

DATABASE PROJECT REPORT

CodeCorpus

Team Members

Deependra Patel 120050032

Nishant Kumar Singh 120050043

Aditya Kumar Akash 120050046

Guide

Prof. Nandlal Sarda

Introduction

There are various competitive coding sites available online which are used by students for practising questions on different topics based on level of difficulty. Very few of such sites provide searching problems based on tags and searching people.

None of the present popular sites provides features related to following different users and suggesting users to follow as well as problems based on interest received from user.

We have tried our best in implementing these features in our project “Code Corpus” along with the existing features.

Prospects

This website will be highly useful for students who are interested in coding and prepare for competitive coding contests like ACM-ICPC. Various features like suggestions and search will help save time.

In light of such a scenario, we might in future host this website online and open it for students outside the institute.

Database Design

Tables

Following are the tables in our database: (These have been derived from the above ER diagram)

users

This table stores information about the users. It has attributes like handle, password, rank, role and personal information like institution, region, email id etc.

Primary key : userid (indexed since primary key is by default indexed)

problems

This table stores details of all the problems and has attributes code, problem name, link to the problem, date of addition, difficulty and statistics.

tags

Primary key : code

tags

This table has attributes tagid, tag name and a description. Tags are nothing but groupings of problems.

Primary key: tagid

attempted

User table and problem table are related by a relation called attempted as shown in the ER diagram. This relation has been modelled by the table attempted which has attributes userid, problem code and a solved flag.

Primary key: (userid, code)

userid, code are already primary keys of users and problems respectively.

problems_tags

The problems table and tags table are related by a relation called problem_tags as shown in the ER diagram. This relation has been modeled by the table problems_tags. This table has attributes problem code and tag id.

Primary key: (code, tagid)

Both of these are primary keys in their respective tables.

users_interests

This is the table which denotes the relation between users and tags.

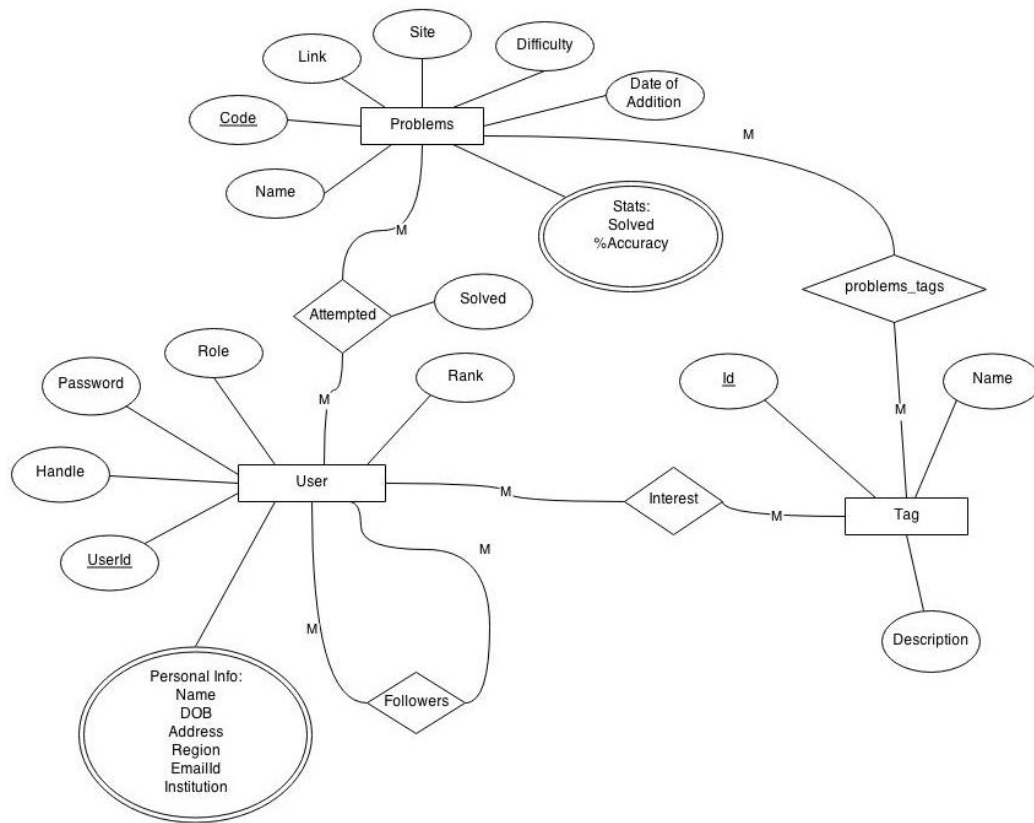
Primary key: (userid, tagid)

followers

This table denotes the relation between users. A user can follow many other users and can be followed by others. Primary key: (userid, userid)

ER Model

ER Diagram for Code Corpus



Descriptive Details on Important Parts

Admin Panel

The admin panel encompasses the features which are associated with periodic update of database information like set of problems included on codechef site, updating the list of solved problems by all people and updating rank of person as per displayed on codechef site.

