NEWS TRACKER

1. INTRODUCTION

Understanding users' online behaviour is of growing interest to academic researchers in a variety of fields. Traditionally, in the marketing domain, commercial research companies map consumer behaviour to understand when and where customers decide to buy products. For this purpose, Web metrics of individual websites serve as detailed source of information on when, how, and at which section a user enters a website. Recently this type of data is also being used by cultural heritage institutes to understand the interest of their visitors to track where their digital content is being reused or to understand the query's users perform in search systems by analysing the log files. In this type of research, the website is the central research object providing traces that calls 'Horizontal Data sets'. These contain data that are 'organized around a specific type of trace, for example search terms, web browsing log files, tweets, hashtags, likes or friend and follower ties'. An advantage of using this type of data is that they are not obtrusive to the respondents, since they are created automatically as users are surfing the Web. However, this also leads to an ethical disadvantage, since users are not aware that their online behaviour is being examined, nor could they give their consent to have their data being analysed. While Horizontal data sets are organized around one type of trace, Vertical data sets are organized around research participants that deliberately 'give permission for researchers to collect their digital traces'.

Since mid–1990s of the previous century, commercial research agencies have started to collect these types of vertical data by building tools and panels of respondents whose online behaviour is monitored 24/7 to provide data on usage across media and purchase behaviour. In the USA, companies such as comScore and Nielsen created 'Online Netview Panels', while in The Netherlands, TNS Nipo and Wakoopa offer similar tools to create aggregated lists of the most visited websites on all platforms and devices. Similar to television viewing rates, these lists are mainly created to gain more insight in the background of website visitors to provide potential advertisers with information on how to reach their online target audience in the best possible manner. Obviously these commercial research data contain wealthy information, also for academics who are interested in collecting real-world Web use data. However, apart from lists

of the most popular domains that are published as open data by companies such as Alexa and Similar web,data containing information about visits to each individual page and information about the background of the panel are not available. Main arguments of commercial agencies to not collaborate with scholars are to ensure the confidentiality of their respondents' identity and to prevent scholars to gain insight into the techniques applied by the companies.

1.1 PROJECT OVERVIEW

We designed the Newstracker to study how the consumption of news websites fits in the daily surfing behaviour of university students. We tracked the Web behaviour of forty-two university students who used their laptop as their main informational device and agreed to have their browsing behaviour on their laptops being monitored in the period April–July 2015. We found the respondents through the personal network of our research assistants. Each assistant was the main contact person for around 12 respondents whom they did not know personally. They were in touch with them several times a week, creating mutual trust, guaranteeing their privacy, and preventing the respondents to quit their participation. Each respondent signed a consent form and filled in a survey about their background and current news use. At the end of the tracking period, they filled in an exit survey which results indicate the Internet speed was not affected by the implementation of the proxy in the respondents' browser. They did indicate they were aware of the tool running when starting up their browsers but soon forgot about it also because we had minimal breakdowns of the server which we were able to fix soon. To gain a better understanding of the registered browsing behaviour, we held in-depth interviews with twenty of our respondents.

1.2 PURPOSE

Setting up such a multimethod design can obviously only be done with a relatively small group of respondents, given the labour-intensive nature of conducting in-depth interviews. We are of course aware that especially commercial research agencies are mainly interested in the big data nature of large-scale monitoring studies to give detailed information of website visitors to the owners of the websites. However, as our research findings suggest, also those types of studies should be aware that a website click does not automatically reflect a uniform interest of the visitors.Our multimethod design enabled us to first register online browsing behaviour and

then find explanations for the registered browsing patterns. Though this allowed us to understand more fully how the consumption of news websites fits in the daily surfing behaviour of university students, setting up this research design was not trivial. Therefor we finish by elaborating on the technical, methodological and future analytical challenges for researchers who also want to conduct online monitoring studies.

2. LITERATURE SURVEY

S. NO	AUTHOR	PAPER	YEAR	DESCRIPTION
1.	Weal M.S Yafooz	Challenges and Issues on Online News Portal	2006	The online news will be viewed almost every second in order to follow the evolution of any desired global events. There are many organizations or political parties employ agents for tracking news by grouping the event. Therefore, news clustering is helpful and worthy for many researchers and online news readers in order to view events from multiple perspectives
2.	Morgan & Claypool	Incidentals Exposure to online News	2016	The prevalence of news on the Web provides opportunities for people to come across news in an incidental way as a byproduct of their online activities. It presents conception framework of IEON that advances research and an understanding news discovery
3.	Ali Al-Laith, Muhammad Shahbaz	Tracking sentiment towards news entities from Arabic news on social media	2021	he tracking sentiment of the news entities over time provides important information to governments and enterprises during the decision-making process. Recently, it has attracted the attention of the research community as well due to its popularity in many applications including; tracking news about elections, e-commerce, and egovernance.
4.	Marios Constantinides, John Dowell, David Johnson, Sylvain Malacria	Exploring mobile news reading interactions for news app Personalization	2022	As news is increasingly accessed on smartphones and tablets, the need for personalising news app interactions is apparent. We report a series of three studies addressing key issues in the development of adaptive news app interfaces. We first surveyed users' news reading preferences and

				behaviours; analysis revealed three primary types of reader.
5.	Martijn Kleppe and Marco Otte	Analysing and understanding news consumption patterns by tracking online user behaviour with a multimodal research design	2017	The data collection, preprocessing, and pattern discovery takes more time Digital Scholarship in the Humanities
6.	Oscar Westlund	Mobile News a review and model of journalism in an age of mobile media	2012	The technological convergence of mobile "phones" and multimedia has been taking place since the 1990s, but it was not until the commercial birth of touchscreenenabled mobile devices, offered with flat-rate subscriptions for mobile internet, that widespread production and use of newsrelated content and services began to flourish. Accessing mobile news has gained traction in the everyday life of the public.
7.	Wei Guo and Bo Zhang	Research on Development Strategy of News App under the Background of Artificial Intelligence	2019	With the rapid development of the mobile. The market performance of the media Applications strong, and it has become increasingly prominent in the public opinion. consulting literature, market research, comparative research and other methods, aiming at exploring and studying the future development trend of media APP.
8.	Regonda Nagaraju, Mohammed Farhan Pasha, Mohammed Abdul Majeed, AdapaSujith	An Improved Method for MultiLingual News Feed Application	2019	People increasingly turn to the internet for daily news updates. A Multi-Lingual news feed application is aimed at developing a web-based application named multilingual news feed app. This Application deals with the user who wants to read news from the web application. User can select different countries in which a user is interested, the latest news will be fetched from the selected country

9.	Sagar Patel, Sanket Suthar,Sandip Patel, Nehal Patel and Arpita Patel Chandubhai S	Topic Detection and Tracking in News Articles		This paper for detecting and tracking topics from news articles. Topic detection and tracking are used in text mining process. From data which are unstructured in text mining we pluck out information which are previously unknown. The objective of this paper is to recognize tasks occurred in different news sources. We are going to use agglomerative clustering based on average linkage for detecting the topics, calculate the similarity of topics using cosine similarity and KNN classifier for tracking the topics
10.	Wenpeng Tao, Shuangshuang Zhang,	An Approach to event mining based on massive online news	2018	This paper proposes an approach based on Single-Pass Clustering Algorithm with the characteristic of news comment to extract events from massive online news. Meanwhile we adopt a custom method in similarity calculation in this paper. According to experimental results, the approach has a good performance in event mining.

2.1 EXISTING PROBLEM

The app should ask the user initially, what categories are they interested to read from. And, show 8 out of every 10 news related to that category only (assuming that users get overwhelmed by too much info)Giving filters for notifications (in terms of content category and also frequency) because assuming that users get irritated by too many filters.An App that includes all international and regional news that can be customizable depending on the users' needs, will reduce the number of apps (assuming that people use more than 1 app).The app should provide info about all the trusted worldwide sources, and then in each article, it should mention which source has validated this news, as I'm assuming that users can't differentiate between real and fake news.An app allowing the user to choose/customize the time for notification popup for news (assuming that people only check news notifications during free time).Users don't want to spend time reading the entire content. They need short and crisp news.Too much information on social media can quickly cross users cognitive limits in processing news and can

make them feel overwhelmed and overloaded.As the frequency of news exposure increases,people gradually perceive news overload,which can lead them to shut down cognitively and deny the necessity of news consumption or to put less effort into acquiring news.Older adults are more likely to rely on television,radio,and print media for their news than are those in the youngest adult cohort,who are more likely to use mobile devices.

2.2 REFERENCES

[1] Sangeeta Ruth, Srividhya Raghavan V, Smrithi J, Saira Banu. 2016. "Spatial Preference Newsfeed

System For Android Mobile Users", IJCSITS, Vol-6, NO. 3: 24.

[2] https://newsapi.org/

[3] https://dzone.com/articles/how–to–parse–json–data–from-a-rest-api-using-simpl

[4] https://material.io

[5] https://developer.android.com/guide

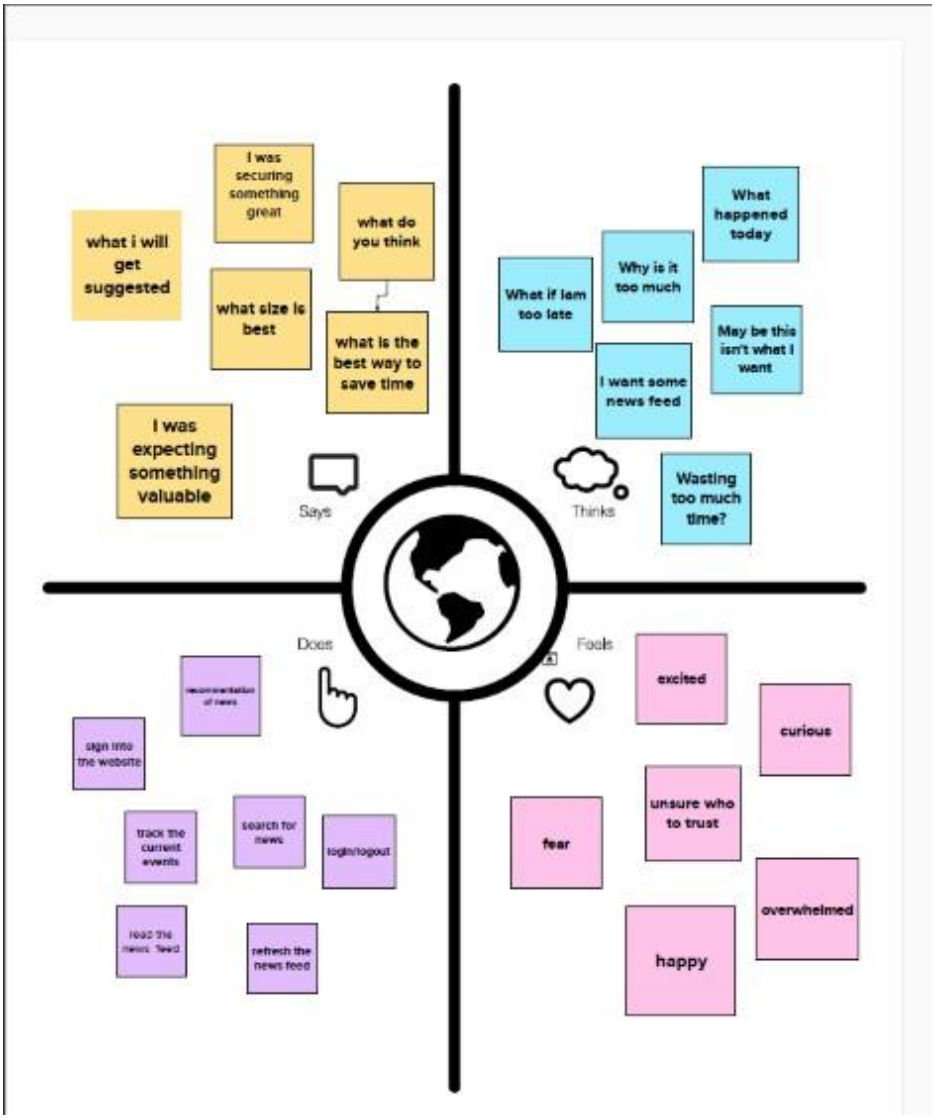
2.3 PROBLEM STATEMENT DEFINITION

There are multiple news-sharing apps used by a single user and are often spammed with notifications. There is also numerous fake news which gets shared. A news-sharing app wants to help users find relevant and important news easily every day and also understand explicitly that the news is not fake but from proper sources.A news sharing app wants to help users find relevant and important news easily everyday and also understand explicitly that the news is not fake but from proper sources.

