News Tracker Application

Introduction

Project Overview

We frequently feel that we need more than 24 hours a day to do everything on our calendar because our lives are so hectic these days. We therefore require a tool that can provide us with all the pertinent news, including the most recent and important ones, in a structured and curated way that enables users to efficiently explore and absorb the content. With the aid of this software, you may search for all available data on indices, commodities, currencies, future rates, bonds, etc., just as on reputable websites.

Our product is ideated in such a way that the users will be able to get only the relevant information and news of their chosen topic thus making sure that the users stay with the app for a longer period of time. The features of our product are as follows:

1. Each user has their own profile which will contain information about their chosen or interested topics, and news will be fetched and displayed from only those topics. Furthermore the topics can later be changed as per the users wish.

2. Displaying various stats regarding the user's analytics based on the viewing of various new articles, which allows the user to know about his own preferences and interests in a deeper manner.

3. Option to save stories which can be seen in the users profile, this will allow the user to keep some news stories for quick reference in the future. This is a very useful feature which saves a lot of time for the user, in case they need to revisit something they read or saw before.

4. Allows the users to follow a particular news publication, so that the user can get to know about all the latest news updates provided by the selected news publication.

5. This application not only focuses on the new readers, it can also act as a portal for the news publication houses to understand their viewership and hence we can monetize this to make the app sustainable for the developers.

Our product will be highly scalable as we have proposed to use IBM's Cloud services. Because of that, our product can scale for millions of users and work flawlessly. This could help generate more revenue for the product and move the product towards profitability.

Purpose

We have gathered all the pertinent drawbacks of the absence of news monitoring statistics, as well as the findings of studies and reports that have examined the problem.

This guide will provide you with an objective analysis of why the media reports bad news. We'll provide you a thorough and knowledgeable review of the topic as a whole.

- Sensationalist stories form 95% of media headlines nowadays.
- Media reports with negative news or statistics catch 30% more attention.
- 26.7% of people exposed to negative news go on to develop anxiety issues.
- 63% of kids aged 12–18 say that watching the news makes them feel bad.
- Moreover, it makes you frustrated when you wake up in the morning and want to know the news in time before getting out to work or college but you are not able to get the required news on time. This may lead to frustration and many people break televisions.
- Nearly 67.92% of the people are not able to track the news they require and a lot of their precious time is wasted which they can use to productively do something.
- Nowadays most of the news are miserable and increases suicidal thoughts and depression seeing them.
- Improper covid 19 figures made several people panic and death due to panic was more than covid deaths.
- So we have built this app where the individual views the news that he likes and wants to know.
- This increases his knowledge and research content in the topic he is willing to invest his time and willingness to know more.
- Lightning News solves this problem and just helps you to focus on the news that the individual required

Literature Survey

Existing problem

We've collected all the relevant negative factors of lack of news tracking statistics, along with results from studies and reports that have analyzed the issue.

This guide will give you an unbiased look at why the media reports negative news. We'll provide you with an informed and educated overview of the subject in general.

Top Negative News Facts:

- Moreover, it makes you frustrated when you wake up in the morning and want to know the news in time before getting out to work or college but you are not able to get the required news on time. This may lead to frustration and many people break televisions.
- Sensationalist stories form 95% of media headlines nowadays.
- Media reports with negative news or statistics catch 30% more attention.
- 26.7% of people exposed to negative news go on to develop anxiety issues.
- 63% of kids aged 12–18 say that watching the news makes them feel bad
- Nearly 67.92% of the people are not able to track the news they require and a lot of their precious time is wasted which they can use to productively do something.
- Nowadays most of the news are miserable and increases suicidal thoughts and depression seeing them.
- Improper covid 19 figures made several people panic and death due to panic was more than covid deaths.
- So we have built this app where the individual views the news that he likes and wants to know.
- This increases his knowledge and research content in the topic he is willing to invest his time and willingness to know more.
- Lightning News solves this problem and just helps you to focus on the news that the individual required

Social Impact for this problem

1. News is important for a number of reasons within a society. Mainly to inform the public about events that are around them and may affect them.
2. Often news is for entertainment purposes too; to provide a distraction of information about other places people are unable to get to or have little influence over. News can make people feel connected too.
3. News is important as a social gathering space too, hence newspapers either online or physical place an emphasis on news. Where there are a lot of people gathered there is opportunity to advertise. This advertising sometimes can cause a conflict of interest in the way news is reported.

References

Flipboard App ->The app's interface shows various types of content, including articles, slideshows, and videos, and it connects with social media sites like Twitter and Instagram.

Google News -> Google News uses its big data muscles by personalizing each user's feed. Under each headline, the "Full Coverage" button lets users round up coverage from the best news sources on a given topic—something discerning readers value in an era of intense partisanship and concern about biased reporting.

DailyHunt -> Dailyhunt covers news from diverse genres - Politics, Entertainment, Sports, Business, Weather, Astrology, Finance, Technology, Fashion, Beauty, Health & Fitness & more.

Inshorts -> Inshorts is a news app that selects latest and best news from multiple national and international sources and summarises them to present in a short and crisp 60 words or less format, personalized for you, in both, English or Hindi.

Problem Statement Definition

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. Hence we need an application which can get us all the relevant news, including all the latest and critical ones in a structured and curated manner allowing the users to navigate through and consume the content in an efficient manner. This app helps you to query for all information about Indices, Commodities, Currencies, Future Rates, Bonds, etc as on official websites.

Edit this template
Right-click to unlock

Empathy Map

Dive into the mind of the user for focused product development

- Build empathy and keep your focus on the user by putting yourself in their shoes.

Why should i

Can I add various types of news

What does this app give

What is the difference between being an individual and being part of a group?

Is this a

Is the news on the app upto date?

Can I Trust

Is my data safe with this app ?

Sets this news category preference

Compares news from this app to other sources

Satisfied about the content

Finds the platform trustabl e

Reads the news/ Consume content

Rate a new article based on this experience

Well Informed

Excited

Share your feedback

Brainstorm & idea prioritization

Use this template in your own

brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How to track news? What are the ways to get notified when a news is spread? Are the sources of the news trustable?

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, bevisual.

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Mo

Karun A

Jayasooryan S

Lokesh N N

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Basic Level

Advanced Level

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance

If each of these

tasks could get done without any difficulty or cost, which would have the most positive impact?

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template

Share template feedback

Share template feedback

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

Proposed Solution

Novelty:

The news tracker application is ideated in such a way that the users will be able to get only the relevant information and news of their chosen topic. The features are as follows:

1. Each user has their own profile which will contain information about their chosen or interested topics, and news will be fetched and displayed from only those topics. Furthermore the topics can later be changed as per the users wish.

2. Displaying various stats regarding the user's analytics based on the viewing of various new articles, which allows the user to know about his own preferences and interests in a deeper manner.

3. Option to save stories which can be seen in the users profile, this will allow the user to keep some news stories for quick reference in the future. This is a very useful feature which saves a lot of time for the user, in case they need to revisit something they read or saw before.

4. Allows the users to follow a particular news publication, so that the user can get to know about all the latest news updates provided by the selected news publication.

5. This application not only focuses on the new readers, it can also act as a portal for the news publication houses to understand their viewership and hence we can monetize this to make the app sustainable for the developers.

Feasibility:

1. The app uses a very simple, lightweight flask backend, optimized for production environments. This makes it easy to engineer, develop and deploy.

2. IBM Cloud provides the necessary infrastructure such as K8s clusters, databases etc. required for efficient deployment.

3. There are many open source news APIs with verified news articles, which may be used as preliminary data sources, for the serving of curated news articles to the users

Since the proposed architecture is simple and lightweight, and the necessary tools and infrastructure are readily available, it can be concluded that this solution is feasible.

Business Model:

We plan to propose the following business model to maximize the revenue:

1. Provide customized advertisements to the user through various ad providers: Google Ads, Amazon Ads, Facebook Ads, etc.

2. As the product is more customer focussed, we could generate a humongous amount of data which is related to user activity, screen time, favorite type of news, etc. This could help sell the data to many other companies with prior consent from the user.

3. Along with a B2C mindset, we also propose a B2B model, where we bring in news publishers which are ad - dependent. This could help get businesses and normal users as customers which could help us generate revenue from either side.

4. We also propose to sell branded content from the news publishers where the normal users could buy something like a subscription and follow their favorite news publishers.

5. A premium subscription model for the normal news consumers is proposed to be introduced so that the users can get rid of ads along with helping us generating more revenue through subscriptions.

Social Impact:

1. News is important for a number of reasons within a society. Mainly to inform the public about events that are around them and may affect them.

2. Often news is for entertainment purposes too; to provide a distraction of information about other places people are unable to get to or have little influence over. News can make people feel connected too.

3. News is important as a social gathering space too, hence newspapers either online or physical place an emphasis on news. Where there are a lot of people gathered there is opportunity to advertise. This advertising sometimes can cause a conflict of interest in the way news is reported.

Scalability:

1. A big part of developing a web app is its capacity to scale. We are aiming to launch a product that has to be ready for the influx of users and expect the system to handle it. Be extra vigilant because there might be a time when our system is not flexible enough and cannot support a heavy load. To prevent this, it is critical to get started on application scalability before the development step comes in.
2. While developing the application, as developers we kept in mind the scalability factor and made sure that our application would have large scalability
3. We have used Python Flask server as our backend which is one of the best when it comes to scalability.
4. Flask by itself is only limited in terms of scaling by your application code, the data store you want to use and the Python implementation and web server you are running on.
5. Moreover we have used IBM DB2 as our database. So now the question arises how does Flask server plus IBM DB2 increase the scalability of the application. The answer to this question is addressed below

Incremental growth

The Parallel Sysplex cluster can grow incrementally. You can add a new Db2

subsystem onto another central processor complex and access the same data through the new Db2 subsystem. You no longer need to manage copies or distribute data. All Db2 subsystems in the data sharing group have concurrent read-write access, and all Db2 subsystems use a single Db2 catalog.

Workload balancing

Db2 data sharing provides flexibility for growth and workload balancing. With the partitioned data approach to parallelism (sometimes called the shared-nothing architecture), a one-to-one relationship exists between a particular DBMS and a segment of data. By contrast, data in a Db2 data sharing environment does not need to be redistributed when you add a new subsystem or when the workload becomes unbalanced. The new Db2 member has the same direct access to the data as all other existing members of the data sharing group.

Db2 works closely with the z/OS Workload Manager (WLM) to ensure that incoming work is optimally balanced across the systems in the cluster. WLM manages workloads that share system resources and have different priorities and resource-use characteristics.

For example, assume that large queries with a low priority are running on the same system as online transactions with a higher priority. WLM can ensure that the queries do not monopolize resources and do not prevent the online transactions from achieving acceptable response times. WLM works in both a single-system and a multisystem (data sharing) environment.

Capacity when you need it

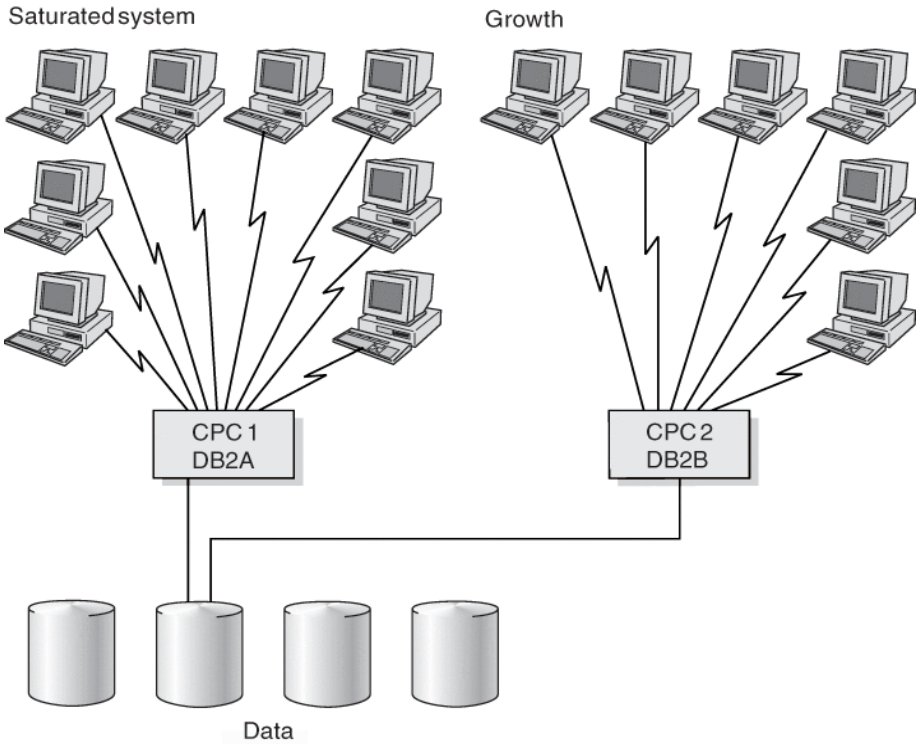
A data sharing configuration can handle your peak loads. You can start data sharing members to handle peak loads, such as end-of-quarter processing, and then stop them when the peak passes.

You can take advantage of all these benefits, whether your workloads are for online transaction processing (OLTP), or a mixture of OLTP, batch, and queries.

