



VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Tan Wei

ONLINE INVESTMENT AND LOAN APPLICATION

Department of Technology and Communication

2013

ABSTRACT

Author	Tan Wei
Title	Online Investment and Loan Application
Year	2013
Language	English
Page	87
Name of Supervisor	Ghodrat Moghadampour

The purpose of this thesis was to build an online investment and loan application for Zoan Oy, a software company whose main office is located in Lahti, Finland.

There are mainly two types of users: customer and administrator. The role of customer depends on the action in the application. The customer can be either a borrower or an investor. What is more, the customer can be one guarantor or beneficiary.

The application was implemented as an online application using PHP and MySQL technologies. The business logic of the application was implemented using PHP Zend framework. For the user interface, Twitter Bootstrap framework was used to build the layout and responsive features of the application. The application allows users to register as customers, manage their personal information, apply for loan and investment. The application also allows the administrator to manage loan and investment applications as well as users.

Until current research version, all core features have been successfully implemented. In short, the borrower can create and update a quick loan application online and the investor can invest the required money. As the middle platform, the administrator can manage the information of the users and applications.

Keywords PHP, MySQL, Zend Framework, Online application

ACKNOWLEDGEMENT

I would like to thank all the people who helped me and inspired me during the final thesis period.

At the beginning, I would like to give my honest thanks to my tutor, meanwhile, my thesis's supervisor, Dr. Ghodrat Moghadampour. He not only instructs me the academic knowledge, but also the way how to handle problems in life. When I encountered difficulties, his patience and professional skills give me a lot of power to overcome the adverse circumstances.

Secondly, I would like to thank Professor Miikka Rosendahl, who is CEO of Zoan Oy. He has given me this interesting project and some basic background information which is the basement for my achievement. In the end, I will thank my beloved parents that encourage me from another remote country. Here are my deepest thanks again for all of you I mentioned above.

Contents

1	INTRODUCTION.....	7
2	TECHNOLOGY OVERVIEW.....	8
2.1	Zend Framework.....	8
2.1.1	ZF application structure.....	8
2.1.2	ZF main components and features.....	8
2.1.2.1	Security.....	8
2.1.2.2	Internationalization.....	9
2.1.2.3	Performance.....	10
2.2	Twitter Bootstrap Framework.....	10
2.2.1	Grid system and responsive design.....	11
2.2.2	Understanding the CSS stylesheet.....	11
2.2.3	Re-usable components.....	11
2.2.4	JavaScript plug-ins.....	11
2.3	Openshift Platform Overview.....	12
2.3.1	PaaS: Platform-as-a-Service.....	12
2.3.2	Main features of Openshift platform.....	12
2.3.2.1	Cooperation Development.....	13
2.3.2.2	One Namespace with more applications.....	13
3	CUSTOMER INVESTMENT SERVICE BOOK.....	14
3.1	Main Functions Specification.....	14
3.1.1	My profile	15
3.1.1.1	The Investor Profile.....	15
3.1.1.2	The Borrower Profile (additional).....	16
3.1.2	Manage users.....	17
3.1.3	Manage loans.....	18
3.1.4	Manage Investments.....	19
3.1.5	Borrowing.....	20
3.1.6	Investing.....	23
3.2	Class Hierarchy.....	26
3.2.1	Controller Class Diagram.....	26
3.2.2	Models Class Diagram.....	28
3.2.3	Forms Class Diagram.....	30
3.3	Detailed Description of Main Functions	33

3.3.1	Manage Users.....	33
3.3.2	Manage Loans.....	36
3.3.3	Manage Investments.....	37
3.3.4	Borrowing.....	38
4	RESPONSIVE DESIGN.....	41
4.1	Dashboard Design.....	41
4.2	Table Design.....	42
5	IMPLEMENTATION.....	44
5.1	Responsive layout.....	44
5.1.1	Layout Structure	44
5.1.2	Dashboard Layout.....	45
5.1.3	Table Layout.....	46
5.2	Manage User.....	47
5.3	Manage Loans.....	50
5.4	Manage Investments.....	54
5.5	Borrowing.....	56
5.5.1	Step 1: Input Loan Criteria.....	60
5.5.2	Step 2: Input Personal Information.....	61
5.5.3	Step 3: Input Income Information.....	62
5.5.4	Step 4: Input Confirm Information.....	63
5.5.5	Step 5: Agreement Step.....	67
5.5.6	Congratulation Step.....	68
5.6	Investing.....	69
5.6.1	Default Dashboard.....	75
5.6.2	Input Personal Information.....	75
6	TESTING.....	77
6.1	Borrowing.....	78
6.2	Investing.....	78
6.2.1	Invest Dashboard.....	79
6.2.2	Invest Step.....	79
6.3	Manage loans.....	80
6.3.1	Loans dashboard.....	80
6.3.2	Loan Detail.....	80
6.4	Manage Investments.....	81

	6
6.4.1 Investment Dashboard.....	81
6.4.2 Investment Detail.....	82
6.5 Manage User.....	82
6.5.1 Mange User Dashboard.....	82
6.5.2 Manage User Detail.....	83
6.6 My profile.....	83
6.6.1 Personal Details.....	84
7 CONCLUSIONS.....	85
8 FUTURE DEVELOPMENT.....	86
9 REFERENCES.....	87

1 INTRODUCTION

Nowadays, borrowings and lendings are happening everyday in human life. Everyone may have some difficult time needing urgent money. On the other hand, many people would prefer to invest on other project or person which has higher return interest, compared with saving money in bank. According to a recent report of loan market, the usage of peer-to-peer lending is increasing by 4.2% in the whole financial market.

Following the trend, Zoan, one of leading companies in IT service, has tried to step into the peer to peer loan market, together with Ok Perintä Oy, which has professional and rich experience in debt and credit field. After several discussions, Zoan Oy will response for building online investment and loan application in the technical part, meanwhile, Ok Perintä Oy will response for checking the income information of customer and then giving credit score to each application.

Considering that most users are middle-aged men according to the survey, the application shall be easier to use, from both borrower and investor point of view. Additionally, more and more people prefer to use phone and ipad to view the web site, so that responsive design should be considered into account, which provides the automation of resizing the layout according the screen width of devices, even when the customer switchs between horizontal gesture and vertical gesture.

In the programming architecture, system module-based design shall be prioritized first place due to scalability. It should be easier to integrate new features or modules into current system in the future development.

2 TECHNOLOGY OVERVIEW

Below is a brief description of the development framework and platform used in the web application. Three main parts included: Zend framework, Twitter Bootstrap framework, Openshift platform. In each part, there will be a definition and summary of main features of the framework or platform.

2.1 Zend Framework

Zend framework(ZF) is an open source, object-oriented web application framework implemented in PHP 5 and licensed under the New BSD License. /1/

2.1.1 ZF application structure

Zend Framework is implemented using 100% object-oriented code. The component structure of Zend Framework is somewhat unique; each component is designed with few dependencies on other components. This loosely coupled architecture allows developers to use components individually. We often call this a "use-at-will" design. The Conventional Modular directory structure allows you to separate different MVC applications into self-contained units, and re-use them with different front controllers. /2/

2.1.2 ZF main components and features

2.1.2.1 Security

Work is implemented using 100% object-oriented code. The component structure of Zend Framework is somewhat unique. Each component is designed with few dependencies on other components. This loosely coupled architecture allows developers to use components individually. We often call this a "use-at-will" design. The Conventional Modular directory structure allows you to separate

different MVC applications into self-contained units, and re-use them with different front controllers. /3/

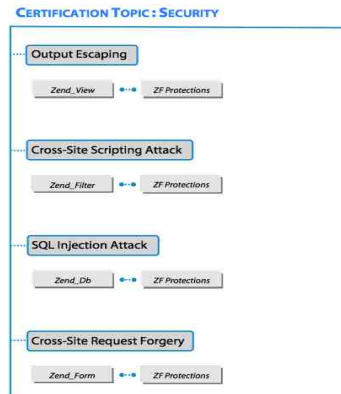


Figure 2.1 Zend Security

2.1.2.2 Internationalization

Internationalizing and localizing a site are fantastic ways to expand your audience and ensure that all visitors can get to the information they need. However, it often comes with a performance penalty. Below are some strategies you can employ to use. /4/

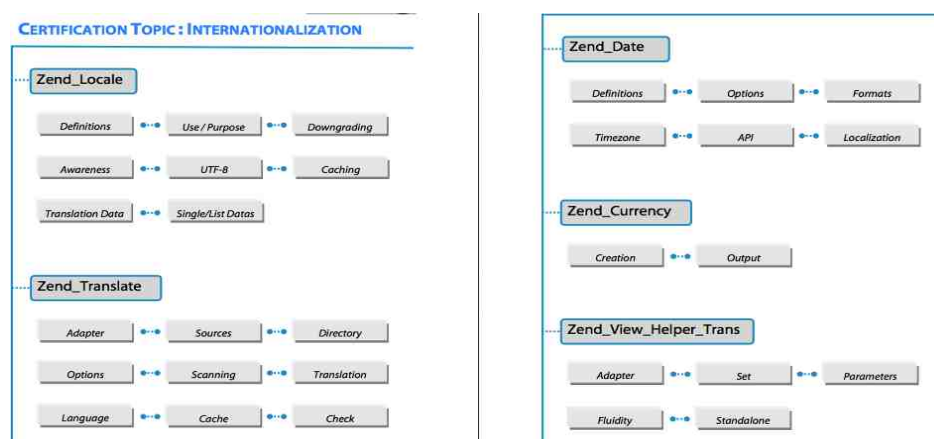


Figure 2.2 Zend Internationalization

2.1.2.3 Performance

In order to keep usage as simple as possible and also to support constantly changing schemas during development, `Zend_Db_Table` does some magic under the hood: on first use, it fetches the table schema and stores it within object members.

Fortunately, there are techniques for improving the situation, namely Use the metadata cache. `Zend_Db_Table` can optionally utilize `Zend_Cache` to cache table metadata. This is typically faster to access and less expensive than fetching the metadata from the database itself./5/

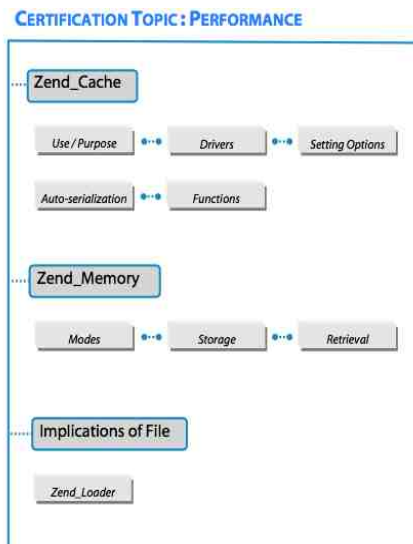


Figure 2.3. Zend Performance

2.2 Twitter Bootstrap Framework

Twitter Bootstrap is a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, charts, navigation and other interface components, as well as optional JavaScript extensions./6/

2.2.1 Grid system and responsive design

Bootstrap comes standard with a 940 pixel wide, grid layout. Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions and types of devices: mobile phones, portrait and landscape format, Tablets and PCs with a low and high resolution (widescreen). This adjusts the width of the columns automatically./7/

2.2.2 Understanding the CSS stylesheet

Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a browser and system-wide uniform, modern appearance for formatting text, tables and form elements./8/

2.2.3 Re-usable components

In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. These include buttons with advanced features (grouping of buttons or buttons with drop-down option, make and navigation lists, horizontal and vertical tabs, navigation, breadcrumb navigation, pagination), labels, advanced typographic capabilities thumbnails, formatting for warning messages and progress bar./9/

2.2.4 JavaScript plug-ins

The JavaScript components of Bootstrap are based on the jQuery JavaScript library. Plugins are accordingly found in the jQuery toolkit plugins. They provide additional user-interface elements such as dialog, tooltips and carousels. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields. In version 2.0, the following JavaScript Plugins are supported: Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collapse, Carousel and Typeahead./10/

2.3 Openshift Platform Overview

OpenShift is Red Hat's Cloud Computing Platform as a Service, namely PaaS. OpenShift gives developers a platform in the cloud to build, test, deploy, and run applications. OpenShift takes care of all the infrastructure, middleware, and management, setting you free to focus on what you do best, such as designing and coding applications.

2.3.1 PaaS: Platform-as-a-Service

The Difference Between PAAS with SAAS, Simplify: SAAS use the own's applications and data based on PAAS , the second major layer of the cloud is known as Platform-as-a-Service, or PaaS, which is sometimes called middleware. The underlying idea of this category is that all of your company's development can happen at this layer, saving you time and resources.

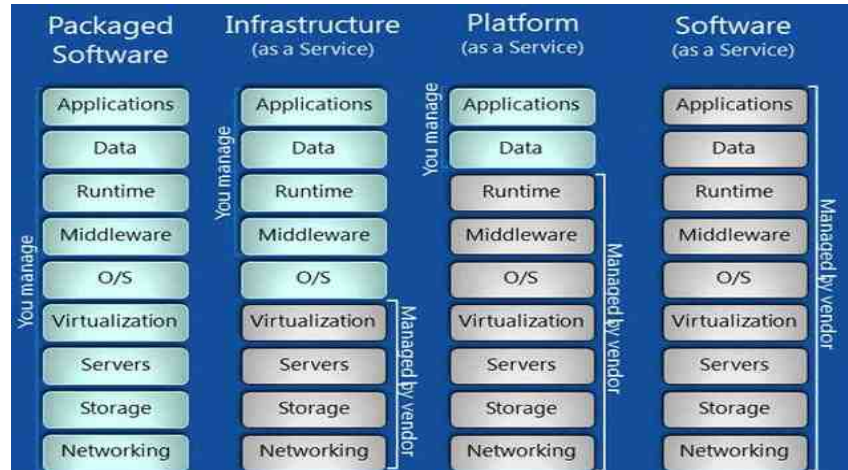


Figure 2.4. PaaS and SaaS

2.3.2 Main features of Openshift platform

For application development, the most important features are cooperation development and one namespace with more applications.

2.3.2.1 Cooperation Development

Public Keys System also increase the security

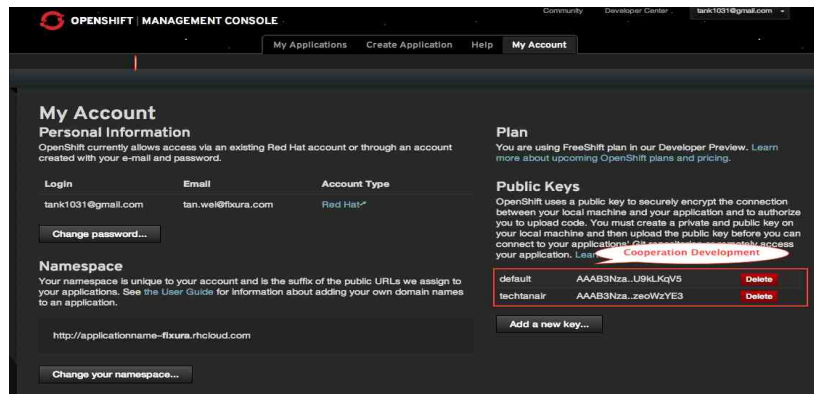


Figure 2.5 Openshift public key

2.3.2.2 One Namespace with more applications

Each application (application block) is sperated with others, only sharing some common data. Therefore, application can be easily added or removed according the user itself.

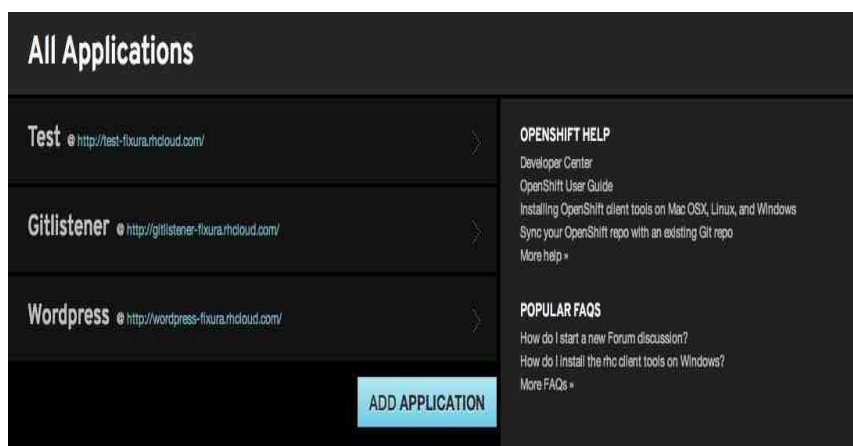


Figure 2.6. Openshift application panel

3 CUSTOMER INVESTMENT SERVICE BOOK

Below I will give a description about functional specification of the application, class structure and detail of main functions. In this section customer will have a general view how the development process was planned and conducted. Moreover, in the part description of main functions, important features of the application will be described in order to achieve customer's overall understanding before the implementation part which will analyze the implementation code.

