

***Capstone Project***

**Report 4**

**Software Design Description**

*Personal Finance Management*

**Author: PFM Team**

**Supervisor: Mr. Nguyen Van Hien**

**Hanoi, 13th Sep, 2012**

Record of changes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Change Item** | **Description** | **By** | **Version** |
|  | All | Create document |  | 1.0 |
|  | Database design | Update content of database design |  | 1.1 |
|  | Sequence diagram | Add sequence diagram to some functions |  | 1.2 |
|  | System Context | Update content |  | 1.3 |

Table of Contents

[1. Design Overview 5](#_Toc342830081)

[2. System Architecture Design 6](#_Toc342830082)

[2.1. Choice of System Architecture 6](#_Toc342830083)

[**2.1.1.** **System Context** 6](#_Toc342830084)

[**2.1.2.** **Overall System Architecture** 7](#_Toc342830085)

[2.2. Discussion of Alternative Designs 8](#_Toc342830086)

[2.3. Description of System Interface 8](#_Toc342830087)

[3. Package Diagram 8](#_Toc342830088)

[3.1. PRESENTATION PACKAGE 8](#_Toc342830089)

[3.2. DATAREPOSITORY PACKAGE 9](#_Toc342830090)

[3.3. COMMON PACKAGE 10](#_Toc342830091)

[4. Detail Description of Components 12](#_Toc342830092)

[4.1. CRC Cards (Class-Responsibility-Collaborators) 12](#_Toc342830093)

[4.2. Class Diagram and Explanation 12](#_Toc342830094)

[**4.2.1.** **Incomes & Expenses management** 12](#_Toc342830095)

[**4.2.2.** **Schedule** 17](#_Toc342830096)

[**4.2.3. Borrowing & Lending** 21](#_Toc342830097)

[**4.2.4.** **Report** 23](#_Toc342830098)

[4. Sequence Diagram 24](#_Toc342830099)

[4.2. Incomes & Expenses management function 24](#_Toc342830100)

[**4.2.1.** **Add a record** 24](#_Toc342830101)

[**4.2.2.** **Edit a record** 25](#_Toc342830102)

[**4.2.3.** **View a record** 25](#_Toc342830103)

[**4.2.4.** **View a detail record** 26](#_Toc342830104)

[**4.2.5.** **Delete a record** 27](#_Toc342830105)

[4.3. Schedule function 27](#_Toc342830106)

[**4.3.1.** **Add a record** 27](#_Toc342830107)

[**4.3.2.**  **Edit a record** 28](#_Toc342830108)

[**4.3.3.**  **View a record** 29](#_Toc342830109)

[**4.3.4.**  **View a detail record** 29](#_Toc342830110)

[4.4. Borrowing & Lending management function 30](#_Toc342830111)

[4.5. Report function 32](#_Toc342830112)

[5. User Interface Design 34](#_Toc342830113)

[6. Database Design 34](#_Toc342830114)

# Design Overview

This document has been developed by PFM Group for Personal Finance Management Project.

This document describes the technical and UI design of the Personal Finance Management system. It includes the architectural design and the detailed design of common functions and business functions. It also includes the design of database model, and overall design of user interface.

The architectural design describes the overall architecture of the system, and the architecture of each main component and subsystem. It will describe the patterns being used, the role of each component and the role of the system in the working environment.

The detailed design describes static and dynamic structure for each component and function. It includes class diagrams, class explanations, and sequence diagrams of the main use cases.

The database design describes the relationship between entities, and details of each entity. It uses notations of SQL Server 2010 as it is the database server for developing the system.

The user interface design describes the layout of the system, and some design for the screens.

# System Architecture Design

## Choice of System Architecture

### **System Context**

|  |
| --- |
|  |

### **Overall System Architecture**

|  |
| --- |
|  |

## Discussion of Alternative Designs

## Description of System Interface

# Package Diagram

This diagram depicts the dependencies between the packages that make up a model. The Personal Finance Management System includes 3 packages: Presentation, Repository and Common. The relationship between packages is described as below diagram:

|  |
| --- |
|  |

## PRESENTATION PACKAGE

This package contains classes that will be used to help user can interact with Personal Finance Management System. This package includes the activities, custom views, adapters…

|  |
| --- |
|  |

## DATAREPOSITORY PACKAGE

This package will get data and transfer them to Presentation package. It includes: DataManager, IRepository, ScheduleRepository, and BorrowingAndLendingRepository…

|  |
| --- |
|  |

## COMMON PACKAGE

This contains common functions which can be used by all packages in the application. It contains functions which are used throughout the application like function for getting the response from server for a request, logging etc.

|  |
| --- |
|  |

# Detail Description of Components

## CRC Cards (Class-Responsibility-Collaborators)

### **SqlHelper class**

|  |  |
| --- | --- |
| **Class name:** SqlHelper | **Super class:** Object |
| **Subclasses:** <None> | |
| **Responsibilities:** create, select, insert, update and delete a table in SQLite database. | |
| **Collaborations:** SqlConnector, AccountProvider, ScheduleDetailViewActivity, ScheduleEditActivity, TabViewActivity, SyncSettingActivity, ScheduleRepository, EntryDetailViewActivity, EntryEditActivity, ReportChartActivity… | |

### **SynchronizeTask class**

|  |  |
| --- | --- |
| **Class name:** SynchronizeTask | **Super class:** AsyncTask<Void, Void, Void> |
| **Subclasses:** <None> | |
| **Responsibilities:** Create another thread for data synchronization. | |
| **Collaborations:** PfmApplication, EmailAccountCustomView, BorrowLendInsertActivity, ScheduleEditActivity, SyncSettingActivity, EntryEditActivity. | |

