# 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.

# 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.

# 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.

# 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. | R.BakerNicholas  Bloom Steven  J.DaviskyleJ.kost | Policy News and stock Market Volatility | 2019 | The History of thoughts in financial market has shown a surprising lack of consensus about very fundamental question.what ultimately cause all those speculative assests like corporate stocks fluctuation in thme price of |
| 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 e- governance. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **AUTHOR** | **PAPER** | **YEAR** | **DESCRIPTION** |
| 4 | jn Kleppe and  Marco Otte | Analysing and understanding news consumption pattern by tracking online user behaviour with a multimodal research design | 2017 | The data collection,pre-processing ,and pattern dicovery takes more time. |
| 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, pre- processing, 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 touchscreen- enabled mobile devices, offered with flat-rate subscriptions for mobile internet, that widespread production and use of news- related content and services began to flourish. Accessing mobile news has gained traction in the  everyday life of the public. |
| 7 | stlund | Mobile News a review and model of journalism is an age of mobile media | 2012 | The importance of trust has been highlighted future research studies can focus on the antecendes of trust user require lower level of trust |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.NO** | **AUTHOR** | **PAPER** | **YEAR** | **DESCRIPTION** | | |
| 8 | Regonda Nagaraju, Mohammed Farhan Pasha, Mohammed Abdul Majeed, AdapaSujith | An Improved Method for Multi- Lingual 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 | 2017 |  | 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 | iGuno and Bo | Research on  Development Strategy of News App under the Background of Artificial Intelligence | 2019 | Alogorithm based recommended of this cutting-edge technology will face numerous tests from journalist,social and ethics and regulations. | | |

* 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.

# 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.

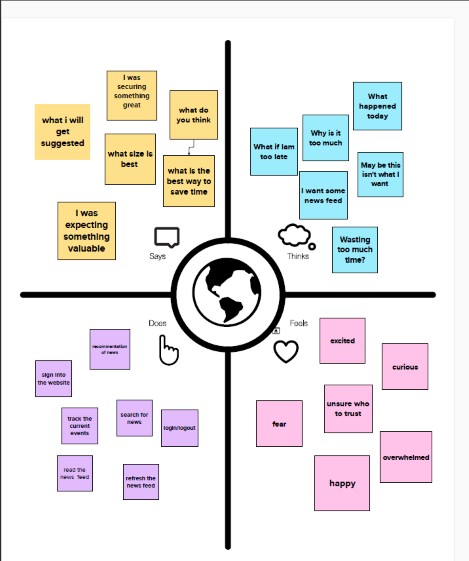
1. https://newsapi.org/
2. https://dzone.com/articles/how-to-parse-json-data- from-a-rest-api-using-simpl
3. https://material.io
4. https://developer.android.com/guide

# 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.

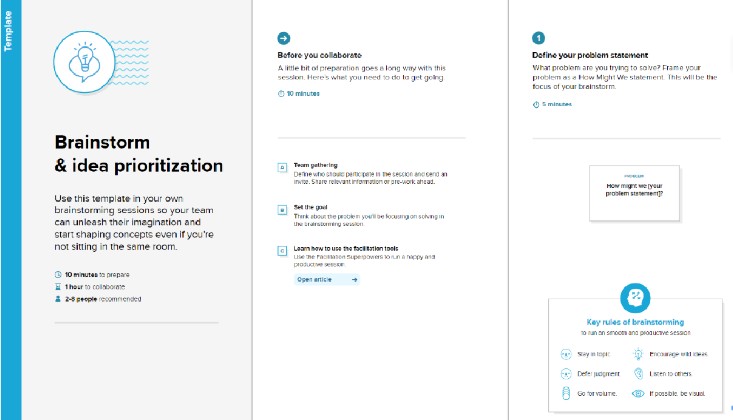
# IDEATION &PROPOSED SOLUTION

* 1. **EMPATHY MAP CANVAS**



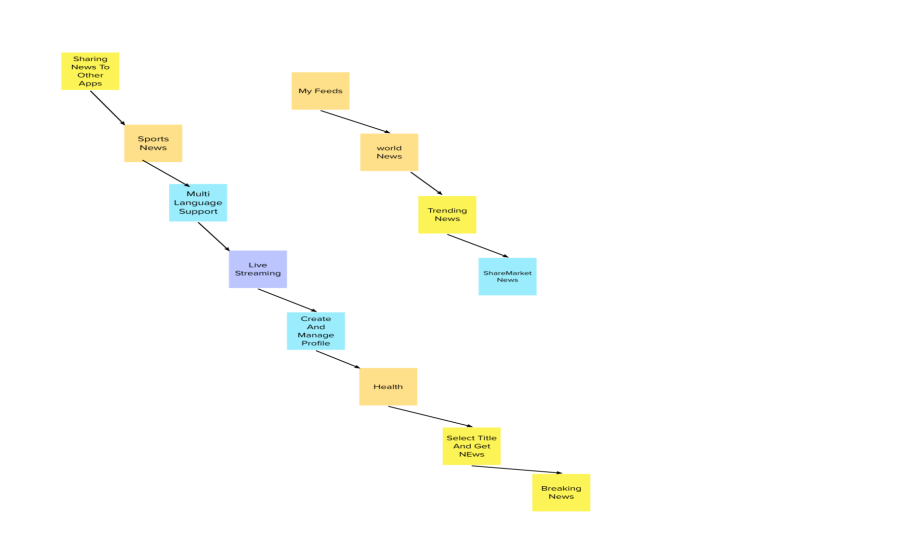
# IDEATION & BRAINSTORMING

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



step-2: Brainstorm,Idea listing and grouping



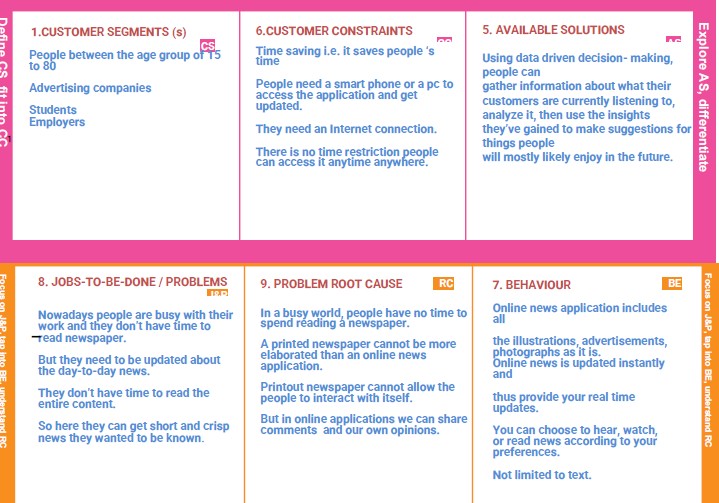
step-3: Idea Prioritization

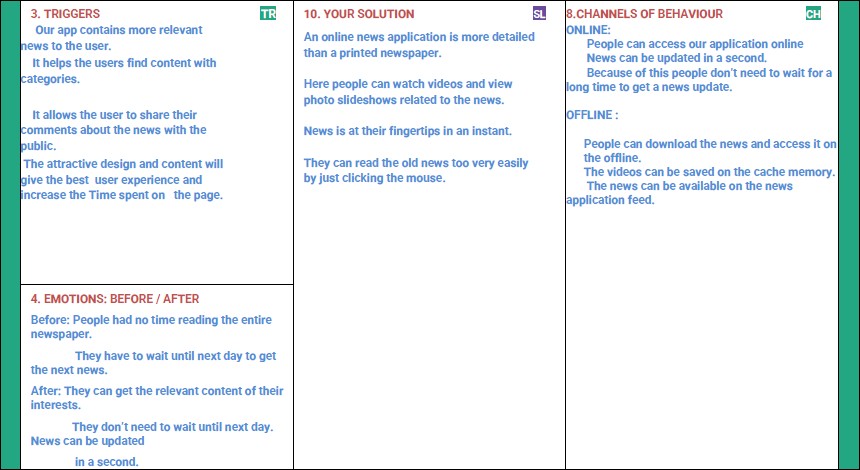
# PROPOSED SOLUTION

|  |  |  |
| --- | --- | --- |
| **S.N**  **o.** | **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. |

