

Book review Website

Rishik TS - 191IT254
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
rishik.tulaa@gmail.com

Shashi Prakash A - 191IT204
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
shashiprakash1729@gmail.com

Naveen R - 191IT133
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
veninavi36@gmail.com

Abstract—Project is identifying expert review in a book review website. Book review website is a site where a user can post his/her reviews on any book, get to know which is expert review out of existing reviews so that he/she can get a clear idea about the book. Django, HTML, CSS are used for front end ML for figuring out expert answer, SQLite3 to store the data are used in the backend

I. INTRODUCTION

Review means to look back and evaluate the work done by someone. Review of a book means to evaluate the work in question based on its strong and weak points, sometimes ending with a recommendation.

Purpose of a Review is basically to encourage observation, perception and general awareness both during and after experiences. Book reviews offer a useful insight into whether or not a book is genuinely worth reading.

Book review gives the user the opportunity to get an idea about what the author wants to convey and how efficient he was. Book reviews help potential readers become familiar with what a book is about, give them an idea of how they themselves might react to it and determine whether this particular book will be the right book for them right now.

We humans are interested to know pros, cons of a product before buying it. We want to know the good qualities and bad qualities of a product to figure out whether it suits us or not. Reading a review sometimes is like a test drive on the book.

Reviews are useful and important because

- They save Time, decrease Risk to Readers
- There is Greater Visibility, Greater Chance of Getting Found
- For an author, book reviews can open doors to new and bigger audiences
- More sales are possible (since people get to know more about the book)

Reviews not only have the power to influence consumer decisions but can strengthen a company's credibility. Reviews have the power to gain customer trust, and they encourage people to interact with the company. Customer interaction ultimately leads to improved profits for businesses.

II. LITERATURE SURVEY

[1] ANSWER QUALITY EVALUATION

Although these studies focus on answers quality rather than their authors, influencing factors of answer quality provide references for identifying high-quality authors, namely, experts. **Hapnes** et al. (2014) find that most high-quality answers are well structured long texts with convenience facts or evidence. Answer texts have abundant information, and making good use of the text information might be conducive to identify high-quality authors. **Chirag** et al. (2010) find that user profiles, such as the number of stars received, level achieved, are critical in evaluating answer quality. Therefore, user profiles should also be considered when identifying experts. In this paper, we try to make full use of texts and use different methods to process answer keywords and full-texts separately. User profiles, such as gender, numbers of followers, are represented as digits and concatenate with text features.

No. of followers was considered in our ML model as one of the attributes to predict expertity in Linear Regression.

[2] IDENTIFICATION OF OPINION LEADERS OR EXPERTS

The difference between opinion leader identification and expert identification is that opinion leader identification is usually inseparable from social networks. **Momtaz** et al. (2011) analyze the centrality, structural holes, and indegree to identify opinion leaders. **Zhang** et al. (2016) find that online social network includes many kinds of relationships, and propose a node importance analysis in the multirelationship online social network. These methods focus on network structure and relationships among users.

However, it is not enough for expert identification to use only the social network without QA text.

III. PROBLEM STATEMENT

Building a Book review website where a user can read, post reviews, react to the reviews and get to know the expert review out of given reviews.

IV. WORK DONE

The book review web app made by us supports user authentication by login and sign up.

Signup page looks like this. It is an animation to improve user experience with the website. Animation has effects which will make it look like a book is opening when the user opens the page.

Fig. 1: Sign Up page

Basic details like username, password are asked and user can edit other info like name and bio in the after login.

Input validation takes place here. It accepts the password only if it satisfies the following conditions that Password must contain:

- Atleast one lowercase character (a-z)
- Atleast one uppercase character (A-Z)
- Atleast one digit (0-9)
- Length must be atleast 6

In the same way, login page looks like this which has similar animation of a book opening when user opens the page.

Fig. 2: Login page

User can post a review for the existing books that are posted by the admin. Admin will post the new book whenever it comes into the market. User. Posts page for admin looks something like this

Fig. 3: Posts

So, here we can see that a user "ris", "snr", "shashi" posted reviews to this book.

Admin has the authority to delete books, reviews. But normal users can just post and react to reviews.

This was done to maintain the website in a proper way to prevent malicious users to spoil the page by deleting others reviews.

Reviews are ordered according to their expertise:

All the reviews of the posts are ordered in the order of their expertise.

