GameSwap Report

Table of Contents:

1. **Data Types**

User

Attribute	**Data type**	**Nullable**
email	String	Not Nullable
password	String	Not Nullable
firstName	String	Not Nullable
lastName	String	Not Nullable
nickName	String	Nullable
location	Location *(custom type)	Not Nullable
phoneNumber	String	Nullable
phoneNumberType	String	Nullable
showPhoneNumber	Boolean	Not Nullable
rating	Double	Not Nullable
listedItems	List<Item> *(custom type)	Not Nullable
allItems	List<Item> *(custom type)	Not Nullable
isAvailableForSwap	Boolean	Not Nullable

Location

Attribute	**Data type**	**Nullable**
city	String	Not Nullable
state	String	Not Nullable
postalCode	String	Not Nullable
latitude	Float	Nullable
longitude	Float	Nullable

Item

Attribute	**Data type**	**Nullable**

name	String	Not Nullable
listingNumber	Integer	Not Nullable
gameType	String	Not Nullable
details	String	Not Nullable
condition	String	Not Nullable
description	String	Nullable
hasPreviousSwap	Boolean	Not Nullable
hasSwapPending	Boolean	Not Nullable
mySwapHistory	List<Swap> *(custom type)	Not Nullable

Swap

Attribute	**Data type**	**Nullable**
proposer	User *(custom type)	Not Nullable
counterparty	User *(custom type)	Not Nullable
proposalDate	Date	Not Nullable
finalStatusDate	Date	Not Nullable
proposedItem	Item *(custom type)	Not Nullable
counterpartyItem	Item *(custom type)	Not Nullable
finalStatus	String	Not Nullable
ratingEnabled	Boolean	Not Nullable

2. Business Constraints / Pseudocode

Underlined items are forms or GUI items.
Italicized items are “classes” or attributes of classes.
Bold are super-sections.

User

- When accessing the app initially, all *users* will be directed to the login form.
- Users can login with an *email* or *phone number*, and *password*.

- A register link must be provided on the login form.
- *Phone type* is limited to either home, work, or mobile.
- An *email* can only be registered once in the system. Check if the email the user is attempting to register already exists in the database.
- If *number of proposed swaps* user has received is greater than zero, display link to accept/reject swap form.
    - If this is greater than five, the number should be printed in bold and in red.
- If *number of unrated swaps* user has is greater than zero, display link to swap rating page.
    - If this is greater than two, the number should be printed in bold and red and user can not propose another swap.
- *Rating* must be rounded to hundreths, or display “None” if there is none.
- *Rating,* when entered after a completed swap, is a choice from 0 to 5 (whole numbers).
- Logout button must be visible which returns the *user* to the login form.
- If a user has any *unapproved swaps* or *unrated swaps*, can not make any updates to user info.
- If a *user* is trying to update user info, they can not update their email address only.
- If a *user* attempts to change phone number but it is already in use, appropriate error message shown.

Item

- *Game type* must be one of: board game, card game, video game, computer game, jigsaw puzzle.
- If *game type* is jigsaw puzzle:
    - Attribute *piece count* is now present.
- If *game type* is computer game:
    - *Platform* is limited to choices of Linux, macOS, or Windows.
- If *game type* is video game:
    - *Platform* must be listed as Nintendo, Playstation, or Xbox. This list of values should be updatable in DB (not necessary to update it in the GUI).
- *Game condition* must be one of: mint, like new, lightly used, moderately used, heavily used, or damaged/missing parts.
- User-entered *description* is optional.
- If a user has more than two unrated swaps, or more than five unaccepted swaps, they can not list a new item.
- Only fields needed for chosen game are displayed.
- When viewing item, if no description exists, don’t show the field.
- If viewing item belonging to another user:
    - Their nickname, city, state, postal code should be displayed with swapper rating rounded to hundreths and distance to user rounded in tenths (unless in same postal code in which case the field shouldn’t be displayed).
    - If distance is between 0 and 25.0 miles, highlight with a green background.
    - If distance is between 25.0 and 50.0 miles, highlight with a yellow background.

- If distance is between 50.0 and 100.00 miles, highlight with an orange background.
- If distance is over 100.0 miles, highlight with a red background and warning message shown.

