SmartMate

Requirements and Specification Document

2018-09-28, version 1.5

Project Abstract

Creating an amicable living situation between roommates always seems like an easy task, yet again and again, roommate dynamics always seem to tarnish even the strongest relationships, from acquaintances to friends to even family members. For too long, roommates have floundered with cumbersome techniques that prove ineffective in rendering peaceful living quarters— what if you could replace that passive aggressive stare at the dirty dishes with an empty sink? What if the trash *magically* took itself out before you had to remind your roommate even once? *SmartMate* is the most effective tool to date in creating living harmony out of the most strained shared living situations. *SmartMate* is a central application to take care of all things roommate wise, from bill splitting to assigning chores, making peaceful roommate living more attainable than ever.

Team members

Jacob Cordover (cordover@wisc.edu)
Shaoheng Zhou (szhou228@wisc.edu)
Qiuxuan Wu (qwu79@wisc.edu)
Jiatao Ye (jye26@wisc.edu)
Xiaoyang Gong (xgong35@wisc.edu)
Jack Gellerman (jngellerman@wisc.edu)
Anansha Upadhyaya (anansha.divya@wisc.edu)
Patrick Budyn (budyn@wisc.edu)

Document Revision History

Rev. 1.0 <2018-09-21>: initial version, abstract added, customer added
Rev. 1.1 <2018-09-23>: User requirements added, Security requirement finished
Rev. 1.2 <2018-09-24>: Competitive landscape, Use cases, system and UI added
Rev. 1.3 <2018-09-25>: Use cases finished, handscheme for UI revised
Rev. 1.4 <2018-09-27>: Specification flowchart finished, UI images revised
Rev. 1.5 <2018-09-28>: Final review, System finished, Final version

Customer

First, immediate customers will include all developers' roommates. They volunteered to test our application and will provide feedback throughout the iterative process. End users will include anyone living with at least one roommate, as this creates a need to share living responsibilities, such as paying bills, doing chores on time, and updating a calendar record. For testing, developers will enlist roommates, volunteers from university housing, and any acquaintances of the developers' roommates willing to test and use the app. Testing plans include exposures to various use cases to determine an optimal user interface and functionality. First customers will be the following set (each has no previous experience with software engineering): Michael Gui, David Gui, Matt Zeman, Brandon Domash, Jake Horwitz, Gabe Rashba, Ari Seckler, Hengjiali Xu, Jingyi Wang. After a version has been prototyped, the product's audience will broaden to include students of the UW-Madison campus and other potential users with shared living conditions.

Competitive Landscape

After research of current applications on the market and speaking with our customers, SmartMate appears to be a novel, unexplored idea. Each aspect of what SmartMate does is not, per se, unique; the novelty is that SmartMate acts as a central hub to deal with all things revolving around shared living.

That said, competition exists for each core functionality that the application will support, rather than the application as a whole. Examples include SplitWise, a popular bill splitting application, OurHome, a chore management solution, and Venmo and Apple pay, two mobile payment solutions.

OurHome is a chore tracking application that a few of our customers had previously used and enjoyed. Some of the features of OurHome that our customers enjoyed were a simple, easy to use UI, a point system to keep track of who was contributing to what, multiple ways to view who had what chores when, and a multitude of options of who and when to assign chores to. There was also a built in chat and calendar that our users appreciated. Our customers had no major gripes with OurHome— they appreciated that everything was easy to use and that there were multiple ways to complete a task.

SplitWise provides similar functionality as to what is envisioned for the bill splitting component of SmartMate. Strengths include the ability to easily select with who and how to split various payments, direct integration with PayPal and Venmo, the ability to document a receipt from a bill split, all relevant history between two entities, and the ability to complete owed payments in smaller amounts, keeping track of how much is still owed and how much has been payed off. However, there are many perceived weaknesses within SplitWise that SmartMate can capitalize off of, making the bill splitting aspect of SmartMate more advantageous to use.

Perhaps the biggest complaint from our customers that have used SplitWise is that the interface is cumbersome and difficult to learn. We strongly believe in the notion that no matter how much functionality an application offers, a sub-optimal user interface turns many end users away. By employing different HCI data collection methods, such as contextual inquiries, cultural probes, and heuristic evaluations, we think we can capitalize on creating an easier to use interface that will be more appealing for more users.

Another weakness of SplitWise is that when making a payment, it requires the user to leave the application. For example, when settling a payment with Venmo, after a user chooses how much to pay and when they proceed, the user exits the SplitWise application and he or she is taken to the Venmo application. Of course there are technical limitations to APIs, but this is another opportunity for improvement. We will attempt to complete all transactions within the app.

SmartMate’s opportunity lies within taking the best from bill splitting and chore assigning applications, improving upon the negatives of each application and creating a central hub so users do not have to deal with headache of keeping track a number of different applications— SmartMate will replace the need of four to five different applications, including bill splitting, chore distribution and calendars, messaging, and money transfer.

User Requirements

*Priority points are indicated with the numbers on the right, with “1” meaning it will be completed in the first version (iteration) of the project

1. Create Account 1
    a. Allows users make an account. They will use this to connect with their roommates
        i. Username = preferred email address
        ii. Password, which may be re-entered if incorrect
        iii. Name
        iv. Image
        v. Date of Birth

2. Account Activity
    a. Give users log in with their email and password ` 1
    b. View profile 1
        i. Name
        ii. Image
        iii. Email
        iv. Link to Venmo app for reimbursement
    c. Log out 1
    d. Update profile 2
        i. Name
        ii. Image
        iii. Other future personal information to be added
        iv. Link to Venmo account
    e. Delete One’s Own Account 2
    f. Sends passwords to user’s email if the password is forgotten 3

3. Roommate Groups Management
    a. Allow users to create roommate groups 1
    b. Invite other users by typing in their email 1

c. Accept the invites to roommate groups 1
d. Remove group members 2
e. Leave the roommate groups 2
f. Dismiss a roommate group 2
g. Recess the group membership when user is out of town 3

4. Chore Responsibilities
   a. Create chores for group 1
      Each Chore will have requirements:
      i. Name of task 1
      ii. Date to be completed 1
      iii. Assignee (Manual Assignment / Random Assignment Option) 1 / 2
      iv. Repeat frequency 2
         1. Every day
         2. Every week
         3. Every other week
         4. Every month
         5. Every year
   b. Mark Complete for chores 1
   c. Chore History Recording 2
      i. Submit photo documenting the chore’s completion 3
   d. Automatic birthday chore based on members’ profile 2
      i. Advanced Notification (see Notification System)
   e. Special Event Scheduling such as End of Semester Cleaning 3
      i. Find best scheduling time based on user availability
      ii. Automatically create event chore based on schedules above

5. Notification System
   a. Have individual reminders for one user to complete a chore they were assigned 1
   b. Have individual reminders for one user to complete a chore they were assigned 3
   c. Event Notifications for Roommates’ Birthdays in Advance 3

6. Bill payment system 3
   a. Enable users save their credit card information
   b. Give users option to set up/cancel weekly/ monthly/ yearly pay to certain host user
   c. Enable users connect finance platform accounts to make payments
      i. Platforms includes: Paypal / Venmo / Apple Pay
   d. Allow users to pay/receive money on their prefered finance platform

7. Roommate Score System 4
   a. Allow users to set up points for each chores
   b. Report summary of chore score distributions on each roommate member monthly/yearly

Use Cases

*This section particular focuses on use cases with priority 1 (Must have) and priority 2 (Useful). Cases and corresponding features with priority 3+ are listed in subsection “Additional” for future product trending.

