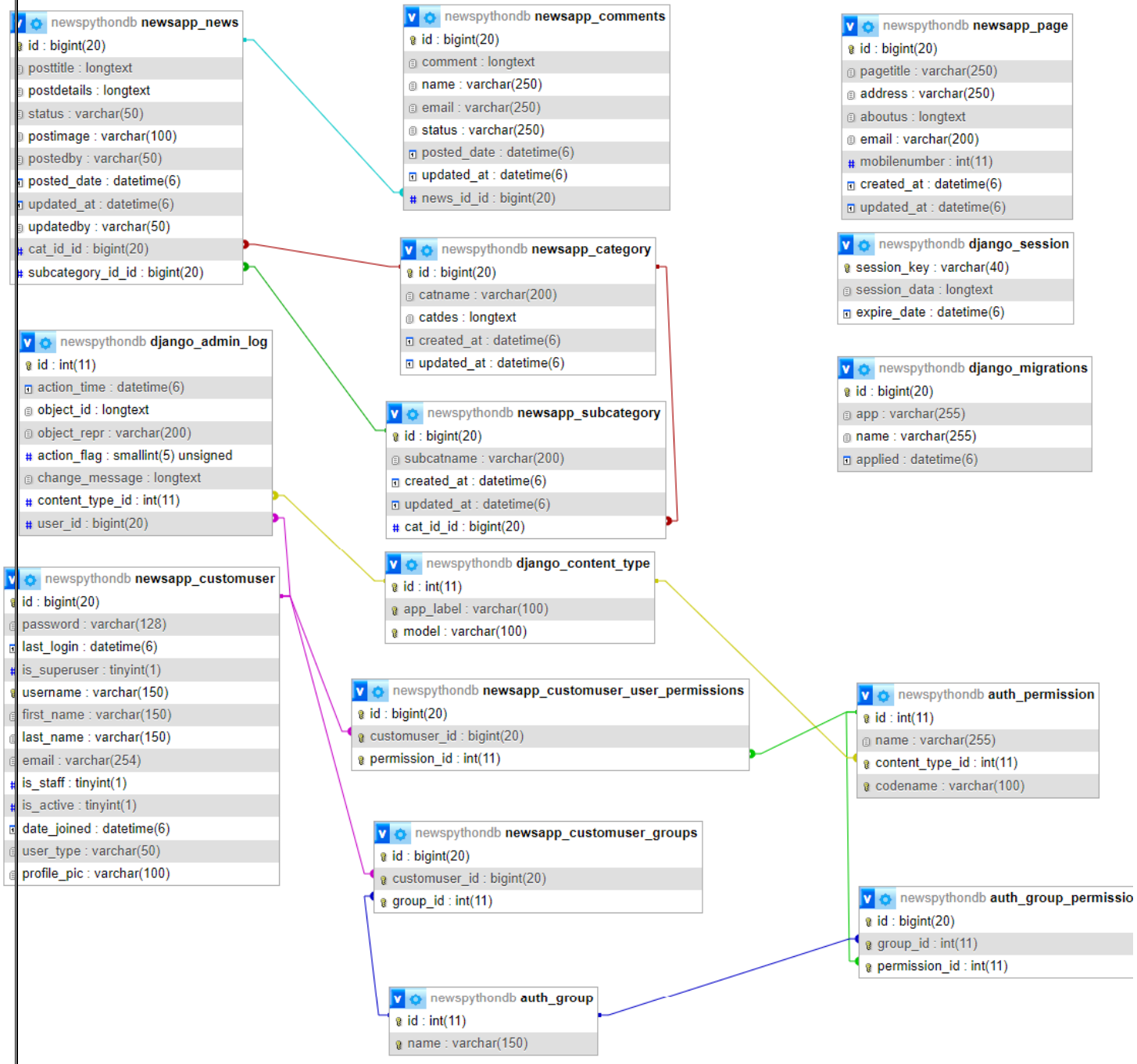
<u>django_migrations table Structure :</u> This table table is part of the database schema and is used to keep track of which migrations have been applied to the database.

```
mysql> DESCRIBE django_migrations;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | bigint       | NO   | PRI | NULL    | auto_increment |
| app     | varchar(255) | NO   |     | NULL    |                |
| name    | varchar(255) | NO   |     | NULL    |                |
| applied | datetime(6)  | NO   |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

<u>django_session table Structure</u> : This table store the session data for user.

```
mysql> DESCRIBE django_session;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| session_key  | varchar(40) | NO   | PRI | NULL    |       |
| session_data | longtext    | NO   |     | NULL    |       |
| expire_date  | datetime(6) | NO   | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Relationship between tables (Class Diagram)



4.3 PROCEDURAL DESIGN:

Process logic (flowchart) of each module

4.3.1 User Panel Design

In user panel design we have done our task for user. Here we provide facility about Online News Paper. In index page user can select any options which is needed by him/her. By selecting options he/she can see the desired page. Then he/she can get the all oriented information finally. The design of user panel is shown in following flow chart….

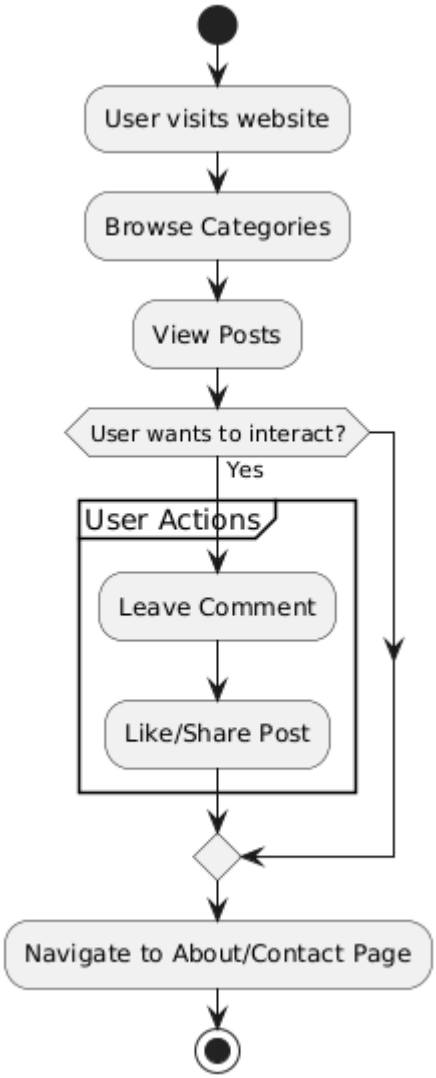


Fig. 4.1: The user panel flowchart part.

