## List Ninja Project Documentation

**Problem Statement**

Do you have a lot of roommates? Do you have problems keeping track of expenses between your roommates? We’ve all been there, trying to keep track of expenses between housemates or any group of people who share common expenses. It is hard to keep a common list of items between groups of people in which everyone is responsible for buying items on the list. Some things may get bought twice or some items may not get bought at all. And then there is the awkward task of divvying up what who owes who.

**Objectives**

1. Create individual and household accounts.
2. Create lists to track expenses.
3. Share bills, expenses, and list items.
4. Split expenses
5. Calculate money owed to individuals based money spent by each individual.

**Functional Requirements**

* Users must be able to signup
* Users must be able to login
* Users must be able to create a new list
* Users must be able to share the list with other users
* Users must be able to add friends (other users) in the app
* Users must be able to import Facebook friends
* Users must be allowed to add a price and quantity for an item
* Users must be able to view a cost distribution between list members
* Users must be able to create an infinite number of lists
* User must be able to prioritize or close lists

**Non-Functional Requirements**

* App will be built on the java web framework
* App will use phone gap to build mobile app version
* App will use bootstrap to look nice
* App will use facebook API to grab user information + photo

**Use Cases**

# Use Case Description

|  |  |
| --- | --- |
| **Use Case name:** | Login |
| **Product name:** | ListNinja |
| **Team:** | Robert Buser, Avi Dey, Gianna Fusaro, Andrew Sheffield |
| **Date:** | September 15, 2014 |

|  |
| --- |
| 1. Goal User logs into ListNina |
|  |

|  |
| --- |
| 2. Summary A registered ListNinja user enters credentials to access their account on the site |
|  |

|  |
| --- |
| 3. Actors |
| Actor 1: User |

|  |
| --- |
| 4. Preconditions |
| * The user has already signed up with ListNinja * The user is on the homepage   *.* |

|  |
| --- |
| 5. Trigger |
| The user clicks the “login” button |

|  |  |
| --- | --- |
| 6. Primary Sequence |  |
| **Step** | **Action** |
| 1 | User enters email and password |
| 2 | User clicks “login” |
| 3 | System validates user credentials |
| 4*.* | System redirects customer to their dashboard |

|  |
| --- |
| 7. Primary Postconditions |
| * User is logged into the site and can access site and all information associated with their account * User is not logged in and remains on the homepage |

|  |  |
| --- | --- |
| 8. Alternate Sequences |  |
| **Alternate Trigger** |  |
|  | User does not enter an email or a password or both |
| **Step** | **Action** |
| 1 | System displays error message to user |
| Alternate Postconditions |  |
|  | User is not logged into site |
| **Alternate Trigger** |  |
|  | User enters invalid email or password or both |
| **Step** | **Action** |
| 1 | System displays error message to user |
| Alternate Postconditions |  |
|  | User is not logged into site |

# Use Case Description

|  |  |
| --- | --- |
| **Use Case name:** | Create List |
| **Product name:** | ListNinja |
| **Team:** | Robert Buser, Avi Dey, Gianna Fusaro, Andrew Sheffield |
| **Date:** | September 15, 2014 |

|  |
| --- |
| 1. Goal |
| User creates a new list |

|  |
| --- |
| 2. Summary |
| A logged-in user clicks “create new list” button to create a new list |

|  |
| --- |
| 3. Actors |
| Actor 1: User |

|  |
| --- |
| 4. Preconditions |
| * User must be logged in * User is on a place in the site where “create new list” button is viewable |

|  |
| --- |
| 5. Trigger |
| The user clicks “create new list” button |

|  |  |
| --- | --- |
| 6. Primary Sequence |  |
| **Step** | **Action** |
| 1 | User clicks “create new list” |
| 2 | System redirects user to new list form |
| 3 | User edits blank list form |
| 4 | User clicks “save changes” |
| 5 | System saves list |

