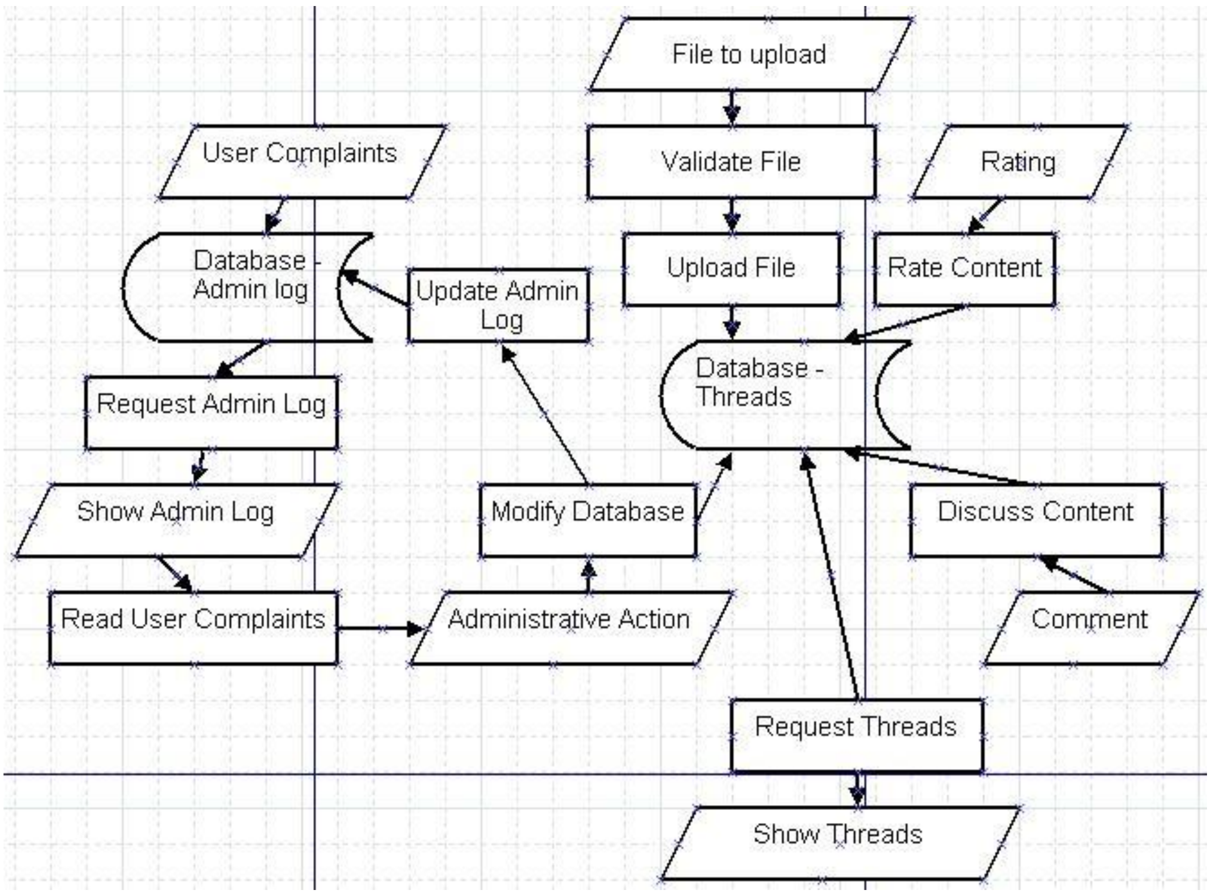
2.1 Overall System Design

The system consists of several inputs which are responsible for the user upload, download, and rating of uploaded content on the user side, and inputs for the administration and control of that information on the administration/moderation side. This will be entirely a point-and-click web interface which will not require extensive knowledge to manipulate, although there may be some typing in the administrative interface for selecting and comments are typed by users.

The minimum hardware requirements required to run my system are a graphics card capable of running a 1024x768x16 display, with 16MB of VRAM, and sufficient memory to operate a graphical web browser.

2.1.1 System Flow Charts



2.2 Description of modular structure of system.

The modular structure of this system is in that the entire system can easily be unpacked without extensive pre-configuration, as long as the server has PHP 5 and MySQL installed the installer PHP file will create the relevant databases to operate correctly, and also create an administrator account to allow for further bespoke setup based on client needs. Also the administrative section is not dependant on the threads database being present truly and as such can technically be changed or replaced entirely as long as it hooks back into the interface at the expected points.

