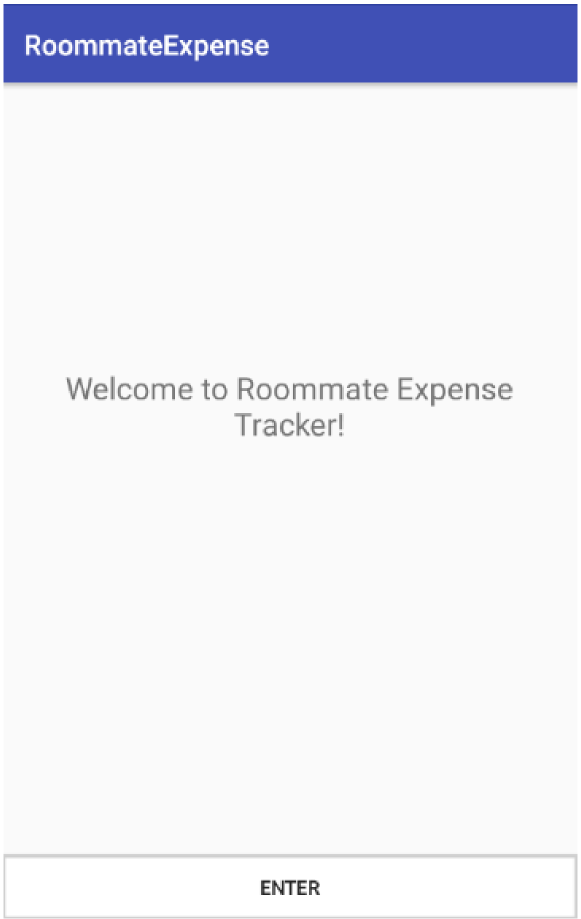
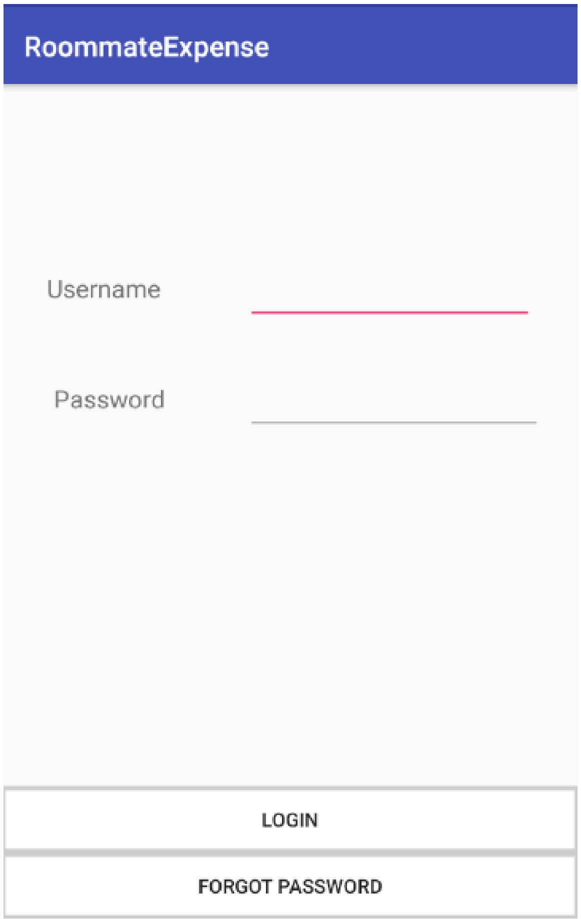
Group #26: Huiyi Ruan, Lily (Yi Di) Wu, Kelly (Pei Lun) Hung, Eve Phelps
CIS 350 001
April 23rd, 2018
Handoff Documentation

## 1. User Manual

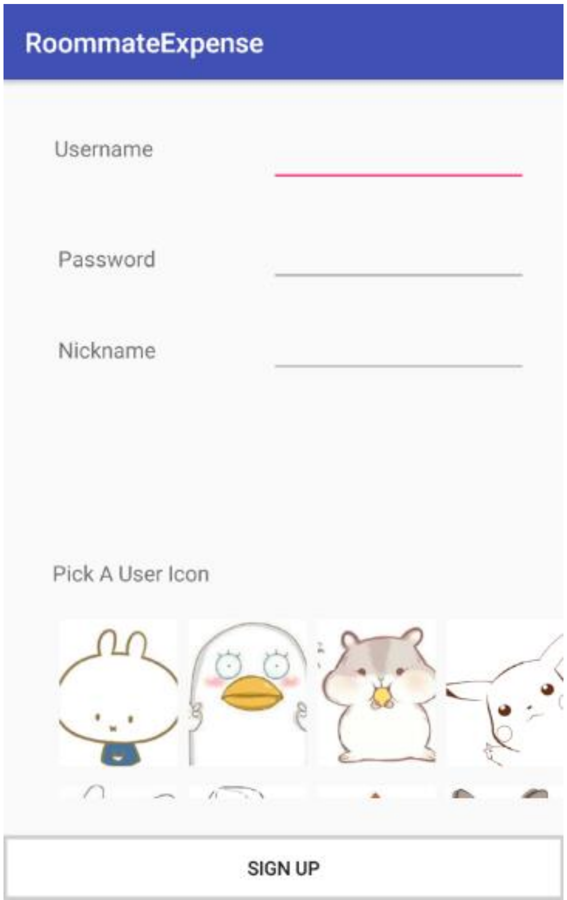


Welcome to Roommate Expense Tracker!

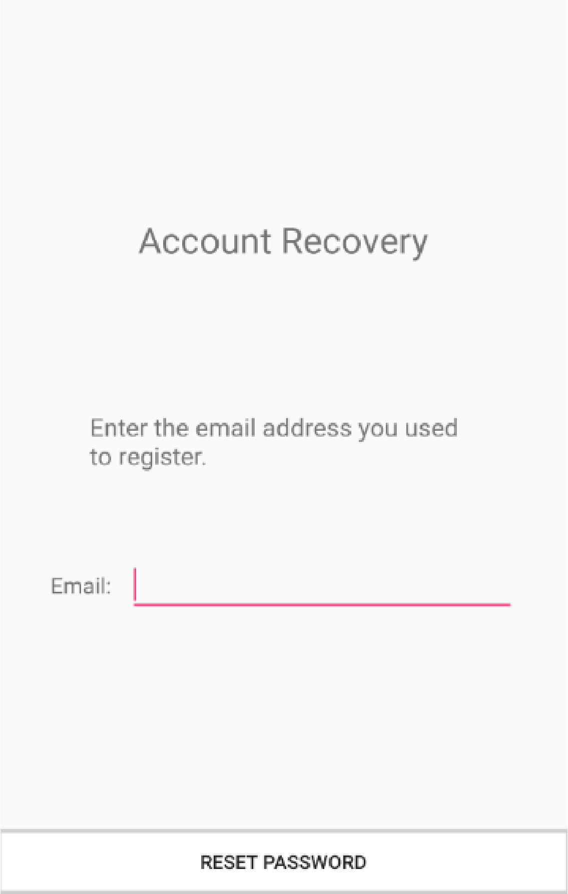
Splash Screen. Press [Enter] to enter the app.



Login. Enter your username-password combination, then press [Login] to get to the Menu page. If the username entered is not associated with an account, you will be redirected to the Signup page. If the combination is entered incorrectly three times, you will be redirected to the Password Reset page and forced to reset your password. Also, if you have forgotten your password, press [Forgot Password] to begin the password reset process.

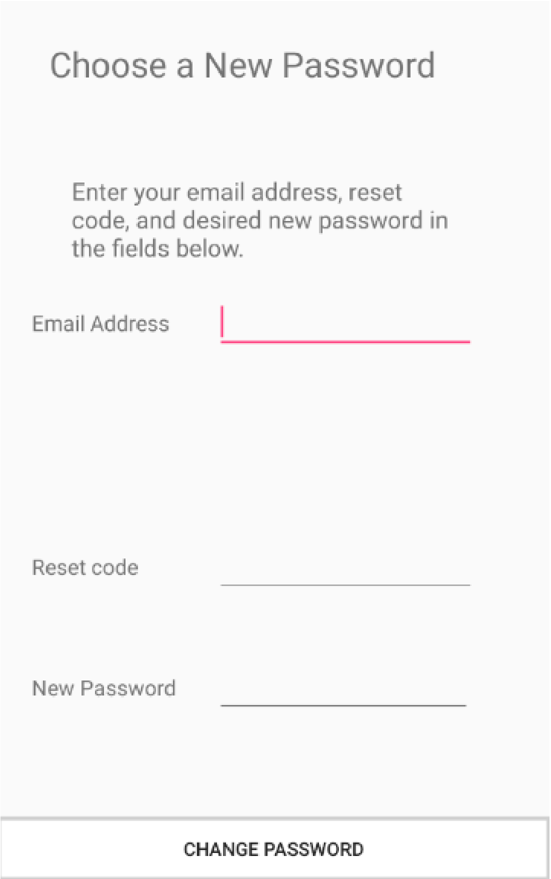


Signup. Enter your desired username (email address), password, and nickname. Additionally, choose an icon (avatar). Scroll through the icon list to see all avatar options. Press [Sign Up] to complete the transaction. If the entered username is already associated with an account, you will be told to pick a different username (email address). Upon successful signup, you will be redirected to the Login screen.



Enter the email address you used to register.

Forgot Password. Enter the username (email address) of the account for which you have forgotten your password, and then press [Reset Password], which will re-direct you to the Password Reset page. If the email is associated with an account, we will send an automated recovery code (which expires in one week) to that email address, so check your email!



Choose a New Password

Enter your email address, reset code, and desired new password in the fields below.

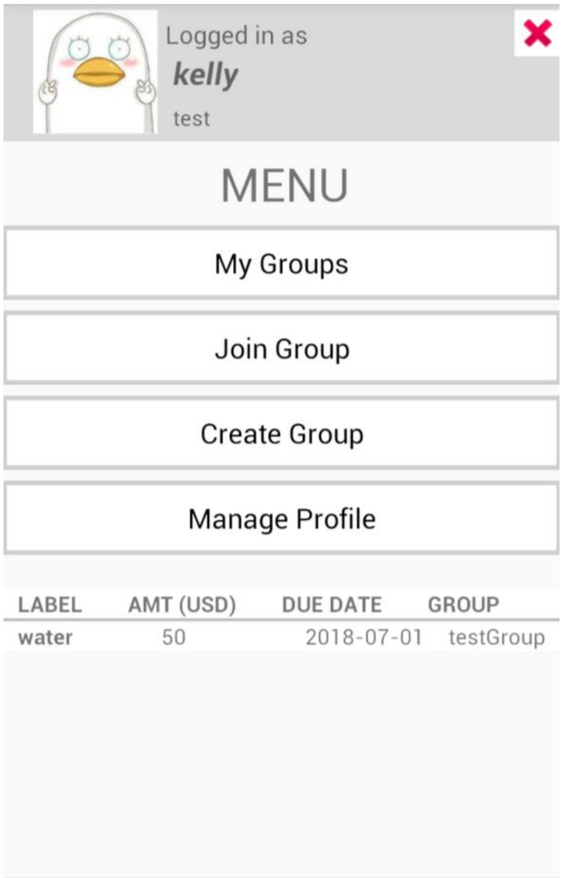
Email Address

Reset code

New Password

CHANGE PASSWORD

Password Reset. Enter your email address (username), the reset code we just sent you, and a new password, and then press [Change Password]. If the reset code is correct, you will be re-directed to the Login page, where you can use your new password to login. If the reset code is incorrect, you will be told so. If the reset code has already expired, you will be told so.