## Class Diagram

### **Incomes & Expenses management**

#### **Incomes and expenses management view**

|  |
| --- |
|  |

#### **Incomes and expenses management edit or delete**

|  |
| --- |
|  |

#### **Detailed incomes and expenses management view**

|  |
| --- |
|  |

### **Schedule**

#### **Schedule view function**

|  |
| --- |
|  |

#### **Schedule edit or delete function**

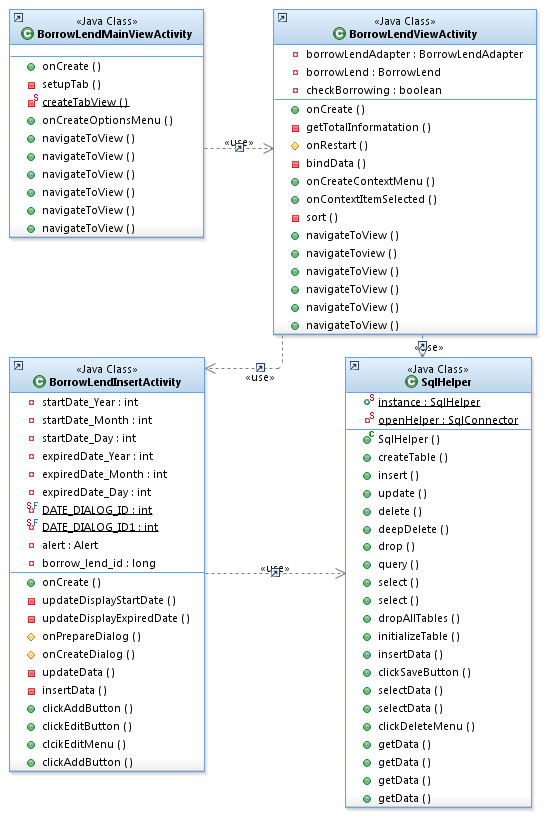
|  |
| --- |
|  |

#### **Class diagram for detailed Schedule view function**

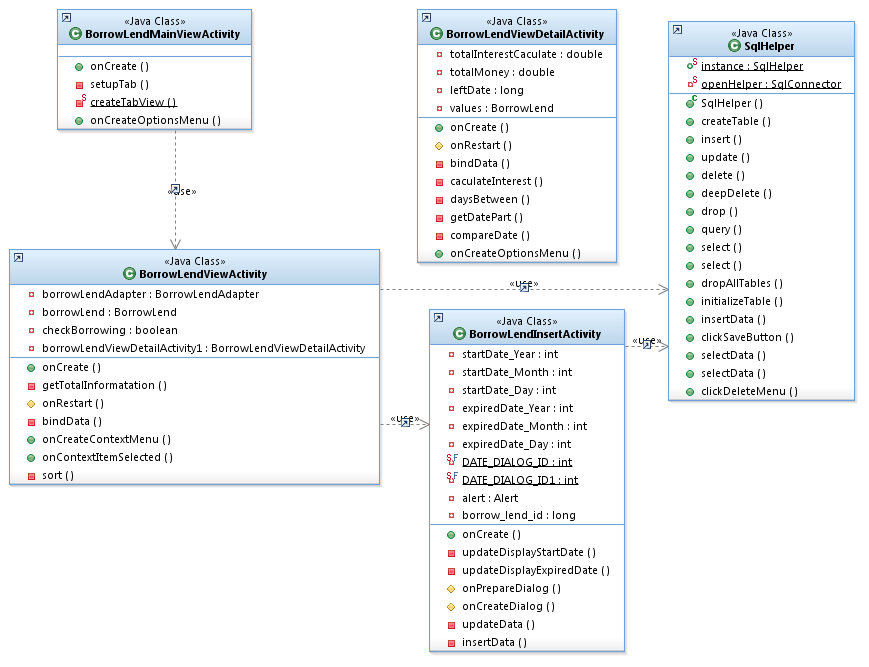
|  |
| --- |
|  |

### **Borrowing & Lending**

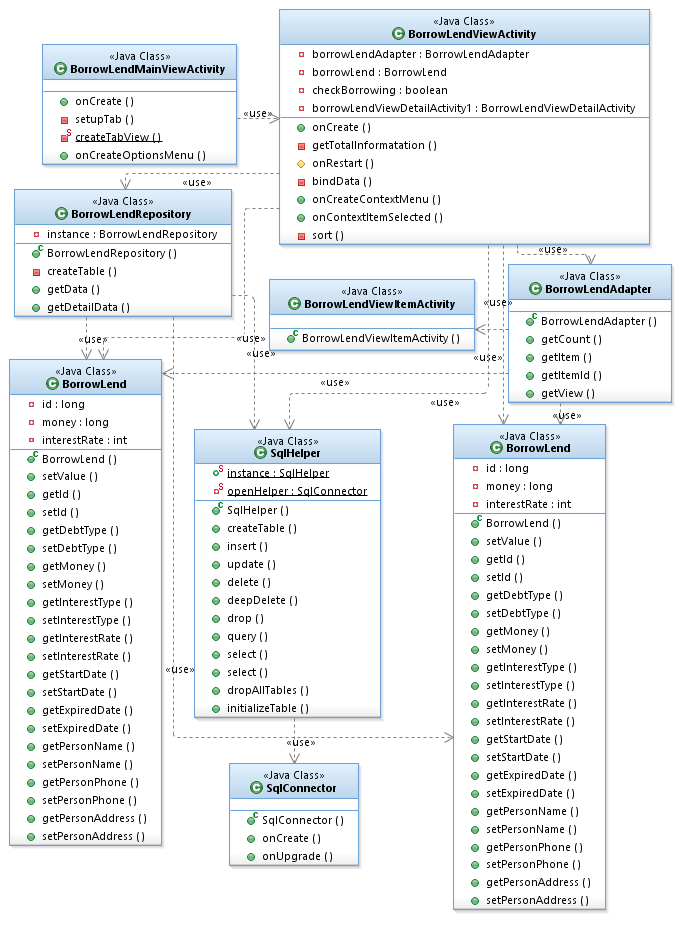
#### **Borrowing & lending add**



#### **Borrowing & Lending update**



#### **Borrowing & Lending View**



#### **Borrowing & Lending View Detail**

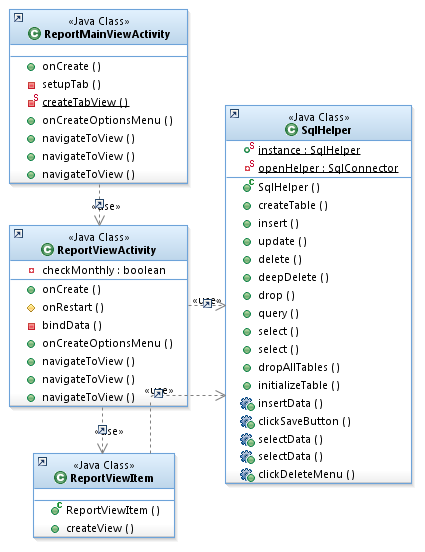
|  |
| --- |
|  |

#### **Borrowing & Lending Delete**

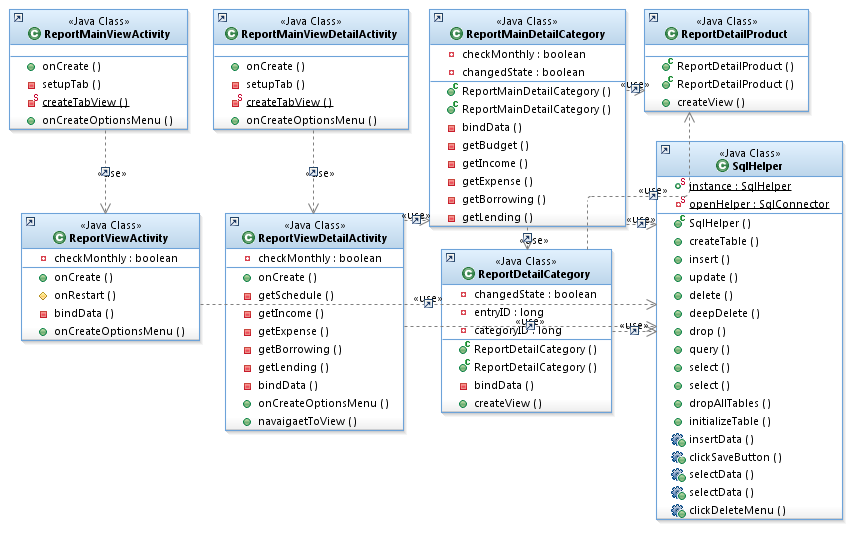


### **Report**

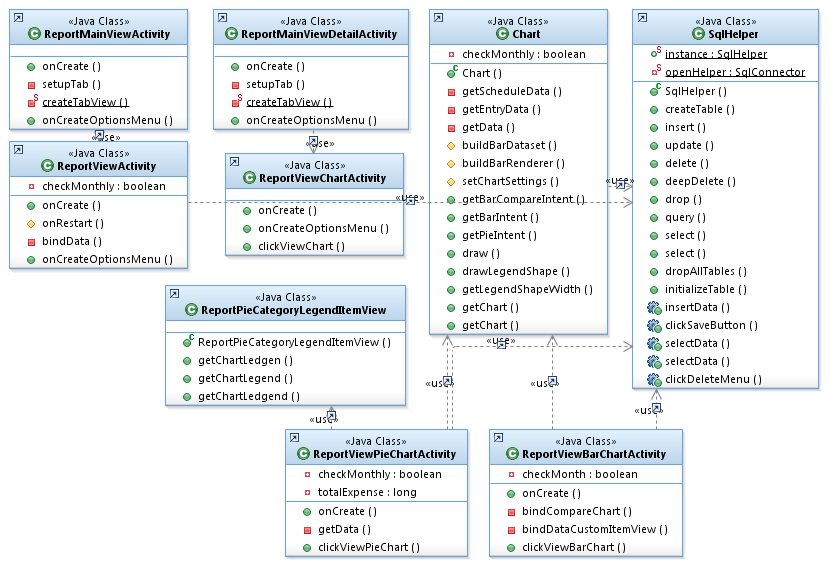
#### **Report View**



#### **Report View Detail**



#### **Report View Chart**



## Explanation

### **SqlHelper class**

#### **Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |
| Instance | SqlHelper | Public | A static field that helps using SqlHelper class without initializing an instance of SqlHelper many times. |
|  |  |  |  |

