Iteration II

Trello Link: https://trello.com/b/GXCvTyOf/4-amigos

Web: https://www.your4amigos.com/

Git: https://github.com/YzkGao/CSI-3471-team-project-4-amigos-

Team: 4 amigos

Members:

Yangzekun Gao, Jingke Shi, Zhengyan Hu, Maiqi Hou

Content

Project Vision

We are planning to make an accounting application. It provides functions for users to check their daily expense or income records. A record is composed by date, event, type, amount, and additional comments. The date indicates the time of the record, event indicates the scene of the record, type indicates that is the record a expense or income. The amount shows that how much money users expended or earned in the record, and we allow users to add an additional comment.

Functions:

(1) Adding records: It gives user a way to add new records.

(2) Viewing records: It allows users to view their records in a fixed time range. Users are able to select the time range they want. The application will generate a table to show the records. In addition, we will show some statistical data, such as total balance, total income, and total expenses.

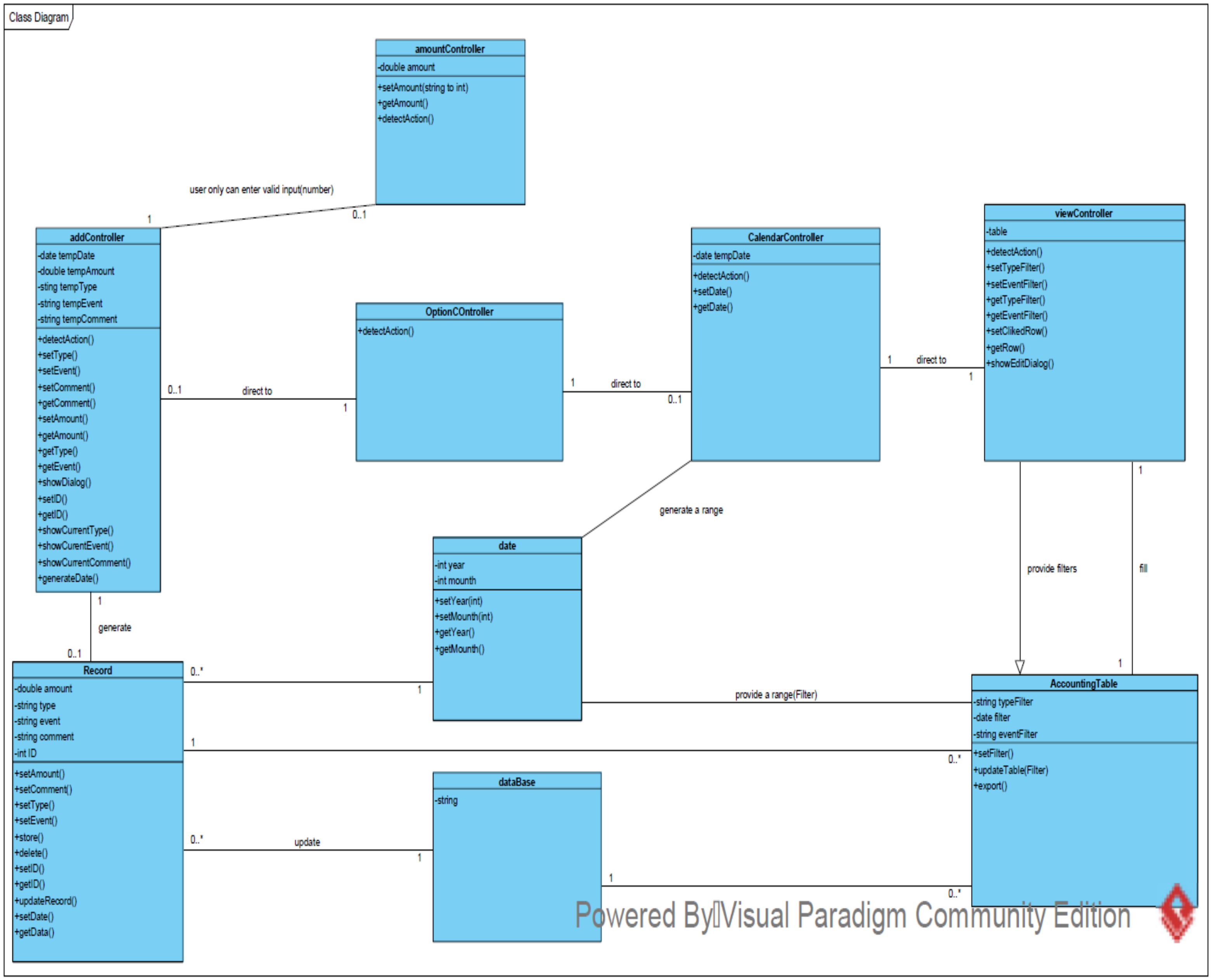
(3) Viewing by using filters: We allow users to minimize the table by adding some filters, such as which event or type they only want to view. Also, the statistical data will update.

(4) Export: We allow users to export the table based on applied filters in a readable way to a certain file.

(5) Edit: Users can edit records while they are viewing.

(6) Delete: Users can delete records while they are viewing.

Class Diagram



GRASP

1. Creator:

AddCountroller is a creator that will create a record and store the record to the database.

2. Information Expert:

We don’t have this patten in order to avoid the problem of cohesion and coupling.
Example:
If we are shopping, one product only appears once in the shopping cart. So when we add product to the shopping cart we need to know if this produce is in the cart. And this responsibility should go to produce, because the produce has their unique id number.

3. Low Coupling:

This pattern is prefer to assign responsibilities and then reduce the impact change in depended upon elements on dependent elements. That means we should minimize the connection between classes.
In our class diagram, we use Low Coupling between amountController class and addController class. Because the amountController is to input value only so the only related class is addController.

4.Controller:

We have the controllers:

amountController:

show a page like a calculator to read the input.

addController:

show a page like a form to read the imformation of a record.

OptionController:

There is 2 buttom in this page. One is add, another is view

CalendarCountroller:

This page performs a calendar to set a filter.

viewController:

This page shows the table of AccountingTable.

5. High cohesion:

This pattern is prefer to place the functionally closely related responsibilities in a class and let those responsibilities work together to accomplish limited functionality. High cohesion tries its best to guarantees that programs occur within each section will not affect other sections. Record class uses this pattern in our class diagram, for example, Record class has both of the setDate

and getDate, so if there are problems occur in date related functionality, because of the high cohesion, it will not influence other responsibilities.

6. Indirection

This pattern is prefer to assign responsibilities to a mediation object, isolating the object from other artifacts or services so that they do not have direct coupling. So the indirection pattern’s benefit always with low coupling. For example, The OptionController class in our project is the “mediation object” between addController class and CalendarController class, which helps the isolating of those two classes.

7. Polymorphism:
We don’t have Polymorphism on our project.
Example: If we want to calculate the tax of a produce, We will have a method call calculateTax(). However, different country has different tax. So in the calculateTax method, we need to use many if-else to calculate tax for different country. So we can use an interface with a calculateTax() and many country can implements from this interface to have their own calculateTax() method.

