W4156 - Preliminary Project Proposal

Project: Go Dutch

Project Team: Dutchers

Qianrui Zhao (qz2338)
Zhihao Li (zl2695)
Yushi Lu (yl3974)
Qian Zheng (qz2348)

Sept 24th 2018

Language: Python (OCR), Java (Android development), MySQL (database)
Platform: Android

Introduction:

We would like to develop an Android mobile application named “Go Dutch”. Unlike many budget planning and expense recording applications on the current market, this application would be more focused on record, manage and maintain expense shared by multiple people, i.e. group bills. This application helps users to record expense by simply scanning receipts of purchase, and users are allowed to specify their own items with customized quantity. Besides, this application can automatically split common expense, including tips and taxes, among all involving consumers. If a shared purchase paid by one or multiple people, the money flow will be clear so that every involving consumer can realize how much they should pay for the payer(s) or received from others. Apart from this, group chat will be supported in this application. Once the sub-bills is worked out, invitations which invite consumers to check out those sub-bills will be automatically sent out to all group members as a gentle reminder to press for payment. We believe by using this application, users can be free from tedious and error-prone calculation process. As a result, making budget plans and maintaining expense-logging books will no longer be tedious and inefficient. Last but not least, some additional social functionalities, as listed below, would also be implemented to enhance the communication between users and the user experience as well.

The development process of this application involves two main parts: mobile device (Android) programming (Java) and Receipt Recognition which is based on OCR technology (Python). Due to the time limit, we may not be going to build these two parts from scratch, which means we may take advantage of some available open-source resources on Github or other third party platforms. Even so, we will try our best to keep the originality of this application. Besides, Java will also be used to develop the database of this application to store data persistently.

User stories:

- Basic functionalities
    1. As a budget planner in life, I want to have an e-budget book in my mobile cell so that all record of expense is portable and easily maintained. My conditions of satisfaction are whenever and wherever I go, I can adding, removing or modifying my expense record.
    2. As a lazy budget planner in life, I want this app can recognize receipts by simply scanning so that I am free from the tedious typing process. My condition of satisfaction is that the quantity and price of each item, the sum of items price, and the extra cost such as tax and tips can be recognized correctly by scanning the receipt. The receipt should be in good condition and be scanned properly, which means rejection or incorrect information is allowed for receipts with destruction and rotation.

3. As a budget planner, I want this app can show not only items information but also some additional information about each purchase so that I can easily be recalled each expense when I review it. My condition of satisfaction is that additional information should include the vendor's name and the date when the purchase happened.

- Main functionalities
    1. As a tenant who shares an apartment with other people, I want this application is able to work out the expense of each person for the shared cost. My condition of satisfaction is that this application should be worked out the average cost of shared cost: utility bills, purchases of commonly used items, their tax and tips.
    2. As a tenant who shares an apartment with other people, I want this app can figure out the expense of each person when my roommates and I buy items with the different price shown on one receipt so that everyone knows how much they should pay to the person who paid the bill. My condition of satisfaction is shown in the following case. If Amy and I are in the same group and we have an unresolved receipt which was paid by Amy. This receipt includes an apple (2 dollars) I bought, a banana (1 dollar) Amy bought, and 1 dollars tax, this application should be worked out the amount of money I pay should pay to Amy which is 2.5 dollars since the 1 dollar tax shared by us.
    3. As a tenant who shares an apartment with other people, I want this app is able to calculate the expense of each person when my roommates and I buy the same item but different quantity, so that we can easily know how much we spent. My condition of satisfaction is shown in the following case. In one purchase paid by Amy, Amy bought 3 bananas (3 dollars) and I bought 2 (2 dollars), which ends up with the total expense 6 dollars after adding up 1 dollar tax. After specifying the number of items of each of us, the application is supposed to figure out I should pay to Amy 2.5 dollars.
    4. As a tenant who shares an apartment with other people, I want this app can work well when we buy different quantity of items, not only in some cases where items shown on receipts are countable, such as 5 apples or 10 oranges, but also where they are not, such as 1 bag of apples, or 1 pizza for 4 people. The difference between these two cases is in the former case, items are allocated to people by number (countable), while in the latter one, items are allocated by part (uncountable). My condition of satisfaction would be this application is assumed to be able to work out the average price of these items and assign it to each of involving persons.

- Additional functionalities
    1. As a social butterfly, I want an application to help me calculate my part of the bill when hanging out with friends, so that I won't mess myself up in complicated billing scenarios. My conditions of satisfaction include recognizing the receipt and

sending me a notification of my amount automatically, and inform the one who pays the bill when I've paid him or her.

2. As a social butterfly, I want to share my purchase with my friends online in real-time, so that the cost seems to be more worthy. My conditions of satisfaction include posting the names and photos of the goods online.
3. As a social butterfly, I want to give my friends suggestions on extra-value goods or restaurants, so that I can help my friends to save meaningless costs. My conditions of satisfaction include sharing my ranks of the vendors online, and inquiring the best vendors of a specific category.
4. As a tenant who shares an apartment with other people, I want this app can calculate the money flow between people when a purchase is paid by multiple persons so that we can get rid of this tedious and error-prone process of working this out manually. My condition of satisfaction is shown in the following case. Suppose there is a purchase involving 3 persons, Amy, Jane and I, where each of us bought 2, 3 and 1 banana(s) (1 dollar for each). After adding up 1.5 dollars tax to the total cost, the total expense of this purchase is 7.5 dollars paid by Amy (3 dollars) and Jane (4 dollars). I assume this application is capable of telling me that I should pay 0.5 dollars and 1.5 dollars to Amy and Jane respectively.

OCR API:

- Tesseract
  Tesseract is an open-source OCR API. It has unicode (UTF-8) support, and can recognize more than 100 languages "out of the box".
  Tesseract supports various output formats: plain-text, hocr(html), pdf, tsv, invisible-text-only pdf.

- OpenCV
  OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

Database (draft overview)**:**

- User table (User id [P], username, password, preferences)
- Friend table (User id [P], friend id)
- Group table (Group id [P], owner id)
- Group-User table (Group id, user id)
- Group-Order (Group id, order id, receipt image)
- Allocations (Allocation id [P], Item id, quantity, user id)
- Items (Item id [P], Order id, quantity)
- Orders (Order id [P], number)
  [P] for primary key