* 1. **PROBLEM SOLUTION FIT**





# REQUIREMENT ANALYSIS

* 1. **FUNCTIONAL REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form Registration 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 interests News feed can show news based on their frequently searched news. |
|  |  | It is user friendly |

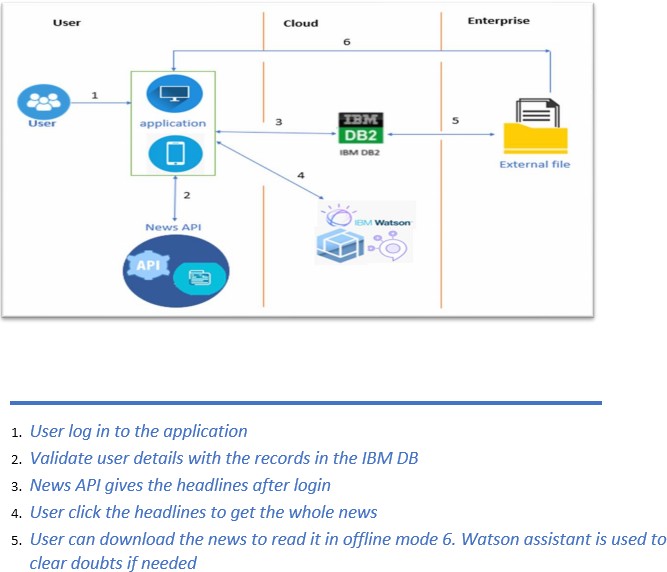
|  |  |  |
| --- | --- | --- |
| FR-4 | User access | Any user can access easily with their mobile phones. They can also download the news and access it on oﬄine. |
| FR-5 | User authentication | It has a good security authentication. Where user’s data cannot be stolen.  User’s data can be safe and secured. |
| 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 if it real or not. |

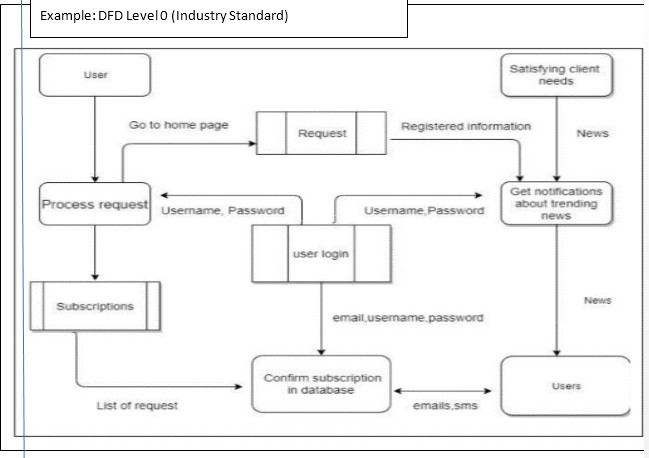
* 1. **NON-FUNCTIONAL REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | Any user can use this application easily because of its 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 |

|  |  |  |
| --- | --- | --- |
|  |  | 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 and gets 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. |

1. **PROJECT DESIGN**
   1. **DATA FLOW DIAGRAMS**





|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requireme**  **nt (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** |  | **Relea se** |
| Customer (Searching news) | Registrati on | USN-1 | As a user, I can register for the application by entering my email, password, and conﬁrming my  password. | I can access me account / dashboard | High | Sprin t1 |
|  |  | USN-2 | As a user, I will receive conﬁrmation  email once I | I can receive conﬁrmation email and click  conﬁrm | High | Sprin t1 |

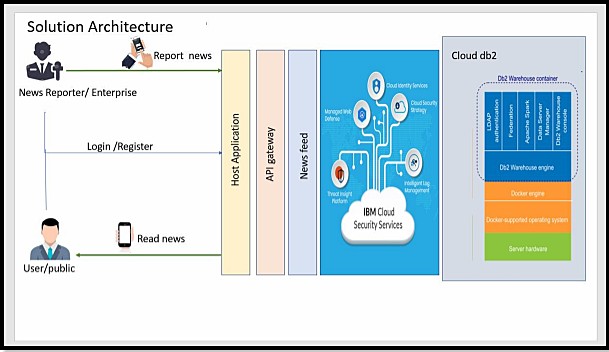
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | have registered for  the application |  |  |  |
|  |  | USN-3 | As a user, I can register for the application through their  given website | I can register and access the dashboard with Gmail or in  Browser Login | Low | Sprin t2 |
|  |  | USN-4 | As a user, I can register for the application  through Gmail |  | Medi um | Sprin t1 |
|  | 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 | Sprin t1 |
|  | 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 | Sprin  t-1 |

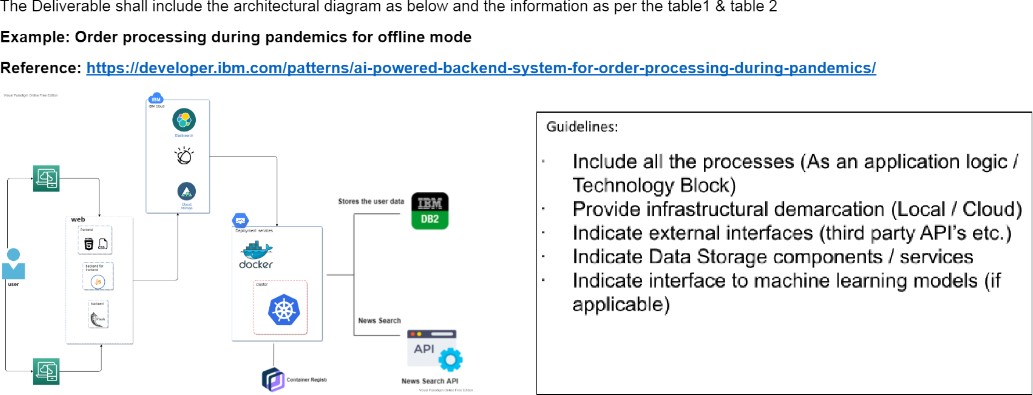
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | medium between client  and server | application and easily resolve my  speciﬁc issues |  |  |

* 1. **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, and delivered.





