



**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 fasten 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

|   |    |
|---|----|
| Abstract .....                                | 3  |
| Executive Summary.....                        | 7  |
| Project Overview.....                         | 8  |
| Purpose and Scope of this Specification ..... | 8  |
| Product/Service Description .....             | 9  |
| Product Context .....                         | 9  |
| User Characteristics.....                     | 10 |
| Assumptions.....                              | 11 |
| Constraints .....                             | 11 |
| Dependencies.....                             | 11 |
| Apportioning of requirements .....            | 11 |
| Requirements.....                             | 12 |
| Functional Requirements .....                 | 12 |
| User Interface Requirements .....             | 16 |
| Non- functional requirements.....             | 16 |
| Usability.....                                | 16 |
| Performance.....                              | 17 |
| Capacity.....                                 | 17 |
| Availability.....                             | 17 |
| Latency .....                                 | 18 |
| Protection.....                               | 18 |
| Authorization and Authentication .....        | 18 |
| Manageability/Maintainability.....            | 18 |
| Monitoring .....                              | 18 |
| Maintenance .....                             | 19 |
| Operations.....                               | 19 |
| System Interface/Integration.....             | 20 |
| Network and Hardware Interfaces.....          | 20 |
| Systems Interfaces .....                      | 20 |
| Portability.....                              | 20 |
| INTERFACE.....                                | 21 |

|   |     |
|---|-----|
| User Scenarios/Use Cases .....                        | 40  |
| User Stories .....                                    | 40  |
| Scenarios .....                                       | 42  |
| General Use cases .....                               | 52  |
| DIAGRAMS.....   | 76  |
| USE CASE DIAGRAM .....                                | 77  |
| USER DIAGRAM .....                                    | 78  |
| EMPLOYEE DIAGRAM .....                                | 79  |
| ADMIN DIAGRAM.....                                    | 80  |
| ACTIVITY DIAGRAM .....                                | 81  |
| SEQUENCE DIAGRAMS .....                               | 99  |
| STATE DIAGRAM.....                                    | 112 |
| COLLABORATION DIAGRAM.....                            | 121 |
| CONTEXT DATA-FLOW DIAGRAM .....                       | 130 |
| DATA-FLOW DIAGRAM 1 .....                             | 131 |
| DATA-FLOW DIAGRAM 2 .....                             | 132 |
| DATA-FLOW DIAGRAM 3 .....                             | 133 |
| ENTITY RELATIONSHIP DIAGRAM.....                      | 134 |
| DATABASE SCHEMA.....                                  | 135 |
| CLASS DIAGRAM .....                                   | 136 |
| COMPONENT DIAGRAM.....                                | 137 |
| DEPLOYMENT DIAGRAM.....                               | 138 |
| Software Testing .....                                | 140 |
| Installation Manual .....                             | 174 |
| PROJECT MANAGEMENT .....                              | 175 |
| Requirements Confirmation/ Stakeholder Sign Off ..... | 176 |
| Contact US.....                                       | 178 |

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 constraints 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 its 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 IUTEccredit. 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 IUTEccredit.

## ***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***

Constraints 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 constraint 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 constraint of our application can be the hacking, because even if we try to keep it safe, new hacking techniques 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              |

| <b>Req#</b> | <b>Requirement</b> | <b>Comments</b>  | <b>Priority</b> | <b>Date Rvwd</b> | <b>SME Reviewed / Approved</b> |
|-------------|--------------------|--|-----------------|------------------|--------------------------------|
| 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 from 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                     |

| <b>Req#</b> | <b>Requirement</b>    | <b>Comments</b>   | <b>Priority</b> | <b>Date Rvwd</b> | <b>SME Reviewed / Approved</b> |
|-------------|-----------------------|---|-----------------|------------------|--------------------------------|
| 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.<br><br>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                     |

| <b>Req#</b> | <b>Requirement</b>            | <b>Comments</b>   | <b>Priority</b> | <b>Date Rvwd</b> | <b>SME Reviewed / Approved</b> |
|-------------|-------------------------------|---|-----------------|------------------|--------------------------------|
| 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:

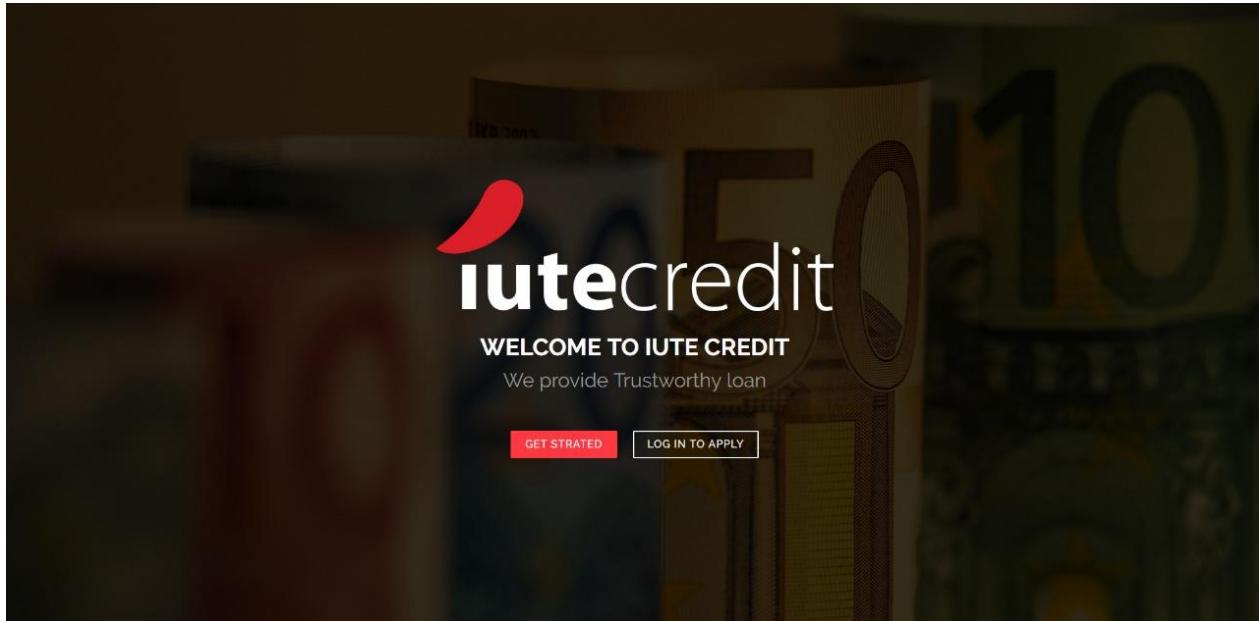
- The system may use the protocols of HTTP and HTTPS when needed for transfer of publishing.
- The system, if necessary, will use the SQL and technologies of database for backup, user information.
- The system may use different networking protocols like UDP, TCP/IP or others.

## ***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

### OUR SERVICE

The customer experience is the next competitive battleground.

- MAKE THE APPLICATION**  
In less than 5 minutes
- SIGN LOAN AGREEMENT**  
at iuteCredit Office
- GET MONEY SAME DAY**  
in local postal office or on your bank account

#### LOAN CONDITIONS

Without collateral  
Fixed interest rate 15.0-17.0% of outstanding balance  
Loan amount 10.000-100.000 ALL for new clients and 10.000-200.000 for returning clients  
Loan period 1-12 month  
Commission fee depending on loan period

#### REQUIREMENTS FOR APPLICANTS

Age between 18-68 years old  
Albanian citizen or person with long-term residence permit  
Correct payment behavior

# ABOUT US



Home About Us Service Testimonials Team More Contact Us Log In Sign Up

## ABOUT US

Iute Credit Albania SHA issues consumer loans without collateral on the basis of personal identity cards, transferring loan amounts through Posta Moldovei or to your bank or card account on the day of signing loan agreement.



**Trustworthy loan**  
Iute Credit is a reliable partner in financial services, being presented in the market of the Albania since year 2014. Our work is supervised by the Financial Supervision Authority. We deeply respect the principles of responsible lending and provide loans only to persons, who are able to get it back.

**Flexibility**  
Loan is available at reasonable interest rate of 15.0% for the period up to 12 months in the amount up to 200.000 ALL. It means, that you can choose suitable amount of the monthly payment and the date of payment.

**Answer on application in 30 minutes**  
To submit application short questionnaire should be filled in one of the Iute Credit offices or on our website. You will get an answer in 1 hour.

# CONTACT US

## CONTACT US

Iute Credit Sha

 Rruga e Durrësit nr. 6  
Tirane, Albania 1001

 info@iutecredit.al

 067 505 5500

 069 243 0750

 Egnatia Street  
Durrës, Albania 2001

 info@iutecredit.al

 067 505 5500

 069 243 0750

Your Name

Your Email

Subject

Message

© Copyright Iute Credit. All Rights Reserved

# ADMIN DASHBOARD

The dashboard features several key components:

- Top Left:** A red button labeled "Dashboard" with a white square icon.
- Top Right:** A teal button labeled "Logout" with a white square icon.
- Left Sidebar:**
  - Profile:** "Mire se vini, xhulio!" with a red profile picture.
  - Navigation:** "Manage Users", "Manage Businesses", "Manage Departments", "Manage Loans", "Messages", and "Logout".
- Central Metrics:**
  - Loans:** 9 loans (orange box).
  - Businesses:** Businesses 2 (red box).
  - Bad Loans:** Bad Loans 2 (red box).
  - New Loans:** New Loans 6 (teal box).
- Charts:**
  - Daily Sales:** A green line chart showing sales over the week (Monday to Sunday). Values range from 0 to 7.5.
  - Monthly Loans:** A blue line chart showing monthly loan performance from Jan to Dec. Values range from 0 to 75.
  - Daily Traffic:** A red line chart showing website traffic from 12am to 9pm. Values range from 0 to 1000.
- Employee Stats:** A purple box showing employee statistics. It includes a table with columns: SSN, Name, Salary, and Department. Data points include:
 

| SSN  | Name          | Salary  | Department |
|------|---------------|---------|------------|
| 134  | xhulio jemaku | \$10000 | BOARD      |
| 1234 | Jemim Jemaku  | \$1234  | sales      |
| 100  | xhma kamben   | \$100   | Marketing  |
- Tasks:** A purple box listing tasks with checkboxes:
 

| Task                           | Status |
|--------------------------------|--------|
| Add messenger availability*    | ✓      |
| Add other Financial Statements | ✗      |
| Remove unnecessary information | ✗      |

## ADMIN- MANAGE USERS PAGE

| Username   | Email                | Edit                                | Delete |
|------------|----------------------|-------------------------------------|--------|
| oo@vi      | elvi.jamal@gmail.com | <input checked="" type="checkbox"/> | /      |
| [REDACTED] | [REDACTED]           | <input type="checkbox"/>            | /      |
| [REDACTED] | [REDACTED]           | <input type="checkbox"/>            | /      |
| [REDACTED] | [REDACTED]           | <input type="checkbox"/>            | /      |
| [REDACTED] | [REDACTED]           | <input type="checkbox"/>            | /      |
| xh@lio     | xhlio.jamal@live.com | <input type="checkbox"/>            | /      |
| jemin      | jemin@yahoo.com      | <input type="checkbox"/>            | /      |
| le@da      | le@yahoo.com         | <input type="checkbox"/>            | /      |
| adminUser  | x@gmail.com          | <input type="checkbox"/>            | /      |
| an@b       | an@b.com             | <input type="checkbox"/>            | /      |

Miré se vini xhullo!

Add Employee  
Complete Employee profile

Name \_\_\_\_\_ Surname \_\_\_\_\_ Salary \_\_\_\_\_  
Position \_\_\_\_\_ Department \_\_\_\_\_ Marketing

SSN \_\_\_\_\_ Username \_\_\_\_\_ Email \_\_\_\_\_  
Email \_\_\_\_\_  
Password: \_\_\_\_\_ Confirm Password: \_\_\_\_\_

**SUBMIT**



Dashboard  
**Manage Users**

Manage Businesses  
Manage Departments  
Manage Loans  
Messages  
Log Out

## ADMIN – MANAGE BUSINESSES PAGE

The screenshot shows the 'Admin – Manage Businesses' page. On the left, there is a sidebar with the 'luteCredit' logo and several navigation links: Dashboard, Manage Users (highlighted in red), Manage Departments, Manage Loans, Messages, and Log Out.

The main content area has two sidebars. The left sidebar is green and contains the 'Add Business' section, which includes fields for Name (Njt), Logo (with a 'Choose File' button and 'No file chosen' message), and Description. A purple 'ADD BUSINESS' button is located at the bottom of this sidebar.

The right sidebar is red and displays a table of businesses. It has a header row with columns for Logo, Name, Njt, Email, and Description. Below this, there is one visible row for 'KASTRATI'. The 'KASTRATI' entry includes a yellow logo, the name 'KASTRATI', Njt '12345678', Email 'info@kastri.al', and a description: 'We aim to support long-term economic growth, social stability, prosperity and progress in the regions where we operate, as well as caring for the environment and ensuring sustainable use of natural resources. We want to achieve consistent and long-term growth'. To the right of the table, there is a note: 'Vodafone Albania Sh.A. është pjesë e Vodafone Group Plc. Kompania lider në botë me fushën e komunikimeve elektronike. Marka Vodafone tregonit si marka e shërbimeve më profesionale'. At the bottom of the red sidebar, there are icons for Vodafone and its website address: [www.vodafone.al](http://www.vodafone.al).

## ADMIN- MANAGE DEPARTMENT PAGE

The screenshot displays a user interface for managing departments. At the top, there is a navigation bar with a logo on the left and several menu items: Dashboard, Manage Users, Manage Businesses, Manage Loans, Messages, and Log Out. The 'Manage Departments' item is highlighted with a red background and white text. Below the navigation bar, the main content area has a green header bar on the left containing the text 'Miti se vini xhuljo!' and 'Add Department Complete department profile'. The main body of the page features a red sidebar on the left with the title 'Departments' and a sub-section 'Manage Departments'. This sidebar contains a list of department names: Marketing, Management, Loans, Finance, IT, and Board. A purple button labeled 'ADD DEPARTMENT' is located at the bottom of this sidebar. The main content area also includes a form for adding a new department, which has fields for 'Name' and 'Description'.

## ADMIN-MANAGE LOANS PAGE

The screenshot displays the Admin-Manage Loans page with three main sections:

- Add Loan**: A form to complete loan requirements. Fields include Client ID (123 elvi), Amount (1200), Interest (0.5), Maturity (6), Data Fillimit (2017-05-28), Status (Mbanar), and an Edit button.
- Expired Loans**: A table showing two expired loans. Both loans have Client ID 123, Amount 2000, Interest 0.5, Maturity 4, Data Fillimit 2017-05-04, Status Mbanar, and an Edit button.
- Loans**: A table showing two active loans. Both loans have Client ID 123, Client Name elvi, Amount 1200, Interest 0.5, Monthly Payment 120.2757763913, Maturity 10 Months, Data Fillimit 2017-05-28, Deadline 06/07/2017, Status Aktiv, and an Edit button.

## ADMIN- EDIT PROFILE

Mirë se vini xhullo!

**Profile**  
Admin profile

Dashboard

Manage Users

Manage Businesses

Manage Departments

Manage Loans

Messages

Log Out

**Profile**

CEO / CO-FOUNDER  
Xhullo Jamaku

First Name: Xhullo

Last Name: Jamaku

Address: Tirane-Dajt

City: Tirane

Country: Albania

Postal Code: 10000

About Me:  
CEO of lute Credit

Username: oomulio

Email address: xhullo.jamaku@hotmail.com

**FOLLOW**

## **MANAGER- DASHBOARD**

## MANAGER- MANAGE CLIENTS PAGE

The screenshot shows a user interface for managing clients. At the top, there's a header with the title "MANAGER- MANAGE CLIENTS PAGE". Below the header is a navigation bar with several items: "Dashboard" (with a grid icon), "Manage Clients" (highlighted with a red background and a person icon), "Manage Businesses" (with a building icon), "Manage Departments" (with a briefcase icon), "Manage Loans" (with a dollar sign icon), "Messages" (with a speech bubble icon), and "Log Out" (with a right-pointing arrow icon). On the left side, there's a vertical sidebar with a green header labeled "Add Client" and a sub-section "Complete Client profile". The main content area contains a form for adding a client. It has fields for "Name" (text input), "Surname" (text input), "Salary" (text input), "Bank Account Nr" (text input), "Card Id" (text input), "Email" (text input), "Username" (text input), "Password" (text input), and "Confirm Password" (text input). A purple "SUBMIT" button is located at the bottom of the form. To the right of the form, there's a red sidebar with the text "Clients" and "Manage Clients". Below this sidebar is a table listing clients. The first row shows columns for "Name" (with entries "eVi"), "Card ID" (with entry "123"), and "Edit" (with a grey button containing a checkmark and a slash). The bottom of the page features a footer with the number "31" between two brackets.

| Name | Card ID | Edit |
|------|---------|------|
| eVi  | 123     | /    |

## MANAGER- MANAGE BUSINESSES PAGE

The screenshot displays the 'Manage Businesses' section of the iutecredit platform. At the top, there's a green header bar with the title 'Add Business' and a sub-instruction 'Complete Business profile'. Below this, there are several input fields: 'Name' (Nict), 'Email', 'Logo' (with a placeholder 'Choose file No file chosen'), 'Description', and a purple 'ADD BUSINESS' button.

On the right side, there's a red header bar with the title 'Businesses' and a sub-instruction 'Manage Businesses'. Below this, a table lists two businesses:

| Logo | Name     | Nict     | Email           | Description  |
|------|----------|----------|-----------------|--|
|      | Kastrati | 12345678 | info@kastrati.a | We aim to support long-term economic growth, social stability, prosperity and progress in the regions where we operate, as well as caring for the environment and ensuring sustainable use of natural resources. We want to achieve consistent and long-term |
|      | KASTRATI |          |                 |  |

At the bottom left, a sidebar menu includes links for 'Dashboard', 'Manage Clients', 'Manage Businesses' (which is highlighted in red), 'Manage Departments', 'Manage Loans', 'Messages', and 'Log Out'.

## MANAGER- MANAGE DEPARTMENTS PAGE

The screenshot displays a user interface for managing departments. At the top, there is a navigation bar with icons for Home, Manage Clients, Manage Businesses, Manage Departments (highlighted in red), Manage Loans, Messages, and Log Out. Below the navigation bar, the main content area has a light gray background with a large white rectangular input field. On the left side of this field, there is a vertical sidebar with a red header containing the text "Departments" and "Manage Departments". The sidebar also lists several department names: Name, Marketing, Management, Loans, Finance, IT, and Board. To the right of the sidebar, there is a green rectangular box containing the text "Add Department" and "Complete department profile". Below this, there is a form with a single input field labeled "Name" and a purple "ADD DEPARTMENT" button.

Add Loan

[Complete Loan Requirements](#)

**Client ID:** 123 elvi jämäku

**Amount:** 1200

**Interest:** 0.5

**Maturity:** 6

**Date Fälligkeit:** 2017-05-28

**Status:** Aktiv

**Interest:** 1 Month

**Type:** Aktiv

**Amount:** 20000

**Maturity:** 14

**Date Fälligkeit:** 2017-06-04

**Status:** Maturity

**Client ID:** 123

**Amount:** 1200

**Interest:** 0.5

**Maturity:** 6

**Date Fälligkeit:** 2017-05-28

**Status:** Maturity

Expired Loans

**Manage Loans**

| Client ID | Amount | Interest | Maturity | Date Fälligkeit | Status   |
|-----------|--------|----------|----------|-----------------|----------|
| 123       | 1200   | 0.5      | 6        | 2017-05-28      | Aktiv    |
| 123       | 20000  | 14       | 4        | 2017-06-04      | Maturity |

Loans

**Manage Loans**

| Client ID | Amount | Interest | Monthly Payment | Month Payed | Maturity  | Date Fälligkeit | Deadline   | Status     | Manage |
|-----------|--------|----------|-----------------|-------------|-----------|-----------------|------------|------------|--------|
| 123       | 1200   | 0.5      | 144.33          | 2           | 10 Months | 2017-05-28      | 2018-03-28 | Aktiv      |        |
| 123       | 2000   | 10       | 251.13          | 2           | 10 Months | 2017-04-26      | 2018-02-26 | Aktiv      |        |
| 123       | 60000  | 0.17     | 18006.37        | 1           | 4 Months  | 2017-06-06      | 2017-10-06 | Aktiv      |        |
| 123       | 200000 | 0.15     | 17158.93        | 0           | 14 Months | 2017-05-06      | 2018-06-06 | We.Vorasse |        |
| 123       | 68500  | 0.17     | 27407.76        | 0           | 3 Months  | 2017-01-08      | 2017-09-08 | We.Vorasse |        |
| 1234521   | 92500  | 0.17     | 27759.62        | 0           | 4 Months  | 2017-05-09      | 2017-11-09 | We.Vorasse |        |
| 10291     | 100000 | 0.17     | 20009.91        | 0           | 6 Months  | 2017-06-10      | 2017-12-10 | Aktiv      |        |

Requested Loans

**Manage Loans**

| Client ID | Amount | Interest | Monthly Payment | Maturity | Date Fälligkeit | Status     |
|-----------|--------|----------|-----------------|----------|-----------------|------------|
| 1         | 68000  | 0.17     | 20407.22        | 5 Months | 2017-06-08      | 2017-10-08 |
| 1         | 100000 | 0.17     | 30010.62        | 4 Months | 2017-06-09      | 2017-10-09 |

34

# MANAGER- LOAN PAYMENT INVOICE

## INVOICE

Xhulio Jamaku

Rruga e Durrësit 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



**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 |

## TERMS

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

## CLIENT- APPLY FOR LOAN

**Apply for Loan**  
Calculate Monthly Payments and apply for loan!

New client      Returning client

Loan amount      Loan period

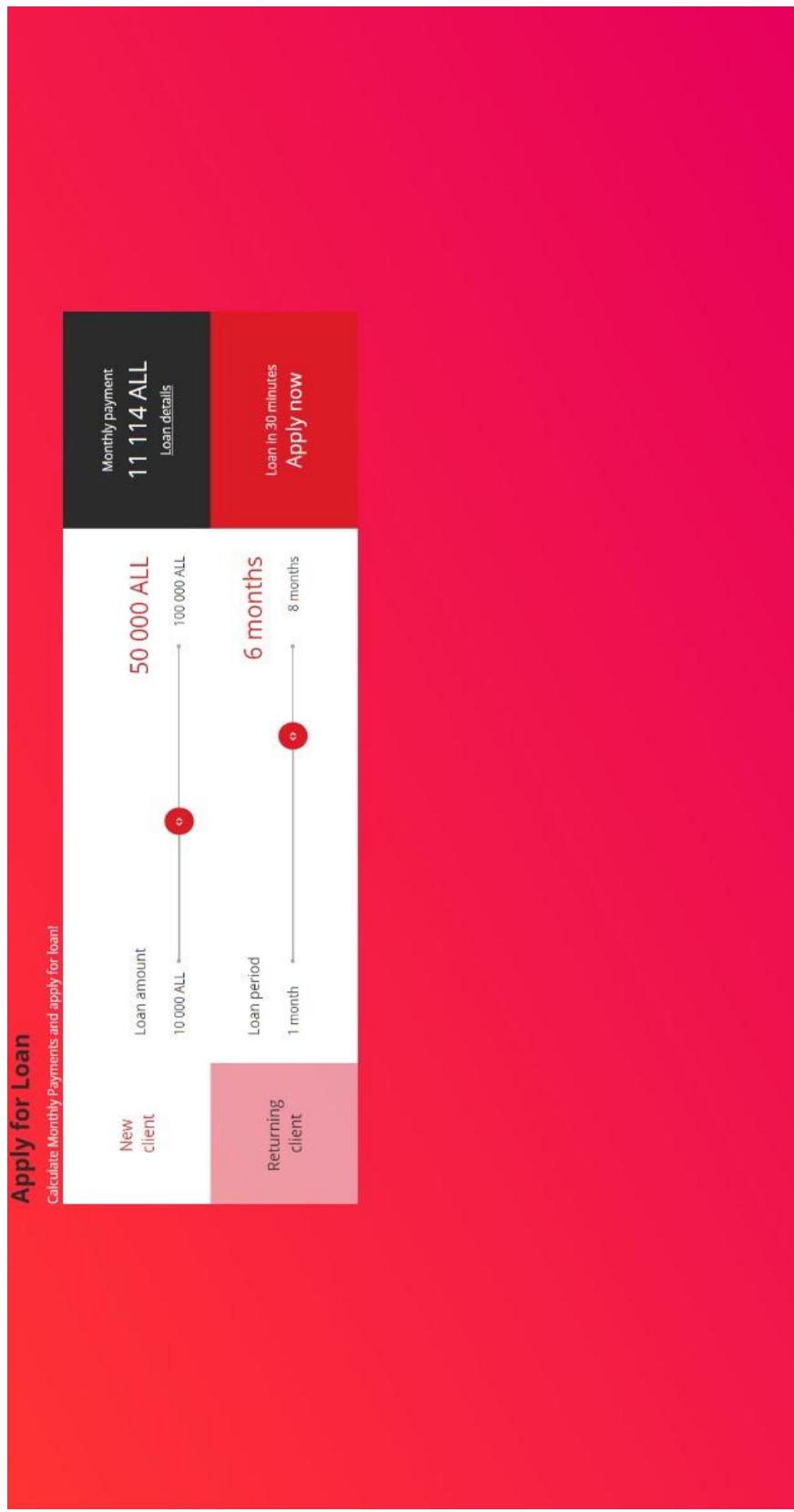
10 000 ALL      1 month

50 000 ALL      6 months

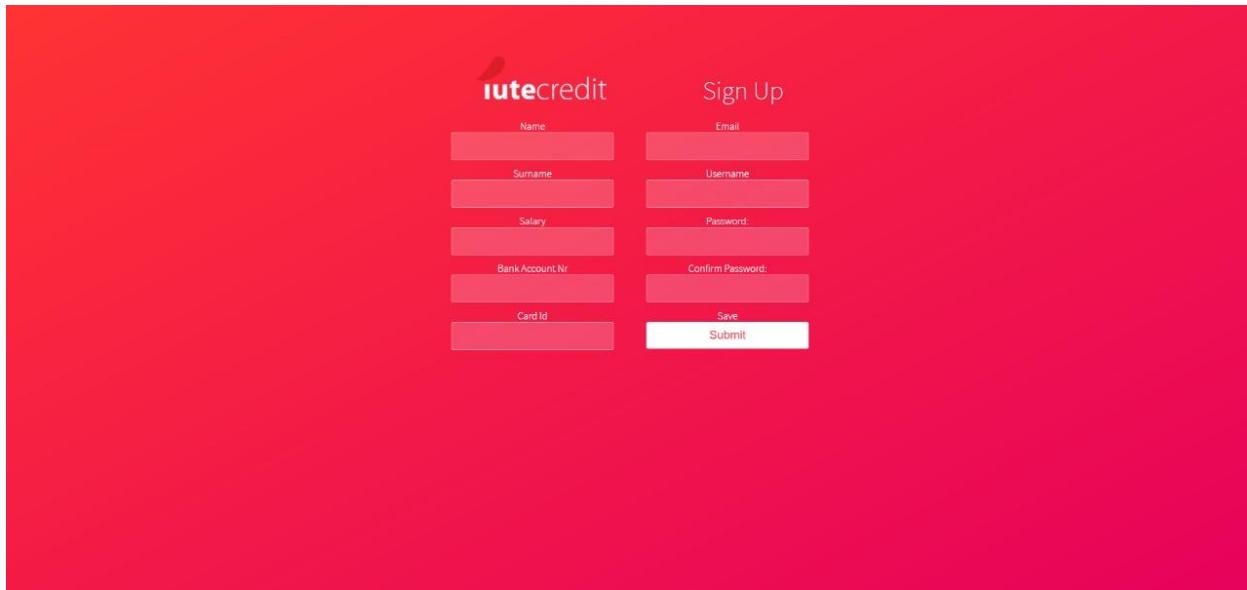
100 000 ALL      8 months

Monthly payment  
**11 114 ALL**

Loan details      Apply now



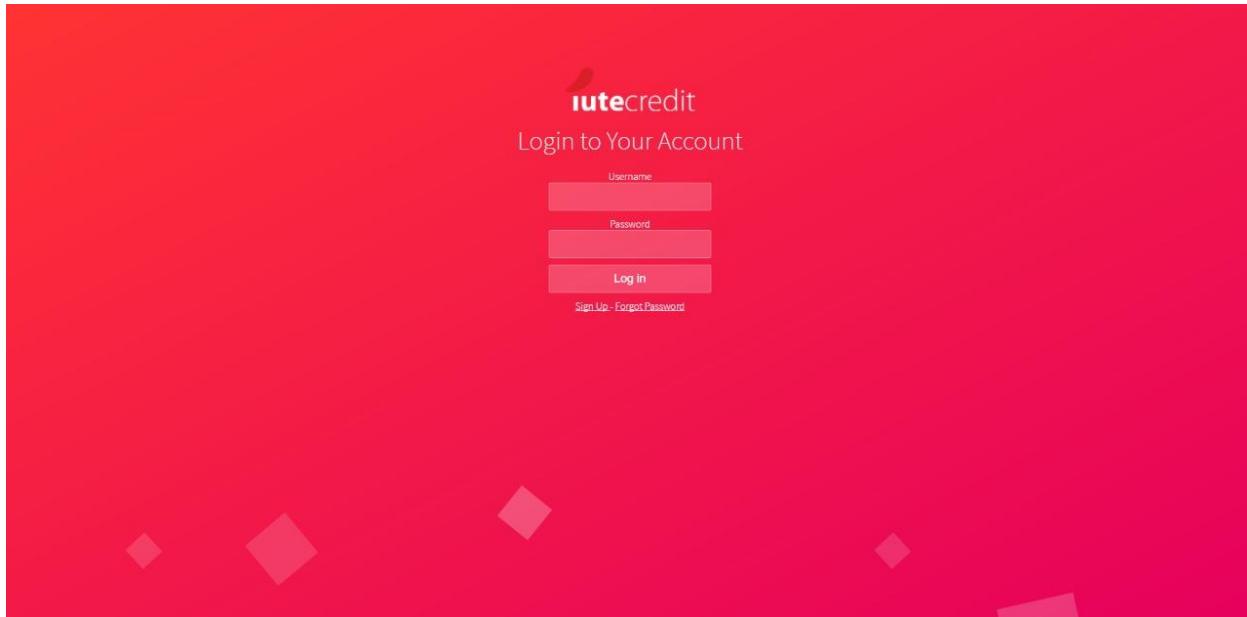
## **CLIENT- SIGN UP**



The image shows a client sign-up form titled "Sign Up". The form is divided into two columns. The left column contains fields for Name, Surname, Salary, and Bank Account Nr. The right column contains fields for Email, Username, Password, Confirm Password, Save, and Submit. The "Save" and "Submit" buttons are located at the bottom of their respective columns.

|                 |                   |
|-----------------|-------------------|
| Name            | Email             |
| Surname         | Username          |
| Salary          | Password          |
| Bank Account Nr | Confirm Password: |
| Card Id         | Save              |
|                 | Submit            |

