**NEWS TRACKER APPLICATION**

**Team ID: PNT2022TMID12670**

Bachelor of Engineering

Computer Science and Engineering

VelTech MultiTech Dr.Rangarajan Dr.Sakunthala

Engineering College -Avadi,Chennai-600 062.

**Faculty Evaluator** : Dr.Nehru

**Faculty Mentor** : Mr.Prabhu Sankar

**TEAM MEMBERS**

Arunkumar.Y 113119UG03008

Abishek.P.Y 113119UG03002

Jachin.I 113119UG03036

MohamedAmiz.R 113119UG03057

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page. No** |
| **1** | **INTRODUCTION** | **4** |
|  | 1.1 Project Overview | 4 |
|  | 1.2 Purpose | 4 |
| **2** | **LITERATURE SURVEY** | **4** |
|  | 2.1 Existing problem | 4 |
|  | 2.2 References | 5 |
|  | 2.3 Problem Statement Definition | 5 |
| **3** | **IDEATION & PROPOSED SOLUTION** | **7** |
|  | 3.1 Empathy Map Canvas | 7 |
|  | 3.2 Ideation & Brainstorming | 8 |
|  | 3.3 Proposed Solution | 11 |
|  | 3.4 Problem Solution fit | 12 |
| **4** | **REQUIREMENT ANALYSIS** | **14** |
|  | 4.1 Functional requirement | 14 |
|  | 4.2 Non-Functional requirements | 14 |
| **5** | **PROJECT DESIGN** | **15** |
|  | 5.1 Data Flow Diagrams | 15 |
|  | 5.2 Solution & Technical Architecture | 16 |
|  | 5.3 User Stories | 17 |
| **6** | **PROJECT PLANNING & SCHEDULING** | **18** |
|  | 6.1 Sprint Planning & Estimation | 18 |
|  | 6.2 Sprint Delivery Schedule | 19 |
|  | 6.3 Reports from JIRA | 20 |
| **7** | **CODING & SOLUTIONING** | **22** |
|  | 7.1 Feature 1 | 22 |
|  | 7.2 Feature 2 | 26 |
|  | 7.3 Feature 3 | 28 |
|  | 7.4 Feature 4 | 30 |
| **8** | **TESTING** | **34** |
|  | 8.1 Test Cases | 34 |
|  | 8.2 User Acceptance Testing | 35 |
| **9** | **RESULTS** | **36** |

**Table of the Content**

|  |  |  |
| --- | --- | --- |
|  | 9.1 Performance Metrics | 36 |
| **10** | **ADVANTAGES & DISADVANTAGES** | **36** |
| **11** | **CONCLUSION** | **37** |
| **12** | **FUTURE SCOPE** | **37** |
| **13** | **APPENDIX** | **37** |
|  | Source Code | 37 |
|  | GitHub & Project Demo Link | 60 |

**1.INTRODUCTION**

**1.1.Project Overview**

News articles are collected from various news channels and news sources from across the internet. These news articles are then categorized into various sections. All the news, belonging to a particular category will be displayed under a specific section.

The news articles are displayed on the basis of the interests and preference of the user. News feed is used to analyzed the interest of the user. Based on the type of news the users views, their interests is analyzed.

User also will have the option to save snippets from news articles, mark some news articles as bookmark and later to see their views about the news.

Prefer the language for the user based on their location and user wants to change the language manually .

**1.2.purpose**

The goal of this project is to collect all the news articles from across the internet and display it in an orderly manner, based on the interests and preferences of the user, at a single destination.

**2.LITERATURE SURVEY**

**2.1.Existing `problem**

News are collected from various news sources and are displayed without being properly categorized into various sections.

News articles are recommended randomly.

Every news articles, whether important or not, is given the same preference and are displayed by being sorted on the basis of the time it was published.

Every news articles will be displayed whether or not the content is correct.

The language of the UI is English and users have no option to change the language.

**2.2.References**

* Allan, J., Papka, R., Lavrenko, V.: On-line New Event Detection and Tracking. In: Proceedings of 21st ACM SIGIR, Melbourne (1998)
* Bacan, H., Pandzic, I.S., Gulija, D.: Automated News Item Categorization. In: JSAI (2005)
* Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
* Yamron, J.P., Carp, I., Gillick, L., Lowe, S., Van Mulbregt, P.: Topic Tracking in a News Stream. In: Proceedings of DARPA Broadcast News Workshop (1999)
* Mori, M., Miura, T., Shioya, I.: Topic Detection and Tracking for News Web Pages. In: IEEE/WIC/ACM International Conference on Web Intelligence, pp. 338–342 (2006)

**2.3 Problem Statement Definition**

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service

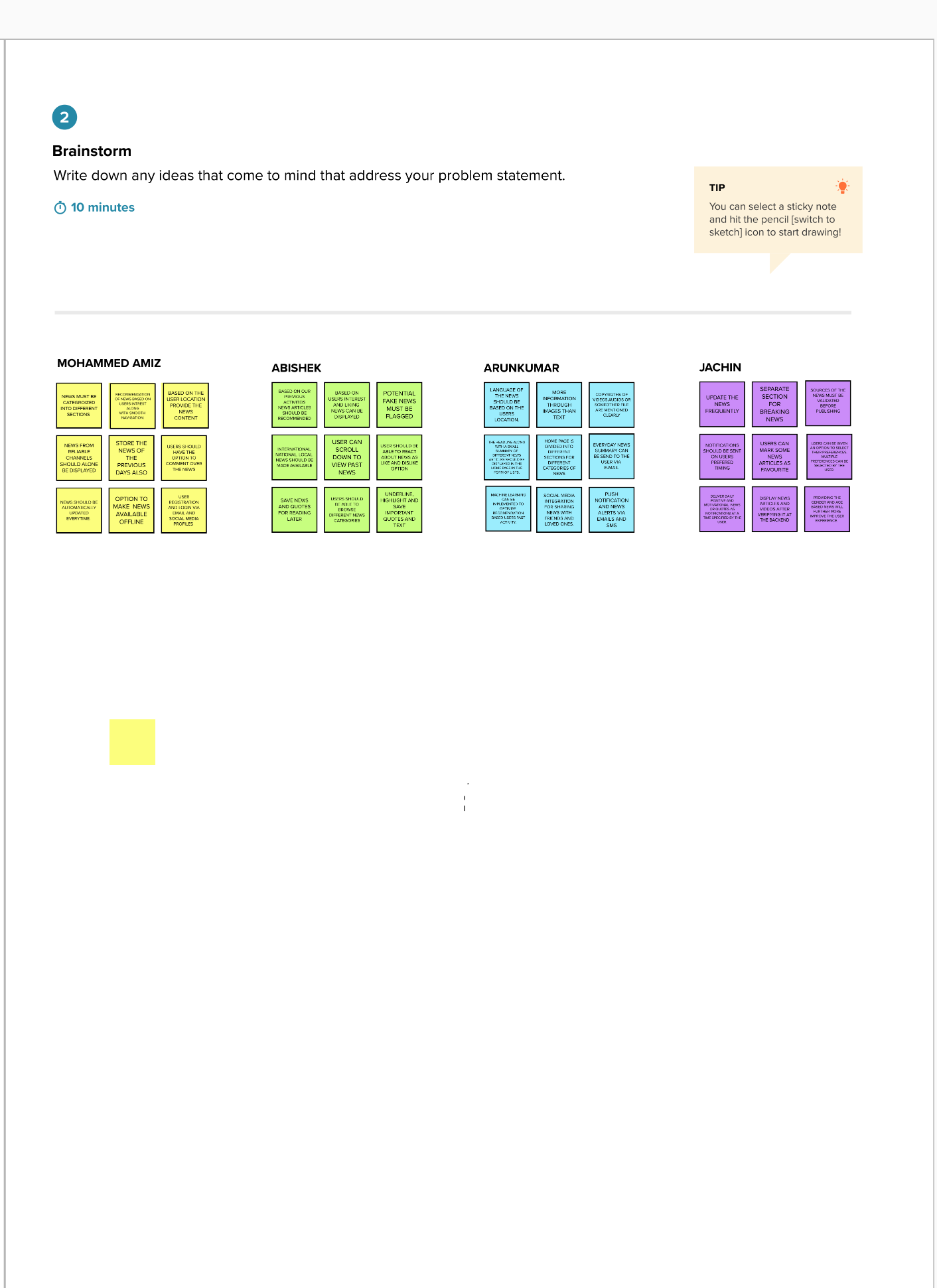
|  |  |
| --- | --- |
| **Iam** | Regular news reader in online news tracker applications |
| **I’m trying to** | Read online articles without any annoying contents |
| **But** | * There are multiple news-sharing apps used by a single user and are often spammed with notifications. * There is also a lot of fake news which gets shared. * A news-sharing app wants to help users find relevant and important news easily every day and understand explicitly that the news is not fake but from proper sources. |
| **Because** | * News apps are trying to be like social media apps * News apps want to increase the time that user spends on their app so that they can show ads and generate revenue * To increase the user screen time, news apps make users encounter eye-catching news rather than credible ones * Apps generate income through subscriptions. |
| **Which makes**  **me feel** | * Fake news hurts individuals and society as it persuades consumers to accept false beliefs that are shared to forward specific agendas. * Identifying relevant news from excessive amounts of information on social media requires substantial time, energy, and mental efforts * Constant news updates and pop-ups of breaking news in social   media may increase the feeling of news overload. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***I am*** | ***I’m trying to*** | ***But*** | ***Because*** | ***Which makes me feel*** |
| Online news articles reader | Read online articles without any annoying contents | Spamming of messages usually leads to clearing of the content without viewing thus probably leading the user to lose access to important information. | Businesses must publish irrelevant news because younger generations prefer news with more fun instead of reliable news. | Irritated |
| Read online articles without any ads | Ads in the apps might irritate the user while reading the news | Apps generate income through subscriptions and ads | Frustrated |
| Read precise contents | News apps want to increase the time that user spends on their app so that they can show ads and generate revenue | Users don’t want to spend time reading the entire content. They need short and crisp news | Annoying |
| Avoid irrelevant news | Irrelevant news makes the user stop viewing the news thus losing access to credible news | Businesses must publish irrelevant news because younger generations prefer news with more fun instead of reliable news. | Exhausted |