3.1 Main Functions Specification

Customer who is registered in Zoan Oy, will have three modules to manage their own account. It not only includes his own information, such as profile, but also be capable of investing and borrowing a loan.

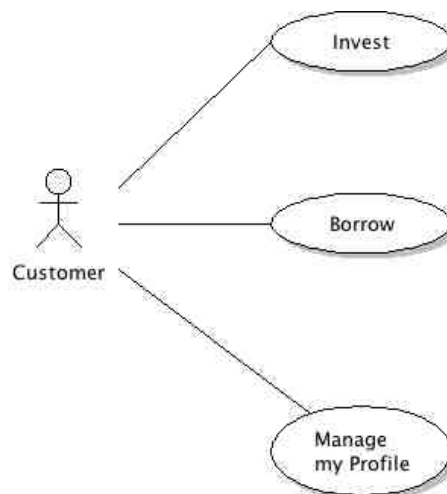


Figure 3.1 Customer Use Case Diagram

Zoan administrator employed by Zoan Oy will have four modules to manage loans, investments, and his own account. Four modules are displayed as icons in dashboard after logging in.

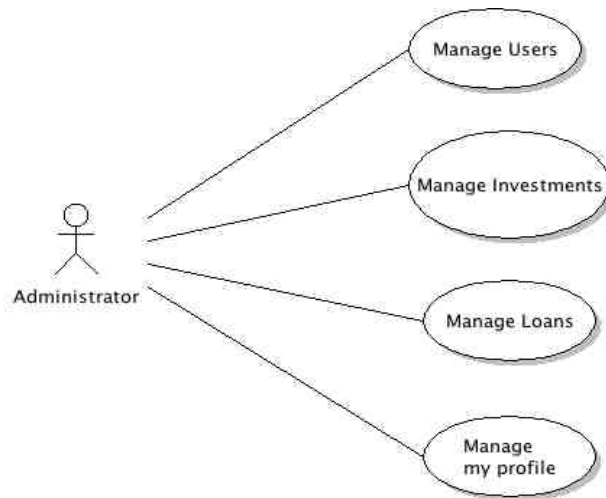


Figure 3.2 Administrator Use Case Diagram

3.1.1 My profile

The user shall be able to look at his or her profile and also make changes to some of the inputted information in section of personal details. The profile of an investor and a borrower differs slightly where Zoan needs more information about the borrower than the investor. It shall not be possible to apply for a loan without all necessary information about the lender.

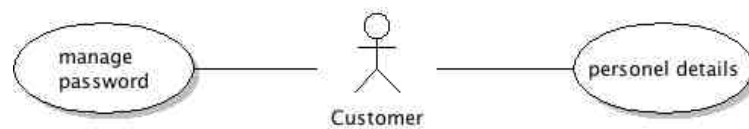


Figure 3.3 My profile Use Case Diagram

3.1.1.1 The Investor Profile

The following fields should be visible to the investor when looking at his or her profile.

- First Name

- Last Name
- E-mail
- GSM

Address Detail:

- Address
- Postal Code
- City
- Country
- Accept news letters from Zoan
- Accept news SMS from Zoan

3.1.1.2 The Borrower Profile (additional)

- resident and marital status
 - marital status (choose from)
 - other
 - divorced
 - single
 - sambo
 - married
 - widow
 - children (choose from)
 - none
 - 1
 - 2
 - more

- size of household (choose from)
 - ensamboende
 - two persons
 - tree persons
 - four persons
 - five or more persons
- summer place
 - none
 - have summer place
- residential type
 - other
 - house
 - block of flats/tower block
 - radhus
- economical situation
- educational degree
- income yearly

3.1.2 Manage users

As administrator, he or she should be able to manage the information and status of all the users in the admin panel.

First of all, the user table list will display in the container box after the administrator click the 'manage users' button. In the table list, only basic information of each user are shown, such as user id, user name.

Normally the administrator would like to check the detail information of the user, therefore, he or she could click the detail button which is in the end of each row.

What is more, the administrator is able to add new user. In the new user form, only basic information could be given by administrator, like username, password, email, and role. Those information would be updated by customer themselves after logging in.

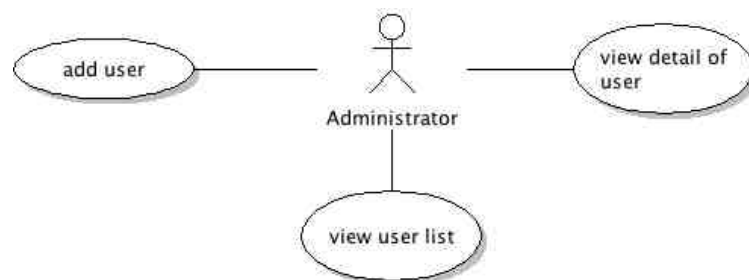


Figure 3.4 Manage user Use Case Diagram

3.1.3 Manage loans

After the borrower applied loan application, defaultly the status of loans is pending, meaning that needs the approval of Zoan Oy. Meanwhile, the pending applications are displayed in the loan table, sorting by the submit date.

As administrator of Zoan investment application, he is responsible for check the information of loan application.

Firstly, the basic loan information in the loan table list could show the basic but important information, such as loan id, loan amount, loan duration, loan interest. Continuely, the administrator could also check the detail of loan application, such as loan term, personal information, income information. Additionally, the detail of borrower could be found in the detail page too, because the administrator also need to check the borrower information.

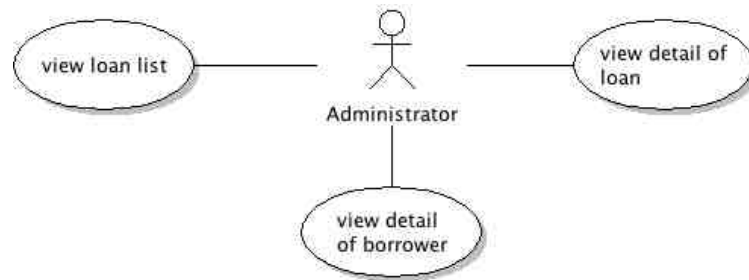


Figure 3.5 Manage loan Use Case Diagram

3.1.4 Manage Investments

To make one investment application, firstly, the investor need to choose one loan opportunity which is approved by zoan oy. Then he or she would to make the investment application step by step.

On the other hand, the administrator of Zoan investment application is responsible for check the information of investment application.

Firstly, the basic investment information in the investment table list can only show the basic investment id, investment amount and investor country and state, because different countries have different law about the peer to peer investment service.

Table list of investments

It shall be possible to browse in what loan applications is active. The following information shall be available

- Generated date: when borrower got money
- Investment ID & number: autoinvest agreement number and loan application number
- Interest: interest rate for the loan application in %
- Investment & Maturity: amount invested and for how long time

- Last payment date: the due date when the loan should be repaid in full
- Payment batch status: what is the status of the loan applications and which how many batches are paid of total amount
 - Status active: normal status when all is OK
 - Status pending: the borrower has not yet got the loan or accepted the loan

Continuely, the administrator could check the detail information. The investor information, investment application information, and loan application information are those three main information group.

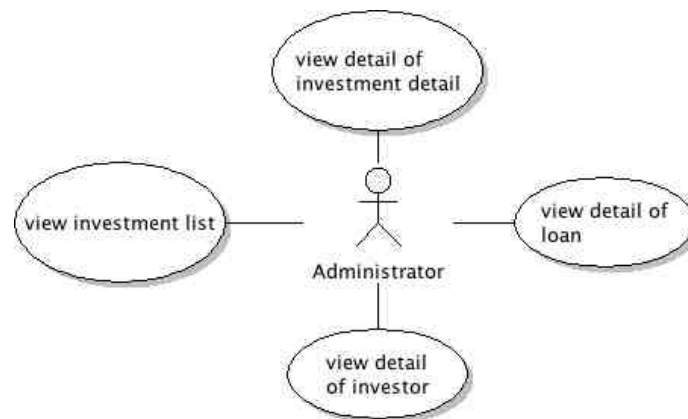


Figure 3.6 Manage investment Use Case Diagram

3.1.5 Borrowing

Borrowing module is the process of applying personal loan, which is available for all the customers.

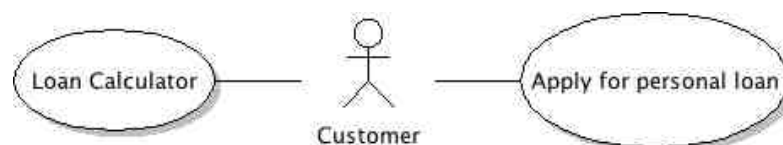


Figure 3.7 Borrowing Use Case Diagram

With the loan calculator, the borrower is able to apply for a loan according to desired criteria by the customer. Customer shall be guided when possible so that inputted criteria is relevant and that it is a high probability for the loan to get filled.

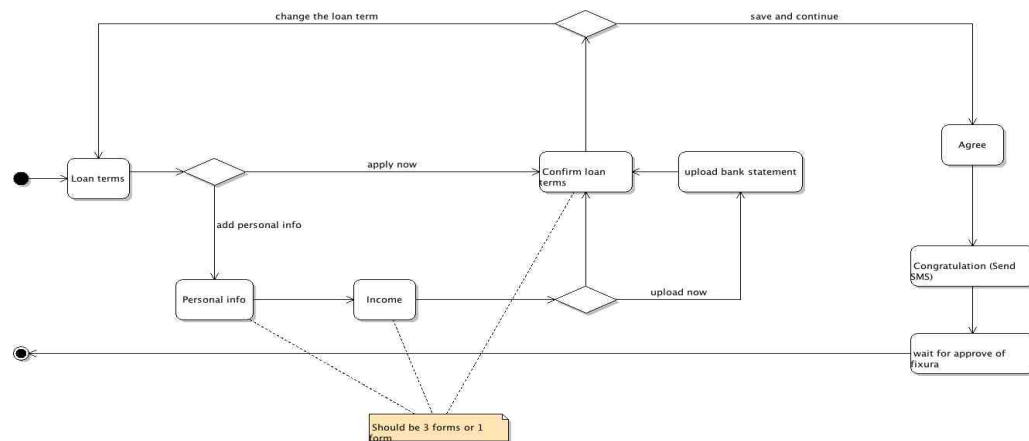


Figure 3.8 Borrow Activity Diagram

Step 1: Input Loan Term

Loan term main features:

- Amount: input amount in the box manually. When amount changes, monthly repayment amount changes accordingly.
- Repayment time: how long will the payment last. The borrower select the number of month from the options.
- Interest rate: customer chose interest rate he or she is willing to pay. It should notify customer that the system can not estimate how fast he or she will be approved this loan if not the optional scoring test is made. Customer can however change these values later after signed into user back web or created an free Zoan account.

- Payment day: borrower can choose the payment day during the whole month. Because the date of salary payment is different for borrowers.

Step 2: Personal Information

When customer applies for the loan and is logged in, system shall ask the customer personal info.

- Home Ownership
- Employment Status
- Loan Use
- Total WorkExperience In Years
- Start Of Employment

Step 3: Input Income Information

Customer shall be able to upload a bank statement (gif, png, pdf, doc, docx, odt..). Customer should also be notified that by uploading a bank statement, he or she will more likely get the loan faster and fully filled. Income information:

- How much do you earn
- Other income
- Comment on your income

Step 4: Confirm

In the confirm step, customer would have the overview and the repayment schedual of loan application, which will tell borrower how much money he or she will pay each month according the loan term inputted in first step. And borrower is advised to add one guarantor to increase the credit score. Our partner will contact the guarantor to sign another contract with him or her.

Step 5: Agreement

Before customer signs the agreement, he or she should be notified that the loan recipient information and general term.

What is more, the overview of loan application could be notified last time before signing the agreement.

Step 6: Congratulation

Before the loan is published, customer will receive one thanks message and be informed that Zoan will process their application.

3.1.6 Investing

Investors can manually chose loans to invest in. When borrowers apply for loans, they appear in on the loan applications site, browsable by logged in users.

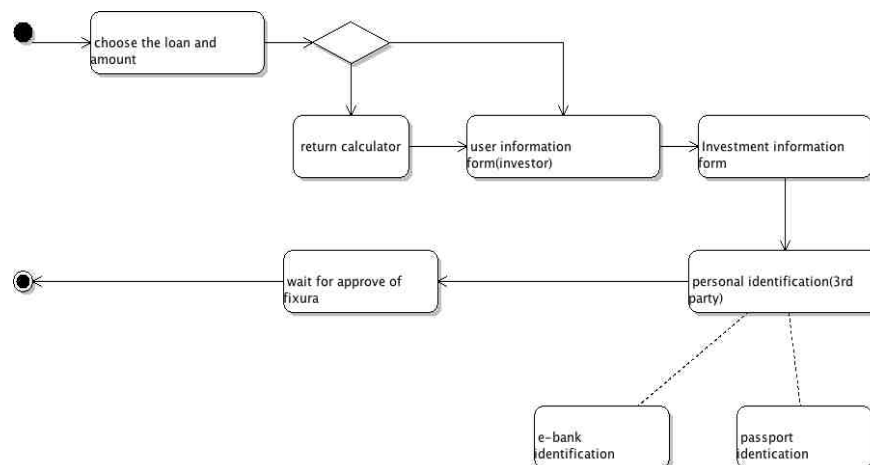


Figure 3.9 Invest Activity Diagram

Choose Loan Applications

As default, all loans should be listed in the loan applications list. The investor can chose an loan application and input the desired amount he or she is willing to

invest. Loan applications shall not be browsable by non registered or logged in customers. If not logged in or registered, only show examples.

The following fields with information are available

- ID: the id of loan application by the borrower
- Sum: the total amount of money the borrower has applied for
- Duration: for how long the borrower needs the money
- Interest: the chosen interest rate by the borrower

Step 1: Beneficiary Information

Defaultly the beneficiary is investor himself. But in some cases, the investor want to invest for others, like his son or daughter, parents, etc. In those condition, the beneficiary information could be different

- IBAN Account
- BIC of Bank
- Monthly withdraw
- Monthly withdraw day

Step 2: Investor Information

After investor has chosen investment amount and click “invest”, the investor is taken to the page with investor information. Normally different location and countries have different law about the peer to peer investment. Therefore the location information of investor is important.

Additionally, the personal information, like work experience, employment status, is also available.

Step 3: Income Information

Investor shall be able to upload a bank statement (gif, png, pdf, doc, docx, odt..).
Investor should also be notified that by uploading a bank statement, he or she will more likely get the loan faster and fully filled.

Income information:

- How much do you earn
- Other income
- Comment on your income

Step 4: Investment Information

The investment information is used to help Zoan Oy to gather the invest habit. Then according to the invest habit, the corresponding investment opportunity will be sent to customer.

- How often will you invest
 - daily
 - weekly
 - monthly
- What kind of investment do you prefer
 - short-term
 - mid-term
 - long-term
- What kind of risk willness
 - Little
 - Middle
 - High

Step 5: Electronic Signature

Before investor signs the signature, he or she should be notified that the zoan general term. What is more, the investor can choose the other loan application if he or she did change the investment idea.

Step 6: Congratulation

Before the investment application is published, customer will receive one thanks message and be informed that Zoan will process their application.

3.2 Class Hierarchy

Investment includes borrow, invest and administrator panel. Controller layer is engines' related information and model layer is database related details about individual actions of a controller. Depending on different layers, we have different class structures and methods. Some complex relations are cut into different parts to simplify the diagram.

