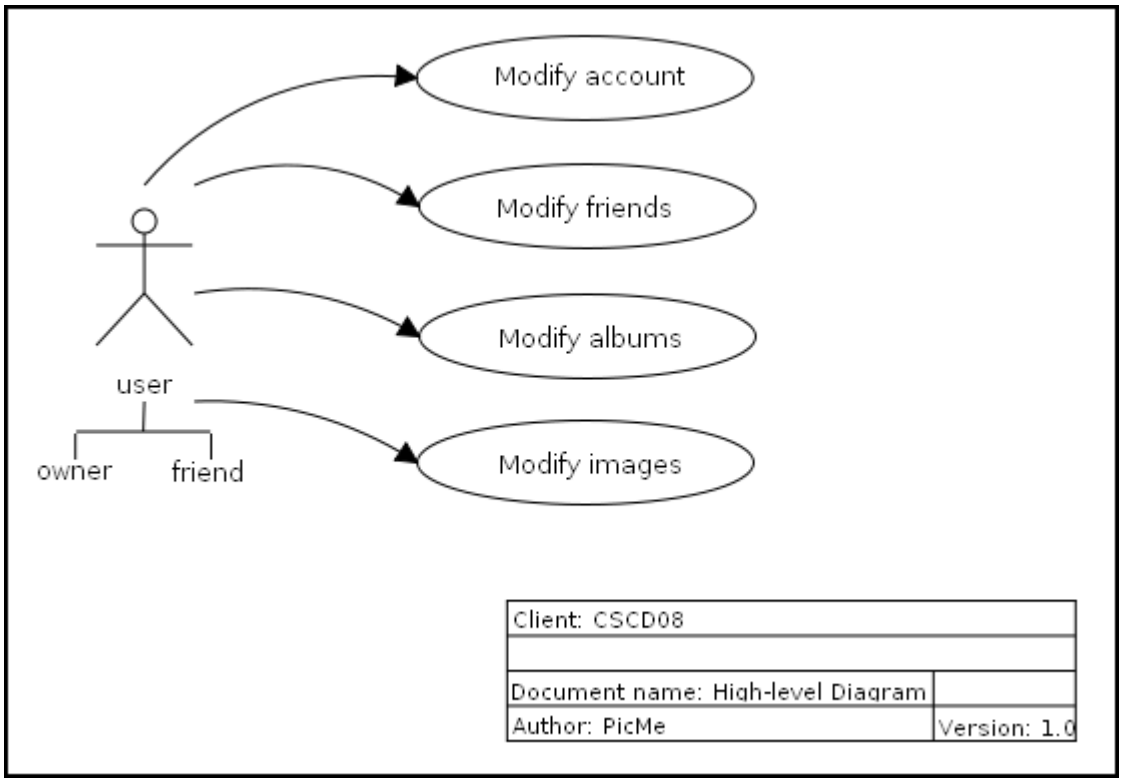
1. Functional Requirements

1.1 Base Features

The following is a description of precisely what the user should be able to accomplish with each feature, and exactly what functionality the developers will implement. The following high level diagram is to be referenced well reading sections 1.1.2 through 1.1.6



1.1.1 Images Features

Below is an outline of features that directly modify image files or assist with their modifications. The specifications of these features are also provided. To access these features the user must login to their account, select their album page, select a specific picture (or sometimes more than one picture), and select edit to open the photo manipulation applet. Once the applet is open, all of these features are available to the users through a toolbar available on the side of the user interface.

Undo/Redo
These two features are support features for other image manipulation functionality that is to be implemented. If a user performs one of the transformations outlined below then they may revert to the previous state of the picture before that transformation occurred. This allows for the user to correct errors they may have made while editing a picture. The redo function works similar to that of undo. A redo will allow the user to return to the previous state of a picture directly before an undo operation was performed.

Crop/Resize
These two features allow the user to modify the size of the picture or modify what is contained inside the actual photo. For simplicity, the user will be able to select a crop tool and drag the frame of a rectangle across the actual photo manipulation applet which indicates the area the user wishes to have cropped. For the resize feature, the user will be provided scroll bars (one for each of vertical and horizontal resizing) which will allow for the image to be stretched in either direction.

Collage/Scrapbook
This feature will allow for the combination of two pictures into one single file. Though this is considered an image feature it is currently planned to be implemented outside the current editor tool. Two pictures will be combined directly from an album, this results in the creation of a new picture file with each of the two pictures placed side by side. The user will then be able to edit the new picture file through the use of the tool to perform additional modifications.

Brightness/Contrast/Sharpness/Scale
These image processing features are straightforward and will be explained briefly, they are accessible through the use of scrolls bars as well. Each bar will be labeled separately and the user may be able to adjust each of these values simply by scrolling to the left or right. In other words brightness, contrast, grayscale, and sharpness are increased by scrolling to the right, or decreased by scrolling to the right. These changes are immediately applied to the picture as the user is scrolling.

Tagging
(Note: Tagging preformed through the editor tool is described below. Initial tagging done when a photo is uploaded is described under album functionality)
The user will have the option of adding tags to certain pictures in the editor tool. They may select a certain area of the picture through the use of a tagging tool, and attach a certain text keyword to that area. This tag can then be searched using the search features provided by the user interface.

1.1.2 Accounts

Accounts are the only method through which a user will have access to the entire system. The desired functionality is outlined below and is also displayed on the attached use case diagram. In addition, the account use case diagram makes reference to the 'find user details' and 'validate user' use cases, which aid with the implementation of the functionality described below.

1.1.2.1 Account Creation/Deletion

Account creation is achieved through the user interface on the main page. As mentioned above, a user must create an account, and log in to it, in order to access any of the main functionality. Account creation will be quick and simple. To create an account the user simply needs to fill in 4-5 text fields. These text fields will allow for the retrieval of desired username/password/email and other initial information that is deemed necessary. After account creation a confirmation email will be sent to the email provided. The user must confirm in order to activate the account fully; otherwise login will not be possible. After confirmation, the user's information will need to pass authentication before that user is allowed to log in.

Account deletion is accomplished by logging into a created account and selecting the option which will be available in the option lists to the left of the screen. To help prevent with accidental deletion the user will also need to input a complex text string to verify the deletion of the account. Afterwards all files associated with that account are removed from the system and that user is removed from the database. All traces of that account are also removed from the system upon deletion (i.e. Automatic removal of this account from other user's friends lists).

1.1.2.2 General Account Functions

Login/ Logout
An account is clearly not of much use if a user cannot use it to log into the system. One of the more primitive features of an account is simply to allow a user to login to the system and access all pictures and albums saved under that account name.

Alternatively a user that has logged in will also need the ability to logout. This function will need to be made available in multiple areas of the user interface for easy access.

Update details / email
Certain information is attached to an account beyond that of user name, password, and email. A user's first name, last name, age, time-zone, and other additional information will also be saved and tied to a specific account. Since a user's information may change at a later date, it is necessary that a user be able to update that information. Through the use of this option the user will be directed to a page listing all currently stored details with an ability to modify each.

Change password
To aid with account security a user will also have the option to modify his password. The user must first input his old password and then the new desired password, twice, before the change is finalized. Afterwards an email will be sent to the email attached to the account to notify the user of the change.