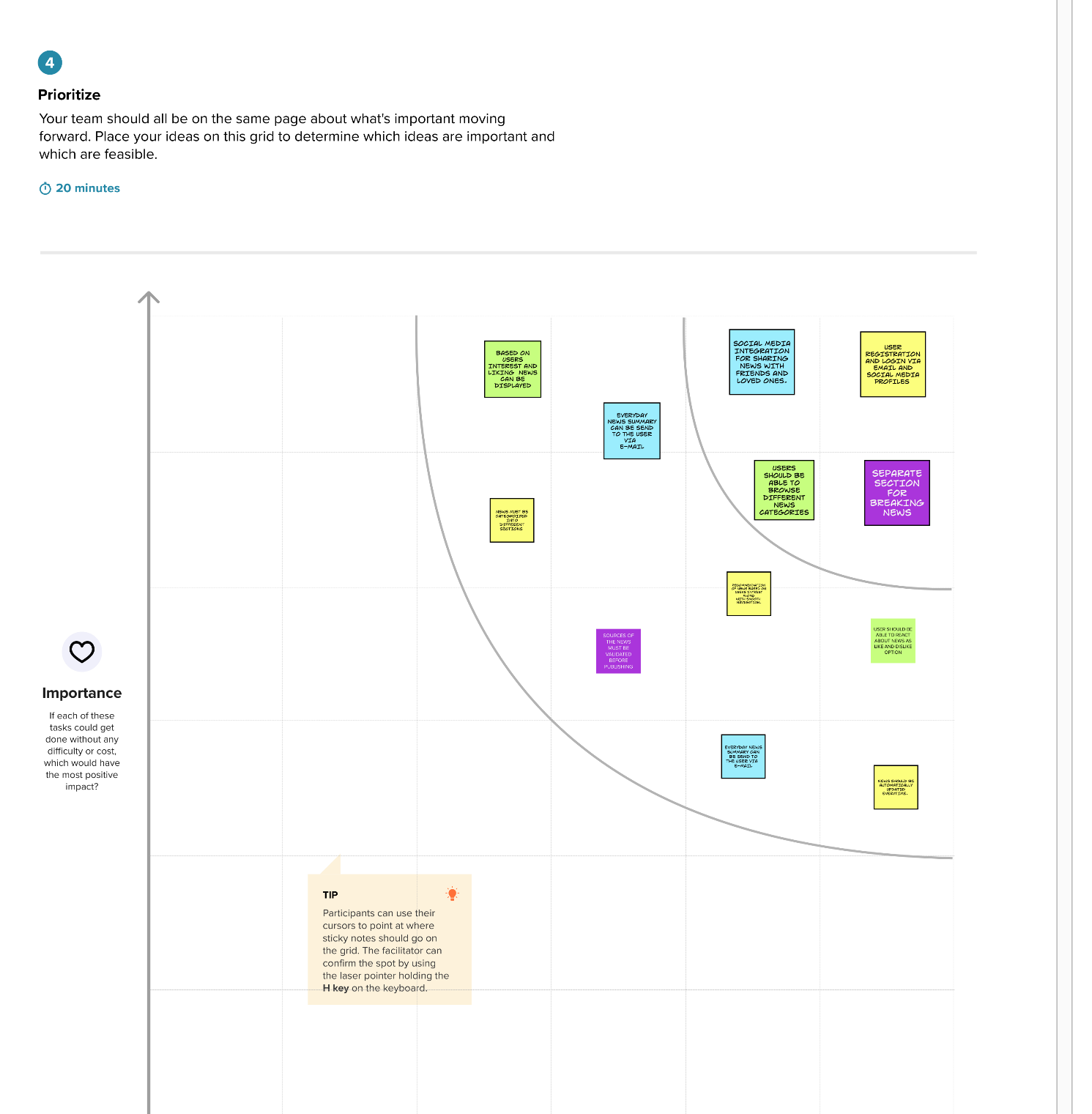
**3. IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas**

****

**3.2Ideation&Brainstorming **

****



**3.3 Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | Display news from various news sites and news platforms in a single destination along with personalising the news according to users  interests. |
| 2. | Idea / Solution description | Instead of the user having to search across the internet for news; news articles from various news sites and news platforms across the internet must be collected and displayed in an organized manner, by segregating them into various categories, at a single destination. |
| 3. | Novelty / Uniqueness | 1. Based on the user's past activity and interest,   news articles will be recommended.   1. News are categorized into various sections for the convenience of the user. 2. News is updated in real time. 3. User will have the option to select what varieties of news he would like to see. |
| 4. | Social Impact / Customer Satisfaction | 1 As news is recommended according to the  user's interests and past activity, users will find the recommendations interesting and useful. 2 Users time is greatly saved because they will never have to search through the internet to find the required news. Every news will be available at a single destination.   1. Users will have the option to customize the appearance, look and feel of the app according to their liking. 2. They can even change the way the news will be displayed in the home page according to their convenience.   These factors will surely make the customers more satisfied. |
| 5. | Business Model (Revenue Model) | The major revenue stream is the adds that are published throughout the app.  The secondary revenue stream can be from the news channels and news sites whose news will be published in this application. Based on the |
| 6. | Scalability of the Solution | As this application is hosted entirely on cloud, when there is an increase in demand, the configurations and processing power can be varied inorder to provide users with a seamless experience. |

**3.4 Problem Solution fit**

or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

**AS**

**5. AVAILABLE SOLUTIONS**

* Many online news channels are avaliable, which publishes news for people to read news online.
* But it is really hard for people to

find the right and proper news.

* Different varieties of news is avalable in the internet, but it will be effictive only if the news articles are organized properly.
* cost effective
* no need pay as uses.
* portable,interopertability.
* reliable and user friendly

**CC**

**6. CUSTOMER CONSTRAINTS**

**CS**

**1. CUSTOMER SEGMENT(S)**

* No need to buy the newspaper instantly download the application easily view the news article at anytime,anywhere,anyone.
* now world's turn as modernize so it's easily to reach news readers.

i.e. working parents of 0-5 y.o. kids

**Explore AS, differentiate**

**Define CS, fit into CC**

i

* Based on the user location provide the news articles it's usefull to the customer
* user should have the option to comment over news content and user likes,dislikes the articles should be placed.

