Christopher Nishimura
Willard Peralta

I.O.U: The Restaurant Bill Tracker - ICS466 Project Part 4

Design

The Service

Customers have the option to split a check at a restaurant. This means that a customer can ask the waiter to separate each person's order so that each person only pays for what they got, rather than putting all the orders on one check and having one person pay it. This is often done specifically by college students because they are trying to save money. Only paying for what they ordered is beneficial to the whole group of students. However, depending on the restaurant or the number of people in the group, a restaurant will decide not to let a group separate a check into individual orders. There are different reasons for this but the main reason is because it can be a lot of work for the busy waiter to manually create separate receipts for everyone. This reason is especially true for large groups of people greater than ten. These large groups are common among college students because college students associate and interact through clubs or other school organizations such as marching band and sports. This situation will force one person to pay the entire bill before they leave the restaurant. Of course everyone will pay back the person for what they ordered, but again this can get complex with a large group of people. To add to the complexity, things get more complex when these large groups eat out often. This is the core issue the app, "I.O.U", intends to solve, tracking. Our app will help large student groups like this track what they owe, allowing the students to seamlessly pay each other back. Here we describe the service, the requirements, the market/demographics, the value, and the competition.

Requirements

The main requirement of this app is that it will require an Android device since it was developed in Android Studio. There currently is no iOS version of the app. There is no specific hardware requirement as long as it is on the Android operating system. The primary feature of IOU is the ability for users to login and report the amount of money they owe someone. That being said this app will not require intense processing speed or memory as calculation is not the main focus. However it does require a network connection because users will sign in to an account in order to report what they owe to other users.

Market/Demographics

As mentioned earlier, college students typically do not have a large amount of disposable income so they are more likely to end up splitting a bill rather than one person just paying for it. So generally the target demographic is college students. But the more specific demographic is large groups of college students who are either friends or acquaintances, that know each other through school and eat out often. This specific group is exactly who this app caters to. They will use this app because this app simply takes away the complexity when it comes to remembering who owes what. There are also no other apps that have the feature of letting users self report how much money they owe. All bill splitting apps have only one source of input, the single user of the app.

Scenario

It has been established that the point of this app is the case when there can only be one check, because the restaurant doesn’t allow it or the group simply forgot. In either case, there will be a group of at least more than three people that want to split a check to pay for only what they ordered. This group decides to designate a person to pay the whole bill, and then pay back that person for what they ordered rather than having the waiter create separate receipts. Once the waiter arrives with the check containing the total for the meal, the designated person paying for the meal pulls out the app, and takes a picture of the receipt (figure B). The user of the app will be able to decide how much tip they will add to the amount, and also add the sales tax. Depending on the situation, the user can then decide to create a list in the app now or later, this list is what will allow the other people in the group to self-report how much money they owe the person. The user may choose to create the list later if they are in a rush or if the environment around them is too hectic to do so. Once a list is created by the person that paid the bill, they can "invite" other users to see that list so that they can type in what they owe the person. So in order for a person to report what they owe, they would also have the app on their phone, and sign in to their own account. The picture of the receipt will allow the other people in the group to refer back to what they ordered in case they forgot. Users can create names for lists as well to help them remember when and where they ate out on different occasions. There are separate categories for bills that the user has paid to which their friends owe them money, and the other way around where the user owes a payer money. In essence the payer will use the app to create an initial list to edit, and the payees will use the app to type in what they owe on that initial list created by the payer. Everyone will be able to remember what to pay back what they intended to pay at the restaurant had they gotten separate checks to begin with.

Value

There aren't a lot of obvious ways to make money with our app. Ads are too invasive for the type of app and buying the app seems like too much for the service. A possible way to create revenue is to do a partnership with Venmo, an existing app that is used to pay people electronically. I.O.U is an app that serves the purpose of tracking money and who has to pay each other back. Venmo is the perfect app to do a partnership with because Venmo is already popular among college students who don't usually carry around cash, and prefer digital methods of paying.

