Milestone 2

Dominique Wehner
Kasey Powers
Sponsor: Dr. Gallagher

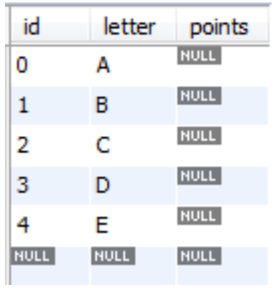
Milestone 2 matrix

Task	**Completion**	**Dominique**	**Kasey**	**To-do**
Setup SQL database on a local machine	100%	50%	50%	
Connect SQL database to a java class (This was a learning task to get familiar with connecting to the database via java code)	100%	100%		
Create a java servlet class	100%	100%		
Install Tomcat and implemented a java servlet	100%	100%		
Write a function to save inputs to xml	100%		100%	
Create system to display important info about saved xml data	75%		75%	Displays how many of each letter, still needs breakdown info
Update documents (test, requirements, & design) with database and servlet components	95%	50%	45%	Continue updating while we build out product

Summary of Milestone 2 tasks

1. Set up a SQL database named pentotext using SQL workbench on our local machines. Was a relatively easy task. We figured out how to export the sql file and import them into another database once we determine where we host the database.
2. Created a java servlet in eclipse and connected to the pentotext database successfully. The program connected to the databases location and outputted the table contents.
3. Created a java servlet class to implement with tomcat. Was also relatively easy after having already created a program that outputted the contents, we needed to extend the http class in the code and make a few adjustments to different methods.
4. Downloaded Tomcat successfully and attempted to implement the java servlet on it. We compile the Java servlet code and move it to the webapps/classes directory in tomcat, and ran startup.bat to start tomcat and accessed localhost:808/<servlet class name> in the web browser. We're getting a 505 ERROR though and so we needed to figure out what is wrong with the code, turns out it was a very simple error, we just forgot to drop in a jar file into tomcat. We successfully connected to our database through tomcat and displayed it as a webapp.
5. Rewrote many UI elements in current main window, making the clearing and submitting of data smoother process, added window to display breakdown of saved data
6. Wrote code to read and write XML, and data window shows count of letters inputed.
7. Work started on showing breakdown of algorithm results.
8. Updated document with a UML diagram and the database approach we decided to take this milestone.