**BE**

**7. BEHAVIOUR**

**RC**

**9. PROBLEM ROOT CAUSE**

* paid subscription for daily
* news nowadays, nobody have the time to search and read the right news from the internet.
* people find it really difficult read the daily news papers. So it will be very convinient if it possible to read news effectively via smartphones itself.
* It is really difficult for the user to get loation based news

**J&P**

**2. JOBS-TO-BE-DONE / PROBLEMS**

* Being suggeseted with inaccurate news from unreliable news channels
* every news article in the internet cannot be trusted .It can be fake news also.
* news articles that are arranged in a proper manner may cause confusion,having go through unrelatable news articles is a waste of time.
* advertising and news overloading

**Focus on J&P, tap into BE, understand RC**

**Focus on J&P, tap into BE, understand RC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identify strong TR & EM** | **3. TRIGGERS TR**   * User can mark some articles as favourites * User can be given an option to express his views and emotions by commenting about the news article in the comment section. | **10. YOUR SOLUTION SL**   * news from trusted news channel should alone be displayed. * fake news should be flagged * news articles can be categorized into different sections so that It will be user to read the required news * based on user past activity and interest news articles can be recommended | 1. **CHANNELS of BEHAVIOUR CH**  * copyrights of video,audio or someother files are mentioned clearly. * source of the news must be verified before the publishing |  |
| **4. EMOTIONS: BEFORE / AFTER EM**   * User liking news articles should be recommended to the use |

**4.REQUIREMENT ANALYSIS**

**4.1.Functional Requirement**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form Registration through Gmail Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email Confirmation via OTP |
| FR-3 | User language selection | From a list of languages, users should select a language in which the news must be displayed. |
| FR-4 | User preferences | User is asked to select the topics regarding which he would like to see the news i.e cinema, cricket, technology, climate etc. |
| FR-5 | Notification preference | User is given the option to choose the means through which he would like to receive notifications eg. SMS, email, mobile notification.  User is also given the option to select the topics on  which he would like to receive notification. |
| FR-6 | Appearance selection | Option is provided for the user to select the manner in which he wants the news to appear in the home page, how it should be organised and so on. |

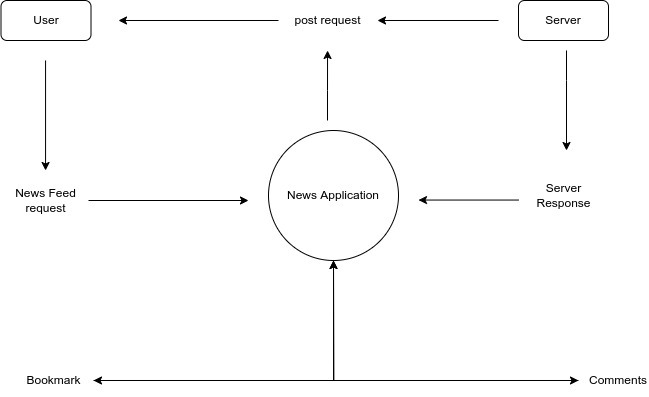
**4.2 Non-Functional Requirement**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | News articles must be fetched quickly from the internet and should be displayed as soon as the user opens the application.  While scrolling down the homepage, it shouldn't  take too long for the news articles to load. |
| NFR-2 | **Security** | Proper authentication is done to ensure that only authenticated persons are accessing the news.  The personal information of the user such as the  email, name etc is stored in an encrypted database. |

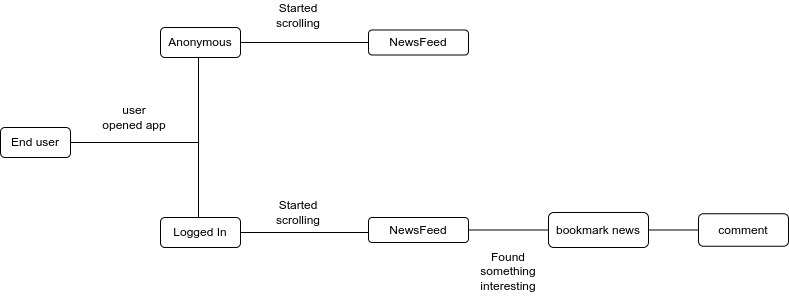
|  |  |  |
| --- | --- | --- |
| NFR-3 | **Reliability** | The server on which this application is running is configured in such a way that the connection is reliable no matter what the network traffic is. |
| NFR-4 | **Performance** | The RAM and the processing power of the server is configured in such a way that the user is able to quickly navigate across diﬀerent sections and the news articles load in no time. |
| NFR-5 | **Availability** | Irrespective of the time of the day the application should be up and running. The server configuration  is done in such a way that it is available 24/7. |
| NFR-6 | **Scalability** | This application will be hosted in IBM cloud and it will be made sure that it is easier to scale the server and storage up or down according to rise or fall of the total number of users accessing the application at a given time. |

**5.PROJECT DESIGN**

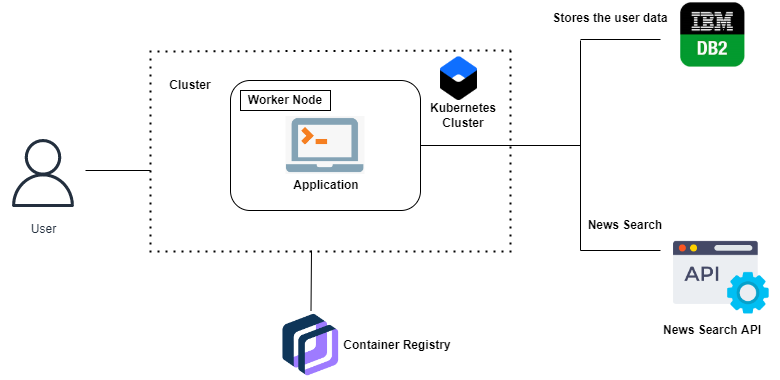
**5.1.Data Flow Diagram**

****

**5.2.1Solution Architecture**

****

**5.2.2 Technology Architecture**



**5.3 User stories**

| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| --- | --- | --- | --- | --- | --- | --- |
| Customer (Mobile user) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can access my account / dashboard | High | Sprint-1 |
|  |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
|  |  | USN-3 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | Low | Sprint-2 |
|  |  | USN-4 | As a user, I can register for the application through Gmail |  | Medium | Sprint-1 |
|  | Login | USN-5 | As a user, I can log into the application by entering email & password |  | High | Sprint-1 |
|  | Dashboard |  |  |  |  |  |
| Customer (Web user) | Registration form | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can access my account / dashboard | High | Sprint-1 |
|  |  | USN-2 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | High | Sprint-1 |
| Customer Care Executive | Query | USN-1 | As a user ,I have any queries means asked immediately | Watson assistant Bot helped to the user any time ,any where | High | Sprint-1 |
|  |  | USN-2 | As a user report the news content | Copyrights issues are solved | High | Sprint 3 |
| Administrator | Database | USN-1 | As a user | Store ,retrive the based on particular user data | High | Sprint -3 |
|  | ShortNews | USN-2 | As a user ,I can summary of the particular news content | View content like reels | High | Sprint-3 |

**6.PROJECT PLANNING & SCHEDULING**

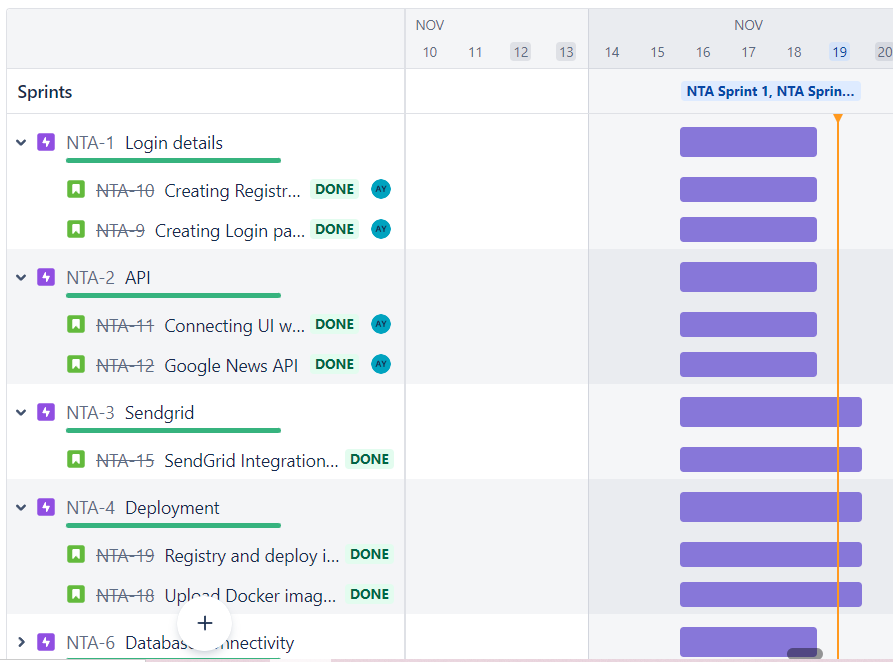
**6.1.** **Sprint Planning & Estimation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Registration | USN-1 | Creating Login page Creating Registration page | 10 | High | Arunkumar,Abishek,Jachin |
| Sprint-1 | Database Connectivity | USN-2 | To Store details of the customer Connecting UI with Database | 10 | Medium | Arunkumar  ,Abishek |
| Sprint-2 | News Tracker UI | USN-3 | Building UI News Tracker Application | 10 | High |  |
| Sprint-2 | API | USN-4 | Connecting UI with News API, Google News API | 10 | High | Abishek |
| Sprint-3 | SendGrid Integration | USN-5 | SendGrid Integration With Python Code | 10 | Low |  |
| Sprint-3 | News Reader (Voice) | USN-6 | Building Voice Assistant to read the news | 10 | Medium | Jachin,Arun |
| Sprint-4 | Containerization | USN-7 | Containerizing the app | 10 | High |  |
| Sprint -4 | Upload image and deployment | USN-8 | Upload Docker image to the IBM Registry and deploy it in the Kubernetes Cluster | 10 | High |  |