NEWS TRACKER

3. IDEATION &PROPOSED SOLUTION

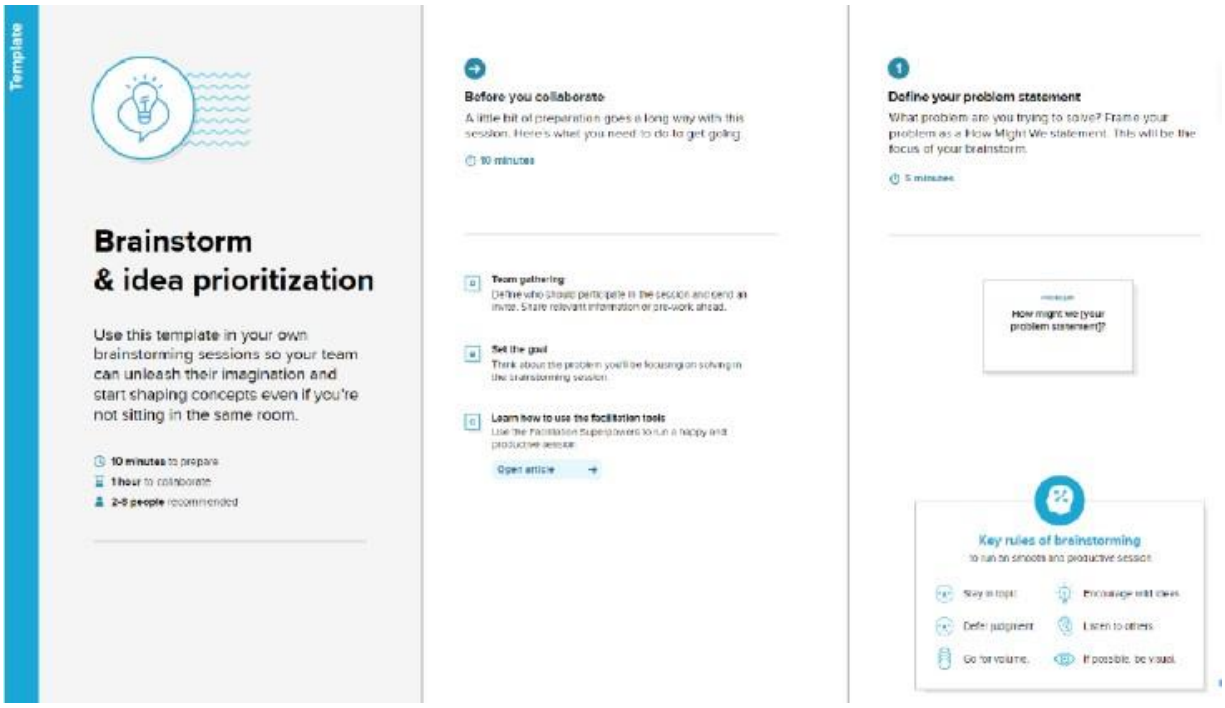
3.1 EMPATHY MAP CANVAS



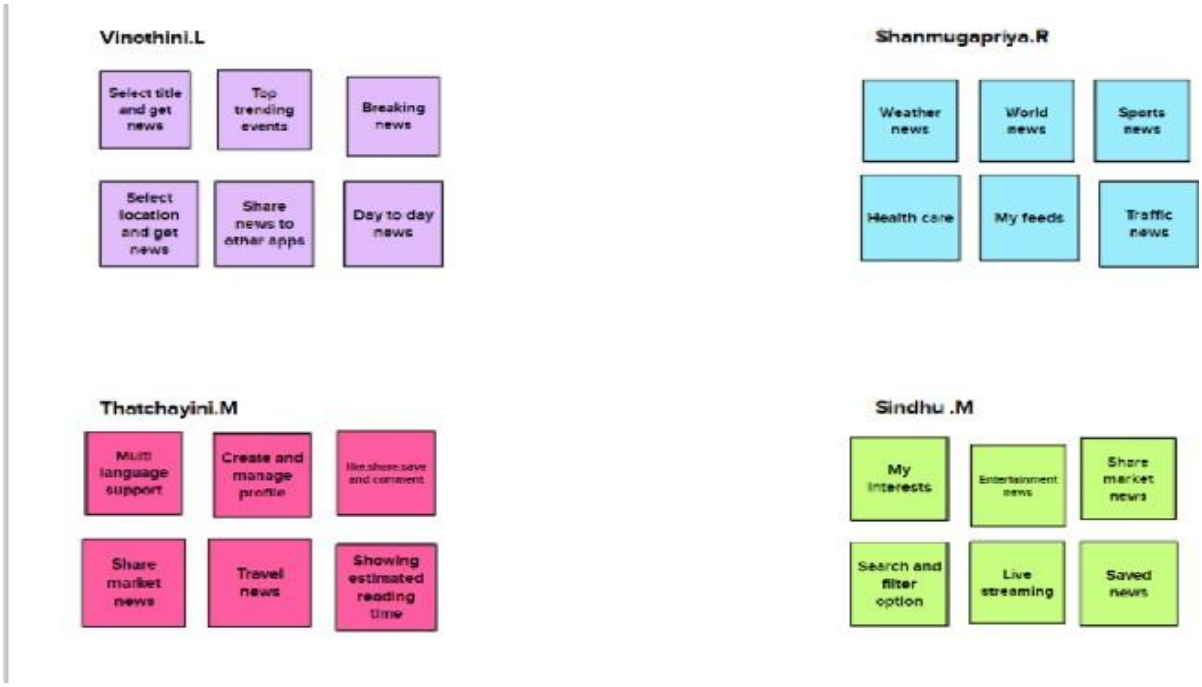
NEWS TRACKER

3.2 IDEATION & BRAINSTORMING

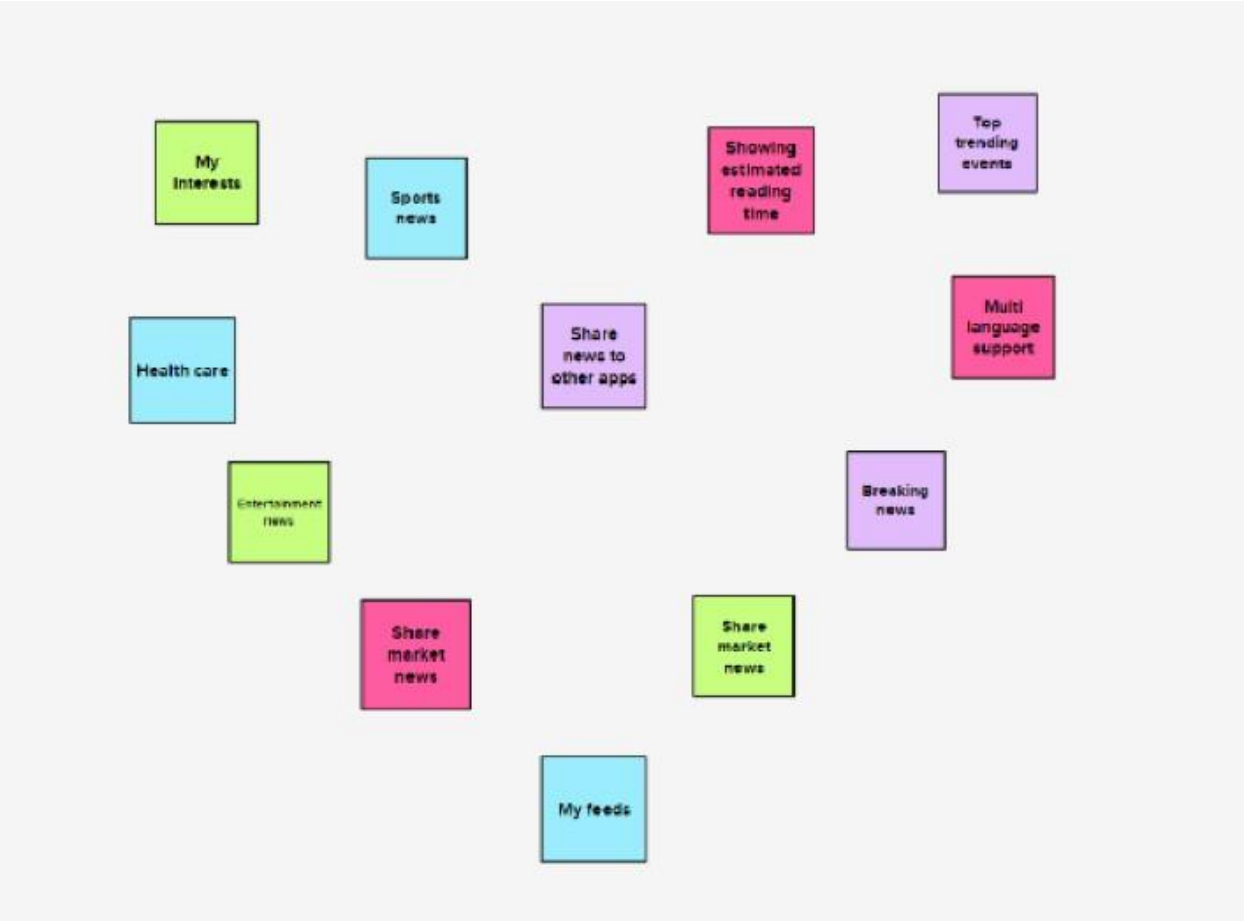
Step-1:Team gathering,collaboration and select the problem statement



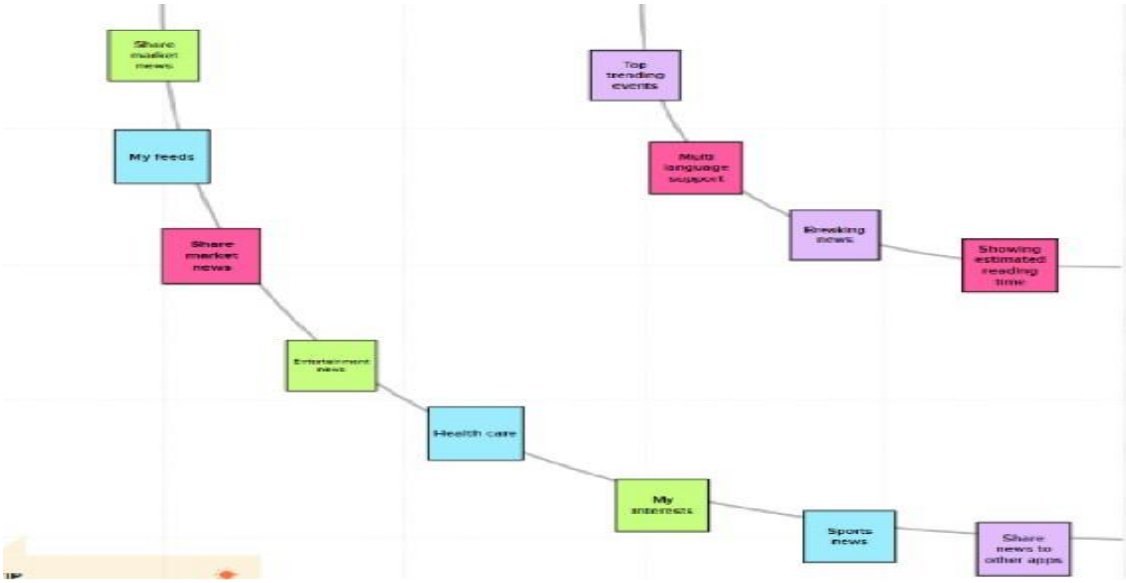
step-2: Brainstorm,Idea listing and grouping



NEWS TRACKER



step-3: Idea Prioritization



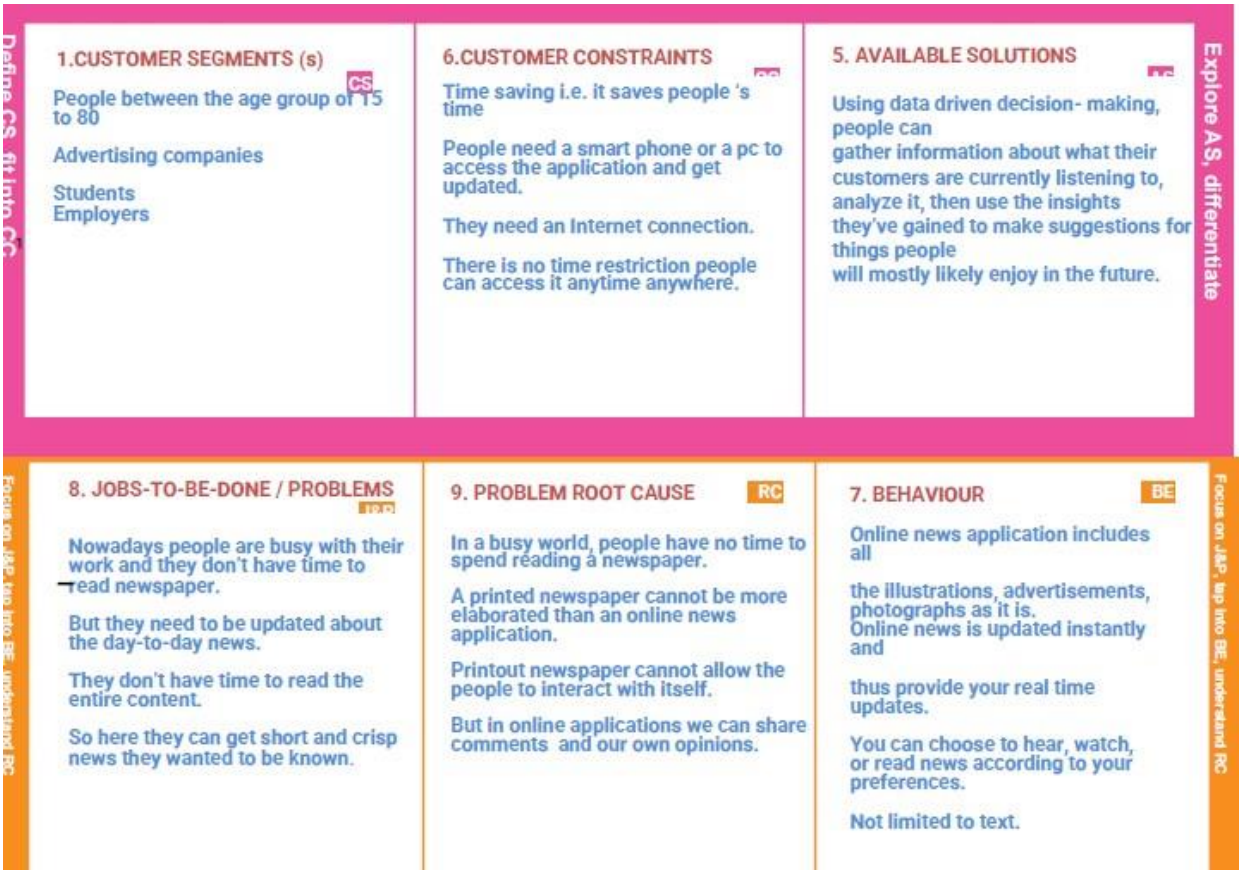
NEWS TRACKER

3.3 PROPOSED SOLUTION

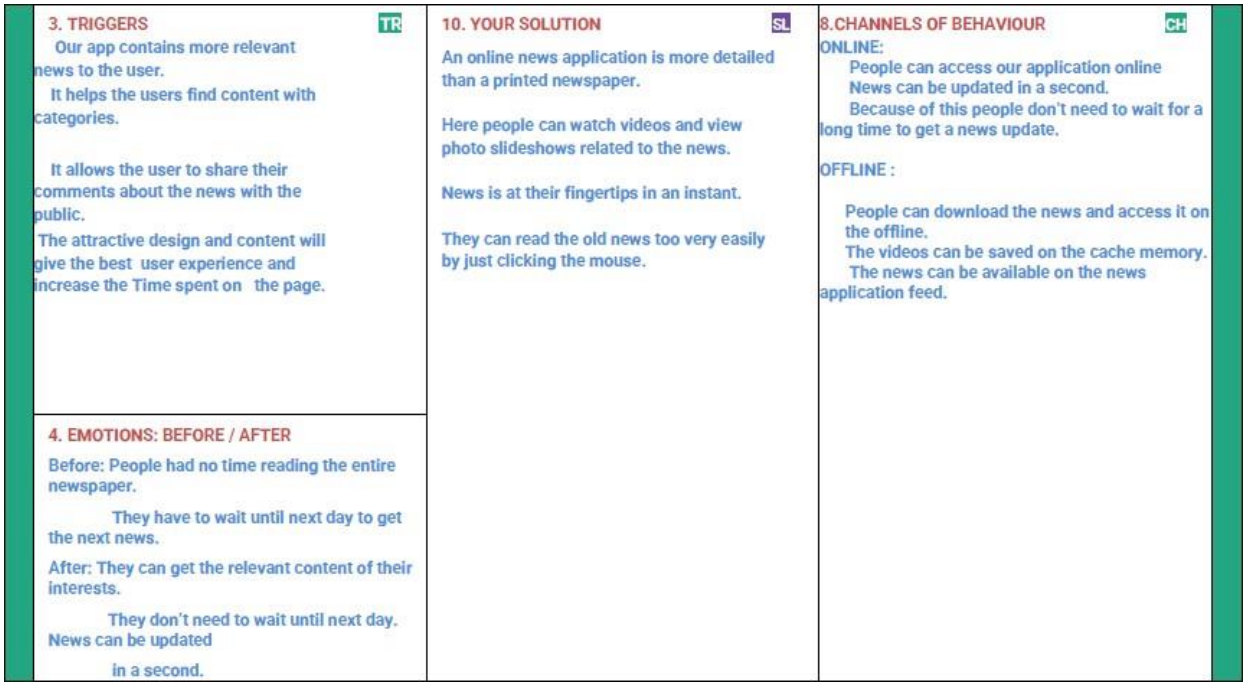
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	There are multiple news-sharing apps used by a single user and are often spammed with notifications which cause users to miss important events across the Globe. There is lot of fake news which gets shared. A news-sharing app wants to help users find relevant and important news easily every day and also understand explicitly that the news is not fake but from proper sources.
2.	Idea / Solution description	Since the user gets numerous content (also in terms of notifications) this solves that problem by allowing the user too narrow down the topics he is interested in. This way the app does not bombard the user with unnecessary information. This is also more likely to retain the user onto the app as the user will experience lesser negative emotions like feeling overwhelmed and agitated.
3.	Novelty / Uniqueness	Large volumes of news are published each day, but some of the themes are duplicated and do not possess novelty. In order to detect novelty of themes in news, introduced news to the annotators who judged the story based on novelty.
4.	Social Impact / Customer Satisfaction	Most of the internet content is filled with sexually explicit content which may affect the students mental behaviour. 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 may increase the feeling of news overload. Providing credible links for the user to refer to in case of any doubts on news credibility.
5.	Business Model (Revenue Model)	Promoting the advertisements for all quality products, one of our most important task is to provide the public with quality information.
6.	Scalability of the Solution	We created a scalable, responsive and user friendly newsfeed application for an audience across the globe including the ones that are not too tech-savvy.

