VIRTUAL IDENTITY DATABASE

- Bitan Paul 160001016

- Mohit Malviya 160001035

Today’s world has the gift of social media for people to

connect with their friends and family seamlessly no matter

where they are. But this gift is a curse in disguise because of

all the different social media apps people use to connect to

each other. They are bound by the separation of social

media apps.

To solve this problem we present our database, Virtual

Identity Database which brings together all of the user’s

social media platforms making it much easier for the user to

view their online content and connect with all of the their

friends and families more efficiently.

We aim to aggregate people’s social media activities at one single platform so that they don’t loose their precious time to open different apps one by one and still stay updated. Also data about the user’s is available to view at one place making it very convenient.

Advantages of Virtual Identity Database :

* ***Reduced data inconsistency and redundancy***

The relationship sets are designed in such a way so as to minimize the redundancy of data by making all the domains atomic and decomposing all the entity sets in such a way that all the attributes are fully functionally dependent only on the unique candidate key of the set eliminating the partial dependencies in accordance with and therefore verifying 3NF. This implies that theis databse is optimized to occupy less storage space,cost and prevent redundancy and inconsistency.

* ***Integrity constraints***

Restrictions in the form of integrity constraints have been imposed on appropriate attributes of relationship sets and entity sets to ensure consistency of the database.

* ***Automation***

To make sure Virtual Identity Database is consistent, it is updated in accordance with the transactions that took place automatically by using stored procedures and triggers.

* ***Security***

As it is a social media based database,security is of utmost importance. In order to keep unauthorized access to one’s data and modifications in check,passwords and privacy modes are employed in the schema.

***Entity sets***

For all the strong identity sets, the primary key is the only candidate key. For the weak entity sets, discriminator and primary key of the owner entity set form the only candidate key of the weak entity sets.

1. **user**(userId,username,firstName, lastName,password,gender,dob,image,bio,contact,email,lastonline )
2. **interests**(userId,interests)
3. **eduQual**(userId, degree,field,institute)
4. **work**(userId, duration)
5. **socialMedia**(accId, site, handle, pass, firstName,lastName, type, bio, mode)
6. **posts**(postId, text, media, likes, comments,timestamp,location, shares)
7. **messages**(msgId, participantId,site, content, time, msgTag)
8. **Location (**userId,tag,city,state,country)

**Weak Entity Sets**

**1. Search(**userId,content,timestamp**)**

***Relationship sets***

* 1. **userAcc**(userId,acid )
  2. **accPost**(acid.postId,)
  3. **accMsg**(msgId,accId)
  4. **accBlock**(accId,blockId,timestamp)
  5. **accFol**(accId,folId,timestamp)

***Triggers***

1. after\_delete\_accMsg –

Triggers after deletion on accMsg

Deletes messages of the accId which was deleted from the accMsg

1. after\_delete\_accPost –

Triggers after deletion on accPost

Deletes posts of the accId which was deleted from the accPost

1. before\_delete\_messages-

Triggers before deletion on messages.

Deletes tuples from relationship set accMsg containing the same messageId that is to be deleted from table messages

1. before\_delete\_socialMedia-

Triggers before deletion on socialmedia.

Deletes tuples from relation acMsg containing accId of the accounts which are to deleted from the table socialMedia

Similarly,

Deletes tuples from relation accPost

Deletes tuples from relation accmsg

Deletes tuples from relation accBlock

Deletes tuples from relation accFol

1. before\_delete\_users-

Triggers before deletion on users.

Deletes tuples from socialMedia containing accId of the users who are to be deleted

Deletes tuples from table location.

Deletes tuples from table work.

Deletes tuples from table eduQual.

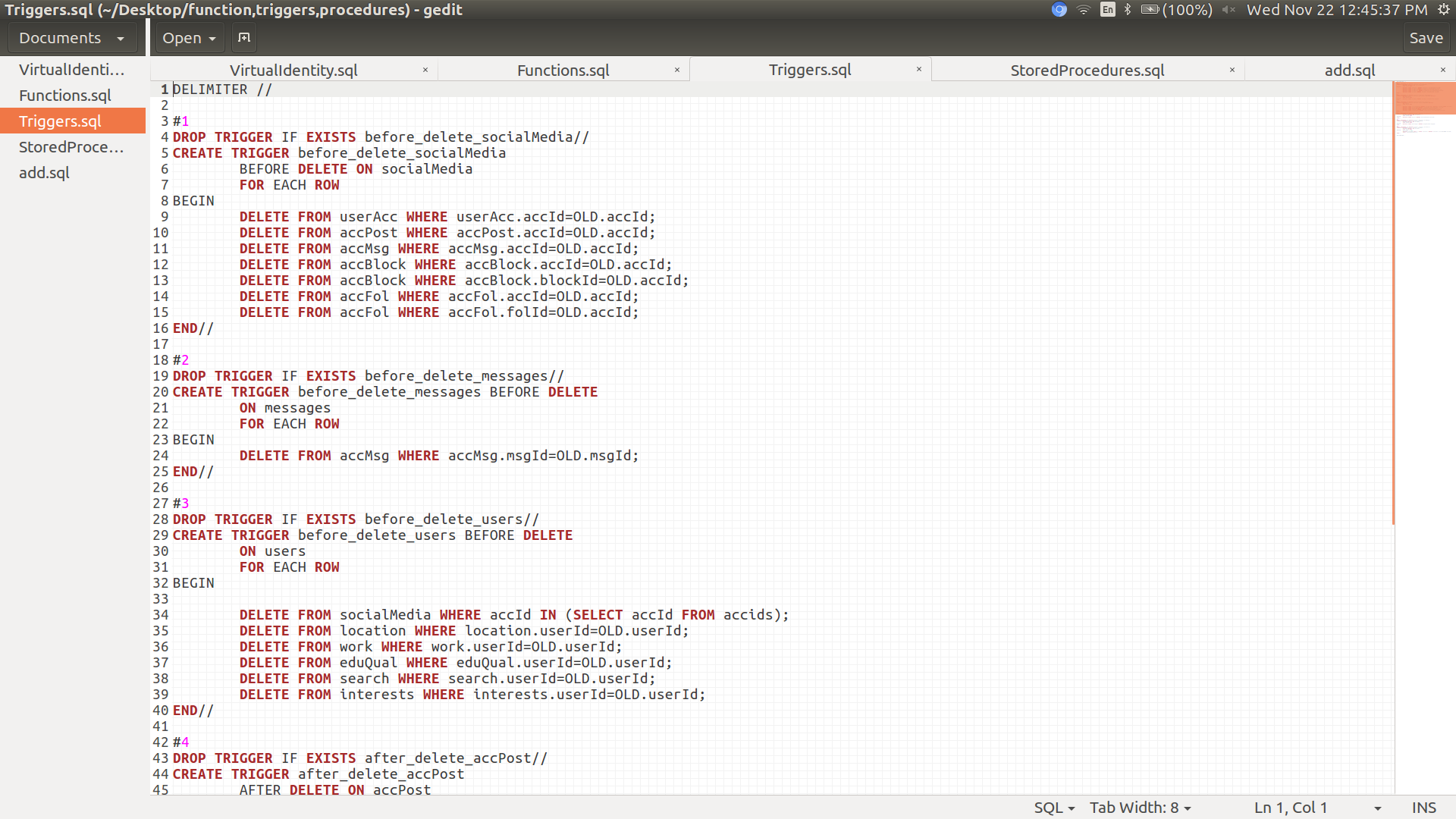
Deletes tuples from table search.

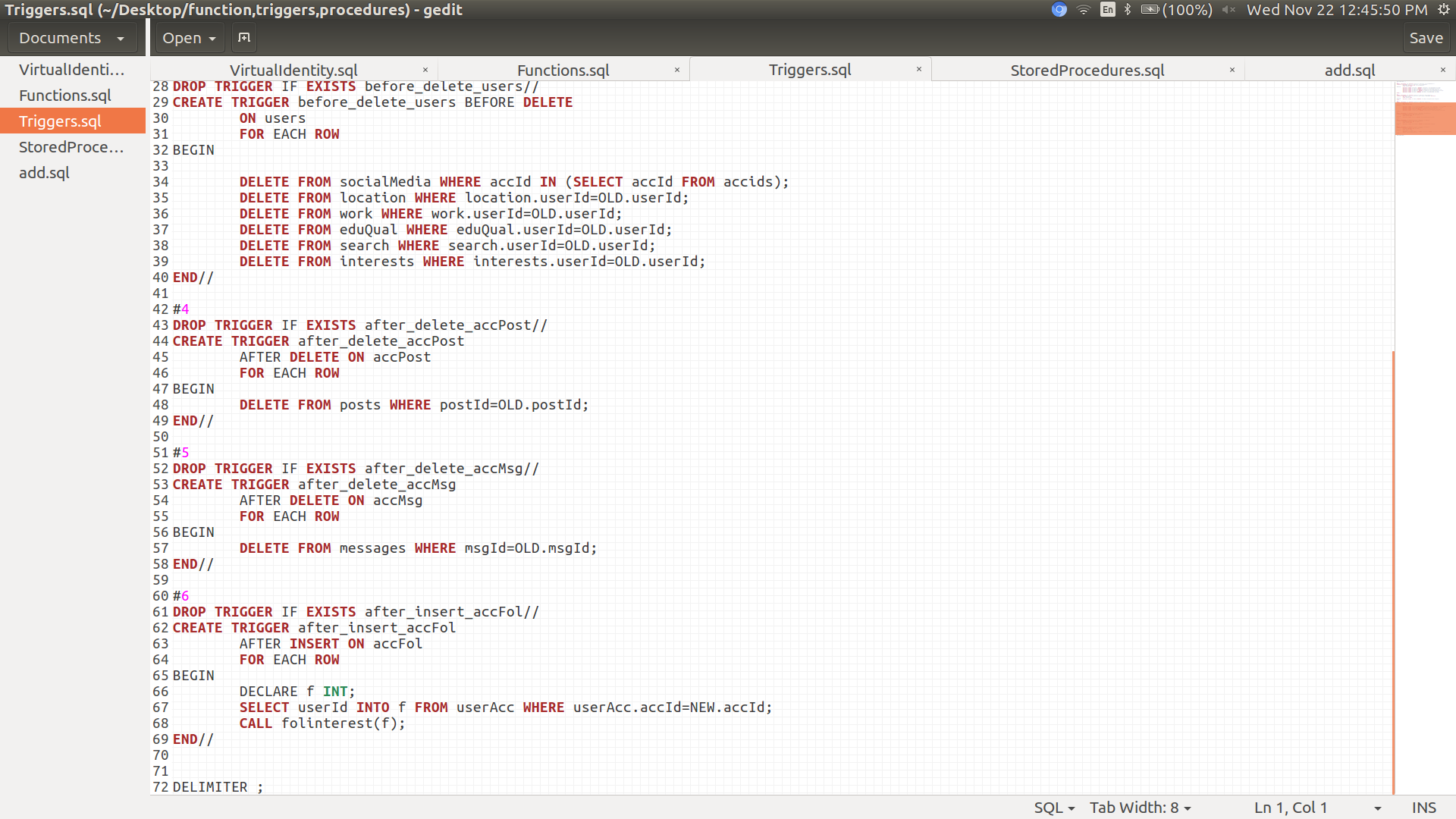
Deletes tuples from table interests.

1. After\_insert\_accFol

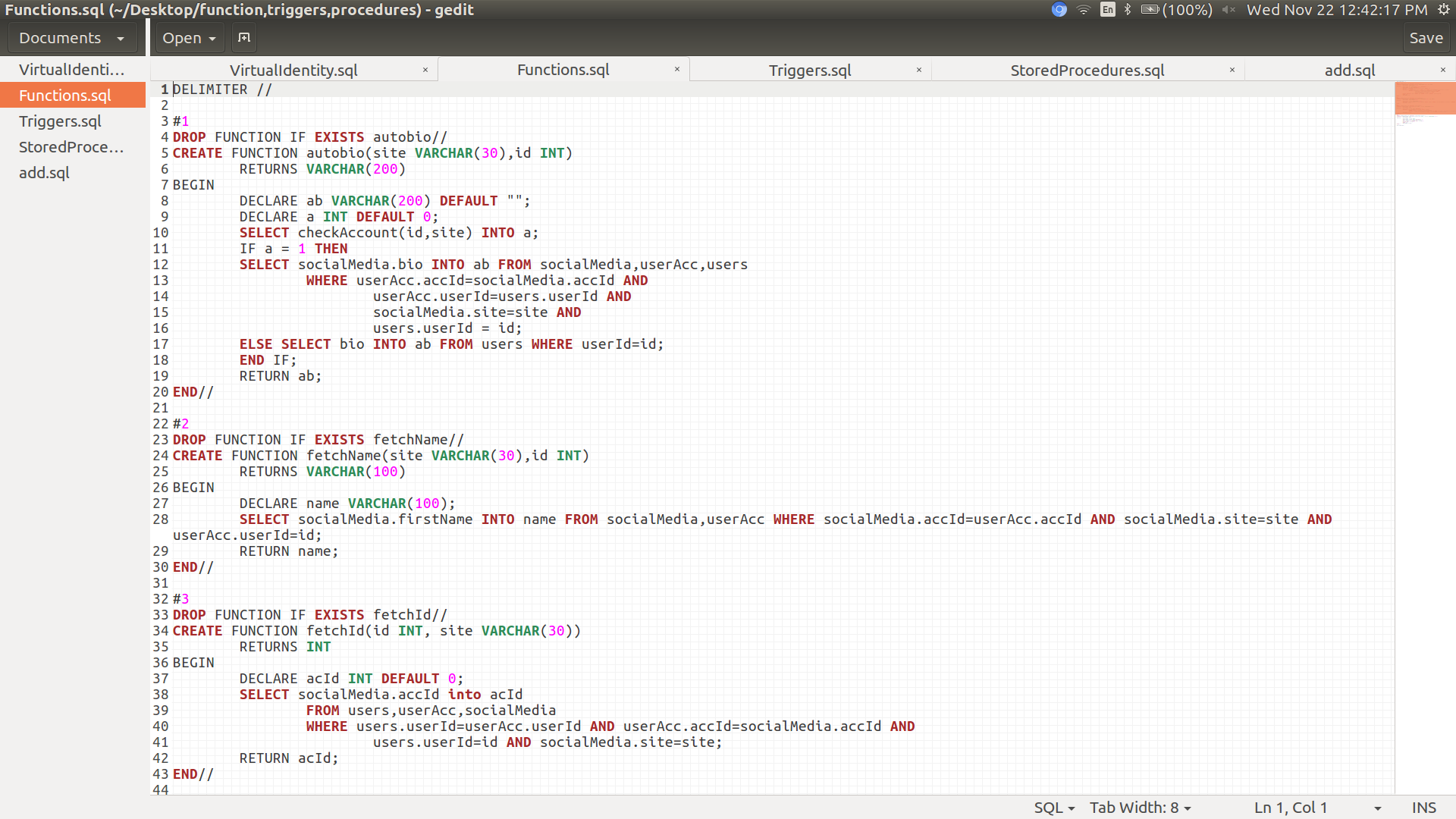
Triggers after insertion on relation accFol.