## **CLIENT -LOG IN**



The image shows a client log-in form titled "Login to Your Account". It features two input fields for "Username" and "Password", followed by a "Log in" button. Below the form is a link for "Sign Up - Forget Password". The background of the page is red with white diamond shapes.

|          |
|----------|
| Username |
| Password |
| Log in   |

[Sign Up - Forget Password](#)

# CLIENT- DASHBOARD

The dashboard consists of four main sections:

- Edit Profile:** A red section for updating personal information. It includes fields for Name (Linda), Surname (shkrepka), Bank Account Nr (1291), Card id (10291), Email (a@xam.com), and Password.
- Loans:** A purple section showing loan information. It lists a single loan with the following details:

| Amount | Interest | Monthly Payment | Maturity | Data Fillmit | Deadline   | Status     |       |
|--------|----------|-----------------|----------|--------------|------------|------------|-------|
| 100000 | 0.17     | 16674.93        | 3108     | 6 Months     | 2017-06-10 | 06/16/2017 | Aktiv |
- Change Password:** A white section for password management. It includes fields for Current Password and Confirm Password, and a **SUBMIT** button.
- Requested Loans:** A green section showing requested loans. It lists a single loan with the following details:

| Amount | Interest | Monthly Payment | Maturity | Data Fillmit | Deadline   |            |
|--------|----------|-----------------|----------|--------------|------------|------------|
| 50000  | 0.17     | 10004.95        | 8918648  | 6 Months     | 2017-06-10 | 2017-12-10 |

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 is 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 users.   |

|    |                        |   |
|----|------------------------|---|
| 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 prints 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       | <ol style="list-style-type: none"><li>1. The user opens the login page.</li><li>2. The system asks the user for his information like password or username.</li><li>3. The user enters the password and the username.</li><li>4. The system validates the password and the username and then logs the user into their account.</li></ol>   |
| Alternative flow | <ol style="list-style-type: none"><li>1. The user opens the login page.</li><li>2. The system asks the user for his information like password or username.</li><li>3. The user enters the password and the username.</li><li>4. The password and the username entered are not correct.</li><li>5. The system shows an error, and the page is loaded again in the log in page.</li></ol> |

## (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     | <ol style="list-style-type: none"><li>1. The user opens the login page.</li><li>2. The system asks the user for his information like password or username.</li><li>3. The user enters the password and the username.</li><li>4. The system validates the password and the username and then logs the user into their account.</li><li>5. User works in his session.</li><li>6. User finds the log out button and logs out.</li></ol>  |
| Basic flow 2     | <ol style="list-style-type: none"><li>1. The user opens the login page.</li><li>2. The system asks the user for his information like password or username.</li><li>3. The user enters the password and the username.</li><li>4. The system validates the password and the username and then logs the user into their account.</li><li>5. User works in his session.</li><li>6. After a period of inactivity, the system logs out, without the user having to do anything.</li></ol> |

|                  |  |
|------------------|--|
|                  | need for a log out button.   |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. User works in his session.</li> <li>6. User wants to log out, but he there does not exist a log out button.</li> </ol> |

## (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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page.</li> <li>6. User goes to <b><i>setting</i></b> menu option.</li> <li>7. User can edit all his information that is in this page.</li> <li>8. User saves the changes made.</li> </ol> |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> </ol>   |

|  |  |
|--|--|
|  | <ol style="list-style-type: none"> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page.</li> <li>6. User goes to <b><i>setting</i></b> menu option.</li> <li>7. User can edit all his information that is in this page.</li> <li>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.)</li> <li>9. Changes cannot be saved.</li> <li>10. The system asks the user to enter again the information he want to change, based on the rule of validation.</li> </ol> |
|--|--|

## (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     | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page.</li> <li>6. User goes to <b><i>manage user's</i></b> menu option.</li> <li>7. Admin search the employee/client/business by name, surname, id or other data.</li> <li>8. The search is successful. The employee/client/loan exist in the database.</li> </ol> |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> </ol>   |

|  |   |
|--|---|
|  | <ol style="list-style-type: none"><li>3. The user enters the password and the username.</li><li>4. The system validates the password and the username and then logs the user into their account.</li><li>5. The system opens automatically in the <b><i>home</i></b> page.</li><li>6. User goes to <b><i>manage user's</i></b> menu option.</li><li>7. Admin search the employee/client/business by name, surname, id or other data.</li><li>8. The search is unsuccessful. The employee/client/loan doesn't exist in the database.</li></ol> |
|--|---|

## (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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page where all loans are showed.</li> <li>6. User check all the loans</li> </ol>                       |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page where all loans are showed.</li> <li>6. The system does not load the <b>home</b> page.</li> </ol> |

## (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         | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page where all loans are showed.</li> <li>6. Admin can check the amount of money available.</li> <li>7. User can have changed this amount if he wants.</li> </ol> |
| Alternative flow 1 | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page where all loans are showed.</li> </ol>   |

|                    |   |
|--------------------|---|
|                    | <p>6. The amount available it's not shown.</p>  |
| Alternative flow 2 | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page where all loans are showed.</li> <li>6. The amount available is shown.</li> <li>7. User tries to change the amount. The change is not saved.</li> </ol> |

## (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 <b>Statements</b> 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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page.</li> <li>6. User goes to <b>Statements</b> menu option.</li> <li>7. User select <b>Get income statement</b> option.</li> <li>8. The income statement is shown.</li> <li>9. User can print the statement.</li> </ol> |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like passw or user</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page.</li> <li>6. User goes to <b>Statements</b> menu option.</li> <li>7. User selects <b>Get income statement</b> option.</li> <li>8. The income statement is not shown.</li> </ol>                                      |

## (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 <b>Statements</b> 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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page.</li> <li>6. User goes to <b>Statements</b> menu option.</li> <li>7. User select <b>Get balance sheet statement</b> option.</li> <li>8. The balance sheet statement is shown.</li> </ol>     |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page.</li> <li>6. User goes to <b>Statements</b> menu option.</li> <li>7. User select <b>Get balance sheet statement</b> option.</li> <li>8. The balance sheet statement is not shown.</li> </ol> |

## (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 <b><i>Statements</i></b> 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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page.</li> <li>6. User goes to <b><i>Statements</i></b> menu option.</li> <li>7. User select <b><i>Get cash flow statement</i></b> option.</li> <li>8. The cash flow statement is shown.</li> </ol>      |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like pass or user.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page.</li> <li>6. User goes to <b><i>Statements</i></b> menu option.</li> <li>7. User select <b><i>Get cash flow statement</i></b> option.</li> <li>8. The cash flow statement is not shown.</li> </ol> |

## (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: <b><i>Home</i></b> .  |
| Precondition     | First must have been completed the login case.   |
| Priority         | Primary and essential  |
| Frequency of use | Many times, per day  |
| Basic flow       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page where all loans are showed.</li> <li>6. Employee can check all the loans that are shown in the desktop.</li> </ol> |
| 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 <b>home</b> 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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page where all loans are showed.</li> <li>6. The user click <b>Pay loan</b> button.</li> <li>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.</li> <li>8. User enters the information.</li> <li>9. The information is correct and validated.</li> <li>10. User clicks the button <b>Pay Loan</b>.</li> <li>11. System asks: “Are you sure you want to pay the loan?”.</li> <li>12. Click “Yes”. The amount of loan that needs to be payed</li> </ol> |

|                  |   |
|------------------|---|
|                  | decreases, and the available amount increases.  |
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page where all loans are showed.</li> <li>6. The user click <b><i>Pay loan</i></b> button.</li> <li>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.</li> <li>8. User enters the information.</li> <li>9. The information is not correct. For example: ID of the loan doesn't exist.</li> <li>10. System gives an alert: "The loan with this ID doesn't exist."</li> <li>11. User enters again the information.</li> <li>12. The information is correct.</li> <li>13. System asks: "Are you sure you want to pay the loan?"</li> <li>14. Click "Yes". The amount of loan that needs to be payed decreases, and the available amount increases.</li> </ol> |

## (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 <b>home</b> 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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page where all loans are showed.</li> <li>6. Employee clicks <b>Give loan</b> button.</li> <li>7. System opens a new window, which asks the client information.</li> <li>8. The information entered is correct and validated.</li> <li>9. System check if there is enough balance.</li> <li>10. The balance is enough for the loan.</li> <li>11. System asks: “Are you sure you want to give this loan”.</li> <li>12. Available amount shown in <b>Home</b> view changes.</li> </ol> |

|                  |   |
|------------------|---|
| Alternative flow | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b><i>home</i></b> page where all loans are showed.</li> <li>6. Employee clicks <b><i>Give loan</i></b> button.</li> <li>7. System opens a new window, which asks the client information.</li> <li>8. The information entered is not correct.</li> <li>9. An alert is shown which says “Please, enter correct information!”</li> <li>10. User enters again the information. This time information is correct.</li> <li>11. System check if there is enough balance.</li> <li>12. The balance is not enough for the loan.</li> <li>13. System gives an alert “There is not enough balance. Please, check the amount of money you entered!”.</li> <li>14. User enters the max amount available.</li> <li>15. System asks: “Are you sure you want to give this loan”.</li> <li>16. Available amount shown in <b><i>Home</i></b> view changes.</li> </ol> |
|------------------|---|

### (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       | <ol style="list-style-type: none"> <li>1. The user opens the login page.</li> <li>2. The system asks the user for his information like password or username.</li> <li>3. The user enters the password and the username.</li> <li>4. The system validates the password and the username and then logs the user into their account.</li> <li>5. The system opens automatically in the <b>home</b> page where all loans are showed.</li> <li>6. Employee clicks <b>Delete loan</b> button.</li> <li>7. Systems prompts for permission.</li> <li>8. Loan is deleted.</li> <li>9. Available loans that are shown in <b>Home</b> view change.</li> </ol> |
| 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 communicates 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       | <ol style="list-style-type: none"><li>1. User enters username and password</li><li>2. System validates the entered information</li><li>3. Username and password are added correctly</li><li>4. Employee send message/feedback to the Admin</li><li>5. Log out</li></ol>  |
| Alternative flow | <ol style="list-style-type: none"><li>1. User enters username and password</li><li>2. System doesn't validate the entered information</li><li>3. User enters again the username and password.</li><li>4. Username and password are added correctly</li><li>5. Employee send message/feedback to the admin</li><li>6. Log out</li></ol> |

## (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       | <ol style="list-style-type: none"> <li>1. User enters username and password</li> <li>2. System validates the entered information</li> <li>3. Username and password are added correctly</li> <li>4. User enter the information like salary etc. to check if he is able to take the loan.</li> <li>5. System show if the user can take or not any loan from the IUTE.</li> <li>6. Log out</li> </ol>                                 |
| Alternative flow | <ol style="list-style-type: none"> <li>1. User enters username and password</li> <li>2. System validates the entered information</li> <li>3. Username and password are added correctly</li> <li>4. User enter the information like salary etc. to check if he is able to take the loan.</li> <li>5. The user enters incorrect info, or he does not meet the conditions of the IUTE to get the loan.</li> <li>6. Log out</li> </ol> |

## (UC\_16) Use case: Get alerts

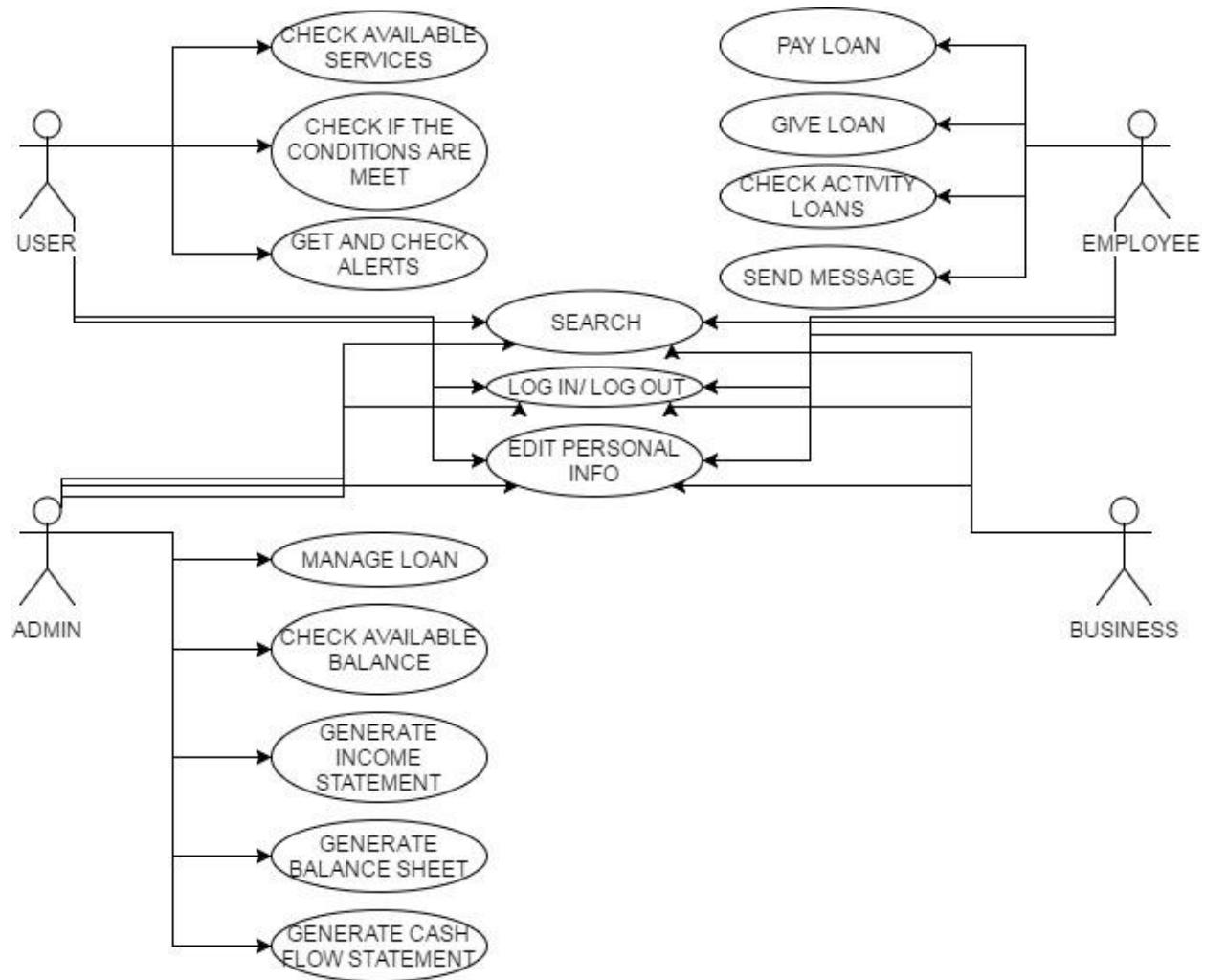
|                  |   |
|------------------|---|
| Use case number  | UC_16   |
| Use case name    | Get and check alerts  |
| Actors involved  | User  |
| Description      | This use case describes how the user can get alerts from the IUTE. Alerts are in types of: "The deadline of paying your is near" 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       | <ol style="list-style-type: none"> <li>1. User enters username and password</li> <li>2. System validates the entered information</li> <li>3. Username and password are added correctly</li> <li>4. When the payments of the loan pass the deadline to be paid the system send alert to the user.</li> <li>5. User can disable the alert after being send, or can mark it as <b><i>checked</i></b>.</li> <li>6. Log out</li> </ol> |
| Alternative flow | No alternative flow   |

## (UC\_17) Use case: Check the services available

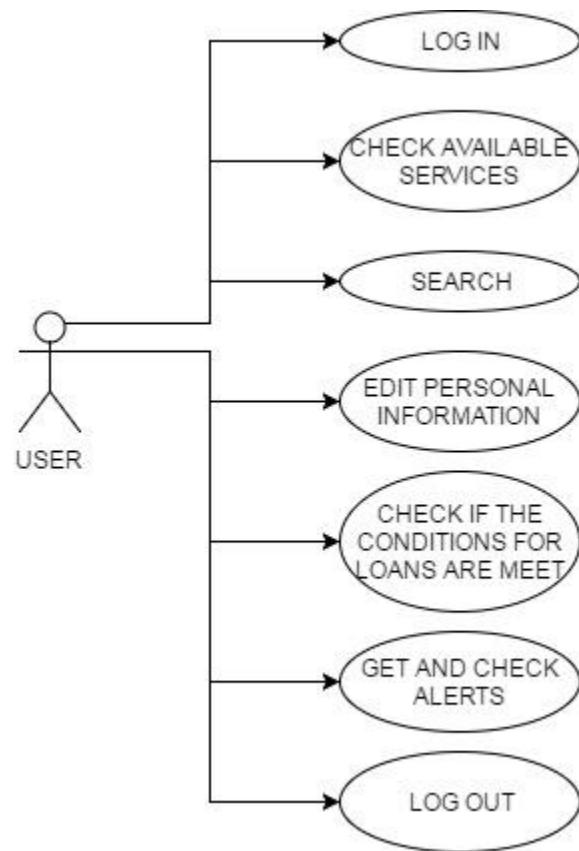
|                  |  |
|------------------|--|
| Use case number  | UC_17  |
| Use case name    | Check the services available   |
| Actors involved  | User   |
| Description      | This use case describes how the user can check all the services that IUTE credit has for the customer. Also the customer can decide if he wants to take other available service and put them in <b><i>My services</i></b> option, which is under the <b><i>Services</i></b> menu.                                  |
| Precondition     | First must have been completed the login case.   |
| Priority         |  |
| Frequency of use | Many times, per day  |
| Basic flow       | <ol style="list-style-type: none"><li>1. User enters username and password</li><li>2. System validates the entered information</li><li>3. Username and password are added correctly</li><li>4. User goes under the section My services to check for services that the company offers.</li><li>5. Log out</li></ol> |
| Alternative flow | No alternative flow  |