Create Account

Name	Create Account
Priority	1
Actors	All roommates users
Triggers	User click on “Sign up”
Events	1. User type in email, password, confirm password 2. User type in name, date of birth 3. User upload image or choose “skip” 4. User click on “Create Account” 5. Submit data to database
Exit Condition	The accounts creating request are accepted or declined
Post-conditions	1. User is sent to their homepage, or 2. Error messages for unfilled information show up
Acceptance Test	Users download the app and sign up the accounts. Users can view and edit their accounts, or start creating a roommate group.

Account Activity

Name	Logging in
Priority	1
Actors	Unlogged in roommate users
Triggers	User opens the app and it is unlogged
Events	1. User types in email as username 2. User types in password

	3. User click “Log in”
Exit Condition	Login request is accepted/denied
Post-conditions	1. User logged in and directed to homepage 2. Error message appears with wrong username/password
Acceptance Test	Users already with accounts who log in to the app with correct email and password are directed to homepage. Users who enter the wrong email or wrong password gets an error message.

Name	View profile
Priority	1
Actors	Logged in roommate users
Triggers	Users click on their profile pictures
Events	Users are directed to profile page
Exit Condition	Profile page shows up
Post-conditions	User is able to see their personal information and a link that can direct them to their Venmo app
Acceptance Test	Logged in users is able to see their personal information via click on the profile pictures. User who have the Venmo app is able to be directed to Venmo by clicking on the link in profile page

Name	Logging out
Priority	1
Actors	Logged in roommate users
Triggers	User click on “Log out” in the homepage
Events	1. Users log out their accounts and have no access to any information and actions with in account. 2. Users are directed to log in page
Exit Condition	Users are directed to “Log in” page
Post-conditions	Users cannot access its account and profile unless they log in again

Acceptance Test	Logged in users are able to log out their accounts via “log out” button and no longer have access to accounts. User is returned to log in page.

Name	Update profile
Priority	2
Actors	Logged in roommate user
Triggers	In profile page, click on “Edit”
Events	1. All information including name, birthday and are other personal information are editable 2. User change information
Exit Condition	User click on “Save” button
Post-conditions	User profile is changed
Acceptance Test	Logged in users can change profile information and sends all information to database

Name	Delete account
Priority	2
Actors	Logged in roommate users
Triggers	User click “Delete account” in profile page
Events	1. Message window pop up confirming the action of deleting account 2. If user clicks “confirm”, delete account from database, return to log in page 3. If user clicks “cancel”, cancel message window.
Exit Condition	1. Account is delete from database 2. Delete action and confirmation window is canceled
Post-conditions	1. Account is completely removed from database 2. Account is kept
Acceptance Test	1. Users with account in SmartMate successfully delete accounts and are unable to log in 2. Users’ accounts are unaffected after cancelling the deleting

	action

Roommate Groups Management

Name	Create Roommate Group
Priority	1
Actors	The host roommate user
Triggers	User click on "Create Roommate Group" after log in
Events	Users start a new roommate group
Exit Condition	Users stay on the new roommate group page
Post-conditions	Users can click the "Invite" to invite other users
Acceptance Test	After the user create/ log in the account, the user can create the roommate group

Name	Invite roommates
Priority	1
Actors	The host roommate user
Triggers	The user click on the "Invite" on the roommate group page
Events	The user clicks on "Invite" and invite other users by typing their emails in the search bar.
Exit Condition	After an user is found in the search bar, the host user can click on the invitee and then an invitation is sent to the invitee.
Post-conditions	The invitee will receive the invitation.
Acceptance Test	The host user click on the "Invite" and search an email in the search bar. After the host found and click on the invitee. An invitation is sent to the invitee.

Name	Accept invites

Priority	1
Actors	Any logged in users
Triggers	Host roommate uses sends an invitation to the invitee user
Events	1. Invitee have a message window pop up 2. Invitee clicks on “Accept” or “Decline”
Exit Condition	Message window is removed
Post-conditions	1. If the invitee clicked the “Accept” button, he/she should be added into the roommate group. Database should be updated. Roommate group should also appear on the home page. 2. If the invitee clicked the “Decline” button, he/she should return to the app page. Message should be removed.
Acceptance Test	Users who gets invitations can see the message window. By clicking “Accept”, user is successfully added to the corresponding roommate group. By clicking “Decline”, the message is properly cancelled.

Name	Remove members in the Roommate Group
Priority	2
Actors	All roommates
Triggers	Click the Group Member Management button
Events	1. A list of group member appears 2. Click the remove button on the right side of the member to remove it from the group
Exit Condition	The remover member will receive a notification and be removed from the group.
Post-conditions	The list of group member will be updated after the member is removed
Acceptance Test	The user removes another user in the roommate group. And that user receive a notification and be removed from the group.

Name	Leave the roommate group
Priority	2

Actors	All roommates
Triggers	Click the Group Member Management button
Events	1. A list of member appears 2. Click the leave button on the right side of the user
Exit Condition	The user leave the group and other members in the group receive notifications.
Post-conditions	The list of group member will be updated after the member leaves.
Acceptance Test	The user leaves the roommate group. And other users receive notifications.

Name	Dismiss roommate group
Priority	2
Actors	Host users of roomate groups
Triggers	Host click “Dismiss group” from group management page
Events	1. Message window pop up confirming the action of dismissing groups 2. If host user clicks “confirm”, delete group from database. 3. All group member receive notification of dismissment 4. If host user clicks “cancel”, cancel message window.
Exit Condition	Host user clicks “confirm”/”cancel”
Post-conditions	1. Remove group information on all group members’ home page 2. Group information are kept
Acceptance Test	1. Host roommate in groups successfully dismiss a group where no one in group still has access to the group 2. If action is cancelled, nothing in the group should be affected

Chore Responsibilities

Name	Create chores for group
Priority	1
Actors	Roommate user in roommate groups

Triggers	There occur needs for chores. Roommate user click “Create new Chore”
Events	1. Create name of task 2. Create date to be completed 3. Assign task to Assignees Manual Assignment (Potentially Random Assignment Option) 4. Set up repeat frequency 5. Set up task completion indicator
Exit Condition	Finish configuring task specifications.
Post-conditions	Close task after it is marked as finished
Acceptance Test	User can successfully add chores to a log of Chores. Chore tab would show up at assignee’s chore list.

