Prj_4_FamilyBudget Analysis and Design Document

Version 1.0

1 March, 2017

Developed by:

Pop Ioana Andreea

936, 2017

Version History

Version	Description of Change	Author	Date
V_01	Modification of Use Cases section	Pop Ioana Andreea	1 March, 2017

Contents

Analysis and design Document

1 Functional Requirements

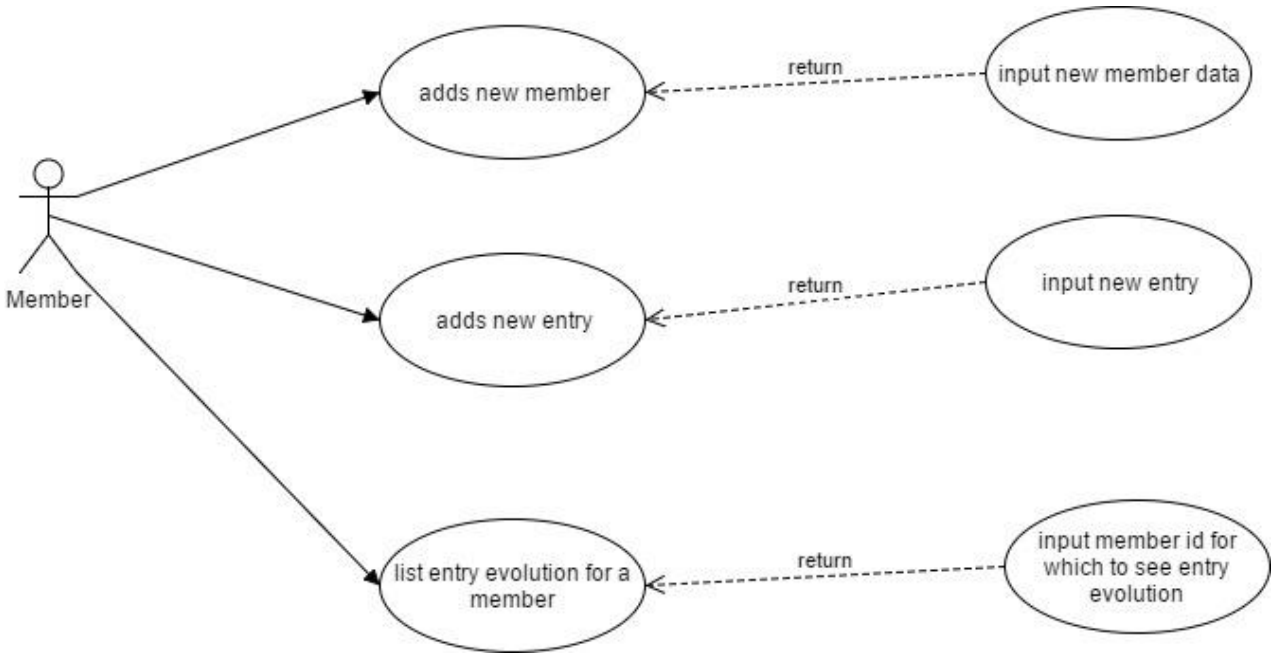
List the functional requirements (FR) of the system.

Section/ Requirement ID	Requirement Definition
FR_1.0.	The system shall allow the user to add a new member to the community.
FR_1.1	The system shall request information about the new member from to user.
FR_1.1.1	The system shall validate the input data provided by the user.
FR_1.2	The system shall inform the user if the member was added successfully or not.
FR_2.0	The system shall allow the user to add a new entry at the budget management.
FR_2.1	The system shall request information about the new budget entry from the user.
FR_2.1.1	The system shall validate the input data provided by the user.
FR_2.2	The system shall inform the user if the entry was added successfully or not.
FR_3.0	The system shall allow the user to view the evolution of incomes/costs for an existing member.
FR_3.1	The system shall request information from to user about the member whose incomes and costs should be listed.
FR_3.1.1	The system shall validate the input data provided by the user.
FR_3.2	The system shall inform the user if there are no entries to be shown.