4.3.2 Admin Panel Design

We have design user login facility to manage and update all of the information. It is fully secured page. Without appropriate username and password it cannot be accessed by anyone. For admin login after giving username and password we need to click a login button , when we click login button it is not directly entered in home page , it stay in login page. Then it starts a session and set two variables called username and password. If the username and password are matched with database, it can enter in home page. It is not possible without click login button. In case if username or password are not matched with database than Invalid username or password massage is shown. We can describe the login facility in admin login by using below flow chart given below—

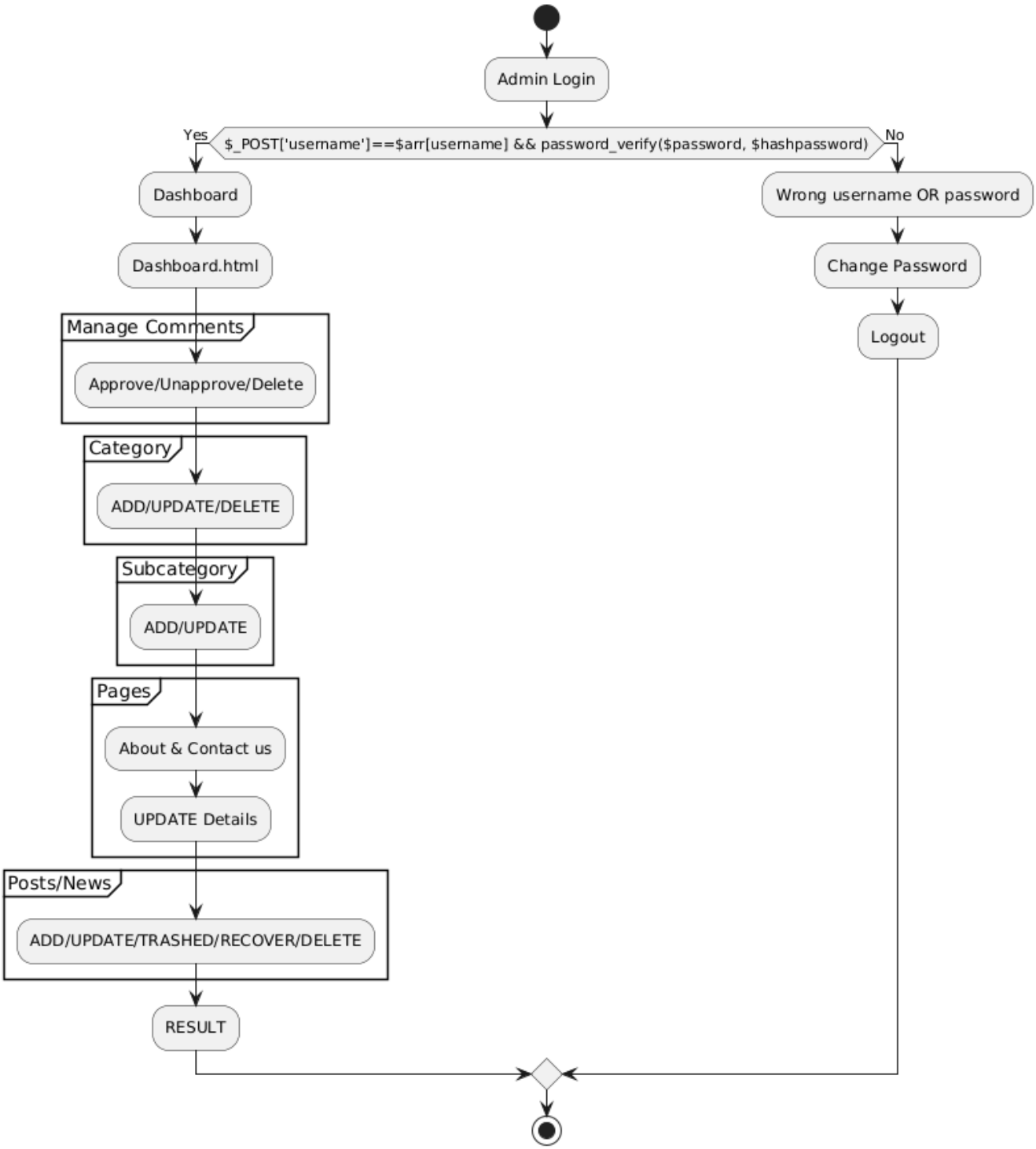
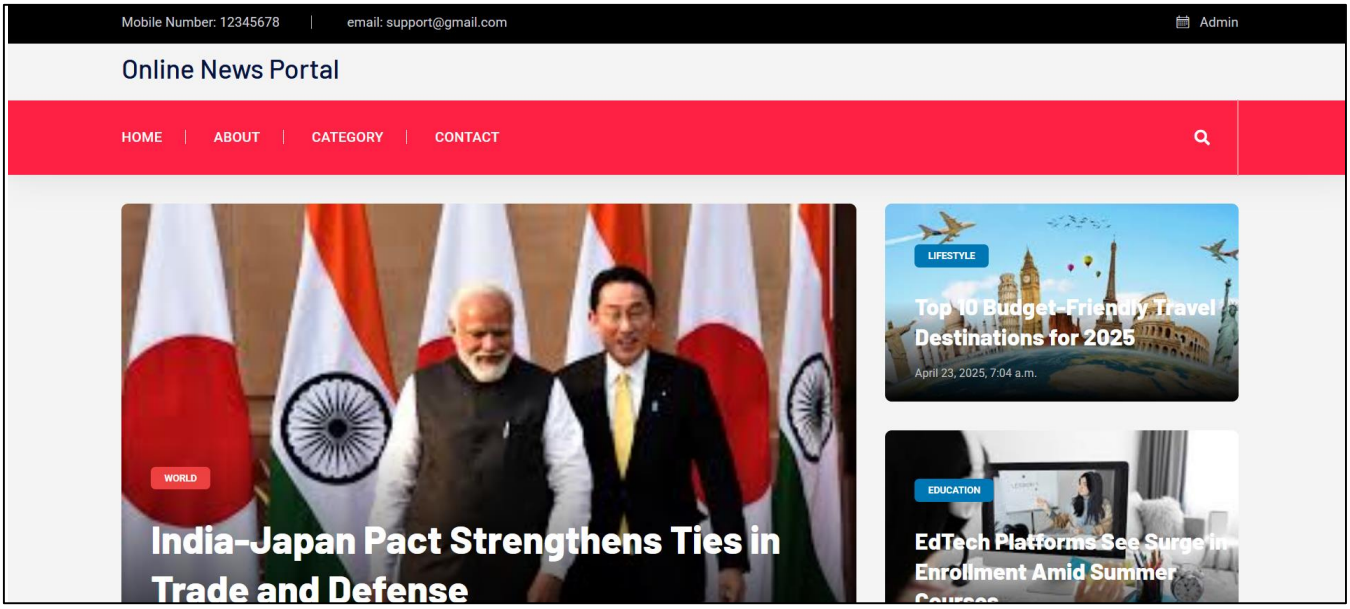


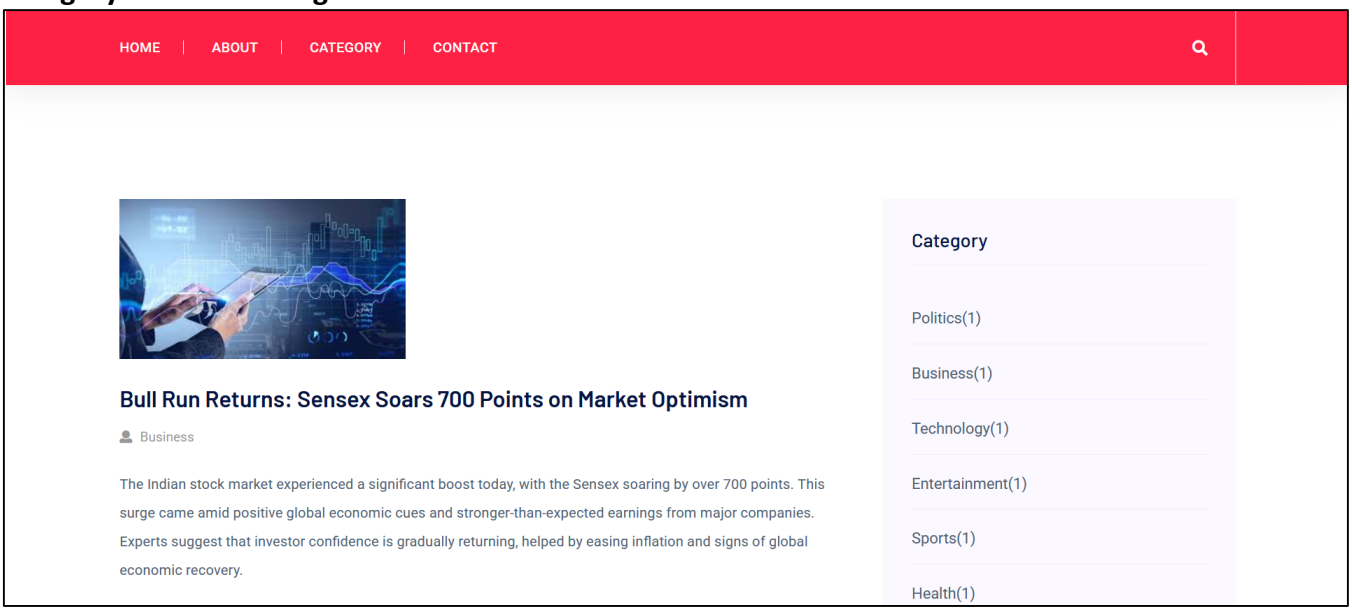
Fig. 4.2: Admin Login Flow Chart.