#### **Operations**

|  |  |  |
| --- | --- | --- |
| **public** **boolean** createTable(String tableName, String columnsInfo) | | |
| This method is used to create a table with specified name and column information if this table doesn’t exist. | | |
| Parameter | Type | Description |
| tableName | String | Name of table will be created |
| columnsInfo | String | Information of columns includes: column name, type of column, Primary Key |
| Return | Boolean | True if creation is successful. If the name of table is existing in local database or have some errors in creation progress, then this operation should return false |

|  |  |  |
| --- | --- | --- |
| **public long** insert(String tableName, String[] columnNames,  String[] columnValues) | | |
| Insert a record with specified table name, column names and column values to database. | | |
| Parameter | Type | Description |
| tableName | String | Name of table that the new record will be inserted |
| columnNames | String[] | A list of column names of inserted record |
| columnValues | String[] | A list of column values of inserted record |
| Return | Boolean | The row ID of the newly inserted record, or -1 if an error occurred |

### BorrowLendInsert Class

#### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |
| startDate\_Year | Int | Private |  |
| startDate\_Month | Int | Private |  |
| startDate\_Day | Int | Private |  |
| expiredDate\_Year | Int | Private |  |
| expiredDate\_Month | Int | Private |  |
| expiredDate\_Day | Int | Private |  |
| startDateEditText | EditText | Private |  |
| expiredDateEditText | EditText | Private |  |
| alert | Alert | Private | A static field that helps using alert user when they input mistake |
| debtTypeTextView | TextView | Private |  |
| debtTypeButton | ToogleButton | Private |  |
| nameEditText | AutoCompleteTextView | Private |  |
| phoneEditText | EditText | Private |  |
| addressEditText | EditText | Private |  |
| moneyEditText | EditText | Private |  |
| interestType | ToogleButton | Private |  |
| interestRate | EditText | Private |  |
| interestTypeString | String | Private |  |
| column | ArrayList<String> | Private | A static field that helps using record changed columns |
| valuesChanged | ArrayList<String> | Private | A static field that helps using record values of changed columns |

#### Operation

|  |  |  |
| --- | --- | --- |
| **public** **void** onCreate(Bundle savedInstanceState) | | |
| This method is used to prepare view | | |
| Parameter | Type | Description |
| saveInstanceState | Bundle |  |

|  |  |  |
| --- | --- | --- |
| **public** **void** updateData() | | |
| This method is used to update data to database after the user change information of a record | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** insertData() | | |
| This method is used to insert new data to database | | |
| Parameter | Type | Description |

### BorrowLendViewActivity

#### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |
| displayText | TextView | Private |  |
| borrowLendList | ListView | Private |  |
| borrowLendAdapter | BorrowLendAdapter | Private |  |
| borrowLend | BorrowLend | Private |  |
| checkBorrowing | Boolean | Private |  |
| totalMoneyTextView | TextView | Private |  |
| latesExpiredDateTextView | TextView | Private |  |
| values | ArrayList<Object> | Private |  |

#### Operation

|  |  |  |
| --- | --- | --- |
| 1. **public** **void** onCreate(Bundle savedInstanceState) | | |
| This method is used to prepare view | | |
| Parameter | Type | Description |
| saveInstanceState | Bundle |  |

|  |  |  |
| --- | --- | --- |
| **public** **void** bindData() | | |
| This method is used to get data from database and display them to view | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** sort() | | |
| This method is used to sort data following date time | | |
| Parameter | Type | Description |

### BorrowLendRepository Class

#### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |

#### Operation

|  |  |  |
| --- | --- | --- |
| **public** **void** ArrayList<Object> getData(String condition) | | |
| This method is used to get all data from database | | |
| Parameter | Type | Description |
| condition | String |  |
| Return | ArrayList<Objects> | Return list of BorrowLend model |

|  |  |  |
| --- | --- | --- |
| **public** **void** BorrowLend getDetailData(String condition) | | |
| This method is used to get detail data of a record from database | | |
| Parameter | Type | Description |
| Condition | String |  |
| Return | BorrowLend | Return BorrowLend model |

### BorrowLendViewDetail Class

#### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |
| totalInterestCaculate | double | Private |  |
| totalMoney | double | Private |  |
| leftDate | long | Private |  |

#### Operation

|  |  |  |
| --- | --- | --- |
| **public** **void** bindData() | | |
| This method is used to get data from database and display them to view | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** caculateInterest() | | |
| This method is used to calculate interest of a record | | |
| Parameter | Type | Description |

### ReportViewDetail class

#### Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |
| startDate | Date | Private |  |
| endDate | Date | Private |  |
| reportDetail | LinearLayout | Private |  |
| checkMonthly | Boolean | Private |  |

#### Operation

|  |  |  |
| --- | --- | --- |
| **public** **void** getSchedule() | | |
| This method is used to get data about Schedule from database to show them on the view | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** getIncome() | | |
| This method is used to get data about Income from database to show them on the view | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** getExpense() | | |
| This method is used to get data about Expense from database to show them on the view | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** getBorrowing() | | |
| This method is used to get data about Borrowing from database to show them on the view | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** getSchedule() | | |
| This method is used to get data about Lending from database to show them on the view | | |
| Parameter | Type | Description |

### Chart class

#### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Accessibility | Description |
| startDate | Date | Private |  |
| endDate | Date | Private |  |
| checkMonthly | boolean | Private |  |
| entryCategoryName | List<String> | Private |  |
| entryCategoryValue | List<Double> | Private |  |
| entryCategoryColor | List<String> | Private |  |
| scheduleCategoryName | List<String> | Private |  |
| scheduleCategoryValue | List<Double> | Private |  |

#### Operation

|  |  |  |
| --- | --- | --- |
| 1. **public** **void** getScheduleDate() | | |
| This method is used to get data about Schedule from database | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **void** getEntryData() | | |
| This method is used to get data about Expense from database | | |
| Parameter | Type | Description |

|  |  |  |
| --- | --- | --- |
| **public** **View** getBarChartCompareIntent(Context context, boolean checkMonthly, List<Date[]> dateList) | | |
| This method is used to make bar chart to compare data between month or week | | |
| Parameter | Type | Description |
| context | Context |  |
| checkMonthLy | boolean |  |
| dateList | List<Date[]> |  |
| Return | View |  |

|  |  |  |
| --- | --- | --- |
| **public** **View** getBarChartIntent(Context context, boolean checkMonthly, Date startDate, Date endDate) | | |
| This method is used to make bar chart to compare category in a month or week | | |
| Parameter | Type | Description |
| context | Context |  |
| checkMonthly | Boolean |  |
| startDate | Date | The begun date of a month or a week |
| endDate | Date | The finished date of a month or a week |

|  |  |  |
| --- | --- | --- |
| **public** **View** getPieChartIntent(Context context, boolean checkMonthly, Date startDate, Date endDate) | | |
| This method is used to make pie chart to compare category in a month or week | | |
| Parameter | Type | Description |
| context | Context |  |
| checkMonthly | boolean |  |
| startDate | Date | The begun date of a month or a week |
| endDate | Date | The finished date of a month or a week |

