CS 251 Project: A Report by Group 22

Utkarsh Kumar
130050022
utkarsk@cse.iitb.ac.in

Dibyendu Mondal
130050046
dibyendu@cse.iitb.ac.in

Sai Sandeep
130050052
saisandeep@cse.iitb.ac.in

November 3, 2014

CS 251 Project: A Report by Group 22

Contents

CS 251 Project: A Report by Group 22

Contents

CS 251 Project: A Report by Group 22

Contents

CS 251 Project: A Report by Group 22

Contents

Lab 10: java Implementation

Contents

Lab 10: java Implementation

Contents

Lab 10: java Implementation

Contents

1. Gale Shapley Allocation [2]

Contents

(i) It is relatively simple to implement.

1. Gale Shapley Allocation [2]

Contents

(i) It is relatively simple to implement.
(ii) MeritList class isn't needed.

1. Gale Shapley Allocation [2]

Contents

(i) It is relatively simple to implement.
(ii) MeritList class isn't needed.
(iii) We stored the ranks in candidate class and used them.

1. Gale Shapley Allocation [2]

Contents

(i) It is relatively simple to implement.
(ii) MeritList class isn't needed.
(iii) We stored the ranks in candidate class and used them.
(iv) Foreign candidates can be handled easily after everything is done.

2. Merit List Allocation [2]

Contents

(i) Checking it's correctness is simply too tough.

2. Merit List Allocation [2]

Contents

(i) Checking it's correctness is simply too tough.
(ii) You need to manually evaluate the test cases to test the correctness of algorithm, which is a very tedius task in this case.

3. Info

Contents

(i) Candidates will map candidate id to the object candidate.

3. Info

Contents

(i) Candidates will map candidate id to the object candidate.
(ii) Programs will map the program id to the corresponding object VirtualProgramme.

3. Info

Contents

(i) Candidates will map candidate id to the object candidate.
(ii) Programs will map the program id to the corresponding object VirtualProgramme.
(iii) In GS, we stored the categories,
1 : General, 2 : OBC, 3 : SC, 4 : ST,
5 : GEPD, 6 : OBCPD, 7 : SCPD, 8 : STPD.

3. Info

Contents

(i) Candidates will map candidate id to the object candidate.
(ii) Programs will map the program id to the corresponding object VirtualProgramme.
(iii) In GS, we stored the categories,
1 : General, 2 : OBC, 3 : SC, 4 : ST,
5 : GEPD, 6 : OBCPD, 7 : SCPD, 8 : STPD.
(iv) We stored these categories in the Candidate and VirtualProgramme as well. It helped to reduce the code a lot.

3. Info

Contents

(i) Candidates will map candidate id to the object candidate.
(ii) Programs will map the program id to the corresponding object VirtualProgramme.
(iii) In GS, we stored the categories,
1 : General, 2 : OBC, 3 : SC, 4 : ST,
5 : GEPD, 6 : OBCPD, 7 : SCPD, 8 : STPD.
(iv) We stored these categories in the Candidate and VirtualProgramme as well. It helped to reduce the code a lot.
(v) And then to compare based on two ranks, we used multiplying factor of 10^5, assuming all ranks will be under 10^5.

Lab 11: Python and Django [1]

Contents

Lab 11: Python and Django [1]

Contents

Lab 11: Python and Django [1]

1. pdf to csv conversion

Contents

(i) Tried using pdfMiner, pdfTables, pdfpy etc, but all to no avail. Finally used pdftotext by import os to mine it.

1. pdf to csv conversion

Contents

(i) Tried using pdfMiner, pdfTables, pdfpy etc, but all to no avail. Finally used pdftotext by import os to mine it.
(ii) update.py creates an organised csv file output.csv.

1. pdf to csv conversion

Contents

(i) Tried using pdfMiner, pdfTables, pdfpy etc, but all to no avail. Finally used pdftotext by import os to mine it.
(ii) update.py creates an organised csv file output.csv.
(iii) Sorted the courses of output.csv on the basis of their codes.

1. pdf to csv conversion

Contents

(i) Tried using pdfMiner, pdfTables, pdfpy etc, but all to no avail. Finally used pdftotext by import os to mine it.
(ii) update.py creates an organised csv file output.csv.
(iii) Sorted the courses of output.csv on the basis of their codes.
(iv) Merged the two csv files which in turn was to be used for populating the sqlite database of our application.

2. Populating the database

Contents

(i) Populated the sqlite database, using python shell in the django [1] environment by p̈ython3 manage.py shell.̈

2. Populating the database

Contents

(i) Populated the sqlite database, using python shell in the django [1] environment by ̈python3 manage.py shell ̈.
(ii) Ran the commands in file populate_db in the python shell (manually).

3. Web framework to display a list of elegible courses

Contents

(i) Made models [3] for courses and UserProfile class which contained the attributes of the classes (name, id etc) along with their types (models.py [3]).

3. Web framework to display a list of elegible courses

Contents

(i) Made models [3] for courses and UserProfile class which contained the attributes of the classes (name, id etc) along with their types (models.py [3]).
(ii) Added our django [1] app jeeinterface in the mysite/urls.py file to create urls such as jeeinterface/login.

3. Web framework to display a list of elegible courses

Contents

(i) Made models [3] for courses and UserProfile class which contained the attributes of the classes (name, id etc) along with their types (models.py [3]).
(ii) Added our django [1] app jeeinterface in the mysite/urls.py file to create urls such as jeeinterface/login.
(iii) Implemented the user register-login system using the built-in django-forms [1] .

3. Web framework to display a list of elegible courses

Contents

(i) Made models [3] for courses and UserProfile class which contained the attributes of the classes (name, id etc) along with their types (models.py [3]).
(ii) Added our django [1] app jeeinterface in the mysite/urls.py file to create urls such as jeeinterface/login.
(iii) Implemented the user register-login system using the built-in django-forms [1] .
(iv) The rank of the logged user is used to find the courses one is elegible for.

3. Web framework to display a list of elegible courses

Contents

(i) Made models [3] for courses and UserProfile class which contained the attributes of the classes (name, id etc) along with their types (models.py [3]).
(ii) Added our django [1] app jeeinterface in the mysite/urls.py file to create urls such as jeeinterface/login.
(iii) Implemented the user register-login system using the built-in django-forms [1] .
(iv) The rank of the logged user is used to find the courses one is elegible for.
(v) Courses that the user is elegible for, are displayed via college-wise collapsable lists.

Conclusions

This is our submission for the CS251 project. Hope you enjoyed going through the project report as much as we enjoyed making it. Any other reviews and/or suggetions are most welcome. Looking forward to hearing from you.

References

- "Django tutorials."
  http://www.tangowithdjango.com/book17/.
  Accessed: 2014-10-30.
- "Discussion forum."
  http://stackoverflow.com/questions/19027473/writing-a-csv-file-using-buffered-writer-in-java.
  Accessed: 2014-10-30.
- "Django documentation."
  https://docs.djangoproject.com/en/1.7/.
  Accessed: 2014-10-30.