Software Requirements  
Specifications

Last Revision Date: 10/19/11

1. Product Overview and Summary
2. Information Description
   1. User Interface
   2. High Level Data Flow Diagram
   3. Data Structure Representation
   4. Data Elements Dictionary
3. Functional Description
   1. Functions
   2. Processing Narrative
   3. Design Constraints
   4. Detailed Data Flow Diagrams
4. Performance Requirements
5. Exception Conditions and Exception Handling
6. Implementation Priorities
7. Foreseeable Modifications and Enhancements
8. Acceptance Criteria
9. Sources of Information
10. Revision History

1. Product Overview and Summary

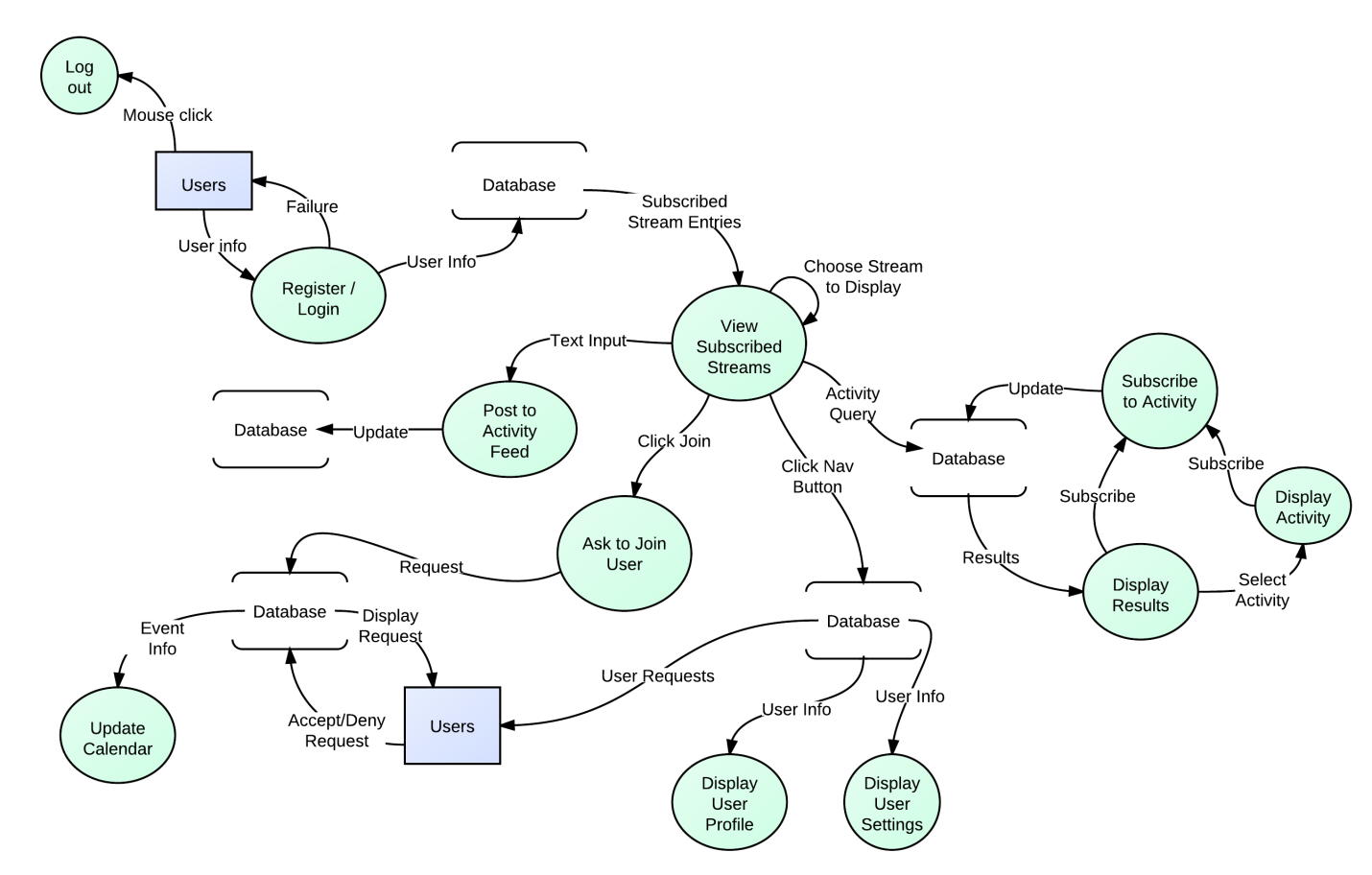
ConnActiv is a social network meant to connect people who participate in physical activities. Its goal is to allow users to meet other people in an area based on a mutual interest in sports, running, hiking, etc. Users will be able to subscribe to different activities based on their interest (e.g. a running “activity”). Each activity feeds into its own public stream containing posts that mention or "tag" the activity, as well as any other relevant updates. The stream is able to be seen in a user's “Home” view if he or she is subscribed to the particular activity. This public stream is intended for users to post what activity they are doing and when. This allows other users to join them if the original user allows invitations to his or her posts. Users will also be able to post to their private stream where only users he or she has connected with may see.

