CSU498 PROJECT
REPORT

ASYNCHRONOUS CANVAS SHARING

Submitted in partial fulfilment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

ANOOP K S	B080193CS
ARJUN S	B080011CS
JAIMY EMMANUEL	B080042CS
JITHIN PAUL	B080484CS

Under the Guidance of
Mr. G. Gopakumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
NIT CAMPUS PO, CALICUT
KERALA, INDIA 673601

April 10, 2012

Certificate

This is to certify that the project work entitled "**Asynchronous Canvas Sharing**", submitted by Anoop K S (B080193CS), Arjun S (B080011CS), Jaimy Emmanuel (B080042CS) and Jithin Paul (B080484CS) to National Institute of Technology Calicut towards partial fulfillment of the requirements of the award of Degree Of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the work carried out by them under my supervision and guidance.

Project Guide
Mr. G. Gopakumar
Assistant Professor

Place : Calicut
Date : 01-05-2012

Head of Department

Office Seal

Acknowledgement

We would like to sincerely thank our guide, **Mr. G. Gopakumar** (Associate Professor, Dept.of Computer Science & Engineering, NIT Calicut), for his invaluable support and guidance towards this project. We would also like to express our sincere thanks to **Dr. Priya Chandran** (Professor and Head, Dept. of Computer Science & Engineering, NIT Calicut) and **Mrs. Subhasree M** (Assistant Professor, Dept. of Computer Science & Engineering, NIT Calicut) for their suggestions, encouragement, and constant support throughout the project. We also thank our friends and family for their help and support throughout. Finally, we thank all the faculty and staff of Department of Computer Science and Engineering for their help.

Abstract

Usually, making a drawing or design by two or more persons located at different places is done by users sending their work to other users; which is a tedious task. Here, we introduce a web application named "Asynchronous canvas sharing" that facilitates multiple online users to share a single canvas (drawing board) in real time inside a browser. If one user edits the drawing board, it is immediately visible to other user without refreshing the page. Using this service, people can collaborate easily to develop designs and other graphic drawings. This can be used for building multi-user graphics editors.

A text/voice chat client is integrated with the web application to enhance the interactivity. The drawing tools and server pages will be realised using Javascript with the help of libraries like jQuery, and PHP/ASP respectively.

Contents

Chapter 1

Problem Definition

Long distance collusion between faculty, students and researchers has become an important part of engineering education. An area that needs concrete improvement is web based collaboration. There is a strong requirement for tools that assist the above mentioned parties to work together and solve problems easily and efficiently from a distance.

The broad problem to be addressed is how multiple users, physically distant from each other, can collude over projects effectively. Specifically, the focus will be on collaborative designing and drawing. The problem involves developing an online utility that facilitates vector- based drawing communication and text communication between the parties.

This problem has two sides. The drawing activities of all the actors must be synchronized in real-time. Also, the integrity of the data must be preserved in due course of the activities. But the project face significant challenges and this is the other side of the problem we address.

In this project, we implement an application Asynchronous Canvas Sharing where multiple users can share a drawing board and make changes to it in real-time. They can interact with each other using the associated chat client.

Chapter 2

Introduction

2.1 Background and Recent Research

Traditionally, collaborating on a drawing or design using internet is done by users in turns, where each user sends their work to other users who make corrections and then re-forward them. Here, our project introduces a web application named "Asynchronous Canvas Sharing" that enables multiple online users to share a single canvas (drawing board) in real time inside a browser. Any change made by one user is visible to other user immediately without refreshing the page.

There exist other applications that implement the functionalities of Asynchronous canvas sharing to a certain extent. But their implementation differs from ours in certain key aspects. First of all, the client and the server interact with each other via a technique called polling in which the client sends request to the server continuously.[10] The server responds to each request with a no until the request is fulfilled. These unnecessary sessions takes up the bandwidth of the network which slows down the systems. A new technique called comet will be used where the request from a client is always fulfilled by the server. The request is held back by the server until it can be granted. The inept sessions can be thus avoided which improves the performance of the systems.

