CS699: Run-a-prog, a toolbar for firefox

Instructor: Prof. G. Sivakumar

Prashanth K	(08305006)
Sree Shankar	(08305023)
Sumair Ahmed	(08305032)

{pkamle, sree, sumair}@cse.iitb.ac.in

November 12, 2008

Contents

1 ProblemStatement

Most of the Sites have a lot of content in their Web Pages that User don't desire to see. Few of them would be Flash/Advertisement/Images embedded in a page. Removal of these features from the Web Pages would be more convenient, which inspires us in designing these webpages to their needs. Such Customized WebPages would be more helpful in Users Perspective. Our goal in this Project is to define a Toolbar that can be added to Firefox and has the following features.

- **Save Selected Text**: Toolbar can be used to save the text selected as a File onto the local directory
- **Add Program**: Run the custom applications on the text selected at just adding this to the Toolbar.
- **Flash/Image/Text Block**: User can customize the WebPages by removing Flash and/or Images and/or Text. We provide the feasibility in selecting either or all of the options.

2 Functional Description

2.1 Add/Remove Program

Toolbar can run the custom applications on the selected text. It embeds different applications/executables such as TextPad/lpr/wc or any script written by the user by clicking Add Program and browsing to the executable path. Subsequently User can directly run this application on the text selected. Toolbar also provides the feature to remove the application.

2.2 Save Selected Text

Toolbar has the capability to select the text from a webpage and save it on the local directories. On the First attempt, toolbar would request for a FileName and also open up the Browse window for the User to decide the location and save it. All the subsequent saves of the files would just request for a FileName and automatically put it in the before mentioned directory.

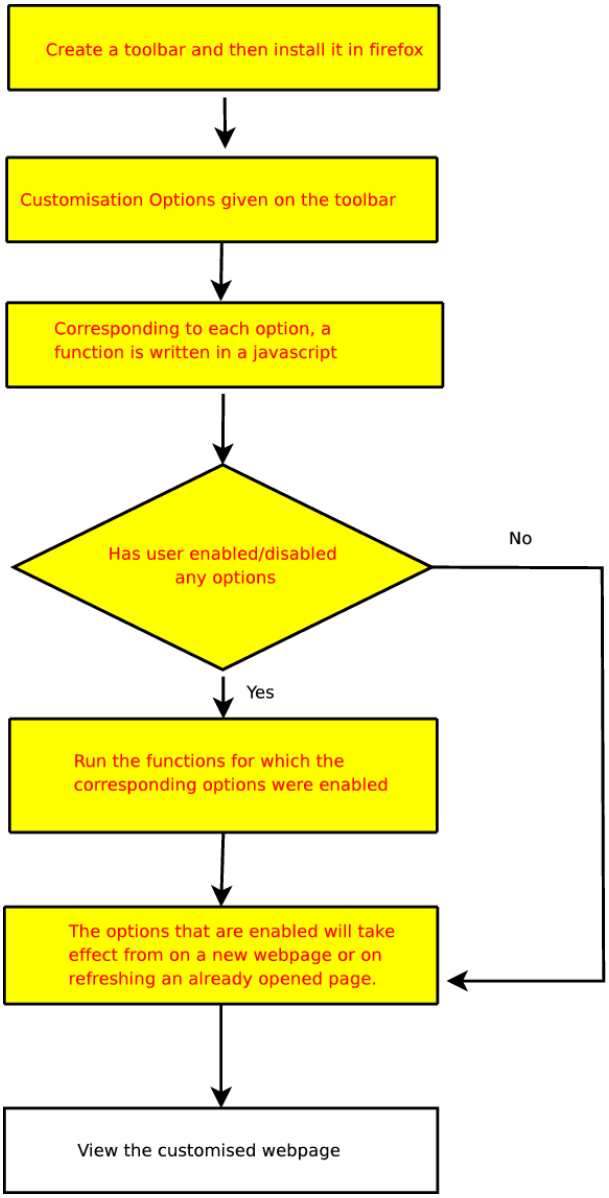
2.3 Flash/Image/Text Block

Toolbar also has the option of blocking few features from a WebPage such as Flash, Image and Text. Based on these choices either or all of them will be removed. Usually as the Web Page loads, our JavaScript validates these DOM based pages against the block functions. The Script removes/hides these elements from the WebPage and renders the new HTML DOM to display.

3 SolutionDesign

Our Design would be similar to the below flowchart 1. The foremost part would be to create a new toolbar that could get installed in Firefox and has both static and dynamic features. Static means, the Toolbar already has few read-only sections like Block/Add

Figure 1: Solution Design



Program/Remove Program/Save Selected Text. And Dynamic means the Toolbar should be able to withhold the new sections that would get added/removed as and when User selects Add/Remove Program. The Design of these various features are mentioned below.

decision = [diamond, draw, fill=blue!20, text width=4.5em, text badly centered, node distance=3cm, inner sep=0pt] block = [rectangle, draw, fill=blue!20, text width=10em, text centered, rounded corners, minimum height=1em] line = [draw, -latex'] cloud = [draw, ellipse,fill=red!20, node distance=3cm, minimum height=2em]

[scale = 0.1][node distance = 2cm, auto] [block] (init) Install the toolbar; [block, below of=init] (options) Various blocking and text selecting options; [decision, below of=options] (decide) Has user enabled any options?; [block, below of=decide, node distance=3cm] (effect) The options that are enabled will take effect whenever a new webpage is opened or when refreshing an already opened webpage; [block, below of=effect] (view) View the modified webpage; [line] (init) – (options); [line] (options) – (decide); [line] (decide) – (effect); [line] (effect) – (view);

3.1 Save Selected Text

As the user selects the text and clicks on the SaveSelectedText feature, the Toolbar immediately recognizes the text saved and requests the user for a FileName. On entry of a FileName, it also opens up a Browse window that enables the user to save it into his desired location.

3.2 Add Program

Toolbar provides the user- Add/Remove Program options to run his custom applications from the Browser. When the User selects add, he will be prompted to locate its executable path and consequently the chosen application gets added to the Toolbar. Later the User opens the selected text with these custom applications.

3.3 Flash/Image/Text Block

Blocking Algorithm follows this procedure:

1. Get the HTML DOM of the WebPage. Apply all or some of the below based on the User preferences.
2. Modify the images by hiding it. Check for <IMG> elements and set its style to hidden.
3. Flash elements are usually within <embed> or <object> tags. We search for ShockWave/Flash/SWF/eyewonder in these elements and then remove that node if found.
4. Modify the text elements by emptying its content.
5. Render the new HTML DOM to display.

4 Implementation

Toolbar is designed and implemented in Javascript. Initially, we thought of using the GreaseMonkey scripts and build the Toolbar over it. But we faced a lot of problems in integrating it. And so we have come up with the Javascript code that not just creates the Toolbar, but also emulates GreaseMonkey.

4.1 Save Selected Text

On click of the SaveText field, toolbar prompts the user to enter the FileName. And later user is prompted to browse the path and click OK, in which this file is created, the selected content is automatically saved in the file. While this is the procedure for the initial attempt, the subsequent saves would just request for the FileName remembering the Directory path.

4.2 Add Program

Add Program field in the Toolbar adds a custom application onto Firefox. When this button is clicked, toolbar prompts the user to browse the path to executables. On selection of this, user adds a FileName and subsequently the script adds a new element to this toolbar and links this element to the executable path provided.