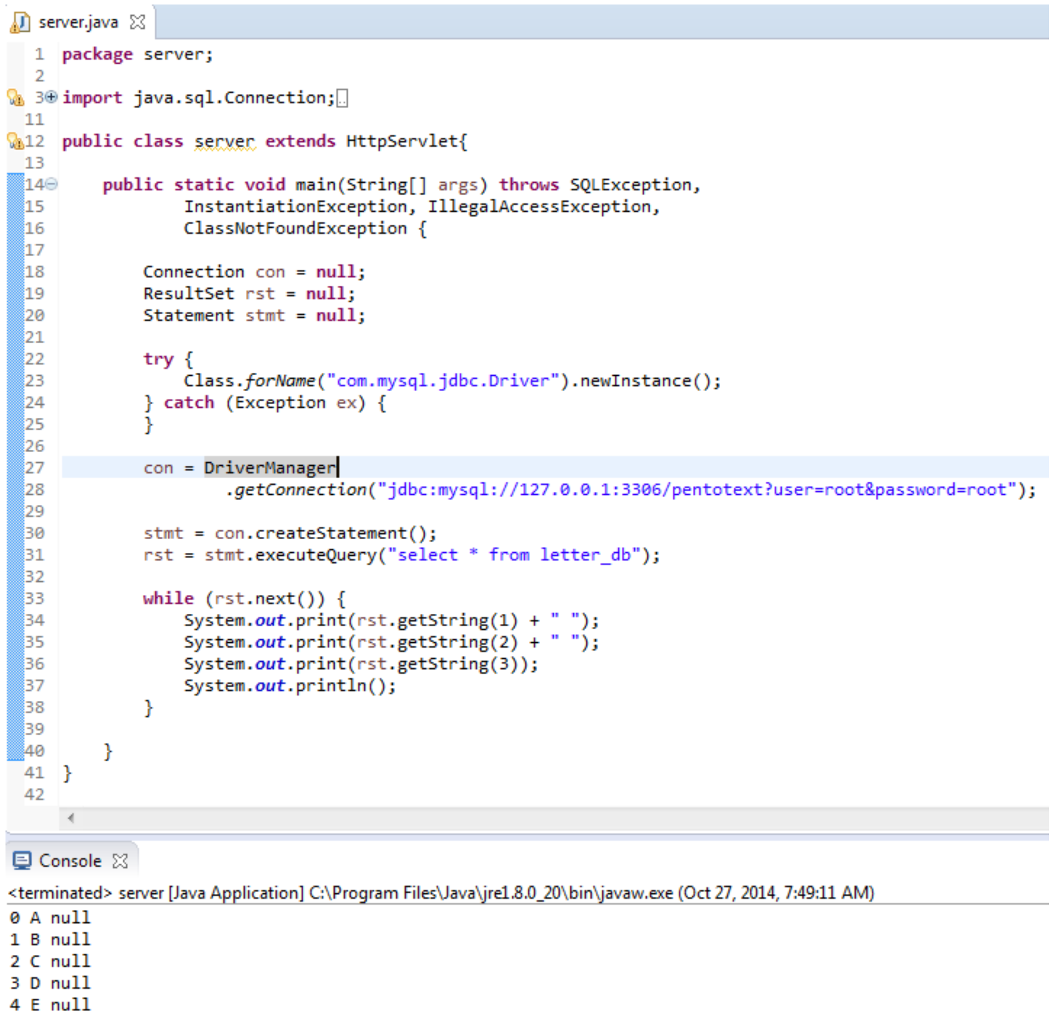
Database setup



id	letter	points
0	A	NULL
1	B	NULL
2	C	NULL
3	D	NULL
4	E	NULL
NULL	NULL	NULL

- The point of this database is to have something we can compare user input to.
- We will store the points of the letters we input into the database.
- Because of the variety of inputs for each letter, for example a lower case ‘a’ and ‘ɑ’, we will have a variety of different sets of points for each letter. So we may have a few rows for ‘a’, etc.
- The way we store our points may change, depending on how accurate and fast it is comparing user input to the input in the database

