2011

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

FACE RECOGNITION APPROACH TO AUTOMATING FILM ANALYSIS

Dominique Jackson

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
Of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2011

Approved By

Advisory Committee

Dr. Karl Ricanek Dr. Devon Simmonds

Dr. Judith Gebauer, Chair

Accepted By

Dean, Graduate School

Table of Contents

Abstract 4

Introduction 4

- Problem Statement 5
- Features/Benefits 6
- Related Work 7

System Overview 8

- Technologies 10
- Limitations 11
- Description of Completed Project 12

Implementation 12

- System Requirements 12
- Registration 13
- Security 13
- Face Mining & User interface 13
- Character Identification Process 14
  - Movie Selection 14
  - Select a Character 14
  - Character Review 14
  - Character Interactions 15
- System Architecture 15
- Obstacles 15
- Testing and Analysis 16

Conclusion 16

- Observations 17
- Future Work 17

Works Cited 21

- Appendix 24
  - Use Cases 24
  - Actor Diagram 27
  - Resources 28

Database Diagram 28
Multi-tier Web Application 29
Structured Query Language Procedures 29
User Guide 34
Source Code 42

Abstract

Film scholars compel us to develop automated approaches to support scientific film studies. Current related organizations and researchers focus on developing video analysis software which provides management, organization, data mining, querying and retrieval of videos or video content (efficient databases and unlimited annotating). Other features these researchers desire includes 'digital formalism' and collaboration. The desired video analysis systems are either lacking desired functionality, have not been developed due to lack of funds or approval, are using closed sources, or are mainly geared towards sports or physics. Although the technologies vary in many ways, majority have one thing in common, biometrics. For example, Transana has incorporated voice recognition to transcribe dialogue into files. The intention of this paper is to propose an automated film analysis system which will help familiarize film scholars with a film and reduce the time and manual labor spent on analysis.

Introduction

What would a movie be without its characters? Characters can set the speed of the movie as well as propel the story (Bordwell & Thompson, 2004). In using face recognition, faces can be extracted to aid in the critique and analysis of the film. This paper explains the implementation of a video analysis system which supports the automation of film analysis in the areas or terms of why individuals were casted concerning their traits which capture the essence of the character intended, the design aspects of a film, and framing(Part 2: Mise-en-scene, 2002)(YALE FILM STUDIES, 2002). Throughout a movie, new characters emerge onto the scene and it is often difficult to keep up with each character while discovering their importance or relevance to a particular film. This system will be used to keep track of all characters and

their appearance(s) in the film. This information will make it easier to reveal character transformations as well as other analytical conclusions. This system will also allow users to document and store annotations to be viewed by other colleagues.

Problem Statement

The task of film analysis is time-consuming and manual (Schild, 2007). One of the professors within the film studies department stated that when analyzing a film, it is normally reviewed in its entirety at least three times not including the number of times a certain scene is examined. There is an inability to easily analyze and compare films over time. Considering film analysis involves the hidden aspects which are not always seen, film scholars often have to start from the very beginning even when evaluating a film that has already been analyzed or search for previously written papers. Film scholars are not able to easily access annotations made in reference to a film or their colleagues in order to discuss complex topics.

Films are often stored in a variety of places and are typically not well organized to provide for efficient availability or accessibility. Research has been conducted to examine how data has been collected, managed and indexed in the past and the importance of bringing it up to date in order alleviate prior difficulties in functionality (Hampapur, 1999). One of the reasons for this lack in consistency in file storage is poor execution (Juwei, Plataniotis, & Venetsanopoulos, Face recognition using LDA-based algorithms, 2003). The main objective of storing metadata in a central place is to provide a highly detailed, time-encoded, comprehensive range of data for users to search through and find relevant content with pinpoint accuracy (Deep Video Indexing, 2009). The data provided would then be incorporated into producing results which match users' interests with the videos they are searching for, and

similar content which is a lot of data to manage. The overall success of this system depends on the management and delivery of data. XML based methods can be used to index content and better manage data (Mittal & Walker, 2006).

Features/Benefits

The main objectives of this system are to save time and reduce manual labor. In order to save time, users can process the film ahead of time using the mentioned system. It is estimated that film analysts use at least two hours just trying to familiarize with the movie itself. Once the movie has been processed, they will be able to see the characters and the scenes they are in. The system will inform users what scenes a character shared with another character. It is important to examine small chunks of a film at a time, which is exactly what the system will allow (cfloud, 1999-2011). In actuality, users might only save an hour using this system, but other benefits will include automation and the ability to store and organize data. The system will also allow users to view their peers' analyses.

Professors and students do not have a way to readily access films for review. A professor mentioned the difficulty students have in keeping up with the films they've created themselves. Students are searching through their numerous created tapes to find a current or old film for review. Students are not able to easily store their created films for reuse. Instead, technology could be used make films accessible at any time, so instead of spending hours or days searching for a film, it could be spent on reviewing the film itself.

Individuals who will benefit from this system are film scholars/researchers, faculty, students, actors, directors, producers and possibly choreographers. The system will make it

easier for users to get familiarized with the film which will cut down on the amount of time spent doing analysis. With any work of art, it helps to have words to go along with them to provide understanding, which is exactly what the system will do. Years later, when a user returns to a film, they will not have to start from the very beginning and they can also analyze how a thought process or a moral lesson will change over time. The identification portion of the system will allow users to focus on the interaction between characters. This will also allow the comparison of characters in other films whether looking at the character in the movie or the actor.

Related Work

There are numerous researchers who make it their goal to create systems that perceive as humans do. These systems are trained to recognize faces similar to the way humans recognize individuals as well as strive for even greater accuracy. There have been many different algorithms and proposed methods for face recognition (Juwei, Plataniotis, & Venetsanopoulos, Face recognition using LDA-based algorithms, 2003)(Juwei, Plataniotis, & Venetsanopoulos, Face recognition using kernel direct discriminant analysis algorithms, 2003). Students have also made evaluations of current face recognition methods. The first successful example of a face recognition technology is the Eigen face developed by Sirovich and Kirby (EigenFace, 2010). It is probably also the most commonly used.

Video processing, the segmentation of video streams, has been a work in progress. It enables videos to be easily captured and shared (Fry & Reas, 2002). Where and how to efficiently segment videos is determined on a case by case basis. A video sequence is divided into segments which is a basic part of indexing. There are numerous ways to index, store and

retrieve video elements. A lot of exploration has been done to not only determine effective ways to create content management systems but also how they work together (Yoshitaka & Ichikawa, 1999)(WCER, 2005-2009). The main goal for any of these automated systems is to increase speed and accuracy in comparison to performing the actions manually. Anytime a user retrieves information they should be able to browse all records for a particular record. For example, most of the selections made will be in regards to a certain movie, so they will be able to search through a list of movies and type in the name of the movie, director, etc. and search for the film when there is a large amount of movies listed. Overall success of these automated systems will depend upon how well they are organized (Yoshitaka, Ishii, Hirakawa, & Ichikawa, 1997).

Macromedia developed an editing software application which allows users to capture video and record it onto a hard drive for editing, processing and output in various formats (Final Cut Pro, 2011). Final Cut Pro supports an open standard-based XML interchange format. The movie industry is in the midst of standardizing their approach to digitalization of cinema and uses XML as a part of their approach (Mittal & Walker, 2006). Final Cut Pro offers simple tools for users to read and manipulate the data stored.

System Overview

The main components of this system are identification, annotation, archiving, extraction and collaboration. In identification, users will be able to identify characters throughout the film. After each film is processed and the data has been inserted the characters are displayed as well as an image gallery consisting of other images of the character throughout the film. Each

character should only be identified once and can be corrected if that is not the case. Users will specify the name of the character, the name of the actor the character is played by, and any details to describe the character. After characters have been identified, other users will be able to click on a character and view information about that character. Information revealed will include the name of the character, the actor they are played by, details describing the character's behavior or role in the film, what scenes or frames they are in, and who they appeared in scenes with.

Users will have to log into the system in order to use it. Having the users log in will assist in keeping track of the users of the system and make it easy to provide documentation of who provided which documents or annotations. It will ensure proper credit is given where it is due.

Users will be able to upload short films directed by students which can be stored using this system. Anytime a movie is uploaded, users are required to enter a certain number of details related to the film. These details would include movie title, the name of the director, year and country. Users will be able to list other details if they so desire.

The front-end of this multi-tiered system is created in ASP.NET and written in C# which is compatible with the face mining technology used. This enables users to access the program from anywhere at any time. It is also chosen for its ease of use which allows for transition from page to page to view the information in a simplistic way.

Considering there is a need to manage metadata, sequel server is a good choice because it will support XML data as well as relational data. XML is the chosen metadata standard. Sequel server will be used store data and to create queries to retrieve the data. This system requires a

database which will serve two main purposes. For the first purpose, the database will be used to store the information gathered from the user about the film. It will also be used to store the films and documents. The second purpose of the database is solely to store the photos and information collected from the processed films. Finally, there is the server. A server is a software system running on a dedicated computer. The server will do all the work behind the scenes because the face recognition software will not be run from the website.

Technologies

Biometrics involves recognizing humans by physical or behavioral traits. In computer science, biometrics are normally used in terms of identity access management which involves authentication to grant or deny access to data or resources or access control which involves authority to particular areas or resources. Face recognition is an example of a physical trait. Face recognition is the ability to verify or identify a person from a digital image.

Video indexing involves the ability to store and retrieve movies as well as related information such as documents and annotations. Retrieval of videos and related information is dependent upon the ability to search for them. The success of querying and retrieval is dependent upon the metadata. Metadata is data used to describe data. Metadata standards will be used to store and retrieve data. An example of a metadata standard is MPEG-7 and an example tool is XML.

An example deployment of face recognition is a face mining technology developed by Pittsburgh Pattern Recognition (Pittsburgh Pattern Recognition, 2004-2011). Their technology applies face detection, tracking and recognition. Face detection is the process of automatically

locating human faces (Face Detection, 2004-2011). Face tracking is the ability to build a sequence of face instances across time (Face Tracking, 2004-2011). Face recognition is the ability to determine if two people are the same person (Face Recognition, 2004-2011). These applications extract the faces and cluster them into a small number of same-person groupings. The face mining demo enables users to navigate to scenes of interest, embed scenes in other media and summarize occurrences of characters.

Transana is a software program geared towards researchers seeking to analyze digital video or audio data. Functionalities include data management, search optimization and collaboration. Transana is additionally beneficial because it is an open source and a cross-platform (WCER, 2005-2009). Transana has also experimented with voice recognition software to improve their current applications. Voice recognition will allow automated transcribing which is more efficient than manual transcribing (Voice Recognition, 2005-2009).

Dartfish is a commercial company which develops digital imaging applications. Their main research focus is video and image processing (About Us, 1999-2011). Their goal is to provide video accessibility, transform videos into useful documents, and to set the de facto video standard (Vision, 1999-2011). Dartfish developed numerous applications, such as Dartfish TeamPro. Dartfish TeamPro aims to simplify analysis by providing functionality which includes tagging, visual feedback, and sharing of data (Dartfish TeamPro, 1999-2011).

Limitations

Topics of concern are identification errors and less anticipated results. Currently, the system processes video results extremely slow. Users would only be able to examine seconds of

a film within a reasonable amount of time. In order to speed up the processing time, the code for face recognition will have to be revised but there is not a guarantee the video will process within a reasonable amount of time. It is also possible for characters to be left out of the processed results or for characters to be identified incorrectly and stored twice.

Description of Completed Project

The end result of this project is the implementation of the proposed system. The success of this implementation results in the completion of one use case. One use case serves the purpose of proving the system is feasible. The use case to be executed is character identification. Deliverables will include the proposal, diagrams, the system and the final document write-up for the project.

Implementation

System Requirements

- Provide an interface to allow a user to analyze a film
- Upload a movie along with its details
- Process the video and return its results
- Select a character and input the character details
- Store data for review and revisions
- Data to be stored in the database: personal information about the user, information about the characters, information about the films, and images of characters, actors, and scenes/frames
- The system will allow the user to add or edit characters which may not have been identified correctly

Registration

In order to register, the user provides a user name, first name, last name, email address, password, and whether they are a student or faculty member. Once logged in they can view and choose which actions they would like to perform. Each user will only have to register once, but has to log in every time they use the system. When the user registers a record is generated in the database which will allow users to be remembered when they log into the website in the future. This also enables a record to be kept of who performed what actions.

Security

It is important to protect the rights of the individuals using this system concerning their thoughts and ideas. Requiring users to be logged in and agree to terms as far as use of available resources contributes to the protection of each user as well as the resources being used.