## **DIAGRAMS**

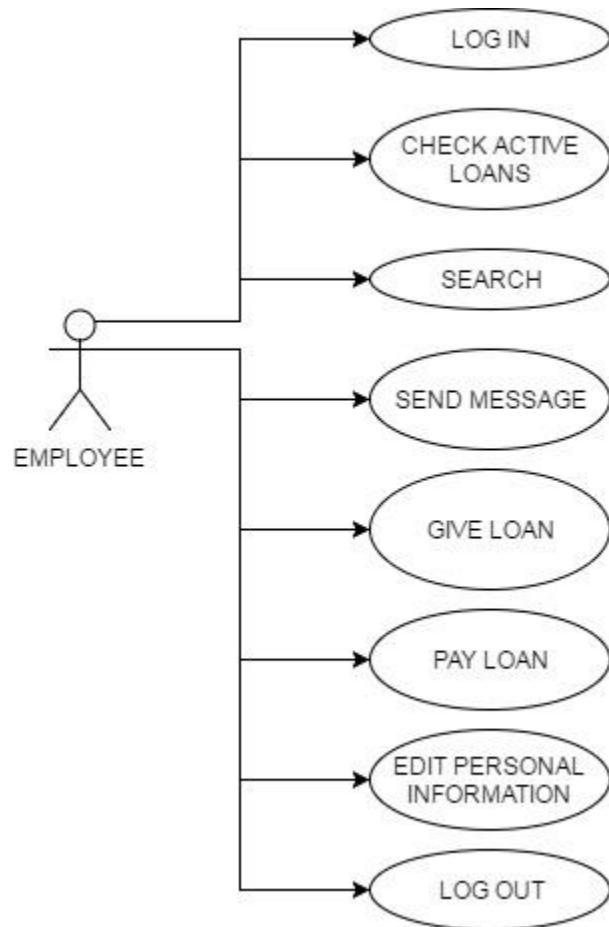
## USE CASE DIAGRAM



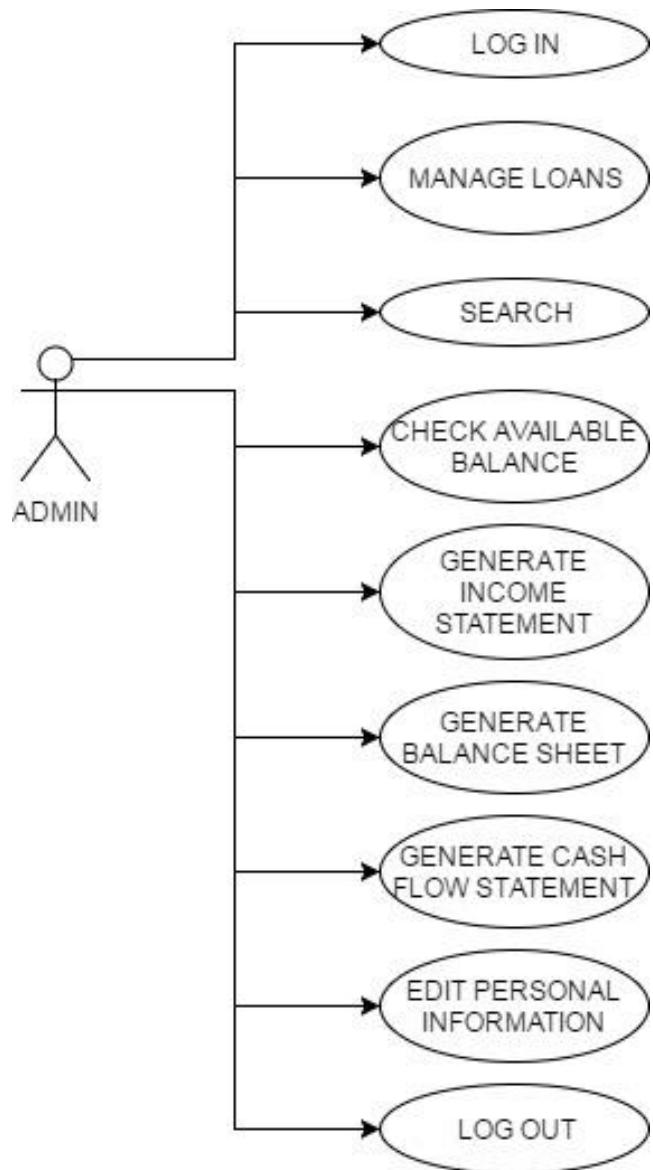
## **USER DIAGRAM**



## **EMPLOYEE DIAGRAM**

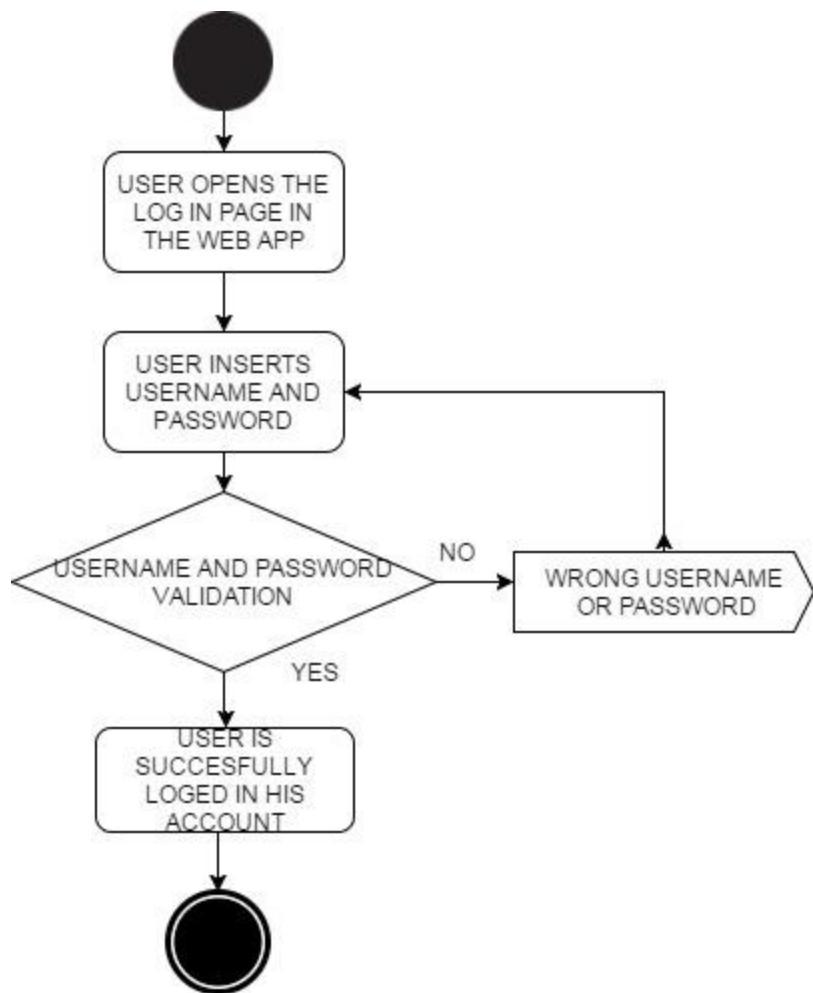


## **ADMIN DIAGRAM**

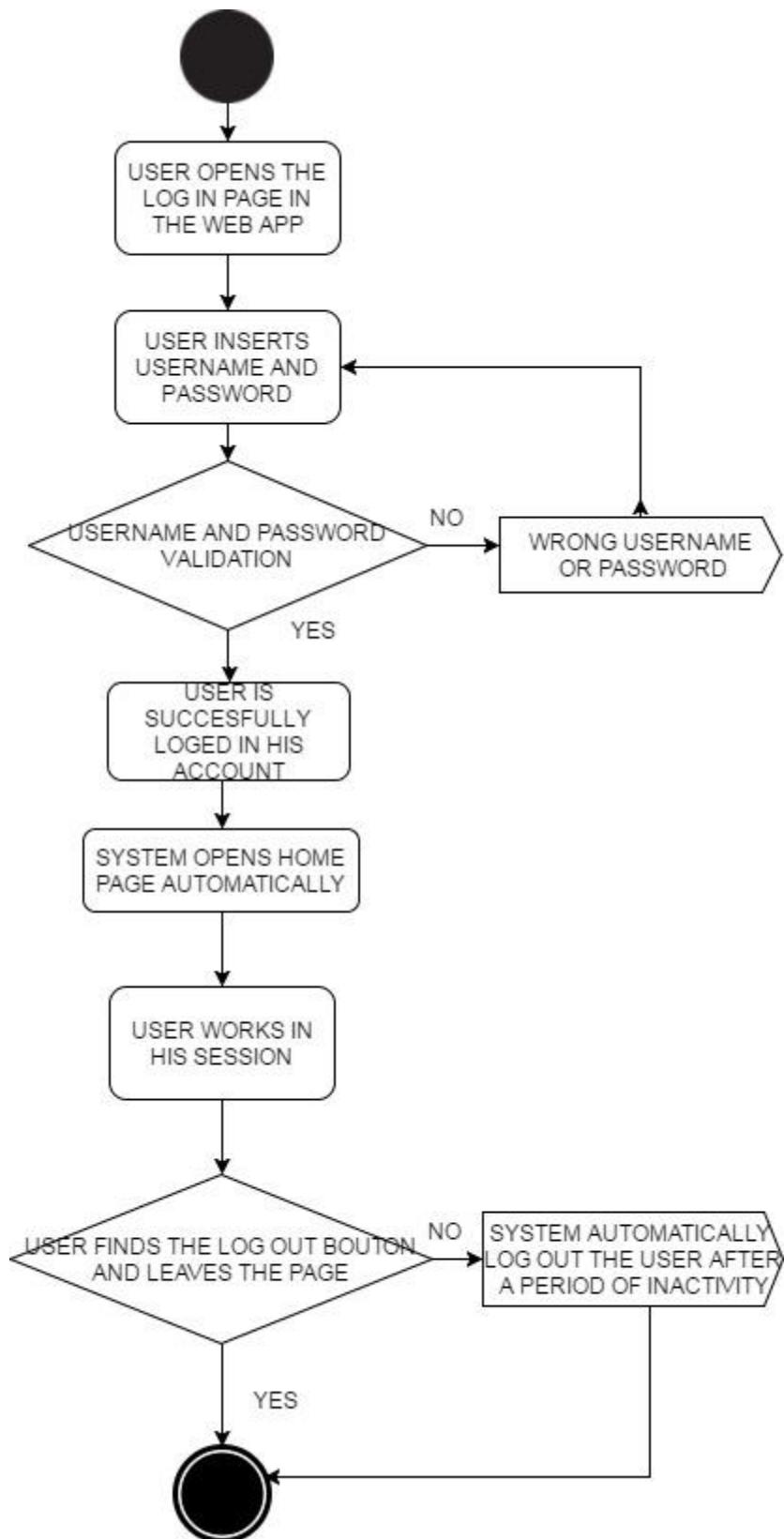


## **ACTIVITY DIAGRAM**

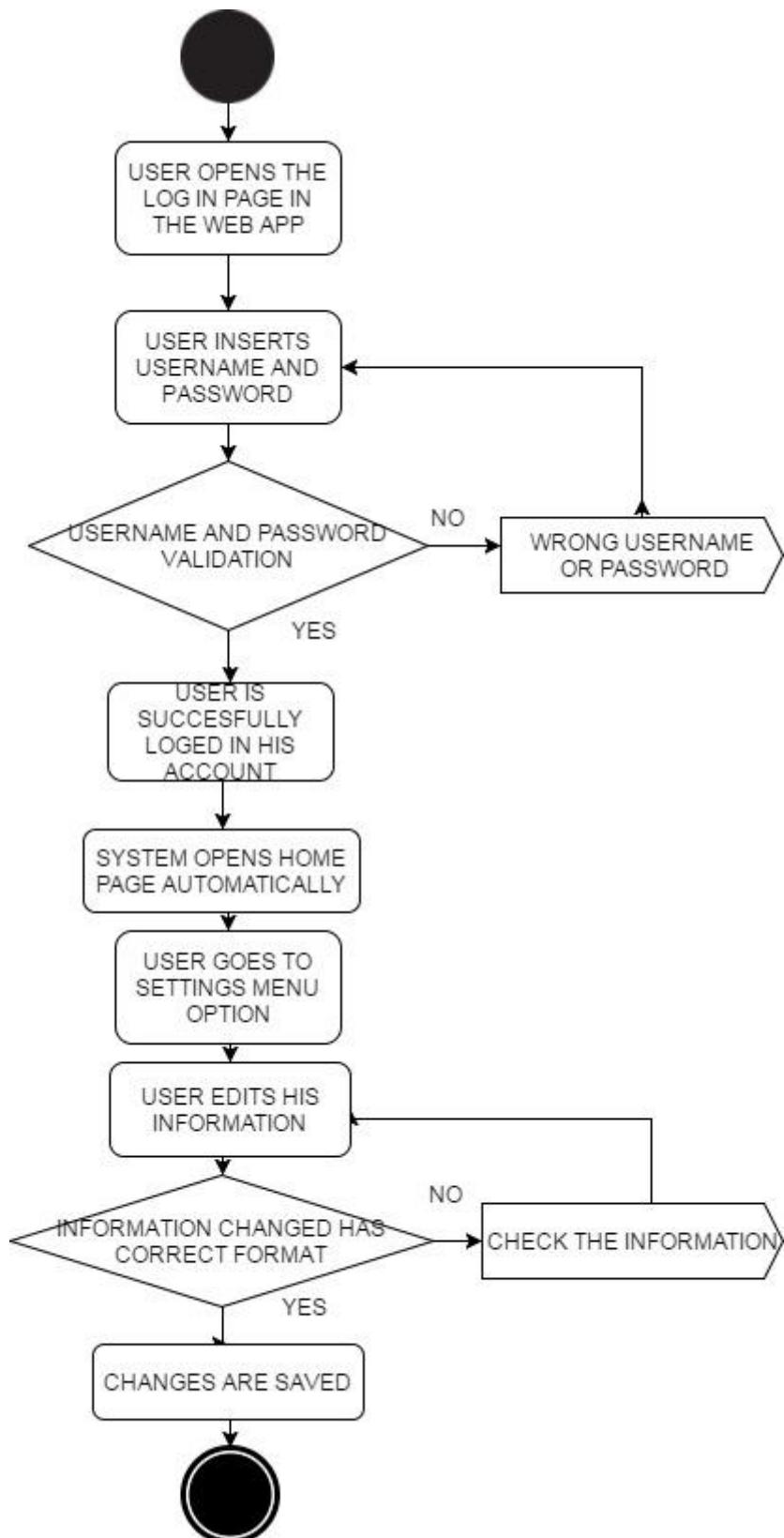
## (AD\_01) Activity Diagram of UC\_01



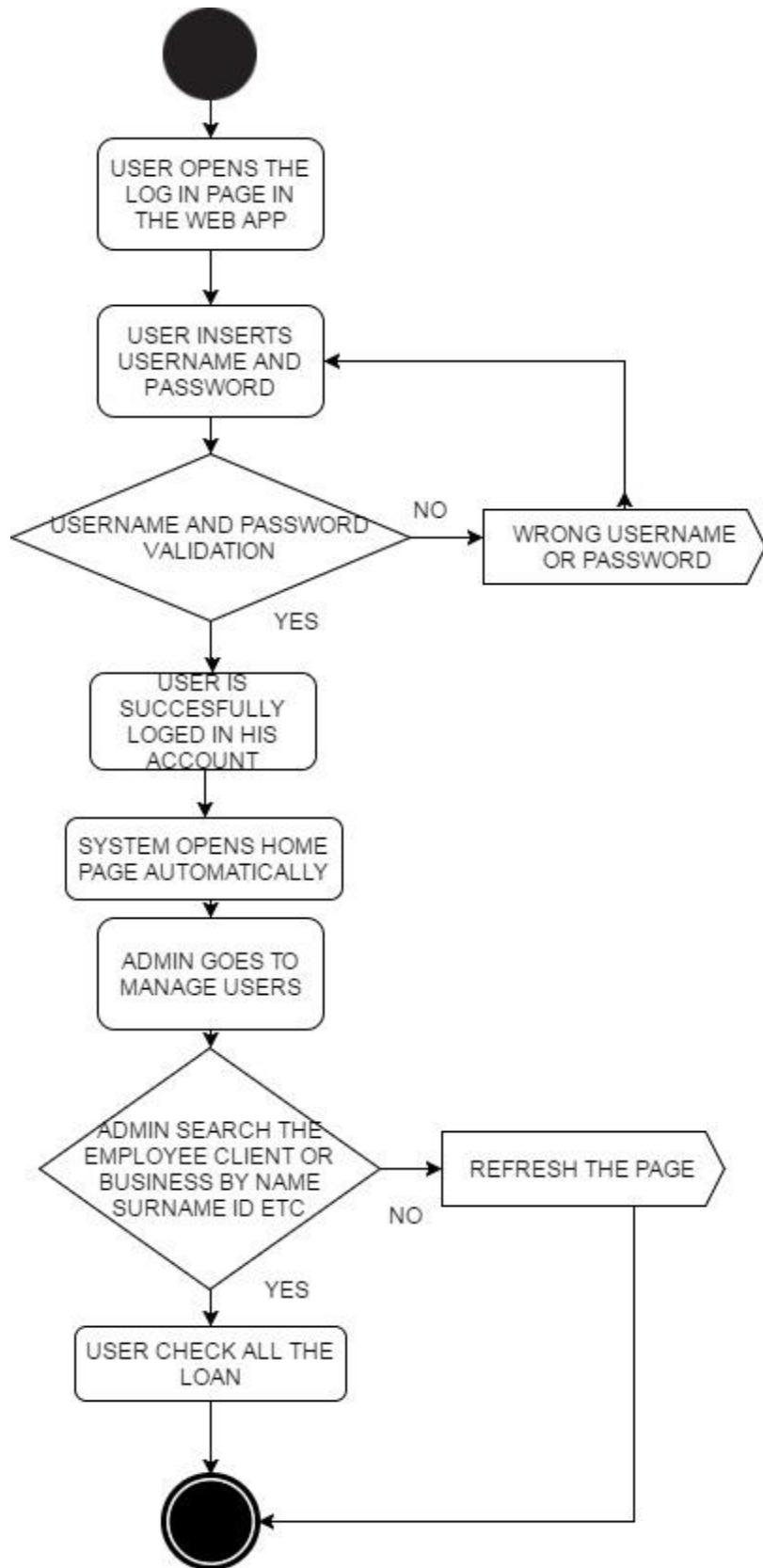
## (AD\_02) Activity Diagram of UC\_02



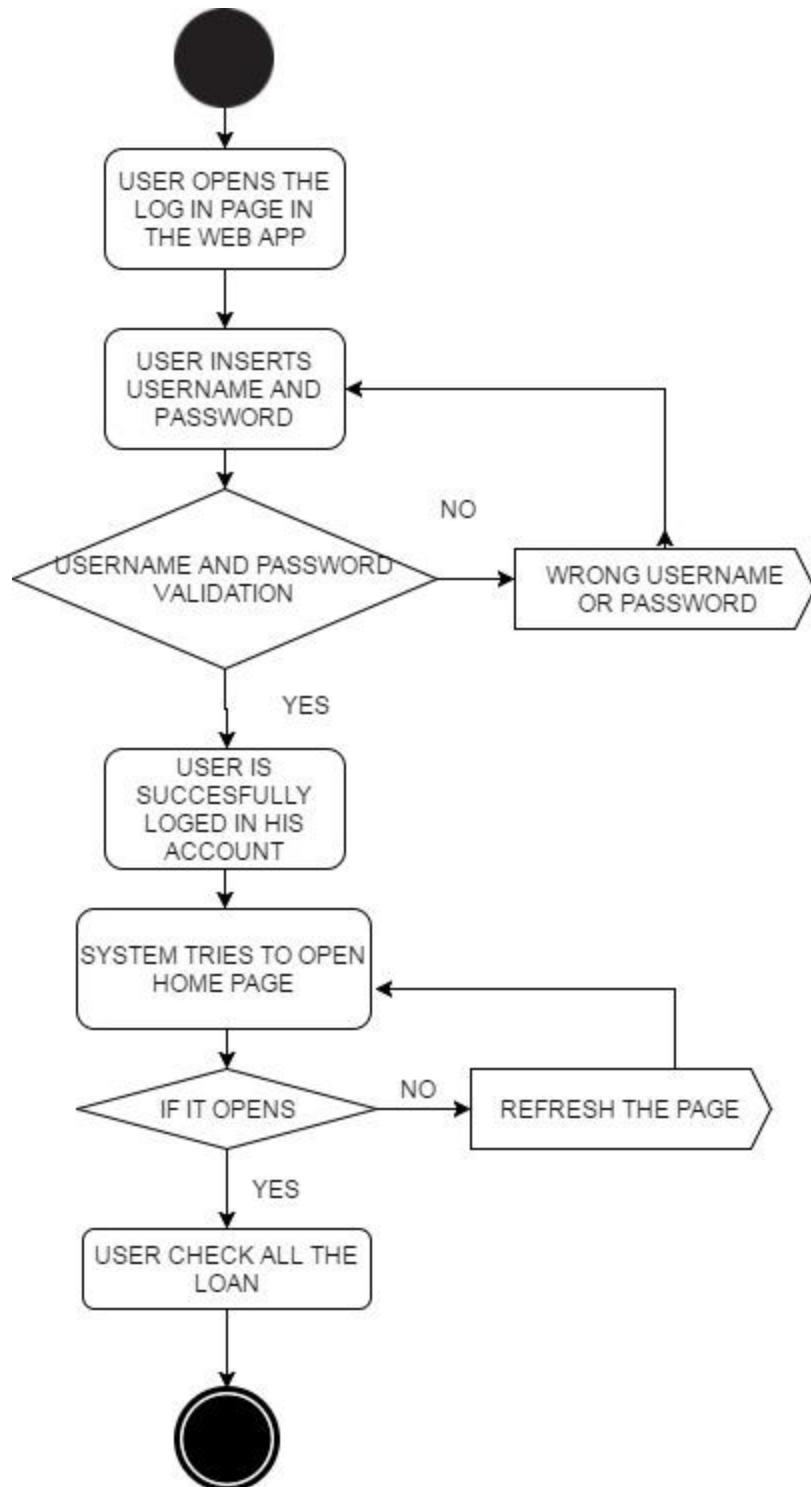
### (AD\_03) Activity Diagram of UC\_03



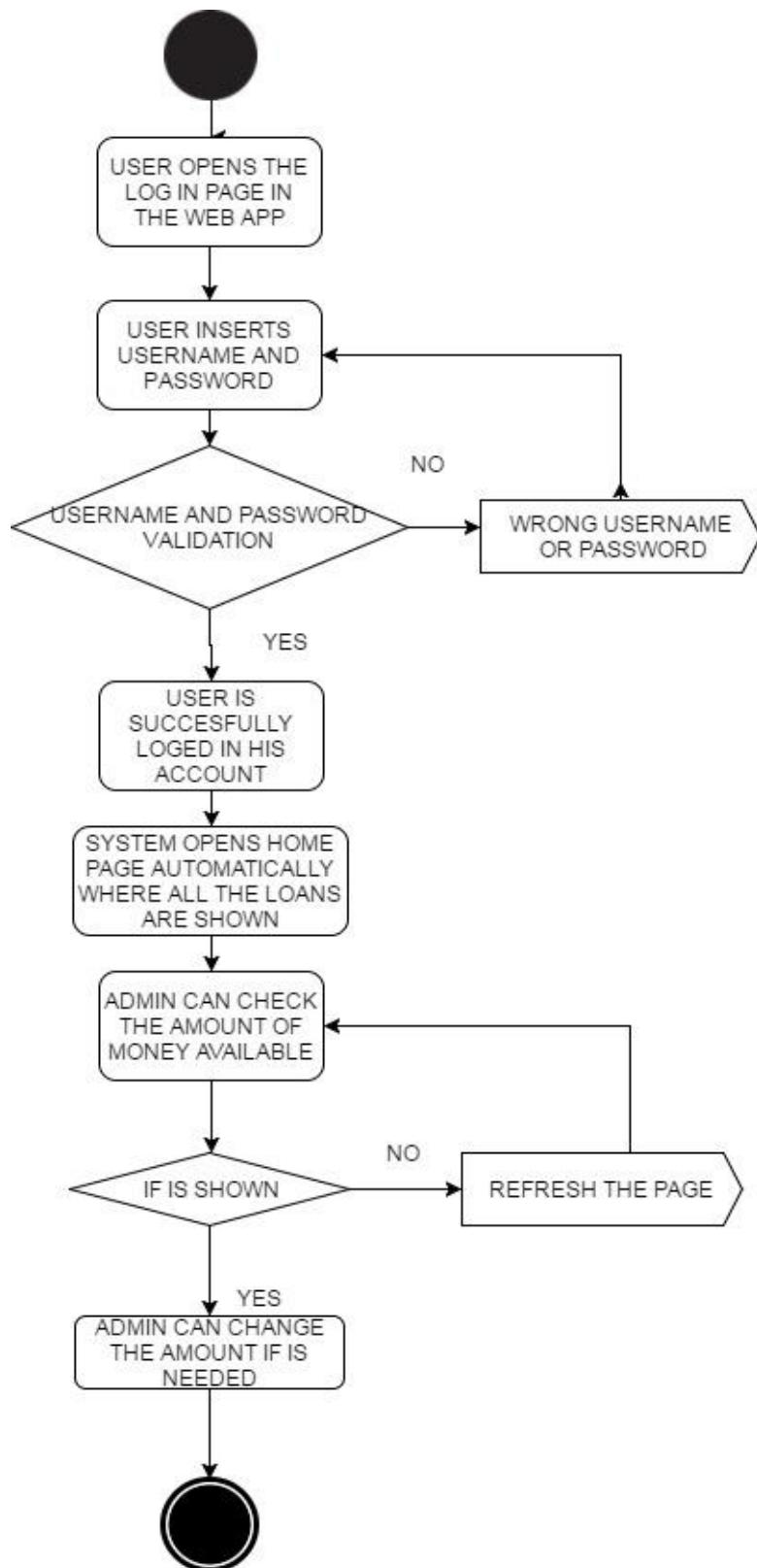
### (AD\_04) Activity Diagram of UC\_04



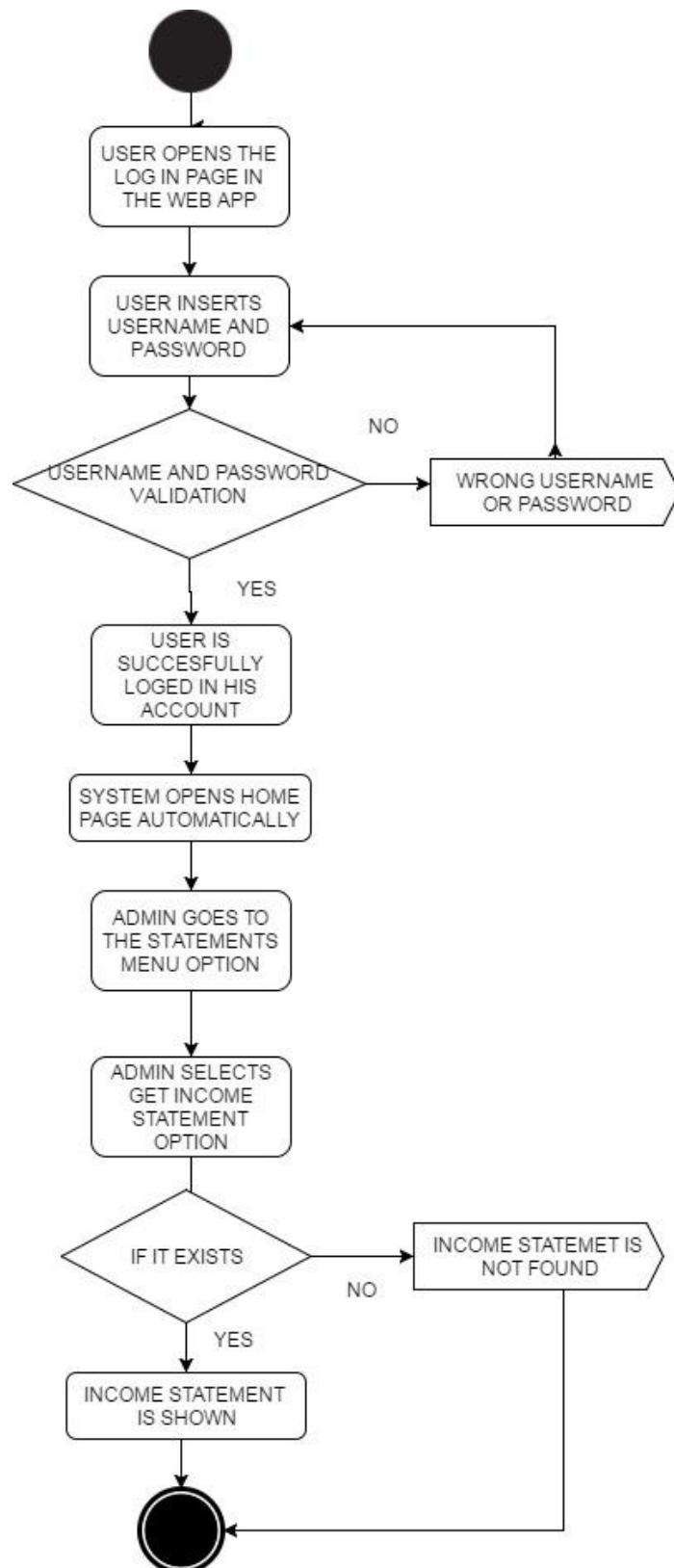
## (AD\_05) Activity Diagram of UC\_05



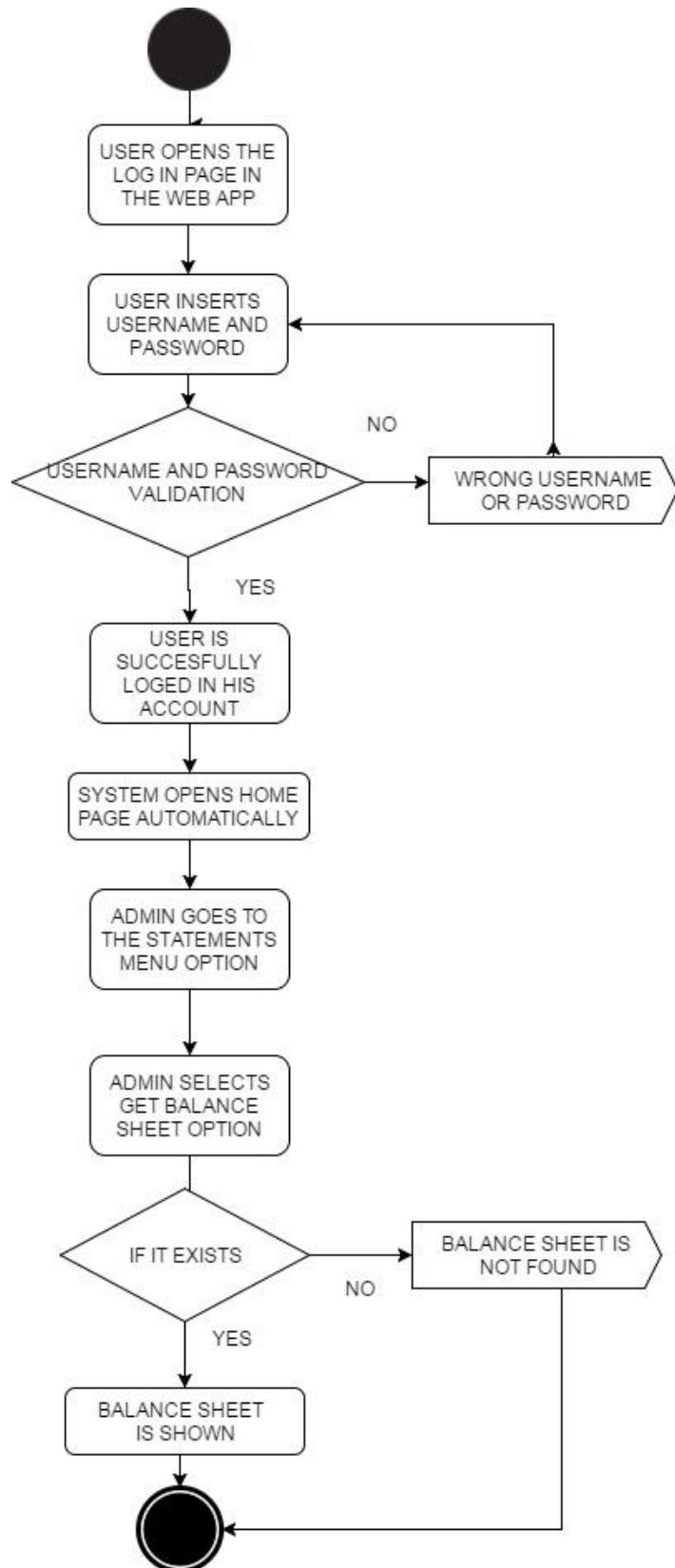