2 Actors

The actors are the members of the community. There is no classification between members specified.

3 Use cases – diagram



3.1 Use case number 1 (Add a new member to the community)

Actors: Member

Description: Adds a new member to the community

Precondition: New member input data is valid

Postcondition: Member is successfully added; Message is displayed telling the user that the request action was completed successfully

User action	System response
1 Enters the add a new member option	
	2 Request new member data from the user
3 Inputs data	
	4 Validates input data
	5 Adds the new member
	6 Notifies user that the action was completed successfully

Exceptions:

- Input data is invalid
- Member already exists

3.2 Use case number 2 (Add a new entry to the budget)

Actors: Member

Description: Adds a new entry to the budget

Precondition: New entry input is valid

Postcondition: Entry is successfully added; Message is displayed telling the user that the request action was completed successfully

User action	System response
1 Enters the add a new entry option	

	2 Request new entry data from the user
3 Inputs data	
	4 Validates input data
	5 Adds the new entry
	6 Notifies user that the action was completed successfully

Exceptions:

- Input data is invalid
- Member does not exist

3.3 Use case number 3 (List entry evolution for a member)

Actors: Member

Description: Lists entry evolution for a member

Precondition: Member input data is valid

Postcondition: Entry evolution for a member is listed successfully

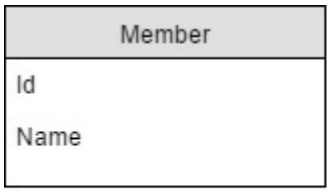
User action	System response
1 Enters the list entry evolution for a member option	
	2 Request member data from the user
3 Inputs data	
	4 Validates input data
	5 Gets entry evolution for the given member
	6 Lists the entry evolution for the given member

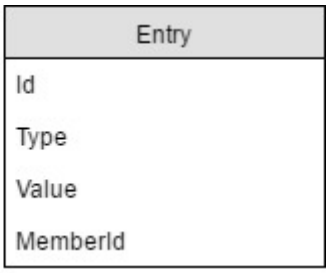
Exceptions:

- Input data is invalid
- Member does not exist

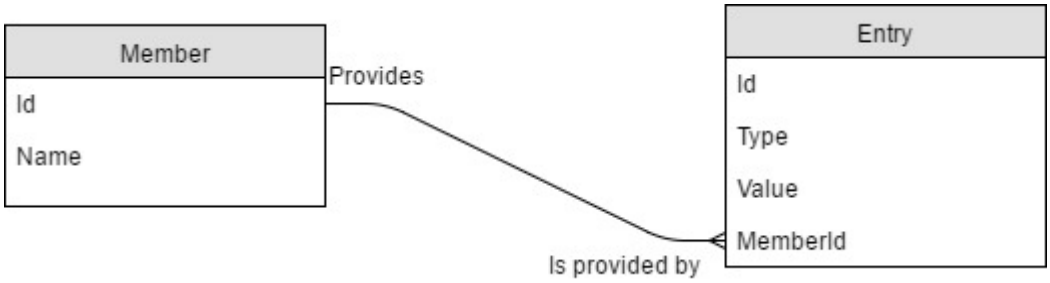
4 Analysis

4.1 Entities





4.2 Relations between entities



4.3 Attributes

For Member: - Id: Integer as id for class

- Name: String as member name

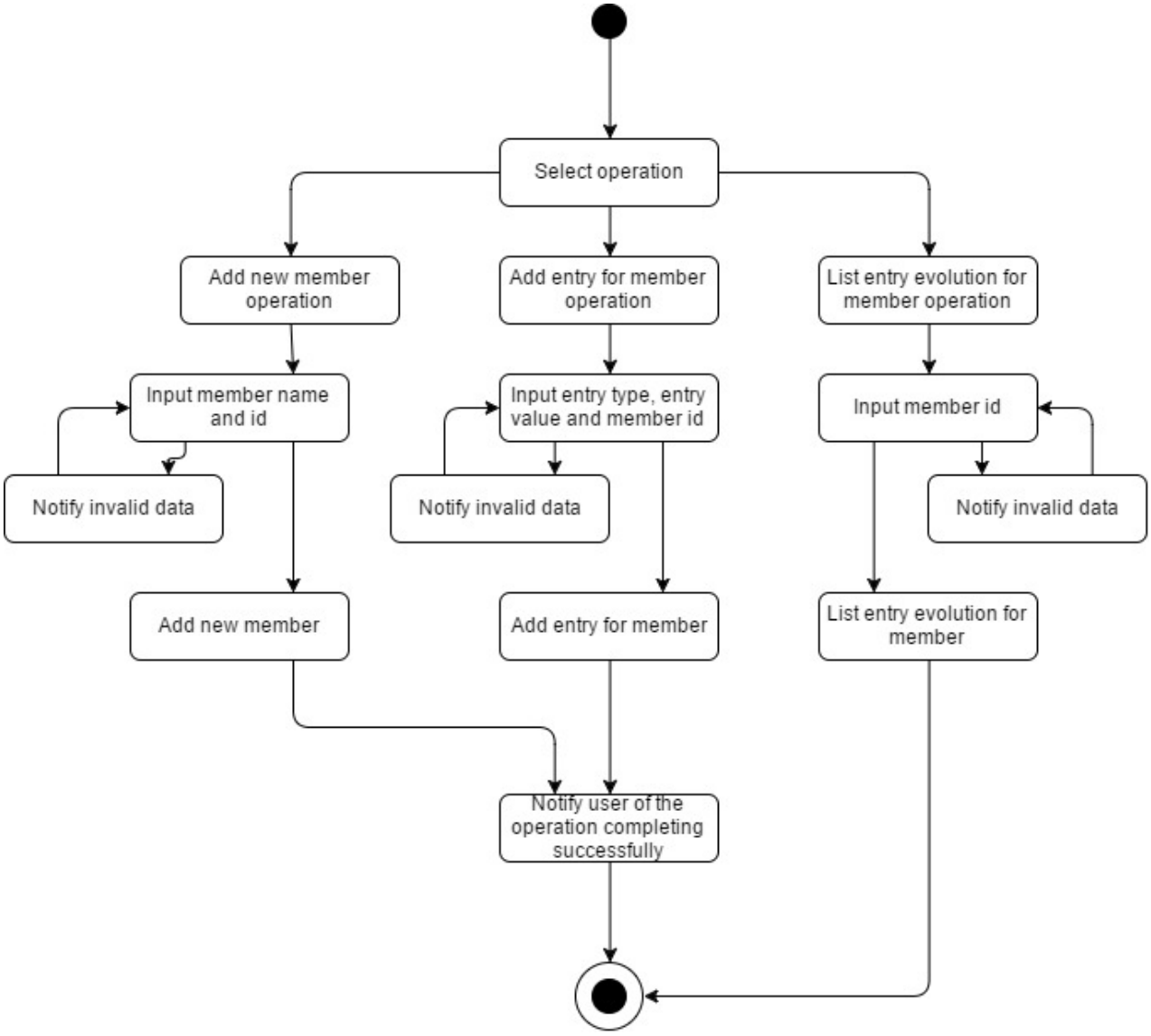
For Entry: - Id: Integer as id for class

- Type: Income/Cost as type of entry

- Value: Integer as value of entry

- MemberId: Integer as member id provided for entry

4.4 System behavior



4.4.1 Use case 1

Normal behavior: It adds the new member in the list of members
Exceptions: Input data is invalid/Member already exists

4.4.2 Use case 2

Normal behavior: It adds the new entry in the list of budget entries
Exceptions: Input data is invalid/Member does not exist

