BLOOD WALLET

DESIGN SPECIFICATION

Student Number	Name
2017310794	서성윤
2014311338	모경준
2014310967	곽희주
2017311290	김진범
2018312494	박상현
2020318179	Snaedis Perla Sigurdardottir

Contents

1. Preface

This chapter defines the expected readership of the document, and briefly introduces the content of each chapter. This chapter also describes version history including a rationale for the creation of a new version and a summary of the changes made in each version.

1.1 Readership

본 문서는 다양한 독자에게 읽힐 것을 상정하고 있다. 따라서 각 부분을 서술하는 데 있어 어떠한 독자층을 상정하고 있는지를 설명한다.

1.2 Document Structure

A. Introduction

본 문서를 서술하는 데 사용된 다양한 다이어그램과 표현 도구들에 대해 설명하고, 본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

B. System Architecture

시스템과 각 서브시스템의 구조를 개괄적으로 기술하고, 시스템의 전체 기능이 각 서브시스템과 하드웨어에 어떻게 할당되었는지 설명한다.

C. Protocol Design

프론트엔드 시스템과 백엔드 시스템간의 상호작용을 규정하는 인터페이스와 프로토콜을 어떻게 구성하는지에 대해 기술하고, 해당 인터페이스가 어떤 기술에 기반해 있는지 설명한다.

D. Database Design

Requirements 문서에서 규정된 데이터베이스 요구 사항을 기반으로, 각 데이터 엔티티의 속성과 관계를 ER diagram 을 통해 표현하고 최종적으로 Relational Schema 를 작성한다.

E. Testing Plan

미리 작성된 Test 를 이용해, verification 과 validation 을 시행한다. 이 Test 작성에 대한 계획을 설명한다.

F. Development Plan

시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 시스템 개발 일정을 기술한다.

G. Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2. Introduction

2.1 Objectives

이번 챕터에서는 본 시스템의 설계에 사용된 다양한 다이어그램과 도구를 소개하고, 본 시스템의 개발 범위를 기술한다.

2.2 Applied Diagram

A. UML

UML is a general purpose and developmental modelling language and technique that combines different aspects of a system to represent relations, processes or results of an overall model or system. It is essential to mention that we have used it thoroughly in this document to visualize the workflow of the system.

Since it provides different modelling techniques and a handful subset of diagrams. It can be efficiently used to provide means of communication between developers and users as it covers wide range of symbols and definitions and it consists of the following diagrams: Package Diagram, Deployment Diagram, Class Diagram, State Diagram, Sequence Diagram and ER Diagram.

B. Package Diagram

Package diagrams are kind of structural diagrams which show the arrangement and organization of model elements. Package diagram can show both structure and dependencies between sub-systems or modules in a more abstract way than other types of UML diagrams. This abstraction leads to the use of package diagrams in simplifying complex class diagrams by grouping them in packages.

C. Deployment Diagram

The deployment diagram describes the physical deployment of information generated by the software program (artifact)on hardware components.

Deployment diagrams are made up of several UML shapes. The three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or nodes, in the system. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed.

D. Class Diagram

It is a diagram that is used to showcase the object classes of a system and the relationship between classes. One of the most fundamental reasons we are using it is because it provides a clear distinction between each class and show the hierarchy and dependency between them.

As far it goes for the inner structure of Class diagram, it consists of some fields indicating some variables, class methods and links or associations between classes.

E. State Diagram

State Diagram is a technique to represent different states of a system and all possible next states based on some particular stimuli which triggers the change of the state.

This kind of diagram is very important to analyze different scenarios of the system as the states are represented as nodes and events as arcs which helps in identifying the behavior of the object classes defined in class diagram.

F. Sequence Diagram

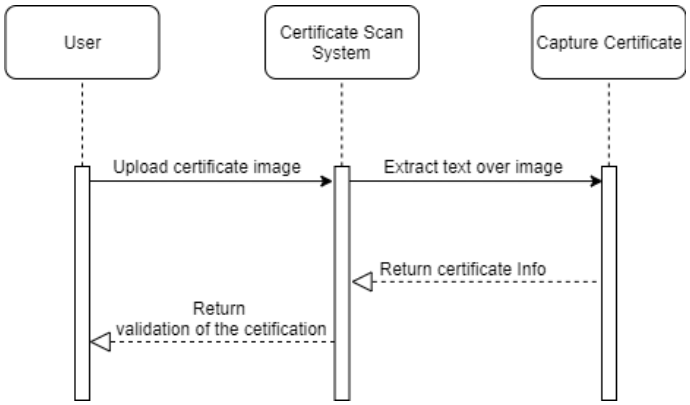


Figure 1: Example of Sequence Diagram

A sequence diagram to represent the interactions between the actors and objects of the system. To be more specific, the goal of this diagram is present the sequence of interaction and processes that take place in a specific use case instance so that a result could be generated. It is important to notice that the direction of the arrows here is essential to indicate the correct flow of actions.

G. ER Diagram

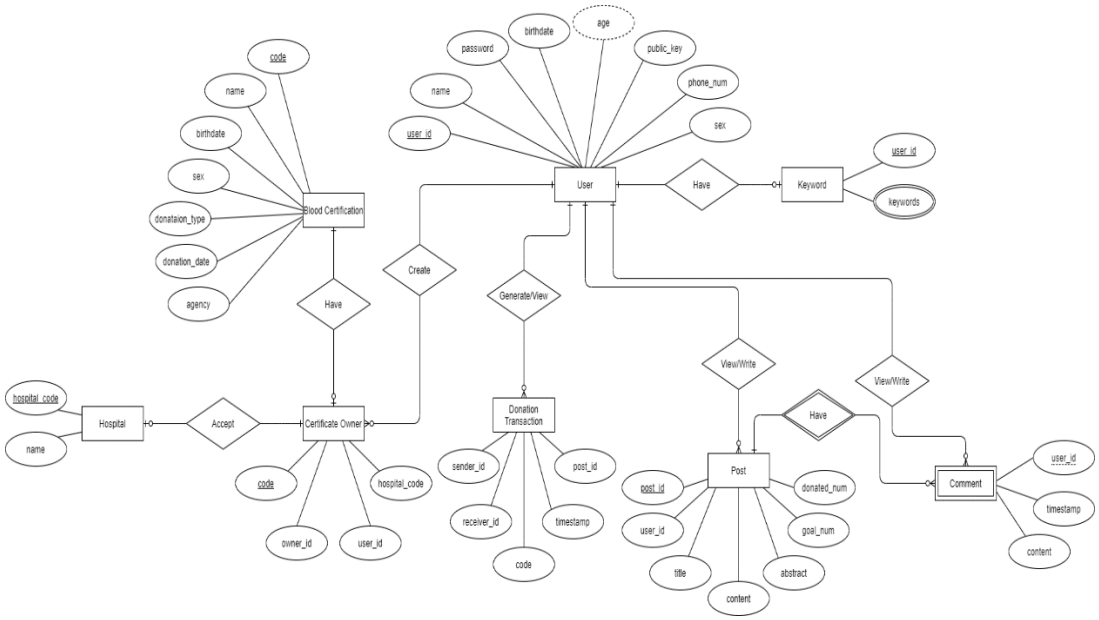


Figure 2: Example of ER Diagram

ER Diagram 은 Entity 가 가지고 있는 속성과 Entity 간의 관계를 나타낸 다이어그램이다. 이 다이어그램은 주로 데이터베이스를 설계하는 데 사용되며, 해당 다이어그램을 기반으로 Relational Schema 를 작성하게 된다.

2.3 Applied Tool

A. Draw.io

Figure 3: Draw.io Logo

Draw.io 는 온라인 모델링 툴로서 많은 기본 템플릿과 도형을 제공하기 때문에 사용자가 직접 다이어그램에 사용하기 위해 도형을 만들 필요가 없다. 또한 도형 간 연결선을 간단하게 만들 수 있고, 격자에 위치를 맞출 수 있기 때문에 도형을 정렬하기 편리하다. 이 문서에서 사용된 대부분의 다이어그램은 본 도구로 작성되었다.

B. PowerPoint

Figure 4: Powerpoint Logo

Powerpoint 는 그래픽 프레젠테이션 툴이다. 주로 발표용으로 사용되지만 내장된 도형 작성 기능이 매우 강력하기 때문에 draw.io 에서 만들기 힘든 복잡한 다이어그램을 작성하기 위해 사용하였다.

2.4 Project Scope

본 시스템은 물리적으로 존재하는 헌혈 증서의 불편한 관리와 헌혈 증서 기부 요청을 직접 일일이 해야 하는 번거로움을 해결하기 위해 클레이튼 블록체인 플랫폼을 도입하여 헌혈 증서의 물리적 양도 기록을 신뢰도 높게 관리하는 시스템이다. 본 시스템의 핵심 기능은 헌혈 증서의 기부 기록을 블록체인 네트워크를 이용하여 관리하지만 동시에 사용자는 본 시스템에서 블록체인에 대한 지식이 없어도 사용할 수 있도록 블록체인 기술을 추상하여 사용자 경험을 제공한다. 따라서 사용자는 더 쉽게 헌혈 증서를 관리할 수 있고 동시에 더 많은 사람에게 기부할 수 있고, 기부를 요청할 수 있다.

먼저 Frontend System 은 시스템과 사용자와의 상호작용을 담당하며, Backend System Frontend System 에서 오는 데이터 요청에 응답하고 Certificate Scan, Certificate donation 등 sub-system 을 실행시키는 역할을 담당한다. 사용자는 발급받은 물리 헌혈 증서를 촬영하면 Certificate Scan System 에서 헌혈 증서의 유효성을 판단하고, Post Ranking System 으로 사용자 선호 키워드로 커스텀된 기부 요청 글을 읽을 수 있고, Certificate Donation System 으로 헌혈 증서를 기부할 수 있다.

3. System Architecture – Overall

3.1 Objectives

이번 챕터에서는 본 시스템의 전체적인 구조를 설명한다. 시스템 전체의 구조와 각 서브시스템의 개략적인 구조, 서브시스템 간의 관계를 서술한다.

3.2 System Organization

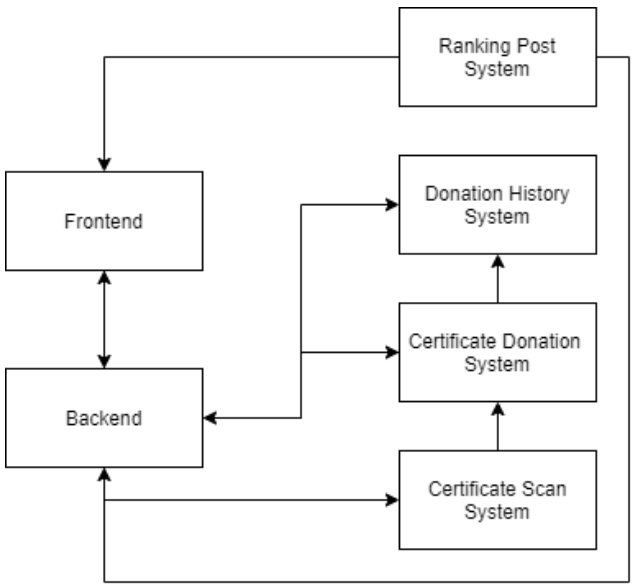


Diagram 1: Overall System Organization

본 서비스는 클라이언트-서버 모델을 적용해 설계했으며, Frontend Application 이 사용자와의 모든 상호작용을 맡고, 프론트엔드 애플리케이션과 백엔드 애플리케이션은 JSON 을 기반으로 한 HTTP 통신으로 데이터를 송수신한다. 본 시스템에서 백엔드 애플리케이션은 Firebase 를 통해 데이터베이스에서 필요한 객체 정보를 JSON 포맷으로 가공하여 전달한다.

백엔드 애플리케이션은 사용자가 등록하고자 하는 헌혈 증서를 촬영한 이미지가 전달 받으면 Certificate Scan System 을 실행하는데, 이 시스템을 통해 이미지의 정보를 인식하고 사용자의 정보 확인 후, 해당 헌혈 증서의 유효성을 판단한다. 이 과정을 통해 등록된 헌혈 증서는 Post Ranking System 통해 고른 기부 사연에 대해 헌혈 증서를 Certificate Donation System 을 통해 기부가 되고 완료되면 변경된 사용자 정보를 데이터베이스에 저장한다. Certificate Donation System 을 통해 블록체인 네트워크에 반영된 기부 기록은 Donation History System 을 통해 사용자의 기부 내역을 확인할 수 있다.

