



# GospelPreaching.com

## Analysis Report

## Table of Contents

<a href="#">System Requirements.....</a>	<a href="#">5</a>
<a href="#">Brief Overview.....</a>	<a href="#">5</a>
<a href="#">Event Table.....</a>	<a href="#">6</a>
<a href="#">Use Case Diagrams.....</a>	<a href="#">7</a>
<a href="#">Administrator Use Cases.....</a>	<a href="#">7</a>
<a href="#">Visitor Use Cases.....</a>	<a href="#">8</a>
<a href="#">Domain Model Class Diagram.....</a>	<a href="#">9</a>
<a href="#">Use Case Descriptions.....</a>	<a href="#">10</a>
<a href="#">Fully Described Use Cases.....</a>	<a href="#">10</a>
<a href="#">Add Article.....</a>	<a href="#">10</a>
<a href="#">Update Article.....</a>	<a href="#">11</a>
<a href="#">View Article.....</a>	<a href="#">12</a>
<a href="#">Add Author.....</a>	<a href="#">12</a>
<a href="#">Create Category.....</a>	<a href="#">13</a>
<a href="#">Register User.....</a>	<a href="#">13</a>
<a href="#">Approve New User.....</a>	<a href="#">14</a>
<a href="#">Update User Profile.....</a>	<a href="#">14</a>
<a href="#">Add Comment.....</a>	<a href="#">15</a>
<a href="#">Remove Comment.....</a>	<a href="#">15</a>
<a href="#">Like An Article.....</a>	<a href="#">16</a>
<a href="#">Unlike An Article.....</a>	<a href="#">16</a>
<a href="#">Ask Question.....</a>	<a href="#">17</a>
<a href="#">Update Question.....</a>	<a href="#">18</a>
<a href="#">Add Product.....</a>	<a href="#">19</a>
<a href="#">Create New Order.....</a>	<a href="#">20</a>
<a href="#">Record Order As Shipped.....</a>	<a href="#">21</a>
<a href="#">Adjust Order.....</a>	<a href="#">22</a>
<a href="#">Brief Use Case Descriptions.....</a>	<a href="#">23</a>
<a href="#">Delete Article.....</a>	<a href="#">23</a>
<a href="#">View Author.....</a>	<a href="#">23</a>
<a href="#">Update Author.....</a>	<a href="#">23</a>
<a href="#">Delete Author.....</a>	<a href="#">23</a>
<a href="#">View Category.....</a>	<a href="#">23</a>
<a href="#">Update Category.....</a>	<a href="#">23</a>
<a href="#">Delete Category.....</a>	<a href="#">23</a>
<a href="#">View Tagged Articles.....</a>	<a href="#">23</a>
<a href="#">View User Profile.....</a>	<a href="#">23</a>
<a href="#">Delete User.....</a>	<a href="#">23</a>
<a href="#">View Question.....</a>	<a href="#">24</a>
<a href="#">View Product.....</a>	<a href="#">24</a>
<a href="#">Update Product.....</a>	<a href="#">24</a>
<a href="#">Delete Product.....</a>	<a href="#">24</a>
<a href="#">View Recent Orders.....</a>	<a href="#">24</a>
<a href="#">System Sequence Diagrams.....</a>	<a href="#">25</a>

<a href="#">Add Article.....</a>	<a href="#">25</a>
<a href="#">Update Article.....</a>	<a href="#">26</a>
<a href="#">View Article.....</a>	<a href="#">27</a>
<a href="#">Add Author.....</a>	<a href="#">27</a>
<a href="#">Create Category.....</a>	<a href="#">27</a>
<a href="#">Register User.....</a>	<a href="#">28</a>
<a href="#">Approve New User.....</a>	<a href="#">28</a>
<a href="#">Update User Profile.....</a>	<a href="#">28</a>
<a href="#">Add Comment.....</a>	<a href="#">29</a>
<a href="#">Remove Comment.....</a>	<a href="#">29</a>
<a href="#">Like An Article.....</a>	<a href="#">29</a>
<a href="#">Unlike An Article.....</a>	<a href="#">30</a>
<a href="#">Ask Question.....</a>	<a href="#">30</a>
<a href="#">Update Question.....</a>	<a href="#">30</a>
<a href="#">Add Product.....</a>	<a href="#">31</a>
<a href="#">Create New Order.....</a>	<a href="#">32</a>
<a href="#">Record Order As Shipped.....</a>	<a href="#">33</a>
<a href="#">Adjust Order.....</a>	<a href="#">33</a>
<a href="#">Attribute Definitions.....</a>	<a href="#">34</a>
<a href="#">Article.....</a>	<a href="#">34</a>
<a href="#">Audio.....</a>	<a href="#">34</a>
<a href="#">Video.....</a>	<a href="#">34</a>
<a href="#">Document.....</a>	<a href="#">34</a>
<a href="#">OtherMedia.....</a>	<a href="#">34</a>
<a href="#">Category.....</a>	<a href="#">35</a>
<a href="#">Tag.....</a>	<a href="#">35</a>
<a href="#">Author.....</a>	<a href="#">35</a>
<a href="#">UserProfile.....</a>	<a href="#">35</a>
<a href="#">Comment.....</a>	<a href="#">35</a>
<a href="#">Like.....</a>	<a href="#">35</a>
<a href="#">Question.....</a>	<a href="#">36</a>
<a href="#">Product.....</a>	<a href="#">36</a>
<a href="#">OrderItem.....</a>	<a href="#">36</a>
<a href="#">Order.....</a>	<a href="#">37</a>
<a href="#">Alternate Design Concepts and Recommendation.....</a>	<a href="#">38</a>
<a href="#">Varying the scope of the system.....</a>	<a href="#">38</a>
<a href="#">Varying the degree of automation.....</a>	<a href="#">38</a>
<a href="#">Varying the type of technology used.....</a>	<a href="#">38</a>
<a href="#">Varying the approach to implementation.....</a>	<a href="#">38</a>
<a href="#">Project Alternatives.....</a>	<a href="#">38</a>
<a href="#">Recommended Alternative.....</a>	<a href="#">39</a>
<a href="#">Design and Implementation to Date.....</a>	<a href="#">39</a>
<a href="#">Technology and Working Environment.....</a>	<a href="#">39</a>
<a href="#">View Article Extended Sequence Diagram.....</a>	<a href="#">40</a>
<a href="#">Implementation to Date.....</a>	<a href="#">41</a>
<a href="#">Use Cases Implemented.....</a>	<a href="#">41</a>

<a href="#">Example Implementation.....</a>	<a href="#">41</a>
<a href="#">Initial Menu Hierarchy.....</a>	<a href="#">45</a>
<a href="#">Report Samples.....</a>	<a href="#">47</a>
<a href="#">Article Report by Category.....</a>	<a href="#">47</a>
<a href="#">Article Report by Author.....</a>	<a href="#">47</a>
<a href="#">Project Monitoring/Reporting To Date.....</a>	<a href="#">48</a>
<a href="#">Appendix A: Code Excerpts.....</a>	<a href="#">51</a>
<a href="#">models.py.....</a>	<a href="#">51</a>
<a href="#">views.py.....</a>	<a href="#">53</a>
<a href="#">urls.py.....</a>	<a href="#">53</a>
<a href="#">admin.py.....</a>	<a href="#">54</a>
<a href="#">index.html.....</a>	<a href="#">55</a>
<a href="#">detail.html.....</a>	<a href="#">56</a>
<a href="#">Legal Notes.....</a>	<a href="#">58</a>
<a href="#">Other Notes.....</a>	<a href="#">58</a>

## System Requirements

### ***Brief Overview***

CFTF Publications is a small publisher of religious books, tracts, and other media related to the Church of Christ. Allen Bailey is the owner/operator. Writers or extra editors are hired or volunteer on an as-needed basis. The company's main goal is to spread the Gospel and edify existing Christians through the publishing/availability of religious materials. CFTF doesn't have any physical publishing facilities.

The system is going to be an online portal for CFTF's electronic resources (documents, audio, video, etc.) and a storefront for the company's published goods. The system will simply be known as "GospelPreaching.com". Along with the system's basic functions mentioned above, it will also provide the ability to categorize and tag electronic materials, post and respond to questions (Q&A), allow users to connect their profiles with social networking sites (Facebook, Twitter, etc.), post comments, and "like" (favorite) articles.

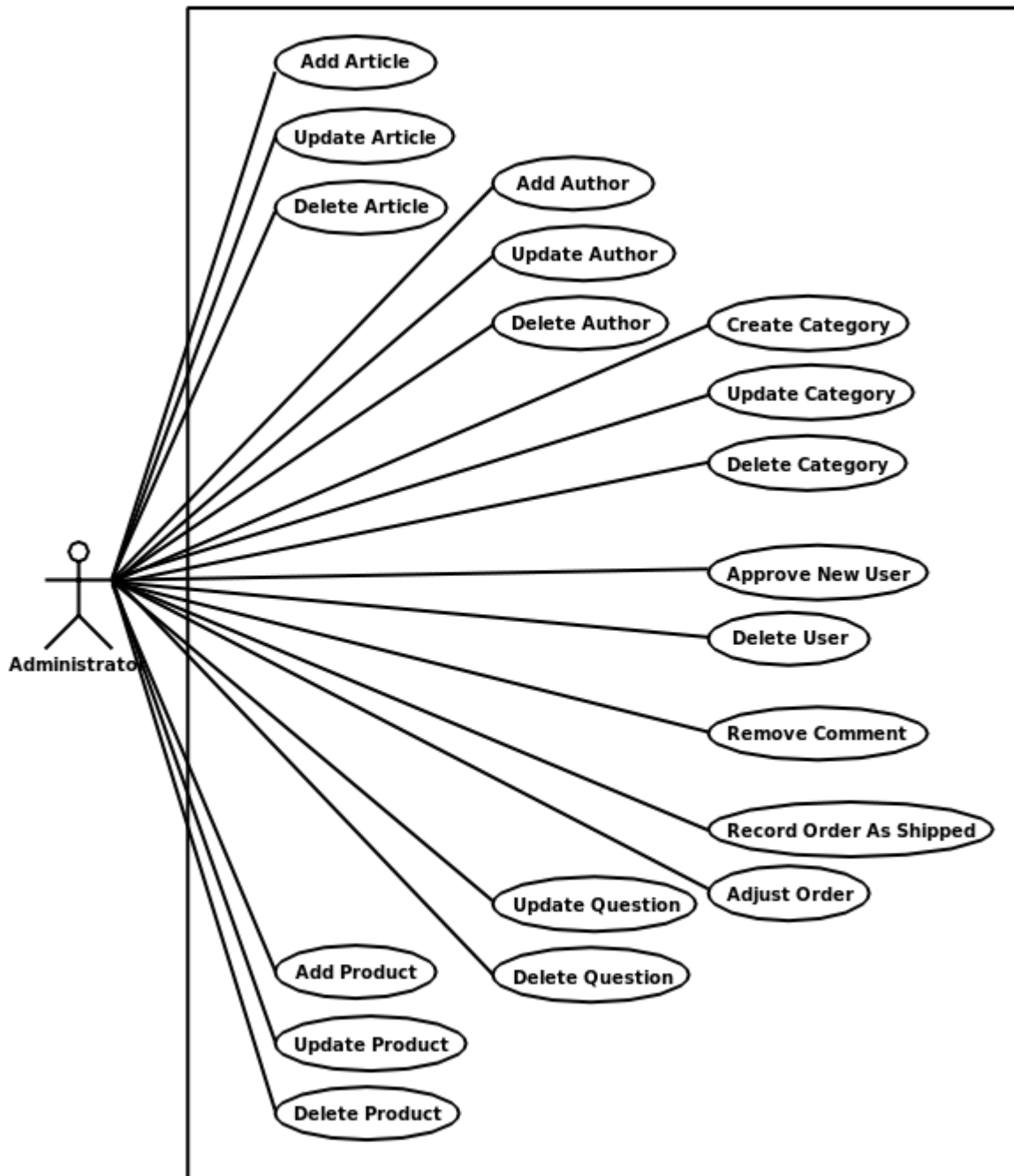
Allen Bailey would be the principle user of the site, performing administrative functions in the system. Other users would include the site visitors would be viewing the electronic resources, purchasing published goods, and utilizing the social aspects of the site. There is also the possibility of hiring or finding volunteers to perform the conversion of materials to electronic format. These people would need to have the ability to upload new materials to the site.

