EPOKA UNIVERSITY

Faculty of Economics and Administrative Sciences

Business Informatics Department

Software Engineering Project

June 2017

THIS PROJECT IS PREPARED BY:

ARMENO SADIKU

XHINA KAMBERI

XHULIO JAMAKU

ANA BASHA

PREPARED FOR:

M.Sc. IGLI HAKRAMA

Abstract

In this project we presented everything that we learned in the Software Engineering course. This project shows all the information needed to use and understand the web application that is built by our group.

The web application is called “IUTEcredit” and it is an application that will be used by the company with the same name. “IUTEcredit” is a company that provides consumer loans without collateral on the basis of personal ID. This application will help the company to keep client information about the loans, will fastens the process of giving a loan, will manage the employees and business and also will help the company to be more efficient and to have real time information about what’s going on at the company.

Table of Contents

This page intentionally left blank

Executive Summary

One of the most important aspects in the development of the software is to be clear what we are going to build. In other words, first we will work on understanding the problem and then build the code.

So, first section of the Software Requirements Specification (SRS) will provide not only an overall overview of the entire Software Requirements Specification document and project itself, but also the purpose and the scope.

The main aim of this document is to give an in-depth insight of the IUTE credit Albania software system, gather and analyze the problem statement in detail.

Project Overview

This document provides all the information related with the system software IUTE credit Albania.

There will be 6 sections, where each of them will present different information related with general product description, main users (and their characteristics) that the product will have, constrains, assumptions, functional and non-functional requirements, possible use scenarios and use cases. The idea, and the description of the product is presented in section 2, together with the users, and their characteristics. In the following section (section 3) will give all the requirements (functional and non-functional) explained in detail. All the possible scenarios that may happen within the program will be given in the 4th chapter.

The technologies that will be used to make the software work and other topics will be described in more detail later in this document.

Purpose and Scope of this Specification

The purpose of this document is to collect, analyze and outline all the ideas and the requirements for the system software that will help IUTE Credit in their management. Also, we will try to figure out or predict how we wish that this product can be used, all the ideas that will be considered but not completed as the product develops, and finally give a general overview of the concepts that can be developed later.

More shortly, the main purpose of this document is to provide a detailed overview, with all the parameters and goals for our web application. In this document will be described the target audience of the project, followed by software and hardware requirements and its user interface. Also, is defined how the product is seen by the perspective of the audience and client.

The intended audience of this software system is the stakeholder and the developer of the system.

This software system will be a web application, which not only give the information to the client but also will function as an EPR of the IUTE credit Albania. This web application is constructed to increase the efficiency, effectiveness and productivity of the IUTE credit company. The system software will meet all the needs of user, admin (and other type of users) while in the same time it will be easy to use and understand.

Product/Service Description

The second section of the Software requirements specification will give an introduction on the idea of the project, reasons why this idea was chosen, a general description what this product will be and what it will do, to whom it serves most, reasons that the stakeholders choose to use the program versus the mechanical work.

Further on, it contains the list of the user, including their characteristics. A detailed product functions of IUTE with all the constrains and assumptions will be provided.

Product Context

IUTE credit company, located in the Center of Tirana, functions as a financing institution which gives small loans to the clients. These types of loans can be given within 30 minutes and the client must have only it's ID and a job verification, which contains also his salary.

IUTE credit system software will be a web application which will be created to make the process of decision making easier. It will help the workers and the managers of IUTE credit Albania to be more secure when making decisions (all the stakeholders of the product will be discussed later). This web application will help not only the workers of IUTE credit but also the clients, by providing them a real-time information for the loan.

Also, clients have the possibility to check directly the amount that they can borrow, by providing to the application their data, like salary, time they think they will pay the loan etc. All user communicates with each other through messages (an option shown in the menu).

This web application will be built using PHP, MySQL, Apache, symphony framework. The web application will be accessible from all types of browsers and from mobile devices. The most basic requirement of the application to run is having a web browser and an active internet connection.

User Characteristics

Software loses its value when it's not used from others. So, to get the maximum from the efforts done building the product we will focus our work in making the product: useful, useable, stable and less efforts will be given to speed and making it pleasant. By following this rule, we try to not only get the best from the efforts, but also, we provide to the user more satisfaction while using the web application.

The general users that the IUTE credit Albania web application is planning to have are:

- Manager- which will be the admin, and will have access to all the information displayed to other users. He can view, edit, approve, delete or add other users. Also, he can check the financial statements, send messages to the employees etc. More features for the admin will be added later, while the product will be developed.

- Workers- a contracted employee who can view loans given to clients (more tasks will be added later). The employee can get messages from the admin. While a client comes to the offices of IUTE credit Albania, and interested to pay or get a loan, the employee will do this transaction with the web application. The available amount of money available is shown under the HOME option.

- Clients- everyone who wants or is interested in getting a loan can be registered and can check information about company and the loans that he/she can take. If the user has got a loan from the IUTE credit, he can also check in real time, the remaining amount that must pay to IUTE Credit Company, and the deadlines for each payment. The web application will give alerts to the user if the deadline for making the payment is near.

- Businesses – every company that wants to cooperate with IUTEcredit. This means that all the companies that are often at the retail sector provide to their customers the possibility to buy thing with monthly payments by applying for a small loan at IUTEcredit.

Assumptions

• Since the program is user friendly it is assumed that every user will have it easy to use it.

• Since it is a web application it is assumed that the computer when it will be open has a stable internet connection.

• Since it is a web application it is assumed that the user has at least a web browser in their computer.

• It is assumed that everything will go as planned and all the requirements specified in the SRS will be provided.

Constraints

Constrains exist because of real business conditions. An example may be: not having an available amount to give to a customer, or even the legislation or the risk.

The biggest constrain of the web application will be considered internet connection or any problem with electricity. Every single data of web application is stored in a database, and to access it every user needs the internet connection.

Since technology is increasing a constrain of our application can be the hacking, because even if we try to keep it safe, new hacking technics are being discovered.

Dependencies

The web application is independent from any other program or software that is used by the IUTE credit company. To run this application there should be an internet connection and the computer must have at least one web browser.

Apportioning of requirements

The time available to finish this project is only 14 weeks some requirements may not be completed.

Requirements

This section will provide all the functional and non-functional requirements of the system. A detailed description of all the features of the application will be covered below.

Functional Requirements

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
R_01	User registration	Every user that is in website of IUTE credit can register as a simple user, by providing their username, email and password		28 March	4-11 April
R_02	User log in	Every user that has been registered before will be able to log in with their confidential information. The log in information can be stored in their browser and in the future this user can be automatically logged in.		28 March	4-11 April
R_03	Log In failure	If the information that entered by the user does not exist in the database, the user will failure to log in into their account.		28 March	4-11 April

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
R_04	Log Out	All types of user can log out from the system whenever they want.		28 March	4-11 April
R_05	User role check	After putting the right credentials, the user will log in and the system will check the role of the user form database. After this a user interface will be created, depending by the role that the user has.		28 March	4-11 April
R_06	Edit	All users, whatever the role of them, can edit their personal information like: name, surname, password etc.		28 March	4-11 April
R_07	Display	All user can display information form the database. For example, the client can display the personal information and the information for his loan, the date of the payments that he has done etc. The employee will display the personal information and the information about all the loans of the company.		28 March	4-11 April

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
R_08	Update authentication	Only the admin will have this feature. By using this feature the admin can change the role of the user. For example, an employee gets promoted and becomes a manager.		28 March	4-11 April
R_09	Search	Admin can do some types of search. First one is related with the users. Admin can search the users and update their information, or can search for the loans. A user with the role of an employee can search only for the loans.		28 March	4-11 April
R_10	Add a new employee	The user with the role of admin can add new employees. The admin will have to put all the personal information of the employee. All the employees will have an ID.		28 March	4-11 April

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
R_11	Generate financial statements	The user with the role of admin will be able to generate the financial statements like: income statement, cash flow or balance sheet.		28 March	4-11 April
R_12	Check the balance	The user with the role admin will be able to check the amount of money that the company has available and can give as a loan to different clients.		28 March	4-11 April
R_13	Get and send messages	The user will be able to communicate with each other by messages. The admin can send messages to all the users, whatever role they have. The employee, the same as admin can send messages to all the users. A simple user, with the role of client or business can send messages to the employees.		28 March	4-11 April

User Interface Requirements

First thing that a user will see when opens the page is the login page. To enter to their account the user must provide his credentials. Every type of user will have their first page different from the other user.

If not registered before, the new user will have the possibility to registered by clicking the Register button, and then providing the information that's required.

After registered or logged in, every type of user will have a page where they can edit their information (like name, surname, email etc.). Also, all users have the option to choose the language they want (English or Albanian).

The interface of the Bits Please project will be user friendly.

The characteristics from human point of view:

- This application it's runnable in most browsers (Chrome, Firefox, Internet Explorer). The user only needs to have knowledge in how operate a web browser, be familiar with the computer, keyboard and mouse.

The characteristics between the web application and the hardware components:

- The hardware components will be accessed in an indirect way. From the client-side it will be accessed via a web browser, and from server side it will accessed via programs as APIs (for example PHP, Apache etc.).

Non- functional requirements

Usability

- The web application will be very easy to learn and to use.
- The web application will come with instructions that will make easier the use of the system.
- A video tutorial will be created to show how everything works.
- The look and the feel between all the webpages will be uniform so the user can achieve effective and maximal performance.
- Multithreading the system can support the interaction of the user in more than one task.
- Task completeness.
- Error messages will be displayed in natural language.

Performance

• The software is going to be web based, so this software requires a server machine with high band internet access. In a way of handling more users at the same time the CPU of the server machine needs to be powerful and the speed of internet access needs to be high.

• Another requirement related with performance is the storage space. If the storage space is higher this means that more users can be registered and every user will have a bigger workspace. So, the higher is the storage of the software the higher is the performance.

• The number of users that access the page simultaneous is unlimited.

• All the data, like data for loans or users, are obtained from the database, so the time for the response of the query shouldn't be more than 5 seconds.

• To handle all the run time errors in the program, the error handling needs to be implemented.

• The web application should be flexible for future research analyses questions.

• The performance depends in the hardware of the user.

• The performance depends on the internet connection strength.

Capacity

• The capacity of the web application is enough to support all the daily operations of the company.

• It has unlimited number of users but it has a limited number of workers and managers that will use this application.

• The capacity of this web application will be limited to the amount of money that the company has available for loans.

Availability

• The web application will be available 24 hours 7 days a week.

• It has no geographical limitation for the online applications.

• It reduces the process time so the clients can be more satisfied.

• The maintenance for this web application is required for the database and for possible updates in the future.

• This web application is very reliable.

Latency

The maximum acceptable time for an application to be approved is 20 min if the application is done in the office and 5 min if the application for the loan is done online.

Protection

• The web application will be secured, so it can keep the client information secured. Users will log in only with their credentials and will have only one session. They cannot log in to other user's sessions.

• It will be used md5 encryption. The admin cannot see the password of the simple user, so in other word the admin will not have the chance to decrypt the simple user information.

• The system will terminate when the period of inactivity of the user is long.

• The information of the database will be backed up.

Authorization and Authentication

The users which are connected to internet will be able to get into the system. To access the interface and get into their account, every user must put his credentials which are authenticated.

User account will be authorized. Both these functions are implemented by the head of the function of Authentication and Authorization.

Manageability/Maintainability

Monitoring

When a user is logged in with different credentials than the one that registered, then in the screen it will be displayed that the password or the username is not correct. The program will ask for its email or phone number and will send a confirmation code to change the password.

Maintenance

• Product updates- if there is any feature that the business find the need to have it in the web application there will be updates. Updates will also bring more reliability and speed to the app. Also, the update is directly related with the improvement done on security.

• Test website speed- testing download speed. This is needed especially in downloading the financial statement.

• Back up- before any major changes the page will be backed up.

Operations

This web application has many operations but most important ones are:

- Add new client
- Add new Employee
- Add business
- Check available amount for loan
- Check loans to be paid
- Edit user information
- Give loan
- Pay loan
- Generate income statement
- Generate balance sheet
- Generate cash flow statement

System Interface/Integration

Network and Hardware Interfaces

The characteristics between the web application and the hardware components:

• The hardware components will be accessed in an indirect way. From the client-side it will be accessed via a web browser, and from server side it will access via programs as APIs (for example PHP, Apache etc.).

Systems Interfaces

Related with the system interface we can say that:

• The system may use the protocols of HTTP and HTTPS when needed for transfer of publishing.

• The system, if necessary, will use the SQL and technologies of database for backup, user information.

• The system may use different networking protocols like UDP, TCP/IP or others.

Portability

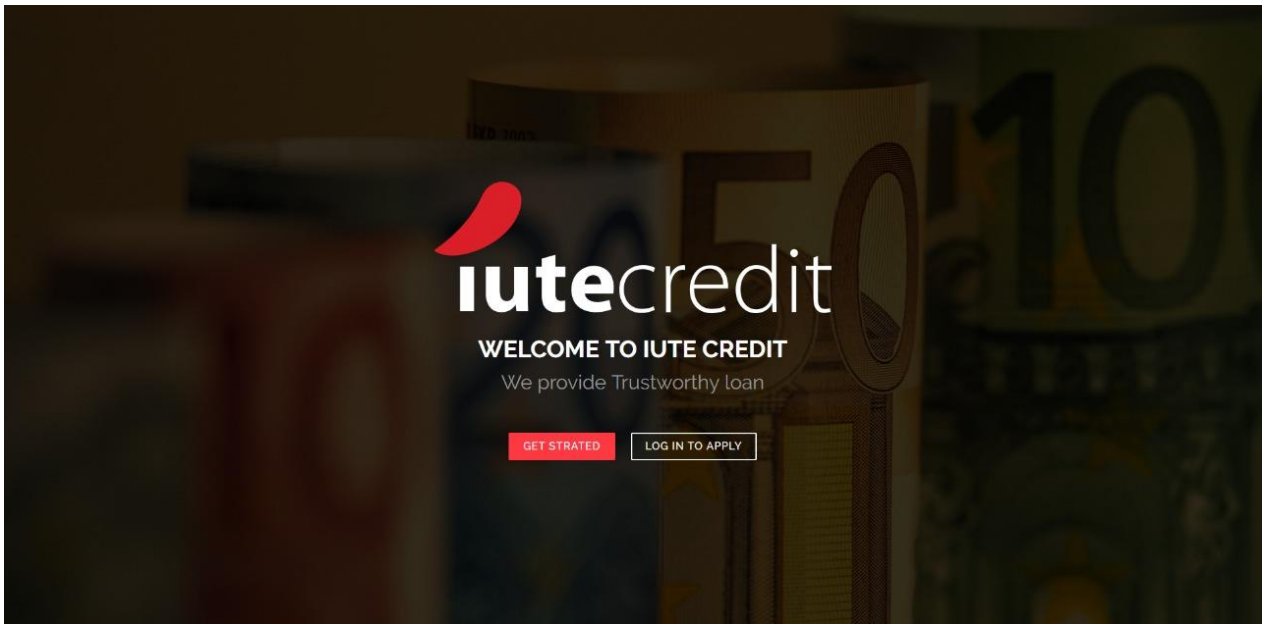
• The web application will be very simple to use so it doesn't need powerful computers but it only needs a simple computer or a phone to be accessible.

• The software will run in different platforms, and with different internet speeds.

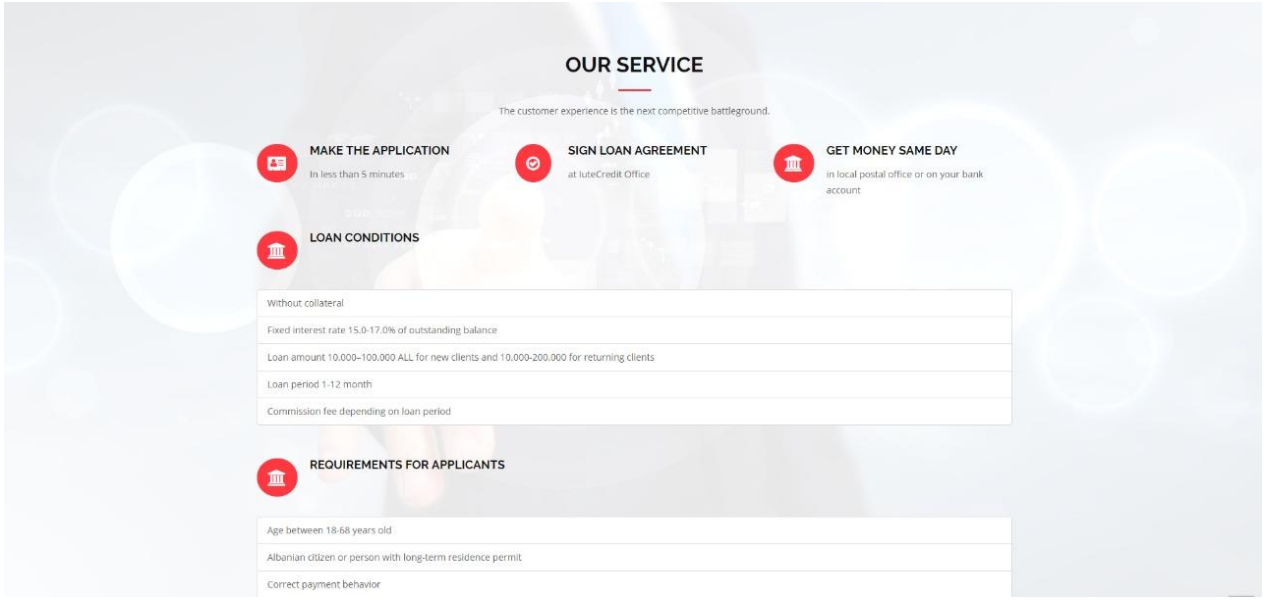
• The software it will be lightweight so it can load faster even if the internet connection it's not very high. To be a more lightweight application there will be used different libraries and tools like JavaScript.