8. Protected Variations:

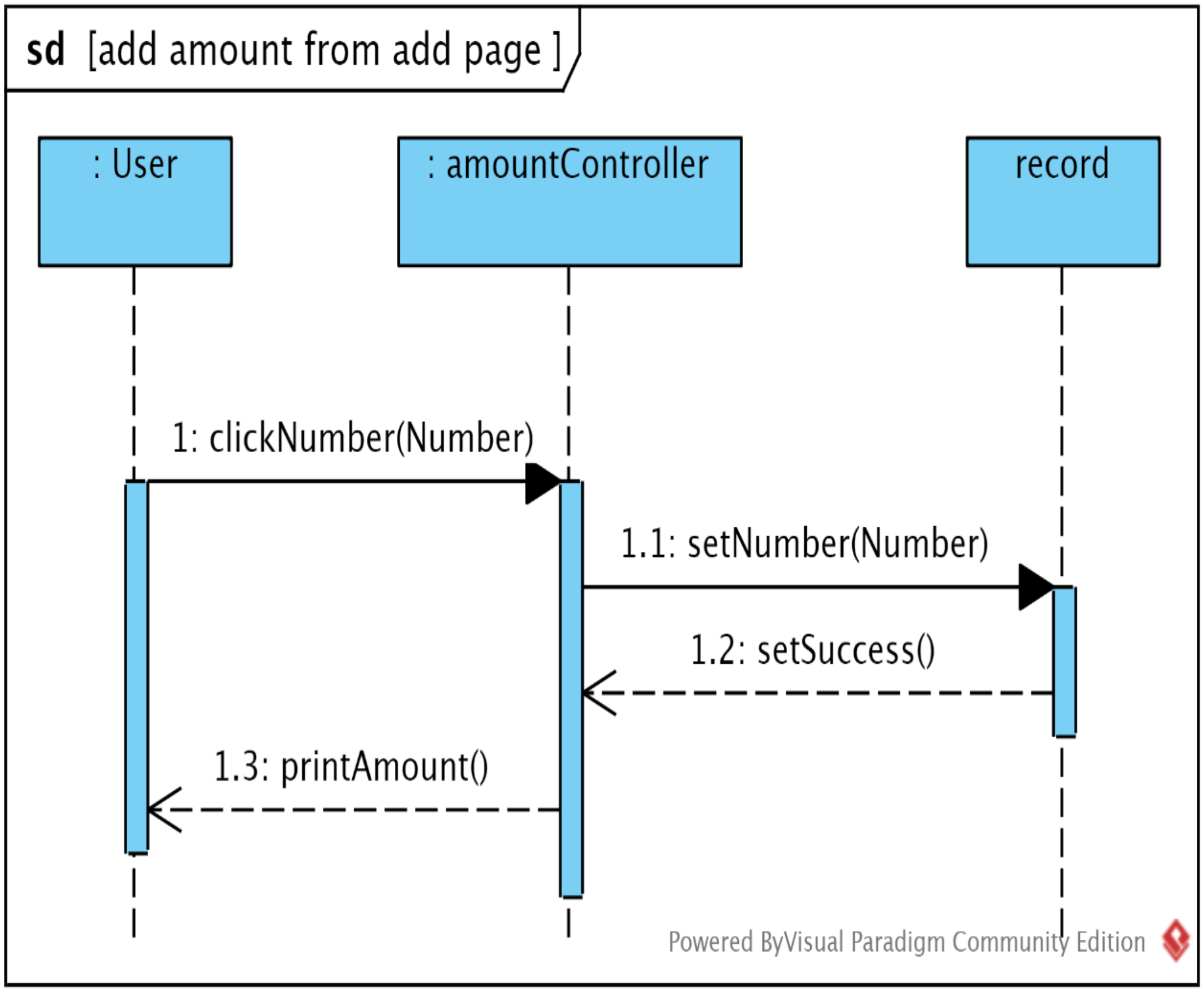
This pattern is prefer to provides flexibility and protection from variations. Pre-identify unstable change points, encapsulate with a unified interface, and if the future changes, new functionality can be extended through the interface without having to modify the old implementation. For example, providing a well defined interface so that the there will be no affect on other units.

9. Pure Fabrication:

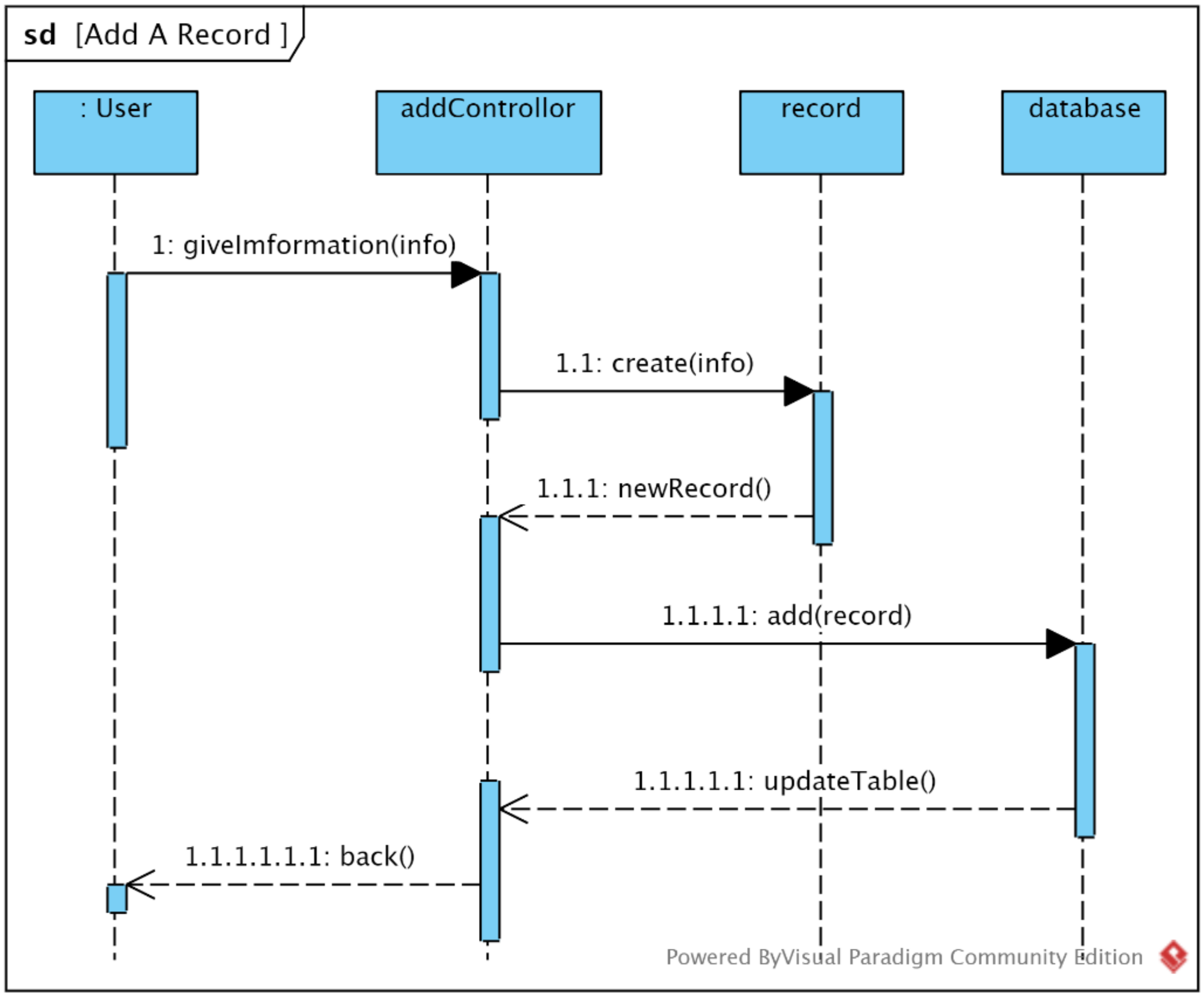
Example:
The system operates on the database. If we want to persist some objects in the system. If we use information expert pattern, it will assign this responsibility to each class. However, it will violent the high cohesion and low coupling. So we will make a new class such as DAO. And assign this responsibility to the DAO class.