Face Mining & User interface

This system was developed with Pittsburgh Pattern Recognition's face mining technology, C# programming language, and XML, extensible markup language. The recognition across tracks application processes a video sequence and selects faces from each track to produce templates. A few templates are then chosen to represent each track and are examined to determine which tracks belong to the same subject. Results from the processed video is a video with the characters outlined, grouped and organized thumbnails of the characters, and information detailing tracks and frames the character was in. Use of a website approach allows for a smooth transition between pages and flow of events. The data is stored in sequel server as well as XML. The character information is displayed and can be updated by the user.

Character Identification Process

This section provides a full scenario of the implementation conducted on the character identification use case to demonstrate the intended flow of events to identify a character. Screenshots can be found in the appendix.

Movie Selection

The first time a user is registered, they are directed to the main page which consists of movies to pick from and a list of options. One of the options is the ability to upload a movie. The specified format of movies for this system is mp4. The user is required to enter the title, year produced, country produced in, the director, genre, length of the film and the producer. Users can also search for a movie by the name of the director, title of the film, or year produced.

Select a Character

Users can browse the system for films that have already been entered. Once a film is selected a list of characters from the movie are displayed and can be chosen from. Each character also has a character profile which consists of a gallery containing all pictures retrieved of each character.

Character Review

Once a character is chosen, a list of details will be displayed pertaining to the selected character. This information may or may not be already listed. The user will be able to input information such as the name of the character, name of the actor, and a description of the character's importance to the movie. If the information is incorrect such as a typo or a falsely identified character, it can be revised. If the system identifies one character as two different people or identifies different characters as the same character, the user can correct the error.

Character Interactions

Details provided for each character will be the track, start frame and end frame. There will also be a list of characters the selected character interacted with. Users will be able to click on these characters and view their profiles as well.

System Architecture

The three main parts of this system is the face mining technology written in C, the website written in C#, and the connection between them. The face mining technology extracts the data from the video such as images and data related to tracks. The windows service is used to monitor a folder to determine when a video is uploaded. Once a video is uploaded, it is then processed. The results processed from the video are exported into an XML file and thumbnails which are stored in the directory provided. A query is then used to determine which videos uploaded have been processed. Once the results are processed, the user can click a button to insert them into the database. Once the results are in the database the website can then access the information and display it for users to view. The connection between the technologies pertains to the way data is stored. In this case, data is stored in sequel server as well as XML. XML is used mainly to store the information returned from processing the video. This information consists of tracks, start frames, and end frames. Sequel server is used to store the information added about the user, details added by the user about the characters or scenes, and data retrieved from the processed videos such as thumbnails.

Obstacles

A desired improvement of this system is the amount of time it takes to process a video. To process a film about a minute in length only takes a matter of seconds. However, to process a regular length film close to two hours long would possibly take a couple of days depending on

the processor used. Another issue is the accuracy of the data returned. It is possible for tracks to not be identified due to lighting, angles or other interferences. It is also possible for two characters to not be recognized as the same person due to facial expressions or aging. These things will hopefully be corrected in the future. Something unexpected was the inability to retrieve more information from the results of processed videos. It was hoped to identify each frame and scene that a character is in instead of just the starting and end frame. There was also a lack of time stamps for when characters appeared or when each frame or scene begins.

Testing and Analysis

At least five videos have been tested on the system. It takes about seven minutes to process a minute length of film. Users do not have to wait for a video to process to upload another video. Out of five videos, two of the videos had an error in character identification. One of the characters has been identified as either two or three different characters instead of one. Besides this error, each character has been accounted for and only counted once. Some of the images do appear to be a little blurry.

Conclusion

The purpose of this system is to automate a small portion of film analysis and save film scholars time. A key benefit of this system is to familiarize film analysts with the film which will reduce the amount of times and the length of a film is looked at. It is uncertain how much time this system will be able to help users save, but it will reduce the manual labor in analysis and help users organize their thoughts.

Observations

The system performs better than expected as far as timing as concerned. It was anticipated it would take days to process a regular length video when in actuality it only takes about half a day. On the downside the information returned isn't as detailed as expected. For instance, timestamps were not retrieved for beginning and ending of frames or scenes.

This system served the purpose in proving the ability to provide a link between returned results of a processed video and an interface to present them to the user. The most difficult task of this assignment was to provide a way to first process the results automatically and then retrieve the results as soon as they were outputted. Requiring the user to process the video manually or ask them to rummage through folders containing thumbnails would defeat the purpose of this system. The automated implementation of this system mentioned earlier allowed for a seamless transition.

Future Work

This system serves as a starting point. Only one use case has been demonstrated and does not show the full capacity. The system was intended to allow users to watch while they analyze. Users are able to view the movie while using the system, but are unable to analyze a scene or frame and then go to that spot in the film. This could easily be done if the time stamps were a part of the processed results. This system would also be able to keep track of the amount of screen time each character had and rank it from most to least which could possibly allude to each characters importance in the film. Currently users could use the number of scenes a character is in or the interactions a character has to help with character analysis, but it would be of greater assistance if the time stamps and all scenes were made available. The

system is intended to allow users to assess small pieces of the film at a time which will enable them to recognize aspects they might normally overlook.

An important use case to be implemented is collaboration. A researcher from the film studies department shared the importance for each individual to conduct research to discuss analyses with others in the film community. This feature would enable professors and students to collaborate because without the ideas of others, research is not as productive. Users would collaborate with their peers similar to that of a discussion forum. Users could choose which topic and be given a search option where they can query by subject, reviewers, or date. Topics will be an accumulation of movie titles and users opinions. Just as it is demanded to have instant access to films, instant access to peers is also necessary without having to use other tools such as Twitter or Facebook. In providing availability of films and information the process of conducting research in this area will be more efficient.

The results from processing each film would be greater if extraction was based more upon frames or scenes rather than the characters themselves. The software could be used to create a log of each frame and scene so users would have the ability to click on any of the frames and review the analyses related to them. For example, the user will select a movie and the system will return a list of all the frames within the movie and allow the users to select a frame to insert desired details. They would be able to see which characters are involved and insert annotations for that frame. Annotations will be extracted and viewed along with the original frame. Annotations can be searched by the movie title, reviewer or date which is a part of the review use case. The information listed would include the scene number, frame number,

the time interval for the frame, information about the characters, and the analysis for that frame entered by the user.

Another feature that would be immensely useful concerning film studies would be voice recognition. If voice recognition could be used to transcribe the dialogue similar to the research done by Transana, users would have a story to go along with the picture. If the dialogue could be captured for each scene, users would be able to see characters names and determine what is happening in the scene. Instead of just knowing a character appeared in a certain scene with another character, they would know if they characters interacted with each other.

This system could be built differently to better suit the athletic, health and security departments. The athletic department could use this for scouting, evaluating a player's skills or conduct, or assessing their playbook routine. The health department could use this system in keeping up with the treatment of their patients as well as activities performed by their employees. The security department could use this application in keeping logs for user access and to identify a criminal if there is not a match between the person in the video and the images stored in the database. Lack of a match indicates an intruder.

Depending on how this system is used would determine what level of security is needed. For example, creating a homepage to give an introduction to the public the purpose and capabilities of the system, but the rest of the website would be available to only those who are logged in. For use within schools maybe creating levels of access. For instance, create an administrator, faculty and student role. The administrator would be able to view the database, folders, and user entries. Faculty would be able to view their entries, other faculty entries and

student entries. Students would only be able to view their own entries unless they are given access by a faculty member.

[illegible]

About Us. (1999-2011). Retrieved January 21, 2011, from Dartfish: http://www.dartfish.com/en/about-us/index.htm

Dartfish TeamPro. (1999-2011). Retrieved January 21, 2011, from Dartfish: http://www.dartfish.com/en/software/dartfish-teampro/index.htm

Vision. (1999-2011). Retrieved January 21, 2011, from Dartfish: http://www.dartfish.com/en/about-dartfish/vision.htm

Part 2: Mise-en-scene. (2002). Retrieved January 21, 2011, from YALE FILM STUDIES: http://classes.yale.edu/film-analysis/

YALE FILM STUDIES. (2002). Retrieved January 21, 2011, from Part 3: Cinematography : http://classes.yale.edu/film-analysis/

Face Detection. (2004-2011). Retrieved January 21, 2011, from pittpatt: http://www.pittpatt.com/face_detection/

Face Recognition. (2004-2011). Retrieved January 21, 2011, from pittpatt: http://www.pittpatt.com/face_recognition/

Face Tracking. (2004-2011). Retrieved January 21, 2011, from pittpatt: http://www.pittpatt.com/face_tracking/

Pittsburgh Pattern Recognition. (2004-2011). Retrieved January 21, 2011, from pittpatt: http://www.pittpatt.com/

Voice Recognition. (2005-2009). Retrieved January 21, 2011, from Transana: http://www.transana.org/support/VoiceRecognition.htm

Videana: A Software Tool for Scientific Film Analysis. (2007, June 18-19). Retrieved January 21, 2011, from Digital Tools in Film Studies: http://www.digital-tools-in-film-studies.com/en/programme/ewerth-et-al.html

Deep Video Indexing. (2009). Retrieved January 21, 2011, from Autonomy Virage: http://www.virage.com/rich-media/technology/understanding-video/deep-video-indexing/index.htm

EigenFace. (2010, November 22). Retrieved January 21, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Eigenface

Facial Recognition System. (2011, January 27). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Face_recognition

Final Cut Pro. (2011, January 29). Retrieved January 21, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Final_cut_pro

Bordwell, D., & Thompson, K. (2004). *Film Art: An Introduction.* Boston: McGraw-Hill.

cfloud. (1999-2011). *How to Analyze a Scene in Film*. Retrieved February 11, 2011, from eHow: http://www.ehow.com/how_2138317_analyze-scene-film.html

Ewerth, R. (2007, June 18-19). *Videana: A Software Tool for Scientific Film Analysis*. Retrieved January 21, 2011, from Digital Tools in Film Studies: http://www.digital-tools-in-film-studies.com/de/programme/ewerth-et-al.html

Fry, B., & Reas, C. (2002). *Video*. Retrieved January 21, 2011, from Processing: http://www.processing.org/reference/libraries/video/index.html

Hampapur, A. (1999, March 1). Semantic video indexing: approach and issues. New York, New York, USA. Retrieved from ACM Digital Library.

Juwei, L., Plataniotis, K. N., & Venetsanopoulos, A. N. (2003, January). *Face recognition using LDA-based algorithms.* Retrieved January 21, 2011, from IEEE.

Juwei, L., Plataniotis, K., & Venetsanopoulos, A. (2003, January). *Face recognition using kernel direct discriminant analysis algorithms.* Retrieved January 21, 2011, from IEEE: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?&arnumber=1176132&abstractAccess=no&userType=inst

Kropf, V., & Zeppelzauer, M. (2007, June 18-19). *Introducing the project "Digital Formalism: The Vienna Vertov Collection".* Retrieved January 21, 2011, from Digital Tools in Film Studies: http://www.digital-tools-in-film-studies.com/en/programme/kropf-zeppelzauer.html

Mittal, K., & Walker, G. (2006, September 5). *Merge XML and Java with XMLBeans in commerce*. Retrieved January 21, 2011, from IBM: http://www.ibm.com/developerworks/xml/library/x-vertxmlbeans/

Schild, M. (2007, June 18-19). *Text Based Film Retrieval 2006.* Retrieved January 21, 2011, from Digital Tools in Film Studies: http://www.digital-tools-in-film-studies.com/en/programme/schild.html

WCER. (2005-2009). *Transana*. Retrieved January 21, 2011, from Transana: http://www.transana.org/index.htm

Yoshitaka, A., & Ichikawa, T. (1999, January). *A Survey on Content-Based Retrieval for Multimedia Databases*. Retrieved January 21, 2011, from IEEE: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=755617&abstractAccess=no&userType=

Yoshitaka, A., Ishii, T., Hirakawa, M., & Ichikawa, T. (1997, January 23-26). *Content-based retrieval of video data by the grammar of film*. Retrieved January 21, 2011, from IEEE: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=626599&abstractAccess=no&userType=

Appendix

Use Cases

Use Case Name:	Character Identification
Scenario:	Identify known characters within the film by indicating the name as well as descriptions
Triggering Event:	User chooses character identification option, selects a movie, and then selects a character to identify
Brief Description:	The user makes a selection of which character to identify from a group of images, enters known characteristics that can be viewed or edited at a later time
Actors:	Reviewer
Related Use Cases:	
Stakeholders:	Reviewer: to verify information is adequate and for intended purposes
Pre-Conditions:	The film must exist and be processed in order for the characters to be extracted.
Post-Conditions:	New information will be stored in the database.
Flow of Events:	Interaction between actor and system
	1. User logs into the system 2. Verification of log in information is correct 3. User clicks on character identification option 4. A grid view is made available and returns movies that have already been entered into the system 5. User selects a movie from a grid view or adds a new movie to the selection 6. A new page is displayed with all of the characters that have made appearances in the selected film 7. User selects an image for a list of photos 8. User fills in known information pertaining to selected character 9. Data is inserted into the database 10. User can either view the character's image gallery, view who the current character interacted with, go on to another page, or logout 11. Logs the user out or redirects to another page
Exception Conditions:	1. If the user fails verification at the log in, an error message will display, user will not be able to access the system, and the user will be prompted to try again.

