**Data Types**

|  |  |
| --- | --- |
| User | |
| Attribute | Data Type |
| Email | nvarchar(30) |
| Name | nvarchar(30) |
| 4 Digit Pin | nvarchar(4) |
| Follow | List<User> |

|  |  |
| --- | --- |
| Corkboard | |
| Attribute | Data Type |
| Title | nvarchar(30) |
| Visibility | boolean |
| Owner | User |

|  |  |
| --- | --- |
| PushPin | |
| Attribute | Data Type |
| Date/Time | DateTime |
| Owner | nvarchar(30) |
| Url | nvarchar(100) |
| Description | nvarchar(200) |
| Tags | list< nvarchar(30)> |
| Likes | integer |
| Comments | list<Comment> |

|  |  |
| --- | --- |
| Comment | |
| Attribute | Data Type |
| Date/Time | DateTime |
| Text | nvarchar(100) |
| User | User |

|  |  |
| --- | --- |
| Like | |
| Attribute | Data Type |
| User | User |
| PushPin | PushPin |

|  |  |
| --- | --- |
| Category | |
| Attribute | Data Type |
| Type | nvarchar(50) |

**Business Logic & Constraints**

**User**

1. A user can only follow a corkboard that they do not own
2. A user cannot like their own PushPins
3. A user can only view private corkboards with a password
4. No user can watch a private corkboard
5. Must have a 4 digit pin
6. Must have an email address

**Corkboards**

1. A corkboard title cannot be null
2. A corkboard visibility cannot be null
3. A corkboard category cannot be null
4. A private corkboard password cannot be null
5. A corkboard is only updated when a new PushPin is added
6. A corkboard cannot switch between public and private access
7. A corkboard cannot be deleted

**Login**

1. A user must enter an email address and a 4 digit pin

**Pushpins**

1. The URL for a PushPin cannot be null
2. The Description for a PushPin cannot be null
3. A PushPin comment cannot be deleted
4. A PushPin on a private corkboard will never appear on anyone’s search result

**Tags**

1. Tags can contain all characters except commas

**Task Composition**

**Log In**

**Lock Types:** Read on USER

**Frequency:** low

**Consistency:** non-critical

**Schema:** Single

**Abstract Code**

* User enters *email* and *pin*
* User clicks <**Login>** button:
  + if form is validated, then:
    - if *email* does not have a known record, login page is redisplayed with an error message
    - else if *email* is found, and *pin* does not match, then login page is redisplayed with error message
    - Else create a session with the login information and proceed to Home Screen
  + Else show an error message indicating the form was not validated

**View Home Page**

**Lock Types:** Read on USER, PERSONAL, UPDATES, CORKBOARDS

**Frequency:** High

**Consistency:** USER must be looked up first to retrieve the additional locks

**Enabling Conditions:** Must be successfully logged in

**Schema:** Multiple

**Subtasks:** Personal, Updates and Corkboards after a successful user lookup

**Abstract Code**

* Find the current USER using the *email*
* Find the Corkboards followed by the USER
* Find users followed by USER
* Display the USERs username
* Find each update to boards that are owned or watched my USER, or owned by a user followed by USER
  + Sort updates and display the 4 newest updates
  + If no updates
    - Display message “No Updates”
* Display all Corkboards Sorted alphabetically that are owned by USER
* While no buttons are pushed, do nothing
* When a button is push, then do:
  + if POPULAR\_TAGS, **View Popular Tags**
  + if ADD\_CORKBOARD, **Add Corkboard**
  + if PUSHPIN\_SEARCH, **Search PushPins**
  + if CORKBOARD\_UPDATE, Write to Corkboard, **View Corkboard**
  + if MY\_CORKBOARD, **View Corkboard**
  + if CORKBOARD STATS, **View Stats**
  + if POPULAR\_SITES, **View Popular Sites**

**Add Corkboard**

**Lock Types:** Write to USER corkboards

**Frequency:** Low

**Consistency:** non-critical

**Schema:** Single

**Subtasks:** none

**Abstract Code**

* Display window with input forms and a dropdown containing Corkboard Categories alphabetically
* if USER selects Private
  + USER must enter password
* USER click Add
* Write to USER the new corkboard
* While no buttons are pushed, do nothing
* If a button is pushed , then do:
  + if ADD, Write to Corkboard, **View Corkboard**

