# Table of Contents

# Data Types:

**User**

| **Attribute** | **Data Type** | **Nullable** |
|---|---|---|
| email | String | Not Null |
| password | String | Not Null |
| first_name | String | Not Null |
| last_name | String | Not Null |
| nick_name | String | Null |
| city | String | Not Null |
| state | String | Not Null |
| postal_code | Integer | Not Null |
| phone_number | String | Null |
| phone_number_type | String | Null |
| share_phone | Boolean | Null |

**Item**

| **Attribute** | **Data Type** | **Nullable** |
|---|---|---|
| name | String | Not null |
| condition | String | Not null |
| item_number | Integer | Not null |
| type_name | String | Not null |
| platform | String | Null |
| pieces | int | Null |
| media | String | Null |

**Swap**

| **Attribute** | **Data Type** | **Nullable** |
|---|---|---|
| swap_status | String (Enum) | Not Null |
| proposer_rating | Integer | Null |
| counterparty_rating | Integer | Null |
| proposed_date | Date | Not Null |
| accept_reject_date | Date | Null |

# Business Logic Constraints

### User

- Users who are new to GameSwap must register first.
- Users who have an existing GameSwap account will not be able to Register.
- User after login will have the name displayed in the main page

### Items

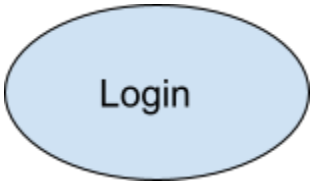
- User cannot list the same item more than once
- Only Items available for swapping can be searched
- If a user has more than two unrated swaps, or more than five unaccepted swaps, They cannot list a new item.

### Swaps

- Items associated with a pending or completed swap are not available for swapping.
- A user cannot swap items with themself.
- A user with no available listed items cannot swap.
- If a swap is rejected, that specific item-for-item swap cannot be proposed again.
- If a user has more than 2 unrated swaps, or more than 5 unaccepted swaps they cannot propose a swap.

# Login

## Task Decomp




**Lock Types**: Read-only on RegularUser table

**Number of Locks**: Single

**Enabling Conditions**: None

**Frequency**: Around 200 logins per day (High frequency)

**Consistency (ACID)**: not critical, order is not critical.

**Subtasks**: Mother Task is not needed. No decomposition needed.

**Abstract Code**

- When the user clicks the **Register** button, navigate to the register form.
- User enters email and password  input fields.
- If data validation is successful for both email and password input fields, then:
    - When the **Login** button is clicked:
    - If the User record is found but the user.password != '$Password':
        - Go back to the **Login** form, with an error message.
    - Else:
        - Store login information (email)
        - Go to the **View Profile** form.
- Else *email* and *password* input fields are invalid, display **Login** form, with error message

# User Registration

## Task Decomp


**Lock Types**: Write and Read on the RegularUser table

**Number of Locks**: Single

**Enabling Conditions**: None

**Frequency**: Medium

**Consistency (ACID)**: Is needed to avoid duplicate phone numbers/emails.

**Subtasks**: Mother Task is not needed. No decomposition needed.

## Abstract Code:

In the **Registration** form, the User is prompted to enter the details below :-

1. *First Name*
2. *LastName*
3. *Nick Name*
4. *Email*
5. *City*
6. *State*
7. *Postal Code*
8. *Password*
9. *Address Type*
10. *Phone Number* [Optional field]

Other than *Phone number*, all other fields are mandatory.

If any of the mandatory fields are not entered, the User is prompted to enter the missing details when **Register** button is clicked

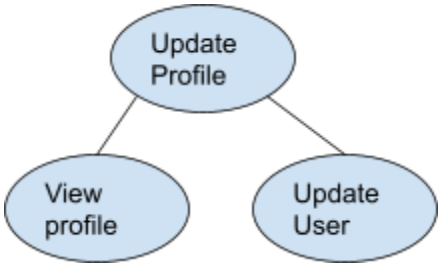
If the *email* entered is already existing in the database, the User should be displayed with an error message stating to use another email address.

Similarly, if a *phone number* is entered and is already existing in the database, the User should be displayed with an error message.