|  |
| --- |
| 7. Primary Postconditions |
| * User is able to edit new list form * User is able to save new list after editing it |

|  |  |
| --- | --- |
| 8. Alternate Sequences |  |
| **Alternate Trigger** |  |
|  | User does not enter any information in new created list |
| **Step** | **Action** |
| 1 | System displays error message to user |
| 2 |  |
|  |  |
| Alternate Postconditions |  |
|  | User does not create new list |

|  |
| --- |
| 9. Nonfunctional Requirements |
| * System highlights invalid text fields with red |

|  |
| --- |
| 10. Glossary |
| User = person who is logged into ListNinja |

|  |
| --- |
| 9. Nonfunctional Requirements |
| * System will highlight invalid text fields in red |

|  |
| --- |
| 10. Glossary |
| User = person who has signed up for ListNinja or is interested in signing up |

# Use Case Description

|  |  |
| --- | --- |
| **Use Case name:** | Add an item to a list |
| **Product name:** | List Ninja |
| **Team:** | Robert Buser, Avi Dey, Gianna Fusaro, Andrew Sheffield |
| **Date:** | September 15, 2014 |

|  |
| --- |
| 1. Goal User adds an item to a list |
|  |

|  |
| --- |
| 2. Summary A registered ListNinja user adds an item to an already created list. |
|  |

|  |
| --- |
| 3. Actors |
| Actor 1: User |

|  |
| --- |
| 4. Preconditions |
| * The user must already be logged in * A list that the user is adding an item too must already exist * The user must already be viewing the list and see the ‘add an item’ button |

|  |
| --- |
| 5. Trigger |
| The user clicks “Add an Item” button |

|  |  |
| --- | --- |
| 6. Primary Sequence |  |
| **Step** | **Action** |
| 1 | User clicks “Add an Item” |
| 2 | System redirects user to Add an Item Form |
| 3 | User fills out “Add an Item” Form |
| 4 | User clicks “Add” button at the bottom of the form to confirm the submission. |
| 5 | System saves the item into the list |

|  |
| --- |
| 7. Primary Postconditions |
| * User is able to edit the item that was added into the list * User is able to save new item after editing it |

|  |  |
| --- | --- |
| 8. Alternate Sequences |  |
| **Alternate Trigger** |  |
|  | User does not enter all necessary information in add new item to list |
| **Step** | **Action** |
| 1 | System displays error message to user |
| 2 | System brings user back to the main view of the list |
| Alternate Postconditions |  |
|  | User does not add a new item to the list |

|  |
| --- |
| 9. Nonfunctional Requirements |
| * System will highlight any invalid fields when adding information about the item to be added |

|  |
| --- |
| 10. Glossary User = person who has signed up for ListNinja or is interested in signing up |
|  |

**Conceptual Design**

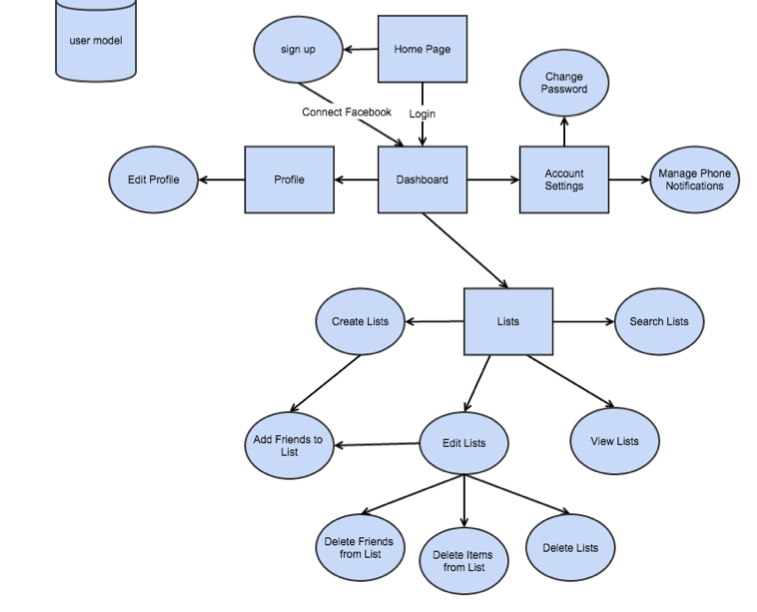
List Of Major Features

* Easily keep track of multiple lists with different people
* Keep track of who is purchasing what items
* Manage expenses within the group