Updating / Collecting Data Module

Updating Problem Set

There are 5 types of update provided for Problem set -

1. Easy
2. Medium
3. Hard
4. Challenge
5. All of these

These category provide flexibility in the update action based on traffic load.

The updates are possible only from admin panel. Considering that < 5 contests (on an average) are hosted on codechef every month the count of new problems added would be low. So making such updates periodic with period as 1 - 2 week is justified.

Description of algorithm used for updating problem set -

The following points describe best the actions undergoing for the update -

1. Get the list of problems present in our database and sort them into an array, say P
2. Scrape the problems dynamically from the codechef site based on category of update, put them in another array, say C
3. For each element in C do Binary Search in P to check if the problem is present
4. If not present go to the page of the problem using the Problem Code which describes the URL in case of codechef
5. Scrape the date of addition of problem and the tags associated with the problems
6. Update the tables in the database for the new problem and the tags.

Updating user Information (all at once)

The user information changes are computed only after contests. There is an option for users to update their information from their side. Thus the update is only needed for users which are infrequent on our site. Thus the periodic update is best option for user information update with period as a fortnight to month .

This update is rather expensive and hence an option for users has been provided which lets them do manual update on their profile.

Description of algorithm for User Info Update -

The following points best describe the actions for updating user information from admin side

1. Obtain the list of users from database
2. For each user Scrape the user page from codechef - url is based on username
3. Obtain the list of solved problems from the page, say set C
4. Obtain the list of solved problems from database, say set P
5. Update the new problems if any based on comparison between C and P (Use binary search to find out $C \setminus P$). Put the new information in the related tables.
6. Get the new rank from page and update the rank of the user

Updating user information by users

The users have an option to update their information regarding the list of solved problems and rank . The steps followed are the same as updating user information from admin side.

Suggestion Module

Suggesting Problems to users

Suggestion about which problems to solve is one of the key features provided by our project which is absent on codechef as well as most coding platforms.

Following are the factors which are considered for suggesting problems to users -

1. User's interests - consisting of tags (concepts)
2. Tags from the past problems solved by the user
3. Tags from the problems solved by the users which are being followed
4. Unsolved problems from the users which are being followed
5. Number of followed people who solved a given problem

Considering the above factors we design an algorithm which make the best suggestions consisting of 25 problems.

Algorithm Description -

Notations -

IN - set of interest tags

T - set of tags which would be considered for suggestion

Ti - ith tag, WTi - weight of ith tag

P - set of problems which are taken for suggestion

Pi - ith problem , PWi - weight for ith problem

The steps -

1. Read the interest tags of user and append them to T as well as IN. Assign weight C1 to each such tag
2. Read the tags from the past solved problems and add them to T. Assign weight C2 to them if they did not exist earlier else add C2 to existing value
3. Read all the problems from followers and add them to Set P if they are not in the list of problems solved by user.
4. For each problem consider the tags associated with the problem. For each tag which exists in set T add weight C3 to the tag and for those not present initialize them with weight C3.
5. Obtain the list of problems from database which have tags from the set IN. Add these problems to set P.
6. In case $|P| < 25$ take $\max(50, 2 * (\text{number of solved problems}))$ number of problems from list of problems sorted by accuracy in descending order. Add these to set P.
7. Now for each Pi, add to PWi the weight of the tags as found from set T if they belong to the set else add nothing.
8. Sort the set P based on PWi and take top 25 problems . Output these problems as the suggestion list.

As we can see C1 accounts for the interests , C2 for the past solved problems and C3 along with adding C3 for each followed person takes care of followed users dependency.

Adding problems based on accuracy takes care of cases where the user is new and has not provided any information on interest and has very few solved problem.

Suggesting people to follow

The users also get a suggestion of a list of people whom they would ideally like to follow.

Factors on which the suggestion depends:

1. User Interests: The users have interests in form of tags. Other users whose interests match these should be followed.
2. Current followers: The people who are being followed currently also follow some other users. These users should also be connected to the current user.
3. Problems solved: The users who have a lot of problems in common with the current user.

4. Tags of the problems solved: If other users have in their interests the tags of the problems solved by the current user, they should also be connected with the current user.

Taking the above factors into account, we provide a list of 15 users who the current user should follow.

A simple algorithm to select these users is as follows:

We have four constants c_1 , c_2 , c_3 , c_4 each denoting the weights assigned to the four factors as stated above.

c_1 - Fetching the users who have in their interest table, tags which correspond to the problems solved by the user

c_2 - Fetching the users based on the current followers

c_3 - Fetching the users based on interests

c_4 - Fetching the users based on common problems solved

We make a hashmap of type $\langle \text{Integer}, \text{double} \rangle$: Integer for userid and double for weight

Steps are outlined below:

1. First we fetch the userid of the users who have in their interest table, tags corresponding to the problems solved by the user.
2. For the entries in the above list, we add an entry in the hasmap with value as c_1 .
3. Next, we fetch the userid of the users based on the current followers.
4. For the entries in the above list, we check if the userid is already present in the hashmap.
5. If it is we simply add c_2 to its current weight.
6. If not we make a new entry in hashmap corresponding to userid and assign the weight as c_2 .
7. We repeat steps 4-6 for userids which have been fetched based on interests and also which have been fetched based on common problems solved.
8. We also need to filter the above obtained set of userids. It should not contain the userid of the user itself and also the people who they currently follow.