**Event Table**

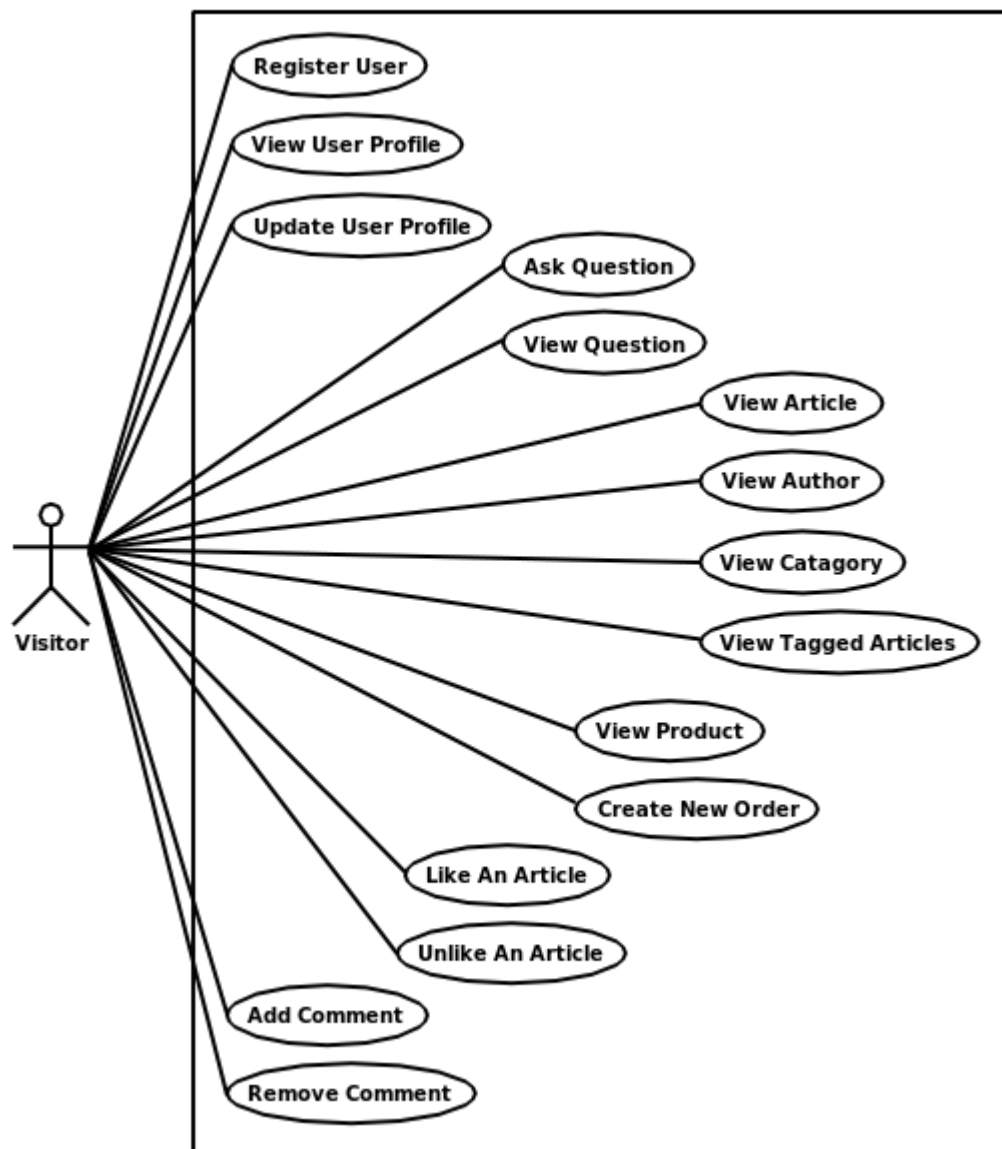
Event	Trigger	Source	Use Case	Response	Destination
Administrator adds an article	New article	Administrator	Add Article		
Visitor views an article	Article request	Visitor	View Article	Article details	Visitor
Administrator updates an article	Article update details	Administrator	Update Article		
Administrator deletes an article	Delete article request	Administrator	Delete Article		
Administrator adds an author	New author	Administrator	Add Author		
Visitor views an author	Author request	Visitor	View Author	Article report by Author	Visitor
Administrator updates an author	Author update details	Administrator	Update Author		
Administrator deletes an author	Delete author request	Administrator	Delete Author		
Administrator creates a category	New category	Administrator	Create Category		
Visitor views a category	Category request	Visitor	View Category	Article report by Category	Visitor
Administrator updates a category	Category update details	Administrator	Update Category		
Administrator deletes a category	Delete category request	Administrator	Delete Category		
Visitor views tagged articles	Article request by tag	Visitor	View Tagged Articles	Report of tagged articles	Visitor
Visitor wants to register a new user account	New user	Visitor	Register User	Registration confirmation New user notice	Visitor Administrator
Administrator approves a new user account	User account approval	Administrator	Approve New User	Account approval notice	Visitor
Visitor views a user profile	User profile request	Visitor	View User Profile	User profile details	Visitor
Visitor updates their profile	User profile update details	Visitor	Update User Profile		
Administrator or Visitor deletes a user account	Delete user request	Administrator or Visitor	Delete User		
Visitor adds a new comment	New comment	Visitor	Add Comment		
Administrator or Visitor removes a comment	Delete comment request	Administrator or Visitor	Remove Comment		
Visitor likes an article	Like article request	Visitor	Like An Article		
Visitor unlikes an article	Unlike article request	Visitor	Unlike An Article		
Visitor submits a question	New question	Visitor	Ask Question	New question notice	Administrator
Visitor views a question	Question request	Visitor	View Question	Question details	Visitor
Administrator updates a question	Question update details	Administrator	Update Question		
Administrator deletes a question	Delete question request	Administrator	Delete Question		
Administrator adds a new product	New product	Administrator	Add Product		
Visitor views product	Product request	Visitor	View Product	Product details	Visitor
Administrator updates a product	Product update details	Administrator	Update Product		
Administrator deletes a product	Delete product request	Administrator	Delete Product		
Visitor places an order	New order	Visitor	Create New Order	Order confirmation New order notice	Visitor Administrator
Administrator records an order as being shipped	Record order as shipped request	Administrator	Record Order As Shipped	Shipping confirmation	Visitor
Administrator adjusts an order	Order update details	Administrator	Adjust Order		
Administrator requests report of recent orders	Recent order report request	Administrator	View Recent Orders	Recent order report	Administrator

## Use Case Diagrams

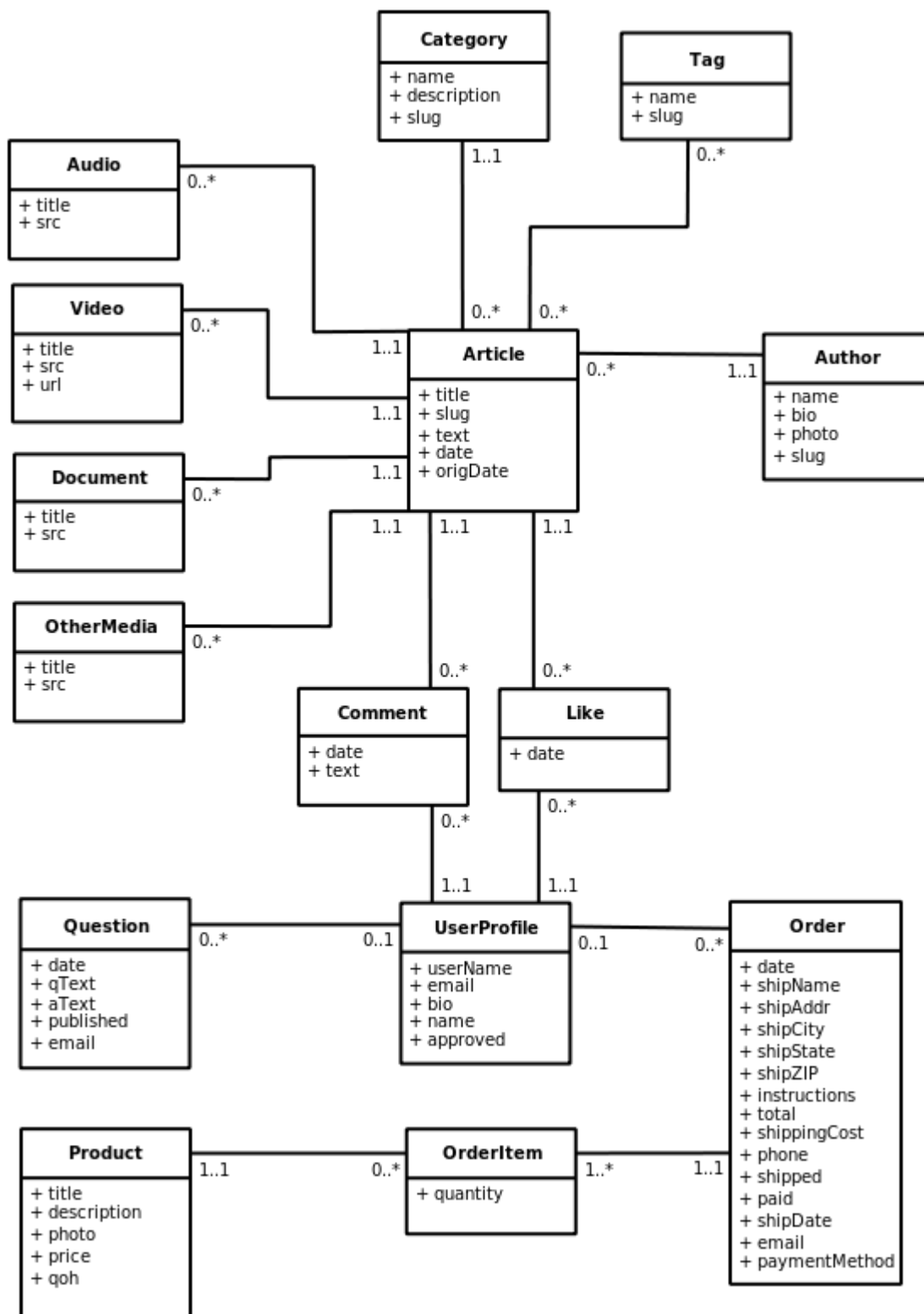
### Administrator Use Cases



### Visitor Use Cases





**Domain Model Class Diagram**

**Use Case Descriptions****Fully Described Use Cases**

Use Case Name:	<b>Add Article</b>	
Scenario:	Add article	
Triggering Event:	Administrator enters new Article details.	
Brief Description:	Administrator logs on and navigates to the Add Article page. Administrator fills in Article details; selects a Category, Author, and Tags; attaches new media items; and submits finished Article.	
Actors:	Administrator	
Stakeholders:		
Preconditions:	Category and Author must exist.	
Postconditions:	Article and media items (Audio, Video, Document, OtherMedia) must be created. Media items must be associated with Article. Article must be associated with Category and Author.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>Administrator connects to the site and navigates to the administration page.</li> <li>Administrator logs in.</li> <li>Administrator navigates to the Add Article page.</li> <li>Administrator fills in Article details.</li> <li>Administrator selects Category.</li> <li>Administrator selects Author.</li> <li>Administrator selects Tags.</li> <li>Administrator requests to add a new media item (Audio, Video, Document, or OtherMedia).</li> <li>Administrator fills in new media item details.</li> <li>Repeat steps 8 and 9.</li> <li>Administrator submits completed Article.</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>3.1 Display new Article form.</li> <li>8.1 Add a new form to the page for the new media item.</li> <li>11.1 Add Article confirmation and return to administration page.</li> </ol>
Exception Conditions:	<ol style="list-style-type: none"> <li>5.1 If Category does not exist yet, user can click link to open new window and add a new Category.</li> <li>5.1 If Author does not exist yet, user can click link to open new window and add a new Author.</li> <li>5.1 If a Tag does not exist yet, user can click link to open new window and add a new Tag.</li> </ol>	

Use Case Name:	<b>Update Article</b>	
Scenario:	Update article	
Triggering Event:	Administrator updates Article details.	
Brief Description:	Administrator logs on and navigates to the Article update page. Administrator updates Article details and submits finished Article.	
Actors:	Administrator	
Stakeholders:		
Preconditions:	Category and Author must exist.	
Postconditions:	Article and media items (Audio, Video, Document, OtherMedia) must be updated. Media items must be associated with Article. Article associations with Category and Author must be updated.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>Administrator connects to the site and navigates to the administration page.</li> <li>Administrator logs in.</li> <li>Administrator finds the Article to update and selects the Update option.</li> <li>Administrator updates Article details.</li> <li>Administrator updates Category.</li> <li>Administrator updates Author.</li> <li>Administrator updates Tags.</li> <li>Administrator requests to add a new media item (Audio, Video, Document, or OtherMedia).</li> <li>Administrator fills in new media item details.</li> <li>Repeat steps 8 and 9.</li> <li>Administrator marks media item for deletion.</li> <li>Administrator updates existing media item details.</li> <li>Administrator submits updated Article.</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>3.1 Display Article update form.</li> <li>8.1 Add a new form to the page for the new media item.</li> <li>13.1 Update Article confirmation and return to administration page.</li> </ol>
Exception Conditions:	<ol style="list-style-type: none"> <li>5.1 If Category does not exist yet, user can click link to open new window and add a new Category.</li> <li>5.1 If Author does not exist yet, user can click link to open new window and add a new Author.</li> <li>5.1 If a Tag does not exist yet, user can click link to open new window and add a new Tag.</li> </ol>	

Use Case Name:	<b>View Article</b>	
Scenario:	View Article	
Triggering Event:	Visitor requests to view an Article	
Brief Description:	Visitor locates an Article and requests to view the Article details.	
Actors:	Visitor	
Stakeholders:		
Preconditions:	Article must exist.	
Postconditions:		
Flow of Activities:	<b>Actor</b> 1. Visitor connects to site and navigates to the main Article list. 1a. Visitor navigates to Category, Author, or Tag view. 2. Visitor selects Article from list to view details. 2a. Visitor enters Article URL directly.	<b>System</b> 1.1 Display list of all Articles. 1a.1 Display list of Articles by Category, Author, or Tag. 2.1 Display Article details and media item (Audio, Video, Document, OtherMedia) details/players.
Exception Conditions:	2.1 If Article does not exist, redirect Visitor to error page.	