3.2.1 Controller Class Diagram

Each controller contains many functions or actions. First of all, in the very beginning of each controller there is a basic action named "Init" where there is basic and common functions, such as add banner. Like common controller, the main action of a controller contains indexAction(), editAction() and addAction(). Most actions can include communications with model layer and form layer.

A member of special functions can also have been included. The structure of controller class can be visualized in below class diagram.

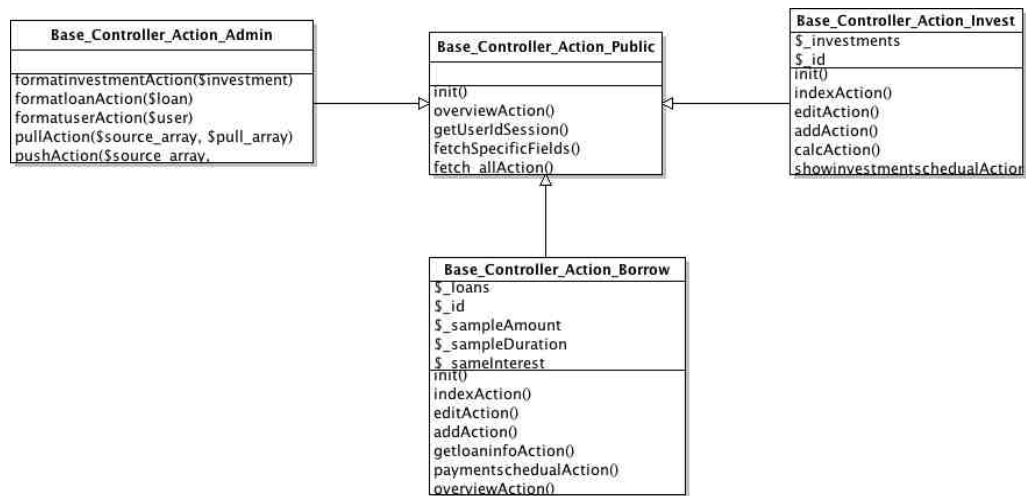


Figure 3.10 Controller Class Structure

Base_Controller_Action_Public class represents an instance of publicAction. It has four main functions:

- `init()`: This instance of class Base_Controller will call `init` function when the class is instanced, such as add banner.
- `getUserIdSesion()`: This function gives the `userId` which is stored in session.

Base_Controller_Action_Admin class supply the common functions for administrator panel to use, such as: manage loans, manage investments, manage users, inherits from Base_Controller_Action_Public class:

- `formatinvestmentAction($investment)`: fomat the investment information array, pulling some elements out from array or pushing some elements into array. This function needs the `pullAction()` and `pushAction()`.
- `formatloanAction($loan)`: fomat the loan information array, similar with above function.
- `formatuserAction($user)`: fomat the user information array, similar with above function too.

- `pullAction($source_array, $pull_array)`: remove the elements which name is stored in `$pull_array` from the `$source_array`
- `pushAction($source_array, $push_array)`: use `array_merge` to merge `$push_array` into `$source_array`.

`Base_Controller_Action_Borrow` class supply the common functions for borrow steps, such as loan information step, personal information step, etc. And inherits from `Base_Controller_Action_Public` class too.

- `Init()`: mostly initialization of class in model layer, called `Borrow_Model_Table_Loans`.
- `EditAction`: firstly display the values in the form stored in loan table of database. Secondly, the data should be updated after the values are changed by user.
- `AddAction`: it initialize `Borrow_Model_Row_Loan` class, receive form values, save into loan object. In the end, the object will save into database.

`Base_Controller_Action_Invest` class supply the common functions for invest steps, such as investor information, Income information, etc. And also inherits from `Base_Controller_Action_Public` class too.

- `Init()`: mostly initialization of class in model layer, called `Invest_Model_Table_Investments`.
- `EditAction`: firstly display the values in the form stored in Investment table of database. Secondly, the data should be updated after the values are changed by user.
- `AddAction`: it initialize `Invest_Model_Table_Investments` class, receive form values, save into investment object. In the end, the object will save into database.

3.2.2 Models Class Diagram

The structure of a model contains two primary roles: entity and table.

Table Class is responsive for mapping with the related table in database. Therefore, some attributes are predefined in abstract class, such as: `$_name`, `$_primary`, `$_rowClass`, `$_referenceMap`, `$_dependentTables`. The detail of those predefined attributes will be explained later.

Correspondingly, Entity Class is mapping with one row in that table. As the result, some attributes are predefined also, like: `$_data`, `$_tableClass`

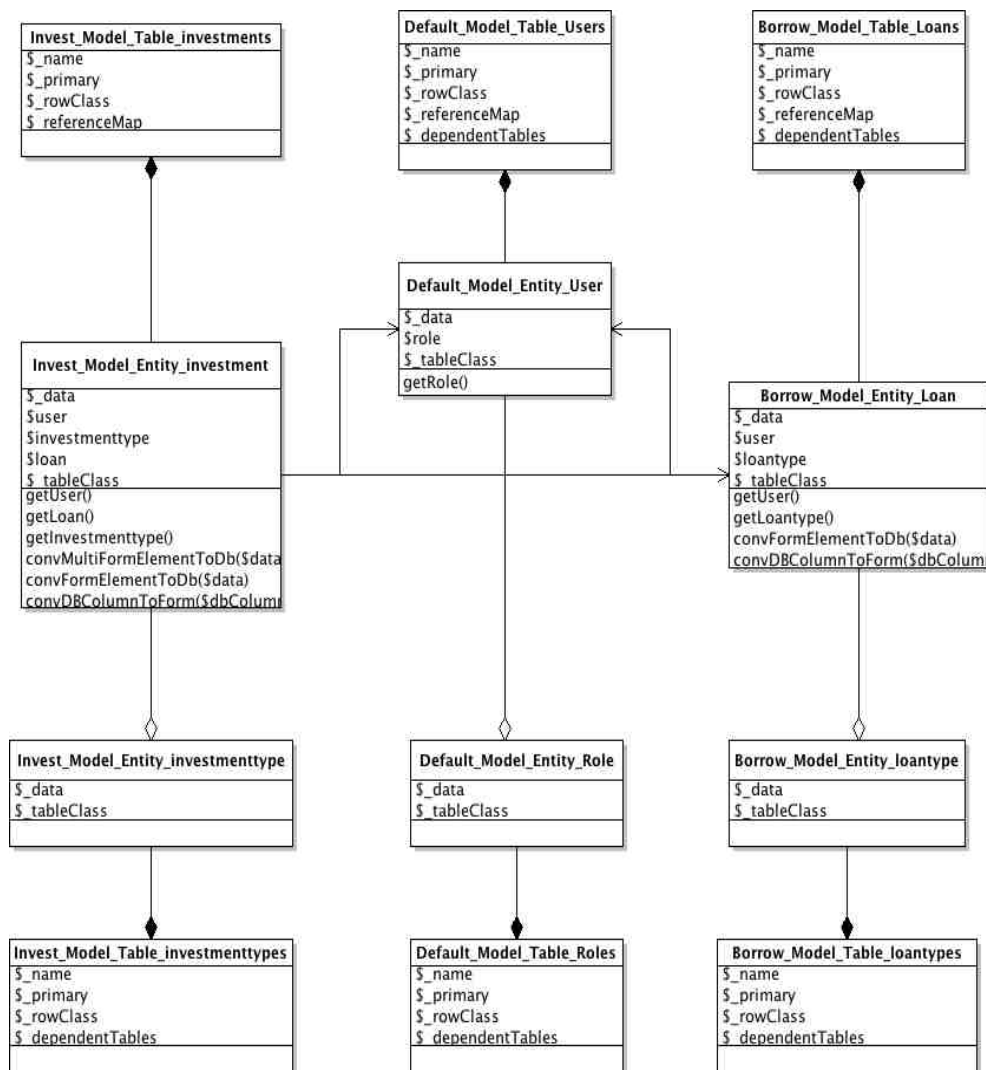


Figure 3.11 Model Class Structure(Every place)

Table Class will define the core information of table in database

- `$_name`: table name
- `$_primary`: table primary key
- `$_referenceMap`: parent role in One-To-Many relationship
- `$_dependentTables`: child role in One-To-Many relationship
- `$_rowClass`: corresponding Entity class name

What is more, Some other customized attributes can also be defined, besides the predefined attributes. Entity Class defines one row in that specific table of database.

- `$_data`: structure of table, namely elements in one row.
- `$_tableClass`: corresponding Table class name.

on the other hand, the operation on the Entity object will be defined in Entity Class, such as `getUser()`, `getLoantype()`, etc.

Investment Entity Class Operation:

- `getUser()`: get the corresponding user who applied the investment application.
- `getLoan()`: get the corresponding loan which the investment go to.
- `getInvestmenttype`: get which type or status of the investment.
- `convFormElementToDb`: convert form input values to data saved in database.
- `convDBCcolumnToForm`: convert data in one row of table to default values displayed in form.

3.2.3 Forms Class Diagram

`Zend_Form` simplifies form creation and handling in web application. It performs the following tasks:

- Element input filtering and validation
- Element ordering
- Element and Form rendering, including escaping
- Element and form grouping
- Element and form-level configuration

Therefore all the form class will inherit Zend_Form abstract class, namely create element, decorate element, group element.

Investment Related Form

During investing process, several steps is needed. In each step, one corresponding form is needed. In other word, the investing process is divided into several forms, such as financial status, goal for investment, electronic signature.

In the very beginning of the project, I prefer to use the subform to deal with this multipage, called steps in Zoan Investment project. But subform only save to database in the last step and store that data to session during the middle steps. In this solution, the step navigation is difficult to display and update the values which is inputted before and stored in session, on the other hand, the security to store data in database is also considered deeply.

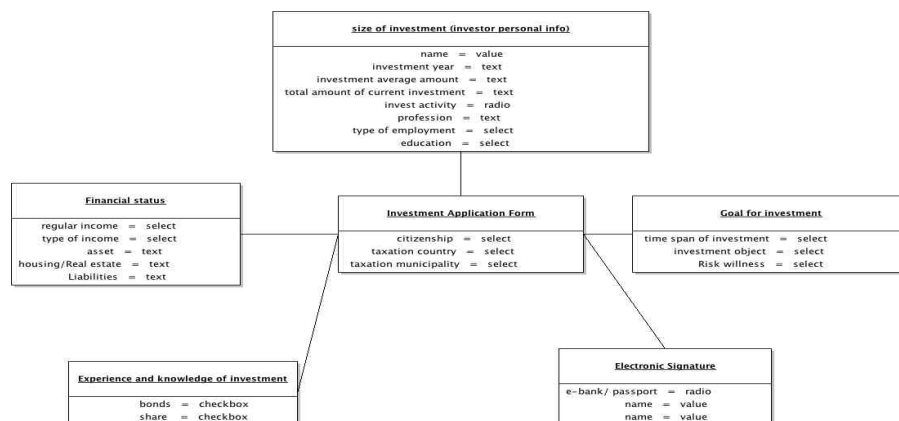


Figure 3.12 Investment Form Class Structure

User Related Form

There are two types of user: Borrower and Investor. Those have the common attributes, like username, password, etc. In the other hands, the different specific attributes belong to different type.

In Zend Form Class, there is one child form class inheriting the Zend_Form class, called Zend_Subform. That can be used for two types of users.

Sub forms serve several purposes:

- Creating logical element groups. Since sub forms are simply forms, you can validate subforms as individual entities.
- Creating multi-page forms. Since sub forms are simply forms, you can display a separate sub form per page, building up multi-page forms where each form has its own validation logic. Only once all sub forms validate would the form be considered complete.
- Display groupings. Like display groups, sub forms, when rendered as part of a larger form, can be used to group elements. Be aware, however, that the master form object will have no awareness of the elements in sub forms.

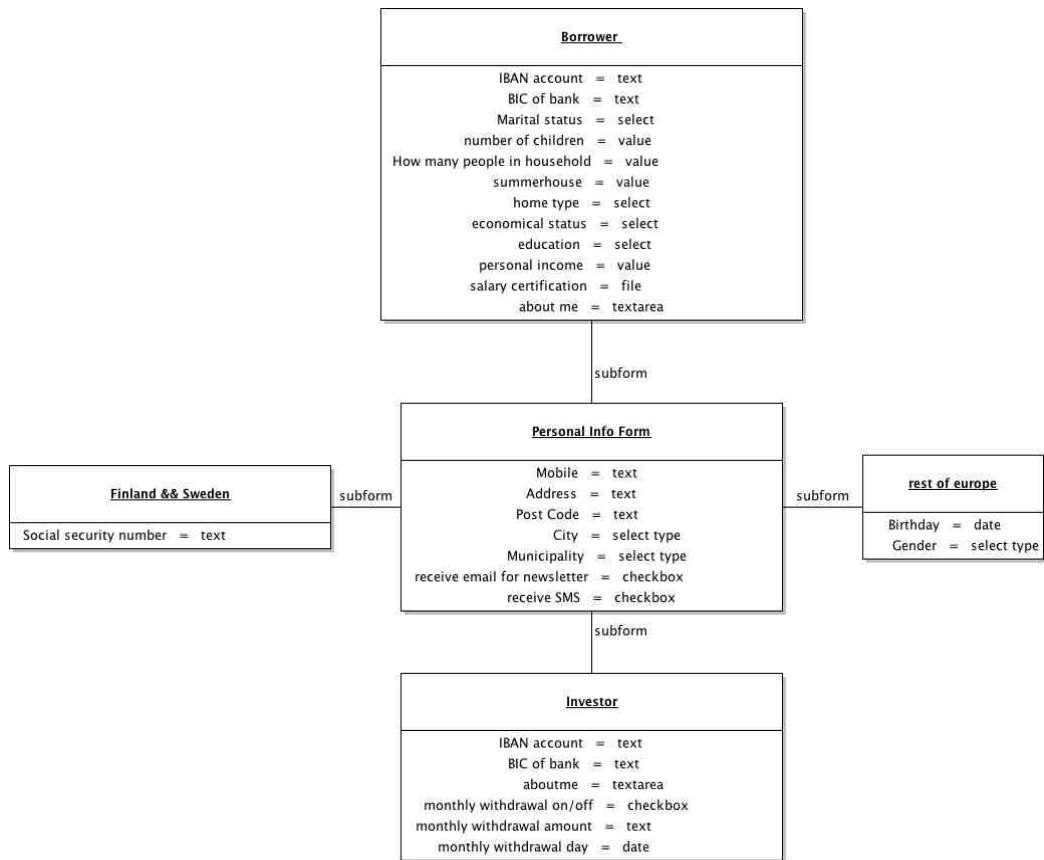


Figure 3.13 User Form Class Structure

3.3 Detailed Description of Main Functions

This part will give you the general idea of functionality of the application. All the functions listed in the use-case diagram mentioned above will be described in detail. The implementation codes of these functions will be analyzed in later chapters.

3.3.1 Manage Users

Four modules in admin panel will be shown after authentication as administrator. One of them is 'Manage Users', other three modules will be explained in other chapters.

After you click the 'Manage Users' Icon in dashboard, one user list will display as one table, shortly including user id, username, user role. What is more, there is one detail button for each row of table. Full information about the user will be shown in the user detail form, additionally some specific user informations could be updated by administrator.

