Personalized Buzzfeed

Submitted September 2021, in fulfillment of
the conditions for the award of the degree **MSc Advanced Computer Science.**

Sittukala Saravana Alagappan
199034381

Supervised by
Panneerselvam, John K. (Dr.) and Ulidowski, Irek (Dr.)

Department of Informatics
University of Leicester

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Date : 10/09/2021

Word Count : 10059

Abstract

Personalized Buzzfeed is a simple user-friendly news feed desktop application with responsive designs which is developed for the users to login and read news around the globe. The application is designed considering three major features which includes Personalization, Visualization and Recommendation. Personalization is more important in any web application to keep the users around. But Visualization helps them to feel interactive and keep engaged. Additionally Recommendation helps users to understand their preferences much better and explore the application with the suggestions. In this project, Personalization is approached with few ideas from both Contextual and Behavioral aspects and Visualization is represented using amcharts chart and map projections with Recommendations using the Apache Mahout Recommendation Model which is a machine-learning framework. So, the application is designed to develop all these features at its possibly the best first version which still can refined and enhanced in the future to a business product. The project development is aligned with SMART goals which makes it achieve the project standards set by the university at the first level and the industry in the future[19]. **SMART** is an acronym that means Specific, Measurable, Achievable, Realistic and Time-bound.

- **Specific** - The requirements were determined and documented in the preliminary report and was not deviating in the project.
- **Measurable** - The project progress were monitored periodically and tracked in the interim report which provided targets achieved and the tasks yet to achieve.
- **Achievable** - The interim report made clear the pending tasks and feasibility of achieving them as agreed.
- **Realistic** - The requirements had been done feasibility check by background analysis through research papers and going through conceptual ideas and made clear.
- **Time-bound** - Timely delivery of the reports and code components had been performed to meet the respective deadlines.

Achieving this overall goals to almost 80-90% helped the project go through various stages of development and testing efficiently and reach the desired results.

The project is evaluated through various types of testing. From the front end user perspective and achieving the stakeholders requirements, **Usability** testing was carried out and from the back-end to assess all the functionalities, **Test Driven Development** (TDD) which covers Junit testcases has been written and successfully passed all the testcases.

Acknowledgements

I would like to thank my Supervisors Dr.John Panneerselvam and Dr.Ulidowski, Irek for providing guidance and support throught the project. The ideas and approach provided by Professor John was very helpful to develop the application according to the software development cycle. And the inputs from Professor Irek during the meeting gave a clear picture of the how the project report should come up and also to do some enhancements in the application. And a sincere thanks to Dr. John Drake who organized our master project conducting up-to-date guidance sessions and cleared our queries throughout the dissertation.

Also, I would like to extend my acknowledge to my parents and my sister who constantly supported to achieve my goals throughout the university.

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Sittukala Saravanan

Date: 10/09/2021

Contents

List of Figures

Chapter 1

Introduction

Personalized Buzzfeed is a news aggregator that gathers news all around the world and present it to the users considering and analysing users personal interests making it more personalized application. This application enables the users to explore all around the global news with a click anywhere on the globe thus giving the users enhanced visualization look and feel.

1.1 Aims and Objectives

The main Aim of this project is to gather worldwide news and provide them to users across the countries knowing their interests and likes. The three key objectives of the application are observed as Personalization, Visualization and Recommendation.

1. **Personalization**

   Personalized Buzzfeed app provides the users, news and information of their personal interests and likes and also stays up-to-date on their interests by analysing their day-to-day behaviour on the application. So It stays unique to individuals preferences anytime.

   Definition for Personalization

   The adaptivity to individual users or user groups in a website is called personalization.

Web Personalization is defined as the process of tailoring the content of a website to suit the user's needs and interests by making use of their behavioral activities in the application and their contextual details[9].

2. **Visualization**

   Visualization is apparently achieved in this application providing a more appealing UI for users enhancing the user-friendliness. Visualization is provided in a more sophisticated way to the users in the form of more **interactive charts** and **geographical map** which is well achieved using amcharts library.

3. **Recommendation**

   Recommendation is key feature which helps the users to handle increasing number of items or articles in case of news feed with the personalized suggestions provided by the application based on their preference patterns. According to research there are many recent development of recommendation techniques understanding its vital role. So, this application too integrates one of the Machine learning framework called Apache Mahout for recommendation system[17].

Evaluation

The project is evaluated to identify and resolve the bugs and make desired flawless results. Two types of testing was performed - one at the user level which is called **Usability testing** or **Observation Testing**. And the other at the back-end level to ensure the logical functionalities using the industry based testing approach called the TDD - **Test Driven Development**. Both the test approaches are useful in figuring out the errors and help them locate and resolve more effectively.

Report Structure

The report follows the standard template which covers the **introduction** and **motivation** which overall explains the project idea and the reasons behind doing this project.

Then the **literature survey** which provides evidences of existing news applications and the **proposed system** which shows some adds-on to the existing ones came across the literature survey.

The in-depth report structure starts with the **requirements gathering** section which explains the stakeholders different levels of requirements and **technical specifications** gives enough idea on the development technology.

Going forward the project **design phase** gives the ideas and concepts for developing the application with notes on choice of various technologies or concepts used and the **implementation phase** with brief inputs on the code structure and technical solutions applied. To make it complete, the project explains the testing strategies applied and the results and fixes happened in the **evaluation** section. Additionally the report provides the **challenges faced** in development process and observed challenges for the future enhancements. Due to time constraints now, the report proposes possible enhancements that could be made in the future and make the application transform to a business product in the market. Finally the report summarises the content by reflecting the **overall achievement** of the project.

1.2 Motivation

- The main motivation in developing the personalized buzzfeed application is to gather various news data across the countries for its end users. The application is developed keeping in mind of two major category of stakeholders of this system. The **one** being the general casual news readers who are interested in various category news data. The **second** user-group being journalists, news companies or any organization who are interested in knowing popular category-wise locality data to better study the statistics of news by country or category.

- The other main motivation of this application is that to get a better understanding of how web personalization is being automated and achieved instead of one-size-fits-all approach. **Personalization** is now the **key-role** in promoting any kind of websites and business to its end users and knowing this standard way of automation would benefit in the long-run learning and development. This system has implemented well-formed personalization effectively achieved through two dimensions of web-personalization.

Chapter 2

Literature Review

2.1 Background study and Traditional applications

The traditional systems currently in the market are providing various news data like popular ones and breaking news for its readers. The increasing number of mobile phones contribute to large number of news readers in mobile phones and tablets. To maintain this growing population towards news-reading and to meet their expectations there is an excessive need of adaptive personalization to suit individual taste and interests which is limited in traditional systems.

Several applications in the marketplace were reviewed before giving a kick start for the project. For Example, **BBC News app** personalization works in a way letting the user to customize the ares of interests and populate '**My News**' page accordingly. It works more towards user-customization but lacks adaptability and presents outdated content if there aren't any recent news which makes it bit awful [10].

Others work in a way aggregating news from various sources which are knows as 'News Aggregators'. One such example is **Flipboard** which works in a way allowing users to pick their favorite stories from various sources or categories and populate most relevant content. Additionally it has Machine learning algorithms which learns from users interactions like follow, flip and heart and populate contents based on the high interactions[3].

And there is an observation made on **Google News** personalization which works considering two approaches.

- Taking users past click logs
- Building user profiles from personal preferences and interests [16].

Also, according to recent research, about **60 percent** of the applications follow hybrid recommendation system [8].The hybrid recommendation system refers to the efficient combination of content based filtering and collaborative filtering approaches.This enhances the accuracy of the recommendations quite well[12].

Experimental Investigation

And in my research and analysis I came across one brief experimental analysis on my current work which gave me more clear picture what is the exact problem situation and why its more important. According to the experimental investigation, while a gigantic and enormous amount of news gets released every hour, it creates a challenging situation to gather all the news from the right source and recommend the right news to the users that too match their reading preference as much as possible.[14]. This issue is termed Personalized News Recommendation by the authors. This paper also insisted we get huge resources of news information across the world through the internet and the recommendation system has to wisely recommend the news for the user's based on user profile and news content to the readers which is why this issue is more important to consider. This research study has provided detailed understanding of existing personal news recommendation systems and helped to identify the importance and enhancements to pick from there. The research work explains the different methods of recommenders which are Content-based Methods, Collaborative filtering and the hybrid recommendations which helped in understanding different approaches and choose the right-most one.

User Profiling

Another major challenge is to construct the user profile. The user profiling is important to provide the preferred news articles to the readers from a wide range of sources. One of the research papers which explains briefly on User Profiling for News personalization. This paper follows an approach for user profile building by following the entities the user included in the comments[20]. These entities are then extracted from each news article for constructing the user profile and recommending similar articles which the user has commented. This helped in getting more clear on the concepts of user profile and what are different ways to build them for each user. This gives an understanding of using comments which are some sort of user interactions and recommend news articles based on that.

Location Preference

There is a research study which states that adding location preference to the news recommendation system helps in increasing the customer satisfaction from 8.6 to 9.4. The combination of user profiling, popularity and location improves the customer satisfaction and enhances the application to a desired level[22].

Theoretical Architecture of News Recommendation System

A research paper proposes the overview of the structure of recommendation system and the overall functioning elements.
There are three major components in the system which are the user component, correlation table and the ranking filter.

- **User Component:**

  The user component comprises the user attributes characteristics, user behaviour characteristics. User Attribute characteristics as stated in the paper indicates the user demographic elements.User behavior characteristics are of two types as stated by the author which are implicit and explicit types. The explicit types are the ones which are tracked from direct user interactions and the implicit types are derived from the browser history and so on[15].

- **Correlation Table:** The user information along with the collaborative filtering outputs like Item-Item, User-Item, User-User correlation information forms the correlation table and provides the input for the recommendation filter.

- Ranking algorithm: The final step is to provide the filtered set of recommendation results going through optimization removing all the blacklists and providing the refined list of news articles for the readers. This is processed through some specific ranking algorithm and finally gives the desired results.

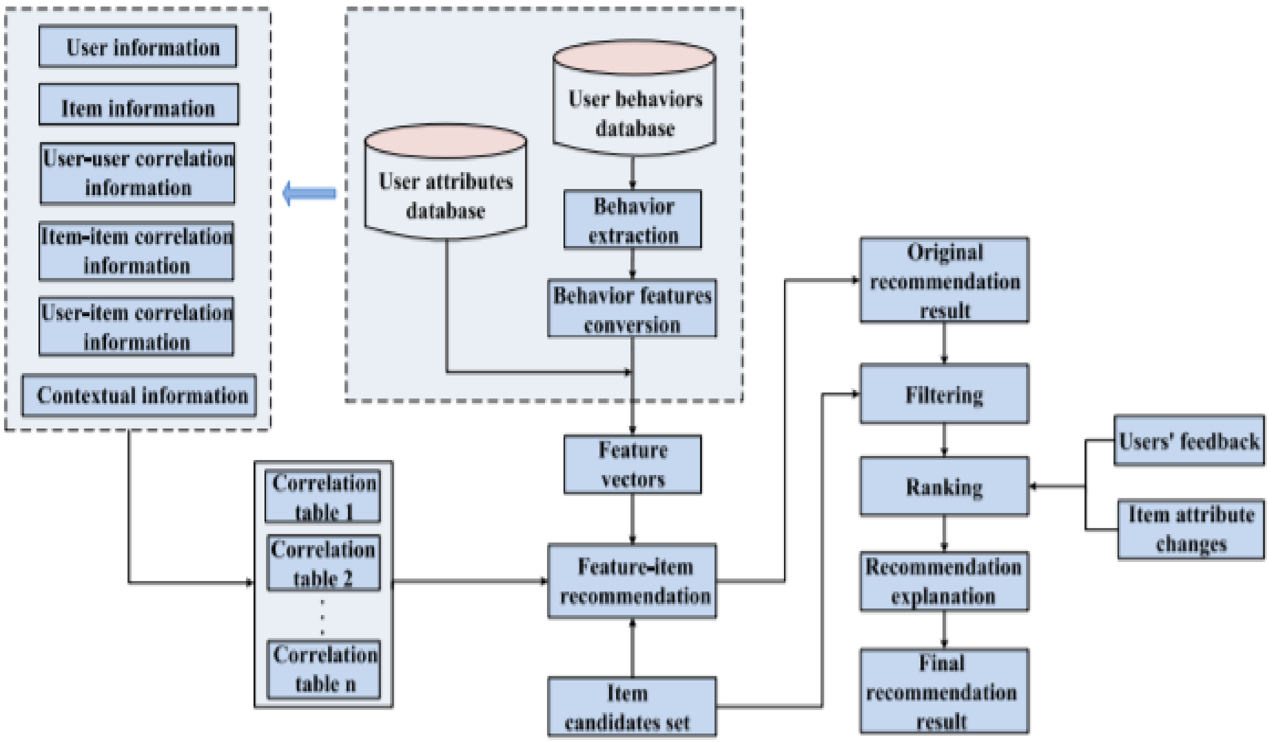

FIGURE 1. **The overall framework of personalized news recommendation.**

Figure 2.1: System Architecture

Recommendation Model

The overall system design is supposed to work in a workflow model basis which is described as below. Through effective analysis of user information with context details both explicit and implicit, user profile model is created and with the approaches of several types of collaboration filtering or the hybrid approach correlation model is created.Based on the combination of both, user prediction model is developed which recommends the right set of news articles to the readers following the optimized ranking algorithm along with the process.

The overall process flow model is represented as below.

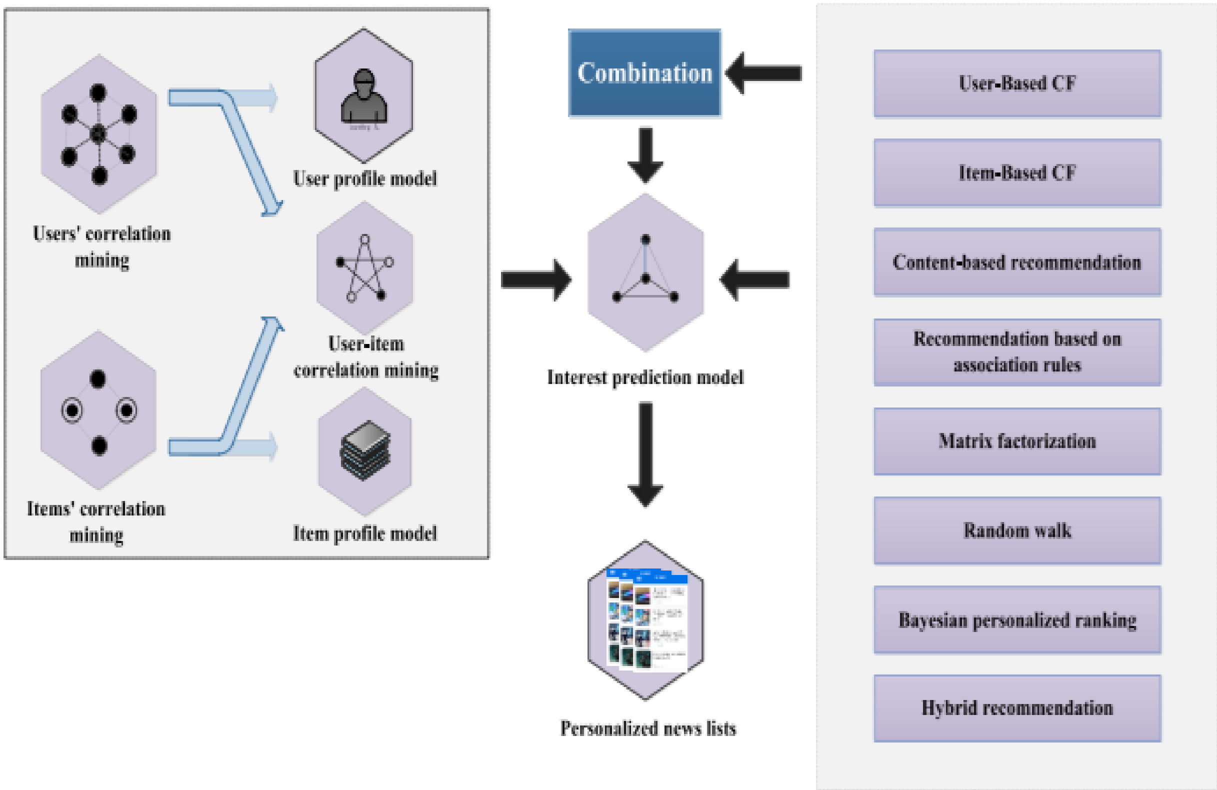

Figure 2.2: System Model

The literature review was done on various angles and collected information to include as many enhancements and features for the proposed system. **Missing features on existing applications**

The applications in the marketplace shows some lag in few features which are considered in our proposed system.

1. Lack of Visual Evidence of Users Interests

While such applications already exists in the market which learns from users context details like interested categories and interactions, there is a no explicit and user-friendly visualization of what the users favorite zone and order of interests and **numeric comparisons** of user interests. There is no clear picture of what the users interests are after a many interactions and the system starts to overload the content making it difficult for users to understand what their major interests are over a period of time.

2. Overloaded feeds

The recommendations and the personal stories are overloaded in the apps sometimes making it a noise. This in-turn affects personalization and leads to criticism if the algorithm is not considering best relationship factors between the users or the relevancy score.

3. Minimal or No Scope on Visualization

The present news app provides more data with but less ideal for users who look for more friendly and appealing visual tools.

2.2 Proposed System

The proposed system works in a way fulfilling three major targets of the application.

1. Personalization
2. Visualization
3. Recommendation

This all-in-one system helps enhancing the news app one step further making it more user-friendly and easy to use.

1. Personalization

Personalization is implemented in this Buzzfeed application with two possible approaches as shown in the figure.

