캡스톤디자인

기획/제안서

교수 윤대균교수님

조교 김용준조교님

팀이름 Connection

소프트웨어학과 201320974 박명진

소프트웨어학과 201320987 김범근

소프트웨어학과 201420954 김지선

소프트웨어학과 201520973 김희연

목차

Figure

Table

1. 프로젝트 개요 및 배경

1.1 문제정의

1.1.1 Pain Points

버스는 대중교통 이용객들의 40%가 이용할 정도로 많은 사람들에게 사랑을 받고 있는 대중교통이다. 버스를 많이 이용하는 이유는 많은 목적지를 제공하고, 다른 교통 수단에 비해 가격이 저렴하기 때문이다. 하지만 이와 같은 장점과 동시에 버스는 여러 문제점을 안고 있다.

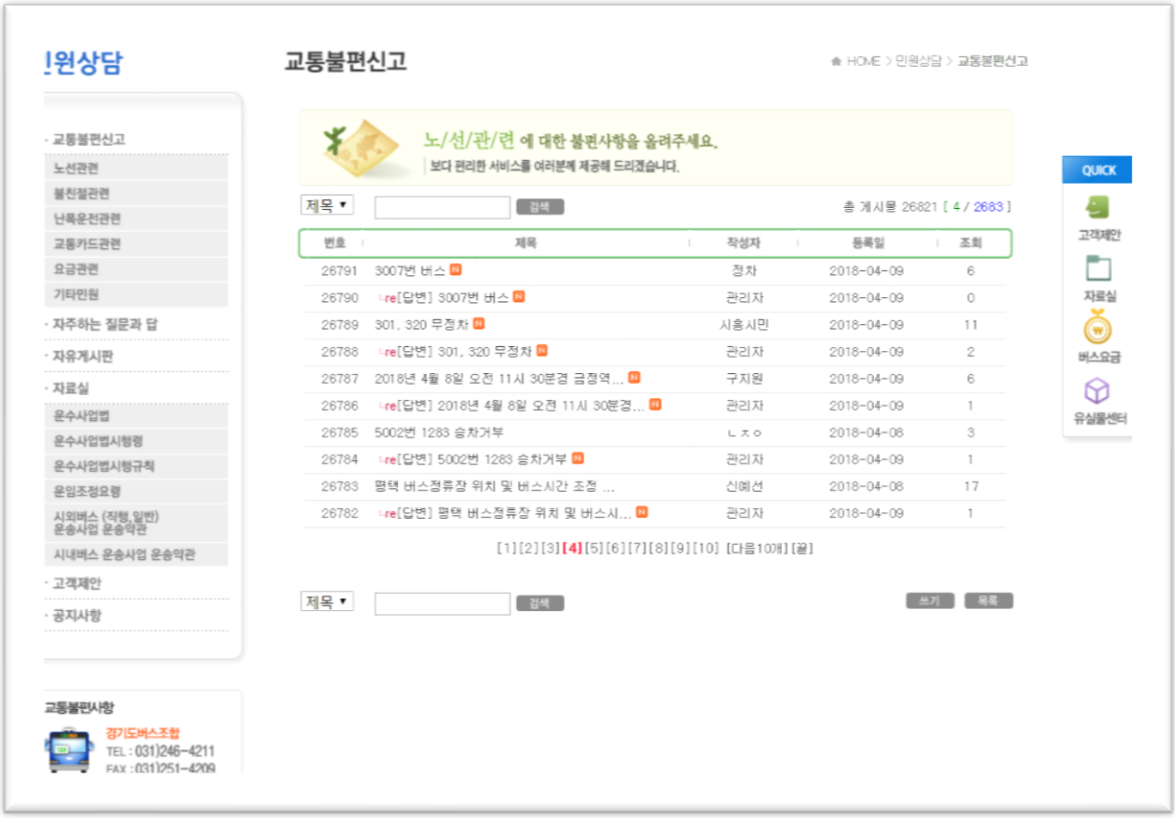

Figure 1 GBUS 사이트 내 민원상담 현황

<Figure 1>은 경기도 버스인 GBUS의 민원 상담 현황이다. 이를 보면 하루에도 4페이지 이상 무정차에 관한 신고가 있음을 알 수 있다. 무정차를 경험했음에도 불구하고 교통 불편 신고를 하지 않는 사람들도 많다는 것을 고려한다면, 많은 사람들이 버스 무정차에 대해서 불편함을 느끼고 있다는 것을 알 수 있다.