SCREENSHOTS

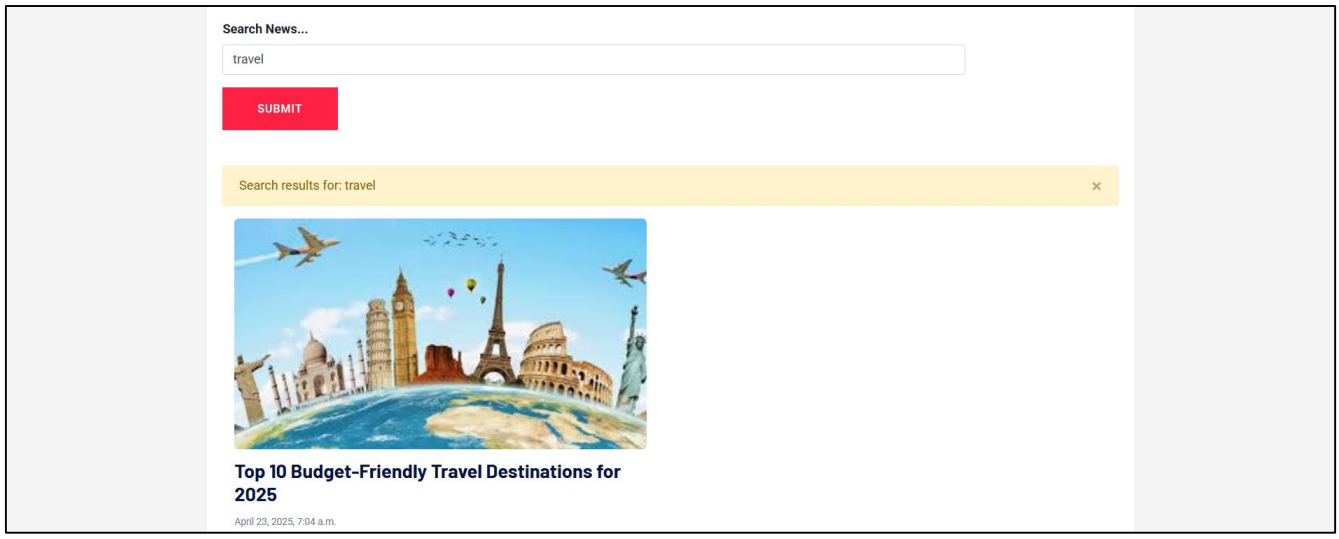
Home Page



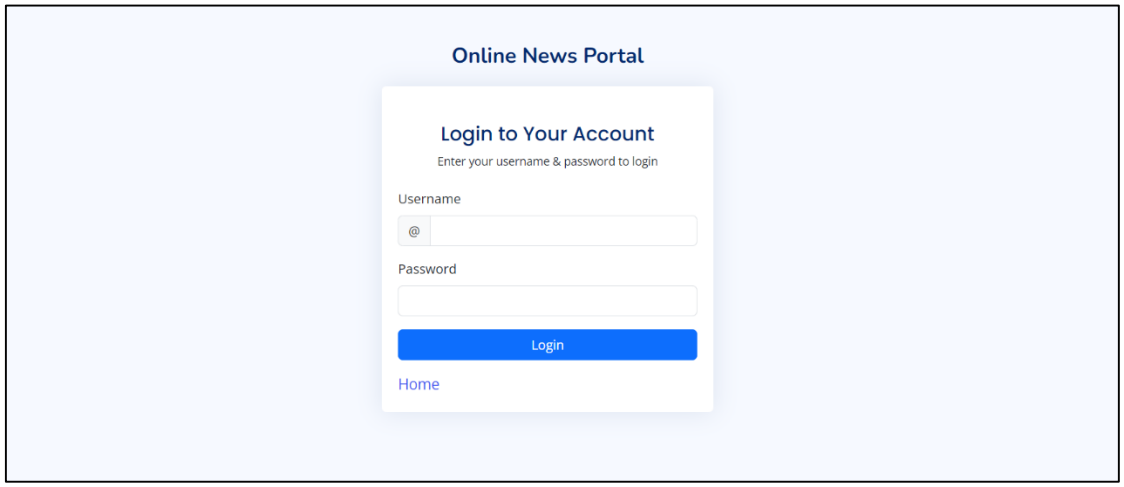
Category wise News single news details



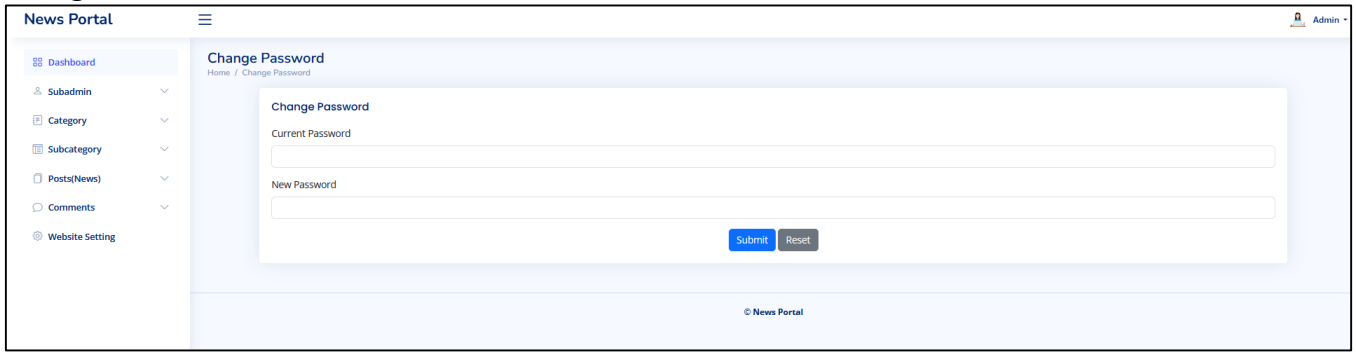
Search News



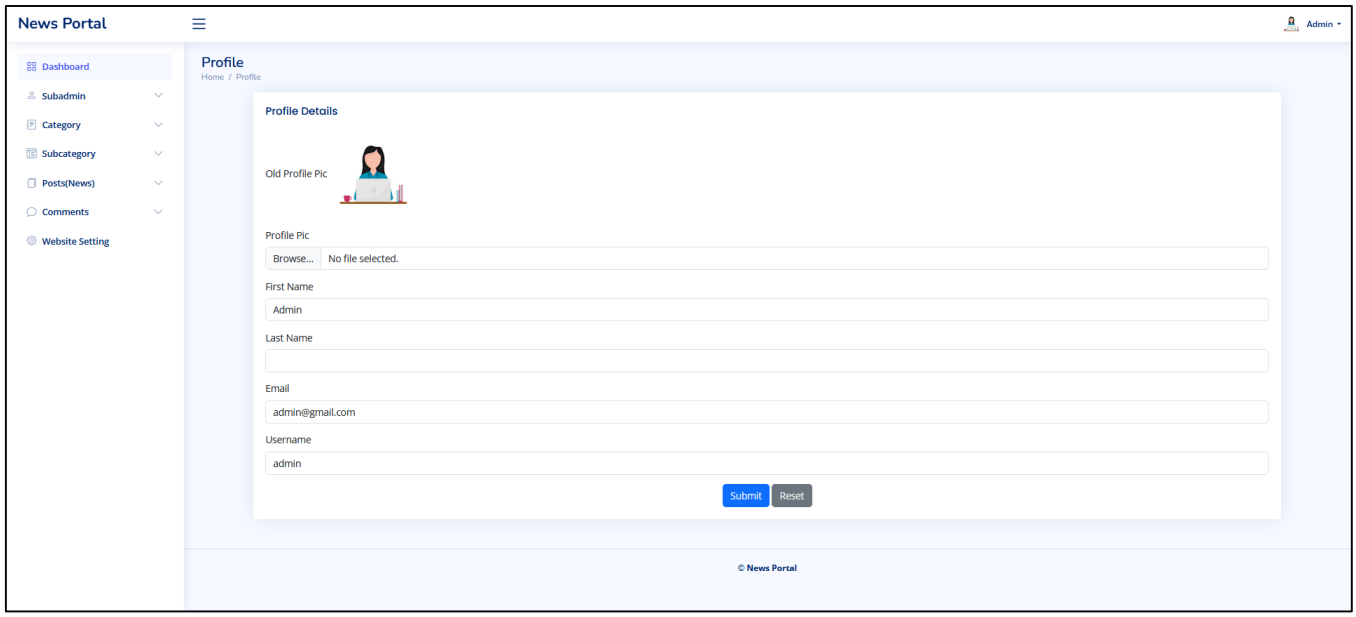
Admin Login



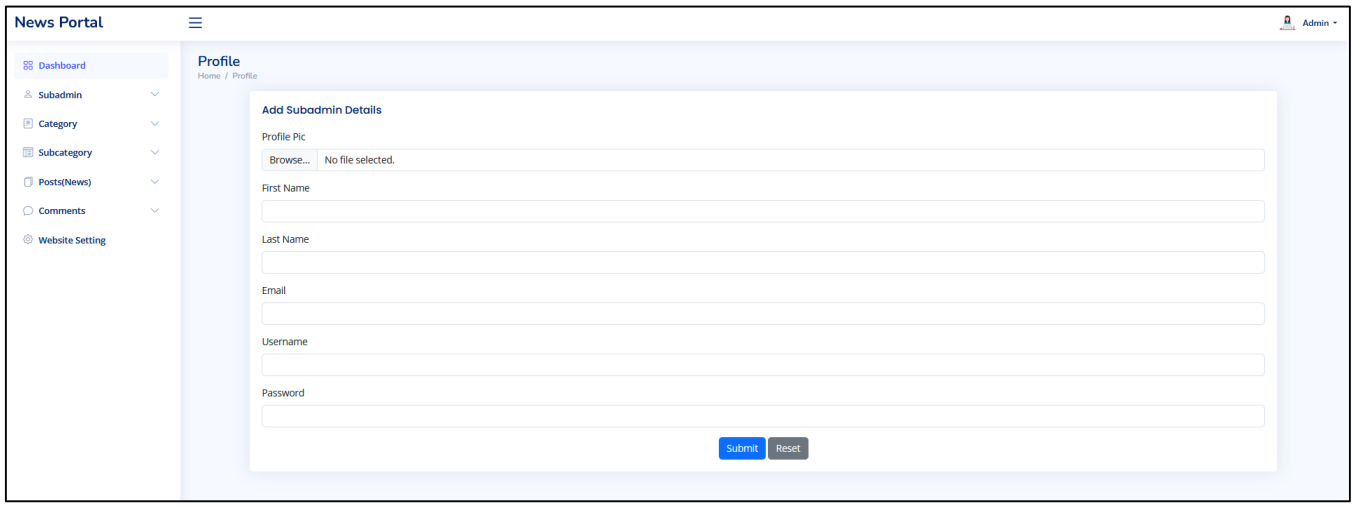
Change Password



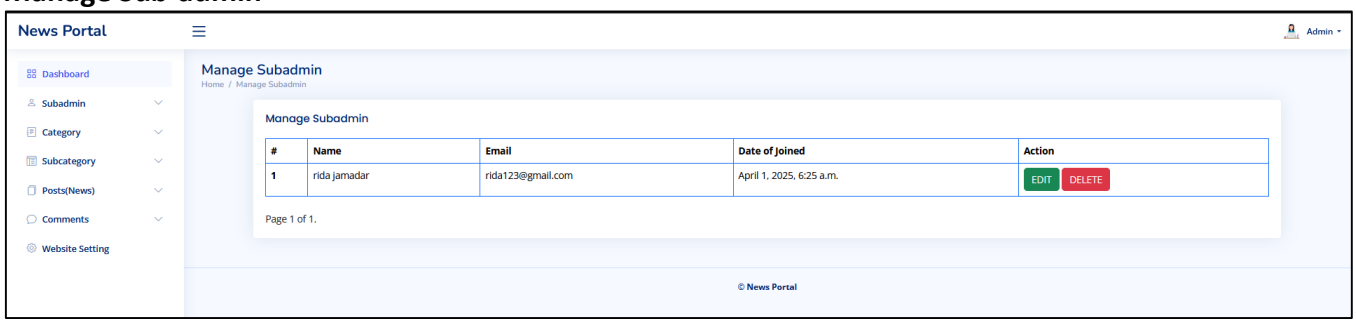
Admin Profile



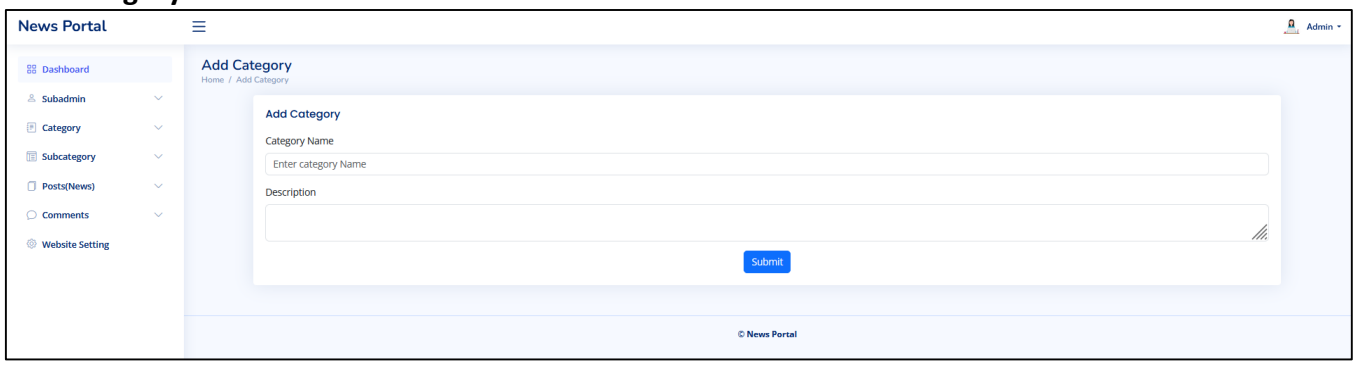
Add Sub Admin



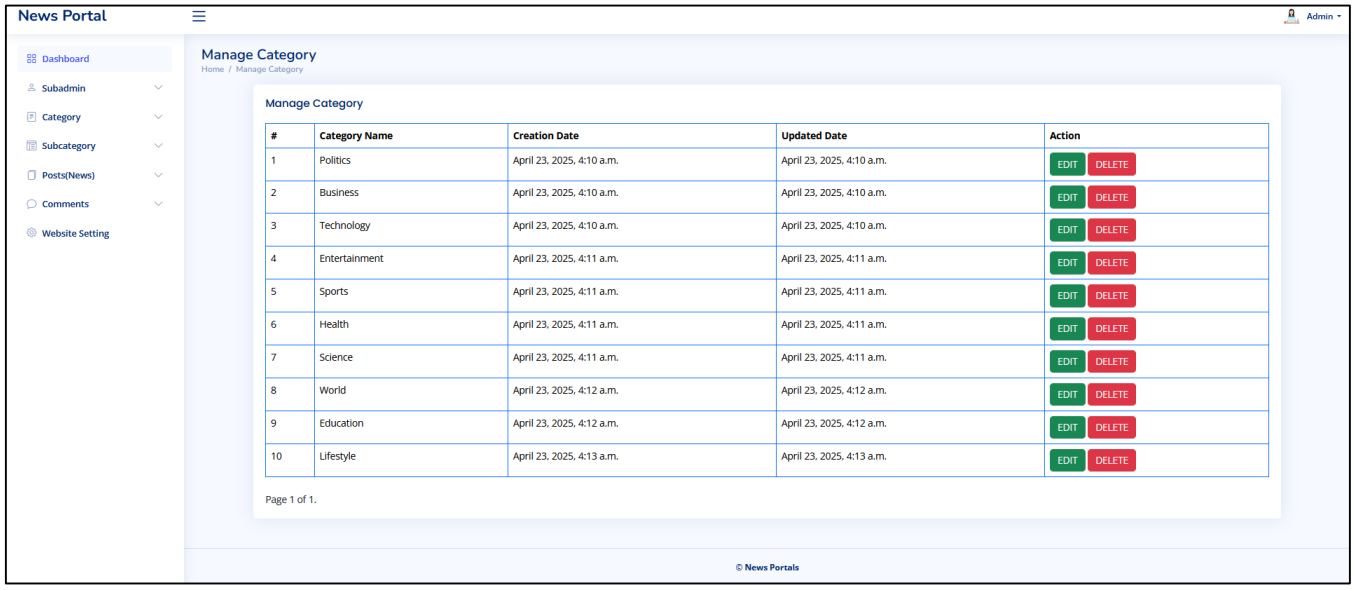
Manage Sub-admin



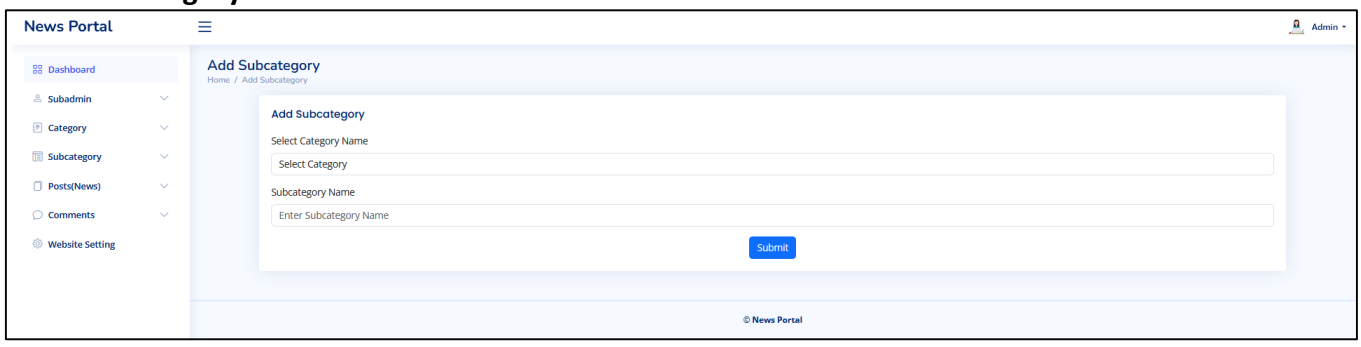
Add Category



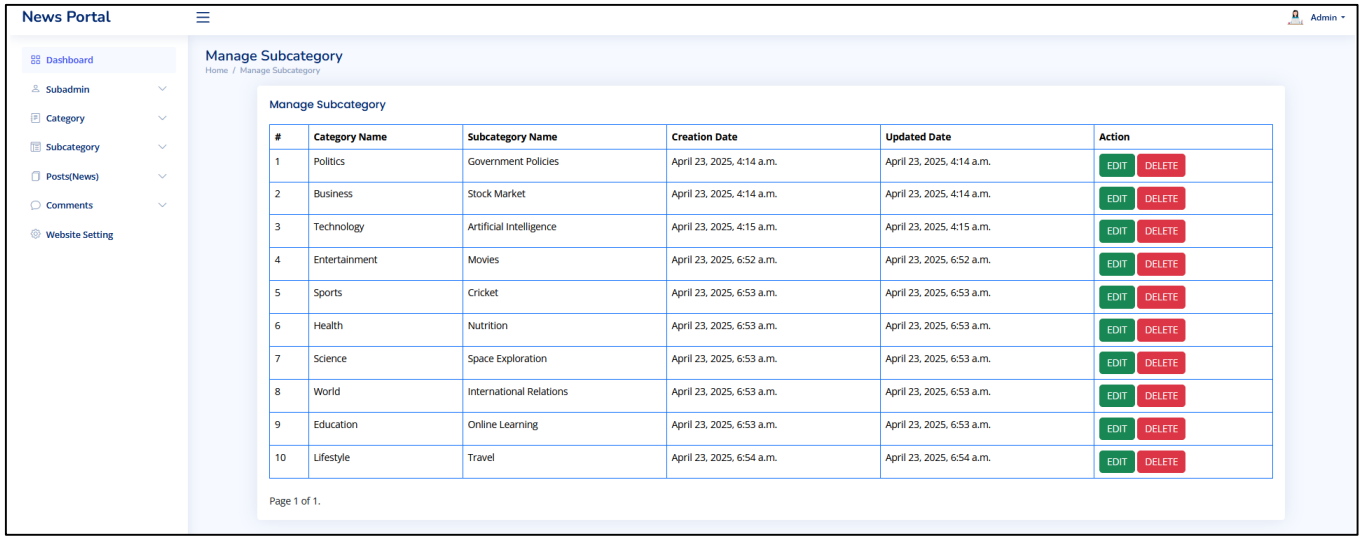
Manage Category



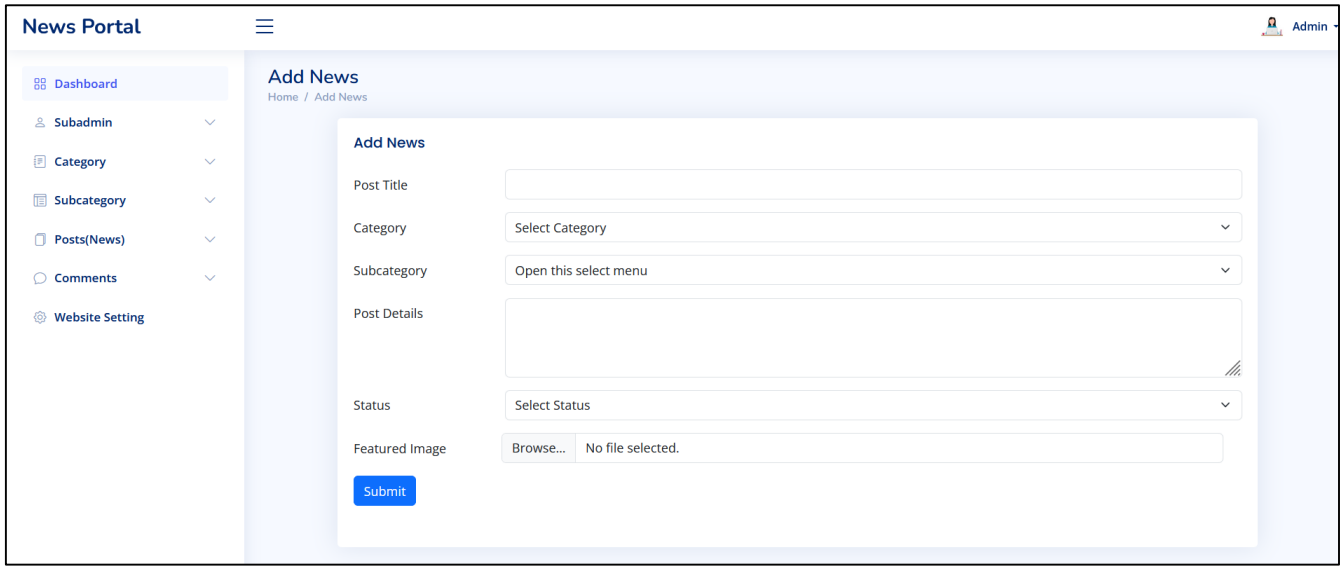
Add Sub-Category



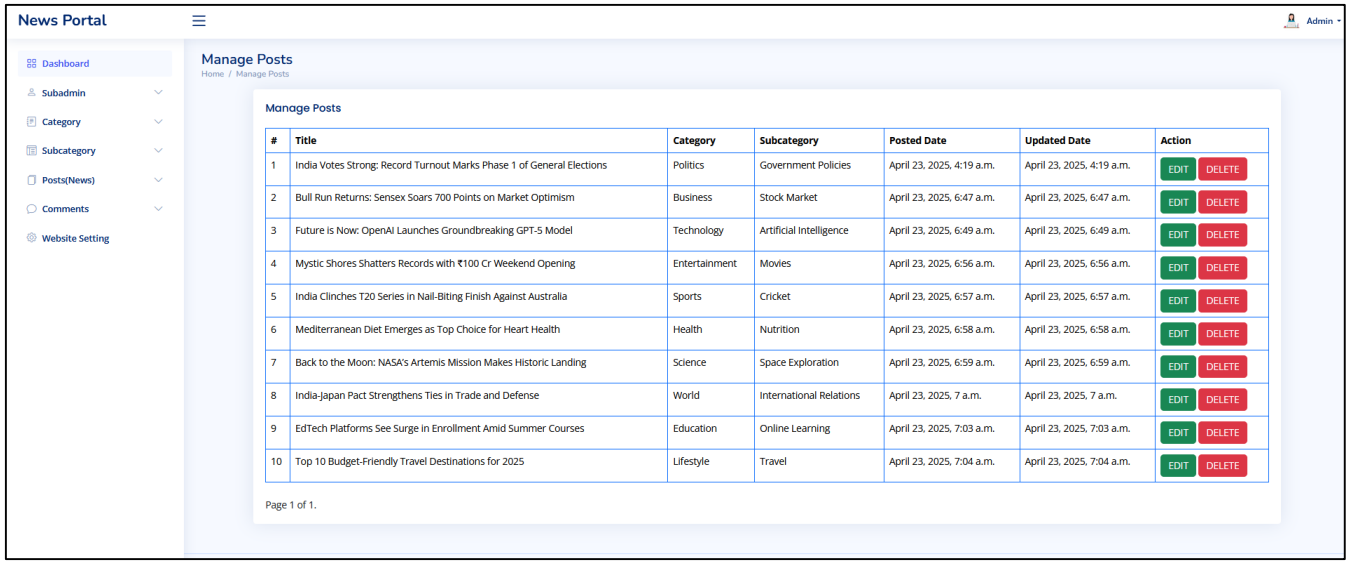
Manage Sub-Category



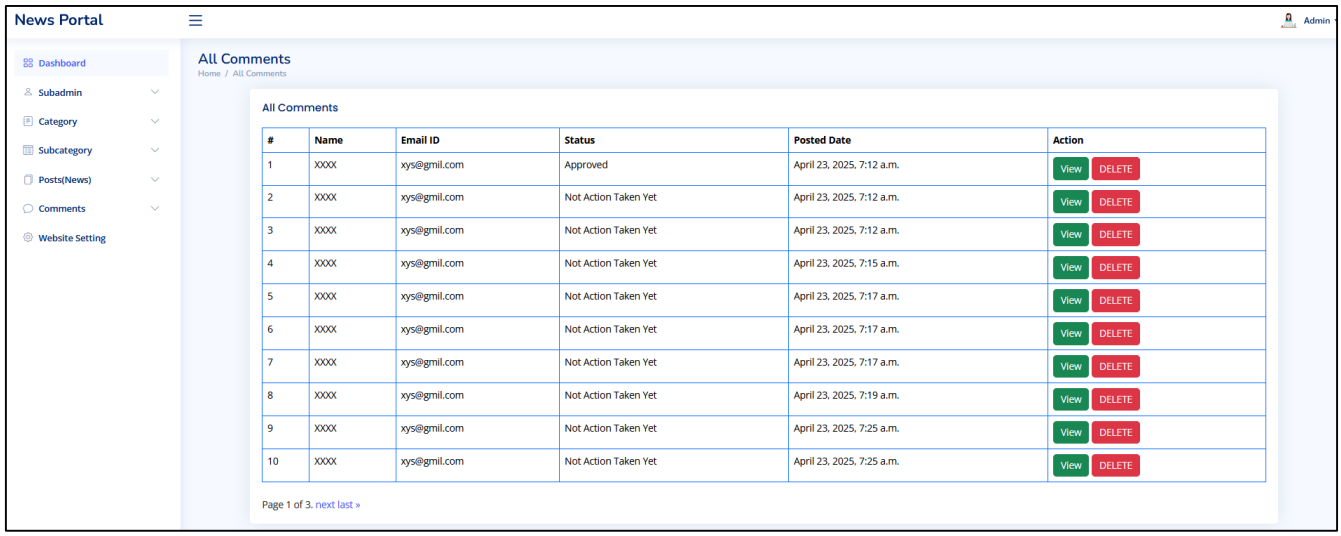
Add Post



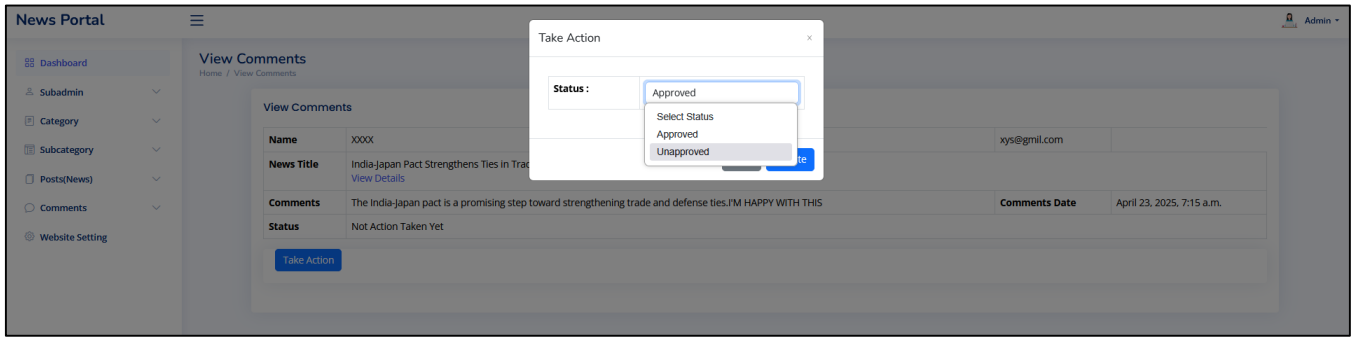
Manage Post



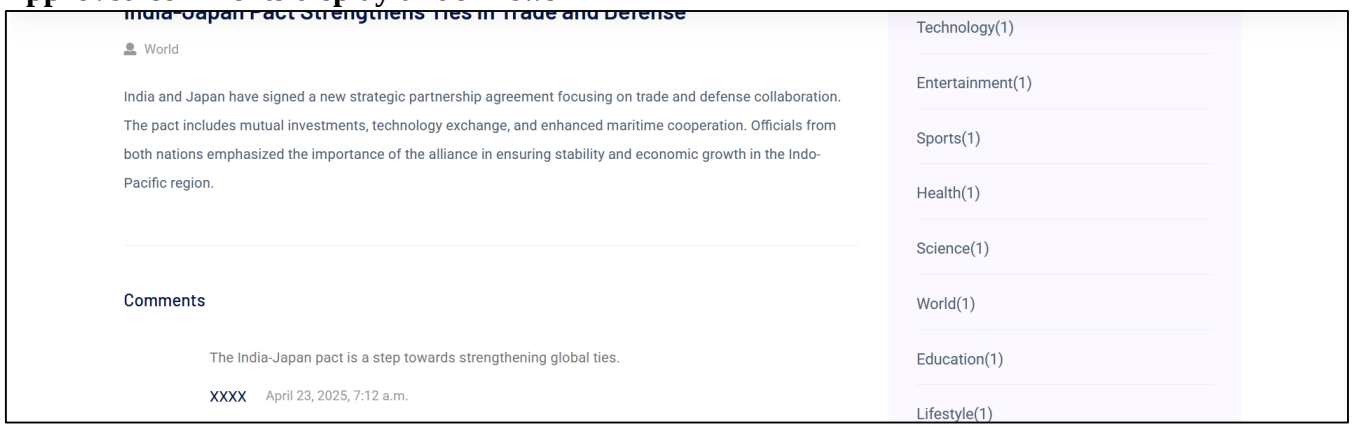
All Comments



Approved comments



Approved comments display under news



Coding

About.html

```
{% extends 'base1.html' %}
{% block content %}
{% load static %}
 <div class="about-details section-padding30">
       <div class="container">
          <div class="row">
             <div class="offset-xl-1 col-lg-8">
                <div class="about-details-cap mb-50">
                   <h4>About Us</h4>