INTERFACE

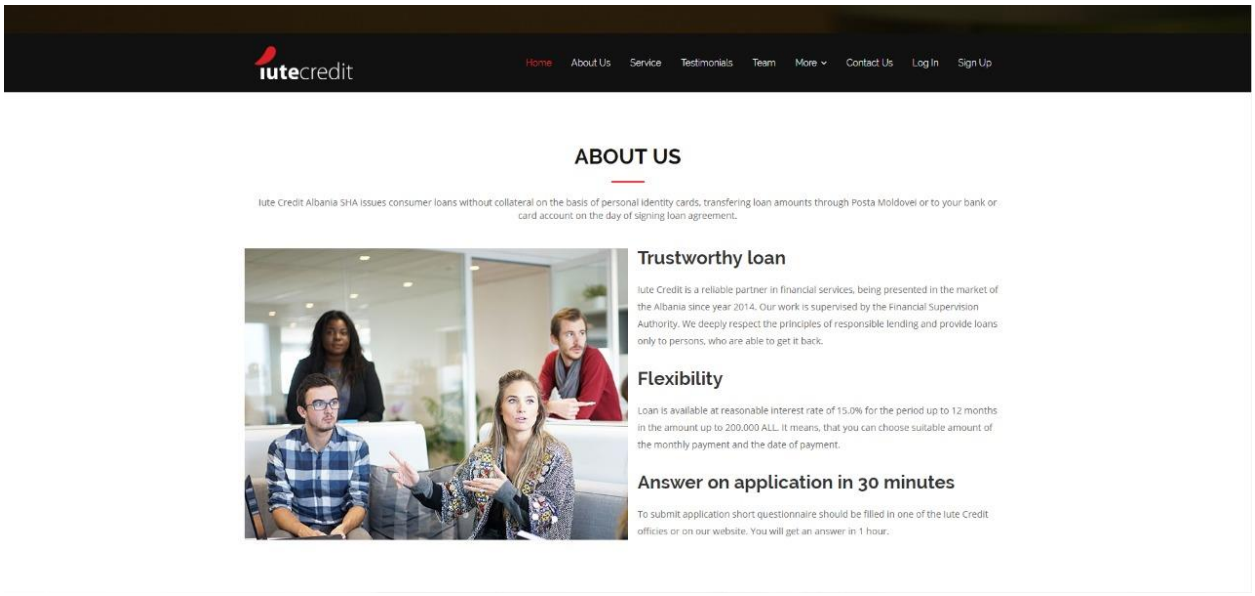
LOG IN PAGE



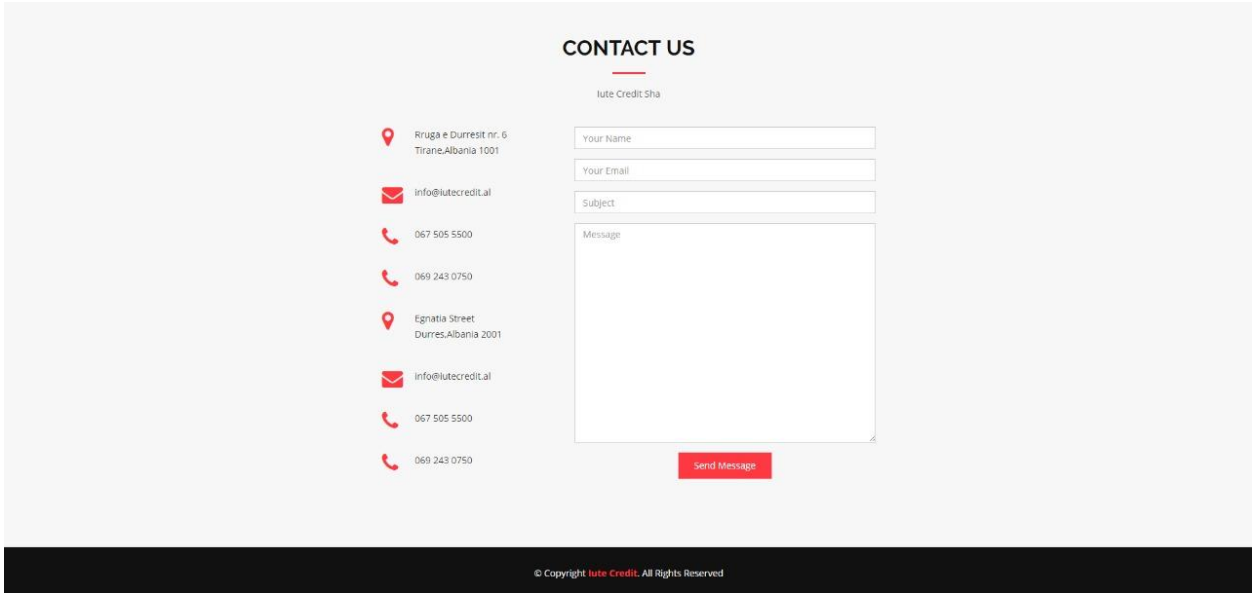
OUR SERVICES



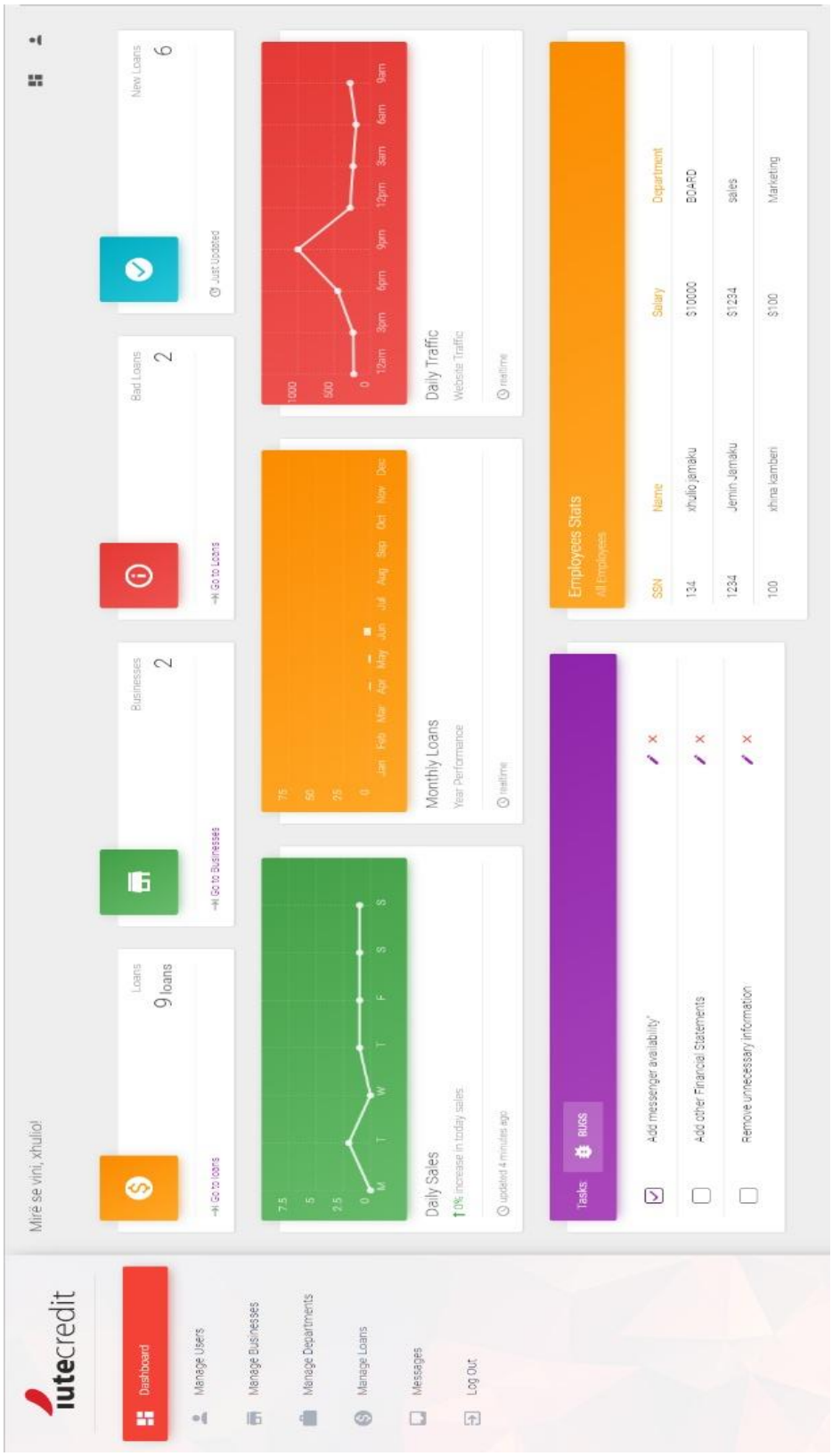
ABOUT US



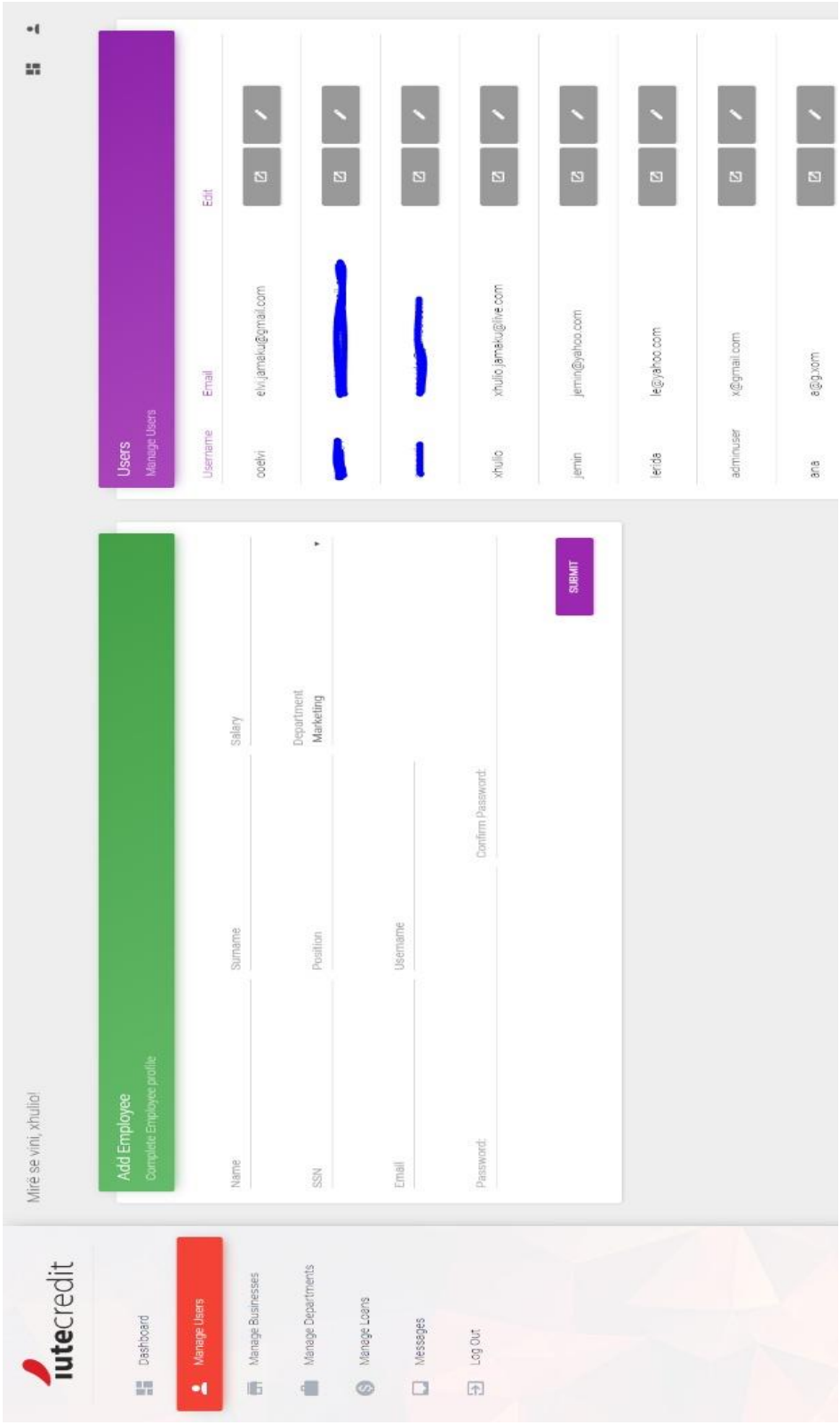
CONTACT US



ADMIN DASHBOARD

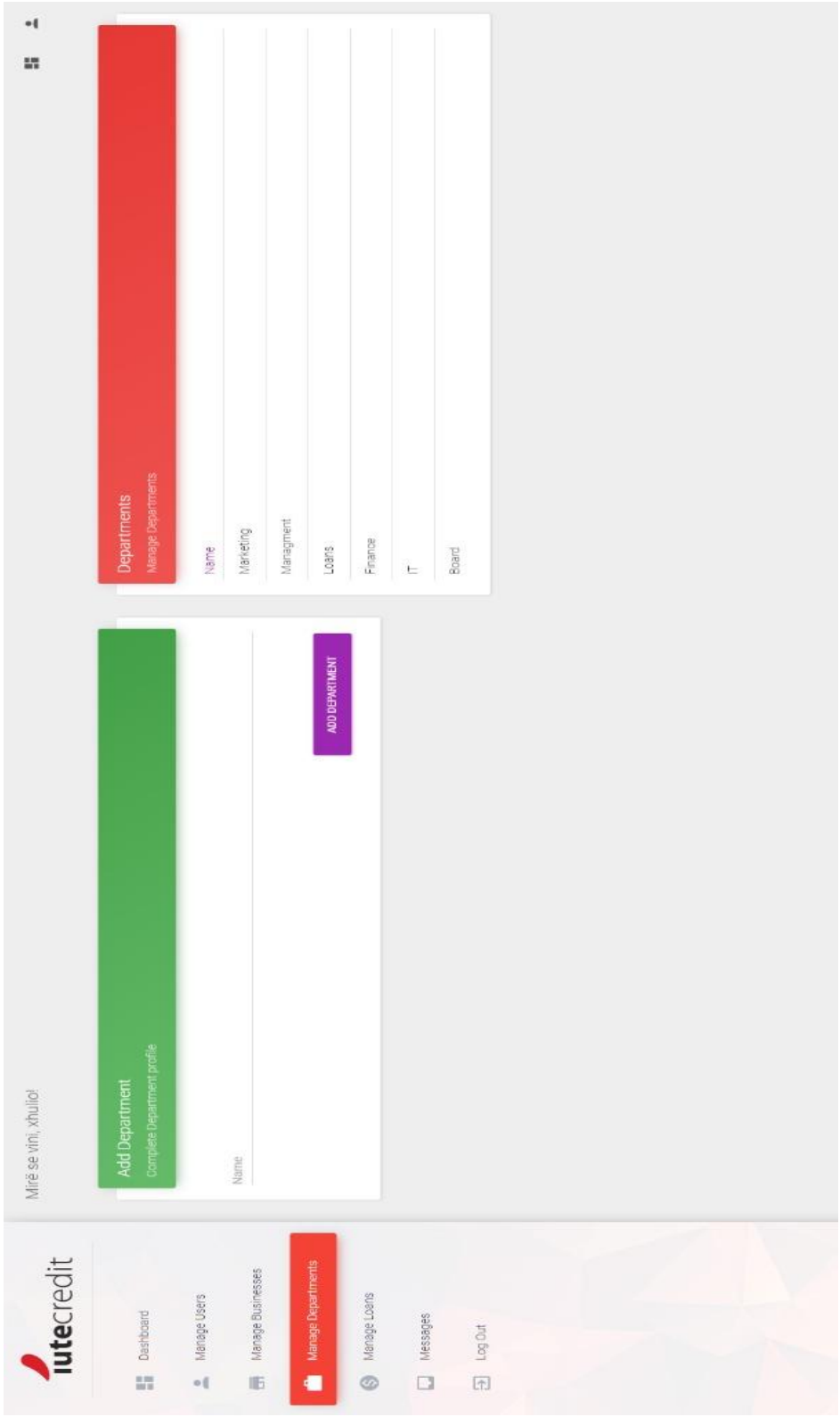


ADMIN- MANAGE USERS PAGE

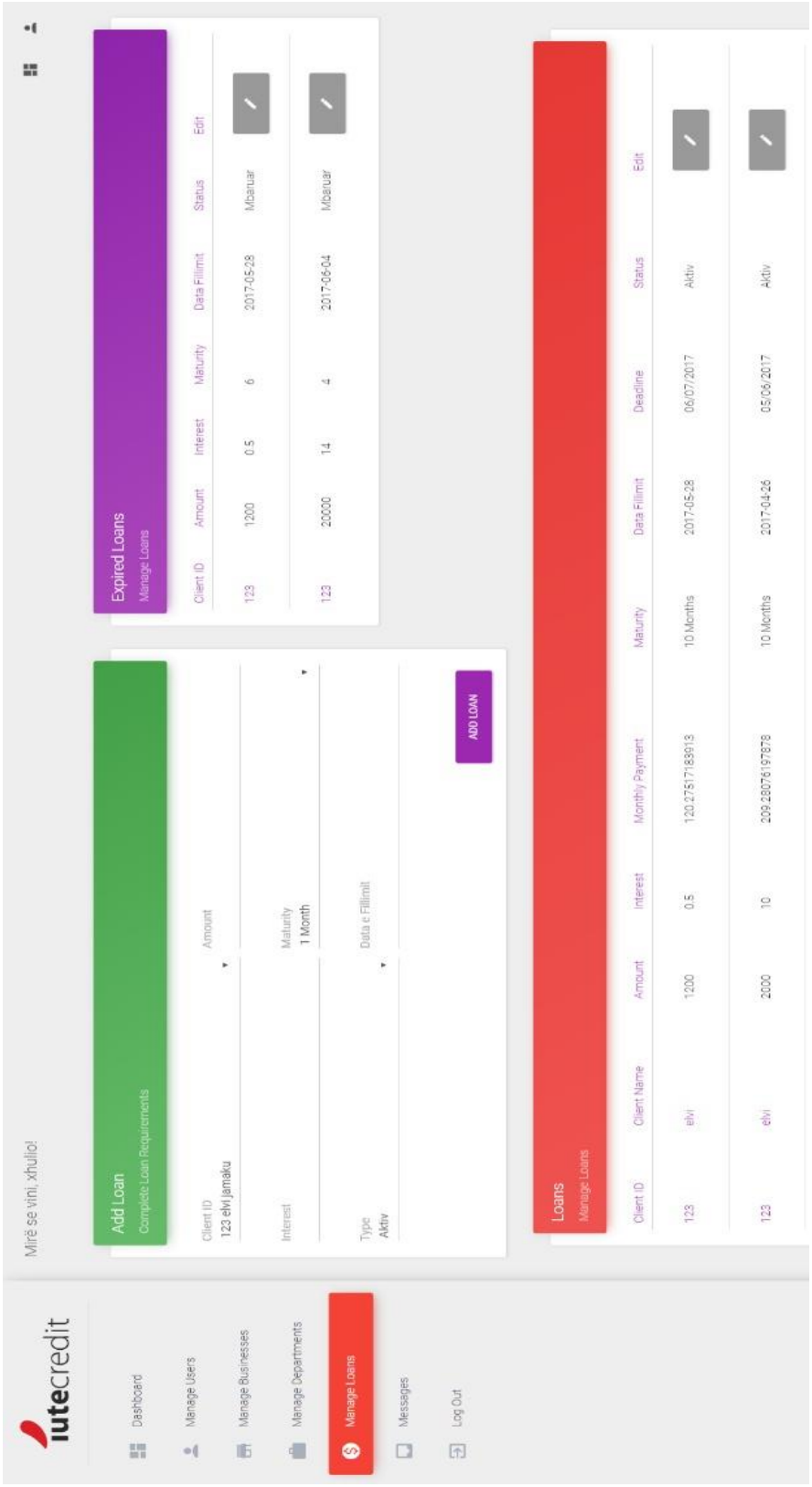


ADMIN – MANAGE BUSINESSES PAGE

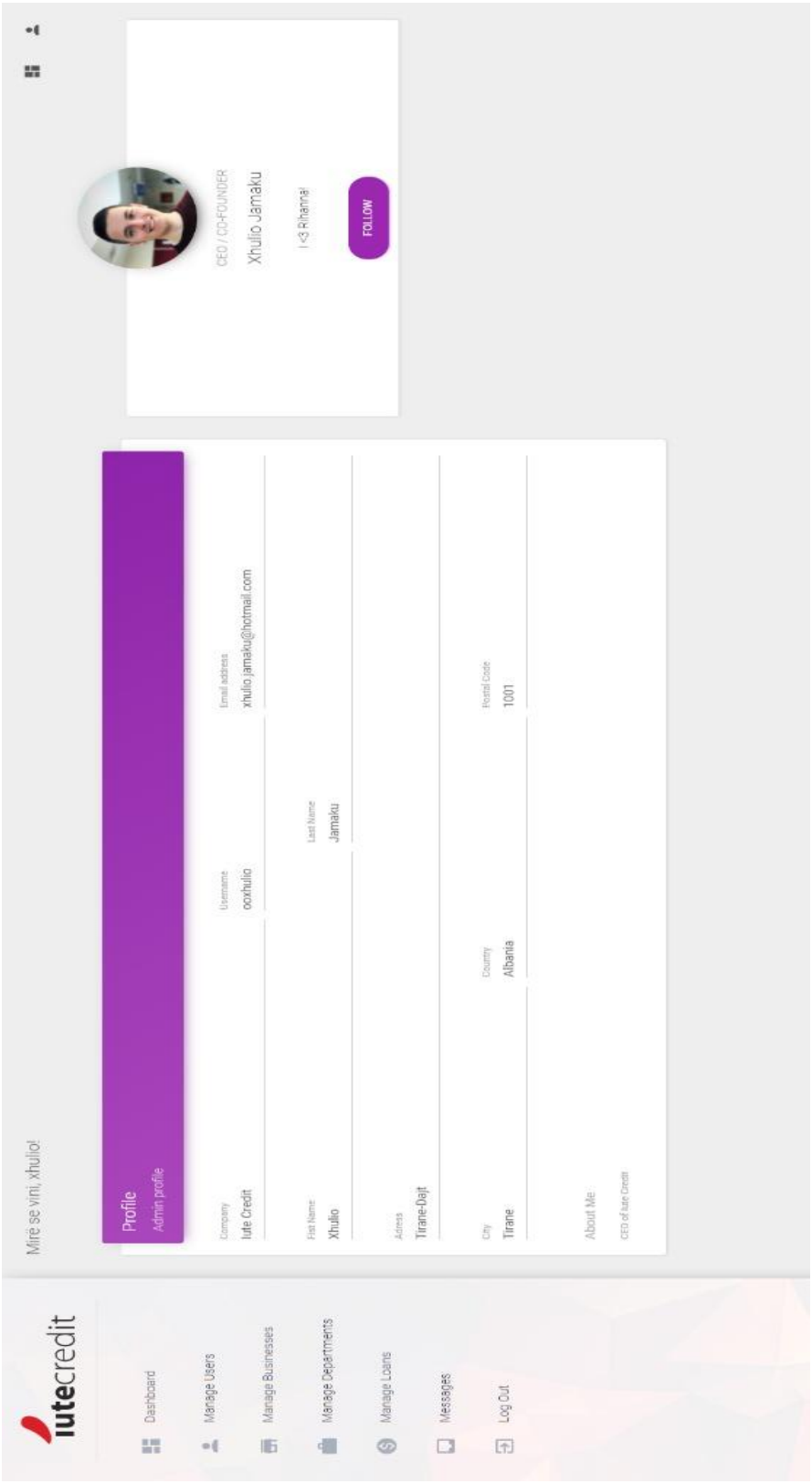
ADMIN- MANAGE DEPARTMENT PAGE



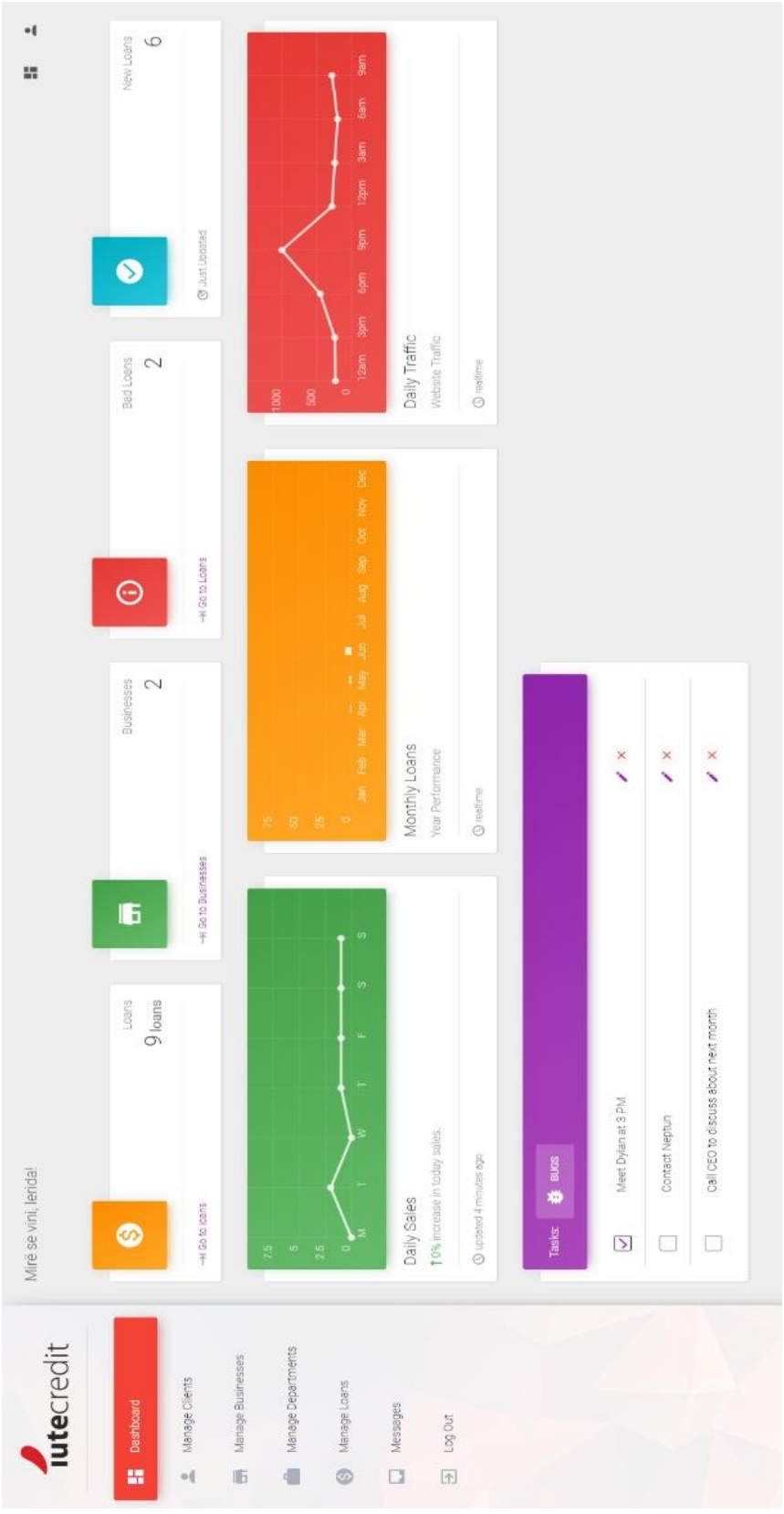
ADMIN-MANAGE LOANS PAGE



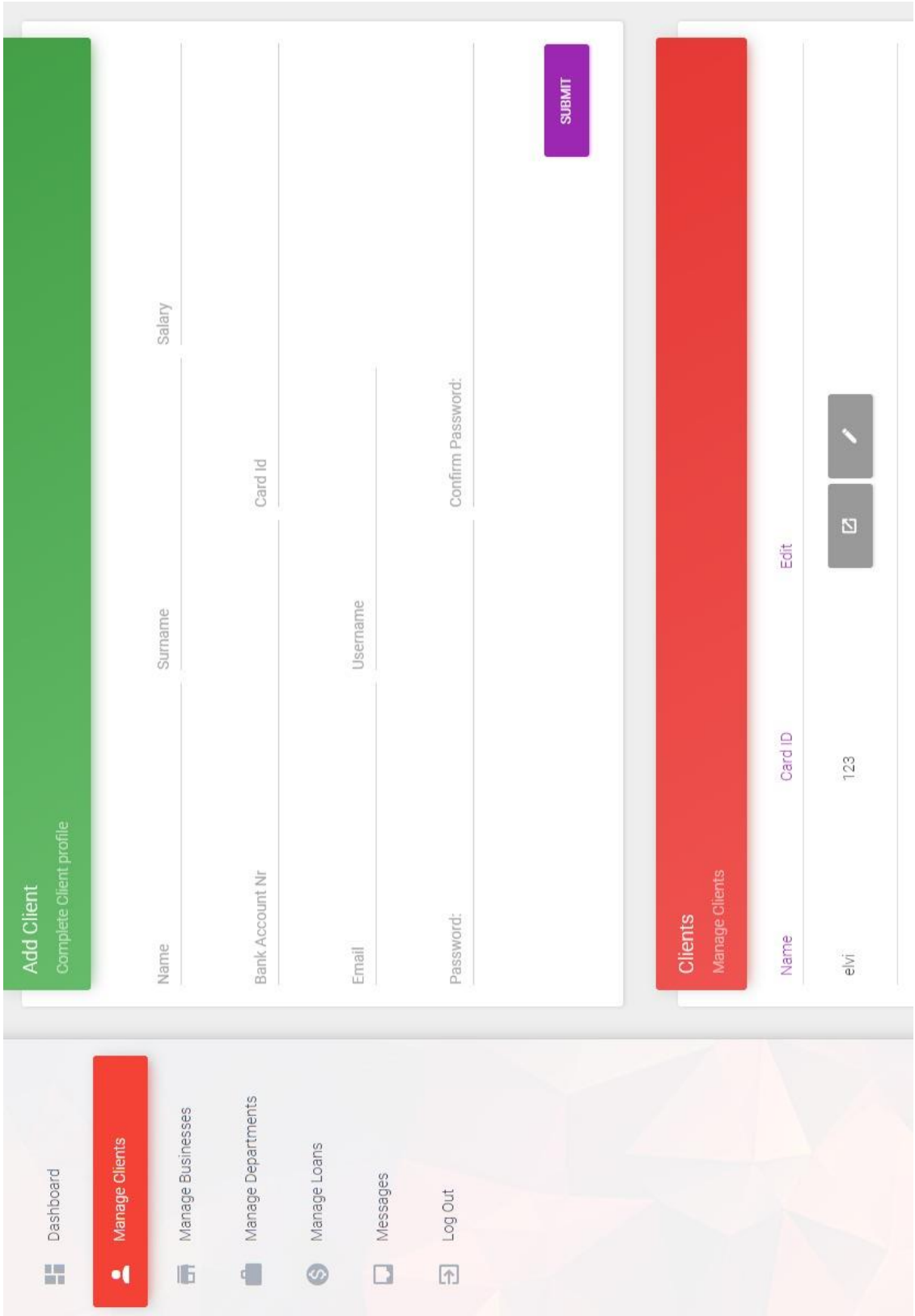
ADMIN- EDIT PROFILE



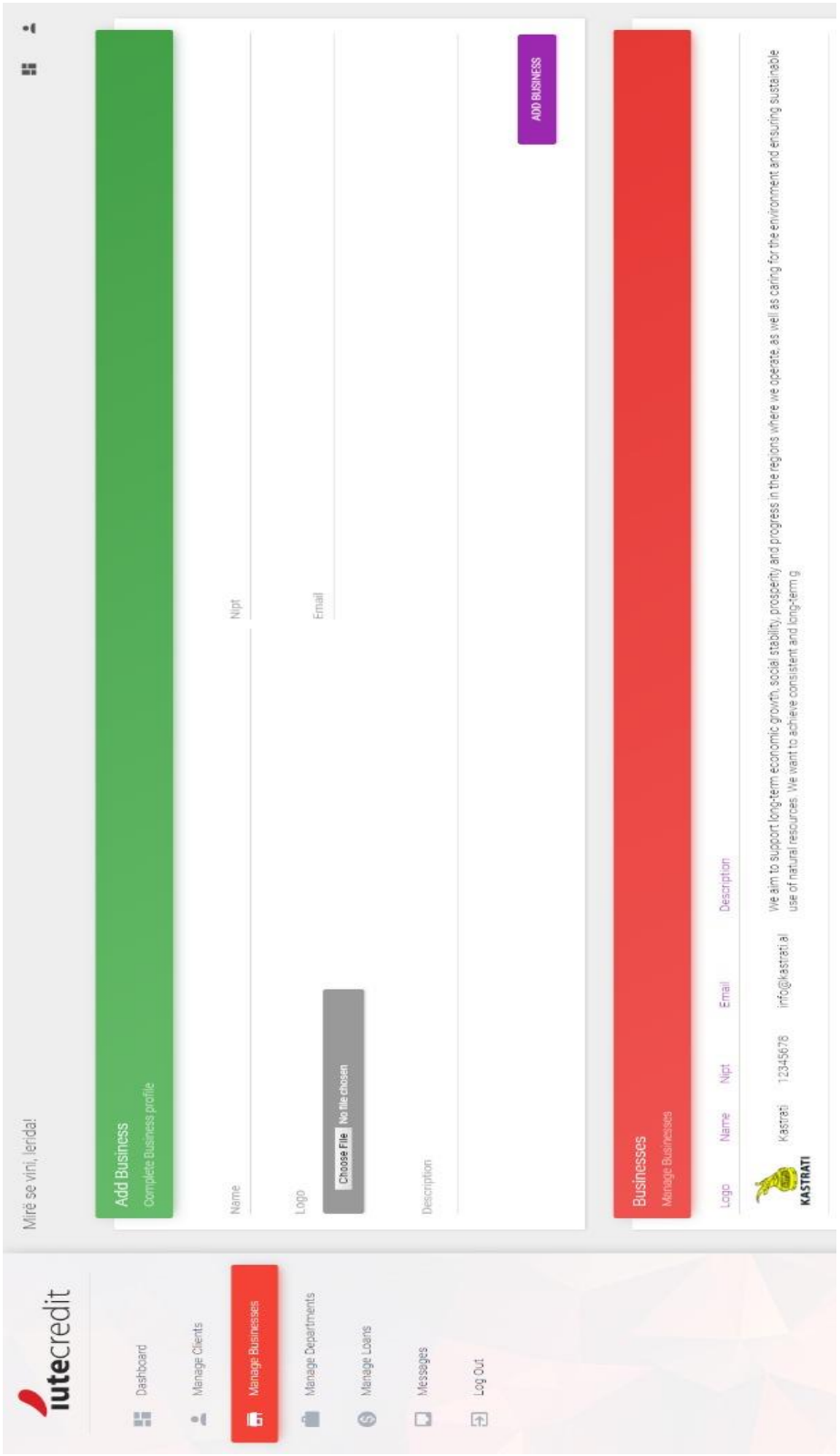
MANAGER- DASHBOARD



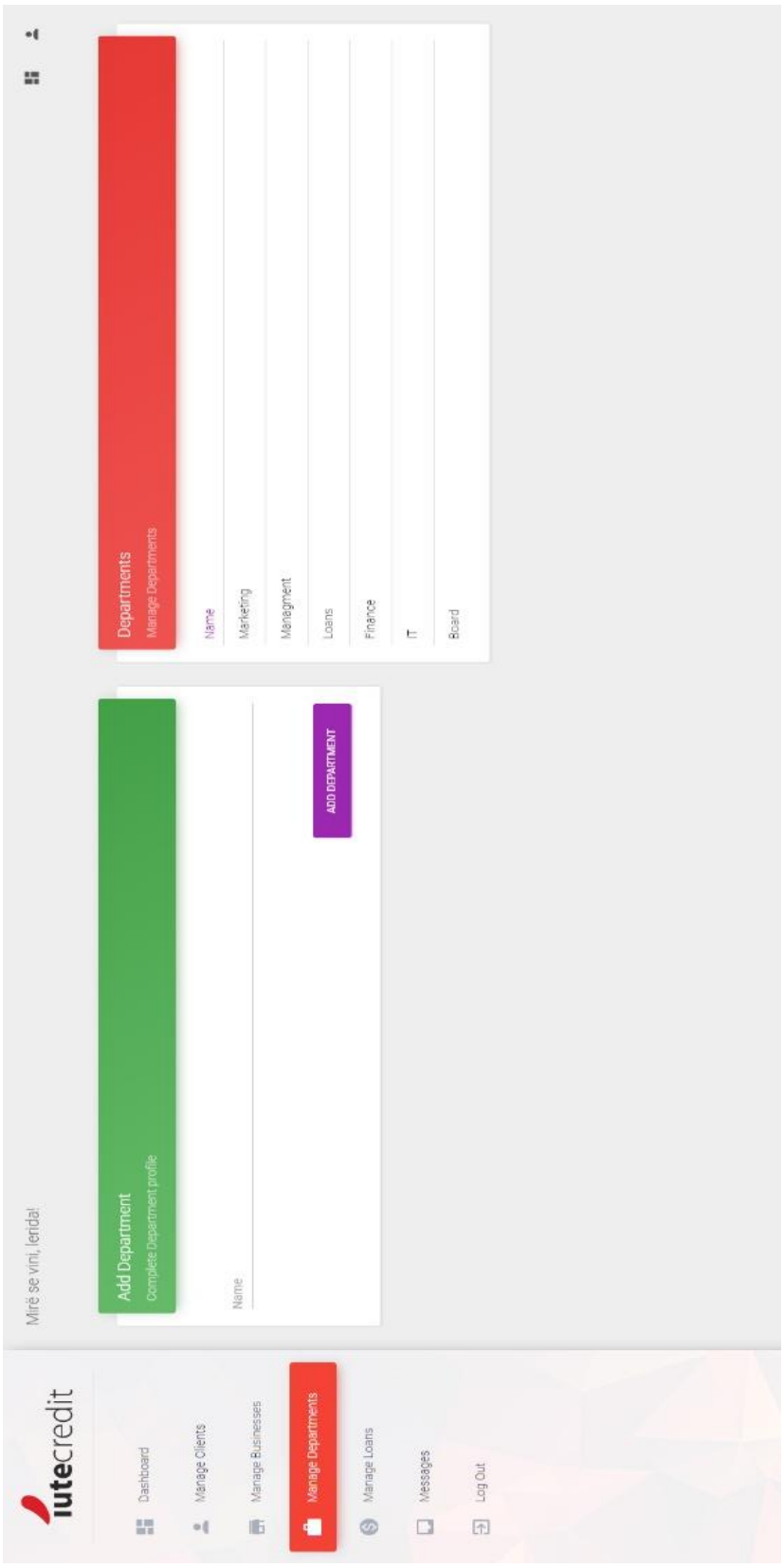
MANAGER- MANAGE CLIENTS PAGE



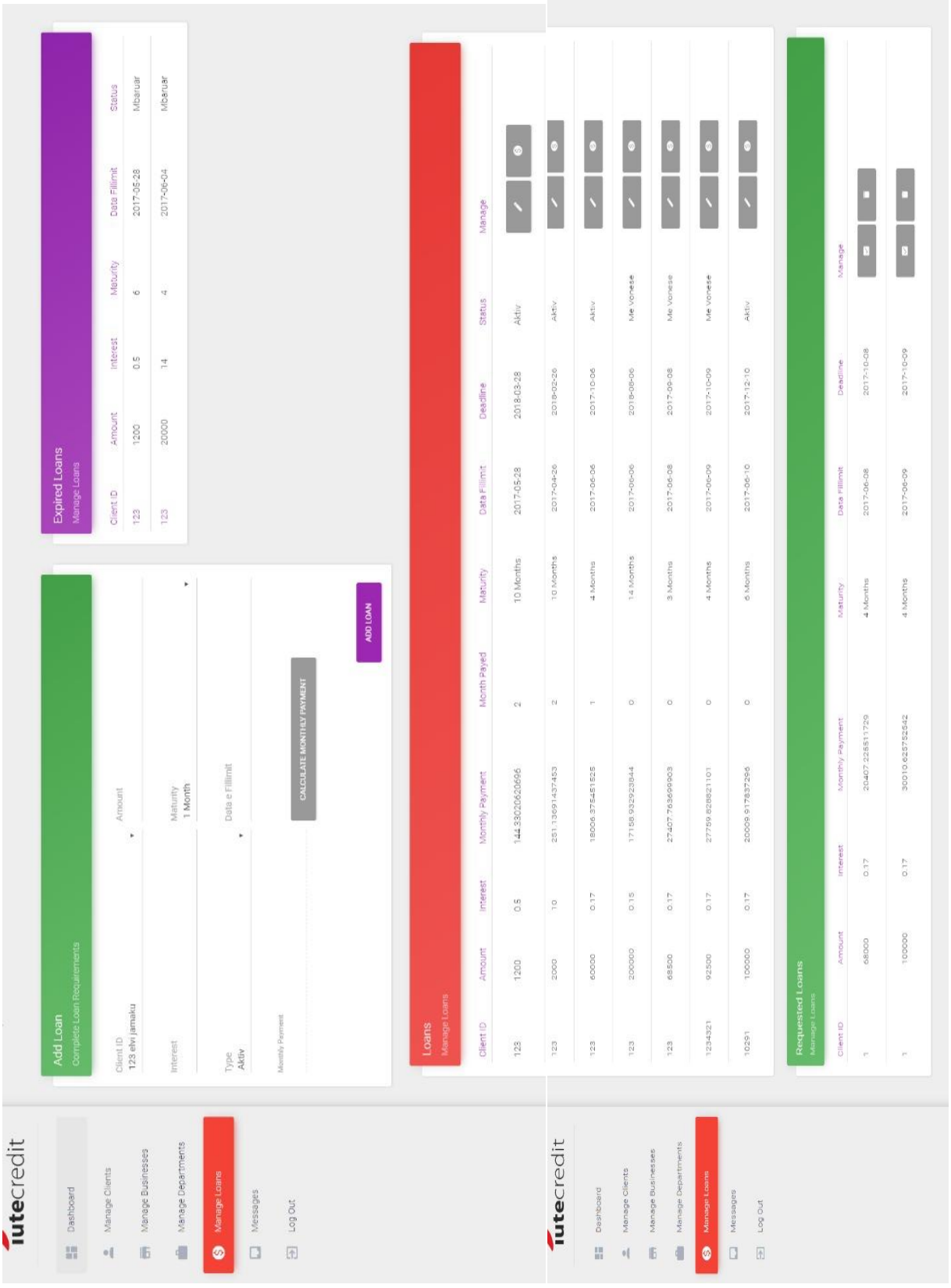
MANAGER- MANAGE BUSINESSES PAGE



MANAGER- MANAGE DEPARTMENTS PAGE



MANAGER-MANAGE LOANS PAGE



MANAGER- LOAN PAYMENT INVOICE

I N V O I C E

Xhulio Jamaku
Rruga e Durresit nr. 6 (near Vatican embassy)
Working hours:
Mo-Sat 08:00–20:30

Phone: +355 4 223 9111
Mobile : 069 243 0750
E-mail: info@iutecredit.al

iutecredit

Elvi Jamaku
Card Id: 123

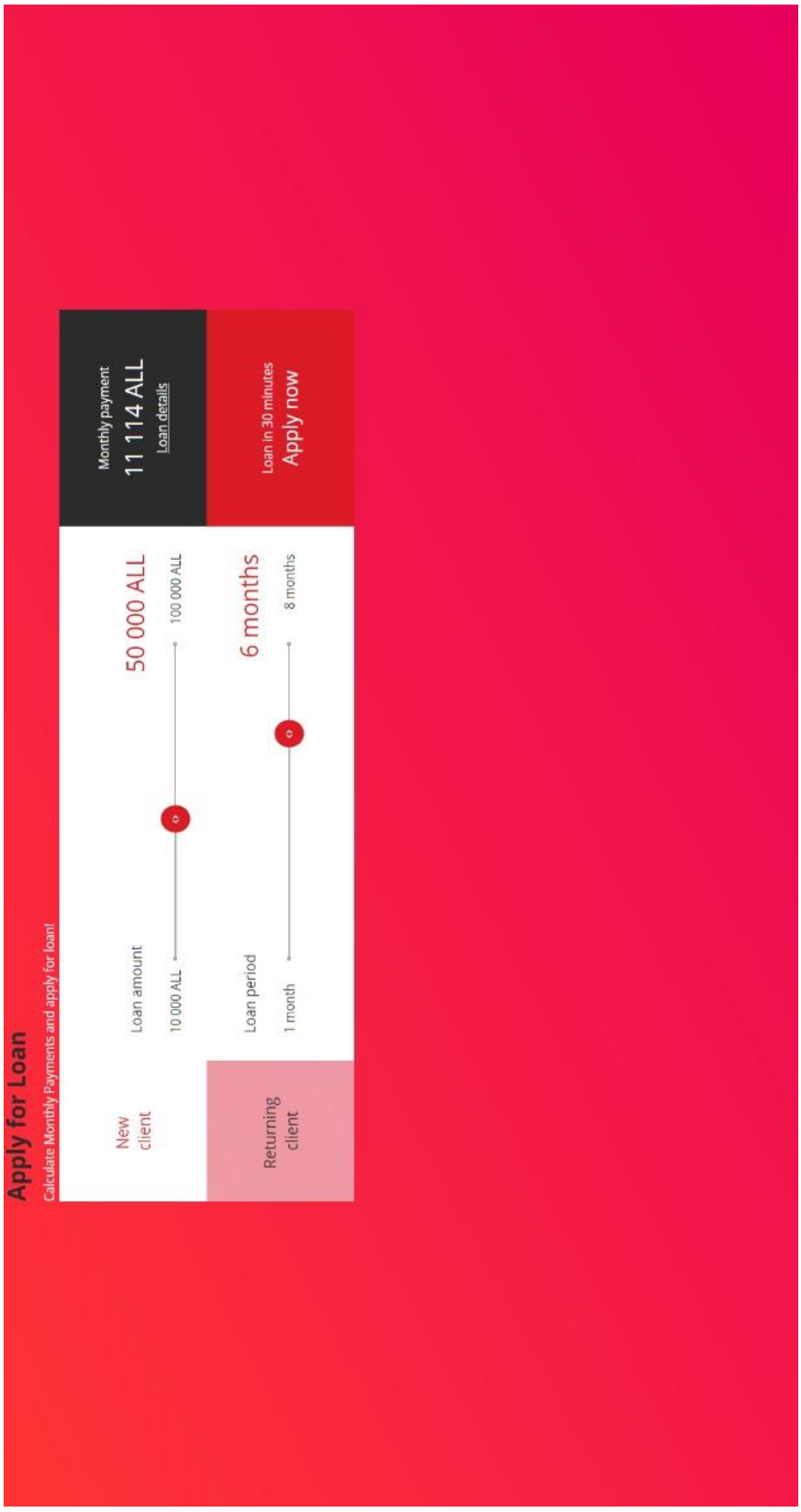
Invoice #	524624
Date	06/10/2017
Amount Payed	120.27517183913

Item	Description	Unit Cost	Quantity	Price
Monthly Payment	Monthly payment for loan	144.33020620696	1	144.33020620696
			Subtotal	144.33020620696
			Total	144.33020620696
			Amount Paid	144.33020620696
			Balance Due	1010.3114434487

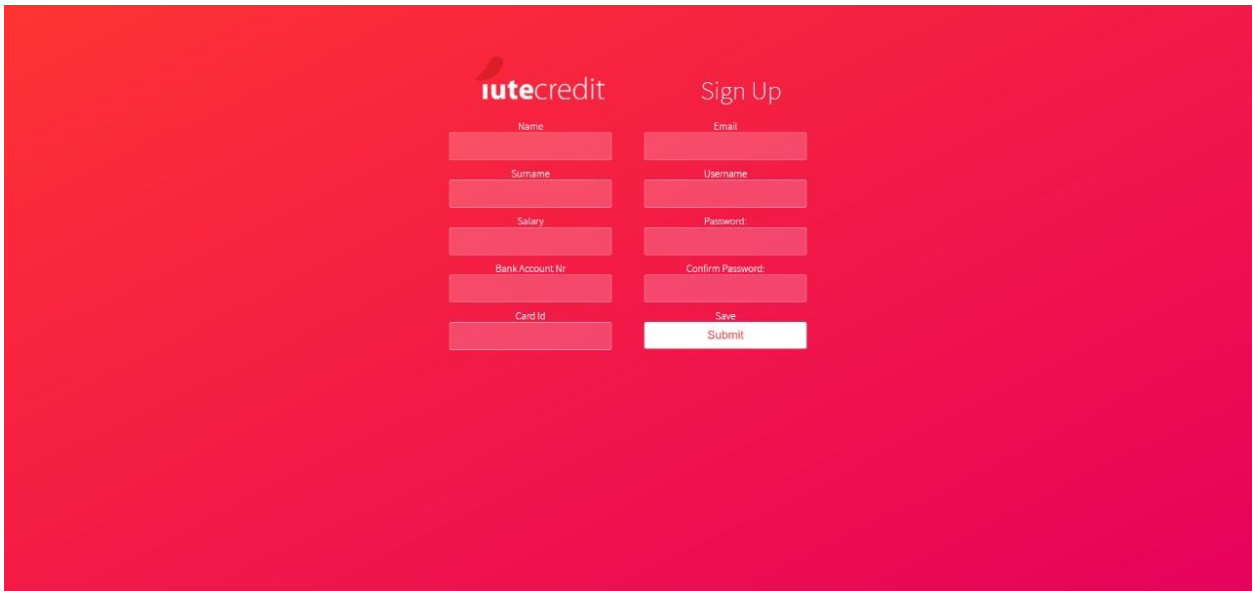
T E R M S

NET 30 Days. Finance Charge of 1.5% will be made on unpaid balances after 30 days.

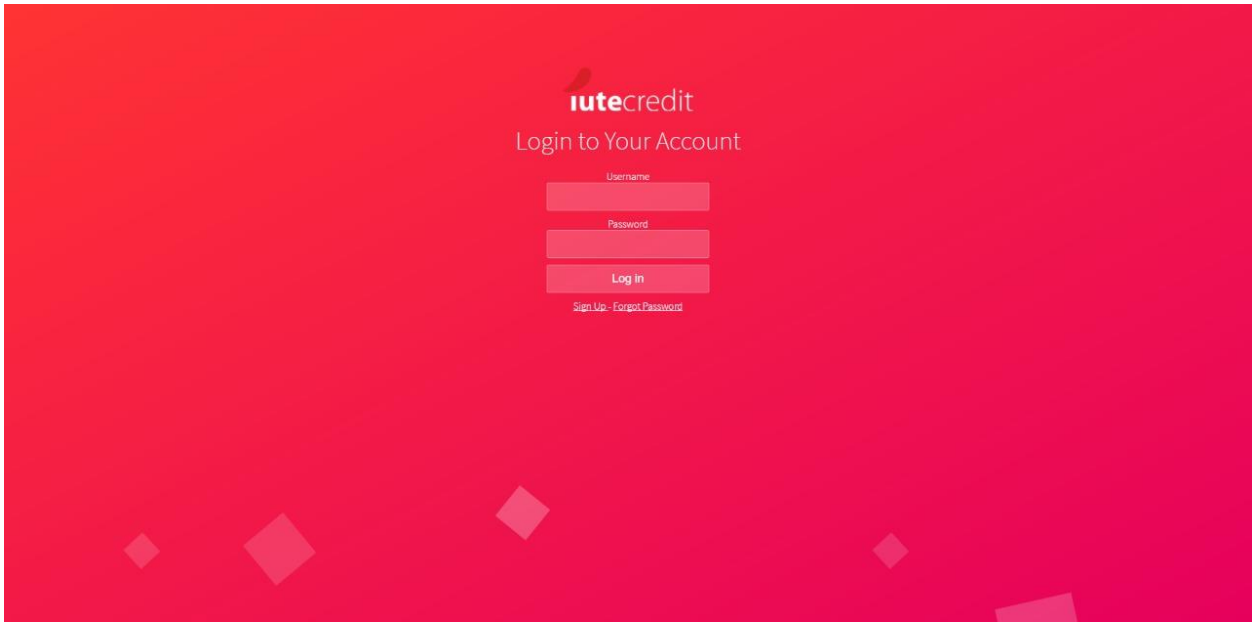
CLIENT- APPLY FOR LOAN



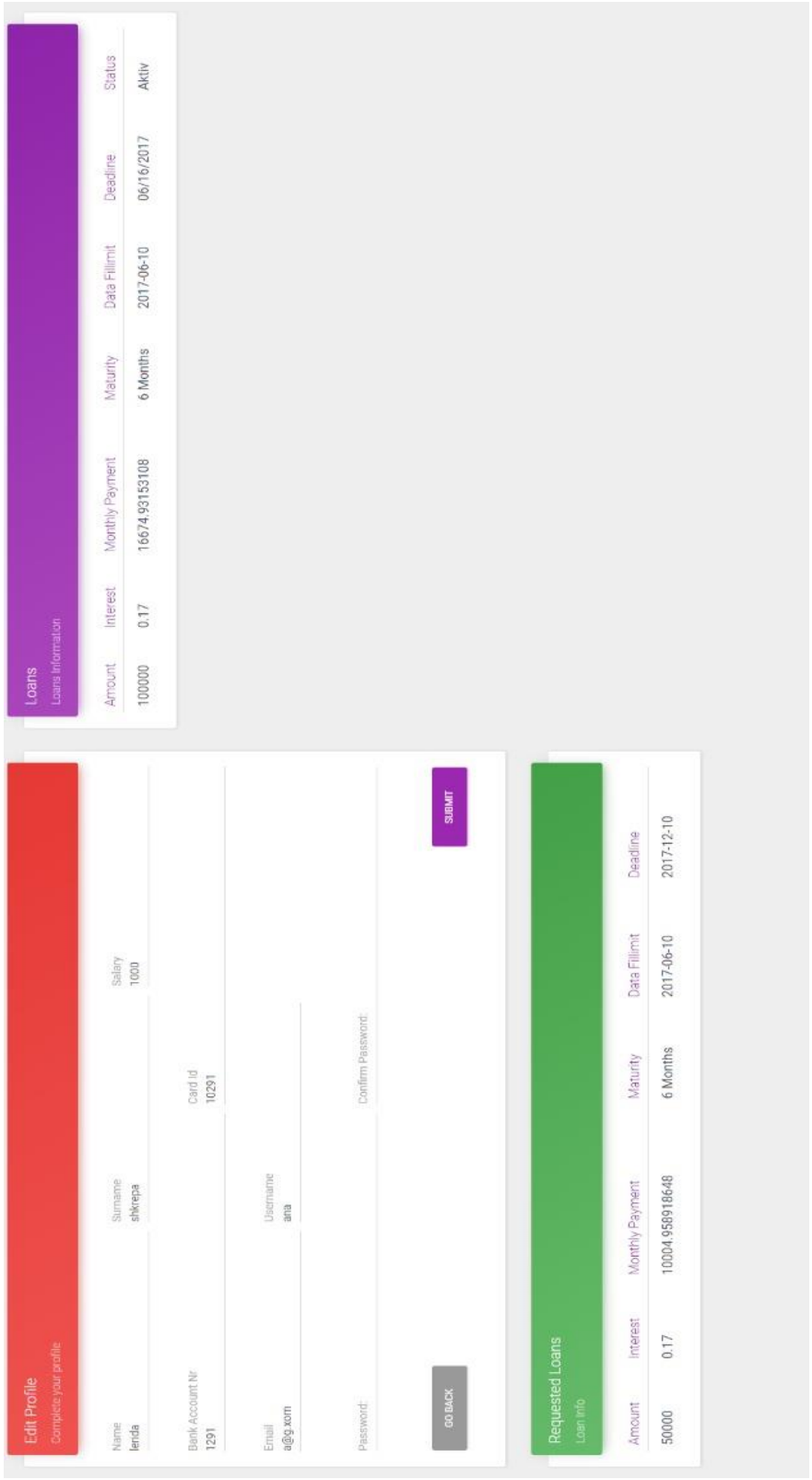
CLIENT- SIGN UP



CLIENT –LOG IN



CLIENT- DASHBOARD



This page intentionally left blank

User Scenarios/Use Cases

This section provides all the use cases and the use scenarios for the web application. A detailed description will be given for all the possible scenarios that can be done within the web application.

The IUTE credit web application has four actors: admin, employee, user and businesses. Any user or employee communication with the system is through messages option.

User Stories

Nr.	User Story Names	Description
1	Admins logs in	Only the admins can log in the system. They will be registered with a username and password, to have access in the program.
2	Register client	The admin can add the clients who apply for a loan. They must give their personal information. A record will be kept for them
3	Register employees	The admin can register employees along with their personal information. A record will be kept for them.
4	Send messages	The admin can send messages to both clients and employees depending on the situation.
5	Delete employees, clients	The admin can delete former employees or clients. However, their information will still exist in a special folder.
6	Edit employees, clients	The admin can edit updated information about either employees or clients.
7	Generate balance sheet, income and cash flow statement	The admin can generate the three most important financial statements for any business. They can be generated monthly, semi-annually or annually depending on the requirement.
8	Approves/Denies loans	The admin has the responsibility of approving or denying loans depending on whether the client it creditworthy or no.
9	Search by name	The admin can search the users by name to ease the search process of a program with many user.

10	Generate report	The admin can generate reports of loan approval/non-approval which will also contain information about the time and date, client name and the type of loan approved/non-approved.
11	Print income statement	The admin can print the income statement to have a hard copy saved.
12	Print Balance sheet	The admin can print the balance sheet statement to have a hard copy saved.
13	Print cash flow	The admin can print the cash flow statement to have a hard copy saved
14	Search employee	The admin can search the employee by name or surname.
15	Search client	The admin and employee can search the client by name, surname or loan id.
16	Search business	The admin can search the business by name.
17	Log out	Admin can log out after finishing with the program.

Scenarios

Scenario 1

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User adds a New Employee
6. User adds employee info
7. The changes are saved
8. List of employees is updated
9. Log out

Scenario 2

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Employee
5. User accesses Client list
6. User searches client by ID
7. User successfully finds client
8. User print client's invoice
9. Log out

Scenario 3

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as admin
5. User accesses Employee list
6. User searches employee by ID
7. User successfully finds employee
8. User checks employee information
9. Log out

Scenario 4

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Employee
5. User accesses Client list
6. User searches client by ID
7. User successfully finds client
8. User checks client information
9. Log out

Scenario 5

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User enters as admin
5. User checks balance sheet statement
6. User prints the statement
7. Log out

Scenario 6

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User enters as admin
5. User checks income statement
6. User prints the statement
7. Log out

Scenario 7

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User enters as admin

5. User checks cash flow statement
6. User prints statement
7. Log out

Scenario 8

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User adds new Client
6. User fills the Client information
7. The changes are saved
8. The list of clients is updated

Scenario 9

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as admin
5. Admin sends message to simple user
6. User receives the message successfully
7. Log out

Scenario 10

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User accesses Client List
6. User deletes Client
7. The list of clients is updated
8. Log out

Scenario 11

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User accesses Employee List
6. User deletes Employee
7. The list of employees is updated
8. Log out

Scenario 12

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Employee
5. User accesses client list
6. User find client
7. User checks if client is valid for a loan
8. User send client a message
9. Log out

Scenario 13

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Employee
5. User accesses loans list
6. User find delayed loans payment
7. User send alert to client
8. Log out

Scenario 14

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User checks available balance
5. Log out

Scenario 15

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Employee
5. User accesses client list
6. User finds client by ID/Name
7. User updates client info
8. The changes are saved
9. Log out

Scenario 16

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly

4. User is logged as Admin
5. User accesses Employee list
6. User finds Employee by ID/Name
7. User updates employee info
8. The changes are saved
9. Log out

Scenario 17

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User checks loans list
6. User find loan
7. User deletes loan
8. Log out

Scenario 18

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User checks loans list

6. User find loan
7. User updates loan information
8. The changes are saved
9. Log out

Scenario 19

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User is logged as Admin
5. User adds new loan
6. User fills loan information
7. Log out

Scenario 20

1. User enters username and password
2. System validates the entered information
3. Username and password are added correctly
4. User send message/feedback to the company
5. Log out

General Use cases

(UC_01) Use case: Login

Use case number	UC_01
Use case name	Login
Actors involved	User, admin, employee, businesses.
Description	This use case describes how a user logs into the web application. When the user finds the need to do something to their account (or just look for information) he enters its username and password and will log in in their account.
Precondition	None
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The password and the username entered are not correct. 5. The system shows an error, and the page is loaded again in the log in page.

(UC_02) Use case: Log out

Use case number	UC_02
Use case name	Log Out
Actors involved	User, admin, employee, businesses.
Description	This use case describes how a user logs out the web application. After the user is done with his work on the web application he can log out, so his information won't be accessible from others who use the same computer.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow 1	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. User works in his session. 6. User find the log out button and logs out.
Basic flow 2	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. User works in his session. 6. After a period of inactivity, the system logs out, without the

	need for a log out button.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. User works in his session. 6. User wants to log out, but he there does not exist a log out button.

(UC_03) Use case: Edit info

Use case number	UC_03
Use case name	Edit information
Actors involved	User, admin, employee, businesses.
Description	This use case describes how a user can edit his information like: name, surname or other data. When data that the user enters at the beginning need to be changed the user (admin, user, employee or businesses) find the possibility to change the wrong data they entered and to put correct data.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***setting*** menu option. 7. User can edit all his information that is in this page. 8. User saves the changes made.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username.

	4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***setting*** menu option. 7. User can edit all his information that is in this page. 8. The fields that the user can edit are validated. The user enters the information which is not as it should be (because of validation.) 9. Changes cannot be saved. 10. The system asks the user to enter again the information he want to change, based on the rule of validation.

(UC_04) Use case: Search employee/client/loan

Use case number	UC_04
Use case name	Search employee/client/ loan
Actors involved	User, admin, employee, businesses.
Description	This use case describes how a user can search information in the database. For example: when the admin wants to make some changes to the profile or the data of a client or employee, he can use the option search to make his job easier. When one employee want information about a client, he just searches by using the client name, or other data. In this way all the users, whatever role they have, will save time.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow 1	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***manage user's*** menu option. 7. Admin search the employee/client/business by name, surname, id or other data. 8. The search is successful. The employee/client/loan exist in the database.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like password or username.

	3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***manage user's*** menu option. 7. Admin search the employee/client/business by name, surname, id or other data. 8. The search is unsuccessful. The employee/client/loan doesn't exist in the database.

(UC_05) Use case: Manage loan

Use case number	UC_05
Use case name	Manage loans
Actors involved	Admin
Description	This use case describes how the admin can manage all the loans. All the loans are displayed in the home option, in the admin account. The admin when needed he can do different changes to the loans.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like pass or user. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. User check all the loans
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like pass or userna 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. The system does not load the ***home*** page.

(UC_06) Use case: Check available balance

Use case number	UC_06
Use case name	Check available balance
Actors involved	Admin
Description	This use case describes how the admin can check the available balance. The available amount of money that the company has is displayed in the home option on the admin account. When needed the admin can change this amount.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. Admin can check the amount of money available. 7. User can have changed this amount if he wants.
Alternative flow 1	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed.