(a) Contextual Personalization

Contextual personalization is the most widely adopted model where the

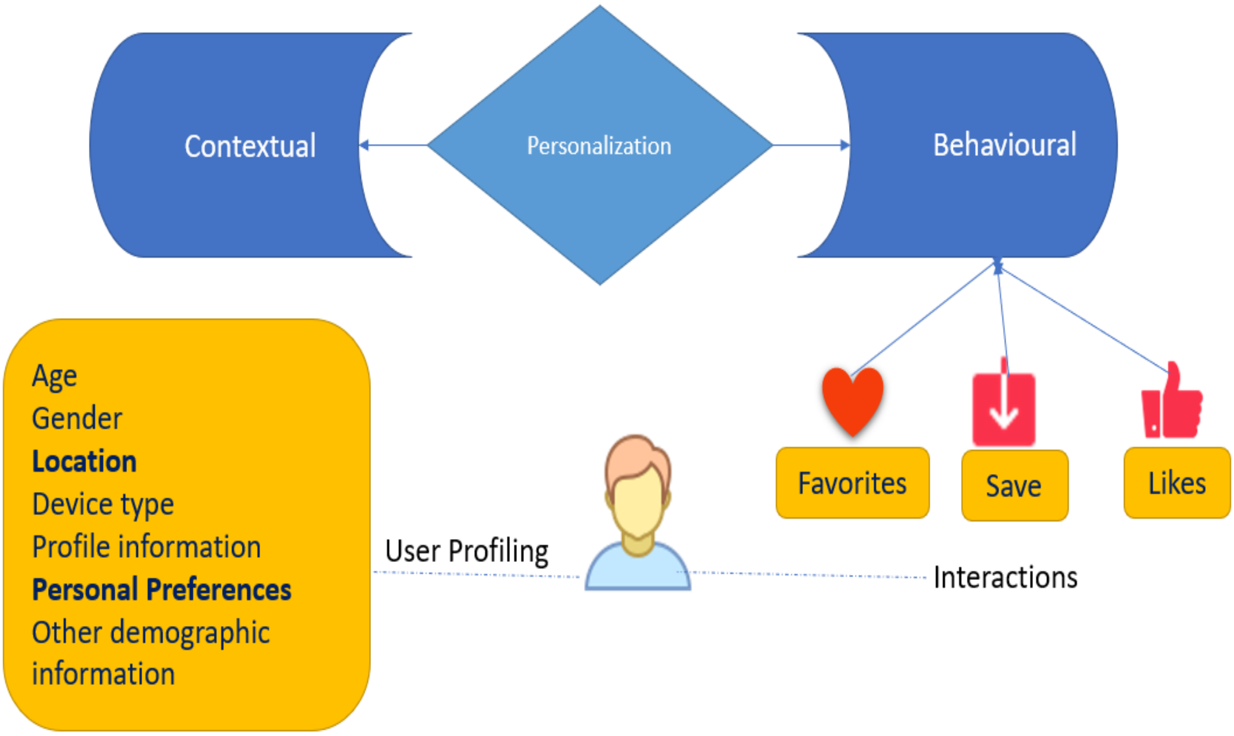

Figure 2.3: Personalization

user profiling is considered for customization and recommendation. The user profile is gathered and analysed thoroughly thus understanding individual selected preferences and several other factors expressed during the signup process for creating his/her profile.

It generally comprises several factors including:

- Age
- **Location**
- **Selected Preferences**
- Gender information
- Demographic information [5]. Personal Buzzfeed considered two of the above Location and Selected Preferences and works with the personal feed.

(b) Behavioral Personalization

Behavioral Personalization is measuring user's every interaction and moves in the application to predict their interests and personalize their feed with most likely items. This is considered more important as it determines the users trending mindset and gets updated with their moods to adapt to their current preferences. This is assessed in our application considering three major interactions for loading their personal feed.

- **Likes** with a thumbs-up icon
- **Favorites** with a heart icon
- **Save** with a download icon

How It All Works

Weight-Map Algorithm

For Personalization, the application works out an algorithm called '**Weight-Map**' Algorithm which takes all combination of Selected preferences from Contextual and other interaction based preferences and sorts the personal feed to present the users with top likes news articles.

User Location

User location is considered separately and loaded initially with the Global data component and list down the news of his/her location with option to switch to other countries and any categories.

2. Visualization

Visualization is implemented using amcharts which is a library that was integrated into the frontend for accomplishing geographical maps and pie charts.

Why Amcharts?

Amcharts is chosen for integration as it has good projection of advanced charts and also supports plugins for maps with well-defined projections and compatible with the existing front-end technologies design and development [2].

Where in Buzzfeed?

This is implemented in two primary areas in our application.

- **Map View** The amcharts projects globe with all country geodata and initially it loads the user's location data on getting to this component in the frontend with possible option to see any kind of category the user likes to know in this location. The globe is clickable and allows to switch to any country and loads respective news based on selected category.

- **User charts** The amcharts makes it possible to project any kind of data having advanced chart types. Each user interests are mapped with respective preference weights associated for various category items in the news feed an they are represented in terms of pie-chrts to let users know explicitly what their interests and zones are.

3. Recommendation

Recommendation is another area where the users interests are analysed and compared with other similar users and suggested some categories which they might be interested in. They are a kind of Machine learning algorithms which learns the users and provide relevant and optimal recommendations [21]. This is accomplished using **Apache Mahout Recommendation System**.

There are many types of recommendations possible like popularity-based recommendations, Content-based recommendations, Collaborative recommendation systems,etc

(a) **Content-based Recommendations:** Content-based Recommender system recommends the things or similar items which are most liked by the user. It observes the users like patterns and search history and recommends similar items to the users.

(b) **Collaborative Filtering:** Collaborative filtering is the highly beneficial and effective methods of recommendation system being implemented and provided high success rates in many companies. For example, Amazon recommendation

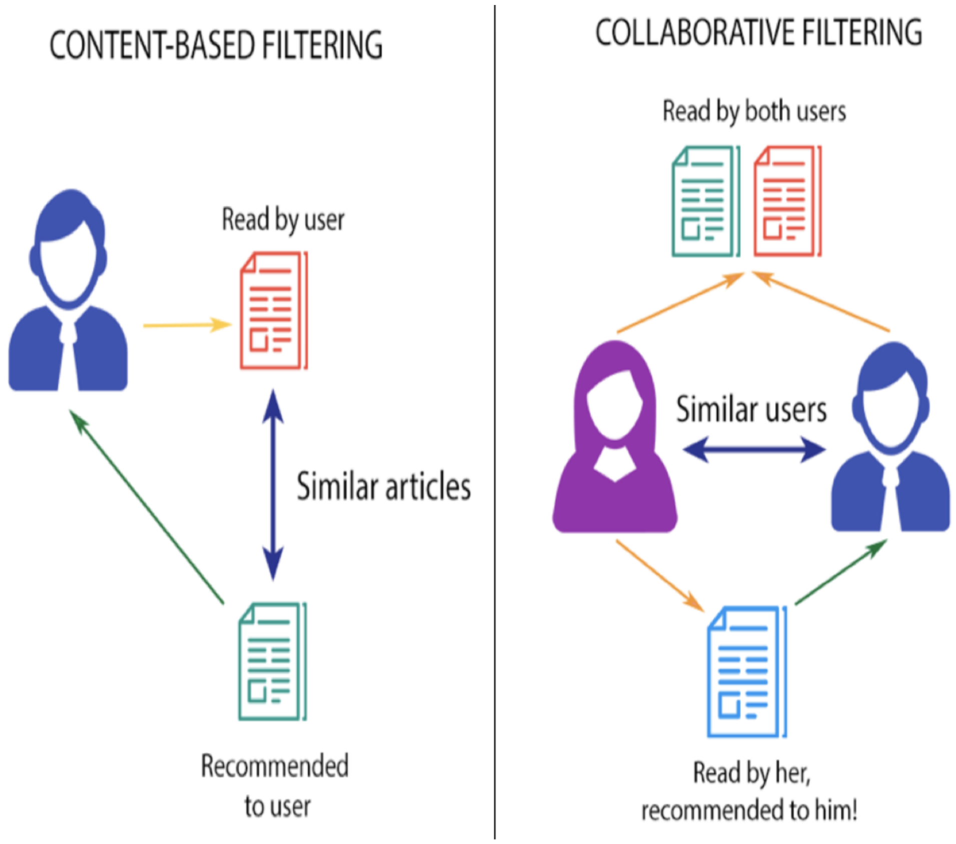

Figure 2.4: Recommendation Types

system which suggests on selecting a product, a pair of other products being bought together or similar products bought by other users which has increased sales by 29%. Other similar success use cases with Collaborative filtering are Netflix movie rentals by 60% and Google news click-through rates increased by 30.9% [13].

Reason for Opting Collaborative Filtering

So, similar seeing increase in success rates on recommendations with collaborative filtering, this application too incorporates the Collaborative recommendation system comparing similar interests of users and recommend news articles.

Why Apache Mahout?

Apache Mahout is a popular Apache-licensed open-source library for scalable machine learning algorithms [28]. Mahout provides most useful frameworks for Collaborative Filtering implementation. It has in-built framework, functionalities and

APIs for machine learning algorithms to achieve the appropriate recommendations to users.

Chapter 3

Requirements Analysis

3.1 Top Level Requirements:

1. **Gather News Feed:** The application must gather news articles around the world for the users. This is planned to be accomplished through querying News Api which has categories and country fields and updating database with sample data to make it reliable for application.

2. **Personalization:** The application is mainly focused on delivering news based on user personalization aspects. For achieving this requirement, the application targets two main concepts[5] which are Contextual and Behavioral Personalization. Application is expected to develop Contextual personalization by building location-specific news and user-preferred categories news data in their personal feed. Also, in terms of behavioral the application is targeted to achieve personalization based on user interactions which includes the likes, save or add to favorites. Additionally, the application has to develop a way to recommend news articles to the users for which Mahout Recommendation system is considered for analysis.

3. **Visualization:** The application is aimed to have more interactive views for the user where the amcharts map for the location filter and charts for projecting user category-specific interest rate are desired to be achieved.

3.2 Detailed Requirements

Essential Requirements

1. **User Authentication:** The user registration should be feasible and the registered users should be able to login with a valid token.

2. **News Feed:** The application has to gather news from a well-defined api which has all essential fields of our news model.

3. **Database:** The news api has to be queried and updated to database with sample data for reliability of application especially during presentation to avoid any issues.

4. **Personalization:** The main feature of personalization to be well-designed and also updating for changing user's interests.

Recommended Requirements

1. **User Charts:** The application should be designed to project user category chart which shows percentage of interests for various news categories.

2. **User Profile:** There should have options for users to view their profile anytime in the application view.

3. **Visualization:** The application has to have a globe view to help users to choose any country to view the location-specific data and filter through categories if applicable.

Optional Requirements:

1. **Category Newsfeed:** The application has to load separate category data into various view for easy access.

2. **Saved/ Favorite feed:** The application could have the option to show users saved and favorite feeds for them to revisit anytime.

3. **Update Profile:** The option to update the profile details for the users could be better.

4. **Recommendation System:** The application is desired to have the recommendation of news articles for users based on their interest pattern.

3.3 Technical Specification

Specifications	Software	Version	Description
Back-end Framework	Java	8	Used for efficient back-end.
	Spring boot	2.0	Provides supporting framework with in-built dependencies.
Front-end Framework	Angular	12.0	Front-end framework which aids component-based application design.
	HTML	5	Used to build individual components.
	CSS	4	Better styling.
	Bootstrap	4.5	Provides built-in designs and development
	JavaScript	ES6	Validation support
Query Language	SQL	Language support	Performing queries from database
Database Application	H2 - RDBMS	1.4	To store the application data and ability for having either in-memory or remote accessible database through url.
Repository	JPA	2.0	Access data from database and perform CRUD operations on the data
Web Services	Rest APIs		To post Http requests and receive http responses for transferring data.
Testing	Postman	5.5	To test API end-point.
Version Control	SVN Repository		To manage the code and documents during the entire project without any conflicts

Chapter 4

Design

The application comprises the system design, architecture adopted, approach followed and the database used with a brief justification on why and how they are incorporated.

4.1 Approach

The Approach followed for developing this application from scratch is a step-by-step practise which are as follows:

- Wireframe Diagrams
- H2 Database Installation
- Initial Springboot setup
- Installed Angular modules and implemented design
- Agile approach

Let's see how it evolved one-by-one and the reason for adopting the model.

1. **Wireframe Diagrams**

   Wireframe diagrams act as a key step to begin any web development and get consent from the clients regarding the requirements on how it might look on screen. So it is considered more vital for any development process. Wireframes provides the layout for the final product and it acts as a rough sketch for the application[24]. So, I have completed the requirements analysis phase and then designed the wireframe diagrams for getting approval from the Professor for proceeding with the initial designs and ideas which further evolved on development.

So, here are some of the screenshots of wireframe diagrams in the intial phase of design.

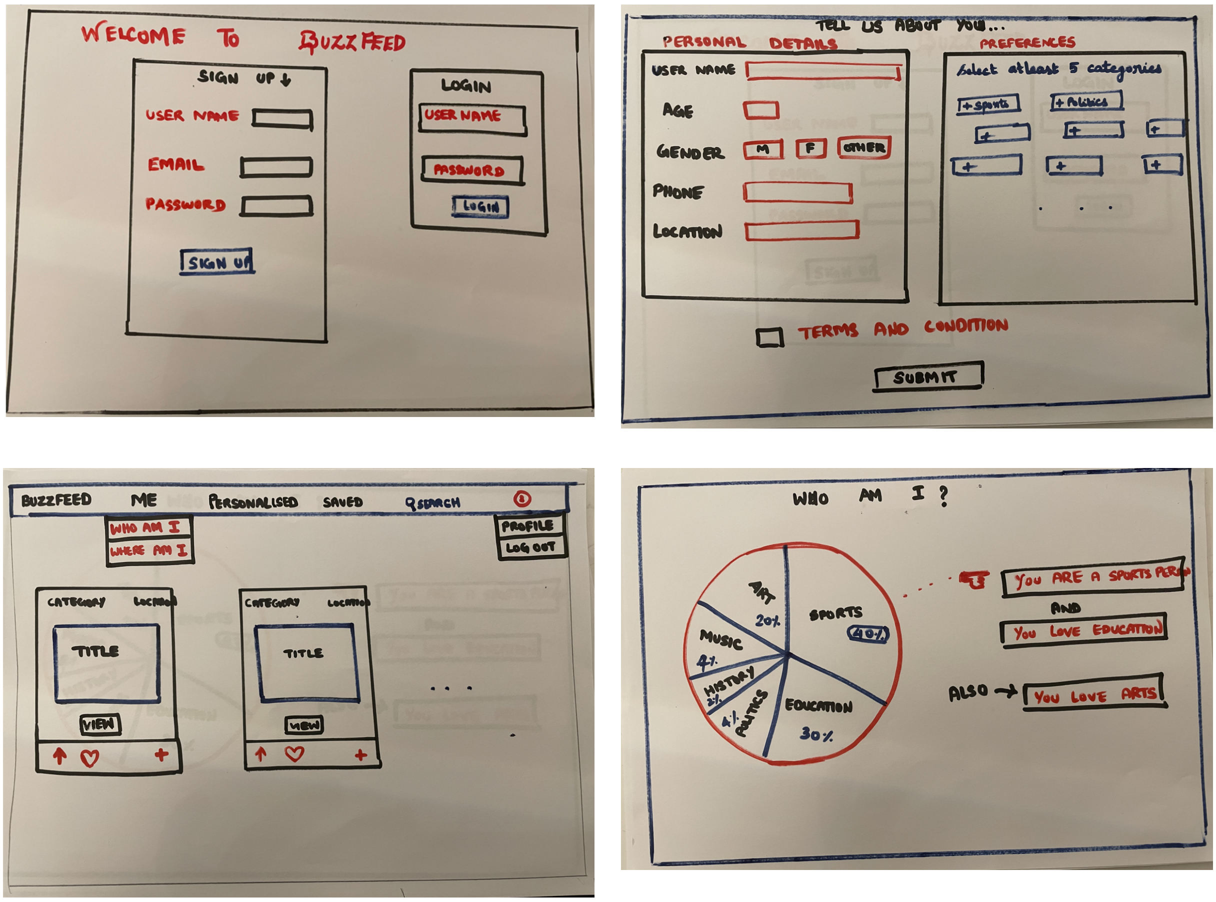

Figure 4.1: Wireframe diagrams

2. **H2 Database**

   H2 is an open-source database written in java and hence it makes a very good combination with java springboot.

   Features of H2 Database

   H2 Database has the features compared against other databases which makes it

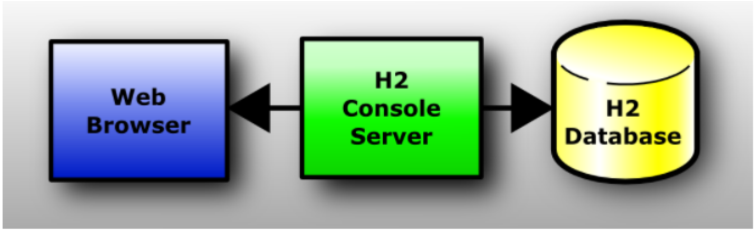

Figure 4.2: H2 Database

more preferable to use. The major reasons H2 database is used in our application

Features

	H2	Derby	HSQLDB	MySQL	PostgreSQL
Pure Java	Yes	Yes	Yes	No	No
Memory Mode	Yes	Yes	Yes	No	No
Encrypted Database	Yes	Yes	Yes	No	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	No	No	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Footprint (embedded)	~2 MB	~3 MB	~1.5 MB	—	—
Footprint (client)	~500 KB	~600 KB	~1.5 MB	~1 MB	~700 KB

Figure 4.3: H2 Features

are:

- **Java**: It is a Java open source database compatible with Springboot.
- **Embedded Persistent Database:** It supports several modes like embedded, in-memory and in both the modes it has persistent and in-memory databases. So, if we don't require to persist any changes to the database we go for in-memory or otherwise embedded database. This application works with Em-

bedded Persistent database as there might be requirement for users to anytime update their details in their profile.

