Description

GroupShare is a catalog database for a co-operative set of members to share items among eachother. Primary goal is to allow users to share their books, dvd, and cd with the group members. Members list items they are willing to share and retain ownership of the item. The item is assigned a estimated replacement value. At any given point in time users can only borrow from the GroupShare up to a maximum of the total value of items that they have put up available for sharing (Total Checked out <= Total Contribution). Items out on loan can have a queue of requests against them for next in line for loan. Loans can be up to a indefinite period of time unless a loan request is made against the item, at which time the item must be passed onto the next in line within 2 weeks.

Members must provide contact information where they can be reached as well as emails for notification of items available/request queued against on loan items.

Ratings / comments on the item can be made by people who have used the item for other users to browse

Transactions

1. post a new item into loan pool
2. Show all items in share stuff with checked in/out status
3. request a loan item (either add to wait queue or initiate a borrow request)
4. shows total value available for an person - calculate total contribution, total on loan, balance remaining (R)
5. show all items not on loan (currently available) (R)
6. see item transaction history/comment history, current queue (R)
7. cancel loan request
8. remove item from loan pool - only allowed by owner
9. make a comment on item
10. get user score - total contribution value/items, popularity of items offered, qty of items borrowed. eg a goodness rating for the member to encourage contributions by members, (R)

Project will be done on UBC linux servers using apache web server with php and oracle as the backend.

Exclusions - while this is a multi user system, no implementation of user security will be done in this version of implementation. However, since we need to emulate user specific views (eg show me my stuff, check out item for me, add me to queue, add stuff for others to borrow from me) we need to define user specific views as well as an administrator view that can access all. At this point it is unclear how we emulate a user login - perhaps a global php variable or cookie, that all queries will have to be able to acces to customize view and abilities for user.

Initial Work Plan:

The work will be done as a group up to the database implementation and then each of us will take a portion of the query pages/reports and implement against identical copies if the base database. We will then integrate all pages into a single server along with test data at the end stage. We decided to divide work vertically to ensure everyone had a chance to get exposure to the full end to end implemtation.

Rough Schedule:

	Richard	Rafael	Kevin	Yaphet
Initial Draft of ER	X			
Translation to schema -	X	X		
ER and schema	X	X	X	X
iinitial Translation to SQL/ initial oracle database tables	X		X	
Definition of output targets	X	X	X	X
Initial apache, php, oracle test impl	X			
apache server implemenation	X	X	X	X
Refine Initial SQL and schema	X	X	X	X
query and report generation	X	X	X	X
web page implmentation	X	X	X	X
generate test data	X	X	X	X
weekly integration of all pages/reports	X	X	X	X
final integration of all pages/reports	X	X	X	X
final report	X	X	X	X

First pass Definition of output

0. HomeScreen : Link to all other pages and reports. All pages should have link back here
    a. possibly a drop down menu to set userID for all other pages to base their views/queuries

1. ShowAllStuff: all items in share stuff with checked in/out status - output table of
    a. ItemID - link to 5 ItemDetail (ItemID)
    b. Description,
    c. status (available or not)
        i. if not available, current queue list
        ii. if available button to check out
    d. member comments on item
    e. button link to 8. MakeComment(ItemID)
    f. indicator showing if current user is queued on shown item
        i. if currently, queued, button to remove from queue
    g. link/button to 2 LoanItToMe(ItemID)

2. LoanItToMe(ItemId): request a loan item (either add to wait queue or initiate a borrow request): script behind item 1. Result should be
    a. web page confirming queue entry and show current queue list or
    b. web page confirming dispatch of item request to owner

3. PersonStatReport: shows total value available for an person - calculate total contribution, total on loan, balance remaining Report showing statistics **for current member**
    a. Total items in pool
    b. Total value of items in pool
    c. number of items that have ever been loaned
    d. total item*days of loans (sum of days on loan for all items owned by member)
    e. average loan period for all items that have ever been loaned (exclude items that have never been checked out)
    f. Top 3 items (items with longest queues at any given point in time)
    g. Top 3 items (items borrowed most frequently - highest totals)
    h. Top 3 items (items out the longest - highest totals)
    i. call to 9. User Goodness(person.email) showing result

4. ShowAvailStuff: show all items not on loan (currently available) - Indentical to 1 but only for items currently available

5. ItemDetail(ItemId): show
    a. item transaction history - who, when, what
    b. comment history,
    c. queue history

    d. current queue
    e. current status : where is it now

6. ManageLoansRequest
    a. list of all items with wait queue
    b. each item shows all queued members against given item
    c. for each outstanding queue, show current age
    d. for those items that current user is queued against, show button for removing queue request - implement php implementation with result confirmation page

7. ManageMyLoanPool: List of items owned by current user showing
    a. itemid
    b. item descption
    c. date added
    d. times borrowed
    e. total days on loan
    f. longest queue
    g. list of borrowers with days on loan
    h. button for remove item for each item - script to execute. Note this should not delete item from database but record a removal transaction and make item unavailable for other users to check out but history details should be preserved.
    i. button to go to add item screen.

8. MakeComment(ItemID) on item
    a. Item ID
    b. Item Description
    c. input screen for text
    d. other current comments

9. UserGoodness(person.email)- = Total (Value of Items Loaned * Days on Loan) -Total (Value of Items Borrowed * Days on loan) : positive number means net contributor, negative means net borrower. eg a goodness rating for the member to encourage contributions by members. Returns single numeric value of unit dollarDays.

10. GlobalPersonStat: show comparitive stats by person for all members
    a. Total items in pool
    b. Total value of items in pool
    c. number of items that have ever been loaned
    d. total item*days of loans (sum of days on loan for all items owned by member) (desireablity)
    e. average loan period for all items that have ever been loaned (exclude items that have never been checked out) (attractiveness flag)

f. total items*days for borrowed items by given user (sum of days for all items borrowed by this member) (appetite flag)
g. average days on load for all items ever borrowed for given user (hog flag)
h. User Goodness = Total (Value of Items Loaned * Days on Loan) -Total (Value of Items Borrowed * Days on loan) : positive number means net contributor, negative means net borrower

Schema and ER

- Minor changes to attribute names. As we discovered we had inadvertently used sql reserved words
- Largley the same as milestone 1.
- Transactions will be a weak entity tied to item and user through a borrowReturn relationship. This will allow saving and processing of transaction histories so that statistics can be obtained - items still out, average loan period per item, etc.
- User ratings will be on a per item basis. The rating is intended to capture the comments regarding the specific item condition - eg, “this book is missing pages.”

Identified Hurdles

- How to identify user - login screen saves userid for global php variable to prevent user specific views?
- How to re-integrate everyones work on a regular basis to avoid divergence of solution parts.
- Review of above target deliverables to ensure we can actually implement - both from a database and from a user presentation view. As our group has no prior experience with either HTML or php, the implementation details are not clear enough yet that we can state that this is the final deliverable target. While we have done early test implementations of the web server and some html and php test implementations, deliverable Items may be added or removed depending upon what we learn about what capabilities we actually have at hand.

Feedback requests

- any general comments appreciated
- any specific comments related as to the mechanics of user specific identification. eg is it acceptable to have a drop down menu on the front page to emulate a specific user so that all the subsequent pages will present views and queries from that users perspective? Can we store it in a php global variable and will all pages have access to it or do we have to send it along with the link or button click?