1.1.3 Albums

This section outlines the functionality of our Albums section.

Create album
This function allows the user to create a folder in which he or she will store photos. A default name will be given to created folders (i.e. Album1, Album2, etc.) along with predefined permission levels which the user will be allowed to change with the "Edit Album" function.

Upload Image
This function allows the user to browse his or her own computer for image files to upload to the server (Refer to Album Use Case). Users may only upload image files of file extension .jpeg. Any other image file type will give the user a notification of our supported image file type. After the user has selected an image for upload, he or she may write descriptions or write tags for the image for other user's to see or for personal purposes.

Delete Image

This function allows the user to browse his or her images in whatever album he/she chooses. The user will be allowed to browse for the image in selected album and, after chosen, the user will be notified if he/she 'really wants to delete the image?'(Refer to Album Use Case). If the user selects to do so, the image will be deleted from the album and from the database entirely, if not, the user will be returned to the Albums section with the image intact. This function is restricted to the owner wanting to delete his own image only.

Browse Album

This function allows the user to browse existing images stored on the server, inclusive to the pictures in the albums with viewing permission levels set to include him/her (Refer to Album Use Case). The user will be able to see the picture along with any photo tags or descriptions the picture has alongside it.

Download Image

This function allows the user to download an edited image file from the server to his or her own computer. An image downloaded to a user's computer will be saved in .jpeg format and cannot be resized during download. This function will only be applicable to pictures in albums whose permission rights include that of the user attempting to download.

Search Image

This function allows the user to search for an image stored on the server by entering keyword(s) into a search box on the album's page. The "Tags" section of the database will be searched for matching keyword(s) and the results will be shown to the user on a page in a list ordered lexically or by similarity, as chosen by the user. The images listed will be pictures in albums which the user has viewing permission rights to.

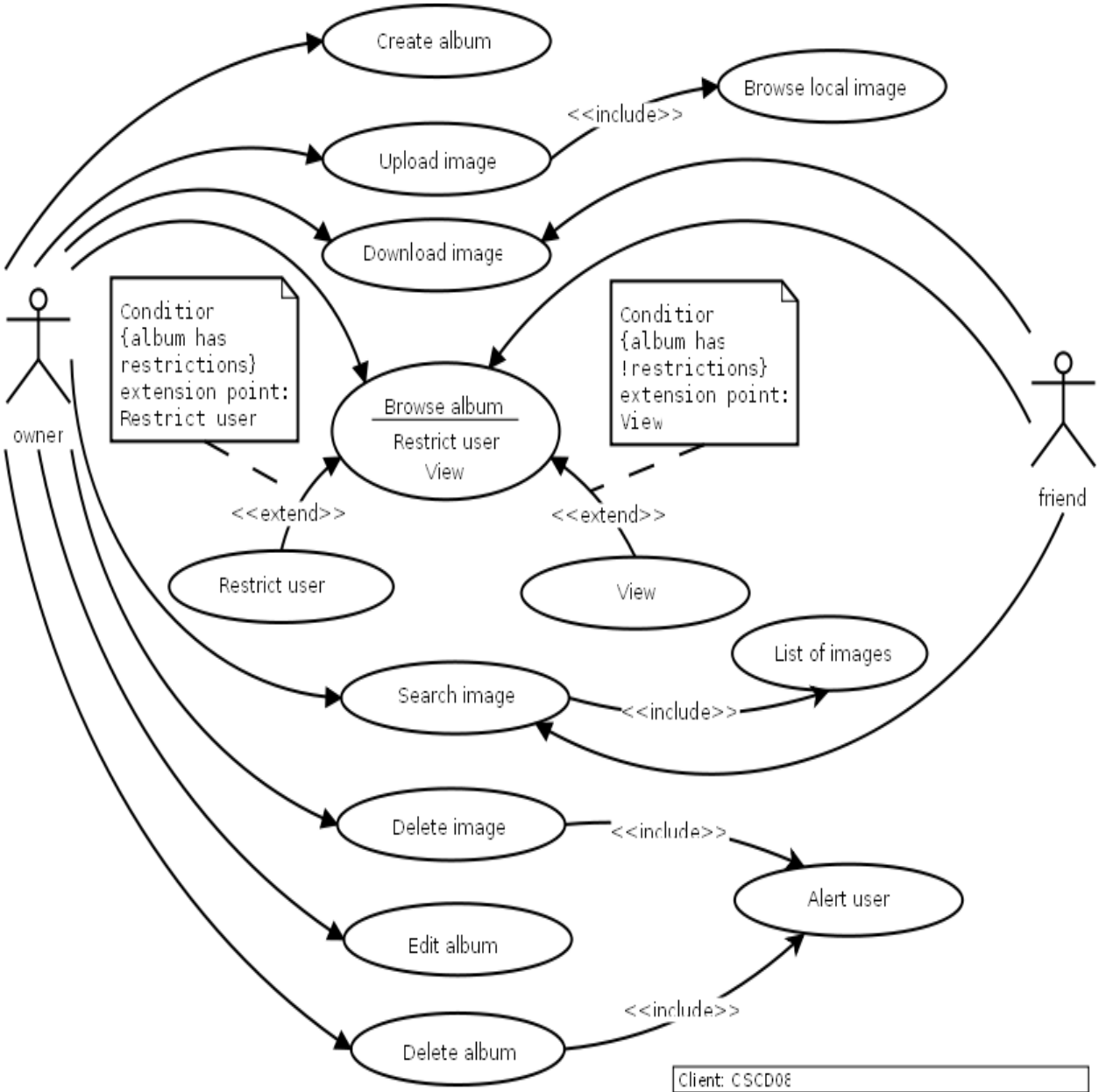
Edit Album

This function, which allows the user to edit the details of an album, is inclusive to solely the owner of the album (Refer to Album Use Case). Album details which can be changed are the name of the album and permission rights, pertaining to whom shall be allowed to view that certain album.

Delete Album

This function, which allows the user to delete and album, is inclusive to solely the owner of the album (Refer to Album Use Case). Upon use of this function, the user will be prompted with a notification as to whether he or she 'really wants to delete this album?'(Refer to Album Use Case). If the user selects to do so, the album and all pictures within will be deleted from the server, if not, the user will be returned to the Albums section with album and pictures intact.

1.1.4 Album Use Case



1.1.5 Group Editing

Interpretation
The group editing functionality is clearly an important part of this system. The design specification requests that multiple users be able to modify a picture simultaneously. This poses an interesting problem. The team chose to strongly consider the context in which the photo manipulation applet was being implemented. This context is that of a social networking site and not that of a large software engineering firm. When a user stores images on this site the intention is to display those pictures to friends and possibly allow for modifications to those pictures. There is no formal organized process with which 'work' is performed on those pictures. If changes are discarded on this system it is not nearly as much of an issue as it is for a firm who may lose a large portion of money over the same situation.

The team also feels that ultimately the decision to allow for changes to be made to a picture belongs to the user who 'owns' that picture. At the same time, we feel that allowing two users to edit the same picture together is an interesting function which may draw attention to the site.