Use Case Name:	<b>Add Author</b>	
Scenario:	Add Author	
Triggering Event:	Administrator enters new Author details.	
Brief Description:	Administrator logs on and navigates to the Add Author page. Administrator fills in Author details and submits finished Author details.	
Actors:	Administrator	
Stakeholders:		
Preconditions:		
Postconditions:	Author must created.	
Flow of Activities:	<b>Actor</b> 1. Administrator connects to the site and navigates to the administration page. 2. Administrator logs in. 3. Administrator navigates to the Add Author page. 4. Administrator fills in Author details. 5. Administrator submits completed Author details.	<b>System</b> 2.1 Validate administrator account. 3.1 Display new Author form. 5.1 Add Author confirmation and return to administration page.
Exception Conditions:		

Use Case Name:	<b>Create Category</b>	
Scenario:	Create Category	
Triggering Event:	Administrator enters new Category details.	
Brief Description:	Administrator logs on and navigates to the Add Category page. Administrator fills in Author details and submits finished Category details.	
Actors:	Administrator	
Stakeholders:		
Preconditions:		
Postconditions:	Category must created.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Administrator connects to the site and navigates to the administration page.</li> <li>2. Administrator logs in.</li> <li>3. Administrator navigates to the Add Category page.</li> <li>4. Administrator fills in Category details.</li> <li>5. Administrator submits completed Category details.</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>3.1 Display new Category form.</li> <li>5.1 Add Category confirmation and return to administration page.</li> </ol>
Exception Conditions:		

Use Case Name:	<b>Register User</b>	
Scenario:	Register User	
Triggering Event:	Visitor requests to register a new account.	
Brief Description:	Visitor requests to register a new account and provides profile information.	
Actors:	Visitor	
Stakeholders:	Administrator: to respond to registration request.	
Preconditions:		
Postconditions:	Unapproved User Profile created.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Visitor connects to site and clicks the register link.</li> <li>2. Visitor submits registration form with profile information.</li> </ol>	<ol style="list-style-type: none"> <li>1.1 Display User registration form.</li> <li>2.1 Confirm successful registration.</li> <li>2.2 Notify Administrator of new user account.</li> </ol>
Exception Conditions:		

Use Case Name:	<b>Approve New User</b>	
Scenario:	Approve New User	
Triggering Event:	Administrator requests user approval.	
Brief Description:	Administrator logs onto site and approves user.	
Actors:	Administrator	
Stakeholders:	Visitor: account approval notice.	
Preconditions:	Unapproved User Profile must exist.	
Postconditions:	User Profile updated with approved status.	
Flow of Activities:	<b>Actor</b> 1. Administrator connects to site and navigates to the administration page. 2. Administrator logs in. 3. Administrator navigates to the user approval page. 4. Administrator requests approval of User Profile.	<b>System</b> 2.1 Validate administrator account. 4.1 User approval confirmation and return to administration page.
Exception Conditions:	4.1 If Administrator decides to not approve user profile, it can be deleted instead.	

Use Case Name:	<b>Update User Profile</b>	
Scenario:	Update User Profile	
Triggering Event:	Visitor updates User Profile details.	
Brief Description:	Visitor logs on and navigates to the User Profile update page. Visitor updates User Profile details and submits finished details.	
Actors:	Visitor	
Stakeholders:		
Preconditions:	User Profile must exist.	
Postconditions:	User Profile must be updated.	
Flow of Activities:	<b>Actor</b> 1. Visitor connects to site and logs in. 2. Visitor navigates to the profile update page. 3. Visitor updates User Profile details. 4. Visitor submits updated User Profile.	<b>System</b> 1.1 Validate user account. 2.1 Display User Profile update form. 4.1 User Profile update confirmation and return to User Profile page.
Exception Conditions:		

Use Case Name:	<b>Add Comment</b>	
Scenario:	Add Comment	
Triggering Event:	Visitor creates new Comment.	
Brief Description:	Visitor logs onto site and navigates to an Article. Visitor submits Comment text using new Comment form.	
Actors:	Visitor	
Stakeholders:		
Preconditions:	Article must exist.	
Postconditions:	Comment must be created and associated to Article.	
Flow of Activities:	Actor	System
	1. Visitor connects to site and logs in. 2. Visitor navigates to an Article. 3. Visitor submits new Comment details.	1.1 Validate user account.  2.1 Display Article details and new Comment form. 3.1 Add Comment confirmation.
Exception Conditions:		

Use Case Name:	<b>Remove Comment</b>	
Scenario:	Remove Comment	
Triggering Event:	Visitor or Administrator requests removal of Comment.	
Brief Description:	Visitor or Administrator logs onto site and navigates to an Article. Visitor or Administrator removes a Comment from the Article.	
Actors:	Visitor, Administrator	
Stakeholders:		
Preconditions:	Comment must exist.	
Postconditions:	Comment must be removed.	
Flow of Activities:	Actor	System
	1. Visitor or Administrator connects to site and logs in. 2. Visitor or Administrator navigates to an Article. 3. Visitor or Administrator requests removal of Comment.	1.1 Validate account.  2.1 Display Article details including existing comments. 3.1a If Visitor: validate ownership of comment and confirm removal. 3.1b If Administrator: confirm removal.
Exception Conditions:		

Use Case Name:	<b><i>Like An Article</i></b>	
Scenario:	Like An Article	
Triggering Event:	Visitor submits Like request.	
Brief Description:	Visitor logs onto site and navigates to an Article. Visitor requests a Like for the Article.	
Actors:	Visitor	
Stakeholders:		
Preconditions:	Article and User Profile must exist.	
Postconditions:	Like must be created and associated with Article and User Profile.	
Flow of Activities:	Actor	System
	1. Visitor connects to site and logs in. 2. Visitor navigates to an Article. 3. Visitor requests a new Like	1.1 Validate user account. 2.1 Display Article details 3.1 New Like confirmation.
Exception Conditions:		

Use Case Name:	<b><i>Unlike An Article</i></b>	
Scenario:	Unlike An Article	
Triggering Event:	Visitor submits Unlike request.	
Brief Description:	Visitor logs onto site and navigates to an Article. Visitor requests Like for the Article be removed.	
Actors:	Visitor	
Stakeholders:		
Preconditions:	Article, User Profile, and Like must exist.	
Postconditions:	Like must be removed.	
Flow of Activities:	Actor	System
	1. Visitor connects to site and logs in. 2. Visitor navigates to an Article. 3. Visitor requests removal of Like for this Article.	1.1 Validate user account. 2.1 Display Article details 3.1 Like removal confirmation.
Exception Conditions:		



Use Case Name:	<b><i>Ask Question</i></b>	
Scenario:	Ask Question	
Triggering Event:	Visitor submits new Question details.	
Brief Description:	Visitor submits a new Question and Administrator is alerted.	
Actors:	Visitor	
Stakeholders:	Administrator: alert to respond to question.	
Preconditions:		
Postconditions:	Question is created and either has an email address or is associated with a User Profile	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Visitor connects to site and logs in.</li> <li>2. Visitor navigates to the Ask Question page.</li> <li>3. Visitor submits complete new Question details.</li> </ol>	<ol style="list-style-type: none"> <li>1.1 Validate user account.</li> <li>2.1 Display Ask Question form.</li> <li>3.1 Question submission confirmation and return to main question page.</li> </ol>
Exception Conditions:	3.1 If Visitor did not log in under step 1, require email address field.	

Use Case Name:	<b><i>Update Question</i></b>	
Scenario:	Update Question	
Triggering Event:	Administrator updates Question details.	
Brief Description:	Administrator logs on and navigates to the Question update page. Administrator updates Question details and submits finished Question.	
Actors:	Administrator	
Stakeholders:	Visitor: alert that question has been answered.	
Preconditions:	Question must exist.	
Postconditions:	Question must be updated.	
Flow of Activities:	<p style="text-align: center;">Actor</p> <ol style="list-style-type: none"> <li>1. Administrator connects to the site and navigates to the administration page.</li> <li>2. Administrator logs in.</li> <li>3. Administrator finds the Question to update and selects the Update option.</li> <li>4. Administrator updates Question details.</li> <li>5. Administrator submits updated Question.</li> </ol>	<p style="text-align: center;">System</p> <ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>3.1 Display Question update form.</li> <li>5.1 Update Question confirmation and return to administration page.</li> <li>5.2 Send notice of answered/updated question to Visitor if published.</li> </ol>
Exception Conditions:		

Use Case Name:	<b>Add Product</b>	
Scenario:	Add Product	
Triggering Event:	Administrator enters new Product details.	
Brief Description:	Administrator logs on and navigates to the Add Product page. Administrator fills in Product details and submits finished Product.	
Actors:	Administrator	
Stakeholders:		
Preconditions:		
Postconditions:	Product must be created.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Administrator connects to the site and navigates to the administration page.</li> <li>2. Administrator logs in.</li> <li>3. Administrator navigates to the Add Product page.</li> <li>4. Administrator fills in Product details.</li> <li>5. Administrator submits completed Product.</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>3.1 Display new Product form.</li> <li>5.1 Add Product confirmation and return to administration page.</li> </ol>
Exception Conditions:		

Use Case Name:	<b>Create New Order</b>	
Scenario:	Create New Order	
Triggering Event:	Visitor enters new Order details.	
Brief Description:	Visitor connects to site. Visitor selects desired products and fills in Order details. Visitor is taken to payment screen if needed and fills it out. Visitor submits finished Order.	
Actors:	Visitor	
Stakeholders:	Administrator: process new order. Paypal: payment details.	
Preconditions:	Products must exist.	
Postconditions:	Order and Order Items must exist. Order Items must be associated with Order. User Profile, if logged in, must be associated with Order. Product QOH must be updated.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Visitor navigates to the Product page.</li> <li>2. Visitor requests Product details.</li> <li>3. Visitor requests Product be added to Order.</li> <li>4. Repeat steps 1-3.</li> <li>5. Visitor clicks the checkout link to finish placing Order.</li> <li>6. Visitor submits Order details.</li> <li>7. Visitor completes payment information on Paypal site.</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Display product details.</li> <li>3.1 Product added to cart.</li> <li>5.1 Display checkout form.</li> <li>6.1a If Visitor indicates payment by check: inform them about check policy and submit for manual processing by Administrator.</li> <li>6.1b If Visitor indicates Paypal payment: redirect to Paypal payment screen/site.</li> <li>7.1 Paypal confirms payment receipt.</li> <li>7.2 Paypal sends IPN message to listener and payment details are processed and saved.</li> <li>7.3 If Paypal payment is successful: Order is marked paid and Administrator is notified.</li> </ol>
Exception Conditions:	7.1 If Paypal payment is not successful: Order is left marked unpaid and Administrator is notified for manual processing.	

Use Case Name:	<b><i>Record Order As Shipped</i></b>	
Scenario:	Record Order As Shipped	
Triggering Event:	Administrator requests to mark Order as shipped.	
Brief Description:	Administrator logs onto site and marks Order as shipped.	
Actors:	Administrator	
Stakeholders:	Visitor: alert that order has shipped.	
Preconditions:	Order must exist.	
Postconditions:	Order must be updated with shipped status.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Administrator connects to site and navigates to the administration page.</li> <li>2. Administrator logs in.</li> <li>3. Administrator navigates to the unshipped Order page.</li> <li>4. Administrator requests to mark Order as shipped</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>4.1 Order shipped confirmation and return to administration page.</li> <li>4.2 Send notification of Order shipment to Visitor.</li> </ol>
Exception Conditions:		

Use Case Name:	<b><i>Adjust Order</i></b>	
Scenario:	Adjust Order	
Triggering Event:	Administrator updates Order details.	
Brief Description:	Administrator logs on and navigates to the Order adjustment page. Administrator adjusts Order details and submits adjusted Order.	
Actors:	Administrator	
Stakeholders:		
Preconditions:	Order must exist.	
Postconditions:	Order must be updated.	
Flow of Activities:	Actor	System
	<ol style="list-style-type: none"> <li>1. Administrator connects to the site and navigates to the administration page.</li> <li>2. Administrator logs in.</li> <li>3. Administrator finds the Order to update and selects the Update option.</li> <li>4. Administrator updates Order details.</li> <li>5. Administrator submits adjusted Order.</li> <li>6. Administrator manually adjusts payment and notifies visitor of any changes.</li> </ol>	<ol style="list-style-type: none"> <li>2.1 Validate administrator account.</li> <li>3.1 Display Order update form.</li> <li>5.1 Order adjustment confirmation and return to administration page.</li> </ol>
Exception Conditions:		

## Brief Use Case Descriptions

### **Delete Article**

Administrator logs onto site and navigates to the Article page. Administrator requests to delete Article.

### **View Author**

Visitor navigates to the Author page and requests to view Author details. System displays report of associated Articles and Author details.

### **Update Author**

Administrator logs onto site and navigates to the Author update page. Administrator submits updated Author details.

### **Delete Author**

Administrator logs onto site and navigates to the Author page. Administrator requests to delete Author.

### **View Category**

Visitor navigates to the Category page and requests to view Category details. System displays report of associated Articles and Category details

### **Update Category**

Administrator logs onto site and navigates to the Category update page. Administrator submits updated Category details.

### **Delete Category**

Administrator logs onto site and navigates to the Category page. Administrator requests to delete Category.

### **View Tagged Articles**

Visitor navigates to the Tag page and requests to view tagged Articles. System displays Articles associated with that Tag.

### **View User Profile**

Visitor navigates to User page and requests to view User Profile details. System displays User Profile details.

### **Delete User**

Administrator or Visitor logs onto site and navigates to the User Profile detail page. Administrator or Visitor requests to delete User.

***View Question***

Visitor navigates to Question page and requests to view Question details. System displays Question details.

***View Product***

Visitor navigates to Product page and requests to view Product details. System displays Product details.

***Update Product***

Administrator logs onto site and navigates to the Product update page. Administrator submits updated Product details.

***Delete Product***

Administrator logs onto site and navigates to the Product page. Administrator requests to delete Product.