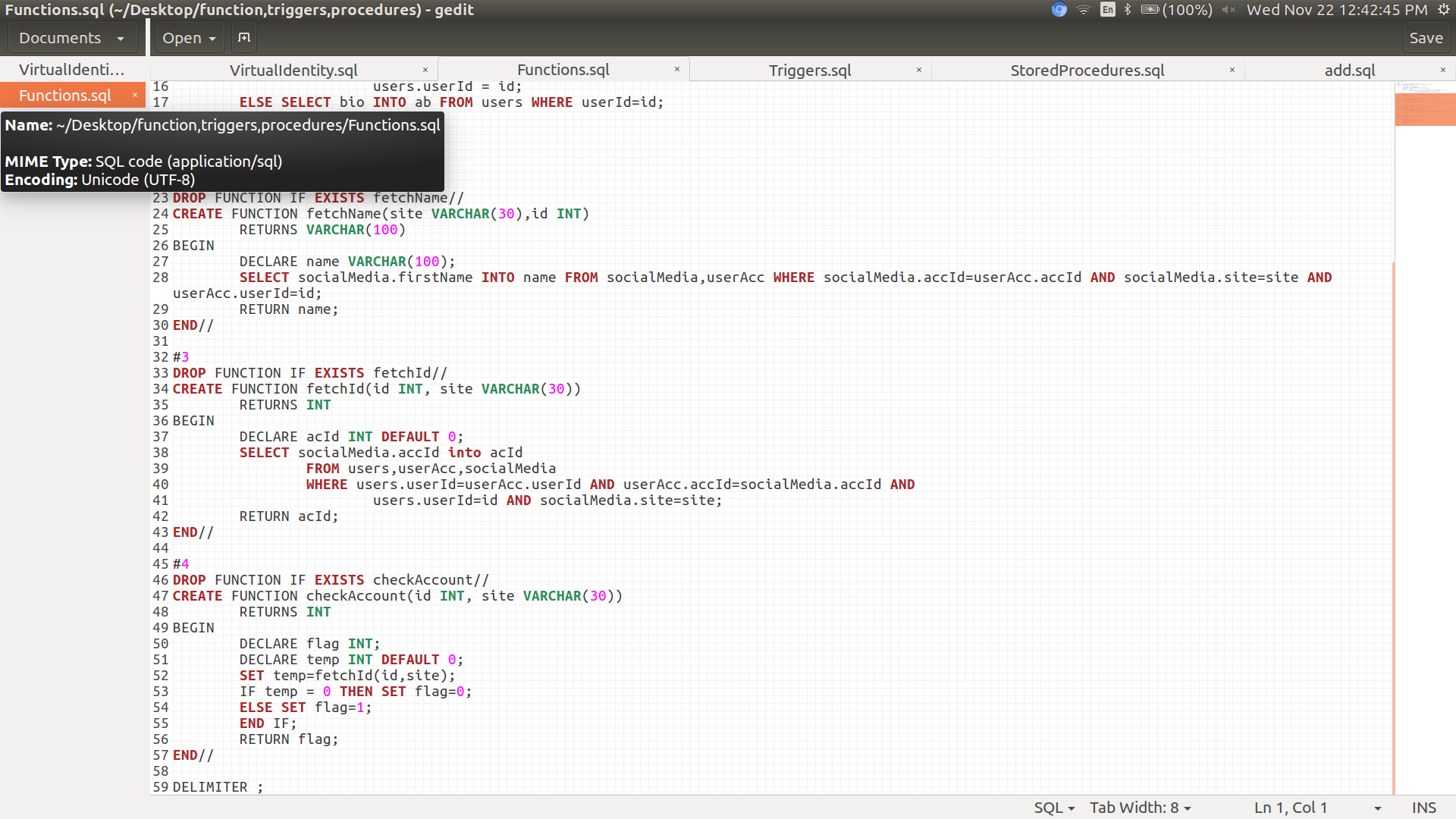
Calls procedure folinterest for the accounts which were inserted.

******

******

***Functions***

1. Autobio - takes site and userId as input and checks the user has account on that site then returns the bio from the table social media,if not then it returns the bio from users table
2. fetchName – takes input and userId and returns firstname from the table socialMedia
3. fetchId – takes userId and site as input and returns accId from the table socialMedia
4. checkAccount – takes userId and site as input and checks if there is any account stored from the given userId on the given site,if yes returns 1 else returns 0



***Stored Procedures:***

1. **addAccount** :

Creates a new account on socialMedia by taking input a userId,a username,password for the new account and the site for which the new account is being created.

Updates first name,last name on socialMedia from the users table.

Updates site on socialMedia from input.

Also creates tuple in relationhip set userAcc for the created account thus automating the process of filling relationship set.

2.**deleteAccount**:

Takes the involved userId and site as input.

Uses function fetcheId to retrieve the Id of the account of the given user on the given site.

Deletes the account having the retrieved Id hence automating the process of deleting accounts.

3.**deletePost:**

Takes postId as input.

Deletes tuples from relation accPost containing the postId specified in the input.

4.**deleteMsg**:

Takes messageId as input.

Deletes tuples from relation accMsg containing the messageId specified in the input.

1. **interest**

Inputs the name of the the user searched, where the account type is business into the interest table of the user’s account as tags.

1. **location**

Takes userId ,tag specifying if the address is current or not,city,state and country as input.

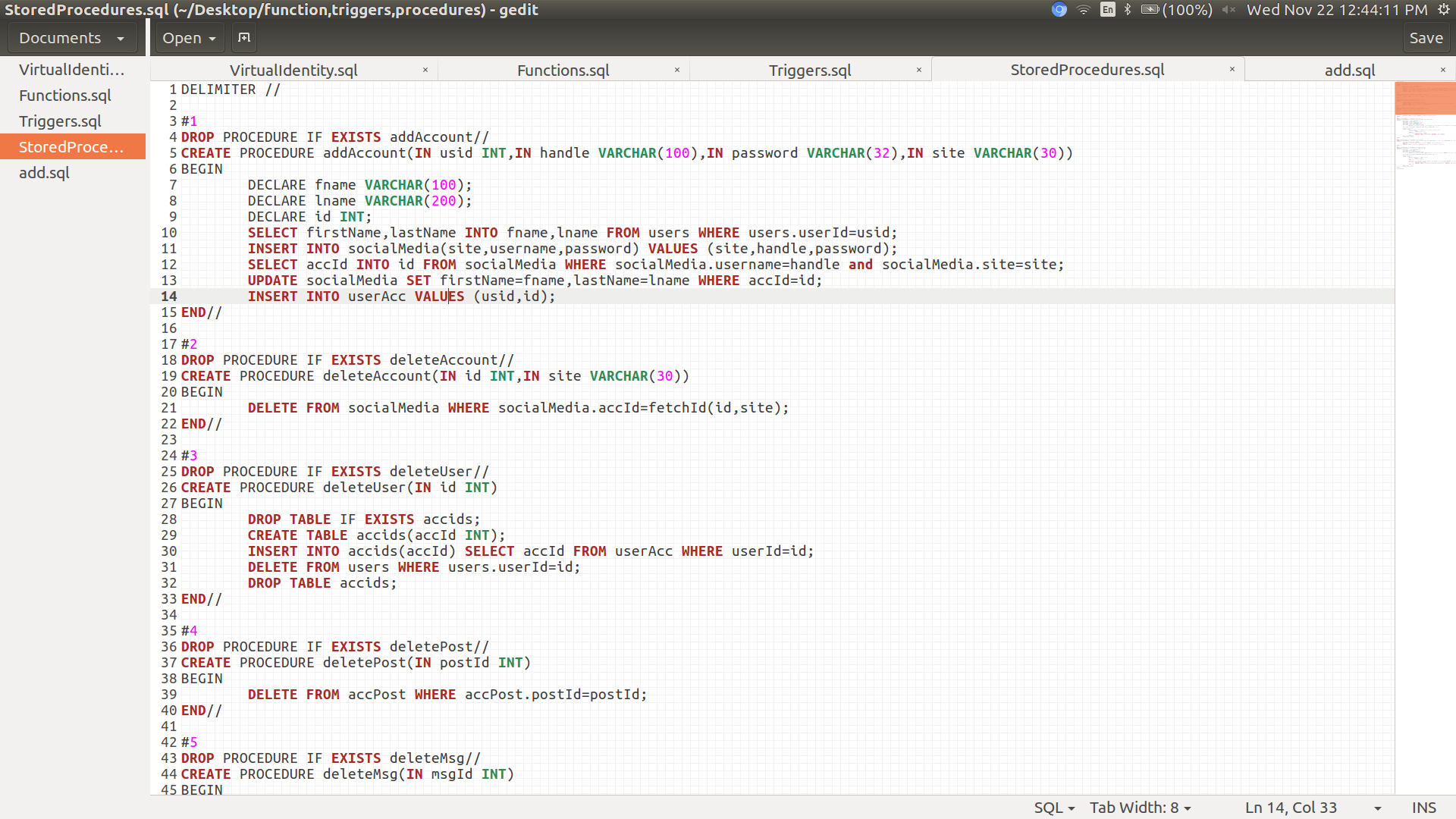
Checks if the input tag is “current” and if it is,updates the old current address to “lived”.

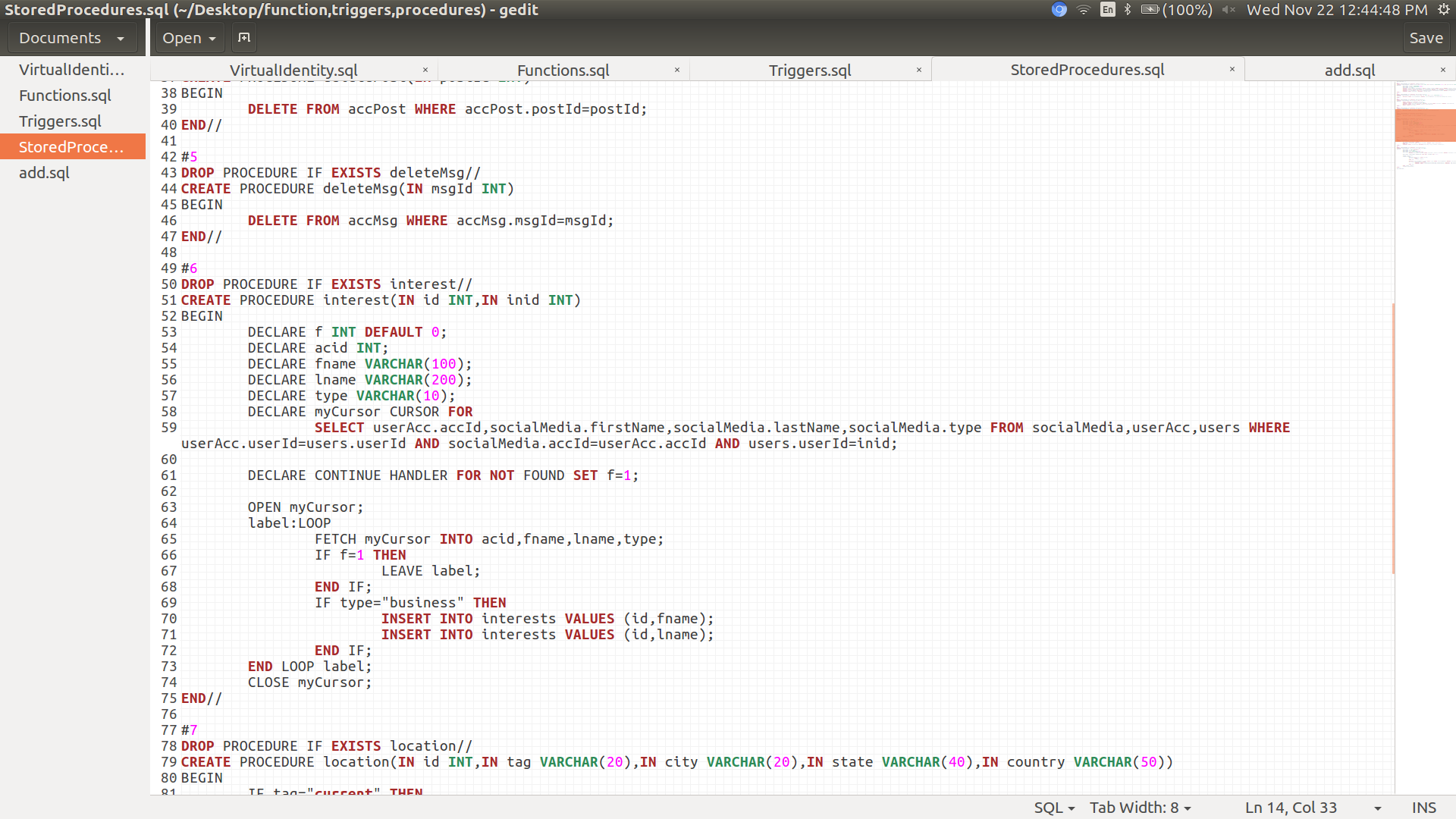
Inserts the new values in table location.