Implementation
We have divided this implementation into two parts which combined provide the full functionality of our interpretation.

Single User Edit
When a user edits a picture it is considered a single user edit by default. A copy of the picture is made and sent to the editor for modification. The master picture is untouched during this editing process. When the user is done editing this picture it is 'delivered' to the owner of that specific picture. The owner will now receive a notification on his main login page, and when the master picture is highlighted (within the album) the proposed edited copy is presented to the owner who may then select to update the master. Multiple proposed edits may be delivered to the owner and these proposed edits may actually be combined at the discretion of the owner. For example, if an owner selects two proposed edits, one which edits the color scale and one which edits the size, he may then select any of those two individually to update the master, or he may select both transformations to be combined and updated as master – resulting in a resized picture with edited color scale. The algorithm for the combination of these manipulations has yet to be determined.

Multi-User Edit
The team considered the idea of multiple users editing pictures in real time and while we disliked the idea of immediately updating another user's photo while they are editing it, we felt that the idea of two people editing a picture together had much potential. A multi-user edit embraces this idea. A user will be able to open a picture in the editor and invite a friend to also edit this picture. At this point a mini 'chat-room' will allow these two users to communicate to each other while the editing of the picture is taking place. Additional users may also join this editing process. Each user will be able to edit the currently active picture as in a regular single user edit. However a user will also have the option to 'save' and 'refresh'. During this process the same algorithms used for combining single edits will be applied for every save made by a user.

1.1.6 User Interface

The required user interface functionality will now be outlined below as well as a brief description of what each sub page under the main page will entail.

Register / Login
The first page the user should see upon accessing the site. The user will require some method for which to create an account and then login to that account. In addition functionality that supports the aid of password/account retrieval if that information has been lost should be provided.

Main Page
The main page will need to organize the options the users have in terms of site navigation and account modification. Also additional misc information will also need to be displayed through the main page. This additional misc information includes: recent picture edits, a news feed, and friend requests. The main page will provide access to the friends page, album page, and account options.

Friends Page
Through this page the user will be able to search for other users on the database and request friendship. In addition, current friends will need to be displayed. The user should be able to access friend's albums through this page by selecting a friend on this page.

Album Page
This page will list all albums created by the user in addition to various album options as outlined in the previous sections. Each album can be selected which will display all pictures currently stored in that album. A similar page is loaded when a friend is selected from the friend's page; the main difference is that a user will be viewing his friend's album instead of his. In that case, certain options will be made unavailable.

Account Options
This page will allow for the modification of personal information attached to each account. Current information will be displayed and text fields, or other methods, will be used to allow for editing of this information.

Log Out
Upon logging out the user will need to be taken directly to the login page.

1.1.7 Feature Enhancements

Rotate functionality
This pertains to an extra feature of the photo manipulation editor. Functional requirements include our implementation of a slider bar to rotate a picture by however many degrees the user desires. The pivot point in the rotation of the picture will be the center of the picture, dictated by the size of the photo.

Image Templates

This pertains to an extra feature of the photo manipulation editor. Functional requirements include our need to design a number of borders of common image sizes, e.g. 640x480 px (common web image size), 900x600 px, 1800x1200 px, 600x900 px, 1200x1800 px(common photograph sizes, both landscape and portrait). The user will be able to select a template from a drop-down list in the photo manipulation editor and, after selected, a new object in the form of the template will be placed over top the photo being edited.

Editing Status Border-use

This pertains to an extra feature of the album section. There will be a color system based on whether or not a user's photo is being edited. A red border around the thumbnail of a photo will mean that the photo is currently being edited. If the photo is not in use however, a green border will be used around the thumbnail of a photo meaning the photo is currently free of users editing it. In a user's account settings, he or she may choose to turn off this feature.

Cross-browser compatibility

This pertains to an extra feature of our system as a whole. Cross-browser compatibility can often be found an issue with the use of CSS. Extra code and method calls will be used to bypass this otherwise inefficient viewing issue. We plan to allow the user to interact on one browser (i.e Firefox) with another user on a different browser (i.e. Internet Explorer) as well. It seems only fair that we do this if we support viewing of our system at all on multiple browsers.

Newsfeed

This pertains to an extra feature of our main page. A user's main page, displayed after login, will have a list of notifications based on the actions of themselves or friends including editing one of the user's photos, tagging a user's photo or uploading a new photo/album inclusive to the user's viewing permission levels. It will also include any number of alerts the user may have received since last login including friend requests from other users.

Type tool

This pertains to an extra feature of our photo manipulation editor. This functionality will allow the user to write text on his or her image. Inclusive will be a number of fonts we support which the user will have to choose from a drop-down list along with other drop-down lists for color and font size. Separate buttons for boldness, italics, underline, and cross-out options will be available for the user's text as well. A textbox will appear on the canvas and after typed out, the user will have the option to drag his or her text within the confines of the canvas.

1.2 Function Ratings

The below chart explains various functions as well as their priority level for implementation.

Functional Requirements	High Priority	Medium Priority	Low Priority
Undo	X		
Redo	X		
Crop	X		
Rotate		X	
Collage	X		
Search functions	X		
Browse images	X		
Upload images	X		
Download images	X		
Delete images	X		
Save images	X		
Resize	X		
Brightness	X		
Contrast	X		
Grayscale	X		
Sharpness	X		
Image templates			X
Editing status Border-use			X
Cross-browser compatibility		X	
Newsfeed			X
Type tool			X

2. Project Plan

2.1 Software Development Life Cycle

Our team has decided to adopt a hybrid approach of the Spiral and Scrum Software Development Life Cycle (SDLC).

The basic phases of our SDLC still include the initiation of the project, planning for the project, implementation of the product, and maintenance of the product. Unlike traditional methodologies the main agile methodology is generally more adaptive and people-oriented rather than being predictive and process-oriented. For project planning, the agile method encourages stakeholders to get involved and team members to work together. An agile team is characterized by teamwork and collaboration, sharing of information and being adaptive to change.

The scrum methodology is an agile approach to software development but instead of providing complete and detailed descriptions of how everything must be done on the project, it is left to the team to dictate the criteria by which they should perform. Scrum relies on a self-organizing and cross-functional team and as such, there is no overall team leader who decides how and by whom a problem will be solved. These issues will be decided by the team as a whole. Scrum teams are supported by two individuals called a ScrumMaster and a product owner. The product owner represents stakeholders or users that will help the team by giving feedback about the product being implemented.

Scrum methodology is an iterative and incremental process, so the project is organized into a series of sprints which usually lasts anywhere from a week to a month. At the beginning of each sprint, the product owner informs the team of the functional requirements that they want and the team determines how much can be done during the sprint. During these sprints, the team conducts something called a daily scrum where all team members, ScrumMaster, and product owners are in attendance. Essentially the team gets together to answer three questions: what they have worked on yesterday; what they plan to do today; and what problems might hinder their progress. During the sprint, requirements are not allowed to be changed until the next sprint. At the end of each sprint, the functionality and features decided at the beginning of the sprint are implemented. A review of the work done is conducted where the team invites stakeholders to attend who provide feedback and may influence the next sprint.