3.4 PROBLEM SOLUTION FIT



NEWS TRACKER



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through FormRegistration through Gmail Registration through phone Number
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User interests	People can read news based on their interestsNews feed can show news based on their frequently searched news.
		It is user friendly

FR-4	User access	Any user can access easily with their mobile phones. They can also download the news andaccess it on offline.
FR-5	User authentication	It has a good security authentication. Where user's data cannot be stolen. User's data can be safe and secured.
FR-6	Location access	User can access it from anywhere at any time. News can be updated with the original location sources So, it can be easy to identify and witness ifit real or not.

4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Any user can use this application easily because ofits simple interface which can be easily understandable Any type of people such as kids, specially abled,aged people can access it easily.
NFR-2	**Security**	It has a high security. It comes with strong passwords and

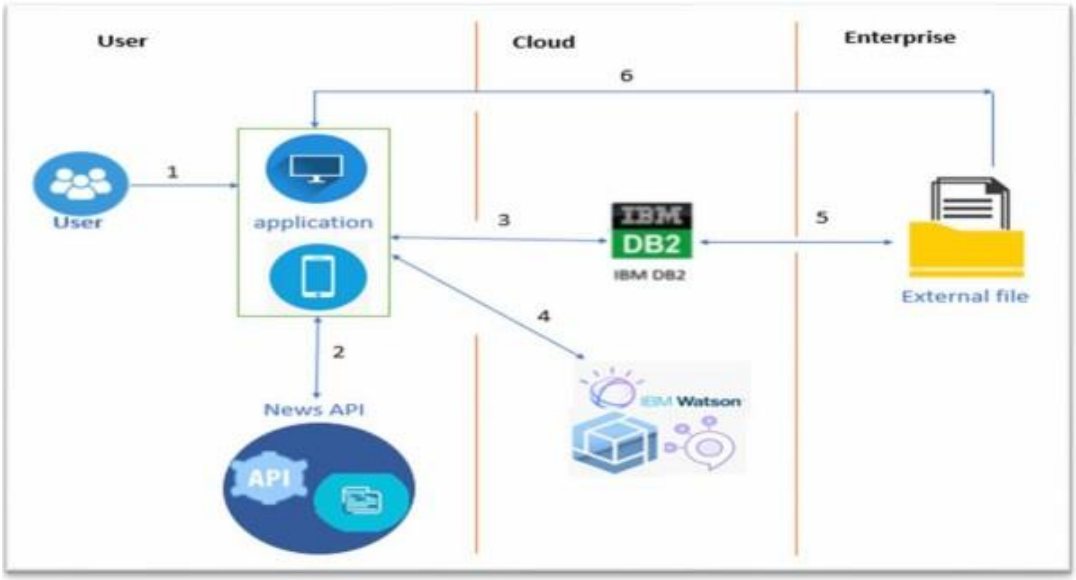
NEWS TRACKER

		authentication which is highly secured.
NFR-3	**Reliability**	News can be real and fake news can be automatically deleted if it has proven to be fake. News can be got from reliable sources.
NFR-4	**Performance**	News can be updated every second. User can share their comments publicly. User can be attracted to the better UI design andgets engaged with the page.
NFR-5	**Availability**	User can access at anytime All categories of news can be available. User can search their category and read.
NFR-6	**Scalability**	High capacity to handle growth. It can handle more users at a time.It can also satisfy the user’s needs.

NEWS TRACKER

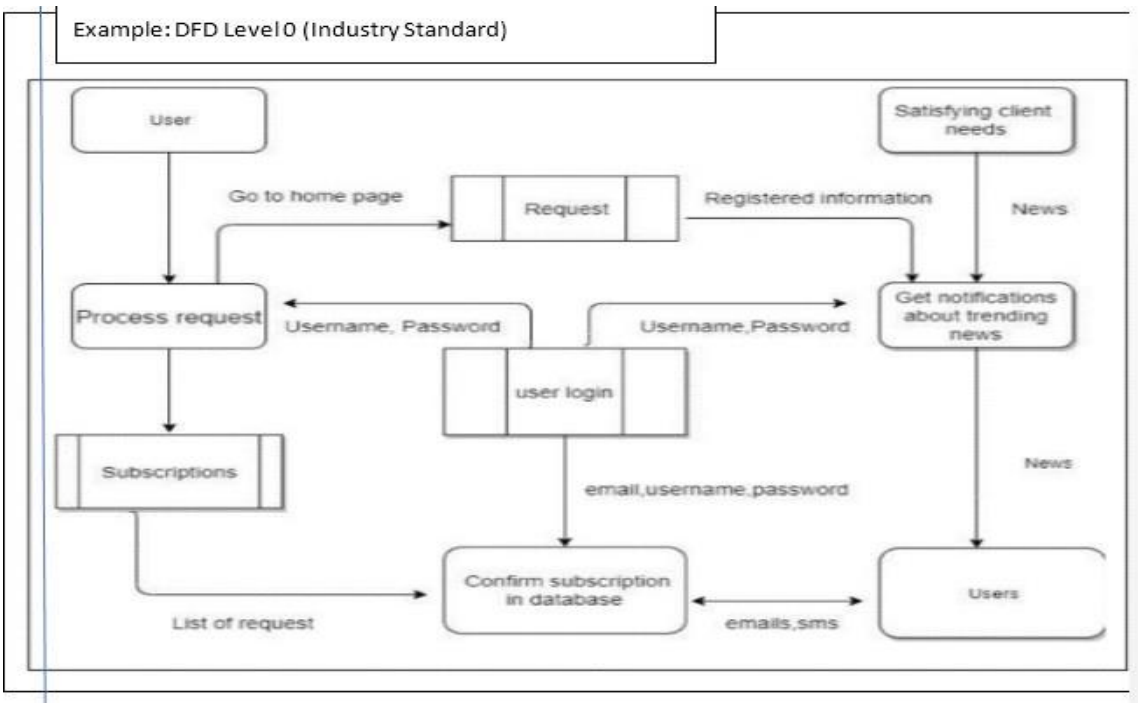
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



1. *User log in to the application*
2. *Validate user details with the records in the IBM DB*
3. *News API gives the headlines after login*
4. *User click the headlines to get the whole news*
5. *User can download the news to read it in offline mode 6. Watson assistant is used to clear doubts if needed*

NEWS TRACKER



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria		Release
Customer (Searching news)	Registration	USN-1	As a user, I can register for the application by entering my email, password, andconfirming my password.	I can access me account / dashboard	High	Sprint1
		USN-2	As a user, I will receive confirmation email once I	I can receive confirmation email and click confirm	High	Sprint1

NEWS TRACKER

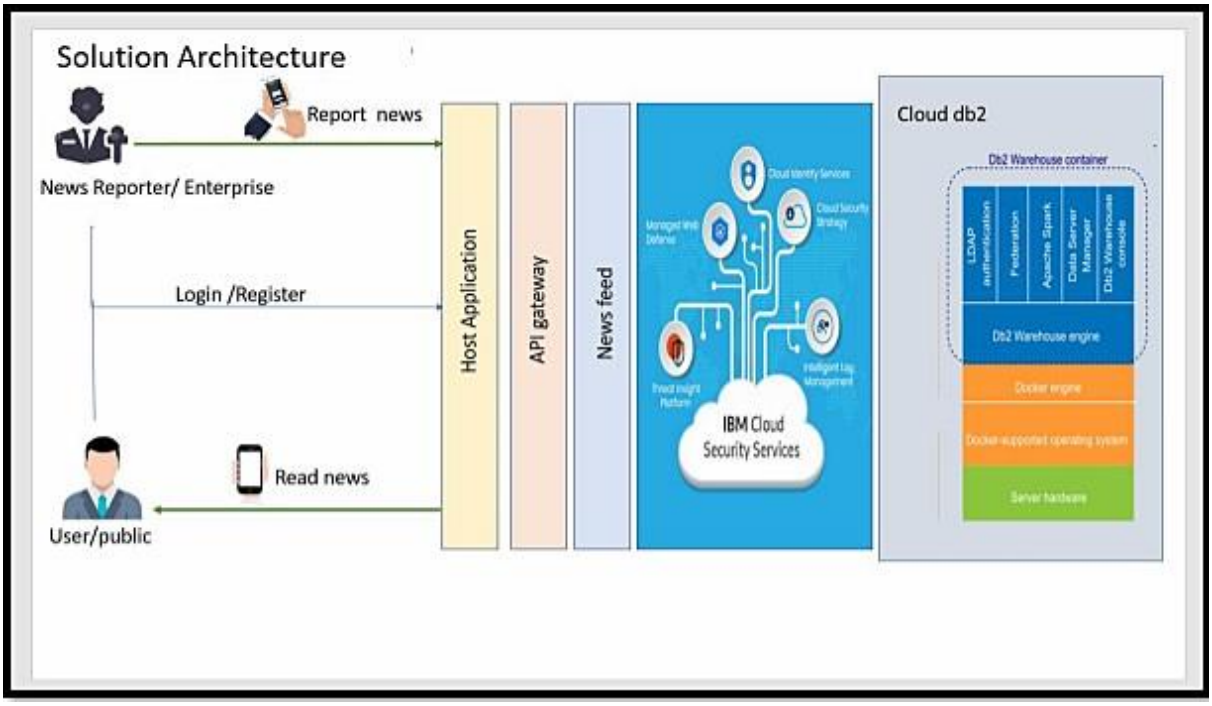
			have registered for the application			
		USN-3	As a user, I can register for the application through their given website	I can register and access the dashboard with Gmail or in Browser Login	Low	Sprint2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint1
	Login	USN-5	As a user, I can log into the application by entering email and password	I can view all types of information through this application	High	Sprint1
	Dashboard	USN-6	To see their histories about recently viewed, updates for search related news, current progress, feedback			
Customer (Web user)	Browser	USN-7	Have interactive	I have a clarity to use this	high	Sprint-1

			medium between client and server	application and easily resolve my specific issues		

5.2 SOLUTION & TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

1. Find the best tech solution to solve existing business problems.
2. Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
3. Define features, development phases, and solution requirements.
4. Provide specifications according to which the solution is defined, managed, anddelivered.

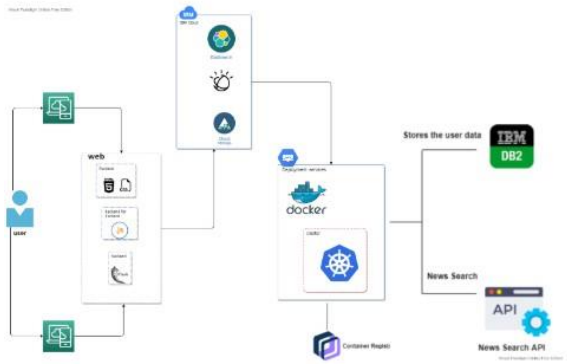


NEWS TRACKER

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/



Guidelines:

- Include all the processes (As an application logic / Technology Block)
- Provide infrastructural demarcation (Local / Cloud)
- Indicate external interfaces (third party API's etc.)
- Indicate Data Storage components / services
- Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	User interact with Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
2.	Application Logic-1	create a flask application which connects ibm db2	Python flask
3.	Application Logic-2	convert voice into text using IBM Watson STT search in the search engine using elastic search, upload images or files to object storage.	IBM Watson STT service ,object storage,elastic search
4.	Application Logic-3	Ask queries using watson assistant	IBM Watson Assistant
5.	Database	Data Type and Configurations.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2,

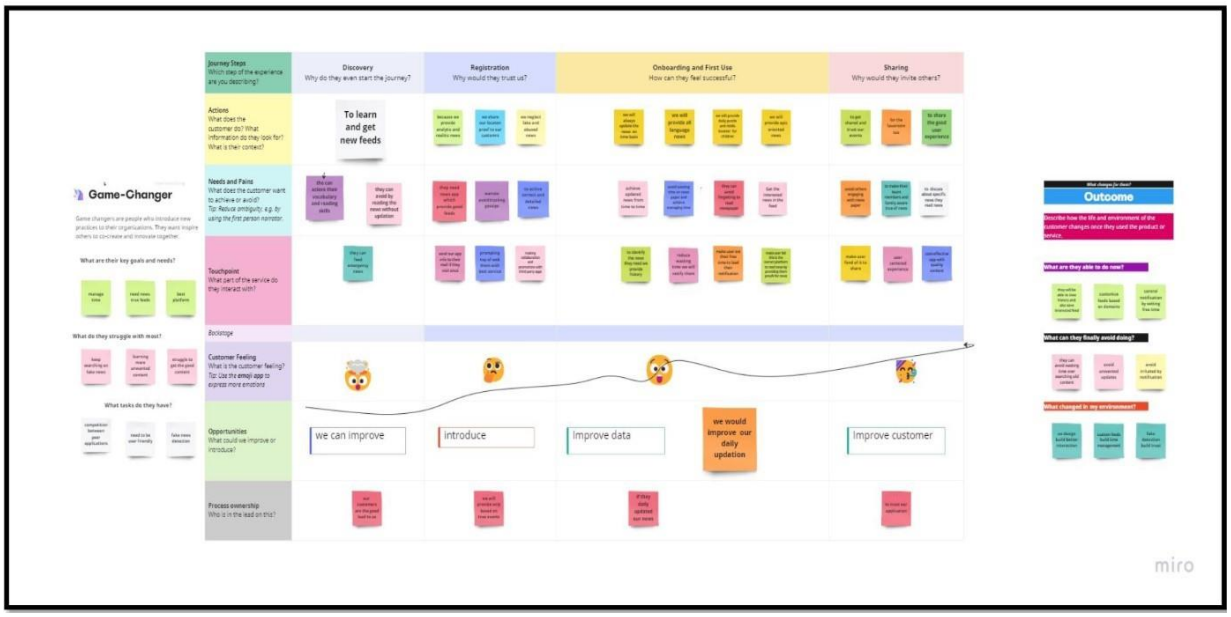
7.	File Storage	News feeds , video, audio,image	IBM Object Storage
8.	External API-1	secure ,socialize,manage and monetize, helping power digital transformation on premises and across the cloud.	IBM News API, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Cloud Server Configuration : SLA	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	A software for which the original source code is made freely available and may be redistributed and modified according to the requirement of the user.	Python
2.	Security Implementations	Cloud Security Posture Management(CSPM), Detect cloud security and compliance configuration risk, anomalous activity, vulnerabilities, and misconfigurations.	Built-in encryption, BYOK
3.	Scalable Architecture	Python is one of the pioneers of programing languages that developers can use to do all the scaling work. To improve scalability, you can enable or disable services run by the dispatcher on individual servers to balance the load for a given computer by request type.	Technology used in the architecture is that with the Python and the IBM cloud.
		Availability is the ability of a system to	Technology

4.	Availability	withstand or recover from exceptional situations, such as a computer failure. IBM Cloud is on-demand access, via the internet, to computing resources applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more hosted at a remote data centre managed by a cloud servicesprovider (or CSP).	used are the IBM cloud and the database.
5.	Performance	The updation of trending news occurs without any interruption. So, it performance is good.	Container Registry, Kubernetes Cluster.

