Indian Institute of Technology Bombay

Database Web Application: GoTweet

Authors:
Shubham Jadhav
(130050011)
Akash Garg
(130070060)
Leena Madhuri
(130050078)

Supervisor:
Prof. N.L. Sarda

November 7, 2015

Acknowledgement

Simply put, we could not have done this work without the help we received from Prof. N.L Sarda. The entire learning experience was really awesome. We learned various aspects of Database Application Development and design. Overall it was very fulfilling experience.

I would like to express my gratitude towards Prof. N.L Sarda and all the TAs for providing nice ideas to work upon. Not only did they guide us about the project but also always encouraged us to discuss ideas with them and corrected our mistakes. Without those discussion it would have been impossible to complete the work in such a manner.

I would also like thank all the TAs for making it a valuable experience.

Authors

Preface

The report documents the details about the Gotweet application we have made as a part of the the Database application project. The report first describes the backend that is supporting the application along with the description of the database. Then we have described the User Interface development. We have described the working of the application along with the main features and screenshots of the same.

The various aspects of the project including main features of the database and various features to maintain database consistency have been explained. Our focus in this project was on both database as well as UI part. So we have tried to make the interface as easy to use as possible. We have tried to make the project as good possible. We hope to succeed in our attempt.

Authors

Introduction

Our world has changed a lot in the last two decades. Technology has impacted almost every sphere of life. With all these advancements everything has changed a lot or is undergoing massive changes. Databases have been at the forefront of these changes. Advances in databases have affected almost every sphere of life. Databases are used in almost every online website/application which manages large data. These includes applications like Banking, E-commerce, Sensex, social networking etc. All these are possible only because of the database development. They provide the various features without harming the quality of the data like concurrent transaction, 24*7 connectivity consistency etc. Behind the back database makes it easier for programmer to maintain the data. Thus database provides benefits for all. Today amongst the highest produces of data are the social networking websites. They produce enormous amounts of data which have to be maintained and presented to the user in an interactive manner.

Social networking has become an integral part of life in the recent years. Various social networking application are hugely popular these days like facebook, twitter, quora, linkedIn etc. They have made it possible for people in various parts of the world to interact in many possible ways. Every app has a different approach towards this interaction. Twitter is one such application. Twitter allows people to make statements which can then be viewed by others. Today almost all eminent personalities are on twitter. People get to know their views via tweets. Also poeple can comment, like or retweet them. Twitter allows information to be passed onto a large number of people in short time.

It is important to have an understanding of how these applications works and to know the backend infrastructure supporting them. Of course just doing it on small scale is not enough but it gives us a very good perspective of the functioning. So we have made a prototype of the twitter in the form of Gotweet. We have tried to implement as many features as possible along with a simplistic UI to give user a twitter like experience.

Backend Database

Database Representation

The database consists of 7 main tables.These tables are Junta, Followers, Tweets, Comments, Messages, Likes and Retweets. A brief description of each of the table is given below:

Junta: This table stores information about the users who are registered with the application. It stores information like name, user_id, password, date of birth, number of followers, number following, email-id etc.

Followers: This table stores a row, for each 'follow' relation, which contains the follower and person being followed. This is similar to the following relationship in twitter. This table can be used both ways, i.e. to find the followers of a person or to find the people the person is following.

Tweets: This table stores the actual tweets. Since the final motive of the app is to create social networking site like twitter, for each tweet we store some information like who tweeted it(user_id of the person), tweet time, number of retweets, number of likes etc.

Comments: We have also provided the user with the facility of commenting on tweets. Comments are stored separately from tweets with foreign key reference to the tweet id. Comments also have commenting time, comment_id and the userid of the person who wrote the comment.

Messages: Along with public tweets the user also gets the facility to send private messages to other users. These messages are stored in the table 'messages'. The attributes include the sender, receiver, message_time, etc.

Likes: People might wish to know who has liked their tweet and also to prevent one person from liking a tweet twice we explictly store the 'like' relation. This relation contains who has liked which tweet.