Search

- Searches should be available by keyword, postal code, within x miles, and within a specific postal code.
- If description is greater than 100 characters, show ellipsis at end.
- If keyword search, match keyword highlighted with blue background in the results.
- Detail button should be provided to show item's specifics.
- If within a specific postal code:
  - Ensure postal code is valid.
- If within x miles:
  - X must be a user-entered whole number.

Swap

- Items that are associated with a pending swap are not available to swap.
- Items that are associated with completed swaps are not available to swap.
- If a user has a listed item, they are able to swap items with each other.
- If a swap is rejected, that specific item-for-item swap can not be proposed again.
- If a swap is accepted, the counterparty will see the contact info for the proposer.
- To mark swap as completed, after swapping items, users must rate each other on a scale of 0-5.
- Items that have been swapped cannot be swapped again.
- If % of swaps rejected historically is greater than 50.0%, highlight number in red.
- If users have not rated each other yet, show Rating option in swap history page.

3. Task Decomposition / Abstract Code

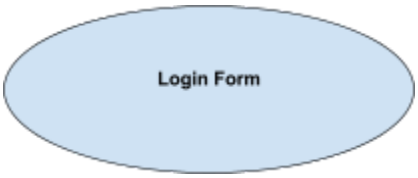
Rule of Thumbs

- Lookup vs insert, delete, and update?
- How many schema constructs are involved?
- Are enable conditions consistent across tasks?
- Are frequencies consistent across tasks?
- Is consistency essential?
- Is mother task control needed or not?

Abstract Code Formatting:
<u>Underlined items are forms, buttons, or other GUI items.</u>

Italicized items are “classes” or attributes of classes.

Login Form View
Task Decomposition



Lock Types: Read only for email/phone number and password
Number of Locks: 1
Enabling Conditions: User must click the login button
Frequency: High
Consistency (ACID): Order is critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
Login
- *User* enters an *email* or *phoneNumber*, and a *password*.
- Data validation occurs for all inputted fields.
- User hits Register button:
    - Go to Registry
- *User* hits Enter button:

If (*email* != empty AND *phoneNumber*!=empty):
    Error(“Login with either email or phone number”)
If(*email*==found AND !*password*.verify())
    Error(‘Incorrect password for ‘+email)
If(*phoneNumber*==found AND !*password*.verify())
    Error(‘Incorrect password for ‘+phoneNumber)
if((*email*==found AND *password*.verify()) OR (*phoneNumber*==found AND *password*.verify())
    Go to Main Menu

User Registration Form View
Task Decomposition

Lock Types: Insert User information(email, nickname, password, city, first name state, last name, postal code, number(opt).
Number of Locks: 1
Enabling Conditions: User must click the register button on the Login Form View

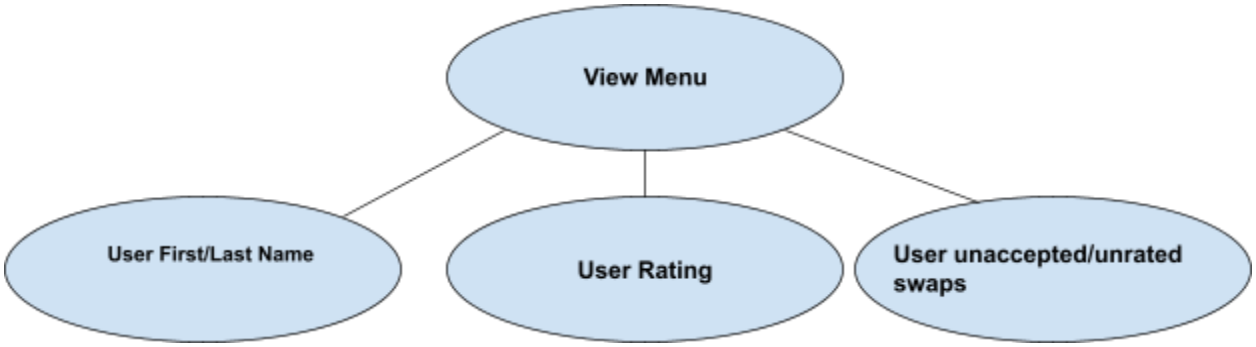
Frequency: Low
Consistency (ACID): Order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
Registry

- Display text boxes for the following fields: *email, nickname* (optional), *city, first name, last name, state, postal code, phone number* (optional).
- Display password box (any inputted text is masked by '*') for field *password*.
- Display *checkbox* for a boolean to show phone number in swaps or not.
- Display *Register* button at bottom.
- *User* hits *Register button*:
    - Do data validation.