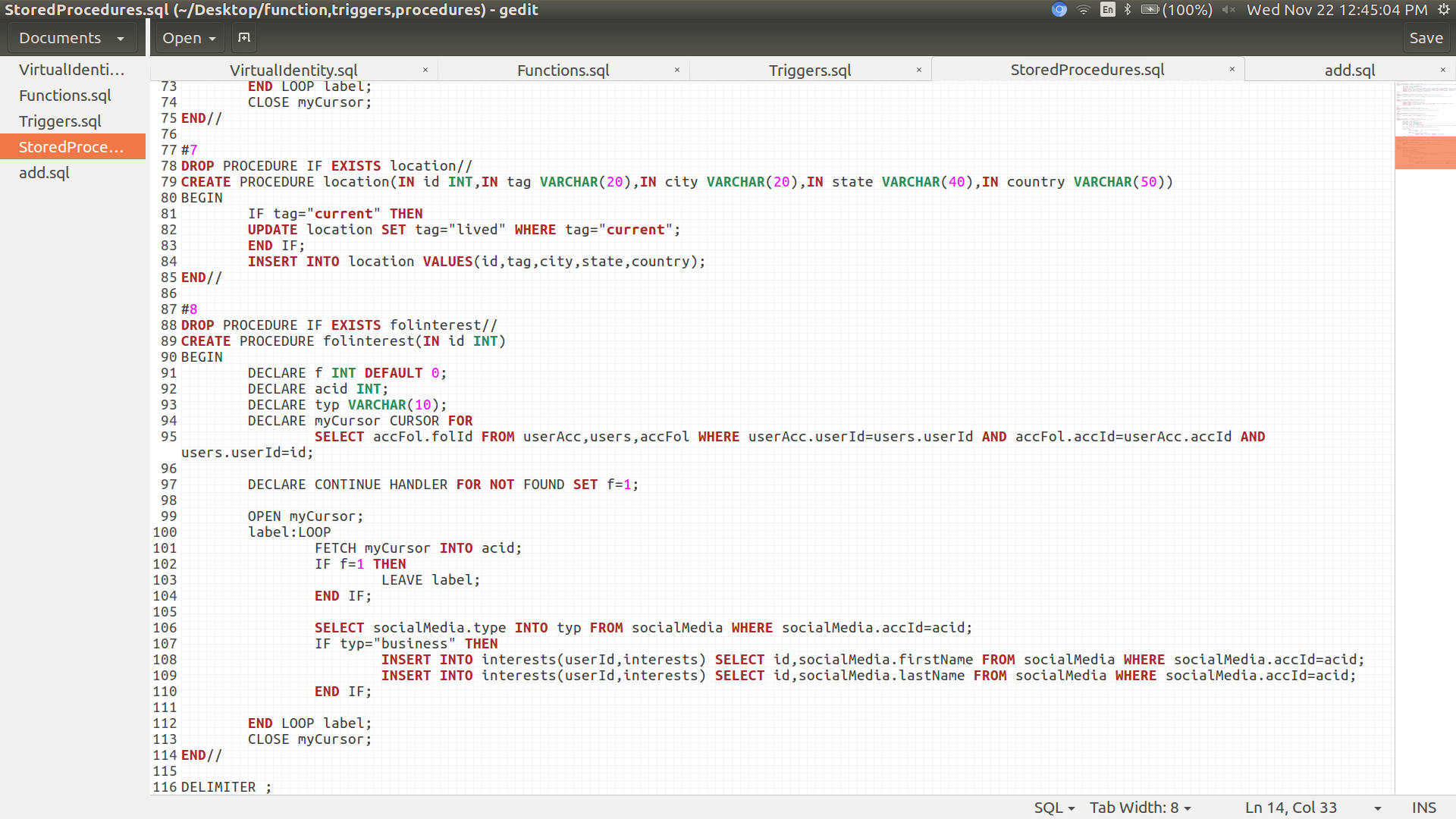
1. **Folinterest**

Takes userId as input.

Inserts into interests table the names of the accounts from the user’s followers where the account type is business.







***Constraints***

1. To check field validations in forms and password length.
2. To check if username is already present because it is to be unique.
3. User can message another user only if he/she is not blocked.
4. User can see another user’s profile only if the accouunt’s privacy mode is public.
5. User can view profile or send posts or send messages from a particular site only if the user has an account on that site.
6. At a given time only one location has the tag current.
7. Appropriate attributes such as userId,accId are constrained to be not null and unique.
8. Appropriate Foreign Key references are added to ensure no illegal entries.

For eg. userAcc references users and socialMedia so that only those userIds which are present in table users are allowed to be filled in the relation userAcc.

**Trasactions:**

1.**Insertion**: Records can be added in for a user from name to work and qualification.

2.**Viewing and Editing Profile**: Already existing details of a user can be edited but only by the user itself.

3.**Adding accounts on SocialMedia :** User can add his account on social media platforms from the dashboarditself.

4.**Searching for other Users on the database:**  User can search for other users in the database view their posts and messages.

5. **Posting Posts:** User can post from his/her socialMedia accounts

which are recorded in the database automativally.

6. **Sending** **Messages:**User can send messages to other users

their socialMedia accounts.

7.**Deleting Social Media Accounts :** User can delete his/her Social Media accounts while he/she is logged in.

9.**Deleting User Accounts:**  User can also delete the main user account on the database,doing so deletes all user’s social media accounts along with posts and messages atomaticallly.

**9. Viewing Messages and posts Sorted with time and also filtered with time intrval**

User can view another user’s messages and post sorted according to time they were sent.

Also user has the power to **view messages and posts which were exchanged in a chosen time interval**.

User can view posts and also number of likes,comments and shares on the posts.

**Functions and features of the Virtual Identity database:**

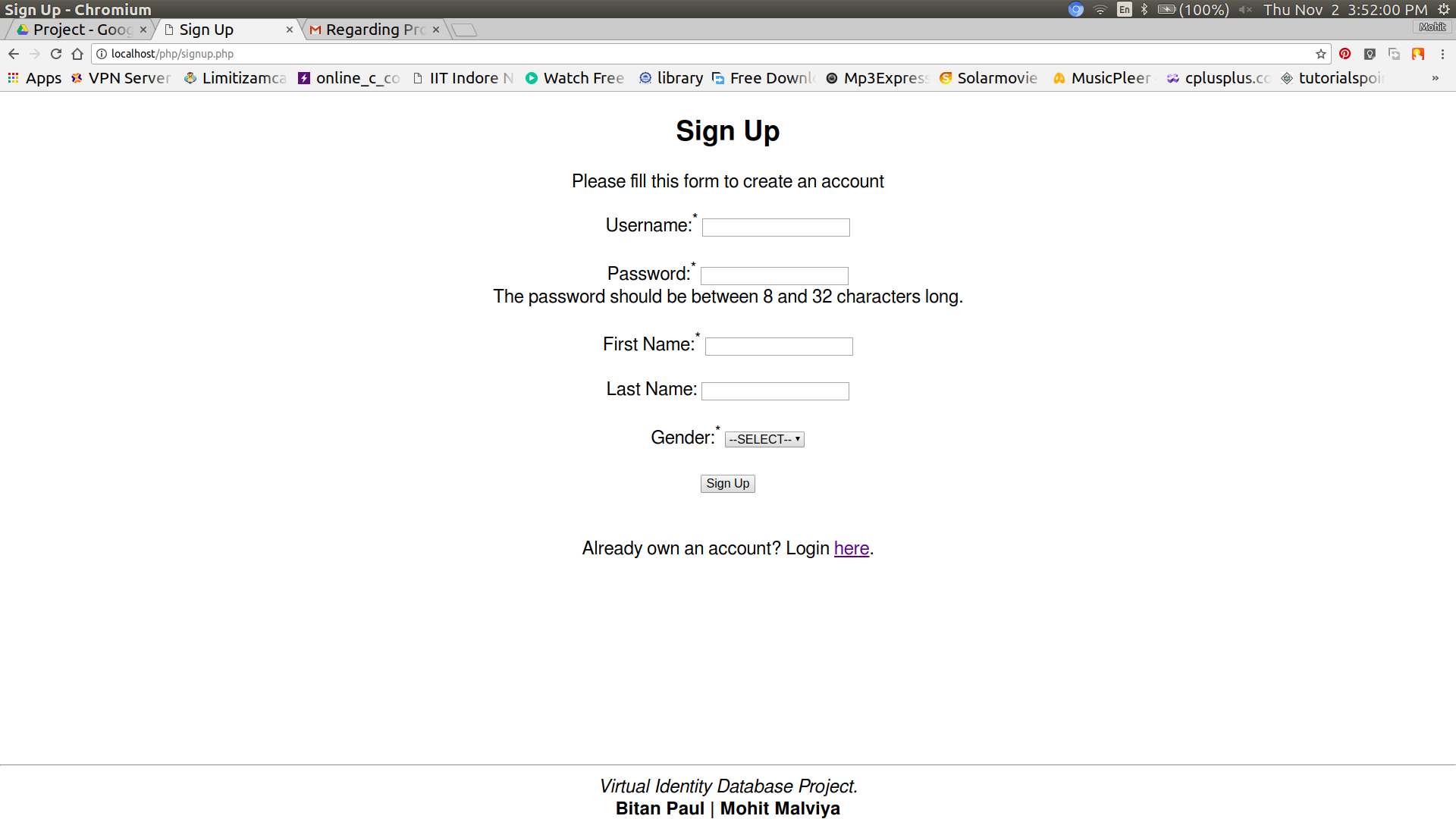
*User Registration/Signup:*

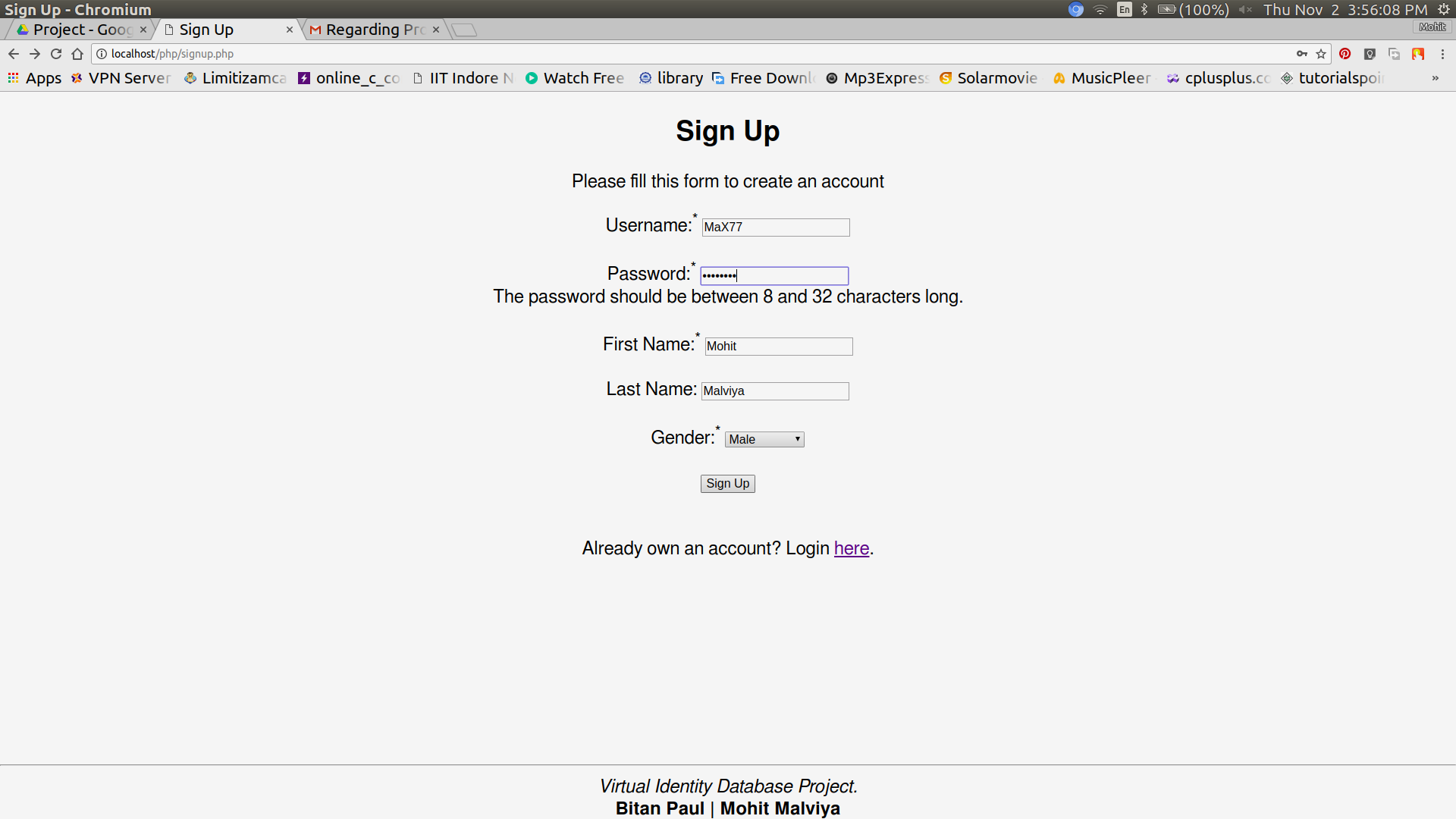
In order to use the Virtual Identity database one has to first register by creating an account to become an authorized user.

For this the customer will be prompted to a signup page.

To signup the user has to fill details specifying what username will he/she be using,actual name,gender,and choose a password for the sake of seccurity.