                   <p>{{ page.aboutus }} </p>
                </div>  </div></div> </div> {% endblock %}
```

Login.html

```
 {% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport"
  <title>Online News Portal System|| Login Page</title>
  <!-- Favicons -->
  <link href="{% static 'assets/img/favicon.png'%}" rel="icon">
  <link href="{% static 'assets/img/apple-touch-icon.png'%}" rel="apple-touch-icon">
  <!-- Google Fonts -->
  <link href="https://fonts.gstatic.com" rel="preconnect">
```

```
  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Nunito
:300,300i,400,400i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i"
rel="stylesheet">

  <!-- Vendor CSS Files -->

  <link href="{% static 'assets/vendor/bootstrap/css/bootstrap.min.css'%}" rel="stylesheet">

  <link href="{% static 'assets/vendor/bootstrap-icons/bootstrap-icons.css'%}" rel="stylesheet">

  <link href="{% static 'assets/vendor/boxicons/css/boxicons.min.css'%}" rel="stylesheet">

  <link href="{% static 'assets/vendor/quill/quill.snow.css'%}" rel="stylesheet">

  <link href="{% static 'assets/vendor/quill/quill.bubble.css'%}" rel="stylesheet">

  <link href="{% static 'assets/vendor/remixicon/remixicon.css'%}" rel="stylesheet">

  <link href="{% static 'assets/vendor/simple-datatables/style.css'%}" rel="stylesheet">