Retweet: Since retweet signifies that the tweet is not original from the person thus we have stored the retweets separately. Also this prevents a person from retweeting twice. The attributes of retweet are tweet_id(references to main tweet table) and person id who retweeted it.

The main entities that are present in the database are Junta, Tweet, Message and Comment. The ER diagram of the relation is given below:

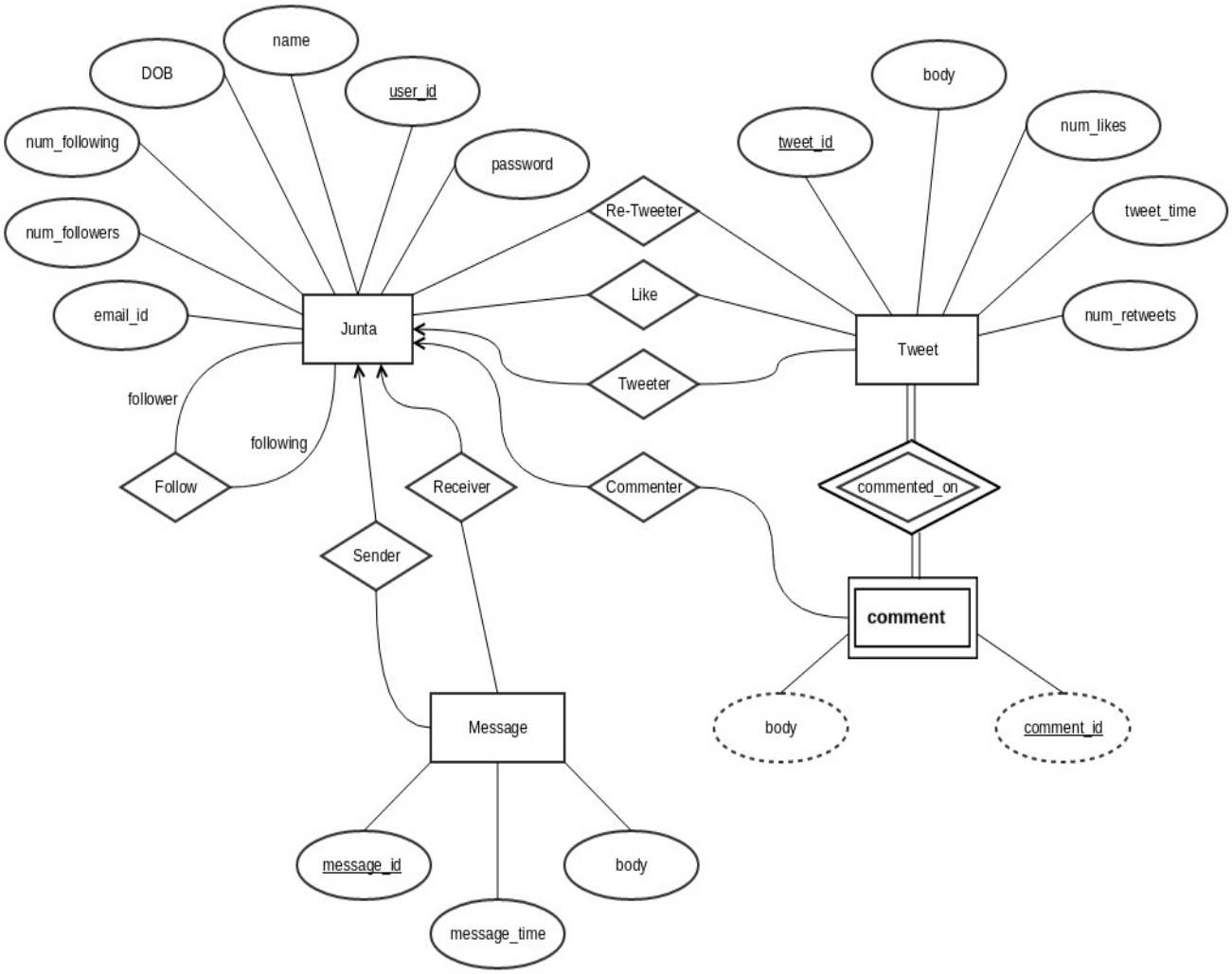

Figure 1: ER Diagram of the Database

Physical database design

Database Consistency

For a social networking application, database consistency is very important. There are various consistency requirements which have to be met. These include things like the username of commenter,liker,retweeter should be present in the junta table; the number of retweets, likes, comments should accurately represent actual number of comments. We have used various triggers for checking consistency, and all updates are done through functions. Thus the database is consistent all the time.