**6.2.Sprint Delivery Schedule**

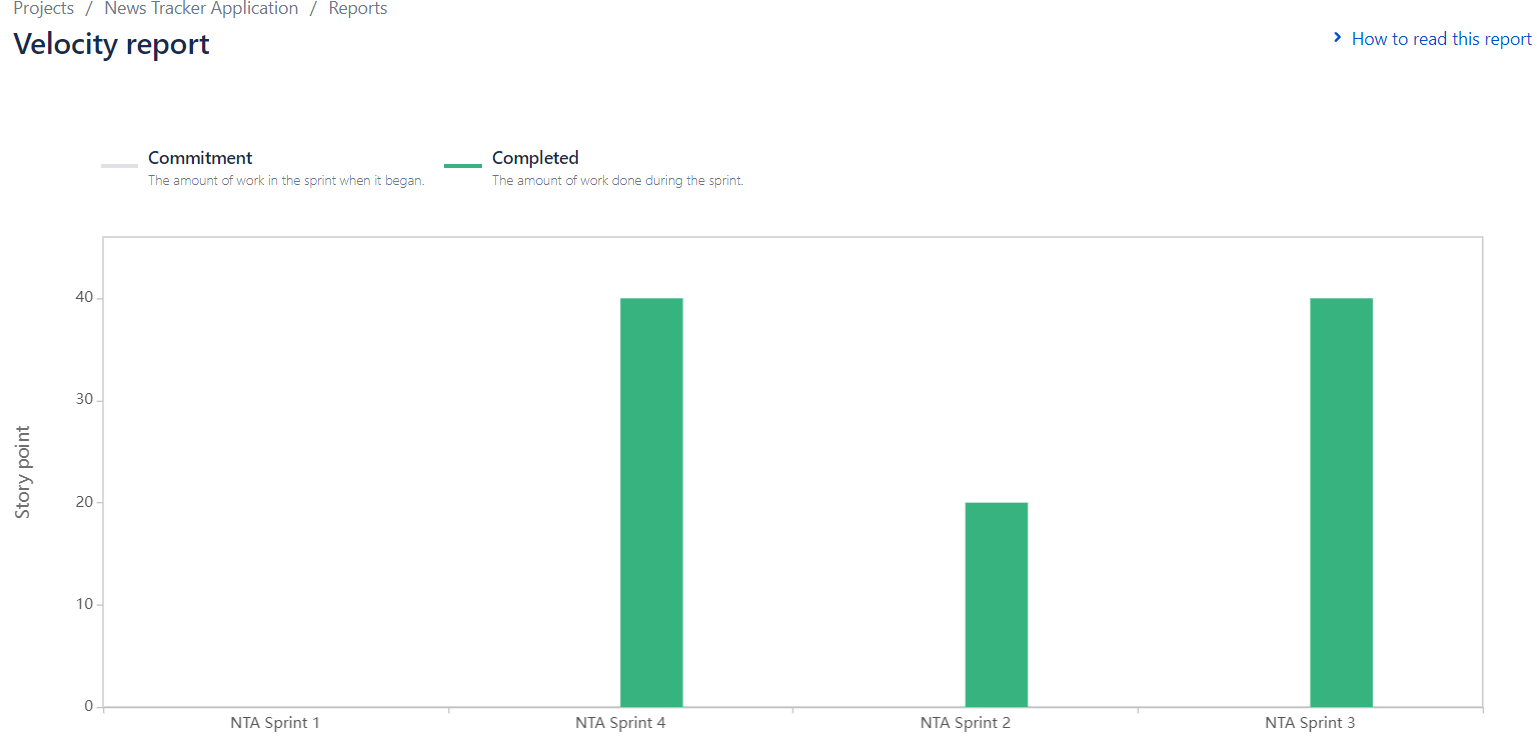
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on**  **Planned End Date)** | **Sprint Release Date (Actual)** |
| Sprint-1 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 20 | 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

**6.3.Report from Jira**

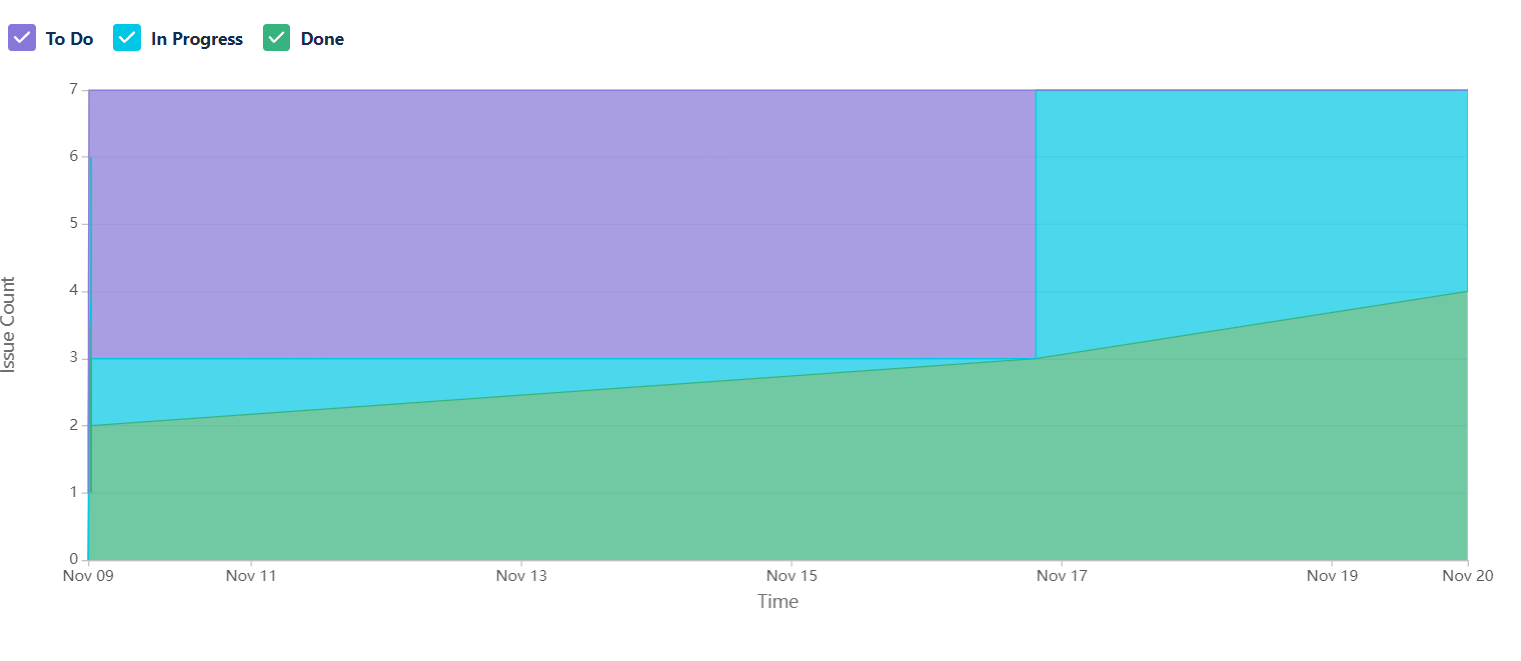
****

****

**Velocity report**

****

**Cumulative Report**

****

**7. CODING & SOLUTIONING**

**7.1.Bookmark**

import Header from "@components/header";

import News from "@components/news";

import { isMobile } from "react-device-detect";

import { Swiper, SwiperSlide } from "swiper/react";

import "swiper/bundle";

import "swiper/css";

import { useEffect, useState } from "react";

import BottomNav from "@components/bottomNav";

import Select from "@components/Select";

import DialogComponent from "@components/Dialog";

import { unstable\_getServerSession } from "next-auth";

import { authOptions } from "./api/auth/[...nextauth]";

export default function IndexPage({ data }: any) {

const [space, setSpace] = useState(0);

const [currentData, setCurrentData] = useState<any>([]);

const [swiperRef, setSwiperRef] = useState();

useEffect(() => {

if (!isMobile && typeof window !== "undefined") {

setSpace(-80);