Logged in as
*kelly*
test

MENU

My Groups

Join Group

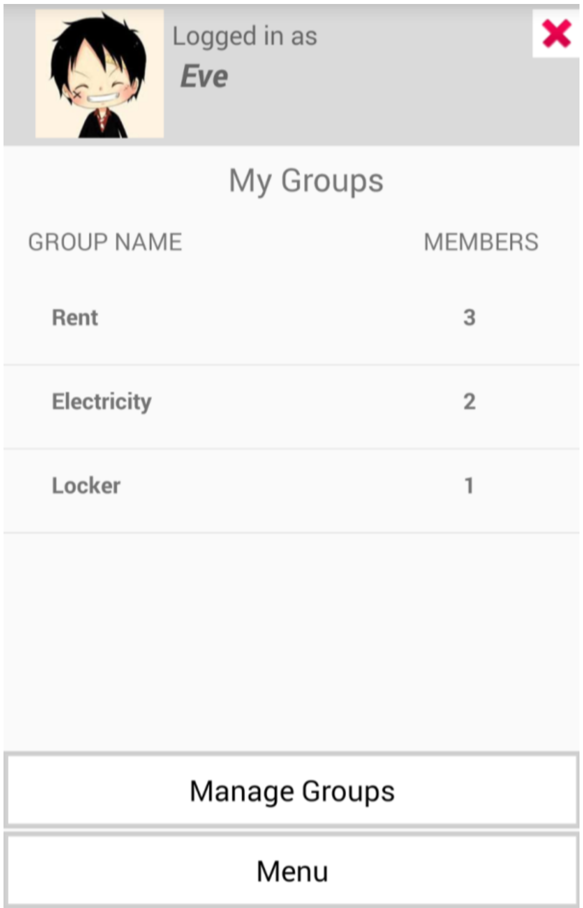
Create Group

Manage Profile

| LABEL | AMT (USD) | DUE DATE | GROUP |
|---|---|---|---|
| water | 50 | 2018-07-01 | testGroup |

Menu. This page provides you with options to check your group information (My Groups), to join an existing group (Join Group), to create a group (Create Group), and to manage your profile (Manage Profile); pressing on each of these buttons brings you to the corresponding page. This page also shows bill information for upcoming, unpaid bills (sorted chronologically). Clicking on a bill will redirect you to the Bill List of the group to which the bill belongs.

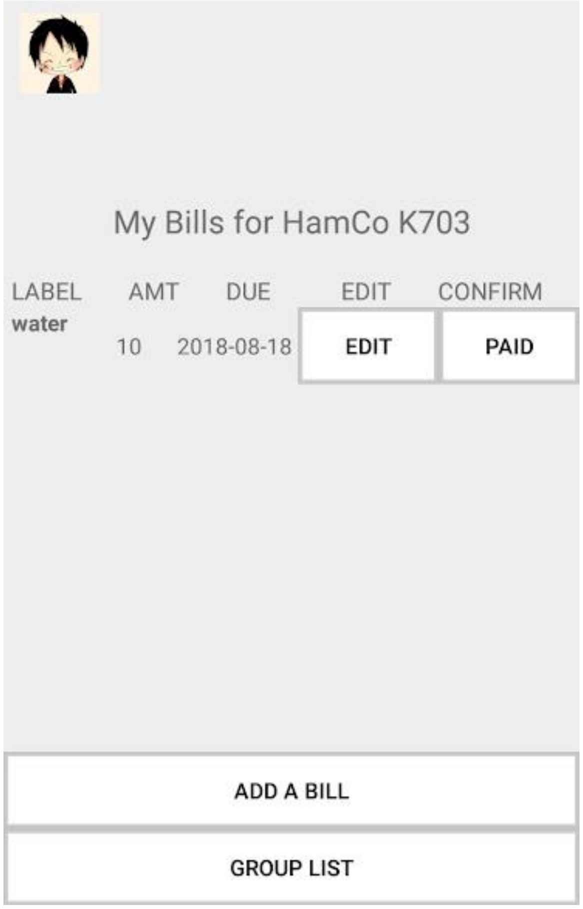
Press the red cross in the top-right corner to be signed out and brought back to the Splash page.


Logged in as
*Eve*

My Groups

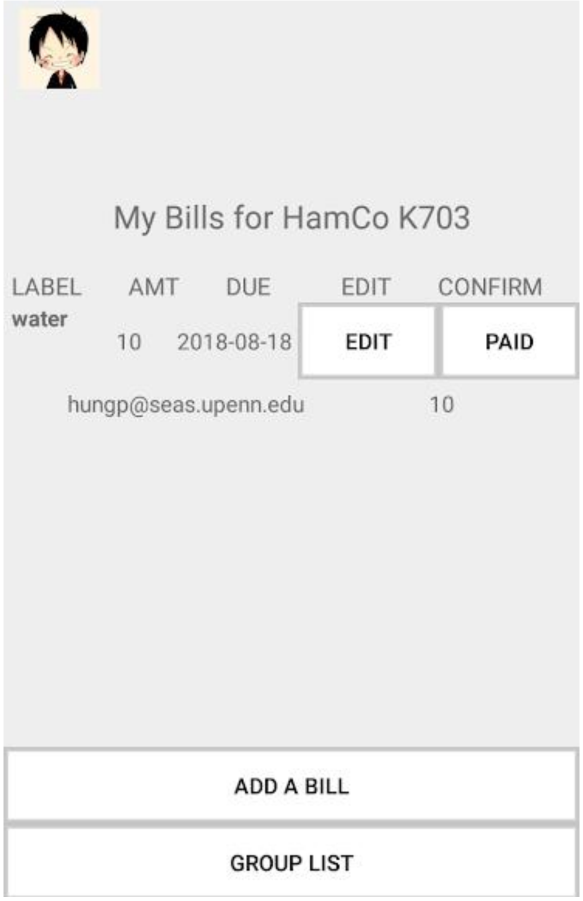
| GROUP NAME | MEMBERS |
|---|---|
| Rent | 3 |
| Electricity | 2 |
| Locker | 1 |

My Groups. This page lists the names of each group you are in, as well as the number of users in the group. Clicking on a group redirects you to the Bill List for that group. Click on [Manage Groups] to manage your groups, and click on [Menu] to be redirected to the menu page.


My Bills for HamCo K703

| LABEL | AMT | DUE | EDIT | CONFIRM |
|---|---|---|---|---|
| water | 10 | 2018-08-18 | EDIT | PAID |

Bill List. This page lists all the pending bills you have for a given group, with the following information: label, amount due, and due date. Clicking on any of the bills will reveal the other payers for that bill and the amount they owe (see next screenshot). Click on [Edit] to Edit Bill, and click on [Paid] to confirm payment for a bill, which removes this bill from this list. Additionally, you can [Add a Bill] or click on [Group List] to go back to My Groups.

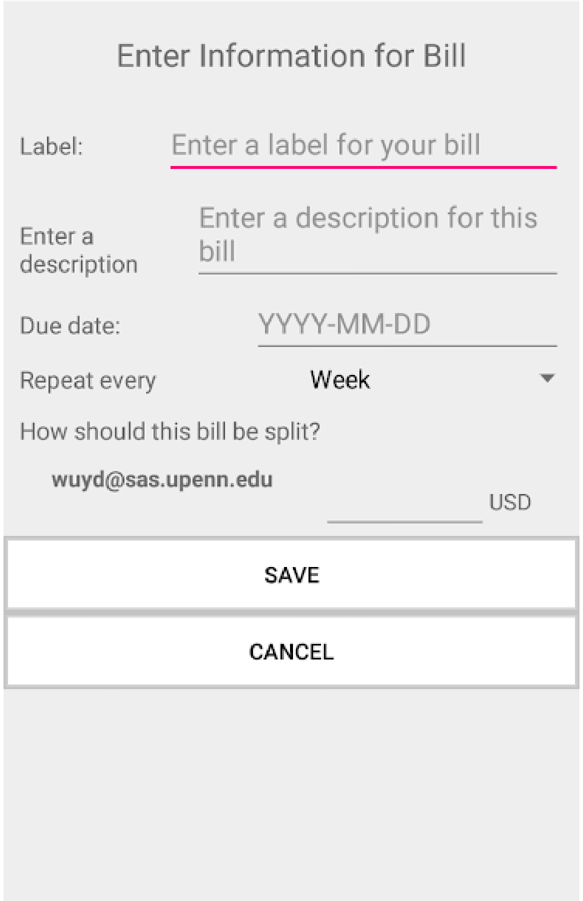


My Bills for HamCo K703

| LABEL | AMT | DUE | EDIT | CONFIRM |
|---|---|---|---|---|
| water | 10 | 2018-08-18 | EDIT | PAID |

hungp@seas.upenn.edu 10

<u>Bill List – Other Users.</u> Clicking on any of the bills will reveal the other payers for that bill and the amount they owe.



Enter Information for Bill

Label: Enter a label for your bill

Enter a description: Enter a description for this bill

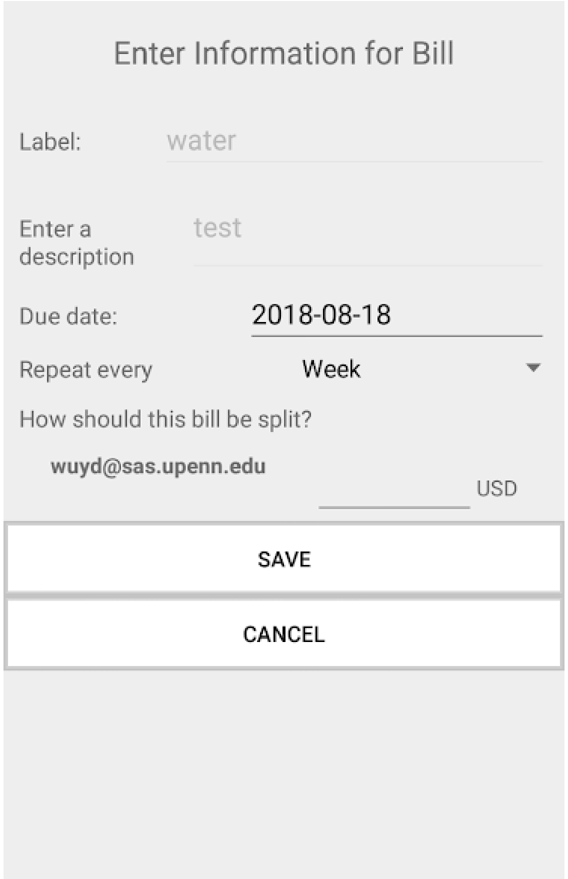
Due date: YYYY-MM-DD

Repeat every: Week

How should this bill be split?

wuyd@sas.upenn.edu USD

<u>Add a Bill.</u> By clicking on [Add a Bill] on the <u>Bill List</u> page, you can add information for a new bill: label, description, due date, and the split of amount owed between group members. Press [Save] to complete the transaction, or you can also [Cancel].



