

Georgia State University

ScholarWorks @ Georgia State University

Computer Science Theses

Department of Computer Science

4-21-2009

A Browser-Based Collaborative Multimedia Messaging System

Susan Gayle Gentner

Follow this and additional works at: https://scholarworks.gsu.edu/cs_theses



Part of the [Computer Sciences Commons](#)

Recommended Citation

Gentner, Susan Gayle, "A Browser-Based Collaborative Multimedia Messaging System." Thesis, Georgia State University, 2009.

doi: <https://doi.org/10.57709/1059408>

This Thesis is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Theses by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

A BROWSER-BASED COLLABORATIVE MULTIMEDIA MESSAGING SYSTEM

by

SUSAN GENTNER

Under the Direction of Xiaolin Hu

ABSTRACT

Making a communication tool easier for people to operate can have profound and positive effects on its popularity and on the users themselves. This thesis is about making it easier for people to publish web-based documents that have sound, video and text. Readily available software and hardware are employed in an attempt to achieve the goal of providing a software service that enables users to compose audio-video documents with text.

INDEX WORDS: Applet, Communication tools, Internet, Java, Software as a service, Multimedia, Web Cam, Web Service

A BROWSER-BASED COLLABORATIVE MULTIMEDIA MESSAGING SYSTEM

by

SUSAN GENTNER

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree
of

Master of Science

in the College of Arts and Sciences

Georgia State University

2009

Copyright by
Susan Gentner
2009

A BROWSER-BASED COLLABORATIVE MULTIMEDIA MESSAGING SYSTEM

by

SUSAN GENTNER

Committee Chair: Xiaolin Hu

Committee: Anu Bourgeois
Michael Weeks

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
May 2009

DEDICATION

This goes out to my aunts Barbara and Martha, whose warm and steady love gives me strength even when I am lonely in my search to build and learn.

ACKNOWLEDGEMENTS

My husband Matt gave me guidance and love that has gotten me to this point in my career. He is always there to lend a helping hand, to give me encouragement when I need it, and to look out for me even when I am stubborn.

Next, my advisor and committee, Dr. Hu, Dr. Bourgeois, and Dr. Weeks. It goes without saying they taught me much but they also inspired me to learn and to seek what I was interested in.

Last, but certainly not least, are my best friends and fellow academics Stefanie Markham and Mary Hudacheck-Buswell. They help me in numerous simple ways like going to lunch and commiserating with me about various topics. They have and always will hold a special place in my life.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	v
LIST OF FIGURES.....	ix
1. INTRODUCTION.....	1
1.1 Background.....	1
1.1.1 Participation Grows.....	1
1.1.2 An Aesthetic Mosaic.....	1
1.2 Problem Definition.....	4
1.2.1 Awkward Composition Process.....	5
1.2.2 Clogged Transport Medium.....	6
1.2.3 Configuration Mismanagement.....	6
1.3 Alternative Methods	8
1.3.1 Publishing Beats Sending.....	8
1.3.2 No System is Perfect.....	9
2. APPROACH AND CHALLENGES.....	11
2.1 Approach.....	11
2.2 Challenges.....	13
2.3 Concepts.....	16
2.4 Related Technologies.....	17
2.4.1 Webmail.....	17
2.4.2 Transport Layer Security.....	18

2.4.3 Spam Control.....	20
2.4.4 Automatic Updates.....	21
2.5 Related Work.....	22
2.5.1 Multimedia.....	22
2.5.2 Collaboration.....	23
2.5.3 Architecture.....	24
3. SYSTEM DESIGN AND ARCHITECTURE.....	27
3.1 Design Overview.....	27
3.2 Architecture.....	28
4. DESCRIPTION OF PROTOTYPE.....	32
4.1 Send a Moment.....	32
4.1.1 Capture the Video Portion.....	33
4.1.2 Compose the Moment.....	33
4.1.3 Send the Moment.....	33
4.2 Receive a Moment.....	34
4.2.1 Select a Moment.....	34
4.2.2 View a Moment.....	35
4.2.3 Save a Moment.....	36
4.3 Environment and Hardware Notes.....	36
4.3.1 Mac Mini.....	37
4.3.2 Logitech UVC Webcam.....	38
5. CONCLUSION AND FUTURE WORK.....	41

5.1 Future Work.....	41
5.2 Development Environment.....	42
BIBLIOGRAPHY.....	48
APPENDIX: SOURCE CODE LISTING.....	52

LIST OF FIGURES

Figure 1: Firefox peeks from beneath the text-only Lynx web browser.....	2
Figure 2: A graph of early web usage data from [1].....	3
Figure 3: Something is missing on Bob's end.....	7
Figure 4: Viewer in my MomentShare desktop application from 2006.....	13
Figure 5: Browser-based moment viewer with better video in 2009.....	14
Figure 6: A browser warning about an untrusted certificate signature.....	19
Figure 7: PushToShow has a three-tier architecture.....	27
Figure 8: Class Diagram of the Client Side.....	29
Figure 9: Web Service Classes.....	30
Figure 10: Record moment sequence diagram.....	30
Figure 11: Send moment sequence diagram.....	31
Figure 12: The ShareMoment web page.....	32
Figure 13: The ListMoments web page	34
Figure 14: The TakeMoment web page	35
Figure 15: A PushToShow workstation.....	37
Figure 16: Some cameras did not perform as advertised.....	39
Figure 17: PushToShow achieves high video quality.....	40
Figure 18: The pgAdmin3 database management tool.....	43
Figure 19: The NetBeans IDE.....	44
Figure 20: Applet class diagram.....	52

Figure 21: CapturePanel class.....54

Figure 22: FileUploader class.....58

Figure 23: Moment class.....60

Figure 24: MomentUploader class.....63

Figure 25: MomentUploadSvc class.....65

Figure 26: VideoUploadSvc class.....68

Figure 27: WebCam class.....71

1. INTRODUCTION

This thesis is about making it easier for people to publish web-based documents that have sound, video and text.

1.1 Background

The main motivation for this work is that making a communication tool easier for people to operate can have profound and positive effects on its popularity and on the users themselves. A more elegant tool that requires fewer steps to use is easier to learn and will return some emphasis from limits of technology to the vastness of users' creativity.

1.1.1 Participation Grows

Sometimes a little difference can mean a lot. When web pages could be viewed with stylized text flowing around colorful in-line images, Internet content began to grow exponentially and the number of web pages doubled every three months. And the same functional power of publishing and linking live documents from all over the Earth that was still there.

1.1.2 An Aesthetic Mosaic

To users in 1993, the Mosaic graphical web browser made it much easier on their eyes to read web pages than the text-based Lynx web browser, created just one year earlier. Mosaic was received like a dazzling new theater in

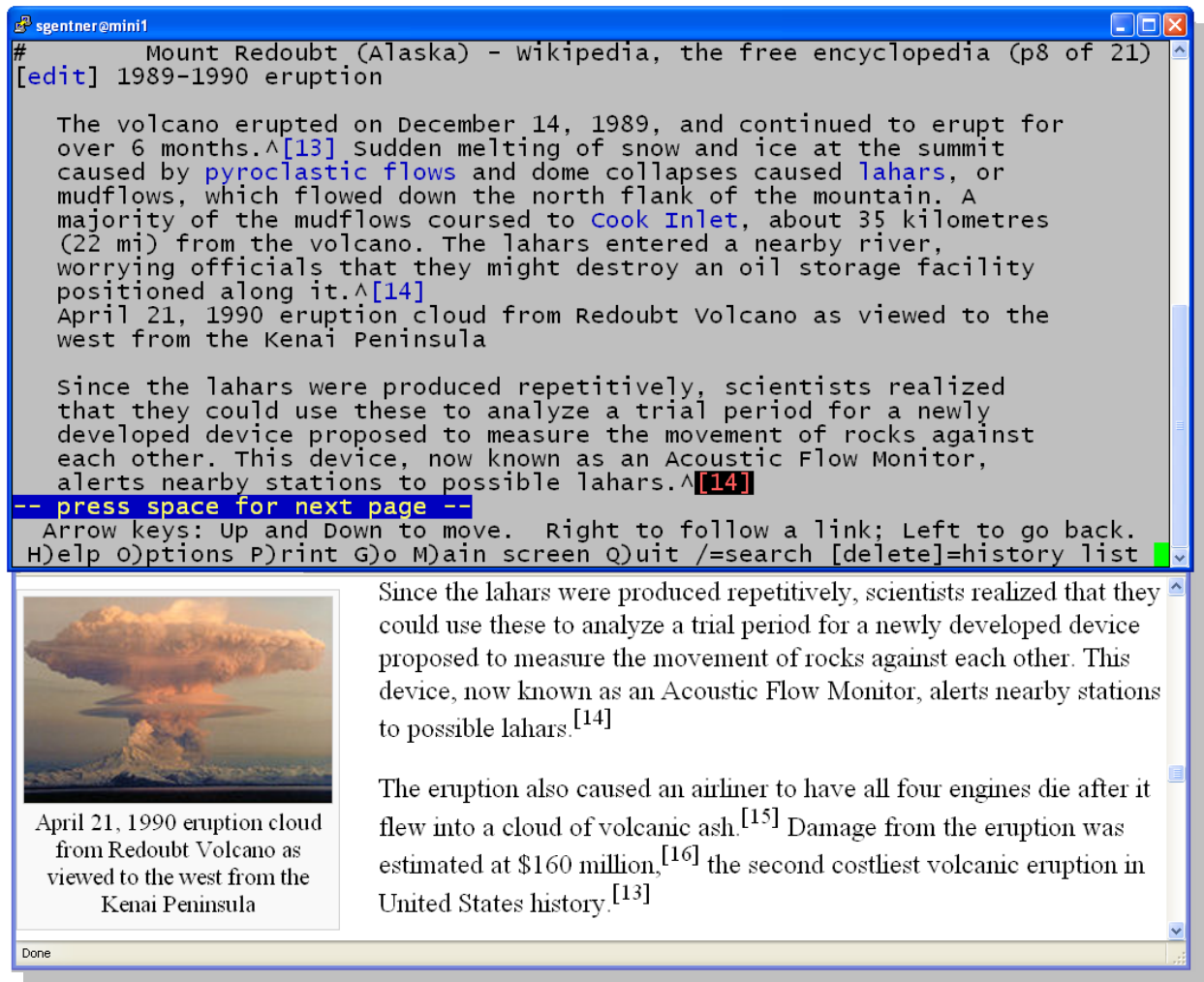


Figure 1: Firefox peeks from beneath the text-only Lynx web browser

towns and cities across the globe. Its HTML audience grew wildly, and, as more was given to users, they frequently got the itch to give back. Figure 1 illustrates the visual differences between a text-only and graphical web browsers. And figure 2 shows the increase in HTTP traffic of web pages after the introduction of the graphical Mosaic web browser in 1993 [1].

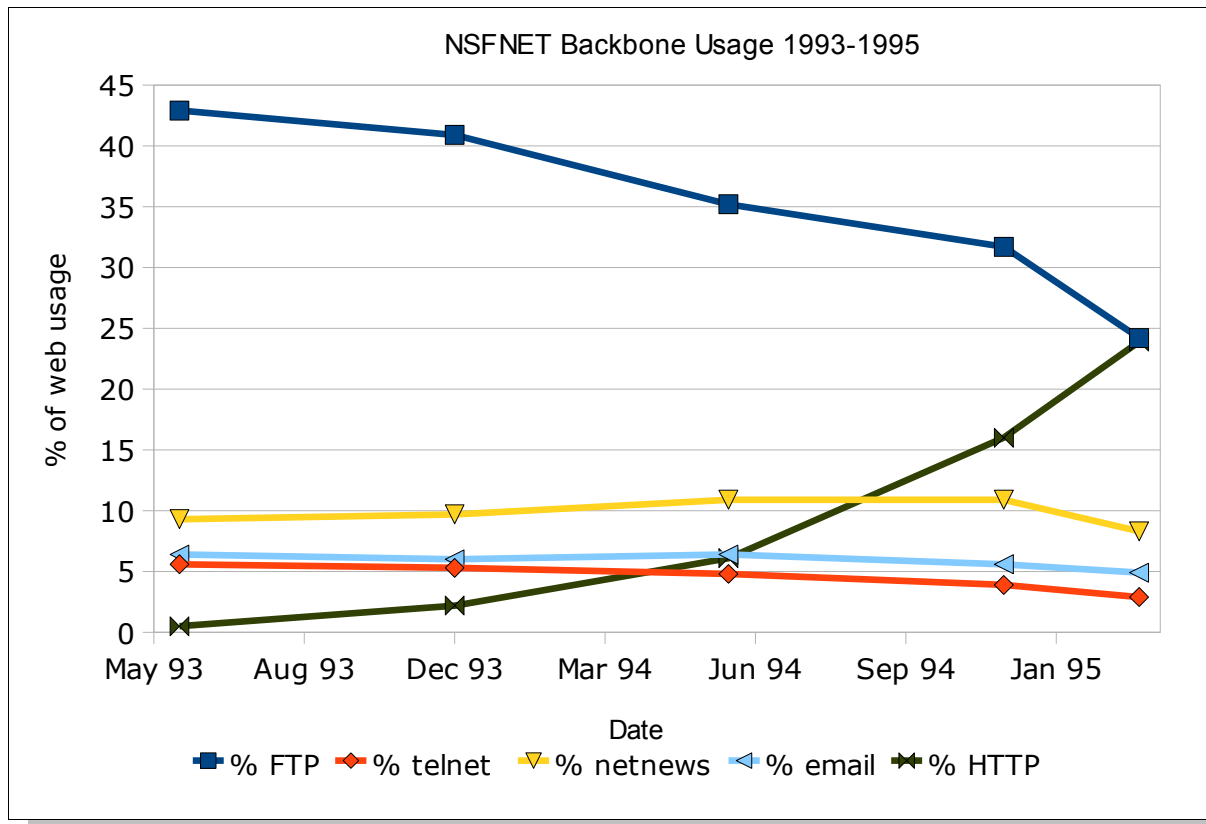


Figure 2: A graph of early web usage data from [1].

In retrospect, the move from text-only to graphical web browsers seems like an obvious change that might also have been an easy change. For a dozen years prior, Tim Berners-Lee and countless others had spent millions of hours (and billions of dollars) to do the “heavy lifting” while building the world wide web. In American history, it is the railroads' and interstate highways' completions that we still celebrate today, more than specific models of locomotives and automobiles.

In figure 2, I have graphed some per-protocol traffic measurements slightly before and after the debut of Mosaic. It is probably for the simple reason that Mosaic could graphically display the multimedia of web pages that the audience for hypertext transfer protocol (HTTP) began to grow, usurping the file transfer protocol (FTP) in early 1995 as the most-used category of Internet traffic. Users *really* appreciate elegance in their tools, and Mosaic changed the way that most people worked on the Internet.

The graphical Mosaic browser was built by two guys in just six months; just two! [2] That does not seem to be enough time to create a single folk song like those sung by the teams that were working on the railroad. And yet their achievement is stunning nonetheless. When Marc Andreessen and Eric Bina created Mosaic, they brought a tremendous amount of new technology to bear fruit from a simple vision. The small dare between Marc and Eric produced an elegant tool that made such a big and positive difference. It would be thrilling for me to achieve one millionth of the public service that Marc Andreessen and Eric Bina brought to us, by creating Mosaic.

1.2 Problem Definition

Part of the challenge here is seeking an “ideal” elegance, similar to reducing a complex and verbose mathematical proof. Often, the first impression that a user gets with document creation software is a lengthy installation process

that consumes part of their computing device. And after a customary reboot, the user braces herself to descend into a new “confusopoly” [3] that she has just committed to, and begin the complex and esoteric process of creating her first multimedia document. The logistics involved in sharing her art with its intended recipients are another unsavory challenge, but she keeps her mind clear to focus on what *should* be simple: creating a message that has some video and text.

1.2.1 Awkward Composition Process

When Alice bought her laptop, she was thinking ahead and she picked a model that included a built-in web cam. Her friend Bob was stationed in Iraq then, and now Bob is fighting in Afghanistan. Since the camera was already attached and staring back at her, Alice assumed that there would be some easy software included that would help her to share a moment with Bob that was comprised of some sound, video, and text. She wanted to send something personal just for Bob, that would remind him about the great home he had to return to someday.

For her Microsoft Windows-based laptop, Alice found a free Microsoft download for something called MovieMaker that helped her to make a fun 15-minute home movie. The movie was ready and Alice was proud of it, so she attached it to an e-mail message and sent it off to Bob. Instead of any

thanks from Bob on the next day, Alice found a cryptic error message that indicated her home-movie e-mail message to Bob had been rejected because it “exceeded his incoming e-mail message size limit(s).”

1.2.2 Clogged Transport Medium

Luckily for Alice, the MovieMaker tool had video editing features that she learned to use to pare down her home movie to just the juicy highlights. The movie for Bob was a small fraction of its original length and size, and Alice used zip compression to shrink the movie another 12% before attempting to re-send it as an e-mail attachment to Bob.

This time Bob received the message from Alice. He called Alice on the phone while her message, bigger than most, was being retrieved from the e-mail server. Bob apologized that he would need to delete the message from his inbox after it was safely on his Mac, because he needed inbox space to receive other e-mail that might come.

1.2.3 Configuration Mismanagement

Bob was an Army man, good at following instructions. So he quickly and carefully followed the instructions in Alice's e-mail to:

1. save the attached movie,
2. note its location on his file system,

3. unzip the compressed file to get at the real movie that Alice had sent,
4. and delete the zipped clutter to save some disk space.

Now Bob was starting to feel some suspense because he had been growing to like Alice more during his lonely missions overseas. It made Bob feel great that Alice had sent this special treat.