}

}, []);

useEffect(() => {

// setCurrentData(data);

const parsed = JSON.parse(data);

const filtered = parsed.map((item: any) => JSON.parse(item.CONTENT));

setCurrentData(filtered);

}, [data]);

return data ? (

<>

<Header />

<Select />

<Swiper

// @ts-ignore

onSwiper={setSwiperRef}

spaceBetween={space}

direction={"vertical"}

mousewheel={true}

className="mySwiper"

>

{currentData?.length &&

currentData.map((item: any, i: number) => {

return (

<SwiperSlide key={`${Date.now()}\_${item.id}\_${i}`}>

<News data={item} />

</SwiperSlide>

);

})}

</Swiper>

<BottomNav swiperRef={swiperRef} />

<DialogComponent />

</>

) : (

<div>Loading...</div>

);

}

export async function getServerSideProps({ req, res, query }: any) {

res.setHeader(

"Cache-Control",

"public, s-maxage=10, stale-while-revalidate=59"

);

const session = await unstable\_getServerSession(req, res, authOptions);

if (!session) {

return {

redirect: {

permanent: false,

destination: "/api/auth/signin",

},

};

}

try {

const response = await fetch(`${process.env.SERVER\_URL}getbookmarks`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email: session.user?.email,

}),

});

const data = await response.json();

if (!data.success)

return {

props: { data: null },

};

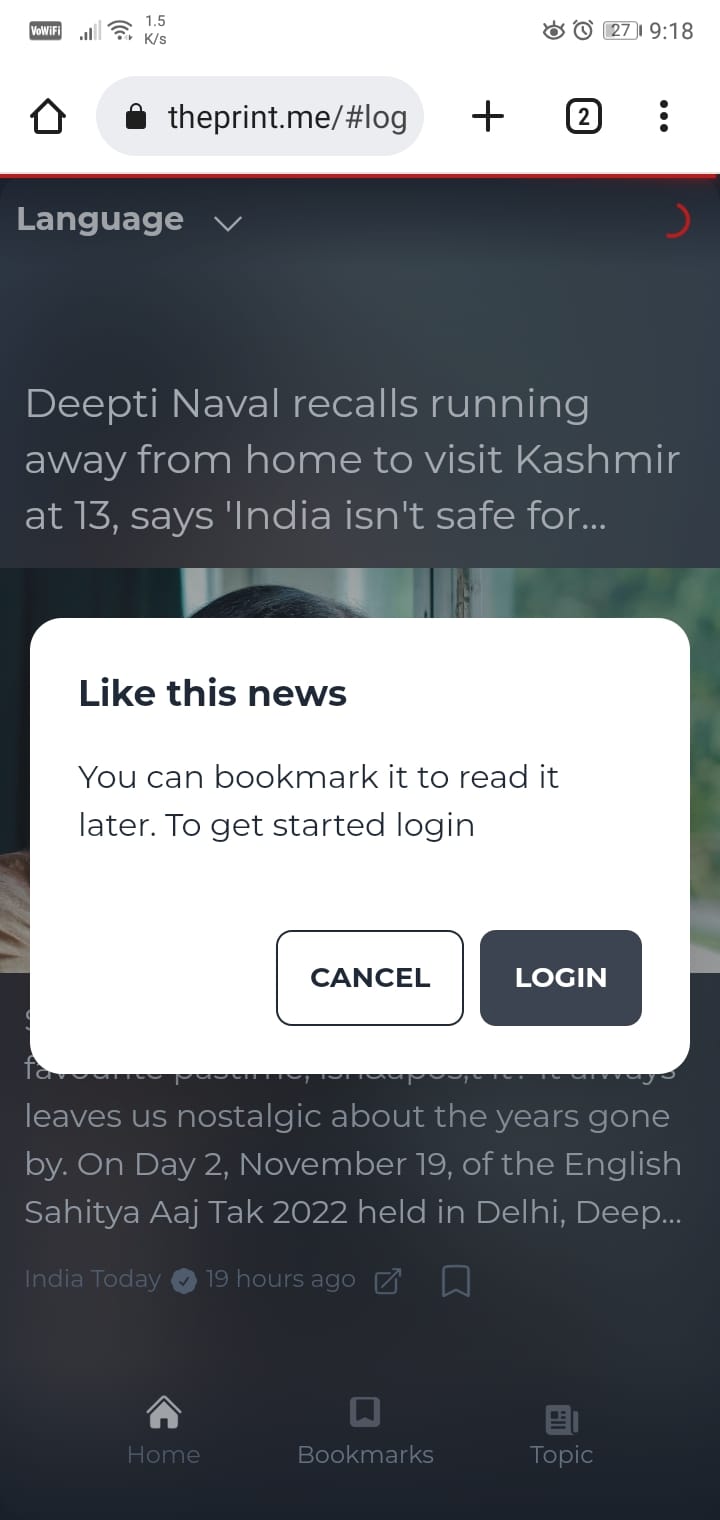
return { props: { data: data.bookmarks } };

} catch (err) {

console.log(err);

}

}



**7.2.Choosetopics**

import { NewspaperIcon } from "@heroicons/react/24/solid";

import \* as Popover from "@radix-ui/react-popover";

import \* as ScrollArea from "@radix-ui/react-scroll-area";

import { useRouter } from "next/router";

export default function ChooseLang({ swiperRef }: any) {

const router = useRouter();

const handleClick = (topic = "For You") => {

swiperRef?.slideTo(0);

router.query.topic = topic;

router.push(router);

const body = document.querySelector("body");

body?.click();

};

const TOPICS = [

"For You",

"Business",

"Entertainment",

"Technology",

"Politics",

"Movies",

"India",

];

return (

<Popover.Root>

<Popover.Trigger>

<div className="flex items-center flex-col cursor-pointer pt-4 mb-4">

<NewspaperIcon className="h-5 w-5 mt-1 text-gray-500" />

<p className="text-slate-400 text-xs">Topic</p>

</div>

</Popover.Trigger>

<Popover.Portal>

<Popover.Content className="PopoverContent">

<ScrollArea.Root className="ScrollAreaRoot">

<ScrollArea.Viewport className="ScrollAreaViewport">

<ul className="menu compact bg-base-100 p-2">

{TOPICS.map((itm: string, i: number) => (

<li key={`TOPICS\_RENDERED\_${i}`}>

<a onClick={() => handleClick(itm)}>{itm}</a>

</li>

))}

</ul>

</ScrollArea.Viewport>

<ScrollArea.Scrollbar

className="ScrollAreaScrollbar bg-slate-200"

orientation="vertical"