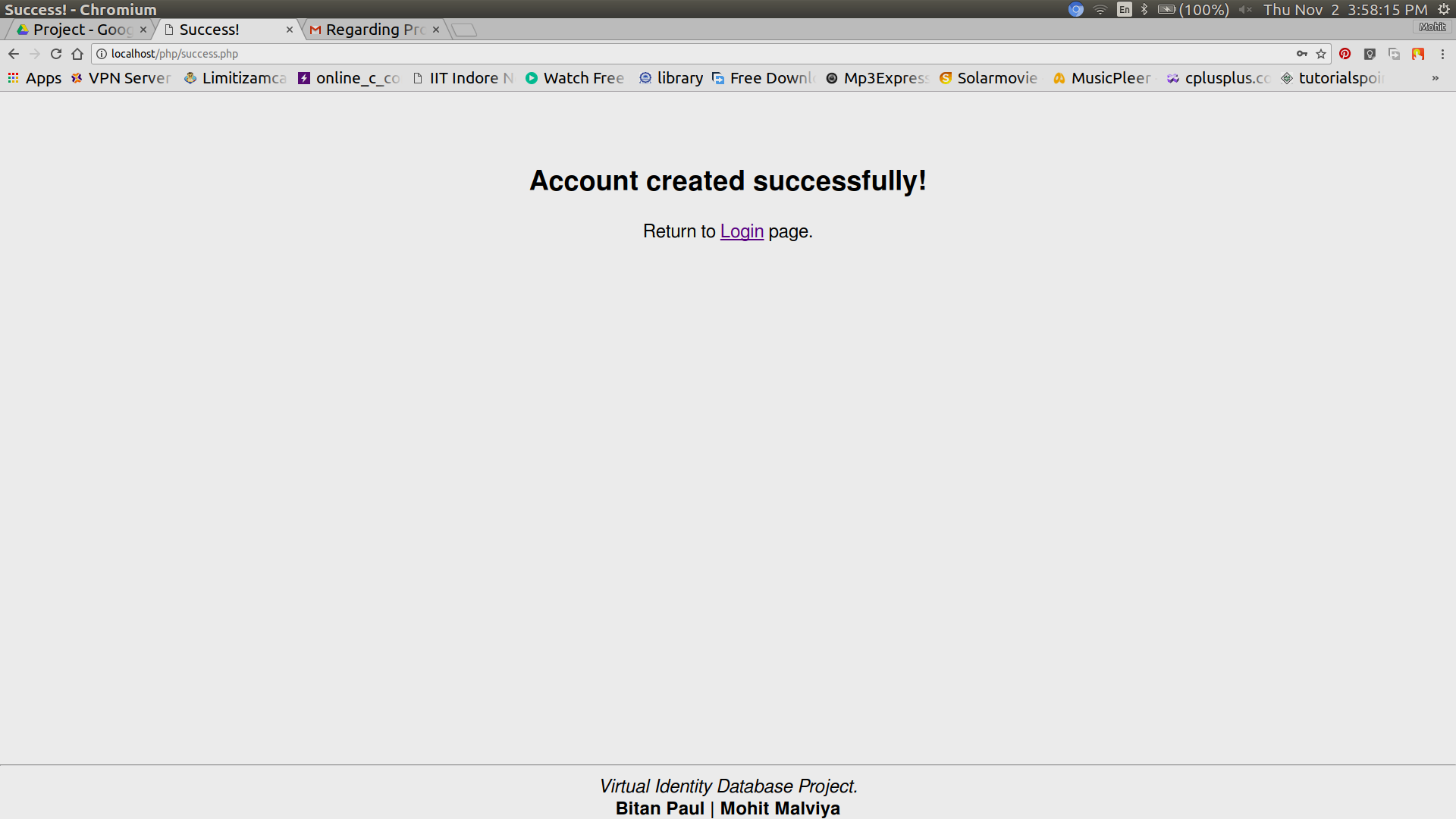
A UserID will be generated automatically and the details wil be inserted in sql database using php.





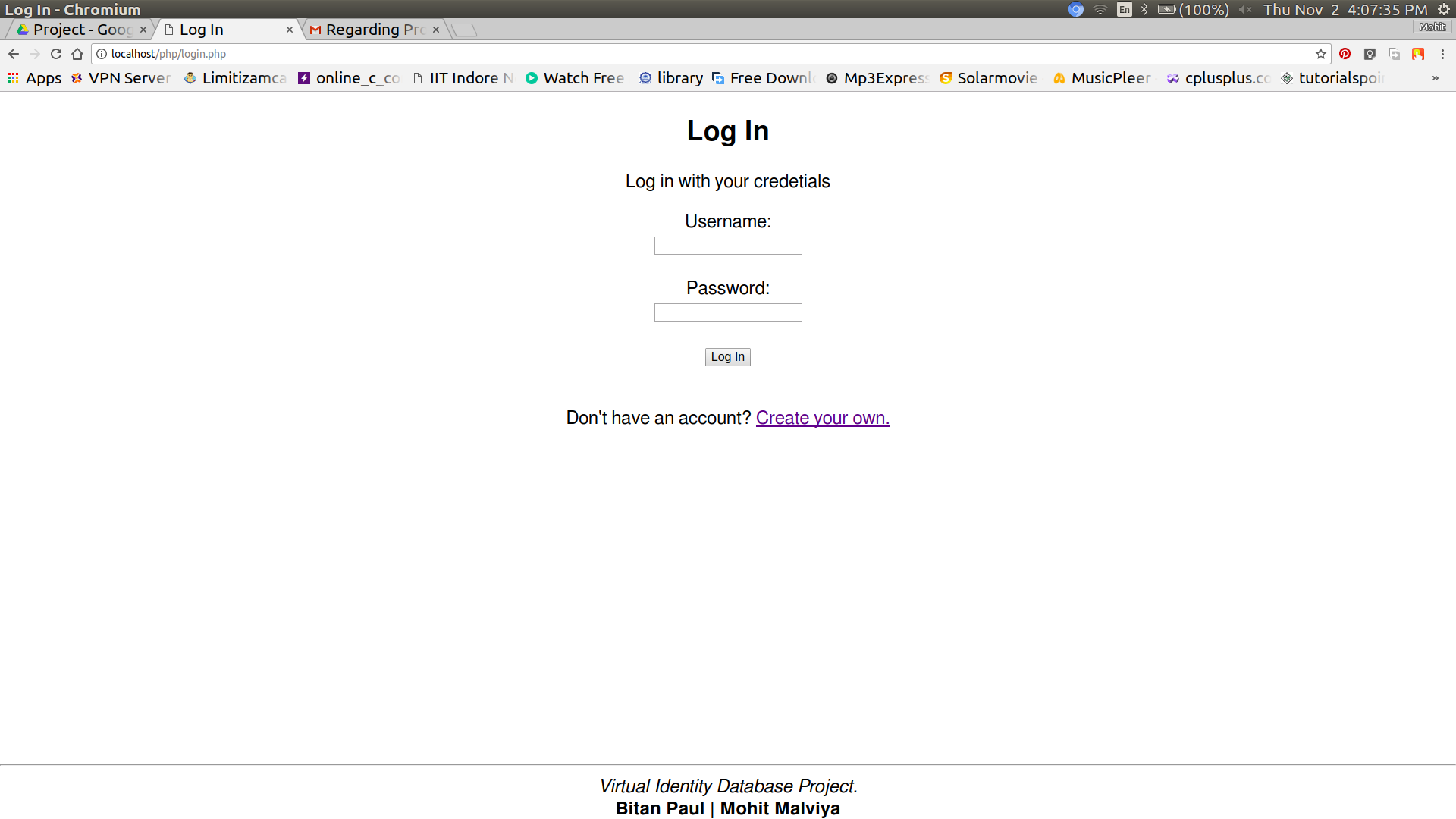
After filling up the details and clicking on signup person is directed to a successful signup page which declares that the person is now officially a “user”.

Also a link to the login page is given to redirect the user to the login page.



*User Login:*

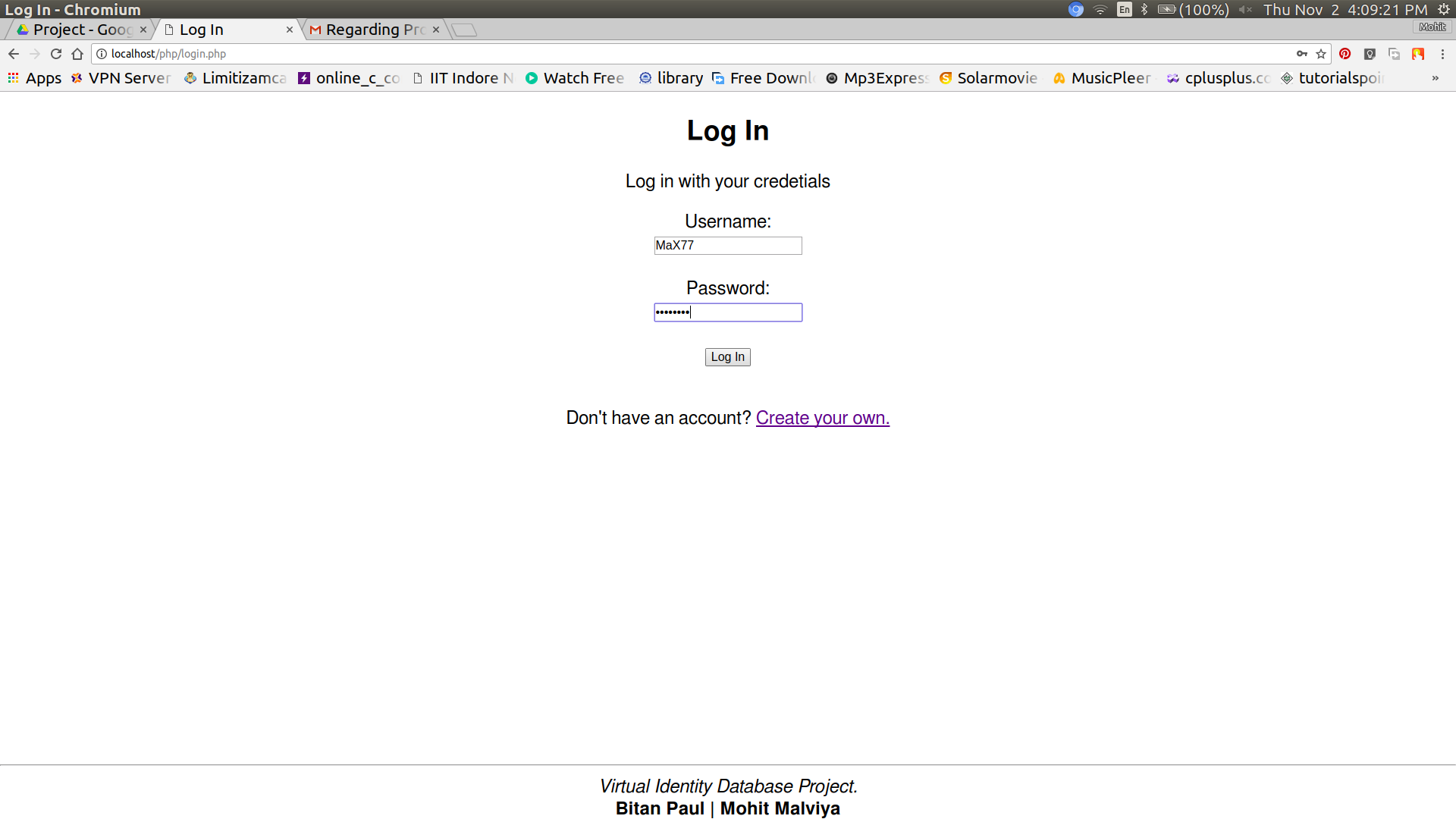
A user can login to his existing account using his username and his password from the login page.



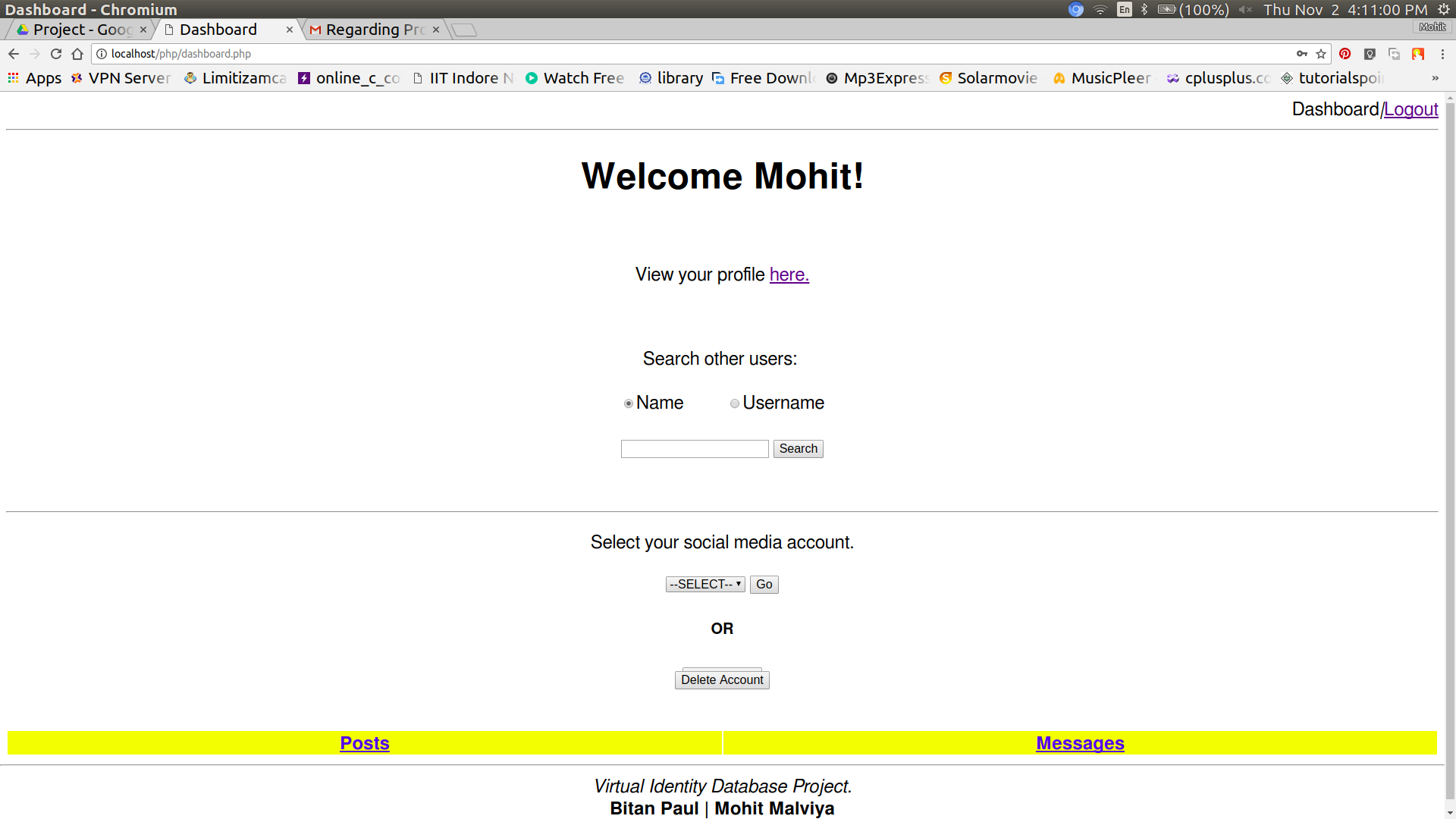
Also a link to the signup page is given if the user doesnot has a account.

