NEWS TRACKER APPLICATION

IBM NALAIYATHIRAN

PROJECT REPORT

SUBMITTED BY

TEAM_ID:PNT2022TMID04862

SANCHANA SHREE KI	**737819ITR078**
SAMYUKTHA C	**737819ITR077**
SANDHIYA G	**737819ITR079**
SANDHIYA S	**737819ITR080**

TABLE OF CONTENTS

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 requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 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 & SOLUTIONING (Explain the features added in the project along with code)**
   - 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

CHAPTER 1

1. INTRODUCTION:

Nowadays, the news media are subject to the social scrutiny because of the lack of credibility,the manipulative media, the misinformation campaigns, and the propagation of fake news point out that the main difference between fake news and true news relies on lack editorial norms and processes that ensure the accuracy and credibility of the information.Thus, to arrange a way that allows guaranteeing these editorial processes (or at least part of them) can suppose a big step in the fight against the above-mentioned issues.To track the evolution of the news reports and the relevant data and information it contains as they change over time, and therefore to trace how the related news evolves, constitute other instruments to face the previous issues. This is not only useful for end readers but for fact-checking agencies and those tools that perform automatic indexing and extraction of relevant information of news. Once the information has been verified and fact-checked, these tools need a way to guarantee that the extracted data have not changed.

1.1 PROJECT OVERVIEW

The main focus of this application is to connect news articles from all around the world and deliver it to user as fast as possible in best visualize way.The database used in this news tracker application is DB2. The work flow of the project is:

- Create a user interface to interact with the application.
- Peoples can use the application by logging in by giving their details.
- Integrate the application with news APIs and store the data in the database.
- The database will have all the details and the user can search the news by

using a search bar.

1.2 PURPOSE

As our lives are very busy these days, we often feel we need more than 24 hrs. a day to cope up with everything we have in our schedule. Well, that's not possible but reducing the time by changing the conventional method of reading news can help. Just tell us what market news you're interested in and get a quick peek for the day. Only read what you feel is relevant and save your time. This app helps you to query for all information about Indices, Commodities, Currencies, Future Rates, Bonds, etc.… as on official websites.

CHAPTER 2

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

News is one of the primary source of gaining information about the actions and events that happen all around. It may be an event that happened in the past, happening now or going to happen in the future. In the present days where there is a rapid increase in the development and adaptability of technologies throughout all the demographic of people, it is necessary to provide news in such a way that it is interconnected with the current technological trends. As our lives are very busy these days, we often feel we need more than 24 hrs. a day to cope up with everything we have in our schedule. Well, that's not possible but reducing the time by changing the conventional method of reading news can help. Just tell us what market news you're interested in and get a quick peek for the day. Only read what you feel is relevant and save your time. This app helps you to query for all information about Indices, Commodities, Currencies, Future Rates, Bonds, etc.… as on official websites.

2.2 REFERENCES

S. NO	Paper Title	Author Name	Year	Methods Used	Link
1	Exploring mobile news reading interactions for news app personalization	Marios Constantinides,John Dowell, David Johson, Sylvain Malacria	2018	1. Identification of news reader types 2.Interaction logging and classification study 3. Deployment and data collection 4. Adaptive UI	https://dl.acm.org/doi/10.1145/2785830.2785860
2.	Tailored News in the Palm of Your HAND: A Multi - Perspective Transparent Approach to News Recommen dation	Mozhgan Tavakolifar d Jon Atle Gull a , Kevin Almeroth	2018	1. Identification of news reader types 2.Interaction logging and classification study 3. Deployment and data collection 4. Adaptive UI	https://dl.acm.org/doi/10.1145/2487788.2487930
3.	An End-to end Weaklysuper vised News Aggregation Framework	Xijin Tang, Xiaohui Huang	2022	The framework combines Snorkelbased weaklysupervised classification, Latent Dirichlet Allocation (LDA) topic modeling, and topic signal detection model to classify and aggregate unlabeled.	https://link.springer.com/chapter/10.1007/978-981-19-3610-4_4
4.	Detection and Tracking in News Articles	Sagar Patel, Sanket Suthar, Sandip Patel, Neha Patel	2015	1. Preprocessing 2. Tokenization 3.Vector SpaceModel 4. Topic tracking	https://www.researchgate.net/publication/315657099_Topic_Detection_and_Tracking_in_News_Articles

2.3 PROBLEM STATEMENT DEFINITION

There are multiple news sharing apps used by single user are often spammed with notifications.There is also a lot of news with different categories which gets shared .A news sharing app wants to help users find relevant and important news easily.

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Student who is preparing for civil service examination.	Know about the current affairs.	There are so many news which are not related to that and also distract me to study.	It does not contain the news which are updated	morse
PS-2	Vikram ,a student who is interested in sports	Know about the results of the match	there is no news about the match	the sport which I like is not much more popular	unhappy
PS-3	Ram, a film director who is interested to take films	Search or read the short stories written by some little authors	This will available once in a week	Someone only interested	discouraged
PS-4	Sam, Politician who wants to become a mayor	Know about the local news around me	There is only a few news which are not enough for me	There are so many news are important than this news	down

CHAPTER 3

3. IDEATION AND PROPOSEDSOLUTION

3.1 EMPATHY MAP CANVAS

3.2 IDEATION

- By using media cloud we can collect news which is a massive datasets for storing news and we can use UI/UX design which can be created using HTML, CSS , Bootstrap, Java script, Flask and SQL-lite and we can deploy it using IBM cloud.

- There are so many feeds for collecting news for example cloud-tech RSS

cloud and also we can monitor the news using monitoring serves and create a app using web development and deploy it in the cloud and we can feed the news using cloud storage.

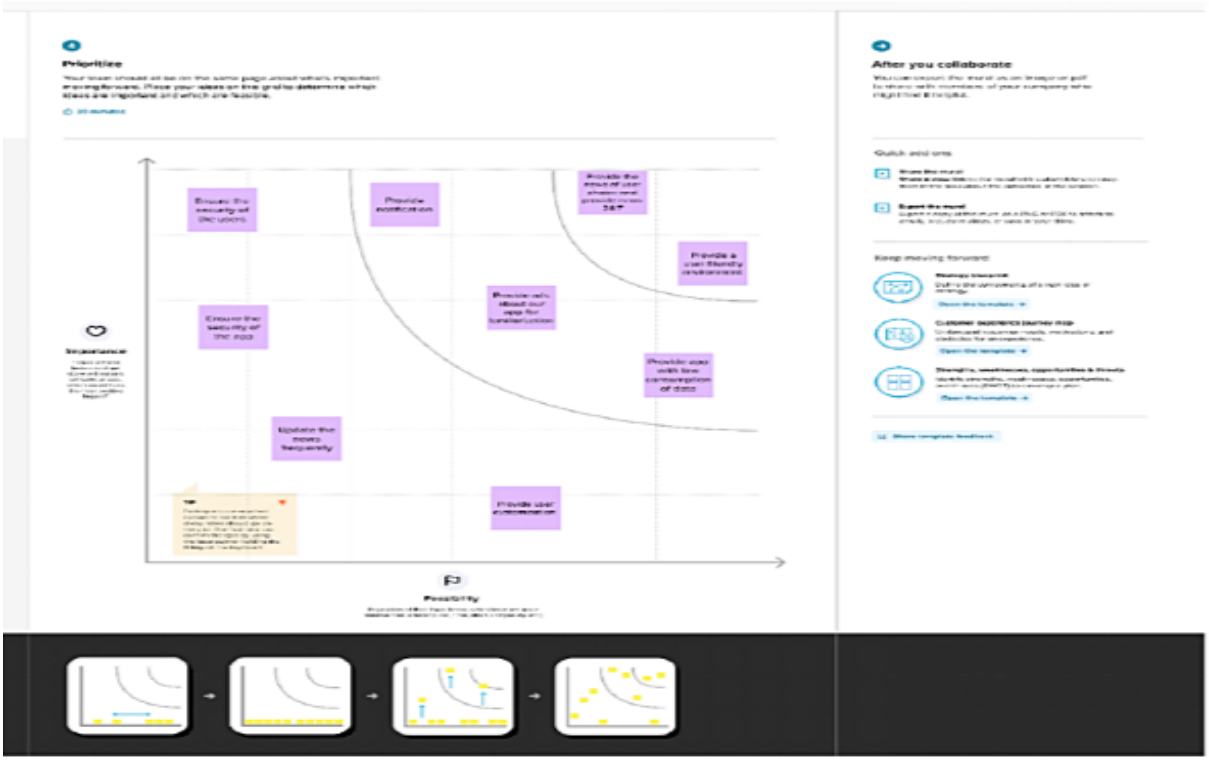
- News Tracker can be implement by creating Application using HTML, CSS, Bootstrap, Java script, Flask and for storing news we can use Cloud (Google, IBM, Microsoft).

