Summary

Develop a basic web-based interface to submit additions or changes to the class schedule at the college. The class schedule is managed by a central group in Academic Services and various academic departments on campus submit requests for changes. The process is currently performed via printed forms that get approved and sent via inter-office mail. Tracking the forms can be difficult. The goal is to allow authorized staff to submit changes electronically via the web, have the school dean or secretary authorize the change, and then to process the information in the central office.

Various people submit forms, and in this document, these people will be called “requesters”. Examples of requesters include faculty (rarely), program coordinators, department heads, support staff, academic administrative secretaries, and deans. Once a change request is submitted, it needs to be approved. The Dean, or in some cases the Academic Administrative Secretary, can review, modify, and approve the request. Since there are two categories (Academic Secretaries and Deans), Academic Administrative Secretaries will be called “reviewers” or “approvers”, and Deans will be called “approvers” or “deans”, depending on the context.

In some cases, a single person will be a requester, a reviewer, and an approver (such as an Academic Secretary). Therefore, he/she should not have to re-login to perform different actions, which would be inconvenient.

The Schedule Change Request perform five actions:
1. Add a new course
2. Delete an existing course
3. Change an existing course
4. Submit Non-Instruction Assignments (Release Time)
5. Submit sabbatical workload

Vision

It is envisioned that the application will support the same look-and-feel for all users. It is envisioned that the same primary list screen will be used by all users. Depending on the access-level, the list will include different items (using a filter based on access-level to limit access to specific records). Therefore, at the core level, the application is a filter that lists requests based on a users access level and school. Each requester has a primary school (queue) that new requests are assigned to.

The other key functionality of the system is logic built into the forms. The forms will provide different options and access.

Efforts should be made to keep data in a form in the same screen, to reduce the need to scroll up-and-down the form. This is especially important for the Schedulers that have to process data.
Class Schedule Change Request (CSCR)
Project Specifications

Delivery and Components

It is envisioned that three primary scripts will be delivered, along with a database schema.

Web Scripts
  1) Login Script
  2) Queue List (displays requests based on a filter)
  3) Data Entry Script (this might include various sub-scripts to display different forms)
  4) Confirmation Script (after a form is submitted)

User Interaction

The following areas are identified as interaction points in the system. There are a number of requesters that can make changes. Each school has a Dean and Academic Secretary. The following steps simulate the process of submitting a class schedule change from beginning to end.

Access-Levels
Each account can have one of the following access-levels.
  0. Locked: the account is locked.
  1. Requester: can only view his/her own items in the queue.
  2. Reviewer: view is limited to all requests in a particular school/queue. Some Academic Secretaries will have this access-level (cannot approve requests).
  3. Approver: view is limited to all requests in a particular school/queue. This access level can approve requests that exist in the assigned school (queue.) Some Academic Secretaries will have this level (they can approve any request within their own school).
  4. Dean: view is generally limited to all requests in a particular school/queue, but this account can review and approve requests in any school. A dean can open requests from any school as long as the status is AUTH_REQ. If a ticket has the status AUTH_REQ, the Dean can open a request, and can approve or decline the request. All Dean’s will have this access level, so they can approve requests when they are filling in for other Deans. The expectation is that the URL will be emailed to the dean in another school if approval is required.
  5. Scheduler: Scheduler accounts can view all submitted tickets by using search functionality. 6. Administrator: Can view ALL requests.

STEP 1 (Login Script)

The client enters a username and password. Each account has six possible access-levels. Accounts are also associated with a primary school (queue).

NOTE: In some ways this application is similar to the PeopleSoft Ticket System.

STEP 2 (Queue List)

All users are brought to the same screen, which is a list of requests. Depending on the users access-level, the list will include different items (filtered and sorted differently).
The queue listing displays all requests, such as those waiting for approval, were “returned”, or were “completed”.

NOTE: The list will get very large over time, so the display listing should be limited to the past three months. Clients can use search functionality to view older requests.

This screen also displays the option to search for an older record, or to create a new request.