If a *phone number* is entered, then a **checkbox** needs to be enabled, to let the user select if "Phone number needs to be displayed in swaps"

# Update User Information

## Task Decomp




## Task Decomp

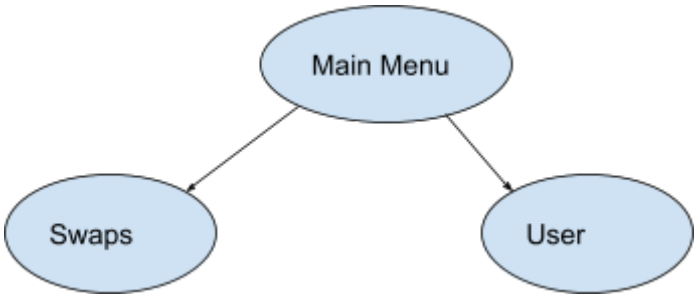
**Lock Types**: 1 read and 1 write lookups User info for a RegularUser
**Number of Locks**: schema construct is needed
**Enabling Conditions**: User login should be successful
**Frequency**: Low
**Consistency (ACID)**: Is needed to avoid duplicate phone numbers
**Subtasks**: The View profile task must be done before update. Update is optional.

## Abstract Code

- Retrieve user information using email as identifier and display information.
- If needed, User should be given an option to edit and update the profile (email is not editable).
- Once the **Update** profile button is clicked, all the **Registration** validations will apply.
- Update user information to the database and refresh page.

# Main Menu Page

## Task Decomp




## Task Decomp

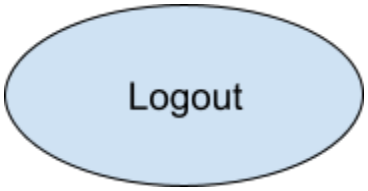
**Lock Types**: 1 read lookups User, 1 for Swaps
**Number of Locks**: schema construct is needed
**Enabling Conditions**: User login should be successful
**Frequency**: High
**Consistency (ACID)**: is not critical
**Subtasks**: All tasks must be done, but can be done in parallel. Mother task required to coordinate subtasks. Order is not necessary.

## Abstract Code

- In the **MainMenu** Page, the User's *first name* and *last name* need to be displayed.
- Retrieve User first and last name using email
- Retrieve associated swaps using email and display number of unaccepted swaps and unrated swaps.
- Users have an option to logout by clicking the **logout** button.
- Users have an option to view **unrated swaps**, if any, by clicking the link under unrated swaps.
- Similarly, **unaccepted swaps** can be also viewed by clicking the link under the unrated swaps.
- My ratings will be displayed based on the calculations.
- If the **"list item"** button is clicked, User will have an option to list a new item.
- To look for all the items listed by the User, **the My Items** button needs to be clicked.
- Users have the option to search items by clicking the Search **Items** button. To view swap history, the **Swap History** button needs to be clicked.

# Logout

## Task Decomp




**Lock Types**: Read only on the User  table

**Number of Locks**: Single

**Enabling Conditions**: User should be logged in

**Frequency**: Medium

**Consistency (ACID)**: not critical, order is not critical.

**Subtasks**: Mother Task is not needed. No decomposition needed.

## Abstract Code

If the **Logout** button is clicked in the **MainMenu** page, the User will be navigated to the **Login** page.

# List Item

## Task Decomp


**Lock Types**: 1 Read lock for users swap rates, 1 Read/Write lock for User items

**Number of Locks**: Several different schema constructs are needed

**Enabling Conditions:**

- User login
- User has item
- Item is not already listed

**Frequency**: Moderate frequency for read items and users, and Low frequency for write items.

**Consistency (ACID)**: Critical, all items and user details need to be up to date before an item is listed

**Subtasks**: All tasks must be done. The Mother task is required to coordinate subtasks. Tasks need to happen in order: first Read Items, then Read User swap rates and finally Write Item

## Abstract Code

- User clicked List Item from the Main Menu
- Run the **Read Item** task for the Item (Proposed Item): query information about the Item Type
    - Ensure the fields needed for the chosen game are listed
- Run the **Read User** task for the proposer: query information about unrated swaps.
    - If a user has more than two unrated swaps, or more than five unaccepted swaps, they cannot list a new item and an appropriate message should be displayed.

- When the List Item button is pressed then:
    - **Write Item** to the database with the associated item type, name, condition, and item number
    - Display confirmation message

- Return to **Main Menu**.

# My Items

## Task Decomp


**Lock Types**: Read only on the Item table, Read only on User, Read only on Swap

**Number of Locks**: 1 Read lock for Item 1 Read lock for User 1 Read lock on Swap

**Enabling Conditions:**

- User login

**Frequency**: Moderate frequency for read items and read user

**Consistency (ACID)**: Critical, all items and user details need to be up to date before items are shown

**Subtasks**: All tasks must be done. The Mother task is required to coordinate subtasks. Tasks need to happen in order: first Read Items, then Read User

## Abstract Code

- User clicked My items from the Main Menu
- Run the **Read Item** task for the Item (Proposed Item): query information about the Item
    - **Read Item**: query information about the Item by selected criteria: Keyword, in my postal code, within x miles of me, in postal code:
        - **Item number**
        - **Count of items for each game type**
        - **Total of items**
        - **Game type**
        - **Item name/title**
        - **Condition**
        - **First 100 chars of description**
        - **Distance from user**

- Display search results of the queried attributes sorted by distance and item number in ascending order. If searched by keyword then fields that matched the keyword should be highlighted with a blue background.