Tackling huge database problem

Problem This is huge data centric application. The number of tweets, comments, likers could be very large. under these conditions it could be very difficult to retrive the tweets of a particular user from the table of all tweets. A user can also be following many users and to get tweets of each and every user from the tweets table would involve huge delays.
Solution This can be optimized easily. Since at one time we retrive tweets of a particular user so we store the tweets of one user on contagious location. We can also have b+ tree on users which can give the location on the tweets of user in the table. Thus the process of getting tweets is easier. The same could be applied for getting comments of a particular tweet and for other purposes. Also when we need tweets of more than one user we get tweets separately for each and then mix them.

The database schema is presented below:

Implementation Details

Classes

We have classes in java corresponding to many table in the database including junta, followers, tweets, comments, messages, retweets etc. They have similar names. Thus it makes java functionality more easier. We can return tweets, person, comment etc. directly from any java function call.

Functions

We have used the following main functions for retriving and updating data.

AddUser: Takes the user details and adds him to the database. The user can now login using his credentials.

Gettweets: Takes as input the username of the person. Returns all the tweets the user is involved in. These may be either originally generated by the user, liked or retweeted.

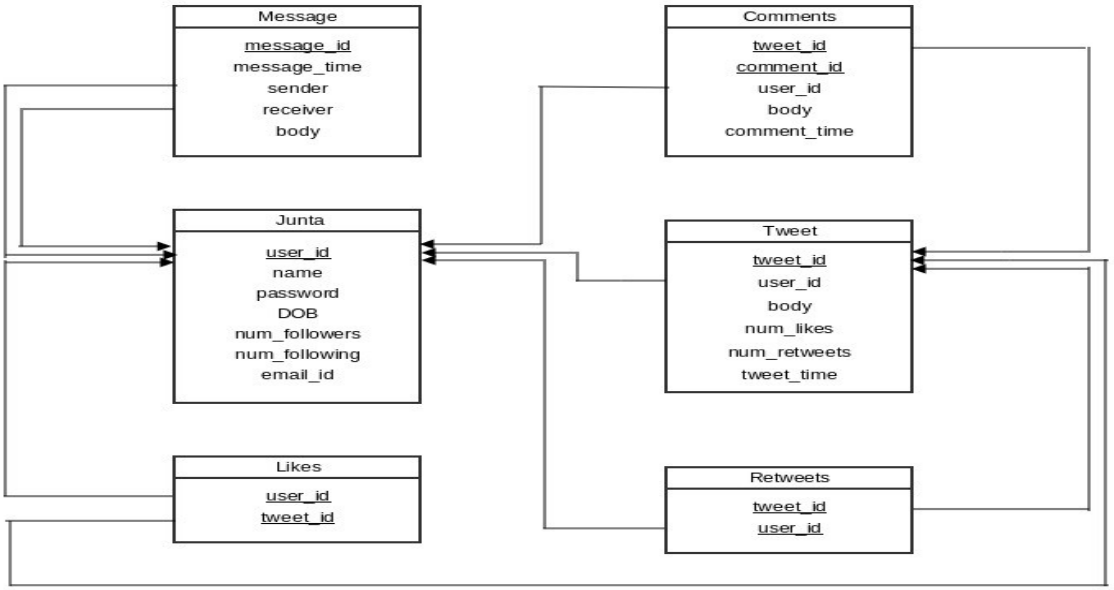

Figure 2: Schema of the Database

Getmessages: Takes input username. Returns a list of object message either sent or received by the user.