Java program that simply connects to the database and outputs the contents



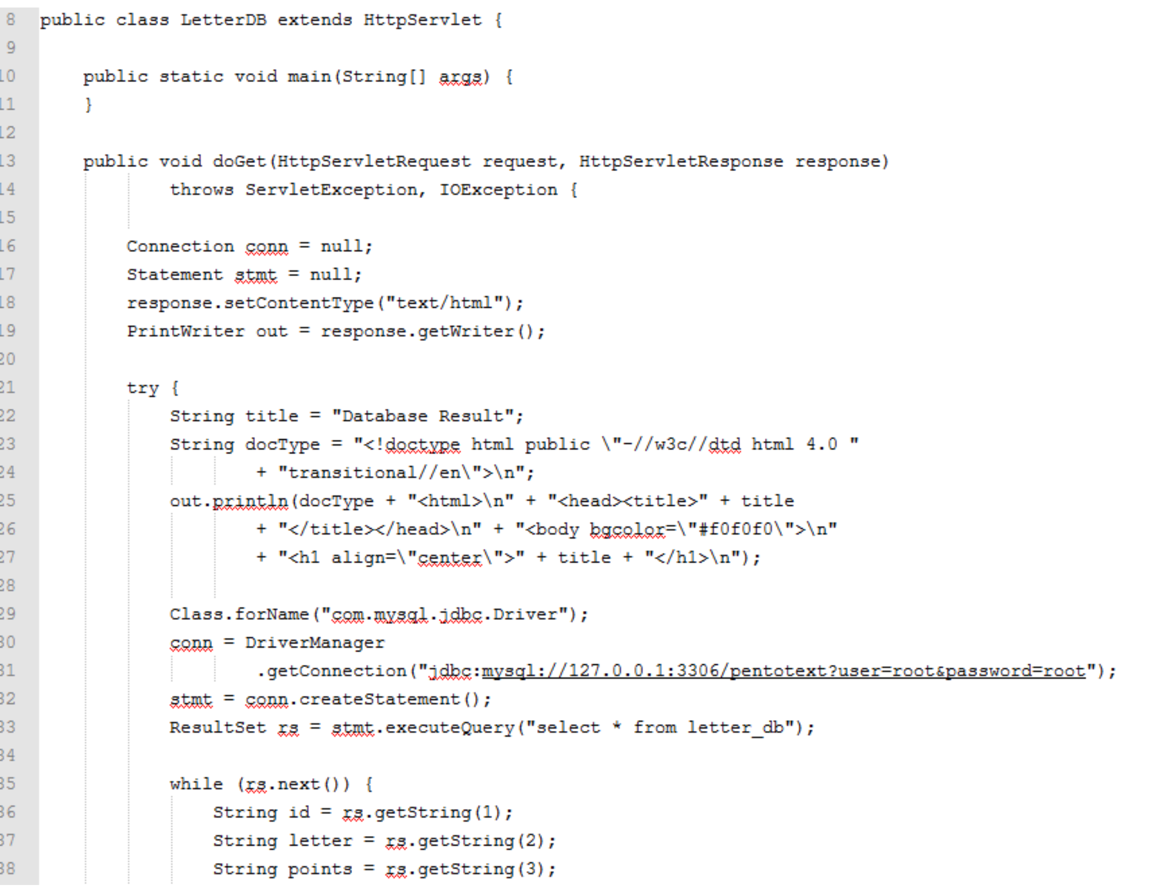
server.java

```java
package server;

import java.sql.Connection;

public class server extends HttpServlet{

    public static void main(String[] args) throws SQLException,
            InstantiationException, IllegalAccessException,
            ClassNotFoundException {

        Connection con = null;
        ResultSet rst = null;
        Statement stmt = null;

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception ex) {
        }

        con = DriverManager