A. Frontend Application

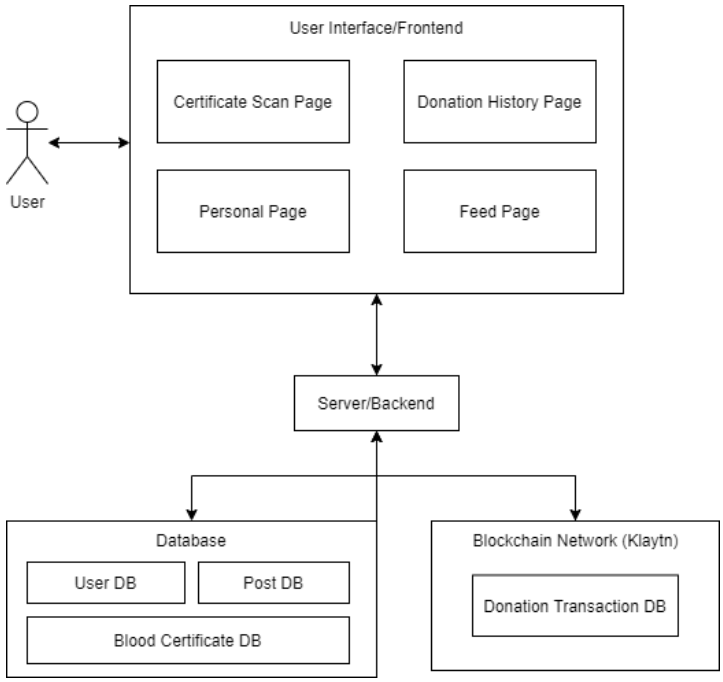


Diagram 2: System Architecture – Frontend

사용자와의 상호작용을 전담하는 시스템으로, Android Studio 를 통해 XML View 와 Java activity 를 통해 관리한다. Certificate Scan Page, Donation History Page, Personal Page, Post Feed page 가 존재하고 Local Data Repository 서브시스템을 통해서 네트워크가 끊긴 상태에서도 작동 가능하도록 설계하였다.

B. Backend Application

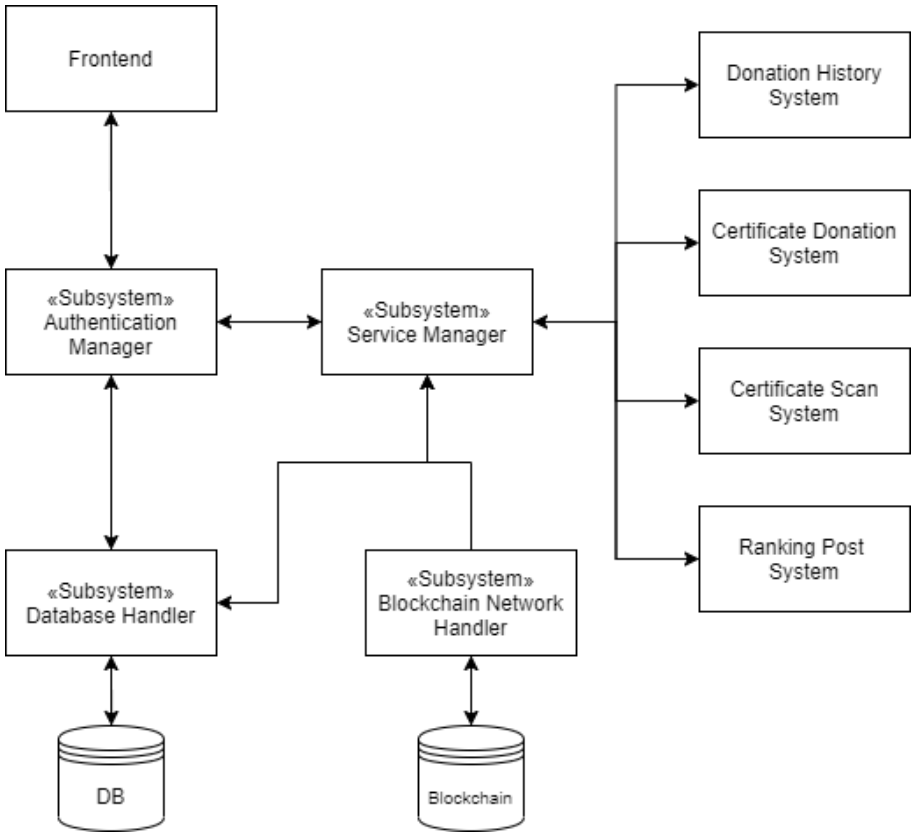


Diagram 3: System Architecture - Backend

4. System Architecture – Frontend

4.1 Objective

전체 시스템 아키텍처 중 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트간의 관계를 서술한다.

