

Bekezhan Abdykarimov

2020297

1) Reading the Spotify Tracks dataset.
(By default it sorted by Track popularity, on the next slide we gonna sort it by danceability)

Hello Bekezhon Here you can see our Tracks.

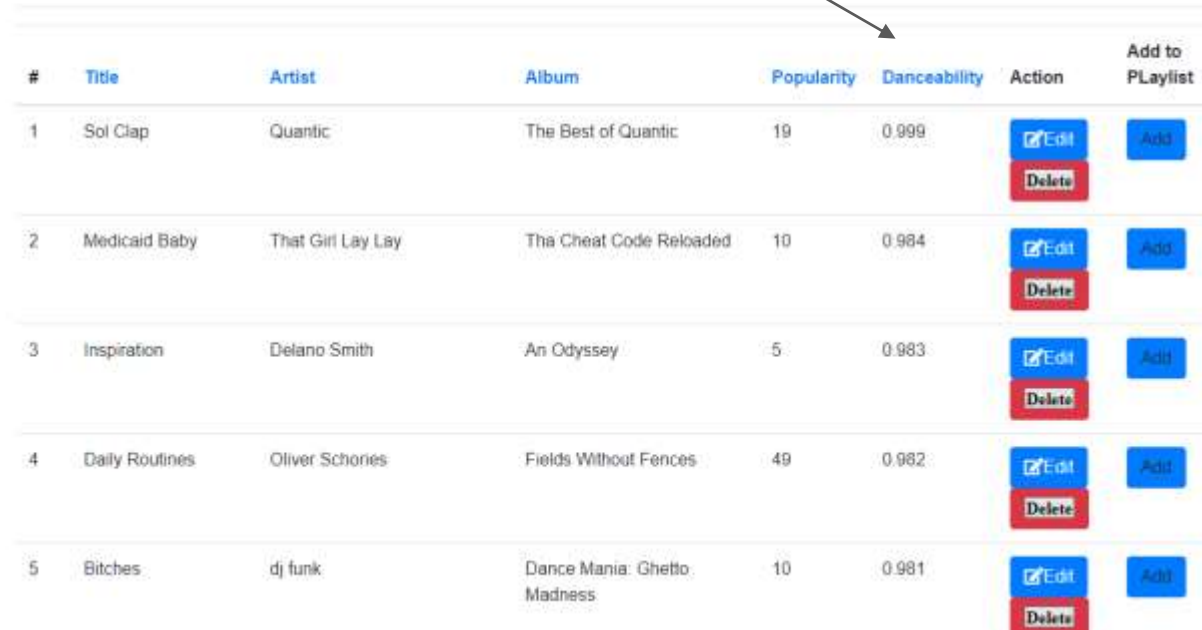
HOME CREATE NEW USER FILTER USERS ALL REGISTERED USERS

HOT 100

#	Title	Artist	Album	Popularity	Danceability	Action	Add to Playlist
1	I'm Good (Blue)	David Guetta, Bebe Rexha	I'm Good (Blue)	100	0.561	Edit Delete	Add
2	La Bachata	Manuel Turizo	La Bachata	98	0.835	Edit Delete	Add
3	Me Porto Bongo	Bad Bunny, Chencho Corleone	Un Verano Sin Ti	97	0.911	Edit Delete	Add
4	Tití Me Preguntó	Bad Bunny	Un Verano Sin Ti	97	0.65	Edit Delete	Add
5	Efecto	Bad Bunny	Un Verano Sin Ti	96	0.801	Edit Delete	Add

2)Now it sorted by Danceability:

HOT 100



#	Title	Artist	Album	Popularity	Danceability	Action	Add to Playlist
1	Sol Clap	Quantic	The Best of Quantic	19	0.999	Edit Delete	Add
2	Medicaid Baby	That Girl Lay Lay	Tha Cheat Code Reloaded	10	0.984	Edit Delete	Add
3	Inspiration	Delano Smith	An Odyssey	5	0.983	Edit Delete	Add
4	Daily Routines	Oliver Schories	Fields Without Fences	49	0.982	Edit Delete	Add
5	Bitches	dj funk	Dance Mania: Ghetto Madness	10	0.981	Edit Delete	Add

