**Section 1**

**Name:** Antara pal

**Section 2**

**Web Application URL** : <http://apalwebapp.appspot.com/>

**Section 3**

**The Google Code repository name:** cosc603pal

**Section 4**

**A description (2-3 paragraphs) of what you learned from this project and how it relates to some of the topics covered in lecture.**

I have learned a lot from the provided web resources and this project.

In Task 1 and 2, I have learned about how to extend the development environment to use eclipse for a web application project. In these tasks, I have learned installation of some google plugins like google web tool and application engine.

I have set up a google application engine account and created my first web application using eclipse. I also learned how to run the web application locally and how to deploy it. These tasks are related to maintenance of software. In software maintenance, we need to add new features or functionalities into the existing software for other application needs. I had installed eclipse in previous project. But it was not able to run or support any web application. So, this kind of maintenance is perfective maintenance.

In Task 3, I created a web application named Stutter Web App. I have learned about a new technology Ajax, and how is this useful in web applications. I have learned how this web application works by running it locally. I paid much attention to some of the source code files like css and html and server/client source codes. I have learned about debugging. Debugger is very useful to see the code step by step putting break points and it could capture the variable values by putting watch on them. Debugging is one of the useful techniques I have learned from the lecture also.

In Task 4, added and changed some of the functionalities with the *Stutter* functionality from project 1. On client side code Stutter\_Web\_App.java was modified to change TextBox feature TextArea. In this task, other files were also modified according to project requirements. The details are given in Section 6. This task was about corrective/perfective maintenance to meet the requirements. In Task 5, I deployed my final web application on the web (<http://apalwebapp.appspot.com/>).

**Section 5**

**A brief description (1-2 paragraphs) of what was the hardest part about this project.**

For me, the hardest part was to modify the server code GreetingServiceImpl.java to meet stutter requirements. It could be helpful to use debugger while I was not getting the actual result. But my computer has some issues with the debugger, so it was harder to find out errors during running the application locally.

**Section 6**

**A brief description (1-2 paragraphs), how you changed your original code to accommodate it into a web application.**

I have modified the main HTML file (war/Stutter\_Web\_App.html), the Cascading Style Sheet (war/Stutter\_Web\_App.css), the FieldVerifier.java source file (src/\*\*\*.shared/FieldVerifier.java) and

the Stutter\_Web\_App.java source file (src/\*\*\*.client/Stutter\_Web\_App.java) file to implement TextArea while taking input in client side instead of TextBox.

I have modified the GreetingServiceImpl.java source file (src/\*\*\*.server/GreetingServiceImpl.java) file for server requirements.

In **war/Stutter\_Web\_App.html** file,

<h1>Web Application Starter Project</h1> was replaced by <h1>Stutter Web Application by Antara Pal</h1>.

In **war/Stutter\_Web\_App.css** file, width of DialogBox changed, it gives width for dialog box while showing the output. And I added new code for TextArea, which provides client side input area, as shown below:

*.gwt-DialogBox* {

width: *500px*;}

*.gwt-TextArea* {

lenght: *20px*;

width: *600px*;}

In **src/\*\*\*.shared/FieldVerifier.java** file,

**public** **static** **boolean** isValidName(String name) {

**if** (name == **null**) {

**return** **false**;

}

**return** name.length() > 3;

}

was replaced with text instead of name, and length() > 9 instead of 3.

In the **Stutter\_Web\_App.java** source file,

**import** com.google.gwt.user.client.ui.TextBox was replaced with **import** com.google.gwt.user.client.ui.TextArea.

**In method onModuleLoad(),**

**final** TextBox nameField = **new** TextBox();

nameField.setText("GWT User");

was replaced with

**final** TextArea textArea = **new** TextArea();

textArea.setCharacterWidth(80);

textArea.setVisibleLines(15);

textArea.getCharacterWidth();

textArea.getVisibleLines();

textArea.setText("The software testing and maintenence is Fun!");

RootPanel.*get*("nameFieldContainer").add(nameField) was replaced with RootPanel.*get*("textFieldContainer").add(textArea);

In method **sendNameToServer()**, errorLabel.setText("Please enter at least four characters") was replaced with errorLabel.setText("Please enter at least ten characters").

In **src/\*\*\*.server/GreetingServiceImpl.java** file, the code was modified to a great extent. So I am putting the whole code here: Modified/added code shown in sky blue color. It was modified to add the Stutter functionalities of Project 1.

**package** com.example.mywebapp.server;

**import** com.example.mywebapp.client.GreetingService;

**import** com.example.mywebapp.shared.FieldVerifier;

**import** com.google.gwt.user.server.rpc.RemoteServiceServlet;

/\*\*

\* The server side implementation of the RPC service.

\*/

@SuppressWarnings("serial")

**public** **class** GreetingServiceImpl **extends** RemoteServiceServlet **implements**

GreetingService {

**public** String greetServer(String input) **throws** IllegalArgumentException {

// Verify that the input is valid.

**if** (!FieldVerifier.*isValidName*(input)) {

// If the input is not valid, throw an IllegalArgumentException back to

// the client.

**throw** **new** IllegalArgumentException(

"Name must be at least 10 characters long");

}

**else**

{

**int** numberOfLines = 1;

String[] lines = **null**;

String lastWord = **null**;

String results = "";

// Get the first line from the file argument

// line = input.readLine();

lines = input.split("\n");

// Keep reading and analyzing lines until end of file

**for** (**int** x = 0; x < lines.length; x++){

// Splits the line into words and puts in an array of strings

String[] words = lines[x].split(" ");

// Compare with the last word on the previous line

// if (words [0].equals (lastWord))

// System.out.println ("Repeated word on line " + numberOfLines + ": " + words[0]);

// Stop before the end, nothing to compare the last word with

**for** (**int** i = 0; i < (words.length-1); i++){

// Check to see if the current and subsequent words are the same

**if** (words [i].equals (words [i+1]))

results = results + "Repeated word on line " + numberOfLines +

": " + words [i] + "<br>";

//else

//return(" No repeated words found.");

}

// Save last word in the line

lastWord = words [words.length-1];

// Increment line counter

numberOfLines++;

}

**if**(results.isEmpty())

results = " No repeated words found.\n";

**return** results;

}

}

}

**Section 7**

**A brief description (1-2 paragraphs) on what you did to test your web application to ensure that it meets the requirements.**

I ran the web application locally. I tested the client side URL with the input, and checked if it can process the input to show the actual output or not. If it showed any erroneous result, I tried to find out the reason by analyzing the server and client side codes. I tried and tested many times to get actual output.

**Section 8**

**A brief description on ways that the user interface of your *Stutter* web application could be changed to improved.**

To improve the user interface of the Stutter web application, CSS file code should be modified. Padding can be added to give a uniform border in text area (client side input space) to make the interface good looking. Contrast colors can be added to manage focus and attention. The main HTML file could be modified for giving it a look of a form style and auto-focus on input area could be added as an attractive feature. The buttons could be modified in this file too.