Software Domain Analysis & Design

-Elaboration Iteration 1-



Kick Share!

Team name: 김말국

201621028 황지훈 (goharrm@ajou.ac.kr)

201621046 용상호 (yonghole@ajou.ac.kr)

201620999 김민창 (rlarudfbf12@ajou.ac.kr)

201723305 안상희 (sanghee0610@ajou.ac.kr)

**Contents**

[1. Vision 4](#_Toc55226788)

[1.2. Positioning 4](#_Toc55226789)

[1.2.2. Problem Statement 4](#_Toc55226790)

[1.2.3. Product Position Statement 5](#_Toc55226791)

[1.3. Stakeholder Descriptions 5](#_Toc55226792)

[1.3.1. User summary 5](#_Toc55226793)

[1.3.2. Non-User summary 5](#_Toc55226794)

[1.3.3. User-Level Goals 5](#_Toc55226795)

[1.3.4 User Environment 5](#_Toc55226796)

[1.4. Product Overview 6](#_Toc55226797)

[1.4.1. Product Perspective 6](#_Toc55226798)

[1.4.2. Assumptions and Dependencies 6](#_Toc55226799)

[1.5. Summary of System Features 7](#_Toc55226800)

[1.6. Other Requirements and Constraints 7](#_Toc55226801)

[2. Requirements 8](#_Toc55226802)

[2.1. Function Requirements 8](#_Toc55226803)

[2.1.1. Use Case Diagram 8](#_Toc55226804)

[2.1.2. Use Case Brief 8](#_Toc55226805)

[2.1.2.1. <Use Case 1> Rent Kickboard 8](#_Toc55226806)

[2.1.2.2. <Use Case 2> Return KickBoard 9](#_Toc55226807)

[2.1.2.3. <Use Case 3> Charge Point 9](#_Toc55226808)

[2.1.2.4. <Use Case 4> Manage Kickboard 9](#_Toc55226809)

[2.1.2.5. <Use Case 5> Kickboard Enrollments 9](#_Toc55226810)

[2.1.2.6. <Use Case 6> Calculate Fare 10](#_Toc55226811)

[2.1.3. Use Case Text 10](#_Toc55226812)

[2.1.3.1. <Use Case 1> Rent KickBoard 10](#_Toc55226813)

[2.1.3.2. <Use Case 2> Return Kickboard 11](#_Toc55226814)

[2.1.3.3. <Use Case 3> Charge Point 13](#_Toc55226815)

[2.1.3.4. <Use Case 4> Manage Kickboard 14](#_Toc55226816)

[2.1.3.4. <Use Case 5> Kickboard Enrollments 16](#_Toc55226817)

[2.1.3.6. <Use Case 6> Calculate Fare 17](#_Toc55226818)

[2.2. Non-Functional Requirements 19](#_Toc55226819)

[3. Analysis Modeling 20](#_Toc55226820)

[3.1. Domain Model Diagram 20](#_Toc55226821)

[3.2. System Sequence Diagram 20](#_Toc55226822)

[3.2.1. <Use Case 1> Rent Kickboard 20](#_Toc55226823)

[3.2.1.1 Domain model 20](#_Toc55226824)

[3.2.1.2. SSD 21](#_Toc55226825)

[3.2.1.3. Operation Contracts 22](#_Toc55226826)

[3.2.2. <Use Case 2> Return Kickboard 22](#_Toc55226827)

[3.2.2.1. Domain Model 22](#_Toc55226828)

[3.2.2.2. SSD 23](#_Toc55226829)

[3.2.2.3. Operation Contracts 23](#_Toc55226830)

[3.2.3. <Use Case 3> Charge Point 24](#_Toc55226831)

[3.2.3.1. Domain Model 24](#_Toc55226832)

[3.2.3.2. SSD 24](#_Toc55226833)

[3.2.3.3. Operation Contracts 25](#_Toc55226834)

[3.2.4. <Use Case 4> Manage KickBoard 25](#_Toc55226835)

[3.2.4.1. Domain Model 25](#_Toc55226836)

[3.2.4.2. SSD 26](#_Toc55226837)

[3.2.4.3. Operation Contracts 26](#_Toc55226838)

[3.2.5. <Use Case 5> KickBoard Enrollments 27](#_Toc55226839)

[3.2.5.1 Domain Model 27](#_Toc55226840)

[3.2.5.2. SSD 27](#_Toc55226841)

[3.2.5.3. Operation Contracts 28](#_Toc55226842)

[3.2.6. <Use Case 6> Calculate Fare 28](#_Toc55226843)

[3.2.6.1. Domain Model 28](#_Toc55226844)

[3.2.6.2. SSD 28](#_Toc55226845)

[3.2.6.3. Operation Contracts 29](#_Toc55226846)

[4. Design Modeling 30](#_Toc55226847)

[4.1. <Use Case 1> Rent Kickboard Realization 30](#_Toc55226848)

[4.1.1. Design Sequence Diagrams 30](#_Toc55226849)

[4.1.1.1. System Operation 1 : check\_userPoint() 30](#_Toc55226850)

[4.1.1.2. System Operation 2 : check\_KickBoard() 31](#_Toc55226851)

[4.1.1.3. System Operation 3 : scan\_QRcode() 32](#_Toc55226852)

[4.1.1.4. Design Class Diagram 33](#_Toc55226853)

[4.2. <Use Case 2> Return Kickboard Realization 33](#_Toc55226854)

[4.2.1. Design Sequence Diagrams 33](#_Toc55226855)

[4.2.1.1. System Operation 1 : scan\_QRcode() 33](#_Toc55226856)

[4.2.1.2 Design Class Diagram 34](#_Toc55226857)

[5. References 35](#_Toc55226858)

[6. Appendix 35](#_Toc55226859)

[6.1. Glossary 35](#_Toc55226860)

[7. Revision History 37](#_Toc55226861)

1. Vision

**1.1. Introduction**

기술 발전에 따라 전기 배터리의 용량이 커지고 사이즈가 작아지면서, ‘개인 이동성’이 중요시되는 시대가 도래하였다. 특히 전동 킥보드는 편리하고 가격도 저렴해 전 세계적으로 인기가 많고 유명하다. 최대 시속 30km까지 주행할 수 있으며, 집에서 편리하게 충전할 수 있다. 또한, 자전거보다 훨씬 빠르고 편리하며 대중교통으로 가기 힘든 목적지까지 더 빠르게 이동할 수 있게 해준다. 공유 또한 우리 사회의 새로운 트렌드가 되면서, 자동차와 자전거 공유 서비스가 일상화되고 있다. 전동 킥보드는 이미 편리하고 효율적이지만, 공유 서비스와 결합하면 더욱 강력하다.

1.2. Positioning

**1.2.1. Business Opportunity**

사람들이 이동 시간의 단축과 효율성을 추구하게 되면서, 빠르고 편리한 이동수단인 전동 킥보드가 각광을 받고 있다. 버스 혹은 지하철을 타고 이동해야 했던 곳을 간편하고 저렴하게 이동할 수 있다는 점에서 사람들은 전동 킥보드를 이용한다. 다만, 목적지에 도착하거나 전동 킥보드를 보관하기 어려운 장소에 가면, 전동 킥보드는 짐이 되기 마련이다. 따라서 사람들은 언제나 편리하게 대여와 반납이 가능한 공유 전동 킥보드 서비스를 선호한다. 공유 킥보드 업체는 증가하는 사용자들의 이용 흐름에 맞춰 킥보드를 효율적으로 수거, 분배하여 이익을 극대화하여야 한다. 이에 따라 사용자들의 흐름을 유동적으로 분석하고 최적의 분배 계획을 업데이트 할 수 있는 시스템이 필요하다.

1.2.2. Problem Statement

학생들은 대부분의 시간을 이동하는데 보낸다. 대부분의 대학들은 면적이 넓기 때문에, 캠퍼스 안에서도 학생들은 수업을 듣기 위해 이동하는데 많은 시간을 보내야 한다. 수업과 수업 사이에 매점에 들리거나 프린트를 하는 것은 대부분의 경우 불가능하다. 혹여 건물과 건물 사이의 거리가 멀 경우, 이동하는 것 만으로도 충분히 지각할 가능성도 존재한다. 공강 시간에 학교 주변에서 밥을 먹거나 어딘가로 가야할 때, 학생들은 항상 서둘러야 한다.

