RooMate
Requirements and Specification Document
2016-09-27, version 1.0

Project Abstract

For decades, apartment and household residents have struggled to efficiently and equally distribute chores amongst themselves. Common recurring tasks like buying communal goods (soap, toilet paper, etc.), organizing household meetings, or equally distributing chores could be greatly simplified through the use of a mobile app. Our application, RooMate, will simplify the daily complexities of having roommates. The app makes it easier to see the distribution of chore labor amongst roommates. The app also makes communal expenses easier to split up and keep track of. Communicating with one another will be simple, and events will be easier to organize.

Team Members

SeongJae Kim (skim387@wisc.edu)
Hunter Koeshall (hkoeshall@wisc.edu)
Aaron Levin (awlevin2@wisc.edu)
Corey Allen Pett (cpett@wisc.edu)
Ritvik Upadhyaya (rupadhyaya2@wisc.edu)
Zhanpeng Zeng (zzeng38@wisc.edu)

Document Revision History

Rev. 1.0: 2016-09-27: initial version

Customer

Our customer is any person that lives with roommates and needs a better system to organize chores, pay communal bills, and communicate better with one another. The app is ideal for college students, most of which are experiencing communal living for the first time, but the app is also applicable to anyone who lives with roommates. Some features of the app (like organizing chores and bulletin boards) can even be used by families with young children. Users will need smartphones with internet connection to use the app/service, and although iOS will be the first platform to support the app, an Android version will later be released after the proof of concept is demonstrated.

Our primary customer is Tommy Wolfe, a current UW-Madison student who lived communally for an entire year with notably unfair distribution of chores amongst roommates. Tommy and his roommates have agreed to test the app and provide feedback. All of the people in their household are not particularly tech savvy, so their testing will provide strong evidence for how simple/seamless the app is to use. Several other households have agreed to help test the app and provide feedback, but the perpetual issues in Tommy's household will make them a good indicator of the app's effectiveness.

Competitive Landscape

Our application is trying to simplify the complex management of a life shared with roommates. There are currently applications available on iOS App Store that do similar things like splitting up chores and bills, or sharing a communal shopping list or message board. However, after some research, we found several application only have a subset of features that we try to implement in our project, and others are not designed for the specific purpose of managing a shared household among roommates. One application has a similar user interface as our project, but we noticed some apparent weaknesses within this application. In addition to these weaknesses we also feel the application is a bit outdated. These competitor applications will hopefully be useful in getting some real world user feedback of some of our desired features.

1. **HomeSlice**
   a. **Pro**:
      i. Integrates several features for managing life sharing with roommates.
      ii. Whiteboard provides a space for communication among roommates.
      iii. Has a shared shopping list for common household supplies or groceries, and the ability to keep track of shared bills.
      iv. Simple way to mark household supplies "in stock" or "out of stock".
      v. Ability to create chores schedule, and know who has done their chores.
      vi. Keeps track of all shared bills and payments among roommates.
      vii. Synchronized among devices.
   b. **Con**:
      i. Not compatible with iOS and most newer Android devices.
      ii. Difficult to use.
      iii. Doesn't have push notifications.
      iv. Doesn't allow users to mark completed tasks.
      v. Has several bugs and needs improvement.
      vi. Doesn't have statistics about how much work each roommates has done.
      vii. Doesn't have Venmo/PayPal integration.
2. **Chorma for Chores**
   a. **Pro**:
      i. Uses rewards to motivate users into doing chores.
      ii. Simple ways to add, view, modify and delete chores.
      iii. Synchronized among devices.
      iv. Ability to view the history of who has done what chores.
      v. Can create unclaimed chores for others to do.

        vi. Schedule repeated chores and rotating chore schedules.
    b. **Con**:
        i. Cannot easily reschedule the chores if completed after due date.
        ii. Difficult to edit existing chores.
        iii. App is only for chore management.