2.3 Definition of Data Requirements.

2.3.1 Design Data Dictionary

DOCUMENTATION OF DATA STORED IN DATABASES:

Name	Description	Type	Notes

UID	Unique Identifier of the user.	Long integer	Automatically Incrementing
Name	Username of the user.	Variable Character	Must be unique and shorter than 32 characters
Email	Email address of the user.	Variable Character	Must be unique and shorter than 256 characters
PWord	User's Password	Binary Blob	Encrypted
Loc	User's present location	Variable Character	Shorter than 256 characters
Avatar	User's avatar (name of file INCLUDING extension)	Variable Character	Shorter than 256 characters
PostCount	Amount of threads created by the user.	Long integer	
NetRating	Present rating of the user (based on ratings of feedback)	Integer	
PermissionLevel	Level of permissions available to the user	Integer	

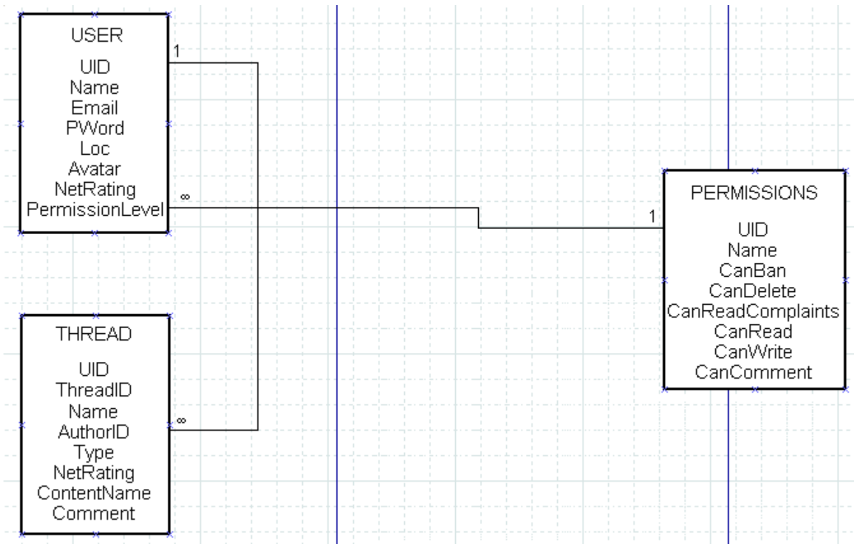
UID	Unique identifier of the thread.	Long Integer	Automatically Incrementing
ThreadID	Identifier of the thread this thread was created in response to.	Long Integer	-1 if there is no responding thread.
Name	Name of the thread	VariableCharacter	Shorter than 256 characters
AuthorID	Identifier of the author of the thread	Long Integer	refers to entity of USER/UID
Type	The "type" of thread.	Short Integer	(0 is a comment/contentless, 1 is an image post, 2 is a video post)
NetRating	The present rating of the thread.	Integer	
ContentName	Actual name of the file involved	Variable Character	(can be blank if THREAD/Type is 0)
Comment	A message for the thread	Variable Character	(must not be empty if THREAD/Type is 0)

UID	Unique identifier of the permissions configuration.	Long Integer	Auto-incrementing
Name	Name of the permissions configuration.	Variable Character	Must be shorter than 256 characters
CanBan	Can a user with this configuration ban another user?	Boolean	
CanDelete	Can a user with this configuration remove a	Boolean	

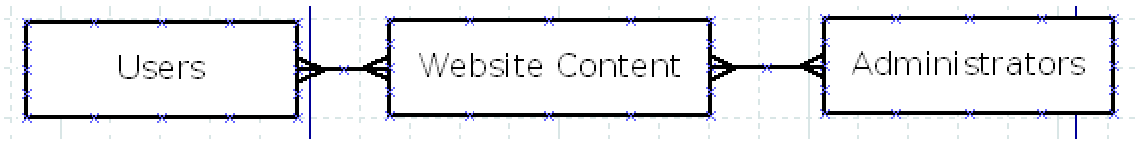
	thread?		
CanReadComplaints	Can a user with this configuration read the reports log?	Boolean	
CanRead	Can a user with this configuration view content in the site?	Boolean	
CanWrite	Can a user with this configuration create threads and rate?	Boolean	