## (AD\_06) Activity Diagram of UC\_06



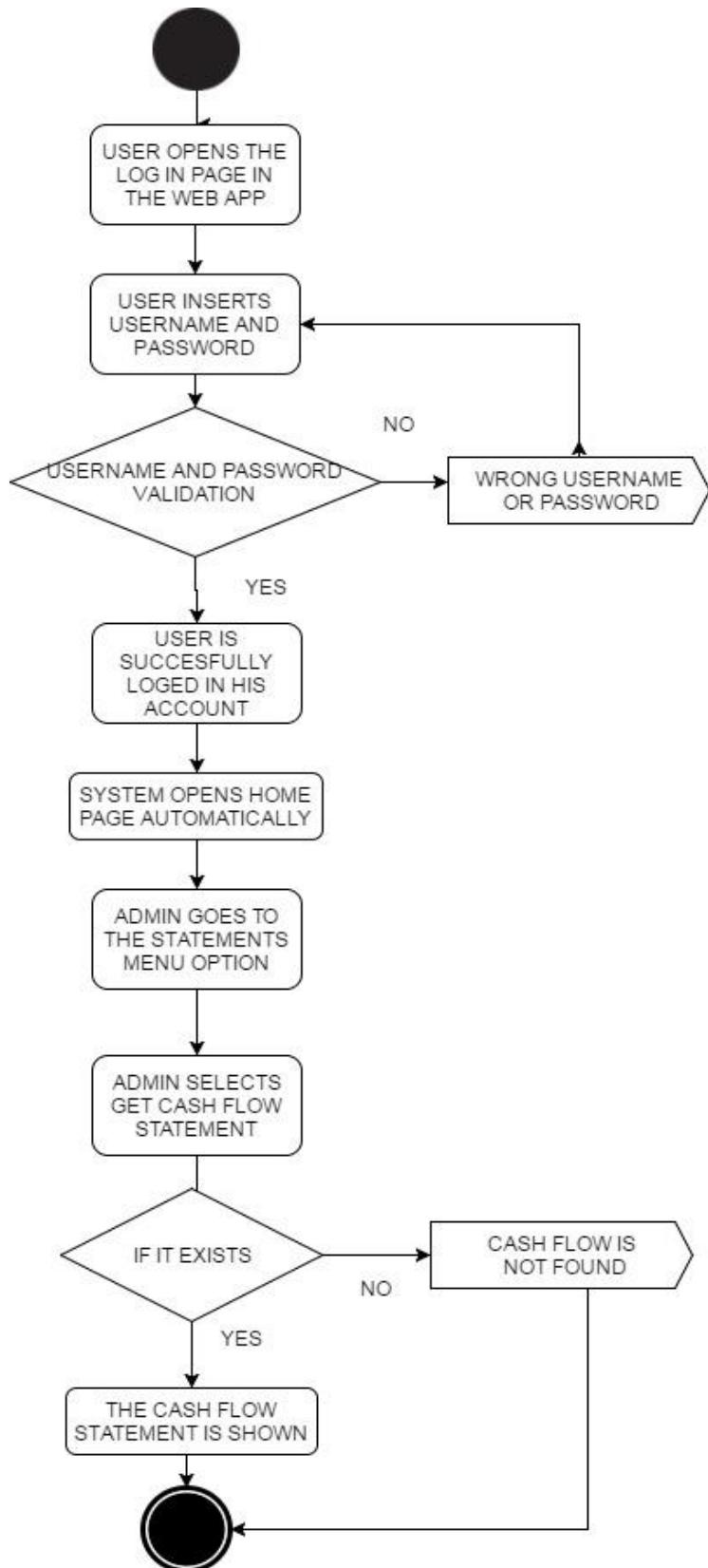
## (AD\_07) Activity Diagram of UC\_07



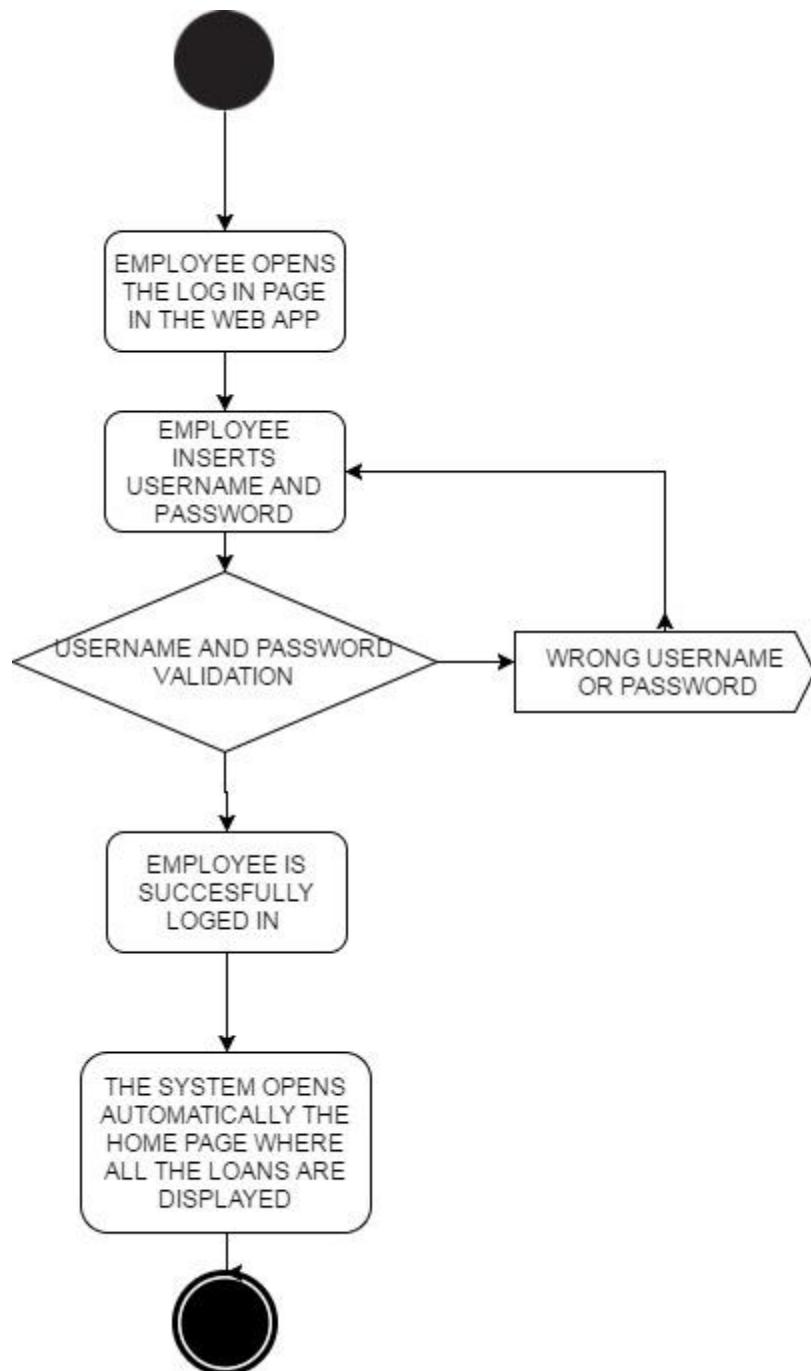
## (AD\_08) Activity Diagram of UC\_08



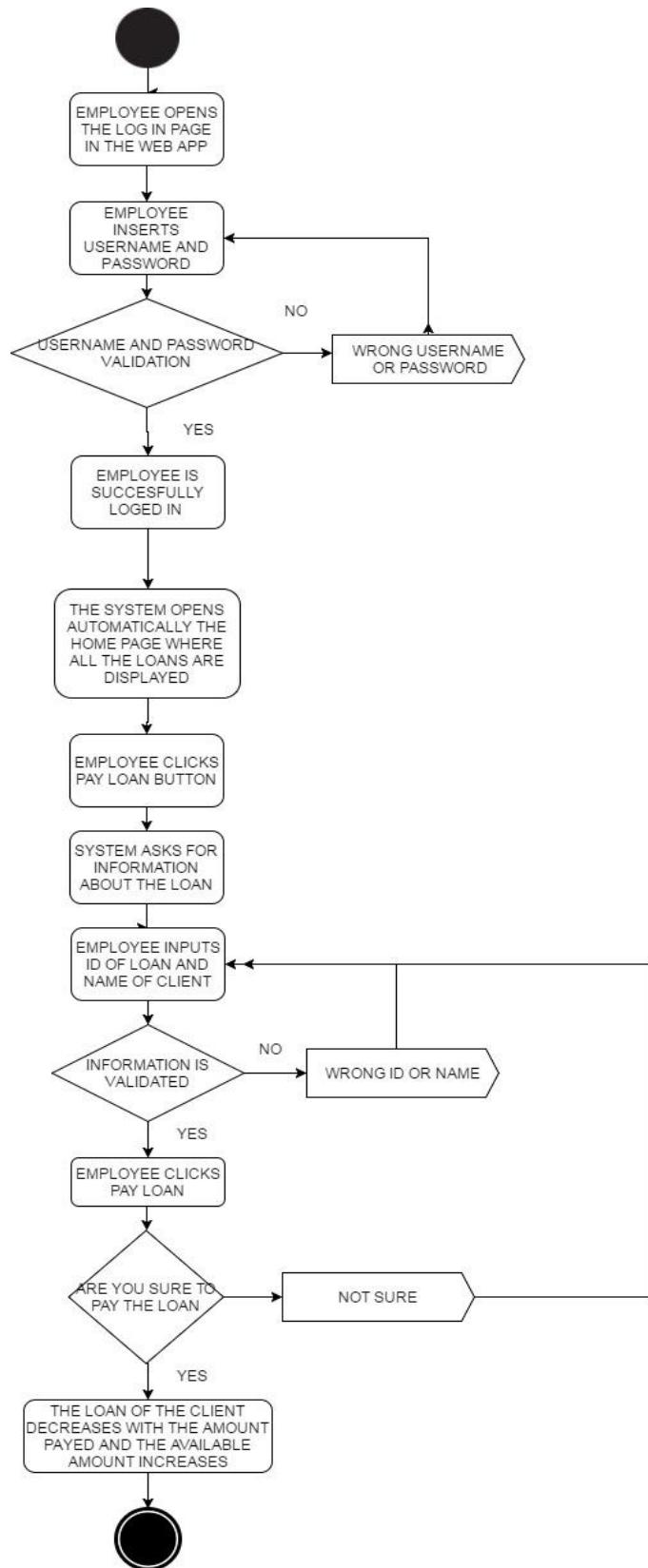
## (AD\_09) Activity Diagram of UC\_09



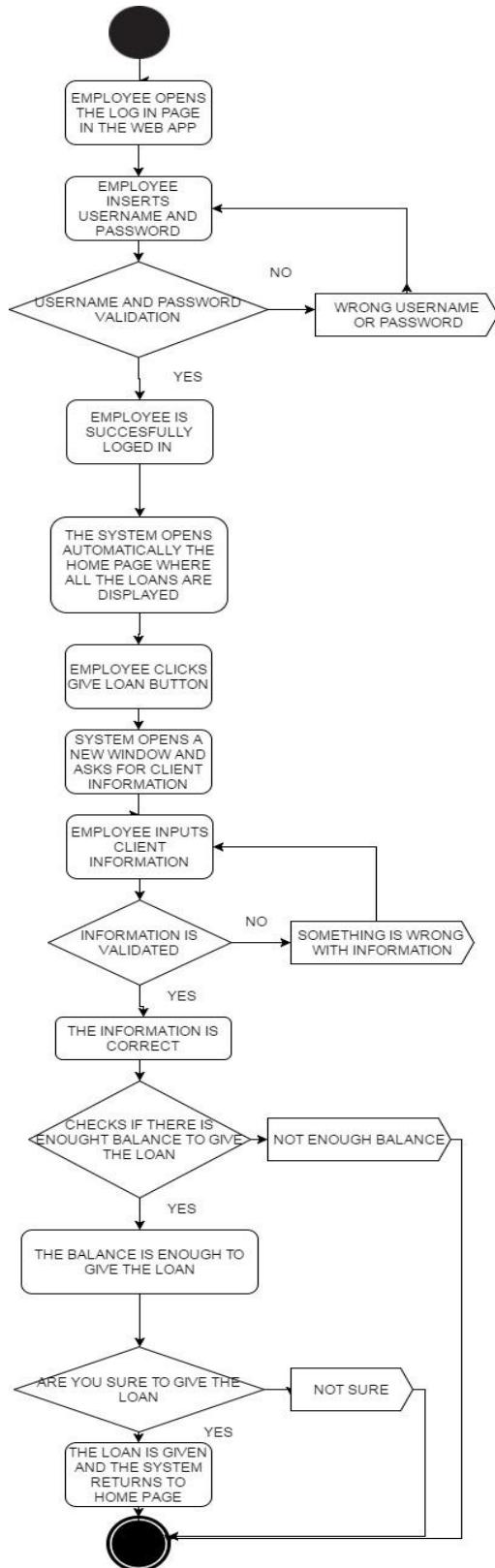
## (AD\_10) Activity Diagram of UC\_10



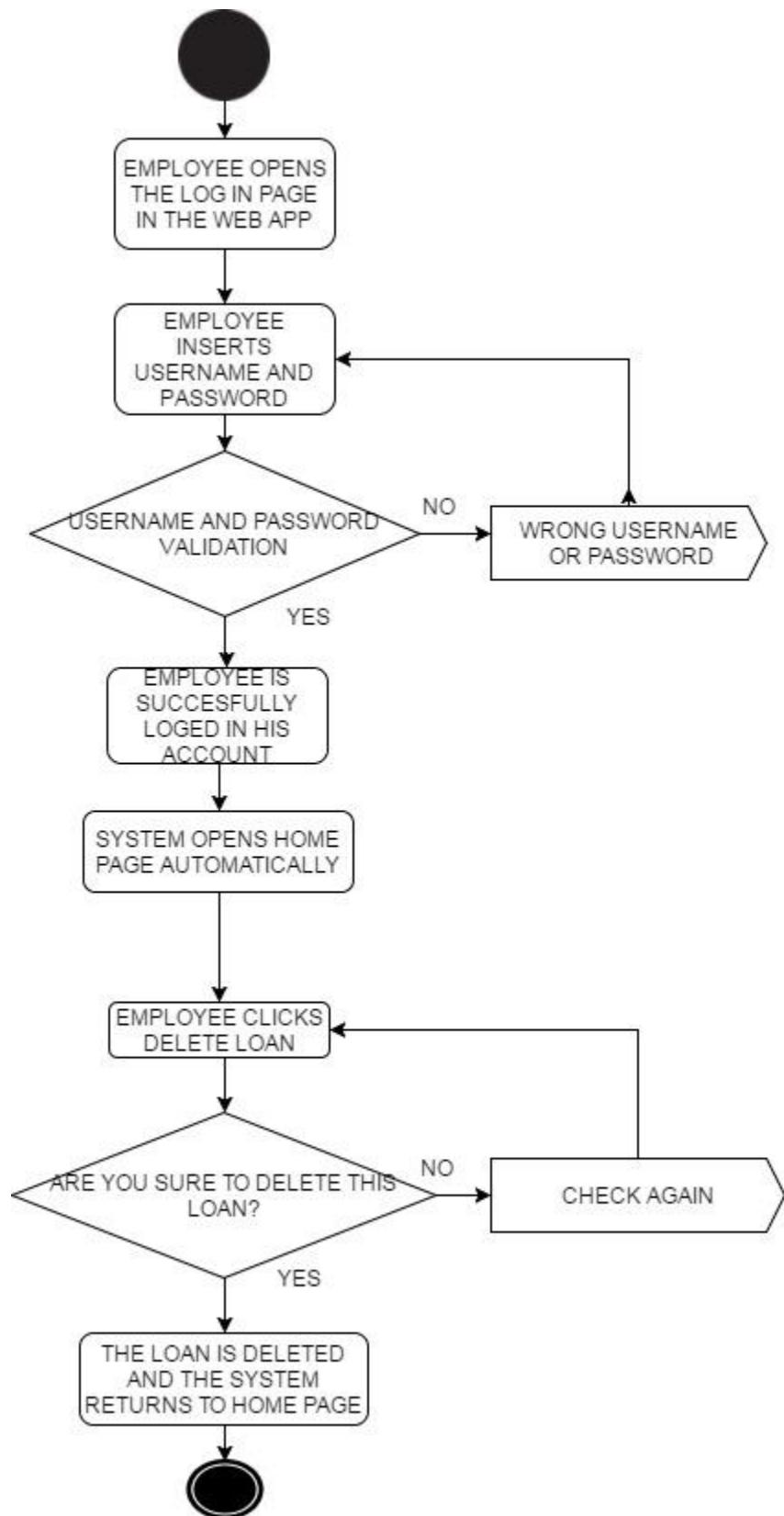
## (AD\_11) Activity Diagram of UC\_11



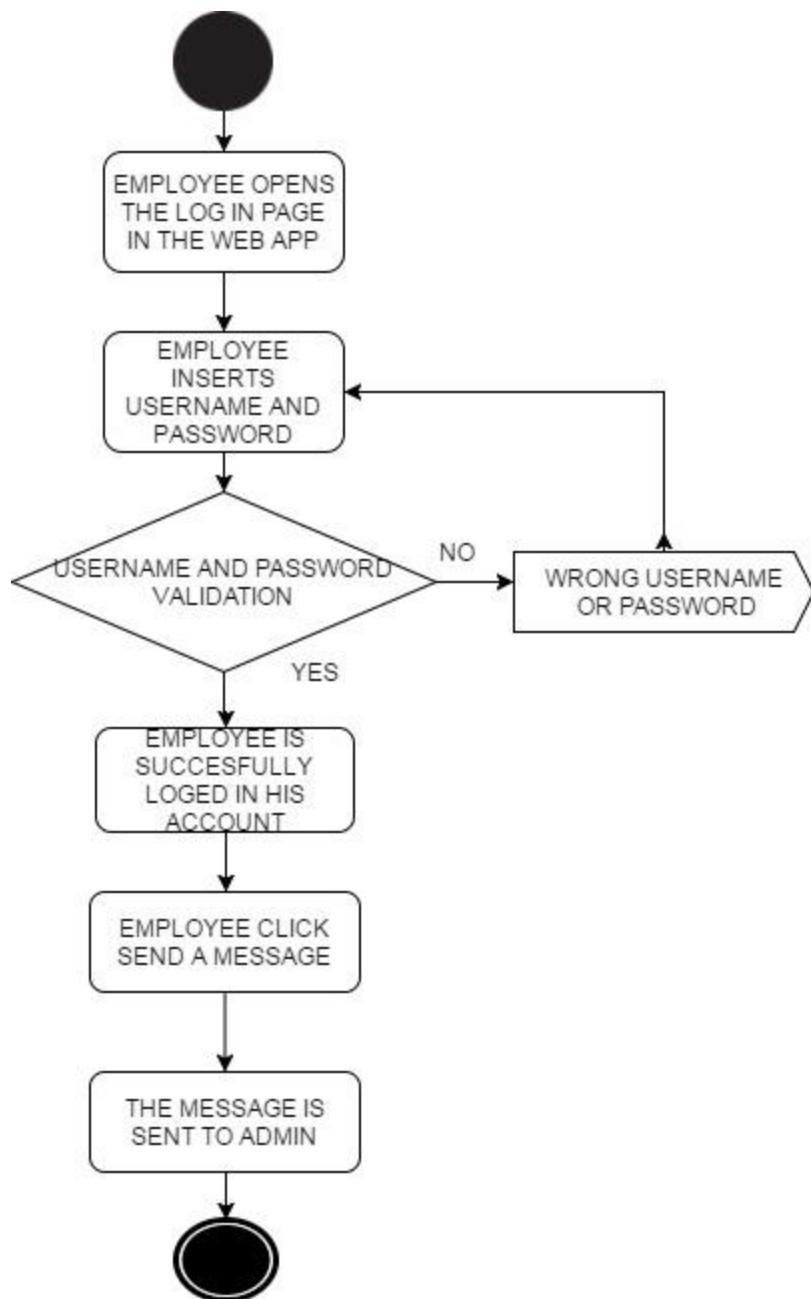
## (AD\_12) Activity Diagram of UC\_12



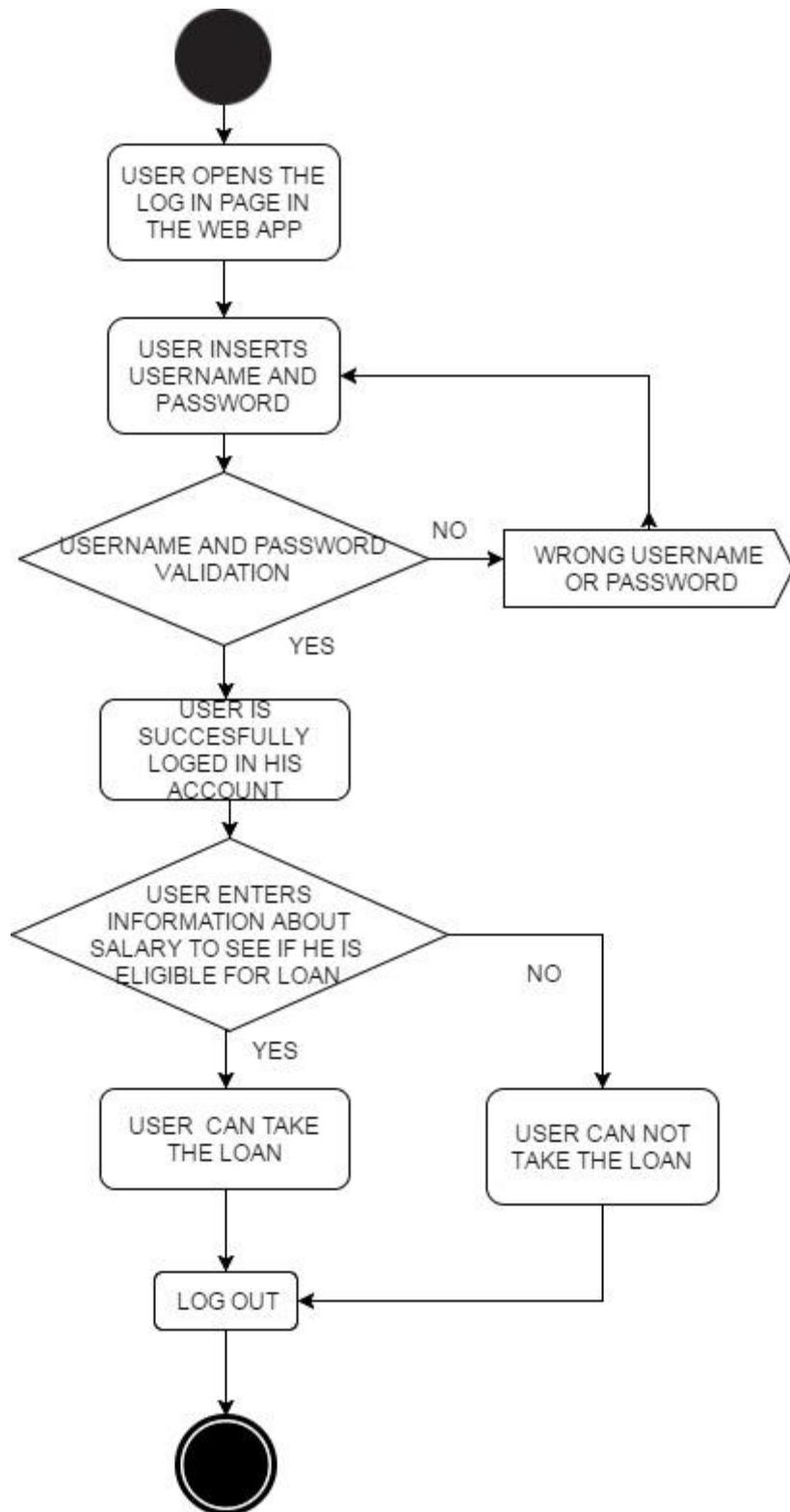
### (AD\_13) Activity Diagram of UC\_13



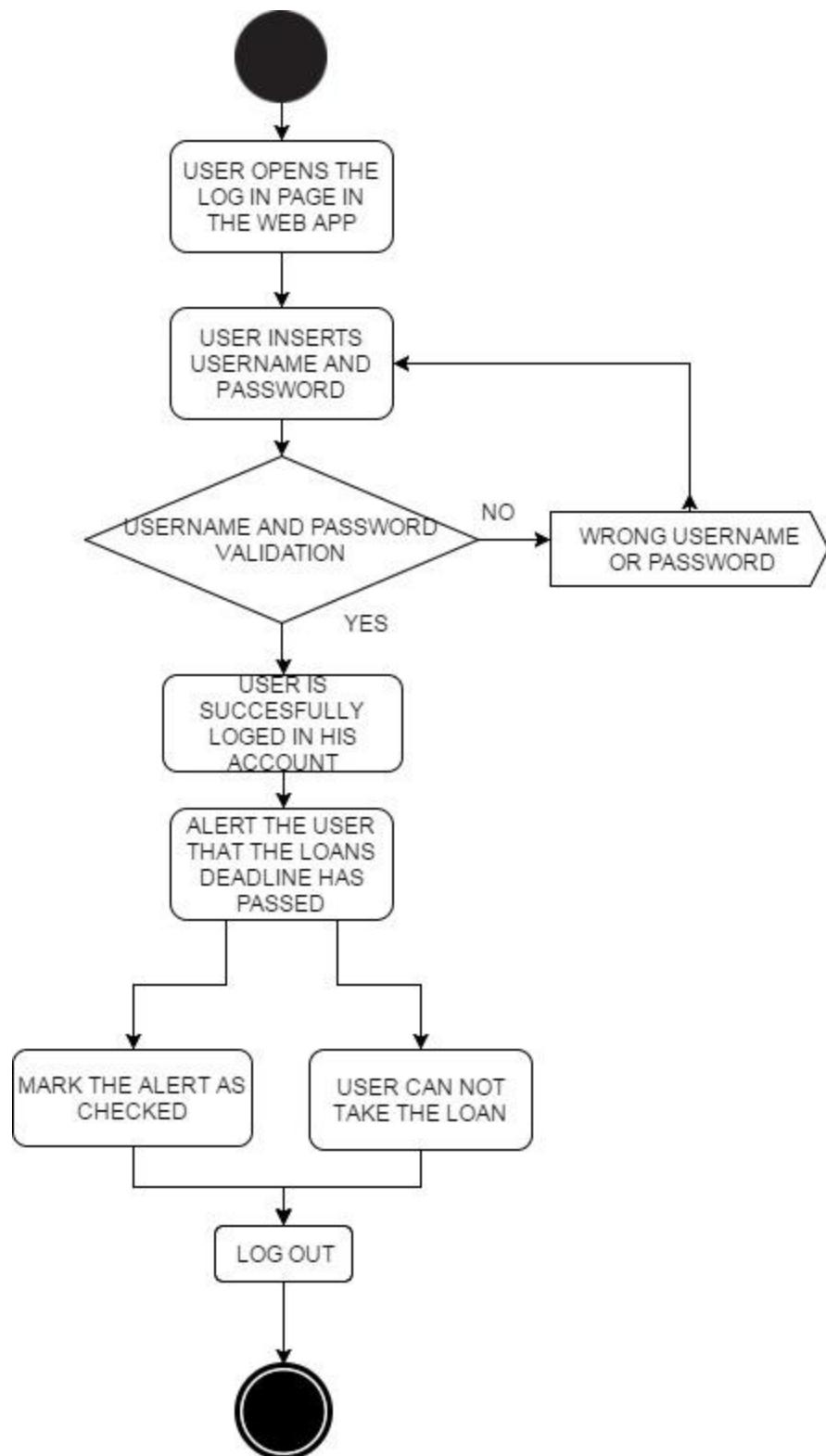
### (AD\_14) Activity Diagram of UC\_14



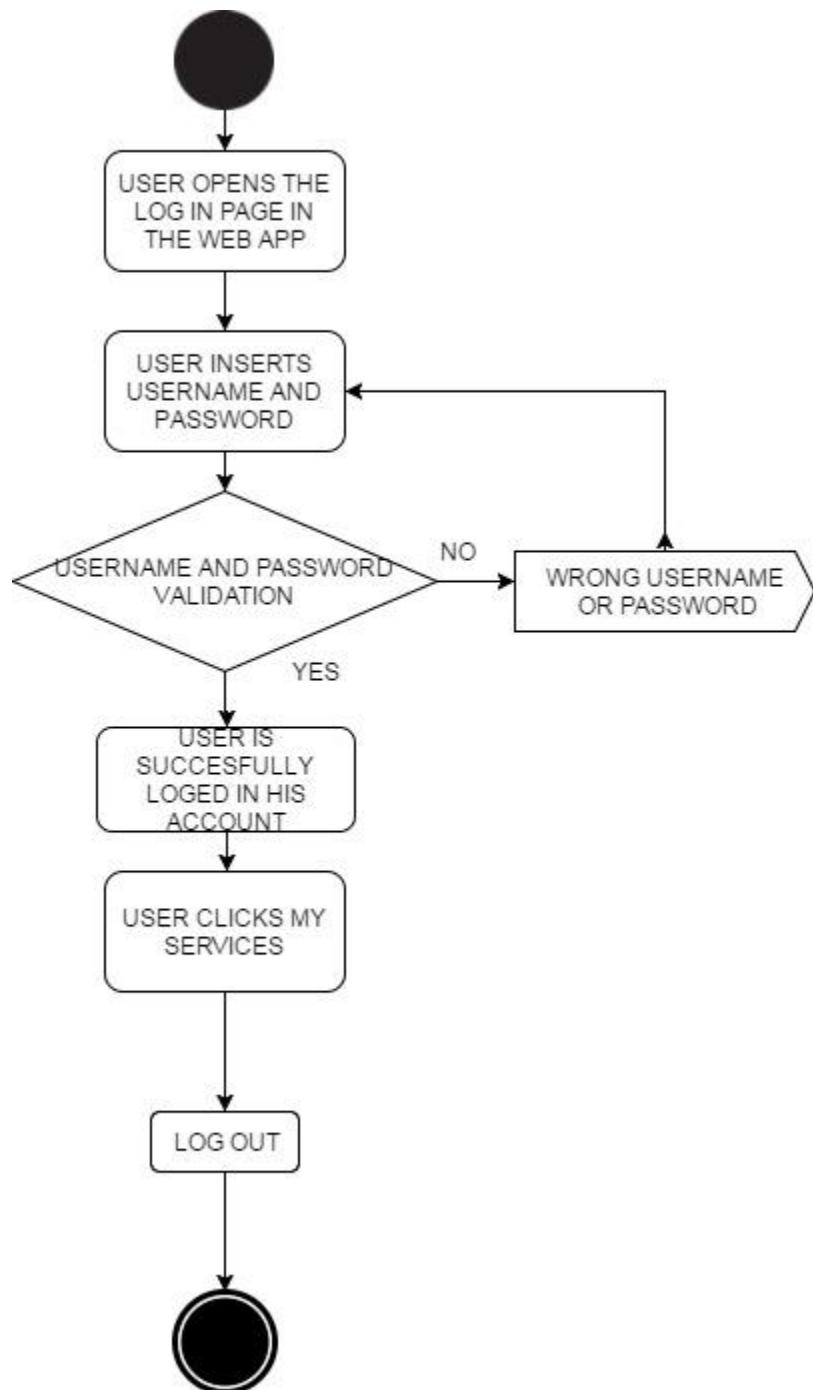