Bob was curious and uncertain as to what he would see, but he was positively certain that he would enjoy the home movie from Alice and be watching it frequently until she sent the next movie. When the file was done being decompressed, Bob used Finder on his Mac PowerBook to Open the movie file. Yet instead of Alice's smiling face, a terse error message appeared that read:

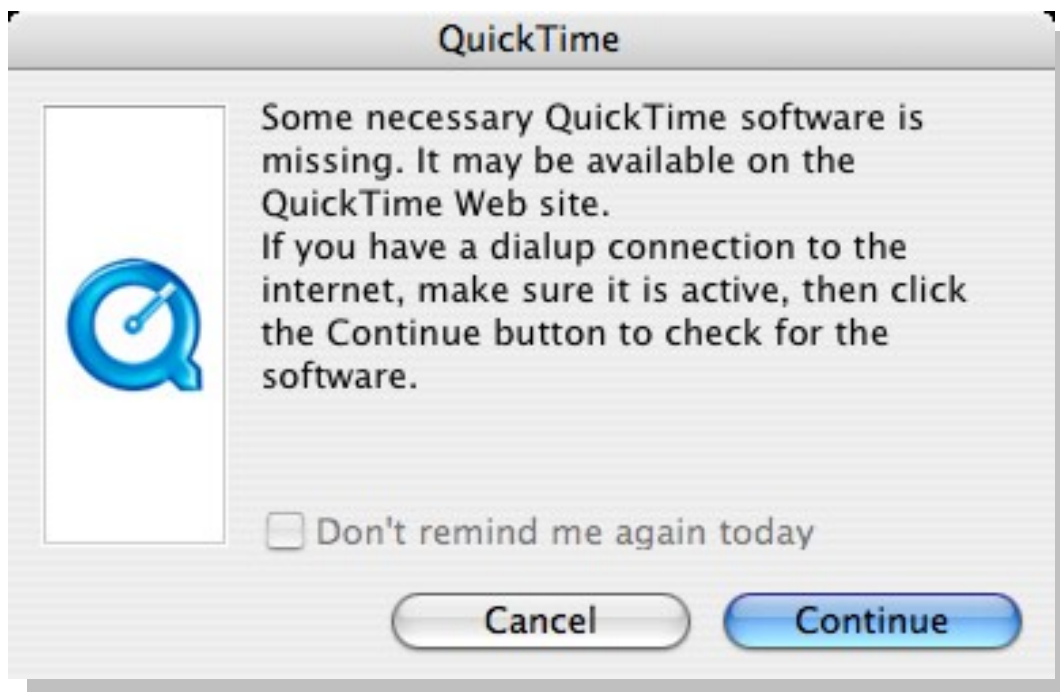


Figure 3: Something is missing on Bob's end

1.3 Alternative Methods

Soon Alice was back to square #1 with a different approach. She would make a third movie, this time less personal. And she would revise the written note to be less personal too. Then Alice would upload the movie with the message to YouTube and e-mail the URL to Bob.

Alice and Bob were grateful that this YouTube approach worked pretty well. Alice did not get everything she originally wanted in her movie to Bob, and it was still awkward and time-consuming to capture the video, edit a movie and upload her final cut. But YouTube had helped to solve the codec and transport problems they had had at first, so Alice and Bob could feel better connected despite being apart.

1.3.1 Publishing Beats Sending

The publishing and distribution method seemed cohesive, and the codec mismatch problem was solved in one fell swoop. And, assuming the sender already had a video capture tool installed, there was no required software except for a Flash-enabled web browser. All Alice needed to do was to register for a YouTube account and use her web browser to upload her video moment, complete with sound and with text. This software as a service (SaaS) freed Alice and Bob from some obstacles that are side-effects of proprietary software. But, was this a trade-off into other forms of lock-in?

Although the original intended audience was just Bob, Alice needed to edit her home video because YouTube does not have fine-grained security and most video is more or less public. Once Alice uploads her home movie, it becomes transcoded into a proprietary Adobe Flash codec and *that* version is held as content owned by YouTube. It is likely that keyword matching is done by the parent company Google to arrange advertisements in and around Alice's home movie. So Bob sees a few sidebar gift ideas for Alice along with the moment that she originally intended to share.

1.3.2 No System is Perfect

Being centralized and proprietary were key business aspects to protecting the investors first and then later Google as the parent company of YouTube. This seems prudent and fair because YouTube cost hundreds of millions of dollars to launch, *plus* about a million dollars per day for bandwidth costs. After 18 months, Google announced that it had bought YouTube for \$1.65 billion- which serves as a ballpark figure of the startup cost for the new video sharing website. But when founders of YouTube Chad Hurley, Steve Chen and Jawed Karim launched their website in February of 2005, they eliminated most of the obstacles that Alice and Bob had encountered.

Part of the motivation of this work is to explore the feasibility of restoring privacy and freedom to a medium that is similar to YouTube in convenience

to users, but less encumbered with proprietary modification and ownership restrictions. Rather than an ending, I see that there has been an exciting new opportunity for building helpful software. Instead of a “killer” application that might lock-out developers, I see a fascinating new range of ways that I might be able to serve users by helping to make it quicker and easier to publish their new creations.

The central question of this thesis is whether it is feasible to make it easier for people to record and exchange moments with current commodity personal computing hardware. To answer this, in the next chapters I describe my understanding of the technical challenges. Then I detail an approach to meet these challenges with a system design and an implementation that demonstrate success for the most difficult challenges.

My PushToShow prototype is included as an original part of this thesis work, and this prototype will prove that browser-based web pages today can “see” and “hear” their end-user's scenario. This state of the art in web pages may be easily and precisely controlled by casual users to capture and share important moments from their time and space with others.

2. APPROACH AND CHALLENGES

The approach that I took was to build a prototype that could be used to effectively demonstrate my concept of an easy tool for people to create messages with sound, video and text.

2.1 Approach

This thesis has an emphasis on development because Murphy's Law seems to be the *only* law when it comes to building a browser-based multimedia creation and exchange prototype. Without trying some techniques by building and testing bits of software together, I anticipated that my planning and research would seem to have little meaning because, as Murphy states, "Whatever can go wrong WILL go wrong." So I set to work to build a working prototype with four main attributes:

1. ease of use
2. browser-based
3. great video quality
4. use free or cheap components

Ideally, I hope that my prototype will grow to a stage where it may be adopted for incubation into an open source project that students and volunteers will enjoy benefiting-from and contributing-to. So I have tried to use free or cheap hardware and software components. The ease of use and

browser-based goals were intended to greatly help me in demonstrating the prototype, which itself is meant to promote the feasibility of concepts for improvement in this thesis. And the browser-based goal also brings a tacit advantage in that the prototype will be a web application; that there will be no lengthy installation for end users, and that it may be extensible with standard off-the-shelf security enhancements.

My skepticism for planning too far out (without a prototype) comes from a term project that I built three years ago during a course called "Digital Signal Processing" taught by Dr. Michael Weeks. In a way, that term project was a prototype for *this* prototype (figures 4 and 5 respectively); at that time called "MomentShare." It succeeded in showing that it is feasible to build a Java desktop application that will capture from a web cam and help a user to send the "moment" with text over an e-mail transport. And it showed that there were two big areas of improvement:

1. browser-based instead of desktop installation
2. web publishing instead of message sending

In the time since I built my MomentShare prototype, an unrelated website for sharing family photos [4] has been established on the Internet. So although I still refer to my new message type as a "moment," I refer to my new browser-based prototype as PushToShow.

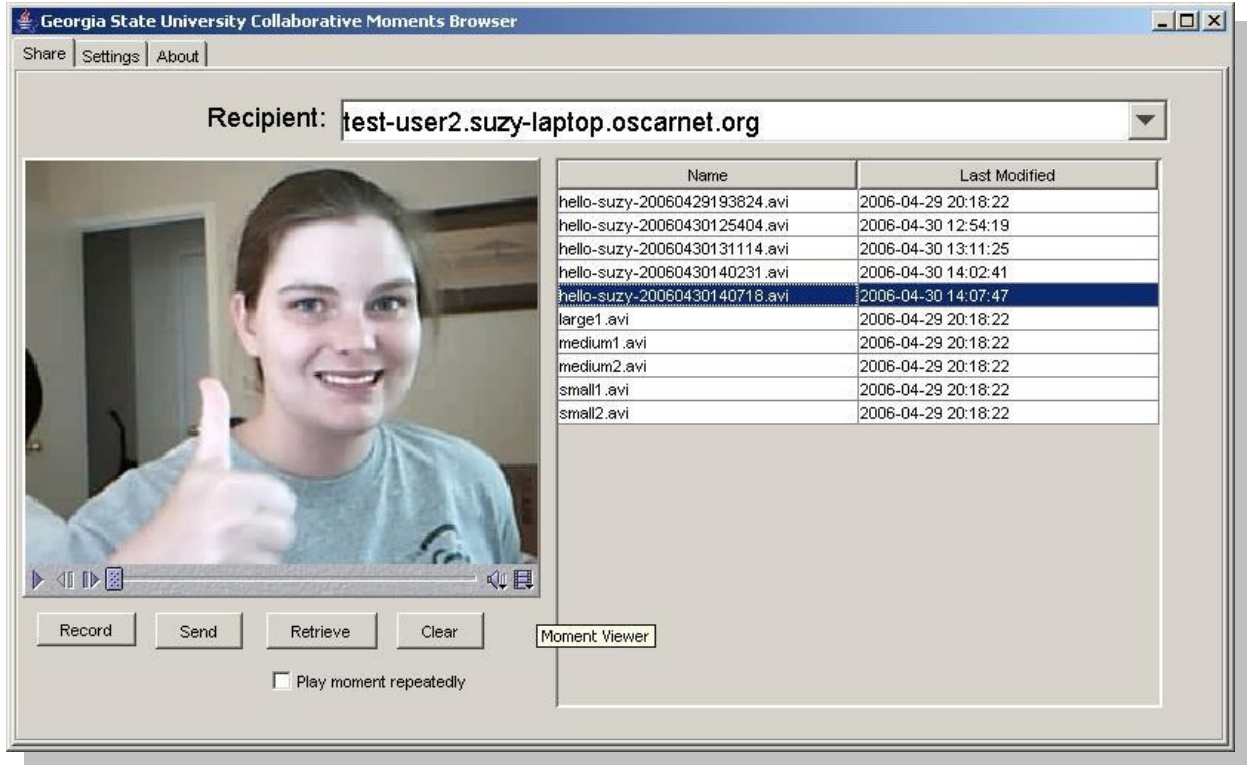


Figure 4: Viewer in my MomentShare desktop application from 2006

2.2 Challenges

The goal of great video quality comes from the typical “how fast can it go?” curiosity that I share with other software developers. But it also stems from a probable future requirement that users will want their moments to look good, even when the video portion of each moment is presented in full screen mode. I wanted the moments to have a fast frame rate that keeps recorded motions fluid and with a high resolution that approaches conventional NTSC analog TV ([640x480@25hz](#)) to keep image shapes crisp and clear. So during a summertime of trial and error I was able to find a configuration that captured 15 frames per second at VGA resolution

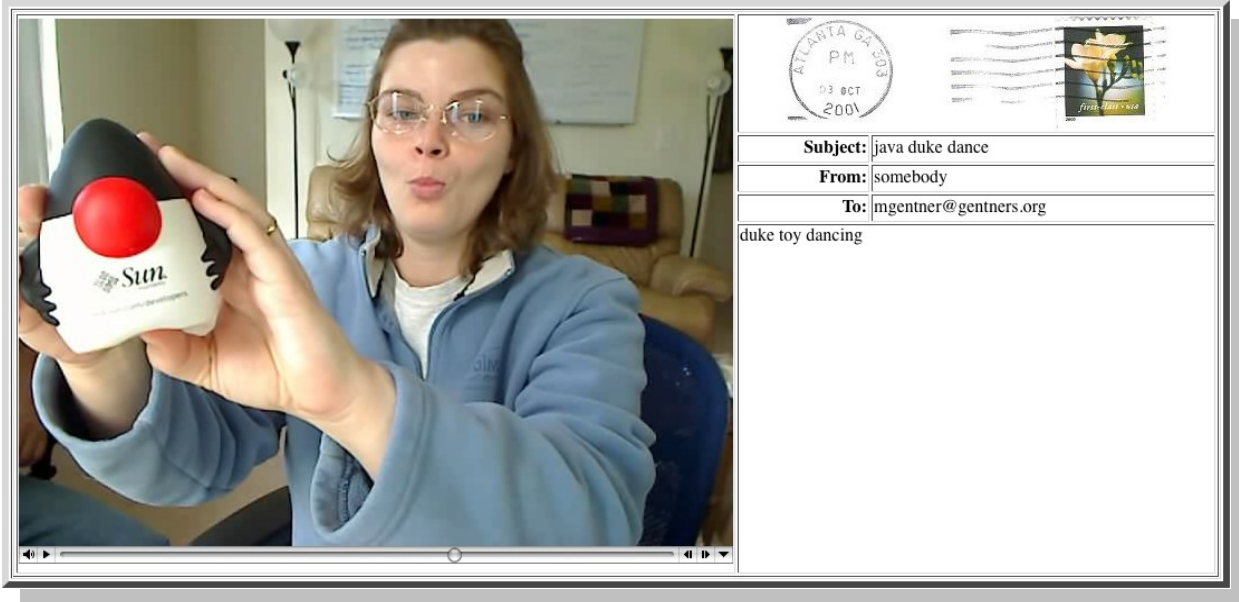


Figure 5: Browser-based moment viewer with better video in 2009

(640x480@15hz) and still keep CPU utilization down to 25% and affordable compressed bit-rate bandwidth of barely 1 megabit per second.

I found that there are copious quantities of published research in a myriad of topics that pertain to the “signs of aging” that e-mail is exhibiting [5] [6].

The main problems seem to be related to lack of security. For decades now, billions of people have learned first-hand that it is too easy for attackers to get fixed e-mail addresses and that leaving the door open to one's mailbox is not a sustainable option.

Billions of people have also been suffering unsolicited postal mail for decades as well. And although their postal addresses are fixed, there seem to be far fewer concerns about security of people's postal mailboxes.

If we try to compare an e-mail inbox to a U.S. Postal mailbox, then senders need to pay postage for each postal message and federal law prohibits anyone other than an official mail carrier to put things into someone's mailbox. Although recipients often disdain the deluge junk mail received, their suffering would be much worse if it were free to send junk mail. Here, another small difference means a lot: a small 20-cent bulk mail rate IS a cost, and this cost makes senders more eloquent. If the public were allowed to physically deposit whatever we liked into people's mailboxes for free, then sadly the term "*junk* mail" would take on greater scale and a whole new meaning. But that is the case with e-mail. After senders get an e-mail address, they may attempt to send whatever they like; directly to the recipient and without any postage paid.

To attach video files to this antiquated medium would resemble the "pig in a python" metaphor and make a bad situation even worse. Even with compression, video files are much bigger than today's typical e-mail message, by two or three orders of magnitude. Here web-mail and RSS technologies have helped to show that, at a minimum, the user must be given a chance to see the sender's identification, the subject line, and

message data prior to requesting the rest of the message. And, that when it comes to giving the user some features to inspect attachments prior to downloading them: the *more*, the merrier.

2.3 Concepts

Today we have so many new multimedia tools! There are person to person aids that range from instant-messaging clients to VoIP and video streaming tools including AIM, Skype, and Ekiga. Audio-enabled remote desktop sharing tools such as Microsoft Remote Desktop (RDP) Virtual Network Computing (VNC) and LiveMeeting. And also we have browser-based collaborative tools such as YouTube and Wiki knowledge bases.

If we pick one or more of these tools for substitution, then some valid questions might take the form "why not use *X* instead of PushToShow?" So here are some attempts to answer those permutations with some hasty generalizations. Instant messengers are interpersonal with an emphasis on *text*-based exchange, and media other-than text are handled as attachments. VoIP is interpersonal and session-based, with an emphasis on *audio* exchange. The RDP and VNC tools work well to interact with a remote GUI desktop but the video quality during capture and/or playback is low in terms of resolution, color depth and frame rate. YouTube and Wikis are very close, in that they allow authors to easily publish multimedia documents

which include video. But they tend to be *role*-based rather than interpersonal and do little to help capture video for a shared moment, the way webmail helps writers to compose an e-mail in an integrated editor. So (to me) the current technology that is most related seems to be webmail.

2.4 Related Technologies

Webmail helps to retract the transmission cost from the recipient's terminal workstation to somewhere closer to the recipient's e-mail server. This is subtle but evident in that the recipient can peruse her inbox, mark her spam for deletion, view and respond to her good mail- all without saving any residue to the workstation she is using. Rather than dumping each message on each recipient's workstation before a message preview is available, the recipient may get her work done at arm's length and may opt to download mail pieces of particular interest after quick and careful scrutiny of their sizes and types. With webmail, she gets the meta data first to help her decide whether she wants to get her hands dirty with the complete message's details. Still, candy from strangers might be flung into her bag, but with webmail she can run a series of taste-tests without toxic risk.

2.4.1 Webmail

Some specific examples webmail that have great features and security are: Google Gmail, Yahoo! Mail, and the open-source SquirrelMail project.

In terms of security, webmail often shines with the following features:

1. identity management
2. transport layer security
3. collaborative spam control
4. freedom from software installation
5. automatic vulnerability patching

The first two items are configurable and standards-based and they are optional features that come along with any browser-based web application. First, the configurable options for identity management in web applications are plentiful these days, reminiscent of the pluggable authentication management (PAM) framework for UNIX logins.

2.4.2 Transport Layer Security

Second, transport layer security (TLS) now has a new name since it has succeeded the patented [7] Netscape secure sockets layer (SSL) protocol ten years ago. But TLS is rock-solid, familiar to users, and has enormous industry support. TLS version 1 came from SSL version 3 with a handshaking protocol so advanced that checksum and cipher algorithms may be relaxed or made stronger on-the-fly *within* an established session. But casual users would hardly know that the attributes of their cipher suite are being re-negotiated and instead, another “small change” is probably one that has had greater meaning to end users.



Figure 6: A browser warning about an untrusted certificate signature

From a more practical point of view, the *default* lists of recognized certificate authorities that browsers and the Java runtime keystore come with has grown. As a specific example, ten years ago a website administrator may have been asked to pay over \$800 to have her web server's certificate signed by a VeriSign [8] CA, or over \$600 for a Thawte [9] signature. But today she can get an equivalent GoDaddy signature for \$100 [10]. Her users will enjoy the same benefits of secure TLS and they will not be made nervous by any "Untrusted Certificate" warnings from their web browsers.

Technical users may often know how to manage their registries of trusted CAs and they will comfortably navigate a browser warning with confidence. Yet casual non-technical users are likely to be confused, unsettled, annoyed or otherwise dissuaded from continuing to a website whose certificate is signed by a CA that their browser does not recognize. Figure 6 shows an example of the type of warning a user may get when a certificate is not recognized. This is a small security-related configuration change that has been ironed-out over the years. The browser's trusted CA lists continue to grow and improve for users, and these improvements are another important incentive enjoyed by secure browser-based web applications.

2.4.3 Spam Control

Collaborative spam control (#3) in webmail is an innovative way for peer users to submit instant feedback that identifies spam and cooperate in reducing its damage to webmail systems. This supplies critical early warning value to a webmail system that is similar to Google Flu Trends [11], a site that geocodes web search queries for influenza-related answers and estimates the current risk of flu outbreaks for a given geographic region. These are modern “complaint boxes” that can record details during the onset of an attack instantly and from the front line.

2.4.4 Automatic Updates

The webmail security features listed #4 (freedom from software installation) and #5 (automatic vulnerability patching) are also small differences that were touched upon in chapter 1. Software installation and patching are obstacles to end-users' adoption for three main reasons.

They are obstacles first because they are annoying; installation procedures interrupt productivity, they are time consuming and they may require the user to save all work and then reboot. Second- because astute users have been conditioned over time to suspect side-effects to installation and patching: that they might not easily be able to un-install, that something else that they have may be broken, that their system will be slower, and/or that they might be admitting a trojan horse or "spyware" onto their system. And third, they might not be allowed to install or patch software on a public or shared workstation because their account lacks sufficient permission; or they may work daily with diverse systems such as Linux, FreeBSD and/or Solaris for which the installations and/or patches are not available.