**Basic Example**

| Schedule Change Requests |
|--------------------------|---------------|
| Last Modified | Name | School | Status | Action | Course |
| 7/10/08 9:00AM | Shannon Runningbear | LANG | AUTH_REQ | Change | ENGL10A |
| 7/10/08 8:35AM | Jim Pardon | LANG | AUTH_REQ | Add | FREN229 |
| 7/10/08 7:35AM | Sally Neperbalmquest | LANG | APPROVED | Add | GERM87 |
| 7/10/08 7:34AM | Lester Zanterhexer | LANG | RETURNED | Delete | SPAN872 |

The above example could be from a reviewer, approver, or dean account. Notice that requests are from different users in the same school. A requester will only see his/her own requests.

**STEP 3 (Data Entry)**

This step describes the process of staff creating a schedule change request.

The Schedule Change Form can perform five actions:

1) Add a class
2) Delete a class
3) Change a class
4) Update workload for reassigned time (non-instructional assignment)
5) Update workload for sabbatical.

1) Staff selects “NEW REQUEST” from the main screen.

The web application prompts the user for the type of request.
2) Based on the type of request, a specific form is displayed.
3) The user can type in the subject of the request in the SUBJECT box at the top of the form, such as “Adding ENGL 101 to the Fall Semester.”
4) If the request is a CHANGE, the form appears with the FROM data and TO data listed.
5) Log data is listed at the bottom of the form including who created the form, and who accessed it along the way.
6) The form includes a comments box to make notes. PUBLIC comments are used to communicate between the requester, approver, and schedule, and are displayed on all views of the form data. Comments are available for editing until a form is closed (CANCELLED, DENIED, RETURNED, or COMPLETED). Comments can be edited when the status of a request is: SAVED, AUTH_REQ, APPROVED, RECEIVED, and PENDING.

The requester might type in a comment such as, please give me a nice classroom”.

Example Forms

Adding a Class
Deleting a Class

If the request is a CHANGE, the staff member types the current value in the FROM and the new value in the TO area.

Many of the data fields are similar between Add, Change, and Delete.

The user should be able SAVE the request regardless which fields have data.

7) Staff finishes making changes to the form and selects the SAVE button for later editing (which assigns the ticket a “SAVED” status. They can also select the SUBMIT button to submit the form to the Approver for approval, which assigns the request the status of “APP_REQ”.

If a user opens a previously submitted form with the status of APP_REQ, the requester can still CANCEL the request, and changes the status to CANCELLED. This allows the requester to change their mind. They can later REVISE the cancelled request to copy all of the data from the previous request to a new request for re-submission.

8) During the SAVE & SUBMIT processes, on a CHANGE request, the FROM (before) and TO (after) data are saved to a local database for later comparison and display. If the course data is later changed,
we need to know what the person saw before the change was made – that is why it is important to record
the Before and After data, not just the After.
9) If the form was successfully SAVED or SUBMITTED, the form is assigned a Schedule Change
Request Number (SCRN). The user is provided with a confirmation page that lists the SCRN and a URL
to access the ticket. This SCRN and URL can be used by staff to locate the schedule change request.
10) A SAVED or SUBMITTED Schedule Change Request is assigned to a School (for access purposes),
based on the users account. This provides access to the reviewer(s) and approver(s).
11) The form is assigned the status of “AUTH_REQ” (Submitted) while it waits for approval from an
account with Approver access (in the same school), or a Dean account (in any school).
12) The script sends an email message to the various people assigned to the school (queue), including
REVIEWERS, APPROVERS, and DEANS assigned to the school. A table in the database needs to
include the list of email addresses for each queue/school. The email message includes the URL to
access the request.

STEP 4 (Request Review)

1) Various staff get notified by email that a request has been submitted and is waiting for approval. The
Academic Secretary might get notified, but may not be authorized to approve requests if she/he has
reviewer-level access. If the request is urgent, she/he might contact the Dean to rush the approval.
2) The Academic Secretary can monitor the requests, keeping an eye on requests that are AUTH_REQ,
COMPLETED, and RETURNed.