BRAINSTORMING

Step 1:Team Gathering,Collaboration and Select the Problem Statement

Step2:Brainstorm,Idea Listing and Grouping

Step3:Idea Prioritization



3.3 PROPOSED SOLUTION

1. Problem Statement (Problem to be solved)

In this busy world Peoples do not have time to read newspaper. Some of the people do not like all the news. They only need the news what they are interested. And we cannot carry newspaper wherever we go.

2. Idea / Solution description

To develop a application which can provide a news with good content and to get a news frequently and to get a news everywhere by hosting this app in mobile phone and PC. And peoples can read the news based on their interest.

3. Novelty / Uniqueness

Many app provides a news which are interesting but our app focuses on providing a based on the user interest. The personal information of the users are safely maintained. And we try to provide a advertisement free application.

4. Social Impact / Customer Satisfaction

Users can provide a feedback and also give ratings for the app. In critical situation this app is very useful for reading news.

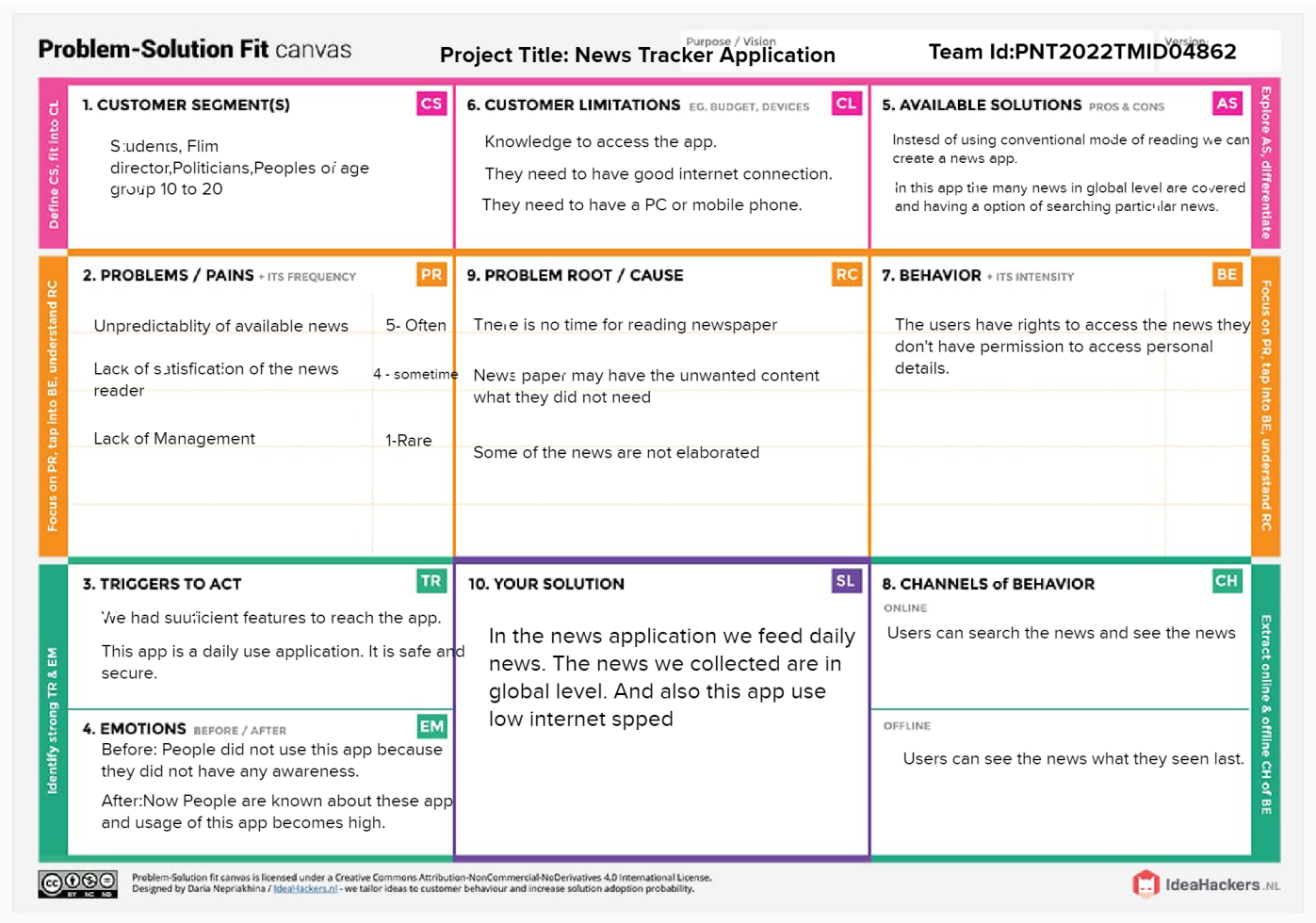
5. Business Model (Revenue Model)

Now this app is advertisement free but for the purpose of revenue we can add advertisement to this app and also we can made a premium plan.

6. Scalability of the Solution

This application is scalable because this app provides all information in any kind of situation and it contains updated information. And this app have less error and low bugs.

3.4 PROBLEM SOLUTION FIT



CHAPTER 4

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

User Registration:

Users can register through google form and also through mail.

User Confirmation:

Users receive their confirmation via mail or via OTP(One Time

Password)

User Login:

Users can login using their email and password or using verification code they can login.

Database Management:

Authenticate of end user login information with the data stored in the database and the application necessary information stored in the database.

Home Page:

Using home page users can view the headline news of that day.

Tracker Page:

This page shows the interested news to the user based on their interest.

Testing:

To find the bugs and error necessary tests are performed.

Deploying:

Deployment of the application in the server with the ability to automatically scale up the resource.

4.2 NON-FUNCTIONAL REQUIREMENTS

Usability:

Usability shows how well the application is built for the end users in order to accept. Our app/website is used to gain knowledge by reading news.

Security:

Security system consists off the user’s login information along with the information provided to the application for the subscription

Reliability:

Reliability shows how far the application can satisfy the user needs with respect to the queries passed by the user.

Performance:

Performance matters when the application loads faster from the server and the request and response passed from user to server and vice-versa.

Availability:

Availability of the application is 24/7, which means the application can be accessed by the user at anywhere and anytime.

Scalability:

Scalability depends on the number of resources used by the application to process the user’s request and to provide the necessary information without crash in the database.