3. **Splitwise**
    a. **Pro**:
        i. Easy to use.
        ii. Keeps track of shared bills and history.
        iii. Sums up several small payments to larger transactions.
        iv. Categorizes the bills.
        v. Notifications for bills due and bill updates.
        vi. Integrates Paypal and Venmo.
    b. **Con**:
        i. Has only a bill splitting function, which is good since it has larger user group, but bad since only has limited features for managing life on a shared household.
4. **Our Groceries Shopping List**
    a. **Pro**:
        i. Keeps track of a shared shopping list, and can create several different lists for different stores.
        ii. Categorizes different items for better view and management.
        iii. Word recommender when entering the name of an item.
        iv. Can add a photo or barcode when adding an item so that the roommates can get the right item.
        v. Has a lot of customized settings for enhanced user experience.
        vi. Implements on different platforms.
    b. **Con**:
        i. Has only limited functions for managing shared household items.
        ii. Too many customized settings, which can make it difficult to use.
        iii. Users need a shared email to synchronize among devices.
5. **OurHome**
    a. **Pro**:
        i. Can assign and schedule chores to specific users within the group.
        ii. Ability to add weighted point values to each chore.
        iii. Can view the chores history to see how much work each person has done for each type of chore.
        iv. Has a shared calendar to post events.
        v. Has a shared groceries list.

b. **Con**:
   i. User interface was designed for family use and motivating kids to do chores, not for roommates.
   ii. Doesn't have a feature to keep track of groceries bills.

User Requirements

The mobile application provides an all in one roommate experience to satisfy various household needs. The main idea to the application is to bring chores, groceries, payments, and a way to notify all the roommates about different activities going on in the house at one place. The features of the mobile application are listed below in order from when the user opens the app and touches each tab in the tab bar presented.

1. User opens the application

A. The application loads and displays our RooMate launch screen. After the presentation of the launch screen, our login screen is displayed. The user will need a Facebook account, which then the user will have to click the FB button to login.
   I. If the user opened the application for the first time - The user is presented with the onboarding process where if the user opens a tab for the first time, a small animated run through of the how that specific tab works is displayed.
   II. Else - the first tab that is launched is the bulletin board tab with no additional information.

2. Bulletin board tab viewed

A. This tab will show post-it notes that any roommate can create to be able to notify other roommates about miscellaneous things such as memes, reminders, and random fun notes.
B. Another feature on the billboard is the recent activity view. This will show recent completed activities such as completed payments or completed chores. The user can swipe or tap the recent activity view to access the past 30 recent events.

3. Chore tab viewed

A. This tab will show the list of chores that need to be completed.
B. There will be an add chore bar button item that will let you add a chore.
   a. The user can add a predetermined chore to the chore list that we as developers will think of such as taking of the trash, cleaning dishes, etc. (the essentials)

    b. The user can also add a custom chore that will be added to the predetermined chore list so the user does not have to create the same chore again.

C. Another bar button item will be used on the left side of the screen for statistics on chore completion
    a. At the top of the statistics view, the user will be presented at the top of the view with personal statistics on what the current user has completed. Such as 3x for the cleaning dishes and 1x for taking out the trash.
    b. Below the personal statistics view, there will be a bar graph that shows how each roommate stacks up in the amount of chores completed. So that everybody in the house knows is contributing a fair amount of work.

4. Groceries tab viewed

A. This tab will show two segmented tabs at the top of the view and the first segment is the user's personal grocery list. This will allow every roommate in the household to create a personal grocery list.
    a. The user can add a grocery item and set whether it will be in aggregate list can add additional notes to the item such as get "so and so brand."

B. From every roommate's personal grocery list, there will be another segment called Communal list. This allow every roommate to add communal items (such as toilet paper) to a shared list.

C. There will be a aggregate segment (name needs to be simpler for the dummies such "Household"). This will show every roommates grocery list and communal list concatenated into one table view. If two people used the word eggs, then we will show in the concatenated list that the household needs two items of eggs.

D. The user who is going to buy groceries will press a button that means the user is going grocery shopping for the household and that will delete a subset of the grocery list that the user already bought and split the costs among everybody's personal list. This will also send a push notification to everybody.