	6. The amount available it's not shown.
Alternative flow 2	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. The amount available is shown. 7. User tries to change the amount. The change is not saved.

(UC_07) Use case: Generate income statement

Use case number	UC_07
Use case name	Generate income statement
Actors involved	Admin
Description	This use case describes how the admin can generate income statement. The income statement can be generated under the ***Statements*** option in the menu.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like pass or user. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***Statements*** menu option. 7. User select ***Get income statement*** option. 8. The income statement is shown. 9. User can print the statement.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like passw or user 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***Statements*** menu option. 7. User selects ***Get income statement*** option. 8. The income statement is not shown.

(UC_08)Use case: Generate balance sheet statement

Use case number	UC_08
Use case name	Generate balance sheet statement
Actors involved	Admin
Description	This use case describes how the admin can generate balance sheet statement. The balance sheet statement can be generated under the ***Statements*** option in the menu.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like pass or user 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***Statements*** menu option. 7. User select ***Get balance sheet statement*** option. 8. The balance sheet statement is shown.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like pass or user 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***Statements*** menu option. 7. User select ***Get balance sheet statement*** option. 8. The balance sheet statement is not shown.

(UC_09) Use case: Generate cash flow statement

Use case number	UC_09
Use case name	Generate cash flow statement
Actors involved	Admin
Description	This use case describes how the admin can generate cash flow statement. The cash flow statement can be generated under the ***Statements*** option in the menu.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like pass or user 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***Statements*** menu option. 7. User select ***Get cash flow statement*** option. 8. The cash flow statement is shown.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like pass or user. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page. 6. User goes to ***Statements*** menu option. 7. User select ***Get cash flow statement*** option. 8. The cash flow statement is not shown.

(UC_10) Use case: check active loans

Use case number	UC_10
Use case name	Check active loans
Actors involved	Employee
Description	This use case describes how the employee can check the active loans of the company. The active loans can be checked in the option on the menu: ***Home***.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. Employee can check all the loans that are shown in the desktop.
Alternative flow	No alternative flow.

(UC_11) Use case: Pay Loan

Use case number	UC_11
Use case name	Pay loan
Actors involved	Employee
Description	One of main task of the employee will be: pay loan. When the client makes the payment in the offices of IUTE credit the employee uses the options: pay loan, which is under the ***home*** option, in the menu.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. The user click ***Pay loan*** button. 7. A window is shown which asks for the information of the loan, for example ID of loan, or name of the person that has the loan. 8. User enters the information. 9. The information is correct and validated. 10. User clicks the button ***Pay Loan.*** 11. System asks: "Are you sure you want to pay the loan?". 12. Click "Yes". The amount of loan that needs to be payed

	decreases, and the available amount increases.
Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. The user click ***Pay loan*** button. 7. A window is shown which asks for the information of the loan, for example ID of loan, or name of the person that has the loan. 8. User enters the information. 9. The information is not correct. For example: ID of the loan doesn't exist. 10. System gives an alert: "The loan with this ID doesn't exist." 11. User enters again the information. 12. The information is correct. 13. System asks: "Are you sure you want to pay the loan?". 14. Click "Yes". The amount of loan that needs to be payed decreases, and the available amount increases.

(UC_12) Use case: Give Loan

Use case number	UC_12
Use case name	Give loan
Actors involved	Employee
Description	Another task of the employee will be: give loan. When the client want a loan and meets all the conditions the employee uses the options: gice loan, which is under the ***home*** option, in the menu.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. Employee clicks ***Give loan*** button. 7. System opens a new window, which asks the client information. 8. The information entered is correct and validated. 9. System check if there is enough balance. 10. The balance is enough for the loan. 11. System asks: “Are you sure you want to give this loan”. 12. Available amount shown in ***Home*** view changes.

Alternative flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. Employee clicks ***Give loan*** button. 7. System opens a new window, which asks the client information. 8. The information entered is not correct. 9. An alert is shown which says "Please, enter correct information!" 10. User enters again the information. This time information is correct. 11. System check if there is enough balance. 12. The balance is not enough for the loan. 13. System gives an alert "There is not enough balance. Please, check the amount of money you entered!". 14. User enters the max amount available. 15. System asks: "Are you sure you want to give this loan". 16. Available amount shown in ***Home*** view changes.

(UC_13) Use case: Delete Loan

Use case number	UC_13
Use case name	Delete loan
Actors involved	Employee
Description	Another task of the employee will be: delete loan. When the employee wants to delete a loan he can go to the loan and delete it.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. The user opens the login page. 2. The system asks the user for his information like password or username. 3. The user enters the password and the username. 4. The system validates the password and the username and then logs the user into their account. 5. The system opens automatically in the ***home*** page where all loans are showed. 6. Employee clicks ***Delete loan*** button. 7. Systems prompts for permission. 8. Loan is deleted. 9. Available loans that are shown in ***Home*** view change.
Alternative flow	There is no alternative flow

(UC_14) Use case: Send messages

Use case number	UC_14
Use case name	Send messages
Actors involved	Employee
Description	This use case describes how the employee comunicates with the user. after the employee get the any message from admin he shares this (if it's needed) with the simple user.
Precondition	First must have been completed the login case.
Priority	Primary and essential
Frequency of use	Many times, per day
Basic flow	1. User enters username and password 2. System validates the entered information 3. Username and password are added correctly 4. Employee send message/feedback to the Admin 5. Log out
Alternative flow	1. User enters username and password 2. System doesn’t validate the entered information 3. User enters again the username and password. 4. Username and password are added correctly 5. Employee send message/feedback to the admin 6. Log out

(UC_15) Use case: Check if you meet the conditions for loan

Use case number	UC_15
Use case name	Check if you meet the conditions for loans
Actors involved	User
Description	This use case describes how the user can check if he can get the loan he wants from IUTE credit based on their income.
Precondition	First must have been completed the login case.
Priority	
Frequency of use	Many times, per day
Basic flow	1. User enters username and password 2. System validates the entered information 3. Username and password are added correctly 4. User enter the information like salary etc. to check if he is able to take the loan. 5. System show if the user can take or not any loan from the IUTE. 6. Log out
Alternative flow	1. User enters username and password 2. System validates the entered information 3. Username and password are added correctly 4. User enter the information like salary etc. to check if he is able to take the loan. 5. The user enters incorrect info, or he does not meet the conditions of the IUTE to get the loan. 6. Log out

(UC_16) Use case: Get alerts

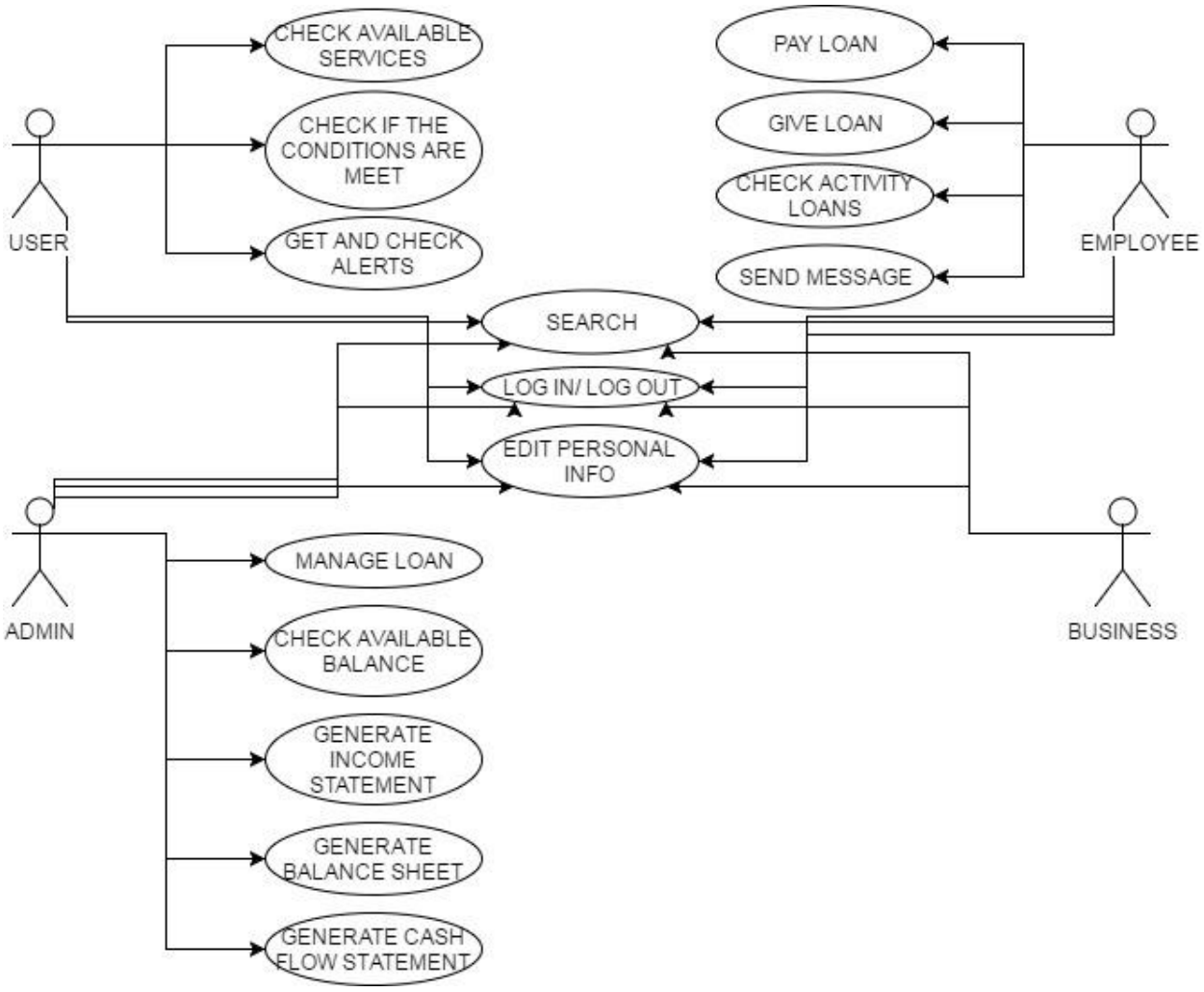
Use case number	UC_16
Use case name	Get and check alerts
Actors involved	User
Description	This use case describes how the user can get alerts from the IUTE. Alerts are in types of: "The deadline of paying your is near" or any other type. The user can disable the alert.
Precondition	First must have been completed the login case.
Priority	
Frequency of use	Many times, per day
Basic flow	1. User enters username and password 2. System validates the entered information 3. Username and password are added correctly 4. When the payments of the loan pass the deadline to be paid the system send alert to the user. 5. User can disable the alert after being send, or can mark it as ***checked.*** 6. Log out
Alternative flow	No alternative flow

(UC_17) Use case: Check the services available

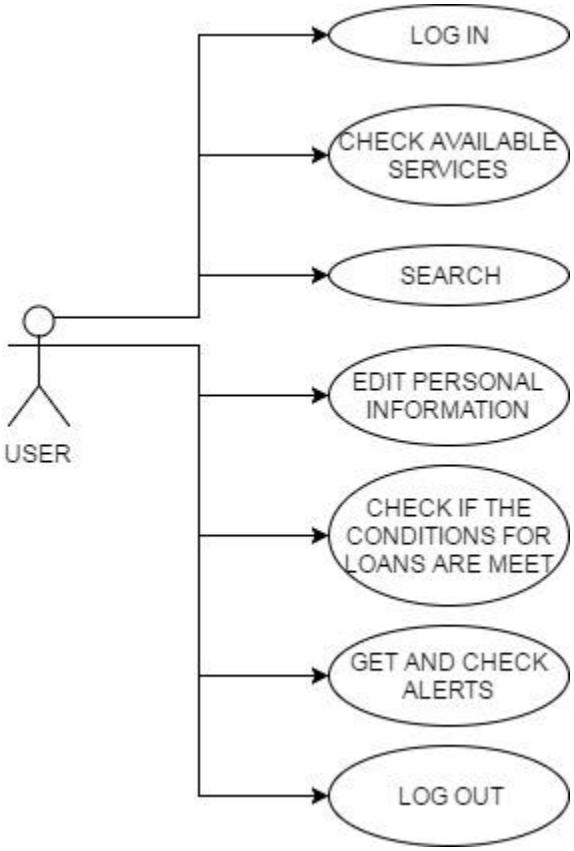
Use case number	UC_17
Use case name	Check the services available
Actors involved	User
Description	This use case describes how the user can chech all the services that IUTE credit has for the customer. Also the customer can decide if he wants to take other available service and put them in ***My services*** option, which is under the ***Services*** menu.
Precondition	First must have been completed the login case.
Priority	
Frequency of use	Many times, per day
Basic flow	1. User enters username and password 2. System validates the entered information 3. Username and password are added correctly 4. User goes under the section My services to check for services that the company offers. 5. Log out
Alternative flow	No alternative flow