Use Case Name:	Frame Analysis
Scenario:	List descriptions and other annotations to a particular frame
Triggering Event:	User chooses movie review option, selects a movie, and then selects a frame to analyze

Brief Description:	The user makes a selection of which frame to identify from a list of frames, enters known descriptions or thoughts that can be viewed or edited at a later time
Actors:	Reviewer
Related Use Cases:	
Stakeholders:	Reviewer: to verify information is adequate and for intended purposes
Pre-Conditions:	The film must exist and be processed in order for the frames to be extracted.
Post-Conditions:	New information will be stored in the database.
Flow of Events:	Interaction between actor and system
	1. User logs into the system 2. Verification that log in information is correct 3. User clicks on review movie link 4. A grid view is made available which returns movies that have already been entered into the system 5. User selects a movie from a grid view 6. A new page displays with a drop down list containing frames from the selected film 7. User selects a frame from a drop down list 8. Data is inserted into the database 9. User fills in known information pertaining to selected frame 10. User clicks on the save button User can either exit by logging out or return to the main menu 11. Logs the user out or redirects to the home page
Exception Conditions:	1. If the user fails verification at the log in, an error message will display, user will not be able to access the system, and the user will be prompted to try again.

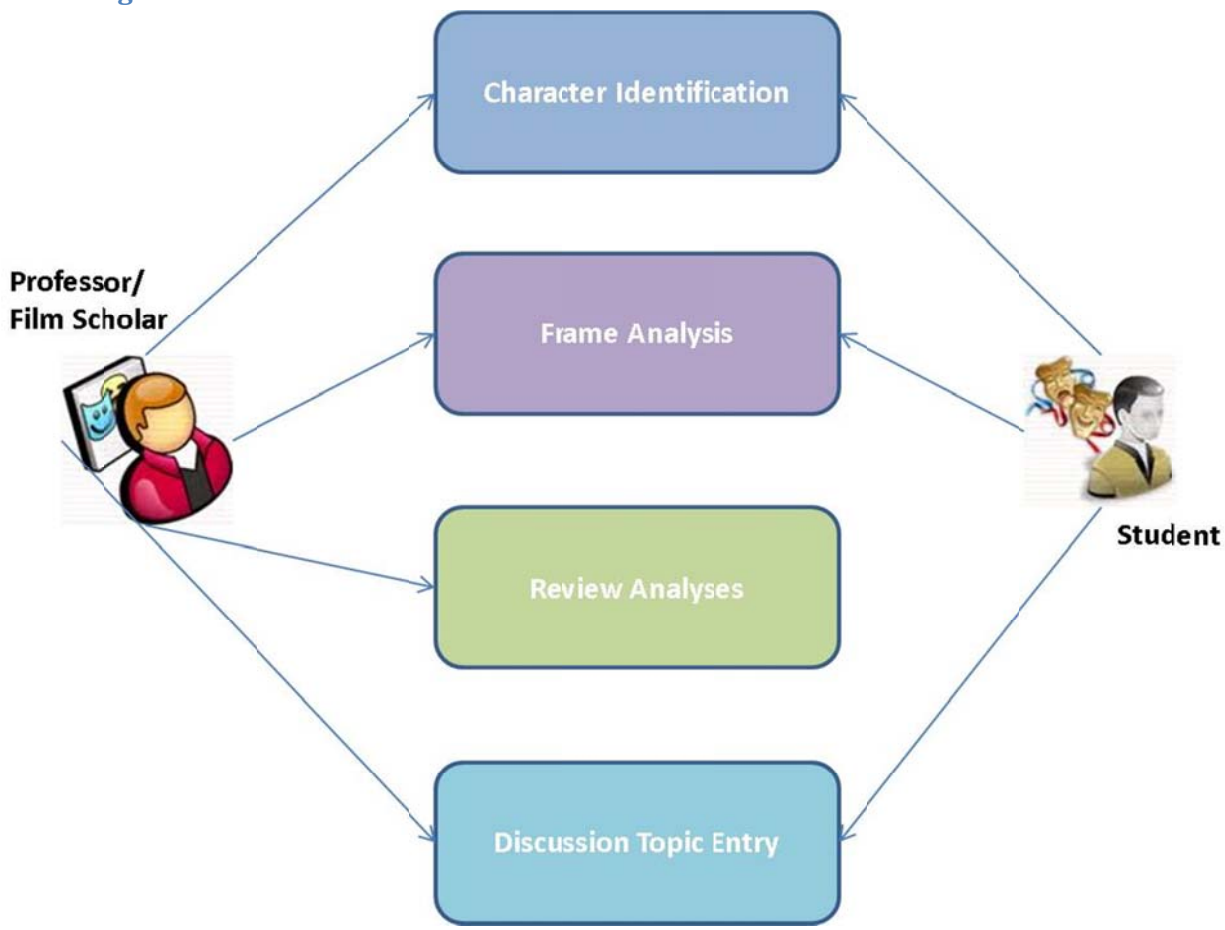
Use Case Name:	Review Analyses
Scenario:	View frames or documents related to a particular film
Triggering Event:	User chooses movie review option, selects a movie, and then selects a frame or document to review
Brief Description:	The user is able to look frame analyses or documents by other users or their own.
Actors:	Reviewer
Related Use Cases:	
Stakeholders:	Reviewer: to verify information is adequate and for intended purposes
Pre-Conditions:	Frames must already be analyzed and documents must be submitted in order for them to be reviewed later.
Post-Conditions:	Users can only make changes to their own frame analyses or documents

	submitted
Flow of Events:	Interaction between actor and system
	1. User logs into the system 2. Verification log in information is correct 3. User clicks on review movie link 4. A grid view is made available returning movies that have already been entered into the system 5. User selects a movie from a grid view 6. A new page displays with a drop down list of frames from a particular film and documents submitted 7. User selects a frame from a drop down list or a document from a grid view 8. User views this information 9. Verification user is able to make changes, redirects to the home page, logs the user out 10. User can make changes, return to main menu, or exit by logging out
Exception Conditions:	1. If the user fails verification at the log in, an error message will display, user will not be able to access the system, and the user will be prompted to try again. 2. If user fails verification to make changes, an error message will display and the user will be unable to make any changes

Use Case Name:	Discussion Topic Entry
Scenario:	Enter a topic for discussion
Triggering Event:	User chooses discuss topics option, selects a movie, and then enters thoughts/views
Brief Description:	The user enters thoughts and ideas that can be viewed and built upon at a later time
Actors:	Reviewer
Related Use Cases:	
Stakeholders:	Reviewer: to verify information is adequate and for intended purposes
Pre-Conditions:	
Post-Conditions:	New information will be stored in the database and user can only make changes to their own entries.
Flow of Events:	Interaction between actor and system
	1. User logs into the system 2. Verification log in information is correct 3. User clicks on discuss topics link 4. A grid view is made available returning movies that have already been entered into the system

	5. User selects a movie from a grid view 6. A new page is displayed with a grid view of characters that have made appearances in the selected film 7. User can enter a new topic discussion or view existing discussions 8. User clicks the add button if they have entered a new topic 9. Data is inserted into the database 10. User can return to view other entries, edit entries, return to main menu, or exit by logging out Logs the user out or redirects to the home page
Exception Conditions:	1. If the user fails verification at the log in, an error message will display, user will not be able to access the system, and the user will be prompted to try again. 2. If user fails verification to make changes, an error message will display and the user will be unable to make any changes