## (AD\_15) Activity Diagram of UC\_15



## (AD\_16) Activity Diagram of UC\_16

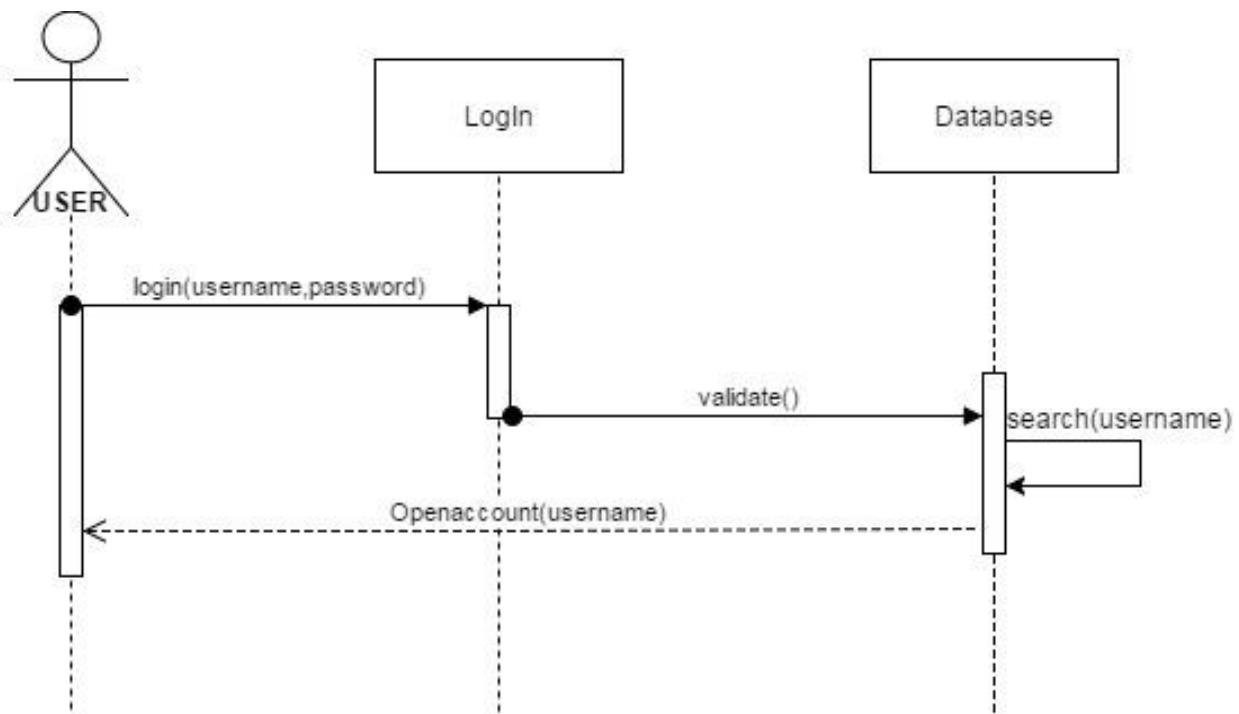


## (AD\_17) Activity Diagram of UC\_17

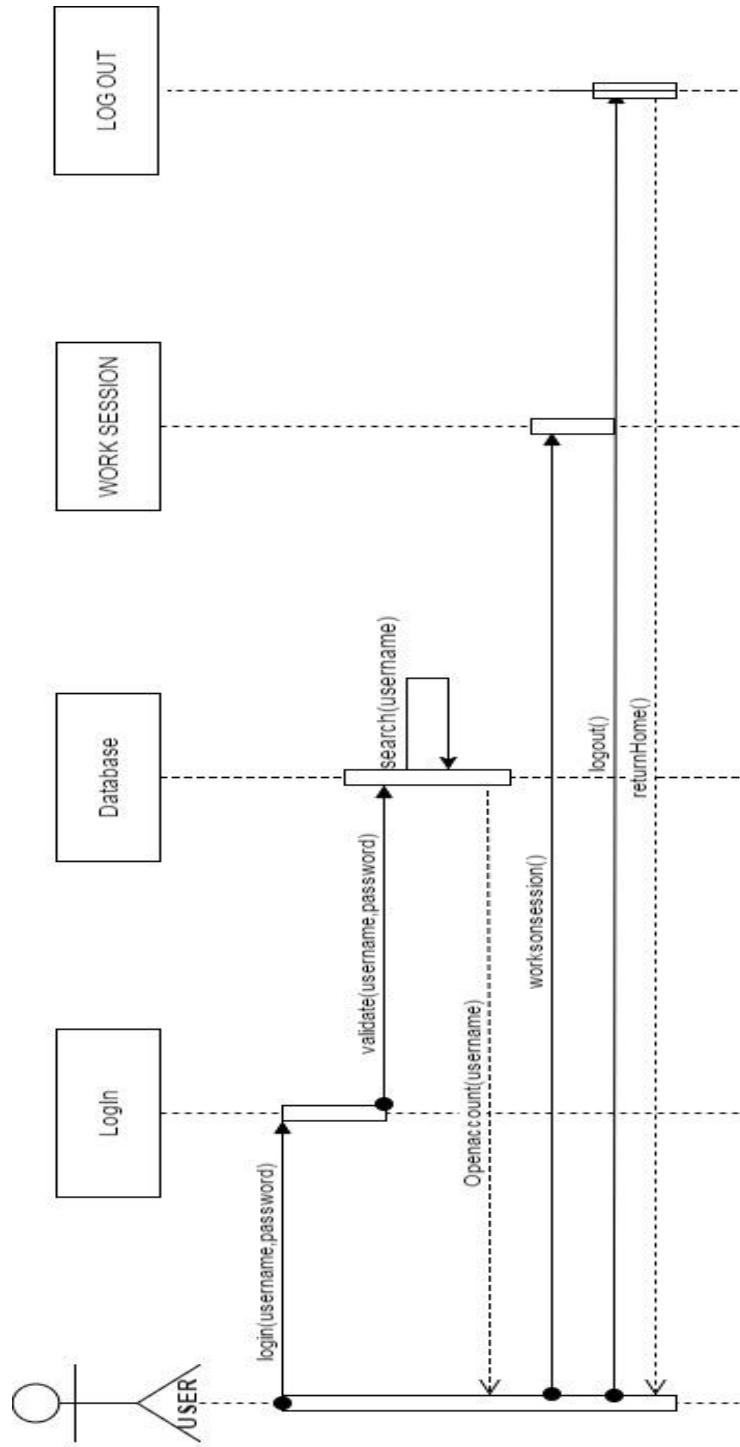


## **SEQUENCE DIAGRAMS**

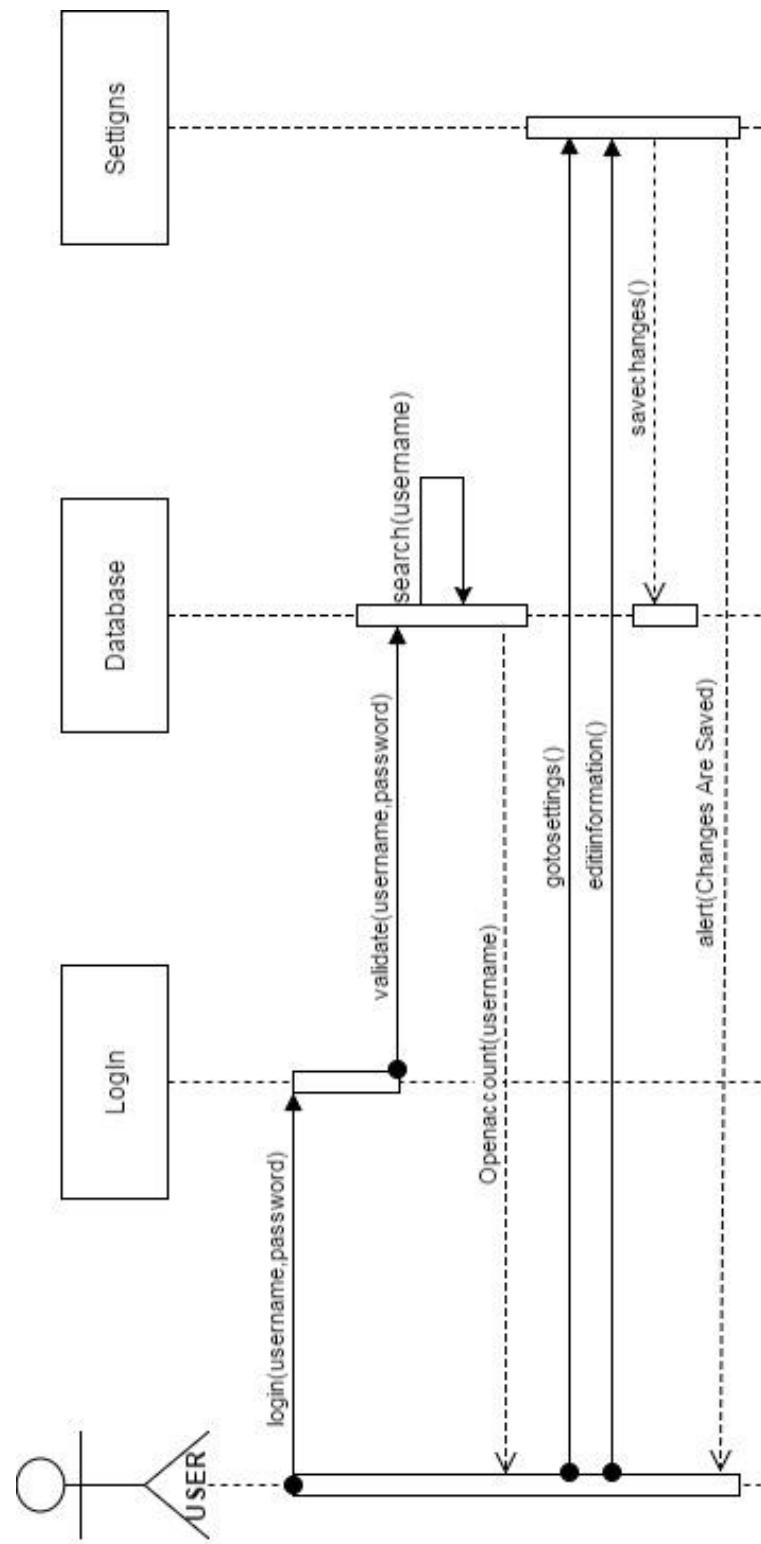
## (SD\_01) SEQUENCE DIAGRAM OF UC\_01



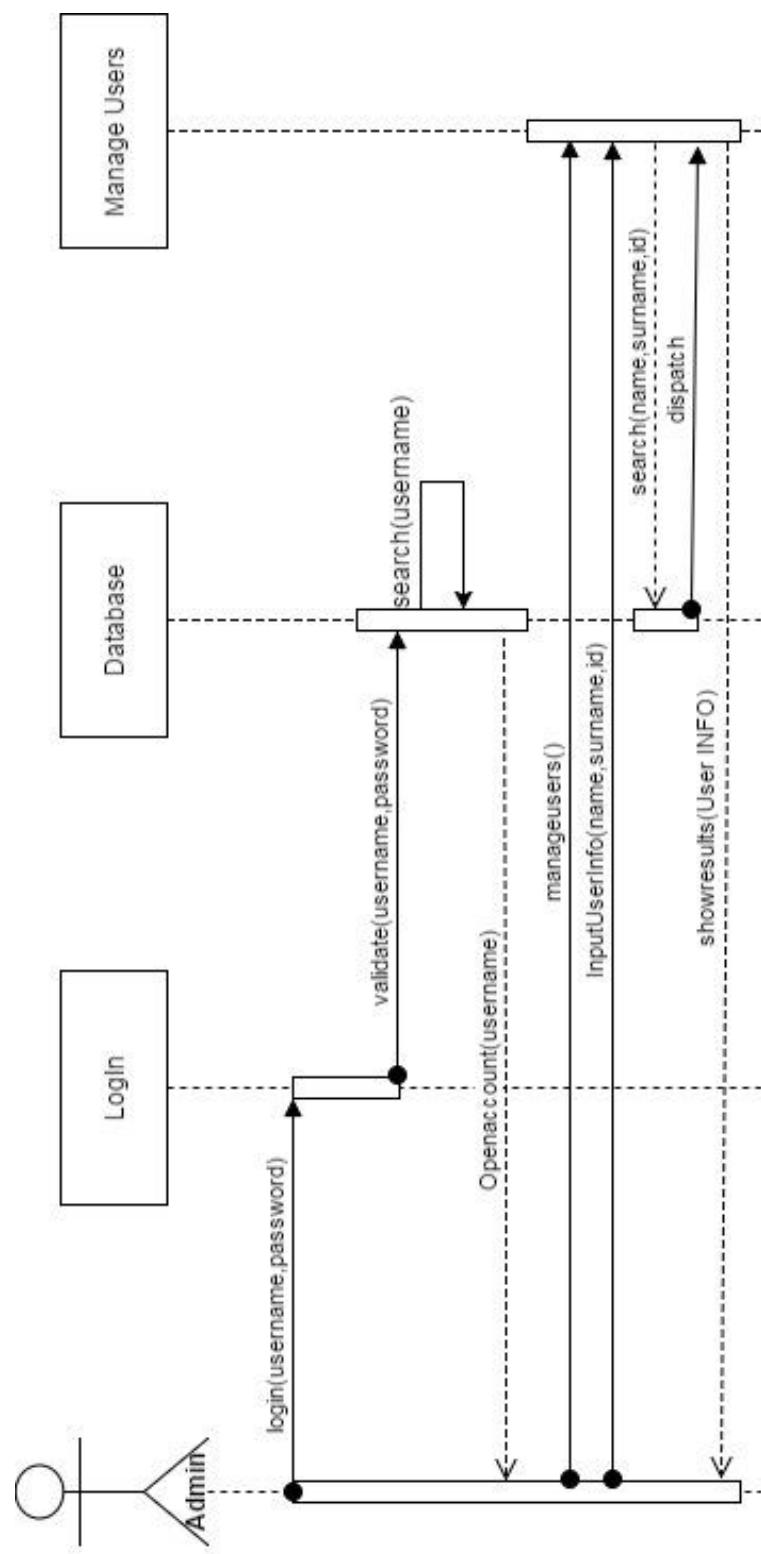
## (SD\_02) SEQUENCE DIAGRAM OF UC\_02



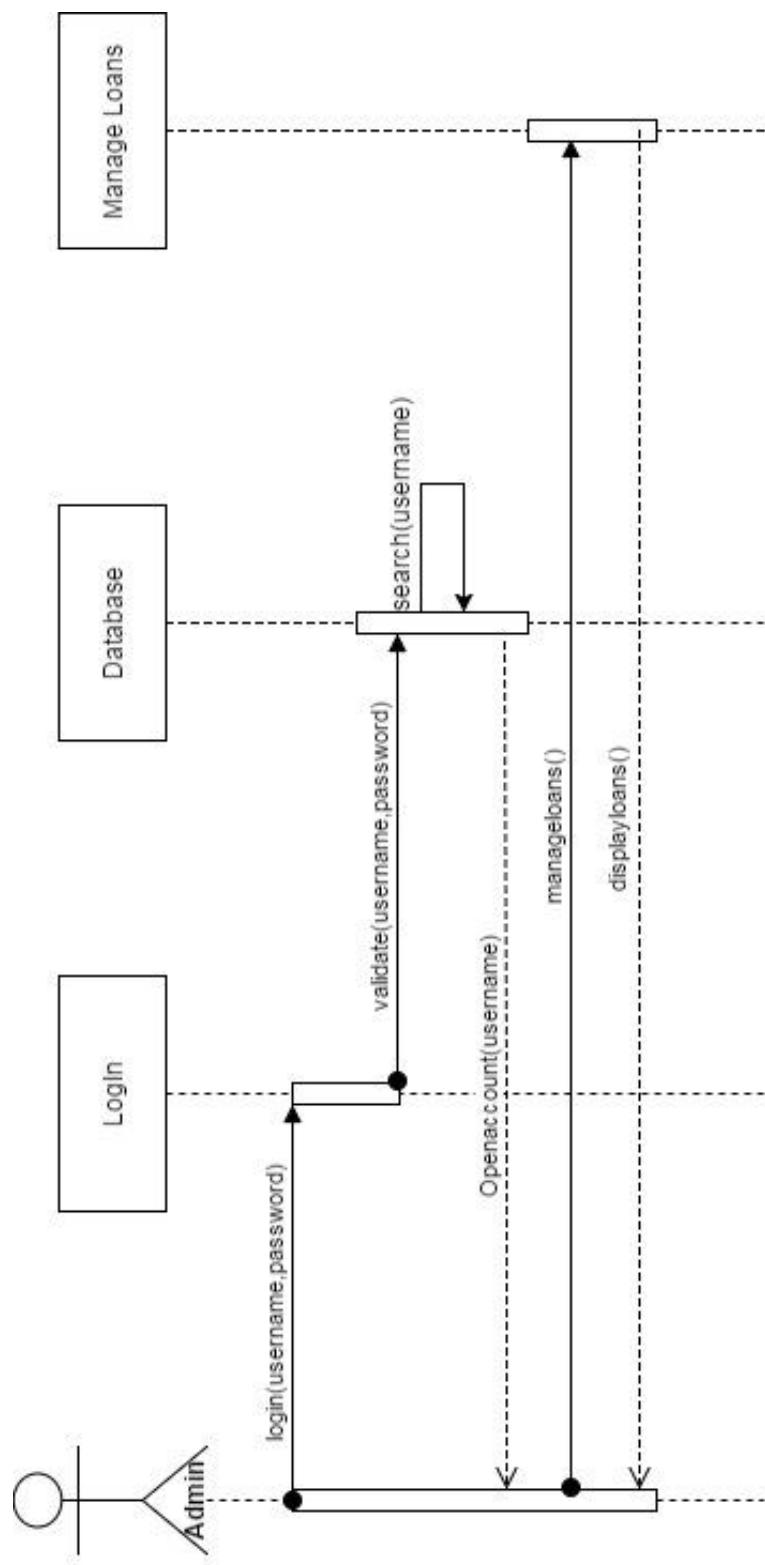
### (SD\_03) SEQUENCE DIAGRAM OF UC\_03



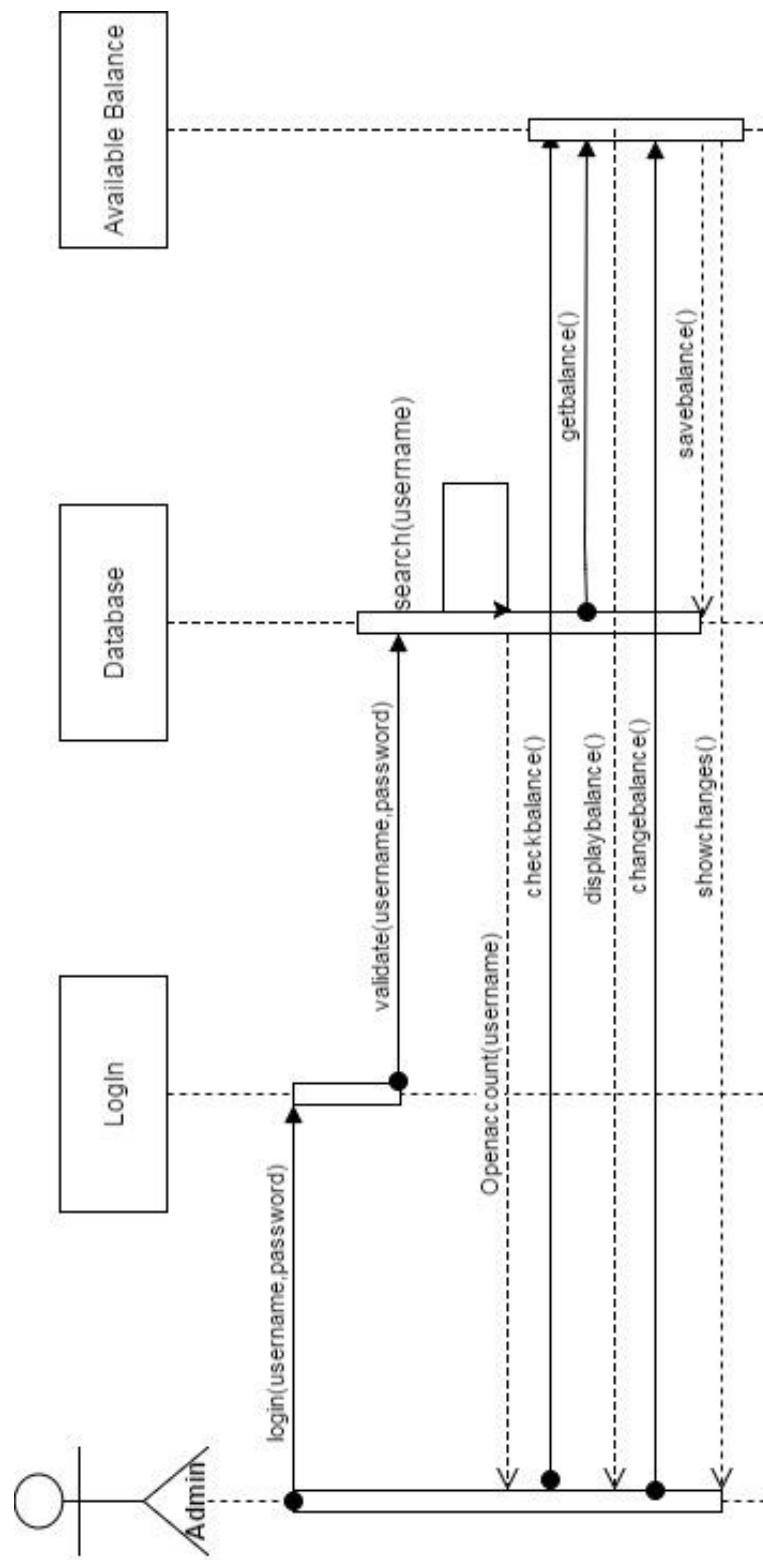
## (SD\_04) SEQUENCE DIAGRAM OF UC\_04



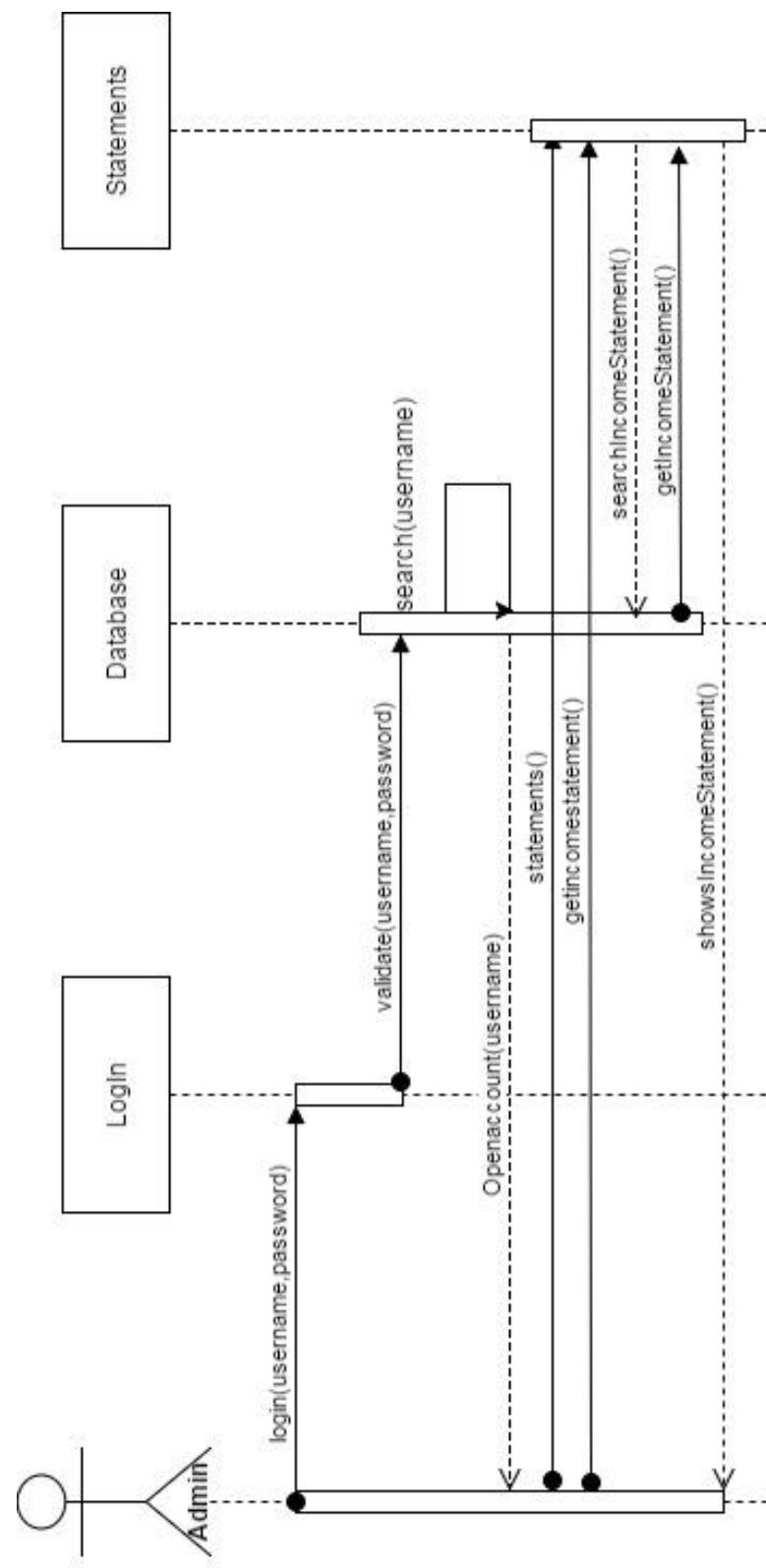
## (SD\_05) SEQUENCE DIAGRAM OF UC\_05



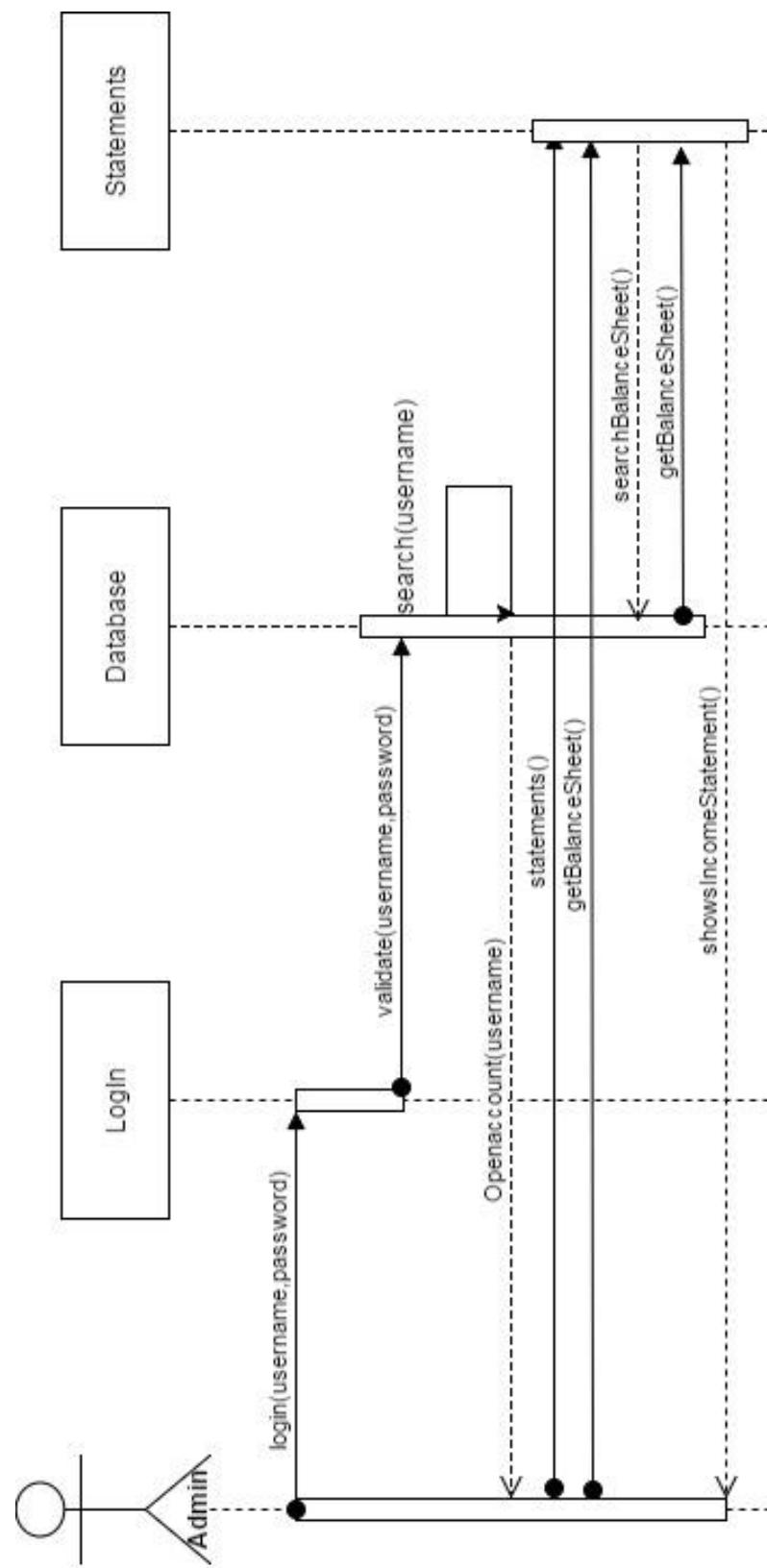
## (SD\_06) SEQUENCE DIAGRAM OF UC\_06



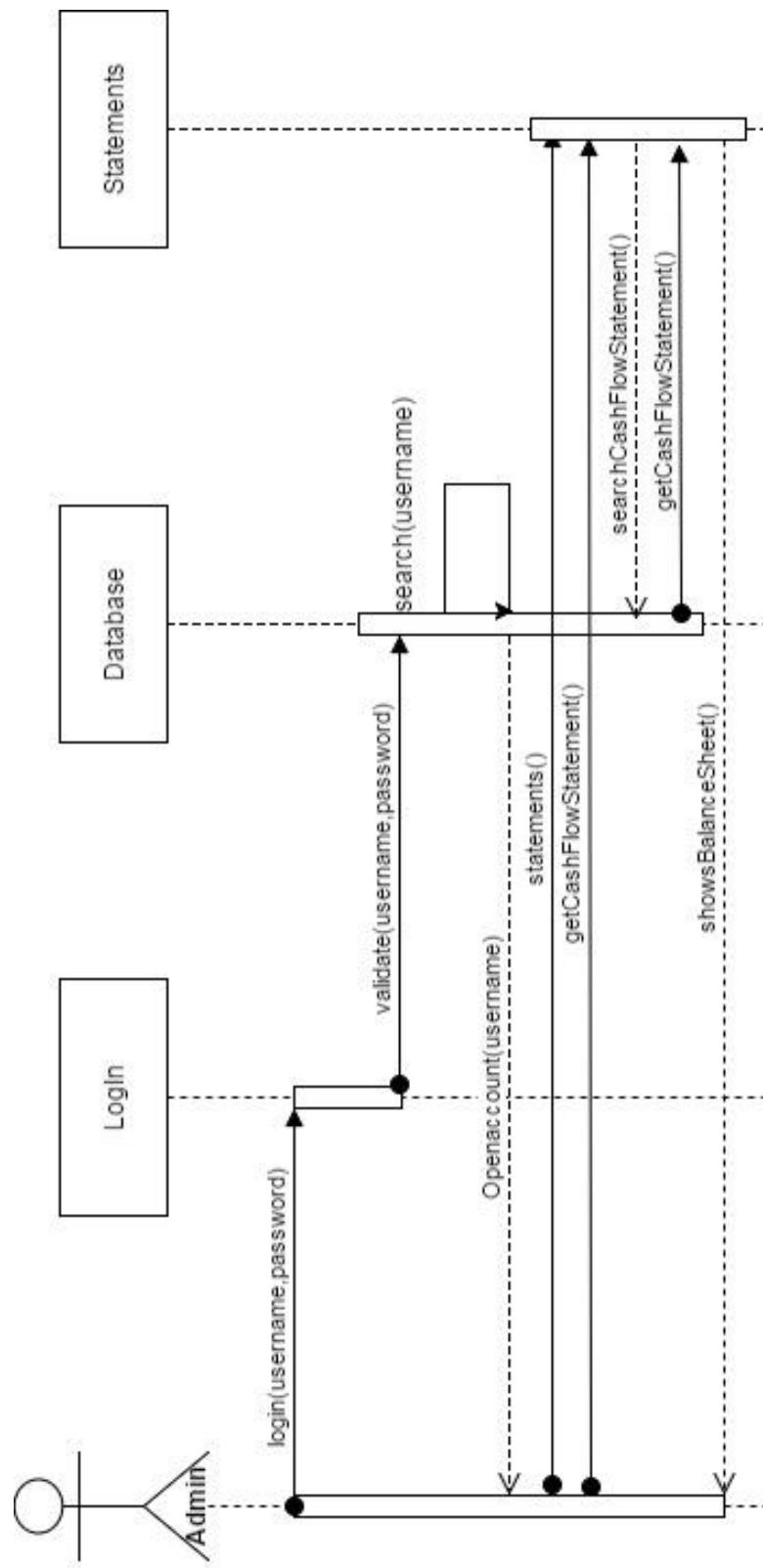
## (SD\_07) SEQUENCE DIAGRAM OF UC\_07



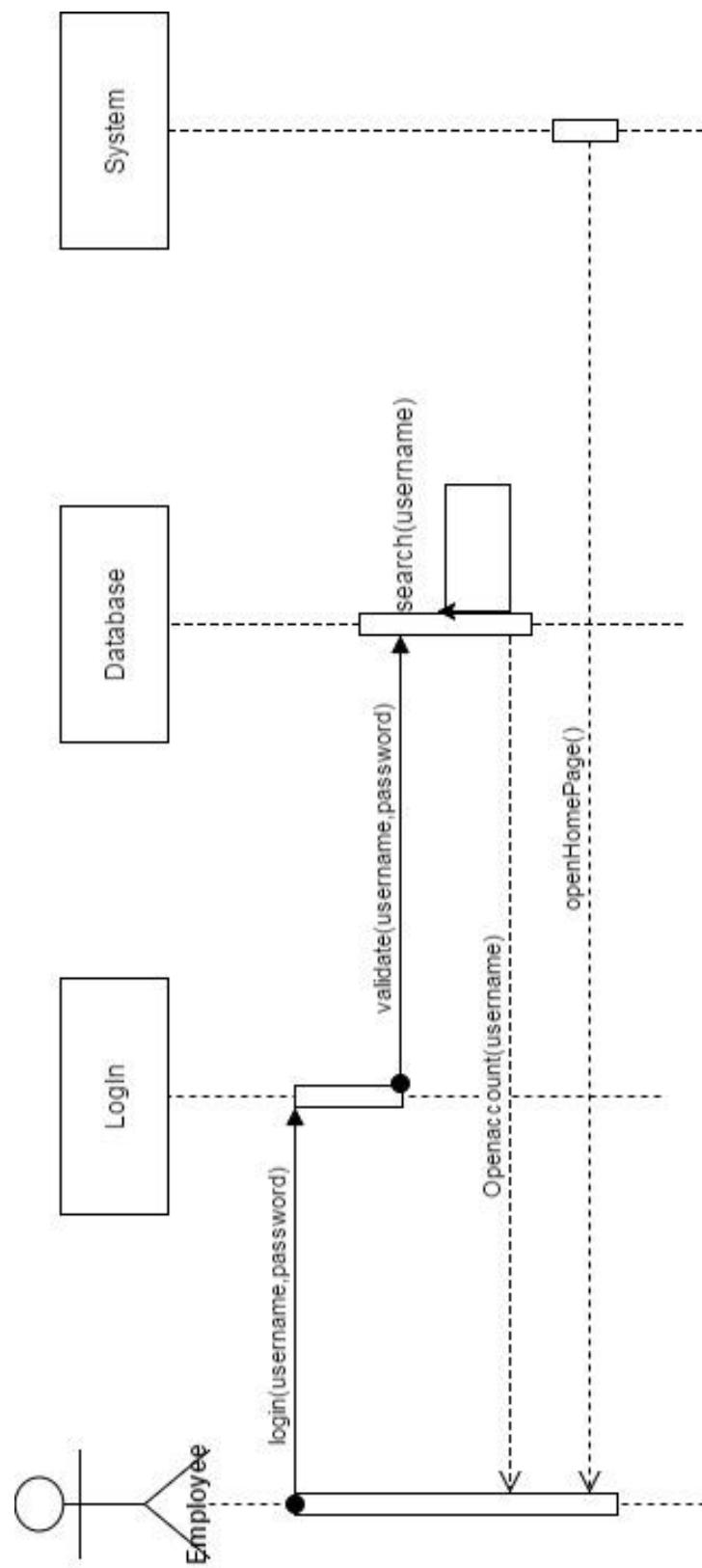