Name	Mark complete for chores
Priority	1
Actors	Roommate user in roommate groups
Triggers	Chore being completed by assignee and record it electronically
Events	1. User click on “mark complete” in specific chore bar 2. Chore appear to be completed 3. Chore is removed from to do list
Exit Condition	User click “mark complete” and click “return”
Post-conditions	User do not have the completed chore showing up in the to do chore list
Acceptance Test	Users can successfully enter certain chore tab. The chore tab disappears from chore list after marked complete

Name	Record Chore History
Priority	2
Actors	All roommates
Triggers	When any roommate completes a chore
Events	Create entry with chore name

	Add date of completion Give credit to the roommate that completed the chore Rate how well the chore was done
Exit Condition	Finish configuring log entry of completed chore
Post-conditions	Close entry after all fields have been entered for completed chore
Acceptance Test	Chore History tab has proper entry

Name	Birthday Chore
Priority	2
Actors	All roommates
Triggers	When any roommate has a birthday
Events	1. Notification that it is a roommate's birthday 2. Add date of completion
Exit Condition	Finish configuring app's notification of a birthday
Post-conditions	Yearly occurrence, will stop sending notifications on days besides the roommates birthday
Acceptance Test	Birthday Notification is sent

Notification System

Name	Individual reminders for completing a chore
Priority	1
Actors	Roommate user in roommate groups
Triggers	A task is assigned
Events	1. When the task is assigned to a specific user on a specific day, an auto reminder is set on that day.
Exit Condition	The user can click on " Completed!" on the reminder to close the reminder
Post-conditions	The task is labeled as completed in the task list.

Acceptance Test	When the task is assigned, the user will receive a reminder about the task on the specific day. When the user click on " Completed" on the reminder, the reminder is closed and the task is labeled completed in the task list.

Additional Features (Priority 3+)

Name	Set up the monthly/yearly payments (Bill Payment System)
Priority	3
Actors	One of the roommates (host) who is responsible for paying the whole rent
Triggers	Send an auto reminder each month
Events	1. Set up the auto reminder for the host to pay; assign different pay amount to other roommates when creating the group 2. The app will remind the host to pay the whole rent at a specific day each month
Exit Condition	The auto reminder appears on the phone
Post-conditions	1. The host click "Paid!" on the reminder window to close the reminder 2. Notifications are sent to other roommates to let them know that the rent is paid
Acceptance Test	The user set up the auto reminder for paying the rent and assign different amount to other roommates when the group is created. A reminder is sent to the user each month at the specific day. After the user pays the rent, the user clicks "Paid!" to close the window. And then notifications will be sent to other roommates.

Name	Pay through finance platforms (e.g. venmo) (Bill Payment System)
Priority	3
Actors	Roommates other than the host
Triggers	Send an auto reminder each month
Events	1. The auto reminder is set up after the host decides the pay day 2. A reminder will be sent to those roommates with specific amount they need to pay at a specific day before the pay day

	3. The users can click a button and jump to venmo to pay the bill (Further: The users can click a button to send money via their prefered platform, but stays inside SmartMate)
Exit Condition	The users click the button and jump to venmo to pay the bill. And then they close the reminder.
Post-conditions	When the reminder is closed, a notifications will be sent to the host.
Acceptance Test	Other roommates receive the pay reminder at a specific day. They click a button on the reminder and jump to Venmo to pay the bill. After they pay the bill, a notification will be sent to the host.

Name	Special Event Scheduling (Chore Responsibilities)
Priority	3
Actors	All roommates
Triggers	When any roommate realizes a need to schedule a special event
Events	1. Create entry with chore name 2. Add date of completion and description 3. Specify “Special Event”, e.g. End of Semester Cleaning 4. Rate time each roommate should complete
Exit Condition	Finish configuring entry of special event
Post-conditions	Close entry after all fields have been entered for the Special Event to be Completed
Acceptance Test	Special Events tab has proper entry

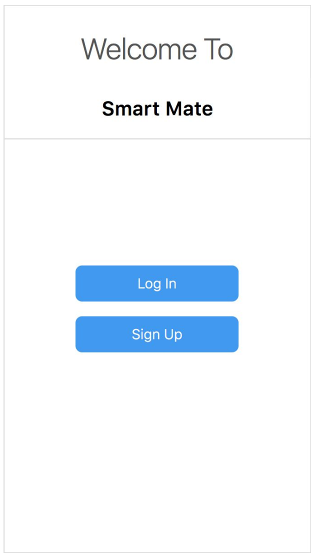
User Interface Requirements

In user interface requirement sections, we will include our UI flow diagram as well as our app screen layouts for iOS. The main tool that we used for the UI design is

- Proto.io.
- InVision

As a result of our discussion, a few concepts/features must be included if we wanted to integrate different functionalities such as chore, payment, calendar, into one app.:

- Clear and simple user interface, i.e., clear stated functionality and little complex procedures (only 1-4 functionalities on each page of the app).
- Back button can be always found near the bottom of the page for users to navigate back (consistency and reversible)
- Popup window for user confirmation to prevent user input error
- Certain features such as adding payment method, removing members and leaving group ask user's confirmation via pop-up window.
- Groups/members management should not be complicated, i.e.

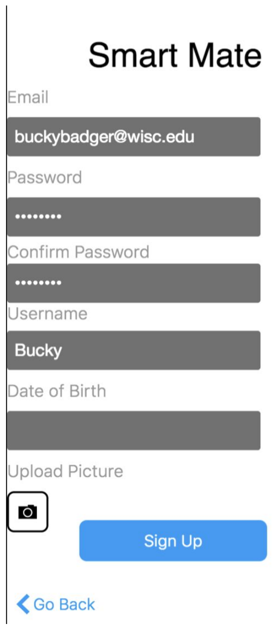


(figure 1.1)

First Page

This serves the purpose of user sign-up/log-in.

Account Activity


Smart Mate

Email

buckybadger@wisc.edu

Password

••••••••

Confirm Password

••••••••

Username

Bucky

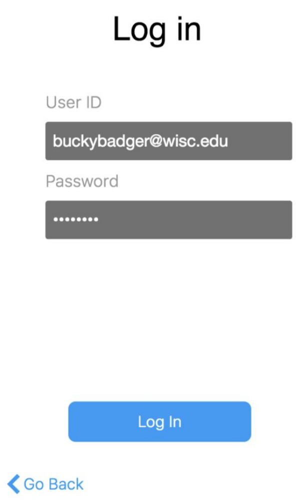
Date of Birth

Upload Picture

(figure 1.2)

Sign up

User inputs username, email, password and optional image to create an account.


Log in

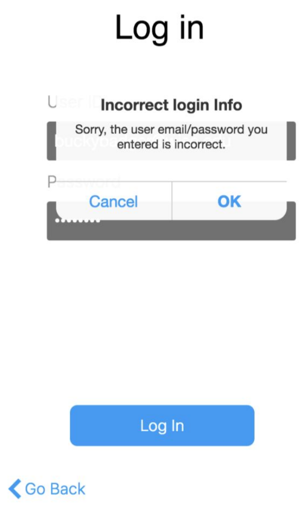
User ID

buckybadger@wisc.edu

Password

••••••••

(figure 1.3)


Log in

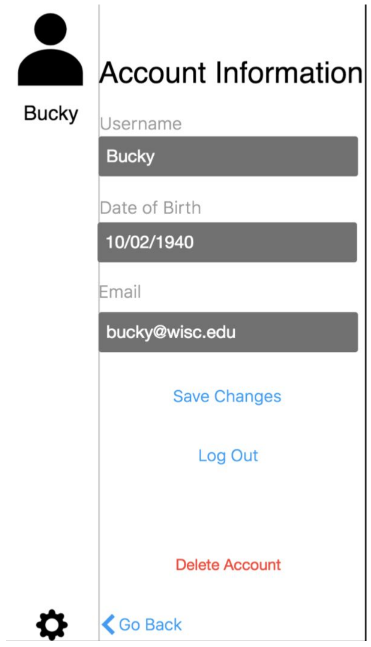
Incorrect login Info

Sorry, the user email/password you entered is incorrect.