# Search Item


**Lock Types**: Read only on the Item table

**Number of Locks**: 1 Read lock for Item

**Enabling Conditions:**

- User login
- Item exists

**Frequency**: Moderate frequency for read items

**Consistency (ACID)**: not critical, order is not critical.

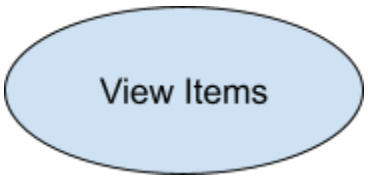
**Subtasks**: Mother Task is not needed. No decomposition needed.

## Abstract Code

- User clicked Search items from the Main Menu
- Run the **Read Item** task for the Item (Proposed Item): query information about the Item by selected criteria: Keyword, in my postal code, within x miles of me, in postal code:
    - **Read Item**: query information about the Item by selected criteria: Keyword, in my postal code, within x miles of me, in postal code:
        - **Item number**
        - **Game type**
        - **Item name/title**
        - **Condition**
        - **First 100 chars of description**

        - **Distance from user**
- Run the **Read User** task for the Swap(Proposer): query information about the User
    - **Read Swap**: query information about the Swap details
        - **Swap rating**
- Display search results of the queried attributes sorted by distance and item number in ascending order. If searched by keyword then fields that matched the keyword should be highlighted with a blue background.

# View Item




**Lock Types**: Read only on the Item table, Read only on the User table

**Number of Locks**: 1 Read lock for Item, 1 Read item for User

**Enabling Conditions:**

- User login
- Item exists
- Item offered

**Frequency**: Moderate frequency for read items

**Consistency (ACID):** Critical, all items and user details need to be up to date before an item is viewed

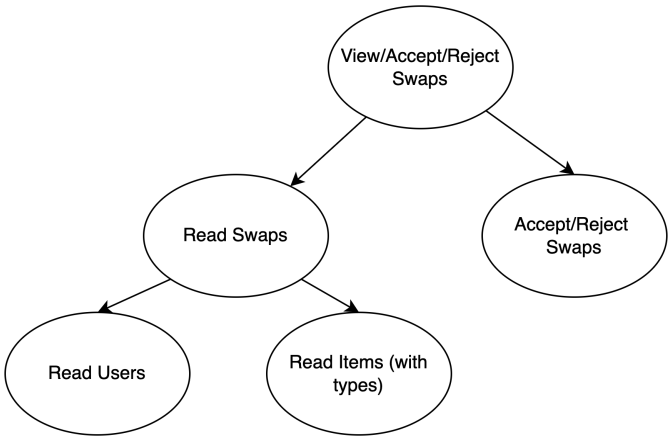
**Subtasks:** All tasks must be done. The Mother task is required to coordinate subtasks. Tasks need to happen in order: first read item details, then read user details

## Abstract Code

- User clicked Details from the Search Items menu
- Run the **Read Item** task for the Item (Proposed Item): query information about the Item
    - **Read Item**: query information about the Items details
        - **Item number**
        - **Game type**
        - **Item name/title**

        - **Platform**
        - **Condition**
        - **Distance from user**
        - **Media**
- Run the **Read User** task for the User(Proposer): query information about the User
    - **Read Item**: query information about the Items details
        - **Name**
        - **Location**
        - **Rating**
        - **Distance**
- Display search results of the queried attributes