Our team is using a hybrid approach and therefore we do not follow just one SDLC rigidly. Although the scrum methodology is less structured than the traditional project planning methodologies, the basic phases of project planning still hold. In the scrum methodology there are no overall roles and responsibilities for each member of the team. Our team does assign certain roles and responsibilities for each member of the team (refer to documentation on personnel roles and team organization) however, the more important decisions such as the delegation of tasks and how a problem will be solved are decide by the team as a whole.

During the implementation of functional requirements, our team has decided to use the daily scrum concept where the sprint is the project phases (phase A, B and C). But instead of having requirements frozen during each sprint, they are adaptive since most of the implementation design is left up to the team and the documentations are always being updated. The daily scrum is conducted through e-mail instead of 15 minute daily meetings because the team does not necessarily see each other every day. In this way, every member of the team is informed about the progress we are making.

In regards to product design, although the agile methodology does not include the design of UML diagrams, our team will be using UML diagrams however as a way of implementing this methodology, the UML diagrams will be open to changes throughout the project phases to adapt to the software product.

2.2 Personnel Roles and Team Organization

2.2.1 Project Team Structure

Roles	Assigned	Areas of Responsibility
Project Manager	Teresa Shih	Delegate tasks; prepare team meetings and oversee the project. Schedule and plan throughout the product life cycle to ensure the success of the project. Communicate with client.
Product Management	Jeff Flores Joe Bettridge	In collaboration with the PM, select desired system functionality, and overall design. Also, consider possible platforms/languages/frameworks for superior implementation.
Lead Programmer	Joe Bettridge	Create and execute programming work plans and revise as appropriate to meet changing needs and requirements. Responsible for the efficient management of the programming team as well as distribution of required tools to assist in creation of required software.
Programmer (Web/Editor)	David Pereira Jeff Flores Tahir Butt Teresa Shih	Develop the proposed system using the design specification provided by the lead programmer, while maintaining proper coding standards. Communicate with the team quality assurance director to assure upmost software value.
Database Programmer	Tahir Butt Teresa Shih	Create an efficient and logical database for the storage of user account information and user photographs. Also responsible for communicating database layout to technical publicists for accurate documentation.
Graphical Artist	Jeff Flores	Plan and create visual solutions. Apply specialized techniques through various forms of media, including website and logo design.
Technical Publications	David Pereira Jeff Flores Joe Bettridge Tahir Butt Teresa Shih	Prepare technical documentation of the design specifications, user manuals, and other system related documents. Proofread, edit and format the documents before finalizing and including them in the final draft.
Software Quality Assurance	David Pereira	Develop a thorough testing strategy/plan, including both unit and integration testing. In addition, provide descriptions on how various tests are carried out, and on their expected results.
Minutes Taker	Teresa Shih	Responsible for the accurate recording of all relevant discussion and information exchanged during team meetings.

2.3 Project Milestone

Project Milestone	Project Artifact	Due Date
Project start		01/18/2010
Prototype showcase	Prototype	02/02/2010
Phase A	Phase A documentations	02/05/2010
Status report		02/09/2010
Phase B	Phase B documentations	03/05/2010
Project swap	Software product	03/12/2010
Status report		March 2010
Phase C	Phase C delivery	04/02/2010
Status report		April 2010

* due dates that are unknown at this time are written as (month year) format

2.3.1 Work Breakdown Structure

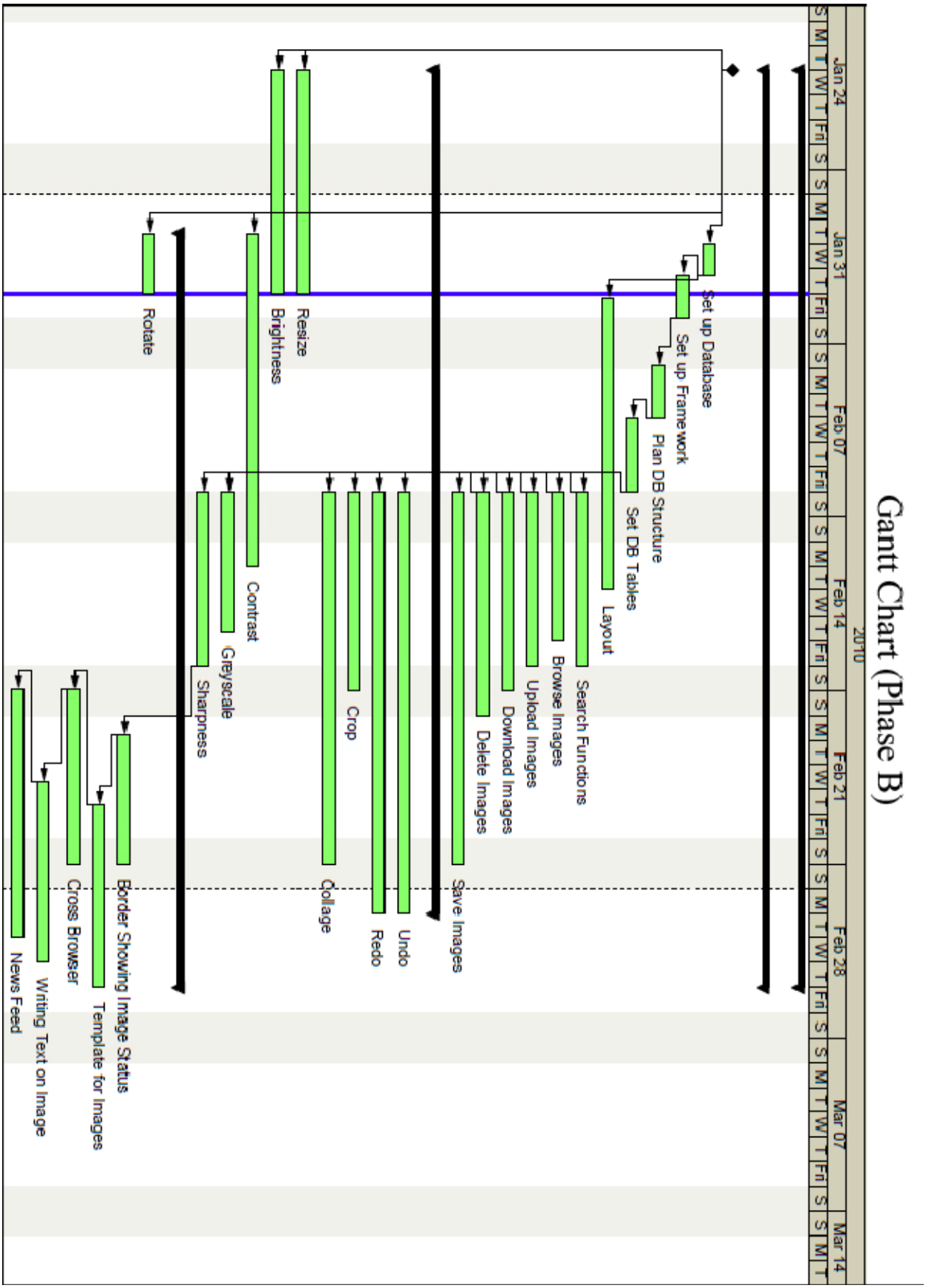
ID	Task name	Start date	Finish date
1	Phase B	01/27/2010	03/05/2010
2	Set up database	02/03/2010	02/04/2010
3	Set up framework	02/04/2010	02/06/2010
4	Plan DB structure	02/08/2010	02/10/2010
5	Layout	02/05/2010	02/17/2010
6	Set DB tables	02/10/2010	02/13/2010
7	Search functions	02/13/2010	02/20/2010
8	Browse images	02/13/2010	02/19/2010
9	Upload	02/13/2010	02/20/2010
10	Download images	02/13/2010	02/21/2010
11	Delete images	02/13/2010	02/22/2010
12	Save images	02/13/2010	02/28/2010
13	Undo	02/13/2010	03/02/2010
14	Redo	02/13/2010	03/02/2010
15	Crop	02/13/2010	02/21/2010
16	Rotate	02/02/2010	02/17/2010
17	Collage	02/13/2010	02/28/2010
18	Resize	01/27/2010	02/10/2010
19	Brightness	01/27/2010	02/10/2010
20	Contrast	02/02/2010	02/16/2010
21	Grayscale	02/13/2010	02/18/2010
22	Sharpness	02/13/2010	02/20/2010
23	Border showing image status	02/22/2010	02/28/2010
24	Template for images	02/25/2010	03/05/2010
25	Cross browser	02/20/2010	02/28/2010
26	Writing text on images	02/24/2010	03/04/2010
27	Newsfeed	02/20/2010	03/03/2010