Sendmessage: Takes the username of the sender and receiver and sends the message i.e. the message gets added to the database and the receiver can now see the message in his inbox.

Getcomments: This function returns the comments for a particular tweet. This takes as input tweet-id.

Searchpeople: It takes string as input and searches the junta(table of users) and returns the list of users with similar username. We have used special function to define this matching.

Addtweet: Takes two entities as input, namely username who is posting the tweet and the message or the body of the tweet he/she is posting. It returns nothing since it will only add the pair to the database.

Retweet/like: Void functions. The input values tweet_id and username correspond to the user who retweeted/liked the tweet and makes appropriate changes to the database.

UpdateUser: Takes the attributes to be changed and username and makes corresponding changes to the database.

AddComment: Takes user_id and tweet_id as input and adds the comments to the tweet. The comment will now be visible below the tweet.

GetFollowers/GetFollowing: Takes the user_id of the person and searches the database for the people who are following him/whom he is following and returns the list.

Servlets

The servlets provide the main user experience. They provide many sorts of data checks each specific to a page. We have many servlets and the perform all the checks on the data and calls the appropriate java functions for performing the changes as per the user instructions. Then they lend the people on the appropriate page as par the task done.

Login and Signup: These servlets implements the functionality for login and signup. When a user tries to signup it checks all the details are proper and the username has not been taken and adds the user to the list. The user than lands back on the homepage from where he can login. When user tries to login it checks whether the user is registered and if everything is correct the user lands on his homepage.

Tweet/Commenting: Has the functions which enables the user to tweet and comment on a particular tweet. Takes two entities as input, namely username who is posting and the body he/she is posting. When called lands on the respective pages, where the change is visible. In case of comment it also takes the tweet_id as input.

Liking/Retweeting: It takes the username and tweetid as input and performs the required task. This lands the user on the same page where he was before but now the changes are visible.

Messaging: Implements the messaging functionality. Invokes the function getmessages when called. Takes input username. Returns a list of object message either sent or received by the user. After invoking, the user is able to see the message. Also performs the sendmessage functionality. When the user wishes to send a message, this is the servlet which respond to the request by calling the function Sendmessage.

Search: Calls the search function and displays the list of people returned. This servlet respond to the search people requests of the user.

User Interface

We have designed all the pages in html and css using the bootstrap. The servlets as described above provide movement and redirection between pages. We have tried to make the interface as simple as possible for new users. Some details as well as snapshots of the user interface are provided below:

- The user initially lands on the mainpage. There he can either choose to login or signup. Choosing the option lands him on another page where has has to fill his details accordingly and can signup/login. Only those users who have valid username and password can login.

Figure 3: Main Page of the app

- Once the user has logged in. He lands on his homepage. Here he can see his tweets and tweets from people he is following. From here he can either go his own profile, search people, see people who he is following/who are following him, check messages or see details of a tweet.

