Nanuda: A Group Expenses Manager

1st Abia Herlianto
School of Computer Science
BINUS University
Team MONEY
Jakarta, Indonesia
abiaph@live.com

2nd Hugo Ravé
General Engineering
ESILV PARIS-LA DEFENSE
Team MONEY
Paris, France
hugorave07@gmail.com

3rd Nadya Hartanto
School of Information Systems
BINUS University
Team MONEY
Jakarta, Indonesia
ndhartanto1@gmail.com

***Abstract*—Users create a ”group” composed of either one person (themselves) or several people. Users then invite these people to the group. Users add expenses and split the costs with either all, some, or no members of the group. Users can categorise and add descriptions for each expense. Users can also customise how the costs are split. Users can see a summary of how much they owe to others and how much others owe to others. The summary of how much users owe changes as users spend and pay.**

***Index Terms*—expenses, finance, management**

Role	**Name**	**Task Description etc.**
User/Customer	Nadya Hartanto	Tests the application for bugs and issues. Provides feedback to the Development Manager. Also provides feedback on UX and UI quality.
Software Developer	Abia Putrama Herlianto	Implements software based on design. Fixes bugs and issues.
Development Manager	Hugo Ravé	Designs software based on requirements and feedback from User/Customer. Organises, schedules, and manages development.

I. Introduction

When there are shared expenses in a group, it can be complicated to split the costs and gather the money that each person should pay. The usual solution is to have one or several individuals cover the cost of the entire group, and from there the remaining group members would pay their share to those who covered the cost.

The problem that arises from this is keeping track of these expenses. This is especially true for groups that meet regularly; for example, a group of friends going on a trip. Keeping track of who covered what for how much and who owes whom how much can quickly get out of hand. The solution that we propose is a simple application where these expenses can be organised, viewed, and accessed by all members of a certain group.

The name Nanuda comes from Korean , meaning ’to divide (up), to split (up)’. The name was chosen because our own experiences in splitting expenses in Korea was the reason we thought up of the idea.

There are several existing applications that perform a similar function, with different individual features. These include Tricount and Sesterce.

1) **Tricount**

Fig. 1. Tricount

(Data from Google Play Store)
Rating: 4.8 stars
Downloads: 1 million+
Key features: Users can share a simple link to share their expenses. Members of a group can add expenses and see the balance. Expenses can be shared unevenly. Tricount works both offline and online, with logging in being optional. It also has both a mobile app and a website.

2) **Sesterce**

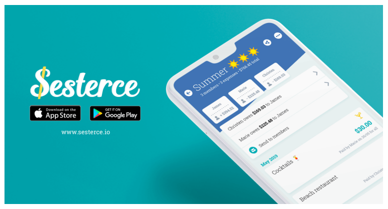

Fig. 2. Sesterce

(Data from Google Play Store)
Rating: 4.7 stars
Downloads: 5,000+
Key features: Similar to Tricount, users can join a single group and add expenses, keeping track of all bills and costs. Sesterce is completely anonymous with no login or email required. In addition, Sesterce allows the

addition of custom categories for expenses, the option to view statistics, and exporting all the data to a .csv file.