DIAGRAMS

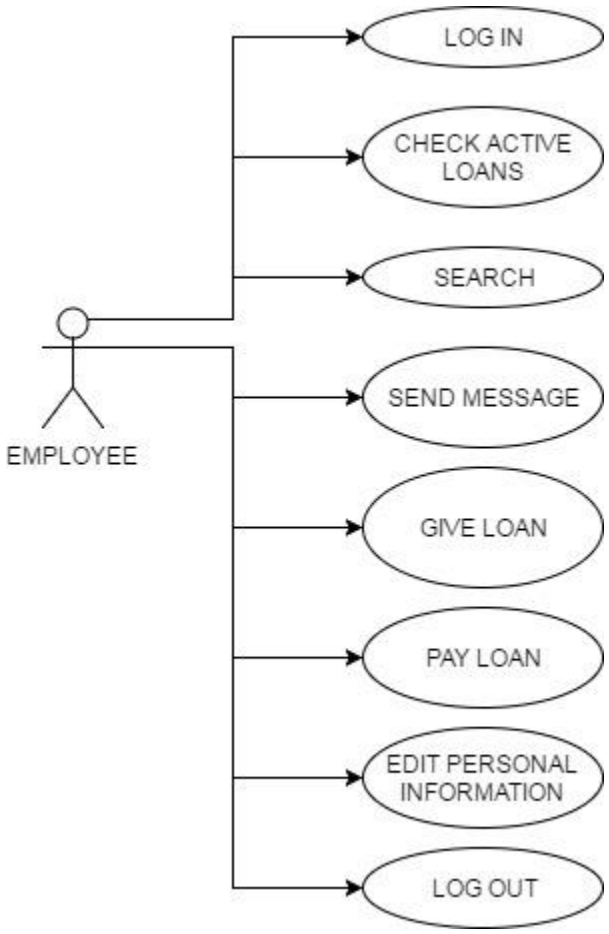
USE CASE DIAGRAM



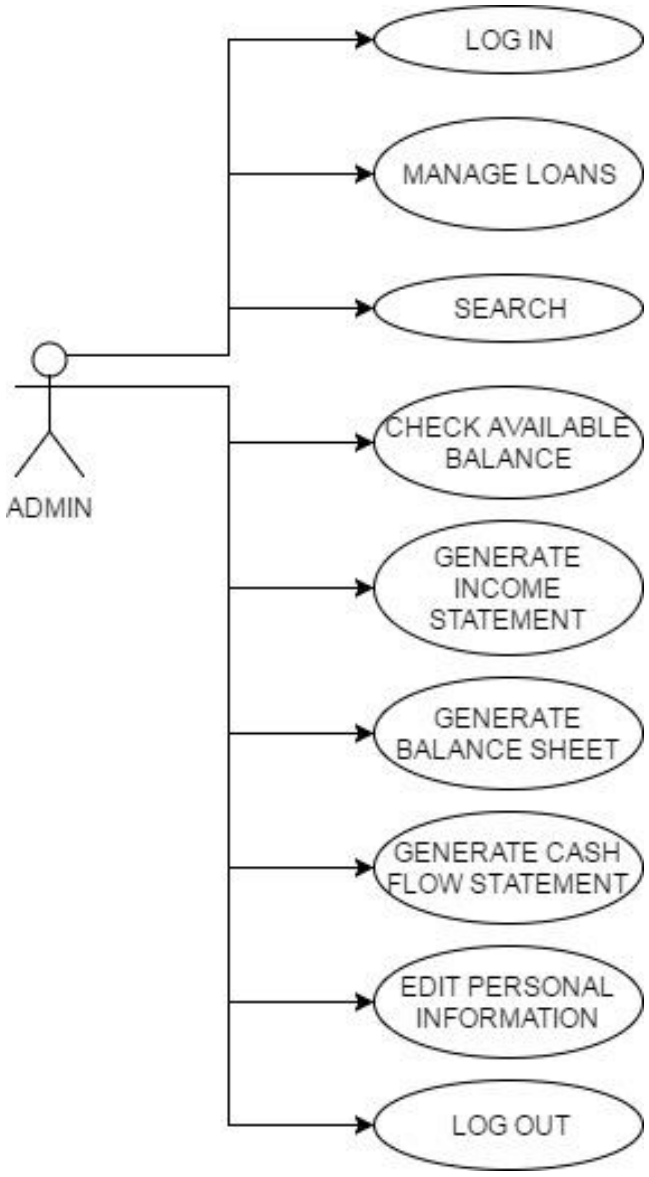
USER DIAGRAM



EMPLOYEE DIAGRAM

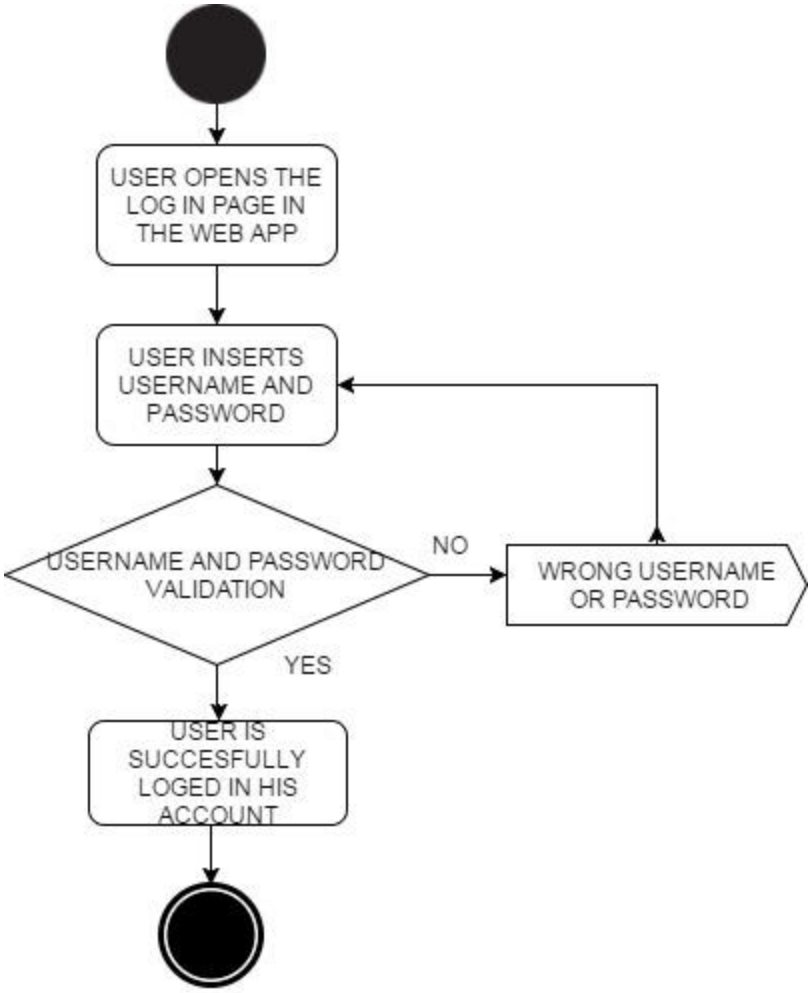


ADMIN DIAGRAM

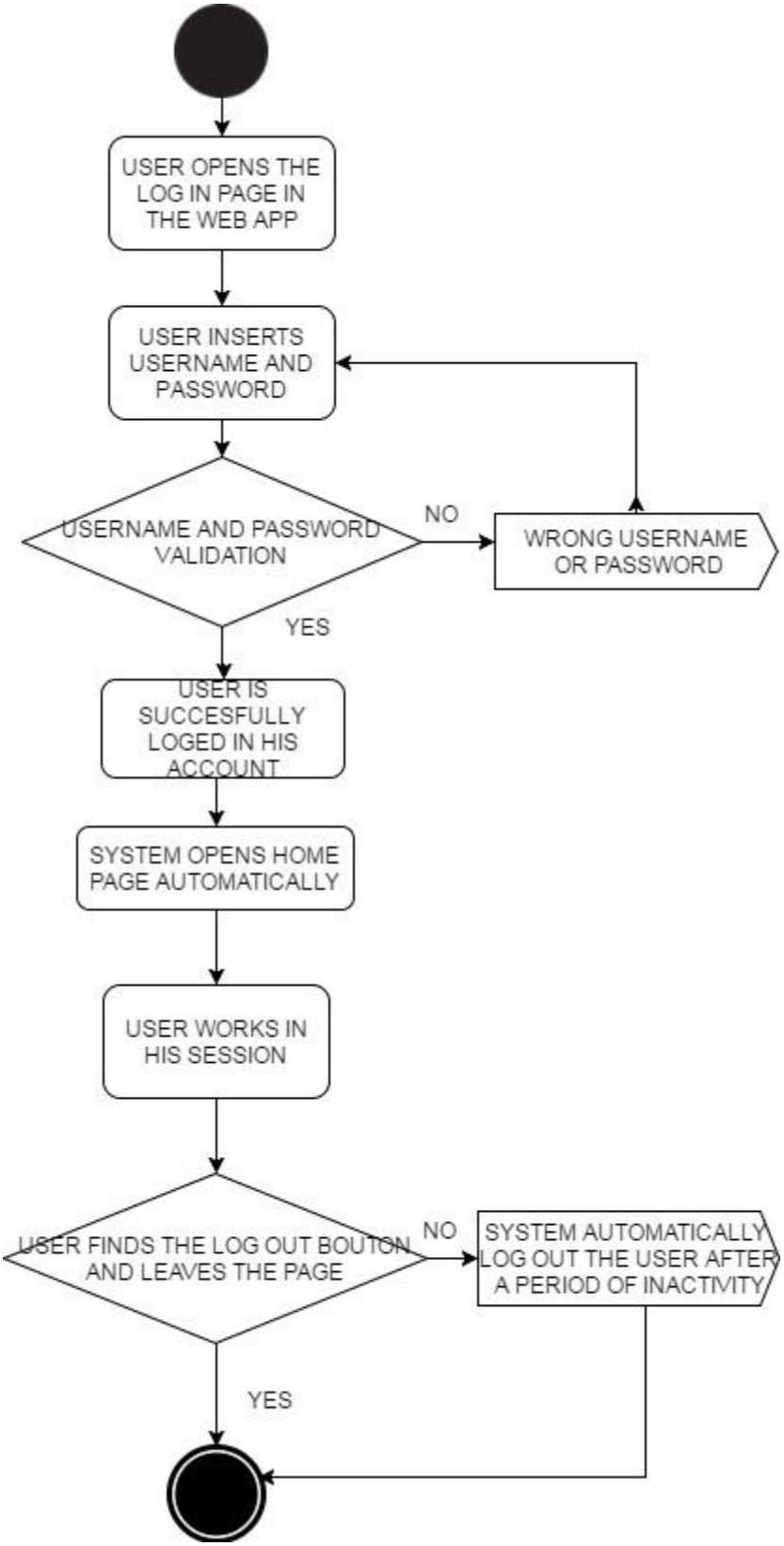


ACTIVITY DIAGRAM

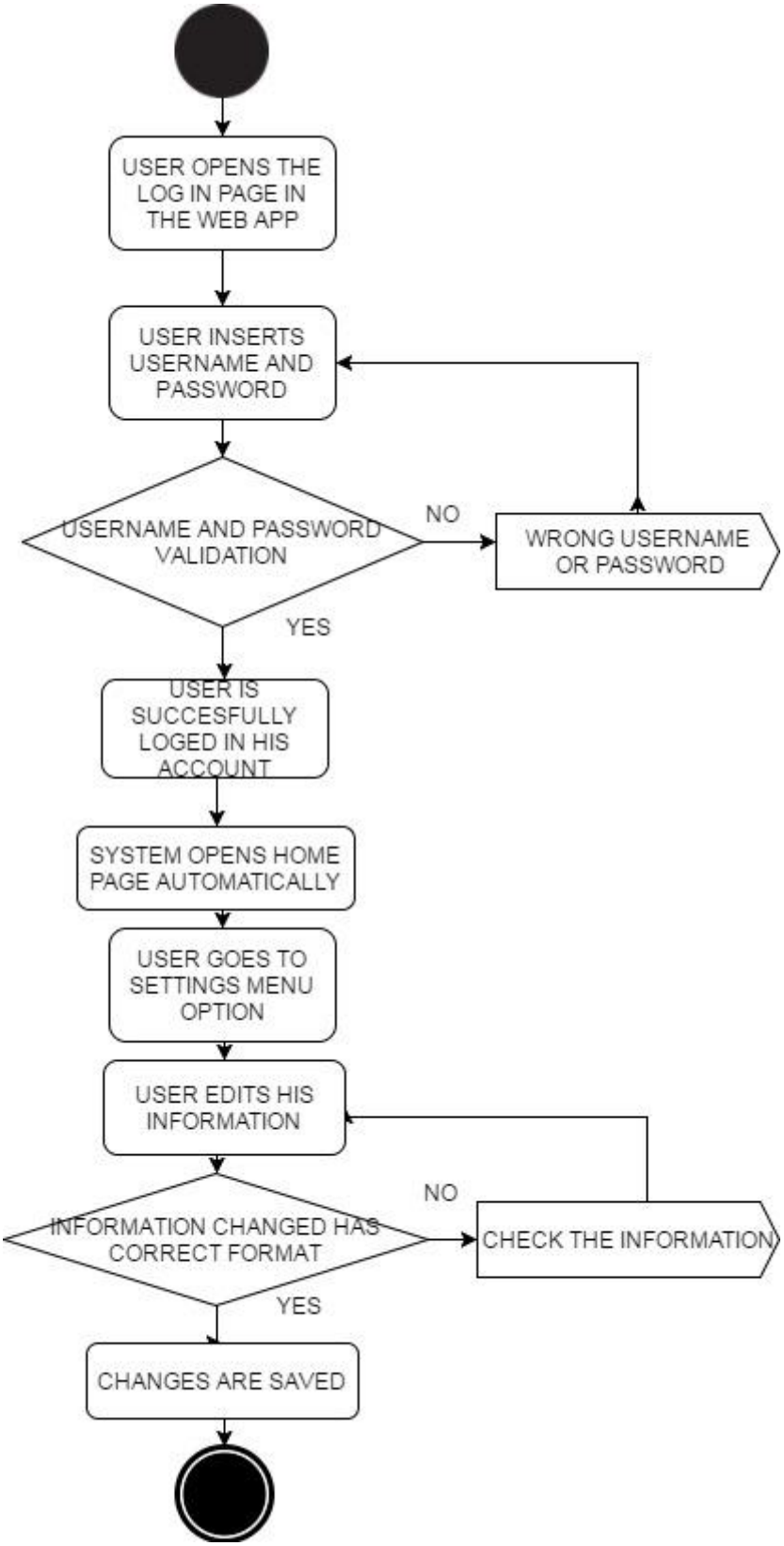
(AD_01) Activity Diagram of UC_01



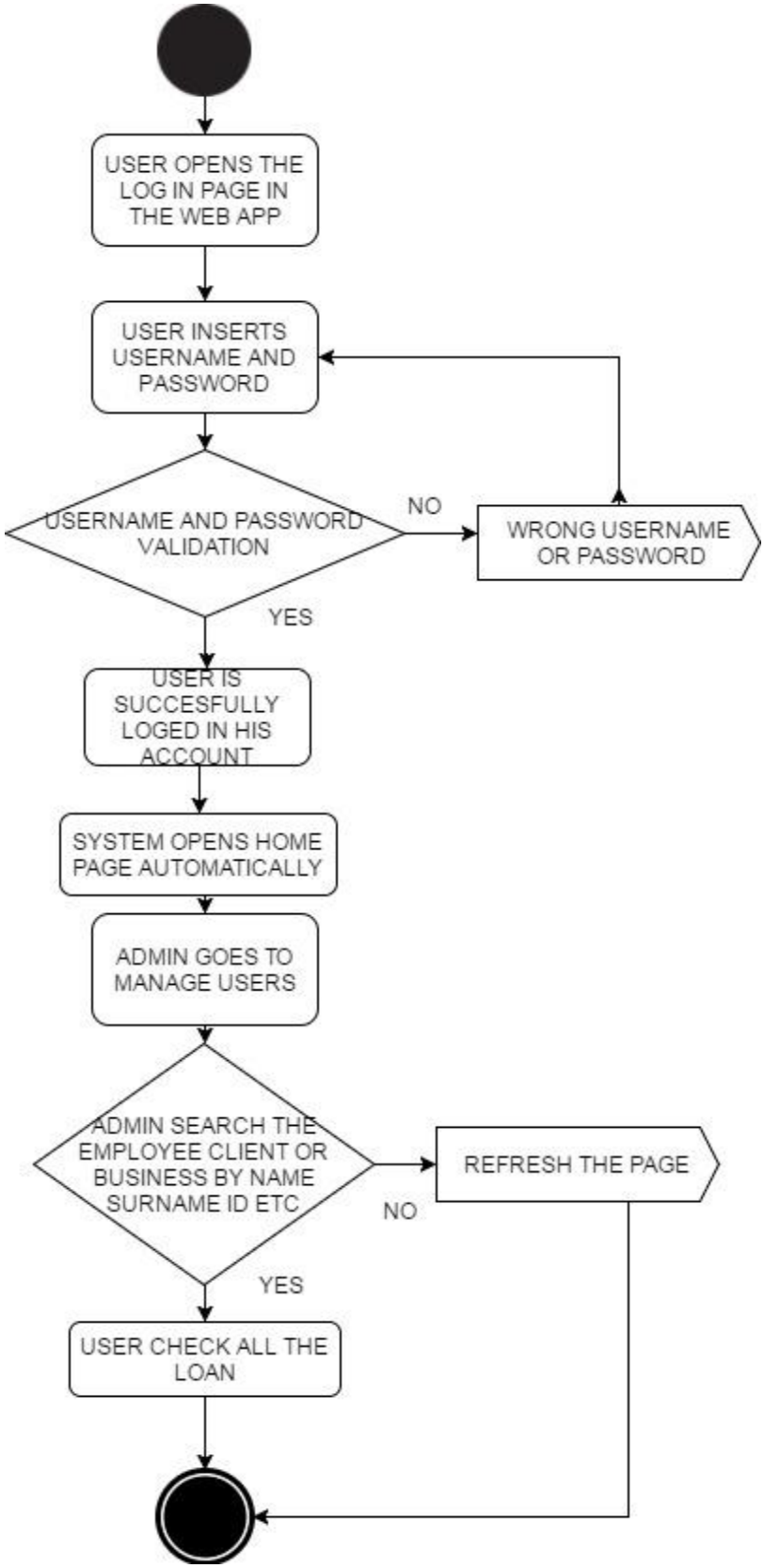
(AD_02) Activity Diagram of UC_02



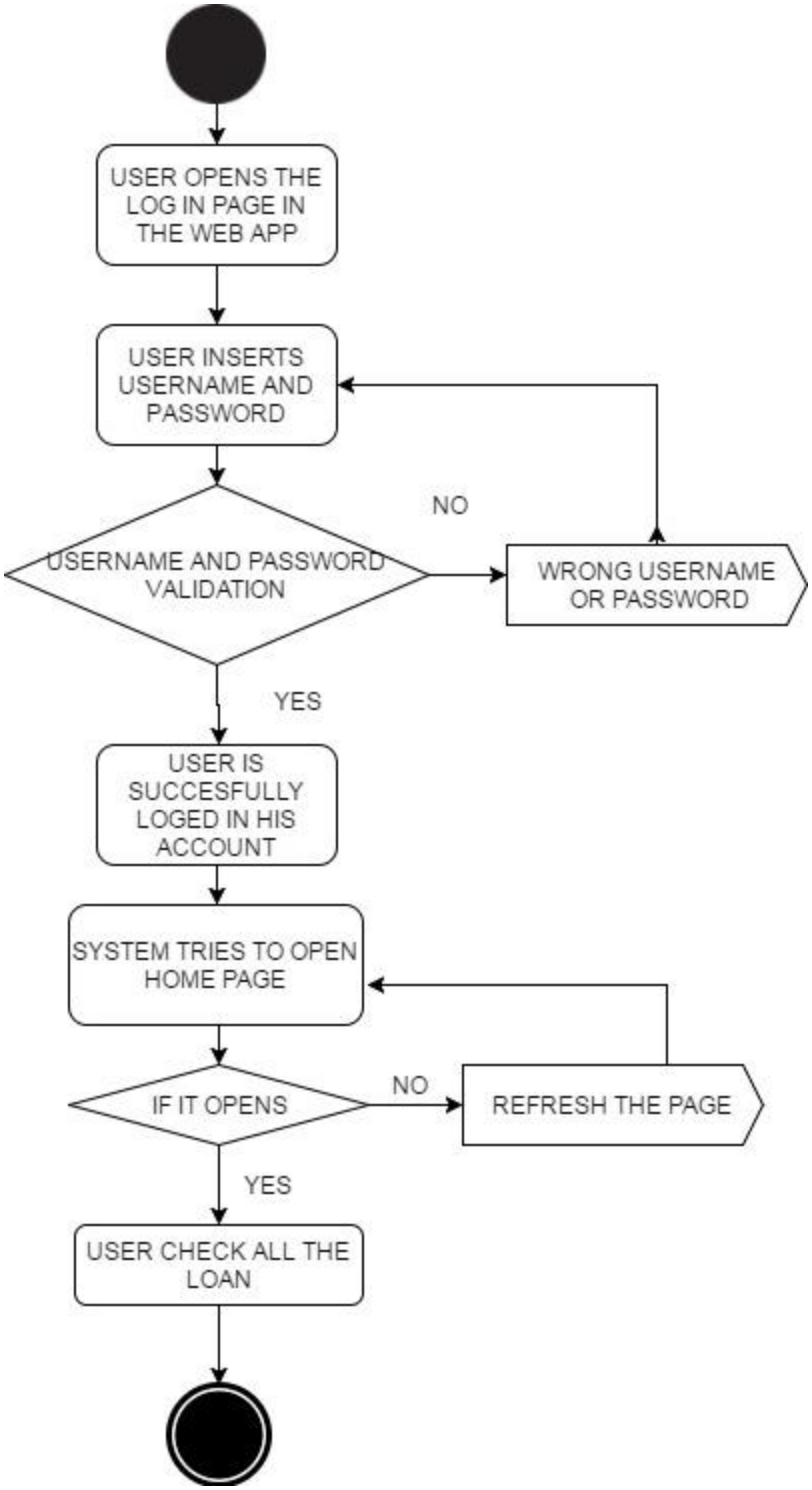
(AD_03) Activity Diagram of UC_03



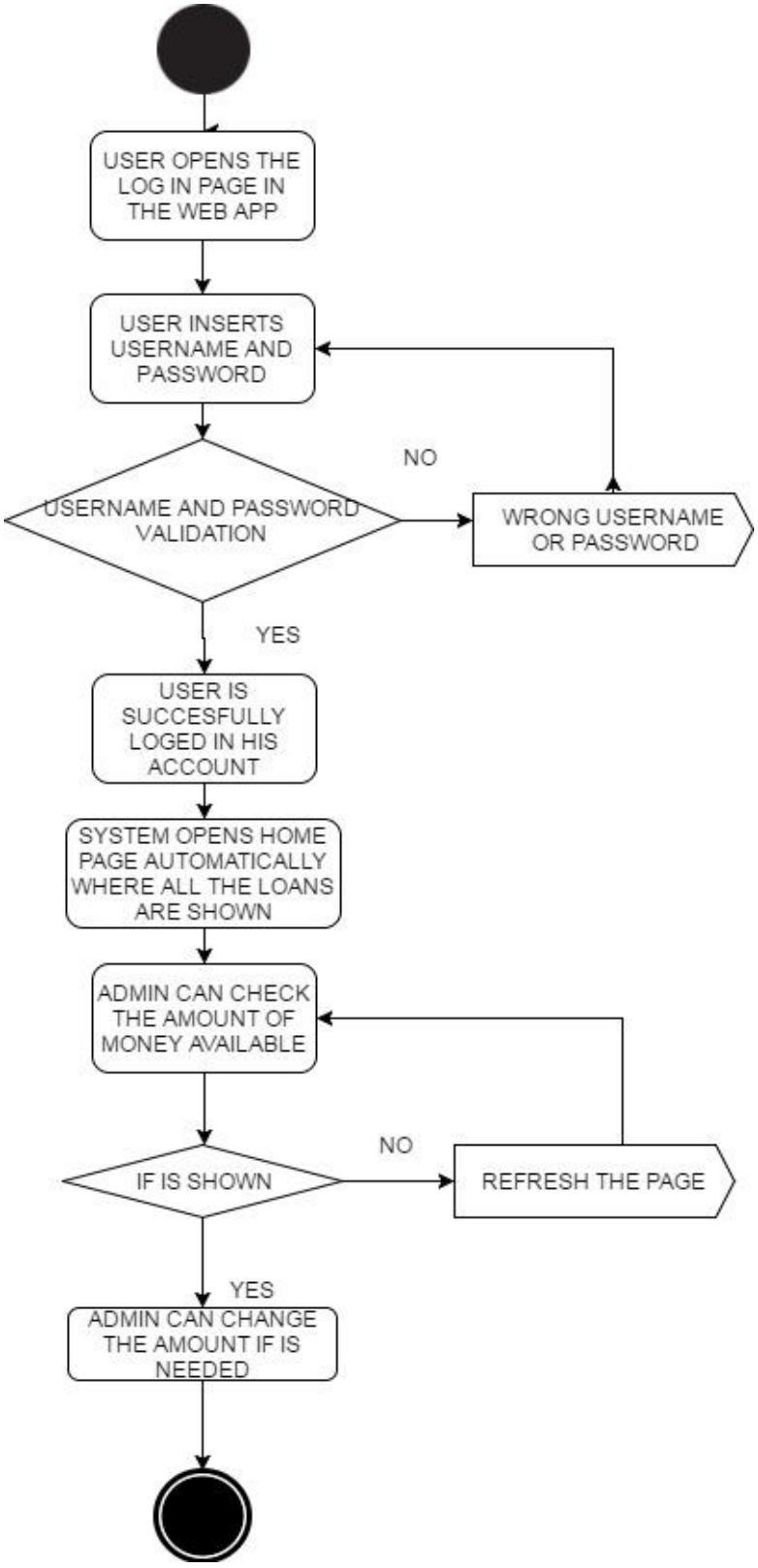
(AD_04) Activity Diagram of UC_04



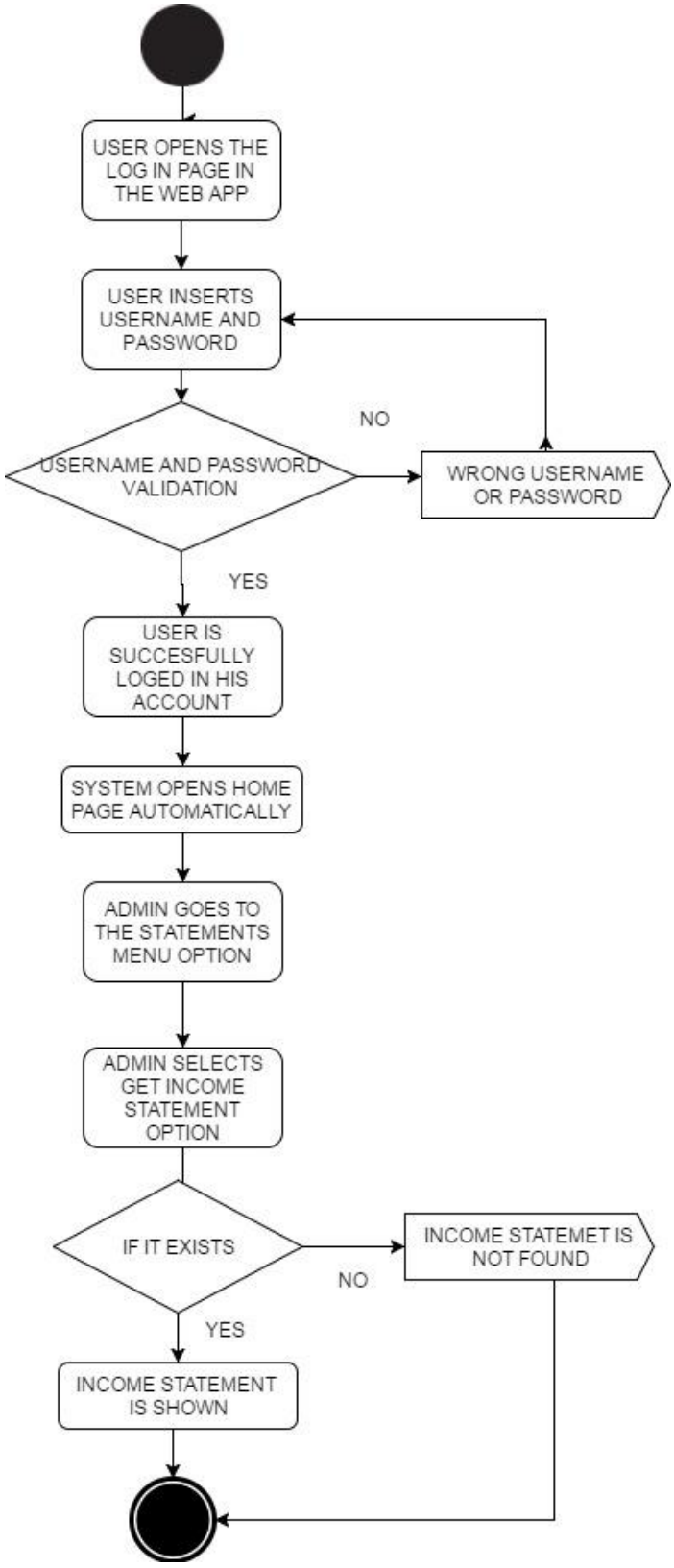
(AD_05) Activity Diagram of UC_05



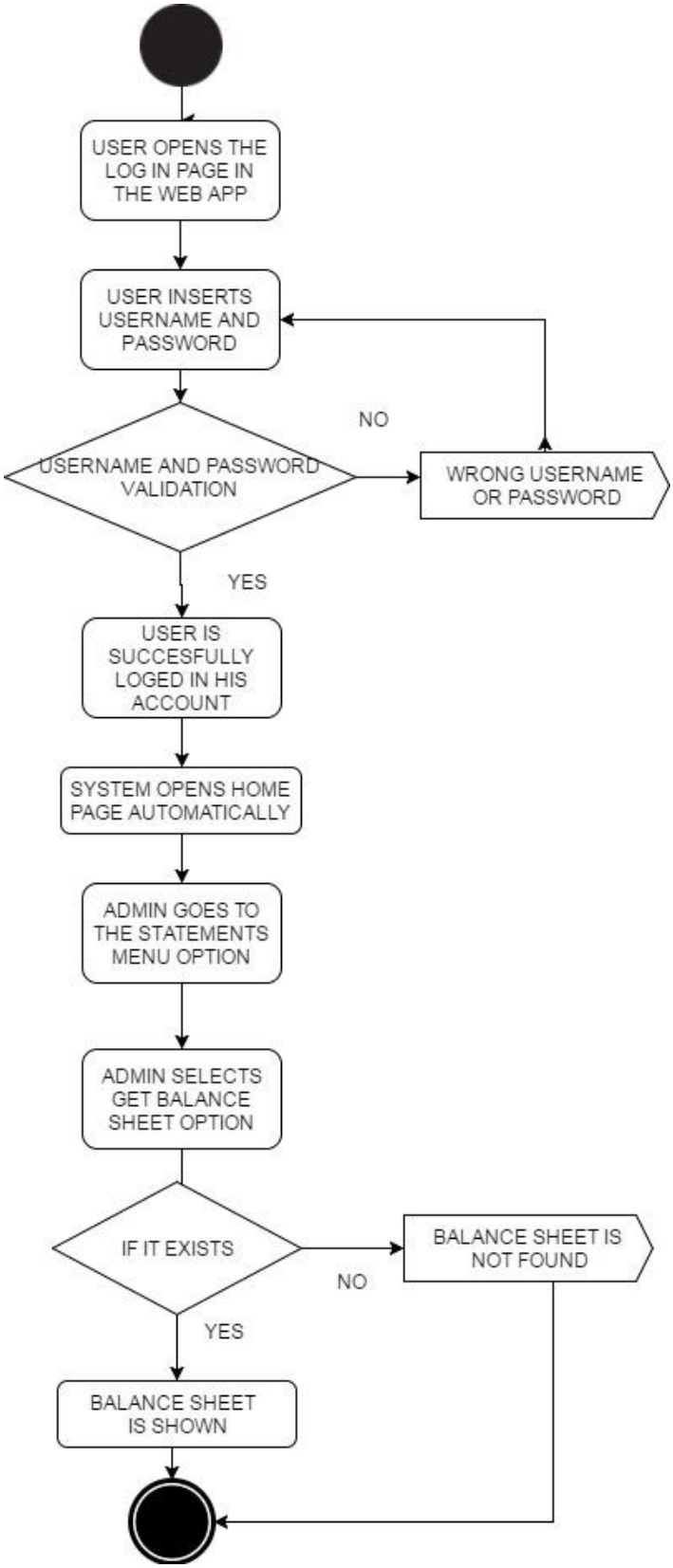
(AD_06) Activity Diagram of UC_06



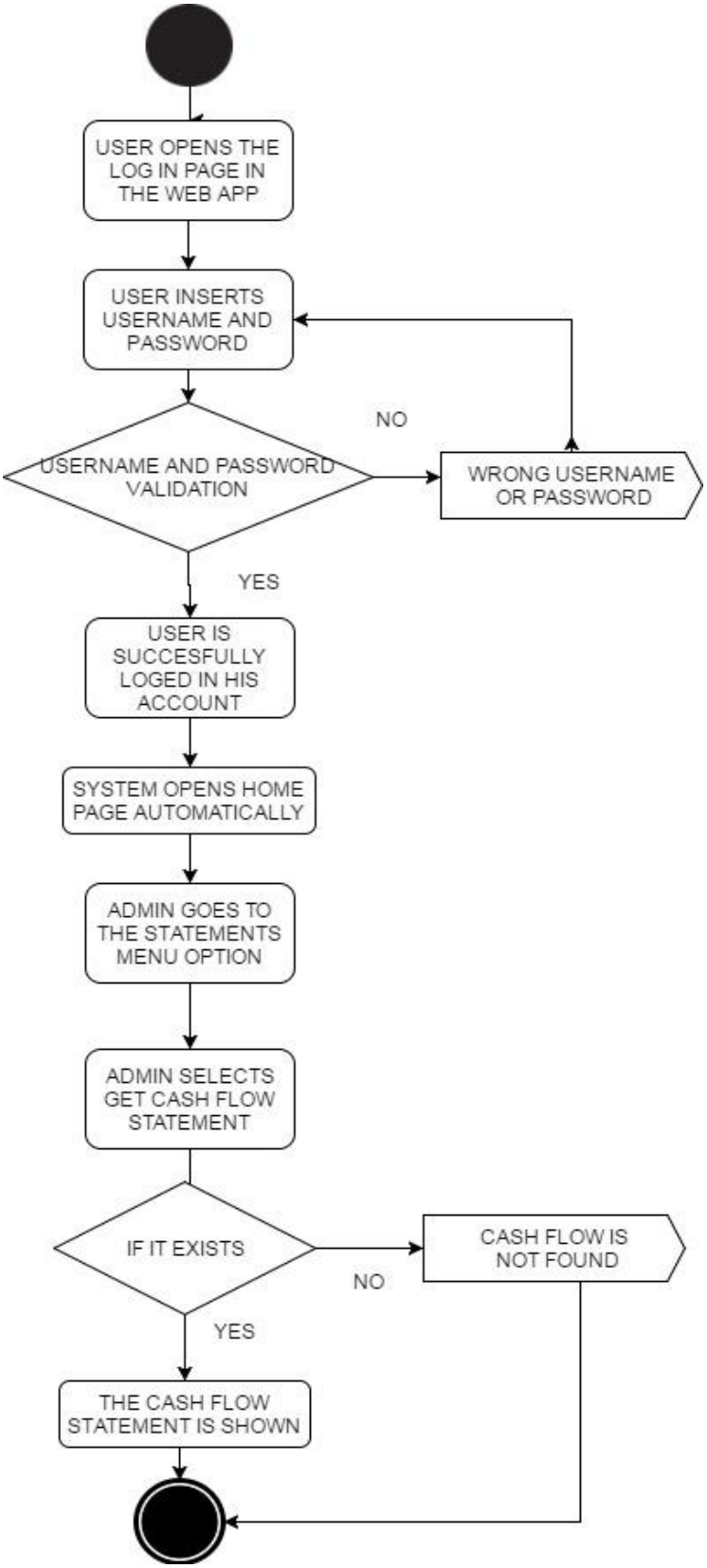
(AD_07) Activity Diagram of UC_07



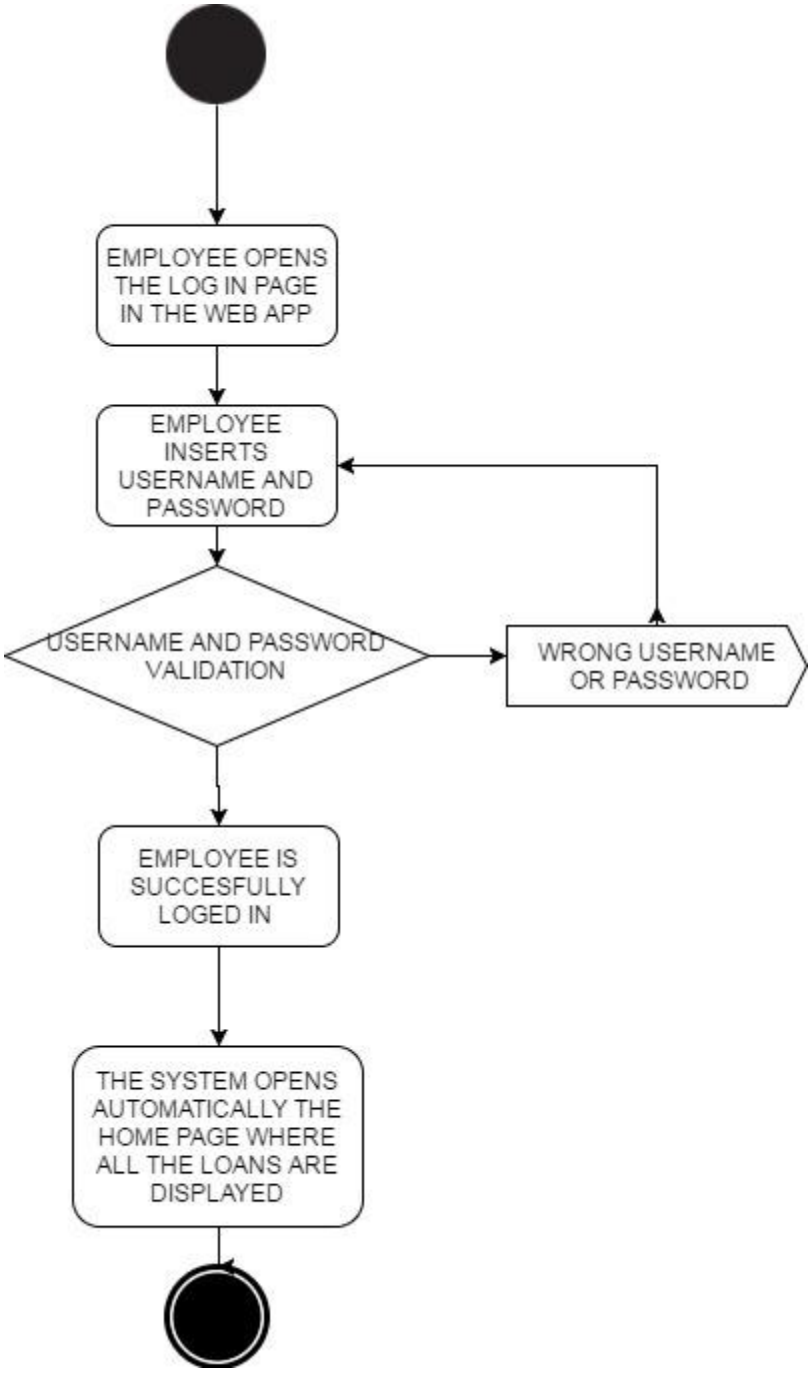
(AD_08) Activity Diagram of UC_08



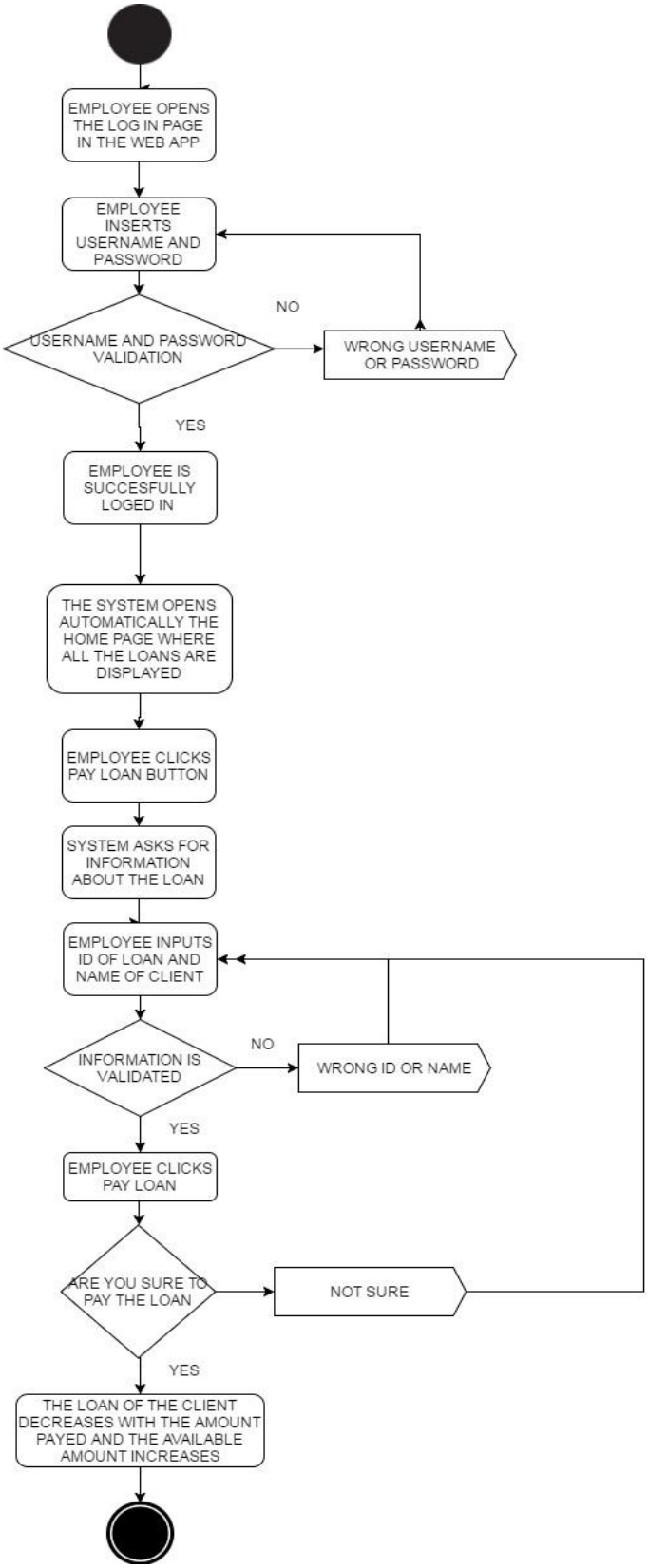
(AD_09) Activity Diagram of UC_09



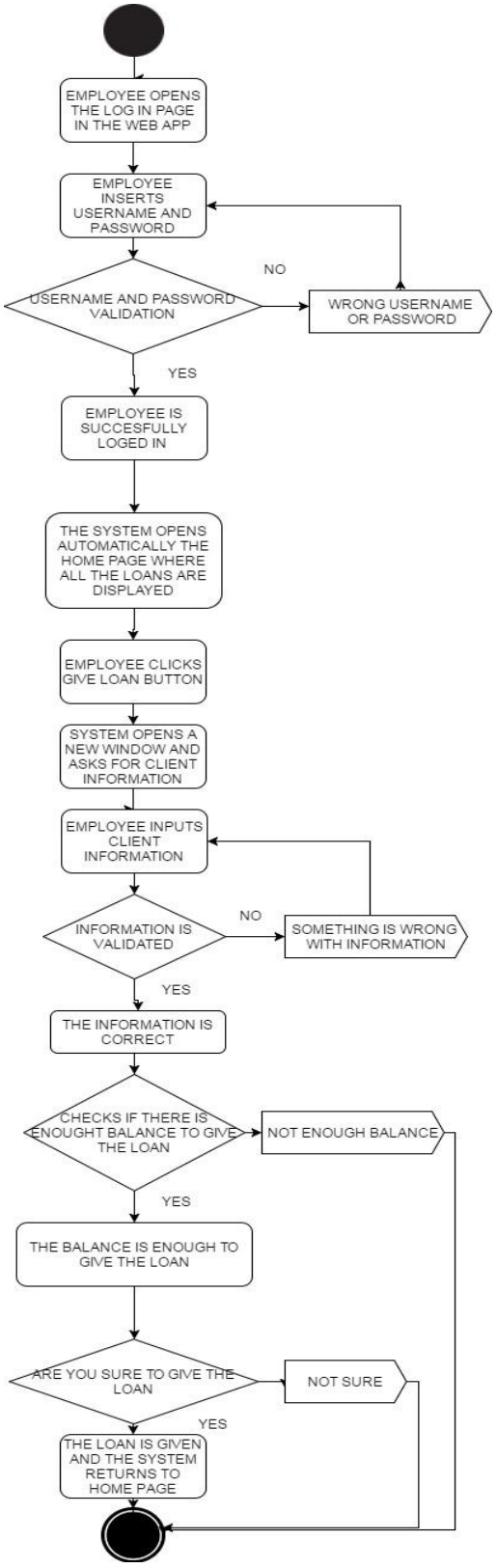
(AD_10) Activity Diagram of UC_10



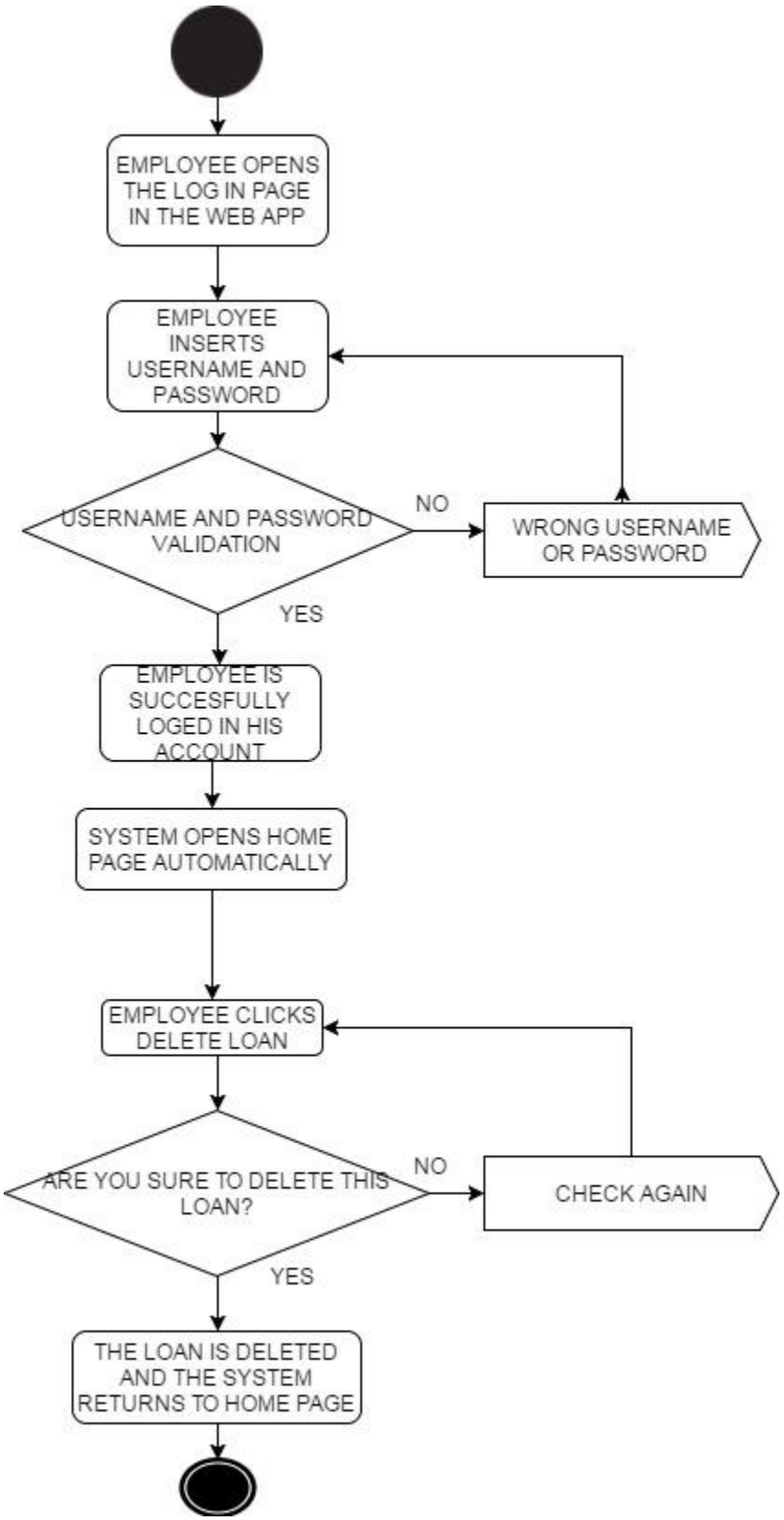
(AD_11) Activity Diagram of UC_11