Most applications provide the drawing board facility alone. It is not possible to engage in real-time web based learning unless the users can interact with each other via voice or text. A chat facility is included along with the drawing board which nullifies this problem. The most notable feature of Asynchronous Canvas Sharing when compared to other similar applications is its real-time collaborative digital sketching. The changes made by one user are instantaneously visible to the other users with no noticeable lag. When the brush moves on the drawing board, the coordinates are recorded

and are sent through the server to the recipients, where it is redrawn with the received data. The packets of vertex points sent by the drawing board allow the rendering of the vectors.

As part of HTML 5 standard for the web, there are currently developments undergoing to forge the concept of web sockets that facilitates communication over World Wide Web. Most browsers at present do not support HTML 5 and also specifications of HTML 5 are not still fully standardized.

2.2 Motivation

Visual media such as drawing and text chat is essential for effective web based learning. People often find it difficult to engage in web learning or effective discussions when they are physically apart. Modern Applications that allow online chatting via voice, audio or video only bridges this gap to a certain extent. It is not possible to explain diagrams and drawings unless a transfer mode where the actual drawing can be recreated is established.

As a solution to this small but relevant problem, we came up with the application Asynchronous canvas sharing which acts as an effective channel for web based learning. This application enables multiple online users to share a single canvas (drawing board) in real time inside a browser and changes made by one user is instantaneously visible to the other users. A chat room is included to enable interaction between users. This application can be used for online teaching, web conferencing and engineering education. Using this service, people can collaborate easily to develop designs and other graphic drawings.

Chapter 3

Design

After doing considerable research to find a mechanism that suits real-time web applications, the design for the application has been modeled on Client-Server architecture. As is given in the Sequence diagram (fig 2), session management is designed using a client-side interface and a server-side database. The client can either start a session with the server or join an existing session. The model designed to handle data transfer between the clients and the server is specified in fig 1. Here, in this system, in the event of any change, the client sends an update to the server (using HTTP methods) and the server notifies other clients that are alive in the same session.

We chose to render our drawings using vector graphics with the help of geometrical primitives like point, line, circle etc. rather than using a raster bitmap, so as to make data transfer efficient.

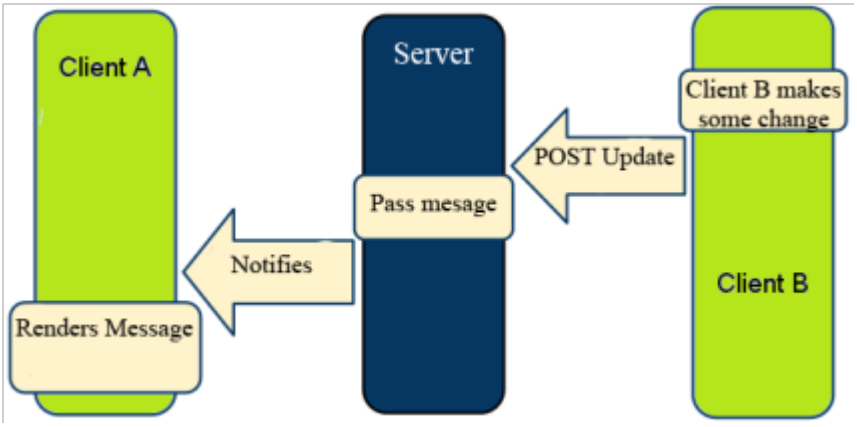

Fig 1: Communication model for the application.