CHAPTER 5

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

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	Registration	USN-1	As a user I can register my account.	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
	Profile updating	USN-3	As a user, I have to enter my Login Credentials To watch daily news	I can update this information on dashboard	High	Sprint-1
	Login	USN-4	As a user, I can log	I can access my	High	Sprint-1

			into the application by entering email & password	account / dashboard		
	Dashboard	USN-5	As a user, I can search news	I can get search results	High	Sprint-2
		USN-6	As a user, I can watch and read news based on category	I can access my account / dashboard	Medium	Sprint-3
Administrator	Maintain the applications	USN-7	Containerizing the application and Maintaining details for users	I can access database	High	Sprint-4

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

TITLE	DESCRIPTION	DATE
Literature Survey and Information Gathering	Literature survey on the selected project & gathering information by referring the technical papers, research publications, etc.	3 September 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user pains & gains, prepare the list of problem statements.	10 September 2022

Customer Journey	Prepare the customer journey maps, understand the user interactions and experiences with the application.	8 October 2022
Solution Requirement	Prepare the functional requirement document.	10 October 2022
Data Flow Diagram	Draw the data flow diagrams and submit for review.	15 October 2022
Technology Architecture	Prepare the technology architecture diagram	20 October 2022
Prepare Milestone & Activity List	Prepare the milestones and activity list of the project.	28 October 2022
Project Development Delivery of Sprint-1,2,3 & 4	Develop and submit the developed code by testing it.	19 October 2022

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Sanchana Shree K I
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Sandhiya S
Sprint-1		USN-3	As a user, I can register for the application through Facebook	3	Low	Sandhiya G
Sprint-1		USN-4	As a user, I can register for the application through Gmail	3	Medium	Samyuktha C
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	3	High	Sanchana Shree K I
Sprint-2	Dashboard	USN-6	Choose the particular area of interest (news domain) of the people.	5	High	Sandhiya S
Sprint-2	Layout	USN-7	As a user I should be able to access the portal with different devices with the same comfort.	3	High	Sandhiya G
Sprint-3	Data Store and retrieval	USN-8	Get Data from API and store as JSON in DB2	3	Medium	Samyuktha C
Sprint-3		USN-9	Get bin data from API and store in DFS	2	High	Sanchana Shree K I
Sprint-4	User Segregation and data access	USN-10	As a CC executive I should be able to uniquely identify the customer and offer help	1	Low	Sandhiya S
Sprint-4	Change code	USN-11	As a administrator I should be able to modify code according to the future requirements.	2	Medium	Sandhiya G
Sprint-4	Monitor the system	USN-12	As a administrator I should be able to monitor the cloud system and fix errors before customer	1	High	Samyuktha C

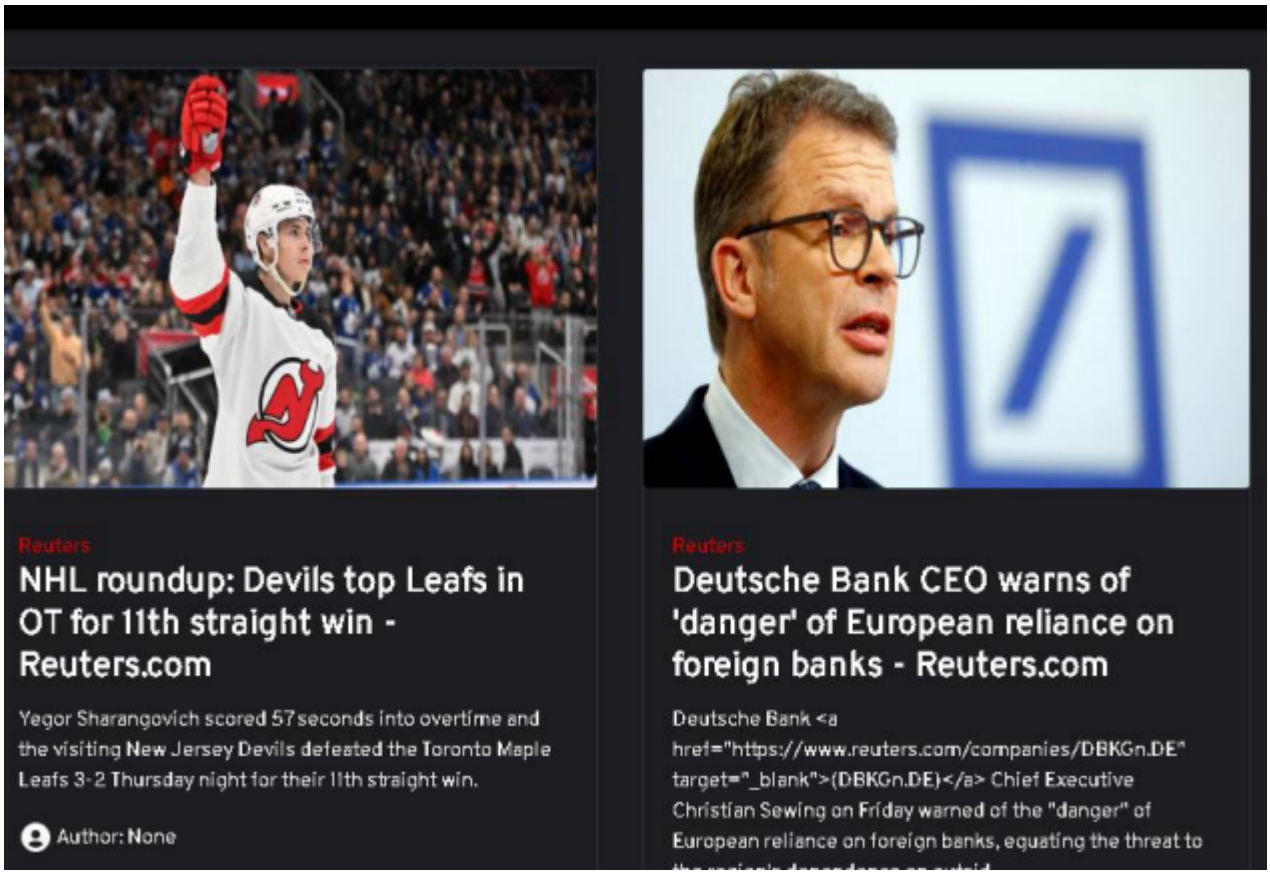
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	29 Oct 2022	02 Nov 2022	8	02 Nov 2022
Sprint-2	20	6 Days	02 Nov 2022	05 Nov 2022	4	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	4	19 Nov 2022