*refer to Gantt chart (below)

2.3.2 Gantt Chart

Project Schedule (Phase B)

Activity ID	Activity Name	Start	Finish	Comments
Project Schedule		27-Jan-10 A	05-Mar-10	
Phase B		27-Jan-10 A	05-Mar-10	
A1000	Start of Phase B	27-Jan-10 A		
A1010	Set up Database	03-Feb-10 A	04-Feb-10 A	
A1015	Set up Framework	04-Feb-10 A	06-Feb-10	
A1020	Plan DB Structure	07-Feb-10	10-Feb-10	
A1030	Set DB Tables	10-Feb-10	13-Feb-10	
A1040	Layout	05-Feb-10	17-Feb-10	
A1050	Search Functions	13-Feb-10	20-Feb-10	
A1060	Browse Images	13-Feb-10	19-Feb-10	
A1070	Upload Images	13-Feb-10	20-Feb-10	
A1080	Download Images	13-Feb-10	21-Feb-10	
A1090	Delete Images	13-Feb-10	22-Feb-10	
A1100	Save Images	13-Feb-10	28-Feb-10	
Photo Editor Functions		27-Jan-10 A	02-Mar-10	
A1110	Undo	13-Feb-10	02-Mar-10	
A1120	Redo	13-Feb-10	02-Mar-10	
A1130	Crop	13-Feb-10	21-Feb-10	
A1150	Collage	13-Feb-10	28-Feb-10	
A1160	Resize	27-Jan-10 A	05-Feb-10 A	
A1170	Brightness	27-Jan-10 A	05-Feb-10 A	
A1180	Contrast	02-Feb-10 A	16-Feb-10	
A1190	Greyscale	13-Feb-10	18-Feb-10	
A1200	Sharpness	13-Feb-10	20-Feb-10	
Enhancements		02-Feb-10 A	05-Mar-10	
A1140	Rotate	02-Feb-10 A	05-Feb-10 A	
A1210	Border Showing Image Status	22-Feb-10	28-Feb-10	
A1220	Template for Images	25-Feb-10	05-Mar-10	
A1230	Cross Browser	20-Feb-10	28-Feb-10	
A1240	Writting Text on Image	24-Feb-10	04-Mar-10	
A1250	News Feed	20-Feb-10	03-Mar-10	


Gantt Chart (Phase B)
2010
Jan 24
Jan 31
Feb 07
Feb 14
Feb 21
Feb 28
Mar 07
Mar 14
Set up Database
Set up Framework
Plan DB Structure
Set DB Tables
Layout
Search Functions
Browse Images
Upload Images
Download Images
Delete Images
Save Images
Undo
Redo
Collage
Crop
Contrast
Greyscale
Sharpness
Rotate
Brightness
Resize
Border Showing Image Status
Template for Images
Cross Browser
Writing Text on Image
News Feed

2.4 Risk Analysis

There are a number of problems which may arise through the lifespan of the project; they will be described briefly followed by prevention methods in the next subsection. We will cover three of the more grave risks, as determined by the team. The risk table provided below also contains additional risks that were considered.

Programming Language
The team decided on the use of ActionScript3.03.0 for the photo editor. This brings with it the visual advantage of using Adobe Flash Player to run the program. The main risk involved with this decision is that of familiarity. Four out of five programmers on the team have no prior experiance with this language. The selection of a language that our team may be more accustomed to (for example, Python) could result in less time spent programming and with a tight time budget this could translate into an excellent advantage. This is by far the most serious of all the risks and will need to be monitored closely during the early stages of the project.

Database Essentials
A great deal of information will need to be stored on the database, including user accounts, graphical information, friends' lists and permissions. The proper design and layout of this database is essential for retrieval of valuable information. Closely related is the section following.

Multiple User Access to a Single Picture
The most serious threat in terms of the editor tool, will be allowing multiple users to edit a single picture. The client suggested that we consider what changes were made to the picture by each respective user and attempt to combine those changes. After careful consideration the team decided that through the use of a feature provided by ActionScript3.03.0 we will be able to immediately refresh each user picture as another user edits the same file. It remains to be seen if this will present the best solution to the problem, or what this solution entails in terms of database resources.

Control and Prevention
This section addresses methods to monitor the above risks, as well as strategies to follow in the event that risks become actual problems.

Programming Language
Once a programming language has been selected and the project is well underway there is simply no turning back in terms of selecting a different language. To counteract this measure the team has decided to begin programming as early as possible, attempting to accomplish as many of the basic features as possible to thoroughly convince all team members that this language is capable of meeting our project needs. Should ActionScript3.03.0 fail to incorporate any of the basic needs efficiently, then the team will select a different programming language early, rather than later.

Database and Multiple Accesses

When the time comes to implement this key functionality, if the ActionScript3.03.0 feature does not meet our initial expectations then we will need to fall back onto our client's suggestion and reconsider how to handle multiple accesses to a single file object. Though this plan might seem to imply that much time will be wasted, this is not the case. Our programmers will be able to immediately tell if the ActionScript3.03.0 feature fails, and the PM is already taking this additional time into consideration.

Risk Table

Risk Description	Category	Probability	Impact	Risk Exposure
Database Layout	Development	85%	8	critical
User Multi Access	Development	65%	8	critical
Language Difficulties	Development	70%	7	critical
Not enough time	Project size	45%	10	catastrophic
Other course conflicts (ex assignments)	Development	90%	4	marginal

3 Test Strategy and Test Plan

3.1 Test Strategy

Introduction

The following is a description of the testing strategy and plans which will be put into use throughout the project lifecycle. This includes explanations of the tests for our photo manipulation applet, our webpage, which contains the embedded applet and initiates queries to the database, as well as the backend database itself. The main testing and tracking tools to be used will be Adobe Flash Player and Trac, a web-based bug tracking software. This test schedule will depend heavily on the provided project plan and, as such, testing will be performed as the project plan dictates.

