

News Tracker Application

**St. Joseph's College of Engineering and
Technology Thanjavur-613403.**

Project Done by

Team ID:PNT2022TMID46931

1) K. Monika	Team Leader
2) V. Josephine Prasitha Mary	Team Member-1
3) A. Angelin Petrishiya	Team Member-2
4) S. Santhiya	Team Member-3

FACULTY MENTOR

Mrs. P. Suganya (AP/CSE)

S. No	Table of Content	Pg.no
1	Introduction	
1.1	Project Overview	
1.2	Purpose	
2	Literature Survey	
2.1	Existing Problem	
2.2	References	
2.3	Problem Statement Definition	
3	Ideation & Proposed Solution	
3.1	Empathy Map Canvas	
3.2	Ideation & Brainstorming	
3.3	Proposed Solution	
3.4	Problem Solution Fit	
4	Requirement Analysis	
4.1	Functional Requirement	
4.2	Non-Functional Requirement	
5	Project Design	
5.1	Data Flow Diagram	
5.2	Solution & Technical Architecture	
5.3	User Stories	
6	Project Planning & Scheduling	
6.1	Sprint Planning & Estimation	
6.2	Sprint Delivery Schedule	
7	Coding & Solution	
7.1	Feature 1	
7.2	Feature 2	
7.3	Database Schema (if Applicable)	

8	Testing	
8.1	Test Cases	
8.2	User Acceptance Testing	
9	Results	
9.1	Performance Metrics	
10	Advantages & Disadvantages	
11	Conclusion	
12	Future Scope	
13	Appendix	
13.1	Source code	
13.2	GitHub & Project Demo Link	

PROJECT REPORT

1.Introduction:

The development of portal for web based newspaper generally means creating a website in which the management of all news item sent by crowd about any type of news & activities are done by the administrator where all people (viewers) can view and know all the relevant information about the knowledge which they seek. This project is about the designing of a newspaper which displays the news which a normal person want to show. This portal is designed by using HTML, Python , Flask, IBM DB2.

The portal has basically three user parts where one is registered user (authentication required) who can view, add comment can have general discussion with another user and another is administrator (has an authentication) who will manage or control the website and other user (no authentication required) can only view and search. The website consists of basic pages from which the user can view and know the relevant information like history, upcoming. In other case, the administrator manages all the relevant actions for which the users can view properly and also make reports.

1.1 Project Overview:

The main objective of the project is to provide people a web application through which people can access all types of news and information. Through this application any user can gain technical knowledge of the world and its surrounding with just one click ahead. User does not have to visit multiple sites for different related information. All information is going to be in one place.

1.2 Purpose:

The purpose is to develop an application, which will eliminate the problems faced in the current scenario. This application will provide all the information and news related to cyber security, E=sport, Science, and Technology that are in trend in one place. So, it will save time and efforts for the users by making it more efficient. Using this application will terminate the possibility of information redundancy.

2.Literature Survey

2.1 Existing Problem:

Different type of newspaper will be available from all around the world in different languages with this user will be able to get news from all around the world. Short News: News will be displayed in short format with title, image and little description in list view. It will help user to access required news faster. Search Option: User will be able to search from not only one source but many different sources available within API. Favourites / Offline Reading: News can be added as

favourites which will automatically will be saved for offline reading. Sharing: User will be able to share news easily on social media. User was allowed to use this application in his smartphone and screenshots were taken as a result for this study. First User need to Sign Up in order to access the application which provides security for this application. Also predicted user error handling with pop-up messaging was done before this experiment like entering invalid data in fields, not selecting a field before clicking on action button.

2.2 References:

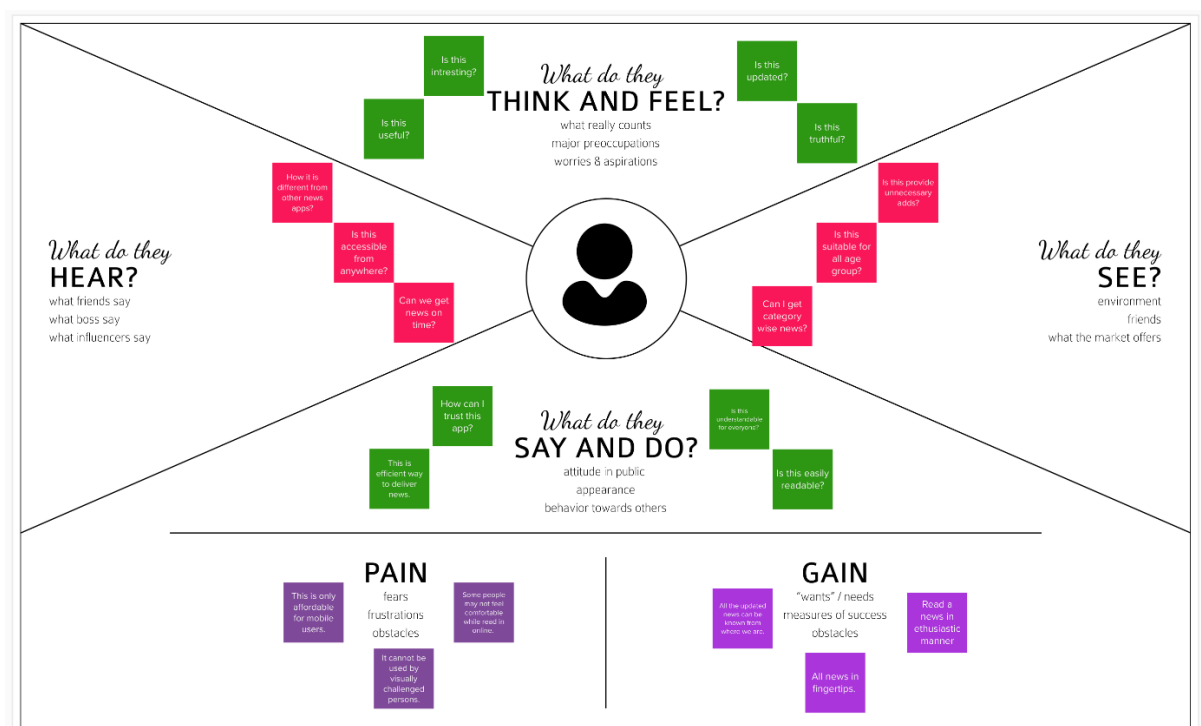
1. Ofcom, *News consumption in the UK*, Public report (2014).
2. Pew Research Centre, *The Future of Mobile News*, Public report (2012).
3. Reuters Institute, *Tracking the future of news*, Public Report (2014)
4. Billsus. D& Pazzani, M. A hybrid user model for news story classification. Springer Vienna (1999), 99-108.
5. Billsus, D& Pazzani, M. Adaptive news access. In *The adaptive web*. Springer Berlin Heidelberg (2007).
6. Carreira, R., Crato, J. M., Gonçalves, D., & Jorge, J. A. Evaluating adaptive user profiles for news classification. In *ACM IUI 2004*.