User can log in after filling the correct credentials and clicking log

in button on the page.

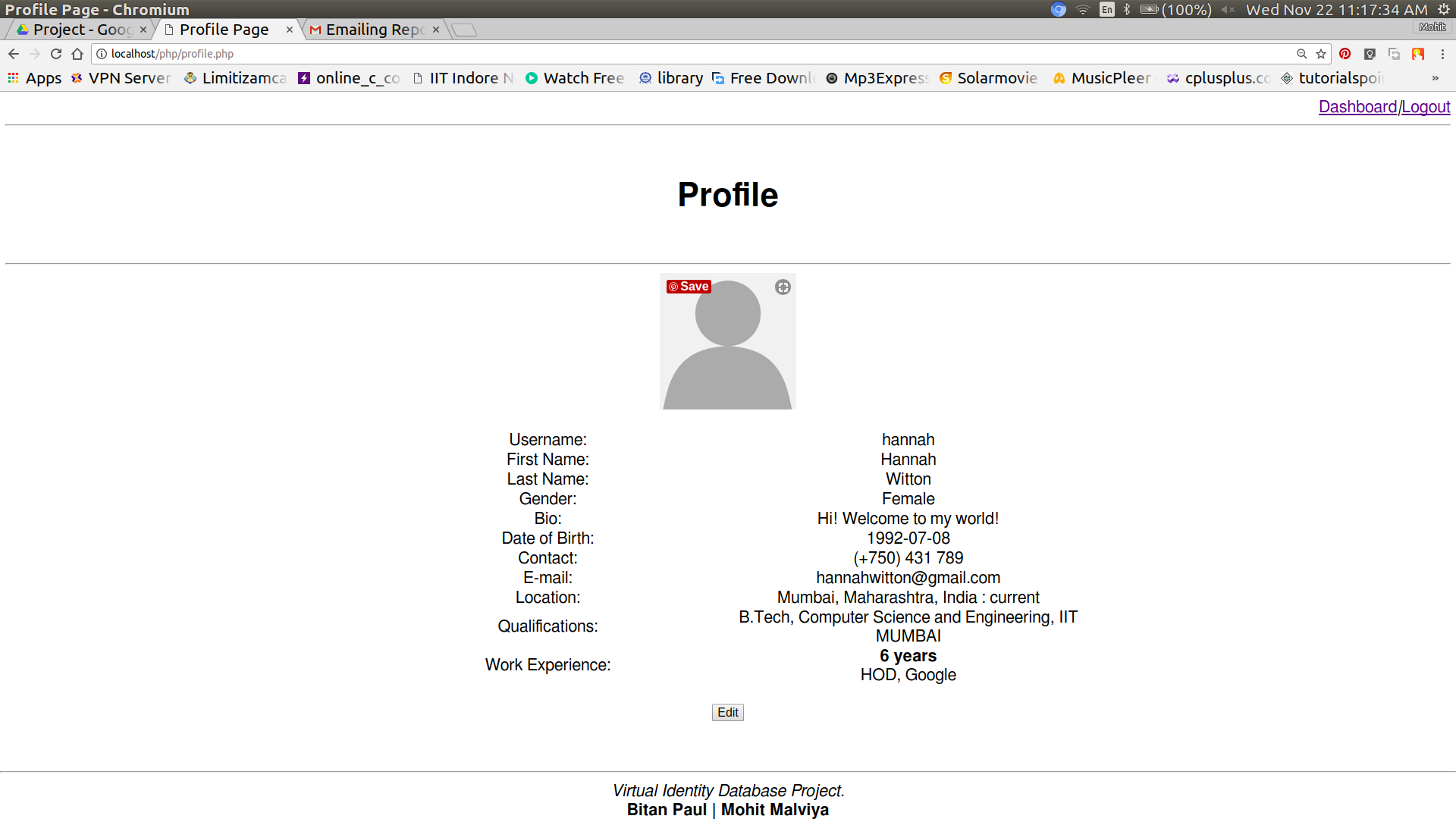


If the log in is successful the user is directed to the dashboard where more options await him.

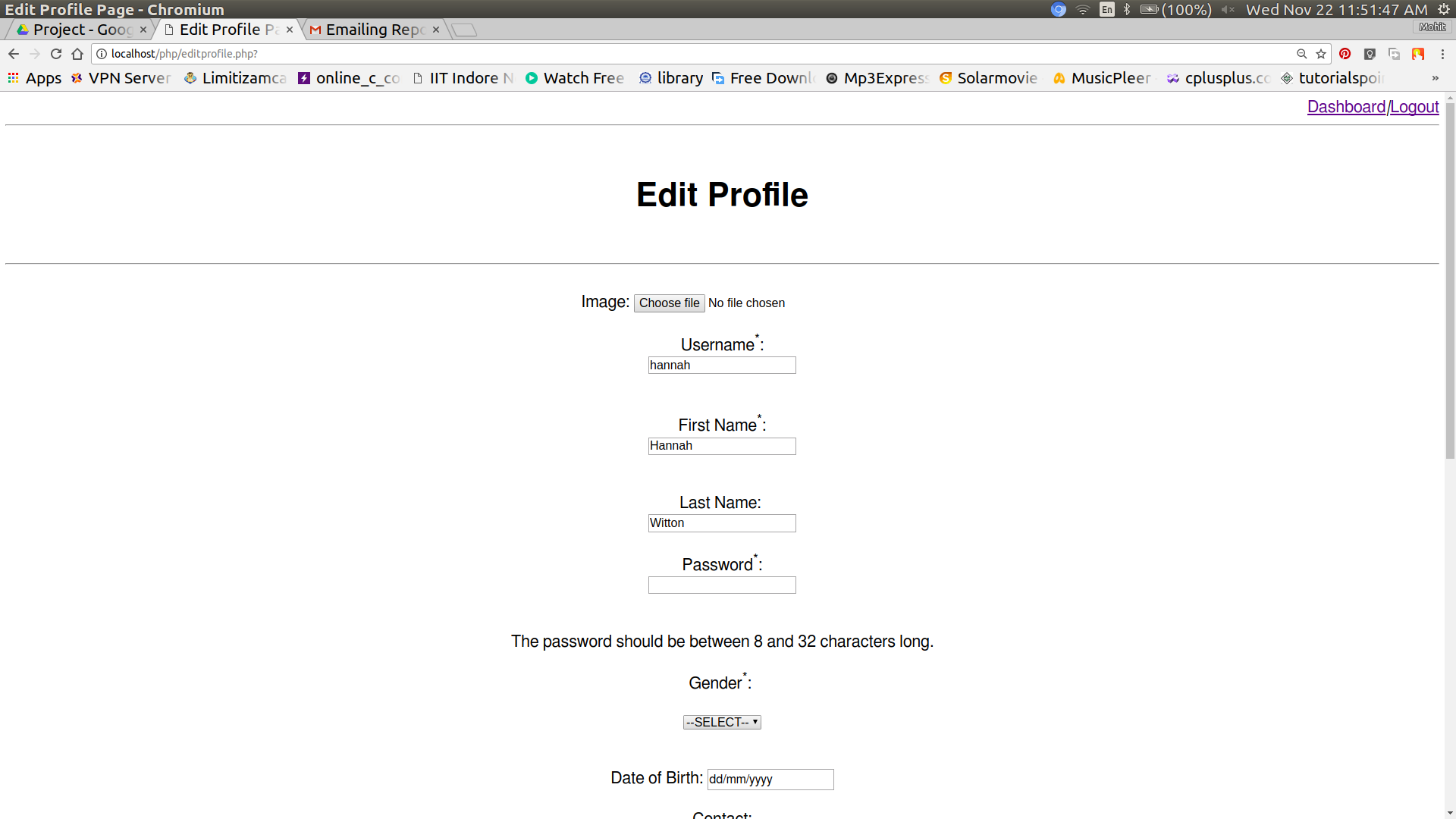


*Viewing user’s own profile*

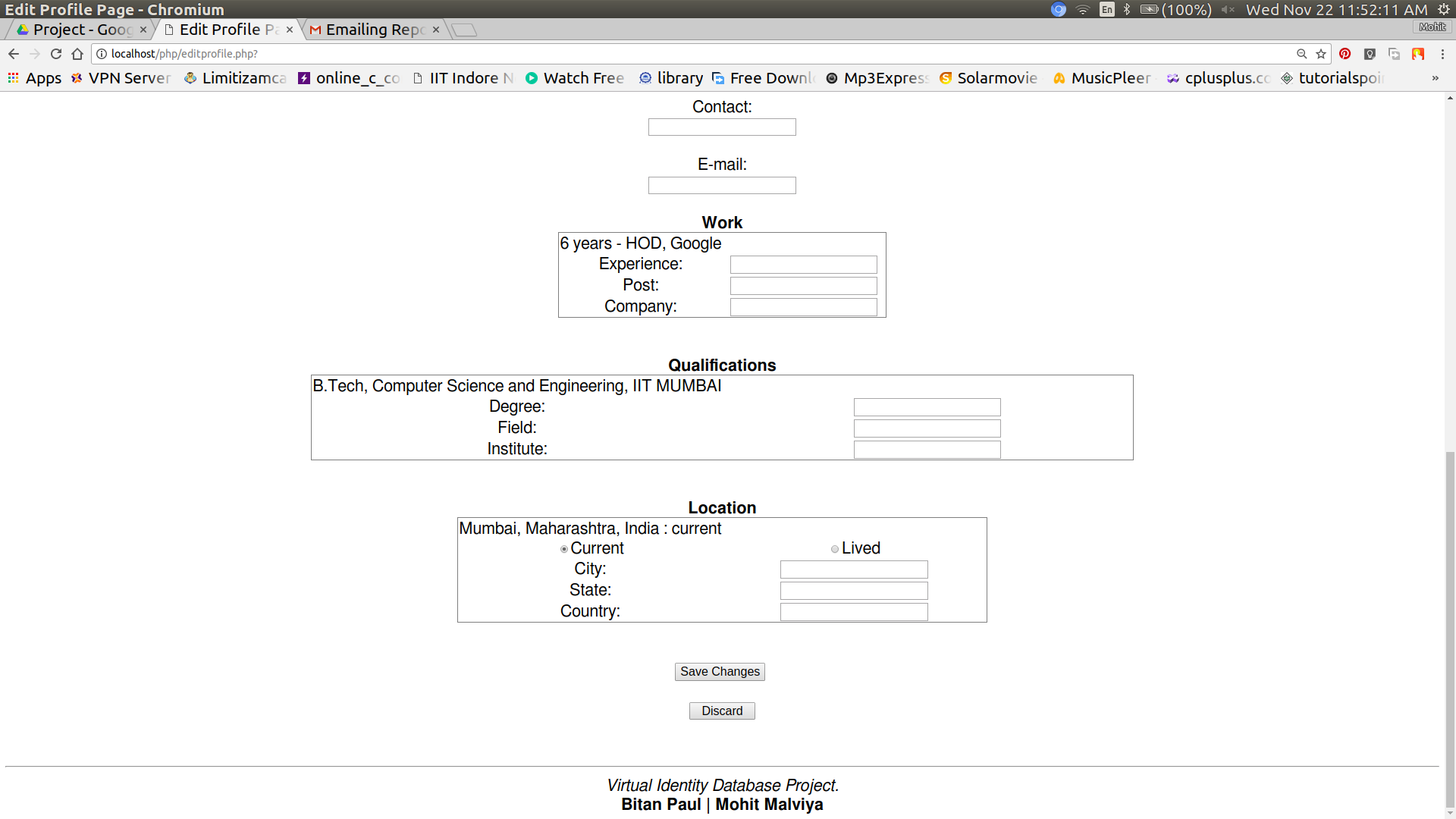
The user can view his/her own profile by clicking on the ink given on the dashboard.The page then redirects to user’s profile page.