2.3.2 Database Design

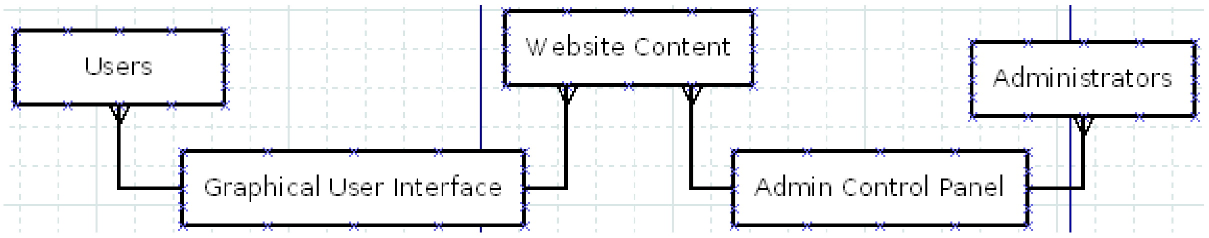
2.3.2.1 Normalised relations



2.3.2.2 Entity Relationship Diagram



2.3.2.3 Resolved Entity Relationship Diagram



2.3.2.4 Description of entities

Users: Users are the users of the system who do not have administrative rights or duties.

Graphical User Interface: The GUI allows the Users to access (and manipulate) the Website Content.

Website Content: The Website Content is the data that is outputted and affected to and by users, controlled affectively by the administrators, and is the primary attraction of the site.

Admin Control Panel: The Admin Control Panel is how Administrators will effect the Website Content, allowing control over the data in there effectively. This is modular and will be supplied by my end-user, Anthony.

Administrators: Administrators control the website content, using the admin control panel to do this.

2.3.3 Identification of appropriate storage media

Appropriate storage media for everything in the web site is the web server itself, in a MySQL database, and this will be sufficient for the present needs for the website and leave room for expansion as the website grows in popularity (and as such used upload/download bandwidth increases)

2.4 Identification of Processes and Suitable Algorithms for Data Transformation

Registration Algorithm

Enter Name

Enter Email

Enter Password

Generate Encryption Key:

Pass name through UUED encryption

For every character in UUED Name {

Convert character to ordinal Value

Multiply ordinal value of character by ordinal value of corresponding character in "ABRAXAS" (Looping around to character 1 if UUED name is longer than 7 characters)

Add UID value + character position to the ordinal value of character

Convert character back to ASCII

Get SHA1 value of password.

AES encrypt the SHA1 value of password using the previously generated encryption key as the key.

Store in database Username, Email Address and AES encrypted SHA1 hashed password.

Login Algorithm

```
Enter Name
Enter Password
Generate Encryption Key:
Pass name through UUED encryption
For every character in UUED Name {
Convert character to ordinal Value
Multiply ordinal value of character by ordinal value of corresponding character in “ABRAXAS” (Looping around to character 1 if UUED name is longer than 7 characters)
Add UID value + character position to the ordinal value of character
Convert character back to ASCII }
Get SHA1 value of password.
AES encrypt the SHA1 value of password using the previously generated encryption key as the key.
Compare resultant AES Encrypted value with the encrypted value stored with the Username.
```

Upload Form Algorithm

Read uploading config file:

Find conf.cfg and open it for reading

Make 3 passes to find values for

[begin] indicates opening of the config data

[end]   indicates ending of the config data

allowdropdown (=TRUE) indicates whether or not to show the multiple locations menu.

[locmenu] indicates start of list of upload locations

[/locmenu] indicates end of list of upload locations

imgmaxsize (=2048) indicates, in kilobytes, the max upload size

vidmaxsize (=20480) indicates, in kilobytes, the max upload size

truncsize (=TRUE) indicates whether or not to cap upload size

trunctype (=FALSE)indicates whether or not to restrict filetypes uploaded.

[imgtypetrunc] indicates start of whitelist of MIME types

[/imgtypetrunc] indicates end of whitelist of MIME types

[vidtypetrunc] indicates start of whitelist of MIME types

[/vidtypetrunc] indicates end of whitelist of MIME types

#   is a comment, and indicates the rest of the line can be ignored.

First pass:

Finds [begin],[end] and checks for [locmenu],[/locmenu],[typetrunc] and [/typetrunc].

The lines of these are then stored as integers.

Second pass:

flags NOT between two headers are read and stored as parameters.

Third pass:

Contents of areas between headers are read and stored as parameters.

Send all parameters to the page and create the upload dialog accordingly.

Upload Parse Algorithm

Read uploading config file:

Find conf.cfg and open it for reading

Make 3 passes to find values for

[begin] indicates opening of the config data

[end]   indicates ending of the config data

allowdropdown (=TRUE) indicates whether or not to show the multiple locations menu.

[locmenu] indicates start of list of upload locations

[/locmenu] indicates end of list of upload locations

imgmaxsize (=2048) indicates, in kilobytes, the max upload size

vidmaxsize (=20480) indicates, in kilobytes, the max upload size

truncsize (=TRUE) indicates whether or not to cap upload size

trunctype (=FALSE)indicates whether or not to restrict filetypes uploaded.