Higher transaction rates

Data sharing gives you opportunities to put more work through the system. As the following figure illustrates, you can run the same application on more than one Db2 subsystem to achieve transaction rates that are higher than are possible on a single subsystem.

Figure 1. How data sharing enables growth. You can move some of your existing Db2 workload onto another central processor complex (CPC).



We have deployed our server and the application in Kubernetes

Considerations for large clusters

A cluster is a set of nodes (physical or virtual machines) running Kubernetes agents, managed by the control plane. Kubernetes v1.25 supports clusters with up to 5000 nodes. More specifically, Kubernetes is designed to accommodate configurations that meet all of the following criteria:

- No more than 110 pods per node
- No more than 5000 nodes
- No more than 150000 total pods
- No more than 300000 total containers

You can scale your cluster by adding or removing nodes. The way you do this depends on how your cluster is deployed.

Cloud provider resource quotas

To avoid running into cloud provider quota issues, when creating a cluster with many nodes, consider:

- Requesting a quota increase for cloud resources such as:
    - Computer instances
    - CPUs
    - Storage volumes

    - In-use IP addresses
    - Packet filtering rule sets
    - Number of load balancers
    - Network subnets
    - Log streams
- Gating the cluster scaling actions to bring up new nodes in batches, with a pause between batches, because some cloud providers rate limit the creation of new instances.

Problem Solution Fit

Our solution involves building a cross platform progressive web application that can be used by the users to view the news in a feed based view.

By this, the user gets a dopamine hit whenever the user opens the application.

Our product is ideated in such a way that the users will be able to get only the relevant information and news of their chosen topic thus making sure that the users stay with the app for a longer period of time. The features of our product are as follows:

1. Each user has their own profile which will contain information about their chosen or interested topics, and news will be fetched and displayed from only those topics. Furthermore the topics can later be changed as per the users wish.

2. Displaying various stats regarding the user's analytics based on the viewing of various new articles, which allows the user to know about his own preferences and interests in a deeper manner.

3. Option to save stories which can be seen in the users profile, this will allow the user to keep some news stories for quick reference in the future. This is a very useful feature which saves a lot of time for the user, in case they need to revisit something they read or saw before.

4. Allows the users to follow a particular news publication, so that the user can get to know about all the latest news updates provided by the selected news publication.

5. This application not only focuses on the new readers, it can also act as a portal for the news publication houses to understand their viewership and hence we can monetize this to make the app sustainable for the developers.

Our product will be highly scalable as we have proposed to use IBM’s Cloud services. Because of that, our product can scale for millions of users and work flawlessly. This could help generate more revenue for the product and move the product towards profitability.

Why it solves the problem:

We’ve collected all the relevant negative factors of lack of news tracking statistics,

along with results from studies and reports that have analyzed the issue.
This guide will give you an unbiased look at why the media reports negative news. We'll provide you with an informed and educated overview of the subject in general.

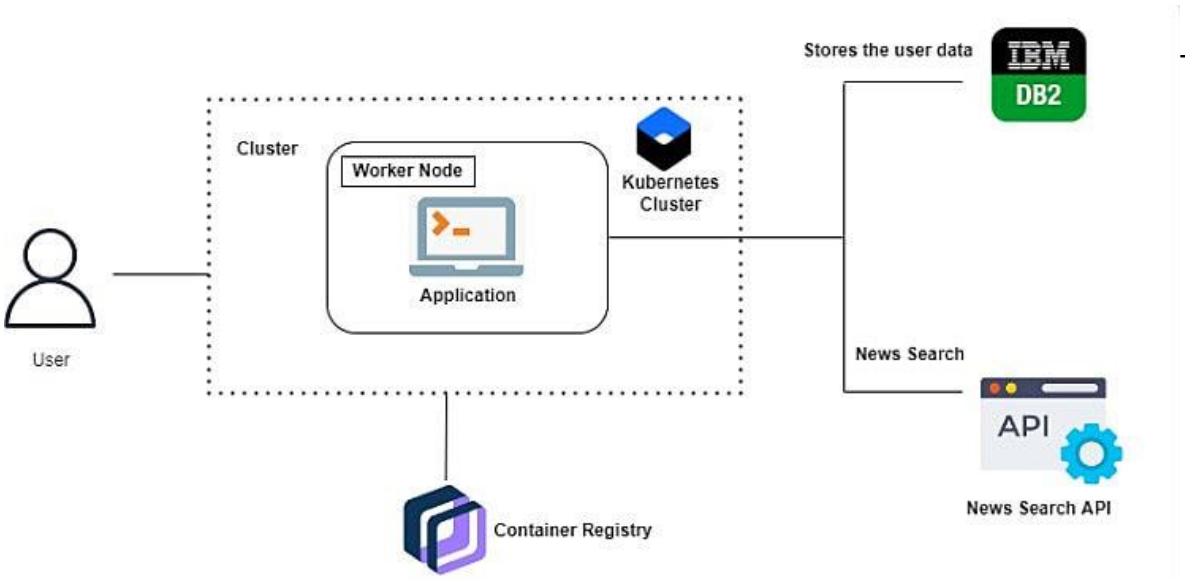
Top Negative News Facts

- Sensationalist stories form 95% of media headlines nowadays.
- Media reports with negative news or statistics catch 30% more attention.
- 26.7% of people exposed to negative news go on to develop anxiety issues.
- 63% of kids aged 12–18 say that watching the news makes them feel bad.
- Moreover, it makes you frustrated when you wake up in the morning and want to know the news in time before getting out to work or college but you are not able to get the required news on time. This may lead to frustration and many people break televisions.

- Nearly 67.92% of the people are not able to track the news they require and a lot of their precious time is wasted which they can use to productively do something.
- Nowadays most of the news are miserable and increases suicidal thoughts and depression seeing them.
- Improper covid 19 figures made several people panic and death due to panic was more than covid deaths.
- So we have built this app where the individual views the news that he likes and wants to know.
- This increases his knowledge and research content in the topic he is willing to invest his time and willingness to know more.

- Lightning News solves this problem and just helps you to focus on the news that the individual required

Architecture:



The proposed solution is a web application, with a ReactJS frontend and a Python Flask backend.

The Flask Server deals with all computational problems such as:

1. Retrieving news articles from various data sources
2. Creating users and registering new new sources
3. Maintaining and updating user data such as preferences, following etc.,

The web application is containerised using docker as container technology supports streamlined build, test, and deployment from the same container images. Also, containerized applications make the application independent of the host environment, thus mitigating the platform/device specific errors.

The containerized web application is then pushed to the IBM Container Registry, which is a service for storing private container images in an easily accessible manner.

A Kubernetes cluster is provisioned on the IBM Cloud Kubernetes Service. This provides easy scalability. Kubernetes clusters support ebay vertical and horizontal scaling, and simple orchestration of containerized applications.

The containerised application requires 3 main services:

1. A Load Balancer to distribute incoming load and provide a Public IP for accessing the application
2. A Web Services to expose the Flask Server
3. A scheduled cron job to periodically collect, filter, organize and classify news from various sources.

IBM DB2, a scalable distributed SQL-based database is used for efficient storage of news articles, metrics and user data.

The news APIs polled include News API, Bing News API, Mediastack etc,

Functional Requirements

The system shall allow viewing curated news articles by all customers.

For customers, this will eliminate the current delay between their search for relevant articles and them getting information. This will reduce the time a customer spends to find news by 10%. The reach of relevant new articles to the people increases by 15%.

The system shall reflect new and changing news articles within 5 minutes of the database being updated by the API Poller. This will reduce the number of incidents of outdated displayed information by 40%. This eliminates the current redundant update of information, leading to money savings.

The system shall display information that is customised based on the user's job function, application and locale. This feature will improve service by reducing the mean number of web pages a user must navigate per session.

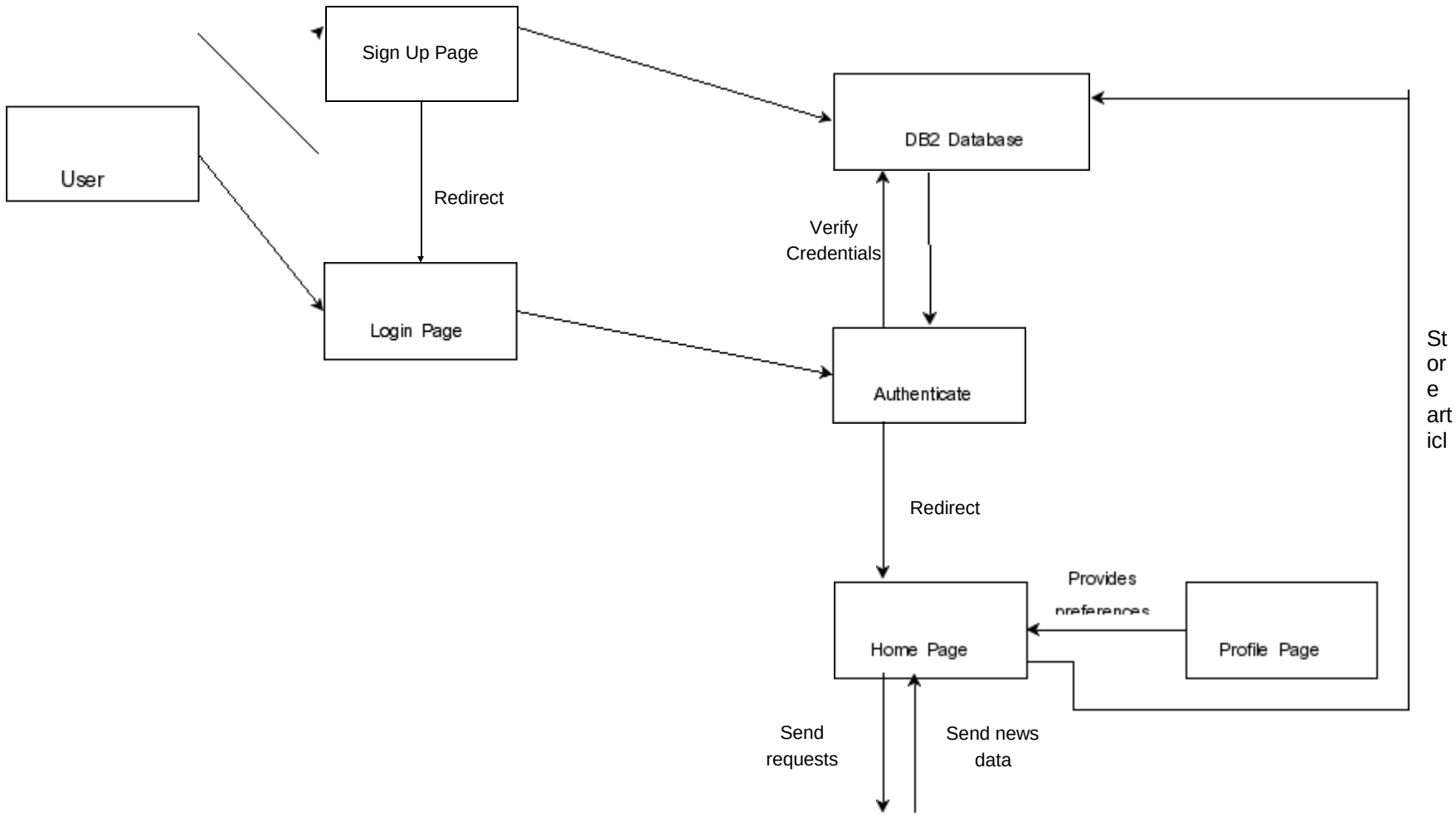
The system shall allow customers to view the source of any new articles. This allows the customer to know the articles are verified.

The system shall allow a customer to directly contact the nearest sales office in his region. This will improve service by reducing the time to respond to a customer request to no more than 1 day.

The system shall provide a search facility that will allow full-text searching of all news articles available. The system must support the following searches:
find all words specified
find any word specified
find the exact phrase
Boolean search

The system shall allow the user's status to be stored for the next time he returns to the web site. This will save the user x minutes per visit by not having to reenter already supplied data.

The system shall translate web pages into the languages of the countries where the user resides.


Sign Up Page
User
Redirect
DB2 Database
Verify
Credentials
Login Page
Authenticate
St
or
e
art
icl
Redirect
Provides
preferences
Home Page
Profile Page
Send
requests
Send news
data

News API

Technology Architecture of News Tracker Application

Arul Bathra M-PSNAET-921319104023
Gokul Krishna J -PSNACET-921319104043
Hariharan S-PSNACET-921319104049
Dhanush Kumar S- PSNACET-921319104038
Guru Sankar B -PSNACET-921319104046

Technologies Used

1. React
2. Flask
3. Docker
4. Kubernetes
5. IBM DB2

Technology Architectural Capabilities

React

- It is a frontend library for creating single page applications for the end-users using JavaScript.
- As the users will be using the news tracker application on their own browsers on PCs or smartphones, the minimum versions of these browsers needs to be considered.
- We propose the users to use Google Chrome as the default browser for using our product. The latest end-of-life version of chrome is 0.3945. We ensure that our product is supported till that version.
- Along with the browser compatibility, the user also need to ensure that they are on a proper network connection so that the application can download and upload necessary data through the frontend which are necessary for smooth functioning of the app.
- As there could be animations used in the frontend with libraries such as *Animate-On-Scroll,* the user's machine should also be able to perform these kind of animations for an overall better user experience.