**

User can also edit their profile here by clicking **edit profile button.**

**

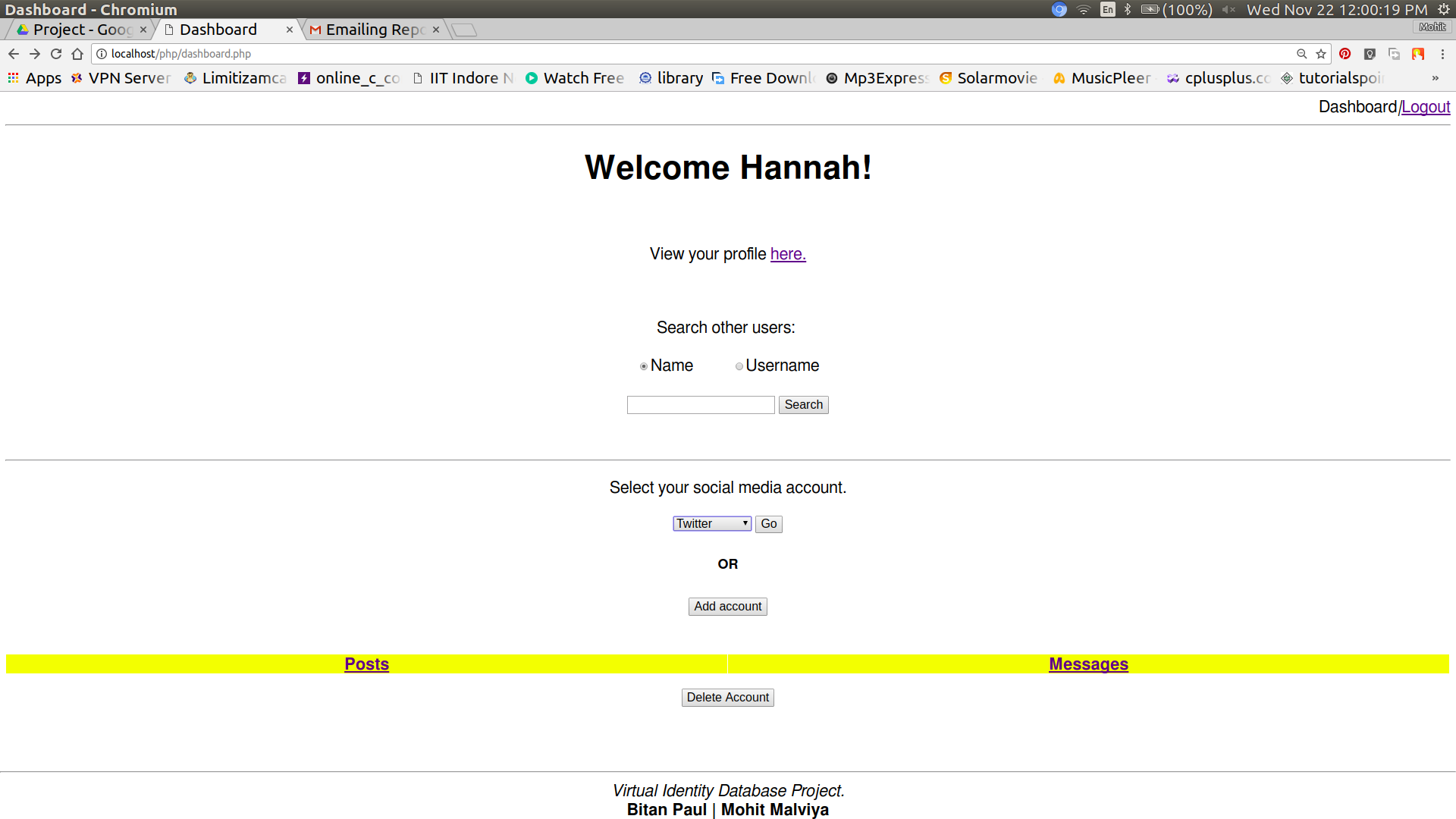
The changes done can be either saved or discarded by using the respective **save changes** or **discard** buttons.



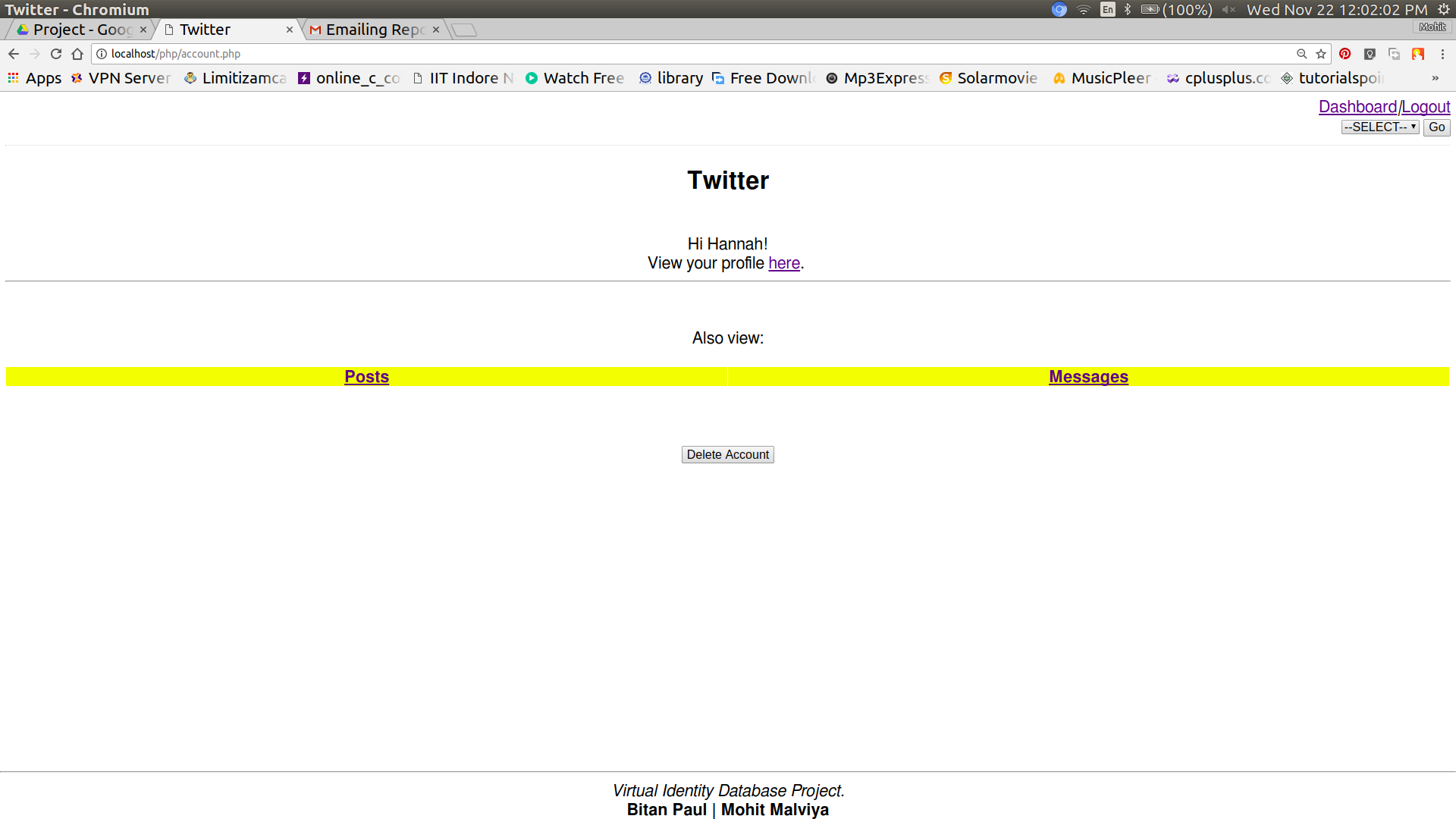
*Viewing User’s Posts and Messages*

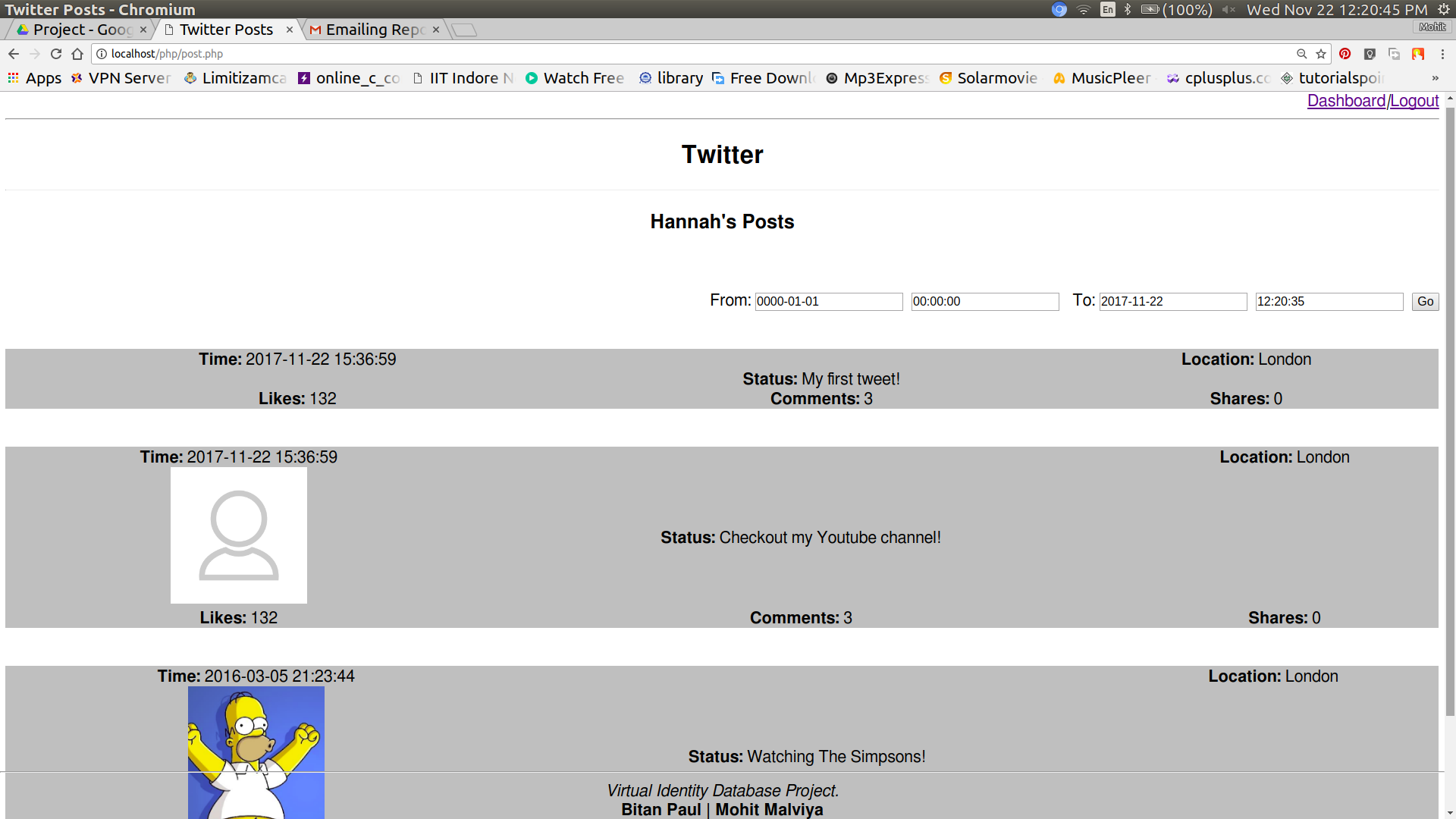
For viewing one’s posts or messages one has to first select social media account from the **drop down menu** so as to view site **specific messages and posts.**

Or the user can just click on posts and messages to view all posts and messages combined.