Approach (Strategy)

As specified above, the main tools we will be using will are Adobe Flash CS3 and Trac, a web-based bug tracking software. Trac requires little training, and ActionScript3.0 knowledge is required for proper use of Adobe Flash CS3 tool in a programming sense.

The SQA director will gather knowledge on the functional requirements of the system as well as the lead programmer's strategy for designing the system. At this point, tests will be developed to assure that the requirements are being met as specified by the project plan. Furthermore, additional tests will be created to attack potential weak points in the system/design and to reveal flaws to be documented and corrected.

There are two main sections of this project which will require heavy testing. The first involves the photo manipulation applet. Much of the actual code will be incorporated in communications between flash and a backend database. Due to the extensiveness of this portion of the code, these functions will need to be tested with a great deal of care. Test function cases will be programmed using ActionScript3.0 and these will feed various inputs into the editor functions, afterwards outputs will be measured to determine if the test passed or failed. In addition to these unit tests, the entire system will need to be retested once each individual component is assembled into the full editor.

The second section will involve the web-page and database which is essentially the information acquisition/display section of the project. Various pieces of information will need to be stored and retrieved from the database, either through the use of scripts or manual input, and then checked to ensure expected results. The web-page will need to be loaded on a selection of browsers to ensure compatibility. In addition, use of the applet, together with picture storage, will be looked into to ensure collaboration of both sections of the project.

Responsibilities

The main responsibilities for testing and planning will be taken on by the SQA director. The SQA director will ensure that the picture editing tool, web-page, and database are working according to the

specifications. It will also be the responsibility of each programmer to properly document their code to ensure that a general explanation at the function level is provided along with an overview as well.

3.2 Test Plan

Test Items

The project consists of three major components: UI, database, and picture editor. This test plan hopes to encompass these three elements to ensure a quality product is delivered to the client.

User Interface

The user interface consists of the web page from which users will access their accounts and pictures. Basic social networking features such as accounts, friends lists, and album creation will be accessed directly using this avenue. Each feature will need to be tested individually, and the questions of cross browser compatibility will be addressed.

Database

The database will house all account and account related information. The integrity of the database will need to be analyzed before it is integrated into the entire project system as a whole, then will need to be rechecked to ensure proper cohesion with other project elements (UI and editor)

Picture Editor

This is the major feature of the entire social networking site and therefore will undergo the most testing. The picture editors main functions will be tested independently of other picture editor functions, then reconsidered once all features are combined. Once the editor is embedded into the web page interface additional tests will need to be passed to ensure data is being stored and retrieved appropriately.

Features to be Tested

Below is a list of items to be tested from the user's point of view, followed by an indicator concerning the level of risk each feature entails. A high level of risk, for example, is considered to mean that this feature is at a high risk of failure when tested. (Additional features may be added included at a later date).

Features	Risk
Account Management (Creation, Editing, Deletion)	Low
Miscellaneous Friend Features (Adding, Editing, Searching, Viewing)	Medium
Miscellaneous Album Features (Creation, Permissions, Viewing, Modifying)	Medium
Miscellaneous Picture Features (Uploading, Naming, Tagging)	Medium
Embedded Picture Editing Tool	High

Test Deliverables

Each major element as outlined in test items will include it's own sub-plan which will outline exactly what functionality will be tested. To this end both formal and informal test cases will be presented and documented. It is the goal of the SQA director to organize the various tests into categories and present adequate test cases to ensure product quality.

Test Cases

Though the actual composition of a general test case has not yet been decided it is assumed that each test case will present the following information:

- Test ID (if necessary).
- Test category
- Test case description.
- Test steps (order of execution).
- Input (/Groups of inputs for generalized cases).
- Expected output.
- Output.
- Conclusion (Pass/Fail).
- Remarks, Related Documentation, Trac Link.

4. Software Architecture and High-level Design

4.1 System's Interaction with its environment and external systems

4.1.1 External Interaction

Currently we are independent of any external systems.

4.1.2 Environmental Interaction

In the case of interaction with browsers, our service aims to be compatible with the browsers we think to be of use in the mainstream public, including Mozilla Firefox 3.6, Internet Explorer 8 and Opera 10.10. We've considered supporting mobile browsers, but currently we are focused more on internet capabilities. On the upside, the newer phones, running the Android mobile operating system, have Adobe Flash support. As our photo manipulation editor is essentially a Flash applet, we could have just the photo manipulation editor be available for those phones but of course the social network side of it all would be omitted.

Catering to the notion that we aim to have a marketable product by the end of this project, our service will properly operate on current Microsoft and Mac operating systems, the two leading O/Ss in the market right now. The fact that we are using only open source languages and products should allow us to say we would have no problem in the utilization of any other O/S as well.

As for our choice of web server, the Apache HTTP Server is the most popular web server in current use and because we are familiar in its use, the choice was clear. Apache Version 4.4.4 is the most recent release of the software available and we plan to use it in tandem with PHP scripts.

4.2 High Level System Architecture

System Architecture
At the high level, the system will follow the standard model-view-controller architecture in the manner described below. In addition, the technology/language selected for each implementation is provided.

Model - (Database - MySQL)
Our database will be the backend of the system whose sole responsibility is that of data storage and retrieval. MySQL will provide us the required database management system, and will run as a server, allowing multi-user access.

View - (Webpage - HTML, PHP, CSS)
In essence, the user interface of our project. The webpage will be responsible for laying out information in a visually pleasing manner, and providing information to the controller and model as required. A subset of the provided technologies will be used.

Controller - (Editor/Webpage – ActionScrip3.0, Ajax)
The controller works as a liaison between view and model aspects of the system. It takes required inputs, performing whatever manipulations are necessary and stores that information as required. The team will use Ajax calls to provide the functionality required from the webpage and ActionScript3.0 to power the functionality behind the photo manipulation applet.

4.3 System Decomposition

The following is a description of the project system broken down into three main components. Afterwards a brief look into the interactions between these components is outlined.

Components
There are three major components to the project system as may have been alluded to in the previous section. These are: The webpage, the database server, and the picture editor. Each can be focused on independently in development and will need to be integrated for the final product to be delivered.