4.3 Flash/Image/Text Block

- **Flash Block**: WebPages normally have Flash elements within <embed> or <object> tags. Using XPATH in JavaScripts, search in the HTML DOM for these elements. XPATH is to find the desired pattern from the HTML page. Then check if these are actually Flash objects by looking at the 'type' or 'src' attributes are either ShockWave/Flash/SWF/eyewonder.
- **Image Block**: All the Images are denoted with <img> tags. Using the XPATH, search these elements and simultaneously modify its attributes to set the Visibility property to HIDDEN
- **Text Block**: Text could be anywhere within the HTML DOM ie., body of the page. It may exist within <P> or <FONT> or <DIV> tags or even more. XPATH has a feature that filters all these elements. Later setting the nodes' DATA to null will remove the text.

5 Analysis and future work

We use the event-driven programming approach just like thousands of other plugin developers. Since the project is small, this method is best suited. As it grows bigger, an object oriented approach will be better.

This project only explored the surface of firefox add-on development. If we had more time, we would have loved to explore the depths of firefox addon development and come up with something as good as the popular firefox add-on DownloadThemAll.

We would like to make the following improvements

- At present Toolbar blocks all the Images from a WebPage. We shall make selective restriction on few images. This blocks all the Images from one WebSite.
- Improve the various blockers to block asynchronously loaded content through AJAX.
- Improve the User Interface of the Toolbar.

Through this project, we learnt how to work in a team and divide tasks effectively. Also, we are now familiar with firefox add-on development and understand the limitations and power of javascript.