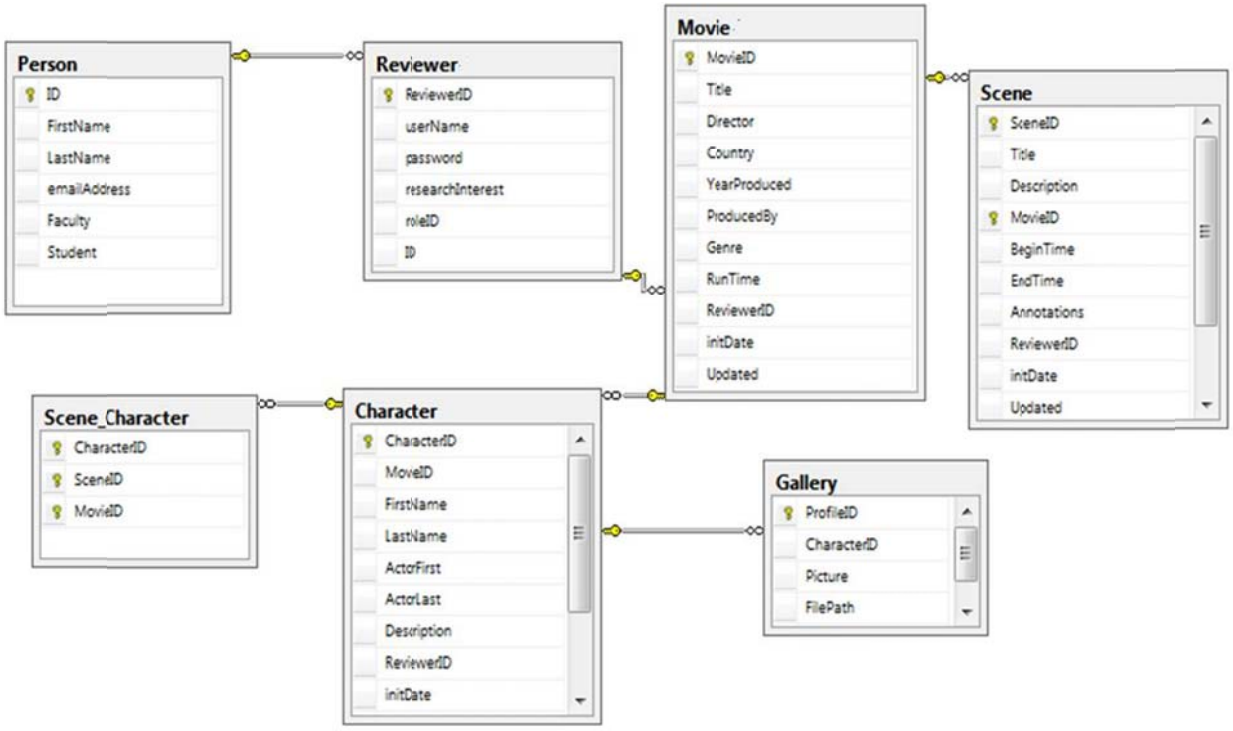
Actor Diagram



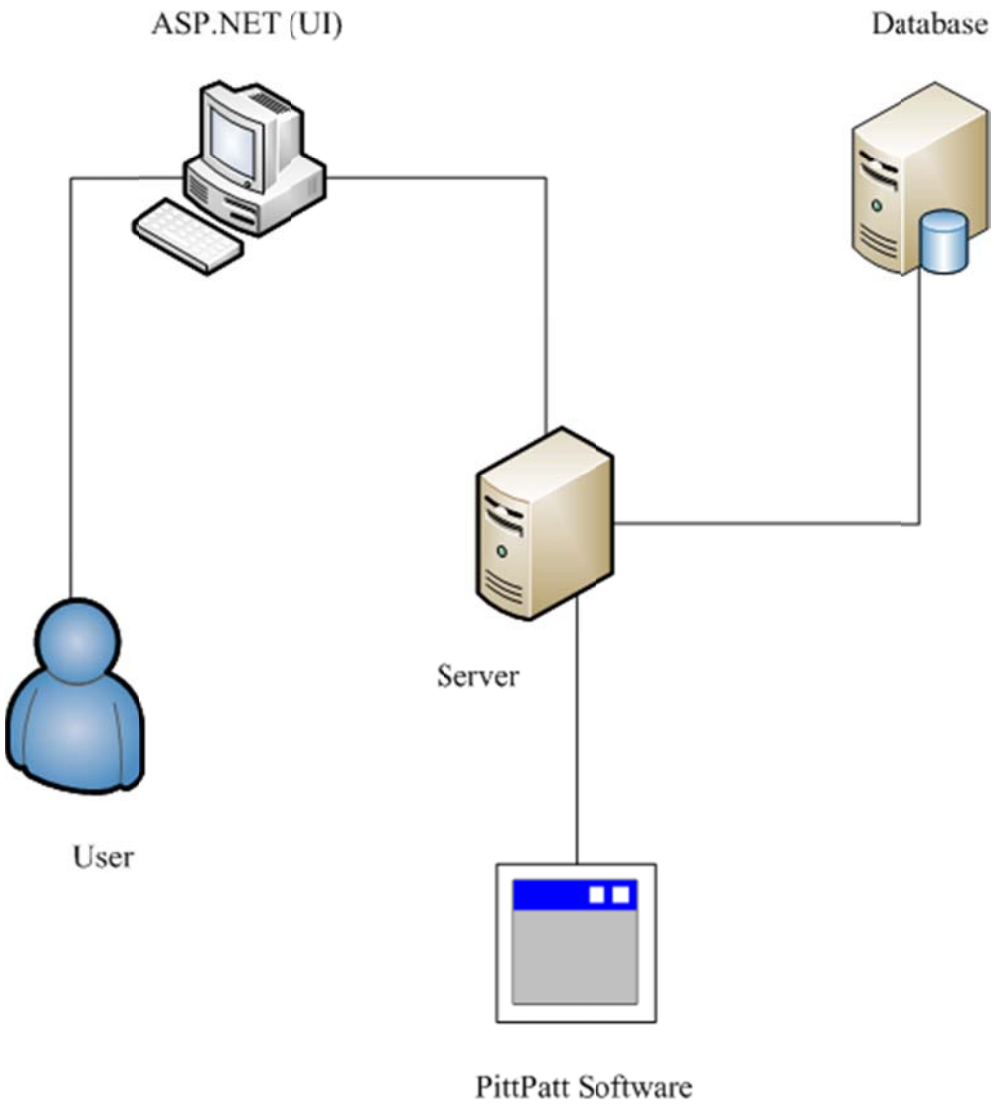
Resources

Dr. Judith Gebauer	Capstone Chair – Feasibility, Business & Functional requirements
Dr. Karl Ricanek	Capstone Committee – User Interface & Programming
Dr. Devon Simmonds	Capstone Committee – Diagrams & Software Process
Dominique Jackson	Design & Implementation
Benjamin Barbour	Face Aging Group – Windows Service
PittPatt 4.1	Face Recognition Software
Visual Studio 2010	Used for Deployment
SQL Server 2005	Store and return data

Database Diagram



Multi-tier Web Application



Structured Query Language Procedures

```
CREATE PROCEDURE dbo.CharacterDelete
        (@CharacterID int)
AS
        /* SET NOCOUNT ON */
DELETE FROM Character
        WHERE CharacterID = @CharacterID
        RETURN
```

```
CREATE PROCEDURE dbo.CharacterSelect
	(@MoveID int)
AS
	/* SET NOCOUNT ON */
SELECT	CharacterID, MovieID, FirstName, LastName, ActorFirst, ActorLast, Description,
ReviewerID, initDate, Updated, FilePath, binImage, FileName
FROM	Character
WHERE	(MovideID = @MovieID)
	RETURN

CREATE PROCEDURE dbo.CharacterUpdate
	(@CharacterID int, @FirstName varchar(50), @LastName varchar(50), @ActorFirst
varchar(50), @ActorLast varchar(50), @Description varchar(50), @ReviewerID int, @initDate
datetime, @Updated datetime, @FilePath nvarchar(255), @binImage image, @FileName
varchar(255))
AS
	/* SET NOCOUNT ON */
UPDATE	Character
SET	FirstName = @FirstName, LastName = @LastName, ActorFirst = @ActorFirst,
ActorLast = @ActorLast, Description = @Description, ReviewerID = @ReviewerID, initDate =
@initDate, Updated = @Updated, FilePath = @FilePath, binImage = @binImage, FileName =
@FileName
WHERE	(CharacterID = @CharacterID)
	RETURN

CREATE PROCEDURE dbo.CharInsert
	(@MovieID int, @FirstName varchar(50), @FilePath nvarchar(255), @binImage image,
	@FileName varchar(255))
AS
	/* SET NOCOUNT ON */
INSERT INTO Character
		(MovieID, FirstName, FilePath, binImage, FileName)
VALUES	(@MovieID, @FirstName, @FilePath, @binImage, @FileName)
	SELECT @@IDENTITY

	RETURN @@IDENTITY


CREATE PROCEDURE dbo.GalleryInsert
	(@CharacterID int, @Picture image, @FilePath nvarchar(255), @FileName
	nvarchar(255))
AS
INSERT INTO Gallery
```

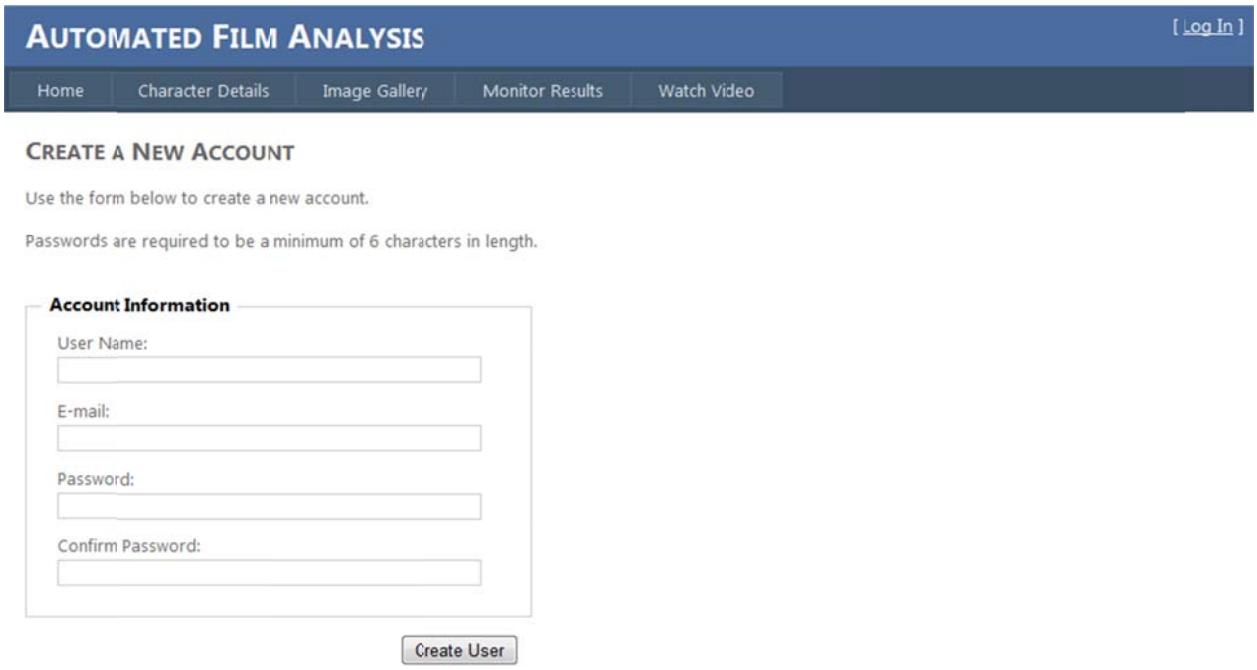
```
                    (CharacterID, Picture, FilePath, FileName)
VALUES      (@CharacterID, @Picture, @FilePath, @FileName)
        RETURN

CREATE PROCEDURE dbo.GallerySelect
        (@CharacterID int)
AS
        /* SET NOCOUNT ON */
SELECT      ProfileID, CharacterID, Picture, FilePath, binImage, FileName
FROM         Gallery
WHERE      (CharacterID = @CharacterID)
        RETURN


CREATE PROCEDURE dbo.InsertSceneChar
        (@CharacterID int, @SceneID int, @MovieID int)
AS
        /* SET NOCOUNT ON */
INSERT INTO Scene_Character
                (CharacterID, SceneID, MovieID)
VALUES      (@CharacterID, @SceneID, @MovieID)
                RETURN

CREATE PROCEDURE dbo.InteractQuery
        (@CharacterID int, @startFrame int, @endFrame int, @MovieID int)
AS
        /* SET NOCOUNT ON */
SELECT      Scene.SceneID, Scene_Character.CharacterID, Scene.MovieID, Scene.startFrame,
Scene.endFrame
FROM         Scene INNER JOIN
                Scene_Character ON Scene.SceneID = Scene_Character.SceneID
WHERE      (Scene.startFrame BETWEEN @startFrame AND @endFrame) AND
(Scene_Character.CharacterID <> @CharacterID) AND (Scene.MovieID = @MovieID) OR
                (Scene.endFrame BETWEEN @startFrame AND @endFrame) AND
(Scene_Character.CharacterID <> @CharacterID) AND (Scene.MovieID = @MovieID) OR
                (Scene.startFrame < @startFrame) AND (Scene_Character.CharacterID <>
@CharacterID) AND (Scene.endFrame > @endFrame) AND (Scene.MovieID = @MovieID)
        RETURN

CREATE PROCEDURE dbo.MovieDelete
        (@MovieID int)
AS
        DELETE FROM Movie
                WHERE MovieID = @MovieID
```

```
	RETURN

CREATE PROCEDURE dbo.MovieInsert
	(@Title varchar(50), @Director varchar(50), @Country varchar(50), @YearProduced
	datetime, @ProducedBy varchar(50), @Genre varchar(50), RunTime datetime,
	@ReviewerID int)
AS
	INSERT INTO Movie
			(Title, Director, Country, YearProduced, ProducedBy, Genre, RunTime,
ReviewerID, initDate)
	VALUES
(@Title,@Director,@Country,@YearProduced,@ProducedBy,@Genre,@RunTime,@ReviewerID
,GetDate())
	RETURN

CREATE PROCEDURE dbo.MovieSelect
	(@MovieID int)
AS
	/* SET NOCOUNT ON */
	SELECT	MovieID, Title, Director, Country, YearProduced, ProducedBy, Genre,
RunTime, ReviewerID, initDate, Updated
	FROM	Movie
	RETURN

CREATE PROCEDURE dbo.MovieUpdate
	(@MovieID int, @Title varchar(50), @Director varchar(50), @Country varchar(50),
@YearProduced datetime, @ProducedBy varchar(50), @Genre varchar(50), @RunTime
datetime, @ReviewerID int, @initDate datetime, @Updated datetime)
AS
	UPDATE	Movie
	SET	Title =@Title, Director =@Director, Country =@Country, YearProduced
=@YearProduced, ProducedBy =@ProducedBy, Genre =@Genre, RunTime =@RunTime,
ReviewerID =@ReviewerID, initDate =@initDate, Updated =@Updated
	WHERE		(MovieID = @MovieID)
	RETURN


CREATE PROCEDURE dbo.PersonInsert
	(@FirstName varchar(50), @LastName varchar(50), @emailAddress varchar(50),
@Faculty bit, @Student bit)
AS
	INSERT INTO Person
			(FirstName, LastName, emailAddress, Faculty, Student)
	VALUES	(@FirstName,@LastName,@emailAddress,@Faculty,@Student)
```

```
        SELECT @@IDENTITY

        RETURN @@IDENTITY

CREATE PROCEDURE dbo.SceneInsert
        (@MovieID int, @startFrame int, @endFrame int)
AS
        /* SET NOCOUNT ON */
        INSERT INTO Scene
                (MovieID, startFrame, endFrame)
VALUES      (@MovieID, @startFrame, @endFrame)
        SELECT @@IDENTITY

        RETURN @@IDENTITY
CREATE PROCEDURE dbo.SceneQuerySelect
        (@CharacterID int)
AS
        SELECT Scene.SceneID, Scene.MovieID, Scene.BeginTime, Scene.EndTime,
Scene.startFrame, Scene.endFrame, Scene_Character.CharacterID FROM Scene INNER JOIN
Scene_Character ON Scene.SceneID = Scene_Character.SceneID
        WHERE       (CharacterID = @CharacterID)
        RETURN

CREATE PROCEDURE dbo.SceneQuery
        (@SceneID int, @CharacterID int)
AS
        SELECT Scene.SceneID, Scene.MovieID, Scene.BeginTime, Scene.EndTime,
Scene.startFrame, Scene.endFrame, Scene_Character.CharacterID FROM Scene INNER JOIN
Scene_Character ON Scene.SceneID = Scene_Character.SceneID
        WHERE       (Scene.SceneID = @SceneID AND CharacterID = @CharacterID)
        RETURN
```