**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, |

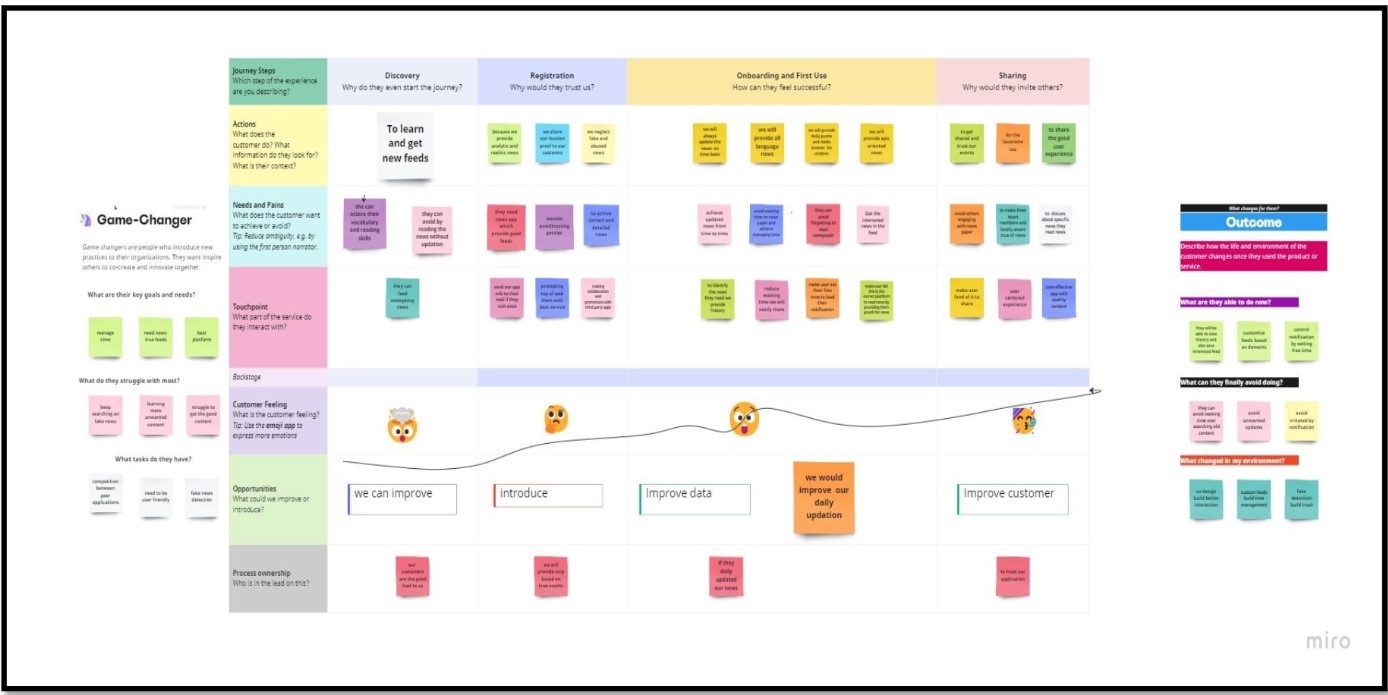
|  |  |  |  |
| --- | --- | --- | --- |
| 7. | File Storage | News feeds , video, audio,image | IBM Object Storage |
| 8. | External API-1 | secure ,socialize,manage and monetize, helping power digital transformation on premises and across the cloud. | IBM News API, etc. |
| 9. | Infrastructure (Server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

**Table-2: Application Characteristics:**

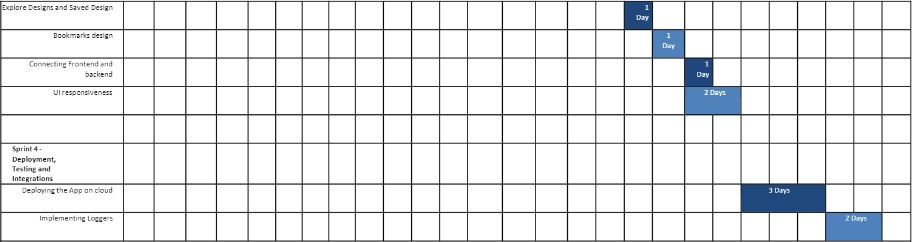
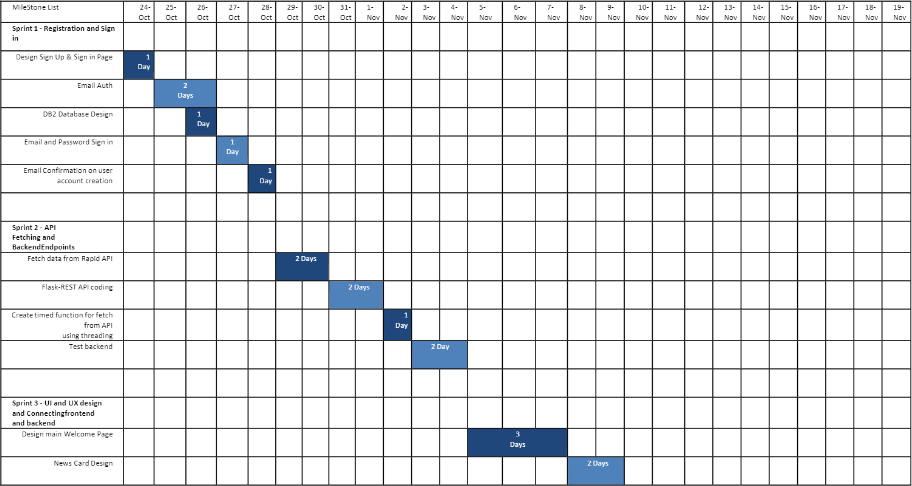
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | A software for which the original source code is made freely available and may be redistributed and modified according to the requirement of the user. | Python |
| 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. |

* 1. **USER STORIES**



1. **PROJECT PLANNING & SCHEDULING**
   1. **SPRINT PLANNING & ESTIMATION**



* 1. **SPRINT DELIVERY AND SCHEDULE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprin**  **t** | **Functional Requiremen**  **t (Epic)** | **Us**  **er St or y**  **Nu m**  **be**  **r** | **User Story / Task** | **St or y**  **Po int s** | **Priorit y** | **Team Members** |
| Sp  rint -1 | Registration | USN-  1 | C  r  e a  t i  n g  L o g  i  n  p a g e  C  r  e a t i  n g  R e g  i | 10 | Hig h | B.Venkatchevalian,M.yahoop,M.A.Ashwin,Chidhambaram |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | s  t  r  a  t i  o n  p a g  e |  |  |  |
| Sp  rint -1 | Da  ta  ba  se  Co nn  ectivit y | USN-  2 | To Stor e deta  ils of the custome  r  Con necting UI with Dat aba  se | 10 | Hig h | B.Venkatchevalian,M.yahoop,M.A.Ashwin |
| Sp  rint -2 | News  Tracker UI | USN-  3 | Building UI News  Tracker  Application | 10 | Hig h | B.Venkatchevalian, Chidambaram, M.A.Ashwin |
| Sp  rint -2 | API | USN-  4 | Co nnec tin g UI wit h  Ne  ws  A  PI,  G  oogl  e  Ne  ws  A  PI | 10 | Hig h | B.Venkatchevalian,M.yahoop,Chidambaram |
| Sp  rint -3 | S e n d  G  ri  d | USN-  5 | SendGrid  Integration With Python Code | 10 | Hig h | B.Venkatchevalian,M.yahoop,M.A.Ashwin |
|  | I  n  t  e g  r  a  ti o  n |  |  |  |  |  |
| Sp  rint -3 | News  Reader  (Voice) | USN-  6 | Building Voice Assistant to read the news | 10 | Hig h | B.Venkatchevalian, Chidambaram, M.A.Ashwin |
| Sp  rint -4 | Containerization | USN-  7 | Containerizing the app | 10 | Hig h | B.Venkatchevalian,M.yahoop,Chidambaram |
| Sp  rint -4 | Upload image and deploym  ent | USN-  8 | Uplo ad Dock  er  imag  e to  the IBM Regi  stry and deplo  y it in the  Kube  rnete s Clust er | 10 | Hig h | B.Venkatchevalian,M.yahoop,M.A.Ashwin |

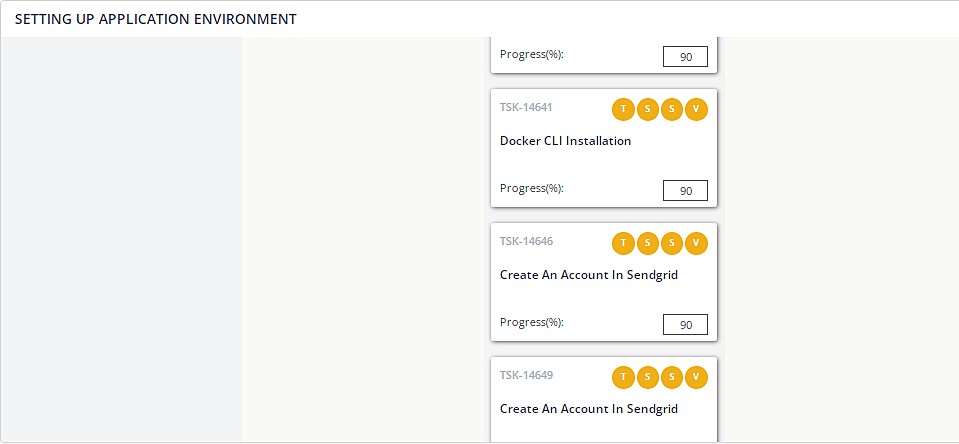
**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