Competition

Looking at both iOS and Android, there are a number of bill splitting mobile apps. They all have the features one would expect in a bill splitting app such as calculating tax and tip, being able to assign orders to different names, and showing different prices. However most if not all of them do not have a feature where users can self-report what they owe. They all require the one user of the app to input the name and amounts that each person owes. Most of the apps also do not store the list that was created, rather it is just an on the fly bill calculator. Taking note of this, IOU stands out from the competition because of its feature to store a history of receipts to which users have self-reported their totals. Essentially IOU tackles the bill splitting problem on larger scales where the groups of people are large, and the frequency to which they dine out is also much larger. The average bill splitting app on the market will not be useful in keeping track of the receipts of a regular friend group of ten people that eat out every other weekend after football practice.

Features and Design of I.O.U

When the user opens IOU, they are greeted with a home screen that has two buttons, one with the option to "Store a receipt" which is the option to create the initial list, and the other option is to "view receipts" which is a history of past receipts. The user is also able to login at the top right of the screen. When the user decides to create a new receipt, they will first be prompted to take a picture of the receipt for their reference, or choose a picture from their album that they have already taken. Then ideally, the app is able to extract the text from the receipt to add the items on the list as well as their respective prices, avoiding the need for anyone to manually input the items. The purpose of having a list of items in the app itself is to be able to properly calculate the tax and tip. If the receipt is not readable and for some reason cannot extract the items, then the user would just need to manually input the items and their prices by hand. After inputting items, the user can then write down the names of the people in the group for the purposes of counting how many are in the group as well as just simply being able to recall later who was there. After that the user has the choice of assigning items themselves if they feel like doing so, or they can just store the receipt for the others to self-report what they owe later. In the final results an evenly distributed tax and tip amount will be initially assigned to each person in the group, the user can then store that receipt in history, and invite other users to edit that receipt. In the "view receipts" activity, there will be lists of past receipts that the user created, as well as lists of past receipts to which the user still owes money to the payer of that receipt. The user will be able to "invite" other users to edit and self-report on a created receipt by simply pressing a button and adding a user. The interface to which users will self report their totals will be similar to what the user saw when they initially created the receipt. There will also be timestamps to see when the receipt was last edited.

Implementation

Design decisions

Most of the design decisions of IOU were made with a large group in mind. Since there will be a lot of people in a group, naturally there will be a lot of items on the receipt as well. It is cumbersome to have to manually type in all of the items on the receipt, which is the reason for the camera scanner. Of course it won't be perfect so there will need to be some way to manually input the items if need be. This is generally the case for most features of the app. There is some manual alternative in the case that the main feature either does not work or does not apply to a certain situation. It's also worth noting again that there are two types of users for this app. The first user is the one that pays the entire bill and creates the initial receipt, and the second user is the one that owes that person money so they use the app to self-report their totals. When signing in to the app there is no actual distinction between these two roles because a user can be either of the two. A user can use the app to create a list, and to also self-report amounts they owe. This separation is however seen in the "view history" activity (figure G) where there are separate receipt categories for receipts that the user has created or has been invited to.

Implementation vs. Design

The current implementation has all the main features of our design, except the actual sign in feature does not work. All activities are set up in such a way that a user is able to go through the main flow of the app where they can create the initial list (see all images below), but there is currently no real way to create an account and invite other users to view the receipt. On the home page there are the two buttons as stated in the design, and there is a button in the upper right corner that would be used to sign in to an account. Right now it is a placeholder button with a welcome user message since the sign in feature is not implemented yet. This is where the user would be able to create an account. The next activity is the camera activity which works exactly as stated in the design, the only difference is that currently the picture is not stored anywhere in the app yet. In the actual app the picture would be saved somewhere to have the user be able to refer to. The feature to extract text information also is not implemented yet, currently the only way to input items is by manual input, which is where the next screen will lead to. This screen is where the user will be able to add an item and its price, as well as the tax and tip. Adding names works just as intended where a list is created of names, and then the next activity is used to assign initial amounts if the user chooses to. In the "view receipts" activity, in place of an intended scrollable list is a group of icons of receipts to which the user can click on to bring up the respective receipts (figure G). Users can see the receipts to which their friends owe them money, or the receipts to which they owe their friends money. These are the intended activities in which users will be able to self-report what they owe. A user of the current implementation can assign amounts but it will currently not go anywhere.

