Uomi: A Group Transaction Management System

Eric Gonzalez
eric.gonzalez@tamu.edu

Yerania Hernandez
hernandez.yerania@tamu.edu

Kevin Nguyen
kevinjnguyen@tamu.edu

ABSTRACT

UOMi is a mobile application that focuses on providing a transaction management system in relation to a specific event instead of a group of people or a single transaction. The purpose of this prototype is to provide insight on social and cognitive issues that arise when people are in debt. The application has various features including two different split mode systems for users and factors in transparency by showing all transactions within the event to all contributors along with attaching a reciept. We provide possible evaluation methods that could be used in order to analyze the usability, efficiency, and effects this application could have on users. In addition, we discuss the potential implications of this work and provide recommendations for future work.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interface

Author Keywords

Application software; payment debts; transaction management; human-computer interaction

INTRODUCTION

Consider the following scenario: Adam and a few of his friends decide to go on a hiking trip out of town. Word spreads, and several friends-of-friends decide to join in, bringing their group to near a dozen loose acquaintances. They carpool to minimize travel expenses, and Adam volunteers to drive. The first day of the trip Adam needs to refuel his car, so he stops at a gas station and fills up. Out of fairness his passengers all agree to split the cost of gas, but since it is just the start of the trip and they foresee a several more stops, they'll postpone reimbursement until the end of the trip so they can settle in one payment rather than several. At the end of their first day of travel, they head to a restaurant to share a meal together. After the meal, since Adam already paid for gas, one of his passengers volunteers to pick up his tab out of friendship (and because it makes up for his portion of the gas bill). He also picks up a couple other friends, and suggests that they can pay him back later. So begins the first of several similar days.

ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

Returning from their trip, Adam and his companions all get together to settle up. Adam remembers his gas bills and calculates the amount owed by each passenger. He arrives at a happy balance but then realizes that half the people in his car are different than the people he initially drove out with. Additionally, there are some people who covered him during several outings scattered about the group, to whom he owes more than they owe him. Adam wonders to himself who owes whom, and how much. At least he remembers that he owes Suzan twenty dollars, but wait, he only has fifteen on him. Not being the only one confused about how to proceed, the group devolves into confused chaos, struggling to determine how to recover (sometimes significant) expenses–especially from people they hardly expect to see again!

Sharing one-time expenses with someone seems a simple task considering there are myriad applications that help transfer money via digital wallets. For longer-running occasions such as trips with multiple parties, however, the task becomes considerably more complex. Sharing expenses between multiple people over multiple transactions is difficult to track mentally, can cause social awkwardness and tension.

Social pressures can negatively impact a generous participant who feels uneasy asking for payment back and can degenerate into conflict when the parties involved disagree on the transactions. Individuals could end up losing a great deal of money or jeopardize their friendships over disagreements. In fact, a study found that roughly 60% of Americans who loan out money do not get paid back [7]. Three of the most common reasons found for not returning money are that the loaner simply refuses to pay back; the parties involved disagree that there is a loan in the first place [7]; or there is a falling out in the relationship.

During long-running events such as the example above, it is not uncommon for people to keep running tabs of who owes whom how much with the expectation that in the end all parties will seek to fairly settle shared expenses. In the end, they work out any discrepancies in the shared burden and settle imbalances to the best of their ability. Often the participants will be lacking in physical currency or exact change to settle their debts, though, and must either negotiate complicated currency exchanges; opt for electronic payments; pay at a later time; or request/grant forgiveness for the outstanding debts. We present an approach to address these issues through a mobile application that mediates the many facets of transaction management such as tracking, splitting, negotiating and settling expenses, while considering social and cognitive issues that arise from lending and owing money.

RELATED WORKS

When it comes to debt, being confronted about the situation instead of being reminded by an application could have different effects. However, Dana et al. [5] investigated the social question about if generosity indeed exists without selfish motives and is able to demonstrate through a dictator game that the participant will not want to appear selfish in front of the receiver. As a result, the participant is more concerned about the perception of the receiver than their welfare and are more focused on trying to give up what they are expected to give. This indicates how the application we are developing is meant to resolve this issue of selfishness from the participant and tries to take advantage of this giving expectation in order to make sure the participant pays back their debt. This social issue of selfishness is even further investigated in Loewenstein's et al. [9] study about people's preferences in decision making when considering oneself and when comparing oneself with another person based on the nature of the situation. As a result, the authors examined an individual's behavior when making decisions that had consequences not only for them but for another person demonstrating that indeed people tend to maximize their advantages but also will try to maximize the other person's displeasure. Such an effect provides insight on debt issues and how people react to each other's claims about who owes what, the main issue that brings us to create a transaction management application in order to tackle this desire of displeasing another party.

Striving for and expecting a semblance of fairness in payments may seem unrealistic at face value, given that users would only need to convince others to cover their expenses and minimize transactions until an event concludes and then simply neglect payback. According to research on trust and reciprocity, however, we have reason to expect this not to be the typical case. Berg et al.'s findings [3] show that when participants with social history extend investment payments toward their fellows, it's received as an indication of trust. That trust drives the recipient to reciprocate and provide returns when possible. This encourages us to expect that participants will seek fairness and reciprocity in sharing expenses rather than gaming the system.