CHAPTER 7

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

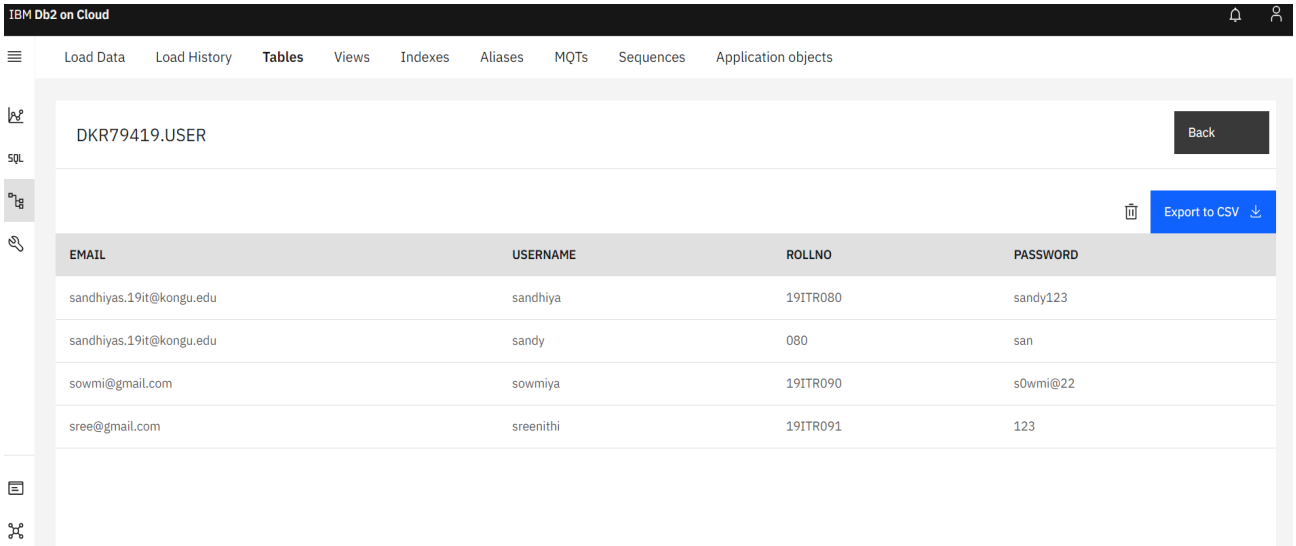
7.1 Feature 1

```
<script>
    window.watsonAssistantChatOptions = {
        integrationID: "fa18fb77-6c75-43f6-b1a1-a4f29fbbbc16",
        region: "eu-gb",
        serviceInstanceID: "87be378b-e186-46e4-
9050a97b5f524d5b",
        onLoad: function(instance) { instance.render(); } };
        setTimeout(function(){ const
t=document.createElement('script');
t.src="https://web[1]chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js"; document.head.appendChild(t); });
```



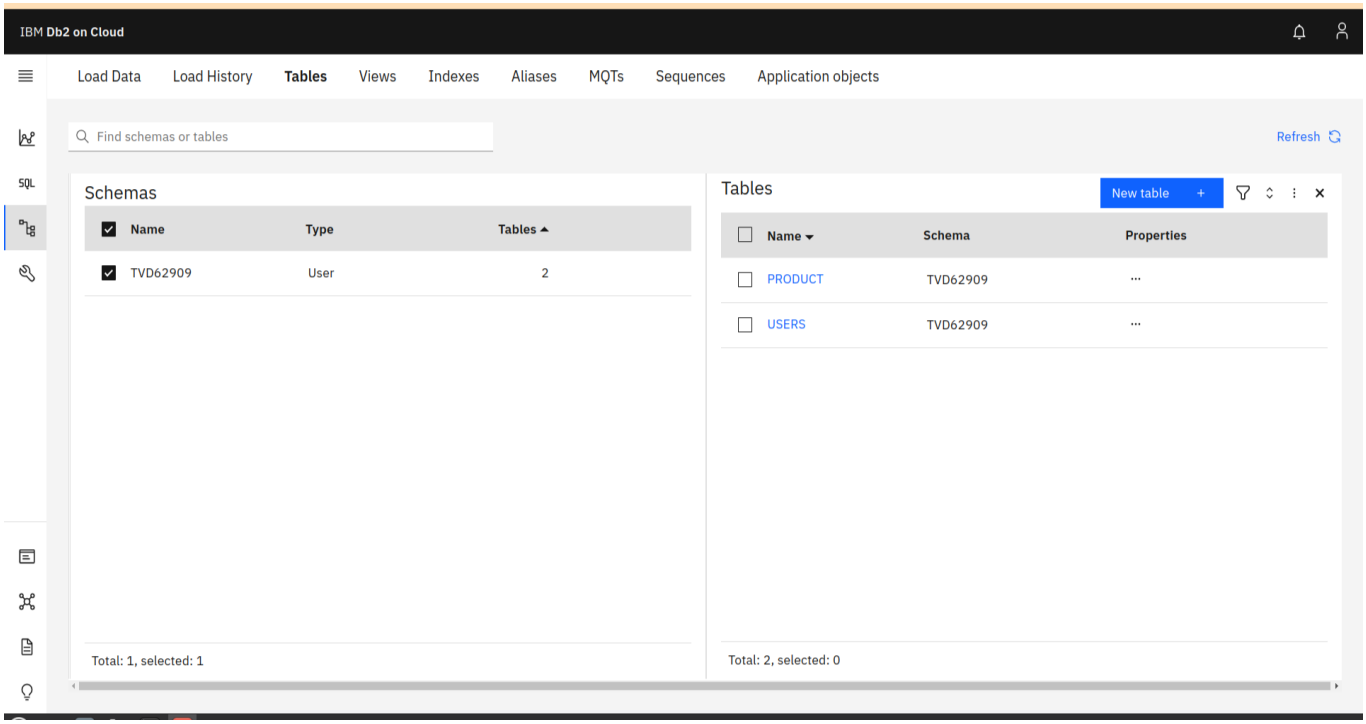
7.2 Feature 2

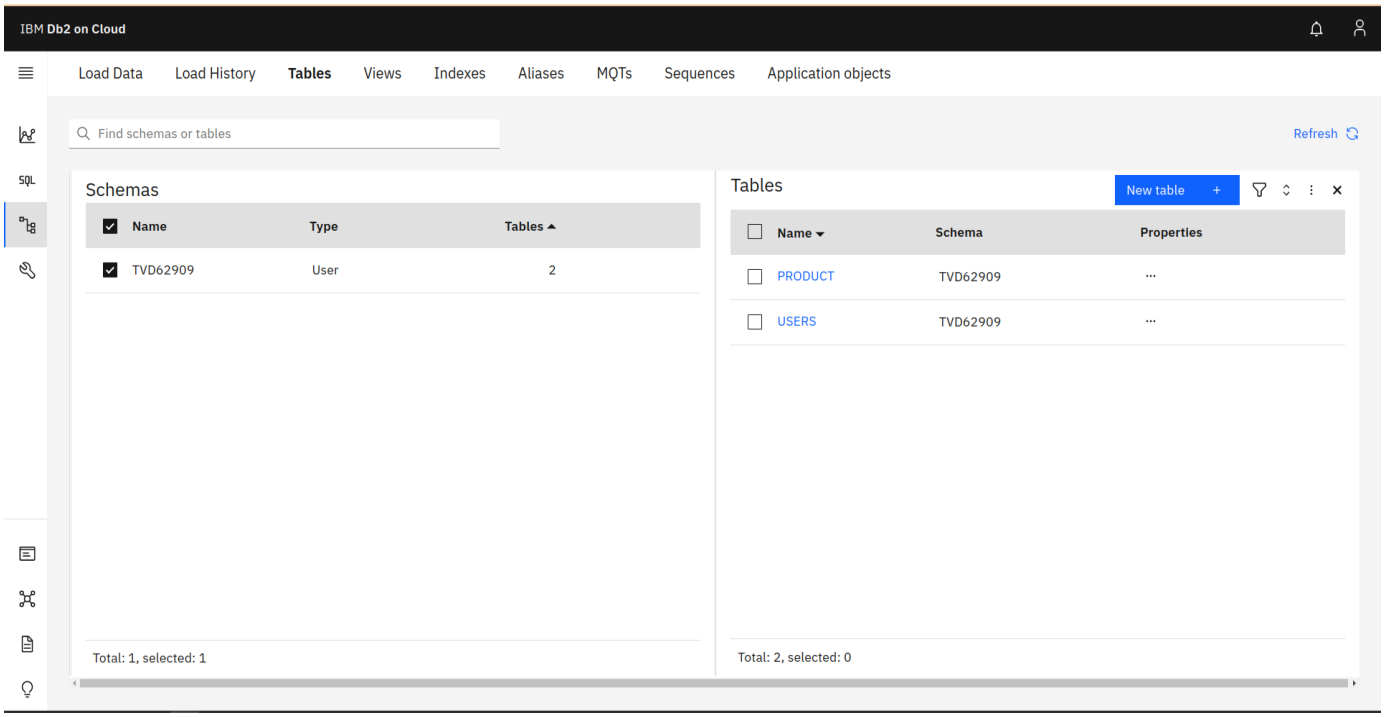
For the user information consist of user name, user id, gmail, password**.** If account exists then it alerts.