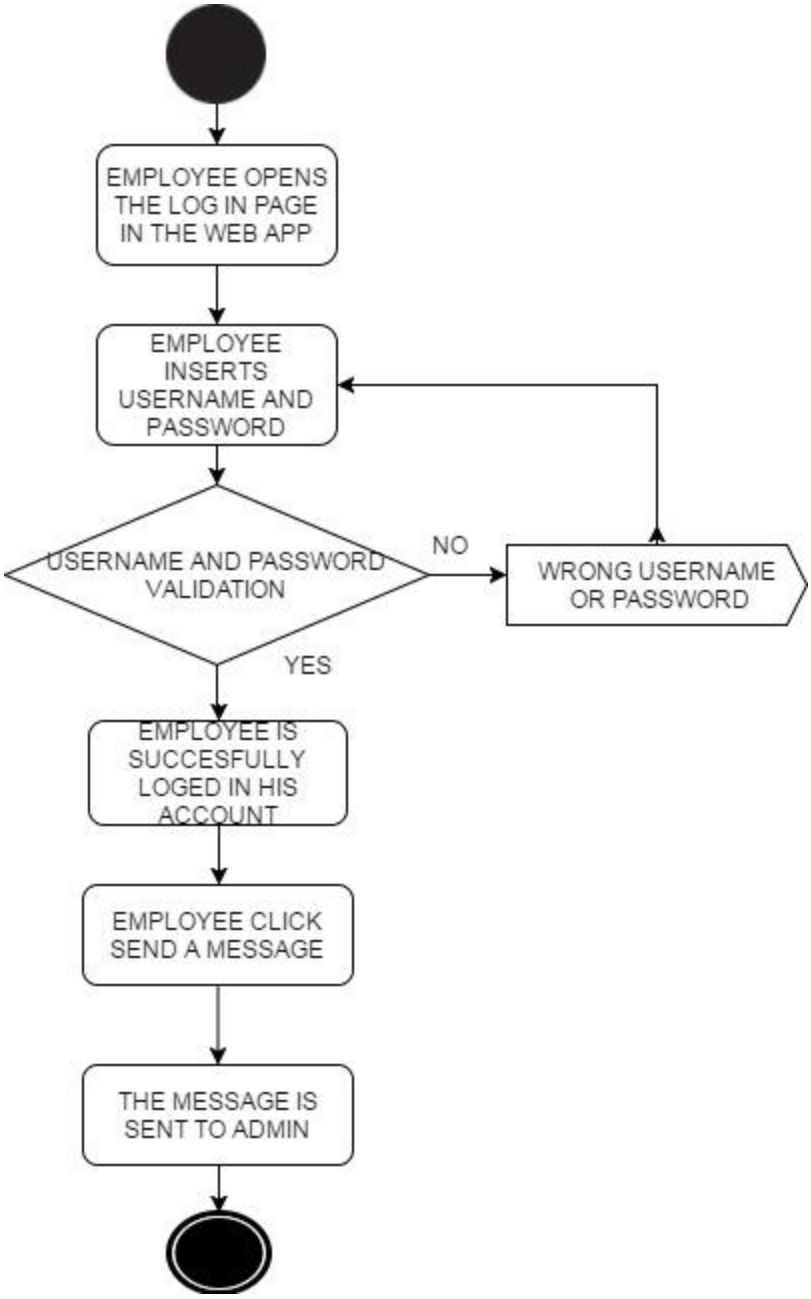
(AD_12) Activity Diagram of UC_12



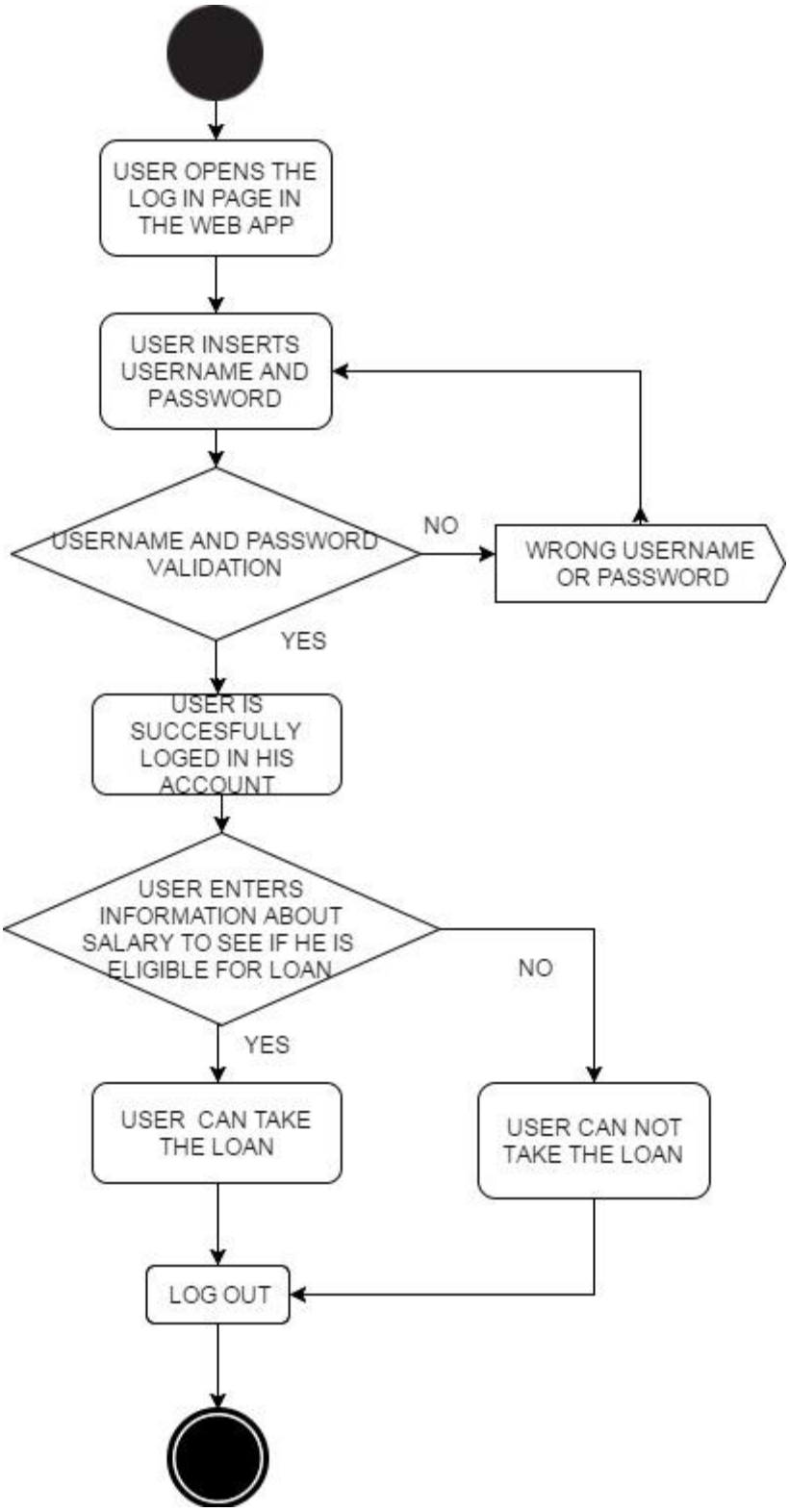
(AD_13) Activity Diagram of UC_13



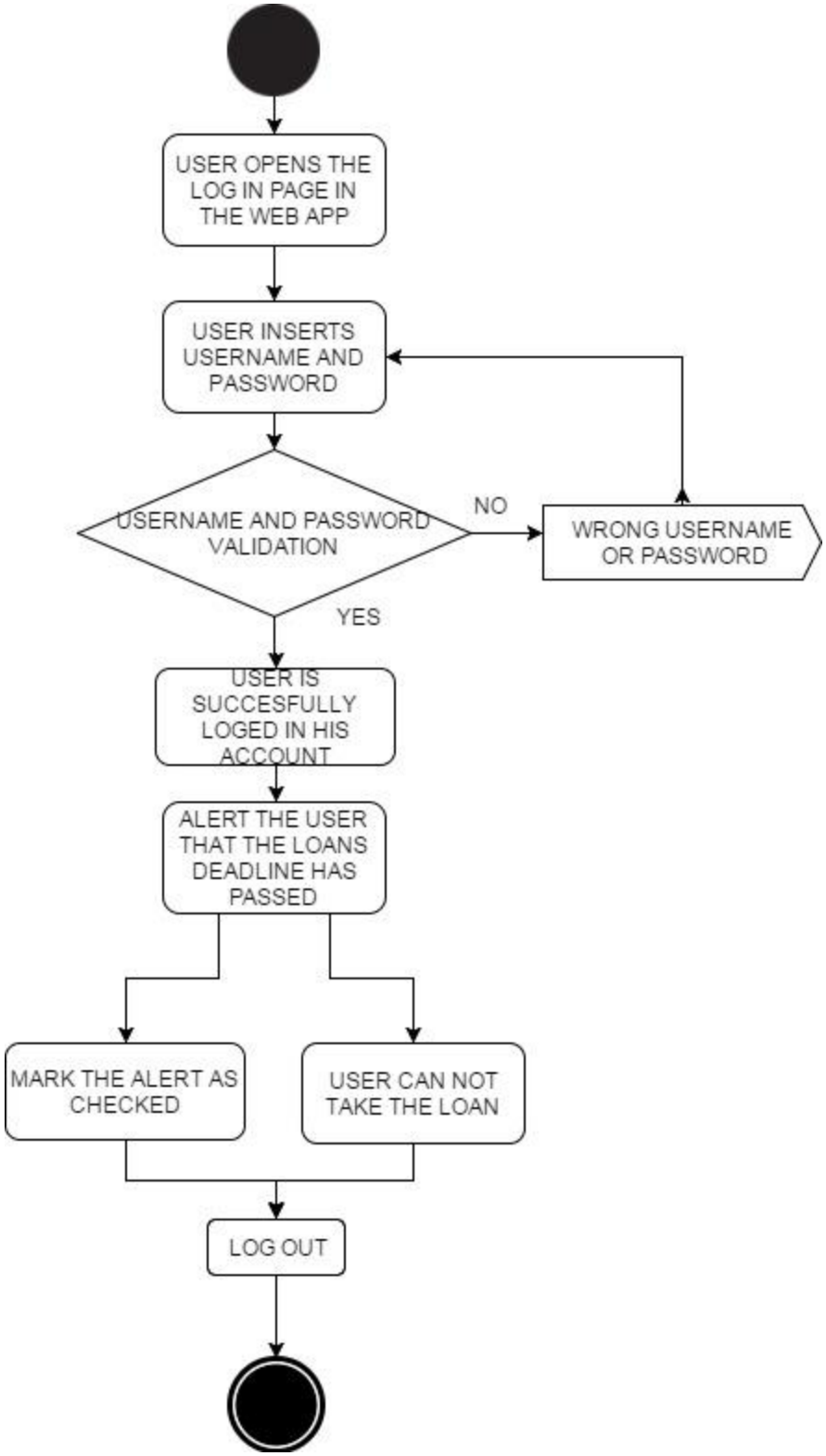
(AD_14) Activity Diagram of UC_14



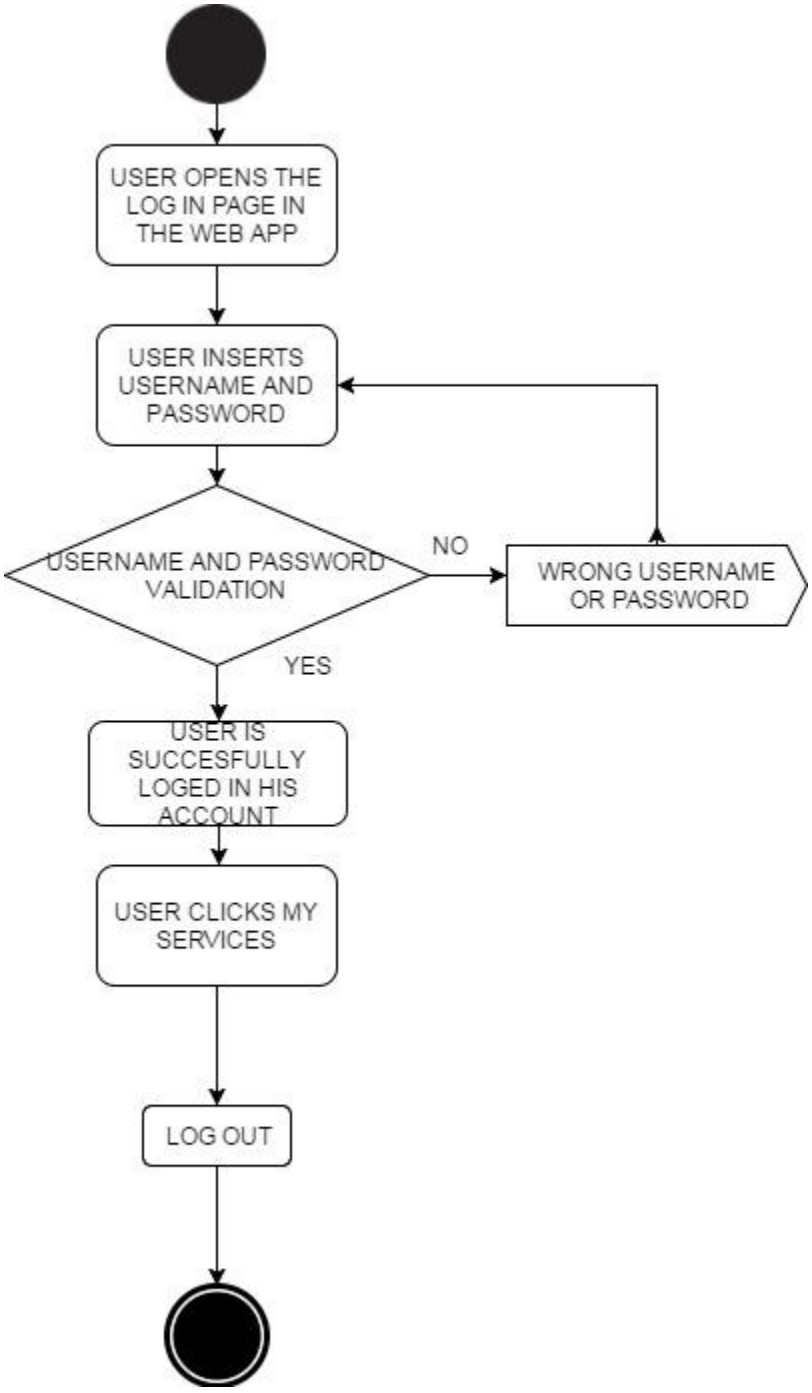
(AD_15) Activity Diagram of UC_15



(AD_16) Activity Diagram of UC_16

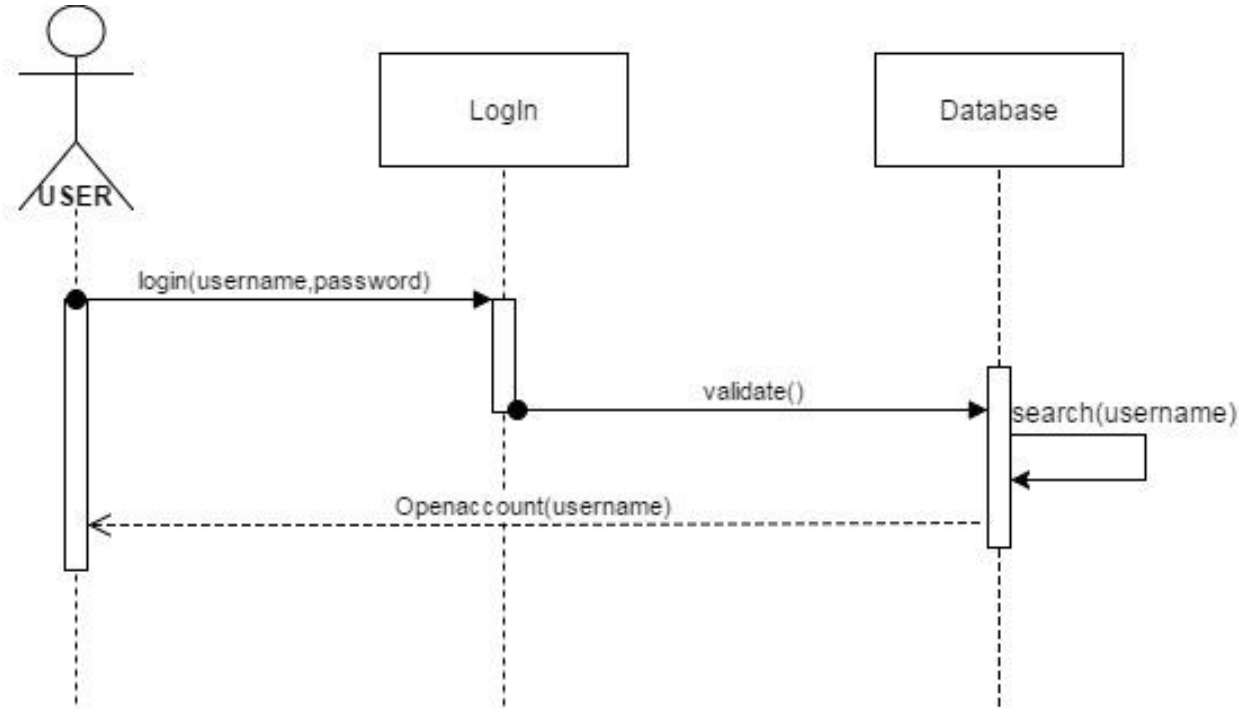


(AD_17) Activity Diagram of UC_17

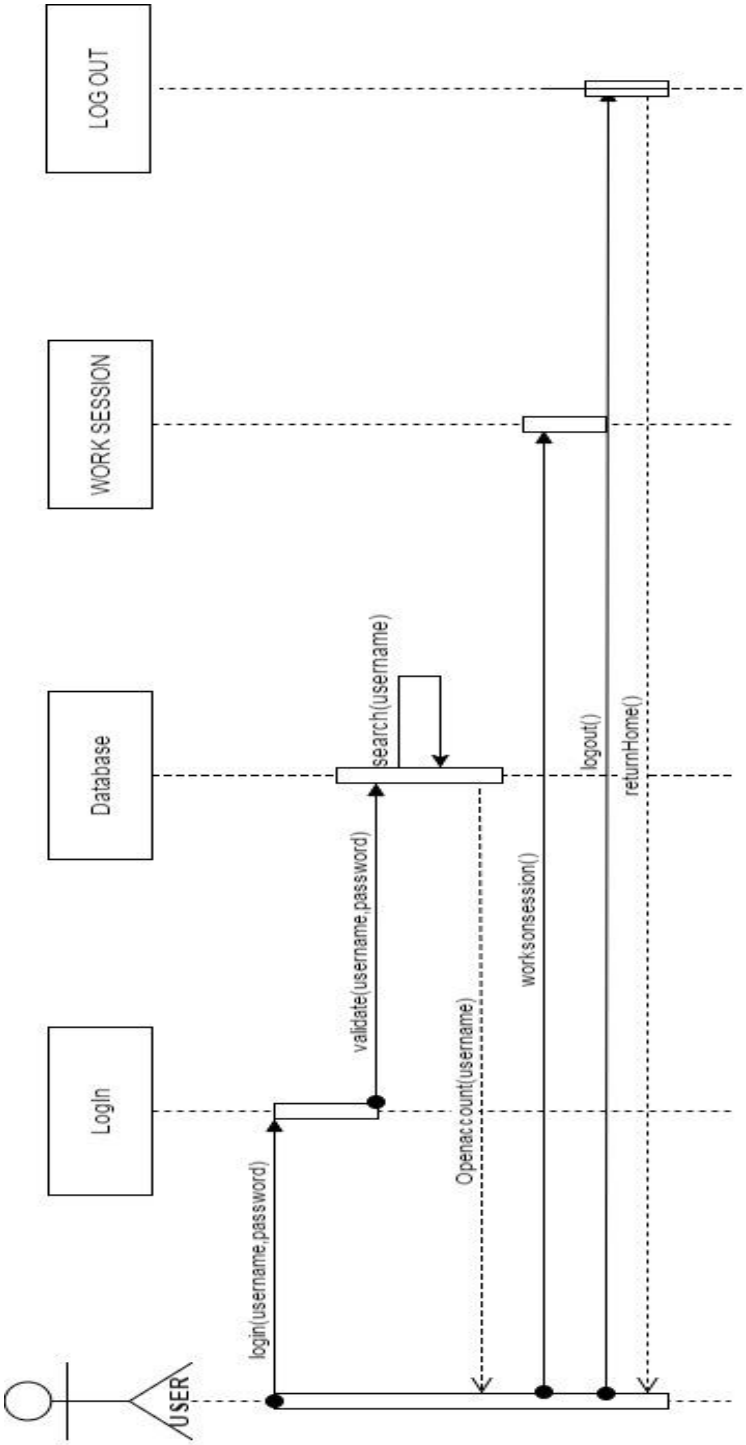


SEQUNCE DIAGRAMS

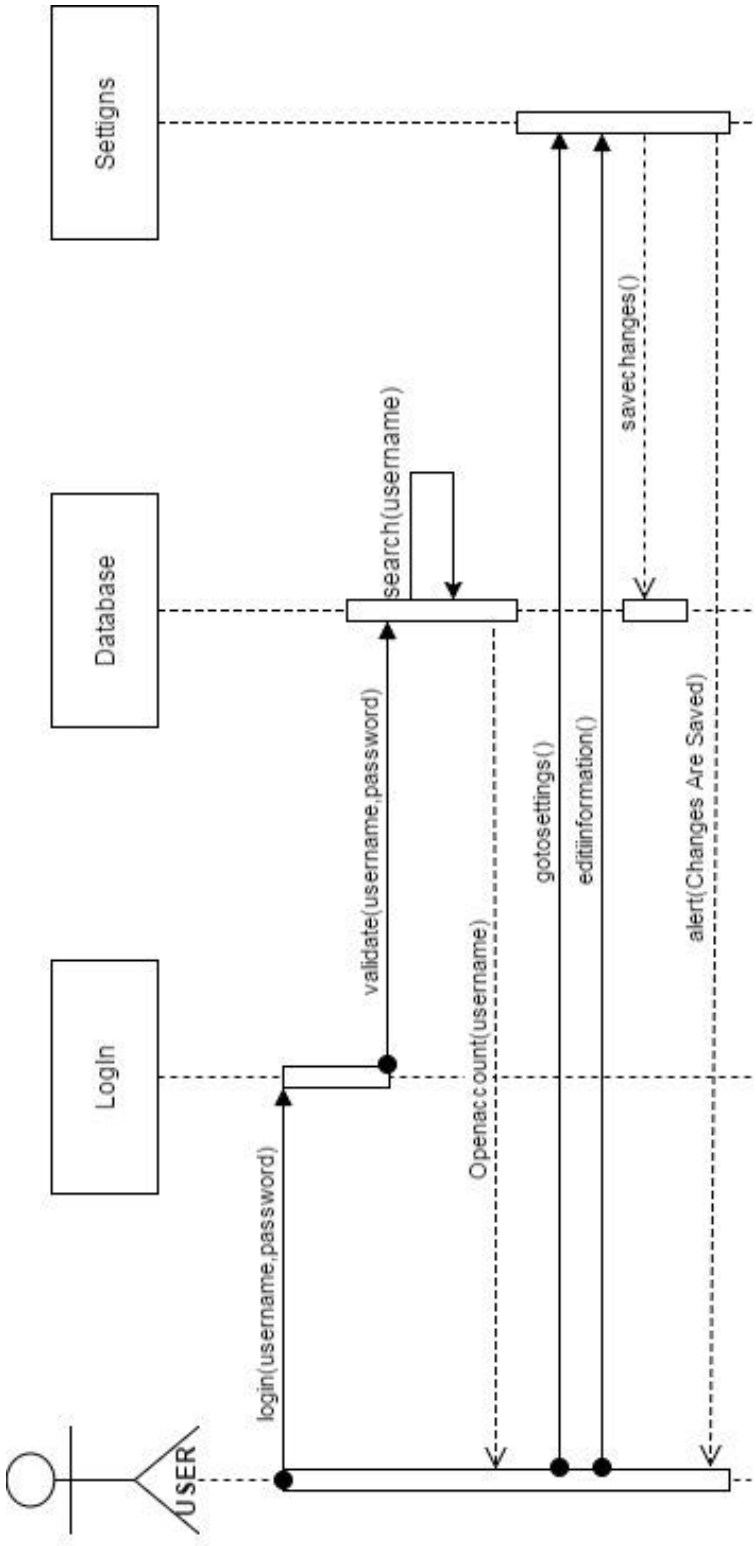
(SD_01) SEQUENCE DIAGRAM OF UC_01



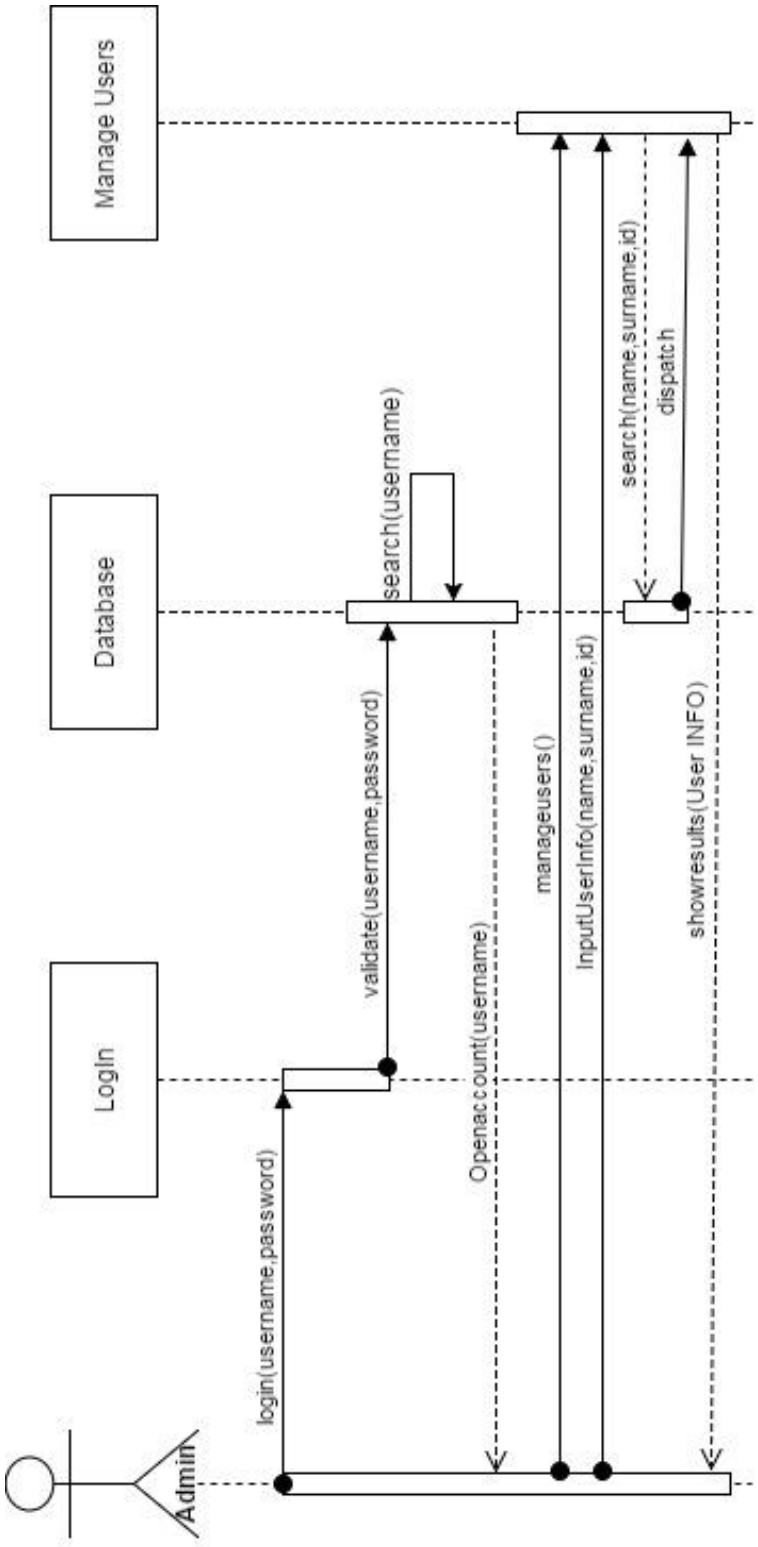
(SD_02) SEQUENCE DIAGRAM OF UC_02



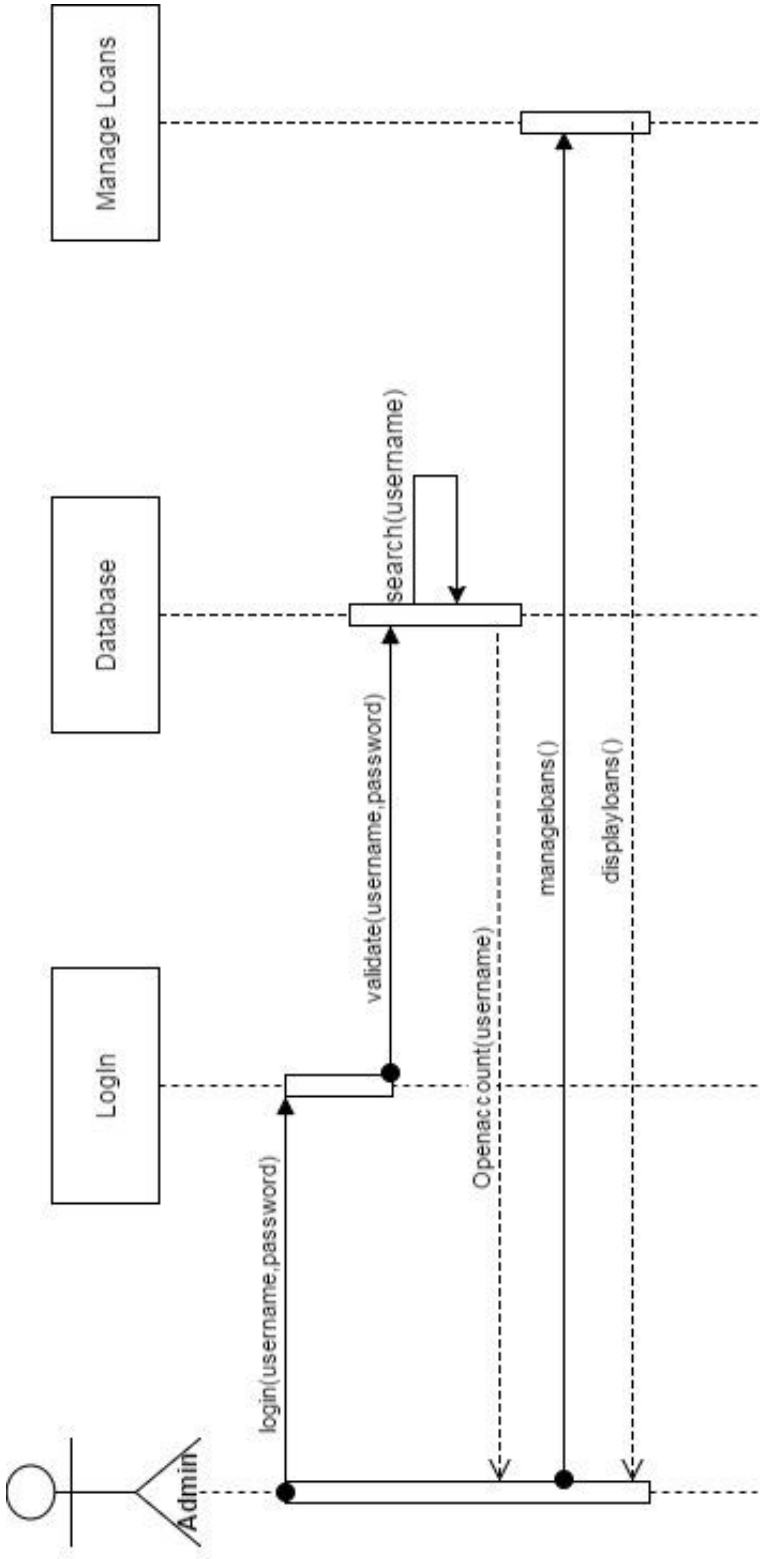
(SD_03) SEQUENCE DIAGRAM OF UC_03



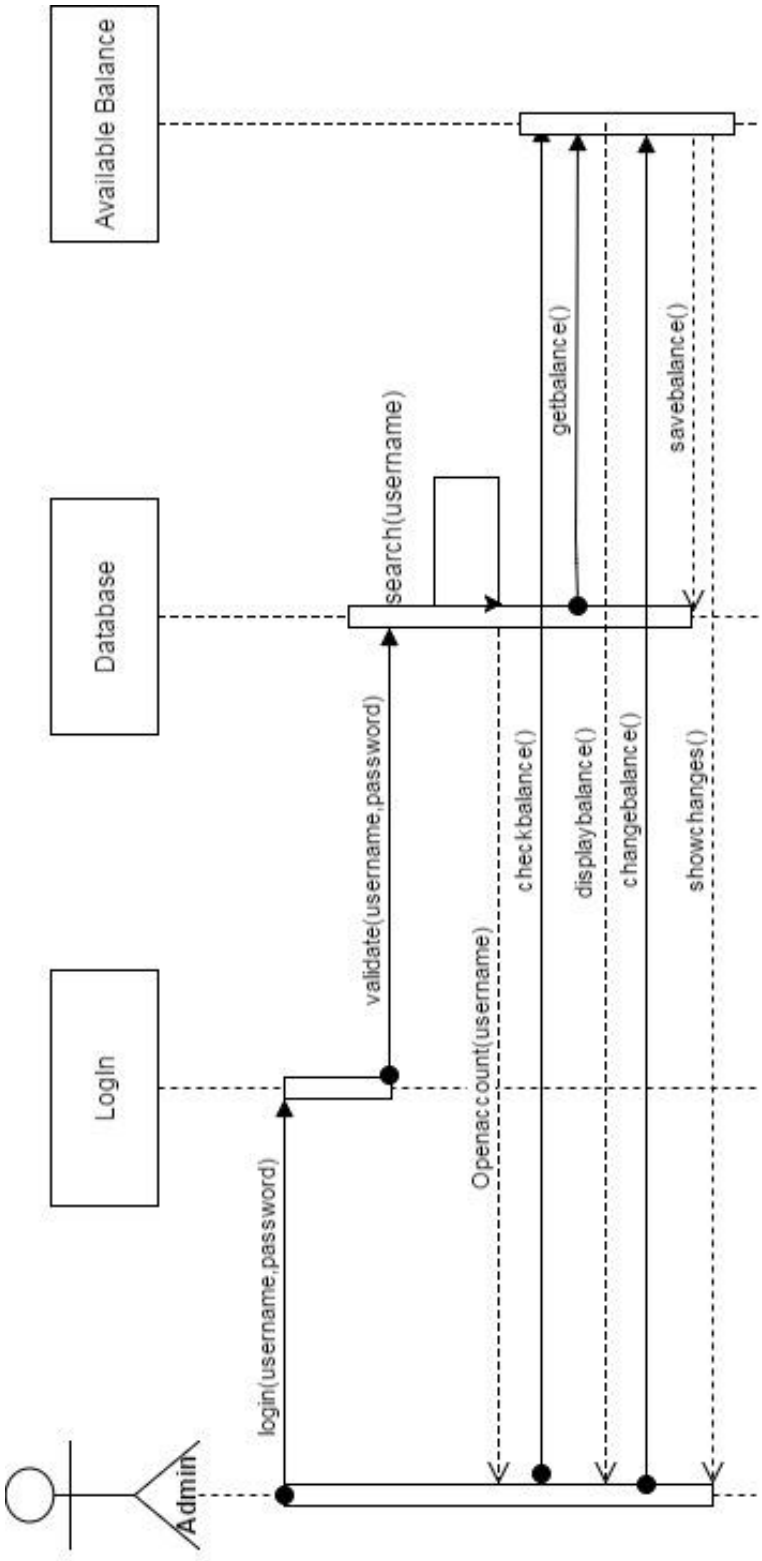
(SD_04) SEQUENCE DIAGRAM OF UC_04



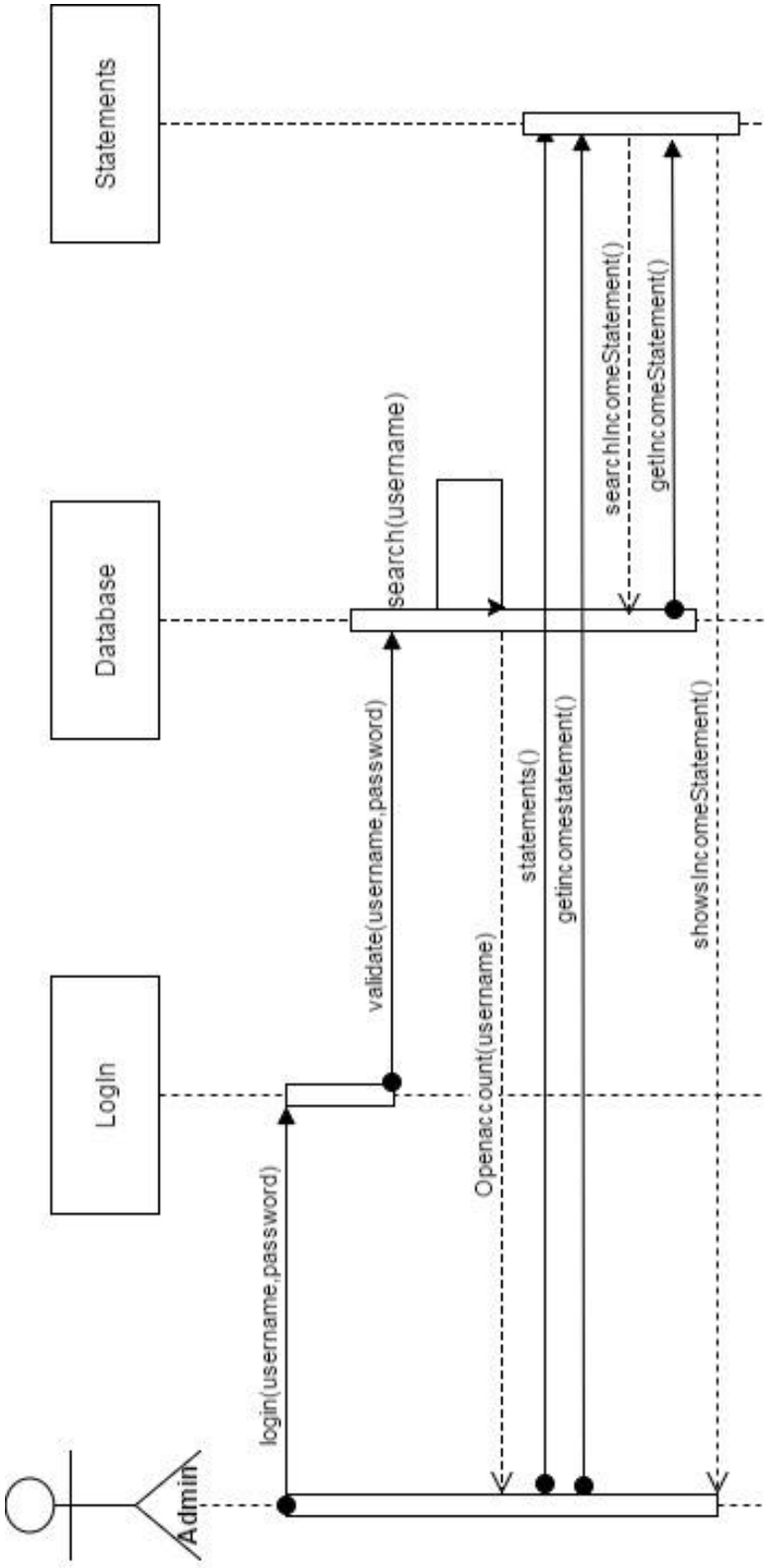
(SD_05) SEQUENCE DIAGRAM OF UC_05



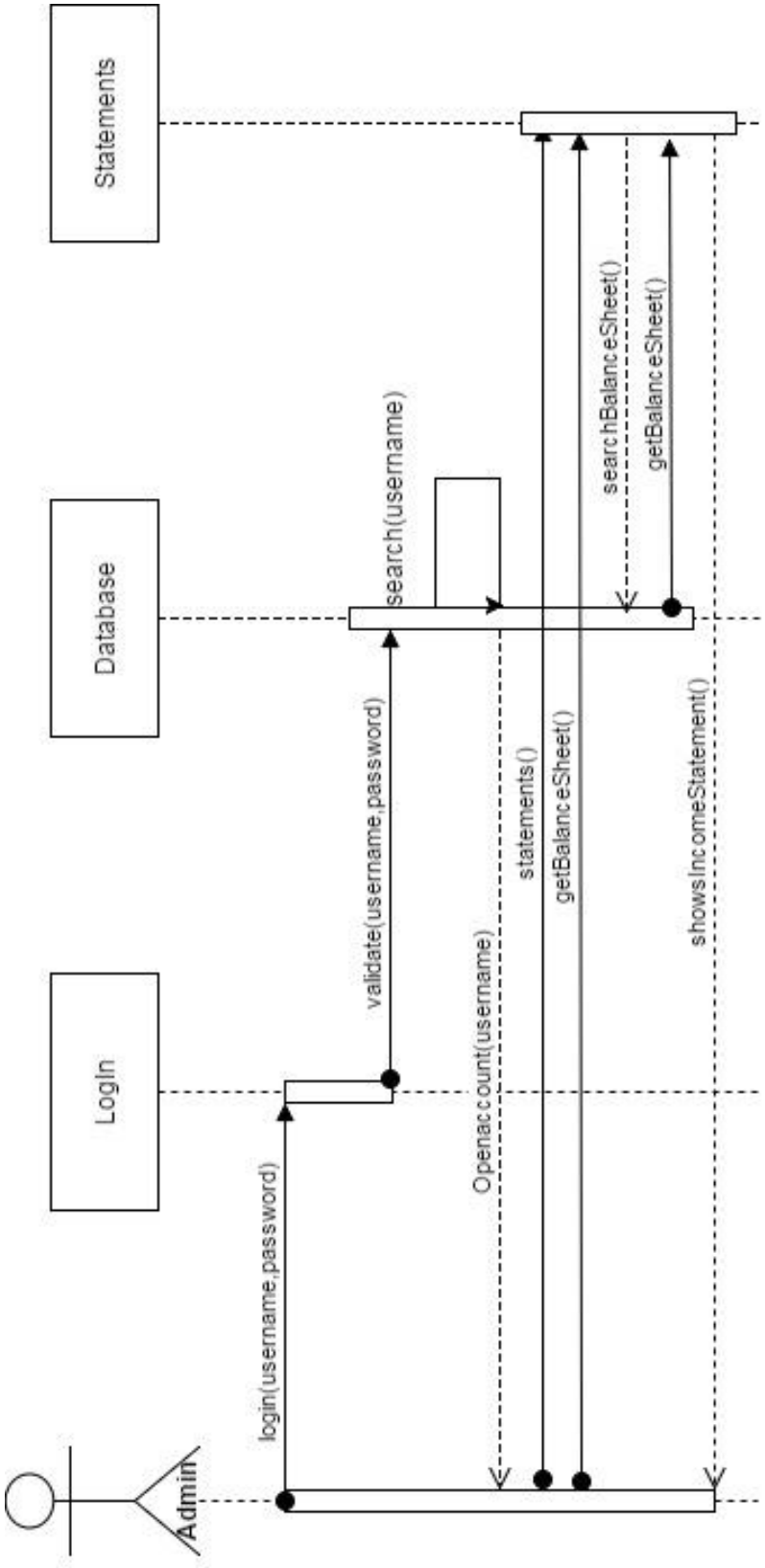
(SD_06) SEQUENCE DIAGRAM OF UC_06



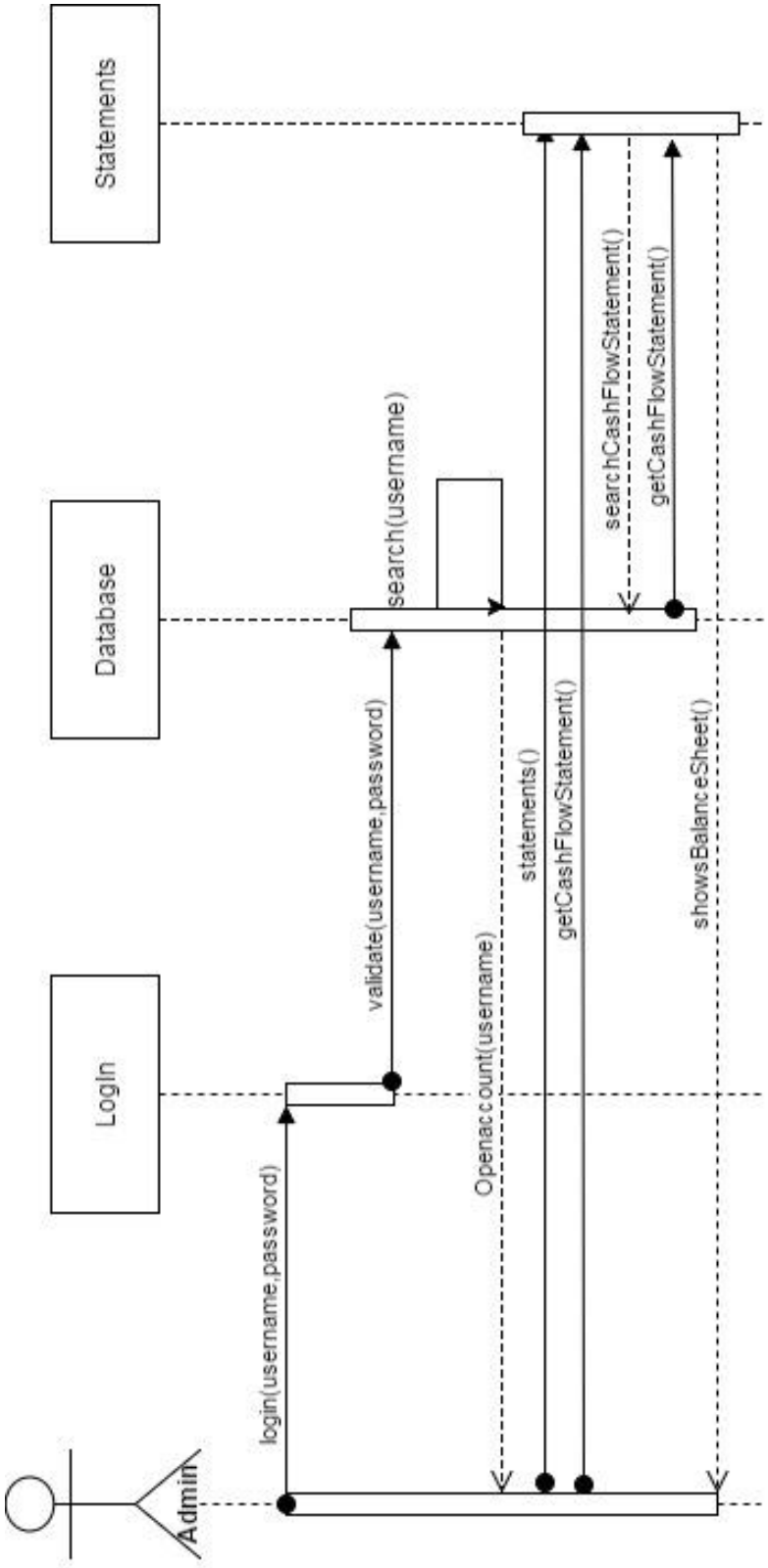
(SD_07) SEQUENCE DIAGRAM OF UC_07



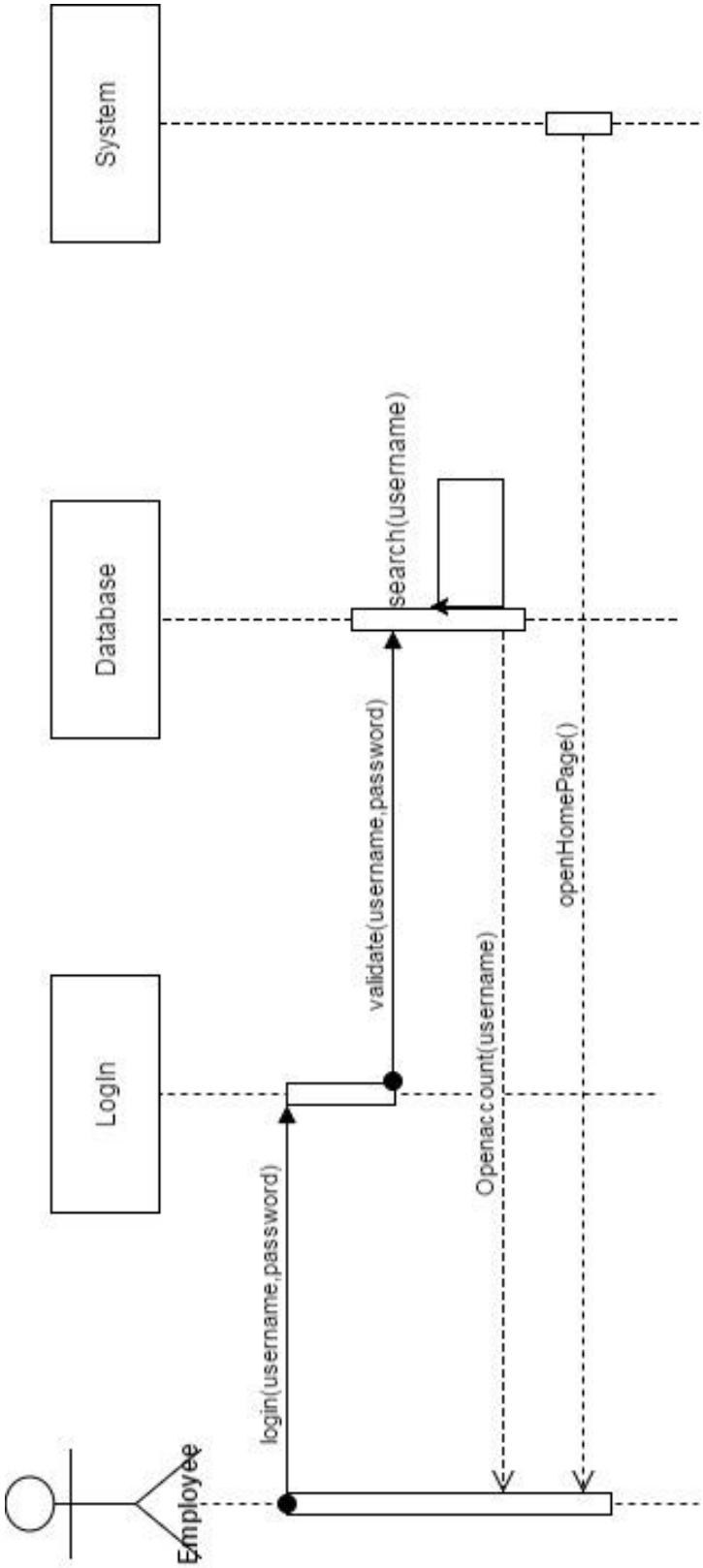
(SD_08) SEQUENCE DIAGRAM OF UC_08



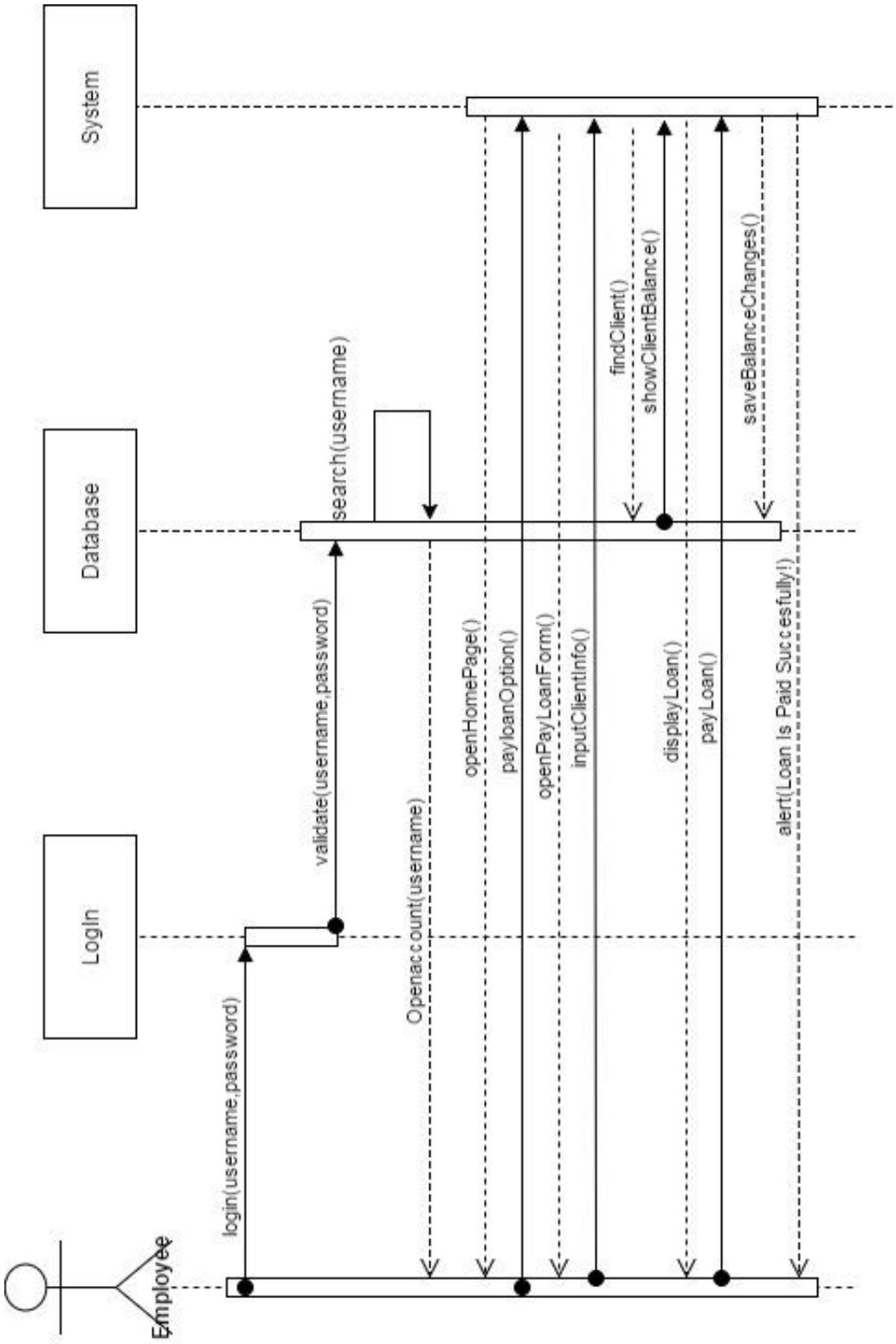
