A

PROJECT REPORT

ON

BUZZNEWS – NEWS AGGREGATOR

SUBMITTED BY

ANSHU V BARAIYA

VAIDEHI R SHRESHTH

ACADEMIC YEAR 2025-26

T.Y B.C.A SEM -5

UNDER THE GUIDANCE OF

MR. SHAUNAK PUROHIT

SMT. C.Z.M GOSRANI B.C.A. COLLEGE

JAMNAGAR

SUBMITTED TO

SAURASHTRA UNIVERSITY

RAJKOT

INDEX

project report on

BuzzNews

-dynamic news aggregator

Developers:

Anshu V Baraiya

Vaidehi R Shreshth

ABSTRACT

This project ‘BuzzNews’ is a web-based news portal built using Flask.

The project aims to develop a user-friendly news website along with gathering public participation by giving a voice to everyday readers using ‘local voices ’.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have supported us throughout the development of this project ‘BuzzNews’.

First and foremost, we extend our gratitude to our project guide Mr. Shaunak Purohit, who has provided us valuable insights, timely feedback and constant encouragement in shaping this project.

Finally, thank you to my peers and family for the constant motivation throughout the journey.

PROJECT PROFILE

Developers	
NAME:	ANSHU V. BARAIYA 23SI002UG03572 VAIDEHI R. SHRESHTH 23SI002UG03740

Project Details	
Project title:	Dynamic News Aggregator
Duration:	3 months
Project name:	BuzzNews
Platform:	Flask (backend) Html, CSS (frontend)
Team mates:	2
Guide's name:	Mr. SHAUNAK PUROHIT

1.SDLC OVERVIEW

1. DEVELOPER INPUT

I. Project initiators:

Anshu Baraiya
Vaidehi Shreshth

II. Vision for the project:

> *"We were creating a news aggregator website at the state level coverage, but something was required to make it stand out. So, we created a website that allows readers to input a news of value to create a section for localization through which people can feel connect with their locality better."*

1.3 Input data to the system :

- Text
- Image

1.4 Output data from the system :

- Text
- Images

1.5 Type of project:

- Web-Based Application

1.6 Method for requirement collection

- Brainstorming
- Discussion with peers
- Online research on similar platforms

2.REQUIREMENT GATHERING

2.1 Project context :

1.name of the project :

BuzzNews – dynamic News Aggregator

2. nature of the project :

This project is a self-initiated academic web application developed as part of the university curriculum. It is not affiliated with any real-world organization and is designed to showcase technical skills, understanding of web technologies, and implementation of software development life cycle phases.

2.2 Idea development :

Since the idea was based on self-generated idea, no external organization meeting/ interviews were conducted.

The ideas were brainstormed and refined through personal research and taking feedbacks from peers and faculty.

3. SYSTEM REQUIREMENT GATHERING

3.1 Introduction:

3.1.1 Purpose

'BuzzNews' is a web-based community driven news aggregator which aims in connecting the people through news – local and state level.

3.1.2 Intended audience

The intended audience of this document is the potential end user. This document also serves as a reference guide.

3.1.3 Product scope

The product will be an online News reading platform. The product is scaled at Gujarat level and any user can register and read news and write 'local news' as well.

3.2 OVERALL DESCRIPTION :

3.2.1 PRODUCT FUNCTION

I. General User Module

Register on the website

Read articles

Write on 'local voices'

II. Journalist Module

Read articles

Create articles

III. Moderator Module

Read articles

Review/approve/reject *local voices*

Review/approve/reject journalist articles

Manage articles, categories and districts

Add/remove users

Assign roles (journalist)

IV. Admin Module

Read articles

Create articles

Review/approve/reject *local voices*

Review/approve/reject journalist articles

Manage articles, categories and districts

Add/remove users

Assign roles (moderator/journalist)

Access to admin panel

3.2.2 User classes and characteristics

1. User Classes:

General User

Journalist

Moderator

Admin

2. User Characteristics:

1. General user:

User should have basic knowledge of internet browsing to be able to navigate the website comfortably on mobile or desktop.

2.Journalist:

Journalist should be capable of submitting articles, uploading media files, and categorizing news properly.

3.Moderator:

Moderator must be familiar with using dashboards and handling multiple content entries.

4.Admin:

Administrator must be capable of managing the database, user accounts, and platform content.

3.2.3 operating environment

This App will be operated through the internet including hardware platforms like laptop also operating system.

3.2.4 design and implementation constraints

GUI is only in English.

Internet connection is necessary.

Users should have basic knowledge of internet surfing.

3.2.5 assumptions and dependencies

- Admin is already registered in the system.
- The database can be managed by the admin and partially managed by the moderator.
- Roles and tasks are predefined.

3.3 EXTERNAL INTERFACE REQUIREMENT :

3.3.1 user interface

In case the user is not registered, they can create an account by registering general information. Once the account is created, he/she can 'login' by entering their account credentials.

3.3.2 software interface

- Operating System

- Front end – Html, CSS, JS
- Back end – Flask, Mysql

3.4 SYSTEM FEATURES :

- To provide secure login for different user roles(user, journalist, moderator).
- Display district wise news and enables browsing for local/ state-level news.

3.5 OTHER NON-FUNCTIONAL REQUIREMENTS :

3.5.1 performance requirement:

- Overall system should be fast and error free.
- The system should be able to handle
- amount of data.

3.5.2 security requirement :

- The admin has more rights than user.
- Moderators and journalist account are created by the admin.