2.3 Problem Statement Definition:

The customer needs a way to get relevant news based on his choices so that the customer does not have to spend a lot of time on searching news. News is filled with ads and spams annoys and irritates the customer and affects the user experience. Market is full of news with all categories. Customers are not interested in all the categories and will be more interested with their personal choices. Traditional way of tracking news is slow and obsolete. User needs a new innovative application to.

3. Ideation & Proposed Solution

3.1 Empathy map Canvas:



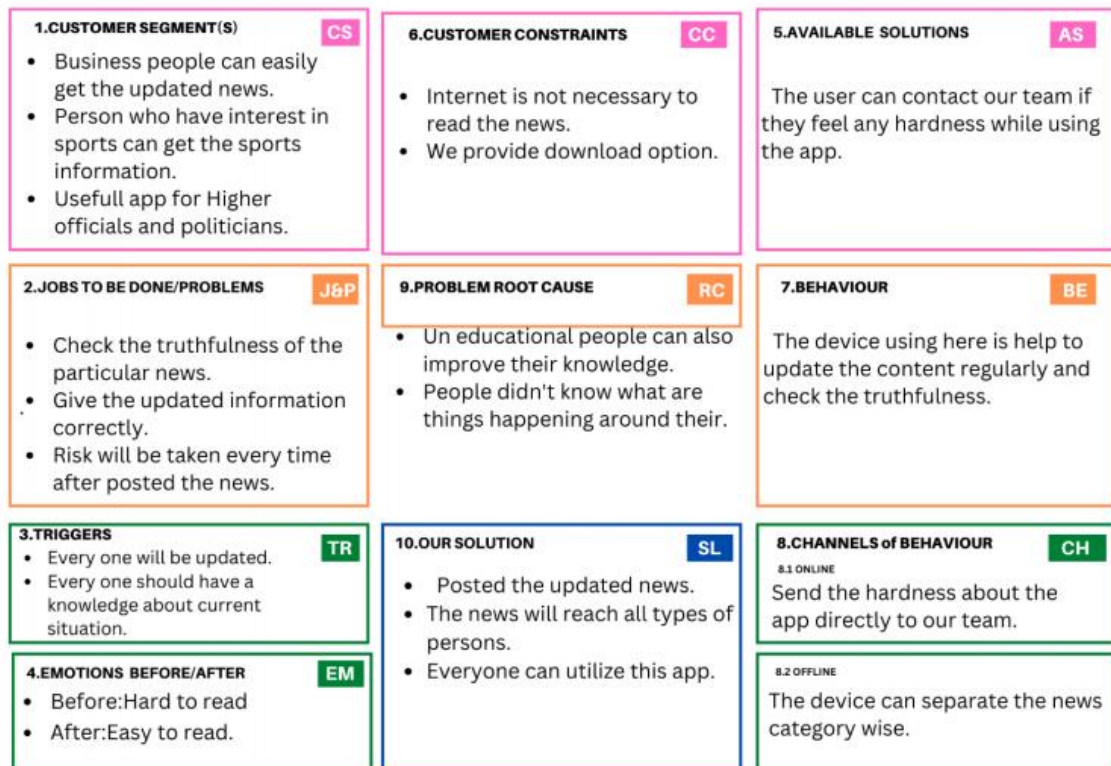
3.2 Ideation & Brain Stroming:



3.3 Proposed Solution:

When user want to get a news at any place and feel hard to read news papers, magazines. They don't have enough time to read news papers. In this Application user can directly read news in online or offline. User can categorize their favourites. In this application we provide Audio form of news also and we enable the option like comment. User can comment their own opinion. The major goal of our project is change our subscriber into a best reader. The news will reached to each and every one.

3.4 Proposed Solution fit:



4.Requirement Analysis

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email or Messenger.
FR-3	Subscription	As a user, can subscribe their favourite/wishing channels.