When further analyzing the different aspects of the system we are implementing, we also need to consider Wu et al.'s [12] work, which provided a study utilizing a trusted third party software agent that helps broker a deal in a non-cooperative, one-time transaction by holding guarantee payments. While the dynamic described does not be strictly apply to our system, we can take a longer view of individual transactions as components of a long-running tab, to be settled in later summary transactions. Instead, we can introduce mediation to our system to help determine who should pay the cost of a new transaction. This is key because the question of who pays is a non-cooperative choice in which one party will incur an upfront cost in hopes of future compensation.

A key component of the software agent in Wu's work is that it establishes trust, without which parties would be reluctant to exchange money. This is reflected also in Ferreira et al.'s work [6] where a text-to-pay system was studied which transferred digital currency between two parties. In this work the inherent limitations of the technology resulted in delayed, asynchronous confirmation for the two parties, and in the meantime was found to be an occasion for social interactions. This fostered social ties and established trust, which in turn eased apprehensions about using the system or being cheated by users frequent acquaintances. In this case, too, we see a digital mediator between a transaction, strengthening our intuition that an application that facilitates transaction management will receive and foster trust between users, especially over time.

Understanding debt from a psychological pointview is also essential, especially when considering behaviors and habits of people. Lea et al. [8] created a study to analyze if debt was indeed independent from different attitude and economic variables. Through the methods implemented it became evident that economic, social, and psychological factors all play significant roles in relation to debt, including the consequence of serious debtors being part of a low social class, the ideology of serious debtors that their friends would not disapprove of their large amounts of debt, and social impact of having more children, being single parents, and being younger were main characteristics of serious debtors as well. Being able to analyze these findings will allow us to take advantage of these type of dependencies in order to provide a marketable product for consumers that have debt issues among their social groups. Another psychological perspective that needs to be considered is people's habits and being able to recognize their tendencies. In the text-to-pay system [6] mentioned above, users had to text payment information to the system, including the dollar amount and the recipient's information. This had the effect of forcing users to actively acknowledge how much money they are spending rather than passively accepting an amount from a payment terminal, prompting some to reflect on their habits and more consciously consider purchases. We hope to capture this effect by requiring users to manually enter dollar amounts to send to other parties when settling up in our system.

Since our target system is inherently social, we must consider how to encourage adoption by not just individuals, but their social circles as well. Tuk [11] focuses on understanding the power of word of mouth as a source for consumer influence. The main focus of this study was to gain insight on differences between recommendations that were rewarded or not and investigating how people react to these differences when evaluating the different recommendations they receive. Results demonstrate that interactions even with subtle cues have a long-lasting impact on the perception from the evaluator, including facial cues, relationship norms, and in-group persuasion. Recommendations that were rewarded demonstrated both a positive and negative attitude depending on the amount of transparency the recommender had with the motivation for this reward and the amount of trust the evaluator perceived when interacting with the recommender. With this type of effect, we need to design and promote our application in a manner that appears trustworthy to the user along with provides transparency of any transactions made.

In addition, when designing the interface for this system we can consider the experiences of novice users of mobile ap-

plications by looking to developing countries where mobile phones are increasingly available, but familiarity with their applications and literacy is low. Medhi et al. [10] performed ethnographies in several developing countries that had mobile phone-based banking systems. They found that novice users had little familiarity with hierarchical navigations, and that task completion rates were (predictably) higher when a rich media UI was used in place of a text-based UI. Supporting non-literate users is not a focus of our system, but these findings are in line with common interface guidelines that recommend employing rich icons over textual content when possible, and adopting this approach should result in easier comprehension and greater usability of the system.

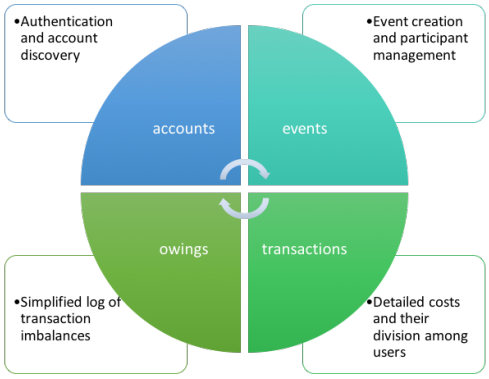


Figure 1: An overview of the Firebase database data structure

SIMILAR METHODOLOGIES

Besides social and cognitive issues that arise based on debt, other approaches have been taken in order to investigate possible ways in order to deal with transactions. Pay with a (group) selfie (PGS) [4] is a project utilizing personal pictures or "selfies" as a form of transactional proof toward safe exchanges of money. The PGS implementation of a decoupled payment system from a transaction system allows multiple transactions between the same parties to share the cost of an individual money transfer while utilizing visual cryptography to secure the transactions themselves from tampering. This allowed PGS to successfully manage and secure different transactions in locations with limited connectivity and we can consider a similar implementation of transaction tracking in our mobile application.

More closely related to our system, various existing commercial applications enable payments between friends and family, such as Venmo, Paypal, Apple Pay and Facebook Messenger. These applications require you to specify the amount you owe (or are owed), are only a one-time transaction, and seldom support payment splitting as a built-in function without having to initiate separate requests. Other applications facilitate splitting transactions among multiple parties and dividing expenses. Some, like Plates, Divvy and Billr take a specialized approach toward a specific kind of one-time purchase, namely restaurant costs. Users add restaurant bills manually or through capturing a picture of the receipt, and the app assists with splitting the bill and including tips. A more general version, Splitwise[2], is more akin to the system that we envision as it tracks and splits transactions over time between multiple parties, but it has a couple of shortcomings.