For example, Jon posts the following message on the site’s “Running” section: “Hey, going out for a run in Oakland at 9AM.” If Jon has allowed it, Stacy may ask to join Jon in running that day.

Finally, users may give other users recommendations after they have done an activity together. This gives other users an opportunity to see if the user in question would be a suitable activity partner; however, because this could potentially lead to cyber bullying in the form of malicious reviews users are limited to one review per activity with the person in question.

1. Information Description
   1. User Interface  
      See ‘User Interface’ section in binder.
   2. High-level Data Flow Diagram

See next page.



* 1. Data Structure Representation

**User**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| USER\_ID | INTEGER NOT NULL PK |  |
| FIRST\_NAME | STRING |  |
| STREET | STRING |  |
| CITY | STRING |  |
| STATE | CHAR(2) |  |
| ZIP | INTEGER |  |
| PHONE | CHAR(12) |  |
| INTERESTS | CLOB |  |
| PROFILE\_PIC | INTEGER | --FK TO PICTURES TABLE |

**Networks**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| NETWORK\_ID | INTEGER NOT NULL PK |  |
| AREA | STRING | --CITY,LOCALE |
| ACTIVITY | INTEGER | --FK INTO ACTIVITIES |

**Feed\_Comments**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| NETWORK\_ID | INTEGER NOT NULL |  |
| COMMENT | INTEGER NOT NULL |  |

**User\_Networks**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL |  |
| NETWORK\_ID | INTEGER\_ID |  |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | | Descriptions |
| ACTIVITY\_ID | INTEGER NOT NULL PK |  | |
| ACTIVITY\_NAME | STRING NOT NULL UNIQUE |  | |
| DESCRIPTION | CLOB |  | |

**User\_Activities**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL PK | --USER\_ID and ACTIVITY\_ID = pk |
| ACTIVITY\_ID | INTEGER NOT NULL |  |
| LOW\_LEVEL | INTEGER NOT NULL |  |
| HIGH\_LEVEL | INTEGER NOT NULL |  |

**Favorites**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL |  |
| NETWORK\_ID | INTEGER NOT NULL |  |

**Connactions**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | | Descriptions |
| CONNACTION\_ID | INTEGER NOT NULL PK |  | |
| LOCATION | STRING |  | |
| START\_TIME | DATE |  | |
| END\_TIME | DATE |  | |
| ACTIVITY\_ID | INTEGER FK INTO ACTIVITIES |  | |

**Connaction\_Attending**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| CONNACTION\_ID | INTEGER NOT NULL | --CONNACTION\_ID and USER\_ID = pk |
| USER\_ID | INTEGER NOT NULL |  |

**Reviews**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER |  |
| CONNACTION\_ID | INTEGER UNIQUE FK- ONLY ONE REVIEW PER CONNACTION |  |
| TH\_U/D | BOOLEAN |  |
| REVIEW\_DATE | DATE |  |
| REVIEW | CLOB |  |

**Wall**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL PK UNIQUE |  |
| COMMENT\_ID | INTEGER FK INTO COMMENT TABLE |  |

**Requests**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| FROM\_USER | INTEGER NOT NULL |  |
| TO\_USER | INTEGER NOT NULL |  |
| MESSAGE | CLOB |  |

**Friends**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL |  |
| FRIEND\_ID | INTEGER NOT NULL |  |

**Messages**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| FROM\_USER | INTEGER NOT NULL |  |
| TO\_USER | INTEGER NOT NULL |  |
| SUBJECT | STRING |  |
| BODY | CLOB |  |
| DATE | DATE |  |

**Preferences**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL PK |  |
| SECURITY\_TYPE | INTEGER NOT NULL DEFAULT 0 |  |

**Pictures**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| USER\_ID | INTEGER NOT NULL |  |
| PICTURE\_ID | INTEGER NOT NULL PK |  |
| PICTURE\_LINK | STRING NOT NULL UNIQUE |  |

**Comments**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| COMMENT\_ID | INTEGER NOT NULL PK |  |
| FROM\_USER | INTEGER NOT NULL |  |
| TO\_USER | INTEGER NOT NULL |  |
| COMMENT | CLOB |  |
| COMMENT\_DATE | DATE |  |

**Subscriptions**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| SUBSCRIPTION\_ID | INTEGER NOT NULL PK |  |
| NETWORK\_ID | INTEGER NOT NULL |  |
| USER\_ID | INTEGER NOT NULL |  |

**Events**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| EVENT\_ID | INTEGER NOT NULL |  |
| START\_DATE | DATE |  |
| END\_DATE | DATE |  |
| ACTIVITY\_ID | INTEGER |  |
| LOCATION | STRING |  |
| RECURRENCE | STRING |  |