6 Results

Below are the screenshots of the Blockers - Text, Image, Flash

6.1 Screenshots

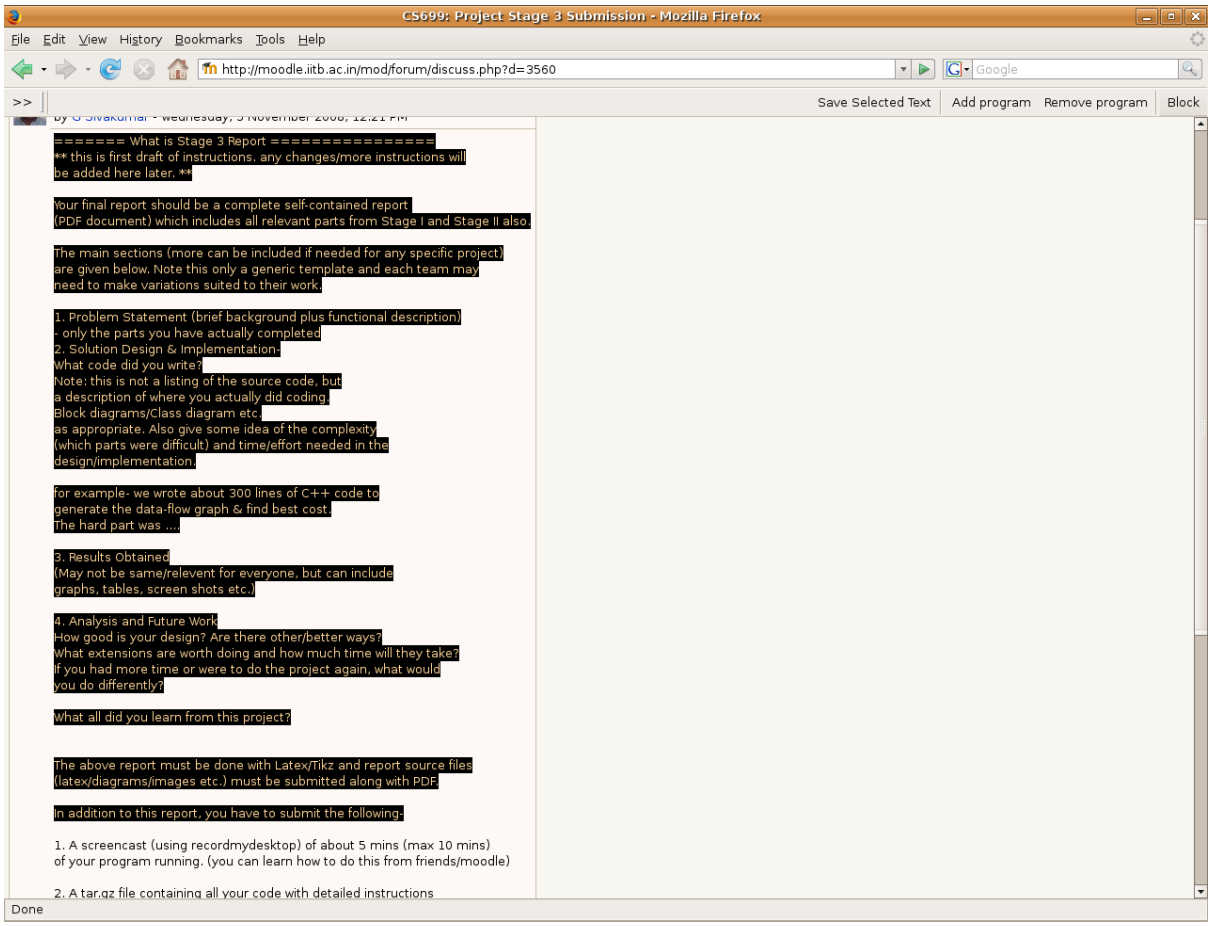
6.1.1 Add Item

Figures 2 to 6

6.1.2 Save Selected Text

Figures 7 to 9