STEP 5 (Request Approval)

1) The Approver and/or Dean gets an email message that a Schedule Change Form is waiting for
approval. The email notification includes the URL. The Approver/Dean can also review the Queue List
to see is any requests are waiting for approval.
2) The Approver/Dean opens the URL, and authenticates to the application to view the request.
3) Rather than the options at the bottom of the form being “Save” and “Save and Submit”, anyone with
Approver/Dean level access, will instead see “Save”, “Save and Approve”, and “Decline”. The save
exists because changes can be made to the form. If the form is saved, the log should be updated.
4) The Approver/Dean can select to APPROVE or DECLINE the request. Once a request is
APPROVED, it can be reopened, and as long as the request has not been RECEIVED, it can still be
DECLINED (in case the approver/Dean changes their mind). Once a ticket is RECEIVED, it cannot be
DECLINED or CANCELLED. Instead, the client must contact Academic Services and ask for the
request to be RETURNED.
5) Once the Approver/Dean selects “Approve” or “Deny”, the request is saved.
6) A timestamp, IP Address, and Browser Information are stored in the activity log associated with the
request number. This will be used as a form of signature. Logging should be stored in a separate table,
since there could be a one-to-many relationship.
7) Status of the form changes from “AUTH_REQ” to “APPROVED” (or DENIED).
8) An email is sent to the schedulers letting them know a request has been approved and is waiting for
processing. This email list should be customizable.

NOTES: There are three other methods of approving:
Class Schedule Change Request (CSCR)
Project Specifications

1) The school secretary might be authorized to approve on the School Dean’s behalf and would therefore have the access-level of “Approver”.
2) Another Dean might be in charge of the school temporarily and can approve requests.
3) The Dean of Academic Services can approve any request (via Administrator access).

**STEP 6  (Scheduler Form Handling)**

1) The schedulers receive and email message or check the Queue Listing to identify if there is a new request.
2) Requests that are ready to be handled will have a status of “APPROVED”. Forms that have the status “AUTH_REQ” and “DENIED” are generally ignored by the schedulers. A blank list means there are no forms waiting to be processed.
3) Once the scheduler sees the form, they can select “Received” button on the web form, which changes the status of the ticket to RECEIVED.
4) The scheduler reviews the form, and begins processing.
5) The scheduler places comments in the PRIVATE comments box. The form includes an additional comments box for use by the schedulers, called the PRIVATE comments box. Private notes are only visible to Schedulers and Administrators. Requesters, Approvers, and Deans do not see the Private comments. They are used to internally keep notes about the processing of the request. The private comments can always be edited, even after the request is Completed or Returned. PUBLIC comments can still be used by the schedulers to make a comment to the request.
6) If the form is successfully completed, they can type a note in the PUBLIC or PRIVATE comments section on the form, and then select the “Completed” button, which changes the status of the request to “COMPLETED”.
7) If there is a problem with the form, the scheduler can selected “Return” and the status of the request becomes RETURNED. If a request is RETURNED, the web application prompts the scheduler for an explanation of the return. “This form is being returned to the requester because…” This data is logged to the form.
8) If the form was RETURNED to the requester, the script sends an email message to the requester to notify them they the form was not completed successfully, along with the contents of the return explanation.

**STEP 7  (Back to Requester)**

Now we return to the end-user that placed the original request.

1) The requester may get an email if the request was DENIED or RETURNED.
2) The requester opens the application and looks up the form, perhaps by typing in the SCGN number in the search box, using the URL in an email message, or finding the request in their queue.
3) The form displays the PUBLIC NOTES.
4) If the form was RETURNED, a special box will display the reason for the return.
5) If the form is COMPLETED, RETURNED, or DENIED, no changes can be made. It is for reference only.
4) If the form has the status of RETURNED, they can select “REVISE” to clone the request.
5) If the requester selects REVISE, a new form is created with a copy of the data on the previous form, including the comments (PRIVATE comments are not copied). The previous SCRN number is copied
over to the new form and placed in a special field. The previous form data is copied into the form from the previous cancelled form, so they don’t have to re-type the data, and the process begins again. The PUBLIC NOTES are also copied to the new form so they notes can be edited.

6) Once the request is SAVED or submitted for approval, the new request gets a new SCRN (but the old SCRN is still saved in a special field in the new request.

7) A revised request gets a special indicator that it is a REVISED form, and the previous SCRN is listed on the form. The user can open the older form by selecting on a URL in the web form.