Check for email or phone existing in system already.

Main Menu View
Task Decomposition



Lock Types: Read only lookup for Name, Rating, unaccepted/unrate swaps
Number of Locks: 2
Enabling Conditions: User must successfully login
Frequency: High
Consistency (ACID): Order is not critical
Subtasks: Mother Task is needed to display two different data together.

Abstract Code:
Main Menu

- Display "welcome" message at the top along with name of application, GameSwap.
- Show *Logout* button, *My Rating* field, *Unaccepted Swaps* field, *Unrated swaps* field, *List Item* button, *My Items* button, *Search Items* button, *Swap History* button, and *Update my Info* button.
  Clicking any of these buttons leads to a gui form of their respective names (e.g. clicking the *My Items* button will lead the *user* to the My Items gui page). Each of these gui pages is described in below sections.
- *MyRating* field is rounded to 2 decimal places, *Unaccepted Swaps and Unrated Swaps* fields are both whole numbers (integers).

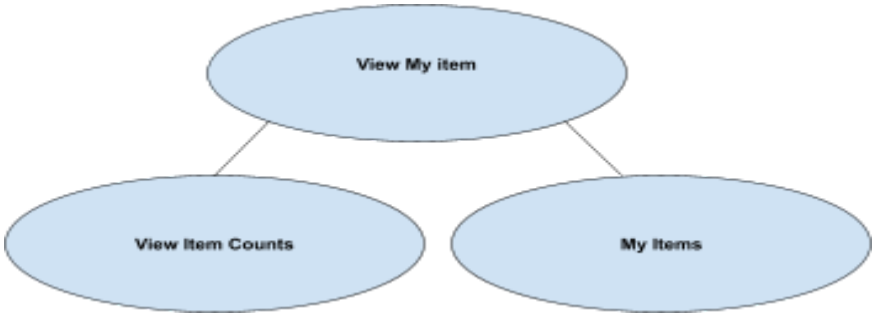
- On every single other page, there should be a link/button at the bottom that leads back to Main Menu.

List Item Form View
Task Decomposition

Lock Types: Read only Game Type/Condition, Insert (If all non-optional fields have been filled, display "List Item" button)
Number of Locks: 2
Enabling Conditions: If a user has any unapproved swaps or unrated swaps, can not make any updates to user info.
Frequency: High
Consistency (ACID): Order is not critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
List Item
- If *user.unratedSwaps* > *2* OR *user.unacceptedSwaps* > 5 , disallow listing new items and display message.
- Only show appropriate fields (for the game type chosen). See Business Logic Constraints for more details.
- Once data is validated and List Item button is clicked, display a success alert msg showing the cardinal *itemID.*

View My Items Form View
Task Decomposition



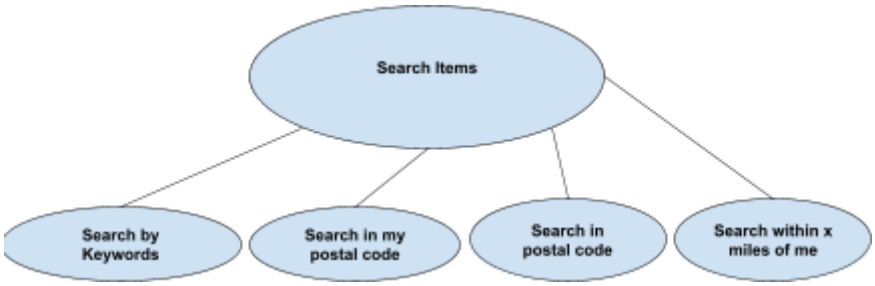
Lock Types: Read only Items, and items counts
Number of Locks: 2

Enabling Conditions: My items is enabled only if total number of items is > 0
Frequency: High
Consistency (ACID): Order is critical
Subtasks: Mother Task needed. Items Game Type must be counted

Abstract Code:
My Items

- Display in section called “Item Counts” a table with the following names and the respective number of instances: board games, card games, computer games, jigsaw puzzles, video games, total.
- Display in section called “My Items” a table with fields: item # (cardinal id *itemID* of the item), *gameType*, *gameTitle*, *condition*, *description*, and link to detail page for that item.