Figure 2: Moodle Page - Select Text



6.1.3 Blockers -Text/Image/Flash

Figures 10 to 15

Figure 3: Moodle Page - Click Add Program button

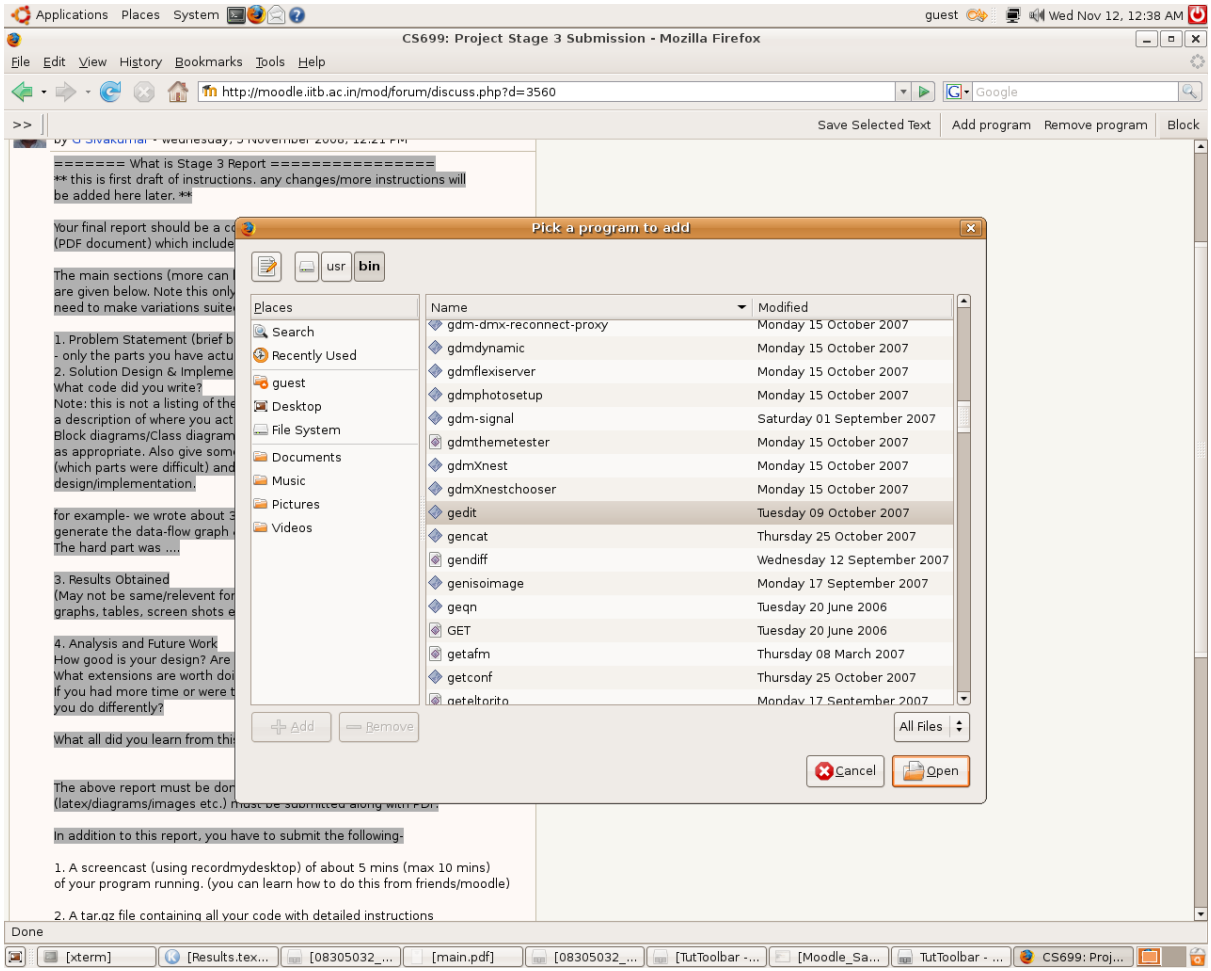


Figure 4: Moodle Page - Enter FileName