- **Encrypted security:** It has Encrypted security which makes the data secured.
- **H2 Console:** Additionally it has an option of H2 console which provides browser based access to the database.

3. **Springboot Setup**

   Springboot provides all the java application framework ready to start the development process. It comes with all the packages as separate dependencies which can be added anytime into the pom.xml which inturn builds the Maven and makes the dependencies available to use.

   Once the project is created with Java springboot the next step is to create Controller, Repositories and the Service class which makes the application run with respect to the Multi-layered Architecture.

4. **Angular modules**

   Angular npm and node.js are installed and the project is started from scratch angular CLI command. Angular is chosen because it supports code reusability, framework that supports component based development and supports many enhanced integration for appealing front-end UI components that can be used.

   Once the angular modules are installed the designs are implemented having wireframe diagrams as a reference.

5. **Agile approach**

   The application is developed following agile standards which works very well with short-term standard software development. It is a results-focused development process which works adapting to the rapidly changing requirements.

There are many agile methodologies in software development process out of which the following are adopted in our application development cycle.

- **Working product:** The sprints are planned weekly and anytime the application is delivered or made available with a minimum viable functionalities ready to deliver.
- **Client Collaborations:** Meetings happened with the Professor and developer throughout the development.
- **Responding to Change:** The changing or updated requirements from Professors are considered and developed regularly to develop a complete product.

4.2 Spring Boot Architecture

Springboot works in a layered architecture consisting of four different layers communicating directly above or below them. This can be run easily with the embedded tomcat server.

- **Presentation layer:** This is the front-end view where the application presents the features to the users and handles the http requests transferring to the business layer.
- **Business layer:** This layer performs the business logic and handles the authentication and validation of application.
- **Data Access layer:** This layer plays a key role in handling data to and from the databases which has a repository to do so.
- **Database layer:** This is where the application data resides and all the operations like CRUD are performed[29].

Architecture Flow:

A Springboot application has a controller which process the http requests received from the client and it then interacts with the service layer where it does all the business logic for the application and data access or update using the repositories of the model [26].

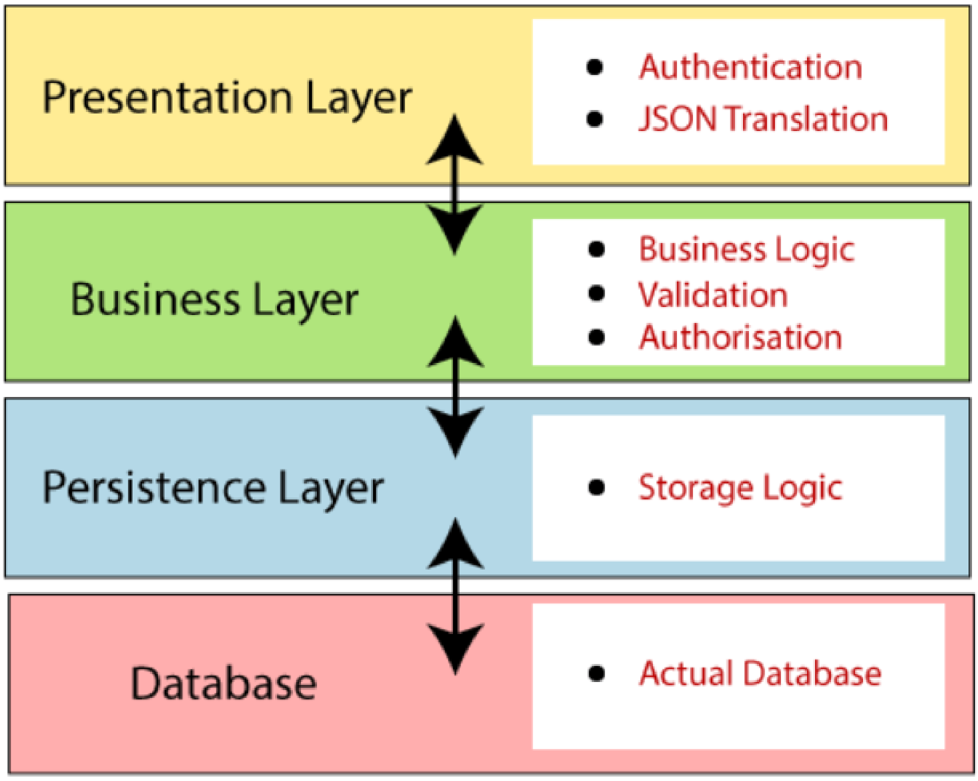

Figure 4.4: Springboot Architecture

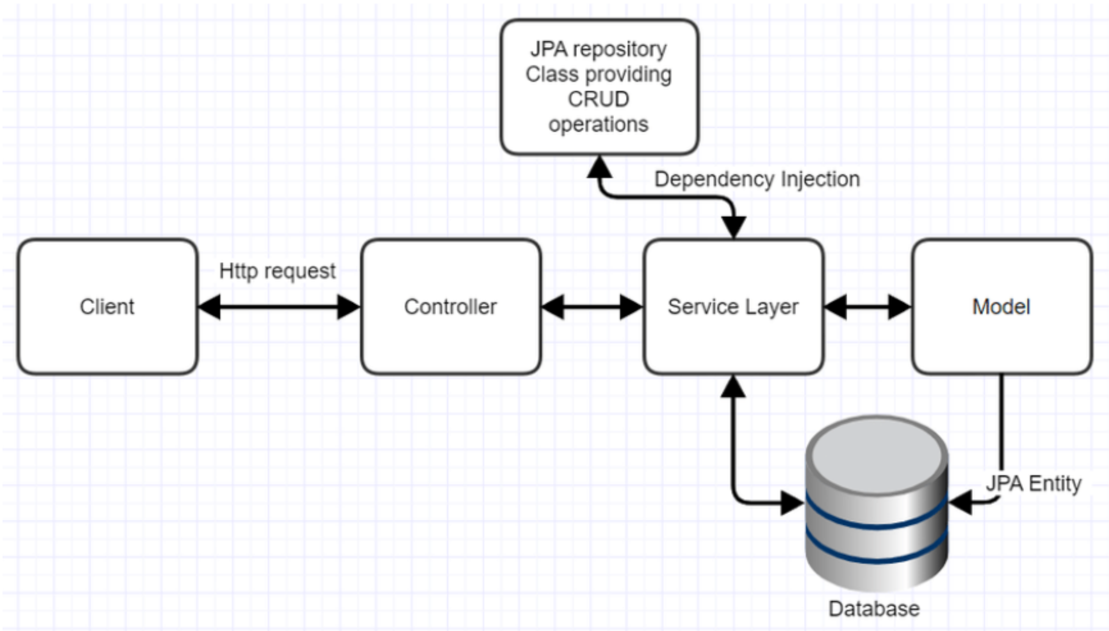

Figure 4.5: Architecture Flow

How It works in our Buzzfeed: Sample architecture flow in our application is represented as follows:

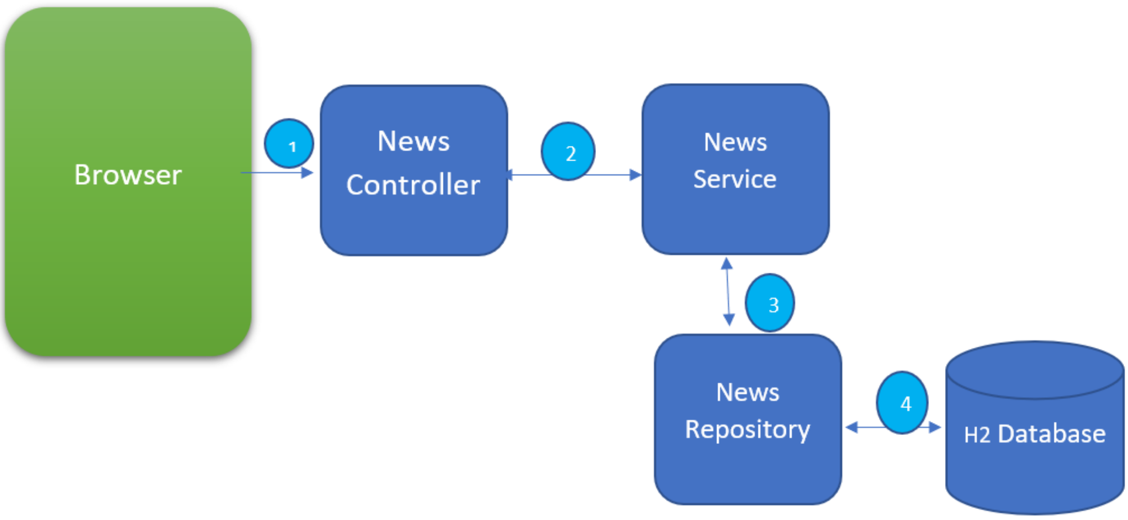

Figure 4.6: NewsFeed Server Flow

1. Springboot News Controller receives the http requests from the frontend.

2. The News Controller then process the requests to the News Service class which handles all the business logic and actions.

3. The News Service class then process the data from the database or updates the data to the database through the database access layer called New Repository.

4. The H2 Database stores the data and handles the data storage and updates if any. It is embedded Persistent database in our application so it can effectively persist data in case of changes and it is immediately available.

4.3 Object Oriented Design

4.3.1 Use Case Diagram

Use Case diagrams is the visual representation of system requirements. It gives a glimpse of what the system is intended to do to the stakeholders and the users of the system.

Here the main actor is the user in various forms and an external Mahout recommendation system which interacts with the system. The below diagram gives an idea of what the system does in brief.

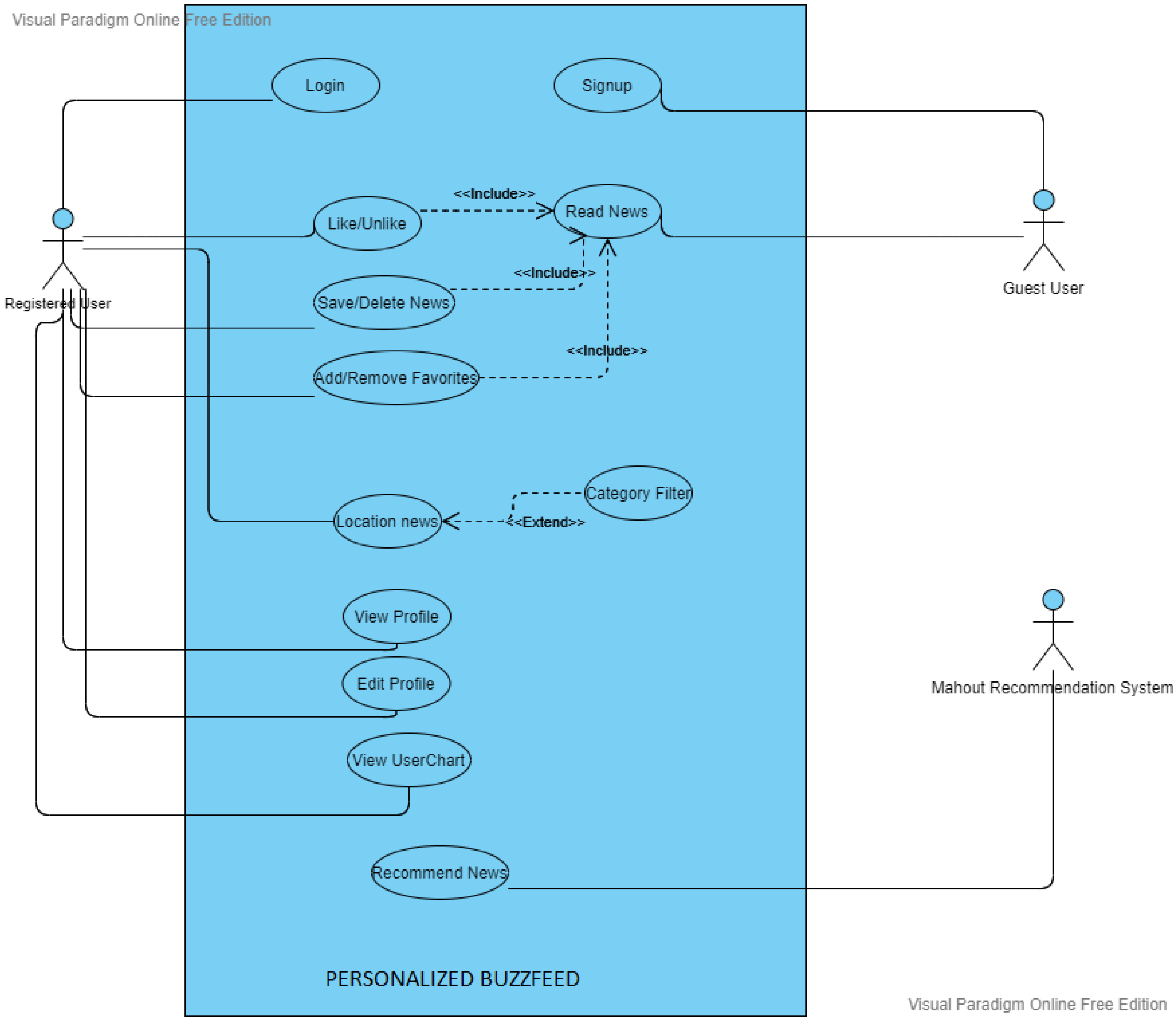

Figure 4.7: Use Case Diagram

4.3.2 Class Diagram

Below Class diagram represents the static view of the application and it has attributes and operations which represents the structural and behavioral features of the application.

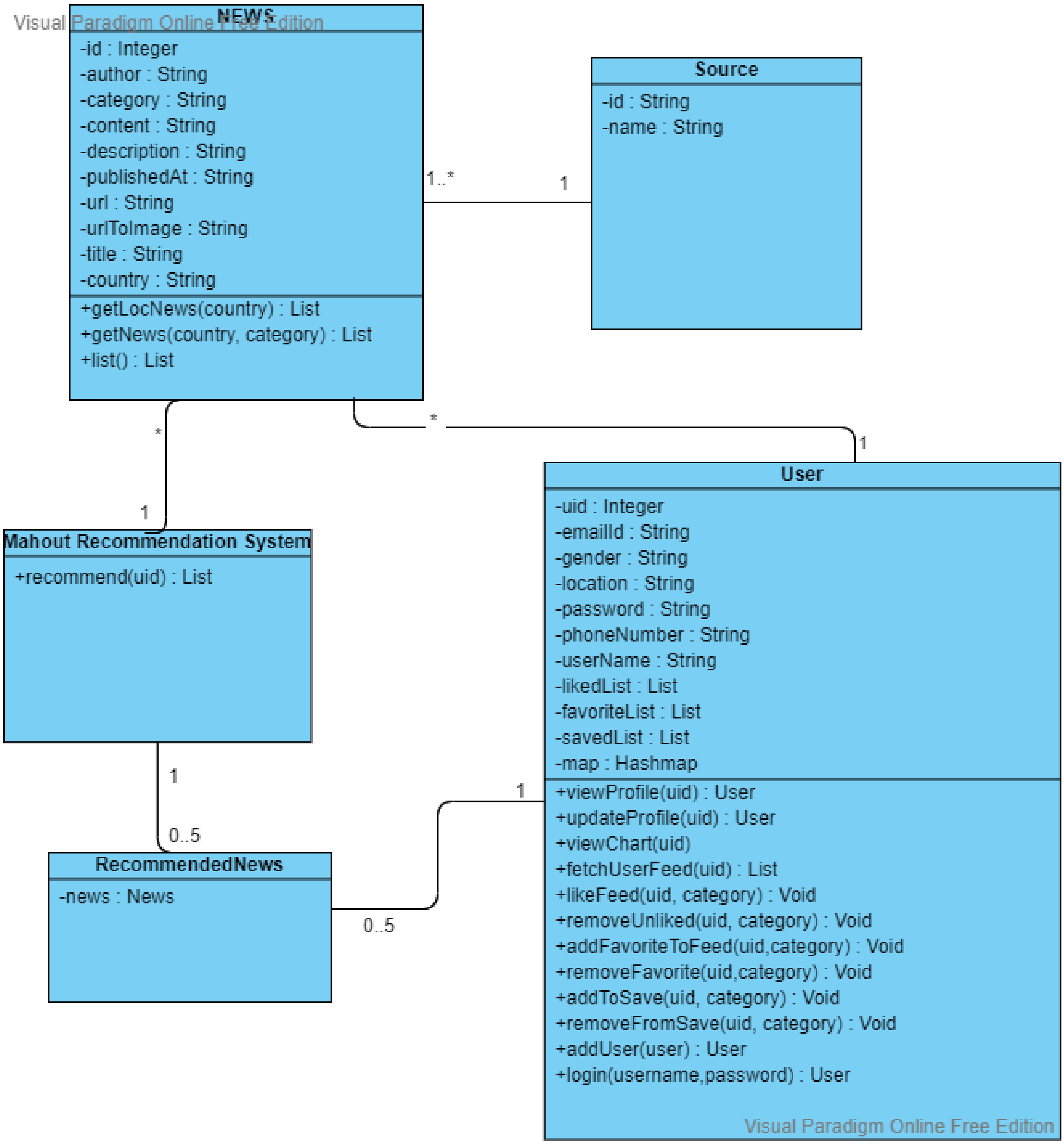

Figure 4.8: Class Diagram

4.4 Database Design

The database design for the application is represented as :