공유 킥보드 업체는 수익을 극대화하기 위해 최대한 많은 사용자에게 킥보드를 이용할 수 있도록 해야 한다. 다만 사용자의 대부분이 불규칙한 패턴을 가지고 이동하거나, 모두 비슷한 방향으로 이동하는 경우 킥보드들은 오랜 시간 방치될 수 있다. 특히 캠퍼스의 경우, 킥보드가 인적이 드문 곳에 반납될 경우, 다음 사용자가 이용하기까지 오랜 시간이 걸려 효율이 떨어진다.

현재 존재하는 킥보드 공유 서비스 앱은 특정 지역을 대상으로 맞춤화 된 서비스를 제공하지 못한다. 특정 지역 내의 인구 이동 흐름과 무관하게 킥보드를 배치, 분배하기 때문에 킥보드의 분포가 고르지 못해 사용자들이 많이 사용할 수 없는 경우가 많으며, 킥보드 사용 효율이 떨어지는 문제가 발생한다.

1.2.3. Product Position Statement

우리의 서비스는 전동 킥보드 공유 서비스 업체와 대학생들에게 매우 유용하다. 공유 업체는 대학교를 목표로 할 수 있다. 캠퍼스는 매 학기마다 학생들이 규칙적인 시간표를 가지고 반복적으로 이동하기 때문에 학생들의 이동이 특정한 패턴을 가지게 된다. 공유 업체는 우리의 시스템을 사용하여 킥보드를 적절한 장소에 분배하고 관리하여 최대한 많은 학생들이 이용하도록 할 수 있다. 요일마다 시간대별로 학생들의 수업 일정은 반복되기 때문에, 학생 이동이 많은 건물을 기준으로 킥보드를 분배할 수 있다. 또한 학생들은 자신의 캠퍼스에 특화되어 관리되는 킥보드 서비스를 이용함으로써, 보다 더 편리하게 공유 킥보드를 사용할 수 있다. 따라서 우리의 서비스는 학교라는 특정 지역의 인구 이동 흐름을 분석하여 그에 맞는 킥보드 공유 계획을 산출해주어, 사용자는 편리한 이용을, 관리자는 효율 및 수익성의 극대화를 기대할 수 있다.

1.3. Stakeholder Descriptions

1.3.1. User summary

- 학생: 역 혹은 버스 정류장에서 학교로 접근이 힘든 사람, 수업 듣는 강의실 간의 거리가 먼 학생, 학교 주변을 이동할 일이 많은 학생.

- 킥보드 매니저: 학교 캠퍼스를 대상으로 공유 킥보드 사업을 원하는 사람, 공유 킥보드 사업의 효율성을 극대화하여 수익 구조를 개선하고 싶은 사람

1.3.2. Non-User summary

- 시스템 관리자: 프로그램의 유지 보수가 가능한 사람, 사용자 정보의 보안을 책임질 수 있는 사람, 알고리즘 설계 및 관리가 가능한 사람

1.3.3. User-Level Goals

- 학생: 면허 등록, 킥보드 대여, 킥보드 반납, 포인트 충전,

- 킥보드 매니저: 알고리즘 계산 결과 확인, 킥보드 상태 확인, 킥보드 위치 정보 접근, 사용자 정보 확인

1.3.4 User Environment

- 시스템은 사용하기 편리해야 한다.

- 모든 사용자는 킥보드의 현재 위치에 접근이 가능해야 한다.

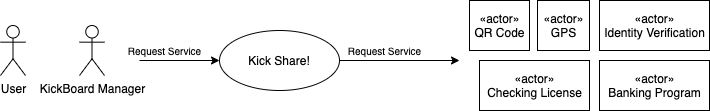
- 사용자의 모바일 기기는 어플리케이션이 설치되기 적합한 환경이야 한다.

- 사용자는 인터넷이 연결된 환경이다.

1.4. Product Overview

1.4.1. Product Perspective

Kick Share! 시스템은 캠퍼스에 최적화된 전동 킥보드 공유 서비스를 통해 학생들이 건물 간 이동시간을 단축하고 편리하게 이동할 수 있도록 한다. 전동 킥보드는 캠퍼스 내부와 사전 협의가 완료된 캠퍼스 주변에서만 사용 가능하고, 전동 킥보드의 보관소는 각 건물의 출입구와 캠퍼스 주변에 설치된다. 학생들은 QR Code를 통해 보관소에서 전동 킥보드를 대여 또는 반납하고, Banking Program을 통한 결제로 전동 킥보드를 이용할 수 있는 포인트를 충전할 수 있다. 또한 Kick Share! 시스템을 외부 시스템과 연동하여 운전면허증의 유효성을 확인하고, 전동 킥보드의 위치를 실시간으로 확인할 수 있다.



<그림1. Kick Share! 시스템 컨텍스트 다이어그램>

1.4.2. Assumptions and Dependencies

- Kick Share! 시스템은 학교 총장님과 시청에 승인이 되어있다.

- 전동 킥보드는 캠퍼스 내부와 사전 협의가 된 캠퍼스 인근에서만 사용 가능하다.

- 이 시스템을 사용하는 User는 모두 운전면허를 갖고 있다.

- 운전면허가 없는 사람은 회원가입을 할 수 없다.

- User는 한 번에 한 개의 전동 킥보드만 대여 가능하다.

- 전동 킥보드의 보관소 위치와 전동 킥보드와 관련된 정보는 데이터베이스에 저장되어있다.

- 모든 전동 킥보드의 보관소와 각각의 전동 킥보드에는 고유한 QR Code가 부여되어 있다.

- 전동 킥보드가 대여 중일 때에는 GPS를 이용하여 실시간으로 전동 킥보드의 위치를 업데이트한다.

- System은 항상 GPS를 이용하여 전동 킥보드의 위치를 실시간으로 추적 가능하다.

- 페널티를 받은 사람은 시스템 이용에 제한이 있다.

- 전동 킥보드는 두 가지 상태(사용 중, 대여 가능)를 가진다.

- 전동 킥보드는 항상 네트워크와 연결되어 있다.

1.5. Summary of System Features

- 회원 가입 시 운전면허증 유효성 확인

- User 정보 관리 (포인트 확인, 킥보드 대여 상태, 페널티 확인)

- Banking Program으로 포인트 충전 및 차감

- 각각의 전동 킥보드는 모두 고유한 QR Code를 가짐

- GPS를 이용하여 전동 킥보드의 위치, 킥보드 보관소의 정보 (남아있는 킥보드의 수와 배터리 상태) 실시간 업데이트

- 전동 킥보드의 사용 패턴과 이용량을 분석하는 시스템 내부의 분배 알고리즘 구현

1.6. Other Requirements and Constraints

- 애플리케이션 UI는 사용하기 쉽고 간단하며, 보기 좋아야 한다.

- 모든 학생들이 운전면허증을 갖고 있는 것은 아니다.

- 전동 킥보드 대여 중에 사고가 발생할 경우, 관련 법안이 제정되어 있지 않기 때문에 보상 기준을 정하기가 어렵다.

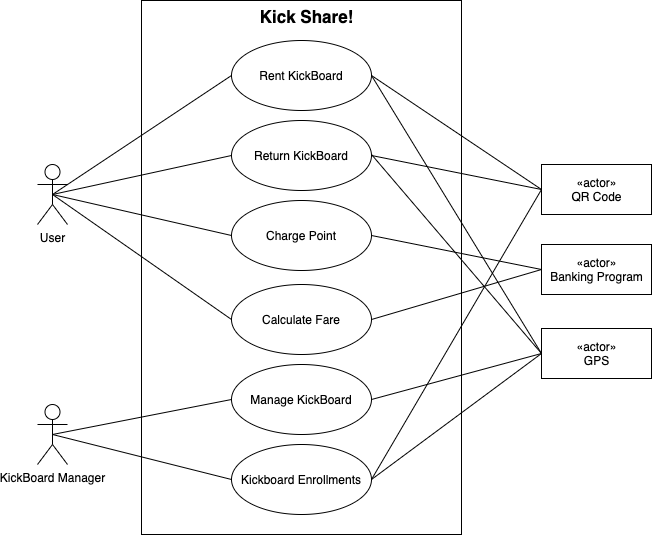
- 이 시스템에 회원 가입한 사람이 회원이 아닌 다른 사람에게 전동 킥보드를 빌려주는 경우, Kick Share! 시스템에서 확인할 방법이 없다.

- 전동 킥보드의 네트워크 통신이 끊긴 상태로 발생한 문제에 대해서는 Kick Share! 시스템에서 책임질 수 없다.

