|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

|  |  |
| --- | --- |
| Capstone Project Document | |
| **Peer-to-Peer Lending System** | |
| **Group 10 - IS** | |
| **Group members** | Nguyễn Trung Dũng – SE62576 (Leader)  Huỳnh Văn Lộc – SE62291  Lê Ngô Minh – SE62705 |
| **Supervisor** | Nguyễn Huy Hùng |
| **Ext. Supervisor** | N/A |
| **Project Code** | PPLS |

– **Ho Chi Minh City, 13 May 2019** –

Table of contents

[List of tables 5](#_Toc16843112)

[List of Figures 10](#_Toc16843113)

[A. Introduction 13](#_Toc16843114)

[1. Project Information 13](#_Toc16843115)

[2. Introduction 13](#_Toc16843116)

[3. Current Situation 13](#_Toc16843117)

[4. Problem Definition 13](#_Toc16843118)

[5. Proposed Solution 14](#_Toc16843119)

[**5.1 Feature functions** 14](#_Toc16843120)

[**5.2** **Advantages and disadvantages** 14](#_Toc16843121)

[6. Functional Requirements 14](#_Toc16843122)

[7. Role & Responsibility 15](#_Toc16843123)

[B. Software Process Model 15](#_Toc16843124)

[1. Project Organization 15](#_Toc16843125)

[2. Project Management Plan 17](#_Toc16843126)

[C. Conceptual Diagram: 19](#_Toc16843127)

[D. Use-case Diagram: 22](#_Toc16843128)

[1. User Requirement Specification 22](#_Toc16843129)

[2. System Requirement Specification 22](#_Toc16843130)

[E. System Architectural Diagram 61](#_Toc16843131)

[F. Component Diagram 63](#_Toc16843132)

[G. Class Diagram 65](#_Toc16843133)

[**1.1.** **Class Diagram** 65](#_Toc16843134)

[**1.2.** **Class Diagram Explanation** 66](#_Toc16843135)

[H. Entity Relationship Diagram (ERD) 74](#_Toc16843136)

[1. Entity relationship diagram (ERD) 74](#_Toc16843137)

[2. Entity Data Dictionary 75](#_Toc16843138)

[I. Interaction Diagram (Sequence Diagram) 79](#_Toc16843139)

[J. Physical Diagram – Relationship Diagram 87](#_Toc16843140)

[**1.** **Physical diagram** 87](#_Toc16843141)

[**2.** **Data dictionary** 88](#_Toc16843142)

[K. Framework Architectural Diagram 89](#_Toc16843143)

[L. Algorithms 90](#_Toc16843144)

[**1. Record Transaction** 90](#_Toc16843145)

[**2. Interest lending** 91](#_Toc16843146)

[**3. Lending/Payback** 92](#_Toc16843147)

[M. Future Plan 94](#_Toc16843148)

[N. Explanation 94](#_Toc16843149)

## List of tables

[Table 1 - Roles and responsibilities 13](#_Toc16844302)

[Table 2 - Tools and techniques 13](#_Toc16844303)

[Table 3 - Deliverables 14](#_Toc16844304)

[Table 4 – Conceptual Diagram Data Dictionary 17](#_Toc16844305)

[Table 5– <Authorized User> Create request use case specification 24](#_Toc16844306)

[Table 6 – <Authorized User> View own request use case specification 26](#_Toc16844307)

[Table 7 – <Authorized User> View trading request use case specification 28](#_Toc16844308)

[Table 8 – <Authorized User> View history request use case specification 30](#_Toc16844309)

[Table 9 – <Authorized User> View lending request use case specification 32](#_Toc16844310)

[Table 10 – <Authorized User> Do payment use case specification 33](#_Toc16844311)

[Table 11 – <Authorized User> Accept request use case specification 35](#_Toc16844312)

[Table 12 – <Authorized User> Make deal use case specification 37](#_Toc16844313)

[Table 13 – <Authorized User> View detail request use case specification 39](#_Toc16844314)

[Table 14 – <Authorized User> Valid transaction with blockchain use case specification 41](#_Toc16844315)

[Table 15 – <Authorized User> Upload document use case specification 44](#_Toc16844316)

[Table 16 – <Admin> Accept document use case specification 47](#_Toc16844317)

[Table 17 – <Admin> Reject document use case specification 50](#_Toc16844318)

[Table 18 – <System> Process make transaction use case specification 52](#_Toc16844319)

[Table 19 – <System> Process save transaction use case specification 54](#_Toc16844320)

[Table 20 – Conceptual Diagram Data Dictionary 56](#_Toc16844321)

[Table 21 – Front End Component Diagram Dictionary 60](#_Toc16844322)

[Table 22– Back End Component Diagram Dictionary 60](#_Toc16844323)

[Table 23 – Class Diagram PPLS 62](#_Toc16844324)

[Table 24 – <Class Diagram Attributes> User 63](#_Toc16844325)

[Table 25 – <Class Diagram Methods> User 63](#_Toc16844326)

[Table 26 – <Class Diagram Attributes> Request 64](#_Toc16844327)

[Table 27 – <Class Diagram Methods> Request 64](#_Toc16844328)

[Table 28 – <Class Diagram Attributes> Deal 65](#_Toc16844329)

[Table 29 – <Class Diagram Methods> Deal 65](#_Toc16844330)

[Table 30 – <Class Diagram Attributes> Milestone 65](#_Toc16844331)

[Table 31 – <Class Diagram Methods> Milestone 66](#_Toc16844332)

[Table 32 – <Class Diagram Attributes> Transaction 66](#_Toc16844333)

[Table 33 – <Class Diagram Methods> Transaction 66](#_Toc16844334)

[Table 34 – <Class Diagram Attributes> Backup Deal 67](#_Toc16844335)

[Table 35 – <Class Diagram Methods> Backup Deal 67](#_Toc16844336)

[Table 36 – <Class Diagram Attributes> Backup milestone 67](#_Toc16844337)

[Table 37 – <Class Diagram Methods> Backup milestone 67](#_Toc16844338)

[Table 38 – <Class Diagram Attributes> Document 68](#_Toc16844339)

[Table 39 – <Class Diagram Methods> Document 68](#_Toc16844340)

[Table 40 – <Class Diagram Attributes> Document Type 69](#_Toc16844341)

[Table 41 – <Class Diagram Attributes> DocumentFile 69](#_Toc16844342)

[Table 42 Entity Relationship Diagram Dictionary 72](#_Toc16844343)

[Table 43 - ERD Data Dictionary 75](#_Toc16844344)

[Table 44 Physical Diagram Dictionary 84](#_Toc16844345)

## List of Figures

[Figure 1 - Scrum Framework 12](#_Toc16844346)

[Figure 2- Conceptual Diagram 15](#_Toc16844347)

[Figure 3 – System overview use case 21](#_Toc16844348)

[Figure 4 – Authorized User Overview Use Case 22](#_Toc16844349)

[Figure 5 - <Authorized User> Overview use case 23](#_Toc16844350)

[Figure 6 – <Authorized User> Create Request Use Case 23](#_Toc16844351)

[Figure 7 – <Authorized User> View Own Request Use Case 25](#_Toc16844352)

[Figure 8 – <Authorized User> View trading request use case 27](#_Toc16844353)

[Figure 9 – <Authorized User> View History Request Use Case 29](#_Toc16844354)

[Figure 10 – <Authorized User> View Lending Request Use Case 31](#_Toc16844355)

[Figure 11 – <Authorized User> Do Payment Use Case 32](#_Toc16844356)

[Figure 12 – <Authorized User> Accept Request Use Case 34](#_Toc16844357)

[Figure 13 – <Authorized User> Make Deal Use Case 36](#_Toc16844358)

[Figure 14 – <Authorized User> View Detail Request Use Case 38](#_Toc16844359)

[Figure 15 – <Authorized User> Valid Transaction With Blockchain Use Case 40](#_Toc16844360)

[Figure 16 – <Authorized User> Upload Document Use Case 42](#_Toc16844361)

[Figure 17 – <Admin> Overview Use Case 45](#_Toc16844362)

[Figure 18 – <Admin> Accept Document Use Case 45](#_Toc16844363)

[Figure 19– <Admin> Reject Document Use Case 48](#_Toc16844364)

[Figure 20 – <System> Overview Use Case 50](#_Toc16844365)

[Figure 21 – <System> Process Make Transaction Use Case 51](#_Toc16844366)

[Figure 22 – <System> Process Save Transaction Use Case 52](#_Toc16844367)

[Figure 23 – Software Architecture Design 57](#_Toc16844368)

[Figure 24 – Front End User Component Diagram 59](#_Toc16844369)

[Figure 25 – Back End Component Diagram 60](#_Toc16844370)

[Figure 26 – Class Diagram 61](#_Toc16844371)

[Figure 27 Entity Relationship Diagram 70](#_Toc16844372)

[Figure 28 <Sequence Diagram> Guest Login 76](#_Toc16844373)

[Figure 29 <Sequence Diagram> Accept Deal 77](#_Toc16844374)

[Figure 30 <Sequence Diagram> Create request 78](#_Toc16844375)

[Figure 31 <Sequence Diagram> Make deal 80](#_Toc16844376)

[Figure 32 <Sequence Diagram> Valid document 81](#_Toc16844377)

[Figure 33 <Sequence Diagram> Do payment and sent to BigchainDB 82](#_Toc16844378)

[Figure 34 Physical Diagram 83](#_Toc16844379)

[Figure 35 – Blockchain Architecture Diagram - BigchainDB 85](#_Toc16844380)

[Figure 36 – ReactJS Architectural Diagram 85](#_Toc16844381)

[Figure 37 – Spring Boot Architecture Diagram 86](#_Toc16844382)

[Figure 38. Example of BigchainDB transaction 87](#_Toc16844383)

[Figure 39– Lending/Payback process 89](#_Toc16844384)

# Introduction

## Project Information

* Project name:  **Peer-to-peer Lending System**
* Project Code: **PPLS**
* Product Type: **Website Application**
* Start Date: **May 13rd, 2019**
* End Date: **--**

## Introduction

These days, borrowing money is no longer strange. Borrowing money to invest in a project or to buy something like washing machine, phone, etc.… that can be seen every day around our lives. However, borrowing money from banks still exists many cumbersome procedures that make borrowers encounter many difficulties in borrowing money. Moreover, whether transactions can be changed, or even deleted to protect an individual may cause someone to be guilty.

In this document, we introduce a solution to make loan transactions transparent, a web application called Peer-to-peer Lending System. This Peer-to-peer Lending System helps people borrow money directly by making an agreement with the lender, from which all transactions will be stored on the Blockchain and viewed by anyone in the system.

## Current Situation

As mentioned in the introduction, borrowing money causes many difficulties for borrowers because many procedures need to be authenticated by the authorities. This makes many people in urgent need to earn illegal money such as interest loans.

Also, trust between people and people is still deficient to be able to lend directly between two people. That's why creating trust among people is a necessity.

## Problem Definition

Below are the disadvantages of the current situation:

* Currently, if you want to borrow money from the bank, you need a lot of documents, as well as many complicated procedures. Not to mention whether you are approved for a loan after going through a customer appraisal step.
* The current interest rate is quite high. For unsecured loans VPBank from 15.91-21% / year, TPBank from 8.28 - 17% / year, ACB up to 22% / year. As for mortgage loans, about 5-12%/year.
* The amount can be borrowed from the bank is not too much. The maximum limit of banks falls between 300 million VND and 1 billion VND.

## Proposed Solution

We will build a website using blockchain to reduce the time of borrowing money from banks and increase the borrowing limit to higher than the current banks. Features will be implemented in the system:

### **5.1 Feature functions**

* Create requirements to borrow money from people who need to borrow money.
* Create and set the necessary milestones in borrowing as well as the most appropriate payments created by borrowers and lenders.
* Store transactions using Blockchain
* Use online payment gateways for transferring loans as well as paying such as Paypal, Ethereum, Momo, ….

### **5.2** **Advantages and disadvantages**

The advantages and disadvantages of the proposed solution:

* **Advantages**:
* Optimize time to do loan procedures.
* The loan limit is not dependent on banks so the limit may be much higher.
* **Disadvantages**:
* There are still risks when lending.

## Functional Requirements

Function requirements of the system are listed below:

**6.1 Admin:**

* Admin can activate, or deactivate the account.
* Amin can view list users.
* Admin can approvedocuments for increasing loan limit of user.
* Admin can reject document of user.
* Admin can manage document type.

**6.2 Authorized User:**

* User can create a borrowing request.
* User can detect a borrowing request.
* User can view detail of a request.
* User can accept request after view detail.
* User can make a deal with the chosen request.
* User can view them all own transactions.
* User can view them all history requests.
* User can upload identity document for verifying to get loan limit

**6.3 Guest:**

* User can sign up a new account.
* User can log in into the system.
* User can view all transactions.

## Role & Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Nguyễn Huy Hùng | Project Manager | Supervisor | hungnh@fpt.edu.vn |
| 2 | Nguyễn Trung Dũng | Developer | Leader | dungntse62576@fpt.edu.vn |
| 3 | Huỳnh Văn Lộc | Developer | Member | lochvse62291@fpt.edu.vn |
| 4 | Lê Ngô Minh | Developer | Member | minhlnse62705@fpt.edu.vn |

# Software Process Model

## Project Organization

* 1. **Software Process Model**

This project is developed using the Scrum model – part of an agile framework for the Software development project. Our team chooses the Scrum model because of the following reasons:

* Our team only has three members, and tasks are assigned vertically, do all steps from design, coding, testing, and implementation. Scrum is the most suitable model for the small and medium project.
* After research about BigchainDB, Flask framework, React Framework, we have defined problems. The risk of changing algorithm is high because proving the accuracy of those algorithms is complicated. We need to use “try and test” method.
* There are many new technologies that need to be learned in the project. With the Scrum model, the team can be learned and develop in parallel to meet the deadline.
* There is no leader, no hierarchy in team, so team members work cheerfully, stimulating the initiative and creativity of each member.
* The product owner can change the requirement or extend the scope. The team will adapt to change better.