Major Modules

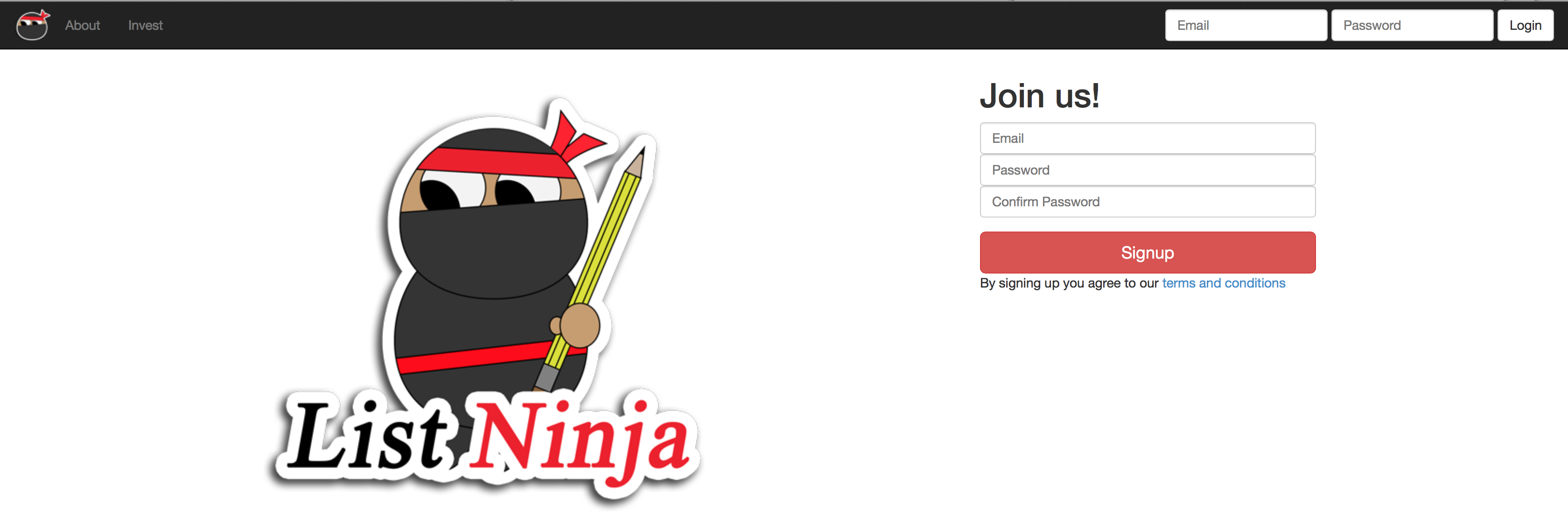
* Login to ListNinja using Facebook instead of having to create a brand new account
* Be able to manage a list and share it with other people
  + Be able to change the name of a list
  + Be able to change who is able to see and add items to the list
* Keep track of who is purchasing what items in a list
* Be a multiplatform application that talks to the same database
  + It is important for the app to me mobile so you can take lists of items with you to the grocery store