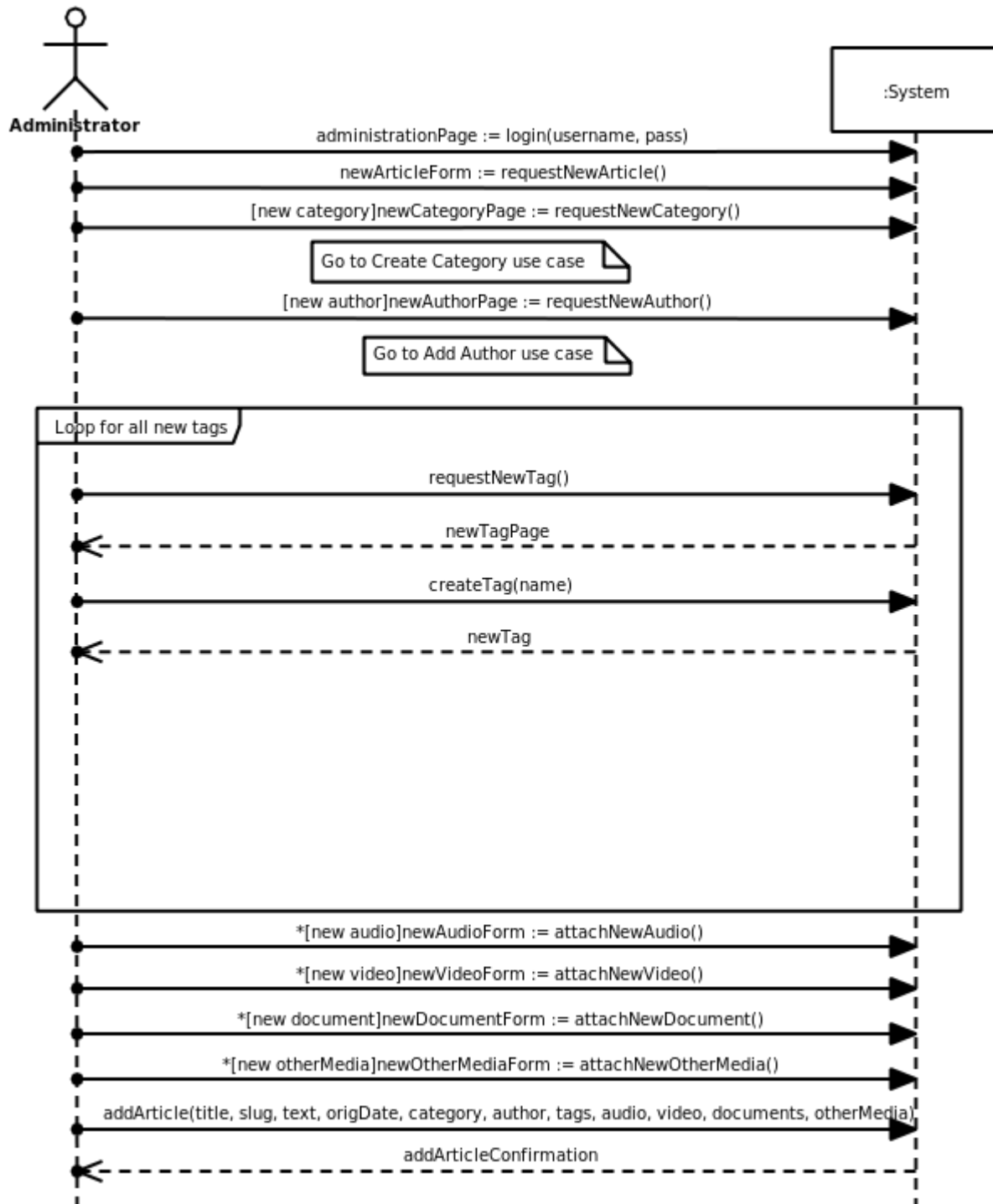
***View Recent Orders***

Administrator logs onto site and requests a report of all recent orders by date. System displays report.