II. Requirements

1) **Loading Screen**: When the user opens the application, the loading screen will be displayed until the application has completed loading all the data from the server.
2) **Groups**: If there are existing groups, their names and brief descriptions will be displayed. If it is the first time the application has been opened, a sample group will be present. The user can add another group by using the add group button. The user can enter an existing group by clicking on it. The user can also modify and delete existing groups by long-pressing the group which will display a "Group Details" pop-up.
    - a) **Enter Group**: The user can enter an existing group by clicking the group in the list.
    - b) **Add Group**: Two options will be displayed. The user can join an group or make a new one.
        - i) **Make New Group**: A "New Group" form will be displayed. The user must input the name of the group, the currency used, and the names of the participants. The user can also optionally write a brief description of the group. The user can copy the link to the group for other participants to join the group.
        - ii) **Join Group**: A "Join Group" form will be displayed. The user must input a valid link of an existing group.
    - c) **Edit Group**: An "Edit Group" form will be displayed. Similar to a "New Group" form, the user can change the name of the group, the currency used, and the number and names of the participants.
    - d) **Delete Group**: The user can delete the group.
3) **Expenses**: If there are existing expenses, the name of the expense, the amount, and the date will be displayed. All participants in the group can see the expenses list and add another expense by clicking the "Add Expense" button. The user can also see the details of an expense by clicking on it.
    - a) **Add Expense**: An "Add Expense" form will be displayed. The user must input the title, amount, date, category, payer participant, and paid for participants of the expense. Users can split the expense equally automatically by default or edit the amounts manually by percentage or by amount.
    - b) **Expense Detail**: The details of the expense will be displayed: the title, amount, date, payer participant, and paid for participants. The exact amount each paid for participant owes for this expense is also displayed. Users can also edit this expense by clicking the "Edit" button and delete the expense by clicking the "Delete" button.
        - i) **Edit Expense**: An "Edit Expense" form will be displayed Similar to a "New Expense" form, the user can edit the title, amount, date, payer participant, and paid for participants.
        - ii) **Delete Expense**: The user can delete the expense.
4) **Balances**: A summary of how much is owed will be displayed. The information on who owes whom how much is also displayed e.g. A owes B 3 dollars, A owes C 7 dollars, C owes B 2 dollars. The total amount of money owed or owed to for each participant is listed e.g. in total, A owes 10 dollars, B is owed 5 dollars, C is owed 5 dollars. C is owed 5 dollars because A owes C 7 dollars, but C owes B 2 dollars.

III. Development Environment

A. Choice of Software Development Platform

1) **Platform**

Fig. 3. Android

For Nanuda, we have chosen the **Android** mobile operating system as the platform for several reasons:

- a) Nanuda needs to be on a mobile platform for quick access and easy use
- b) Android is the best-selling mobile OS worldwide
- c) Resources for development on Android are widely available and at no cost

2) **Programming Language**

Fig. 4. Java

For the development of Nanuda, we use the **Java** programming language. Java is a class-based, object-oriented programming language and one of the world's most popular and widely used programming languages. We chose to use Java for two reasons:

- a) Java is the language used to develop Android apps through the Android SDK
- b) We are familiar with Java and have used it in the past

3) **Cost Estimation**

For the development of Nanuda, all of the resources used are either free of charge or have both free and paid options. For the resources with free and paid options, we have decided to use the free option. As such, the development of Nanuda requires no cost.

Resource	Role Description	Cost
Android Studio	Android Development IDE	0
AWS Educate	Backend Server	0
Overleaf	Online, Collaborative LaTeX Writing Tool	0
GitHub	Remote Repository and Version Control System	0
GitHub Desktop	GUI for GitHub	0

4) **Development Environment**
   - **Android Studio Version 4.1**
     Android Studio was chosen because it is the official IDE for the Android operating system.
   - **AWS Educate**
     AWS Educate is Amazon's programme for students to use AWS services at no cost.
   - **Overleaf**
     Overleaf was chosen because of its popularity and the effectiveness of its features, namely simple, in-browser editing, immediate compilation, easy sharing and collaborative editing.
   - **GitHub**
     GitHub was chosen for Nanuda's remote repository because of our familiarity with it in comparison to other version control systems.
   - **GitHub for Desktop**
     GitHub for Desktop was chosen instead of the regular, CLI-based system to interact with GitHub because of its simplicity and intuitiveness.