>

<ScrollArea.Thumb className="ScrollAreaThumb bg-slate-600" />

</ScrollArea.Scrollbar>

<ScrollArea.Corner className="ScrollAreaCorner" />

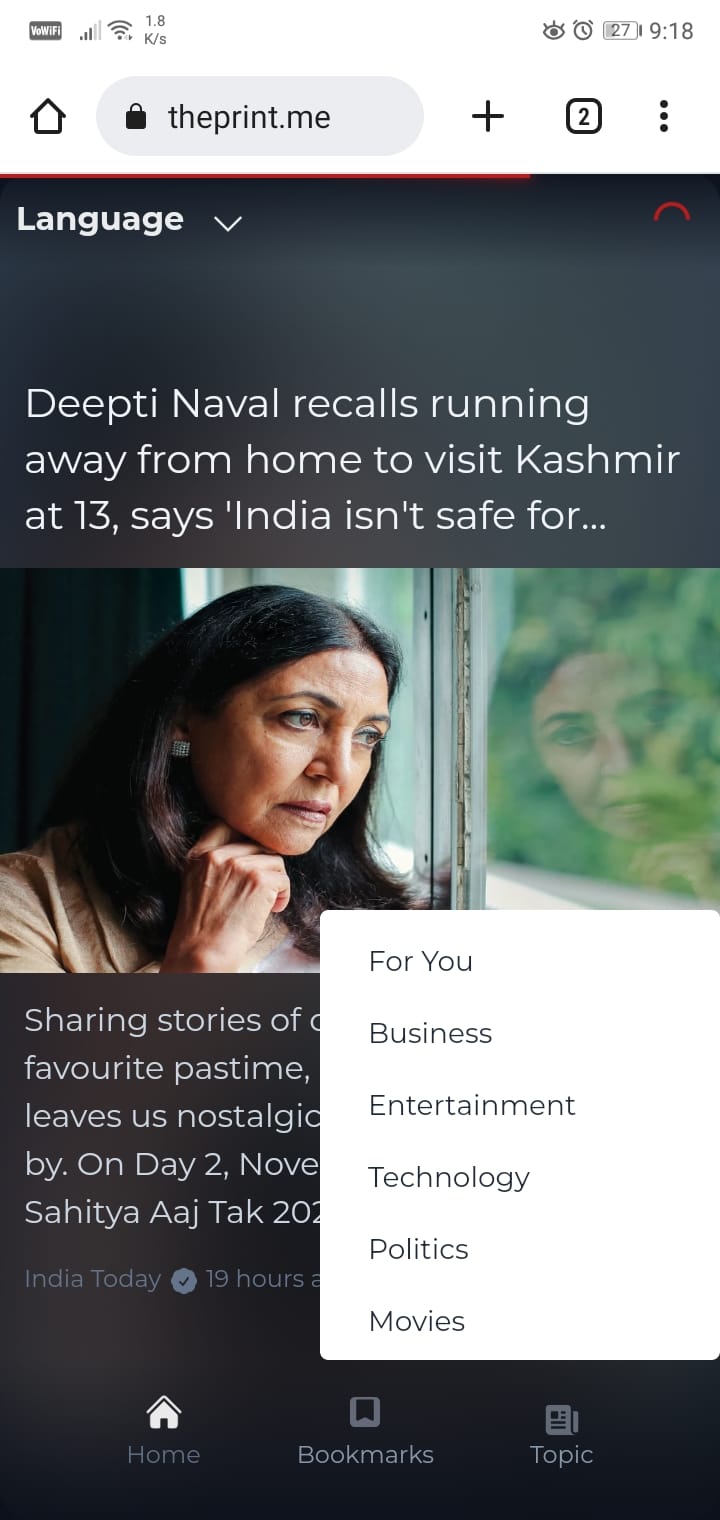
</ScrollArea.Root>

</Popover.Content>

</Popover.Portal>

</Popover.Root>

);

}

**7.3.news feed**

import type { NextApiRequest, NextApiResponse } from "next";

import { ALLOWED\_ORIGINS } from "../../lib/origins";

type Data = {

data: [];

next: null | string;

error?: string;

nextIndex?: string;

activeTopic?: string;

};

export default async function handler(

req: NextApiRequest,

res: NextApiResponse<Data>

) {

try {

const { url, nextIndex, activeTopic } = req.body;

const { origin } = req.headers;

if (origin && ALLOWED\_ORIGINS.indexOf(origin) === -1) {

return res.status(403).json({ data: [], error: "Forbidden", next: null });

}

res.setHeader("Access-Control-Allow-Origin", origin || "\*");

if (typeof url !== "string")

return res

.status(400)

.json({ error: "Invalid url", data: [], next: null });

const parsedURL = `${process.env.API\_URL}?url=${encodeURIComponent(

url

)}&nextIndex=${nextIndex}&activeTopic=${activeTopic}&activeNavIndex=0&topicEngName=${activeTopic}`;

const response = await fetch(parsedURL);

const json = await response.json();

res.status(200).json({

data: json?.data?.rows || [],

next: json?.url || null,

nextIndex: json?.nextIndex || null,

activeTopic: json?.activeTopic || null,

});

} catch (err) {

console.log(err);

res.status(500);

}

}

**7.4.Database schema**

from flask import Flask, request, jsonify

from flask\_cors import CORS, cross\_origin

import os

from os.path import join, dirname

import ibm\_db

from dotenv import load\_dotenv

from threading import Thread

from PIL import Image

import requests

from io import BytesIO

import blurhash

import numpy

import json

app = Flask(\_\_name\_\_)

cors = CORS(app)

app.config['CORS\_HEADERS'] = 'Content-Type'

dotenv\_path = join(dirname(\_\_file\_\_), '.env')

load\_dotenv(dotenv\_path)

connectionstr = os.environ.get('DB2\_CONNECTION\_STRING')

conn = ibm\_db.connect(connectionstr, '', '')

def userPresent(email=None):

if email:

sql = "SELECT Email FROM User WHERE Email = ?"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt)

if account:

return "true"

return "false"

@app.route("/")

def hello\_world():

return "<p>Welcome to news tracker api</p>"

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

def db2():

email = request.json['email']

isPresent = userPresent(email)

return isPresent

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

def createuser():

try:

Thread(target=userTask, args=(

request.json['email'], request.json['name'])).start()

return jsonify(success=True, message="User created")

except:

return jsonify(success=False, error="Missing email or name")

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

def getblurhash():

url = request.json['url']

response = requests.get(url)

hash = blurhash.encode(numpy.array(Image.open(

BytesIO(response.content)).convert("RGB")))

print(hash)

return jsonify(hash=hash)

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

@cross\_origin()

def Bookmark():

try:

Thread(target=bookMarkTask, args=(

request.json['email'], request.json['content'])).start()

return jsonify(success=True, message="Bookmarked")

except:

return jsonify(success=False, error="Missing email or content")

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

def getbookmarks():

email = request.json['email']

sql = "SELECT Content FROM Bookmark WHERE Email = ?"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.execute(stmt)

response = []

bookmarks = ibm\_db.fetch\_assoc(stmt)

if not bookmarks:

return jsonify(success=True, error="No bookmarks found")

while bookmarks != False:

response.append(bookmarks)

bookmarks = ibm\_db.fetch\_assoc(stmt)

response = json.dumps(response)

return jsonify(success=True, bookmarks=response)

def bookMarkTask(email, content=None):

sql = "INSERT INTO Bookmark (Email, Content) VALUES(?, ?)"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.bind\_param(stmt, 2, content)

ibm\_db.execute(stmt)

def userTask(email, name=''):

isPresent = userPresent(email)

if isPresent != "true":

sql = "INSERT INTO User (Name, Email) VALUES (?, ?)"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, name)

ibm\_db.bind\_param(stmt, 2, email)

ibm\_db.execute(stmt)

print('created user')



**8.TESTING**

**8.1.Test case**

The Test cases for the News Tracker application are as follows

● Verify If user can Sign up to the account

● Verify If already signed up user cannot log into the account

● Verify if user is able to see Login/Register when clicked on it

● Verify if user is able to filter articles based on categories

● Verify if user is able to see detailed information when clicked on read more

**8.2.User Acceptance Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resolution** | **Severity 1** | **Severity 2** | **Severity 3** | **Severity 4** | **Subtotal** |
| By Design | 10 | 3 | 4 | 2 | 19 |
| Duplicate | 0 | 1 | 0 | 0 | 1 |
| External | 2 | 0 | 1 | 0 | 3 |
| Fixed | 10 | 3 | 4 | 15 | 32 |
| Not Reproduced | 0 | 0 | 0 | 1 | 1 |
| Skipped | 0 | 0 | 1 | 0 | 1 |
| Won't Fix | 1 | 0 | 1 | 0 | 2 |
| Totals | 23 | 7 | 11 | 18 | 58 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **Fail** | **Pass** |
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 35 | 0 | 0 | 35 |
| Security | 5 | 0 | 0 | 5 |
| Outsource Shipping | 0 | 0 | 0 | 0 |
| Exception Reporting | 15 | 0 | 0 | 15 |
| Final Report Output | 6 | 0 | 0 | 6 |
| Version Control | 2 | 0 | 0 | 2 |

**9.RESULT**

**9.1.Performance metrics**

#### CPU usage

The Python V3.7.0 is make the best use of the CPU. For every loop the program runs in O(1) time, neglecting the network and communication. The program sleeps for every 1 second for better communication with MQTT. As the program takes O(1) time and the compiler optimizes the program during compilation there is less CPU load for each cycle. The upcoming instructions are on the stack memory, so they can be popped after execution.

