New Features to Implement

Here are a few features related to user accounts that still have to be implemented:

1. Enforce stronger passwords

Add functionality to enforce stronger passwords. On account creation (and on edit account), require that the password be a certain length (at least 5 or 6 characters long) and include at least one number and at least one capital letter, etc. (You can decide what the rules for a strong password should be).

2. On account creation, require that the user also enter in their email (in addition to a username and a password.

We need this feature for (4).

Set the email column in the table NGSTAR_Auth to NOT NULL.
Email addresses should be unique. Set email column in the table NGSTAR_Auth to UNIQUE.
Add server side logic to ensure that the email address being entered on account creation is unique (is not already being used on another account).

3. Functionality to edit account

(Please ensure that (2) has been completed before working on this feature)

The administrator and curator should have the ability to edit (change) their username, email and password when they are logged in. This setting can be accessible from an Account Settings link for example.

Remember that passwords are not stored in the database in plain text (they are salted and hashed before they are stored in the database).
Also check that usernames and email addresses are unique.

4. Functionality to allow the user (admin or curator) to reset their password in case they have forgotten it

(Please ensure that (2) has been completed before working on this feature)

In the log in page, there should be a link that says “Forgot my Password” and when they click on this link it should allow the user to enter in their email address. When they click on submit, an email should be sent (by the NGSTAR application) which will allow the user to reset their password.

http://security.stackexchange.com/questions/1918/can-anyone-provide-references-for-implementing-web-application-self-password-res
http://stackoverflow.com/questions/1102781/best-way-for-a-forgot-password-implementation

Approach:

This is a fairly large feature. I have included a general template of how I would implement a password reset feature based on some of the research that I did in the links above. I would recommend starting with very small features at first, making sure it works and when it does, incrementally add additional functionality. For example, first create a link that says "Forgot my Password". Then figure out how to send emails with Catalyst MVC. Then try sending a demo email with the message "Hello World" when the user clicks on "Forgot my Password". Then build the template and controller that allows the user to enter in the token given to them (without any logic at first). Then include the link to submit the token in the email message. And so on ...

Requirements:

i. You will need to create a new table in NGSTAR_Auth called "password_reset_requests" with the columns "id" (just an incremental id with auto increment set to true of type INT) as the primary key, "token" (TEXT), "timestamp" (DATETIME) and "user_id" as a foreign key which references "id" in the "users" table in NGSTAR_Auth and is a one-to-one relationship. The column "user_id" is later used to associate a token with the corresponding user account.

ii. For now, you can try sending password reset emails via Catalyst MVC by using a gmail account for now. Create a gmail account such as admin.ngstar@gmail.com and send password reset emails from this account (which is done by the application). Make sure you are sending emails securely. Of course, this email will probably need to be changed once we get into production (which should be as easy as changing a few lines in the Catalyst project settings somewhere).

iii. Take a look at how to send email using Catalyst:
http://search.cpan.org/~dhoss/Catalyst-View-Email-0.35/lib/Catalyst/View/Email.pm
http://www.catalystframework.org/calendar/2008/6

Flow of application and logic:

a. User clicks on Forgot My Password
b. User enters in Email
c. User clicks on Submit
d. User will see a notification saying something like "If your email address exists in our

system, an email with further instructions has been sent to you.".

e. Application handles the password reset request with the following logic:

```
Determine if the email addressed entered is associated with a user account.

If the email address exists:
    -Generate a random string token using a CPAN module such as Session::Token
     (http://search.cpan.org/~fractal/Session-Token-0.82/README.pod)
        -Make sure that your tokens have a good length (at least 128 bits)
        -There are several other CPAN modules that can generate random strings.
         A review has been written by another user that compares these CPAN
         modules (http://neilb.org/reviews/passwords.html). If you find an
         alternative module that does an even better job than Session::Token
         then feel free to use that
        -This should be done in the Business Logic layer
    -Store random string token in the "token" column by salting and hashing it
     (do not store token in plain text)
        -This should be similar to salting and hashing user account passwords
    -Determine the current time and store it in "timestamp" column
    -Using the email address, determine the "id" it corresponds to in the
     "users" table in NGSTAR_Auth. In the "password_reset_requests" table,
     set "user_id" to this id.
    -Send an email to the email address that contains:
        -The random string token in plaintext
        -A link to a password reset page
        -A message that says to ignore this email if they did not request to
         reset their password
Else:
    -Do not send an email with password reset instructions
    -Do NOT send an error message back to the user saying that their email
     address was not found (we don't want hackers to know which email addresses
     are in the system)
```

f. User will receive an email that contains
  -Random string token in plaintext
  -A link to a password reset page

```
If User clicks on link to password reset page:

        User will be redirected to password reset page in their browser

        If User pastes in random string token and clicks on submit:

            Upon submit the application logic will be:

            -Get the random string token from the request parameters on a POST
            -Salt and hash this random string token.

            If the salted and hashed random string matches an entry in the
            database:

                -Retrieve the timestamp corresponding to this random string token

                If the timestamp is within the time limit (say 1 hour):
                    -Redirect user to password reset page
                    -User will enter in the new password in one textbox and the
                     same password again in another textbox to verify that both
                     passwords match

                    Upon submit the application logic will be:

                    -Using the token, get the corresponding "user_id" in the
                     "password_reset_requests" table
                    -Using "user_id" find the corresponding "id" in the "users"
                     table. Update the "password" field in the "users" table to the
                     new password. All passwords are salted and hashed before
                     storing them in the database.

                    -Delete the record/row that corresponds with the token (that
                     the user inputted) from the "password_reset_requests" table
                     (since the token should only be good for one valid use)
                    -Clean the password_reset_requests table: delete all other
                     records in the database that are expired (gone over 1 hour)
                     AND are 7 days old
                    -Redirect User to main log in page
                Else:
                    -Send an error message to the user saying that their request
                     has exceeded the time limit and they must submit a new
                     password reset request
            Else:
                -Send an error message to user saying that random string token is
                 invalid
        Else (user clicks on Cancel):
            -Do nothing and redirect to the main log in page
```

Additional features:
A user may forget what email address they used to register for an account. So the user should be able to use their username instead when requesting their password to be reset. This feature is best implemented after resetting the password using an email address is fully functional.

5. Add the random string generator that you used (such as Session::Token) to the installation documentation.

For example, Ubuntu_Desktop_Installation_Guide.md informs the user of what perl modules to install by providing a list of CPAN modules needed. Add Session::Token to this list.

Update all other documentation that specify a list of CPAN modules that are required by the project.

6. Send email to user (via the application) when the password is changed in the Account Settings

The message should say that their password has successfully been updated. Do not include the plaintext password in the email (although this should not be possible because salting and hashing only goes one way, the hash function with a salt is a one-way function).

7. Send email to user (via the application) when the password is changed via the “Forgot my Password” link (after the token has been verified and the user has submitted the new password).

The message should say that their password has successfully been updated. Do not include the plaintext password in the email (although this should not be possible because salting and hashing only goes one way, the hash function with a salt is a one-way function).

8. Be able to sign in using email address

Right now users can sign in using only their username. We want users to also be able to sign in using their email address. In the login page, there should just be one textbox that accepts either a username or an email, plus a textbox for the password.