Mainly, data input is currently manually entered when generating bill amounts and determining division of expenses. In this regard we would prefer to utilize a digital assistant to help accession transaction receipts, such as using OCR. Another benefit of keeping the receipt as a digital record is that it provides the additional benefit of providing a verifiable evidence of the transaction should it be disputed by parties. Whereas in SplitWise the amounts are generated by hand, it does not record location, date or time. We hypothesize that these components are important to establishing trust, especially for recent associates.

DESIGN

Our implementation revolved around a mobile phone application that provides a transaction management system to alleviate the issues of recording and splitting expenses throughout the duration of an event. The mobile application was prototyped on iOS 11.0.3 and targeted iPhone 6S/7/8. The following sections describe our approach to designing the system in order to support peer interactions in a distributed environment. We discuss first the structural design of our data, followed by a description of our prototype application interface.

System Design

Our system uses a central server to store and manage user accounts and event information. To simplify our setup, we opted to use the Firebase service [1] to leverage its Backend-as-a-Service features for authentication and data storage. Firebase provides a highly reliable and accessible platform, as well as a responsive, real-time connection to data for live updates to connected clients. Using Firebase encourages adopting a flat hierarchy of stored data, since all child elements of a particular *site* (e.g., a resource's relative path in the database) are downloaded with each site node. Since Firebase is also a non-relational database, our setup required some data duplication not only to reduce network traffic for performance and security reasons, but to link data elements together.

Our system is fragmented into several key components: an *Account Manager*, an *Event Manager*, a *Transaction Manager*, and a *Balance Manager*. Each manager provides facilities for performing the various operations necessary for the system to operate. The *Account Manager* handles account creation, authentication, and enables user discovery. The *Event Manager* provides capabilities for managing event configurations. The *Transaction Manager* primarily logs transactions recorded by the users, and the *Balance Manager* identifies imbalances in cost splitting between event-goers. Each manager corresponds with a particular site of the Firebase database: *accounts*; *events*; *transactions*; and *owings*, respectively.

Accounts Management

As shown in Figure 2, the Firebase database consists of an *accounts* site that contains the basic user information such as name, email, and a list of events that the specific user is a part of. However, the events that are listed are only a hash

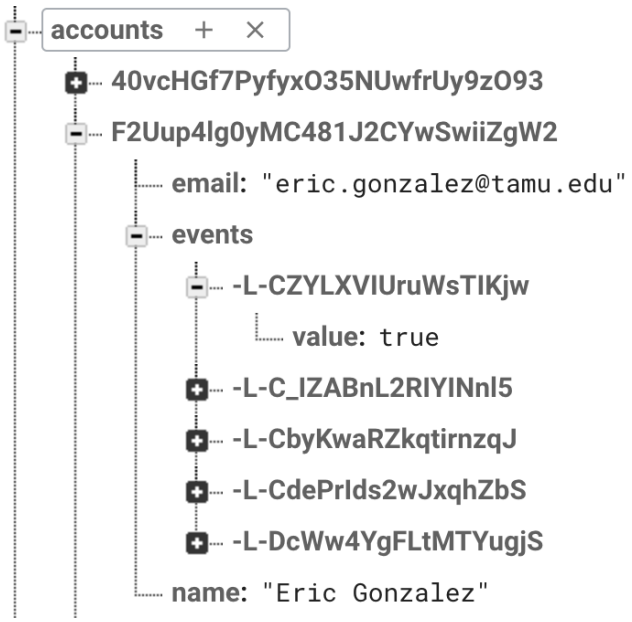


Figure 2: Demonstrates the Firebase organization of the Accounts branch.

Figure 3: Demonstrates the Firebase organization of the Events branch.

of keys with no additional information on the event itself or the transactions involved. This type of information will be found in the other sites within the database and is purposely organized in this manner in order to quickly create and retrieve the minimum relevant information required for each task, in accord with Firebase's recommended structure. This specific site is used primarily during user registration and authentication in the mobile application, described further in the next section.

Events Management

Using any of the *events* keys from the *accounts* site's users provides access to the corresponding event's metadata. This includes the name of the event; its description or purpose; the creator of the event; and keys that reference all the participating users, such as shown in Figure 3. In addition to the information stored in the database, the *Event Manager* provides the functionality for creating and saving an event, as well as loading multiple events or a specific event.

Transactions Management

The *transactions* site in the database houses one of the primary features of our entire application: the recorded purchase events that occurred during events, from the total to the cost sharing distribution details. Transactions are stored under the key for their respective event for context and to assist in client-side access. There are two kinds of transactions as far as our system is concerned: expense transactions, and repayments. The former pertains to any purchases made by one of the participants for a shared good or service, while the latter documents a settlement between two participants.

Expense transaction data includes a field to record either a name or description identifying the transaction; the date the transaction occurred; the identifier of the user who paid the cost of the transaction; the total paid; the method for splitting the transaction costs; and a breakdown of the cost division per contributor.

Within the list of contributions, there are two formats for the metadata based on the split mode. As shown in Figure 4a, if the split mode is set to itemized, the contributions consist of two main elements: a contributor identifier and a textual description. The identifier corresponds to one of the participants in the event, whereas the description is a textual representation of line-item as (optionally) the number of items, a human-readable description of the item, and the unit price the item delimited with the special '@' character. If, on the other hand, the split mode is entered as percentage-based, as shown on Figure 4b, the contribution consists of the contributor identifier and a numeric percentage amount relative to the total that the contributor will cover for to this transaction.