**Target**

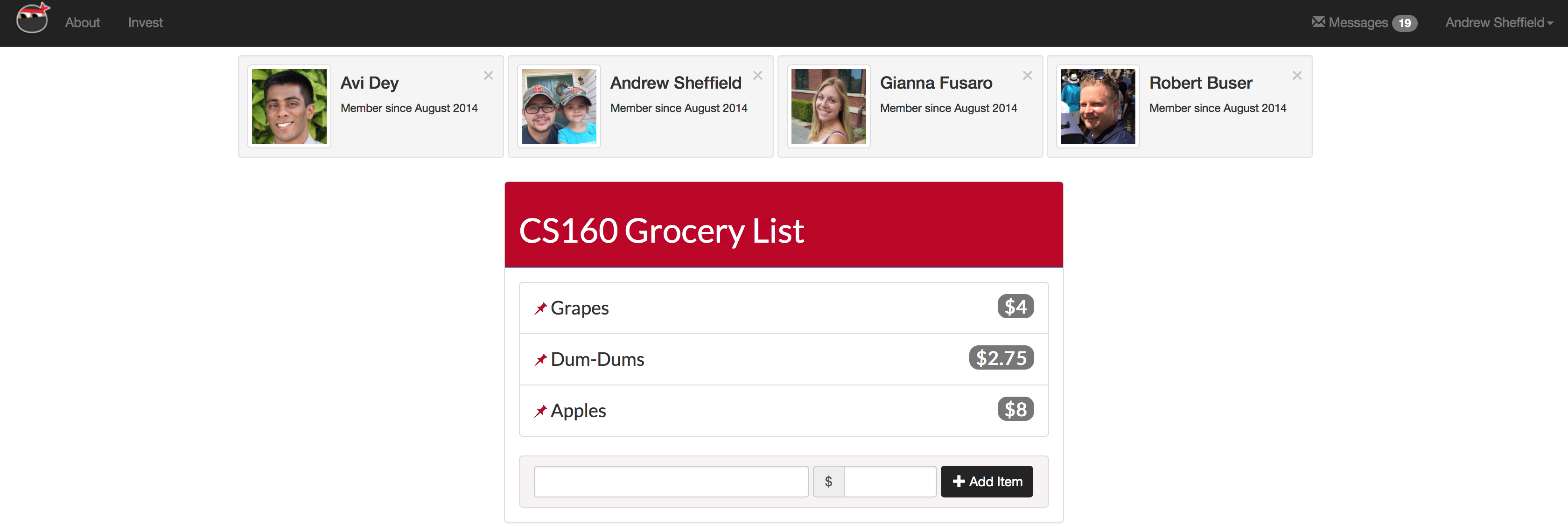
High Level Architecture Diagram – Also on Keynote/PDF presentation ****