Now, we simply sort the above hashmap using values and display the top 15 results.

Search Module

This module has been designed for assistance of users to dig out the useful data for their use.

Options have been provided to search:

- ❖ Users based on handle
- ❖ Users based on their name
- ❖ Problems based on the name
- ❖ Tags based on name

Autocompletion has also been added in the searchbox. The set of values to choose from while providing the suggestions for autocompletion is being dynamically changed based on the type which is selected by the user using the dropdown provided.

To implement the above functionality an onclick function has been put in the button which calls a function which does the task of updation. This requires an update in the array which stores the suggestions.

Implementation:

1. For searching the users based on handle and name, simple sql queries using the 'like' option have been used.
2. The result is being displayed in form of a table with two columns first of which has the handle and second shows the name
3. Clicking on the handle would lead to user profile page
4. For searching problems based on the name, the problems have been fetched using problem table using 'like'.
5. The problem details consist of problem name and its tags.
6. Clicking on the tags will display all the problems which have that tag
7. Similar queries have been used to implement search based on tag name.

Register/Security Module

For registration user has to input his codechef handle along with other personal information. All of them are checked for data integrity. If all is good then data of the given user is scraped from the codechef website which is inserted into the database tables. User can also insert new interests and can also update personal info and password.

We don't allow any user to view any page without registering and logging to our website. During logging in, we check if user is already registered on our system and the entered credentials are correct, we set the session handle. User can also logout from the system whenever he wants which destroys the session.

Profile module

Each user has a profile displaying his personal info, problems solved by him, his followers and his interests. Any user can follow/unfollow other user by clicking a button on the profile. This helps for problem suggestions and user suggestions. The list of problems solved by that user is given. On clicking any particular problem code, an ajax request is sent which brings back all the information about that problem. We used this to minimise the data transfer and the number of queries.

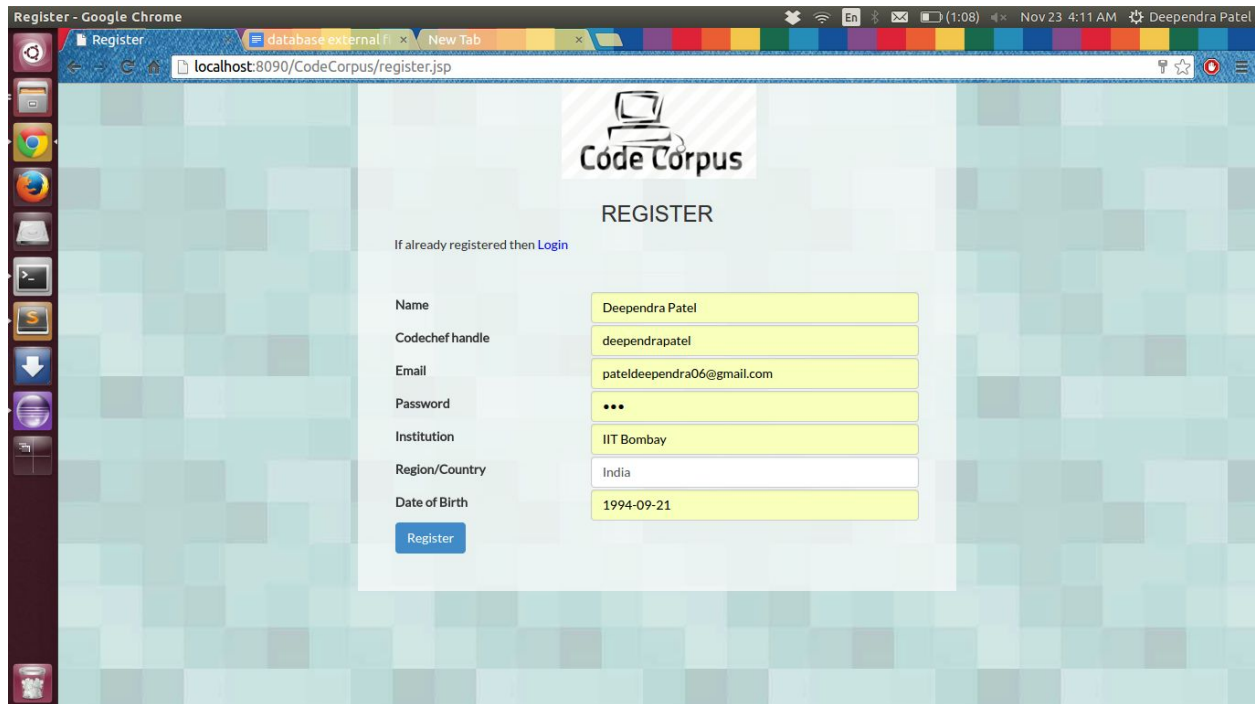
Packages/Classes

1. **login** - Deals with all login/logout links. Sets and destroys session variables correspondingly.
2. **profile** - Displays user profile after executing query.
3. **register** - Handles all transactions related to user profile like registering for first time, updating personal info, passwords, etc.
4. **search** - Searches for users, problems, tags based on selected option and given keyword.
5. **suggestions** - Suggests the problems and followers based on heuristic described in detail earlier
6. **updateProblems** - Scrapes the problems from site (www.codechef.com) and updates the database tables accordingly