The *Transaction Manager* brokers access to creating, loading, and saving transactions. When the *Transaction Manager* saves a transaction, it updates three sites in the database: the *events* site to ensure that the transaction is associated with the event; the *transactions* site to record the body of the transaction data; and the *owings* site, in which it stores a much more basic representation of the transaction as a relationship between purchaser, owers, and their respective debts for that transaction.

The other kind of transaction, repayments, is a much-simpler version; it only comprises the payer, date, total, and a recipient identifier. Upon saving, the *Transaction Manager* updates the *owings* as described above, which helps to even out imbalances.

(a) An itemized transaction. (b) A percentage-split transaction. (c) A repayment transaction.

Figure 4: Demonstrates the Firebase organization of the Transactions branch.

Figure 5: Demonstrates the Firebase organization of the Owings branch.

In saving updating the *owings*, the transaction manager computes the amount owed for the transaction by each party, based on the kind of transaction and split mode. For percentage-based transactions it's a simple percentage computation of the total according to the contribution amount. For itemized transactions, it's computed from parsing the textual representation for the number of units times the cost per unit. Lastly, and most simply, for the repayment transaction the transaction manager simply uses the total value.

Balance Management

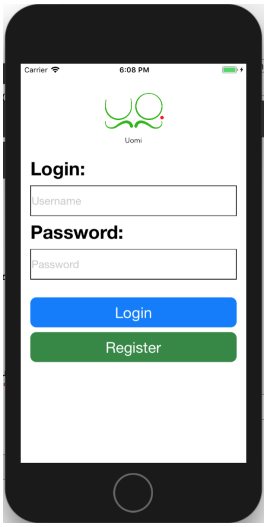
The *owings* site (Figure 5) in the database is interrelated with the *transactions* site as described above, written to on each transaction save. In fact, the owings mimic the hierarchy of the *transactions* site as elements under an event identifier. The *Balance Manager* reads the recorded amounts of this site to compute imbalances between participants of the same event. Each identifier in the site matches the identifier of the transaction from which it is derived, and contains the key of the payer along with a hash of owers and the exact amount owed by each (computed as described above).

Storing these values separate from the main transaction body helps in two ways. First, it separates the concerns of the application, where one site is dedicated to recording transaction details, and the other for holding related, calculated values. The other benefit comes from the flat hierarchy of Firebase, which reduces the amount of data required to perform balance-related computation operations and simplifies the logic needed by the *Balance Manager*.

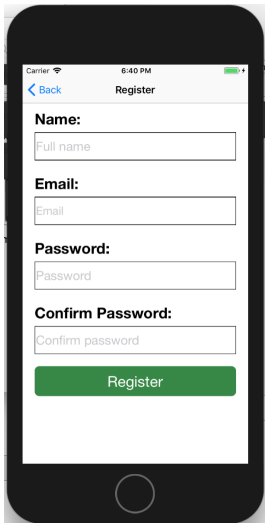
The *Balance Manager* is an essential component of the system, which keeps track of imbalances between the current user and the other participants of the respective events. The main function of this manager is exclusively in computing balances, both for monitoring by the active user and in assisting with settlements. When two different users have covered for each other over various transactions, the *Balance Manager* will handle calculating the summed difference in order to determine the real balance of who owes whom within the specific event. Since this behavior is strictly passive, the *Balance Manager* does not write to the database, leaving that responsibility to the *Transaction Manager* upon its write operation.

Prototype

After various considerations in design aspects, such as key user interactions and required features; visual elements necessary to assist each feature; transaction logging mechanics; color implications with respect to imbalances between users; etc., we developed a functional prototype. Playing on the acronym IOU, our system was dubbed UOMi.

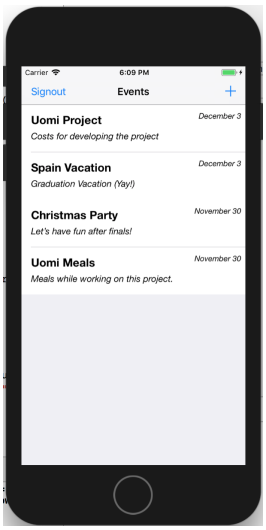


(a) Demonstrates the log in screen with an email and password already populated.

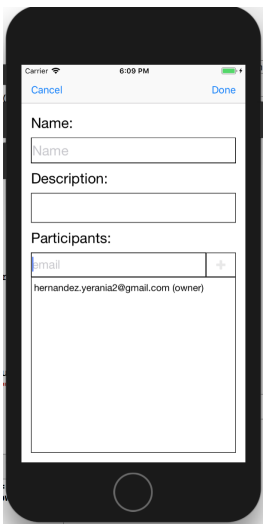


(b) Demonstrates the register screen with a name, email,password,and password confirmation

Figure 6: The Account manager maintains the login screen (6a) and the register screen (6b).



(a) Demonstrates the events screen where users see all the events they are contributing to and the last active date.



(b) Demonstrates creating a new event based on the name, description, and a list of participants

Figure 7: The Event manager maintains the event dashboard screen (7a) and creating a new event screen (7b).