Flask

- It is a microframework for building backend servers using Python that can server HTTP/RPC requests.
- The user data needs to be stored and the news needs to be fetched from the API. To facilitate all this, we use Flask as our backend server framework.
- As our product has several interesting and innovative features, we are expecting atleast 100,000 to 1,000,000 users in our platform - if built and marketed properly.
- At that scale, we have planned to use asynchronous programming to serve all the requests for smooth functioning of our product.
- We also plan to horizontally scale the system - add more machines that run the flask server which can in turn server numerous clients.
- Other open source libraries are also going to be used, for example - libraries that can perform CRUD on the DB, send HTTP requests, store cookies, cache the data, etc.

Docker

- Docker is an open source containerization platform that uses OS-level virtualization to deliver software in packages called containers.
- As we wish to horizontally scale our product across many machines because of the high user load, we plan to use Docker to containerize our app and use it across the machines.
- By this, horizontal scaling can be achieved easily.

Kubernetes

- Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management.
- It allows you to run your Docker containers and workloads and helps you to tackle some of the operating complexities when moving to scale multiple containers, deployed across multiple servers. This can tremendously help serve the enormous load that our product is expected to experience.

IBM DB2

- Db2 is a family of data management products, including database servers, developed by IBM.
- We are primarily interested in the relational database that DB2 offers.
- We plan to us this for storing user data, news API data, user data cache, session IDs and other advertising data.
- For our product to be up and running all the time along with the database, we plan to shard the database which can help us quickly get outputs for queries made on the DB.
- IBM DB2 could also help us tackle the issue of localization of users where we can host our databases at locations where there are users who consume news for a specific topic. This can significantly improve user experience.

Customer Journey Map

Arul Bathra M-PSNAET-
921319104023
Gokul Krishna J -PSNACET-
921319104043
Hariharan S-PSNACET-
921319104049
Dhanush Kumar S- PSNACET-
921319104038
Guru Sankar B -PSNACET-
921319104046

In our News Tracker application,
The customer has to create an account and login with the credentials. If the customer logs in for the first time then he has to fix his preferences for the news.

If he logs in for further times he would get his page with all the news filtered according to his preferences. These preferences are actually taken as input when the customer logs in for the first time.

The entire news list is skimmed and the most relevant information and matching to the preferences news are shown up to the user. The user can also save and share the news to others.

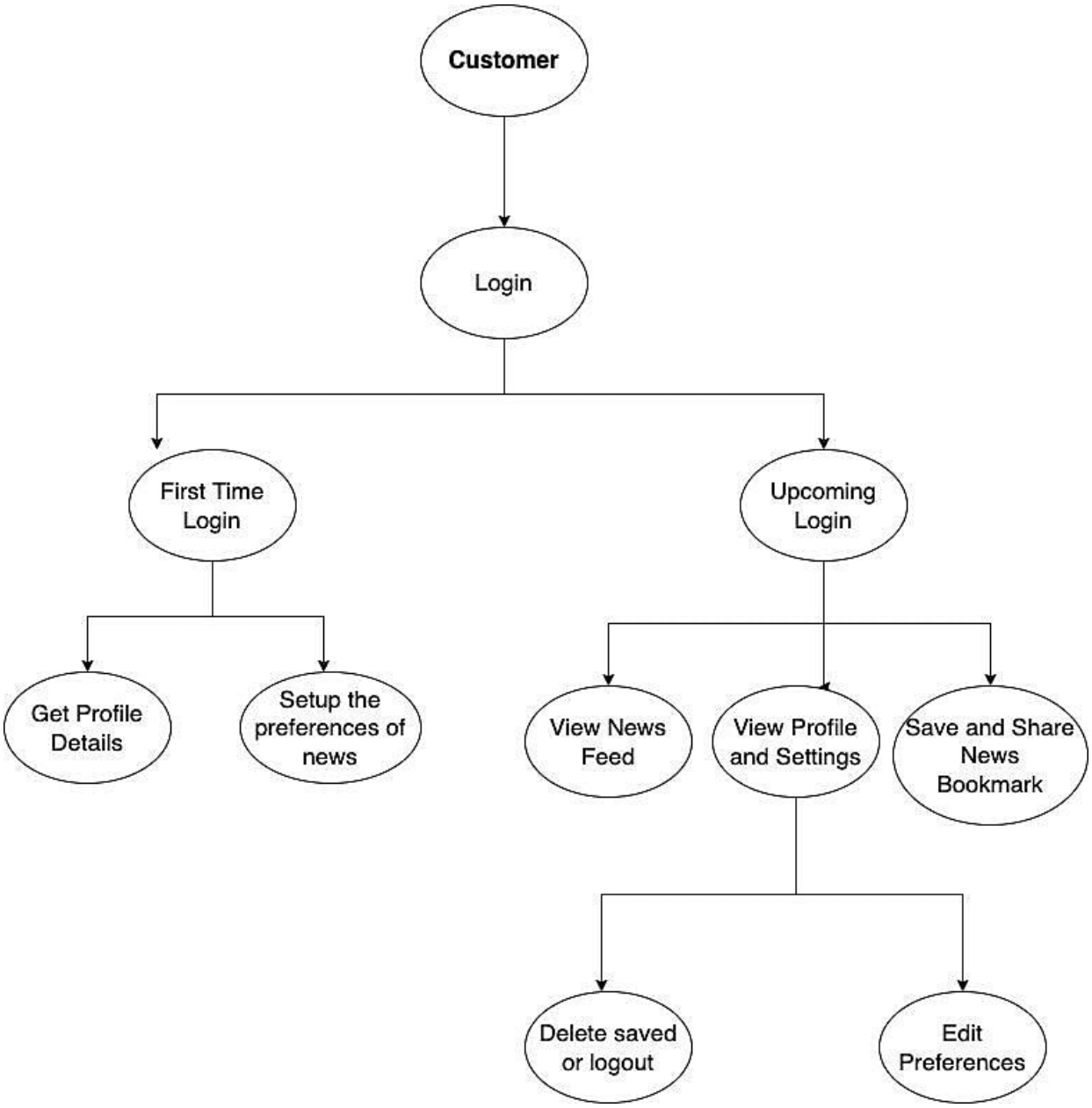
Saving of the news: The user can save or bookmark the news and it will be available as bookmarked under his profile and can be viewed later if it is not available in his feed due to the arrival of new news matching his preferences.

The customer can actually share the news to others as well. Similarly, if he feels the news that he has saved is no longer needed and he wants to remove it from the bookmarked list then he can just remove the bookmark from it. The news is no longer available in the bookmark or saved list.

If the customer feels that he has changed his point of interest in news and wants to change and apply filters to the news accordingly. He can visit his profile page. There would be an option in the profile page to preferences. The preferences keep track of the preferences section of the user or customer. Now the customer is given the rights to modify his or her own profile and preferences.
For the subsequent news filtering and tracking the new preferences of the user is taken into consideration.

The user is also given the option to logout from the application if required and has to login to the application when he comes back to the application.

The diagram of showing the customer journey map



Project Planning Phase

Milestone and Activity List

Date	18 October 2022
Team ID	PNT2022TMID04914
Project Name	News Tracker Application

Title	Description	Date
Literature Survey & 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 list of problem statements	10 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	17 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	24 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	1 OCTOBER 2022
Solution Architecture	Prepare solution architecture document	1 OCTOBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences	8 OCTOBER 2022

	with the application (entry to exit).	
Functional Requirement	Prepare the functional requirement document.	15 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	15 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	15 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project	22 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	19 NOVEMBER 2022

Project Planning Phase

Date	5 November 2022
Team ID	PNT2022TMID04914
Project Name	Project News Tracker Application
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Account Creation	USN-1	As a user, I can create an account on the application.	5	High	Arul Bathra M, Hariharan S
Sprint-1	Login	USN-2	I can successfully login to the application using provided login credentials.	5	High	Gokul Krishna J, Dhanush Kumar s
Sprint-1	Storage	USN-3	As a user, my data will be stored on cloud in IBM Database.	5	High	Arul Bathra M, Hariharan S
Sprint-2	News API integration	USN-4	I need to know the latest news in the app which can be pulled from the News API.	3	Low	Gokul Krishna J, Dhanush Kumar s
Sprint-2	Add routes to display news with respect to user prefere	USN-5 ces	As a user, I want only the news that I prefer.	10	High	Arul Bathra M, Hariharan S
Sprint-3	Front End	USN-6	Create front end for all above listed services an connect them to back end and database.	d 5	Medium	Gokul Krishna J, Dhanush Kumar s
Sprint-3	Front End	USN-7	Create front end for all above listed services an connect them to back end and database.	d 5	Medium	Arul Bathra M, Hariharan S
Sprint-4	Deploy the application	USN-8	Deploy the application to Kubernetes and host the application using IBM services	5	Medium	Gokul Krishna J,

						Dhanush Kumar s
Sprint-4	Additional Features	USN-9	Implement all additional features of the application	5	Low	Jayasooryan, Karun A
Sprint-4	Testing	Testing	Testing all the features of the application.	15	High	Lokesh N N, Moniesh R

Project Tracker, Velocity & Burndown Chart: (4 Marks)

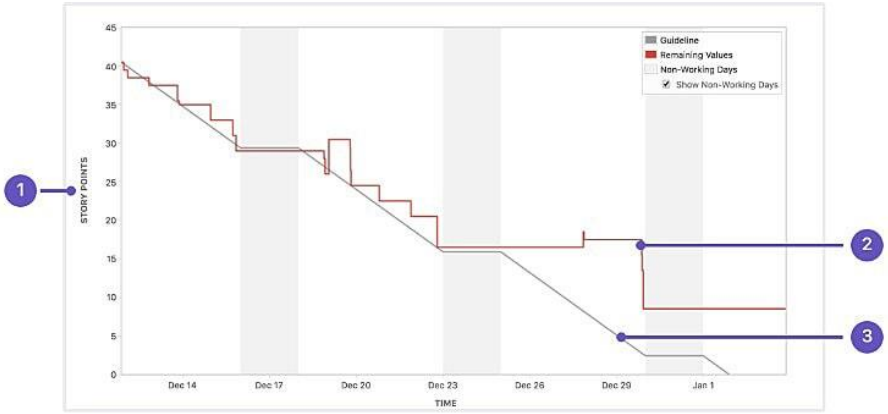
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	15	6 Days	23 Oct 2022	29 Oct 2022	Will be updated as we go.	
Sprint-2	13	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	10	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	25	6 Days	14 Nov 2022	19 Nov 2022		

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

$$AV = (15 + 13 + 10 + 25)/6 = 10.5$$

Burndown Chart:



1. **Estimation statistic**: The vertical axis represents the estimation statistic that you've selected.

2. **Remaining values**: The red line represents the total amount of work left in the sprint, according to your team's estimates.

3. **Guideline**: The grey line shows an approximation of where your team should be, assuming linear progress. If the red line is below this line, congratulations - your team's on track to completing all their work by the end of the sprint. This isn't foolproof though; it's just another piece of information to use while monitoring team progress.

