
Software Documentation

for

The Film Club Content Management System

Version 2.0 approved

Prepared by

Chad Golden
O'Neal Jones
Patrick Meehan
Lamar Sims
Matthew Stratton

Group #5

February 24, 2015

Executive Summary

This document is the *Software Specification* document for the Georgia Southern Film Club *Content Management System (CMS)*. Provided will be an introduction to the system including the scope and purpose. Next the limitations and problems of the current system in place for the organization will be analyzed along with the plan for the solution proposed by the development team. The requirements for this proposed system will be elicited and analyzed in the form of *use cases*. Lastly, a glossary of technical terms will be provided, and a collection of diagrams and *user interface* designs will be provided in the Appendix along with the proposed project schedule which includes a diary of meetings and tasks amongst the developers and *stakeholders*.

Table of Contents

Executive Summary	ii
Table of Contents	iii
Revision History	v
1. Introduction	1
1.1 Purpose of the System	1
1.2 Scope of the System	1
1.3 Development Methodology	2
1.4 Definitions, Acronyms, and Abbreviations	2
1.5 Overview	3
2. Current System (Limitations and problems).....	4
2.1 Existing System	4
3. Project Plan	5
3.1 Project Organization	6
3.2 Hardware and Software Requirements	7
3.3 Work Breakdown	8
4. Proposed System	9
4.1 Functional and Nonfunctional Requirements	9
4.2 Explanations of Diagrams for Implemented Use Cases	40
4.3 Requirements Analysis	42
5. Software Architecture.....	44
5.1 Overview	45
5.2 Subsystem Decomposition	47
5.3 Hardware and Software Mapping.....	49
5.4 Persistent Data Management.....	51
5.5 Security/Privacy.....	66
6. Detailed Design.....	68
6.1 Overview	68
6.2 Object Interaction	71
6.3 Detailed Class Design	73
7. Testing Process	74
7.1 Software Quality Assurance Plan.....	74

7.2	System Tests.....	75
7.3	Subsystem Tests.....	103
7.4	Evaluation of Tests.....	109
7.5	Testing Tools	113
8.	Glossary.....	115
9.	Appendix	117
9.1	Appendix A: Project Schedule	117
9.2	Appendix B: Diagrams.....	118
9.3	Appendix C: User Interface Designs	131
9.4	Class Diagrams.....	136
9.5	Class Interfaces	140
9.6	Testing	141
9.7	References	159
9.8	Appendix D: Diary of Meeting and Tasks.....	160

Revision History

Table i-1 History of Changes

Name	Date	Reason For Changes	Version
Team	03/02/2015	Software Requirements Specification	1.0
Team	04/02/2015	SRS Revision	1.1
Team	05/01/2015	Final Software Documentation	2.0

1. Introduction

This section is devoted to giving an overview of the contents in this *software specification* document. Provided are the scope of the system, its purpose, and a collection of definitions and abbreviations, and an overview of what to expect in the coming parts of this document.

1.1 Purpose of the System

The purpose of the system is to provide the Georgia Southern Film Club with a useful set of tools to aid in the tasks involved within the organization. The proposed solution is a *content management system (CMS)* that will provide administration over information and media that the film club wishes to share over web pages accessible through the Internet.

1.2 Scope of the System

The system shall allow for the public viewing of information pertinent to the organization and its members. Among the major benefits the system provides to the organization is the content management aspect of the solution. A user with administrative privileges to the website shall have the capability to add, edit, and delete relevant content to web pages through a simple to use interface. Through this interface, the system shall provide content management without possessing any software development skills.

The public shall be able to view basic information about the organization that is not limited to: the mission of the organization, what entities to contact for further information, news and blog, a roster, an event calendar, and media (e.g. images and videos). More specialized features shall be provided to users registered through a login system (e.g. posting to a discussion board and sending personal messages to an administrator of the organization).

The overarching benefit, objective, and goal is to provide a tool in order to assist the organization in the advancement its own endeavors using the proposed system.

1.3 Development Methodology

The development team for the Film Club CMS exercised a software development method known as “agile software development”. Agile Alliance states that agile software development is a method that emphasized “close collaboration between the programmers and business experts; face-to-face communication; frequent delivery of new deployable business value; tight, self-organizing teams.”

Because the client was available for instant feedback for the implementations of the software requirements, the agile methodology was a good fit for the project. Meetings with client often composed of showing off the latest implementations on a weekly basis and eliciting additional requirements in the process.

The development team utilized *extreme programming (XP)* techniques that fall in line with agile methodology. The intent of the XP philosophy is to provide rapid development cycles. The development team adhered to this in order to gather feedback and further advance into more development cycles. Pair programming, another feature of the XP paradigm where the development team often had two members writing the software, provided on the spot code review and forced a coding style that each team member can comprehend and maintain.

1.4 Definitions, Acronyms, and Abbreviations

Table 1.3-1 Definitions, Acronyms, and Abbreviations

Term	Description
CMS	<i>Content Management System.</i> A means for privileged users to add, edit, and remove content such as web pages, blog posts, and media through the aid of software.
Extreme Programming (XP)	A software development methodology whose intent is to provide rapid software release cycles and a high level of adaptability to changing stakeholder requirements.

1.5 Overview

The remainder of this document will cover topics pertinent to the Film Club CMS. These include: the limitations of the organization's current system, the plan of execution for this project, the system's requirements and analysis of these requirements, the architectural design of the software system, detailed design of the systems specific subsystem, the testing process, a glossary of terms, and an appendix containing graphical representations of some of the concepts found in the document.

2. Current System (Limitations and problems)

This section will identify the specific problems and limitations of the current system provided for the Georgia Southern Film Club.

2.1 Existing System

The current content management system the organization possesses is provided by the university to all student organizations. A major problem with the current system is that it is a “one-size-fits—all” solution. Nearly every website for each organization looks exactly the same. Aside from social media links, profile pictures, and contact information, the current system lacks distinguishing features in terms of appearance and content. In addition, the current website is difficult to find outside of a specific search query through a search engine. As a result of the current system’s shortcomings, the current system is rarely updated by the administrators because there is very little to update given the current feature set. Outside of the public obtaining general information (e.g. contact information), the system provides little worth to the club. Current members have no motivation to utilize the system as there is no feature that provides anything useful from their perspective.

The current system has eight navigation links which include: “Home”, “News”, “Profile”, “Events”, “Roster”, “Gallery”, “Documents”, and “Forms”. The “Home” page simply shows website users a profile picture, the goal of the organization, and the contact information for the organization. It also provides website users a button to join the organization and a direct connection to the film club on Facebook and Twitter. The “News” page currently contains no information. The “Profile” page merely restates the organization’s goals. The “Events” page shows both upcoming and past events; both fields are currently blank. The “Roster” page shows the user the officers and members of the film club. This page is the most utilized page of the current system as it contains the most information. The “Gallery” page shows users a collection of pictures of what the club has done in past events. The “Documents” page shows the constitution and bylaws of the organization. The “Forms” page shows forms that users must be signed in to view. As a whole, the website uses an identical layout across all webpages. If a page is not completely filled with information the page will contain a large void at the bottom of the screen.

3. Project Plan

This section will outline the development team's plan of tackling the project. The roles of the team members of the project will be specified. In addition some baseline hardware and software requirements will be examined that are necessary in order to deploy the application. Lastly, there will be a breakdown of the project milestones and deliverables.

3.1 Project Organization

Table 3.1-1 Project Team Member Roles

Team Member	Role(s)	Description
Chad Golden	i) Project Leader ii) Database Administrator iii) Software Developer iv) Architectural Design	i) Leads project. ii) Implements data requirements into database system. iii) Writes code for software. iv) In charge of designing the system architecture and module design.
O'Neal Jones	i) Human-Computer Interaction ii) Software Developer iii) Author, User Manual	i) Manages the usability of the system. ii) Writes code for software. iii) Authored the user manual.
Patrick Meehan	i) Software Developer ii) Object-Oriented Design	i) Writes code for software. ii) Models the required classes and objects for the application and this document.
Lamar Sims	i) Software Testing Lead ii) Software Developer iii) Functional Testing Tool Specialist	i) Extensively tests the software. ii) Writes code for software. iii) Organized and implemented Selenium for testing with the system.
Matthew Stratton	i) Business Analyst ii) Software Developer iii) Software Testing Specialist	i) Consults with the stakeholders. ii) Writes code for software. iii) Extensively tests the software.

3.2 Hardware and Software Requirements

The system was developed with both *PHP* and *HTML*. As an *IDE*, the team will use *PHP-Storm*, which is an integrated web development software by *JetBrains* where the team can easily pull all of the necessary *HTML* and *PHP* code online. For visual design, the team is using a popular web *framework* known as *Bootstrap*. This makes the design process much easier, and will make the website look much better.

On the *server* end, an *operating system* (Debian Linux or Microsoft Windows) is required to run the web software stack. At a minimum to run the required software stack (Apache Web Server + PHP + MySQL + Film Club CMS) the server machine needs to have at least 128 megabytes of RAM, an 800 MHz CPU, and one gigabyte of hard disk storage. To run smoothly and scale for the possibility of many simultaneous users, it is recommended the server have at a minimum: 2 gigabytes of RAM, a 3000 MHz dual-core CPU, and ten gigabytes of hard disk storage. In terms of disk storage, the system that uses the most of this resource is the media upload system. The system does place limits on the size (in terms of bytes) of media that can be uploaded to fit the constraints of limited hard disk space.

On the *client* end, there are no specific hardware requirements for the system other than having a device that runs a *web browser*. The system is optimized on both mobile and PC platforms. This is thanks to the *Bootstrap* framework. The system shall be optimized for the Google Chrome web browser, but other major browsers should display the website reasonably well.

3.3 Work Breakdown

This identifies the milestones and deliverables for this project. Please refer to Appendix A for a graphical representation of these deliverables. Refer to Appendix B for a detailed log of operations among the *stakeholders* and team members.

Table 3.3-1 Milestones and Deliverables

Name of Deliverable	Date	Description
SRS	03/02/2015	Software Requirements Specification.
SRS Presentation	03/26/2015	A brief description of the SRS.
Final Software Specification	05/01/2015	Written description of the software product.
Final Presentation	05/05/2015	An oral presentation of the software system delivered by the development team.

4. Proposed System

The system being developed for the GSU film club is aimed to provide members with an organized and collaborative environment to cultivate their interests. This is done by allowing club leaders and members to create profiles, events, project suggestions, and personal opinion discussions whereby an administrator can manage the content through a simple dashboard interface and users of the system without needing the low-level knowledge of how the system works under the hood. on one easy to use web system. Rather than relying on social media pages or email, the system gives members a home base to coordinate all club functions. Tools such as the calendar and discussion board allow the users to see a clear layout of upcoming events, and talk about ideas. Users not in the film club can learn more about it and have access to contact information.

Below is the core of this document that outlines the key features and requirements in the form of *use cases*.

4.1 Functional and Nonfunctional Requirements

Listed here are the system's use cases in tabular form. Individual use cases shall be referenced by their use-case identification (UCID) number that corresponds to their *4.1.X* headings (i.e. the first use-case is *UC-4.1.1*).

Diagrams of the subsystems pertaining to these use cases can be found in Appendix B.

4.1.1 System Navigation

Use Case 4.1-1 Traverse the Site/System

Function	Navigate the System
Description	A visitor, account holder, or administrator navigates the site.
Inputs	Links, Navigation Bar Controls
Actor	Visitor, Account Holder, Film Club Member, or Administrator
Source	Actor
Action	The actor clicks a link on a webpage via a hyperlink or button, and the actor is redirected to the page indicated by the link or graphical element.
Requirements	The actor has an internet connection and a web browser.
Pre-Condition	The actor has navigated to the site in his or her web browser.
Post-Condition	N/A.

4.1.2 User Registration

Use Case 4.1-2 User Registration

Function	User Registration
Description	A visitor shall be able to register an account with the system.
Inputs	Username, Email, Password, and Password Confirmation. (Text)
Actor	Visitor
Source	Actor
Action	<p>Inputs are taken in and evaluated to ensure specified Username and Email have not been used already and the password is of the minimum length. If specified information is valid then the information is saved for the actor.</p> <p>Authentication Error: If the information specified is invalid, system will discard information and prompt actor for the information again.</p>
Requirements	Refer to Table 4.1-1.
Pre-Condition	To not have a member already registered with the same username and email.
Post-Condition	Actor now holds an account with the system and is allowed to login.

4.1.3 User Login

Use Case 4.1-3 User Login

Function	User Login
Description	The system shall allow a user to sign-in as a member.
Inputs	Email and Password. (Text)
Actor(s)	Account Holder
Source	Actor
Action	<p>Inputs are taken in and evaluated for validity against the information stored in the system. If specified inputs are valid then the actor is signed-in to the system.</p> <p>Authentication Error: If the inputs specified by the actor is invalid, system will discard inputs and prompt actor for the inputs again.</p>
Requirements	Refer to UC 4.1-3.
Pre-Condition	Actor must be registered in the system and not signed-in at the time of trying to sign-in.
Post-Condition	Actor is now signed-in to the system.

4.1.4 User Logout

Use Case 4.1-4 User Logout

Function	User Logout
Description	The system shall allow a user to sign out if they are signed in.
Inputs	The “Logout” button on the navigation bar.
Actor(s)	An account holder signed into the system
Source	Actor
Action	Actor clicks the logout button on the navigation bar.
Requirements	The navigation bar is visible.
Pre-Condition	Actor must be signed into the system.
Post-Condition	Actor is now signed out of the system.

4.1.5 Social Media Navigation

Use Case 4.1-5 Social Media Navigation

Function	Social Media Navigation
Description	A website user clicks one of the links to navigate to the Film Club's profile on an external social media website.
Inputs	Mouse-click of one of the social media buttons.
Actor(s)	Any user of the website.
Source	Actor
Action	The actor clicks one of the social media buttons identified by the particular social media entity's logo. These buttons are located within the footer of each page of the website.
Requirements	The actor is on a page where the website footer is visible.
Pre-Condition	See Table 4.1-1
Post-Condition	The actor is now on a social media site external to the system.

4.1.6 Profile Image Creation

Use Case 4.1-6 Profile Image Creation

Function	Account Holder Uploads a Profile Image
Description	A website account holder should be allowed to upload their profile image.
Inputs	Image file (File Explorer Dialog)
Actor(s)	Account Holder
Source	Actor
Action	<p>Actor will press upload image button and supply the image file through a file selection dialog. If successful image file will be saved for actor.</p> <p>File Error: If unsuccessful inputs will be deleted and actor will be prompted to enter image file again.</p>
Requirements	Actor is on their personal profile page.
Pre-Condition	Actor must be signed in.
Post-Condition	Actor now has a profile picture.

4.1.7 Profile Image Replacement

Use Case 4.1-7 Profile Image Replacement

Function	Replace Profile Image
Description	Should allow account holder to replace their profile image.
Inputs	Image file (File Explorer Dialog)
Actor(s)	Account Holder
Source	Actor
Action	<p>Actor will press upload image button and supply the image file. If successful image file will be saved for actor.</p> <p>File Error: If unsuccessful inputs will be deleted and actor will be prompted to enter image file again.</p>
Requirements	Actor is on their profile page.
Pre-Condition	Actor is signed in and has a profile picture.
Post-Condition	Actor has changed profile picture.

4.1.8 Profile Image Deletion

Use Case 4.1-8 Profile Image Deletion

Function	Delete Profile Image
Description	An account holder should be allowed to delete their profile image.
Inputs	Delete Button (Mouse click)
Actor(s)	Account Holder
Source	Actor
Action	<p>Actor presses button to delete profile image. If successful profile image will be deleted for actor.</p> <p>File Error: If unsuccessful image will not be deleted, and actor must try again.</p>
Requirements	Actor is on their profile page.
Pre-Condition	Actor is signed in and has a profile picture.
Post-Condition	Actor no longer has a profile picture.

4.1.9 Discussion Thread Creation

Use Case 4.1-9 Discussion Thread Creation

Function	Create a discussion thread
Description	Shall allow a member to create a discussion thread, which allows other members to reply to the created thread for discussion.
Inputs	Topic to be discussed The body of text on the discussion topic.
Actor(s)	Account Holder
Source	Actor
Action	<p>Actor will go to the discussion page and press the button to create a new thread. The actor will then be presented with a form to enter topic to be discussed and an empty field for the body of text on the discussion topic. The actor will then post and create the discussion thread. If successful the thread will now be viewable to other users and allowing other members to reply to the created thread.</p> <p>Error: If unsuccessful actor will be prompted to try again or to enter a required input.</p>
Requirements	Actor must be on the Discussion page.
Pre-Condition	Actor must be signed in.
Post-Condition	Actor's discussion thread is created and displayed in the discussion page.

4.1.10 Reply to a Discussion Thread**Use Case 4.1-10** Reply to a Discussion Thread

Function	Reply to a Discussion Thread
Description	Shall allow a member to create a sub-thread to another member's discussion thread or sub-thread, which allows other members to reply to the created sub-thread.
Inputs	Body of text for the reply.
Actor(s)	Account Holder
Source	Actor.
Action	<p>Actor will go to the discussion page and click reply to an existing discussion thread to make a reply. The actor will then be presented with an empty field, for the body of text, to reply. Once the actor is finished they will posts the reply. If successful the reply will now be viewable to other users and allowing other members to offer a reply.</p> <p>Error: If unsuccessful actor will be prompted to try again.</p>
Requirements	Actor must be on the discussion page and there must be a discussion thread to be replied to.
Pre-Condition	Actor must be signed in.
Post-Condition	Actor's reply is now displayed in the discussion page under the discussion thread that was replied to.

4.1.11 Calendar Event Creation**Use Case 4.1-11** Event Creation

Function	Create an Event on the Calendar
Description	Shall allow a member to create an event and place it on the calendar, to be viewed by other users of the system.
Inputs	Name of event, date and time of event, location, description.
Actor(s)	Account Holder with Administrator Privileges
Source	Actor
Action	<p>Actor will go to the events page and will then have to click the create events button, where they will be prompted with a form to fill out the events' information. Once the actor is finished filling out the events form they can then save and create the event. If successful the event will be saved, created, and will be displayed in the calendar, on the events page.</p> <p>Error: If unsuccessful the event will not be saved and created and the actor will be prompted to try again or ensure all required inputs have inputs.</p>
Requirements	Actor must be on events page and the event being made must not conflict with another previous created events.
Pre-Condition	Actor must be signed in and have administrator privileges.
Post-Condition	Event is now created and displayed, in the calendar on the events page, to allow anyone to view.

4.1.12 Event Modification****Use Case 4.1-12** Edit an Event

Function	Edit an event on the calendar
Description	Shall allow a user to edit a previously created event and redisplay the edited event.
Inputs	Name of event, date and time of event, location, description.
Actor(s)	Administrator
Source	Actor
Action	<p>Actor will go to the events page and will then have to click on the previously created event, where they will be prompted with an option to edit the event, upon selecting edit a form to edit the events' information will appear. Once the actor is finished editing the event they can then save and create the event again. If successful the event will be saved, created, and will be redisplayed.</p> <p>Error: If unsuccessful the event will not be saved or created and the actor will be prompted to try again or to check the required inputs.</p>
Requirements	Actor must be on the events page.
Pre-Condition	Actor must be signed in, must be an event available, and the edited event must not conflict with another previously created event.
Post-Condition	Event is now altered and displayed in the calendar on the events page to allow other users to view.

4.1.13 Event Deletion

Use Case 4.1-13 Delete an Event

Function	Delete an Event on the Calendar.
Description	Shall allow a user to delete a previously created event.
Inputs	Delete Button
Actor(s)	Administrator
Source	Actor
Action	<p>Actor will go to the events page and will then have to click on the previously created event, where they will be prompted with an option to delete the event, upon selecting delete a confirmation will display to ensure action should be carried out. If successful event will be deleted.</p> <p>Error: If unsuccessful the actor will be prompted to try again.</p>
Requirements	Actor must be on the events page.
Pre-Condition	Actor must be signed-in, and there must be an event available to delete.
Post-Condition	Event is now deleted and is not displayed in the calendar on the events page anymore.

4.1.14 Calendar Navigation

Use Case 4.1-14 Calendar Navigation

Function	Navigate the Calendar
Description	A website user can use the arrow buttons on the calendar to view different months and the events have taken place or are to take place in the future.
Inputs	Arrow buttons indicating “Prev” (past) and “Next” (future) navigation of the calendar.
Actor(s)	Anyone viewing the site
Source	Actor
Action	The actor uses the “Prev” and “Next” buttons to navigate through the past, current, and future months on the calendar. Events on the calendar will appear as a clickable link.
Requirements	The actor is on the Events page.
Pre-Condition	N/A.
Post-Condition	N/A.

4.1.15 Custom Page Creation

Use Case 4.1-15 Custom Page Creation

Function	Administrator Creates a Page
Description	An administrator creates a page structured with a title, body, and comments.
Inputs	Create a Page Button, Form Fields
Actor(s)	Administrator
Source	Actor
Action	<p>The actor clicks the Custom Pages button on the dashboard menu. From there the actor fills a form containing the title, body content (this could contain text, images, or YouTube embeds). Upon completion of the form, the actor clicks the Submit button. If all input was valid the actor will be redirected to the newly created page.</p> <p>Error: If the information is invalid, an error message will show and the actor will be prompted to reenter valid information.</p>
Requirements	The actor is on the Dashboard page.
Pre-Condition	The actor is logged in.
Post-Condition	The actor has created a new page on the system that can be linked to and viewed by all.

4.1.16 Custom Page Modification**Use Case 4.1-16** Custom Page Modification

Function	Administrator Edits a Page
Description	An administrator edits a previously created custom page. See UC 4.1-15
Inputs	Edit Button Form Fields
Actor(s)	Administrator
Source	Actor
Action	<p>The actors clicks the Edit Button located on the page he or she wishes to modify. The actor will fill out the form information as seen in the page creation process. The form will appear with the information filled in the fields that reflects the page's current state. When satisfied with the changes, the actor will click the Submit button and he or she will be redirected to the newly modified page.</p> <p>Error: If the information is invalid, an error message will show and the actor will be prompted to reenter valid information.</p>
Requirements	The page the actor wishes to modify exists.
Pre-Condition	See Requirements and UC 4.1-15.
Post-Condition	The actor has modified the custom page.

4.1.17 Custom Page Deletion**Use Case 4.1-17** Custom Page Deletion

Function	Administrator Deletes a Page
Description	An administrator deletes a previously created page.
Inputs	Delete Button Confirm Dialog
Actor(s)	Administrator
Source	Actor
Action	The administrator clicks the Delete button located on the page of interest. Upon pressing the button, the actor will confirm or cancel the deletion through a confirmation dialog.
Requirements	The page the actor wishes to delete exists.
Post-Condition	The actor has modified the custom page.

4.1.18 Administrator Reads Messages

Use Case 4.1-18 Administrator Reads Messages

Function	Administrator Reads Messages
Description	An administrator reads messages concerning potentially urgent matters.
Inputs	
Actor(s)	Administrator
Source	Actor
Action	On the Dashboard page a list of messages will appear in the body of the page, sorted to where the newest messages are displayed first. Messages older than 30 days should be deleted automatically.
Requirements	The actor is on the Dashboard page.
Pre-Condition	The actor is signed in.
Post-Condition	The actor is able to read the messages.

4.1.19 Administrator Suspends User**Use Case 4.1-19** Administrator Suspends User

Function	Administrator Suspends Account Holder
Description	An administrator suspends the account of a problematic user.
Inputs	Dashboard View Users Button Suspend User Button
Actor(s)	Administrator Account Holder
Source	Actor
Action	The administrator clicks on the Manage Users Button on the Dashboard page. A list of the users on the website will appear. From this list the administrator may click the Suspend User Button next to the user's record. This will disable the account holder's profile. The account holder will now be unable to sign in.
Requirements	The actor is on the Dashboard page.
Pre-Condition	The actor is signed in.
Post-Condition	The actor has disabled the account of a problematic user.

4.1.20 Administrator Promotes Account Holder**Use Case 4.1-20** Administrator Promotes Account Holder

Function	Administrator Promotes Account Holder to Film Club Member Status
Description	An administrator promotes an account to film club member status.
Inputs	Dashboard View Users Button Promote Button
Actor(s)	Administrator Account Holder
Source	Actor
Action	The administrator clicks on the Manage Users Button on the Dashboard page. A list of the users on the website will appear. From this list the administrator may click the Promote User Button. This action will advance the account holder's privilege level to that of a film club member.
Requirements	The actor is on the Dashboard page.
Pre-Condition	The actor is signed in.
Post-Condition	The actor has promoted the account to film club member status.

4.1.21 Administrator Adds to Videos Page**Use Case 4.1-21** Administrator Adds to Videos Page

Function	Administrator Adds Video to Videos Page
Description	The system shall allow admins to embed the club's videos, project, and films to allow users of the system to easily view them.
Inputs	URL of YouTube video (The YouTube ID Number) Embed Button
Actor(s)	Administrator YouTube
Source	Actors
Action	Actor will go to the videos page. On the video page the actor will have to provide at least the YouTube ID and Video Description, and possibly Tags. Actor will then submit the video.
Requirements	Actor must be on the videos page.
Pre-Condition	Actor must be signed-in.
Post-Condition	Video will now be visible on the video page for all users to view.

4.1.22 Administrator Adds to Images Page**Use Case 4.1-22** Administrator Adds to Images Page

Function	Allow admin to embed pictures into the pictures page.
Description	The system shall allow admin of the GSU Film Club to embed pictures to allow users of the system to easily view them.
Inputs	Image file Client file explorer dialog
Actor(s)	Administrator
Source	Actor.
Action	Actor will go to the pictures page. On the pictures page members will have to click the add picture button where they will provide the image file to embed the picture into the pictures page.
Requirements	Actor must be on the pictures page.
Pre-Condition	Actor must be signed-in as a member.
Post-Condition	Added pictures will now be visible on the pictures page, for all users to view.

4.1.23 User Makes Request or Suggestion to the Organization**Use Case 4.1-23** System User Makes Request or Suggestion to the Organization

Function	Make requests/suggestions for the GSU Film Club
Description	Shall allow a user to place a request/suggestion, for a film project or something else, to the GSU Film Club.
Inputs	Contact information: <ul style="list-style-type: none"> • first name • last name • email address • phone number Suggestion information: <ul style="list-style-type: none"> • subject • body of text for the suggestion
Actor(s)	Any User of the System
Source	Actor
Action	Actor will go to the suggestions page. A form will then appear prompting the actor to provide their contact information and their suggestion. Actor must then submit the suggestion, which will then be sent to the GSU Film Club for an administrator to review.
Requirements	Actor must be on the suggestions page
Pre-Condition	N/A.
Post-Condition	Suggestion will be sent to GSU Film Club to be evaluated.

4.1.24 User Changes Personal Look and Feel of the System****Use Case 4.1-24** User Changes Personal Look and Feel of the System

Function	User Changes the Template/Style of Web System.
Description	Should allow the changing of the template/style of the web system dynamically.
Inputs	Template/Style Selection Box
Actor(s)	Account Holder
Source	Actor
Action	Actor will only be allowed to click on the change template/style button. On clicking the button, actor will then be prompted to provide the template/style selection they desire. The system's template/style will now be changed to the specified template/style.
Requirements	Actor must hold an account with the system.
Pre-Condition	User must be signed-in.
Post-Condition	The template/style of the web system will now be set to the specified template/style.

4.1.25 User Profile Modification****Use Case 4.1-25** User Profile Modification

Function	Account Holder Edits Personal Profile
Description	Shall allow account holders within the system to edit their own personal profile upon registering.
Inputs	(Text fields) Name, phone #, interests, hobbies, age, grade.
Actor(s)	Account Holder
Source	Actor.
Action	Actor upon logging in or registering, actor will click next to where it says "Logged in as:" and select the drop down arrow. By pressing "Your Profile," the actor will be directed to their own personal account and can thus press edit to change existing information or add new information.
Requirements	Actor must hold an account with the system.
Pre-Condition	Actor is logged in and has pressed the edit button on their profile page.
Post-Condition	The new information is saved to their profile.

4.1.26 Contact Information Retrieval**Use Case 4.1-26** Retrieve Contact Information

Function	Find out contact information for the GSU Film club.
Description	Shall allow visitors or users of the GSU Film Club website to access and view contact information.
Inputs	Text-fields: <ul style="list-style-type: none"> • Name • phone number • interests • hobbies • age • grade
Actor(s)	Visitor User Admin
Source	Actor.
Action	Actor will click on the "About" button at the top navigation bar. A drop down menu will then be displayed. The actor will click on the "Contact" selection. This will re navigate them to the "Contact" page.
Requirements	N/A.
Pre-Condition	Actor is on the website.
Post-Condition	The contact information is displayed.

4.1.27 Contact Information Modification**Use Case 4.1-27** Modify Contact Information

Function	An Administrator Edits Contact Information
Description	The system shall allow an admin of the GSU Film club to change the information on the contact page if need be.
Inputs	Address Email
Actor(s)	Admin
Source	Actor
Action	The Actor will click the Site Information button, to go to the site information page. They will then see the "Contact Us" dialog with the current contact information displayed. The actor will write in the current contact information in the body text field. The actor will then click the Submit Changes button and update the contact information.
Requirements	N/A.
Pre-Condition	Actor is logged in as an admin.
Post-Condition	The new contact information is displayed on the "Contact" page.

4.1.28 Registration Failure**Use Case 4.1-28 Failure to Register**

Function	Failing to register as a member
Description	Shall happen when the user fails to enter proper information when registering as a member.
Inputs	Text-fields: <ul style="list-style-type: none"> • Username • Email • Password • Password Confirmation
Actor(s)	Visitor
Source	Actor
Action	If Username and Email are already in the system, then actor is prompted to enter different Username and Email until there is no conflict with other members of the system. If password is not of the minimum length then actor will be prompted to enter another password until minimum length is met. If actor enters any other password then the one they specified in Password Confirmation, they will be prompted to re-enter the Password Confirmation until it matches initial password entered.
Requirements	Actor enters invalid Username, Email, Password, and/or Password Confirmation
Pre-Condition	N/A.
Post-Condition	Actor enters new information after being prompted to change their initial information.

4.1.29 Add User Roles

Use Case 4.1-29 Add User Roles

Function	An Administrator Adds a new User Role
Description	Shall allow an admin of the GSU Film club to add an individual role to the list of roles.
Inputs	Text-fields: <ul style="list-style-type: none">• Role Name• Role Description
Actor(s)	Admin
Source	Actor
Action	The Actor will visit the "Manage User Roles" on the left of the dashboard window. The Actor will now enter the Role Name and Role Description, and then click "Submit."
Requirements	User is logged into an administrator account.
Pre-Condition	Actor is logged in as an admin and on the Dashboard Page
Post-Condition	The new role will be listed under "Current Roles."

4.1.30 Manage User Roles**Use Case 4.1-30** Modify Contact Information

Function	An Administrator can assign roles to individual club members.
Description	The system shall allow an admin of the GSU Film club to manage the roles of the members of the GSU Film Club.
Inputs	Set New Role
Actor(s)	Admin
Source	Actor
Action	The Actor will navigate to the "Manage Roles of Current Members" panel. From there, the Admin will find the user that they would like to assign a role. Under "New Role" they will click the drop down arrow and assign the particular role to the user. The Admin will then click "Set."
Requirements	User is an Admin.
Pre-Condition	Actor is logged in as an admin and on the Dashboard page.
Post-Condition	The User now has an updated role.

****** Denotes that the feature has not yet been implemented (NYI).

4.2 Explanations of Diagrams for Implemented Use Cases

4.2.1 Basic Website Navigation Subsystem (Figure 6.2.1-1, Appendix)

- An actor (any person viewing the application) is able to traverse the system via the navigation bar, submit a suggestion via the suggestion submission page, and follow the organizations link to an external social media outlet.

4.2.2 Login Subsystem (Figure 6.2.2-1, Appendix)

- A visitor may register an account to log into the system.
- An account holder with the system may sign in and sign out to end a session with the application.
- A club member (an account holder with elevated privileges) is able to flag other account holders as film club members.

4.2.3 User Profile Subsystem (Figure 6.2.3-1, Appendix)

- An account holder may add, change, or remove their profile image for their account.

4.2.4 Administrative Tools Subsystem (Figure 6.2.4-1, Appendix)

- An administrator may:
 - Add, change, and remove custom pages.
 - Read messages from the suggestions system.
 - Promote, demote, and suspend account holders of the system.
 - Add videos that will be displayed on the public videos page.
 - Add images that will be displayed on the public pictures page.

4.2.5 Discussion Subsystem (Figure 6.2.5-1, Appendix)

- Any application user (including the general public) may view discussion posts posted by account holders of the system (On the main discussion board page or custom page discussions).
- An account holder of the system in addition to viewing may create and reply to other posts.

4.2.6 Calendar of Events Subsystem (Figure 6.2.6-1, Appendix)

- Any application user, account holder, club member, and administrator may view events and navigate the calendar.

- Club members and administrators may add and remove events from the calendar.

4.3 Requirements Analysis

Contained here are brief explanations of the varied subsystems and their requirements within the whole system. Refer to Appendix B for the diagrammatic representations of these subsystems.

4.3.1 Data Requirements

The Data Requirements diagram in *Appendix A Figure 6.1-1* outlines the necessary data in order to implement features articulated by the requirements.

4.3.2 Website Navigation Subsystem

Figure 6.2.1-1 shows the features as a set of use cases that are integral to the Website Navigation Subsystem.

Figure 6.2.1-2 shows the flow of logic between the user and the system in order to communicate suggestions to the administrators.

4.3.3 Login Subsystem

Figure 6.2.2-1 illustrates the functional requirements of the Login Subsystem.

Figure 6.2.2-2 shows the sequence of events that take place between the user and components of the system during the account registration process.

4.3.4 User Profile Subsystem

Figure 6.2.3-1 lays out the features of the User Profile Subsystem that are to be utilized by account holders.

Figure 6.2.3-2 illustrates the steps taken by the parts of the system in order to delete a user profile image.

4.3.5 Administrative Tools Subsystem

Figure 6.2.4-1 shows the tasks an administrator may complete.

Figure 6.2.4-2 shows the series of actions that take place within the system when an administrator wants to view messages.

4.3.6 Discussion Subsystem

Figure 6.2.5-1 shows the requirements of the Discussion Message Subsystem.

Figure 6.2.5-2 shows the sequence of events that occur within the system when any user of the system wants to view a discussion.

4.3.7 Calendar of Events Subsystem

Figure 6.2.6-1 expresses the necessary features the Calendar of Events Subsystem should provide.

Figure 6.2.6-2 displays the steps taken by the components within the system when any person viewing the system navigates the calendar.