Browser-based web applications help users to enjoy effortless security enhancements and automatic vulnerability patching at a few levels that are worthy of some mention here. The web application may be improved incrementally on its server-side, such that the user gets the latest and

greatest version each time she logs-in to get webmail. The built-in web browsers that come bundled with operating systems are automatically updated since they are considered system components. And some third-party web browsers such as Firefox are beginning to include their own integrated and automatic update mechanism for the core browser and any installed plug-ins. For example, the Java runtime and applet plug-in has its own update manager that may be automatic on Microsoft Windows or manual (push-button) on other operating systems [12].

2.5 Related Work

Here I have listed some works that are related to this thesis topic and can be grouped into three main areas: multimedia, collaborative systems, and multimedia architectures.

2.5.1 Multimedia

A system called PECOLE provides portability by using Java and JXTA to keep systems compatible and well-performing on the widest range of devices [13]. Second, PECOLE eases on-line teleconferencing and topic presentation with a layered and peer to peer architecture. Third, PECOLE assists in locating peers, maintaining sessions, and in locating (or providing) translations for multilingual content.

Separately, author W. L. Yeung describes some of the current strengths of wiki technology as 1.) students already use wikis and accept wikis as great tools to create and explore content, and as such 2.) the use of wikis by students has been growing quickly. Wikis have potential for accelerated growth through integration of multimedia content including instant messaging and video conferencing to help in holding wiki users' attention [14].

In [15] a detailed description is provided for a system called "mc³" which makes multimedia publishing tasks quicker and easier. First, mc³ helps promote reuse of previous and related publications. Second, mc³ enables on-line access to a search-able and growing collection of learning modules. Third, mc³ facilitates federation between institutions to examine, exchange and enrich research and learning materials. The learning materials apparently tend to be in multimedia, and the mc³ services are designed to help teachers to collaborate. This leads to the next section, where I have listed some works whose primary emphasis seems to be on aiding-in users' collaboration while working with multimedia documents.

2.5.2 Collaboration

Anupam and Bajaj describe their work to support cooperative and accelerated application-level work-group development [16]. They call their

system "Shastra" and it has been documented to coordinate and serialize concurrent substrate transactions which contribute to group-wise scientific and engineering design. The heart of their system is a generic Shastra layer that is extensible and supports a wide variety of collaborative applications.

Mobile generation and representation of multimedia content is provided by a system built by Prabhu and Gadh [17]. One key to their design was a unified file format with delivery via message queue middle-ware. The middle-ware described in their work is intended to help direct migration from desktop collaborative systems toward mobile collaborative systems.

A work that explores both collaborative *and* architectural aspects of a multimedia system is described in [18]. Their methods for creation, revision and collaborative exchange of multimedia documents with high performance are kept flexible by developing a Jabber service oriented architecture (SOA). The article elaborates on an emphasis for keeping the multimedia document state consistent, and in keeping concurrent document changes coherent in their appearance for up to six concurrent end users.

2.5.3 Architecture

Requirements traceability during design, implementation and maintenance of large scale systems may be aided by adopting a *SOA service strategy*

planning space, which helps to map problems to solutions [19]. In their perspective, there seems to be a conceptual problem space above the service strategy planning space which contains a domain area, a context, and one or more business drivers. Below the service strategy planning space, their solution space contains engineering, business, and operations aspects of a SOA. Their interesting “service strategy” seems to be a set of functions that relate a domain of problems to a range of solutions in a traceable way that will succinctly indicate precise service changes required to accommodate healthy changes in the problem and solution spaces.

Separation of concerns for authoring services and publishing services of multimedia content is exemplified in the LimSee3 platform architecture [20]. According to Mikáč, Roisin, and Le Duc, adoption of new technologies is simplified and deployment of authoring services is facilitated by having an intermediate export format. This separation of concerns helps to insulate authoring tools from document formats and makes it easier to tailor authoring services to users' custom needs.

An experimental media sharing application for mobile phones called “Sandboxes” seeks to promote interactivity, flexibility, and cohesiveness for mobile authors of multimedia [21]. Sandboxes allow users to compose

messages that mix multiple media into a single document. The Sandboxes creators refer to this as a "multimedia collage on a shared 2D canvas."

These advances in multimedia, collaborative, and architectural technologies would help to sustain and entice a user base that is eager to create and exchange multimedia about their lives. So the Internet continues to grow with more personal computers and hand-held devices being added each minute. Now there are new possibilities opened to bring last mile broadband to subscribers with government help. Rural wireless "third pipes" that are both fat and cheap over 802.11n or 802.11y may spur growth of green shoots on grass roots networks that are fed by ubiquitous and instant authors of community home grown content [22][23].

3. SYSTEM DESIGN AND ARCHITECTURE

Some of the technologies planned for use in my approved thesis proposal were found to be unfit for one reason or another. And some goals proved to be too optimistic for my pace of development in the time that I had. My prototype still has a lot of growing to do if it will become an everyday tool. Both the design and implementation of the PushToShow have been kept as small and simple as possible in order to make it easier for adoption by others in their work.

3.1 Design Overview

The PushToShow prototype was built with a simple three-tier architecture that consists of a server-side with an application server and database that supports browser-based clients over HTTP:

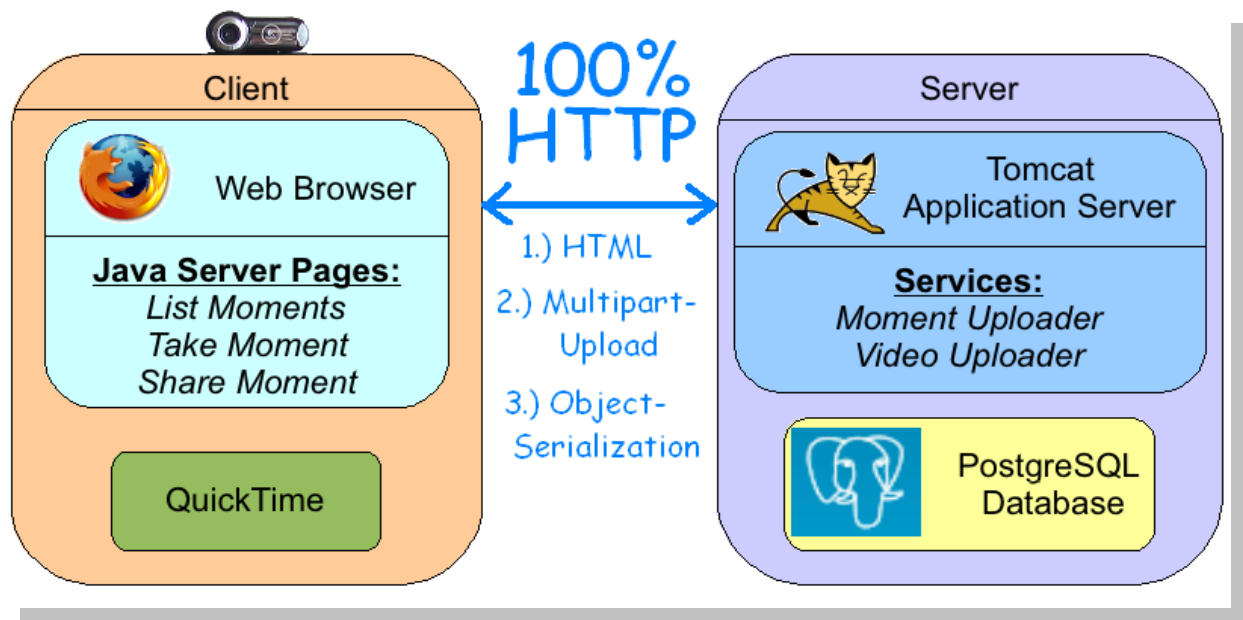


Figure 7: PushToShow has a three-tier architecture

As figure 7 shows, on the client-side there are three web pages to list moments, view moments, and capture moments. These HTML pages incorporate JavaScript, the QuickTime and Java plug-ins respectively. The ListMoments page uses JavaScript to open a new tab to view each moment that is clicked. The TakeMoment page uses the QuickTime plug-in to play the video portion of each moment. And the ShareMoment page uses the Java plug-in to capture, compress, and upload new moments.

All end-user features are browser-based, and all of the traffic between the client and server is over HTTP. Displayed output is downloaded over HTTP as HTML, images and video. Newly created moments are uploaded from the end-user's browser to the server over HTTP by using multi-part upload for the video portion and object serialization for the textual portions of shared moments. The term moment is used here to describe a new type of audio-visual document which is composed and published by a PushToShow author, such that a PushToShow recipient may then view the moment upon request.

3.2 Architecture

The following figures 8 and 9 show the UML class diagrams for the PushToShow Applet and web services.

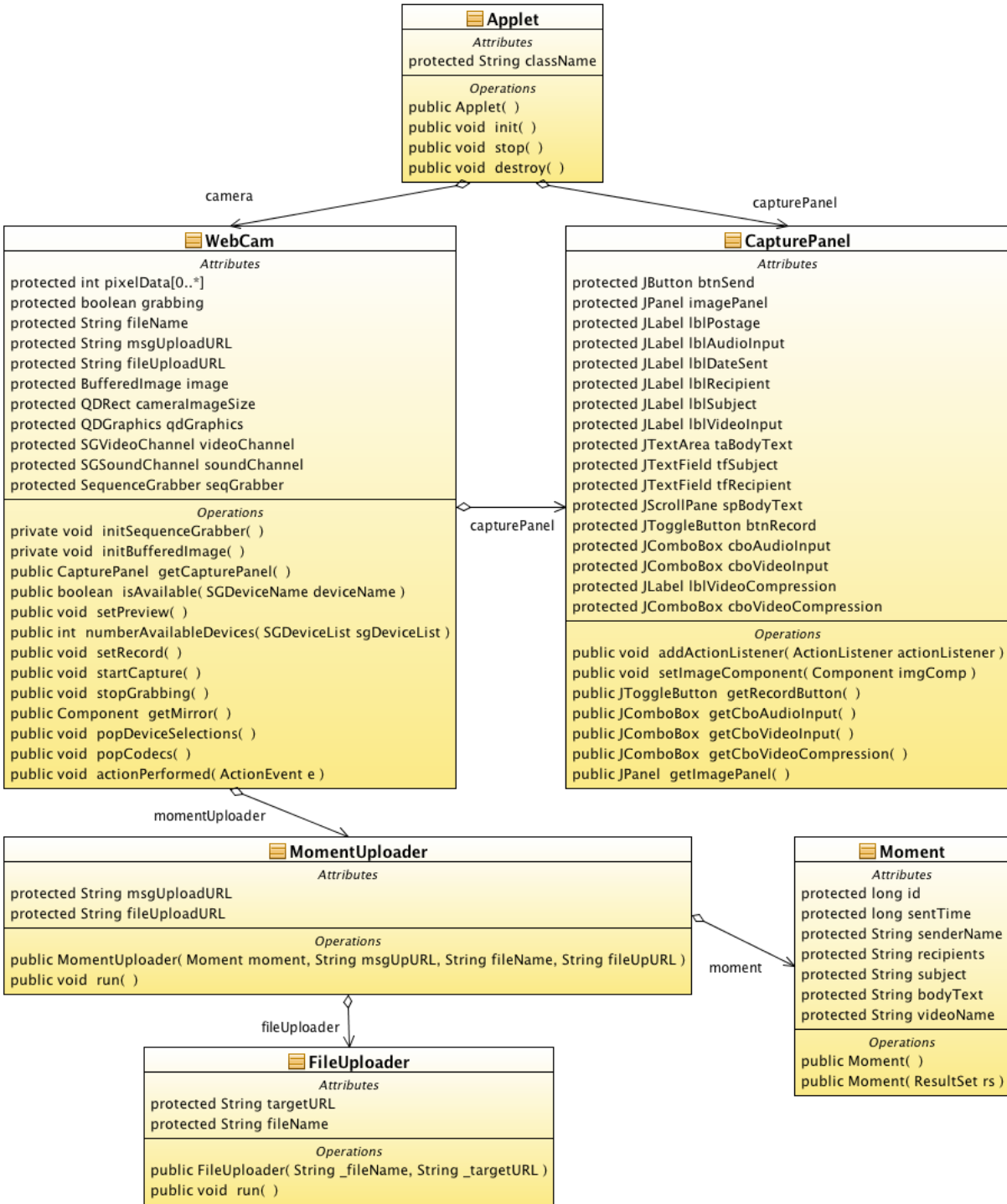


Figure 8: Class Diagram of the Client Side

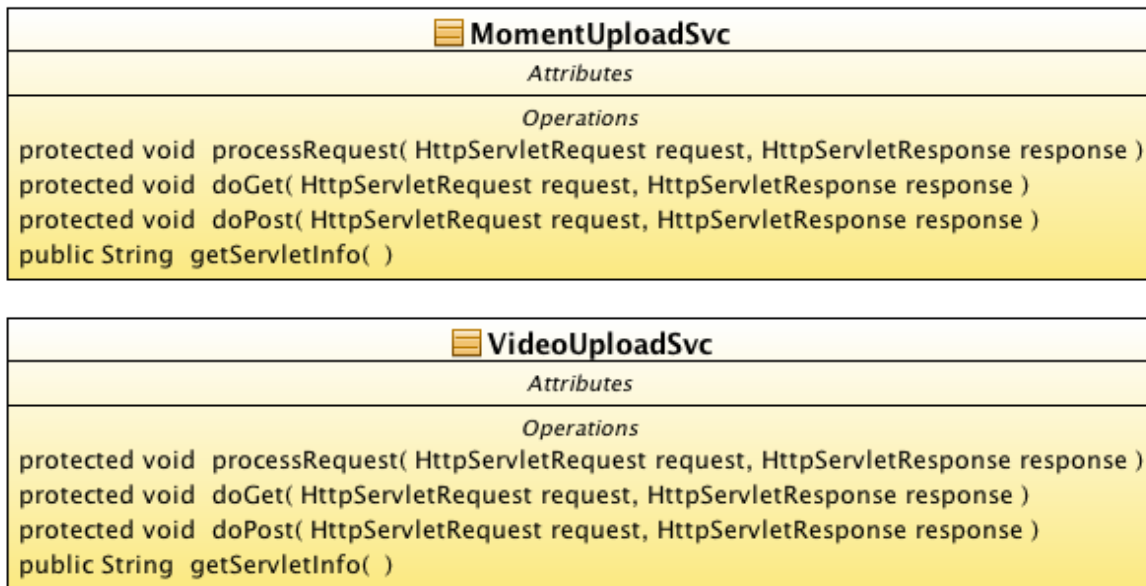


Figure 9: Web Service Classes

The following figures 10 and 11 show UML sequence diagrams for recording and sending a moment.

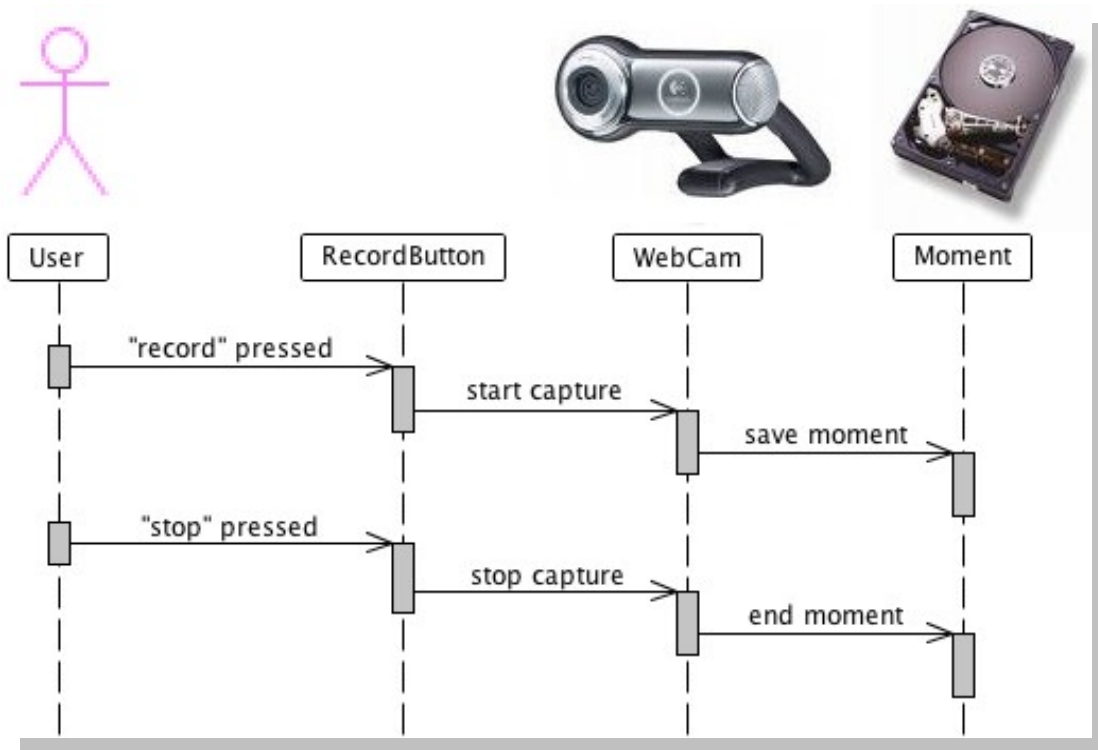


Figure 10: Record moment sequence diagram

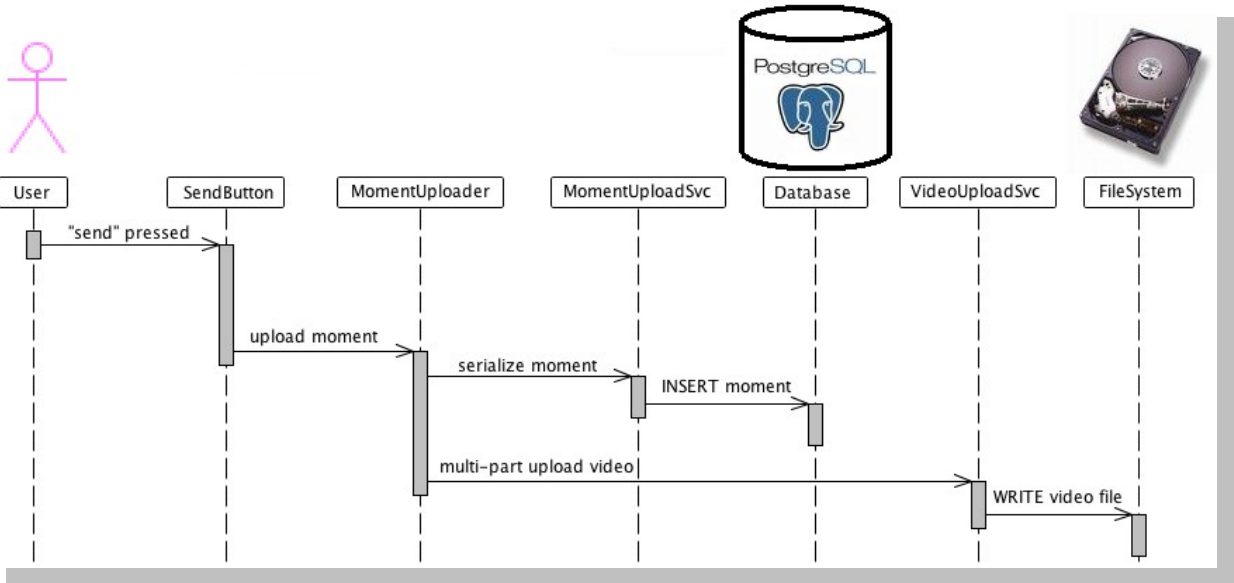


Figure 11: Send moment sequence diagram

4. DESCRIPTION OF PROTOTYPE

This chapter describes ordinary use cases for the PushToShow prototype, how to setup its development environment, and an overview of its design.