5. Finance tab viewed

A. This tab will also contain a segmented control. The first segment will contain the user's debt. The total of how much the user owes, with a list of everybody who the users owes with the individual debts listed.

B. The second segment will provide the user with everybody who owes the user money. There will be a remind an individual user about a payment button, which will send a push notification to that specific user. There will also be a "REMIND ALL" button to send a push notification to everybody. We will handle spam accordingly.

6. Settings tab viewed

A. Customizable UI, such as changing background of the bulletin board.
B. Change when automatic notifications are sent
C. Log in using facebook account
D. Create a group if the user is not in a group
E. Invite other roommates to your household
F. Leave a group

7. Notifications

A. If there are payments that are due at end of the month, a notification is sent automatically to everybody who owes money.
B. Each user that isn't spamming reminders, will be able to send a friendly reminder to another user who owes money.
C. When a user clicks the button that states that the user is going to buy groceries, this will send a notification to every roommate to let them know that they are going shopping. This will help other roommates think if they need to change their grocery list such as removing or adding an item.

Use Cases

Name	**Open the app - Log in**
Actor	User
Trigger	Open the app when no account is logged in
Events	-user presses "LOGIN" to log in with facebook account -app verifies account via Facebook -user is directed to the Bulletin Board page -user presses "EXIT" to close app
Exit Condition	User presses "LOGIN" or "EXIT"
Post Condition	User gets access to his/her own stored data and setting.
Acceptance Test	User should be able to load personal settings after login.

Name	**Settings - Create a group**
Actor	User

Trigger	Press “Create a group” in settings view
Events	-user presses “Create a group” -user is directed to a view for creating a group -user fills in the group’s information -user submits the request by pressing “Create”
Exit Condition	User presses “Create” or “Cancel”
Post Condition	A group is created and user is added in the group. User is directed to setting view. The “Create a group” button changes to “Leave my group.”
Acceptance Test	After creating a group, the user should be able to see the group’s information in the setting view.

Name	**Settings - Invite another user to join the group**
Actor	User
Trigger	Press “Add roommate” in the settings view
Events	-user presses “add roommate” -user is directed to a view for searching people -user fills in the account or the name of the person they want to add -a list of possible people is displayed -user select one person and presses “Invite” -a notification is sent to the person
Exit Condition	Press “Invite” or “Cancel”
Post Condition	User is directed back to the settings view. An error message is displayed if the invited person is already in a group.
Acceptance Test	The invited user will receive a request to join this specific group. The user will see an error message is the invited person is already in a group.

Name	**Setting - Leave the group**
Actor	User
Trigger	Press “Leave the group”

Events	-user presses the “Leave this group” button -the “Leave this group” button changes to “Confirm” -user presses the “Confirm” button -a notification is sent to all members within the group
Exit Condition	Press “Confirm” or press anywhere else on the screen to cancel
Post Condition	User will no longer belong to this group. The “Leave this group” button then change to “Create a group”.
Acceptance Test	After this action, user should no longer see the group information, and all group functions within the app will be disabled.

Name	**Setting - Accept an invitation**
Actor	User
Trigger	A request is sent to the user
Events	-user presses “Accept” or “Decline”
Exit Condition	Press “Accept” or “Decline”
Post Condition	User is admitted into the group if the invite is accepted. The “Create a group” button changes to “Leave this group”. No change if the user presses “decline”.
Acceptance Test	After this action, the user should be able to see the group information, and all group app functions are enabled.

Name	**Chores - view chores statistics**
Actor	User
Trigger	Press “Group statistics” button in chore view
Events	-user presses “Group statistics” -user is directed to the statistics view -user presses “Return” and is directed to the chores view
Exit Condition	Press “Return”
Post Condition	User is directed to the chores view
Acceptance Test	User can go to statistics view by pressing “Group statistics”.

	User can return to the chores view by pressing “Return”.

Name	**Statistics view - view the completion history**
Actor	User
Trigger	Press “History” button within the statistics view
Events	-user presses the “History” button -user is directed to a view for chore history -user presses “Return”
Exit Condition	Press “Return”
Post Condition	User is directed to the statistics view.
Acceptance Test	User can navigate to the history view by pressing “History”. User can return to stats view by pressing “Return”.