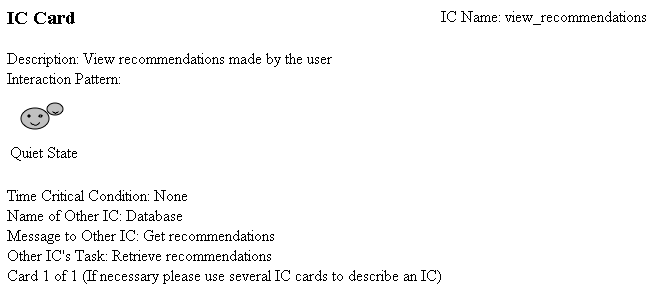
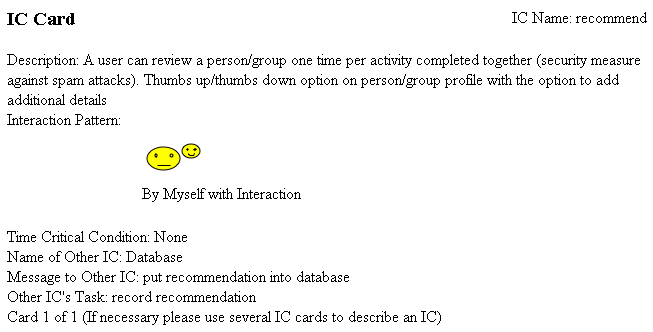
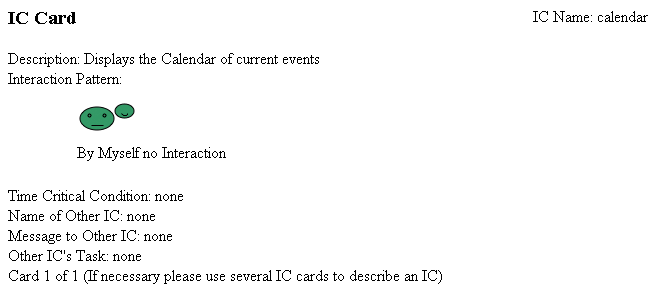
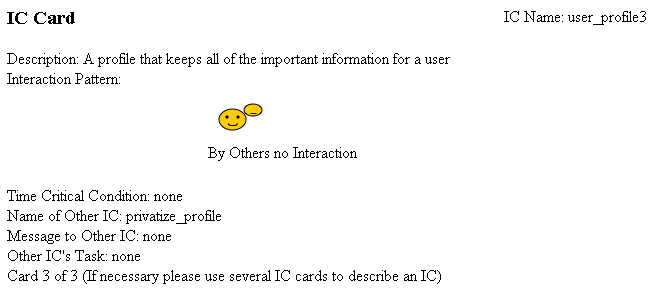
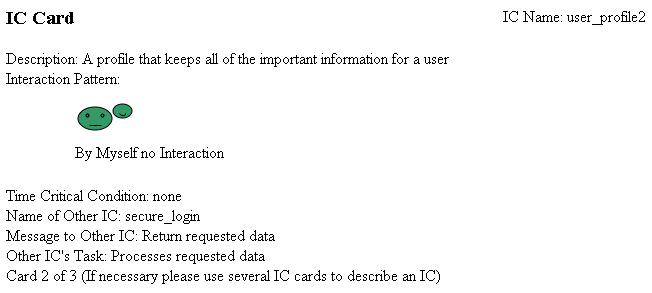
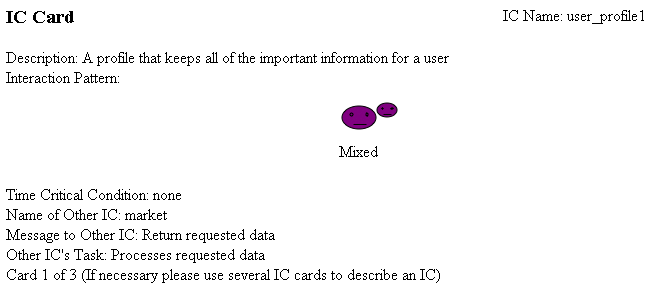
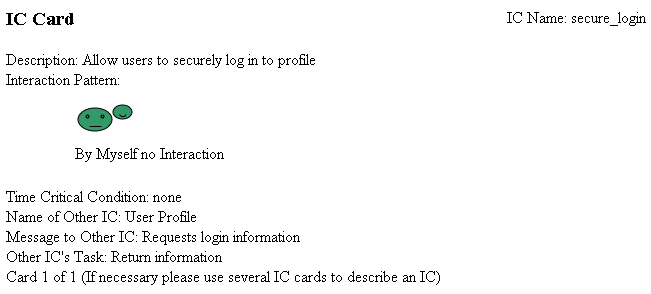
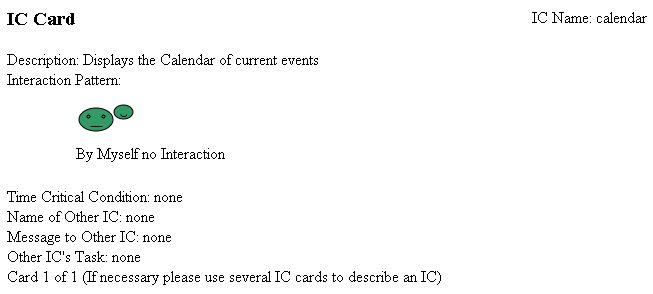
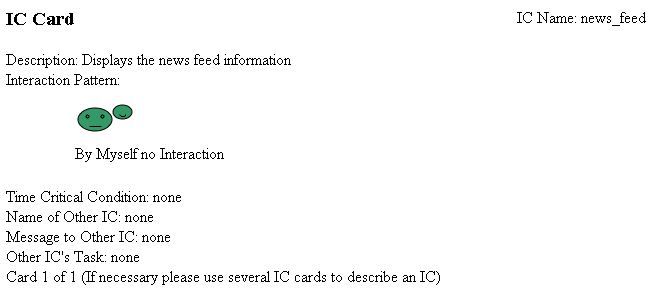
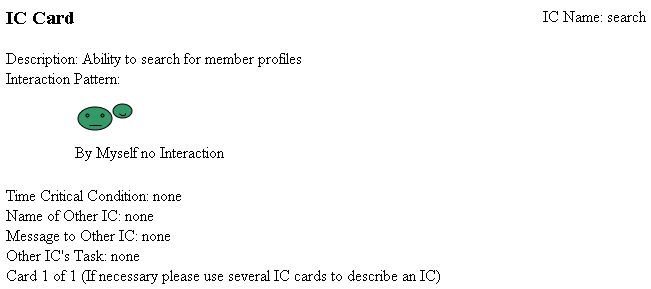
