**TRD - E-WALLET**

|  |  |
| --- | --- |
| **PRD** | - |
| **Owner** | Helmi |
| **Engineer** | Helmi |
| **Advisor** | Merah putih CTO |
| **Approval** | Merah putih CTO |
| **Status Work** | IN PROGRESS |

Revision

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 1.0.0 | 22/02/2023 | Helmi Susanto | * Initial document * Background * Goals * Milestone * Solution * Task breakdown |

Background

Transaction with cash is a old method that used since couple of thousand years ago and that method isn’t handy for the users. The user problem are

1. User have to save in the pocket much cash
2. User have to manual calculate if want to pay or transfer to another user.

Goals

Design Goals

1. User can make a deposit transactions.
2. User can make a withdraw transactions.
3. User can make a transfer among users.

Milestone

Milestone 1 Sprint-1: 22 February 2023 - 1 march 2023

* Feature: TRD
  + Initial document
  + Design goals
  + Define milestones
  + Define solution
  + Define task breakdown

Milestone 2 Sprint-2: 1 march 2023 - 15 march 2023

* Feature: e-wallet
  + Authentication
  + UI/UX
  + Manage users
  + Create wallet
  + Create Unit test

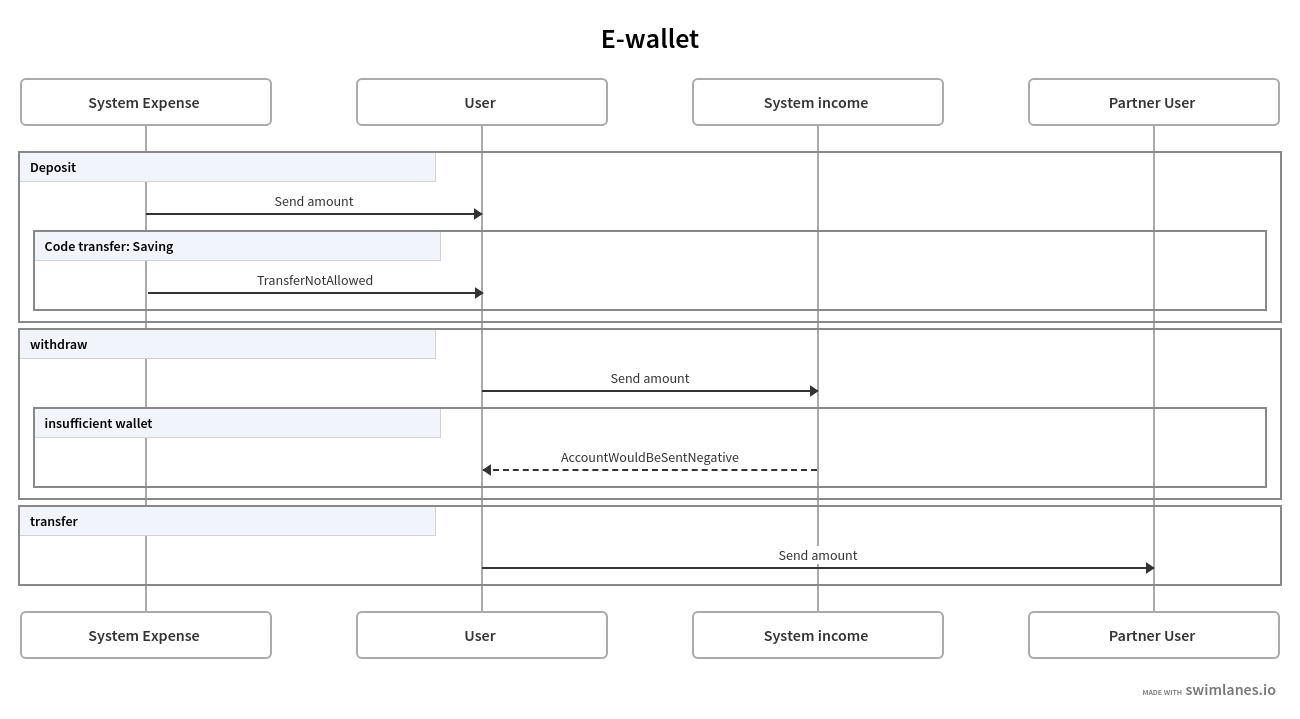
Solution

Summary

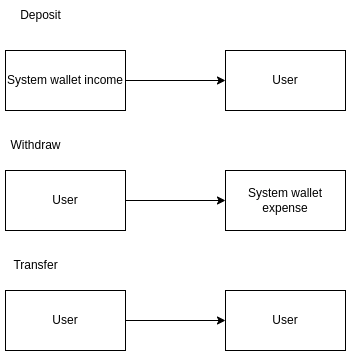
E-wallet is a solution pain point of the cash method.

Details

System Flow (Swimlanes)



Picture 1: Authentication ([link](https://swimlanes.io/u/mrRzClPi3))



Picture 2: Technical Flow

Operations

Summary

Details

To make a transactions among users In e-wallet the user should have deposit first to get a balance in the e-wallet.

As far as I know to handle transaction journal we can use double\_entry gem, please visit <https://github.com/envato/double_entry>to read the documentation.

|  |
| --- |
| **Double entry define identifier and scope**  require 'double\_entry'  DoubleEntry.configure do |config|  # Use json(b) column in double\_entry\_lines table to store metadata instead of separate metadata table  config.json\_metadata = true  config.define\_accounts do |accounts|  user\_scope = ->(user) do  raise 'not a User' unless user.class.name == 'User'  user.id  end  accounts.define(identifier: :savings, scope\_identifier: user\_scope, positive\_only: true)  accounts.define(identifier: :checking, scope\_identifier: user\_scope)  end  config.define\_transfers do |transfers|  transfers.define(from: :checking, to: :savings, code: :deposit)  transfers.define(from: :savings, to: :checking, code: :withdraw)  end  end |

|  |
| --- |
| **How to transfer between accounts**  DoubleEntry.transfer(  Money.new(20\_00),  from: one\_account,  to: another\_account,  code: :a\_business\_code\_for\_this\_type\_of\_transfer,  metadata: {key1: ['value 1', 'value 2'], key2: 'value 3'},  ) |

Testing

This document suggests having the following type of testing:

● Unit: Engineer will test each individual piece of code, which may include class or method.

● Regression: QA creates an automated test to run every time the engineer does a pull request to GitHub.

Task Breakdown

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Task** | **PIC** | **Ticket** | **Status** |
| 1 | PBI Authentication | Helmi |  | TBD |
| 2 | PBI UI/UX | Helmi |  | TBD |
| 3 | PBI Manage Users | Helmi |  | TBD |
| 4 | PBI Create Wallet | Helmi |  | TBD |
| 5 | PBI Create Unit test | Helmi |  | TBD |