Expertivity of each review is predicted using Machine learning and finally they are displayed in the decreasing order.

So, review which is the expert review out of all other reviews. It takes in 2-3 attributes to predict that which we'll see later in Methodology part.

When user clicks on "My Profile", he can see the following page.

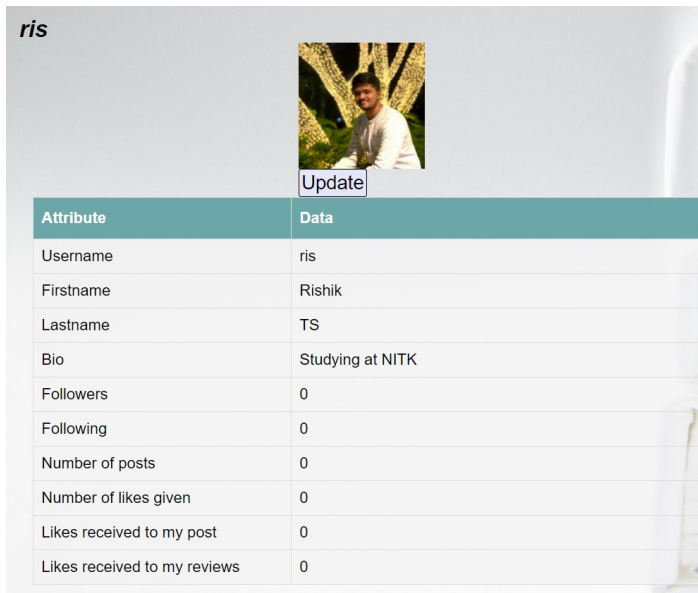


Fig. 4: My Profile

If user follows someone and if he is followed by others, then he can even see the list of followers and he can visit their profiles too.

Followers	
testuser45	View Profile
jayaprakash	View Profile

Fig. 5: Followers data

Users can Follow or Unfollow other users by visiting their profile.

Users can also goto their profile from the posts page by clicking on the review also.

Every User has two 'ManyToManyField's (followers,you_follow) in the database schema.So that the other users can be linked as a foreignkey.

There also exists a relation between users i.e., Relationships which is created if there is relation between two users .A Relationships object is created when a user follows another user with a status code 'follow'.

Users profiles give us an info about his reputation, expertise in giving expert reviews i.e., no. of likes received to his review, no. of reviews he/she posted.

Note :

- Every page created is responsive.
- Every page contains Navbar as header and a footer also. Navbar can be used to navigate to Home page, Posts page, My Profile, Log Out, Search for a book.

Footer also has some links and Copyright, Contact Us.

V. METHODOLOGY

See the below flowchart to get an idea of how exactly the website works.

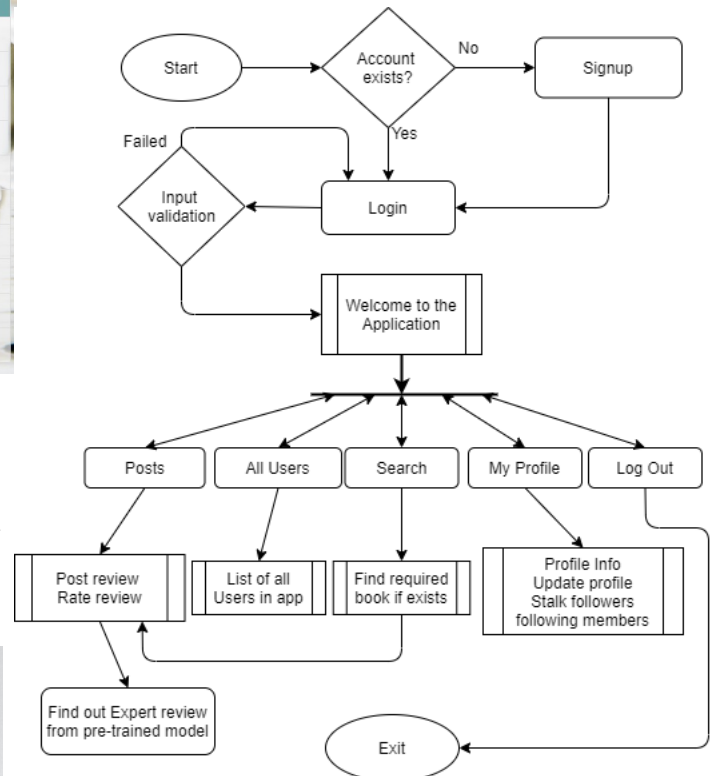


Fig. 6: Flowchart diagram

Entire project is divided into some sections:

A. Backend

Backend basically consists of database which was built using SQLite, Machine Learning concepts to figure out expert review,

In the Backend part, Django's default 'dbsqlite3' is used.