Name	**Chores - add a chore**
Actor	User
Trigger	Press “Add” button within chores view
Events	-user presses “Add” button -user is directed to a view for adding chore -user selects preexisting chore or fills in chore information and sets schedule (optional) -user presses “Add chore”
Exit Condition	Press “Add chore” or “Cancel”
Post Condition	A new chore is added to the chores list, and the user is directed to the chores view.
Acceptance Test	User is notified that a chore has been added to the list.

Name	**Chores - delete a chore**
Actor	User
Trigger	User swipes left on a chore within the chores view

Events	-user swipes left on a chore -a “Remove” button is pulled out -user presses “Remove” and the button change to “Confirm” -user presses “Confirm” -the chore is removed from the chores list
Exit Condition	Press “Confirm” or press anywhere else on screen to cancel
Post Condition	The chore is removed from the chores list.
Acceptance Test	After removal, user is notified that the chore has been removed.

Name	**Chores - complete a chore**
Actor	User
Trigger	Press a chore in chores view
Events	-user presses a chore -a small window pops up -user fills in information on chore they completed and presses “Confirm” -a notification is sent to other roommates.
Exit Condition	Press “Confirm” or press anywhere else on screen to cancel
Post Condition	The chore is removed from the chores list, added to the history view, and the statistics page is updated.
Acceptance Test	After this action, a notification is sent to other users within the group and the users should not see the chore in the chores list.

Name	**Groceries - add a Communal item**
Actor	User
Trigger	Press “Add item” in communal item view
Events	-user presses “Add item” -user selects a preexisting item or fills in new item information -user presses “Add” and the item is added to the list
Exit Condition	Press “Add” or “Cancel”
Post Condition	User is directed to communal item list and item is added.

Acceptance Test	Users can see the newly added item within the communal item list.

Name	**Groceries - delete a Communal item**
Actor	User
Trigger	User swipes left on an item in the communal item view
Events	-user swipes left an item -a “Remove” button is pulled out -user presses “Remove” and the button change to “Confirm” -user presses “Confirm” -the item is removed from the communal item list
Exit Condition	Press “Confirm” or press anywhere else on screen to cancel
Post Condition	Item is removed from communal item list.
Acceptance Test	After deleting, users will not see this item in the list.

Name	**Groceries - add a personal item**
Actor	User
Trigger	Press “Add items” in the personal item view
Events	-user presses “Add item” -user selects a preexisting item or fills in new item information -user selects whether show this item in aggregate list. -user presses “Add” and the item is added to their personal list.
Exit Condition	Press “Add” or “Cancel”
Post Condition	User is directed to their personal item list and item is added.
Acceptance Test	User can see the newly added item in their personal item list.

Name	**Groceries - delete a personal item**
Actor	User
Trigger	User swipes left on an item in the personal item view
Events	-user swipes left on an item

	-a “Remove” button is pulled out -user presses “Remove” and button changes to “Confirm” -user presses “Confirm” -the item is removed from their personal item list
Exit Condition	Press “Confirm” or press anywhere else on screen to cancel
Post Condition	Item is removed from the user’s personal item list.
Acceptance Test	After deleting, the user will not see the item in their personal item list.

Name	**Groceries - edit a personal item**
Actor	User
Trigger	Press “edit” in personal item view
Events	-user presses edit -user can edit item information -user presses “Add”
Exit Condition	Press “Add” or “Cancel”
Post Condition	User is directed to personal item list and the item is edited.
Acceptance Test	User should see the updated item.

Name	**Groceries - complete a shopping list**
Actor	User
Trigger	Press the “Complete a list” button in the aggregate list view
Events	-user presses the “Complete a list” button -user can select a list of items within the aggregate list -user inputs the total price of shopping trip -user imports a photo of their receipt -user presses “Complete” -notification is sent to other users in the group -cost is split to each roommate’s balance
Exit Condition	Press “Complete” or “Cancel”
Post Condition	A subset of aggregate items is removed, this record is added to shopping list history.