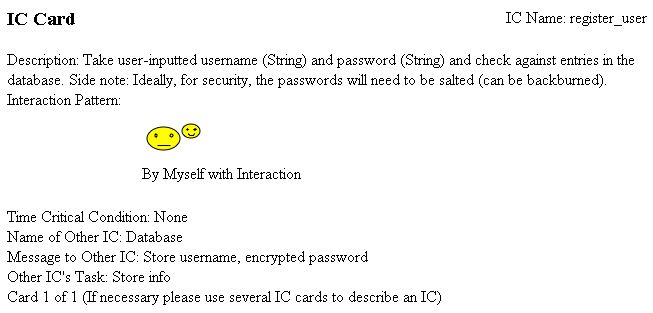
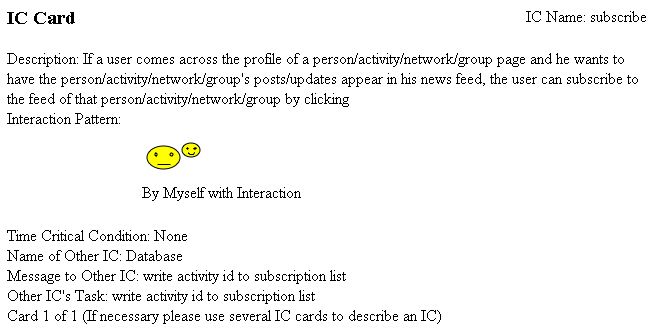
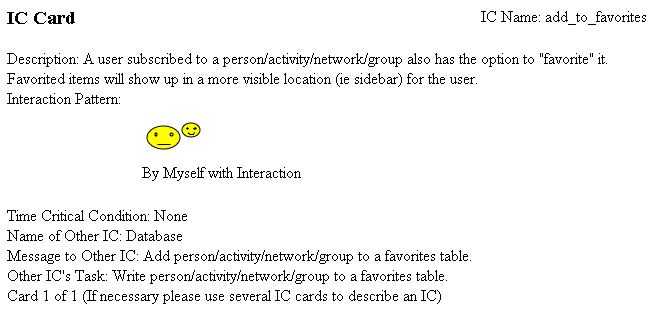
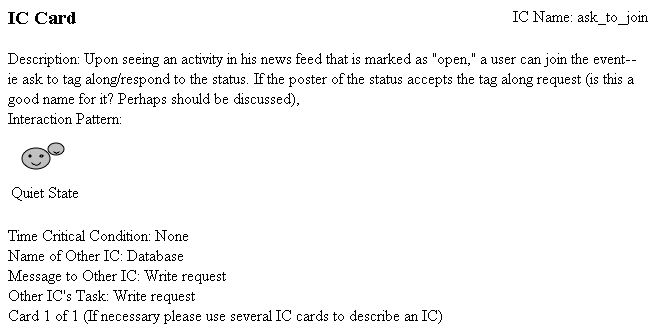
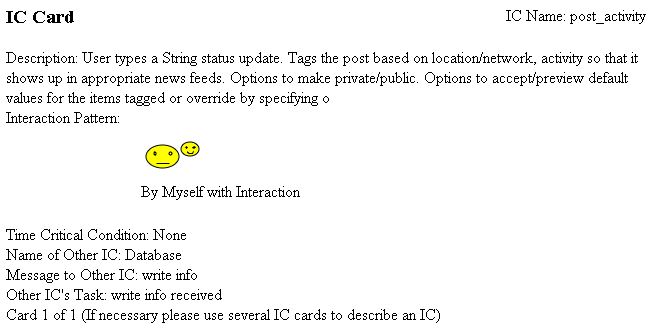
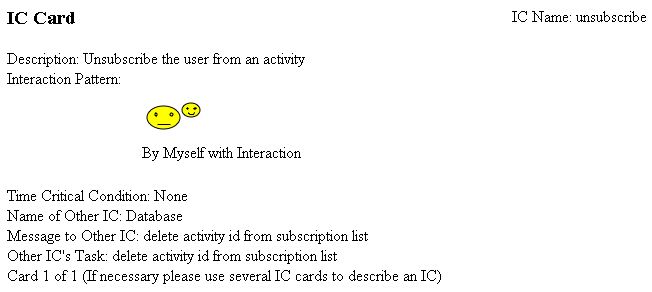
**Event\_Atendees**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Descriptions |
| EVENT\_ID | INTEGER NOT NULL |  |
| USER\_ID | INTEGER NOT NULL |  |