Search Items Form View
Task Decomposition



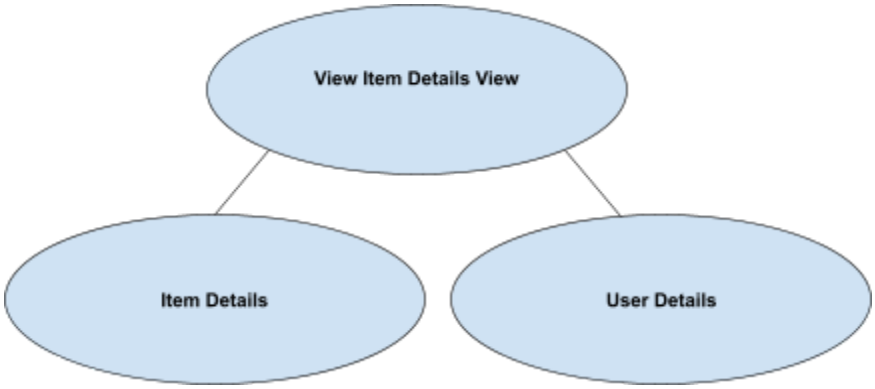
Lock Types: Read only Item
Number of Locks: 1
Enabling Conditions: User must choose at least one of the 4 options of searching.
Frequency: High
Consistency (ACID): Order is not critical
Subtasks: Mother Task needed, item search in 4 different ways.

Abstract Code:
Search Items

- Display form with four choices: by keyword, in my postal code, within x miles of me, and in postal code.
- These choices are checkboxes, meaning only one can be selected at a time.
- When Search button is clicked:
    - Perform initial data validation of all fields
    - If no results are found, display a message with text “Sorry, no results found!”
    - If results were found:
        - The form title should indicate what kind of search was executed.
        - If the *search* was a keyword search, highlight the matching text in blue.

Display all items with a link to Detail page.

View Items Form View
Task Decomposition

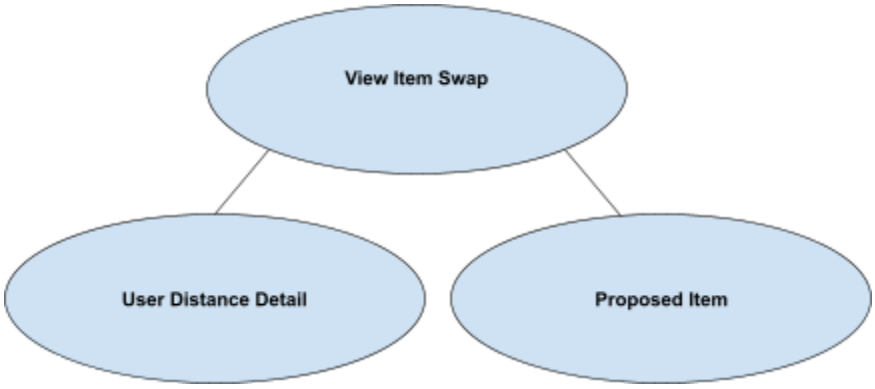


Lock Types: Read only Item Details, User detail (First/Last Name, Location, Rating, Distance
Number of Locks: 2
Enabling Conditions: Items exist
Frequency: High
Consistency (ACID): Order is not critical
Subtasks: Mother Task is needed to display two different data together.

Abstract Code:
View Items

- Display form with all item details as mentioned in Business Constraints section.
- If item belongs to another user, display their nickname, postal code, swapper rating, distance to user (assuming it's different postal code than user). Again see Business Constraints section for more details.
- These choices are checkboxes, meaning only one can be selected at a time.
- See Business Constraints section for highlighting of distance.
- If user clicks Propose Swap:
    - Jump to the Propose Swap GUI page.

Propose Items Swap View
Task Decomposition

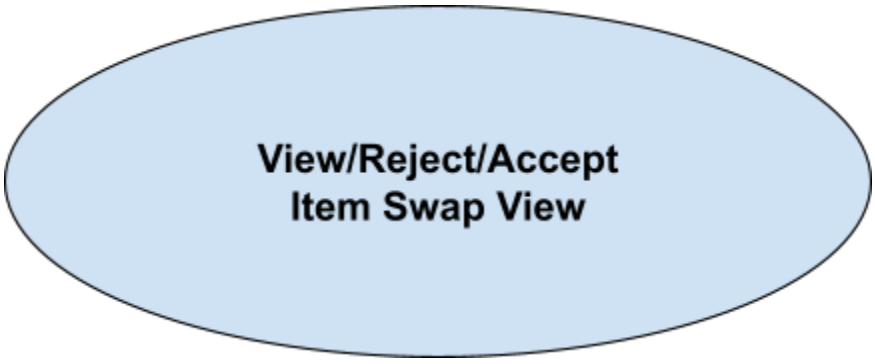


Lock Types: Read only User, Update proposed Item
Number of Locks: 2
Enabling Conditions: Users must not have completed and pending swaps.
Frequency: High
Consistency (ACID): Order is critical
Subtasks: Mother Task is needed to display two different data together.