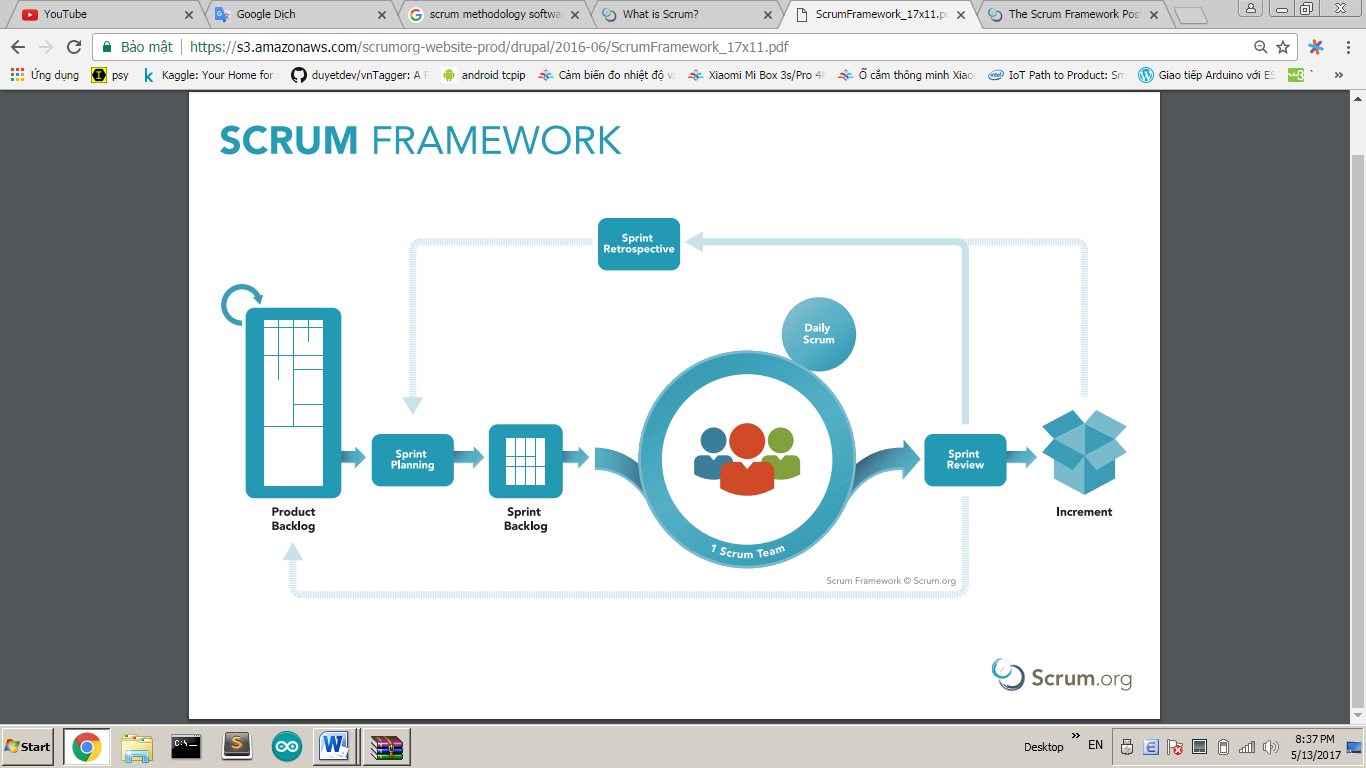


Figure 1 - Scrum Framework

* 1. **Roles and responsibilities**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| **1** | Nguyễn Huy Hùng | Product Owner | * Specify user requirement * Control the development process * Give out technique and business analysis support |
| **2** | Nguyễn Trung Dũng | Scrum Master | * Managing process * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create a test plan * Coding * Testing * Arrange Meeting * Risk Management |
| **3** | Huỳnh Văn Lộc | Scrum team member | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create a test plan * Coding * Testing |
| **4** | Lê Ngô Minh | Scrum team member | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create a test plan * Coding * Testing |

Table 1 - Roles and responsibilities

* 1. **Tools and Techniques**

|  |  |
| --- | --- |
| Tool/Technique | Name |
| Front-end | HTML, CSS, JavaScript, ReactJS, Reactstrap. |
| Back-end | JavaEE, SpringBoot framework, JPA, Hibernate, Python, Flask framework |
| IDE | Eclipse, IntelliJ IDEA 17.2 |
| DBMS | MySQL |
| Source Control | Github |
| Modeling tool | StarUML, Software Idea Modelers |

Table 2 - Tools and techniques

## 2. Project Management Plan

* 1. **Product Backlog**

Product Backlog could be found here.

* 1. **Sprint Backlog**

Sprint Backlog can be found [here](https://docs.google.com/spreadsheets/d/1DvHD5Bwmk1IL5XokeVP7r67h5yaPtwHpret4qHMtcmw/edit?usp=sharing)

* 1. **Deliverables**

|  |  |  |
| --- | --- | --- |
| **No** | **Deliverable** | **Note** |
| 1 | Introduction, Entity Relationship Diagram, Use Case Overview, Mock UI | Sprint 1 |
| 2 | Study Spring Boot Framework, React JS, BigchainDB, Design User Interface for Web Application | Sprint 2 |
| … |  |  |
|  |  |  |

Table 3 - Deliverables

* 1. **All Meeting Minutes**

All sprint meeting minutes could be found here.

# Conceptual Diagram:

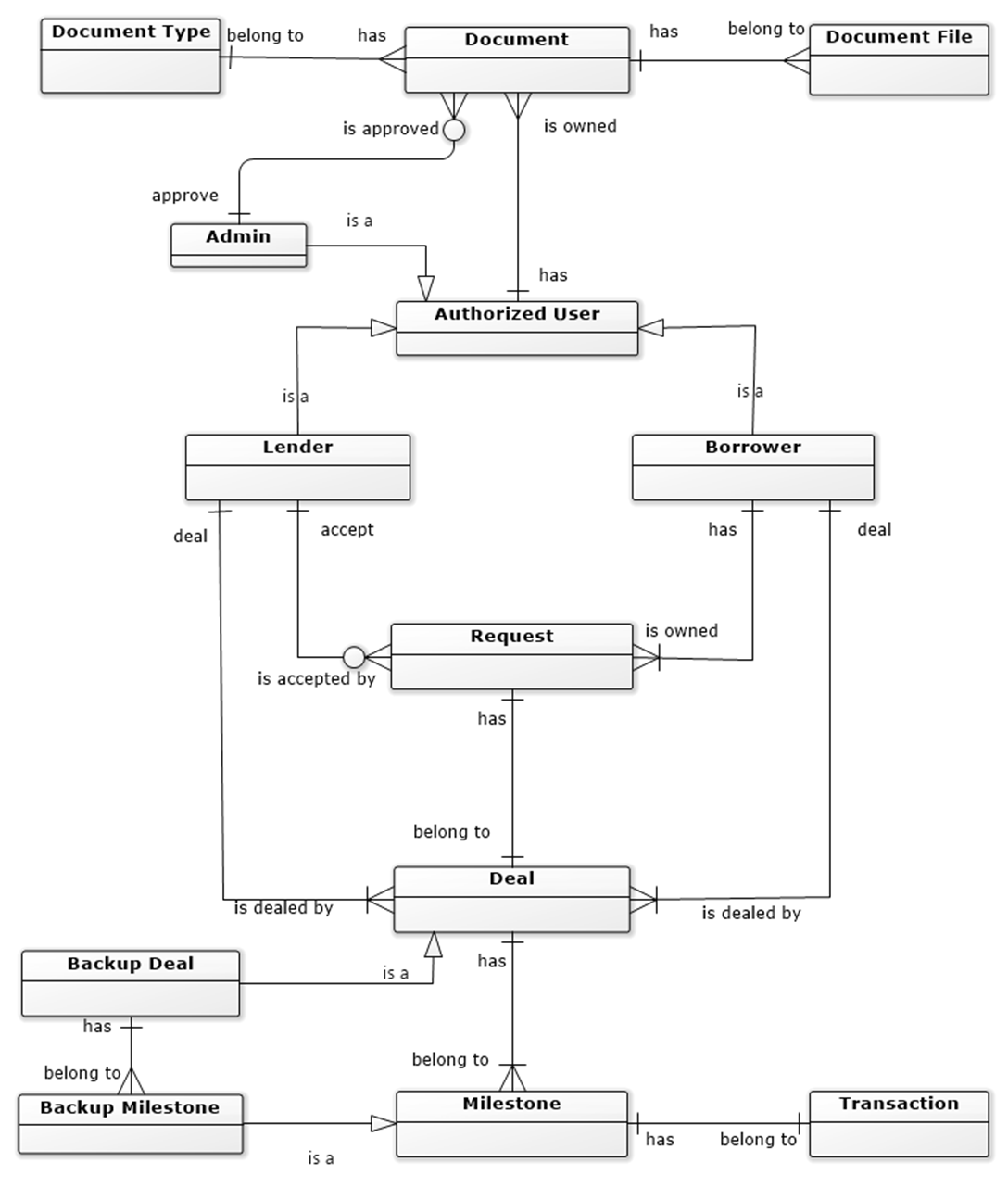


Figure 2- Conceptual Diagram

**Data Dictionary**

|  |  |
| --- | --- |
| **Entity Data dictionary: describe all content of all entities** | |
| **Entity Name** | **Description** |
| Authorized User | Abstract entity describes a user in the system. Contain user information include:   * ID * Username * FirstName * LastName * Password * Status * Role |
| Lender | Contain the lender information (authorized user information). |
| Borrower | Contain the borrower information (authorized user information). |
| Admin | Contain the admin information (authorized user information). |
| Request | Contain the request information include:   * ID * LenderID * BorrowerID * Amount * InterestRate * CreateDate * BorrowDate * Status |
| Deal | Contain the deal information include:   * ID * BorrowTimes * PaybackTimes * Status |
| Backup Deal | Contain the backup deal information (deal information), represent for the first deal |
| Milestone | Contain the milestone information include:   * ID * PreviousDate * PresentDate * Type * Percent |
| Backup Milestone | Contain the backup milestone information (milestone information), represent for the first milestone |
| Document | Contain the document information include:   * ID * DocumentID * Status |
| Document Type | Contain the document type information include:   * ID * FileName * FileType * Data |
| Document File | Contain the document file information include:   * ID * Name * AmountLimit * Acronym |

Table 4 – Conceptual Diagram Data Dictionary

# Use-case Diagram:

## User Requirement Specification

* 1. **Guest Requirement**

*Guest is a person who doesn’t have access to the system. Guest-only tries some demo features. To use all functions, guest must log in. These are some functions guest can use:*

* Login.
* View History Transactions.
  1. **Authorized User Requirement**

*Authorized User is an authorized user who has permission to login into the system. With authorized user role, user can use below functions:*

* Create borrowing requests.
* View list borrowing requests.
* View list pending requests.
* View list dealing requests.
* View list history requests.
* View list trading requests.
* View list owns requests.
* View detail of a request.
* Accept request after view detail.
* Cancel a request after view list dealing requests.
* Make a deal with the chosen request.
* Pay for the deal borrowed.
  1. **Admin Requirement**

*Admin is an authorized user who has permission to login into the system under the Admin role. Admin can use below management functions:*

* Active/Deactive User account.
* View all pending documents.
* Accept user documents.
* Reject user documents.

## System Requirement Specification

* 1. **External Interface Requirement**
     1. **User Interface**
* GUI should be simple, clear, intuitive, and reminiscent.
* The interface design is an iterate process includes design, sketching, prototyping, user assessment.
* The User Interface uses language is English.
  + 1. **Hardware Interface**
* Desktop PC.
  + 1. **Software Interface**
* Web application: work with browsers Firefox (v52 or higher), Chrome (v28 or higher), Internet Explorer (v10 or above) or with any web browser that supports HTML5 & CSS3.
  + 1. **Communication Protocol**

Use HTTP protocol 1.1 for communication between:

Web application and web server

* 1. **System Overview Use Case**

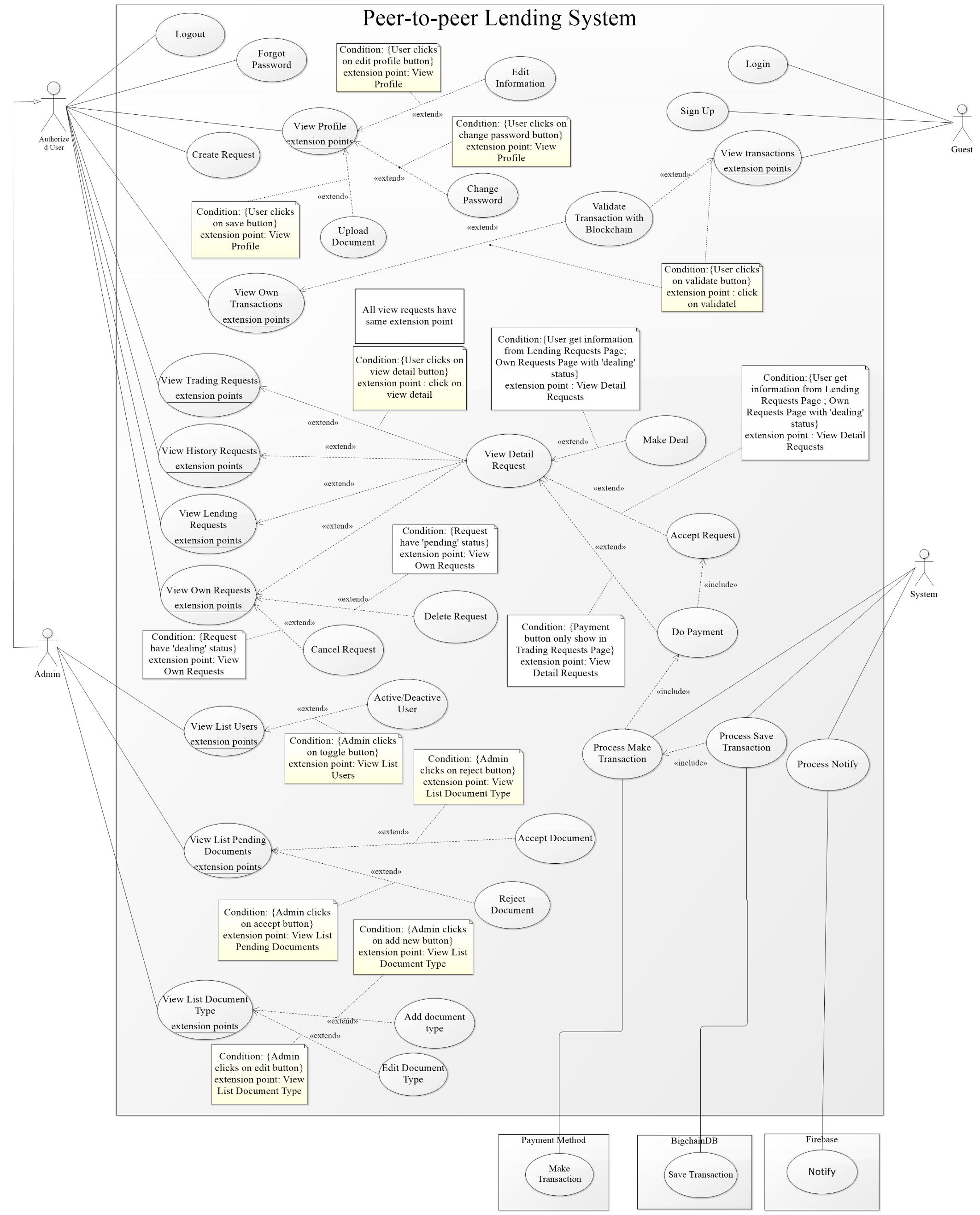


Figure 3 – System overview use case

* 1. **List of use case**
     1. **<Authorized User> Overview Use Case**

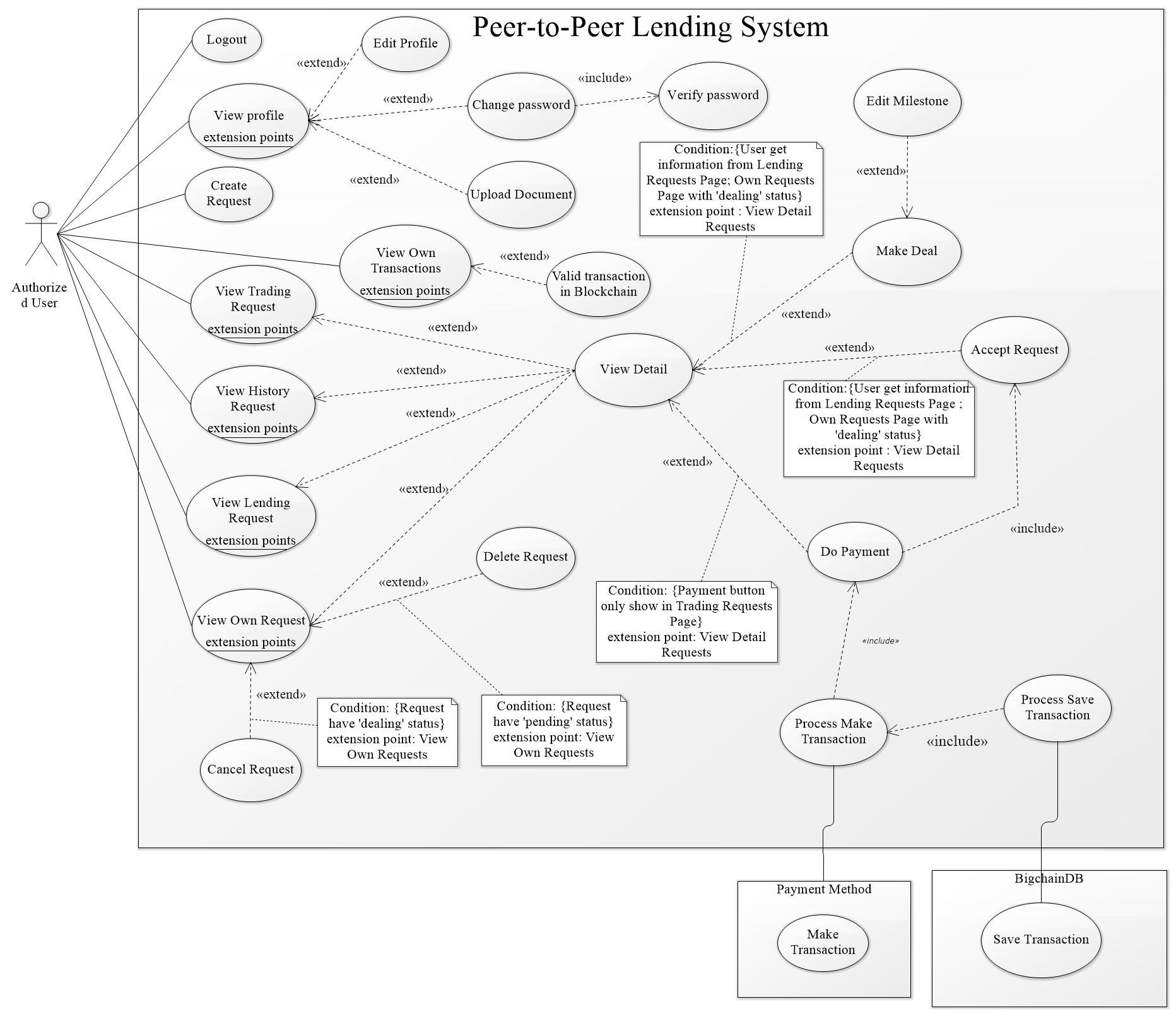
****

Figure 4 – Authorized User Overview Use Case

Figure 5 - <Authorized User> Overview use case

* + - 1. **<Authorized User> Create Request**

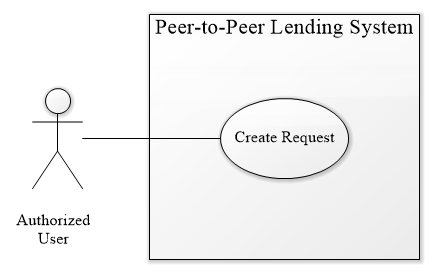


Figure 6 – <Authorized User> Create Request Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_05** | | | |
| **Use Case No.** | PPLS\_AU\_05 | **Use Case Version** | 1.0 |
| **Use Case Name** | Create request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to create borrow request.   **Goal:**   * Authorized User can create a request to borrow money.   **Triggers:**   * Authorized User sends create request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User create a borrow request. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Go to create request page. | System show creates request form:   * Amount: Input type number * Interest rate: text * Lending timeline * Payback timeline | | 2 | Authorized user enter information into the form | [Alternative 1] [Alternative 2] | | 3 | Authorized user click create request button | System shows dialog success and redirect to view own request page.  [Exception 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Guest enter the wrong format. | The system shows error message. Ex: “The amount must be a multiple of 500.000 VND!”, “The amount must be lower than 1 billion!”, “The amount”. | | 2 | The guest not enough loan limit | The system shows the error message “The amount must be lower than loan limit.”. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to the login page when the token timeout |   **Relationships:** N/A  **Business Rules:**   * Amount less than 1.000.000.000 VND * Amount must be a multiple of 500.000 VND * Interest rate 18% per year * Create success will redirect to view own request page. | | | |

Table 5– <Authorized User> Create request use case specification

* + - 1. **<Authorized User> View Own Request**

****

Figure 7 – <Authorized User> View Own Request Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_06** | | | |
| **Use Case No.** | PPLS\_AU\_06 | **Use Case Version** | 1.0 |
| **Use Case Name** | View own request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to view their request include pending request (new request) and dealing request (request is dealing).   **Goal:**   * Authorized User can view their request.   **Triggers:**   * Authorized User sends view own request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User can view their request. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view own request. | System show view own request page with two tabs include pending request and dealing request.  Each tab includes information:   * ID * Amount * Create Date * Duration * Status * View Detail * Delete (Pending Request) * Cancel (Dealing Request)   [Exception 1][Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view own request. | System show view own request page with two tabs and each tab not contain data yet. System show message “No data” |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_07, PPLS\_AU\_08, PPLS\_AU\_15  **Business Rules:**   * View own request page will get all user’s request with pending and dealing status. * This page use paging. * When delete a pending request, remove a pending request from database (pending request do not need to keep in database so that we decide to remove it) * When cancel deal, system restore original deal automatically. * If user do not have request in each tab, system show message “No data”. | | | |

Table 6 – <Authorized User> View own request use case specification

* + - 1. **<Authorized User> View Trading Request**

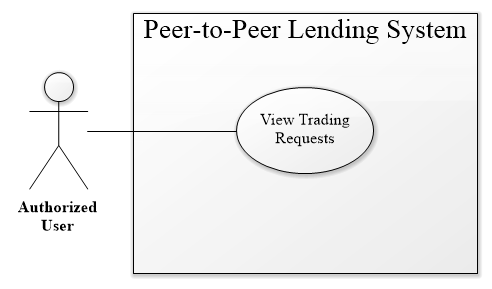
****

Figure 8 – <Authorized User> View trading request use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_09** | | | |
| **Use Case No.** | PPLS\_AU\_09 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Trading Request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to view their trading request (request is in the process).   **Goal:**   * Authorized User can view their request that is in process.   **Triggers:**   * Authorized User sends view trading request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User can view their trading request. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view trading request page. | System show trading request with two tabs include borrow and lend request.  Each tab includes information:   * ID * Amount * Create Date * User * View Detail   [Exception 1] [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view trading request. | System show view trading request page with two tabs and each tab not contain data yet. System show message “No data” |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_15, PPLS\_AU\_12  **Business Rules:**   * View trading request page will get all user’s request with trading status, split to borrow and lend request. * This page use paging. * If user do not have request in each tab, system show message “No data”. | | | |

Table 7 – <Authorized User> View trading request use case specification

* + - 1. **<Authorized User> View History Request**

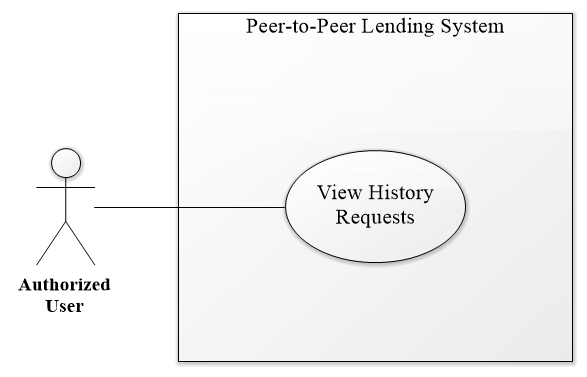
****

Figure 9 – <Authorized User> View History Request Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_10** | | | |
| **Use Case No.** | PPLS\_AU\_10 | **Use Case Version** | 1.0 |
| **Use Case Name** | View History Request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to view history request (Request had done).   **Goal:**   * Authorized User view request that had done.   **Triggers:**   * Authorized User sends view history request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User can view history request. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view history request. | System shows view history request page includes information:   * ID * Amount * Create Date * Status * View Detail   [Exception 1] [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view history request. | System show view history request page. System show message “No data” if user has no data. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_15  **Business Rules:**   * View history request page will get all user’s request with done status. * This page use paging. * If user do not have request in each tab, system show message “No data”. | | | |

Table 8 – <Authorized User> View history request use case specification

* + - 1. **<Authorized User> View Lending Request**

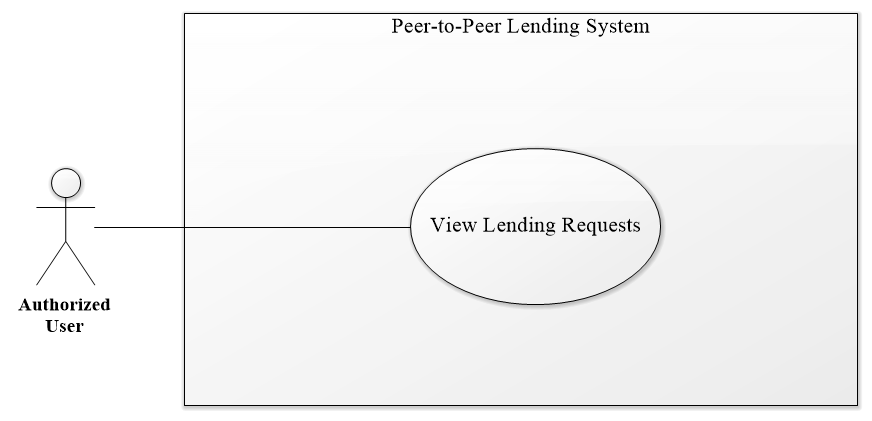
****

Figure 10 – <Authorized User> View Lending Request Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_11** | | | |
| **Use Case No.** | PPLS\_AU\_11 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Lending Request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to view the other request (borrow request from the other user).   **Goal:**   * Authorized User view borrows request.   **Triggers:**   * Authorized User sends view lending request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User can view lending request. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view lending request. | System show view lending request page includes information:   * ID * Amount * User * Create Date * View Detail   [Exception 1] [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User goes to view lending request. | System show view lending request page. System show message “No data” if user has no data. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_15  **Business Rules:**   * View lending request page will get all request from other user with pending status * This page use paging. * If user do not have request in each tab, system show message “No data”. | | | |

Table 9 – <Authorized User> View lending request use case specification

* + - 1. **<Authorized User> Do Payment**

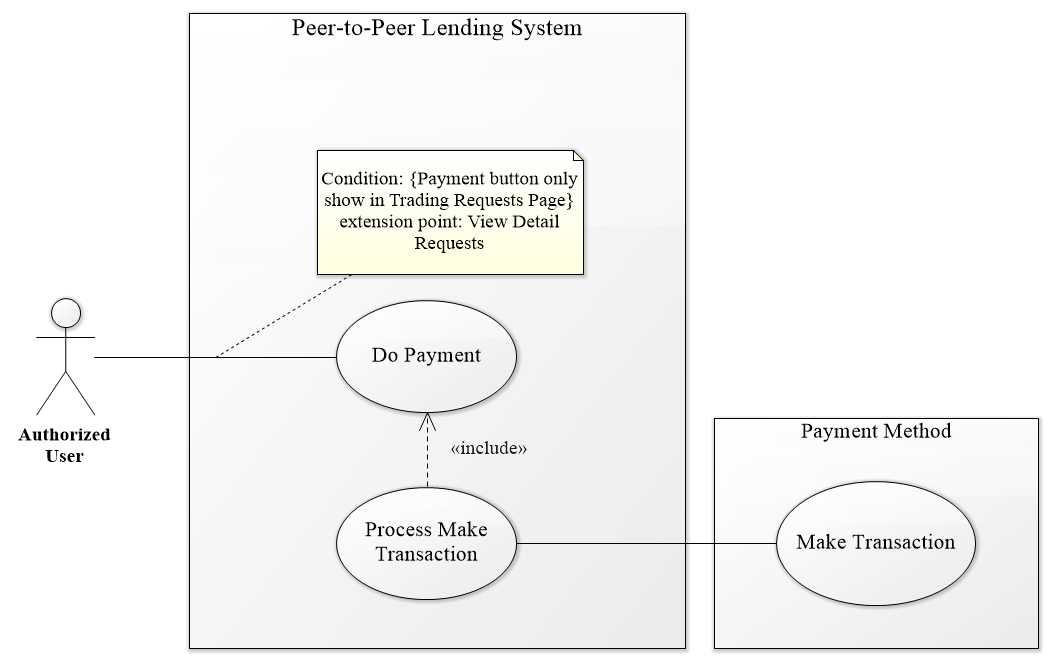
****

Figure 11 – <Authorized User> Do Payment Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_12** | | | |
| **Use Case No.** | PPLS\_AU\_12 | **Use Case Version** | 1.0 |
| **Use Case Name** | Do Payment | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to pay transaction.   **Goal:**   * Authorized User pays their transaction (lend, payback).   **Triggers:**   * Authorized User sends do payment command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User can pay their transaction. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User click on payment button (PayPal button) | System show Paypal page. | | 2 | Authorized User login to PayPal using PayPal account | Paypal show payment form | | 3 | Authorized User click on pay now button | System show success dialog |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_15, PPLS\_SY\_  **Business Rules:**   * Do payment use to accept the request and pay money for trading request. * View detail trading request, system show timeline with check timeline button to pay transaction. | | | |

Table 10 – <Authorized User> Do payment use case specification

* + - 1. **<Authorized User> Accept Request**

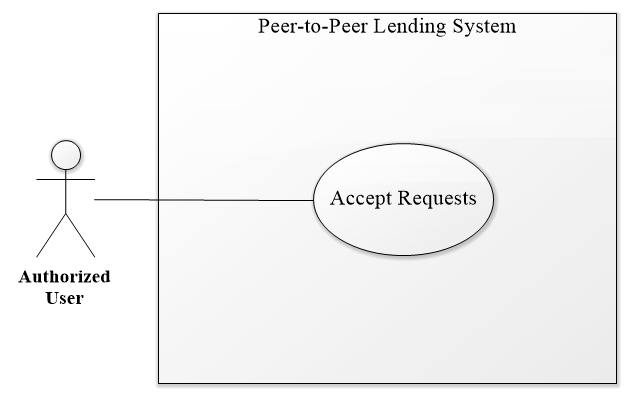
****

Figure 12 – <Authorized User> Accept Request Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_13** | | | |
| **Use Case No.** | PPLS\_AU\_13 | **Use Case Version** | 1.0 |
| **Use Case Name** | Accept request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to accept another user request.   **Goal:**   * Authorized User accepts borrow request from another user.   **Triggers:**   * Authorized User sends accept request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User accepts the request and request change to trading status. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User click on accept request | System show accepts the dialog contain paypal button. | | 2 | Authorized User do payment | System shows success dialog and redirect to view trading request page.  [Exception 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_15, PPLS\_AU\_12  **Business Rules:**   * Authorized User can accept dealing request and pending request. * Authorized User cannot accept another request. * After accept request, system save transaction and deal to blockchain and change request’s status to trading. * Accept button will show pop-up contain paypal button. | | | |