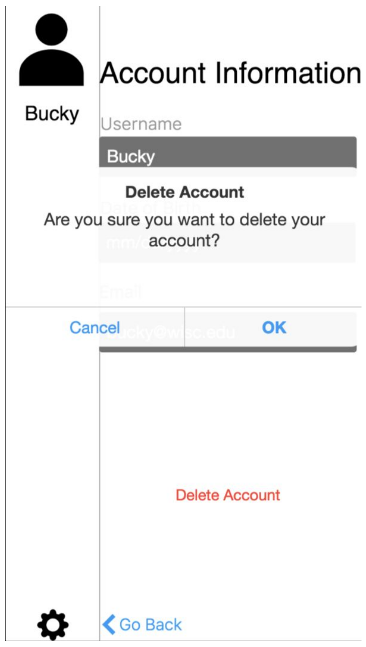
(figure 1.4)

Log in

User enters username/email address and password. A window will pop up for wrong login information.



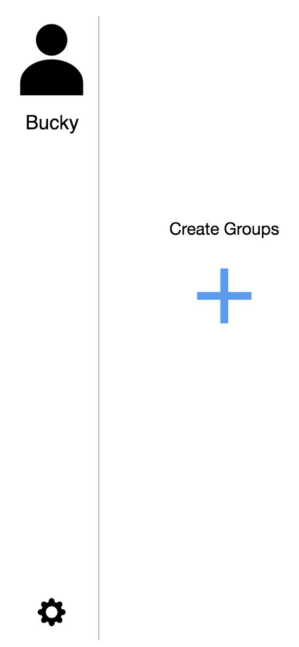
(figure 1.5)



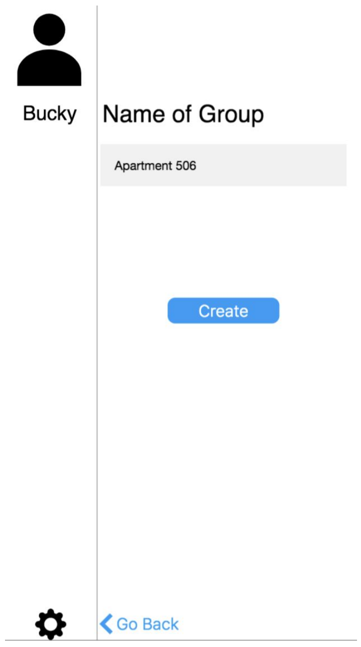
(figure 1.6)

Clicking user's ***profile picture*** *(located on the top left of the app)* allows user to change and save user information including account modification and removal. A window will pop up for user confirmation to prevent user error.

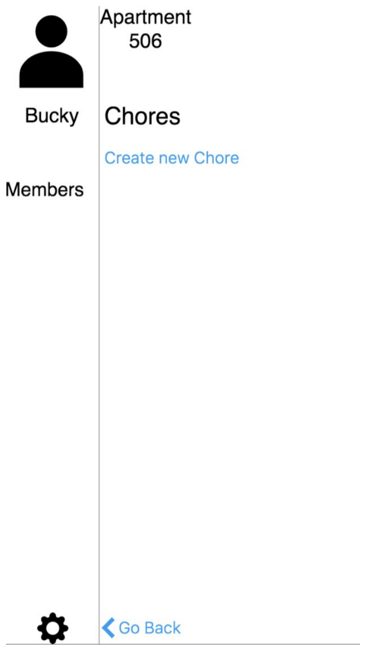
<u>Roommate Group Management</u>



(figure 1.7)



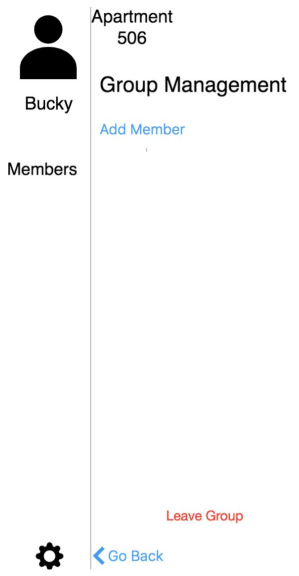
(figure 1.8)



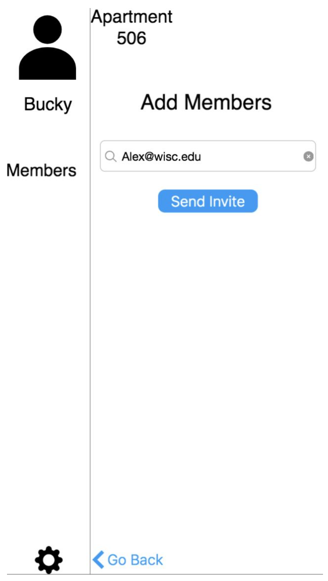
(figure 1.9)

Empty Home Page

The process of user creating a new group in the empty home page. During this process, user clicks the ***blue plus button*** in figure 1.7 to navigate to a new page in figure 1.8. Once the user inputs the name of the group and clicks the ***Create*** button, user will successfully build a new group, shown in figure 1.9. (This is the main page where user can see chores and group members).

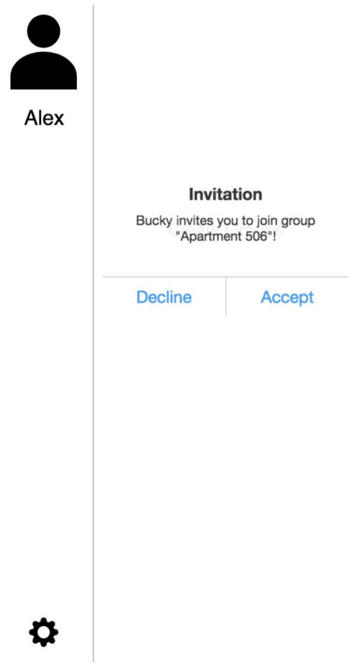


(figure 1.10)



(figure 1.11)

By clicking the ***gear*** button located on the ***bottom left*** of the app, user can manage the group. In the group management, after clicking the ***Add Member*** button, user can invite member to join the group by simply entering member's email and clicking the ***Send Invite*** button