FR-4	Comment Box	User can comment their own thoughts and wish by the help of comment box and share his emotios.
FR-5	Download	In this App user want to read news in offline means they can download it through data.

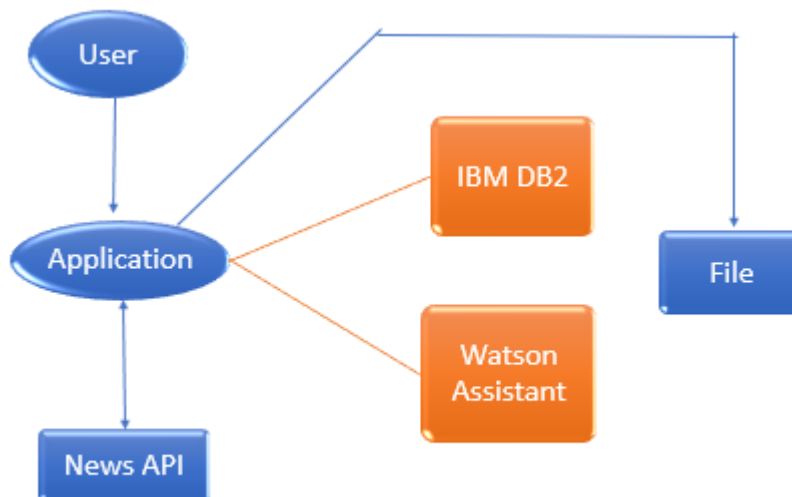
4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

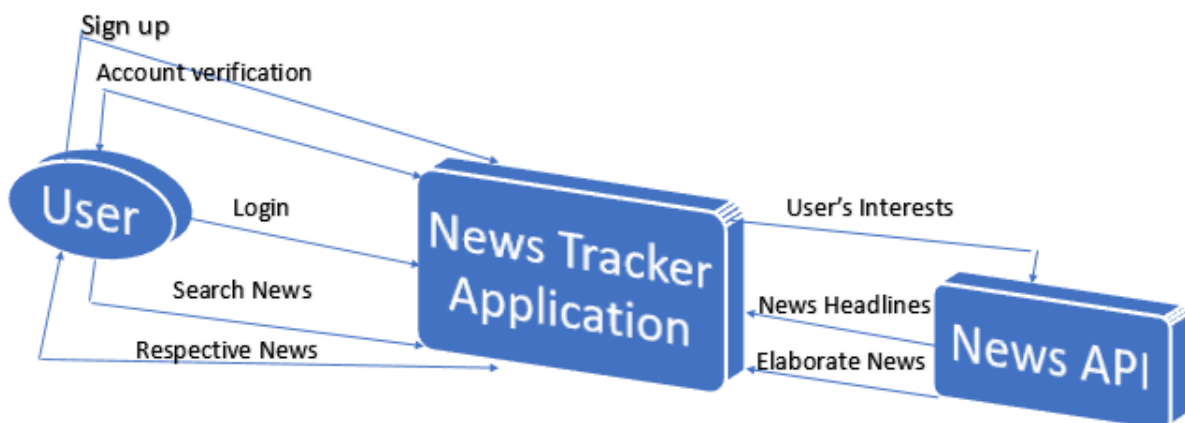
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	This app is very usefull for everyone because we provide this with simple&understandable language.
NFR-2	Security	We will keep user's personal details very secure.
NFR-3	Reliability	We only provide truthfull and real contents. We will not modify the news for the intrest of the user.
NFR-4	Performance	We provide clear user interface and the app will perform very good manner not irritating the user with low performance.

5. Project Design

5.1 Data Flow Diagram:



5.2 Solution & Technical Architecture:



5.3 User Stories:

User Type		Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)		Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard.	High	Sprint-1
			USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm.	High	Sprint-1
		Login	USN-3	As a user, I can log into the application by entering email & password	I can access my dashboard.	High	Sprint-2
		Home Page	USN-4	As a user I can view the headlines of the news that interest me	I can read the news elaborately by clicking on the headlines.	High	Sprint-2
			USN-5	As a user I can view the elaborate content of the headlines	I can download that to read it when I am free even in offline mode.	Medium	Sprint-3
			USN-6	As a user I can search a news I want	I can read the news elaborately.	High	Sprint-4
Customer Care Executive		Chatbot	USN-6	As a user I can solve my doubts about the application with the help of the chatbot .If the doubt is not resolved the chatbot guides us on how to contact the customer care executive	I can contact the customer care executive if needed	Medium	Sprint-3
Administrator		About us	USN-7	As a user I can view about the application and the developers of the application	I can be able to see their other applications	low	Sprint-4

6. Project Planning & Scheduling

When it comes to managing projects, it can be hard to get everyone on the same page. With multiple moving parts, different deliverables, and cross-departmental collaboration, sometimes an initial project meeting just isn't enough. Project design is an opportunity to align ideas, processes, and deliverables. It's an early phase in the project lifecycle and often comes before a project plan or charter. This is because it focuses on the project overview rather than the specific details. Visual aids such as flowcharts, Gantt charts, and timelines are often used to help paint a picture for project stakeholders in this early step.