무정차 문제 이외에도 승객 본인이 내려야 하는 정류장을 확인하기 위해서 계속 현재 위치를 확인해야 한다는 불편함도 존재한다. 만약 초행길이거나 지리를 잘 파악하지 못하는 승객들은 창문 밖을 확인하더라도 본인이 내려야 할 정류장을 지나치곤 한다.

1.1.2 Solution

위의 문제점을 해결하기 위해 StopBus는 먼저 버스 승차 예약 기능을 제공한다. 이 기능은 정류장의 패널이나 모바일 앱을 통해 승객이 승차 예약을 하게 되면 기사에게 예약 여부를 알려주는 기능이다. 승객이 버스 기사에게 승차 의사를 명확하게 전달할 수 있기 때문에 정류장에 승객이 있음에도 버스가 정류장을 지나치는 일이 줄어든다.

또한, StopBus는 버스 승차 예약 기능과 함께 버스 하차 예약 기능도 제공한다. 하차 예약된 정류장 근처에 왔을 때 승객에게 모바일 앱을 통해 알림을 주는 기능으로, 승객은 내릴 정류장을 놓치는 경우가 줄어들게 된다.

1.2 타켓 및 Stakeholders

1.2.1 버스 승객

버스를 이용할 때, 적지 않은 사람들이 불편함을 호소한다. 버스를 이용하는 것이 익숙하지 않은 승객들은 정차하는 버스마다 원하는 장소로 가는지 묻기 바쁘다. 원하는 버스가 정차하지 않고 가버리는 경우도 있으며, 버스가 여러 대 도착했을 때 다른 버스에 가려져 타지 못하는 경우도 존재한다. 따라서 본 프로젝트를 통해서 승객이 타려는 버스의 기사에게 탑승 의사를 알려주고, 승객이 있음에도 육안으로만 보고 지나가는 경우를 없애고자 한다.

승객은 버스를 탑승하고 난 뒤, 하차를 위해 현재 정류장의 위치를 계속 확인해야만 한다. 버스 내 디스플레이가 고장 난 경우에는 안내방송을 계속 듣고 확인해야하는 번거로움이 있다. 버스에서 잠이라도 든 경우에는 의도치 않게 목적지를 지나쳐서 내리는 경우도 빈번하다. 승객은 버스의 하차 정보만 입력하면, 하차할 정류장에 도착하기까지 승객은 신경을 쓰지 않아도 된다.

1.2.2 버스 기사

버스를 운행하면서 의도치 않은 승객과의 불화가 종종 일어난다. 버스기사들은 승객의 안전과 버스 간 배차간격을 신경 쓰면서 운행을 한다. 그렇기 때문에 실수로 하차를 할 승객이 있음에도 불구하고 정차하지 않고 지나가는 경우가 존재한다. 날이 흐리거나 안개가 많이 낀 날에는 육안으로만 정류장 내 승객을 확인해야하기 때문에 승객을 보지 못하고 정류장을 지나치는 경우도 존재한다. 모든 정류장에 버스가 정차하는 경우, 도로 상황으로 인해 버스 간 간격을 맞추지 못해 승객에게 불평을 듣는 일도 비일비재하다. 본 프로젝트를 통해서 버스기사는 탑승할 승객이 있는 정류장에만 정차를 함으로써 버스 운행의 효율을 증가시킬 수 있다. 더불어 승객의 마찰이 줄어들기 때문에 쾌적한 근무환경을 보장받을 수 있을 것이다.