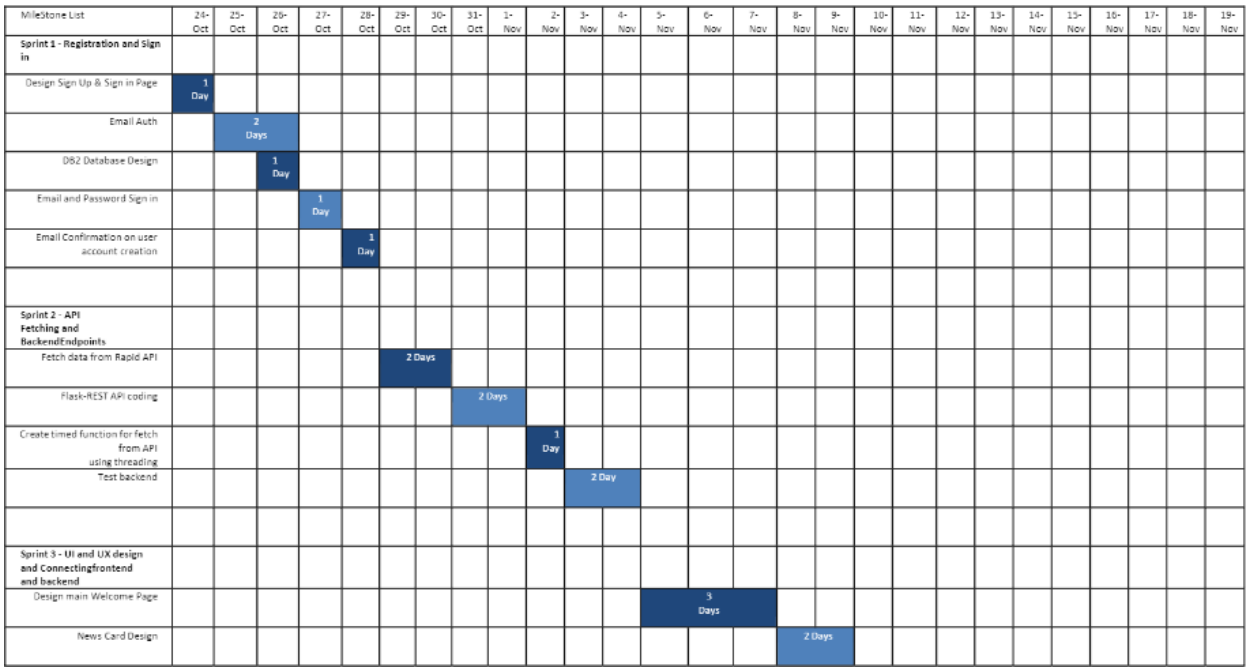
5.3 USER STORIES

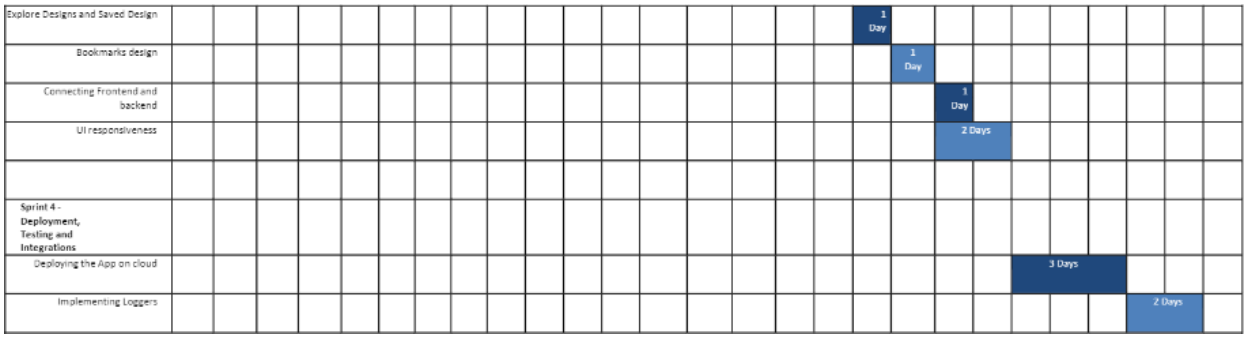


NEWS TRACKER

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION





6.2 SPRINT DELIVERY AND SCHEDULE

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	Priyadharshinik,Archana S,Deepika K,deepika M

NEWS TRACKER

NEWS TRACKER

Sprint 1	Database Connectivity	USN -2	To store details of the Customer connecting UI With database	10	High	Priyadharshini K,Archana S,Deepika K, Deepika M

NEWS TRACKER

Sp rint -2	News Tracker UI	USN- 3	Building UI News Tracker Application	10	High	Priyadharshini K,ArchanaS ,Deepika K, Deepika M
Sp rint -2	API	USN- 4		10	High	Priyadharshini K,ArchanaS ,Deepika K, Deepika M

NEWS TRACKER

CODING AND SOLUTIONS

sprint 1

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page Title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="style.css">
</head><html>
  <head>
  <meta name="viewport" content="width=device-width, initial-scale=1">

<body>

  <div class="navbar">
    <a href="#">HOME</a>
    <a href="register.html">REGISTER</a>
    <a href="login.html">LOGIN</a>
    <a href="#" class="right">CHATBOT</a>
  </div>

<div class="header">
  <div class="bg-image"></div><div class="bg-text">
    <img src="img.jpeg" alt="Avatar" class="avatar">
    <p><marquee><b><i>More News! More Often! Move Closer To Tour
World!</i></b></marquee></p>
</div>


  <div class="row">
    <div class="column nature">
      <div class="content">
        <img src="breaking news.jpeg" alt="breaking news" style="width:50%">
        <h4>BREAKING NEWS</h4>
      </div>
```

NEWS TRACKER

```
</div>
<div class="column nature">
  <div class="content">
  <img src="COMMERCIAL NEWS.jpg" alt="Lights" style="width:30%">
    <h4>COMMERCIAL NEWS</h4>
  </div>
</div>
<div class="column nature">
  <div class="content">
  <img src="ent.jpg" alt="Nature" style="width:100%">
    <h4>ENTERTAINMENT NEWS</h4>
    <p></p>
  </div>
</div>

<div class="column cars">
  <div class="content">
    <img src="HN.jpg" alt="Car" style="width:60%">
    <h4>HISTORICAL NEWS</h4>
  </div>
</div>
<div class="column cars">
  <div class="content">
  <img src="inews.jpg" alt="Car" style="width:80%">
    <h4>INTERNATIONAL NEWS</h4>
  </div>
</div>
<div class="column cars">
  <div class="content">
  <img src="lcn.png" alt="Car" style="width:100%">
    <h4>LOCAL NEWS</h4>
  </div>
</div>

<div class="column people">
  <div class="content">
    <img src="spn.jpeg" alt="Car" style="width:100%">
```

NEWS TRACKER

```
      <h4>SPORTS NEWS</h4>
    </div>
  </div>
  <div class="column people">
    <div class="content">
    <img src="wea.jpeg" alt="Car" style="width:80%">
      <h4>WEATHER NEWS</h>
    </div>
  </div>
  <div class="column people">
    <div class="content">
    <img src="bn.png" alt="Car" style="width:50%">
      <h4>BUSINESS NEWS</h4>
    </div>
  </div>
 </div>

 </div>

 <script src="script grid.js"></script>
 <br></div>
<div class="footer">
  <h2>VNEWS</h2>
  <ul>About us</ul>
  <ul>Resouces</ul>
  <ul>Get started</ul>
  <ul>help</ul>
</div>

</body>
</html>
```

login.html

```
    <!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
```

NEWS TRACKER

```
body {font-family: Arial, Helvetica, sans-serif;}

/* Full-width input fields */ input[type=text],
input[type=password] {width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

/* Set a style for all buttons */
button {
  background-color: #43255e;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none; cursor:
  pointer; width: 100%;
}

button:hover {
  opacity: 0.8;
}

/* Extra styles for the cancel button */
.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color:  #f44336;
}

/* Center the image and position the close button */
.imgcontainer {
  text-align: center;
```

```
  margin: 24px 0 12px 0;
  position: relative;
}

img.avatar {
  width: 10%;
  border-radius: 20%;
}

.container {
  padding: 16px;
}

span.psw { float:
  right;
  padding-top: 16px;
}

/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */left:
  0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
  overflow: auto; /* Enable scroll if needed */
  background-color: rgb(0,0,0); /* Fallback color */
  background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
  padding-top: 60px;
}

/* Modal Content/Box */
.modal-content { background-
  color: #fefefe;
  margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
```

```
  border: 1px solid #888;
  width: 80%; /* Could be more or less, depending on screen size */
}

/* The Close Button (x) */
.close {
  position: absolute;
  right: 25px;
  top: 0;
  color: #000;
  font-size: 35px;
  font-weight: bold;
}

.close:hover,
.close:focus {
  color: red;
  cursor: pointer;
}

/* Add Zoom Animation */
.animate {
  -webkit-animation: animatezoom 0.6s;animation:
  animatezoom 0.6s
}

@-webkit-keyframes animatezoom {
  from {-webkit-transform: scale(0)} to {-
  webkit-transform: scale(1)}
}

@keyframes animatezoom {
  from {transform: scale(0)} to
  {transform: scale(1)}
}

/* Change styles for span and cancel button on extra small screens */
```

NEWS TRACKER

```
@media screen and (max-width: 300px) {
  span.psw {
    display: block;float:
    none;
  }
  .cancelbtn {
    width: 100%;
  }
}
</style>
</head>
<body>

  <form class="modal-content animate" action="/action_page.php" method="post">
    <div class="imgcontainer">
      <span onclick="document.getElementById('id01').style.display='none'" class="close"
title="Close Modal">&times;</span>
      <img src="img.jpeg" alt="Avatar" class="avatar">
    </div>

    <div class="container">
      <label for="uname"><b>Username</b></label>
      <input type="text" placeholder="Enter Username" name="uname" required>

      <label for="psw"><b>Password</b></label>
      <input type="password" placeholder="Enter Password" name="psw" required>

      <button type="submit">Login</button>
      <label>
        <input type="checkbox" checked="checked" name="remember"> Remember me
      </label>
    </div>

    <div class="container" style="background-color:#f1f1f1">
      <span class="psw">Forgot <a href="#">password?</a></span>
    </div>
  </form>
```

NEWS TRACKER

```
<script>
var modal = document.getElementById('id01');
window.onclick = function(event) {
    if (event.target == modal) { modal.style.display
        = "none";
    }
}
</script>

</body>
</html>
register.html
   <!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
  font-family: Arial, Helvetica, sans-serif;
  background-color: black;
}

* {
  box-sizing: border-box;
}

/* Add padding to containers */
.container {
  padding: 16px;
  background-color: white;
}

/* Full-width input fields */ input[type=text],
input[type=password] {width: 100%;
```

```
  padding: 15px; margin:
  5px 0 22px 0;display:
  inline-block; border:
  none; background:
  #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

/* Overwrite default styles of hr */hr
{
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Set a style for the submit button */
.registerbtn {
  background-color: #43255e;
  color: white;
  padding: 16px 20px;
  margin: 8px 0;
  border: none; cursor:
  pointer; width: 100%;
  opacity: 0.9;
}

.registerbtn:hover {
  opacity: 1;
}

/* Add a blue text color to links */a {
  color: dodgerblue;
```

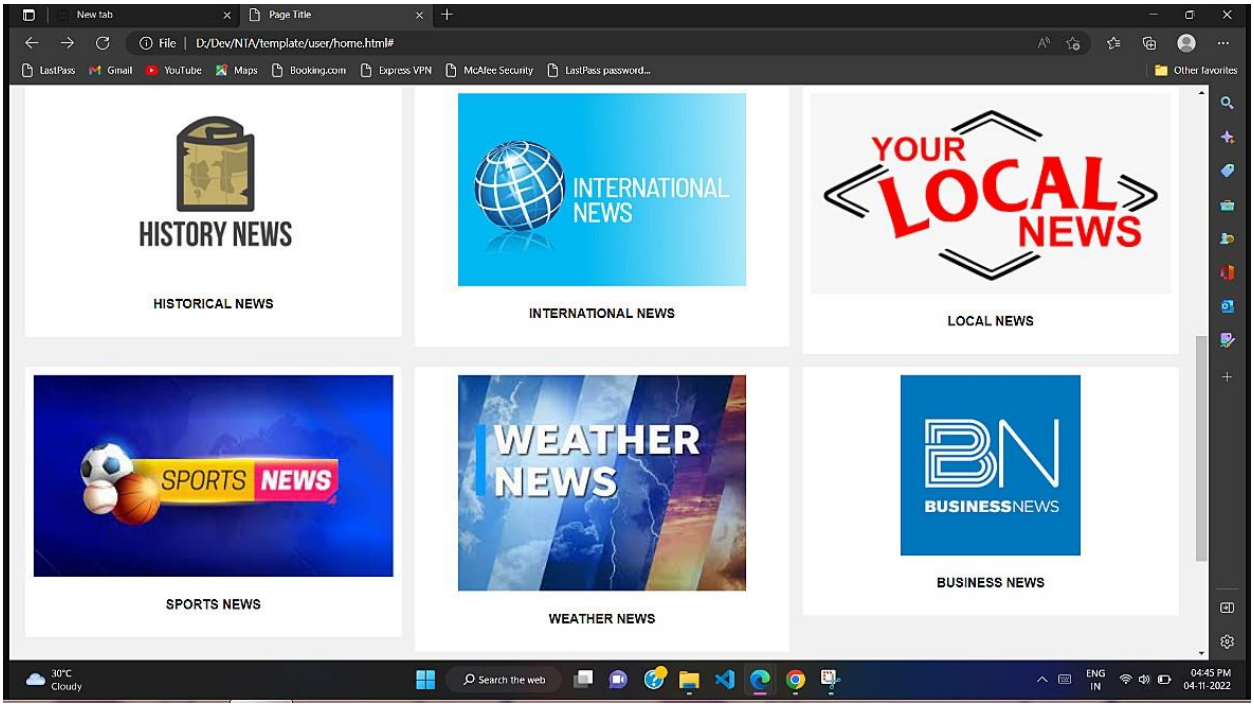
```
}

/* Set a grey background color and center the text of the "sign in" section */
.signin {
  background-color: #f1f1f1;
  text-align: center;
}
</style>
</head>
<body>

<form action="/action_page.php">
  <div class="container">
    <h1>Register</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email" id="email" required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" id="psw"required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
    <input type="password" placeholder="Repeat Password" name="psw-repeat" id="psw-repeat"
required>
    <hr>
    <p>By creating an account you agree to our <a href="#">Terms & Privacy</a>.</p>

    <button type="submit" class="registerbtn">Register</button>
  </div>

  <div class="container signin">
    <p>Already have an account? <a href="login.html">Sign in</a>.</p>
  </div>
</form>
```

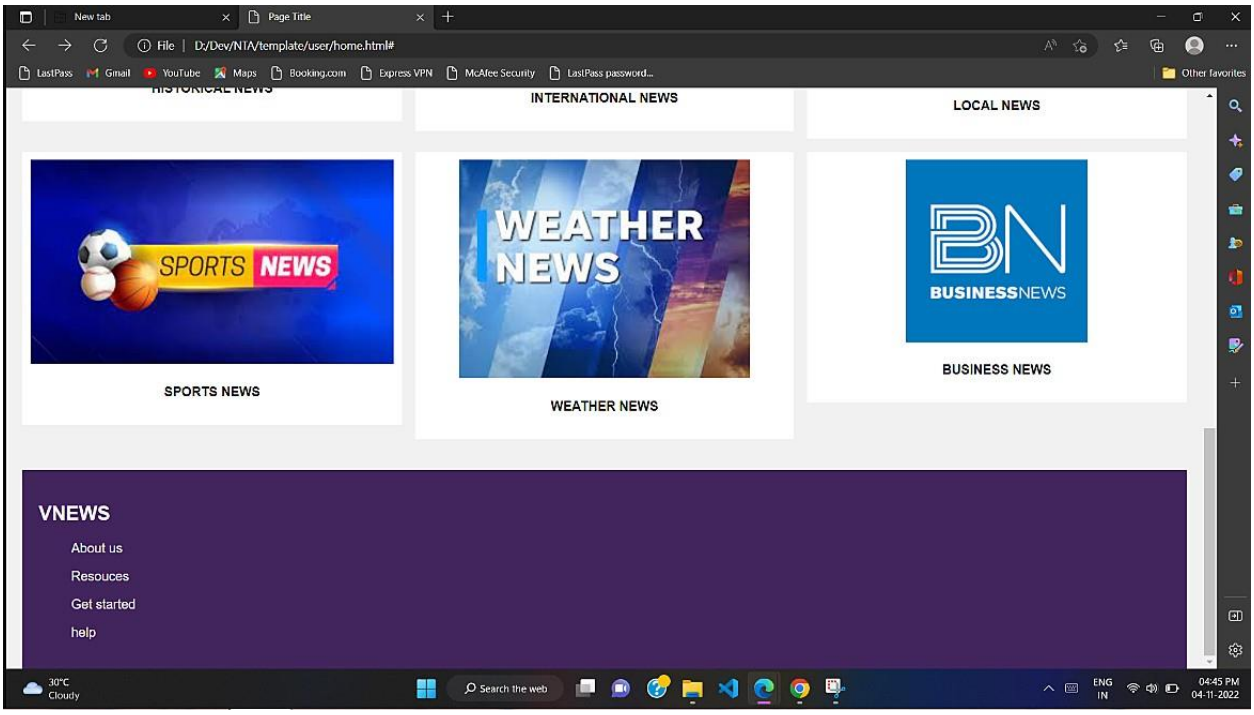
NEWS TRACKER

