

# Requirement & Analysis Document

## 1. Introduction

### 1.1 Purpose of application

The project aims to create a personal finance application that serves the purpose of handling debt between participants in a certain group. The application users should be able to create accounts in the application and add other users into groups. The application simplifies debt between the participants in each group.

### 1.2 General characteristics of the application

- The application is an Android app written in Java.
- The application simplifies debt by subtracting, adding, and distributing debts between users in a group.
- Users in a group should be able to add new debts/transactions in the group and select other users to which the debt applies.
- Users should be able to see a sorted log of all transactions.
- Users should be able to add other users (by phone number) to a group.
- Users should be able to create an account with a name and phone number.

### 1.3 Scope of application

- No actual financial value will be transferred through the application. It will resemble a simplified ledger for personal debt between participants.
- The application will be limited to an offline prototype only. Users will be simulated and retrieve data from the local storage of the device.

### 1.4 Objectives and success criteria of the project

1. Users should be able to create an account.
  - a. With a name, a phone number, and a password.
2. Users should be able to create a group and add other people to the group.
  - a. Group has a name.
  - b. Users can be added to the group by phone number.
3. Users should be able to add debt to others.
  - a. Either individually or collectively as a group.
4. It should be possible to see the most recent transactions.

## 2. Requirements

### 2.1 Backlog

USxx, [Template Title]

1. Implemented?
  - No
2. Description
  - As a [type of user], I want to [] so I can [].
3. Acceptance criteria
  - Functional
    - i.
  - Non-functional
    - i. Accessibility
    - ii. Portability
    - iii. Monitorability
    - iv. Personalization
    - v. Performance
    - vi. Maintainability
    - vii. Authorization
    - viii. Localization
    - ix. Testability

#### 2.1.1 High Priority

US01: Show debts for specific groups.

1. Implemented?
  - No
2. Description
  - As a user, I want to see all the added debts in a certain group so I can understand my debt-situation.
3. Acceptance criteria
  - Functional
    - i. There exists a way to reach the View for the log of debts.
    - ii. There exists a View for the log of debts.
    - iii. The debts must be logged somewhere to be able to display them.
    - iv. The debts are sorted by date and viewable.
    - v. There exists a way to exit the View.
    - vi. There exists a way to get more information about a specific debt.
  - Non-functional
    - i. Accessibility

1.

## US02: Create an account

1. Implemented?
  - No
2. Description
  - As a user, I want to be able to create an account so I can use the application.
3. Acceptance criteria
  - Functional
    - i. If the user is not logged in, the user is supplied with a button that leads the user to create an account (or login).
    - ii. The user is prompted to enter Phone number, Name, and Password on account creation.
    - iii. The user can complete the account creation and the information is saved/stored and an account is created.
  - Non-functional
    - i. Accessibility
    - ii. Portability
    - iii. Monitorability
    - iv. Personalization
    - v. Performance
    - vi. Maintainability
    - vii. Authorization
    - viii. Localization
    - ix. Testability

## US03: Log in

1. Implemented?
  - No
2. Description
  - As a registered user, I want to log in so I can access my account and use the application.
3. Acceptance criteria
  - Functional
    - i. If the user is not logged in, the user is supplied with a button to access a login page. (Or create an account).
    - ii. On the login page, a user is prompted to enter the Phone number and password and can subsequently log in.
    - iii. There exists a way to log out of the account.
  - Non-functional
    - i.

## US04: Create groups. (WEEK 1)

1. Implemented?
  - No
2. Description
  - As a user who owes/lends money to other user(s), I want to create a group of users so that I can keep my/their debt on track.
3. Acceptance criteria
  - Functional
    - i. There exists a group creation View.
    - ii. There exists a way to navigate to the group creation View.
    - iii. In the group creation View, the user can specify the name of the group.
    - iv. In the group creation View, the user can invite other users to the group.
    - v. Once the group is created, the user should be able to see it somewhere.
  - Non-functional
    - i. Groups should be created towards a database abstraction to easily switch between database structures and models. (Portability)

## US05: Add user(s) to groups. (WEEK 1)

1. Implemented?
  - No
2. Description
  - As a user who just met my SO, I want to add them to a group so I can add debt to them.
3. Acceptance criteria
  - Functional
    - i. There exists a way to add users to a group.
      1. The group owner should be able to add other users to the group by:
        - a. Phone number
        - b. From the accounts in-app contacts
        - c. (From device contacts)
    - ii. Multiple users should be able to be added at once.
    - iii. There should be a way for added users to accept or decline the invite to the group.
  - Non-functional
    - i.

## US06: Add debt to user(s) in a group. (WEEK 1)

1. Implemented?
  - No.
2. Description

- As a user, I want to add debt to one or more users in a group so that I can track it in the future.
- 3. Acceptance criteria
  - Functional
    - i. There exists a View where the user can create a debt specification.
    - ii. There is a way in the group view to navigate to the debt specification View.
    - iii. The user can add multiple users to add debt to.
    - iv. The user can specify the total amount of debt.
    - v. The user can specify the reasoning behind the debt.
    - vi. The user can choose to split the debt to all users selected.
    - vii. The user can send the debt specification to the group/all users selected.
  - Non-functional
    - i.
    - ii. Accessibility
    - iii. Portability
    - iv. Monitorability
    - v. Personalization
    - vi. Performance
    - vii. Maintainability
    - viii. Authorization
    - ix. Localization
    - x. Testability