Enter Information for Bill

Label: water

Enter a description: test

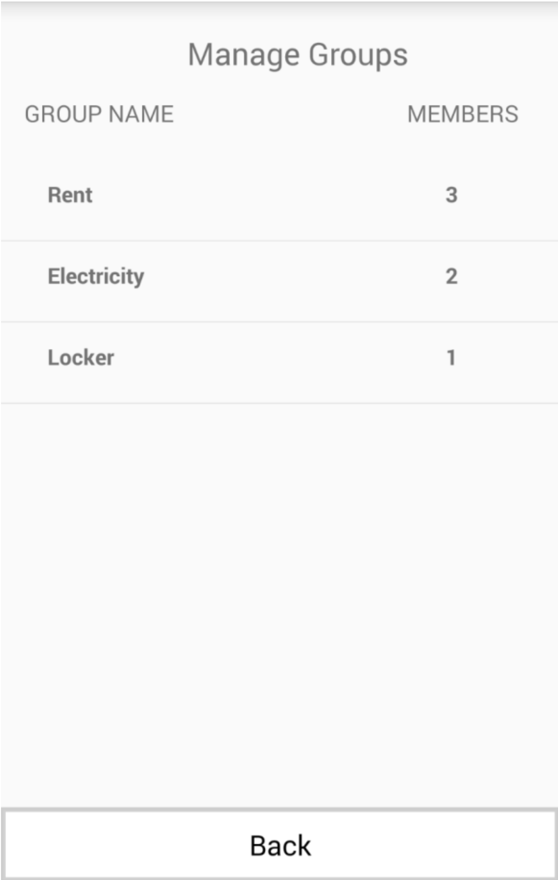
Due date: 2018-08-18

Repeat every: Week

How should this bill be split?

wuyd@sas.upenn.edu ____ USD

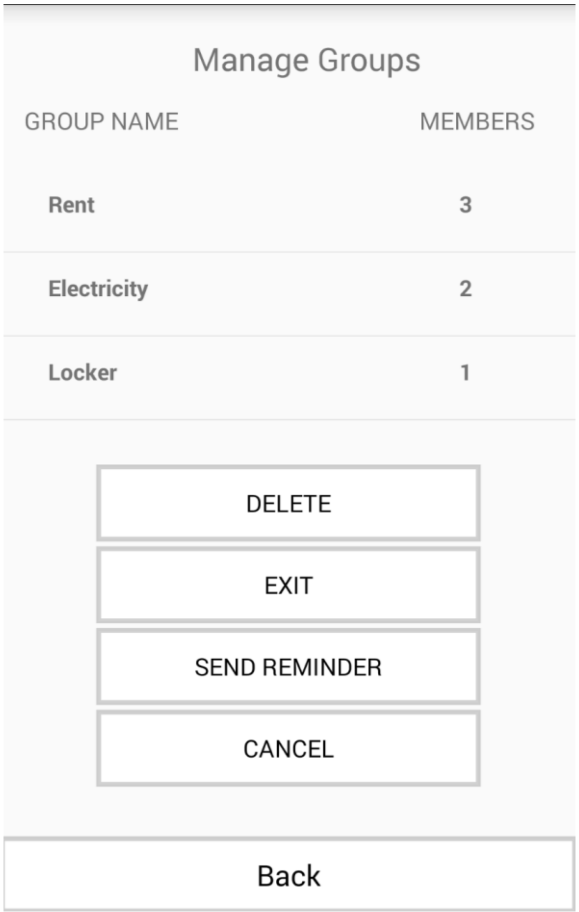
Edit Bill. Clicking on the [Edit] button on any bill from the Bill List page will allow you to edit the information for that bill (the due date and split method). Press [Save] to save the new information, or [Cancel].



Manage Groups

| GROUP NAME | MEMBERS |
|---|---|
| Rent | 3 |
| Electricity | 2 |
| Locker | 1 |

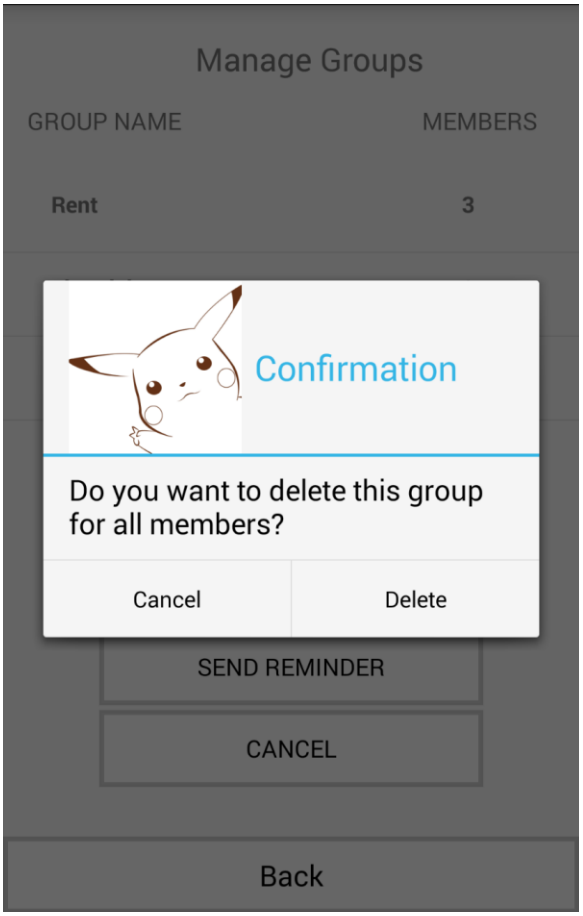
Manage Groups. You can reach this page by clicking [Manage Groups] on the My Groups page. From here, you can click on a group name in order to manage a given group (see next description).



Manage Groups

| GROUP NAME | MEMBERS |
|---|---|
| Rent | 3 |
| Electricity | 2 |
| Locker | 1 |

Manage Groups - Options. Upon clicking on a group, these buttons appear and present the options for managing the selected group (see the following screenshots and descriptions). Pressing on [Send Reminder] causes your phone's default email app to pop up, populated with group members' email addresses in the "To" field and a default reminder message. Pressing on [Cancel] closes the pop-up.



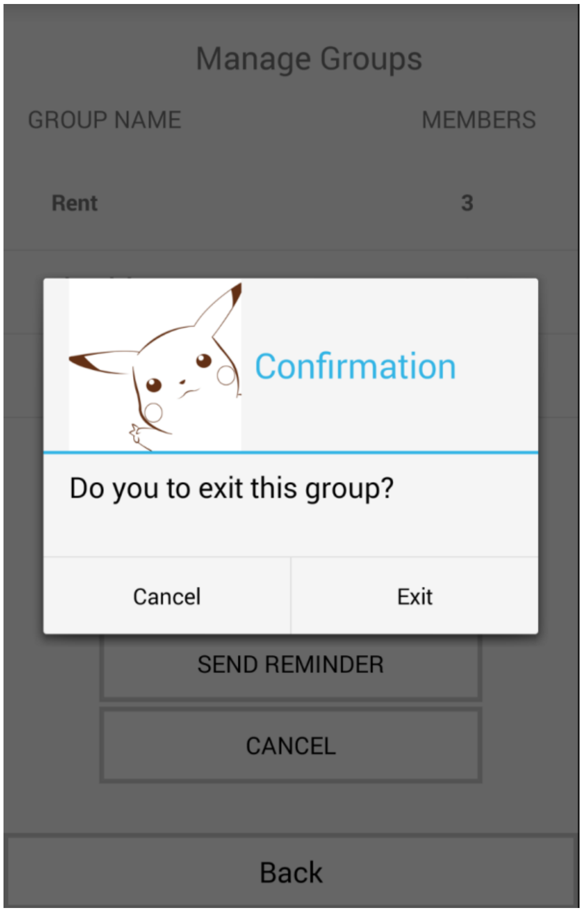
Manage Groups

| GROUP NAME | MEMBERS |
|---|---|
| Rent | 3 |

Confirmation

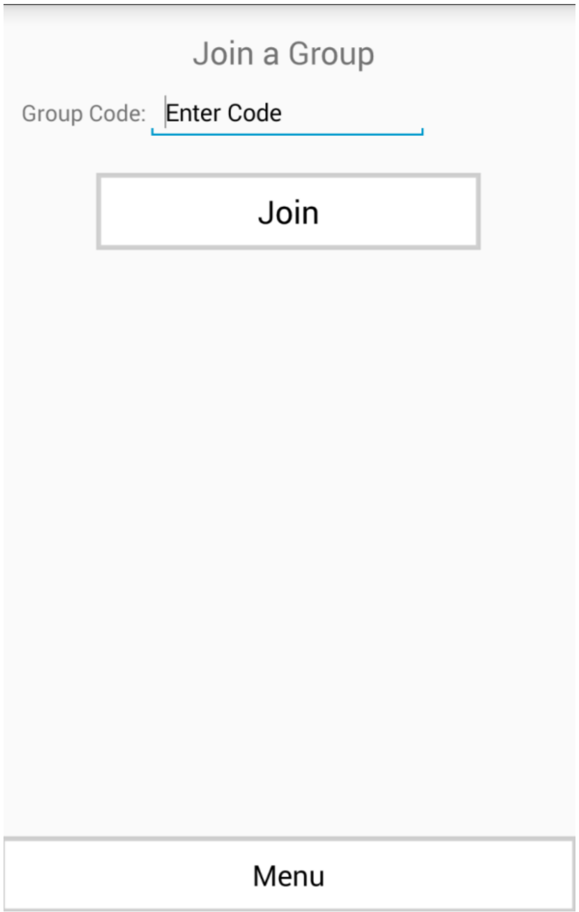
Do you want to delete this group for all members?

Delete Group Confirmation. When requesting to delete a group, a confirmation window verifies the request before carrying it out. After a group has been deleted, each group member is notified of it via email; this email lists the member's unpaid bills for this deleted group.

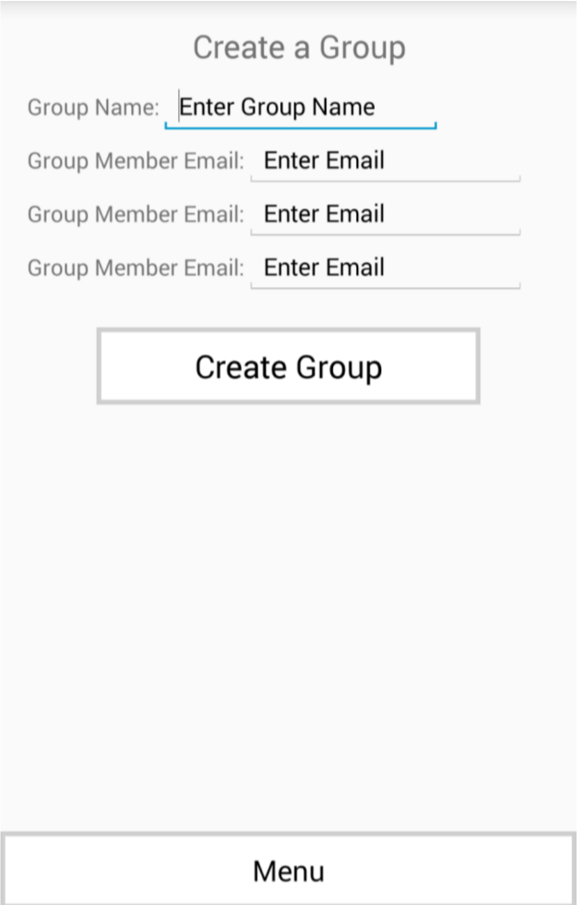


Do you to exit this group?

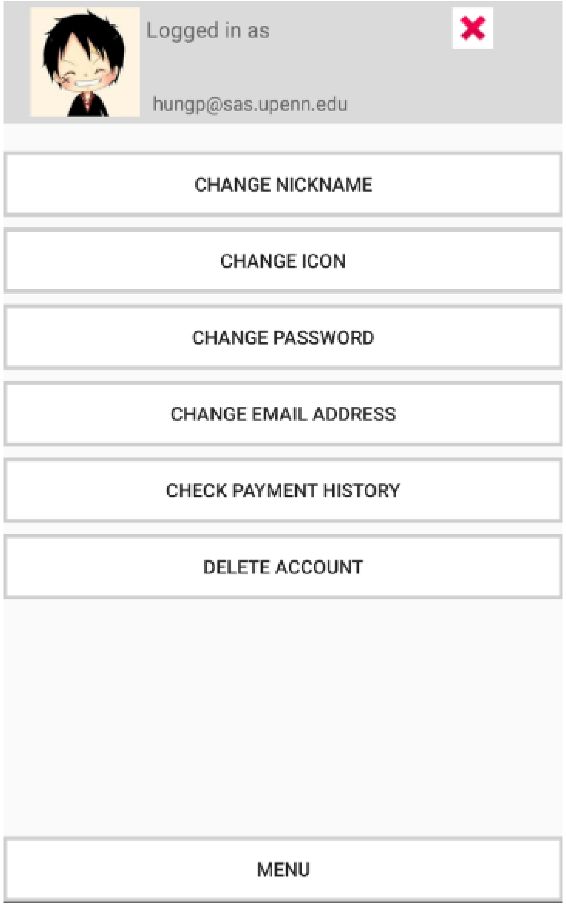
Exit Group Confirmation. When requesting to exit a group, a confirmation window verifies the request before carrying it out. After exiting a group, you will be notified of all your unpaid bills for that group via email.