8) The user can decide that a previously SAVED form is not longer necessary and can decide to delete the form (so they do not get confused with multiple saved requests that are similar cluttering their queue). The requests status becomes DELETED. This status can only occur if the form was previously in SAVED.

NOTES

1. The request form will remain editable until it is approved. Each time the form is saved or changes status, the activity log related to the form should record who made the change, and the type of change (changes saved, approve, return, complete, etc.)

2. The initial vision of the application specified that the business analyst manually update the table fields when new users were added, rather than designing a web interface for account management. ACIT can decide to build an account management panel at this time.

3. The Business Analyst in Academic Services should have access to the core data.

4. The SCR application will be expanded in the future. Specifically, the same users that submit schedule change requests, also build a schedule from scratch at the beginning of each term. Also, workload for contract faculty will be synchronized between HR and AS. These future applications may integrate with the Schedule Change Request application.

5. There are a few other applications that are planned which may integrate data from multiple sources. Therefore, a single/integrated data schema should be used (rather than storing data in multiple places, or in MS Access files.) For example, we currently manage combinations of courses, and a future application will allow school secretaries to build combinations. If the CSCR application removes a class from a combination, it should check to see if the change affects a combination.

6. Default list view. Users should by default see forms in their listing that are up to six months old, at which time they no longer appear on the list. Users can use the ability to search for older forms using a search form. Requests should never be removed from the system.

7. Requests should generally not be actually deleted, but users that have “SAVED” forms, that were never submitted, should be able to “remove” them from the list by “deleting” them, which would essentially hide them. The status can be changed from “SAVED” to “DELETED”, and they should not appear on the users list.
Class Schedule Change Request (CSCR)
Project Specifications

8. Each Schedule Change Request is associated with a School (a Queue), so that the staff and Approver/Dean assigned to that school/queue can see the requests. The quantity and names of schools change, so the schools should not be hard-coded, but rather accessed via a database table. The database schema needs to allow for the addition of new schools and staff.

Schools/Queues

The Queue names and school accounts should be as follows:

BSS – Business and Social Sciences
CAAS – Creative Arts and Applied Sciences
HSM – Health, Science, and Mathematics
LA – Language Arts
PE – Physical Education
LRTT – Learning Resources, Teaching, & Technology
TRADES – Trades and Industrial Technology

Queue Notes

Each School gets a queue.
Each account is assigned to a specific School/Queue.
Schedulers can see all queues.
Administrator accounts can see all queues.

Access-Levels

Each account can have one of the following access-levels.
0. Locked – The account is locked, no access.
1. Requester – Submit requests.
2. Reviewer – Review and monitor requests for entire school/queue.
3. Approver - Approve requests that are assigned to the accounts school/queue.
4. Dean - Approve a request in any queue (but their view is limited to their own school)
5. Scheduler – Schedulers process the requests.
6. Administrator: Monitor and support the requests.

Request Status

Each ticket can be in one of the following status categories.
1. SAVED: The request was never submitted, but the data is saved.
2. DELETED: The requester decides to delete a previously saved request.
3. APP_REQ (Submitted) : The request was submitted and waiting for approval.
4. APPROVED: The request is approved.
5. DECLINED: The request was declined by the Approver or Dean.
6. RECEIVED: The request was received by Academic Services.
7. PENDING: The request is on “hold”.
8. COMPLETED: The request has been processed.
9. RETURNED: The request was returned to the requester by Academic Services.
10. CANCELLED: The request was cancelled by the requester.
11. EXPIRED: If a request remains in AUTH_REQ for too long, it can be EXPIRED.

**Request Revision**

Once a ticket is APPROVED, it can no longer be edited (except for the comment boxes). Once a ticket is COMPLETED or RETURNED, the public comment box can no longer be edited. A DECLINED or RETURNED ticket can be REVISED, which copies the data to a new ticket. The new ticket should have a status indicator specifying that it is a Revision, and the original ticket number should be included with a URL to open the previous ticket in a new window. This allows the NEW request to be LINKED to the OLD request.

Example of a REVISION on a Request:

Revision: Yes
Prev Request: 12332

By selecting the SCRN “12332” above, the previous ticket will be opened in a new window for review.