User Guide

Register



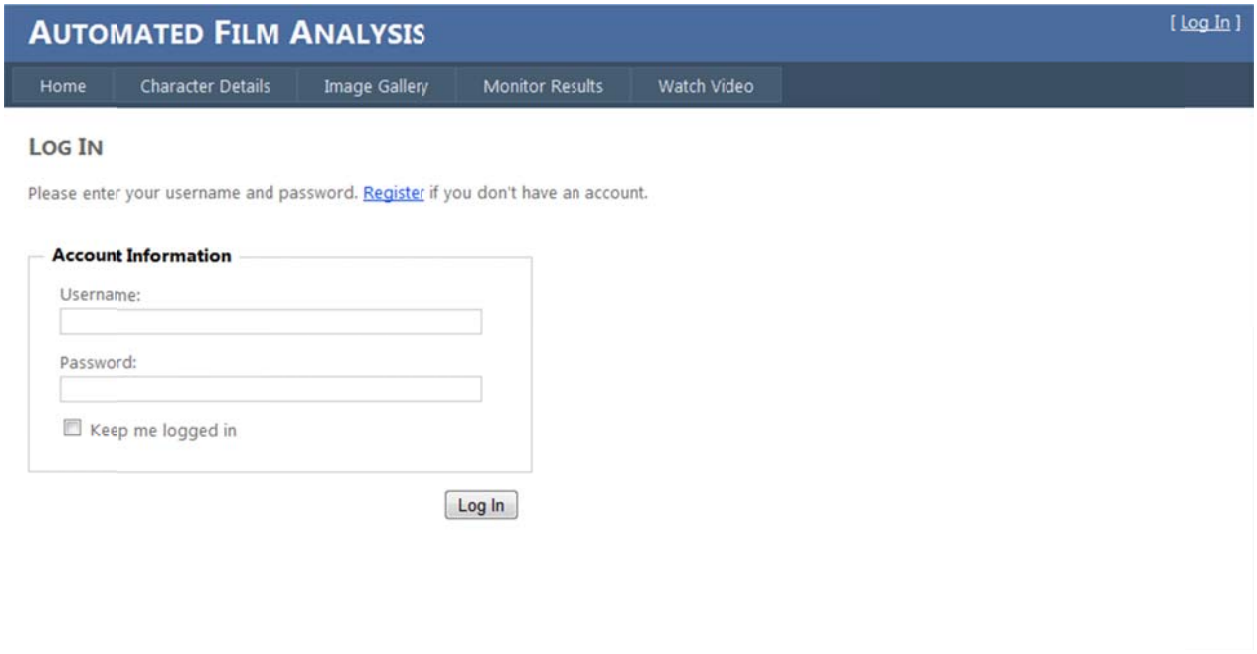
For first time use, generate an account by entering an e-mail address and create a user name and password. Once the information has been entered click create user to proceed to a new page where personal information is inquired.

User Information

After registering enter first and last name and select an occupation. Once complete, select submit to advance to the main page of the system.

User Information

Login page



After registering, future entry into the system requires users to log in. Provide username and password and click the login button to advance to the main page of the system.

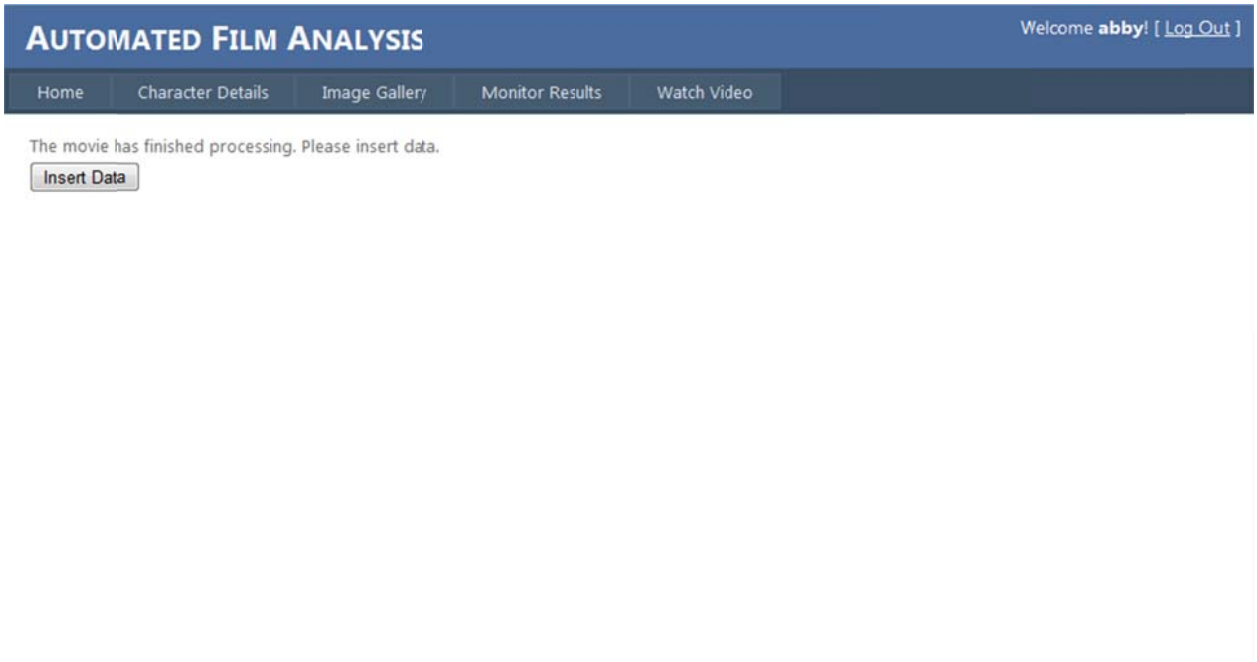
Selection of movie and action to perform

This is the main page of the system. To view characters and character details select character identification. To watch a video that has already been uploaded select watch video. To verify whether a video has been processed select monitor results. If a video has not been processed characters will not be displayed. To search for a video, enter an attribute of the video, select its correlation from the dropdown list and click the search button. Once a video is selected the system will transition to a new page. In order to upload a new video, click upload movie. A new page is displayed to collect information about the video being uploaded.

Upload Movie

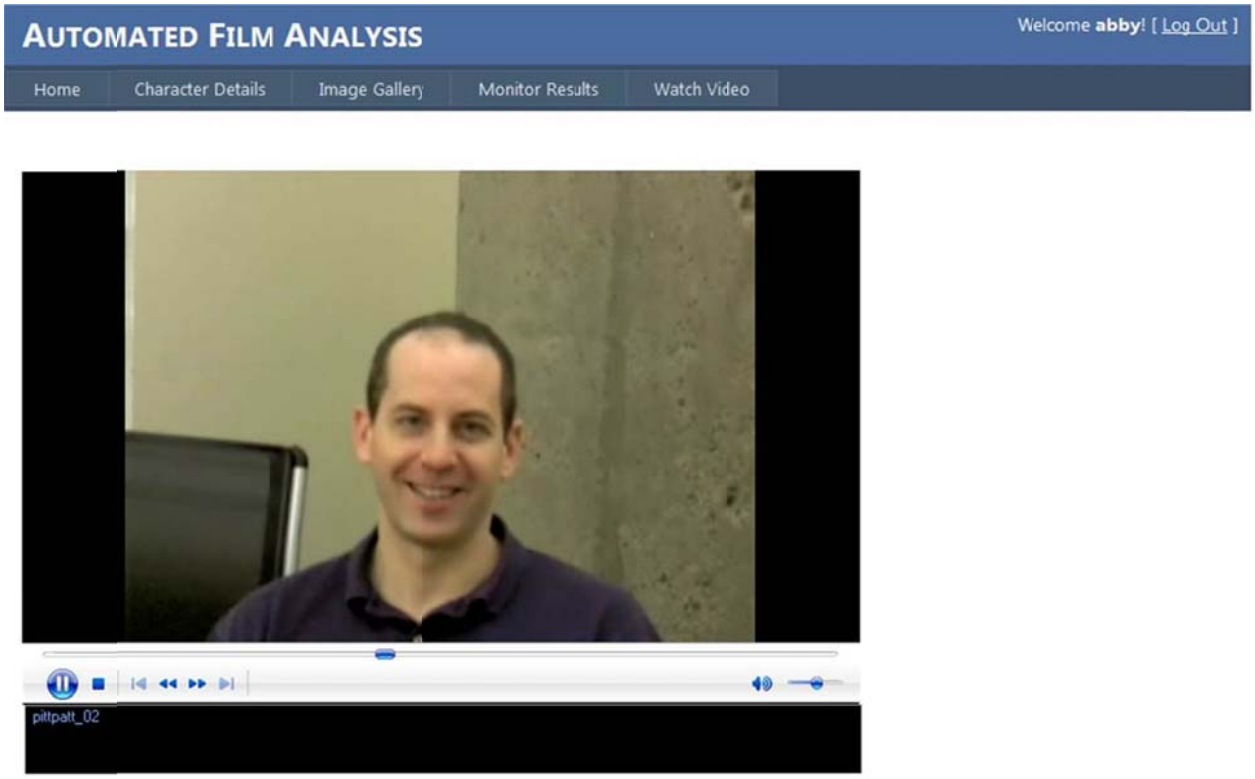
Enter the title, year, director, and country. Click the browse button to locate where the desired movie is stored. Once the information has been entered select the submit button to return to the main page.

Monitor results



If the video selected has been processed the insert data button will not be enabled. If the video selected has not been processed click the insert data button to insert video results into the database.

Watch video

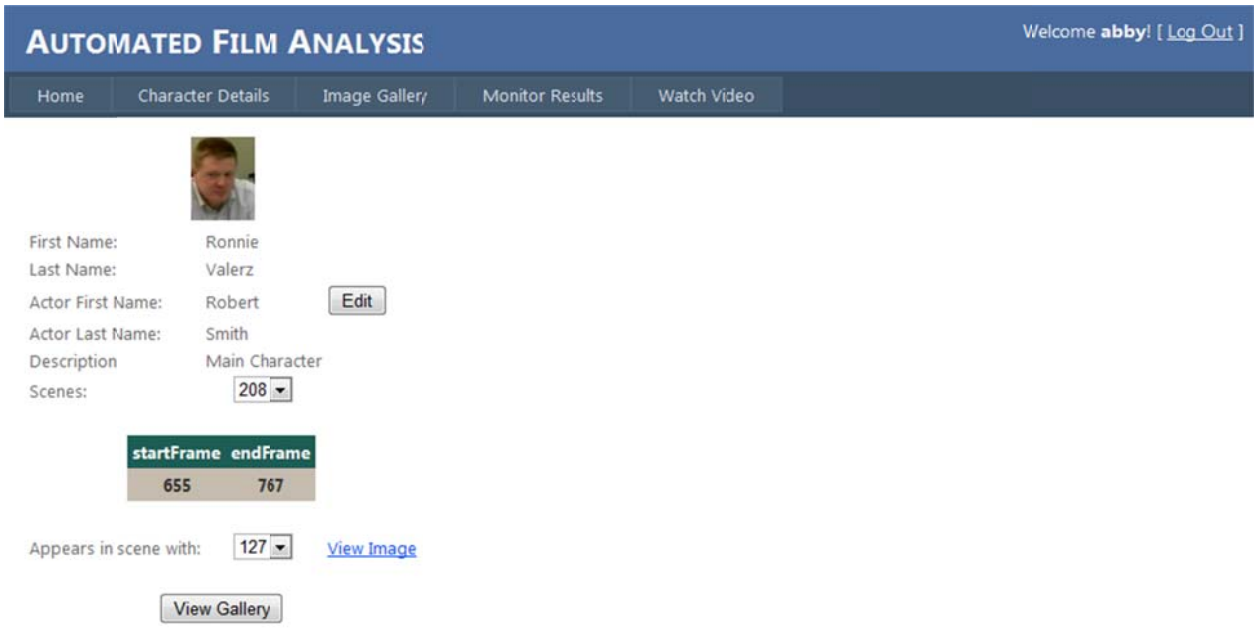


The video can be played, paused, resumed, stopped, fast forwarded, and rewound like when watching a normal video. The volume can also be adjusted, muted or unmuted.