2. Requirements

2.1. Function Requirements

2.1.1. Use Case Diagram

****

<그림2. Use case Diagram>

2.1.2. Use Case Brief

2.1.2.1. <Use Case 1> Rent Kickboard

User가 전동 킥보드를 대여하기 위해서는, Kick Share! 시스템에 회원으로 등록되어 있어야 하고, 킥보드 보관소에는 킥보드가 존재해야 한다. 전동 킥보드 보관소는 학교 내부를 포함한 학교 주변 지역에 설치되어 있으며, User는 보관소에서 전동 킥보드를 대여할 수 있다. 시스템이 User가 사용할 전동 킥보드를 알아야 하므로 User는 전동 킥보드에 부착된 QR코드를 애플리케이션을 통해 인식해야 한다. 시스템에서는 인식된 QR코드로부터 해당 전동 킥보드가 사용 가능한 상태인지 확인한다. 킥보드가 ‘대여 가능’상태라면, 시스템은 User의 계정에 일정 금액 이상의 포인트가 존재하는지 확인하고 일정 금액조차 없다면 User에게 사용 불가 알림과 함께 포인트 충전을 안내한다. User계정에 일정 포인트 이상이 존재한다면 User는 전동 킥보드를 사용할 수 있다. 시스템은 전동 킥보드와 User의 상태를 사용 중으로 바꾸고 User의 사용 위치, 시간 등을 기록한다. 시스템은 사용자의 이용 시간을 기준으로 실시간으로 이용 요금을 계산한다. 만약 이용 요금이 사용자의 잔여 포인트를 초과한다면 시스템은 유저에게 알림을 보내고 유저의 킥보드 이용을 중단시킬 수 있다.

2.1.2.2. <Use Case 2> Return KickBoard

User는 전동 킥보드 보관소 중 아무 곳에서나 반납하여 킥보드의 사용을 종료하는 것이 가능하다. 보관소에서 User는 킥보드의 QR코드를 인식시킨다. 킥보드는 반드시 보관소에서 보관되어야 하므로 시스템에서는 해당 전동 킥보드가 실제로 보관소에 위치하고 있는지 확인한다. 킥보드가 보관소에 위치하고 있다면, 시스템은 현재 시간과 대여를 시작한 시간을 이용하여 User가 지불해야 할 요금(포인트)을 계산한다. 시스템은 User의 계정에서 포인트를 차감하고 킥보드 반납 처리를 완료한다. 반납 처리가 완료되면, 시스템은 User와 킥보드의 상태를 대여 가능으로 변경한다.

2.1.2.3. <Use Case 3> Charge Point

포인트는 현금으로 결제하는 방식으로 진행되며, 이 시스템 내부에서만 사용 가능한 화폐 개념이다. 사용자의 계정에 일정 금액 이상의 포인트가 없다면 사용자는 킥보드를 대여할 수 없다. 따라서 킥보드를 대여하기 전에 최소 금액의 포인트를 충전(결제)해야 한다. 포인트 충전은 외부 결제 대행 서비스를 이용하여 진행된다.

2.1.2.4. <Use Case 4> Manage Kickboard

시스템은 사용자들이 킥보드 사용을 시작한 위치, 사용을 끝낸 위치 등을 시간 별로 기록한다. 그리고 킥보드의 배터리 잔량을 실시간으로 모니터링 한다. 시스템은 기록된 데이터와 현재 킥보드의 위치, 배터리 상태를 고려하여 킥보드 매니저의 킥보드 수거 및 분배 계획 수립 알고리즘을 계산한다. 알고리즘의 계산 결과를 토대로 킥보드 매니저는 킥보드를 수집하며 충전이 필요한 킥보드는 충전을, 사용 가능한 킥보드는 미리 사용자들이 많은 곳으로 분배한다. 만일 사용자가 킥보드를 타고 정해진 구역을 벗어 났을 때는 시스템이 이를 킥보드 매니저에게 알리고, 킥보드 매니저는 해당 유저에게 킥보드 반환을 요청한다. 필요 시, 킥보드 매니저는 시스템을 통해 사용 지역을 벗어난 유저의 킥보드 작동을 중단시킬 수 있다.

2.1.2.5. <Use Case 5> Kickboard Enrollments

새로운 킥보드가 생기는 경우에 대해 킥보드 매니저가 시스템에 킥보드를 추가할 수 있다. 추가된 킥보드는 바로 User가 사용할 수 있고, 유저가 사용한 정보는 시스템에 기록된다.

2.1.2.6. <Use Case 6> Calculate Fare

사용자는 KickBoard를 사용중 일 때, 실시간 사용요금을 확인할 수 있도록 한다. 따라서 사용자는 자신이 point를 더 충전 해야 할 것인지 미리 알 수 있다. 시스템은 1분 간격으로 사용자의 요금을 계산하여 제공한다.

2.1.3. Use Case Text

2.1.3.1. <Use Case 1> Rent KickBoard

|  |
| --- |
| **Use case uc1:** Rent KickBoard  **Scope:** Kick Share!  **Level:** User goal  **Primary Actor:** System User  **Stakeholders and Interests:**  - User: 수업에 늦었을 경우나 다음 수업 건물이 멀리 있는 경우, 학교 주변을 이동할 일이 생긴 경우 킥보드를 이용해 편리하게 이동하고 싶다.  **Preconditions:**  - User는 Kick Share! 애플리케이션을 설치하고 회원가입을 완료한 상태이다.  - User는 로그인이 완료된 상태이다.  - 전동 킥보드 보관소에는 충분히 많은 전동 킥보드가 있다.  - User의 계정에는 최소한의 일정 포인트가 충전되어 있다.  **Postconditions:**  - 지정된 보관소에서 전동 킥보드를 반납할 수 있다.  **Main Success Scenario**  1. User는 애플리케이션을 통해 계정에 잔여 포인트가 있는지, User가 대여하고자 하는 보관소에 대여 가능한 전동 킥보드가 있는지 확인한다.  2. User는 보관소로 이동하여 QR코드를 인식한다.  3. 시스템은 사용자의 계정에 잔액이 충분하며, 해당 전동 킥보드가 사용 가능한 상태인지 확인한다.  4. 시스템은 사용자와 킥보드를 사용중인 상태로 변경한다.  5. 시스템은 이용 시작 시간 정보를 저장하고, 실시간 이용 요금을 계산한다.  **Extension**  1a. 계정에 잔여 포인트가 부족함.  1. 사용자는 대여 서비스 이용을 할 수 없다.  1b. 보관소에 대여 가능한 킥보드가 없음.  1. 사용자는 그 다음으로 가까운 보관소로 이동하여 킥보드를 대여한다.  3a. 사용자의 잔액이 최소 이용 가능 금액 미만이다.  1. 시스템은 사용자의 대여 요청을 거절한다.  2. 시스템은 최소 이용 포인트가 부족하다는 메시지를 띄운다.  3b. 해당 전동 킥보드가 사용 가능한 상태가 아님.  1. User는 다른 킥보드의 QR코드를 인식시킬 수 있다.  5a. 실시간 이용 요금이 사용자의 잔여 포인트를 초과한다.  1. 시스템은 유저에게 포인트 부족 안내 메시지를 띄운다.  2. 시스템은 해당 사용자의 킥보드 이용을 중단시킨다.  **Special Requirements**  - QR코드 시스템을 사용하기 위해서는 외부 API가 필요하다.  - 포인트 결제는 외부 결제 수단을 사용한다.  - QR코드 시스템은 항상 사용 가능해야 한다.  - 킥보드의 GPS정보는 항상 사용 가능해야 한다.  **Technology and Variations list**  - 대여 완료 시 킥보드의 상태정보(사용중/대여가능)를 서버에 업데이트.  **Frequency of Occurrence**  - 시스템은 24시간 운영된다.  - 포인트 충전 과정은 외부 시스템을 사용하므로 점검 시간이 발생할 수 있다.  **Open Issue**  - 믿을 만한 User인지 어떻게 확인할 것인가? |