3.6 FEASIBILITY STUDY :

A feasibility study is a preliminary investigation of a system that checks if the system can operate smoothly for the desired purpose.

3.6.1 operational feasibility:

Operational feasibility refers to how effectively the proposed system addresses the problems and leverages the opportunities identified during the scope definition phase. It also evaluates how well the system meets the requirements outlined during the requirements analysis stage of system development.

3.6.2 technical feasibility :

Technical feasibility evaluates whether the proposed system can be developed using the existing hardware, software technologies, and available human resources.

3.6.3 economic feasibility:

Economic feasibility assesses whether the expected benefits of the system justify the costs involved in its development and implementation.
This involves a cost-benefit analysis to determine if the project offers sufficient returns in terms of efficiency.

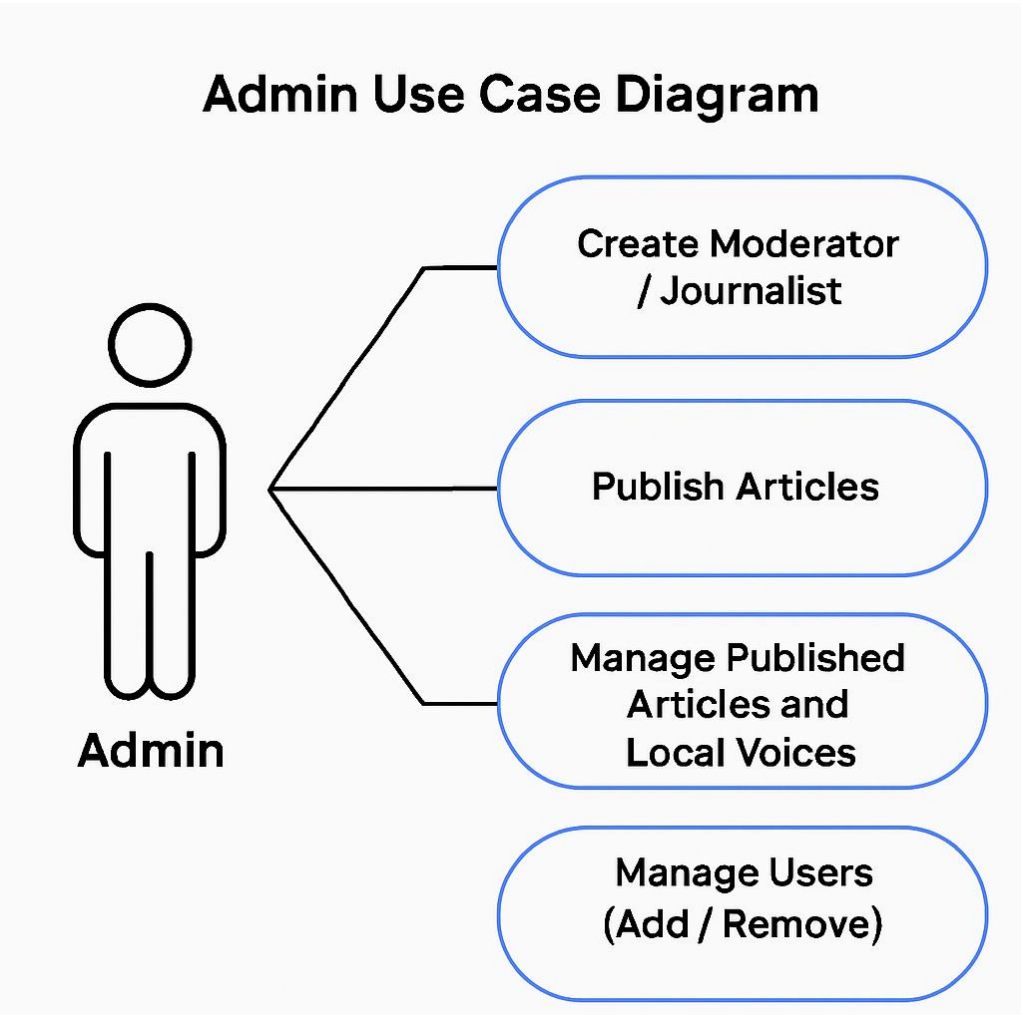
3.6.4 schedule feasibility:

Schedule feasibility examines whether the project can be completed within the desired time frame, considering the availability of required technical skills and resources.