                .getConnection("jdbc:mysql://127.0.0.1:3306/pentotext?user=root&password=root");

        stmt = con.createStatement();
        rst = stmt.executeQuery("select * from letter_db");

        while (rst.next()) {
            System.out.print(rst.getString(1) + " ");
            System.out.print(rst.getString(2) + " ");
            System.out.print(rst.getString(3));
            System.out.println();
        }

    }
}
```

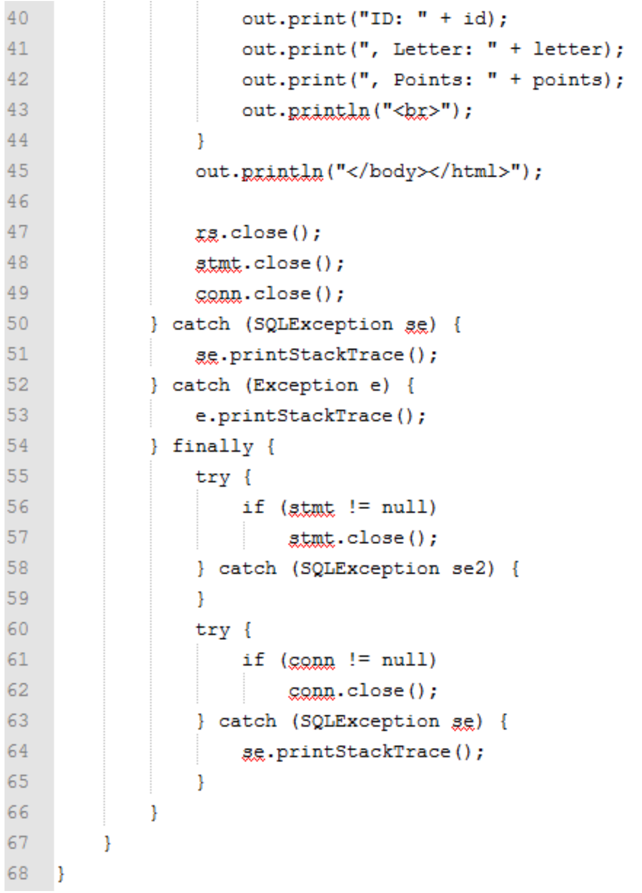
Console

```
<terminated> server [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Oct 27, 2014, 7:49:11 AM)
0 A null
1 B null
2 C null
3 D null
4 E null
```