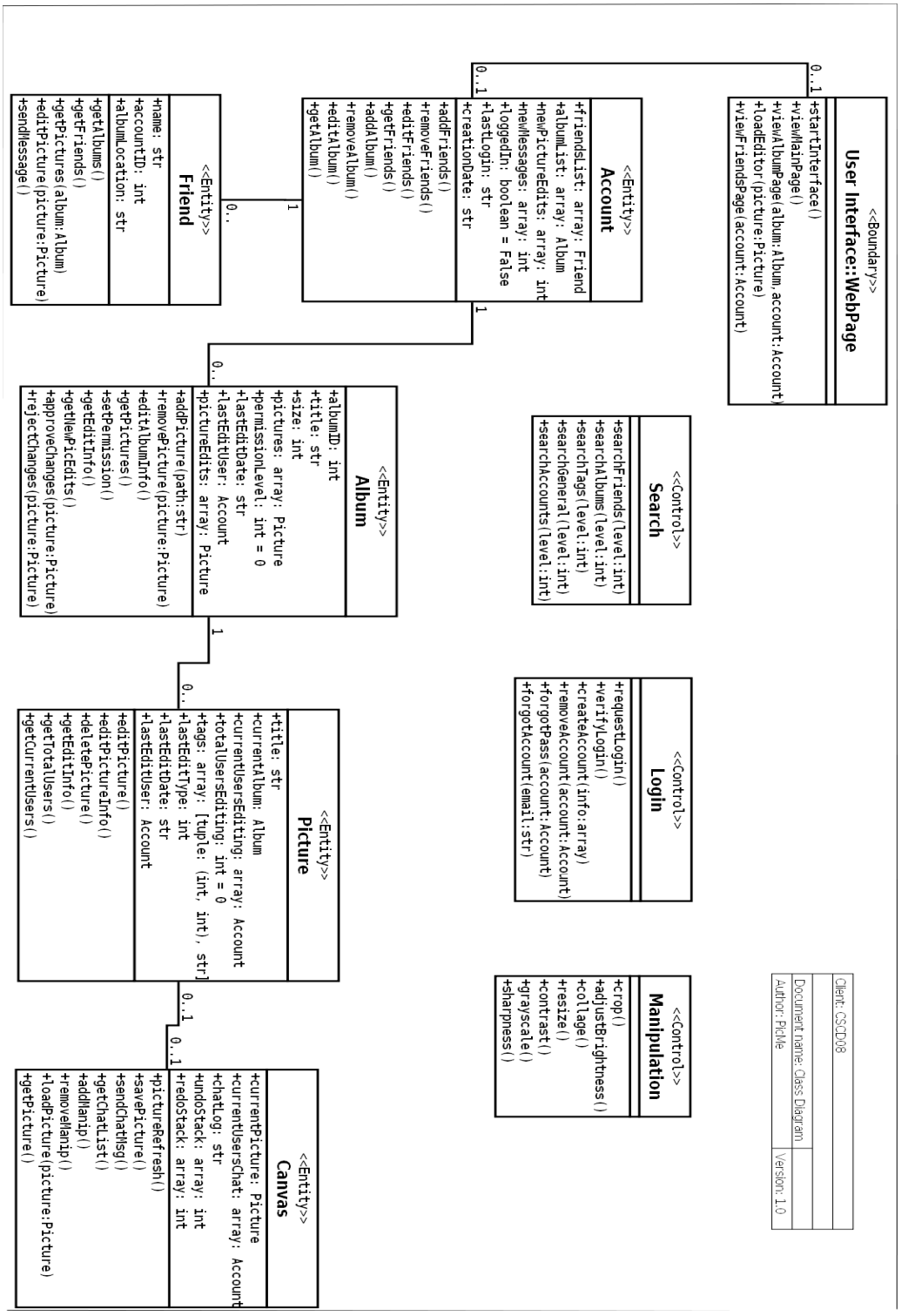
Webpage
This section pertains to the user interface for the social networking site. It will display all account related information including (but not limited to) friends, albums, pictures, site logo, miscellaneous account information, news feeds, updates on account activity (friends editing users photos), etc. The webpage will be the user's portal to the photo manipulation applet, where each user will be able to manipulate their pictures and possibly those of their friends (with proper permissions set). All the information that is stored in the database will at one point or another have gone through the main web page.

Database
The database is simply a storehouse for all required information for the site to operate, this is anticipated to include the following information: Miscellaneous user information (first name, last name, nick name, age, etc), albums, pictures, photo editing information (undo, redo), friends lists, and possibly other information which has yet to be discovered useful.

Picture Editor
The photo editor takes a digital photograph (including jpg, jpeg, png, bmp, and gif) and performs some basic manipulations to it. These manipulations include: cropping, resizing, creating a collage, adjusting brightness, contrast, grayscale, sharpness, etc. As well, the editor will be able to use helper functions including undo, and redo.

4.4 System Design (Class Diagram)

4.4.1 Class Diagram



4.4.2 Class Diagram Description

The following is an explanation of the UML class diagram presented in the previous section. A brief look into each individual class will be presented along with a few comments on their interactions.

Note: To avoid ambiguity, when speaking about a class as it appears on the diagram, the first letter will be capitalized. When speaking about the concept of a class, letters will be in lower case. (ie 'Account' class, as opposed to a user 'accounts')

User Interface::WebPage
The User Interface::WebPage class works as a concept and will not be implemented as shown in the class diagram. The class involves functionality provided by the browser. This class is responsible mainly for the sending and fetching of information from the database for display on the screen.

Account
The Account class maintains information regarding the users account. This information includes friends list, album lists, recent edits made to pictures on that account, recent logins, and other miscellaneous account information. It is through this class that manipulation of albums and friends lists will take place. This means that an Album cannot delete itself, it relies on the Account to provide that service.

Friend / Album
The Friend and Album class are two major components of the Account class. An Account class will contain a list of both of these classes. Since the main theme of the site is that of editing pictures there needs to be a way to maintain an organized collection of what couple possibly be many hundreds of pictures. In addition, editing of friends pictures will be a large feature of the site and so a manner in which to maintain a proper collection of those friends is paramount.

Picture / Canvas
The Picture class stores information regarding a specific picture, every picture will be wrapped into a Picture class. Through this class you can edit general picture information including names and tags as well as view recent edit information to that picture.

The Canvas class provides a means for the graphical manipulation of a Picture. A Picture must be placed onto a Canvas and only then can it be edited. The Canvas itself does not manipulate the picture; it however can perform some required functionality such as saving and providing a chat interface for two users editing the same picture.

Search / Login / Manipulation
Search, Login and Manipulation are the three control classes for the current system. The Search control class is mainly responsible for working with the user interface to return information from the database regarding particular queries.

The Login control class handles all functionality required on the login page with regards to account creation, password retrieval, account deletion, verification, etc.

Lastly, the Manipulation class performs transformations to whatever picture is currently loaded on the canvas. The current base requirements for functionality are demonstrated on the class diagram but additional features will be added at a later date.

Overall Interaction
From the information presented, an intuitive feel of how each class interacts with one another can be developed. The User Interface class loads information pertaining to the active Account which is logged in. That Account was able to log in through the use of the Login control class operations.

Once an Account is logged in, it may make changes to its current friends list which contains an array of Friend classes (empty on creation of account). Through the User Interface class, and with the help of the Search control class, a user will be able to search for friends to wrap into a Friend class and add to their friend's list attribute, which is, in turn, tied to their Account class. Various other searches may be used through the User Interface and Search classes.

In addition to the manipulation of the friends list, a user may also create an Album class which is added to the Account class's album list. Each Album can contain any number of Picture classes which describe information regarding digital pictures uploaded on the website.

A Picture class can be 'placed' onto a Canvas class. Once this occurs, the Manipulation control is allowed to perform the desired modifications to that picture. The Canvas class may then store that information back to the web server.

4.4.3 System Design (Component Interactions / Class Allocation)

Below is a description of how the components in this system interact and where each class described in the previous section may be found in relation to the overall system.