5. Software Architecture

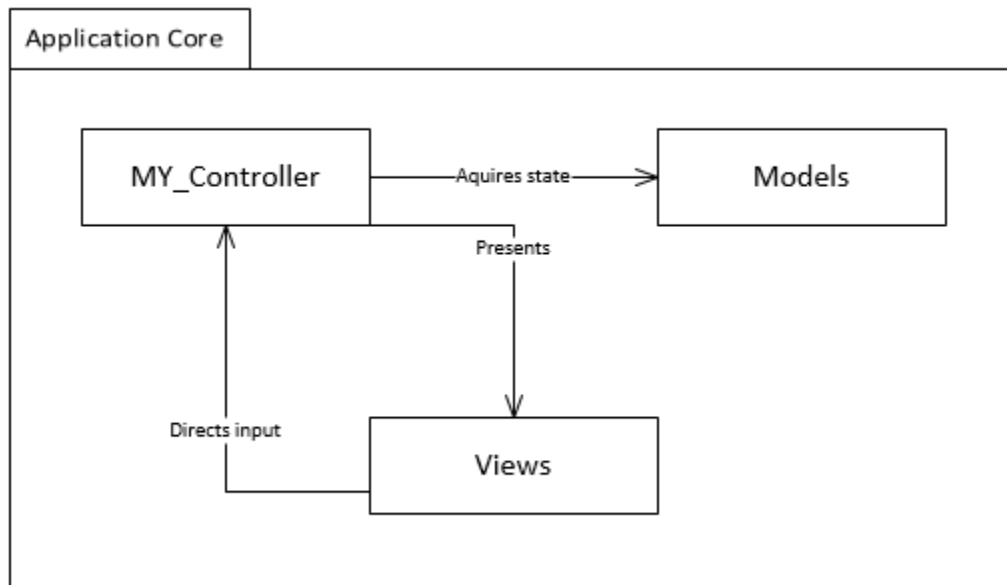
The system utilizes two software architectural patterns. The overall goal of these patterns is to adhere with the Open-Closed Principle of the object-oriented programming paradigm. The Open-Closed Principle states that “*software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.*” (Meyer, B.)

One of the patterns is called the *Front Controller* pattern. This pattern is commonly used in web applications and the frameworks on which they are built. The goal is for the application to have a main entry point (in the case of the Film Club CMS, the “index.php” in the application’s root file directory) for which all requests funnel through. This encourages a modular implementation of the underlying structure of the application.

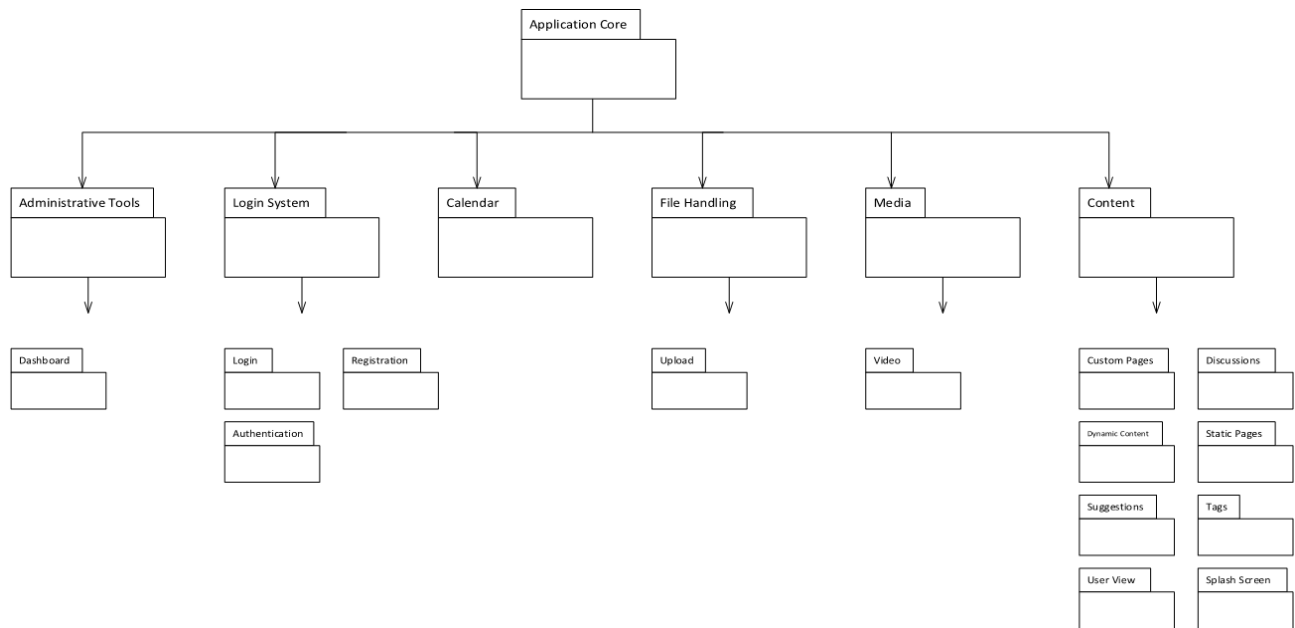
The next pattern the Film Club CMS utilizes is the *Hierarchical Model-View-Controller* (HMVC) software architectural pattern. The goal of this pattern is to divide the structure of the classes in the application into logical widgets in order to decouple the subsystems and have the subsystems operate independently of each other. The end result is that features may be modified and added without the need to modify existing widgets.

5.1 Overview

Below is the package diagram outlining a (hierarchical) Model-View-Controller (HMVC) software architecture pattern between the base controller class (MY_Controller) and models and views within the system. Any model and view may utilize the attributes and functions of the base controller.



At the application's core are the top-level MVC components. This package contains the application's super controller (MY_Controller) that stores data to be passed to a view (An abstracted operation common to all controllers). The view may direct input to the controller where it may process the desired operations. The data stored by the controller may include stateful information stored in the database. This is done through interaction with the model classes.



The above diagram shows the hierarchy packages of the application in their entirety. The two architectural patterns are clearly here visible here. The application core contains the Front Controller (MY_Controller) as part of the Front Controller pattern. The Hierarchical MVC pattern is directly illustrated by the hierarchy of packages containing more specialized controller classes.

For the purposes of this diagram each second-level package represents a subsystem. Each package represents a set of MVC components. A brief description of each subsystem follows.

5.2 Subsystem Decomposition

Below is a mapping of the system modules with the specific use cases.

5.2.1 Administrative Tools

This is responsible for dashboard activities such as creating new content and user management.

Encompasses use cases (UCID):

- UC#4.1-15
- UC#4.1-16
- UC#4.1-17
- UC#4.1-18
- UC#4.1-19
- UC#4.1-20
- UC#4.1-21
- UC#4.1-22

5.2.2 Login System

This handles the authentication and registration processes. Encompasses use cases (UCID):

- UC#4.1-2
- UC#4.1-3
- UC#4.1-4

5.2.3 Calendar System

This is responsible for the viewing and creation of events on the calendar. Encompasses use cases (UCID):

- UC#4.1-11
- UC#4.1-13
- UC#4.1-14

5.2.4 File Handling System

This manages the storage of files on the server. Encompasses use cases (UCID):

- UC#4.1-6
- UC#4.1-7
- UC#4.1-8

5.2.5 Media System

Responsible for the presentation, creation, and organization of club videos. Encompasses use cases (UCID):

- UC#4.1-21
- UC#4.1-22

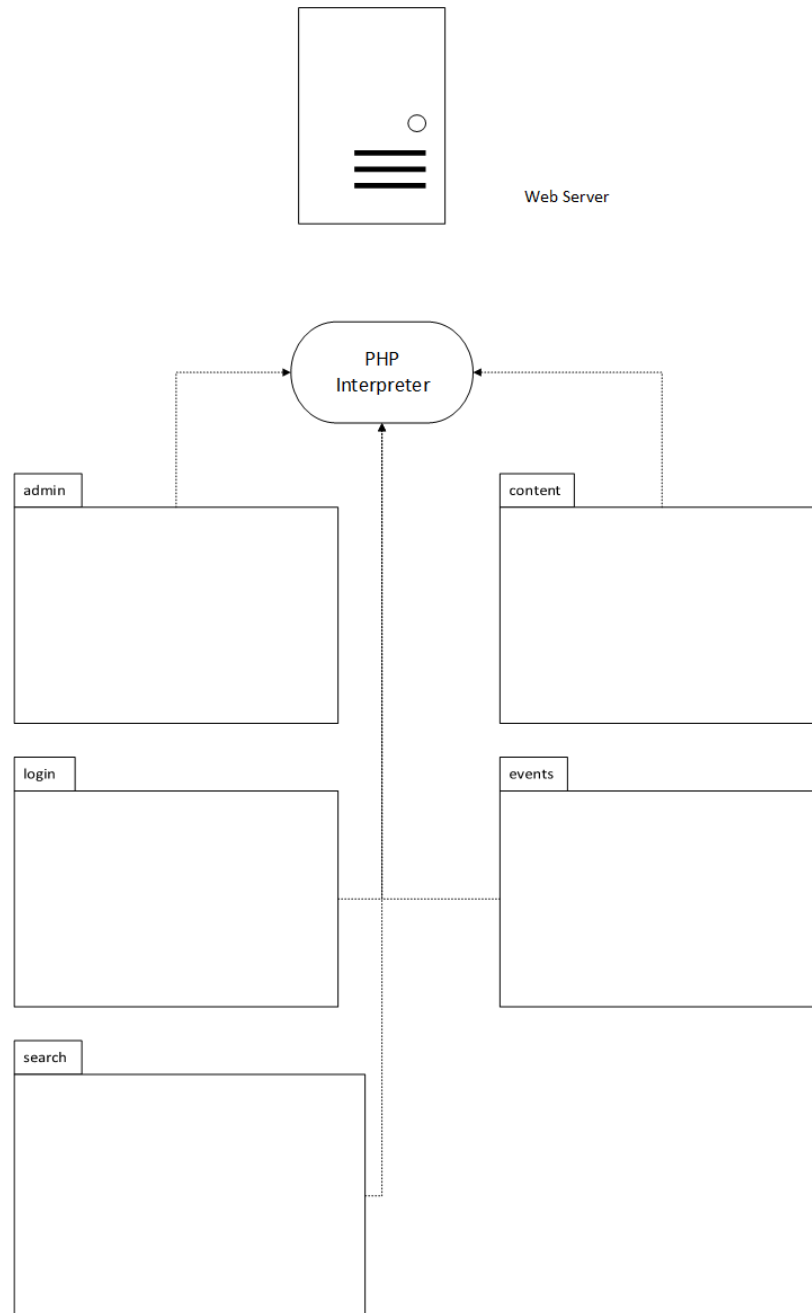
5.2.6 Content

The largest subsystem. It is responsible for the management of dynamic content. Encompasses use cases (UCID):

- UC#4.1-26
- UC#4.1-27
- UC#4.1-29
- UC#4.1-30

5.3 Hardware and Software Mapping

The Film Club CMS runs completely on a server machine that has the required software stack installed (See hardware and software requirements).



5.4 Persistent Data Management

The persistent data required by the application is provided by a MySQL database management system server located on the host machine. As typical with most database management systems, the data is organized into tables. Other schema objects include database views, stored procedures, and triggers.

The goal of the database views is to provide a simpler interface for the application domain to interact with the database system. For example the “custom_pages_with_authors” removes a need for the PHP application code to have knowledge of the relationships and joins required of the tables.

The stored procedures and triggers provide similar benefits in that the PHP application code need not manage all of the individual transactions. That responsibility is deferred to the MySQL database system to where the transactions are done in a simpler, safer way.

The next page contains the schema definition of the Film Club CMS in the form of Entity-Relationship diagrams:

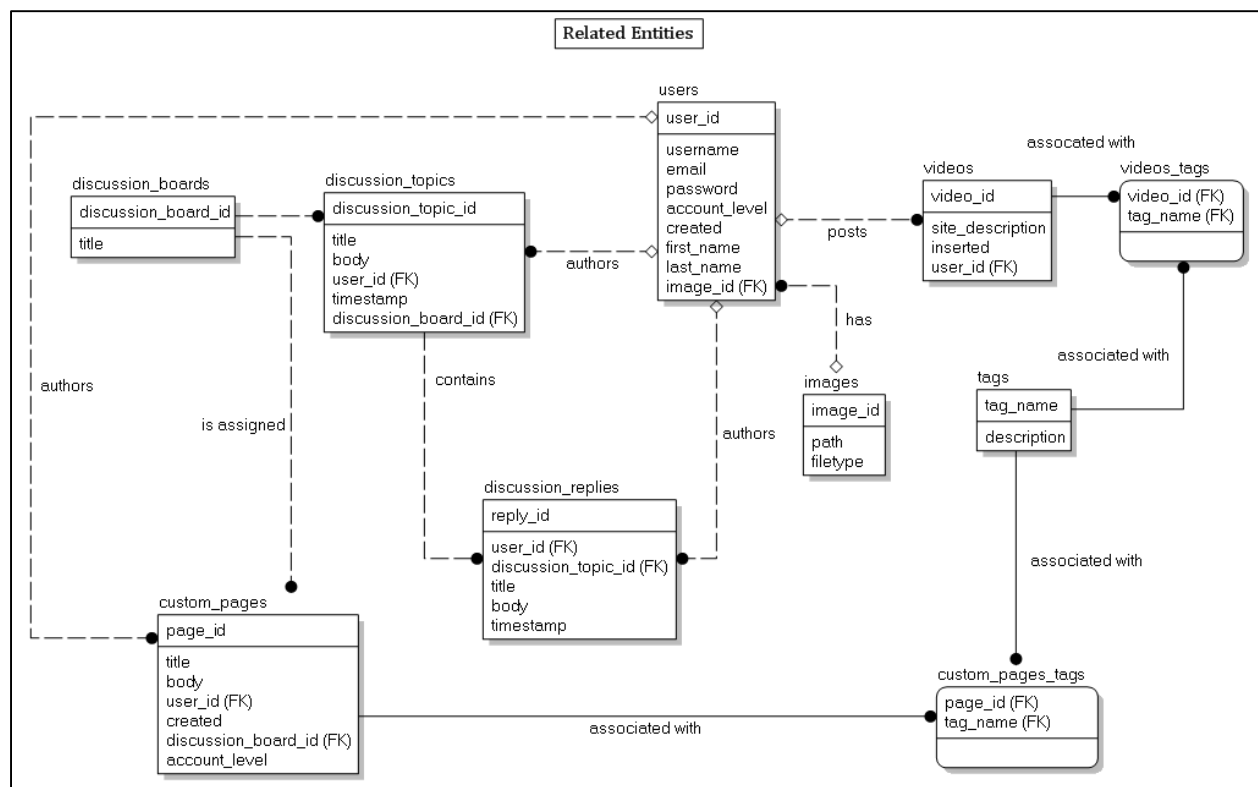
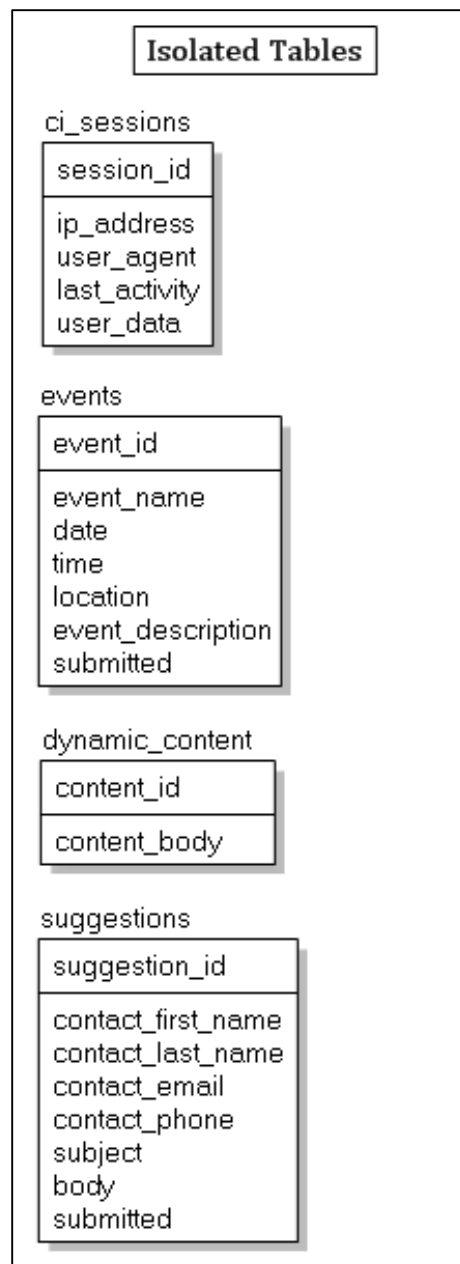
Figure 5.4.1 Entity Relationship Diagram of Related Entities of the Film Club CMS

Figure 5.4.2 Diagram of Isolated Storage Tables within the Database

These tables are required only for their storage capabilities for simple persistent data needs. The “ci_sessions” table saves session information for users after the application has halted on the client end. The “events” table stores the necessary information for the events subsystem. The “dynamic_content” table allows for simple storage of the content of the information pages within the website. The “suggestions” table handles the storage for the suggestion submission system. Because there are no membership requirements, the table does not require any foreign key constraints.



5.4.1 Data Dictionary

Below are the database tables and views used by the Film Club CMS.

ci_sessions

This table is used to store sessions for the application.

Column	Type	Null	Default	Comments
session_id	varchar(40)	No	o	
ip_address	varchar(45)	No	o	
user_agent	varchar(120)	No		
last_activity	int(10)	No	o	
user_data	text	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	session_id	7	A	No	
last_activity_idx	BTREE	No	No	last_activity	7	A	No	

custom_pages

Stores the records for the custom pages. Each custom page is assigned a discussion board via an pre-insertion trigger.

Column	Type	Null	Default	Comments
page_id	int(11)	No		
title	varchar(100)	No		
body	text	No		
user_id	int(11)	Yes	NULL	
created	timestamp	Yes	CURRENT_TIMESTAMP	
discussion_id	int(11)	Yes	NULL	
discussion_board_id	int(11)	No		
account_level	tinyint(4)	Yes	-1	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	page_id	11	A	No	
user_id	BTREE	No	No	user_id	11	A	Yes	
custom_pages_ibfk_2	BTREE	No	No	discussion_id	11	A	Yes	
discussion_board_id	BTREE	No	No	discussion_board_id	11	A	No	

custom_pages_comments_authors

View that joins the custom pages, discussion board, and users table for use by the application.

Table comments: VIEW

Column	Type	Null	Default	Comments
page_id	int(11)	No	0	
title	varchar(100)	No		
body	text	No		
user_id	int(11)	No	0	
username	varchar(20)	No		
created	timestamp	Yes	NULL	
discussion_board_id	int(11)	No		
comments	bigint(21)	No	0	

custom_pages_tags

Emcompasses the many-to-many relationship of custom pages to tags.

Column	Type	Null	Default	Comments
page_id	int(11)	No		
tag_name	varchar(30)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	page_id	1	A	No	
				tag_name	1	A	No	
tag_name	BTREE	No	No	tag_name	1	A	No	

discussion_boards

This table defines a discussion board record. Discussion boards are intended to be associated with discussion topics.

Column	Type	Null	Default	Comments
discussion_board_id	int(11)	No		
title	varchar(50)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	discussion_board_id	14	A	No	

discussion_boards_number_of_comments

This view joins the discussion_boards table with the discussion_topics table to summarize the number of comments for the application to use.

Table comments: VIEW

Column	Type	Null	Default	Comments
discussion_board_id	int(11)	No	0	
number_of_comments	bigint(21)	No	0	

discussion_replies

This table stores the replies to discussion topics. Each discussion reply is associated with a single discussion topic.

Column	Type	Null	Default	Comments
reply_id	int(11)	No		
user_id	int(11)	Yes	<i>NULL</i>	
discussion_topic_id	int(11)	No		
title	varchar(100)	Yes	<i>NULL</i>	
body	text	No		
timestamp	timestamp	No	CURRENT_TIMESTAMP	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	reply_id	30	A	No	
user_id	BTREE	No	No	user_id	10	A	Yes	
discussion_topic_id	BTREE	No	No	discussion_topic_id	30	A	No	

discussion_replies_with_authors

This view joins the discussion_replies table with the authors table for the application to use.

Table comments: VIEW

Column	Type	Null	Default	Comments
body	text	No		
discussion_topic_id	int(11)	No		
reply_id	int(11)	No	o	
timestamp	timestamp	No	0000-00-00 00:00:00	
username	varchar(20)	Yes	NULL	

discussion_topics

This table contains the records for discussion topics. Discussion topics belong to a discussion board.

Column	Type	Null	Default	Comments
discussion_topic_id	int(11)	No		
title	varchar(100)	No		
body	text	No		
user_id	int(11)	Yes	NULL	
timestamp	timestamp	No	CURRENT_TIMESTAMP	
discussion_board_id	int(11)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	discussion_topic_id	18	A	No	
user_id	BTREE	No	No	user_id	4	A	Yes	
discussion_board_id	BTREE	No	No	discussion_board_id	18	A	No	

discussion_topics_with_authors

This view joins the users and discussion_topics table for use by the application.

Table comments: VIEW

Column	Type	Null	Default	Comments
body	text	No		
title	varchar(100)	No		
discussion_topic_id	int(11)	No	0	
discussion_board_id	int(11)	No		
timestamp	timestamp	No	0000-00-00 00:00:00	
username	varchar(20)	Yes	NULL	

discussions_for_custom_page

This joins the discussions and custom_pages tables.

Table comments: VIEW

Column	Type	Null	Default	Comments
page_id	int(11)	No	0	
discussion_id	int(11)	No	0	
reply_id	int(11)	Yes	NULL	
title	varchar(100)	No		
body	text	No		
user_id	int(11)	Yes	NULL	
timestamp	timestamp	No	0000-00-00 00:00:00	

dynamic_content

This table simply stores content for a static page.

Column	Type	Null	Default	Comments
content_id	varchar(32)	No		
content_body	text	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	content_id	2	A	No	

events

This table stores the records for the event pages for use by the calendar.

Column	Type	Null	Default	Comments
event_id	int(11)	No		
event_name	varchar(50)	No		
date	date	No		
time	time	No		
location	varchar(50)	No		
event_description	varchar(1000)	No		
submitted	timestamp	No	CURRENT_TIMESTAMP	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	event_id	13	A	No	

images

This stores the name (the primary key ID), path, and file type of images.

Column	Type	Null	Default	Comments
image_id	int(11)	No		
path	varchar(256)	No		
filetype	varchar(10)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	image_id	21	A	No	

non_user_images

This view does an anti-join of the users and images tables to display images not assigned to a user.

Table comments: VIEW

Column	Type	Null	Default	Comments
image_id	int(11)	No	0	
filetype	varchar(10)	No		
path	varchar(256)	No		

roles

This table stores the roles records.

Column	Type	Null	Default	Comments
role_id	int(11)	No		
role_name	varchar(50)	No		
role_description	varchar(256)	Yes	<i>NULL</i>	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	role_id	4	A	No	

Suggestions

This table stores the suggestion records for the suggestion message system.

Column	Type	Null	Default	Comments
suggestion_id	int(11)	No		
contact_first_name	varchar(30)	No		
contact_last_name	varchar(30)	No		
contact_email	varchar(50)	No		
contact_phone	varchar(20)	No		
subject	varchar(50)	No		
body	varchar(5000)	No		
submitted	timestamp	No	CURRENT_TIMESTAMP	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	suggestion_id	7	A	No	

tags

This table stores the tag records.

Column	Type	Null	Default	Comments
tag_name	varchar(30)	No		
description	varchar(100)	Yes	<i>NULL</i>	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	tag_name	7	A	No	

unit_test_results

For developer use only. This table stores the results of unit tests.

Column	Type	Null	Default	Comments
test_id	int(11)	No		
test_name	varchar(30)	No		
test_datatype	varchar(30)	Yes	<i>NULL</i>	
expected_datatype	varchar(30)	Yes	<i>NULL</i>	
result	varchar(10)	Yes	<i>NULL</i>	
file_name	varchar(100)	Yes	<i>NULL</i>	
line_number	int(11)	Yes	<i>NULL</i>	
notes	varchar(200)	Yes	<i>NULL</i>	
timestamp	timestamp	No	CURRENT_TIMESTAMP	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	test_id	2	A	No	

Users

This table stores the user information.

Column	Type	Null	Default	Comments
user_id	int(11)	No		
username	varchar(20)	No		
email	varchar(30)	No		
password	varchar(60)	No		
account_level	int(11)	Yes	0	
created	timestamp	Yes	CURRENT_TIMESTAMP	
first_name	varchar(50)	Yes	NULL	
last_name	varchar(50)	Yes	NULL	
image_id	int(11)	Yes	NULL	
role_id	int(11)	Yes	NULL	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	user_id	31	A	No	
email	BTREE	Yes	No	email	31	A	No	
fk_user_image_id	BTREE	No	No	image_id	6	A	Yes	

Videos

This table stores the video using the YouTube video ID number.

Column	Type	Null	Default	Comments
video_id	varchar(30)	No		
site_description	text	Yes	NULL	
inserted	timestamp	No	CURRENT_TIMESTAMP	
user_id	int(11)	Yes	NULL	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	video_id	8	A	No	
user_id	BTREE	No	No	user_id	8	A	Yes	

videos_tags

This table stores the many-to-many association between the videos and tags tables.

Column	Type	Null	Default	Comments
video_id	varchar(30)	No		
tag_name	varchar(30)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	video_id	6	A	No	
				tag_name	6	A	No	
tag_name	BTREE	No	No	tag_name	6	A	No	

5.5 Security/Privacy

One of the largest security concerns for the Film Club CMS is the storage of passwords. There will always be the possibility of a user selecting an insecure password, but the system aims to make it as difficult as possible for malicious users to access the raw passwords.

The user login authentication process is handled by database queries. Upon registration, the user chooses a text password that contains at least six characters. Upon validation of the content in the registration form, a database record is inserted and the user's password is stored in the database as a hash using the "bcrypt" hashing algorithm.

The "bcrypt" algorithm is a dynamic hash function based on the Blowfish block cipher cryptographic algorithm. Below is the method signature for PHP's "password_hash" function (it uses "bcrypt")

```
string password_hash( string $password, int $algorithm [, array $options] )
```

The \$password argument is the user's raw text password. The \$algorithm argument is the algorithm PHP should use to hash the password (currently the "bcrypt" algorithm by default). The \$options associate array contains two keys, "salt", and "cost". The "salt" key provides the algorithm a custom salt to use in the hash, but a random salt is generated when this field is left blank (PHP recommends against developers using their own salts). The "cost" key is the magic behind this algorithm. An increasing cost value increases the complexity of the algorithm by orders of magnitude. This is especially useful in preventing malicious users from using computational brute force tactics to crack the original password from the hash.

For example, a "cost" value of 10 will result in the algorithm taking 200 milliseconds to execute on the host machine. The "cost" value is the base-2 logarithm of N where N is the number of iterations executed within the algorithm. Thus, each increase of 1 in cost will result twice the number iterations the function performs which doubles the time needed to compute the result. This is beneficial to combat machines becoming more powerful and capable of cracking the hash. The parameter for the function call need only be increased over time.

The standard secure user authentication process would be as follows: The user sends their email address and password over a secure channel (HTTPS). A precompiled MySQL database query is then executed to find a user of the supplied email address. If a row in the database exists, the database returns the hash column for that user. This hash is then matched with the user input through the PHP function `password_verify($password, $hash)`. Correct login credentials will set a session variable called `logged_in` to true, and incorrect credentials will redirect the user back to the login page with an error message, “Invalid credentials.”

The “Invalid credentials” error message is shown for all login errors (such as email address and password mismatch or email address does not exist) to further obfuscate attackers efforts from learning sensitive details (such as confirming the existence of an email of a user by receiving a message, “That email address does not use that password”).

6. Detailed Design

This section will illustrate some of the module design principles (namely the package and class design for objects) used by the Film Club CMS. The design principles revolve around extending the principles implemented by the CodeIgniter MVC PHP framework.

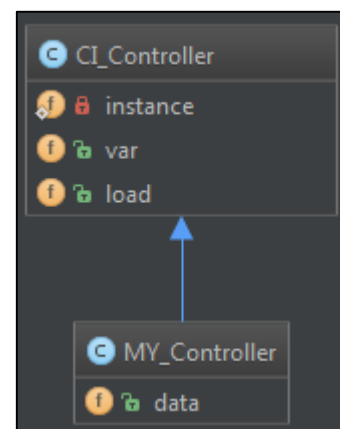
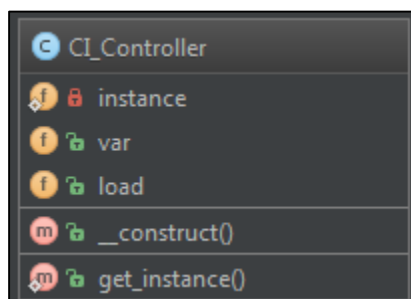
The interactions of objects involve the communication between the models, views, and controllers. Controller classes are responsible for validating input from view requests, and sending data to the models. The models are responsible for the communication between the database and the application and encompass the business logic so as to provide a layer of abstraction between the database and application for the benefit of the controllers. Views simply contain the HTML and inline PHP scripts for viewing a page to the user.

The result of the designs of the Film Club CMS result in a loose coupling and high cohesion so that new modules and features are easily added and that changes to classes have a minimal footprint to other classes of the system.

6.1 Overview

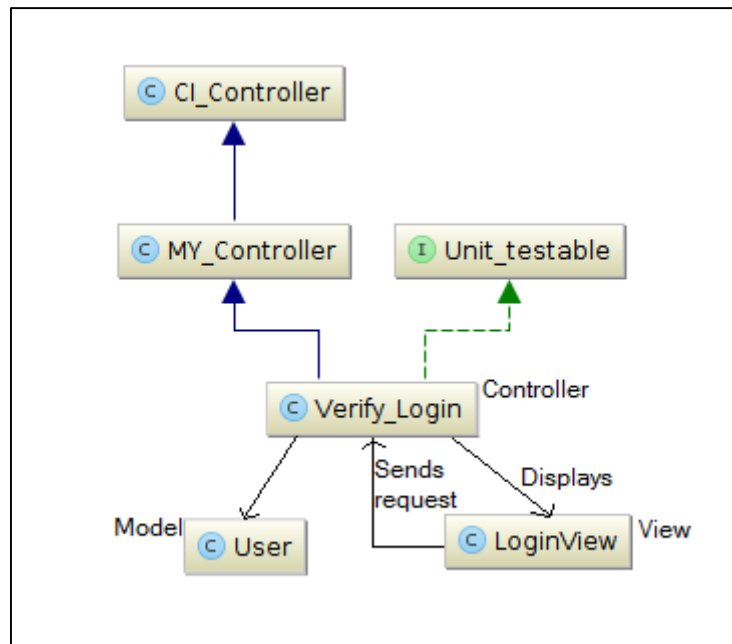
6.1.1 Singleton Pattern

Below is a minimal UML class diagram of the super controllers of the application. These are responsible for the flow of logic between the views and model classes. Notice the private static *instance* attribute and public static `get_instance()` function. This is an implementation of the singleton object-oriented design pattern and is how the system maintains a global state across the execution of the application.



6.1.2 Model-View-Controller (MVC) Pattern

Pictured below is an example usage of a **Model-View-Controller pattern**. The controller (Verify_Login) hears out requests and sends out a response and validates input for the View (LoginView). The controller alerts the model (Users) of changes in state that needs to be stored in the database.

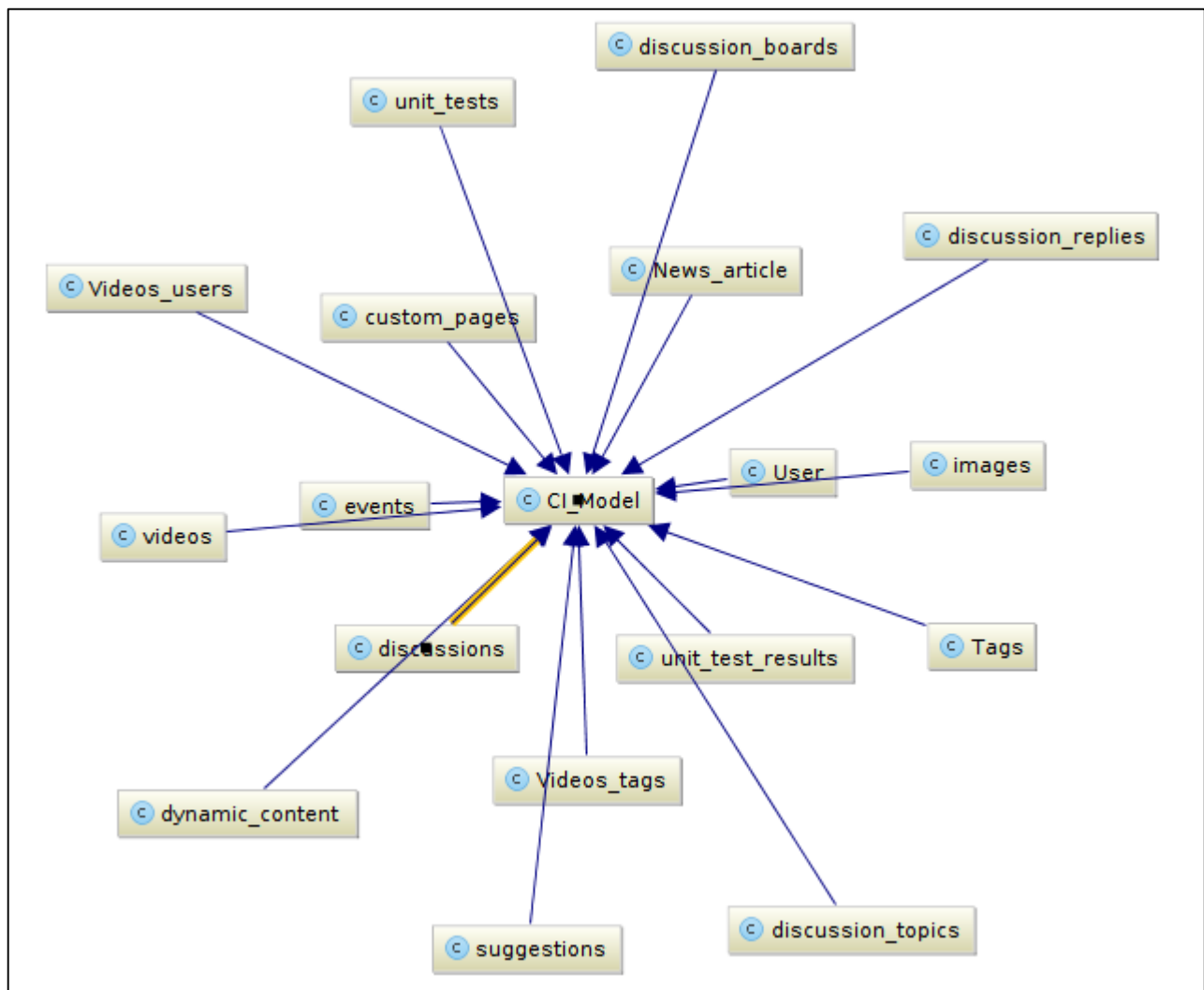


6.1.3 Active Record Pattern

The CodeIgniter PHP web application framework implements a variation of the active record pattern. The Film Club CMS extends this design to fit with the needs of the modules/classes of the system.