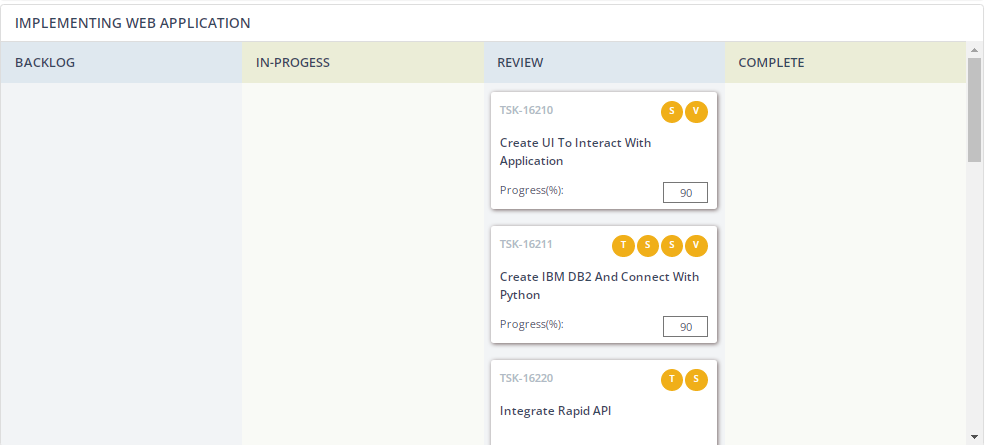
**Velocity:**

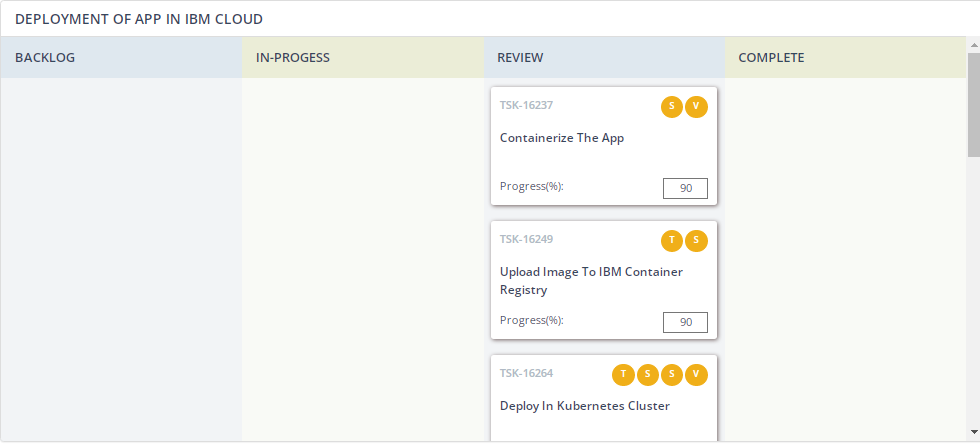
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day

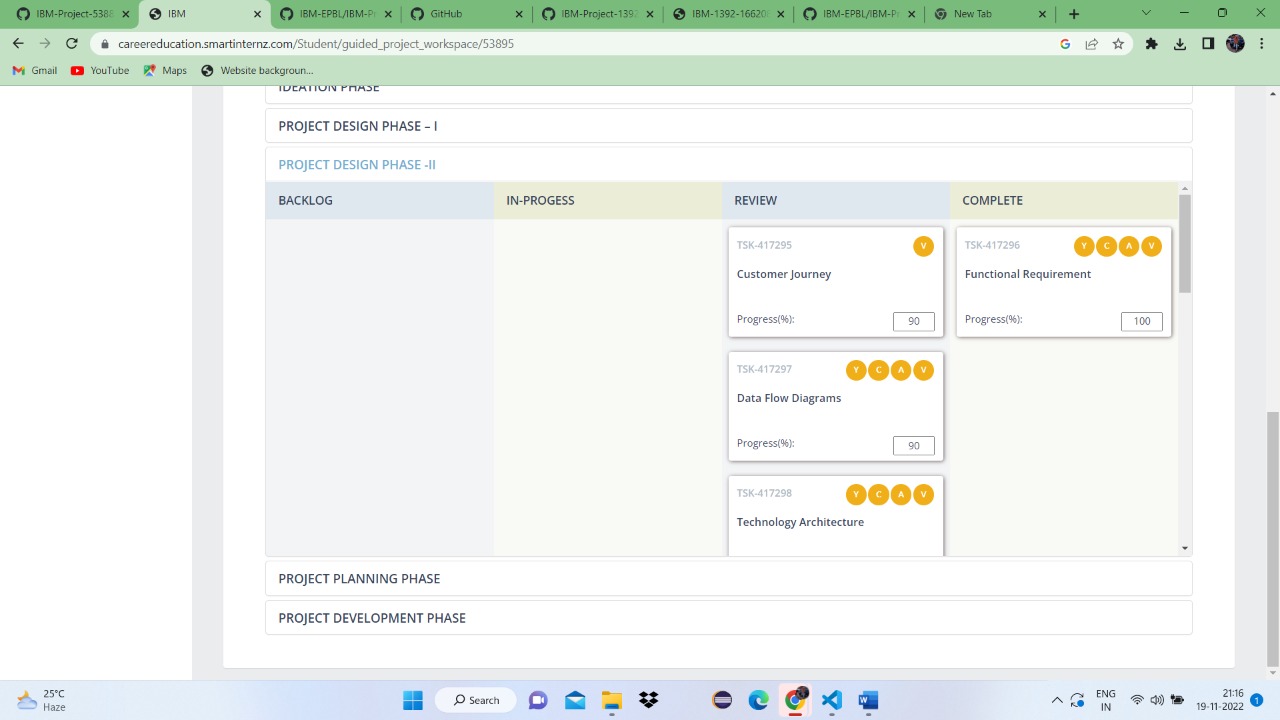
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total**  **Story**  **Points** | **Duration** | **Sprint**  **Start Date** | **Sprint End**  **Date**  **(Planned)** | **Story Points**  **Completed**  **(as on**  **Planned End**  **Date)** | **Sprint**  **Release**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