2.1.3.2. <Use Case 2> Return Kickboard

|  |
| --- |
| **Use case uc2:** Return KickBoard  **Scope:** Kick Share!  **Level:** User goal  **Primary Actor:** System User  **Stakeholders and Interests:**  - User: 지불해야할 킥보드 이용 요금을 알고 싶다. 킥보드 보관소의 위치를 알고 싶다.  **preconditions:**  - 킥보드 대여 과정이 성공적으로 완료되었다.  **postconditions:**  - User의 계정에서 포인트가 차감된다.  - 킥보드의 상태가 대여 가능으로 변경된다.  **Main Success Scenario**  1. User는 목적지 근처의 킥보드 보관소로 간다.  2. User는 보관소에서 킥보드에 있는 QR코드를 인식시킨다.  3. 시스템은 킥보드의 위치가 보관소 중 한 곳이 맞는지 확인한다.  4. 시스템은 QR코드를 인식한 시간 정보를 통해 현재 시간과 시스템에 기록된 대여 시작 시간을 이용하여 이용 시간과 차감 포인트를 계산한다.  5. 시스템은 User의 계정에서 계산된 포인트만큼 차감한다.  6. 시스템은 해당 킥보드를 대여 가능한 상태로 전환시킨다.  **Extension**  3a. 킥보드의 위치가 보관소가 아님.  1. 시스템은 반납 서비스를 거절한다.  2. 시스템은 User에게 반납은 보관소에서만 가능하다는 메시지를 띄움.  3. User는 보관소로 이동하여 반납 절차를 진행할 수 있다.  **Special Requirements**  - QR코드 시스템을 사용하기 위해서는 외부 API가 필요하다.  - 포인트 결제는 외부 결제 수단을 사용한다.  - QR코드 시스템은 항상 사용 가능해야 한다.  - 킥보드의 GPS 정보는 항상 사용 가능해야 한다.  **Technology and Variations list**  - 반납 완료 시 킥보드의 상태정보(사용 중/대여 가능)를 서버에 업데이트.  **Frequency of Occurrence**  - 시스템은 24시간 운영된다.  - 포인트 충전 과정은 외부 시스템을 사용하므로 점검 시간이 발생할 수 있다.  **Open Issue**  - GPS가 킥보드의 잘못된 위치 정보를 제공한다면 어떻게 반납을 진행할 것인가?  - User가 포인트를 충전할 수 없는 상황이라면 어떻게 할 것인가? |

2.1.3.3. <Use Case 3> Charge Point

|  |
| --- |
| **Use case uc3:** Charge Point  **Scope:** Kick Share!  **Level:** User goal  **Primary Actor:** Kickboard Manager  **Stakeholders and Interests**   * User: kickboard를 이용하기 위해 최소의 point를 충전하기를 원한다.   **Pre-conditions:**   * 외부 결제 시스템이 안정적인 상태여야 한다.   **Postconditions:**  - 사용자의 계정에 point가 충전된다.  **Main Success Scenario**   1. 사용자가 포인트 충전을 누르고 충전할 금액을 선택한다. 2. 외부 결제 시스템이 로딩되며 사용자는 결제를 진행한다. 3. 시스템은 결제한 금액 만큼 사용자 계정에 포인트를 충전시킨다.   **Extension**  \*a. 외부 결제 시스템에 문제가 생긴경우  1. 결제를 이용할 수 없거나 다른 결제 시스템을 이용한다.  **Special Requirements**  - 충전 금액은 ‘최소 포인트’이상이어야 한다.  **Technology and Variations list**  - 외부 결제 시스템을 다양하게 수용한다.  **Frequency of Occurrence**  - 사용자가 충전을 원할때, 또는 잔여 포인트가 부족할 때.  **Open Issue**  - 다른 모든 외부 결제 시스템에 오류가 생긴다면 어떻게 해야할 것인가? |

2.1.3.4. <Use Case 4> Manage Kickboard

|  |
| --- |
| **Use case uc4:** Manage Kickboard  **Scope:** Kick Share!  **Level:** User goal  **Primary Actor:** Kickboard Manager  **Stakeholders and Interests**   * User: 킥보드가 필요할 때 항상 근처에 있기를 원한다. 킥보드의 정확한 위치를 알고 싶다. * Kickboard Manager: 학생들의 이동 흐름에 맞춰 킥보드를 분배하여 최대한 많은 학생이 킥보드를 이용할 수 있도록 하고 싶다.   **Pre-conditions:**   * GPS가 연결되어 있어야 한다. * 네트워크가 연결되어 있어야 한다.   **Postconditions:**  - 사용자가 많이 유동하는 곳에 킥보드가 배치된다.  **Main Success Scenario**   1. 시스템은 킥보드의 위치 정보와 학생 이동 흐름을 이용해 위치 분배 알고리즘을 계산한다. 2. 시스템은 킥보드의 수거, 분배가 필요한 경우 킥보드 매니저에게 요청한다. 3. 킥보드 매니저는 해당 알람을 받는다. 4. 킥보드 매니저는 시스템이 도출한 결과를 바탕으로 수거, 분배를 실시한다.   **Extension**  \*a. Sharing Kickboard의 시스템에 계속 문제가 생겼을 경우.  1. 킥보드 매니저는 시스템을 다시 시작할 수 있다.  2. 킥보드 매니저는 모든 킥보드를 수거해 올 수 있다.  1a. 특정 킥보드의 위치가 정해진 장소가 아닌 다른 장소에서 확인된다.  1. 킥보드 매니저는 시스템으로부터 알람을 받아 해당 킥보드를 회수해 올 수 있다.  2. 킥보드 매니저는 해당 킥보드를 사용하고 있는 유저에게 반환 요청을 할 수 있다.  3. 킥보드 매니저는 킥보드를 사용중인 유저의 사용을 중지시킬 수 있다.  1b. 특정 킥보드의 위치가 확인이 안될 경우.  1. 해당 킥보드가 마지막으로 확인된 위치를 확인해 킥보드를 찾는다.  2. 해당 킥보드를 마지막으로 사용한 사용자에게 알람을 전송한다.  4a. 킥보드의 충전이 필요한 경우.   1. 킥보드를 수거하여 충전 후 분배한다   **Special Requirements**  - 킥보드의 위치와 배터리 잔량을 실시간으로 확인할 수 있어야 한다.  **Technology and Variations list**  - 시스템은 알고리즘을 계산하기 위해 주기적으로 DB에 접근해 데이터를 가져온다.  **Frequency of Occurrence**  - 킥보드의 이동 경로가 평소와는 다를 때  **Open Issue**  - 사용자의 위치 정보를 가지고 오는 것이 문제가 되지 않는가?  - 알고리즘의 정확도는 믿을 수 있는가? |

2.1.3.4. <Use Case 5> Kickboard Enrollments

|  |
| --- |
| **Use case uc5:** Kickboard Enrollments  **Scope:** Kick Share!  **Level:** User goal  **Primary Actor:** Kickboard Manager  **Stakeholders and Interests**   * User: 킥보드 수가 부족할 수 있다. 더 많은 킥보드가 필요하다. * Kickboard Manager: 학생들에게 더 많은 킥보드를 지원하고 싶다.   **Pre-conditions:**   * 새로운 킥보드가 1대 이상 있어야 한다.   **Postconditions:**  - 원래 배치 되어 있던 킥보드의 수보다 더 많은 수의 킥보드가 배치 된다.  **Main Success Scenario**   1. 킥보드 매니저는 새로운 킥보드를 시스템에 등록한다. 2. 시스템이 새로운 킥보드에 id를 부여한다. 3. 시스템이 새로 등록된 킥보드의 상태정보(배터리, 위치) 등을 기록한다 .   **Extension**  2 ~ 3a. 킥보드의 등록이 정상적으로 진행되지 않는다.   1. 킥보드 매니저는 시스템을 재시작할 수 있다. 2. 킥보드의 수리를 할 수 있다.   **Special Requirements**  - 킥보드의 위치와 배터리 잔량을 실시간으로 확인할 수 있어야 한다.  **Technology and Variations list**  - 시스템은 현재 킥보드의 정보를 업데이트 하기 위해 주기적으로 DB에 접근한다.  **Frequency of Occurrence**  - 새로운 킥보드가 등록될 때.  **Open Issue**  - 킥보드의 GPS가 정상 작동 하는가?  - 킥보드에 부착된 시스템이 정상 작동 하는가? |

2.1.3.6. <Use Case 6> Calculate Fare

|  |
| --- |
| **Use case uc6: Calculate Fare**  **Scope:** Kick Share!  **Level:** System goal  **Primary Actor:** User  **Stakeholders and Interests**   * User: 실시간으로 이용한 요금을 확인하고 싶다.   **Pre-conditions:**   * 사용자는 성공적으로 킥보드를 대여했다. * 사용자는 자신이 가진 point 내에서 킥보드를 사용했다. * 킥보드가 반납처리를 위해 지정된 station 근처에 위치한다. * 사용자는 킥보드를 그만 이용하고자 한다.   **Postconditions:**  - 사용자의 이용 요금이 계산된다.  - 사용자의 이용 요금이 보유 포인트로부터 차감된다.  - 킥보드가 성공적인 반납을 위한 조건을 갖춘다.  **Main Success Scenario**   1. 시스템이 사용자의 실시간 요금 계산을 요청한다. 2. 시스템이 사용자의 Time info로 부터 이용 시작 시간을 확인한다. 3. 시스템이 이용 시작 시간으로부터 요금 계산 요청 시간까지의 요금을 계산한다. 4. 계산된 요금을 반환한다.   **Extension**  **Special Requirements**  - 사용자의 이용 시작 시간에 접근할 수 있어야 한다.  **Technology and Variations list**  - 시스템은 현재 킥보드의 정보를 업데이트 하기 위해 주기적으로 DB에 접근한다.  **Frequency of Occurrence**  - 새로운 킥보드가 등록될 때.  **Open Issue**  - 킥보드의 GPS가 정상 작동 하는가?  - 킥보드에 부착된 시스템이 정상 작동 하는가? |

2.2. Non-Functional Requirements

**Functionality**

- 오류가 발생할 경우 프로그램은 데이터를 저장하지 않는다.