Table 11 – <Authorized User> Accept request use case specification

* + - 1. **<Authorized User> Make Deal**

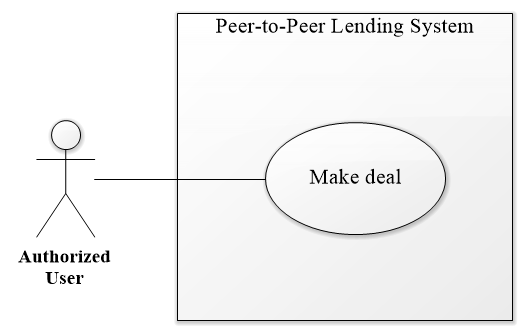
****

Figure 13 – <Authorized User> Make Deal Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_14** | | | |
| **Use Case No.** | PPLS\_AU\_14 | **Use Case Version** | 1.0 |
| **Use Case Name** | Make deal | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to edit borrow request to make a deal with another user.   **Goal:**   * Authorized User negotiates borrow request with another user.   **Triggers:**   * Authorized User sends make deal command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User make a deal with another user. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User click on make deal button | System show edit borrows request form.   * Borrow name: text * Total amount: text * Borrow amount: text * Interest rate: text * Interest Received: text * Lending timeline (Editable) * Payback timeline (Editable) | | 2 | Authorized User edit form |  | | 3 | Authorized User click on save deal button | System show Yes/No dialog | | 4 | Authorized User click on Yes button | System hide Yes/No dialog  [Alternative 1][Exception 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User click on No button | System hide Yes/No dialog |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_15, PPLS\_AU\_12  **Business Rules:**   * Authorized User can make a deal with the pending request. * Authorized User after make deal will wait for a response (another user can make a deal or accept the request). * After make deal, request’s status change to dealing. | | | |

Table 12 – <Authorized User> Make deal use case specification

* + - 1. **<Authorized User> View Detail Request**

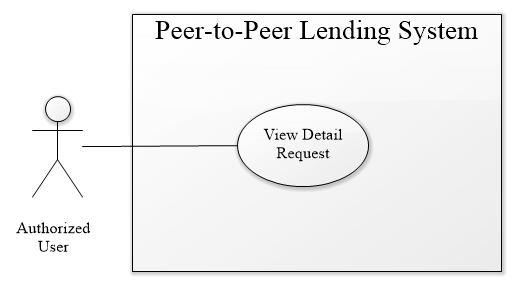
****

Figure 14 – <Authorized User> View Detail Request Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_15** | | | |
| **Use Case No.** | PPLS\_AU\_15 | **Use Case Version** | 1.0 |
| **Use Case Name** | View Detail Request | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to view detail borrow request.   **Goal:**   * Authorized User can view detail borrow request to make decision ignore, make a deal or accept this request.   **Triggers:**   * Authorized User sends view detail request command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User view detail borrows request. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User click on the view detail button | System shows borrow request page.   * Borrow name: text * Total amount: text * Borrow amount: text * Interest rate: text * Interest Received: text * Lending timeline * Payback timeline * Make deal (button) * Accept (button)   [Exception 1] [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized user reload view detail page | System will redirect to previous view page. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | The system redirects to login when the token timeout |   **Relationships:** PPLS\_AU\_06, PPLS\_AU\_09, PPLS\_AU\_10, PPLS\_AU\_11, PPLS\_AU\_12, PPLS\_AU\_13, PPLS\_AU\_14  **Business Rules:**   * View detail page with hide/show make deal and accept button depend on view page before you go to view detail page. * Pending, trading and history request’s status will hide make deal and accept button (view detail only). * Dealing request’s status and lending requests page will show make deal and accept button * View detail page allow to edit lending timeline and payback timeline when user clicks on make deal button * Accept button will show pop-up contain paypal button. | | | |

Table 13 – <Authorized User> View detail request use case specification

* + - 1. **<Authorized User> Valid Transaction with Blockchain**

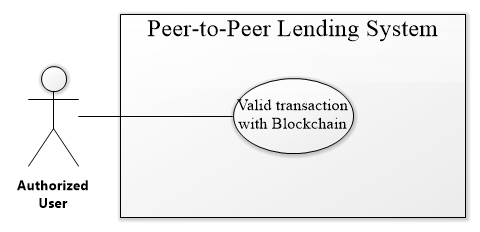
****

Figure 15 – <Authorized User> Valid Transaction With Blockchain Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_17** | | | |
| **Use Case No.** | PPLS\_AU\_17 | **Use Case Version** | 1.0 |
| **Use Case Name** | The valid transaction with Blockchain | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to valid their transaction with blockchain.   **Goal:**   * Authorized User can check valid their transaction with a transaction that saved in the blockchain.   **Triggers:**   * Authorized User sends a valid transaction with blockchain command.   **Preconditions:**   * Mustlog in as Authorized User   **Post conditions:**   * Success: Authorized User check validates of the transaction. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User click on validate. | System show check validate modal:   * Check Sender * Check Receiver * Check Amount * Check Create Date   After check this information system show:   * ID Transaction * Status (valid transaction)   [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User click on validate. | System show check validate modal:   * Check Sender * Check Receiver * Check Amount * Check Create Date   After check this information system show:   * ID Transaction * Status (invalid transaction) |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Relationships:** PPLS\_AU\_16  **Business Rules:**   * System will get transaction from database compare to transaction in blockchain. | | | |

Table 14 – <Authorized User> Valid transaction with blockchain use case specification

* + - 1. **<Authorized User> Upload Document**

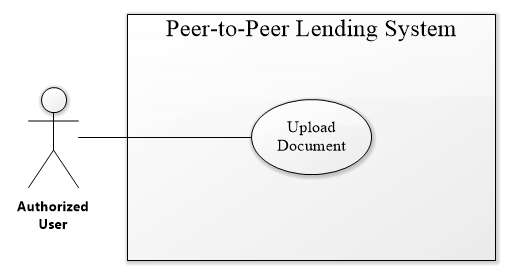
****

Figure 16 – <Authorized User> Upload Document Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AU\_18** | | | |
| **Use Case No.** | PPLS\_AU\_18 | **Use Case Version** | 1.0 |
| **Use Case Name** | Upload Document | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Authorized User.   **Summary:**   * Allow authorized user to upload the document that can identify the user.   **Goal:**   * Authorized User can upload a document such as identity card, driving license, passport, and identify the video. The document will be checked by the system administrator to verify the user.   **Triggers:**   * Authorized User sends upload document command.   **Preconditions:**   * Must log in as Authorized User   **Post conditions:**   * Success: Authorized User upload document. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Authorized User clicks on My Profile. | System show user information, document collapse, and two-button:   * Phone Number * Email address * First name * Last name   Current document collapse   * Identify Card * Password * Driving License * Identify Video   Button   * Edit Profile * Change Password | | 2 | Authorized User click on Identify Card collapse | System show identifies card collapse:   * Input file * Save button   [Alternative 4] | | 3 | Authorized User click on the Input file | System show select files input  [Exception 1] [Exception 2] | | 4 | Authorized User click on Save Button | System show success message: “save success.”  [Exception 3] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User click on Passport collapse | System show passport collapse:   * Input file * Save button   [Alternative 4] | | 2 | Authorized User click on Driving License collapse | System show driving license collapse:   * Input file * Save button   [Alternative 4] | | 3 | Authorized User click on Identify video collapse | System show identifies video collapse:   * Start camera (button) * Start Recording (button) * Play (button) * Save video (button)   [Alternative 5]  Step 1: Authorized User click on Start Camera button:   * Browser show media accept   Step 2: Authorized User click on Start Recording button   * System record video   Step 3: Authorized User click on the Play button   * System show video that is recorded   Step 4: Authorized User click on Save video button   * System show success message: “save success.”   [Exception 3] | | 4 |  | System show uploaded an image if the user had already loaded image | | 5 |  | System show message: “You had already upload video, system will keep your Identity Video due to User Privacy.” |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Authorized User click on Save button. | System show error message: “please select image to load.” | | 2 | Authorized User select another file type | System show error message: “please select image files.” | | 3 |  | The system redirects to the login page when the token timeout |   **Relationships:** PPLS\_AU\_02  **Business Rules:**   * When authorized user open identify video collapse, the system activates start camera button and deactivate start recording button, play button and save video * When authorized user to accept using media, the system activates start recording button and deactivate play button and save the video. * When authorized user clicks to save video button, system shows success message and show the message “You had already upload video, waiting for validating from Admin” instead of system show tool to record video. * System will show image after user load document except video. * Authorized user can upload document again if admin reject document. * Authorized user can view document if admin accept document. * System will show status message depend on document’s status. Ex: pending status: “Document is waiting for validation”, valid status show image only, invalid status show upload button to upload again. | | | |

Table 15 – <Authorized User> Upload document use case specification

* + 1. **<Admin> Overview Use Case**

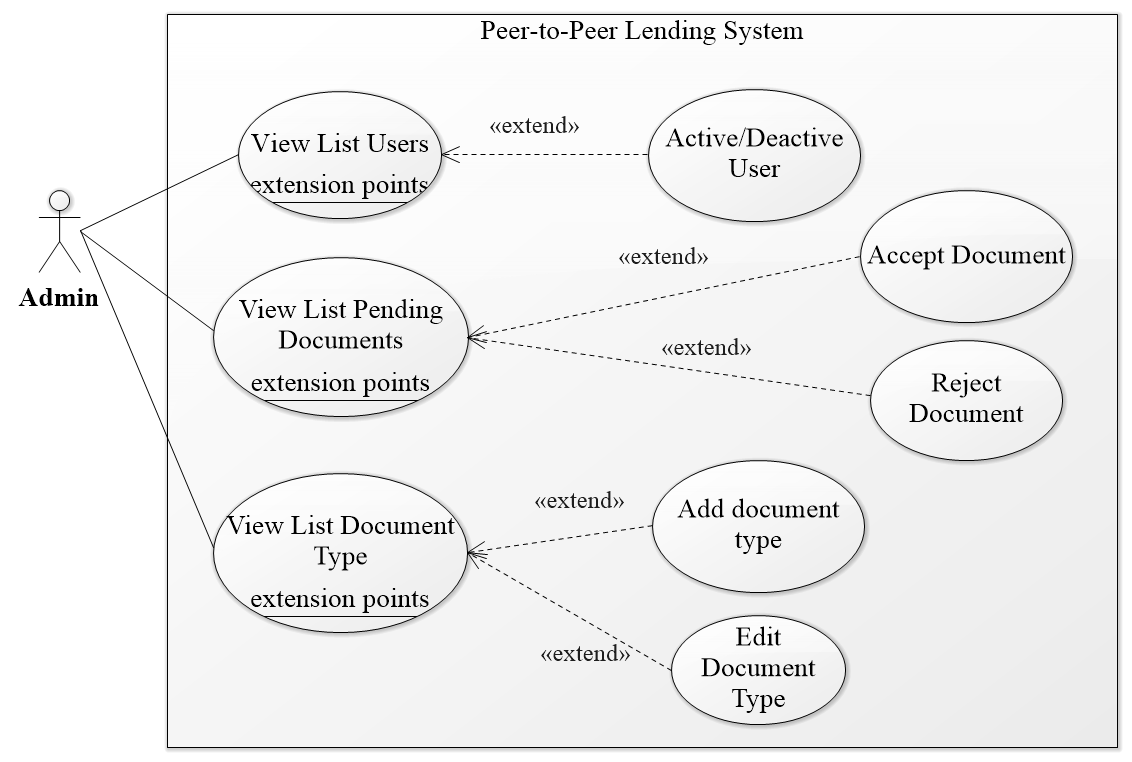
****

Figure 17 – <Admin> Overview Use Case

* + - 1. **<Admin> Accept Document**

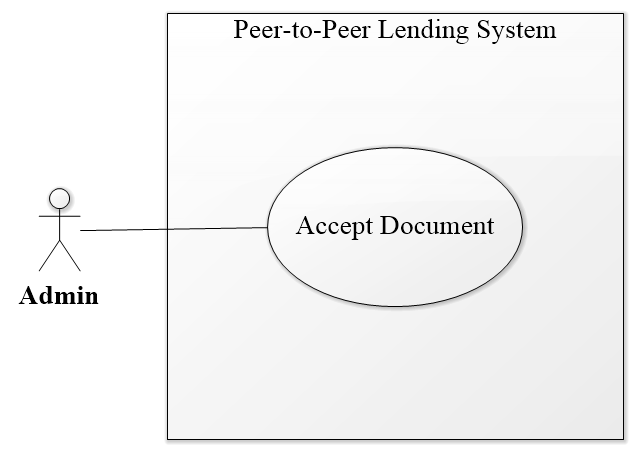
****

Figure 18 – <Admin> Accept Document Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AD\_04** | | | |
| **Use Case No.** | PPLS\_AD\_04 | **Use Case Version** | 1.0 |
| **Use Case Name** | Accept Document | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Admin.   **Summary:**   * Allow admin to accept a document from the user.   **Goal:**   * Admin can accept the document.   **Triggers:**   * Admin sends accept document command.   **Preconditions:**   * Must log in as Admin   **Post conditions:**   * Success: Admin can accept the document. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin click on Identity Card collapse | System show identifies card collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button)   [Alternative 2] [Alternative 3] [Alternative 4] | | 2 | Admin enter id document |  | | 3 | Admin click on approve button | System show confirms approve modal:   * Approve **Identity Card** of **dungnt** ? (dungnt is username)   [Exception 2] | | 4 | Admin click on Yes button | The system will accept and send a notification to user  [Exception 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin click on No button | System hide confirm approve modal | | 2 | Admin click on Passport collapse | System show passport collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button) | | 3 | Admin click on Driving License collapse | System show driving license collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button) | | 4 | Admin click on Identity Video collapse | System show identifies video collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button) |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout | | 2 | Admin click on approve button but not enter id document | System show message: “Field is required” |   **Relationships:** PPLS\_AD\_03  **Business Rules:**   * System show confirm modal when click on approve button. * ID document is an id of each document. Ex: document id of identity card is identity card number, document id of passport is passport number and so on. (document id of video is username\_video). | | | |