[imgtypetrunc] indicates start of whitelist of MIME types

[/imgtypetrunc] indicates end of whitelist of MIME types

[vidtypetrunc] indicates start of whitelist of MIME types

[/vidtypetrunc] indicates end of whitelist of MIME types

#   is a comment, and indicates the rest of the line can be ignored.

First pass:

Finds [begin],[end] and checks for [locmenu],[/locmenu],[typetrunc] and [/typetrunc].

The lines of these are then stored as integers.

Second pass:

flags NOT between two headers are read and stored as parameters.

Third pass:

Contents of areas between headers are read and stored as parameters.

Upload file to a temporary location on the server.

Send all parameters to the page and examine if the temporary data is valid.

If it is valid, copy it to the appropriate location, depending on filetype.

If it is invalid, return the user to the upload screen indicating as such.

Delete the temporary file.

Send the uploaded name and filetype to the message poster.

Post Message Algorithm

First, make sure the user has permission to post a message.

If he does, take the user’s topic name, as well as a comment (if he does not supply an image or video, this is needed), and an image or video file if they have specified that they are uploading one.

If they have uploaded an image, make sure the image is of type .jpg, .png or .gif, and that the filesize is not abusively large, before moving the image to folder /userimages/

If they have uploaded a video, examine the filesize (too large a file and the php execution time will timeout, and the recommended setting for this is 300 seconds) and if not too large, convert them to FLV using ffmpeg, move the converted file to /uservideos/ and discard the original.

Create the new thread, containing information as to the user’s identity, the topic name, the type of thread it is and the name of any files uploaded with the thread, as well as the comment if supplied, and set all other entries to their initial values.

Show Threads Algorithm

Check to make sure the user has permission to read threads.

If they do, check which thread to jump to, signified by the start variable

Connect to the database, and find the next ten threads, putting them into a numerical array, and outputting their topic name, with a hyperlink to view the topic, author, with a hyperlink to view the author's information, and rating.

Using the content type, output an image based on it, signifying text, image, or video.

Continue doing this until the ten threads have been outputted.

Output hyperlinks to the next and previous pages, assuming they exist, adjusting the start variable by +10 for the next page, and -10 for the previous.

View Thread Algorithm

Check the user is logged in and allowed to view topics.

Connect to the database.

Find the topic the user selected.

Find the author of that topic.

Find the author's permission level.

Output the topic's title, comment, and image or video content if it exists, as well as netrating.

Output the author's name, permission title, avatar and postcount, as well as his rating.

Call the **recursive** replies function, with the selected topic's UID and the value 0 as the arguments.

REPLIES FUNCTION:

Find the topics that replied to the topic specified in the first argument.

Find the authors of those topics.

Find those author's permission levels.

Output those topic's titles, comments, and images or video content if they exist, as well as netratings.

Output those author's names, permission titles, avatars and postcounts, as well as their ratings.

Call the **recursive** replies function, with the topic currently being outputted's UID and the value (previous value of this argument + 1) as the arguments.

<u>Rating Content Algorithm</u>

Check if the user has the permissions to view topics

Check if the “rate” variable was equal to ‘up’ or ‘down’.

Check that this user has not already rated this thread, checking the ratings table for the compound key of the user’s UID and the thread’s UID.

If all the above are true, then find the thread in the threads table that matches the UID of the variable “thread”

If the variable “rate” was equal to “up”, increment the thread’s netrating value by one.

If the variable “rate” was equal to “down”, decrement the thread’s netrating value by one.

Redirect the user back to the rated thread.

2.5 User Design Rationale

2.5.1 Sample of Planned Data Capture and Entry Designs [SCAN]

2.5.2 Sample of Planned Valid Output Designs [SCAN]

2.6 Description of measures planned for security and integrity of data

The database will be kept secure by eNom's range of security tools and automated anti-cracking/spyware/malware tools and anti SQL/MySQL Injection tools.

The passwords of users will be kept secure via 128bit AES encryption, which is the most cryptographically secure encryption available in a standard MySQL installation (meaning that minimal porting will be needed if in the future the present server is rendered unusable for any reason)

2.7 Description of measures planned for system security

The servers are owned by eNom, a well-known company based in Redmond, Washington and second largest domain registrar in the world, and are well-trusted for security and the company backups their servers frequently meaning that in the case of a disaster the damage to the system is limited and can be reinstated at a later date with a minimal amount of effort required by the end user.

2.8 Overall test strategy in relation to the problem being solved and tested.

My test strategy will be to use black box testing to test my system’s inputs and outputs, white box testing to test the variables contained within my system, smoke testing to check that the system “works”, that there are no programming errors rather than logic errors.