- 모든 기능을 사용하기 이전에 사용자 인증이 필요하다.

- 앱 사용 중에 오류가 발생하면, 이전 페이지로 다시 로드된다.

**Usability**

- 사용자를 위한 애플리케이션 설계는 명확하고 단순해야 한다.

**Reliability**

- 시스템과 킥보드 사이의 데이터 전달은 정확해야 한다.

- 데이터 전송을 위해 통신 환경이 원활 해야 한다.

- 데이터를 전송하는 동안 끊어짐이 없어야 한다. 데이터가 변경되거나 파괴되는 경우, 데이터 전송을 그만 두어야 한다.

**Performance**

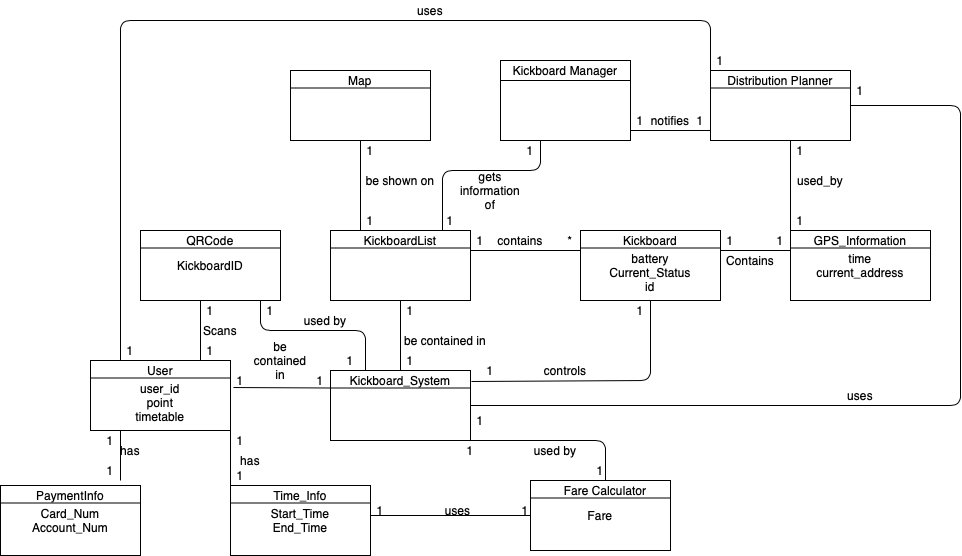
* 유저가 사용하려는 킥보드의 정보는 1초 이내에 응답해 킥보드의 사용을 알린다.
* 킥보드의 위치 정보는 사용자에게 제공 되어야 하므로 비사용중인 킥보드의 위치는 계속해서 기록되고, 위치가 변동될 경우 3초 이내에 서버로 업데이트된다.

**Supportability**

* 이 애플리케이션은 여러 학교에서 사용 가능하다. 자체 학습 알고리즘이 있고, 알고리즘과 데이터베이스는 사용자의 요건에 따라 변경이 가능하다.
* IOS, Android 등과 같이 Kick Share! 애플리케이션에 대해 다양한 모바일 OS를 원하는 고객이 있다. 따라서 애플리케이션은 이러한 요구를 반영해야 하므로 다양한 구동 환경의 분석이 필요하다.

3. Analysis Modeling

3.1. Domain Model Diagram

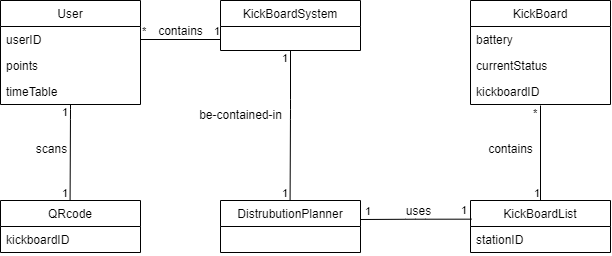


<그림 3. Domain Model Diagram>

3.2. System Sequence Diagram

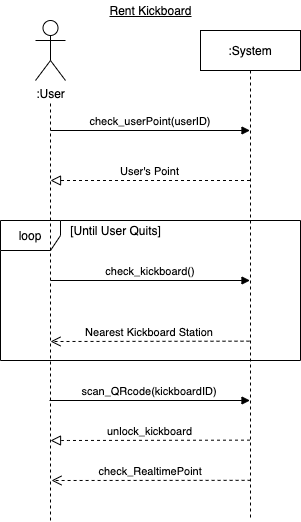
3.2.1. <Use Case 1> Rent Kickboard

3.2.1.1 Domain model

****

<그림 4. Use Case 1 - Domain Model>

3.2.1.2. SSD

****

<그림 5. Use Case 1 – System Sequence Diagram>

**System Operation List**

- check\_userPoint(userID) : 사용자가 자신이 가지고 있는 포인트를 확인하기 위해 해당 메뉴를 선택하고 시스템은 확인을 요청한 사용자 계정의 잔여 포인트를 반환한다.

- check\_kickboard() : 사용자가 Map interface를 통해 자신의 주변에 사용 가능한 킥보드의 위치를 확인한다.

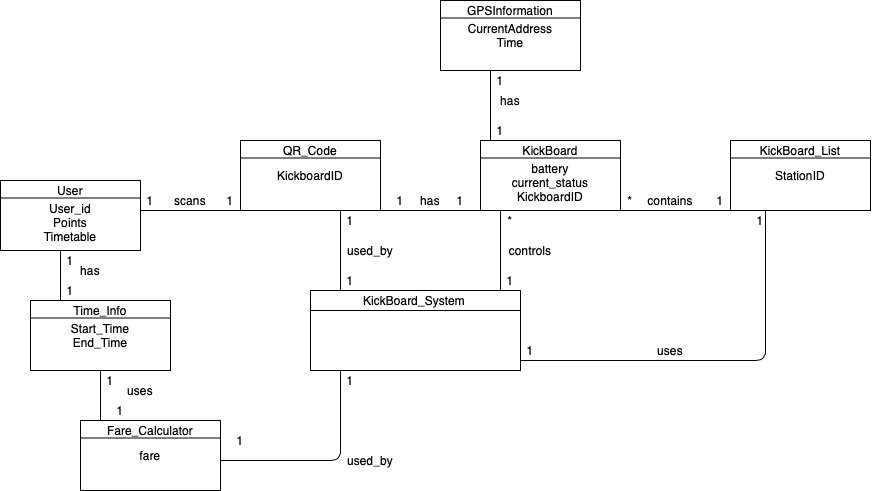
- scan\_QRcode(kickboardID) : 사용자가 킥보드에 부착되어 있는 QR code를 스캔한다. 시스템이 사용자의 대여 자격을 확인하고 사용자에게 킥보드를 대여해주며, 해당 킥보드는 대여 불가능 상태로 변경한다. 동시에 사용자의 실시간 요금 계산을 시작한다.