3) Edit Track Page, where user can change track's data, it has validation for all fields, so it won't be possible to leave it blank or put a String in the Popularity section for example

Edit Track

Artist:

Album Name:

Track name:

Danceability:

Popularity:

Edit Track

Artist:

Album Name:

Track name:

Danceability:

Popularity:

4) After updating the file user will be redirected to the same page he was before, and now we can see our changes :

HOT 100

#	Title	Artist	Album	Popularity	Danceability	Action	Add to PPlaylist
1	I'm Good (Blue)	David Guetta, Bebe Rexha	I'm Good (Blue)	100	0.561	Edit Delete	Add
2	Sol Clap	Quantic	The Best of Quantic	99	0.999	Edit Delete	Add
3	La Bachata	Manuel Turizo	La Bachata	98	0.835	Edit Delete	Add
4	Tití Me Preguntó	Bad Bunny	Un Verano Sin Ti	97	0.65	Edit Delete	Add
5	Me Porto Bonito	Bad Bunny, Jhay Cortez	Un Verano Sin Ti	97	0.911	Edit Delete	Add

5) Navigation buttons that helps user to navigate through the dataset, plus makes it easier and faster to load the data from the Database

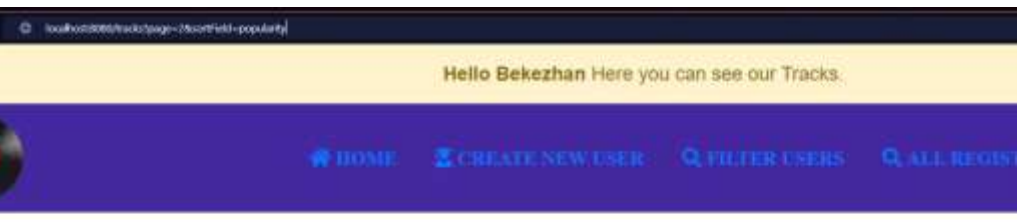
99	Mi Gente (feat. Beyoncé)	J Balvin; Willy William; Beyoncé
100	Know me	GEMINI

<< < 1 >

6) On this slide we have deleted first 2 songs, so now song#3 will become #1:

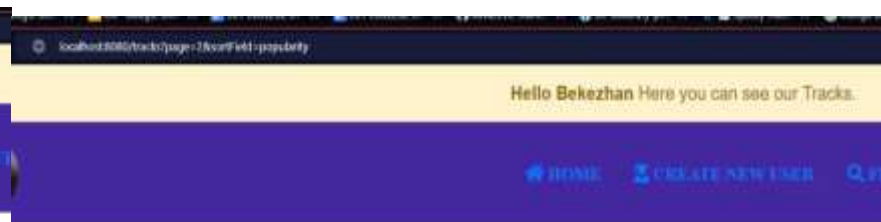
Before:

After:



HOT 100

#	Title	Artist	Album	Popularity	Dance
1	Clarks	Vybz Kartel;Popcaan;Gaza Slim	Stronger We Get	56	0.63
2	Fever	Wizkid	Fever	56	0.75
3	I Feel Love	Donna Summer	I Remember Yesterday	56	0.67