4.1 Send a Moment

A *moment* is like an e-mail message that includes sound and video. And like e-mail messages, moments must be composed and sent. PushToShow makes composing moments easier with a browser-based WYSIWYG authoring tool that includes a real-time graphical web-cam “mirror:”

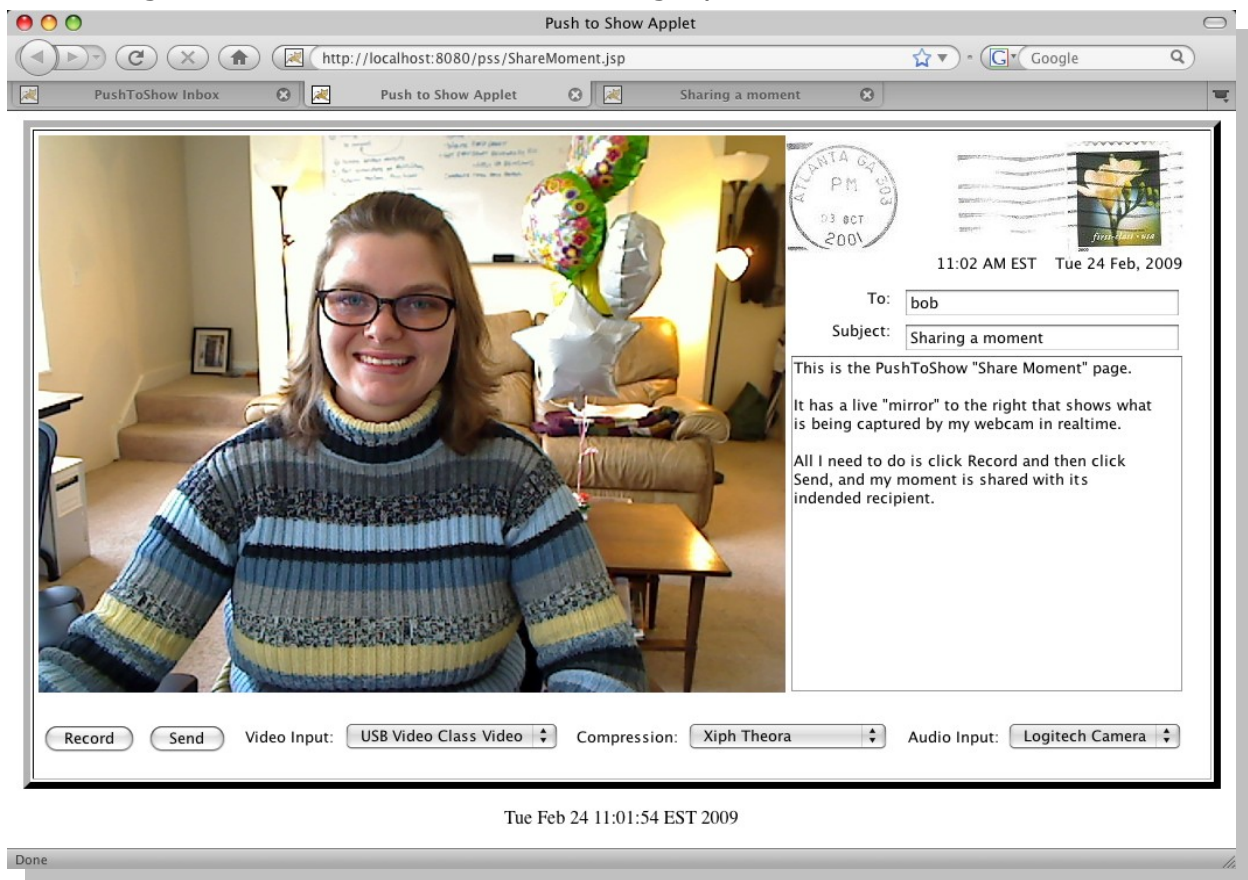


Figure 12: The ShareMoment web page

It has a large web-cam mirror and is used to compose and send moments.

4.1.1 Capture the Video Portion

Here are the steps to capture your moment with PushToShow:

1. Browse to <http://localhost:8080/pss/ShareMoment.jsp>
2. Verify video input, compression type, and audio input settings
3. Click the "Record" button to begin video capture
4. Click the "Stop" button when finished with a video "take"

4.1.2 Compose the Moment

Life's best moments must be composed with care. With PushToShow, it is easy to retry video takes and add a little side note to your moment:

1. The last take is the take that counts
2. Video takes are saved to the home directory with a time-stamp in the form `psv_YYYYMMDD_hhmmss.mov`
3. It is easy to edit the video portion with QuickTime Pro before sending
4. Type a caption or text message into the large text area on the right-hand side of the ShareMoment page

4.1.3 Send the Moment

It is time to share the moment that you have captured and composed:

1. Address the recipient
2. Enter the subject line
3. Click the "Send" button

Date	Sender	Subject
11:40 AM Tue 24 Feb, 2009	suzy	Sharing a moment
11:05 AM Tue 24 Feb, 2009	suzy	editing test 1
10:39 AM Tue 24 Feb, 2009	suzy	fred's balloons
10:35 AM Tue 24 Feb, 2009	suzy	Fred and the balloons
10:35 AM Tue 24 Feb, 2009	suzy	monkey cats
10:18 AM Tue 24 Feb, 2009	suzy	hello bob and alice
3:59 AM Tue 24 Feb, 2009	suzy	is psv working?
3:53 AM Tue 24 Feb, 2009	suzy	still messy
6:04 PM Sun 22 Feb, 2009	alice	sick suzy
3:31 PM Sat 24 Jan, 2009	alice	some subject
2:05 PM Sat 24 Jan, 2009	alice	java duke dance
1:59 PM Sat 24 Jan, 2009	alice	Jan 24 2009 test 2
1:52 PM Sat 24 Jan, 2009	alice	Jan 24 2009 test

Tue Feb 24 12:00:23 EST 2009 [Compose New Moment](#)

Figure 13: The ListMoments web page

This page shows links to the moments for the recipient Bob.

4.2 Receive a Moment

It is fun and easy to take a moment with PushToShow, and saving moments locally is the same as saving ordinary web pages:

4.2.1 Select a Moment

1. Browse to <http://localhost:8080/pss/ListMoments.jsp?recipient=alice>
2. Moments are listed chronologically with sender ID and subject line
(see figure 13)
3. Click an entry to view a moment

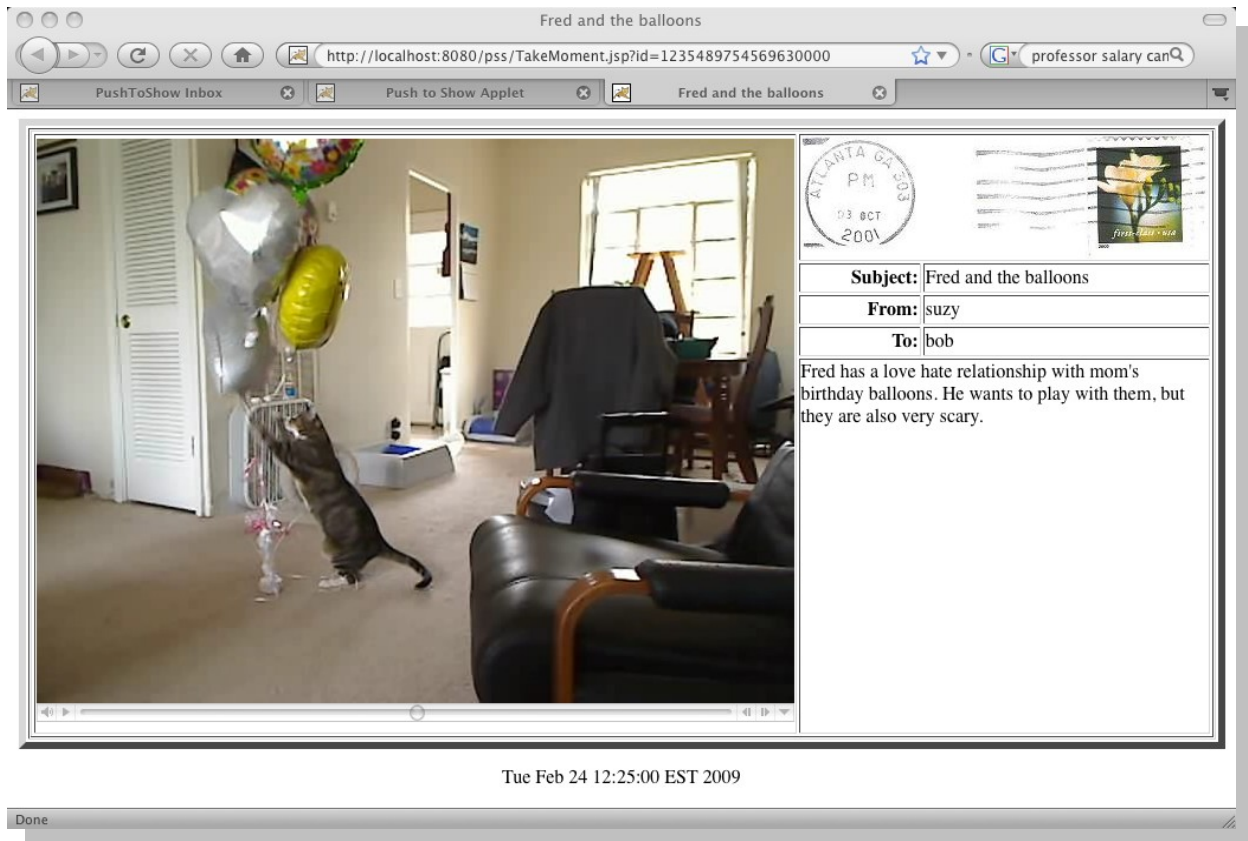


Figure 14: The TakeMoment web page

This page displays the moment and has compact video playback controls.

4.2.2 View a Moment

The viewer has plain text fields for meta-data and each URL is of the form:

<http://localhost:8080/pss/TakeMoment.jsp?id=123573625140463>

Beneath the image, the compact video playback controls may be used to:

1. adjust sound volume,
2. start or pause playback,
3. scan through frames of video,
4. single-step forward or backward,
5. and save the video portion of the moment to disk.

4.2.3 Save a Moment

Each moment is an ordinary web page, and may be saved to disk locally:

1. Use your built-in browser "Save page as.." feature
2. Keep your locally saved moments organized with browser bookmarks

4.3 Environment and Hardware Notes

Faster computers today have more storage and gigabit network connections that helped to make this prototype feasible. Providing video capture with compression and a simulcast viewing window each add up to a computing cost that has recently become affordable, even for video quality that approaches that of analog television.

The PushToShow prototype was built with many free and open source tools. And PushToShow is designed to work on any modern Mac. But here I want to quickly list my hardware and software suggestions for great video and low cost. Here is a parts list with early 2009 price estimates:

Item	Description	Cost
1	22-inch DVI monitor	\$250.00
2	Logitech QuickCam Vision Pro	\$150.00
3	Apple Mac Mini	\$900.00
4	USB Mouse Controller	\$20.00
5	USB Keyboard	\$30.00
6	QuickTime Pro (video editing software)	\$35.00

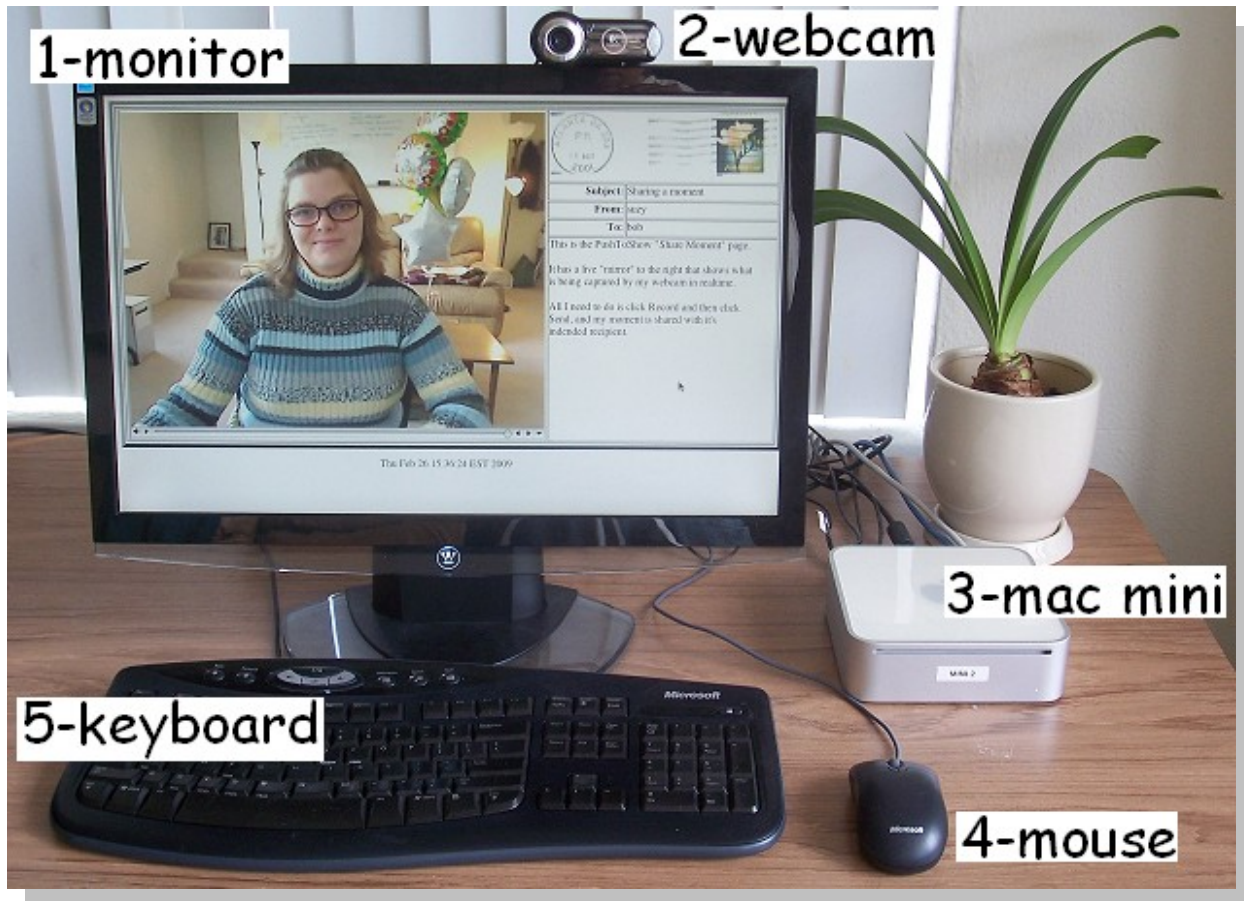


Figure 15: A PushToShow workstation

4.3.1 Mac Mini

PushToShow was developed-on and tested-with “Mac Mini” PCs from Apple, purchased new in August of 2007. This version of the Mac Mini has a 64-bit 1.8GHz Intel Core 2 Duo processor which has 2GB of memory and a front-side bus frequency of 667MHz. While testing PushToShow, I found that moments occupied about 1GB per hour of recorded video and that PushToShow uses about one megabit of network bandwidth per user.

The Mac Mini is the cheapest Mac offered by Apple, and it is the most versatile. By "versatile," I mean that the Mac Mini is compatible with PC hardware and can run Microsoft Windows or Linux in a dual-boot method with a new Mac OS 10.5 feature called "boot camp." The Mac Mini's small size and light weight make it portable between school, work, and home; the Mini has very low power consumption and it is virtually silent. The tamper-resistant case for the Mac Mini is wrapped with a heat-sink which helps keep required airflow (and dust) minimal and acts as a tough aluminum "bumper." Also the legendary security of Mac OS 10.5 helps to keep the Mac Mini safe against attack and "student resistant" to damage. So I think that these attributes make the Mac Mini a good PC for a classroom environment.

4.3.2 Logitech UVC Webcam

The Mac laptop models from Apple come with a built-in camera positioned just above the screen. But in my showroom testing (at BestBuy) they only seemed to capture about 4 frames per second at a resolution of 640x480 pixels. Still, those built-in cameras performed better than most of the external USB web-cams that I tested. Because in my testing I found that the performance specifications listed on the side of each box were wildly overstated. Often the cameras could not be set at the resolutions that were claimed, and rarely did I get the frame rates that were promised. Luckily they are quite cheap though and I was able to test quite a few cameras.

The Logitech QuickCam Vision Pro is a new Universal Serial Bus Video Class (UVC) camera that debuted about twelve months ago, in March of 2008. It is the only UVC camera that Apple would permit an Apple-logo for, from Logitech. I found that the QuickCam Vision Pro captures 15 frames per second of 640x480 (VGA) resolution video, about three time more video than any other USB camera that I tried; and it is compatible with both Microsoft Windows and Mac. This is the first and only UVC camera that I tried, since it lived up to its advertised performance. It would be interesting to try some other UVC cameras because this first UVC camera works so well.



Figure 16: Some cameras did not perform as advertised

The details of a movie that was captured using PushToShow with the Logitech QuickCam Vision Pro are shown in figure 17.



Figure 17: PushToShow achieves high video quality

5. CONCLUSION AND FUTURE WORK

For this thesis I designed and built a prototype of a small tool that might help users to express themselves. My research and development took about ten months and I found several ideas for features that did not seem to work yet, but they might pan-out in later revisions. The video capture portion seemed more difficult than imagined and the web-cam “mirror” is the part that I am most proud of.

The questions that I intended to explore are centered around how pleasant this chore could be made for users. And the PushToShow prototype is roughly as pleasant as I can imagine in that it is easy to use and it produces the best quality moments feasible with ordinary consumer equipment. The PushToShow prototype is clearly a big step forward from my earlier MomentShare project, both in terms of performance and ease of use.

5.1 Future Work

With another iteration this prototype will be ready for incubation as an open source project with educational value. After the PushToShow project is opened up for collaborative scrutiny and development then I believe that some features may be added to make PushToShow fit for an enterprise setting.