Abstract Code:
Propose Swap

- Display form with all swap details as mentioned in Business Constraints section.
- This form should contain a list of the user's items, and a
- These choices are checkboxes, meaning only one can be selected at a time.
- If user clicks Confirm button:
    - Change status of the *swap* to completed. This will ensure that rating is enabled for users to each other.
    - Add swap to *swap history.*

Accept/Reject Swap Form View
Task Decomposition



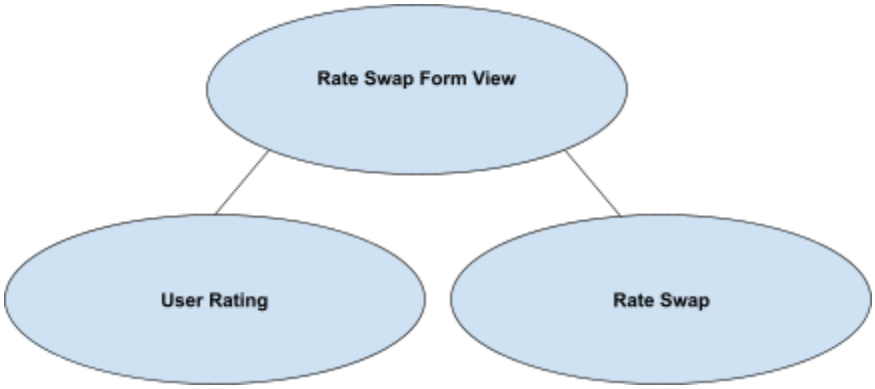
Lock Types: Read only Swap Detail, Update Proposed Item
Number of Locks: 1
Enabling Conditions: Item must be proposed

Frequency: High
Consistency (ACID): Order is critical
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code:
Accept/Reject Swap

- Display form which shows the desired item from the other user, and two buttons Accept and Reject.
- If the user clicks either button, record the date in the database.
- If user clicks Reject:
    - Ensure this swap can not be proposed again.
- If user clicks Accept:
    - Display a dialog with proposer's email, first name, phone number/type if available and if sharing option is set.
    - Remove the item from listing.
    - If no more swaps exist, return *user* to Main Menu automatically.

Rate Swap Form View
Task Decomposition



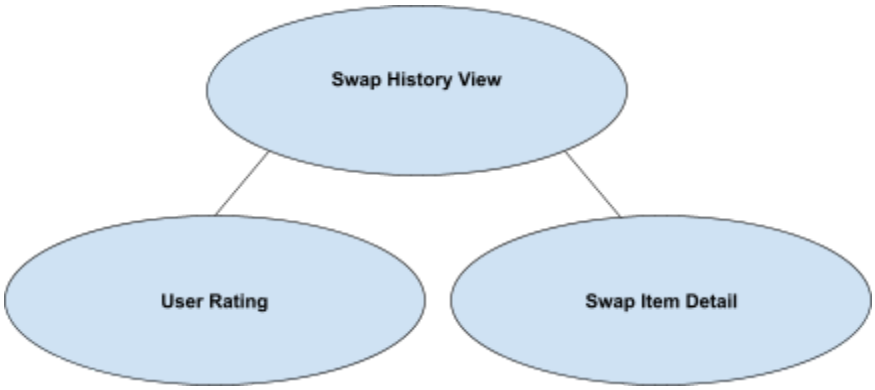
Lock Types: Read only Swap Detail, Update User Rating
Number of Locks: 2
Enabling Conditions: Item must be swapped
Frequency: High
Consistency (ACID): Order is critical
Subtasks: Mother Task is needed to display two different data together.

Abstract Code:
Rate Swaps

- Display form which shows the swaps a user has completed, and a dropdown with whole numbers 0,1,2,3,4,5 descending with the title *Rating*.
- This will only be enabled after a swap has been completed and items are traded.