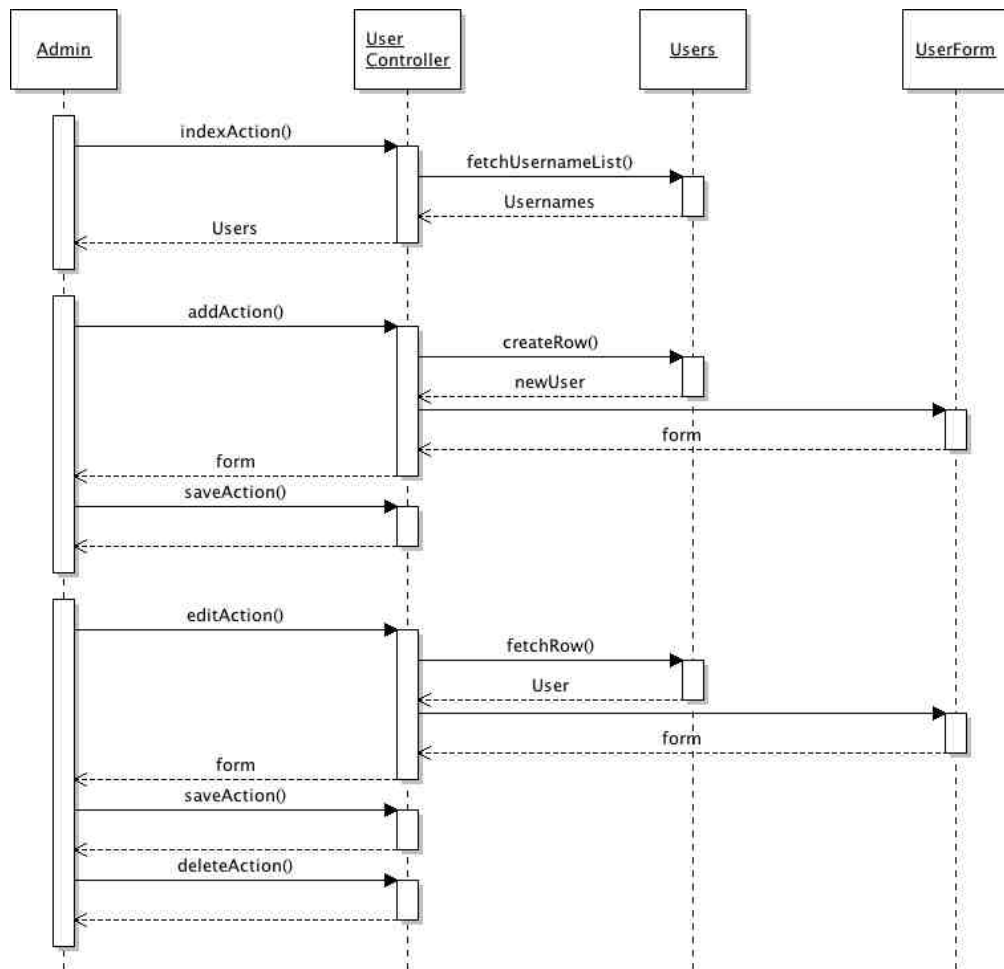


Figure 3.14. Sequence diagram – Manage Users

The sequence can be described below.

- Administrator click the 'manage users' icon in dashboard, UserController will call the model layer, namely Users to fetch all user names, then transfer to view layer to render it.

- Administrator click on 'Add User' button, addAction will be called and user is redirected to add user page where one user form exists.
- Administrator click on 'Detail' button, then is redirected to edit page where user form will display all user informations there.
- In the user detail page, administrator can either update the user information or delete the user account.

3.3.2 Manage Loans

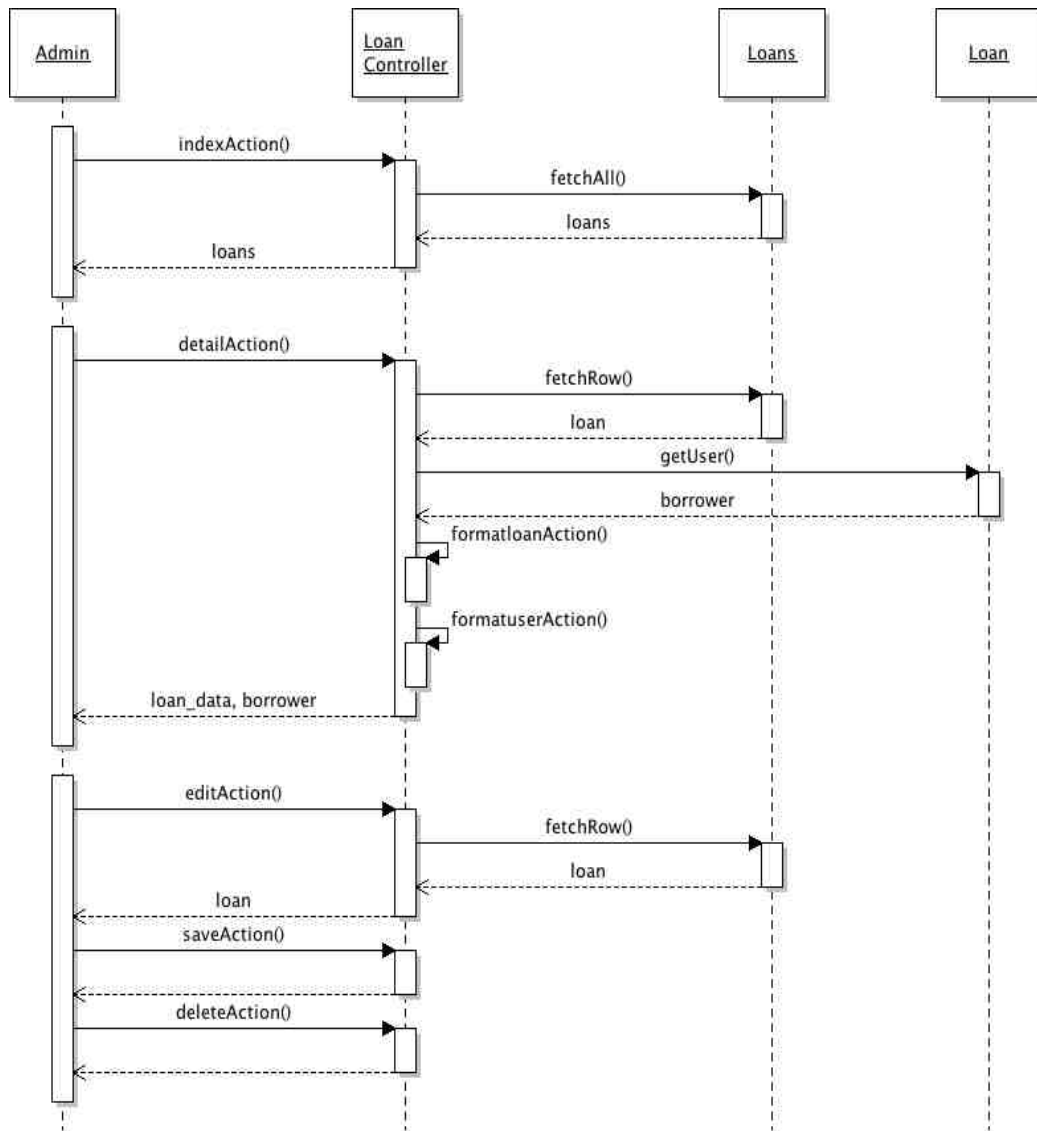


Figure 3.15. Sequence diagram – Manage Loans

The sequence can be described below.

- Administrator click the 'manage loans' icon in dashboard, LoanController will call the model layer, namely Loans to fetch all loans, then transfer to view layer to render it.

- Administrator click on 'Detail' button, LoanController will fetch one row according to given id, returning loan object. The loan object can get borrower information. Before passing to view layer, the loan data and user data should be filtered and formatted.
- In the user edit page, administrator can either update the loan information or delete the loan application.

3.3.3 Manage Investments

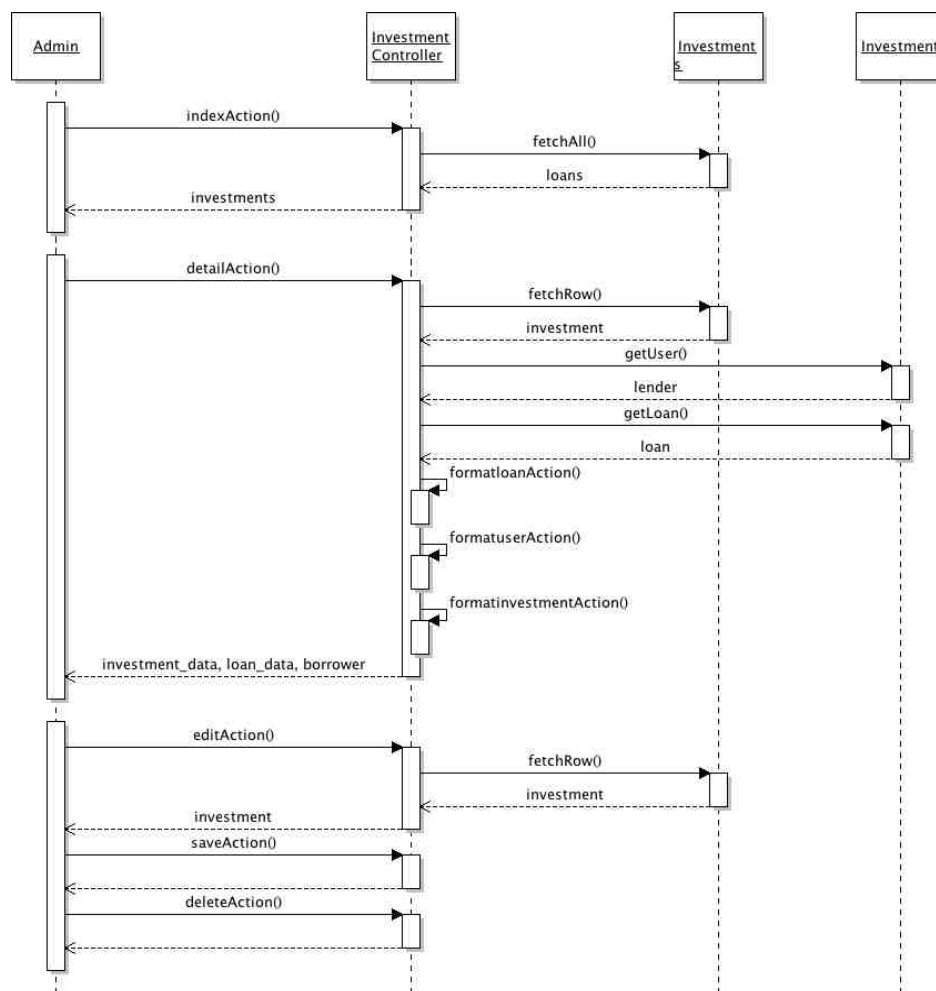


Figure 3.16. Sequence diagram – Manage Investments

The sequence can be described below.

- Administrator click the 'manage investments' icon in dashboard, InvestmentController will call the model layer, namely Investments to fetch investments, then transfer to view layer to render it.
- Administrator click on 'Detail' button, InvestmentController will fetch one row according to given id, returning investment object. The investment object can get lender information and loan information. Before passing to view layer, the investment data, loan data and user data should be filtered and formatted.
- In the user edit page, administrator can either update the investment information or delete the investment application.

3.3.4 Borrowing

The borrowing process includes five steps: Input Loan Criteria, Input Personal Information, Input Income, Confirm, Agreement. All steps work with same table: Borrow_Model_Table_Loans, and share similar function. First of all, the basic borrow sequence diagram will be introduced. The sequence can be described below.

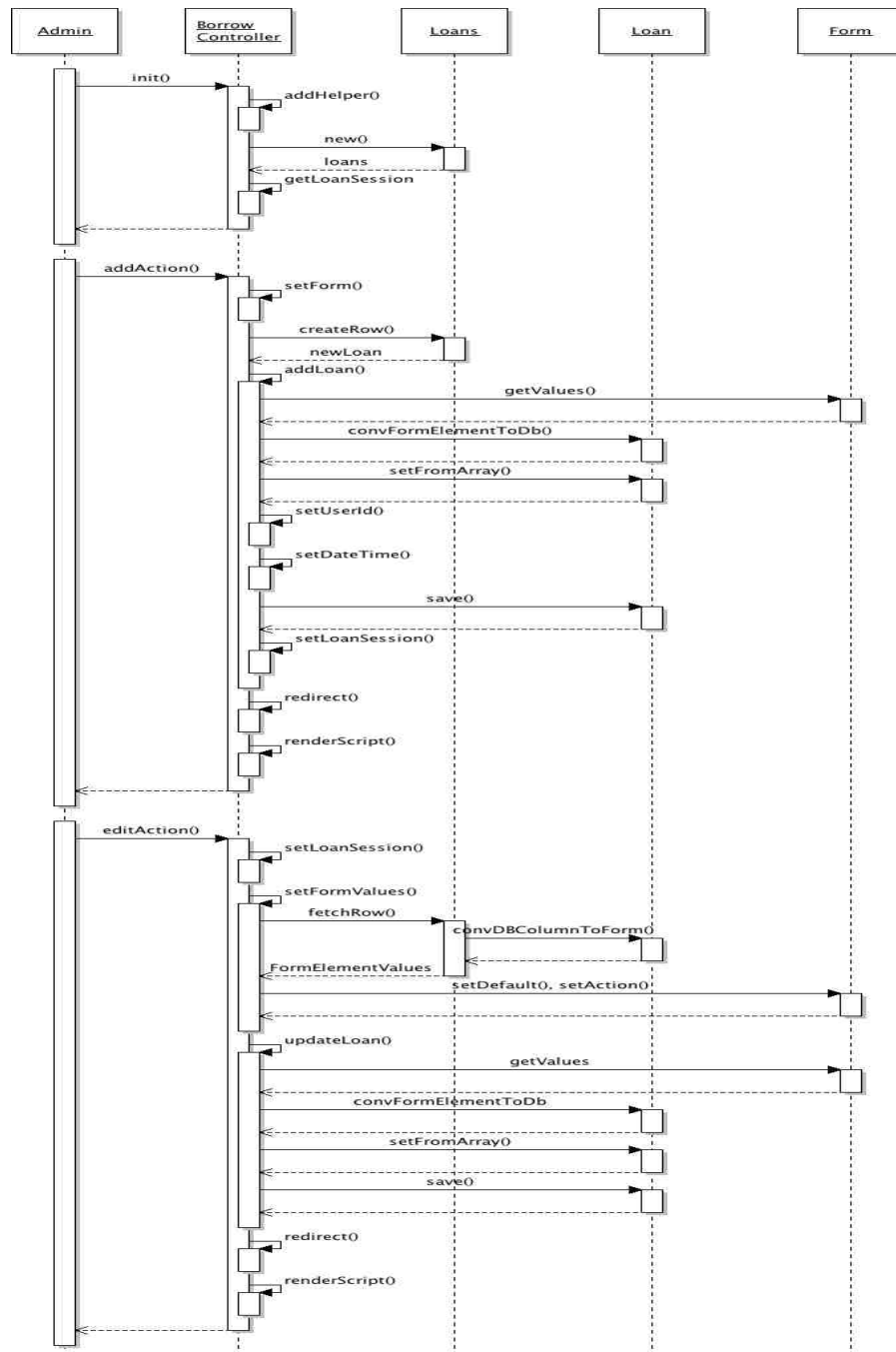


Figure 3.17. Sequence diagram – Borrow

- Customer click the 'Continue last application' icon in dashboard, BorrowController will call the editAction, namely display the values in form which is saved last time.

- Customer click the 'Apply new application' icon in dashboard, BorrowController will call addAction, meaning creating new blank form for user to input values
- Initial function will be called immediately BorrowController is called. Inside of that, the model layer, loan table, will be defined.

4 RESPONSIVE DESIGN

The core feature of this web platform is responsive design. Responsive web design is a web design approach aimed at crafting sites to provide an optimal viewing experience, such as easy reading and navigation with a minimum of resizing, panning, and scrolling, namely across a wide range of devices from desktop computer monitors to mobile phones.

4.1 Dashboard Design

After user log in, user will see one dashboard consisting of several icons. For desktop computer, icons display horizontally.

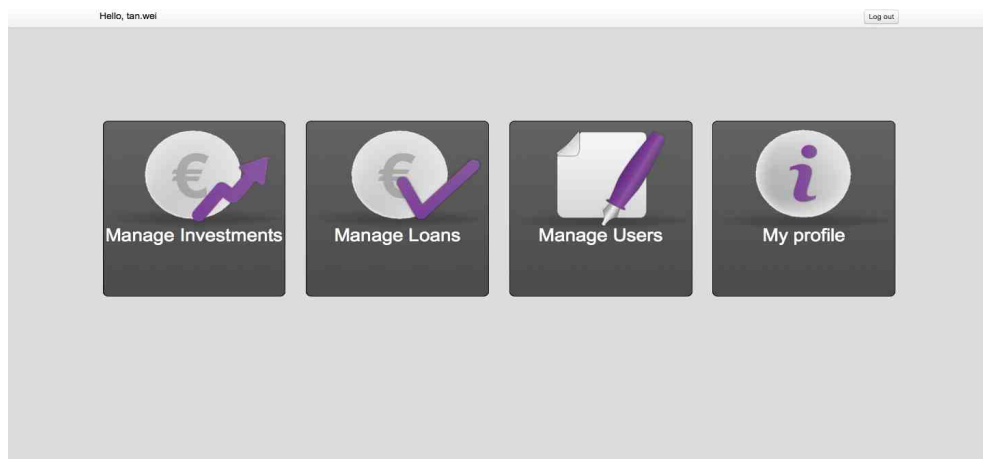


Figure 4.1 Dashboard design – desktop

on the other hand, icons would switch to display vertically for mobile device, which the height of screen is bigger than width of screen.