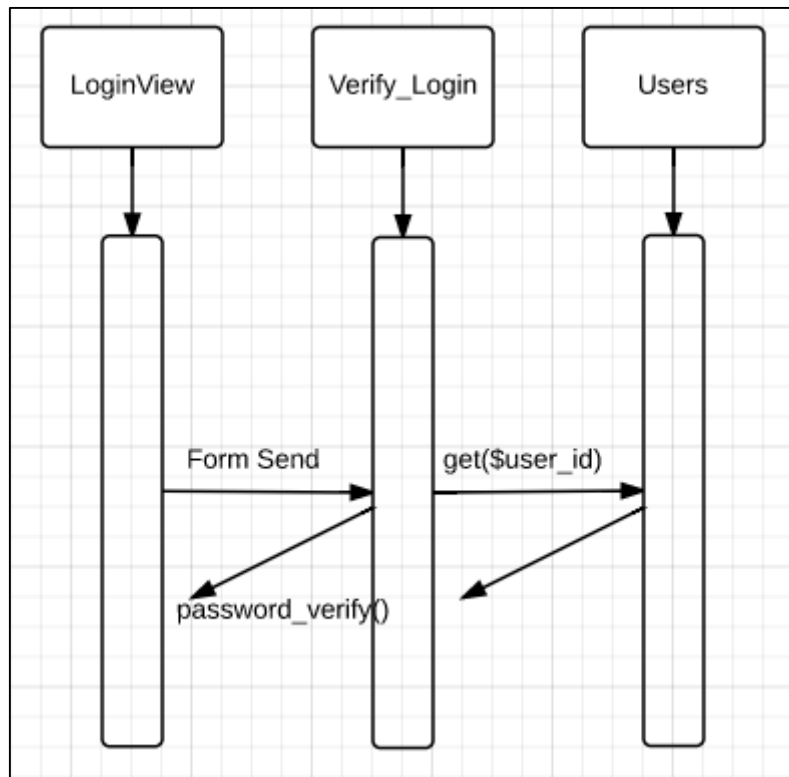
The active record pattern is a pattern that seeks to achieve storage of temporary object data as data stored in relational database management systems.

Pictured below is the Film Club's implementation of the active record pattern. Each model class is a representation of the table located within the MySQL database.

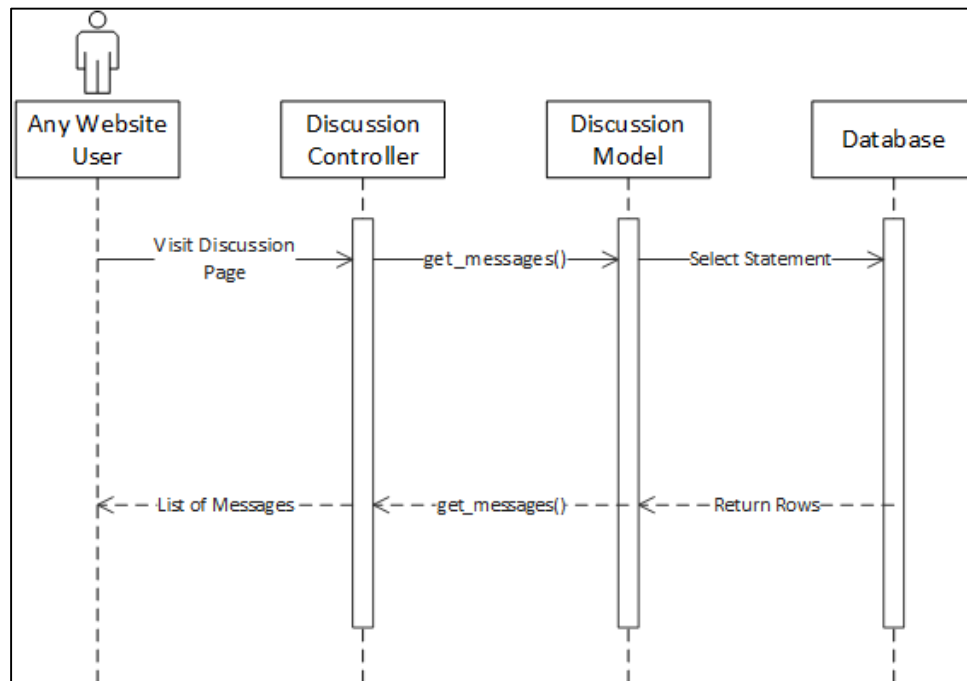


6.2 Object Interaction

Pictured below are examples of a **model-view-controller (MVC)** sequence within the execution of the Film Club CMS. Notice the name of the model class reflects the database table (**active record pattern**).



Sequence diagram of an actor posting a discussion to the discussion page.



6.3 Detailed Class Design

The purpose of each class referenced in the class diagrams in Appendix D can be found in the phpdoc.

7. Testing Process

This section aims to elaborate on the SQA Plan in place for the Georgia Southern Film Club CMS.

7.1 Software Quality Assurance Plan

7.1.1 Purpose

The SQA plan in place aims to guarantee the quality of the system. Using additional software testing tools, a detailed and complete plan will ensure an optimal system.

7.1.2 Reference Documents

7.1.3 SQA Team Organization

Lamar Sims: Test Team Leader

- Generate Test Plan and organize documentation.
- Assign test cases to test analyst.
- Conveying issues to development team.
 - Assisting development team with such issues.
- Test all functionality thoroughly.
 - This involves the selenium IDE testing tool.
- Assist test analyst.
- Maintaining and organizing all testing documentation.

Matthew Stratton: Test Team Analyst

- Assist the test team leader in generating a test plan and creating documentation.
- Define the testing requirements.
- Designing the format of the test cases in section 7.2.
- Carrying out test cases and testing for expected results.
 - This involves the selenium IDE testing tool.
- Relaying test findings to test team leader and development team.
 - Assisting development team with such issues.

7.1.4 Required Software and Hardware

- Selenium IDE (Firefox add-on)
- Mozilla Firefox

- MS Word 2010

7.2 System Tests

Each system test will aim to ensure the quality of each functional aspect of the Georgia Southern Film Club CMS. These tests will involve taking each aspect of what the system shall do and comparing the actual results with the expected results. This will ensure that the users of the GSU Film Club CMS do not experience any failures or unexpected behavior.

TC_001 – Registering a new User

Test 1:

- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-2**
- **Description:** Test to see if the system properly allows a new user to create an account.
- **Inputs for input fields:**
 - **'Username':** usertest20
 - **'Email':** ohioborn1991@yahoo.com
 - **'First Name':** Greg
 - **'Last Name':** Allman
 - **'Password':** greenCity
 - **'Confirm':** greenCity4
- **Test Steps:**
 1. From any page on the website, click on the **'Register'** button in the top right corner.
 2. Enter all user information as listed in the specified inputs above.
 3. Click **'Register'**.
- **Expected Output:** The registration page will re-open with a message indicating that the account has been created, and prompts the user to **'Sign In'**.

Test 2:

- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-2**
- **Description:** Test to see if the system properly detects an account that has been created with the same email or username.

- **Inputs for input fields:**
 - **'Username':** usertest20
 - **'Email':** ohioborn1991@yahoo.com
 - **'First Name':** Timmy
 - **'Last Name':** Turner
 - **'Password':** greenCity4
 - **'Confirm':** greenCity4
- **Test Steps:**
 1. From any page on the website, click on the **'Register'** button in the top right corner.
 2. Enter all user information as listed in the specified inputs above.
 3. Click **'Register'**.
- **Expected Output:** The registration page will re-open with a message indicating that an account exists with the same username or email.

Test 3:

- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-2**
- **Description:** Test to see if the system properly detects when user information is left out while registering, as all information is required.
- **Inputs for input fields:**
 - **'Username': (empty)**
 - **'Email':** email24@gmail.com
 - **'First Name':** Timmy
 - **'Last Name': (empty)**
 - **'Password':** greenCity4
 - **'Confirm':** greenCity4
- **Test Steps:**
 1. From any page on the website, click on the **'Register'** button in the top right corner.
 2. Enter all user information as listed in the specified inputs above.
 3. Click **'Register'**
- **Expected Output:** The registration page will re-open with a message indicating what fields were left out. In this case, **Username** and **Last Name**

TC_002 – User Login**Test 1:**

- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-3**
- **Description:** Test to see if the system allows users to log in to their created accounts upon entering the correct information.
- **Inputs for input fields:**
 - **'Email':** ohioborn1991@yahoo.com
 - **'Password':** greenCity4
- **Test Steps:**
 1. From any page on the website, navigate the cursor to the textbox that says **'Email'** in the top right of the webpage.
 2. Enter the **'Email'** information as listed in the specified inputs above.
 3. Tab over to the **'Password'** textbox and enter the **'Password'** information as listed in the specified inputs above.
 4. Click **'Sign In'**.
- **Expected Output:** The homepage will reload and you will see a 'Logged in as: **'Username'** in the top right. The user has successfully logged in.

Test 2:

- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-3**
- **Description:** Test to see if the system catches when an incorrect password has been entered for a user account.
- **Inputs for input fields:**
 - **'Email':** ohioborn1991@yahoo.com
 - **'Password':** notcorrect2
- **Test Steps:**
 1. From any page on the website, navigate the cursor to the textbox that says **'Email'** in the top right of the webpage.
 2. Enter the **'Email'** information as listed in the specified inputs above.
 3. Tab over to the **'Password'** textbox and enter the **'Password'** information as listed in the specified inputs above.

4. Click 'Sign In'.

- **Expected Output:** An incorrect password or email notification appears

Test 3:

- **Pre-Requirements:** The user must be on the website.
- **Use Case Reference #4.1-3**
- **Description:** Test to see if the system catches when login information is left blank.
- **Inputs for input fields:**
 - **'Email': (empty)**
 - **'Password': (empty)**
- **Test Steps:**
 1. From any page on the website, navigate to the button in the top right corner of the web page, **'Register'**.
 2. Leave both the **'Username'** and **'Password'** text boxes empty and click **'Sign In'**.
- **Expected Output:** Prompt the user to enter a email and password.

TC_003 – User Log Out**Test 1:**

- **Pre-Requirements:** The user must be on the website and currently logged in.
- **Use Case Reference #4.1-4**
- **Description:** Test to see if the system properly logs out a currently logged in user.
- **Inputs for input fields:** (NONE)
- **Test Steps:**
 1. Navigate to the top right of the webpage and find the option to **'Sign Out'**.
 2. Click **'Sign Out'**.
- **Expected Output:** The webpage is redirected to the home page, and the fields as referenced in Test Case **TC_002** are visible, indicating no user is logged in.

TC_004 – Social Media Navigation**Test 1:**

- **Pre-Requirements:** The user must be on the website.
- **Use Case Reference #4.1-5**
- **Description:** Test to see if the system properly directs the user to the external YouTube page as related to the GSU Film Club.
- **Inputs for input fields:** (NONE)
- **Test Steps:**
 1. Scroll to the bottom of the webpage.
 2. Find the social media buttons under '**Follow Us**'.
 3. Click the first one that contains the YouTube logo.
- **Expected Output:** The web page will be redirected to the GSU Film Club's YouTube channel.

Test 2:

- **Pre-Requirements:** The user must be on the website.
- **Use Case Reference #4.1-5**
- **Description:** Test to see if the system properly directs the user to the external Twitter page as related to the GSU Film Club.
- **Inputs for input fields:** (NONE)
- **Test Steps:**
 1. Scroll to the bottom of the webpage.
 2. Find the social media buttons under '**Follow Us**'.
 3. Click the second one that contains the Twitter logo.
- **Expected Output:** The web page will be redirected to the GSU Film Club's Twitter page.

Test 3:

- **Pre-Requirements:** The user must be on the website.
- **Use Case Reference #4.1-5**
- **Description:** Test to see if the system properly directs the user to the external Facebook page as related to the GSU Film Club.
- **Inputs for input fields:** (NONE)
- **Test Steps:**

1. Scroll to the bottom of the webpage.
2. Find the social media buttons under '**Follow Us**'.
3. Click the third one that contains the Facebook logo.
- **Expected Output:** The web page will be redirected to the GSU Film Club's Facebook page.

TC_005 – Profile Image Creation & Replacement

Test 1:

- **Pre-Requirements:** The user must be on the website and logged on.
- **Use Case Reference #4.1-6**
- **Description:** Test to see if the system allows a user to upload an image to use as their personal profile image.
- **Inputs for input fields:**
 - '**Choose a File**': '**Browse**' to select an image saved somewhere on the computer
- **Test Steps:**
 1. Navigate to the top right of the web page where it says 'Logged in as: '**Username**', and click on the drop down arrow
 2. Choose the first option, '**Your Profile**'.
 3. Scroll down to the panel listed as 'Manage Profile Picture'.
 4. Under 'Add or Edit Profile Picture', click '**Browse**'.
 5. Navigate to where the desired png, jpg, or gif file is stored on the computer.
 6. Press '**Open**' and then '**Upload**'.
- **Expected Results:** The users profile page will re-load and the new profile picture will be visible.

Test 2:

- **Pre-Requirements:** The user must be logged on and already has specified a profile image.
- **Use Case Reference #4.1-7**
- **Description:** Test to see if the system allows a users profile image to be replaced with a new one.
- **Inputs for input fields:**
 - '**Choose a File**': '**Browse**' to select an image saved somewhere on the computer

- **Test Steps:**

1. Navigate to the top right of the web page where it says 'Logged in as: **'Username'**', and click on the drop down arrow.
 2. Choose the first option, **'Your Profile'**.
 3. Scroll down to the panel listed as 'Manage Profile Picture'.
 4. Under 'Add or Edit Profile Picture', click **'Browse'**.
 5. Navigate to where the desired png, jpg, or gif file is stored on the computer.
 6. Press **'Open'** and then **'Upload'**.
- **Expected Results:** The users profile page will re-load and the updated profile picture will be visible.

Test 3:

- **Pre-Requirements:** The user must be on the website and logged on.
- **Use Case Reference #4.1-6 & #4.1-7**
- **Description:** Test to see if the system does not allow for non image type files to be used as a profile picture.
- **Inputs for input fields:**
 - **'Choose a File': 'Browse'** to select a non image type file stored somewhere on the computer.
- **Test Steps:**
 1. Navigate to the top right of the web page where it says 'Logged in as: **'Username'**', and click on the drop down arrow.
 2. Choose the first option, **'Your Profile'**.
 3. Scroll down to the panel listed as 'Manage Profile Picture'.
 4. Under 'Add or Edit Profile Picture', click **'Browse'**.
 5. Navigate to where a non image file such as a docx, txt, mp3, is stored on the computer.
 6. Press **'Open'** and then **'Upload'**.
- **Expected Results:** The users profile page will re-load with an error regarding the file format of the image selected.

TC_006 – Profile Image Deletion**Test 1:**

- **Pre-Requirements:** The user must be on the website and logged in & must have a profile picture for their personal account.
- **Use Case Reference #4.1-8**
- **Description:** Test to see if the system properly allows a user to remove their pre-existing profile picture from their account.
- **Inputs for input fields: (none)**
- **Test Steps:**
 1. Navigate to the '**Your Profile**' page. (See steps 1 - 2 in TC_005, Test 1).
 2. In the manage profile picture panel, look for 'Delete Profile Picture'
 3. Click '**Delete**'
- **Expected Results:** The users profile picture will be removed and replaced with the default empty profile picture view.

Test 2:

- **Pre-Requirements:** The user must be on the website and logged in & must not have a profile picture for their personal account.
- **Use Case Reference #4.1-8**
- **Description:** Test to see if the system properly allows catches when a user tries to delete a profile picture from their account when one does not exist
- **Inputs for input fields: (none)**
- **Test Steps:**
 1. Navigate to the '**Your Profile**' page. (See steps 1 - 2 in TC_006, Test 1).
 2. In the manage profile picture panel, look for 'Delete Profile Picture'.
 3. Click '**Delete**'.
- **Expected Results:** The system will display an error saying that a profile picture does not exist.

TC_007 – Reply to Discussion Thread**Test 1:**

- **Pre-Requirements:** The user must be on the Discussion page, and has at least Club Member account level.
- **Use Case Reference #4.1-10**
- **Description:** Test to see if the system allows a user to respond to a currently existing discussion thread.
- **Inputs for input fields:**
 - Desired reply text in the '**Message**' text box.
- **Test Steps:**
 1. On the discussion page, find a currently existing thread to reply to
 2. On the right side of the thread, click the button titled '**Reply**'.
 3. Fill in desired reply in the text area of the '**Message**' text box.
 4. Click '**Submit**'.
- **Expected Results:** The discussion page will reload with the new comment under the desired thread.

Test 2:

- **Pre-Requirements:** The user must be on the Discussion page, and has at least Club Member account level.
- **Use Case Reference #4.1-10**
- **Description:** Test to see if the system allows a user to respond with an empty response to a currently existing thread.
- **Inputs for input fields:**
 - Empty reply text in the '**Message**' text box.
- **Test Steps:**
 1. On the discussion page, find a currently existing thread to reply to
 2. On the right side of the thread, click the button titled '**Reply**'.
 3. Click '**Submit**'.
- **Expected Results:** The discussion page will reload with the error saying that the message field is required.

Test 3:

- **Pre-Requirements:** The user must be on the Discussion page, and has at least Club Member account level.

- **Use Case Reference #4.1-10**
- **Description:** Test to see if the system allows a user to respond to a currently existing discussion thread and include an image.
- **Inputs for input fields:**
 - Desired reply text in the '**Message**' text box.
 - '**Source**' and '**Image Description**' when adding image.
- **Test Steps:**
 1. On the discussion page, find a currently existing thread to reply to
 2. On the right side of the thread, click the button titles '**Reply**'.
 3. Enter desired text into the text box.
 4. Click '**Insert**'.
 5. Click '**Insert/edit image**'.
 6. Copy the file path for where the image is found on the computer.\
 7. Paste it into the 'Source' field.
 8. Give it an image description in the 'Image Description' field.
 9. Unclick the check box that says 'Constrain Properties'.
 10. Click '**OK**'
- **Expected Results:** The discussion page will reload with the new comment & image under the thread.

Test 4:

- **Pre-Requirements:** The user must be on the Discussion page, and has at least Club Member account level.
- **Use Case Reference #4.1-10**
- **Description:** Test to see if the system allows a user to respond to a currently existing discussion thread and include a video.
- **Inputs for input fields:**
 - Desired reply text in the '**Message**' text box.
 - Embed code for link of the video
- **Test Steps:**
 1. On the discussion page, find a currently existing thread to reply to
 2. On the right side of the thread, click the button titles '**Reply**'.
 3. Enter desired text into the text box.
 4. Click '**Insert**'.

5. Click '**Insert/edit video**'.
 6. Click on the '**Embed**' tab.
 7. Paste it into the provided text field.
 8. Click '**OK**'
- **Expected Results:** The discussion page will reload with the new comment & video under the thread.

TC_008 – Calendar Event Creation

Test 1:

- **Pre-Requirements:** User must be logged in and have administrator privileges.
- **Use Case Reference #4.1-11**
- **Description:** Test the system to see if it correctly allows a new event to be created for a proper date
- **Inputs for input fields:**
 - '**Event Name**'
 - '**Date**'
 - '**Time**'
 - '**Location**'
 - '**Event Description**'
- **Test Steps:**
 1. At the top navigation bar, click '**Events**'.
 2. Click '**Event Submission**' found at the top of the calendar.
 3. Input all required fields above (for '**Date**' enter 05/20/2015).
 4. Click '**Submit**'.
- **Expected Results:** The event creation page will reload and notify the administrator user that the event has been submitted. The event can now be found on the calendar.

Test 2:

- **Pre-Requirements:** User must be logged in and not have administrator privileges.
- **Use Case Reference #4.1-11**
- **Description:** Test the system to see if it does not allow for a non administrator to post new

events. This feature is available only to admins.

- **Inputs for input fields:**

- 'Event Name'
- 'Date'
- 'Time'
- 'Location'
- 'Event Description'

- **Test Steps:**

5. At the top navigation bar, click '**Events**'.

6. Click '**Event Submission**' found at the top of the calendar.

7. Input all required fields above (for '**Date**' enter 05/21/2015).

8. Click '**Submit**'.

- **Expected Results:** The event creation page will reload and notify the user that the event has been submitted.

Test 3:

- **Pre-Requirements:** User must be logged in at any account level.

- **Use Case Reference #4.1-11**

- **Description:** Test the system to see if it correctly allows for multiple events to be created on the same date.

- **Inputs for input fields:**

- 'Event Name'
- 'Date'
- 'Time'
- 'Location'
- 'Event Description'

- **Test Steps:**

9. At the top navigation bar, click '**Events**'.

10. Click '**Event Submission**' found at the top of the calendar.

11. Input all required fields above (for '**Date**' enter 05/20/2015).

12. Click '**Submit**'.

- **Expected Results:** The event creation page will reload and notify the user that the event has been submitted. The event will be listed under the event created from TC_009 Test 1.

TC_009 – Event Modification **NYI

- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-12**
- **Description:** Test to see if the system properly allows a user to sign in to their account.
- **Test Plan:**
 1. Sign in to a test account that has been created using correct credentials.
 2. Sign in to a test account that has been created using caps lock.
 3. Sign in to a test account that has been created using an incorrect password.
 4. Sign in to an account that does not exist.

TC_010 – Event Deletion **NYI

- **Pre-Requirements:** The user must be logged in as an admin and on the event page.
- **Use Case Reference #4.1-13**
- **Description:** Test to see if an event created on the calander is allowed to be deleted.
- **Inputs for input fields: (none)**
- **Test Steps:**
 1. On the Events page, find the event listed in the calendar that is to be deleted.
 2. Click the event link
 3. Click '**Delete Event**' in the upper right hand corner.
 4. A Confirmation dialog box will appear. Click '**Delete**'.
- **Expected Results:** The event page will reload with a confirmation that the event has been deleted.

TC_011 – Calender Navigation**Test 1:**

- **Pre-Requirements:** The user must be on the website and logged in. The user must currently be on the Events page.
- **Use Case Reference #4.1-14**
- **Description:** Test to see if the system allows for any user to browse through the events listed by month.

- **Inputs for input fields: (none)**
- **Test Steps:**
 1. On the Events page, locate the arrows next to the current month
 2. Click the arrow to the right of the month twice.
 3. Click the arrow to the left of the month 4 times.
 4. Click the arrow to the right of the month 6 times.
- **Expected Output:** The user can properly navigate to any month they desire.

TC_012 – Custom Page Creation**Test 1:**

- **Pre-Requirements:** Must be logged in as an administrator and on the 'Dashboard' page.
- **Use Case Reference #4.1-15**
- **Description:** Test the creation of a custom page for the website
- **Inputs for input fields:**
 - **'Title':** Test Title
 - **'Body Content':** This is the body of the test for the Body Content.
- **Test Steps:**
 1. Click on the '**Custom Pages**' button, on the left side of the page. Takes user to the '**Manage Custom Pages**' page.
 2. Then click the '**Create New Page**' button, at the top of the page. Takes user to the '**Create Custom Page**' page.
 3. On the '**Create Custom Page**' page, use the specified inputs for the '**Title**' field and the '**Body Content**' field.
 4. Click on the '**Submit**' button, on the bottom of the page, to create the new custom page.
- **Expected Output:** At the top of the '**Create Custom Page**' page a green notification panel displays 'Your page was successfully created!'. On the '**Manage Custom Pages**' page your created custom page should be at the bottom of the listed custom pages with your username.

Test 2:

- **Pre-Requirements:** Must be logged in as an administrator and on the 'Dashboard' page.
- **Use Case Reference #4.1-15**
- **Description:** Test the creation of a custom page for the website
- **Inputs for input fields:**
 - **'Title':** Test for embedding videos
 - **'Body Content':** `<iframe width="560" height="315" src="https://www.youtube.com/embed/h2shopQ3owY" frameborder="0" allowfullscreen></iframe>`
- **Test Steps:**
 1. Click on the '**Custom Pages**' button, on the left side of the page. Takes user to the '**Manage Custom Pages**' page.
 2. Then click the '**Create New Page**' button, at the top of the page. Takes user to the '**Create**

Custom Page' page.

3. On the '**Create Custom Page**' page, use the specified input for the '**Title**' field. For the '**Body Content**' field, on the toolbar click on '**Insert**', which will bring up a drop down of options. Click on '**Insert/edit video**' to bring up the input box. Click on the '**Embed**' tab and insert the specified input, for the '**Body Content**', for Test 2. Click the '**Ok**' button.
4. Click on the '**Submit**' button, on the bottom of the page, to create the new custom page.
- **Expected Output:** At the top of the '**Create Custom Page**' page a green notification panel displays 'Your page was successfully created!'. On the '**Manage Custom Pages**' page your created custom page should be at the bottom of the listed custom pages with your username.

Test 3:

- **Pre-Requirements:** Must be logged in as an administrator and on the 'Dashboard' page.
- **Use Case Reference #4.1-15**
- **Description:** Test the creation of a custom page for the website
- **Inputs for input fields:**
 - '**Title**':
 - '**Body Content**':
- **Test Steps:**
 1. Click on the '**Custom Pages**' button, on the left side of the page. Takes user to the '**Manage Custom Pages**' page.
 2. Then click the 'Create New Page' button, at the top of the page. Takes user to the '**Create Custom Page**' page.
 3. On the '**Create Custom Page**' page, leave the 'Title' field and the '**Body Content**' field empty.
 4. Click on the '**Submit**' button, on the bottom of the page, to create the new custom page.
- **Expected Output:** At the top of the '**Create Custom Page**' page a red notification panel displays 'The Title field is required.' and 'The Body Text field is required.'.

TC_013 – Custom Page Modification**Test 1:**

- **Pre-Requirements:** See TC_014 Pre-Reqs. A page must exist to modify.
- **Use Case Reference #4.1-16**
- **Description:** Testing the editing of already existing custom pages
- **Input for input fields:**
 - **'Title':** New Test Title
 - **'Body Content':** This is the Edit for the body of the test for the Body Content.
- **Test Steps:**
 1. Click on the **'Custom Pages'** button, on the left side of the page. Takes user to the **'Manage Custom Pages'** page.
 2. Search for the existing custom page to be modified and then click the **'Edit'** button.
 3. On the **'Edit Custom Page'** enter the specified inputs for the **'Title'** field and for the **'Body Content'** field.
 4. Click the **'Submit'** button, on the bottom of the page, to finalize the edit for the custom page.
- **Expected Output:** At the top of the **'Create Custom Page'** page a green notification panel displays 'Your page was successfully modified!'. On the **'Manage Custom Pages'** page your modified custom page should be in the same position as before, of the listed custom pages, with the modified title and time it was modified.

Test 2:

- **Pre-Requirements:** See TC_014 Pre-Reqs. A page must exist to modify.
- **Use Case Reference #4.1-16**
- **Description:** Testing the editing of already existing custom pages
- **Input for input fields:**
 - **'Title':**
 - **'Body Content':**
- **Test Steps:**
 1. Click on the **'Custom Pages'** button, on the left side of the page. Takes user to the **'Manage Custom Pages'** page.
 2. Search for the existing custom page to be modified and then click the **'Edit'** button.
 3. On the **'Edit Custom Page'** do not change the current inputs for the **'Title'** field and for the **'Body Content'** field.

4. Click the **'Submit'** button, on the bottom of the page, to finalize the edit for the custom page.
- **Expected Output:** At the top of the **'Create Custom Page'** page a green notification panel displays 'Your page was successfully modified!'. On the **'Manage Custom Pages'** page your modified custom page should be in the same position as before, of the listed custom pages, with the modified title and time it was modified.

Test 3:

- **Pre-Requirements:** See TC_014 Pre-Reqs. A page must exist to modify.
- **Use Case Reference #4.1-16**
- **Description:** Testing the editing of already existing custom pages
- **Input for input fields:**
 - **'Title':**
 - **'Body Content':**
- **Test Steps:**
 1. Click on the **'Custom Pages'** button, on the left side of the page. Takes user to the **'Manage Custom Pages'** page.
 2. Search for the existing custom page to be modified and then click the **'Edit'** button.
 3. On the **'Edit Custom Page'** delete the current inputs for the **'Title'** field and for the **'Body Content'** field and do not enter any new information for either field.
 4. Click the **'Submit'** button, on the bottom of the page, to finalize the edit for the custom page.
- **Expected Output:** At the top of the **'Create Custom Page'** page a red notification panel displays 'The Title field is required.' and 'The Body Text field is required.'.

TC_014 – Custom Page Deletion**Test 1:**

- **Pre-Requirements:** See TC_014 Pre-Reqs. A page must exist to delete.
- **Use Case Reference #4.1-17**
- **Description:** Testing the deletion of created custom pages
- **Test Steps:**
 1. Click on the **'Custom Page'** button, on the left side of the page. Takes user to the **'Manage Custom Pages'** page.
 2. Search for the existing custom page to be modified and then click the **'Delete'** button, bringing up an alert for deletion confirmation.

3. Click the 'Yes, delete the page' button.

- **Expected Output:** On the top of the '**Manage Custom Pages**' page a green notification panel displays 'Custom Page was successfully deleted', and the custom page is no longer displayed on the '**Manage Custom Pages**' page.

Test 2:

- **Pre-Requirements:** See TC_014 Pre-Reqs. A page must exist to delete.
- **Use Case Reference #4.1-17**
- **Description:** Testing the deletion of created custom pages
- **Test Steps:**
 1. Click on the '**Custom Page**' button, on the left side of the page. Takes user to the '**Manage Custom Pages**' page.
 2. Search for the existing custom page to be modified and then click the '**Delete**' button, bringing up an alert for deletion confirmation.
 3. Click the '**No, do not delete**' button.
- **Expected Output:** Takes user back to the '**Manage Custom Pages**' page with the custom page still being displayed.

TC_015 – Administrator Reads Messages**Test 1:**

- **Pre-Requirements:** Must be logged in as an administrator.
- **Use Case Reference #4.1-18**
- **Description:** Test the reading of messages and/or suggestions to sent to the Film Club.
- **Test Steps:**
 1. On the toolbar, from the top of any page, click your username to bring down an option drop down menu.
 2. In the drop down menu, click the '**Dashboard**'.
- **Expected Output:** Takes user to the '**Dashboard**' page where the messages are displayed in the body of the page to be read.

TC_o16 – Administrator Suspends User**Test 1:**

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page. User accounts must exist to be suspended.
- **Use Case Reference #4.1-19**
- **Description:** Testing the activity of an administrator to be able to suspend a registered account.
- **Test Steps:**
 1. Click on the '**Manage Users**' button, on the left of the '**Dashboard**' page. Takes user to '**User Management**' page.
 2. Current user accounts will appear on the body of the page. Find the user whose account is to be suspended.
 3. In the row, for the user to be suspended, at the far right column there will be three buttons. Click on the red '**Suspend**' button.
- **Expected Output:** '**User Management**' page will be refreshed. Find the user, whose account was suspended, and in there '**Type**' column it will be blank where their previous title/rank was displayed.

TC_017 – Administrator Promotes User**Test 1:**

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page. User accounts must exist to be suspended.
- **Use Case Reference #4.1-20**
- **Description:** Testing the activity of an administrator to be able to promote a registered account.
- **Test Steps:**
 1. Click on the '**Manage Users**' button, on the left of the '**Dashboard**' page. Takes user to '**User Management**' page.
 2. Current user accounts will appear on the body of the page. Find the user whose account is to be promoted.
 3. In the row, for the user to be promoted, at the far right column there will be three buttons. Click on the green '**Promote**' button.
- **Expected Output:** '**User Management**' page will be refreshed. Find the user, whose account was promoted, and in there '**Type**' column there will be a new title/rank than before, indicating the user has been promoted.

Test 2:

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page. User accounts must exist to be suspended.
- **Use Case Reference #4.1-20**
- **Description:** Suspends the user's account and then promotes into to active again.
- **Test Steps:**
 1. Execute TC_016: Test 1.
 2. Find the account that TC_017: Test 1 was executed on. In the row, for the user to be promoted, at the far right column there will be three buttons. Click on the green '**Promote**' button.**Expected Output:** '**User Management**' page will be refreshed. Find the user, whose account was promoted, and in there '**Type**' column the previous title/rank, before suspension, will reappear showing that the account is now active, not suspended, again.

TC_018 – Administrator adds Videos**Test 1:**

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page.
- **Use Case Reference #4.1-21**
- **Description:** Adds an embedded video to the site's '**Browse Videos**' page.
- **Inputs:**
 - **YouTube ID:** 6TIL_qW8Mts
 - **Tags:** Select Cool Tag
 - **Site Description:** Test Video: Unreal Tournament
- **Test Steps:**
 1. Click on the '**Videos**' button, on the left of the '**Dashboard**' page. Takes user to '**Add Videos**' page.
 2. Enter in the specified inputs for the '**YouTube ID**' field, '**Tags**' selection, and '**Site Description**' field.
 3. At the bottom of the '**Add Videos**' page click the '**Submit**' button.
- **Expected Output:** At the top of the '**Add Videos**' page a green panel displays, 'You have successfully added the video!'. On the '**Browse Videos**' page the submitted video will be displayed.

Test 2:

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page.
- **Use Case Reference #4.1-21**
- **Description:** Adds an embedded video to the site's '**Browse Videos**' page.
- **Inputs:**
 - **YouTube ID:** LlReioKMP1g
 - **Tags:**
 - **Site Description:** Test Video: AMV Anime Fight
- **Test Steps:**
 1. Click on the '**Videos**' button, on the left of the '**Dashboard**' page. Takes user to '**Add Videos**' page.
 2. Enter in the specified inputs for the '**YouTube ID**' field and '**Site Description**' field.
 3. At the bottom of the '**Add Videos**' page click the '**Submit**' button.
- **Expected Output:** At the top of the '**Add Videos**' page a green panel displays, 'You have

successfully added the video!'. On the '**Browse Videos**' page the submitted video will be displayed with the selected '**Site Description**'.

Test 3:

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page.
- **Use Case Reference #4.1-21**
- **Description:** Attempts to an embedded video to the site's '**Browse Videos**' page, without using inputs.
- **Inputs:**
 - **YouTube ID:**
 - **Tags:**
 - **Site Description:**
- **Test Steps:**
 1. Click on the '**Videos**' button, on the left of the '**Dashboard**' page. Takes user to '**Add Videos**' page.
 2. Leave the '**YouTube ID**' field and '**Site Description**' field empty.
 3. At the bottom of the '**Add Videos**' page click the '**Submit**' button.
- **Expected Output:** At the top of the '**Add Videos**' page a red panel displays, 'The YouTube ID field is required.' and 'The Site Description field is required.'.

TC_019 – Administrator adds Images**Test 1:**

- **Pre-Requirements:** Must be signed in as administrator.
- **Use Case Reference #4.1-22**
- **Description:** Test the uploading of images to the websites 'Browse Our Gallery' page.
- **Test Steps:**
 1. From the top of any website page click on the '**Media**' tab to open a drop-down menu.
 2. Click on the first option in the drop-down menu, '**Pictures**'. Takes user to '**Browse Our Gallery**' page.
 3. Click on the '**Add Photos**' button. Will take user to '**File Upload**' page.
 4. Click the '**Choose File**' button. Will open up a File Dialog, navigate to the folder to the photo to be uploaded. Click the file, then click the '**Open**' button on the File Dialog.
 5. Click the '**Upload**' button, at the bottom of the '**File Upload**' page.
- **Expected Result:** At the top of the '**File Upload**' page will be a green panel saying, 'Your file was successfully uploaded!'. Uploaded picture will now be displayed in the '**Browse Our Gallery**' page.