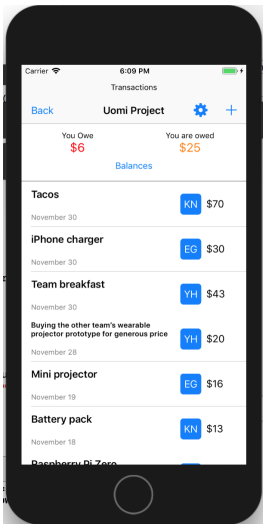
Registration and authentication

Once the application is installed and opened, the first screen the user sees is the login screen as shown in Figure 6a, since all interactions with the system are executed with respect to the current user, and so one must be logged in to accomplish anything meaningful to the user. Users either enter their login credentials–their email and password–or, if they are a new user, they can register for an account. Creating an account leads to the screen shown in Figure 6b. The *Account Manager* uses the provided information to register an account, and upon successful creation logs in to the application.

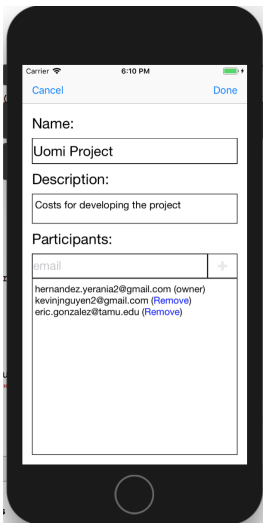
Event Management

After the user has registered or logged into the UOMi application, the user is presented with the Events dashboard, shown in Figure 7a. This screen consists of all the events the current user is a part of and will contain the name of the event, the description of the event, and the last date of activity. The list is sorted reverse chronologically with the understanding that a user's most visited event will be an ongoing one, rather than older events that are likely to be settled.

If the user has not already been added to or registered any events, they will either wait for an invitation to become part of an already existing event, or create an event of their own. The plus sign at the top right of the screen allows the user to create an event, collecting the data displayed in Figure 7b. An important feature in creating an event is the restriction of adding only registered users as participants, which is verified by the *Account Manager* whenever a potential email address is entered. When the email address entered in matches a registered user's email, the neighboring addition button changes to an enabled state indicating that the email corresponds with a valid user. Adding users is immediately efficacious rather



(a) Demonstrates the transactions screen where users see all the transactions of an entire event



(b) Demonstrates creating a new event based on the name, description, and a list of participants

Figure 8: The Transaction manager maintains the transaction dashboard screen (8a) and the event manager maintains the editing event screen(8b).

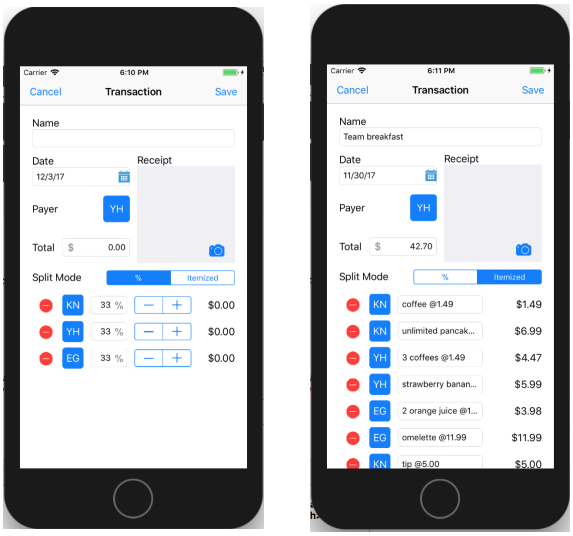
than requiring confirmation, with the expectation that email addresses will likely be manually entered by the users themselves on the device belonging to a current participant in the event.

The events list polls Firebase but does not take advantage of live updates in order to limit network usage on the device. Instead, users can drag downward in a platform-typical gesture indicating the need to refresh. This prompts the app to query the *Event Manager* for updates to the events. The manager pulls in the live state of the events with which the user is associated, including any new events since the last refresh as a result of being included by other participants.

Transaction Management

Selecting an event displays the Transactions dashboard, shown in Figure 8a. This screen leverages mainly the *Transactions Manager* to retrieve all the transactions that have occurred for this specific event. In addition, it displays a quick summary of the total amount of money the current user owes to all other participants in the event and vice versa.

There are three features that can be found in the Transactions dashboard: editing the event's information, editing new and existing transaction, and viewing the event balances with other participants. In order to edit the current event, the settings button can be selected from the top right-hand side, displaying the screen shown in Figure 8b. This provides the opportunity of removing or adding additional participants to this specific event if necessary along with editing the name of the event and the description. Figure 8b continues to use the Events manager to load updates at appropriate moments, such as when event information.



(a) Demonstrates the transactions screen where users see all the transactions of an entire event

(b) Demonstrates creating a new event based on the name, description, and a list of participants

Figure 9: The Transaction manager maintains the transaction dashboard screen (9a) and the event manager maintains the editing event screen(9b).

In order to create a transaction, the addition sign can be selected from the Transactions dashboard. The resulting view, the transaction editor, appears as shown in Figure 9a. The information collected includes the user-designated name of the transaction; the date on which it occurred; the participant that paid the bill; the total cost paid; an image upload of the receipt; and the contribution breakdown.

All details of the transaction are editable by any participant of the event for transparency and to facilitate corrections to honest mistakes in data entry. The inclusion of a picture of the receipt is critical in establishing trust in the accuracy of data entered in the transaction log, and exists as an record of the payment to help avoid disputes that may otherwise arise over discrepancies in amounts due to lapses in memory. The image is attached to the transaction by tapping the camera icon, which triggers the device's camera to record an image.