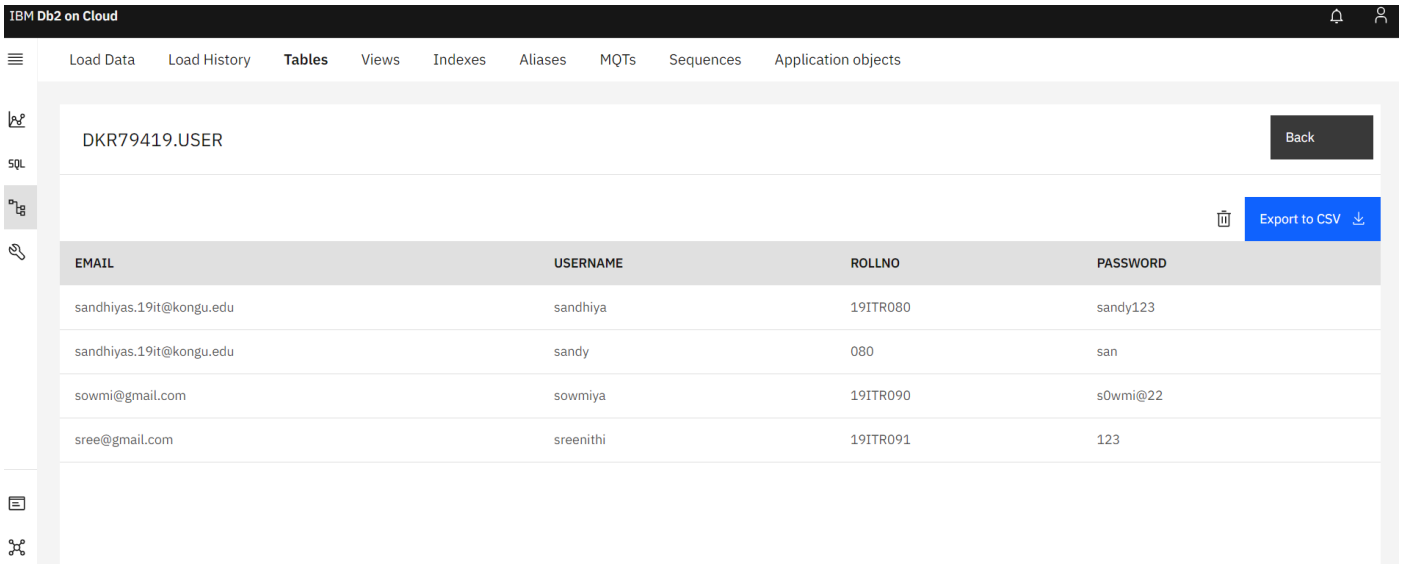


EMAIL	USERNAME	ROLLNO	PASSWORD
sandhiyas.19it@kongu.edu	sandhiya	19ITR080	sandy123
sandhiyas.19it@kongu.edu	sandy	080	san
sowmi@gmail.com	sowmiya	19ITR090	s0wmi@22
sree@gmail.com	sreenithi	19ITR091	123

7.3 Database Schema


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Name
Type
Tables
TVD62909
User
2
Total: 1, selected: 1
Tables
New table
Name
Schema
Properties
PRODUCT
TVD62909
USERS
TVD62909
Total: 2, selected: 0


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Name
Type
Tables
TVD62909
User
2
Total: 1, selected: 1
Tables
New table
Name
Schema
Properties
PRODUCT
TVD62909
USERS
TVD62909
Total: 2, selected: 0



CHAPTER 8

TESTING

8. TESTING

8.1 Test Cases

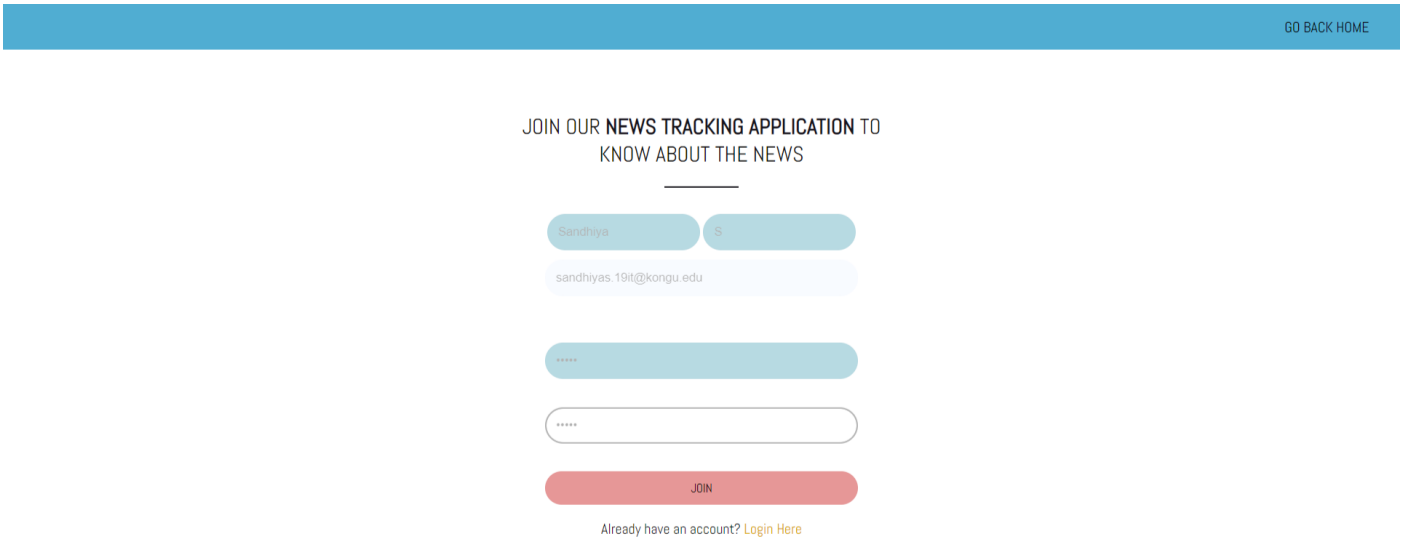
Login

1. Verify user is able to see login page
2. Verify user is able to loginto application or not?
3. Verify login page elements

Register

1. Verify if user is able to enter all the details and register
2. Verify if user is redirected to login page once registered.

8.2 User Acceptance Testing



CHAPTER 9

9. RESULTS

FInally we obtained a web application for news tracker application for people it gives the major outcome of this application . All the requirements for news tracker application is obtained as mucha s possible.

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES:

It provides higher accuracy through cross validation.

High stability compare to Existing system.

It is an user-friendly application.

To explore and discover trending news and topics.

Easily accessible and portable.

Better user experience.

Minute by minute updates of news.

.

10.2 DISADVANTAGES:

Occurrence of Advertisement disturb the user.

Sometimes the news gives brief information.

It works only through internet.

Device fault may affect the application.

Fake news may mislead the readers.