TC_020 – User Creates Request or Suggestion**Test 1:**

- **Pre-Requirements:** Currently on the website.
- **Use Case Reference #4.1-23**
- **Description:** Create a Suggestion message to send to the Film Club.
- **Inputs:**
 - **First Name:** Bob
 - **Last Name:** Bobby
 - **Email:** Bob@email.com
 - **Phone:** 1234567
 - **Subject:** Test Suggestion
 - **Message:** We would like to test suggestion messages.
- **Test Steps:**
 1. From the top of any website page click on the '**About**' tab to open a drop-down menu.
 2. Click on the fourth option in the drop-down menu, '**Suggestions**'. Takes user to '**Suggestions**' page.

3. Enter in the specified inputs for '**First Name**', '**Last Name**', '**Email**', '**Phone**', '**Subject**', and '**Message**'.
4. At the bottom of the page click the '**Submit**' button.
- **Expected Output: 'Suggestions'** page refreshes. At the top of the page a green panel says 'Thank you! Your suggestion was successfully submitted. We may contact you soon.'

Test 2:

- **Pre-Requirements:** Currently on the website.
- **Use Case Reference #4.1-23**
- **Description:** Create a blank Suggestion message to send to the Film Club.
- **Inputs:**
 - **First Name:**
 - **Last Name:**
 - **Email:**
 - **Phone:**
 - **Subject:**
 - **Message:**
- **Test Steps:**
 1. From the top of any website page click on the '**About**' tab to open a drop-down menu.
 2. Click on the fourth option in the drop-down menu, '**Suggestions**'. Takes user to '**Suggestions**' page.
 3. Leave the '**First Name**', '**Last Name**', '**Email**', '**Phone**', '**Subject**', and '**Message**' fields empty.
 4. At the bottom of the page click the '**Submit**' button.
- **Expected Output: 'Suggestions'** page refreshes. At the top of the page a red panel says, 'The First Name field is required;', 'The Last name field is required', 'The Email field is required', 'The Phone field is required', 'The Subject field is required', and 'The Message field is required'.

TC_021 – User Changes Template

- *****
- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-3**
- **Description:** Test to see if the system properly allows a user to sign in to their account.
- **Test Plan:**
 5. Sign in to a test account that has been created using correct credentials.
 6. Sign in to a test account that has been created using caps lock.
 7. Sign in to a test account that has been created using an incorrect password.
 8. Sign in to an account that does not exist.

TC_022 – User Profile Modification

- *****
- **Pre-Requirements:** The user must be on the website and not currently logged in.
- **Use Case Reference #4.1-3**
- **Description:** Test to see if the system properly allows a user to sign in to their account.
- **Test Plan:**
 9. Sign in to a test account that has been created using correct credentials.
 10. Sign in to a test account that has been created using caps lock.
 11. Sign in to a test account that has been created using an incorrect password.
 12. Sign in to an account that does not exist.

TC_023 – Contact Information Retrieval**Test 1:**

- **Pre-Requirements:** The user must be on the website.
- **Use Case Reference #4.1-26**
- **Description: Retrieves the Film Club's contact information.**
- **Test Steps:**
 1. From the top of any website page click on the '**About**' tab to open a drop-down menu.
 2. Click on the third option in the drop-down menu, '**Contact**'. Takes user to **Contact Information** page.
- **Expected Output:** User is on the '**Contact Information**' page, with the contact information displayed on the body of the page.

TC_024 – Contact Information Modification**Test 1:**

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page.
- **Use Case Reference #4.1-27**
- **Description:** Change the contact information for the GSU Film Club.
- **Input:** This is a test for contact information modification. P.O. Box 5555 Statesboro, GA 30460
- **Test Steps:**
 1. On the left of the '**Dashboard**' page click the '**Site Information**' button. Takes user to '**Site Information**' page.
 2. At the bottom of the '**Site Information**' page, will be the '**Contact Us**' dialog box. Copy the specified input into the bottom of the '**Contact Us**' dialog box.
 3. At the bottom of the '**Contact Us**' dialog box click the '**Submit Changes**' button;
- **Expected Output:** At the top of the 'Site Information' page will be a green panel saying, 'Your page was successfully modified!'. The '**Contact Us**' dialog box should now have the specified information in it.

Test 2:

- **Pre-Requirements:** Must be logged in as administrator and on the '**Dashboard**' page.
- **Use Case Reference #4.1-27**
- **Description:** Submit a change for the contact information without changing anything.
- **Input:**
- **Test Steps:**
 1. On the left of the '**Dashboard**' page click the '**Site Information**' button. Takes user to '**Site Information**' page.
 2. At the bottom of the '**Site Information**' page, will be the '**Contact Us**' dialog box. Leave the current contact information in the '**Contact Us**' dialog box.
 3. At the bottom of the '**Contact Us**' dialog box click the '**Submit Changes**' button;
- **Expected Output:** At the top of the 'Site Information' page will be a green panel saying, 'Your page was successfully modified!'.

7.3 Subsystem Tests

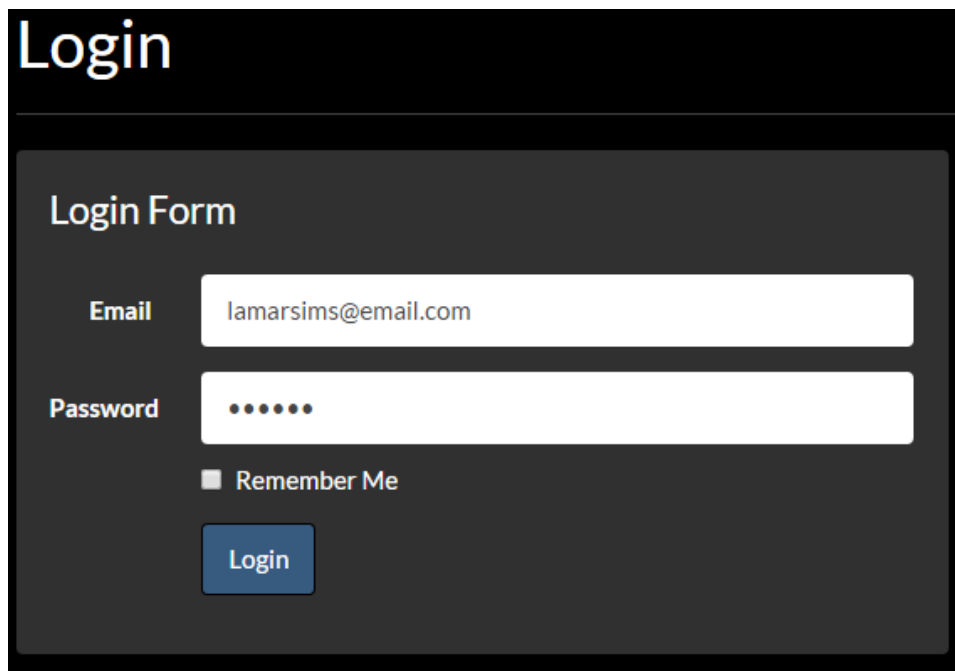
7.3.1 Introduction

The Subsystem Test tests our Login subsystem for the GSU Film Club's website. It deals with the verification and validation of the email and password specified to see if it matches the database's email and that email's password. It will also test if the user does not specify an email but a password, does not specify a password or email, and does not specify either password or email. It as well tests the Logout function for the website.

7.3.2 Unit Tests

1. **Unit Test 1**

Test a valid login, with a member that is already registered in the website's DB.



The screenshot shows a web interface for logging in. At the top, the word "Login" is displayed in a large, white, sans-serif font against a dark background. Below this, there is a dark gray rectangular box containing the "Login Form". Inside the form, the title "Login Form" is written in a white, sans-serif font. There are two input fields: the first is labeled "Email" in white text and contains the text "lamarsims@email.com"; the second is labeled "Password" in white text and contains six black dots. Below the password field is a checkbox labeled "Remember Me" in white text. At the bottom of the form is a blue button with the word "Login" in white text.

Unit Test Driver Running

[Stop](#)

Results

[Refresh](#)

Test Name	Test Datatype	Expected Datatype	Result	File Name	Line Number	Notes	Timestamp
Unit Test test...	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	25		2015-05-01 14:29:14
Email or password specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	280		2015-05-01 14:29:14
Email was specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	300		2015-05-01 14:29:14
Password was specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	320		2015-05-01 14:29:14
Good password!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	344		2015-05-01 14:29:14

Summary

Successes: 5
 Failures: 0
 Rate: 100%
 Coverage: 50%

2. Unit Test 2

Test the Logout of a member.

Unit Test Driver Not Running

[Start](#)

Results

[Refresh](#)

Test Name	Test Datatype	Expected Datatype	Result	File Name	Line Number	Notes	Timestamp
Unit Test test...	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	25		2015-05-01 14:30:36
Actual and session account lev	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	252		2015-05-01 14:30:36
Bad password does not match ba	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	259		2015-05-01 14:30:36
No email or password specified	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	274		2015-05-01 14:30:36
No email specified!	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	294		2015-05-01 14:30:36
No password specified!	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	314		2015-05-01 14:30:36

Summary

Successes: 3
 Failures: 3
 Rate: 50%
 Coverage: 40%

3. Unit Test 3

Test an invalid login where the email is valid and the password is not.

Login

Login Form

Email

Bob@email.com

Password

.....

☐ Remember Me

Login

Invalid email address and password combination.

Unit Test Driver Running

Stop

Results

Refresh

Test Name	Test Datatype	Expected Datatype	Result	File Name	Line Number	Notes	Timestamp
Unit Test test...	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	25		2015-05-01 14:32:11
Email or password specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	280		2015-05-01 14:32:11
Email was specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	300		2015-05-01 14:32:11
Password was specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	320		2015-05-01 14:32:11
Bad password attempt	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	337		2015-05-01 14:32:11

Summary

Successes: 4

Failures: 1

Rate: 80%

Coverage: 50%

4. Unit Test 4

Tests invalid login where, no email is specified and password is specified.

Login

Login Form

Email

Password

☐ Remember Me

Login

The Email field is required.
Invalid email address and password combination.

Unit Test Driver Running

Stop

Results

Refresh

Test Name	Test Datatype	Expected Datatype	Result	File Name	Line Number	Notes	Timestamp
Unit Test test...	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	25		2015-05-01 14:36:31
Email or password specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	280		2015-05-01 14:36:31
No email specified!	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	294		2015-05-01 14:36:31
Password was specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	320		2015-05-01 14:36:31

Summary

Successes: 3
Failures: 1
Rate: 75%
Coverage: 30%

5. Unit Test 5

Test invalid login, where email is specified and password is not.

Login

Login Form

Email

Bob@email.com

Password

Password

☐ Remember Me

Login

The Password field is required.

Unit Test Driver Running

Stop

Results

Refresh

Test Name	Test Datatype	Expected Datatype	Result	File Name	Line Number	Notes	Timestamp
Unit Test test...	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	25		2015-05-01 14:37:40
Email or password specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	280		2015-05-01 14:37:40
Email was specified!	Boolean	Boolean	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	300		2015-05-01 14:37:40
No password specified!	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	314		2015-05-01 14:37:40

Summary

Successes: 3
Failures: 1
Rate: 75%
Coverage: 30%

6. Unit Test 6

Test invalid login where neither email nor password is specified.

Login

Login Form

Email

Password

☐ Remember Me

Login

The Email field is required.

The Password field is required.

Unit Test Driver Running

Stop

Results

Refresh

Test Name	Test Datatype	Expected Datatype	Result	File Name	Line Number	Notes	Timestamp
Unit Test test...	String	String	Passed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	25		2015-05-01 14:40:49
No email or password specified	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	274		2015-05-01 14:40:49
No email specified!	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	294		2015-05-01 14:40:49
No password specified!	Boolean	Boolean	Failed	/home/chadgold/public_html/se/test/application/controllers/login_system/verify_login.php	314		2015-05-01 14:40:49

Summary

Successes: 1

Failures: 3

Rate: 25%

Coverage: 30%

7.4 Evaluation of Tests

Test Case ID	Actual Results	P/F	Intent	Actions to be taken
TC_001_Test1	Same as Expected Result	P	Register new user	None
TC_001_Test2	Same as Expected Result	P		
TC_001_Test3	Same as Expected Result	P		
TC_002_Test1	Same as Expected Result	P	Login to currently existitng account	
TC_002_Test2	404 Page Not Found Error	F	Login to account with incorrect password	Error checking to check for incorrect login.
TC_002_Test3	404 Page Not Found Error	F	Login to without specifying email or password	Error checking to check for incorrect login.
TC_003_Test1	Same as Expected Result	P	User logs out	None
TC_004_Test1	Same as Expected Result	P	Use social media link to go to GSU Film Club YouTube page	None
TC_004_Test2	Same as Expected Result	P	Use social media link to go to GSU Film Club Twitter page	None
TC_004_Test3	Just takes to Facebook home page	F	Use social media link to go to GSU Film Club Facebook page	Fix linking error to link to GSU Film Club Facebook page

TC_005_Test1	Same as Expected Result	P	Upload a profile picture	None
TC_005_Test2	Same as Expected Result	P	Replace a previous profile picture	None
TC_005_Test3	Page refreshed, and nothing happened	F	Catch trying to upload any file that was not a picture file	Add Error checking for uploading of incorrect file types.
TC_006_Test1	Same as Expected Result	P	Delete a picture	None
TC_006_Test2	Throws PHP error	F	Delete a profile picture when there is not one there	Add Error checking for deleting profile pictures
TC_007_Test1	Same as Expected Result	P	Reply to an existing discussion thread	None
TC_007_Test2	Same as Expected Result	P	Was to get error message if the 'Message' field was left empty	None
TC_007_Test3	Reply displayed without the picture	F	Was to reply to a discussion thread with an embedded picture	Implement a file dialog to be able to choose a picture to embedded in the reply
TC_007_Test4	Same as Expected Result	P	Reply to a discussion thread with an embedded video	None
TC_008_Test1				
TC_008_Test2				
TC_008_Test3				
TC_009_Test1				
TC_010_Test1	Same as	P	Delete a previously created	None

	Expected Result		event from the calendar	
TC_011_Test1	Same as Expected Result	P	Navigate the calendar, from month to month, and year to year	None
TC_012_Test1	Same as Expected Result	P	Create a custom page	None
TC_012_Test2	Same as Expected Result	P	Create a custom page with video embedded in the body	None
TC_012_Test3	Same as Expected Result	P	Create a custom page without 'Title' and 'Body Content' to generate error messages	None
TC_013_Test1	Same as Expected Result	P	Modify a custom page	None
TC_013_Test2	Same as Expected Result	P	Choose to modify a custom page, but don't change any of the previous info	None
TC_013_Test3	Same as Expected Result	P	Choose to modify a custom page, remove it's previous info, but don't reenter the info	None
TC_014_Test1	Same as Expected Result	P	Delete a custom page	None
TC_014_Test2	Same as Expected Result	P	Click 'Delete' button, but decide not to delete the custom page	None
TC_015_Test1	Same as Expected Result	P	Navigate to messages to be read	None
TC_016_Test1	Same as Expected	P	Suspend a user's account	None

	Result			
TC_017_Test1	Same as Expected Result	P	Promote a user's account	None
TC_017_Test2	Same as Expected Result	P	Suspends user's account, then promotes then back to an active member	None
TC_018_Test1	Same as Expected Result	P	Add a embedded video to the webpage	None
TC_018_Test2	Throws PHP error	F	Add a embedded video to the webpage without a specified tag	Use error catching to ensure the tags are not being evaluated to go into the database
TC_018_Test3	Same as Expected Result	P	Add a video without specifying the 'YouTube ID' or the 'Site Description' to get the appropriate error messages	None
TC_019_Test1	Same as Expected Result	P	Add a picture to the webpage's gallery page	None
TC_020_Test1	Same as Expected Result	P	Submit a suggestion to the GSU Film Club	None
TC_020_Test2	Same as Expected Result	P	Submit a suggestion to the GSU Film Club, without filling out the form, to get the appropriate error messages	None
TC_021_Test1				
TC_022_Test1				
TC_023_Test1	Same as Expected Result	P	Navigate to the GSU Film Club's contact information page	None
TC_024_Test1	Same as	P	Modify GSU Film Club's	None

	Expected Result		contact information	
TC_024_Test2	Same as Expected Result	P	Click on button to modify contact information, but don't change any of the information and just re-submit the contact information	None

7.5 Testing Tools

1. **Selenium IDE:** An automated web regression testing tool. This is actually installed as an add-on for Mozilla Firefox, and has several components. Using Selenium, a list of actions can be recorded as the tester navigates the website. These actions are saved as scripts which can then be changed to test how the web system would act under different actions. Simplifies the testing process and makes it more effective. Below details the main features of Selenium used.
 1. **Record feature:** By hitting record, a test can be started as the tester uses the system for its intended purpose. Upon completing the desired test, the record feature takes all those user interactions (button clicks, text field entries, database validations, etc.) and creates a script which can be ran again. It streamlines the repetitive process of attempting to test the system with multiple different data values. Once one test has been recorded and saved, the data can be changed, thus testing for accurate functionality regarding the test case.
 2. **Play Current Test Case:** Upon completing a record, the test case can be saved. The Play Current Test Case is the feature that takes the script generated by the record feature and plays it back with the manipulated data. This is essentially what will provide us with the actual results that will be compared to the expected results of the test case.
 3. **Play Entire Test Suite:** When multiple test cases are saved as scripts, they are listed to the left of the User Interface. Upon clicking play entire test suite, the entire list of independent test scripts are ran. This feature is effective when trying to run larger tests, such as if a user were to use many features in a short amount of time.
2. We were unable to implement PHPUnit with our project, as it had problems with the Codeigniter framework that we were using. So instead we made our own little unit testing

PHP class that we also made into a webpage to visualize our unit tests that we ran. Since we could not implement PHPUnit, we could not use it's coverage tool, so we also implemented our own way to get the coverage for each unit test ran.

8. Glossary

Table 5.1-1 Glossary of Terms

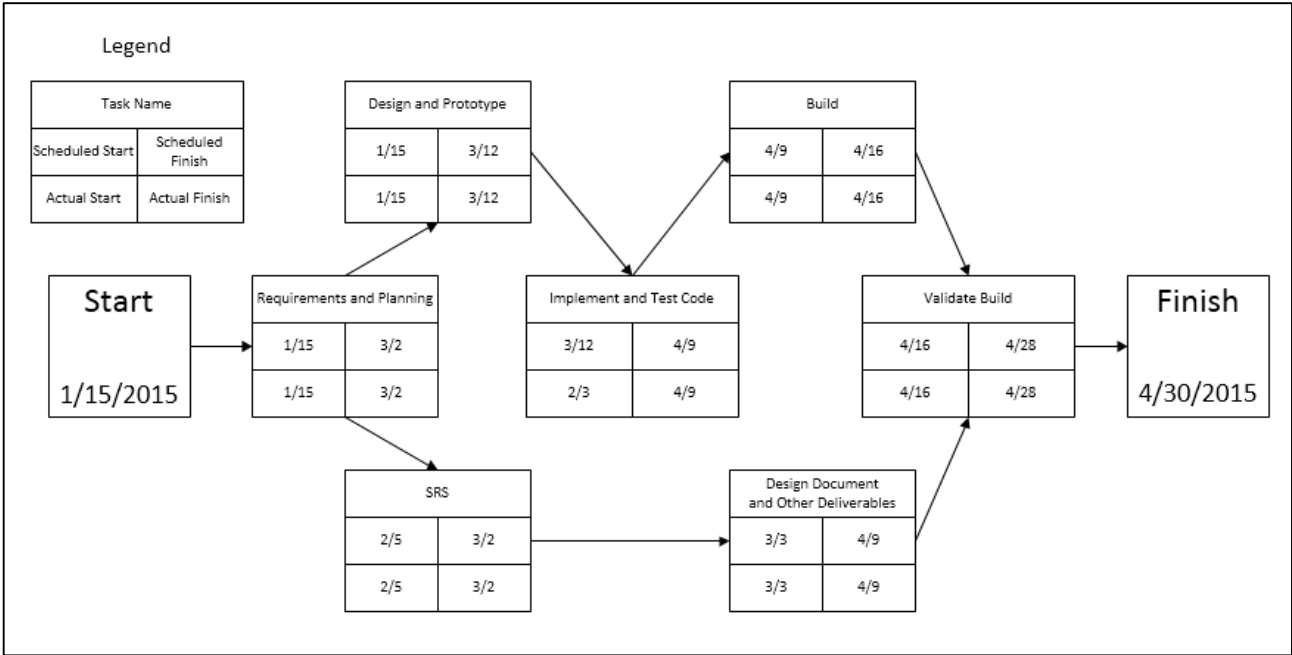
Term	Description
SRS	<i>Software Requirements Specification</i> . A description of a software system being developed intended to articulate a set of requirements between the user and the developers. (e.g. this document)
CMS	<i>Content Management System</i> . A means for privileged users to add, edit, and remove content such as web pages, blog posts, and media through the aid of software.
Programming Languages	The means in which software developers create applications in the form of a structured code to communicate with computer software.
HTML	<i>Hypertext Markup Language</i> . The code that allows web browsers to view content on the web.
PHP	<i>PHP: Hypertext Processor</i> . The code that allows for dynamic content in web pages.
IDE	<i>Integrated Development Environment</i> . A tool that assists the developers in writing and testing code.
PHP-Storm	A specific IDE by JetBrains that allows for convenient, fast, and powerful development of web applications.
JetBrains	A software company whose products are geared toward software developers.
Framework	A reusable software development environment that provides functionality for a developer creating a larger application.
Bootstrap	A framework for creating websites and web applications. It provides the look and feel for the particular web system.

Server	A software running on a computer that provides a service (such as sending a web page).
Client	Software or hardware that makes use of the service provided by a server.
Operating System (OS)	A software that manages the hardware within a computer. (e.g. Microsoft Windows, Apple Mac OS)
Interpreter	A software that takes code written by a programmer and executes the code into instructions for the computer to complete a task.
DBMS	<i>Database Management System</i> . A collection of software that retrieves, inserts, alters, and deletes information from a database.
Web Browser	The software that presents web content. (e.g. Google Chrome, Mozilla Firefox, Internet Explorer)
Stakeholder	An individual or group that has an interest in the software system. (Not inclusive of the development team)
User Interface (UI)	A way in which the user interacts with the system.
Use Case	A list of steps that defines interactions between the users and the system in order to achieve a goal.

9. Appendix

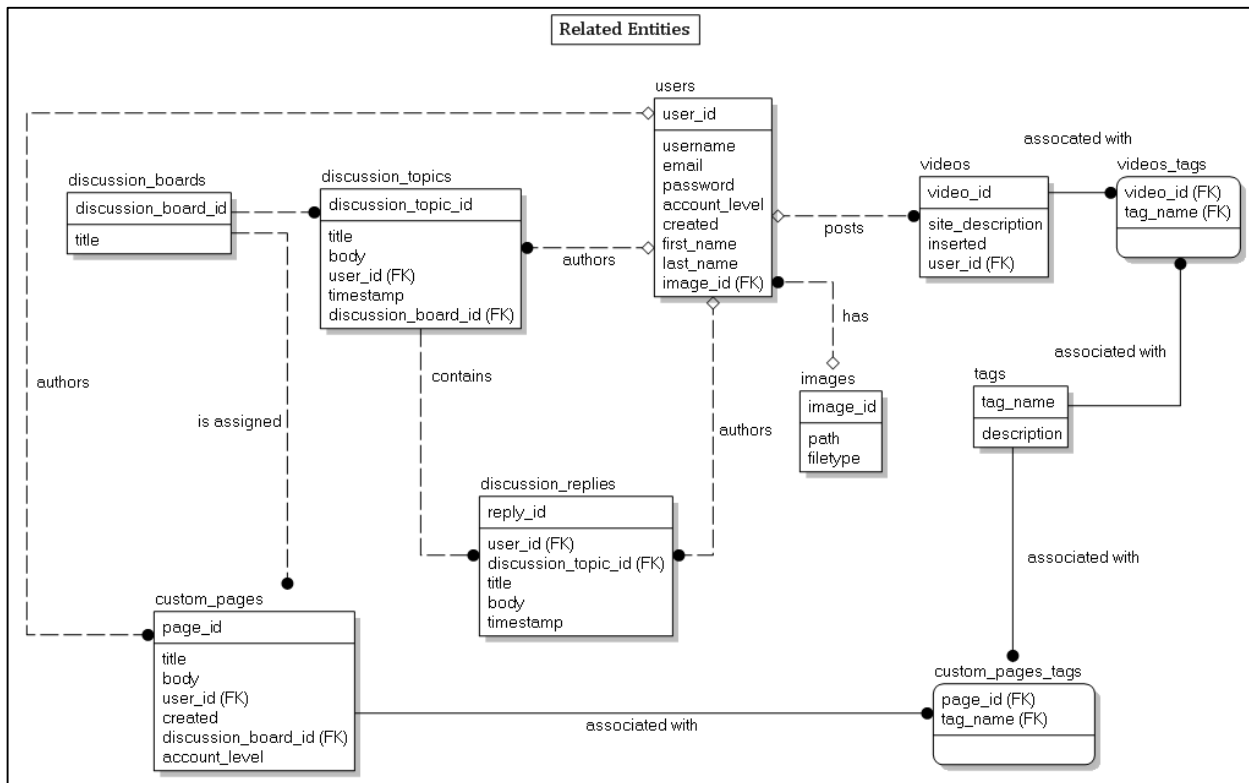
9.1 Appendix A: Project Schedule

Figure 6.1-1 Project Evaluation and Review Technique (PERT) Chart



9.2 Appendix B: Diagrams

Figure 6.2.0-1 Data Requirements (Shown as E-R Data Model Diagram in IDEF1X notation)



9.2.1 Basic Website Navigation Subsystem

Figure 6.2.1-1 Website Navigation Use Case Diagram

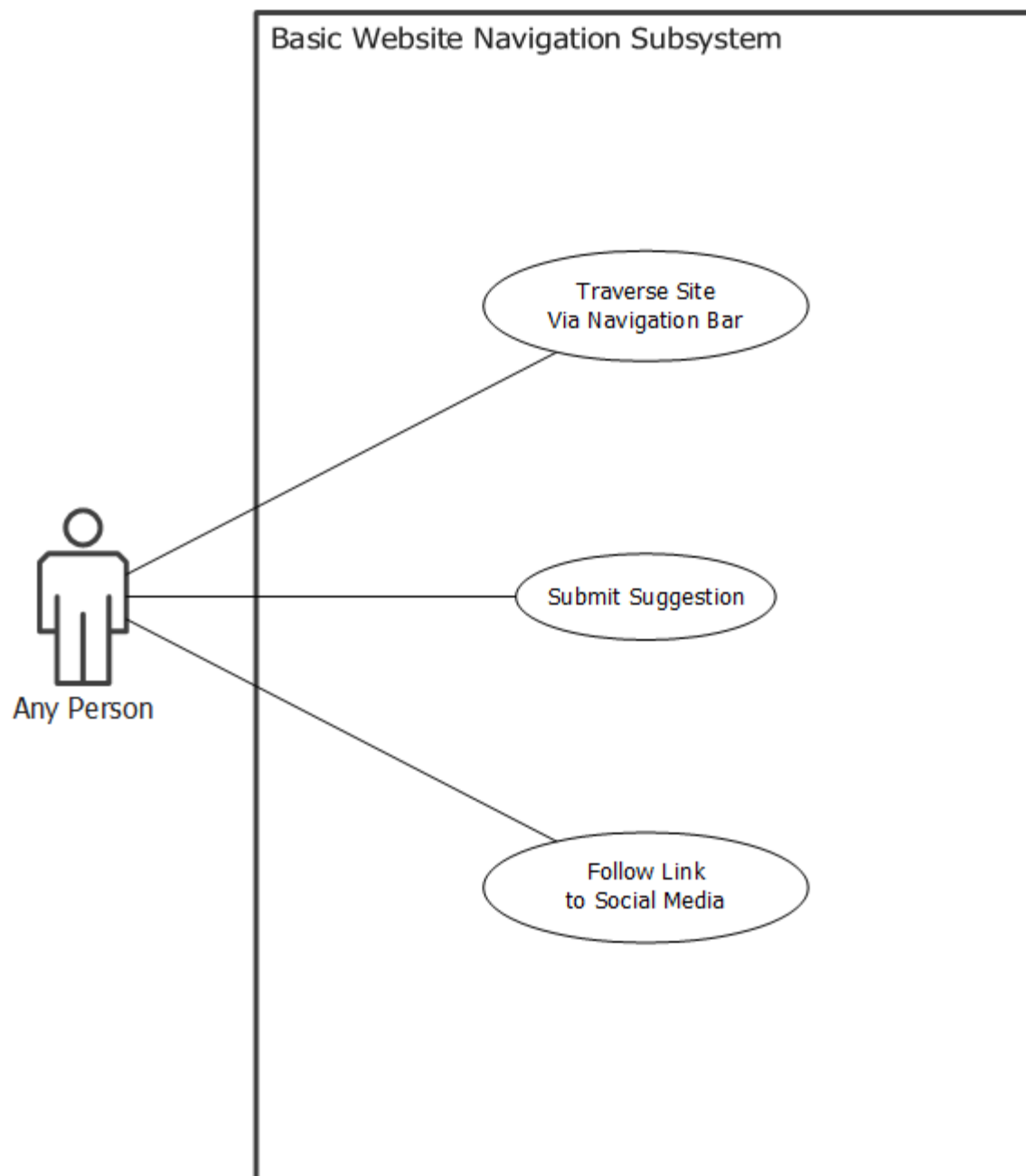
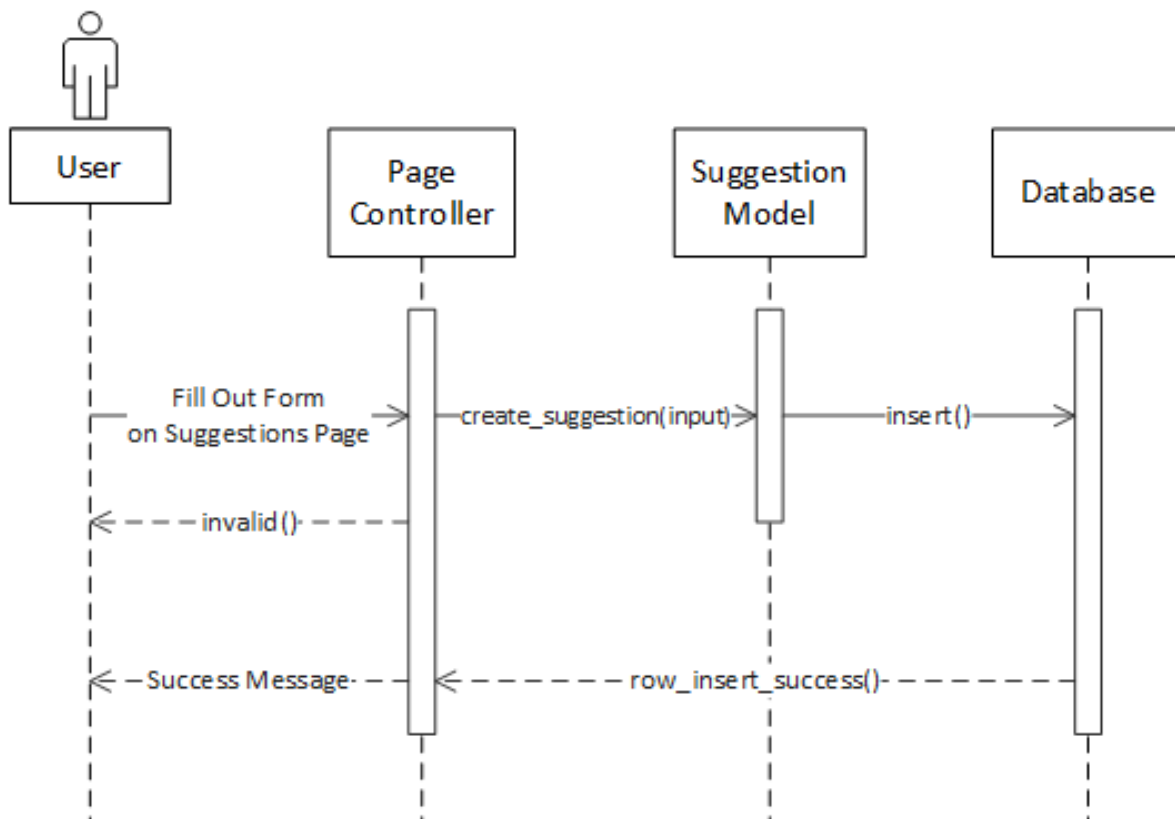