Tables created:

1.Profile (to store user details)

Fields taken directly from user

-firstname,lastname,emailid,bio,avatar,country

Fields filled after processing the user data

-followers,you_follow,exp,slug(a unique string for every user)

2.Relationship (to establish relation between users)

Fields

-Sender (user who wants to follow),Receiver (user who is being followed)

3.Post (to store a book, which is called as post most commonly) (Edit only by Admin)

Fields taken as input given by Admin

-caption,book_image(upload the image of the book),book_name,book_author

Fields filled by actions of Admin/Other users

-created(post created date,time),updated(updated date,time),liked(a 'ManyToManyField' of Profile (user table))

4.Review (all reviews)

Fields

-user,post (foreign keys of users who are filling the review,to which book respectively),body (review given by the user),expertivity (to store the review's score by the user's details)

5.Post_like

-user,post (foreign keys of users who are liking/disliking the book,to which book respectively),value (Like/Unlike)

6.Review_like

-user,review (foreign keys of users who are liking/disliking the review,to which review respectively),value (Like/Unlike).

ML work:

Regarding ML, We got dataset from the research paper, we removed some columns and made it suitable to our website.

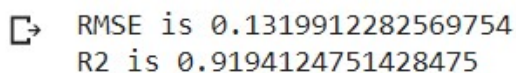
In ML, We considered our X to have follower count, total no.of upvotes that user received for all of his reviews/posts. Y had the expertivity of the corresponding user.

We used train test split to split the entire dataset into 80% training 20% test set for which we test the accuracy of our model.

We have used standard scalar to standardise the data of each column to make the data internally consistent. This ensures no bias to columns with values of larger magnitude and smaller magnitude. This will convert the column such that mean of all the values is 0.

Now, we trained the model using this Xtrain (followers count, total no.of upvotes) and Ytrain (his expertivity in giving reviews).

Upon testing our model with the test set, It gave us the following result.



```
RMSE is 0.1319912282569754
R2 is 0.9194124751428475
```

Fig. 7: RMSE and R2 on test set

But while using this ML on our website, we cant train the model always. It wastes time. So, we saved the model after it gets trained as "model" in file format. This was done by using pickle.dump() function belonging to pickle library.

The same was done even with standard scalar. Since, while we use this model to predict the expertise to some dynamic data, we need to scale it as model was trained to values of that magnitude. So, we import both model and standard scalar object (sc) into predict.py by using pickle.load() function belonging to pickle library.

So, train.py will train the model and saves model, standard scalar object (sc). These are used in predict.py to predict

expertise of dynamic values that we give from Django.

When a user posts a new review, his/her details like total no. of followers, total upvotes he/she received for all of his/her reviews are passed as arguments to predict.py which returns the expertivity of that user by using the ML model which we trained eariler. So, basically it uses a pretrained model to predict the expertivity.

Expertivity is also an attribute to each review by which we order them finally on the page.

In this way, we will be showing the user the expert review out of given reviews.

Its accuracy keeps on increasing as more and more people use it because users will be reacting to it, liking, disliking. In this way, prediction of expertise from dynamic values goes on improving day by day as people use it.

Django work:

This basically explains how the web pages, connectivity, information flow between them is achieved.

3 Django apps are created namely miniproj, posts, profiles.

[1] miniproj:

mini proj is like parent to the other two. When we launch the server, this miniproj app is executed first and it renders a home page where user is greeted and asked to login or signup.

In Signup: Form (method POST) is used to capture the details given by the user. Right now to keep it simple, we asked only Username, password. Input validation is kept to ensure security.

When the "Submit" button is clicked, checkForm() function is called which checks for the valid inputs. We kept some constraints on password which was told above and username cant be empty. In Login: User can enter his details and after user authentication, the user is signed in.

User Authentication:

User Authentication is provided so that we can keep track on the number of users liking the review with no duplicate likes from a single user which can lead to false outputs for the reviews' expertivity variable. Used Django's default User and connected it with our custom Profile for more details of the user.You can sign-up with only a unique username and password, then later on can fill up additional details you want to fill by updating the profile.User's details are ofcourse secure since Django uses the PBKDF2 algorithm with a SHA256 hash, a password stretching mechanism recommended by NIST.