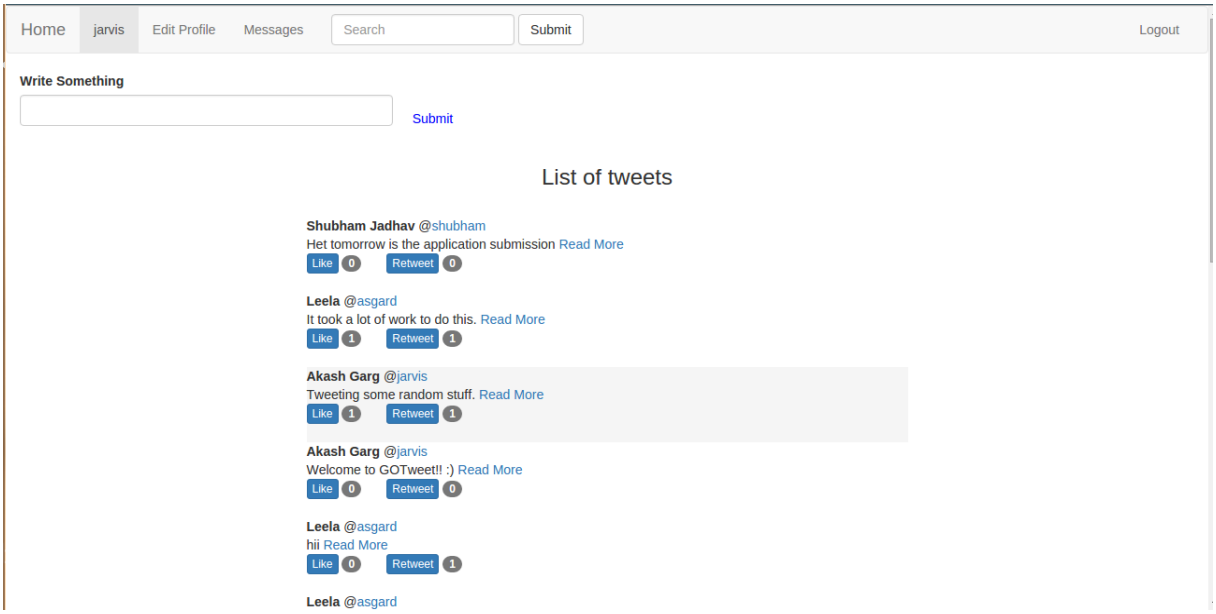

Figure 4: The homepage of the user

- When a user visits his profile he can various details about himself and also has option to edit his details. He also see his tweets which he has tweeted, liked, commented on or retweeted.

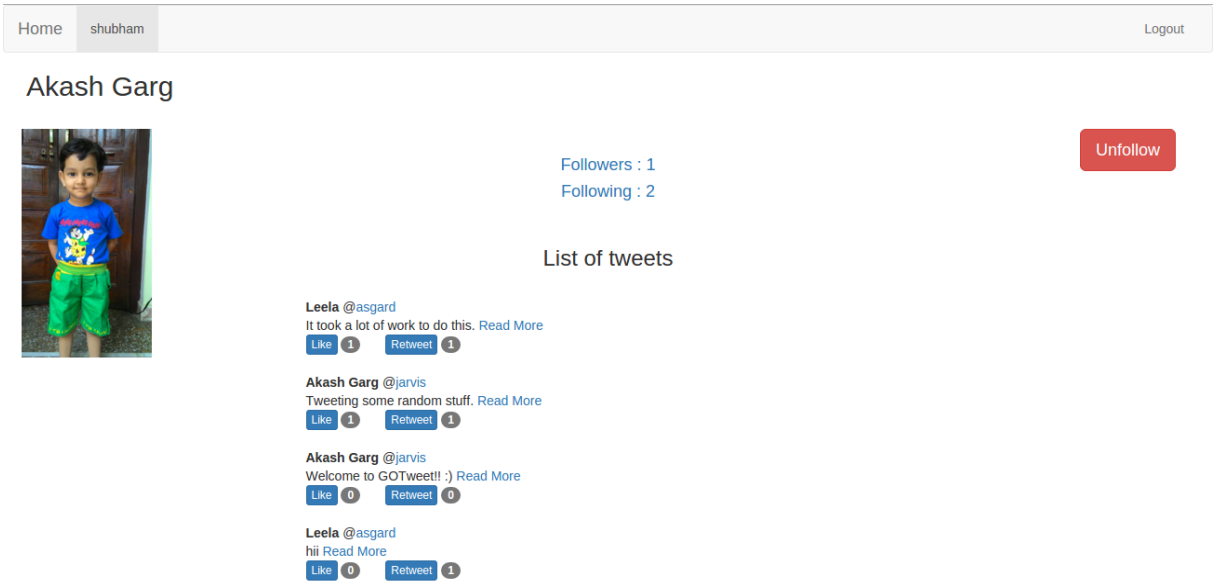

Figure 5: The profile page of the user

- When a user searches a person, he sees list of people in form of grid which have similar username. He can choose any user and visit his profile to see whether he is the one being searched. He also has the option to follow/unfollow the person.