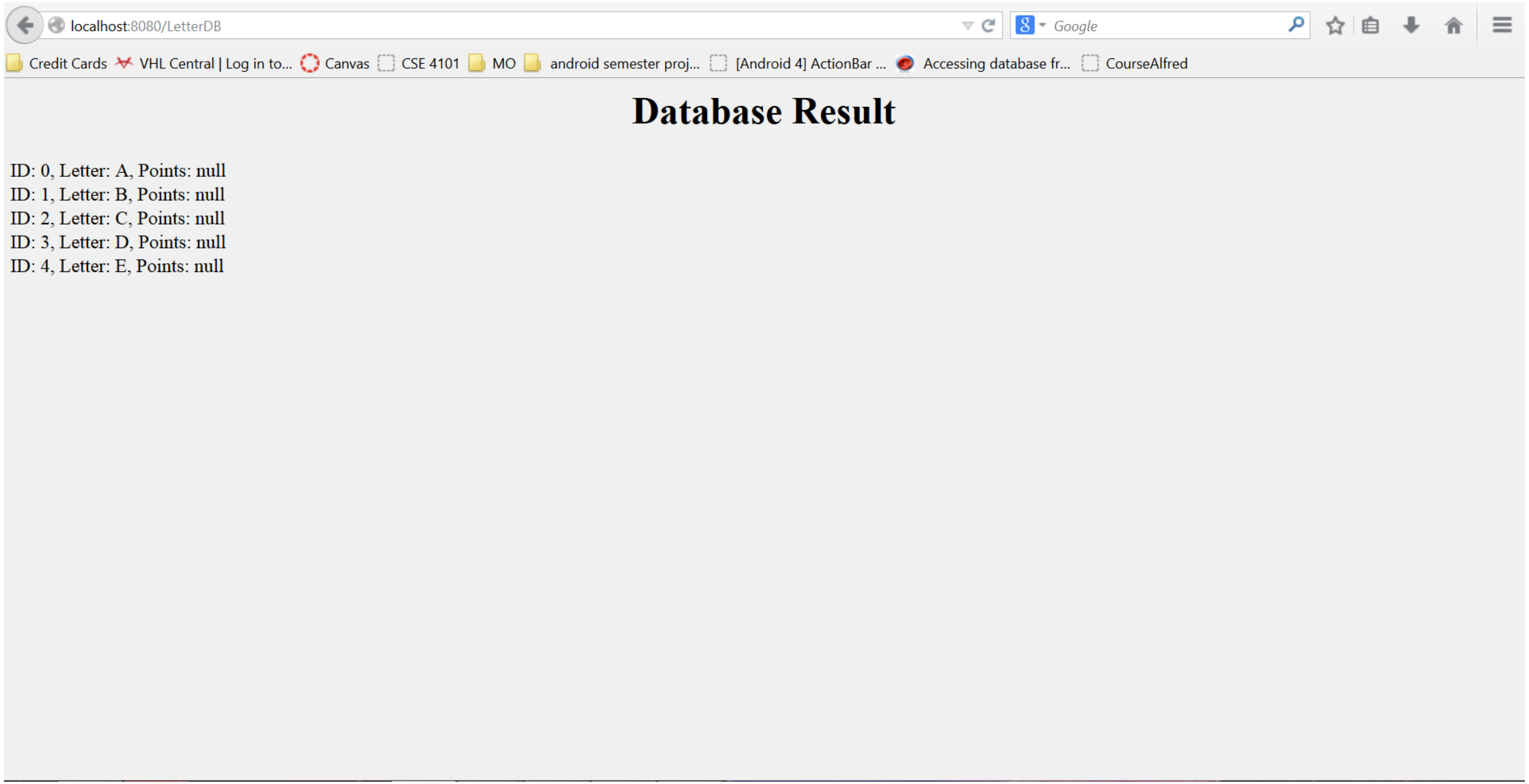
Java servlet webapp with Tomcat



```java
public class LetterDB extends HttpServlet {

    public static void main(String[] args) {
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        Connection conn = null;
        Statement stmt = null;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            String title = "Database Result";
            String docType = "<!doctype html public \"-//w3c//dtd html 4.0 "
                    + "transitional//en\">\n";
            out.println(docType + "<html>\n" + "<head><title>" + title
                    + "</title></head>\n" + "<body bgcolor=\"#f0f0f0\">\n"
                    + "<h1 align=\"center\">" + title + "</h1>\n");

            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager
                    .getConnection("jdbc:mysql://127.0.0.1:3306/pentotext?user=root&password=root");
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("select * from letter_db");

            while (rs.next()) {
                String id = rs.getString(1);
                String letter = rs.getString(2);
                String points = rs.getString(3);
```



```java
                out.print("ID: " + id);
                out.print(", Letter: " + letter);
                out.print(", Points: " + points);
                out.println("<br>");
            }
            out.println("</body></html>");

            rs.close();
            stmt.close();
            conn.close();
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (stmt != null)
                    stmt.close();
            } catch (SQLException se2) {
            }
            try {
                if (conn != null)
                    conn.close();
            } catch (SQLException se) {
                se.printStackTrace();
            }
        }
    }
}
```