Join Group. This page is accessed by clicking on [Join Group] on the Menu page. If you have been invited to join a group, you would have been emailed a six character code that expires after one week. After entering that code here, you will be added to the group.

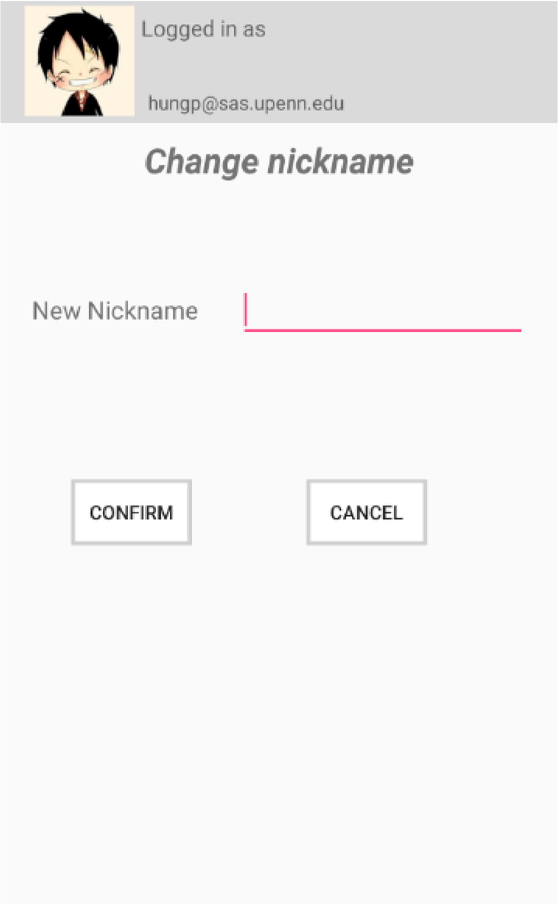

Create a Group
Group Name: Enter Group Name
Group Member Email: Enter Email
Group Member Email: Enter Email
Group Member Email: Enter Email
Create Group
Menu

Create Group. You can reach this page by clicking [Create Group] from the Menu page. You can give the group a name and list up to three email addresses to invite to the group. Invitees do not need to have an account already. The invitees will each receive a six character code—that expires after one week—via email, and they can use that code to be added to the group through whichever account they wish. After creating the group, you will be redirected to your My Groups page.


Logged in as

hungp@sas.upenn.edu

Manage Profile. This page provides you with profile management options to change your nickname (Change Nickname), icon (Change Icon), password (Change Password), and email address (Change Email). You can also check your payment history (Payment History), delete your account (Delete Account), or go back to the Menu page.

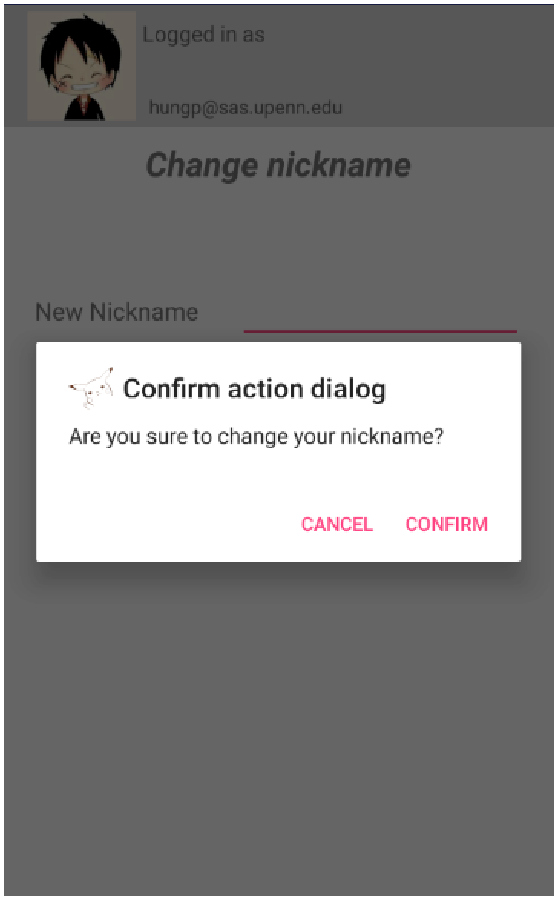


Logged in as

hungp@sas.upenn.edu

Change nickname

Change Nickname. You can input your new nickname and then click on [Confirm]. A confirm dialog will then pop up to let you double check your changes (see next).



Logged in as

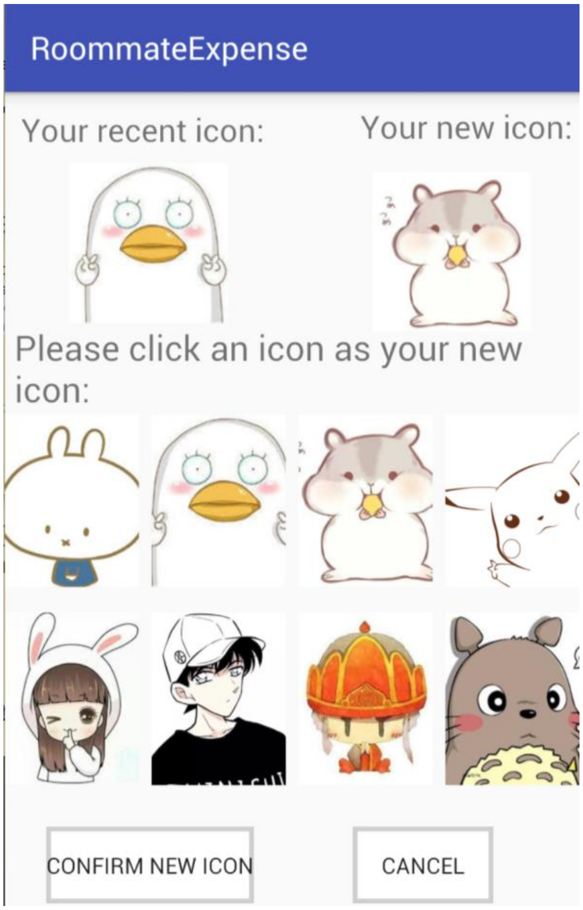
hungp@sas.upenn.edu

Change nickname

Confirm action dialog

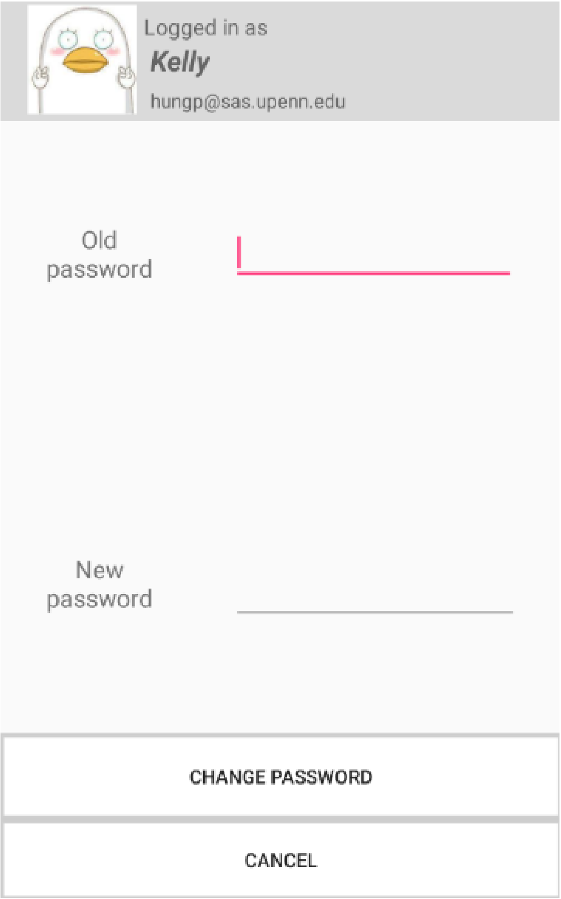
Are you sure to change your nickname?

Nickname Confirm Dialog. Check your action. If [Confirm] is clicked, your change will be saved, a success message will be shown, and you will be redirected to the Manage Profile page. Otherwise, if [Cancel] is clicked, you will be brought back to the Manage Profile page without having a new nickname.



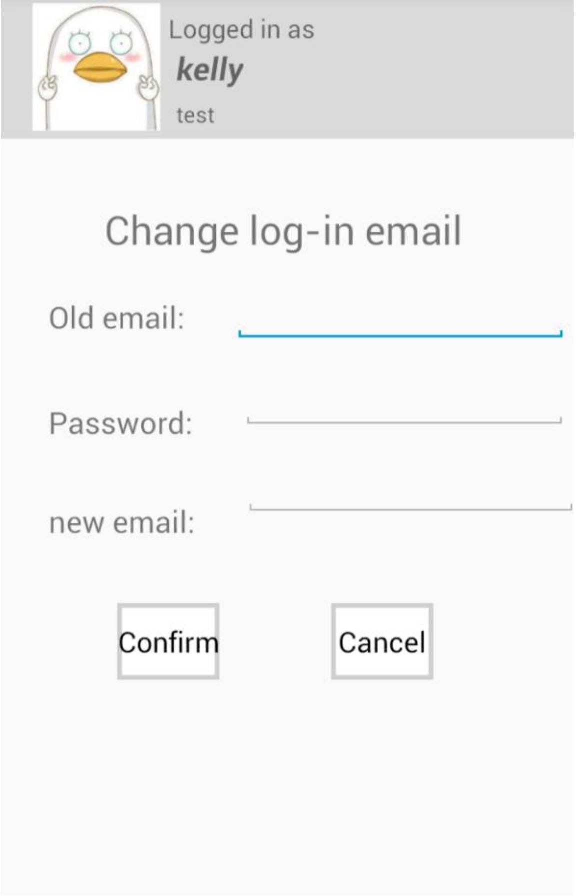
Please click an icon as your new icon:

Change Icon. You can see your current icon in the top-left corner (your recent icon) and click on available icons to set your avatar to a new picture. The latest clicked icon image will then be shown in the top-right corner (your new icon).



Logged in as
Kelly
hungp@sas.upenn.edu

Change Password after Logging In. Enter your old password and your desired new password, and then press [Change Password]. If the combination is valid (i.e. if you accurately entered your old password), then you will be redirected to the Login page, where you can login with your new credentials. However, if you entered the wrong old password, you will be told that the entered old password is incorrect. Press [Cancel] if you no longer want to change your password, and you will be redirected to your profile page.