# Sequence Diagram

## Incomes & Expenses management function

### **Add a record**

|  |
| --- |
|  |

### **Edit a record**

|  |
| --- |
|  |

### **View a record**

|  |
| --- |
|  |

### **View a detail record**

|  |
| --- |
|  |



## Schedule function

### **Add a record**

|  |
| --- |
|  |

### **Edit a record**

|  |
| --- |
|  |

### **View a record**

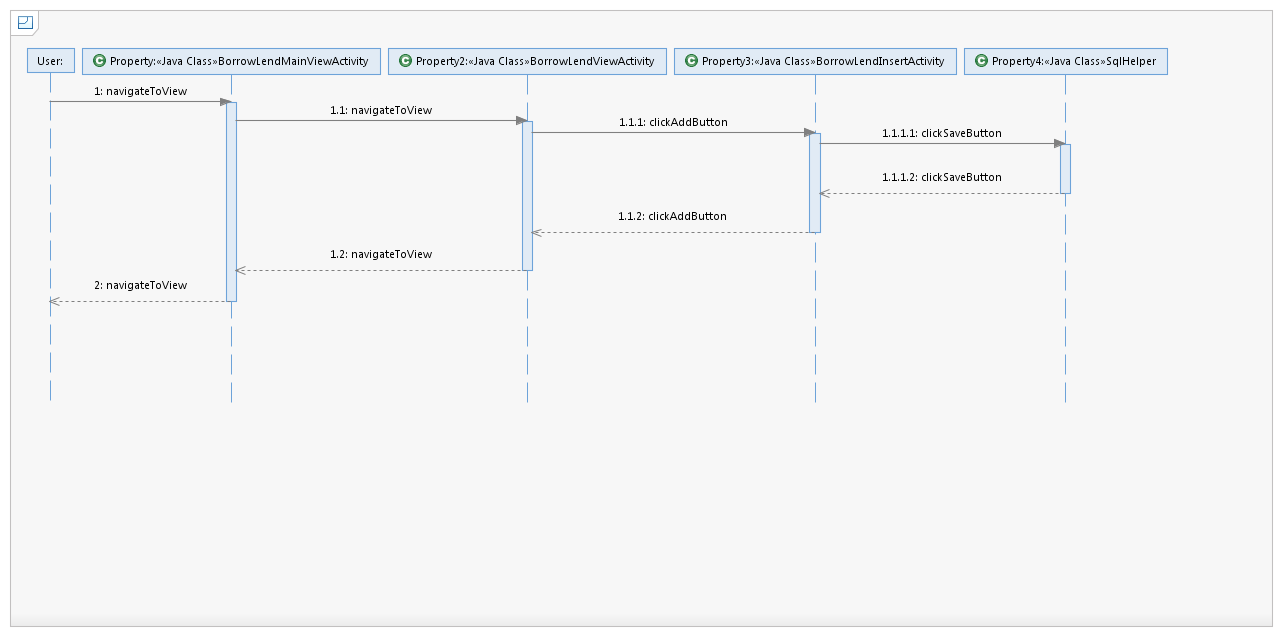
|  |
| --- |
|  |

### **View a detail record**

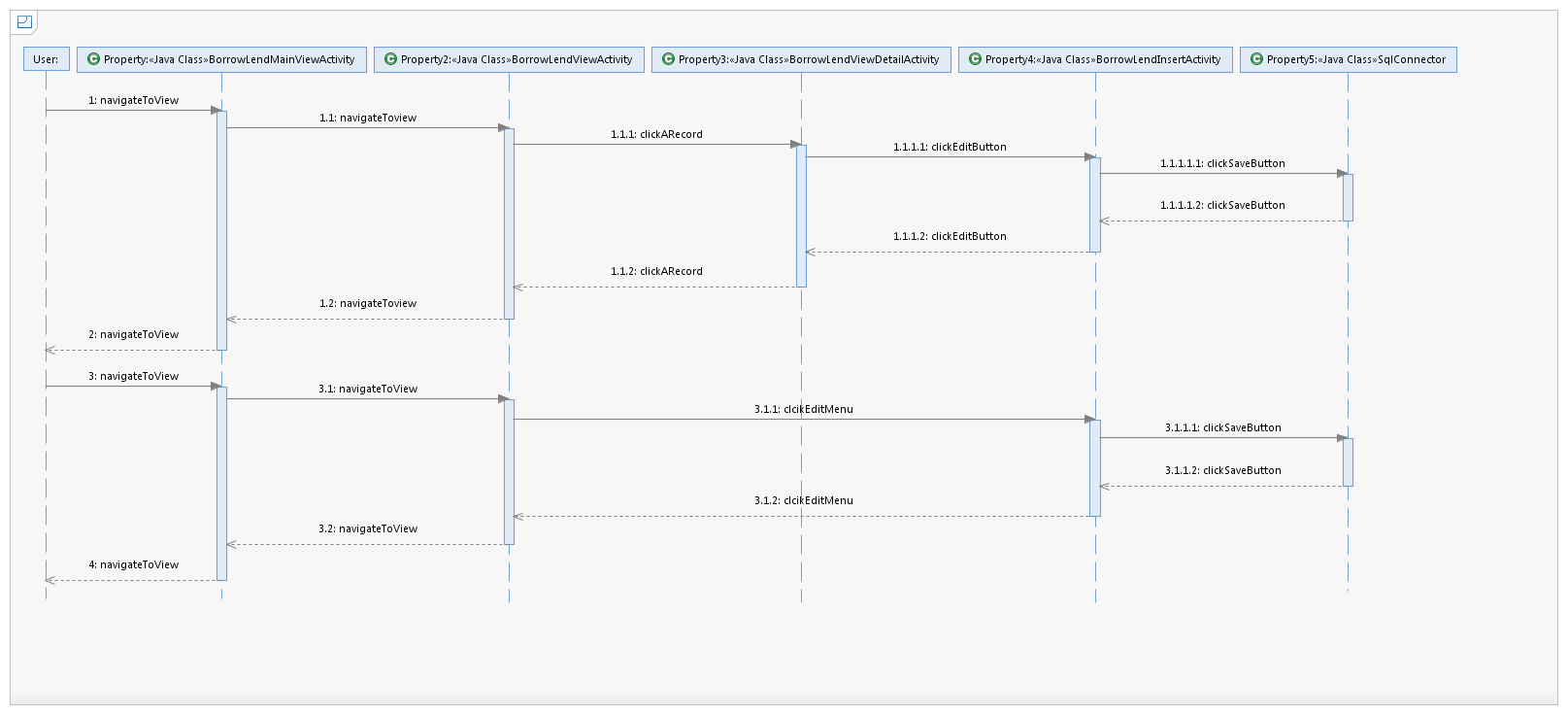
|  |
| --- |
|  |