## (SD\_08) SEQUENCE DIAGRAM OF UC\_08



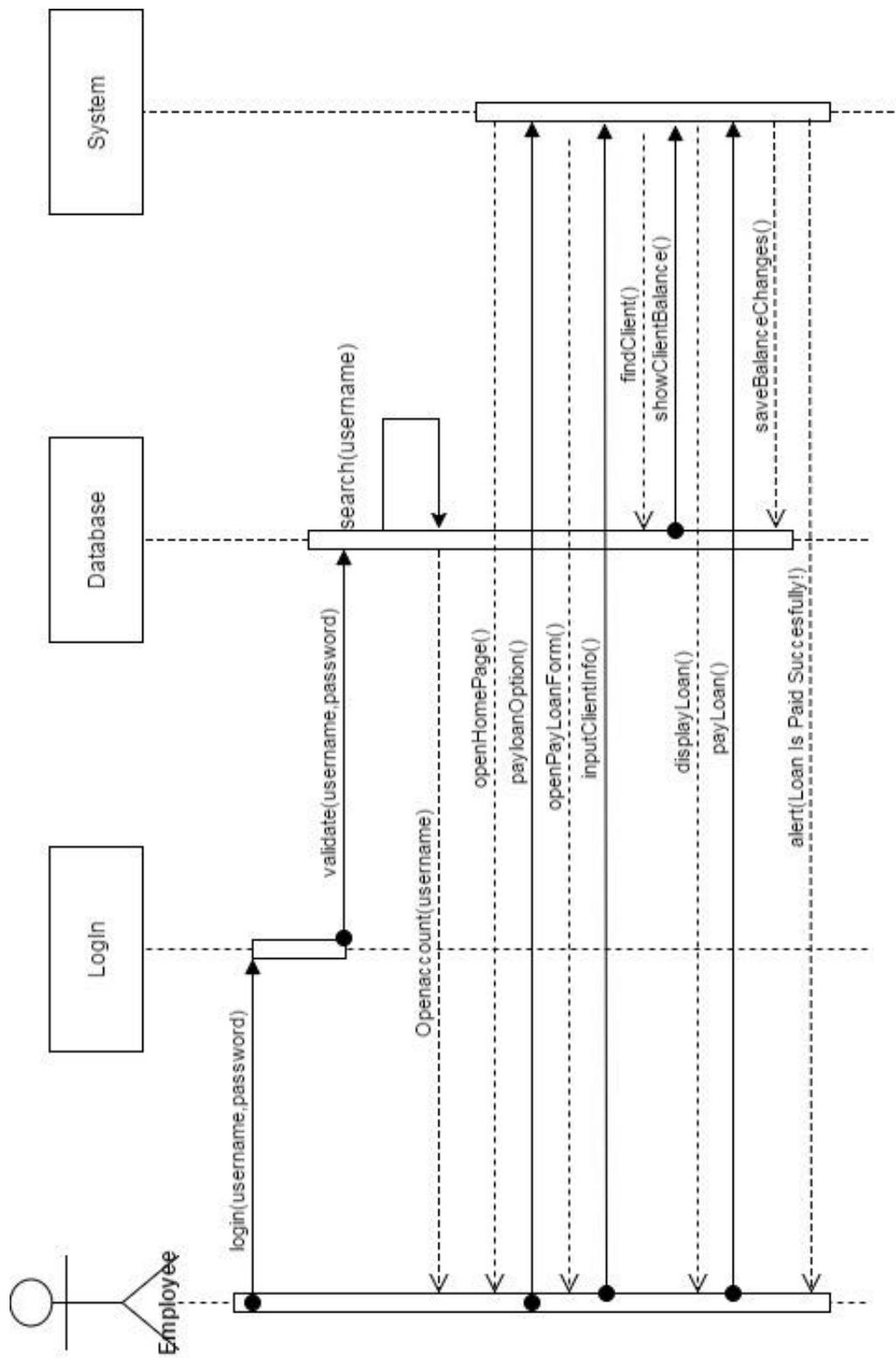
## (SD\_09) SEQUENCE DIAGRAM OF UC\_09



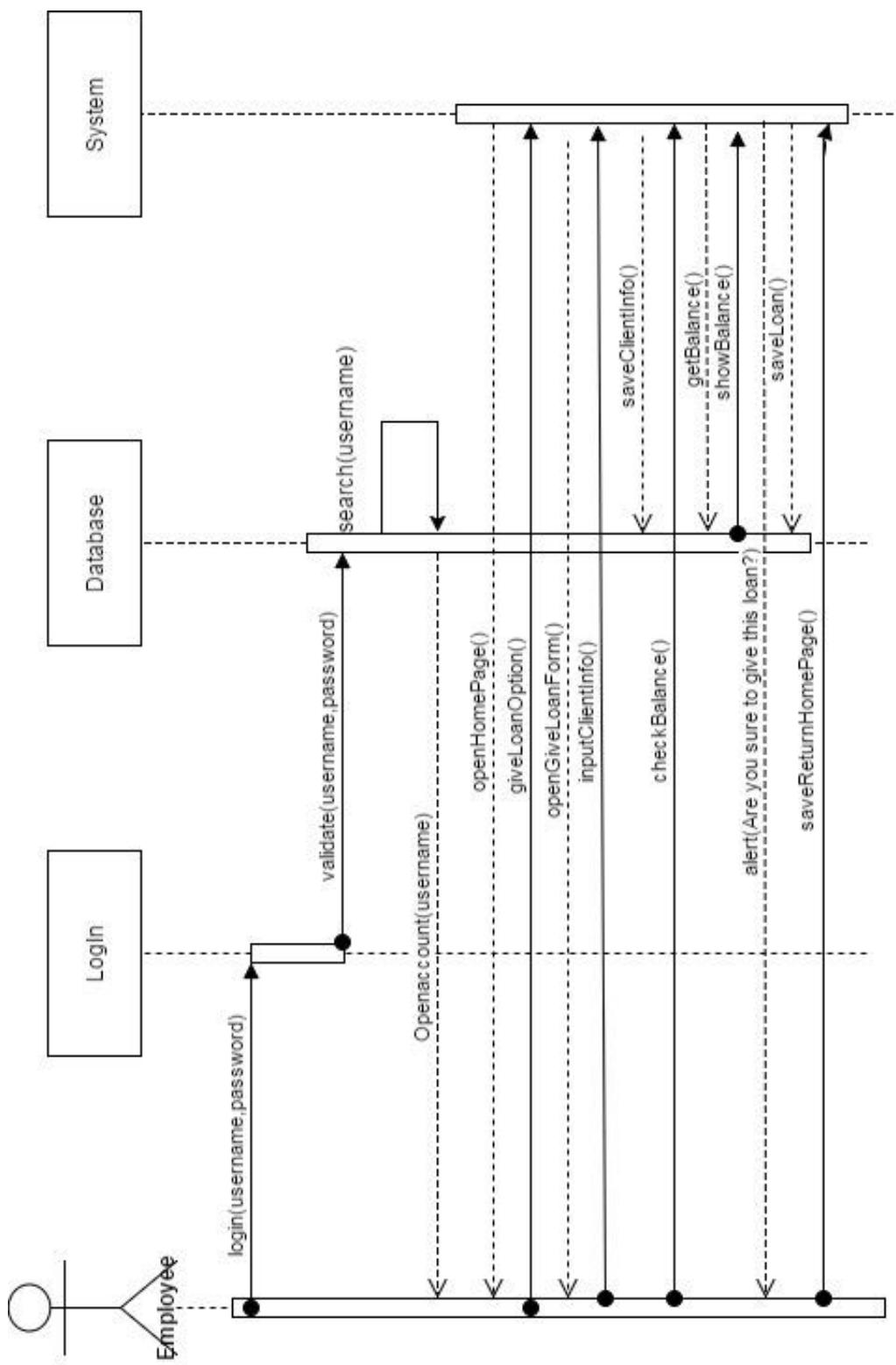
## (SD\_10) SEQUENCE DIAGRAM OF UC\_10



## (SD\_11) SEQUENCE DIAGRAM OF UC\_11

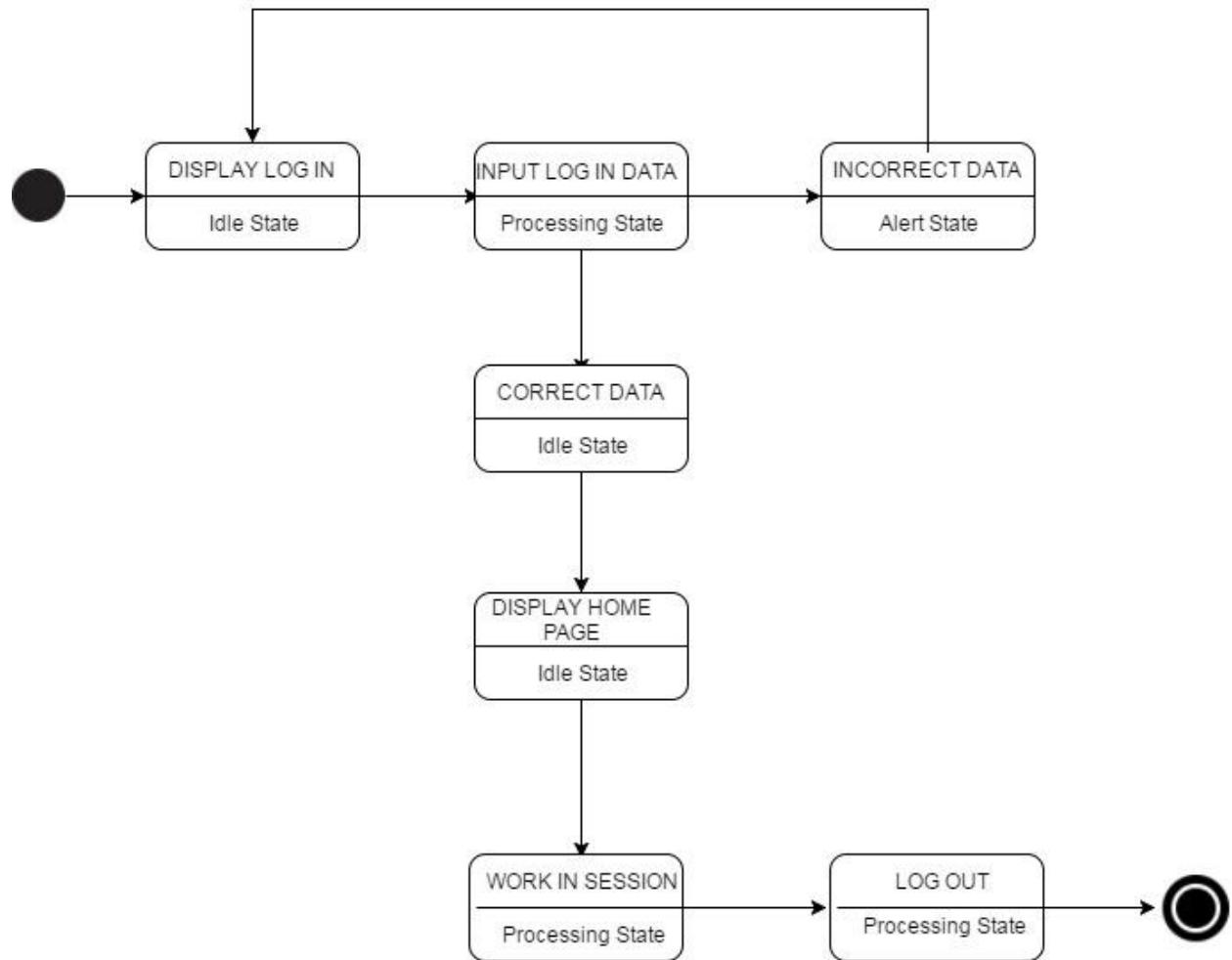


## (SD\_12) SEQUENCE DIAGRAM OF UC\_12

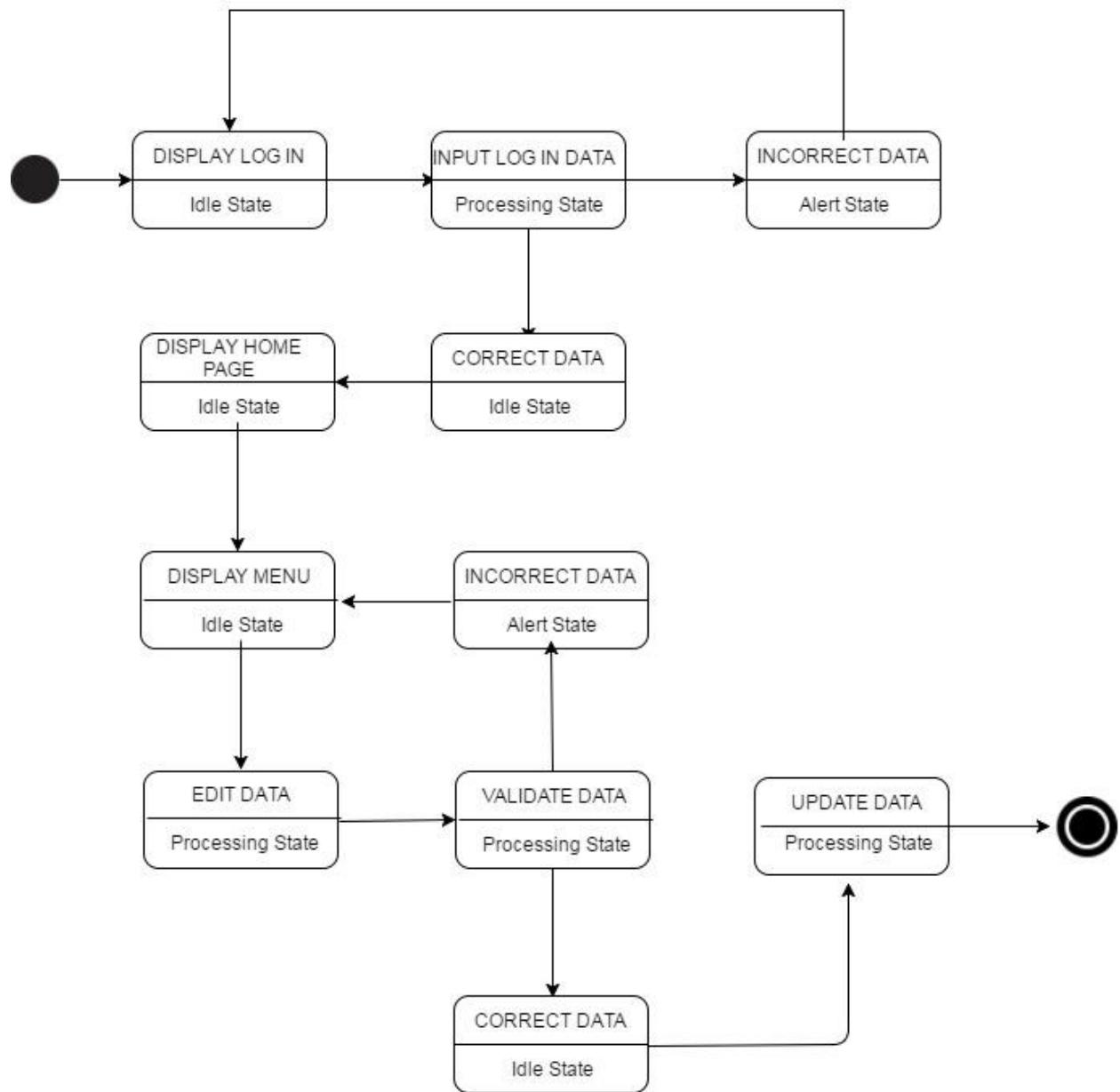


## **STATE DIAGRAM**

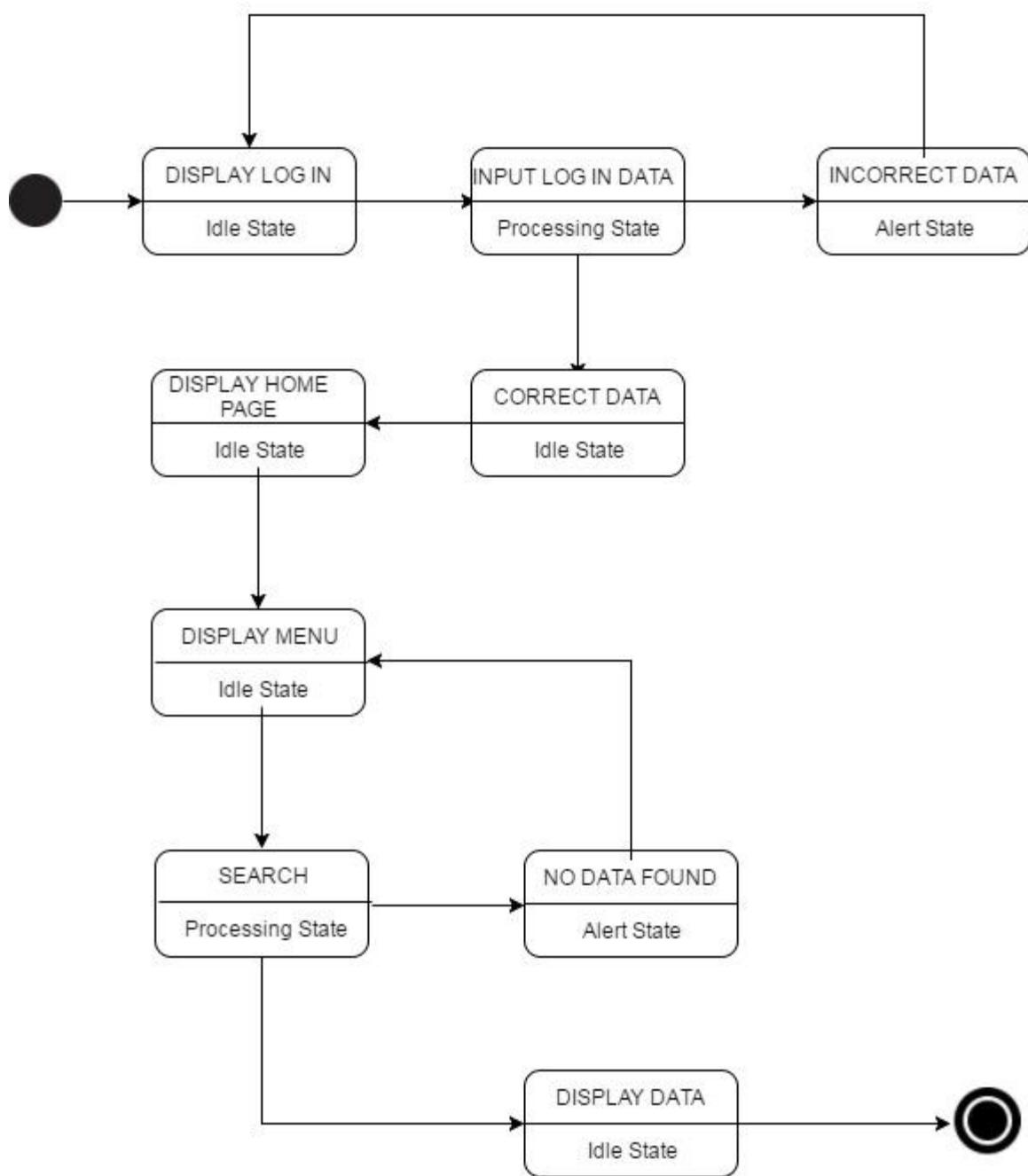
## STATE DIAGRAM 1 (STD\_01)



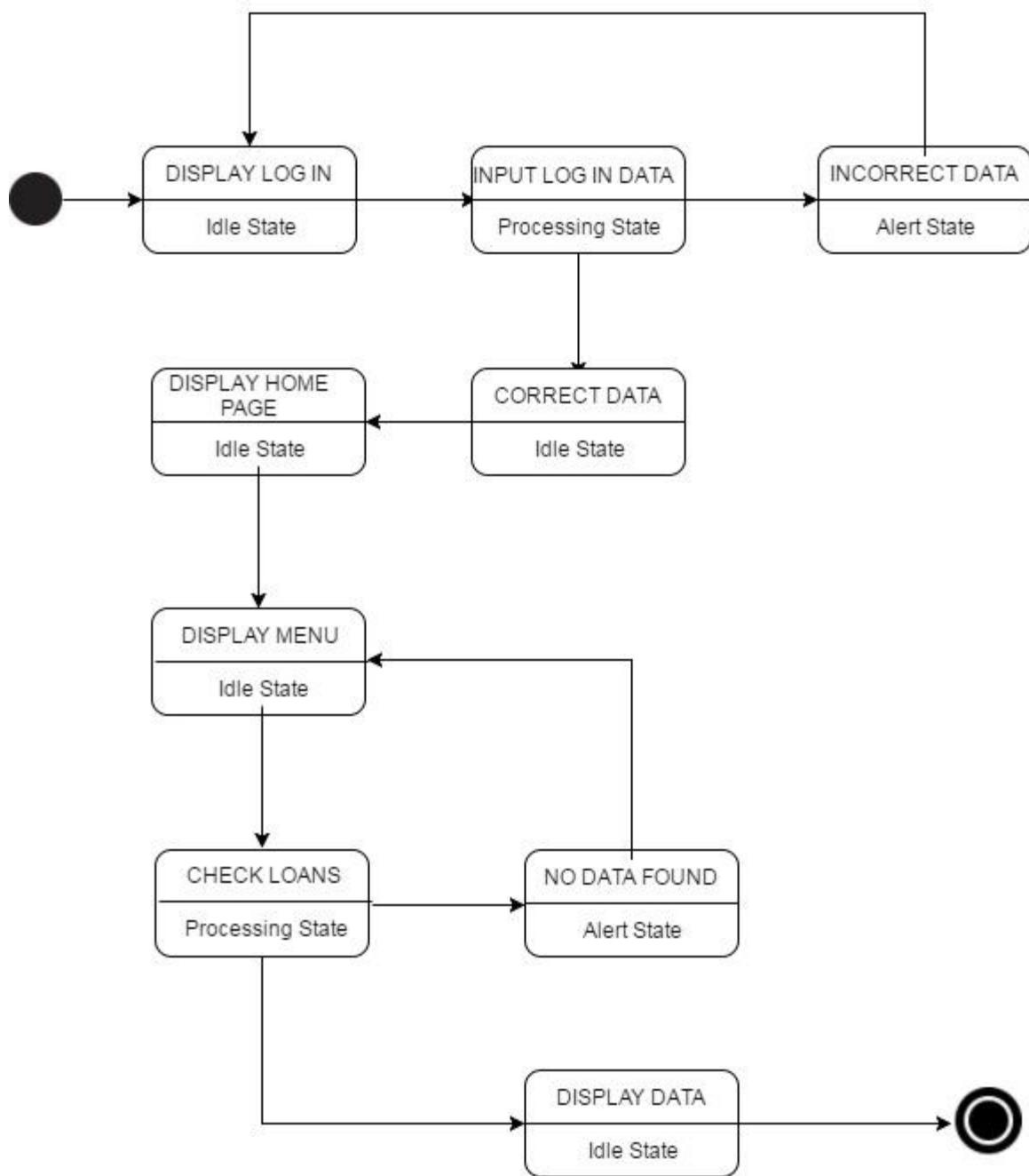
## STATE DIAGRAM 2 (STD\_02)



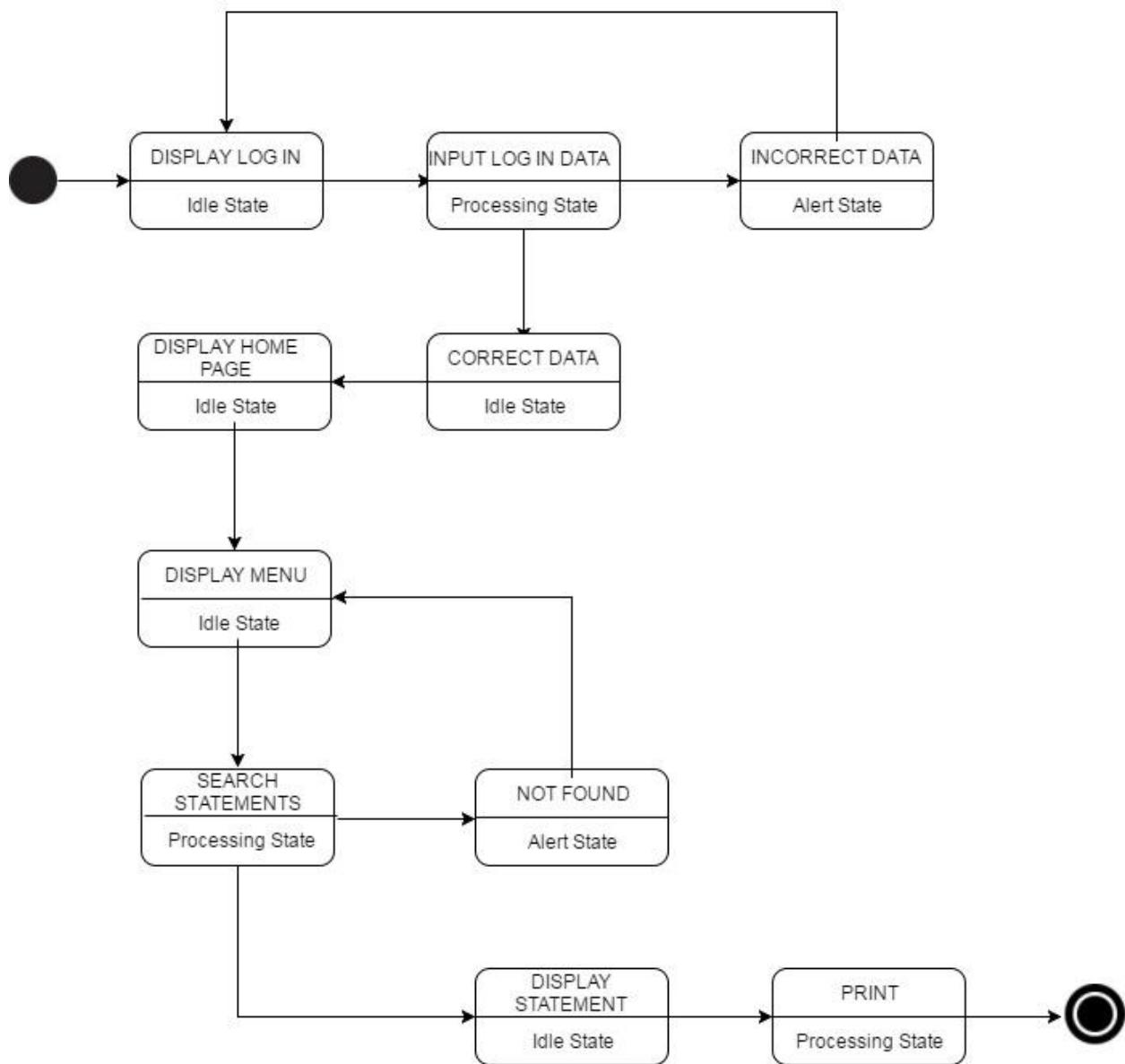
### STATE DIAGRAM 3 (STD\_03)



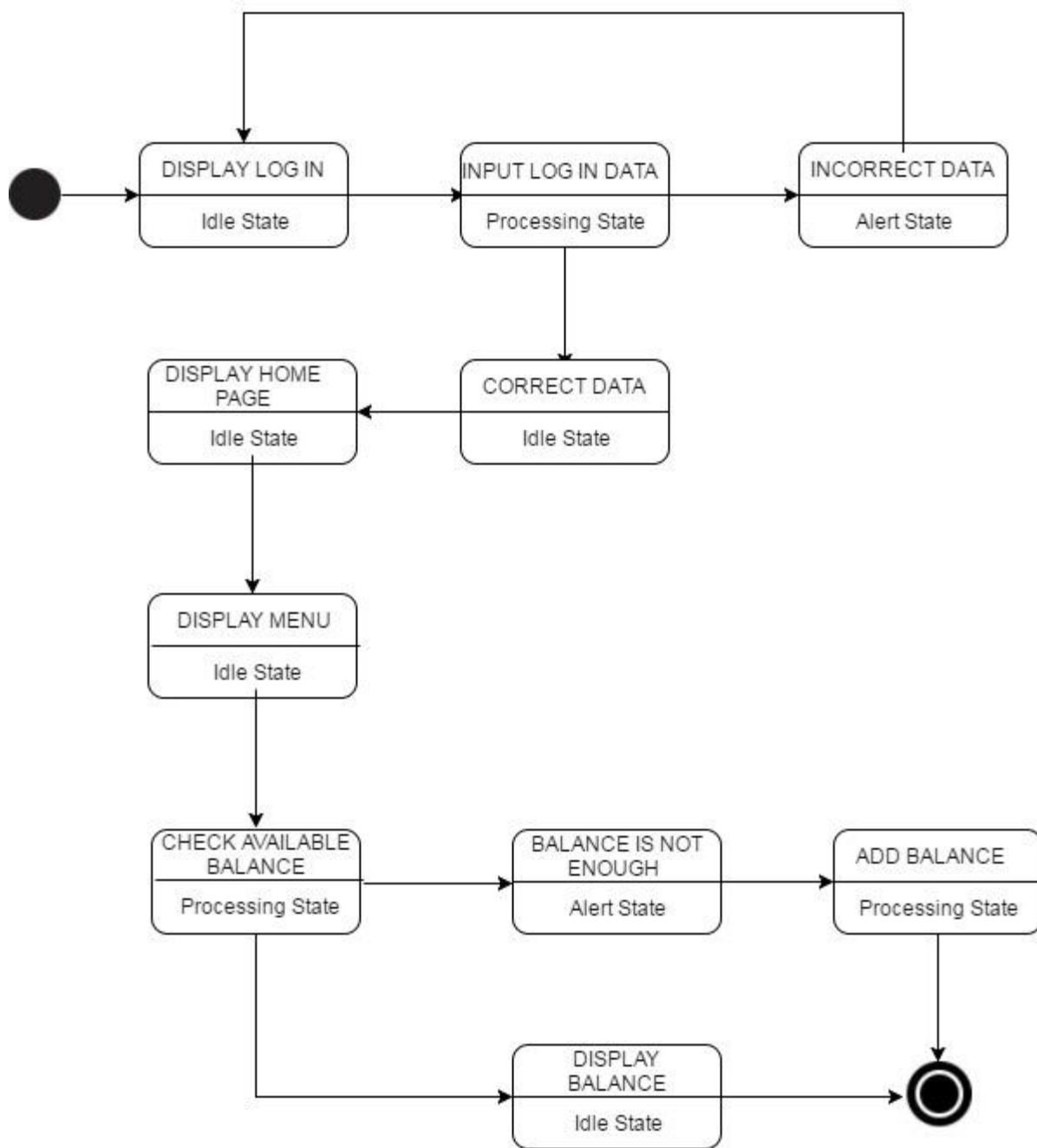
## STATE DIAGRAM 4 (STD\_04)



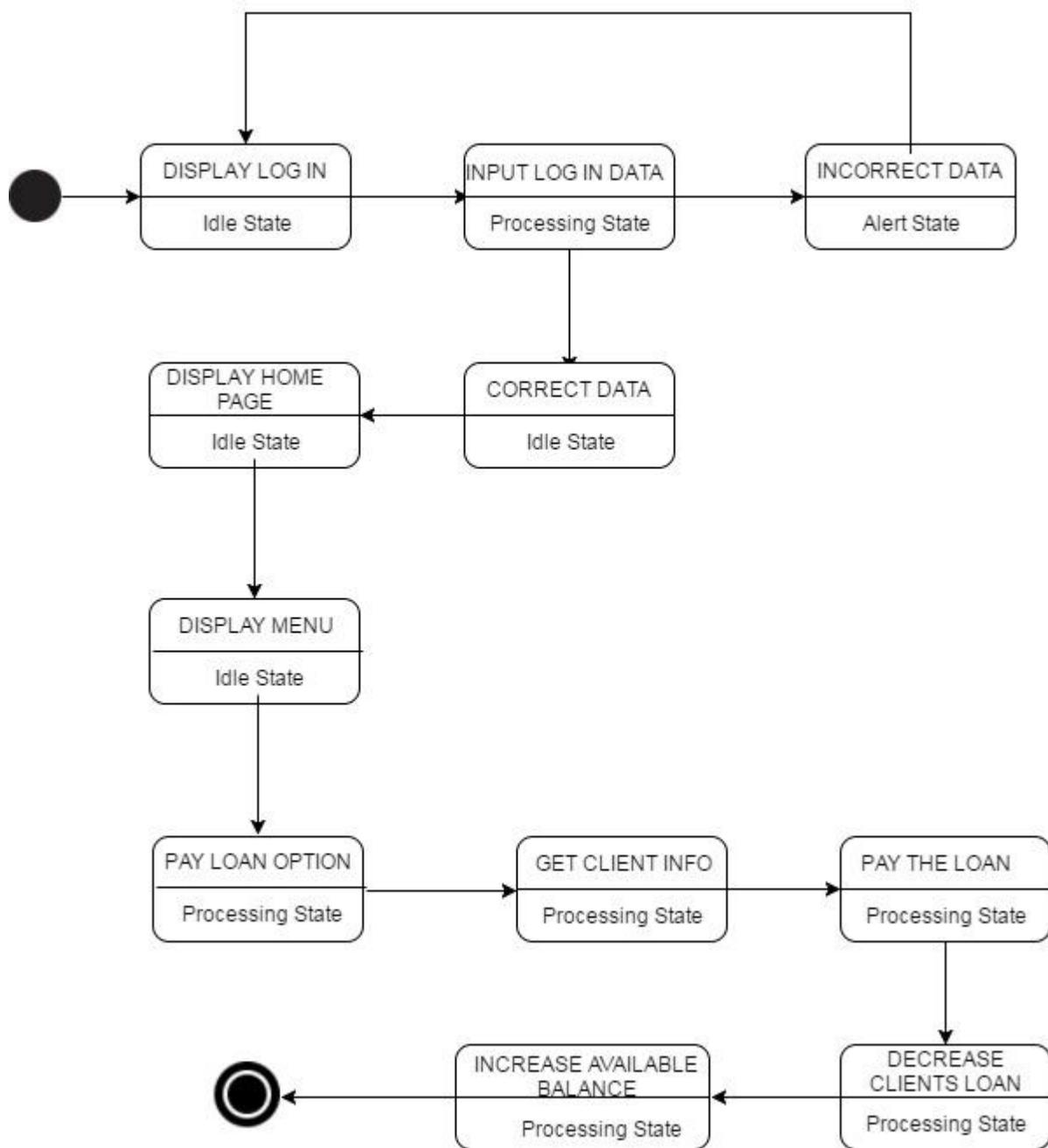
## STATE DIAGRAM 5 (STD\_05)