#### Memory usage

The sensor values, networking data are stored in sram of the ESP32 . It’s a lot of data because ESP32 has only limited amount of memory (520 KB) .For each memory cycle the exact addresses are overwritten with new values to save memory and optimal execution of the program

#### 

#### Garbage collection

In the server-side garbage collection is done by the React framework.python does not have any garbage collection features. But it is not necessary in this scenario as the memory is used again for storing the data. Any dangling pointer or poorly handled address space is not allocated.

**10.ADVANTAGE AND DISADVANTAGE**

**Advantages**

* As the news articles are properly categorised into different sections, user finds it easier to find the right news.
* News articles are displayed in the home page in an order such that the important news always appears on the top.
* The UI of the app is designed in such a way that it is easier for the user to navigate**.**

**Disadvantage**

* Users are unable to express their opinion by commenting and liking the news articles.
* Currently the app won’t let the user to read news

**11.CONCLUSION**

The way we consume news has shifted dramatically in the last decade and having a dedicated website is no longer enough. Users expect updates to be immediately available and accessible via multiple devices, and easy to share across their social media networks. News apps have also become increasingly important for users who want to avoid consuming news via social media and digest news from a reliable source.

**12.FUTURE SCOPE**

News content along with the video it will be more glad to use the application and easily user can understand the subject of the news in a short duration of time(60 seconds).it just like reels.

We plan to provide community based news.

**13.APPENDIX**

**Source code**

**//index page of the app**

import Header from "@components/header";

import News from "@components/news";

import { isMobile } from "react-device-detect";

import { Swiper, SwiperSlide } from "swiper/react";

import "swiper/bundle";

import "swiper/css";

import { useEffect, useState } from "react";

import BottomNav from "@components/bottomNav";

import Select from "@components/Select";

import DialogComponent from "@components/Dialog";

export default function IndexPage({ data, next, nextIndex, activeTopic }: any) {

const [space, setSpace] = useState(0);

const [isSent, setIsSent] = useState(false);

const [currentNext, setCurrentNext] = useState("");

const [currentData, setCurrentData] = useState<any>([]);

const [swiperRef, setSwiperRef] = useState();

const [currentIndex, setCurrentIndex] = useState<any>();

const [currentTopic, setCurrentTopic] = useState();

useEffect(() => {

if (!isMobile && typeof window !== "undefined") {

setSpace(-80);

}

}, []);

useEffect(() => {

setCurrentNext(next);

}, [next]);

useEffect(() => {

setCurrentData(data);

}, [data]);

useEffect(() => {

setCurrentIndex(nextIndex);

}, [nextIndex]);

useEffect(() => {

setCurrentTopic(activeTopic);

}, [activeTopic]);

const handleChange = async (e: any) => {

if (isSent) {

return;

}

const reachedEnd = e.realIndex > e.slides.length - 5;

try {

if (reachedEnd) {

setIsSent(true);

const ni = parseInt(currentIndex) + 16;

const url = window.location.origin;

const response = await fetch(`${url}/api/next`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

url: encodeURIComponent(currentNext),

nextIndex: ni,

activeTopic: currentTopic,

}),

});

const json = await response.json();

setCurrentIndex(ni);

setCurrentTopic(json.activeTopic);

if (json.data) {

setCurrentData((old: []) => [...old, ...json.data]);

setIsSent(false);

}

if (!json.next) {

setIsSent(true);

return;

}

if (json.next) {

setCurrentNext(decodeURIComponent(json.next));

}

// @ts-ignore

swiperRef?.update();

}

} catch (err) {

console.log(err);

}

};

return (

<>

<Header />

<Select />

<Swiper

// @ts-ignore

onSwiper={setSwiperRef}

spaceBetween={space}

direction={"vertical"}

mousewheel={true}

className="mySwiper"

onSlideChange={handleChange}

>

{currentData?.length &&

currentData.map((item: any, i: number) => {

return (

<SwiperSlide key={`${Date.now()}\_${item.id}\_${i}`}>

<News data={item} />

</SwiperSlide>

);

})}

</Swiper>

<BottomNav swiperRef={swiperRef} />

<DialogComponent />

</>

);

}

export async function getServerSideProps({ req, res, query }: any) {

const lang = query.lang || "english";

const topic = query.topic || "For You";

res.setHeader(

"Cache-Control",

"public, s-maxage=10, stale-while-revalidate=59"

);

try {

const url =

process.env.NODE\_ENV !== "production"

? "http://localhost:3000"

: "https://theprint.me";

const encodedUri = encodeURI(`lang=${lang}&topic=${topic}`);

const response = await fetch(`${url}/api/headlines?${encodedUri}`);

const { data, next, nextIndex, activeTopic } = await response.json();

return { props: { data, next, nextIndex, activeTopic } };

} catch (err) {

console.log(err);

}

// Pass data to the page via props

}

//bookmarks

import Header from "@components/header";

import News from "@components/news";

import { isMobile } from "react-device-detect";

import { Swiper, SwiperSlide } from "swiper/react";

import "swiper/bundle";

import "swiper/css";

import { useEffect, useState } from "react";

import BottomNav from "@components/bottomNav";

import Select from "@components/Select";

import DialogComponent from "@components/Dialog";

import { unstable\_getServerSession } from "next-auth";

import { authOptions } from "./api/auth/[...nextauth]";

export default function IndexPage({ data }: any) {

const [space, setSpace] = useState(0);

const [currentData, setCurrentData] = useState<any>([]);

const [swiperRef, setSwiperRef] = useState();

useEffect(() => {

if (!isMobile && typeof window !== "undefined") {

setSpace(-80);

}

}, []);

useEffect(() => {

// setCurrentData(data);

const parsed = JSON.parse(data);

const filtered = parsed.map((item: any) => JSON.parse(item.CONTENT));

setCurrentData(filtered);

}, [data]);

return data ? (

<>

<Header />

<Select />

<Swiper

// @ts-ignore

onSwiper={setSwiperRef}

spaceBetween={space}

direction={"vertical"}

mousewheel={true}

className="mySwiper"

>

{currentData?.length &&

currentData.map((item: any, i: number) => {

return (

<SwiperSlide key={`${Date.now()}\_${item.id}\_${i}`}>

<News data={item} />

</SwiperSlide>

);

})}

</Swiper>

<BottomNav swiperRef={swiperRef} />

<DialogComponent />

</>

) : (

<div>Loading...</div>

);

}

export async function getServerSideProps({ req, res, query }: any) {

res.setHeader(

"Cache-Control",

"public, s-maxage=10, stale-while-revalidate=59"

);

const session = await unstable\_getServerSession(req, res, authOptions);

if (!session) {

return {

redirect: {

permanent: false,

destination: "/api/auth/signin",

},

};

}

try {

const response = await fetch(`${process.env.SERVER\_URL}getbookmarks`, {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email: session.user?.email,

}),

});

const data = await response.json();

if (!data.success)

return {

props: { data: null },

};

return { props: { data: data.bookmarks } };

} catch (err) {

console.log(err);

}

}

**//API**

import { JSDOM } from "jsdom";

import { NextApiRequest, NextApiResponse } from "next";

import { ALLOWED\_ORIGINS } from "../../lib/origins";

const order = ["For You"];

const fetchNParse = async (url: string) => {

try {

const data = await fetch(url);

const html = await data.text();

const dom = new JSDOM(html, { runScripts: "dangerously" });

const response = dom.window.\_\_STATE.topicsList || {};

return response;

} catch (err) {

console.log(err);

return null;

}

};

function parseTopic(topic = "For You", data: any) {

let topics: any = null;

if (topic === "For You") {

topics = data[0];

} else {

topics = data.find((tp: any) => tp.name === topic);

}

if (topics) {

return {

data: topics?.data?.data.rows || [],

next: topics.data?.data?.nextPageUrl || null,

nextIndex: topics.data?.data?.count,

activeTopic: topics?.topicType,

};

}

}

export default async function getHeadlines(

req: NextApiRequest,

res: NextApiResponse

) {

const { query } = req;

const { lang, topic } = query;

let base\_url = process.env.BASE\_URL;

const { origin } = req.headers;

if (origin && ALLOWED\_ORIGINS.indexOf(origin) === -1) {

return res.status(403).json({ data: [], error: "Forbidden", next: null });