```
</body>
</html> script
grid.js

filterSelection("all") function
filterSelection(c) {var x, i;
  x =  document.getElementsByClassName("column");if (c
  == "all") c = "";
  for (i = 0; i < x.length; i++) {
    w3RemoveClass(x[i], "show");
    if (x[i].className.indexOf(c) > -1) w3AddClass(x[i], "show");
  }
}

function w3AddClass(element, name) {var
  i, arr1, arr2;
  arr1  =  element.className.split("  ");arr2
  = name.split(" ");
  for (i = 0; i < arr2.length; i++) {
    if (arr1.indexOf(arr2[i]) == -1) {element.className += " " + arr2[i];}
  }
}

function w3RemoveClass(element, name) {var
  i, arr1, arr2;
  arr1  =  element.className.split("  ");arr2
  = name.split(" ");
  for (i = 0; i < arr2.length; i++) {  while
    (arr1.indexOf(arr2[i]) >  -1)  {
      arr1.splice(arr1.indexOf(arr2[i]),   1);
    }
  }
  element.className = arr1.join(" ");
}
style.css
```

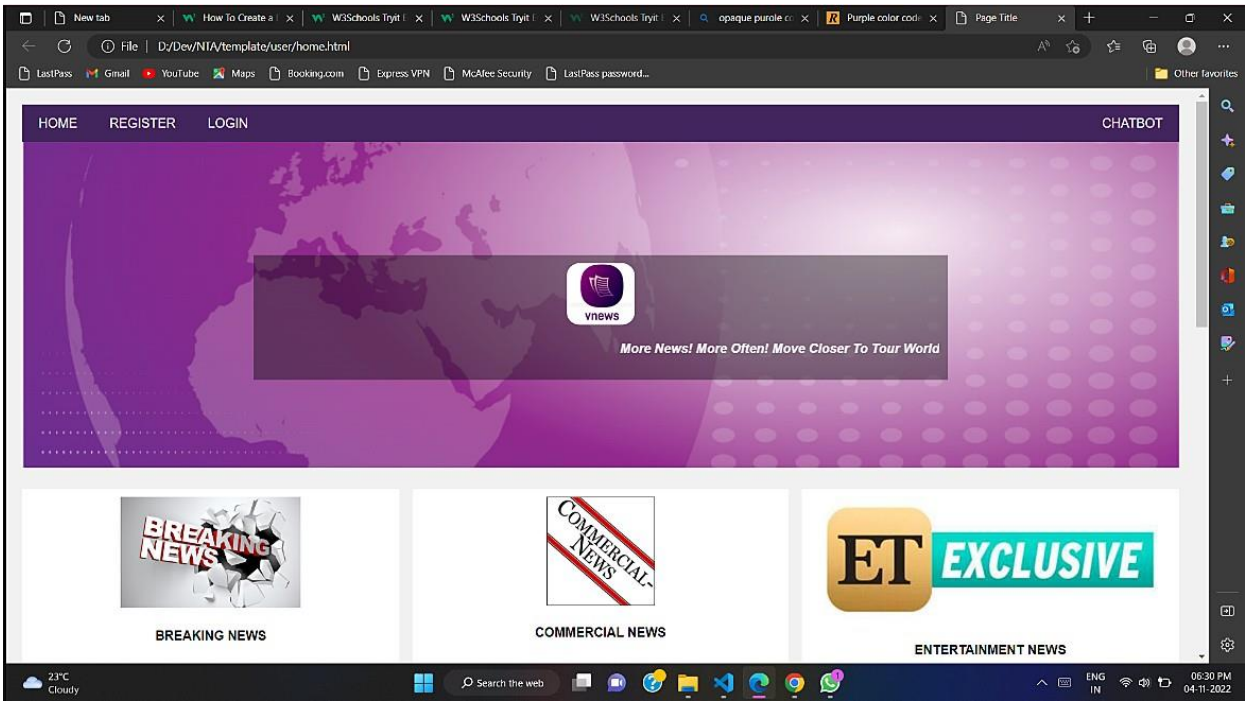
NEWS TRACKER

FIRST VIEW

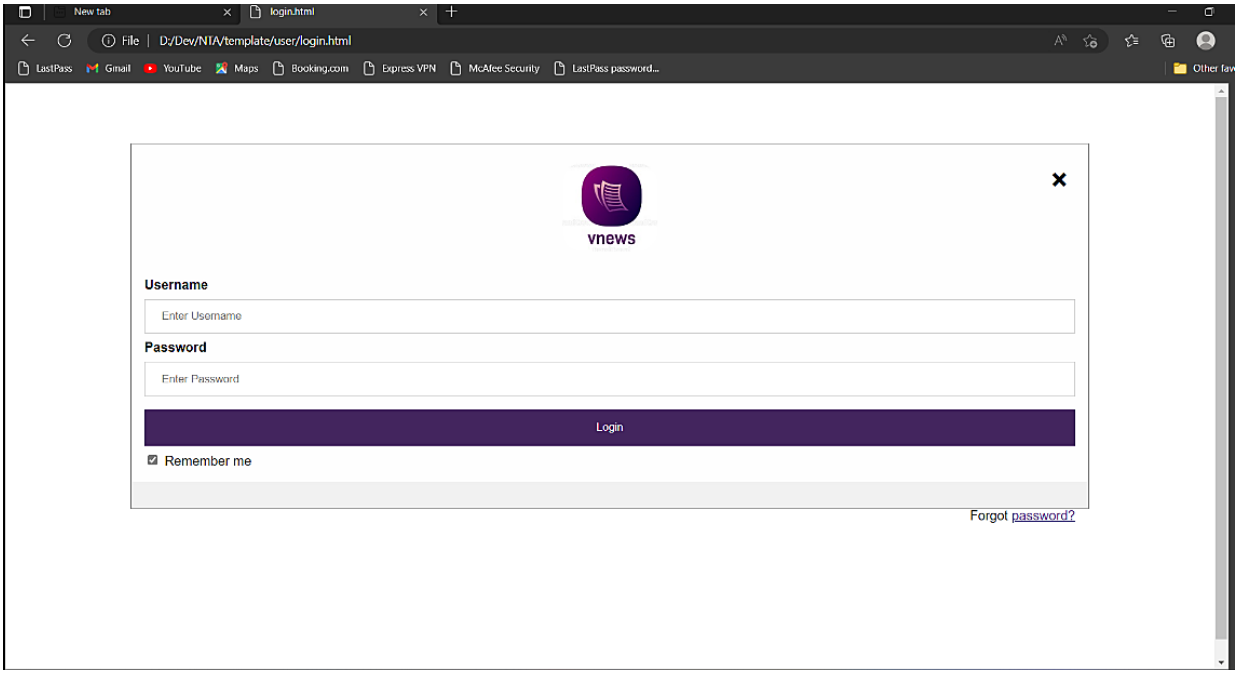




NEWS TRACKER

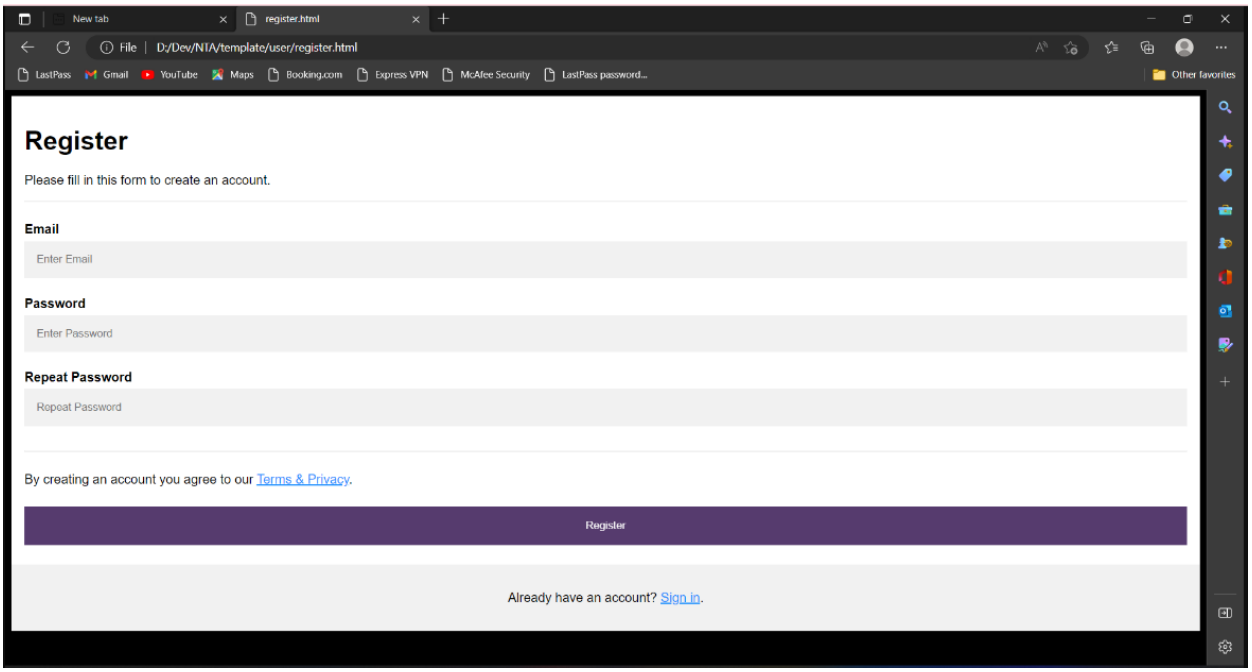


LOGIN



NEWS TRACKER

REGISTRATION



SPRINT 2

INTEGRATING NEWS API

```
from flask import
Flask,render_template
from newsapi import NewsApiClient

app = Flask(__name__)


@app.route('/')
def index():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2
    topheadlinesindia= newsapi.get_top_headlines(sources = "
headlines?country=in")
    articles =topheadlinesindia['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
```

```
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist1=zip(news,desc,img)
    topheadlinessouthkorea = newsapi.get_top_headlines(sourc
headlines?country=kr")
    articles = topheadlinessouthkorea['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist2=zip(news,desc,img)


    topheadlinesthailand= newsapi.get_top_headlines(sources
headlines?country=th")
    articles = topheadlinesthailand['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist3 =zip(news,desc,img)


    topheadlinesuk= newsapi.get_top_headlines(sources = "htt
headlines?country=gb")
    articles =topheadlinesuk['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
```

```
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist4 =zip(news,desc,img)


    topheadlinesukrane= newsapi.get_top_headlines(sources =
headlines?country=ua")
    articles =topheadlinesukrane['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist5 =zip(news,desc,img)


    topheadlinesrussia= newsapi.get_top_headlines(sources =
headlines?country=ru")
    articles =topheadlinesrussia['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist6 =zip(news,desc,img)


    topheadlinestiwan= newsapi.get_top_headlines(sources = "
headlines?country=tw")
    articles =topheadlinestiwan['articles']

    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
```

```
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist7 =zip(news,desc,img)
```

INTEGRATING SENDGRID API

```
import
os
import json

from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *


# NOTE: you will need move this file to the root
# directory of this project to execute properly.


def build_hello_email():
    ## Send a Single Email to a Single Recipient

    message = Mail(from_email=From('from@example.com', 'Example From Name'),
                   to_emails=To('to@example.com', 'Example To Name'),
                   subject=Subject('Sending with SendGrid is Fun'),
                   plain_text_content=PlainTextContent('and easy to do anywhere, even with
                   html_content=HtmlContent('<strong>and easy to do anywhere, even with Pyt

    try:
        print(json.dumps(message.get(), sort_keys=True, indent=4))
        return message.get()

    except SendGridException as e:
        print(e.message)

    mock_personalization = Personalization()
    personalization_dict = get_mock_personalization_dict()

    for cc_addr in personalization_dict['cc_list']:
        mock_personalization.add_to(cc_addr)

    for bcc_addr in personalization_dict['bcc_list']:
```

```
        mock_personalization.add_bcc(bcc_addr)

    for header in personalization_dict['headers']:
        mock_personalization.add_header(header)

    for substitution in personalization_dict['substitutions']:
        mock_personalization.add_substitution(substitution)

    for arg in personalization_dict['custom_args']:
        mock_personalization.add_custom_arg(arg)

    mock_personalization.subject = personalization_dict['subject']
    mock_personalization.send_at = personalization_dict['send_at']

    message.add_personalization(mock_personalization)

    return message

def get_mock_personalization_dict():
    """Get a dict of personalization mock."""
    mock_pers = dict()

    mock_pers['to_list'] = [To("test1@example.com",
                               "Example User"),
                            To("test2@example.com",
                               "Example User")]

    mock_pers['cc_list'] = [To("test3@example.com",
                               "Example User"),
                            To("test4@example.com",
                               "Example User")]

    mock_pers['bcc_list'] = [To("test5@example.com"),
                             To("test6@example.com")]

    mock_pers['subject'] = ("Hello World from the Personalized "
                            "SendGrid Python Library")

    mock_pers['headers'] = [Header("X-Test", "test"),
                            Header("X-Mock", "true")]

    mock_pers['substitutions'] = [Substitution("%name%", "Example User"),
                                  Substitution("%city%", "Denver")]
```

NEWS TRACKER

```
    mock_pers['custom_args'] = [CustomArg("user_id", "343"),
                                CustomArg("type", "marketing")]

    mock_pers['send_at'] = 1443636843
    return mock_pers

def build_multiple_emails_personalized():
    # Note that the domain for all From email addresses must match

    message = Mail(from_email=From('from@example.com', 'Example From Name'),
                subject=Subject('Sending with SendGrid is Fun'),
                plain_text_content=PlainTextContent('and easy to do anywhere, even with
                html_content=HtmlContent('<strong>and easy to do anywhere, even with Pyt

    mock_personalization = Personalization()
    mock_personalization.add_to(To('test@example.com', 'Example User 1'))
    mock_personalization.add_cc(Cc('test1@example.com', 'Example User 2'))
    message.add_personalization(mock_personalization)

    mock_personalization_2 = Personalization()
    mock_personalization_2.add_to(To('test2@example.com', 'Example User 3'))
    mock_personalization_2.set_from(From('from@example.com', 'Example From Name 2'))
    mock_personalization_2.add_bcc(Bcc('test3@example.com', 'Example User 4'))
    message.add_personalization(mock_personalization_2)

    try:
        print(json.dumps(message.get(), sort_keys=True, indent=4))
        return message.get()

    except SendGridException as e:
        print(e.message)

    return message

def build_attachment1():
    """Build attachment mock. Make sure your content is base64 encoded before passing in
    Another example: https://github.com/sendgrid/sendgrid-python/blob/HEAD/use_cases/att

    attachment = Attachment()
    attachment.file_content = ("TG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNvbnNl"
                           "Y3RldHVyIGFkaXBpc2NpbmcgZWxpdC4gQ3JhcyBwdW12")
    attachment.file_type = "application/pdf"
```