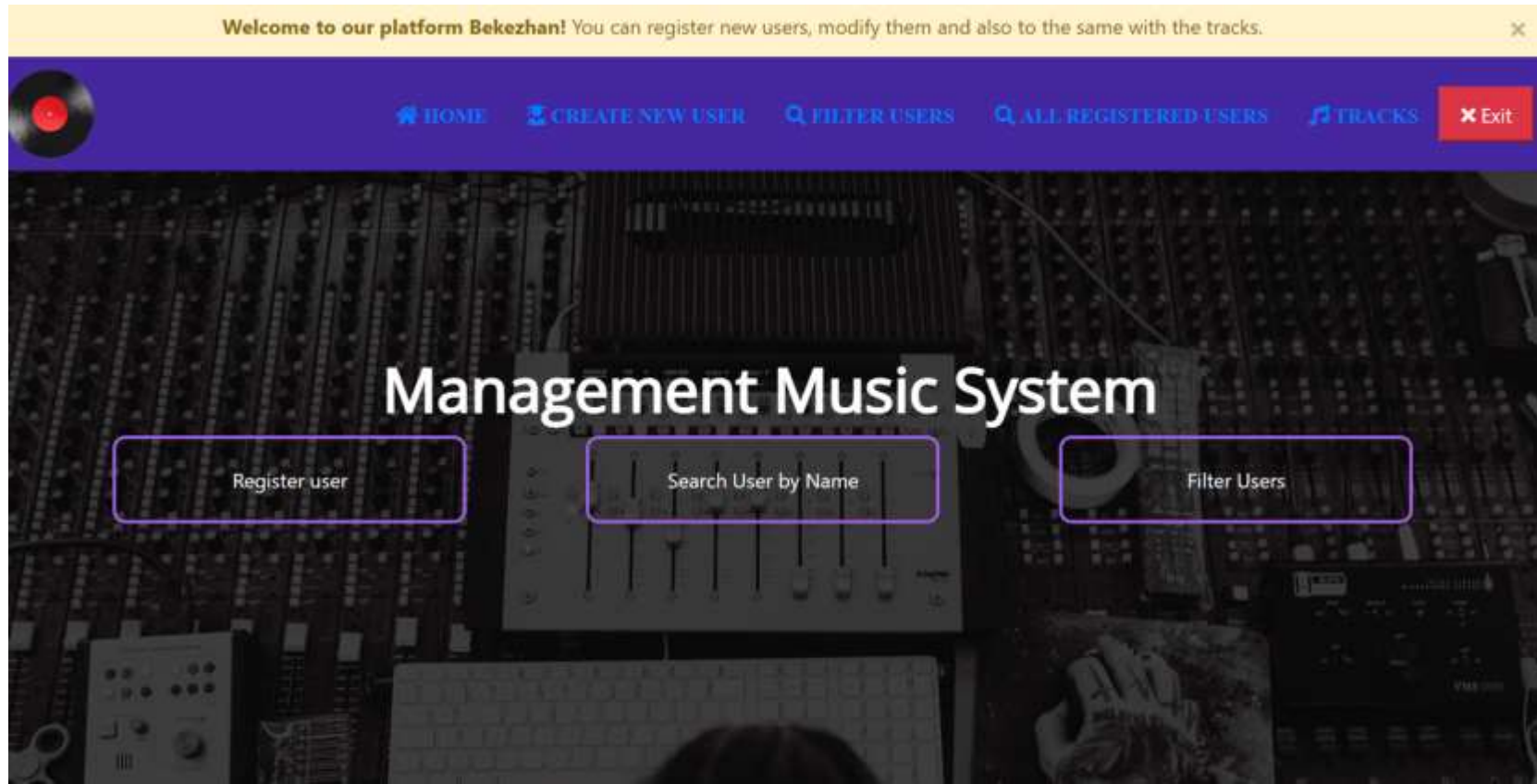
HOT 100

#	Title	Artist	Album	Popularity
1	I Feel Love	Donna Summer	I Remember Yesterday	56
2	Kiss the Girl - From "The Little Mermaid" Soundtrack Version	Samuel E. Wright;Disney	The Little Mermaid	56
3	Green Utopia	The Vines	Wicked Nature	56

Jefferson De Oliveira Lima

2020373

1) Home page:



2) User Registration Form:

+ Register User

+

+

User Name:

Ex: James

User Username:

Ex: Bond

Password:

Ex: Password

Email:

Ex: jamesbond@gmail.com

Musical Genre:

ROCK




SAVE

BACK


3) User Search page:

Here Users can search for the other users by their name, or favourite music genre:


+ Search Users




Users who likes Rock
Search for all user who has rock as preference.
[Click here!](#)




Users who likes classic
Search for all user who has classic as preference.
[Click here!](#)




Users who likes pop
Search for all user who has pop as preference.
[Click here!](#)



Users who likes rap
Search for all user who has rap as preference.
[Click here!](#)



Users who likes reggae
Search for all user who has reggae as preference.
[Click here!](#)



all users
See all users
[Click here!](#)

4) Registered Users Page, where we can perform CRUD operations on them:

[HOME](#) [CREATE NEW USER](#) [FILTER USERS](#) [ALL REGISTERED](#)

Registered Users

Name	Username	Password(Encrypted)	Email	Course	Action
john	travis	202cb962ac59075b964b07152d234b70	123@123.com	ROCK	Edit Delete
CCT	CCT	827ccb0eea8a706c4c34a16891f84e7b	cct@cct.cct	ROCK	Edit Delete
Bekezhan	Beka	f71c73d5833e1ea59986f3262fa591f	by.bekardi@gmail.com	ROCK	Edit Delete

5) Mongo DB

Created a collection “users” to store all the registered users:

The screenshot displays the MongoDB Atlas web interface. On the left sidebar, the 'SongsAPI' project is selected, and the 'users' collection within the 'songs-api-db' database is highlighted. The main panel shows the 'songs-api-db.users' collection with the 'Documents' tab active. Below the tabs, there is a search bar and a 'Filter' button. The document list shows three entries, each with a JSON representation of a user document. The first document is for 'James Bond', the second for 'Bob Marley', and the third for 'Justin Bieber'. Each document includes fields for _id, name, surname, password, email, musicalGenre, and _class.

```
{
  "_id": "7673d276-8e03-4eba-9ddd-079bfe1cac9f",
  "name": "James",
  "surname": "Bond",
  "password": "",
  "email": "james@bond.com",
  "musicalGenre": "CLASSIC",
  "_class": "com.integratedAssessment.cct.model.User"
}
```

```
{
  "_id": "ba177913-fda2-4193-8574-e91b2cdea856",
  "name": "Bob",
  "surname": "Marley",
  "userNumber": "3732020",
  "email": "anything@something",
  "musicalGenre": "REGGAE",
  "_class": "com.integratedAssessment.cct.model.User"
}
```

```
{
  "_id": "e61224f8-4f6e-488d-9e2a-a73fbf2fca33",
  "name": "Justin",
  "surname": "Bieber",
  "userNumber": "3732020",
  "email": "anything@something.com",
  "musicalGenre": "POP",
  "_class": "com.integratedAssessment.cct.model.User"
}
```

Daniel Bezerra Martellini

2020356

1- imported CSV dataset to Hadoop

```
hduser@integratedCA:~$ hadoop fs -put ./dataset.csv /user1
```

2- Executed MapReduce

```
hduser@integratedCA:~/Desktop/Hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.4.jar -mapper ./mymapper.py -reducer ./myreducer.py -input /user1/datasetupdated.csv -output /out111
2023-05-06 01:16:28,458 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-05-06 01:16:28,543 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
```

3- Mapper.py

```
*mymapper.py
~/Desktop/Hadoop

#!/usr/bin/env python3
import sys

# which column want to map and reduce
column_index = 21
delimiter = ";"

for line in sys.stdin:
    # Split the line
    data = line.strip().split(delimiter)

    # If the column has data
    if len(data) > column_index:
        value = data[column_index].strip()

        print(f"{value}\t1")
```