After logging in, it greets the user.

Every page has navbar(by which user can go anywhere from everywhere) and footer(which has some links, copyright, Contact Us).

navbar and footer are written in a folder from where we use it as template for all the pages.

basic.html is the main page in which we import "style.css" and "home.css" from staticproject folder which serves as a common template for all the pages.

More or less in all the webpages that we created will extend

basic.html.

This has been done to do the work in an organised way and a small change here will reflect in all other pages.

[2] Posts:

Posts app has all the code for the posts page, search bar and its results page.

classes and ids are properly used and linked to css. In this way, styling the page is achieved.

If a user puts the cursor on a post or review it gets highlighted, which was done by using hover.

main.html in posts app is the web page which consists the code for this posts page. for loop, if statements are used to display all the posts, reviews.

Objects of classes are used which stores the info of each review (no. of likes, book name, review, review by which user) etc.

Appropriate functions are defined in views.py belonging to posts app which gets executed when user searches something or likes or dislikes.

[3] Profiles:

Profiles app has all code for the Profiles of users. This handles User's data. Profiles usually accesses/stores/edits the User's data to database. All the operations like following a User is done here and presenting his/her details if logged in (appropriate functions are present in Profiles/views.py). Also shows all other User's profiles (users_list.html) & can see logged in user's followers' profiles with their details been rendered to followers.html, also if the logged in user wants to view Users those who he/she follows are rendered to you_follow.html.

B. Frontend

Frontend basically is made using HTML, CSS (for styling), JavaScript and Django (For routing between various apps created above).

All web pages import basic.html which serves as a template for all pages like navbar, footer, background image are the similar for all. Only the content changes in each page.

Front end basically has many web pages:

Home:

This page is the home page for the app. As user is not logged in it has below functionalities:

Login: When clicked on the Login button the login page opens up and book will be visible with one of its pages as a login page (for desktop sized screens) or a normal login page will be visible (for mobile sized screens). For desktop sized screens a flipping book is visible for which the flipping effects are done by pure css using transform property with which it can rotate along the y direction value. A login page will be visible on single page of the book with username and password entry values. If we check the 'show password' checkbox then through javascript password input type attributes' value will change from password to text type. If u do not have prior account then click on the Signup link visible at the bottom of the login page which redirects you

to the signup page. If we click on cancel button we will be redirected to the home page. If we click on the submit button it will open the home page with our account logged in.

Signup: When clicked on the Sign Up button the login page opens up and book will be visible with one of its pages as a Sign Up page (for desktop sized screens) or a normal Sign Up page will be visible (for mobile sized screens). For desktop sized screens a flipping book is visible for which the flipping effects are done by pure css using transform property with its rotateY() value different for every flipping page. A sign up page will be visible on single page of the book with username, password and confirm password entry values. If we check the 'show password' checkbox then through javascript password input type attributes' value will change from password type to text type. If u have a prior account then click on the login link visible at the bottom of the sign up page which redirects you to the login page. If we click on cancel button we will be redirected to the home page. If we click on the submit button it redirects you to the login page to login to the newly created account.

After he/she is logged in, it shows other links like:

- Posts: Where user can enter and use the book review features like see and react to reviews.

Posting new books is the functionality given only to the admin. All others can only react to the posts which are posted by the admin.

When this page is opened, all posts and reviews are passed to this page. A nested for loop is used which iterates on posts and reviews. Reviews belonging to appropriate posts (i.e., when the condition review.postid=postid is satisfied) are printed on the screen with appropriate styling. Actions like, dislike are kept in forms whose click will trigger the functions that are written in views.py of posts Django app that will update the database.

- All Users: Where user can open, stalk and follow all the users that have accounts in this website.

This basically contains all the users userids and link to their profiles in a table.

This was achieved using a for loop on profiles (which was fetched from the database). When user clicks on "View profile" of a user, specific-profile-view() function is called which is defined in views.py of profiles Django app.

This page also has a search bar where user can type the user which he/she is searching for. If user is not found, it reports the same.

- Search bar: Where user can search for a particular book and access the reviews (See and react to them).