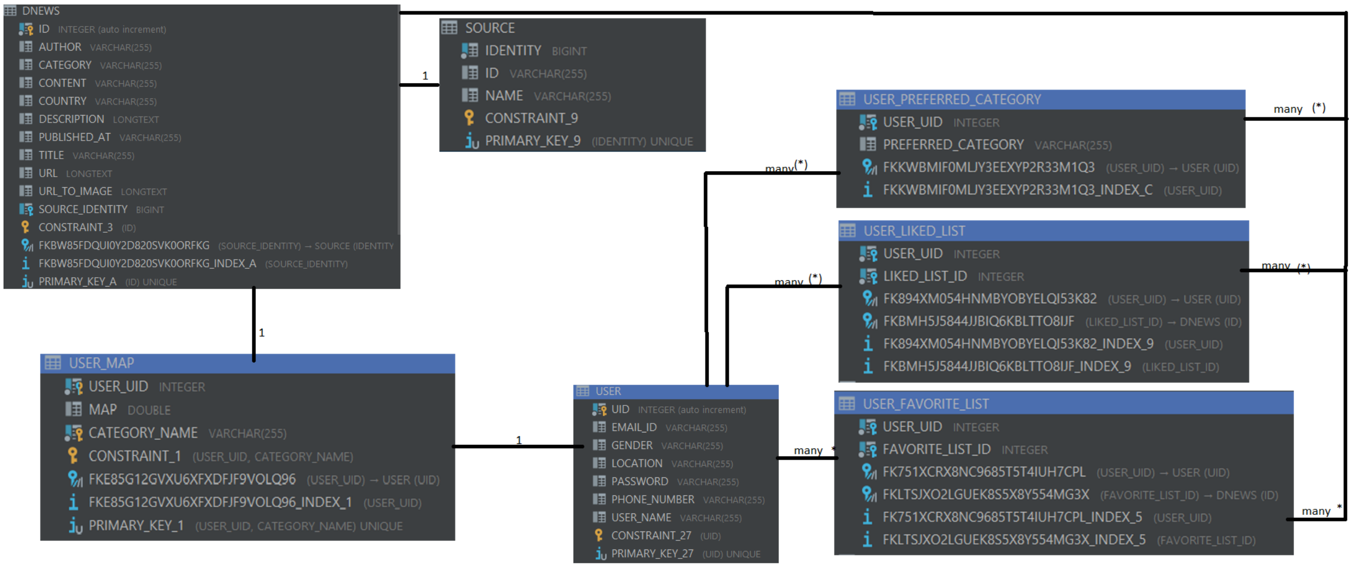

Figure 4.9: Database Design

4.5 Application Design and Workflow

4.5.1 News Dataset

The first and the foremost activity for this application is to gather the right dataset that is required to accomplish the product. There are many datasets and third party service APIs that provides the news data that are required but there were certain limitations in the necessary fields that out news model needs.

NEWS API

There is a third party rest API service called - 'NEWS API' which provides live news articles throughout the globe. It is accessible through - https://newsapi.org/

The application sticks to the database data for news data even when we get the live news without any restrictions from NEWS API service is to stay in some safe zone for the final project presentation and avoid any dependencies in order to avoid any case of unexpected issues of loss of data at the time of presentation [23].

Why NEWS API:

The News Api is chosen as the news dataset for our project as we require few major fields like category, url, url Image and ccountry details to present the news to any news readers in our application. And most importantly there is a requirement to classify the articles based on country and category and these fields can be made available with the news api request. Hence we decided to go with News API service.

Let's see the structure of news api request and response and how it can be made useful for us.

NEWS API Request:

The NEWS API request can be obtained from https://newsapi.org/v2/top-headlines which will result in all country top news that can be processed.

The is carried out for few countries and categories from the NEWS API and then processed for the response.

Request Parameters:

The request parameters we have chosen for our application includes the following:

- **APIKEY:** This is more important and unique for individual users and allows access to the news api.
- **country:** The two-letter country code for whichever countries we need news.
- **category:** The category news we may be interested in can be selected from the available set of categories list: Business, technology, entertainment, health, sport,general and science.

NEWS API Response:

The NEWS API Response data has many fields which makes it more flexible for our application.

- **Title:** The headline for the news.
- **description:** Descriptive information about the article to know more details.
- **URL:** To navigate to the respective url for the news publications to go through the whole article.

- **URL Image:** To have the respective images with the news article that makes more sensible.
- **Source:** The valid source information for the article who has published the news.
- **Published At:** The published time and date for the article by the source.

  The response does not have country and category fields directly available but appended to the items in the database.

4.5.2 News Feed

The next phase of the application is to present the news to the guest users to view the articles but to like or add them to favorites they need to have a valid login credentials and logged into the application.

NEWS FEED Design: News articles is now available on the database. The only operation we do here is to just fetch or read the news for a guest user. The Guest user access is controlled to view just view and read the news and restricted from interactions with the article as they interactions cannot be monitored for personalization.

4.5.3 Registration and Login

1. **Add User Profile**

   - **User Registration:** The users are requested to give some basics information about them to derive their interests from their context details for setting up their initial preferences. This user details page is primarily important for 'User Profiling'.
   - **User Login:** Users login module works with JWT authentication mechanism to allow users to login and view and interact with the application.

Spring Security and JWT Authentication This module is designed inline

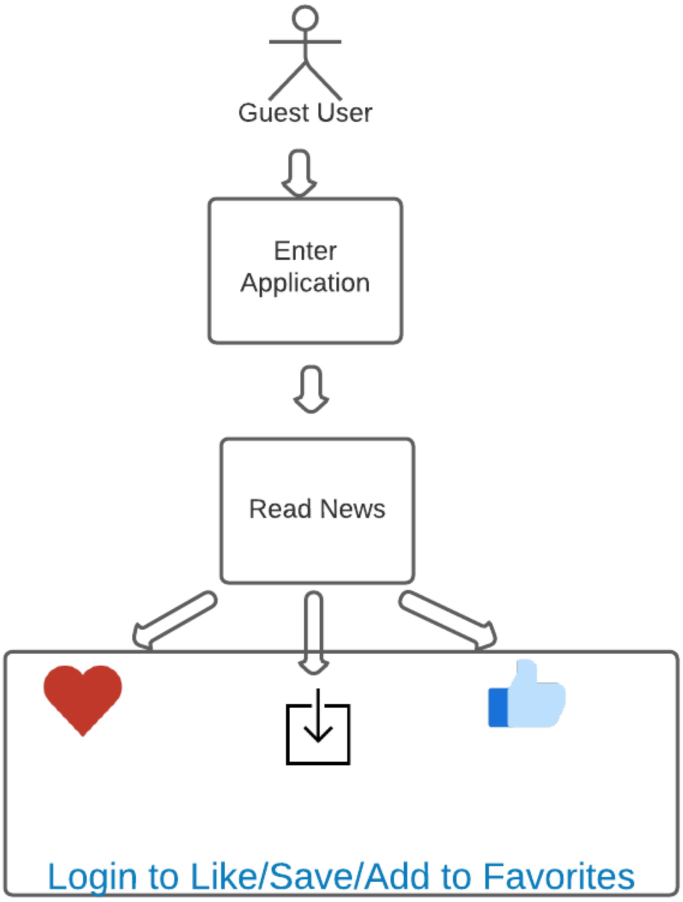

Figure 4.10: Guest User

with the Spring Security concepts which authenticates the user with their login credentials. On valid Authentication with the username and password, the server responds to the client with a valid **JWT token** generated with the user details. Upon receiving the JWT token, the client has to send all the http requests with the authentication header in bearer token form otherwise it the request would be rejected by the server.

This way it secures all the http requests and response in Spring Security application [6].

Spring Security Architecture Authentication manager:

Spring Security works with Authentication manager processing the http requests and the response. **Authentication Providers:**

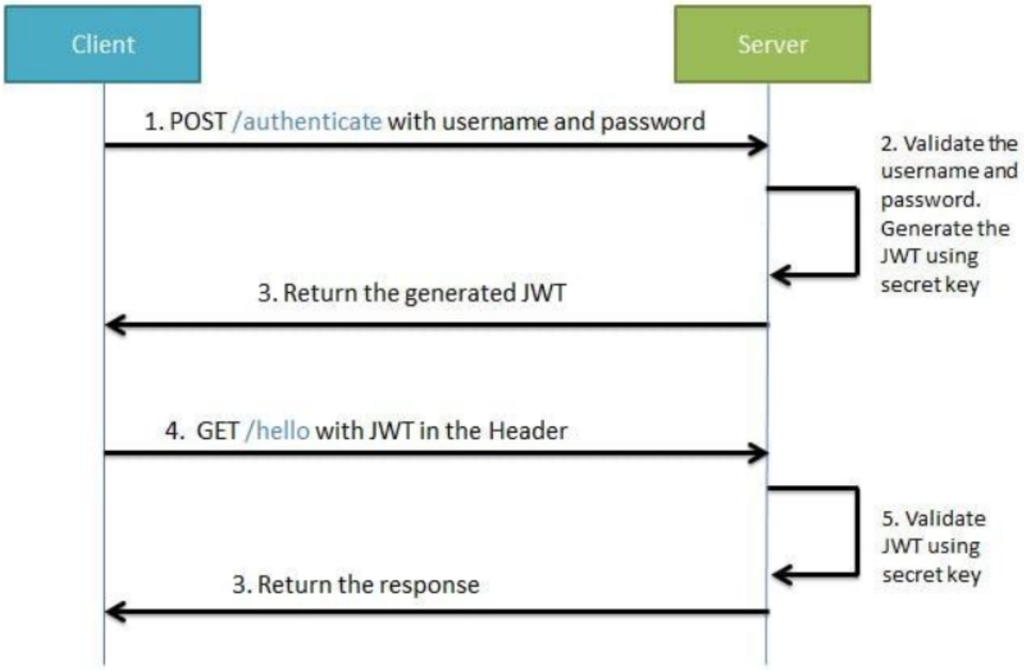

Figure 4.11: JWT Authentication

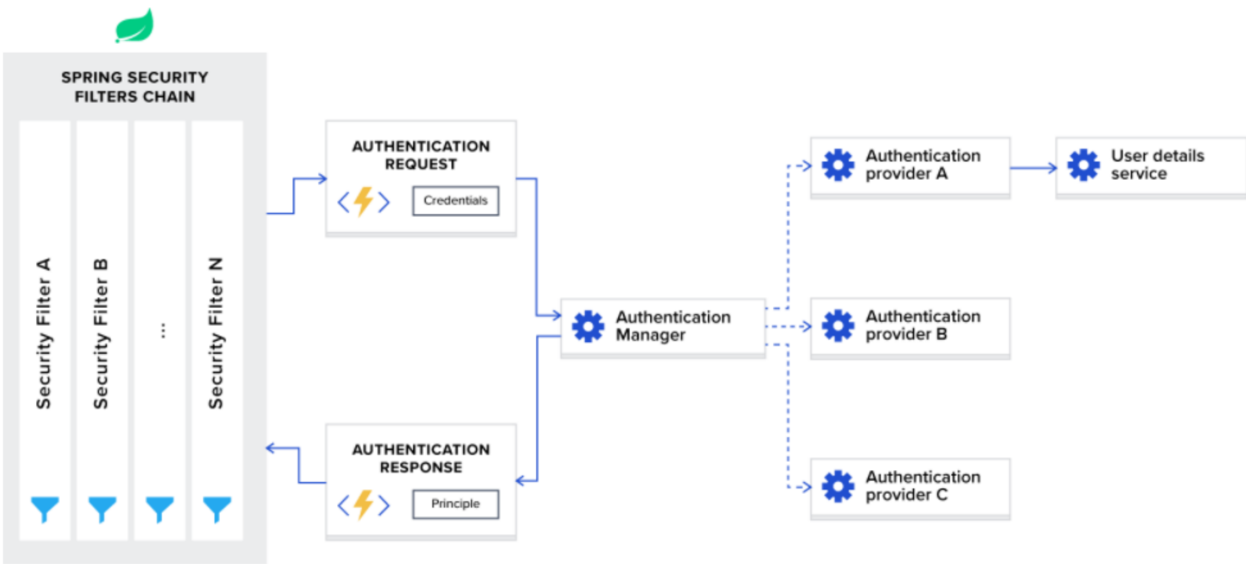

Figure 4.12: Spring Security

Authentication manager has multiple Authentication providers which process different types of authentication services. This is an interface which has a function called 'authenticate' which authenticates the user upon request.

User Details Service Class:

Additionally on valid authentication, it is necessary to generate JWT token with the username. Hence to achieve this we implemented **UserDetailsService** class in our application which loads user data with the username using the method loadByUsername. This helps in generating the JWT tokens with user claimed details.

2. **View User Profile:**

   Once the user is logged into the application, he lands in the user details page where he can view his/her profile details given during the registration process. This is processed as GET REST API call to the backend.

3. **Update User Profile :**

   The logged in user can also update his user profile from the client side anytime and view the updated details of his profile. This is processed as UPDATE REST API call to the backend.

4.5.4 User Personal Feed

User personal feed loads different content for each user based on user's likes and interests. It analyses individual's interests with the static and dynamic states and decides the overall weightage for the preferences of categories for each user and loads them respectively.
Here is where personalization comes into picture. Personalization plays a key role in this application and configured in two different ways as mentioned.

User Category Map: Each user on signup is assigned a Hashmap which maps users preferred category and their respective weight. So this keeps track of the users most favorite category based on weight.

- **Contextual Personalization:** Achieved with the User Profiling. Here the selected preferences during the signup and the location are considered. The

selected categories on signup will be added **+2.0** weights and the rest **0.0** . This map stores individual users category weightage. This is performed each time when a new user is created.
The location preference is used in visualized personalization context which comes in a later view.

- **Behavioral Personalization:** This works understanding the users varying interests everytime and keeps it updated in the category map on change. It analyses the users likes, dislikes, favorites, saved news and understands the preferences implicitly and maps the preference weights accordingly.
So, lets discuss how it judges the various interactions programmatically in brief with a pseudocode following the explanation.

  (a) **Likes:** The likes are identified with a filled thumbs-up icon and everytime the users like any article the respective category is identified and that category weight is increased by **+1.0** each time. This way it keeps the liked category into account and the user's category-map is updated.

  (b) **Dislikes:** The likes are identified with an unfilled thumbs-up icon and everytime the users like any article the respective category is identified and that category weight is decreased by **-1.0** each time.

  This way it keeps the liked category into account and the user's category-map is updated. This is marked a lower weight as the likes and dislikes sometimes are random hits based on the story or the article and cannot completely assure the interest rate.

  Like Zone: Also, the associated liked articles are mapped to a list and made available for users anytime to re-visit their liked ones.

  (c) **Favorites:** The favorites are denoted by a filled heart icon and the user's favorite article's category is identified and the respective category weight is increased by **+3.0**.

  (d) **Unfavorites:** The favorites are denoted by an unfilled heart icon and the unfavorite article's category is identified and the respective category weight is reduced by **-3.0**.

This is marked a higher weight compared to other interaction as the favorites explicitly states they are the most interested and remain almost the same.

Favorite Zone: Similarly users favorite articles are monitored and stored in a separate view for the users to revisit and go through their favorites anytime.

(e) **Saved Articles:** The users are allowed to save their news articles to make them available anytime for later read. This is a similar functionality that we have in many applications like Read Later and Watch later items. This helps the users to keep track and not miss the articles from reading at a later time. This is highlighted by a filled download icon and the weight associated with this option is **+2.0** for the respective category on save.

(f) **Delete Saved:** This allows the user to remove any article from saved list once they read through the article and there is no purpose having it in the saved items or read later items anymore. The respective added weight while saving is decreased here to maintain the consistency of the category preferences

Read Later: As mentioned there is a separate view for read later items to make it easy for users to know where exactly the saved items are and help them read and delete anytime based on their interests.

Automated Personalization:

The above theoretically described personalization is achieved using a custom-designed algorithm called **'Weight-Map' Algorithm**.

Weight-Map Algorithm:

Weight Map Algorithm builds user Personalization programmatically based on above mentioned factors or user engagements. Each user engagement is counted and assigned respective weights as described above. The algorithm is named as 'Weight-Map' algorithm because a hash map is created for each user which maps users category based weights in the backend.

Similar Existing Algorithm: Facebook determines the personalization of feeds based on an algorithm called 'Edge Rank' Algorithm[7]. This is a rank-based approach considering three major factors: Affinity Score, Edge Rank and Time Decay. These factors ranks and decides the order of appearance of stories in the user's newsfeed.

The updated facebook algorithm works based on the likelihood of the user interatcions with the post. This is a complex algorithm which has to cross four systems.

1. **Inventory :** This is the initial phase where the facebook decides what all contents to load for any user based on his affinity which means the close relationship graph.
2. **Signal :** This phase has to approve the content based on various assessments like type of feed, whom and when the content was posted, the likes, shares,comments and all other factors and given **weightage**.
3. **Predictions:** This phase predicts how far the respective user might like the posts based on his previous behaviour[1].
4. **Score:** The above two phases are combined like the predicted story and the respective weight for that story are combined and assigned a relevancy score which helps in fetching the news feed sorted in descending order.

Comparison - Similarities and Differences To Weight-Map Algorithm:

1. **Phase 1:**

   Weight-Map algorithm works in a simplified pattern taking all the contents at the first phase.
2. **Phase 2:**

   The second Signal phase works bit differently. Instead of evaluating each content, we assign weights for user interacted content's category everytime the user likes, saves or add the news to favorites. Actually this is the third phase of predictions according to facebook algorithm which is our only filtering mechanism and scoring method for assigning weights.