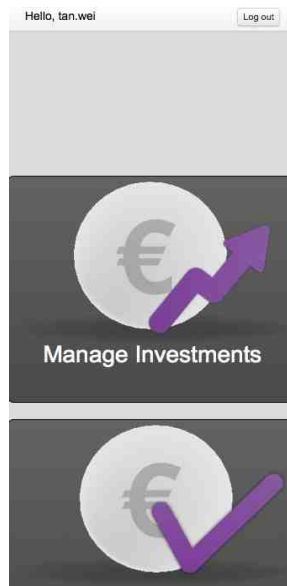


Figure 4.2 Dashboard design – mobile

4.2 Table Design

The administrator can view loan list table, user list table, and investment list table. The width of table should be stretched according to the screen width. For desktop monitor, table should be centered, leaving equal left space and right space.

 A desktop dashboard interface. At the top, it says "Hello, tan.wei" and has a "Log out" button. Below this is a large grey area containing a table titled "Pending Loan Application". The table has four columns: "Loan Id", "Loan Amount", "Loan Interest", and "Loan Duration". There are two rows of data. Each row has a "Detail" button to its right.

Loan Id	Loan Amount	Loan Interest	Loan Duration	
37	10000	28	3	Detail
46	5000	28	6	Detail

Figure 4.3 Table design – desktop

For the mobile device, conversely the table should fully stretch to whole screen because of the limited width.

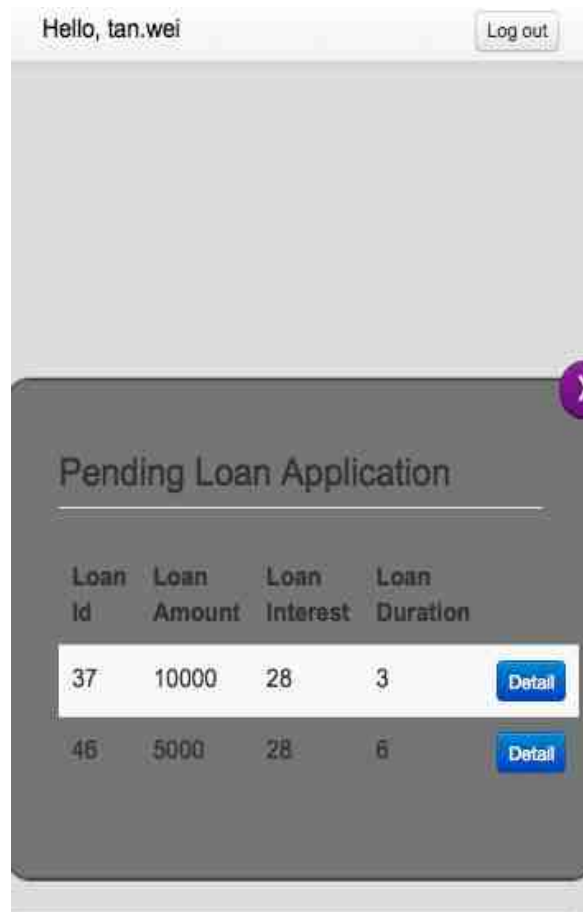


Figure 4.4 Table design – mobile

5 IMPLEMENTATION

Because of the complexity of this application, hereby I will give a description or explanation of some important functions and classes. Details of each function can be found in the comments of the attached source code.

5.1 Responsive layout

Twitter Bootstrap framework is chosen to be the foundation of responsive design. In responsive design, Media queries is the core feature, which allows for custom CSS based on a number of conditions—ratios, widths, display type, etc.

5.1.1 Layout Structure

In zend application structure, zend_layout implements a classic Two Step View pattern, allowing developers to wrap application content within another view, usually representing the site template. Such templates are often termed layouts by other projects, and zend framework has adopted this term for consistency.

In layout.phtml file, those below is the core application template and also designed according to responsive requirement.

```
<div class="main container-fluid">
<div class="row-fluid">
<span class="span1"></span>
<span class="span10">
<div class="content-container">
<div class="row-fluid">
<?php echo $this->layout()->content; ?>
<?php echo $this->inlineScript() ?>
</div>
<a class="fixura-close-button" href=?php echo $this->baseUrl("/index/index")?>></a>
</div>
</span>
<span class="span1"></span>
```

```
</div>
</div>
```

Snippet 1. Layout structure

Briefly caption about implementation of above structure code:

In bootstrap, template should start with tag with classname 'container-fluid'.

```
<div class="main container-fluid">
```

Each row of template should be given tag name 'row-fluid'.

```
<div class="row-fluid">
```

Each row consisted of 12 grid. Each grid is represented by one div tag or span tag named 'span1', namely 'spanX' means X grid.

```
<span class="span10">
```

5.1.2 Dashboard Layout

In the view layer, The navigation helpers are used for rendering navigational elements from Zend_Navigation_Container instances.

```
<div id="navigation" class="row-fluid">
<?php
$partial = array('index/menu.phtml', 'default');
$this->navigation()->menu()->setMaxDepth(0)->setMinDepth(0)->setPartial($partial);
echo $this->navigation()->menu()->render();
?>
</div>
```

Snippet 2. Dashboard responsive layout

Briefly caption about implementation of above dashboard code:

Those above shows how to render customized menus using a partial view script. By calling setPartial().

```
setPartial($partial)
```

you can specify a partial view script that will be used when calling render().

```
$partial = array('index/menu.phtml', 'default');
```

After we get page object in menu.phtml, we can call the methods of zend_navigation_page object, like getHref() and getParams().

```
$icon_folder = Zend_Controller_Front::getInstance()->getBaseUrl(). "/media/img/";
foreach ( $this->container as $page ) {
    $params = $page->getParams ();
    echo "<div class='span3 link homepage-item' style='height: 280px; padding-top: 1%;'>";
    echo "<a href=" . $page->getHref () . "></a>";
    echo $this->menu ()->htmlify ( $page ), PHP_EOL;
    echo "</div>";
}
```

Snippet 3. Navigation in dashboard responsive layout

5.1.3 Table Layout

For the bootstrap table makeup, it is easy-use to add class name 'table'. But when integrated with php, we need to build the table structure by hands.

```
foreach ( $this->users as $user ) {
    $table = '</tr><tr>';
    foreach ( $user->toArray () as $key => $content ) {
        if ($key == 'id') {$sid = $content;}
        $table .= '<td>' . $content . '</td>';
    }
    $table .= "<td><a href=" . $this->baseUrl () . "/user/edit/id/" . $sid . ">
    <button class='btn btn-mini btn-primary' type='button'>Detail</button></a></td>";
}
```

```
echo $table;
}
```

Snippet 4. Table responsive layout

Briefly caption about implementation of above table layout code:

In the view layer, view helpers is often used to generate wanted html code. `BaseUrl Helper` is one of them.

```
$this->baseUrl ()
```

5.2 Manage User

Admin click on the 'manage users' icon, User list will be displayed, shortly consisting of `UserId` and `UserName`. After user click on the 'detail' button, User information form comes out, completely consisting all information of specific user. Admin can update the information or delete the user account. Method code and explanation:

```
class Default_UserController extends Base_Controller_Action_Public {
// dispaly user list
public function indexAction() {
    $usertable = new Default_Model_Table_Users ();
    $usenames = $usertable->fetchUsernameList ();
    $this->view->assign ( 'users', $usenames );
}

public function detailAction() {
    // get user object
    $id = ( int ) $this->getRequest ()->getParam ( 'id' );
    $users = new Default_Model_Table_Users ();
    $user = $users->fetchRow ( "id = '$id'" );
    // format user information
    $user_data = $this->formatuserAction ( $user );
    // pass to view layer
    $this->view->assign ( 'user_data', $user_data );
}
```

```

// delete user according on the id
public function deleteAction($table) {
    $id = $this->getRequest ()->getParam ( 'id' );
    $table = new Default_Model_Table_Users ();
    $table->delete ( "id = '$id'" );
    $this->_redirect ( '/user/index' );
}

// edit user information
public function editAction() {
    // get user object
    $id = ( int ) $this->getRequest ()->getParam ( 'id' );
    $users = new Default_Model_Table_Users ();
    $user = $users->fetchRow ( "id = '$id'" );
    $role = $user->getRole ();
    $user_data = $user->toArray ();
    // get form object
    $form = new Default_Form_UserForm ();
    $form->setDefaults ( $user_data );
    $form->role_id->setValue ( $role ["id"] );
    $form->save->setLabel ( 'save' );
    $form->delete->setLabel ( 'delete' );
    // pass form object to view layer
    $this->view->assign ( 'form', $form );
    if ( $this->getRequest ()->isPost () ) {
        if ( $form->isValid ( $_POST ) && $form->save->isChecked () ) {
            // update user information if save button is clicked
            $this->saveAction ( $form, $user );
        } elseif ( $form->isValid ( $_POST ) && $form->delete->isChecked () ) {
            // delete user account if delete button is clicked
            $this->deleteAction ( $users );
        }
    }
}

}

}

public function addAction() {

```



```

        // get form object
        $form = new Default_Form_UserForm ();
        $this->view->form = $form;
        // create one new user object
        $users = new Default_Model_Table_Users ();
        $newuser = $users->createRow ();
        // insert user info into user object
        if ($this->getRequest ()->isPost ()) {
            if ($form->isValid ( $_POST )) {
                $this->saveAction ( $form, $newuser );
            }
        }
    }
    public function saveAction($form, $user) {
        // get form data
        $data = $form->getValues ();
        $data ['password'] = md5 ( $data ['password'] );
        // insert data into user object
        $user->setFromArray ( $data );
        // save user object to database
        $user->save ();
        $this->_redirect ( '/user/index' );
    }
}

```

Snippet 5. Manage user controller

Briefly caption about above functions in manage user module: Two buttons blinded with two actions is one place needed be taken care.

First of all, in the view layer, UserForm Object, create two elements

```

$save = $this->createElement ( 'submit', 'save' );
$save->setIgnore ( true );
$delete = $this->createElement ( 'submit', 'delete' );
$delete->setIgnore ( true );

```

In order to build two buttons: save and delete in one form, first of all, you should also define the label for elements in the controller layer: `$form->save`, `$form->delete`

```
$form->save->setLabel ( 'save' );
$form->delete->setLabel ( 'delete' );
// pass form object to view layer
$this->view->assign ( 'form', $form );
```

Snippet 6. Form buttons in manage user controller

After admin click one button, the controller will check which button is clicked, and then decide which action should response.

```
if ($this->getRequest ()->isPost ()) {
    if ($form->isValid ( $_POST ) && $form->save->isChecked ()) {
        // update user information if save button is clicked
        $this->saveAction ( $form, $user );
    } elseif ($form->isValid ( $_POST ) && $form->delete->isChecked ()) {
        // delete user account if delete button is clicked
        $this->deleteAction ( $users );
    }
}
```

Snippet 7. Update action in manage user controller

5.3 Manage Loans

Admin click on the 'manage loans' icon, Loan list will be displayed, shortly consisting of Loan amount and Loan interest. Core function code and explanation:

```
class Default_LoanController extends Base_Controller_Action_Admin {
    public function indexAction() {
        $loantable = new Borrow_Model_Table_Loans ();
        $select = $loantable->select ()->from ( 'loans', array ( 'id', 'term_amount',
            'term_interest', 'term_duration' ) )->where ( "typeid = 1" );
        $loans = $loantable->fetchAll ( $select );
```

```

        $this->view->assign ( 'loans', $loans );
    }

    public function deleteAction() {
        $id = $this->getRequest ()->getParam ( 'id' );
        $users = new Default_Model_Table_Users ();
        $users->delete ( "id = '$id'" );
        $this->_redirect ( '/user/index' );
    }

    public function detailAction() {
        $id = ( int ) $this->getRequest ()->getParam ( 'id' );
        $loans = new Borrow_Model_Table_Loans ();
        $select = $loans->select ()->where ( "id = '$id'" );
        $loan = $loans->fetchRow ( $select );
        // get loan info
        $loan_data = $this->formatloanAction ( $loan );
        // get borrower
        $borrower = $loan->getUser ();
        // format user information
        $borrower_data = $this->formatuserAction ( $borrower );
        $this->view->assign ( 'loan_data', $loan_data );
        $this->view->assign ( 'borrower', $borrower_data );
    }
}

```

Snippet 8. Manage loan module

Briefly caption about above functions in manage loans module:

After user click on the 'detail' button, Loan application information comes out, completely consisting all information of specific loan and borrower. Fetch the specific row of loans table according to given id, then return the loan object.

```
$loan = $loans->fetchRow ( $select );
```

In the loan object, there is one-to-many relationship with user object, meaning one user may have many loan applications. Hence, from the loan object, we can get parent role: the user object. The getUser function is defined in Borrow_Model_Entity_Loan class.

```
public function getUser() {
    if (! $this->user) {
        $this->user = $this->findParentRow ( 'Default_Model_Table_Users' );
    }
    return $this->user;
}
```

Snippet 9. Find user function in manage loan module

The one-to-many relationship between loan and user is defined in Borrow_Model_Table_Loans class and Users class.

First, in Borrow_Model_Table_Loans class, \$_referenceMap defines the parent role in the relationship. In this case, loans table have two parent-roles: Users and Loantypes.

```
protected $_referenceMap = array(
    'User' => array(
        'columns' => 'userid',
        'refTableClass' => 'Default_Model_Table_Users',
        'refColumns' => 'id'
    ),

    'Loan' => array(
        'columns' => 'typeid',
        'refTableClass' => 'Borrow_Model_Table_Loantypes',
        'refColumns' => 'id'
    )
);
```

Snippet 10. Relationship between user and loan model

on the other hand, in the parent table, for example, Users Table. Accordingly it should define the child role in the relationship. In this case, Default_Model_Table_Users have two child roles: Borrow_Model_Table_Loans and Invest_Model_Table_investments.

```
protected $_dependentTables = array('Borrow_Model_Table_Loans',
    'Invest_Model_Table_investments');
```

After fetching the loan row from database, the data need to be filtered and formatted so that some useless or security information will not be passed to view layer.

```
// get loan info
$loan_data = $this->formatloanAction ( $loan );
```

The formatloanAction function will also be called by other controller class, like investment controller. Therefore, it could be putted in the parent class in base library, where all the class are autoloaded defaultly Base_Controller_Action_Admin.

```
class Default_LoanController extends Base_Controller_Action_Admin
```

In Base_Controller_Action_Admin.

```
public function formatloanAction($loan) {
    $loantype = $loan->getLoantype ();
    $loan_raw_data = $loan->toArray ();
    // remove some elements
    $pull_array = array ('userid', 'id', 'typeid' );
    $pulled_array = $this->pullAction ( $loan_raw_data, $pull_array );
    // add some elements
    $push_array = array ('status' => $loantype ['name' ] );
    $loan_data = $this->pushAction ( $pulled_array, $push_array );
    return $loan_data;
}
```

Snippet 11. Format loan function in manage loan module

5.4 Manage Investments

Admin click on the 'manage investments' icon, Investments list will be displayed, shortly consisting of Investment amount and Investor location. Core function code and explanation:

```
public function detailAction() {
    // get loan
    $id = ( int ) $this->getRequest ()->getParam ( 'id' );
    $investmenttable = new Invest_Model_Table_investments ();
    $select = $investmenttable->select ()->where ( "id = '$id'" );
    $investment = $investmenttable->fetchRow ( $select );
    // get investment info
    $investment_data = $this->formatinvestmentAction ( $investment );
    // get lender info
    $lender = $investment->getUser ();
    $lender_data = $this->formatuserAction ( $lender );
    // get loan info
    $loan = $investment->getLoan ();
    $loan_data = $this->formatloanAction ( $loan );
    $this->view->assign ( 'investment_data', $investment_data );
    $this->view->assign ( 'investor', $lender_data );
    $this->view->assign ( 'loan', $loan_data );
}
```