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

https://www.visual-paradigm.com/scrum/scrum-burndown-chart/

https://www.atlassian.com/agile/tutorials/burndown-charts

Projects / News Hub

Roadmap

#* Givefeedback Share Ljt Export

Status category Ep< • Cearflt em View settings

Sprints

NH-1 Sprint 1

NH-10 Account Creation

NH-2 Sprint 2

NH-13 News API Integration

NH-3 Sprint 3

t Create Epic

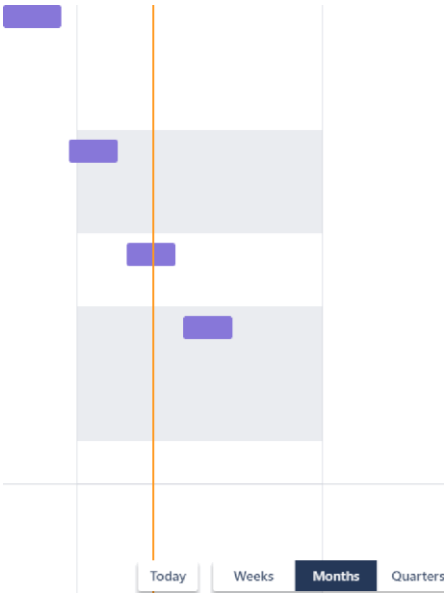
NH-4 Sprint 4

NH-14 Add routes to display news with respect to user preferences

NH-16 Deploy the application

NH-17 Additional Features

NH-18 Testing



CODING & SOLUTIONING

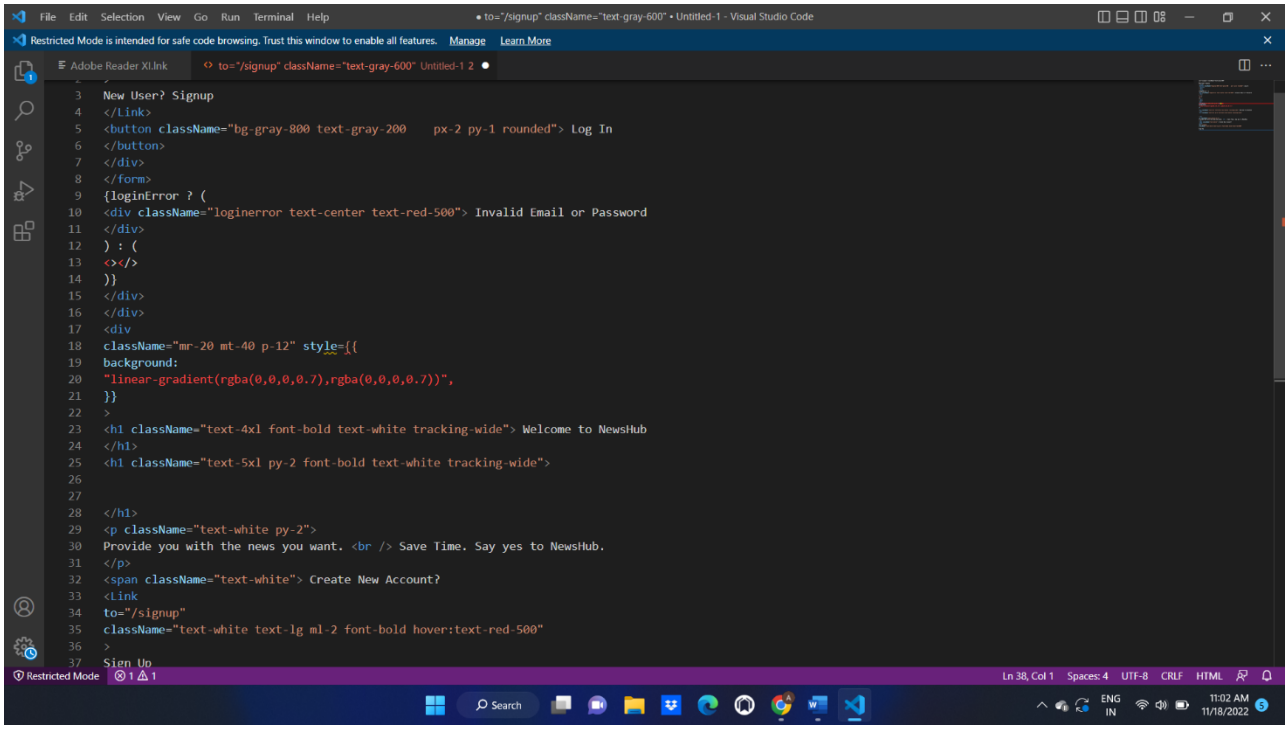
Feature 1:

The NewsHub is an application which provides relevant news for the users. The Feature provides a login page which authenticates the users of the system. They have to provide their credentials, that is their email and password. If they provide the correct credentials, they are logged in to the page, else an error message “Invalid Credentials” is displayed.
Once the user is logged into the system, the system shows the list of news related to the users preferences.

Login Page

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';

const Login = () => {

 const [email, setEmail] = useState();
 const [password, setPassword] = useState();
 const [loginError, setLoginError] = useState();

 const loginHandler = (e) => {
   e.preventDefault();
   fetch('http://169.51.205.76:32522/authenticate', {
     method: 'POST',
     mode: 'cors',
     redirect: 'manual',
     body: JSON.stringify({
       email: email,
       password: password
     })
   }).then((res) => {
     if (res.status === 200) {
       sessionStorage.setItem('@user', email);
       window.location.reload();
     }
```

```
    else setLoginError(true);
  })

  if (loginError) {
    setLoginError(true);
  }
}

return (
  <>
    <div className="w-full flex items-center justify-between">
      <div className="pt-40 pl-20">
        <div className="card bg-white shadow-md rounded-lg px-4 py-4 mb-6 ">
          <form onSubmit={loginHandler}>
            <div className="flex items-center justify-center">
              <h2 className="text-2xl font-bold tracking-wide">
                Welcome back
              </h2>
            </div>
            <h2 className="text-xl text-center font-semibold text-gray-800 mb-2">
              Log In
            </h2>
            <input
              onChange={(event) => {
                setEmail(event.target.value);
              }}
              type="text"
              className="rounded px-4 w-full py-1 bg-gray-200  border border-gray-400
mb-6 text-gray-700 placeholder-gray-700 focus:bg-white focus:outline-none"
              placeholder="Email id"
            />
            <input
              onChange={(event) => {
                setPassword(event.target.value);
              }}
              type="password"
              className="rounded px-4 w-full py-1 bg-gray-200  border border-gray-400
mb-4 text-gray-700 placeholder-gray-700 focus:bg-white focus:outline-none"
              placeholder="Password"
            />
            <div className="flex items-center justify-between">
              <Link
```




```
            </Link>
          </span>
        </div>
      </div>
    </>
  )
}

export default Login;
```

The system also provides the signup page for first time users. The signup page gets the user information, his news preferences as well. Once the user signs up to the page his data is added to the IBM DB2.

```
import React, { useState } from "react";
import { Link } from "react-router-dom";
import Multiselect from 'multiselect-react-dropdown';
const Signup = () => {
 const [name, setName] = useState();
 const [email, setEmail] = useState();
 const [phone, setPhone] = useState();
 const [password, setPassword] = useState();
 const [confirm_password, setConfirmPassword] = useState();
 const [dob, setDob] = useState();
 const [options, setOptions] = useState([{ name: 'business', id: 1 }, { name:
'entertainment', id: 2 }, { name: 'general', id: 3 }, { name: 'health', id: 4 }, {
name: 'science', id: 5 }, { name: 'sports', id: 6 }, { name: 'business', id: 7 }, {
name: 'technology', id: 8 }]);
 const [selectedVal, setSelectedVal] = useState([]);

 const signupHandler = (e) => {
   e.preventDefault();
   if (password == confirm_password) {
     const payload = {
       email: email,
       password: password,
       options: selectedVal,
       name: name,
       phone: phone,
       dob: dob
     }
```

```
    fetch('http://169.51.205.76:32522/sign-up', {
      method: 'POST',
      mode: 'cors',
      redirect: 'manual',
      body: JSON.stringify(payload)
    }).then((res) => {
      if (res.status === 200) window.location.replace("/");
    })
  }
}
const onSelect = (selectedList, selectedItem) => {
  selectedVal.push(selectedItem)
}

const onRemove = (selectedList, removedItem) => {
  const index = selectedVal.indexOf(removedItem);
  if (index > -1) { // only splice array when item is found
    selectedVal.splice(index, 1); // 2nd parameter means remove one item only
  }
}

return (
  <>
    <div className="min-h-screen flex flex-col">
      <div className="m-20 bg-white container max-w-lg mx-auto flex-1 flex flex-col
items-center justify-center px-2">
        <div className="px-6 py-8 rounded shadow-md text-black w-full">
          <h1 className="mb-8 text-3xl text-center">Sign up</h1>
          <input
            type="text"
            className="block border border-grey-light w-full p-3 rounded mb-4"
            name="fullname"
            placeholder="Full Name"
            onChange={(e) => {
              setName(e.target.value);
            }}
          />

          <input
            type="text"
            className="block border border-grey-light w-full p-3 rounded mb-4"
            name="email"
```

```
          placeholder="Email"
          onChange={(e) => {
            setEmail(e.target.value);
          }}
        />

        <input
          type="text"
          className="block border border-grey-light w-full p-3 rounded mb-4"
          name="phoneno"
          placeholder="Phone Number"
          onChange={(e) => {
            setPhone(e.target.value);
          }}
        />

        <input
          type="password"
          className="block border border-grey-light w-full p-3 rounded mb-4"
          name="password"
          placeholder="Password"
          onChange={(e) => {
            setPassword(e.target.value);
          }}
        />
        <input
          type="password"
          className="block border border-grey-light w-full p-3 rounded mb-4"
          name="confirm_password"
          placeholder="Confirm Password"
          onChange={(e) => {
            setConfirmPassword(e.target.value);
          }}
        />

        <input
          type="date"
          className="block border border-grey-light w-full p-3 rounded mb-4"
          name="dob"
          placeholder="Date of Birth"
          onChange={(e) => {
            setDob(e.target.value);
```

```
            }}
          />

          <Multiselect
            options={options}
            selectedValues={selectedVal}
            onSelect={onSelect}
            onRemove={onRemove}
            displayValue="name"
          />
          <button
            type="submit"
            onClick={signupHandler}
            className="mt-10 w-full text-center py-3 rounded bg-green-500 text-white
hover:bg-green-dark focus:outline-none my-1"
          >Create Account</button>

              <div className="text-center text-sm text-grey-dark
            mt-4"> By signing up, you agree to the  
            <a className="no-underline border-b border-grey-dark text-grey-dark"
href="#">
              Terms of Service
            </a> and  
            <a className="no-underline border-b border-grey-dark text-grey-dark"
href="#">
              Privacy Policy
            </a>  
          </div>
        </div>

        <div className="text-grey-dark mt-6">
          Already have an account?
          <Link to="/" className="no-underline border-b border-blue text-blue"
href="../login/">
            Log in
          </Link>.
        </div>
      </div>
    </div>
  </>
 )
}
```

```
export default Signup;
```

NewsFeed Page:

```
import React, { useState, useEffect } from 'react';
import News from '../component/News';
import axios from "axios";
import { CgProfile } from "react-icons/cg";
import { Link } from 'react-router-dom';

const NewsFeed = () => {
 const [articles, setArticles] = useState([])
 const [fetched, setFetched] = useState(false);
 const [likedArticles, setLikedArticles] = useState();
 const [bookedArticles, setBookedArticles] = useState();
 const [changed, setChanged] = useState(false);
 useEffect(() => {
   const fn = () => {
     axios.get("http://169.51.205.76:32522/get-top-headlines?userName=" +
sessionStorage.getItem('@user')).then((data, error) => {
       axios.get("http://169.51.205.76:32522/profile?userName=" +
sessionStorage.getItem('@user')).then((profileData, err) => {
         console.log(profileData);
         let likedList = [], bookedList = [];
         for(const article of profileData?.data.likes){
           var name = article.NEWS_ARTICLE_LINK;
           var obj = {};
           obj[name] = true;
           likedList = {
             ...likedList,
             ...obj
           };
         }
         for(const article of profileData?.data.bookmarks){
           var name = article.NEWS_ARTICLE_LINK;
           var obj = {};
           obj[name] = true;
           bookedList = {
             ...bookedList,
```

```
            ...obj
          };
        }
        setLikedArticles(likedList);
        setBookedArticles(bookedList);
        setArticles(data.data.articles);
        setFetched(true);
      })
    })
  }
  fn();
}, [])

return (
  <div className='w-full flex'>
    <div className='m-auto justify-center p-20 rounded-lg'>
      <Link to="/profile"><CgProfile className='text-white text-5xl float-right
-mr-3/4 -mt-16' /></Link>
      {fetched ?
        <>
          {articles.map((item, index) => {
            if(index==0)
            console.log('lister',likedArticles,bookedArticles);
            return (
              <News new={{
                ...item,
                liked: likedArticles[`${item.url}`],
                bookmark: bookedArticles[`${item.url}`]
              }} key={index} />
            )
          }
          )
          }
        </>
        : null}
    </div>
  </div>
);
}

export default NewsFeed;
```

Flask Backend API for the authentication pages:

```
from flask import Flask, request, Response, send_from_directory
import requests
import os
import random
import json
from flask_cors import CORS
import db_crud
app = Flask(_name_) #, static_folder='build', template_folder='build',
static_url_path='')
CORS(app)

NEWS_API_TOPHEADLINES_ENDPOINT = 'https://newsapi.org/v2/top-headlines'
NEWS_API_KEY = 'aeb9762b06d74d1a8ece0f3b896feb4c'
user_preferences =
['business','entertainment','general','health','science','sports','technology']

@app.route('/get-top-headlines', methods=['GET'])
def get_top_headlines_for_user():
 query = request.args.to_dict()
 if query.get('userName', -1) == -1:
   return Response('userName must be provided in query string', status=400)
 country = query.get('country', 'in')
 q = query.get('q', '')
  user_topics = db_crud.get_topics(query.get('userName'))
 user_prefs = 'category=' + '&category='.join(user_topics)
 url =
NEWS_API_TOPHEADLINES_ENDPOINT+f'?country={country}&{user_prefs}&apiKey={NEWS_API_KEY}
'
 if q != '':
   url += f'&q={q}'
 print(url)
  api_response = requests.get(url=url)
 return json.loads(api_response.content.decode()), 200

@app.route('/authenticate', methods=['POST'])
def authenticate_user():
 form = json.loads(request.data.decode())
 if db_crud.authenticate(form['email'], form['password']):
   return {'status':'success'}, 200
 return {'status':'failure'}, 400
```

```
@app.route('/sign-up', methods=['POST']) def
sign_up_user():
 form = json.loads(request.data.decode()) print(form)
 if db_crud.create_account(form['name'], form['email'], form['password'],
form['phone'], form['dob'], [x['name'] for x in form['options']]):
   return {'status':'success'}, 200
 return {'status':'failure'}, 400

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

Feature 2:

Apart from providing the relevant news for the users. We also provide options of liking, bookmarking and sharing of the news article. If the user reaches his profile page he has the option to view his past liked and bookmarked articles. When the user shares the news the app automatically opens up the whatsapp API and helps the user to share the news to the person he wants through Whatsapp.