```
    attachment.file_name = "balance_001.pdf"
    attachment.disposition = "attachment"
    attachment.content_id = "Balance Sheet"
    return attachment


def build_attachment2():
    """Build attachment mock."""
    attachment = Attachment()
    attachment.file_content = "BwdW"
    attachment.file_type = "image/png"
    attachment.file_name = "banner.png"
    attachment.disposition = "inline"
    attachment.content_id = "Banner"
    return attachment

def build_kitchen_sink():
    """All settings set"""
    from sendgrid.helpers.mail import (
        Mail, From, To, Cc, Bcc, Subject, PlainTextContent,
        HtmlContent, SendGridException, Substitution,
        Header, CustomArg, SendAt, Content, MimeType, Attachment,
        FileName, FileContent, FileType, Disposition, ContentId,
        TemplateId, Section, ReplyTo, Category, BatchId, Asm,
        GroupId, GroupsToDisplay, IpPoolName, MailSettings,
        BccSettings, BccSettingsEmail, BypassListManagement,
        FooterSettings, FooterText, FooterHtml, SandBoxMode,
        SpamCheck, SpamThreshold, SpamUrl, TrackingSettings,
        ClickTracking, SubscriptionTracking, SubscriptionText,
        SubscriptionHtml, SubscriptionSubstitutionTag,
        OpenTracking, OpenTrackingSubstitutionTag, Ganalytics,
        UtmSource, UtmMedium, UtmTerm, UtmContent, UtmCampaign)
    import time
    import datetime

    message = Mail()

    # Define Personalizations

    message.to = To('test1@sendgrid.com', 'Example User1', p=0)
    message.to = [
        To('test2@sendgrid.com', 'Example User2', p=0),
        To('test3@sendgrid.com', 'Example User3', p=0)
```

```
]

message.cc = Cc('test4@example.com', 'Example User4', p=0)
message.cc = [
    Cc('test5@example.com', 'Example User5', p=0),
    Cc('test6@example.com', 'Example User6', p=0)
]

message.bcc = Bcc('test7@example.com', 'Example User7', p=0)
message.bcc = [
    Bcc('test8@example.com', 'Example User8', p=0),
    Bcc('test9@example.com', 'Example User9', p=0)
]

message.subject = Subject('Sending with SendGrid is Fun 0', p=0)

message.header = Header('X-Test1', 'Test1', p=0)
message.header = Header('X-Test2', 'Test2', p=0)
message.header = [
    Header('X-Test3', 'Test3', p=0),
    Header('X-Test4', 'Test4', p=0)
]

message.substitution = Substitution('%name1%', 'Example Name 1', p=0)
message.substitution = Substitution('%city1%', 'Example City 1', p=0)
message.substitution = [
    Substitution('%name2%', 'Example Name 2', p=0),
    Substitution('%city2%', 'Example City 2', p=0)
]

message.custom_arg = CustomArg('marketing1', 'true', p=0)
message.custom_arg = CustomArg('transactional1', 'false', p=0)
message.custom_arg = [
    CustomArg('marketing2', 'false', p=0),
    CustomArg('transactional2', 'true', p=0)
]

message.send_at = SendAt(1461775051, p=0)

message.to = To('test10@example.com', 'Example User10', p=1)
message.to = [
    To('test11@example.com', 'Example User11', p=1),
    To('test12@example.com', 'Example User12', p=1)
```

```
]

message.cc = Cc('test13@example.com', 'Example User13', p=1)
message.cc = [
    Cc('test14@example.com', 'Example User14', p=1),
    Cc('test15@example.com', 'Example User15', p=1)
]

message.bcc = Bcc('test16@example.com', 'Example User16', p=1)
message.bcc = [
    Bcc('test17@example.com', 'Example User17', p=1),
    Bcc('test18@example.com', 'Example User18', p=1)
]

message.header = Header('X-Test5', 'Test5', p=1)
message.header = Header('X-Test6', 'Test6', p=1)
message.header = [
    Header('X-Test7', 'Test7', p=1),
    Header('X-Test8', 'Test8', p=1)
]

message.substitution = Substitution('%name3%', 'Example Name 3', p=1)
message.substitution = Substitution('%city3%', 'Example City 3', p=1)
message.substitution = [
    Substitution('%name4%', 'Example Name 4', p=1),
    Substitution('%city4%', 'Example City 4', p=1)
]

message.custom_arg = CustomArg('marketing3', 'true', p=1)
message.custom_arg = CustomArg('transactional3', 'false', p=1)
message.custom_arg = [
    CustomArg('marketing4', 'false', p=1),
    CustomArg('transactional4', 'true', p=1)
]

message.send_at = SendAt(1461775052, p=1)

message.subject = Subject('Sending with SendGrid is Fun 1', p=1)

# The values below this comment are global to entire message

message.from_email = From('help@twilio.com', 'Twilio SendGrid')
```

NEWS TRACKER

```
message.reply_to = ReplyTo('help_reply@twilio.com', 'Twilio SendGrid Reply')

message.subject = Subject('Sending with SendGrid is Fun 2')

message.content = Content(MimeType.text, 'and easy to do anywhere, even with Python'
message.content = Content(MimeType.html, '<strong>and easy to do anywhere, even with
message.content = [
    Content('text/calendar', 'Party Time!!'),
    Content('text/custom', 'Party Time 2!!')
]

message.attachment = Attachment(FileContent('base64 encoded content 1'),
                                FileName('balance_001.pdf'),
                                FileType('application/pdf'),
                                Disposition('attachment'),
                                ContentId('Content ID 1'))
message.attachment = [
    Attachment(FileContent('base64 encoded content 2'),
            FileName('banner.png'),
            FileType('image/png'),
            Disposition('inline'),
            ContentId('Content ID 2')),
    Attachment(FileContent('base64 encoded content 3'),
            FileName('banner2.png'),
            FileType('image/png'),
            Disposition('inline'),
            ContentId('Content ID 3'))
]

message.template_id = TemplateId('13b8f94f-bcae-4ec6-b752-70d6cb59f932')

message.section = Section('%section1%', 'Substitution for Section 1 Tag')
message.section = [
    Section('%section2%', 'Substitution for Section 2 Tag'),
    Section('%section3%', 'Substitution for Section 3 Tag')
]

message.header = Header('X-Test9', 'Test9')
message.header = Header('X-Test10', 'Test10')
message.header = [
    Header('X-Test11', 'Test11'),
    Header('X-Test12', 'Test12')
]
```

NEWS TRACKER

```
message.category = Category('Category 1')
message.category = Category('Category 2')
message.category = [
    Category('Category 1'),
    Category('Category 2')
]

message.custom_arg = CustomArg('marketing5', 'false')
message.custom_arg = CustomArg('transactional5', 'true')
message.custom_arg = [
    CustomArg('marketing6', 'true'),
    CustomArg('transactional6', 'false')
]

message.send_at = SendAt(1461775053)

message.batch_id = BatchId("HkJ5yLYULb7Rj8GKSx7u025ouWVlMgAi")

message.asm = Asm(GroupId(1), GroupsToDisplay([1,2,3,4]))

message.ip_pool_name = IpPoolName("IP Pool Name")

mail_settings = MailSettings()
mail_settings.bcc_settings = BccSettings(False, BccSettingsTo("bcc@twilio.com"))
mail_settings.bypass_list_management = BypassListManagement(False)
mail_settings.footer_settings = FooterSettings(True, FooterText("w00t"), FooterHtml(
mail_settings.sandbox_mode = SandBoxMode(True)
mail_settings.spam_check = SpamCheck(True, SpamThreshold(5), SpamUrl("https://exampl
message.mail_settings = mail_settings

tracking_settings = TrackingSettings()
tracking_settings.click_tracking = ClickTracking(True, False)
tracking_settings.open_tracking = OpenTracking(True, OpenTrackingSubstitutionTag("op
tracking_settings.subscription_tracking = SubscriptionTracking(
    True,
    SubscriptionText("Goodbye"),
    SubscriptionHtml("<strong>Goodbye!</strong>"),
    SubscriptionSubstitutionTag("unsubscribe"))
tracking_settings.ganalytics = Ganalytics(
    True,
    UtmSource("utm_source"),
    UtmMedium("utm_medium"),
```

```
        UtmTerm("utm_term"),
        UtmContent("utm_content"),
        UtmCampaign("utm_campaign"))
    message.tracking_settings = tracking_settings

    return message

def send_multiple_emails_personalized():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-python/blob/HEAD/TROUBLESHOOTING.md#environme
sendgrid-api-key
    message = build_multiple_emails_personalized()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)

def send_hello_email():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-python/blob/HEAD/TROUBLESHOOTING.md#environme
sendgrid-api-key
    message = build_hello_email()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)


def send_kitchen_sink():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-python/blob/HEAD/TROUBLESHOOTING.md#environme
sendgrid-api-key
    message = build_kitchen_sink()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers
```

SPRINT 3

CREATING DASH BOARD

NEWS TRACKER

```
<!DOCTYPE
html>
                <html lang="en">
                <head>
                <title>home.html</title>
                <meta charset="UTF-8">
                <meta name="viewport" content="width=device-width, initial-scale=1">
                <link rel="stylesheet" href="style.css">
                </head><html>
                  <head>
                  <meta name="viewport" content="width=device-width, initial-scale=1">

                <body>

                  <div class="navbar">
                    <a href="#">HOME</a>
                    <a href="register.html">REGISTER</a>
                    <a href="login.html">LOGIN</a>
                    <a href="#" class="right">CHATBOT</a>
                  </div>

                <div class="header">
                  <div class="bg-image"></div><div class="bg-text">
                    <img src="img.jpeg" alt="Avatar" class="avatar">
                    <p><marquee><b><i>More News! More Often!  Move Closer To Tour
                World!</i></b></marquee></p>
                </div>



                  <div class="row">
                    <div class="column nature">
                      <div class="content">
                        <img src="breaking news.jpeg" alt="breaking news" style="width:50%">
                        <h4>BREAKING NEWS</h4>
                      </div>
                    </div>
                    <div class="column nature">
                      <div class="content">
                      <img src="COMMERCIAL NEWS.jpg" alt="Lights" style="width:30%">
                        <h4>COMMERCIAL NEWS</h4>
                      </div>
                    </div>
                    <div class="column nature">
```

NEWS TRACKER

```
  <div class="content">
  <img src="ent.jpg" alt="Nature" style="width:100%">
    <h4>ENTERTAINMENT NEWS</h4>
    <p></p>
  </div>
</div>

<div class="column cars">
  <div class="content">
    <img src="HN.jpg" alt="Car" style="width:60%">
    <h4>HISTORICAL NEWS</h4>
  </div>
</div>
<div class="column cars">
  <div class="content">
  <img src="inews.jpg" alt="Car" style="width:80%">
    <h4>INTERNATIONAL NEWS</h4>
  </div>
</div>
<div class="column cars">
  <div class="content">
  <img src="lcn.png" alt="Car" style="width:100%">
    <h4>LOCAL NEWS</h4>
  </div>
</div>

<div class="column people">
  <div class="content">
    <img src="spn.jpeg" alt="Car" style="width:100%">
    <h4>SPORTS NEWS</h4>
  </div>
</div>
<div class="column people">
  <div class="content">
  <img src="wea.jpeg" alt="Car" style="width:80%">
    <h4>WEATHER NEWS</h>
  </div>
</div>
<div class="column people">
  <div class="content">
  <img src="bn.png" alt="Car" style="width:50%">
    <h4>BUSINESS NEWS</h4>
  </div>
```

NEWS TRACKER

```
    </div>
  </div>

  </div>

 <script src="script grid.js"></script>
 <br></div>
<div class="footer">
  <h2>VNEWS</h2>
  <ul>About us</ul>
  <ul>Resouces</ul>
  <ul>Get started</ul>
  <ul>help</ul>
</div>

</body>
</html>
```

```
from flask import Flask,
render_template
                                        from newsapi import NewsApiClient

                                        app = Flask(__name__)
                                        @app.route('/')
                                        def dashboard():
                                            return render_template('dashboard.html')

                                        @app.route('/india')
                                        def india():
                                            newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
                                            topheadlinesindia= newsapi.get_top_headlines(sources =
                                        headlines?country=in")
                                            articles =topheadlinesindia['articles']

                                            desc = []
                                            news = []
                                            img = []
                                            for i in range(len(articles)):
                                                myarticles=articles[i]
                                                news.append(myarticles['title'])
                                                desc.append(myarticles['description'])
                                                img.append(myarticles['urlToImage'])
                                            mylist1=zip(news,desc,img)
```

```
    return render_template ("india.html", context = mylist1

@app.route('/southkorea')
def southkorea():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinessouthkorea = newsapi.get_top_headlines(sour
headlines?country=kr")
    articles = topheadlinessouthkorea['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist2=zip(news,desc,img)
    return render_template ("southkorea.html", context = my

@app.route('/thailand')
def thailand():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinesthailand= newsapi.get_top_headlines(sources
headlines?country=th")
    articles = topheadlinesthailand['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist3 =zip(news,desc,img)
    return render_template ("thailand.html", context = myli

@app.route('/unitedkingdom')
def unitedkingdom():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinesuk= newsapi.get_top_headlines(sources = "ht
headlines?country=gb")
    articles =topheadlinesuk['articles']
    desc = []
    news = []
```

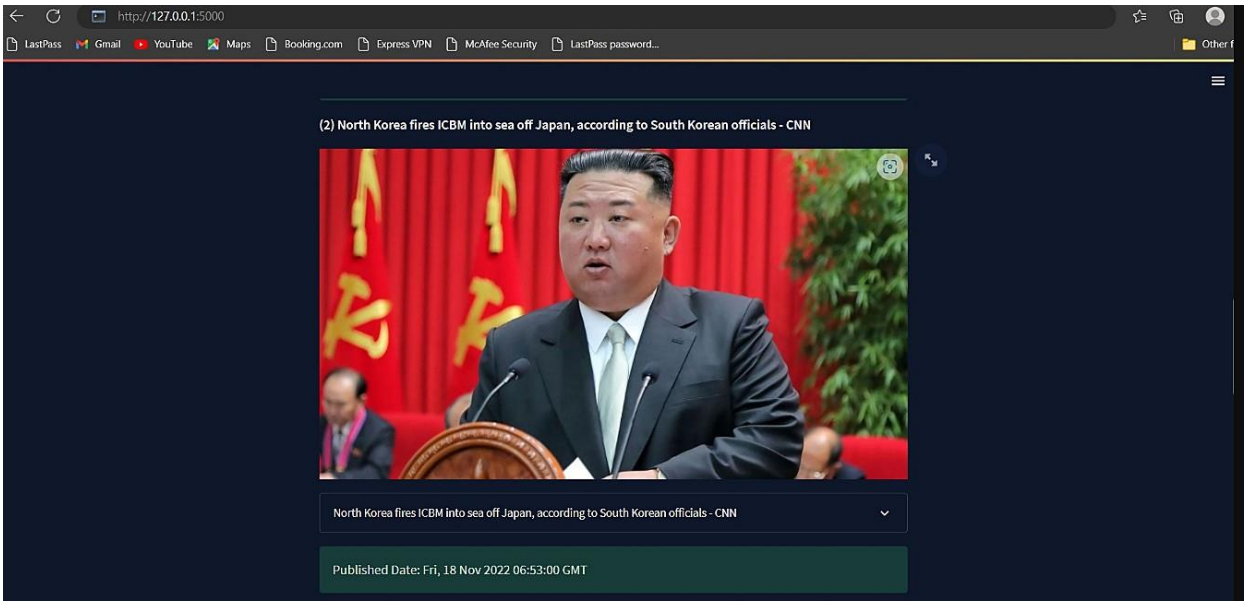
```
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist4 =zip(news,desc,img)
    return render_template ("unitedkingdom.html", context =