**Logging**

Each ticket should have an unrestricted number of log entries. Changes to each request should be recorded.

Here is a proposed sample of the logging database. Only records associated with a particular request (via the scrn #) will be displayed for a specific request.

<table>
<thead>
<tr>
<th>SCRN</th>
<th>Timestamp</th>
<th>UserID</th>
<th>Status</th>
<th>Type</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>7/1/2008 2:34PM</td>
<td>jsmith</td>
<td>SAVED</td>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>676</td>
<td>7/1/2008 2:34PM</td>
<td>lbaum</td>
<td>SUBMITTED</td>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>7/1/2008 3:35PM</td>
<td>hgerol</td>
<td>SUBMITTED</td>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>7/1/2008 4:35PM</td>
<td>rtuft</td>
<td>APPROVED</td>
<td>Add</td>
<td>IP=123.11.231.1, Browser: Mozilla.</td>
</tr>
<tr>
<td>123</td>
<td>7/1/2008 4:45PM</td>
<td>cgower</td>
<td>RECEIVED</td>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>7/1/2008 5:05PM</td>
<td>cgower</td>
<td>COMPLETED</td>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>287</td>
<td>7/1/2008 6:45PM</td>
<td>jbreton</td>
<td>COMPLETED</td>
<td>Delete</td>
<td></td>
</tr>
</tbody>
</table>

**LIST VIEWS**

**Access-Level Primary Views**

Each account can have one of the following access-levels.

0. **Locked**: this account is locked and nothing is visible.
1. **Requester**: can only view his/her own items in the queue.
2. **Reviewer**: view is limited to all requests in a particular school/queue. Some Academic Secretaries will have this access-level (cannot approve requests).
3. **Approver**: view is limited to all requests in a particular school/queue. This access level can approve requests that exist in the assigned school (queue.) Some Academic Secretaries will have this level (they can approve any request within their own school).

4. **Dean**: view is generally limited to all requests in a particular school/queue, but this account can review and approve requests in any school. A dean can open requests from any school as long as the status is AUTH_REQ. If a ticket has the status AUTH_REQ, the Dean can open a request, and can approve or decline the request. All Dean’s will have this access level, so they can approve requests when they are filling in for other Deans.

5. **Scheduler**: Scheduler accounts can view all submitted tickets by using search functionality. Their primary list is limited to APPROVED, SUBMITTED, and PENDING requests (requests waiting for processing). They cannot edit forms (except for the public and private comment boxes.)

6. **Administrator**: Can view ALL requests. The list should display AUTH_REQ, APPROVED, SUBMITTED, PENDING, and RETURNED requests sorted by date.

Sample Views

**Example view for a Requester:**

<table>
<thead>
<tr>
<th>School</th>
<th>Username</th>
<th>Status</th>
<th>Modified</th>
<th>Type</th>
<th>Term</th>
<th>Course</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>RETURNED</td>
<td>08/09/08</td>
<td>D</td>
<td>1420</td>
<td>AVMT</td>
<td>224</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>SAVED</td>
<td>08/09/08</td>
<td>N</td>
<td>1425</td>
<td>AVMT</td>
<td>221</td>
</tr>
</tbody>
</table>

**Example view for Reviewer/Approver:**

<table>
<thead>
<tr>
<th>School</th>
<th>Username</th>
<th>Status</th>
<th>Modified</th>
<th>Type</th>
<th>Term</th>
<th>Course</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADES</td>
<td>Senna Koll</td>
<td>AUTH_REQ</td>
<td>08/09/08</td>
<td>C</td>
<td>1415</td>
<td>AVMT</td>
<td>225</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>RETURNED</td>
<td>08/09/08</td>
<td>D</td>
<td>1420</td>
<td>AVMT</td>
<td>224</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>APPROVED</td>
<td>08/09/08</td>
<td>N</td>
<td>1425</td>
<td>AVMT</td>
<td>221</td>
</tr>
<tr>
<td>TRADES</td>
<td>Jill Smith</td>
<td>COMPLETED</td>
<td>08/12/08</td>
<td>R</td>
<td>1410</td>
<td>Dept Head</td>
<td>32554</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>APPROVED</td>
<td>08/09/08</td>
<td>C</td>
<td>1415</td>
<td>AVMT</td>
<td>244</td>
</tr>
</tbody>
</table>

Note: The reviewer/approver needs to see all of the tickets in the school, plus their own (which can include SAVED). The scheduler should not see SAVED forms by other users (since they are limited to a single users view), but this could change. Since the requester can Submit the form, and the form is still editable, the requester can submit the form to make the form visible by the reviewer/approver.