3.2.1.3. Operation Contracts

|  |
| --- |
| * Operation : checkUserPoint(userID) * Cross References : Rent Kickboard * Precondition : None * Postcondition : User의 Point가 체크된다. |
| * Operation : checkKickboard() * Cross References : Rent Kickboard * Precondition : KickboardDescription의 instance location이 있음 * Postcondition : KickboardDescription의 instance location이 MapInterface와 association이 형성된다. |
| * Operation : scanQRCode(userID, kickboardID) * Cross References : Rent Kickboard * Precondition : 사용 가능한 Kickboard가 현재 station에 존재한다. * Postcondition :   + Time Info instance tii가 생성된다.   + tii.start\_time이 현재 시간으로 초기화된다.   + 현재 User instance와 tii 간의 association이 형성된다.   + 현재 KickBoard의 currentStatus가 ‘사용중’으로 변경된다.   + GPSInformation instance가 생성되며 현재 KickBoard의 실시간 위치 정보가 GPSInformation instance의 attribute에 갱신된다. |

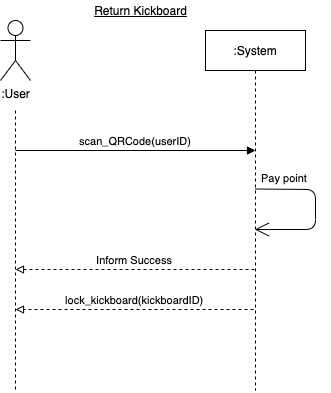
3.2.2. <Use Case 2> Return Kickboard

3.2.2.1. Domain Model

****

<그림 6. Use Case 2 - Domain Model>

3.2.2.2. SSD



<그림 7. Use Case 2 – System Sequence Diagram>

**System Operation List**

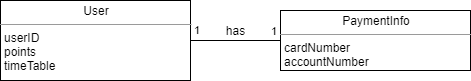
- scan\_QRcode(userId) : 사용자가 킥보드를 반납하기 위해 QR code를 스캔한다. 시스템이 대여를 요청한 사용자를 확인하고, 킥보드의 위치와 이용 시간 등을 확인하여 사용자의 계정에서 포인트를 차감한다. 그리고 사용자에게 포인트가 제대로 지불되었다고 안내하고, 해당 킥보드를 대여 가능 상태로 변경한다.

3.2.2.3. Operation Contracts

|  |
| --- |
| * Operation : scan\_QRcode() * Cross References : Return Kickboard * Precondition   + 사용자가 킥보드를 대여한 상태이다.   + 사용자는 요금을 지불하기에 충분한 포인트를 보유하고 있다.   + 사용자가 목적지 근처의 킥보드 보관소에 도착하였다. * Postcondition   + 시스템은 KickBoard의 currentStatus를 ‘대여 가능’으로 변경한다.   + User의 point attribute의 값이 갱신된다.   + GPSInformation과 KickkBoard의 association이 해제되며 GPSInformation의 정보를 시스템이 기록한다.   + KickBoardList가 갱신된 kickboard 객체로 업데이트된다.   + time info instance가 모두 삭제된다. |
| * Operation : free\_kickboard() * Cross References : Return Kickboard * Precondition : User의 point가 차감된 상태이다. * Postcondition : Kickboard Description instance가 삭제된다. |

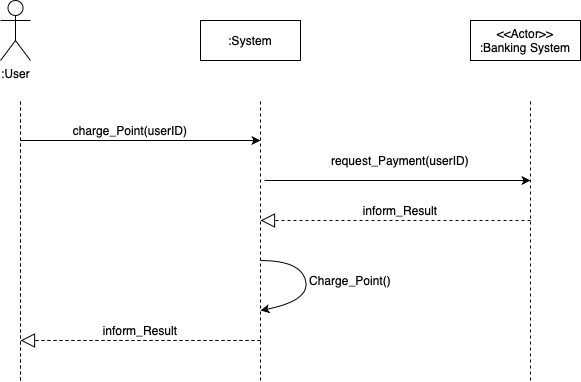
3.2.3. <Use Case 3> Charge Point

3.2.3.1. Domain Model

****

<그림 8. Use Case 3 - Domain Model>

3.2.3.2. SSD



<그림 9. Use Case 3 – System Sequence Diagram>

**System Operation List**

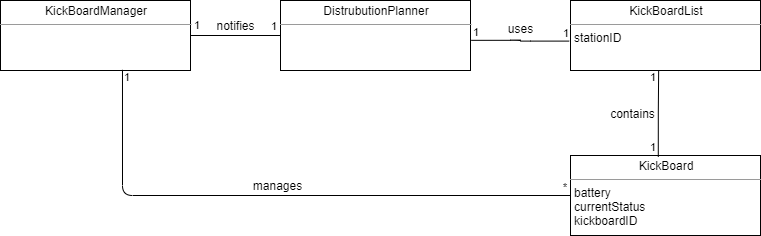
* charge\_Point() : 사용자가 포인트 충전을 요청한다. 시스템이 외부 결제 인터페이스 (banking system)를 활용하여 사용자의 포인트를 충전하고 결과를 반환한다.

3.2.3.3. Operation Contracts

|  |
| --- |
| * Operation : charge\_Point() * Cross References : Charge Point * Precondition :   + 사용자의 올바른 결제 정보가 시스템에 등록되어야 한다.   + 외부 시스템인 banking system이 정상적으로 작동해야 한다. * Postcondition :   + PaymentInfo와 User의 association이 설정된다.   + 사용자의 point attribute가 업데이트 된다. |

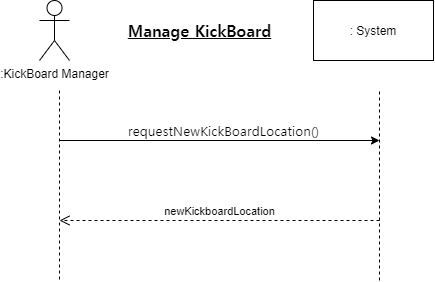
3.2.4. <Use Case 4> Manage KickBoard

3.2.4.1. Domain Model

****

<그림 10. Use Case 4 - Domain Model>

3.2.4.2. SSD



<그림 11. Use Case 4 – System Sequence Diagram>

**System Operation List**

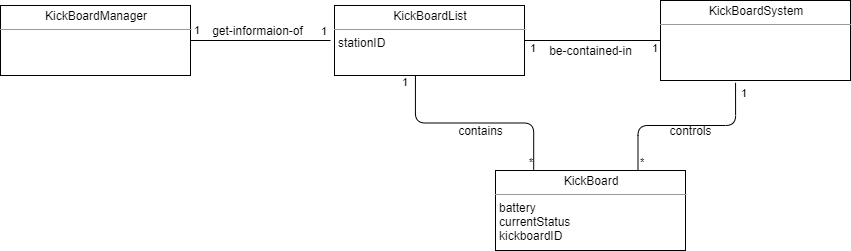
* requestNewKickBoardLocation() : KickBoard Manger가 System에게 킥보드 들을 배치할 새로운 정보를 요청한다. System은 분배 알고리즘을 적용하여 새로운 킥보드 배치 정보를 계산하고 KickBoard Manager에게 반환한다.

3.2.4.3. Operation Contracts

|  |
| --- |
| * Operation : requestNewKickBoardLocation() * Cross References : Manage Kickboard * Precondition :   + 킥보드의 location, current\_state 정보가 시스템에 올바르게 저장되어 있어야 한다. * Postcondition :   + DistributionPlanner가 GPSInformaion을 사용하여 분배 알고리즘을 적용한다.   + 킥보드의 location 값이 시스템에 동기화 된다. |

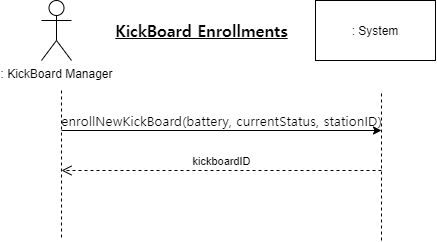
3.2.5. <Use Case 5> KickBoard Enrollments

3.2.5.1 Domain Model

****

<그림 12. Use Case 5 - Domain Model>

3.2.5.2. SSD

****

<그림 13. Use Case 5 – System Sequence Diagram>

**System Operation List**

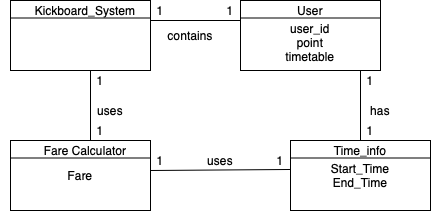
* enrollNewKickBoard(battery, currentStatus, stationID) : 킥보드 매니저는 새로운 킥보드의 배터리, 상태정보 등을 기록 시스템에 등록한다. 시스템이 새로운 킥보드에 id를 부여한다.

