Task 1

- How did you use connection pooling?

Line 3-6 in the picture below is about connection pooling in **context.xml**
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/META-INF/context.xml

```
1  <Context>
2
3      <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
4          maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
5          password="wei123456" driverClassName="com.mysql.jdbc.Driver"
6          url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false" />
7
8          <Resource name="jdbc/insert" auth="Container" type="javax.sql.DataSource"
9          maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
10         password="wei123456" driverClassName="com.mysql.jdbc.Driver"
11         url="jdbc:mysql://172.31.30.134:3306/moviedb?autoReconnect=true&amp;useSSL=false" />
12
13 </Context>
14
```

Create the first resource, which use the localhost on the instance. Therefore, read operation will go into either master instance or slave instance.

Line 13-27 in the picture below **web.xml**
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/WEB-INF/web.xml

```
13   <resource-ref>
14     <description>
15             Resource reference to a factory for java.sql.Connection
16             instances that may be used for talking to a particular
17             database that
18             is configured in the server.xml file.
19         </description>
20     <res-ref-name>
21             jdbc/moviedb
22         </res-ref-name>
23     <res-type>
24             javax.sql.DataSource
25         </res-type>
26     <res-auth>Container</res-auth>
27   </resource-ref>
28
29   <resource-ref>
30     <description>
31             Resource reference to a factory for java.sql.Connection
32             instances that may be used for talking to a particular
33             database that
34             is configured in the server.xml file.
35         </description>
36     <res-ref-name>
37             jdbc/insert
38         </res-ref-name>
39     <res-type>
40             javax.sql.DataSource
41         </res-type>
42     <res-auth>Container</res-auth>
43   </resource-ref>
44
```

In the following file, change how to connect the database, using connection pooling now:
In **AdvancedSearch.java**: line 64 - line 85
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AdvancedSearch.java
In **AutoComplete.java**: line 63 - line 84
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AutoComplete.java
In **BrowseByGenre.java**: line 64 - line 84
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/BrowseByGenre.java
In **BrowseByTitle.java**: line 56 - line 77
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/BrowseByTitle.java
In **CheckOut.java**: line 141 - line 162
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/CheckOut.java
In **EmployeeLogin.java**: line 83 - line 104
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/EmployeeLogin.java
In **Login.java**: line 84 - line 105
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Login.java
In **Metadata.java**: line 58 - line 79
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Metadata.java
In **Search.java**: line 51 - line 74
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Search.java
In **SingleMovie.java**: line 64 - line 84
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/SingleMovie.java
In **SingleStar.java**: line 62 - line 83
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/SingleStar.java

- Snapshots

```
        long startTime1 = System.nanoTime();
    // the following few lines are for connection pooling
        // Obtain our environment naming context

        Context initCtx = new InitialContext();
        if (initCtx == null)
            out.println("initCtx is NULL");

        Context envCtx = (Context) initCtx.lookup("java:comp/env");
        if (envCtx == null)
            out.println("envCtx is NULL");

        // Look up our data source
        DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");

        // the following commented lines are direct connections without pooling
        //Class.forName("org.gjt.mm.mysql.Driver");
        //Class.forName("com.mysql.jdbc.Driver").newInstance();
        //Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);

        if (ds == null)
            out.println("ds is null.");

        Connection dbcon = ds.getConnection();
        if (dbcon == null)
            out.println("dbcon is null.");
/
/         //Class.forName("org.gjt.mm.mysql.Driver");
/         Class.forName("com.mysql.jdbc.Driver").newInstance();
/
/         Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
        // Declare our statement
```

- How did you use Prepared Statements?

In **Search.java** (where I use fulltext search): line 99-106, using “?” in query
In line 133, create PreparedStatement
In line 135 - 139 put value into PreparedStatement
In line 147, run PreparedStatement
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Search.java

```
98
99              for (int i=0; i<splited.length; i++)
100             {
101              total_input = total_input + "MATCH (title) AGAINST (? IN BOOLEAN MODE) ";
102              if (i != splited.length-1)
103                    {
104                        total_input+="AND ";
105                    }
106             }
107
108
109             System.out.println(total_input);
110
111
112
113 //              String query = ""
114 //                  + "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ') AS stars, GROUP_CONCAT(DISTINCT genres.name ORDEI
115 //                      "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
116 //                      "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND ratings.movieId=mov
117 //                      "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
118 //                      "ORDER BY ratings.rating DESC;" ;
119 //
120
121             String query = ""
122                   + "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ') AS stars, GROUP_CONCAT(DISTINCT genres.name ORDEI
123                       "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
124                       "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND ratings.movieId=mov
125                       "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
126                       "ORDER BY ratings.rating DESC;" ;
127
128
129             System.out.println("query111111 = "+query);
130
131             //movies.title LIKE '%home%'
132             //movies.title LIKE '%"+input+"%'\n" +
133             PreparedStatement pstmt = dbcon.prepareStatement( query );
134             System.out.println("after preparement statement = "+query);
135             for (int i=1; i<=splited.length; i++)
136             {
137                pstmt.setString( i,splited[i-1]+"*");
138
139             }
140
141             System.out.println("query222222 = "+query);
142
143
144             long startTime2 = System.nanoTime();
145
146             // Perform the query
147             ResultSet rs = pstmt.executeQuery();
148
149             long endTime2 = System.nanoTime();
150
151
152             JsonArray jsonArray = new JsonArray();
153             while (rs.next())
154             {
155                 String movie_id = rs.getString(1);
156                 String movie_title = rs.getString(2);
157                 int movie_year = rs.getInt(3);
158                 String movie_director = rs.getString(4);
159                 String star_name = rs.getString(5);
160                 String genre_type = rs.getString(6);
```