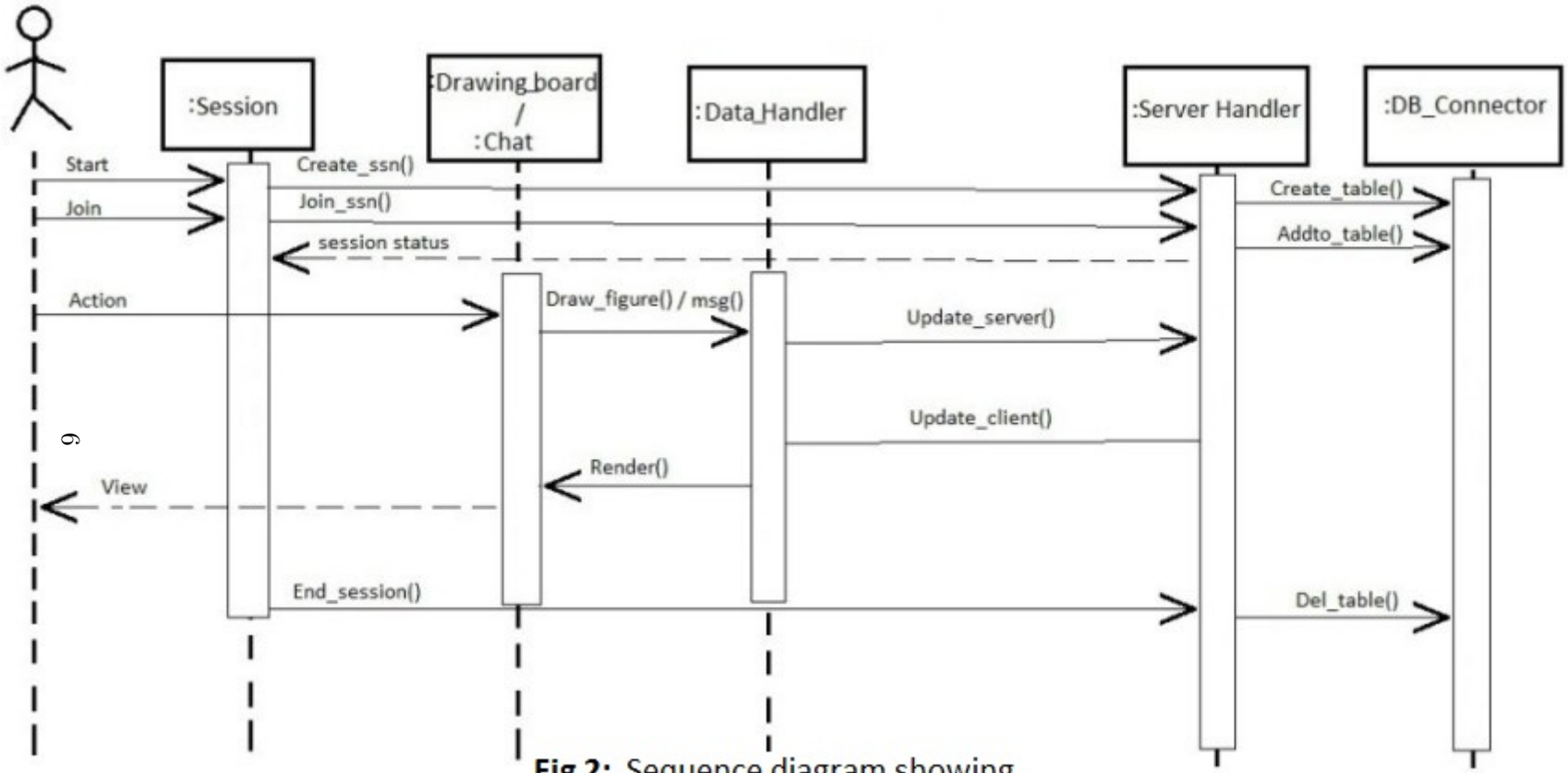

Fig 2: Sequence diagram showing client server interaction

The program is designed as a multiuser, multisession web application where the user can either create or join a session. The user will be required to enter a name and unique number as sessionID for creating a new session and others can join existing sessions with their respective sessionIDs.

A drawing board (initially white) is placed, where the user will be able to draw or design figues. The following tools fall under the design. They appear on the left hand side of the window. The tools implemented are enlisted below.

1. Pencil
2. Line
3. Rectangle
4. Circle
5. Ellipse
6. Whitener
7. Undo
8. Redo
9. Reset
10. Download
11. Stroke-width selector
12. Color-palette

A chat container is located at right side adjacent to the drawing board, which facilitates chat between users.

Chapter 4

Implementation and Results

4.1 Login Interface

The GUI has been developed as shown in fig 3. A user can either begin a session or join a session. When the user clicks either of the options, he/she is asked for a session ID and a username.

- Begin session In order to begin a session, the user has to enter a unique session ID. If the session ID already exists , an error is displayed. A new table is created in the database with the session ID as the name. The new user is added to the table 'users' along with the session ID.
- Join session In order to join a session, the user has to enter a unique username along with the ID of an already existing session. If the session ID doesnt exists , an error is displayed. If the username already exists, an error is displayed. The joining users' names are added along with their session IDs to the table 'users'.