The user can as well unlike and unbookmark the news articles. The user also has provision to update the news preferences if he wishes to do so.

After updating the profile that is the news preference when the person goes to the news page it covers all the articles that are related to the updated news preference. The user can logout from the system.

Profile page:

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import 'react-responsive-carousel/lib/styles/carousel.min.css'
import { Carousel } from 'react-responsive-carousel';
import News from "../component/News";
import Multiselect from "multiselect-react-dropdown";
```

```
const Profile = () => {
 const [fetched, setFetched] = useState(false);
 const [data, setData] = useState();
 const [likedArticles, setLikedArticles] = useState([]);
 const [bookedArticles, setBookedArticles] = useState([]);
 const [options, setOptions] = useState([{ name: 'business', id: 1 }, { name:
'entertainment', id: 2 }, { name: 'general', id: 3 }, { name: 'health', id: 4 }, {
name: 'science', id: 5 }, { name: 'sports', id: 6 }, { name: 'business', id: 7 }, {
name: 'technology', id: 8 }, { name: 'Virat Kohli', id: 9 }, { name: 'Narendra Modi',
id: 10 }, { name: 'Rain', id: 13 }, { name: 'MK Stalin', id: 12 }, { name: 'Cinema',
id: 13 }, { name: 'Floods', id: 14 }, { name: 'politics', id: 15 }, { name: 'Donald
Trump', id: 16 }, { name: 'Putin', id: 17 }, { name: 'Ukraine-Russia', id: 18 }, {
name: 'Biden', id: 19 }, { name: 'ADMK', id: 20 }, { name: 'Rahul Gandhi', id: 21 }, {
name: 'China', id: 22 }, { name: 'corona', id: 23 }, { name: 'elon musk', id: 24 }, {
name: 'worldcup', id: 25 }, { name: 'BJP', id: 26 }, { name: 'Taiwan China crisis',
id: 27 }, { name: 'job opputunities', id: 28 }, { name: 'tourism', id: 29 }, { name:
'metroplian', id: 30 }]);
 const [selectedVal, setSelectedVal] = useState([]);
 useEffect(() => {
   const fn = async () => {
     const profileData = await
axios.get("http://169.51.205.76:32522/profile?userName=" +
sessionStorage.getItem('@user'));
     console.log('data', profileData.data.topics);
     setData(profileData.data);
     const selval = [];
     let ids = 0;
     for (const topic of profileData.data.topics) {
       selval.push({
         name: topic,
         id: ids
       })
       ids++;
     }
     setSelectedVal(selval);
     const likedLinks = profileData.data?.likes.reduce((prev, cur) => {
       return prev.concat(cur.NEWS_ARTICLE_LINK);
     }, [])
     const bookmarkedLinks = profileData.data?.bookmarks.reduce((prev, cur) => {
       return prev.concat(cur.NEWS_ARTICLE_LINK);
     }, [])
     setLikedArticles(likedLinks);
```

```
    setBookedArticles(bookmarkedLinks);
    setFetched(true);
  }
  fn();
}, [])
const onSelect = (selectedList, selectedItem) => {
  selectedVal.push(selectedItem)
}

const onRemove = (selectedList, removedItem) => {
  let newVal = [];
  for(const topic of selectedVal){
    if(topic.id==removedItem.id){
      console.log(topic,removedItem);
      continue;
    }
    newVal.push(topic);
  }
  setSelectedVal(newVal);
}
return (
  <>
    {fetched ?
      <div>
        <div className="pt-20">
          <div className="px-20">
            <div className="flex flex-col min-w-0 break-words bg-white w-full mb-6
shadow-xl rounded-lg">
              <div className="px-6">
                <div className="flex flex-wrap justify-center">
                  <div className="w-full lg:w-3/12 px-4 lg:order-2 flex
justify-center">
                    <div>
                      <img width={180} height={180} alt="..."
src="https://64.media.tumblr.com/bed5455fdd7789656247fb01ed60ad31/93c34c0b6d121aef-6a/
s400x600/15a48a245b0ef5bb4f5215d177116cf9c4150531.png" className="shadow-xl
rounded-full h-auto align-middle border-none absolute -m-16 -ml-20 lg:-ml-16
max-w-150-px" />
                    </div>
                  </div>
                  <div className="w-full lg:w-4/12 px-4 lg:order-3 lg:text-right
lg:self-center">
```

```
                    <div className="py-6 px-3 mt-32 sm:mt-0">
                      <button onClick={() => {
                        sessionStorage.removeItem('@user');
                        window.location.replace('/');
                      }} className="bg-pink-500 active:bg-pink-600 uppercase
text-white font-bold hover:shadow-md shadow text-xs px-4 py-2 rounded outline-none
focus:outline-none sm:mr-2 mb-1 ease-linear transition-all duration-150"
type="button">
                        Logout
                      </button>
                    </div>
                  </div>
                  <div className="w-full lg:w-4/12 px-4 lg:order-1">
                    <div className="flex justify-center py-4 lg:pt-4 pt-8">
                      <div className="mr-4 p-3 text-center">
                        <span className="text-xl font-bold block uppercase
tracking-wide text-blueGray-600">{data?.likes?.length}</span><span className="text-sm
text-blueGray-400">Likes</span>
                      </div>
                      <div className="mr-4 p-3 text-center">
                        <span className="text-xl font-bold block uppercase
tracking-wide text-blueGray-600">{data?.bookmarks?.length}</span><span
className="text-sm text-blueGray-400">Bookmarks</span>
                      </div>
                    </div>
                  </div>
                </div>
                <div className="text-center mt-12">
                  <h3 className="text-4xl font-semibold leading-normal mb-2
text-blueGray-700 mb-2">
                    {data?.user.NAME}
                  </h3>
                  <div className="text-sm leading-normal mt-0 mb-2 text-blueGray-400
font-bold uppercase">
                    <i className="fas fa-map-marker-alt mr-2 text-lg
text-blueGray-400"></i>
                    Date of Birth : {data?.user.DOB}
                  </div>
                  <div className="mb-2 text-blueGray-600">
                    <i className="fas fa-briefcase mr-2 text-lg
text-blueGray-400"></i> Email : {data?.user.USERNAME}
                  </div>
```

```
                  <div className="mb-2 text-blueGray-600">
                    <i className="fas fa-university mr-2 text-lg
text-blueGray-400"></i>Password : {'*'.repeat(data?.user.PASSWORD?.length)}
                  </div>
                  <div className="mb-2 text-blueGray-600">
                    <i className="fas fa-university mr-2 text-lg
text-blueGray-400"></i>Topics: {data?.user.TOPICS}
                  </div>
                </div>
                {data?.likes.length > 0 ?
                  <div className="mt-10 py-10 border-t border-blueGray-200
text-center">
                    <div className="flex flex-wrap justify-center">
                      <h1 className="text-4xl font-semibold leading-normal mb-2
text-blueGray-700 mb-2">Articles Liked</h1>
                      <Carousel autoPlay>
                        {data?.likes.map((item, index) =>
                          <News new={{
                            title: item.NEWS_TITLE,
                            url: item.NEWS_ARTICLE_LINK,
                            description: item.NEWS_DESCRIPTION,
                            urlToImage: item.NEWS_IMAGE_LINK,
                            publishedAt: item.NEWS_DATE,
                            liked: true,
                            bookmark:
bookedArticles?.includes(item.NEWS_ARTICLE_LINK)
                          }} key={index} />
                        )}
                      </Carousel>
                    </div>
                  </div>
                  : null}
                {data?.bookmarks?.length ?
                  <div className="w-full mt-10 py-10 border-t border-blueGray-200
text-center">
                    <div className="flex justify-center">
                      <h1 className="text-center text-4xl font-semibold
leading-normal mb-2 text-blueGray-700 mb-2">Articles BookMarked</h1>
                      <Carousel autoPlay>
                        {data?.bookmarks.map((item, index) =>
                          <News new={{
                            title: item.NEWS_TITLE,
```

```
                            url: item.NEWS_ARTICLE_LINK,
                            description: item.NEWS_DESCRIPTION,
                            urlToImage: item.NEWS_IMAGE_LINK,
                            publishedAt: item.NEWS_DATE,
                            liked: likedArticles?.includes(item.NEWS_ARTICLE_LINK),
                            bookmark: true
                          }} key={index} />
                        )}
                      </Carousel>
                    </div>
                  </div>
                  : null}
                <div className="text-center mb-40">
                  <h1 className="text-4xl font-semibold leading-normal mb-2
text-blueGray-700 mb-2">Update Tags</h1>
                  <Multiselect
                    options={options}
                    selectedValues={selectedVal}
                    onSelect={onSelect}
                    onRemove={onRemove}
                    displayValue="name"
                  />
                  <div className="flex justify-center mt-40">
                    <button onClick={async() => {
                      let topicSelected = [];
                      for(const topics of selectedVal){
                        topicSelected.push(topics.name);
                      }
                      const payload = {
                        user: sessionStorage.getItem('@user'),
                        topics: topicSelected
                      }
                      await
axios.post("http://169.51.205.76:32522/update-profile",payload);
                    }} className="bg-pink-500 active:bg-pink-600 uppercase text-white
font-bold hover:shadow-md shadow text-xs px-4 py-2 rounded outline-none
focus:outline-none sm:mr-2 mb-1 ease-linear transition-all duration-150"
type="button">
                      Update
                    </button>
                  </div>
                </div>
```

API for Action and Update profile

```
              </div>
            </div>
          </div>
          <div className="relative bg-blueGray-200 pt-8 pb-6 mt-8">
            <div className="container mx-auto px-4">
              <div className="flex flex-wrap items-center md:justify-between
justify-center">
                <div className="w-full md:w-6/12 px-4 mx-auto text-center">
                  <div className="text-sm text-white font-semibold py-1">
                    Made by Saitama Squad
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
      : null}
   </>
 )
}

export default Profile;
```

```
from flask import Flask, request, Response, send_from_directory
import requests
import os
import random
import json
from flask_cors import CORS
import db_crud
app = Flask(_name_)
CORS(app)

NEWS_API_TOPHEADLINES_ENDPOINT = 'https://newsapi.org/v2/top-headlines'
NEWS_API_KEY = 'aeb9762b06d74d1a8ece0f3b896feb4c'
user_preferences =
['business','entertainment','general','health','science','sports','technology']

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

```
def user_profile():
 query = request.args.to_dict()
 ret = {
   "likes" : db_crud.get_likes(query['userName']),
   "bookmarks" : db_crud.get_bookmarks(query['userName']),
   "topics": db_crud.get_topics(query['userName']),
   "user" : db_crud.get_user(query['userName'])
 }
 return ret, 200

@app.route('/action', methods=["POST"])
def action():
 form = request.data.decode()
 form = json.loads(form)
 print(form)
 if(form['type']=='A'):
   print("ADD")
   if db_crud.add_action(form['email'], form['title'], form['url'],
form['urlToImage'], form['publishedAt'], form['description'], form['action']):
     return {'status':'success'}, 200
   else:
return {'status':'failure'}, 400 if(form['type']=='R'):
```

```
print("REMOVE")
if db_crud.remove_action(form['email'], form['url']): return
{'status':'success'}, 200
else:
return {'status':'failure'}, 400


@app.route('/update-profile', methods=['POST']) def updateprofile():
form = json.loads(request.data.decode())
db_crud.update_topics(form['user'], form['topics']) return
{'status':'success'}, 200

if  name    == ' main ': app.run(debug=True)
```

DB Schema:

```
CREATE TABLE USERS(
name varchar(255),
username varchar(255) not null primary key,
password varchar(255),
phno varchar(10),
dob varchar(10),
topics varchar(10000)
);

CREATE TABLE NEWS(
user varchar(255),
news_title varchar(255),
news_article_link varchar(1024),
news_image_link varchar(1024),
news_date varchar(255),
news_description varchar(255),
news_type varchar(1) check (news_type in ('B','L')) ,
foreign key(user) references USERS(username)
);
```

Acceptance Testing
UAT Execution & Report Submission

Date	03 November 2022
Team ID	PNT2022TMID04914
Project Name	Project – News Tracker Application
Maximum Marks	4 Marks

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	0	0	0	0	0
Duplicate	5	0	0	0	5
External	0	1	0	1	2
Fixed	10	1	3	5	19
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	1	0	1
Totals	15	2	4	6	27

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
Client Application	10	0	0	10
News API	3	0	0	3
Auth APIs	5	0	0	5

Connection between client and server	10	0	0	10
Kubernetes deployment	5	0	0	5
Version Control	1	0	0	1