For example, The user likes certain article, then the category weight for that article increases by +1.0 and the vice-versa.

3. **Phase 3:**

   This is the final phase where we filter based on weights or score assigned to categories interested for each user. Each user will have a category-weight map assigned in previous phase which is sorted in descending order and then based on more range of weights to load respective number of articles.

   For Example. category score more than 50, loads 20 articles from that category and for a lesser score the limits decreases. Then with the overall category weightage we provide the users with their personal feed.

Visual Representation of Algorithm: The below figure helps us to understand more clearly on how this algorithm works. **Pseudo-code of Weight-Map Algorithm:**

```
User register:
   if preferredcategory in categoryList:
      category-map.weight = 2.0
    else
      category-map.weight = 0.0
    Add user with cmap
Monitor Actions:
  if(logged in):
   if click(Like):
      if(liked):
         fetch category for this feed
          Add category-map.weight - 1.0
        else:
```

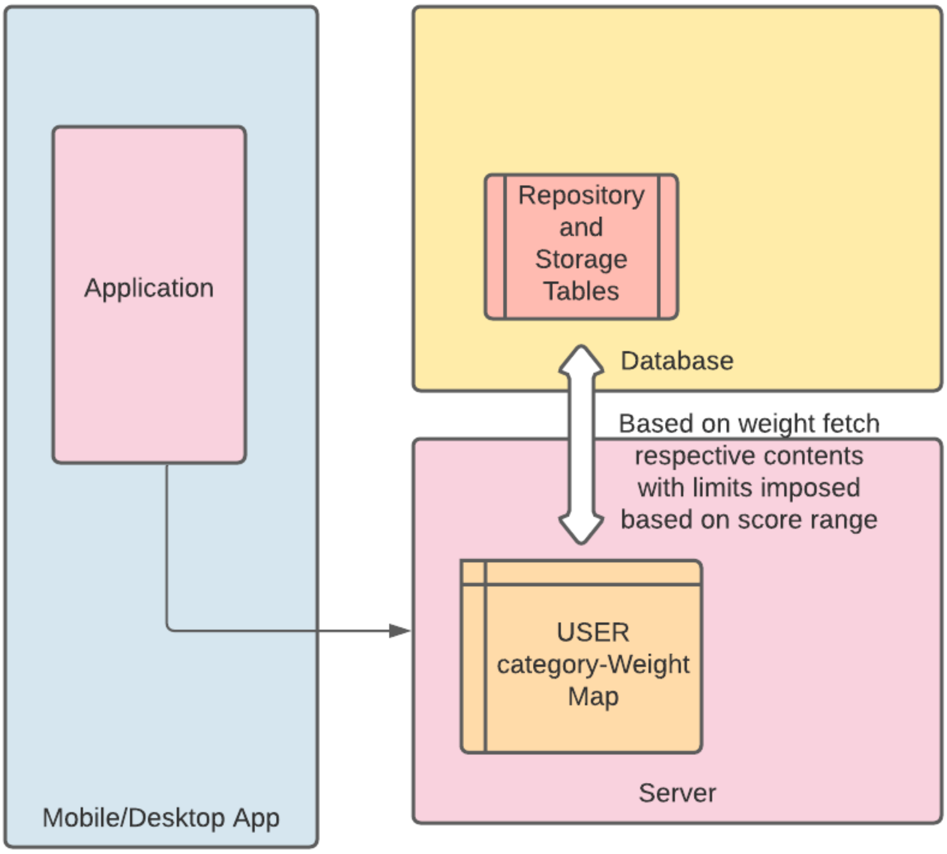

Figure 4.13: Weight-Map Algorithm Flow

```
      fetch category for this feed
      Add category-map.weight + 1.0

if click(Favorites):
    if(favorite):
      fetch category for this feed
      Add category-map.weight - 3.0
     else:
      fetch category for this feed
      Add category-map.weight + 3.0

if click(Save):
    if(saved):
```

```
        fetch category for this feed
        Add category-map.weight - 2.0
      else:
        fetch category for this feed
        Add category-map.weight + 2.0
else:
  print(Login to personalize!)
```

And to prevent the userfeed from overload with all news of a particular preferred category, the application imposed limits on the results to load the news collection from specified category based on its weight. More the range of weight, more are the number of results from that particular category and vice-versa. This enables to prioritize the top category preference and load them more instead of loading everything equally.

```
User-feed:
      fetch sorted category-map based on weight
   for sorted category in category-map
      if(weight > 50)
         fetch top 20 newsfeed on this category
      if(weight > 10)
         fetch top 12 newsfeed on this category
      if(weight > 5)
         fetch top 8 newsfeed on this category
      else
         fetch top 5 newsfeed on this category
```

4.5.5 Visualization

Visualization is accomplished by a special library called amcharts which gives sensational maps and charts which are effective for locating the news around the globe and understanding the user chart data.

Visualization is applied in two areas in our application.

1. **User Charts:** User charts are visible in the user profile view where every user will be able to see their category chart which shows them their most favorite category and the overall percentage of interests he expressed while reading through the news articles in the application for all the available categories.
   This is represented in terms of pie-chart which shows the category-name and rate of interests in percentage along with the number of articles which made the figure.

   Pie-charts: Pie-charts represents a part-to-whole relationship. Every piece or slice of a pie-chart represents one component which together contributes the whole[2].
   Why Pie-charts: Pie-charts help clearly help us to understand how much each category contribute in a user's interests because pie-charts are best used in case of explaining and understanding part-whole relationship. Also, it is bit more attractive than simple column or bar charts in a way hence the choice of pie-charts.
   Pre-requisites: We installed the external library - **@amcharts/amcharts4** to have the basic functionalities from amcharts 4. Once integrated the amcharts, we can extend to any additional pluggins which we require in future.
   To achieve amcharts pie-charts we required two modules - **am4core and am4charts** which were installed.

2. **Globe View:** Another area where we have the visualization is the map or the globe view in the application where it projects a map in orthogonal projection.
   Pre-requisites: For acquiring the maps, we need **am4maps** module which gives the globe view and we can make this to appear in any projection like a globe or a flat map.
   Why Orthographic projection: We have chosen **Orthographic projection** in this application which is a globe structure because it is more interactive and looks appealing to the users as it keeps rotating around which makes it lively.

   Here is how it looks when projected: **How it is designed to help us:**

Figure 4.14: Amcharts Map

Visual Country Filter: This view of globe allows users to chose which country's news they would like to read through anytime. This helps in filtering the news for each users preferred country.

Category Filter: At the same time, when the users prefer to know more about each country specific category news, the application gives them another option to select any category they need from a list of categories which again filters selected category data for selected country.

Combined Filters: Thus two filters are applied on user's preference. So, users can read any country news and any category news and compare the category news data against any other country if required.

So, technically the combination of filters is achieved using **Behavioral subject** in angular. This helps in retaining the category selected for change in country and change the category selection only on subscription.

For example: When the user selects 'Australia' the application loads the Australia country results with 'All' categories selected. When the user changes the category to 'Sports' the user will be able to look specifically Australia's Sports data and change to any other category or country as preferred.

Personalized Visualization:

Initially to make it more personalized, we load the **user location** by default with **'All'** categories loaded in the results. This allows the user to open the view and have a look at his/her country data and option to see any category data for the location.

Whom It helps Mostly: It is mostly helpful for journalists or news organizations who compare news against countries and want to understand the worldwide news in compare and contract approach. Also, it obviously helps the general users who are interested to know more information anytime.

4.5.6 Recommendation

Recommendation is provided for each user based on user's interests rate compared against other users who have similar interest pattern.

Mahout Recommendation System: This is achieved using **Mahout Recommendation system** which handles real-time users interests data and recommends articles based on Collaborative Filtering algorithm. It is chosen because it supports different approaches for Collaborative filtering both item-based and user-based and also highly scalable for large datasets[28]. For Example, Amazon and Netflix recommendation systems.

Recommendation System Architecture: Recommendation works in several stages as follows:

Proposed Approach - User Based Recommendation: User-based approach is chosen for this application and it works on the concept that users who have similar interests will be interested towards similar categories[18].

Algorithm works in three steps[18].

```
For every item I that u has no preference for yet
For every other user v that has a preference for i
   Compute a similarity s between u and v
```

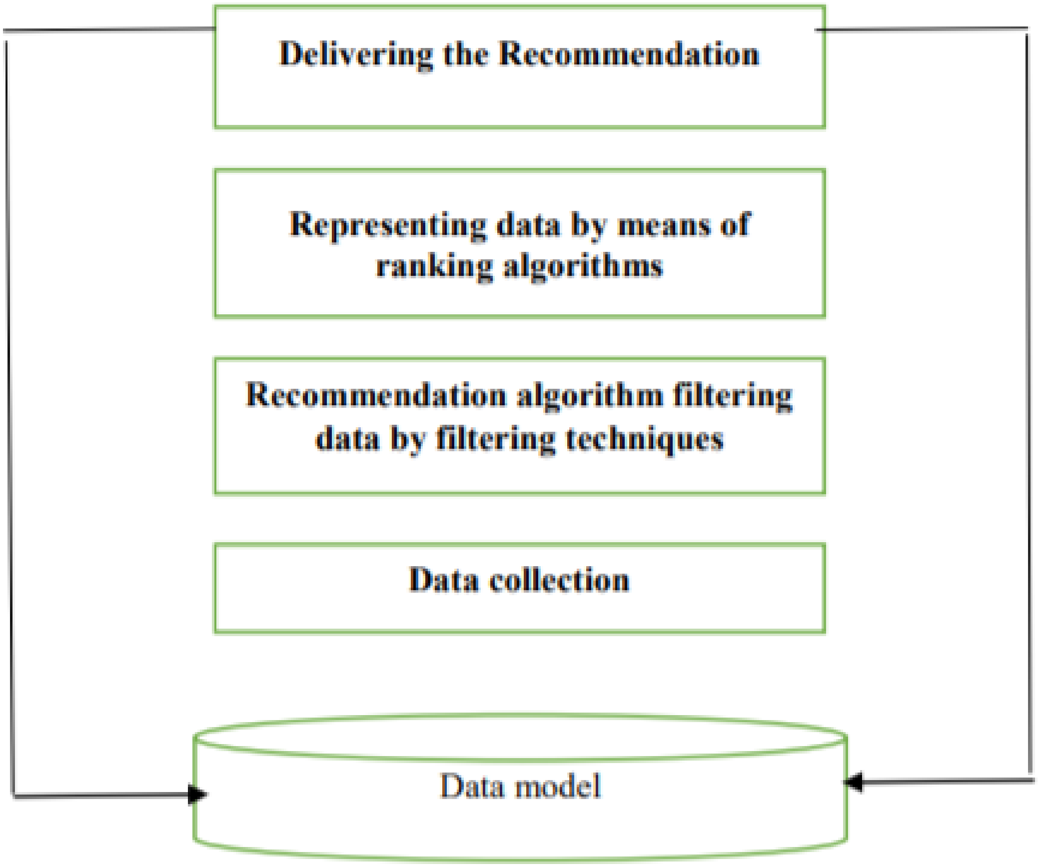

Figure 4.15: Recommendation Architecture

```
Incorporate  vs  preference for I weighted by s to avg
Return the top items, ranked by weighted average
```

Explanation of Algorithm:

1. 1. Find user u with no preference for item say. i
2. 2. Find other users v who has preference towards i and have similarity model between u and v
3. 3. Compute average for the preference for items and recommend items with top weight.

How does it work for Buzzfeed: Buzzfeed also works out this algorithm in three stages except that incorporating weight is not necessary here as we have them pre-calculated for every user in user-category map.

Recommendation Input Data: The input data is obtained from user-category map stored in database table where we have the user category and respective weights for every user which we convert to csv files for Recommendation process.

How it Works: Let's discuss the working process with an example. When user 'u' has more interest weights for categories 'Sports' and 'Science' and there are other similar users with more which is around similar weight as of 'u' for 'Sports', 'Science' and additionally has more weight for other category 'Club', so the user 'u' is recommended with some items in 'Club' which might sound interesting for the user.

Observed Limitation: This is more related when the there are more number of categories and users are not sure which one they find to read sometimes. For our application, we have less number of categories due to some dataset limitations but it helps in understanding how it works clearly.

Chapter 5

Implementation

Implementation of designs has taken place step by step following the timeline plans and wireframe diagrams to achieve the desired output on screen. The implementation phase of the project involves the execution of below tasks in an organized way.

Tasks Planned for Execution:

1. Authentication
2. News Data Script
3. Access News feed
4. Contextual Personalization Implementation
5. Behavioral Personalization Implementation
6. Visualization and Projections
7. Recommendation thoughts and concepts

Authentication:

Springboot authentication is performed using spring security with **UsernamePasswordAuthenticationFilter** which is implemented using **WebSecurityConfigurerAdapter**. Code snippet for spring security which is implemented inside configure method:

```java
@Override
protected void configure(HttpSecurity http) throws Exception {

    http

            .csrf().disable().authorizeRequests().antMatchers(HttpMethod.POST, "/authenticate/","/signin", "/api/Users","/api/UserFeed/**")
            .permitAll()
            .antMatchers(HttpMethod.GET, "/", "/api/News/news",".api/UserFeed/recommend").permitAll()
            .anyRequest().authenticated()
            .and().exceptionHandling().and().sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS) .and()
            .httpBasic();
    http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
    http.cors();
}
```

Figure 5.1: Spring Security

News Data Script:

News data is loaded into database from the api json response using the **jackson-databind** dependency in springboot. We will read the DeatiledNews.json and map the news to our NewsApi model. This way we can map set of news data to our News Model.


```java
@Override
protected void configure(HttpSecurity http) throws Exception {

    http

            .csrf().disable().authorizeRequests().antMatchers(HttpMethod.POST, "/authenticate/","/signin", "/api/Users","/api/UserFeed/**")
            .permitAll()
            .antMatchers(HttpMethod.GET, "/", "/api/News/news",".api/UserFeed/recommend").permitAll()
            .anyRequest().authenticated()
            .and().exceptionHandling().and().sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS) .and()
            .httpBasic();
    http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
    http.cors();
}
```


Figure 5.2: Spring Security

Access News feed:

News Feed data can be accessed by any user who land on the application url and does not require any login to read the news. News data is accessed with or without parameters country and category based on requirement to get the desired news using the **@GetMapping** Rest call with no authorization header for general news to allow access for all readers. For specific category or country news by any user, the use of http authorization header with bearer tokens becomes necessary.

Contextual Personalization Implementation :

Contextual Personalization is implemented using user details which are provided during the signup. For the initial prototype I have taken the user selected preferences and the

```java
@GetMapping("/news")
public ResponseEntity<Iterable> list() {
    Iterable<DNews> news = dNewsService.getAll();
 return new ResponseEntity<Iterable>(news, HttpStatus.OK);
}


@GetMapping("/location/{country}")
public ResponseEntity<Iterable> getLocNews(@PathVariable String country) {
    Iterable<DNews> news = dNewsService.findNewsByCountry(country);
    return new ResponseEntity<Iterable>(news, HttpStatus.OK);
}

@GetMapping("/cateloc/{country}/{category}")
public ResponseEntity<Iterable> getnews(@PathVariable String country, @PathVariable String category) {
    Iterable<DNews> news = dNewsService.findNewsByCountryCategory(country, category);
    return new ResponseEntity<Iterable>(news, HttpStatus.OK);
}
```

Figure 5.3: News Feed

location for providing personalization.

Selected Preferences

The user preferences are taken into account when each user is being added to the user database and their category map is setup with the standard initial weight which is assigned +2. The weight is chosen by the developer as a standard for the application.

- **Zonal News**

  Location is considered while providing the personalization and to make it more lively the user's location news is made available with visualization setup using amcharts which provides geographical geojson data and map to load all the country details and load respective data based on selection. And the users are provided options to filter any category data upon country selection. This helps in filtering and learning any specific news for any country to understand the zonal news much better.

- **Implementation with Amcharts and BehaviorSubject:**

  Amcharts are implemented using some basic scripts from amcharts.com and enhanced with BehaviorSubject to store any value and return immediately on subscription. This is helpful in holding the category selection while switching to any other country and return the previously help category data for the selected country.

```
loadglobe(country){
  var globe = am4core.create("chartdiv", am4maps.MapChart);
  globe.panBehavior = "rotateLongLat";
  globe.geodata = am4geodata_worldLow;

  globe.projection = new am4maps.projections.Orthographic();

  var pseries = globe.series.push(new am4maps.MapPolygonSeries());

  pseries.useGeodata = true;
```

Figure 5.4: Amcharts Visualization

Behavioral Personalization Implementation:

Behavioral Personalization is implemented by tracking user interactions with **weight-map algorithm** which assigns respective assigned weights for various reactions and measures their interests based on varying weights.

How it works? The different category news are added or reduced based on users changing interests everytime and their userfeed is filled with respective number of news data. More the interests towards a category, more the number of those news data and vice-versa. There are various methods which works for every user interactions like add to favorites, likes, save feeds and updates the user interests everytime through the user-category map. Then finally the user feed is fetched using **FetchByUid** which loads respective number of data based on interests neither more nor less. The entire logic goes inside the back-end Controller and Service domains- 'UserFeedController' and 'UserFeedService' classes.