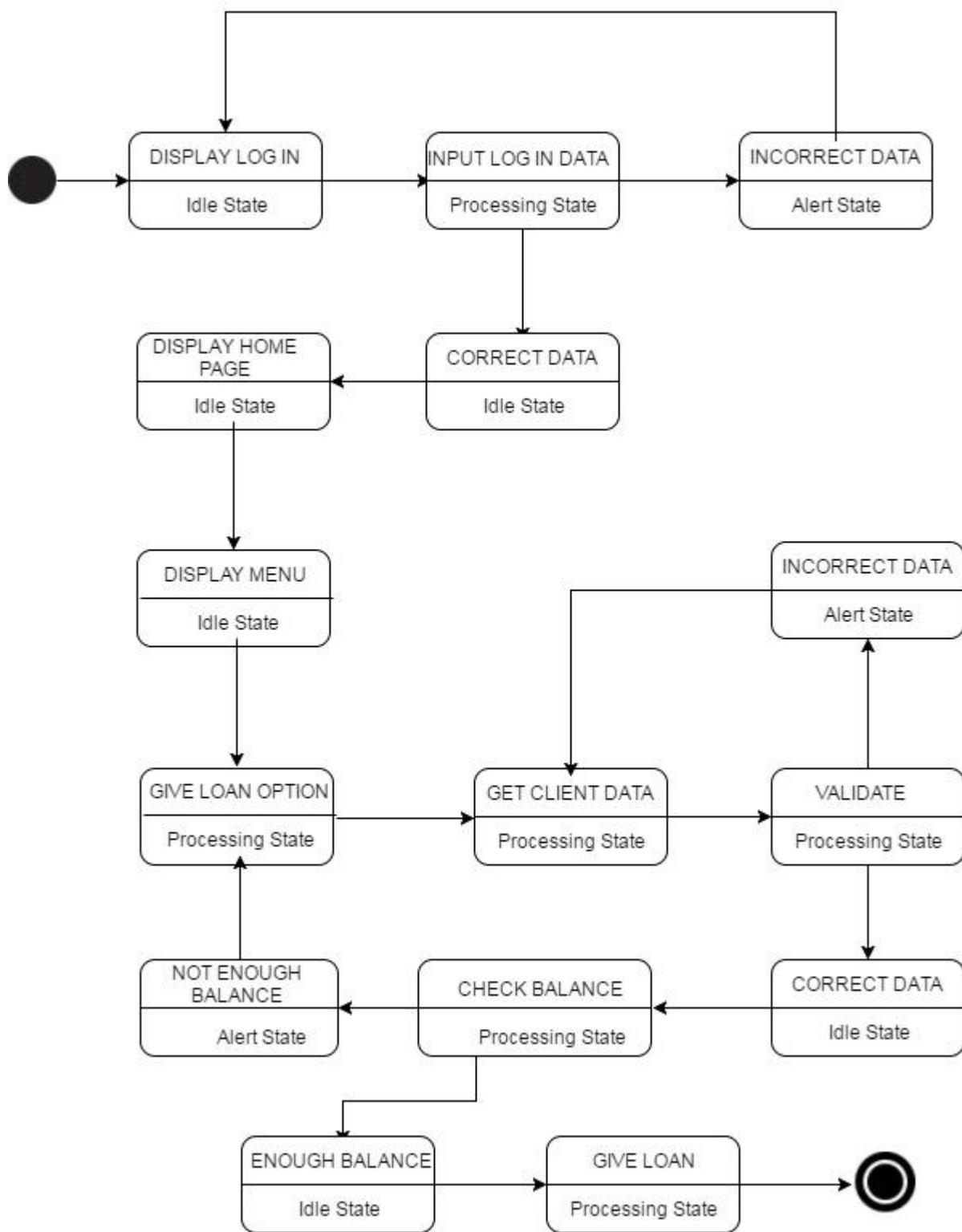
## STATE DIAGRAM 6 (STD\_06)



## STATE DIAGRAM 7 (STD\_07)

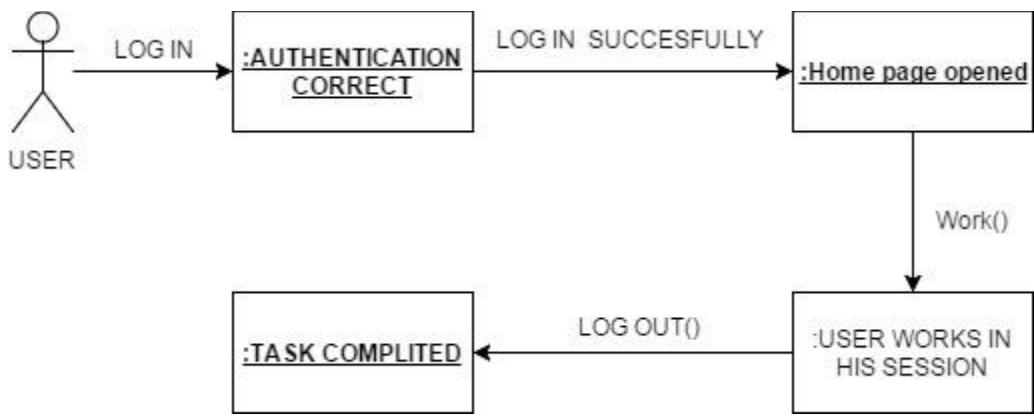


## STATE DIAGRAM 8 (STD\_08)

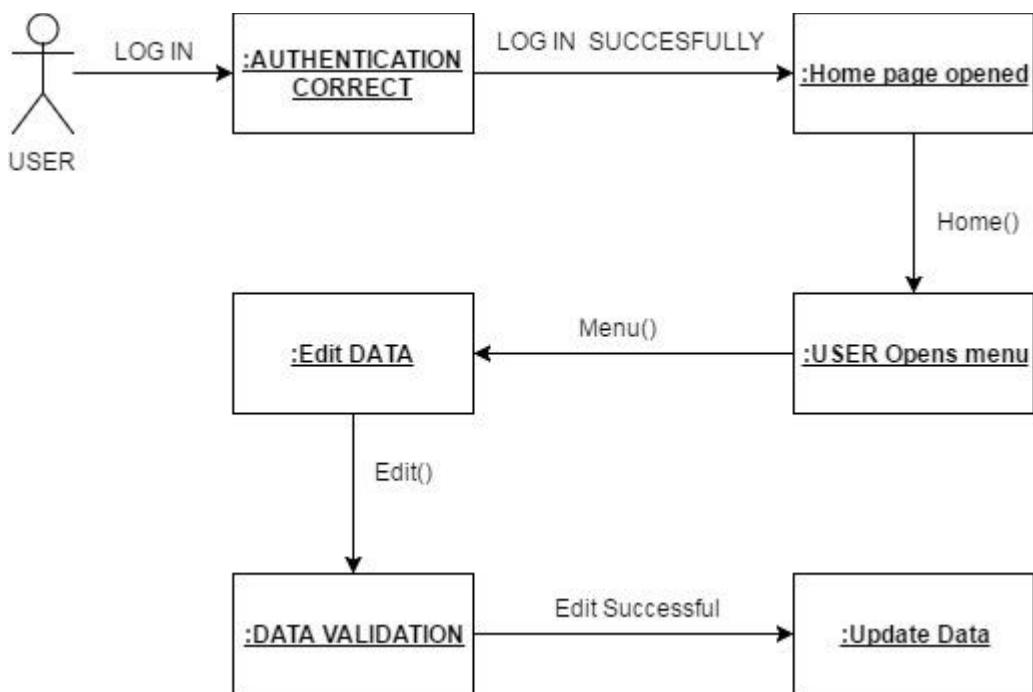


## **COLLABORATION DIAGRAM**

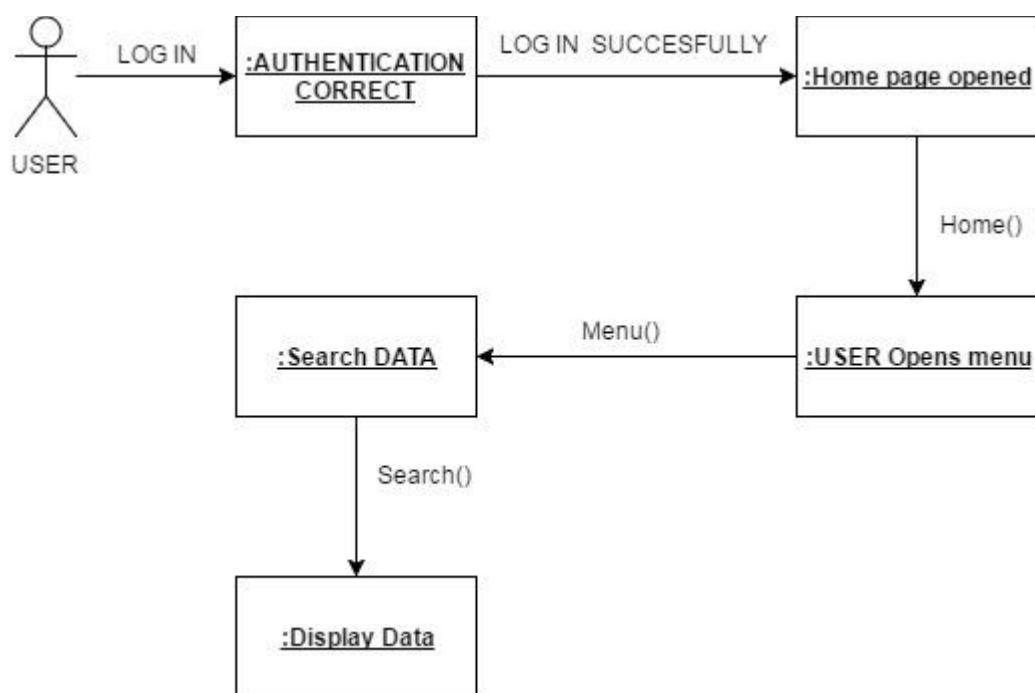
## COLLABORATION DIAGRAM 1 (CD\_01)



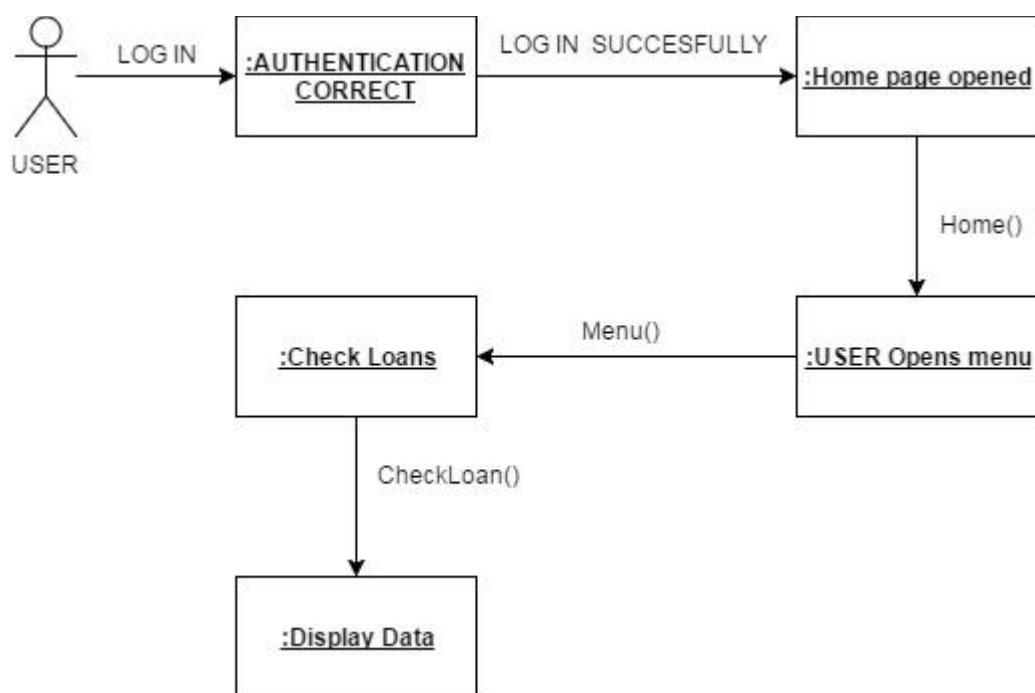
## COLLABORATION DIAGRAM 2 (CD\_02)



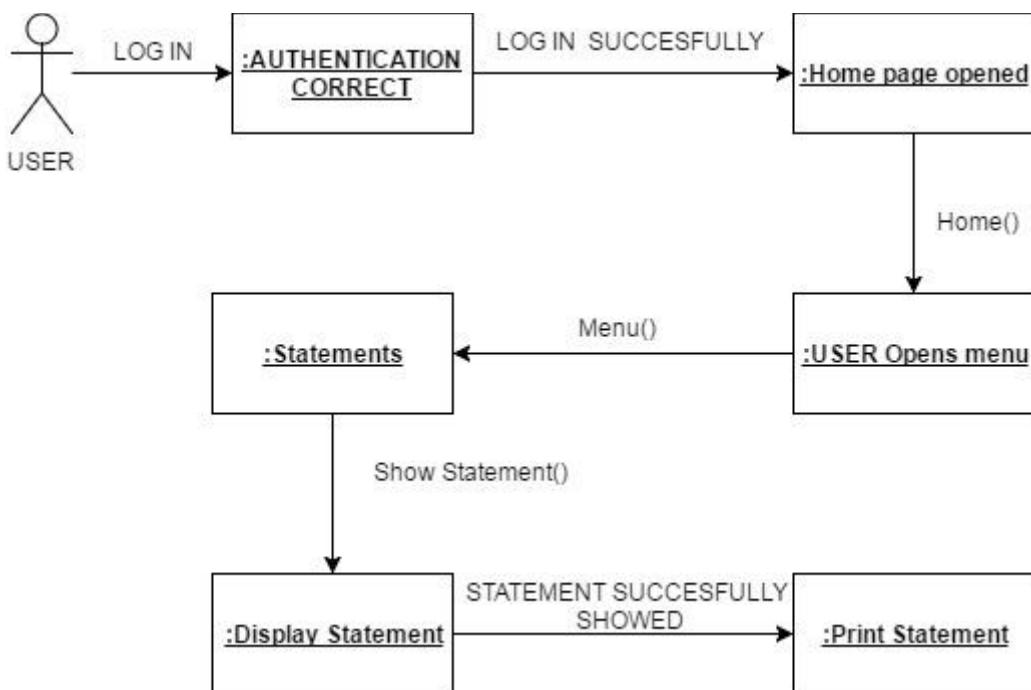
### COLLABORATION DIAGRAM 3 (CD\_03)



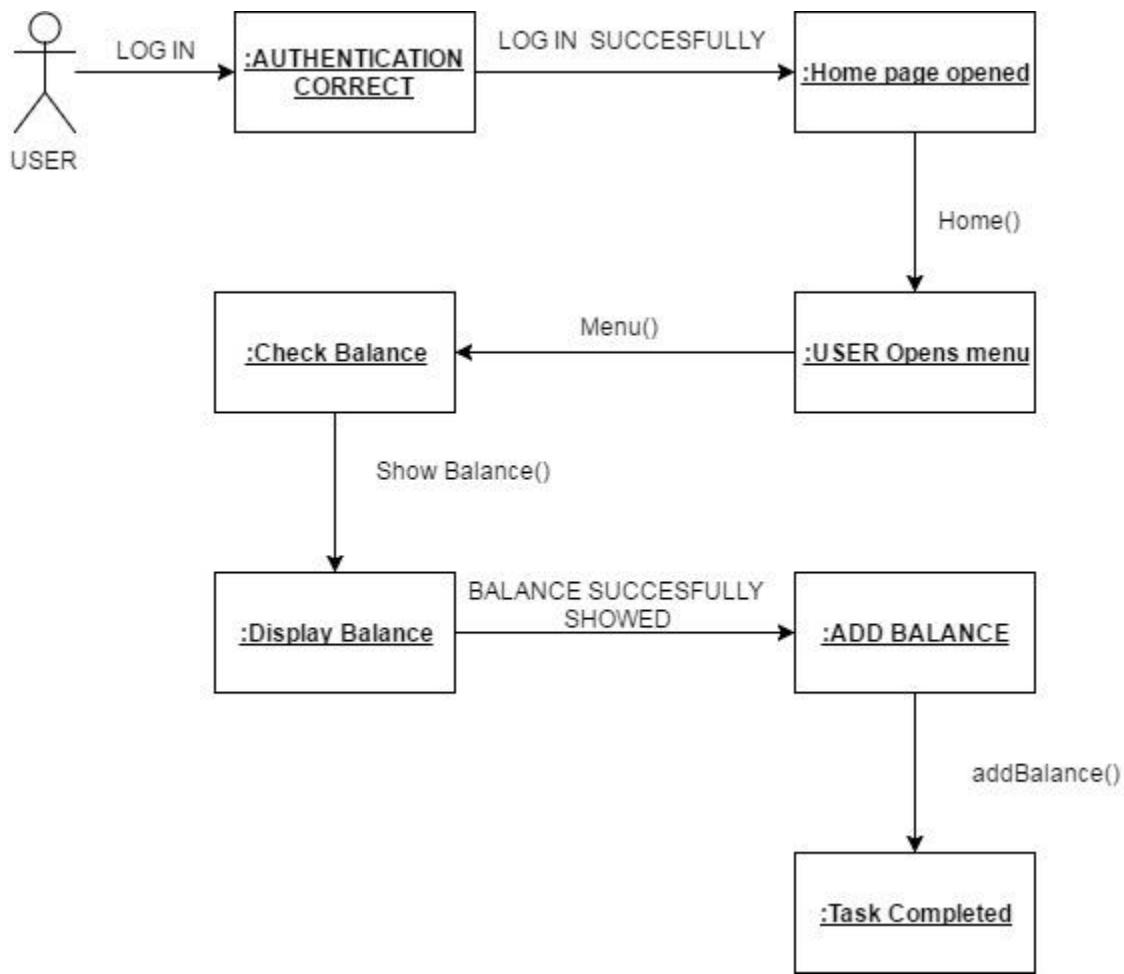
## COLLABORATION DIAGRAM 4 (CD\_04)



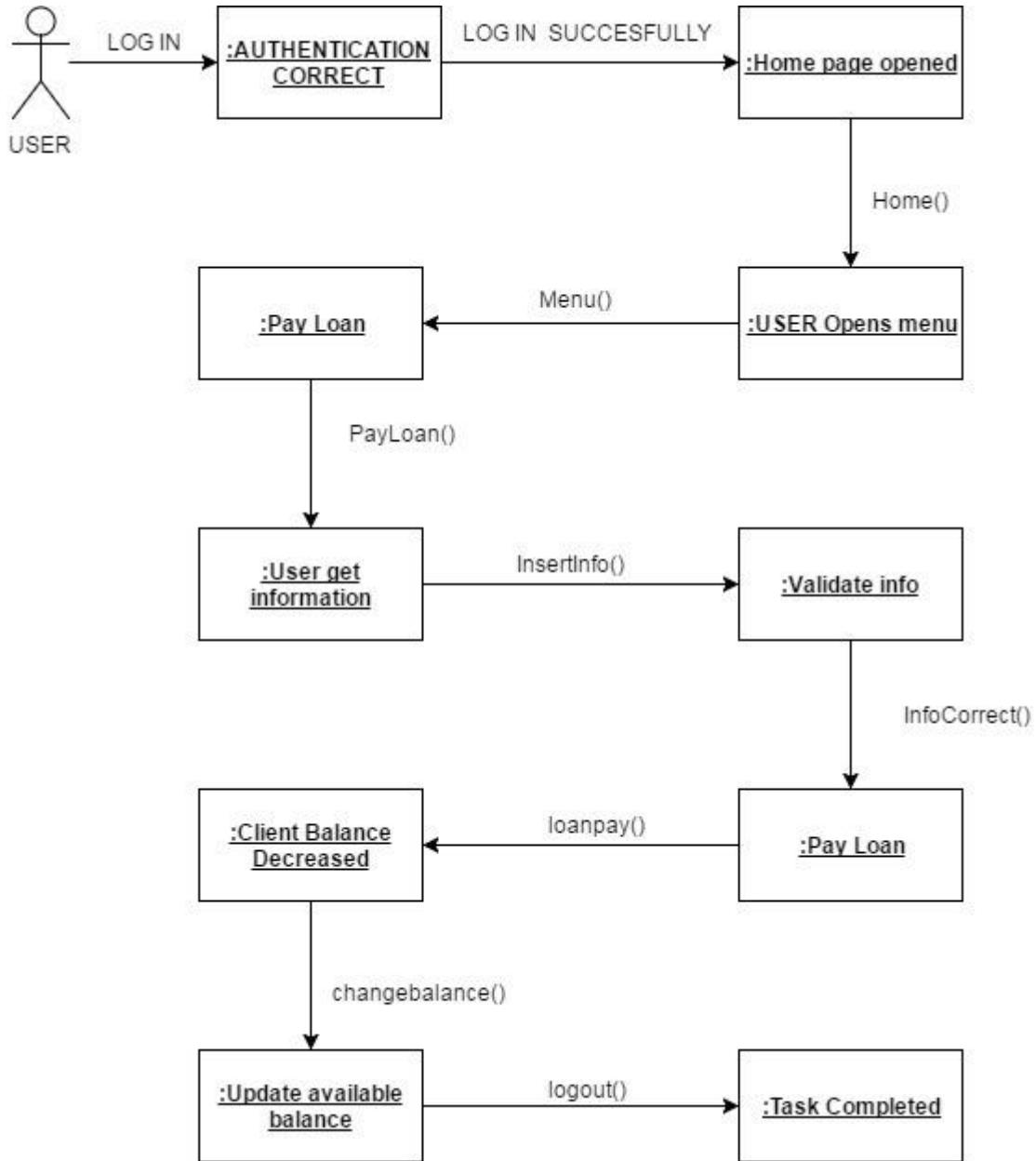
## COLLABORATION DIAGRAM 5 (CD\_05)



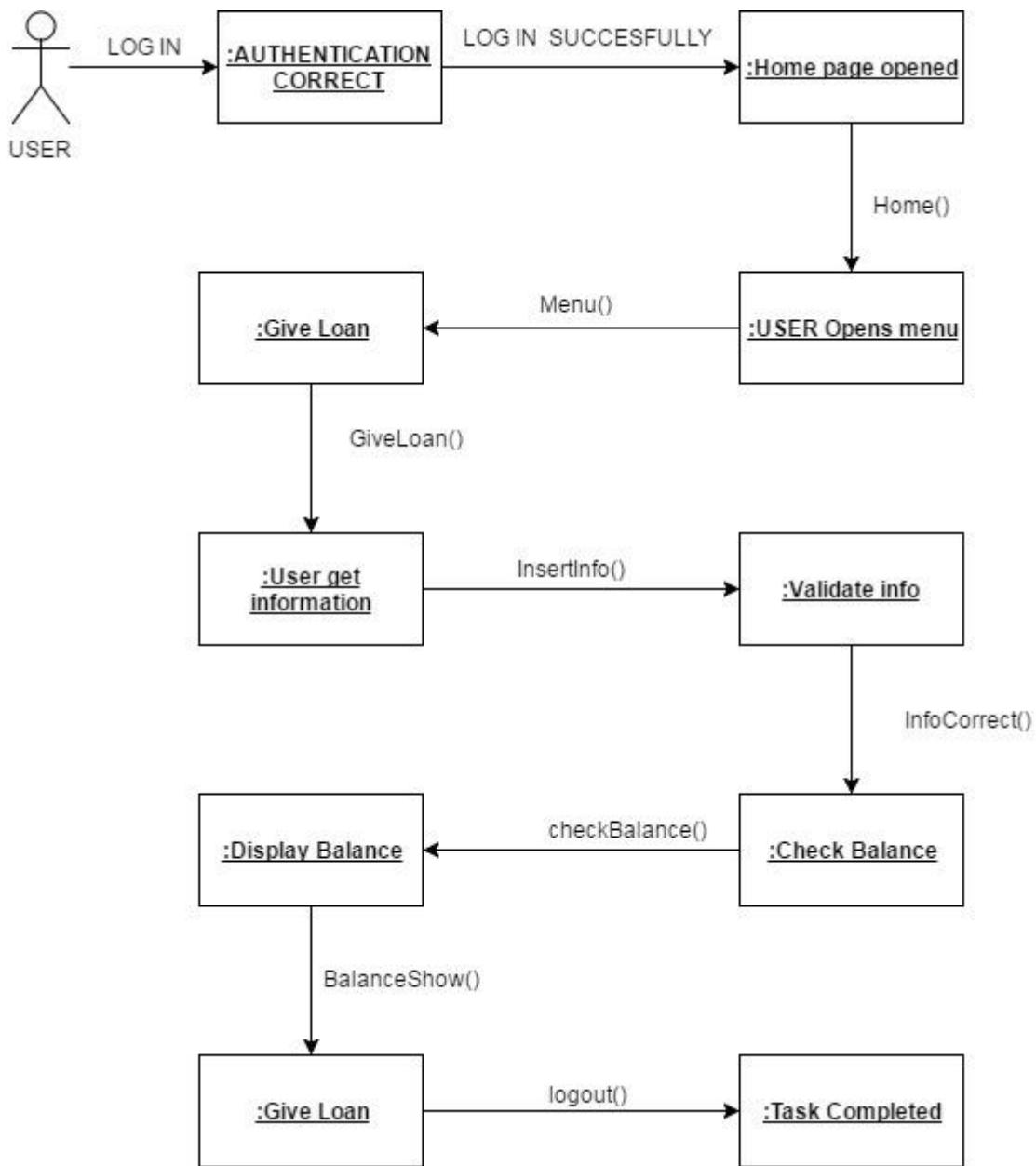
## COLLABORATION DIAGRAM 6 (CD\_06)



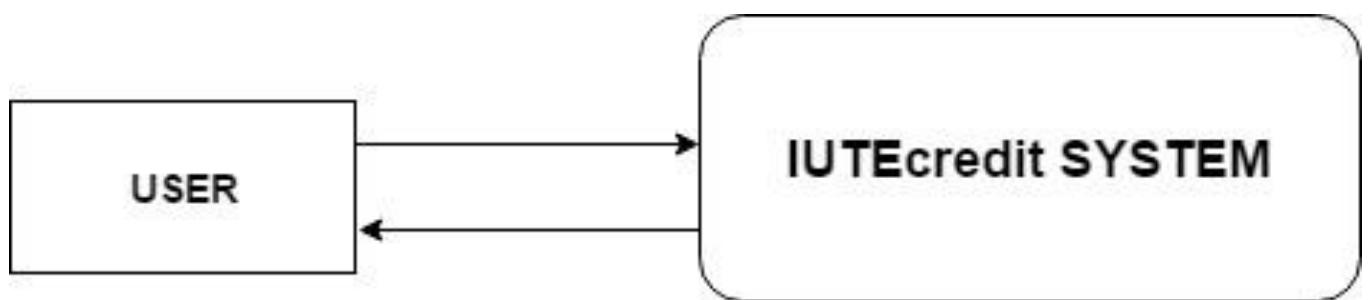
## COLLABORATION DIAGRAM 7 (CD\_07)



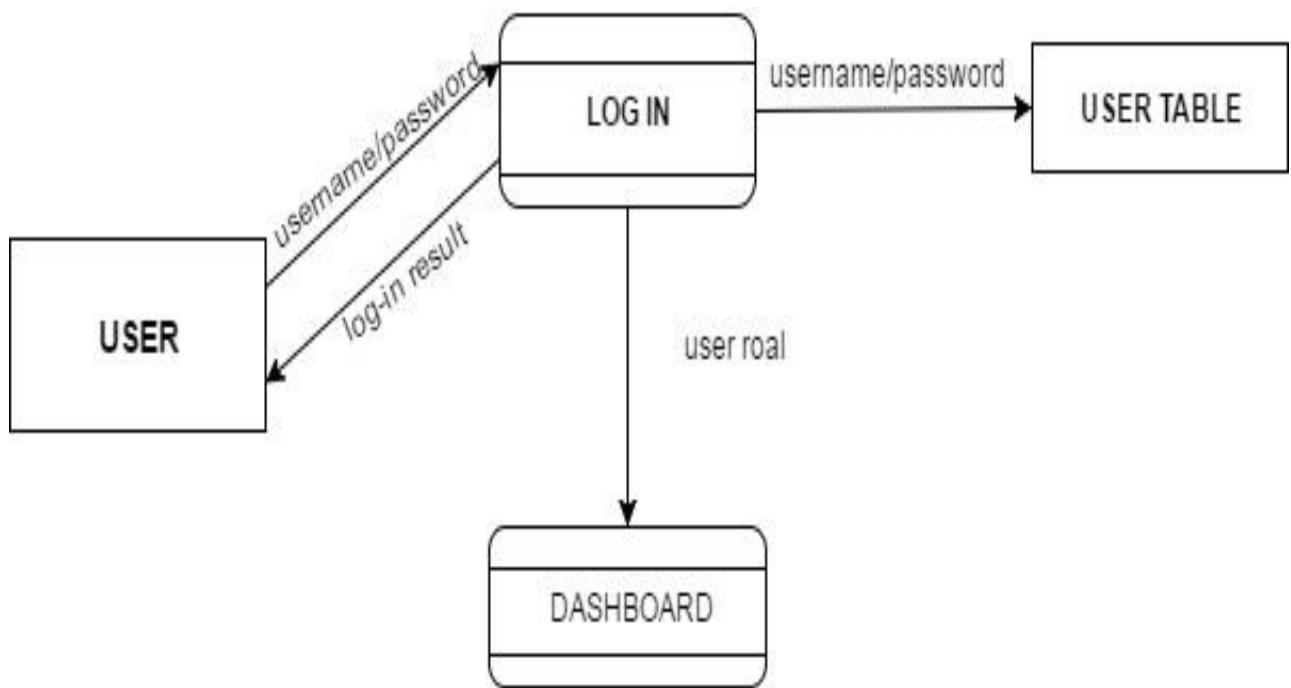
## COLLABORATION DIAGRAM 8 (CD\_08)