URL mappings and functionality

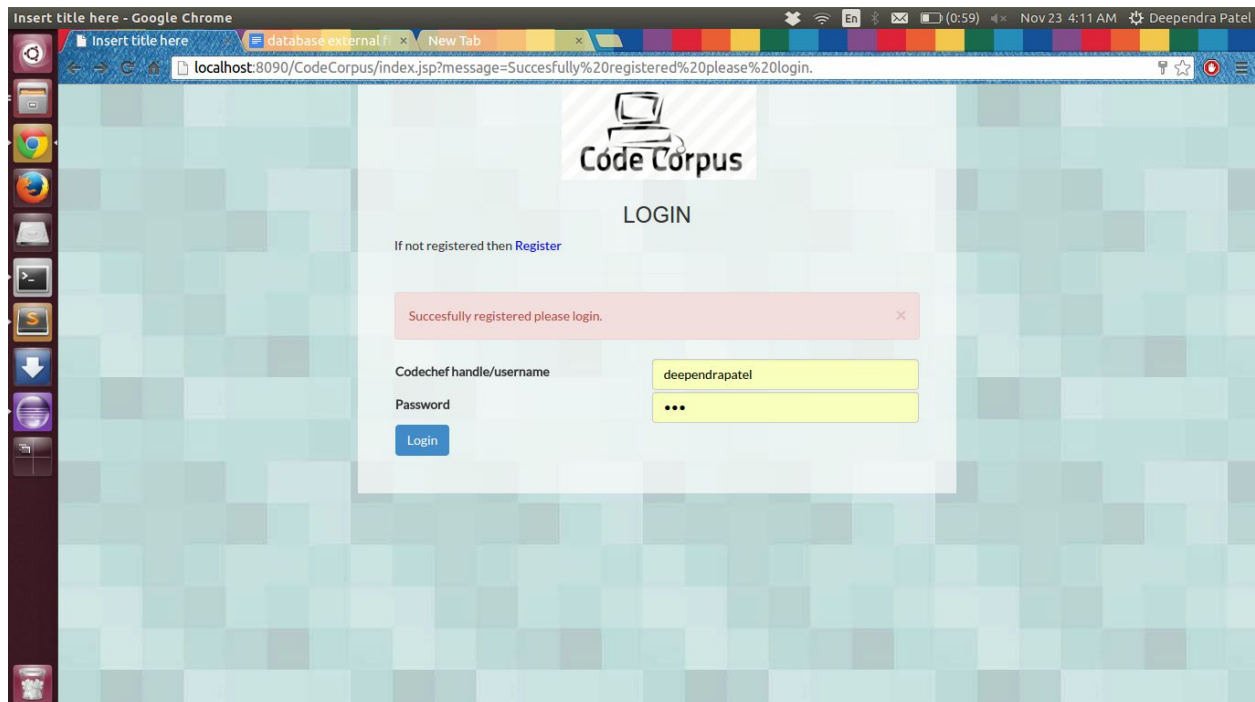
1. **/profile[profile package]** - Retrieves handle in get request, if nothing found displays users own profile. otherwise the profile of the asked user.
2. **/index.jsp** - Displays forms for logging in user
3. **/register.jsp** - Displays form for registering new user
4. **/home [search package]** - Provides search box for searching any user or problem
5. **/suggestions [suggestions package]** - Displays suggested problems and users to follow to the logged in user
6. **/accountUpdate[register package]** - Displays forms to update password/personal info. User can also add his interests here. It also lists all the users he is following.
7. **/logout [login package]** - Destroys the session of the user.