4.2 Workspace

The GUI has been designed with the necessary functional components as shown in fig 4.

- **Drawing tools**

  The drawing tools were created using JavaScript and jQuery.[8] They utilize properties of Scalar Vector Graphics (SVG).[1] The drawing tools modify the underlying code of SVG to create/edit drawings on the board.[6] When a tool is selected, its name is stored into a global variable. When a person starts drawing with the tool, an element is created with this name and its attributes are modified. The tools are placed on the left hand side of the window. The tools implemented are enlisted below.

1. **Pencil-**
It is a free-hand drawing tool which creates a continuous series of points in any direction exactly same as the points traversed by the mouse pointer. This tool is implemented using the <path> element.

2. **Line-**
When a user clicks the mouse at a point with this tool and drags the pointer to another point, a straight line is created between the two points. This tool is implemented using the <line> element.

3. **Rectangle-**
This tool creates a <rect> element which is axis aligned with the current coordinate system. The difference of X and Y coordinates before and after a mouse drag is analysed and a rectangle is created with the resultant values.

4. **Circle-**
A circle tool creates a circle based on a given center point and a radius communicated by a mouse drag. This tool is implemented using <ellipse> element.

5. **Ellipse-**
Using this tool, a user tries to draw an ellipse using free hand which will be converted automatically to an ellipse. The minimum and maximum values of X and Y coordinates are analysed to find the center of the possible ellipse and is drawn using the <ellipse> element.

6. **Whitener-**
The whitener tool overwrites the points through which it is passed, with white colour. It is implemented using <path> element.

7. **Undo-**
This tool performs the regular undo function. The <SVG> element is traversed to find its last child node (element). This node is removed from the DOM tree and is pushed into a stack to perform 'redo' at a later stage if required.

8. **Redo-**
This tool performs the regular redo function. The last node in the stack is popped and is appended into the DOM tree.

9. **Reset-**
This tool removes all the child nodes of the DOM tree.

10. **Download-**
This tool converts the current SVG element into a Base64-encoded data URI image which can be downloaded and saved.[7]

11. **Stroke-width selector**
In the previous work, all shapes were created with a stroke width of

1 px, but now there is added options to select the stroke width. It is achieved by manipulating the width attribute of the svg element.

12. **Color-palette**
Our previous implementation only supported drawing in a single colour (black). We have now implemented a colour palette to facilitate drawing in many colours. It is achieved by manipulating the color attribute of the svg element. The palette contains 8 colors - black, red, blue, green, grey, magenta, cyan, yellow.

- **Drawing Board**

  Drawing board is initially a fully white SVG element. SVG includes various predefined events which pass the details of the specific event to the assigned event-handling functions. Using XML parsing techniques we create and modify the underlying code with the help of event handlers. As shown in fig 3, the drawing board is located to the right hand side of the tools pallette. We used the following event handlers.

  1. **onLoad**:
  This event handler is triggered when the blank SVG file is loaded into the HTML file. The initial SVG element is converted into a DOM object and is stored in a variable.

  2. **onMouseDown**:
  This event handler is triggered when the user presses a mouse button. On this event the element corresponding to the currently selected tool is created and appended into the DOM tree.

  3. **onMouseMove**:
  This event handler is triggered when the user moves the mouse pointer. On this event and if the mouse is pressed down the SVG content is modified/updated in the DOM tree and transferred to the server.

  4. **onMouseUp**:
  This event is triggered when the user releases the mouse button. On this event the final values of the attributes are set.

  The current X and Y coordinate are passed to the handler function during onMouseDown, onMouseMove, onMouseUp events. These coordinate values are used to set the attributes of the elements during their creation and modification.