# Accept/Reject Swap




## Task Decomp

**Lock Types:** 2 read locks for **items**, **users** and 1 read/write lock to **swaps**

**Number of Locks:** Several different schema constructs are needed

**Enabling Conditions:**

- User login
- User has at least 1 pending swap

**Frequency:** Moderate frequency for read items, users and types and Low frequency for accept/reject swaps.

**Consistency (ACID):** Critical, all items and user details need to be up to date before a swap is accepted/rejected

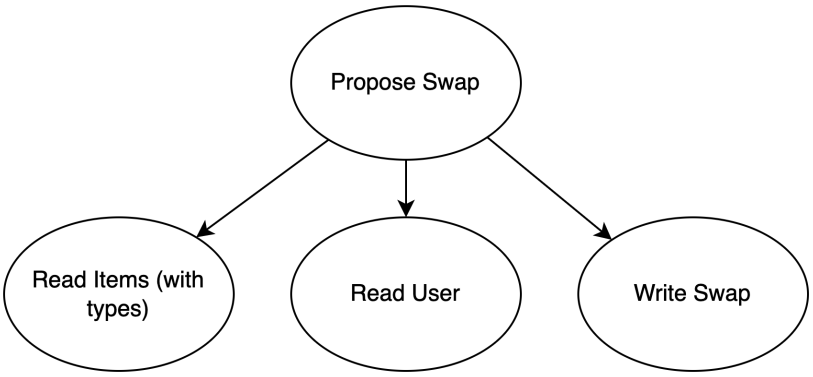
**Subtasks:** All tasks must be done. The Mother task is required to coordinate subtasks. Tasks need to happen in order: first View Swaps: Read user, then Read items. Then, Accept/Reject Swap

## Abstract Code

- User clicked the number in the **Unaccepted Swaps** panel
- Run the **Read Swaps** Task:
    - Find associated swaps using the logged user email and querying for swap status: “pending”
    - **Read Users**: query information about the proposers using their email.
        - Nickname
        - **Calculate Rating**
        - **Calculate Distance**
    - **Read items**: query information about the proposed items
    - Display pending swaps
- When the Accept button is pressed:
    - Confirm consistency of items
    - **Accept Swap:**
        - Record the Accepted Date
        - Update Swap Status to “accepted”
    - display a dialog with the proposer’s email, first name, and phone number/type, if available and if sharing option is set.
    - Run the **Read Swaps** task to repopulate the list (removing the accepted)
- When the Reject button is pressed:
    - **Reject Swap:**
        - Record the Rejected Date
        - Update Swap Status to “rejected”
    - Run the **Read Swaps** task to repopulate the list (removing the rejected)
- If no more swaps are pending return to **Main Menu**

# Propose Swap

## Task Decomp




**Lock Types:** 3 read locks for **items**, **users** and 1 read/write lock to **swaps**

**Number of Locks:** Several different schema constructs are needed

**Enabling Conditions:**

- User login
- The user has available items to swap (proposed items that were rejected for that desired item are not available)
- The user has less than 3 unrated swaps, AND less than 6 unaccepted swaps.

**Frequency:** Moderate frequency for read items, users and types and Low frequency for the write swap

**Consistency (ACID):** Critical, all items and user details need to be up to date before a swap is proposed

**Subtasks:** All tasks must be done. The Mother task is required to coordinate subtasks. Tasks need to happen in order: first Read user, then Read items and finally Write Swap

## Abstract Code

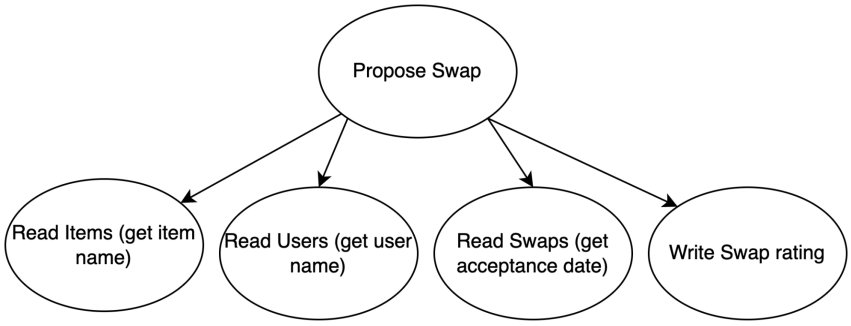
- User clicked Propose Swap from the Item Details view
- Run the **Read User** task for the proposer (logged user): query information about the location, pending swaps and unrated swaps.
    - If the number of unrated swaps is greater than 2 OR the number of unaccepted swaps is greater than 5, then return to the previous screen and alert the user
- Run the **Read User** task for the counterparty: query information about location.

    - Calculate the distance between the users' stored addresses.
    - If the counterparty is >= 100.0 miles from the user, display a warning message containing that distance.

- Run the **Read Items** Task: query information about the proposer's available items to populate the list.
    - Remove items that are already associated with a swap (proposed or desired)
- Run the **Read Swaps** Task: query information about the swaps
- When the Confirm button is pressed AND an item is selected, then:
    - **Write Swap** to the database with the associated items, users and proposed date
    - Display confirmation message
    - Return to **Main Menu**.