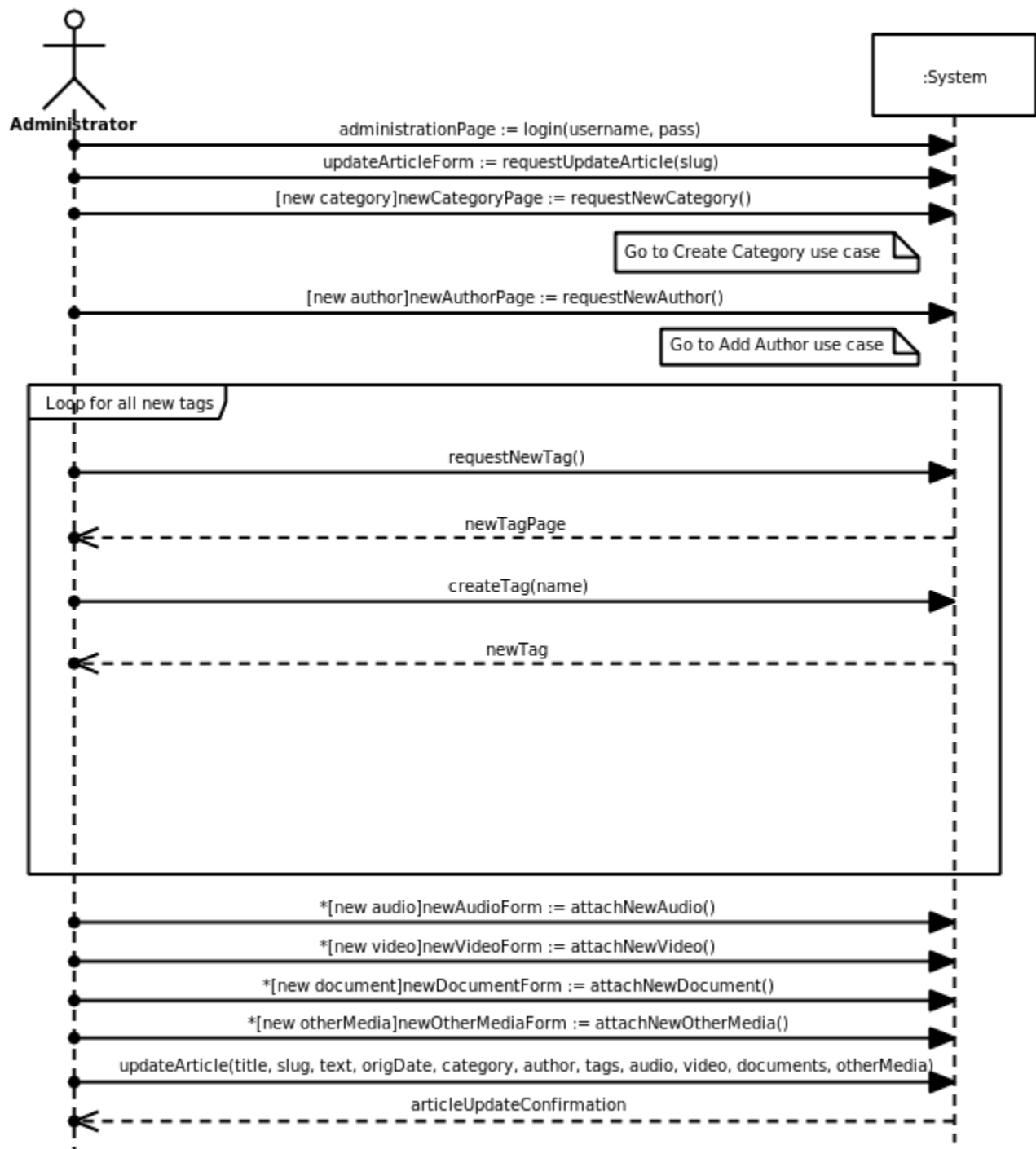


## System Sequence Diagrams

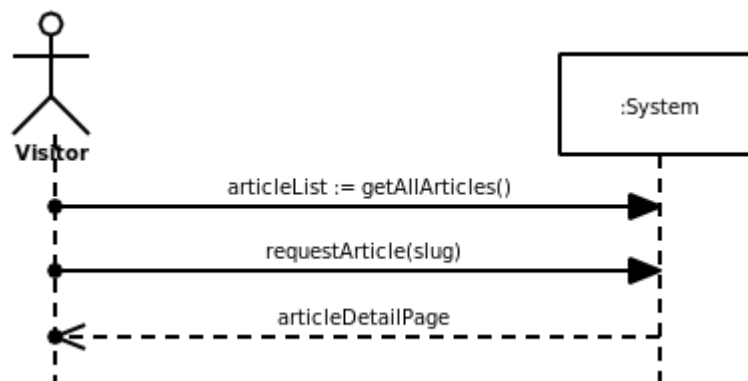
### Add Article



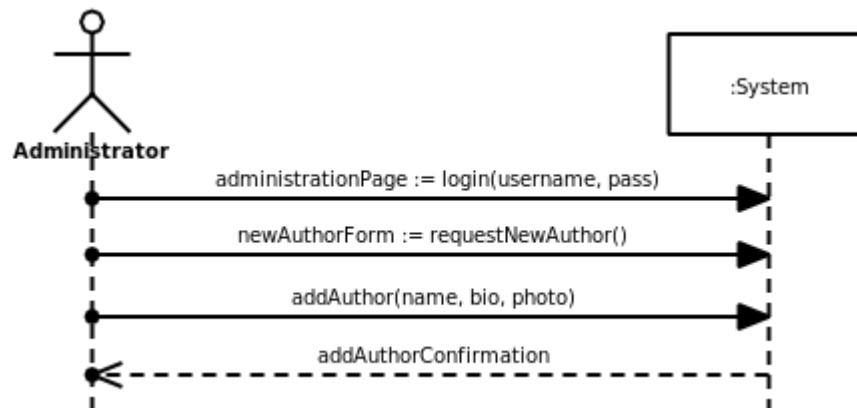
## Update Article



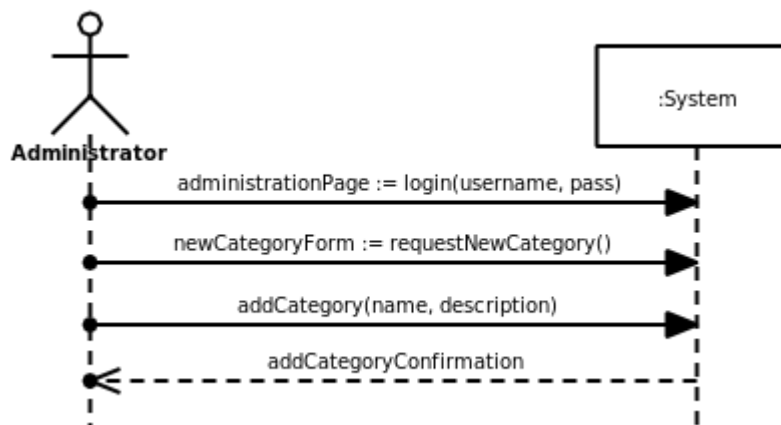
### View Article



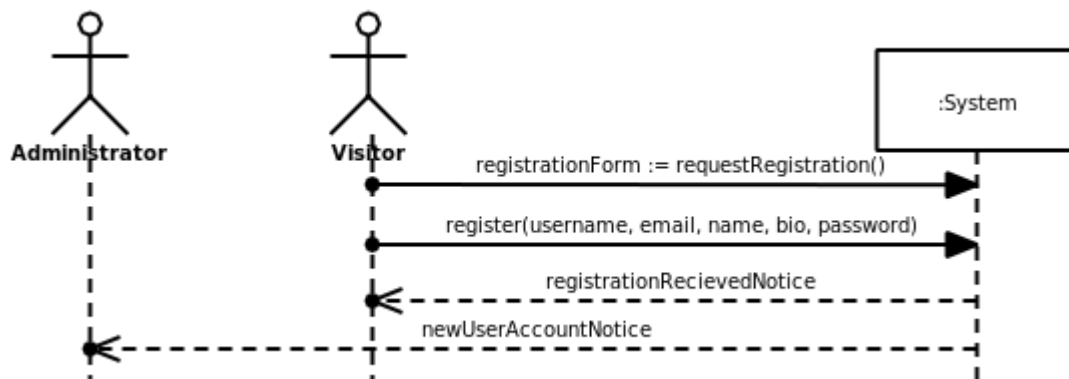
### Add Author



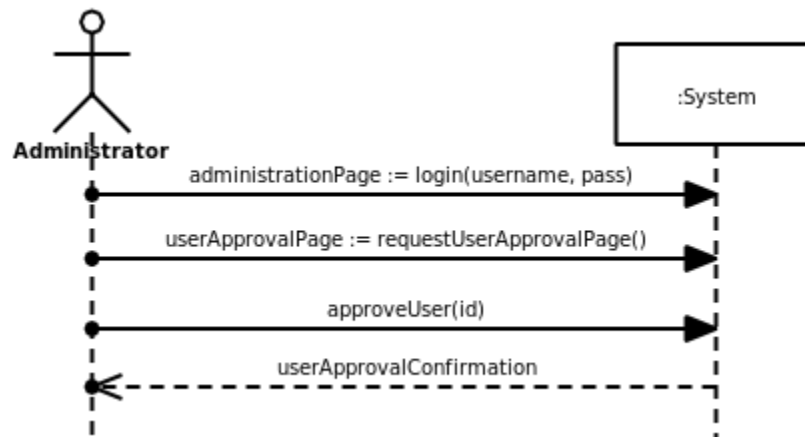
### Create Category



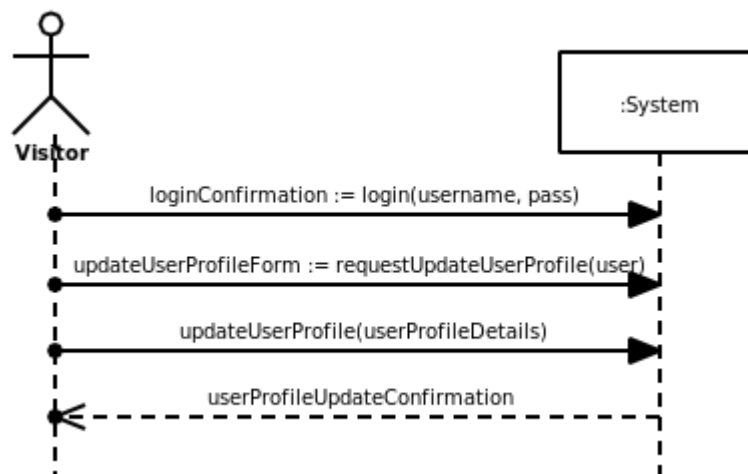
### Register User



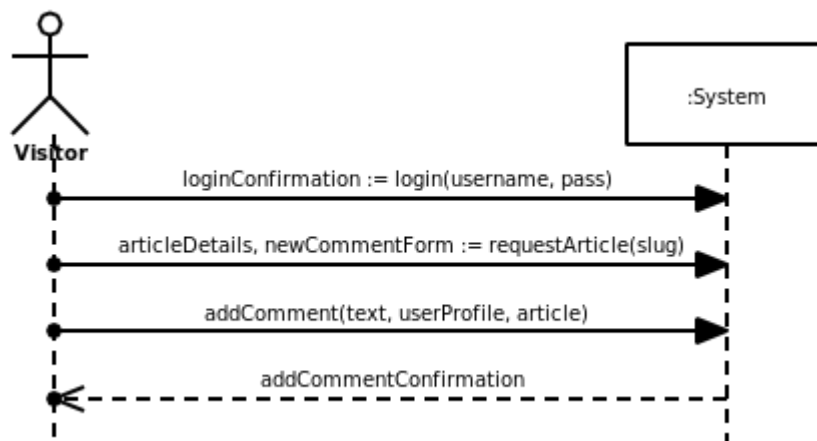
### Approve New User



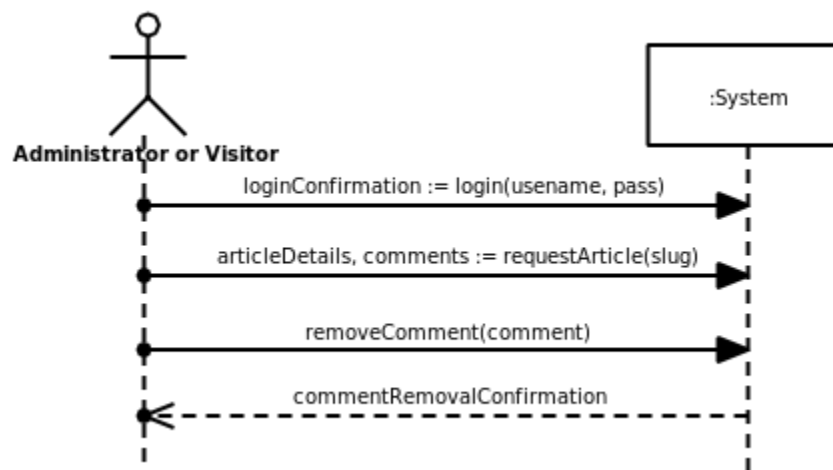
### Update User Profile



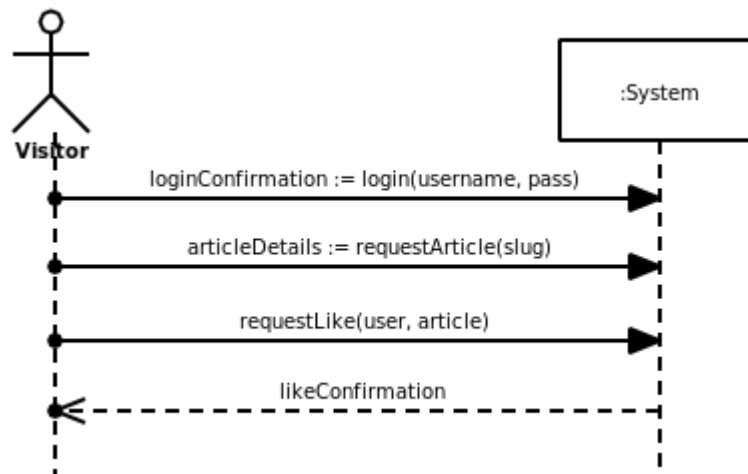
### Add Comment



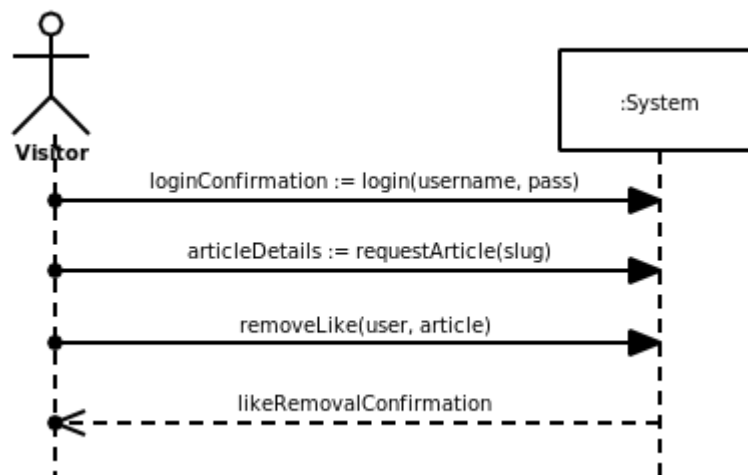
### Remove Comment



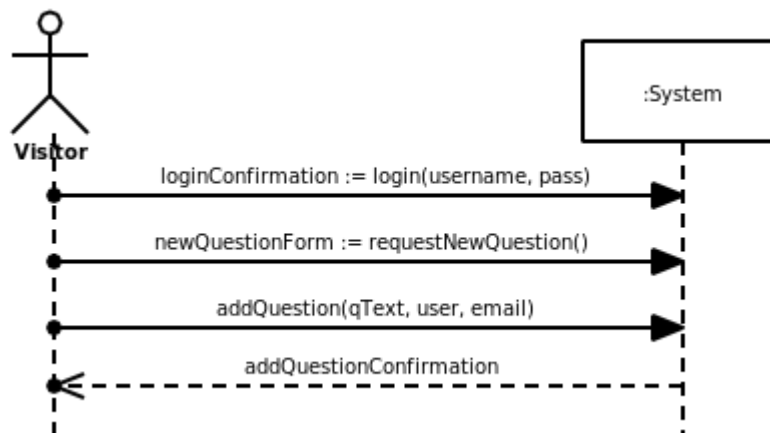
### Like An Article



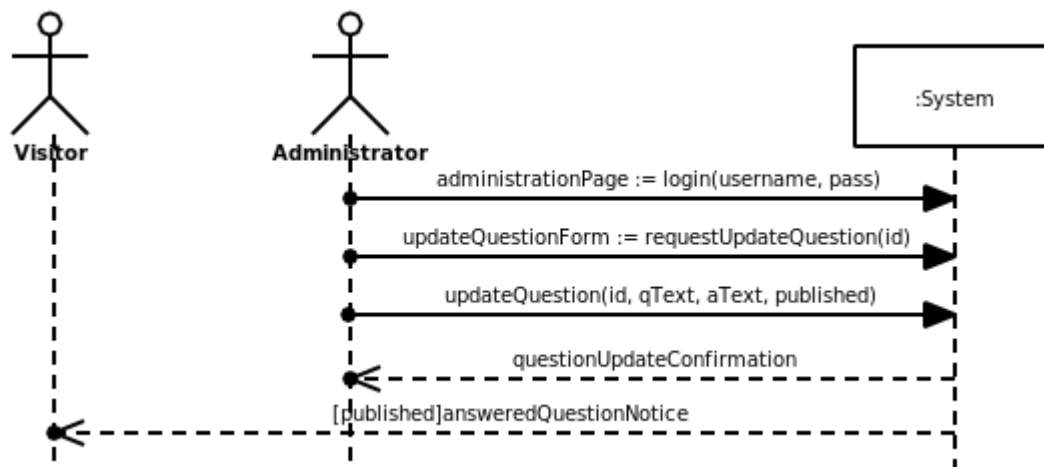
### Unlike An Article



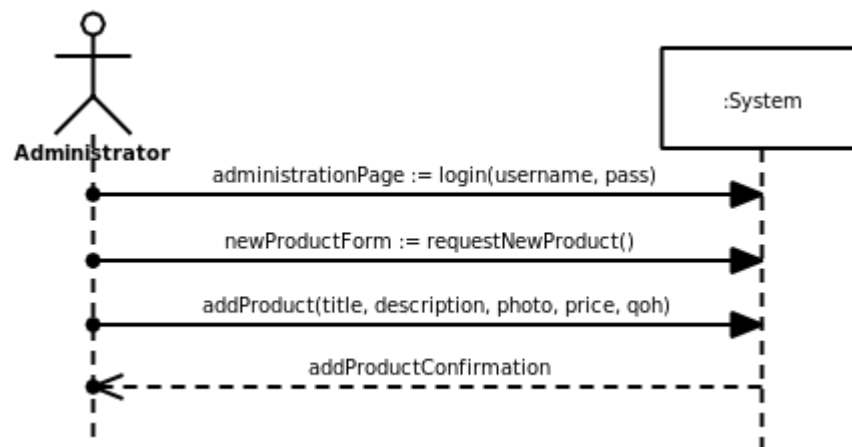
### Ask Question



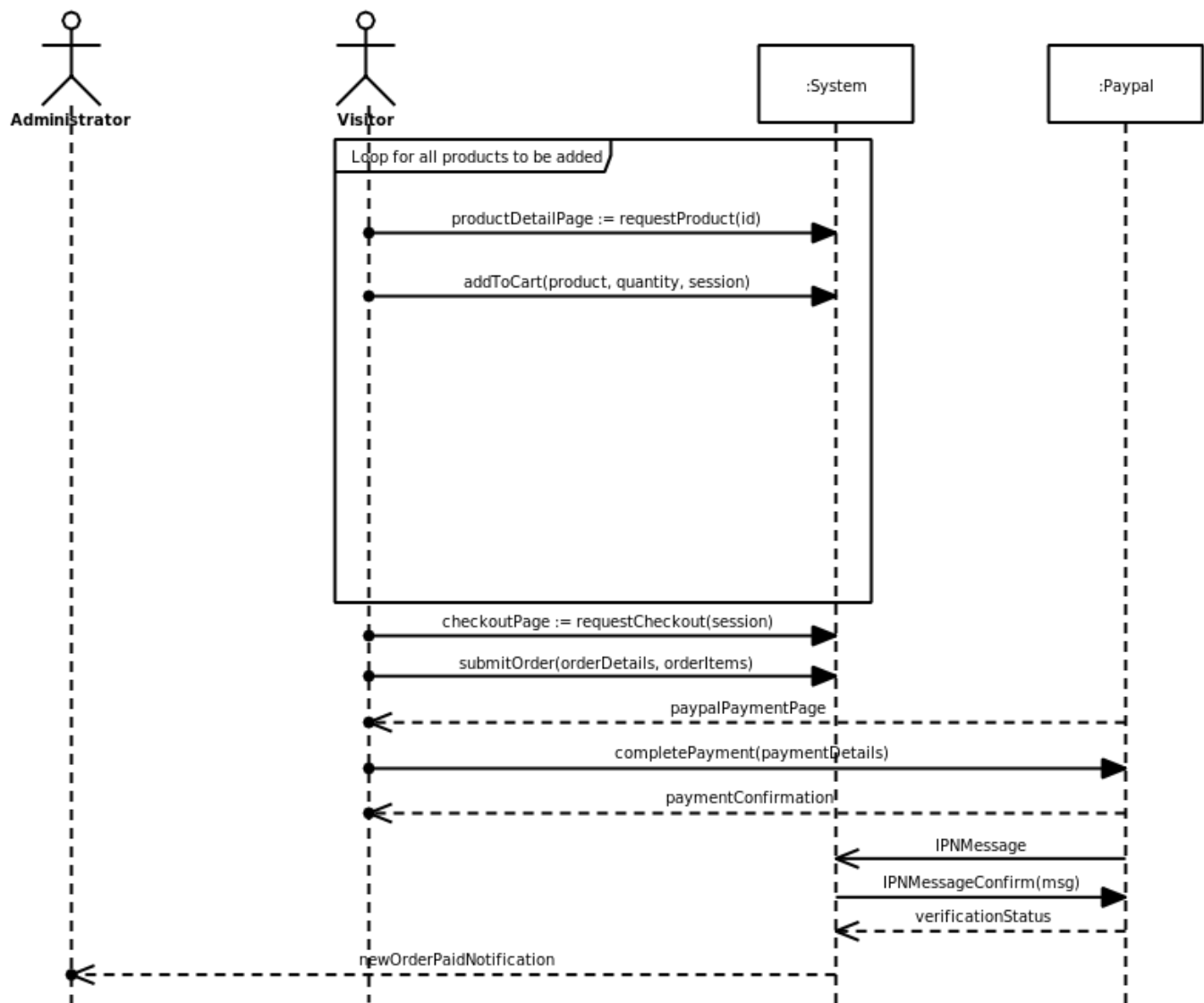
### Update Question



## Add Product

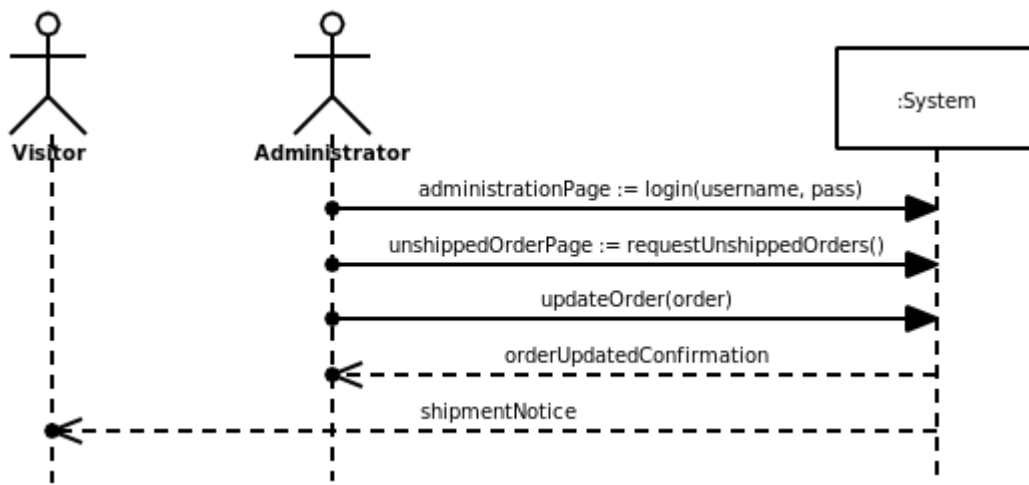


## Create New Order

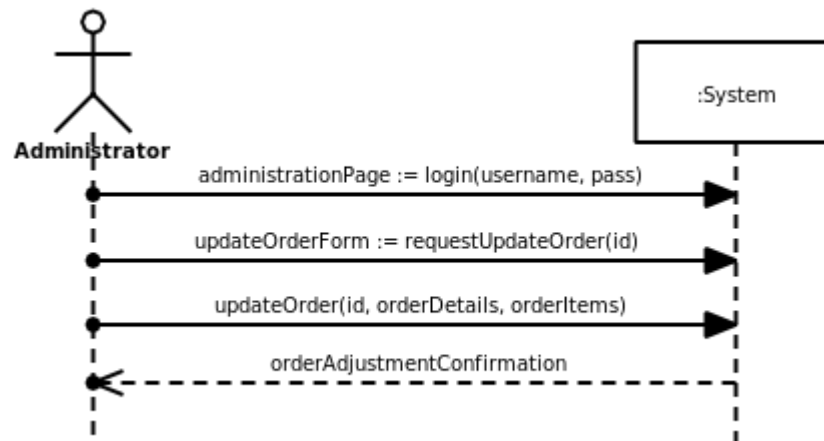




## Record Order As Shipped



## Adjust Order



**Attribute Definitions****Article**

Attribute	Meaning	Data Type	Rules
Title	The title of the article.	String	Required
Slug	A URL-friendly identifier for the article, based on the title.	String	Required
Text	Rich text content.	String	Optional
Date	Date the article was published on the site.	DateTime	Required, Timestamp
OrigDate	Date of the article's original publication.	Date	Required

**Audio**

Attribute	Meaning	Data Type	Rules
Title	The title of the audio file.	String	Required
Src	Path to the file on the server.	String	Required

**Video**

Attribute	Meaning	Data Type	Rules
Title	The title of the video file.	String	Required
Src	Path to the file on the server.	String	Either scr or url must be selected.
Url	URL to a YouTube video.	String	Either scr or url must be selected.

**Document**

Attribute	Meaning	Data Type	Rules
Title	The title of the document.	String	Required
Src	Path to the file on the server.	String	Required

**OtherMedia**

Attribute	Meaning	Data Type	Rules
Title	The title of the media file.	String	Required
Src	Path to the file on the server.	String	Required

## Category

Attribute	Meaning	Data Type	Rules
Name	Category name or title.	String	Required
Description	Rich text description of category.	String	Optional
Slug	A URL-friendly identifier for the category, based on the name.	String	Required

## Tag

Attribute	Meaning	Data Type	Rules
Name	Tag name/title.	String	Required
Slug	A URL-friendly identifier for the tag, based on the name.	String	Required

## Author

Attribute	Meaning	Data Type	Rules
Name	Author's actual name.	String	Required
Bio	Brief biography in rich text.	String	Optional
Photo	Path to a photo of the author.	String	Optional
Slug	A URL-friendly identifier for the author, based on the name.	String	Required

## UserProfile

Attribute	Meaning	Data Type	Rules
UserName	Username for the profile.	String	Required
Email	User's email address.	String	Required
Bio	A brief, rich text, "About Me" section.	String	Optional
Name	User's real name.	String	Required
Approved	Flag to indicate if the user is approved to post comments.	Bool	Required

## Comment

Attribute	Meaning	Data Type	Rules
Date	Date the comment was posted.	DateTime	Required, Timestamp
Text	Plain text body of the comment.	String	Required

## Like

Attribute	Meaning	Data Type	Rules
Date	Date the article was liked.	DateTime	Required, Timestamp

### Question

Attribute	Meaning	Data Type	Rules
Date	Date the question was entered.	DateTime	Required
qText	Plain text question body.	String	Required
aText	Rich text answer body.	String	Optional until published.
published	Flag to mark Question as published, otherwise draft.	Bool	Required
email	Email address if the visitor was not a registered user.	String	Required if not associated with a UserProfile object.

### Product

Attribute	Meaning	Data Type	Rules
Title	Name of product.	String	Required
Description	Rich text description of product.	String	Required
Photo	Path to photo of product.	String	Optional
Price	Product price.	Double	Required
QOH	Quantity on hand, for tracking inventory.	Int	Required

### OrderItem

Attribute	Meaning	Data Type	Rules
Quantity	Quantity of product ordered.	Int	Required

**Order**

Attribute	Meaning	Data Type	Rules
Date	Date order was placed.	DateTime	Required, Timestamp
Email	Contact e-mail address for visitors not logged in.	String	Optional, Required if not associated with a user.
Phone	Contact phone number.	String	Required
ShipName	Shipping Name if not using Paypal.	String	Optional
ShipAddr	Shipping Address if not using Paypal.	String	Optional
ShipCity	Shipping City if not using Paypal.	String	Optional
ShipState	Shipping State if not using Paypal.	String	Optional
ShipZIP	Shipping ZIP code if not using Paypal.	String	Optional
Instructions	Special instructions in plain text.	String	Optional
ShippingCost	Calculated shipping cost.	Double	Required, Calculated
Total	Calculated order total.	Double	Required, Calculated
Paid	Flag to indicate if order has been paid.	Bool	Required
Shipped	Flag to indicate if order has been shipped	Bool	Required