4- reducer.py

```
myreducer.py
~/Desktop/Hadoop

#!/usr/bin/env python3
import sys

current_value = None
current_count = 0

for line in sys.stdin:
    line = line.strip()
    if '\t' in line:
        value, count = line.split('\t')
    elif ';' in line:
        value, count = line.split(';')
    else:
        continue

    if current_value != value:
        if current_value is not None:
            print("%s\t%d" % (current_value, current_count))
        current_value = value
        current_count = 0

    current_count += int(count)

if current_value is not None:
    print("%s\t%d" % (current_value, current_count))
```

5- Part of my MapReduce

output:

Zumbis no Espaço	1
Zuris	1
Zuukou mayzie	1
Zuukou mayzie;Tom Hardy	1
Zuukou mayzie;Vickie Cherie	1
Zvláštný škola	2
Zwette;Tom Rosenthal	1
Zyon	1
Zyrtck;Radical;Mothz	1
Zé Bigode Orquestra	2
Zé Bigode Orquestra;Guinu	1
Zé Cantor;Avine Vinny	1
Zé Cantor;Luan Estilizado	1
Zé Cantor;Rai Saia Rodada	2
Zé Felipe & Miguel;Paraná	1
Zé Henrique & Gabriel	5
Zé Henrique & Gabriel;Marília Mendonça	1
Zé Ketí	1
Zé Ramalho	25
Zé Ramalho;Andreas Kisser	1
Zé Ramalho;Elba Ramalho	1
Zé Ramalho;Geraldo Azevedo	1
Zé Ramalho;Paulinho Moska	2
Zé Ricardo & Thiago	1
Zé Vaqueiro	20
Zé Vaqueiro Estilizado	3
Zé Vaqueiro;Dilsinho	1
Zé Vaqueiro;Vitor Fernandes	1
Zélia Duncan	7

1- Importing my CSV file from Hadoop and creating a database in mongoDB with it:

Daniel Bezerra Martellini

```
hduser@integratedCA:~$ hadoop fs -cat /user1/dataset.csv | mongoimport --db mydb --collection mycollection --type csv --headerline
2023-05-06T00:24:30.599+0100    connected to: mongodb://localhost/
2023-05-06T00:24:33.599+0100    mydb.mycollection            9.18MB
2023-05-06T00:24:36.600+0100    mydb.mycollection            19.2MB
2023-05-06T00:24:36.686+0100    mydb.mycollection            19.2MB
2023-05-06T00:24:36.686+0100    114000 document(s) imported successfully. 0 document(s) failed to import.
```

2- Successfully create database with desired parameters:

```
> db.mycollection.find().pretty()
{
  "_id" : ObjectId("6455902e05219e80871f2944"),
  "n" : 0,
  "track_id" : "5Su0ikwiRyPMVoIQDJUgSV",
  "artists" : "Gen Hoshino",
  "album_name" : "Comedy",
  "track_name" : "Comedy",
  "popularity" : 73,
  "duration_ms" : 230666,
  "explicit" : "False",
  "danceability" : 0.676,
  "energy" : 0.461,
  "key" : 1,
  "loudness" : -6.746,
  "mode" : 0,
  "speechiness" : 0.143,
  "acousticness" : 0.0322,
  "instrumentalness" : 0.00000101,
  "liveness" : 0.358,
  "valence" : 0.715,
  "tempo" : 87.917,
  "time_signature" : 4,
  "track_genre" : "acoustic"
}
```