* 1. Data Elements Dictionary

|  |  |
| --- | --- |
| Elements | Description |
| USER\_ID | Unique User ID |
| FIRST\_NAME | User first name |
| LAST\_NAME | User last name |
| STREET | User street address |
| CITY | User city |
| STATE | User state |
| ZIP | User zip |
| PHONE | User phone number |
| INTERESTS | Where the user’s interests will be stored |
| PROFILE\_PIC | Integer into pictures table |
| NETWORK\_ID | Integer uniquely identifying the network |
| AREA | Network area locale |
| ACTIVITY\_ID | ID Number into activity table |
| ACTIVITY\_NAME | Name of an activity |
| DESCRIPTION | Description of the activity |
| LOW\_LEVEL | Low skill level that a user will accept |
| HIGH\_LEVEL | High skill level that a user will accept |
| CONNACTION\_ID | Unique ID number identifying a connaction |
| LOCATION | Location that a connaction will take place |
| START\_TIME | Date/Time that a connaction will start |
| END\_TIME | Date/Time that a connaction will end |
| TH\_U/D | Thumbs up or down on a recommendation |
| REVIEW\_DATE | Date the recommendation was submitted |
| REVIEW | Comments on the recommendations |
| COMMENT\_ID | Unique ID that maps to comments table |
| FROM\_USER | User ID that sent comment/request |
| TO\_USER | User ID that receives comment/request |
| MESSAGE | Message that is included with requests |
| FRIEND\_ID | User ID of a user’s friends |
| SUBJECT | Subject of a message |
| BODY | Body of a message |
| DATE | Date message was sent |
| SECURITY\_TYPE | Level of security a user wishes to have on their profile |
| PICTURE\_ID | Unique ID given to a picture a user uploads |
| PICTURE\_LINK | File path to the picture |
| COMMENT | Text that is the comment a user leaves |
| COMMENT\_DATE | Date a comment was left |
| SUBSCRIPTION\_ID | Unique ID that identifies a user’s subscription |
| RECURRENCE | Date interval an event may recur at |

1. Functional Description
   1. Functions



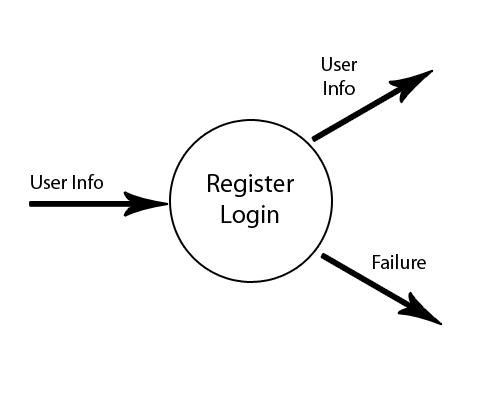
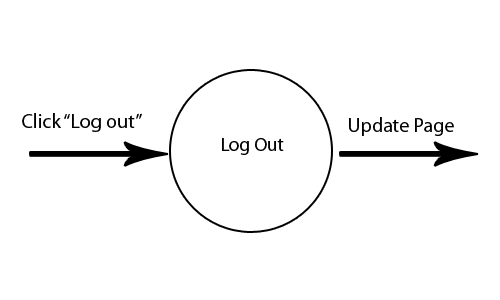
* 1. Processing Narrative