Acceptance Test	After completing a shopping list, users will not see the items he or she selected, and item records are added to shopping history. A notification is sent to users within the group.

Name	**Groceries - edit a completed shopping list**
Actor	User
Trigger	Press “Edit” button in the view
Events	- user presses “Edit” - user can now delete, add, or revise item - user presses “Done”
Exit Condition	Press “Done” or “Cancel”
Post Condition	The list is revised based on the user’s actions.
Acceptance Test	After completing the revision of the shopping list, the list should be updated, retain the updated information, and notify each user within the group of the revision.

Name	**Finance - add a shared bill**
Actor	User
Trigger	Press “Add” button in the view
Events	- user presses “Add” button - user can add roommate(s) to the bill - user can add a description for the bill - user can add an image of the bill (gas, electric, etc.) - user presses “Done”
Exit Condition	Press “Cancel” or “Done” button
Post Condition	A bill is presented to each user within the group that was selected. Their finance tabs are updated.
Acceptance Test	After completing a shared bill, every user within the group who was selected will have an updated finance tab.

Name	**Finance - pay other roommates**

Actor	User
Trigger	Press “Pay” button on the bill
Events	- user presses “Pay” - Venmo payment method initiates (Venmo API) - user is presented with a touch ID for authentication or pin number - user can cancel or proceed with confirming payment
Exit Condition	Press “Cancel” or confirm payment
Post Condition	The user is presented with a UIAlertView of a checkmark on completion of payment. The other user will receive a notification that the user paid him/her.
Acceptance Test	The user who received a payment will receive a notification and their Venmo account will be credited accordingly.

Name	**Finance - view bill history**
Actor	User
Trigger	User presses the “History” within the finance tab
Events	- user presses the “History” button - user is shown a list view of every past bill with the total amounts of those transactions
Exit Condition	User presses “Done”
Post Condition	User is returned to the finance view.
Acceptance Test	The user was able to view previous bills from within the group.

Name	**Finance - remind individual roommate to pay**
Actor	User
Trigger	User presses on the finance tab and clicks the “Remind” button
Events	- user clicks on “Remind” button
Exit Condition	N/A
Post Condition	The user who owes money will be sent a push notification.

Acceptance Test	The user who owes money receives the push notification.

Name	**Finance - remind all roommates to pay**
Actor	User
Trigger	User presses the “Remind all” button
Events	- user is in the finance tab and presses the “Remind all” button at the top of the view
Exit Condition	N/A
Post Condition	Every user who owes money to the current user will be notified with a push notification about the amount they owe.
Acceptance Test	Every other user will receive a push notification with a total amount owed to the reminding user.