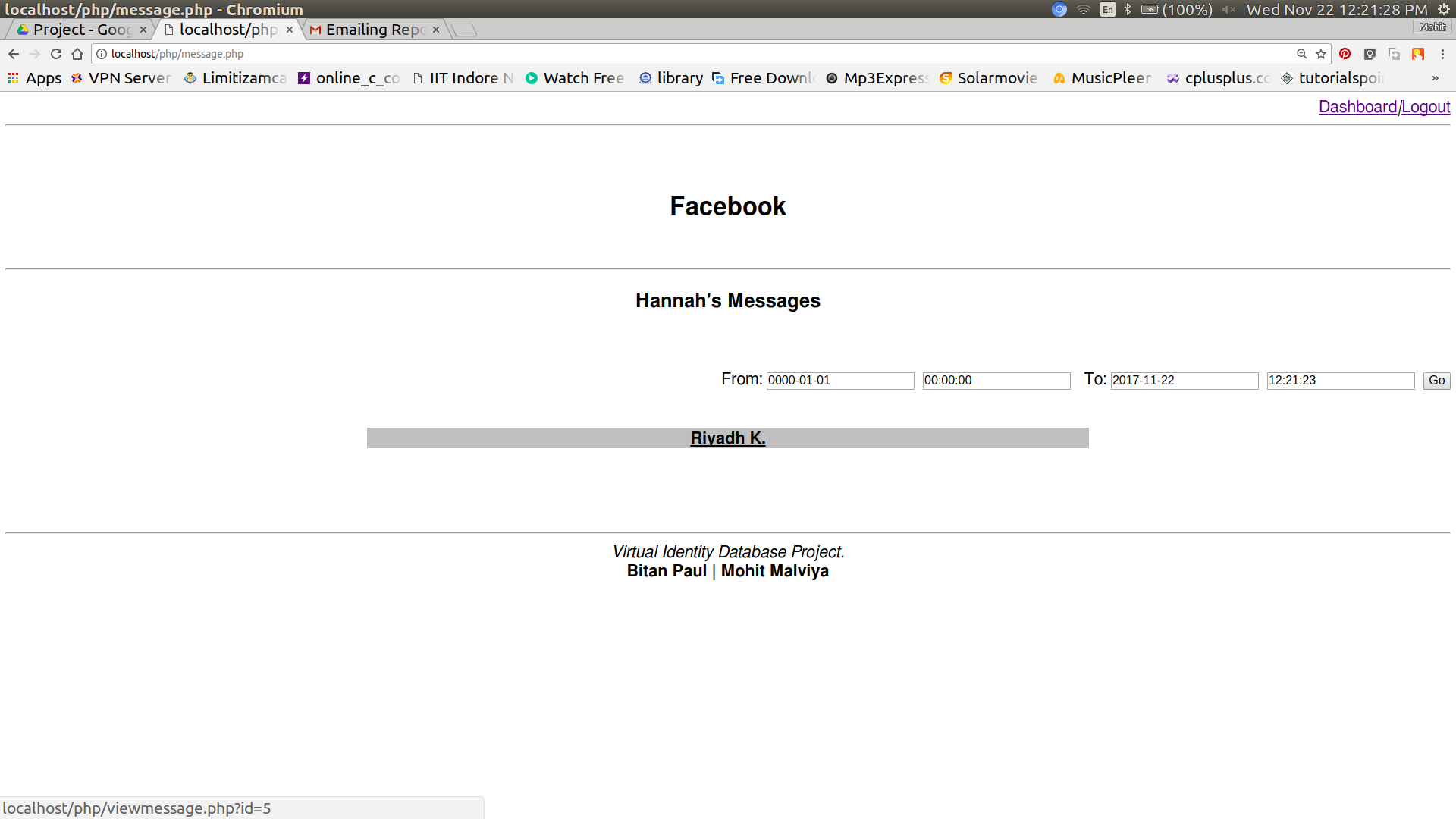


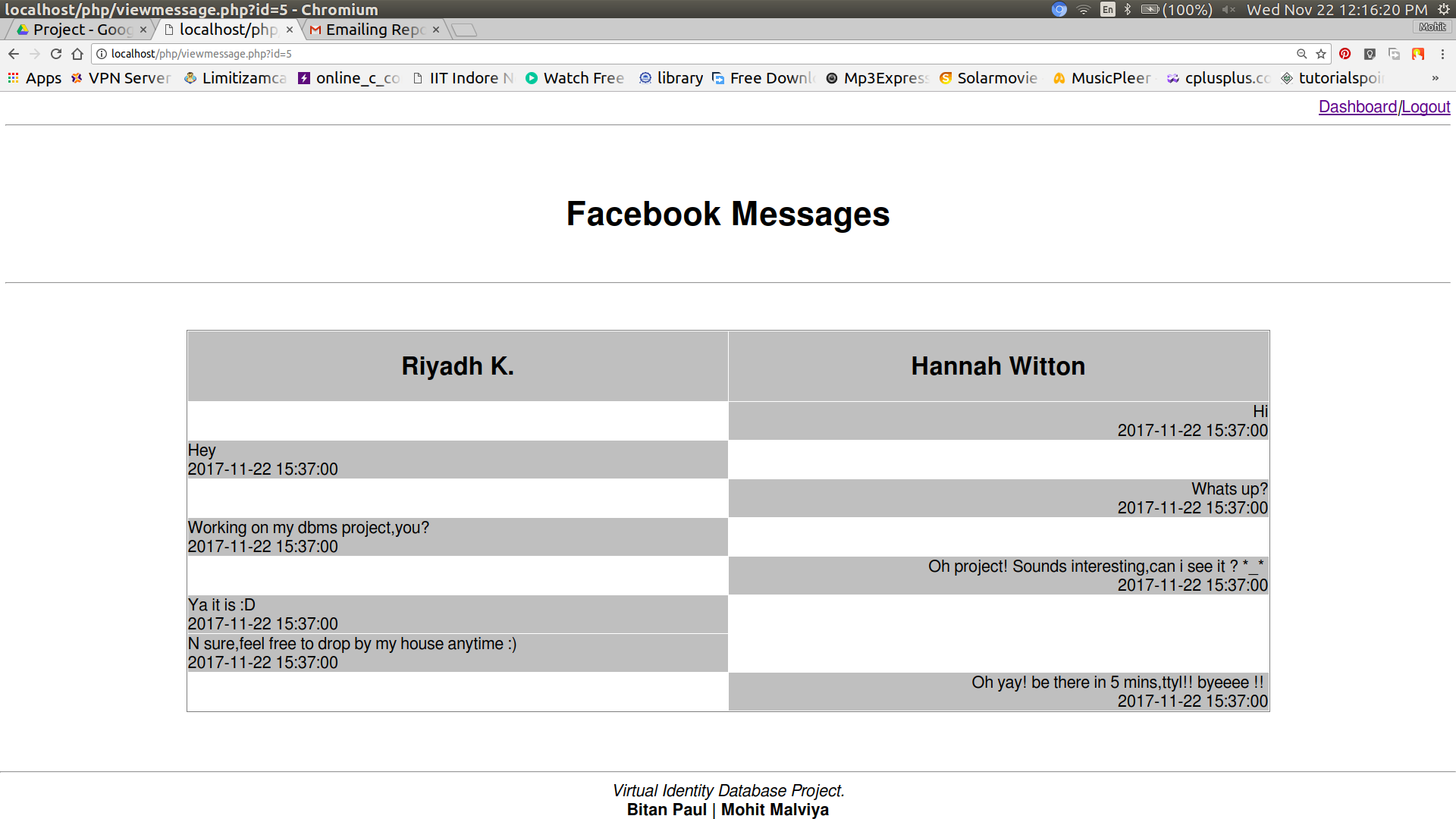
After selecting the site,the selects either posts or messages to view.

**

**

If messages are to be seen then select the user whose converstion you have to read.

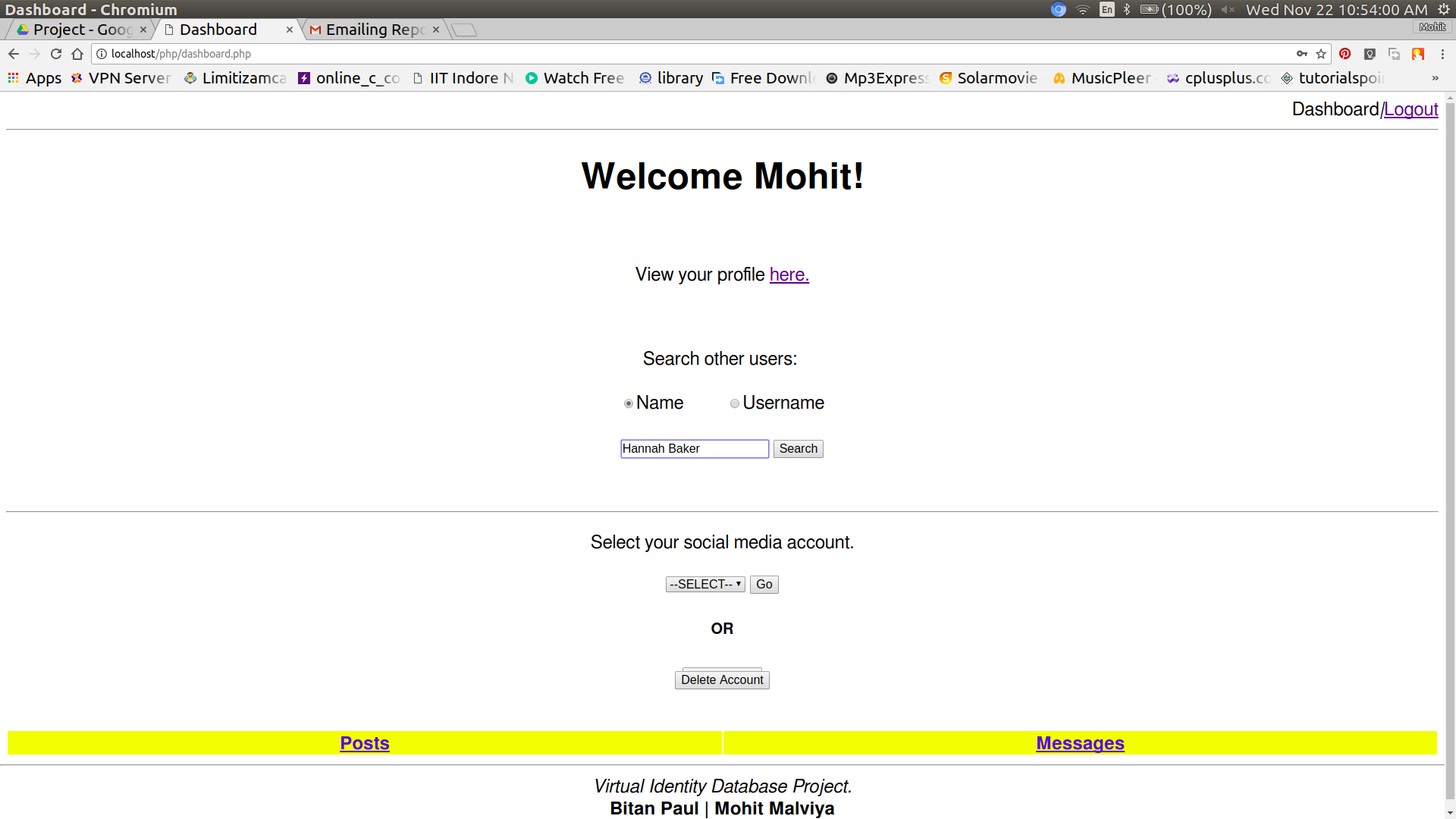


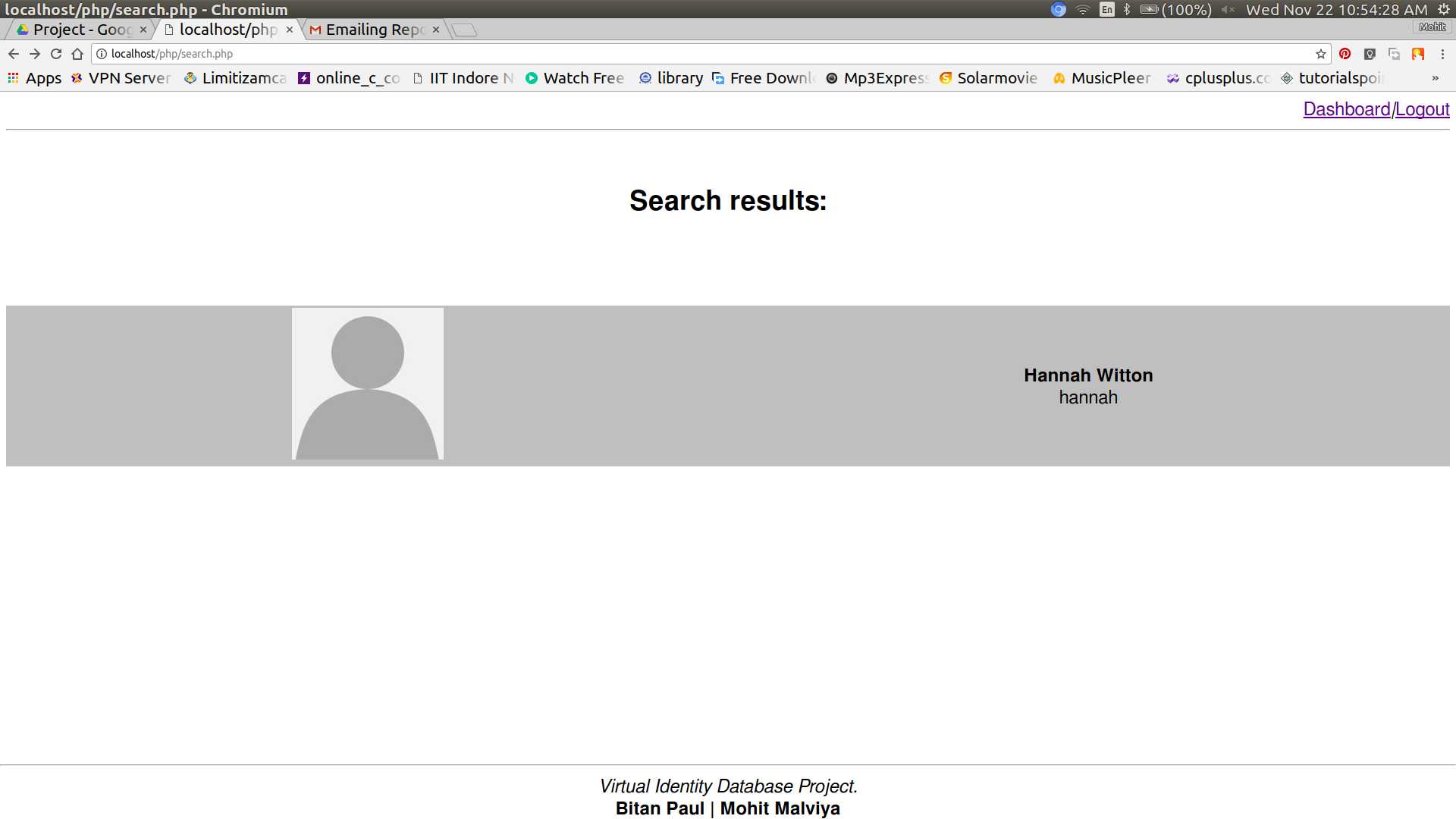
**

*Searching-*

User can search for other users from the dashboard using either their username or their name.

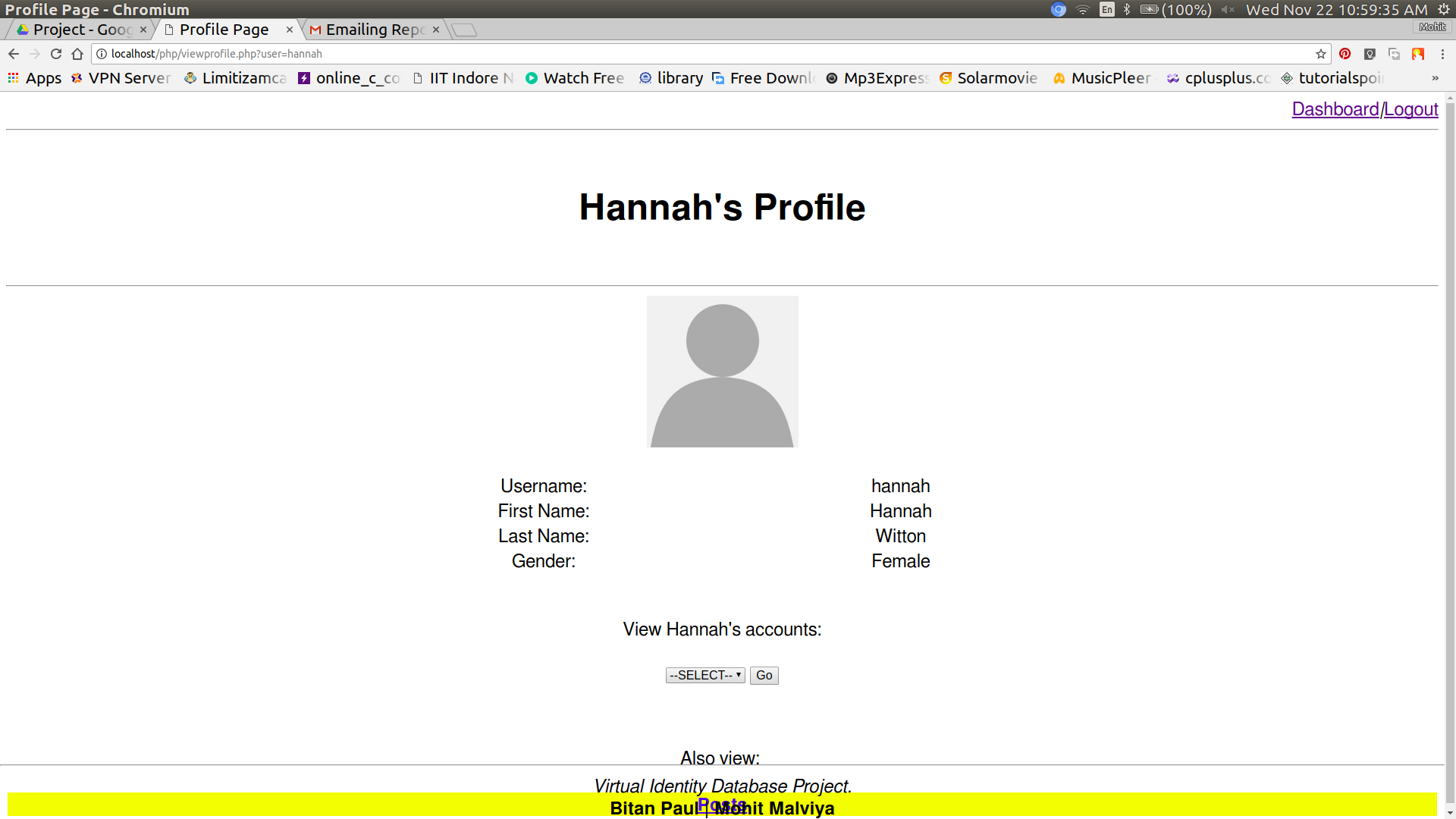
Even if the user types only the first name or only the last name correct,the searches will still be able to find the user.