- **Online users** The SQL table 'users' contains the name of the users along with their session IDs. The names of the users present in a session are retrieved from this table via an AJAX request and are displayed on this container dynamically. To avoid unnecessary network congestion, we use COMET connection which returns the request only when a change occurs on the number of users online.[5]

- **Chat container**

  When a user starts a session, a SQL table with the session ID is created. It stores the entries of the users of the session along with the timestamp, textid and texttype. The timestamp is a datastructure that holds the date and time of the entry. Textid is an integer primary key that auto increments upon each message entry.[9] The entries of the table can either be a command or a chat data which are differentiated using 'textype'. '0' corresponds to logout command. '1' corresponds to chat data and '2' corresponds to login command. The entries are retrieved from this table via an AJAX request and are displayed on this container dynamically. To avoid unnecessary network congestion, we use COMET connection which returns the request only when the table is updated.

4.3 Data Transfer

Data transfer functionality is logically divided into two units one to update the server and the other to update the client. We created the asynchronous data transfer required to simulate realtime interactivity using a technique known as COMET.[5]

- **Client updating the server**

  The handler function assigned to the onMouseMove and onMouseUp events sends an AJAX POST request to the server whenever a change occurs in the drawing board (client).[2] The requests contain, as payload, commands to modify the svg code to reflect only the changes produced on the canvas and not the whole svg code itself. These commands are create, modify, undo, redo and reset. On receiving this, the server updates the data store. The handler for onMouseMove event sends requests at an interval of 300ms to facilitate synchronization of clients‘ drawing boards even before an element is completely drawn.

- **Client updating itself**

  Once a client updates the server data store as explained above, the other clients need to update itself. In our implementation, the client gives a AJAX POST request to the server sending the last modified time of the datastore(initially 0) time as payload.[2] After the first POST request, each client gets the last modified time from the server. This value is circulated back to the server to make a decision when to send the modified content. On successful receipt of response, the received data is added to the DOM tree and this AJAX request method is recursed. So we make sure that the request is continuously sent.

On server side, the last modified time of the SVG filestore is compared with the value in the payload (which is the time upto which client has been updated) received from the client. Since the value of time in the request, if there has been no modification to the data store, the request is put in a loop and made to wait. When the data store is modified by some other client, the control breaks out of the loop and the current change in code in the data store is sent to the client. This is our implementation of the web application model of COMET.[5]

4.4 Concurrency

We have solved the issue of concurrency while drawing by transfering to the server only the SVG elements which are modified or newly created. Whenever the SVG code in the client's drawing board changes, the corresponding modified elements are sent to the server along with a command to notify server whether to append or to replace an element. When two different users make modifications to the drawing board, they are in effect modifying different elements of the svg code. The server modifies them as different independent transactions. The inbuilt requeset handling system of the mySQL database server queues the incoming request and processes them one by one only. The client's browser updates the SVG code displayed to the user by inserting the received content in the backend.

Fig 3: Login interface.