6.1 Sprint Planning & Estimation:

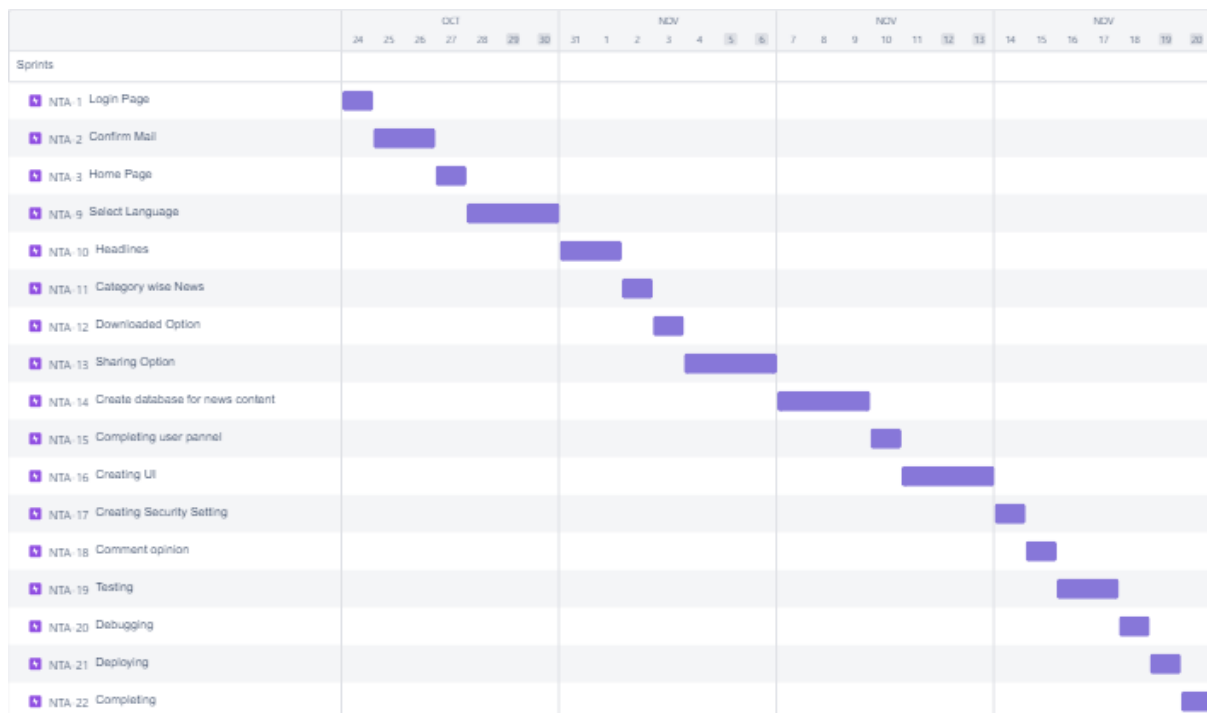
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, ✓ Register. ✓ Entering mail, Password. ✓ Confirming Password.	10	High	K. Monika
Sprint-1		USN-2	As a user, ✓ Receive email.	10	High	V. Josephine Prasitha Mary, S.Santhiya
Sprint-1	Login	USN-3	As a user, ✓ Login by entering email & password.	15	High	A.Angelin Petrishiya
Sprint-2	Input Necessary Details	USN-4	As a user, ✓ Search the news ✓ Category wise	15	High	K. Monika
Sprint-2	Data Pre-processing	USN-5	Application, ✓ Give search related details	15	High	A.Angelin Petrishiya, S. Santhiya

Sprint-3	Searching of news	USN-6	As a user, ✓ Search for the accurate news. ✓ Comment opinion	20	High	K. Monika
Sprint-3		USN-7	As a user, ✓ can get accurate news. ✓ Asking doubt frequently.	5	Medium	V. Josephine Prasitha Mary
Sprint-4	Review	USN-8	As a user, ✓ Can give feedback of the application.	20	High	K. Monika S. Santhiya

6.2 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	7 Days	24 Oct 2022	30 Oct 2022	20	30 Oct 2022
Sprint-2	20	7 Days	31 Oct 2022	06 Nov 2022	20	06 Nov 2022
Sprint-3	20	7 Days	07 Nov 2022	13 Nov 2022	20	13 Nov 2022
Sprint-4	20	7 Days	13 Nov 2022	20 Nov 2022	20	20 Nov 2022