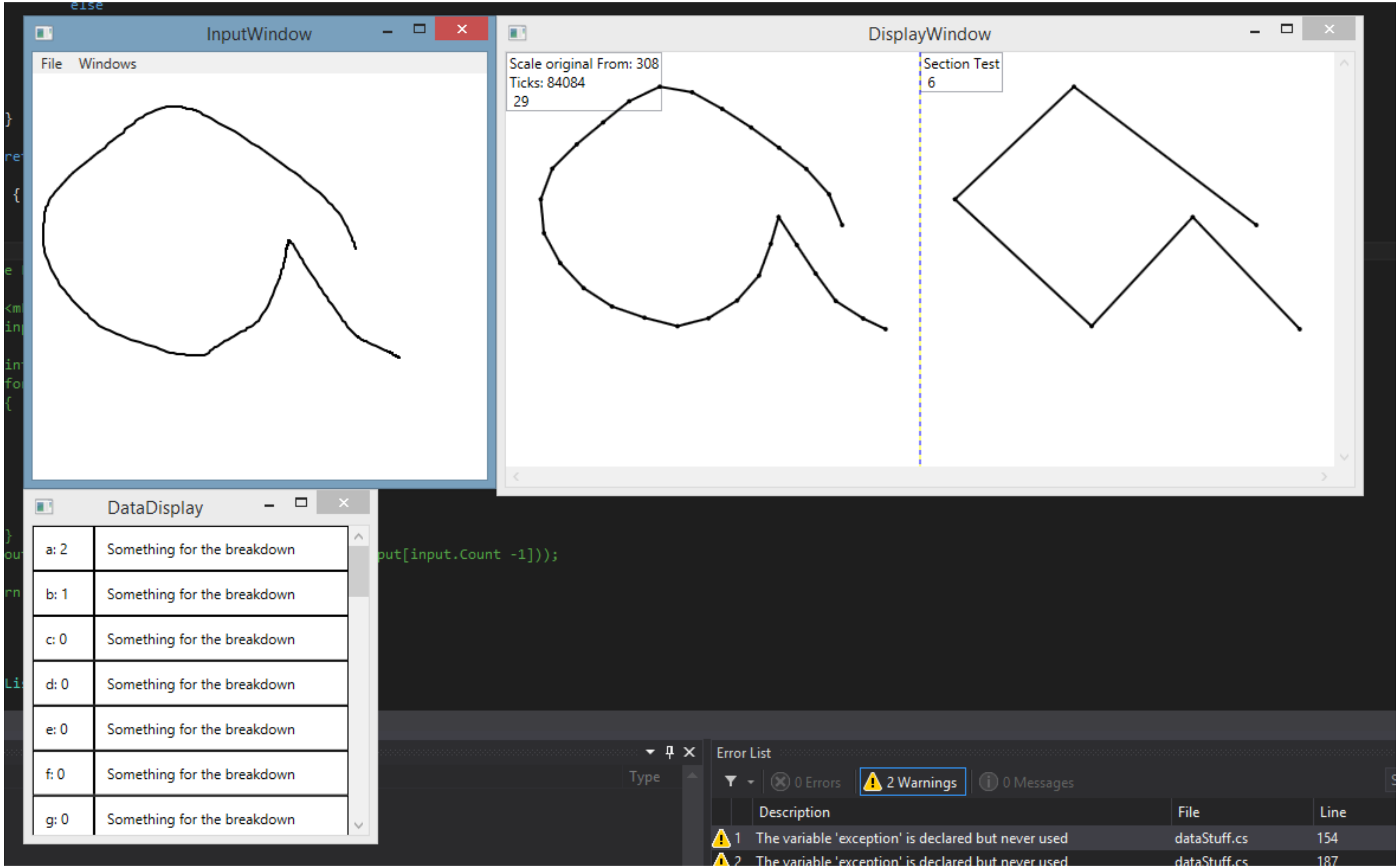

localhost:8080/LetterDB

Credit Cards · VHL Central | Log in to... · Canvas · CSE 4101 · MO · android semester proj... · [Android 4] ActionBar ... · Accessing database fr... · CourseAlfred

Database Result

```
ID: 0, Letter: A, Points: null
ID: 1, Letter: B, Points: null
ID: 2, Letter: C, Points: null
ID: 3, Letter: D, Points: null
ID: 4, Letter: E, Points: null
```

New UI elements



```
Scale original From: 308
Ticks: 84084
 29
```

```
Section Test
 6
```

a: 2	Something for the breakdown
b: 1	Something for the breakdown
c: 0	Something for the breakdown
d: 0	Something for the breakdown
e: 0	Something for the breakdown
f: 0	Something for the breakdown
g: 0	Something for the breakdown

```
put[input.Count -1]));
```

Error List

0 Errors | 2 Warnings | 0 Messages

	Description	File	Line
1	The variable 'exception' is declared but never used	dataStuff.cs	154
2	The variable 'exception' is declared but never used	dataStuff.cs	187

Milestone 3 matrix

Task	Dominique	Kasey
Connect c# code to the java servlet webapp	100%	
Import xml file into database	50%	50%
Determine how to compare input to the points in the database based on a certain threshold of input and pattern	50%	50%
Code the determined way to compare the inputs based on a certain threshold		100%

Summary of Milestone 3 tasks

1. Connect our c# code to connect to the java servlet so we can read the database. We have looked into a few tutorials so far.
2. Import an xml file full of letters and points into the database. This makes populating the database much faster. If we didn't have the database we would have to populate the database manually.
3. Determine the best way to approach comparing input to the database. So how are we going to determine the users input is close to the input in the database? We need to determine a threshold and way to read the patterns.
4. Start coding the approach determined in step 4 and see if we run into anything we have missed.

Demo