Snippet 12. Manage investment controller

Briefly caption about above functions in manage investments module:

In the function view, Manage investments is similar with the 'Manage loans'. One difference is that Invest_Model_Table_investments class have three parent class: Users, Investmenttypes, Loans.

```
protected $_referenceMap = array(
    'User' => array(
        'columns' => 'userid',
        'refTableClass' => 'Default_Model_Table_Users',
        'refColumns' => 'id'
```

```

    ),
    'Invest' => array(
        'columns' => 'typeid',
        'refTableClass' => 'Invest_Model_Table_Investmenttypes',
        'refColumns' => 'id'
    ),
    'Loan' => array(
        'columns' => 'loanid',
        'refTableClass' => 'Borrow_Model_Table_Loans',
        'refColumns' => 'id'
    )
);

```

Snippet 13. Relationship between investment, user and loan model

In the controller, investment also need to getUser and getLoan, which is defined in Invest_Model_Entity_investment.

```

public function getUser()
{
    if (!$this->user) {
        $this->user = $this->findParentRow('Default_Model_Table_Users');
    }
    return $this->user;
}

public function getLoan() {
    if (!$this->loan) {
        $this->loan = $this->findParentRow('Borrow_Model_Table_Loans');
    }
    return $this->loan;
}

public function getInvestmenttype()
{
    if (!$this->investmenttype) {
        $this->investmenttype = $this-
>findParentRow('Invest_Model_Table_investmenttypes')->toArray();
    }
}

```

```

    }
    return $this->investmenttype;
}

```

Snippet 14. Function to find parent row in manage investment module

After getting lender and loan, you also need to format and filter the information, meaning calling the `formatuserAction` and `formatloanAction`. That is the reason why this class should inherit from parent class: `Base_Controller_Action_Admin`.

```

// get lender info
$lender = $investment->getUser ();
$lender_data = $this->formatuserAction ( $lender );
// get loan info
$loan = $investment->getLoan ();
$loan_data = $this->formatloanAction ( $loan );

```

Snippet 15. Function to find and format parent row in manage loan module

5.5 Borrowing

Borrowing process includes five steps: Input Loan Criteria, Input Personal Information, Input Income, Confirm, Agreement. All steps work with same table: `Borrow_Model_Table_Loans`, and share similar function. Therefore, parent class is necessary to share public functions for child class.

Parent Class: `Base_Controller_Action_Borrow`. Core functions is below.

```

class Base_Controller_Action_Borrow extends Base_Controller_Action_Public {
    protected $_loans;
    protected $_id;
    protected $_SampleAmount;
    protected $_SampleDuration;
    protected $_SampleInterest;

    public function init() {

```



```

$banner = $this->_helper->block->add ( "banner" );
// id parameter
$loansession = new Zend_Session_Namespace ( 'loan' );
if (isset ( $loansession->id )) {
$this->_id = $loansession->id;
// for view parameter
$this->view->id = $this->_id;
}
// create one investment row
$this->_loans = new Borrow_Model_Table_Loans ();
}
public function indexAction() {
$this->editAction ();
}
public function editAction() {
// build to session for edit if it comes with parameters, not session
if ($this->_hasParam ( 'id' )) {
$this->_id = $this->_getParam ( "id" );
$loan_session = new Zend_Session_Namespace ( 'loan' );
$loan_session->id = $this->_id;
}
// get loan
$loanrow = $this->_loans->fetchRow ( "id = '$this->_id'" );
// special for the loans table
$FormElementValues = $loanrow->convDBCcolumnToForm ( $loanrow->toArray () );
// get loan form
$this->_form->setDefaults ( $FormElementValues );
$this->_form->setAction ( $this->_formAction );
$this->view->form = $this->_form;
// update operation
if ($this->getRequest ()->isPost () ) {
if ($this->_form->isValid ( $_POST )) {
// get Form Data
$Formdata = $this->_form->getValues ();
// special for the loans table
$DBdata = $loanrow->convFormElementToDb ( $Formdata );
$loanrow->setFromArray ( $DBdata );

```

```

// save to DB
$loanrow->save ();
}
// for navi step session
$module = $this->getRequest ()->getModuleName ();
$controller = $this->getRequest ()->getControllerName ();
$navi_session = new Zend_Session_Namespace ( $module );
$navi_session->$controller = true;
// redirection
$this->_redirect ( $this->_editRedirection );
}
$this->renderScript ( $this->_renderscript );
}

```

Snippet 16. Borrowing module controller

Briefly caption about above functions in public child class:

In the init action, loan object and loan id should be defined. Loan id is stored in session, which is `Zend_Session_Namespace('loan')`.

```

$banner = $this->_helper->block->add ( "banner" );
// id parameter
$loansession = new Zend_Session_Namespace ( 'loan' );
if (isset ( $loansession->id )) {
$this->_id = $loansession->id;
// for view parameter
$this->view->id = $this->_id;
}

```

Loans object should be created, which is defined in model layer.

```

// create one investment row
$this->_loans = new Borrow_Model_Table_Loans ();

```

In the edit function, first of all, it should get parameter 'id' from the request URL, and then store it into session.

```

if ($this->_hasParam ( 'id' )) {
    $this->_id = $this->_getParam ( "id" );
    $loan_session = new Zend_Session_Namespace ( 'loan' );
    $loan_session->id = $this->_id;
}

```

Loans table object is used to get loan row object from database.

```

// get loan
$loanrow = $this->_loans->fetchRow ( "id = '$this->_id'" );

```

Loan row object information should be converted to loan form information, so that it will display form as default values.

```

// special for the loans table
$FormElementValues = $loanrow->convDBCcolumnToForm ( $loanrow->toArray () );
// get loan form
$this->_form->setDefaults ( $FormElementValues );

```

formAction is defined in child class, but used in the parent class.

```

$this->_form->setAction ( $this->_formAction );
$this->view->form = $this->_form;

```

Update operation is the core of editAction. Before saving to database, form data is converted database data, so that setFromArray is called.

```

// update operation
if ($this->getRequest ()->isPost () ) {
    if ($this->_form->isValid ( $_POST )) {
        // get Form Data
        $Formdata = $this->_form->getValues ();
        // special for the loans table
        $DBdata = $loanrow->convFormElementToDb ( $Formdata );
        $loanrow->setFromArray ( $DBdata );
        // save to DB
        $loanrow->save ();
    }
}

```

```

}
// for navi step session
$module = $this->getRequest ()->getModuleName ();
$controller = $this->getRequest ()->getControllerName ();
$navi_session = new Zend_Session_Namespace ( $module );
$navi_session->$controller = true;
// redirection
$this->_redirect ( $this->_editRedirection );
}

```

Snippet 17. Update function in borrowing module

renderscript is used to define which script file is rendering. By this way, some extra script files can be avoided.

```

$this->renderScript ( $this->_renderscript );

```

5.5.1 Step 1: Input Loan Criteria

Child class: Borrow_LoanController, which inherits the public child class: Base_Controller_Action_Borrow.

```

class Borrow_LoanController extends Base_Controller_Action_Borrow {
protected $_form;
protected $_formAction;
protected $_editRedirection;
protected $_renderscript;
public function init() {
parent::init ();

// form
$this->_form = new Borrow_Form_StepOne ();
$this->_formAction = Zend_Controller_Front::getInstance ()-
>getBaseUrl() .'/borrow/loan/edit';
$this->_editRedirection = 'borrow/personalinfo/index/';
$this->_renderscript = 'loan/index.phtml';
}
}

```

```
}
}
```

Snippet 18. Loan controller in borrowing module

Briefly caption about above functions in Borrow_LoanController class:

In the child class, normally only four variables should be defined. Form inherits from Zend_Form, where form elements are defined and decorated.

```
$this->_form = new Borrow_Form_StepOne ();
```

FormAction defines which action will be called by the form submit button, namely edit action in loan controller.

```
$this->_formAction = Zend_Controller_Front::getInstance ()->getBaseUrl () .
'/borrow/loan/edit';
```

EditRedirection defines where the page goes after those above action as next step.

```
$this->_editRedirection = 'borrow/personalinfo/index/';
```

Renderscript defines which script will be used to render, namely index.phtml in loan folder.

```
$this->_renderscript = 'loan/index.phtml';
```

5.5.2 Step 2: Input Personal Information

Child class: Borrow_PersonalinfoController, which inherits the Base_Controller_Action_Borrow too. That means only different variable setting.

```
public function init() {
    parent::init ();
    // form
    $this->_form = new Borrow_Form_StepTwo ();
    $this->_formAction = Zend_Controller_Front::getInstance ()->getBaseUrl
().'/' . 'borrow/personalinfo/edit';
```

```

$this->_editRedirection = 'borrow/income/index/';
$this->_renderscript = 'personalinfo/index.phtml';
}

```

Snippet 19. Init function in borrowing module

Form class: Borrow_Form_StepTwo, which inherits the Zend_Form. Core function is explained below.

Form object add form elements, which is defined by creatingElement.

```

$this->addElement ( array ( $homeownership, $employmentStatus, $purpose,
    $TotalWorkExperienceInYears, $StartOfEmployment, $submit ) );

```

Form object set attributes for form object, like name and legend.

```

$this->setAttribs ( array ( 'name' => $this->_name, 'legend' => $this->_title )

```

Form object divided some elements into different group, called DisplayGroup.

```

$this->addDisplayGroup ( array ( 'homeownership', 'employmentStatus', 'purpose',
    'TotalWorkExperienceInYears', 'StartOfEmployment', 'saveandcontinue' ), $this->_groupname,
    array ( 'legend' => $this->_title ) );

```

Form element name would be annotation like 'groupname_elementname' for those group division.

```

// annotation for form
$this->setIsArray ( true );

```

Set form name

```

$this->setName ( $this->_name );

```

5.5.3 Step 3: Input Income Information

Form class: Borrow_Form_StepThree, which inherits the Zend_Form. To upload bank statement is necessary in this step. In contrast, controller layer is similar with

other steps. Code in form layer related file uploading process will be explained below.

File element in form is created

```
// Upload File
$bankstatement = new Zend_Form_Element_File('bankstatement');
```

Set label to file element, especially destination.

```
$bankstatement->setLabel('Bank statement:')->setDestination(UPLOAD_PATH);
```

Add validator to file element, including file number, file size, file type.

```
// ensure only 1 file
$bankstatement->addValidator('Count', false, 1);
// limit to 100K
$bankstatement->addValidator('Size', false, 1024000);
// only JPEG, PNG, and GIFs
$bankstatement->addValidator('Extension', false, 'jpg,png,gif');
```

Snippet 20. Bank statement in borrowing module

At last, to the form object, the special attribute should be defined

```
// Add Attribute
$this->setAttrib('enctype', 'multipart/form-data');
```

5.5.4 Step 4: Input Confirm Information

In confirm step, payment schedual and overview of loan is useful for borrower to make the final decision.

In the controller layer, Borrow_ConfirmController, those three functions in parent class will be called: paymentschedualAction(), overviewAction(), editAction());

```
public function indexAction() {
    $this->paymentschedualAction ();
```

```

$this->overviewAction ();
// edit form
$this->editAction ();
}

```

In the parent class:

```

public function getloaninfoAction() {
// get row
$this->_loanrow = $this->_loans->fetchRow ( "id = '$this->_id'" );
// data special for the loans table
$FormElementValues = $this->_loanrow->convDBCcolumnToForm ( $this->_loanrow
->toArray () );
// get data for calculator
$this->_SampleAmount = $FormElementValues ['amount'];
$this->_SampleDuration = $FormElementValues ['duration'];
$this->_SampleInterest = $FormElementValues ['interest'];
}
public function paymentschedualAction() {
$this->getloaninfoAction ();
$calc = new Borrow_Model_Calculator ();
$payment_schedual_result = $calc->payment_schedual_calculator
( $this->_SampleAmount, $this->_SampleDuration, $this->_SampleInterest );
$this->view->schedual = $payment_schedual_result;
}
public function overviewAction() {
$this->getloaninfoAction ();
$calc = new Borrow_Model_Calculator ();
$loan_overview = $calc->overview_calculator
( $this->_SampleAmount, $this->_SampleDuration, $this->_SampleInterest );
$this->view->overview = $loan_overview;
}

```

Snippet 21. Payment schedual controller in borrowing module

Briefly caption about above functions in calculator class in model layer:

In the model layer, `Borrow_Model_Calculator` is the core class to calculate loan repayment schedual and overview with `overviewAction()` and `paymentschedualAction()`.

```
function payment_schedual_calculator($amount, $duration, $interest)
```

`Zend_Currency` handles all issues related to currency, money representation, formatting, exchange services and calculation.

```
$currency = new Zend_Currency('de_AT');
```

Displaying localized currency values by view helper

```
$helper = new Zend_View_Helper_Currency($currency);
```

Calculate the payment schedual, one is with currency values and other is only with values.

```
$payment_schedual = array();
$payment_schedual_no_currency = array();
```

Calculate the schedual by period, which is defined as duration

```
for ($period=1; $period<=$duration; $period++){
    // balance
    $current_balance[$period] = $this->loan_balance_calculator($amount, $duration, $interest,
    $period);
    if ($period == 1) {
        $previous_balance = $amount;
    }else{$previous_balance = $current_balance[$period-1];}
    // interest payment
    $principal_payment[$period] = $previous_balance - $current_balance[$period];
    // principal payment
    $interest_payment[$period] = $this->paymentMonthly - $principal_payment[$period];
    // Build the array without the currency
    $payment_current_no_currency = array
```

```

('paymentno'=>$period, 'amount'=>$this->_paymentMonthly,
'principal'=>$principal_payment[$period],
'interest' =>$interest_payment[$period], 'balance' => $current_balance[$period]);
array_push($this->_payment_scheduled_no_currency, $payment_current_no_currency);

```

Snippet 22. Calculate payment schedual in borrowing module

Display the currency values using the currency helper.

```

// Rebuild the array with currency
$payment_current = array('paymentno'=>$period,'amount'=>$helper->currency($this-
>_paymentMonthly), 'principal'=>$helper->currency($principal_payment[$period]),
'interest' => $helper->currency($interest_payment[$period]),
'balance' => $helper->currency($current_balance[$period]));
array_push($payment_scheduled, $payment_current);
}
return $payment_scheduled;
}

```

Snippet 23. Display currency in payment schedual

Paid-out Loan Amount: borrowed money – service fee. In our service

```

$Amount_Paid_Out = $amount - $this->_CreditScoringFee - $this->_LoanServiceFee;
// Loan Interest AND Principal
$Interest_payable = 0;
$Principal_payable = 0;
$this->payment_scheduled_calculator($amount, $duration, $interest);
foreach ($this->_payment_scheduled_no_currency as $Monthlypayment){
$Interest_payable = $Interest_payable + $Monthlypayment['interest'];
$Principal_payable = $Principal_payable + $Monthlypayment['principal'];
}
// Total cost
$Total_financing_cost = $Interest_payable + $this->_CreditScoringFee + $this-
>_LoanServiceFee;
// Total Payment
$Amount_repaid = $amount + $Interest_payable;

```

Snippet 24. Service fee calculator in borrowing module

Initial view helper for currency display.

```
// overview result array with decoration for view
$currency = new Zend_Currency('de_AT');
// initiate the helper
$helper = new Zend_View_Helper_Currency($currency);
```

Display the currency values using the currency helper.

```
$loan_info = array(
    'Amount' => $helper->currency($amount, array('precision' => 2)),
    'Duration' => $duration.' month(s)',
    'Maximum interest' => $interest.'%'
);
$loan_result = array(
    'Credit Scoring Fee' => $helper->currency($this->_CreditScoringFee, array('precision' => 2)),
    'Loan Service Fee' => $helper->currency($this->_LoanServiceFee, array('precision' => 2)),
    'Amount Paid Out' => $helper->currency($Amount_Paid_Out, array('precision' => 2)),
    'Interest Payable' => $helper->currency($Interest_payable, array('precision' => 2)),
    'Total Financing Cost' => $helper->currency($Total_financing_cost, array('precision' => 2)),
    'Amount Repaid' => $helper->currency($Amount_repaid, array('precision' => 2))
);
return $loan_overview = array($loan_info, $loan_result);
}
```

Snippet 25. Currency helper in borrowing module

5.5.5 Step 5: Agreement Step

In agreement step, the borrower can have the overview of loan application at last time and can not change any information about the loan after he or she click the agreement button.