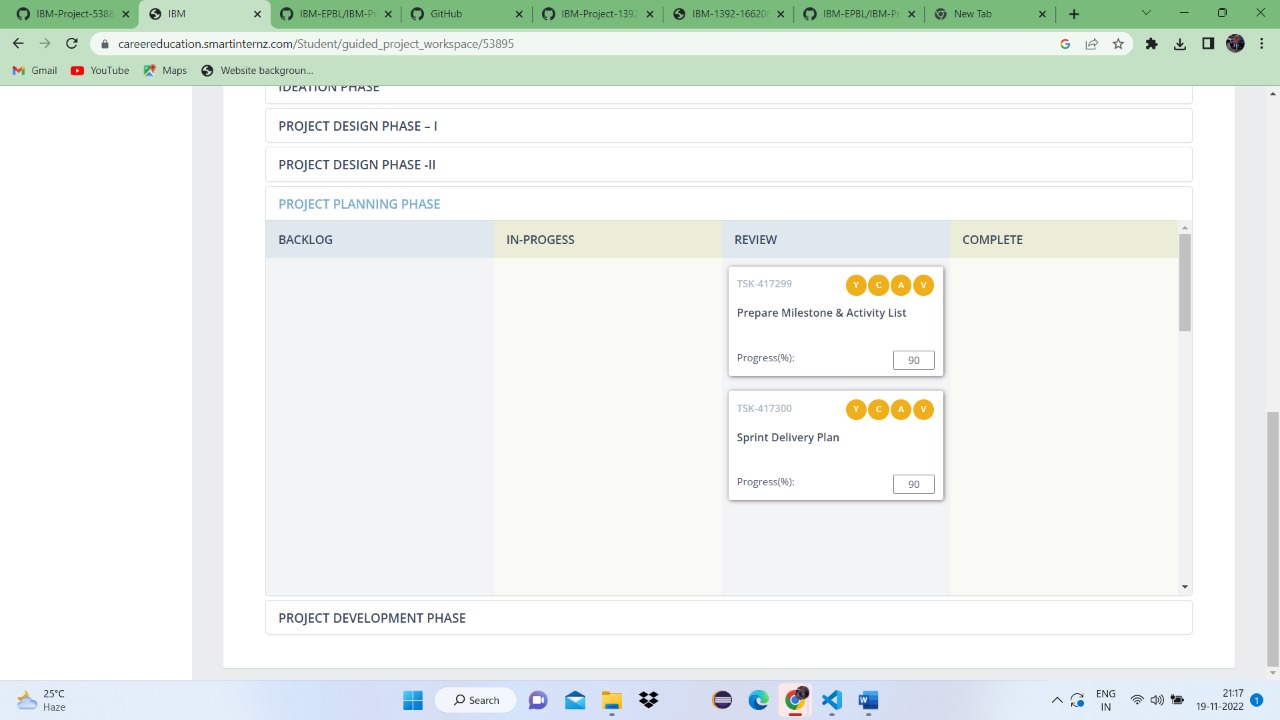
# REPORTS FROM JIRA

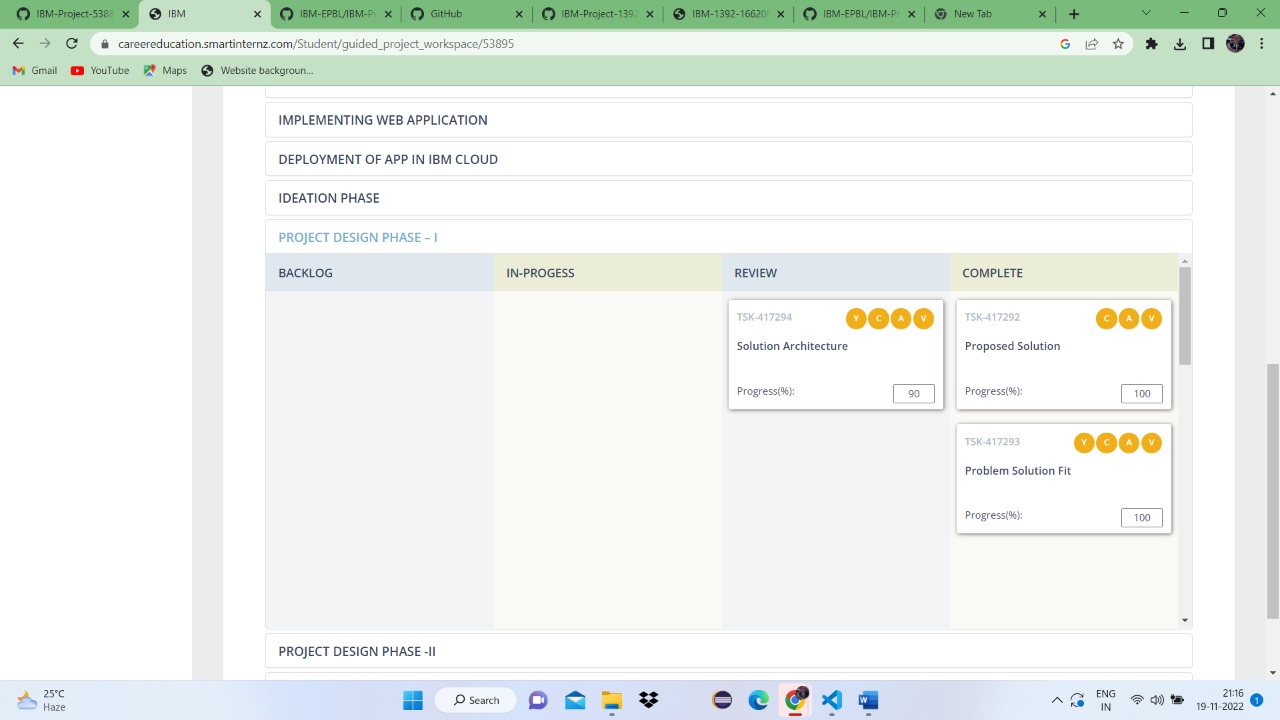


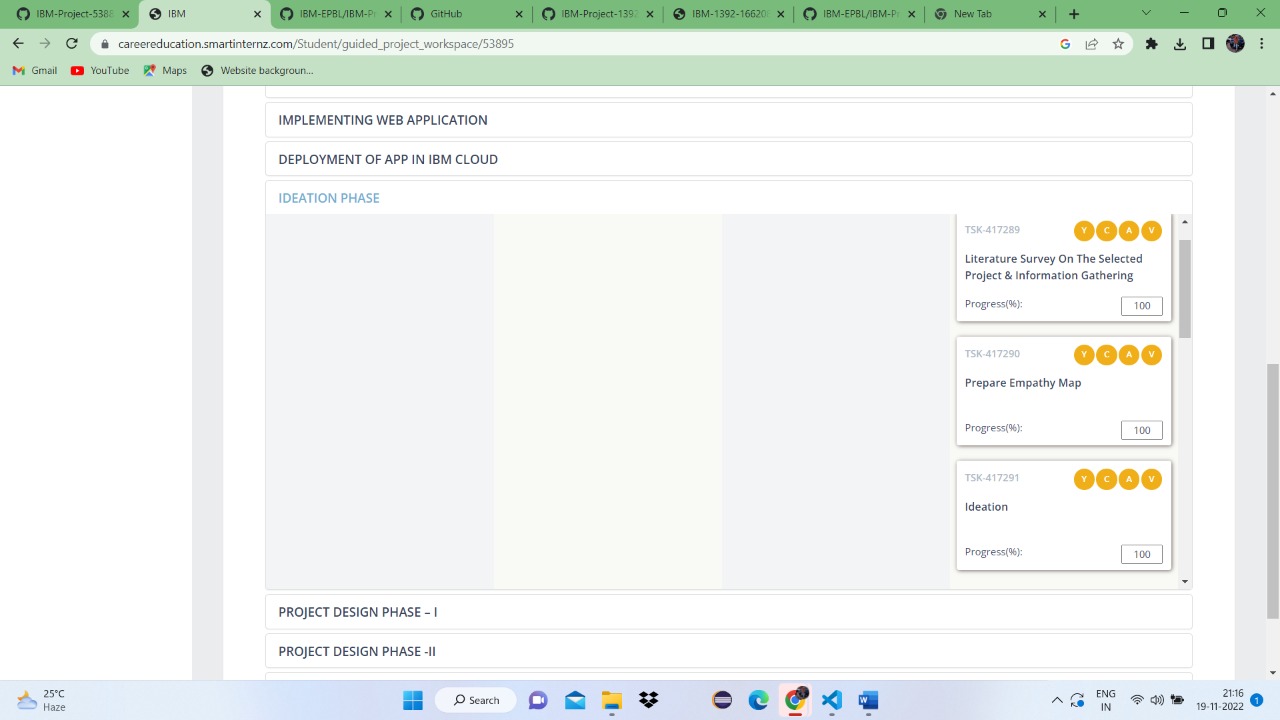












1. **CODING AND SOLUTIONS**

# 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>

</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%">

<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;}

/\* 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 \*/

@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>

<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>

</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);