Table 16 – <Admin> Accept document use case specification

* + - 1. **<Admin> Reject Document**

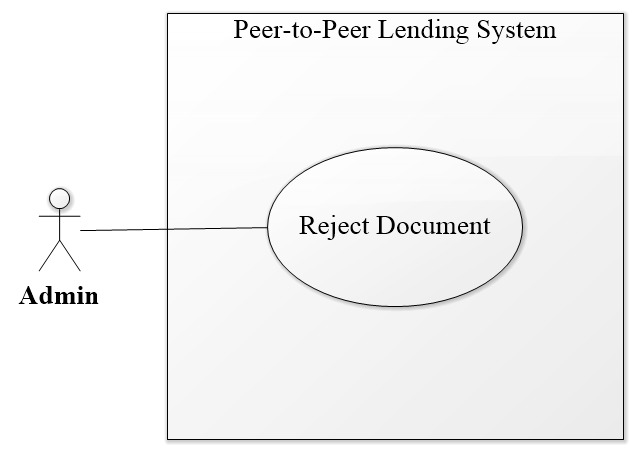
****

Figure 19– <Admin> Reject Document Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_AD\_05** | | | |
| **Use Case No.** | PPLS\_AD\_05 | **Use Case Version** | 1.0 |
| **Use Case Name** | Reject Document | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * Admin.   **Summary:**   * Allow admin to reject document from the user.   **Goal:**   * Admin can reject document.   **Triggers:**   * Admin sends recject document command.   **Preconditions:**   * Must logged in as Admin   **Post conditions:**   * Success: Admin can reject document. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Admin click on Identify Card collapse | System show identifies card collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button)   [Alternative 2] [Alternative 3] [Alternative 4] | | 2 | Admin enter id document |  | | 3 | Admin click on reject button | System show confirms reject modal:   * Reject **Identity Card** of **dungnt** ? (dungnt is username)   [Exception 2] | | 4 | Admin click on Yes button | The system will reject and send a notification to user  [Exception 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Admin click on No button | System hide confirm reject modal | | 2 | Admin click on Passport collapse | System show passport collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button) | | 3 | Admin click on Driving License collapse | System show driving license collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button) | | 4 | Admin click on Identify Video collapse | System show identifies video collapse:   * Document Image * Valid ID document (Input) * Approve (button) * Reject (button) |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 |  | System redirects to login when the token timeout |   **Relationships:** PPLS\_AD\_03  **Business Rules:**   * System show confirm modal when to click on reject button. | | | |

Table 17 – <Admin> Reject document use case specification

* + 1. **<System> Overview use case**

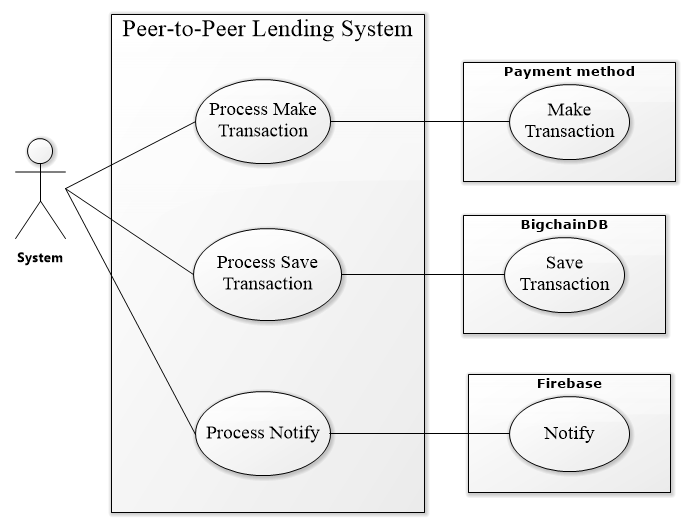
****

Figure 20 – <System> Overview Use Case

* + - 1. **<System> Process Make Transaction**

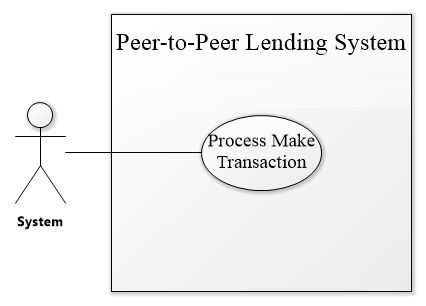
****

Figure 21 – <System> Process Make Transaction Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_SY\_01** | | | |
| **Use Case No.** | PPLS\_SY\_01 | **Use Case Version** | 1.0 |
| **Use Case Name** | Process make transaction | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * System.   **Summary:**   * Allow the system to process make a transaction.   **Goal:**   * The system can process make a transaction.   **Triggers:**   * The system sends process make transaction command.   **Preconditions:**   * N/A   **Post conditions:**   * Success: System can process make transaction. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 |  | System gets the amount and call PayPal to execute the payment. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Relationships:** N/A  **Business Rules:**   * Before the PayPal call, the system converts currency from Vietnam dong to US dollar | | | |

Table 18 – <System> Process make transaction use case specification

* + - 1. **<System> Process Save Transaction**

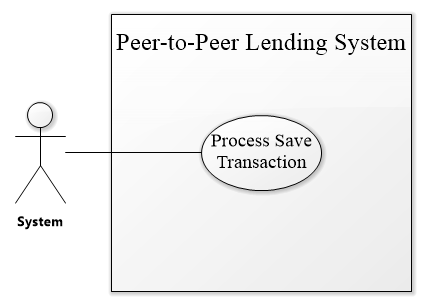
****

Figure 22 – <System> Process Save Transaction Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – PPLS\_SY\_01** | | | |
| **Use Case No.** | PPLS\_SY\_01 | **Use Case Version** | 1.0 |
| **Use Case Name** | Process save transaction | | |
| **Author** | LocHV | | |
| **Date** | 22/05/2019 | **Priority** | Normal |
| **Actor:**   * System.   **Summary:**   * Allow the system to process save the transaction.   **Goal:**   * System can process save transaction.   **Triggers:**   * System sends process save transaction command.   **Preconditions:**   * N/A   **Post conditions:**   * Success: System can process save transaction. * Fail: System shows error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 |  | After PayPal return success, system gets information of user, PayPal and call bigchainDB to save the transaction and deal. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Exceptions:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | |  |  |  |   **Relationships:** N/A  **Business Rules:**   * Information save to bigchainDB include:   + transaction id   + amount   + lender username   + borrower username   + create date   + deal data     - deal id     - deal amount     - interest rate     - borrower username     - lender username     - milestone       * id       * previous date       * present date       * type       * percent       * transaction         + transaction id         + amount         + amount valid         + create date         + transaction id         + lender username         + borrower username         + status         + milestone id | | | |

Table 19 – <System> Process save transaction use case specification

|  |  |
| --- | --- |
| **Entity Data dictionary: describe all content of all entities** | |
| **Entity Name** | **Description** |
| Authorized User | Abstract entity describes a user in the system include:   * ID * Username * FirstName * LastName * Password * Role * Status |
| Lender | Contain the lender’s information (Authorized user information) |
| Borrower | Contain the borrower’s information (Authorized user information) |
| Admin | Contain the admin’s information (Authorized user information) |
| Request | Contains the request’s information include:   * ID * LenderID * BorrowerID * Amount * InterestRate * CreateDate * BorrowDate |
| Deal | Contains the deal’s information include:   * ID * Status * BorrowTimes * PaybackTimes |
| Backup Deal | Contains the backup deal’s information (Deal information). |
| Milestone | Contains the milestone’s information include:   * ID * PreviousDate * PresentDate * Type * Percent |
| Backup Milestone | Contains the backup milestone’s information (Milestone information). |
| Transaction | Contains the transaction’s information include:   * ID * Sender * Receiver * Amount * AmountValid * IdTrx (ID Transaction) * Status * CreateDate |
| Document | Contains the document’s information include:   * ID * DocumentID * Status |
| Document Type | Contains the document type’s information include:   * ID * Name * AmountLimit * Acronym |
| Document File | Contains the document file’s information include:   * ID * FileName * FileType * Data |

Table 20 – Conceptual Diagram Data Dictionary

# System Architectural Diagram

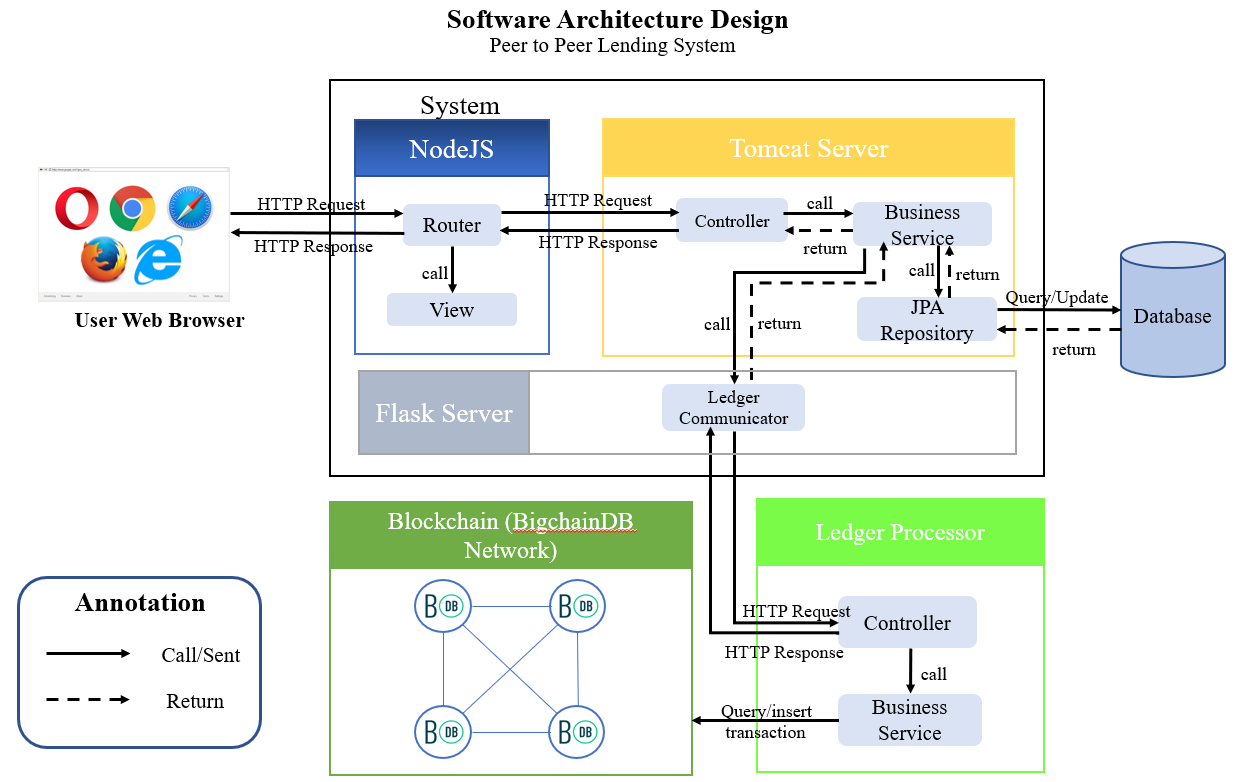


Figure 23 – Software Architecture Design

In Web Application, the system is developed under the MVC architecture style using Spring Framework. We choose this architecture for Web Application because of following advantages:

* The Spring Framework features its own MVC web application framework, defines interfaces for all of the responsibilities that must be handled by a modern request-based framework (such as Controller, View, HandlerAdapter,...)
* Spring Boot provided by Spring framework makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run."
* Spring Security offers a powerful and convenient process for authentication.
* Its inversion of control feature creates the object in runtime and satisfies application dependencies.
* ReactJS is quite easy to be mastered.
* ReactJS is easy to maintain isolated components in React.
* Convenient React is highly competitive to MVC. One-way data flow provides maintainability and efficient arrangement of data and DOM elements.
* It is the most light-weighted framework among the ones that are widely used today.

As illustrated in Figure 40, the architecture consists of 3 layers: application layer, ledger processor layer, and BigchainDB network.

* The application layer provides the users essential functions of a content publishing platform. Besides, the digital wallet functionality, which allows the user to view transaction history and pay directly for the transaction is also supported.
* The ledger processor layer is the middleware communicator connecting the application layer with the BigchainDB network. This layer receives a query, register and transfer requests from the application to build a standard transaction before sending to the blockchain storage; transaction result is also retrieved in this layer to send back as a response to the application.
* The BigchainDB network is the underlying blockchain network recordings all transaction related to transaction payment assets.

# Component Diagram

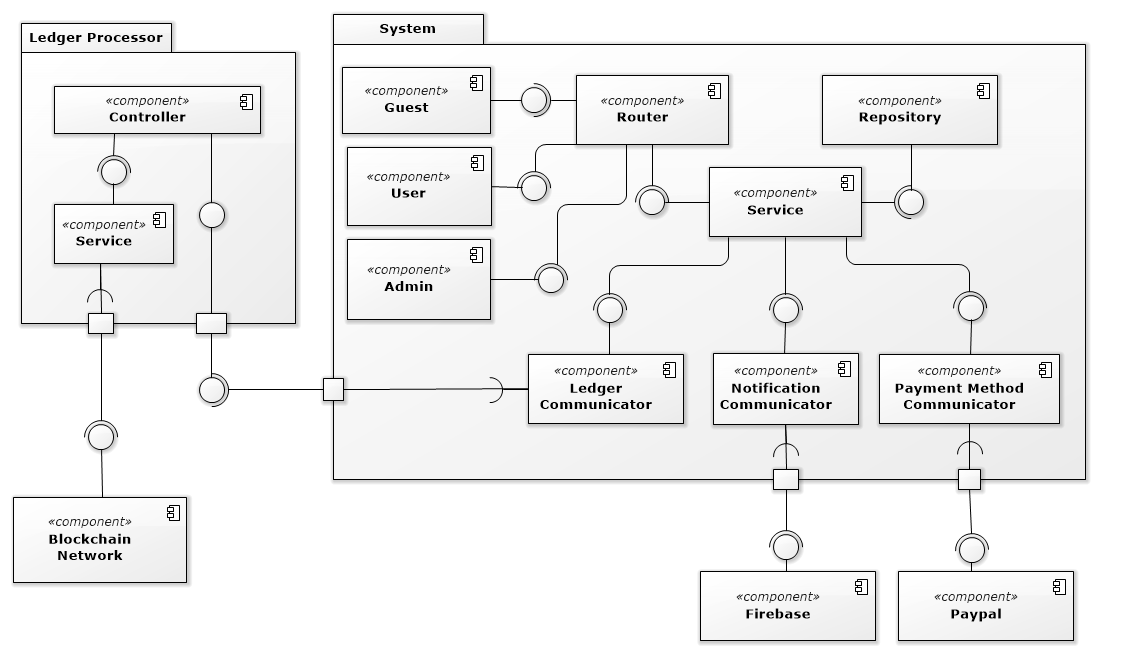


Figure 24 – Front End User Component Diagram

|  |  |
| --- | --- |
| COMPONENT DICTIONARY: DESCRIBES COMPONENTS | |
| System |  |
| Router | Handle request and response, accept input, and convert it to commands for the back end. |
| Repository Component | Component to handle the interaction between the system and database |
| Service Component | Component to handle the system’s business operations |
| Admin Component | Component to handle admin activities in the system |
| User Component | Component to handle user activities in the system |
| Guest Component | Component to handle guest activities in the system |
| Ledger communicator | Component to handle communication between system and ledger processor |
| Notification communicator | Component to handle communication notification process |
| Payment method communicator | Component to handle communication payment process |
| PayPal | Handle payment process with PayPal API |
| Firebase | Handle payment process with Firebase API |
| Ledger processor | Ledger processor to process business-related to the ledger |
| Controller | Handle HTTP request |
| Service | Handle ledger business operations |
| Blockchain network | Handle interaction between ledger processor and a blockchain database |