**Screenshots**

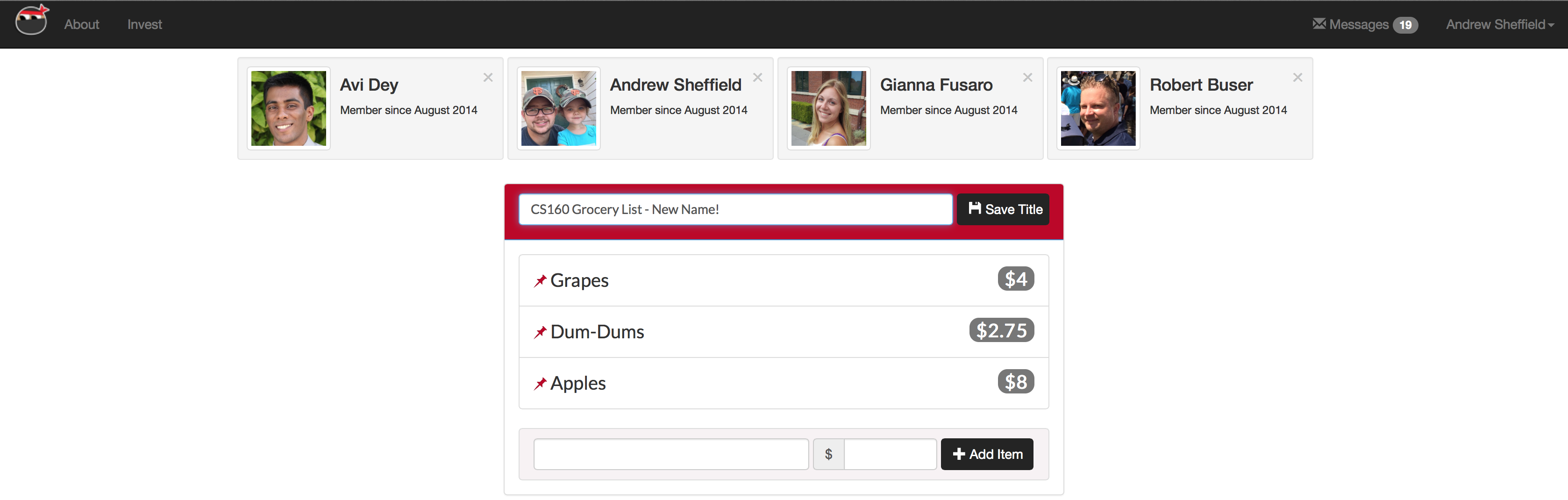
**LogIn Screen**

****

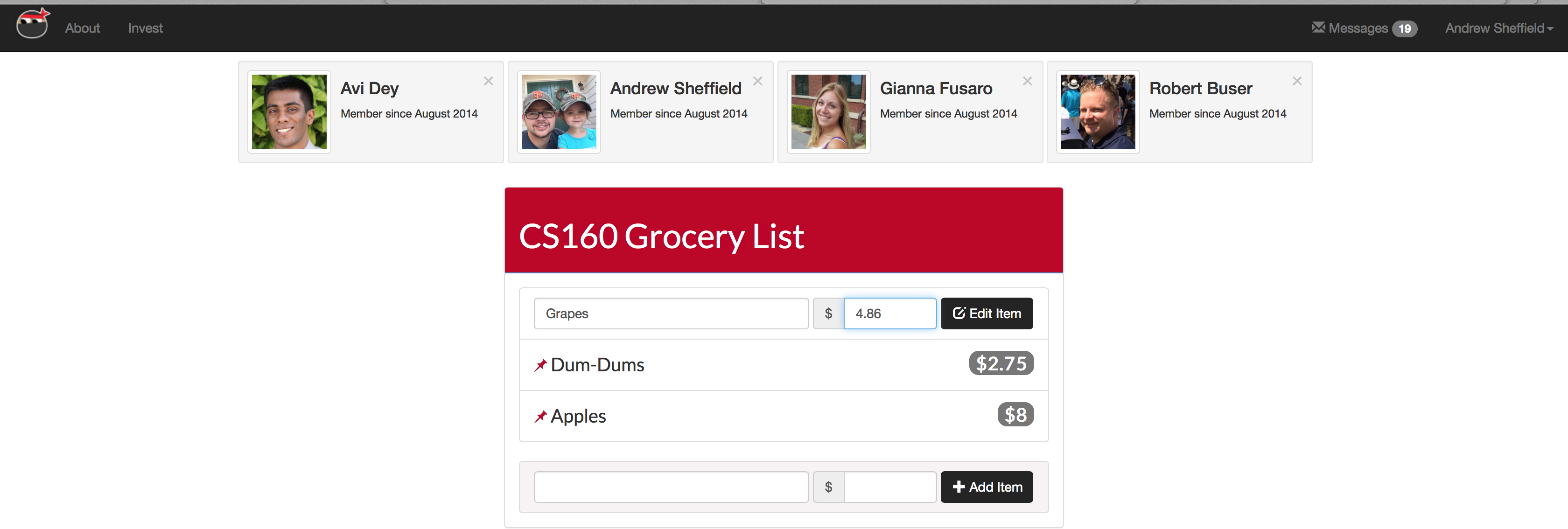
**Homescreen – Most Recent List**

****

**Editing a List Name**

****

**Editing an item**

****