Visualization and Projection Visualization and projection of user charts using amcharts works with the defined user-category map where each user holds a map with category name and respective weights where the weights gets updated with personalization aspects based on the Weight-Map Algorithm. The data is projected with pie-charts inbuilt chart data with values filled from user map and the code snippet for pie-chart projection looks like :

```
  loadcharts(){
    this.user=this.getUser()
    console.log("user data" + this.user)
    this.editprofile = false;
    this.loadprofile = false;
    this.chartsdata = true;

var chart = am4core.create("chartdiv", am4charts.PieChart);

// Add data
chart.data = [{
  "category": "Business",
  "value": this.user.map['business'],
  "color": am4core.color("#cf0a80")
}, {
  "category": "Entertainment",
  "value": this.user.map['entertainment'],
  "color": am4core.color("#a72dbd")
}, {
```

Figure 5.5: Chart Projection

Recommendation thoughts and concepts: Recommendation works based on conceptual Machine learning algorithmic implementation. The recommendation algorithm is implemented by Mahout Recommendation system which is used in this application to gather the interests and preferences of various users and match them based on their similarity of interests and provide various categories which might be preferred by the users who might haven't gone through those news categories before. This is the logical working process of **User based Similarity** which suggests users based on similar taste.

Mahout Recommendation System: The algorithm works very well

Chapter 6

Evaluation

The application is evaluated mainly based on user perspective. The major aim for any performing user-based evaluation is to revolve around user-friendly and user-focused development[27].

Usability Testing:

Usability testing is performed with or without users by various techniques. In this project, we have done usability testing with users by carrying out various tasks in the application by selected users and analysing and observing the tasks carried out at several stages and collecting the results and resolving few bugs based on the test report[27].

Usability testing measures the overall quality and quantitative results of an application. It mainly focuses on the following parameters.

1. Effectiveness
2. Efficiency
3. Accuracy
4. User-Friendliness[11].

The Overall Process: The overall process flows through several stages [25]

1. **Participants Selection:** Testing was performed selecting 3 participants based on age, gender and location. Two participants in below 30 (with more experience

of mobile) and one above 40 (with limited experience using mobile or system) with different genders and location were chosen to know the feedback.

2. **Task Design:** Set of Tasks were designed to be carried out by all the users. The tasks include the overall functionalities and performance measures of the application.

Task#	Task Items
1	Register new User
2	User Login to application
3	Go through UI Menus and Navigation to pages
4	Newsfeed data access
5	Like/Save/Favorite interactions
6	View all interactions and understand personalization based on user details and interactions
7	View User charts and data
8	View Globe and news data and understand how to change category or country to access data
9	Navigate to User profile
10	Edit User profile
11	Go to saved feeds and remove after reading if needed
12	Land on any interested news
13	Validate the user details on registration
14	Understand recommendation process and view recommended items if any.
15	Logout the application anytime.
16	Is it easy to understand the application?
17	Limited navigation and satisfied single page application purpose?
18	How far the data is accurate in user profile and user charts?
19	Is training required for users?

3. **Task Analysis and Report:** The tasks are carried out in order by all the participants and observed for tasks completion, duration to complete the tasks and provided help and support to complete in case required. Based on observation, below findings were made and reported accordingly.

   Collective findings of user-study:

Task#	Limitations
1	User registration not properly validated in front-end and unable to understand why the signup doesn't complete
2	First name and last name not distinguished like in other applications.
3	No option to search easily for any news item
4	Unable to understand why like/save not working initially after landing on url.
5	Some urls broken in menus
6	Help Video or Tour for new users

Observations and bug fixes:

Some of the issues observed in test reports are fixed and updated in the application and the respective screenshots are as below for reference.

- **Missing Front-end validations:** Though the validations are performed at the back-end, the user is unable to understand why the application is struck since there is no message at the front-end. So, the front-end validation is performed with sufficient messages for user to understand.

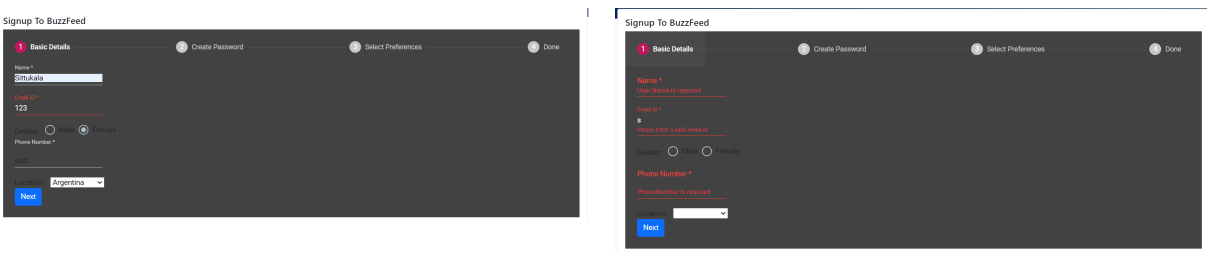

Figure 6.1: Validation message

- **Interactions not working :** The user reactions are not taken initially. The it was made clear it requires user login to allow interactions. But this should be properly conveyed to the users in a way they understand the pre-requisite to login. The popup appears if not logged in.

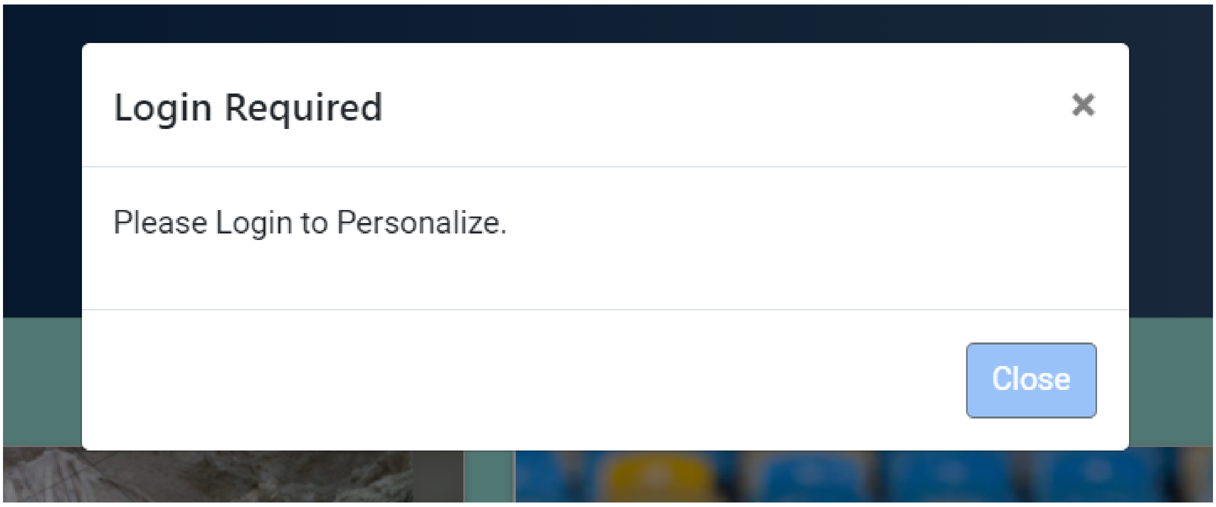

Figure 6.2: Pre-requisite for interaction

- **Search option :** Users were trying to search some news but there was no option to do, so it is added now.

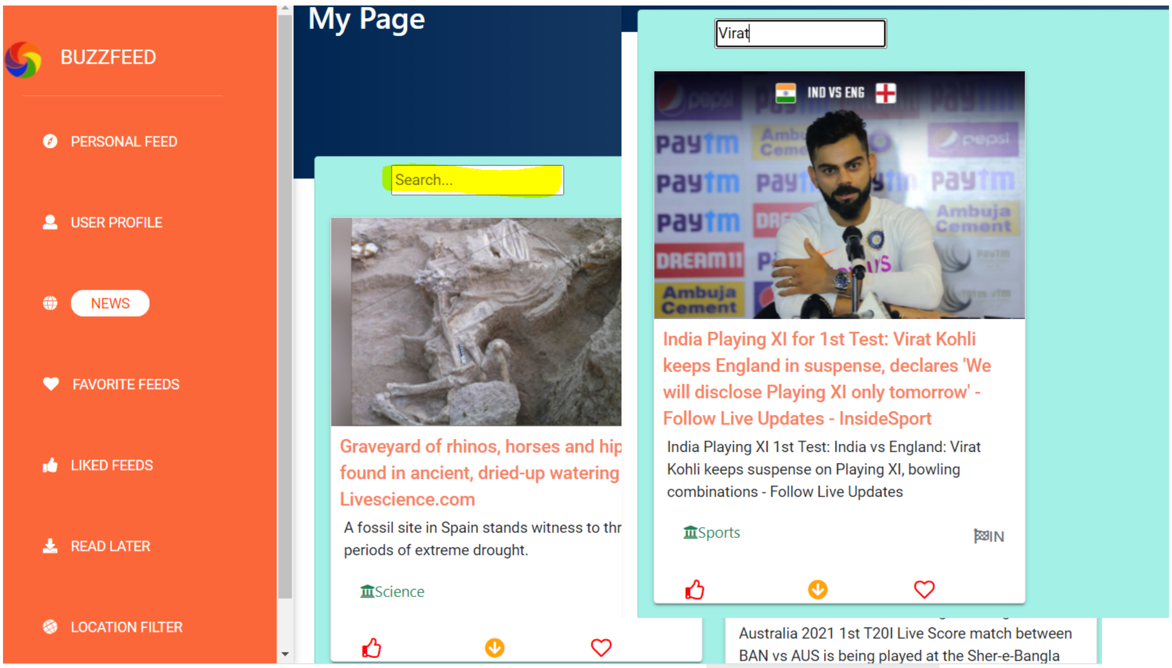

Figure 6.3: Search

Test-Driven Development:

Apart from User testing, the application is also tested in-line with the development and made sure all the functionalities are tested for both positive and negative outcomes using Test Driven Development approach. Developers intend to follow this approach incrementally writing unit test-cases and acceptance test-cases throughout the software development life-cycle [4]. The unit testing is mostly carried out using the Junit testing techniques for Java.

Structure of TDD :

TDD runs incrementally with the rapid iterations of the below steps in sequence[4].

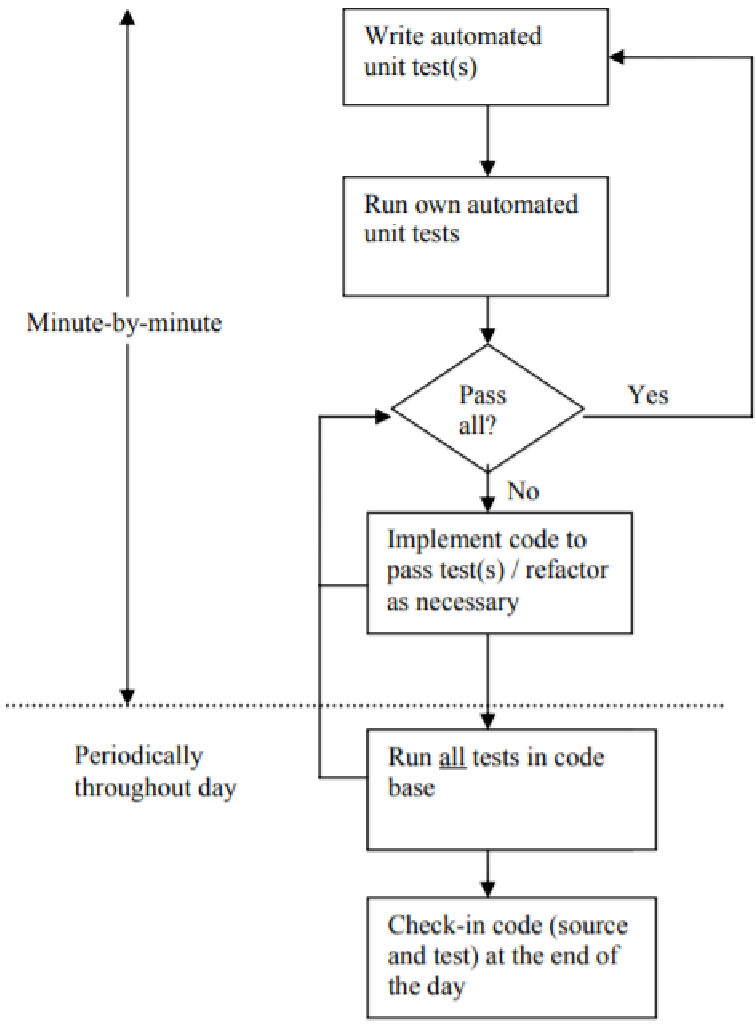

Figure 1: Test-Driven Development

Figure 6.4: Pre-requisite for interaction

Successful test-suites :

We have run several Junit test-cases for the functionalities and collected the test results. The test-cases are available for review under appendix with the sample test result screenshots here.

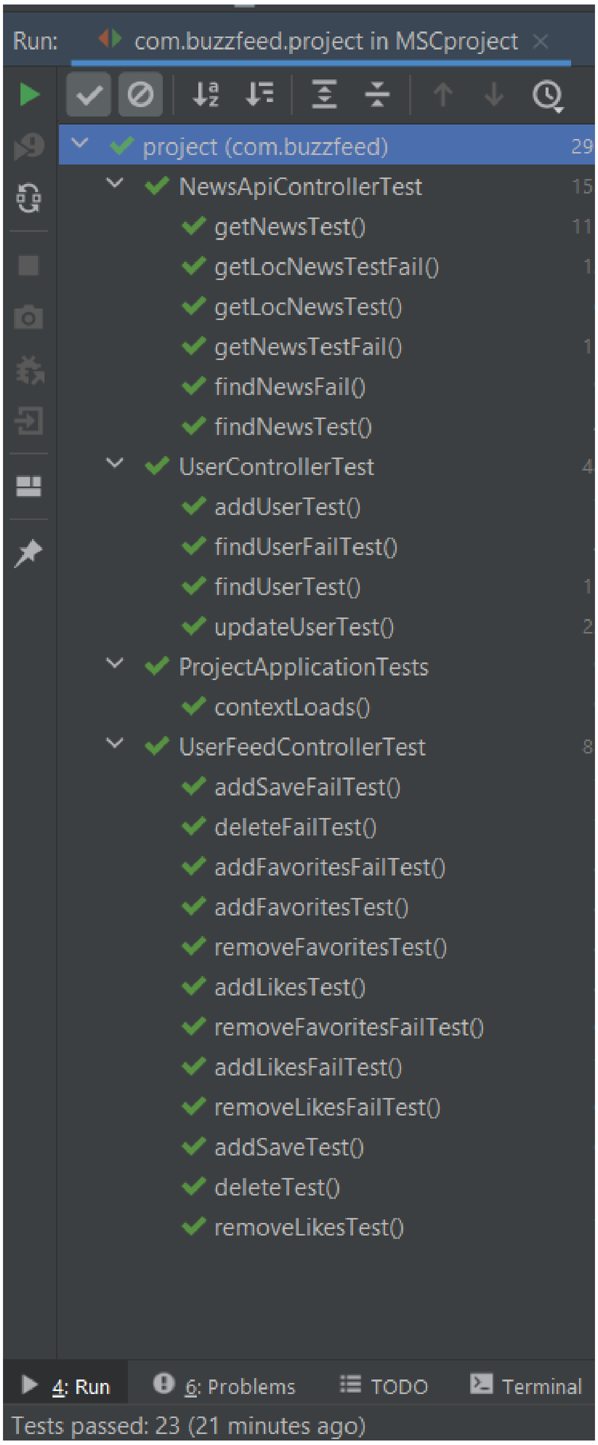

Figure 6.5: Test suite

Chapter 7

Challenges and Contributions

This chapter gives an overall idea of the challenges faced during the development process and the ways they were handled to sort the issues and overcome them with a workaround or a different possible solution. Also, the application highlights some of the future woks which can make the application to a successful product in the market.

7.1 Challenges Faced

The developer has undergone various challenges during the development of the application which includes the ground research and understanding of the application and core study process to develop the initial prototype. Though there were some challenges the developer successfully overcame most of them and able to develop the working model with sufficient guidance from supervisors. Here are some of the explicit challenges faced during the whole process:

- Selecting the right **news dataset** was the foremost challenge as the news content is the primary importance for the application. The news dataset comes with the image, image url, source information, description and title, etc. The news dataset was then taken from the Newsapi service which provides api json response from various sources and countries.

- **Personalization** is one of the most interested feature for any web applications and it required ground research and complex study to understand how well it can be made to this application. After reviewing many research papers and understanding existing applications, the developer was able to collaborate many views of personalization into this application.

- Developing the **front end components** from scratch in a user-friendly manner was bit time consuming for the project to decide on the layout and the wireframe diagrams.

- The other viewpoint of projections required brief study on how different kinds of charts work with angular and support of interactive map libraries. After analysing D3, leaflet, finally **amcharts** has been chosen as it has support for many charts and also geographical map projects with sufficient documentation for understanding.

- Updating category and country filter at the same time was bit tricky to keep in memory the old selection of one while the other changes to list the appropriate results. This was resolved with the usage of Behavioral subject which effectively resolved this issue.

- **JWT authentication** is a key mechanism to control login and token management and allows secured access to the functions. This was initially hard to understand and bit complex but resolved over time.

- The system needs to manage each user with own preferred news and weight based on their interests as a part of user profiling which was challenging to accomplish with a hashmap. The development of logical algorithm for the programming and making sure it works as expected was complex to achieve and required to note very minute details in updating the user behaviour everytime.

- Finally Recommendation system required major research on various existing applications and better way to recommend news articles with machine learning algorithm. This puzzle was successfully resolved using Mahout recommendation system which has very good java implementation support.

7.2 Results Achieved

The application called 'Personalized Buzzfeed' works as expected and is observed and tested for the successful achievement of its main objectives. The application provides the news articles for all the users and once the user logs in, it takes the user interactions and presents the user's personalized news articles which is worked out in background by personalization algorithm called weight-map algorithm. These are the general and personalized news feed in the front-end.