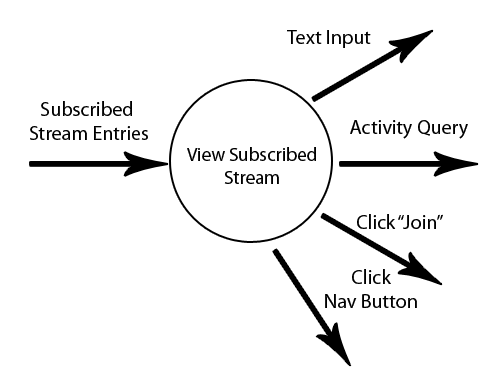
|  |  |
| --- | --- |
| Function Name | Details |
| login | Allow the user to sign into his/her existing account by inputting a string for his/her username and the corresponding password string, hidden as a series of asterisks or dots, and then submitting the information. The user input is passed into the database and checked first to determine if the username is an existing entry (if not, returns false) and then checks if the inputted password matches the stored password for the user. Upon failure, the user is asked to retry his/her credentials. Upon success, the user is logged in and granted access to his/her Home page. |
| logout | Allows the user to exit his/her current session. |
| sign\_up | Allows an individual to register for an account. First, the user submits his/her preferred username (string), which is then sent to the Users table in the database to check for uniqueness. On failure, the user is notified that the username is already taken and asked to select another one. On success, the user is prompted to choose a password that is between six and fifteen characters in length and includes at least one numeric character. The user is also then prompted for additional details:   * *Primary email address* (string) – This will be used to confirm the registration as the final step. * *Real name* (string) - The user can choose to use this as his/her public display name within the site or to use his/her selected username, but the real name must remain in the database as a security measure. The user’s real name is revealed to connactions (those individuals with whom he/she meets up for activities) to add a measure of security. * *Primary network* (string) – The user can select this from dropdown menu of existing networks (populated from the database), with the option to add a network if his/hers is not in the list. The user can choose whether or not to display this information publicly. * *Gender* (character [“m” or “f”]) – The user can choose whether or not to display this information publicly.   Once a user has supplied all the information up to this point, he/she is considered “registered,” although supplementary information is required to make status updates and collect news feeds. The user may complete these details when he/she first registers or may elect to return to fill in supplemental details later, but for ease of use (ie. in order to enable default values to appear in status updates), the user must supply the following for *each activity* he/she wishes to post about:   * *My skill level* *– low* (int) – The lower bound of the range for the user’s best approximation of his/her own abilities for the activity in question. * *My skill level – high* (int) – The upper bound of the range for the user’s best approximation of his/her own abilities for the activity in question. * *Preferred skill level – low* (int) – The lower bound of the range of skill levels the user prefers to interact with. * *Preferred skill level – high* (int) – The upper bound of the range of skill levels the user prefers to interact with. * *Accepted skill level – low* (int) – The lower bound of the range of skill levels the user will accept interacting with. * *Accepted skill level – high* (int) – The upper bound of the range of skill levels the user will accept interacting with.   Once these details are filled in for an activity, when the user goes to create a post involving it, the values for skill levels are autocompleted by default, although the user may elect to override them for any instance. |
| subscribe | Allows a user to receive all updates and activity posts from any person, activity, network, or group he/she chooses. The user then will see these updates in his/her Home news feed. |
| unsubscribe | Allows a user to stop receiving updates and activity posts from any person, activity, network, or group to which he/she was previously subscribed. |
| approve | A user can review a person or group *one time* per activity completed together. (The limit is a security measure against malicious spam attacks.) At a minimum, a user may recommend the connaction in the form of a thumbs up, which then simply increments the corresponding user/group’s recommendations (int) in the database. Thumbs up/down are displayed on the user’s profile directly underneath the profile picture so that it is clearly visible. At a more detailed level, a user wishing to recommend a connaction has the option to provide information beyond a simple thumbs up/down—a user can input a supplemental description or critique that is then attached to the connaction in the database. The reviewing user has the option to post the critique anonymously. |
| disapprove | Similar to the *approve* function, a user may disapprove of the connaction in the form of a thumbs down, which then simply increments the corresponding user/group’s disapprovals (int) in the database. Thumbs up/down are displayed on the user’s profile directly underneath the profile picture so that it is clearly visible. The reviewing user may provide supplemental details/criticisms and he/she has the option to remain anonymous. |
| post\_an\_activity | A user may type an activity or status post (string) to be displayed on his/her profile. The post will also be visible in the news feeds of his/her subscribers. The user may tag the post based on the location or *network* it concerns so that it displays in the appropriate subscriber news feeds. Additionally, the user may *tag* the post based on the activity involved. When he/she tags a particular activity, the skill levels he/she specified during registration for the activity will appear in the status, although these may be overridden for any instance (or edited under Settings). A user also can make the post *private* or *public* so that it is limited to particular audiences (Connactions, Buddies, or Networks).  The user may also specify that the post is a *recurring* activity or even a *lesson*. Finally, a user may mark the post as “open” or “closed.” (Open activity posts mean that the user is looking for or is accepting partners/joiners.)  Example of an activity post:  *Clark Kent: “Anyone want to play squash tomorrow at 9pm at the Pete?”*  *Location: The Pete (Pittsburgh)*  *I am a level 5-6 looking for levels 4-8, accepting levels 4-9.*  *This activity is open!* |
| ask\_to\_join | Upon viewing a desirable activity in his/her news feed that is marked “open,” a user may ask to join the activity by responding to the status. The poster of the status has the option to accept the join request; if he/she does accept, the users then become “connacted” and have “connaction” (more than “limited” but less than “buddy”) access to each other’s profiles and can message one another to discuss and finalize the details of the meet-up. |
| view\_news\_feed | Based on the people/networks/groups to which he/she subscribes, the most recent activity updates are drawn from relevant databases and are displayed in his/her Home section. All subscription updates are grouped into the “All” tab and then are also individually sorted in corresponding tabs. |
| add\_to\_favorites | A user who subscribes to a person/network/group also has the option to favorite it. Favorited items will show up in a more visible location on the user’s Home page and Profile – directly under the user’s profile image and name in the sidebar. |
| view\_calendar | A user can view a calendar that features dated activities for all of his/her subscriptions. It is sortable for each tag so the user may select and deselect what types of events are displays |