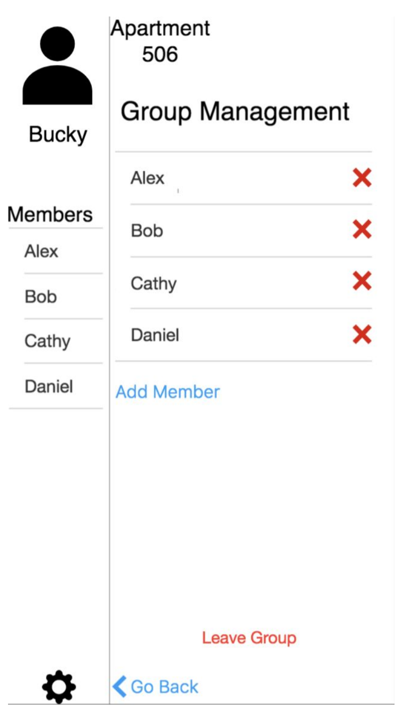


Invitation

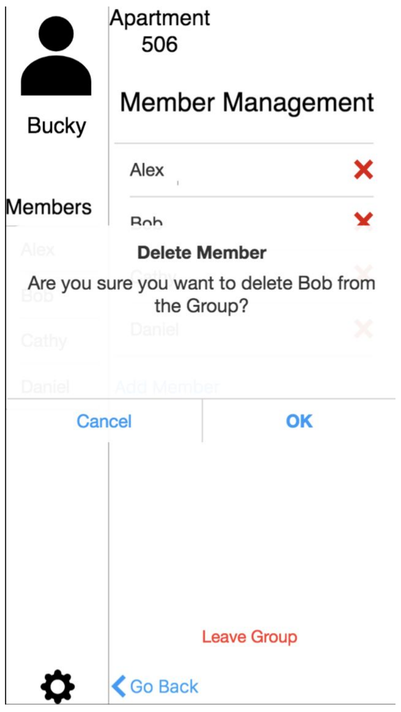
Bucky invites you to join group "Apartment 506"!

(figure 1.12)

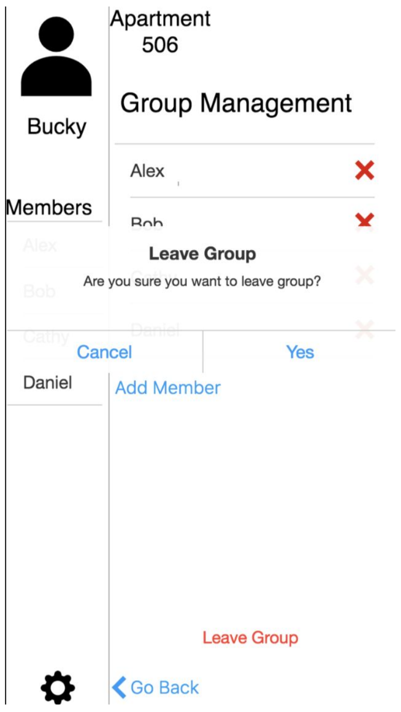
In figure 1.12, a user named ***Alex*** received an invitation from ***Bucky*** to join group "Apartment 506". When a user sends an invitation to a member, the member will receive a notification pop up window on their end. The member can accept or decline the invitation.



(figure 1.13)



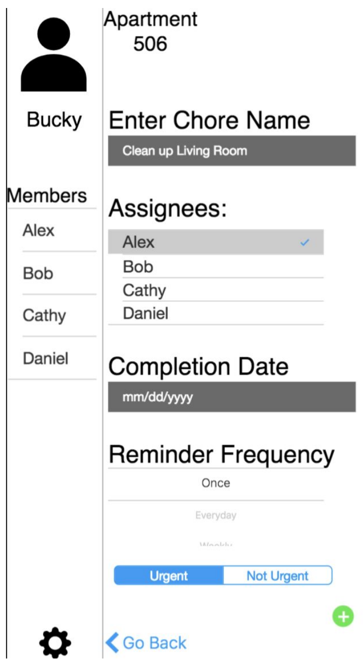
(figure 1.14)



(figure 1.15)

Now, we have a list of members in the group. In the Group Management feature (***gear button***), user can easily add members or remove members from the group. Removing a member requires users to click the ***red cross*** (figure 1.13) and a window will pop up for user's confirmation (figure 1.14). Users can also leave the group at anytime. A confirmation window will pop up when user clicks Leave Group option for error prevention (figure 1.15).

Chores Responsibility



Enter Chore Name

Clean up Living Room

Assignees:

✓ Alex

Bob

Cathy

Daniel

Completion Date

mm/dd/yyyy

Reminder Frequency

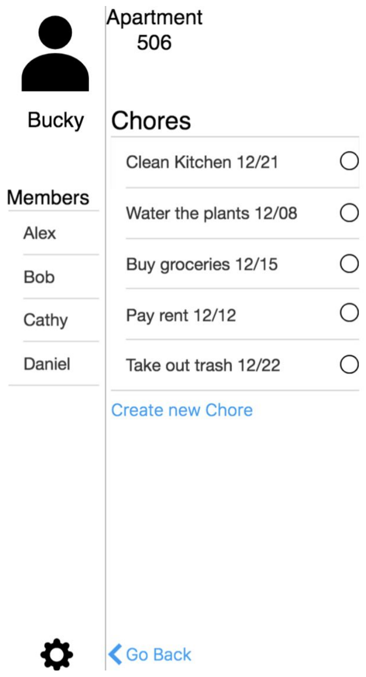
Once

Everyday

Weekly

[✓ Urgent] Not Urgent

(figure 1.16)



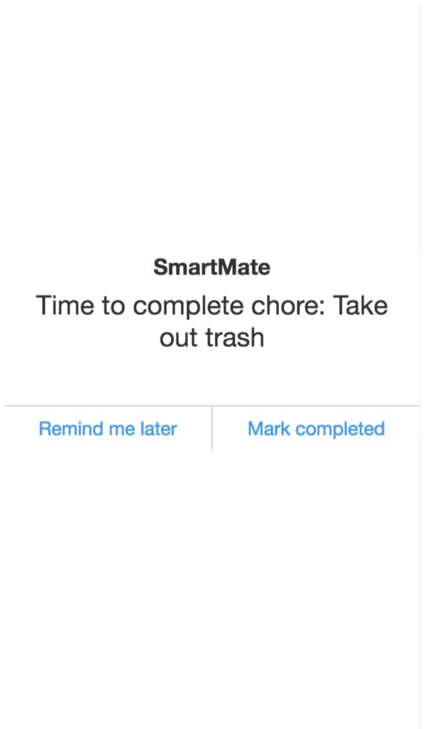
Chores

Clean Kitchen 12/21	○
Water the plants 12/08	○
Buy groceries 12/15	○
Pay rent 12/12	○
Take out trash 12/22	○

(figure 1.17)

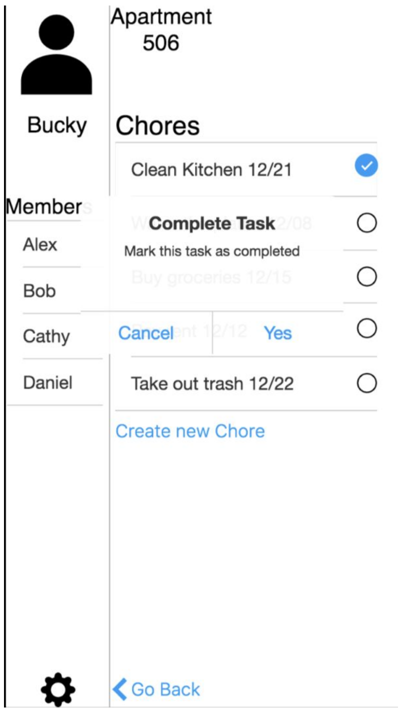
In figure 1.9, there is a ***Create new Chore*** button located in the middle of the app. Clicking that allows users to create a chore task with a chore name, assignee, a completion date, a repeat feature (frequency), and an urgent highlight feature (figure 1.16). When users finish inputting the chore detail, user can hit the ***green cross*** button located on the bottom right of the app. Doing so will send this chore to assignees that are related to this chore.