The two variations of the split interface are demonstrated in both Figure 9a and Figure 9b. If the user selects the percentage mode, the transaction is automatically evenly split between all members of the event. The user will then have the option of removing contributors that are not related to the transaction, which automatically redistributes the remaining percentage among the remaining contributors. Additionally, the user can specify the exact percentage to be covered by each respective party. Specifying a percentage amount locks the percentage owed by that participant in and redistributes the remainder again among other participants whose contributions have not been locked in through manual override. If all all contributors' percentage owings are entered manually and they sum to over 100%, only the last value remains locked, and all other values are reset to an equal share of the remaining percentage.

If the user selects the alternative split method, *Itemized*, then the user will need to enter line-item components additively, compared to the subtractive behavior of the percentage contributions. Upon tapping the add button to generate a new line-item, a participant selector pops up that prompts the user to specify who will cover the line-item's cost.

After selecting a participant, the user annotates in a text field a description of the components according to a structured syntax that specifies a quantity, description, and cost per unit. The format is captured using a regular expression, with the quantity defaulting to 1 in absence of a numeric value. Interpreting the statement using a structured syntax allows the user to enter pertinent data in a manner akin to a verbal description, without leaving a single text field. This ought to assist with rapid data entry.

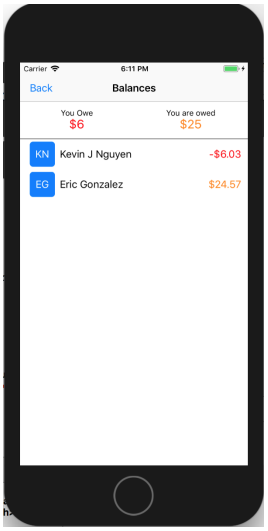
Each line-item includes a readable label that displays the computed contribution inferred from the description. A running subtotal allows the user to track the logged item costs compared with the transaction's specified total to identify and account for discrepancies. This can be used to log a final value to include tip and/or tax to make sure that the total and subtotal equal out, or the user can simply omit the difference in the line-item costs and the contributors will not need to pay for the remaining sum.

Selecting the created transaction from the Transactions dashboard will lead to the screen shown in Figure 9b, which allows the user to edit the information provided for the transaction, such as replacing the picture of the receipt, updating contributors, or even changing split modes.

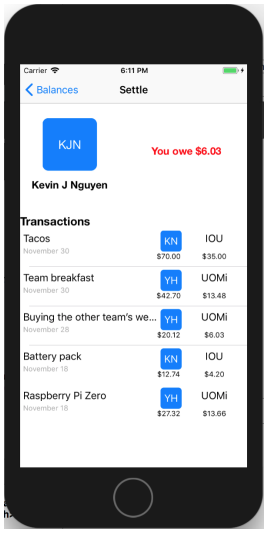
Balance Management

The Balance dashboard, as shown in Figure 10a, provides detailed information about the amount owed between users in relation to the other users within the context of the event. The variation of color identifies the type of imbalance. Red signifies that the current user owes money to another participant, matching visual conventions in the finance world for conveying money owed or paid, and due to its psychological effect in western cultures of imbuing a sense of urgency, to encourage users to settle their debts. Orange identifies that another user owes the current user, and was chosen due to its proximity to red and because it conveys a similar, but less severe, state of imbalance. A final color, green, is used to represent that there is a zero-balance between participants, and that no money need change hands. Typically this would be observed before transactions are logged, and after balances have been settled.

When selecting one of the users, the screen shown in Figure 10b appears, detailing the transactions between the current user and the user selected, which justify the displayed balance information. This transaction list will be based only expenses that have occurred between these two participants, allowing the users to confirm why they see the indicated values. Each transaction includes the use of phrases such as "UOMi" and "IOU" to demonstrate the direction of which the payments were made. An important feature of this page is its ability of logging the payments.

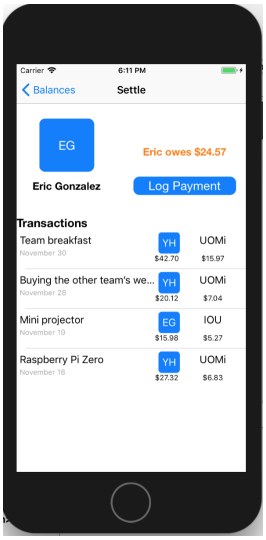


(a) Demonstrates the balance screen where the amount owed to the current user and owing to other users is displayed

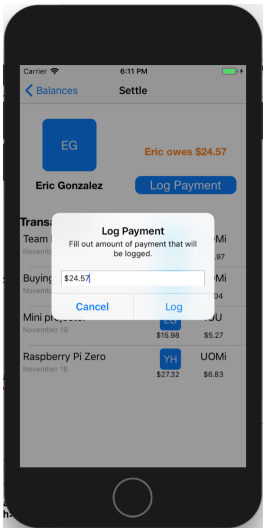


(b) Demonstrates the settlement screen with the transactions between two specific users and the balance left

Figure 10: The Balance manager maintains the balance dashboard screen (10a) and the settlement dashboard screen(10b).



(a) Demonstrates the settlment screen with the log payment available



(b) Demonstrates the settlement alert in order to log the transaction

Figure 11: The Balance manager maintains the settlment screen with the Transaction manager logging the payment.