# Rate Swap

## Task Decomp




**Lock Types:** 3 read lock for **swaps, user and item** and 1 write lock to **swaps**

**Number of Locks:** Several different schema constructs are needed

**Enabling Conditions:** The user has completed swaps, Some swaps are unrated

**Frequency:** Mid frequency for querying and filtering swaps, and low frequency for the swap rating write

**Consistency (ACID):** Not critical.

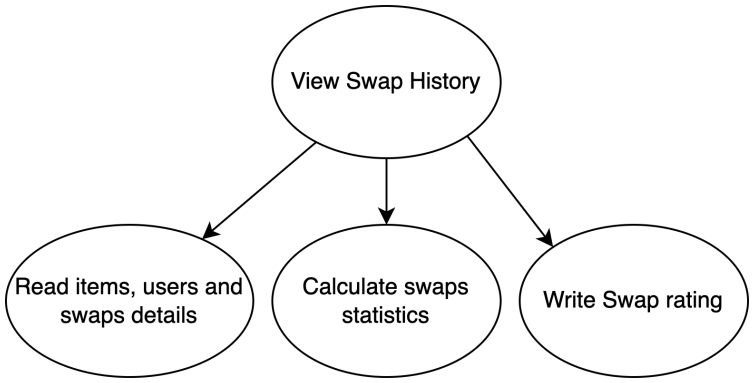
**Subtasks:** None

## Abstract Code

- User clicks **Unrated Swap** in the Main Menu view
    - When there is no unrated swap, showing ‘Empty’ message
        - Querying swaps from database, filtering by user is proposer or counterparty, and there is no corresponding proposer or counterparty rating.
        - Querying item and user details
        - Display swaps in descending order of acceptance date, with item and user details
        - User input 0-5 rating for any swaps
        - User clicks submit
        - Writing rating into database
    - Else:
        - showing ‘Empty’ message
- User clicks **Return** button to return to the **All Swap** tab

# View Swap History

## Task Decomp




**Lock Types:** 3 read lock for **swaps, item and user,** and 1 write lock to **swaps**

**Number of Locks:** Several different schema constructs are needed

**Enabling Conditions:**

- The user has swaps

**Frequency:** High frequency for querying and filtering swaps, and low frequency for the swap rating write

**Consistency (ACID):** Not critical

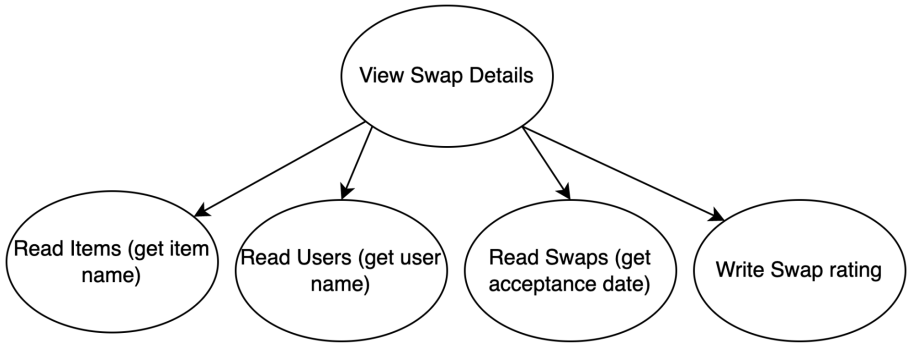
**Subtasks:** optional view swap details before rating

## Abstract Code

- User clicks **Swap History** from the Main Menu view
- Querying swaps from database, filtering by user is proposer or counterparty
- Calculating user swap statistics
- Display swaps statistics and a list of all swaps.
    - When there are unrated swaps
        - Show “submit” button
        - User can input 0-5 rating for any swaps
        - User clicks submit
        - Writing rating into database
- (optional) User clicks **Details** and enter next detail page.
- User clicks **Return** button to return to the **All Swap** tab

# View Swap Details

## Task Decomp




**Lock Types:** 3 read locks for **swaps, items, users,** 1 write lock for **swap**

**Number of Locks:** Single

**Enabling Conditions:**

- The user has swaps

**Frequency:** High frequency for querying and filtering swaps,

**Consistency (ACID):** Not critical

**Subtasks:** optional rating

## Abstract Code

- User clicks **Details** in the **Swap History** page
- Querying **item** details
- Querying proposer and counterparty user information, Calculating distance between them
- Querying items details
- Display proposer and counterparty and their items.
- Querying **swap** details
    - When it is unrated swap
        - Show “submit” button
        - User can input 0-5 rating for any swaps
        - User clicks submit
        - Writing rating into database
- User clicks **Return** button to return to the **All Swap** tab