Characters in video

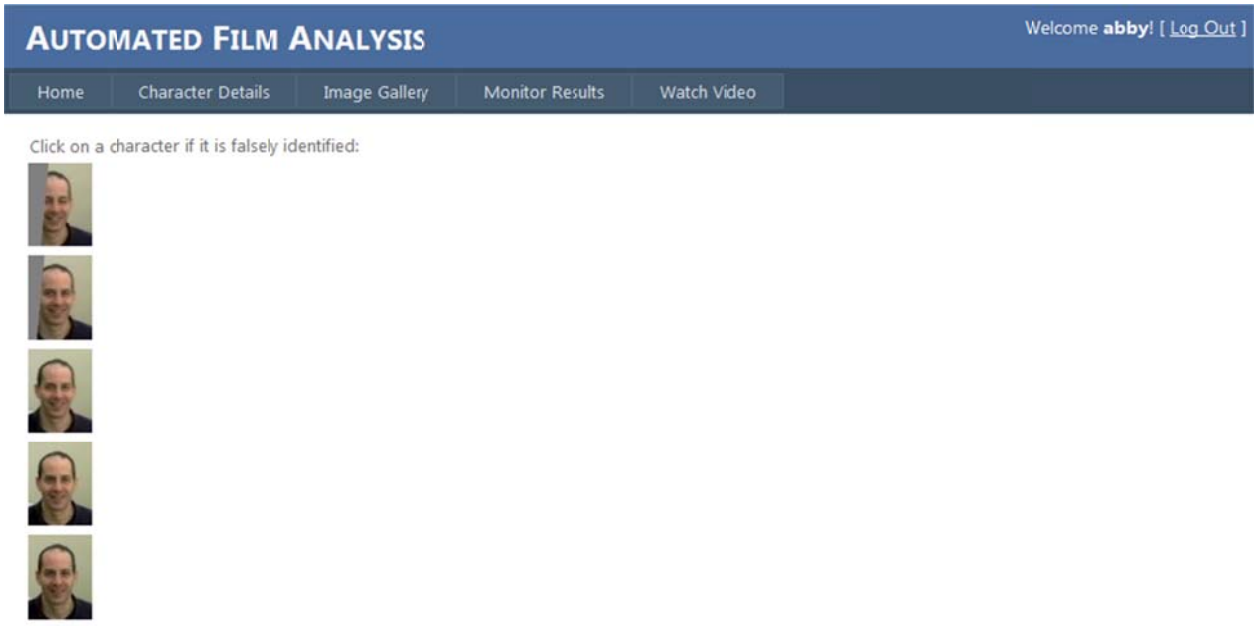
This page displays all the characters in the selected video. Click on a character to advance to a new page which displays character details.

Character Details



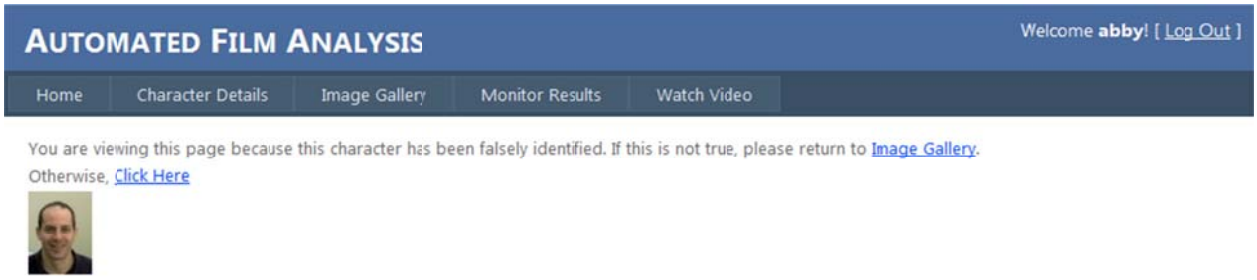
At the top is a picture of the selected character. The first name, last name, actor first name, actor last name, and description is either empty or generic if no information has been entered by a user. Click the edit button to update or revise. The first dropdown lists all of the scenes the character is in. Select different scenes to see correlated start frames and end frames. The second dropdown lists the characters the selected character appeared in a scene with and can be viewed by clicking the view image link. The view gallery button advances to a page which displays all the pictures of the chosen character in the selected video.

Correct Character Identification



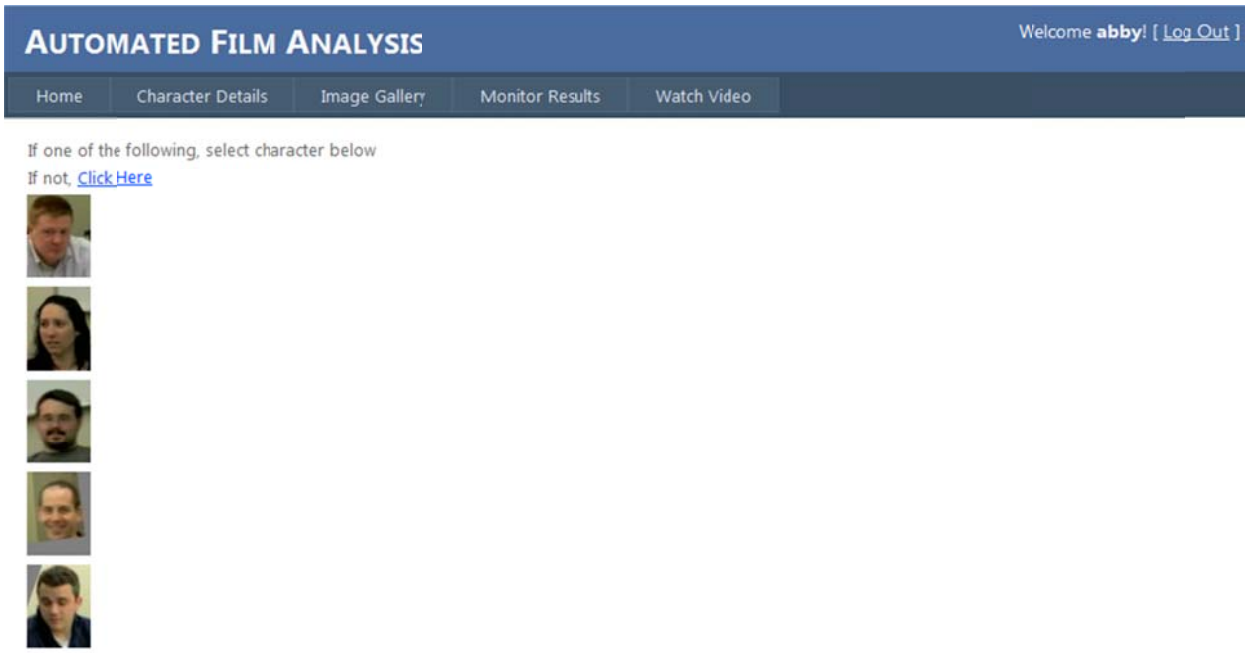
This is the image gallery which displays all the images found in the selected video of the chosen character. If a character displayed does not belong here, click on it to correct identification.

Confirm character is falsely identified



This page allows for verification of a character that has been incorrectly identified. A picture of the falsely identified character is visible. If this is correct, choose the click here link. If this is not correct click the image gallery link to return to the previous page.

False character is another character in movie or add new



If the falsely identified character is one of the characters already identified in the video, select the character. Otherwise, select the click here link which will create a new character and return to the page that displays all of the characters in the selected video.

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;

public partial class Character : System.Web.UI.Page
{
   public string file, movie;
   protected void Page_Load(object sender, EventArgs e)
   {
      //If the id is null return to previous page
      if (Page.ClientQueryString != "")
      {
      }
      else
      {
         Response.Redirect("Cphoto.aspx");
      }
      //Retrieve start frame and end frame
      DataView dv = (DataView)SqlDataSource4.Select(DataSourceSelectArguments.Empty);

      foreach (DataRowView drv in dv)
      {
         txtStart.Text = drv["startFrame"].ToString();
         txtEnd.Text = drv["endFrame"].ToString();
      }

      gvFrame.SelectedIndex = 0;
      imgCharacter.ImageUrl = file;
      btnUpdate.Visible = false;
      txtFirstName.Visible = false;
      txtLastName.Visible = false;
      txtActFirst.Visible = false;
      txtActLast.Visible = false;
      txtDesc.Visible = false;
```

```
    lnkImage.Text = "View Image";

    //Use the character id to retrieve character details to be displayed
    SqlConnection con = new SqlConnection();
    con.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
    SqlCommand cmd = new SqlCommand("Select FirstName, LastName, ActorFirst, ActorLast,
            Description, FileName, MovieID FROM Character WHERE CharacterID=" +
            Request.QueryString["CharacterID"], con);
    cmd.CommandType = System.Data.CommandType.Text;
    con.Open();
    SqlDataReader myReader = null;
    myReader = cmd.ExecuteReader();

    while (myReader.Read())
    {
        lblFirstName.Text = myReader[0].ToString();
        lblLastName.Text = myReader[1].ToString();
        lblActFirst.Text = myReader[2].ToString();
        lblActLast.Text = myReader[3].ToString();
        lblDesc.Text = myReader[4].ToString();
        file = myReader[5].ToString();
        movie = myReader[6].ToString();
    }

    con.Close();
}
protected void lnkImage_Click(object sender, EventArgs e)
{
    string getCharacterID = ddlTest.SelectedItem.Value;
    Response.Redirect("Character.aspx?CharacterID=" + getCharacterID);

}
protected void btnVG_Click(object sender, EventArgs e)
{
    string url;
    url = "ImageGallery.aspx?CharacterID=" + Request.QueryString["CharacterID"];
    Response.Redirect(url);
}
protected void btnUpdate_Click(object sender, EventArgs e)
{
    //Update character details inserted by user
    SqlConnection updCon = new SqlConnection();
    updCon.ConnectionString = ConfigurationManager.ConnectionStrings
```

```
            ["FilmAnalysisDBConnectionString"].ConnectionString;
        SqlCommand updCmd = updCon.CreateCommand();
        updCmd.CommandText = "UPDATE Character SET FirstName = @FirstName,
            LastName = @LastName, ActorFirst = @ActorFirst, ActorLast = @ActorLast,
            Description = @Description  WHERE CharacterID = @CharacterID";
        updCmd.Parameters.Add("@CharacterID", SqlDbType.Int);
        updCmd.Parameters["@CharacterID"].Value = Request.QueryString["CharacterID"];
        updCmd.Parameters.AddWithValue("@FirstName", txtFirstName.Text);
        updCmd.Parameters.AddWithValue("@LastName", txtLastName.Text);
        updCmd.Parameters.AddWithValue("@ActorFirst", txtActFirst.Text);
        updCmd.Parameters.AddWithValue("@ActorLast", txtActLast.Text);
        updCmd.Parameters.AddWithValue("@Description", txtDesc.Text);
        updCon.Open();
        updCmd.ExecuteNonQuery();
        updCon.Close();

        string url;
        url = "Character.aspx?CharacterID=" + Request.QueryString["CharacterID"];
        Response.Redirect(url);
    }
    protected void btnEdit_Click(object sender, EventArgs e)
    {
        btnEdit.Visible = false;
        btnUpdate.Visible = true;

        txtFirstName.Visible = true;
        txtLastName.Visible = true;
        txtActFirst.Visible = true;
        txtActLast.Visible = true;
        txtDesc.Visible = true;

        lblFirstName.Visible = false;
        lblLastName.Visible = false;
        lblActFirst.Visible = false;
        lblActLast.Visible = false;
        lblDesc.Visible = false;
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System.IO;
using System.Web.Security;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblUserName.Text = GetCurrentUserWindowsLogin();
        if (Session["Registration"] == "Visited")
        {
            Response.Redirect("Review.aspx");
        }
    }
    public string GetCurrentUserWindowsLogin()
    {
        string windowsLogin = Page.User.Identity.Name;
        return windowsLogin;
    }
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        MembershipUser member = Membership.GetUser(User.Identity.Name);
        string email = member.Email;

        bool isOk = false;
        string occupation, myID;
        int fac = -1, stu = -1;
        string windowsLogin = Page.User.Identity.Name;
        occupation = ddlOcupation.SelectedValue;
        if (occupation == "0")
        {
            ddlOcupation.Focus();
        }
        if (occupation == "1")
        {
            fac = 1;
            stu = 0;
        }
        if (occupation == "2")
        {
```

```
            stu = 1;
            fac = 0;
        }

        //Insert new user personal data
        SqlConnection con = new SqlConnection();
        con.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
        SqlCommand cmd = new SqlCommand("PersonInsert", con);
        cmd.CommandType = System.Data.CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@FirstName", txtFirstName.Text);
        cmd.Parameters.AddWithValue("@LastName", txtLastName.Text);
        cmd.Parameters.AddWithValue("@emailAddress", email);
        cmd.Parameters.AddWithValue("@Faculty", fac);
        cmd.Parameters.AddWithValue("@Student", stu);
        cmd.Parameters.Add("@Return_Value", SqlDbType.Int, 4);
        cmd.Parameters["@Return_Value"].Direction = ParameterDirection.ReturnValue;
        con.Open();
        cmd.ExecuteNonQuery();
        myID = cmd.Parameters["@Return_Value"].Value.ToString();
        isOk = true;
        uiMessage.Text = "some message";
        con.Close();

        string url;
        url = "Review.aspx?id=" + myID;
        Response.Redirect(url);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.IO;

public partial class FalseIdentify : System.Web.UI.Page
{
    public string id, mFile;
```

```
public int charID = -1;
protected void Page_Load(object sender, EventArgs e)
{
   Page.DataBind();
   lnkFalse.Text = "Click Here";
   mFile = (string)(Session["PathDir "]);