## US07: Debt simplifier in Group.

- 1. Implemented?
  - No.
- 2. Description
  - As a user, I want my debts to different users in a specific group to be calculated to what I owe every single specific user, so I can pay them.
- 3. Acceptance criteria
  - Functional
    - i. If a debt is added to a group or a user in a group, from a user, the exact debt to each user is calculated.
    - ii. There exists functionality to simplify the debt between users.
  - Non-functional
    - i. Accessibility
    - ii. Portability
    - iii. Monitorability
    - iv. Personalization
    - v. Performance
    - vi. Maintainability
    - vii. Authorization
    - viii. Localization
    - ix. Testability

## US08: Settle debt between users in a group. **(WEEK 1)**

4. Implemented?
  - No.
5. Description
  - As a user, I want to settle an existing debt between one or more users in a group so that I am no longer indebted.
6. Acceptance criteria
  - Functional
    - i. There exists a View where the user can select and settle an existing debt which can be accessed by the user.
    - ii. The user can specify how much of the debt to settle.
    - iii. The user cannot settle more than the existing debt amount.
    - iv. The debt is updated after a settlement.
  - Non-functional

## US16: ModelEngine for developers. **(WEEK 1)**

7. Implemented?
  - No.
8. Description
  - As a developer I want to interact with the Model Package through a facade Design pattern so that I can rely on abstraction and not implementation.
9. Acceptance criteria
  - Functional
    - i. There exists a ModelEngine.
    - ii. There exists an interface that ModelEngine implements.
  - Non-functional

## 2.2 Medium Priority

### US09: Activity feed.

1. Implemented?
  - No
2. Description
  - As a user, I want to be able to access a page so I can get a general understanding of my usage and to easily get to different views of the application.
3. Acceptance criteria
  - Functional
    - i. When the user is logged in to the application, it is greeted by an Activity feed.
    - ii. The will show
  - Non-functional
    - i. Accessibility
    - ii. Portability

- iii. Monitorability
- iv. Personalization
- v. Performance
- vi. Maintainability
- vii. Authorization
- viii. Localization
- ix. Testability

## US10: Add personal contacts

1. Implemented?
  - No
2. Description
  - As a user with an account, I want to be able to add personal contacts so that I can save their information and quickly access them in groups.
3. Acceptance criteria
  - Functional
    - i. There exists a contacts View that can be accessed.
    - ii. On the contacts-page, all added contacts are listed.
    - iii. On the contacts page, there exists a way to add a new contact.
  - Non-functional

## US11: Access contacts on the device

1. Implemented?
  - No
2. Description
  - As a user, I want to be able to add a new person through the inbuilt contacts (in the phone) so that I don't have to manually write in their information.
3. Acceptance criteria
  - Functional
    - i. Able to see contacts.
    - ii. Able to select a contact.
    - iii. The app should be able to see if it is already an existing contact.
    - iv. The app should verify that the added contact has an account on the app.
  - Non-functional
    - i.

## 2.3 Low Priority

### US12: My profile page

1. Implemented?
  - No
2. Description
  - As a user, I want to be able to see and edit my profile to keep it updated.
3. Acceptance criteria

- Functional
  - i. Can I see my name on my profile?
  - ii. Can I see my picture on my profile?
  - iii. Can I see a biography in my profile?
  - iv. Can I edit the information on my profile page?
    - 1. Name
    - 2. Image
    - 3. Biography
- Non-functional
  - i. Monitorability
    - 1. Can I always see my name on my profile page?
    - 2. Can I always see my image on my profile page?
    - 3. Can I always see my biography on my profile page?
    - 4. Can I easily change to edit-mode on my profile page?
  - ii. Security
    - 1. If text fields are left empty, will I not be able to save the edited information then?
    - 2. Are special signs allowed?

### US13: Graph functionality for debt in each group

- 4. Implemented?
  - No
- 5. Description
  - As a user, I want to be able to see all the debts in a group displayed in a graph so that I can more easily understand it.
- 6. Acceptance criteria
  - Functional
    - i. There is a way to navigate to the group-specific debt-graph.
    - ii. All the debts in the groups are displayed in a graph.
    - iii. There is also a graph for displaying debt distribution by category.

### US14: Graph functionality for debt for the logged-in account.

- 7. Implemented?
  - No
- 8. Description
  - As a user, I want to be able to see all the debts to individuals displayed in a graph so that I can more easily understand it.
- 9. Acceptance criteria
  - Functional
    - i. There exists a way to navigate to the graph.

### US15: Debt simplifier between individuals across groups

- 4. Implemented?
  - No.



#### 5. Description

- As a user, I want my debts to a specific user across all mutual groups to be calculated to what I owe that user in total, so I can pay that user.

#### 6. Acceptance criteria

- Functional
  - i. If a debt is added to a group or a user in a group, from a user, the exact debt to each user is calculated.
  - ii. There exists functionality to simplify the debt between users.
- Non-functional
  - i.

### US17: Notification of actions involving debt

#### 7. Implemented?

- No.

#### 8. Description

- As a user, I want to be notified of all actions done to the debts that affect me so I won't miss it.

#### 9. Acceptance criteria

- Functional
  - i. If a debt is added to a user or more, a notification will be sent to all users affected by it.
  - ii. If a debt is settled between two or more users, a notification will be sent to all users affected by it.
- Non-functional

## 2.2 Definition of Done

- The code is peer-reviewed.
- Task has met the acceptance criteria.
- The code has passed JUnit tests.
- The application builds and compiles.
- The code is documented (JavaDoc).
- Merged into master-branch.
- Updated UML.

## 2.3 GUI

- Sketch

## 3. Domain model