1.2.3 버스 운수회사

버스 운수회사는 버스들의 운행을 관리하는 입장에서 효율적인 관리시스템을 필요로 한다. 그리고 버스기사들의 운행시간을 조절하여 휴식여건을 보장해주어야 한다.

1.2.4 교통정보센터

대중교통 데이터를 제공하는 센터로 어떻게 하면 승객들이 버스를 편리하게 이용할 수 있을지 생각한다. 매년 버스 데이터를 활용한 이용 편리성 향상과 이용자 중심의 교통정책 아이디어 공모전을 주최해오고 있다. 우리가 제시하는 버스 시스템은 이용자 중심으로 이용자가 타고자 하는 의사를 밝히면 맞춤형으로 서비스를 제공한다. 그렇기 때문에 우리의 프로젝트에 관심을 가질 것이다.

1.3 타켓 및 Stakeholder 관점의 가치

1.3.1 승객의 편리성 증가

일방적인 버스기사의 판단이 아닌 승객의 탑승여부를 버스기사에게 알려줌으로써 승객의 의사가 전달되도록 한다. 버스기사와 승객 사이에 서로의 생각을 예측하는 것이 아닌 제대로 소통함으로써 정확도를 올려 편리함을 증가시킨다. 그리고 기존의 수많은 버스 노선도를 검색해 자신이 원하는 정

보를 단시간에 찾을 수 있도록 검색기능을 제공하여 손쉽게 찾을 수 있도록 한다. 또한 자주 이용하는 버스는 정류장에 도착하기만 해도 알아서 승차 예약이 되기 때문에 따로 신경을 쓰지 않아도 된다.

1.3.2 버스 이용률 증가

버스를 이용하면서 겪은 불편한 경험으로 인해 자가용이나 다른 대중교통을 이용하는 사람들이 많다. 우리의 서비스를 사용하면서 고객의 불편한 부분을 다소 해결할 수 있으며, 전반적인 버스 시스템의 효율성 향상을 통해 다른 교통수단보다 더 적은 시간을 소모하게 될 것이다. 그리고 버스를 승차하고 하차할 때까지 계속 버스의 위치를 확인할 필요가 없기 때문에 버스를 타고 이동하는 시간을 활용할 수 있다는 장점이 있다. 이러한 이점들로 버스의 이용률이 증가할 것이다.

2. 기존 사례분석

2.1 유사 제품 및 서비스

2.1.1 카카오버스

카카오버스는 실시간 교통정보, 잔여좌석, 버스도착알림, 버스하차알림 서비스를 제공한다. 하지만 버스에 대한 정보를 얻기 위해서는 직접 자신이 승차할 정류장을 찾고, 버스를 찾아 선택해야한다. 하차예약을 하기위해서는 여러 단계를 선택해야하기 때문에 불편하다. 그리고 GPS 기반의 서비스라서 부정확하고, 버스기사가 승객이 있는지 모르고 무정차하는 경우에는 탑승하지 못한다는 점은 계속 존재한다.

2.2 관련 기술 동향

2.2.1 원격 하차벨 시스템

버스에서 내리기 전에 모바일 앱을 통하여 원격으로 하차벨을 누를 수 있다. 이 시스템을 사용하더라도 원격으로 하차벨을 누를 뿐 승객이 하차할 정류장을 놓치지 않기 위해 현재 정류장의 위치를 계속 확인해야 하는 불편함을 해결해주지 못한다.

2.2.2 자동 요금 결제 시스템

비콘을 활용한 자동 결제 시스템으로서, 사용자가 스마트 기기를 가지고 버스에 탑승하면 자동으로 버스 요금이 결제되는 시스템이다. 버스에 비콘을 설치하였을 때, 스마트 기기와의 거리를 측정하여 승객이 버스에 승차한 상태인지 혹은 이미 하차한 상태인지를 구분할 수 있다. 비콘과 스마트 기기를 활용하여 서비스를 자동으로 제공하는 좋은 예시이다.