(SD_09) SEQUENCE DIAGRAM OF UC_09



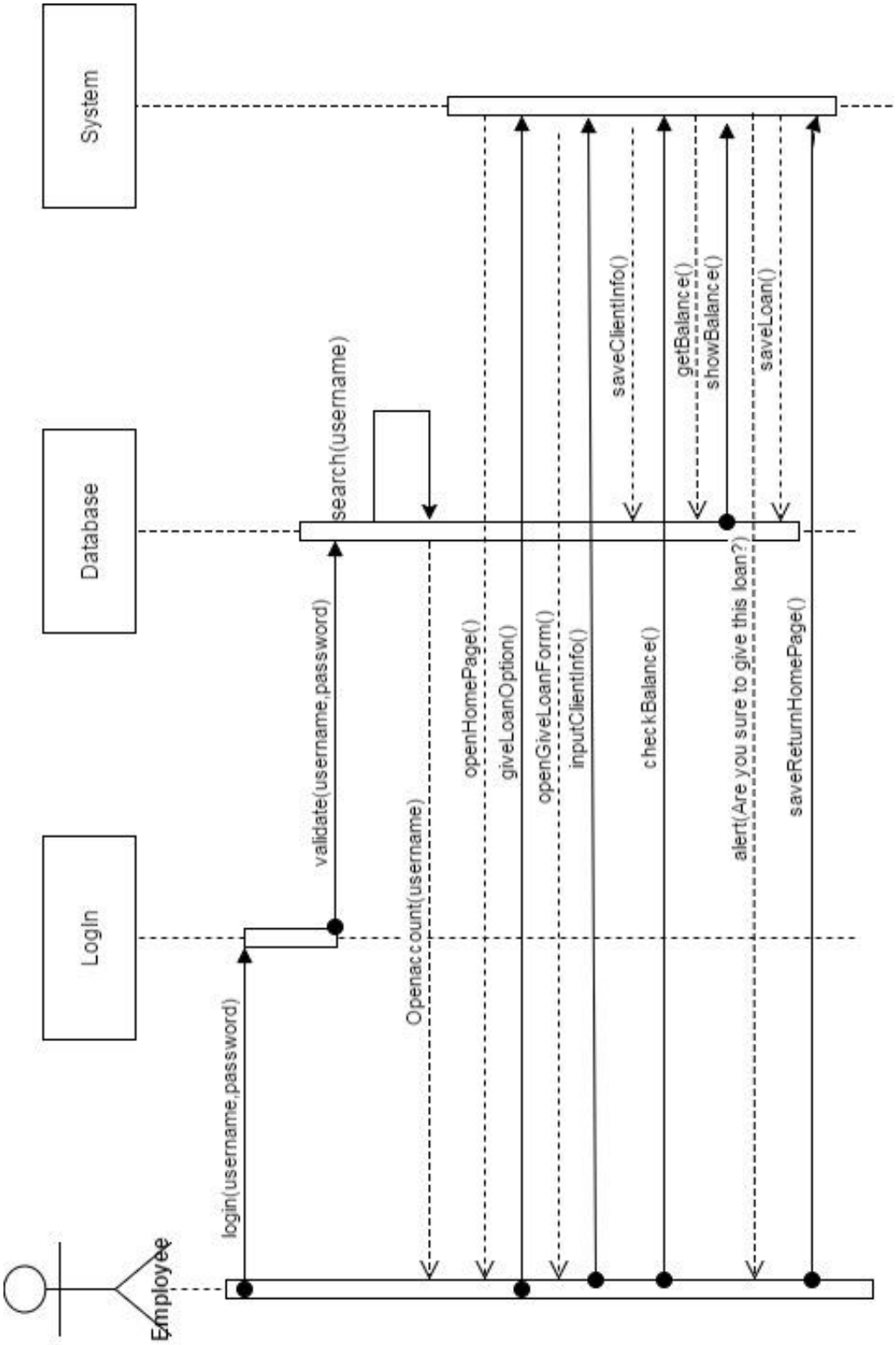
(SD_10) SEQUENCE DIAGRAM OF UC_10



(SD_11) SEQUENCE DIAGRAM OF UC_11

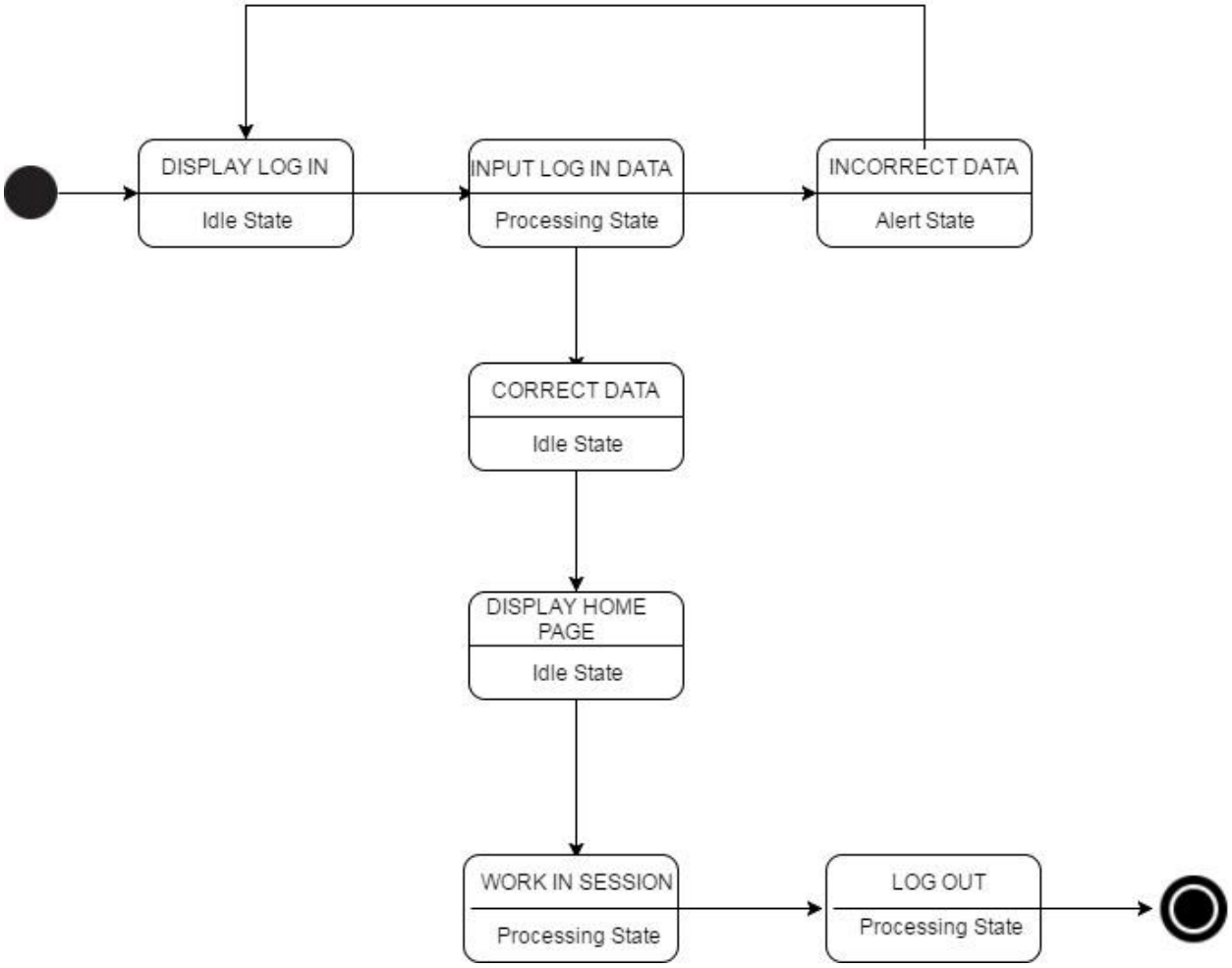


(SD_12) SEQUENCE DIAGRAM OF UC_12

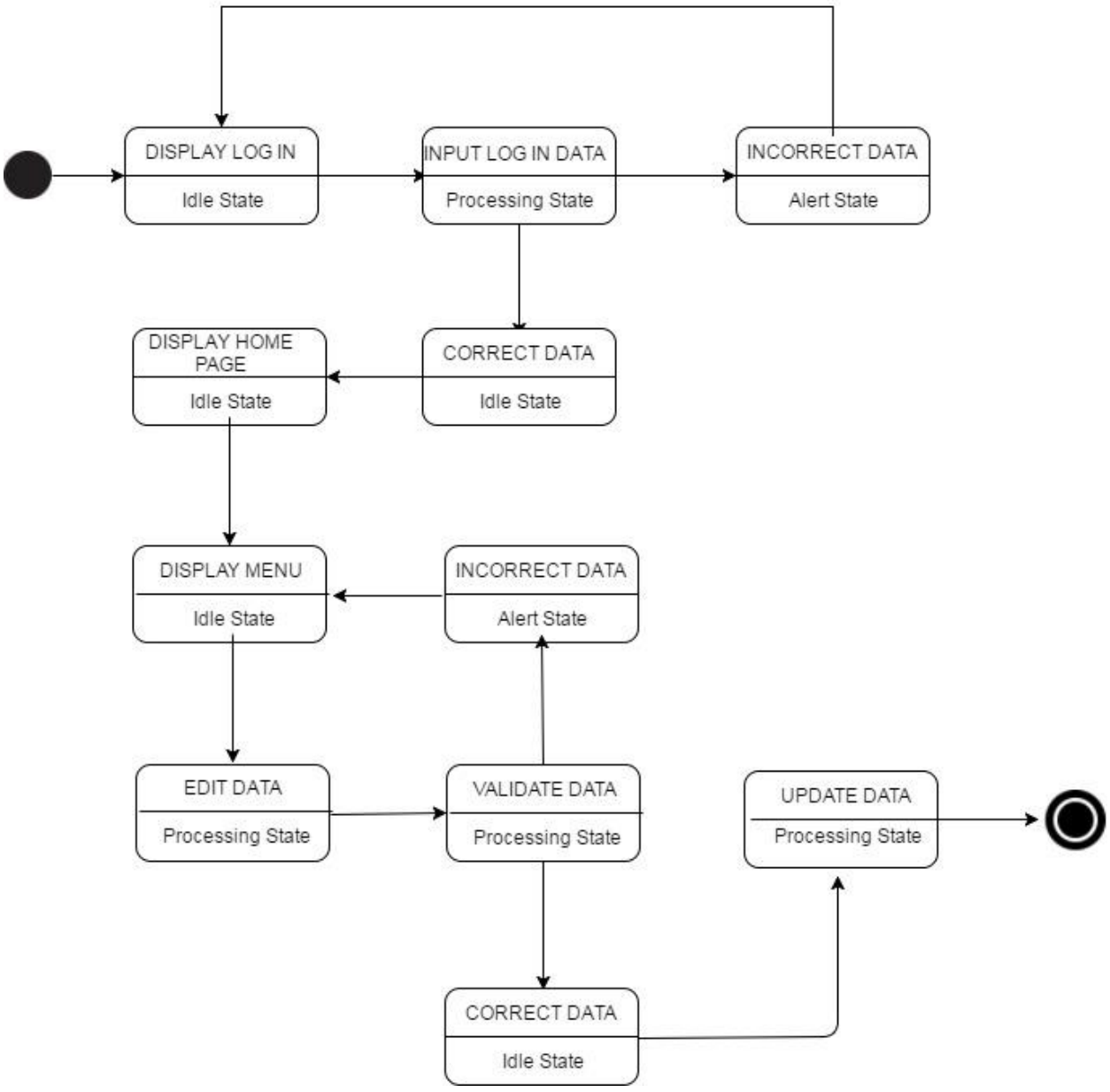


STATE DIAGRAM

STATE DIAGRAM 1 (STD_01)



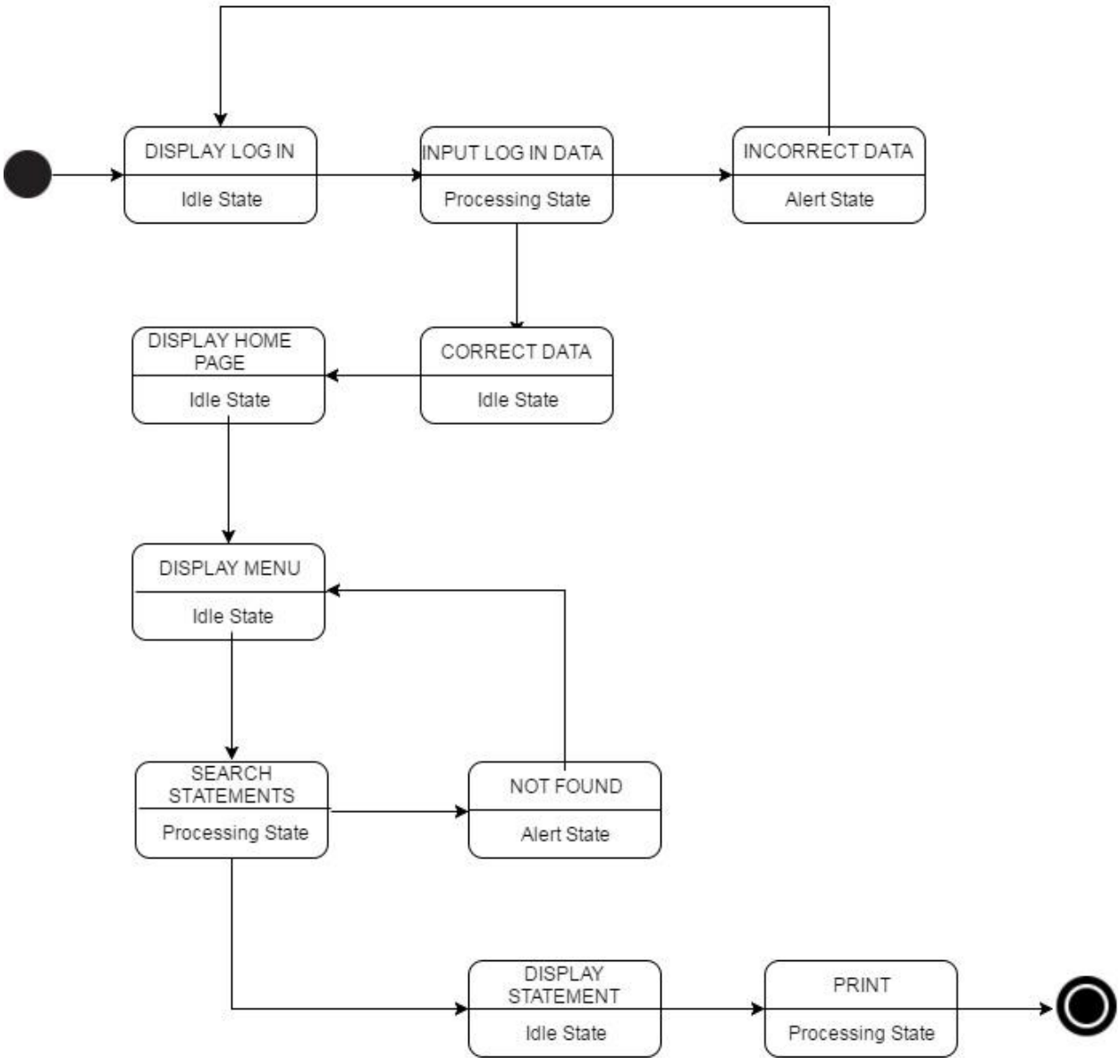
STATE DIAGRAM 2 (STD_02)



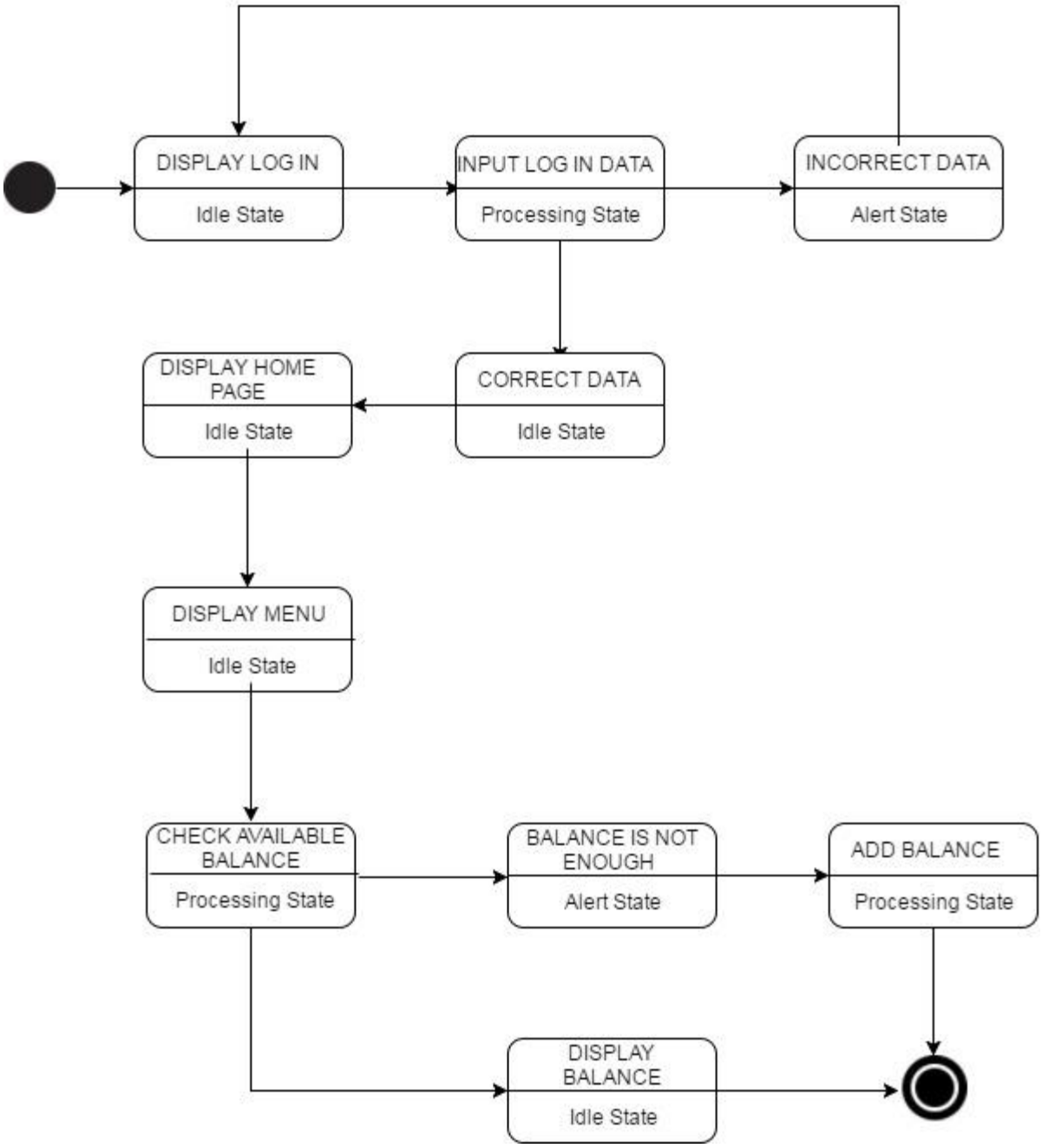
STATE DIAGRAM 3 (STD_03)

STATE DIAGRAM 4 (STD_04)

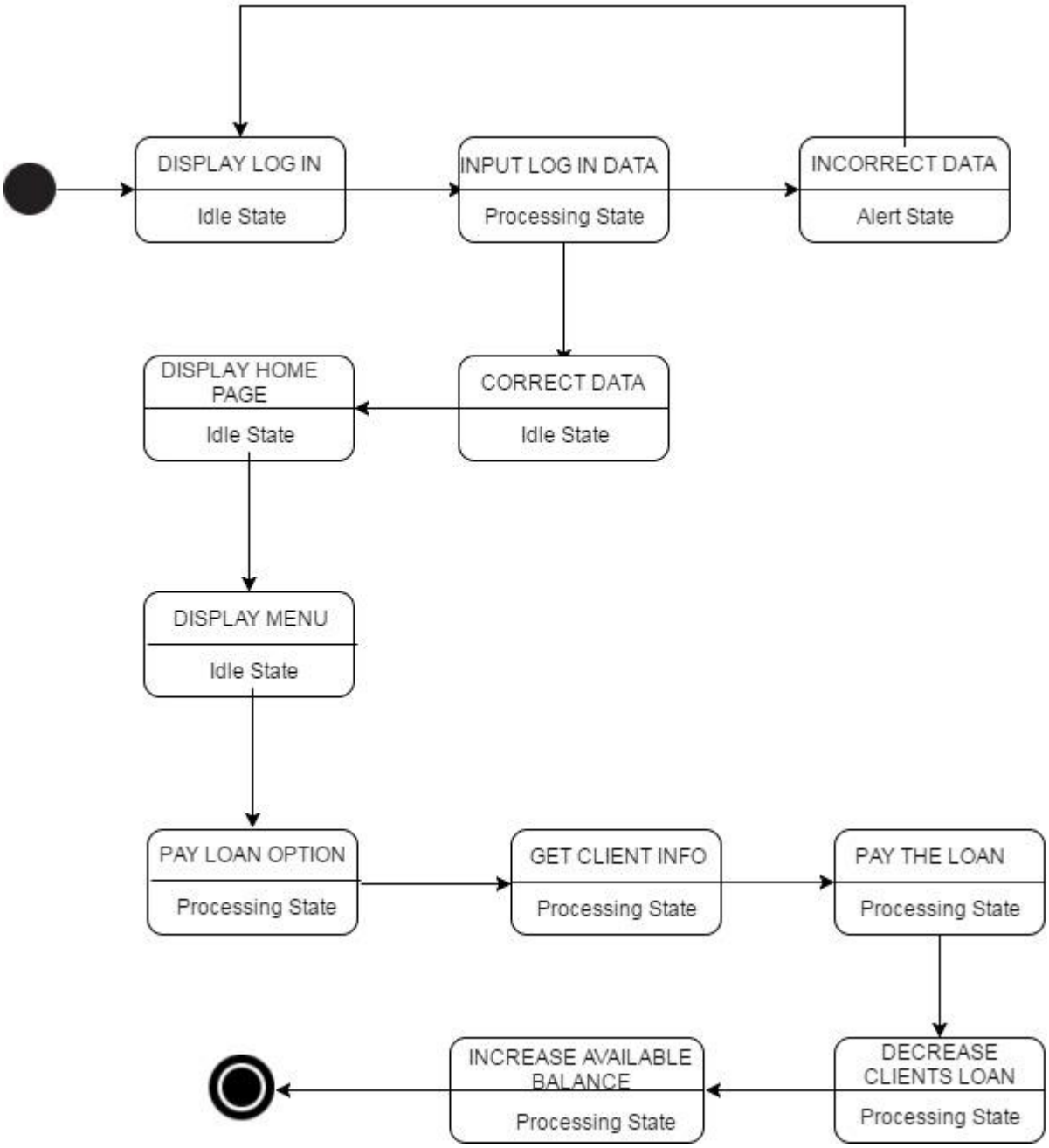
STATE DIAGRAM 5 (STD_05)



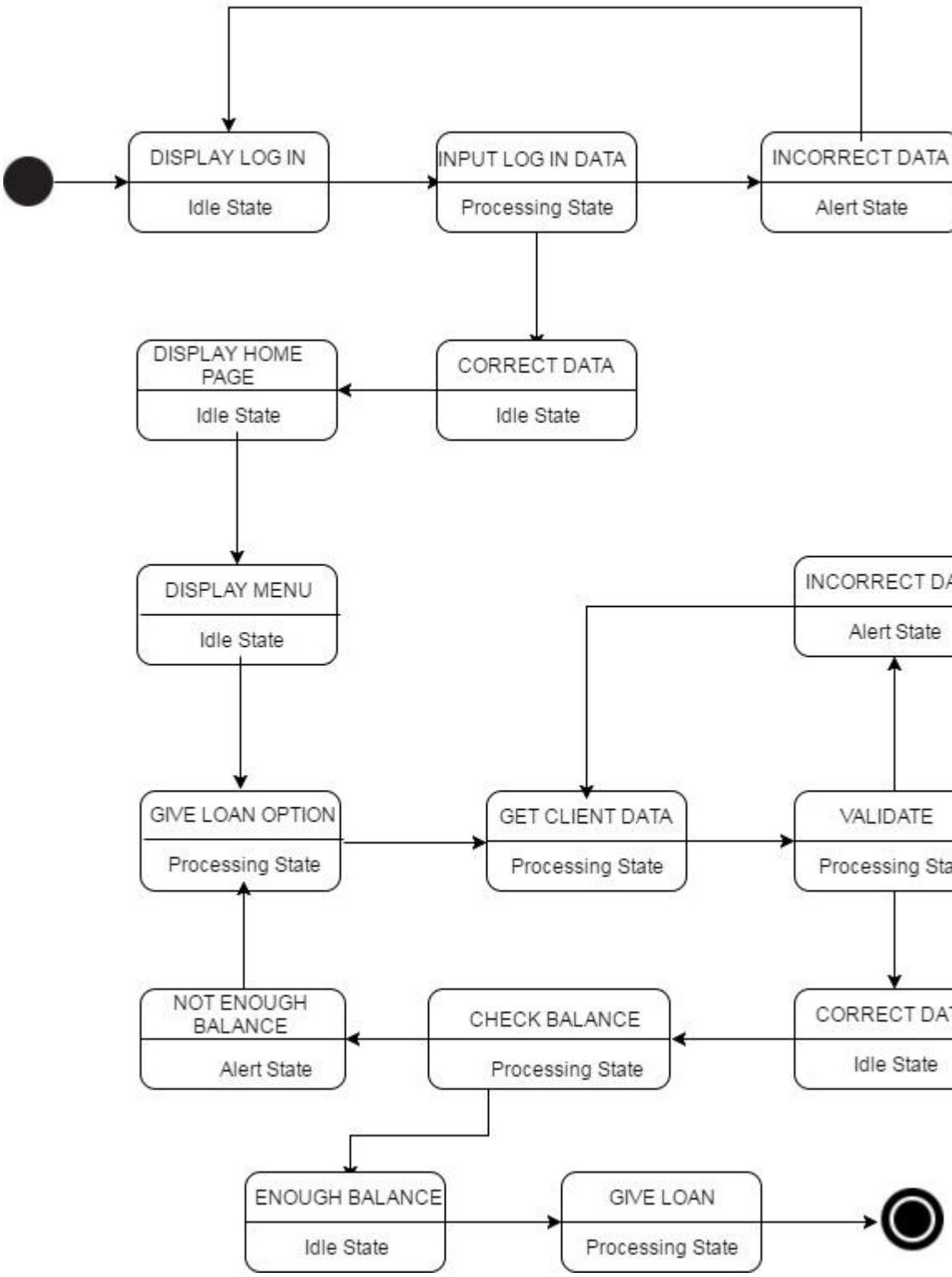
STATE DIAGRAM 6 (STD_06)



STATE DIAGRAM 7 (STD_07)

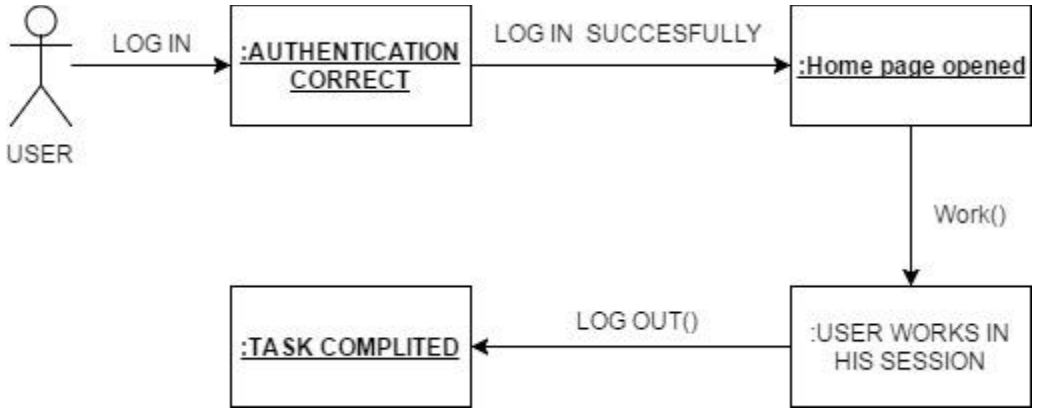


STATE DIAGRAM 8 (STD_08)

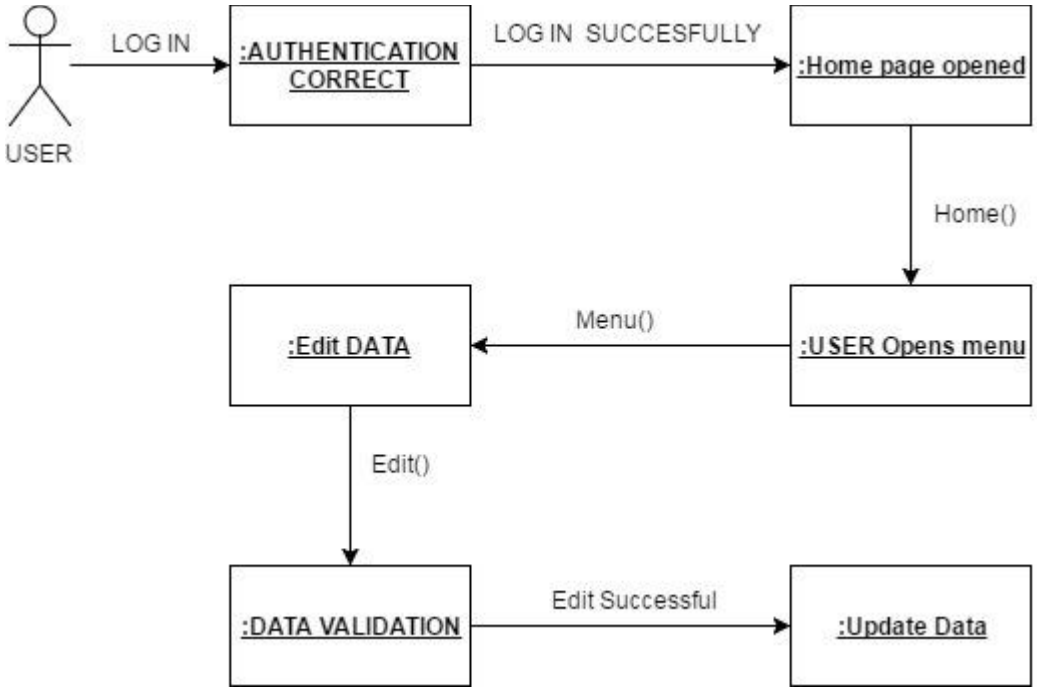


COLLABORATION DIAGRAM

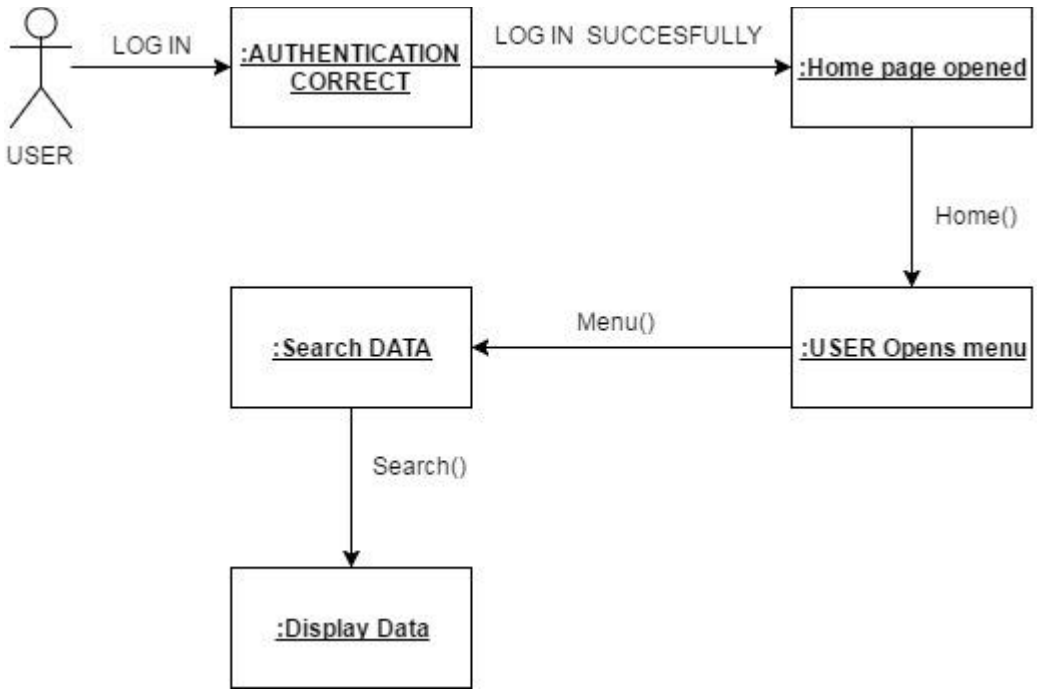
COLLABORATION DIAGRAM 1 (CD_01)



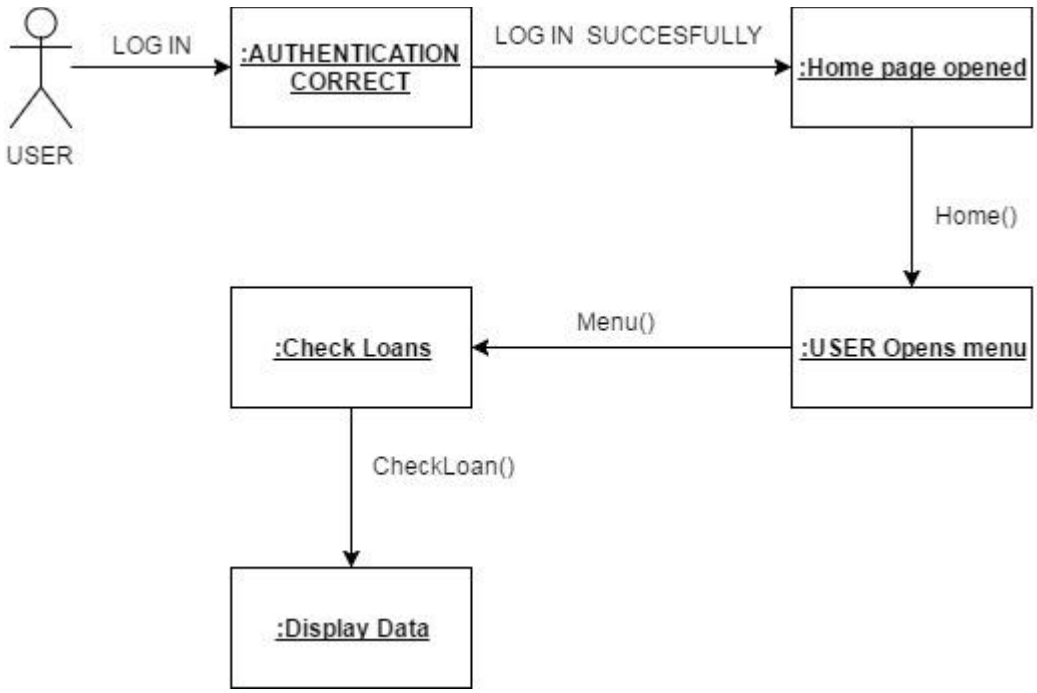
COLLABORATION DIAGRAM 2 (CD_02)



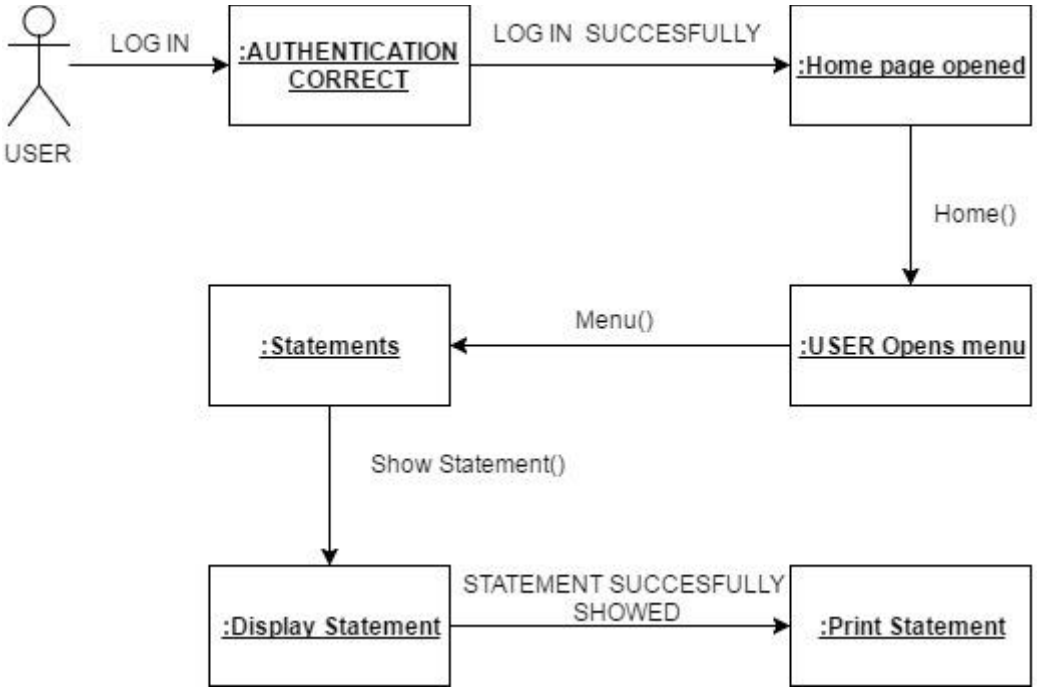
COLLABORATION DIAGRAM 3 (CD_03)



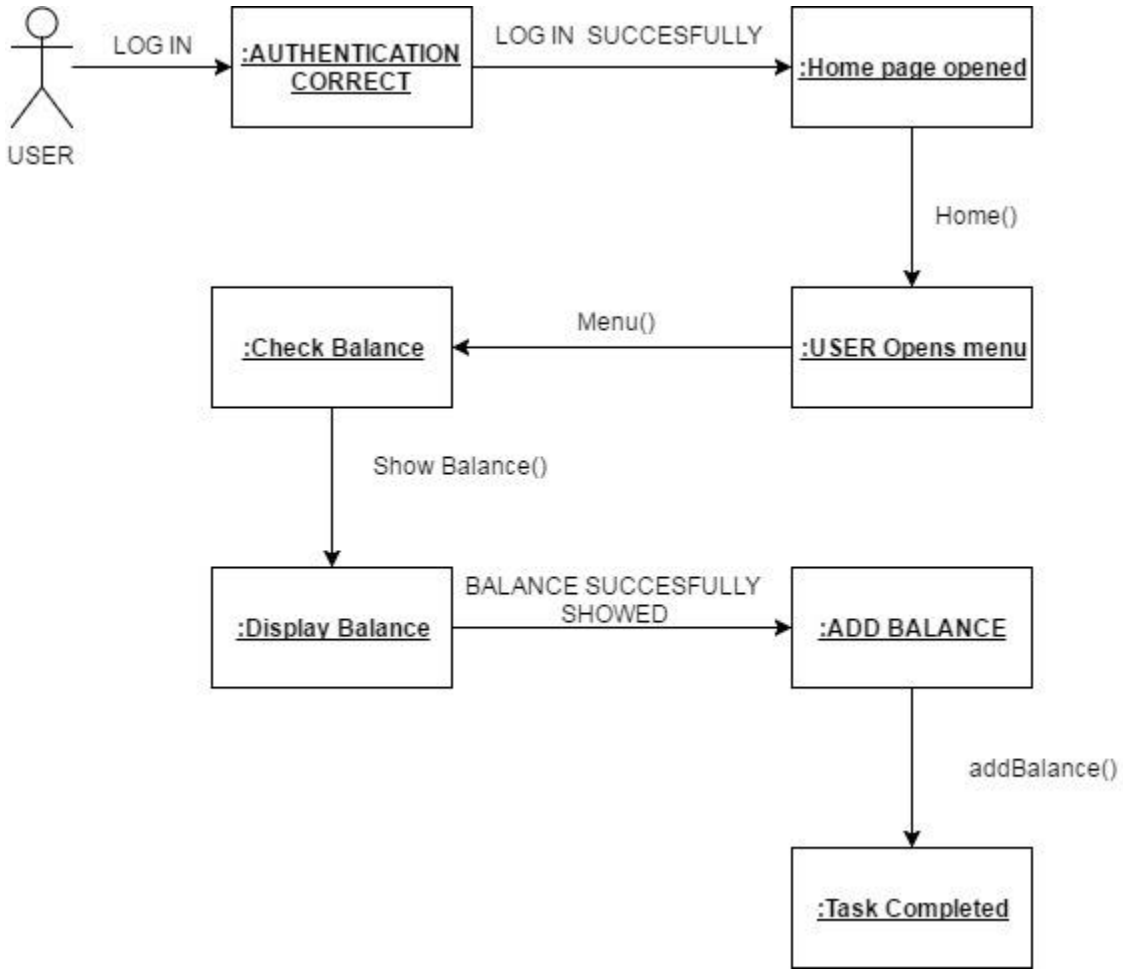
COLLABORATION DIAGRAM 4 (CD_04)



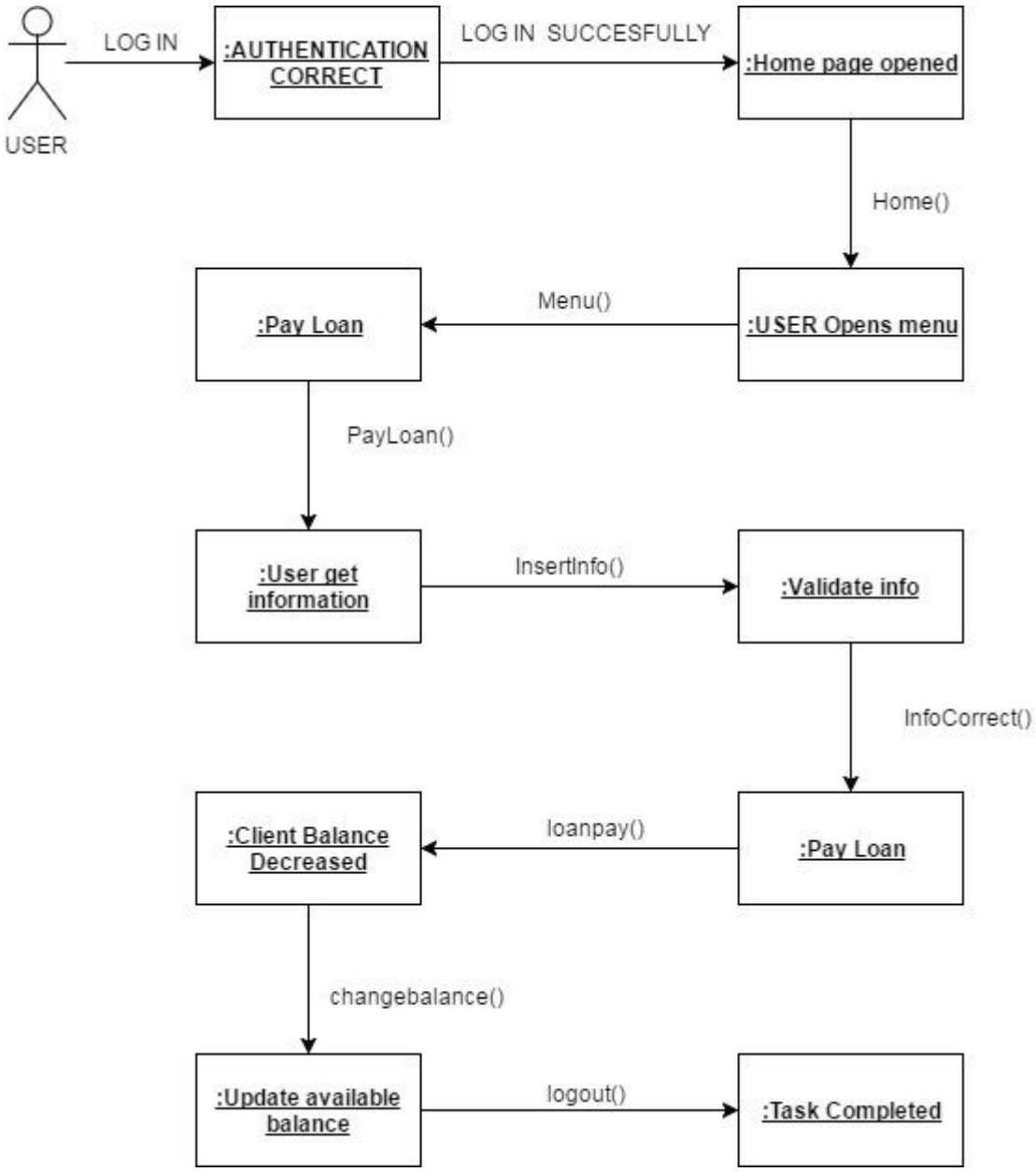
COLLABORATION DIAGRAM 5 (CD_05)