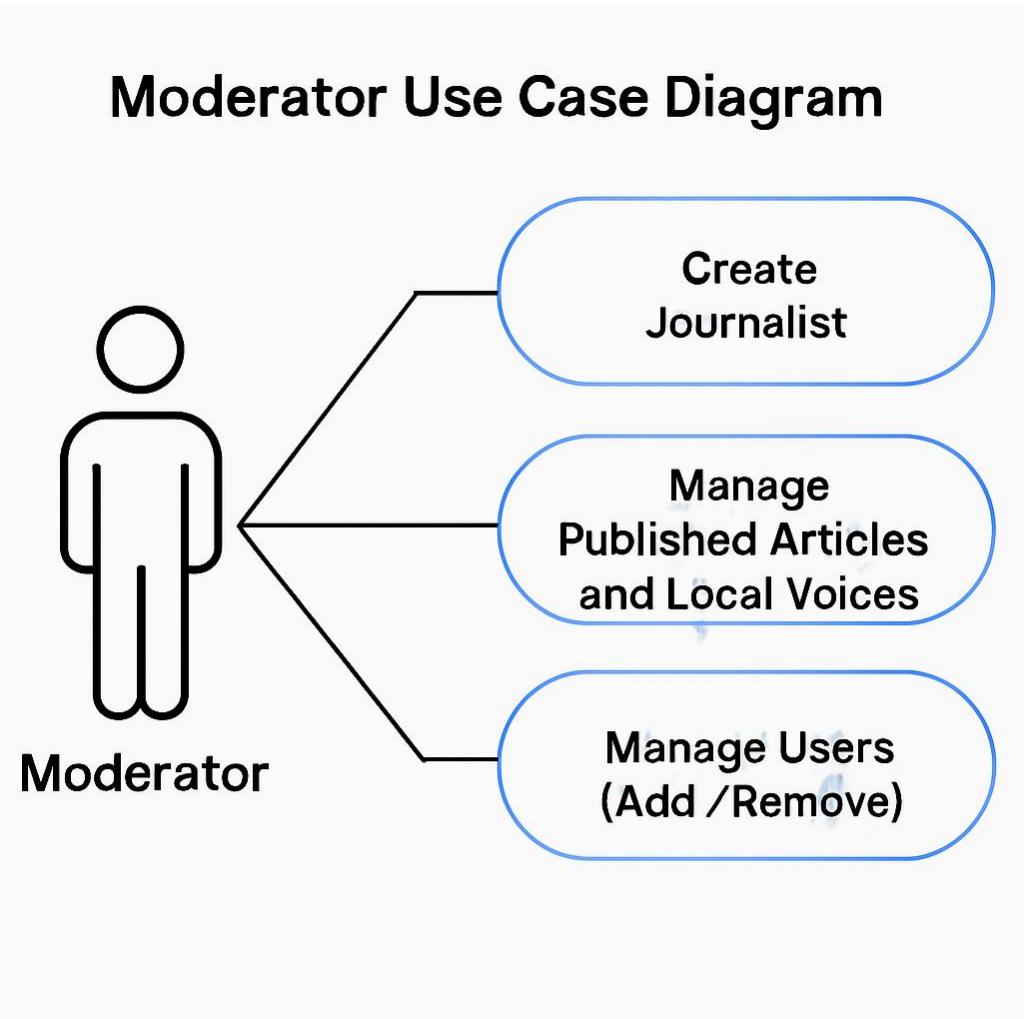
4. SYSTEM ANALYSIS AND MODELING

4.1 use case diagram :

I. Admin side

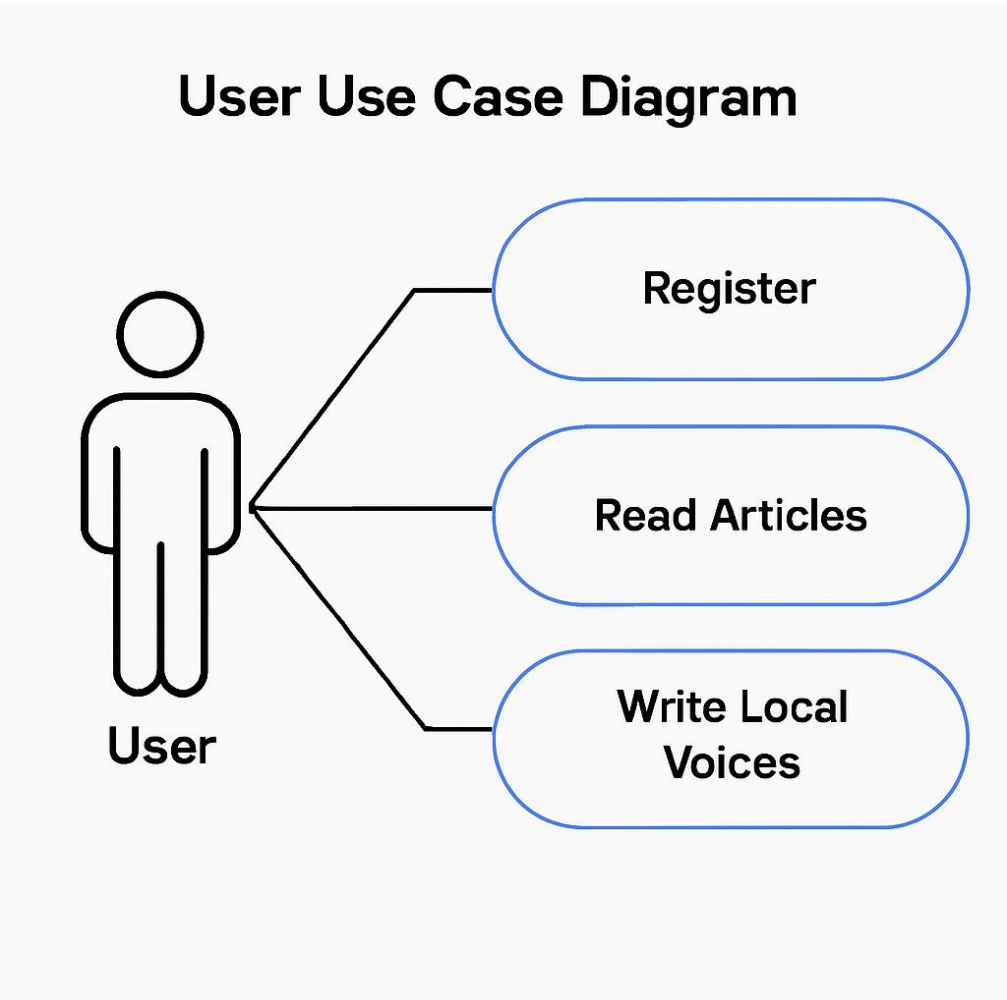

Admin Use Case Diagram
Admin
Create Moderator / Journalist
Publish Articles
Manage Published Articles and Local Voices
Manage Users (Add / Remove)

II. Moderator side



III. Journalist side

IV. General User side



4.2 Normalization and E-R diagram:

Normalization is the process of organizing table data and relations of a relational database to reduce data redundancy and improve data integrity.