4.2 Subcomponents

A. Login

1. Class Diagram

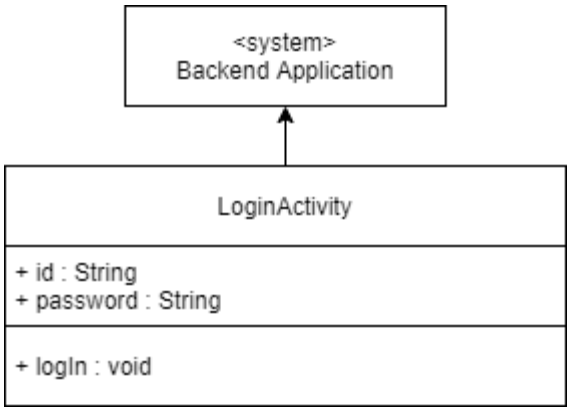


Diagram 4: System Architecture - Frontend – Login

1. LogInActivity – 로그인 액티비티

A. Attributes

+ id : 사용자 아이디

+ password : 사용자 비밀번호

B. Methods

+ logIn() : 아이디와 비밀번호를 서버로 보내서 로그인 요청

2. Sequence Diagram

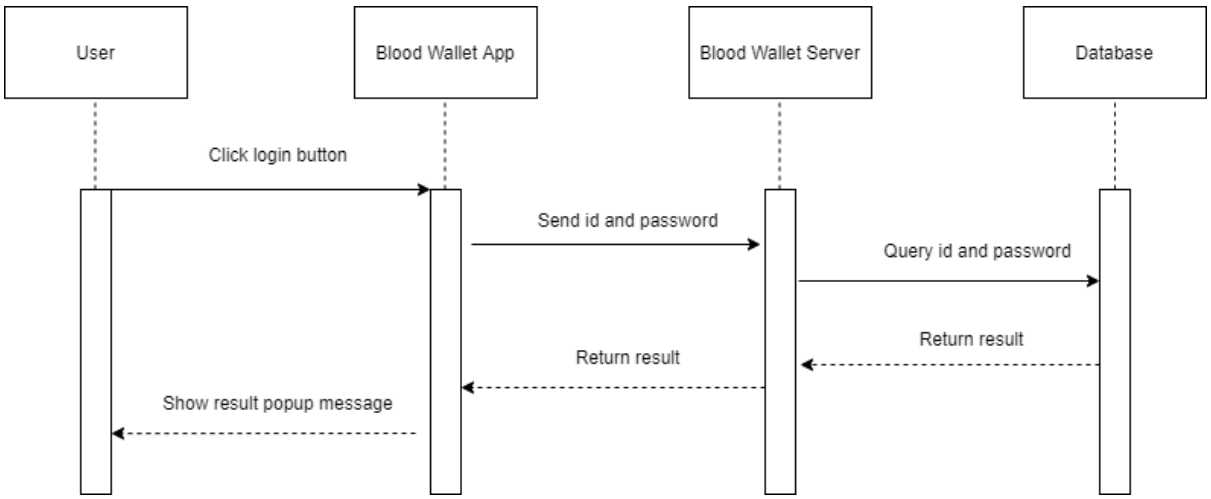


Diagram 5: System Architecture – Frontend - Login - Sequence Diagram

B. Register

1. Class Diagram

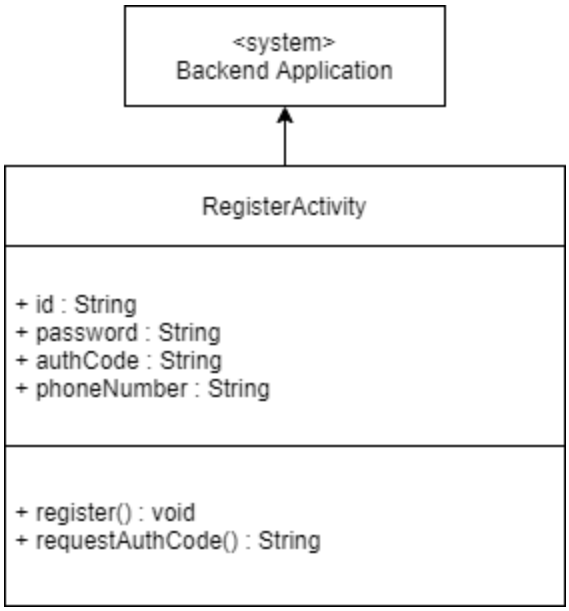


Diagram 6 - System Architecture - Frontend – Register

1. RegisterActivity – 회원가입 액티비티

A. Attributes

+ id : 사용자 아이디

+ password : 사용자 비밀번호

+ authCode : 휴대폰 인증 문자

+ phoneNumber : 사용자 휴대폰 번호

B. Methods

+ register() : 서버에 회원가입 요청
+ requestAuthCode() : 서버에 휴대전화 인증 문자 요청

2. Sequence Diagram

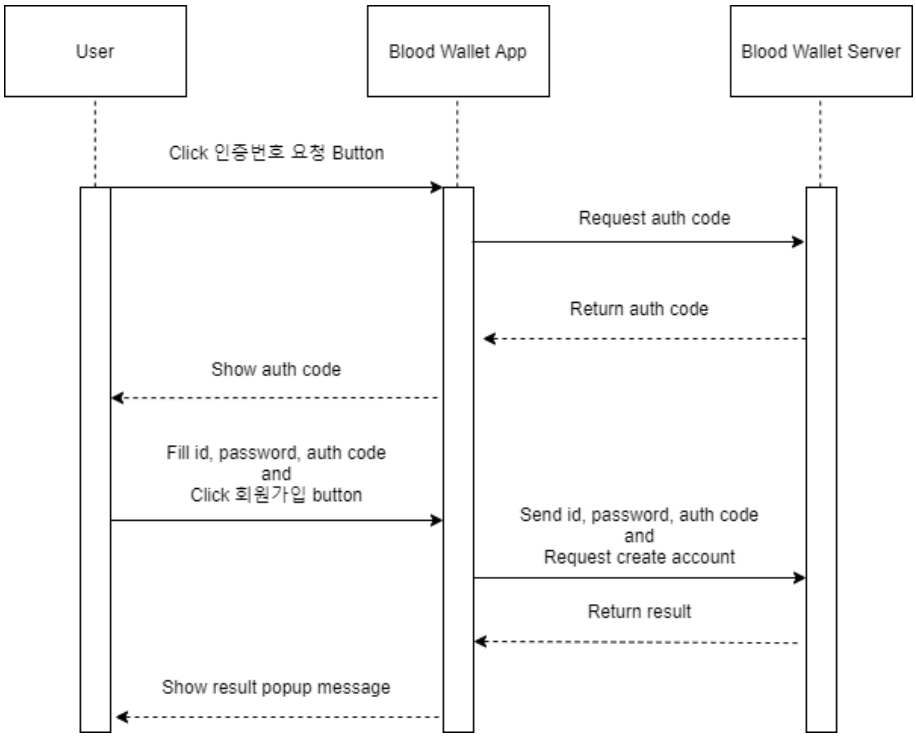


Diagram 7. System Architecture - Frontend – Register - Sequence Diagram

C. Searching a Post

1. Class Diagram

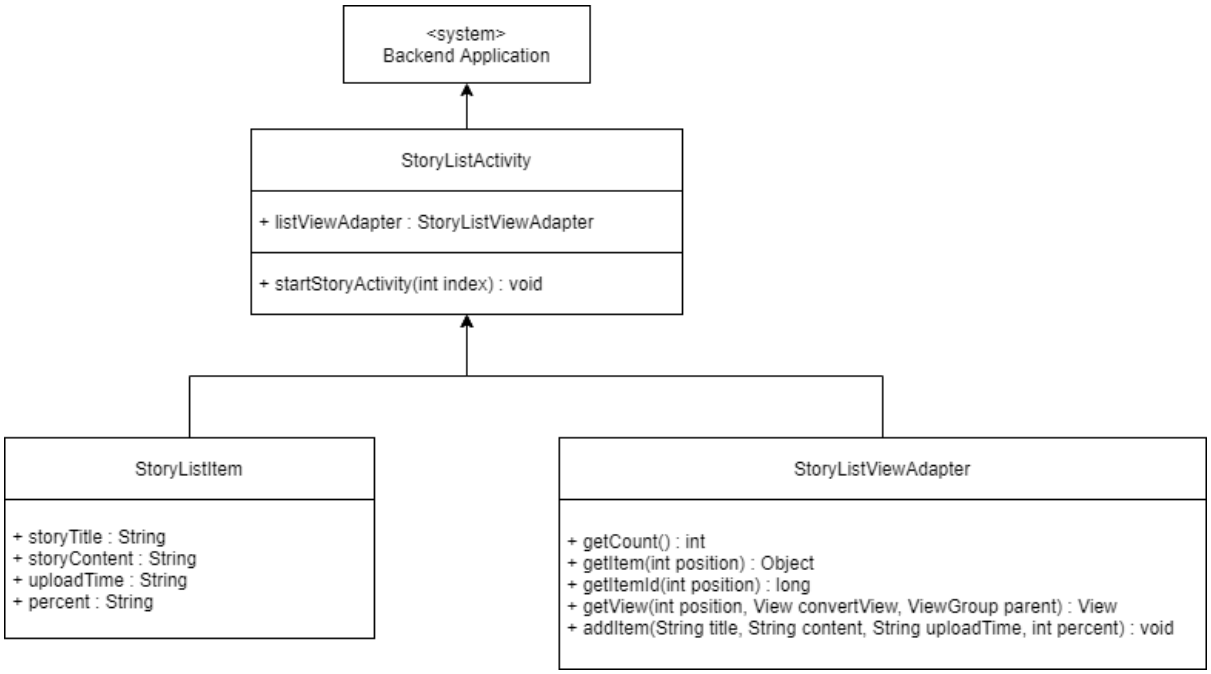


Diagram 8: System Architecture - Frontend – Searching Post

1. StoryListActivity – 스토리 검색 액티비티

A. Attributes

+ listViewAdapter : 스토리 목록을 관리하는 객체

B. Methods

+ startStoryActivity(int index) : 클릭한 스토리의 상세 액티비티로 이동

2. StoryListItem – 스토리에 대한 정보를 가지고 있는 객체

A. Attributes

+ storyTitle : 스토리의 제목

+ storyContent : 스토리의 내용

+ uploadTime : 스토리를 작성한 시간

+ percent : 기부가 진행된 정도

3. StoryListViewAdapter – 스토리를 관리하고 화면으로 연결해주는 객체

A. Methods

+ getCount() : 스토리의 개수를 반환

+ getItem(int position) : 해당 인덱스의 스토리를 반환

+ getItemId(int position) : 스토리의 인덱스를 반환

+ getView(int position, View convertView, ViewGroup parent) : 화면 출력

+ addItem(String title, String content, String uploadTime, int percnet) : 관리할 스토리 객체 추가