1- Tested MongoDB with preloaded workload, then edited preloadb to test it with 10,000 rows instead of 1,000

```
Command line: -load -db site.ycsb.db.MongoDbClient -s -P workloads/workloadA
YCSB client 0.17.0
```

```
Loading workload...
```

```
Starting test.
```

```
2023-05-06 01:55:48:078 0 sec: 0 operations; est completion in 0 second
mongo client connection created with mongodbd://localhost:27017/ycsb?w=1
DBWrapper: report latency for each error is false and specific error codes
are: []
```

```
2023-05-06 01:55:50:126 2 sec: 10000 operations; 4816.96 current ops/sec;
=2211, Min=2210, Avg=2211, 90=2211, 99=2211, 99.9=2211, 99.99=2211] [INSERT:
4063, Min=85, Avg=166.88, 90=255, 99=608, 99.9=4071, 99.99=14135]
```

```
[OVERALL], RunTime(ms), 2077
```

```
[OVERALL], Throughput(ops/sec), 4814.6364949446315
```

```
[TOTAL_GC_PS_Scavenge], Count, 5
```

```
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 20
```

```
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 0.9629272989889264
```

```
[TOTAL_GC_PS_MarkSweep], Count, 0
```

```
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
```

```
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
```

```
[TOTAL_GCs], Count, 5
```

```
[TOTAL_GC_TIME], Time(ms), 20
```

```
[TOTAL_GC_TIME_%], Time(%), 0.9629272989889264
```

```
[CLEANUP], Operations, 1
```

```
[CLEANUP], AverageLatency(us), 2211.0
```

```
[CLEANUP], MinLatency(us), 2210
```

```
[CLEANUP], MaxLatency(us), 2211
```

```
[CLEANUP], 95thPercentileLatency(us), 2211
```

```
[CLEANUP], 99thPercentileLatency(us), 2211
```

```
[INSERT], Operations, 10000
```

```
[INSERT], AverageLatency(us), 166.8767
```

```
[INSERT], MinLatency(us), 85
```

```
[INSERT], MaxLatency(us), 104063
```

```
[INSERT], 95thPercentileLatency(us), 338
```

```
[INSERT], 99thPercentileLatency(us), 608
```

```
[INSERT], Return=OK, 10000
```

```
hduser@integratedCA:~/ycsb-0.17.0$
```

```
YCSB client 0.17.0
```

```
# Copyright (c) 2010 Yahoo! Inc. All rights reserved.
```

```
#
```

```
# Licensed under the Apache License, Version 2.0 (the "License"); you
```

```
# may not use this file except in compliance with the License. You
```

```
# may obtain a copy of the License at
```

```
#
```

```
# http://www.apache.org/licenses/LICENSE-2.0
```

```
#
```

```
# Unless required by applicable law or agreed to in writing, software
```

```
# distributed under the License is distributed on an "AS IS" BASIS,
```

```
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
```

```
# implied. See the License for the specific language governing
```

```
# permissions and limitations under the License. See accompanying
```

```
# LICENSE file.
```

```
#
```

```
# Yahoo! Cloud System Benchmark
```

```
# Workload A: Update heavy workload
```

```
# Application example: Session store recording recent actions
```

```
#
```

```
# Read/update ratio: 50/50
```

```
# Default data size: 1 KB records (10 fields, 100 bytes each, plus key)
```

```
# Request distribution: zipfian
```

```
#
```

```
recordcount=10000
```

```
operationcount=1000
```

```
workload=site.ycsb.workloads.CoreWorkload
```

```
#
```