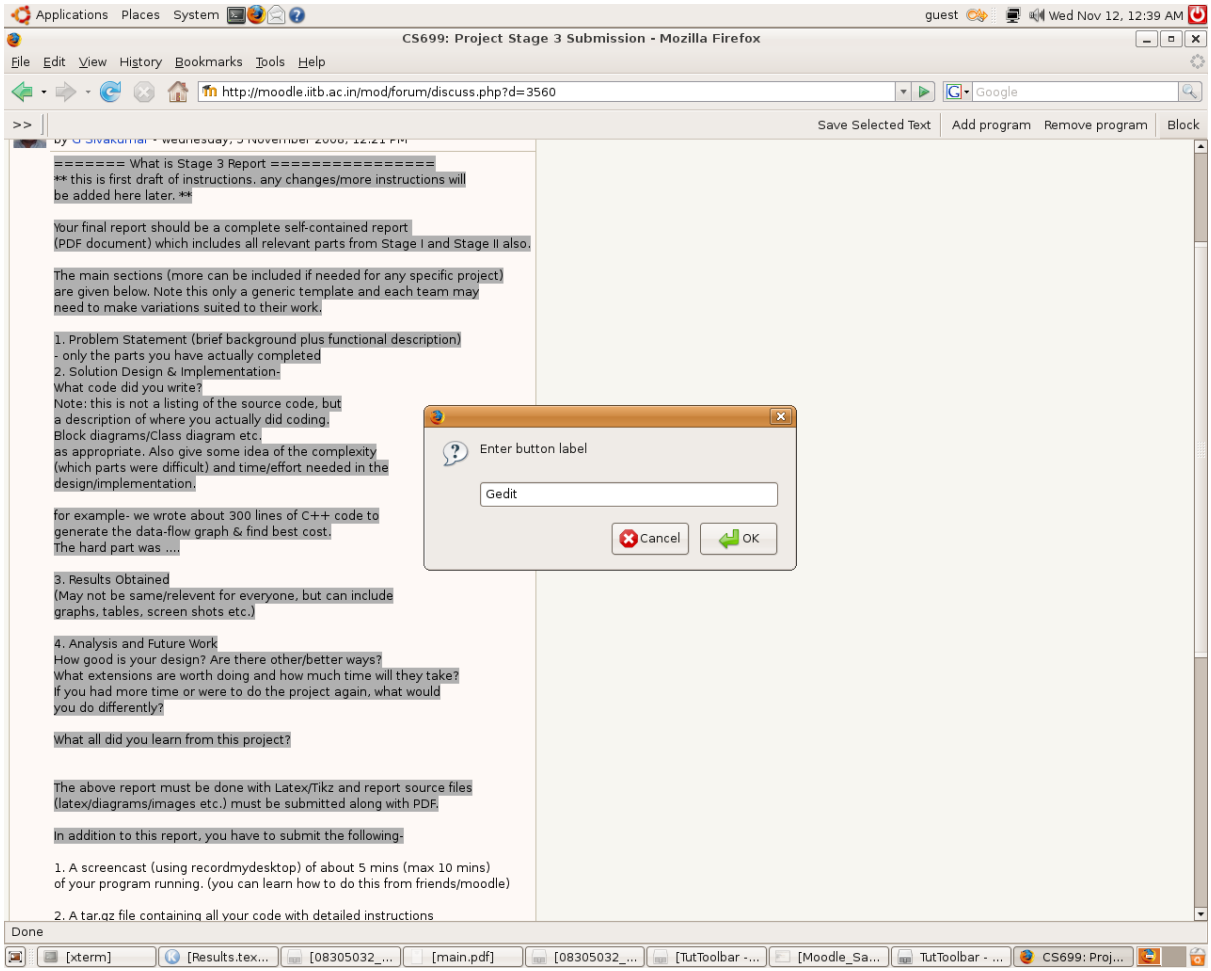


Figure 5: Moodle Page - Click GEDIT on the Toolbar

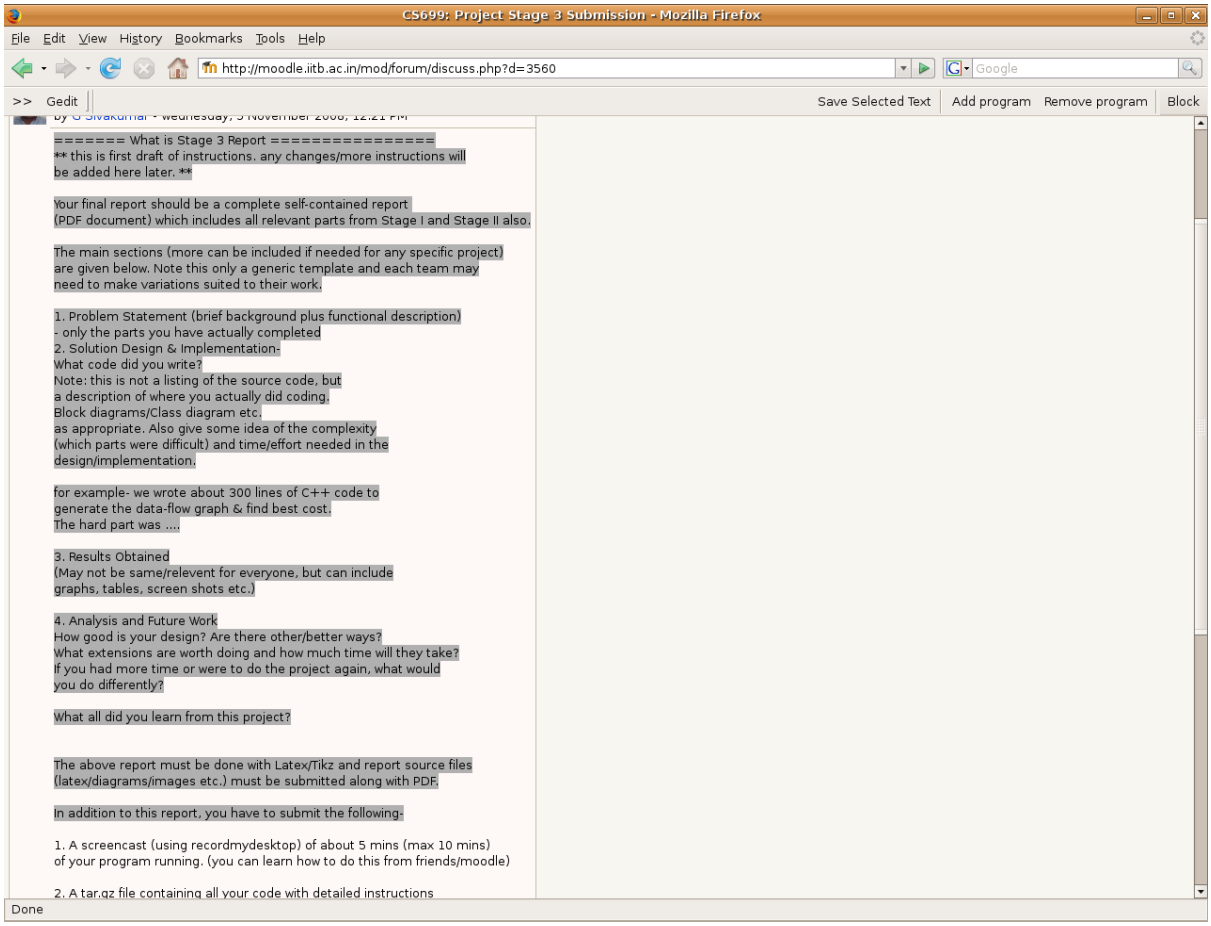


Figure 6: Moodle Page - Selected Text in GEDIT

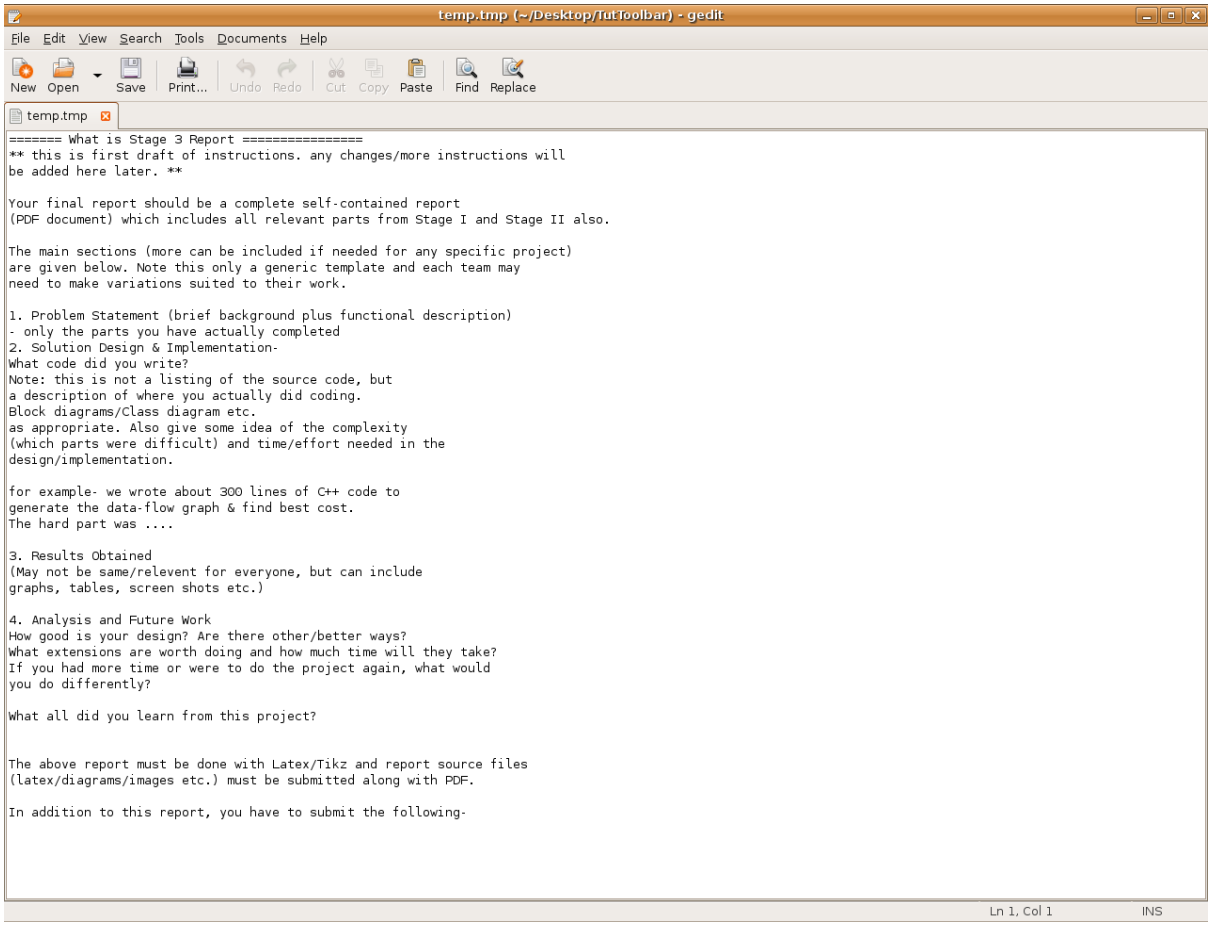


