US02.01.01

As an owner, I want to track people I know. Adding a textual username should be enough.

Use Case 2.1:

This use case describes how the user can add textual usernames to track people.

Actors: Owner(user), App Database

Preconditions: There is an active network connection with the database, User is using the app, User is authenticated

Flow of events:

1) User opens up the app

2) User goes to the friends screen

3) User adds another user by username to tracking list

4) App goes through database to check that it is a valid user

5) The user exists

6) The user now has the other user on the tracking list

Alternate Flows:

* If in step 4 the user doesn’t exist:

4.1) The user doesn’t exist

7.1) App prevents user from adding the other user to the tracking list

8.1) App prompts user of invalid username

9.1) App goes back to …screen

US02.02.01

As an owner, I want to add friends

Use Case 2.2:

This use case describes how the user can add friends through their username

Actors: Owner(user), Other Owner/user, App Database

Preconditions: There is an active network connection with the database, User is using the app, User is authenticated

Flow of events:

1) User opens up the app

2) User goes to friends screen

3) User adds another user by username

4) App goes through database to check that it is a valid user

5) The user exists

6) The request is then sent to the other user for approval

7) The request is approved by the other user

8) Both users are now friends

Alternate Flows:

* If in step 4 the user doesn’t exist:

4.1) The user doesn’t exist

7.1) App prevents user from adding the other user to the tracking list

8.1) App prompts user of invalid username

9.1) App goes back to friends screen

* If in step 7 the user doing the adding cancels the request

7.2) The request is then cancelled by sending user

8.2) The request is then deleted, preventing the other user from responding to it

* If in step 7 the user doing the adding cancels the request

7.3) The request is denied by the other user

8.3) The request is then deleted, and the adding user gets notified of the denial

US02.03.01

As an owner, I want to remove friends

Use Case 2.3:

This use case describes how the user can delete friends through their username/friend list

Actors: Owner(user), Other Owner/user, App Database

Preconditions: There is an active network connection with the database, User is using the app, User is authenticated. Users must be friends.

Flow of events:

1) User opens up the app

2) User goes to friends screen

3) User inputs the username to delete

4) App goes through database to check that it is a valid user and that they’re currently friends

5) The user exists

6) The database then removes that friendship

7) Both users are no longer friends with each other

Alternate Flows:

* If in step 4 the user doesn’t exist:

4.1) The user doesn’t exist

7.1) App prevents user from adding the other user to the tracking list

8.1) App prompts user of invalid username

9.1) App goes back to the friends screen

* If in step 7 the user doing the adding cancels the request

7.2) The request is then cancelled by sending user

8.2) The request is then deleted, preventing the other user from responding to it

* If in step 2 the user goes to friend list instead

2.3) User goes to their friends list screen

3.3) User holds and selects delete on the user

4.3) The database then removes that friendship

5.3) Both users are no longer friends with each other

US02.04.01

As an owner or borrower, I will have a profile where by my contact information and city are recorded.

Use Case #2.4:

This use case describes how the user can record contact information and city in their profile

Actors: Owner/Borrower(user), App Database

Preconditions: System has access to network

Flow of events:

1) User clicks on Register button on app’s welcome screen

2) System displays RegisterActivity

3) User fills in required EditText fields for profile and clicks on SignUp button

4) System validates user’s entries

5) System checks for network availability

6) System checks server to make sure username is unique

7) System adds user to server (profile is created) and displays the user’s AllActivity

8) User clicks on Settings icon

9) System displays Settings Activity

10) User clicks on Edit Profile button

11) System displays UserProfileActivity with user’s profile

Exceptions:

4) User’s entries are not valid

4.1) System toasts “Form contains error” and displays error message on corresponding EditText Field

5) System determines that no network is available

5.1) System toasts no network available message (no registration performed)

6) System determines that user-entered username is not unique

6.1) System toasts “Form contains error” and displays error message “Username already taken”

US02.05.01

As an owner or borrower, I will be able to view the profile of anyone I know of including friends.

Use Case 2.5:

This use case describes how the user can view friend’s profiles

Actors: Owner/borrower(user), Other user, App Database

Preconditions: There is an active network connection with the database, User is using the app, User is authenticated, other user exists

Flow of events:

1. User navigates to Friends tab in AllActivity
2. System displays Friends tab in AllActivity
3. User clicks on Add Friend button
4. System prompts user to enter any username
5. User enters username
6. System checks server to see if the user exists
7. System adds found username to user’s friend list and displays friend in ListView
8. User clicks on friend’s username in ListView
9. System retrieve’s friend’s user object from server and saves it in the cache
10. System displays friend’s inventory (InventoryActivity) from the cache
11. User clicks on User Profile button
12. System displays friend’s user profile (UserProfileActivity)

Exceptions:

6) System determines that no network is available

6.1) System toasts no network available message

7) System cannot find username on server

7.1) System toasts “User not found”