2. Sequence Diagram

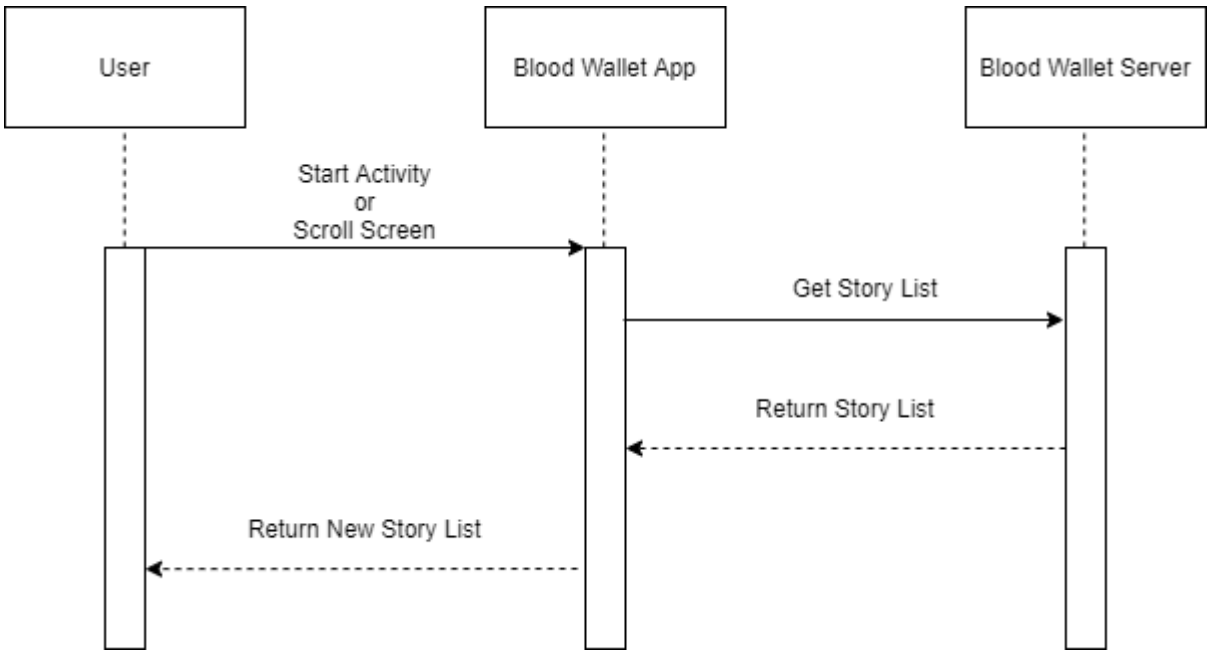


Diagram 9. System Architecture - Frontend – Searching Post – Sequence Diagram

D. Writing a Post

1. Class Diagram

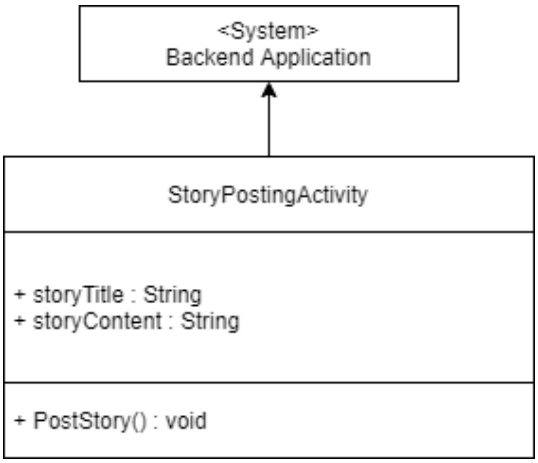


Diagram 10: System Architecture - Frontend – Writing a Post

1. StoryPostingActivity – 스토리 작성하는 액티비티

A. Attributes

+ storyTitle : 스토리의 제목

+ storyContent : 스토리의 내용

B. Methods

+ PostStory() : 스토리 업로드를 서버에 요청

2. Sequence Diagram

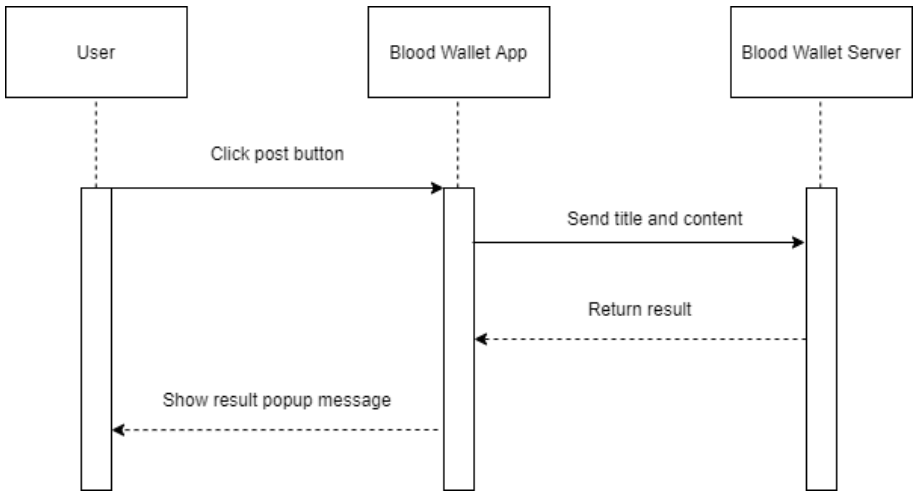


Diagram 11. System Architecture - Frontend – Writing a Post – Sequence Diagram

E. Donation

1. Class Diagram

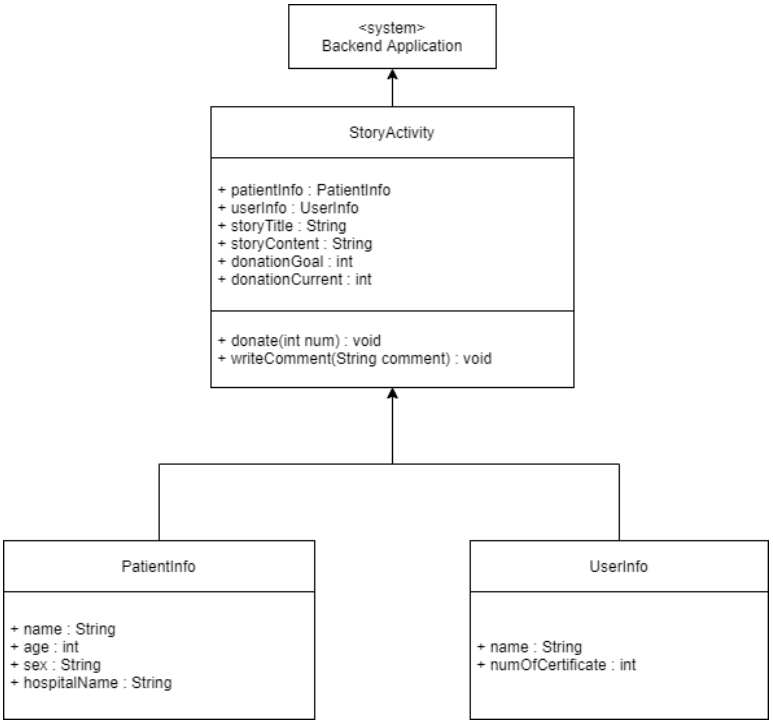


Diagram 12. System Architecture - Frontend – Donation

1. StoryActivity – 기부 액티비티

A. Attributes

+ patientInfo : 환자 정보

+ userInfo : 사용자 정보

+ storyTitle : 스토리 제목

+ storyContent : 스토리 내용

+ donationGoal : 목표 헌혈증 갯수

+ donationCurrent : 현재 기부된 헌혈증 갯수

B. Methods

+ donate(int num) : 선택한 헌혈증 수만큼 기부

+ writeComment(String comment) : 응원 댓글 작성

2. PatientInfo – 환자 정보

A. Attributes

+ name : 환자 이름

+ age : 환자 나이

+ sex : 환자 성별

+ hospitalName : 소속 병원 이름

3. UserInfo – 사용자 정보

A. Attributes

+ name : 사용자 이름

+ numOfCertificate : 보유하고 있는 헌혈증 개수

2. Sequence Diagram

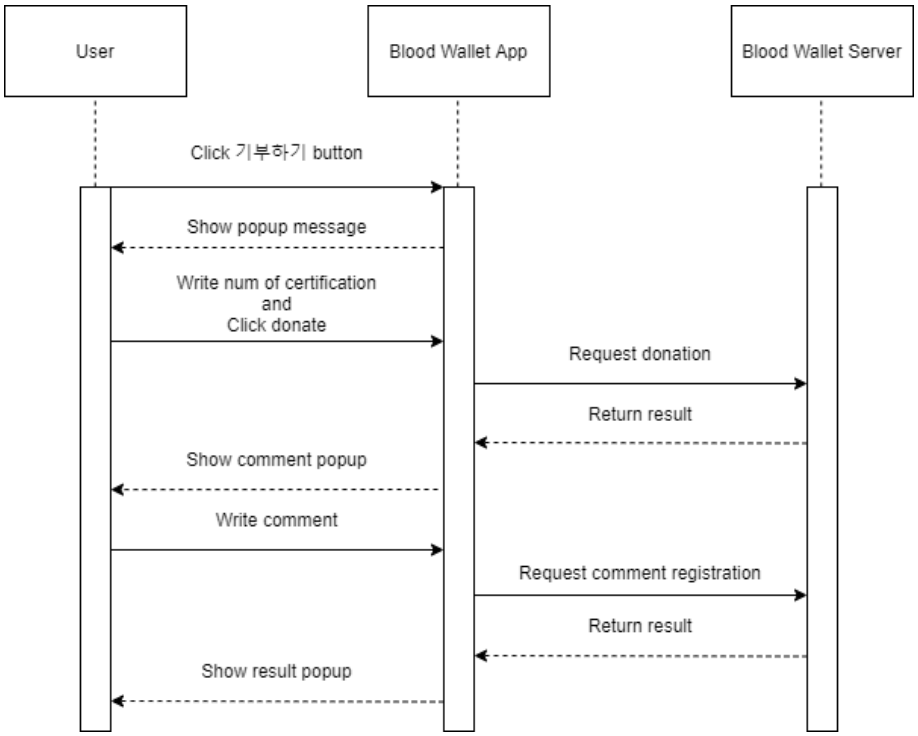


Diagram 13. System Architecture - Frontend – Donation – Sequence Diagram

F. Donation History

1. Class Diagram

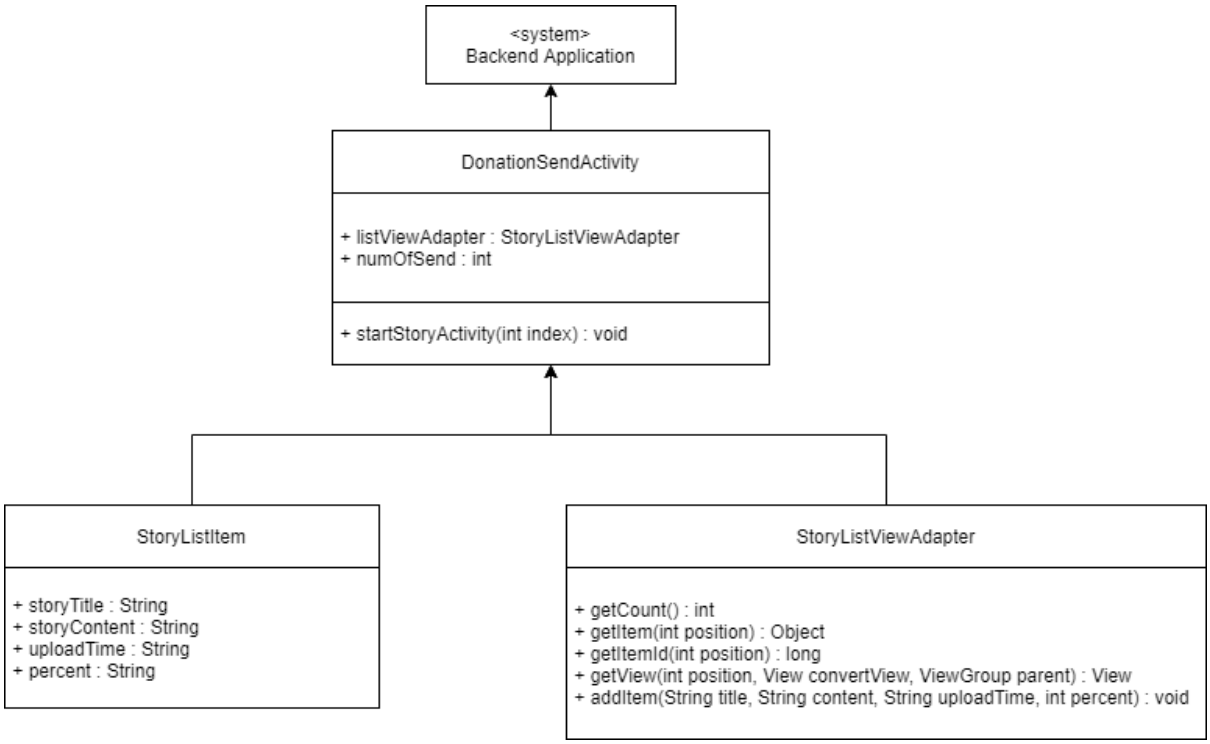


Diagram 14. System Architecture - Frontend – Donation History

1. DonationSendActivity – 기부한 내역 액티비티

A. Attributes

+ listViewAdapter : 스토리 목록을 관리하는 객체

+ numOfReceive : 기부를 받은 헌혈증 갯수

B. Methods

+ startStoryActivity(int index) : 클릭한 스토리의 상세 액티비티로 이동

2. StoryListItem – 스토리에 대한 정보를 가지고 있는 객체

A. Attributes

+ storyTitle : 스토리의 제목

+ storyContent : 스토리의 내용

+ uploadTime : 스토리를 작성한 시간

+ percent : 기부가 진행된 정도

3. StoryListViewAdapter – 스토리를 관리하고 화면으로 연결해주는 객체

A. Methods

+ getCount() : 스토리의 개수를 반환

+ getItem(int position) : 해당 인덱스의 스토리를 반환

+ getItemId(int position) : 스토리의 인덱스를 반환

+ getView(int position, View convertView, ViewGroup parent) : 화면 출력

+ addItem(String title, String content, String uploadTime, int percnet) : 관리할 스토리 객체 추가

2. Sequence Diagram

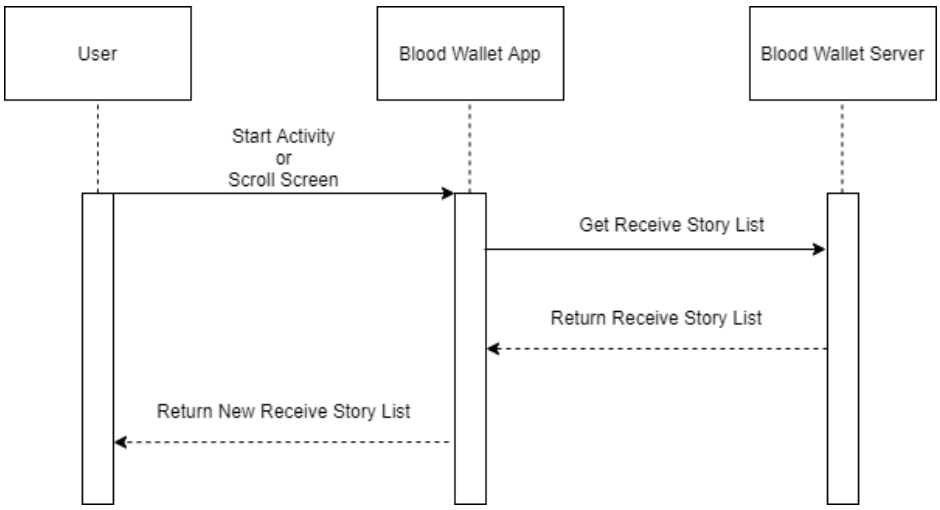


Diagram 15. System Architecture - Frontend – Donation History – Sequence Diagram

G. My Page

1. Class Diagram

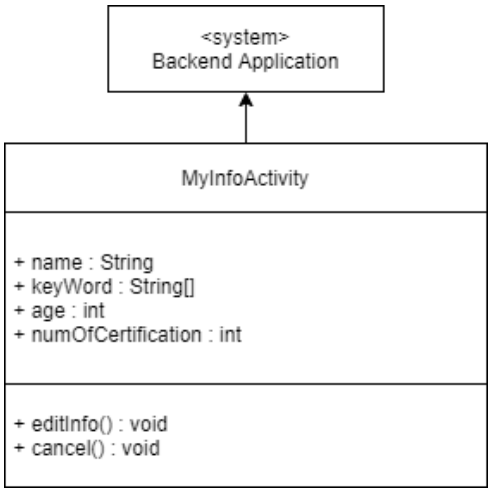


Diagram 16. System Architecture - Frontend – My Page

1. MyInfoActivity – 내 정보 액티비티

A. Attributes

+ name : 사용자 이름

+ keyWord : 관심 질병 단어 리스트

+ age : 사용자 나이

+ numOfCertificate : 보유하고 있는 헌혈증 갯수

B. Methods

+ editInfo() : 변경한 내용대로 내 정보 수정

+ cancel() : 변경한 내용을 수정하지 않고 취소

2. Sequence Diagram

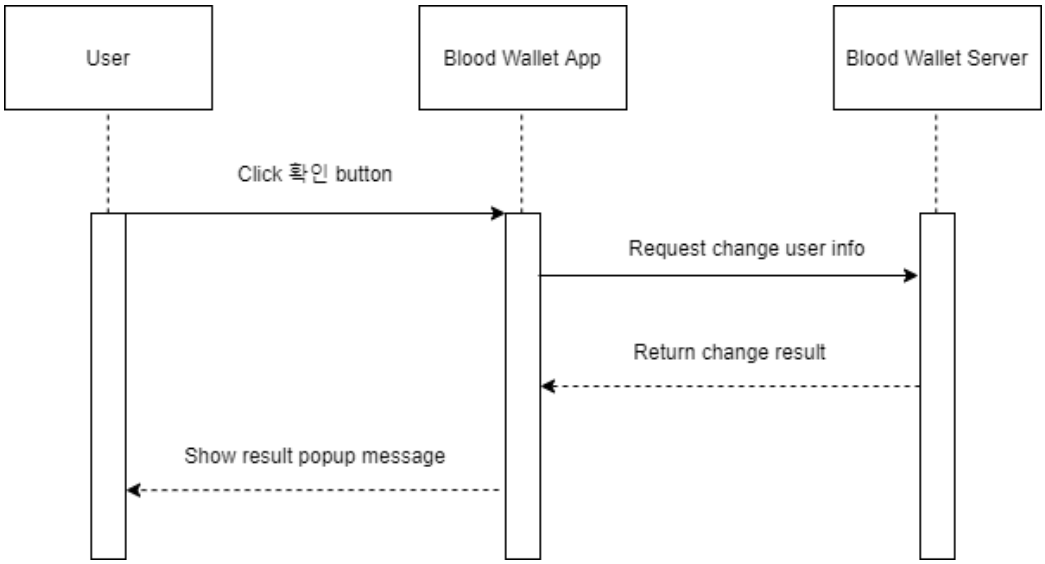


Diagram 17. System Architecture - Frontend – My Page – Sequence Diagram

5. System Architecture – Backend

5.1 Objectives

이번 챕터에서는 전체 시스템 중 사용자와의 상호작용을 담당하는 프론트엔드를 제외한 백엔드 시스템과 각 서브시스템의 구조에 대해 설명한다.