- Denormalized form-

Here, the data is stored in a single table, with repeating groups and redundancy. For example, in this table all the entities are crammed up in one table.

USER ID	USERNAME	PASSWORD	ROLE	DISTRICT	ARTICLE ID	ARTICLE TITLE	ARTICLE CONTENT
1	admin	****	Admin	Tapi	101	Economy Growth	Full content
2	Mod01	***	Moderator	Ahmedabad	102	Flood relief	Full content
3	User03	***	User	Jamnagar	205	Health camp in the city	Full content

IMAGE URL	CATEGORY	STATUS	IS_LOCAL_VOICE	SUBMIT-TIME	REJECTION_REASON
Economy.jpg	Economy	Approved	False	2025-08-15 12:30	Content quality
Flood.png	Disaster	Approved	False	2025-07-25 15:00	Repetitive Information
Camp.jpg	Health	Pending	True	2025-08-24 11:26	NULL

- First Normal Form (1NF):

Repeating data is removed, ensures atomic values and provides some structure. Redundancy is still there.

Users table

User ID	Username	Password	Role_id	District_id
1	admin123	****	Admin	Ahmedabad
2	mod01	****	Moderator	Surat
3	jour01	****	Journalist	Rajkot
4	user01	****	Reader	Vadodara

Articles table

ArticleID	**UserID**	**Title**	**Content**	**Status**	**Is_local_voice**
101	1	Economy Growth	Full content…	Approved	1
102	2	Flood Relief	Full content...	Pending	0
103	3	Sports Event	Full content...	Draft	1

- Second Normal Form (2NF):

Removes partial redundancy, creates separate table for roles and districts.

Users table

user_id	username	password	role id	district id
1	Admin123	***	1	101
2	Mod01	***	2	102
3	Jour01	***	3	103
4	User01	***	4	104

Roles table

RoleID	RoleName
1	Admin
2	Moderator
3	Journalist
4	Reader

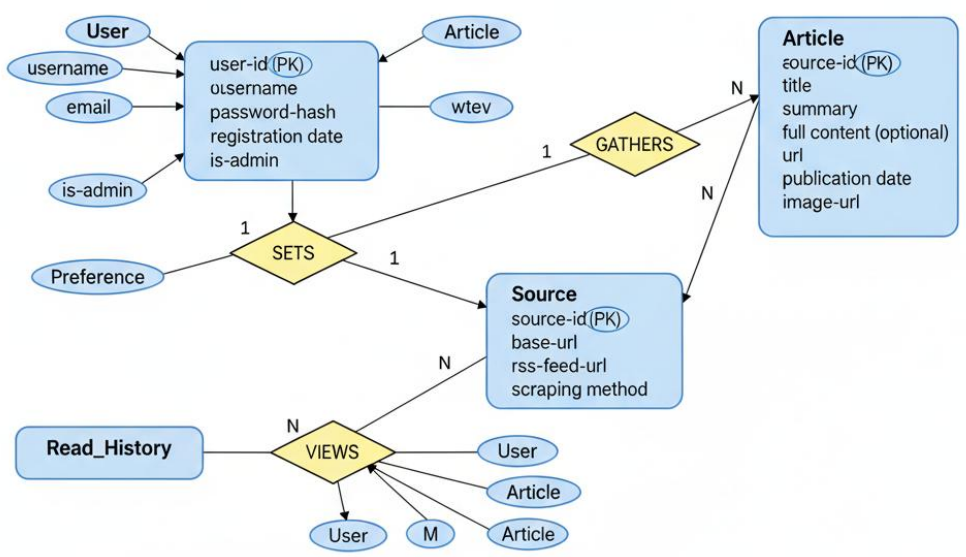
Districts table

DistrictID	DistrictName
101	Ahmedabad
102	Surat
103	Rajkot
104	Jamnagar

Articles table

article id	user id	title	content	status
201	1	Economic growth	Full content	Published
202	2	Flood relief	Full content	Pending
203	3	Sports event	Full content	Rejected

- E-R diagram



4.3 Data dictionary

1. users

field name	data type	Constraints	Description
id	int	PRIMARY KEY, AUTO_INCREMENT	Unique Id for each
username	varchar(50)	UNIQUE, NOT NULL	Unique login name
password	varchar(255)	NOT NULL	Password for authentication
role_id	int	FOREIGN KEY -> roles(id)	Identify user's role
district_id	int	FOREIGN KEY -> districts(id)	Identify user's district

2. roles

field name	data type	Constraints	Description
Id	Int	PRIMARY KEY, AUTO_INCREMENT	Unique Id for each
name	Varchar(50)	UNIQUE, NOT NULL	Role name (reader, journalist, moderator, admin)

3.districts

field name	data type	Constraints	Description
Id	Int	PRIMARY KEY, AUTO_INCREMENT	Unique Id for each
name	Varchar(50)	UNIQUE, NOT NULL	User's district's name