Note: An account with access-level of Dean will look similar to Review/Approver, they this account has access to open requests in other schools that have the status “AUTH_REQ” (submitted for approval).

**Example view for Scheduler:**

<table>
<thead>
<tr>
<th>School</th>
<th>Username</th>
<th>Status</th>
<th>Modified</th>
<th>Type</th>
<th>Term</th>
<th>Course</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADES</td>
<td>Senna Koll</td>
<td>APPROVED</td>
<td>08/09/08</td>
<td>C</td>
<td>1415</td>
<td>AVMT</td>
<td>225</td>
</tr>
<tr>
<td>BSS</td>
<td>John Smith</td>
<td>APPROVED</td>
<td>08/09/08</td>
<td>D</td>
<td>1420</td>
<td>BUS</td>
<td>224</td>
</tr>
<tr>
<td>CAAS</td>
<td>Jack Smith</td>
<td>RETURNED</td>
<td>08/09/08</td>
<td>N</td>
<td>1425</td>
<td>CA</td>
<td>221</td>
</tr>
<tr>
<td>PE</td>
<td>Jill Smith</td>
<td>RETURNED</td>
<td>08/12/08</td>
<td>R</td>
<td>1410</td>
<td>Dept Head</td>
<td>32554</td>
</tr>
<tr>
<td>LRTT</td>
<td>Gene Wong</td>
<td>COMPLETED</td>
<td>08/09/08</td>
<td>C</td>
<td>1415</td>
<td>LEARN</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Schedulers see Approved, Returned, and Completed.

**Example view for Administrator (everything):**
Class Schedule Change Request (CSCR)

Project Specifications

<table>
<thead>
<tr>
<th>School</th>
<th>Username</th>
<th>Status</th>
<th>Modified</th>
<th>Type</th>
<th>Term</th>
<th>Course</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADES</td>
<td>Senna Koll</td>
<td>APPROVED</td>
<td>08/09/08 14:45</td>
<td>C</td>
<td>1415</td>
<td>AVMT 225</td>
<td>23665</td>
</tr>
<tr>
<td>BSS</td>
<td>John Smith</td>
<td>APPROVED</td>
<td>08/09/08 14:45</td>
<td>D</td>
<td>1420</td>
<td>BUS 224</td>
<td>29887</td>
</tr>
<tr>
<td>CAAS</td>
<td>Jack Smith</td>
<td>APPROVED</td>
<td>08/09/08 14:45</td>
<td>N</td>
<td>1425</td>
<td>CA 221</td>
<td></td>
</tr>
<tr>
<td>PE</td>
<td>Jill Smith</td>
<td>RETURNED</td>
<td>08/12/08 14:45</td>
<td>R</td>
<td>1410</td>
<td>Dept Head</td>
<td>32554</td>
</tr>
<tr>
<td>LRTT</td>
<td>Gene Wong</td>
<td>RETURNED</td>
<td>08/09/08 14:45</td>
<td>C</td>
<td>1415</td>
<td>LEARN 1</td>
<td>23321</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>RETURNED</td>
<td>08/09/08 14:45</td>
<td>D</td>
<td>1420</td>
<td>AVMT 224</td>
<td>29887</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>SAVED</td>
<td>08/09/08 14:45</td>
<td>N</td>
<td>1425</td>
<td>AVMT 221</td>
<td></td>
</tr>
<tr>
<td>TRADES</td>
<td>Senna Koll</td>
<td>REVIEW*</td>
<td>08/09/08 14:45</td>
<td>C</td>
<td>1415</td>
<td>AVMT 225</td>
<td>23665</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>RETURNED</td>
<td>08/09/08 14:45</td>
<td>D</td>
<td>1420</td>
<td>AVMT 224</td>
<td>29887</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>APPROVED</td>
<td>08/09/08 14:45</td>
<td>N</td>
<td>1425</td>
<td>AVMT 221</td>
<td></td>
</tr>
<tr>
<td>TRADES</td>
<td>Jill Smith</td>
<td>COMPLETED</td>
<td>08/12/08 14:45</td>
<td>R</td>
<td>1410</td>
<td>Dept Head</td>
<td>32554</td>
</tr>
<tr>
<td>TRADES</td>
<td>John Smith</td>
<td>APPROVED</td>
<td>08/09/08 14:45</td>
<td>C</td>
<td>1415</td>
<td>AVMT 244</td>
<td>38892</td>
</tr>
</tbody>
</table>