}
protected void lnkFalse_Click(object sender, EventArgs e)
{
   string chId = "", path = "", pth="", name="";

   //Get information of character falsely identified
   SqlConnection fgtnCon = new SqlConnection();
   fgtnCon.ConnectionString = ConfigurationManager.ConnectionStrings
         ["FilmAnalysisDBConnectionString"].ConnectionString;
   SqlCommand fgtnCmd = new SqlCommand("Select ProfileID, CharacterID, FilePath FROM
         Gallery WHERE FileName LIKE '" + mFile + "'", fgtnCon);
   fgtnCmd.CommandType = System.Data.CommandType.Text;
   fgtnCon.Open();
   SqlDataReader fgtnRead = null;
   fgtnRead = fgtnCmd.ExecuteReader();
   while (fgtnRead.Read())
   {
      id = fgtnRead[0].ToString();
      chId = fgtnRead[1].ToString();
      path = fgtnRead[2].ToString();
   }

   fgtnCon.Close();

   //See if falsely identified character is the main profile pic
   SqlConnection cCon = new SqlConnection();
   cCon.ConnectionString = ConfigurationManager.ConnectionStrings
         ["FilmAnalysisDBConnectionString"].ConnectionString;
   SqlCommand cCmd = new SqlCommand("Select CharacterID FROM Character WHERE
         FileName LIKE '" + mFile + "'", cCon);
   cCmd.CommandType = System.Data.CommandType.Text;
   try
   {
      if (cCon.State == System.Data.ConnectionState.Closed)
      {
         cCon.Open();
      }
```

```
    charID = (int)cCmd.ExecuteScalar();
}
catch (Exception ex)
{
    this.lblMessage.Text = "err on inserting stored procedure " + ex;
}
finally
{
    cCon.Close();
}

//Create new character
string ncID, past = "Past CharacterID = " + chId;
byte[] imgData = ReadFile(path);
SqlConnection udCon = new SqlConnection();
udCon.ConnectionString = ConfigurationManager.ConnectionStrings
        ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand udCmd = new SqlCommand("CharInsert", udCon);
udCmd.CommandType = System.Data.CommandType.StoredProcedure;
udCmd.Parameters.AddWithValue("@MovieID", Request.QueryString["MovieID"]);
udCmd.Parameters.AddWithValue("@FirstName", past);
udCmd.Parameters.AddWithValue("@FilePath", path);
udCmd.Parameters.AddWithValue("@binImage", (object)imgData);
udCmd.Parameters.AddWithValue("@FileName", mFile);
udCmd.Parameters.Add("@Return_Value", SqlDbType.Int, 4);
udCmd.Parameters["@Return_Value"].Direction = ParameterDirection.ReturnValue;
udCon.Open();
udCmd.ExecuteNonQuery();
ncID = udCmd.Parameters["@Return_Value"].Value.ToString();
udCon.Close();

//Retrieve scenes of falsely identified character appeared in
SqlConnection soCon = new SqlConnection();
soCon.ConnectionString = ConfigurationManager.ConnectionStrings
        ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand soCmd = new SqlCommand("Select SceneID FROM Scene_Character
        WHERE CharacterID = '" + chId + "'", soCon);
soCmd.CommandType = System.Data.CommandType.Text;
soCon.Open();
SqlDataReader soRead = null;
soRead = soCmd.ExecuteReader();
while (soRead.Read())
{
    //Insert scenes for new character
```

```
    SqlConnection scCon = new SqlConnection();
    scCon.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
    SqlCommand scCmd = new SqlCommand("InsertSceneChar", scCon);
    scCmd.CommandType = System.Data.CommandType.StoredProcedure;
    scCmd.Parameters.AddWithValue("@CharacterID", ncID);
    scCmd.Parameters.AddWithValue("@SceneID", soRead[0].ToString());
    scCmd.Parameters.AddWithValue("@MovieID", Request.QueryString["MovieID"]);
    scCon.Open();
    scCmd.ExecuteNonQuery();
    scCon.Close();
}
soCon.Close();

//Update gallery
SqlConnection galCon = new SqlConnection();
galCon.ConnectionString = ConfigurationManager.ConnectionStrings
        ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand galCmd = galCon.CreateCommand();
galCmd.CommandText = "UPDATE Gallery SET CharacterID = @CharacterID
        WHERE FileName = @FileName";
galCmd.Parameters.Add("@FileName", SqlDbType.NVarChar, 255);
galCmd.Parameters["@FileName"].Value = mFile;
galCmd.Parameters.AddWithValue("@CharacterID", ncID);
galCon.Open();
galCmd.ExecuteNonQuery();
galCon.Close();

if (charID == -1 || charID == 0)
{
}
else
{
    //If falsely identified character is the main picture it must be replaced
    SqlConnection sCon = new SqlConnection();
    sCon.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
    SqlCommand sCmd = new SqlCommand("Select FilePath, FileName FROM Gallery
          WHERE CharacterID = '" + chId + "'", sCon);
    sCmd.CommandType = System.Data.CommandType.Text;
    sCon.Open();
    SqlDataReader sRead = null;
    sRead = sCmd.ExecuteReader();
    while (sRead.Read())
```

```
        {
            pth = sRead[0].ToString();
            name = sRead[1].ToString();
        }
        sCon.Close();

        byte[] imageData = ReadFile(pth);

        //Update character with new image from gallery
        SqlConnection chCon = new SqlConnection();
        chCon.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
        SqlCommand chCmd = chCon.CreateCommand();
        chCmd.CommandText = "UPDATE Character SET FilePath = @FilePath,
              FileName = @FileName WHERE CharacterID = @CharacterID";
        chCmd.Parameters.Add("@CharacterID", SqlDbType.Int);
        chCmd.Parameters["@CharacterID"].Value = chId;
        chCmd.Parameters.AddWithValue("@FilePath", pth);
        chCmd.Parameters.AddWithValue("@binImage", (object)imageData);
        chCmd.Parameters.AddWithValue("@FileName", name);
        chCon.Open();
        chCmd.ExecuteNonQuery();
        chCon.Close();
    }

    Response.Redirect("CPhoto.aspx?MovieID=" + Request.QueryString["MovieID"]);

  }

  byte[] ReadFile(string sPath)
  {
    //Initialize byte array with a null value initially.
    byte[] data = null;
    //Use FileInfo object to get file size.
    FileInfo fInfo = new FileInfo(sPath);
    long numBytes = fInfo.Length;
    //Open FileStream to read file
    FileStream fStream = new FileStream(sPath, FileMode.Open, FileAccess.Read);
    //Use BinaryReader to read file stream into byte array.
    BinaryReader br = new BinaryReader(fStream);
    data = br.ReadBytes((int)numBytes);
    return data;
  }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Xml;
using System.Text;

public partial class Monitor : System.Web.UI.Page
{
   public int cId;
   public string getCharacterID, getSceneID, mTitle;
   protected void Page_Load(object sender, EventArgs e)
   {
      btnInsert.Enabled = false;
      string dir;

      SqlConnection cCon = new SqlConnection();
      cCon.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
      SqlCommand cCmd = new SqlCommand("Select CharacterID FROM Character
            WHERE MovieID=" + Request.QueryString["id"], cCon);
      cCmd.CommandType = System.Data.CommandType.Text;
      try
      {
         if (cCon.State == System.Data.ConnectionState.Closed)
         {
            cCon.Open();
         }
         cId = (int)cCmd.ExecuteScalar();
      }
      catch (Exception ex)
      {
         this.uiMessage.Text = "This movie is still processing.";
      }
      finally
      {
```

```
            cCon.Close();
        }
        lblStatus.Text = uiMessage.Text;

        if (cId != null && cId != 0)
        {
            lblStatus.Text = "This movie has already been processed";
            btnInsert.Enabled = false;
        }
        else
        {
            SqlConnection con = new SqlConnection();
            con.ConnectionString = ConfigurationManager.ConnectionStrings
                    ["FilmAnalysisDBConnectionString"].ConnectionString;
            SqlCommand cmd = new SqlCommand("Select Title FROM Movie
                    WHERE MovieID=" + Request.QueryString["id"], con);
            cmd.CommandType = System.Data.CommandType.Text;
            con.Open();
            mTitle = (string)cmd.ExecuteScalar();
            con.Close();

            dir = "C:\\processed\\" + mTitle + ".mp4\\thumbnails\\subject_00";
            if (Directory.Exists(dir))
            {
                lblStatus.Text = "The movie has finished processing. Please insert data.";
                btnInsert.Enabled = true;
            }
            else
            {
                lblStatus.Text = "This movie is still processing.";
                btnInsert.Enabled = false;
            }
        }

    }
    protected void btnInsert_Click(object sender, EventArgs e)
    {
        int mId;
        string title = mTitle + ".mp4.xml", characterName, nomen, filepath,
                file = "", fileName, destFile, targetPath = "", xmlPath = "C:\\processed\\",
                mainDir = "C:\\Users\\Domino\\Documents\\Visual Studio 2010\\Web
                Sites\\FilmAnalysis\\Images";
        xmlPath += mTitle + ".mp4\\" + mTitle + ".mp4.xml";
```

```
//Create directory for new subfolder under the current active folder
string newPath = System.IO.Path.Combine(mainDir, mTitle);

// Create the subfolder
System.IO.Directory.CreateDirectory(newPath);

SqlConnection con = new SqlConnection();
con.ConnectionString = ConfigurationManager.ConnectionStrings
        ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand cmd = new SqlCommand("Select MovieID FROM Movie
        WHERE Title LIKE '" + mTitle + "'", con);
cmd.CommandType = System.Data.CommandType.Text;
con.Open();
mId = (int)cmd.ExecuteScalar();
con.Close();

string name = " ", previous = " ";
string[] store, subject, start, end, scenes;
store = new string[30];
scenes = new string[30];
subject = new string[30];
start = new string[30];
end = new string[30];
int count = 0, subCount = 0, startCount = 0, endCount = 0, insert = 0, sceneCount = 0;
XmlTextReader reader = new XmlTextReader(xmlPath);
while (reader.Read())
{
   XmlNodeType nodeType = reader.NodeType;
   switch (nodeType)
   {
      case XmlNodeType.Element:
         name = reader.Name;
         if (reader.HasAttributes)
         {
            for (int i = 0; i < reader.AttributeCount; i++)
            {
               reader.MoveToAttribute(i);

               if (reader.Name == "id" && name == "subject")
               {
                  previous = reader.Value;
                  characterName = "subject_0" + reader.Value;

                  filepath = "C:\\processed\\" + mTitle + ".mp4\\thumbnails\\" +
```

```
        characterName;
foreach (string f in Directory.GetFiles(filepath))
   file = f;

//Modify path for storage in database
string cut = newPath.Replace("C:\\Users\\Domino\\Documents\\Visual
        Studio 2010\\WebSites\\FilmAnalysis\\", "");
string tfm = cut.Replace("\\", "/");
nomen = tfm + "/" + Path.GetFileName(file);
byte[] imageData = ReadFile(file);
//Insert character data
SqlConnection charCon = new SqlConnection();
charCon.ConnectionString = ConfigurationManager.ConnectionStrings
         ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand charCmd = new SqlCommand("CharInsert", charCon);
charCmd.CommandType = System.Data.CommandType.StoredProcedure;
charCmd.Parameters.AddWithValue("@MovieID", mId);
charCmd.Parameters.AddWithValue("@FirstName", characterName);
charCmd.Parameters.AddWithValue("@FilePath", file);
charCmd.Parameters.AddWithValue("@binImage", (object)imageData);
charCmd.Parameters.AddWithValue("@FileName", nomen);
charCmd.Parameters.Add("@Return_Value", SqlDbType.Int, 4);
charCmd.Parameters["@Return_Value"].Direction =
        ParameterDirection.ReturnValue;
charCon.Open();
charCmd.ExecuteNonQuery();
getCharacterID = charCmd.Parameters["@Return_Value"].Value.ToString();
store[count] = getCharacterID;
charCon.Close();

foreach (string f in Directory.GetFiles(filepath))
{
   file = f;
   nomen = tfm + "/" + Path.GetFileName(file);
   byte[] galImageData = ReadFile(file);
   //Insert character images in gallery
   SqlConnection galCon = new SqlConnection();
   galCon.ConnectionString = ConfigurationManager.ConnectionStrings
          ["FilmAnalysisDBConnectionString"].ConnectionString;
   SqlCommand galCmd = new SqlCommand("GalleryInsert", galCon);
   galCmd.CommandType = System.Data.CommandType.StoredProcedure;
   galCmd.Parameters.AddWithValue("@CharacterID", store[count]);
   galCmd.Parameters.AddWithValue("@Picture", (object)galImageData);
   galCmd.Parameters.AddWithValue("@FilePath", file);
```