4.categories

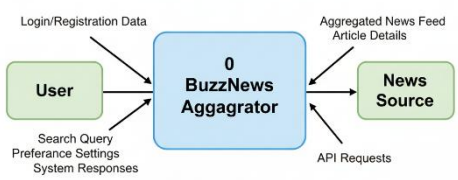
field name	data type	Constraints	Description
Id	Int	PRIMARY KEY, AUTO_INCREMENT	Unique Id for each
name	Varchar(50)	UNIQUE, NOT NULL	News categories

5.articles

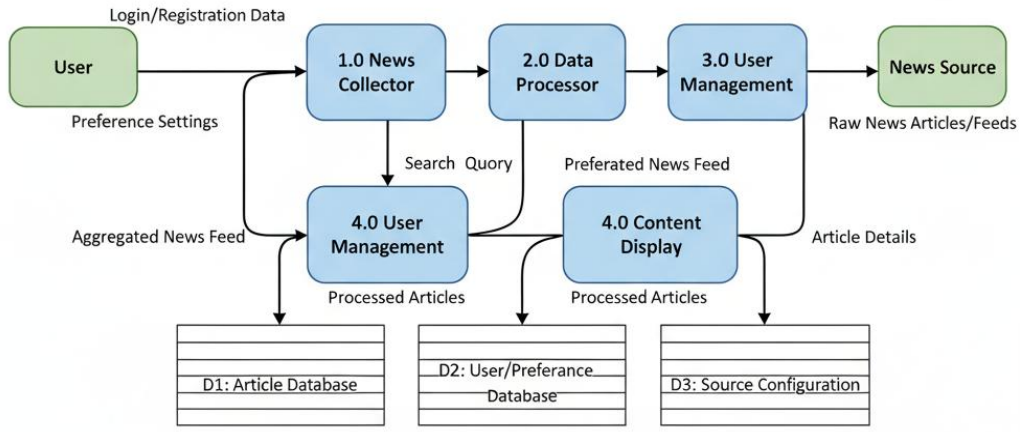
field name	data type	constraints	description
id	int	primary key, auto_increment	unique id for each
title	varchar(50)	unique, not null	title of the article
content	varchar(255)	not null	content inside the text area
image_path	int	null	path of the image file uploaded
author_id	int	foreign key -> roles(id)	identify author's id
category_id	int	foreign key -> categories(id)	news categories
district_id	int	foreign key -> districts(id)	district of the news
status	enum ('pending','approved', 'rejected')	default 'pending'	status updated when article approved/rejected by mod/admin
is_local_voice	boolean	default 1 for readers	whether article written by readers or not
created_at	datetime	default current_timestamp	timestamp when article is submitted

4.4 Functional and behavioural model

4.4.1 Context diagram (0-level diagram)



4.4.2 First level diagram



5. TEST CASE

Login/register module	
Test case: 1	Test case date: 10-09-2025
Test title: login/register module	
Description: In this test login / register module will be tested	
Precondition: Login/ Register must work	

Test id	Test step	Test data	Expected result	Actual result	status
1	Navigate to reader home page	Login activity	Home page can be accessed	Same as expected	Pass
2	Login	False password	Login successful	Incorrect password	fail
3	Navigate to admin dashboard	Admin credentials	Admin dashboard	Same as expected	pass

Add user module	
Test case: 1	Test case date: 15-09-2025
Test title: add user module	
Description: In this test add user module will be tested	
Precondition: User must get registered	

Test id	Test step	Test data	Expected result	Actual result	status
1	Admin must be able to add a journalist	Journalist Credentials	User should get registered	Same as expected	Pass
2	Moderator must be able to add another moderator	moderator credentials	User should get registered	Same as expected	Pass

Create module	
Test case: 1	Test case date: 17-09-2025
Test title: create module	
Description: In this test login / register module will be tested	
Precondition: Articles should get created	

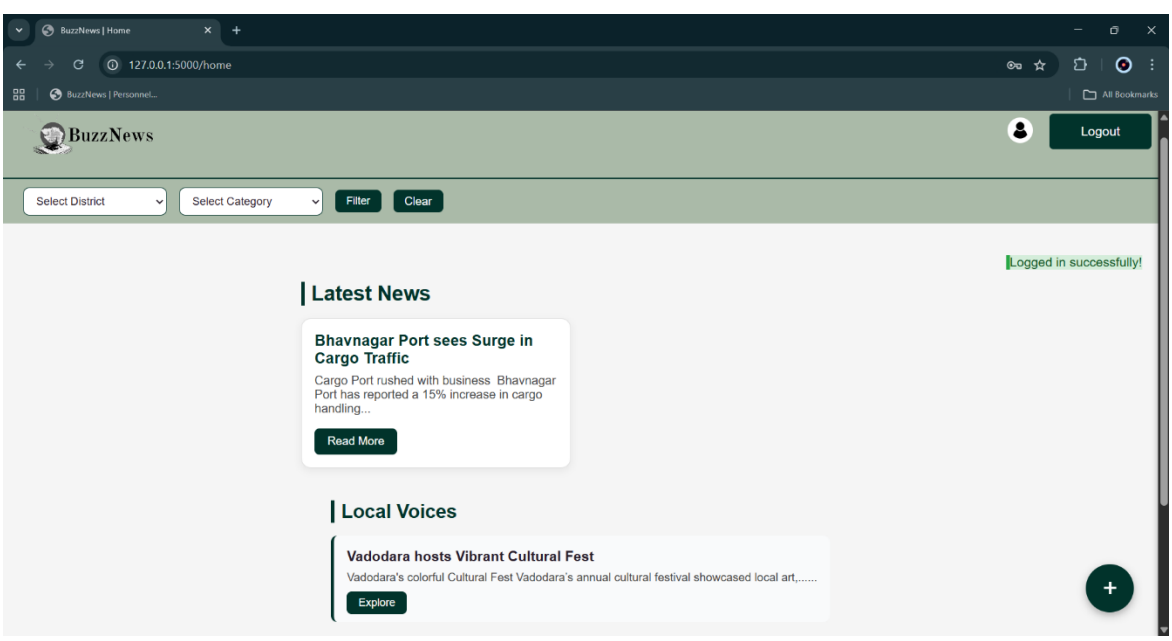
Test id	Test step	Test data	Expected result	Actual result	status