}

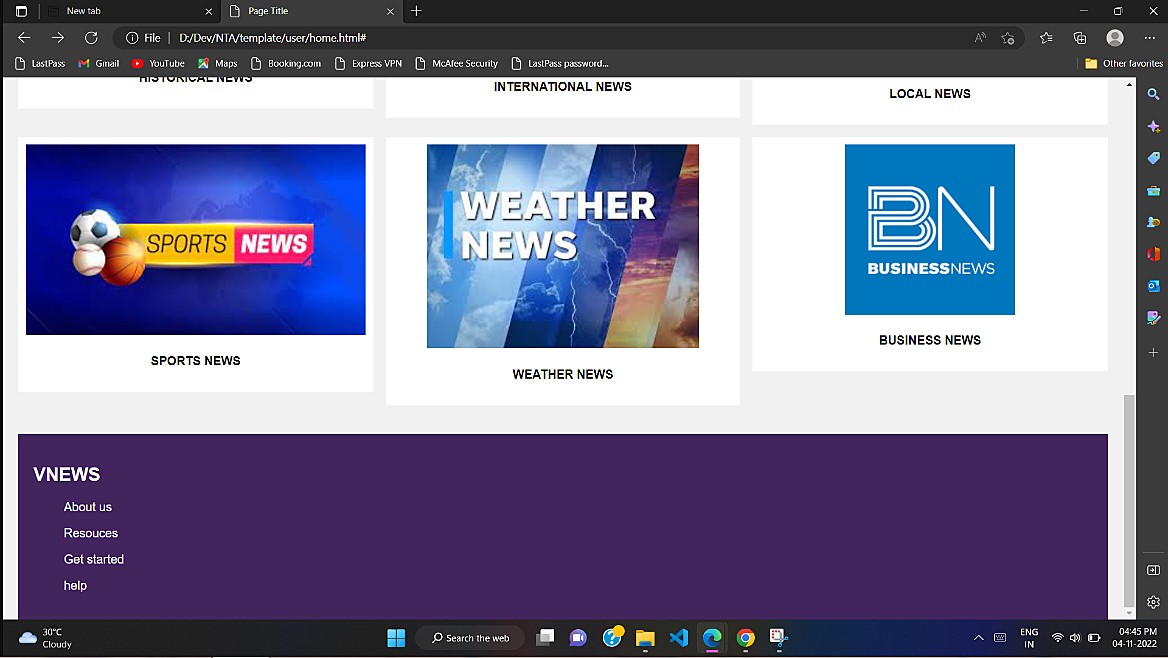
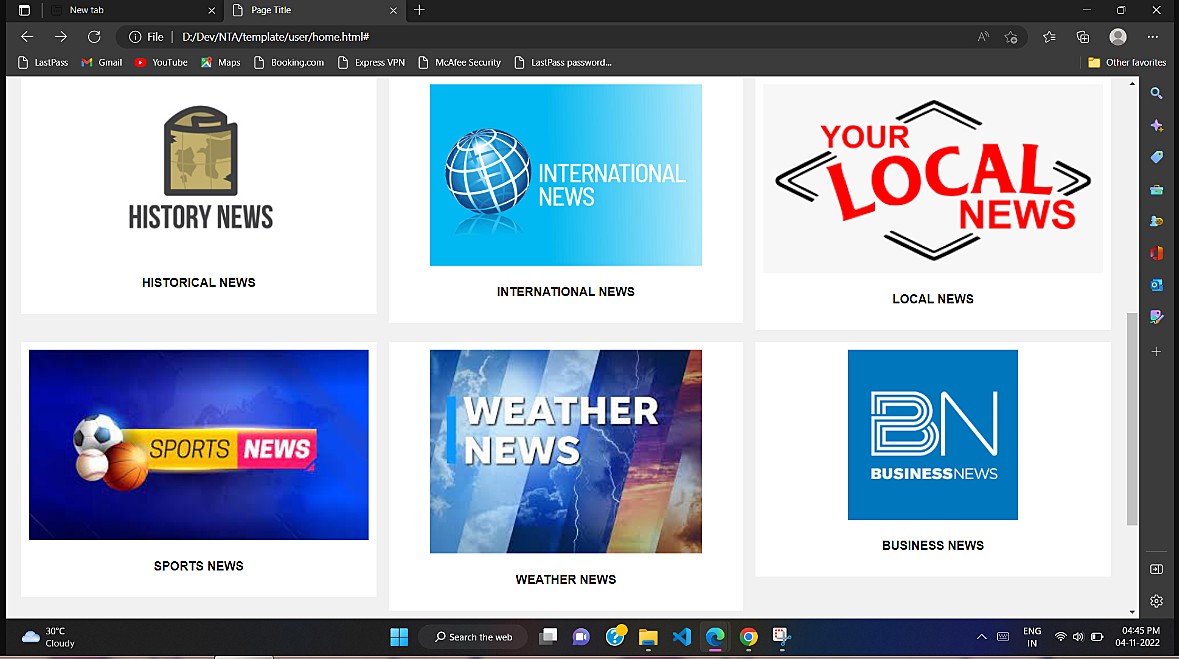
}

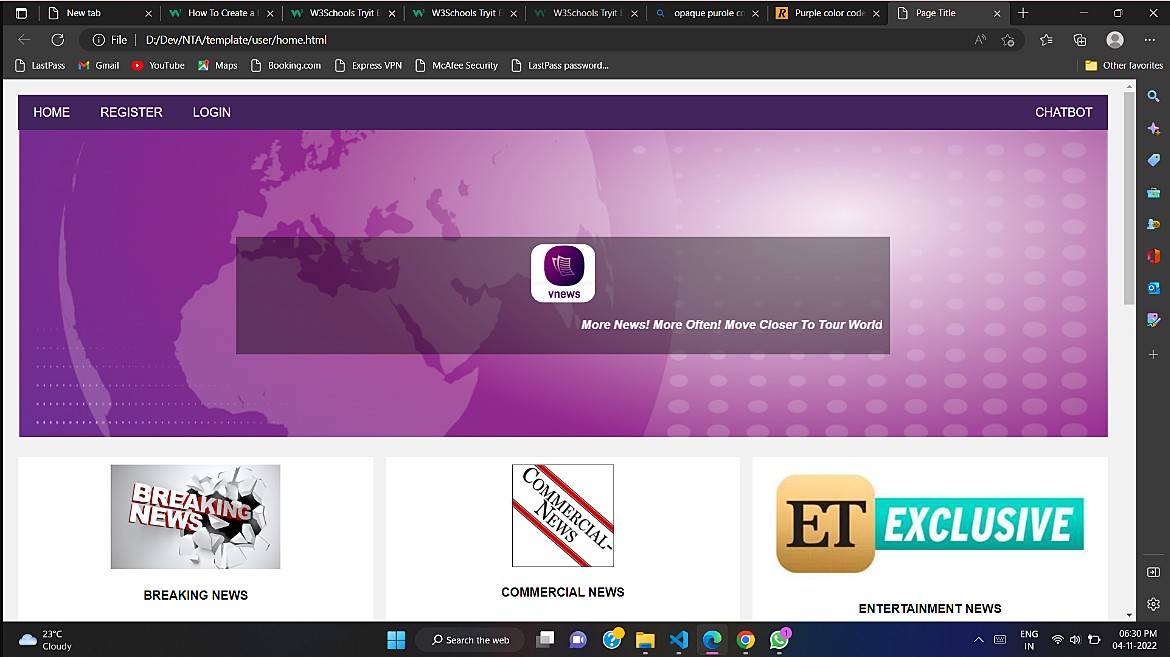
element.className = arr1.join(" ");