				Date	3-Nov-22								
				Team ID	PNT2022TMID04914								
				Project Name	Project – News Tracker Application								
				Maximum Marks	4 marks								
Test case ID	**Feature Type**	**Component**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Commets**	**TC for Automation(Y/N)**	**BUG ID**	**Executed By**
LoginPage_TC_OO 1	Functional	Login page	Verify user is logged in when user clicked on My account button		1. Enter URL and click go 2. Click on My Account button 3.Verify username displayed	Username: test@gmail.com password: Testing123	Username should be there	Working as expected	Pass				Moniesh R
LoginPage_TC_OO 2	UI	Login page	Verify the UI elements in Login/Signup popup		1. Enter URL and click go 2. Verify login/Singup popup with UI elements		Application should show below UI elements: a.email text box b.password text box c.Login button with black colour d.New User? Signup link	Working as expected	Pass				Lokesh N N
LoginPage_TC_OO 3	Functional	Login page	Verify user is able to log into application with Valid credentials		1.Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password text box 4. Click on login button	Username: test@gmail.com password: Testing123	User should navigate to homepage	Working as expected	Pass				Jayasooryan S
LoginPage_TC_OO 4	Functional	Login page	Verify user is unable to log into application with InValid credentials		1.Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password text box 4. Click on login button	Username: test@gmail.com password: Testing1234	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Karun A
HomePage_TC_OO1	UI	Home page	User is able to see the news	User has selected their preferences earlier	1.Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password text box 4. Click on login button	Username: test@gmail.com password: Testing123	Application should show all the news t	Working as expected	Pass				Moniesh R
HomePage_TC_OO2	Functional	Home page	User is able to like a news article	User has selected their preferences earlier	1. Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password	Username: test@gmail.com password: Testing123	App should save the liked news	Working as expected	Pass				Lokesh N N
HomePage_TC_OO3	Functional	Home page	User is able to bookmark a news ar	User has selected their preferences earlier	1. Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password	Username: test@gmail.com password: Testing123	App should save the bookmarked new	Working as expected	Pass				Jayasooryan S
ProfilePage_TC_OO1	Functional	Profile Page	User is able to view liked news	User has selected their preferences earlier	1. Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password	Username: test@gmail.com password: Testing123	App should save the bookmarked new	Working as expected	Pass				Karun A
ProfilePage_TC_OO2	Functional	Profile Page	User is able to view bookmarked ne	User has selected their preferences earlier	1.Enter URL(http://169.51.205.76:32522/) and click go 2. Enter Valid username/email in Email text box 3. Enter valid password in password	Username: test@gmail.com password: Testing123	App should save the bookmarked new	Working as expected	Pass				Moniesh R

Performance Metrics

NFT - Risk Assessment									
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Authentication	New	High	No Changes	High	Nil	>10 to 30%	ORANGE	Auth change is crucial to the platform
2	News API	New	High	No Changes	High	Nil	>10 to 30%	RED	News API is the core of the platform
3	Profile page	New	High	No Changes	High	Nil	>10 to 30%	GREEN	Profile page is required for a user.

NFT - Detailed Test Plan				
S.No	Project Name	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	News Tracker Application	End to end manual testing	1. The user knows to operate a browser. 2. The user has internet. 3. The user only selects the domains they like.	Approved.

Advantages

1. It is a Webapp hence no need to download the app, can be used in a normal browser.

2. Stores all the data on cloud, hence no usage of unnecessary device space, hence no hassle for the user.

3. Saves a lot of time by showing only the related news articles and preferred topics.

4. Completely dynamic and customizable, the users requirements are always fulfilled.

5. Trusted news articles are the only ones that make it to the users screen, so rare occurrences of mis-information.

Disadvantages

1. The app does not cache anything, hence even bookmarked or liked articles need internet access to be opened.

2. Occurrences of error in classification of news, as the app only displays the returned value from the API call.

3. Failure of the API might result in crash of the news feed page, while the links of saved and liked might still be accessible.

Conclusion

We conclude that our project will be helpful for the end user in the following ways:

- The user can see the latest news of their choice.
- The user can add/remove their preferences.
- The user can like a news article to show support.
- The user can bookmark a news article and view it when they like it.

We hope that with all these features we are able to develop a product that can reach millions of people and improve the news reading experience.

Future Scope for the project:

The application developed fulfills the promised features and satisfies the requirements.

But there can be some more improvement to it, which we would like to keep under future scope.

Future Scope and improvements:

1. Addition of more topics for the user to choose from.

2. Perform a test whether the labeled article is correctly labeled or not, else we shouldn’t provide it to the feed of a user.

3. Integrating various other sources of APIs along with the News API, for diversification and more trust.

4. Caching of news articles and ability to download them as pdf so that the user can read them even while being offline.

5. Ability to webscrape more results related to an obtained article, so that a user might continue to read on the same topic, but from a different resource.

Appendix

Source Code

Client folder

Src folder

App.js

```
import { history } from "./history";
import { Route, Router } from "react-router-dom";
import Login from "./pages/Login";
import Profile from "./pages/Profile";
import NewsFeed from
"./pages/NewsFeed"; import Signup
from "./pages/Signup";

const App = () => {
  return (
    <>
      {sessionStorage.getItem('@user') ?
        <>
          <Router history={history}>
            <Route path="/profile" exact component={Profile} />
            <Route path="/" exact component={NewsFeed} />
          </Router>
        </> :
        <Router history={history}>
          <Route path="/" exact component={Login} />
          <Route path="/signup" exact component={Signup} />
        </Router>
      }
    </>
  );
}

export default App;
```

Index.css

```
@tailwind base;
@tailwind
import { createBrowserHistory } from "history";


export const history =createBrowserHistory();

html {
  font-family: 'Montserrat', sans-
  serif; min-height: 100vh;
  background: url("./assets/bgimg.jpeg")
  !important; background-repeat:no-repeat;
  background-size:cover;
  background-position:center;


}


body {
  margin: 0;
  background: linear-gradient(rgba(0,0,0,0.4),rgba(0,0,0,0.4))
  !important; min-height: 100vh;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
    'Oxygen', 'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica
    Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}


code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier
    New', monospace;
}
```

Pages folder

Login.js

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';

const Login = () => {

  const [email, setEmail] = useState();
  const [password, setPassword] = useState();
  const [loginError, setLoginError] = useState();

  const loginHandler = (e) => {
    e.preventDefault();
    fetch('http://169.51.205.76:32522/authenticate', {
      method: 'POST',
      mode: 'cors',
      redirect: 'manual',
      body: JSON.stringify({
        email: email,
        password: password
      })
    }).then((res) => {
      if (res.status === 200) {
        sessionStorage.setItem('@user', email);
        window.location.reload();
      }
      else setLoginError(true);
    })

    if (loginError) {
      setLoginError(true);
    }
  }

  return (
    <>
      <div className="w-full flex items-center justify-between">
        <div className="pt-40 pl-20">
          <div className="card bg-white shadow-md rounded-lg px-4 py-4 mb-6 ">
```

```
          <form onSubmit={loginHandler}>
            <div className="flex items-center justify-center">
              <h2 className="text-2xl font-bold tracking-
                wide"> Welcome back
              </h2>
            </div>
            <h2 className="text-xl text-center font-semibold text-gray-800
mb-
2">             Log In
            </h2>
            <input
              onChange={(event) => {
                setEmail(event.target.value);
              }}
              type="text"
              className="rounded px-4 w-full py-1 bg-gray-200  border
border-gray-400 mb-6 text-gray-700 placeholder-gray-700 focus:bg-
white focus:outline-none"
              placeholder="Email id"
            />
            <input
              onChange={(event) => {
                setPassword(event.target.value);
              }}
              type="password"
              className="rounded px-4 w-full py-1 bg-gray-200
                                                                  borde
r border-gray-400 mb-4 text-gray-700 placeholder-gray-700 focus:bg-
white focus:outline-none"
              placeholder="Password"
            />
            <div className="flex items-center justify-between">
              <Link
                to="/signup"
                className="text-gray-600"
              >
                New User? Signup
              </Link>
              <button className="bg-gray-800 text-gray-200    px-2 py-1
                rounded"> Log In
```

```
              </button>
            </div>
          </form>
          {loginError ? (
            <div className="loginerror text-center text-red-
              500"> Invalid Email or Password
            </div>
          ) : (
            <></>
          )}
        </div>
      </div>
      <div
        className="mr-20 mt-40 p-
        12" style={{
          background:
            "linear-gradient(rgba(0,0,0,0.7),rgba(0,0,0,0.7))",
        }}
      >
        <h1 className="text-4xl font-bold text-white tracking-
          wide"> Welcome to NewsHub
        </h1>
        <h1 className="text-5xl py-2 font-bold text-white tracking-wide">

        </h1>
        <p className="text-white py-2">
          Provide you with the news you want. <br /> Save Time. Say yes to
NewsHub.
        </p>
        <span className="text-white">
          Create New Account?
          <Link
            to="/signup"
            className="text-white text-lg ml-2 font-bold hover:text-red-
            500"
          >
            Sign Up
          </Link>
        </span>
      </div>
```

```
      </div>
    </>
  )
}

export default Login;
```

NewsFeed.js

```
import React, { useState, useEffect } from 'react';
import News from '../component/News';
import axios from "axios";
import { CgProfile } from "react-
icons/cg"; import { Link } from 'react-
router-dom';

const NewsFeed = () => {
  const [articles, setArticles] =useState([])
  const [fetched, setFetched] =useState(false);
  const [likedArticles, setLikedArticles] = useState();
  const [bookedArticles, setBookedArticles] =
  useState(); const [changed, setChanged] =
  useState(false);
  // const [found1, setFound1] = useState(0);
  // const [found2, setFound2] = useState(0);
  // useEffect(() => {
  //   setFound1(found1+1);
  // }, [likedArticles])
  // useEffect(() => {
  //   setFound2(found2+1);
  // },
  [bookedArticles])
  useEffect(() => {
    const fn = () => {
      axios.get("http://169.51.205.76:32522/get-top-headlines?userName=" +
sessionStorage.getItem('@user')).then((data, error) => {
        axios.get("http://169.51.205.76:32522/profile?userName=" +
sessionStorage.getItem('@user')).then((profileData, err) => {
          console.log(profileData);
          let likedList = [], bookedList = [];
          for(const article of
```

```
          obj[name] = true;
          likedList = {
            ...likedList,
            ...obj
          };
        }
        for(const article of
          profileData?.data.bookmarks){ var name =
          article.NEWS_ARTICLE_LINK;
          var obj = {};
          obj[name] = true;
          bookedList = {
            ...bookedList,
            ...obj
          };
        }
        setLikedArticles(likedList);
        setBookedArticles(bookedList);
        setArticles(data.data.articles)
        ; setFetched(true);
      })
    })
  }
  fn();
}, [])