Logged in as
kelly
test

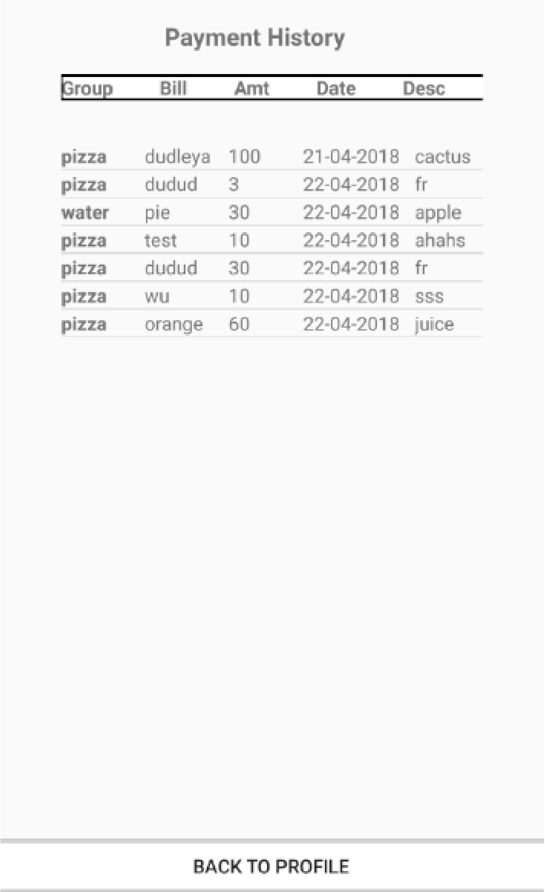
Change log-in email

Old email:

Password:

new email:

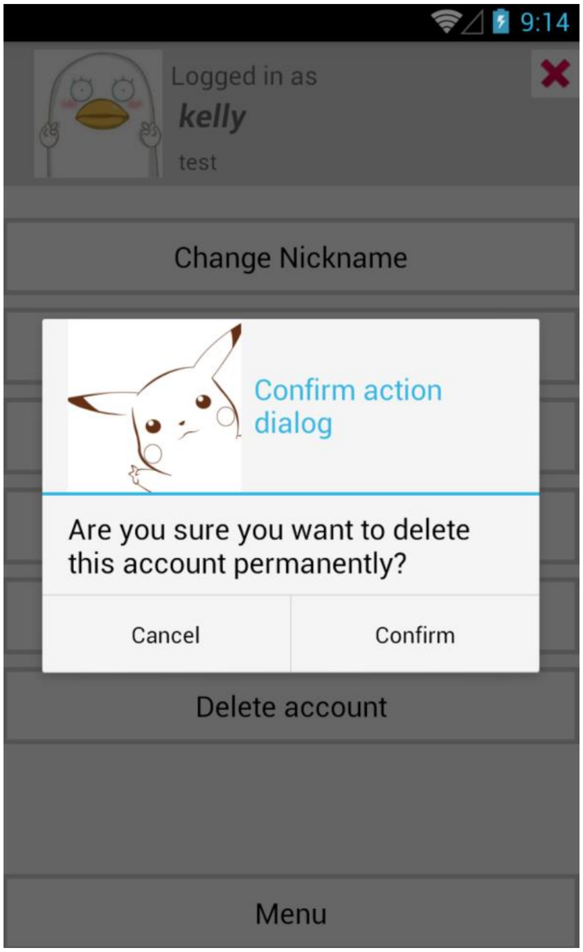
Change Email. Input your current email address, your password, and your new email address. If the old email and password is correct for the account, and the new email is not already associated with an account, you will be redirected to the Login page with the change saved. If the old email address or password is wrong, you will be told so.



Payment History

| Group | Bill | Amt | Date | Desc |
|---|---|---|---|---|
| pizza | dudleya | 100 | 21-04-2018 | cactus |
| pizza | dudud | 3 | 22-04-2018 | fr |
| water | pie | 30 | 22-04-2018 | apple |
| pizza | test | 10 | 22-04-2018 | ahahs |
| pizza | dudud | 30 | 22-04-2018 | fr |
| pizza | wu | 10 | 22-04-2018 | sss |
| pizza | orange | 60 | 22-04-2018 | juice |

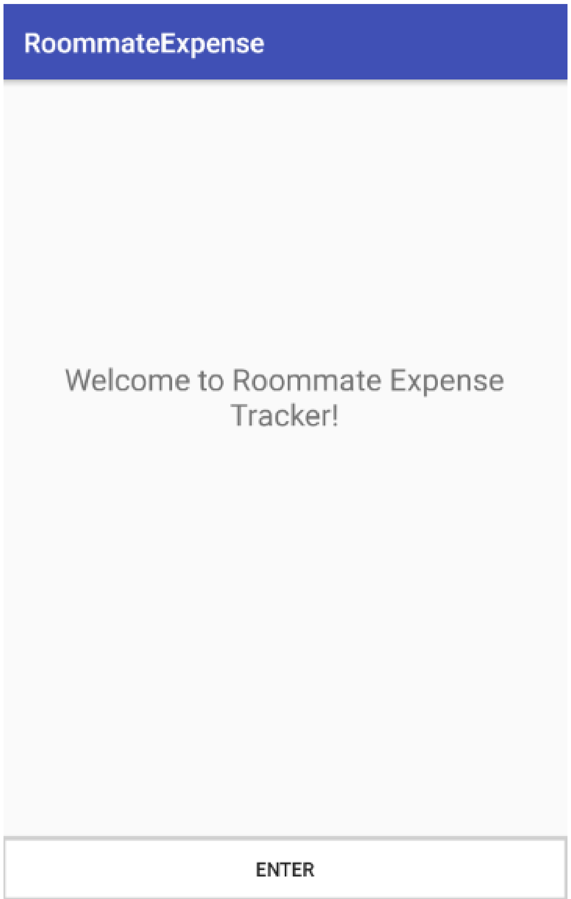
Payment History. See your payment history here. These are all the bills for which you have confirmed payment; information includes the group from where the bill came, the bill's name, the amount, the date it was paid (i.e. when you confirmed payment), and the description. Press [Back to Profile] to navigate back to your profile page.



Are you sure you want to delete this account permanently?

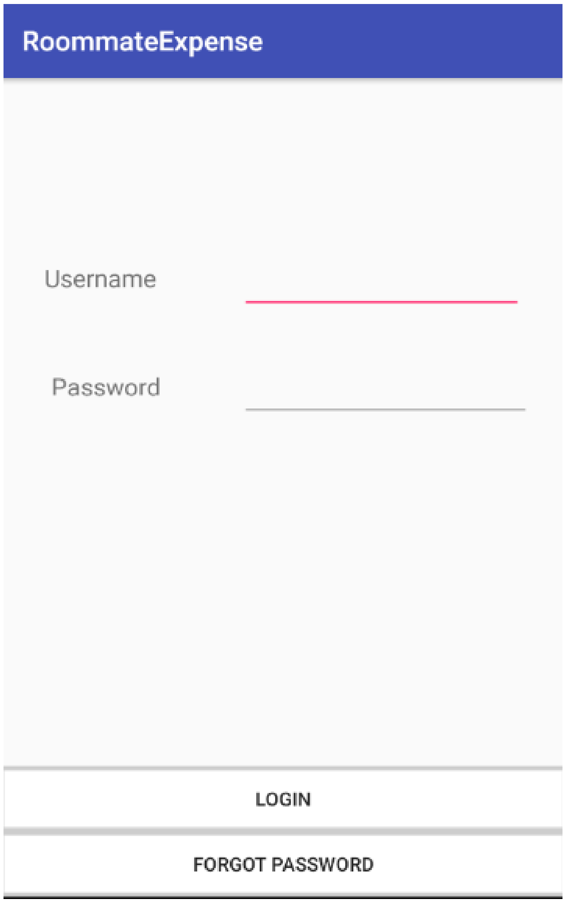
Delete Account. After clicking on [Delete Account] on the Manage Profile page, a confirmation box will pop up to let you double check your action. If you click on [Confirm], your account will be deleted, and you will be redirected to the Login page. If you click on [Cancel], you will be redirected to the Manage Profile page.

## 2. Technical Manual



Welcome to Roommate Expense Tracker!

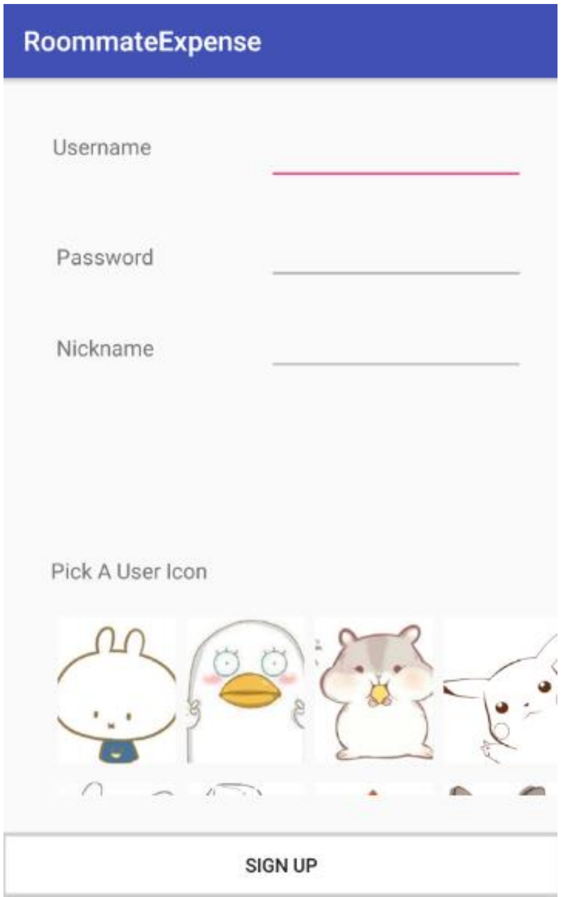
MainActivity.java



UserPkg/LoginActivity.java

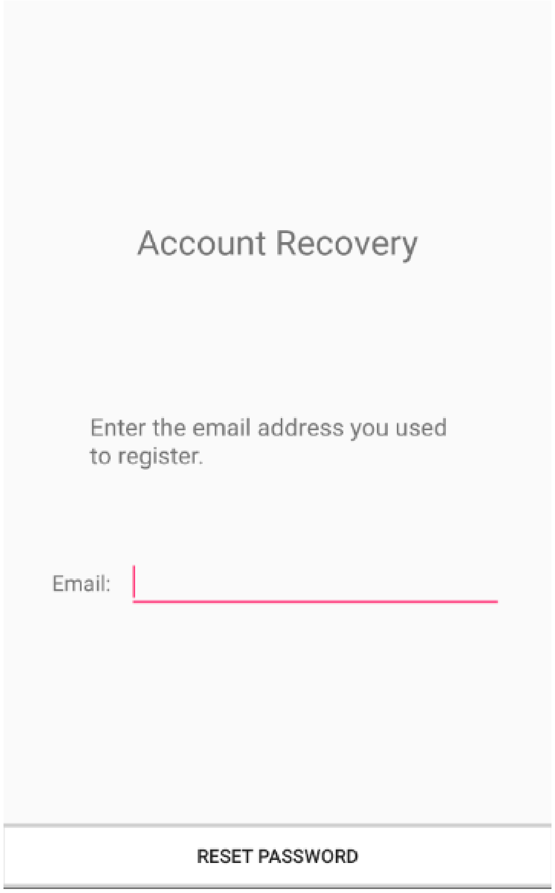
Helper classes:
1.) DB/DBQueries.java – Java library for database queries (in SQL) [1]

[1] Used throughout the entire app.