}

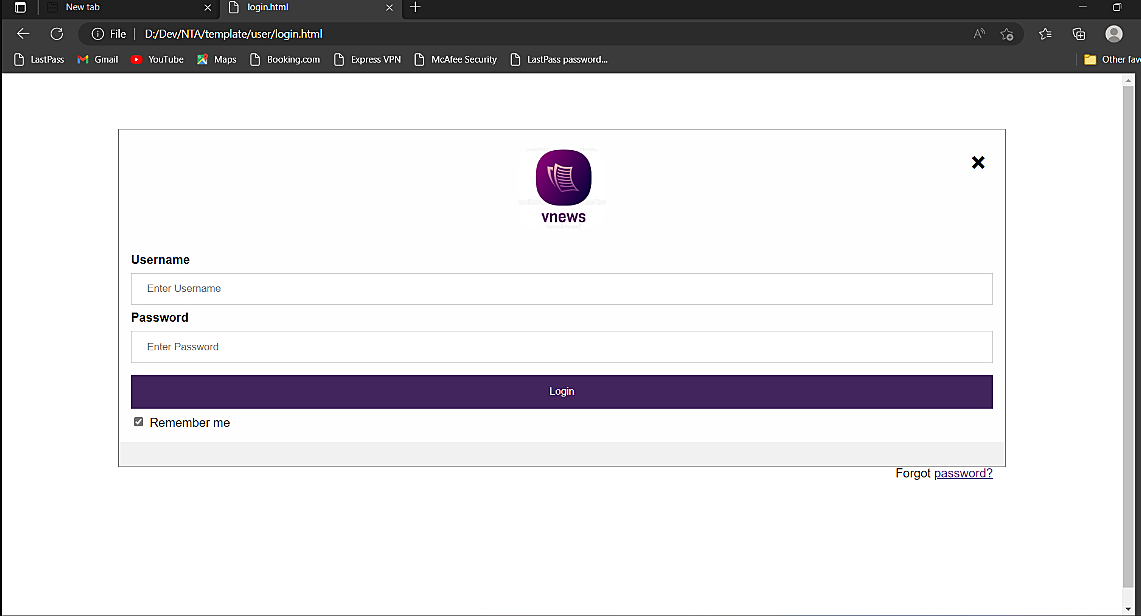
style.css

FIRST VIEW

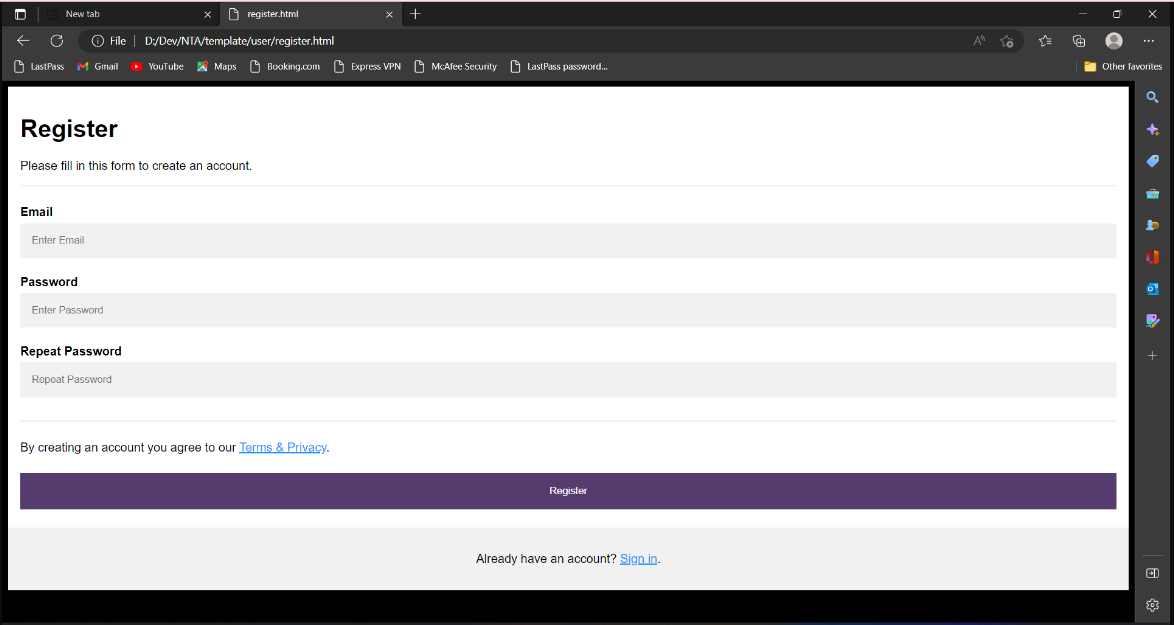




LOGIN



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"](mailto:test1@example.com),

"Example User"), To(["test2@example.com"](mailto:test2@example.com),

"Example User")]

mock\_pers['cc\_list'] = [To(["test3@example.com"](mailto:test3@example.com),

"Example User"), To(["test4@example.com"](mailto:test4@example.com),

"Example User")]

mock\_pers['bcc\_list'] = [To(["test5@example.com"](mailto:test5@example.com)),

To(["test6@example.com"](mailto: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")]

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')

message.reply\_to = ReplyTo('help\_reply@twilio.com', 'Twilio SendGrid Reply')

Content('text/calendar', 'Party Time!!'), Content('text/custom', 'Party Time 2!!')

|  |  |  |
| --- | --- | --- |
| 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 | = | [ |

]

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')

]

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"](mailto: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

<!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">

<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>

</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

@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)):

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 = []

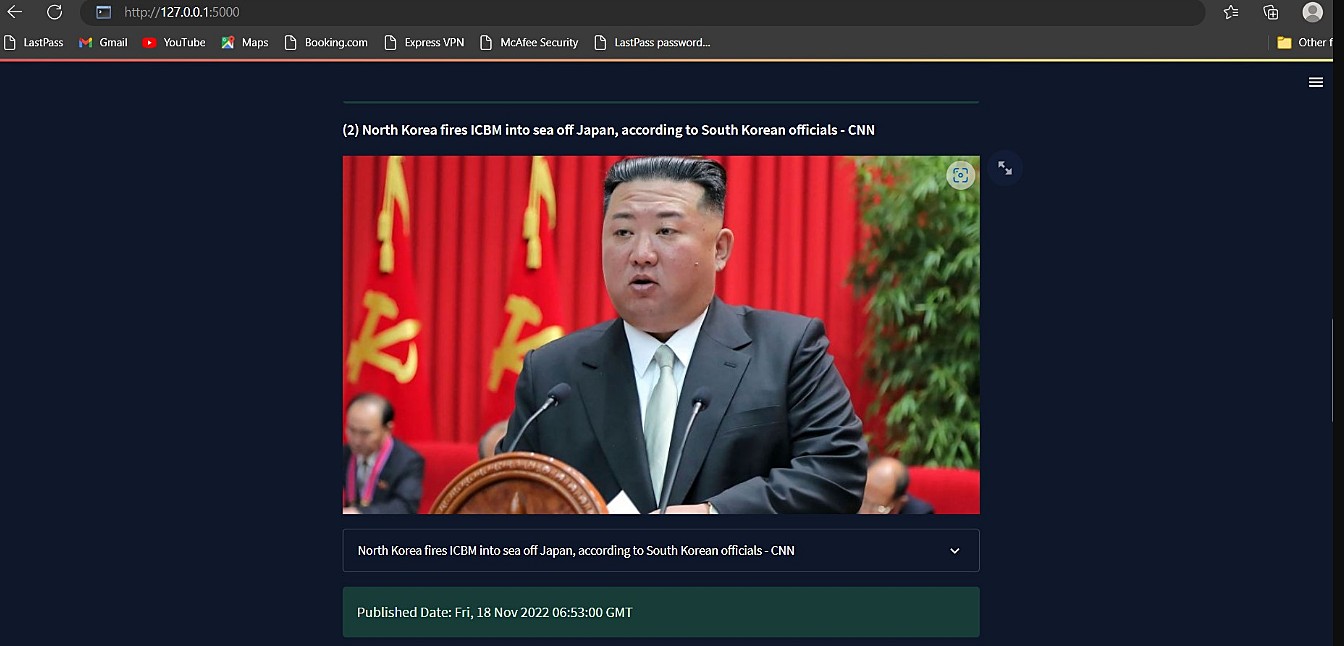
news = [] img = []