5.2 Subcomponents

A. Certificate Scan System

1. Class Diagram

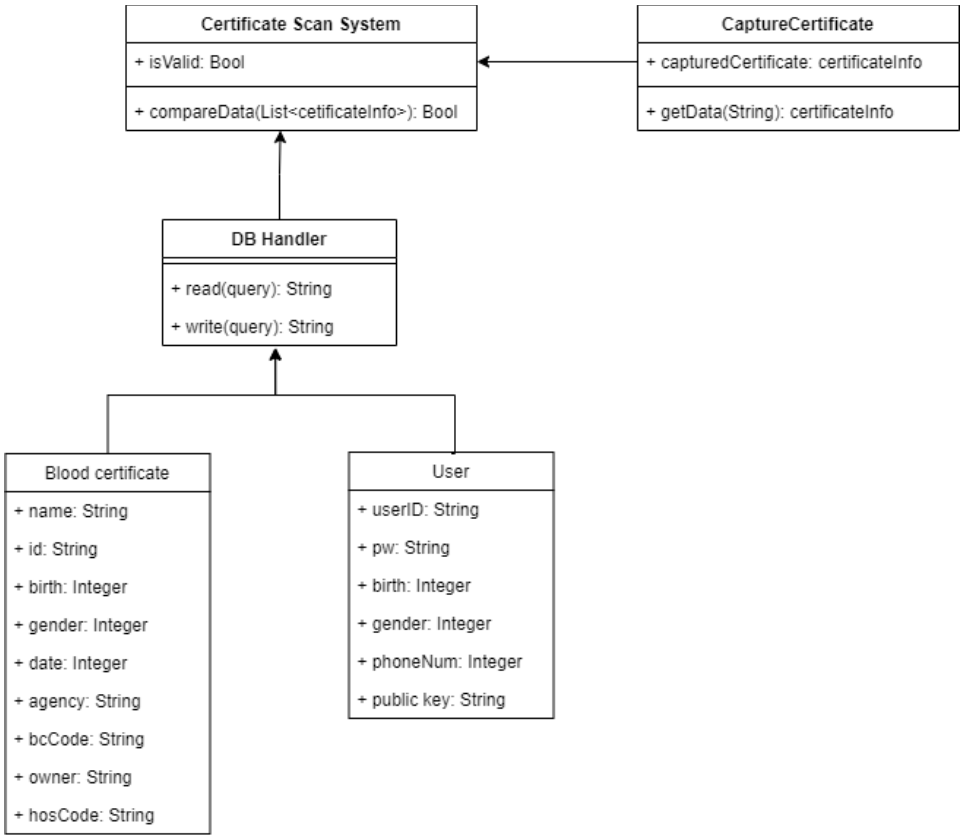


Diagram 18. System Architecture - Backend – Certificate Scan System

1. Certificate Scan System Class: 이 클래스는 사진 찍은 헌혈 증서가 유효한 것인지 판단하고 데이터베이스에 등록하는 클래스이다. 이 클래스는 아래 필드와 메소드를 가지고 있다.

   A. isValid field: 백엔드로 넘어온 헌혈 증서의 정보가 유효한지 나타낸다.

   B. compareData method: 이 함수는 데이터베이스에서 조회한 헌혈 증서와 검증하고자 하는 헌혈 증서를 비교하여 실제 발급 여부와 사용자 본인의 헌혈 증서가 맞는지 확인한다.

2. DB Handler Class: 이 클래스는 백엔드 데이터베이스와 작동한다.

   A. read method: 데이터베이스에서 확인하고자 하는 헌혈 증서의 발급 여부와 유효성을 확인하기 위해 데이터를 읽어드린다.

   B. write method: 헌혈 증서가 성공적으로 등록이 가능한 경우에 이를 데이터베이스에 반영하기 위해 데이터를 업데이트하는 메소드이다.

3. Capture Certificate Class: 이 클래스는 프론트에서 넘어온 이미지에서 헌혈 증서 정보를 회수한다.

   A. capturedCertificate field: 헌혈 증서의 명시적으로 나타난 정보를 담는 기본 클래스 필드이다.

   B. getData method: 헌혈 증서 이미지에서 Firebase ML kit OCR API 를 통해 텍스트를 추출하여 정보를 가져온다.

4. Blood certificate, User Classes: 사용자와 헌혈 증서 발급 정보를 담은 기본적인 클래스들이다.

2. Sequence Diagram

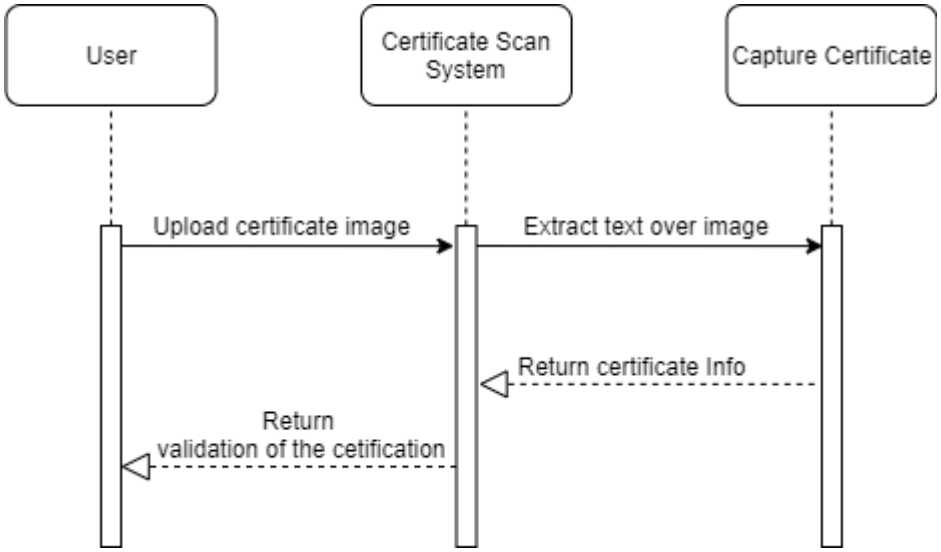


Diagram 19. System Architecture - Backend – Certificate Scan System – Sequence Diagram

B. Certificate Donation System

1. Class Diagram

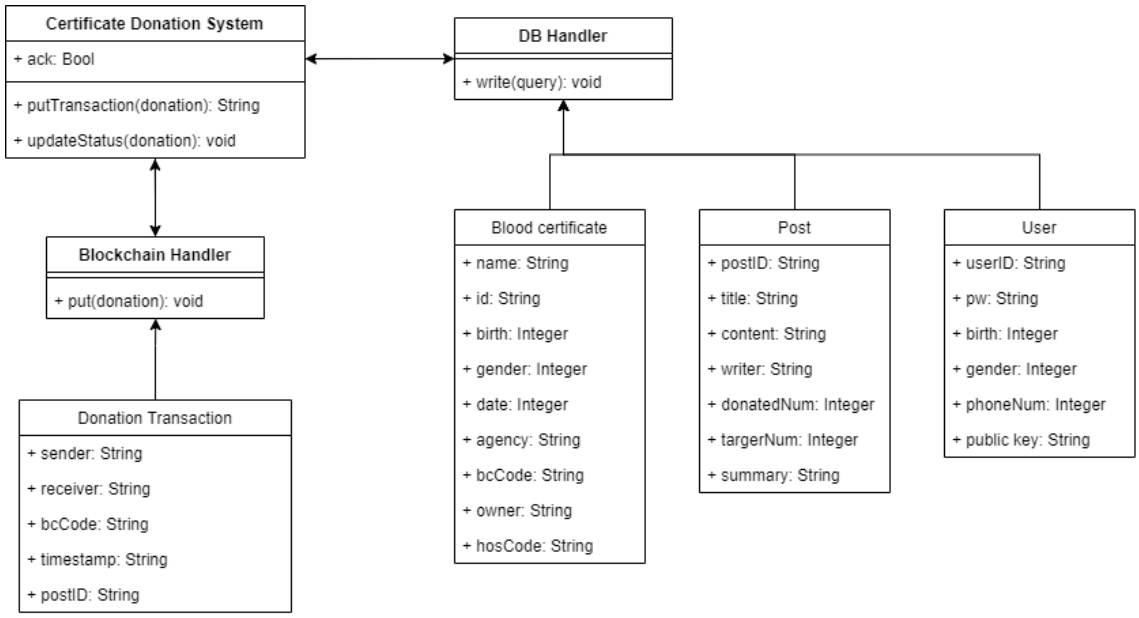


Diagram 20. System Architecture - Backend – Certificate Donation System

1. Certificate Donation System Class: 이 클래스는 헌혈 증서를 기부하는 트랜잭션을 생성하고 기록하는 시스템이다. 해당 클래스는 아래와 같은 필드와 메소드를 가진다.
   A. ack field: 헌혈 증서 양도 트랜잭션이 모두 성공적으로 수행되었는지에 대한 값을 가진다.
   B. putTransaction method: 프론트엔드에서 제공한 헌혈 증서 기부 관련 정보를 가지고 블록체인에 트랜잭션을 보내는 메소드이다.
   C. updateStatus method: 블록체인에 성공적으로 기부 트랜잭션이 기록되었다면 데이터베이스에 기부된 헌혈 증서, 어떤 사연을 통해 기부되었는지 update 를 한다.

2. DB Handler Class: 이 클래스는 백엔드 데이터베이스와 작동한다.
   A. read method: 데이터베이스에서 확인하고자 하는 사용자 정보와 헌혈 증서 발급 조회를 위해 데이터를 읽어드린다.
   B. write method: 헌혈 증서 기부가 성공적으로 완료가 되면 이를 데이터베이스에 반영하기 위해 데이터를 업데이트하는 메소드이다.

3. Blockchain Handler Class: 블록체인 네트워크에 트랜잭션을 보내는 클래스이다.
   A. put method: 보내고자 하는 트랜잭션에 사용자 private key 로 디지털 서명을 한 후 Fee Payer 를 통해 블록체인 네트워크에 반영한다.

4. Donation Transaction, Blood certificate, Post, User classes: 객체의 정보를 담고 있는 기본적인 클래스들이다.

2. Sequence Diagram

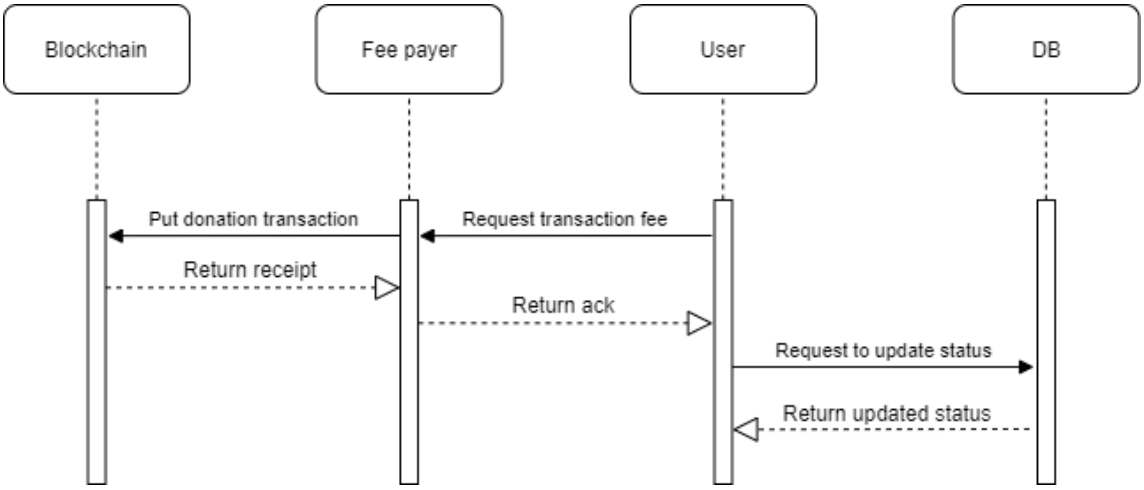


Diagram 21. System Architecture - Backend – Certificate Donation System – Sequence Diagram

사용자에게 본 시스템에서 블록체인을 사용하고 있음을 추상화하기 위하여 Fee payer 를 도입하였다. Fee payer 는 User 의 트랜잭션에 대한 수수료를 대신 지불하는 서비스 제공자이다. 따라서 사용자는 지갑에 토큰이 없어도 클레이튼 블록체인 네트워크에 트랜잭션을 기록할 수 있다.

C. Donation History System

1. Class Diagram

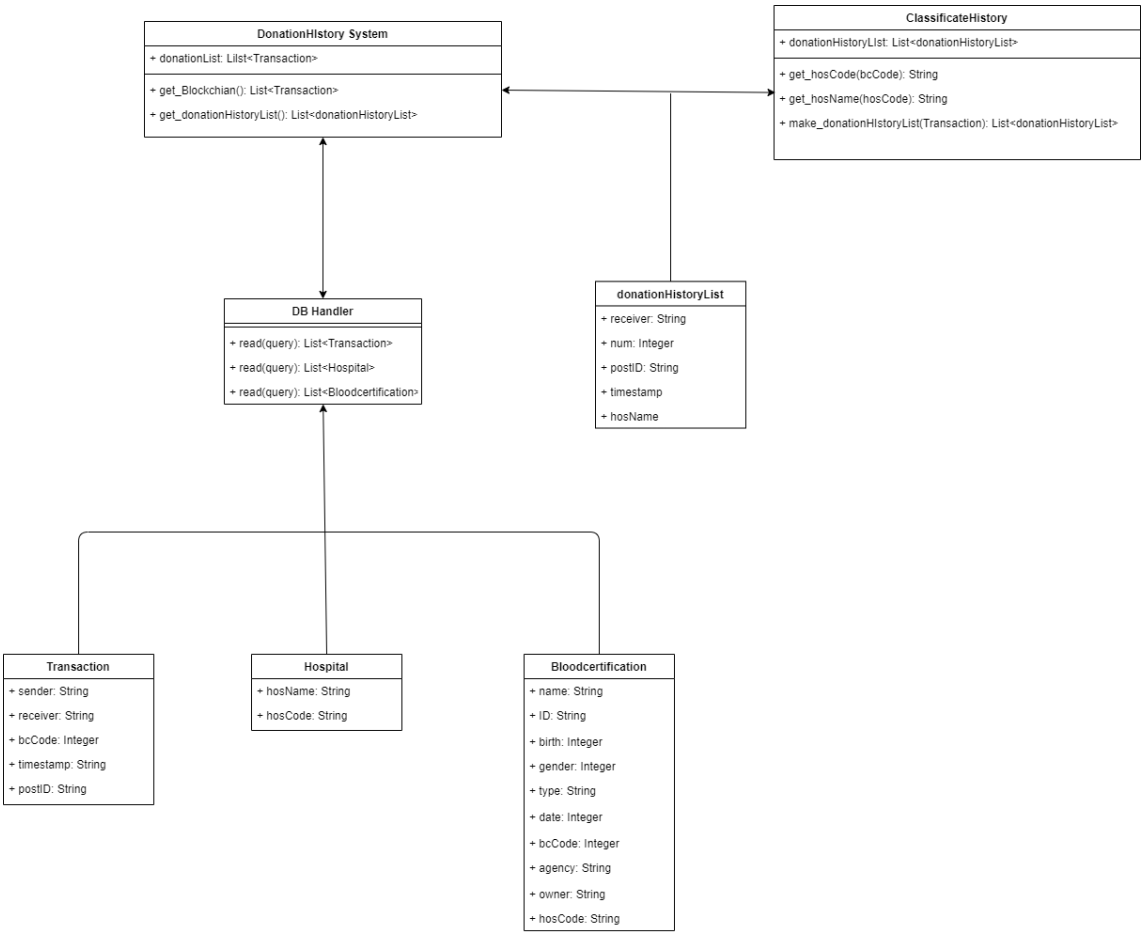


Diagram 22. System Architecture - Backend – Donation History System

1. DonationHistory System Class: 이 클래스는 다른 클래스로부터 함수들을 불러오기 위한 driver class이다. 또한 아래와 같은 field와 method를 갖는다.

    A. donationHistoryList field: 사용자의 기부 내역들의 리스트로 ClassificateHistory 로부터 load 한다.

    B. get_Blockchain method: 이 함수는 DB Handler 로부터 Blockchain DB 의 정보를 가져와 리스트로 저장한다.

    C. get_donationHistoryList method: 이 함수는 ClassificateHistory Class 로부터 donationHistoryList 를 가져온다.

2. DB Handler Class: 이 클래스는 backend database와 상호작용을 한다.
   A. read(query) method: 이 함수는 Blockchain, Hospital, Bloodcertification DB 로부터 값을 가져와 리스트 형태로 저장한다.