B. *Software in use*

1) **Tricount**

Fig. 5. Tricount

Starts by setting up your account, sign up/login through different ways (Facebook, phone number,etc...). Create your group by choosing a name, purpose of the group (trip, group of friends, couple, family) then choose your currency, add the name of the members (up to 30 participants) and finally share the link of the group to the members and you're all set to start keeping up with your expenses.
Inside a group, you have two main pages: Expenses and Balance. Expenses is where you can see all the expenses of the group as well as the name of the group, who is in it, your total personal expenses and the total expenses of the group. An expense is presented on the screen like this: name of the expense, amount, currency, paid by who and date of the expense. You can click on it to see a picture of the object or receipt if it was added, modify the title of the expense, amount, date, paid by, and for who it was paid. You can also delete the expense. You can sort the expenses by different categories, such as amount paid, date of payment, payer and title.
The second main screen is Balance which shows who owes money and who is owed money. When you click on the owed money to someone, you can mark the debt as paid or you can invite them to pay you through a link sent via messaging platforms.
You also have the main menu screen of the app which allows you to see all the groups you're a part of and where you can add or delete groups.

2) **Sesterce**

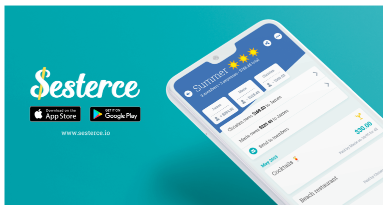

Fig. 6. Sesterce

The setup is pretty similar to create a group and add members.
There's only 1 main screen in Sesterce. It shows the group name, the number of members, the number of expenses, how much money was spent, who owes money or is owed money by whom. It also shows the list of all expenses sorted by date. The expenses display is pretty similar to Tricount as well, showing the title, the amount of the expense, who it is paid by and for whom. You can also add expenses. You have to chose who paid (one or multiple persons), the amount paid by each person, then select a title, add a date and choose a category for the expense (accommodation, groceries, transportation, you can even add your own category).

IV. Specifications

1) **Loading Screen**

```
display loading screen

if first time opened
    initialise sample group
else
    get groups data from backend server

go to groups screen
```

Fig. 7. Loading Screen Pseudocode

2) **Groups**:

```
display existing groups

if long press on group
    display group details popup
```

Fig. 8. Groups Pseudocode

a) **Enter Group**:

```
go to expenses screen
```

Fig. 9. Enter Group Pseudocode

b) **Add Group**:

```
display add group popup
```

Fig. 10. Add Group Pseudocode

i) **Make New Group**:

```
input group name
input description
choose currency
choose date
input participants

initialise group

add group to database

display group link
```

Fig. 11. Make New Group Pseudocode

ii) **Join Group**:

```
display join group form

input group link

get group from database

add group to local groups data

go to expenses screen
```

Fig. 12. Join Group Pseudocode

c) **Edit Group**

```
input group name
input description
choose currency
choose date
input participants

modify group

update group in database
```

Fig. 13. Edit Group Pseudocode

d) **Delete Group**

```
delete group from local groups data
delete group in database
```

Fig. 14. Delete Group Pseudocode

3) **Expenses**

```
get expenses from group
display expenses
```

Fig. 15. Expenses Pseudocode

a) Add Expense

```
input title
input amount
choose date
choose paidBy
choose paidForWhom

get title
get amount
get date
get paidBy
get paidForWhom

initialize new expense

add expense to database
```

Fig. 16. Add Expense Pseudocode

b) Expense Detail

```
get expense

display title
display currency
display date
display paidBy
display paidForWhom
```

Fig. 17. Expense Details Pseudocode

c) Edit Expense

```
get expense from database

input title
input amount
choose date
choose paidBy
choose paidForWhom

get title
get amount
get date
get paidBy
get paidForWhom

update expense in database
```

Fig. 18. Edit Expense Pseudocode

d) Delete Expense

```
get expense
delete expense
```

Fig. 19. Delete Expense Pseudocode

4) Balances

```
get expenses

define debtDetails

for each participant
     sum expenses
     for each other participant
          calculate debt
          if debt > 0
               add to debtDetails

for each participant
     display total expenses

for each debtDetails
     display debt details
```

Fig. 20. Balances Pseudocode