Because the business must install specialised systems and software in order to use them, someNews tracker application can be expensive to implement.

11. CONCLUSION

We explored the feasibility of recognizing patterns of news reading interactions and evaluated three adaptive interface designs for different news reader types. We show that from their interaction log, a specific user can be recognized as one of three kinds. The reader types emerging from the online survey are well defined and distinct. The evaluation of the three variant interfaces suggests that different news reader types need different user interfaces. We have

demonstrated a method for monitoring users’ news reading behavior and inferring news reader type from it. In the future we will further explore the design of adaptive interfaces, in order to be in a position to demonstrate a complete adaptive mobile news framework providing automatic personalization of news apps.

12. FUTURE SCOPE

In the future we will further explore the design of adaptive interfaces, in order to be in a position to demonstrate a complete adaptive mobile news framework providing automatic personalization of news apps.

13. APPENDIX

13.1 SOURCE CODE

Login.html

```
<!DOCTYPE html>a http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="
<html lang="en">
<head>
   <meta charset="UTF-8">
   <met
viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <link rel="Stylesheet" href="Login.css">
</head>
<body>
   <section class="login">
      <div class="signup">
       <form action="">
         <h5>Click here to create a new Account</h5>
```

```
<button formaction="/Signup.html">SignUp</button>
        </form>
       </div>
       <div class="login_box">
          <div class="left">
             <div class="top_link"><a href="/"><img
src="https://drive.google.com/u/0/uc?id=16U__U5dJdaTfNGobB_OpwAJ73vM50rPV&export=downloa
d" alt="">Return home</a></div>
             <div class="contact">
                <form action="">
                   <h3>SIGN IN</h3>
                   <input type="text" placeholder="USERNAME">
                   <input type="password" placeholder="PASSWORD">
                   <button class="submit" formaction="/">LET'S GO</button>
                </form>
             </div>
          </div>
          <div class="right">
             <div class="right-text">
                <h2>News Tracker</h2>
                <h5>Login to know about what is happening??</h5>
             </div>
       </div>
    </section>
</body>
</html>
```

Register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SignUp</title>
    <link rel="Stylesheet" href="Signup.css">
</head>
<body>
    <nav>
        <ul class="navigation">
            <!-- Making menu icon clickable to display the navigation menu on smaller screens -->
                <i onclick="navToggle()" id="nav-icon" class="fa fa-navicon" style="font-size:24px"></i>
            </div>
   <div id="toggle" class="nav-container">
                <a class="left" href="/"><li>GO BACK HOME</li></a>
            </div>
        </ul>
    </nav>
<!-- NAVIGATION END HERE -->
<section class="form">
<div class="center">
    <h1>JOIN OUR <b style="color: #17141c ;">NEWS TRACKING APPLICATION</b> TO KNOW ABOUT THE NEWS</h1>
    <hr width="20%" style="border: 1px solid #1c1a20;">
    <br>
<form action="">
        <input class="name-surname" type="text" name="name" placeholder="firstname">
        <input class="name-surname" type="text" name="surname" placeholder="lastname"><br>
        <input type="text" name="email" placeholder="emailid"><br>
         <br> <input type="password" name="password" placeholder="password"><br>
        <input type="password" name="conf_password" placeholder="confirmpassword"><br>
    <button formaction="/">JOIN</button>
        <p>Already have an account? <a href="/Login">Login Here</a></p>
```