return (
  <div className='w-full flex'>
    <div className='m-auto justify-center p-20 rounded-lg'>
      <Link to="/profile"><CgProfile className='text-white text-5xl float-
      right
-mr-3/4 -mt-16' /></Link>
      {fetched ?
        <>
          {articles.map((item, index) =>
            { if(index==0)
            console.log('lister',likedArticles,bookedArticles);
            return (
              <News new={{
                ...item,
              liked: likedArticles[`${item.url}`],
                bookmark: bookedArticles[`${item.url}`]
              }} key={index} />
            )
          }
```

Profile.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import 'react-responsive-
carousel/lib/styles/carousel.min.css' import { Carousel }
from 'react-responsive-carousel';
import News from "../component/News";
import Multiselect from"multiselect-react-dropdown";

const Profile = () => {
  const [fetched, setFetched] =
  useState(false); const [data, setData] =
  useState();
  const [likedArticles, setLikedArticles] = useState([]);
  const [bookedArticles, setBookedArticles] = useState([]);
  const [options, setOptions] = useState([{ name: 'business', id: 1 }, { name:
'entertainment', id: 2 }, { name: 'general', id: 3 }, { name: 'health', id: 4 },
{ name: 'science', id: 5 }, { name: 'sports', id: 6 }, { name: 'business', id: 7
}, { name: 'technology', id: 8 }, { name: 'Virat Kohli', id: 9 }, { name:
'Narendra Modi', id: 10 }, { name: 'Rain', id: 13 }, { name: 'MK Stalin', id: 12
}, { name: 'Cinema', id: 13 }, { name: 'Floods', id: 14 }, { name: 'politics',
id: 15 }, { name: 'Donald Trump', id: 16 }, { name: 'Putin', id: 17 }, { name:
'Ukraine-Russia', id: 18 }, { name: 'Biden', id: 19 }, { name: 'ADMK', id: 20 },
{ name: 'Rahul Gandhi', id: 21 }, { name: 'China', id: 22 }, { name: 'corona',
id: 23 }, { name: 'elon musk', id: 24 }, { name: 'worldcup', id: 25 }, { name:
'BJP', id: 26 }, { name: 'Taiwan China crisis', id: 27 }, { name: 'job
```

```
opputunities', id: 28 }, { name: 'tourism', id: 29 }, { name: 'metroplian', id:
30 }]);
  const [selectedVal, setSelectedVal] =
  useState([]); useEffect(() => {
    const fn = async () =>{
      const profileData = await
axios.get("http://169.51.205.76:32522/profile?userName=" +
sessionStorage.getItem('@user'));
      console.log('data', profileData.data.topics);
      setData(profileData.data);
      const selval = [];
      let ids = 0;
      for (const topic of
        profileData.data.topics) { selval.push({
          name:
          topic, id:
          ids
        })
        ids++;
      }
      setSelectedVal(selval);
      const likedLinks = profileData.data?.likes.reduce((prev, cur)
        => { return prev.concat(cur.NEWS_ARTICLE_LINK);
      }, [])
      const book
markedLinks = profileData.data?.bookmarks.reduce((prev, cur)
        => { return prev.concat(cur.NEWS_ARTICLE_LINK);
      }, [])
      setLikedArticles(likedLinks);
      setBookedArticles(bookmarkedLinks)
      ; setFetched(true);
    }
    fn();
  }, [])
  const onSelect = (selectedList, selectedItem) => {
    selectedVal.push(selectedItem)
  }

  const onRemove = (selectedList, removedItem) => {
    let newVal = [];
```

```
for(const topic of selectedVal){
if(topic.id==removedItem.id){
        console.log(topic,removedItem
        ); continue;
      }
      newVal.push(topic);
    }
    setSelectedVal(newVal);
  }
  return (
    <>
      {fetched ?
        <div>
          <div className="pt-20">
            <div className="px-20">
              <div className="flex flex-col min-w-0 break-words bg-white w-
full mb-6 shadow-xl rounded-lg">
                <div className="px-6">
                  <div className="flex flex-wrap justify-center">
                    <div className="w-full lg:w-3/12 px-4 lg:order-2 flex
justify-
center">
                      <div>
                        <img width={180} height={180}alt="..."
src="https://64.media.tumblr.com/bed5455fdd7789656247fb01ed60ad31/93c34c0b
6d121ae f-6a/s400x600/15a48a245b0ef5bb4f5215d177116cf9c4150531.png"
className="shadow-xl rounded-full h-auto align-middle border-none absolute -
m-16 -ml-20 lg:-ml-16
max-w-150-px" />
                      </div>
                    </div>
                    <div className="w-full lg:w-4/12 px-4
lg:order-3 lg:text-right lg:self-center">
                      <div className="py-6 px-3 mt-32 sm:mt-0">
                        <button onClick={() => {
                          sessionStorage.removeItem('@user');
                          window.location.replace('/');
                        }}className="bg-pink-500 active:bg-pink-600
uppercase text-white font-bold hover:shadow-md shadow text-xs px-4 py-2
rounded outline-none focus:outline-none sm:mr-2 mb-1 ease-linear
transition-all duration-150" type="button">
```

```
                        Logout
                      </button>
                    </div>
                  </div>
                  <div className="w-full lg:w-4/12 px-4 lg:order-1">
                    <div className="flex justify-center py-4 lg:pt-4 pt-8">
                      <div className="mr-4 p-3 text-center">
                        <span className="text-xl font-bold block
uppercase tracking-wide text-blueGray-
600">{data?.likes?.length}</span><span
className="text-sm text-blueGray-400">Likes</span>
                      </div>
                      <div className="mr-4 p-3 text-center">
                        <span className="text-xl font-bold block
uppercase tracking-wide text-blueGray-
600">{data?.bookmarks?.length}</span><span
        className="text-sm  text-blueGray-
              400">Bookmarks</span>
                      </div>
                    </div>
                  </div>
                </div>
                <div className="text-center mt-12">
                  <h3 className="text-4xl font-semibold leading-normal
mb-2 text-blueGray-700 mb-2">
                    {data?.user.NAME}
                  </h3>
                  <div className="text-sm leading-normal mt-0
mb-2 text-blueGray-400 font-bold uppercase">
                    <i className="fas fa-map-marker-alt mr-2
text-lg text-blueGray-400"></i>
                    Date of Birth :{data?.user.DOB}
                  </div>
                  <div className="mb-2 text-blueGray-600">
                    <i className="fas fa-briefcase mr-2
text-lg text-blueGray-400"></i> Email :
{data?.user.USERNAME}
                  </div>
                  <div className="mb-2 text-blueGray-600">
                    <i className="fas fa-university mr-2 text-lg
text-blueGray-400"></i>Password :{'*'.repeat(data?.user.PASSWORD?.length)}
                  </div>
```

```
                    <i className="fas fa-university mr-2
text-lg text-blueGray-400"></i>Topics: {data?.user.TOPICS}
                  </div>
                </div>
                {data?.likes.length > 0 ?
                  <div className="mt-10 py-10 border-t border-blueGray-200
text-
center">
                    <div  className="flex flex-wrap justify-center">
                      <h1 className="text-4xl font-semibold leading-normal
                      mb-2
      text-blueGray-700  mb-2">Articles
                                Liked</h1>
                    <Carousel  autoPlay>
                        {data?.likes.map((item, index) =>
                          <News new={{
                            title: item.NEWS_TITLE,
                            url: item.NEWS_ARTICLE_LINK,
                            description: item.NEWS_DESCRIPTION,
                            urlToImage: item.NEWS_IMAGE_LINK,
                            publishedAt:
                            item.NEWS_DATE, liked:
                            true,
                            bookmark:
bookedArticles?.includes(item.NEWS_ARTICLE_LINK)
                          }} key={index} />
                        )}
                      </Carousel>
                    </div>
                  </div>
                  : null}
                {data?.bookmarks?.length ?
                  <div className="w-full mt-10 py-10
border-t border-blueGray-200 text-center">
                    <div  className="flex justify-center">
                      <h1 className="text-center text-4xl font-
semibold leading-normal mb-2 text-blueGray-700 mb-2">Articles
BookMarked</h1>
<Carousel autoPlay>{data?.bookmarks.map((item, index) =>
                          <News new={{
                            title: item.NEWS_TITLE,
                            url: item.NEWS_ARTICLE_LINK,
                            description: item.NEWS_DESCRIPTION,
```

```
                            urlToImage: item.NEWS_IMAGE_LINK,
                            publishedAt: item.NEWS_DATE,
                            liked:
likedArticles?.includes(item.NEWS_ARTICLE_LINK),
                            bookmark: true
                          }} key={index} />
                        )}
                      </Carousel>
                    </div>
                  </div>
                  : null}
                <div className="text-center mb-40">
                  <h1 className="text-4xl font-semibold leading-normal
mb-2 text-blueGray-700 mb-2">Update Tags</h1>
                  <Multiselect
                    options={options}
                    selectedValues={selectedVal}
                    onSelect={onSelect}
                    onRemove={onRemove}
                    displayValue="name"
                  />
                  <div className="flex justify-centermt-40">
                    <button onClick={async() =>
                      { let topicSelected = [];
                      for(const topics of selectedVal){
                        topicSelected.push(topics.name);
                      }
                      const payload = {
                        user: sessionStorage.getItem('@user'),
                        topics: topicSelected
                      }
                      await
axios.post("http://169.51.205.76:32522/update-profile",payload);
                    }} className="bg-pink-500 active:bg-pink-600
uppercase text-white font-bold hover:shadow-md shadow text-xs px-4 py-2
rounded
outline-none focus:outline-none sm:mr-2 mb-1 ease-linear
transition-all duration-150" type="button">
                      Update
                    </button>
```

Signup.js

```
import React, { useState } from "react";
import { Link } from "react-router-dom";
import Multiselect from 'multiselect-react-
dropdown'; const Signup = () => {
  const [name, setName] = useState();
  const [email, setEmail] =
  useState(); const [phone,
  setPhone] = useState();
  const [password, setPassword] = useState();
  const [confirm_password, setConfirmPassword] =
  useState(); const [dob, setDob] = useState();
  const [options, setOptions] = useState([{ name: 'business', id: 1 }, { name:
```

```
                  </div>
                </div>
              </div>
            </div>
          </div>
          <div className="relative bg-blueGray-200 pt-8 pb-6 mt-8">
            <div className="container mx-auto px-4">
              <div className="flex flex-wrap items-center md:justify-
between justify-center">
                <div className="w-full md:w-6/12 px-4 mx-auto text-center">
                  <div className="text-sm text-white font-semibold py-
                    1"> Made by Saitama Squad
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
        : null}
    </>
  )
}

export default Profile;
```

```
{ name: 'science', id: 5 }, { name: 'sports', id: 6 }, { name: 'business', id: 7
}, { name: 'technology', id: 8 }, { name: 'Virat Kohli', id: 9 }, { name:
'Narendra Modi', id: 10 }, { name: 'Rain', id: 13 }, { name: 'MK Stalin', id: 12
}, { name: 'Cinema', id: 13 }, { name: 'Floods', id: 14 }, { name: 'politics',
id: 15 }, { name: 'Donald Trump', id: 16 }, { name: 'Putin', id: 17 }, { name:
'Ukraine-Russia', id: 18 }, { name: 'Biden', id: 19 }, { name: 'ADMK', id: 20 },
{ name: 'Rahul Gandhi', id: 21 }, { name: 'China', id: 22 }, { name: 'corona',
id: 23 }, { name: 'elon musk', id: 24 }, { name: 'worldcup', id: 25 }, { name:
'BJP', id: 26 }, { name: 'Taiwan China crisis', id: 27 }, { name: 'job
opputunities', id: 28 }, { name: 'tourism', id: 29 }, { name: 'metroplian', id:
30 }]);
  const [selectedVal, setSelectedVal] = useState([]);

  const signupHandler = (e) => {
    e.preventDefault();
    if (password ==
      confirm_password) { const
      payload = {
        email: email,
        password: password,
        options:
        selectedVal, name:
        name,
        phone:
        phone, dob:
        dob
      }
      fetch('http://169.51.205.76:32522/sign-
        up', { method: 'POST',
        mode: 'cors',
        redirect: 'manual',
        body: JSON.stringify(payload)
      }).then((res) => {
        if (res.status === 200) window.location.replace("/");
      })
    }
  }
  const onSelect = (selectedList, selectedItem) => {
    selectedVal.push(selectedItem)
  }
  const onRemove = (selectedList, removedItem) => {
```

```
    const index = selectedVal.indexOf(removedItem);
    if (index > -1) { // only splice array when item is found
      selectedVal.splice(index, 1); // 2nd parameter means remove one item only
    }
  }

  return (
    <>
      <div className="min-h-screen flex flex-col">
        <div className="m-20 bg-white container max-w-lg mx-auto flex-1
flex flex-col items-center justify-center px-2">
          <div className="px-6 py-8 rounded shadow-md text-black w-full">
            <h1 className="mb-8 text-3xl text-center">Sign up</h1>
            <input
              type="text"
              className="block border border-grey-light w-full p-3 rounded
              mb-4" name="fullname"
              placeholder="Full Name"
              onChange={(e) => {
                setName(e.target.value);
              }}
            />

            <input
              type="text"
              className="block border border-grey-light w-full p-3 rounded
              mb-4" name="email"
              placeholder="Email"
              onChange={(e) => {
                setEmail(e.target.value);
              }}
            />

            <input
              type="text"
              className="block border border-grey-light w-full p-3 rounded
              mb-4" name="phoneno"
              placeholder="Phone
              Number" onChange={(e) =>
              {
```

```
            setPhone(e.target.value);
          }}
        />

        <input
          type="password"
          className="block border border-grey-light w-full p-3 rounded
          mb-4" name="password"
          placeholder="Password"
          onChange={(e) => {
            setPassword(e.target.value);
          }}
        />
        <input
          type="password"
          className="block border border-grey-light w-full p-3 rounded
          mb-4" name="confirm_password"
          placeholder="Confirm
          Password" onChange={(e) =>
          {
            setConfirmPassword(e.target.value);
          }}
        />

        <input
          type="date"
          className="block border border-grey-light w-full p-3 rounded
          mb-4" name="dob"
          placeholder="Date of
          Birth" onChange={(e) => {
            setDob(e.target.value);
          }}
        />

        <Multiselect
          options={options}
          selectedValues={selectedVal}
          onSelect={onSelect}
          onRemove={onRemove}
          displayValue="name"
```

```
          />
          <button
            type="submit"
            onClick={signupHandler}
            className="mt-10 w-full text-center py-3 rounded bg-green-
500 text-white hover:bg-green-dark focus:outline-none my-1"
          >Create Account</button>

          <div className="text-center text-sm text-grey-dark mt-
            4"> By signing up, you agree to the  
            <a className="no-underline border-b border-grey-
dark text-grey-dark" href="#">
              Terms of Service
            </a> and  
            <a className="no-underline border-b border-grey-
dark text-grey-dark" href="#">
              Privacy Policy
            </a>  
          </div>
        </div>

        <div className="text-grey-dark mt-
          6"> Already have an account?
          <Link to="/" className="no-underline border-b border-blue text-
blue" href="../login/">
            Log in
          </Link>.
        </div>
      </div>
    </div>
  </>
 )
}

export default Signup;
```