Screen Shots - Project in Action



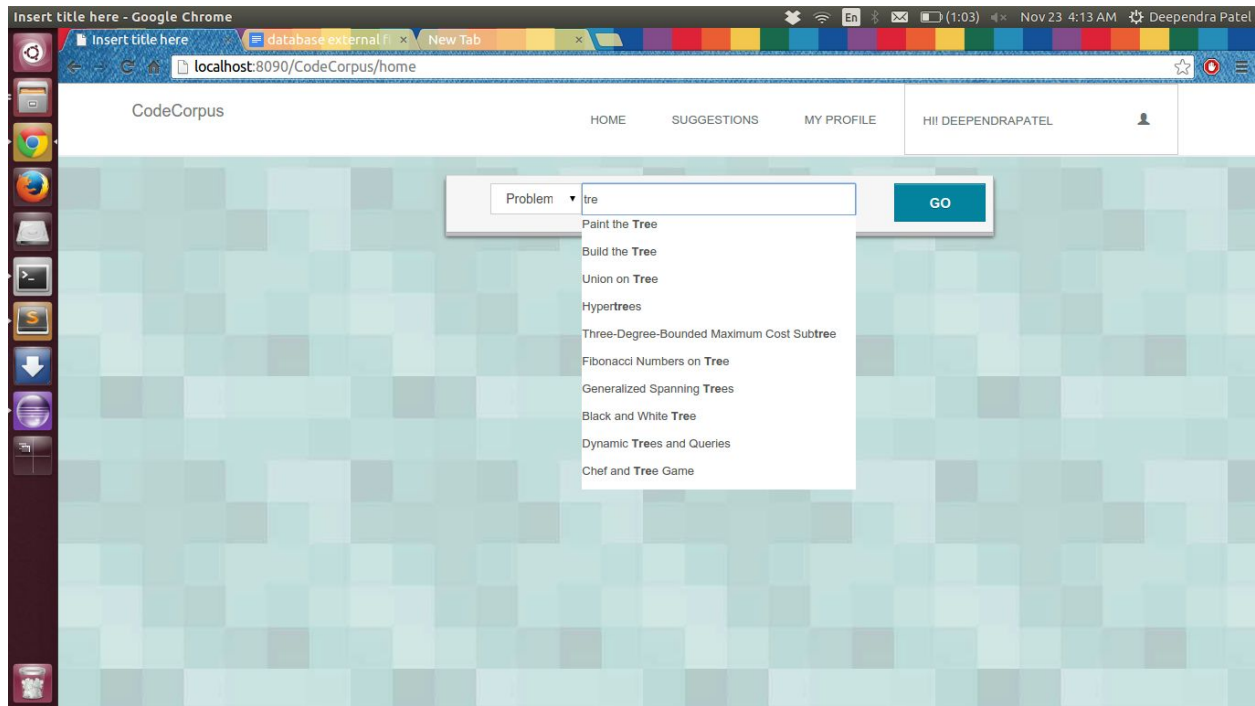
The screenshot shows a web browser window titled "Register - Google Chrome". The address bar displays "localhost:8090/CodeCorpus/register.jsp". The page features the "CodeCorpus" logo at the top, which includes a stylized laptop icon. Below the logo, the word "REGISTER" is prominently displayed. A link "If already registered then Login" is provided. The registration form contains the following fields: Name (filled with "Deependra Patel"), Codechef handle (filled with "deependrapatel"), Email (filled with "pateldeependra06@gmail.com"), Password (masked with "..."), Institution (filled with "IIT Bombay"), Region/Country (filled with "India"), and Date of Birth (filled with "1994-09-21"). A blue "Register" button is located at the bottom of the form.