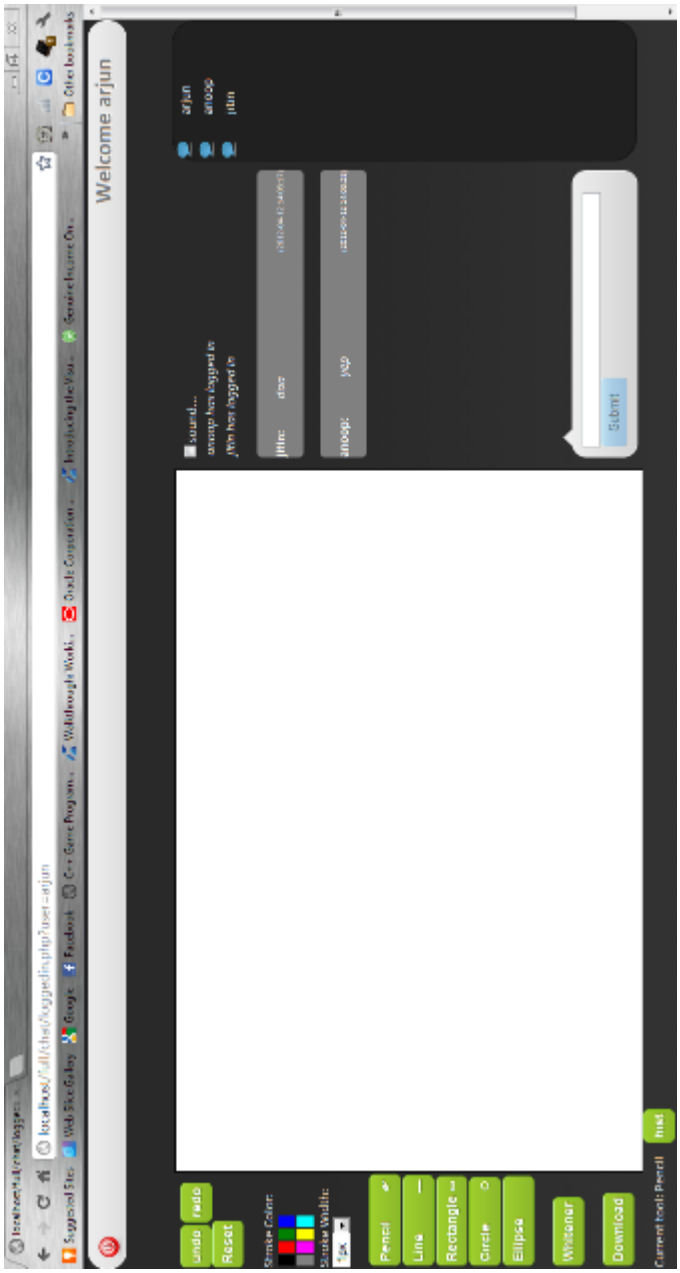

Fig 4: Workspace.

4.5 Sessions

Sessions and support for multiple drawing rooms have now been implemented in our work. Sessions have been created using a predetermined id (session ID) decided by the creating user and managed using a database.

The server now accesses the session ID and user ID from session variables when data is received via the AJAX requests for storing the modififcations. On creating a session, two new database tables, one to store the svg data and the other to store chat lines for that particular session, are created. When all users of a session log out, the table associated with that particular session is deleted.

4.6 Data Storage

The database is used to store the session information and list the users currently active in the session. The username and the session ID is stored in the table 'users'. The SVG content from different sessions is stored in different tables with each table given the concatenation of session ID and an underscore and the string 'file' as name containing the name of the creator of the element, its shape, number, id and flag. Each session of the chat has an associated database table with session ID as its name, containing the author, chatline,its timestamp, textid and texttype. The database also records when a new member has joined the session and the application subsequently notifies all users in the session. Similarly, when a person logs out, every user in that session is notified.

Chapter 5

Conclusion

The team has developed a quite efficient application based on the orignial design of the project. Although there is room for improvement, the model has potential to lead to a standard for inter-operability between shared graphics editors that would allow users to develop and collaborate different applications, and will be able to draw together.

On load testing done using Apache bench on a test server (1.5 GHz and 4 GB RAM), the application was found to handle a maximum of around 256 concurrent connections in accordance with the default maximum value set in apache. This are obtained under test parameters of 1000 requests with a concurrency level of 256.

The approach followed in this project differs significantly from the original model of the world wide web, in which a web browser requests a resource completely at a time. Here we circumvent the limitations of the page-by-page web model and traditional polling by offering real-time interaction, using a persistent or long-lasting connection and multiple sessions.

Bibliography

[1] Pearlman, Ellen; House, Lorien; *SVG for Web developers*, Prentice Hall Professional, 2003

[2] Matthijssen, N.; Zaidman, A.; Storey, M.-A.; Bull, I.; van Deursen, A.; *Connecting Traces: Understanding Client-Server Interactions in Ajax Applications*

[3] S. M. H. Collin; *Setting up a Web server*, Elsevier, 1997

[4] Lambert M. Surhone; *WebSockets: Transmission Control Protocol, Push Technology, Internet Mail Standard*, Betascript Publ., 2010

[5] Crane, Dave; McCarthy, Phil; *Comet and Reverse Ajax: The Next Generation Ajax 2.0*, Apress, 2008

[6] Scalable Vector Graphics (SVG) 1.0 Specification; *http://www.w3.org/2000/svg*

[7] Base64 URI- RFC 2397 - The data URL scheme. Internet Engineering Task Force; *http://tools.ietf.org/html/rfc2397*

[8] jQuery documentation *http://www.jquery.com*

[9] Schneider; *MySQL Database Design and Tuning*, Pearson Education India, 2005

[10] Chin-Wan Chung; *Web and Communication Technologies and Internet-related Social Issues*, Springer, 2003