Figure 7: Moodle Page - Select Text

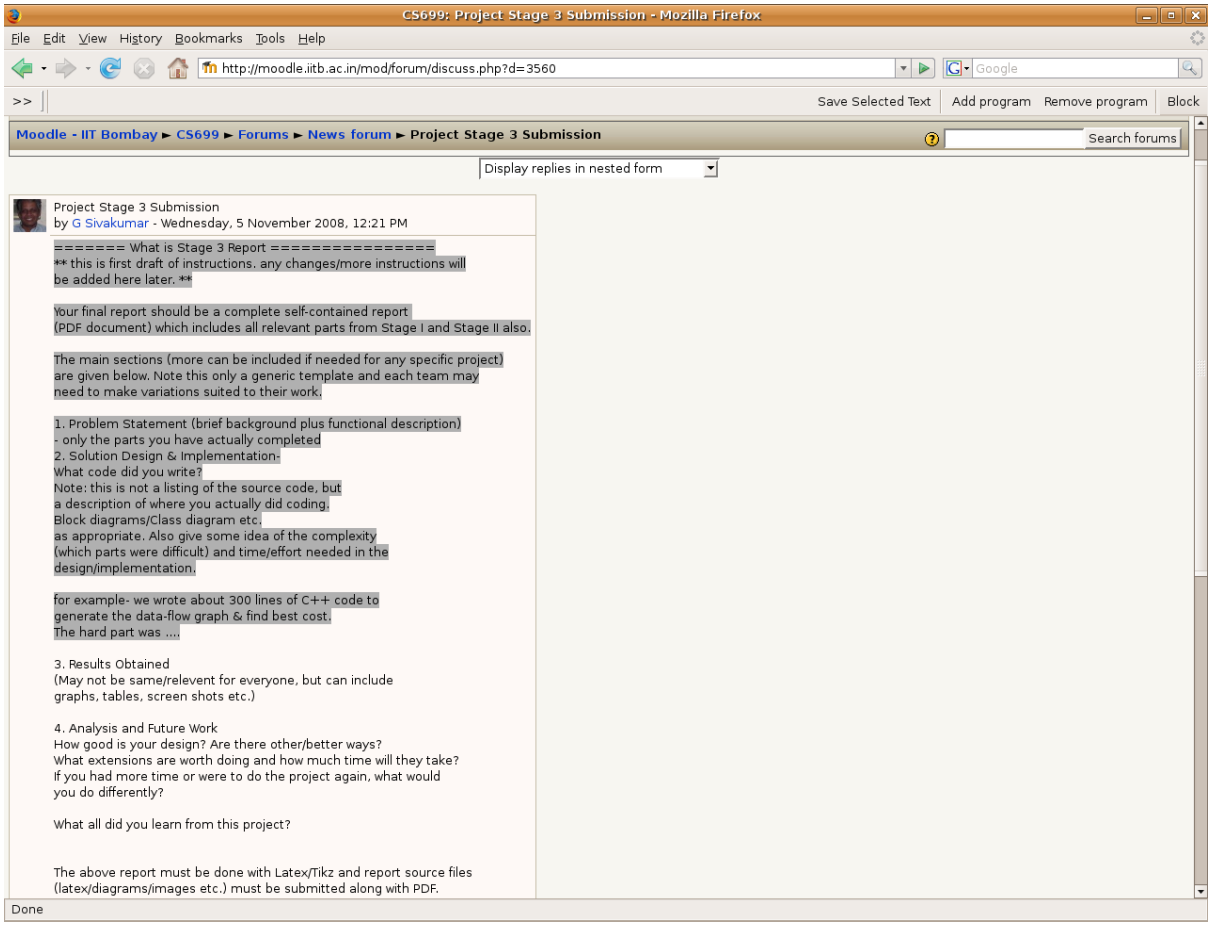


Figure 8: Moodle Page - Click SaveText button and Enter FileName

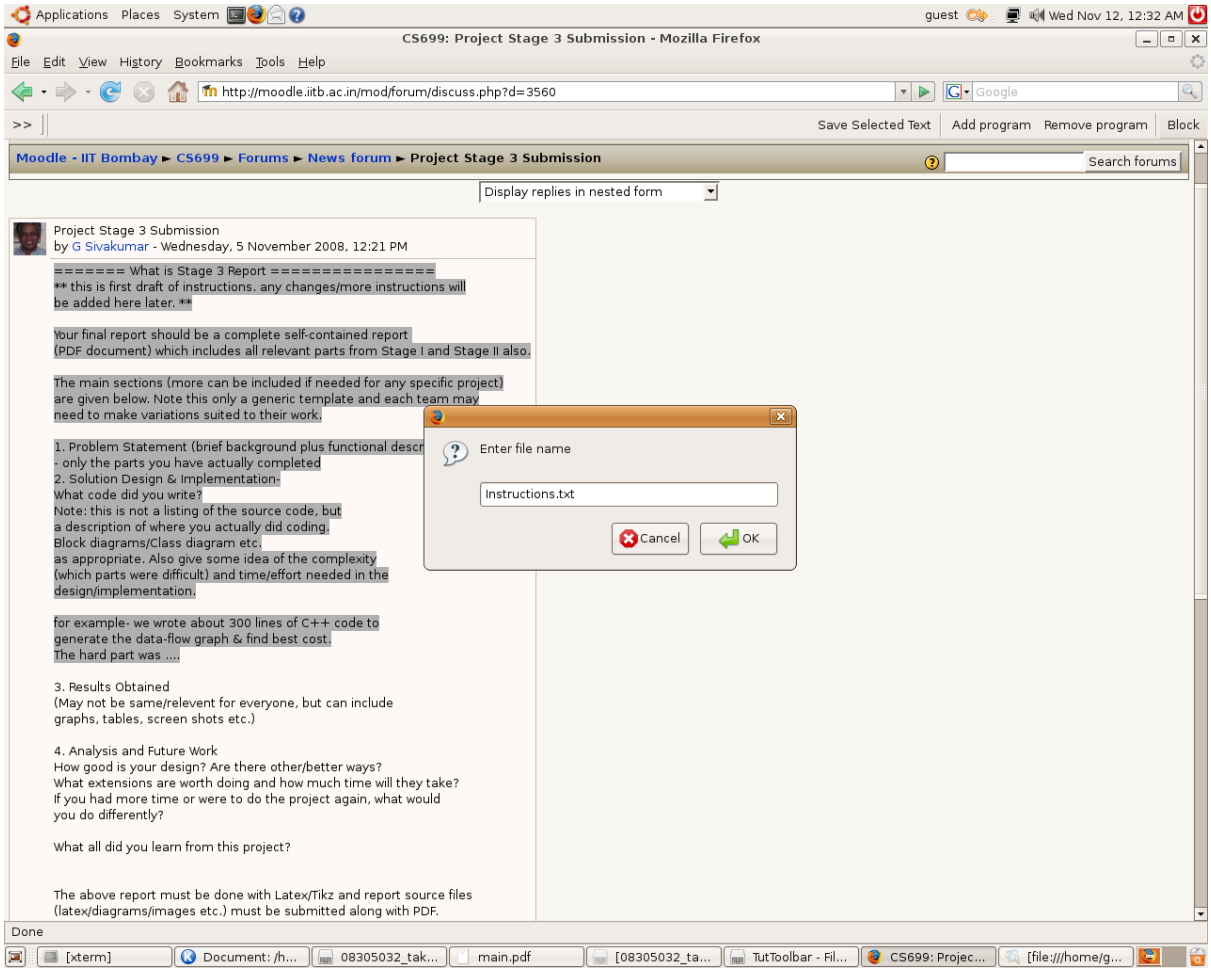


Figure 9: Moodle Page - Text Saved & Opened in Gedit