Register Screen

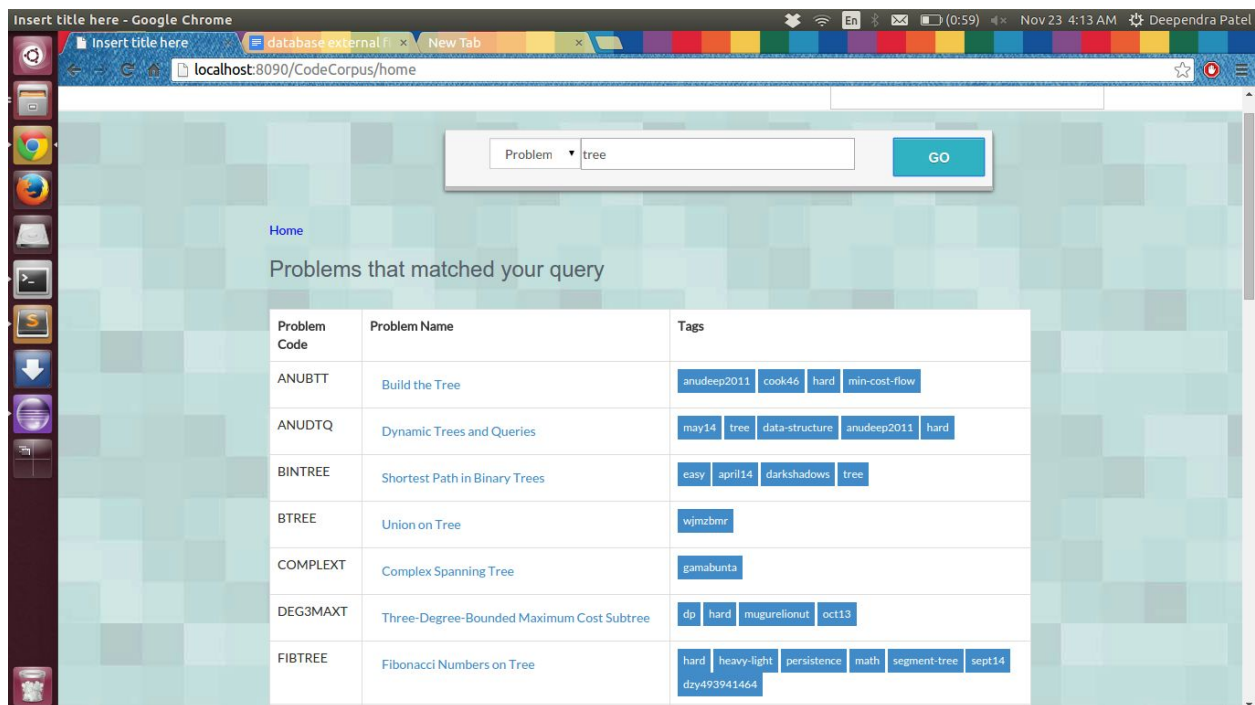


The screenshot shows a web browser window titled "Insert title here - Google Chrome". The address bar displays "localhost:8090/CodeCorpus/index.jsp?message=Successfully%20registered%20please%20login.". The page features the "CodeCorpus" logo at the top. Below the logo, the word "LOGIN" is prominently displayed. A link "If not registered then Register" is provided. A red message box at the top of the login form states "Successfully registered please login.". The login form contains the following fields: Codechef handle/username (filled with "deependrapatel") and Password (masked with "..."). A blue "Login" button is located at the bottom of the form.

Login Screen



Autocomplete in Action



Searching - Here searching 'tree'

Insert title here - Google Chrome

localhost:8090/CodeCorpus/suggestions

CodeCorpus

HOME SUGGESTIONS MY PROFILE HII! DEEPPENDRAPATEL

Suggested Problems

Code	Name	Date	Diff	Users	Acc.	Tags
ANUWTA						
CSUB	Count Substrings	2014-04-20	easy	469	31.55	ad-hoc cakewalk july14 darkshadows
AMMEAT						
TOTR						
DIRECTI	Reversing directions	2012-11-26	easy	720	48.13	cook29 pieguy ad-hoc cakewalk
BUY1GET1						
ATTIC						
LAPIN	Lapindromes	2013-05-01	easy	1600	52.39	ad-hoc vamsi_kavala june13 cakewalk
NAME1						
NAME2						
SALARY						
CHRECT						
TAVISUAL						
CONFLIP						

Suggested Users