2.3 인사이트 및 차별성

2.3.1 승차벨 시스템

승객이 버스에 탑승을 요할 때 패널과, 앱을 사용하여 버스에 승차 알림을 주는 시스템을 적용하여 무정차 문제를 해결할 수 있다.

2.3.2 정확한 위치 제공

비콘을 활용한 정확한 거리 정보를 토대로 버스에 승차하기 전 후를 나눠서 알맞은 서비스 화면을 제공한다.

2.3.3 자동 하차 시스템

버스 승객이 버스에 탑승 후 내일 정류장을 선택하게 된다면 버스가 그 정류장에 도착하기 3 분 전에 승객에게 알림을 줌과 동시에 하차벨이 자동으로 눌려 승객이 하차벨을 누를 수고를 덜어준다.

3. 개발 내용

3.1 개발 목표

StopBus 는 기존의 버스 운행 시스템을 개선한 시스템으로 비콘 기술을 활용하여 승객들이 편리하게 버스를 이용할 수 있도록 하는 서비스이다. Stop Bus 는 정류장의 패널이나 모바일 앱을 사용하여 특정 버스에 승차예약을 할 수 있는 서비스를 제공한다. 승차예약이 완료되면 버스기사에게 승객의 승차 의사가 명확하게 전달되기 때문에 정류장에 버스를 탈 승객이 있음에도 불구하고 버스가 정차를 하지 않는 문제를 개선할 수 있다.

StopBus 는 승차예약 뿐만 아니라 모바일 앱을 이용하여 하차예약을 할 수 있는 서비스도 제공한다. 승객은 모바일 앱을 통해 원하는 정류장에 하차 예약을 할 수 있다. 하차 예약이 된 정류장 근처에 왔을 때 승객은 모바일 앱을 통해서 하차 알림을 받음과 동시에 버스에서는 자동으로 하차벨이 눌린다. 따라서 승객이 버스를 타는 동안 정류장의 위치를 계속 확인하지 않아도 되는 장점이 있다.