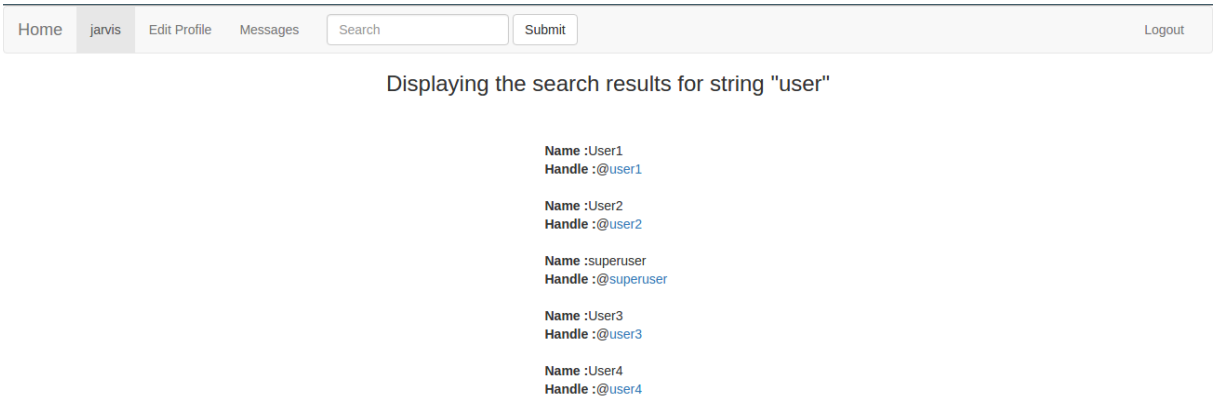

Figure 6: The results obtained after search

- When a user chooses to see the messages he sees a list of user he has communicated in the past. Clicking on any user gives the list of all the messages that have been exchanged amongst them. He can also send message to new user with whom he has not communicated yet just by providing his username and the message to be sent.