UserPkg/SignupActivity.java

Helper classes:
1.) UserPkg/ImageAdapter.java – allows image icons to be shown through a GridView

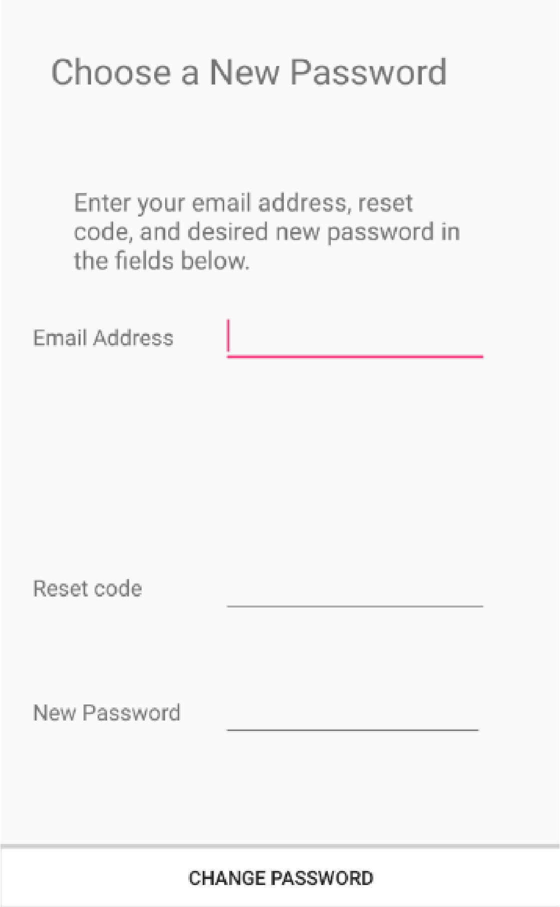


Account Recovery

Enter the email address you used to register.

UserPkg/ForgotPasswordActivity.java

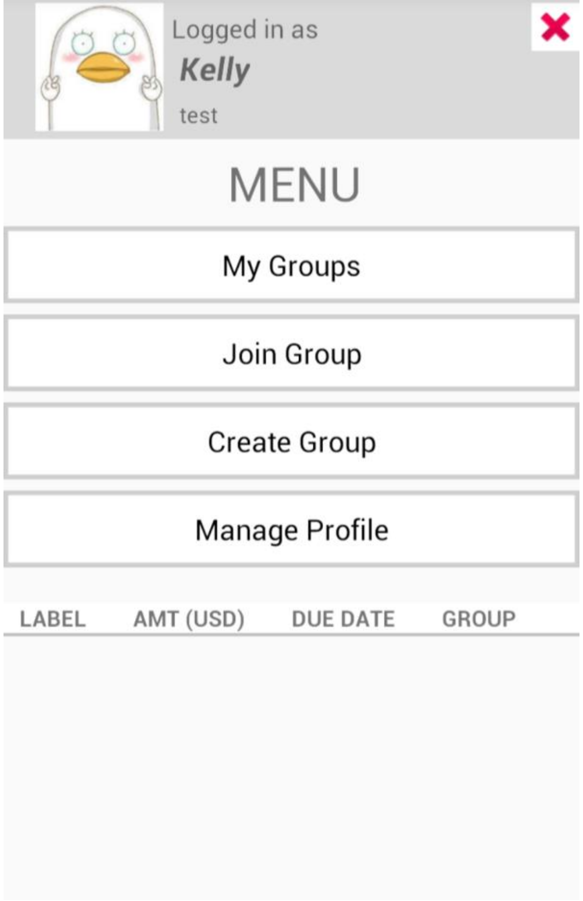
Helper classes:
1.) EmailPkg/GMailSender.java – automates sending emails from admin account to target email address. See login credentials in the “Credentials” section, below



Choose a New Password

Enter your email address, reset code, and desired new password in the fields below.

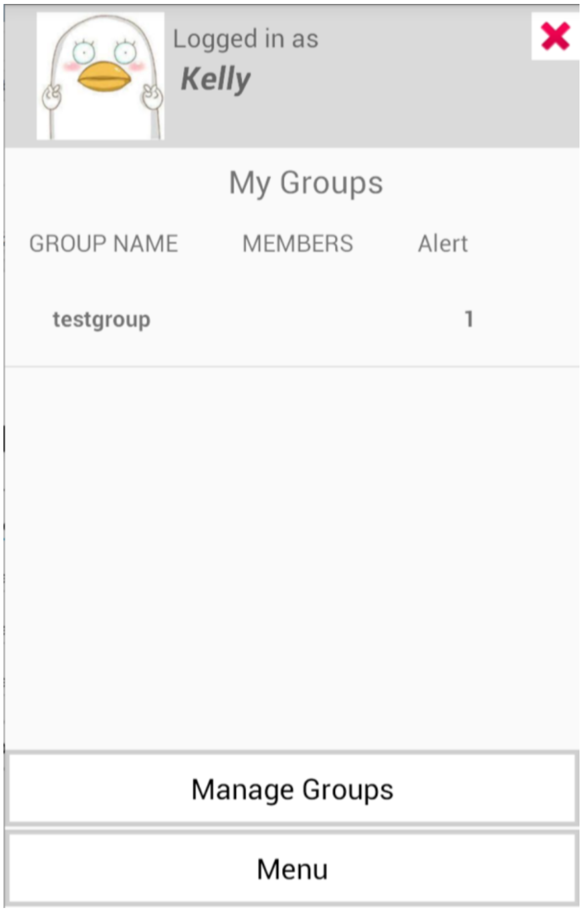
UserPkg/ResetPasswordActivity.java



Logged in as
Kelly
test

MenuActivity.java

Helper classes:
1) BillPkg/UpcomingBillAdapter.java – allows bills information to be shown through a ListView; this uses BillPkg/Bill.java as a POJO to store bill information



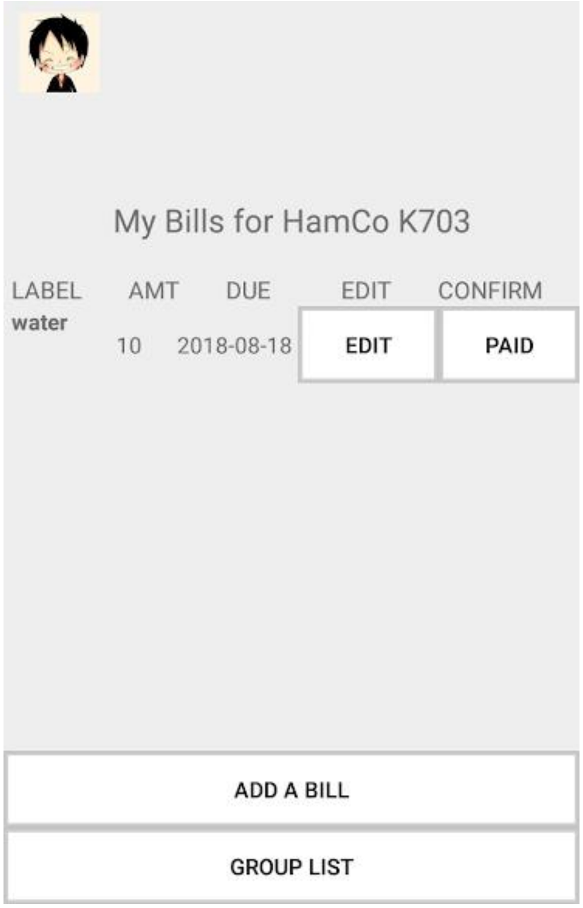
Logged in as
*Kelly*

My Groups

| GROUP NAME | MEMBERS | Alert |
|---|---|---|
| testgroup | | 1 |

GroupPkg/GroupListActivity.java

Helper classes:
1.) GroupPkg/Group.java – stores all user group data from database
2.) GroupPkg/GroupAdapter.java – allows Group information to be listed in a LinearLayout

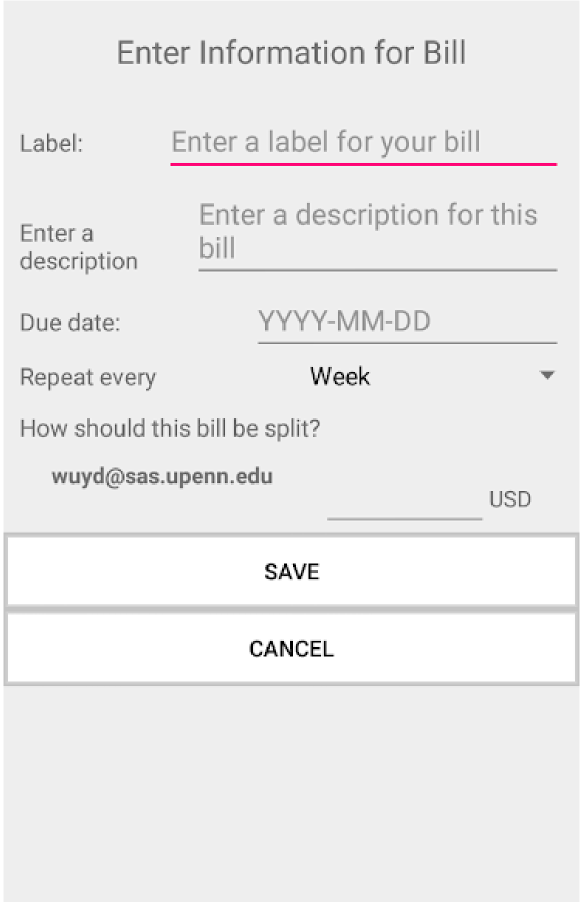


My Bills for HamCo K703

| LABEL | AMT | DUE | EDIT | CONFIRM |
|---|---|---|---|---|
| water | 10 | 2018-08-18 | EDIT | PAID |

BillPkg/BillListActivity.java Loads the list of bills within a group for a user.

Helper classes:
1.) BillPkg/BillAdapter.java – bridges the ListView and data of all the bills
2.) BillPkg/Bill.java – POJO that BillAdapter uses to store information about a bill



Enter Information for Bill

Label: Enter a label for your bill

Enter a description: Enter a description for this bill

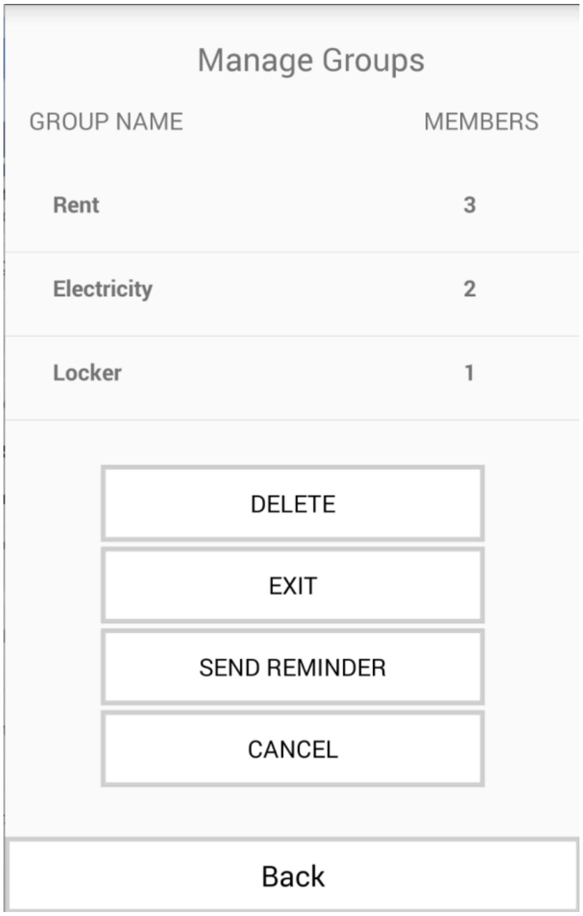
Due date: YYYY-MM-DD

Repeat every: Week

How should this bill be split?

wuyd@sas.upenn.edu USD

BillPkg/AddBillActivity.java Allows user to enter information about a bill for a group. If the user had selected “Add a Bill” from the bill list page, the form allows user to enter the label, a brief description, due date, and split method amongst all members of the group (i.e. how much each group member should pay for this bill). If the user selected “Edit” from one of the bills in the bill list page, the form will only allow the user to enter the due date and split method.