Figure 6.2.1-2 Suggestion Submission Sequence Diagram

9.2.2 Login Subsystem

Figure 6.2.2-1 Login Use Case Diagram

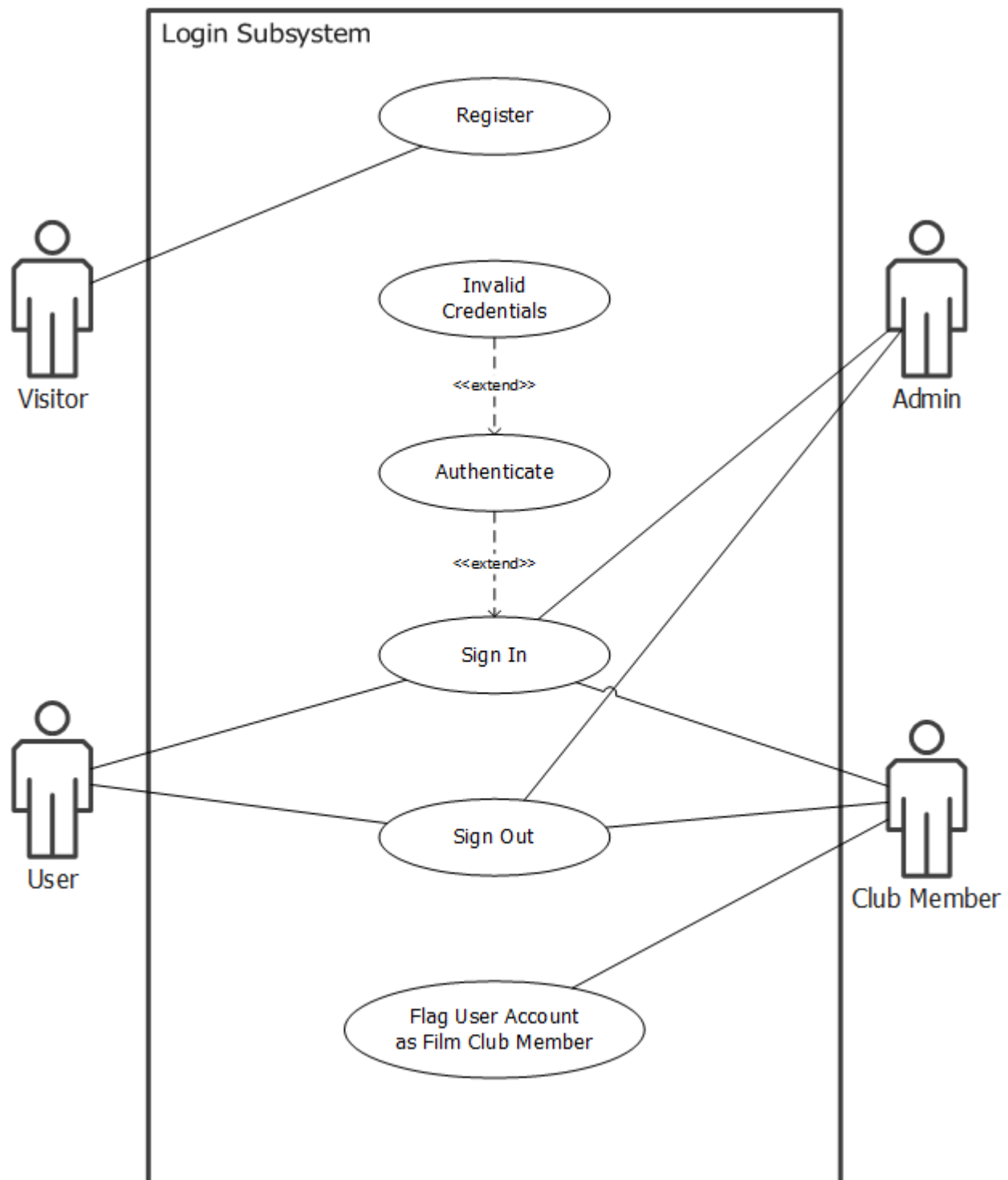
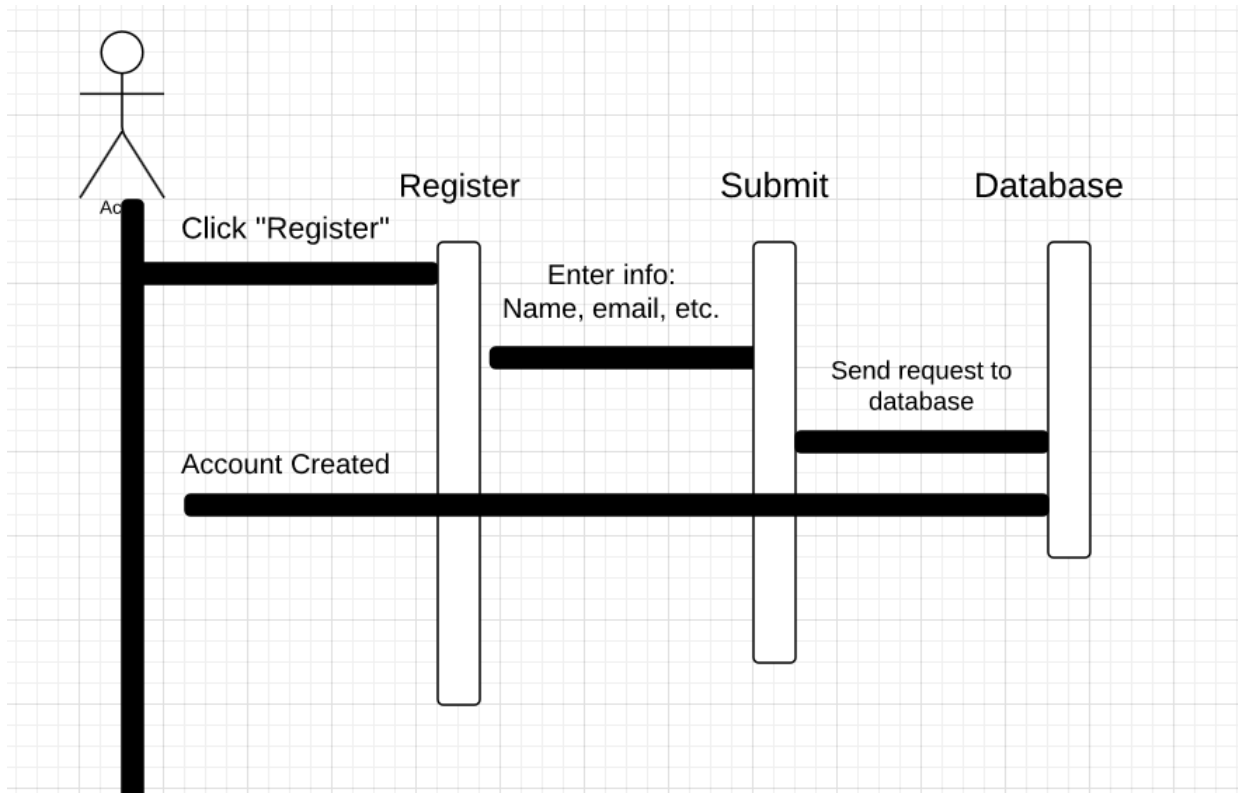


Figure 6.3.2-2 Registration Sequence Diagram

9.2.3 User Profile Subsystem

Figure 6.2.3-1 User Profile Use Case Diagram

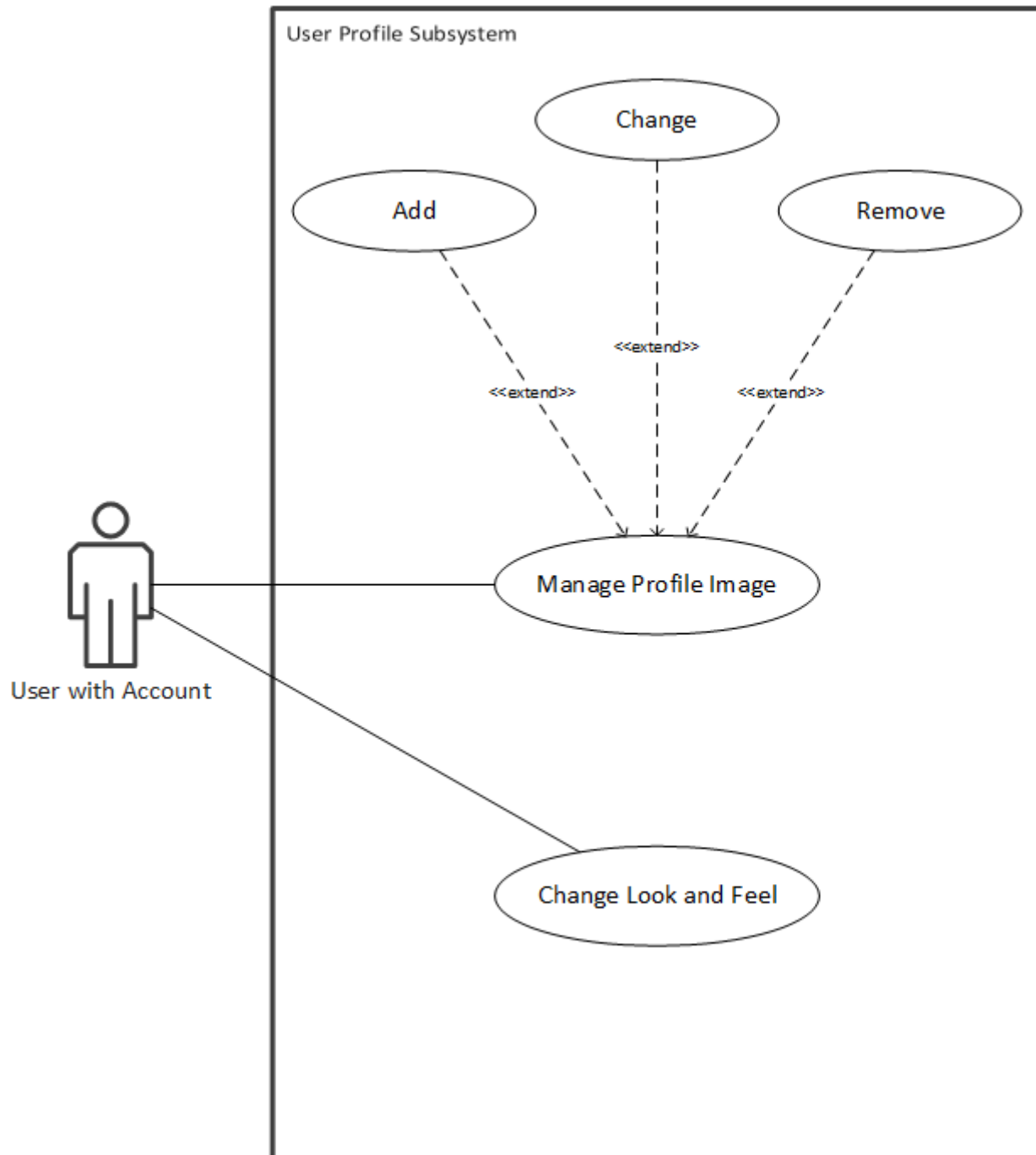
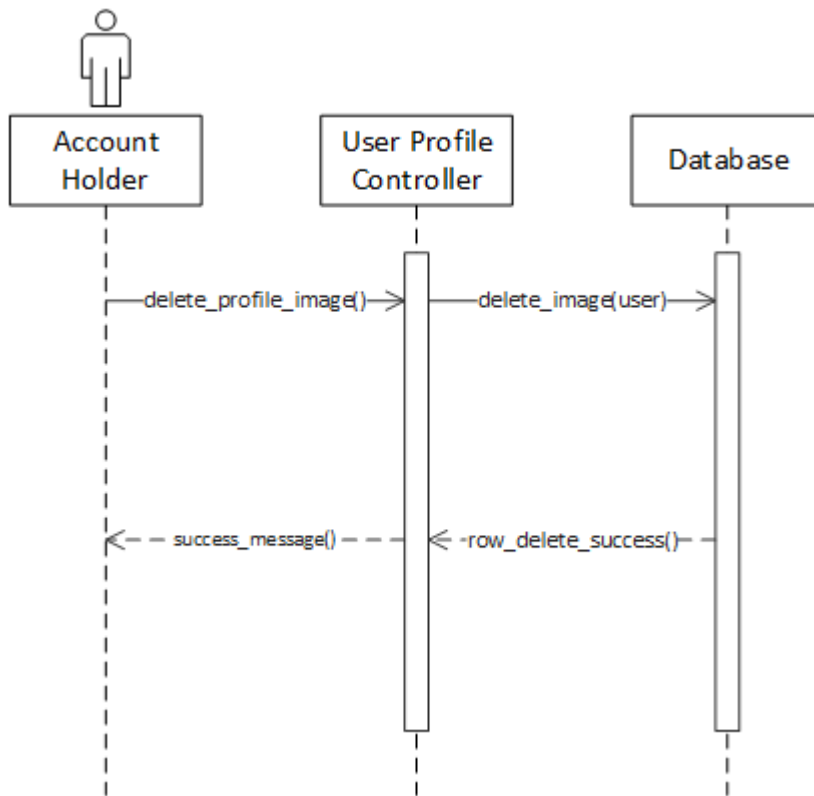


Figure 6.3.2-2 Profile Image Deletion

9.2.4 Administrative Tools Subsystem

Figure 6.3.4-1 Administrative Tools Use Case Diagram

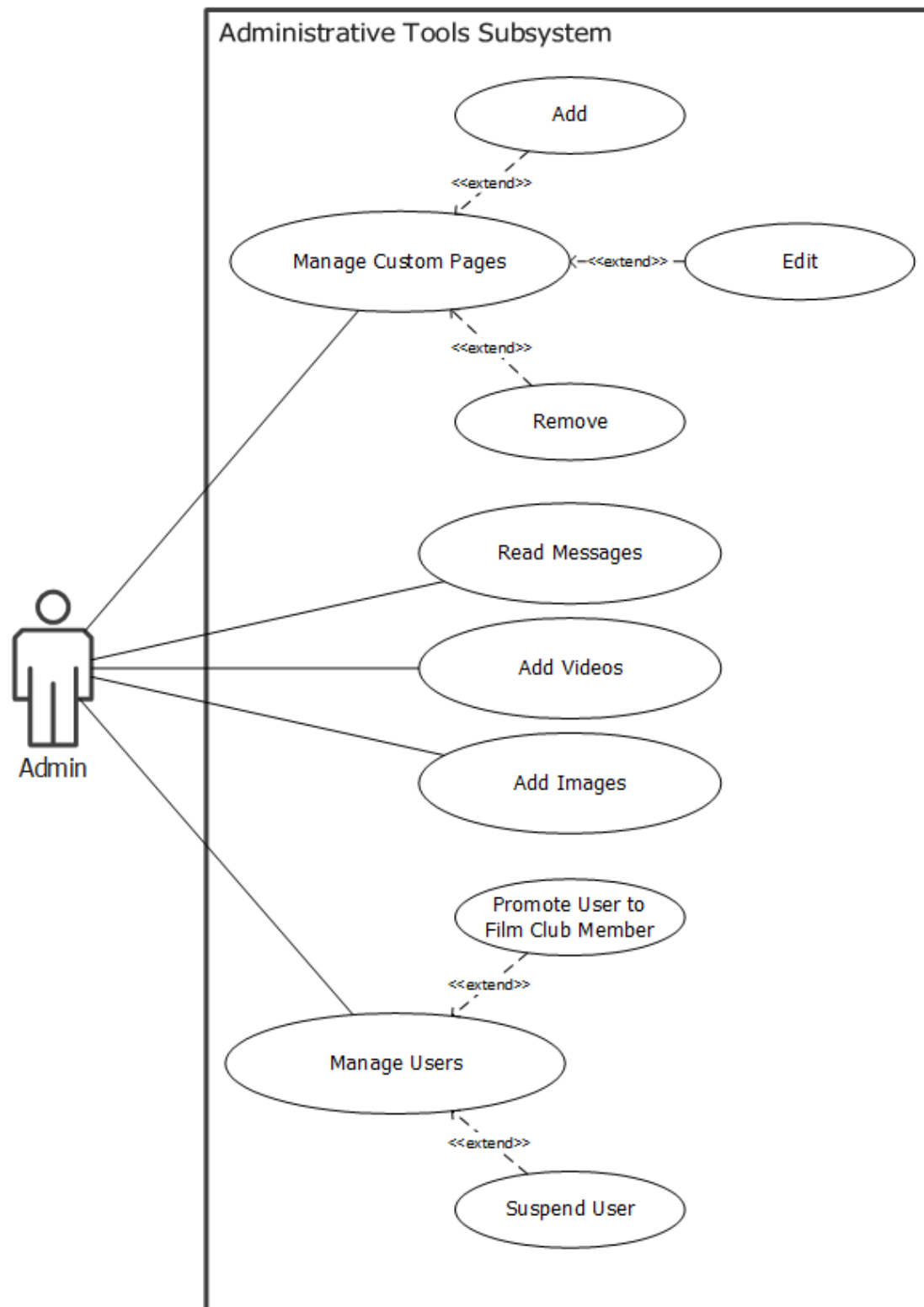
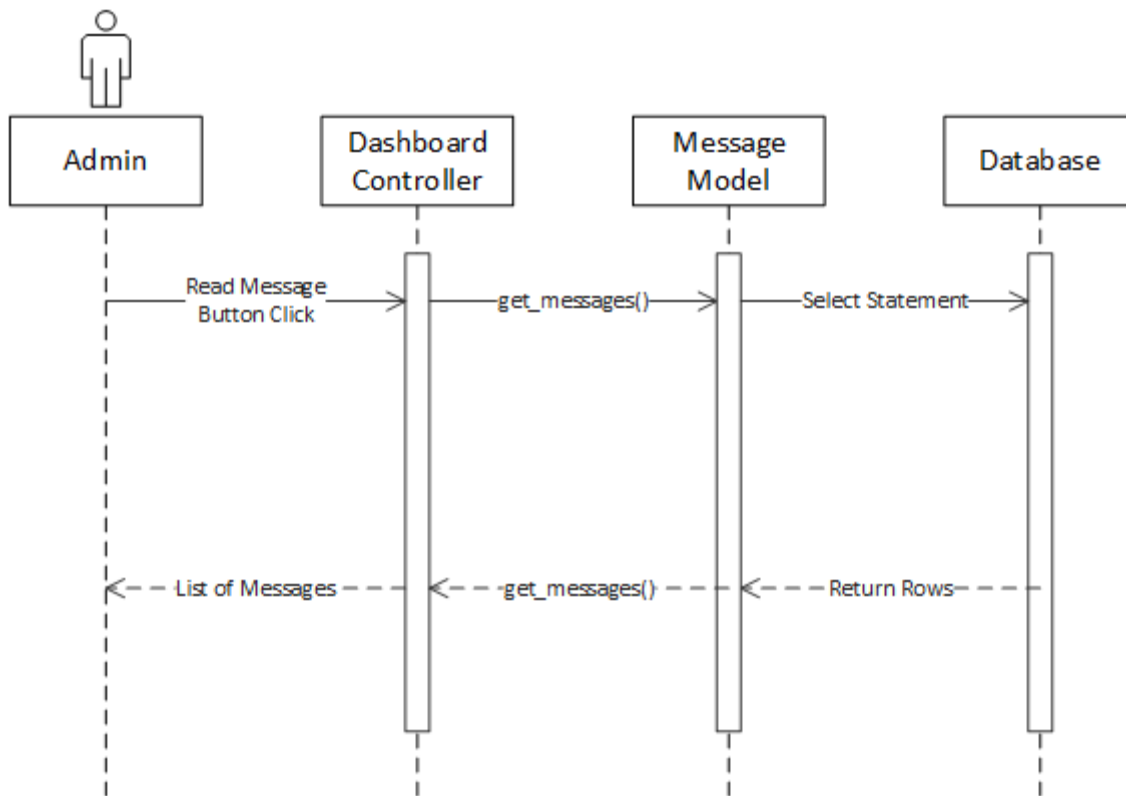


Figure 6.2.4-2 Read Administrative Messages Sequence Diagram

9.2.5 Discussion Subsystem

Figure 6.2.5-1 Discussion Use Case Diagram

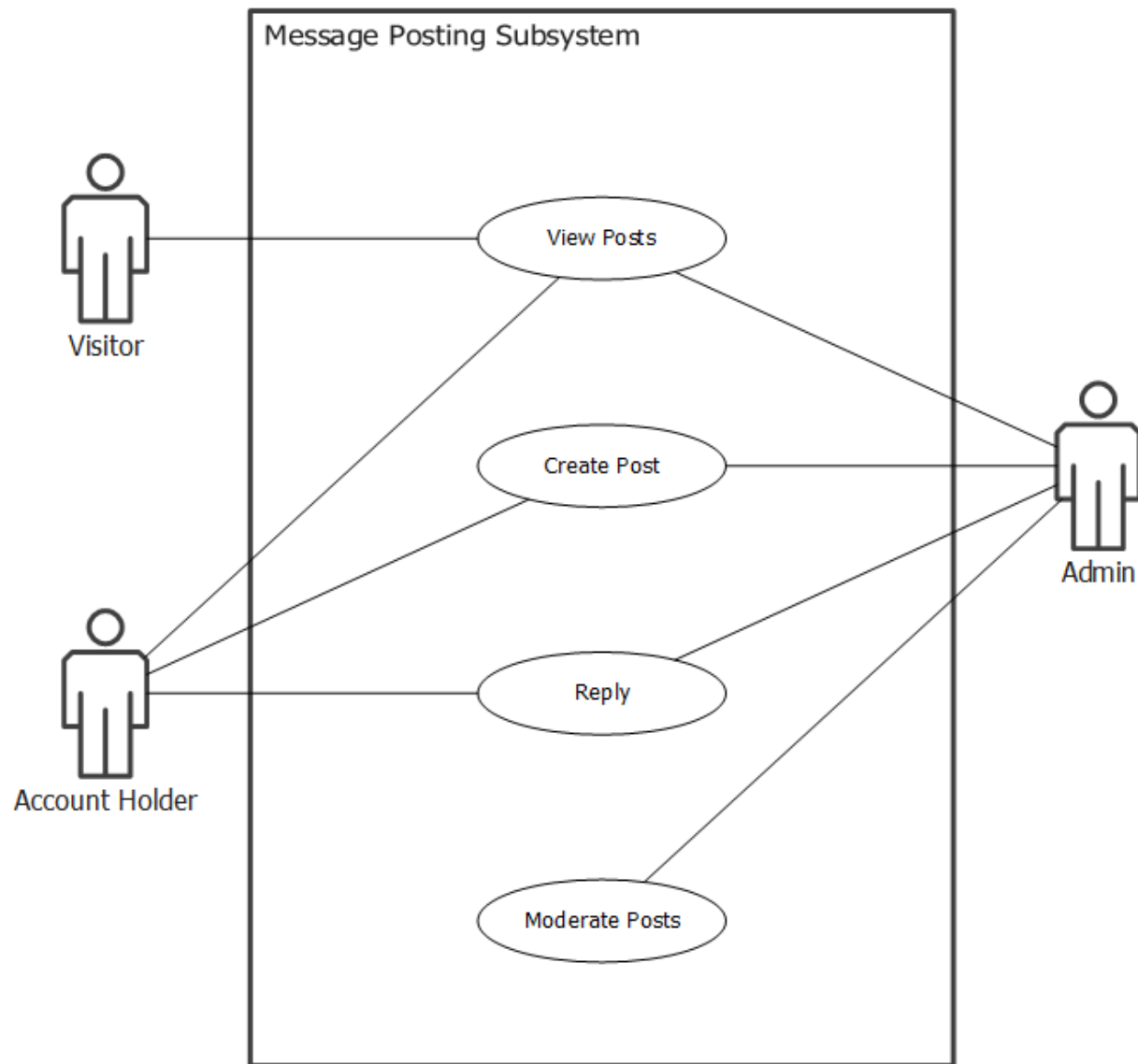
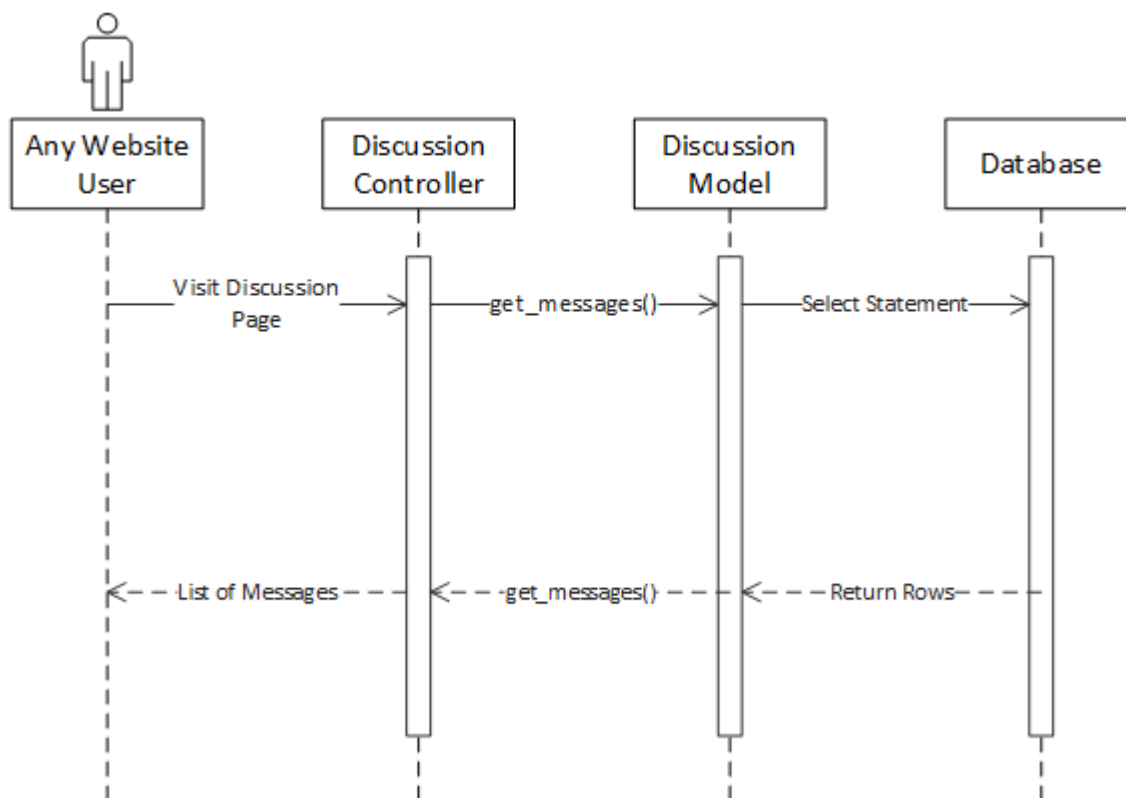


Figure 6.2.5-2 View Discussion Page Sequence Diagram

9.2.6 Calendar of Events Subsystem

Figure 6.2.6-1 Calendar of Events Use Case Diagram

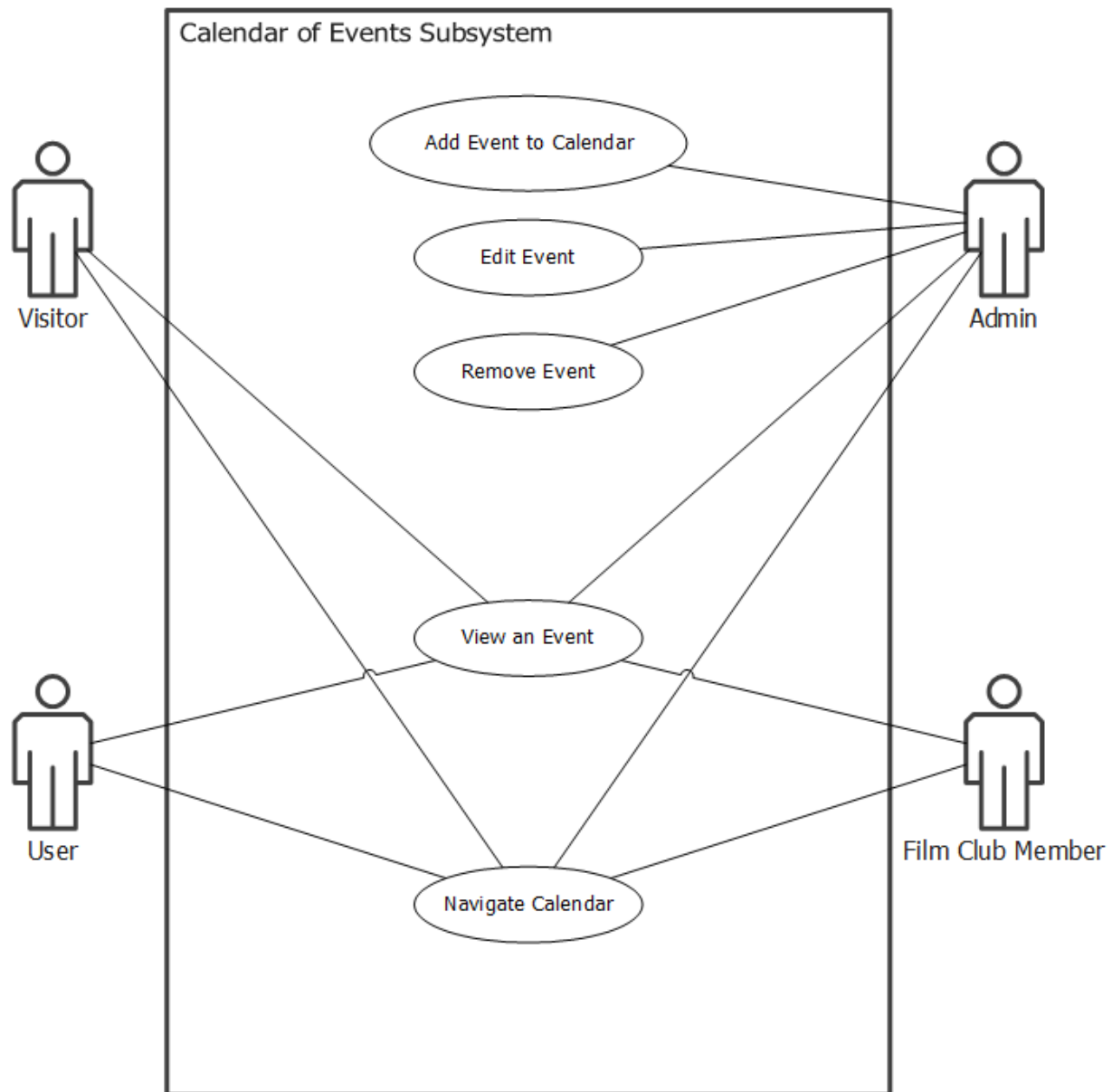
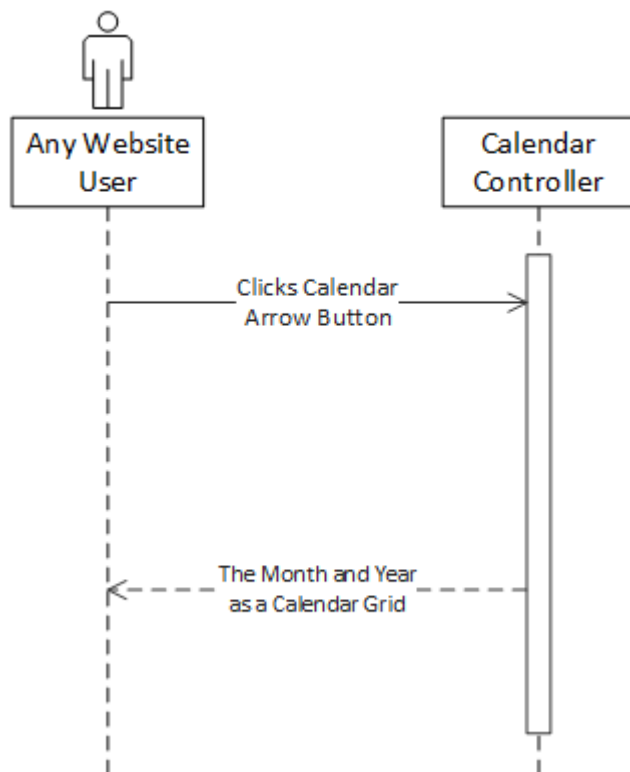


Figure 6.2.6-2 Calendar Navigation Sequence Diagram

9.3 Appendix C: User Interface Designs

9.3.1 Pages

Figure 6.3.1-1 Welcome Page



Figure 6.3.1-2 Navigation Bar



Figure 6.3.1-3 Navigation Bar Dropdown Menus

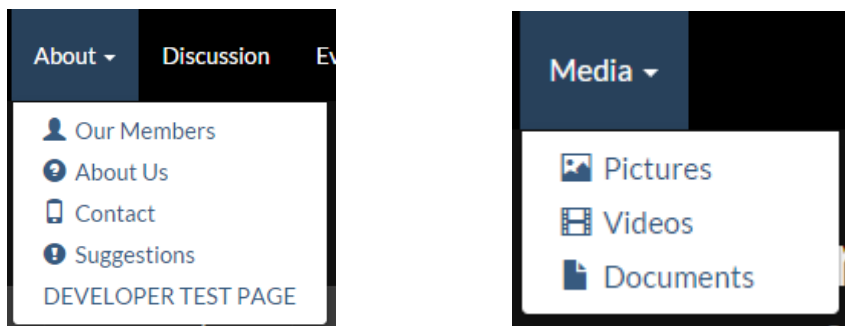


Figure 6.3.1-4 Home Page

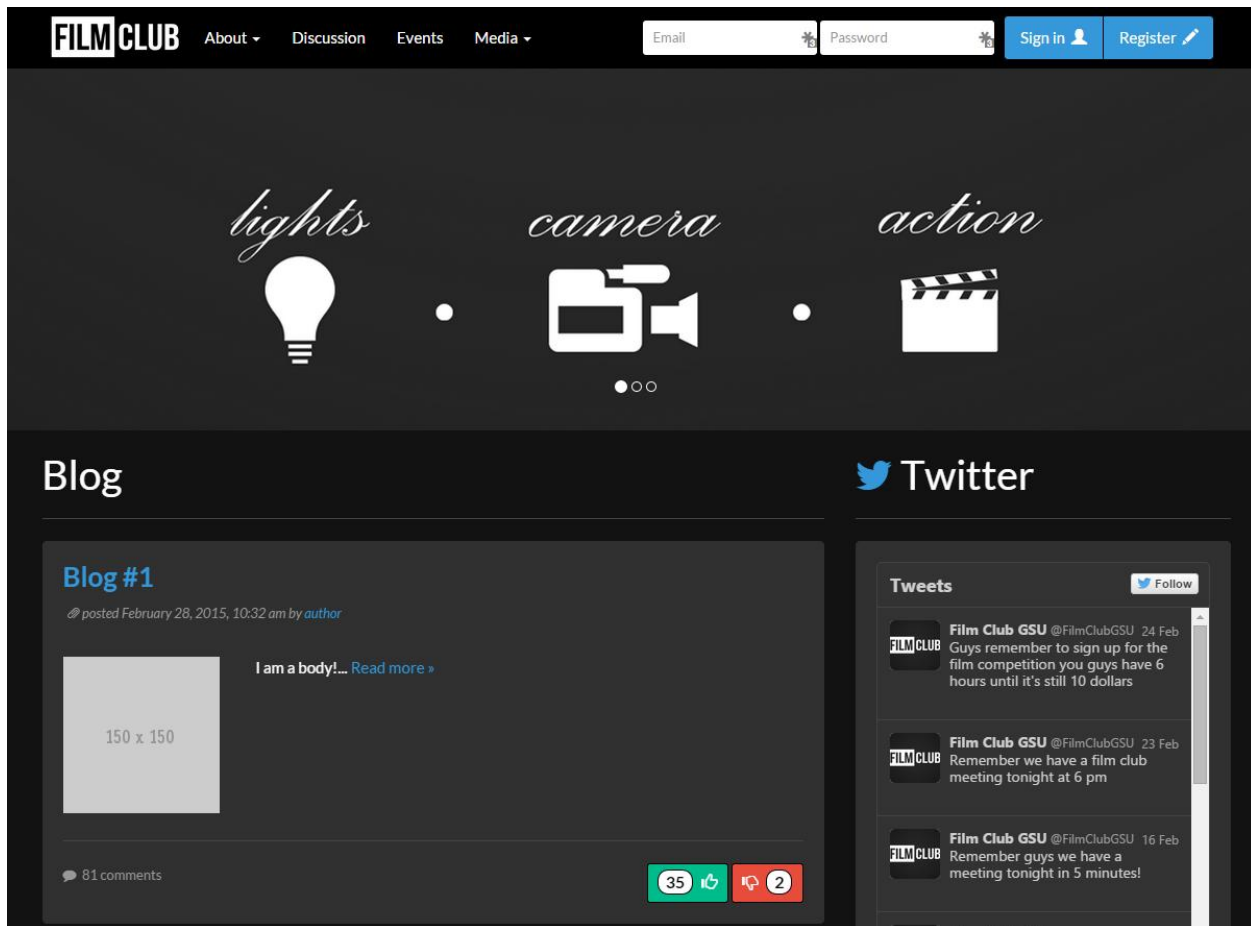


Figure 6.3.1-5 Page Footer

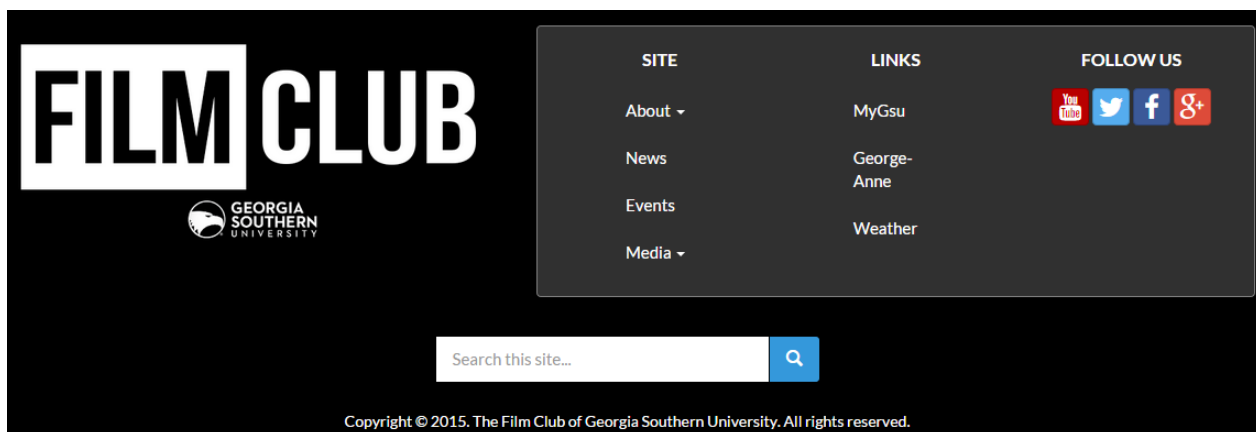
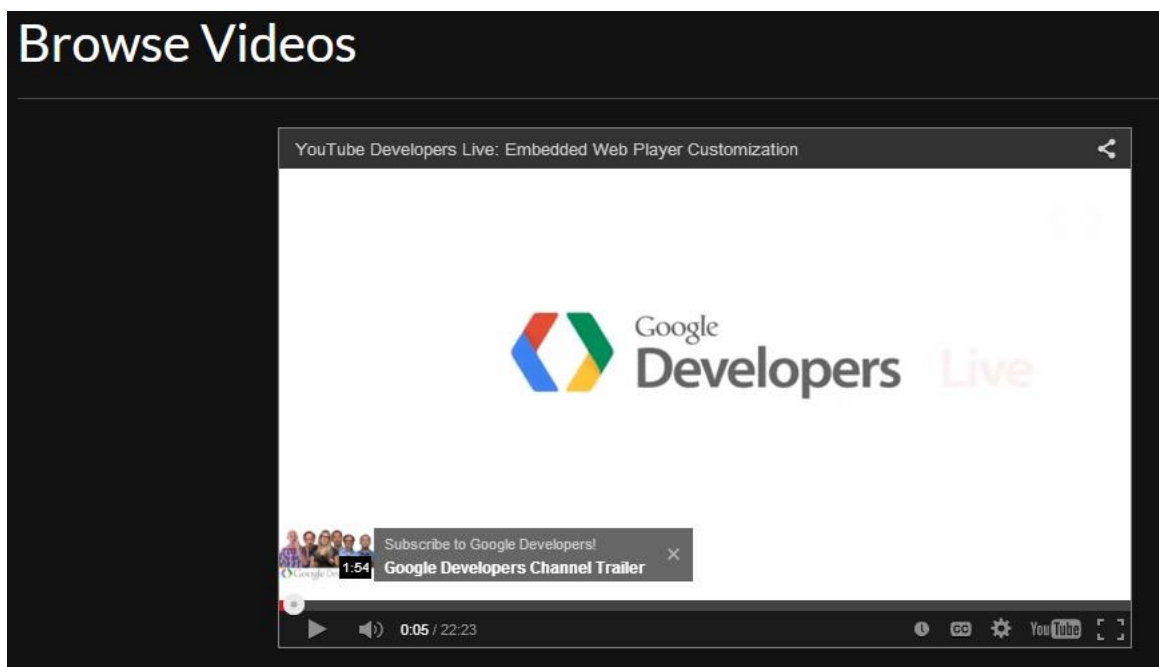