  <!-- Template Main CSS File -->

  <link href="{% static 'assets/css/style.css'%}" rel="stylesheet">

</head>

<body>

  <main>

    <div class="container">

      <section class="section register min-vh-100 d-flex flex-column align-items-center justify-content-
center py-4">

        <div class="container">

          <div class="row justify-content-center">

            <div class="col-lg-4 col-md-6 d-flex flex-column align-items-center justify-content-center">

              <div class="d-flex justify-content-center py-4">

                <a href="{% url 'index' %}" class="logo d-flex align-items-center w-auto">

                  <span class="d-none d-lg-block">Online News Portal</span>

                </a>

              </div><!-- End Logo -->

              <div class="card mb-3">

                <div class="card-body">
```

```
<div class="pt-4 pb-2">
  <h5 class="card-title text-center pb-0 fs-4">Login to Your Account</h5>
  <p class="text-center small">Enter your username & password to login</p>
</div>
{% if messages %}
{% for message in messages %}
 {% if message.tags == 'error' %}
<div class="alert alert-warning alert-dismissible fade show" role="alert">
{{message}}
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
<span aria-hidden="true">&times;</span>
 </button>
  </div>
{% endif %}
{% endfor %}
{% endif %}
   {% if messages %}
{% for message in messages %}
 {% if message.tags == 'success' %}
<div class="alert alert-warning alert-dismissible fade show" role="alert">
{{message}}
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
<span aria-hidden="true">&times;</span>
 </button>
  </div>
{% endif %}
{% endfor %}
{% endif %}
```

```
            <form class="row g-3 needs-validation" method="post" action="{% url 'doLogin' %}">
              {% csrf_token %}
              <div class="col-12">
                <input type="hidden" name="next" value="{{ next }}">
                <label for="yourUsername" class="form-label">Username</label>
                <div class="input-group has-validation">
                  <span class="input-group-text" id="inputGroupPrepend">@</span>
                  <input type="text" name="username" class="form-control" id="username" required
value="{{ username|default:'' }}">
                  <div class="invalid-feedback">Please enter your username.</div>
                </div>
              </div>
              <div class="col-12">
                <label for="yourPassword" class="form-label">Password</label>
                <input type="password" name="password" class="form-control" id="password" required>
                <div class="invalid-feedback">Please enter your password!</div>