**View Corkboard**

**Lock Types:** Read to Corkboard for Owneraship, Category, and Images; Write for selecting Followed

**Frequency:** Low

**Consistency:** non-critical

**Schema:** Single

**Subtasks:** none

**Abstract Code**

* Find the OWNER, CATEGORY, and IMAGES of the CORKBOARD
* Display OWNER, CATEGORY, and IMAGES
* If OWNER is current USER, Enable ‘Add PushPin’ button
* If current USER is not OWNER, display ‘follow’ button
* If current USER is not OWNER and CORKBOARD is public, display ‘watch’ button
* Display number of watchers
* While no buttons are pushed, do nothing
* If a button is pushed, then do:
  + if FOLLOW, Write to USER, **View Corkboard**
  + if ADD\_PUSHPIN, Write to Corkboard, **Add Pushpin to Corkboard**
  + if WATCH, Write to USER, **View Corkboard**
  + if <image>, **View PushPin**

**Add PushPin to CorkBoard**

**Lock Types:** None

**Frequency:** Low

**Consistency:** non-critical

**Enabling Condition:** USER must own

**Schema:** Single

**Subtasks:** none

**Abstract Code**

* Display window with input forms
* Display CATEGORY
* While no buttons are pushed, do nothing
* If a button is pushed, then do:
  + if ADD, Validate Form, Write to Corkboard, **View Corkboard**

**View PushPins**

**Lock Types:** Read OWNER of pin; Read DATE and TIME of initial pin on board; Read COMMENTS, LIKES, TAGS of pin

**Frequency:** High

**Consistency:** non-critical

**Schema:** Single

**Abstract Code**

* Find OWNER of PushPin
* Display OWNER
* Find DATE and TIME of Pin
* Display DATE and TIME of Pin
* Find CATEGORY of Pin
* Display CATEGORY
* Find COMMENTS and LIKES
* Display LIKES
* Display COMMENTS sorted by date/time
* Find TAGS of Pin
* Display TAGS sorted alphabetically
* While no buttons are pushed, do nothing
* If a button is pushed, then do:
  + if FOLLOW, Write to USER, **View PushPin**
  + if *<hyperlinked Category>*, **View Corkboard**
  + if <*image>*, GOTO Hyperlinked Site
  + if POST\_COMMENT, Write to PushPin, **View PushPin**
  + if THUMBS\_UP, Write to PushPin, **View PushPin**

**Search PushPins**

**Lock Types:** Read PUSHPINS

**Frequency:** Low

**Consistency:** Lookup PushPins before looking up other attributes. Otherwise NULL

**Schema:** Single

**Abstract Code**

* Find related PushPins
* For Each PushPin in related PushPins:
  + Find OWNER, and CORKBOARD the Pin is on
  + If CORKBOARD is public
    - Display alphabetically PUSHPIN{hyperlink}, CORKBOARD, OWNER
* While not button is pushed, do nothing
* When button is pushed, then do:
  + if PUSHPIN\_DESCRIPTION, **View PushPin**

**View Popular Tags**

**Lock Types:** Read POPULAR\_TAGS

**Frequency:** Low

**Consistency:** Successful POPULAR\_TAGS needs to be done before other queries

**Schema:** Single

**Abstract Code**

* Find Popular Tags
* For Each Tag in related Popular Tags:
  + Find #PUSHPINS, and #CORKBOARD the Tag is on
  + Display TAG{hyperlink}, #PUSHPINS, and #CORKBOARDs sorted by most Pushpins followed by most Corkboards
* While not button is pushed, do nothing
* When button is pushed, then do:
  + if TAG, Search Tags

**View Popular Sites**

**Lock Types:** Read POPULAR\_SITES

**Frequency:** Low

**Consistency:** Successful POPULAR\_SITES needs to be done before other queries

**Schema:** Single

**Abstract Code**

* Find Popular Sites
* For Each Site in related Popular Sites:
* Find #PUSHPINS the Site is on
* Display SITE, #PUSHPINS sorted by most PushPins
* While not button is pushed, do nothing
* When button is pushed, then do:
  + if TAG, Search Tags

**View Stats**

**Lock Types:** Read USERS

**Frequency:** High

**Consistency:** Successful Read of all users before Read of Stats

**Schema:** Single

**Abstract Code**

* Find all USERS
* For Each USER within USERS
  + Find PUBLIC CORKBOARDS
  + Find PUBLIC PUSHPINS
  + Find PRIVATE CORKBOARDS
  + Find PRIVATE PUSHPINS
  + Display number of PUBLIC CORKBOARDS→ {Sorted by most: Wt=1}
  + Display number of PUBLIC PUSHPINS
  + Display number of PRIVATE CORKBOARDS → {Sorted by most: Wt=2}
  + Display number of PRIVATE PUSHPINS