3.2 Use Cases

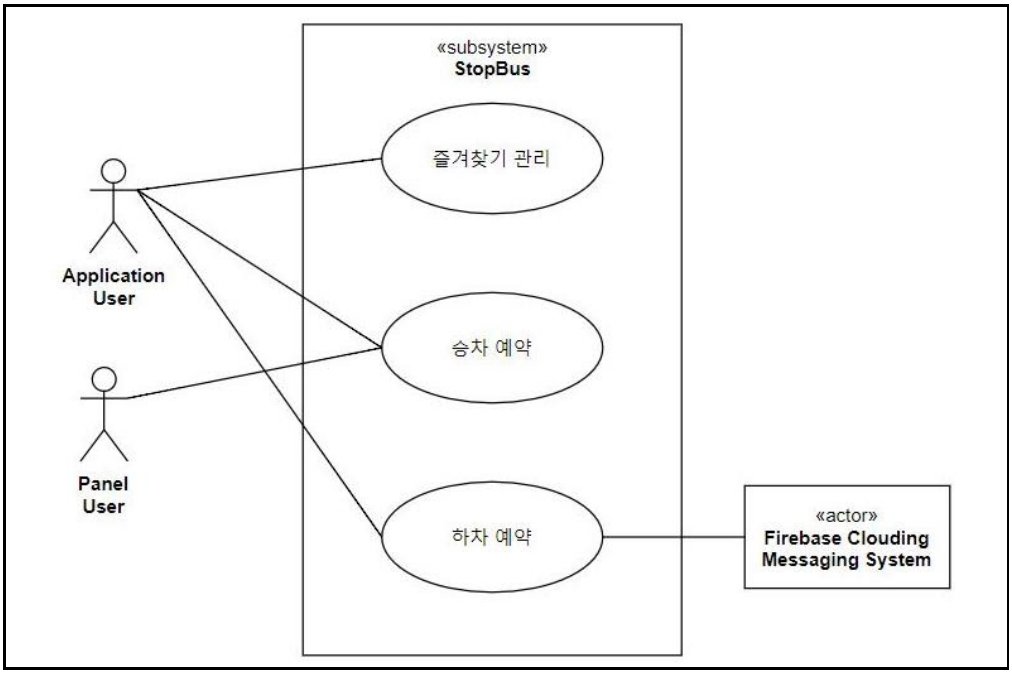

Figure 2 <StopBus> Use Case

3.2.1 Use Case 1

Use Case 1	
Use Case Title	즐겨찾기 관리
Primary Actor	Application User
Text Description	1. 사용자가 즐겨찾기에 버스를 추가할 수 있다. 2. 사용자가 즐겨찾기의 버스를 삭제할 수 있다. 3. 사용자가 즐겨찾기 내 바로예약상태를 변경할 수 있다.
Precondition	없음
Postcondition	즐겨찾기 목록에 버스 항목이 추가되거나 삭제된다.
Main Success Scenario	1. 사용자는 즐겨찾기 목록 메뉴를 선택한다. 2. StopBus 는 즐겨찾기 목록을 보여준다. 3. 사용자는 검색 칸에 버스 번호를 입력한다. 4. StopBus 는 검색한 결과를 보여준다. 5. 사용자는 추가할 버스번호를 선택한다. 6. StopBus 는 확인 메세지를 보여준다. 7. 사용자는 추가를 선택한다. 8. StopBus 는 추가된 즐겨찾기 목록을 보여준다.
Extensions	3a. 사용자는 즐겨찾기 내 버스의 바로예약상태를 변경한다. 1. StopBus 는 상태변경에 대한 확인 메세지를 보여준다. 2. 사용자는 확인을 선택한다. 3. StopBus 는 변경된 즐겨찾기 목록을 보여준다. 3b. 사용자는 검색 칸에 정류장 이름을 입력한다. 3c. 사용자는 검색 칸에 정류장 번호(5 자리)를 입력한다. 5a. 취소버튼을 누른다. 1. StopBus 는 즐겨찾기 목록을 보여준다.

Table 1 Use Case 1

3.2.2 Use Case 2

Use Case 2	
Use Case Title	승차예약
Primary Actor	Application User, Panel User
Text Description	사용자가 자신이 원하는 버스의 승차벨을 눌러 승차예약을 할 수 있다.
Precondition	Application User 의 경우 정류장 근처에 있어야 한다.

Postcondition	승차벨이 눌린 버스에게 정류장에 승차 승객이 있다고 알림이 가도록 설정한다.
Main Success Scenario	1. User 는 정류장 근처에서 application 을 켠다. 2. StopBus 는 User 근처에 있는 정류장의 버스 목록을 보여준다. 3. 사용자는 자신이 승차예약을 하고자 하는 버스를 선택한다. 4. StopBus 는 승차예약 정보가 담긴 확인 메시지를 보여준다. 5. 사용자가 확인 버튼을 누른다. 6. StopBus 는 승차 예약을 한 버스가 정류장에 진입시 알림을 준다.
Extensions	1a. Panel User 의 경우 1. User 는 정류장의 Panel 을 확인한다. 2. StopBus 는 유저에게 정류장의 버스 목록을 보여준다. 3. User 는 자신이 승차예약을 하고자 하는 버스를 선택한다. 4. StopBus 는 승차 예약된 버스가 진입하는 경우 음성 알림을 준다. 2a. 사용자가 정류장을 인식하지 못하였을 때 1. StopBus 는 유저에게 즐겨찾기 화면을 보여준다. 5a. 사용자가 취소 버튼을 누른다. 1. StopBus 는 유저에게 정류장의 버스 목록을 보여준다.