Sequence Diagram

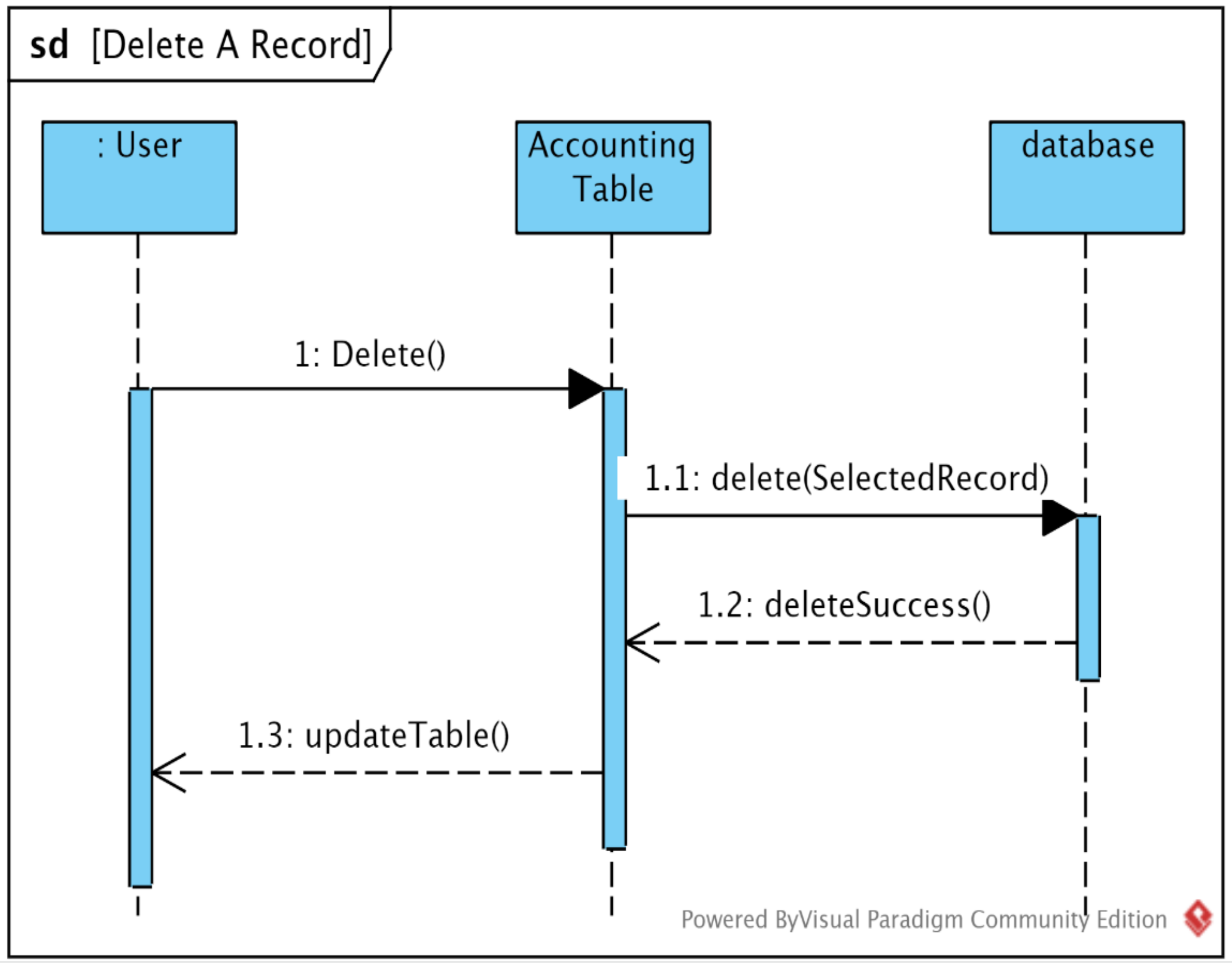
Add Amount-- SD



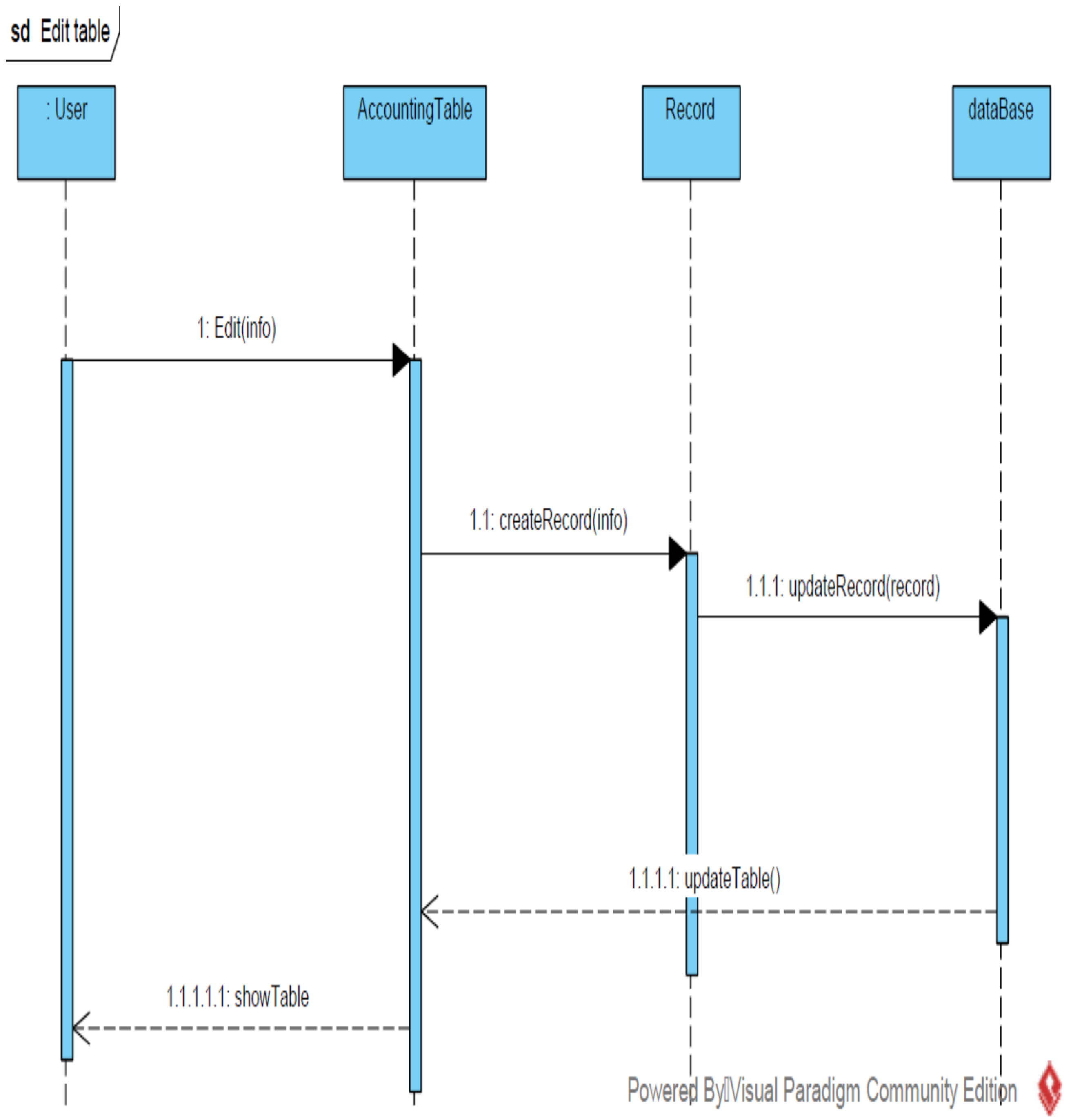
AddRecord – SD



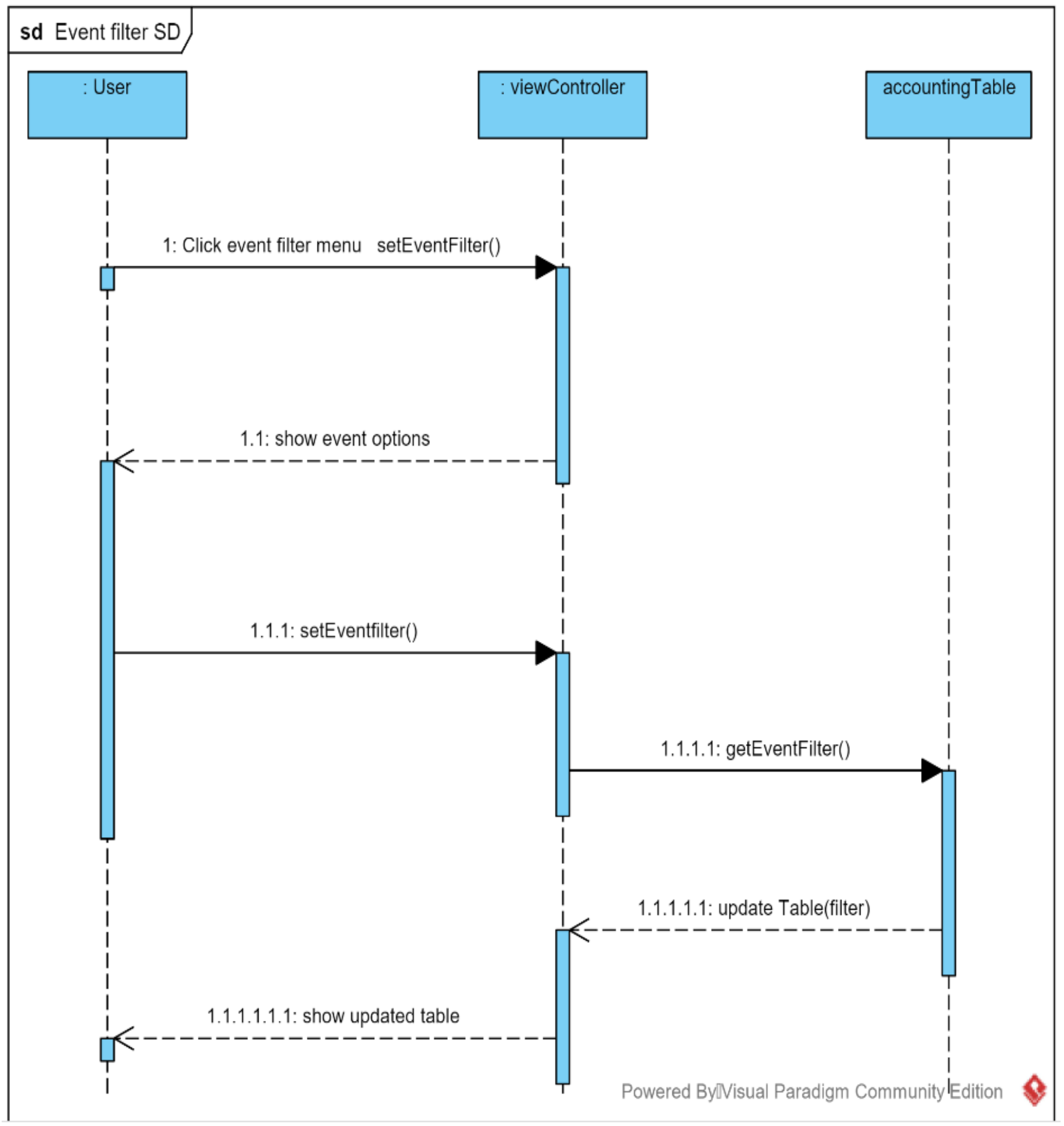
Delete Record -- SD



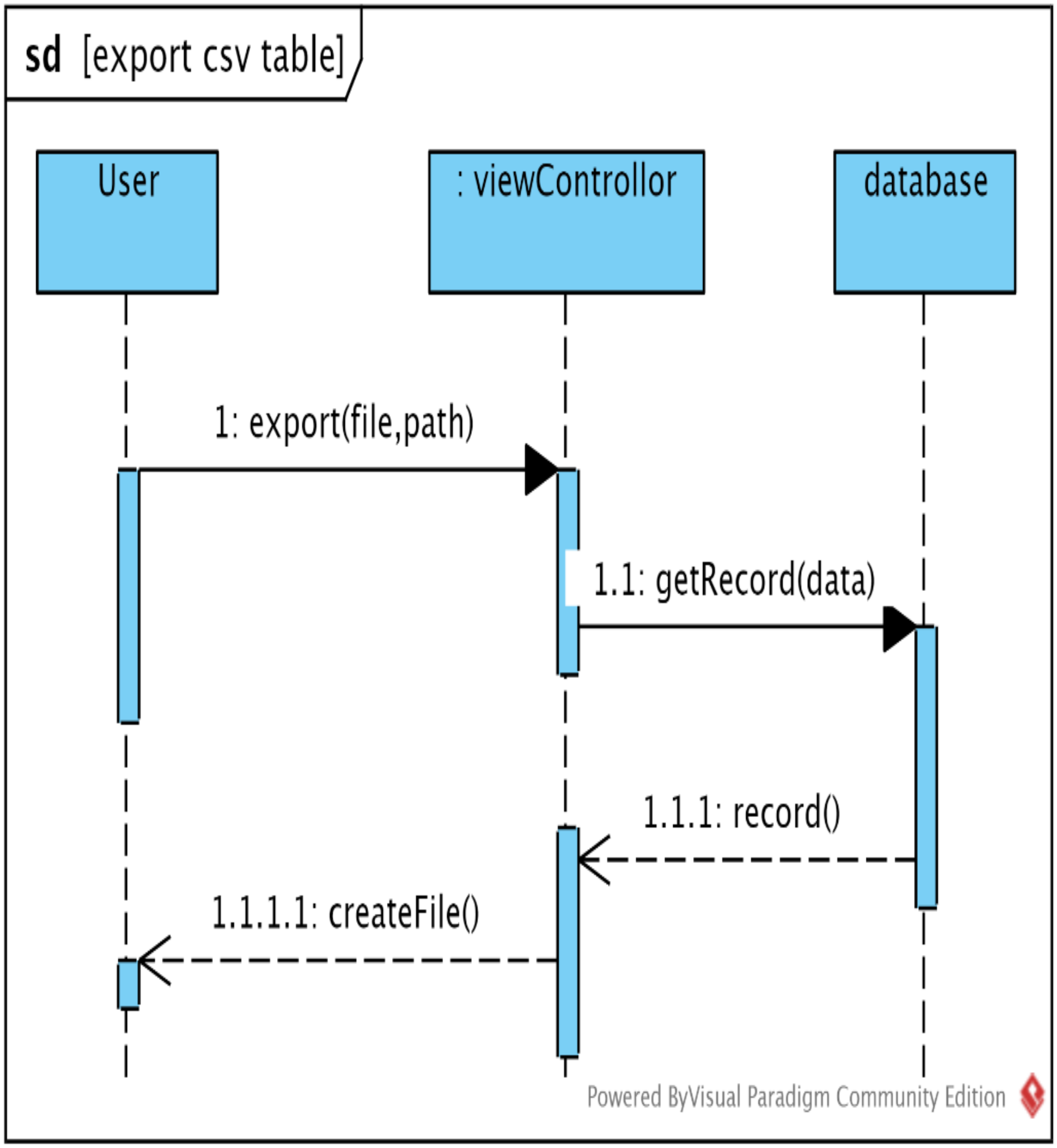
Edit table -- SD



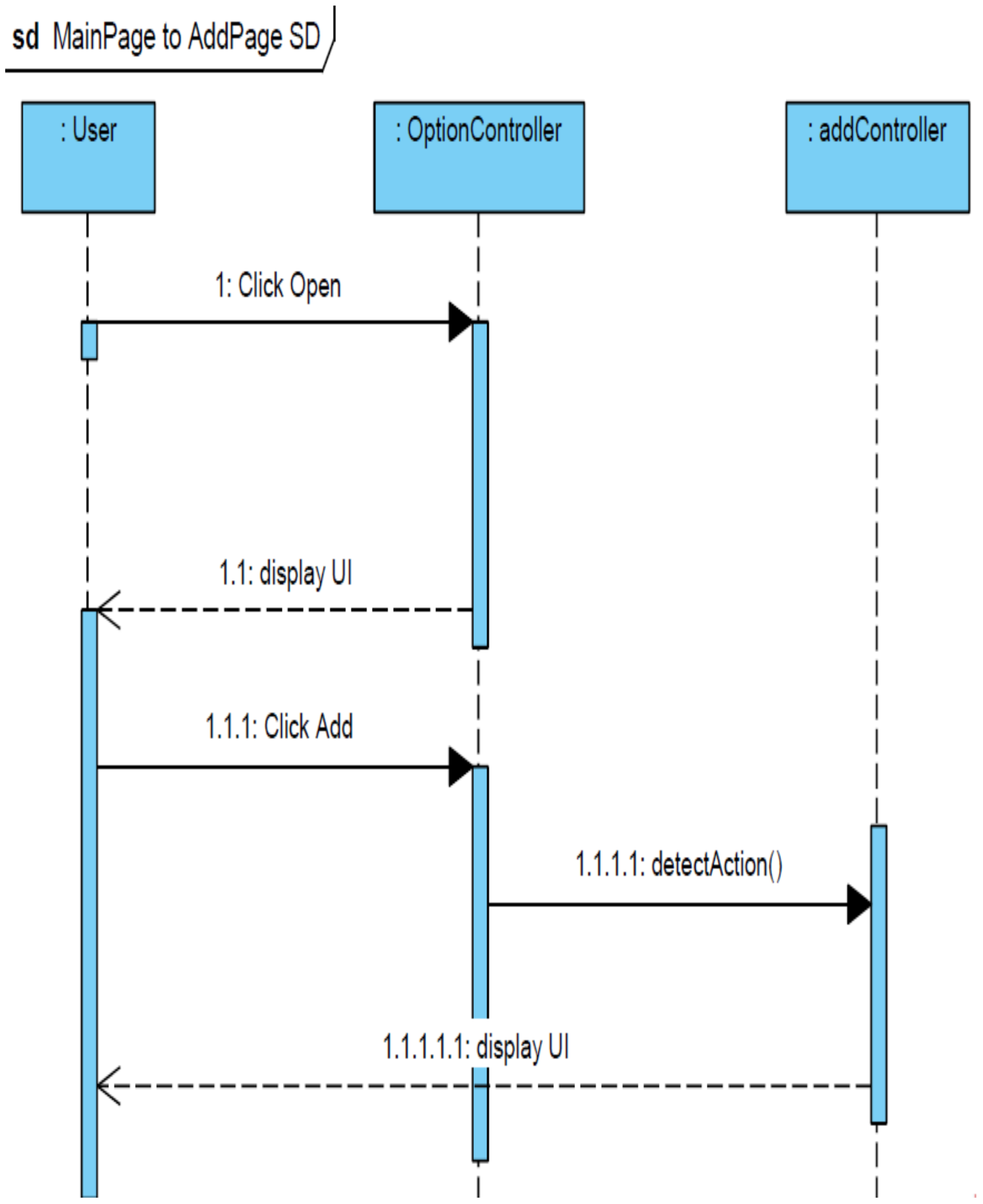
Event filter – SD


sd Event filter SD
: User
: viewController
accountingTable
1: Click event filter menu setEventFilter()
1.1: show event options
1.1.1: setEventfilter()
1.1.1.1: getEventFilter()
1.1.1.1.1: update Table(filter)
1.1.1.1.1.1: show updated table
Powered By Visual Paradigm Community Edition