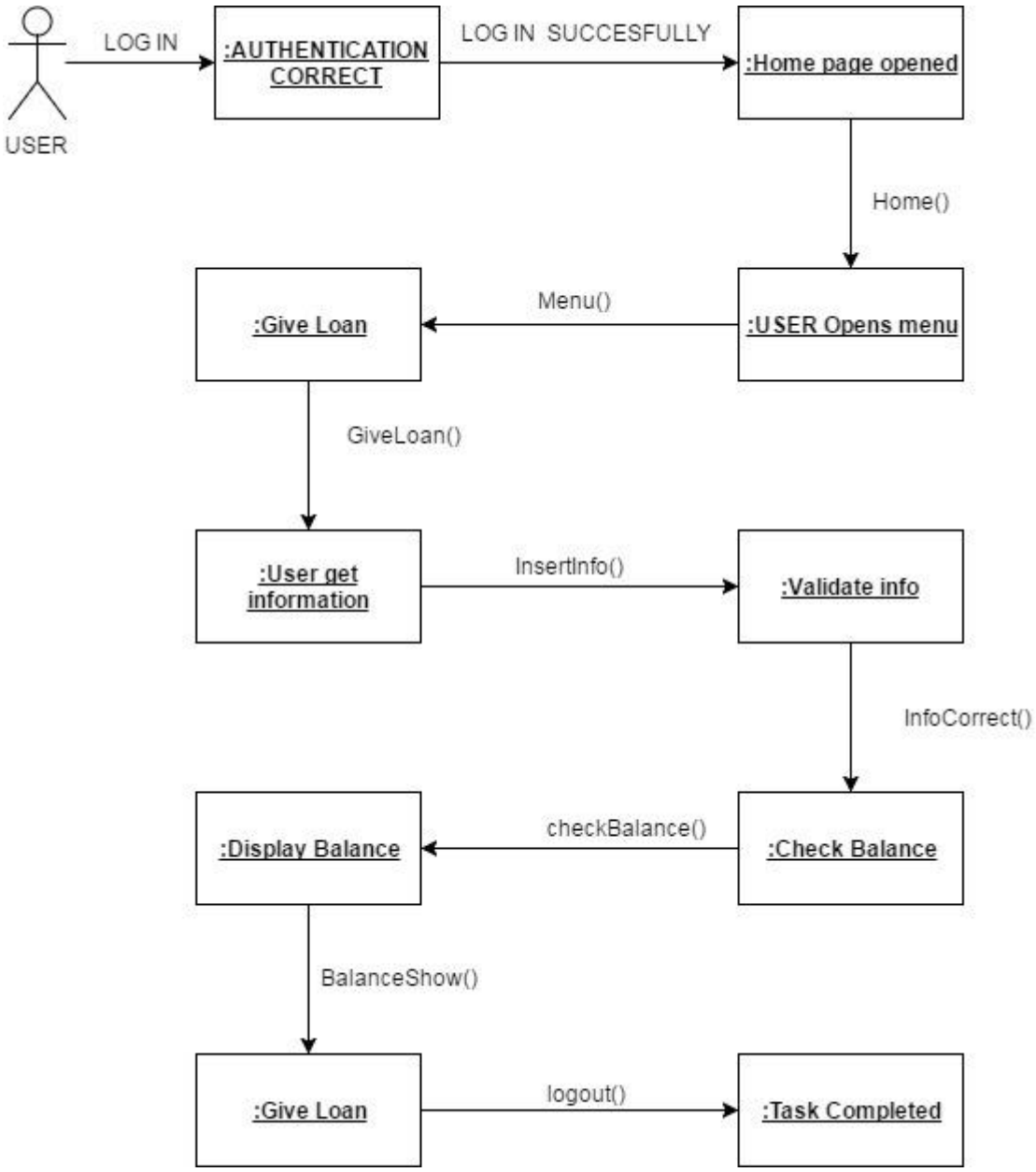
COLLABORATION DIAGRAM 6 (CD_06)



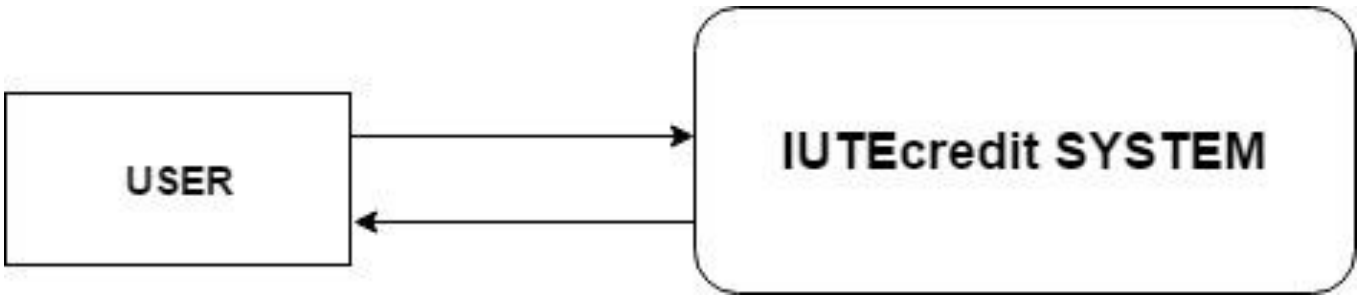
COLLABORATION DIAGRAM 7 (CD_07)



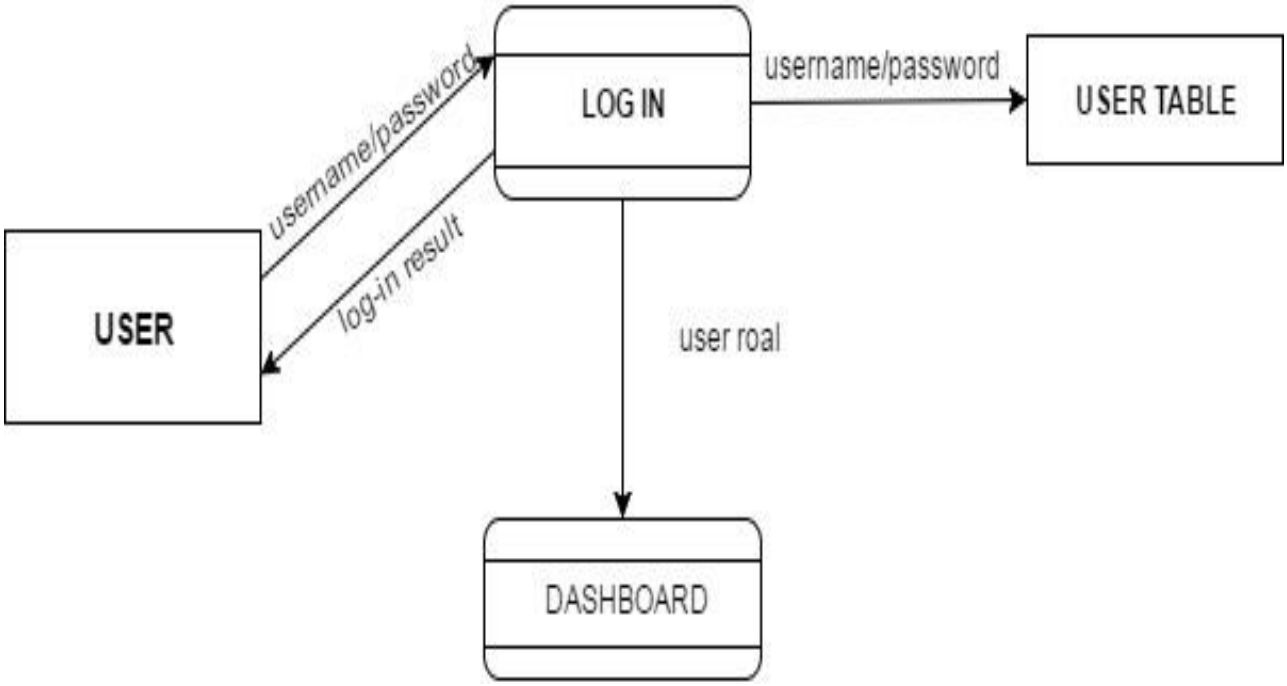
COLLABORATION DIAGRAM 8 (CD_08)



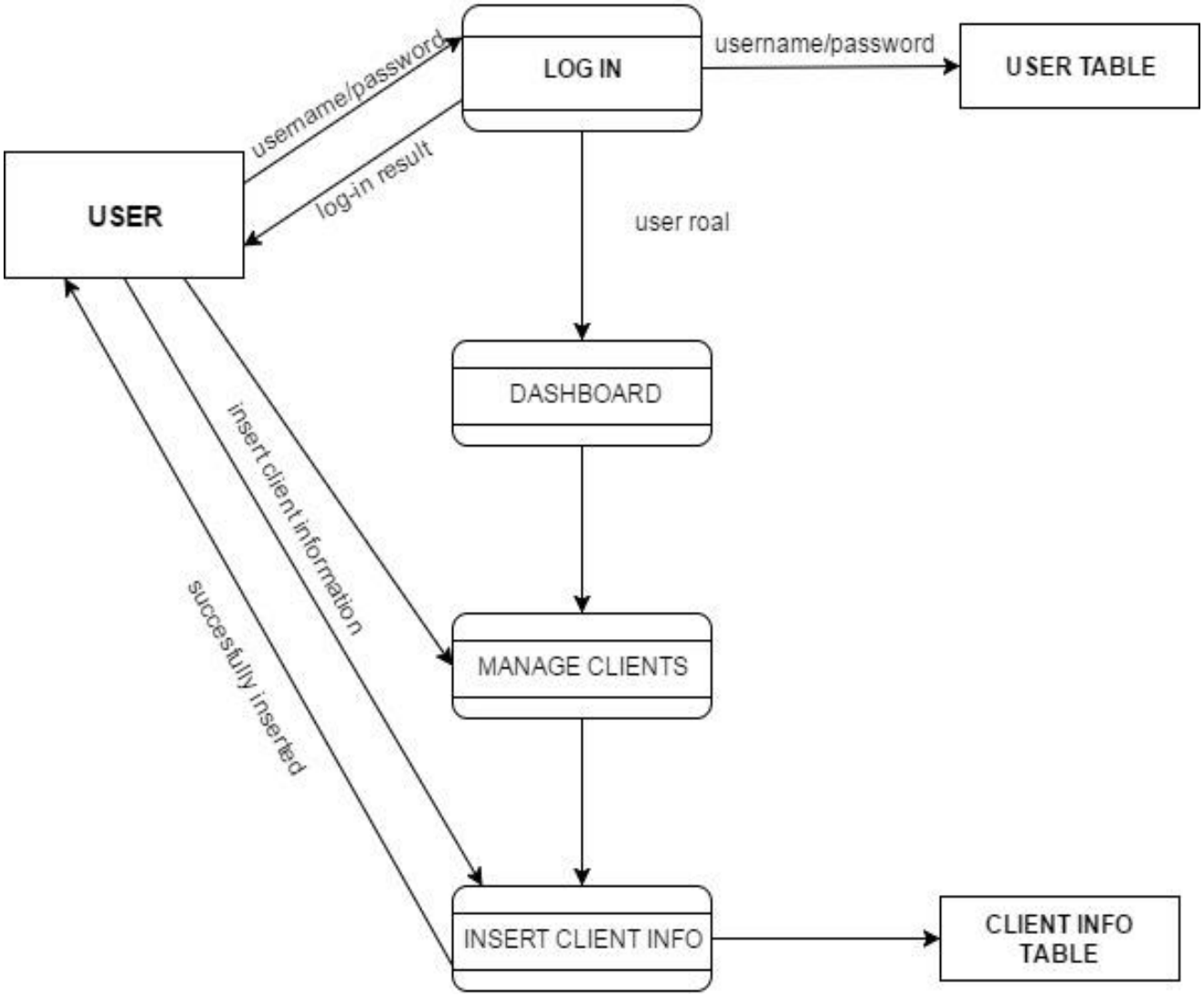
CONTEXT DATA-FLOW DIAGRAM



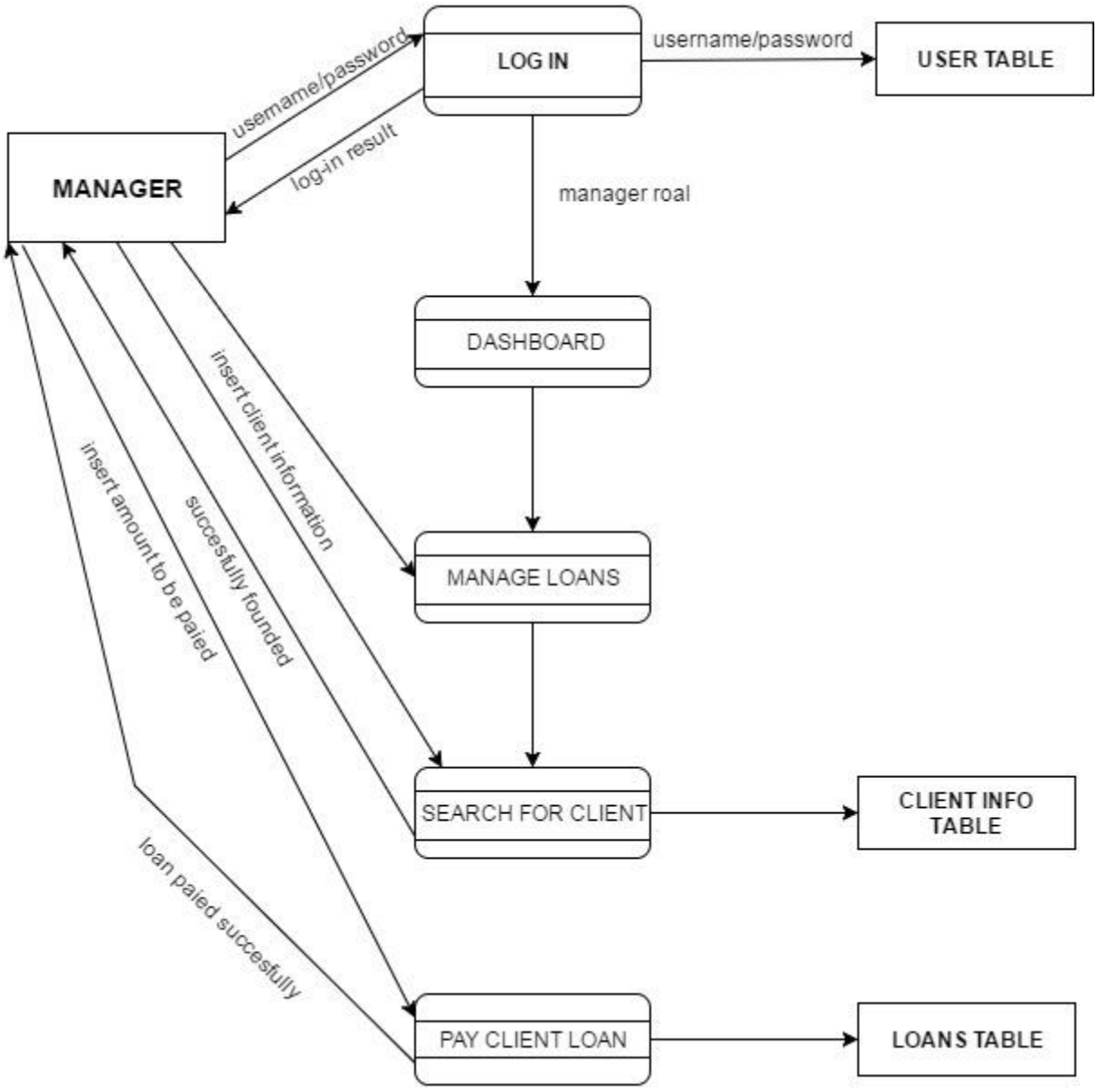
DATA-FLOW DIAGRAM 1



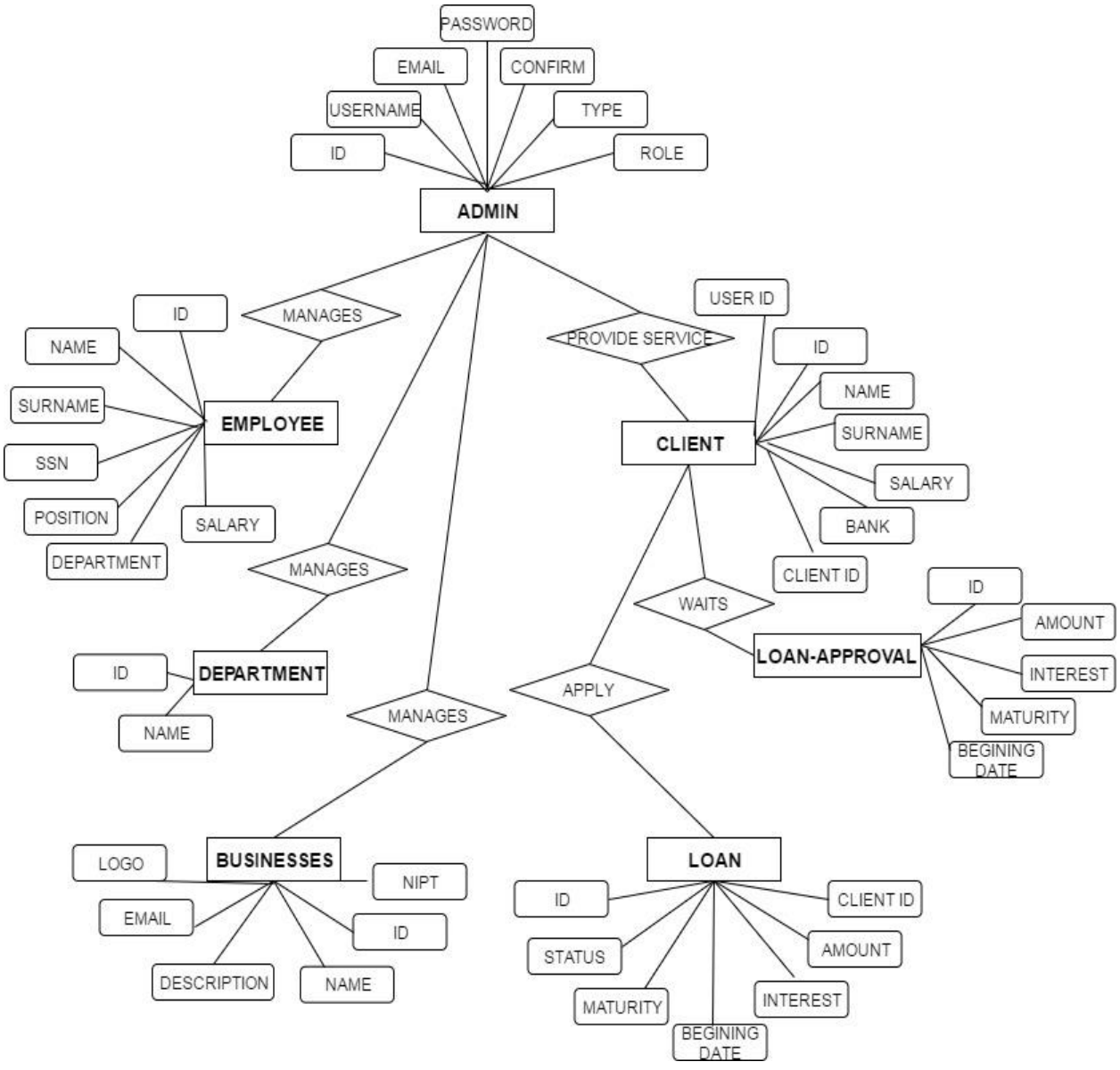
DATA-FLOW DIAGRAM 2



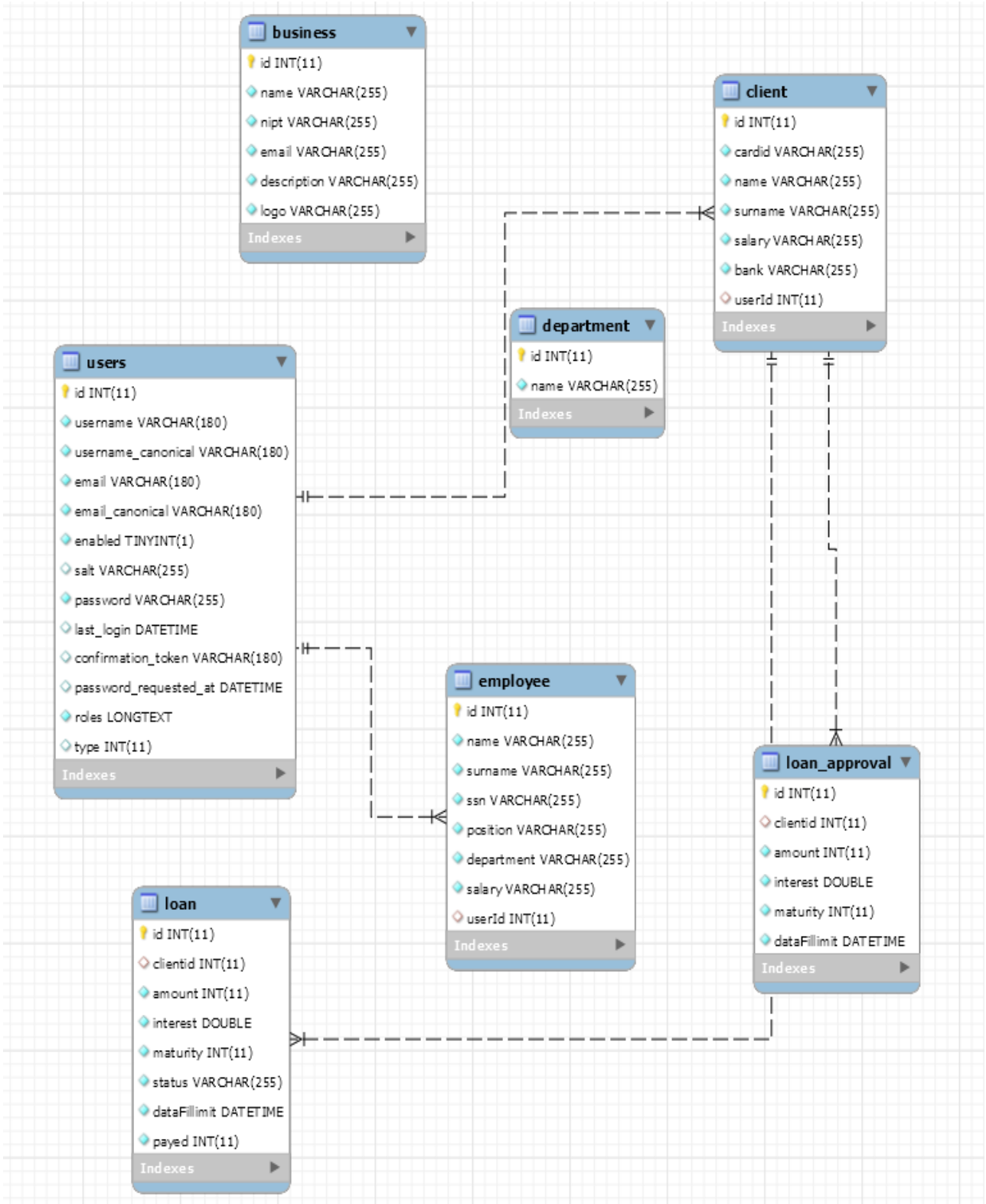
DATA-FLOW DIAGRAM 3



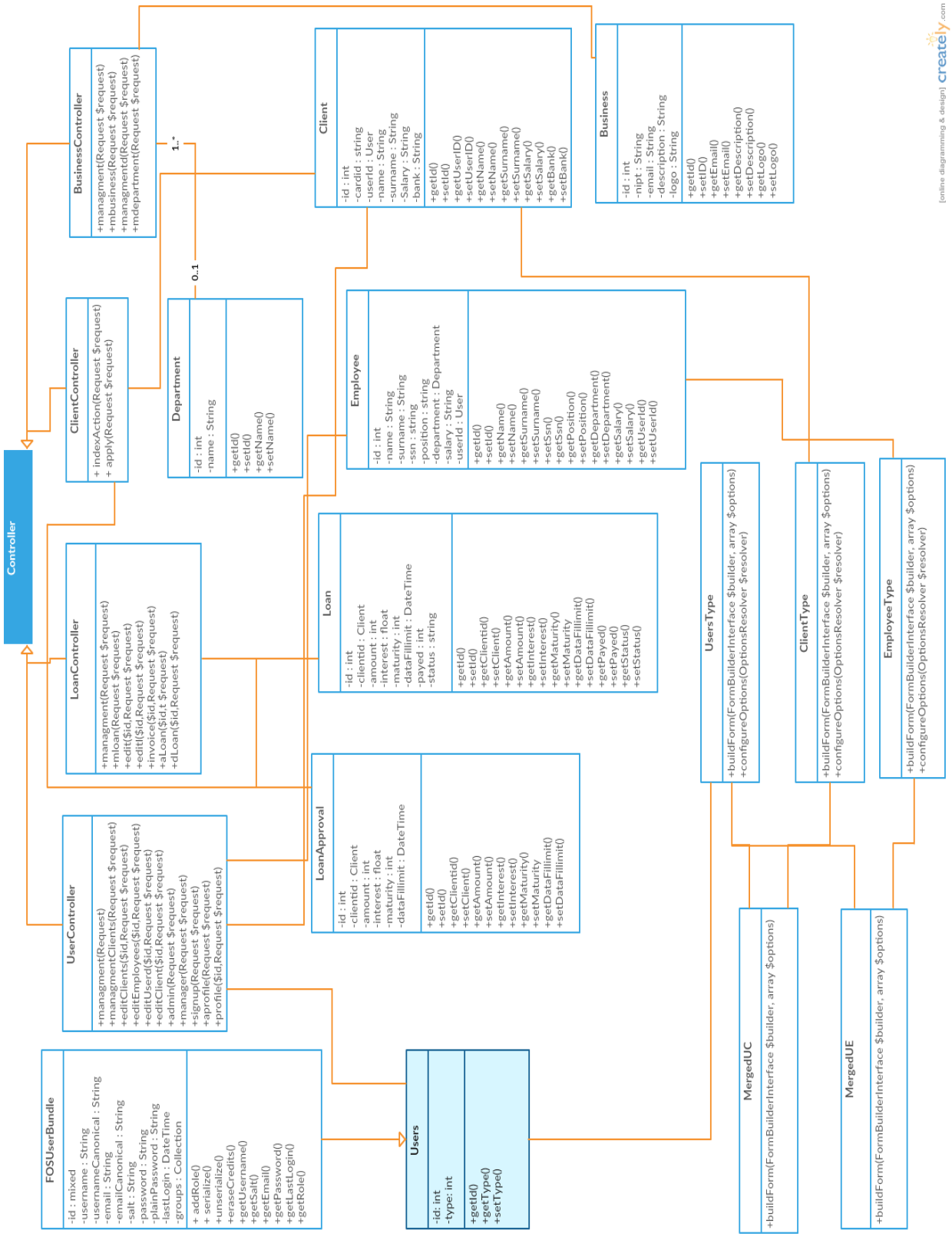
ENTITY RELATIONSHIP DIAGRAM



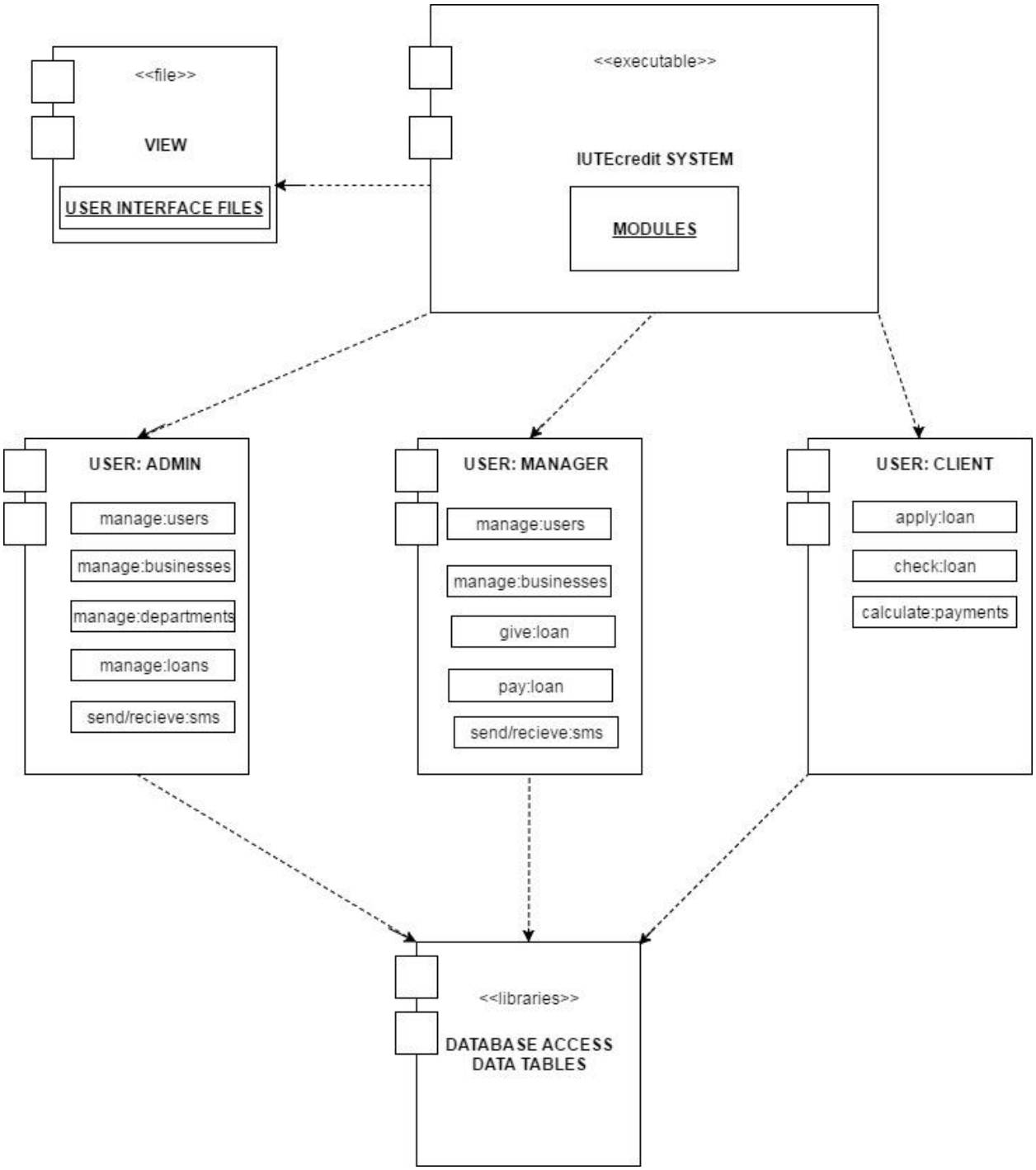
DATABASE SCHEMA



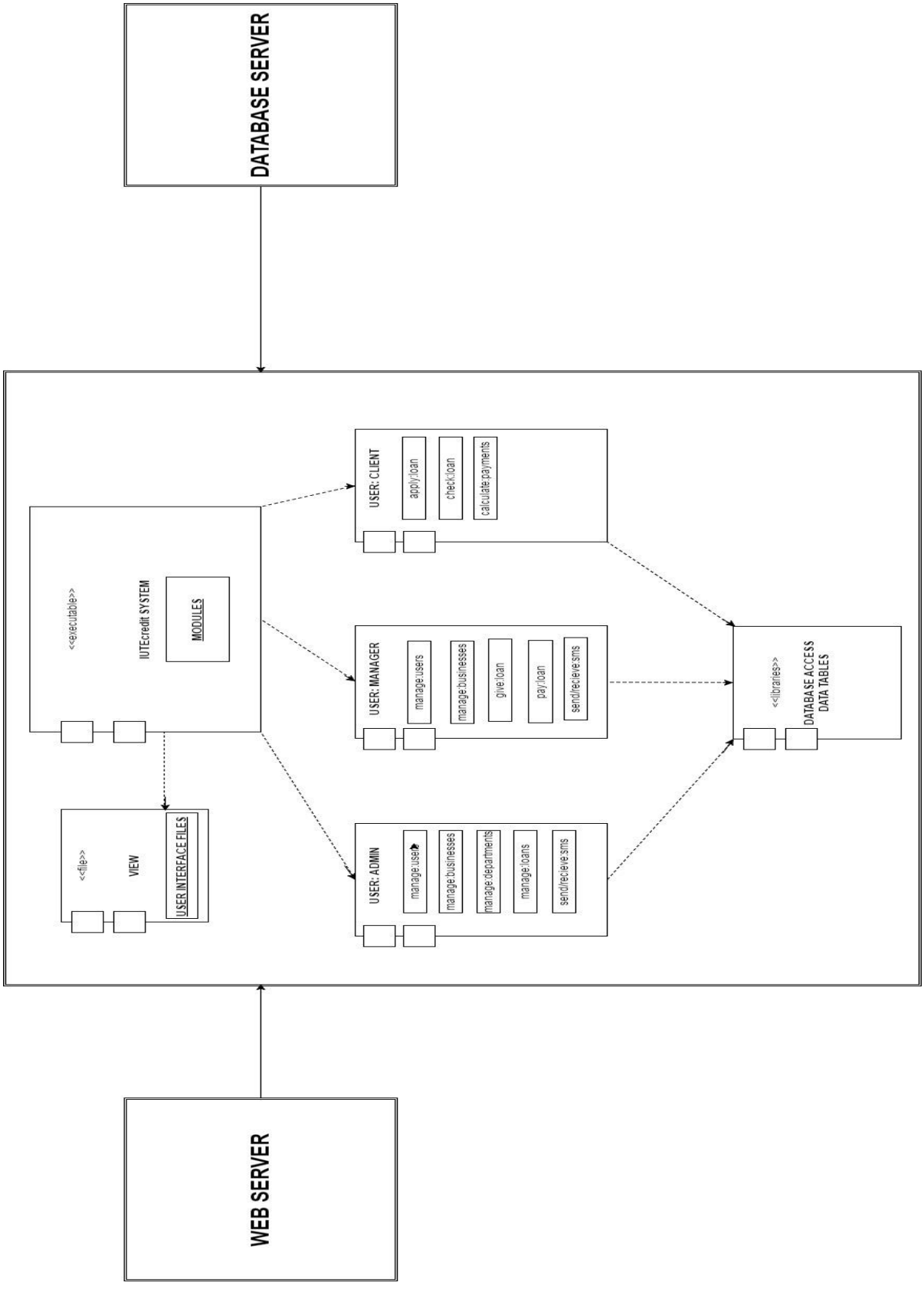
CLASS DIAGRAM



COMPONENT DIAGRAM



DEPLOYMENT DIAGRAM



Error 404

Woops. Looks like this page doesn't exist.

Software Testing

The framework that is used to build the web application for IUTE Company is PHP Symfony 3.14. Using this framework it was more efficient and reliable to test our product by using the testing classes. Testing cla sses are simulated in the way that will help in testing all the functionalities and avoiding the possible bugs that can happen during the execution of the code.

Symfony is one of the best frameworks available to build large systems. We choose this framework because:

1. Reputation. Symfony today is a stable environment that is both well-known and recognized internationally. The number of its references attests to this, as they have grown significantly since its launch.

2. Permanence. Designed by professionals for professionals, Symfony is first and foremost a pragmatic tool, the features of which address real-world-requirements. Permanence is also something that relates to long-term support. Today, this support is naturally provided by SensioLabs. Lastly, it is also with a view towards sustainable development that Symfony is distributed under Open Source MIT license, which does not impose constraints and allows the development of Open Source as well as proprietary applications.

3. References. Intranets, major general public sites, social networks, community sites, management and workflow applications, etc. Examples are not lacking: Hundreds of sites and applications of all sizes and of all types trust Symfony.

4. Innovation. Symfony is everything that you would come to expect from a framework: speed, flexibility, reusable components, etc. Then there is the structure of what has been developed and the use of best practices.

5. Resources. When using Symfony, you are assured of never “being alone with your screen.” Whether a question of community support (mailings lists, IRC, etc.) or company support (consulting, training, etc.), you will always find the answers to your questions.

6. Security. Symfony provides extra security when it comes to any kind of injection a system can get. Starting from forms to database attacks, symfony is the best choice in the market

These are the reasons why there is no need to test the system for bugs.