3. ClassificateHistory Class: 이 클래스는 실질적으로 기부내역을 불러오는 Class이다. 아래 네개의 method에 의해 작동된다.
   A. get_hosCode method: bcCode를 사용해서 Blockchain DB에서 hosCode를 찾아내는 함수이다.
   B. get_hosName method: hosCode를 사용해서 Hospital DB에서 hosName을 가져온는 함수이다.
   C. make_donationHistoryList method: postID별로 Blockchain에 있는 list를 sort한 후, 하나의 post에서 몇 개의 헌혈증을 기부하였는지 저장하는 donationHistoryList를 생성한다.

4. Blockchain, Hospital, Bloodcertification, donationHistoryList Class: 이 네개의 class는 오브젝트를 저장하는 클래스이다.

2. Sequence Diagram

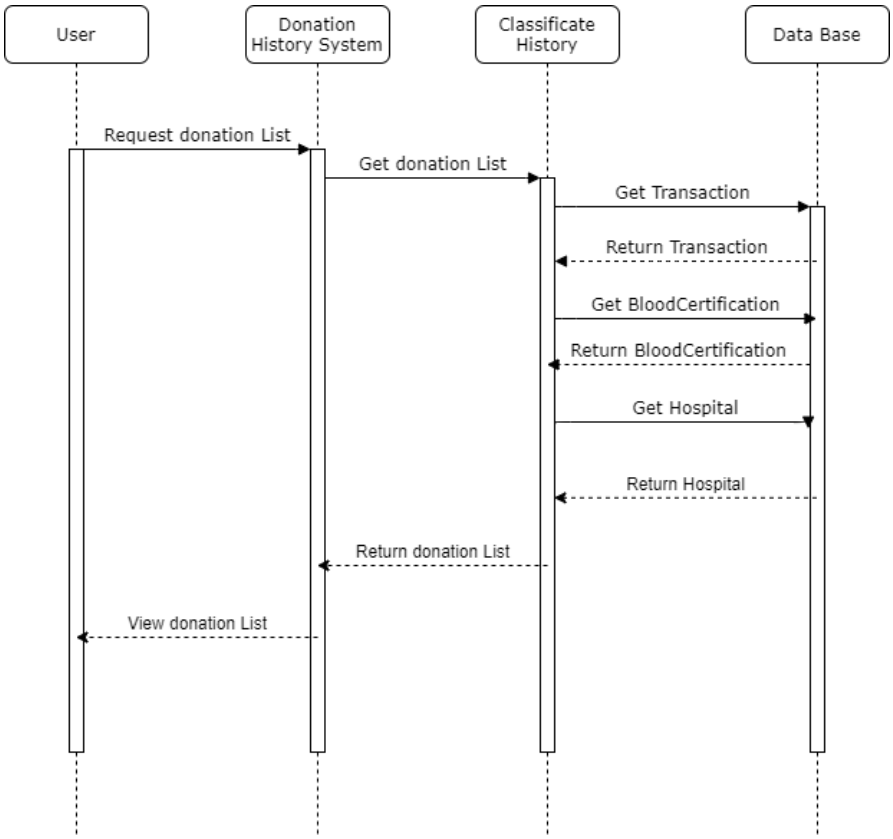


Diagram 23. System Architecture - Backend – Donation History System – Sequence Diagram

D. Ranking Post System

1. Class Diagram

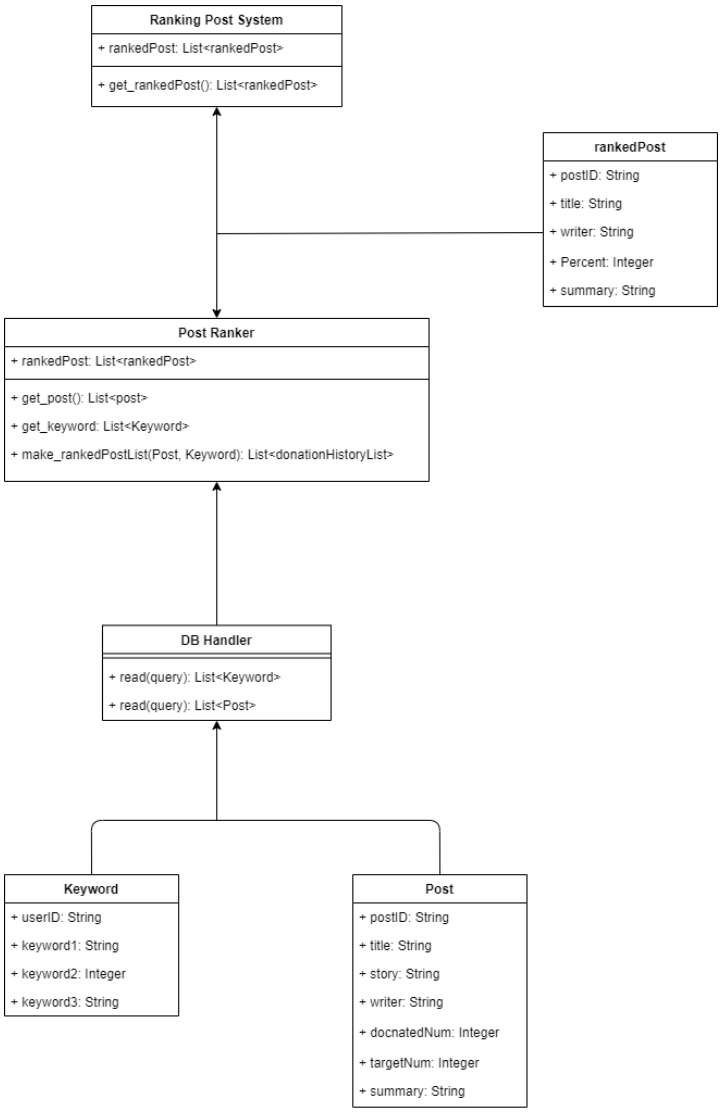


Diagram 24. System Architecture - Backend – Ranking Post System

1. Ranking Post System Class: 이 class는 함수를 다른 class에서 load해오는 driver class이다. 아래와 같은 field와 method를 갖는다.

   A. rankedPost field: post들을 기준에 따라 정렬한 리스트로 Post Ranker class에서 load한다.

   B. get_rankedPost method: Post Ranker class로부터 rankedPost를 불러오는 함수이다.

2. DB Handler Class: 이 클래스는 backend database와 상호작용을 한다.

   a) read(query) method: Keyword, Post DB로부터 데이터를 읽어온다.

3. Post Ranker Class: 이 클래스는 DB에서 읽어온 post들을 정렬하는 실질적인 class 이다. 사용자가 지정한 keywords, post의 헌혈증 기부 현황을 통해 순위를 정한다.

   a) get_post method: DB Handler로부터 post를 리스트로 가져오는 함수이다.

   b) get_keyword method: DB Handler로부터 keyword를 리스트로 가져오는 함수 이다.

   c) make_rankedPostList Method: story에 몇 개의 keyword가 포함되어 있는지 세고, donatedNum/targetNum으로 post들의 순위를 정하고, 순서대로 list에 넣어 반환하는 함수이다.

4. rankedPost, Keyword, Post Class: 이 세개의 class는 오브젝트를 저장하는 클래스 이다.

2. Sequence Diagram

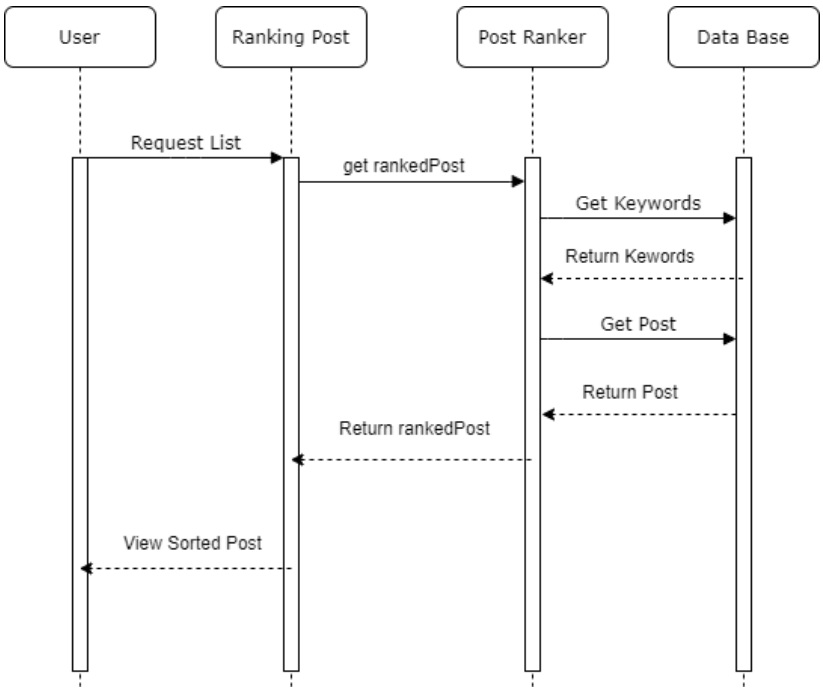


Diagram 25. System Architecture - Backend – Ranking Post System – Sequence Diagram

6. Protocol Design

6.1 Objectives

이번 챕터에서는 각 서브시스템 간의 상호작용, 특히 프론트엔드 시스템과 백엔드 애플리케이션 서버 시스템간 상호작용에 이용되는 프로토콜에 어떤 구조가 사용되는지 설명하고, 각 인터페이스가 어떻게 정의되어 있는지를 기술한다.

6.2 REST API

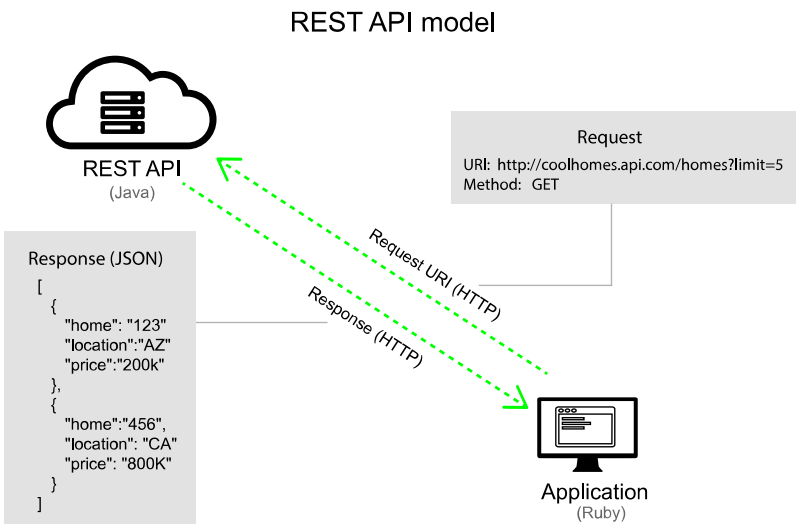


Figure 5: REST API model

본 시스템은 프론트엔드와 백엔드 사이의 통신에 웹 인터페이스, 즉 HTTP 를 이용하며, 요청과 응답 형식은 REST API 에 따른다. REST API 란 Representational State Transfer 의 약자로서, 서버에 저장되어 있는 각 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 API 의 설계 형식을 의미한다. REST API 는 크게 다음 세 부분으로 구성되어 있다.

A. 자원(Resource): URI

- 서버가 보관하고 있는 데이터를 나타내며, 각 자원은 고유한 URI 를 가진다.