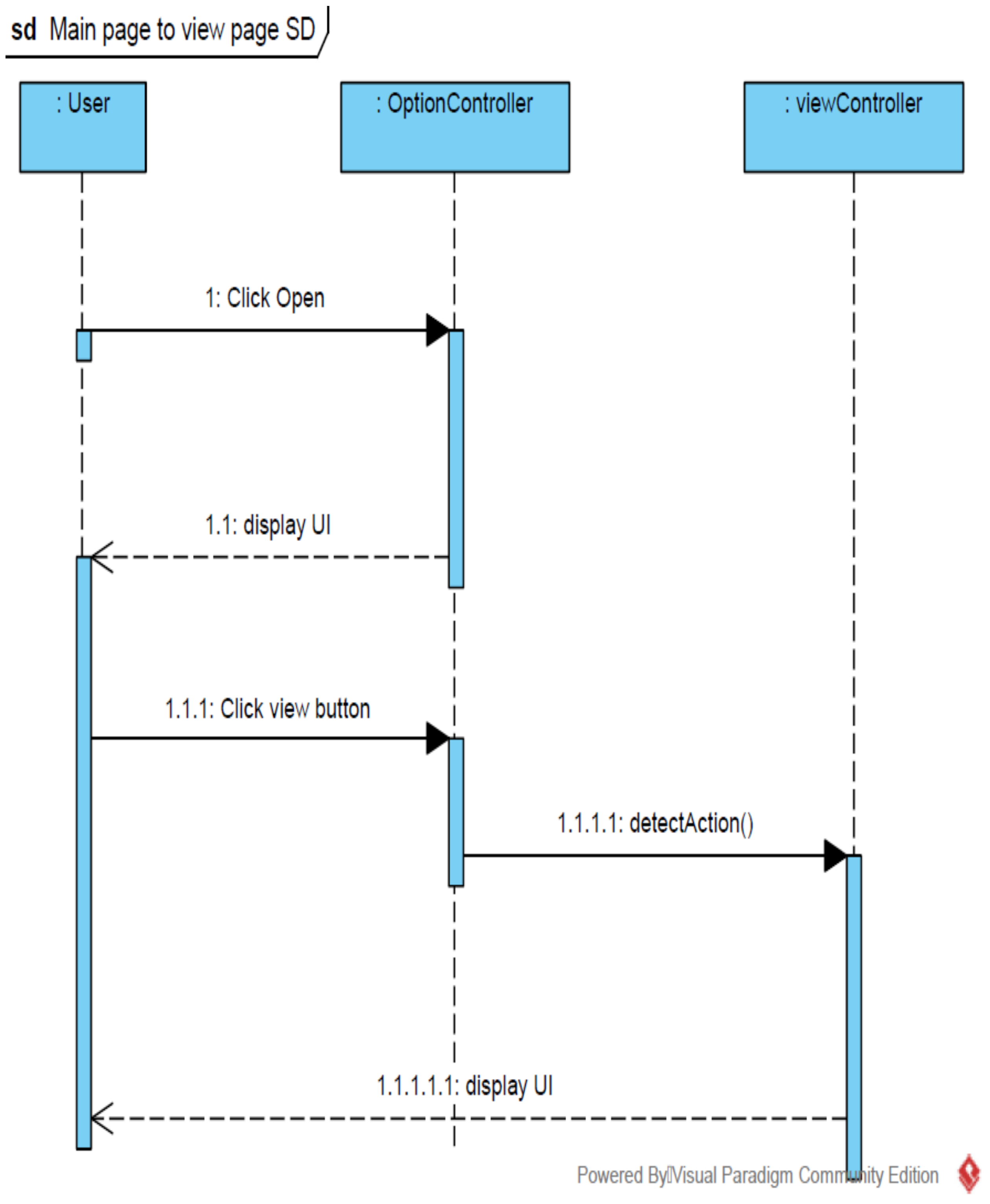
ExportTable – SD



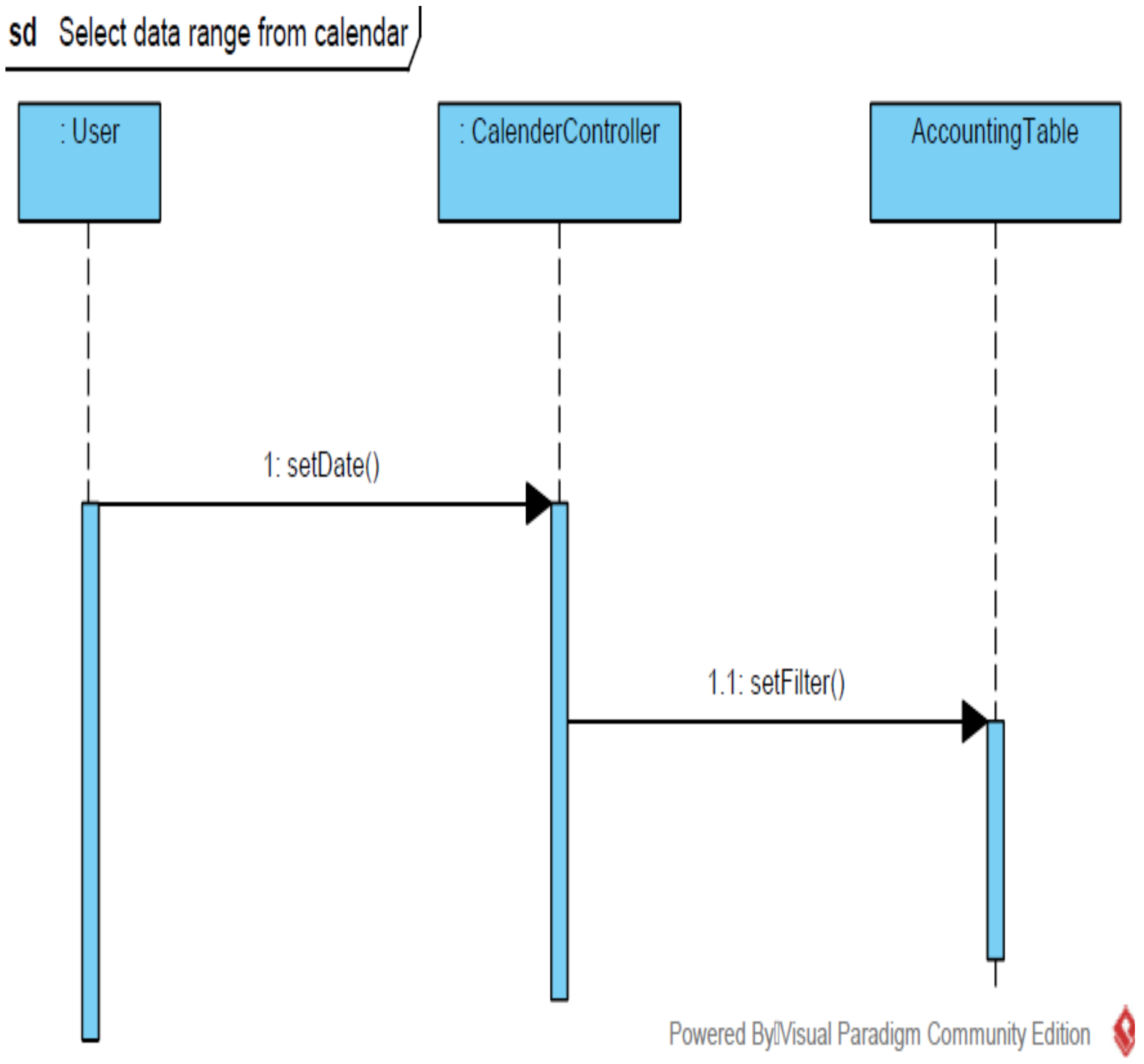
Main page to add page – SD



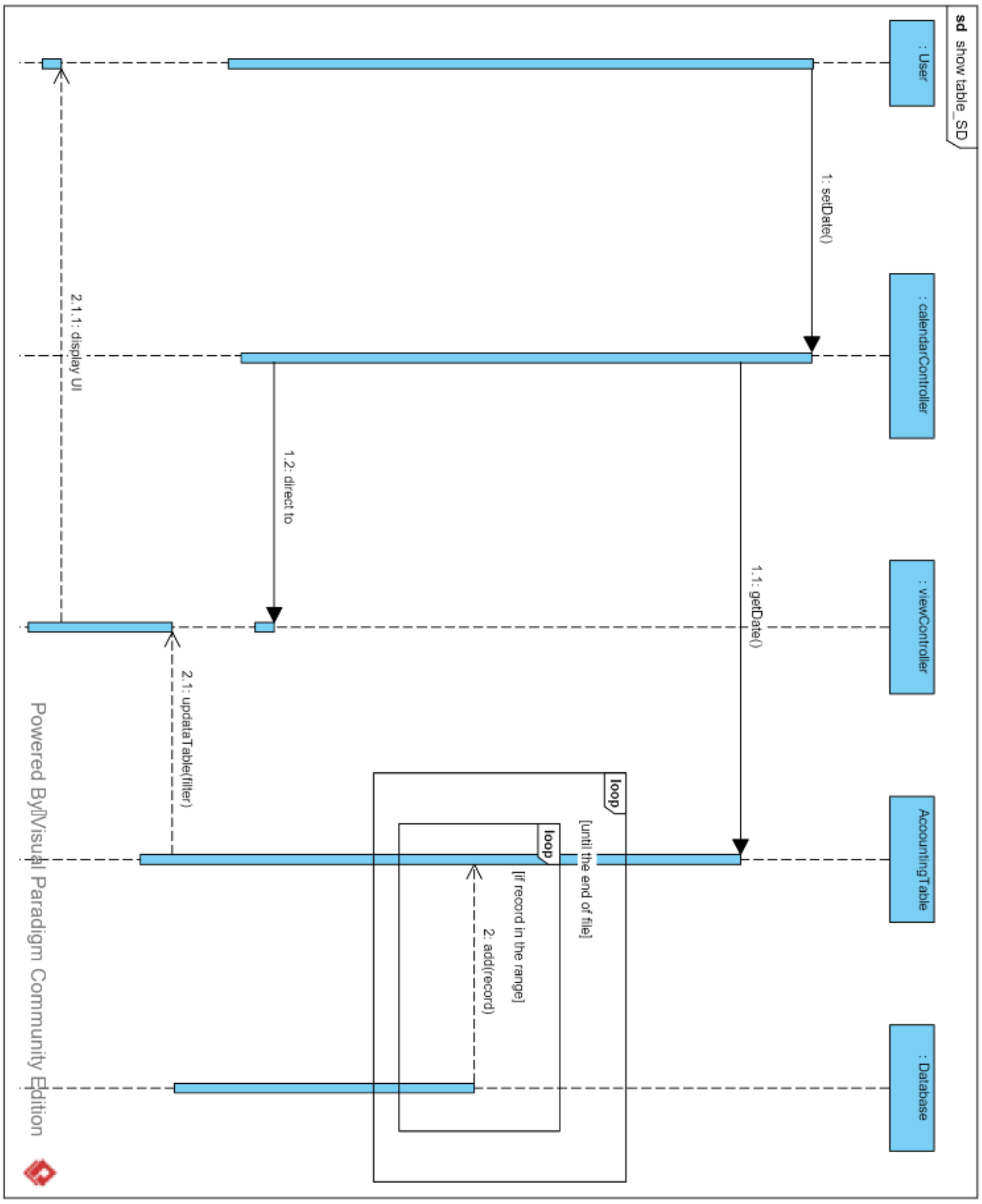
Main page to view page – SD



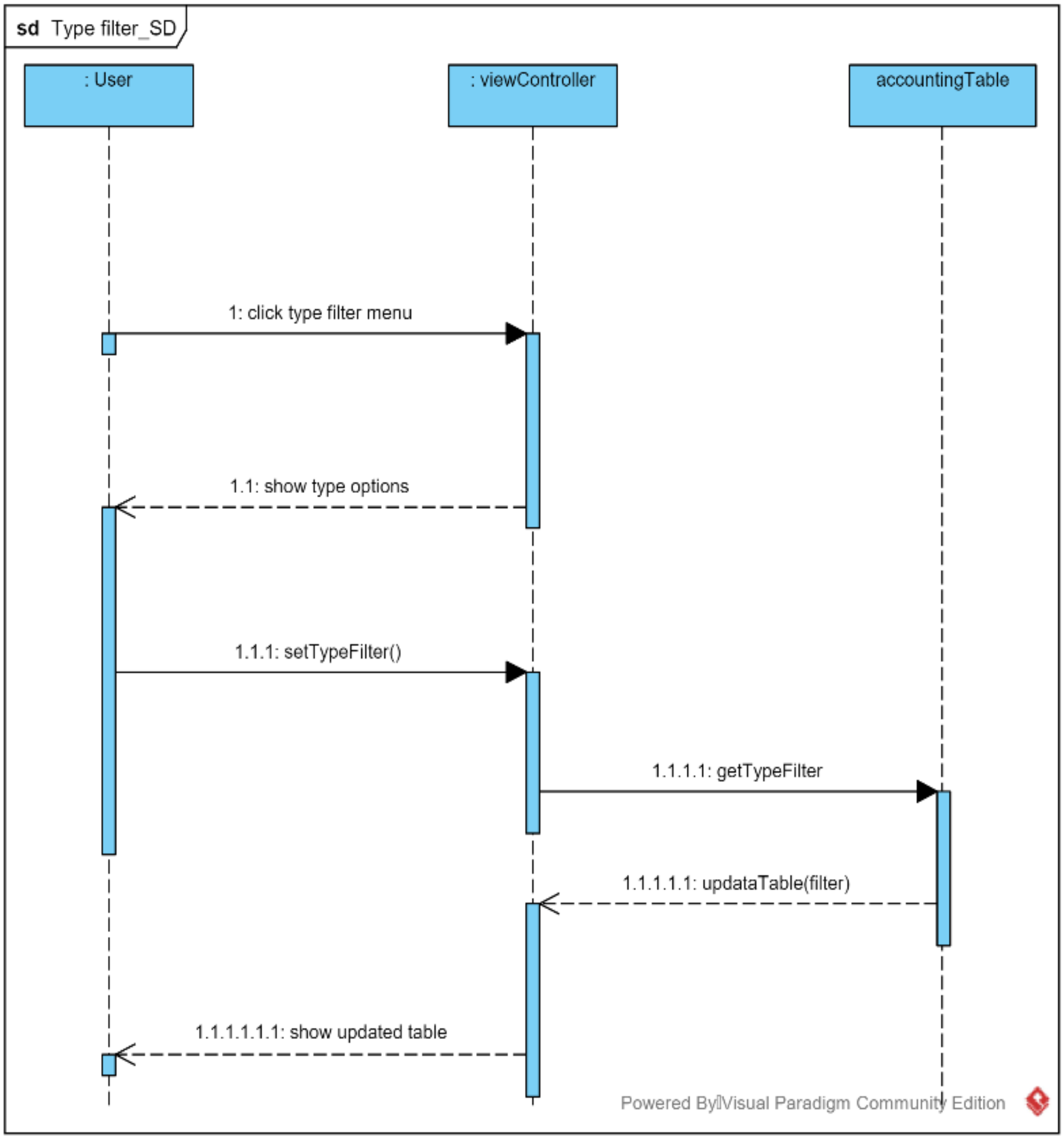
Select data range from calendar – SD


sd Select data range from calendar
: User
: CalenderController
AccountingTable
1: setDate()
1.1: setFilter()
Powered By Visual Paradigm Community Edition

Show table – SD



Typefilter – SD


sd Type filter_SD
: User
: viewController
accountingTable
1: click type filter menu
1.1: show type options
1.1.1: setTypeFilter()
1.1.1.1: getTypeFilter
1.1.1.1.1: updataTable(filter)
1.1.1.1.1.1: show updated table
Powered By Visual Paradigm Community Edition

Test Plan

Testing plan