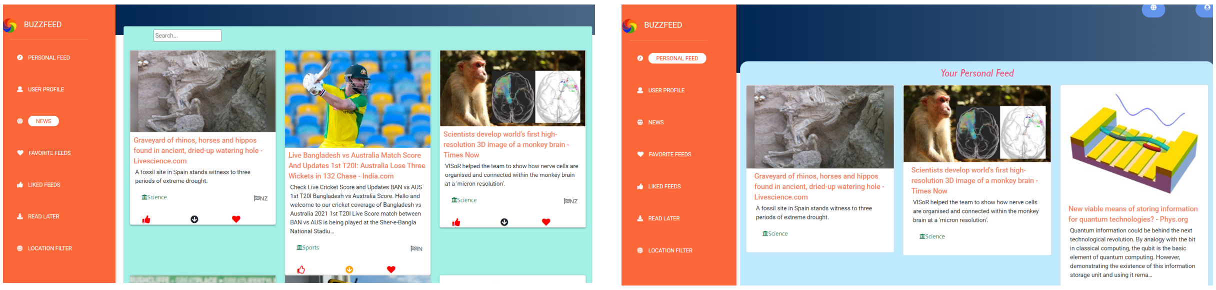

Figure 7.1: Personalization

Visualization is presented in two major areas by amcharts which makes it interactive and more clear for understanding. Below are the user charts projection and global map visualization in the front-end.

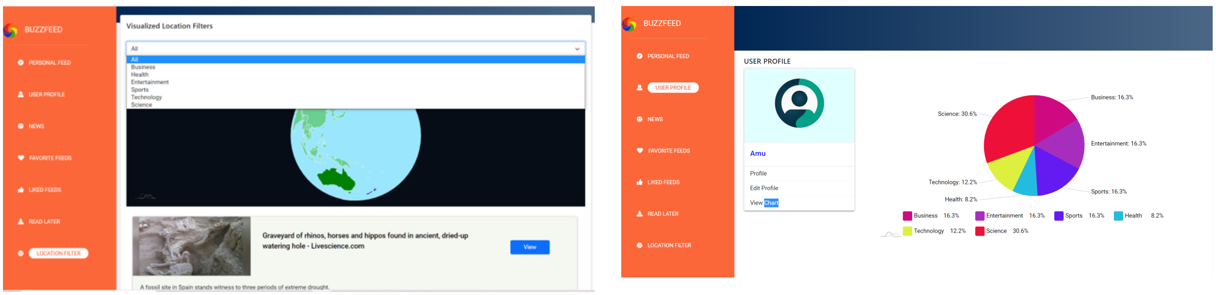

Figure 7.2: Visualization

Recommendation is achieved by Apache Mahout Recommendation machine learning algorithm and it appears in the client if the user similarity pattern is observed. Here is a below example where the user-user similarity is observed and the application notifies the user.

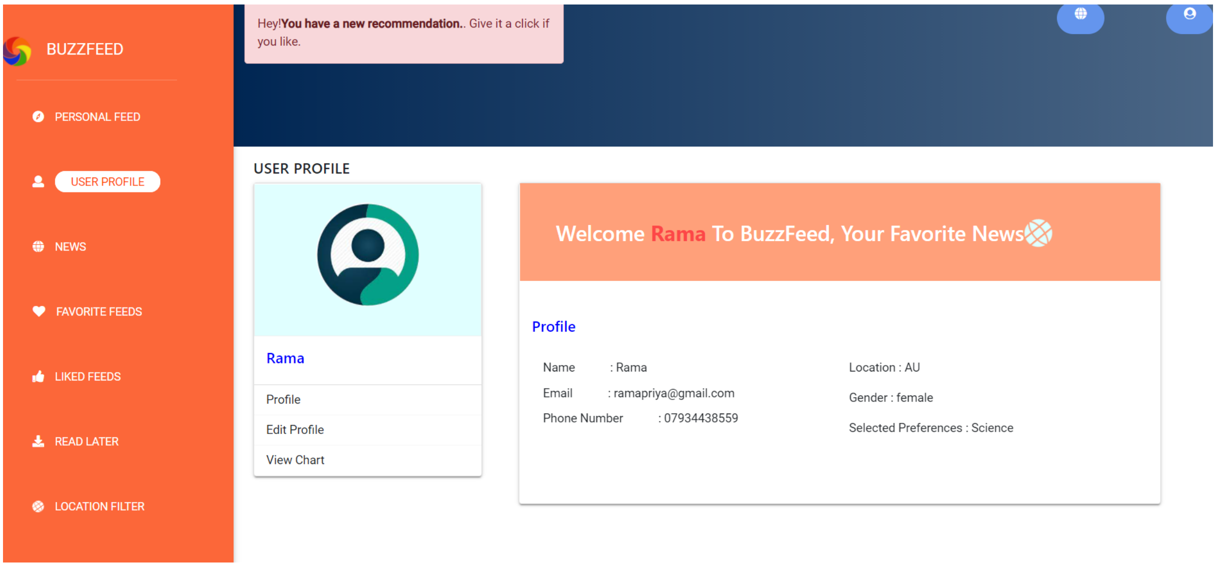

Figure 7.3: Recommendation

Evaluation is all done with Observation testing and Test Driven Development and fixed some of the bugs which popped up during the process and below are the successful test-cases.

7.3 Future Work

The application developed right now resolves all major stakeholder requirements and successfully available as a working prototype which can be further enhanced and developed to deliver it as a product out in the market. The possible enhancements could include some of the following.

- Enhance projections and visualization to make it more interactive and user-friendly.
- News dataset could have been taken with some popularity field to understand how much trending the news is.
- Recommendation model can further include item-user correlation and added to current user-user similarity to understand similar categories which can be correlated like health can be interlinked nutrition and so on. For this the news dataset may be extended further with license to have majority of categories data from same or

different third party APIs who support them.

- User profiling could be extended with options to track implicit user behavior and interests.

Bibliography

[1] 21, S. S. o. A. Use the facebook algorithm to create meaningful interactions, Jun 2021.

[2] Javascript charts and maps, Nov 2018.

[3] ANNE. New german edition gets deep personalization powered by machine learning, May 2018.

[4] Bhat, T., and Nagappan, N. Evaluating the efficacy of test-driven development: Industrial case studies. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (New York, NY, USA, 2006), ISESE '06, Association for Computing Machinery, p. 356363.

[5] DANIELS, C. Personalization deep dive: Contextual vs. behavioral, Apr 2017.

[6] Erin, Y. K. How to set up java spring boot jwt authorization and authentication, Dec 2020.

[7] Facebook news feed algorithm, Dec 2018.

[8] Feng, C., Khan, M., Rahman, A. U., and Ahmad, A. News recommendation systems - accomplishments, challenges amp; future directions. *IEEE Access 8* (2020), 16702–16725.

[9] Garrigs, I., Gomez, J., and Houben, G.-J. Specification of personalization in web application design. *Information and Software Technology 52*, 9 (2010), 991–1010.

[10] Gitsham, O. The new bbc news app good or bad user experience?, Jun 2015.

[11] Hamilton, T. What is usability testing? ux(user experience) testing example, Aug 2021.

[12] Jonnalagedda, N., Gauch, S., Labille, K., and Alfarhood, S. Incorporating popularity in a personalized news recommender system. *PeerJ Computer Science 2* (06 2016), e63.

[13] Lee, Y. Recommendation system using collaborative filtering. Master's Projects. 439.

[14] Li, L., Wang, D.-D., Zhu, S.-Z., and Li, T. Personalized news recommendation: A review and an experimental investigation. *Journal of Computer Science and Technology 26*, 5 (Sep 2011), 754.

[15] Li, M., and Wang, L. A survey on personalized news recommendation technology. *IEEE Access 7* (2019), 145861–145879.

[16] Liu, J., Pedersen, E., and Dolan, P. Personalized news recommendation based on click behavior. In *2010 International Conference on Intelligent User Interfaces* (2010).

[17] Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. Recommender system application developments: A survey. *Decision Support Systems 74* (2015), 12–32.

[18] Madhushree, B. A novel research paper recommendation.

[19] McHale, B. What metrics are important when managing a project?, May 2021.

[20] Meguebli, Y., Kacimi, M., Doan, B.-l., and Popineau, F. Building rich user profiles for personalized news recommendation. In *UMAP Workshops* (2014).

[21] Mishra, U. What is a content-based recommendation system in machine learning?

[22] Natarajan, S., and Moh, M. Recommending news based on hybrid user profile, popularity, trends, and location. In *2016 International Conference on Collaboration Technologies and Systems (CTS)* (2016), pp. 204–211.

[23] News api search news and blog articles on the web.

[24] Ngo, N. Android software development: Case:logo quiz world mobile application.

[25] Or, C., and Tao, D. Usability study of a computer-based self-management system for older adults with chronic diseases. *JMIR research protocols 1*, 2 (Nov 2012), e13–e13. 23612015[pmid].

[26] Rao, R., and Swamy, S. Review on spring boot and spring webflux for reactive web development.

[27] Riihiaho, S. *Usability Testing*. John Wiley & Sons, Ltd, 2018, ch. 14, pp. 255–275.

[28] Seminario, C., and Wilson, D. Case study evaluation of mahout as a recommender platform. vol. 910.

[29] Spring boot architecture - javatpoint.

Appendix A

Computational Code

Some Code snippets that includes the few logical functionalities of the application for review as below.

Some backend service classes in Java Springboot which performs the logical programming part.

User Feed service class where the algorithm works gathering categories and updating respective weights and updating user category-weight map.

UserFeed Service

```
@Service
public class UserFeedService {

    private static final String[] HEADERS = {"Id", "Weight", "Category"};
    private static final CSVFormat FORMAT =
        CSVFormat.DEFAULT.withHeader(HEADERS);
    @Autowired
    private UserService userService;

    @Autowired
    private UserFeedRepository userFeedRepository;

    @Autowired
```

```
private DNewsService dNewsService;

//Adds likes to users and updates that news category weight to 1.0
public void addLikesToFeed(Integer uid, String categoryName) {

    Double weightByPreferences = 0.0;
    User user = userService.findUser(uid);
    HashMap<String, Double> categorymap = new HashMap<>();
    categorymap.putAll(user.getMap());

    List<String> likedList = new ArrayList<>();
    likedList.add(categoryName);
    for (Object category : likedList) {

        categorymap.put(category.toString(), user.getMap().get(category) +
            1.0);
        user.setMap(categorymap);



    }

    userService.addUser(user);
}

//Removes likes from users and updates that news category weight
public void removeUnlikes(Integer uid, String categoryName) {

    Double weightByPreferences = 0.0;
    User user = userService.findUser(uid);
    HashMap<String, Double> categorymap = new HashMap<>();
    categorymap.putAll(user.getMap());
```

```
        List<String> unlikeList = new ArrayList<>();
        unlikeList.add(categoryName);
        for (Object category : unlikeList) {
//
            categorymap.put(category.toString(), user.getMap().get(category) -
                1.0);
            user.setMap(categorymap);



        }


        System.out.println("after setting map" + user.getMap());
        userService.addUser(user);
    }
```

User feed service fetching personalized news feed based on ranking with weights and filtering in algorithm.

UserFeed Service

```
    //Fetched user feed personalized by category-weight map
    public List<DNews> fetchByUid(Integer uid) {
        List<DNews> news = new ArrayList<>();
        User user = userService.findUser(uid);
        HashMap<String, Double> categorymap = new HashMap<>();
        categorymap.putAll(user.getMap());
        //convert to sorted hashmap - code taken from online sample comparator
        List<Entry<String, Double>> list = new LinkedList<Entry<String,
            Double>>(categorymap.entrySet());
        Collections.sort(list, new Comparator<Entry<String, Double>>() {
            public int compare(Entry<String, Double> o1, Entry<String, Double>
                o2) {
```

```
                return o2.getValue().compareTo(o1.getValue());

            }
        });
//prints sorted map
        Map<String, Double> sortedMap = new LinkedHashMap<String, Double>();
        for (Entry<String, Double> entry : list) {
            //Also check not 0 to filter not interested
            if (entry.getValue() > 0.0) {
                sortedMap.put(entry.getKey(), entry.getValue());
            } else {
                continue;
            }
        }
        List<String> keys = new ArrayList<>(sortedMap.keySet());
        Collection<Double> values = sortedMap.values();



        //filter based on weight
        for (Map.Entry<String, Double> entry : sortedMap.entrySet()) {
            System.out.println(entry.getKey() + "and" + entry.getValue());
            if (entry.getValue() > 50) {
                news.addAll(dNewsService.findNewsByCategoryWeight(entry.getKey(),
                    20));
            } else if (entry.getValue() > 10) {
                news.addAll(dNewsService.findNewsByCategoryWeight(entry.getKey(),
                    12));
            } else if (entry.getValue() > 5) {
                news.addAll(dNewsService.findNewsByCategoryWeight(entry.getKey(),
```

```
                8));

        } else {
            news.addAll(dNewsService.findNewsByCategoryWeight(entry.getKey(),
                5));

        }
    }


    return news;
```

News Service

```
@Service
public class DNewsService {

    @Autowired
    private DNewsRepository dNewsRepository;



    public DNews findNews(Integer id) {
        return dNewsRepository.findById(id);
    }

    //Gets all news articles from database for general newsfeed
    public Iterable<DNews> getAll() {
        return dNewsRepository.findAllByOrderByPublishedAtDesc();
    }

    //Add news to database
    public Iterable<DNews> addNews(List<DNews> news) {
```

```
        return dNewsRepository.saveAll(news);
    }

    //Find news by country
    public List<DNews> findNewsByCountry(String country) {
        return dNewsRepository.findByCountry(country);
    }
```

Below Springboot startup code runs the springboot application and also performs conversion of json news objects to database objects using Command line runner.

SpringBoot application startup

```
@SpringBootApplication
public class ProjectApplication {

    public static void main(String[] args) {
        System.out.println("Welcome to Newsfeed");

        SpringApplication.run(ProjectApplication.class, args);
    }

    //Referred for data binding support from
        https://www.danvega.dev/blog/2017/07/05/read-json-data-spring-boot-write-database/
    @Bean
    CommandLineRunner runner(NewsApiService newsApiService, DNewsService
        dNewsService) {
        return args -> {


            ObjectMapper obj_mapper = new ObjectMapper();
            //Code script referred from
                https://www.danvega.dev/blog/2017/07/05/read-json-data-spring-boot-write-datal
```

```
        TypeReference<List<DNews>> typeRef = new
            TypeReference<List<DNews>>() {
        };
        InputStream input =
            TypeReference.class.getResourceAsStream("/json/DetailedNews.json");
        try {
            List<DNews> news1 = obj_mapper.readValue(input, typeRef);
            dNewsService.addNews(news1);
            System.out.println("News Saved with details!");
        } catch (IOException e) {
            System.out.println("Unable to save news details: " +
                e.getMessage());
        }
    };
}
```

Sample Controller classes to explain api calls at backend.

Login Controller with JWT token generation code

Login Controller

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class LoginController {

    @Autowired
    private JWTUtil jwtUtil;

    @Autowired
    private JWTUserDetailsService jwtUserDetailsService;



    @Autowired
```

```
private AuthenticationManager authenticationManager;



@CrossOrigin(origins = "*", allowedHeaders = "*")
@GetMapping("/")
public String greet() {
    return "Welcome !!!!";
}

//authenticate users with username and password and generates token if the
    successful authentication.
@CrossOrigin(origins = "*", allowedHeaders = "*")
@PostMapping("/authenticate")
public String generateToken(@RequestBody AuthRequest authRequest) throws
    Exception {
    try {
        authenticationManager.authenticate(
                new
                    UsernamePasswordAuthenticationToken(authRequest.getUserName(),
                    authRequest.getPassword())
        );
    } catch (Exception ex) {
        throw new Exception("invalid username/password");
    }

    final UserDetails userDetails = jwtUserDetailsService
            .loadUserByUsername(authRequest.getUserName());



    return jwtUtil.generateToken(userDetails);
}
```

User Feed Controller where it recommends user-user similarity of interests based articles.

UserFeed Controller

```
//Recommends users based on similar user interests pattern.
@GetMapping("/csv/{uid}")
public ResponseEntity<List> downloadUsersCSV(@PathVariable Integer uid)
    throws IOException {
    List<DNews> recommendedNewsList = new ArrayList<>();
    List<String> categoriesList = new ArrayList<>();
    List<List<String>> userlist = userFeedService.recommendService();
    //Writes the all users category-map information from database to csv
        file
    String path =
        "C:/Users/sittu/IdeaProjects/MSCproject/src/main/resources/static/";
    FileWriter csvWriter = new FileWriter(path + "newFile.csv");
    System.out.println("user lis" + userlist);
    for (List<String> rowData : userlist) {
        if (!rowData.get(2).equals("0.0")) {

            csvWriter.append(String.join(",", rowData));
            csvWriter.append("\n");
        }
    }

    csvWriter.flush();
    csvWriter.close();

    //Fetches the user category-map information and recommends the
        articles to users based on
    //recommendation model which is calculated based on UserSimilarity
    try {
        //Creating data models
```

```
DataModel datamodel = new
    FileDataModel(ResourceUtils.getFile("C:/Users/sittu/IdeaProjects/MSCproject/s
    //data

UserSimilarity usersimilarity = new
    PearsonCorrelationSimilarity(datamodel);
UserNeighborhood neighbor = new ThresholdUserNeighborhood(0.1,
    usersimilarity, datamodel);
Recommender recomender = new GenericUserBasedRecommender(datamodel,
    neighbor, usersimilarity);