ShipDate	Date order was shipped.	DateTime	Optional, Timestamp
PaymentMethod	Indicates what payment method was used. Currently 2 options: "Check" or "Paypal".	String	Required, "Check" or "Paypal"

## **Alternate Design Concepts and Recommendation**

### ***Varying the scope of the system***

The scope of this system could be varied considerably. The business owner has many more modules and features he would like added on in the future. As a result, the scope could be expanded almost indefinitely. The scope could also be reduced since the e-commerce functionality is not something the client had listed as a priority.

### ***Varying the degree of automation***

The degree of automation could be changed by adding a desktop application that would help handle the conversion of various analog multimedia sources and automatically post the results to the site. The e-commerce section could become more automated by eliminating the manual processing of checks and automatically printing packing slips and printing labels. It could also be changed to require manual handling of all orders.

### ***Varying the type of technology used***

The application is currently using Django, but it could be changed to use Ruby on Rails, PHP, or ASP.NET. The application is web-based by nature, so converting it to a desktop application would not make much sense.

### ***Varying the approach to implementation***

The application could be developed as an add-on to an existing CMS platform such as Wordpress instead of creating a custom system.

### ***Project Alternatives***

1. Add religious discussion board and Gospel music internet radio to the scope.
2. Take e-commerce section out of scope.
3. Include desktop media conversion application.
4. Develop using Ruby on Rails

## ***Recommended Alternative***

I've chosen to go with this particular project scope because I feel the e-commerce section would be highly beneficial to the business and would help contribute a lot of NPV to the project. I also think that a religious discussion board and Gospel music internet radio app would expand the scope beyond what could be managed efficiently in this iteration without contributing very much of a benefit to the site's users. The desktop media conversion application would stretch the scope to much as well, with little improvement in function. I decided to go with Django instead of Ruby on Rails because I feel Django has a much richer set of platform features and I am more familiar with its use.

## **Design and Implementation to Date**

### ***Technology and Working Environment***

The application architecture I am using is a modified version of the MVC approach to web development unique to the Django platform.

The models, or data access takes place in models.py. No SQL strings are ever directly manipulated by user created code; the Django framework handles all of that. The structure of the classes is simply defined in the model and all the SQL string conversion are taken care of automatically based on the connection settings in the main settings.py file.

The controller code is mostly handled by the framework. However, there are configuration settings that manage the functionality of the controller layer. The most notable of these is the urls.py file which defines which view to display.

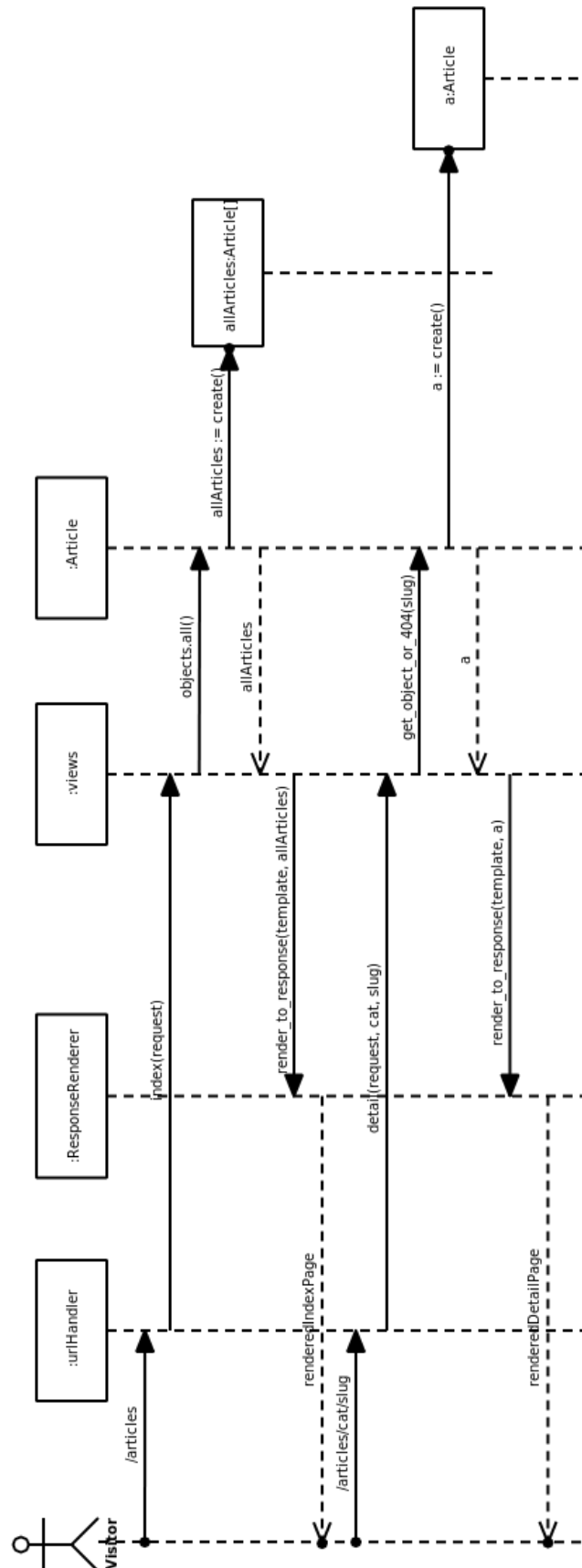
According to the Django's interpretation of MVC, the views.py file manages the view layer. This interpretation defines the view layer as making the decision of ***what*** to display, instead of ***how*** to display it. This logic would traditionally be found in the controller layer in other MVC frameworks, which is one area where Django differs.

The final piece of the puzzle is the template system. The templates define *how* the information is to be displayed, and consist of HTML files with template tags. However, Django does not consider these to be part of any of the three layers.

The code for the system will strongly object-oriented and dynamic thanks to the Python language.

I've provided a sequence diagram for the View Article use case to illustrate this structure as well as possible.

## View Article Extended Sequence Diagram






## Implementation to Date

### Use Cases Implemented

1. Add Article
2. View Article
3. Update Article
4. Delete Article
5. Add Author
6. Update Author
7. Delete Author
8. Create Category
9. View Category
10. Update Category
11. Delete Category

### Example Implementation

View Article Use Case



**Gospel Preaching.com**

**Articles**

**Q & A**
















**NT Reading**

**News**

**Contact Us**

### Articles

Some articles.

Title Author	Date	Original Date	Category Available Media
<b>Third Test</b> David Nichols	03/18/2010	03/18/2010	    
<b>Another test Article</b> David Nichols	03/18/2010	03/18/2010	    
<b>Test Article</b> David Nichols	03/18/2010	03/18/2010	    

*Article Index Page*



## Gospel Preaching.com

[Articles](#)[Q & A](#)[NT Reading](#)[News](#)[Contact Us](#)

### Test Article

Catagory:

Date Uploaded: 03/18/10

Original Date: 03/18/10

Author: David Nichols

This article is for testing.