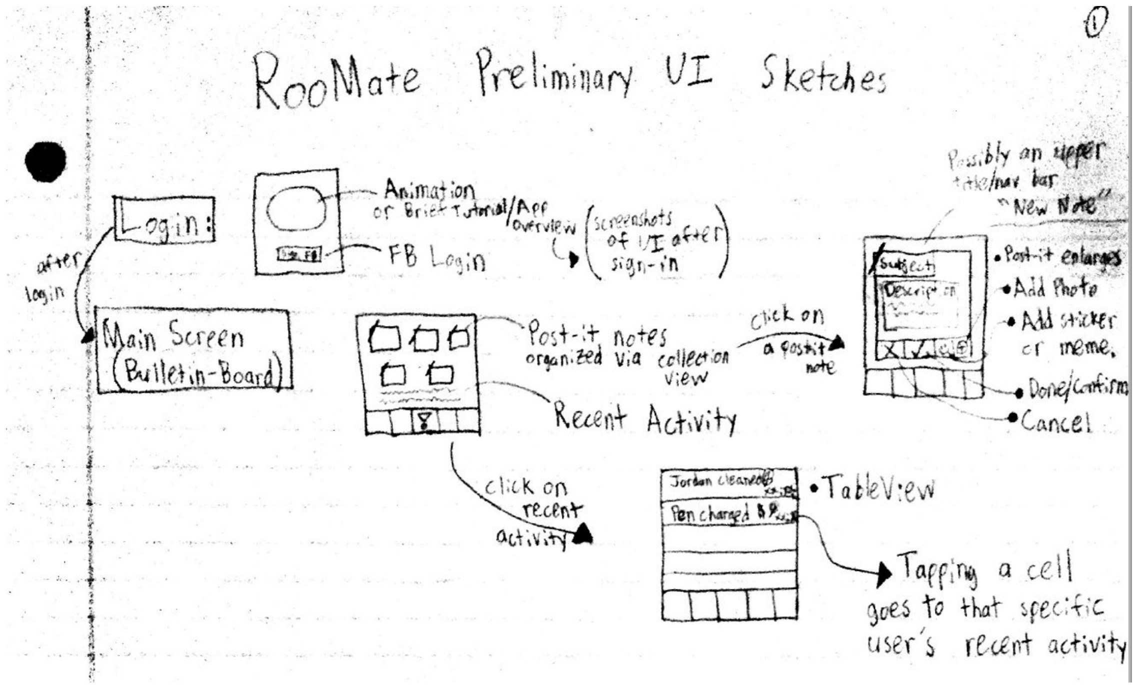
Name	**Bulletin board - add a note**
Actor	User
Trigger	Press “Add note” in bulletin board view
Events	-user presses “Add note” -user is directed to a view for adding a new note -user fills in note and has the option to import a photo -user sets a timer to delete the note (or default timer) -user presses “Add” button
Exit Condition	Press “Add” or “Cancel”
Post Condition	A new note is added in bulletin board and the user is directed to the bulletin board view.
Acceptance Test	After a adding note, users should see the new note in bulletin board. After a certain time, the note should be deleted automatically.

Name	**Bulletin board - delete a note**
Actor	User
Trigger	“3D touch” or “Long Press”

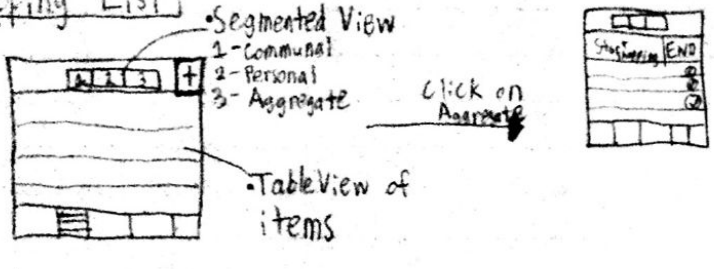
Events	- user is in the bulletin board view - user holds the post-it note for 3 seconds - post note jiggles with an x button on the note - user can press the x to delete the note
Exit Condition	User clicks the “Done” button.
Post Condition	If the user clicks the x button, the note is removed from the board.
Acceptance Test	Users will no longer see the deleted note.

Name	**Bulletin board - view recent activities**
Actor	User
Trigger	User swipes up or taps on the recent activity view
Events	- user taps the recent activity view - recent activity view slides upward showing the last 30 events
Exit Condition	User presses “Done” or swipes downward to close the view
Post Condition	N/A
Acceptance Test	The activity view should show the past 30 events and should be able to slide up and down with ease.

User Interface Requirements

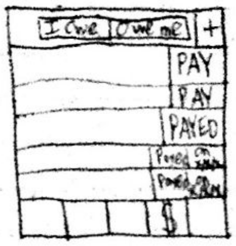

①
RooMate Preliminary UI Sketches
Login:
after login
Main Screen (Bulletin-Board)
Animation or Brief Tutorial/App overview
FB Login
(screenshots of UI after sign-in)
Post-it notes organized via collection view
Recent Activity
click on a postit note
Possibly an upper title/nav bar "New Note"
Subject
Description
• Post-it enlarges
• Add Photo
• Add sticker or meme.
• Done/Confirm
• Cancel
click on recent activity
Jordan cleaned
Pen charged
• TableView
Tapping a cell goes to that specific user's recent activity

Shopping List



- Start Shopping reveals the check marks (to cross off items as they're acquired)
- END button brings user to a page to charge the roommates, but the page can be accessed later from Finances tab.