6.3 Report From Jira:



7. Coding & Solutioning

7.1 Feature 1:

Coding:

```
from flask import Flask, render_template , request , redirect , session
import requests
import json
import ibm_db
```

```
conn_str='<IBM_API>'
conn = ibm_db.connect(conn_str,"")
```

```
url = "https://api.newscatcherapi.com/v2/search"
headers = {
    "x-api-key": "<NEWS_API>"
}
```

```
app = Flask(__name__, static_url_path='/static')
app.secret_key = 'nmc8so7c0no78ypw9o8b[np0'
```

```
@app.route("/db",methods=['GET'])
def db():
#   sql = "create table users (id integer not null GENERATED ALWAYS AS IDENTITY
(START WITH 1 INCREMENT BY 1),name varchar(500),email varchar(500),password
varchar(225),PRIMARY KEY (id));"
#   stmt = ibm_db.exec_immediate(conn, sql)
    return "ok"
```

```
@app.route("/",methods = ['GET'])
def index():
    querystring = {"q":"latest","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("index.html" ,data=json.loads(response.text))
```

```
@app.route("/sports",methods = ['GET'])
def index_sport():
    querystring = {"q":"sports","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("sports.html" ,data=json.loads(response.text))
```

```
@app.route("/international",methods = ['GET'])
def index_inter():
    querystring = {"q":"international","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("inter.html" ,data=json.loads(response.text))
```

```

@app.route("/entertainment",methods = ['GET'])
def index_enter():
    querystring = {"q":"entertainment","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("enter.html" ,data=json.loads(response.text))

@app.route("/search",methods = ['POST'])
def index_search():
    querystring = {"q":request.form['search'],'lang':'en',"sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("search.html" ,data=json.loads(response.text))

@app.route("/login",methods = ['GET'])
def index_login():
    return render_template("login.html")

@app.route("/signin",methods = ['POST'])
def index_signin():
    sql = "select * from users where email='"+request.form["email"]+"'"
    stmt = ibm_db.exec_immediate(conn, sql)
    data = []
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        data.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if(data):
        if(data[0]["PASSWORD"]==request.form["password"]):
            session["user"]=data[0]["ID"]
            session["name"]=data[0]["NAME"]
            session["email"]=data[0]["EMAIL"]
            return redirect("/user")
        else:
            return redirect("/login")
    else:
        return redirect("/login")

@app.route("/register",methods = ['GET'])
def index_register():
    return render_template("register.html")

@app.route("/signup",methods = ['POST'])
def index_signup():
    sql = "INSERT INTO users (name , email ,
password)values('"+request.form["name"]+"','"+request.form["email"]+"','"+request.form["pa
ssword"]+"')"
    stmt = ibm_db.exec_immediate(conn, sql)
    return redirect("/login")

```


USER

```
@app.route("/user/", methods = ['GET'])
def user():
    querystring = {"q": "latest", "lang": "en", "sort_by": "relevancy", "page": "1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("user/index.html", data=json.loads(response.text))

@app.route("/user/sports", methods = ['GET'])
def user_sport():
    querystring = {"q": "sports", "lang": "en", "sort_by": "relevancy", "page": "1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("user/sports.html", data=json.loads(response.text))

@app.route("/user/international", methods = ['GET'])
def user_inter():
    querystring = {"q": "international", "lang": "en", "sort_by": "relevancy", "page": "1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("user/inter.html", data=json.loads(response.text))

@app.route("/user/entertainment", methods = ['GET'])
def user_enter():
    querystring = {"q": "entertainment", "lang": "en", "sort_by": "relevancy", "page": "1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("user/enter.html", data=json.loads(response.text))

@app.route("/user/search", methods = ['POST'])
def user_search():
    querystring = {"q": request.form['search'], "lang": "en", "sort_by": "relevancy", "page": "1"}
    response = requests.request("GET", url, headers=headers, params=querystring)
    return render_template("user/search.html", data=json.loads(response.text))

@app.route("/user/logout", methods = ['GET'])
def user_logout():
    return redirect("/")

if __name__ == '__main__':
    app.run()
```

7.2 Feature 2:

Coding:

from flask import Flask, render_template , request , redirect

app = Flask(__name__, static_url_path='/static')

```
@app.route("/",methods = ['GET'])
def admin():
    return "Home Page"
```

```
if __name__ == '__main__':
    app.run(debug = True)
```

Coding:

```
from flask import Flask, render_template , request , redirect
import requests
import json
```

```
url = "https://api.newscatcherapi.com/v2/search"
```

```
headers = {
    "x-api-key": "API_KEY"
}
```

```
app = Flask(__name__, static_url_path='/static')
```

```
@app.route("/",methods = ['GET'])
def index():
    querystring =
    {"q":"latest","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
    params=querystring)
    print(json.loads(response.text)['articles'][0])
    return render_template("index.html"
    ,data=json.loads(response.text))
```

```
@app.route("/sports",methods = ['GET'])
def index_sport():
```

```

    querystring =
{"q":"sports","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    print(json.loads(response.text)['articles'][0])
    return render_template("sports.html"
,data=json.loads(response.text))

@app.route("/international",methods = ['GET'])
def index_inter():
    querystring =
{"q":"international","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    print(json.loads(response.text)['articles'][0])
    return render_template("inter.html" ,data=json.loads(response.text))





@app.route("/entertainment",methods = ['GET'])
def index_enter():
    querystring =
{"q":"entertainment","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    print(json.loads(response.text)['articles'][0])
    return render_template("enter.html"
,data=json.loads(response.text))

@app.route("/search",methods = ['POST'])
def index_search():
    querystring =
{"q":request.form['search',"lang":"en","sort_by":"relevancy","page":
"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    print(json.loads(response.text)['articles'][0])
    return render_template("search.html"
,data=json.loads(response.text))

```

```
if __name__ == '__main__':  
    app.run()
```

7.3 Database Schema(if Applicable)

Repositories						
Location						
Tokyo						
Search						
Create +						
<input type="checkbox"/>	Name	Image count	Namespace	Last updated		
^	<input type="checkbox"/>  nalayathiran jp.icr.io/nalayathiran/nalayathiran	1	nalayathiran	1 day ago		
Digest		Manifest type	Tags	Created	Size	Security status
 09cbac96d9cc		Docker	latest	1 day ago	450 MB	 3 issues
∨	<input type="checkbox"/>  assignment_4 jp.icr.io/assignment_4/assignment_4	1	assignment_4	15 days ago		
Items per page: 25		1-2 of 2 items			1	1 of 1 page

8. Testing

8.1 Test Cases:

1.Unit testing: It is a method by which individual units of source code, sets of one or more program modules collectively with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming a unit can be an entire module but is more commonly an individual function. In object-oriented programming a unit is an entire interface but could be an individual method. Unit test is created by programmers or by white box testers during the development process. Each test case is independent from the others: substitutes like method stubs, mock objects can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual to being formalized as part of build automation.