4.4.3 Use case 3

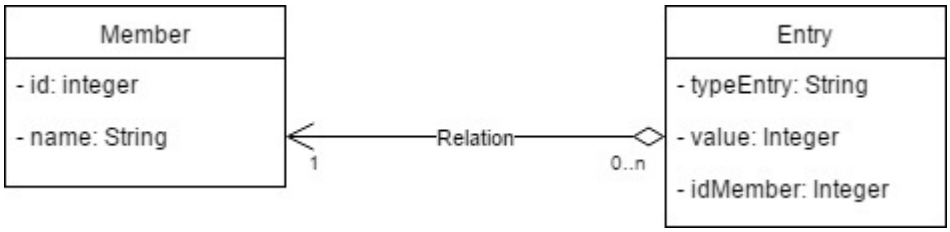
Normal behavior: Lists entry evolution for a member
Exceptions: Input data is invalid/Member does not exist

4.5 System events

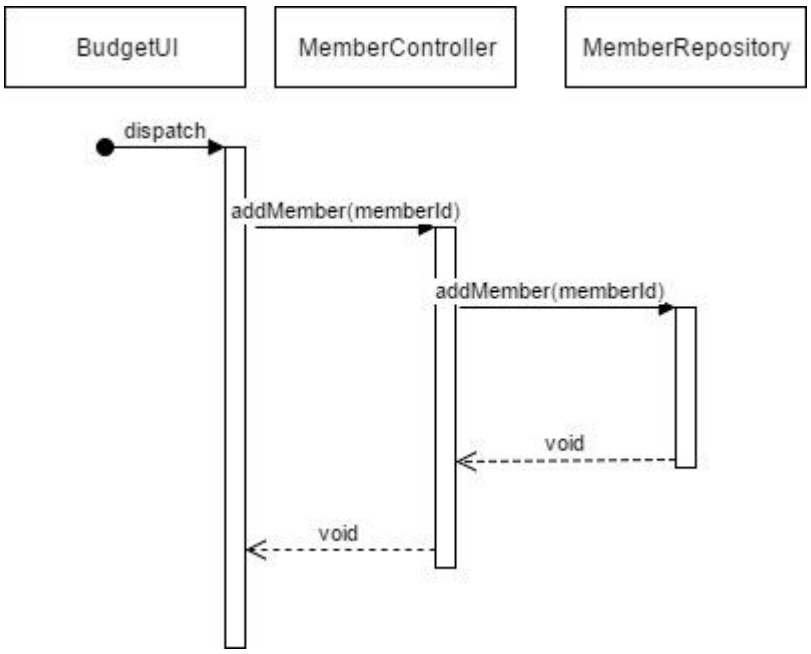
System events are the operations it allows the user to perform (Add new member/Add new entry for member/List entry evolution for member), requests for data (member id and name/entry type, value and member id/just member id) and notifying the user if the operation was executed successfully.