Component folder

News.js

```
import React, { useState } from 'react';
import Card from '@mui/material/Card';
import CardHeader from '@mui/material/CardHeader'; import CardMedia
from '@mui/material/CardMedia';
import CardContent from
'@mui/material/CardContent'; import CardActions
from '@mui/material/CardActions'; import
IconButton from '@mui/material/IconButton';
import Typography from '@mui/material/Typography';
import FavoriteIcon from '@mui/icons-
material/Favorite'; import ShareIcon from
'@mui/icons-material/Share';
import BookmarkIcon from '@mui/icons-
material/Bookmark'; import dateFormat from
"dateformat";
import axios from 'axios';
const News = (props) => {
  const [liked, setLiked] = useState(props.new.liked)
  const [bookmark, setBookMark] =
  useState(props.new.bookmark); const dateOfNew = new
  Date(props.new.publishedAt);
  return (
    <div className="mt-20">
      <Card sx={{ minWidth: 200, maxWidth: 1000 }}>
        <a href={props.new.url}>
          <CardHeader
            title={props.new.title}
            subheader={dateFormat(dateOfNew)}
          />
        </a>
        {props.new.urlToImage ?
          <CardMedia
            component="img"
            height="50"
            style={{margin:"auto",height:"500px",width:"500px"}}
            src={props.new.urlToImage}
            alt="Paella dish"
          /> :
          <CardMedia
            component="img"
```

```
src=https://thumbs.dreamstime.com/b/news-newspapers-folded-stacked-word-
wooden-b lock-puzzle-dice-concept-newspaper-media-press-release-
42301371.jpg alt="Paella dish" />
        }
        <CardContent>
          <Typography variant="body2" color="text.secondary">
            {props.new.description}
          </Typography>
        </CardContent>
        <CardActions disableSpacing>
          {liked ?
            <IconButton style={{ color: 'red' }} onClick={async () => {
              setLiked(false);
              const payload = {
                email:
                sessionStorage.getItem('@user'),
                url: props.new.url ? props.new.url : "",
                action: 'S',
                type: 'R'
              }
              console.log(payload);
              await axios.post('http://169.51.205.76:32522/action',
              payload);
            }} aria-label="add to favorites">
              <FavoriteIcon />
            </IconButton>
            :
            <IconButton style={{ color: 'gray' }} onClick={async () => {
              setLiked(true);
              const payload = {
                email: sessionStorage.getItem('@user'),
                title: props.new.title ? props.new.title :
                "", url: props.new.url ? props.new.url : "",
                urlToImage: props.new.urlToImage ? props.new.urlToImage :
"https://thumbs.dreamstime.com/b/news-newspapers-folded-stacked-word-
wooden-block
-puzzle-dice-concept-newspaper-media-press-release-42301371.jpg",
                publishedAt: props.new.publishedAt ? props.new.publishedAt :
                "",        description:       props.new.description       ?
                props.new.description : "", action: 'S',
```

```
              type: 'A'
            }
            console.log(payload);
            //169.51.205.76:32522
            await axios.post('http://169.51.205.76:32522/action',
            payload);
          }} aria-label="add to favorites">
            <FavoriteIcon />
          </IconButton>
        }
        <IconButton aria-label="settings">
          {bookmark ? <BookmarkIcon onClick={async () => {
            const payload = {
              email:
              sessionStorage.getItem('@user'),
              url: props.new.url ? props.new.url : "",
              action: 'B',
              type: 'R'
            }
            console.log(payload);
            await axios.post('http://169.51.205.76:32522/action',
            payload); setBookMark(false)
          }} style={{ color: "green" }} /> : <BookmarkIcon onClick={async () =>
{
            const payload = {
              email: sessionStorage.getItem('@user'),
              title: props.new.title ? props.new.title :
              "", url: props.new.url ? props.new.url : "",
              urlToImage: props.new.urlToImage ? props.new.urlToImage :
"https://thumbs.dreamstime.com/b/news-newspapers-folded-stacked-word-
wooden-block
-puzzle-dice-concept-newspaper-media-press-release-42301371.jpg",
              publishedAt: props.new.publishedAt ? props.new.publishedAt :
              "",        description:       props.new.description       ?
              props.new.description : "", action: 'B',
              type: 'A'
            }
            console.log(payload);
            //169.51.205.76:32522
            await axios.post('http://169.51.205.76:32522/action',
            payload); setBookMark(true)
          }} />}
```

```
          </IconButton>
          <a href={`https://api.WhatsApp.com/send?text=` + props.new.url}>
            <IconButton aria-label="share">
              <ShareIcon />
            </IconButton>
          </a>
        </CardActions>
      </Card>
    </div>
  );
}

export default News;
```

```
{
  "name": "client",
  "version":
  "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react":
    "latest",
    "@emotion/styled":
    "latest",
    "@mui/icons-material":
    "latest", "@mui/material":
    "latest",
    "@testing-library/jest-dom":
    "^5.16.5", "@testing-
    library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "@types/react": "latest",
    "@types/react-dom": "latest",
    "antd": "^4.24.1",
    "axios": "^1.1.3",
    "dateformat": "^5.0.3",
    "i": "^0.3.7",
    "multiselect-react-dropdown": "^2.0.25",
    "npm": "^8.19.3"
```

```
  "react-redux": "^7.2.9",
  "react-responsive-carousel":
  "^3.2.23", "react-router-dom":
  "^5.3.0",
  "react-scripts": "5.0.1",
  "redux": "^4.2.0",
  "redux-thunk": "^2.4.1",
  "web-vitals": "^2.1.4"
},
"scripts": {
  "start":       "react-scripts
  start",    "build":   "react-
  scripts    build",    "test":
  "react-scripts test",
  "eject": "react-scripts eject"
},
"eslintConfig":
  { "extends": [
    "react-app",
    "react-app/jest"
  ]
},
"browserslist":
  {
  "production":
  [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox
    version", "last 1
    safari version"
  ]
},
"devDependencies": {
  "autoprefixer": "^10.4.13",
  "postcss": "^8.4.18",
  "tailwindcss": "^3.2.2"
```

Server folder

App.py

```
from flask import Flask, request, Response,
send_from_directory import requests
import os
import
random
import json
from flask_cors import
CORS import db_crud
app = Flask(_name_) #, static_folder='build', template_folder='build',
static_url_path='')
CORS(app)

NEWS_API_TOPHEADLINES_ENDPOINT = 'https://newsapi.org/v2/top-headlines'
NEWS_API_KEY = 'aeb9762b06d74d1a8ece0f3b896feb4c'
user_preferences =
['business','entertainment','general','health','science','sports','technolo
gy']

@app.route('/get-top-headlines',
methods=['GET']) def
get_top_headlines_for_user():
  query = request.args.to_dict()
  if query.get('userName', -1) == -1:
    return Response('userName must be provided in query string',
  status=400) country = query.get('country', 'in')
  q = query.get('q', '')

  user_topics = db_crud.get_topics(query.get('userName'))
  user_prefs = 'category=' +
  '&category='.join(user_topics) url =
NEWS_API_TOPHEADLINES_ENDPOINT+f'?country={country}&{user_prefs}&apiKey={NE
WS_API
_KEY}'
  if q != '':
    url += f'&q={q}'
  print(url)

  api_response = requests.get(url=url)
  return json.loads(api_response.content.decode()), 200
```

```
@app.route('/authenticate',
methods=['POST']) def authenticate_user():
  form = json.loads(request.data.decode())
  if db_crud.authenticate(form['email'],
    form['password']): return {'status':'success'}, 200
  return {'status':'failure'}, 400
@app.route('/sign-up',
methods=['POST']) def
sign_up_user():
  form =
  json.loads(request.data.decode())
  print(form)
  if db_crud.create_account(form['name'], form['email'], form['password'],
form['phone'], form['dob'], [x['name'] for x in form['options']]):
    return {'status':'success'}, 200
  return {'status':'failure'}, 400
@app.route('/profile',
methods=['GET']) def user_profile():
  query =
  request.args.to_dict() ret
  = {
    "likes" : db_crud.get_likes(query['userName']),
    "bookmarks" :
    db_crud.get_bookmarks(query['userName']), "topics":
    db_crud.get_topics(query['userName']),
    "user" : db_crud.get_user(query['userName'])
  }return ret, 200
@app.route('/action',
methods=["POST"]) def action():
  form =
  request.data.decode()
  form =json.loads(form)
  print(form)
  if(form['type']=='A'):
    print("ADD")
    if db_crud.add_action(form['email'], form['title'], form['url'],
form['urlToImage'], form['publishedAt'], form['description'],
form['action']): return {'status':'success'}, 200
    else:
      return {'status':'failure'}, 400
```

```
apiVersion:
apps/v1 kind:
Deployment
metadata:
  name: server-
deployment spec:
  replicas:
  1
  selector:
    matchLabels:
      app:
  appserver
  template:
```

```
  if(form['type']=='R'):
    print("REMOVE")
    if db_crud.remove_action(form['email'],
      form['url']): return {'status':'success'}, 200
    else:
      return {'status':'failure'}, 400

@app.route('/update-profile',
methods=['POST']) def updateprofile():
  form = json.loads(request.data.decode())
  db_crud.update_topics(form['user'],
  form['topics']) return {'status':'success'},
  200


# @app.route('/',
methods=['GET']) # def
react_app():
#   return send_from_directory(app.static_folder, 'index.html')

#
@app.errorhandler(404)
# def not_found(e):
#   return app.send_static_file('index.html')
```

```
    spec:
      containers:
      - name: jobportal
        image: icr.io/ibm-project/appserver:latest
---
apiVersion:
v1 kind:
Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app:
  appserver
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort:
```

```
import
ibm_db
import os
dsn_hostname =
"0c77d6f2-5da9-48a9-81f8-
86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cl oud"
dsn_uid = "bsj69971"        # e.g. "abc12345"
dsn_pwd = "J0ww1ELveoQeDVWK"    # e.g. "7dBZ3wWt9XN6$o0J"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"           # e.g. "BLUDB"
dsn_port = "31198"               # e.g. "32733"
dsn_protocol = "TCPIP"           # i.e. "TCPIP"
dsn_security = "SSL"             #i.e. "SSL"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
```

```
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname,
dsn_port, dsn_protocol, dsn_uid, dsn_pwd,dsn_security)


conn = ibm_db.pconnect(dsn, "", "")

def authenticate(username, password):
    stmt = ibm_db.exec_immediate(conn, "SELECT * FROM
    USERS") row = True
    while row!= False:
        row =
        ibm_db.fetch_assoc(stmt) if
        not row: return False
        if(row['USERNAME'] == username and row['PASSWORD'] ==
            password): return True
    return False

def get_user(username):
    stmt = ibm_db.exec_immediate(conn, "SELECT * FROM
    USERS") row = True
    while row!= False:
        row =
        ibm_db.fetch_assoc(stmt) if
        not row: return False
        if(row['USERNAME'] ==
            username): return
            dict(row)
    return dict()

def create_account(name, username, password, phone, dob,
    topics): try:
        sql = "INSERT INTO USERS VALUES(?,?,?,?,?,?)"
        stmt = ibm_db.prepare(conn,
        sql) ibm_db.bind_param(stmt,
        1, name)
        ibm_db.bind_param(stmt, 2, username)ibm_db.bind_param(stmt, 3,
        password)ibm_db.bind_param(stmt, 4, phone)ibm_db.bind_param(stmt,
        5, dob)ibm_db.bind_param(stmt, 6, ','.join(topics))
```

```
        ibm_db.execute(stmt)
        return True
    except:
        return False

def get_likes(user):
    stmt = ibm_db.exec_immediate(conn, "SELECT * FROM NEWS N where N.user='{0}'
and news_type='S'".format(user))
    likes =[]
    row =True
    while row!= False:
        row =
        ibm_db.fetch_assoc(stmt) if
        not row: break
        likes.append(dict(row
    )) return likes

def get_bookmarks(user):
    stmt = ibm_db.exec_immediate(conn, "SELECT * FROM NEWS N where N.user='{0}'
and news_type='B'".format(user))
    bookmarks = []
    row = True
    while row!= False:
        row =
        ibm_db.fetch_assoc(stmt) if
        not row: break
        bookmarks.append(dict(row))
    return bookmarks

def get_topics(user):
    stmt = ibm_db.exec_immediate(conn, "SELECT * FROM USERS
where username='{0}'".format(user))
    row = True
    while row!= False:
        row =
        ibm_db.fetch_assoc(stmt) if
        not row: break
        return
    row['TOPICS'].split(',') return
    []
```

```
FROM
debian:stable
COPY . ./server
```

Dockerfile

```
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, ','.join(topics))
    ibm_db.bind_param(stmt, 2,
    user) ibm_db.execute(stmt)

def add_action(user,title,url,image_url,date,desc,action):
    try:
        sql = "INSERT INTO NEWS VALUES(?,?,?,?,?,?,?)"
        stmt = ibm_db.prepare(conn,
        sql) ibm_db.bind_param(stmt,
        1, user)
        ibm_db.bind_param(stmt, 2, title)
        ibm_db.bind_param(stmt, 3, url)
        ibm_db.bind_param(stmt, 4, image_url)
        ibm_db.bind_param(stmt, 5, date)
        ibm_db.bind_param(stmt, 6, desc)
        ibm_db.bind_param(stmt, 7,
        action) ibm_db.execute(stmt)
        return True
    except Exception as e:
        print(e)
        return False

def remove_action(user,
    url): try:
        sql = "DELETE FROM NEWS N WHERE N.user='{0}' AND
news_article_link='{1}'".format(user, url)
        ibm_db.exec_immediate(conn, sql)
        return True
    except Exception as e:
        print(e)
        return False
```

```
RUN apt update
RUN apt install -y build-essential libxml2
RUN apt install -y python3
RUN apt install -y python3-pip
RUN rm -rf
/var/lib/apt/lists/*
RUN pip3 install -r requirements.txt
EXPOSE 5000
```

```
certifi==2022.9.24
charset-
normalizer==2.1.1
click==8.1.3
colorama==0.4.6
Flask==2.2.2
Flask-
Cors==3.0.10
ibm-db==3.1.3
idna==3.4
importlib-metadata==5.0.0
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
python-dotenv==0.21.0
requests==2.28.1
six==1.16.0
urllib3==1.26.12
Werkzeug==2.2.2
zipp==3.10.0
```

Github Link

https://github.com/IBM-EPBL/IBM-Project-7781-1664352003

Demo Link -

https://youtu.be/E4bXYD79s68