3.2.5.3. Operation Contracts

|  |
| --- |
| * Operation : enrollNewKickBoard(battery, currentStatus, stationID) * Cross References : KickBoard Enrollments * Precondition : none * Postcondition :   - 새로운 KickBoard instance newKB가 생성된다.  - newKB의 battery, currentStatus, kickboardID가 설정된다.  - newKB와 KickBoardList간에 association이 형성된다. |

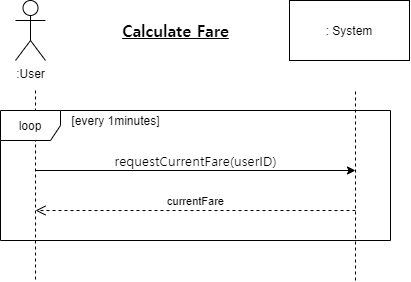
3.2.6. <Use Case 6> Calculate Fare

3.2.6.1. Domain Model

****

<그림 14. Use Case 6 – Domain Model>

3.2.6.2. SSD

****

<그림 15. Use Case 6 – System Sequence Diagram>

**System Operation List**

* requestCurrentFare(userID) : 사용자가 실시간 이용 요금을 알기 위해 1분 간격으로 시스템은 실시간 요금을 계산하여 반환한다.

3.2.6.3. Operation Contracts

|  |
| --- |
| * Operation : requestCurrentFare(userID) * Cross References : Calculate Fare * Precondition :   + User가 사용중인 KickBoard가 사용중인 상태이다.   + User의 startTime이 설정되어 있다. * Postcondition :   + FareCalculator instance의 fare attribute값이 ‘현재 시간 - startTime’에 비례하는 값으로 변경된다. |

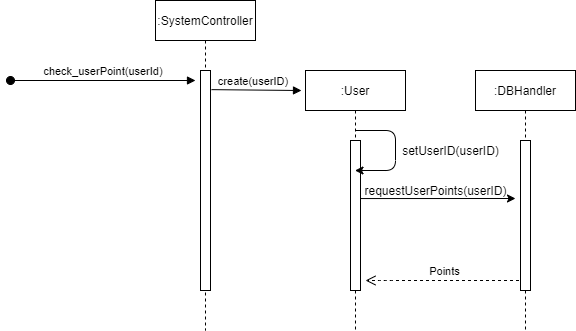
4. Design Modeling

4.1. <Use Case 1> Rent Kickboard Realization

4.1.1. Design Sequence Diagrams

4.1.1.1. System Operation 1 : check\_userPoint()

* check\_UserPoint() : Use Case 1의 system operation중 하나이다. User가 현재 가지고 있는 Point가 얼마인지 알 수 있다.

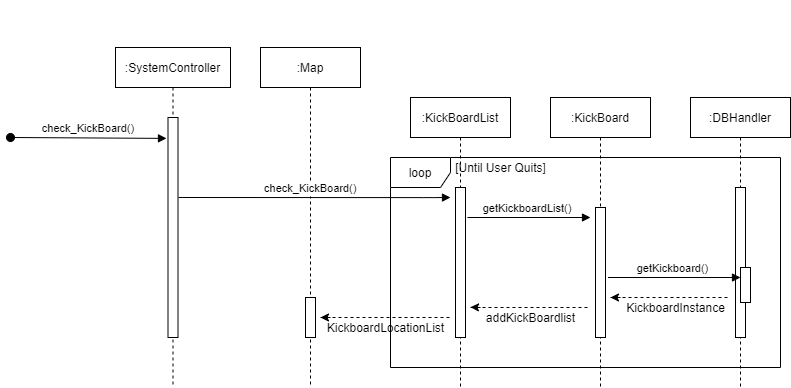
****

<그림 16. check\_userPoint Interaction Diagram>

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| Creator | SystemController가 User를 생성한다. |
| Information Expert | User 객체는 userID를 알고 있는 전문가이다. |
| Low Coupling | SystemController가 직접 DBHandler를 접근하지 않는다. |
| High Cohesion | SystemController는 system operation을 받는 responsibility를 갖는다. |
| Controller | SystemController는 system operation을 받는 responsibility를 갖는다. |

4.1.1.2. System Operation 2 : check\_KickBoard()

* check\_kickBoard() : 현재 사용자의 위치에서 사용 가능한 KickBoard의 위치를 조회한다.

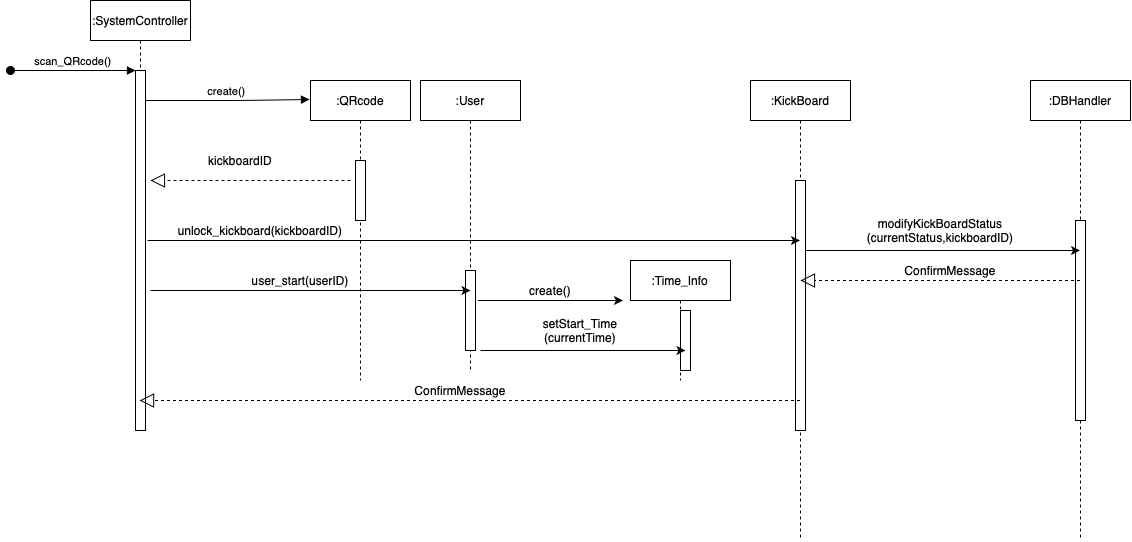
****

<그림 17. check\_KickBoard Interaction Diagram>

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| Creator | 없음 |
| Information Expert | KickBoard의 위치를 모두 알고 있는 KickBoardList가 Map에 List를 보낸다. |
| Low Coupling | KickBoardList는 KickBoard를 목록으로 유지해서 의존도를 줄인다. |
| High Cohesion | SystemController는 system operation을 받는 responsibility를 갖으며 KickBoard관리는 KickBoardList가 담당한다. |
| Controller | SystemController는 System input를 받는다. |

4.1.1.3. System Operation 3 : scan\_QRcode()

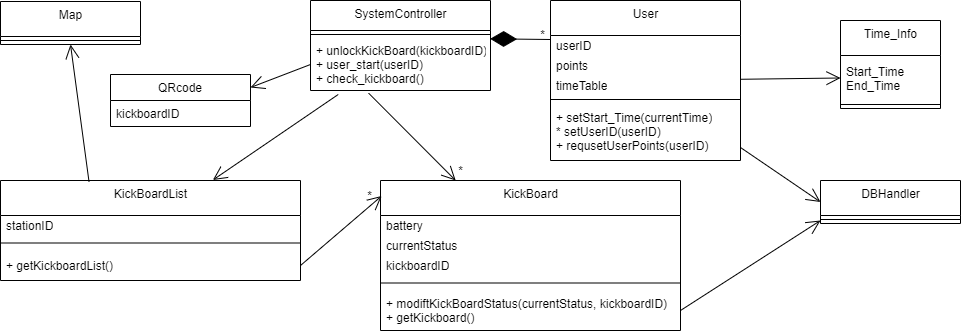
* scan\_QRcode() : 사용자가 KickBoard에 부착된 QRcode를 인식시키면 시스템에게 메시지가 전달된다.