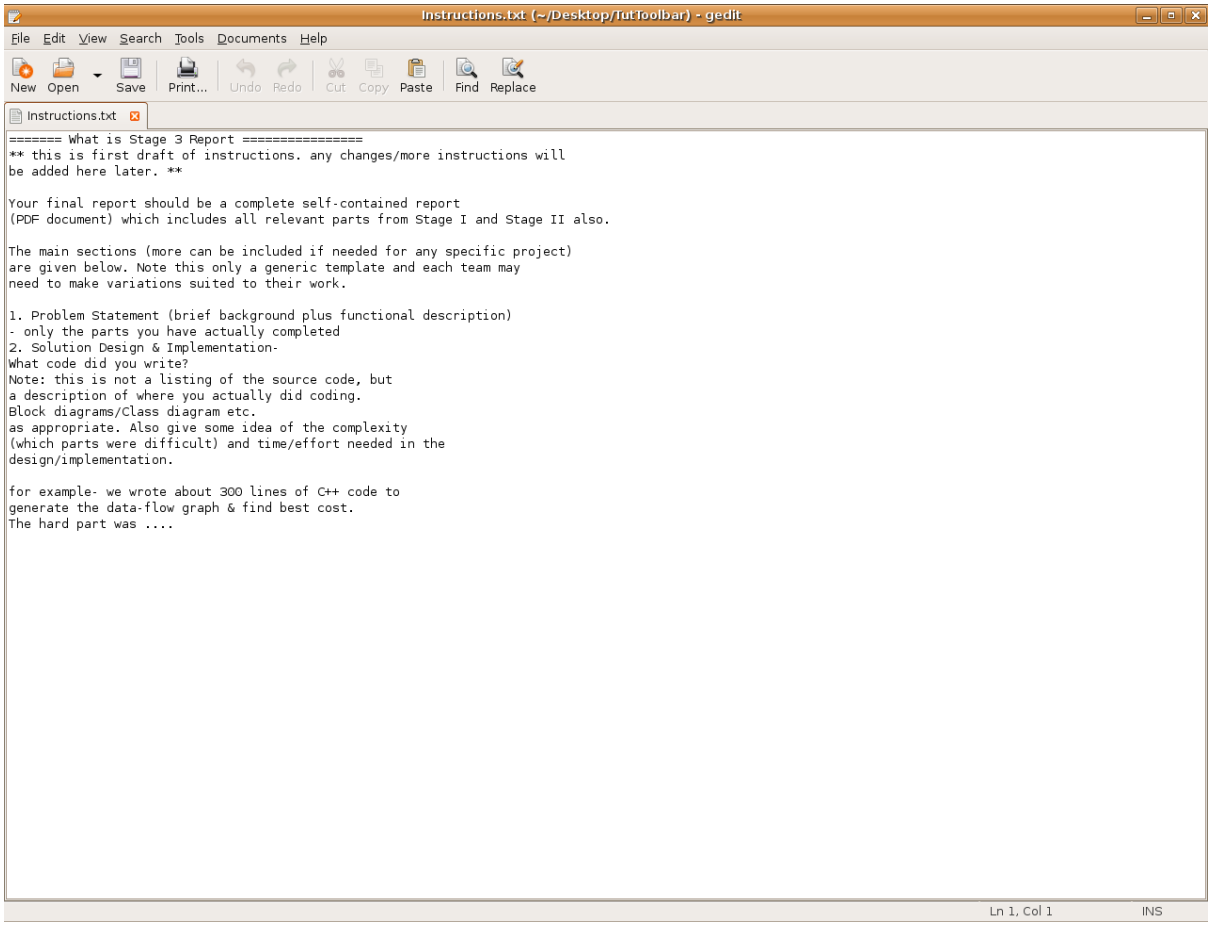


Figure 10: Google Page

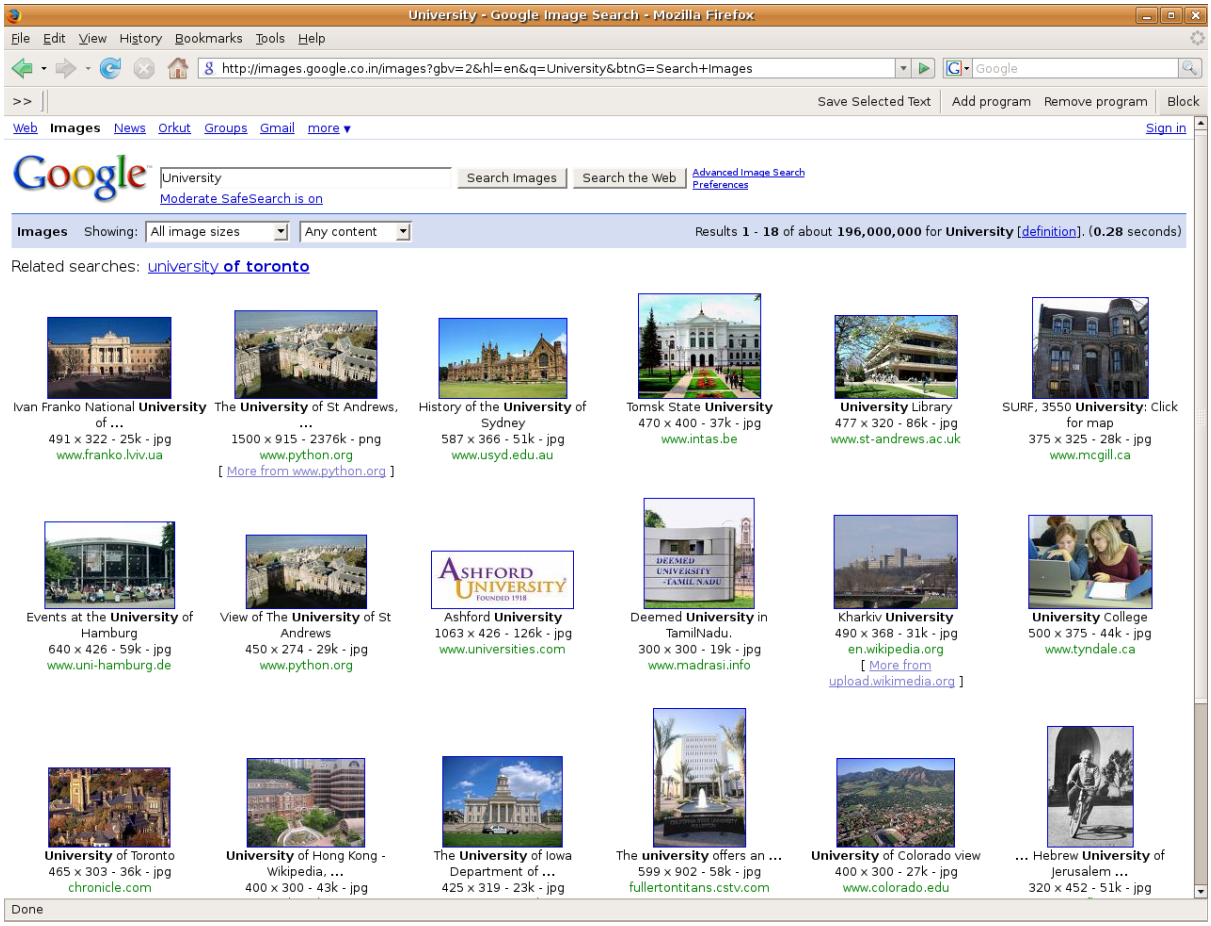


Figure 11: Google-Text Blocked Page

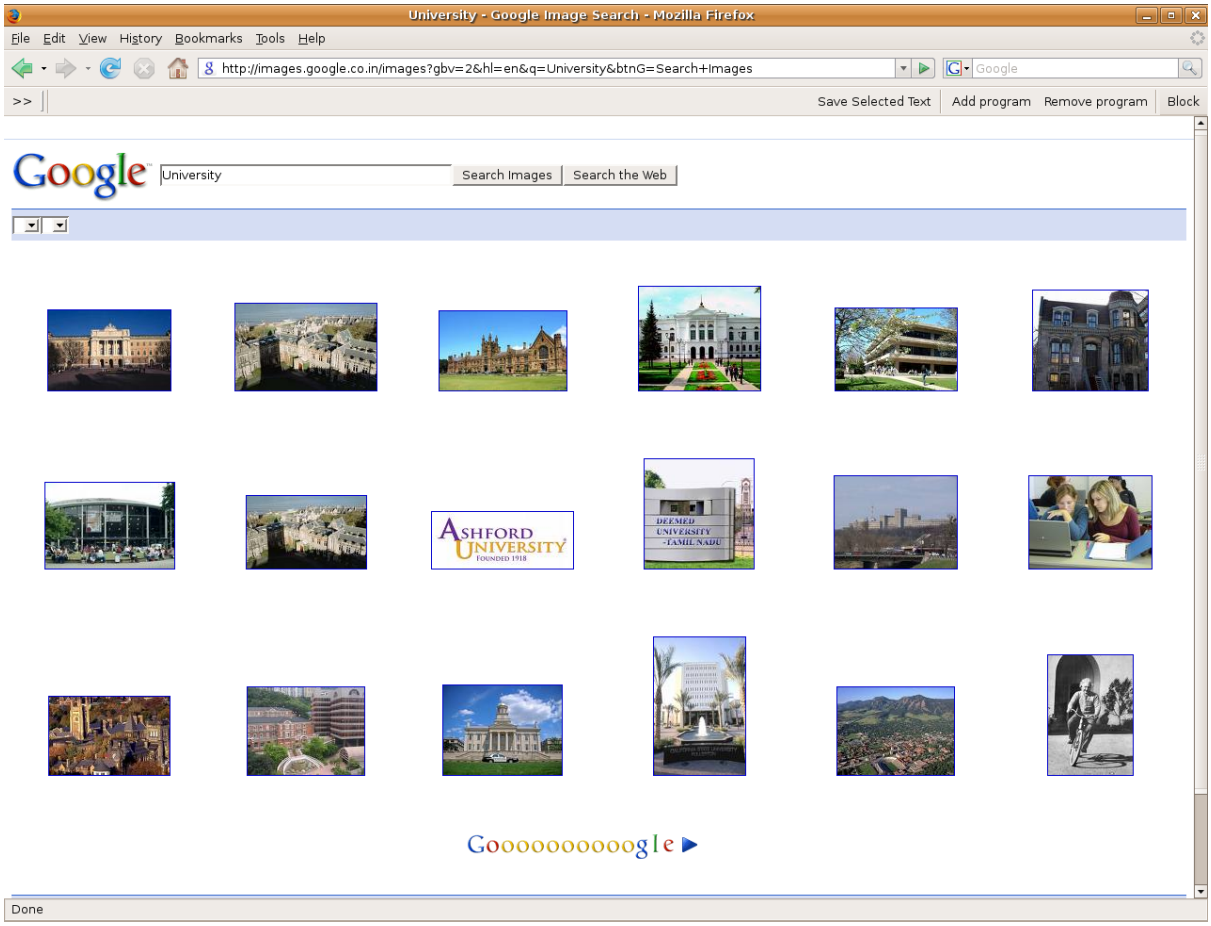


Figure 12: Flickr Page