Here, search bar on the navbar is a form that takes the name of the book (type=text) and sends the same to function called search_book_list_view where the book name is searched in the database. The function redirects the ID of the first book that matches with given text to another function called show_book_view by giving additional parameter in the URL which renders the book details to search_book.html.

- My Profile: Where user can open his/her profile, Update it

(change first and last name, bio, add profile picture), View his followers and view whom he is following and access their profiles.

User details like display picture, bio, name, username, followers, following, likes, posts are fetched by using appropriate attributes defined in the class whose object defines a user.

So, we fetch the object and call the attributes and fetch the info and post it on the page.

Update functionality is provided which gets visible when user clicks on update, javascript written in basic.html will make the display to "block" which was actually hidden. Now, user can update his/her details like Name, Bio, Display picture and press Update so that database gets updated.

- Log Out: To log out of the website.

So, basically these are the functionalities of our web app. User can signup, login, stalk profiles, post review, react to review, get to know the expert review.

VI. EXECUTION AND RESULTS

For demonstration, Our project can be executed by using the command "python3 manage.py runserver" since our project is done by using django. A django server is created at "http://127.0.0.1:8000/" address and the application launches. As soon as it launches, user can create account (if he is not having) and login to access our application in an effective way. In posts page, user can find out the expert review very easily. Since all the reviews are ordered in the order of their expertise which is governed by a number "expertivity" which is predicted by Machine learning code.

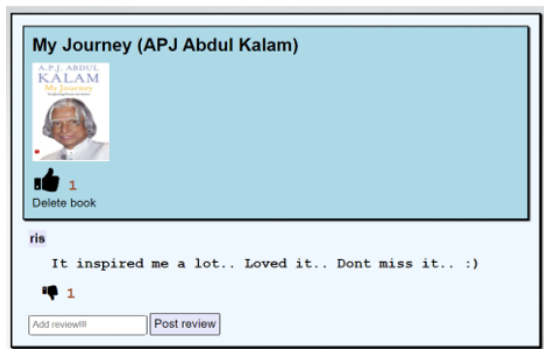


Fig. 8: Before SNR posted his review

When a new review was posted by "SNR", it placed it at first place because its expertivity is high. SNR's followers and total upvotes are relatively high. So, ML model had predicted its expertivity higher than review posted by ris. So, when a review with high expertivity comes, it is inserted at an appropriate place such that the sorting is not violated.

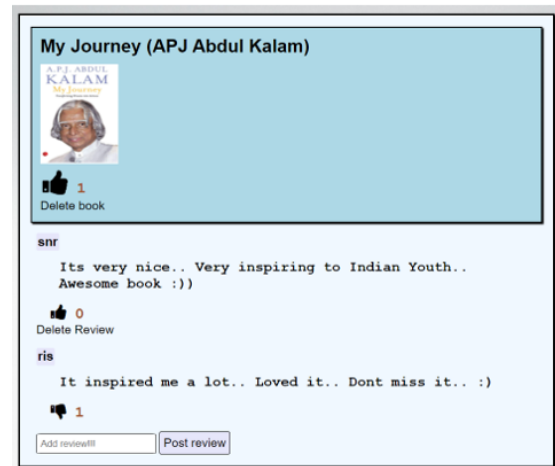


Fig. 9: After SNR posted his review

VII. CONCLUSION

We were successful in creating a web application providing a good user interface. User can create account, login, post his review or get to know the expert reviews and make a choice whether to read it or not.

User can also stalk other profiles, follow or unfollow.

User can like, dislike the reviews also.

This can be extended to a real life application by hosting it on a real web page and real life human beings can access it, get the advantages of using it.

In future, a chatbox can also be placed so that users can chat among themselves and get connected with people who have similar interests.

There is no rule that books must be english only, there can be any language book. When a new user comes in, his expertise and credibility will increase day by day as he posts his reviews and he receives likes for them.

VIII. REFERENCES

[1] Identifying Experts in Community Question Answering Website Based on Graph Convolutional Neural Network research paper by CHEN LIU¹, YUCHEN HAO¹, WEI SHAN^{2,3}, AND ZHIHONG DAI⁴ <https://ieeexplore.ieee.org/document/9151135>.

[2] W3 Schools <https://www.w3schools.com/>.

[3] Django documentation <https://docs.djangoproject.com/en/3.2/>.

[4] File(image) uploading <https://simpleisbetterthancomplex.com/tutorial/2016/08/01/how-to-upload-files-with-django.html>.