Priority number one will be to configure PushToShow to work with a web-based single sign-on authentication package like OpenSSO. This will give the senders of moments identity. With TLS and OpenSSO, PushToShow will have enterprise-class authentication and transport layer security that is easy for end-users, especially if they are already authenticated with another OpenID friendly web application. Other future work should include efforts to make PushToShow portable to Microsoft Windows and Linux by using the Java Media Framework (JMF) for which there are free native "performance packs" available to download and distribute from Sun Microsystems. Also, some usability features ought to be added such as better status messages and progress bars where applicable. And it would make PushToShow more practical if it were a drag-n-drop target for end-users' external home-made videos and/or screen-casts in addition to capturing video from their web-cams.

5.2 Development Environment

I am hopeful that others will gain interest in PushToShow and so for due diligence here I would like to describe how to set up its development environment. The development environment consists of a NetBeans IDE workspace with three projects, a designated directory to store video files and a small PostgreSQL database.

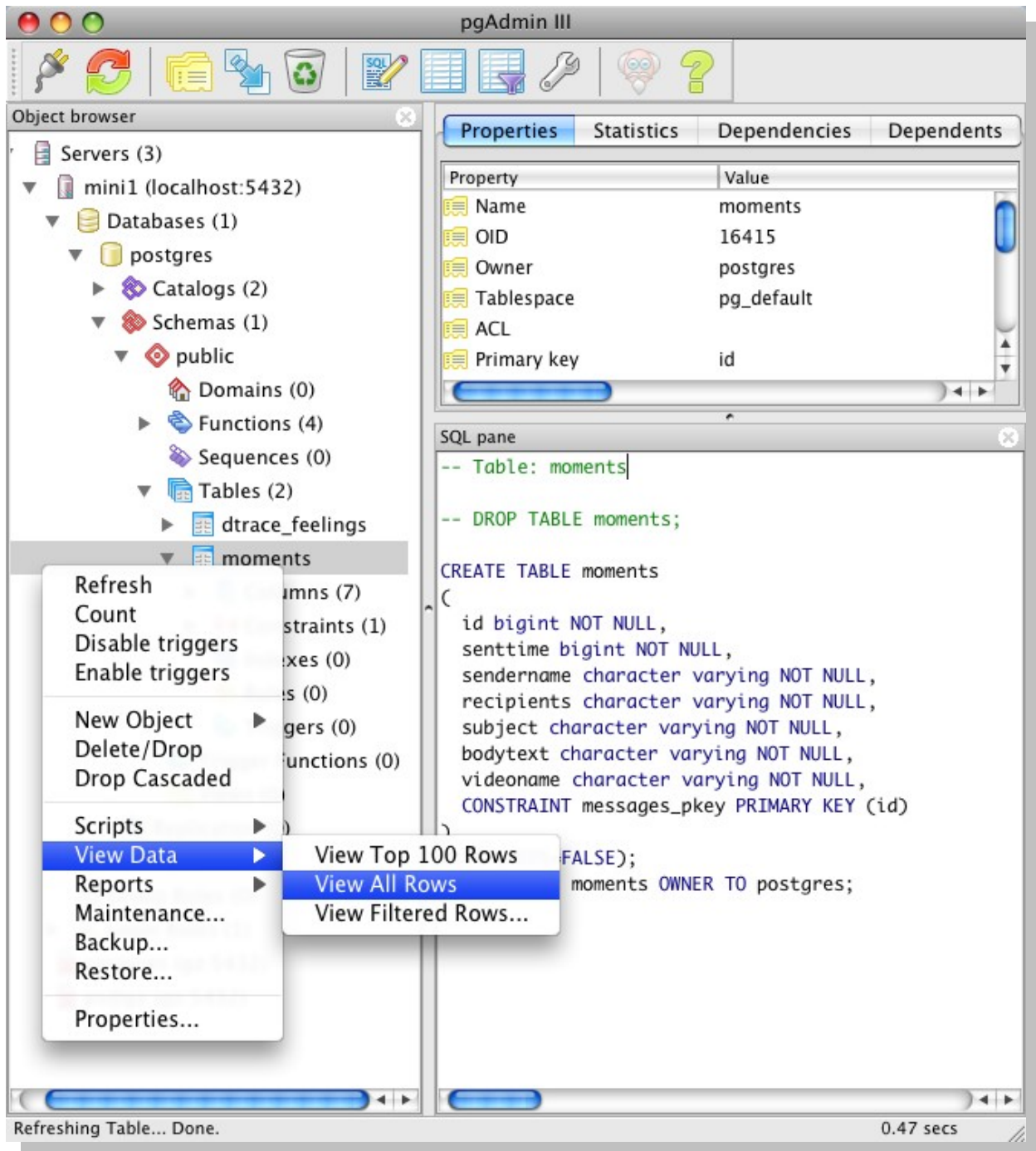


Figure 18: The pgAdmin3 database management tool

All of the application-layer source code is contained in the three small NetBeans projects, named with the three-lettered acronyms (TLAs) **psa** (applet) **pss** (servlets) and **psv** (video store.) There is a minimal source

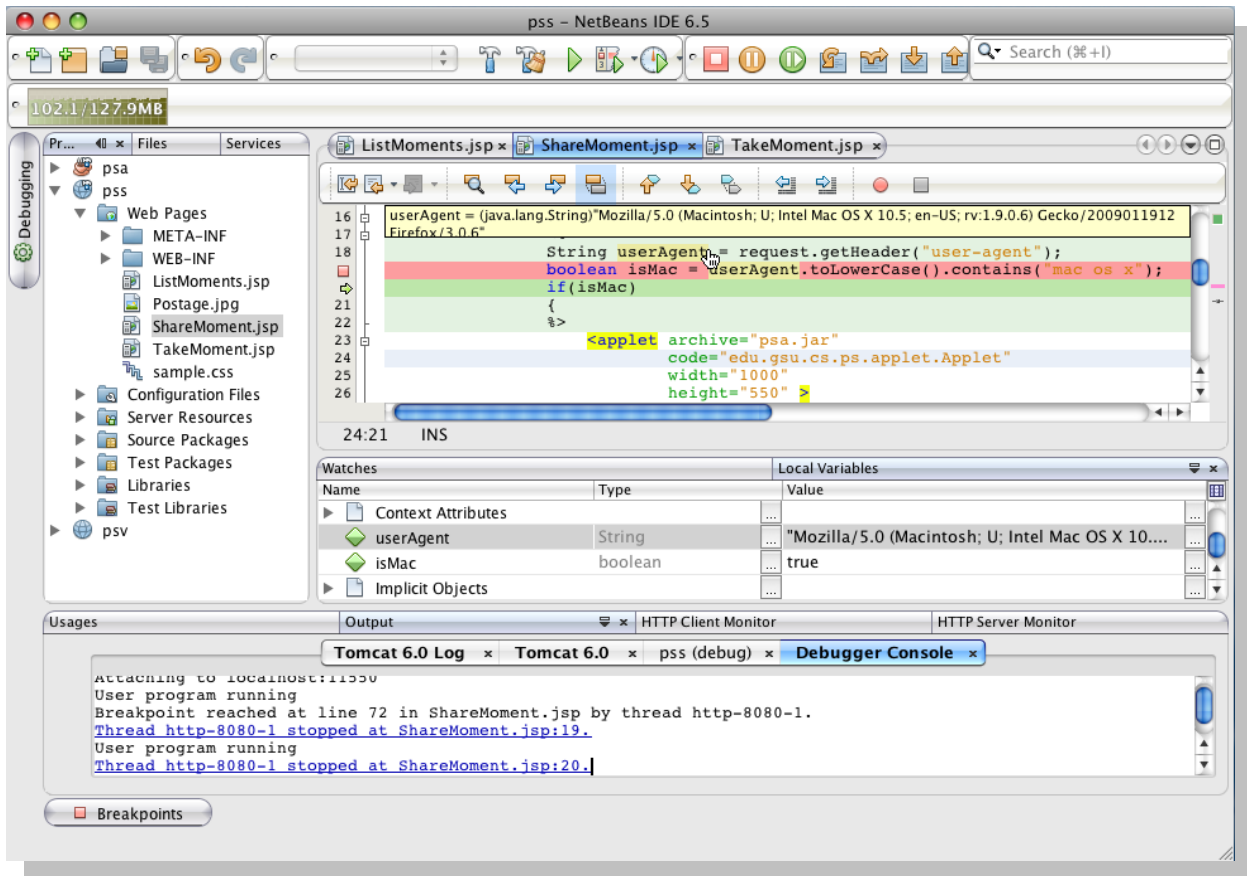


Figure 19: The NetBeans IDE

code base consisting of three JSPs and eight Java files, with a pair of deployment descriptors for the `pss` and `psv` web applications. The `psv` multipart upload service and video store was separated from the `pss` web application because I kept accidentally deleting movies that were in the video store when I performed a “clean” task. With the `psv` code separated from `pss`, I only need to backup my video files when I make changes to the `VideoUploadSvc.java` source code file. Most of the Java code is for the `PushToShow` applet, and runs in the end-user's web browser when she visits the `ShareMoment.jsp` web page to capture a new moment. And to simplify

the NetBeans workspace, all eight of the Java source code files are gathered into the edu.gsu.cs.ps package, which is located in the **psa** project.

The PushToShow database schema is also made to be about as simple as I could have imagined, with just one table to hold the moments, a pair of indices, and three simple stored procedures. The indices are on two table columns that appear in query WHERE and ORDER BY clauses, for recipients and sent-time. Three stored procedures SELECT moments for the ListMoments and ViewMoment web pages, and INSERT moments for the ShareMoment page. Database performance is improved by cheaper record lookups with the indices and by reducing SQL parsing time with the stored procedures. Also, the stored procedures were added because (as parameterized queries) they reduce web application exposure to SQL-injection style attacks and serve as security precautions.

To work with the PushToShow project on Mac OS 10.5 developers need to install three free and open-source tools, download the PushToShow source code, create the database and open the NetBeans workspace. The tools required are NetBeans IDE version 6.5 (figure 19), the PostgreSQL version 8.2 relational database management system (RDBMS) and the PgAdmin III graphical database client utility (figure 18).

Each of these tools has an intuitive installation procedure and their websites are well documented, so their download and installation should only take about an hour of time. In my experience, the only "gotcha" I had was that my Mac needed to be restarted for PostgreSQL to be started as a new service. Since then, PostgreSQL has automatically started every time I have booted my Mac PC. When the tools are installed, developers should download the PushToShow source code from:

<http://codd.cs.gsu.edu/~swatson13/thesis/PushToShowSrc.zip>

or

<http://www.pushtoshow.net>

After unpacking the project source code file there will be two files called "`readme.txt`" and "`PushToShow.sql`" and a subdirectory called "ps." It should only take about an hour to browse the "`readme.txt`" file, load the "`PushToShow.sql`" database DDL file into a PostgreSQL database, and open the three small Java projects in the "ps" directory with the NetBeans IDE.

Then the deployment descriptors (called `web.xml`) will need to be reviewed and maybe edited in the `pss` and `psv` projects, to revise database connection settings and designate a directory for storing moments' video files. This will also be detailed in the `readme.txt` file and should just take a few minutes to do. Finally, the three projects may be highlighted in

NetBeans IDE and a "clean and build" task invoked. When the **pss** and **psv** projects are highlighted and at last their "run" or "debug" tasks are invoked, the ShareMoment and LisMoments pages should be available at:

<http://localhost:8080/pss/ShareMoment.jsp>

and

<http://localhost:8080/pss/ListMoments.jsp>

This project comes from me watching scenes of people receiving important messages that they watched and listened-to on screens or as holograms in futuristic science-fiction movies. It comes from my understanding of previous and current attempts to make communication quicker, easier, and more accurate for ordinary people. And it comes from my practice in learning dozens of tiny elements of modern computing science over the past four years in Atlanta at Georgia State University.

It would mean the most to me if I could string together the little pieces of my understanding to organize a voluntary and educational project whose free and open-source software products help people to communicate and record their experiences. And this thesis came from my desire to be someday recognized for serving others by contributing to some slow sweet progress for the human race.

- [8] Compare All SSL Certificates, VeriSign, <http://www.verisign.com/ssl/buy-ssl-certificates/secure-site-services/index.html>
- [9] Pricing, Thawte, <https://www.thawte.com/pricing/>
- [10] Choose your SSL Certificate!, GoDaddy, <http://www.godaddy.com/gdshop/ssl/ssl.asp?ci=9039>
- [11] Google Flu Trends, <http://www.google.org/flutrends/>
- [12] Java Control Panel, <http://java.sun.com/j2se/1.5.0/docs/guide/deployment/deployment-guide/jcp.html>
- [13] Abdulmotaleb El Saddik, Abdur Rahman, Souhail Abdala, and Bogdan Solomon, "PECOLE: P2P multimedia collaborative environment", *Multimedia Tools and Applications*, 2007
- [14] W. L. Yeung, "Supplementing Wikis with Multimedia Collaboration Support", *International Conference on Hybrid Learning 2008*, August 13-15, 2008, Hong Kong, China
- [15] Gerd Kortemeyer and Wolfgang Bauer, "Multimedia Collaborative Content Creation (mc³)- the MSU LectureOnline System", *Frontiers in Education Conference, FIE '98. 28th Annual*, November 4-7, 1998, Tempe, AZ, USA
- [16] Vinod Anupam and Chandrajit L. Bajaj, "Shastra: Multimedia Collaborative Design Environment", *IEEE MultiMedia*, 1994, Vol. 1, Pg. 39-49

- [17] Xiaoyong Su, B. S. Prabhu, Chi-Cheng Chu, and Rajit Gadh, "Middleware for Multimedia Mobile Collaborative System", *Wireless Telecommunications Symposium (WTS 2004)*, May 14-15, 2004, CalPoly Pomona, Pomona, California, USA
- [18] Andrew Roczniak, Salinah Janmohamed, Christian Roch, Abdulmotaleb El Saddik and Pierre L ´evy, "SOA-based Collaborative Multimedia Authoring", *MCETECH 2006*, May 17-19, 2006, Montreal, Quebec, Canada
- [19] K. Kontogiannis, G. A. Lewis, D. B. Smith, M. Litoiu, H. Müller, S. Schuster, and E. Stroulia, "The Landscape of Service-Oriented Systems: A Research Perspective", *International Workshop on Systems Development in SOA Environments (SDSOA'07)*, May 21, 2007, Minneapolis, MN, USA
- [20] Jan Mikáč, Cécile Roisin, and Bao Le Duc, "An Export Architecture for a Multimedia Authoring Environment", *DocEng '08*, September 16-19, 2008, São Paulo, Brazil
- [21] David Fono and Scott Counts, "Sandboxes: Supporting Social Play through Collaborative Multimedia Composition on Mobile Phones", *Computer Supported Cooperative Work*, November 4-8, 2006, Banff, Alberta, Canada

[22] Wally Bowen, "Local Network Cookbook: A Recipe for Launching a Local Broadband Wireless Network", *MAIN - Mountain Area Information Network*, 2009, <http://main.nc.us/lan-recipe/>

[23] The American Recovery and Reinvestment Act of 2009, http://www.baller.com/pdfs/Baller_Herbst_Stimulus_2-19-09.pdf

APPENDIX: SOURCE CODE LISTING

The Applet class is the main point of entry for the PushToShow application's ShareMoment page.

```

1  /* Susan Gentner
2  * PushToShow Masters Thesis
3  * Georgia State University
4  * March 2009
5  */
6  package edu.gsu.cs.ps;
7
8  import javax.swing.*;
9  import java.awt.Component;
10
11 import quicktime.*;
12
13 public class Applet extends JApplet
14 {
15     protected WebCam camera;
16     protected final String className;
17     protected CapturePanel capturePanel;
18
19     public Applet()
20     {
21         className = this.getClass().getName();
22         System.out.println( (new java.util.Date()).toString() + " - " +
23             className + ".Applet() called.");
24         try
25         {
26             QTSession.open();
27         }
28         catch (QTException ex)
29         {
30             ex.printStackTrace();
31         }
32
33         System.out.println( (new java.util.Date()).toString() + " - " +
34             " QTSession.open() called.");
35
36         System.out.println( (new java.util.Date()).toString() + " - " +
37             " QTSession.canDoFullScreen() = " +
38             QTSession.canDoFullScreen() );
39
40         System.out.println( (new java.util.Date()).toString() + " - " +
41             " QTSession.hasSecurityRestrictions() = " +
42             QTSession.hasSecurityRestrictions() + "\n" );
43     }
44
45     public void init()

```

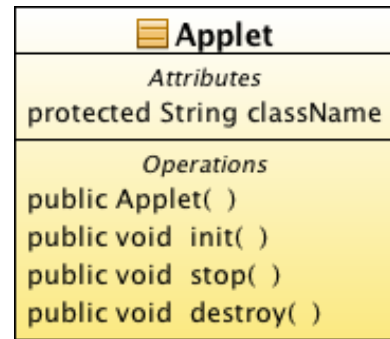


Figure 20: Applet class diagram


```

46     {
47         System.out.println( (new java.util.Date()).toString() + " - " +
48             className + ".init() called.");
49         try
50         {
51             camera = new WebCam();
52             camera.setPreview();
53             camera.startCapture();
54             camera.setPreview();
55             camera.startCapture();
56             camera.popDeviceSelections();
57             camera.popCodecs();
58             camera.setMsgUploadURL(getParameter("MsgUploadServletURL"));
59             camera.setFileUploadURL(getParameter("FileUploadServletURL"));
60             System.out.println("camera.getFileUploadURL()= " +
61                 camera.getFileUploadURL() + "");
62             System.out.println("camera.getMsgUploadURL= " +
63                 camera.getMsgUploadURL() + "\n");
64
65             Component imageComponent = camera.getMirror();
66             capturePanel = camera.getCapturePanel();
67             capturePanel.setImageComponent( imageComponent );
68         }
69         catch( Exception e )
70         {
71             e.printStackTrace();
72         }
73
74         this.setBackground(new java.awt.Color(255, 255, 255));
75         this.setSize(1005, 600);
76         this.getContentPane().add(capturePanel);
77     }
78
79     public void stop()
80     {
81         System.out.println( (new java.util.Date()).toString() + " - " +
82             className + ".stop() called.");
83
84         camera.stopGrabbing();
85         System.out.println( (new java.util.Date()).toString() + " - " +
86             " WebCam.stopGrabbing() called.\n");
87     }
88
89     public void destroy()
90     {
91         System.out.println( (new java.util.Date()).toString() + " - " +
92             className + ".destroy() called.");
93
94         QTSession.exitMovies();
95         System.out.println( (new java.util.Date()).toString() + " - " +
96             " QTSession.exitMovies() called.");
97
98         QTSession.close();
99         System.out.println( (new java.util.Date()).toString() + " - " +
100             " QTSession.close() called.\n");
101     }
102 }