## Borrowing & Lending management function

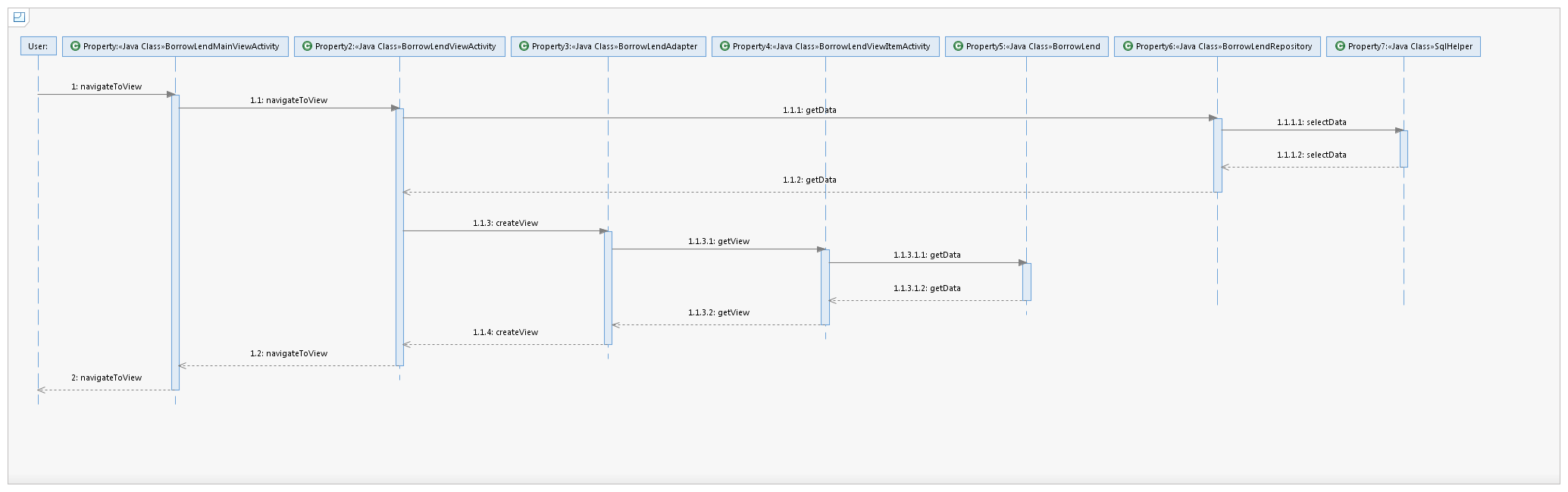
### **Add a record**

****

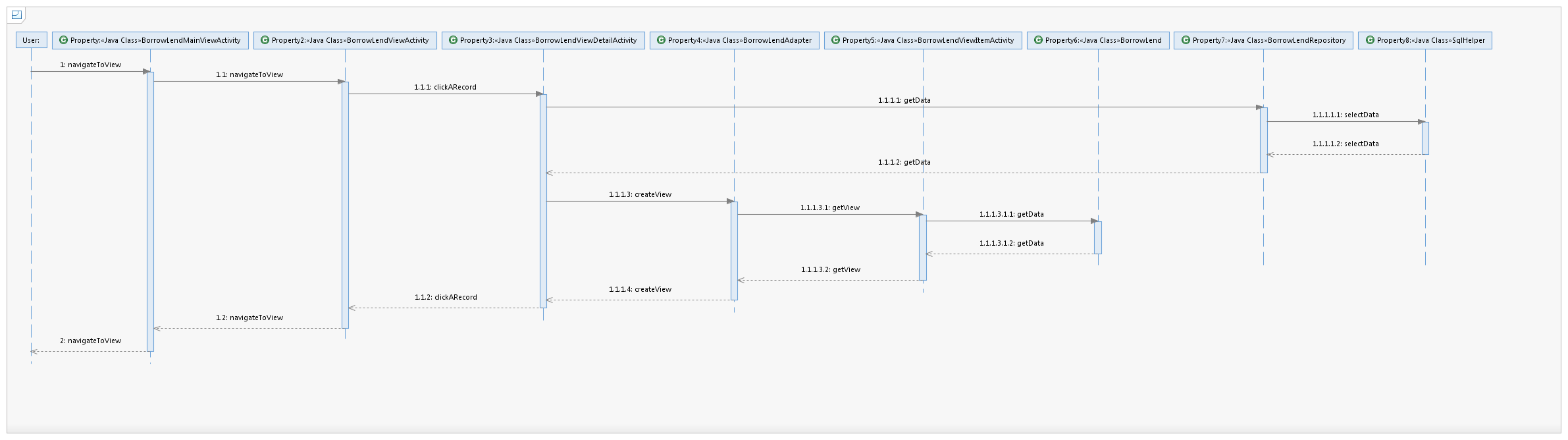
### **Edit a record**

****

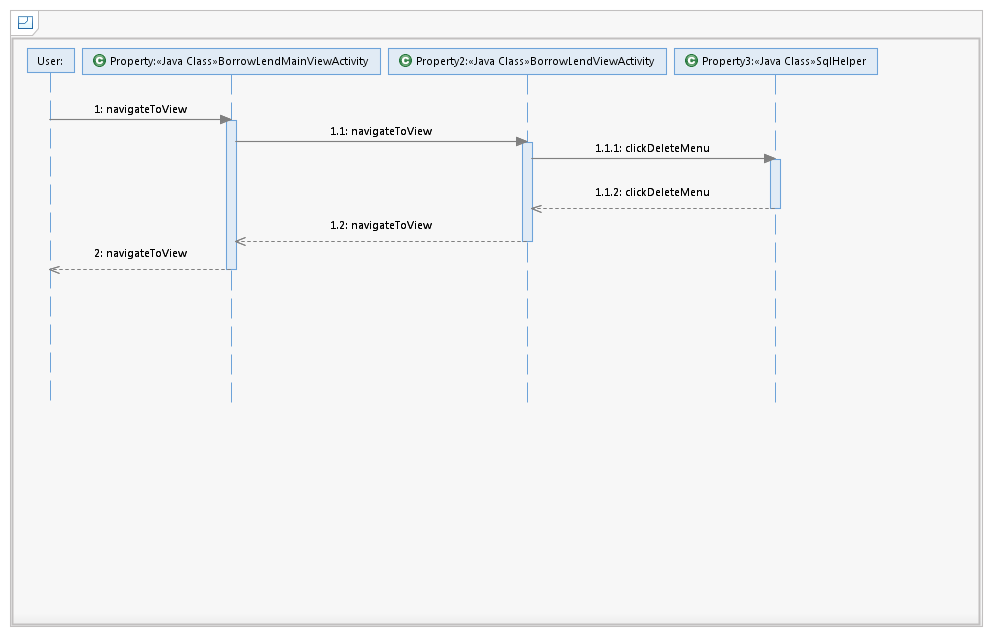
### **View a record**

****

### **View a detail record**

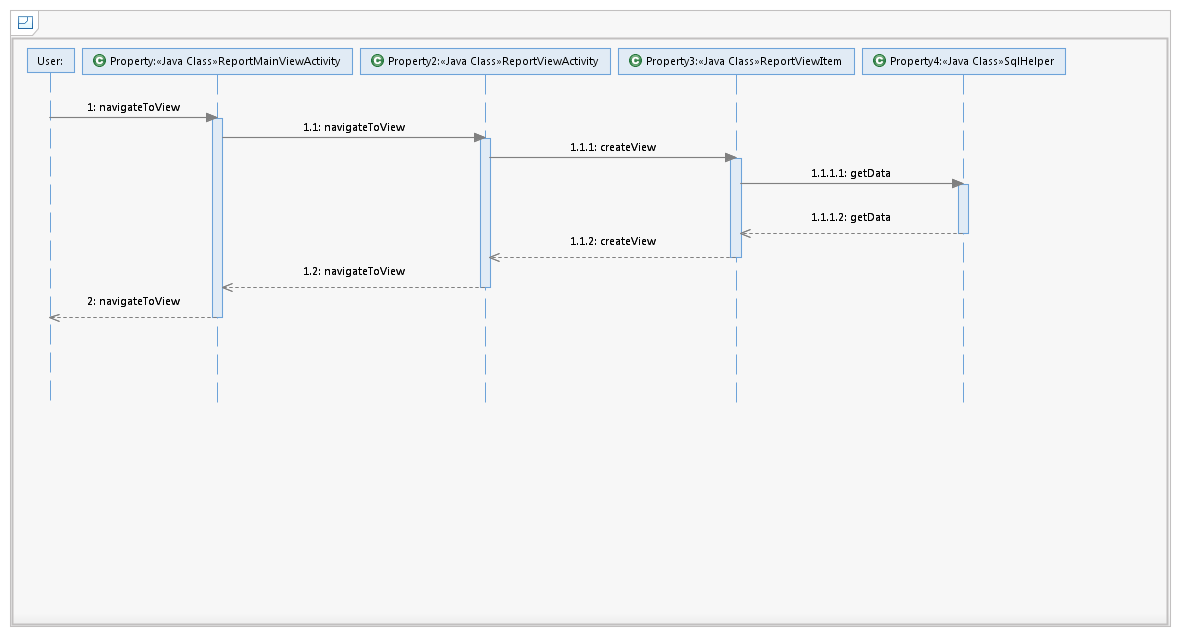
****

### **Delete a record**

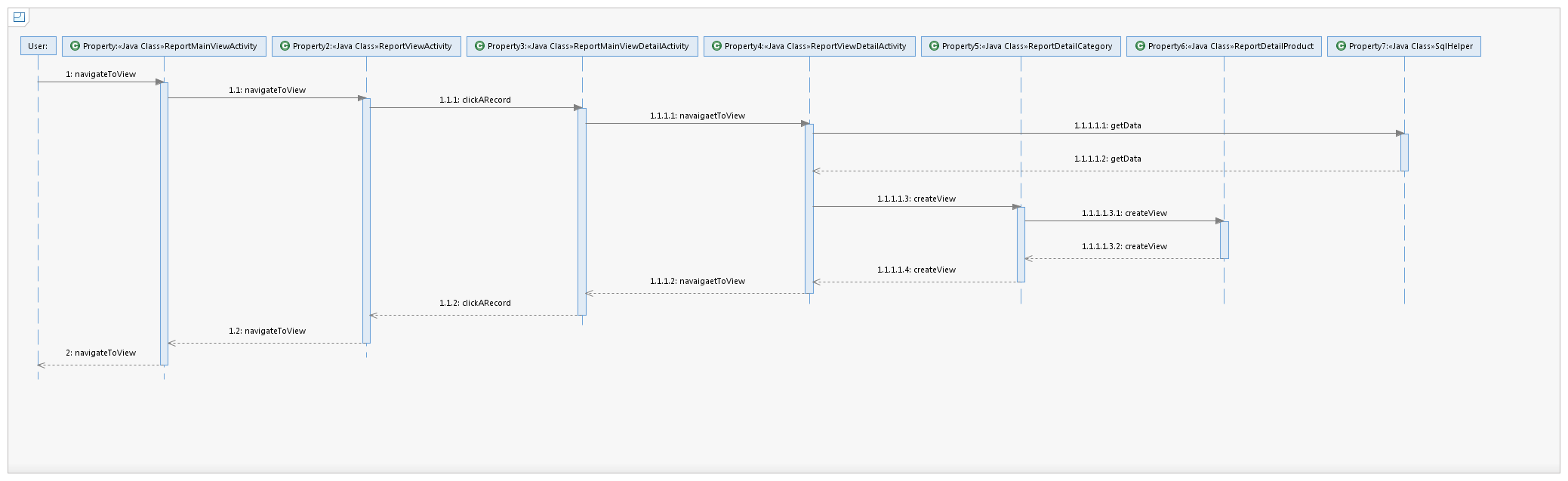


## Report function

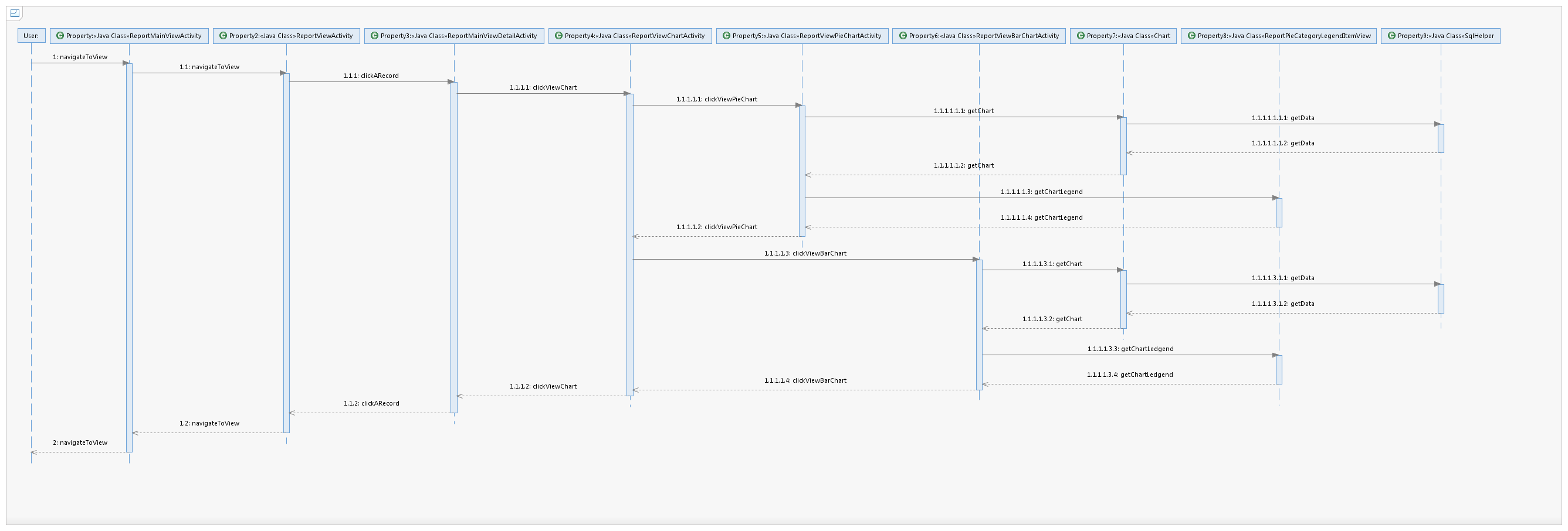
### **View a record**

****

### **View detail a record**

****

### **View chart**

****

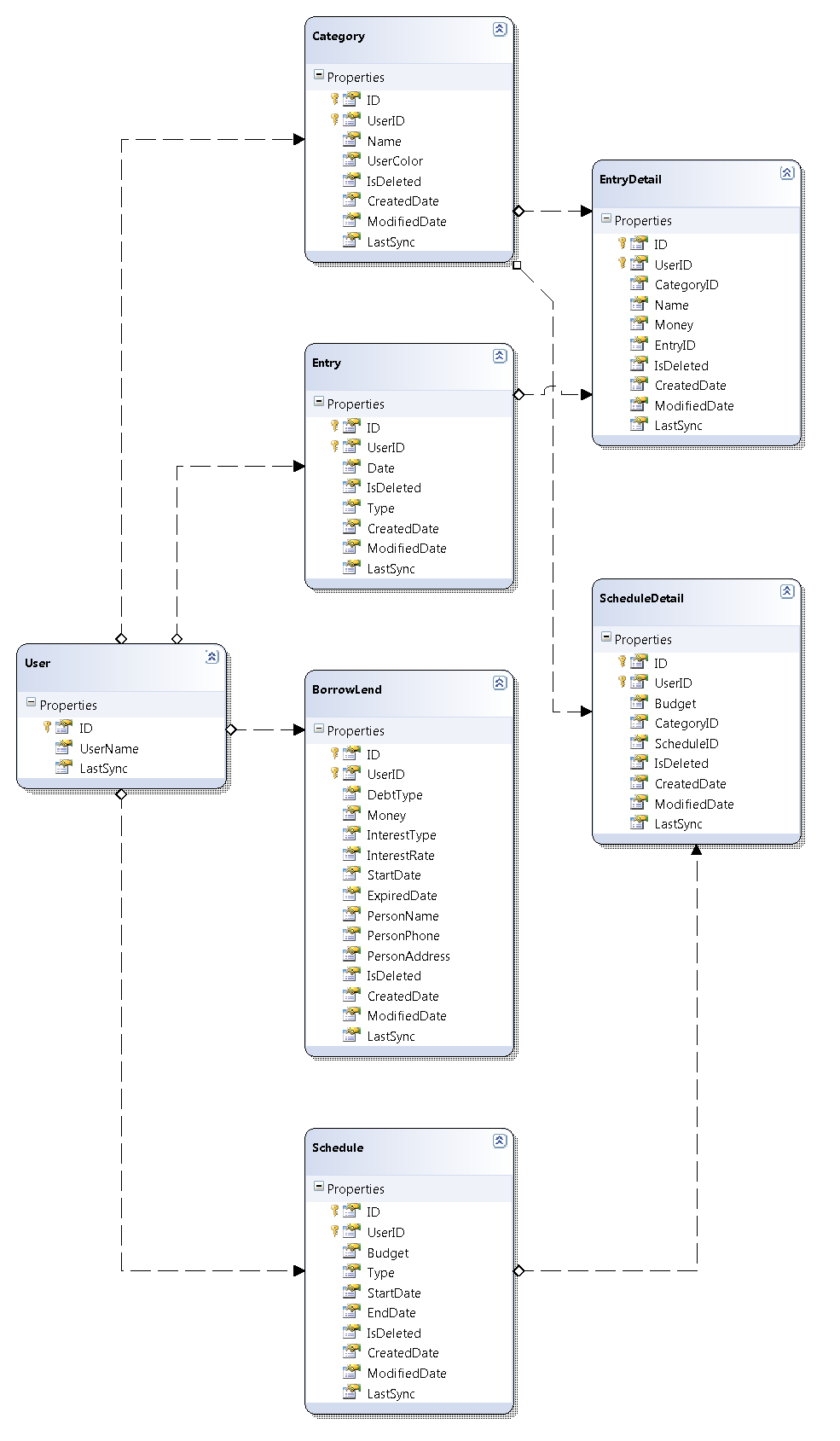
# 

# User Interface Design

# Database Design

## Database Design – Server Side

### **Database Diagram**



### **Tables**

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| User | Stores information of user |
| Category | Stores all category |
| Entry | Stores all entry |
| EntryDetail | Stores detail for each entry |
| Schedule | Stores all schedule |
| ScheduleDetail | Stores detail for each schedule |
| BorrowLend | Stores all borrowing and lending |