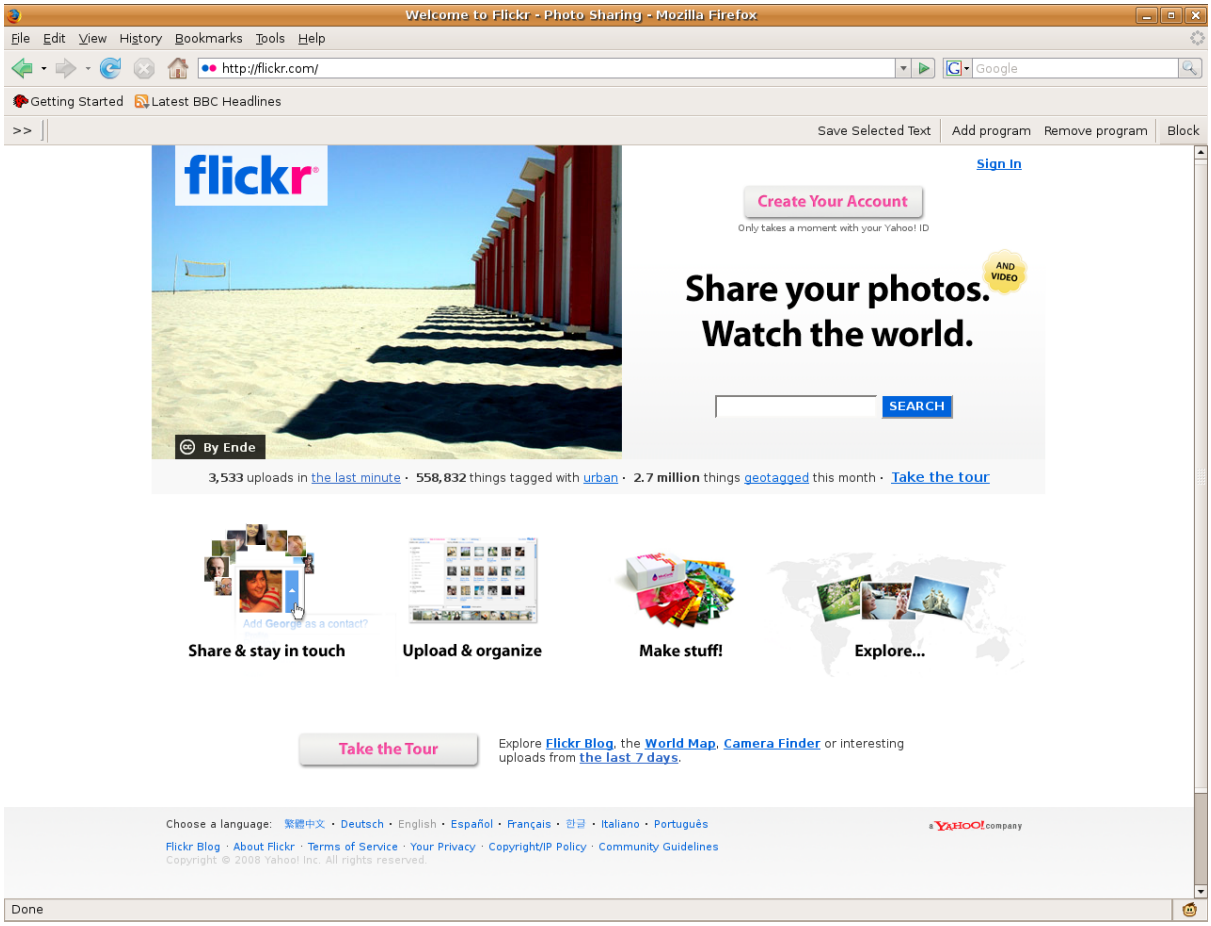


Figure 13: Flickr-Image Blocked Page

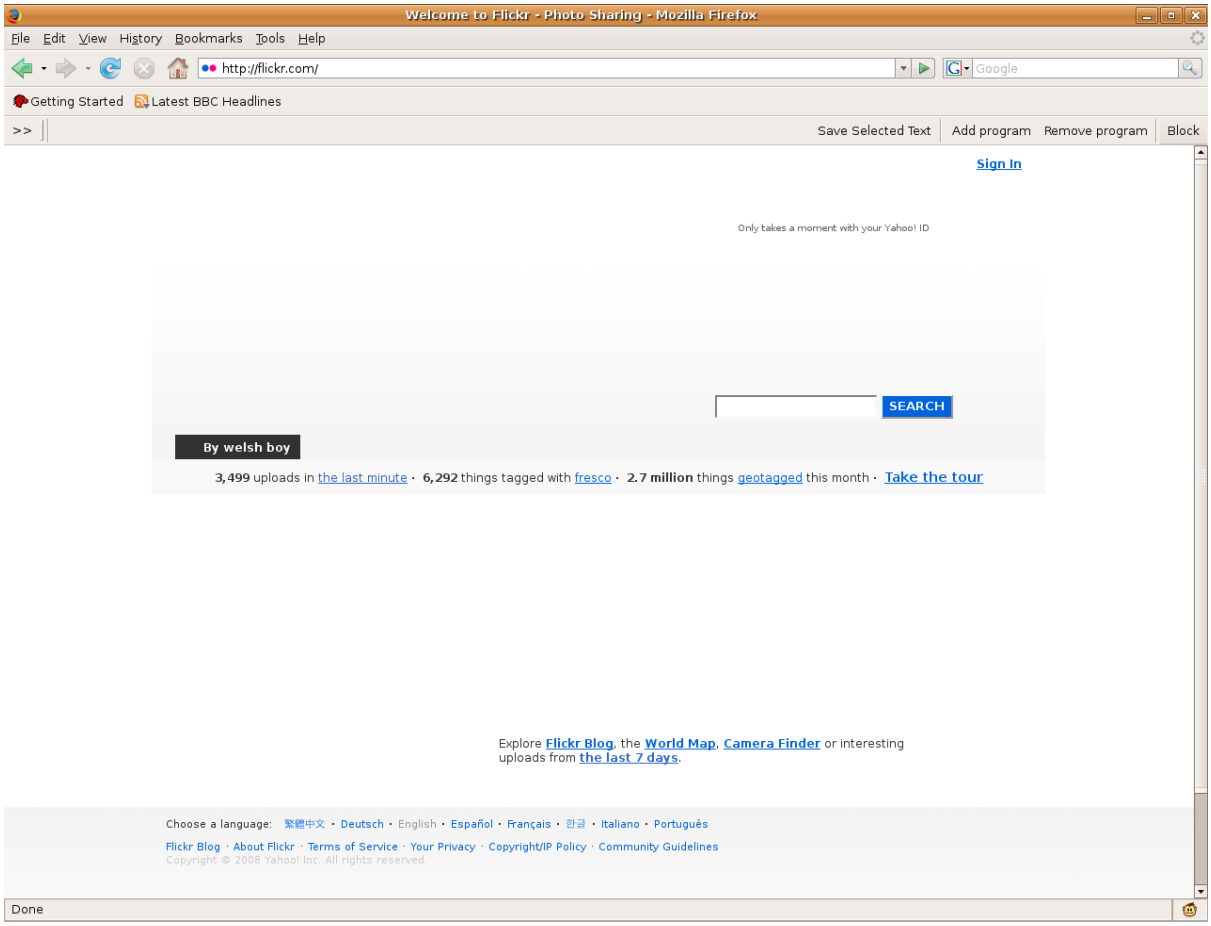


Figure 14: Sony Page

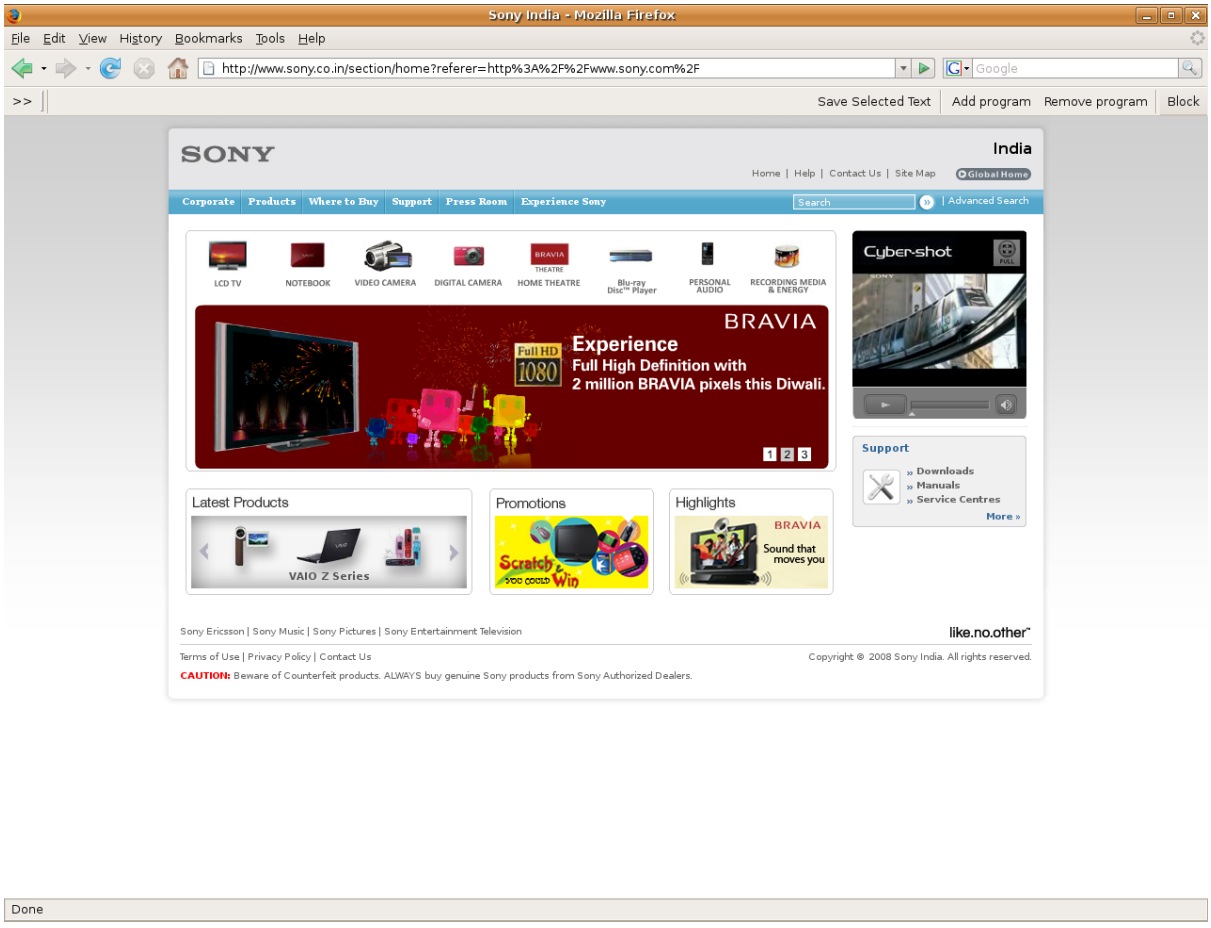


Figure 15: Sony-Flash Blocked Page