List<RecommendedItem> recommendations = recomender.recommend(uid,
    2);

if (recommendations != null) {
    for (RecommendedItem recommendation : recommendations) {
        System.out.println(recommendation.getItemID());
        System.out.println(recommendation.getValue());
        if (recommendation.getItemID() == 1) {
            categoriesList.add("science");
        } else if (recommendation.getItemID() == 0) {
            categoriesList.add("business");
        } else if (recommendation.getItemID() == 2) {
            categoriesList.add("health");
        } else if (recommendation.getItemID() == 3) {
            categoriesList.add("technology");
        } else if (recommendation.getItemID() == 4) {
            categoriesList.add("entertainment");
        } else {
            categoriesList.add("sports");
        }
```

```
                }
            }

        } catch (Exception e) {
        }

        //Recommended list is created based on weights of categories.
        recommendedNewsList =
            dNewsService.findNewsByRecommendation(categoriesList);

        return new ResponseEntity<List>(recommendedNewsList,
            HttpStatus.CREATED);
    }
```

JPA Native query fetching required news data based on parameters.

News Repository

```
//News Repository to query from DNEWS Database.
@Repository
public interface DNewsRepository extends JpaRepository<DNews, Long> {
    DNews findById(Integer id);

    List<DNews> findAllByOrderByPublishedAtDesc();

    //JPA native query to fetch 30 country based news where country is input
        parameter
    @Query(value = "SELECT * FROM DNEWS WHERE COUNTRY=:country ORDER BY
        PUBLISHED_AT DESC LIMIT 30", nativeQuery = true)
    List<DNews> findByCountry(String country);
```

```
    //JPA native query to fetch 10 country and category specific news
    @Query(value = "SELECT * FROM DNEWS WHERE COUNTRY=:country and
        CATEGORY=:category ORDER BY PUBLISHED_AT DESC LIMIT 10", nativeQuery =
        true)
    List<DNews> findByCountryCategory(String country, String category);

    //JPA native query to fetch limited news based on preferences by user
        which is measured in weights in algorithm
    @Query(value = "SELECT * FROM DNEWS WHERE CATEGORY=:category ORDER BY
        PUBLISHED_AT DESC LIMIT :count", nativeQuery = true)
    List<DNews> findNewsByCategoryWeight(String category, Integer count);

    //JPA native query to recommend different category news to users based on
        user-user similarity
    @Query(value = "SELECT * FROM DNEWS WHERE CATEGORY IN :category ORDER BY
        PUBLISHED_AT DESC LIMIT 10", nativeQuery = true)
    List<DNews> findNewsbyrecommend(List<String> category);
```

Security Configuration in Springboot

Spring Security

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(
        prePostEnabled = true,
        securedEnabled = true,
        jsr250Enabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private JWTUserDetailsService jwtUserDetailsService;
    @Autowired
```

```
private JWTFilter jwtFilter;

@Override
protected void configure(AuthenticationManagerBuilder auth) throws
    Exception {
    auth.userDetailsService(jwtUserDetailsService);
}

@Bean
public PasswordEncoder passwordEncoder() {
    return NoOpPasswordEncoder.getInstance();
}

@Bean(name = BeanIds.AUTHENTICATION_MANAGER)
@Override
public AuthenticationManager authenticationManagerBean() throws Exception {
    return super.authenticationManagerBean();
}

//Spring Security Implementation part concepts taken from Spring.io
    official document guides.
//This allows to authorize the functionalities api
@Override
protected void configure(HttpSecurity http) throws Exception {

    http

            .csrf().disable().authorizeRequests().antMatchers(HttpMethod.POST,
                "/authenticate/", "/signin", "/api/Users",
                "/api/UserFeed/**")
            .permitAll()
```

```
            .antMatchers(HttpMethod.GET, "/", "/api/News/news",
               ".api/UserFeed/recommend").permitAll()
            .anyRequest().authenticated()
            .and().exceptionHandling().and().sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
            .httpBasic();
      http.addFilterBefore(jwtFilter,
         UsernamePasswordAuthenticationFilter.class);
      http.cors();
   }
```

Frontend Code to display data in the client accomplished with Angular, HTML, bootstrap, CSS.

Globe data showing specified country information and category filter in addition to country achieved using amcharts and Behavioral Subjects in angular.

Global Visulization

```
export class GlobaldataComponent implements OnInit {

  //This component projects the news data into the country and category
     location filter based on amcharts.
  //It has functions thaat loads the globe and filter the news based on
     selected country and category information.
  newsByCountry: News = {};
  constructor(private apicall: ApicallsService, private authservice:
     AuthenticateService) { }

  test: any = "all";

  // Make category name store Observable
  public categoryname$ = new BehaviorSubject<string>("");
```

```
  // Setter to update category
  setcategoryname(val: string) {
    this.test = val;
    this.categoryname$.next(val);
    console.log("category is", this.categoryname$)
  }
  id = parseInt(localStorage.getItem('ID'));
  country: any = "AR";
  ngOnInit(): void {

    if (this.id) {
      this.getUser()
    }
    else {
      this.loadglobe(this.country);
    }

    this.test = "all";



  }



  user: User;

//Get user details for populating news with respective location of users
  getUser() {
    this.apicall.getUserDetails(this.id).subscribe((data) => {
      this.user = data;
      if (this.user.location) {
        this.country = this.user.location;
```

```
      console.log("user country", this.country)

    }
    console.log("user data", this.user)
    this.loadglobe(this.country)
    return data;

  });
  console.log("users country", this.country)

}

//Basic structure from amcharts to load the globe with polygon series
loadglobe(country) {
  var globe = am4core.create("chartdiv", am4maps.MapChart);
  globe.panBehavior = "rotateLongLat";
  globe.geodata = am4geodata_worldLow;

  globe.projection = new am4maps.projections.Orthographic();

  var pseries = globe.series.push(new am4maps.MapPolygonSeries());

  pseries.useGeodata = true;


  let animate;
  //Rotate globe animation
  setTimeout(function () {
    animate = globe.animate({ property: "deltaLongitude", to: 90000 },
        19000000);
  }, 3000)
```

```
globe.seriesContainer.events.on("down", function () {
  if (animate) {
    animate.stop();
  }
})




var polygonTemplate = pseries.mapPolygons.template;
polygonTemplate.strokeWidth = 0.5;

polygonTemplate.fill = am4core.color("#6ACF90");
polygonTemplate.stroke = am4core.color("#ffffff");





polygonTemplate.tooltipText = "{name}";
var hs = polygonTemplate.states.create("hover");
hs.properties.fill = am4core.color("#C70039");
pseries.mapPolygons.template.events.on(
  'hit',
  async (ev: any) => {
    if (ev) {
      country = ev.target.dataItem.dataContext.id;
    }
    console.log("selected ", country);

    if (this.test === "all") {
      this.apicall
```

```
        .getNewsByCountry(country)
        .subscribe((data) => {

          this.newsByCountry = data;

          console.log("filtered news by selected country", data,
              this.newsByCountry)
        });
    }
    else {
      this.apicall
        .getNewsByCountCat(country, this.test)
        .subscribe((data) => {

          this.newsByCountry = data;

          console.log("filtered news by selected country", data,
              this.newsByCountry)
        });
    }

  });

console.log("in globaldata")

if (this.test === "all") {
  this.apicall
    .getNewsByCountry(country)
    .subscribe((data) => {

      this.newsByCountry = data;
```

```
        console.log("filtered news by selected country", data,
            this.newsByCountry)
      });
  }
  else {
    this.apicall
      .getNewsByCountCat(country, this.test)
      .subscribe((data) => {

        this.newsByCountry = data;

        console.log("filtered news by selected country", data,
            this.newsByCountry)
      });
  }

//Based on changes of behavioral subject subscribe to changes in data
  this.categoryname$.subscribe(() => {
    if (this.test === "all") {

      this.apicall
        .getNewsByCountry(country)
        .subscribe((data) => {

          this.newsByCountry = data;

          console.log("filtered news by selected country", data,
              this.newsByCountry)
        });
    }
```

```
      else {

        this.apicall
          .getNewsByCountCat(country, this.test)
          .subscribe((data) => {

            this.newsByCountry = data;

            console.log("filtered inside news by selected country", data,
                this.newsByCountry)
          });
      }
    });
    globe.backgroundSeries.mapPolygons.template.polygon.fillOpacity = 2;
    // globe.backgroundSeries.mapPolygons.template.polygon.fill =
        am4core.color("#D7F6FC");
    globe.backgroundSeries.mapPolygons.template.polygon.fill =
        am4core.color("#9AE7FD");

//Country data into amcharts
    pseries.data = [{
      "id": "CA",
      "name": "Canada",
      "fill": am4core.color("#8B0000")
    }, {
      "id": "AU",
      "name": "Australia",
      "fill": am4core.color("#008000")
    },
    {
      "id": "NZ",
```

```
    "name": "New Zealand",
    "fill": am4core.color("#140B8A")
  }];

  polygonTemplate.propertyFields.fill = "fill";




}
```

User Profile with user data is displayed and editable in frontend with category weight in pie-charts visualization.

User Profile and chart Visualization

```
export class UserProfileComponent implements OnInit {

  //Collects the user profile information and projects them in the frontend
      and allows users to edit anytime.
  logIn: boolean =false;



  constructor(private apicall: ApicallsService, private authenticate :
      AuthenticateService) {

    this.loadprofile = true;
  }
   id= parseInt(localStorage.getItem('ID'));
   user: User;
   loadprofile = true;
   editprofile = false;
   chartsdata = false;

```

```
 locations = [
  { name: 'United Stastes of America', code: 'us' },
  { name: 'United Kingdoms', code: 'gb' },
  { name: 'India', code: 'in' }
];




currentUser: Object = {};
 successupdate: boolean =false;

ngOnInit(): void {
  this.loadprofile = true;
  this.getUser()

}

//To get user details for providing personal information
getUser(){
  this.apicall.getUserDetails( this.id ).subscribe((data) =>{
    this.user = data;
    console.log("user data", this.user)
    return data ;
});
return this.user;
}

//To load the respective user profile
loadProfile(){
  this.loadprofile = true;
```

```
  this.editprofile = false;
  this.chartsdata = false;
}

//Allow to edit the profile
editProfile(){
  this.editprofile = true;
  this.loadprofile = false;
  this.chartsdata = false;
}



//Update the user data in server
updateUser(){

  this.apicall.updateUser(this.user,this.id).subscribe(data => {
    console.log("current user" + this.user)
    this.successupdate=true;
  });

}



//To check if user logged in
isLoggedIn(){
  this.logIn = this.authenticate.isUserLogged();



 return this.authenticate.isUserLogged();
}
```

```
  //Load user charts to show category-weight rate
  loadcharts(){
    this.user=this.getUser()
    console.log("user data" + this.user)
    this.editprofile = false;
    this.loadprofile = false;
    this.chartsdata = true;

var chart = am4core.create("chartdiv", am4charts.PieChart);

// Add data
chart.data = [{
  "category": "Business",
  "value": this.user.map['business'],
  "color": am4core.color("#cf0a80")
}, {
  "category": "Entertainment",
  "value": this.user.map['entertainment'],
  "color": am4core.color("#a72dbd")
}, {
  "category": "Sports",
  "value": this.user.map['sports'],
  "color": am4core.color("#631bf2")
}, {
  "category": "Health",
  "value": this.user.map['health'],
  "color": am4core.color("#24bbe0")
}, {
  "category": "Technology",
  "value": this.user.map['technology'],
```

```
  "color": am4core.color("#def03e")
},
{
  "category": "Science",
  "value": this.user.map['science'],
  "color": am4core.color("#ed1139")
}
];

// configure Series
var pieSeries = chart.series.push(new am4charts.PieSeries());
pieSeries.dataFields.value = "value";
pieSeries.dataFields.category = "category";
pieSeries.slices.template.propertyFields.fill = "color";

chart.legend = new am4charts.Legend();
chart.radius = am4core.percent(80);
```

Evaluation performed with Test Driven development and sample test case snippets. Sample User add, find and update testcases.

User Testcases

```
@SpringBootTest
@AutoConfigureMockMvc
@Import(SecurityConfig.class)
class UserControllerTest {



    @Autowired
    private MockMvc mockMvc;

    @MockBean
```

```
private UserService userService;

@Test
public void addUserTest() throws Exception {

    User user1 = new User();
    user1.setUid(1);
    user1.setEmailId("user1@gmail.com");
    user1.setPhoneNumber("0993004922");
    user1.setPassword("hry12@S");
    user1.setUserName("user");
    List<String> preferredCategoryList = new ArrayList<String>();

    preferredCategoryList.add("Health");
    user1.setPreferredCategory(preferredCategoryList);
    user1.setGender("Male");
    user1.setLocation(("NZ"));
    user1.setFavoriteList(null);
    user1.setLikedList(null);
    user1.setSavedList(null);

    when(userService.addUser(user1)).thenReturn(user1);

    assertEquals("user", userService.addUser(user1).getUserName());
}

@Test
public void findUserTest() throws Exception {

    User user1 = new User();
    user1.setUid(1);
```

```
        user1.setEmailId("user1@gmail.com");
        user1.setPhoneNumber("0993004922");
        user1.setPassword("hry12@S");
        user1.setUserName("user");
        List<String> preferredCategoryList = new ArrayList<String>();

        preferredCategoryList.add("Health");
        user1.setPreferredCategory(preferredCategoryList);
        user1.setGender("Male");
        user1.setLocation(("NZ"));
        user1.setFavoriteList(null);
        user1.setLikedList(null);
        user1.setSavedList(null);

        userService.addUser(user1);

        when(userService.findUser(1)).thenReturn(user1);

        assertEquals("user", userService.findUser(1).getUserName());
    }

    @Test
    public void findUserFailTest() throws Exception {

        when(userService.findUser(1)).thenReturn(null);

        assertEquals(null, userService.findUser(1));
    }



    @Test
```

```java
public void updateUserTest() throws Exception {

    User user1 = new User();
    user1.setUid(1);
    user1.setEmailId("user1@gmail.com");
    user1.setPhoneNumber("0993004922");
    user1.setPassword("hry12@S");
    user1.setUserName("user");
    List<String> preferredCategoryList = new ArrayList<String>();

    preferredCategoryList.add("Health");
    user1.setPreferredCategory(preferredCategoryList);
    user1.setGender("Male");
    user1.setLocation(("NZ"));
    user1.setFavoriteList(null);
    user1.setLikedList(null);
    user1.setSavedList(null);

    userService.addUser(user1);
    when(userService.findUser(1)).thenReturn(user1);
    assertEquals("user", userService.findUser(1).getUserName());



    user1.setUserName("user1updated");
    userService.addUser(user1);
    when(userService.findUser(1)).thenReturn(user1);
    assertEquals("user1updated", userService.findUser(1).getUserName());
}
```

Userfeed Controller tests that verifies all the interactions are working as expected with assert conditions.

User Feed Testcases

```
    public void addLikesTest() throws Exception {

        User user1 = new User();
        user1.setUid(1);
        user1.setEmailId("user1@gmail.com");
        user1.setPhoneNumber("0993004922");
        user1.setPassword("hry12@S");
        user1.setUserName("user");
        List<String> preferredCategoryList = new ArrayList<String>();
//        List<String> likedList = new ArrayList<String>();

        preferredCategoryList.add("Health");
        user1.setPreferredCategory(preferredCategoryList);
        user1.setGender("Male");
        user1.setLocation(("NZ"));
        user1.setFavoriteList(null);
        user1.setLikedList(null);
        user1.setSavedList(null);

        List<DNews> newsList = new ArrayList<>();
        Source source = new Source();
        source.setId("111");
        source.setName("Raguram Source");
        DNews news = new DNews();
        news.setId(1001);
        news.setAuthor("Raguram");
        news.setCategory("Health");
        news.setCountry("NZ");
        news.setTitle("Healthy atmosphere");
        news.setSource(source);
```

```
        newsList.add(news);

        //User Existence Test
        userService.addUser(user1);
        when(userService.findUser(1)).thenReturn(user1);
        assertEquals("user", userService.findUser(1).getUserName());

        //Algorithm feed service test
        userFeedService.addLikesToFeed(1, "Health");
//        likedList.add("Health");


        //Update user after liketo set his likedlist
        user1.setLikedList(newsList);
        userService.addUser(user1);
        when(userService.findUser(1)).thenReturn(user1);
        assertEquals(newsList, userService.findUser(1).getLikedList());

    }

    @Test
    public void addLikesFailTest() throws Exception {

        User user1 = new User();
        user1.setUid(1);
        user1.setEmailId("user1@gmail.com");
        user1.setPhoneNumber("0993004922");
        user1.setPassword("hry12@S");
        user1.setUserName("user");
        List<String> preferredCategoryList = new ArrayList<String>();
```

```
// List<String> likedList = new ArrayList<String>();

preferredCategoryList.add("Health");
user1.setPreferredCategory(preferredCategoryList);
user1.setGender("Male");
user1.setLocation(("NZ"));
user1.setFavoriteList(null);
user1.setLikedList(null);
user1.setSavedList(null);

List<DNews> newsList = new ArrayList<>();
Source source = new Source();
source.setId("111");
source.setName("Raguram Source");
DNews news = new DNews();
news.setId(1001);
news.setAuthor("Raguram");
news.setCategory("Health");
news.setCountry("NZ");
news.setTitle("Healthy atmosphere");
news.setSource(source);

newsList.add(news);

//User Existence Test
userService.addUser(user1);
when(userService.findUser(1)).thenReturn(user1);
assertEquals("user", userService.findUser(1).getUserName());

//Algorithm feed service test
userFeedService.addLikesToFeed(1, "Health");
```

```
        // likedList.add("Health");



        //Failed to Update user after liketo set his likedlist
        //user1.setLikedList(newsList);
        //userService.addUser(user1);
        when(userService.findUser(1)).thenReturn(user1);
        assertEquals(null, userService.findUser(1).getLikedList());

    }
```