Intended Features

There were some features that were intended to be implemented but did not make it into the final implementation such as features that we believed would have been better design wise, as well as features that were suggested to us through evaluations. One of those features was the ability to assign prices by being able to simply drag and drop prices to names. This was an initial design decision because it is the more intuitive way to assign prices to people, and it was also mentioned in our evaluation. However the implementation itself was a little complex and could not be finished within the timeframe we had, so we chose to use the type in method instead where the user types in a name and a price to assign to that name. This is arguably not the best way to assign prices however a possible benefit of this feature could be seen when the list of people is very large, so it would be difficult to scroll through the list, hence the ability to simply type in the name. Ideally in a more finished version of the app, a user would be able to drag and drop prices to their name as that provides a better user experience.

In our feedback it also was mentioned that at some steps it might be easier to see at the moment when a price is assigned to a person how much they got assigned, namely the PeopleandItems page (figure E). This was a consideration in the design, however it was decided that it might be confusing to the user because an evenly distributed tax and tip amount will be added to the end. Because of this the feature was implemented in such a way that the user sees their assignment in the final results page, to which they can then edit if they want to. And of course the main intended feature that wasn’t added is the sign in feature, this wasn’t because of a decision to, but rather because of the timeframe.

Since the implementation of the camera feature currently doesn’t store the image anywhere, we didn’t include the storage of the picture on the edit receipt activity where users would self-report their own totals (figure H). The picture of the receipt is originally intended for the use of the person that is inputting the prices and items in the app, as with the current implementation, the person creating the list is the one that is inputting everything since there isn’t a way to scan the receipt yet. Ideally an alternative would be to have others input the prices and items themselves, so that’s even less typing for the user that initially creates the list.

In the current implementation of the “view receipts” activity (figure G), icons are displayed to indicate the receipts that the user has created and also been invited to edit. For the current implementation, icons seemed appropriate because there aren’t a lot of receipts created. However in a different implementation, lists might be more appropriate to anticipate the creation of a long list of receipts. Ideally there also would have been some other distinction between the two different categories of receipts, because the two icons are the same receipt logo pictures.

Right now the app is designed such that anyone can edit anyone else’s price, so some kind of feature needs to be implemented to solve that. It is assumed that when the sign in feature is integrated into the app, a simple solution is to be able to only add prices to names that match your own username, that way a user can’t edit anyone but themselves. Currently a feature is implemented to show the last time a receipt is edited, where it shows the date and time similar to how Google Docs does(figure I). If the feature of only being able to edit your name somehow doesn’t work out, another possible solution would be to keep a history log of changes instead of only including a timestamp of when the receipt was last edited. That way everyone is able to see who assigned prices to what names. This feature could be useful in case someone wants someone else to assign their price for them. This could happen because this app is only on Android, it’s very likely that not everyone in the group will have an Android device so they would want someone else to add their price for them.

Evaluation

Evaluation Design

In the actual evaluation that was carried out, our repo was shared on laulima along with a short questionnaire. Users went through the app by cloning the repo and running through a few different scenarios. After they ran through the scenarios there was a short questionnaire for them to answer. This was the best approach for the evaluation because it is the easiest way to reach people to receive feedback. It is also easier to parse the feedback if the evaluators are answering the same questions. The questions are varied and crafted such that they address some of the problems that we already see with the app.