Figure 6.3.1-6 Calendar of Events

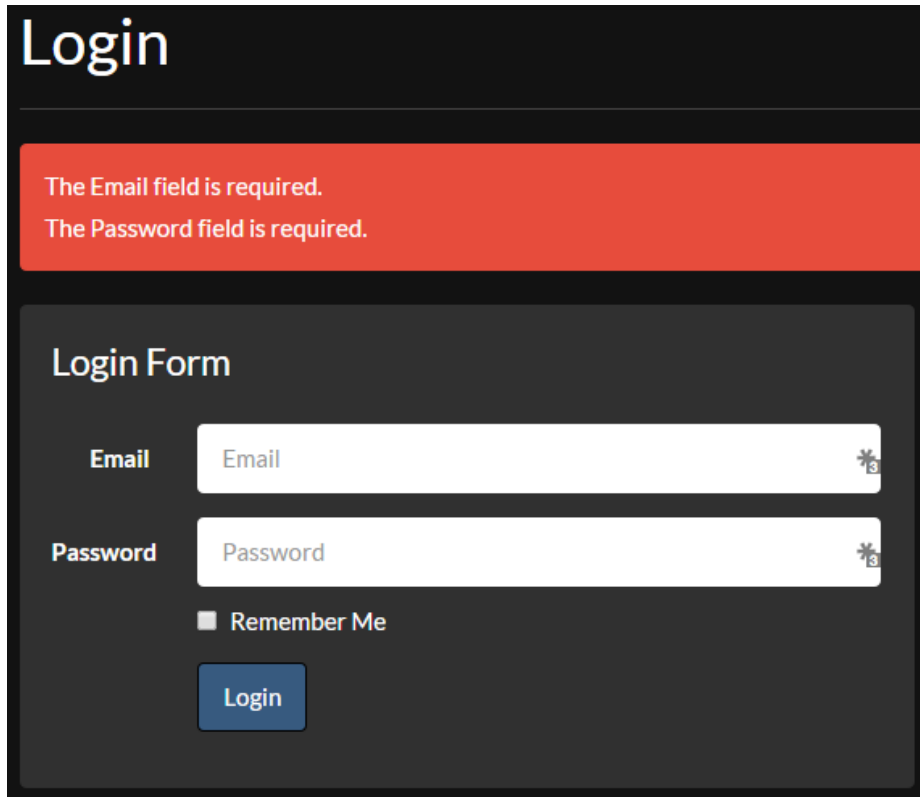
« February »						
2015						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

Figure 6.3.1-7 Videos



9.3.2 Forms

Figure 6.3.2-1 Sign-In Form



The image shows a 'Login' form interface. At the top, the word 'Login' is displayed in a large, white, sans-serif font against a dark background. Below this, a red banner contains two lines of white text: 'The Email field is required.' and 'The Password field is required.' The main form area has a dark gray background. It features a title 'Login Form' in white. There are two input fields: 'Email' and 'Password', both with white text and a small icon on the right. Below the 'Password' field is a checkbox labeled 'Remember Me'. At the bottom is a blue 'Login' button.

Login

The Email field is required.
The Password field is required.

Login Form

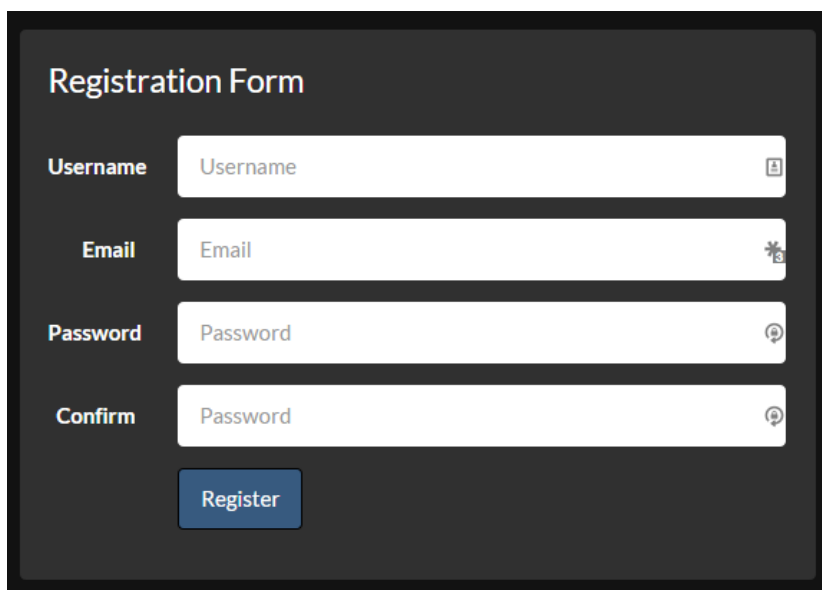
Email

Password

☐ Remember Me

Login

Figure 6.3.2-2 Registration Form



The image shows a 'Registration Form' interface. The title 'Registration Form' is in white at the top. There are four input fields: 'Username', 'Email', 'Password', and 'Confirm', each with a small icon on the right. Below the 'Confirm' field is a blue 'Register' button.

Registration Form

Username

Email

Password

Confirm

Register

Figure 6.3.2-3 Suggestions Form

Suggestions

Have a suggestion? Want to work with us? Tell us here!

Contact Information

First Name

Last Name

Email

Phone

Suggestion Information

Subject

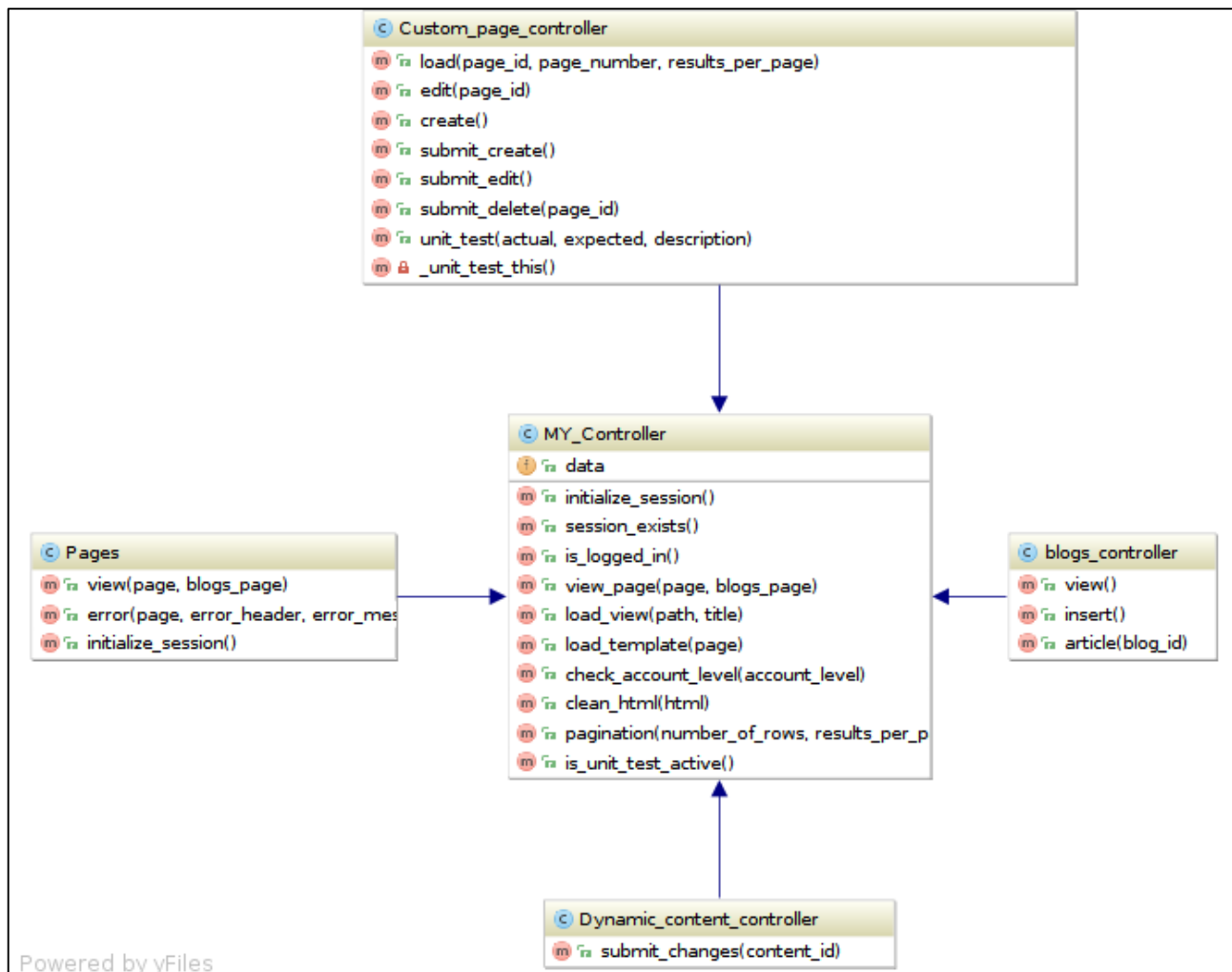
Message

9.4 Class Diagrams

Below are UML class diagrams showing attributes and methods for each class grouped by packages. Due to the size of the packages a representative sample of the controller and models classes have been provided.

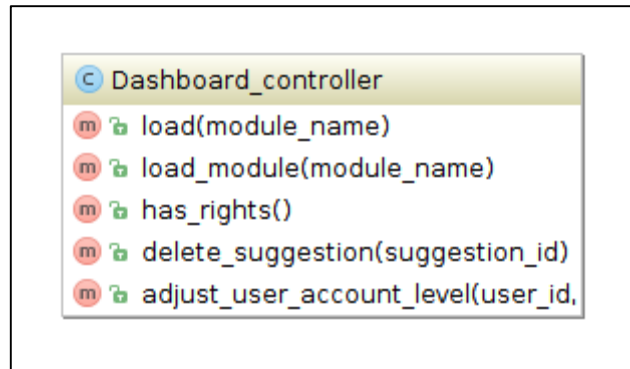
9.4.1 Content Package Class Diagram

Representative sample of the classes within the content package



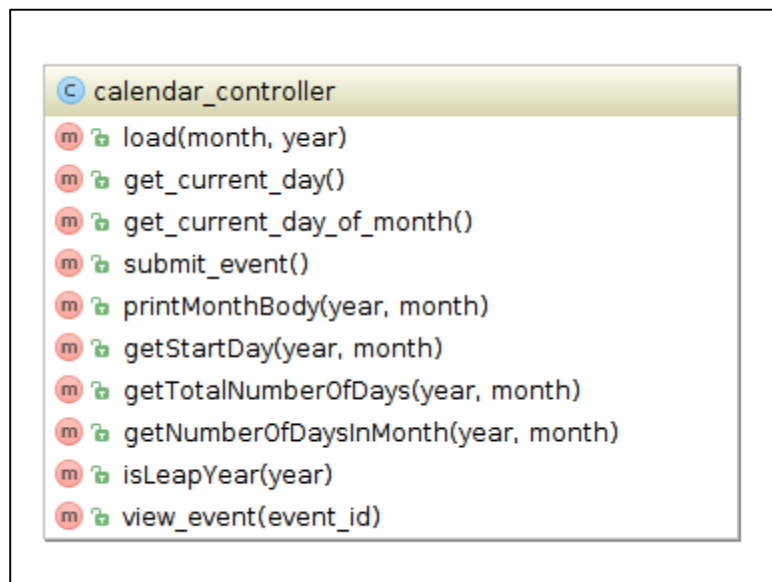
9.4.2 Administrator Dashboard Package

The dashboard controller within the dashboard package.



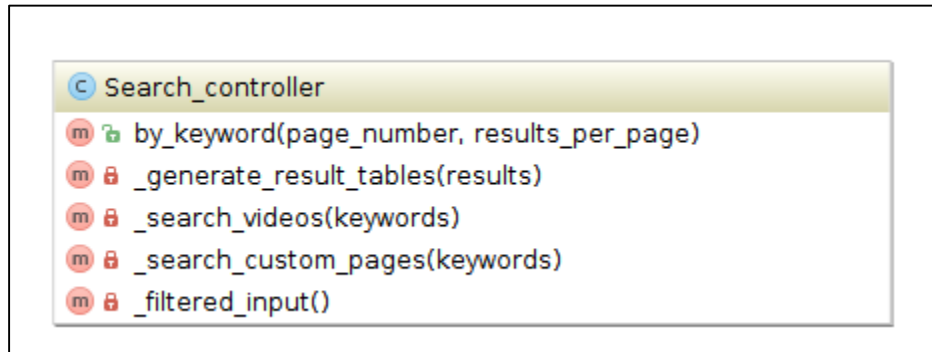
9.4.3 Calendar Package

The calendar controller within the calendar package.



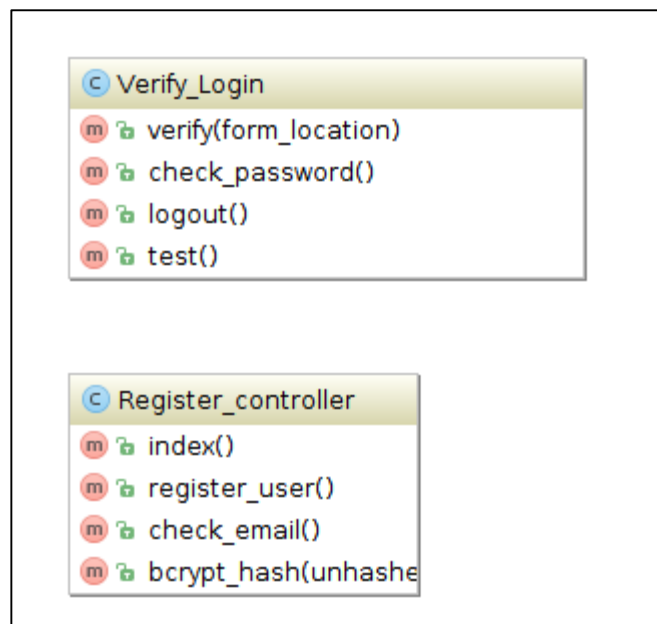
9.4.4 Search Package

The search controller within the search package.



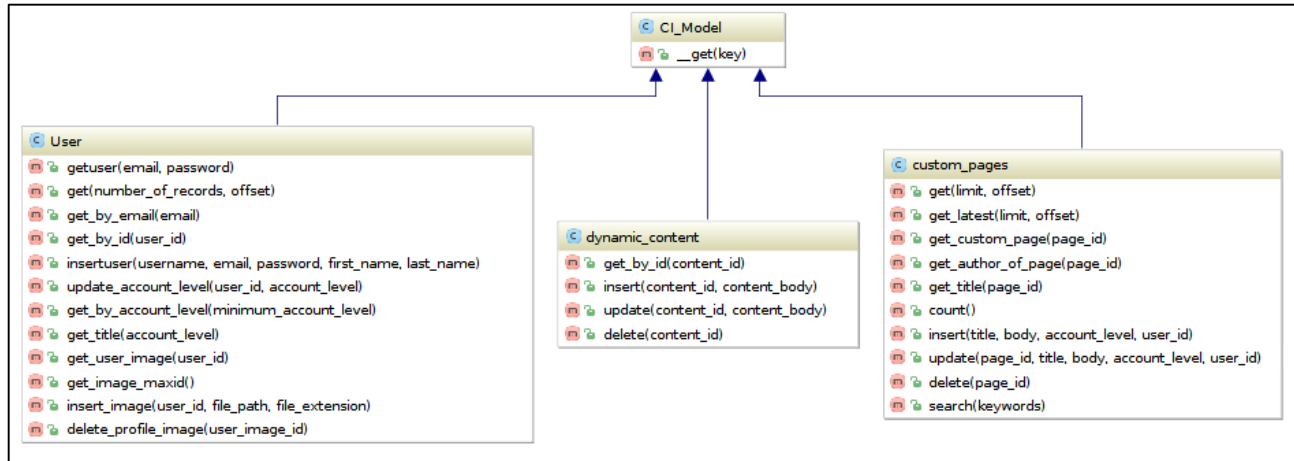
9.4.5 Login Package

Class diagram of the login package controllers.



9.4.6 Package Models

A representative sample of the model classes in the Film Club CMS. The User model is part of the login package and the dynamic_content and custom_pages classes belong to the content package.



9.5 Class Interfaces

Please refer to the Film Club CMS API pages generated in phpDoc.

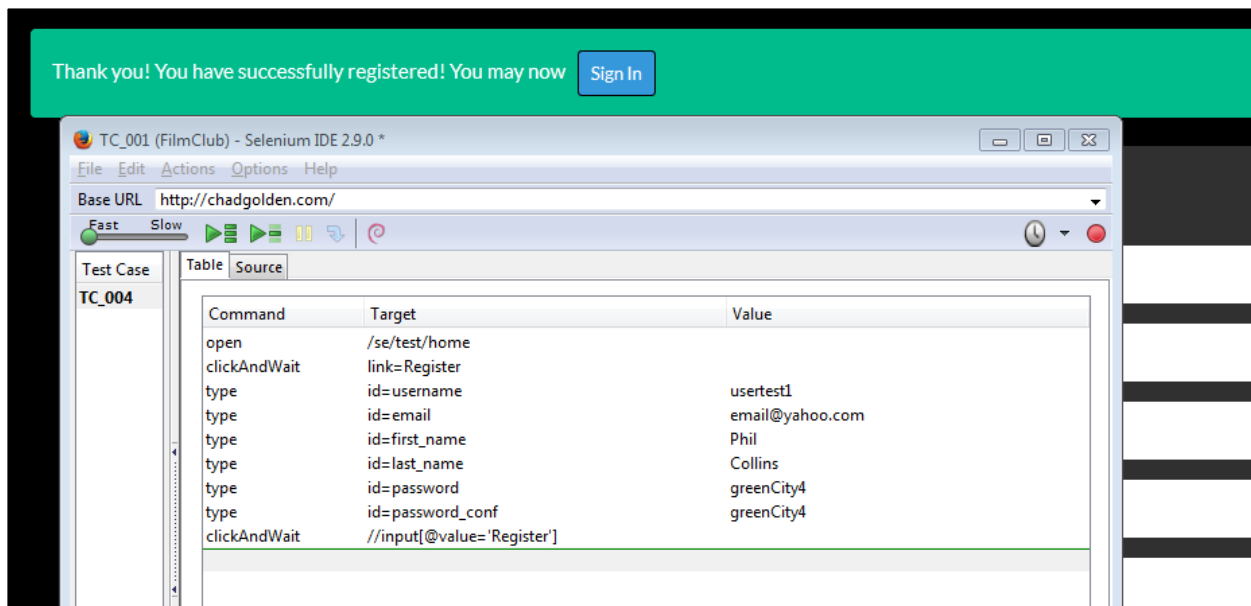
9.6 Testing

Please refer to the Film Club CMS API pages generated in phpDoc for the documented code for the testing driver.

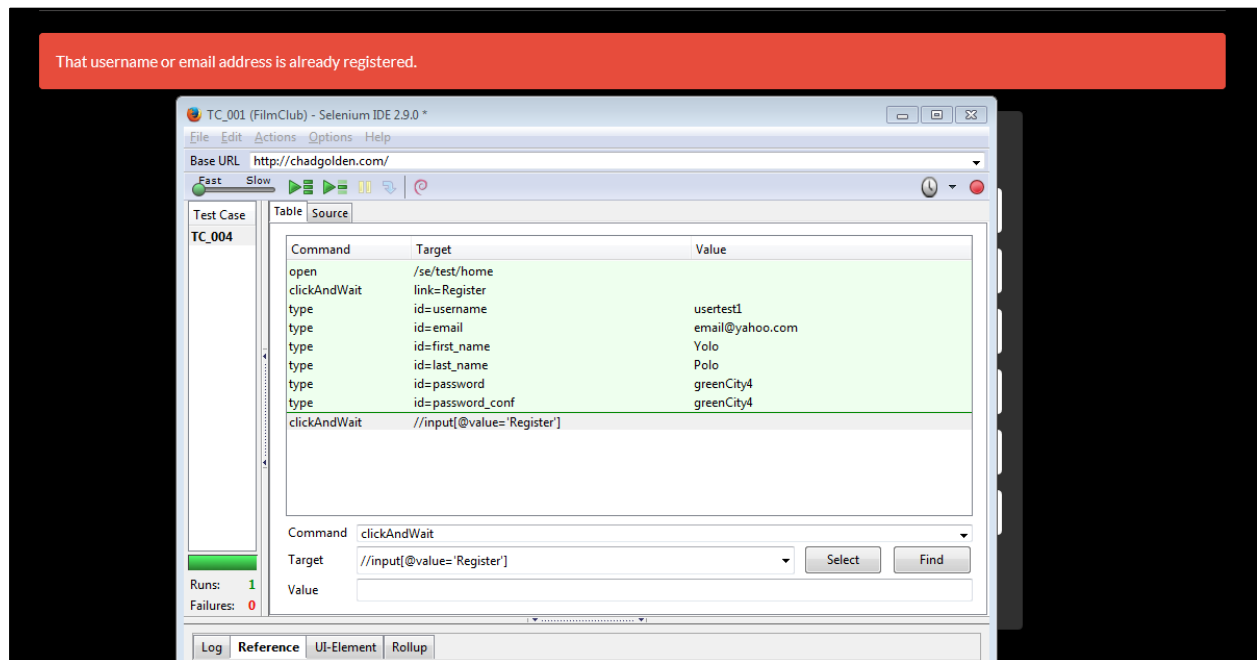
9.6.1 Selenium Tests & Screens

This section contains screen captures saved from the testing process. Note that not all test screens are listed.