General testing is made below:

User MVC

Model:

```
<?php

/**
 * Created by PhpStorm.
 * User: xhulio
 * Date: 5/21/2017
 * Time: 9:55 PM
 */

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use FOS\UserBundle\Model\User as BaseUser;

/**
 * Users
 *
 * @ORM\Table(name="users")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\UsersRepository")
 */
class Users extends BaseUser
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var int
     *
     * @ORM\Column(name="type", type="integer")
```

```
     */
    private $type;


    /**
     * Get id
     *
     * @return int
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set type
     *
     * @param integer $type
     *
     * @return Users
     */
    public function setType($type)
    {
        $this->type = $type;

        return $this;
    }

    /**
     * Get type
     *
     * @return int
     */
    public function getType()
    {
        return $this->type;
    }
}
```

Controller:

```
<?php
/**
```

```php
 * Created by PhpStorm.
 * User: xhulio
 * Date: 5/21/2017
 * Time: 9:55 PM
 */

namespace AppBundle\Controller;


use AppBundle\Entity\Client;
use AppBundle\Entity\Employee;
use AppBundle\Entity\Users;
use AppBundle\Form\MergedUC;
use AppBundle\Form\MergedUE;
use Symfony\Component\Form\Extension\Core\Type\ChoiceType;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Component\Form\Extension\Core\Type\PasswordType;
use Symfony\Component\Form\Extension\Core\Type\RepeatedType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Form\FormBuilder;

class UserController extends Controller
{
    /**
     * @Route("/user_managment", name="users")
     */
    public function managment(Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()-
>getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')) {
                $e = new Employee();
                $u = new Users();
                $total = array(
                    'user' => $u,
                    'employee' => $e
```

```
        );
        $form = $this->createForm(MergedUE::class, $total);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $name = $form['employee']['name']->getData();
            $surname = $form['employee']['surname']->getData();
            $email = $form['user']['email']->getData();
            $username = $form['user']['username']->getData();
            $password = $form['user']['plainPassword']->getData();
            $type = 0;
            $ssn = $form['employee']['ssn']->getData();
            $department = $form['employee']['department']->getData();
            $position = $form['employee']['position']->getData();
            $salary = $form['employee']['salary']->getData();

            $total['employee']->setName($name);
            $total['user']->setEmail($email);
            $total['user']->setUsername($username);
            $total['user']->setPlainPassword($password);
            $total['user']->setType($type);
            $total['user']->setEnabled(true);

            $total['employee']->setSurname($surname);
            $total['employee']->setSsn($ssn);
            $total['employee']->setDepartment($department);
            $total['employee']->setPosition($position);
            $total['employee']->setSalary($salary);

            $em = $this->getDoctrine()->getManager();
            $em->persist($total['user']);
            $em->flush();
            $total['employee']->setUserId($total['user']);
            $em->persist($total['employee']);
            $em->flush();
            return $this->redirectToRoute('users');

        }
        $c = new Client();
        $u1 = new Users();
        $total1 = array(
            'user' => $u1,
```

```
                'client' => $c
            );
            $form1 = $this->createForm(MergedUC::class, $total1);
            $form1->handleRequest($request);
            if ($form1->isSubmitted() && $form1->isValid()) {
                $name = $form1['client']['name']->getData();
                $surname = $form1['client']['surname']->getData();
                $email = $form1['user']['email']->getData();
                $username = $form1['user']['username']->getData();
                $password = $form1['user']['plainPassword']->getData();
                $type = 1;
                $bank = $form1['client']['bank']->getData();
                $cardid = $form1['client']['cardid']->getData();
                $salary = $form1['client']['salary']->getData();

                $total1['client']->setName($name);
                $total1['user']->setEmail($email);
                $total1['user']->setUsername($username);
                $total1['user']->setPlainPassword($password);
                $total1['user']->setType($type);
                $total1['client']->setSurname($surname);
                $total1['client']->setBank($bank);
                $total1['client']->setCardid($cardid);
                $total1['client']->setSalary($salary);
                $em = $this->getDoctrine()->getManager();
                $em->persist($total1['user']);
                $em->flush();
                $total['client']->setUserId($total['user']);
                $em->persist($total1['client']);
                $em->flush();
                return $this->redirectToRoute('users');

            }
            $users = $this->getDoctrine()
                ->getRepository('AppBundle:Users')
                ->findAll();
            $employees = $this->getDoctrine()
                ->getRepository('AppBundle:Employee')
                ->findAll();
            $clients = $this->getDoctrine()
                ->getRepository('AppBundle:Client')
                ->findAll();
```

```
                return $this->render('Admin/menaxhousers.html.twig', array(
                    'form' => $form->createView(),
                    'form2' => $form1->createView(),
                    'users' => $users,
                    'employees' => $employees,
                    'clients' => $clients,
                ));
            }else{
                return $this->redirectToRoute('ClientIndex');
            }
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/manager_client", name="clients")
     */
    public function managmentClients(Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if (!$securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')){
                return $this->redirectToRoute('admin');
            }else if($user->getType()==1){
                return $this->redirectToRoute('ClientIndex');
            }
            $c = new Client();
            $u1 = new Users();
            $departments = $this->getDoctrine()
                ->getRepository('AppBundle:Department')
                ->findAll();
            $total1=array(
                'user'=>$u1,
                'client'=>$c,
            );

            $form1= $this->createForm(MergedUC::class,$total1);
            $form1->handleRequest($request);
            if($form1->isSubmitted()&&$form1->isValid()){
```

```
                $name=$form1['client']['name']->getData();
                $surname=$form1['client']['surname']->getData();
                $email=$form1['user']['email']->getData();
                $username=$form1['user']['username']->getData();
                $password=$form1['user']['plainPassword']->getData();
                $type=1;
                $bank=$form1['client']['bank']->getData();
                $cardid=$form1['client']['cardid']->getData();
                $salary=$form1['client']['salary']->getData();

                $total1['client']->setName($name);
                $total1['user']->setEmail($email);
                $total1['user']->setUsername($username);
                $total1['user']->setPlainPassword($password);
                $total1['user']->setType($type);
                $total1['user']->setEnabled(true);
                $total1['client']->setSurname($surname);
                $total1['client']->setBank($bank);
                $total1['client']->setCardid($cardid);
                $total1['client']->setSalary($salary);
                $em=$this->getDoctrine()->getManager();
                $em->persist($total1['user']);
                $em->flush();
                $em->persist($total1['client']);
                $em->flush();
                return $this->redirectToRoute('clients');

            }
            $users = $this->getDoctrine()
                ->getRepository('AppBundle:Users')
                ->findAll();
            $clients = $this->getDoctrine()
                ->getRepository('AppBundle:Client')
                ->findAll();
            return $this->render('Manager/menaxhousers.html.twig',array(
                'form2'=>$form1->createView(),
                'users'=>$users,
                'clients'=>$clients,
            ));
        }else{
            return $this->redirectToRoute('ClientIndex');
        }
```

```
    }
    /**
     * @Route("/edit_clients/{id}", name="editclients")
     */
    public function editClients($id,Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')) {
                $clients = $this->getDoctrine()
                    ->getRepository('AppBundle:Client')
                    ->find($id);
                $form = $this->createFormBuilder($clients)
                    ->add('cardid', TextType::class, array('label' => 'Card Id', 'attr' => array('class' => 'form-control')))
                    ->add('name', TextType::class, array('label' => 'Name', 'attr' => array('class' => 'form-control')))
                    ->add('surname', TextType::class, array('label' => 'Surname', 'attr' => array('class' => 'form-control')))
                    ->add('salary', TextType::class, array('label' => 'Salary', 'attr' => array('class' => 'form-control')))
                    ->add('bank', TextType::class, array('label' => 'Bank Account Nr', 'attr' => array('class' => 'form-control')))
                    ->add('save', SubmitType::class, array('label' => 'Edit Client', 'attr' => array('class' => 'btn btn-primary pull-right')))
                    ->getForm();
                $form->handleRequest($request);
                if ($form->isSubmitted() && $form->isValid()) {
                    $cardid = $form['cardid']->getData();
                    $name = $form['name']->getData();
                    $surname = $form['surname']->getData();
                    $salary = $form['salary']->getData();
                    $bank = $form['bank']->getData();
                    $clients->setCardid($cardid);
                    $clients->setName($name);
                    $clients->setSurname($surname);
                    $clients->setBank($bank);
                    $clients->setSalary($salary);
```

```
                    $em = $this->getDoctrine()->getManager();
                    $em->persist($clients);
                    $em->flush();
                    return $this->redirectToRoute('users');
                }
                return $this->render('Admin/editUCE.html.twig', array(
                    'form' => $form->createView(),
                    'clients' => $clients,
                ));
            }else{
                return $this->redirectToRoute('ClientIndex');
            }
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/edit_employees/{id}", name="editemployees")
     */
    public function editEmployees($id,Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')) {
                $employee = $this->getDoctrine()
                    ->getRepository('AppBundle:Employee')
                    ->find($id);
                $form = $this->createFormBuilder($employee)
                    ->add('name', TextType::class, array('label' => 'Name', 'attr' => array('class' => 'form-control')))
                    ->add('surname', TextType::class, array('label' => 'Surname', 'attr' => array('class' => 'form-control')))
                    ->add('ssn', TextType::class, array('label' => 'SSN', 'attr' => array('class' => 'form-control')))
                    ->add('position', TextType::class, array('label' => 'Position', 'attr' => array('class' => 'form-control')))
                    ->add('department', TextType::class, array('label' => 'Department', 'attr' => array('class' => 'form-control')))
                    ->add('salary', TextType::class, array('label' => 'Salary', 'attr'
```

```
=> array('class' => 'form-control')))
                    ->add('save', SubmitType::class, array('label' => 'Edit Employee',
'attr' => array('class' => 'btn btn-primary pull-right')))
                    ->getForm();
                $form->handleRequest($request);
                if ($form->isSubmitted() && $form->isValid()) {
                    $surname = $form['surname']->getData();
                    $name = $form['name']->getData();
                    $ssn = $form['ssn']->getData();
                    $position = $form['position']->getData();
                    $department = $form['department']->getData();
                    $salary = $form['salary']->getData();
                    $employee->setSsn($ssn);
                    $employee->setName($name);
                    $employee->setSurname($surname);
                    $employee->setPosition($position);
                    $employee->setSalary($salary);
                    $employee->setDepartment($department);
                    $em = $this->getDoctrine()->getManager();
                    $em->persist($employee);
                    $em->flush();
                    return $this->redirectToRoute('users');
                }
                return $this->render('Admin/editUCE.html.twig', array(
                    'form' => $form->createView(),
                    'employee' => $employee,
                ));
            }else{
                return $this->redirectToRoute('ClientIndex');
            }
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/edit_users/{id}", name="editusers")
     */
    public function editUserd($id,Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
```

```
            $user1 = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user1->hasRole('ROLE_SUPER_ADMIN')) {
                $user = $this->getDoctrine()
                    ->getRepository('AppBundle:Users')
                    ->find($id);
                $form = $this->createFormBuilder($user)
                    ->add('username', TextType::class, array('label' => 'Username', 'attr' => array('class' => 'form-control', 'row' => '5')))
                    ->add('email', TextType::class, array('label' => 'Email', 'attr' => array('class' => 'form-control', 'row' => '5')))
                    ->add('plainPassword', RepeatedType::class, array(
                        'type' => PasswordType::class,
                        'invalid_message' => 'Passworded duhet te jen njelloj.',
                        'options' => array('attr' => array('class' => 'form-control')),
                        'required' => false,
                        'first_options' => array('label' => 'Password:'),
                        'second_options' => array('label' => 'Confirm Password:'),
                        'empty_data' => '',
                    ))
                    ->add('type', ChoiceType::class, array('label' => 'Type', 'attr' => array('class' => 'form-control'), 'choices' => array(
                        'Employee' => '0',
                        'Client' => '1',
                        'Admin' => '2',
                    )))
                    ->add('save', SubmitType::class, array('label' => 'Edit User', 'attr' => array('class' => 'btn btn-primary pull-right')))
                    ->getForm();
                $form->handleRequest($request);
                if ($form->isSubmitted() && $form->isValid()) {
                    $username = $form['username']->getData();
                    $email = $form['email']->getData();
                    $password = $form['plainPassword']->getData();
                    $type = $form['type']->getData();


                    $user->setUsername($username);
                    $user->setEmail($email);
                    $user->setType($type);
                    $user->setEnabled(true);
                    if ($password != null || $password != ""){
```

```
                        $user->setPlainPassword($password);
                    }
                    $em=$this->getDoctrine()->getManager();
                    $em->persist($user);
                    $em->flush();
                    return $this->redirectToRoute('users');
                }
                return $this->render('Admin/editUCE.html.twig', array(
                    'form' => $form->createView(),
                    'user' => $user,
                ));
            }else{
                return $this->redirectToRoute('ClientIndex');
            }
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/edit_client/{id}", name="editclient")
     */
    public function editClient($id,Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if (!$securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')){
                return $this->redirectToRoute('admin');
            }else if($user->getType()==1){
                return $this->redirectToRoute('ClientIndex');
            }
            $clients = $this->getDoctrine()
                ->getRepository('AppBundle:Client')
                ->find($id);
            $form=$this->createFormBuilder($clients)
                ->add('cardid',TextType::class,array('label'=>'Card Id','attr'=>array('class'=> 'form-control')))
                ->add('name',TextType::class,array('label'=>'Name','attr'=>array('class'=> 'form-control')))
```

```
                -
>add('surname',TextType::class,array('label'=>'Surname','attr'=>array('class'=> 'form-
control')))
                -
>add('salary',TextType::class,array('label'=>'Salary','attr'=>array('class'=> 'form-
control')))
                ->add('bank',TextType::class,array('label'=>'Bank Account
Nr','attr'=>array('class'=> 'form-control')))
                ->add('save',SubmitType::class,array('label'=>'Edit
Client','attr'=>array('class'=> 'btn btn-primary pull-right')))
                ->getForm();
            $form->handleRequest($request);
            if($form->isSubmitted()&&$form->isValid()){
                $cardid=$form['cardid']->getData();
                $name=$form['name']->getData();
                $surname=$form['surname']->getData();
                $salary=$form['salary']->getData();
                $bank=$form['bank']->getData();
                $clients->setCardid($cardid);
                $clients->setName($name);
                $clients->setSurname($surname);
                $clients->setBank($bank);
                $clients->setSalary($salary);
                $em=$this->getDoctrine()->getManager();
                $em->persist($clients);
                $em->flush();
                return $this->redirectToRoute('clients');
            }
            return $this->render('Manager/editUCE.html.twig',array(
                'form'=>$form->createView(),
                'clients'=>$clients,
            ));
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/admin", name="admin")
     */
    public function admin(Request $request)
    {
```

```
        $securityContext = $this->container->get('security.authorization_checker');
        if (
```
```
$securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')){
                $loans = $this->getDoctrine()
                    ->getRepository('AppBundle:Loan')
                    ->findAll();
                $clients = $this->getDoctrine()
                    ->getRepository('AppBundle:Client')
                    ->findAll();
                $employees = $this->getDoctrine()
                    ->getRepository('AppBundle:Employee')
                    ->findAll();
                $businesses = $this->getDoctrine()
                    ->getRepository('AppBundle:Business')
                    ->findAll();
                $departments = $this->getDoctrine()
                    ->getRepository('AppBundle:Department')
                    ->findAll();
                $bloans=0;
                for($i=0;$i<sizeof($loans);$i++){
                    if($loans[$i]->getStatus()==0) $bloans++;
                }
                $nloans=0;
                for($i=0;$i<sizeof($loans);$i++){
                    if((time()-(60*60*24*7)) < $loans[$i]->getDataFillimit()->getTimestamp()){
                        $nloans++;
                    }
                }
                $ditet= array('0','0','0','0','0','0','0');
                $dje=0;
                $sot=0;
                for($i=0;$i<sizeof($loans);$i++){
                    if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>strtotime('-7 day') ){
                        $ditet[$loans[$i]->getDataFillimit()->format('w')]++;
                    }
                    if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>strtotime('-2 day') && strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))<strtotime('-1 day') ){
```

```
                        $dje++;
                    }
                    if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtotime('-1 day') ){
                        $sot++;
                    }
                }
                if($dje!=0)$inc=(($sot-$dje)/$dje)*100;
                else $inc=100;
                $muajt= array('0','0','0','0','0','0','0','0','0','0','0','0');
                for($i=0;$i<sizeof($loans);$i++){
                    if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtotime('-365 day') ){
                        $muajt[$loans[$i]->getDataFillimit()->format('n')-1]++;
                    }
                }
                return $this->render('Admin/dashboard.html.twig',array(
                    'loans'=>$loans,
                    'bloans'=>$bloans,
                    'nloans'=>$nloans,
                    'clients'=>$clients,
                    'employees'=>$employees,
                    'businesses'=>$businesses,
                    'departments'=>$departments,
                    'ditet'=>$ditet,
                    'muajt'=>$muajt,
                    'inc'=>$inc
                ));
            }else{
                return $this->redirectToRoute('ClientIndex');
            }

        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/manager", name="manager")
     */
    public function manager(Request $request)
    {
```

```
        $securityContext = $this->container->get('security.authorization_checker');
        if (
$securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()-
>getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')){
                return $this->redirectToRoute('admin');
            }else if($user->getType()==1){
                return $this->redirectToRoute('ClientIndex');
            }
            $loans = $this->getDoctrine()
                ->getRepository('AppBundle:Loan')
                ->findAll();
            $clients = $this->getDoctrine()
                ->getRepository('AppBundle:Client')
                ->findAll();
            $employees = $this->getDoctrine()
                ->getRepository('AppBundle:Employee')
                ->findAll();
            $businesses = $this->getDoctrine()
                ->getRepository('AppBundle:Business')
                ->findAll();
            $departments = $this->getDoctrine()
                ->getRepository('AppBundle:Department')
                ->findAll();
            $bloans=0;
            for($i=0;$i<sizeof($loans);$i++){
                if($loans[$i]->getStatus()==0) $bloans++;
            }
            $nloans=0;
            for($i=0;$i<sizeof($loans);$i++){
                if((time()-(60*60*24*7)) < $loans[$i]->getDataFillimit()-
>getTimestamp()){
                    $nloans++;
                }
            }
            $ditet= array('0','0','0','0','0','0','0');
            $dje=0;
            $sot=0;
            for($i=0;$i<sizeof($loans);$i++){
                if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtotime('-7 day') ){
                    $ditet[$loans[$i]->getDataFillimit()->format('w')]++;
```

```
                }
                if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtotime('-2 day') && strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))<
strtotime('-1 day') ){
                    $dje++;
                }
                if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtotime('-1 day') ){
                    $sot++;
                }
            }
            if($dje!=0)$inc=(($sot-$dje)/$dje)*100;
            else $inc=100;
            $muajt= array('0','0','0','0','0','0','0','0','0','0','0','0');
            for($i=0;$i<sizeof($loans);$i++){
                if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtotime('-365 day') ){
                    $muajt[$loans[$i]->getDataFillimit()->format('n')-1]++;
                }
            }
            return $this->render('Manager/dashboard.html.twig',array(
                'loans'=>$loans,
                'bloans'=>$bloans,
                'nloans'=>$nloans,
                'clients'=>$clients,
                'employees'=>$employees,
                'businesses'=>$businesses,
                'departments'=>$departments,
                'ditet'=>$ditet,
                'muajt'=>$muajt,
                'inc'=>$inc
            ));
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/signup", name="signup")
     */
    public function signup(Request $request)
    {
```

```
        $securityContext = $this->container->get('security.authorization_checker');
        if (!$securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            $c = new Client();
            $u1 = new Users();
            $total1=array(
                'user'=>$u1,
                'client'=>$c
            );
            $form1= $this->createForm(MergedUC::class,$total1);
            $form1->handleRequest($request);
            if($form1->isSubmitted()&&$form1->isValid()){
                $name=$form1['client']['name']->getData();
                $surname=$form1['client']['surname']->getData();
                $email=$form1['user']['email']->getData();
                $username=$form1['user']['username']->getData();
                $password=$form1['user']['plainPassword']->getData();
                $type=1;
                $bank=$form1['client']['bank']->getData();
                $cardid=$form1['client']['cardid']->getData();
                $salary=$form1['client']['salary']->getData();

                $total1['client']->setName($name);
                $total1['user']->setEmail($email);
                $total1['user']->setUsername($username);
                $total1['user']->setPlainPassword($password);
                $total1['user']->setType($type);
                $total1['user']->setEnabled(true);
                $total1['client']->setSurname($surname);
                $total1['client']->setBank($bank);
                $total1['client']->setCardid($cardid);
                $total1['client']->setSalary($salary);
                $em=$this->getDoctrine()->getManager();
                $em->persist($total1['user']);
                $em->flush();
                $em->persist($total1['client']);
                $em->flush();
                return $this->redirectToRoute('ClientIndex');

            }
            $users = $this->getDoctrine()
```

```
                ->getRepository('AppBundle:Users')
                ->findAll();
            $clients = $this->getDoctrine()
                ->getRepository('AppBundle:Client')
                ->findAll();
            return $this->render('Client/signup.html.twig',array(
                'form2'=>$form1->createView(),
                'users'=>$users,
                'clients'=>$clients,
            ));
        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/aprofile", name="aprofile")
     */
    public function aprofile(Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')){
                return $this->render('Admin/profile.html.twig');
            }else{
                return $this->redirectToRoute('ClientIndex');
            }

        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
    /**
     * @Route("/profile/{id}", name="profile")
     */
    public function profile($id,Request $request)
    {
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('IS_AUTHENTICATED_REMEMBERED')) {
```

```
            $user = $this->container->get('security.token_storage')->getToken()->getUser();
            if($user->hasRole('ROLE_SUPER_ADMIN')){
                return $this->redirectToRoute($this->aprofile());
            }else if($user->getType()==1){
                $client = $this->getDoctrine()
                    ->getRepository('AppBundle:Client')
                    ->findBy(array('userId' => $id));
                $total1 = array(
                    'user' => $user,
                    'client' => $client[0]
                );
                $loans = $this->getDoctrine()
                    ->getRepository('AppBundle:Loan')
                    ->findBy(array('clientid'=>$client[0]->getId()));
                $aloans = $this->getDoctrine()
                    ->getRepository('AppBundle:LoanApproval')
                    ->findBy(array('clientid'=>$client[0]->getId()));

                $form1 = $this->createForm(MergedUC::class, $total1);
                $form1->handleRequest($request);
                if ($form1->isSubmitted() && $form1->isValid()) {
                    $name = $form1['client']['name']->getData();
                    $surname = $form1['client']['surname']->getData();
                    $email = $form1['user']['email']->getData();
                    $username = $form1['user']['username']->getData();
                    $password = $form1['user']['plainPassword']->getData();
                    $type = 1;
                    $bank = $form1['client']['bank']->getData();
                    $cardid = $form1['client']['cardid']->getData();
                    $salary = $form1['client']['salary']->getData();

                    $total1['client']->setName($name);
                    $total1['user']->setEmail($email);
                    $total1['user']->setUsername($username);
                    $total1['user']->setPlainPassword($password);
                    $total1['user']->setType($type);
                    $total1['client']->setSurname($surname);
                    $total1['client']->setBank($bank);
                    $total1['client']->setCardid($cardid);
                    $total1['client']->setSalary($salary);
                    $em = $this->getDoctrine()->getManager();
```

```
                $em->persist($total1['user']);
                $em->flush();
                $total1['client']->setUserId($total1['user']);
                $em->persist($total1['client']);
                $em->flush();

                return $this->redirectToRoute('ClientIndex');
            }
            return $this->render('Client/profile.html.twig',array(
                "form2"=>$form1->createView(),
                "loans"=>$loans,
                "aloans"=>$aloans,
            ));
        }else{
            $emp = $this->getDoctrine()
                ->getRepository('AppBundle:Employee')
                ->findBy(array('userId' => $id));
            $total = array(
                'user' => $user,
                'employee' => $emp[0]
            );
            $form = $this->createForm(MergedUE::class, $total);
            $form->handleRequest($request);

            if ($form->isSubmitted() && $form->isValid()) {
                $name = $form['employee']['name']->getData();
                $surname = $form['employee']['surname']->getData();
                $email = $form['user']['email']->getData();
                $username = $form['user']['username']->getData();
                $password = $form['user']['plainPassword']->getData();
                $type = 0;
                $ssn = $form['employee']['ssn']->getData();
                $department = $form['employee']['department']->getData();
                $position = $form['employee']['position']->getData();
                $salary = $form['employee']['salary']->getData();

                $total['employee']->setName($name);
                $total['user']->setEmail($email);
                $total['user']->setUsername($username);
                $total['user']->setPlainPassword($password);
                $total['user']->setType($type);
                $total['user']->setEnabled(true);
```

```
                    $total['employee']->setSurname($surname);
                    $total['employee']->setSsn($ssn);
                    $total['employee']->setDepartment($department);
                    $total['employee']->setPosition($position);
                    $total['employee']->setSalary($salary);

                    $em = $this->getDoctrine()->getManager();
                    $em->persist($total['user']);
                    $em->flush();
                    $total['employee']->setUserId($total['user']);
                    $em->persist($total['employee']);
                    $em->flush();
                    return $this->redirectToRoute('manager');

                }
               return $this->render('Client/profile.html.twig',array(
                    "form"=>$form->createView(),
                ));
            }

        }else{
            return $this->redirectToRoute('ClientIndex');
        }

    }
}
```

Admin View

```
{% extends 'Admin/base.html.twig' %}
{% block info %}
    <div class="content">
        <div class="container-fluid">
            <div class="row">
                <div class="col-md-7">
                    <div class="card">
                        <div class="card-header" data-background-color="green">
                            <h4 class="title">Add Employee</h4>
                            <p class="category">Complete Employee profile</p>
                        </div>
                        <div class="card-content">
                            {{ form_start(form) }}
                            <div class="row">
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form.employee.name) }}</label>
                                            {{ form_widget(form.employee.name) }}
                                            {{ form_errors(form.employee.name) }}
                                    </div>
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form.employee.surname) }}</label>
                                            {{ form_widget(form.employee.surname) }}
                                            {{ form_errors(form.employee.surname) }}
                                    </div>
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form.employee.salary) }}</label>
                                            {{ form_widget(form.employee.salary) }}
                                            {{ form_errors(form.employee.salary) }}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
```

```
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form.employee.ssn) }}</label>
                                        {{ form_widget(form.employee.ssn) }}
                                        {{ form_errors(form.employee.ssn) }}
                                    </div>
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form.employee.position) }}</label>
                                        {{ form_widget(form.employee.position) }}
                                        {{ form_errors(form.employee.position) }}
                                    </div>
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form.employee.department) }}</label>
                                        {{ form_widget(form.employee.department) }}
                                        {{ form_errors(form.employee.department) }}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form.user.email) }}</label>
                                        {{ form_widget(form.user.email) }}
                                        {{ form_errors(form.user.email) }}
                                    </div>
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form.user.username) }}</label>
                                        {{ form_widget(form.user.username) }}
                                        {{ form_errors(form.user.username) }}
                                    </div>
                                </div>
```

```
                            </div>
                            <div class="row">
                                <div class="col-md-6">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form.user.plainPassword.first) }}</label>
                                        {{ form_widget(form.user.plainPassword.first) }}
                                        {{ form_errors(form.user.plainPassword.first) }}
                                    </div>
                                </div>


                                <div class="col-md-6">
                                    <div class="form-group label-floating">
                                        <label class="control-label">  {{
form_label(form.user.plainPassword.second) }}</label>
                                        {{ form_widget(form.user.plainPassword.second)
}}
                                        {{ form_errors(form.user.plainPassword.second)
}}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
                                <div class="form-group label-floating col-md-12">
                                    {{ form_widget(form.save) }}
                                </div>
                            </div>
                            {{ form_end(form) }}
                        </div>
                    </div>
                </div>
                <div class="col-md-5">
                        <div class="card">
                            <div class="card-header" data-background-color="purple">
                                <h4 class="title">Users</h4>
                                <p class="category">Manage Users</p>
                            </div>
                            <div class="card-content table-responsive">
                                <table class="table">
                                    <thead class="text-primary">
                                    <th>Username</th>
```

```
                                    <th>Email</th>
                                    <th>Edit</th>
                                    </thead>
                                    <tbody>
                                     {% for x in users %}
                                        <tr>
                                            <td> {{ x.username }}</td>
                                            <td>{{ x.email }}</td>
                                            <td><button  id="{{ x.id }}" class="btn btn-
delete but1" href=""><i class="material-icons">open_in_new</i></button>
                                                <a  id="{{ x.id }}" class="btn btn-
delete but3" href="{{ path('editusers', {'id': x.id }) }}"><i class="material-
icons">create</i></a></td>
                                        </tr>
                                     {% endfor %}
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                </div>
            <div class="container-fluid">
                <div class="row">
                    <div class="col-md-7">
                        <div class="card">
                            <div class="card-header" data-background-color="green">
                                <h4 class="title">Add Client</h4>
                                <p class="category">Complete Client profile</p>
                            </div>
                            <div class="card-content">
                                {{ form_start(form2) }}
                                <div class="row">
                                    <div class="col-md-4">
                                        <div class="form-group label-floating">
                                            <label class="control-label">{{
form_label(form2.client.name) }}</label>
                                            {{ form_widget(form2.client.name) }}
                                            {{ form_errors(form2.client.name) }}
                                        </div>
                                    </div>
                                    <div class="col-md-4">
```

```
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form2.client.surname) }}</label>
                                            {{ form_widget(form2.client.surname) }}
                                            {{ form_errors(form2.client.surname) }}
                                    </div>
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form2.client.salary) }}</label>
                                            {{ form_widget(form2.client.salary) }}
                                            {{ form_errors(form2.client.salary) }}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
                                <div class="col-md-6">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form2.client.bank) }}</label>
                                            {{ form_widget(form2.client.bank) }}
                                            {{ form_errors(form2.client.bank) }}
                                    </div>
                                </div>
                                <div class="col-md-6">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form2.client.cardid) }}</label>
                                            {{ form_widget(form2.client.cardid) }}
                                            {{ form_errors(form2.client.cardid) }}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form2.user.email) }}</label>
                                            {{ form_widget(form2.user.email) }}
                                            {{ form_errors(form2.user.email) }}
                                    </div>
```

```
                                </div>
                                <div class="col-md-4">
                                    <div class="form-group label-floating">
                                        <label class="control-label"> {{
form_label(form2.user.username) }}</label>
                                        {{ form_widget(form2.user.username) }}
                                        {{ form_errors(form2.user.username) }}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
                                <div class="col-md-6">
                                    <div class="form-group label-floating">
                                        <label class="control-label">{{
form_label(form2.user.plainPassword.first) }}</label>
                                        {{ form_widget(form2.user.plainPassword.first)
}}
                                        {{ form_errors(form2.user.plainPassword.first)
}}
                                    </div>
                                </div>


                                <div class="col-md-6">
                                    <div class="form-group label-floating">
                                        <label class="control-label">  {{
form_label(form2.user.plainPassword.second) }}</label>
                                        {{ form_widget(form2.user.plainPassword.second)
}}
                                        {{ form_errors(form2.user.plainPassword.second)
}}
                                    </div>
                                </div>
                            </div>
                            <div class="row">
                                <div class="form-group label-floating col-md-12">
                                    {{ form_widget(form2.save) }}
                                </div>
                            </div>
                            {{ form_end(form2) }}
                        </div>
                    </div>
```

```
                </div>

                <div class="col-md-5">
                    <div class="card">
                        <div class="card-header" data-background-color="purple">
                            <h4 class="title">Employees</h4>
                            <p class="category">Manage Employees</p>
                        </div>
                        <div class="card-content table-responsive">
                            <table class="table">
                                <thead class="text-primary">
                                <th>Name</th>
                                <th>SSN</th>
                                <th>Edit</th>
                                </thead>
                                <tbody>
                                {% for x in employees %}
                                    <tr>
                                        <td> {{ x.name }}</td>
                                        <td>{{ x.ssn }}</td>
                                        <td><button  id="{{ x.id }}" class="btn btn-
delete but2" href=""><i class="material-icons">open_in_new</i></button>
                                        <a  id="{{ x.id }}" class="btn btn-delete"
href="{{ path('editemployees', {'id': x.id }) }}"><i class="material-
icons">create</i></a></td>
                                    </tr>
                                {% endfor %}
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>

            </div>
        </div>
        <div class="container-fluid">
             <div class="row">
                <div class="col-md-6">
                    <div class="card">
                        <div class="card-header" data-background-color="red">
                            <h4 class="title">Clients</h4>
                            <p class="category">Manage Clients</p>
```

```
                        </div>
                        <div class="card-content table-responsive">
                            <table class="table">
                                <thead class="text-primary">
                                <th>Name</th>
                                <th>Card ID</th>
                                <th>Edit</th>
                                </thead>
                                <tbody>
                                {% for x in clients %}
                                    <tr>
                                        <td> {{ x.name }}</td>
                                        <td>{{ x.cardid }}</td>
                                        <td><button  id="{{ x.id }}" class="btn btn-
delete but3" href=""><i class="material-icons">open_in_new</i></button>
                                            <a class="btn btn-delete" href="{{
path('editclients', {'id':  x.id }) }}"><i class="material-icons">create</i></a></td>
                                    </tr>
                                {% endfor %}
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    {% for x in users %}
        <div id="cont{{ x.id }}" style="display:none">
            <div class="row">
                <div class="col-md-5">
                    <label class="control-label">Email:</label> <div type='text'
class='form-control' name="email" value=''>{{ x.email }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Username:</label> <div type='text'
class='form-control' name="username" value=''>{{ x.username }}</div>
                </div>
                <div class="col-md-3">
                    <label class="control-label">Type:</label> <div type='text'
class='form-control' name="user" value=''>
                        {% if x.type == 0 %}
```

```
                        Employee
                    {% elseif x.type ==1 %}
                        Client
                    {% elseif x.type ==2 %}
                        Admin
                    {% endif %}
                </div>
            </div>
        </div>
    {% endfor %}
    {% for x in employees %}
        <div id="e{{ x.id }}" style="display:none">
            <div class="row">
                <div class="col-md-4">
                    <label class="control-label">Name:</label> <div type='text'
class='form-control' name="name" value=''>{{ x.name }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Surname:</label> <div type='text'
class='form-control' name="surname" value=''>{{ x.surname }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">SSN:</label> <div type='text'
class='form-control' name="ssn" value=''>{{ x.ssn }}</div>
                </div>
            </div>
            <div class="row">
                <div class="col-md-4">
                    <label class="control-label">Position:</label> <div type='text'
class='form-control' name="position" value=''>{{ x.position }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Department:</label> <div type='text'
class='form-control' name="department" value=''>{{ x.department }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Salary:</label> <div type='text'
class='form-control' name="salary" value=''>{{ x.salary }}</div>
                </div>
            </div>
        </div>
```

```
    {% endfor %}
    {% for x in clients %}
        <div id="c{{ x.id }}" style="display:none">
            <div class="row">
                <div class="col-md-4">
                    <label class="control-label">Card Id:</label> <div type='text'
class='form-control' name="cardid" value=''>{{ x.cardid }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Name:</label> <div type='text'
class='form-control' name="name" value=''>{{ x.name }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Surname:</label> <div type='text'
class='form-control' name="surname" value='surname'>{{ x.surname }}</div>
                </div>
            </div>
            <div class="row">
                <div class="col-md-8">
                    <label class="control-label">Bank Account Nr:</label> <div
type='text' class='form-control' name="bank" value=''>{{ x.bank }}</div>
                </div>
                <div class="col-md-4">
                    <label class="control-label">Salary:</label> <div type='text'
class='form-control' name="salary" value=''>{{ x.salary }}</div>
                </div>
            </div>
        </div>
    {% endfor %}
{% endblock %}
{% block javascripts %}

    <script>
        $(document).ready(function(){
            $("#users").addClass("active");
        });
        $(document).ready(function(){
            $(".but1").on('click',function () {
                bootbox.alert({
                    message: $('#cont'+this.id).html(),
                    size: 'medium'
                });
```

```
            })
            $(".but2").on('click',function () {
                bootbox.alert({
                    message: $('#e'+this.id).html(),
                    size: 'large'
                });
            })
            $(".but3").on('click',function () {
                bootbox.alert({
                    message: $('#c'+this.id).html(),
                    size: 'large'
                });
            })
        });
    </script>
{% endblock %}
```

Installation Manual

The IuteCredit web application can be deployed and used from your computer by:

1) Using it on your PC locally.

2) Putting it on a server to use it from any device, since it is responsive.

To be able to use the IuteCredit application in your personal computer you should:

a) Install XAMPP

b) Unzip the IuteCredit.zip file located in the accompanying CD.

c) Copy and paste the folder as it is in the folder named htdocs in the C:/xampp/htdocs

d) Open xampp control panel and start apache and MySql servers.

e) Open your browser and go to-> localhost/phpmyadmin

f) There you import the Bits-Please.sql file to set up the database.

g) Now everything is ready and you can use the application under the path

localhost/IuteCredit/web

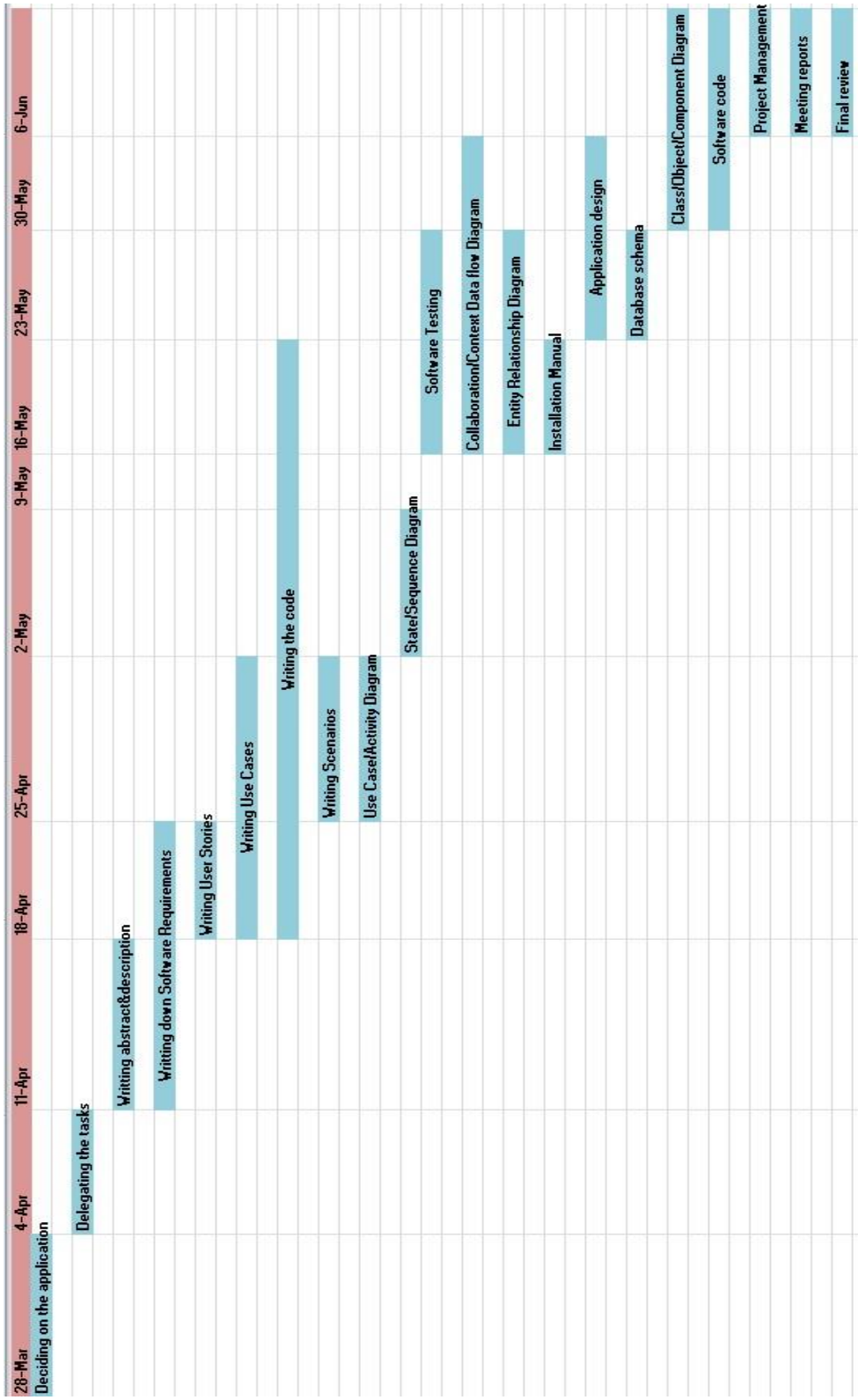
To be able to use the application remotely you should have an online server, or a hosting provided from another company. Despite your hosting the installation of IuteCredit web application is very simple.

a) Open your servers cpanel and copy the IuteCredit folder taken from the CD under the public_html directory.

b) Import the database Bits-Please.sql in the phpmyadmin of your server.

c) You are now ready to open the IuteCredit web application under the path yourdomain/IuteCredit/web

PROJECT MANAGEMENT

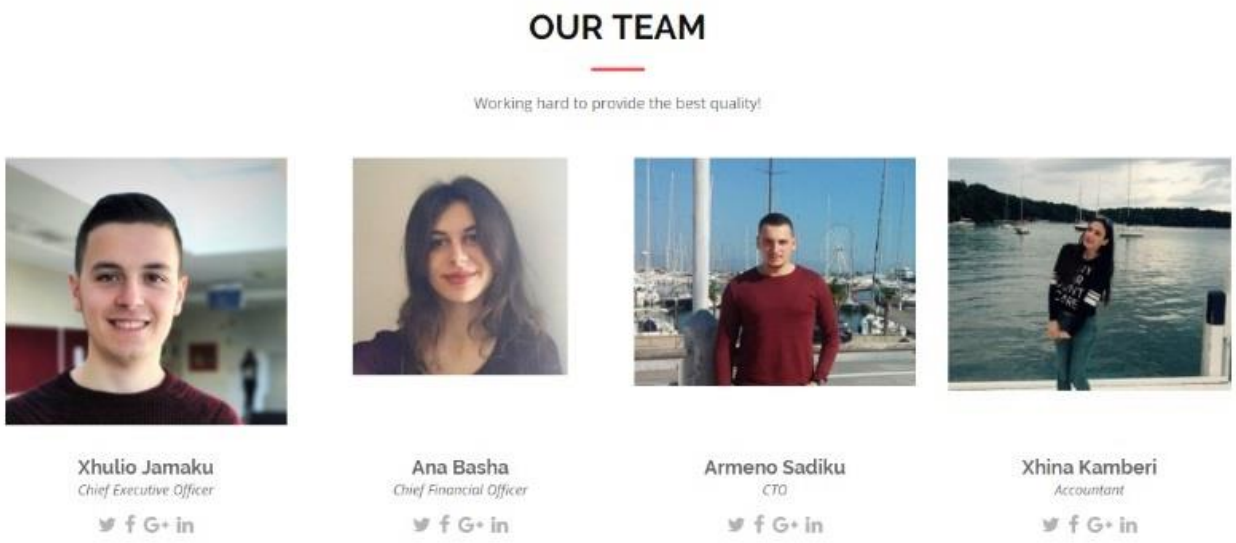


Requirements Confirmation/ Stakeholder Sign Off

Meeting Date	Attendees (name and role)	Comments
21/03/2017	Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Deciding on the application concept
28/03/2017	Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Talking with IUTE Credit's owner
04/04/2017	Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Task delegating
07/04/2017	Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Discussion about the project type and structure.
11/04/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Start writing SRS using templates as guidelines
18/04/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Start writing the code while adapting it to the document
25/04/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Write scenarios
02/05/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Start working with diagrams • Update document
05/05/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Continue coding • Set goals for next meeting

16/05/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Finish coding • Continue work with diagrams
23/05/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Start working with design • Make neccessary updates
30/05/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Write Software Testing and Code • Discuss final changes
06/06/2017	Igli Hakrama - Project Advisor Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Make last remarks • Complete Project Management
10/06/2017	Armeno Sadiku-Group Member Ana Basha-Group Member Xhulio Jamaku-Group Leader Xhina Kamberi-Group Member	• Make a final review

Contact US



Emails:

xjamaku14@epoka.edu.al

abasha14@epoka.edu.al

asadiku14@epoka.edu.al

xkamberi14@epoka.edu.al