In controller layer, it calles overviewAction in parent class.

```
public function indexAction(){
    $this->overviewAction();
}
```

```
// edit form
$this->editAction();
}
```

5.5.6 Congratulation Step

In congratulation step, the loan status would be changed to 'pending' status, waiting for the approval of administrator.

```
public function init() {
// ID parameter
$loansession = new Zend_Session_Namespace ( 'loan' );
if (isset ( $loansession->id )) {
$this->_id = $loansession->id;
}
// update the loan type
$loantable = new Borrow_Model_Table_Loans ();
// get row
$this->_loanrow = $loantable->fetchRow ( "id = '$this->_id'" );
$this->_loanrow->typeid = 1;
// save to DB
$this->_loanrow->save ();
}
```

Snippet 26. Congratulation controller in borrowing module

Briefly caption about above functions in congratulation step class:

Change the loan status and then save into database.

```
$this->_loanrow->typeid = 1;
$this->_loanrow->save ();
```

5.6 Investing

Investing process includes five steps: Input investor information, Input personal information, Input income, Input investment information, Sign contract,

Congratulation. All steps work with same table:

Invest_Model_Table_Investments, and share similar function. Therefore, parent class is necessary to share public functions for child class.

Parent Class: Base_Controller_Action_Invest. Core functions is below.

```

class Base_Controller_Action_Invest extends Base_Controller_Action_Public {
protected $_investments;
protected $_id;
public function init(){
$banner = $this->_helper->block->add("banner");
// ID parameter
$investmentssession = new Zend_Session_Namespace('investment');
if(isset($investmentssession->id)) {
$this->_id = $investmentssession->id;
// for view parameter
$this->view->id = $this->_id;
}
// create one investment row
$this->_investments = new Invest_Model_Table_investments();
}
public function indexAction()
{
$this->editAction();
}
public function editAction() {
// build to session for edit if it comes with parameters, not session
if ($this->_hasParam('id')){
$this->_id = $this->_getParam("id");
$loan_session = new Zend_Session_Namespace('investment');
$loan_session->id = $this->_id;
}
// get loan
$investmentrow = $this->_investments->fetchRow("id = '$this->_id'");
// special for the loans table
$FormElementValues = $investmentrow->convDBCcolumnToForm($investmentrow-
>toArray());

```

```

// get loan form
$this->_form->setDefaults($FormElementValues);
$this->_form->setAction($this->_formAction);
$this->view->form = $this->_form;
// update operation
if($this->getRequest()->isPost()) {
if($this->_form->isValid($_POST)) {
// get Form Data
$Formdata =$this->_form->getValues();
// special for the loans table
$DBdata = $investmentrow->convMultiFormElementToDb($Formdata);
$investmentrow->setFromArray($DBdata);
// save to DB
$investmentrow->save();
}
$this->_redirect($this->_editRedirection);
}
$this->renderScript($this->_renderscript);
}
public function addAction() {
// add one blank loan row
$newinvestment = $this->_investments->createRow();
// add the blank loan form
$this->view->form = $this->_form;
// add operation
if($this->getRequest()->isPost()) {
if($this->_form->isValid($_POST)) {
// get Form Data
$Formdata =$this->_form->getValues();
// special for the loans table
$DBdata = $newinvestment->convFormElementToDb($Formdata);
$newinvestment->setFromArray($DBdata);
// set the FK userid
$userid = $this->getUserIdSession();
$newinvestment->userid = $userid;
// set the created time
$date = new Zend_Date();

```

```

$date->setTimezone('Europe/Helsinki');
$newinvestment->sync_created = $date->toString('yyyyMMddHHmmss');
// set loanid
$this->_loanid = $this->_getParam('id');
$newinvestment->loanid = $this->_loanid;
// save to DB
$this->_investmentid = $newinvestment->save();
$loan_session = new Zend_Session_Namespace('investment');
$loan_session->id = $this->_investmentid;
}
$this->_redirect($this->_editRedirection);
}
$this->renderScript($this->_renderscript);
}

```

Snippet 27. Payment schedual controller in investing module

Briefly caption about above functions in public child class in invest module:

In the init action, object investment and investment id should be defined.

Investment id is stored in session, which is

`Zend_Session_Namespace('investment')`.

```

$banner = $this->_helper->block->add("banner");
// ID parameter
$investmentssession = new Zend_Session_Namespace('investment');
if(isset($investmentssession->id)) {
$this->_id = $investmentssession->id;
// for view parameter
$this->view->id = $this->_id;
}

```

Investment object should be created, which is defined in model layer.

```

// create one investment row
$this->_investments = new Invest_Model_Table_investments();

```

In the edit function, first of all, it should get parameter 'id' from the request URL, and then store it into session.

```
if ($this->_hasParam ( 'id' )) {
    $this->_id = $this->_getParam ( "id" );
    $loan_session = new Zend_Session_Namespace ( 'investment' );
    $loan_session->id = $this->_id;
}
```

Snippet 28. Check session in investing module

Investments table object is used to get investment row object from database.

```
// get investment
$investmentrow = $this->_investments->fetchRow("id = '$this->_id'");
```

Investment row object information should be converted to investment form information, so that it will display form as default values.

```
// special for the investments table
$FormElementValues = $investmentrow->convDBCColumnToForm($investmentrow-
    >toArray());
// get investment form
$this->_form->setDefaults($FormElementValues);
```

formAction is defined in child class, but used in the parent class.

```
$this->_form->setAction ( $this->_formAction );
$this->view->form = $this->_form;
```

Update operation is the core of editAction. Before saving to database, form data is converted database data, so that setFromArray is called. In this case, convMultiFormElementToDb() in model layer is created to deal with the data from form which is made up of subforms.

```
// update operation
if($this->getRequest()->isPost()) {
```



```

if($this->_form->isValid($_POST)) {
// get Form Data
$Formdata =$this->_form->getValues();
// special for the investments table
$DBdata = $investmentrow->convMultiFormElementToDb($Formdata);
$investmentrow->setFromArray($DBdata);
// save to DB
$investmentrow->save();
}
$this->_redirect($this->_editRedirection);
}

```

Snippet 29. Update function in investing module

renderscript is used to define which script file is rendering. By this way, some extra script files can be avoided.

```
$this->renderScript ( $this->_renderscript );
```

In add function, First of all, create new investment object by investments table.

```
$newinvestment = $this->_investments->createRow();
```

Then pass the form object to view layer.

```
$this->view->form = $this->_form;
```

After user click the add button, it will call add operation because it request by post.

```

// add operation
if($this->getRequest()->isPost()) {
if($this->_form->isValid($_POST)) {
// get Form Data
$Formdata =$this->_form->getValues();
// special for the loans table
$DBdata = $newinvestment->convFormElementToDb($Formdata);

```

```
$newinvestment->setFromArray($DBdata);
```

Snippet 30. Add investment function in investing module

Insert userid to database as foreign key.

```
// set the FK userid
$userid = $this->getUseridSession();
$newinvestment->userid = $userid;
```

Get current time according to the selected time zone, using `Zend_Date()`.

```
// set the created time
$date = new Zend_Date();
$date->setTimezone('Europe/Helsinki');
$newinvestment->sync_created = $date->toString('yyyyMMddHHmmss');
```

Snippet 31. Set TimeZone in investing module

Insert the second foreign key into database, namely loanid. That comes from parameter of URL.

```
// set loanid
$this->_loanid = $this->_getParam('id');
$newinvestment->loanid = $this->_loanid;
```

Save the entity object into database and store the investmentid created just now to session.

```
// save to DB
$this->_investmentid = $newinvestment->save();
$loan_session = new Zend_Session_Namespace('investment');
$loan_session->id = $this->_investmentid;
```

Snippet 32. Set session in investing module

5.6.1 Default Dashboard

In dashboard, all the possible investment opportunities would be displayed in table

```
public function showpaymentschedualAction() {
    $select = $this->_loans->select()->from('loans', array('id','term_amount','term_interest',
        'term_duration', 'userid', 'sync_created'))
    // select the user's latest not-approved loans
    ->where('typeid = 2')
    ->order('sync_created desc');
    $loans = $this->_loans->fetchAll($select);
    $this->view->assign('loans', $loans);
}
```

Snippet 33. Show payment schedual controller in investing module

Briefly caption about above functions in dashboard class in invest module:

Only display the approved investment applications which typeid equals 2

```
->where('typeid = 2')
```

The display order is according to the creating time.

```
->order('sync_created desc')
```

5.6.2 Input Personal Information

Personal information section consists of special information, tax related information and common personal information. Therefore SubForm is best solution in this case, meaning one form is made up of several subforms, sharing the same action and other attributes.

```
class Invest_Form_StepTwo extends Zend_Form {
    public function init() {
```

```

$SpecPersonalInfoForm = new Zend_Form_SubForm();
$InvestmentYear = $this->createElement('text', 'investmentyear');
$InvestmentYear->setLabel('How many years have you done investment:')->setValue('1')-
>setRequired(true);
$InvestmentAverAmount = $this->createElement('text', 'investmentamount');
$InvestmentAverAmount->setLabel('How much do you invest once usually:')->setValue('100')-
>setRequired(true);
$SpecPersonalInfoForm->addElements( array (
    $InvestmentYear, $InvestmentAverAmount
));
$TaxPersonalInfoForm = new Zend_Form_SubForm();
$Citizenship = $this->createElement('text', 'citizenship');
$Citizenship->setLabel('What is your citizenship:')->setValue('Finnish')->setRequired(true)
$TaxationCountry = $this->createElement('text', 'taxationCountry');
$TaxationCountry->setLabel('Where do you located:')->setValue('Finland')->setRequired(true);
$TaxationState = $this->createElement('text', 'taxationState');
$TaxationState->setLabel('Which city do you located:')->setValue('Helsinki')-
>setRequired(true);
$TaxPersonalInfoForm->addElements( array (
    $Citizenship,
    $TaxationCountry,
    $TaxationState,
));

```

Snippet 34. Multi subforms in investing module

Briefly caption about above functions in multi form case in invest module: After defining those three subforms, this form object would add subforms and set the form attribute, like name, action, etc.

```

$this->addSubForm($TaxPersonalInfoForm, "TaxPersonalinfo");
$this->addSubForm($SpecPersonalInfoForm, "SpecPersonalinfo");
$this->addSubForm(new Base_Form_Personalinfo, "personalinfo");
$this->setName("investor");
}
}

```

Snippet 35. Group multi subforms in investing module

6 TESTING

According to the project plan described at the beginning of this document, all objectives of the final thesis have been achieved in the Zend based framework associated with this project. The application has been deployed to Amazon cloud server and showed to Zoan people. All the required and extra features have been successfully implemented. The project is on its way to build its commercial version.

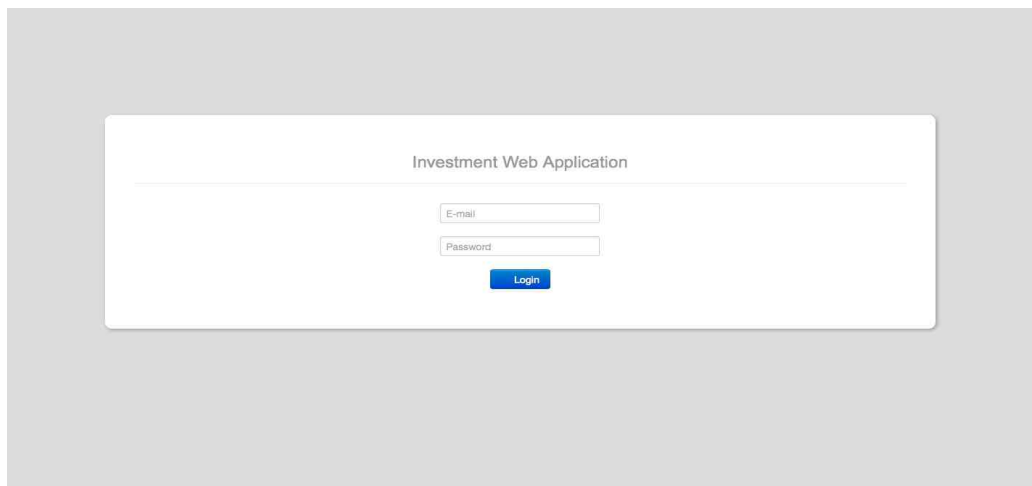


Figure 6.1. Login look and feel

Every implemented method was tested during coding and directly after it. Any found bug was immediately corrected. The applied testing approaches to methods are as follows:

- Checking the responsive design for different device: PC, Tablet, SmartPhone.
- Submitting invalid values of forms.
- Sending request with invalid information.
- Updating when no new information.

In addition, any place in the source code where there might occur an exception has been covered with try, catch clause. The Responsive design has been tested by simple performing operation. All feedbacks of the supervisor and customers have been taken into account and will be fixed when the application is moved to production version.

Below are the testing results and screenshots of some main features of the application.

6.1 Borrowing

During applying for loan application, several steps should be processed one by one. The font color of step navigation will turn to white after the step is processed, on the other hand, the font color is still grey. Below is one important step called confirm step in laptop version and mobile version.

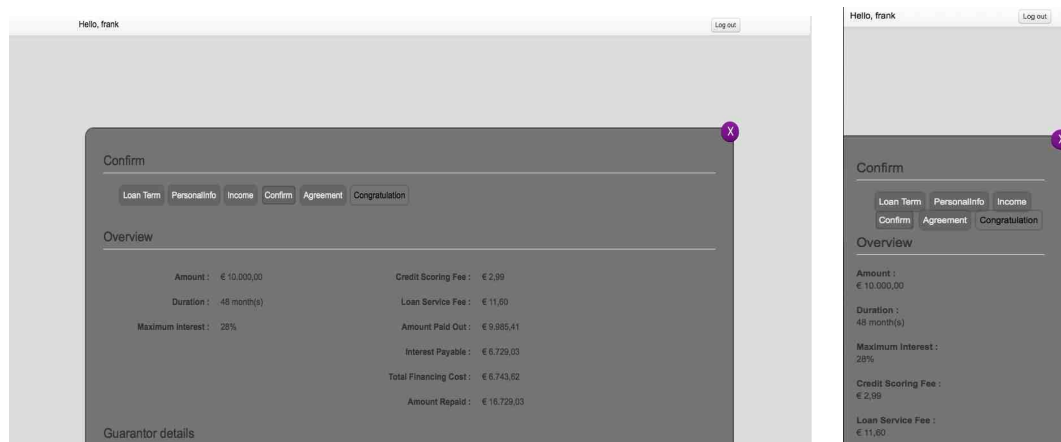


Figure 6.2. Confirm Step in Borrowing process

6.2 Investing

During applying investment application, in the begining, the customer will have a look of all approved loan application opportunity. And then customer can choose to invest some one or continue to invest last application.

6.2.1 Invest Dashboard

During applying investment application, in the beginning, the customer will have a look of all approved loan application opportunity. And then customer can choose to invest some one or continue to invest last application. Below is the dashboard in laptop version and mobile version.

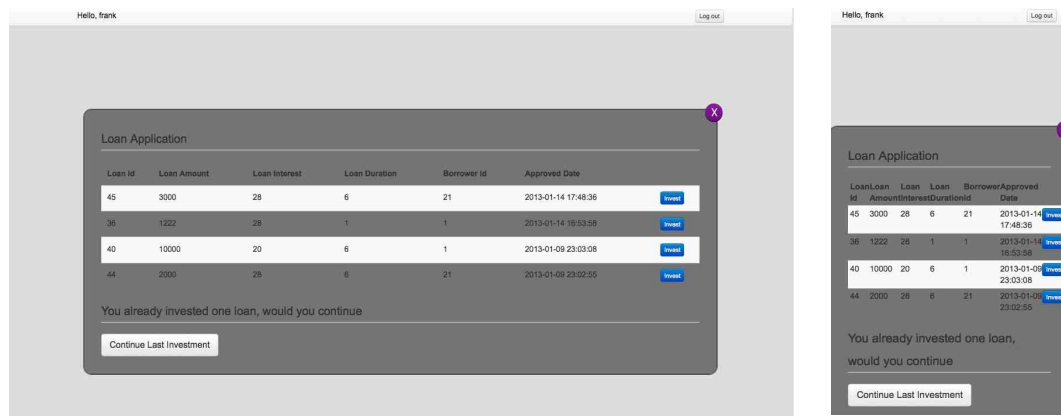


Figure 6.3 Dashboard in Investing process

6.2.2 Invest Step

During applying investment application, several steps should be processed one by one. The font color of step navigation will turn to white after the step is processed, on the other hand, the font color is still grey. Below is one important step called Investment step in laptop version and mobile version.