### **Table Description**

#### **User**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | INT |  | PK | User ID |
| UserName | VARCHAR(100) |  |  | Account username |
| LastSync | DATETIME |  |  | The latest date time which user synchronize |

#### **Category**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | BIGINT |  | PK | Category ID |
| UserID | INT |  | PK, FK | User ID |
| Name | NVARCHAR(100) |  |  | Category Name |
| UserColor | NVARCHAR(100) |  |  | Category Color |
| IsDeleted | INT |  |  | Mark a record is deleted or not |
| CreatedDate | DATETIME |  |  | The creation date time of a record |
| ModifiedDate | DATETIME |  |  | The updated data’s date time of a record |
| LastSync | DATETIME |  |  | The latest date time of record which user synchronized |

#### **Entry**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | BIGINT |  | PK | Entry ID |
| UserID | INT |  | PK, FK | User ID |
| Date | DATETIME |  |  | Entry Name |
| Type | NVARCHAR(100) |  |  | Entry Type (Income/ Expense) |
| IsDeleted | INT |  |  | Mark a record is deleted or not |
| CreatedDate | DATETIME |  |  | The creation date time of a record |
| ModifiedDate | DATETIME |  |  | The updated data’s date time of a record |
| LastSync | DATETIME |  |  | The latest date time of record which user synchronized |

#### **EntryDetail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | BIGINT |  | PK | Entry Detail ID |
| UserID | INT |  | PK, FK | User ID |
| CategoryID | BIGINT |  | FK | Category ID |
| EntryID | BIGINT |  | FK | Entry ID |
| Name | NVARCHAR(100) |  |  | Entry Detail Name |
| Money | FLOAT |  |  | The money of entry detail |
| IsDeleted | INT |  |  | Mark a record is deleted or not |
| CreatedDate | DATETIME |  |  | The creation date time of a record |
| ModifiedDate | DATETIME |  |  | The updated data’s date time of a record |
| LastSync | DATETIME |  |  | The latest date time of record which user synchronized |

#### **Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | BIGINT |  | PK | Schedule ID |
| UserID | INT |  | PK, FK | User ID |
| Budget | FLOAT |  |  | The total budget of schedule |
| Type | INT |  |  | The type of schedule (Month/Week) |
| StartDate | DATETIME |  |  | The start of schedule’s period |
| EndDate | DATETIME |  |  | The end of schedule’s period |
| IsDeleted | INT |  |  | Mark a record is deleted or not |
| CreatedDate | DATETIME |  |  | The creation date time of a record |
| ModifiedDate | DATETIME |  |  | The updated data’s date time of a record |
| LastSync | DATETIME |  |  | The latest date time of record which user synchronized |

#### **ScheduleDetail**

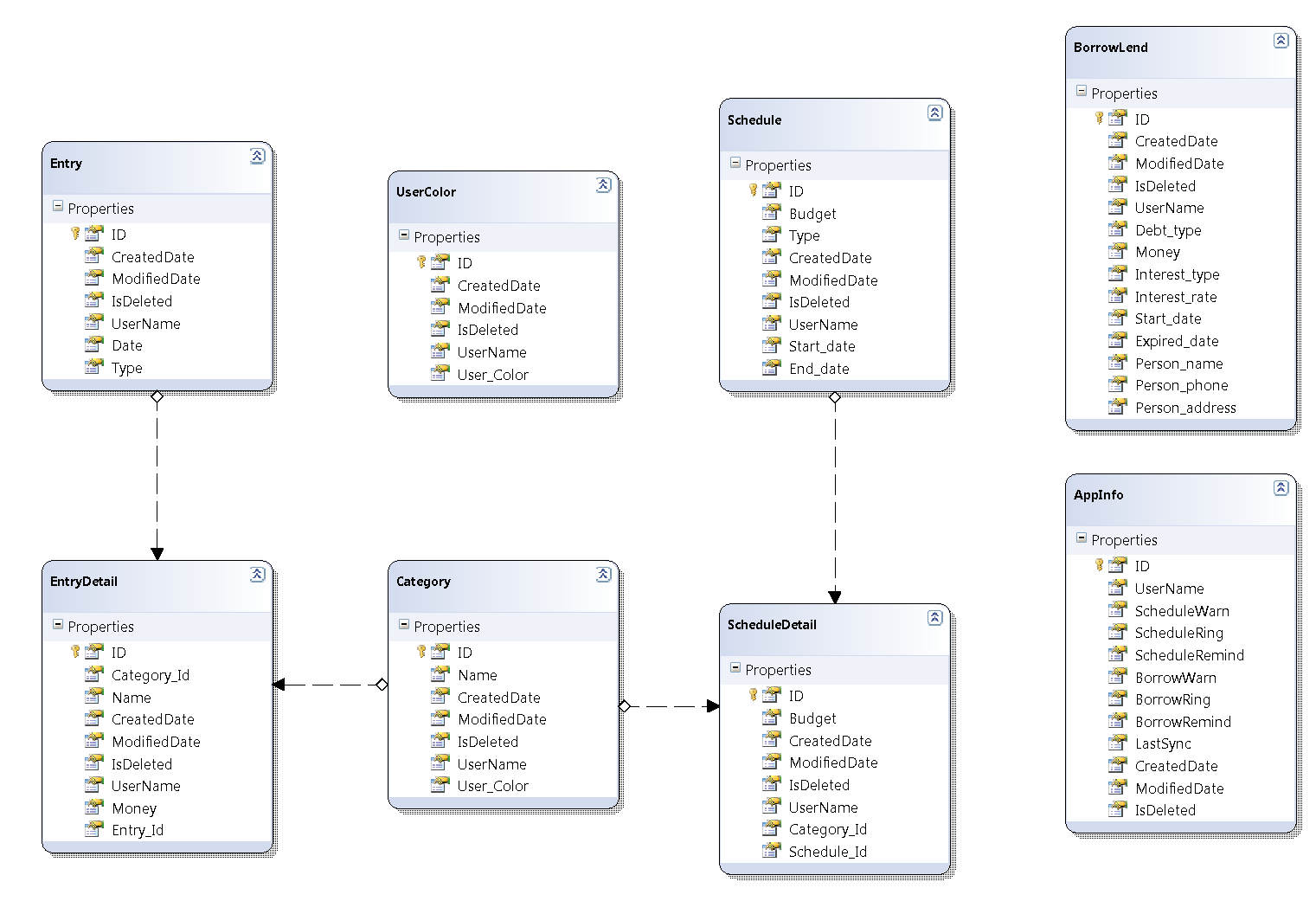
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | BIGINT |  | PK | Schedule Detail ID |
| UserID | INT |  | PK, FK | User ID |
| CategoryID | BIGINT |  | FK | Category ID |
| ScheduleID | BIGINT |  | FK | Schedule ID |
| Budget | FLOAT |  |  | The budget for each category |
| IsDeleted | INT |  |  | Mark a record is deleted or not |
| CreatedDate | DATETIME |  |  | The creation date time of a record |
| ModifiedDate | DATETIME |  |  | The updated data’s date time of a record |
| LastSync | DATETIME |  |  | The latest date time of record which user synchronized |