@app.route('/ukrane')
def ukrane():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinesukrane= newsapi.get_top_headlines(sources =
headlines?country=ua")
    articles =topheadlinesukrane['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist5 =zip(news,desc,img)
    return render_template ("ukrane.html", context = mylist

@app.route('/russia')
def russia():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinesrussia= newsapi.get_top_headlines(sources =
headlines?country=ru")
    articles =topheadlinesrussia['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist6 =zip(news,desc,img)
    return render_template ("russia.html", context = mylist
```

NEWS TRACKER

```
@app.route('/tiwan')
def tiwan():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinestiwan= newsapi.get_top_headlines(sources =
headlines?country=tw")
    articles =topheadlinestiwan['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist7 =zip(news,desc,img)
    return render_template ("tiwan.html", context = mylist7

@app.route('/france')
def france():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinesfrance= newsapi.get_top_headlines(sources =
headlines?country=fr")
    articles =topheadlinesfrance['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist8 =zip(news,desc,img)
    return render_template ("france.html", context = mylist

@app.route('/germany')
def germany():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlinesgermany= newsapi.get_top_headlines(sources
headlines?country=de")
    articles =topheadlinesgermany['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
```

NEWS TRACKER

```
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist9 =zip(news,desc,img)
    return render_template ("germany.html", context = mylis

@app.route('/china')
def china():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a
    topheadlineschina = newsapi.get_top_headlines(sources =
headlines?country=cn")
    articles = topheadlineschina['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist10 =zip(news,desc,img)
    return render_template ("china.html", context = mylist1

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



NEWS TRACKER

CREATING FEATURES

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

<!DOCTYPE
html>
                <html lang="en">
                <head>
                <title>china</title>
                <meta charset="UTF-
                8">
                <body>china
                </head>
                </html>

<!DOCTYPE
html>
                <html lang="en">
                <head>
                <title>china</title>
                <meta charset="UTF-
                8">
                <body>china
                </head>
                </html>

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

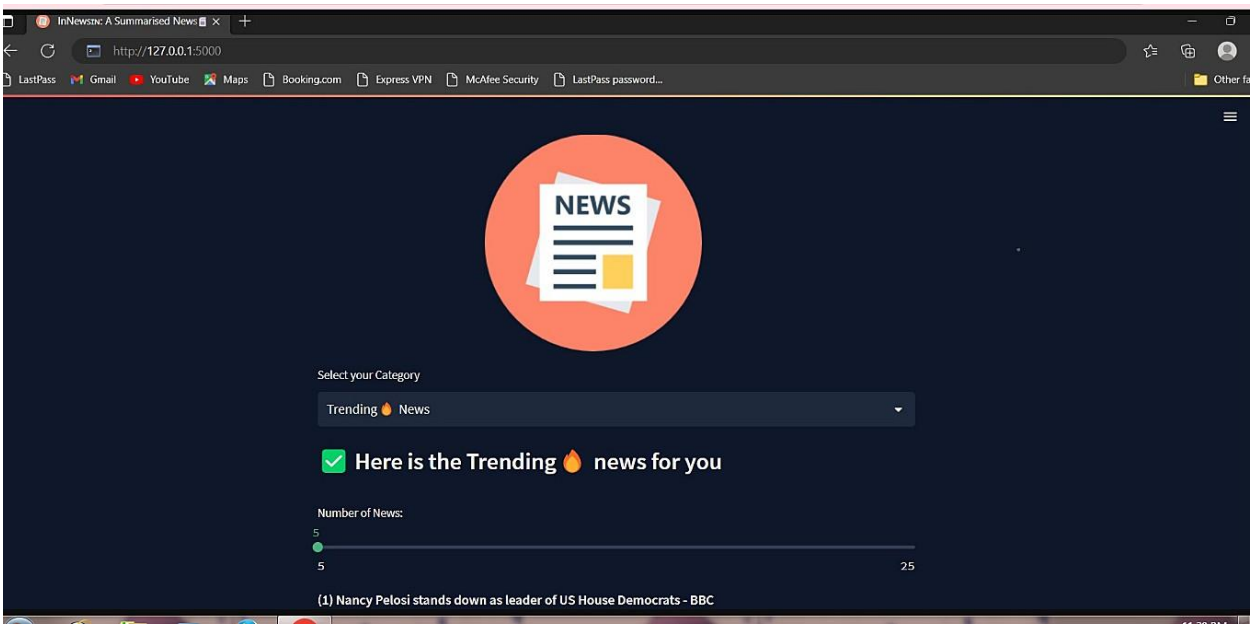
NEWS TRACKER

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

<!DOCTYPE
html>
                <html lang="en">
                <head>
                <title>china</title>
                <meta charset="UTF-
                8">
                <body>china
                </head>
                </html>

<!DOCTYPE
html>
                <html lang="en">
                <head>
                <title>china</title>
                <meta charset="UTF-
                8">
                <body>china
                </head>
                </html>

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

NEWS TRACKER

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

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



SPRINT 4

CONNECTING DB2

```
import
ibm_db
                hostname="2d46b6b4-cbf6-40eb-bbce-
```

NEWS TRACKER

```
6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
uid="pyd03172"
password="yVoUxh8d5GJAup2S"
driver="(IBM DB2 ODBC DRIVER)"
db="bludb"
port="32328"
protocol="TCPIP"
cert="DigiCertGlobalRootCA"
dsn=(
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "UID={3};"
    "SECURITY=SSL;"
    "SSLServerCertificate={4};"
    "PWD={5};").format(db,hostname,port,uid,cert,password)
print(dsn)
try:
    db2=ibm_db.connect(dsn,"","")
    print("conneccted to database")
except:
    print("unable to connect",ibm_db.conn_errormsg())
```

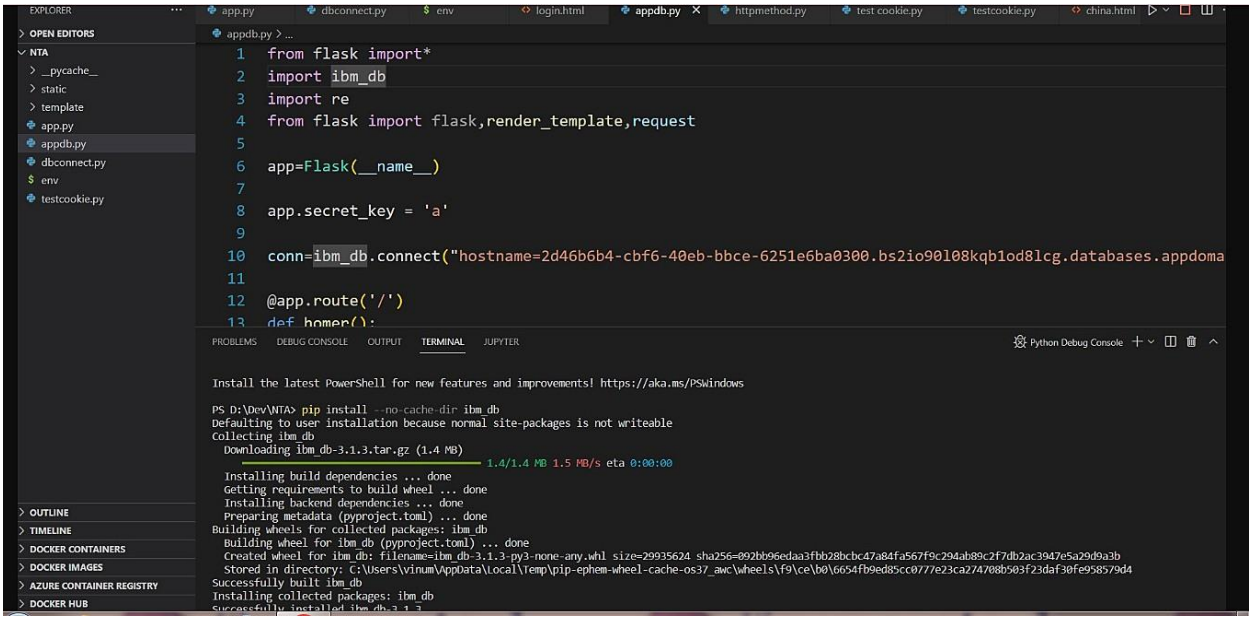
```
from flask
import*
```

```
import ibm_db
import re
from flask import flask,render_template,request

app=Flask(__name__)

app.secret_key = 'a'

conn=ibm_db.connect("hostname=2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kq
DRIVER);db=bludb;port=32328;protocol=TCPIP;cert=DigiCertGlobalRootCA;")

@app.route('/')
def homer():
    return render_template('login.html')

@app.route('/backlogin')
def backlogin():
```

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

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

    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM login WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'

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

@app.route('/register', methods =['GET', 'POST'])
def registet():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        phone_num = request.form['phone_num']
        password = request.form['confirm_password']
        sql = "SELECT * FROM login WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
```

```
        print(account)
        if account:
            msg = 'Account already exists !'
            return render_template('login.html', msg = msg)
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        elif not re.match(r'[0-9]+', phone_num):
            msg = 'phone number must contain only numbers !'
        else:
            insert_sql = "INSERT INTO user_details VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phone_num)
            ibm_db.bind_param(prep_stmt, 4, password)
            ibm_db.execute(prep_stmt)

            insert_sql_1 = "INSERT INTO login VALUES (?, ?)"
            prep_stmt_1 = ibm_db.prepare(conn, insert_sql_1)
            ibm_db.bind_param(prep_stmt_1, 1, username)
            ibm_db.bind_param(prep_stmt_1, 2, password)
            ibm_db.execute(prep_stmt_1)
            msg = 'You have successfully registered !'
            return render_template('login.html', msg = msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form !'
        return render_template('register.html', msg = msg)
    return render_template('register.html', msg = msg)

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

NEWS TRACKER



MAINTAINING COOOKIES

```
from flask import
*
                    from flask import Flask, render_template
                    app=Flask(__name__,template_folder='templat
                    es')
                    @app.route('/')
                    def setcookie():
                        res = make_response("cookie is
                    inserted")
                        res.set_cookie('Flask','framework')
                        return res

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

NEWS TRACKER

8.TESTING

8.1 TEST CASES

Test case

Test case	feature	component	Test scenario	Expected result	Actual result	status	comments	bug	Executed by
Sign in	Functional	Login page	Verify user can see the sign in option	can visible	Yes visible	pass	successful	-	libia
Sign up	Functional	Login page	Verify user has the option to sign up	Can visible	Yes visible	pass	Successful	-	libia
Forgot password	Functional	Login page	Verify user has the option to forgot password	Yes the option is available	Option is available	pass	Successful	-	jeevitha
Fetch news	Functional	Home page	Verify user can get the news	News will be feed to the app	404 error	Fail	unsuccessful	App integeration problem	Arun,Ram
Types of news available in the fetch news page	Functional	Fetch news	Types of news available	Weather, Sport, Economy.	Hover buttons will be shown.	Yes	successful	-	kaviya

8.2 USER ACCEPTANCE TESTING

An easy-to-use, “one-click” system for end-users to report bugs and give feedback.Screenshots and annotations to make reports as actionable as possible.Automatic capture of environment info and console logs.Deep integration with your existing PM tools (Jira, GitHub, Trello…).Support for alpha testing and beta testing test cases.In a nutshell, the app records sessions of users and visitors on your app or website.

During UAT testing, FullStory comes in handy to help you understand what steps

led to a particular bug and how to reproduce the bug for yourself.we use the app in two ways:

- Straightforward recording. If an error pops up, it will show up in the recording or the console logs. At this stage, it's pretty easy for us to conclude what happened and how to fix it.
- Abandoned pages/confusion**.** If a user gets stuck during testing, we can retrace their journey. Then, we set up a meeting to replay the session and ask them what was unclear.

And with the we get the best of both worlds: the exact timestamp of when our tester reported a bug—allowing us to investigate what happened seconds before the report.

9. RESULTS

This will **help the users to share news on various platforms such as Twitter and Facebook**. This will not only give an amazing user experience and also will also increase the views.Google news is a personalized news aggregator that organizes and highlights what's happening in the world so you can discover more about the stories that matter to you.Visible right when the user is looking for a distraction or a clever way to use some free time. Read on to learn more about news app development, including why and how to build your own news app.

9.1 PERFORMANCE METRICES

A mobile app is a powerful business tool — its success needs to be measured just like any other business key metrics. Measurements that require special attention include tracking revenue, average check size, customer acquisition costs, retention rate, downloads, and user satisfaction.

After reading this article, you'll understand the most crucial mobile application performance metrics.

10. ADVANTAGES & DISADVANTAGES

- Enrich Your Knowledge.

- Stay Connected With The World.
- Strengthen your Language skills and Enhance your Vocabulary.
- Be Part of a Larger Conversation.
- Be Informed About the Latest Discoveries and Innovations.

Viewers can get their news straight off their smartphone or tablet computer. News is at their fingertips in an instant. An online newspaper can be read more elaborate than a printed newspaper. You can read the old issues too very easily at the click of the mouse.

- Wastage of Paper: Millions of papers are printed every day using a few million bits of paper.
- Can be Time Wasting: Most individuals who read papers have the habit of perusing it in the first part of the day with their favourite thing in the world.
- Misinformation spreads like wildfire.
- We can live in an ideological bubble.
- There is fierce media competition.
- There is a wider customer base for companies large and small.
- Children can access inappropriate information more easily.

11. CONCLUSION

As news is increasingly accessed on smartphones and tablets, the need for personalising news app interactions is apparent. We report a series of three studies addressing key issues in the development of adaptive news app interfaces. We first surveyed users' news reading preferences and behaviours; analysis revealed three primary types of reader. We then implemented and deployed an Android news app that logs users' interactions with the app. We used the logs to train a classifier and showed that it is able to reliably recognise a user according to their reader type. Finally we evaluated alternative, adaptive user interfaces for each reader type. The evaluation demonstrates the differential benefit of the adaptation for different users of the news app and the feasibility of adaptive interfaces for news apps.

12. FUTURE SCOPE

The scope of your app is not in using the APIs available. The depend only on your successful

implementation of the UI, UX and features that the user will love. There are many apps that provide news based on location. Some are quite popular even though the underlying technology is quite simple due to their effective UI, UX and fluid navigation UI. Some examples may include News republic (Though it is more than just a location based news app), Flipboard, Google News, etc...A news a app needs credibility and a name to which it is associated . And if not not exactlythere will be some apps that revovle around the same idea. But anyways you can go ahead with development as long as the app has its own unique elements . But where are you planning to source the news , the Internet ? You'll have to invest to promote the app and let the users know about it.Once you decide to buya plot, make sure you have all documents in place. Without these, your purchase will be delayed. Having all the proper legal documentation will help protect your land and home from any disputes in the future. Consult a lawyer to help you with every step of the documentation process.Most of the required documents can be grouped into two types - legal and personal.Legal documents: These documents are essential, and missing even one of these canresult in a delay in purchase.

13. APPENDIX

Home.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>home.html</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="style.css">
</head><html>
  <head>
```