**

**

On successful search the page displayed above is prompted showing the search results.As it can be seen in the page **“Hannah Witton**” was shown in the search results evenn though the searched name was “**Hannah baker**” showing that the search is optimized for the time when user ony remembers half the name correctly.

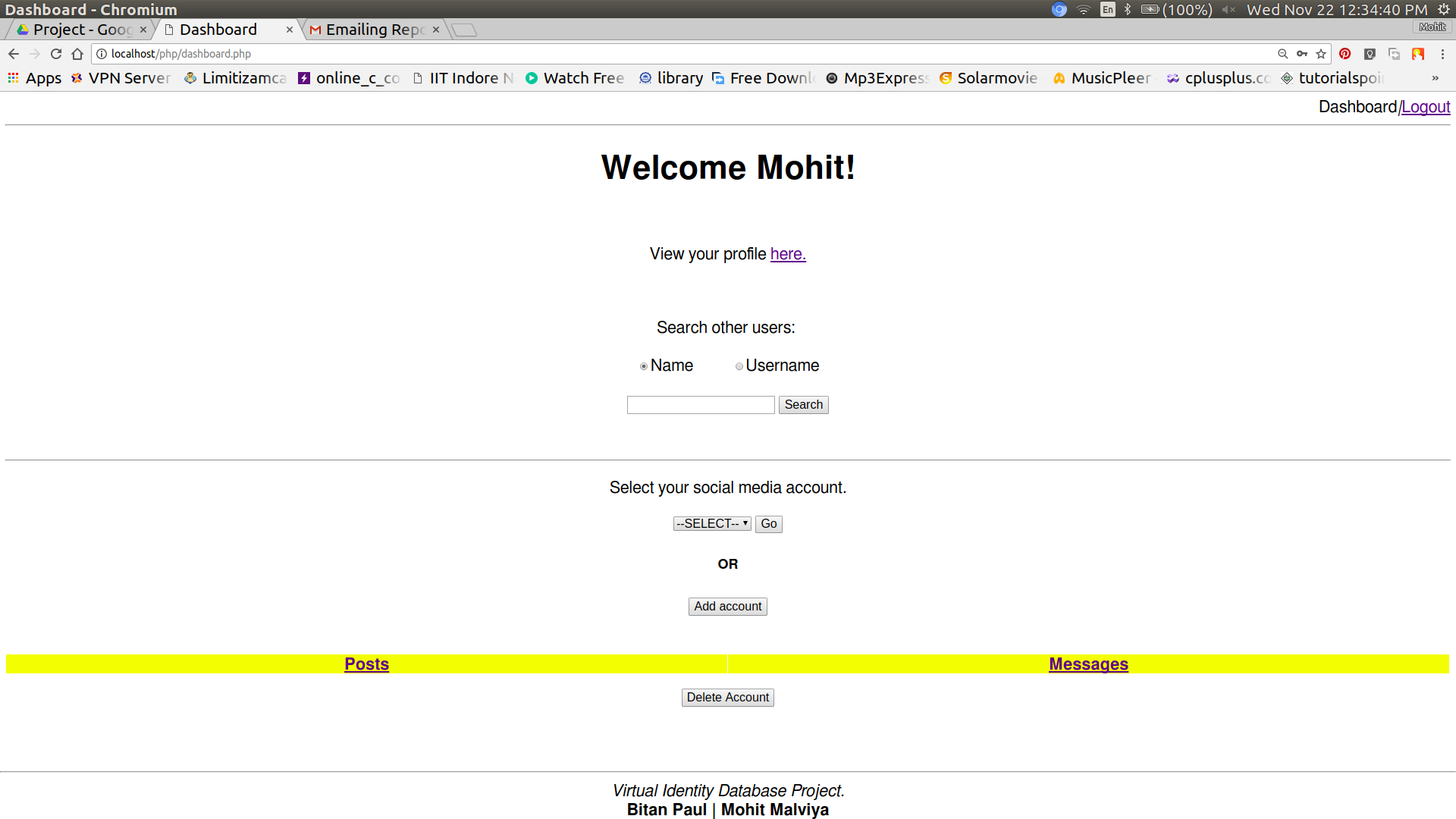
The search results displayed on the page are a link to the respective user profile containing basic information of the user.



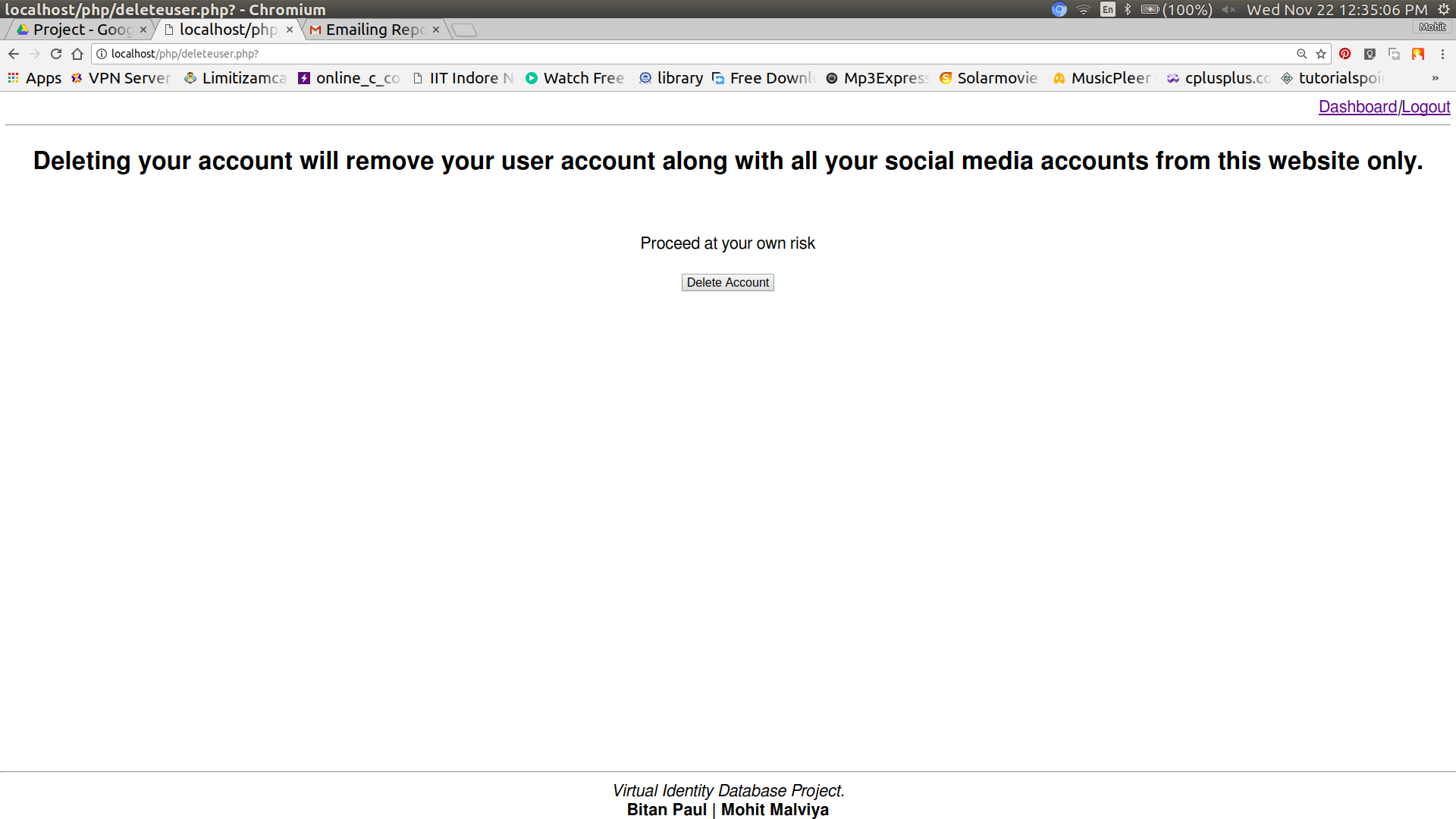
The changes done can be either saved or discarded by using the respective **save changes** or **discard** buttons.

*Deleting Account*

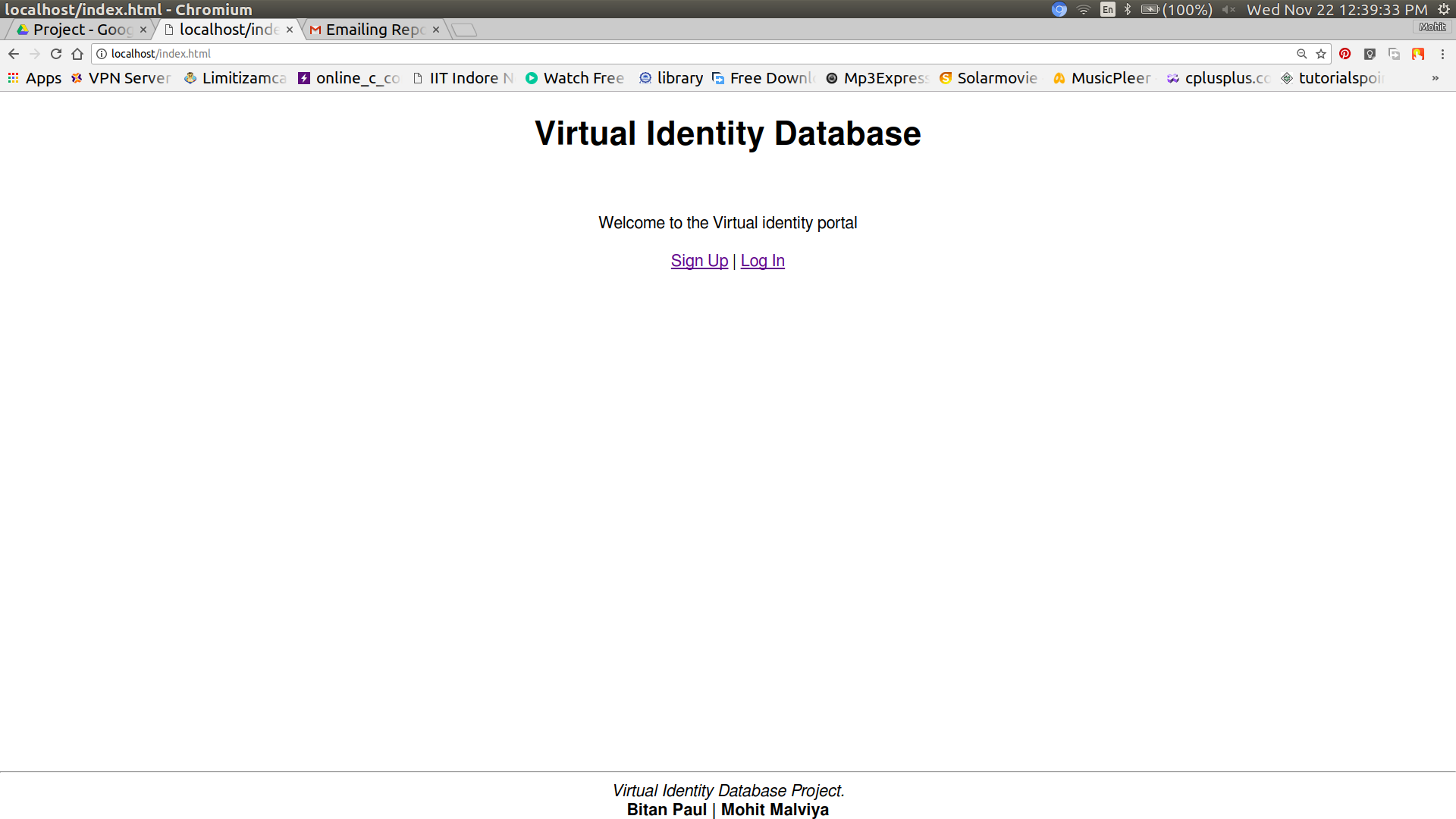
Finally, A user can delete the user account directly rather than having to delete all social media accounts first.The **automation** is handeled by **triggers** and **stored procedures**.



A warning is flashed to the user that on deleting this all records willl be lost.



After deletion the user is directed to the home page.



-----Thank you-----