#### **BorrowLend**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | BIGINT |  | PK | Schedule Detail ID |
| UserID | INT |  | PK, FK | User ID |
| DebtType | NVARCHAR(100) |  |  | The type of debt (borrowing/ lending) |
| Money | FLOAT |  |  | The total money of debt |
| InterestType | NVARCHAR(100) |  |  | The type of interest (simple/ compound) |
| InterestRate | INT | X |  | The rate of interest |
| StartDate | DATETIME |  |  | Starting date time of debt period |
| ExpiredDate | DATETIME | X |  | Expired date time of debt period |
| PersonName | NVARCHAR(100) |  |  | The owner’s name of debt |
| PersonPhone | NVARCHAR(100) | X |  | The owner’s phone of debt |
| PersonAddress | NVARCHAR(100) | X |  | The owner’s address of debt |
| IsDeleted | INT |  |  | Mark a record is deleted or not |
| CreatedDate | DATETIME |  |  | The creation date time of a record |
| ModifiedDate | DATETIME |  |  | The updated data’s date time of a record |
| LastSync | DATETIME |  |  | The latest date time of record which user synchronized |

## Database Design – Client Side

### **Database Diagram**



### **Tables**

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| AppInfo | Stores information of application |
| Category | Stores all category |
| Entry | Stores all entry |
| EntryDetail | Stores detail for each entry |
| Schedule | Stores all schedule |
| ScheduleDetail | Stores detail for each schedule |
| BorrowLend | Stores all borrowing and lending |
| UserColor | Stores all color for category |

### **Table Description**

#### **AppInfo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| Id | LONG |  | PK | Application Information ID |
| UserName | TEXT |  |  | Account username |
| ScheduleWarn | INTEGER |  |  |  |
| ScheduleRing | TEXT |  |  |  |
| ScheduleRemind | LONG |  |  |  |
| BorrowWarn | INTEGER |  |  |  |
| BorrowRing | TEXT |  |  |  |
| BorrowRemind | LONG |  |  |  |
| LastSync | DATE |  |  | The latest date time of record which user synchronized |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |

#### **Category**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | Long |  | PK | Category ID |
| Name | TEXT |  |  | Category Name |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| User\_Color | TEXT |  |  | Category Color |

#### **Entry**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| Id | LONG |  | PK | Entry ID |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| Date | DATE |  |  | Entry Name |
| Type | INTEGER |  |  | Entry Type(1: Income/ 0: Expense) |

#### **EntryDetail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| Id | LONG |  | PK | Entry Detail ID |
| Category\_Id | LONG |  | FK | Category ID |
| Name | TEXT |  |  | Entry Detail Name |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| Entry\_Id | LONG |  | FK | Entry ID |
| Money | LONG |  |  | The money of entry detail |

#### **Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| Id | LONG |  | PK | Schedule ID |
| Budget | LONG |  |  | The total budget of schedule |
| Type | INTEGER |  |  | The type of schedule (1: Month/0: Week) |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| Start\_date | DATE |  |  | The start of schedule’s period |
| End\_date | DATE |  |  | The end of schedule’s period |

#### **ScheduleDetail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| Id | LONG |  | PK | Schedule Detail ID |
| Budget | LONG |  |  | The budget for each category |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| Category\_Id | LONG |  | FK | Category ID |
| Schedule\_Id | LONG |  | FK | Schedule ID |

#### **BorrowLend**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| Id | LONG |  | PK | Schedule Detail ID |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| Debt\_type | TEXT |  |  | The type of debt (borrowing/ lending) |
| Money | LONG |  |  | The total money of debt |
| Interest\_type | TEXT |  |  | The type of interest (simple/ compound) |
| Interest\_rate | INTEGER | X |  | The rate of interest |
| Start\_date | DATE |  |  | Starting date time of debt period |
| Expired\_date | DATE | X |  | Expired date time of debt period |
| Person\_name | TEXT |  |  | The owner’s name of debt |
| Person\_phone | TEXT | X |  | The owner’s phone of debt |
| Person\_address | TEXT | X |  | The owner’s address of debt |

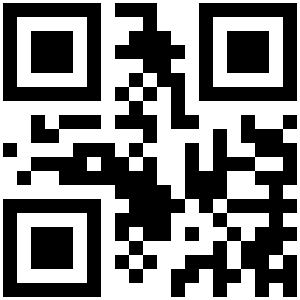
#### **UserColor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Allow Nulls** | **P/F Key** | **Description** |
| ID | Long |  | PK | Color ID |
| CreatedDate | DATE |  |  | The creation date time of a record |
| ModifiedDate | DATE |  |  | The updated data’s date time of a record |
| IsDeleted | INTEGER |  |  | Mark a record is deleted or not |
| UserName | TEXT |  |  | Account username |
| User\_Color | TEXT |  |  | Color code |

# Technology Solution

## QR Code

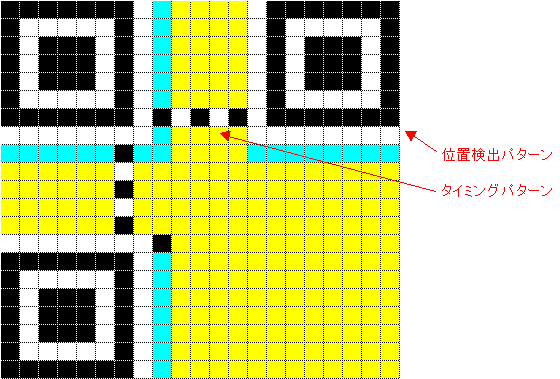
### **What is QR Code?**



They come to us from Japan where they are very common. QR is short for Quick Response (they can be read quickly by a cell phone). They are used to take a piece of information from a transitory media and put it in to your cell phone. You may soon see QR Codes in a magazine advert, on a billboard, a web page or even on someone’s t-shirt. Once it is in your cell phone, it may give you details about that business (allowing users to search for nearby locations), or details about the person wearing the t-shirt, show you a URL which you can click to see a trailer for a movie, or it may give you a coupon which you can use in a local outlet.

The reason why they are more useful than a standard barcode is that they can store (and digitally present) much more data, including url links, geo coordinates, and text. The other key feature of QR Codes is that instead of requiring a chunky hand-held scanner to scan them, many modern cell phones can scan them.

### **Structure of QR Code symbol**

The below figure is a structure of QR Code model 2 version 1. In the below figure, white or black parts are fixed in specification. They are "finder pattern" and "timing pattern".  
"Finder pattern" is used to help detection a position of QR Code in decoder application.  
"Timing pattern" is used to help determine a symbol's coordinate in decoder application.  
  
In yellow color parts, encoded data (including error correct code) are stored.  
And in cyan color parts, information of error correcting level and mask pattern (described later) are stored. This is called format information  
  
  
  
■encoded data (including error correct code)  
■format information

### **Using ZXing to decode QR Code**

With this capstone project, the group decided using Zxing library to read QR Code. The components in QR Code Reader include:

1. Image Conversation
2. QR Detector
3. QR Decoder

#### **Image Conversation**

ZXing will use camera of the device to capture the QR Code. The image will be written on the device storage. Another component will process this image. ZXing will need permission to use camera, write on device storage.

#### **QR Detector**

ZXing proceeds by horizontally column scanning the raw image pixels (or vertically row scanning) to locate the 3 position patterns identified by the 1-1-3-1-1 ratio. After a potential position pattern is located, it is then cross-checked in the “other” direction for the same ratio and position pattern size. Also determines the centre of the position pattern.

ZXing will detect that QR Code maps has right perspective. Simply maps points on a known square to points on the distorted, rotated or skewed square. If not, ZXing will the following algorithm to correct perspective.

for (long i = 0; i < dimension; ++i) // y-axis direction

{

for (long j = 0; j < dimension; ++j) // x-axis direction

{

// The QR code is read as if it were an un-rotated and un-skewed image

POINT pt;

pt.x = (j \* module\_size) + half\_module\_size;

pt.y = (i \* module\_size) + half\_module\_size;

POINT p = transform.TransformPoint(pt); // Does the perspective correcting

qr\_store.set(pt, image.IS\_BLACK(p));

}

}

QR Code will be rotated corrections.

##### 9.1.3.3. QR Decoder

Base on QR Code encode algorithm, ZXing will be decode QR Code. And return the result to software.

## NFC

## Synchronize