Table 2 Use Case 2

3.2.3 Use Case 3

Use Case 3	
Use Case Title	하차 예약
Primary Actor	Application User
Text Description	사용자는 자신이 탑승한 버스의 노선도에서 하차하고자 하는 정류장을 선택하여 하차 예약을 할 수 있다.
Precondition	사용자가 버스에 탄 상태여야 한다.
Postcondition	하차할 정류장의 정차 알림이 활성화된다.
Main Success Scenario	1. StopBus 는 사용자가 탄 버스의 노선도를 보여준다. 2. 사용자가 노선도에서 하차할 정류장을 선택한다. 3. StopBus 는 예약 여부를 물어본다 4. 사용자가 확인 버튼을 누른다 5. StopBus 는 예약정보를 보여주는 확인 메시지를 보여준다.
Extensions	1a. 사용자가 탄 버스를 인식하지 못하는 경우 1. StopBus 는 유저에게 즐겨찾기 화면을 보여준다. 4a. 사용자가 취소 버튼을 누른다 1. StopBus 는 유저에게 사용자가 탄 버스의 노선도를 보여준다.

Table 3 Use Case 3

3.3 Features

3.3.1 Panel user

Features	Description
승차 예약	정류장에 설치된 패널에는 '도착하는 버스'나 '버스 정보 조회'에서 버스 목록이 표시된다. 버스 목록 중 탑승을 원하는 버스 번호를 누르면 승차 예약이 완료된다
선착순 도착 버스 정보	패널의 '도착하는 버스'에서 가장 먼저 오는 순서대로 버스 정보가 표시된다. 버스 번호와 몇 번째 전인지, 그리고 남은 시간이 표시된다.
전체 버스 정보 조회	패널의 '버스 정보 조회'에서 정류장을 지나가는 버스들의 정보를 모두 조회할 수 있다. 버스 번호의 오름차순으로 버스의 정보가 표시되며, 몇번째 전인지를 알 수 있다. 화면에 담기지 않은 정보는 '버스정보 조회'의 좌우 버튼을 통해 조회할 수 있다.
버스 정보 검색	패널의 '버스정보 검색'에서는 버스 번호나 정류장 이름을 직접 입력하여 버스 정보를 검색할 수 있다.

Table 4 Panel user features

3.3.2 Application user

Features	Description
정류장 정보 표시	정류장 근처에 도착하면 모바일 어플 화면의 상단에 정류장 번호, 정류장 이름, 방면이 표시된다. 또한 해당 정류장을 지나가는 버스 목록이 표시된다.
버스 승차 예약	해당 정류장을 지나가는 버스 목록 중 탑승을 원하는 버스 번호를 누르면 승차 예약이 완료된다.
버스 도착 알림	정류장 근처에 승차 예약한 버스가 접근하는 경우, 어플에 알림을 준다.
버스/ 정류장 검색	승객이 버스나 정류장을 검색 할 수 있다.
즐겨찾기 관리	승객이 자주 탑승하는 버스를 지정하여 즐겨찾기 목록에 등록하거나 목록에서 제거할 수 있다.
즐겨찾기를 통한 승차 예약	정류장 근처에 도착하였을 때 모바일 어플에서는 승차예약이 가능한 화면이 나타나고, 즐겨찾기로 등록된 버스 번호를 우선으로 승차 예약 여부를 묻는다. 탑승을 원하는 버스 번호를 누르면 승차 예약이 완료된다.
정류장 하차 예약	버스 승차 시 모바일 어플에서는 탑승한 버스의 노선도가 표시된다. 노선도에서 내리고자 하는 정류장을 선택하면 하차 예약이 완료된다.

하차 알림 및 자동 하차벨 눌림	버스가 하차 예정인 정류장 근처에 접근하면 모바일 어플에서는 알림이 뜨게 되어 승객에게 하차를 준비해야 함을 알려준다. 이와 동시에 버스에서는 자동으로 하차벨이 눌린다.