}

res.setHeader("Access-Control-Allow-Origin", origin || "\*");

if (typeof lang !== "string")

return res.status(400).json({ error: "Invalid location" });

if (typeof topic !== "string")

return res.status(400).json({ error: "Invalid topic" });

let response = null;

if (topic === "For You") {

base\_url = process.env.BASE\_URL?.replace("english", lang) || "";

} else {

base\_url = process.env.BASE\_URL?.replace("for+you", topic) || "";

}

response = await fetchNParse(base\_url);

const news = parseTopic(topic, response);

res.status(200).json(news);

}

//next.ts this file show the next feed of the news content

import type { NextApiRequest, NextApiResponse } from "next";

import { ALLOWED\_ORIGINS } from "../../lib/origins";

type Data = {

data: [];

next: null | string;

error?: string;

nextIndex?: string;

activeTopic?: string;

};

export default async function handler(

req: NextApiRequest,

res: NextApiResponse<Data>

) {

try {

const { url, nextIndex, activeTopic } = req.body;

const { origin } = req.headers;

if (origin && ALLOWED\_ORIGINS.indexOf(origin) === -1) {

return res.status(403).json({ data: [], error: "Forbidden", next: null });

}

res.setHeader("Access-Control-Allow-Origin", origin || "\*");

if (typeof url !== "string")

return res

.status(400)

.json({ error: "Invalid url", data: [], next: null });

const parsedURL = `${process.env.API\_URL}?url=${encodeURIComponent(

url

)}&nextIndex=${nextIndex}&activeTopic=${activeTopic}&activeNavIndex=0&topicEngName=${activeTopic}`;

const response = await fetch(parsedURL);

const json = await response.json();

res.status(200).json({

data: json?.data?.rows || [],

next: json?.url || null,

nextIndex: json?.nextIndex || null,

activeTopic: json?.activeTopic || null,

});

} catch (err) {

console.log(err);

res.status(500);

}

}

//server.py this file is used to the fetch the data from db and insert the data to db connect the flask project to db

from flask import Flask, request, jsonify

from flask\_cors import CORS, cross\_origin

import os

from os.path import join, dirname

import ibm\_db

from dotenv import load\_dotenv

from threading import Thread

from PIL import Image

import requests

from io import BytesIO

import blurhash

import numpy

import json

app = Flask(\_\_name\_\_)

cors = CORS(app)

app.config['CORS\_HEADERS'] = 'Content-Type'

dotenv\_path = join(dirname(\_\_file\_\_), '.env')

load\_dotenv(dotenv\_path)

connectionstr = os.environ.get('DB2\_CONNECTION\_STRING')

conn = ibm\_db.connect(connectionstr, '', '')

def userPresent(email=None):

if email:

sql = "SELECT Email FROM User WHERE Email = ?"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt)

if account:

return "true"

return "false"

@app.route("/")

def hello\_world():

return "<p>Welcome to news tracker api</p>"

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

def db2():

email = request.json['email']

isPresent = userPresent(email)

return isPresent

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

def createuser():

try:

Thread(target=userTask, args=(

request.json['email'], request.json['name'])).start()

return jsonify(success=True, message="User created")

except:

return jsonify(success=False, error="Missing email or name")

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

def getblurhash():

url = request.json['url']

response = requests.get(url)

hash = blurhash.encode(numpy.array(Image.open(

BytesIO(response.content)).convert("RGB")))

print(hash)

return jsonify(hash=hash)

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

@cross\_origin()

def Bookmark():

try:

Thread(target=bookMarkTask, args=(

request.json['email'], request.json['content'])).start()

return jsonify(success=True, message="Bookmarked")

except:

return jsonify(success=False, error="Missing email or content")

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

def getbookmarks():

email = request.json['email']

sql = "SELECT Content FROM Bookmark WHERE Email = ?"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.execute(stmt)

response = []

bookmarks = ibm\_db.fetch\_assoc(stmt)

if not bookmarks:

return jsonify(success=True, error="No bookmarks found")

while bookmarks != False:

response.append(bookmarks)

bookmarks = ibm\_db.fetch\_assoc(stmt)

response = json.dumps(response)

return jsonify(success=True, bookmarks=response)

def bookMarkTask(email, content=None):

sql = "INSERT INTO Bookmark (Email, Content) VALUES(?, ?)"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.bind\_param(stmt, 2, content)

ibm\_db.execute(stmt)

def userTask(email, name=''):

isPresent = userPresent(email)

if isPresent != "true":

sql = "INSERT INTO User (Name, Email) VALUES (?, ?)"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, name)

ibm\_db.bind\_param(stmt, 2, email)

ibm\_db.execute(stmt)

print('created user')

//global.css

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500;700;900&display=swap');

@tailwind base;

@tailwind components;

@tailwind utilities;

#\_\_next { height: 100% }

body {

min-height: 100vh;

min-height: -webkit-fill-available;

}

html {

height: -webkit-fill-available;

}

html,

body {

position: relative;

height: 100%;

background: linear-gradient(to bottom, #141b29, #0c111b 300px);

overflow: hidden;

font-family: 'Montserrat', sans-serif;

}

.swiper {

width: 100%;

height: 100%;

}

.swiper-slide {

@apply flex justify-center;

}

.PopoverContent {

transform-origin: var(--radix-popover-content-transform-origin);

animation: scaleIn 0.5s ease-out;

}

@keyframes scaleIn {

from {

opacity: 0;

transform: scale(0);

}

to {

opacity: 1;

transform: scale(1);

}

}

.PopoverContent {

animation-duration: 0.6s;

animation-timing-function: cubic-bezier(0.16, 1, 0.3, 1);

}

.PopoverContent[data-side='top'] {

animation-name: slideUp;

}

.PopoverContent[data-side='bottom'] {

animation-name: slideDown;

}

[data-radix-popper-content-wrapper] {

z-index: 1 !important;

}

.ScrollAreaRoot {

width: 200px;

height: 225px;

border-radius: 4px;

overflow: hidden;

--scrollbar-size: 10px;

}

.ScrollAreaViewport {

width: 100%;

height: 100%;

border-radius: inherit;

}

.ScrollAreaScrollbar {

display: flex;

user-select: none;

touch-action: none;

padding: 2px;

width: 7px;

transition: background 160ms ease-out;

}

.ScrollAreaScrollbar[data-orientation='horizontal'] {

flex-direction: column;

height: var(--scrollbar-size);

}

.ScrollAreaThumb {

flex: 1;

border-radius: var(--scrollbar-size);

position: relative;

}

.DialogOverlay {

background-color: var(--blackA9);

position: fixed;

inset: 0;

animation: overlayShow 150ms cubic-bezier(0.16, 1, 0.3, 1);

}

.DialogContent {

background-color: white;

border-radius: 6px;

box-shadow: hsl(206 22% 7% / 35%) 0px 10px 38px -10px, hsl(206 22% 7% / 20%) 0px 10px 20px -15px;

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

width: 90vw;

max-width: 450px;

max-height: 85vh;

padding: 25px;

animation: contentShow 150ms cubic-bezier(0.16, 1, 0.3, 1);

}

.DialogContent:focus {

outline: none;

}

@keyframes slideDown {

from {

opacity: 0;

transform: translateY(-10px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

@keyframes slideUp {

from {

opacity: 0;

transform: translateY(10px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

//util.ts

const documentHeight = () => {

const doc = document.documentElement;

doc.style.setProperty("--doc-height", `${window.innerHeight}px`);

};

export default documentHeight;

//Kubernetes file

apiVersion: apps/v1

kind: Deployment

metadata:

name: flask-server

spec:

replicas: 3

selector:

matchLabels:

app: flask-server

template:

metadata:

labels:

app: flask-server

spec:

containers:

- name: flask-server

image: icr.io/abishek/flask-server

imagePullPolicy: Always

ports:

- containerPort: 8080

protocol: TCP

---

apiVersion: v1

kind: Service

metadata:

name: flask-server-service

spec:

type: ClusterIP

ports:

- port: 8080

selector:

app: flask-server

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: flask-server-ingress

annotations:

kubernetes.io/ingress.class: nginx

nginx.ingress.kubernetes.io/ssl-redirect: "false"

spec:

rules:

- http:

paths:

- backend:

service:

name: flask-server-service

port:

number: 8080

path: /

pathType: Prefix

//Dockerfile

FROM python:3.8

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8080

CMD ["waitress-serve", "--host", "0.0.0.0", "server:app"]

**Github link**: <https://github.com/IBM-EPBL/IBM-Project-21574-1659784969>

**Video Link:**