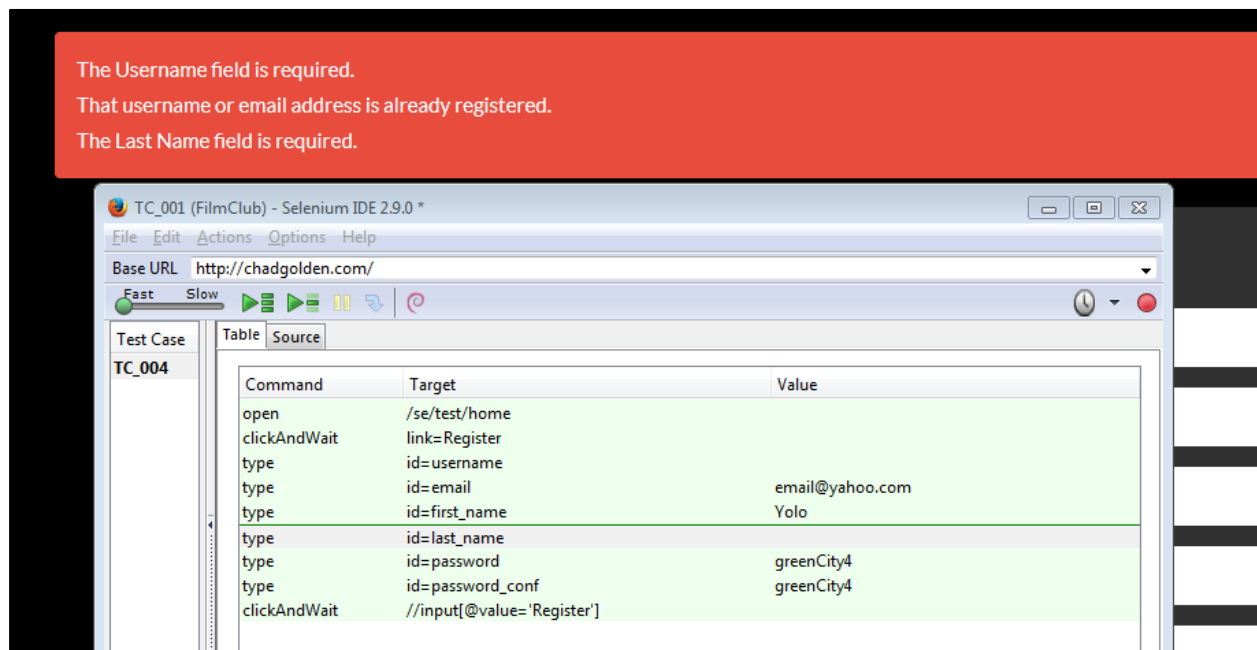
TC_001_Test1



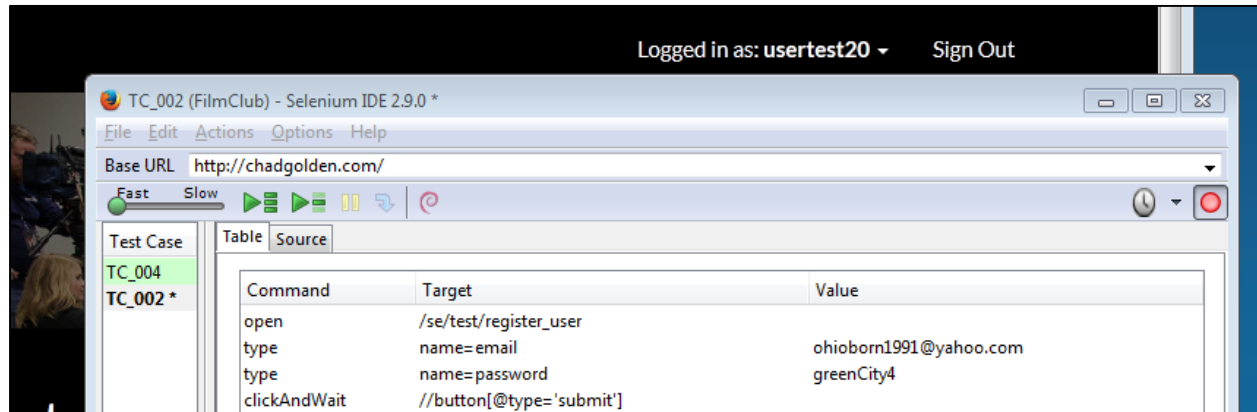
TC_001_Test2



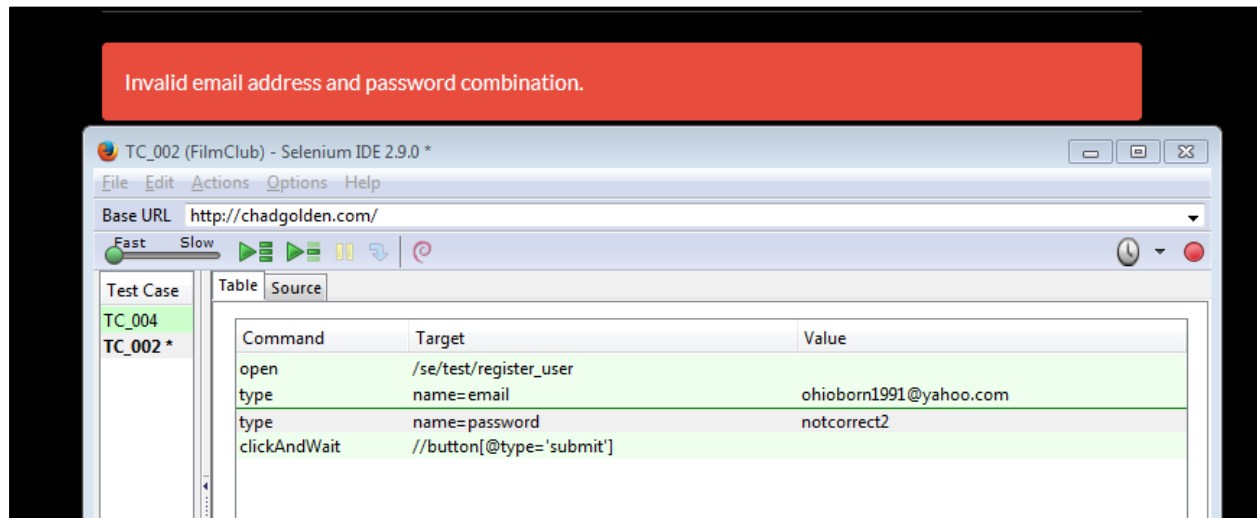
TC_001_Test3



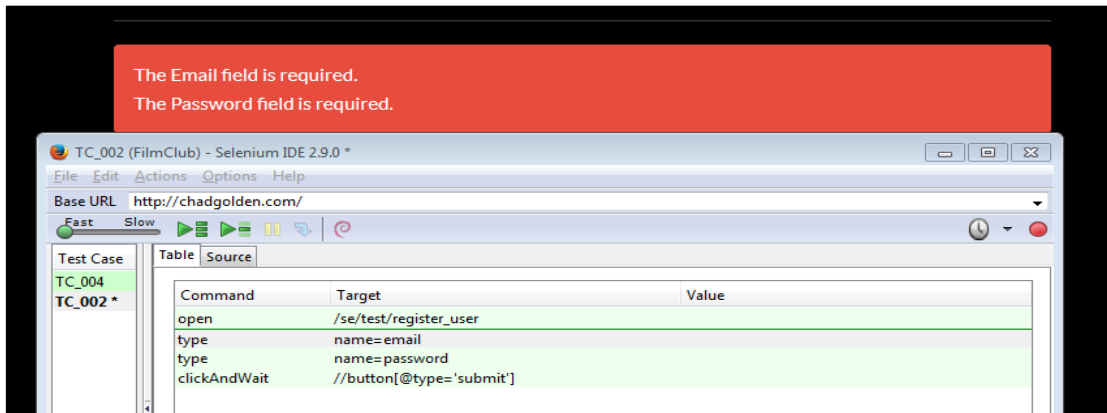
TC_002_Test1



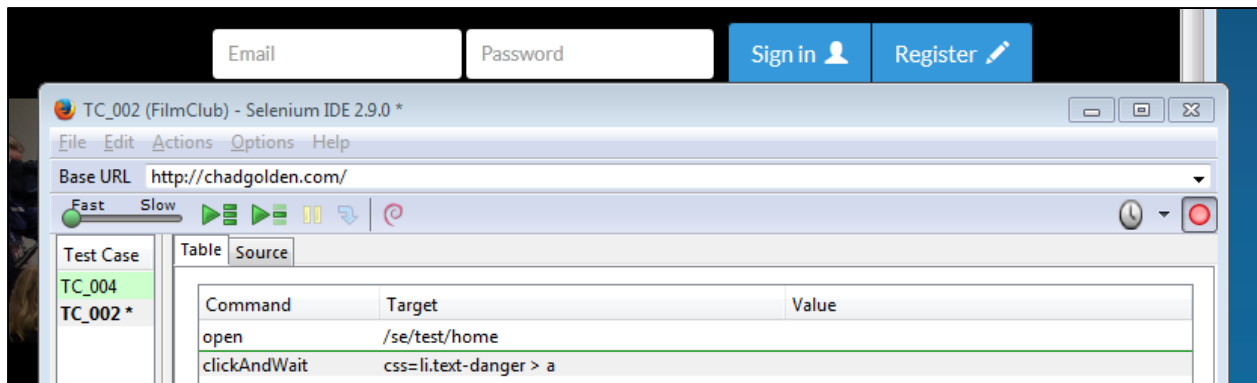
TC_002_Test2



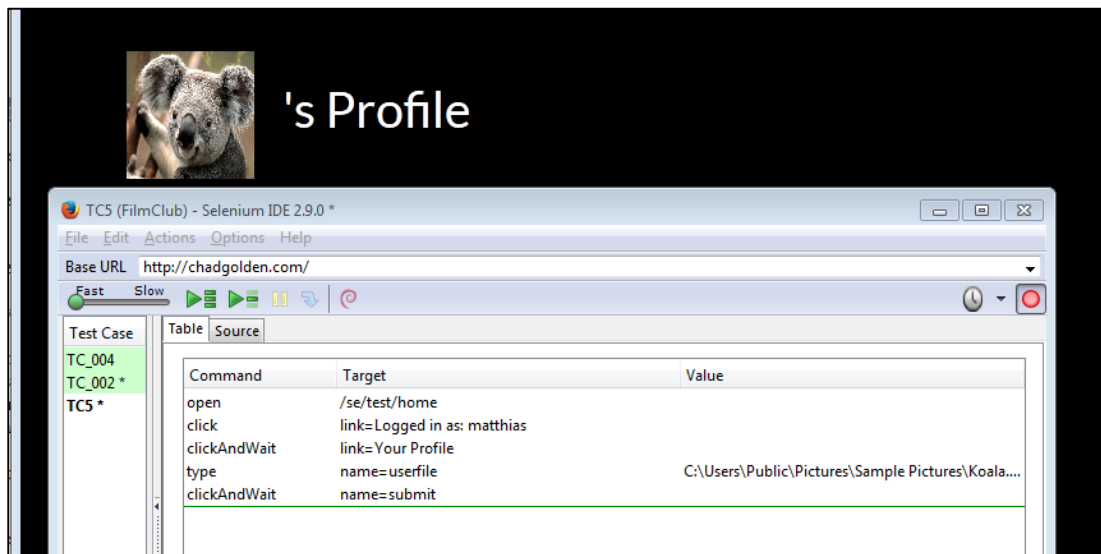
TC_002_Test3



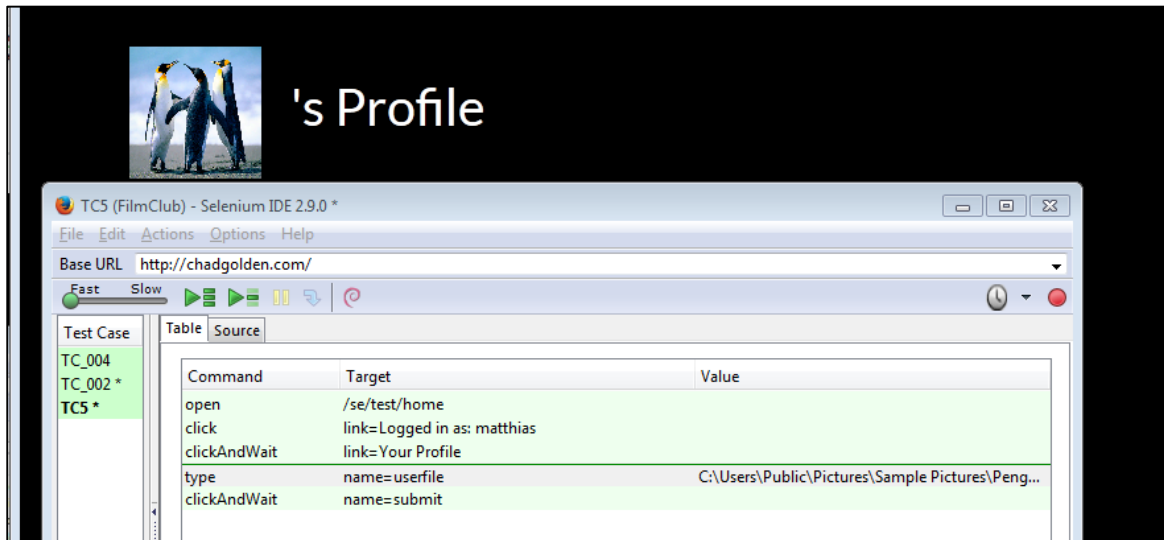
TC_003_Test1



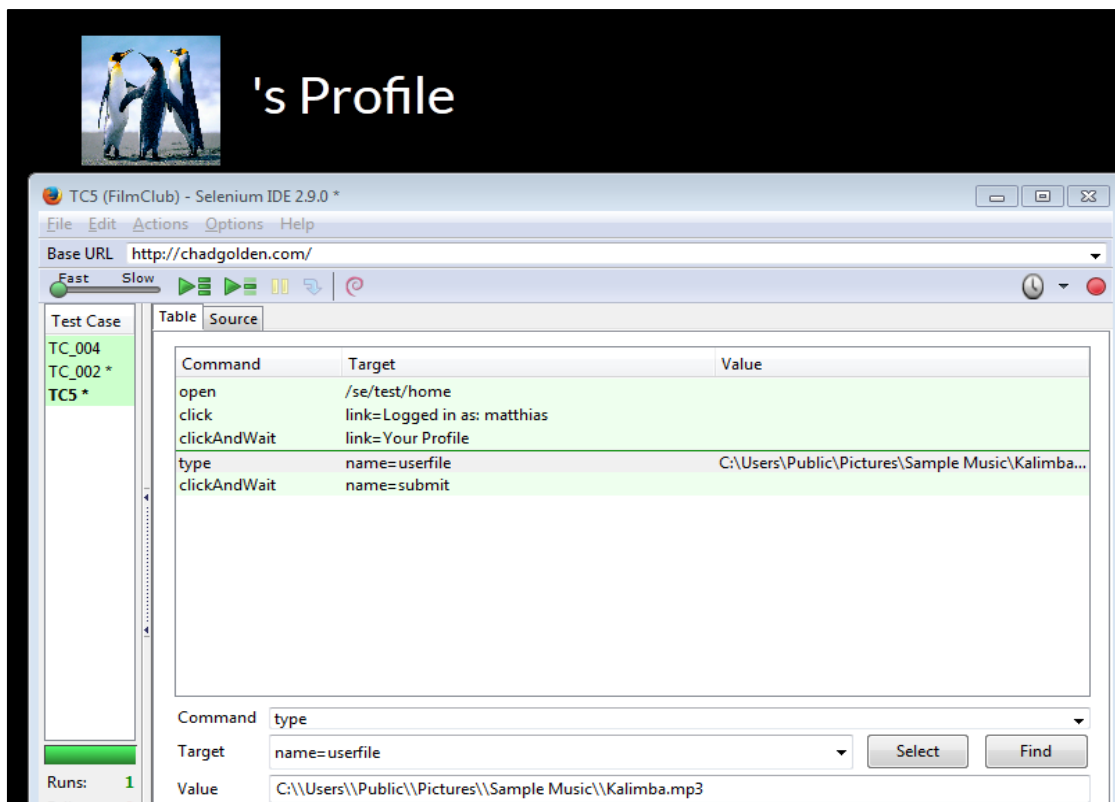
TC_005_Test1



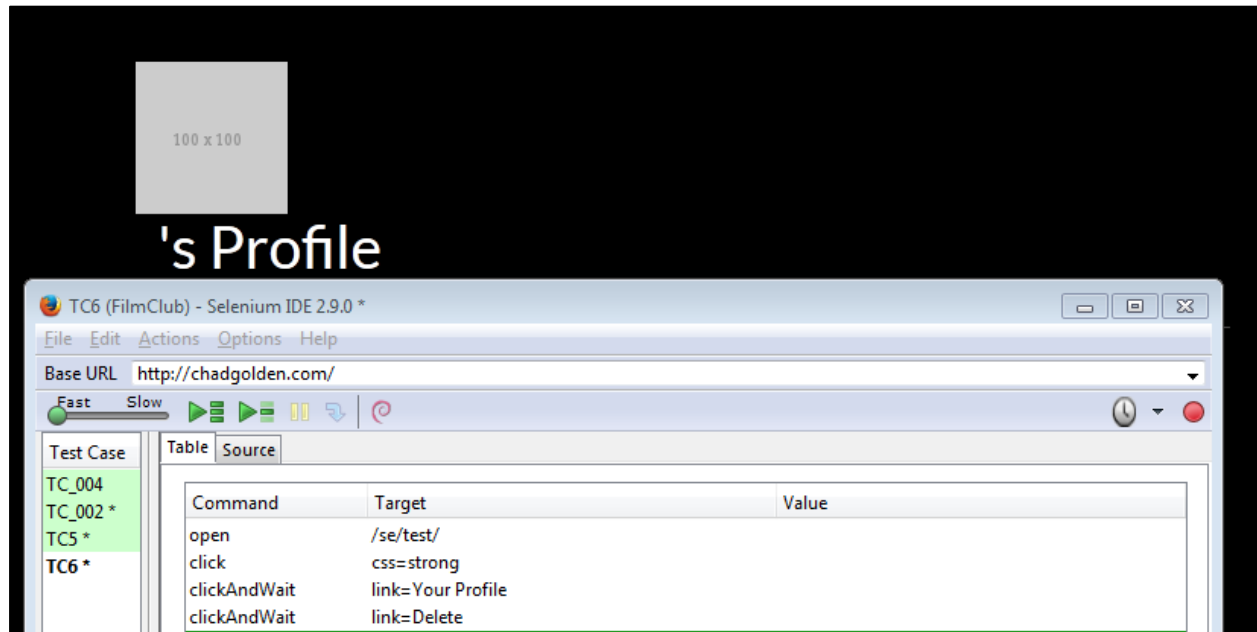
TC_005_Test2



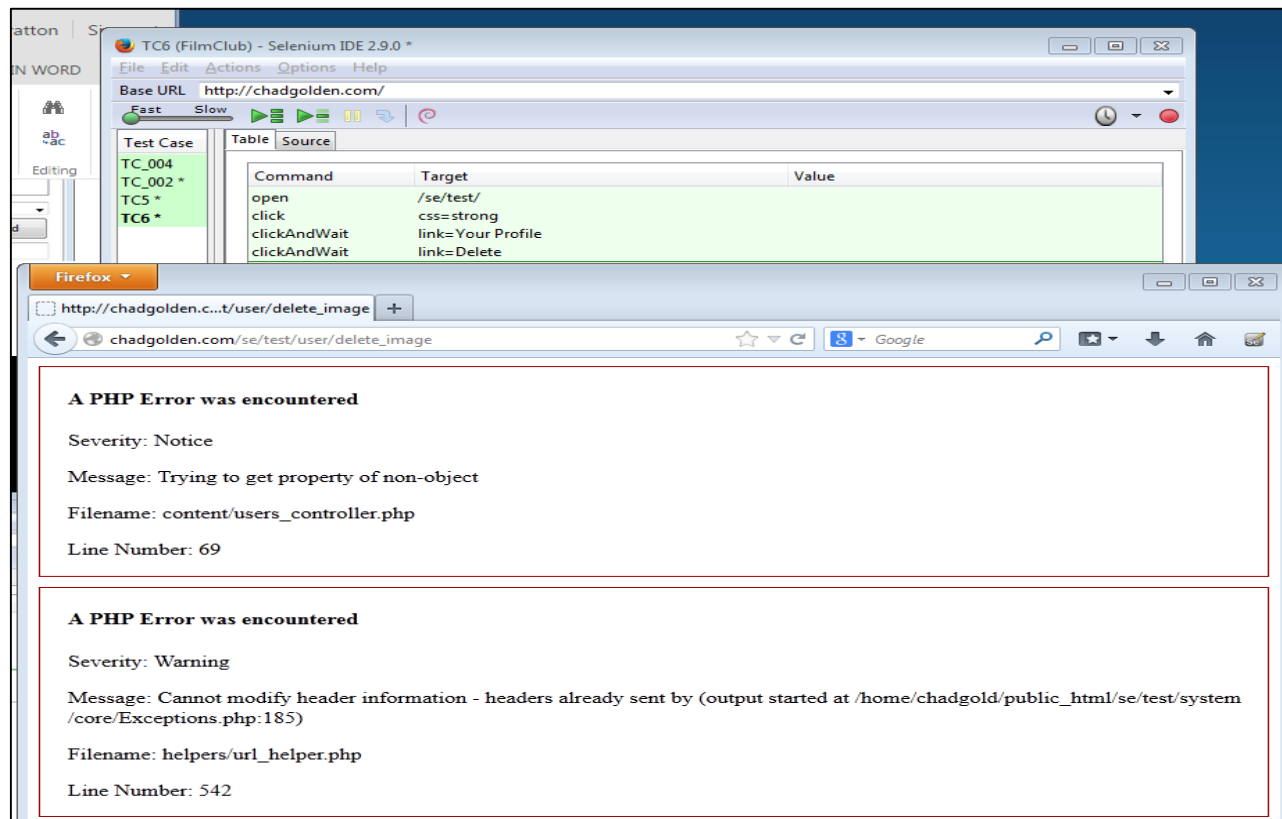
TC_005_Test3



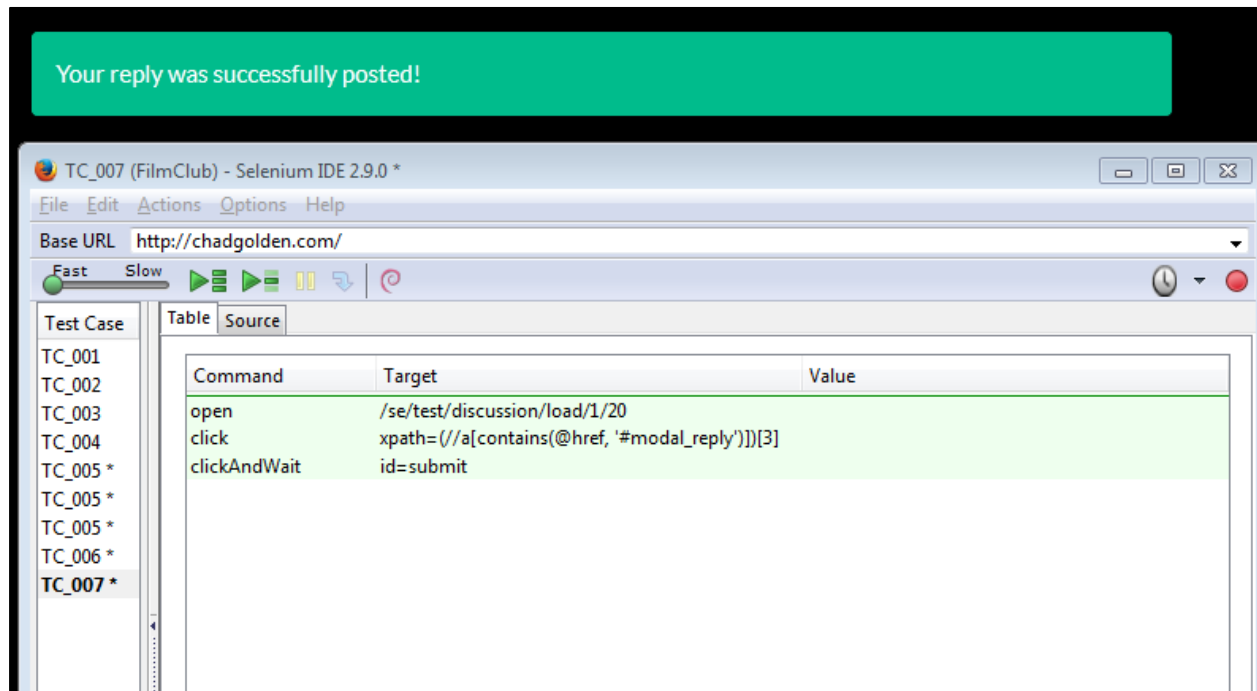
TC_006_Test1



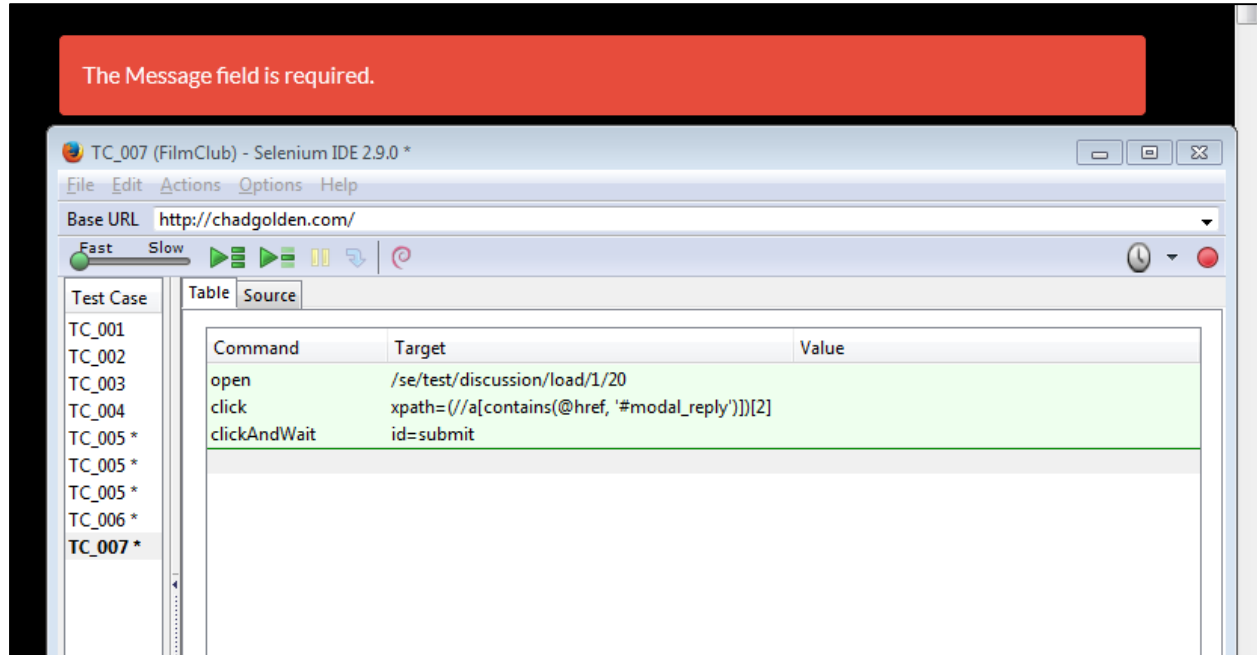
TC_006_Test2



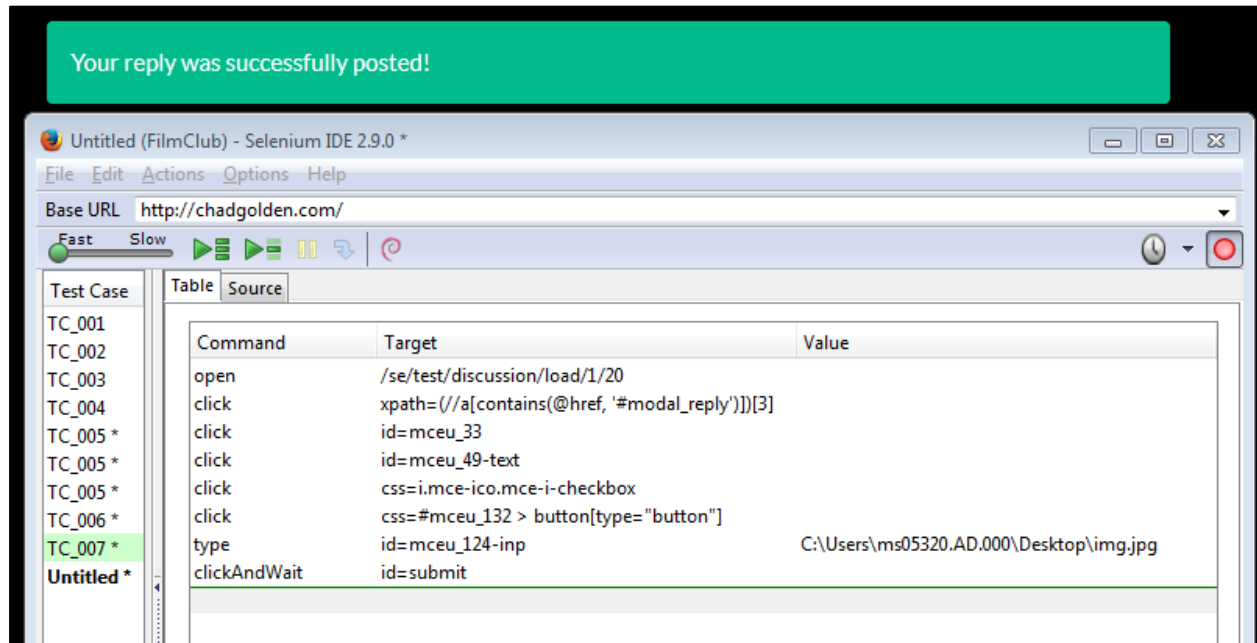
TC_007_Test1



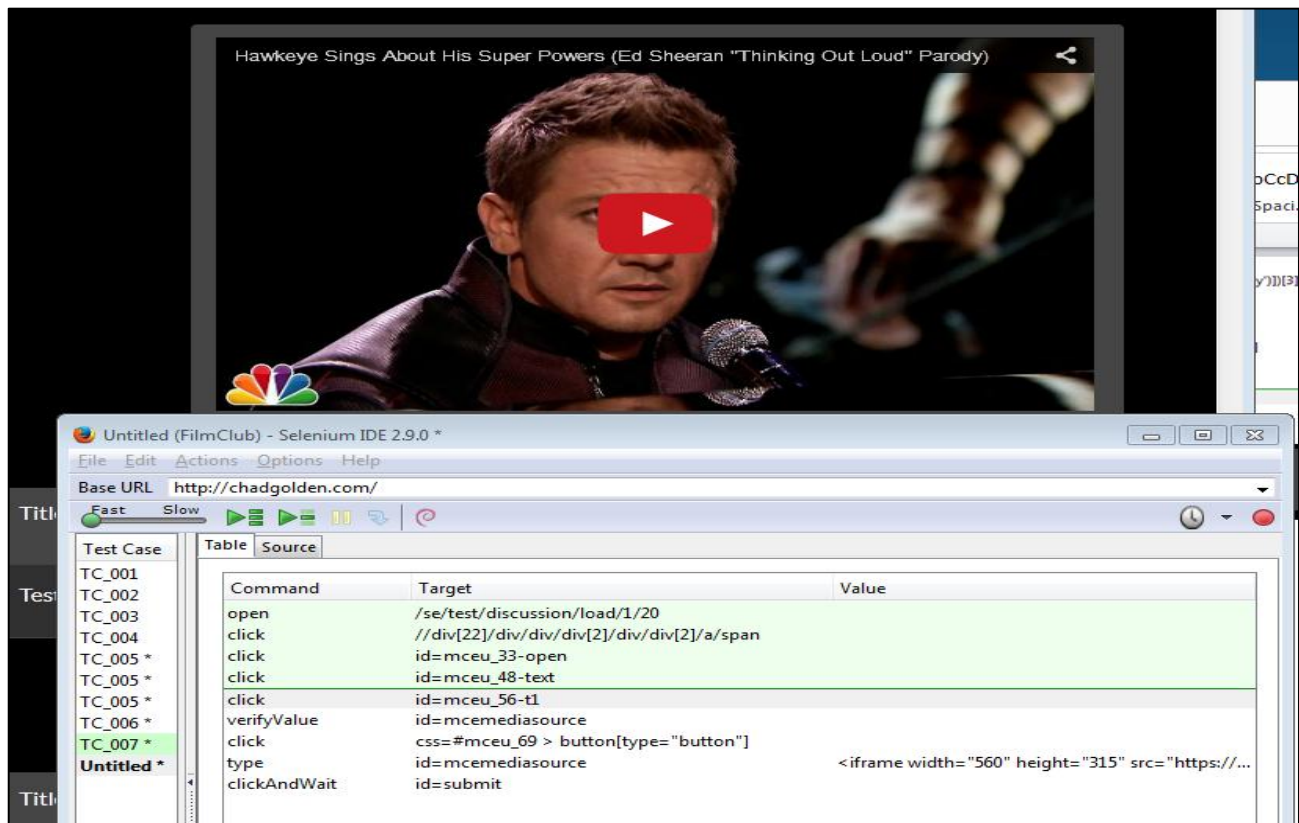
TC_007_Test2



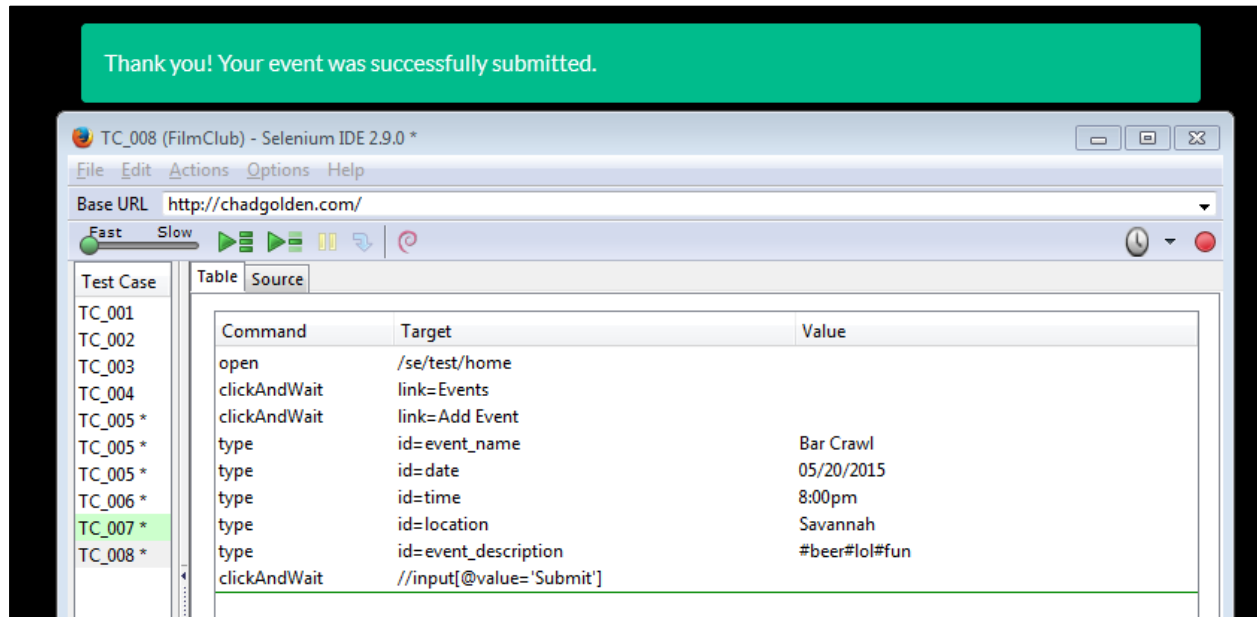
TC_007_Test3



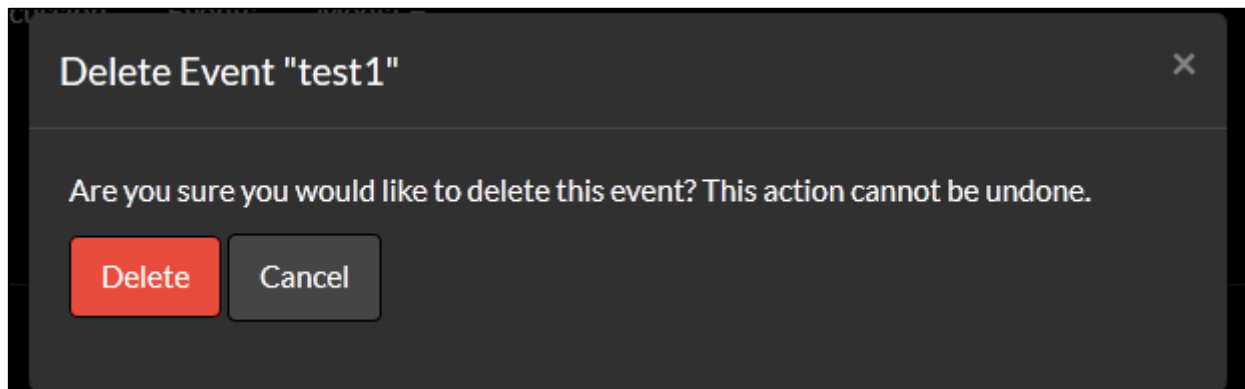
TC_007_Test4

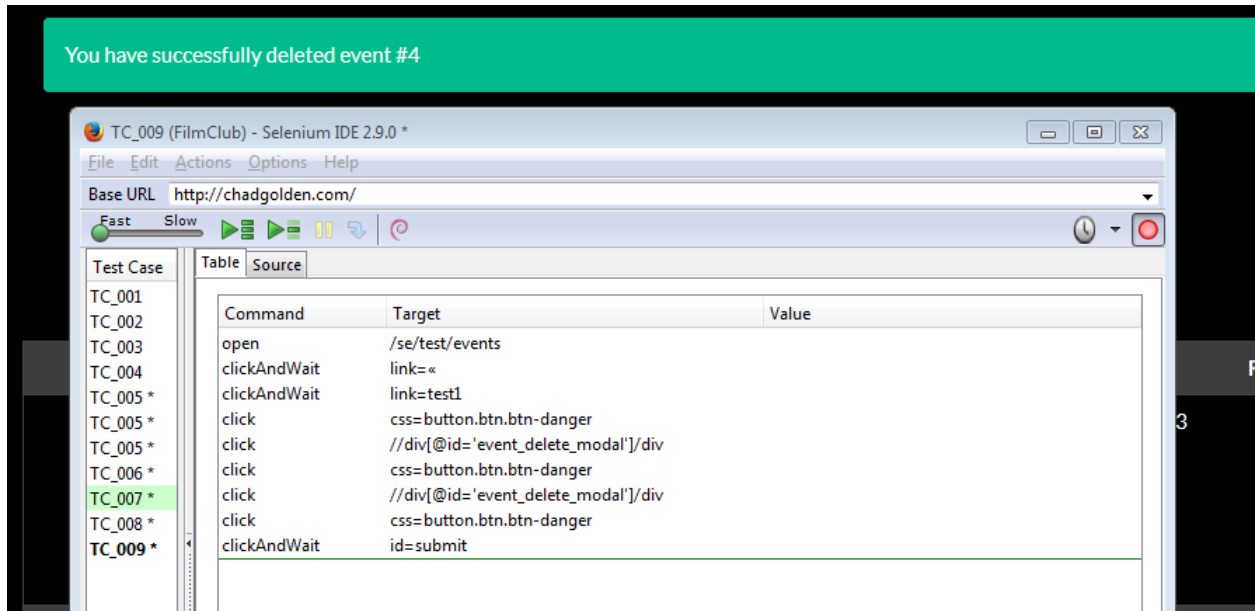


TC_008_Test1

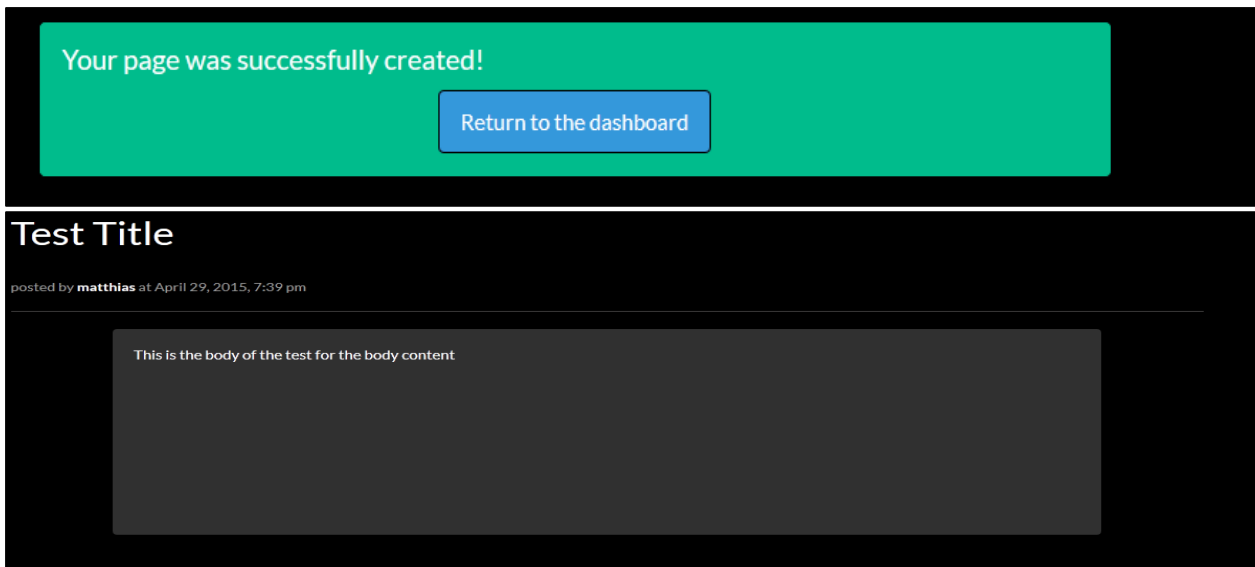


TC_010_Test1

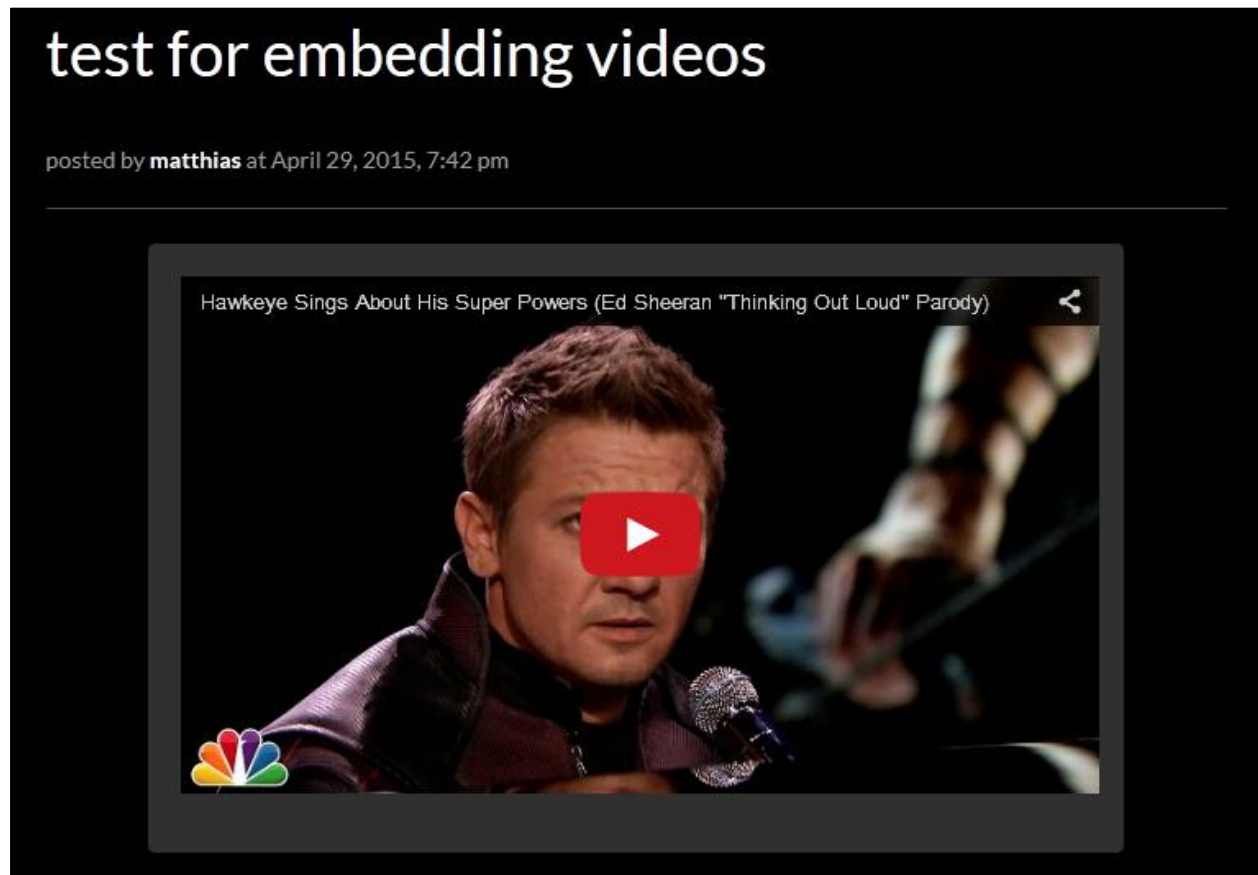




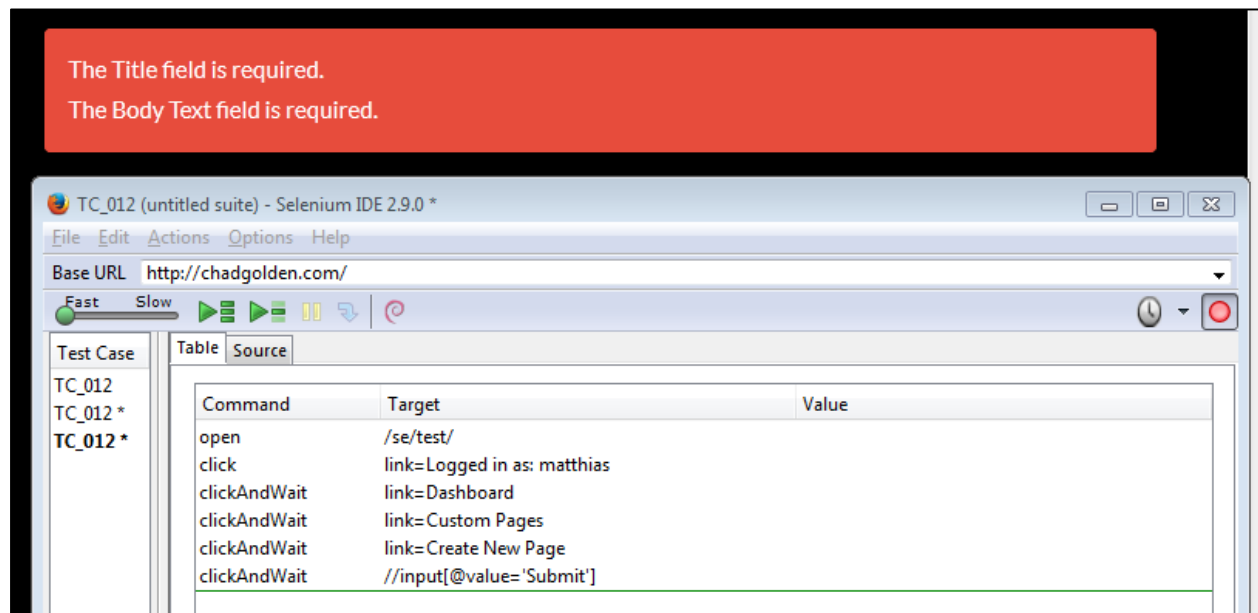
TC_012_Test1



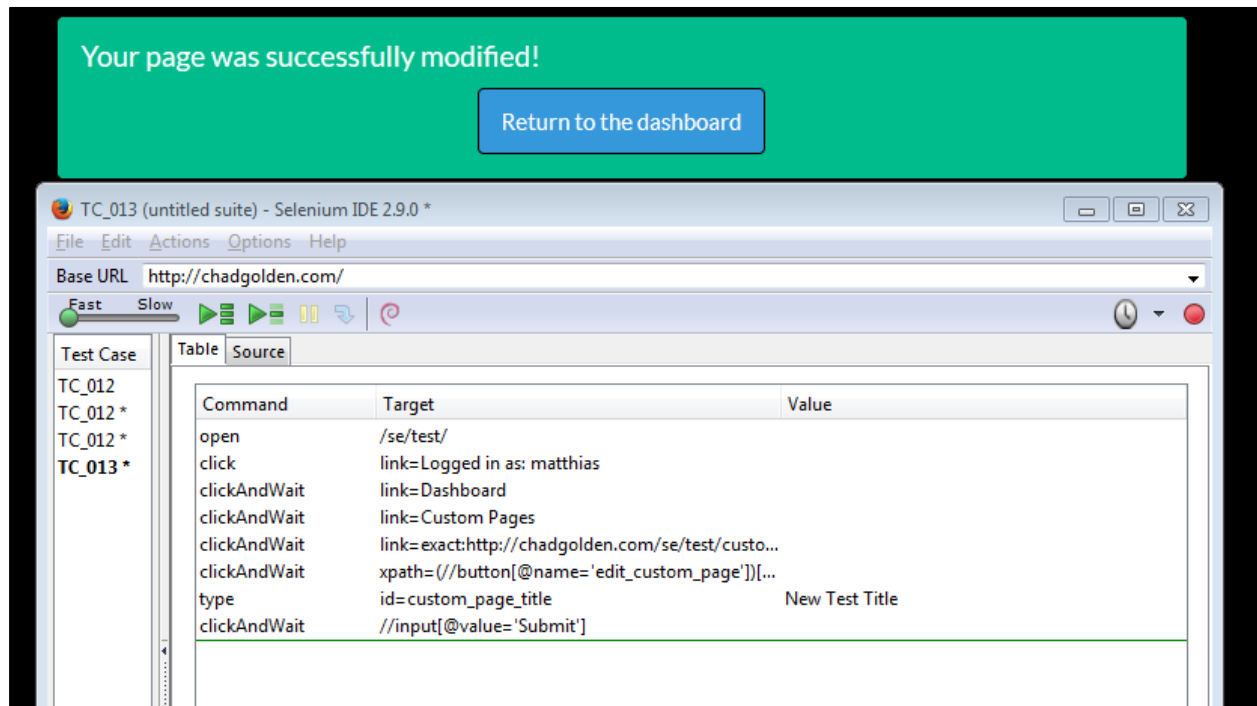
TC_012_Test2



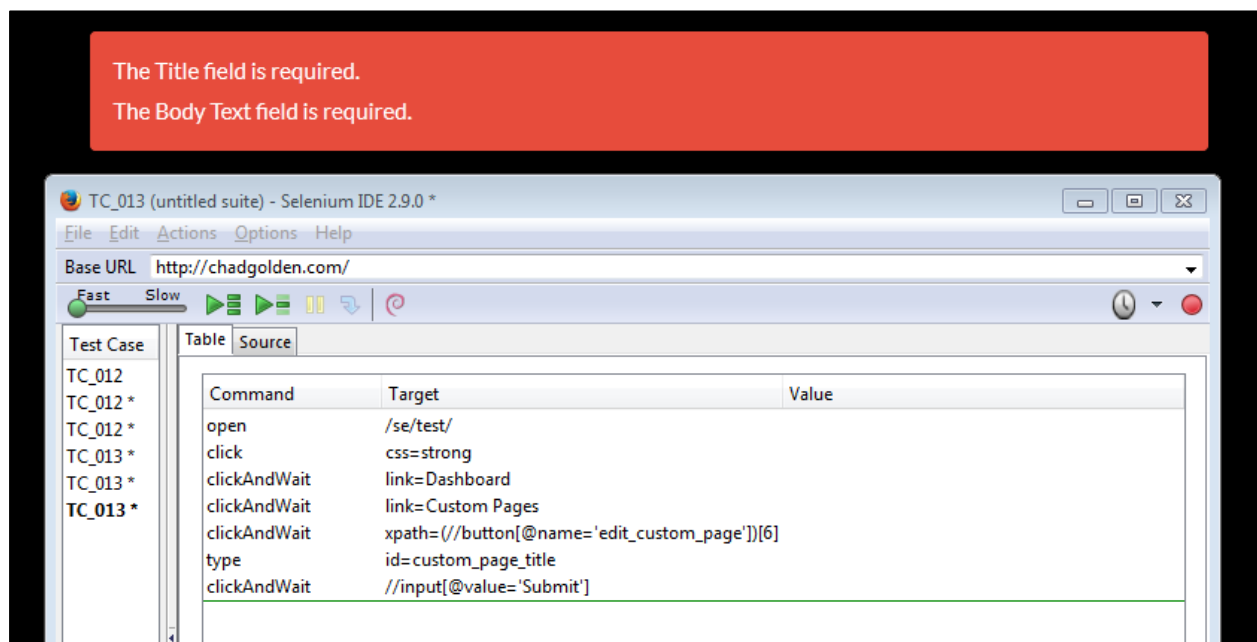
TC_012_Test3



TC_013_Test1



TC_013_Test3



TC_014_Test1

The image shows a 'Delete Page' dialog box with the text: 'Are you sure you would like to delete this? This action cannot be undone.' Below the text are two buttons: 'Yes, delete the page.' (red) and 'No, do not delete.' (blue). Below the dialog is a green notification bar that says: 'Custom page (PageID: 42) was successfully deleted.'

Below the notification bar is a screenshot of the Selenium IDE interface. The browser window shows 'http://chadgolden.com/'. The test suite 'TC_014' is selected, and the 'Table' tab is active. The table contains the following commands and targets:

Command	Target	Value
open	/se/test/	
click	link=Logged in as: matthias	
clickAndWait	link=Dashboard	
clickAndWait	link=Custom Pages	
click	xpath= (//button[@type='button'])[10]	
click	//div[@id='delete_modal']/div	
click	xpath= (//button[@type='button'])[10]	
clickAndWait	id=submit	

TC_015_Test1

The image shows the 'Messages' section of the CMS. The header says 'Messages View the latest suggestions and concerns.' Below the header is a table with the following columns: Subject, Message, First Name, Last Name, Email, Phone, Date, and Delete. The table contains four rows of messages:

Subject	Message	First Name	Last Name	Email	Phone	Date	Delete
Test Suggestion	We would like to test suggestion messages.	Bob	Bobby	Bob@email.com	1234567	2015-04-29 12:06:49	Delete
A Subject	A message.	Chad	Golden	chad@chadgolden.com	1111111111	2015-04-14 14:04:53	Delete
Test	Test	Chad	Golden	asdasf@chad.com	23423435656	2015-03-12 17:36:37	Delete
Test	I am testing the suggestion system.	Chad	Golden	chad@chadgolden.com	9999999999	2015-03-12 09:48:58	Delete

TC_016_Test1

Blah	Blah@Blah.com		Advisor			Suspend	Demote	Promote
------	---------------	--	---------	--	--	---------	--------	---------

Blah	Blah@Blah.com					Suspend	Demote	Promote
------	---------------	--	--	--	--	---------	--------	---------

Command	Target	Value
open	/se/test/	
click	css=strong	
clickAndWait	link=Dashboard	
clickAndWait	//a[contains(text(),'Manage Users')]	
clickAndWait	xpath= (//button[@name='suspend_user'])[3]	

TC_017_Test1

testuser392015	testuser392015@test.com		Regular	2015-03-09 11:10:06		Suspend	Demote	Promote
----------------	-------------------------	--	---------	---------------------	--	---------	--------	---------

testuser392015	testuser392015@test.com		Club Member	2015-03-09 11:10:06		Suspend	Demote	Promote
asdfasdf				1:24:01		Suspend	Demote	Promote
guseagle				5:25:38		Suspend	Demote	Promote
djff54				8:58:33		Suspend	Demote	Promote
pattycake				2:33:26		Suspend	Demote	Promote

TC_017_Test2

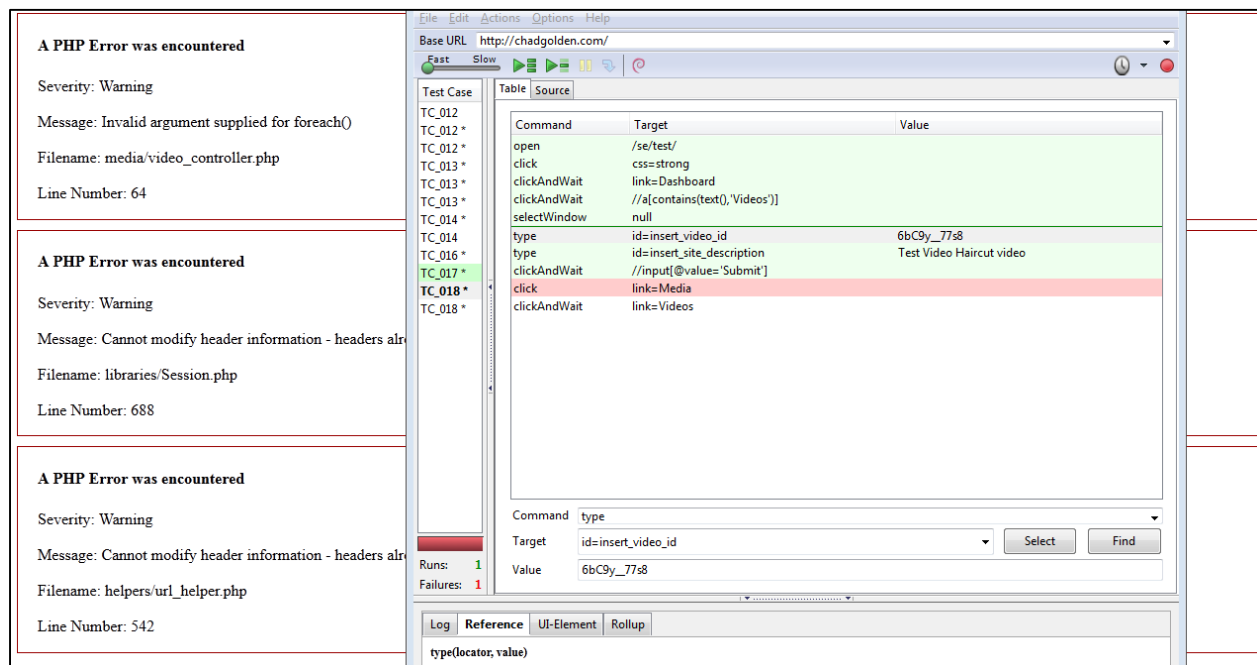
admin3	admin3@chadgolden.com					Suspend	Demote	Promote
--------	-----------------------	--	--	--	--	---------	--------	---------

admin3	admin3@chadgolden.com		Regular			Suspend	Demote	Promote
testuser						Suspend	Demote	Promote
chad				03-09 12:04:26				
chadgolde						Suspend	Demote	Promote
random								
bottom						Suspend	Demote	Promote

TC_018_Test1

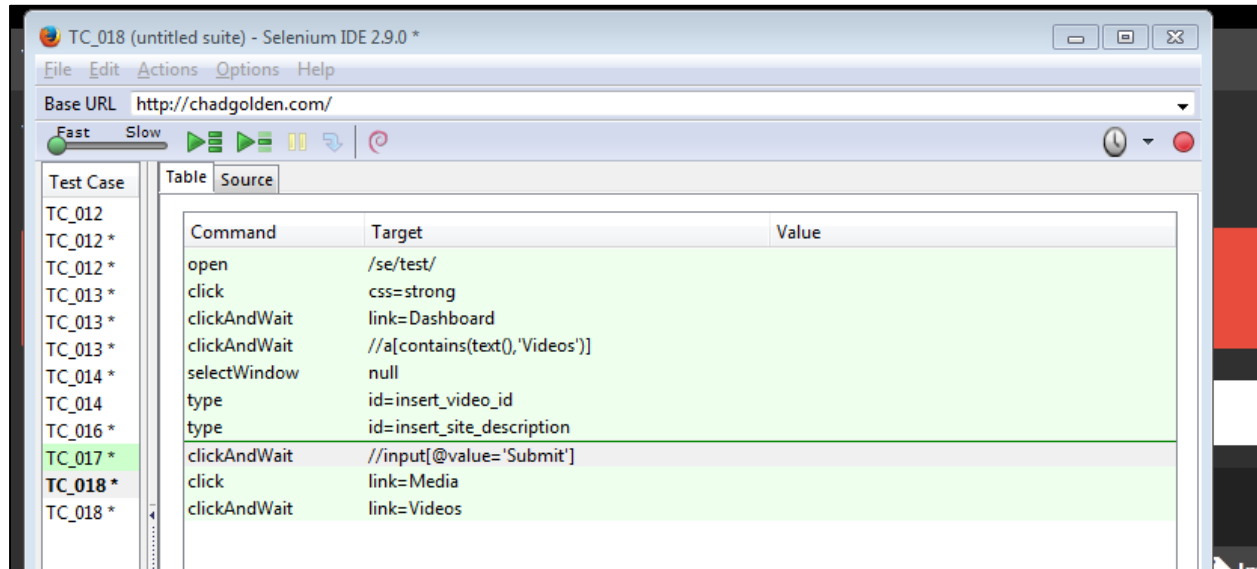


TC_018_Test2



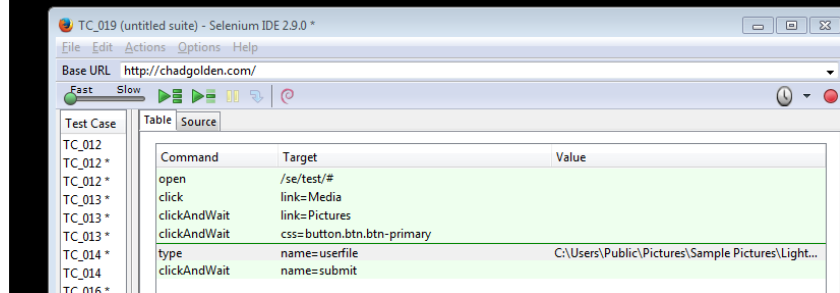
TC_018_Test3

The YouTube ID field is required.
The Site Description field is required.



TC_019_Test1

Your file was successfully uploaded!

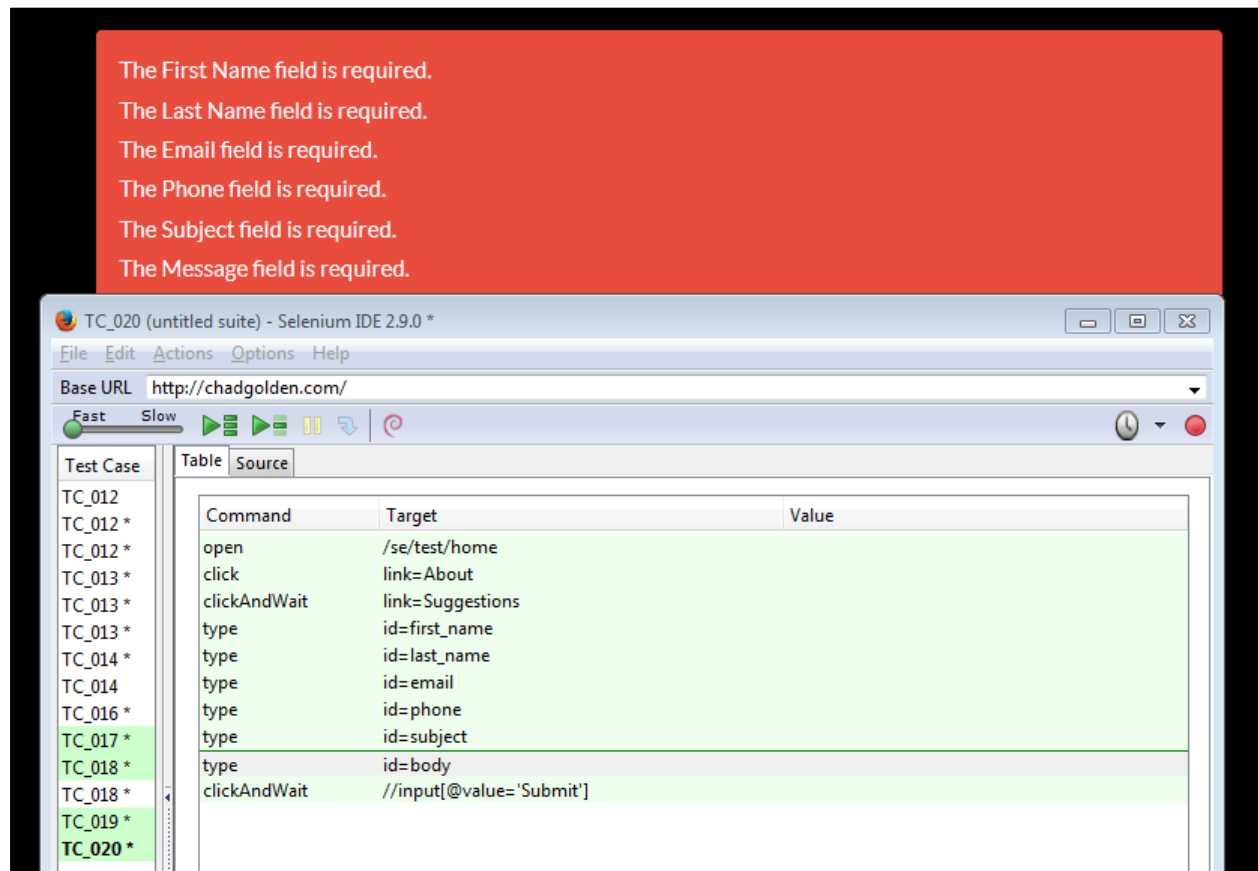


TC_020_Test1

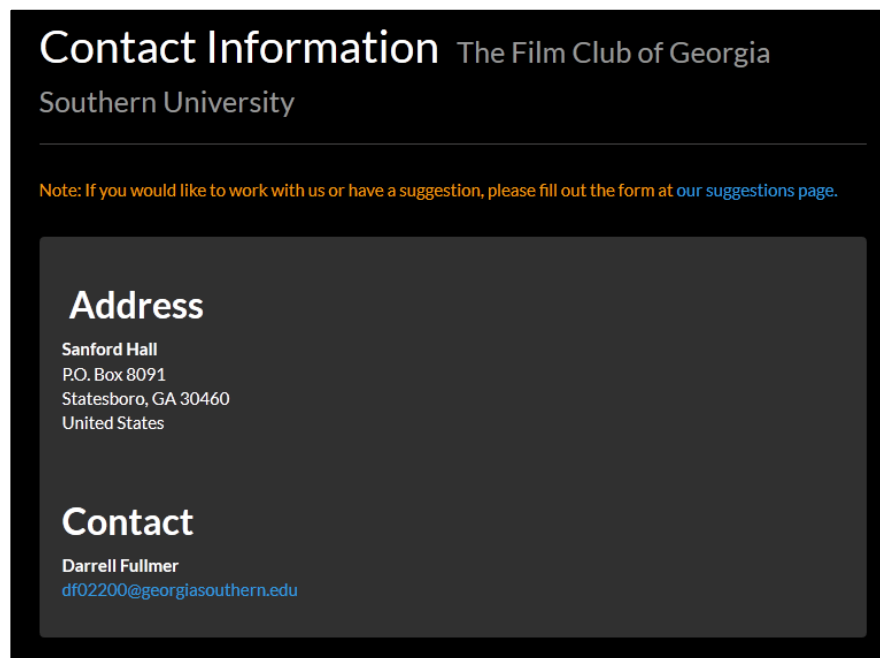
Thank you! Your suggestion was successfully submitted. We may contact you soon.

Subject	Message	Name	Name	Email	Phone	Date	Delete
Wilson!!!!	IM SORRY WILSON!!	Tom	Hanks	tomhanksgod@outlook.com	7707777777	2015-04-29 20:24:22	Delete

TC_020_Test2



TC_023_Test1



TC_025_Test1

You successfully added a role!

Current Roles

Here are the current roles of the club.

Role ID	Role Name	Description
1	Photographer	Responsible for tasks involving photography.
2	Timekeeper	Keeps track of time.
3	Producer	Produces films.
4	Scribe	Writes the script
5	Tester	Test for QA

Add Role

You may add roles here. After which you may assign these roles to current members.

Role Name

Enter the name of the role.

Role Description

Enter the description of the role.

TC_026_Test1

testuser

Tester

Set

Photographer

TC_026_Test2

You may view or modify current member's roles here.

Username	Current Role	New Role
qwer	Scribe	<div>Set</div> Photographer
admin		<div>Set</div> Photographer
testuser	Test 2	<div>Set</div> Photographer
chad	Producer	<div>Set</div> Photographer
random		<div>Set</div> Photographer
matthias	Photographer	<div>Set</div> Photographer
testuser392015		<div>Set</div> Photographer

TC_026 (FilmClub2) - Selenium IDE 2.9.0 *

Base URL http://chadgolden.com/

Test Case Table

Test Case	Command	Target	Value
TC_012	open	/se/test/dashboard	
TC_012	clickAndWait	link=Manage User Roles	
TC_013	select	xpath=//select[@name='role_select_list']	label=Test 2
TC_013	clickAndWait	xpath=//input[@value='Set']	
TC_014			
TC_016			
TC_017			
TC_018			
TC_019			
TC_020			
TC_021			
TC_023			
TC_024			

Runs: 1 Failures: 0

Command select

Target xpath=//select[@name='role_select_list']

Value label=Test 2

9.7 References

Meyer, Bertrand (1988). Object-Oriented Software Construction. Prentice Hall. ISBN 0-13-629049-3.

Agile Alliance. What is Agile Software Development? <http://www.agilealliance.org/the-alliance/what-is-agile/>.

9.8 Appendix D: Diary of Meeting and Tasks

Table 6.4-1

Date	01/14/2015
Location	IT Building
Start Time	6:00 PM
End Time	6:30 PM
In Attendance	Lamar Sims Matthew Stratton
Agenda	Initial Meeting with the Client Discussion of Plans and Client Needs
Summary of Discussion	Lamar and Matthew talked with Darrell about what would be beneficial to his club. He told us that a web system to manage events and engage members is...

Table 6.4-2

Date	02/3/2015
Location	IT Building Room 3214
Start Time	Noon
End Time	2:00 PM
In Attendance	Chad Golden O'Neal Jones
Agenda	Discussion of Languages and Concepts of Web Applications
Summary of Discussion	Discussed PHP concepts and how to handle sessions. Decided that sessions and other baseline data would best be handled by an abstract controller class as part of a Model-View-Controller (MVC) architecture.

Table 6.4-3

Date	2/4/2015
Location	IT Room 3214
Start Time	11:30 AM
End Time	12:30 PM
In Attendance	Chad Golden O'Neal Jones
Agenda	Web Page Design
Summary of Discussion	Designed a "Dashboard" page and explored how the page would interact with a database backend. Added a "privilege level" field within the PHP session so that the page may only be accessed by those users with high privilege levels.