2.Integration testing: It is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing .

Program test: After the programs have been coded, compiled and carried out to working conditions, they must be independently tested

with the prepared test data. Any unwanted happening should be noted and debugged.

System Test: When the program test for each of the programs of the system is written and errors are removed then system test is complete. At this stage the test is done on actual data. The complete system is put into execution on the actual data. At every stage of the execution, the output of the system is studied. During the outcome analysis, it may be found that the outputs are not matching the estimated output of the system. In such situation, the bugs or errors in the particular programs are recognized and are fixed and further verified for the expected output. When it is confirmed that the system is running error-free, the users are called with their own real data so that the system could be presented running as per their requirements.

8.2 User Acceptance Testing

1. PURPOSE OF DOCUMENT:

The purpose of this document is to briefly explain the test coverage and open issues of the News Tracker Application project at the time of the release to User Acceptance Testing UAT.

2.DEFECT ANALYSIS:

This report shows the number of resolved or closed bugs at each severity level and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	1	2	1	9
Duplicate	1	0	0	0	1
External	1	0	0	1	2
Fixed	7	1	2	2	12
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	14	2	4	4	24

3.TEST CASE ANALYSIS:

This report shows the number of test cases that have passed failed and untested.

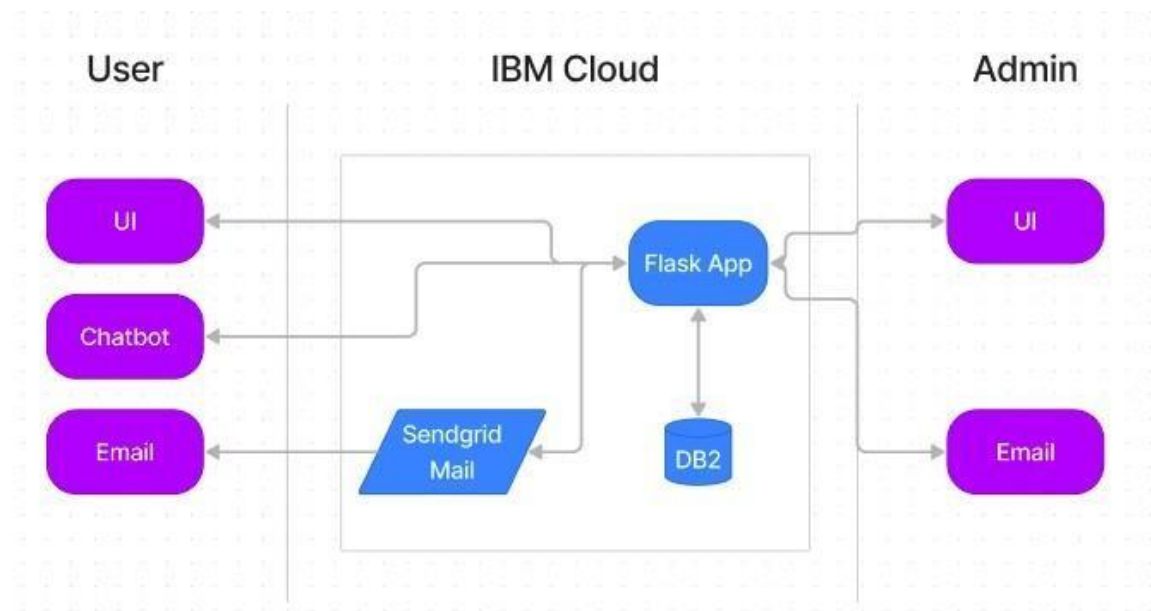
Section	Total Cases	Not Tested	Fail	Pass
Login Page	4	0	0	4
Registration Page	1	0	0	1
Home Page	5	0	0	5

9. Results

9.1 Performance Metrics:

These application performance metrics, commonly known as key performance indicators (KPIs), are a quantitative measure of how effectively the organization achieves the business objectives. Capturing the right metrics will give you a comprehensive report and powerful insights into ways to improve your application.

Technical Architecture:



Register

Name :

Email :

Password :

Register

Do you have an Account? [Click here](#)

2022 © InfoTimes. All Rights Reserved

InfoTimes



[Signin](#) / [Signup](#)

Login

Email :

Password :

Log In

Don't you have an Account? [Click here](#)

2022 © InfoTimes. All Rights Reserved



The Association of International Economic Strategy Was Established in Beijing to Facilitate Internet



Spirit Airlines drops 37 routes, hitting Florida the hardest



MY REWARDS INTERNATIONAL LIMITED : Stock Market News and Information



PXUS INTERNATIONAL INC. : Stock Market News and Information



Opinion: Canada identifies international students as 'ideal immigrants' but supports are lacking



Canada identifies international students as 'ideal immigrants' but supports are lacking



Canada identifies international students as 'ideal immigrants' but supports are lacking



Qual accredited as top vet school



Bumper profits fuel surge in dividends, buybacks at oil firms



Factbox: Bumper profits fuel surge in dividends, buybacks at oil firms



Factbox-Bumper profits fuel surge in dividends, buybacks at oil firms



Oil Profits Fuel Surge in Dividends, Buybacks



Morning: Breaking Down the Win on 'SNF'