handle	Name
himank77	NA
m1y3	NA
starfishbits	NA
vindu525	NA
xyz24	NA
gauravg1	NA
arijit_pande	NA
atul05kumar	NA
krish_agarwal	NA
manicodes	NA
va123	NA
primorial	NA
salsreloaded	NA
centralspike	NA
deep3695	NA


Suggestion - Problems - Users

Insert title here - Google Chrome

localhost:8090/CodeCorpus/profile

CodeCorpus

HOME SUGGESTIONS MY PROFILE HII! DEEPPENDRAPATEL



deependrapatel

IIT Bombay, India

Basic Problems Solved Interests Followers

Basic Information

Rank	NA
Name	Deependra Patel
Codechef Handle	deependrapatel
Email	pateldeependra06@gmail.com
Date Of Birth	1994-09-21

Profile Page of a User

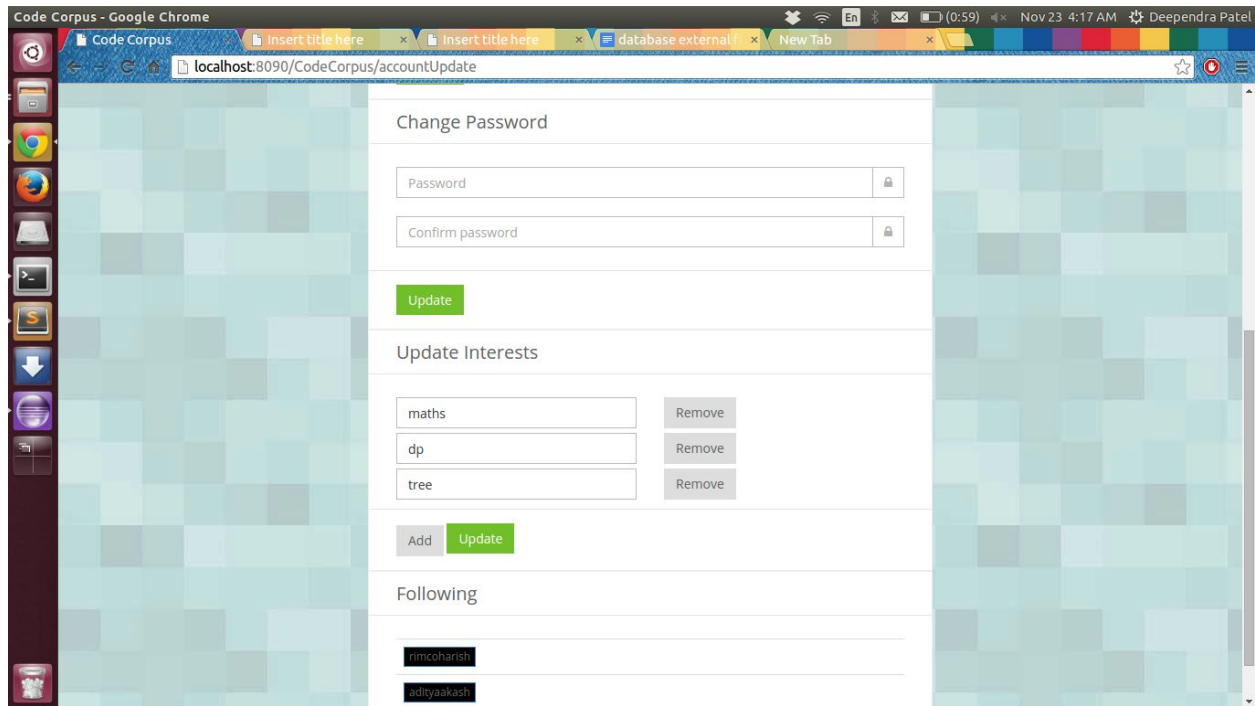
The screenshot shows the CodeCorpus user profile for 'deependrapatel' (IIT Bombay, India). The profile is displayed on a web browser window titled 'Insert title here - Google Chrome'. The browser's address bar shows 'localhost:8090/CodeCorpus/profile'. The profile page has a header with 'CodeCorpus' and navigation links: 'HOME', 'SUGGESTIONS', 'MY PROFILE', and a user greeting 'HI! DEEPENDRAPATEL'. The profile section includes a circular avatar with a red background and a white robot head icon, the name 'deependrapatel', and the institution 'IIT Bombay, India'. Below this, there are tabs for 'Basic', 'Problems Solved' (selected), 'Interests', and 'Followers'. The 'Problems Solved' tab displays a table of solved problems.