#### Audio

Test Audio

[Download](#)

#### Video

a test video

[Download](#)

#### Document

Test Doc

[Download](#)

#### Other Media

Test Other


[Download](#)

Article Detail Page

**Django administration**Welcome, **David**. [Change password](#) / [Log out](#)

[Home](#) > [Articles](#) > [Articles](#)

Select article to changeAdd article +



Go

2010

Title	Author	Cat	Date published	Date originally created
<a href="#">Third Test</a>	David Nichols	test-category	March 18, 2010, 1:09 p.m.	March 18, 2010
<a href="#">Another test Article</a>	David Nichols	test-category	March 18, 2010, 1:09 p.m.	March 18, 2010
<a href="#">Test Article</a>	David Nichols	test-category	March 18, 2010, 1:05 p.m.	March 18, 2010

3 articles

### Administration Page Article List

Django administration

Welcome, David. Change password / Log out

Home » Articles » Articles » test-article

Change article

History

Title:

Test Article

Slug:

test-article

Cat:

test-category

+

Author:

David Nichols

Text:

This article is for testing.

**B** *I* U ABC ↺ ↻ ↺ ↻ ↺ ↻

Date originally created:

2010-03-18

Today

📅

Tags:

Testing

Hold down "Control", or "Command" on a Mac, to select more than one.

Audio

Title	Src	Delete?
test-article/Test Audio	Currently: article/2010_3/audio/test-article.mp3	<input type="checkbox"/>
<input type="text" value="Test Audio"/>	Change: <input type="text"/> <input type="button" value="Browse..."/>	
<input type="text"/>	<input type="text"/> <input type="button" value="Browse..."/>	

Videos

Title	Src	Url	Delete?
test-article/a test video	Currently: article/2010_3/video/test-article.AVI		<input type="checkbox"/>
<input type="text" value="a test video"/>	Change: <input type="text"/> <input type="button" value="Browse..."/>	<input type="text"/>	
<input type="text"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text"/>	

Documents

Title	Src	Delete?
test-article/Test Doc	Currently: article/2010_3/docs/test-article.txt	<input type="checkbox"/>
<input type="text" value="Test Doc"/>	Change: <input type="text"/> <input type="button" value="Browse..."/>	
<input type="text"/>	<input type="text"/> <input type="button" value="Browse..."/>	

Other media

Title	Src	Delete?
test-article/Test Other	Currently: article/2010_3/other/test-article.quotes	<input type="checkbox"/>
<input type="text" value="Test Other"/>	Change: <input type="text"/> <input type="button" value="Browse..."/>	
<input type="text"/>	<input type="text"/> <input type="button" value="Browse..."/>	

✖ Delete

Server: localhost Database: bluevan\_gospel Table: articles\_article

Showing rows 0 - 2 (3 total, Query took 0.0002 sec)

```
SELECT *
FROM articles_article
LIMIT 0, 30
```

Profiling [ Edit ] [ Explain SQL ] [ Create PHP Code ] [ Refresh ]

Show : 30 row(s) starting from record # 0 in horizontal mode and repeat headers after 100 cells

Sort by key: None

+ Options

	id	title	slug	cat_id	author_id	text	date	orig_date
<input type="checkbox"/>	1	Test Article	test-article	1	1	<p>This article is for testing.</p>	2010-03-18 13:05:55	2010-03-18
<input type="checkbox"/>	2	Another test Article	another-test-article	1	1	<p>Yes, another test.</p>	2010-03-18 13:09:23	2010-03-18
<input type="checkbox"/>	3	Third Test	third-test	1	1	<p>The last test.</p>	2010-03-18 13:09:47	2010-03-18

Check All / Uncheck All With selected: [ Edit ] [ Delete ]

Show : 30 row(s) starting from record # 0 in horizontal mode and repeat headers after 100 cells

Query results operations

Print view Print view (with full texts) Export CREATE VIEW

### Article Database View

See Appendix A for code excerpts for this use case. Due to the nature of the Django framework, the models.py file serves as the database schema.

### Initial Menu Hierarchy

- Articles
  - Categories
  - Tags
  - Authors
- Questions and Answers
  - Ask a Question
- Users
- Book Store
  - Products
  - Cart
- Login

- My Profile
  - Update User Profile
  - Delete User
- Administration
  - Articles
    - Add Article
    - Update Article
    - Delete Article
  - Authors
    - Add Author
    - Update Author
    - Delete Author
  - Categories
    - Create Category
    - Update Category
    - Delete Category
  - Users
    - Unapproved Users
    - Delete User
  - Questions
    - Update Question
    - Delete Question
  - Products
    - Add Product
    - Update Product
    - Delete Product
  - Orders
    - Unshipped Orders
    - Adjust Order
    - Recent Orders

**Report Samples****Article Report by Category****Gospel Preaching.com****Articles**

Q &amp; A

NT Reading

News

Contact Us

**Test Category**

This category is for testing.

<u>Title</u> <u>Author</u>	<u>Date</u>	<u>Original Date</u>	Available Media
<b>Third Test</b> David Nichols	03/18/2010	03/18/2010	
<b>Another test Article</b> David Nichols	03/18/2010	03/18/2010	
<b>Test Article</b> David Nichols	03/18/2010	03/18/2010	

**Article Report by Author****Gospel Preaching.com****Articles**

Q &amp; A

NT Reading

News

Contact Us

**David Nichols**

It's me!



<u>Title</u> <u>Author</u>	<u>Date</u>	<u>Original Date</u>	Available Media
<b>Third Test</b> David Nichols	03/18/2010	03/18/2010	
<b>Another test Article</b> David Nichols	03/18/2010	03/18/2010	
<b>Test Article</b> David Nichols	03/18/2010	03/18/2010	

## Project Monitoring/Reporting To Date

Disciplines	Activities	Tasks	Projected	1/14 - 1/20	1/21 - 1/27	1/28 - 2/3	2/4 - 2/10	2/11 - 2/17	2/18 - 2/24	2/25 - 3/3	3/4 - 3/10	3/11 - 3/17	Total
Business Modeling	Understand the business environment	Interview stakeholders.	3	1			0.5	0.5					2
	Evaluate existing architecture.		1	0.5									0.5
	Create the system vision	Generate a list of primary business benefits.	0.5				0.5	0.5					0.5
	Develop a list of system capabilities.		2	1									1.5
	Create business models	Identify business events.	1.5	0.5			0.5						1
	Define information and data flow models		0.5	0.5			0.5						1
			53	3	0	0	1	0.5	2.5	0	7	4.5	18.5
	Gather detailed information	Analyze current system.	3	1								0.5	1.5
	Interview stakeholders.		4	2			1	0.5					3.5
	Define functional requirements	Create event decomposition table.	2								1		1
Requirements	Create use case diagram.		2								1		1
	Write use case descriptions.		8								4	3	7
	Create problem domain class diagram.		2								1		1
	Create interaction diagram.		1									0.5	0.5
	Define nonfunctional requirements												
	Define security requirements.		2									0.5	0.5
	Define reliability requirements.		2										0
	Define technological requirements.		2										0
	Define usability requirements.		2										0
	Prioritize requirements												
Design	Determine core requirements.		3										0
	Schedule requirements across iterations.		2										0
	Develop user interface dialogs		8										0
	Create storyboards.		4						2				2
	Determine if changes to existing design are needed.								0.5				0.5
	Requirements with users		3										0
	Interview stakeholders with requirements.		3										0
	Revise requirements as needed.		3										0
			67	0.5	2	0	0.5	0	0.5	0	0	1	4.5
	Design the support services architecture and deployment environment	Plan production server setup.	3	0.5			0.5						0.5
Design	Find or design acceptable hosting solution.		4										0.5
	Design the software architecture	Establish model view controller architecture details.	4		1							1	2
	Design package diagrams.		3										0
	Design use case realizations		5										0
	Create design class diagram.		12										0
	Create sequence diagrams.												0
	Design the database												0
	Finalize domain class diagram.		2										0
	Specify relationships between domain classes.		2										0
	Design the system and user interfaces	Review storyboards.	3										0
Design	Design view logic.		5										0.5
	Design Dialog templates.		3										0
	Design RSS and social networking interfaces.		8						0.5				1
	Review interface design details with stakeholders.		3										0
	Design the system security and controls												0
	Determine security groups and access levels.		8										0
	Design login page.		2										0
													0
													0
													0



Disciplines	Activities	Tasks	Projected	1/14 - 1/21	1/21 - 1/28	1/28 - 2/4	2/4 - 2/11	2/11 - 2/17	2/18 - 2/24	2/25 - 3/3	3/4 - 3/10	3/11 - 3/17	Total
Implementation	Build software components	Build models.	69	0	4.5	6.5	0	0	3	0	0	3	17
		Build view logic.	20		0.5	2.5						2	5
		Build templates.	10		0.5	1						1	2.5
		Write URL configuration files.	20		0.5	1			2				3.5
		Write settings files.	5		0.5	0.5							0.5
	Acquire software components	Download and install python libraries.	5		0.5	0.5							1
		Download and install <u>any</u> needed <u>Django</u> modules.	2		1	1							1
		Download and install <u>any</u> needed <u>Django</u> modules.	2		1	1							1
	Integrate software components	Configure deployment software on test server.	1		0.5	0.5			0.5				1
		Deploy full system on test server.	2		0.5	0.5			0.5				1
Testing	Define and conduct unit testing	Define unit testing.	40	0	0	0	0	0	0	0	0	0	0
		Conduct core function unit tests.	5										0
		Conduct support function unit tests.	5										0
		Define and conduct integration testing	4										0
		Define integration tests.	5										0
	Define and conduct usability testing	Conduct integration tests.	8										0
		Review design specifications against implementation.	2										0
		Conduct usability testing with users.	3										0
	Define and conduct user acceptance testing	Have administrators test administration functions.	5										0
		Review non-administrative functions with users.	3										0
Deployment	Acquire hardware and system software	Purchase hosting package or setup server hardware.	31	0	0	0	0	0	0	0	0	0	0
		Install and configure system software as needed.	5										0
		Install and configure system software as needed.	5										0
	Package and install components	Deploy finished system to production server.	1										0
		Configure system on production server.	1										0
	Train users	Develop training documentation for system.	5										0
		Hands-on training.	5										0
	Convert and initialize data	Add initial users to system.	1										0
		Import existing information.	8										0
	Project Management	Evaluate the project's scope and risk	22.5	3.5	0	0	8.5	0.2	0.2	0.2	0.2	0.2	13
	Evaluate the project's scope and risk	Create initial problem domain class list.	1	0.75			0.25						1
		Create initial primary use case list.	1	0.75			0.25						1
		Review project scope and evaluate risks and threats.	2				1						1
	Confirm the project's feasibility	Calculate economic feasibility.	1				1						1
		Evaluate technical feasibility.	1				1						1
		Evaluate schedule feasibility.	1				1						1
	Develop the project and iteration schedules	List out activity tasks.	2				2						2
		Review with Dr. Satzinger.	0.5										0
	Monitor and control the project's iterations	Keep log of time spent on tasks.	5				1	0.2	0.2	0.2	0.2	0.2	2
		Create periodic reports for Dr. Satzinger.	8	1			2						3

Disciplines	Activities	Tasks	Projected	1/14 - 1/21	1/21 - 1/28	1/28 - 2/4	2/4 - 2/11	2/11 - 2/17	2/17 - 2/24	2/24 - 2/25	2/25 - 3/3	3/4 - 3/10	3/11 - 3/17	Total
Configuration and Change Management			8	0	2	0.25	0	0	0.5	0	0	0	0	2.75
	Develop change control procedures.		1		1									1
	Setup SVN procedures.		2		1				0.5					1.5
	Develop plan for release to test & production servers.													
	Manage models and software components		2											0
	Update models as needed.		3			0.25								0.25
	Keep proper, detailed revision documentation.		16	0.5	2	1	1	0.5	0	0	0	1	1	6
Environment														
	Select and configure the development tools		1		1									1
	Install python and Django on development machine.		1		1									1
	Create a SVN repository.		0											0
	Install system software on test server.		2			1								1
	Configure test server.													
	Tailor the UP development process		3				0.5						1	1.5
	Setup of deliverables.		3	0.5			0.5	0.5						1.5
	Meeting with Dr. Satzinger.													
	Provide technical support services		3											0
	Provide support to test users		3											0
	Provide support for development environment.													
<b>Total</b>			<b>316</b>	<b>11</b>	<b>10.5</b>	<b>7.75</b>	<b>13</b>	<b>2.2</b>	<b>6.7</b>	<b>0.2</b>	<b>7.2</b>	<b>9.7</b>		<b>68.25</b>