```

CapturePanel is the main GUI component in the ShareMoment web page.

```

1  /* Susan Gentner
2  * PushToShow Masters Thesis
3  * Georgia State University
4  * March 2009
5  */
6  package edu.gsu.cs.ps;
7
8  import java.awt.*;
9  import javax.swing.*;
10 import java.util.Date;
11 import java.text.SimpleDateFormat;
12 import java.awt.event.ActionListener;
13 import org.netbeans.lib.awtextra.*;
14
15 public class CapturePanel
16     extends JPanel
17 {
18     protected final JButton btnSend;
19     protected final JPanel imagePanel;
20     protected final JLabel lblPostage;
21     protected final JLabel lblAudioInput;
22     protected final JLabel lblDateSent;
23     protected final JLabel lblRecipient;
24     protected final JLabel lblSubject;
25     protected final JLabel lblVideoInput;
26     protected final JTextArea taBodyText;
27     protected final JTextField tfSubject;
28     protected final CardLayout cardLayout;
29     protected final JTextField tfRecipient;
30     protected final JScrollPane spBodyText;
31     protected final JToggleButton btnRecord;
32     protected final JComboBox cboAudioInput;
33     protected final JComboBox cboVideoInput;
34     protected final JLabel lblVideoCompression;
35     protected final JComboBox cboVideoCompression;
36     protected final SimpleDateFormat simpleDateFormat;
37
38     public CapturePanel()
39     {

```

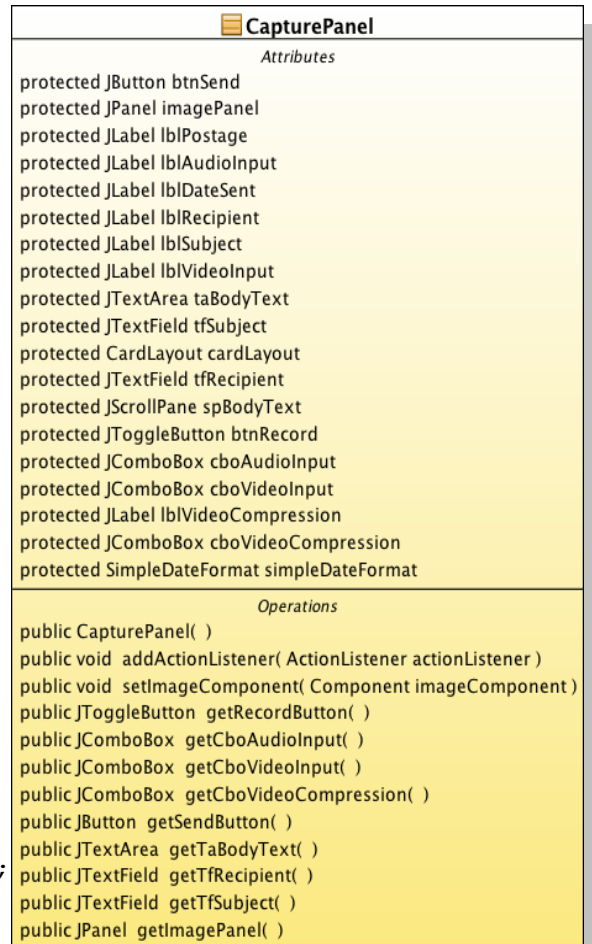


Figure 21: CapturePanel class

```
40     lblVideoInput = new JLabel();
41     lblAudioInput = new JLabel();
42     cboVideoInput = new JComboBox();
43     cboAudioInput = new JComboBox();
44     lblVideoCompression = new JLabel();
45     cboVideoCompression = new JComboBox();
46     spBodyText = new JScrollPane();
47     taBodyText = new JTextArea();
48     tfRecipient = new JTextField();
49     lblRecipient = new JLabel();
50     lblSubject = new JLabel();
51     tfSubject = new JTextField();
52     btnSend = new JButton();
53     btnRecord = new JToggleButton();
54     lblDateSent = new JLabel();
55     lblPostage = new JLabel();
56     cardLayout = new CardLayout();
57     imagePanel = new JPanel(cardLayout);
58     simpleDateFormat =
59         new SimpleDateFormat("h:mm a z    E dd MMM, yyyy");
60
61     setBackground(new Color(255, 255, 255));
62     setMinimumSize(new Dimension(920, 560));
63     setPreferredSize(new Dimension(930, 560));
64     setLayout(new AbsoluteLayout());
65
66     lblVideoInput.setBackground(new Color(255, 255, 255));
67     lblVideoInput.setText("Video Input:");
68     add(lblVideoInput, new AbsoluteConstraints(177, 510, -1, -1));
69
70     lblAudioInput.setBackground(new Color(255, 255, 255));
71     lblAudioInput.setText("Audio Input:");
72     add(lblAudioInput, new AbsoluteConstraints(745, 510, -1, -1));
73
74     cboVideoInput.setBackground(new Color(255, 255, 255));
75     add(cboVideoInput, new AbsoluteConstraints(259, 505, -1, -1));
76
77     cboAudioInput.setBackground(new Color(255, 255, 255));
78     add(cboAudioInput, new AbsoluteConstraints(827, 505, -1, -1));
79
80     lblVideoCompression.setBackground(new Color(255, 255, 255));
81     lblVideoCompression.setText("Compression:");
82     add(lblVideoCompression,
83         new AbsoluteConstraints(461, 510, -1, -1));
84
85     cboVideoCompression.setBackground(new Color(255, 255, 255));
86     add(cboVideoCompression,
87         new AbsoluteConstraints(553, 505, -1, -1));
88
89     spBodyText.setPreferredSize(new Dimension(100, 84));
90     taBodyText.setColumns(20);
91     taBodyText.setRows(5);
92     taBodyText.setPreferredSize(new Dimension(100, 80));
93     taBodyText.setLineWrap(true);
94     spBodyText.setViewportViewView(taBodyText);
95     add(spBodyText, new AbsoluteConstraints(645, 190, 335, 290));
96
```

```

97     tfRecipient.setText("bob");
98     add(tfRecipient,
99         new AbsoluteConstraints(740, 130, 240, -1));
100
101     lblRecipient.setBackground(new Color(255, 255, 255));
102     lblRecipient.setHorizontalAlignment(SwingConstants.RIGHT);
103     lblRecipient.setText("To:");
104     lblRecipient.setHorizontalAlignment(SwingConstants.RIGHT);
105     add(lblRecipient,
106         new AbsoluteConstraints(710, 130, -1, 20));
107
108     lblSubject.setBackground(new Color(255, 255, 255));
109     lblSubject.setHorizontalAlignment(SwingConstants.RIGHT);
110     lblSubject.setText("Subject:");
111     lblSubject.setToolTipText("Subject");
112     lblSubject.setHorizontalAlignment(SwingConstants.RIGHT);
113     add(lblSubject,
114         new AbsoluteConstraints(680, 160, -1, -1));
115
116     tfSubject.setText("sharing a moment with ya...");
117     add(tfSubject,
118         new AbsoluteConstraints(740, 160, 240, -1));
119
120     btnRecord.setBackground(new Color(255, 255, 255));
121     btnRecord.setText("Record");
122     btnRecord.setToolTipText("Record a video clip");
123     add(btnRecord,
124         new AbsoluteConstraints(0, 505, -1, -1));
125
126     btnSend.setBackground(new Color(255, 255, 255));
127     btnSend.setText("Send");
128     btnSend.setToolTipText("Send the complete message");
129     add(btnSend,
130         new AbsoluteConstraints(90, 505, -1, -1));
131
132     lblDateSent.setBackground(new Color(255, 255, 255));
133     lblDateSent.setHorizontalAlignment(SwingConstants.RIGHT);
134
135     lblDateSent.setText(simpleDateFormat.format(new Date()));
136     add(lblDateSent,
137         new AbsoluteConstraints(740, 70, 240, 80));
138
139     ImageIcon keyFrame =
140         new ImageIcon(getClass().getResource("Postage.jpg"));
141     lblPostage.setIcon(keyFrame);
142     add(lblPostage,
143         new AbsoluteConstraints(640, 5, 343, 104));
144
145     add(imagePanel,
146         new AbsoluteConstraints(0, 0, 640, 480));
147 }
148
149 public void addActionListener( final ActionListener actionListener )
150 {
151     btnRecord.addActionListener( actionListener );
152     btnSend.addActionListener( actionListener );
153     cboAudioInput.addActionListener( actionListener );

```

```
154         cboVideoInput.addActionListener( actionListener );
155         cboVideoCompression.addActionListener( actionListener );
156     }
157
158     public void setImageComponent( final Component imageComponent )
159     {
160         imagePanel.add( imageComponent, "capture" );
161     }
162
163     public JToggleButton getRecordButton()
164     {
165         return btnRecord;
166     }
167
168     public JComboBox getCboAudioInput()
169     {
170         return cboAudioInput;
171     }
172
173     public JComboBox getCboVideoInput()
174     {
175         return cboVideoInput;
176     }
177
178     public JComboBox getCboVideoCompression()
179     {
180         return cboVideoCompression;
181     }
182
183     public JButton getSendButton()
184     {
185         return btnSend;
186     }
187
188     public JTextArea getTaBodyText()
189     {
190         return taBodyText;
191     }
192
193     public JTextField getTfRecipient()
194     {
195         return tfRecipient;
196     }
197
198     public JTextField getTfSubject()
199     {
200         return tfSubject;
201     }
202
203     public JPanel getImagePanel()
204     {
205         return imagePanel;
206     }
207 }
208
```

FileUploader is used by the applet to upload the video portion of the moment.

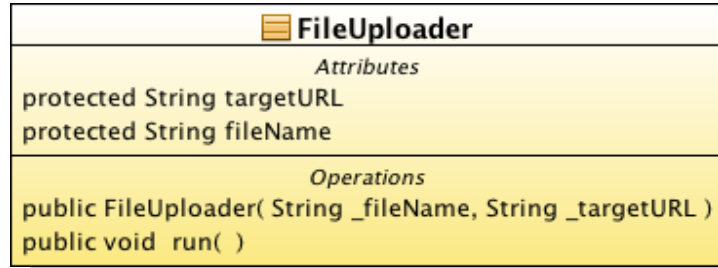


Figure 22: FileUploader class

```

1  /* Susan Gentner
2  * PushToShow Masters Thesis
3  * Georgia State University
4  * March 2009
5  */
6  package edu.gsu.cs.ps;
7
8  import java.io.File;
9  import org.apache.commons.httpclient.*;
10 import org.apache.commons.httpclient.methods.PostMethod;
11 import org.apache.commons.httpclient.methods.multipart.*;
12 import org.apache.commons.httpclient.params.HttpMethodParams;
13
14 public class FileUploader implements Runnable
15 {
16     protected final String targetURL;
17     protected final String fileName;
18
19     public FileUploader( final String _fileName,
20                         final String _targetURL )
21     {
22         fileName = _fileName;
23         targetURL = _targetURL;
24     }
25
26     public void run()
27     {
28         System.out.println( "Transferring File: '" +
29                             fileName + "' \n\n" );
30         try
31         {
32             File transferfile = new File( fileName );
33             PostMethod filePost = new PostMethod( targetURL );
34             filePost.getParams().setBooleanParameter(
35                 HttpMethodParams.USE_EXPECT_CONTINUE, false );
36             try
37             {
38                 final Part[] parts =
39                 {
40                     new FilePart( "userfile", transferfile )
41                 };
42
43                 filePost.setRequestEntity(
44                     new MultipartRequestEntity( parts,
45                                                 filePost.getParams() ) );
46
47                 final HttpClient client = new HttpClient();
48                 client.getHttpConnectionManager().
49                     getParams().setConnectionTimeout( 5000 );

```

```
50         int status = client.executeMethod( filePost );
51
52         if( status == HttpStatus.SC_OK )
53         {
54             System.out.println( "'" + fileName +
55                                 "' - Upload complete, " +
56                                 "response=" +
57                                 filePost.getResponseBodyAsString()
58                                 );
59             System.out.println("Transfer resulted: " +
60                                 filePost.getResponseBodyAsString()
61                                 + "\n\n" );
62         }
63
64         else
65         {
66             System.out.println( "'" + fileName +
67                                 "' - Upload failed, response=" +
68                                 HttpStatus.getStatusText( status )
69                                 );
70             System.out.println("Transfer resulted: " +
71                                 HttpStatus.getStatusText( status )
72                                 + "\n\n" );
73         }
74     }
75     catch( Exception ex )
76     {
77         System.out.println( "ERROR: " + ex.getClass().getName() +
78                             " " + ex.getMessage() );
79         ex.printStackTrace();
80     }
81     finally
82     {
83         filePost.releaseConnection();
84     }
85     System.out.println( "Transfer complete.\n\n" );
86     System.out.println( "Drop files here." );
87
88     return;
89 }
90 catch( Exception e )
91 {
92     e.printStackTrace();
93 }
94 }
95 }
96 }
```

Moment is the metadata for a PushToShow moment, it does not include the video portion but it does have the video file name.

```

1  /* Susan Gentner
2   * PushToShow Masters Thesis
3   * Georgia State University
4   * March 2009
5   */
6  package edu.gsu.cs.ps;
7
8  import java.util.Date;
9  import java.io.Serializable;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 public class Moment
14     implements Serializable
15 {
16     protected long id;
17     protected long sentTime;
18     protected String senderName;
19     protected String recipients;
20     protected String subject;
21     protected String bodyText;
22     protected String videoName;
23
24     public Moment()
25     {
26     }
27
28     public Moment( final ResultSet rs ) throws SQLException
29     {
30         id = rs.getLong( "id" );
31         sentTime = rs.getLong( "sentTime" );
32         senderName = rs.getString( "senderName" );
33         recipients = rs.getString( "recipients" );
34         subject = rs.getString( "subject" );
35         bodyText = rs.getString( "bodyText" );
36         videoName = rs.getString( "videoName" );
37     }
38
39     public String toString()
40     {
41         return( "id = " + id +
42             ", sentTime = '" + (new Date(sentTime)).toString() +
43             ", senderName = '" + senderName +
44             "', recipients = '" + recipients +
45             "', subject = '" + subject +
46             "', bodyText = '" + bodyText +

```

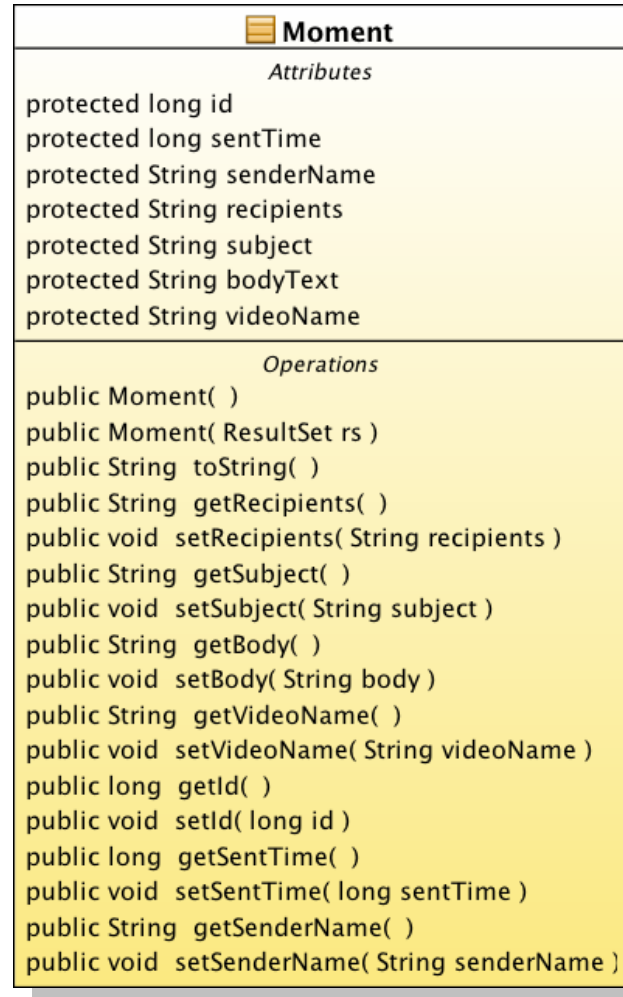


Figure 23: Moment class


```
47         ", videoName = '" + videoName + '"
48     );
49 }
50
51 public String getRecipients()
52 {
53     return recipients;
54 }
55
56 public void setRecipients(String recipients)
57 {
58     this.recipients = recipients;
59 }
60
61 public String getSubject()
62 {
63     return subject;
64 }
65
66 public void setSubject(String subject)
67 {
68     this.subject = subject;
69 }
70
71 public String getBody()
72 {
73     return bodyText;
74 }
75
76 public void setBody(String body)
77 {
78     this.bodyText = body;
79 }
80
81 public String getVideoName()
82 {
83     return videoName;
84 }
85
86 public void setVideoName(String videoName)
87 {
88     this.videoName = videoName;
89 }
90
91 public long getId()
92 {
93     return id;
94 }
95
96 public void setId(long id)
97 {
98     this.id = id;
99 }
100
101 public long getSentTime()
102 {
103     return sentTime;
```

```
104     }
105
106     public void setSentTime(long sentTime)
107     {
108         this.sentTime = sentTime;
109     }
110
111     public String getSenderName()
112     {
113         return senderName;
114     }
115
116     public void setSenderName(String senderName)
117     {
118         this.senderName = senderName;
119     }
120 }
121
```


 MomentUploader	
<i>Attributes</i>	
protected String msgUploadURL protected String fileUploadURL	
<i>Operations</i>	
public MomentUploader(Moment _moment, String _msgUploadURL, String _fileName, String _fileUploadURL) public void run()	

Figure 24: MomentUploader class

MomentUploader uploads the moments to the server.

```

1  /* Susan Gentner
2   * PushToShow Masters Thesis
3   * Georgia State University
4   * March 2009
5   */
6  package edu.gsu.cs.ps;
7
8  import java.net.URL;
9  import java.io.IOException;
10 import com.oreilly.servlet.HttpMessage;
11
12 public class MomentUploader implements Runnable
13 {
14     protected final Moment moment;
15     protected final String msgUploadURL;
16     protected final String fileUploadURL;
17     protected final FileUploader fileUploader;
18
19     public MomentUploader( final Moment _moment,
20                           final String _msgUploadURL,
21                           final String _fileName,
22                           final String _fileUploadURL )
23     {
24         moment = _moment;
25         msgUploadURL = _msgUploadURL;
26         fileUploadURL = _fileUploadURL;
27         fileUploader = new FileUploader( _fileName, _fileUploadURL );
28     }
29
30     public void run()
31     {
32         System.out.println("Uploading moment with subject line '" +
33                             moment.getSubject() + "' \n\t to URL '" +
34                             fileUploadURL + "'");
35         try
36         {
37             System.out.println("Starting to upload the file");
38             fileUploader.run();
39             System.out.println("No exceptions after upload." );
40
41             URL url = new URL( msgUploadURL );
42             HttpMessage msg = new HttpMessage(url);

