Final Implementation

CS4313 Software Engineering Project

Group members:

Computer Systems:

Shane Whelan 09005763

Matthew Grant 10118411

Graduate Diploma in Computing:

Thomas Ryan 11147032

James Hickey 11086068

Description of all Functionality

Functionality that covers the Specification:

The dating website allows users to register an account. On the registration form each field is checked to make sure it's not empty. There is an about you field for users to write a paragraph about themselves. Users are able to search for each other by Username, Age and Interest. Searching by Age is done by age range as the users actual age is kept private. When a search is successful a list of users and their profile pictures are returned. The show all users page shows a list of all users and their profile pictures.

In the My Account section the website administrator can delete users from the site and the database. On the matches page users are automatically matched based on whether they are Straight, Gay or Bisexual, after those results are filtered out the users are then matched by their answers to interest questions. This means the user is only matched to someone they can actually go on a date with.

On the registration page each field is limited to the number of characters the database has set for that field. The date is checked using JavaScript, this is important because the Insert statement will fail otherwise. The user's passwords are checked to make sure they match. The username is checked to make sure it's not already in the database.

Extra Functionality:

Each user has a profile page which they are brought to after login. This profile page displays all your info that other users can see and your profile picture. There is a messaging system on the website so all users can message each other. The user is able to compose a message, go to their Inbox and their Sent messages. The recipient is checked so the user can't send a message to themselves.

There are tables in the database where a list of questions for users to fill out is stored. These questions are dynamically loaded in so the number of questions doesn't matter. These questions are split into two categories: Attributes and Interests. Attributes are human attributes like eye colour or hair colour and Interests are things like Sports and hobbies. This design was chosen so the website owner can easily add and remove these questions as well as the answers to these questions and the PHP scripts and HTML will load them in dynamically. In My Account questions are loaded in and then the answers are loaded in below them in a HTML selection box. Once the users have submitted an answer the question and answer they choose is stored in the database. What's so important about this is that the "Matches" system hinges on these questions and matches users based on similar answers. When a user finds a match a list of users, their profile pictures and the questions they were matched on is displayed.

Any time another users profile picture and name is displayed it is a link to their profile. A GET variable is sent to a Dynamic profile page if the user clicks on a link or a profile picture of another user. On this page the other user's information and profile picture is displayed and a link is provided to message the user. When a user decides to message another user they are brought to the compose message page.

In My Account the user is given many options. A profile picture can be uploaded which is uploaded as a blob to the database. The user can also update their profile picture to a new one. The user is given an option to delete their account if they wish. Their credit card information is asked for here and it's displayed if the user's subscription is paid or not, this information is stored in a database table. The user can update their Attribute and Interest questions too. Their personal information can be changed too except for things like date of birth which doesn't change in the real world. The database is queried and if the user is an Administrator two fields are displayed, one field to ban a user and another field to promote a user to moderator.

A Cookie is set when the user logs in to grant them access for 1 hour, when they logout it is deleted. If the user isn't logged in and they try to access a page like My Account they are redirected to the login page. The footer for each page is a copyright with a year. The year dynamically updates if the year changes.

All passwords entered by the user are MD5 encrypted and stored in the database in their encrypted form. All SQL statements are protected from SQL injection. SQL injection protection is important so that no malicious SQL can be entered into a query. All queries where the user has the chance to input text are protected.

NOTE: As a result of the way the server on campus is set up profile pictures do not upload to the database on testweb2. Screenshots of this functionality on our own server are provided to prove it works. We were told to mention in the report that we received the information about the server not accepting file uploads two days before the deadline and that we didn't have time to implement the complicated workaround.

Megafaun

Basic Details:

Firstname: jsajk
Gender: Male
Orientation: Straight
About Me: dfklmlkjsdflkdfkjnl
County: Carlow
Occupation: Science

Attriutes:

What's your eye colour?
Green
What colour is your hair?