Table 5 Application user features

3.3.3 Application driver

Features	Description
현재 위치 표시	도착할 정류장의 이름과 다음 정류장의 이름을 표시한다.
앞/뒤 차량 위치 표시	앞/뒤 차량이 다음으로 도착할 정류장의 위치가 표시된다. 현재 버스가 앞 차량이 도착할 정류장까지 가는데 걸리는 예상 시간된다. 뒷 차량이 현재 버스가 도착할 정류장까지 가는데 걸리는 예상 시간이 표시된다.
정류장 정차 여부 표시	정차할 정류장에 탑승할 또는 하차할 승객이 있으면, 이를 화면에 표시해준다.

Table 6 Application driver features

4. 설계

4.1 시스템 구조

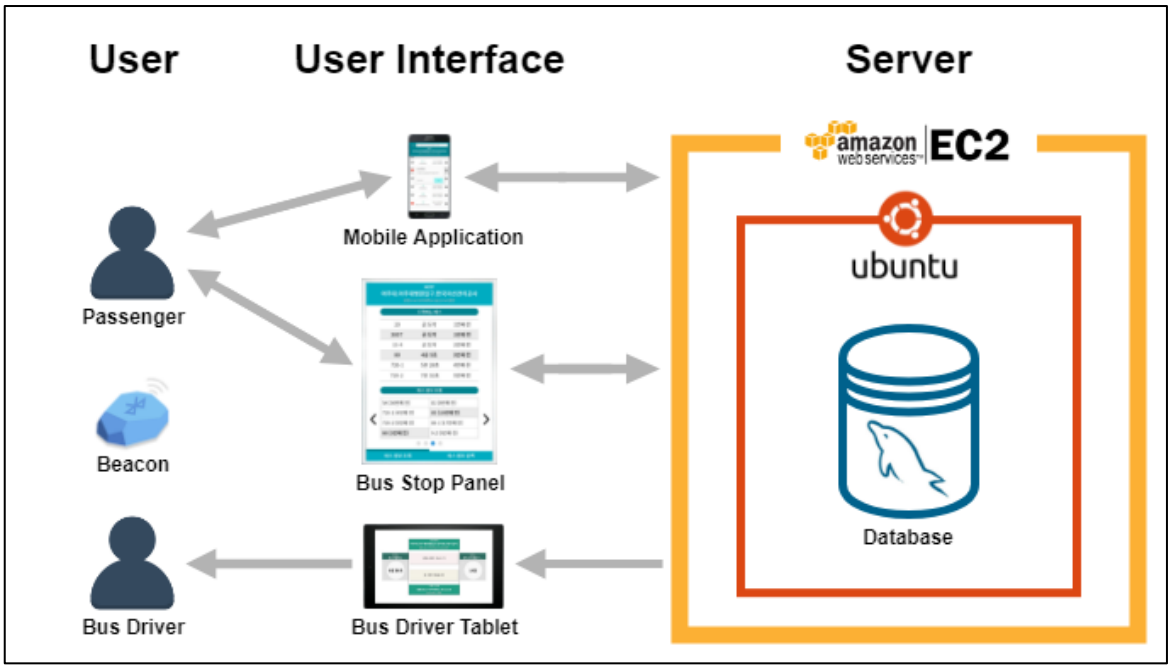

Figure 3 System architecture

4.1.1 User

User 로는 Passenger 와 Bus Driver 가 있다. Passenger 는 Mobile Application 과 Bus Stop Panel 을 이용하여 StopBus 서비스를 사용한다. 마찬가지로, Bus Driver 는 Bus Driver Tablet 을 사용하여 StopBus 서비스를 사용한다.

4.1.2 Beacon

Beacon 은 각 버스마다 2 개, 각 정류장 마다 1 개씩 설치되어 승객이 버스 안에 있는 지 판독하거나, 승객이 정류장에 있는지 확인하기 위해서 사용된다.

4.1.3 User interface

User Interface 는 Mobile Application 과 Bus Stop Panel 그리고 Bus Driver Tablet 로 구성 되어 있다. Mobile Application 은 승하차 예약을 할 수 있고 자신이 원하는 버스 정보를 Server 에서 받아와 표시 해준다. Bus Stop Panel 은 각 정류장마다 설치 되어 있는데 승차 예약이 가능하고 마찬가지로 정류장의 버스 정보를 Server 에서 받아와서 표시 해준다. 마지막으로 Bus Driver Tablet 은 버스마다 설치 되어 Server 에게서 자신의 앞 뒤 버스에 대한 정보를 받아와서 표시 해준다.