B. 행위(Verb): HTTP Method

- 서버의 자원에 접근해 상태를 조작하기 위한 요청 행위로서, 각 조작 행위는 HTTP Method 를 통해 표현된다. (POST, GET, PUT, DELETE)

C. 표현(Representation): JSON

- 클라이언트의 요청에 대한 서버의 응답 형식으로, 주로 JSON 이 사용된다.

클라이언트와 서버 간의 통신에 REST API 를 사용할 경우 서버와 클라이언트 간의 의존성이 줄어들고, 프로토콜이 이해하기 쉬워지는 장점이 있으며, 이를 통해 본 시스템에서 개발하는 프론트엔드 클라이언트뿐 아니라 다른 시스템에서 서버의 자원을 쉽게 이용할 수 있기 때문에 확장성이 높다.

6.3 JSON

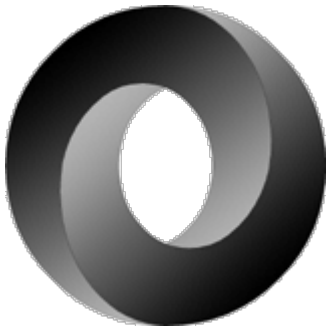


Figure 6: JSON Logo

JSON(**제이슨**[1], JavaScript Object Notation)은 속성-값 쌍(attribute–value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX 가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다.

6.4 Details

본 시스템에서는 Firebase 에서 정의한 프로토콜을 따르기 때문에 이를 생략하고, 사용자들의 트랜잭션 비용을 대납하는 Rest API 에 대해서 기술한다.

A. Send Transaction

- **Fee delegation call**

- Request

Method	POST	
URI	/	
Request Body	id	사용자 ID
	password	사용자 패스워드

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (트랜잭션이 반영되지 않았을 때)	
Success Response Body	success	Transaction Receipt

7. Database Design

7.1 Objectives

요구사항 명세서에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. ER Diagram 을 통해 개괄적인 Entity 간의 관계를 기술하고, Relational Schema, SQL DDL 명세를 만든다.

7.2 ER Diagram

본 시스템에는 User, Item, Bookmark, Authority, Recommend Category, Keyword, Review Reference, Review 로 총 8 개의 Entity 가 존재한다. 각각의 Entity 는 네모 박스의 형태로 표현되고, Entity 간의 관계는 마름모꼴로 표현된다. 이때 특정 Entity 가 다른 Entity 와 복수의 관계를 가질 수 있을 때는 해당 엔티티쪽에 삼지창 모양 선을 세 개 그어 나타내며, 특정 Entity 가 없어도 되는 경우 Client Entity 쪽에 작은 동그라미를 표시한다. 각 Entity 가 가지는 Attribute 는 타원형으로 표현되는데, 각 Entity 를 식별하는 Unique key 는 라벨에 밑줄을 그어 표시하고, 여러 Attribute 를 허용하는 경우에는 테두리를 이중으로 긋는다.

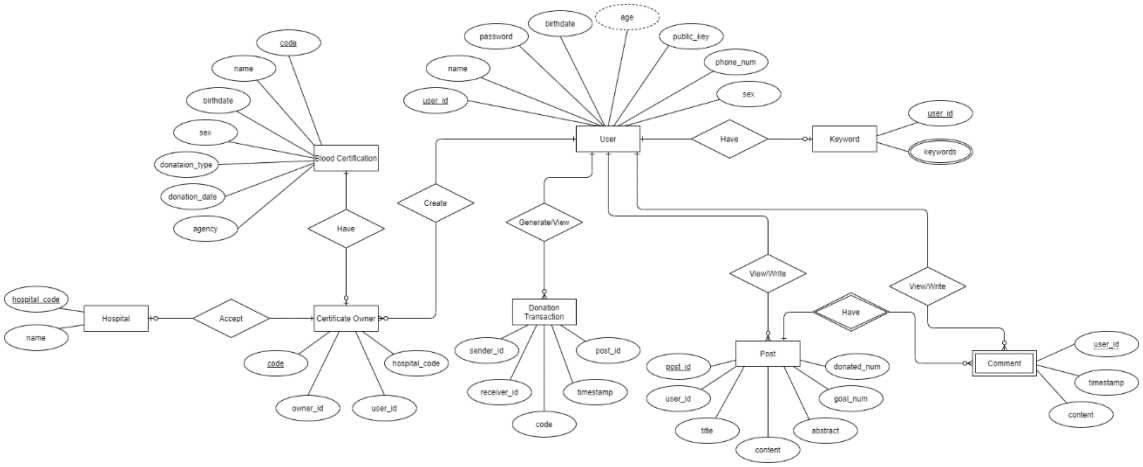


Diagram 26: Overall ER diagram

A. Entities

1. User

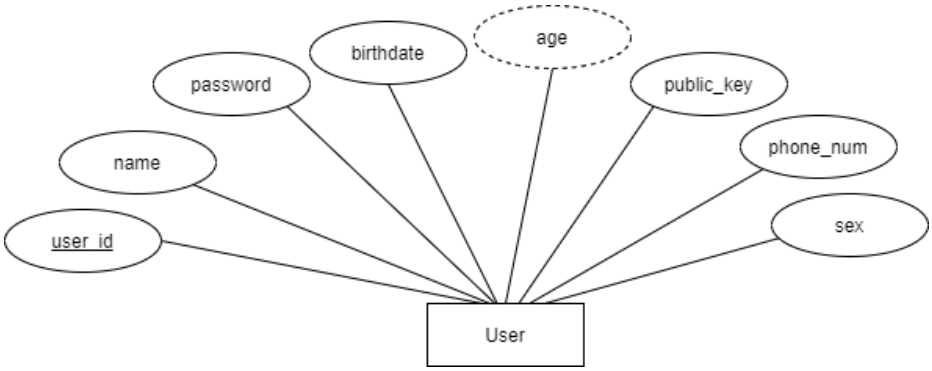


Diagram 27: ER diagram, Entity, User

User Entity 는 사용자의 정보를 표현한다. user_id 속성이 primary key 이며, 헌혈 증서와 대조할 수 있는 기본 정보인 이름, 생년월일, 나이, 성별이 있고 나이는 생년월일을 통해 도출된 어트리뷰트이며 핸드폰 번호, 블록체인 퍼블릭키를 가지고 있다.

2. Keyword

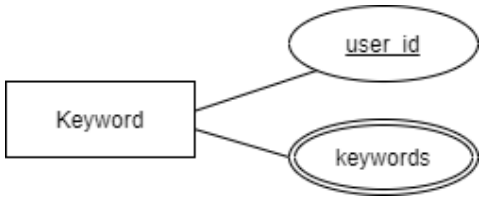


Diagram 28: ER diagram, Entity, Keyword

Keyword Entity 는 각 사용자가 기부 사연에 우선 순위를 두어 보고 싶은 키워드를 표현한다. Primary key 는 user ID 이고 keywords 는 다중 값을 가지고 있다.

3. Post

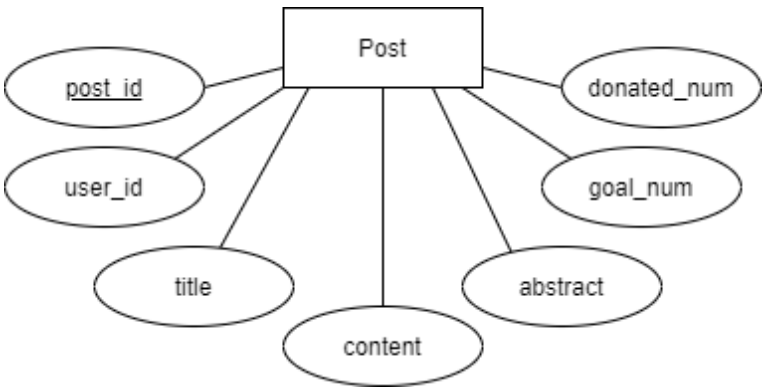


Diagram 29: ER diagram, Entity, Post

Post Entity 는 각 헌혈 증서 기부 요청에 대한 글 객체이다. post ID 가 primary key 이며 해당 post 의 작성자 user ID, 글의 제목, 내용, 요약, 기부 받은 헌혈 증서 수, 목표 수를 가지고 있다.

4. Comment

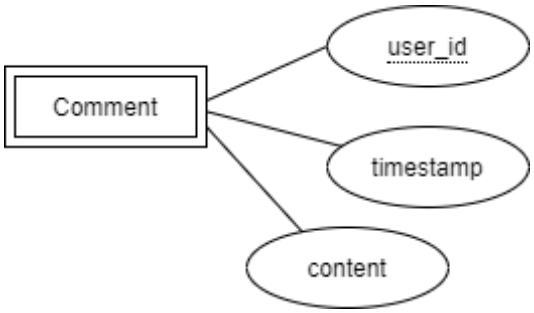


Diagram 30: ER diagram, Entity, Comment

Comment Entity 는 각각의 Post 에 대해서 기부자가 전달한 한 줄의 응원의 말 객체이다. Post 엔티티에 의존하는 약간 개체이며 post ID 와 user ID 를 조합하여 primary key 를 설정하였다. 각 comment 객체는 작성 시각, 내용을 가진다.

5. Donation Transaction

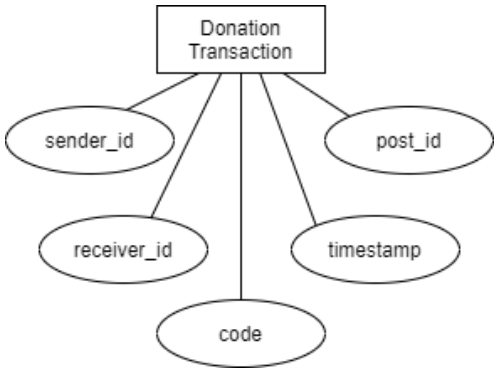


Diagram 31: ER diagram, Entity, Donation Transaction

Donation Transaction 은 블록체인에 올라가는 객체이고, primary key 는 sender ID 와 헌혈 증서 코드를 조합하여 정의한다. 이 객체에는 기부자 ID, 기부 받는 사람의 ID, 헌혈증 코드, 트랜잭션 발생 시각, post ID 정보를 가지고 있다.

6. Certificate Owner

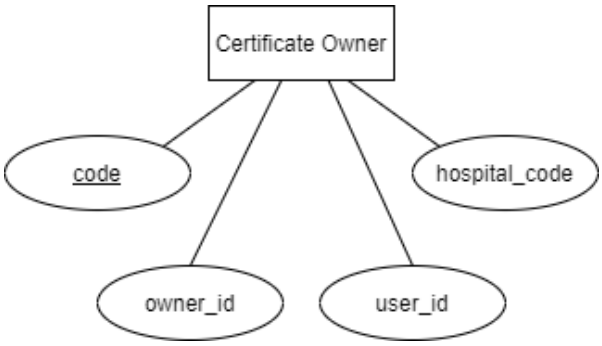


Diagram 32: ER diagram, Entity, Keyword

Certificate Owner Entity 는 해당 헌혈 증서의 소유권을 나타내는 객체이다. Primary key 는 헌혈 증서의 코드이고, owner 는 해당 헌혈 증서를 기부 받은 사람, user 는 해당 헌혈 증서 발급 받은 자이다. 그리고 Hospital code 는 어떤 병원에서 해당 헌혈 증서가 처리되었는지를 나타낸다.

7. Blood Certificate

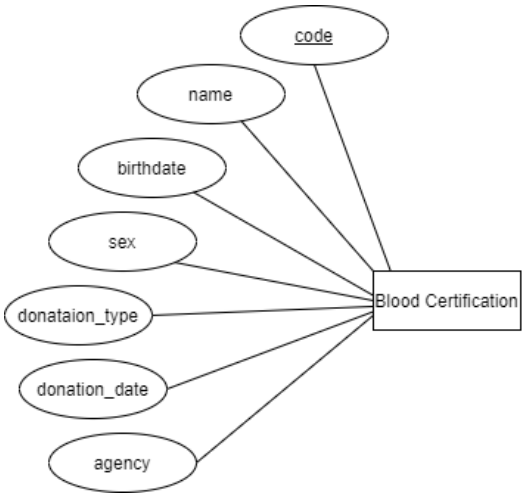


Diagram 33: ER diagram, Entity, Blood Certificate

Blood Certificate Entity 는 헌혈 증서 발급 관련한 엔티티이다. Primary key 는 헌혈 증서의 코드이고 기타 정보로는 헌혈 증서에서 확인 가능한 이름, 생년월일, 성별, 헌혈 종류, 헌혈 일자, 혈액원명이다. 이 객체를 통해 사용자가 등록하는 헌혈 증서의 유효성을 판단한다.