<그림 18. scan\_QRcode Interaction Diagram>

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| Creator | SystemController가 QRcode instance를 생성한다. |
| Information Expert | KickBoard의 정보를 알고 있는 KickBoard 객체가 DB에게 Message를 보낸다. |
| Low Coupling | User는 QRcode에게 kickboard를 unlock할 것을 양도한다. |
| High Cohesion | SystemController는 system input만을 받는 responsibility만을 갖는다. |
| Controller | SystemController는 system input을 받는다. |

4.1.1.4. Design Class Diagram

****

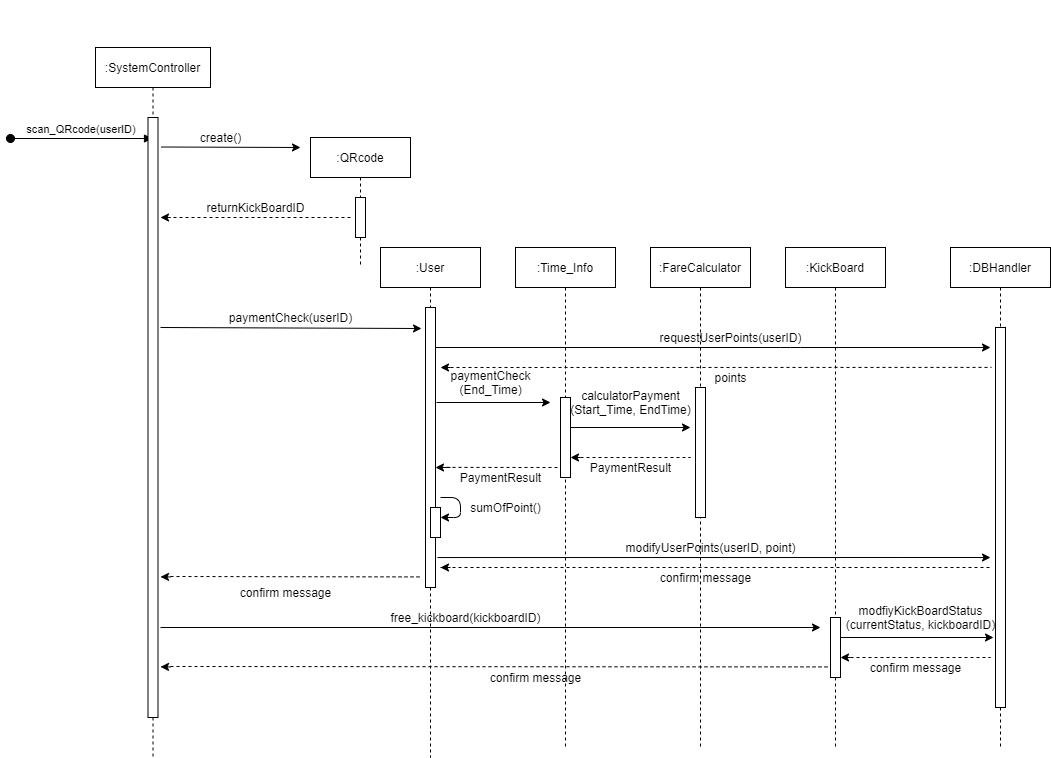
<그림 19. Use Case 1 - Design Class Diagram>

4.2. <Use Case 2> Return Kickboard Realization

4.2.1. Design Sequence Diagrams

4.2.1.1. System Operation 1 : scan\_QRcode()

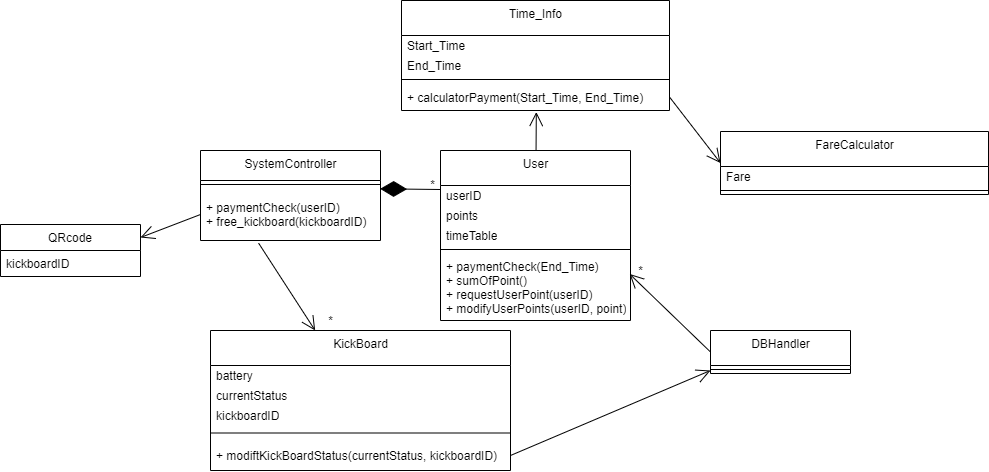
* scan\_QRCode(userID) : Use Case 2에서 가장 중요한 System Operation이다. 사용자가 QR code를 스캔하면 사용자의 포인트를 차감하고 해당 Kickboard를 반납시킨다.

****

<그림 20. scan\_QRcode Interaction Diagram>

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| Creator | SystemController가 QRcode instance를 생성한다. |
| Information Expert | User는 sum of points정보등 사용자에 대한 정보를 갖는다. |
| Low Coupling | 없음. |
| High Cohesion | SystemController는 system input을 받는 responsibility만을 갖는다. |
| Controller | SystemController가 system input을 받는다. |

4.2.1.2 Design Class Diagram



<그림 21. Use Case 2 Design Class Diagram>

5. References

- 플라워로드 사이트, <http://www.flowerroad.ai/>

- 운전면허정보 검증 사이트, <https://dlv.koroad.or.kr/>

6. Appendix

6.1. Glossary

|  |  |  |  |
| --- | --- | --- | --- |
| **용어** | **정의 및 정보** | **형식** | **검증 규칙** |
| User | 전동 킥보드를 사용하는 고객으로, Kick Share! 시스템에 가입이 완료된 사람 |  | 운전 면허증의 유효성 체크 |
| KickBoard Manager | 킥보드와 관련된 모든 것을 처리하는 사람 (킥보드 배치, 킥보드 상태 체크) |  |  |
| QR Code | 킥보드 보관소마다 킥보드 보관소에 관한 정보를 갖고 있는 QR Code가 존재하고, 모든 전동 킥보드는 각자 고유한 QR Code를 갖고 있다. | 흑백 격자 무늬 패턴 |  |
| Banking Program | 외부 결제수단을 이용하여 포인트를 충전할 수 있도록 하는 프로그램 |  |  |
| Checking License | 외부 시스템을 이용하여 운전 면허의 유효성을 체크한다. |  |  |
| Identity Verification | 회원 가입 시, 본인 인증 확인을 위해 필요하다. |  |  |
| 포인트 | Kick Share! 시스템 이용 시, 포인트가 없다면 전동 킥보드를 사용할 수 없다. 포인트는 Banking Program을 통해 충전 가능하다. | 숫자로 구성되어있으며, 한국 돈으로 100원은 100포인트와 같다. |  |
| 분배 알고리즘 | 전동 킥보드의 사용량을 시간, 위치별로 계산하여 전동 킥보드를 적절히 분배할 수 있도록 만드는 알고리즘 |  |  |
| 전동 킥보드 보관소 | 전동 킥보드가 보관되는 장소로, 고유한 QR Code를 가지며 캠퍼스 출입구와 건물 출입구에 설치되어있다. |  |  |
| GPS | 전동 킥보드의 위치를 실시간으로 업데이트 |  |  |
| 대여 가능 | 킥보드의 상태 정보로써 킥보드의 대여가 가능함을 의미한다. |  | QR Code로 대여 |
| 사용 중 | 킥보드의 상태 정보로써 킥보드의 대여가 불가능하며, 사용자가 해당 킥보드를 사용 중임을 나타낸다. |  |  |

7. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** |
| Inception Draft | 2020.09.19 | First Meeting  Brain Storming | Team 김말국 |
| Inception Draft | 2020.09.20 | Second Draft | Team 김말국 |
| Inception Draft | 2020.09.22 | 주제 변경 | Team 김말국 |
| Inception Draft | 2020.09.24 | Inception Report 완성 | Team 김말국 |
| Elaboration 1 Draft | 2020.10.19 | Domain Modeling  First Draft | Team 김말국 |
| Elaboration 1 ver1.0 | 2020.10.25 | Add Mini SSD, OC | Team 김말국 |
| Elaboration 1 ver1.1 | 2020.10.26 | Update SSD, OC | Team 김말국 |
| Elaboration 1 ver1.2 | 2020.10.29 | Update Domain Model | Team 김말국 |
| Elaboration 1 ver1.3 | 2020.10.31 | Update SSD, OC | Team 김말국 |
| Elaboration 1 ver1.4 | 2020.11.01 | Update use cases  Update SSD, OC  Update Interaction Diagram  Add DCD | Team 김말국 |
| Elaboration 1 ver1.5 | 2020.11.02 | Update DCD  Elaboration Iteration 1 완성 | Team 김말국 |