Table 21 – Front End Component Diagram Dictionary

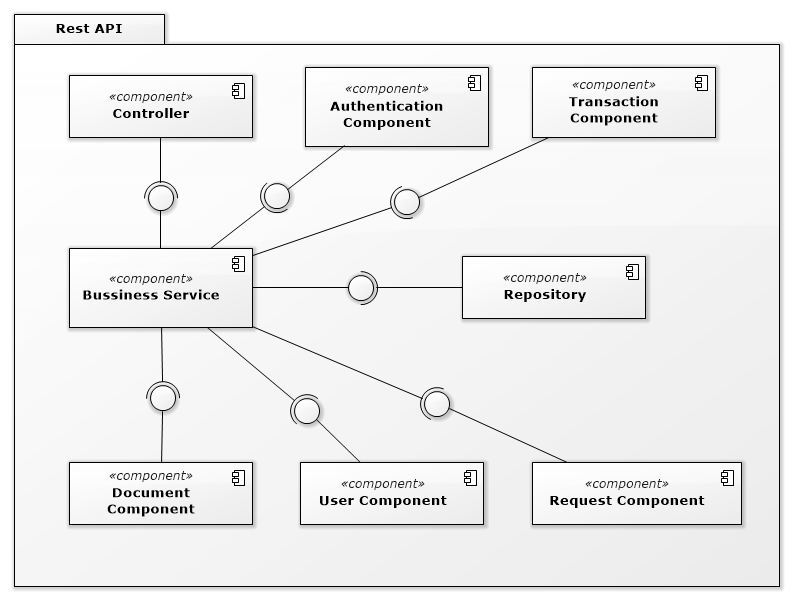


Figure 25 – Back End Component Diagram

|  |  |
| --- | --- |
| COMPONENT DICTIONARY: DESCRIBES COMPONENTS | |
| Rest API |  |
| Controller | Handle requests and responses; accept input and handle response error |
| Bussiness Servies | Handle system’s business operations |
| Repository | Store Data |

Table 22– Back End Component Diagram Dictionary

# Class Diagram

### **Class Diagram**



Figure 26 – Class Diagram

### **Class Diagram Explanation**

|  |  |  |
| --- | --- | --- |
| **CLASS DICTIONARY: DESCRIBE CLASS** | | |
| **Class Name** | **Mapping column with Conceptual diagram** | **Description** |
| **User** | AuthorizedUser | Contains the user information |
| **Request** | Request | Contains the request information |
| **Deal** | Deal | Contains the deal information |
| **Milestone** | Milestone | Contains the milestone information |
| **Transaction** | Transaction | Contains the transaction information |
| **BackupDeal** | BackupDeal | Contains the backup deal information |
| **BackupMilestone** | BackupMilestone | Contains the backup milestone information |
| **Document** | Document | Contains the document information |
| **DocumentType** | DocumentType | Contains the information of document type |
| **DocumentFile** | DocumentFile | Contains the information of document file |

Table 23 – Class Diagram PPLS

* + 1. **User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | String | Private | Unique identifier of an account |
| username | String | Private | Account’s username |
| password | String | Private | Account’s password |
| lastName | String | Private | Account’s full name |
| firstName | String | Private | Account’s address |
| loanLimit | Long | Private | Account’s position |
| role | String | Private | Account’s Role |
| email | String | Private | Account’s Email |
| phoneNumber | String | Private | Account’s Phone Number |
| status | String | Private | Account’s Status |

Table 24 – <Class Diagram Attributes> User

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| checkLogin | Object | Public | Login |
| createUser | boolean | Public | Create a new user |
| activeUser | boolean | Public | Active a user |
| deactivateUser | boolean | Public | Deactivate a user |
| getOneByUsername | Object | Public | Get a user information base on token |
| getUserMaximunLoanLimit | Long | Public | Get user maximum loan limit |
| getUsers | List | Public | Get User List |
| findUsername | object | Public | Get a user base on username |

Table 25 – <Class Diagram Methods> User

* + 1. **Request**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier of a request |
| amount | Long | Private | Amount of the request |
| borrowDate | Long | Private | Borrow date in timestamp |
| interestRate | Float | Private | Request interest rate |
| createDate | Long | Private | Create a date in timestamp |
| status | String | Private | Request’s status |
| borrowerId | Integer | Private | Id of borrower |
| lendId | Integer | Private | Id of Lender |

Table 26 – <Class Diagram Attributes> Request

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| createRequest | boolean | Public | Add new request |
| findAllOtherUserRequest | List | Public | Find all request not belong to current login user |
| findUserAllRequestByStatus | List | Public | Find all borrower request by status |
| findAllRequestByStatusWithLenderOrBorrower | List | Public | Find all borrower and lender request by status |
| findAllRequestByStatusWithLender | List | Public | Find all lender request with status |
| findAllRequestByStatusWithBorrower | List | Public | Find all borrower request by status |
| remove | boolean | Public | Delete request with pending status only |

Table 27 – <Class Diagram Methods> Request

* + 1. **Deal**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | Private | Unique identifier |
| status | String | Private | Deal’s status |
| borrowTime | String | Private | Number of the borrowed time on request |
| paybackTime | Datetime | Private | Number of the lending time on request |
| userId | Integer | Private | Unique identifier of user |
| requestId | Integer | Private | Unique identifier of request |

Table 28 – <Class Diagram Attributes> Deal

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| makeDeal | boolean | Public | Allow user to make a deal to request a loan |
| acceptDeal | boolean | Public | Accept deal that other people make to the request |
| cancelDeal | boolean | Public | Cancel deal and reset deal to the original deal |

Table 29 – <Class Diagram Methods> Deal

* + 1. **Milestone**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier |
| previousDate | Long | Private | Date of the previous milestone |
| presentDate | Long | Private | Date of current milestone |
| type | String | Private | Milestone’s type |
| percent | Float | Private | Milestone’s percent |
| dealId | Integer | Private | Unique identifier of deal |

Table 30 – <Class Diagram Attributes> Milestone

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| newMilestone | boolean | Public | Add new milestone |
| updateMilestone | boolean | Public | Edit milestone |

Table 31 – <Class Diagram Methods> Milestone

* + 1. **Transaction**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | Private | Unique identifier |
| sender | String | Private | Sitemap’s map |
| receiver | String | Private | Url root |
| idTrx | Integer | Private | Type map |
| amountValid | Double | Private | Amount of transaction to check valid |
| amount | Double | Private | Amount of money send |
| status | String | Private | Status of the transaction |
| createDate | Long | Private | Transaction create date |
| milestoneId | Integer | Private | Unique identifier of milestone |

Table 32 – <Class Diagram Attributes> Transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| getTopTransactionOrderByCreateDateDesc | List | Public | Get top 20 transaction currently make of all player |
| getAllUserTransaction | List | Public | Get all transaction of a user |
| newTransaction | boolean | Public | Make a transaction |

Table 33 – <Class Diagram Methods> Transaction

* + 1. **Backup Deal**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier |
| status | String | Private | Backup deal status |
| borrowTime | Integer | Private | Backup deal borrowTime |
| paybackTime | Integer | Private | Backup deal paybackTime |
| dealId | Integer | Private | Unique identifierof deal |

Table 34 – <Class Diagram Attributes> Backup Deal

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| addBackupDeal | Boolean | Public | Add new backup deal |
| removeBackupDeal | Boolean | Public | Remove backup deal |

Table 35 – <Class Diagram Methods> Backup Deal

* + 1. **Backup Milestone**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | Private | Unique identifier of a backup milestone |
| previousDate | Long | Private | Backup milestone previous date |
| presentDate | Long | Private | Backup milestone present date |
| type | String | Private | Backup milestone type |
| percent | Float | Private | Backup milestone percent |
| backupDealId | Integer | Private | Unique identifier of a backup deal |

Table 36 – <Class Diagram Attributes> Backup milestone

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| addBackupMilestone | boolean | Public | Add new backup milestone |
| removeBackupMilestone | Boolean | Public | Remove backup milestone |

Table 37 – <Class Diagram Methods> Backup milestone

* + 1. **Document**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | Private | Unique identifier |
| documentId | Integer | Private | Id of a document |
| status | String | Private | Status of document to know it valid or not |
| documentTypeId | Integer | Private | Unique identifier of document type |
| userId | Integer | Private | Unique identifier of user |

Table 38 – <Class Diagram Attributes> Document

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| uploadDocument | boolean | Public | Upload user document for admin to valid |
| uploadVideo | boolean | Public | Upload user video for admin to valid |
| getUserDocument | List | Public | Get all current login user document |
| getAllUserDocument | List | Public | Get all document of a user |
| validDocumentId | boolean | Public | Decide if a document is valid |
| getAllPendingDocument | List | Public | Get all pending document |
| invalidDocumentId | boolean | Public | Decide if a document is invalid |

Table 39 – <Class Diagram Methods> Document

* + 1. **Document Type**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Integer | Private | Unique identifier |
| name | String | Private | Name of document type |
| amountLimit | Long | Private | Amount gain of this document type |

Table 40 – <Class Diagram Attributes> Document Type

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Redirect Type** | **Visibility** | **Description** |
| listDocumentType | boolean | Public | Upload user document for admin to valid |
| newDocumentType | boolean | Public | Upload user video for admin to valid |
| updateDocumentType | List | Public | Get all current login user document |

* + 1. **DocumentFile**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| id | Long | Private | Unique identifier of a grammar report |
| fileName | Long | Private | Grammar error is found at position |
| fileType | String | Private | Grammar errors excerpt |
| data | String | Private | Appereance info |
| documentId | String | Private | Grammar error reason |

Table 41 – <Class Diagram Attributes> DocumentFile

# Entity Relationship Diagram (ERD)

## Entity relationship diagram (ERD)

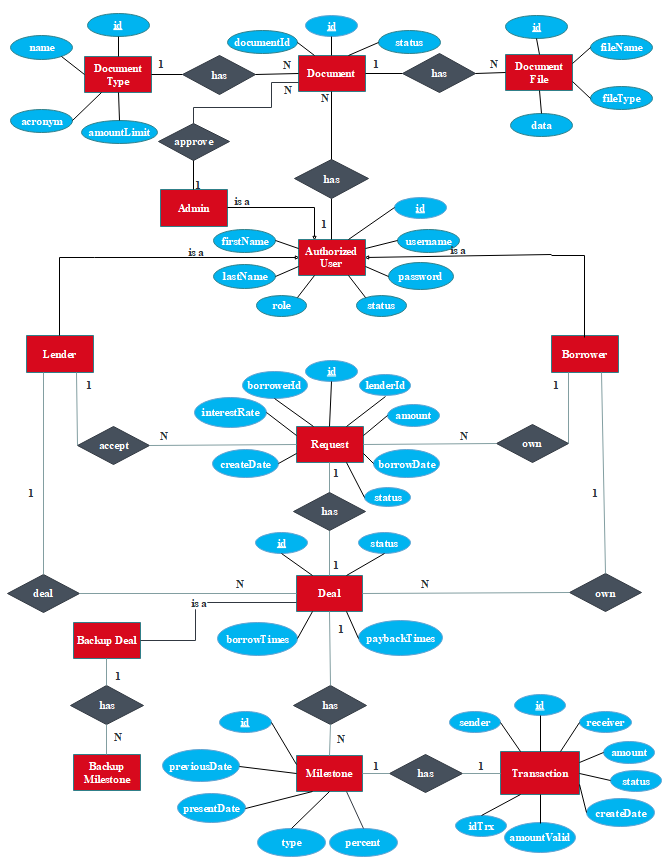


Figure 27 Entity Relationship Diagram

## Entity Data Dictionary

|  |  |
| --- | --- |
| Entity Data Dictionary: describe content of all entities | |
| Entity name | Description |
| Authorized User | Contains the authorized user’s information include:   * ID * Username * FirstName * LastName * Password * Role * Status |
| Admin | Contains the admin’s information (Authorized user information). |
| Lender | Contains the lender’s information (Authorized user information). |
| Borrower | Contains the borrower’s information |
| Request | Contains the request’s information include:   * ID * LenderID * BorrowerID * Amount * InterestRate * CreateDate * BorrowDate |
| Deal | Contains the deal’s information include:   * ID * Status * BorrowTimes * PaybackTimes |
| Backup Deal | Contains the backup deal’s information (Deal information). |
| Milestone | Contains the milestone’s information include:   * ID * PreviousDate * PresentDate * Type * Percent |
| Backup Milestone | Contains the backup milestone’s information (Milestone information). |
| Transaction | Contains the transaction’s information include:   * ID * Sender * Receiver * Amount * AmountValid * IdTrx (ID Transaction) * Status * CreateDate |
| Document | Contains the document’s information include:   * ID * DocumentID * Status |
| Document Type | Contains the document type’s information include:   * ID * Name * AmountLimit * Acronym |
| Document File | Contains the document file’s information include:   * ID * FileName * FileType * Data |

Table 42 Entity Relationship Diagram Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity Name | Attributes | Description | Domain | Null |
| user | id | Id of user’s account | INT(11) | NO |
| username | User name of user’s account | VARCHAR(255 | NO |
| password | Password of user’s account | VARCHAR(255) | NO |
| firstName | First name of user’s account | VARCHAR(255) | NO |
| lastName | Last name of user’s account | VARCHAR(255) | NO |
| role | Role of user’s account | VARCHAR(255) | NO |
| loanLimit | Loan limit of user’s account | BIGINT(20) | NO |
| status | Status of user’s account | VARCHAR(255) | NO |
| phoneNumber | Phone number of user | VARCHAR(255) | NO |
| email | Email of user | VARCHAR(255) | NO |
| request | id | Id of request | INT(11) | NO |
| amount | Role name of account | VARCHAR(255) | NO |
| borrowDate | Borrow date of a request | BIGINT(20) | YES |
| interestRate | Request's interestd rate | BIGINT(20) | NO |
| createDate | Date when request created | BIGINT(20) | NO |
| status | Status's request | VARCHAR(255) | NO |
| deal | id | Id of deal | INT(11) | NO |
| status | Deal’s status | VARCHAR(255) | NO |
| borrowTime | Number of borrow time | VARCHAR(255) | NO |
| paybackTime | Number of payback time | VARCHAR(255) | NO |
| milestone | id | Id of milestone | INT(11) | NO |
| previousDate | Date of previous milestone | BIGINT(20) | NO |
| presentDate | Date of present milestone | BIGINT(20) | NO |
| type | Milestone’s type | VARCHAR(255) | NO |
| percent | Milestone's percent | FLOAT | NO |
| transaction | id | Id of transaction | INT(11) | NO |
| sender | Sender’s username | BIGINT(20) | NO |
| receiver | Receiver user’s name | VARCHAR(255) | NO |
| idTrx | Id of transaction on blockchain | VARCHAR(255) | NO |
| amountValid | To valid the money make | DOUBLE | NO |
| amount | Amount of money send | DOUBLE | NO |
| status | Status of the transaction | VARCHAR(255) | NO |
| createDate | Transaction's create date | BIGINT(20) | NO |
| backupDeal | id | Id of backup deal | INT(11) | NO |
| status | Status of backup deal | VARCHAR(255) | NO |
| borrowTime | Backup borrow times | INT(11) | NO |
| paybackTime | Backup payback times | INT(11) | NO |
| backupMilestone | id | Id of backup milestone | INT(11) | NO |
| presentDate | Backup present date of milestone | BIGINT(20) | NO |
| previousDate | Backup previous date of milestone | BIGINT(20) | NO |
| type | Milestone type’s backup | VARCHAR(255) | NO |
| percent | Milestone’s percent backup | FLOAT | NO |
| document | id | Id of document | INT(11) | NO |
| documentId | Id on the document user upload | VARCHAR(255) | NO |
| status | Status of the document | VARCHAR(255) | NO |
| documentFile | id | Id of document file | INT(11) | NO |
| fileName | Name of the file | VARCHAR(255) | NO |
| fileType | Type of the file | VARCHAR(255) | NO |
| data | Date of file to store | LONGBLOB | NO |
| documentType | id | If of the document type | INT(11) | NO |
| name | Name of document type | VARCHAR(255) | NO |
| amountLimit | Amount to gain when complete document | BIGINT(20) | NO |