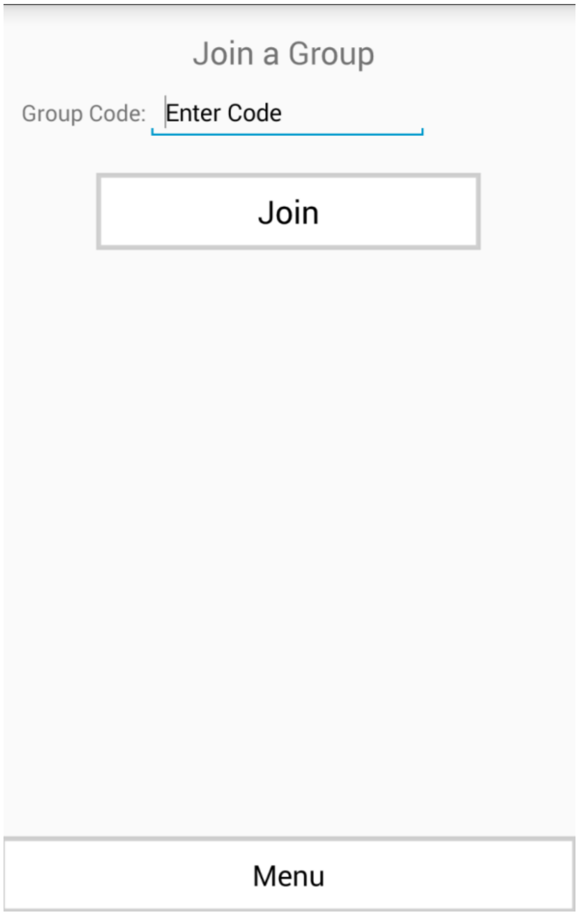


Manage Groups

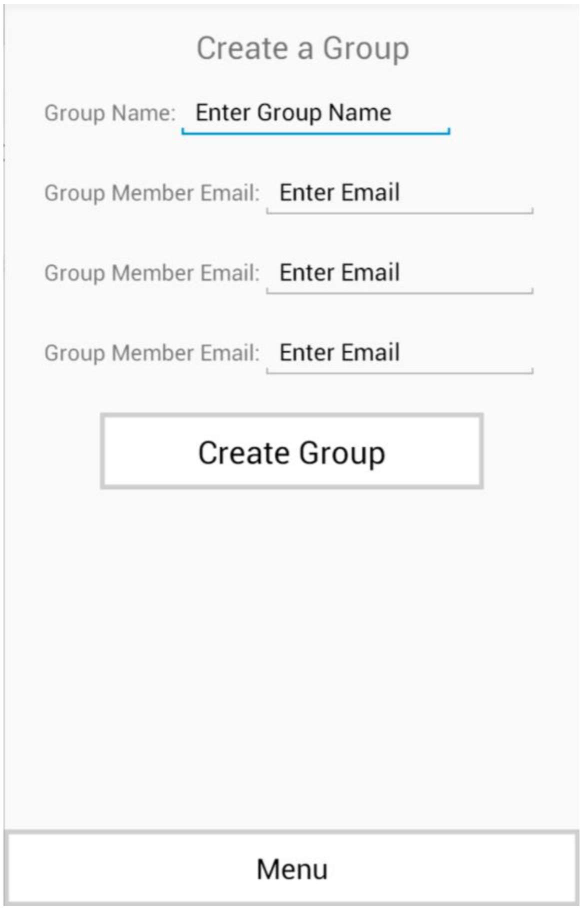
| GROUP NAME | MEMBERS |
|---|---|
| Rent | 3 |
| Electricity | 2 |
| Locker | 1 |

GroupPkg/GroupManageActivity.java

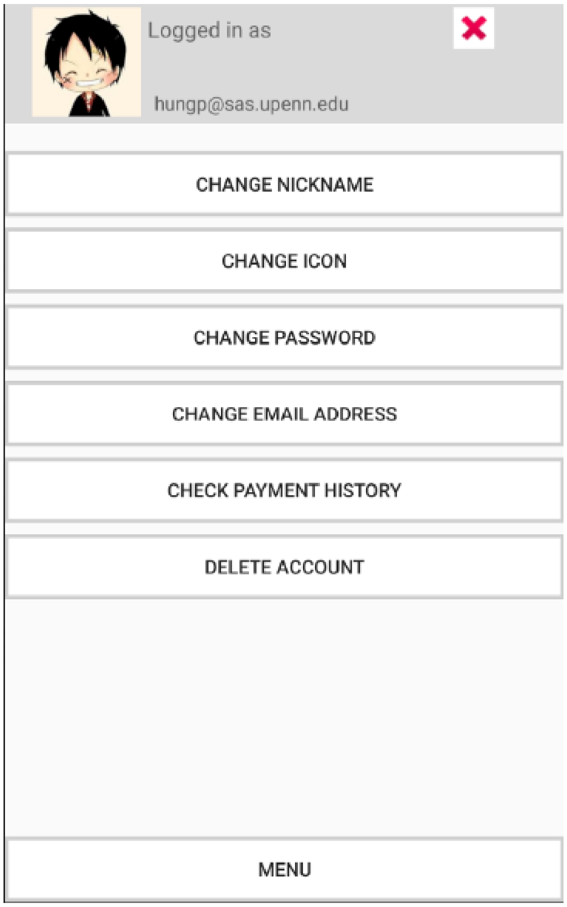
Helper classes:
1.) GroupPkg/GroupAdapter.java – allows Group information to be listed in a LinearLayout