              </div>
              <div class="col-12">
                <button class="btn btn-primary w-100" type="submit">Login</button>
              </div>
              <div class="col-12">
                <p class="small mb-0" style="font-size: large;"><a href="{% url 'index'
%}">Home</a></p>
              </div>
            </form>
          </div>
        </div> </div> </div></div></section></div></main><!-- End #main -->
```

```
  <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>
  <!-- Vendor JS Files -->
  <script src="{% static 'assets/vendor/apexcharts/apexcharts.min.js'%}"></script>
  <script src="{% static 'assets/vendor/bootstrap/js/bootstrap.bundle.min.js'%}"></script>
  <script src="{% static 'assets/vendor/chart.js/chart.umd.js'%}"></script>
  <script src="{% static 'assets/vendor/echarts/echarts.min.js'%}"></script>
  <script src="{% static 'assets/vendor/quill/quill.js'%}"></script>
  <script src="{% static 'assets/vendor/simple-datatables/simple-datatables.js'%}"></script>
  <script src="{% static 'assets/vendor/tinymce/tinymce.min.js'%}"></script>
  <script src="{% static 'assets/vendor/php-email-form/validate.js'%}"></script>
  <!-- Template Main JS File -->
  <script src="{% static 'assets/js/main.js'%}"></script>
</body></html>
```