Final Database Tables

authentication:

username	password
varchar(10)	varchar(255)

For the user to log in they must confirm their identity with their “username” and “password”. These attributes are stored in the table “authentication”. The log in page is the landing page for the website and these are the first details that they are asked for. When they are entered they are compared to entries in the database and if they match up the user is allowed to proceed. The passwords are MD5 encrypted which is a one way encryption algorithm. This design means the database admin can’t access user passwords

userprivate:

username	admin	streetaddress	email	dob	lastname
varchar(10)	varchar(1)	varchar(30)	varchar(255)	date	varchar(16)

Different details about the user are stored in the database. Some of this information is to help match the user with other members on the website and also to help create a profile of the user. However some details are kept hidden from the public to maintain the privacy of the user. This table “userprivate” contains such details and are only visible to the administrators and the user themselves. The exact date of birth is private but you will be matched to other users based on if your date of birth fits into pre-defined age ranges. Street address and admin access are the only fields that can be changed from the website all other fields do not change in the real world.

userpublic:

username	firstname	county	occupation	about	gender	orientation
varchar(10)	varchar(16)	varchar(32)	varchar(16)	varchar(255)	varchar(1)	varchar(1)

The table “userpublic” contains details of the user that are available for any other members to view. They help other users to gain an understanding of the member that they are viewing. All fields are displayed on the user’s profile. Gender and orientation are used so that the user only can see users they would actually go on a date with. About was originally supposed to be stored as a blob but the college server is not set up correctly and doesn’t accept files from the user.

attributequestions:

attributeid	**aquestion**
int	varchar(255)

The attribute questions table stores a list of questions defined by the database administrator that the user can answer based on their human attributes (Hair colour, eye colour). Questions can be added, edited and deleted by the database administrator. These questions are dynamically loaded in so that the addition of questions will be seamless.

attributeresponses:

attributeid	**aresponseid**	**aresponse**
int	Int	varchar(16)

The attribute responses table keeps a list of the answers to the attribute questions that will be provided to the user in a HTML select box. Once again these answers can be added, edited and deleted by the database administrator.

Interestquestions:

interestid	**iquestion**
int	varchar(255)

The interest questions table stores a list of questions defined by the database administrator that the user can answer based on their hobbies and interests (Sports, movies, etc.). Questions can be added, edited and deleted by the database administrator. These questions are dynamically loaded in so that the addition of questions will be seamless.

interestresponses:

interestid	**iresponseid**	**iresponse**
int	int	varchar(16)

The interest responses table keeps a list of the answers to the interest questions that will be provided to the user in a HTML select box. These answers can be added, edited and deleted by the database administrator.

userattributeanswers:

username	attributeid	aresponseid
varchar(10)	int	int

The userattributeanswers table keeps a record of the answers picked by the user and the questions they answered. With a join operation it can be easily figured out who answered what and what users to match to each other.

userinterestanswers:

username	interstid	iresponseid
varchar(10)	int	int

The userinterestanswers table keeps a record of what answers the user picked for each interest question. The design is identical to the userattributeanswers table as with a join operation it can be easily figured out who answered what and what users to match to each other.

creditcardinfo:

username	activeuntil
varchar(20)	date

The creditcardinfo table keeps a record of all users who have successfully paid a subscription and how long they have paid for. This can be used to control access to the website for members only.

usermessages:

messageid	messagetext	usernamefrom	usernameto
int	varchar(255)	varchar(10)	varchar(10)

This table keeps a record of users who have messaged each other. Yet again the college server was not set up correctly so that a blob couldn't be uploaded for the message text. It's probably better to keep messages limited to a number of characters anyway – It worked for twitter. Why this design was chosen was you can retrieve both inbox and sent messages from this table. There is probably plenty of debate to be made for differing designs, we chose this one.

userimages: This table will keep a record of the users photos they have uploaded

username	imagename	imagetype	imagesize	imagedata
varchar(10)	varchar(255)	varchar(255)	int(8)	longblob

This table had to be designed because the college server doesn't allow the PHP temporary area function to be used. This means that user uploads are disabled. Originally the design was to put the users picture in a directory and store the link to that picture in table. In an attempt to get it working this design involving uploading using blobs was derived. This also failed because of the

aforementioned server issue. The design works on our own servers and screenshots of it working are included. On the college server there is a place holder profile picture.

List of copied material and Sources

Javascript was used to validate dates on the user side of the system. The code was used and altered from the following link http://www.redips.net/javascript/date-validation/

The placeholder profile picture which is only used because the testweb2 server doesn't accept file uploads can be found at this link: http://www.dailymail.co.uk/news/article-2129246/Once-lifetime-picture-lightning-striking-San-Franciscos-Bay-Bridge.html

All HTML pages were written using Adobe Dreamweaver and the CSS file was written using Dreamweaver too. All HTML pages have been heavily updated and changed around so the only benefit of using Dreamweaver is so the website could look nice.

PHP was learned with the help of ww.phpacademy.org/ the connection file connect.inc.php was derived from an example in one of their tutorial videos. In the file core.inc.php four lines of code are also used from this site.

The idea for imagefinder.php was inspired by a comment on stackoverflow.com but the implementation is our own.