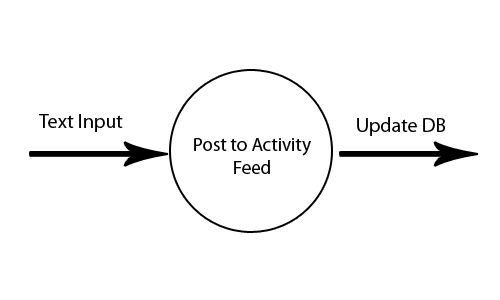
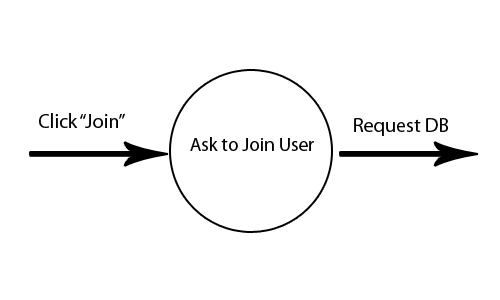
* 1. Design Constraints

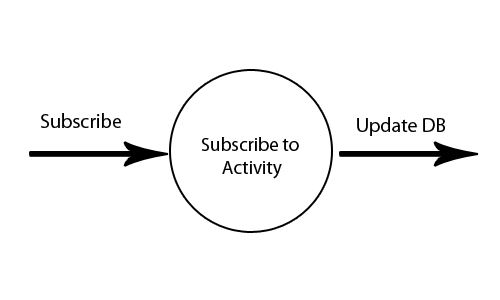
|  |  |
| --- | --- |
| Function Name | Details |
| login | Requires string input for both username and password. Before passing to the database, both inputs are stripped and tested to prevent SQL injections. |
| logout | No design constraints |
| sign\_up | The username and password must be strings, with the password being between six and fifteen characters in length and containing at least one numeric character. Username and password must not contain special characters other than underscores.  Requirements for other necessary details:   * *Primary email address* (string) – Must be valid; must contain @ symbol and ‘.’ symbol. * *Real name* (string) – Must not contain special characters. * *Primary network* (string) – If not already in the database, the user must input his/her preferred network. Must not contain special characters. * *Gender* – Must be a single character [“m” or “f”].   Requirements for other supplemental details:   * *My skill level* – *low (*int) – Must be an integer. * *My skill level – high* (int) – Must be an integer than is >= the lower skill level. * *Preferred skill level – low* (int) – Must be an integer. * *Preferred skill level – high* (int) – Must be an integer than is >= the lower skill level. * *Accepted skill level – low* (int) – Must be an integer. * *Accepted skill level – high* (int) – Must be an integer than is >= the lower skill level. |
| subscribe | A user cannot subscribe to him/herself. A user cannot subscribe to anything to which he/she is already subscribed. |
| unsubscribe | A user cannot unsubscribe from any person/group/network to which he/she is not already subscribed. |
| approve | A user can review a person/group exactly *one time* per activity completed together. They must have participated in an activity together. |
| disapprove | A user can review a person/group exactly *one time* per activity completed together. They must have participated in an activity together. |
| post\_an\_activity | Requires string input |
| ask\_to\_join | Requires that the corresponding activity is “open” to joiners. |
| view\_news\_feed | No design constraints. |
| add\_to\_favorites | Requires that the user be subscribed to the person/network/group he/she wishes to favorite. |
| view\_calendar | No design constraints. |