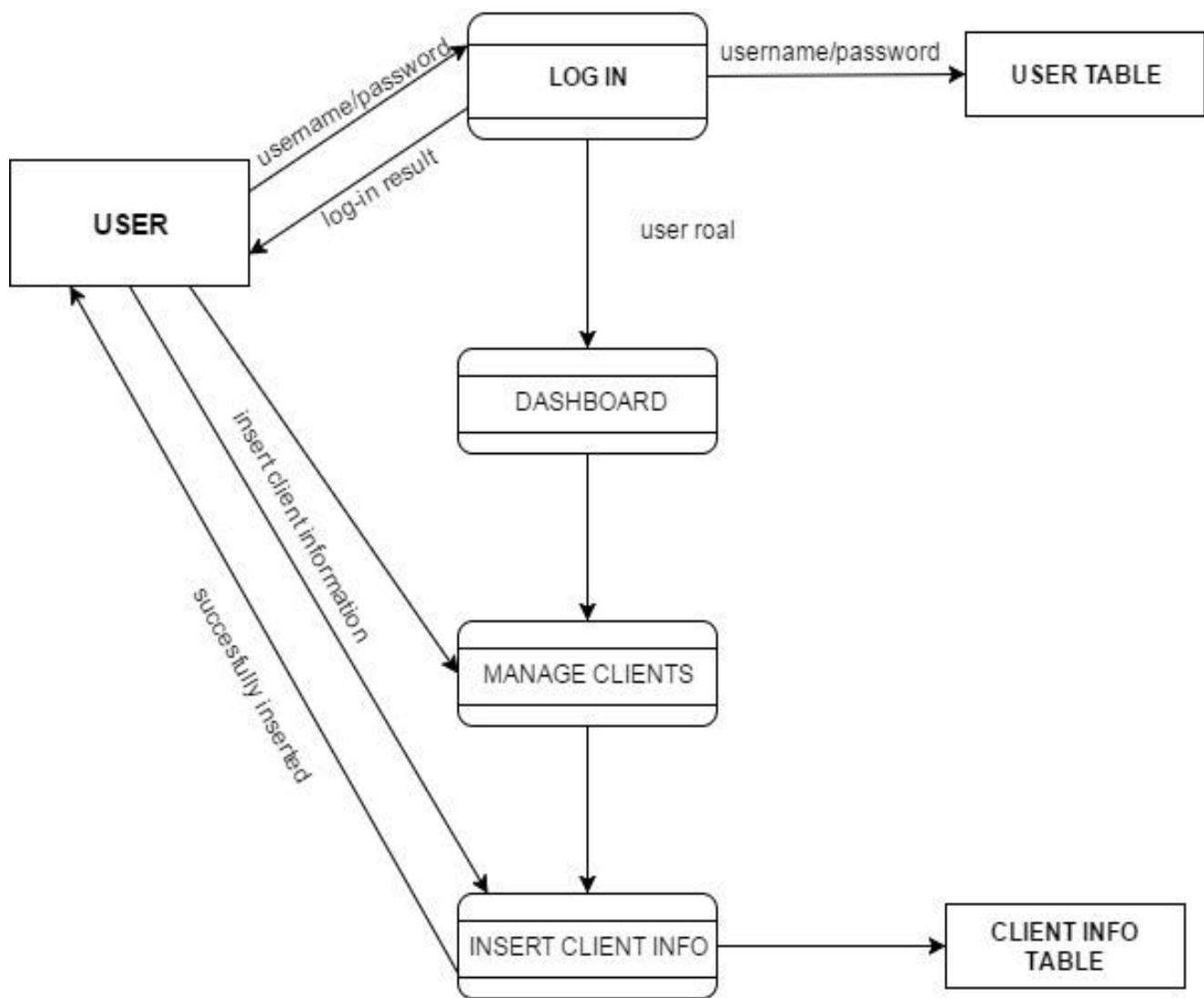
**CONTEXT DATA-FLOW DIAGRAM**



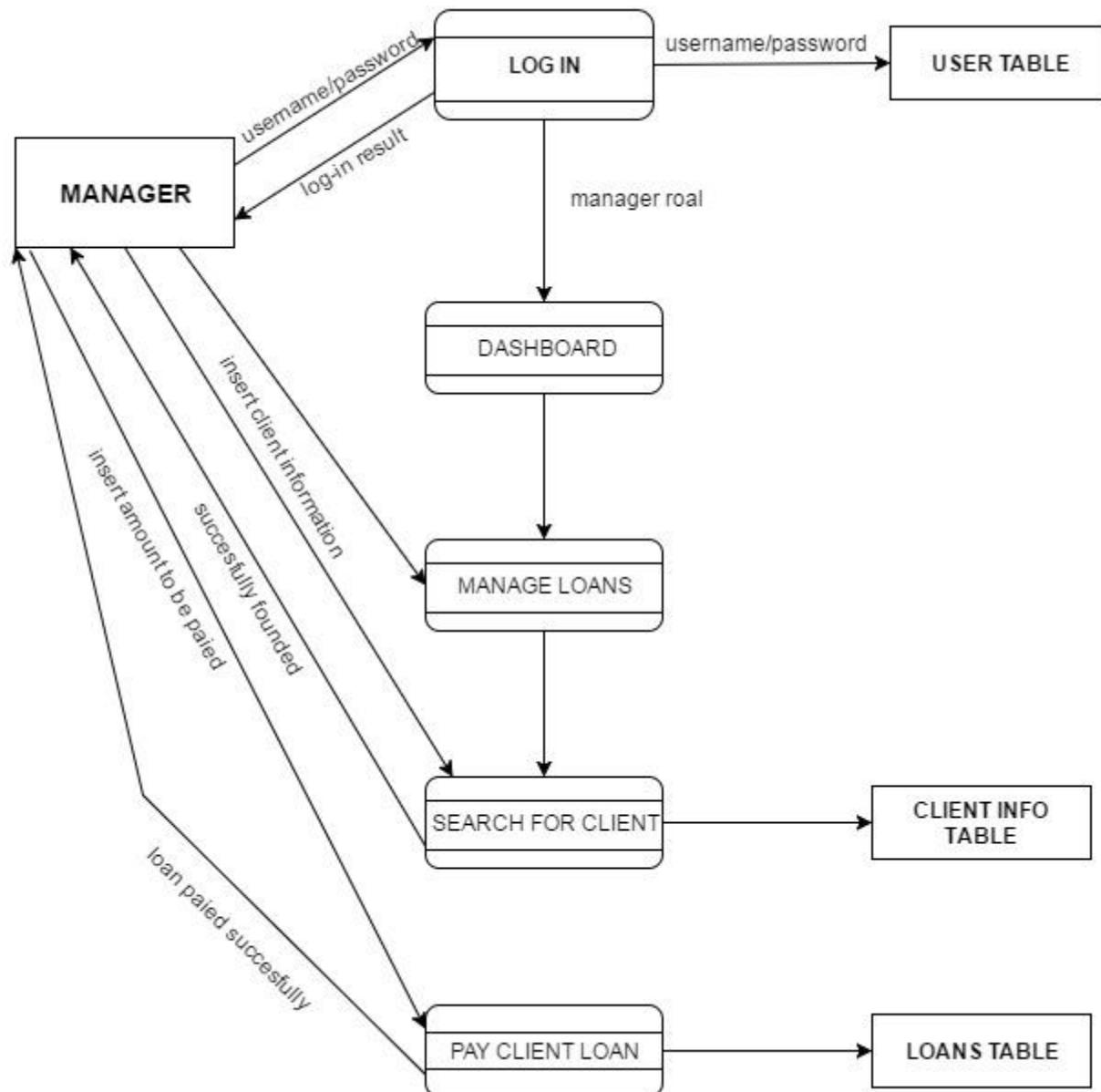
**DATA-FLOW DIAGRAM 1**



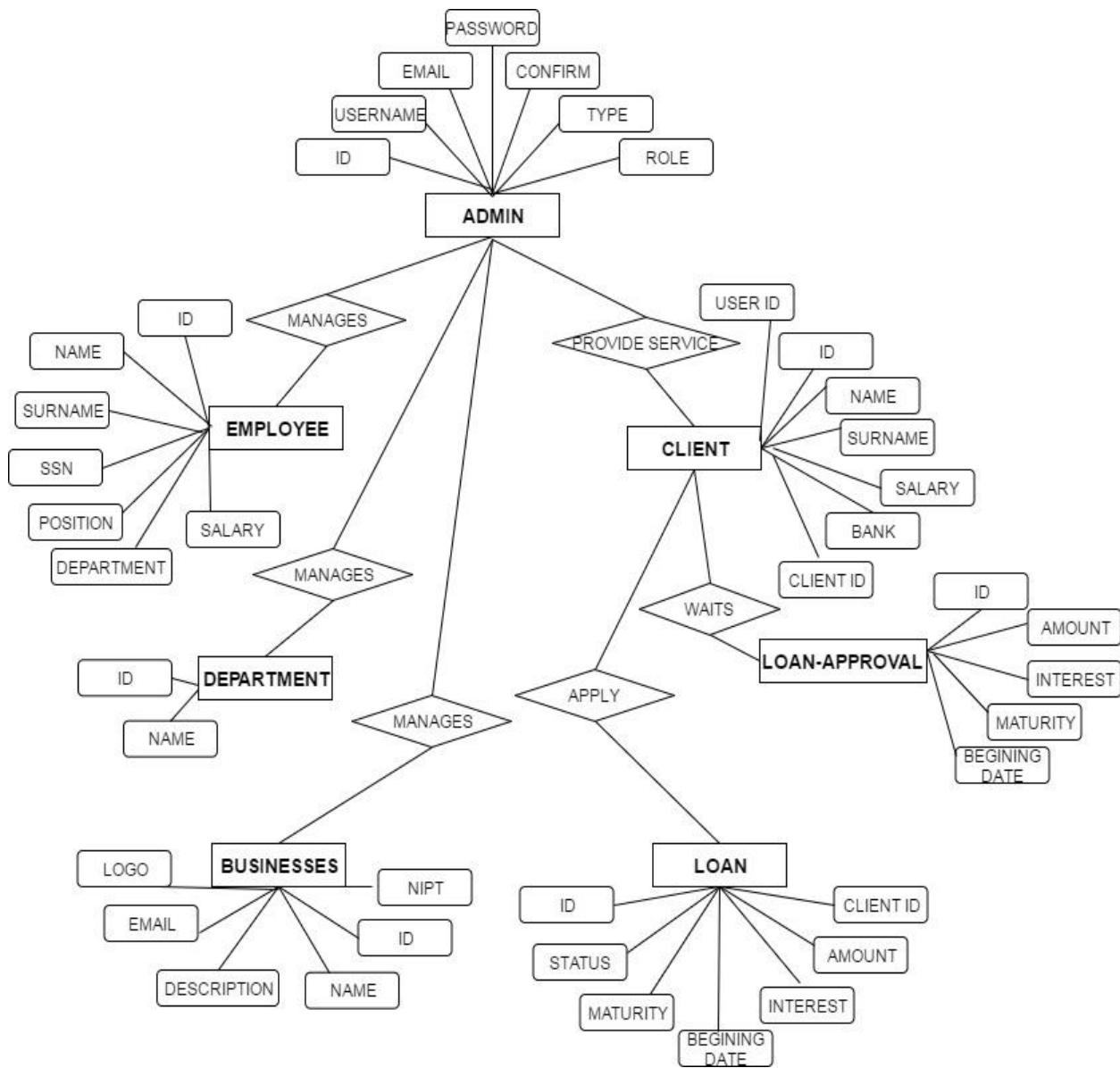
## DATA-FLOW DIAGRAM 2



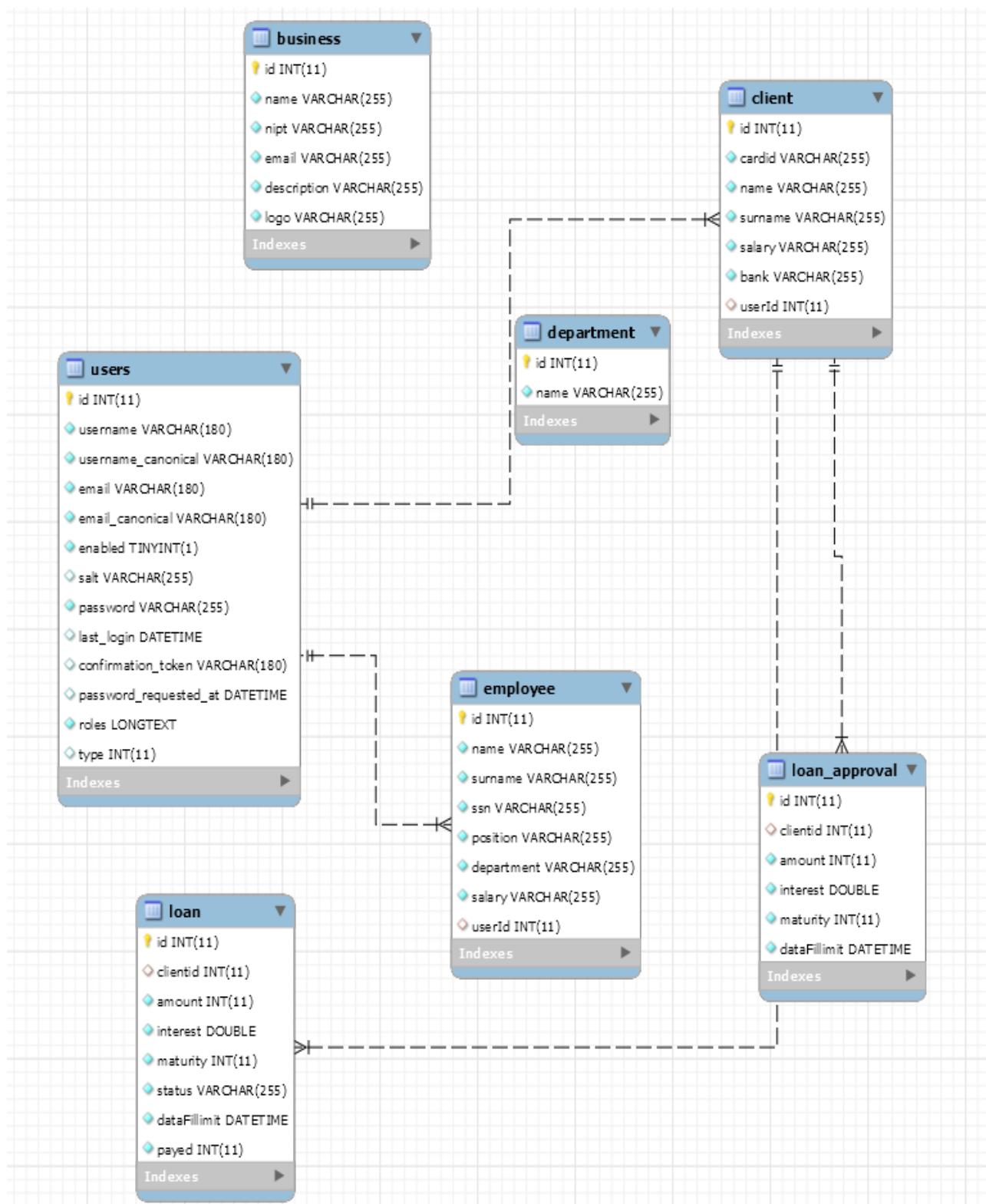
### DATA-FLOW DIAGRAM 3



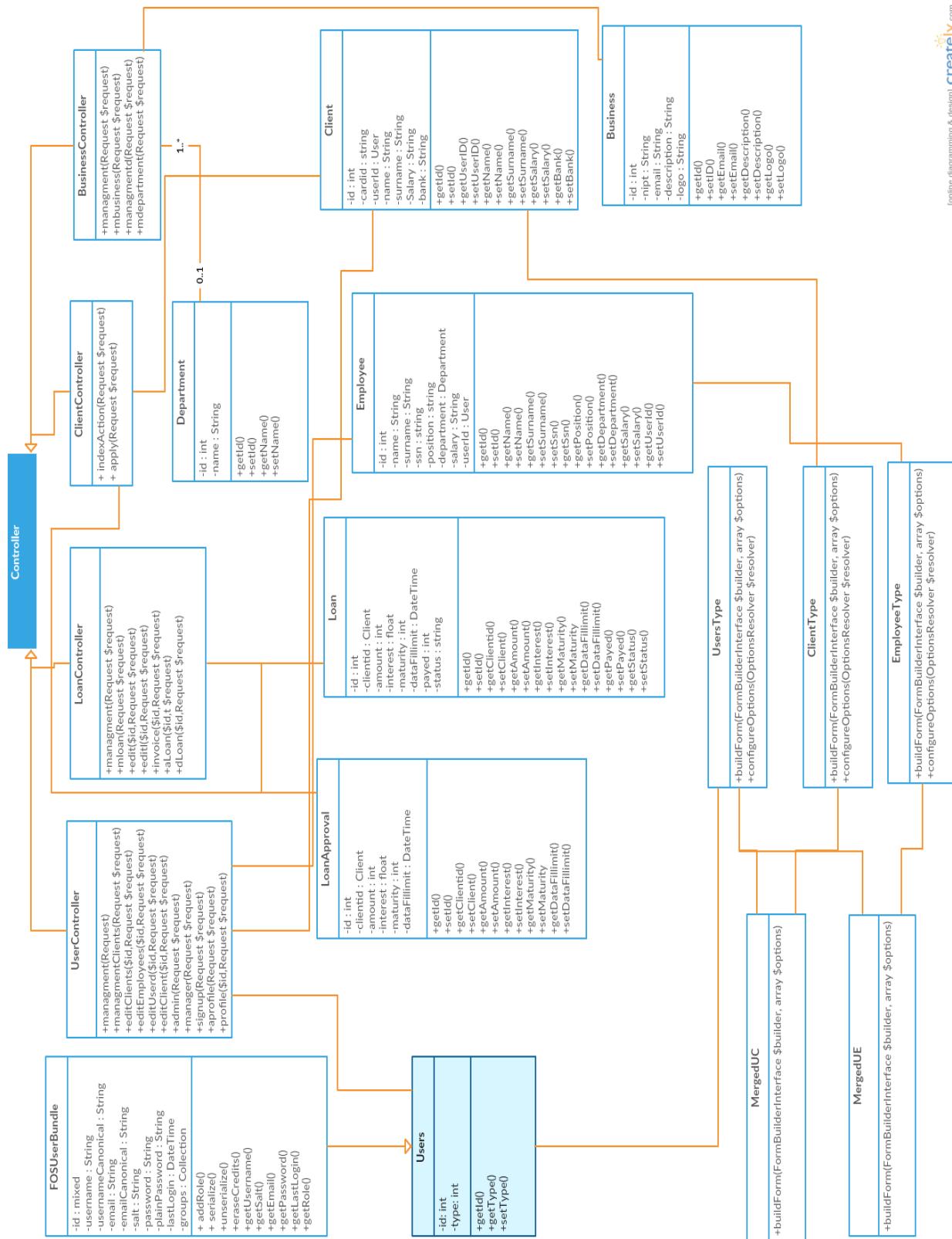
## ENTITY RELATIONSHIP DIAGRAM



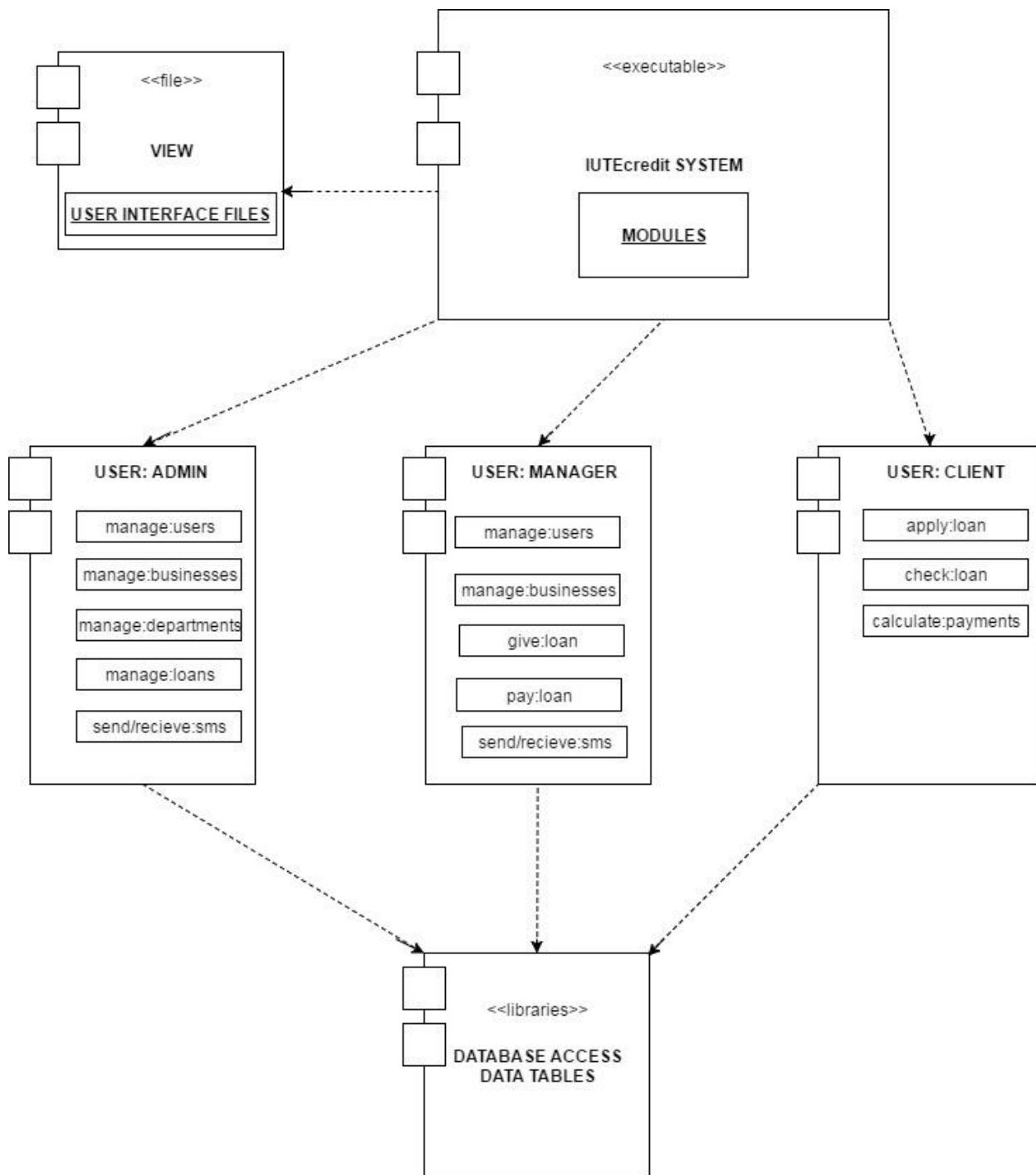
## DATABASE SCHEMA



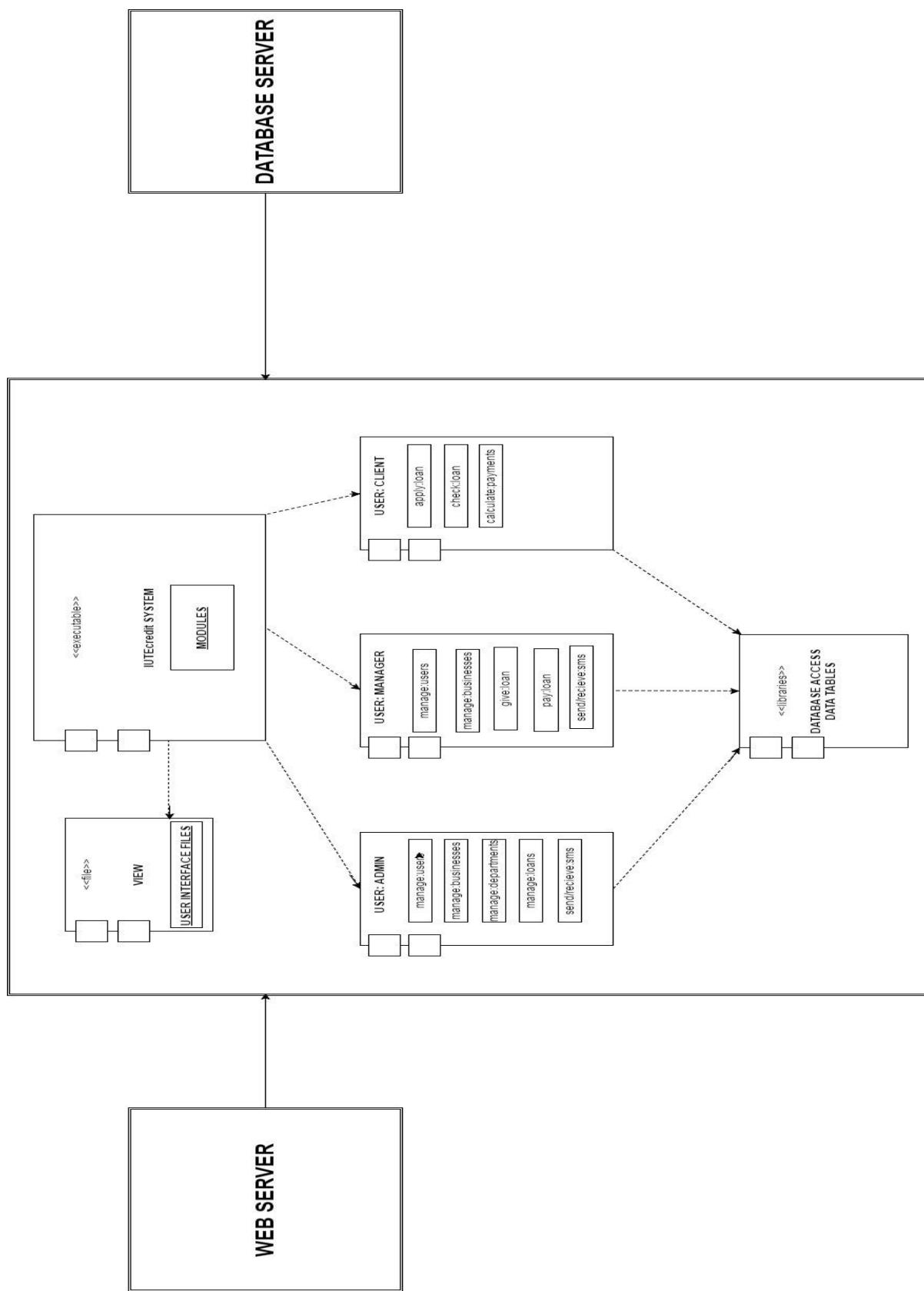
## CLASS DIAGRAM



## 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 classes 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
/**
```

```

 * 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_management", 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();
    $ul = new Users();
    $total1 = array(
        'user' => $ul,

```

```

        '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();
    $total['client']->setUserId($total1['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')));
                $form->handleRequest($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' => 'Passwordeo 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'))>
strtoime('-7 day')) {
                $ditet[$loans[$i]->getDataFillimit()->format('w')]++;
            }
            if(strtotime($loans[$i]->getDataFillimit()->format('Y-m-d H:i:s'))>
strtoime('-2 day') && strtotime($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();
    $ul = new Users();
    $total1=array(
        'user'=>$ul,
        '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 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) }}
            </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 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>
</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>
        </div>
    </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>


```

```

        </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>
{%
    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>
</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=''>{ { 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>

{ % 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:

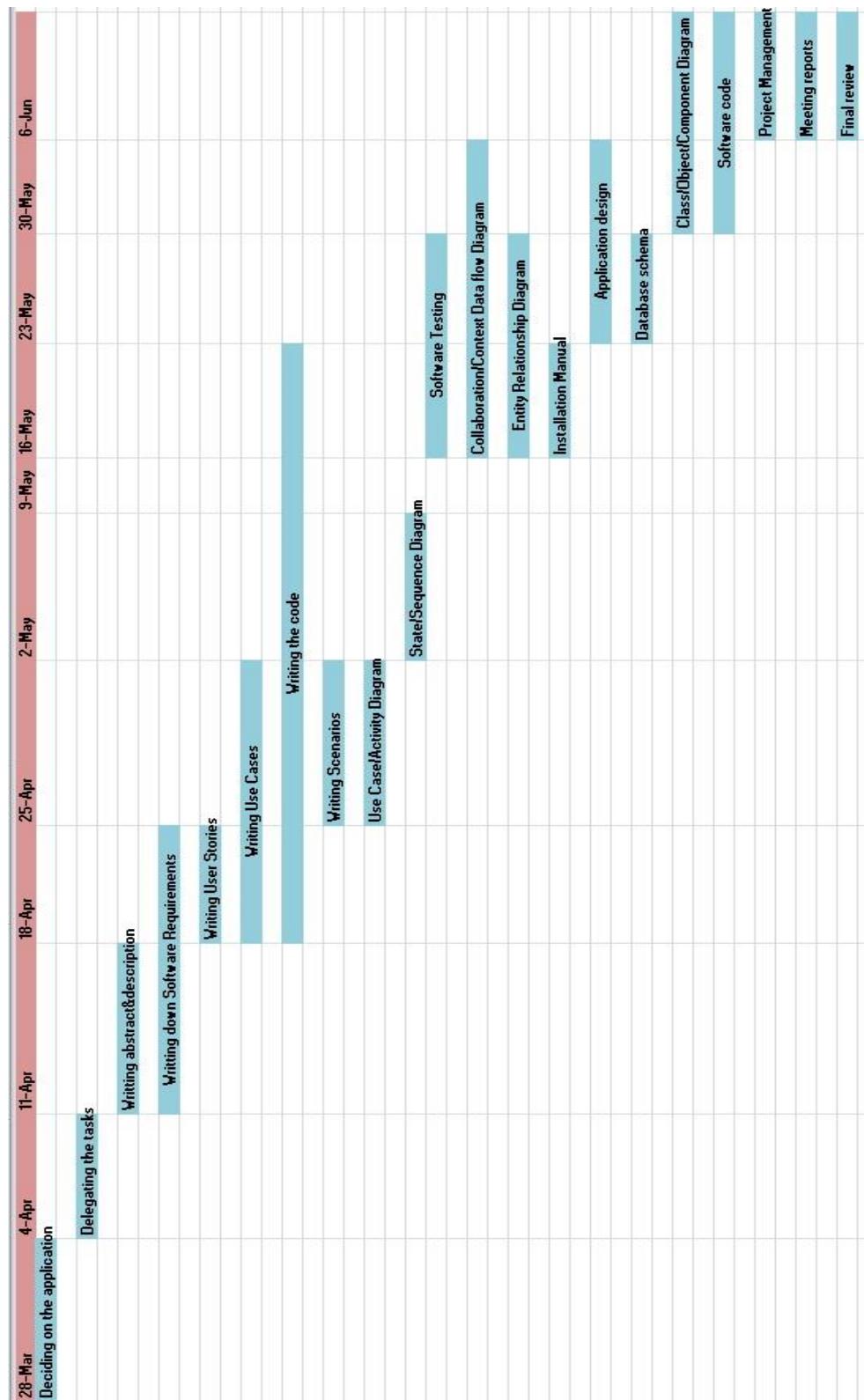
- a) Install XAMPP
- b) Unzip the IuteCredit.zip file located in the accompanying CD.
- c) Copy and paste the folder as it is in the folder named htdocs in the C:/xampp/htdocs
- d) Open xampp control panel and start apache and MySql servers.
- e) Open your browser and go to-> localhost/phpmyadmin
- f) 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<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member                                   | <ul style="list-style-type: none"> <li>Deciding on the application concept</li> </ul>                      |
| 28/03/2017   | Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member                                   | <ul style="list-style-type: none"> <li>Talking with IUTE Credit's owner</li> </ul>                         |
| 04/04/2017   | Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member                                   | <ul style="list-style-type: none"> <li>Task delegating</li> </ul>  |
| 07/04/2017   | Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member                                   | <ul style="list-style-type: none"> <li>Discussion about the project type and structure.</li> </ul>         |
| 11/04/2017   | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>Start writing SRS using templates as guidelines</li> </ul>          |
| 18/04/2017   | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>Start writing the code while adapting it to the document</li> </ul> |
| 25/04/2017   | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>Write scenarios</li> </ul>  |
| 02/05/2017   | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>Start working with diagrams</li> <li>Update document</li> </ul>     |
| 05/05/2017   | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>Continue coding</li> <li>Set goals for next meeting</li> </ul>      |

|                   |  |  |
|-------------------|--|--|
| <b>16/05/2017</b> | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>• Finish coding</li> <li>• Continue work with diagrams</li> </ul>             |
| <b>23/05/2017</b> | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>• Start working with design</li> <li>• Make neccessary updates</li> </ul>     |
| <b>30/05/2017</b> | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>• Write Software Testing and Code</li> <li>• Discuss final changes</li> </ul> |
| <b>06/06/2017</b> | Igli Hakrama - Project Advisor<br>Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member | <ul style="list-style-type: none"> <li>• Make last remarks</li> <li>• Complete Project Management</li> </ul>         |
| <b>10/06/2017</b> | Armeno Sadiku-Group Member<br>Ana Basha-Group Member<br>Xhulio Jamaku-Group Leader<br>Xhina Kamberi-Group Member                                   | <ul style="list-style-type: none"> <li>• Make a final review</li> </ul>  |

# Contact US

## OUR TEAM

Working hard to provide the best quality!



**Xhulio Jamaku**  
*Chief Executive Officer*

[Twitter](#) [Facebook](#) [Google+](#) [LinkedIn](#)



**Ana Basha**  
*Chief Financial Officer*

[Twitter](#) [Facebook](#) [Google+](#) [LinkedIn](#)



**Armeno Sadiku**  
*CTO*

[Twitter](#) [Facebook](#) [Google+](#) [LinkedIn](#)



**Xhina Kamberi**  
*Accountant*

[Twitter](#) [Facebook](#) [Google+](#) [LinkedIn](#)

## Emails:

**xjamaku14@epoka.edu.al**

**abasha14@epoka.edu.al**

**asadiku14@epoka.edu.al**

**xkamberi14@epoka.edu.al**