```
</form>
</div>
</section>
</body>
</html>
```

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <meta name="Description" content="Enter your description here"/>
   <!-- Favicons -->
   <link href="{{ url_for('static', filename='/favicon.ico') }}" rel="icon">
   <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Overpass">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.6.0/css/bootstrap.min.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
   <link rel="stylesheet" href="Style.css">
   <title>One Tap</title>
</head>
<body>
  {% block navbar %}
 <div class="container-fluid" style="background-color: #000;">
       <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top shadow-sm">
          <!--  Show this only on mobile to medium screens  -->
            <a class="navbar-brand d-lg-none" href="home.html"><b class="lg"><span
class="logo">One</span> Tap</b></a>
```

```
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle"
aria-controls="navbarToggle" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
         <!--  Use flexbox utility classes to change how the child elements are justified  -->
          <div class="collapse navbar-collapse justify-content-between" id="navbarToggle">
            <ul class="navbar-nav">
              <li class="nav-item">
                <a class="nav-link" href="headlines.html">Headlines</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="Articles.html">Articles</a>
              </li>
              <li class="nav-item">
                <a class="nav-link"href="sources.html">Sources</a>
              </li>
              <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-
toggle="dropdown" aria-expanded="false">
                  Categories
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                  <a class="dropdown-item" href="{{ url_for('business') }}">Business</a>
                  <a class="dropdown-item" href="{{ url_for('tech') }}">Technology</a>
                  <a class="dropdown-item" href="{{ url_for('entertainment') }}">Entertainment</a>
                  <a class="dropdown-item" href="{{ url_for('science') }}">Science</a>
                  <a class="dropdown-item" href="{{ url_for('sports') }}">Sport</a>
                  <a class="dropdown-item" href="{{ url_for('health') }}">Health</a>
                </div>
              </li>
            </ul>
```

```
<!--   Show this only lg screens and up   -->
            <a class="navbar-brand d-none d-lg-block" href="home.html"><b class="lg"><span class="logo">One</span> Tap</b></a>
          <ul class="navbar-nav">
                <li class="nav-item">
                  <a class="nav-link" href="/headlines">
                    <svg class="svg" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M16.5,8c0,1.5-0.5,3.5-2.9,4.3c0.7-1.7,0.8-3.4,0.3-5c-0.7-2.1-3-3.7-4.6-4.6C8.9,2.4,8.2,2.8,8.3,3.4c0,1.1-0.3,2.7-2,4.4C4.1,10,3,12.3,3,14.5C3,17.4,5,21,9,21c-4-4-1-7.5-1-7.5c0.8,5.9,5,7.5,7,7.5c1.7,0,5-1.2,5-6.4c0-3.1-1.3-5.5-2.4-6.9  C17.3,7.2,16.6,7.5,16.5,8"></path></svg>
                  </a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="#">
                    <svg class="svg" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M19 13.586V10c0-3.217-2.185-5.927-5.145-6.742C13.562 2.52 12.846 2 12 2s-1.562.52-1.855 1.258C7.185 4.074 5 6.783 5 10v3.586l-1.707 1.707A.996.996 0 0 0 3 16v2a1 1 0 0 0 1 1h16a1 1 0 0 0 1-1v-2a.996.996 0 0 0-.293-.707L19 13.586zM19 17H5v-.586l1.707-1.707A.996.996 0 0 0 7 14v-4c0-2.757 2.243-5 5-5s5 2.243 5 5v4c0 .266.105.52.293.707L19 16.414V17zm-7 5a2.98 2.98 0 0 0 2.818-2H9.182A2.98 2.98 0 0 0 12 22z"></path></svg>
                  </a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="/Login.html">
                    <svg class="svg" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M12 2A10.13 10.13 0 0 0 2 12a10 10 0 0 0 4 7.92V20h.1a9.7 9.7 0 0 0 11.8 0h.1v-.08A10 10 0 0 0 22 12 10.13 10.13 0 0 0 12 2zM8.07 18.93A3 3 0 0 1 11 16.57h2a3 3 0 0 1 2.93 2.36 7.75 7.75 0 0 1-7.86 0zm9.54-1.29A5 5 0 0 0 13 14.57h-2a5 5 0 0 0-4.61 3.07A8 8 0 0 1 4 12a8.1 8.1 0 0 1 8-8 8.1 8.1 0 0 1 8 8 8 8 0 0 1-2.39 5.64z"></path><path d="M12 6a3.91 3.91 0 0 0-4 4 3.91 3.91 0 0 0 4 4 3.91 3.91 0 0 0 4-4 3.91 3.91 0 0 0-4-4zm0 6a1.91 1.91 0 0 1-2-2 1.91 1.91 0 0 1 2-2 1.91 1.91 0 0 1 2 2 1.91 1.91 0 0 1-2 2z"></path></svg> Log In / Register
                  </a>
                </li>
            </ul>
```

```
 </div>
        </nav>
    </div>
 {% endblock %}
 {% block content %}
   {% endblock %}
    {% block footer %}
    <div class="container-fluid footer">
        <p class="text-center footer-text">
            © <span class="logo">News</span> NEWS TRACKER APPLICATION
        </p>
    </div>
    {% endblock %}
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.1/umd/popper.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.6.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Base.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="Description" content="Enter your description here"/>
    <!-- Favicons -->
    <link href="{{ url_for('static', filename='/favicon.ico') }}" rel="icon">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Overpass">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
```

```
bootstrap/4.6.0/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="Style.css">
    <title>One Tap</title>
</head>
<body>
 {% block navbar %}
    <div class="container-fluid" style="background-color: #000;">
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top shadow-sm">
            <!--  Show this only on mobile to medium screens  -->
             <a class="navbar-brand d-lg-none" href="home.html"><b class="lg"><span
class="logo">One</span> Tap</b></a>
             <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarToggle" aria-controls="navbarToggle" aria-expanded="false" aria-label="Toggle
navigation">
              <span class="navbar-toggler-icon"></span>
             </button>
             <!--  Use flexbox utility classes to change how the child elements are justified  -->
             <div class="collapse navbar-collapse justify-content-between" id="navbarToggle">
              <ul class="navbar-nav">
                 <li class="nav-item">
                     <a class="nav-link" href="headlines.html">Headlines</a>
                 </li>
                 <li class="nav-item">
                     <a class="nav-link" href="Articles.html">Articles</a>
                 </li>
                 <li class="nav-item">
                     <a class="nav-link"href="sources.html">Sources</a>
                 </li>
                 <li class="nav-item dropdown">
                     <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-
toggle="dropdown" aria-expanded="false">
```

```
                Categories
              </a>
              <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                <a class="dropdown-item" href="{{ url_for('business') }}">Business</a>
                <a class="dropdown-item" href="{{ url_for('tech') }}">Technology</a>
                <a class="dropdown-item" href="{{ url_for('entertainment') }}">Entertainment</a>
                <a class="dropdown-item" href="{{ url_for('science') }}">Science</a>
                <a class="dropdown-item" href="{{ url_for('sports') }}">Sport</a>
                <a class="dropdown-item" href="{{ url_for('health') }}">Health</a>
              </div>
            </li>
          </ul>
        <!--   Show this only lg screens and up   -->
          <a class="navbar-brand d-none d-lg-block" href="home.html"><b class="lg"><span
class="logo">One</span> Tap</b></a>
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link" href="/headlines">
                <svg class="svg" xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M16.5,8c0,1.5-
0.5,3.5-2.9,4.3c0.7-1.7,0.8-3.4,0.3-5c-0.7-2.1-3-3.7-4.6-4.6C8.9,2.4,8.2,2.8,8.3,3.4c0,1.1-0.3,2.7-2,4.4
C4.1,10,3,12.3,3,14.5C3,17.4,5,21,9,21c-4-4-1-7.5-1-7.5c0.8,5.9,5,7.5,7,7.5c1.7,0,5-1.2,5-6.4c0-3.1-1.3-
5.5-2.4-6.9  C17.3,7.2,16.6,7.5,16.5,8"></path></svg>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">
                <svg class="svg" xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M19
13.586V10c0-3.217-2.185-5.927-5.145-6.742C13.562 2.52 12.846 2 12 2s-1.562.52-1.855 1.258C7.185
4.074 5 6.783 5 10v3.586l-1.707 1.707A.996.996 0 0 0 3 16v2a1 1 0 0 0 1 1h16a1 1 0 0 0 1-1v-
2a.996.996 0 0 0-.293-.707L19 13.586zM19 17H5v-.586l1.707-1.707A.996.996 0 0 0 7 14v-4c0-2.757
2.243-5 5-5s5 2.243 5 5v4c0 .266.105.52.293.707L19 16.414V17zm-7 5a2.98 2.98 0 0 0 2.818-
2H9.182A2.98 2.98 0 0 0 12 22z"></path></svg>
              </a>
```

```
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/Login.html">
                        <svg class="svg" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24"
height="24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M12 2A10.13 10.13 0 0 0
2 12a10 10 0 0 0 4 7.92V20h.1a9.7 9.7 0 0 0 11.8 0h.1v-.08A10 10 0 0 0 22 12 10.13 10.13 0 0 0 12
2zM8.07 18.93A3 3 0 0 1 11 16.57h2a3 3 0 0 1 2.93 2.36 7.75 7.75 0 0 1-7.86 0zm9.54-1.29A5 5 0 0 0 13
14.57h-2a5 5 0 0 0-4.61 3.07A8 8 0 0 1 4 12a8.1 8.1 0 0 1 8-8 8.1 8.1 0 0 1 8 8 8 8 0 0 1-2.39
5.64z"></path><path d="M12 6a3.91 3.91 0 0 0-4 4 3.91 3.91 0 0 0 4 4 3.91 3.91 0 0 0 4-4 3.91 3.91 0 0
0-4-4zm0 6a1.91 1.91 0 0 1-2-2 1.91 1.91 0 0 1 2-2 1.91 1.91 0 0 1 2 2 1.91 1.91 0 0 1-2
2z"></path></svg> Log In / Register
                    </a>
                </li>
            </ul>
        </div>
    </nav>
</div>
{% endblock %}
{% block content %}
{% endblock %}
{% block footer %}
<div class="container-fluid footer">
    <p class="text-center footer-text">
        © <span class="logo">News</span> NEWS TRACKER APPLICATION
    </p>
</div>
{% endblock %}
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.1/umd/popper.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.6.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Source.html

```
{% extends 'base.html' %}

{% block content %}

<div class="container-fluid landing">

    <div class="row">

        {% for source in newsSource %}

            <div class="col-md-3 d-flex justify-content-center source-body">

                <div class="card bg-dark">

                    <div class="card-body">

                        <a href="{{ source.url }}" target="_blank">

                            <h5 class="card-title"><svg xmlns="http://www.w3.org/2000/svg" width="36"
height="36" viewBox="0 0 24 24" style="fill: rgba(255, 255, 255, 1);transform: msFilter;"><path d="M12
2C6.486 2 2 6.486 2 12s4.486 10 10 10 10-4.486 10-10S17.514 2 12 2zM4 12c0-.899.156-1.762.431-
2.569L6 11l2 2v2l2 2 1 1v1.931C7.061 19.436 4 16.072 4 12zm14.33 4.873C17.677 16.347 16.687 16 16
16v-1a2 2 0 0 0-2-2h-4v-3a2 2 0 0 0 2-2V7h1a2 2 0 0 0 2-2v-.411C17.928 5.778 20 8.65 20 12a7.947
7.947 0 0 1-1.67 4.873z"></path></svg> {{ source.name }}</h5>