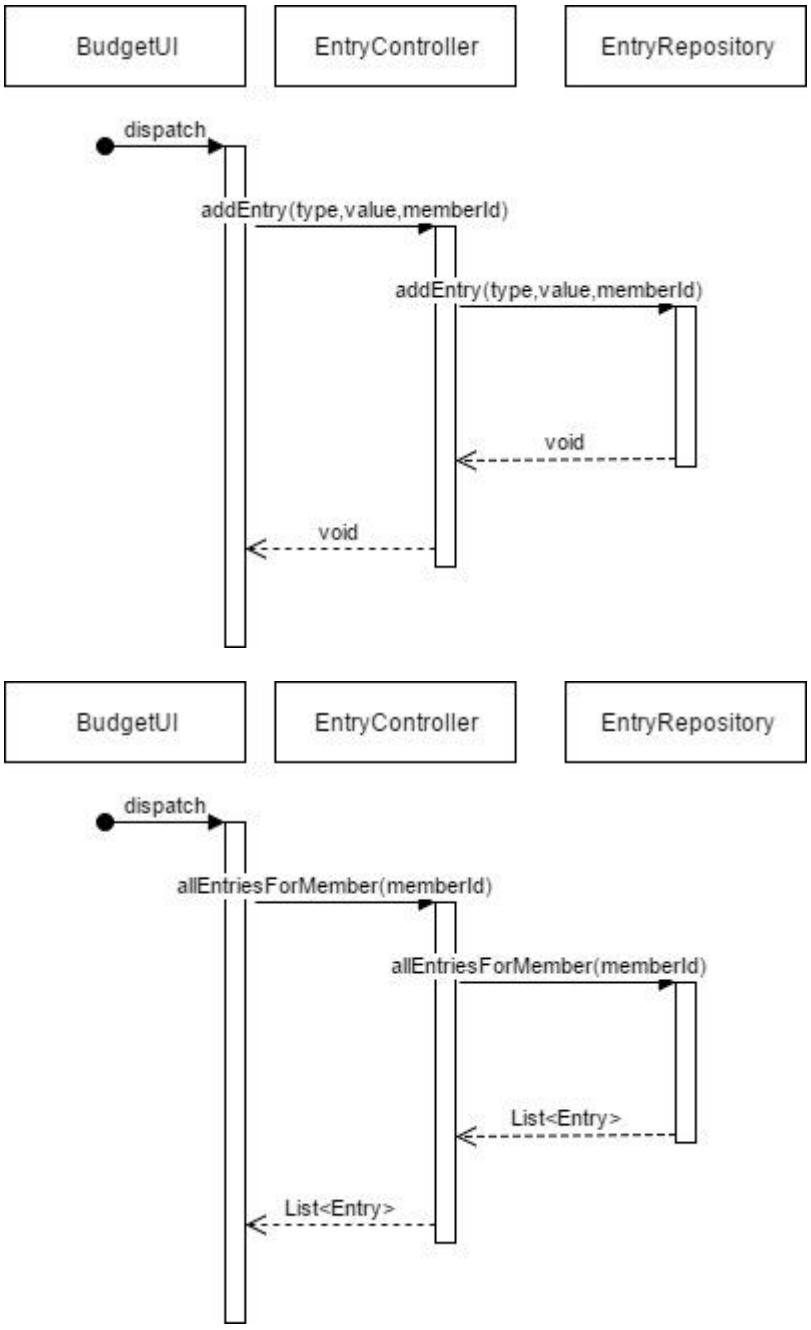
5 Design

5.1 Class diagram

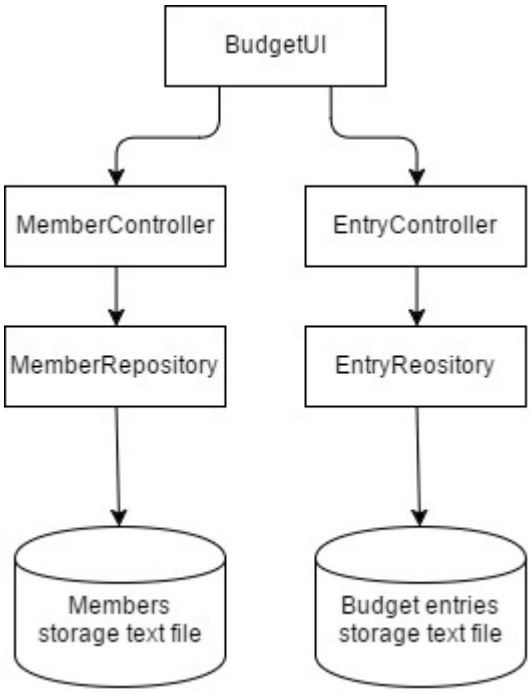


5.2 Sequence diagrams (for each use case)




BudgetUI
EntryController
EntryRepository
dispatch
addEntry(type,value,memberId)
addEntry(type,value,memberId)
void
void
BudgetUI
EntryController
EntryRepository
dispatch
allEntriesForMember(memberId)
allEntriesForMember(memberId)
List<Entry>
List<Entry>

5.3 GRASP


BudgetUI
MemberController
EntryController
MemberRepository
EntryReository
Members storage text file
Budget entries storage text file