```

```
43         msg.sendPostMessage(moment);
44     }
45     catch( IOException e )
46     {
47         e.printStackTrace();
48     }
49
50     System.out.println("Done uploading moment with subject line '" +
51         moment.getSubject() + "'");
52 }
53 }
54
```

MomentUploadSvc	
<i>Attributes</i>	
<i>Operations</i>	
protected void	processRequest(HttpServletRequest request, HttpServletResponse response)
protected void	doGet(HttpServletRequest request, HttpServletResponse response)
protected void	doPost(HttpServletRequest request, HttpServletResponse response)
public String	getServletInfo()

Figure 25: MomentUploadSvc class

MomentUploadSvc - This class implements a web service that receives new moment meta-data by object serialization over HTTP. As each new moment is received, a matching record is created in the "moments" database table.

```

1  /* Susan Gentner
2   * PushToShow Masters Thesis
3   * Georgia State University
4   * March 2009
5   */
6  package edu.gsu.cs.ps;
7
8  import java.io.*;
9  import java.sql.*;
10 import javax.servlet.*;
11 import javax.servlet.http.*;
12 import javax.sql.DataSource;
13 import javax.naming.InitialContext;
14
15 public class MomentUploadSvc extends HttpServlet
16 {
17     protected void processRequest( final HttpServletRequest request,
18                                   final HttpServletResponse response )
19     throws ServletException, IOException
20     {
21         System.out.println( "In MomentUploadSvc.processRequest() ..." );
22
23         Connection cxn = null;
24         CallableStatement stmt = null;
25         final String senderName = "pounce@cs.gsu.edu";
26         final long id = System.nanoTime();
27         final long sentTime = System.nanoTime();
28
29         try
30         {
31             final ObjectInputStream inputStream =
32                 new ObjectInputStream( request.getInputStream() );
33
34             final Moment moment = ((Moment)inputStream.readObject());
35             final String recipients = moment.getRecipients();
36             final String subject = moment.getSubject();

```

```

37     final String body = moment.getBody();
38     final String videoName =
39         moment.getVideoName().substring(
40             moment.getVideoName().lastIndexOf("/") + 1);
41
42     final DataSource dataSource =
43         (DataSource) ((new InitialContext()).lookup(
44             "java:/comp/env/jdbc/postgres" ));
45
46     cxn = dataSource.getConnection();
47     stmt = cxn.prepareStatement( "{ call add_moment( " +
48         "?, ?, ?, ?, " +
49         "?, ?, ?) }" );
50
51     stmt.setLong( 1, id);
52     stmt.setLong( 2, sentTime);
53     stmt.setString(3, senderName);
54     stmt.setString(4, recipients);
55     stmt.setString(5, subject);
56     stmt.setString(6, body);
57     stmt.setString(7, videoName);
58
59     System.out.println( "recordsUpdated = " +
60         stmt.executeUpdate() );
61 }
62 catch( Exception e )
63 {
64     e.printStackTrace();
65 }
66
67 finally
68 {
69     try
70     {
71         if( stmt != null )
72         {
73             stmt.close();
74             stmt = null;
75         }
76     }
77     catch( SQLException e )
78     {
79     }
80
81     try
82     {
83         if( cxn != null )
84         {
85             cxn.close();
86             cxn = null;
87         }
88     }
89     catch( SQLException e )
90     {
91     }
92 }
93 }

```

```
94
95     protected void doGet( final HttpServletRequest request,
96                           final HttpServletResponse response )
97                           throws ServletException, IOException
98     {
99         processRequest( request, response );
100    }
101
102    protected void doPost( final HttpServletRequest request,
103                          final HttpServletResponse response )
104                          throws ServletException, IOException
105    {
106        processRequest( request, response );
107    }
108
109    public String getServletInfo()
110    {
111        return "A web service to receive moment meta-data";
112    }
113 }
114
```

VideoUploadSvc
<i>Attributes</i>
<i>Operations</i>
<pre>protected void processRequest(HttpServletRequest request, HttpServletResponse response) protected void doGet(HttpServletRequest request, HttpServletResponse response) protected void doPost(HttpServletRequest request, HttpServletResponse response) public String getServletInfo()</pre>

Figure 26: VideoUploadSvc class

VideoUploadSvc – This class implements a web service that receives the video portion of a new moment from the end-user's web browser, and saves the video file to disk on the server-side. The new video files are received by multi-part upload method over HTTP.

```

1  /* Susan Gentner
2   * PushToShow Masters Thesis
3   * Georgia State University
4   * March 2009
5   */
6  package edu.gsu.cs.ps;
7
8  import java.io.*;
9  import java.util.*;
10 import javax.servlet.*;
11 import javax.servlet.http.*;
12 import org.apache.commons.fileupload.FileItem;
13 import org.apache.commons.fileupload.FileUploadException;
14 import org.apache.commons.fileupload.disk.DiskFileItemFactory;
15 import org.apache.commons.fileupload.servlet.ServletFileUpload;
16
17 public class VideoUploadSvc extends HttpServlet
18 {
19     protected void processRequest( final HttpServletRequest request,
20                                   final HttpServletResponse response )
21                                   throws ServletException, IOException
22     {
23         response.setContentType("text/html;charset=UTF-8");
24
25         final PrintWriter out = response.getWriter();
26         out.println("<html>");
27         out.println("<head>");
28         out.println("<title>Multipart File Upload Servlet Test</title>");
29         out.println("</head>");
30         out.println("<body>");
31         out.println("<h1>Multipart File Upload Servlet Test Results " +
32                   "from " + request.getContextPath () + "</h1>");
33         try

```



```

34     {
35         final ServletContext servletContext = getServletContext();
36         final String msgsDirectory =
37             servletContext.getRealPath( "/" +
38                 servletContext.getInitParameter("VideosDir") );
39
40         out.println("Server-side real path to upload directory is '"
41             + msgsDirectory + "' <br>" );
42         out.println("Acquiring an iterator for the uploaded file(s) "
43             + "<br>" );
44
45         final DiskFileItemFactory factory =
46             new DiskFileItemFactory();
47         final ServletFileUpload upload =
48             new ServletFileUpload(factory);
49         final List fileItems = upload.parseRequest(request);
50         final Iterator files = fileItems.iterator();
51
52         out.println( "Now, to receive each file.. <br>" );
53         while( files.hasNext() )
54         {
55             final FileItem fileItem = ((FileItem)files.next());
56             final String fileName = fileItem.getName();
57             final InputStream fileInputStream =
58                 fileItem.getInputStream();
59             final File newFile = new File( msgsDirectory, fileName );
60             final FileOutputStream fileOutputStream =
61                 new FileOutputStream( newFile );
62             final BufferedOutputStream bufferedStream =
63                 new BufferedOutputStream( fileOutputStream );
64
65             long fileSize = 0;
66             int fileByte = fileInputStream.read();
67             final long fileStartTime = System.currentTimeMillis();
68             while( fileByte > -1 )
69             {
70                 fileSize++;
71                 bufferedStream.write( fileByte );
72                 fileByte = fileInputStream.read();
73             }
74
75             final long elapsedMillis =
76                 System.currentTimeMillis()
77                 - fileStartTime;
78             final long byteRate =
79                 fileSize * 1000L / elapsedMillis;
80
81             out.print( " &nbsp; &nbsp; &nbsp; &nbsp; " );
82             out.println( "Received file '" + fileName +
83                 "' in " + elapsedMillis + " mS, " +
84                 fileSize + " bytes, " +
85                 byteRate/1024 + " KB/sec. <br>" );
86
87             bufferedStream.flush();
88             bufferedStream.close();
89             fileInputStream.close();
90         }

```

```
91         out.println( "Done receiving file(s) <br>" );
92     }
93     catch( FileUploadException e )
94     {
95         e.printStackTrace(out);
96     }
97     finally
98     {
99         out.println("</body>");
100        out.println("</html>");
101        out.flush();
102        out.close();
103    }
104 }
105
106
107 protected void doGet( final HttpServletRequest request,
108                      final HttpServletResponse response )
109     throws ServletException, IOException
110 {
111     processRequest( request, response );
112 }
113
114 protected void doPost( final HttpServletRequest request,
115                       final HttpServletResponse response )
116     throws ServletException, IOException
117 {
118     processRequest( request, response );
119 }
120
121 public String getServletInfo()
122 {
123     return "A web service to receive the video protion of a moment";
124 }
125 }
126
```

WebCam – This class initializes and receives video from the attached webcam on the end-user's PC.

```

1  /* Susan Gentner
2  * PushToShow Masters Thesis
3  * Georgia State University
4  * March 2009
5  */
6  package edu.gsu.cs.ps;
7
8  import java.awt.*;
9  import javax.swing.*;
10 import java.awt.event.*;
11 import java.awt.image.*;
12 import java.util.Date;
13 import java.text.SimpleDateFormat;
14 import quicktime.*;
15 import quicktime.qd.*;
16 import quicktime.io.*;
17 import quicktime.std.*;
18 import quicktime.std.sg.*;
19 import quicktime.std.image.CodecNameList;
20
21 public class WebCam implements ActionListener
22 {
23     protected int[] pixelData;
24     protected boolean grabbing;
25     protected String fileName;
26     protected String msgUploadURL;
27     protected String fileUploadURL;
28     protected BufferedImage image;
29     protected QDRect cameraImageSize;
30     protected QDGraphics qdGraphics;
31     protected SGVideoChannel videoChannel;
32     protected SGSoundChannel soundChannel;
33     protected MomentUploader momentUploader;
34     protected final CapturePanel capturePanel;
35     protected final SequenceGrabber seqGrabber;
36     protected final SimpleDateFormat simpleDateFormat;

```

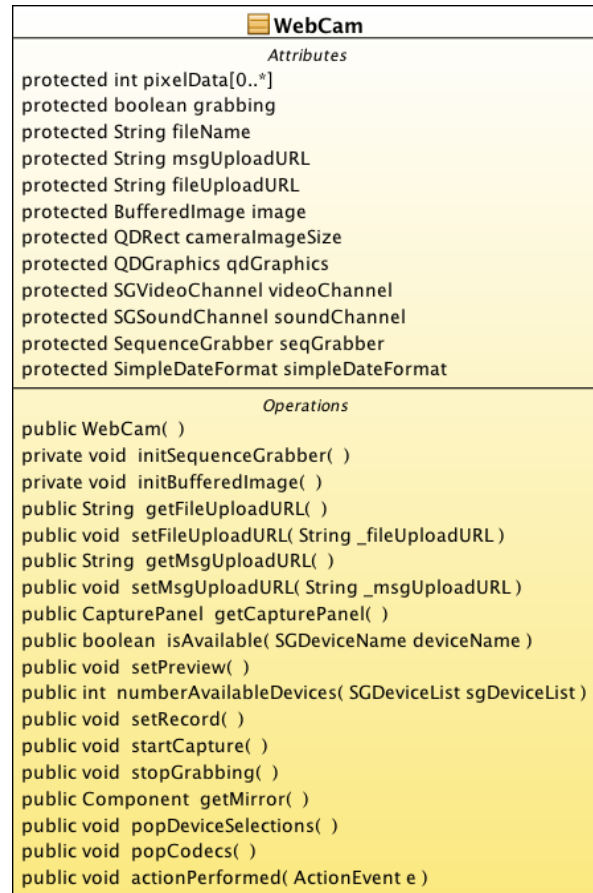


Figure 27: WebCam class

```

37
38 public WebCam() throws Exception
39 {
40     capturePanel = new CapturePanel();
41     seqGrabber = new SequenceGrabber();
42     capturePanel.addActionListener( this );
43     SimpleDateFormat = new SimpleDateFormat( "yyyyMMdd_HH:mm:ss" );
44
45     initSequenceGrabber();
46     initBufferedImage();
47 }
48
49 private void initSequenceGrabber() throws Exception
50 {
51
52     soundChannel = new SGSoundChannel( seqGrabber);
53     soundChannel.setUsage( StdQTConstants.seqGrabPreview |
54                           StdQTConstants.seqGrabRecord );
55
56     videoChannel = new SGVideoChannel( seqGrabber);
57     videoChannel.setCompressorType( 1483297896 );
58     cameraImageSize = new QRect( 640, 480 );
59     qdGraphics = new QDGraphics( cameraImageSize );
60     seqGrabber.setGWorld( qdGraphics, null );
61
62     videoChannel.setBounds( cameraImageSize );
63     videoChannel.setUsage(quicktime.std.StdQTConstants.seqGrabRecord|
64                           quicktime.std.StdQTConstants.seqGrabPreview |
65                           quicktime.std.StdQTConstants.seqGrabPlayDuringRecord );
66
67     videoChannel.setFrameRate( 0 );
68 }
69
70 private void initBufferedImage() throws Exception
71 {
72     final int intsPerRow =
73         qdGraphics.getPixmap().getPixelData().getRowBytes() / 4;
74     final int size = intsPerRow * cameraImageSize.getHeight();
75
76     pixelData = new int[size];
77     final DataBuffer dataBuffer = new DataBufferInt(pixelData, size);
78     final ColorModel colorModel = new DirectColorModel( 32,
79                                                         0x00ff0000,
80                                                         0x0000ff00,
81                                                         0x000000ff );
82
83     final int[] masks = {0x00ff0000, 0x0000ff00, 0x000000ff};
84     final WritableRaster raster = Raster.createPackedRaster(
85         dataBuffer,
86         cameraImageSize.getWidth(),
87         cameraImageSize.getHeight(),
88         intsPerRow, masks, null );
89
90     image = new BufferedImage( colorModel, raster, false, null );
91 }
92
93 public String getFileUploadURL()

```

```
94     {
95         return fileUploadURL;
96     }
97
98     public void setFileUploadURL( String _fileUploadURL )
99     {
100         fileUploadURL = _fileUploadURL;
101     }
102
103     public String getMsgUploadURL()
104     {
105         return msgUploadURL;
106     }
107
108     public void setMsgUploadURL( String _msgUploadURL )
109     {
110         msgUploadURL = _msgUploadURL;
111     }
112
113     public CapturePanel getCapturePanel()
114     {
115         return capturePanel;
116     }
117
118     public boolean isAvailable( final SGDeviceName deviceName )
119     {
120         return((deviceName.getFlags() &
121             StdQTConstants.sgDeviceNameFlagDeviceUnavailable) == 0);
122     }
123
124     public void setPreview() throws QTEException
125     {
126         seqGrabber.setDataOutput( null,
127             StdQTConstants.seqGrabDontMakeMovie );
128     }
129
130     public int numberAvailableDevices( final SGDeviceList sgDeviceList )
131         throws QTEException
132     {
133         int numDevsAvailable = 0;
134         for( int i=0; i<sgDeviceList.getCount(); i++ )
135         {
136             if( isAvailable( sgDeviceList.getDeviceName( i ) ) )
137                 numDevsAvailable++;
138         }
139
140         return numDevsAvailable;
141     }
142
143     public void setRecord() throws QTEException
144     {
145         fileName = System.getProperty( "user.home" ) +
146             "/psv_" + simpleDateFormat.format( new Date() ) +
147             ".mov";
148
149         System.out.println( "Movie file is called '" + fileName + "'" );
150         final java.io.File javaFile = new java.io.File( fileName );
```

```
151     javaFile.deleteOnExit();
152     QTFile movieFile = new QTFile( javaFile );
153
154     seqGrabber.setDataOutput( movieFile,
155                               StdQTConstants.seqGrabToDisk );
156 }
157
158 public void startCapture() throws StdQTEException
159 {
160     seqGrabber.prepare( false, true );
161     seqGrabber.startRecord();
162     grabbing = true;
163
164     Runnable idleCamera = new Runnable()
165     {
166         public void run()
167         {
168             try
169             {
170                 while( grabbing )
171                 {
172                     Thread.sleep( 25 );
173                     synchronized( seqGrabber )
174                     {
175                         seqGrabber.idle();
176                         seqGrabber.update( null );
177                     }
178                 }
179             }
180             catch( Exception ex )
181             {
182                 ex.printStackTrace();
183             }
184         }
185     };
186     (new Thread( idleCamera )).start();
187 }
188
189 public void stopGrabbing()
190 {
191     try
192     {
193         grabbing = false;
194
195         if( seqGrabber != null )
196             seqGrabber.stop();
197     }
198     catch( StdQTEException e )
199     {
200         e.printStackTrace();
201     }
202 }
203
204 public Component getMirror()
205 {
206     final Component component = new Component()
207     {
```

```

208         public void paint( Graphics g )
209         {
210             super.paint( g );
211             g.drawImage( image, 0, 0, this );
212         }
213     };
214 };
215
216 final Runnable imageUpdate = new Runnable()
217 {
218     public void run()
219     {
220         try
221         {
222             while( true )
223             {
224                 synchronized( seqGrabber )
225                 {
226                     qdGraphics.getPixmap().getPixelData().
227                         copyToArray( 0, pixelData,
228                                     0, pixelData.length );
229
230                     component.repaint();
231                 }
232                 Thread.sleep( 10 );
233             }
234         }
235         catch( Exception ex )
236         {
237             ex.printStackTrace();
238         }
239     }
240 };
241 (new Thread( imageUpdate )).start();
242
243     return component;
244 }
245
246 public void popDeviceSelections()
247 {
248     try
249     {
250         int i = 0;
251         int numAvailableDevices = 0;
252         String defaultSelection = null;
253         final SGDeviceList audioDevList =
254             soundChannel.getDeviceList( 0 );
255         final int numAudioDeviceNames = audioDevList.getCount();
256         final int numAudioDevicesAvail =
257             numberAvailableDevices( audioDevList );
258         String[] audioDevNames = new String[numAudioDevicesAvail];
259
260         for( i = 0; i < numAudioDeviceNames; i++ )
261         {
262             final String deviceName =
263                 audioDevList.getDeviceName( i ).getName();
264

```

```

265         if( isAvailable( audioDevList.getDeviceName( i ) ) )
266         {
267             if( deviceName != null )
268             {
269                 System.out.print( "audio #" + i + " is '" +
270                                 deviceName + "'" );
271
272                 if( deviceName.contains( "Camera" ) )
273                 {
274                     defaultSelection = deviceName;
275                     System.out.println( " <-- SELECTED " );
276                 }
277                 else
278                 {
279                     System.out.println( "" );
280                 }
281                 audioDevNames[numAvailableDevices++] =
282                     audioDevList.getDeviceName( i ).
283                     getName();
284             }
285         }
286     }
287
288     capturePanel.getCboAudioInput().setModel(
289         new DefaultComboBoxModel(audioDevNames ) );
290
291     if( defaultSelection != null )
292     {
293         capturePanel.getCboAudioInput().setSelectedItem(
294             defaultSelection );
295
296         System.out.println( "Set soundChannel device to: '" +
297                             defaultSelection + "'" );
298     }
299
300     numAvailableDevices = 0;
301     defaultSelection = null;
302     SGDeviceList videoDevList = videoChannel.getDeviceList( 0 );
303
304     final int numVideoDeviceNames = videoDevList.getCount();
305     final int numVideoDevicesAvail =
306         numberAvailableDevices(videoDevList);
307
308     String[] videoDevNames = new String[numVideoDevicesAvail];
309     for( i = 0; i < numVideoDeviceNames; i++ )
310     {
311         final String deviceName =
312             videoDevList.getDeviceName( i ).getName();
313         if( isAvailable( videoDevList.getDeviceName( i ) ) )
314         {
315             if( deviceName != null ) // usually print this
316             {
317                 System.out.print( "video #" + i + " is '" +
318                                 deviceName + "'" );
319                 if( deviceName.contains( "USB" ) )
320                 {
321                     defaultSelection = deviceName;