1. OptionController

(1) detect user input

2.AddController

(1) Get record type

(2) Get record event

(3) Edit amount dialog work properly

(4) Get amount

(5) Get comment

(6) Store record to database

(7) Set record type

(8) Set record event

(9) Set amount

(10) Set comment

(11) generate ID

3. calendarController

(6) Set date

(7) Get date

4. viewController

(1) Set event filter

(2) Get event filter

(3) Set type filter

(4) Get type filter

(5) Set row

(6) Get row

5.accountingTable

(1) Test if it can export data

(2) Test Updating table

6. amountController

(1) Test conversion between string and integer

7. record

(1) Set record type

(2) Set record event

(3) Set amount

(4) Set comment

(5) Get type

(6) Get event

(7) Get amount

(8) Get comment

8. UI

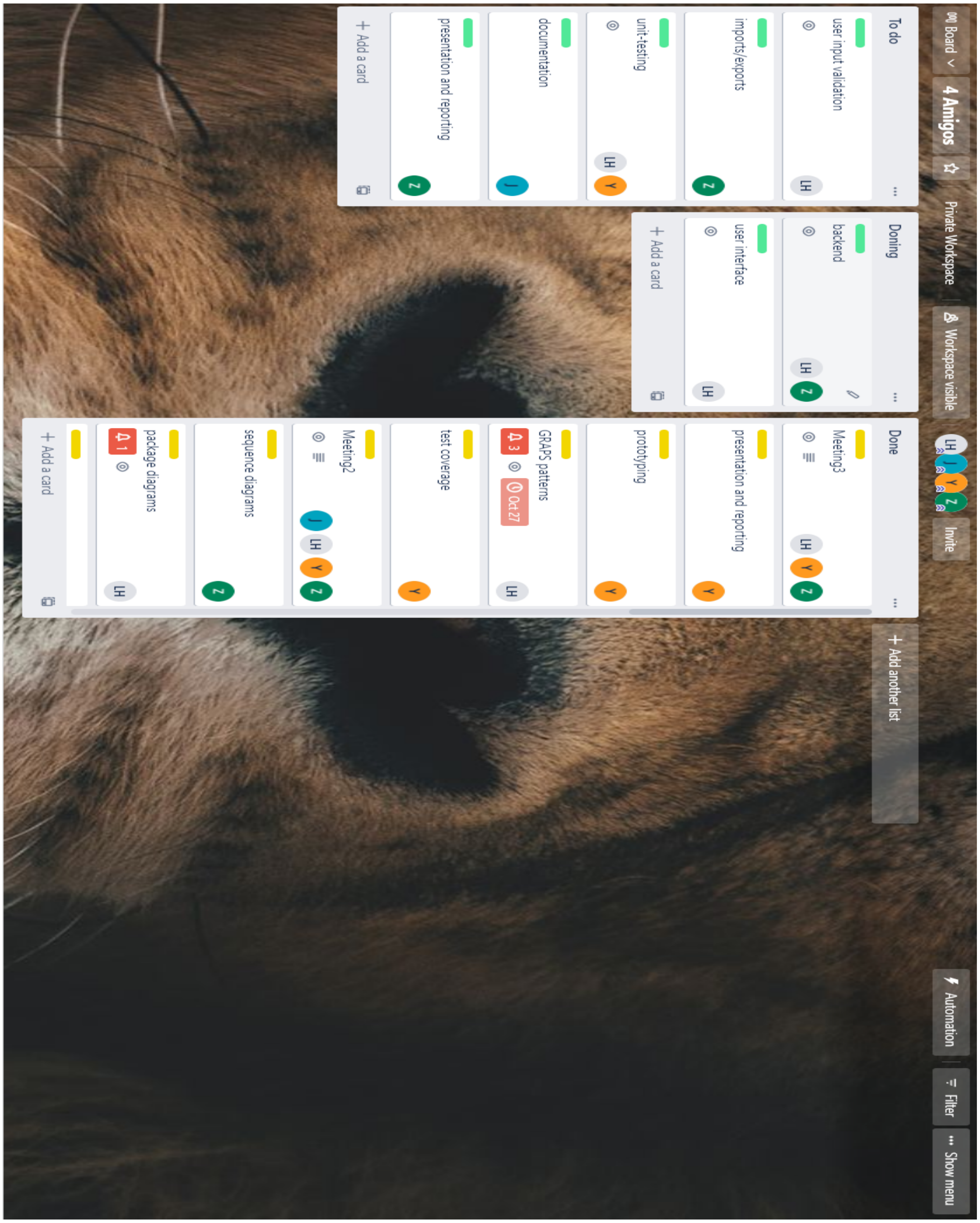
(1) Every button work properly

(2) Can work on different computer system

(3) Link to other frame properly

(4) Close and display properly

Trello



Board 4 Amigos Private Workspace Workspace visible LH J Y Z Invite Automation Filter Show menu

To do

- user input validation — LH
- imports/exports — Z
- unit-testing — LH Y
- documentation — J
- presentation and reporting — Z

+ Add a card

Doing

- backend — LH Z
- user interface — LH

+ Add a card

Done

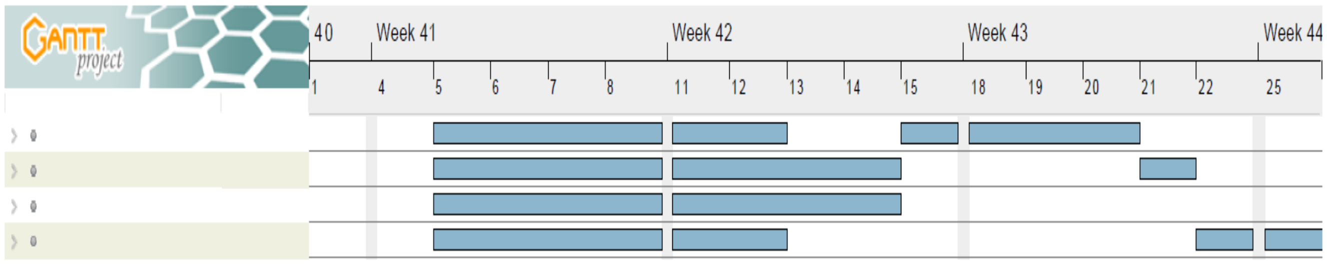
- Meeting3 — LH Y Z
- presentation and reporting — Y
- prototyping — Y
- GRAPS patterns — 3 Oct 27 — LH
- test coverage — Y
- Meeting2 — J LH Y Z
- sequence diagrams — Z
- package diagrams — 1 — LH