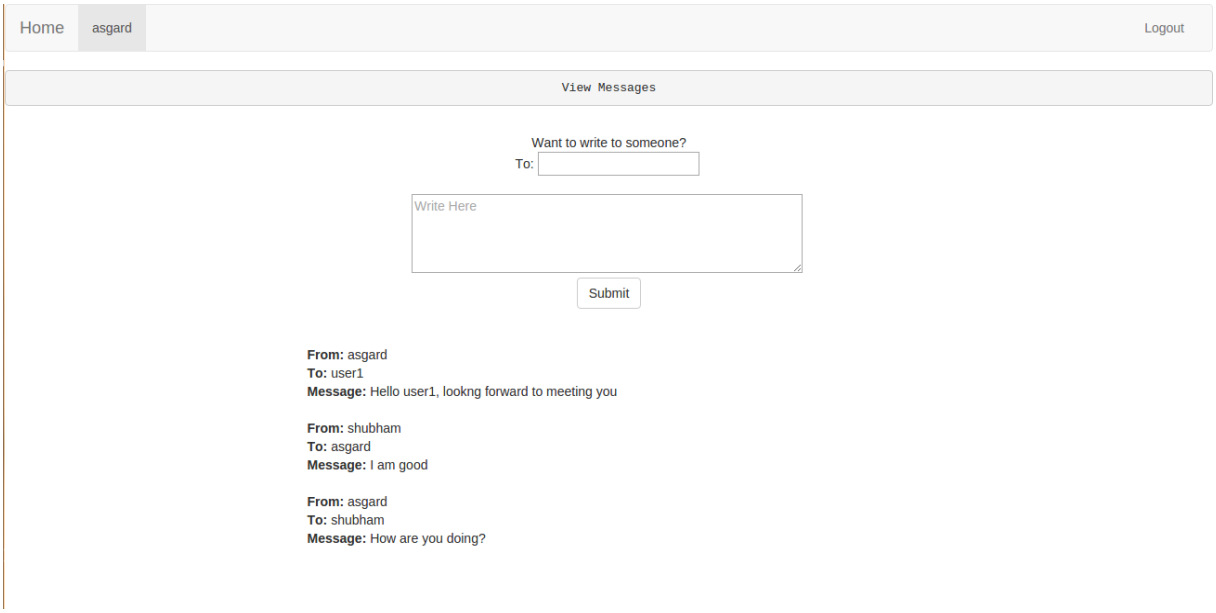

Figure 7: The messages for a user

- When a user wishes too see the people he is following or those who are following him, he can simply click the required option and the list is shown to him in similar manner as when he searches for people.

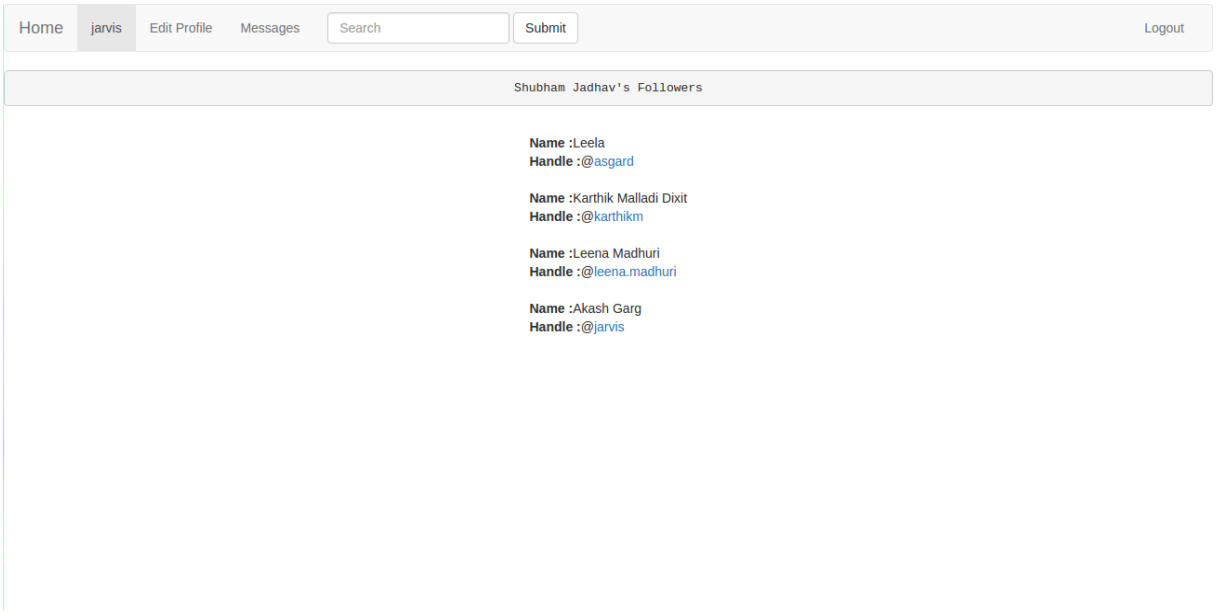

Figure 8: The followers of a user

- When a user click a particular tweet, he can see the details of the tweet. This include all the comments that have been made of the tweet so far. Also he can comment on that tweet.