```



```

322         System.out.println( " <-- SELECTED " );
323     }
324     else
325     {
326         System.out.println( "" );
327     }
328     videoDevNames[numAvailableDevices++] =
329         videoDevList.getDeviceName( i ).getName();
330     }
331 }
332 }
333
334 capturePanel.getCboVideoInput().setModel(
335     new DefaultComboBoxModel( videoDevNames ) );
336
337 if( defaultSelection != null )
338 {
339     capturePanel.getCboVideoInput().
340         setSelectedItem( defaultSelection );
341     System.out.println( "Set videoChannel device to: '" +
342         defaultSelection + "'" );
343 }
344 }
345 catch( QTException qte )
346 {
347     qte.printStackTrace();
348 }
349 }
350
351 public void popCodecs()
352 {
353     try
354     {
355         CodecNameList codecNameList = new CodecNameList( 0 );
356         String[] codecNames = new String[codecNameList.getCount()];
357         String defaultSelection = null;
358
359         for( int i = 0; i < codecNameList.getCount(); i++ )
360         {
361             final String codecName =
362                 codecNameList.getNth( i ).getTypeName();
363
364             if( codecName.contains( "Xiph" ) )
365             {
366                 defaultSelection = codecName;
367                 System.out.println( "codec algorithm #" + i + " is '"
368                     + defaultSelection +
369                     "' <-- SELECTED " );
370             }
371             else
372             {
373                 System.out.println( "codec algorithm #" + i + " is '"
374                     + codecName + "'" );
375             }
376             codecNames[i] = codecName;
377         }
378         capturePanel.getCboVideoCompression().setModel(

```

```

379         new DefaultComboBoxModel(codecNames));
380
381         if( defaultSelection != null )
382         {
383             capturePanel.getCboVideoCompression().
384                 setSelectedItem( defaultSelection );
385         }
386     }
387     catch( QTException qte )
388     {
389         qte.printStackTrace();
390     }
391 }
392
393 public void actionPerformed( ActionEvent e )
394 {
395     if( e.getSource() == capturePanel.getRecordButton() )
396     {
397         stopGrabbing();
398         try
399         {
400             if( capturePanel.getRecordButton().getText().
401                 equals("Record"))
402             {
403                 grabbing = true;
404                 capturePanel.getRecordButton().setText( "Stop" );
405                 capturePanel.getCboAudioInput().setEnabled( false );
406                 capturePanel.getCboVideoInput().setEnabled( false );
407                 capturePanel.getCboVideoCompression().
408                     setEnabled( false );
409                 capturePanel.getSendButton().setEnabled( false );
410                 setRecord();
411             }
412             else if( capturePanel.getRecordButton().getText().
413                 equals("Stop"))
414             {
415                 grabbing = false;
416                 capturePanel.getRecordButton().setText( "Record" );
417                 capturePanel.getCboAudioInput().setEnabled( true );
418                 capturePanel.getCboVideoInput().setEnabled( true );
419                 capturePanel.getCboVideoCompression().
420                     setEnabled( true );
421                 capturePanel.getSendButton().setEnabled( true );
422                 setPreview();
423             }
424         }
425         startCapture();
426     }
427     catch( QTException e1 )
428     {
429         e1.printStackTrace();
430     }
431 }
432
433 else if( e.getSource() == capturePanel.getCboAudioInput() )
434 {
435

```

```

436     System.out.println( "cboAudioInput changed to : '" +
437                         capturePanel.getCboAudioInput().
438                         getSelectedItem().toString() + "'" );
439     try
440     {
441         if(seqGrabber.isPreviewMode() || seqGrabber.isRecordMode())
442             stopGrabbing();
443
444         soundChannel.setDevice( capturePanel.getCboAudioInput().
445                                 getSelectedItem().toString() );
446         startCapture();
447     }
448     catch( Exception ex )
449     {
450         System.out.println( "Cound not set the sound device" );
451         ex.printStackTrace();
452     }
453 }
454
455 else if( e.getSource() == capturePanel.getCboVideoInput() )
456 {
457     System.out.println( "cboVideoInput changed to : '" +
458                         capturePanel.getCboVideoInput().
459                         getSelectedItem().toString() + "'" );
460
461     try
462     {
463         if(seqGrabber.isPreviewMode() || seqGrabber.isRecordMode())
464             stopGrabbing();
465
466         videoChannel.setDevice( capturePanel.getCboVideoInput().
467                                 getSelectedItem().toString() );
468         startCapture();
469     }
470     catch( Exception ex )
471     {
472         System.out.println( "Cound not set the video device" );
473         ex.printStackTrace();
474     }
475 }
476
477 else if( e.getSource() == capturePanel.getCboVideoCompression() )
478 {
479     System.out.println( "cboVideoCompression changed to : '" +
480                         capturePanel.getCboVideoCompression().
481                         getSelectedItem().toString()
482                         + "'" );
483
484     try
485     {
486         final CodecNameList codecNameList = new CodecNameList(0);
487         final int ctype =
488             codecNameList.getNth( capturePanel.
489                                   getCboVideoCompression().
490                                   getSelectedIndex() ).
491                                   getCType();
492

```

```

493         if(seqGrabber.isPreviewMode() || seqGrabber.isRecordMode())
494             stopGrabbing();
495
496         videoChannel.setCompressorType( ctype );
497         startCapture();
498     }
499     catch( Exception ex )
500     {
501         ex.printStackTrace();
502     }
503 }
504
505 else if( e.getSource() == capturePanel.getSendButton() )
506 {
507     System.out.println( "Send button was clicked." );
508
509     final Moment moment = new Moment();
510     moment.setSenderName( "pounce@cs.gsu.edu" );
511     moment.setRecipients( capturePanel.getTfRecipient().
512                             getText() );
513     moment.setSubject( capturePanel.getTfSubject().
514                             getText() );
515     moment.setBody( capturePanel.getTaBodyText().
516                             getText() );
517     moment.setVideoName( fileName );
518     momentUploader = new MomentUploader( moment, msgUploadURL,
519                                         fileName, fileUploadURL );
520
521     (new Thread( momentUploader )).start();
522     System.out.println( "Started thread to upload message with "+
523                         "subject line '" +
524                         moment.getSubject() + "'" );
525 }
526
527 else if( grabbing )
528 {
529     try
530     {
531         seqGrabber.idle();
532         seqGrabber.update( null );
533     }
534     catch( QTException qte )
535     {
536         qte.printStackTrace();
537     }
538 }
539 }
540
541 }
542

```

ListMoments.jsp – This web page lists all moments that are addressed to a specified recipient. The moments are presented in an HTML table that has click-able rows. When the user clicks a row, a new tab will be opened in the web browser with a matching ViewMoment page for the moment entry that was clicked.

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 TRANSITIONAL//EN">
2
3  <!--
4  Susan Gentner
5  PushToShow Masters Thesis
6  Georgia State University
7  March 2009
8  -->
9
10 <%@ page import="java.sql.*" %>
11 <%@ page import="java.text.SimpleDateFormat" %>
12 <%@ page import="javax.sql.DataSource" %>
13 <%@ page import="javax.naming.InitialContext" %>
14
15 <html>
16 <head>
17     <meta http-equiv="content-type" content="text/html; charset=UTF-8">
18     <title> PushToShow Inbox </title>
19     <link rel='stylesheet' type='text/css' href='sample.css' />
20 </head>
21 <body>
22     <form>
23     <center>
24     <table width="1005px" height="550px" border="6px" cellpadding="1px">
25         <tr valign="top"><td>
26             <table border="1px" width="100%">
27                 <thead>
28                     <tr>
29                         <th width="20%">Date</th>
30                         <th width="20%">Sender</th>
31                         <th>Subject</th>
32                     </tr>
33                 </thead>
34                 <tbody>
35
36 <%
37     Connection cxn = null;
38     String recipient = null;
39     ResultSet rs = null;
40     String takeMomentURL = null;
41     PreparedStatement pstmt = null;
42     final SimpleDateFormat simpleDateFormat =
43         new SimpleDateFormat("h:mm a      E dd MMM, yyyy");
44     String shareUrl = null;

```

```

45     try
46     {
47
48         final ServletContext servletContext = this.getServletContext();
49         final int serverPort = request.getServerPort();
50         final String serverName = request.getServerName();
51         final String path = request.getServletPath();
52         takeMomentURL = "http://" + serverName + ":" + serverPort +
53             servletContext.getContextPath() +
54             "/TakeMoment.jsp?id=";
55         shareUrl = "http://" + serverName + ":" + serverPort +
56             servletContext.getContextPath() +
57             "/ShareMoment.jsp";
58         recipient = request.getParameter("recipient");
59
60         DataSource dataSource = (DataSource)((new InitialContext()).
61             lookup("java:/comp/env/jdbc/postgres"));
62         cxn = dataSource.getConnection();
63
64         pstmt = cxn.prepareStatement("SELECT * FROM list_moments('" +
65             recipient + "')");
66         rs = pstmt.executeQuery();
67
68         while(rs != null && rs.next())
69         {
70             out.println("<tr onmouseover='this.style.color=\"red\";' +
71                 "style.cursor=\"pointer\";this.style.background"+
72                 "=\"lightblue\"' onMouseOut = 'this.style.color="+
73                 "\"black\";this.style.background=\"white\"' "+
74                 "onclick=\"window.open('"+ takeMomentURL +
75                 rs.getString( "id" )+"');return false;\">");
76             out.println("<td>" + simpleDateFormat.
77                 format(new Date(Long.parseLong(
78                     rs.getString( "senttime" ))/1000000)) +
79                 "</td>");
80             out.println("<td>" + rs.getString( "sendername" ) + "</td>");
81             out.println("<td>" + rs.getString( "subject" ) + "</td>");
82             out.println("</tr>");
83         }
84     }
85     catch( SQLException e )
86     {
87         e.printStackTrace();
88     }
89
90     finally
91     {
92         try
93         {
94             if( rs != null )
95                 rs.close();
96         }
97         catch( SQLException e )
98         {
99             }
100
101     try

```


ShareMoment.jsp – This web page allows the end user to capture, compose, and send a moment with just a couple of quick mouse clicks.

```

1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3  "http://www.w3.org/TR/html4/loose.dtd">
4
5  <!--
6  Susan Gentner
7  PushToShow Masters Thesis
8  Georgia State University
9  March 2009
10 -->
11
12 <html>
13     <head>
14         <meta http-equiv="Content-Type"
15             content="text/html; charset=UTF-8">
16         <title>Push to Show Applet</title>
17     </head>
18     <body>
19         <center>
20             <table width="1005px" border="6px"cellpadding="1px">
21                 <tr><td>
22                     <table>
23                         <tr>
24                             <td>
25                                 <%
26                                 String userAgent = request.getHeader("user-agent");
27                                 boolean isMac = userAgent.toLowerCase().
28                                     contains("mac os x");
29
30                                 if(isMac)
31                                 {
32                                     <applet archive="psa.jar"
33                                         code="edu.gsu.cs.ps.Applet"
34                                         width="1000"
35                                         height="550" >
36
37                                         <param NAME="FileUploadServletURL"
38                                             VALUE="http://<%= request.getServerName ()
39                                                 + ':'
40                                                 + request.getServerPort ()
41                                                 %>/psv/VideoUploadSvc">
42                                         <param NAME="MsgUploadServletURL"
43                                             VALUE="http://<%= request.getServerName ()
44                                                 + ':'
45                                                 + request.getServerPort ()
46                                                 %>/pss/MomentUploadSvc">
47
48                                         This application needs a Java-enabled browser to run.
49                                     </applet>
50                                 <%
51                                 }

```



```
52         else
53         {
54             %>
55             Sorry this application is currenty <b>Mac</b> only.
56             <%}%>
57         </td>
58     </tr>
59 </table>
60 </td></tr>
61 </table>
62 <p>
63     <%= new java.util.Date() %>
64 </p>
65 </center>
66 </body>
67 </html>
68
```

TakeMoment.jsp – This web page displays a moment for a recipient.

```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 TRANSITIONAL//EN">
2
3 <!--
4 Susan Gentner
5 PushToShow Masters Thesis
6 Georgia State University
7 March 2009
8 -->
9
10 <%@ page import="java.sql.*" %>
11 <%@ page import="javax.sql.DataSource" %>
12 <%@ page import="javax.naming.InitialContext" %>
13 <%@ page import="edu.gsu.cs.ps.Moment" %>
14
15
16 <%
17     Connection cxn = null;
18     PreparedStatement pstmt = null;
19
20     Statement stmt = null;
21     Moment moment = null;
22     ResultSet rs = null;
23     String senderName = null;
24     String recipients = null;
25     String subject = null;
26     String body = null;
27     String videoName = null;
28
29
30     try
31     {
32         final ServletContext servletContext = this.getServletContext();
33         final long id = Long.valueOf( request.getParameter( "id" ) );
34
35         DataSource dataSource = (DataSource)((new InitialContext()).
36             lookup("java:/comp/env/jdbc/postgres"));
37         cxn = dataSource.getConnection();
38
39         pstmt = cxn.prepareStatement("SELECT * FROM get_moment(" +
40             id + ");" );
41         rs = pstmt.executeQuery();
42
43         if( rs.next() )
44         {
45             moment = new Moment( rs );
46             senderName = moment.getSenderName();
47             recipients = moment.getRecipients();
48             subject = moment.getSubject();
49             body = moment.getBody();
50             videoName = moment.getVideoName();
51
52             if( body.contains("\n")
53                 body = body.replaceAll("\n", "<br/>");
54         }

```

```

55     }
56     catch( SQLException e )
57     {
58         e.printStackTrace();
59     }
60
61     finally
62     {
63         try
64         {
65             if( rs != null )
66                 rs.close();
67         }
68         catch( SQLException e )
69         {
70         }
71
72         try
73         {
74             if( pstmt != null )
75                 pstmt.close();
76         }
77         catch( SQLException e )
78         {
79         }
80         try
81         {
82             if( cxn != null )
83                 cxn.close();
84         }
85         catch( SQLException e )
86         {
87         }
88     }
89 %>
90 <html>
91 <head>
92     <meta http-equiv="content-type"
93           content="text/html; charset=UTF-8">
94     <title><%= moment.getSubject() %></title>
95     <link rel='stylesheet' type='text/css' href='sample.css' />
96 </head>
97 <body>
98     <form>
99         <center>
100
101         <table width="1005px" border="6px" cellpadding="1px">
102         <tr><td>
103         <table border="1px" width="100%">
104         <tr>
105             <td rowspan="5" width="640">
106                 <div class="classname">
107                     <object classid=
108                         "clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
109                         width="640" height="490"
110                         codebase=
111                         "http://www.apple.com/qtactivex/qtplugin.cab">

```

```

112         <param name="autoplay" VALUE="false"/>
113         <param name="controller" VALUE=true/>
114         <embed src="<%= "http://" + request.getServerName() +
115                 ":" + request.getServerPort() +
116                 "/psv/msgs/" +
117                 videoName%>"
118                 width="640" height="500"
119                 controller="true" fullscreen="full">
120         </embed>
121     </object>
122 </div>
123 </td>
124 <td valign="top" colspan="2" height="104" align="center">
125     
127 </td>
128 </tr>
129 <tr height ="25px">
130     <td align="right"><b>Subject:</b></td>
131     <td> <%= subject %> </td>
132 </tr>
133 <tr height ="25px">
134     <td align="right"><b>From:</b></td>
135     <td> <%= senderName %> </td>
136 </tr>
137 <tr height ="25px">
138     <td align="right"><b>To:</b></td>
139     <td> <%= recipients %> </td>
140 </tr>
141 <tr>
142     <td colspan="2" valign="top">
143         <%= body %>
144     </td>
145 </tr>
146 </table>
147 </td>
148 </tr>
149 </table>
150
151 <p>
152 <%= new java.util.Date() %>
153 </p>
154 </center>
155 </form>
156 </body>
157
158 </html>
159

```

SQL to create the moments table and indexes

```
-- Table: moments

CREATE TABLE moments

(
    id bigint NOT NULL,
    senttime bigint NOT NULL,
    sendername character varying NOT NULL,
    recipients character varying NOT NULL,
    subject character varying NOT NULL,
    bodytext character varying NOT NULL,
    videoname character varying NOT NULL,
    CONSTRAINT messages_pkey PRIMARY KEY (id)
)

WITH (OIDS=FALSE);

ALTER TABLE moments OWNER TO postgres;

-- Index: recipients_idx

CREATE INDEX recipients_idx

    ON moments

    USING btree

    (recipients);

-- Index: senttime_idx

CREATE INDEX senttime_idx

    ON moments

    USING btree

    (senttime);
```

web.xml is a deployment descriptor for the pss web application.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6
7     <session-config>
8         <session-timeout>30</session-timeout>
9     </session-config>
10
11     <servlet>
12         <servlet-name>MomentUploadSvc</servlet-name>
13         <servlet-class>edu.gsu.cs.ps.MomentUploadSvc</servlet-class>
14     </servlet>
15
16     <servlet-mapping>
17         <servlet-name>MomentUploadSvc</servlet-name>
18         <url-pattern>/MomentUploadSvc</url-pattern>
19     </servlet-mapping>
20
21     <resource-ref>
22         <description>PostgreSQL Datasource example</description>
23         <res-ref-name>jdbc/postgres</res-ref-name>
24         <res-type>javax.sql.DataSource</res-type>
25         <res-auth>Container</res-auth>
26     </resource-ref>
27
28     <welcome-file-list>
29         <welcome-file>ListMoments.jsp</welcome-file>
30     </welcome-file-list>
31
32 </web-app>
33
```

web.xml is a deployment descriptor for the psv web application.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6
7     <context-param>
8         <description>Video upload directory</description>
9         <param-name>VideosDir</param-name>
10        <param-value>msgs</param-value>
11    </context-param>
12
13    <servlet>
14        <servlet-name>VideoUploadSvc</servlet-name>
15        <servlet-class>edu.gsu.cs.psv.VideoUploadSvc</servlet-class>
16    </servlet>
17    <servlet-mapping>
18        <servlet-name>VideoUploadSvc</servlet-name>
19        <url-pattern>/VideoUploadSvc</url-pattern>
20    </servlet-mapping>
21
22    <session-config>
23        <session-timeout>
24            30
25        </session-timeout>
26    </session-config>
27    <welcome-file-list>
28        <welcome-file>index.jsp</welcome-file>
29    </welcome-file-list>
30
31 </web-app>
32
```