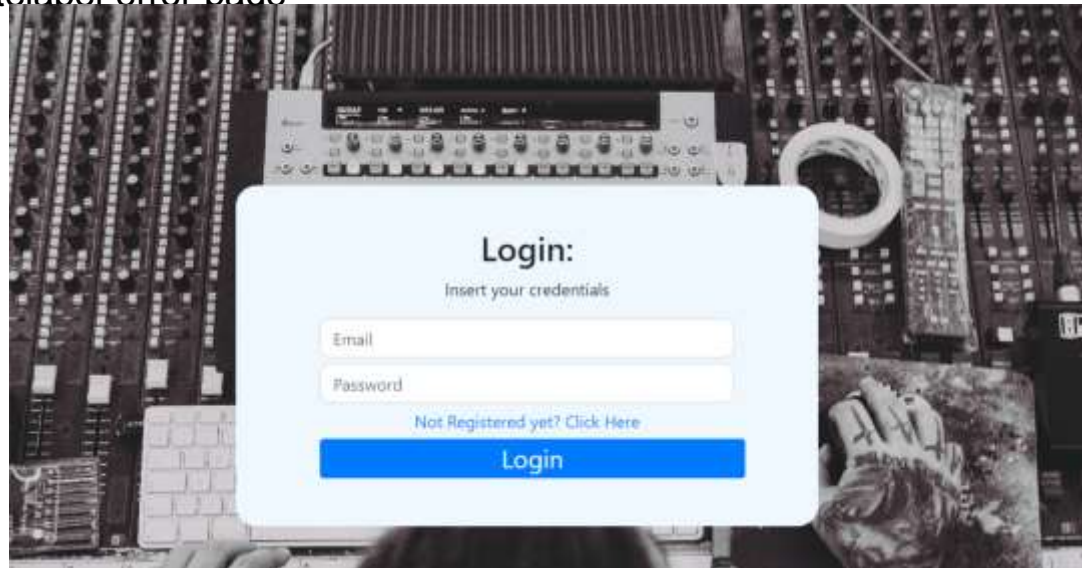
```
readallfields=true
```

```
#
```

```
readproportion=0.5
```

```
updateproportion=0.5
```

1- Created the whole login system, user service class, handled exceptions and customized whitelabel error page



1- YCSB in MongoDB for 100,000 record, compared to time took on MySQL (Right Figure), mongoDB was significantly faster.

```

GNU nano 6.2      output-workloadB100000
/usr/bin/java -classpath /home/hduser/ycsb-0.17.0/conf:/home
mongo client connection created with mongodb://localhost:270
[OVERALL], RunTime(ms), 19559
[OVERALL], Throughput(ops/sec), 5112.735824939926
[TOTAL_GCS_PS_Scavenge], Count, 11
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 196
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 1.0020962216882252
[TOTAL_GCS_PS_MarkSweep], Count, 0
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
[TOTAL_GCs], Count, 11
[TOTAL_GC_TIME], Time(ms), 196
[TOTAL_GC_TIME_%], Time(%), 1.0020962216882252
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 15788.0
[CLEANUP], MinLatency(us), 15784
[CLEANUP], MaxLatency(us), 15791
[CLEANUP], 95thPercentileLatency(us), 15791
[CLEANUP], 99thPercentileLatency(us), 15791
[INSERT], Operations, 100000
[ Read 26 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execu
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justi
connecting to: test

```

```

GNU nano 6.2      output_SQL.txt
/usr/bin/java -classpath /home/hduser/ycsb-0.17.0/conf:/home
Adding shard node URL: jdbc:mysql://localhost:3306/BenchTest
Using shards: 1, batchSize:-1, fetchSize:-1
[OVERALL], RunTime(ms), 763692
[OVERALL], Throughput(ops/sec), 130.9428408311204
[TOTAL_GCS_PS_Scavenge], Count, 124
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 202
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 0.026450453847886322
[TOTAL_GCS_PS_MarkSweep], Count, 0
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
[TOTAL_GCs], Count, 124
[TOTAL_GC_TIME], Time(ms), 202
[TOTAL_GC_TIME_%], Time(%), 0.026450453847886322
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 2331.0
[CLEANUP], MinLatency(us), 2330
[CLEANUP], MaxLatency(us), 2331
[CLEANUP], 95thPercentileLatency(us), 2331
[CLEANUP], 99thPercentileLatency(us), 2331
[INSERT], Operations, 100000
[INSERT], AverageLatency(us), 7621.20515
[INSERT], MinLatency(us), 2920
[INSERT], MaxLatency(us), 156415
[INSERT], 95thPercentileLatency(us), 13015
[INSERT], 99thPercentileLatency(us), 18655
[INSERT], Return=OK, 100000

```