NEWS TRACKER

```
  <meta name="viewport" content="width=device-width, initial-scale=1">

<body>

  <div class="navbar">
    <a href="#">HOME</a>
    <a  href="register.html">REGISTER</a>
    <a href="login.html">LOGIN</a>
    <a  href="#"  class="right">CHATBOT</a>
  </div>

<div class="header">
  <div class="bg-image"></div><div class="bg-text">
    <img src="img.jpeg" alt="Avatar" class="avatar">
            <p><marquee><b><i>More News! More Often! Move Closer To Tour
World!</i></b></marquee></p>
</div>


  <div class="row">
    <div class="column nature">
      <div class="content">
        <img src="breaking news.jpeg" alt="breaking news" style="width:50%">
        <h4>BREAKING NEWS</h4>
      </div>
    </div>
    <div class="column nature">
      <div class="content">
      <img src="COMMERCIAL NEWS.jpg" alt="Lights" style="width:30%">
```

NEWS TRACKER

```
    <h4>COMMERCIAL NEWS</h4>
  </div>
</div>
<div class="column nature">
  <div class="content">
  <img src="ent.jpg" alt="Nature" style="width:100%">
    <h4>ENTERTAINMENT NEWS</h4>
    <p></p>
  </div>
</div>

<div class="column cars">
  <div class="content">
    <img src="HN.jpg" alt="Car" style="width:60%">
    <h4>HISTORICAL NEWS</h4>
  </div>
</div>
<div class="column cars">
  <div class="content">
  <img src="inews.jpg" alt="Car" style="width:80%">
    <h4>INTERNATIONAL NEWS</h4>
  </div>
</div>
<div class="column cars">
  <div class="content">
  <img src="lcn.png" alt="Car" style="width:100%">
    <h4>LOCAL NEWS</h4>
  </div>
</div>
```

NEWS TRACKER

```
    <div class="column people">
      <div class="content">
        <img src="spn.jpeg" alt="Car" style="width:100%">
        <h4>SPORTS NEWS</h4>
      </div>
    </div>
    <div class="column people">
      <div class="content">
      <img src="wea.jpeg" alt="Car" style="width:80%">
        <h4>WEATHER NEWS</h>
      </div>
    </div>
    <div class="column people">
      <div class="content">
      <img src="bn.png" alt="Car" style="width:50%">
        <h4>BUSINESS NEWS</h4>
      </div>
    </div>
  </div>

  </div>

 <script src="script grid.js"></script>
 <br></div>
<div class="footer">
  <h2>VNEWS</h2>
  <ul>About us</ul>
  <ul>Resouces</ul>
```

```
  <ul>Get started</ul>
  <ul>help</ul>
</div>

</body>
</html>
```

Login.html:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}

input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #43255e;
```

```
  color: white; padding:
  14px 20px;margin:
  8px 0; border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color:  #f44336;
}

.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
  position: relative;
}

img.avatar {
  width: 10%;
  border-radius: 20%;
}
```

NEWS TRACKER

```
.container {
  padding: 16px;
}

span.psw { float:
  right;
  padding-top: 16px;
}

.modal { display:
  none; position:
  fixed;z-index:
  1;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgb(0,0,0);
  background-color: rgba(0,0,0,0.4);
  padding-top: 60px;
}

.modal-content { background-
  color: #fefefe;margin: 5%
  auto 15% auto;border: 1px
  solid #888; width: 80%;
}
```

NEWS TRACKER

```
.close {
  position: absolute;
  right: 25px;
  top: 0;
  color: #000;
  font-size: 35px;
  font-weight: bold;
}

.close:hover,
.close:focus {
  color: red;
  cursor: pointer;
}

.animate {
  -webkit-animation: animatezoom 0.6s;animation:
  animatezoom 0.6s
}

@-webkit-keyframes animatezoom {
  from {-webkit-transform: scale(0)} to {-
  webkit-transform: scale(1)}
}

@keyframes animatezoom {
  from {transform: scale(0)} to
  {transform: scale(1)}
```

```
}

@media screen and (max-width: 300px) {
  span.psw {
     display: block;float:
     none;
  }
  .cancelbtn {
     width: 100%;
  }
}
</style>
</head>
<body>

  <form class="modal-content animate" action="/action_page.php" method="post">
    <div class="imgcontainer">
        <span onclick="document.getElementById('id01').style.display='none'" class="close"
title="Close Modal">&times;</span>
      <img src="img.jpeg" alt="Avatar" class="avatar">
    </div>

    <div class="container">
      <label for="uname"><b>Username</b></label>
      <input type="text" placeholder="Enter Username" name="uname" required>

      <label for="psw"><b>Password</b></label>
      <input type="password" placeholder="Enter Password" name="psw" required>
```

```
      <button type="submit"><a href="dashboard.html">Login</a></button>
      <label>
        <input type="checkbox" checked="checked" name="remember"> Remember me
      </label>
    </div>

    <div class="container" style="background-color:#f1f1f1">
      <span class="psw">Forgot <a href="#">password?</a></span>
    </div>
  </form>



<script>
var modal = document.getElementById('id01');
window.onclick = function(event) {
    if (event.target == modal) { modal.style.display
        = "none";
    }
}
</script>

</body>
</html>
```

Register.html:

```
<!DOCTYPE html>
```

NEWS TRACKER

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
  font-family: Arial, Helvetica, sans-serif;
  background-color: black;
}

* {
  box-sizing: border-box;
}

/* Add padding to containers */
.container {
  padding: 16px;
  background-color: white;
}

/* Full-width input fields */ input[type=text],
input[type=password] {width: 100%;
  padding: 15px; margin:
  5px 0 22px 0;display:
  inline-block; border:
  none; background:
  #f1f1f1;
}
```

NEWS TRACKER

```
input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

/* Overwrite default styles of hr */hr
{
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Set a style for the submit button */
.registerbtn {
  background-color: #43255e;
  color: white;
  padding: 16px 20px;
  margin: 8px 0;
  border: none; cursor:
  pointer; width: 100%;
  opacity: 0.9;
}

.registerbtn:hover {
  opacity: 1;
}

/* Add a blue text color to links */
a {
```

```
  color: dodgerblue;
}

/* Set a grey background color and center the text of the "sign in" section */
.signin {
  background-color: #f1f1f1;
  text-align: center;
}
</style>
</head>
<body>

<form action="/action_page.php">
  <div class="container">
    <h1>Register</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email" id="email" required>

    <label for="psw"><b>Password</b></label>
            <input type="password" placeholder="Enter Password" name="psw" id="psw"required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
     <input type="password" placeholder="Repeat Password" name="psw-repeat" id="psw-repeat"
required>
    <hr>
```

NEWS TRACKER

```
    <p>By creating an account you agree to our <a href="#">Terms & Privacy</a>.</p>

    <button type="submit" class="registerbtn">Register</button>
  </div>

  <div class="container signin">
    <p>Already have an account? <a href="login.html">Sign in</a>.</p>
  </div>
</form>

</body>
</html>
```

style.css

```
* {
    box-sizing: border-box;
  }
  .bg-text {
    background-color: black;
    background-color: rgba(0,0,0, 0.4);
    color: white;
    font-weight: bold;
    position: absolute;
    top: 50%;
    left: 44%;
```

```
  transform: translate(-40%, -40%);
  z-index: 2;
  width: 60%;
  padding: 10px;
  text-align: center;
}
body {
  font-family: Arial, Helvetica, sans-serif;margin:
  0;
}
body, html {
  height: 100%;
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}

* {
  box-sizing: border-box;
}

.bg-image {
  background-image: url("news.jpg");height:
  100%;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
```

NEWS TRACKER

```
.header {
    background-image:url("news.jpg") no-repeat;height:
    60%;
    background-position: center;
    background-repeat: no-repeat;
    background-size: cover;
    position: relative;
    text-align: center;

}



.header h1 { font-
  size: 30px;
}

img.avatar {
  width: 10%;
  border-radius: 20%;
}

.navbar  { overflow:
  hidden;
  background-color: #43255e;
}



.navbar a {
  float: left;
```

```
  display: block;color:
  white;
  text-align: center;
  padding: 14px 20px;
  text-decoration: none;
}



.navbar a.right {
  float: right;
}

.navbar a:hover {
  background-color: #ddd;
  color: black;
}



.footer { padding:
  20px;color:
  white;
  background: #43255e;
  margin-top:70%;
}



@media screen and (max-width: 700px) {
  .row {
    flex-direction: column;
```

```
  }
}



@media screen and (max-width: 400px) {
  .navbar a {
    float: none;
    width: 100%;
  }
}
* {
  box-sizing: border-box;
}

body {
  background-color: #f1f1f1;
  padding: 20px;
  font-family: Arial;
}

.main {
  max-width: 1000px;
  margin: auto;
}

h1 {
  font-size: 50px;

  word-break: break-all;
```

```
  }

.row {
  margin: 10px -16px;
}

.row,
.row > .column {
  padding: 8px;
}

.column {
  float: left;
  width: 33.33%;
  display: none;
}
.content {
  background-color: white;
  padding: 10px;
}

.show { display:
  block;
}
```

script.js filterSelection("all")

```
function filterSelection(c) {
```

NEWS TRACKER

```
  var x, i;
  x =  document.getElementsByClassName("column");if (c
  == "all") c = "";
  for (i = 0; i < x.length; i++) {
    w3RemoveClass(x[i], "show");
    if (x[i].className.indexOf(c) > -1) w3AddClass(x[i], "show");
  }
}

function w3AddClass(element, name) {var
  i, arr1, arr2;
  arr1  =  element.className.split("  ");arr2
  = name.split(" ");
  for (i = 0; i < arr2.length; i++) {
    if (arr1.indexOf(arr2[i]) == -1) {element.className += " " + arr2[i];}
  }
}

function w3RemoveClass(element, name) {var
  i, arr1, arr2;
  arr1  =  element.className.split("  ");arr2
  = name.split(" ");
  for (i = 0; i < arr2.length; i++) { while
    (arr1.indexOf(arr2[i]) > -1) {
      arr1.splice(arr1.indexOf(arr2[i]),  1);
    }
  }
  element.className = arr1.join(" ");
}
```

NEWS TRACKER

app.py

```
from flask import Flask, render_template
from newsapi import NewsApiClient

app = Flask(_name_)
@app.route('/')
def dashboard():
    return  render_template('dashboard.html')

@app.route('/india') def
india():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                                topheadlinesindia=          newsapi.get_top_headlines(sources              =
"https://newsapi.org/v2/top-headlines?country=in") articles
    =topheadlinesindia['articles']

    desc  =  []
    news  =  []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist1=zip(news,desc,img)
    return render_template ("india.html", context = mylist1)
```

NEWS TRACKER

```
@app.route('/southkorea')  def
southkorea():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                topheadlinessouthkorea  =  newsapi.get_top_headlines(sources  =
"https://newsapi.org/v2/top-headlines?country=kr") articles
    = topheadlinessouthkorea['articles']
    desc  =  []
    news  =  []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist2=zip(news,desc,img)
    return render_template ("southkorea.html", context = mylist2)


@app.route('/thailand')
def thailand():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                    topheadlinesthailand=  newsapi.get_top_headlines(sources  =
"https://newsapi.org/v2/top-headlines?country=th") articles
    = topheadlinesthailand['articles']
    desc  =  []
    news  =  []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
```

```
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist3 =zip(news,desc,img)
    return render_template ("thailand.html", context = mylist3)


@app.route('/unitedkingdom') def
unitedkingdom():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3") topheadlinesuk=
      newsapi.get_top_headlines(sources = "https://newsapi.org/v2/top-
headlines?country=gb")
    articles =topheadlinesuk['articles']desc =
    []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist4 =zip(news,desc,img)
    return render_template ("unitedkingdom.html", context = mylist4)


@app.route('/ukrane') def
ukrane():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                                topheadlinesukrane=          newsapi.get_top_headlines(sources              =
"https://newsapi.org/v2/top-headlines?country=ua") articles
    =topheadlinesukrane['articles']
    desc = []
```

NEWS TRACKER

```
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist5 =zip(news,desc,img)
    return render_template ("ukrane.html", context = mylist5)


@app.route('/russia')
def russia():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                topheadlinesrussia= newsapi.get_top_headlines(sources =
"https://newsapi.org/v2/top-headlines?country=ru") articles
    =topheadlinesrussia['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist6 =zip(news,desc,img)
    return render_template ("russia.html", context = mylist6)


@app.route('/tiwan')
def tiwan():
```

NEWS TRACKER

```
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                topheadlinestiwan= newsapi.get_top_headlines(sources =
"https://newsapi.org/v2/top-headlines?country=tw") articles
    =topheadlinestiwan['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist7 =zip(news,desc,img)
    return render_template ("tiwan.html", context = mylist7)


@app.route('/france')
def france():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                topheadlinesfrance= newsapi.get_top_headlines(sources =
"https://newsapi.org/v2/top-headlines?country=fr") articles
    =topheadlinesfrance['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
```

```
    mylist8 =zip(news,desc,img)
    return render_template ("france.html", context = mylist8)


@app.route('/germany')
def germany():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                topheadlinesgermany= newsapi.get_top_headlines(sources =
"https://newsapi.org/v2/top-headlines?country=de") articles
    =topheadlinesgermany['articles']
    desc = []
    news = []
    img = []
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist9 =zip(news,desc,img)
    return render_template ("germany.html", context = mylist9)


@app.route('/china')
def china():
    newsapi = NewsApiClient(api_key = "faeeea5c0d764c4faa9a2bcbd4af3ca3")
                topheadlineschina = newsapi.get_top_headlines(sources =
"https://newsapi.org/v2/top-headlines?country=cn") articles
    = topheadlineschina['articles']
    desc = []
    news = []
    img = []
```

```
    for i in range(len(articles)):
        myarticles=articles[i]
        news.append(myarticles['title'])
        desc.append(myarticles['description'])
        img.append(myarticles['urlToImage'])
    mylist10 =zip(news,desc,img)
    return render_template ("china.html", context = mylist10)

if__name___== "_main_":
    app.run(debug = True)
```

dbconnect.py

```
from flask import*
import ibm_db
import re
from flask import flask,render_template,request

app=Flask(__name__)

app.secret_key = 'a'

conn=ibm_db.connect("hostname=2d46b6b4-cbf6-40eb-bbce-
6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;uid=pyd03172pass
word=yVoUxh8d5GJAup2S;driver=(IBM                    DB2
                                                     ODBC
DRIVER);db=bludb;port=32328;protocol=TCPIP;cert=DigiCertGlobalRootCA;")

@app.route('/')
```

NEWS TRACKER

```
def homer():
    return  render_template('login.html')

@app.route('/backlogin') def
backlogin():
    return  render_template('login.html')

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

    if request.method == 'POST' :
        username = request.form['username']password
        = request.form['password']
        sql = "SELECT * FROM login WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)print
        (account)
        if account: session['loggedin']
            = True
            session['id'] = account['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']msg =
            'Logged in successfully !'
```

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

@app.route('/register',  methods  =['GET',  'POST'])def
registet():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']email =
        request.form['email']
        phone_num = request.form['phone_num']
        password = request.form['confirm_password']
        sql = "SELECT * FROM login WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
            return render_template('login.html', msg = msg)elif
        not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'elif not
        re.match(r'[0-9]+', phone_num):
```

NEWS TRACKER

```
        msg = 'phone number must contain only numbers !'else:
            insert_sql = "INSERT INTO user_details VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phone_num)
            ibm_db.bind_param(prep_stmt, 4, password)
            ibm_db.execute(prep_stmt)

            insert_sql_1 = "INSERT INTO login VALUES (?, ?)"
            prep_stmt_1 = ibm_db.prepare(conn, insert_sql_1)
            ibm_db.bind_param(prep_stmt_1, 1, username)
            ibm_db.bind_param(prep_stmt_1, 2, password)
            ibm_db.execute(prep_stmt_1)
            msg = 'You have successfully registered !' return
            render_template('login.html', msg = msg)
    elif request.method == 'POST': msg
        = 'Please fill out the form !'
        return  render_template('register.html',  msg  =  msg)return
    render_template('register.html', msg = msg)

if__name_=='_main_':app.run(host='0.0.0.0')
```