```
            galCmd.Parameters.AddWithValue("@FileName", nomen);
            galCon.Open();
            galCmd.ExecuteNonQuery();
            galCon.Close();
        }

        if (System.IO.Directory.Exists(filepath))
        {
            string[] files = System.IO.Directory.GetFiles(filepath);
            // Copy the files and overwrite destination files if they already exist.
            foreach (string s in files)
            {
                // Use static Path methods to extract only the file name from the path.
                fileName = System.IO.Path.GetFileName(s);
                destFile = System.IO.Path.Combine(newPath, fileName);
                System.IO.File.Copy(s, destFile, true);
            }
        }

        count++;
    }
    else if (reader.Name == "id" && name == "track")
    {
        subject[subCount] = previous;
        subCount++;
    }
    else if (reader.Name == "startFrame")
    {
        start[startCount] = reader.Value;
        startCount++;
    }
    else if (reader.Name == "endFrame")
    {
        end[endCount] = reader.Value;
        endCount++;
        //Insert scene details for each scene
        SqlConnection sceneCon = new SqlConnection();
        sceneCon.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
        SqlCommand sceneCmd = new SqlCommand("SceneInsert", sceneCon);
        sceneCmd.CommandType = System.Data.CommandType.StoredProcedure;
        sceneCmd.Parameters.AddWithValue("@MovieID", mId);
        sceneCmd.Parameters.AddWithValue("@startFrame", start[insert]);
        sceneCmd.Parameters.AddWithValue("@endFrame", end[insert]);
```

```
                    sceneCmd.Parameters.Add("@Return_Value", SqlDbType.Int, 4);
                    sceneCmd.Parameters["@Return_Value"].Direction =
                        ParameterDirection.ReturnValue;
                    sceneCon.Open();
                    sceneCmd.ExecuteNonQuery();
                    getSceneID = sceneCmd.Parameters["@Return_Value"].Value.ToString();
                    scenes[sceneCount] = getSceneID;
                    sceneCount++;
                    sceneCon.Close();

                    SqlConnection finalCon = new SqlConnection();
                    finalCon.ConnectionString = ConfigurationManager.ConnectionStrings
                        ["FilmAnalysisDBConnectionString"].ConnectionString;
                    SqlCommand finalCmd = new SqlCommand("InsertSceneChar", finalCon);
                    finalCmd.CommandType = System.Data.CommandType.StoredProcedure;
                    finalCmd.Parameters.AddWithValue("@CharacterID", getCharacterID);
                    finalCmd.Parameters.AddWithValue("@SceneID", getSceneID);
                    finalCmd.Parameters.AddWithValue("@MovieID", mId);
                    finalCon.Open();
                    finalCmd.ExecuteNonQuery();
                    finalCon.Close();

                    insert++;
                }
            }
        }
        break;
    }
  }
  Response.Redirect("Review.aspx");
}

byte[] ReadFile(string sPath)
{
  //Initialize byte array with a null value initially.
  byte[] data = null;
  //Use FileInfo object to get file size.
  FileInfo fInfo = new FileInfo(sPath);
  long numBytes = fInfo.Length;
  //Open FileStream to read file
  FileStream fStream = new FileStream(sPath, FileMode.Open, FileAccess.Read);
  //Use BinaryReader to read file stream into byte array.
  BinaryReader br = new BinaryReader(fStream);
  data = br.ReadBytes((int)numBytes);
```

```
        return data;
    }
}


using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
using System.IO;

public partial class Movie : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        bool isOk = false;
        //Insert movie data
        SqlConnection con = new SqlConnection();
        con.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
        SqlCommand cmd = new SqlCommand("MovieInsert", con);
        cmd.CommandType = System.Data.CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Title", txtTitle.Text);
        cmd.Parameters.AddWithValue("@Director", txtDirector.Text);
        cmd.Parameters.AddWithValue("@Country", txtCountry.Text);
        cmd.Parameters.AddWithValue("@YearProduced", txtYear.Text);
        cmd.Parameters.AddWithValue("@ProducedBy", txtProducedBy.Text);
        cmd.Parameters.AddWithValue("@Genre", txtGenre.Text);
        cmd.Parameters.AddWithValue("@RunTime", txtRunTime.Text);
        cmd.Parameters.AddWithValue("@ReviewerID", txtId.Text);
        try
        {
            if (con.State == System.Data.ConnectionState.Closed)
            {
```

```
            con.Open();
        }
        cmd.ExecuteNonQuery();
        isOk = true;
        uiMessage.Text = "some message";
    }
    catch (Exception ex)
    {
        this.uiMessage.Text = "err on inserting stored procedure " + ex;
    }
    finally
    {
        con.Close();
    }

    //Enable users to upload a video
    if (FileUpload1.HasFile)
        try
        {
            FileUpload1.SaveAs("C:\\videos\\" +
                FileUpload1.FileName);
            uiMessage.Text = "File name: " +
                FileUpload1.PostedFile.FileName + "<br>" +
                FileUpload1.PostedFile.ContentLength + " kb<br>" +
                "Content type: " +
                FileUpload1.PostedFile.ContentType;
        }
        catch (Exception ex)
        {
            uiMessage.Text = "ERROR: " + ex.Message.ToString();
        }
    else
    {
        uiMessage.Text = "You have not specified a file.";
    }
    string url;

    url = "Review.aspx?id=" + txtId.Text.ToString();
    Response.Redirect(url);
  }
}


using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;

public partial class PossibleFalse : System.Web.UI.Page
{
   public int mId, movie;
   protected void Page_Load(object sender, EventArgs e)
   {
      lnkFalse.Text = "Click Here";
      lnkGallery.Text = "Image Gallery";
      string file = Request.QueryString["FileName"];
      SqlConnection con = new SqlConnection();
      con.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
      SqlCommand cmd = new SqlCommand("Select CharacterID FROM Gallery
            WHERE FileName LIKE '" + file + "'", con);
      cmd.CommandType = System.Data.CommandType.Text;
      con.Open();
      mId = (int)cmd.ExecuteScalar();
      con.Close();
      Session[“PathDir"] = Request.QueryString["FileName"];


      SqlConnection mCon = new SqlConnection();
      mCon.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
      SqlCommand mCmd = new SqlCommand("Select MovieID FROM Character
            WHERE CharacterID =" + mId, mCon);
      mCmd.CommandType = System.Data.CommandType.Text;
      mCon.Open();
      movie = (int)mCmd.ExecuteScalar();
      mCon.Close();
   }
   protected void lnkFalse_Click(object sender, EventArgs e)
   {
      string url;
      url = "FalseIdentify.aspx?MovieID=" + movie;
      Response.Redirect(url);
```

```
    }
    protected void lnkGallery_Click(object sender, EventArgs e)
    {
        string url;
        url = "ImageGallery.aspx?CharacterID=" + mId.ToString();
        Response.Redirect(url);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.IO;

public partial class Remove : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //Character falsely identified is the same as another character
        string mFile = (string)(Session["PathDir "]);
        int charID = -1;
        string id = "", chId = "", path = "";
        string pth = "", name = "";
        //Get information of character falsely identified
        SqlConnection fgtnCon = new SqlConnection();
        fgtnCon.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
        SqlCommand fgtnCmd = new SqlCommand("Select ProfileID, CharacterID, FilePath
                FROM Gallery WHERE FileName LIKE '" + mFile + "'", fgtnCon);
        fgtnCmd.CommandType = System.Data.CommandType.Text;
        fgtnCon.Open();
        SqlDataReader fgtnRead = null;
        fgtnRead = fgtnCmd.ExecuteReader();
        while (fgtnRead.Read())
        {
            id = fgtnRead[0].ToString();
            chId = fgtnRead[1].ToString();
            path = fgtnRead[2].ToString();
```

```
}

fgtnCon.Close();

//Determine if falsely identified character has the main profile picture
SqlConnection cCon = new SqlConnection();
cCon.ConnectionString = ConfigurationManager.ConnectionStrings
        ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand cCmd = new SqlCommand("Select CharacterID FROM Character
        WHERE FileName LIKE '" + mFile + "'", cCon);
cCmd.CommandType = System.Data.CommandType.Text;
try
{
   if (cCon.State == System.Data.ConnectionState.Closed)
   {
      cCon.Open();
   }
   charID = (int)cCmd.ExecuteScalar();
}
catch (Exception ex)
{
   this.lblMessage.Text = "err on inserting stored procedure " + ex;
}
finally
{
   cCon.Close();
}

//Update gallery images
SqlConnection con = new SqlConnection();
con.ConnectionString = ConfigurationManager.ConnectionStrings
        ["FilmAnalysisDBConnectionString"].ConnectionString;
SqlCommand command = con.CreateCommand();
command.CommandText = "UPDATE Gallery SET CharacterID = @CharacterID
        WHERE FileName = @FileName";
command.Parameters.Add("@FileName", SqlDbType.NVarChar);
command.Parameters["@FileName"].Value = mFile;
command.Parameters.AddWithValue("@CharacterID",
       Request.QueryString["CharacterID"]);
con.Open();
command.ExecuteNonQuery();
con.Close();

if (charID == -1 || charID == 0)
```

```
{
}
else
{
   //Replace profile picture with new image in gallery
   SqlConnection sCon = new SqlConnection();
   sCon.ConnectionString = ConfigurationManager.ConnectionStrings
         ["FilmAnalysisDBConnectionString"].ConnectionString;
   SqlCommand sCmd = new SqlCommand("Select FilePath, FileName FROM Gallery
         WHERE CharacterID = '" + chId + "'", sCon);
   sCmd.CommandType = System.Data.CommandType.Text;
   sCon.Open();
   SqlDataReader sRead = null;
   sRead = sCmd.ExecuteReader();
   while (sRead.Read())
   {
      pth = sRead[0].ToString();
      name = sRead[1].ToString();
   }
   sCon.Close();

   if (!pth.Equals(""))
   {
      byte[] imageData = ReadFile(pth);
      //Update character with new image from gallery
      SqlConnection chCon = new SqlConnection();
      chCon.ConnectionString = ConfigurationManager.ConnectionStrings
            ["FilmAnalysisDBConnectionString"].ConnectionString;
      SqlCommand chCmd = chCon.CreateCommand();
      chCmd.CommandText = "UPDATE Character SET FilePath = @FilePath,
            FileName = @FileName WHERE CharacterID = @CharacterID";
      chCmd.Parameters.Add("@CharacterID", SqlDbType.Int);
      chCmd.Parameters["@CharacterID"].Value = chId;
      chCmd.Parameters.AddWithValue("@FilePath", pth);
      chCmd.Parameters.AddWithValue("@binImage", (object)imageData);
      chCmd.Parameters.AddWithValue("@FileName", name);
      chCon.Open();
      chCmd.ExecuteNonQuery();
      chCon.Close();
   }
   else
   {
      //Delete scenes associated with falsely identified character
      SqlConnection dsCon = new SqlConnection();
```

```
            dsCon.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
            SqlCommand dsCmd = dsCon.CreateCommand();
            dsCmd.CommandText = "Delete From Scene_Character
                Where CharacterID = @CharacterID";
            dsCmd.Parameters.Add("@CharacterID", SqlDbType.Int);
            dsCmd.Parameters["@CharacterID"].Value = chId;
            dsCon.Open();
            dsCmd.ExecuteNonQuery();
            dsCon.Close();
            //Delete falsely identified character
            SqlConnection rcCon = new SqlConnection();
            rcCon.ConnectionString = ConfigurationManager.ConnectionStrings
                ["FilmAnalysisDBConnectionString"].ConnectionString;
            SqlCommand rcCmd = rcCon.CreateCommand();
            rcCmd.CommandText = "Delete From Character Where FileName = @FileName";
            rcCmd.Parameters.Add("@FileName", SqlDbType.VarChar, 255);
            rcCmd.Parameters["@FileName"].Value = mFile;
            rcCon.Open();
            rcCmd.ExecuteNonQuery();
            rcCon.Close();
        }
    }

    string url;
    url = "ImageGallery.aspx?CharacterID=" + Request.QueryString["CharacterID"];
    Response.Redirect(url);
  }

  byte[] ReadFile(string sPath)
  {
    //Initialize byte array with a null value initially.
    byte[] data = null;
    //Use FileInfo object to get file size.
    FileInfo fInfo = new FileInfo(sPath);
    long numBytes = fInfo.Length;
    //Open FileStream to read file
    FileStream fStream = new FileStream(sPath, FileMode.Open, FileAccess.Read);
    //Use BinaryReader to read file stream into byte array.
    BinaryReader br = new BinaryReader(fStream);
    data = br.ReadBytes((int)numBytes);
    return data;
  }
}
```