Table 6.4-4

Date	02/5/2015
Location	IT Building
Start Time	6:00 PM
End Time	6:30 PM
In Attendance	Chad Golden Lamar Sims Matthew Stratton
Agenda	Meeting with the Client Discussion of Plans and Client Needs
Summary of Discussion	Lamar, Matthew and Chad talked with Darrell about what would be beneficial to his club.

Table 6.4-5

Date	2/10/2015
Location	IT Room 3214
Start Time	Noon
End Time	4PM
In Attendance	Chad Golden O'Neal Jones Patrick Meehan Matthew Stratton
Agenda	Exploration of Web Application Concepts
Summary of Discussion	Made sure everyone was up to speed on the basics of websites and web application design. Did a walkthrough of how to dynamically generate content on web pages using PHP and a database. In the process the group built a "Blog" system where a user types in a body and message in order to create a page for the blog and an entry on a page querying the database for all blogs.

Table 6.4-6

Date	2/11/2015
Location	IT Room 3214
Start Time	11:30 AM
End Time	12:45 PM
In Attendance	Chad Golden O'Neal Jones
Agenda	Discussed More Web Application Concepts
Summary of Discussion	Experimented with MVC architectures and how models, views, and controller classes interact with one another within a PHP application. Walked through HTML form validation techniques.

Table 6.4-7

Date	2/12/2015
Location	IT Room 3214
Start Time	1:00 PM
End Time	4:00 PM
In Attendance	Chad Golden O'Neal Jones
Agenda	Web Page Design
Summary of Discussion	Designed some prototypical pages and started formulation of some back-end logic for the pages.

Table 6.4-8

Date	02/12/2015
Location	IT Building
Start Time	6:00 PM
End Time	6:30 PM
In Attendance	Chad Golden Lamar Sims
Agenda	Meeting with the Client Discussion of Plans and Client Needs
Summary of Discussion	Lamar and Chad talked with Darrell about what would be beneficial to his club.

Table 6.4-9

Date	2/17/2015
Location	IT Room 3214
Start Time	Noon
End Time	4:00 PM
In Attendance	Chad Golden O'Neal Jones Lamar Sims
Agenda	Calendar Logic
Summary of Discussion	Darrell made it clear that he wanted a calendar of events. Chad, O'Neal, and Lamar began plugging in some back-end components to the calendar page created previously.

Table 6.4-10

Date	2/19/2015
Location	IT Room 3214
Start Time	Noon
End Time	4:00 PM
In Attendance	Chad Golden O'Neal Jones Patrick Meehan
Agenda	Web Page Design and Calendar Logic
Summary of Discussion	Added useful links to the footer. Did more implementation of calendar logic to the calendar of events page. The calendar can now be navigated correctly.

Table 6.4-11

Date	2/24/2015
Location	IT Room 3214
Start Time	Noon
End Time	4:00 PM
In Attendance	The Team
Agenda	SRS Tasks
Summary of Discussion	Made sure everyone was clear on their role for the completion of the SRS. Each team member worked on their particular role for the span of the meeting.

Table 6.4-12

Date	2/26/2015
Location	IT Room 3214
Start Time	Noon
End Time	4:00 PM
In Attendance	Chad Golden O'Neal Jones Lamar Sims Matthew Stratton
Agenda	SRS Tasks
Summary of Discussion	Each member in attendance further worked on completion of the SRS.

Table 6.4-13

Date	3/5/2015
Location	IT Room 3314
Start Time	11:00AM
End Time	1:30PM
In Attendance	Chad Golden Lamar Sims Matthew Stratton
Agenda	Web Page Design and Functionality
Summary of Discussion	Improved Photo Gallery layout and came up with new plan for the discussion page based on client reviews.

Table 6.4-14

Date	3/10/2015
Location	IT Room 3214
Start Time	10:30AM
End Time	1:30PM
In Attendance	The Team
Agenda	Improve website
Summary of Discussion	Focused on small details of the website and functionality.

Table 6.4-15

Date	3/17/2015
Location	IT Room 3214
Start Time	12:30AM
End Time	1:30PM
In Attendance	Chad Golden O'Neal Jones Lamar Sims
Agenda	Profile Photo Functionality
Summary of Discussion	The functionality to add a profile photo to an account was improved. Other details of the website were also worked on.

Table 6.4-16

Date	3/19/2015
Location	IT Room 3214
Start Time	11:30AM
End Time	1:30PM
In Attendance	Chad Golden O'Neal Jones Lamar Sims Patrick Meehan
Agenda	Website design and fixing bugs
Summary of Discussion	Improved layout and some small issues were worked on.

Table 6.4-17

Date	3/26/2015
Location	IT Room 3214
Start Time	12:30AM
End Time	3:30PM
In Attendance	Chad Golden O'Neal Jones Lamar Sims Matthew Stratton
Agenda	Discussion Page and Dashboard improvements
Summary of Discussion	Chad, Lamar, and O'Neal worked on the dashboard feature, Matthew worked on improving the Discussion page.

Table 6.4-18

Date	3/30/2015
Location	The Williams Center
Start Time	2:00PM
End Time	3:30PM
In Attendance	Matthew Stratton Darrell Fulmer
Agenda	Meet Film Club Officers and get feedback
Summary of Discussion	Matthew met with Darrell and the other officers of the Film Club. Got feedback from several officers which included <ul style="list-style-type: none"> • Twitter Feed Design • Discussion Page Design • Access to certain features depending on level Also got information regarding the faculty supervisor to begin planning web hosting from GSU.

Table 6.4-19

Date	4/2/2015
Location	IT Room 3214
Start Time	12:30PM
End Time	2:30PM
In Attendance	The Team
Agenda	Begin planning of Final Requirements Document
Summary of Discussion	The team decided what needed to be changed with the old document, Lamar and Matthew started to look into testing tools, and O'Neal began to plan out a User Guide.

Table 6.4-20

Date	4/7/2015
Location	IT Room 3214
Start Time	11:30AM
End Time	4:00PM
In Attendance	Chad Golden Lamar Sims
Agenda	Improve Dashboard Functionality
Summary of Discussion	Chad and Lamar improved the features of the dashboard and fixed bugs relating to functionality.

Table 6.4-21

Date	4/9/2015
Location	IT Room 3214
Start Time	12:30PM
End Time	4:00PM
In Attendance	Chad Golden Lamar Sims O'Neal Jones Matthew Stratton
Agenda	Improve Dashboard Functionality and discussion page.
Summary of Discussion	Improve the features of the dashboard and fix database bugs. Begin to reformat the discussion page from clients input.

Table 6.4-22

Date	4/16/2015
Location	IT Room 3214
Start Time	11:30AM
End Time	3:00PM
In Attendance	The Team
Agenda	Work on Final Requirements Document.
Summary of Discussion	The Final Requirements Document was worked on and improved.

Table 6.4-23

Date	4/21/2015
Location	IT Room 3214
Start Time	1:30PM
End Time	4:00PM
In Attendance	Chad Golden Lamar Sims
Agenda	Fix database bugs
Summary of Discussion	Chad and Lamar worked on some database errors regarding the photo gallery.

Table 6.4-24

Date	4/23/2015
Location	IT Room 3214
Start Time	11:30AM
End Time	3:00PM
In Attendance	Chad Golden O'Neal Jones
Agenda	Improve Website and work on Final Requirements Document
Summary of Discussion	Fix up some bugs with the website and work on Final Requirements Document.

Table 6.4-25

Date	4/28/2015
Location	IT Room 3214
Start Time	11:30AM
End Time	4:00PM
In Attendance	Chad Golden Lamar Sims O'Neal Jones Matthew Stratton
Agenda	Concluding on Final Requirements Document
Summary of Discussion	The team has collaborated since last week on the Final Requirements Document. The website is near completion and now individual roles regarding documentation are being supervised. Testing is near complete.

Table 6.4-26

Date	5/1/2015
Location	IT Room 3214
Start Time	10:30AM
End Time	4:00PM
In Attendance	Chad Golden Lamar Sims Matthew Stratton
Agenda	Final Touches
Summary of Discussion	The Final Requirements Document is complete. Tweaks and small details are added before finally being submitted to Dr. Allen.