                            <p class="card-text">{{ source.description }}</p>

                        </a>

                    </div>

                </div>

            </div>

        {% endfor %}

    </div>

</div>


{% endblock %}
```

init.py

```
from flask import Flask


app = Flask(__name__)


from app1 import app
```

app.py

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

import ibm_db

import re


app = Flask(__name__)

app.secret_key = 'fasdgfdgdfg'


conn = ibm_db.connect(

"DATABASE=bludb;HOSTNAME=ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;USERNAME=mxDzFYL9acWKSyQR;PASSWORD=dkr79419;SECURITY=SSL;SSLSERVERCERTIFICATE=DigiCertGlobalRootCA(2).crt;", '', '')

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

def register():

    msg = ''

    if request.method == 'POST':

        username = request.form['name']

        email = request.form['email']

        password = request.form['password']

        sql = "SELECT * FROM users WHERE username =?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, username)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

        if account:

            msg = 'Account already exists !'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

            msg = 'Invalid email address !'
```

```
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            insert_sql = "INSERT INTO users VALUES (?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, password)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully registered !'
    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg=msg)

@app.route('/login', methods=['GET', 'POST'])
def login():
    global userid
    msg = ''

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM users WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if account:
```

```
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid = account['USERNAME']
            session['username'] = account['USERNAME']

            msg = 'Logged in successfully !'
            return render_template('Base.html', msg=msg)
        else:
            msg = 'Incorrect username / password !'
            return render_template('Login.html', msg=msg)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

config.py

```
class Config:
    NEWS_BASE_URL_SOURCES = 'https://newsapi.org/v2/top-headlines/sources?apiKey={}'
    NEWS_BASE_EVERYTHING_URL = 'https://newsapi.org/v2/everything?domains={}&apiKey={}'
    NEWS_BASE_HEADLINES_URL = 'https://newsapi.org/v2/top-headlines?country=us&apiKey={}'
    NEWS_BASE_SOURCE = 'https://newsapi.org/v2/top-headlines?sources={}&apiKey={}'
    API_KEY = "367756a948444eb3becb814af0e4bcb8"
class ProdConfig(Config):
    pass
class DevConfig(Config):
    DEBUG = True
config_options= {
    'development': DevConfig,
    'production': ProdConfig
}
```

main.py

```
from .app1 import app


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

model.py

```
class Sources:
    def __init__(self, name, description, url):
        self.name=name
        self.description=description
        self.url=url


class Articles:
    '''Define article model'''
    def __init__(self, source, author, title, description, url, urlToImage, publishedAt):
        self.source = source
        self.author = author
        self.title = title
        self.description = description
        self.url = url
        self.urlToImage = urlToImage
        self.publishedAt = publishedAt
```

request.py

```
from .models import Articles
from .models import Sources
from newsapi.newsapi_client import NewsApiClient
from .config import Config
import urllib.request,json
```

```
api_key=None
base_url=None
base_url_for_everything=None
base_url_top_headlines=None
base_source_list=None

def publishedArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    get_articles = newsapi.get_everything(sources= 'cnn, reuters, cnbc, the-verge, gizmodo, the-next-web, techradar, recode, ars-technica')

    all_articles = get_articles['articles']

    articles_results = []

    source = []
    title = []
    desc = []
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
        author.append(article['author'])
```

```
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def topHeadlines():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    top_headlines = newsapi.get_top_headlines(sources= 'cnn, reuters, cnbc, techcrunch, the-verge,
gizmodo, the-next-web, techradar, recode, ars-technica')

    all_headlines = top_headlines['articles']

    articles_results = []

    source = []
    title = []
    desc = []
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_headlines)):
```

```
        headline = all_headlines[i]

        source.append(headline['source'])
        title.append(headline['title'])
        desc.append(headline['description'])
        author.append(headline['author'])
        img.append(headline['urlToImage'])
        p_date.append(headline['publishedAt'])
        url.append(headline['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def randomArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    random_articles = newsapi.get_everything(sources= 'the-verge, gizmodo, the-next-web, recode, ars-technica')

    all_articles = random_articles['articles']

    articles_results = []

    source = []
    title = []
    desc = []
```

```
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
        author.append(article['author'])
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def businessArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    business_articles = newsapi.get_top_headlines(category='business')

    all_articles = business_articles['articles']
```

```
    business_articles_results = []

    source = []
    title = []
    desc = []
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
        author.append(article['author'])
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        business_articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def techArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)
```

```
tech_articles = newsapi.get_top_headlines(category='technology')

all_articles = tech_articles['articles']

tech_articles_results = []

source = []
title = []
desc = []
author = []
img = []
p_date = []
url = []

for i in range(len(all_articles)):
    article = all_articles[i]

    source.append(article['source'])
    title.append(article['title'])
    desc.append(article['description'])
    author.append(article['author'])
    img.append(article['urlToImage'])
    p_date.append(article['publishedAt'])
    url.append(article['url'])

    article_object = Articles(source, title, desc, author, img, p_date, url)

    tech_articles_results.append(article_object)

    contents = zip(source, title, desc, author, img, p_date, url)
```

```
    return  contents

def entArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    ent_articles = newsapi.get_top_headlines(category='entertainment')

    all_articles = ent_articles['articles']

    ent_articles_results = []

    source = []
    title = []
    desc = []
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
        author.append(article['author'])
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])
```

```
        article_object = Articles(source, title, desc, author, img, p_date, url)

        ent_articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def scienceArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    science_articles = newsapi.get_top_headlines(category='science')

    all_articles = science_articles['articles']

    science_articles_results = []

    source = []
    title = []
    desc = []
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
```

```
        author.append(article['author'])
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        science_articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def sportArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    sport_articles = newsapi.get_top_headlines(category='sports')

    all_articles = sport_articles['articles']

    sport_articles_results = []

    source = []
    title = []
    desc = []
    author = []
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
```

```
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
        author.append(article['author'])
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        sport_articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def healthArticles():
    newsapi = NewsApiClient(api_key= Config.API_KEY)

    health_articles = newsapi.get_top_headlines(category='health')

    all_articles = health_articles['articles']

    health_articles_results = []

    source = []
    title = []
    desc = []
    author = []
```

```
    img = []
    p_date = []
    url = []

    for i in range(len(all_articles)):
        article = all_articles[i]

        source.append(article['source'])
        title.append(article['title'])
        desc.append(article['description'])
        author.append(article['author'])
        img.append(article['urlToImage'])
        p_date.append(article['publishedAt'])
        url.append(article['url'])

        article_object = Articles(source, title, desc, author, img, p_date, url)

        health_articles_results.append(article_object)

        contents = zip(source, title, desc, author, img, p_date, url)

    return  contents

def get_news_source():
  '''
  Function that gets the json response to our url request
  '''
  get_news_source_url = 'https://newsapi.org/v2/sources?apiKey=' + Config.API_KEY
  with urllib.request.urlopen(get_news_source_url) as url:
    get_news_source_data = url.read()
    get_news_source_response = json.loads(get_news_source_data)
```

```
    news_source_results = None

    if get_news_source_response['sources']:
      news_source_results_list = get_news_source_response['sources']
      news_source_results = process_sources(news_source_results_list)

  return news_source_results

def process_sources(source_list):
  '''
  function that process the news articles and transform them to a list of objects
  '''
  news_source_result = []
  for news_source_item in source_list:
    name = news_source_item.get('name')
    description = news_source_item.get('description')
    url = news_source_item.get('url')

    if name:
      news_source_object = Sources(name, description,url)
      news_source_result.append(news_source_object)
  return news_source_result
```

GitHub & Project Demo Link

Github Link:

https://github.com/IBM-EPBL/IBM-Project-23351-1659879719

Project demo link

https://drive.google.com/file/d/1nRSRj1kSzWQgHSXg7l5NUwSzNvZ-1lsf/view?usp=sharing