1	Publishing local voice article	Created article by reader	Successful publishing of article after approval	Same as expected	Pass
2	Draft an article	Journalist article	Article saved as draft for later	'draft' status not successful	fail
3	Rejecting an article	Journalist article	Article rejected by admin with a reason	Same as expected	pass

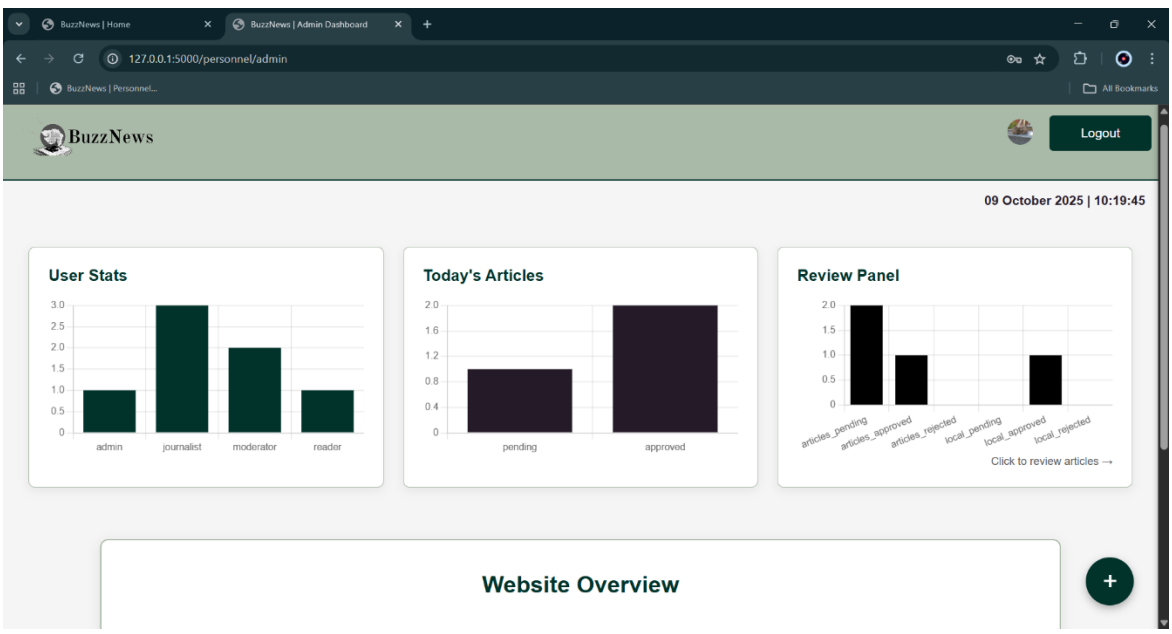
6. SCREEN SHOTS

- Reader side

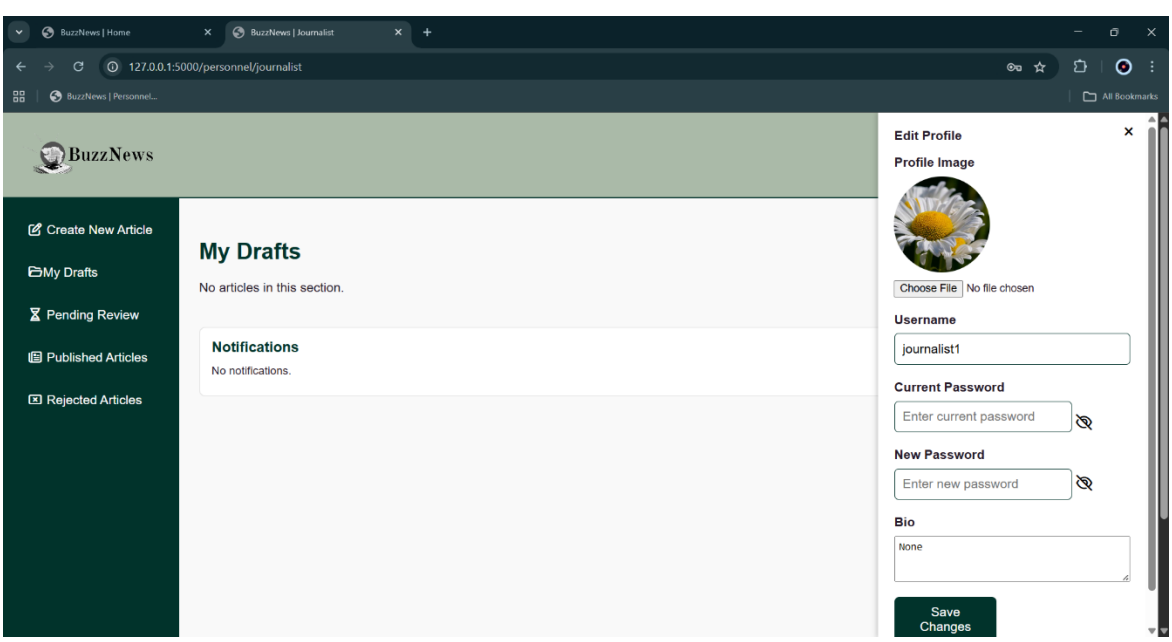
Home screen



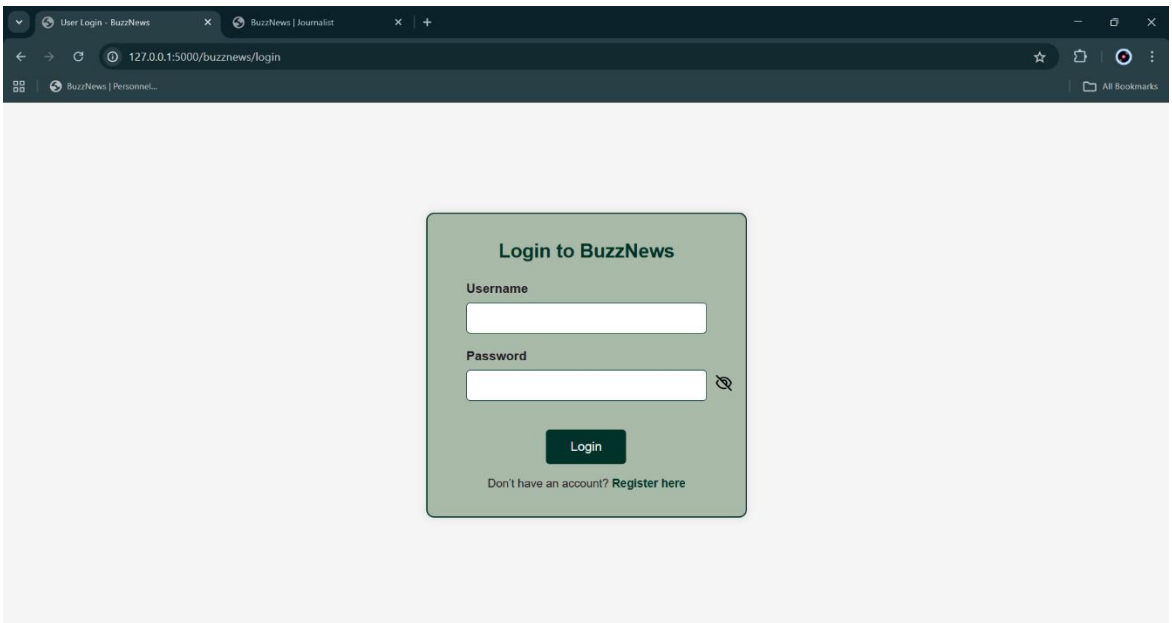
- Admin side
  Dashboard



- Journalist side
  Dashboard



- Login profile



7. Code explanation

1. route for user login/authentication

```
@app.route('/login', methods=['GET', 'POST'])
def login():
if request.method == 'POST':
username = request.form['username']
password = request.form['password']
cursor = connection.cursor(pymysql.cursors.DictCursor)
cursor.execute("SELECT * FROM users WHERE
username=%s", (username,))
user = cursor.fetchone()
```

```
if user and user['password'] == password:
session['user_id'] = user['id']
session['role_id'] = user['role_id']
flash('Login successful!', 'success')
return redirect(url_for('index'))
else:
flash('Invalid username or password', 'danger')
return render_template('login.html')
```

Explanation:
This route authenticates users. On POST, it fetches credentials from the form and checks them against the database. If matched, it stores user_id and role_id in the Flask session, which keeps the user logged in during their visit. If authentication fails, a flash message is shown. On GET, it simply renders the login page.

2. route for article creation

```
@app.route('/article/create', methods=['GET', 'POST'])
def create_article():
  if request.method == 'POST':
    title = request.form['title']
    content = request.form['content']
    image = request.files['image']

    image_path = None
    if image and image.filename:
      image_path = 'static/uploads/' + image.filename
      image.save(image_path)
```

```
    cursor = connection.cursor()
    cursor.execute("""
       INSERT INTO articles (title, content, image_path, author_id, status)
       VALUES (%s, %s, %s, %s, %s)
    """, (title, content, image_path, session['user_id'], 'pending'))
    connection.commit()

    flash('Article submitted for review!', 'success')
    return redirect(url_for('index'))

return render_template('create_article.html')
```

Explanation:

This route allows journalists to create new articles. It handles both text and image uploads. Images are stored in the static/uploads/ directory, and the article is inserted into the database with a default status of pending. This ensures that admins can moderate the article before publishing.

3. admin route (add journalist/ moderator)

```
@app.route('/admin/add_user', methods=['GET', 'POST'])
def add_user():
    if 'role_id' not in session or session['role_id'] != 3:
        return redirect(url_for('login'))  # only admin can access

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        role_id = request.form['role_id']
        district_id = request.form['district_id']

        cursor = connection.cursor()
        cursor.execute("""
            INSERT INTO users (username, password, role_id,
district_id)
            VALUES (%s, %s, %s, %s)
        """, (username, password, role_id, district_id))
```

```
        connection.commit()

        flash('New user added successfully!', 'success')
        return redirect(url_for('admin_dashboard'))

    return render_template('admin_add_user.html')
```

Explanation:

Only admins (role_id = 3) can access this route. It allows adding new users (journalists, moderators, readers) directly from the admin panel. Once submitted, the data is inserted into the users table. This feature supports multi-role access management.