## Appendix A: Code Excerpts

### *models.py*

```

from django.db import models
from datetime import date
from tinymce import models as tinymce_models
import os.path
from django.conf import settings

def get_filename(instance, filename):
    split = filename.split('.')
    ext = ''.join(['.', split[len(split) - 1]])
    month = ''.join([str(date.today().year), '_', str(date.today().month)])
    slug = instance.parent.slug
    t = (None, "audio")[instance(instance, Audio)]
    t = (t, "video")[instance(instance, Video)]
    t = (t, "docs")[instance(instance, Document)]
    t = (t, "other")[instance(instance, OtherMedia)]
    fn = ''.join(['article/', month, '/', t, '/', slug, ext])
    i = 1
    if instance.src:
        instance.src.delete()
    while True:
        if os.path.exists(''.join([settings.MEDIA_ROOT, fn])):
            fn = ''.join(['article/', month, '/', t, '/', slug, '_', str(i), ext])
        else:
            return fn

def convert_youtube(url):
    sep = url.find('?v=')
    if sep > 0:
        url = 'http://www.youtube.com/v/%s' % url[sep + 3:]
    sep = url.find('&')
    if sep > 0:
        url = url[:sep]
    return url

class Category(models.Model):
    name = models.CharField(max_length=200)
    slug = models.SlugField()
    description = tinymce_models.HTMLField(blank=True)
    class Meta:
        verbose_name_plural = "categories"
        ordering = ['name']
    def __unicode__(self):
        return self.slug

class Author(models.Model):
    name = models.CharField(max_length=200)
    slug = models.SlugField()
    bio = tinymce_models.HTMLField(blank=True)
    photo = models.FileField(upload_to=get_filename, null=True, blank=True)
    class Meta:
        verbose_name_plural = "authors"
        ordering = ['name']
    def __unicode__(self):
        return self.name

class Tag(models.Model):
    name = models.CharField(max_length=200)

```

```
slug = models.SlugField()
class Meta:
    verbose_name_plural = "tags"
    ordering = ['name']
def __unicode__(self):
    return self.name

class Article(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField()
    cat = models.ForeignKey('Category')
    author = models.ForeignKey('Author')
    tags = models.ManyToManyField(Tag, related_name='articles')
    text = tinymce_models.HTMLField(blank=True)
    date = models.DateTimeField('date published', auto_now_add=True)
    orig_date = models.DateField('date originally created')
    class Meta:
        ordering = ['-date']
    def __unicode__(self):
        return self.slug

class Audio(models.Model):
    parent = models.ForeignKey('Article')
    title = models.CharField(max_length=200)
    src = models.FileField(upload_to=get_filename, null=True, blank=True)
    class Meta:
        verbose_name_plural = "audio"
        ordering = ['parent', 'title']
    def __unicode__(self):
        return ''.join([self.parent.slug, "/", self.title])

class Video(models.Model):
    parent = models.ForeignKey('Article')
    title = models.CharField(max_length=200)
    src = models.FileField(upload_to=get_filename, null=True, blank=True)
    url = models.URLField(null=True, blank=True)
    class Meta:
        ordering = ['parent', 'title']
    def save(self):
        self.url = convert_youtube(self.url)
        super(Video, self).save()
    def __unicode__(self):
        return ''.join([self.parent.slug, "/", self.title])

class Document(models.Model):
    parent = models.ForeignKey('Article')
    title = models.CharField(max_length=200)
    src = models.FileField(upload_to=get_filename, null=True, blank=True)
    class Meta:
        ordering = ['parent', 'title']
    def __unicode__(self):
        return ''.join([self.parent.slug, "/", self.title])

class OtherMedia(models.Model):
    parent = models.ForeignKey('Article')
    title = models.CharField(max_length=200)
    src = models.FileField(upload_to=get_filename, null=True, blank=True)
    class Meta:
        verbose_name_plural = "other media"
        ordering = ['parent', 'title']
    def __unicode__(self):
        return ''.join([self.parent.slug, "/", self.title])
```

**views.py**

```

from django.shortcuts import render_to_response, get_object_or_404
from gospel_preaching.articles.models import Article, Category
from django.template import RequestContext
from django.core.paginator import Paginator
from django.conf import settings

def index(request):
    order = request.GET.get('order', '-date')
    page = request.GET.get('page', 1)
    try:
        perpage = settings.ARTICLES_PER_PAGE
    except AttributeError:
        perpage = 10
    pager = Paginator(Article.objects.all().order_by(order), perpage)
    return render_to_response('articles/index.html', {'all_articles': pager.page(page).object_list,
'order': order, 'page': pager.page(page), 'pager': pager}, context_instance =
RequestContext(request))

def detail(request, cat, slug):
    a = get_object_or_404(Article, slug__exact=slug)
    return render_to_response('articles/detail.html', {'article': a}, context_instance =
RequestContext(request))

def cat(request, cat):
    order = request.GET.get('order', '-date')
    page = request.GET.get('page', 1)
    try:
        perpage = settings.ARTICLES_PER_PAGE
    except AttributeError:
        perpage = 10
    c = get_object_or_404(Category, slug__exact=cat)
    pager = Paginator(c.article_set.all().order_by(order), perpage)
    return render_to_response('articles/cat.html', {'cat': c, 'articles':
pager.page(page).object_list, 'order': order, 'page': pager.page(page), 'pager': pager},
context_instance = RequestContext(request))

```

**urls.py**

```

from django.conf.urls.defaults import *

# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('',
    # Example:
    # (r'^gospel_preaching/', include('gospel_preaching.foo.urls')),

    # Uncomment the admin/doc line below and add 'django.contrib.admindocs'
    # to INSTALLED_APPS to enable admin documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # (r'^admin/(.*)', admin.site.root),
    (r'^$', 'gospel_preaching.articles.views.index'),
    (r'^(?P<cat>[a-zA-Z0-9_-]+)/$', 'gospel_preaching.articles.views.cat'),
    (r'^(?P<cat>[a-zA-Z0-9_-]+)/(?P<slug>[a-zA-Z0-9_-]+)/$', 'gospel_preaching.articles.views.detail'),
)

```

## **admin.py**

```
from gospel_preaching.articles.models import Article, Audio, Video, Document, OtherMedia,
Category, Author, Tag
from django.contrib import admin

class AudioInline(admin.TabularInline):
    model = Audio
    extra = 1

class VideoInline(admin.TabularInline):
    model = Video
    extra = 1

class DocInline(admin.TabularInline):
    model = Document
    extra = 1

class OtherInline(admin.TabularInline):
    model = OtherMedia
    extra = 1

class ArticleAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("title",)}
    search_fields = ['title', 'text']
    date_hierarchy = 'date'
    list_display = ('title', 'author', 'cat', 'date', 'orig_date')
    inlines = [AudioInline, VideoInline, DocInline, OtherInline]

class CategoryAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("name",)}

admin.site.register(Article, ArticleAdmin)
admin.site.register(Audio)
admin.site.register(Video)
admin.site.register(Document)
admin.site.register(OtherMedia)
admin.site.register(Category, CategoryAdmin)
admin.site.register(Author)
admin.site.register(Tag)
```

***index.html***

```

{% extends "base.html" %}
{% load custom %}

{% block css %}
{% endblock %}
{% block script %}
{% endblock %}

{% block main %}
<h2>{{ cur_app.header }}</h2>
{{ cur_app.description|safe }}
    {% if all_articles %}
        <div class="list_table">
            <div class="list_item_odd">
                <span class="list_title"><a href="{% orderurl 'title' order
%}">Title</a></span>
                <span class="list_cat"><a href="{% orderurl 'cat__title' order
%}">Category</a></span><br />
                <span class="list_author"><a href="{% orderurl 'author' order
%}">Author</a></span>
                <span class="list_dates_media">
                    <span class="list_date"><a href="{% orderurl 'date' order
%}">Date</a></span>
                    <span class="list_orig_date"><a href="{% orderurl 'orig_date' order
%}">Original Date</a></span>
                    <span class="list_media">Available Media</span>
                </span>
            </div>
            {% include "helpers/navigation.html" %}
            {% for article in all_articles %}
                <div class="list_item_{{ cycle 'even' 'odd' % }}">
                    <span class="list_title"><a href="{% url
gospel_preaching.articles.views.detail article.cat.slug article.slug %}">{{ article.title }}</a></span>
                    <span class="list_cat"><a href="{% url gospel_preaching.articles.views.cat
article.cat.slug %}">{{ article.cat.title }}</a></span><br />
                    <span class="list_author">{{ article.author }}</span>
                    <span class="list_dates_media">
                        <span class="list_date">{{ article.date|date:"m/d/Y" }}</span>
                        <span class="list_orig_date">{{ article.orig_date|
date:"m/d/Y" }}</span>
                        <span class="list_media">
                            
                            
                            
                            
                            
                        </span>
                    </span>
                </div>
            {% endfor %}
            {% include "helpers/navigation.html" %}
        </div>
    {% else %}
        <p>No articles are available.</p>
    {% endif %}
{% endblock %}

```

## detail.html

```
{% extends "base.html" %}

{% block css %}
<link rel="stylesheet" href="{{ MEDIA_URL }}style/video.css" type="text/css" />
{% endblock %}
{% block script %}
<script type="text/javascript" src="{{ MEDIA_URL }}scripts/audio-player.js"></script>
<script type="text/javascript" src="{{ MEDIA_URL }}scripts/flowplayer-3.1.0.min.js"></script>
<script type="text/javascript" src="{{ MEDIA_URL }}scripts/flowplayer.controls-3.0.2.js"></script>
{% endblock %}

{% block main %}
    <h2>{{ article.title }}</h2>
    <table class="header_block">
        <tr><td>Catagory:</td><td><a href="{% url gospel_preaching.articles.views.cat
article.cat.slug %}">{{ article.cat.title }}</a></td></tr>
        <tr><td>Date Uploaded:</td><td>{{ article.date|date:"m/d/y" }}</td></tr>
        <tr><td>Original Date:</td><td>{{ article.orig_date|date:"m/d/y" }}</td></tr>
        <tr><td>Author:</td><td>{{ article.author.name }}</td></tr>
    </table>
    {{ article.text|safe }}
    {% if article.audio_set.all %}
        <div class="content_block">
            <h3>Audio</h3>
            {% for audio in article.audio_set.all %}
                <h5>{{ audio.title }}</h5>
                <object id="{{ article.slug }}" height="24" width="290"
data="{{ MEDIA_URL }}scripts/player.swf" type="application/x-shockwave-flash">
                    <param value="{{ MEDIA_URL }}scripts/player.swf" name="movie"/>
                    <param value="playerID={{ article.slug }}&soundFile={{ MEDIA_URL }}
{{ audio.src }}" name="FlashVars"/>
                    <param value="high" name="quality"/>
                    <param value="false" name="menu"/>
                    <param value="transparent" name="wmode"/>
                </object><br />
                <a href="{{ MEDIA_URL }}{{ audio.src }}">Download</a>
            {% endfor %}
        </div>
    {% endif %}
    {% if article.video_set.all %}
        <div class="content_block">
            <h3>Video</h3>
            {% if article.video_set.all %}
                {% for video in article.video_set.all %}
                    <h5>{{ video.title }}</h5>
                    {% if video.url %}
                        <object width="425" height="344" style="display:block;">
                            <param name="movie"
value="{{ youtube.url }}"&hl=en&fs=1&rel=0"></param>
                            <param name="allowFullScreen" value="true"></param>
                            <param name="allowscriptaccess" value="always"></param>
                            <embed src="{{ youtube.url }}"&hl=en&fs=1&rel=0"
type="application/x-shockwave-flash" allowscriptaccess="always" allowfullscreen="true" width="425"
height="344"></embed>
                        </object>
                    {% endif %}
                    {% if video.src %}
                        <a
                            href="{{ MEDIA_URL }}{{ video.src }}"
                            style="display:block;width:425px;height:344px"
                            id="player{{ video.id }}">
                        </a>
                        <!--HTML based control bar: <div id="controls{{ video.id }}"
class="video_controls"></div-->
                        <script>
                        <!--
                            flowplayer("player{{ video.id }}",
```



```

"{{ MEDIA_URL }}scripts/flowplayer-3.1.0.swf", {
    clip: {
        autoPlay: false,
        autoBuffering: true
    },
    plugins: {
        controls: {
            url:
'{{ MEDIA_URL }}scripts/flowplayer.controls-3.1.0.swf',
            backgroundColor: '#000000',
            backgroundGradient: '[0.2, 1.0, 0.6]',
            buttonColor: '#707070',
            sliderColor: '#707070',
            bufferColor: '#A0A0A0',
            progressColor: '#FF0000',
            buttonOverColor: '#A0A0A0',
            volumeSliderColor: '#707070',
            timeBgColor: '#707070',
            timeColor: '#FF1010',
            tooltipColor: '#707070'
        }
    }
});
//-->
</script>
<a href="{{ MEDIA_URL }}{{ video.src }}">Download</a>
{% endif %}
{% endfor %}
{% endif %}
</div>
{% endif %}
{% if article.document_set.all %}
<div class="content_block">
<h3>Document</h3>
{% for doc in article.document_set.all %}
<h5>{{ doc.title }}</h5>
<a href="{{ MEDIA_URL }}{{ doc.src }}">Download</a>
{% endfor %}
</div>
{% endif %}
{% if article.othermedia_set.all %}
<div class="content_block">
<h3>Other Media</h3>
{% for media in article.othermedia_set.all %}
<h5>{{ media.title }}</h5>
<a href="{{ MEDIA_URL }}{{ media.src }}">Download</a>
{% endfor %}
</div>
{% endif %}
{% endblock %}

```

## Legal Notes



This work by David Nichols is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. To view a copy of this license, visit <<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>>. Some rights reserved.

GospelPreaching.com and the associated globe logo are Copyright © 2009 by Contending for the Faith Publications. All rights reserved.



The code included herein is Copyright © 2010 by David Nichols.

The code files included in this documentation are part of GospelPreaching.com.

GospelPreaching.com is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

GospelPreaching.com is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Please see <<http://www.gnu.org/licenses/gpl.html>> for a copy of the GNU General Public License.

## Other Notes

I want to give credit to the amazing Django web framework and thank the amazing people that work on it. "It lets you build high-performing, elegant Web applications quickly." Please check out the Django project's site: <<http://www.djangoproject.com/>>