+ Add a card

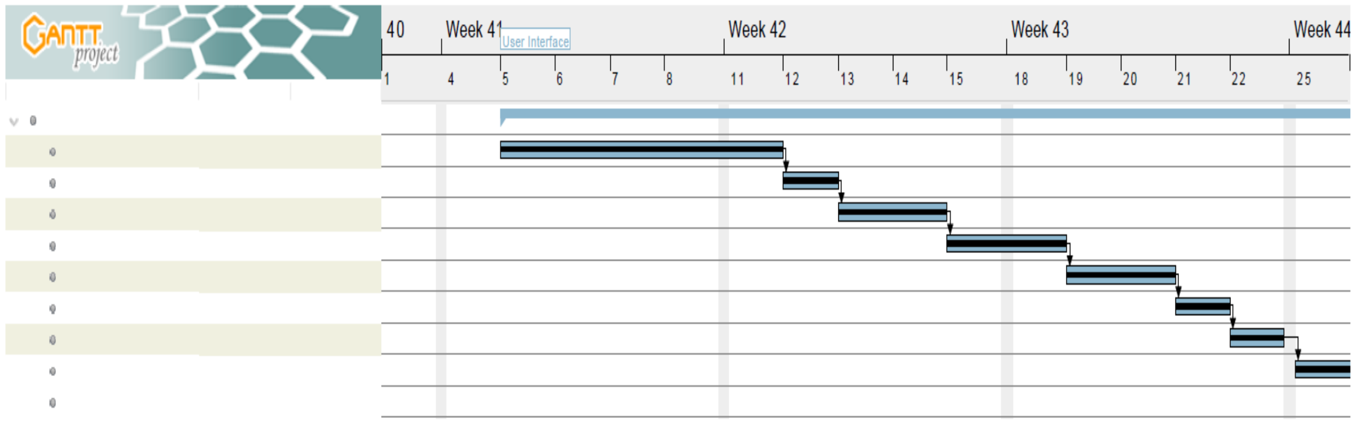
+ Add another list

Gantt

Resources Chart



Gantt Chart



Tasks

Name	Begin date	End date
Iteration 2	2021/10/5	2021/10/27
User Interface	2021/10/5	2021/10/11
Revised iteration 1	2021/10/12	2021/10/12
design model	2021/10/13	2021/10/14
sequence diagrams	2021/10/15	2021/10/18
package diagrams	2021/10/19	2021/10/20
GRAPS patterns	2021/10/21	2021/10/21
test coverage	2021/10/22	2021/10/22
prototyping	2021/10/25	2021/10/26
presentation and reporting	2021/10/27	2021/10/27

Team Time Record

Team 4 amigos Time Record

Meetings:

- Sunday, October 24, 2021 (5 PM – 7 PM)
- Wednesday, October 27, 2021 (1:25 PM – 6 PM)
- Friday, October 29, 2021(2:30 PM – 7Pm)

Individual Hours:

- Yangzekun Gao: 10 + 9.25 + 5 = 24.25 hrs.
- Maiqi Hou: 9 + 8.75 + 5= 23.75hrs.
- Jingke Shi: 9+ 8.0 = 17 hrs.
- Zhengyan Hu: 9 + 7.25 + 5= 21.25 hrs.

Suggested point redistribution

Yangzekun Gao:

-> Initial DEMO of User Interface using Maven

-> Diagram of selecting which month, year you want to view

-> Diagram of editing from view page

-> Test coverage plan

-> revised iteration 1

Zhengyan Hu

-> Diagram of delete records

-> Diagram of add record from add page

-> GRASP

-> suggested point redistribution

-> timecards report with hours per member

-> revised iteration 1

Maiqi Hou

-> Diagram of main page selection

-> Diagram of view the table by given filter

-> Linked issue tracking system and git

-> the issue tracking records

-> revised iteration 1

Jingke Shi

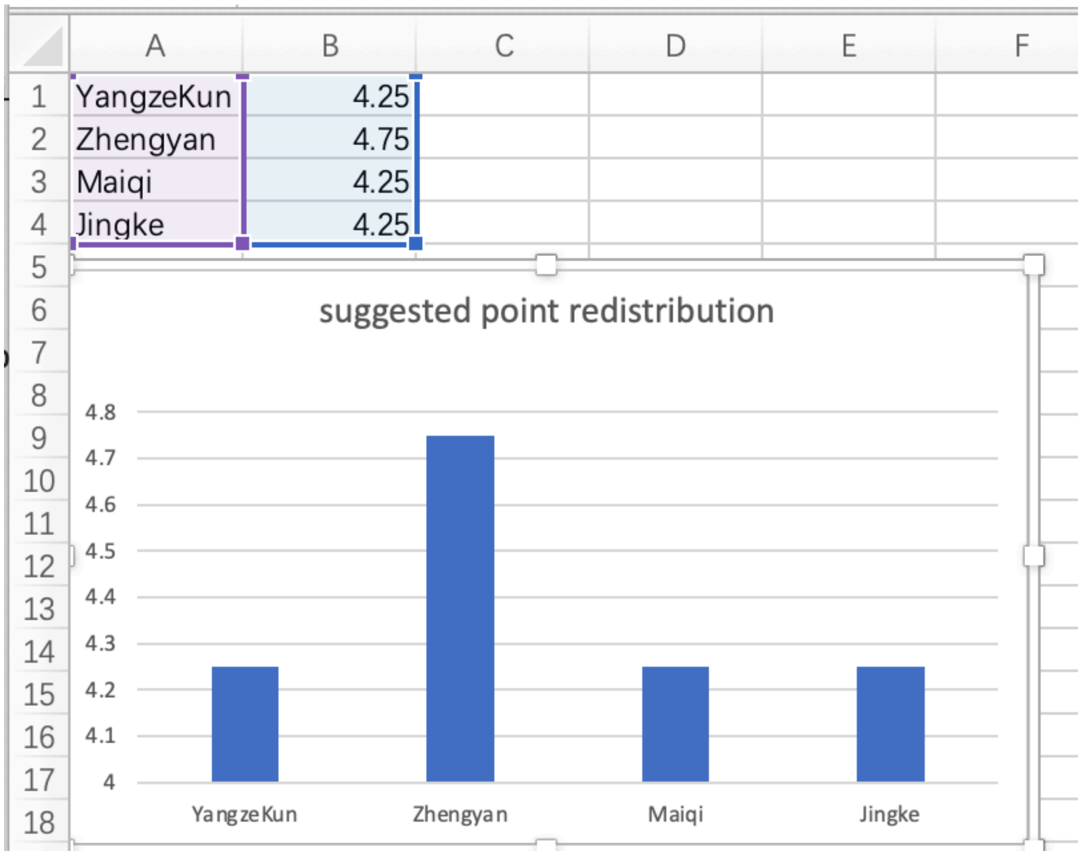
-> Diagram of view the table from the given range

-> Diagram of export table from given range

-> GRASP

-> Updated project plan(Gantt)

-> revised iteration 1



	A	B
1	YangzeKun	4.25
2	Zhengyan	4.75
3	Maiqi	4.25
4	Jingke	4.25

Issue Tracking

Issue	Found	Fixing	Done	Method
For the definition of classes, attribute, operation are not uniform			🟩	Meeting (Uniformed after discussion)
The operation process and design are not uniform (The button position is inconsistent, the frame size is inconsistent,and different ideas of function process and orders)			🟩	Meeting (Uniformed after discussion)
How to display the generate calendar		🟨		
How to save,generate and read the database	🟥			