Profile.html

```
{% extends 'base.html' %}
{% block content %}
<div class="pagetitle">
      <h1>Profile</h1>
      <nav>
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="{% url 'dashboard' %}">Home</a></li>
          <li class="breadcrumb-item">Profile</li>
        </ol></nav>
    </div><!-- End Page Title -->
    <section class="section">
      <div class="row">
        <div class="col-lg-12" style="padding-left: 100px;padding-right: 100px;">
```

```
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Profile Details</h5>
    {% if messages %}
    {% for message in messages %}
     {% if message.tags == 'error' %}
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
    {{message}}
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
     </button></div>
    {% endif %}{% endfor %}{% endif %}
      {% if messages %}
    {% for message in messages %}
     {% if message.tags == 'success' %}
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
    {{message}}
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
     </button></div>
    {% endif %}{% endfor %}{% endif %}
    <!-- Vertical Form -->
    <form class="row g-3" method="POST" action="{% url 'admin_profile_update' %}"
enctype="multipart/form-data">
      {% csrf_token %}
      <div class="col-12">
        <label for="inputNanme4" class="form-label">Old Profile Pic</label>
        <img src="{{user.profile_pic}}">
      </div>
```

```
            <div class="col-12">
              <label for="inputNanme4" class="form-label">Profile Pic</label>
              <input type="file" class="form-control" name="profile_pic">
            </div>
            <div class="col-12">
              <label for="inputEmail4" class="form-label">First Name</label>
              <input type="text" class="form-control" name="first_name" value="{{user.first_name}}">
            </div>
            <div class="col-12">
              <label for="inputPassword4" class="form-label">Last Name</label>
              <input type="text" class="form-control" name="last_name" value="{{user.last_name}}">
            </div>
            <div class="col-12">
              <label for="inputAddress" class="form-label">Email</label>
              <input type="email" class="form-control" readonly="True" name="email"
value="{{user.email}}">
            </div>
            <div class="col-12">
               <label for="inputAddress" class="form-label">Username</label>
               <input type="text" class="form-control" readonly="True" name="username"
value="{{user.username}}">
              </div>
            <div class="text-center">
              <button type="submit" class="btn btn-primary">Submit</button>
              <button type="reset" class="btn btn-secondary">Reset</button>
            </div>
          </form><!-- Vertical Form -->
        </div>
      </div></div></div> </section>{% endblock %}
```

Testing

Unit Testing: Unit testing where individual program units or object classes are tested. Here by using this testing we have focused on testing the functionality of methods.

Module Testing: Where this is the combination of unit program is called module. Here we tested the unit program (5-6 programs) is where the module programs have dependency.

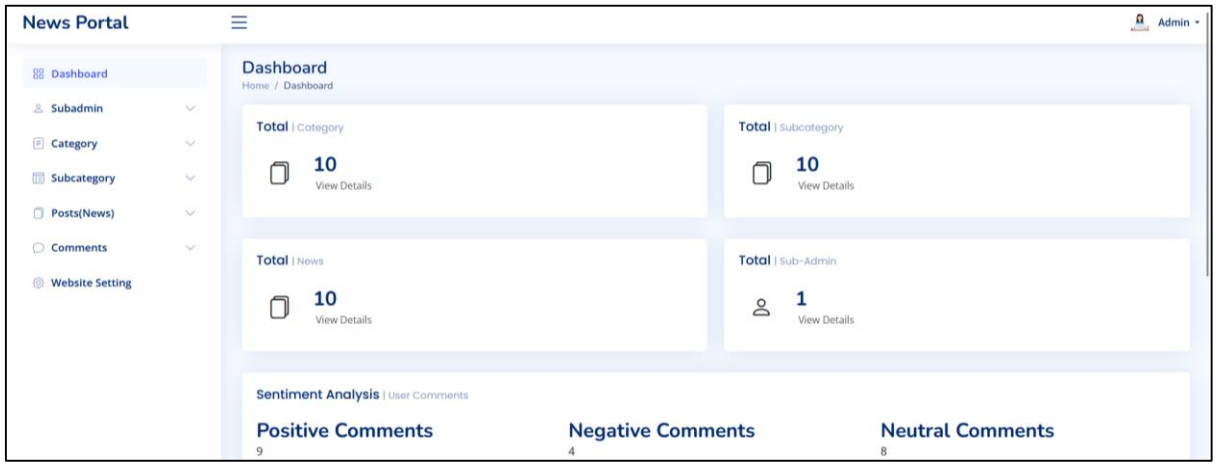
Sub-system Testing: Then we combined some module for the Preliminary System Testing in our Project.

System Testing: Where it is the combination of two or more sub-system and then it is tested. Here we tested the Entire system as per the requirements.

Acceptance Testing: Normally this type of testing is done to verify if system meets the customer specified requirements. After submitting this project to User then they tested it and to determine whether to accept application. It is the system testing performed by the customer(s) to determine whether they should accept the delivery of the system.

Report

Admin Dashboard Report



Sentiment analysis based on comments

6. Future scope

The Online News Portal can be enhanced with the following features:

- **Personalized News Feeds** using AI-based recommendations
- **Mobile App** for better accessibility
- **Real-time News** Updates via APIs or web sockets
- **Multilingual** Support to reach a broader audience
- **Advanced Search & Filters for** refined article discovery
- **AI-driven Comment Moderation** for a safer community
- **User-Generated Content submission** with admin approval
- **Admin Analytics Dashboard** to track content performance

Conclusion

The Online News Portal System offers a comprehensive and efficient solution for managing and delivering news content in a digital format. By automating the process, it enhances productivity, ensures timely updates, and provides a user-friendly interface for both readers and administrators. The system incorporates role-based access, ensuring secure and appropriate functionality for users, admins, and sub-admins. With features like categorized news, comment moderation, and sentiment analysis, it fosters better user engagement and feedback management. Designed with scalability and flexibility in mind, the system is well-equipped to adapt to future enhancements, making it a reliable and robust platform for digital news delivery.

BIBLIOGRAPHY

- *Web Development with Django* by Ben Shaw & Saurabh
- *Django 4 By Example* by Antonio Mele

REFERENCES

· Django Homepage – http://www.djangoproject.com/

· Python Documentation – http://www.python.org/doc

· Django (Web Framework) – http://en.wikipedia.org/wiki/Django

· Django Documentation – http://docs.djangoproject.com

· Python (Programming Language) – http://en.wikipedia.org/wiki/Python

· MySQL Official Site – https://www.mysql.com/

· MySQL Tutorials – http://www.mysqltutorial.org

XAMPP Download & Documentation – https://www.apachefriends.org/download.html