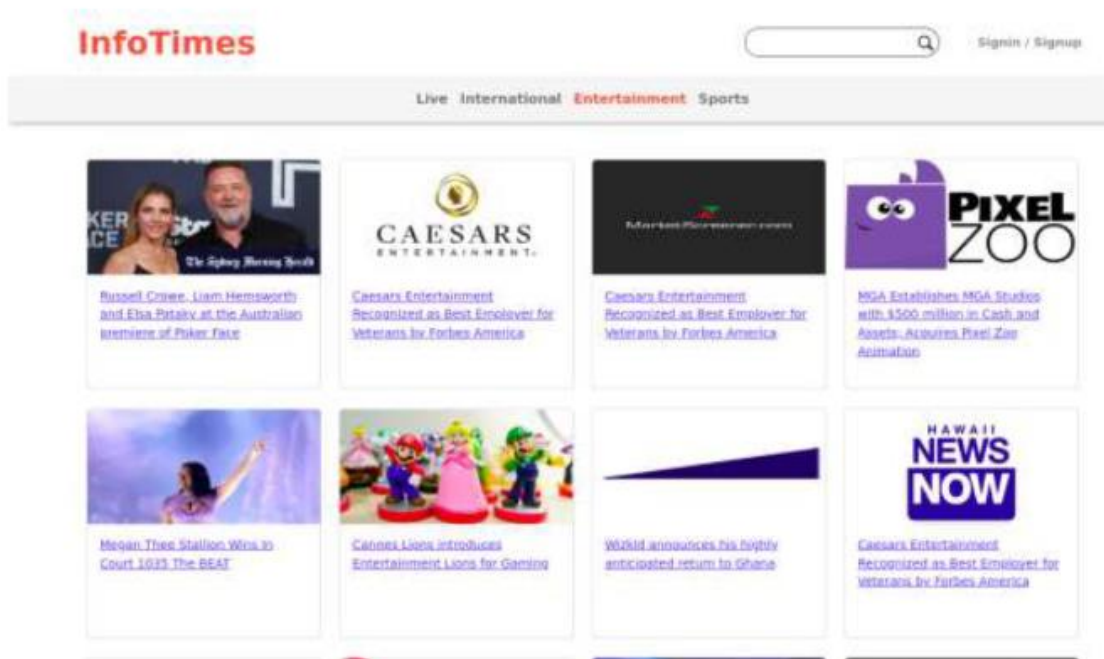
Off the Field: 49ers' Offense Cheer on Golden State Warriors



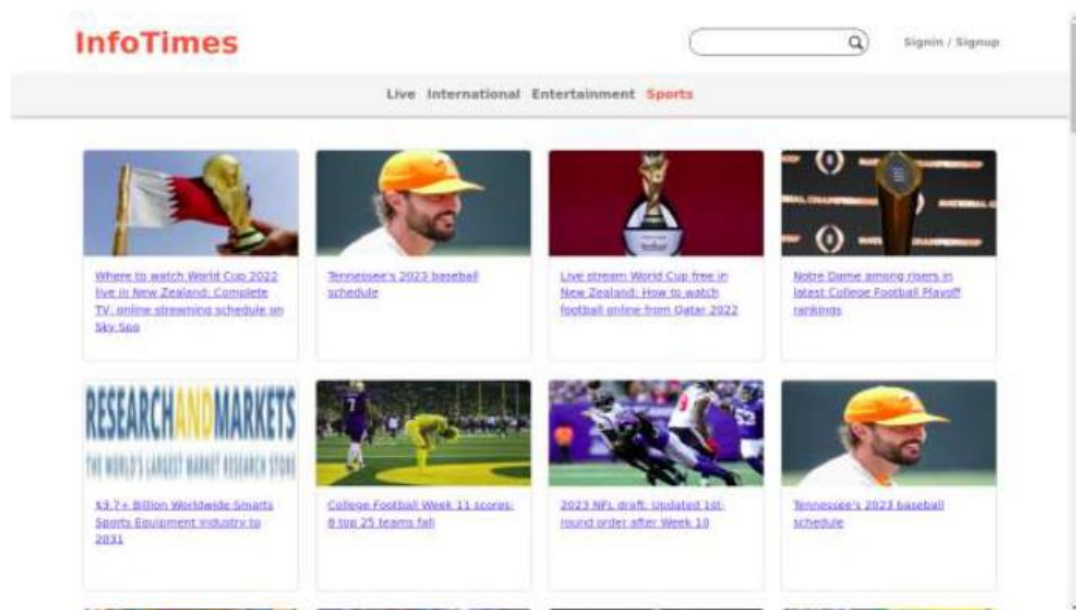
CBS Board Exam Date Sheet 2023 LIVE: CBSE Class 10, 12 Time Tables To Be Released Today, Latest Upd



49ers Activate Four Players from Injured Reserve: Place Verrett on IR



Older audiences may not access digital platforms.



10. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- Real-time application is allowed and has live layout.
- It has a Built-in support for Google Platform.
- News accuracy of specialised API.
- Authenticated users are allowed.
- Up to date and daily information is available.

DISADVANTAGES:

- Require data/wifi to get online.
- Companies not making as much money due to free reading for audiences.
- News spreads quicker online - people find out news before they should.
- Lose money - can't get people to pay for digital.

11. Conclusion:

The developed algorithm personalize news feeds saves the user time and give him only interesting articles, news etc. By single click.