8. Hospital

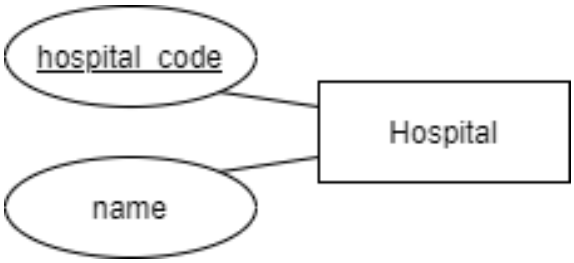
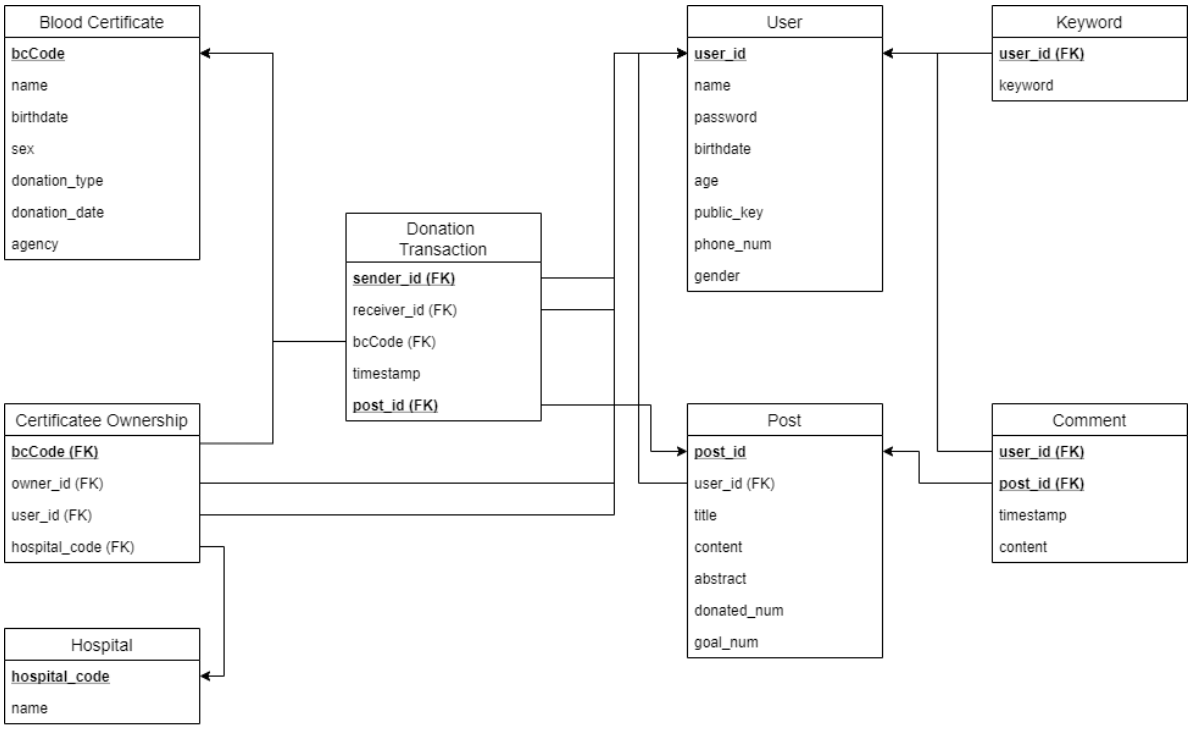


Diagram 34: ER diagram, Entity, Hospital

Hospital Entity 는 병원 엔티티이다. Primary key 는 병원마다 부여된 고유 코드이고 기타 정보로는 병원의 이름이 있다. 이 엔티티를 통해 헌혈 증서가 어떤 병원에서 접수되었는지 정보를 제공한다.

B. Relations

7.3 Relational Schema



8. Testing Plan

8.1 Objectives

의도한 방향대로 실행되고 시스템 내부의 결함을 찾기 위해 testing 을 한다. 이를 위해 설계단계에 미리 계획한다. Testing Plan 에서는 Testing Policy 와 여러 Test Case 에 대해 기술한다.

8.2 Testing Policy

A. Development Testing

1) security

사용자의 개인 정보를 회원가입을 통해 서버 데이터베이스에 저장하고 관리하기 때문에 데이터베이스는 보안적으로 안전해야 한다. 이를 위해 민감한 정보는 해싱 함수를 통해 데이터베이스에 가공되지 않은 데이터 그대로를 올리지 않고 SHA 256 과 같은 해싱 함수를 통해 데이터 보안 이슈를 해결할 수 있다.

또한 블록체인 트랜잭션 발생 시, 사용자의 private key 를 통한 서명이 필요하다. 이 private key 의 유출은 헌혈 증서 양도와 관련된 중요한 트랜잭션이 해킹 당할 수도 있어 private key 는 public key 와 다르게 서버 데이터베이스에서조차 관리하지 않는다. 따라서 로컬 디바이스 스토리지에 암호화되어 관리되어야 하고 이러한 보안 이슈는 해킹 시도를 통해 해결하고 보안 능력을 확인할 수 있다.

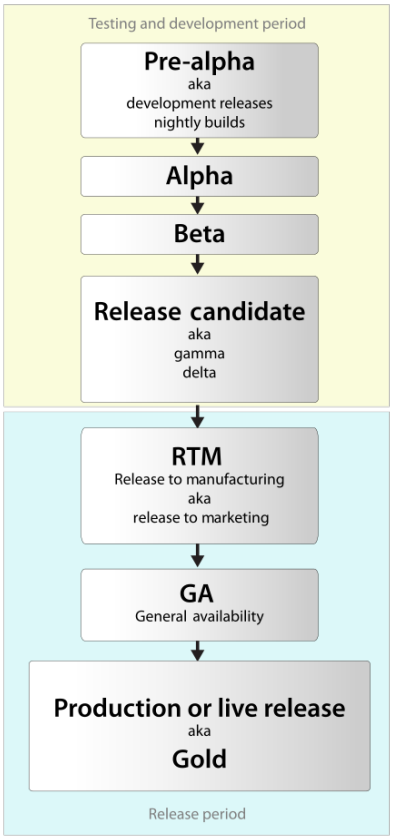
2) Performance

본 시스템에서 가장 큰 성능 병목 지점은 블록체인에 트랜잭션을 전송하고 조회하는 지점이다. 블록체인 네트워크에 트랜잭션을 반영하는 것은 블록 생성 주기와 관련이

있어 일반 서버에 반영하는 것과는 시간적으로 성능이 뒤쳐질 수 밖에 없다. 그래서 최소한의 블록체인 네트워크 접속을 위해 메모리 상에 Java Singleton 기법을 통해 메모리 데이터베이스 기능을 구현하여 Latency 를 줄일 필요가 있다. 이에 관한 성능 평가는 어플리케이션 기능 별 시간 측정을 통해 테스트하고 개선해 나갈 수 있을 것이다.

B. Release Testing

Release testing is a process to test the newer version /build of a software/ application to make sure that is flawless and doesn`t have any issues and works as intended. It has to be done prior to release.



Like above testing and development period, we should test it by alpha version first by developer and then release beta version to proceed final checking. By beta version testing, we can get feedback from real users and prolong the life cycle of software by this process.

C. User Testing

User Testing 은 사용성에 대한 평가로 이해할 수 있다. 본 시스템을 평가하기 위하여 헌혈 증서 등록, 헌혈 증서 기부, 헌혈 증서 제출과 같은 일련의 과정을 수행하여야 한다. 이를 테스트하기 위하여 헌혈 증서 발급 데이터베이스의 정보 가정하고 전체 유저 사용 시나리오를 확인해야 한다.

D. Testing Case

테스트 케이스는 위에서 언급한 바와 같이 헌혈 증서 발급 정보를 가정하고 1) 헌혈 증서 등록, 2) 헌혈 증서 기부, 3) 헌혈 증서 기부 요청 글 작성, 4) 기부 내역 조회, 5) 헌혈 증서 제출 시나리오를 테스트할 것이다.

9. Development Plan

9.1 Objectives

이번 챕터에서는 개발 계획에 대해 서술한다. Gantt chart 를 이용하여 전체적인 개발 수행 계획과 그 일정에 대해 서술한다.

9.2 Frontend Environment

A. Android Studio

Figure 7. Android Studio Logo

Android studio 는 안드로이드 어플리케이션 개발을 위한 공식 통합 개발 환경 (IDE)이고 IntelliJ IDEA 를 기반으로 구현되어 있다. Android studio 를 통해 소스 코드 작성 및 빌드 뿐만 아니라 XML View 를 시각적으로 구현하고 Git 을 통한 버전 관리를 UI 로 직관적으로 제공하고 있다. 본 시스템에서는 안드로이드 스튜디오를 통해 어플리케이션의 전반적인 View 와 사용자 Interaction 을 제공하는 액티비티를 구현하였다.

9.3 Backend Environment

A. Java

Figure 8. Java Logo

Java 는 객체 지향 프로그래밍 언어로서, 대부분의 안드로이드 어플리케이션 개발에 널리 사용되고 있다. 본 시스템에서 안드로이드 어플리케이션 백엔드를 구현하는데 사용했다.

B. Solidity

Figure 9. Solidity Logo

Solidity 란 계약 지향 프로그래밍 언어로 다양한 블록체인 플랫폼에서 스마트 컨트랙트를 구현하는데 널리 사용되고 있다. 이더리움 프로젝트의 일환으로 개발된 언어이지만 클레이튼 네트워크가 이더리움 네트워크의 포크 버전이기 때문에 클레이튼도 공식적으로 솔리디티를 스마트 컨트랙트 언어로 지원한다.

C. Express.js

Figure 10. Express.js Logo

Express.js 는 Node.js 의 핵심 모듈인 HTTP 와 Connect 컴포넌트를 기반으로 하는 오픈소스 웹 프레임워크이다. 가장 인기있는 웹 프레임워크 중 하나이고, 본 프로젝트에서는 Express 를 이용하여 사용자의 Fee delegation call 을 처리할 수 있는 백엔드 서버를 구현하였다.

9.4 Schedule

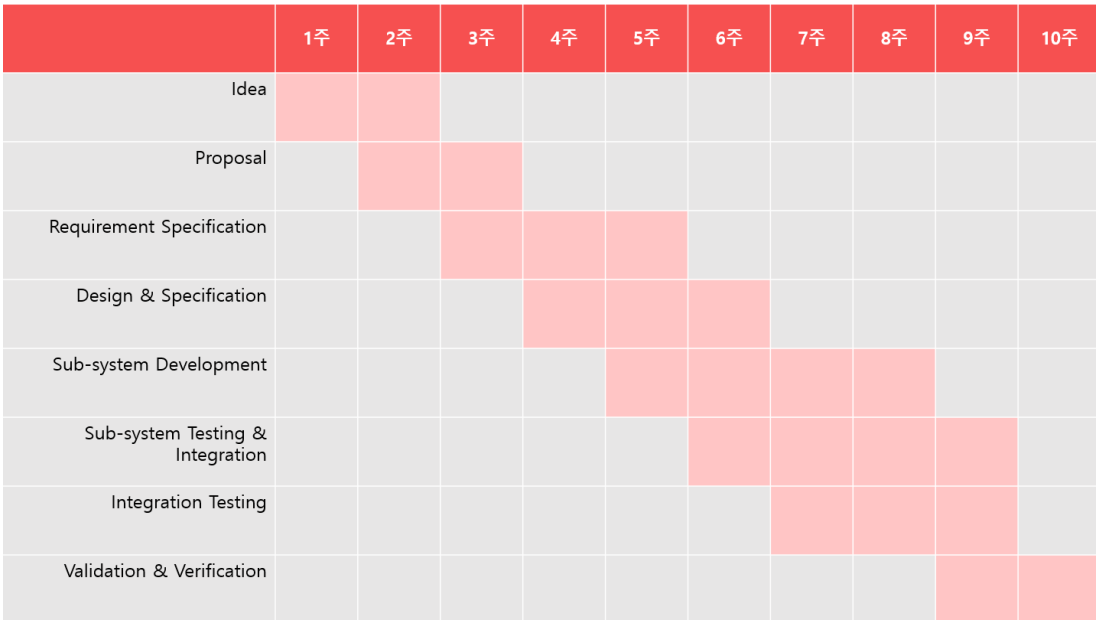


Figure 11. Schedule

문서화 작업이 늦어져 예상 계획보다 지연되고 있다. 구현 단계에서 병렬적으로 개발을 진행하여 시간 단축을 해야 한다.

10. Index

Figures

Diagrams