* 1. Detailed Data Flow Diagrams

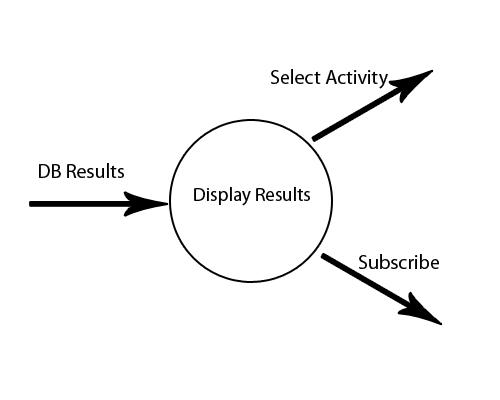
  
Register / Login  
  
Log out

  
View Subscribed Stream

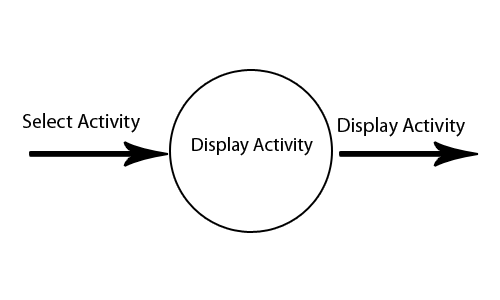
  
Post to Activity Feed  
  
Ask to Join User

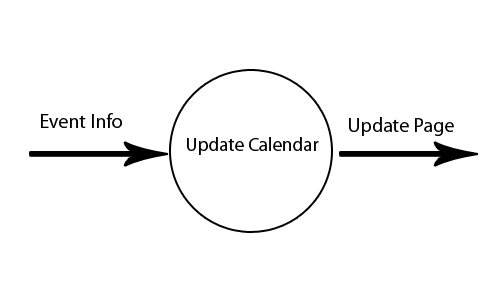


Subscribe to Activity

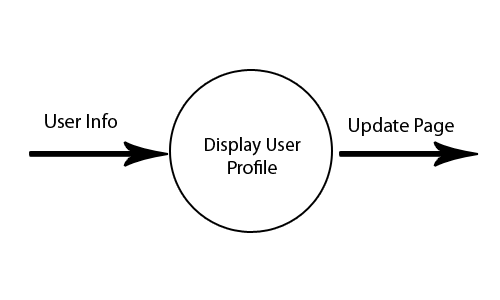


Display Results

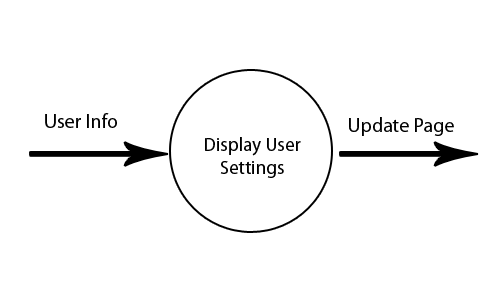
  
Display Activity



Update Calendar



Display User Profile



Display User Settings

1. Performance Requirements

* Considering each user will have many entries in each supporting table, database storage will be our main performance hurdle. The amount of space that we have for database storage will dictate the number of users that our system can handle.
* The processing speed of our server will dictate the number of concurrent users that we can handle at any given time.
* Our system should initially be able to handle up to 1,000 user profiles and subsequent table entries according to the user’s site interaction in the database with up to 100 concurrent users.
* The system should be able to process up to five database queries and display the dynamic page content within five seconds.
* There shouldn’t be any issues with storing dynamic information in the database since inserts are rather inexpensive and should happen one at a time.
* No action or task should take more than five seconds to complete.

1. Exception Conditions

User side:

**Account creation:**

|  |  |
| --- | --- |
| Exception | Handling |
| User enters an existing account name | Inform user the name is taken |
| Password is less than 6 characters | Inform user the password is too short |

**Account logging in:**

|  |  |
| --- | --- |
| Exception | Handling |
| User enters the wrong password | Five chances before a temp lock is issued to prevent account hijacking |
| User cannot remember password | Email is sent to the registered account with a temporary password |

**Activities:**

|  |  |
| --- | --- |
| Exception | Handling |
| User’s wants to list an activity not supported | User selects other and fills in a field  This can be later added to our system |

Server side:

All system handle exceptions will comply with W3’s RFC2616 HTTP status code standards. This includes returning a 500 error to the client when a system encounters an undefined exception.

1. Implementation Priorities

Implementation priorities will follow the schedule outlined in the software plan. The implementation methodology will follow the waterfall model which will allow for a systematic approach to materializing the designs. In each step, the following step is carefully planned.

Initially, the team will implement the user interface using HTML and CSS followed by the database communication and interaction mechanisms by using PHP for the scripting language and MySQL for databases. If time permits, we will implement an Android application for mobile phones. This will be done as a separate interface after the core of the project is functional.

* Outline of the implementation priorities:
* User Interface Prototype
* User Interface Core Functionality
* PHP Scripts
* Android Application, if time permits
* Deployment

1. Foreseeable Modifications and Enhancement

**Modifications/Enhancements – User side**

* **Increase Activities**
  + Activities are currently limited to physical (sport-like) activities, but this could spread to anything.
* **Add Medal Achievements**
  + As users do more activities, they will start to unlock different medals (Gold, Silver, and Bronze).

**Modifications/Enhancements – System side**

* **Advanced Code: Make site even easier to navigate**
  + Auto complete fields.
  + AJAX – Convert functions to asynchronous ones.
* **Optimize Code: If time permits, ensure code is as efficient as possible**
  + MySQL, PHP, CSS, jQuery, HTML.

1. Acceptance Criteria

Nightly releases of the software will be verified using unit testing or a similar automated test suite.

List of user interface acceptance tests that will be performed:

* End-user usability surveys with feedback forms
* Usability heuristics tests
* Usability specification compliance
* Unit tests
* Compliance to this specification document

List of the integrity tests that will be performed:

* Unit tests
* Compliance to this specification document

1. Sources of Information

* GitHub [ <http://github.com> ] – The manual pages located on GitHub for our version control system
* Android SDK [ <http://developer.android.com> ] – Android SDK information for the Android app
* PHP Manual [ <http://php.net/manual/en/index.php> ] – PHP manual pages which will be reference during PHP development
* MySQL Manual [ <http://php.net/manual/en/book.mysql.php> ] – MySQL manual which will be referenced for database interaction
* JSON Manual [ <http://php.net/manual/en/book.json.php> ] – JSON manual which will be referenced for server-client database communication
* PHPMyAdmin [ <http://phpmyadmin.net/home_page/index.php> ] – Information about the database management system we will be using to create initial database requirements

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Person | Date Modified | Section | Description |
| Vince | 9/22 | 1 | First draft |
| Rob | 10/3 | 2.1 | First draft |
| Ray | 10/3 | 3.1 | First draft |
| Dave | 10/5 | 2.3,2.4 | First draft |
| Vince | 10/5 | 2.1 | Added Calendar and proofread |
| Kim | 10/5 | 3.2,3.3 | First draft |
| Vince | 10/6 | Entire Doc | Assembled all pieces |
| Ray | 10/18 | 6, 8 | First draft |
| Dave | 10/18 | 5, 9 | First draft |
| Vince | 10/19 | 3.4 | First draft |
| Dave | 10/19 | 4, 9 | First draft |
| Vince | 10/19 | Entire Doc | Assembled and proofread all pieces |