12.Future Scope:

- We are in a process of developing a algorithm that will help the user to read the new postings and news from his recent data sources.
- In pandemic situations, offline news won't be delivered to anyone, in those time these news apps are the most suitable.
- In the future, we are going to develop a new categories according to their user locations.

13. Appendix:

Source Code:

```
from flask import Flask, render_template , request , redirect , session
```

```
import requests
```

```
import json
```

```
import ibm_db
```

```
conn_str='<IBM_API>'
```

```
conn = ibm_db.connect(conn_str,"")
```

```
url = "https://api.newscatcherapi.com/v2/search"
```

```
headers = {
```

```
    "x-api-key": "<NEWS_API>"
```

```
}
```

```
app = Flask(__name__, static_url_path='/static')
```

```
app.secret_key = 'nmc8so7c0no78ypw9o8b[np0'
```

```
@app.route("/db",methods=['GET'])
```

```
def db():
```

```
#     sql = "create table users (id integer not null GENERATED  
ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY  
1),name          varchar(500),email          varchar(500),password  
varchar(225),PRIMARY KEY (id));"
```

```
#     stmt = ibm_db.exec_immediate(conn, sql)
```

```
     return "ok"
```

```
@app.route("/",methods = ['GET'])
```

```
def index():
```

```
    querystring                                     =
```

```
{ "q": "latest", "lang": "en", "sort_by": "relevancy", "page": "1" }
```

```
    response = requests.request("GET", url, headers=headers,  
params=querystring)
```

```
    return                                     render_template("index.html"  
,data=json.loads(response.text))
```

```
@app.route("/sports",methods = ['GET'])
```

```
def index_sport():
```

```
    querystring                                     =
```

```
{ "q": "sports", "lang": "en", "sort_by": "relevancy", "page": "1" }
```

```
response = requests.request("GET", url, headers=headers,  
params=querystring)
```

```
return render_template("sports.html"  
,data=json.loads(response.text))
```

```
@app.route("/international",methods = ['GET'])
```

```
def index_inter():
```

```
    querystring =  
    {"q":"international","lang":"en","sort_by":"relevancy","page":"1"}  
    response = requests.request("GET", url, headers=headers,  
params=querystring)  
    return render_template("inter.html",data=json.loads(response.text))
```

```
@app.route("/entertainment",methods = ['GET'])
```

```
def index_enter():
```

```
    querystring =  
    {"q":"entertainment","lang":"en","sort_by":"relevancy","page":"1"}  
    response = requests.request("GET", url, headers=headers,  
params=querystring)  
    return render_template("enter.html"  
,data=json.loads(response.text))
```

```
@app.route("/search",methods = ['POST'])
```

```
def index_search():
```

```

        querystring =
{"q":request.form['search'], "lang":"en", "sort_by":"relevancy", "page":
"1"}

    response = requests.request("GET", url, headers=headers,
params=querystring)

    return render_template("search.html"
,data=json.loads(response.text))

```

```

@app.route("/login", methods = ['GET'])
def index_login():
    return render_template("login.html")

```

```

@app.route("/signin", methods = ['POST'])
def index_signin():
    sql = "select * from users where email='"+request.form["email"]+"'"
    stmt = ibm_db.exec_immediate(conn, sql)
    data = []
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        data.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if(data):
        if(data[0]["PASSWORD"]==request.form["password"]):
            session["user"]=data[0]["ID"]
            session["name"]=data[0]["NAME"]

```



```

        session["email"]=data[0]["EMAIL"]
        return redirect("/user")
    else:
        return redirect("/login")
    else:
        return redirect("/login")

@app.route("/register",methods = ['GET'])
def index_register():
    return render_template("register.html")

@app.route("/signup",methods = ['POST'])
def index_signup():
    sql = "INSERT INTO users (name , email ,
password)values('"+request.form["name"]+"','"+request.form["email"
]+"','"+request.form["password"]+"')"
    stmt = ibm_db.exec_immediate(conn, sql)
    return redirect("/login")

# USER

@app.route("/user/",methods = ['GET'])
def user():

```

```

    querystring =
{"q":"latest","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    return render_template("user/index.html"
,data=json.loads(response.text))

```

```

@app.route("/user/sports",methods = ['GET'])
def user_sport():
    querystring =
{"q":"sports","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    return render_template("user/sports.html"
,data=json.loads(response.text))

```

```

@app.route("/user/international",methods = ['GET'])
def user_inter():
    querystring =
{"q":"international","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    return render_template("user/inter.html"
,data=json.loads(response.text))

```

```

@app.route("/user/entertainment",methods = ['GET'])
def user_enter():
    querystring = {
        "q":"entertainment","lang":"en","sort_by":"relevancy","page":"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    return render_template("user/enter.html",
,data=json.loads(response.text))

```

```

@app.route("/user/search",methods = ['POST'])
def user_search():
    querystring = {
        "q":request.form['search',"lang":"en","sort_by":"relevancy","page":
"1"}
    response = requests.request("GET", url, headers=headers,
params=querystring)
    return render_template("user/search.html",
,data=json.loads(response.text))

```

```

@app.route("/user/logout",methods = ['GET'])
def user_logout():
    return redirect("/")

```

```

if __name__ == '__main__':

```

app.run()

13.1 GitHub Link:

<https://github.com/IBM-EPBL/IBM-Project-44380-1664355664>

13.2 Project Demo Link:

<https://youtu.be/I5iNTl45Py4>