When the user is owed money, he or she will have the option of logging a repayment transaction once the debtor has made such a payment. This will be shown through the "Log Payment" button as shown in Figure 11a, which is only visible when the current user is owed money. The decision to limit the visibility of payment logs to the lender is to help prevent against gaming of the system by the debtor to trick the system into seeing the participants as settled up when no money has changed hands. In reality, we would prefer to connect this kind of payment logging to an actual transaction system and tie into electronic payment confirmations to update imbalances.

Selecting the *Log Payment* button prompts an alert to appear to collect the payment amount, pre-populated with the current balance owed as shown in Figure 11b. After updating the amount as appropriate, the user submits the transaction and the transactions list will display a new row with the payment details. Indeed, since the repayment was a transaction, all participants in the event will also see the repayment transaction posted in the event's transactions view. Lastly, the imbalance is recalculated and displayed in the balance view's header section.

The balances page then refreshes to recalculate the imbalance between the two participants.Within the Balance manager, the transaction will be added and calculate the imbalances in order to properly update the information in all the tables that display such transactions and calculations, including the Balance dashboard, the Transactions dashboard, and the Events dashboard.

EVALUATION

Our prototype was designed to ameliorate the cognitive and societal issues and implications discussed earlier from the existing methodologies and research work in the related work.

There are various analyses that can be taken into consideration in order to provide a thorough evaluation on the impact of the developed prototype. Our approach in evaluating our system would use a combination of field studies with quantitative analysis on system usage.

Field Studies

Considering that the prototype is a mobile phone application, it would be essential to analyze how it is leveraged in the wild on a user's own terms, and in the situations for which it was envisioned. To that end we would seek to recruit participants who have existing plans to travel as a group and share the costs of their trip. Since our application is currently only available on iOS, we would need to restrict participants to groups where at least one member has an iPhone in order to ensure that there is the possibility for data to be logged throughout the trip's duration. We would not want to provide separate devices in order to minimize behavioral effects due to unfamiliarity with a device, any reservations on the part of participants about carrying multiple phones, or unease in handling another's device.

Ideally we would envision an ethnographic investigation whereupon we would accompany the participants, but if this were not possible then would substitute with requesting the participants to keep a journal describing their experiences to hand in upon their return. Prior to distributing the application to participants, we would add logging to the application to record interactions such as time spent per page, gestures performed, and data entry characteristics such as how long users require to log transaction data. This would be beneficial in assessing the intuitiveness of transaction logging and possible changes that might be needed in order to improve the application for data entry or comprehension.

Observing behavioral changes is a primary question when it comes observing how transactions are logged over time as users become more familiar with the strengths and weaknesses of the application and with using the application on a daily basis. Particularly of interest would be analyzing such behavioral changes when it comes to splitting costs, paying back, or being paid back.

Pre-Test Questionnaires

Prior to departure of participants for their respective events, we would provide a brief pre-test questionnaire to each participant. Recording prior exposure to similar systems and biases are critical to performing a qualitative study in the field, since each participant brings with him or her a set of prejudices, expectations, and habits that will undoubtedly color their interactions with the system. Questions we propose to ask in a pre-test questionnaire include but not limited to:

- What budgeting systems if any have you used?
- What applications have you used to help with cost sharing?
- How would you describe your usual approach to splitting costs in group events?
- How would you describe your experience with sharing costs on multiple-day events?
- How comfortable are you sharing costs with others?
- What is your primary method of settling costs (if applicable) in group events?

Post-Test Questionnaires

Following the field study, when participants return to hand in their journals, we would follow up with a post-test questionnaire to focus on two aspects of their experience: a system usability scale (SUS) and a task completion assessment. The main use of the SUS questionnaire, with response values being an integer between 1 and 10 to indicate strength of opinion, is to analyze if the system is indeed intuitive for even a small sample of subjects. Possible statements we propose for the SUS questionnaire are:

- I found event management straightforward
- I found transaction management to be straightforward
- I was readily able to identify how much money I owed and was owed
- I found the balance screens easy to understand
- I found the application easy to use as a whole
- I needed to learn a lot before I could use the application
- I quickly learned how to use the application
- I was confident in my ability to log new transactions quickly and accurately by the end of the event
- I found the functions in application well integrated

Other questions in the post-test questionnaire that would be useful in this case would relate to a task completion assessment, analyzing how users accomplish a specific task they are assigned within the application. Similar to analyzing their behavior, it will become essential to understand how the user accomplished tasks along with how easy was it for them to even complete a task without further direction in the application or external assistance. Such questions will provide a better understanding on the application's intuitiveness, invalid assumptions, and design flaws. Specific questions that could be used in this assessment are:

- How would you describe your experience settling costs with the other members?
- Did your approaches to logging change as time went on? If so, why?
- Was your willingness to pay for others affected using the app?
- Were there transactions that you did not log? If so, why?
- How well were you able to record your costs?