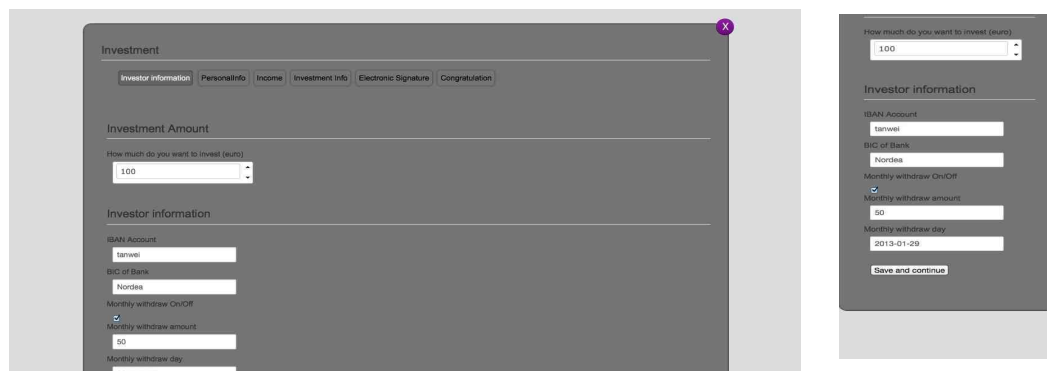


Figure 6.4 Step in Investing process

6.3 Manage loans

As the administrator, he has the first priority to view all the loans which wait for the approval. In dashboard, only some important information will be displayed in table, like loan id, amount, interest, and duration. After administrator click the detail button, all the information related the loan application would display in column.

6.3.1 Loans dashboard

In dashboard, only some important information will be displayed in table, like loan id, amount, interest, and duration. Below is list of loans in table style in laptop version and mobile version.

Id	Amount	Interest	Duration	
37	10000	28	3	Detail
46	5000	28	6	Detail
47	9999	28	6	Detail
48	10000	28	48	Detail

Figure 6.5 Manage loans

6.3.2 Loan Detail

In detail page, all the information related the loan application would display in column, such as loan information, borrower personal information. Below is in laptop version and mobile version.

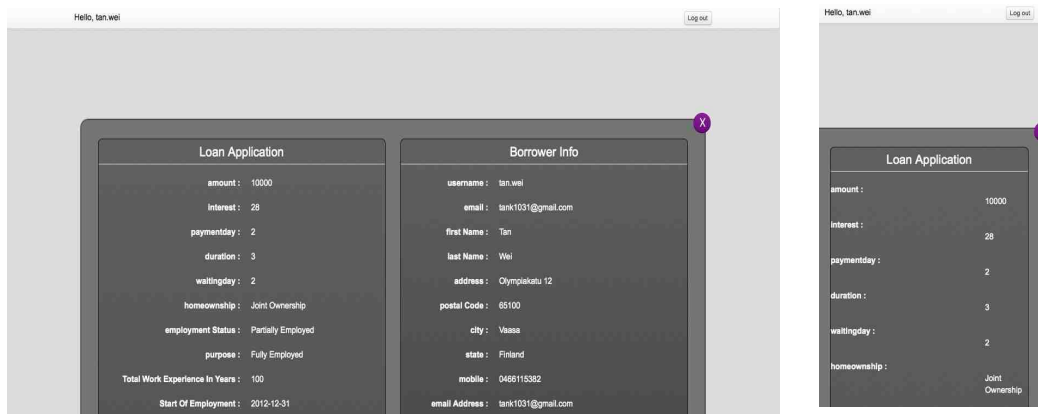


Figure 6.6 Manage loan detail

6.4 Manage Investments

As the administrator, he has the first priority to view all the investments which wait for the approval. In dashboard, only some important information will be displayed in table, like investment id, amount, country, and state. After administrator click the detail button, all the information related the investment application would display in column.

6.4.1 Investment Dashboard

In dashboard, only some important information will be displayed in table, like investment id, amount, country, and state. Below is list of investments in table style in laptop version and mobile version.

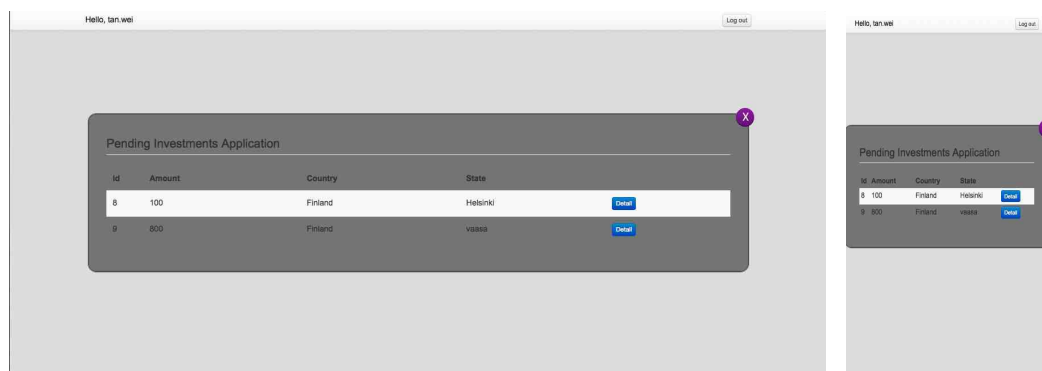


Figure 6.7 Manage Investments Dashboard

6.4.2 Investment Detail

In detail page, all the information related the investment application would display in column, such as investment applicaton, corresponding loan application, investor personal information. Below is in laptop version and mobile version.

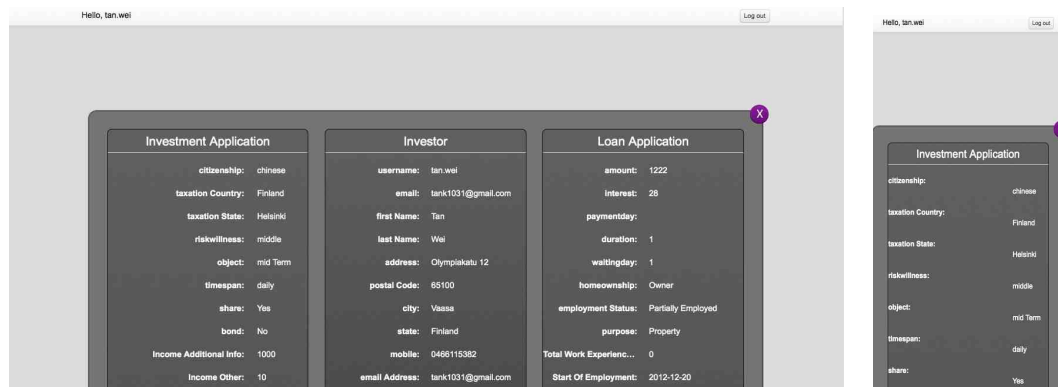


Figure 6.8 Manage Investment Detail

6.5 Manage User

As the administrator, he has the first priority to view all the users and their role in the application. In dashboard, only some important information will be displayed in table, like user id, username. After administrator click the detail button, all the information related the user would display in column.

6.5.1 Mänge User Dashboard

In dashboard, only some important information will be displayed in table, like user id, username. Additionally the administrator can create new user in dashboard. Below is list of users in table style in laptop version and mobile version.



Figure 6.9 Manage User Dashboard

6.5.2 Manage User Detail

In detail page, all the information related the user would display in column. Below is in laptop version and mobile version.

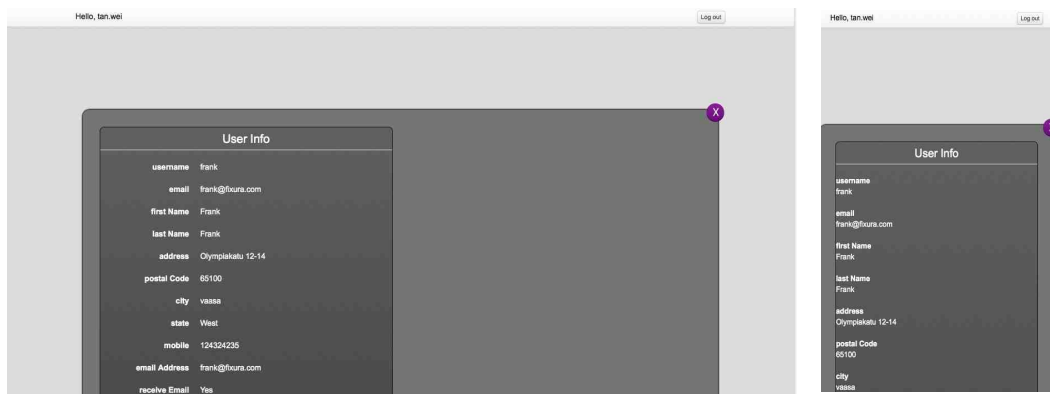


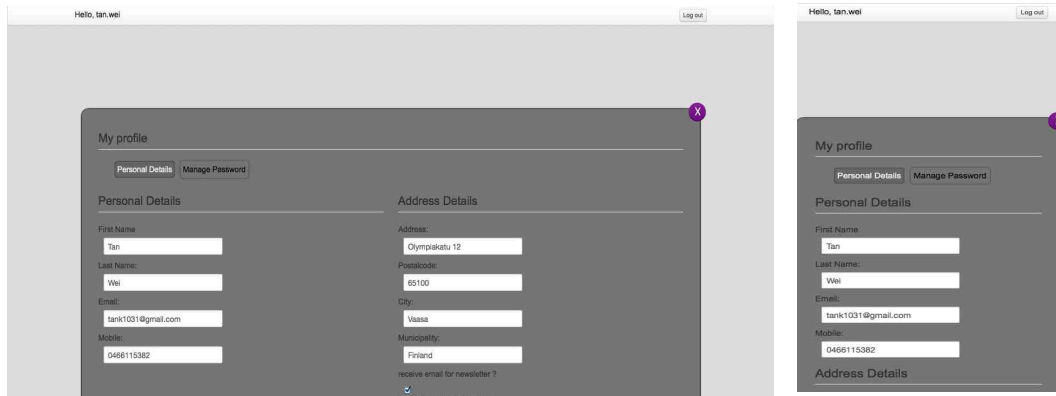
Figure 6.10 Manage User Detail

6.6 My profile

My profile is allowed for both administrator and common customer to access.

6.6.1 Personal Details

The general personal information is consisted of personal information and address information. Those information can be updated immediately the user click the save button.



The figure shows two screenshots of a web application's user profile page. The page is titled "My profile" and has a "Hello, tan.wei" header with a "Log out" button. Below the header are two tabs: "Personal Details" (selected) and "Manage Password".

The left screenshot shows the "Personal Details" and "Address Details" sections. The "Personal Details" section includes fields for First Name (Tan), Last Name (Wei), Email (tank1031@gmail.com), and Mobile (0466115382). The "Address Details" section includes fields for Address (Olympiakatu 12), Postcode (65100), City (Vaasa), and Municipality (Finland). There is also a checkbox for "receive email for newsletter?".

The right screenshot shows the "Personal Details" section, which is identical to the left screenshot, but the "Address Details" section is partially visible at the bottom.

Figure 6.11 Personal Detail

7 CONCLUSIONS

This project was motivated by the idea of 'peer to peer loan and investing' proposed by Zoan Oy. The objective was to build an online application for both borrowers and investors to make financial transactions.

The application allows users to register as customers, manage their personal information, apply for loan and investment. The application allows the administrator to manage loan and investment applications as well as users.

The application was implemented as an online application using PHP, MySQL and Twitter Bootstrap technologies. The business logic of application was implemented using PHP Zend framework. The financial transaction is the most challenging part of the application among all business logics, which requires higher security level. That is one of reasons why zend framework is chosen compared with Joomla. The most obvious benefit of using the zend framework is module-based system which separates so many functions into different modules.

In the technical view, this project was implemented by the usability of Zend framework, together with Twitter Bootstrap, which is used to build the user interface and mobile site for PC and mobile platforms.

One interesting feature is the responsive design layout, which provides the automation of resizing and rebuilding the layout according to the screen width or different devices, including laptop, ipad, mobile. What is more, it works even when customer switches between horizontal gesture and vertical gesture.

8 FUTURE DEVELOPMENT

Currently the application is on the way to be upgraded to commercial version. Additionally, it will be expanded to web service, which will supply restful web service to other cooperation partner. Representational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant web API design model.

And more modules are also needed to integrate into commercial version.

- My account: my account module is response for Zoan virtue bank account in the future, where the customer can view the account balance, account recent statement, and on the other hand, the customer can deposit money from linked bank account to Zoan and vice versa.
- My Loans: the borrower can access a workspace dedicated for information about his or her loans with the ability to take new loans. As with the dashboard, this user space shall follow the same concept with info boxes displaying information about loans that the borrower has applied for. The customer shall also be able to follow the process when he or she has applied for a new loan, how much is filled etc.
- My Investments: when the investor opens this page, he or she should be provided with a fast overview of investments, customers account and linked sub-accounts.
- Autoinvest: autoinvest is a tool with which an investor easily can set up an investment portfolio according to own criteria. When criteria are set, autoinvest constantly monitors loan applications for opportunities according to the criteria made by the investor. When a loan opportunity is found, autoinvest automatically invests in this loan application.

9 REFERENCES

/1/ Zend Framework Overview – Zend Framework Overview (2010). [WWW]. [referred 3.10.2013] available on the Internet:
<URL:<http://framework.zend.com/manual/1.12/en/introduction.overview.html>>

/2/ Zend MVC Structure – Zend Framework Overview(2010). [WWW]. [referred 3.12.2013] A vailable on the Internet:
<URL:<http://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html>>

/3/ PHP Security – Zend Framework Security (2010). [WWW]. [referred 3.15.2013] A vailable on the Internet:
<URL:<http://www.zend.com/en/solutions/php-security>>

/4/ Zend Internationalization – Zend Framework MVC (2010). [WWW]. [referred 3.17.2013] A vailable on the Internet:
<URL:<http://framework.zend.com/manual/1.12/en/performance.localization.html>>

/5/ Zend Performance – Zend Framework MVC (2010). [WWW]. [referred 3.15.2013] A vailable on the Internet:
<URL:<http://framework.zend.com/manual/1.12/en/performance.html>>

/6/ Twitter Bootstrap Introduction – Twitter Bootstrap Framework (2012). [WWW]. [referred 1.10.2013] A vailable on the Internet:
<URL:http://en.wikipedia.org/wiki/Twitter_Bootstrap>

/7/ Twitter Bootstrap Grid System – Twitter Bootstrap Framework(2012). [WWW]. [referred 1.10.2013] A vailable on the Internet:
<URL:http://en.wikipedia.org/wiki/Twitter_Bootstrap>

/8/ Twitter Bootstrap Css Stylesheet – Twitter Bootstrap Framework (2012). [WWW]. [referred 1.10.2013] A vailable on the Internet:
<URL:http://en.wikipedia.org/wiki/Twitter_Bootstrap>

/9/ Twitter Bootstrap Component – Twitter Bootstrap Framework (2012). [WWW]. [referred 1.10.2013] A vailable on the Internet:
<URL:http://en.wikipedia.org/wiki/Twitter_Bootstrap>

/10/ Twitter Bootstrap Javascript Plugin – Twitter Bootstrap Framework (2012). [WWW]. [referred 1.10.2013] A vailable on the Internet:
<URL:http://en.wikipedia.org/wiki/Twitter_Bootstrap>