Code	Name	Date	Diff	Users	Acc.	Tags
TWTCLOSE	Closing the Tweets	2012-06-06	medium	1655	48.07	flying_ant cook23 cakewalk array-string
COINS						
SNAPE	Snape and Ladder	2013-01-24	easy	1681	61.15	implementation cook45 vinayak.garg geometry cakewalk
GCD2						
OJUMPS						
RRSTONE	Stone	2014-04-14	easy	497	22.37	ad-hoc easy Rubanenko may14

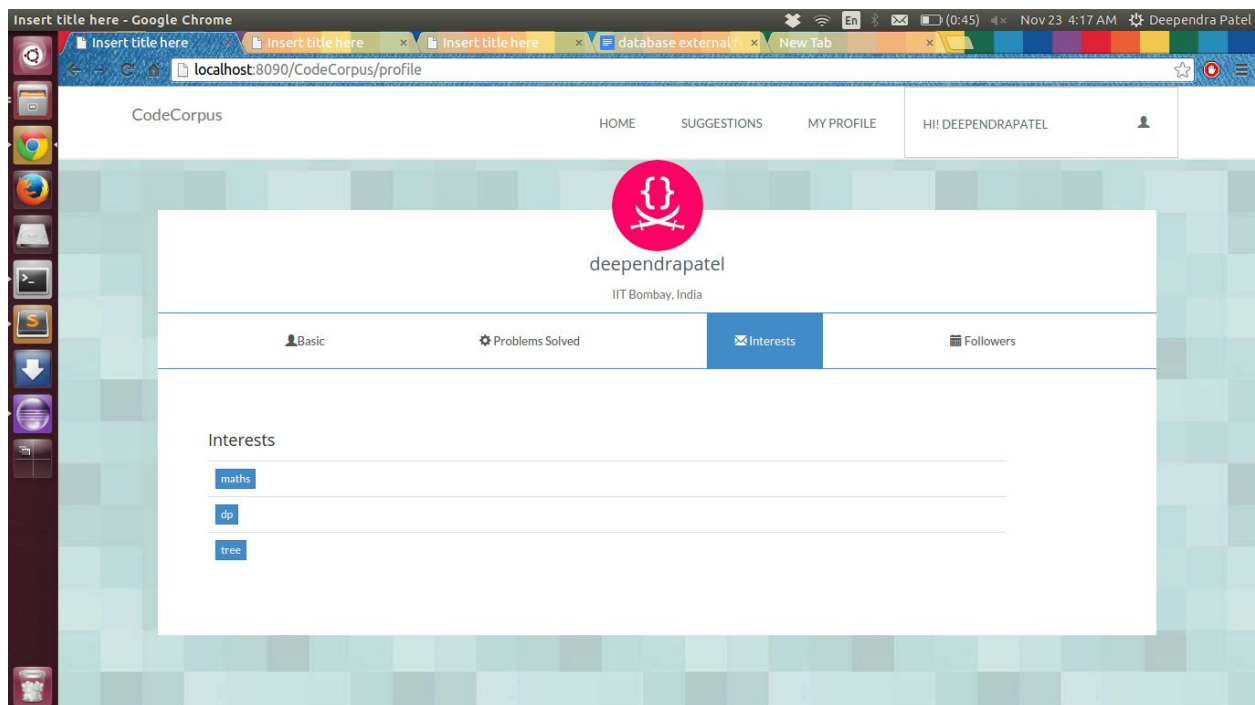
Problems Solved of user

The screenshot shows the 'CodeCorpus - Google Chrome' browser window displaying the 'accountUpdate' page. The browser's address bar shows 'localhost:8090/CodeCorpus/accountUpdate'. The page has a header with 'CodeCorpus' and navigation links: 'HOME', 'SUGGESTIONS', 'MY PROFILE', and a user greeting 'HI! DEEPENDRAPATEL'. The main content area is divided into two sections: 'Update Problems Solved' and 'Update Details'. The 'Update Problems Solved' section has a green 'Update Problems Solved' button. The 'Update Details' section contains a form with the following fields: 'Name' (Deependra Patel), 'Region/Country' (India), 'Institution' (IIT Bombay), and 'Email address' (pateldeependra06@gmail.com). Below the form is a green 'Update' button. The 'Change Password' section has two password input fields: 'Password' and 'Confirm password', each with a lock icon on the right.

Settings



update interest, following



Interests

Insert title here - Google Chrome

localhost:8090/CodeCorpus/profile

deependrapatel
IIT Bombay, India

Basic Problems Solved Interests Followers

Problems Solved

Code	Name	Date	Diff	Users	Acc.	Tags
TWTCLOSE	Closing the Tweets	2012-06-06	medium	1655	48.07	flying_ant cook23 cakewalk array-string
COINS						
SNAPE						
GCD2	GCD2	2008-12-01	easy	1921	25.90	admin
OJUMPS						
RRSTONE	Stone	2014-04-14	easy	497	22.37	ad-hoc easy Rubanenko may14
ANUUND						
ANUDTC						
GUESS	Guessing Game	2014-04-20	easy	271	36.60	ad-hoc combinatorics june14 simple darkshadows
CHEFZOT						

Solved Problems

Insert title here - Google Chrome

localhost:8090/CodeCorpus/profile?handle=adityaakash

CodeCorpus

HOME SUGGESTIONS MY PROFILE HII! DEEPENDRAPATEL

adityaakash
IIT Bombay, India

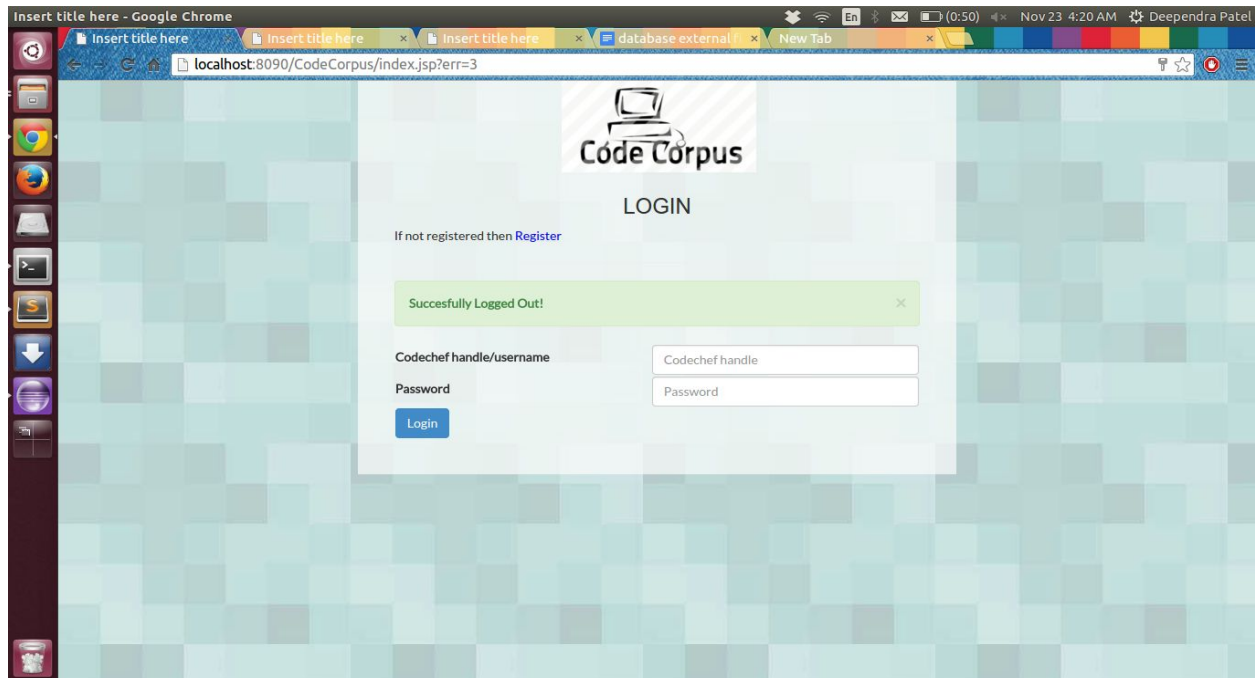
Unfollow

Basic Problems Solved Interests Followers

Followers

- [rimcoharish](#)
- [deependrapatel](#)

Followers



User successfully logged out