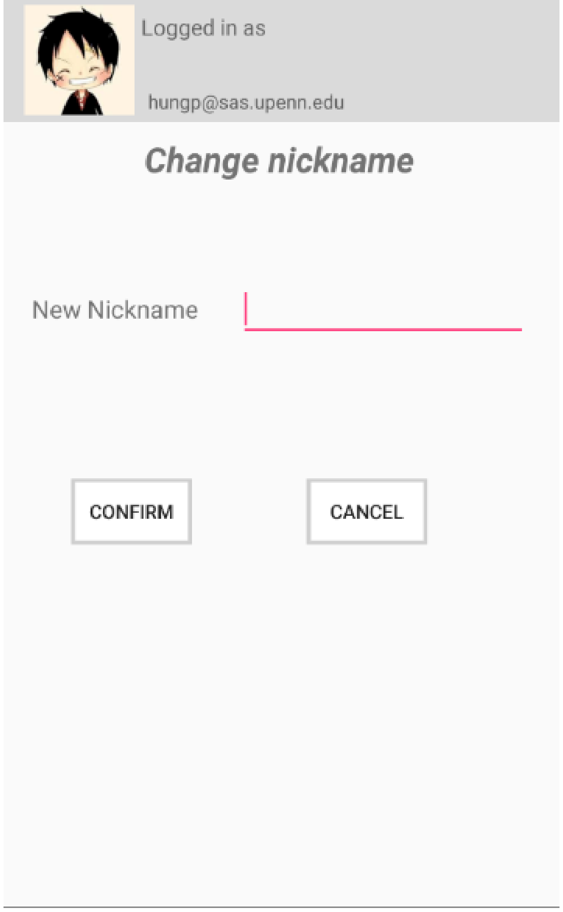
GroupPkg/JoinGroupActivity.java



GroupPkg/CreateActivity.java


Logged in as
hungp@sas.upenn.edu

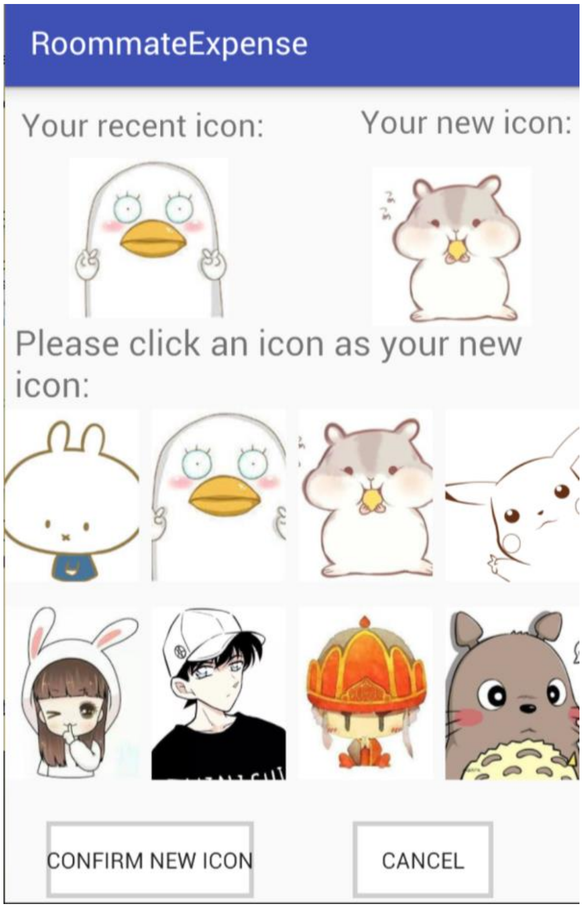
UserPkg/ProfileActivity.java


Logged in as
hungp@sas.upenn.edu

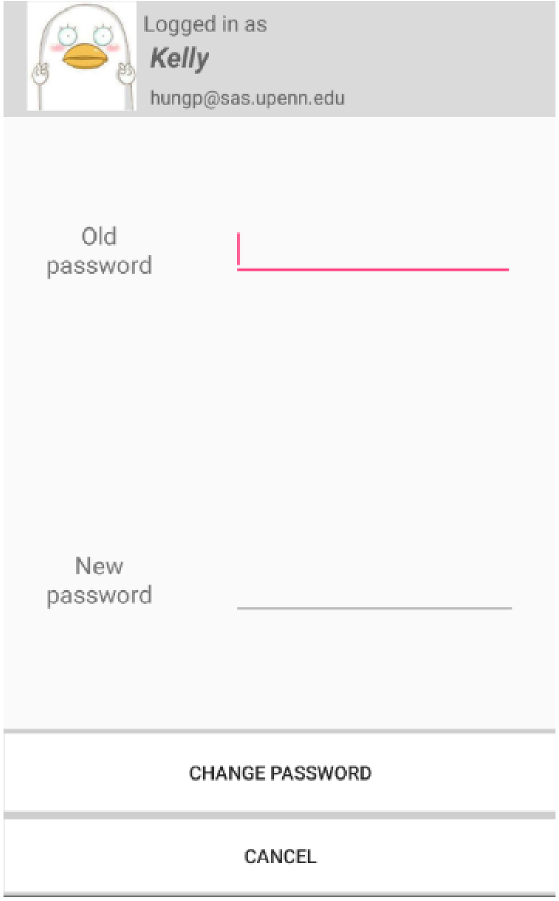
Change nickname

New Nickname

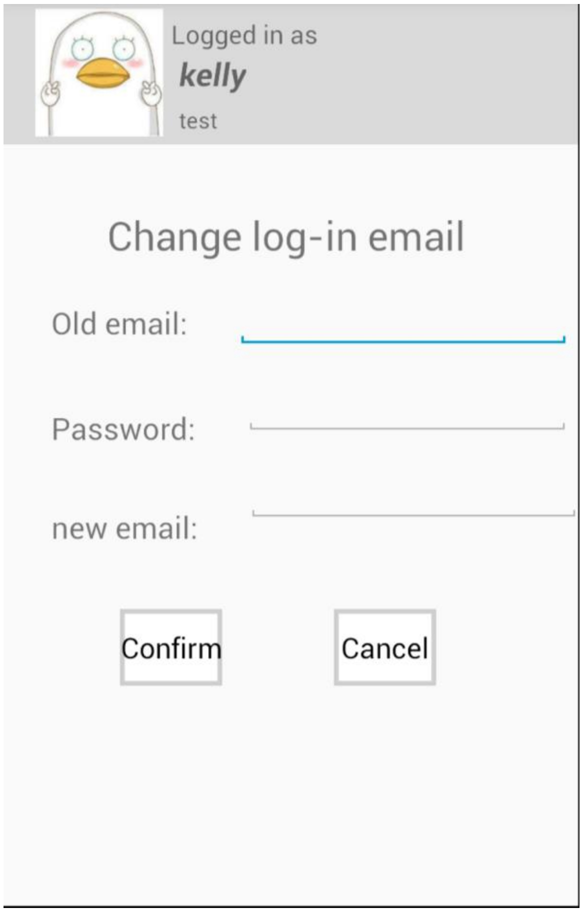
UserPkg/NicknameChangeActivity.java



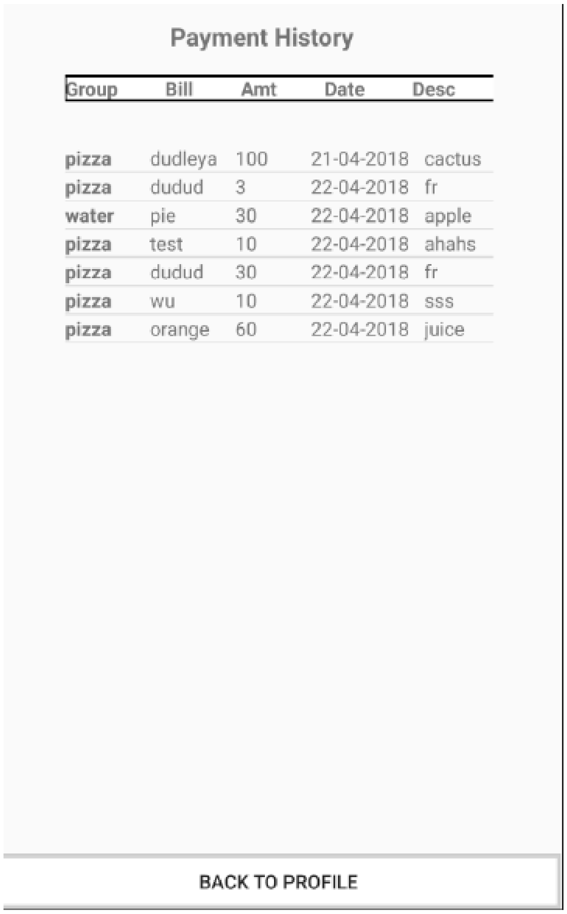
UserPkg/IconChangeActivity.java



UserPkg/PasswordChangeActivity.java



UserPkg/EmailChangeActivity.java



| Group | Bill | Amt | Date | Desc |
|---|---|---|---|---|
| pizza | dudleya | 100 | 21-04-2018 | cactus |
| pizza | dudud | 3 | 22-04-2018 | fr |
| water | pie | 30 | 22-04-2018 | apple |
| pizza | test | 10 | 22-04-2018 | ahahs |
| pizza | dudud | 30 | 22-04-2018 | fr |
| pizza | wu | 10 | 22-04-2018 | sss |
| pizza | orange | 60 | 22-04-2018 | juice |

UserPkg/PaymentHistoryActivity.java

Helper classes:
1.) BillPkg/OldBill.java – POJO for information on old bills, i.e., bills that the user has confirmed payment for
2.) BillPkg/OldBillAdapter.java – adapter so that old bills can easily be displayed in a list

## Database Information[2]

Schemas

1.) Users (user_id VARCHAR(255), password VARCHAR(25), nickname VARCHAR(50), icon_id VARCHAR(30), reset_code CHAR(6), expiry_date CHAR(10), PRIMARY KEY (user_id))

- This table stores all the relevant user profile information, including the email address (user_id), password, nickname, a string identifier for the icon photo chosen, and, if necessary, an expiring code for if the user requests a password reset after forgetting his/her password.

2.) Groups (group_id VARCHAR(10), group_name VARCHAR(20), PRIMARY KEY(group_id))

- This table associates the unique, randomly generated group_id with the user-chosen group name.

3.) Bills (bill_id VARCHAR(20), bill_name VARCHAR(20), user_id VARCHAR(255), amount DOUBLE, due_date CHAR(10), description VARCHAR(255), FOREIGN KEY (user_id) REFERENCES Users (user_id), PRIMARY KEY (bill_id))

- This table stores all relevant information for a bill, including its randomly generated unique id, the user-chosen name, the user it's assigned to, how much is due, its due date, and a description of the bill.

4.) UserGroups (user_id VARCHAR(255), group_id VARCHAR(20), FOREIGN KEY (user_id) REFERENCES Users (user_id), FOREIGN KEY (group_id) REFERENCES Groups (group_id))

- This table associates each user with each group he/she is in.

5.) GroupBills (group_id VARCHAR(20), bill_id VARCHAR(20), FOREIGN KEY (group_id) REFERENCES Groups (group_id), FOREIGN KEY (bill_id) REFERENCES Bills (bill_id))

- This table associates each group with each bill assigned within it.

6.) GroupCodes (invite_code CHAR(6), expiry_date CHAR(10), group_id VARCHAR(10), FOREIGN KEY (group_id) REFERENCES Groups (group_id), PRIMARY KEY(invite_code))

- This table associates a randomly generated, expiring invitation code with the group it allows a user to join.

7.) OldBills (bill_id VARCHAR(10), bill_name VARCHAR(20), group_name VARCHAR(20), amount DOUBLE, description VARCHAR(255), date_paid CHAR(10), PRIMARY KEY (bill_id))

- This table stores all the relevant information for a paid bill, including its name, the group it was paid from within, the amount paid, a description, and the date paid.

8.) OldUserBills (user_id VARHCAR(255), bill_id VARCHAR(10), FOREIGN KEY (user_id) REFERENCES Users (user_id), FOREIGN KEY (bill_id) REFERENCES OldBills (bill_id))

- This table associates each paid bill with the user who paid it.

[2] See the DB Package

## Database Login Credentials

Note: DBConstants.java holds this information, and DBConnect.java uses the information provided by DBConstants.java to return a connection object used by DBQueries.java.

instance ID: cis350project
port: 3306
DB name: CIS350APP
username: roommatespending
password: tracker350
host name: cis350project.crthvh2wxus6.us-east-2.rds.amazonaws.com

Run this to access the database:

```
mysql --local-infile -h cis350project.crthvh2wxus6.us-east-2.rds.amazonaws.com -P 3306 -u roommatespending CIS350APP -ptracker350
```

## Database Queries Library (from DBQueries.java)

### MISC FUNCTIONS

[*helper*] **void nullEmail(String email)** throws IllegalArgumentException if email is null

[*helper*] **void nullPassword(String p)** throws IllegalArgumentException if p is null

[*helper*] **void nullGroup(String group_id)** throws IllegalArgumentException if group_id is null

[*helper*] **void validDueDate(String date)** throws an IllegalArgumentException if date is null, isn’t 10 characters, or if characters 4 and 6 aren’t ‘-“

[*helper*] **void nullGroupName(String name)** throws an IllegalArgumentException if name is null or >20 characters

[*helper*] **String generateCode(int length)** creates a random *length*-character string using the English alphabet and 0-9

### USER FUNCTIONS

**int login(String email, String password)** handles login functionality. Returns 0 if email doesn’t exist in the database, 1 if password was incorrect, 2 if successful

[*helper*] **boolean userExists(String email)** returns whether the email exists in the database

**boolean enteredCorrectPassword(String email, String oldPassword)** returns whether the given password matches the user’s password

**boolean signUp(String email, String password)** handles signing a new user up to the app, returns whether it was successful

**String changeEmail(String oldEmail, String newEmail)** handles updating a user’s email and propagating changes through the database; returns whether the update was successful

**String forgotPassword(String email)** generates, stores, & returns a reset code for the user in the database, and sets the expiration date to 1 week from the current day.

**int allowReset(String email, String code)** returns 2 if the user should be given permission to update his/her password, 1 if the code already expired, 0 if the code is incorrect

**boolean resetPassword(String user, String password)** handles resetting the user's password with the new one, returns whether the update was successful

**boolean setNickname(String user, String nickname)** handles setting the user's nickname, returns whether the update was successful

**String getNickname(String user)** returns the user's nickname, null if it was never set

**boolean setIcon(String user, String icon)** handles setting the user's icon, returns whether the update was successful

**String getIcon(String user)** returns the user's icon, null if it hasn't been set

**boolean deleteAccount(String email)** handles safe deletion of a user's account, and returns whether it was successful

### BILL FUNCTIONS

[*helper*] **boolean billExists(String bill_id)** returns whether bill_id is being used in the database

**ResultSet userBills(String email)** returns all bill info stored for each bill assigned to the user, null if none

**ResultSet userBillsInGroup(String email, String group_id)** returns all bill info stored for each bill assigned to the user in the given group, null if none

**boolean addBill(String user, String group, String name, double amt, String date, String desc)** handles functionality for adding a bill for a given user_id and group_id, returns whether the operation was successful

**boolean deleteBill(String user_id, String group_id, String bill_id)** handles deleting a bill, returns whether the operation was successful

**boolean changeDueDate(String bill_id, String due_date)** handles changing the due date for the specified bill, returns whether it was successful

**boolean changeAmount(String bill_id, double amt)** handles changing the bill's amount value for the specified bill, returns whether it was successful

**String getGroupIdForBill(String bill_id)** returns the group_id associated with that bill_id, null if it doesn't exist

### OLD BILL FUNCTIONS

[*helper*] **boolean oldBillExists(String bill_id)** returns whether bill_id is being used in the database

**ResultSet getOldBills(String user_id)** returns a result set for all of that user's old bills, null if the query fails

[*helper*] **boolean addOldBill(String bill_id, String user_id, String group_name, double amount, String description)** adds the data into the table of old bills

**GROUP FUNCTIONS**

[*helper*] **boolean groupExists(String group_id)** returns whether the group_id is being used in the database

**ResultSet userGroups(String email)** returns all the groups a user is in, null if none

[*helper*] **boolean codeInUse(String code)** checks and returns whether that group invite code is being used in the database already

**String getInviteCode(String group_id)** generates, stores, and returns an invite code for the group. The code expires after one week. Must generate a new code for each user.

**String getGroupForCode(String code)** returns the group_id associated with that invite code, null if the code doesn't exist in the database

**ArrayList<String> getGroupMembers(String group_id)** returns an array list of all the email addresses of the users of a given group. The list is empty if the group doesn't exist.

**String getGroupName(String group_id)** returns the name associated with the given group_id, an empty string if the group_id doesn't exist in the database

[*helper*] **ResultSet groupBills(String group_id)** returns all bill info stored for each bill in a group, null if none

**boolean createGroup(String user_id, String group_name)** handles creating a group for the user with the given name, returns whether it was successful

**int addUserToGroup(String user_id, String code)** returns 0 if the code was wrong, 1 if the code had expired, 2 if the user doesn't exist in the database, 3 if the addition was successful

**boolean leaveGroup(String user_id, String group_id)** handles a safe removal of a user from a group, returns whether it was successful

**boolean deleteGroup(String group_id)** handles safe deletion of a group, returns whether it was successful

**int groupParticipation(String group_id)** returns the number of users in a group, -1 if the group doesn't exist

## Email Credentials[3]

Note: GMailSender.java provides automated email (must be a gmail account; that is, mail host is smtp.gmail.com) sending functionality given an account and password, and JSSEProvider.java provides security permissions so SSL content can be handled. The code in both these classes is adapted from a tutorial on OodlesTechnologies: *http://www.oodlestechnologies.com/blogs/Send-Mail-in-Android-without-Using-Intent*

GMail account and password:
Account: roommatespendingtracker@gmail.com
Password: cis350s18

[3] See EmailPkg