- Selecting a rating will record it in database and remove swap from this list.

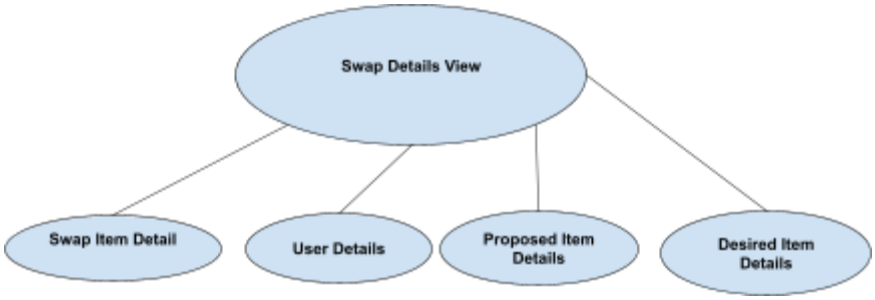
View Swap History Form View
Task Decomposition



Lock Types: Read only Swap Detail, Update User Rating
Number of Locks: 2
Enabling Conditions: Item must be swapped
Frequency: High
Consistency (ACID): Order is critical
Subtasks: Mother Task is needed to display two different data together.

Abstract Code:
Swap History
- Display form with entire user history of their swaps.
- Top part of form will display a summary showing the roles a user has played (proposer and counterparty), the total number of swaps, accepted, rejected. See rules on highlighting in Business Constraints section.
- Second table will be the list of all swaps including proposed date, accepted/rejected date, swap status, role, proposed item, desired item, other user, *option to rate if this hasn't been done already, and link to details page for swaps.
- If user clicks the Detail link for a swap:
    - Go to Swap Details gui page.

Swap Detail Form View
Task Decomposition

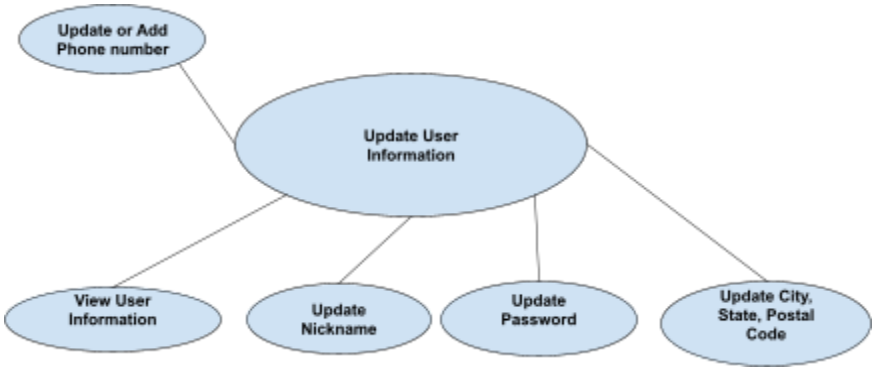


Lock Types: Read only Swap Detail, Proposed Item Details, User Details, Desired Item
Number of Locks: 4
Enabling Conditions: Item must be swapped
Frequency: High
Consistency (ACID): Order is not critical
Subtasks: Mother Task is needed to display different datas together.

Abstract Code:
Swap Details

- Display form that will show the details of the selected swap including proposed date, accepted/rejected date, swap status, swap type, and the rating if done (if not allow option to do it).
- Also display other users' nickname, distance, and if an accepted swap, their user details including first name, email, and phone number if the option to share is set.
- This is a display screen and will not have any buttons on it apart from going back to the Main Menu.

Update User Form View
Task Decomposition



Lock Types: Read only email and phone type, update nickname,password, city, first/last name, state, postal code, phone number
Number of Locks: several
Enabling Conditions: enable by user clicking update on main menu
Frequency: High

Consistency (ACID): Order is not critical
Subtasks: Mother Task is needed

Abstract Code:
Update user information

- Same as the Registry gui but changing email is not allowed.
- If user has any unapproved swaps (as proposer or counterparty) or unrated swaps, they can not make any updates so grey out the Update button and show appropriate error message.
- If user attempts to change phone number:
    - Check if this is in use by another user already. Show appropriate error message if so.
- If user clicks Update button:
    - Do basic data validation as before.
    - Make changes in database and return user to Main Menu.