Component Interactions
As a reminder, the three main components of this system are the webpage, photo manipulation applet and the database. These can be reviewed in section 4.3.

The only interactions taking place by these components are between the web page and the database, and the web page and the photo manipulation applet. In other words, the database and the photo manipulation applet do not directly communicate.
The webpage will interact with both the database and picture manipulation applet directly. It will perform queries on the database to retrieve relevant information as indicated by the user, and it will

also store desired information. The main page will load pictures into the photo editor and then be responsible for saving them to the database.

Class Allocation

Each of the nine classes presented in the class diagram may be associated with a certain component of the system. That class is then considered to be implemented by that specific component. The details about these implementations are provided below.

Webpage

Nearly all classes will be manipulated or created at the webpage level. This includes: the User Interface boundary class, the Account class, the Friend class, the Album class and the Picture entity class. Also, the Search and Login control classes will be manipulated at this level.

Photo Manipulation Applet

The photo manipulation applet will consist of two classes, the Canvas and the Manipulation control class, which is used for the modification of pictures.

Database

The database is only used to store information and while it will interact with nearly all classes it is not, in fact, directly responsible for the creation of these classes. For more information on the database, please refer to Section 5 (below).

5. Information Representation

5.1 Data Representation

In our database design we chose not to store passwords in plain text, for a number of good reasons. The utmost important reason is clearly that any system administrator would have immediate access to all stored user passwords, an immediate danger to users who use the same passwords for multiple services, as many people do. A secondary issue with storing passwords in plain text is if the database were to be compromised (by an external attack on the site or theft of a backup copy of the database). The attacker would have access to all accounts in the database and, in our case, personal information and personal photographs could be stolen (including those of the administrators).

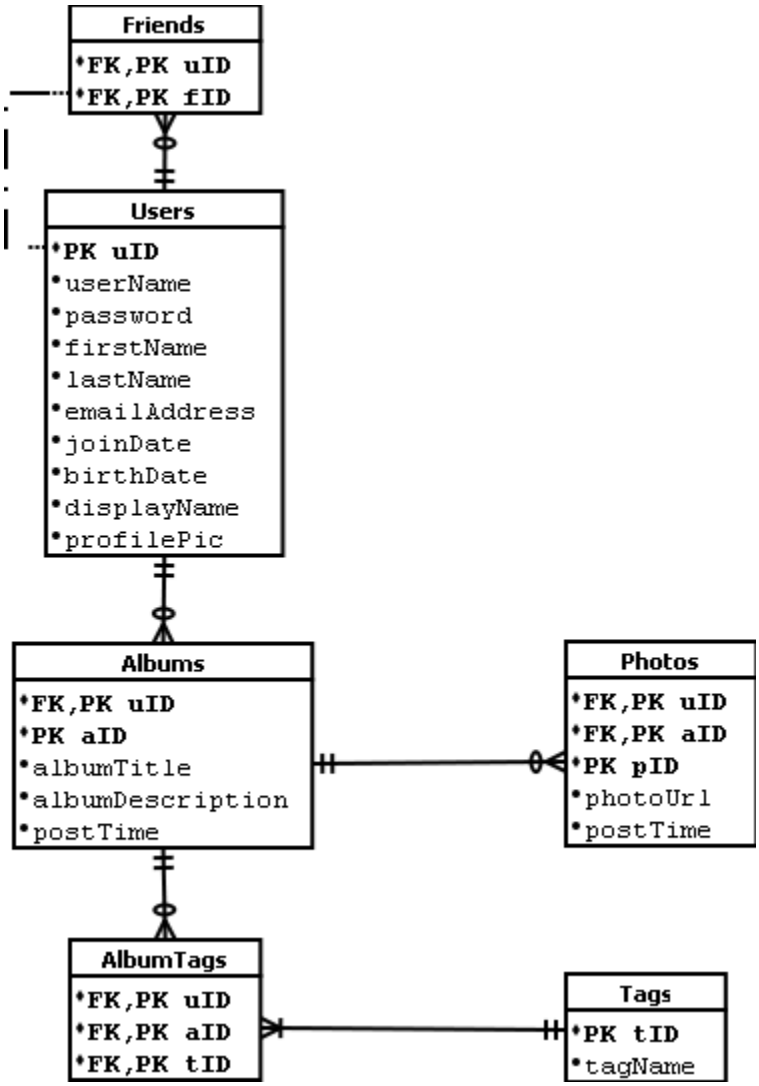
In order to alleviate these problems, we have chosen to hash all passwords being stored in the database. Hashing is a very low cost technique to store passwords. By using an algorithm such as SHA-1, the computation time is very small, and the security that is added is very great. We elected to use SHA-1 over lower-collision algorithms such as SHA-512 because the cost of this algorithm is much lower, and the ability to create reverse collisions is very challenging. If users have 'strong' passwords, it is next to impossible to breach this algorithm. There is one major drawback to hashing passwords however. If a user forgets their password, there is no way of giving them their password back. A workaround for this is to give out a temporary password if the user requests a new password.

As for the types and lengths of the databases' columns, we selected sizes for INT columns that could sufficiently store the required data without too much leftover space. For string-based data, we elected to use VARCHAR over CHAR. This choice was made because VARCHAR will compensate for the size of data being inputted therefore not wasting space with empty filler characters.

In our database model we have chosen to use table normalization over delimited lists due to the fact that normalization increases size efficiency greatly, case and point made being the storing of friends for a specific user. In comparison to what we plan to implement, by storing an additional column containing a delimited list of friends' uIDs in the *Accounts* table, any time a search would be required, this entire list would have to be parsed. Instead of using this obvious low grade way of database structure, we chose to create separate tables, better known as normalization, to store our information. In the same case of storing friends for a specific user, a table, *Friends,* will be created. In this table, we plan to have two foreign keys (uID, fId), where any combination of these keys will be unique (a primary key). The key uID refers to the user's uID from *Accounts* and fID refers to the uIDs of the user's friends. As there are no major drawbacks to using this form of normalization, we have decided to use it anytime we feel it is required within the database.

The Database Management System (DBMS) that will be used is MySQL as it is among the most widely used DBMSs as well as the most popular open source DBMS because of its ease of use, fast performance and reliability.

5.2 Data Model (ER Diagram)



5.3 Database Schema

DatabaseSchema

Users

uID	userName	passwrd	firstName	lastName	emailAddress	joinDate	birthDate	displayName	profilePic
INT(11)	VARCHAR(32)	INT(150)	VARCHAR(64)	VARCHAR(64)	VARCHAR(255)	DATE	DATE	VARCHAR(32)	VARCHAR(255)

Friends

uID	fID
INT(11)	INT(11)

Albums

uID	aID	albumTitle	albumDescription	postTime
INT(11)	INT(9)	VARCHAR(128)	VARCHAR(255)	TIMESTAMP

Photos

uID	aID	pID	photoURL	postTime
INT(11)	INT(9)	INT(5)	VARCHAR(255)	TIMESTAMP

AlbumTags

uID	aID	tID
INT(11)	INT(9)	INT(11)

Tags

tID	tagName
INT(11)	VARCHAR(32)