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)



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>

<!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>

<!DOCTYPE

html>

<html lang="en">

<head>

<title>china</title>

<meta charset="UTF- 8">

<body>china

</head>

</html>

<!DOCTYPE

html>

<html lang="en">

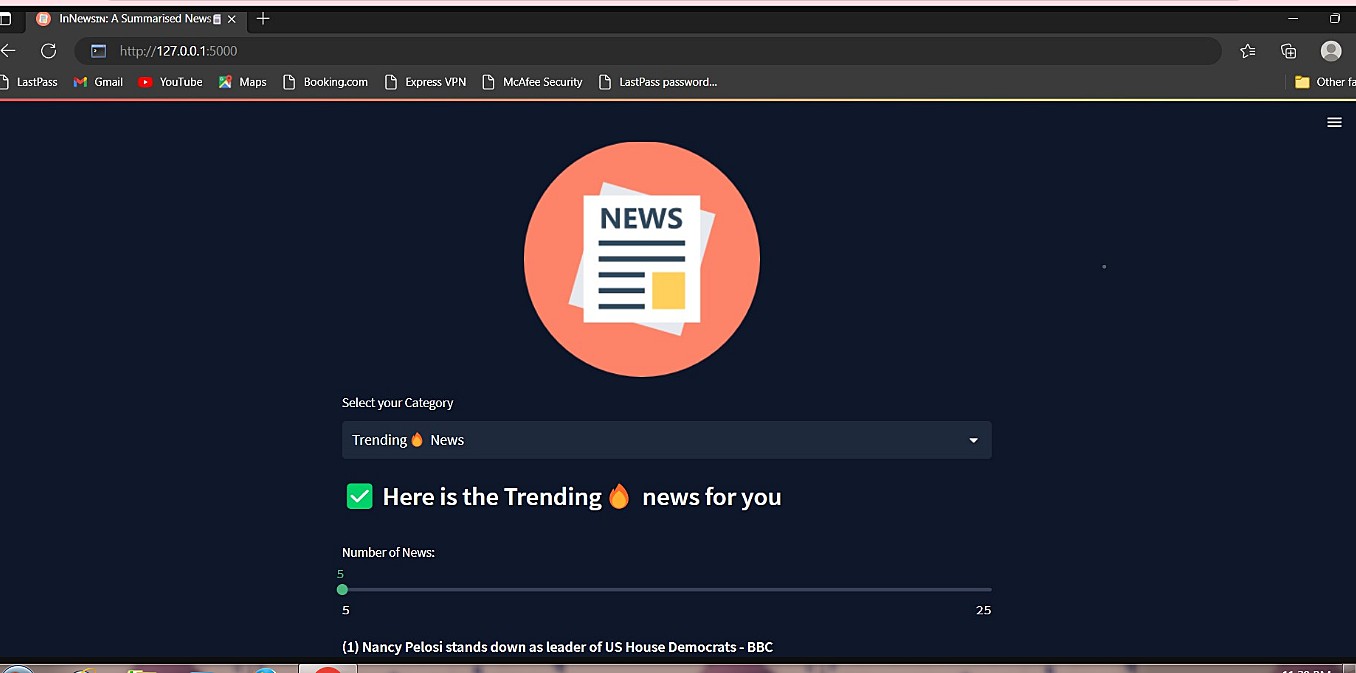
<head>

<title>china</title>

<meta charset="UTF- 8">

<body>china

</head>

</html>

# SPRINT 4

CONNECTING DB2

import ibm\_db

hostname="2d46b6b4-cbf6-40eb-bbce-

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)

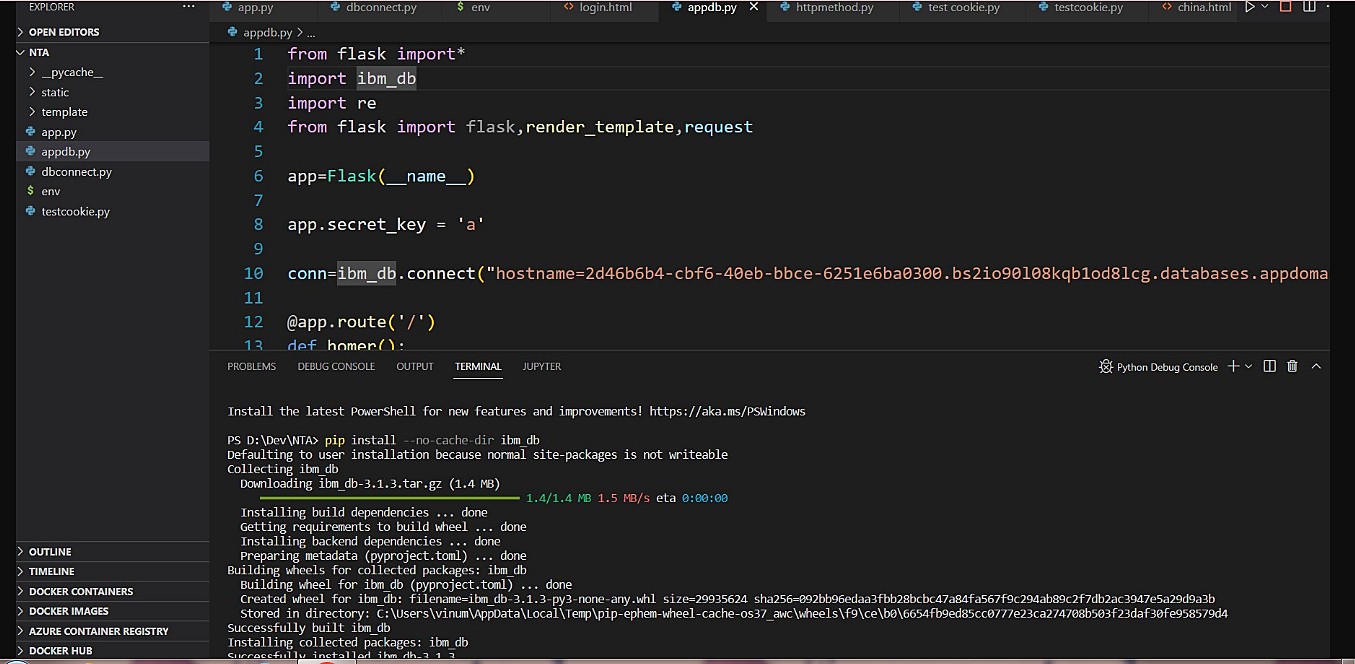
msg = 'You have successfully registered !' return render\_template('login.html', msg = msg)

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')



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)

# 8.TESTING

* 1. **TEST CASES**
  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.

# 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.

# 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.

# 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.

# 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.

# 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 exactly there 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 buy a 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 can result in a delay in purchase.

# 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>

<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">

<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>

</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%;

}

.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%;

}

.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>

<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>

</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;

}

.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-bo**x;**

}

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) {

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(" ");

}

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)

@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 = [] 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():

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('/')

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')

# GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-53894-1661517382

* 1. **VIDEO LINK:** **https://drive.google.com/file/d/1u2HlmZX5oZKKenvi-uTdD5-Gf2CWOdhT/view?usp=share\_link**