Note: Administrator sees everything except for “Deleted”.

Type Codes:  N = New, C = Change, D = Delete, R = Release

* AUTH_REQ is the same as “submitted”.

If possible, sorting should be user-accessible by selecting the column. Forms that are DELETED should not be displayed.

Each link can be selected to open the form.

**ACCOUNTS**

The system needs to have permanent logs. Are userids at LBCC permanent (non-reusable)? If someone in 1995 used the userid ‘JSMITH’ and leaves the college in 2000, can another person in 2009 be assigned the same userid “JSMITH”? If so, the userid itself (“JSMITH”) should not be used as the unique identifier in the system. Instead, the accounts table should use a separate serial number which is incremented each time a new user is added. The UserID (serial number) would be associated with a Login.

Obviously, only one person with the same Username can access the system at the same time. For example, if an account was previously created for “jsmith”, and they leave, and later another person comes along with “jsmith”, the previous “jsmith” should be removed from the database field identifying the login id, but the rest of the record should remain.

A username is accessed by converting the UserID (such as 003443) and looking up the full name of the user. It is possible that there would be the same name for two users. For example, there could be two John Smiths. By opening the form, and looking at the detail, there should be a uniquely listed UserID (003443) to differentiate which John Smith submitted the form. The user can

The Schedule Change Request application should NOT use the “username” of a person. The “username” should only be used to connect the person’s login with the Microsoft Active Directory account that currently exists. Instead, the application should use the person’s full-name.

Sample fields in the Account table:

UserID: 00110  A unique serial number permanently assigned to this record.
Login: jsmith  The username associated with the AD account for Login.
This is the account that the user logs in with. This field can be BLANK if another JSMITH is hired.

Name: John Smith This is the name displayed in the application.
School: 1 The school.
Access 1 The access level.

Example:

<table>
<thead>
<tr>
<th>UserID</th>
<th>Login</th>
<th>Name</th>
<th>School</th>
<th>Access</th>
<th>Locked</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>John Smith</td>
<td>1</td>
<td>0</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>089</td>
<td>jsmith</td>
<td>John Smith</td>
<td>5</td>
<td>1</td>
<td>N</td>
</tr>
</tbody>
</table>

In the above example, the Login “jsmith” was previously used, but at some point, this person left. At some point another person was reassigned the Login “jsmith”. Since the table can only have one “jsmith”, the LOGIN ID for the previous jsmith must be blanked out to allow another record to have that LOGIN ID.

**DATA MODEL NOTES**

During the process of detailing the specifications for this application, ideas on the data model came to mind.

**Table: Accounts**

Each account should be associated with a school and an access-level. Each account should be lockable, without having to delete the account since we need to preserve history.

<table>
<thead>
<tr>
<th>UserID</th>
<th>Login</th>
<th>Name</th>
<th>School</th>
<th>Access</th>
<th>Locked</th>
<th>Add Date</th>
<th>Lock Date</th>
<th>School</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>00456</td>
<td>jsmith</td>
<td>Jesse Smith</td>
<td>NO</td>
<td>8/7/2009</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table: Schools**

Each school should be defined in a schools table.

<table>
<thead>
<tr>
<th>School</th>
<th>School Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Language Arts</td>
</tr>
<tr>
<td>2</td>
<td>FL D</td>
</tr>
<tr>
<td>3</td>
<td>ESL D</td>
</tr>
<tr>
<td>4</td>
<td>MATH</td>
</tr>
</tbody>
</table>