In **AdvancedSearch.java**: line 136-135, using “?” in query
In line 154, create PreparedStatement
In line 157 - 178 put value into PreparedStatement
In line 185, run PreparedStatement
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AdvancedSearch.java

```
115             String total_input =  "";
116             if (title.length()>0)
117             {
118                 total_input += "AND movies.title LIKE ? ";
119                 hm.put("title", title);
120             }
121             if (year.length()>0)
122             {
123                 total_input += "AND movies.year=? ";
124                 hm.put("year", year);
125             }
126             if (director.length()>0)
127             {
128                 total_input += "AND movies.director LIKE ? ";
129                 hm.put("director", director);
130             }
131             if (input_star_name.length()>0)
132             {
133                 total_input += "AND stars.name LIKE ? ";
134                 hm.put("input_star_name", input_star_name);
135             }
136
137
138             System.out.println(" total input !   "+total_input);
139
140
141             System.out.println("before query  ");
142             //genre = "drama";
143             String query = "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ') AS stars, GROUP_CONCAT(DISTINCT genres.name ORDER BY genres.name SEPARATOR ', ') AS genres, ratings.rating\n" +
144                     "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
145                     "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND ratings.movieId=movies.id  "+ total_input +"\n" +
146                     "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
147                     "ORDER BY ratings.rating DESC;" ;//+
148                     //"LIMIT 100;";
149
150             System.out.println("parameterrrrrrr  "+title);
151             System.out.println("go after query  " + query);
152             //total_input[1]
153             // Perform the query
154             PreparedStatement pstmt = dbcon.prepareStatement( query );
155             //pstmt.setString( 1, MovieList.get(i).getTitle());
156
157             int index=1;
158
159             if(hm.get("title")!=null)
160             {
161                 pstmt.setString( index, "%"+title+"%");
162                 index++;
163             }
164             if(hm.get("year")!=null)
165             {
166                 pstmt.setString( index, year);
167                 index++;
168             }
169             if(hm.get("director")!=null)
170             {
171                 pstmt.setString( index, "%"+director+"%");
172                 index++;
173             }
174             if(hm.get("input_star_name")!=null)
175             {
176                 pstmt.setString( index, "%"+input_star_name+"%");
177                 index++;
178             }
179
180
181
182
183
184             long startTime2 = System.nanoTime();
185             ResultSet rs = pstmt.executeQuery();
186             long endTime2 = System.nanoTime();
187
188             JsonArray jsonArray = new JsonArray();
189
```

In **AutoComplete.java**:
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AutoComplete.java

For **search Movie title part**:
line 118-125, using “?” in query
In line 149, create PreparedStatement
In line 151 - 155 put value into PreparedStatement
In line 162, run PreparedStatement

```
115             String total_input="";
116
117
118             for (int i=0; i<splited.length; i++)
119             {
120             total_input = total_input + "MATCH (title) AGAINST ( ? IN BOOLEAN MODE) ";
121             if (i != splited.length-1)
122                     {
123                         total_input+="AND ";
124                     }
125             }
126
127
128             System.out.println(total_input);
129
130
131
132             String query = ""
133                     + "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ') AS stars, GROUP_CONCAT(DIS
134                     "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
135                     "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND r
136                     "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
137                     "ORDER BY ratings.rating DESC\n" +
138                     "LIMIT 5;";
139
140
141
142             System.out.println("query = "+query);
143
144             //movies.title LIKE '%home%'
145             //movies.title LIKE '%"+input+"%'\n" +
146
147             // Perform the query
148
149             PreparedStatement pstmt = dbcon.prepareStatement( query );
150
151             for (int i=1; i<=splited.length; i++)
152             {
153               pstmt.setString( i,splited[i-1]+"*" );
154
155             }
156
157
158
159
160             // Perform the query
161             long startTime2 = System.nanoTime();
162             ResultSet rs = pstmt.executeQuery();
163             long endTime2 = System.nanoTime();
```