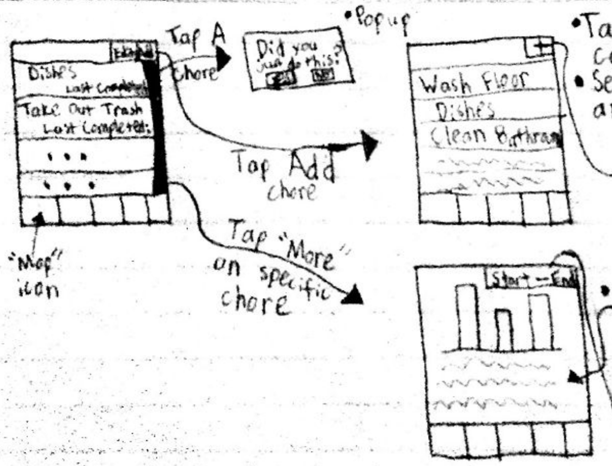
Finances



- Table Cells in "Owe Me" will have a "Remind" button
- Possibly include some type of graph of expenses
- Can click on transactions for more details (like pictures of receipts)

Last tab is "Settings"
- Details of this section will be added during development as they are encountered.

Chores



- TableView of common chores
- Selected ones are grayed out
- Add a custom chore
- Displays chronological history of completions
- Displays bar graph of completions per user over a specified time
- Time Range Picker

Security Requirements

There are multiple points of concern from the safety point of view. We need to have proper checks on both local and networked features. All the payment will be handled via venmo API, but before any outgoing transaction we will perform a TouchID match for the phone's owner. We also will implement a strong auth token check on the backend. Any call that fails due to auth error will result in the user being logged out to prevent malicious activity. The user will also be notified of what happened before we force the log out.

The users will also need to log-in using Facebook, but we will not save any user information beside the names and the auth tokens. Most other parts of the app will be handled by verified third party like Facebook for login, Firebase for secure calls to and from the device, and Venmo for payments. These services have great firewalls in place to prevent any security lapse. We will also be writing Unit Tests to make sure all the safety checks work as desired!

System Requirements

Front End:

- Client Requirements:
  - iPhone 6 or above.
  - iOS 9 or above.
  - WiFi connectivity to access features.
- Technology Requirements
  - Swift 2 or 3
  - Objective - C
  - Xcode 7.3 or above

Justification:

Setting the baseline device at iPhone 6, we want to focus our time and energy to make a fluid, and consistent device across the latest phones that Apple supports. iOS 9 was one of the fastest adopted OS version for iPhones, and has a lot of new API features that are not backwards compatible (stack views, passkit, etc.). Hence we decided on keeping the target iOS version at 9, which works very well with the iPhone 6 and 6+ screen sizes. Since Swift 3 is still fresh and has a few reported stability issues, we will use Swift 2 for the most part. We will also be using Objective-C in moderation, because of the availability of a lot of 3rd party frameworks to implement networking (AFNetworking) and CoreData. Wifi is essential to use the app, because the app will provide a platform to communicate with your roommates and everything will be synced with firebase.

Back End:

- Technology

    - Firebase
    - NodeJS

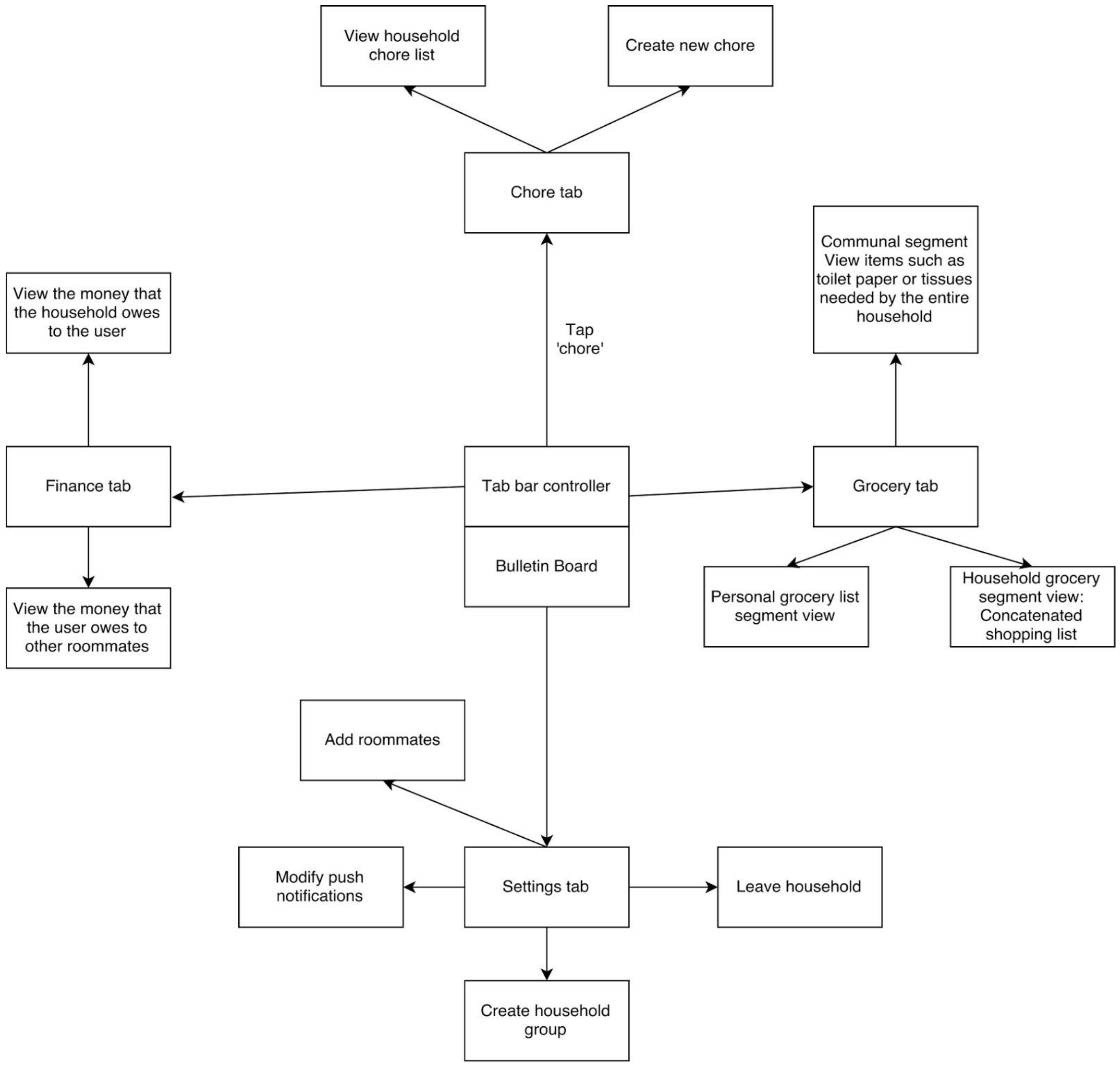
Justification:

Firebase is a real-time backend hosting service. It is necessary for our app to reflect changes in real-time, and firebase provides it with the least amount of setup. Firebase also makes it really easy for us to keep track of logins, and auth tokens. We will implement custom wrappers on firebase for the backend calls using NodeJS to package data using our object requirements, and add an extra layer of authentication to all calls.

Version Control:

- Github for code management (Private Repository)
    - Feature Branching
    - Formal format to assign/accept tickets and reporting issues.
- Google Drive for version specification and release notes.

Specification:


View household chore list
Create new chore
Chore tab
Tap 'chore'
Communal segment View items such as toilet paper or tissues needed by the entire household
View the money that the household owes to the user
Finance tab
Tab bar controller
Grocery tab
Bulletin Board
View the money that the user owes to other roommates
Personal grocery list segment view
Household grocery segment view: Concatenated shopping list
Add roommates
Modify push notifications
Settings tab
Leave household
Create household group