Table 43 - ERD Data Dictionary

# Interaction Diagram (Sequence Diagram)

#### 1. Login

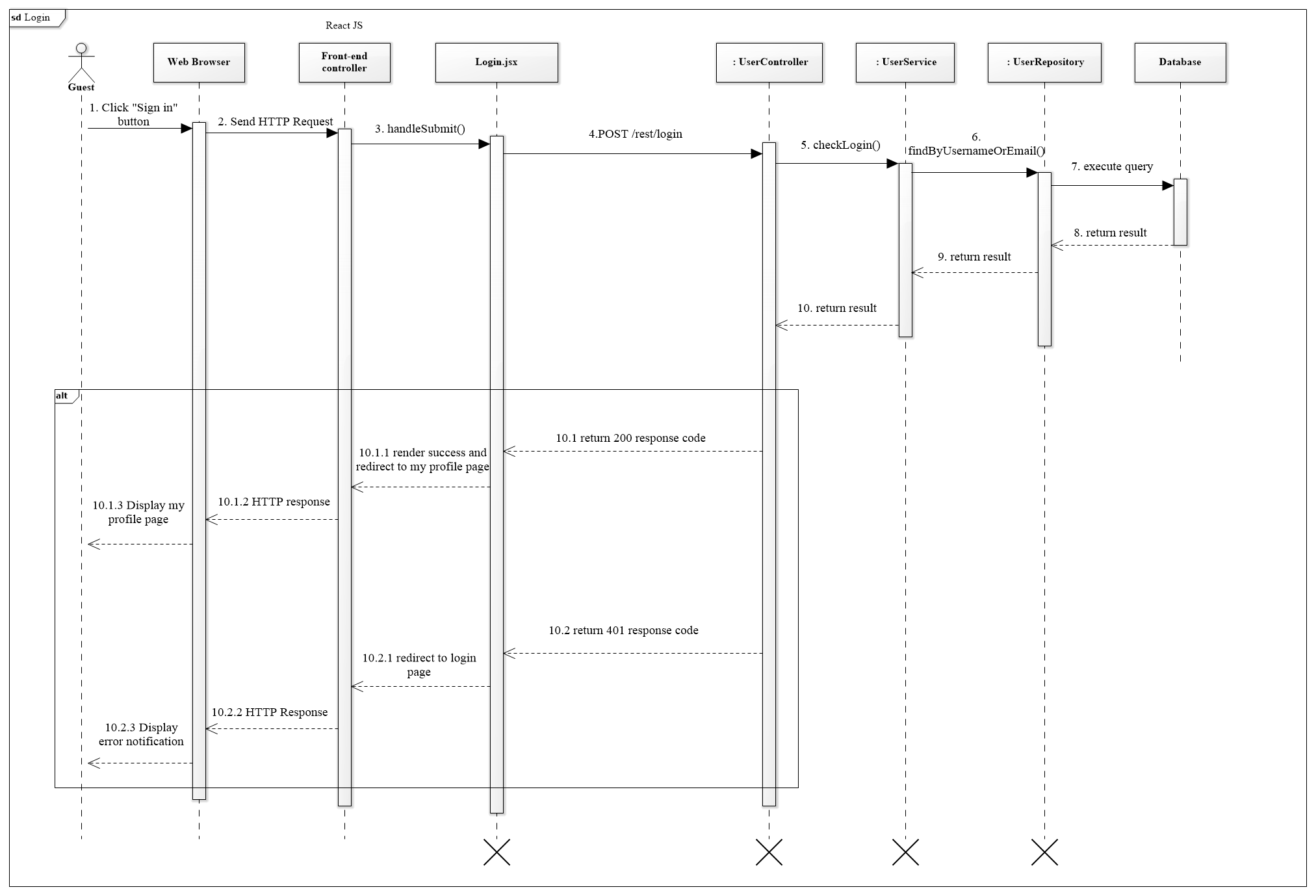
****

Figure 28 <Sequence Diagram> Guest Login

#### 1.2. Accept deal

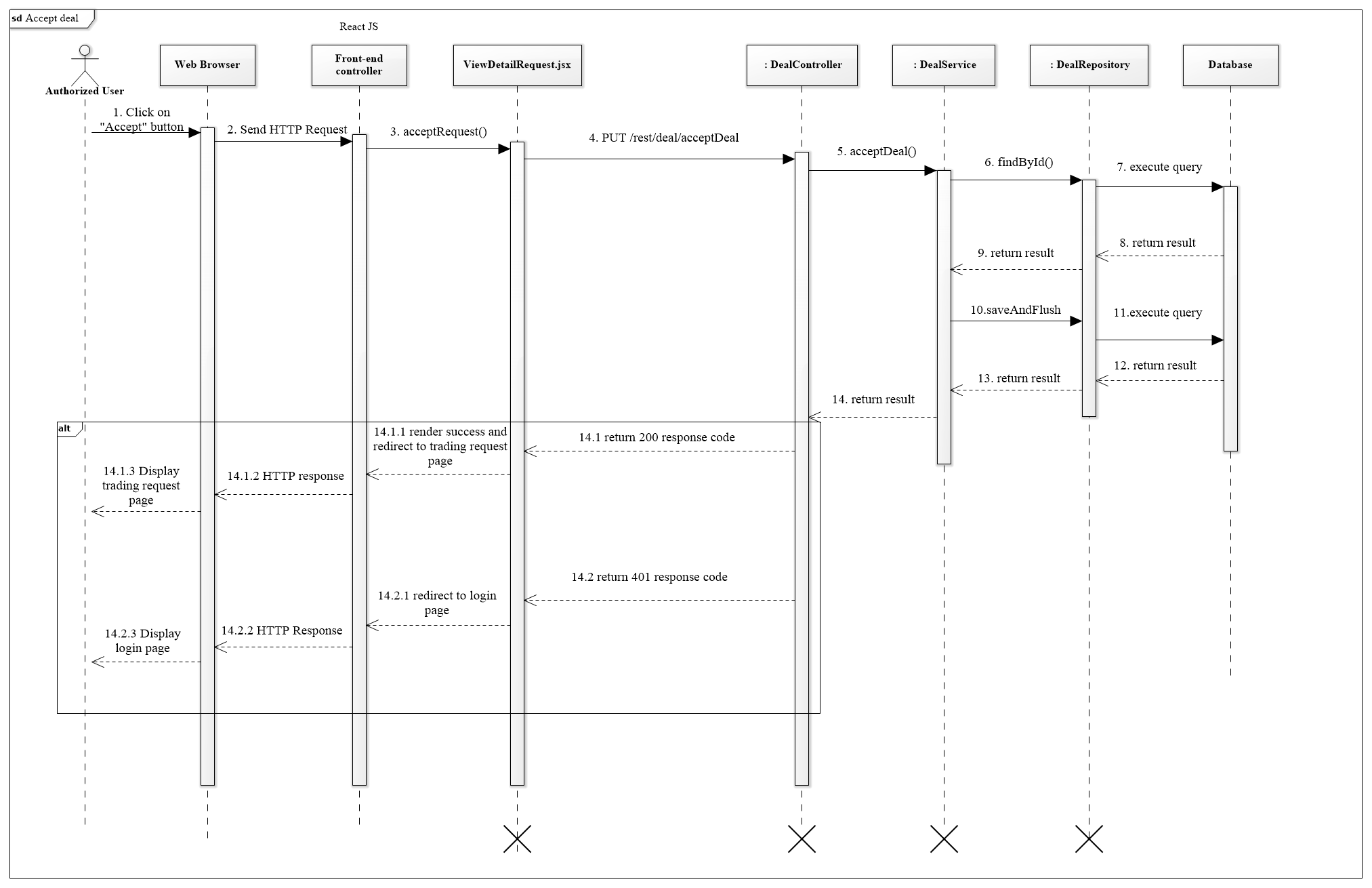


Figure 29 <Sequence Diagram> Accept Deal

#### 1.3. Create request

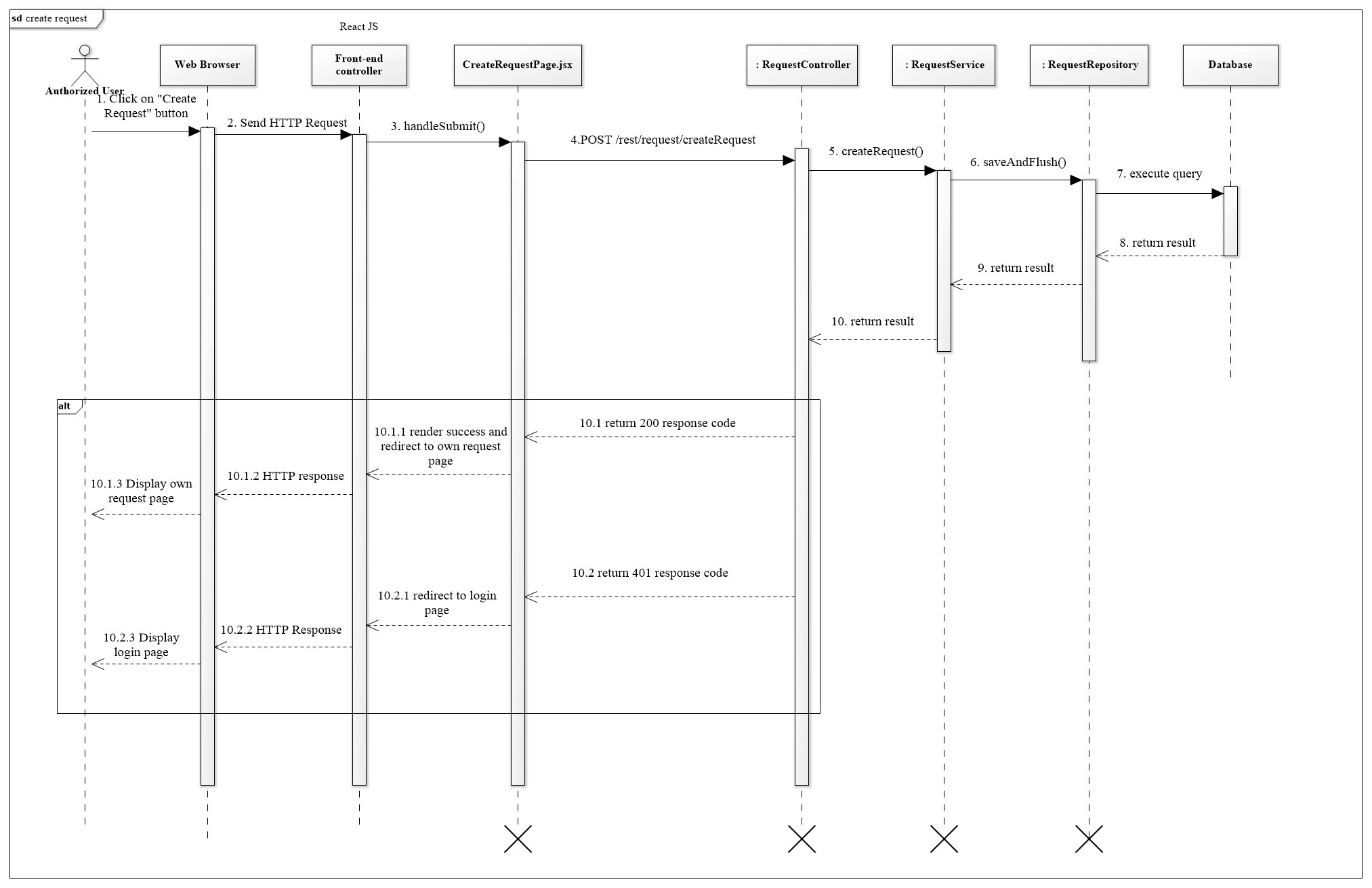


Figure 30 <Sequence Diagram> Create request

#### 1.4. Make deal

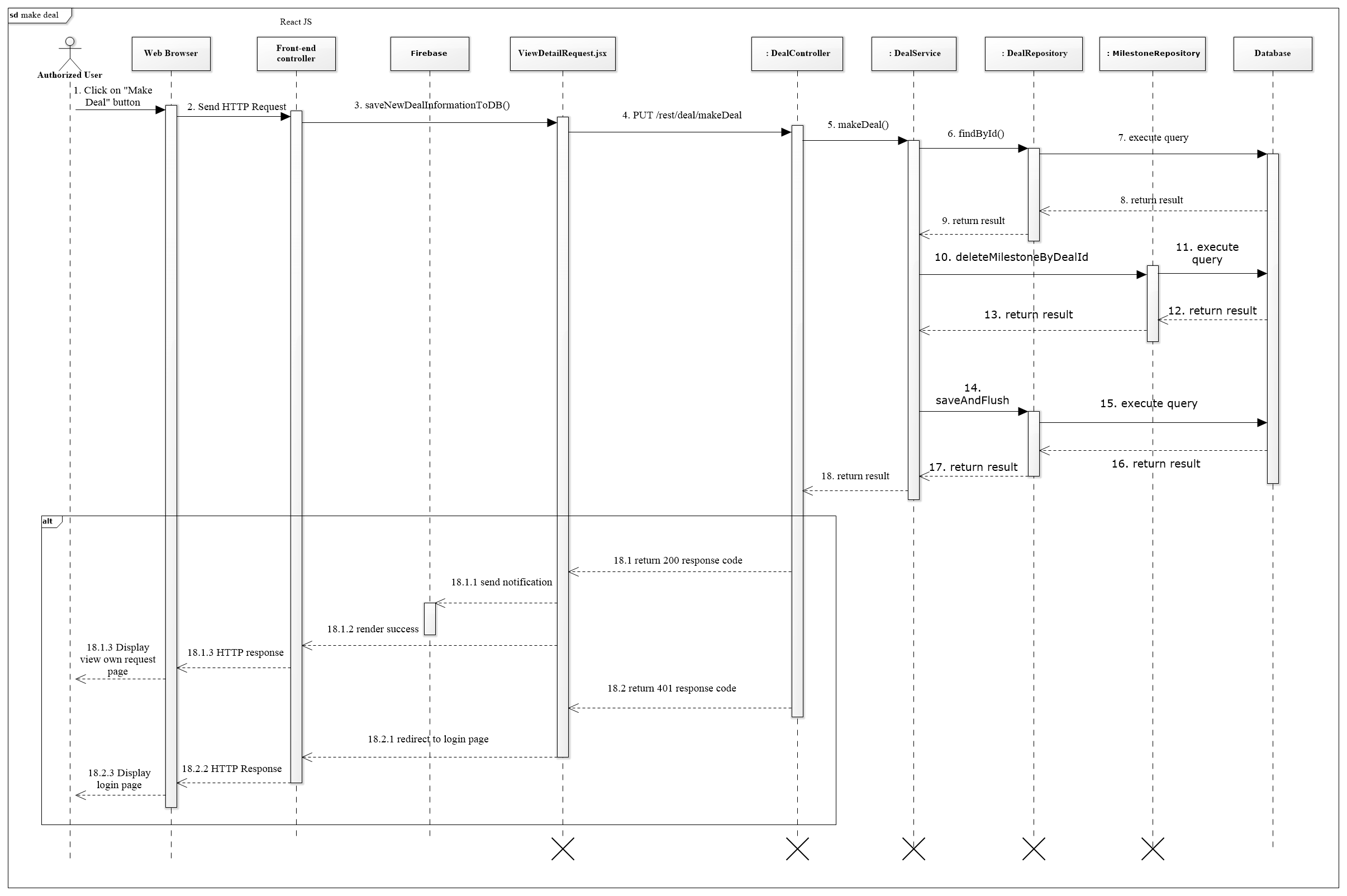


Figure 31 <Sequence Diagram> Make deal

#### 1.5. Valid document

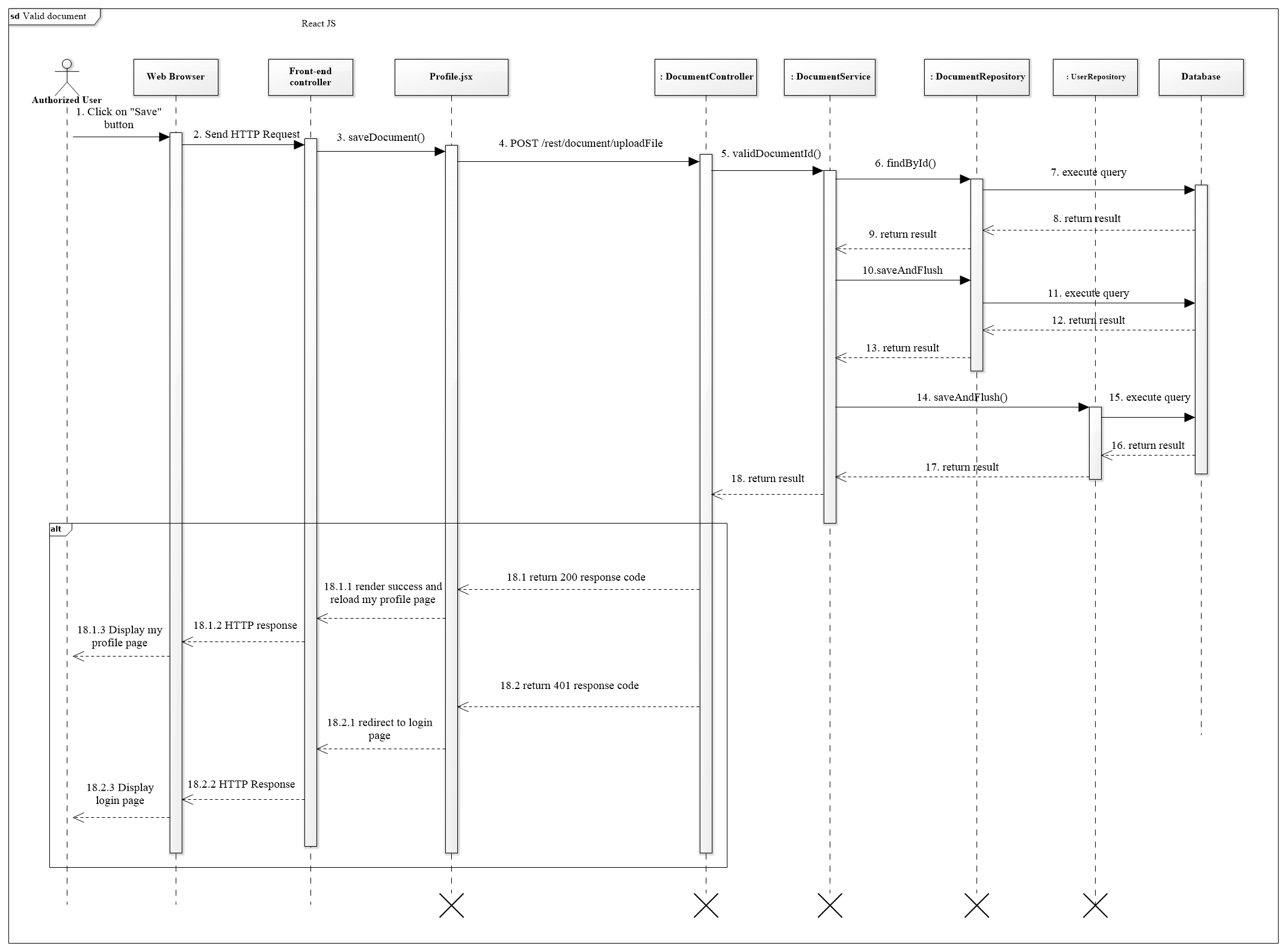


Figure 32 <Sequence Diagram> Valid document

#### 1.6. Do payment and sent to BigchainDB

Figure 33 <Sequence Diagram> Do payment and sent to BigchainDB

# Physical Diagram – Relationship Diagram

### **Physical diagram**

Figure 34 Physical Diagram

### **Data dictionary**

|  |  |  |
| --- | --- | --- |
| **Entity Data Dictionary: Describe content of all tables** | | |
| **No.** | **Table Name** | **Description** |
| 1 | user | Contain account information. |
| 2 | request | Contain request information. |
| 3 | deal | Contain deal information of a request. |
| 4 | milestone | Contain all the milestone of a deal |
| 5 | transaction | Contain all transaction made. |
| 6 | Backup deal | Contain backup version of the original first deal. |
| 7 | Backup milestone | Contain backup version of the original first milestone of the deal. |
| 8 | document | Contain all user document for validation. |
| 9 | Document file | Contain document file. |
| 10 | Document\_type | Contain type of a document and amount of money gain when completed it |

Table 44 Physical Diagram Dictionary

# Framework Architectural Diagram

1. **Blockchain Architectural Diagram - BigchainDB**

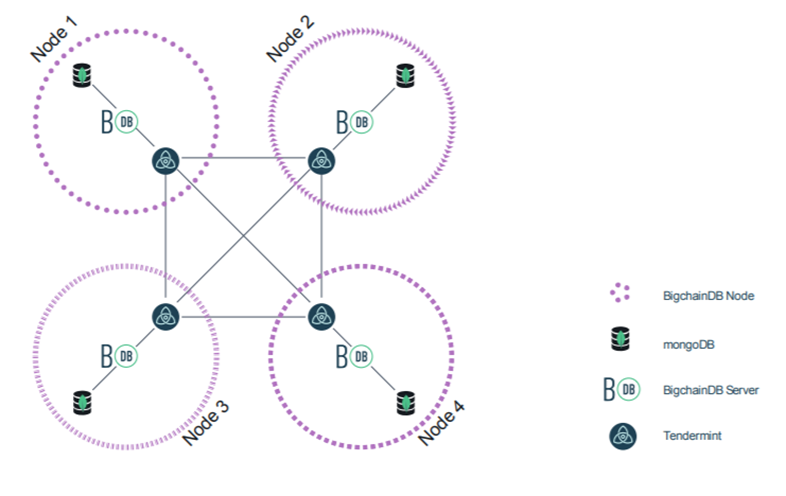
****

Figure 35 – Blockchain Architecture Diagram - BigchainDB

1. **ReactJS Architectural Diagram**

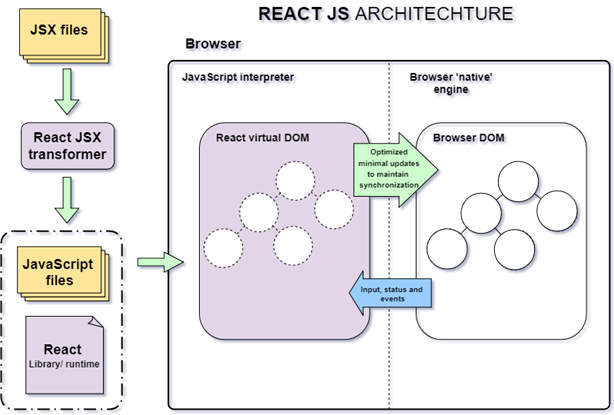
****

Figure 36 – ReactJS Architectural Diagram

1. **Spring Boot Architectural Diagram**

****

Figure 37 – Spring Boot Architecture Diagram

# Algorithms

### **1. Record Transaction**

#### 1.1. Definition

Record transaction is the mechanism to save transaction to blockchain database.

#### 1.2. Define problem

In order to provide proof of ownership, the transaction must be saved as an immutable record to blockchain database.

#### 1.3. Solution

BigchainDB, which is a scalable blockchain database designed to combine the blockchain properties and distributed database, is used in this system for fulfilling business corresponding to ledger. It offers a full decentralization and Byzantine Fault Tolerance, immutability and native assets that allows for the deployment of large-scale applications in a variety of use cases and industries including intellectual property.

The transaction is created as a BigchainDB’s transaction with the unique key and transaction containing information of transaction including txId, sender, receiver, amount,… this transaction is cryptographically signed by the one time key pair and written to the BigchainDB storage after it is verified as a valid transaction.

#### 1.4. Example



Figure 38. Example of BigchainDB transaction