Goals and Questions

The goal of the evaluation is to get the information we need to improve the design. The original app’s purpose was to easily split bills when with a large group of people. The new purpose of the app is to keep a record of bills from when one goes out to eat with a group of people. We changed the purpose of the app based on the feedback we received from our evaluations. To match the new purpose we redesigned certain parts of the app and changed the name to I.O.U.. The goal of the app is to be easy to use and worth the trouble of using. To achieve this we are making the app self-explanatory at each step and colors are being chosen so that it will not distract from the experience. Our research questions are: How often do people use similar apps? Would people use our app? Are the steps easy to follow? and What is a good UI Design? These are questions regarding what can be done to improve overall design. We also are making sure our app is nice to look at and simple to use. The last two questions address this problem.

Methods

The main method we used to collect feedback is to share the repo on laulima along with a few scenarios to run through. After completing the scenarios the user then answers a few questions about their experience and they give whatever feedback that they have. Given the current situation that is all we are able to do but we did have a few alternative ideas as well. The ideal scenario would have been to try the app out while actually at a restaurant with a friend group of ten or more people because this is exactly the situation that calls for this app. The feedback would be more relevant since it would be the actual scenario that the app was made for. In addition, we also could have given the same questionnaire in person to some people on campus to get more diverse results from a random selection of students. The scenario would have been to let them use the app on our own Android device, and then have them answer some questions about their experience. These two methods allow for specific and also diverse results.

Analysis

For the intended scenario where a group of ten or more of our friends split a bill at a restaurant, the most straightforward way to analyze results would be to compile together common trends from a few questions that we ask them. For this scenario, we also need to consider the fact that these are people that we know, and that may affect the evaluation as in feedback may not be very substantial. If that's the case, a general feeling of whether or not it made the situation of splitting the bill easier would need to be assessed. Either way this scenario is where we look for answers to our main objective question of finding if it will make restaurant bill splitting easier. The other method of doing a talk aloud procedure with students on campus and having them fill out a questionnaire would be analyzed from a UI standpoint because that is most likely what a random sample of students will talk about especially if they are not in a mobile design class and don't have comprehensive feedback to offer. For the evaluation that is actually carried out on laulima, since participants are fellow mobile design classmates, we need to consider that their feedback might be more technical and code oriented than our previous two methods.

Participants

Since it may be difficult to get ten random student participants whom we aren't familiar with to eat together at a restaurant, ten of our friends and acquaintances may be the best option for this tailored scenario. The alternative method of simply having students on campus try the app through a talk aloud procedure, and answering a questionnaire would be how we get random student participants. The demographic of this random pool of students would naturally be between the ages of 18-22. This still would have been our target demographic because they are of course college aged students. The random evaluation should take no more than five minutes of a student's time using our Android device, as students will be just passing by to their classes. If we want to get as many participants as possible with this method, compensating in some way, not necessarily monetarily, would be a good idea as it would draw more attention and it'd make the experience fun for everyone. Something as simple as maybe giving out free pens would be one option, anything would do. That being said though, the most accurate evaluation would be one where we have at least ten people in a group setting who are all at least acquaintances. In the actual evaluation participants will include just our classmates, which is more or less the target group considering the fact that most of the people in our class are between the ages of 18-25.