Figure 1.17 shows a list of chores users need to fulfill. This is the complete setup of our Group User Homepage



(figure 1.18)

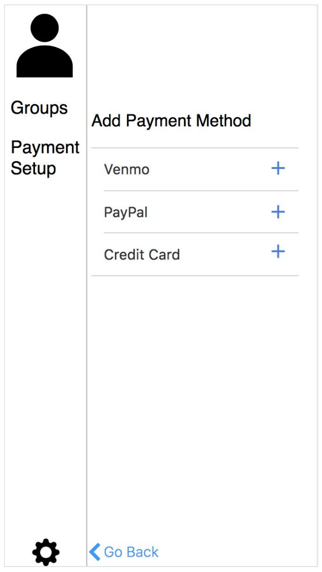
A chores notification will pop up according to the deadline of each task.



(figure 1.19)

Users can mark completed task/chores by hitting the ***empty circle*** next to the chore, a window will pop up and ask for user confirmation.

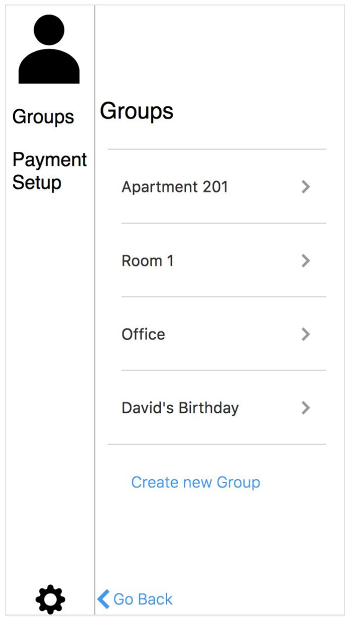
Below are potential features that we can add to our app:



(figure 1.20)

Payment Setup (potential features)

Users can link different finance platform (Venmo/Paypal/Apple Pay/credit card) accounts for sending/receiving payment when they click the ***Payment Setup*** button. Clicking the ***blue cross*** allows users to link user account to desired payment method.

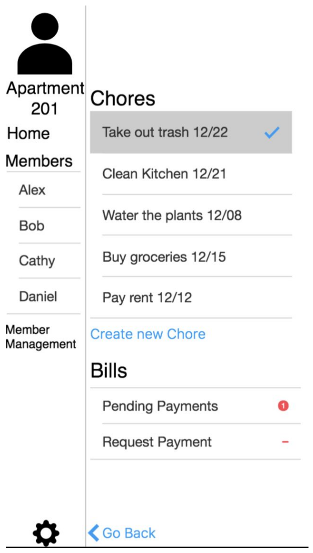


(figure 1.21)

Multiple Groups (potential features)

The Groups feature will return a list of different groups that the user is in.

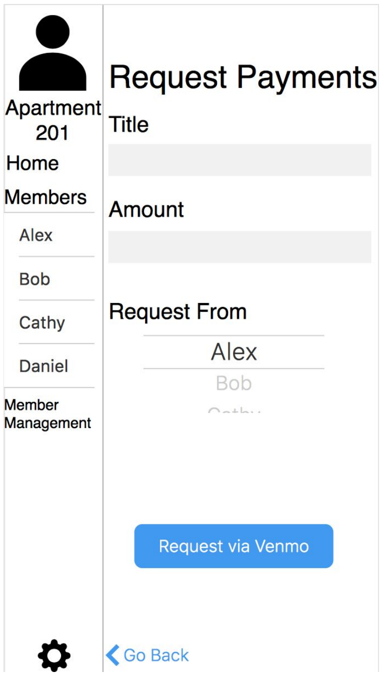
User can also create new group with the ***Create new group*** button.



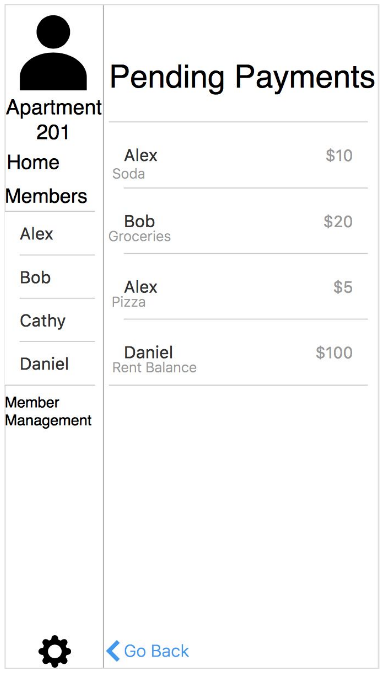
(figure 1.22)

Certain group homepage setup with Bill Payment System (potential features)

When users clicks a specific group, members' information will be presented on the left side of the page while features such as chores, and upcoming bills will be shown on the right.



(figure 1.23)



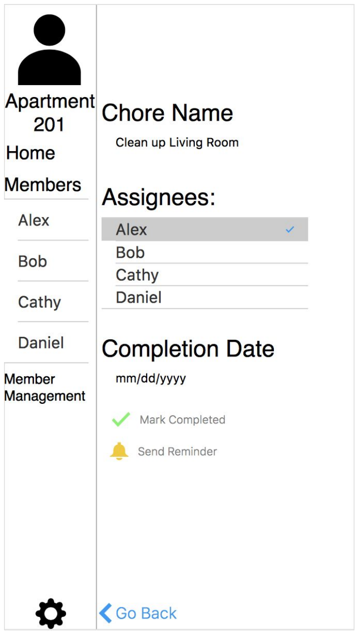
Alex Soda	$10
Bob Groceries	$20
Alex Pizza	$5
Daniel Rent Balance	$100

(figure 1.24)

Request Payments & Pending Payments (potential features)

Under the Bills section, users can click on request payment option in (figure 1.22). Then user can enter the payment title as well as amount of money and send request to other member of the group. Users can choose a member from the ***Select From*** feature.

The pending payments button is found under the Bills section (figure 1.22). In pending payments users can see which payment is not being paid by members (figure 1.24).



(figure 1.25)

Chores detail (potential features)

Clicking a specific chore task will return details of this task. User can edit tasks, mark it as completed, or send reminder to members.

Security Requirements

- **Confidentiality**
  - Two-factor Authentication
    - When user access critical data like bank account information or authorize payment to another party, a PIN other than login password should be required.
  - Encryption
    - Data for user account / roommate group should be encrypted.
    - Outbound traffic from App, i.e. payment request, should be encrypted to prevent eavesdropping.
- **Integrity**
  - Digital certificate
    - A digital certificate should be required when user is redirected to another service (e.g. Venmo's website) to avoid phishing website.
  - Access Control

        - User have access to read/write permission only when he is in the roommate group.
        - Access to write permission for events in user group, such as scheduled bill payment, chore split and calendar functionalities, should be limited to individual users.
- **Availability**
    - Backup
        - Database server should have backup periodically.
        - User should have a local copy of most recent data of his roommate group .
    - DDOS defense
        - KYC process
            - To prevent attacker from spamming the database, access to core functions, such as creating schedules, should be restrained unless user has linked their payment account.
        - CAPTCHA
            - CAPTCHA should be required when registering new account, creating new roommate group and login from unknown device.