### **2. Interest lending**

#### 2.1. Definition

Interest lending is a method that calculate the interest of a borrowing request.

#### 2.2. Define problem

Each transaction needs the consent of both parties, so that interest rates will make lenders' interests more clear.

#### 2.3. Solution

We use the formula below to calculate the article rank score:

a : amount of the request

l : last milestone of the timeline(in timestamp)

f : first milestone of the timeline (in timestamp)

ir: interest rate of the range timeline, default is 18% /year

#### 2.4. Example

A range timeline has amount 1.000.000 VNĐ, first milestone in timestamp is

#### 2.5. Complexity

The Complexity is: O(1)

### **3. Lending/Payback**

#### 3.1. Definition

Lending/Payback is the mechanism used in our system to define the payment and notify to user.

#### 3.2. Define problem

#### Lender or borrower need to know the due date and the amount that they need to pay for the deal. However, nobody can remember all the due date whenever the large amount of lending/borrowing requests.

#### 3.3. Solution

The following steps define the stage of lending/payback process:

* 1. Due date is nearly reach, system will notify to the lender or borrower through email and notification on navigation bar.
  2. Lender/Borrower login into their account and check the notification, Lender/Borrower view directly request which have the milestone need to pay, then, they need to check timeline for paying the milestone.
  3. The transaction after done processing payment will be sent to BigchainDB Driver.
  4. The transaction will be prepared and signed, then, the BigchainDB transaction’s information after being stored will be sent to system.
  5. System will send the notification after transaction done.

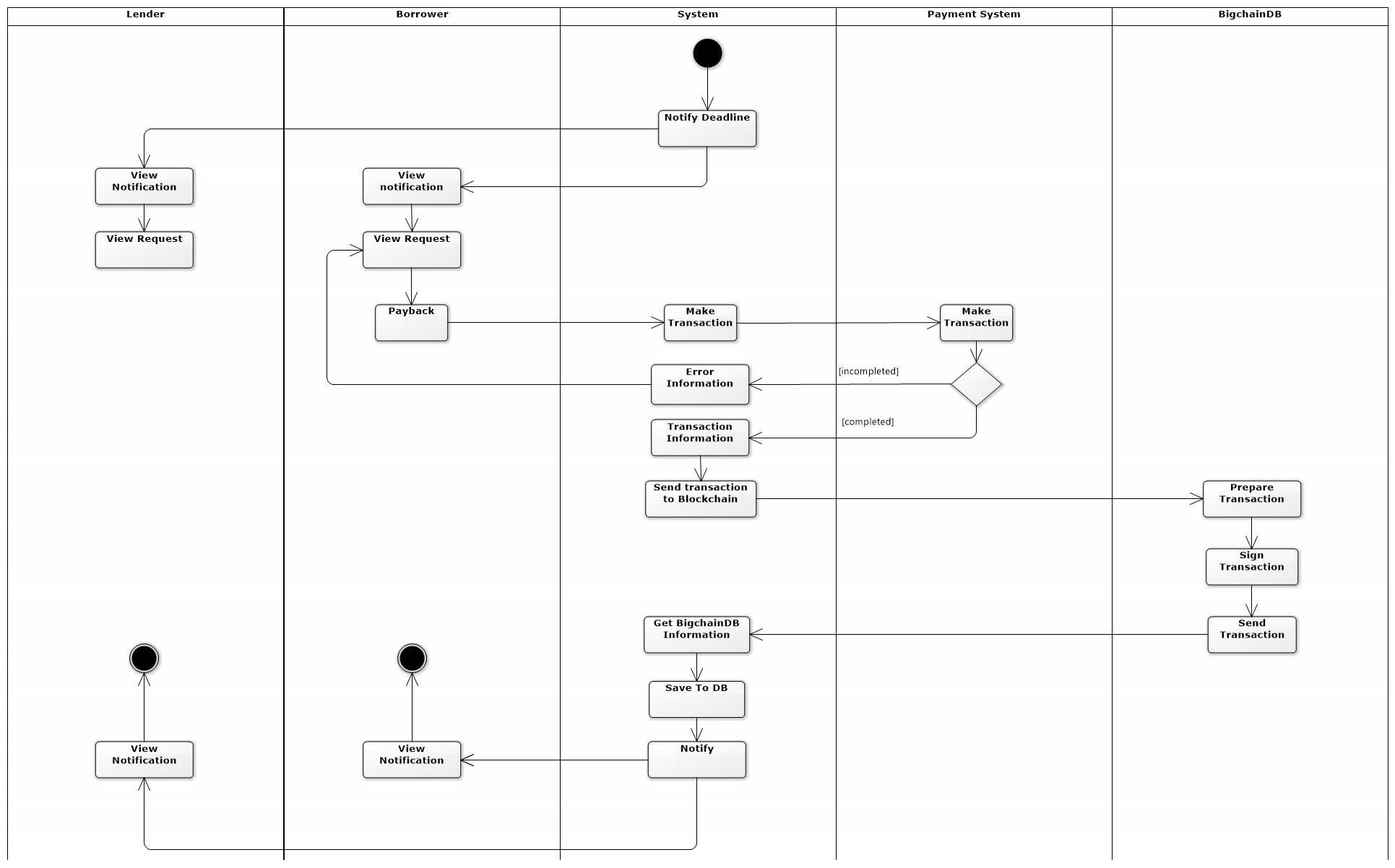


Figure 39– Lending/Payback process

# Future Plan

1. The system is currently being developed on the web platform:
   * We will develop our system on Android and iOS platform
2. The current system uses some user information to identify the user:
   * We will add more other user identification information to reduce risks when users join the system and increase user loan limits.
3. The current system cannot evaluate user reliability.

* We will design a scale to evaluate users after each transaction as well as milestones to be able to evaluate user reliability.

1. The current system uses paypal as a payment method.

* We will support more payment method such as Momo, Ethereum, and so on.

1. The current system cannot support user select deal that they want to choose when they use cancel deal function. Mean that when user cancel deal function system automatically restore deal to the original deal (the first deal that user create when they create request).

* We will design a function that help user can choose a deal that they want when they cancel deal. The function will show all deal between two users. So that, user can choose what they want.

1. The current system still not support investigation function.

* We will add investigation function to our system in the future to support the investigation.

# Explanation