For **search Star name part**:
line 205-209, using “?” in query
In line 215, create PreparedStatement
In line 217 - 221 put value into PreparedStatement
In line 256, run PreparedStatement

```java
String query2 = ""
      + "SELECT name\n" +
      "FROM stars\n" +
      "WHERE " + total_input2+"\n" +
      "LIMIT 5;";



System.out.println("query2 = "+query2);

PreparedStatement pstmt2 = dbcon.prepareStatement( query2 );

for (int i=1; i<=splited.length; i++)
{
  pstmt2.setString( i,splited[i-1]+"*" );

}

// Perform the query
long startTime3 = System.nanoTime();
rs = pstmt2.executeQuery();
long endTime3 = System.nanoTime();
```

Task 2

- Address of AWS and Google instances
  AWS Address: 18.219.36.92
  Google Address:

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?

I comment out the login filter portion, which makes me easier to go to my website every time, but you can check my sticky session using shopping cart.

- How connection pooling works with two backend SQL?

I create second resource in context.xml file, in this resource, it connect to master's instance database. Therefore, all the write operation will pass to master's database.
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/META-INF/context.xml

Line 8-11

```
1  <Context>
2
3      <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
4          maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
5          password="wei123456" driverClassName="com.mysql.jdbc.Driver"
6          url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false" />
7
8          <Resource name="jdbc/insert" auth="Container" type="javax.sql.DataSource"
9          maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
10         password="wei123456" driverClassName="com.mysql.jdbc.Driver"
11         url="jdbc:mysql://172.31.30.134:3306/moviedb?autoReconnect=true&amp;useSSL=false" />
12
13 </Context>
14
```

Also map the resource in web.xml below
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/WEB-INF/web.xml

```
13    <resource-ref>
14      <description>
15              Resource reference to a factory for java.sql.Connection
16              instances that may be used for talking to a particular
17              database that
18              is configured in the server.xml file.
19          </description>
20      <res-ref-name>
21              jdbc/moviedb
22          </res-ref-name>
23      <res-type>
24              javax.sql.DataSource
25          </res-type>
26      <res-auth>Container</res-auth>
27    </resource-ref>
28
29    <resource-ref>
30      <description>
31              Resource reference to a factory for java.sql.Connection
32              instances that may be used for talking to a particular
33              database that
34              is configured in the server.xml file.
35          </description>
36      <res-ref-name>
37              jdbc/insert
38          </res-ref-name>
39      <res-type>
40              javax.sql.DataSource
41          </res-type>
42      <res-auth>Container</res-auth>
43    </resource-ref>
44
```

- How read/write requests were routed?

In **InsertMovie.java** and **InsertStar.java**
In **InsertMovie.java**, it is line 65-81
In **InsertStar.java**, it is line 63-80

The difference between this one and the one in task 1 is in line 74, we now look up (jdbc/insert),which is the database that run writing operation.

```
61          try {
62          // the following few lines are for connection pooling
63              // Obtain our environment naming context
64
65              Context initCtx = new InitialContext();
66              if (initCtx == null)
67                  out.println("initCtx is NULL");
68
69              Context envCtx = (Context) initCtx.lookup("java:comp/env");
70              if (envCtx == null)
71                  out.println("envCtx is NULL");
72
73              // Look up our data source
74              DataSource ds = (DataSource) envCtx.lookup("jdbc/insert");
75
76              if (ds == null)
77                  out.println("ds is null.");
78
79              Connection dbcon = ds.getConnection();
80              if (dbcon == null)
81                  out.println("dbcon is null.");
82
```

Task 3

- Have you uploaded the **log file** to Github? Where is it located?

There are 12 log files for me, 5 for single-instance case, and 2*4-1 for scaled version, because sticky session makes one thread case always go to one instance:

Single-instance case:
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single1.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single2.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single3.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single4.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single5.txt

Scaled version case:
Case1: Use HTTP, without using prepared statements, 10 threads in JMeter.
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled1-master-real.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled1-slave-real.txt
Case2: Use HTTP, without using prepared statements, 10 threads in JMeter.
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled2-master.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled2-slave.txt

Case3: Use HTTP, 1 thread in JMeter.
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single3.txt

Case4: Use HTTP, 10 threads in JMeter.
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled4-master.txt
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled4-slave.txt

- Have you uploaded the **HTML file** to Github? Where is it located?

https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/jmeter_report.html

- Have you uploaded the **script** to Github? Where is it located?

https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/CalculateAverage.java

- Have you uploaded the WAR file and README to Github? Where is it located?

README is on the home page of our repository (at the bottom):
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17

The WAR file include the script is:
https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example.war