Quantitative Analysis

After collecting various questionnaires, observational studies, and logs of user interactions, providing a quantitative analysis the interaction logs could reveal patterns in behavior that either affirm or contradict reported sentiments regarding performance and intuitiveness. We would seek to compare at least the following measures in order to analyze task accomplishment speed and perceived intuitiveness:

- Ratio of percentage to line-item transactions
- Cost dynamics of recorded transactions
- Number of transactions per event
- Number of distinct contributors per transaction
- Number of logged payments per event
- Number of interactions per page
- Time spent on each page
- Page visit behavior over time

DISCUSSION

In addition to iterating our design based on feedback from our evaluation, there are various future improvements that can be implemented to improve our prototype. For instance, in the itemized splitting mode a feature could be added to split the cost of tax and/or tips among interested participants as an exception to the single-payer per line-item approach, which is an aspect that is currently not considered in the application.

Another key feature to incorporate would involve using artificial intelligence and OCR to parse the receipt image and automatically extract transaction details for the business name and total, and to infer a starting point for both the percentage and line-item transactions. This would provide more ease to the user by alleviating discrepancies due to typing errors, though the feature would have its own drawbacks and might require user interjection to correct misread values. This aspect was originally part of our ideal design but was slated for future development due to inconsistency in receipt forms and the issues that this could cause for the user.

In addition to improving the transaction logging facilities, further work would be dedicated to integrating with different payment systems to initiate payments directly from within the application. Currently UOMi only considers logging the transaction from the user that is owed and does not allow for electronic submissions of payments. Integrating existing payment systems would provide further security and confidence in the system for the user and could leverage services commonly used for such a task, such as PayPal, Venmo, and so forth. This would also ease imbalance updates if we could use transfer confirmations to log transactions automatically, replacing the reactive Log Payments button with a proactive button to make payments.

Another aspect that would be valuable to consider, particularly for event-goers who are strangers to each other outside of the event and in lieu of the payment integration mentioned, would be messaging capabilities in order to directly communicate between users. This would theoretically assist when issues or unmet expectations arise, could be used to remind others about payment, or could simply be used for planning purposes by event-goers. This might especially be useful when making transactions with unknown users and allows at least one form of communication to later settle the balance.

CONCLUSIONS

As a result of our prototype and our future endeavors for evaluation, our application shows potential to analyze the implications of social and cognitive issues that arise when trying to settle payments in group settings. After meticulous analysis on our design, our prototype demonstrates the main features that can be used in order to create events, log transactions, transparently view balances between each other, and settle payments. Various evaluation methods, including field studies, pre-test questionnaires, post-test questionnaires, and qualitative analysis, could take place in order to analyze the potential uses and areas for improvement for this application along with observing behavioral changes among users over time. In conclusion, UOMi could potentially open a new series of inquiries on the dynamics of group cost sharing behaviors when taken in context of longer-running event transactions instead of one-time payments.

ACKNOWLEDGMENTS

We would like to thank Dr. Furuta for instructing our class and providing insightful discussions that helped inform and improve our project design, and equipped us with various techniques for further evaluating and improving our prototype.

REFERENCES

1. 2017. Firebase. (2017). https://firebase.google.com/
2. 2017. Split expenses with friends. (2017). https://secure.splitwise.com/
3. Joyce Berg, John Dickhaut, and Kevin Mccabe. 1995. Trust, Reciprocity, and Social History. *Games and Economic Behavior* 10, 1 (1995), 122–142. DOI: http://dx.doi.org/10.1006/game.1995.1027
4. Ernesto Damiani, Perpetus Jacques Houngbo, Rasool Asal, Stelvio Cimato, Fulvio Frati, Joel T. Honsou, Dina Shehada, and Chan Yeob Yeun. 2017. Pay-with-a-Selfie, a human-centred digital payment system. (Jun 2017). https://arxiv.org/abs/1706.07187
5. Jason Dana, Daylian Cain, and Robyn Dawes. 2006. What you dont know wont hurt me: Costly (but quiet) exits in dictator games. *PsycEXTRA Dataset* (2006). DOI: http://dx.doi.org/10.1037/e722842011-057
6. Jennifer Ferreira, Mark Perry, and Sriram Subramanian. 2015. Spending Time with Money. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW 15* (2015). DOI:http://dx.doi.org/10.1145/2675133.2675230
7. Gregory Karp. 2015. Two-thirds of Americans lend money that is never repaid, survey finds. (Mar 2015). http://www.chicagotribune.com/business/ct-lending-money-survey-0320-biz-20150319-story.html

8. Stephen E.g. Lea, Paul Webley, and R.mark Levine. 1993. The economic psychology of consumer debt. *Journal of Economic Psychology* 14, 1 (1993), 85–119. DOI: http://dx.doi.org/10.1016/0167-4870(93)90041-i

9. George F. Loewenstein, Leigh Thompson, and Max H. Bazerman. 1989. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social Psychology* 57, 3 (1989), 426–441. DOI: http://dx.doi.org/10.1037//0022-3514.57.3.426

10. Indrani Medhi, S.n. Nagasena Gautama, and Kentaro Toyama. 2009. A comparison of mobile money-transfer UIs for non-literate and semi-literate users. *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09* (2009). DOI: http://dx.doi.org/10.1145/1518701.1518970

11. Mirjam Tuk. 2008. Is Friendship Silent When Money Talks? How People Respond to Word-of-Mouth Marketing. (Jun 2008). https://repub.eur.nl/pub/12702/

12. Shih-Hung Wu and Von-Wun Soo. 2002. Escape from a prisoners dilemma by communication with a trusted third party. *Proceedings Tenth IEEE International Conference on Tools with Artificial Intelligence (Cat. No.98CH36294)* (2002). DOI:http://dx.doi.org/10.1109/tai.1998.744767