4. local voice submission (for user)

```
@app.route('/local_voice', methods=['POST'])
def local_voice():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    content = request.form['content']
    cursor = connection.cursor()
    cursor.execute("""
        INSERT INTO local_voices (user_id, content, status)
        VALUES (%s, %s, %s)
    """, (session['user_id'], content, 'pending'))
    connection.commit()

    flash('Your local news has been submitted for review.', 'info')
```

```
    return redirect(url_for('index'))
```

Explanation:

This route allows logged-in users to submit short news snippets ("Local Voices"). Submissions are saved with a default status of pending, allowing admins to review them before publishing — supporting citizen journalism while maintaining content quality.

5. Home route

```
@app.route('/home', methods=['GET','POST'])
def home():
   if 'user_id' not in session:
      return redirect(url_for('user_login'))

   user_id = session.get("user_id")
   connection = get_db_connection()
   cursor = connection.cursor(pymysql.cursors.DictCursor)
   cursor.execute("SELECT * FROM users WHERE id=%s",
(user_id,))
   user = cursor.fetchone()

   district_id = request.args.get("district_id")
   category_id = request.args.get("category_id")
   search_query = request.args.get("search_query")

   filters_applied = bool(district_id or category_id or search_query)

   # ARTICLES
   article_query = "SELECT * FROM articles WHERE is_local_voice
= 0 AND status = 'approved'"
   article_params = []

   if district_id and district_id.strip():
      article_query += " AND district_id = %s"
```

```
        article_params.append(district_id)

    if category_id and category_id.strip():
        article_query += " AND category_id = %s"
        article_params.append(category_id)

    if search_query and search_query.strip():
       article_query+= " AND (title LIKE %s OR content LIKE %s)"
        like_pattern = f"%{search_query}%"
        article_params.extend([like_pattern, like_pattern])

    article_query += " ORDER BY submit_time DESC LIMIT 20"

    cursor.execute(article_query, tuple(article_params))
    articles = cursor.fetchall()

    # --------- LOCAL VOICES ---------
   lv_query = "SELECT * FROM articles WHERE is_local_voice = 1
AND status = 'approved'"
    lv_params = []

    if district_id and district_id.strip():
        lv_query += " AND district_id = %s"
        lv_params.append(district_id)

    if category_id and category_id.strip():
        lv_query += " AND category_id = %s"
        lv_params.append(category_id)

    if search_query and search_query.strip():
        lv_query += " AND (title LIKE %s OR content LIKE %s)"
        like_pattern = f"%{search_query}%"
        lv_params.extend([like_pattern, like_pattern])

    lv_query += " ORDER BY submit_time DESC LIMIT 6"
    cursor.execute(lv_query, tuple(lv_params))
    local_voices = cursor.fetchall()

    # --------- DROPDOWNS ---------
```

```
    cursor.execute("SELECT id, name FROM districts ORDER BY
name")
    districts = cursor.fetchall()

    cursor.execute("SELECT id, name FROM categories ORDER BY
name")
    categories = cursor.fetchall()

    cursor.close()
    connection.close()

    return render_template(
        "home.html",
        user=user,
        articles=articles,
        local_voices=local_voices,
        districts=districts,
        categories=categories,
        selected_district=district_id,
        selected_category=category_id,
        search_query=search_query or "",
        filters_applied=filters_applied)
```

Explanation:

This is the homepage route. It fetches only approved articles, optionally filtered by district. This allows users to browse general or district-specific news dynamically. The articles are then rendered on the homepage using the index.html template.

6. Admin Route

```
@app.route('/personnel/admin', methods=['GET'])
def admin_dashboard():
    if session.get('role') != 'admin':
        flash("Access denied.", "danger")
        return redirect(url_for('personnel_login'))
```

```
    # Fetch data
    user_stats_dict = get_user_stats()
    article_sitemap_list = get_todays_articles()
    review_panel_dict = get_review_panel_counts()
    website_stats = get_website_stats()

    user_stats = [{"role_name": k, "total": v} for k, v in
user_stats_dict.items()]
    review_panel = [{"status": k, "count": v} for k, v in
review_panel_dict.items()]

    # Fetch current user & districts
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM users WHERE id=%s",
(session['user_id'],))
    user = cursor.fetchone()
    cursor.close()
    connection.close()
    districts = Districts.query.all()

    return render_template(
        "admin_dashboard.html",
        districts=districts,
        user=user,
        user_stats=user_stats,
        article_sitemap=article_sitemap_list,
        review_panel=review_panel,
        website_stats=website_stats
    )
```

Explanation:

This is admin dashboard route. It fetches statistical data of the website from insights file. The data is then displayed as a visual representation of the activities' data as a bar graph.

8. CONCLUSION

The **BuzzNews** web portal successfully demonstrates the integration of modern web technologies to create a functional, user-friendly, and dynamic news platform. The project provides a structured system where administrators, moderators, journalists, and readers can seamlessly interact. From article creation and moderation to local voice contributions and filtered browsing, the platform ensures smooth content flow and community engagement.

The use of **Flask** for backend logic, **Bootstrap** and custom CSS for responsive design, and a well-structured database ensures both scalability and maintainability. Secure login mechanisms, role-based access, and clean UI/UX enhance the overall reliability of the system.

This project not only fulfills its intended objectives but also lays a strong foundation for future enhancements such as real-time notifications, advanced analytics, and multilingual support. Overall, **BuzzNews** is a practical and efficient solution for a state-level news aggregation and publishing platform.