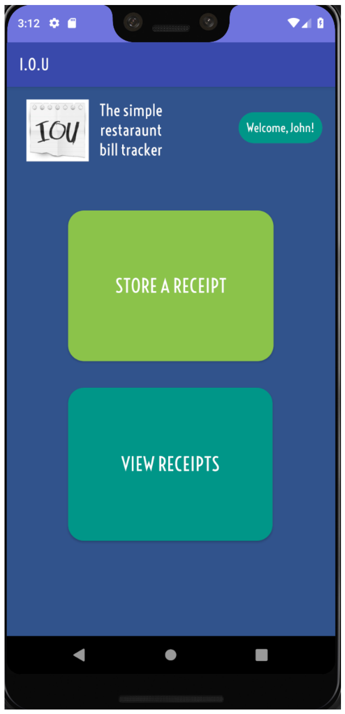
Results

Our goal was to get the information we need to improve our design, and our main research questions were answered from our classmates' feedback. The main research question to be answered was if the steps were easy to follow, and the general consensus was that it was hard to keep track of steps and sometimes there weren't enough instructions given to do what was required. We didn't realize this was an issue and as a result we added more small hints and instructions, and designed our UI in a more self-explanatory way. There were also a lot of comments about small UI fixes that could be made such as the alignment of certain textviews, clearing text inputs when clicked on, having too many buttons, and being able to keep track of changes visually. UI Design was also one of our research questions and as a result we took note of those changes, as well as took note of positive comments such as comments about the theme of our app. Regarding the research question of whether or not people will use our app or if they use other apps as well, we did not get the answer to this question, namely because it was not clear to our classmates that the final implementation and goal of this app would be to allow other users to self-report what they owe through the sign in feature. However they did say that they would not use this app if it was just a simple calculator as it currently is now, so it's worth noting that our sign in feature would be what sets it apart from similar apps. In order to make the goal of the app more clear we renamed the app to I.O.U.. This name better represents that our app is for bill tracking and not just a simple calculator.

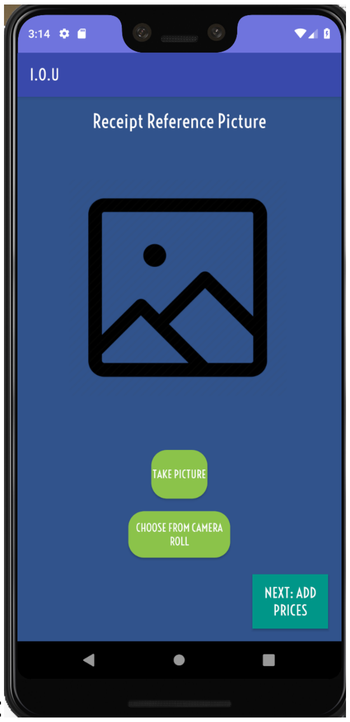
Most of our feedback given was about UI fixes and flow fixes, which are all things we improved on since the evaluation. We aren’t currently able to obtain real evaluation results that closely resemble the ideal use case scenario where we have ten students eating at a restaurant and using this sign in feature but as pointed out before, there currently is no other app on the market that does this. That simple fact alone means that this app will do well with its targeted market. Not to mention the fact that most college students have a smart phone on them, and use them daily. Noting the uniqueness of this app concept, the main detail that will determine its success is the natural flow of the app and ease of use. Since we obtained feedback from our evaluations on this, we have the information we need to finish a well-designed app flow. We ourselves are also college aged students that often eat out at restaurants with groups of friends, so we particularly understand the needs and habits of the target demographic. In short, IOU has a unique concept and it is designed for college students by college students. It targets a very specific demographic with a specific problem, and will grow as more users download the app and create accounts.

Image Reference:

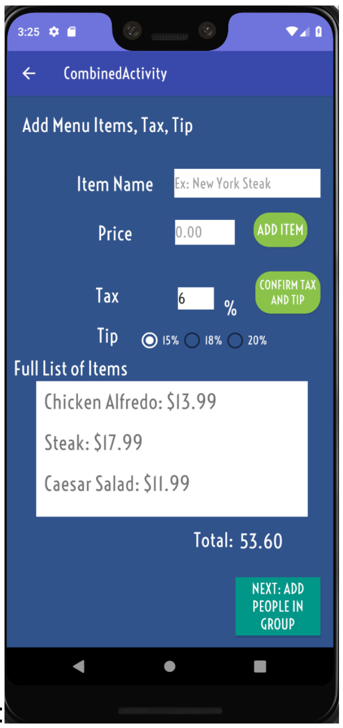
A:



B:



C:



Tax: 6 %

Tip: [✓ 15%] 18% 20%

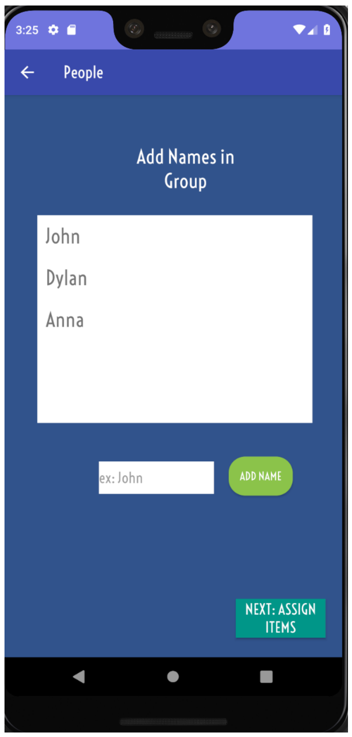
Chicken Alfredo: $13.99

Steak: $17.99

Caesar Salad: $11.99

Total: 53.60

D:

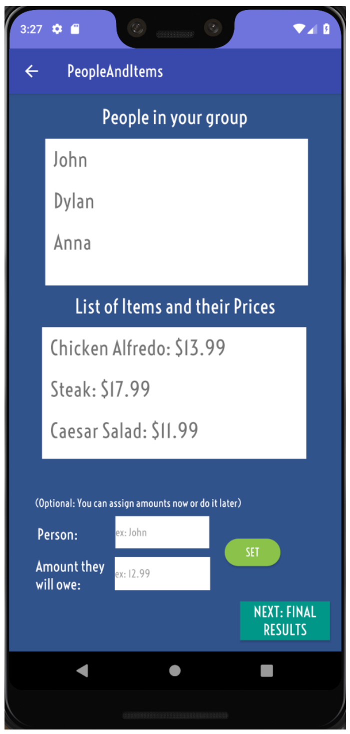


John

Dylan

Anna

E:



John

Dylan

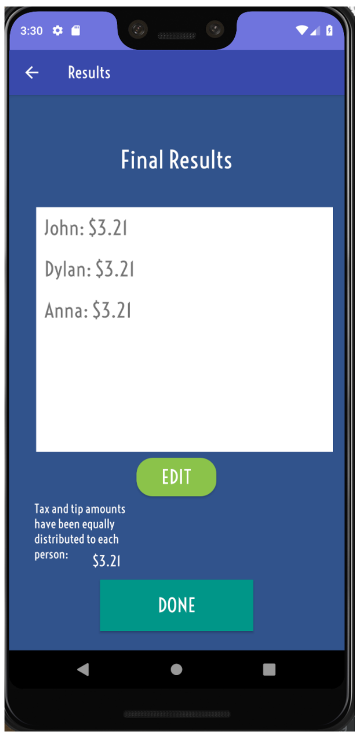
Anna

Chicken Alfredo: $13.99

Steak: $17.99

Caesar Salad: $11.99

F:

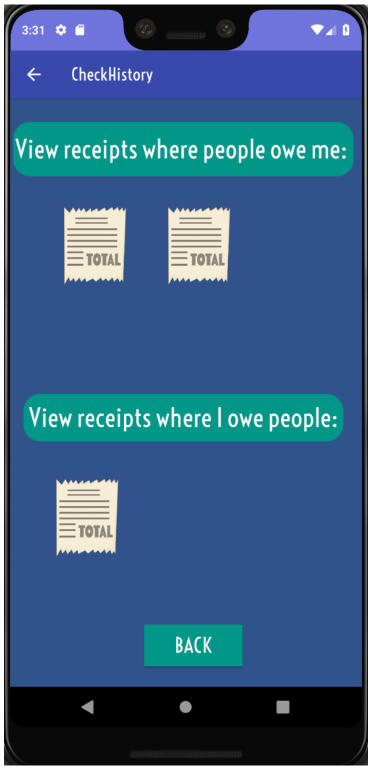


John: $3.21

Dylan: $3.21

Anna: $3.21

Tax and tip amounts have been equally distributed to each person: $3.21

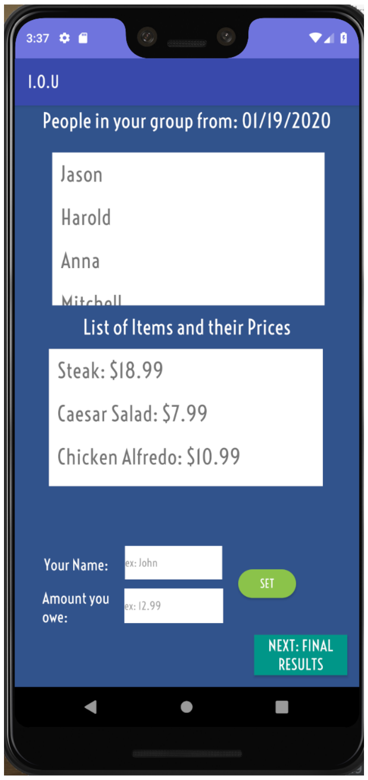
G: 
3:31

CheckHistory

View receipts where people owe me:

View receipts where I owe people:

BACK

H: 
3:37

I.O.U

People in your group from: 01/19/2020

Jason

Harold

Anna

Mitchell

List of Items and their Prices

Steak: $18.99

Caesar Salad: $7.99

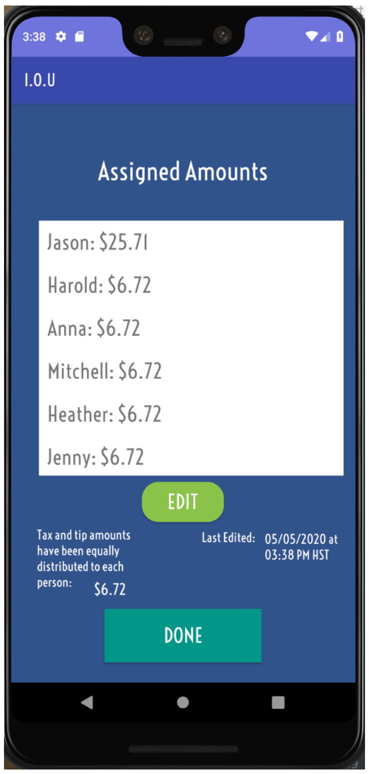
Chicken Alfredo: $10.99

Your Name: ex: John

Amount you owe: ex: 12.99

SET

NEXT: FINAL RESULTS

I: 
3:38

I.O.U

Assigned Amounts

Jason: $25.71

Harold: $6.72

Anna: $6.72

Mitchell: $6.72

Heather: $6.72

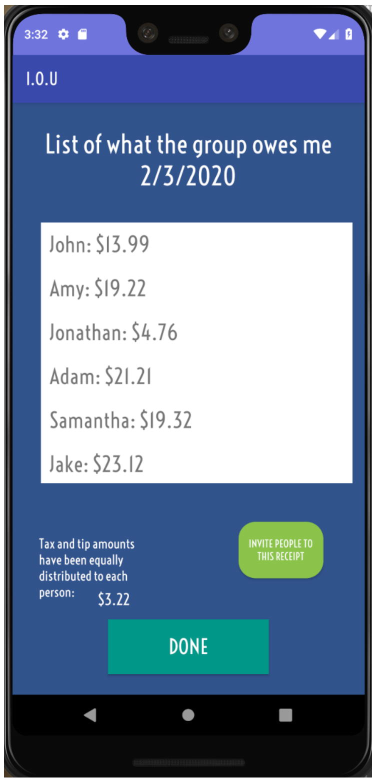
Jenny: $6.72

EDIT

Tax and tip amounts have been equally distributed to each person: $6.72

Last Edited: 05/05/2020 at 03:38 PM HST

DONE

J: 
3:32

I.O.U

List of what the group owes me 2/3/2020

John: $13.99

Amy: $19.22

Jonathan: $4.76

Adam: $21.21

Samantha: $19.32

Jake: $23.12

Tax and tip amounts have been equally distributed to each person: $3.22

INVITE PEOPLE TO THIS RECEIPT

DONE