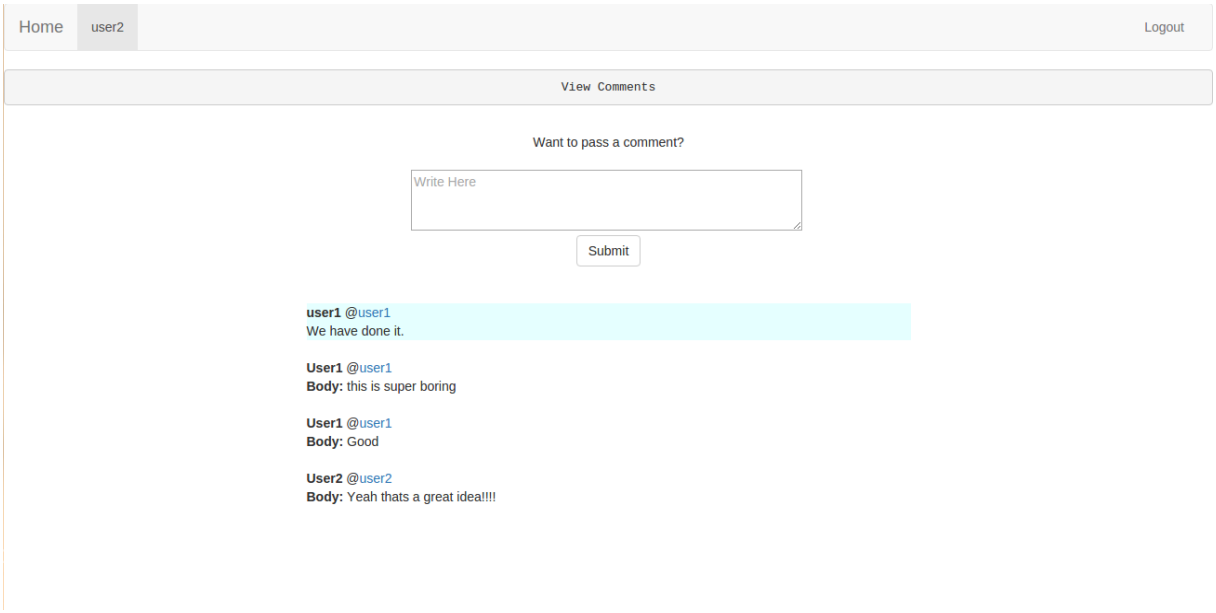

Figure 9: The details about a tweet

Changes in the code

We have made the following changes over the initial design. It includes mainly extra things we have added over the previous design.

- We have made little bit of change in the schema over the previous one. Names of some attributes have been changes which make it easier for join operations in the backend.
- We have added several new servlets for various new features like change password, update profile, two servlets for likes, retweets etc.
- We have also created user session by using POST method instead of GET method which provides certain level of security. So now its not just a bunch of HTML pages but rather an application which is quite secure. Also we have used proper input parsing which makes sql injection attacks difficult. Although apart from userid for rest of the things we have passed them via url only.
- We initially planned to allow user to add images but we could not find much information on how to use images with Postgres. So we had to drop the idea. Now we have hardcoded the profile pic.
- We have made some upgradation in the design of homepage and login/signup form to match the rest of the app. This now looks better than earlier.

Code sources available over Internet

We have learnt a lot using the open source tools available over the internet during the period of development. We have also used come already implemented portions of code instead of starting from scratch. Mainly we have them in designing while the backend has been coded from scratch. The main sources we have used are:

BootStrap: This is a very useful tool for developing application UI. It provides various HTML and CSS elements which can be used in the application. We have used CSS from it.

PostGres Book: We have used this book for referring various things about postgres. This provided a good reference material for it.

Other things were mainly used for bug removal like Stackoverflow.org . They were mainly for small reference and as such not explicitly cited.

Conclusion

As described above the project was a huge learning experience. We have gained a lot of insight into the database application development and their uses. While developing Gotweet we also learned how to provide a better GUI and how to make the application interesting for a user. What features may be desirable and which ones creates an overhead. Also we had to care about various security features so that personal data of a user is not revealed. Its clear that making a complete twitter was very difficult but we have implemented almost all functionality except that of profile pic. It is interesting to know the important role played by database in these applications. In the end we have tried our best to make the application as good as possible. We hope to succeed in our efforts.

References

We have referred to the following sources/used small portions of code from the following sources. A good part of the design we have learnt from the Twitter BootStrap which served very handy for a lot of things.

- https://www.stackoverflow.org/
- http://www.w3schools.com/bootstrap
- https://www.bootswatch.com
- https://www.w3schools.com/css
- https://docs.oracle.com/javase/tutorial/jdbc/basics/
- http://www.postgresql.org/docs/