System Requirements

Client Requirements:

Iphones

- iOS 7 or better
- Internet access (WiFi or data)
- ~15mb of storage

Android

- Android 4.4 or better
- Internet access
- ~15mb of storage

iOS 7 is the base version used by most iphones in circulation today. These devices will need internet access in order for them to be downloaded from the app store and to process transactions between peers. After sampling multiple apps in the app store, we think smartmate won't take more than 15mbs.

Android 4,4 is the base version used by most android devices in circulation today. Like the iphones, android devices will need internet access to download the apps and utilize all of the app's features. After more research we believe the android devices will need a similar amount of storage.

We will focus first on building iOS mainly since majority of our testers use iOS device.

Frontend:

Idea1:

Xcode with Swift

- Xcode 10 include Swift 4.2 and SDK for iOS12
- We need to create a wireframe, a document that creates a user roadmap and an architecture for our app's information

Proto.io, inVision and Sketch for User Interface design

- We will lay out each screen that our users will interact with in storyboards.

Idea2:

React Native

- Compatible with IOS and Android
- React and JavaScript to build mobile UI both for IOS and Android

Backend:

Idea1:

SQLite, PHP

- SQLite/MySQL will be the database that store our user data.
- Access to PHP server.
- We need to write a PHP web service to query the database
- To send data from PHP to an iOS device, we will be having the PHP code send it as JSON

It is possible to use Swift to send some post request to a PHP script, which will happen on a server. Functionalities such as querying the database and registering users can be achieved if we connect our Swift app to a PHP web service.

Idea2:

Firebase

- Compatible with IOS and Android
- Firebase *Authentication* to realize account activity and security requirement

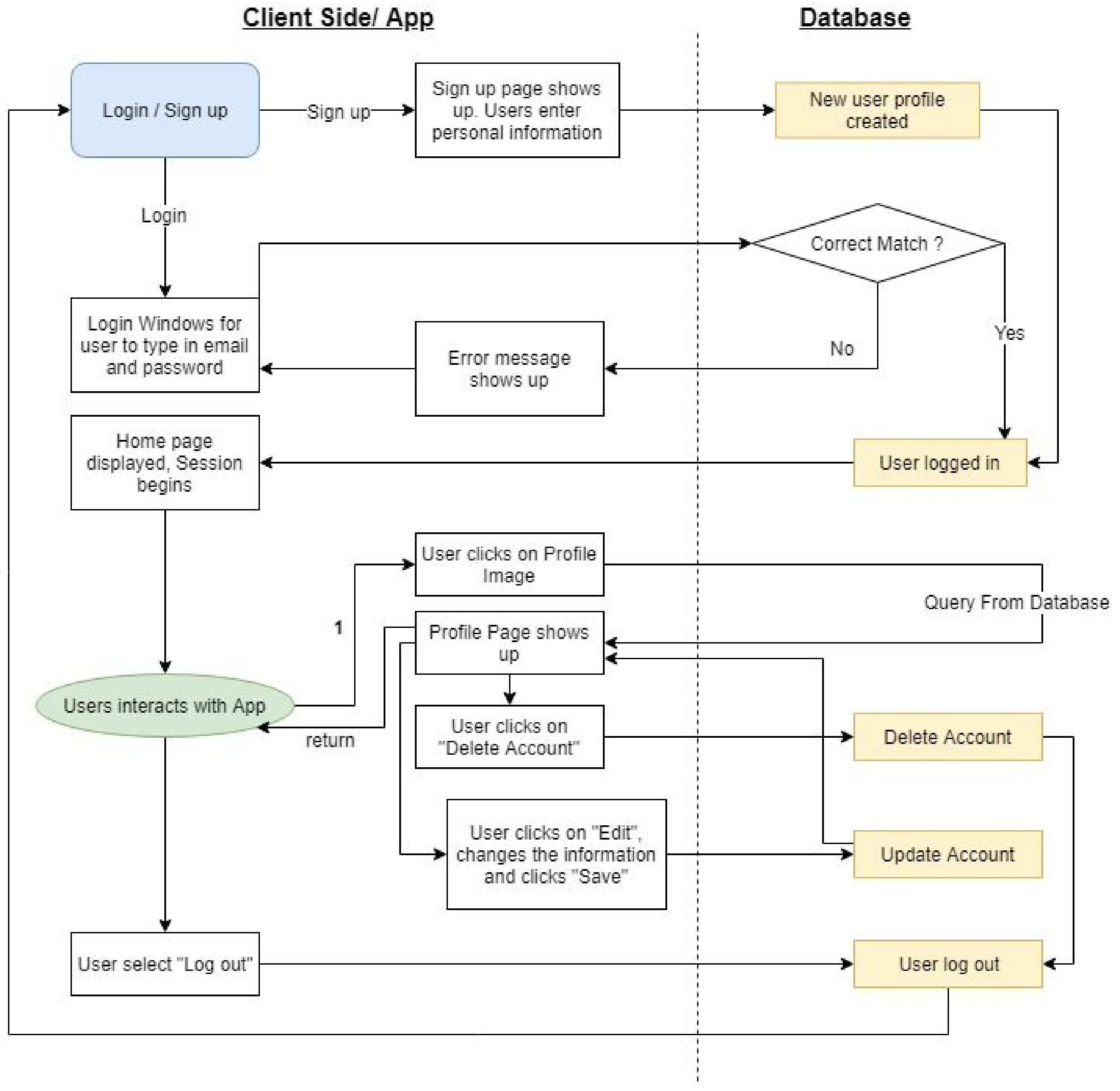
    - Firebase *Realtime Database* to update our chore responsibility, calendar event and instant message in realtime
    - Firebase *Cloud storage* to store any related files and images
    - We could do analytics, databases, messaging and crash reporting about the app all in firebase.

Version control:

- Github repository for code management
  - Private repository for the project
  - Each user owns a fork and branch of the project
- Google Drive for file management and planning

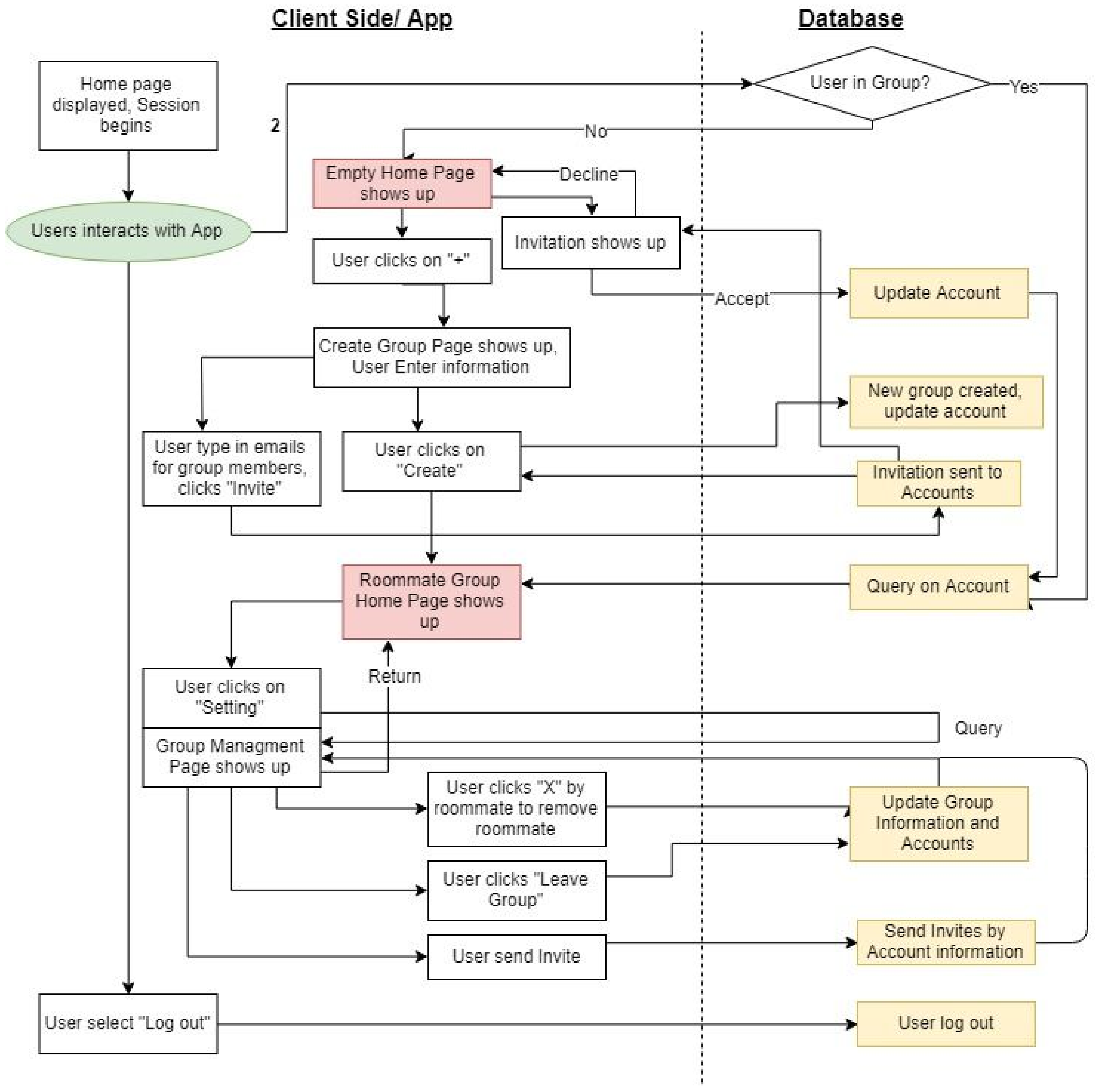
Specification

Account Acticity

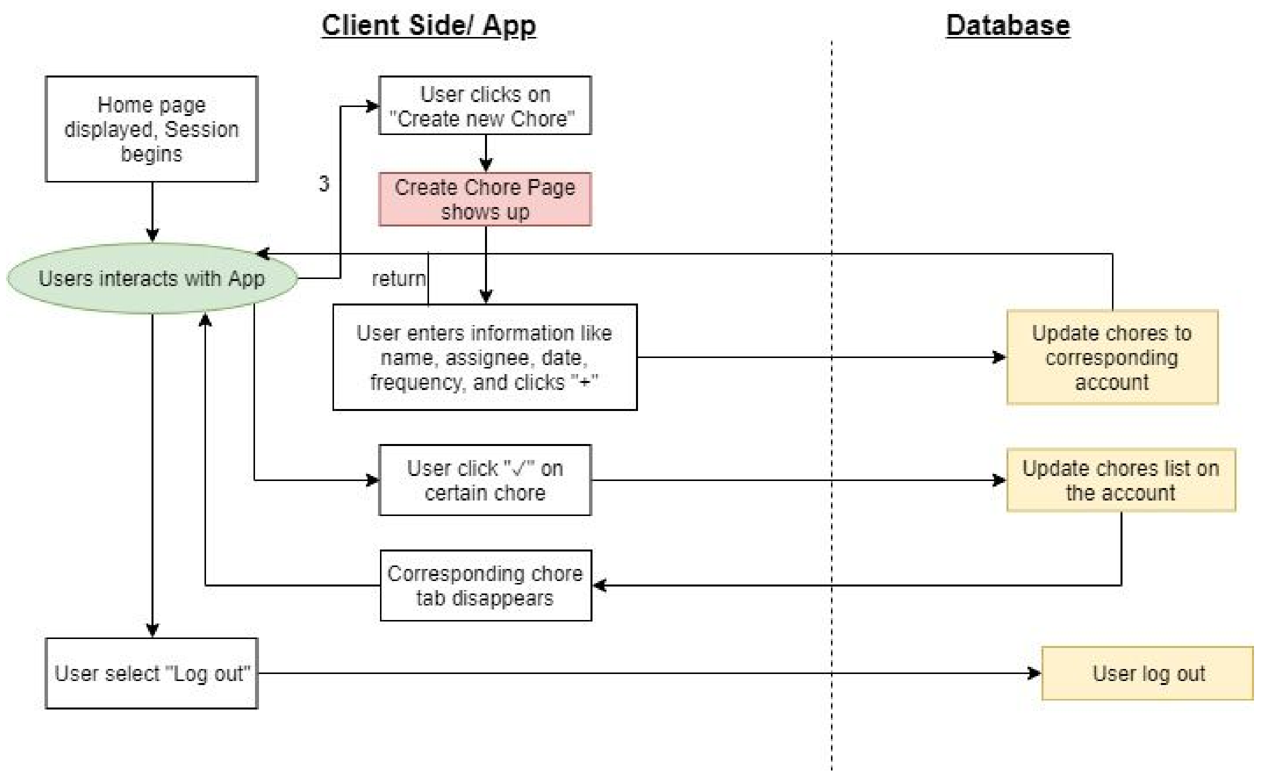


Client Side/ App
Database
Login / Sign up
Sign up
Sign up page shows up. Users enter personal information
New user profile created
Login
Correct Match ?
Login Windows for user to type in email and password
Error message shows up
No
Yes
Home page displayed, Session begins
User logged in
User clicks on Profile Image
Query From Database
1
Profile Page shows up
Users interacts with App
return
User clicks on "Delete Account"
Delete Account
User clicks on "Edit", changes the information and clicks "Save"
Update Account
User select "Log out"
User log out

Roommate Group Management


Client Side/ App
Database
Home page displayed, Session begins
Users interacts with App
2
User in Group?
Yes
No
Empty Home Page shows up
Decline
Invitation shows up
User clicks on "+"
Accept
Update Account
Create Group Page shows up, User Enter information
New group created, update account
User type in emails for group members, clicks "Invite"
User clicks on "Create"
Invitation sent to Accounts
Roommate Group Home Page shows up
Query on Account
Return
User clicks on "Setting"
Group Managment Page shows up
Query
User clicks "X" by roommate to remove roommate
Update Group Information and Accounts
User clicks "Leave Group"
User send Invite
Send Invites by Account information
User select "Log out"
User log out

Chore Responsibility


Client Side/ App
Database
Home page displayed, Session begins
Users interacts with App
3
User clicks on "Create new Chore"
Create Chore Page shows up
return
User enters information like name, assignee, date, frequency, and clicks "+"
Update chores to corresponding account
User click "✓" on certain chore
Update chores list on the account
Corresponding chore tab disappears
User select "Log out"
User log out