4.1.4 Server

Server 는 Amazon Web Service EC2 에서 구동되며, OS 는 Ubuntu 16.04 LTS 를 사용한다. 그리고 기본적인 BIS 정보들은 MySQL 에 저장한다. Server 는 모든 User Interface 기기에게 버스 및 정류장에 대한 정보를 제공한다.

4.2 Components

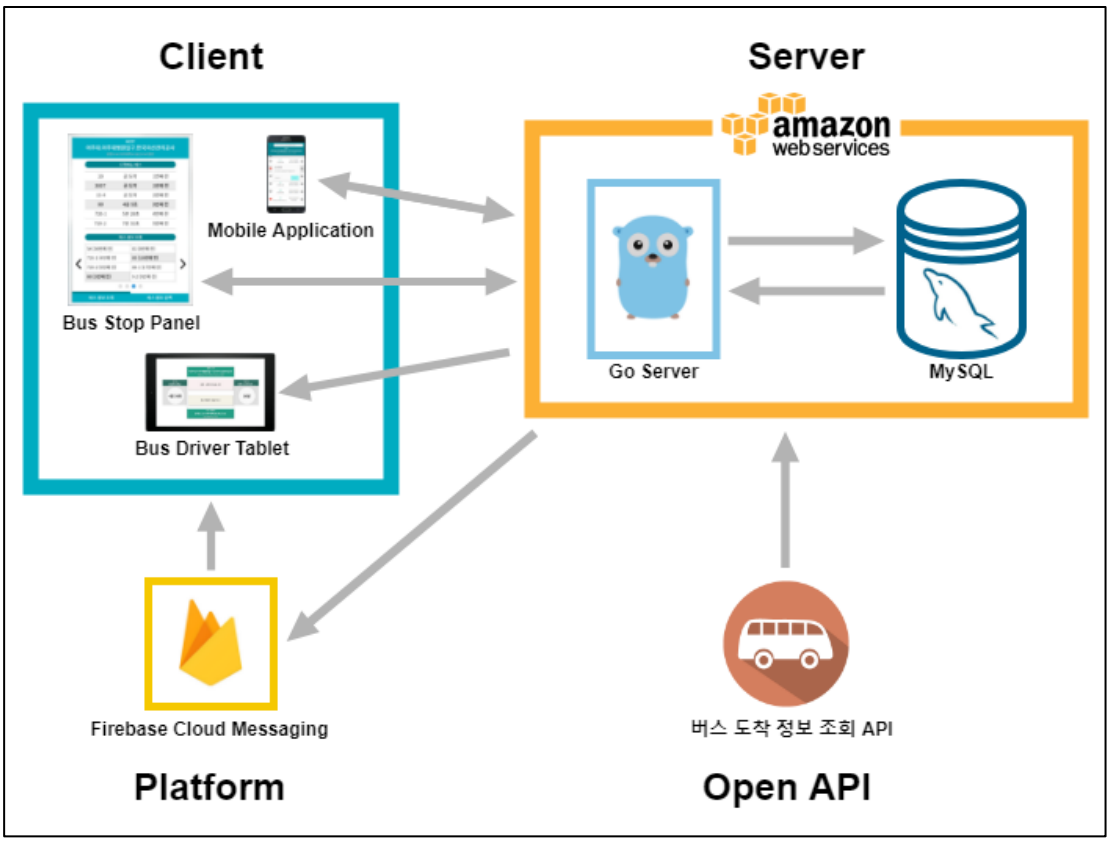

Figure 4 SW architecture

Type	Component	Description
기존	실시간 버스 정보를 위한 API	서버는 GBUS 에서 제공하는 실시간 버스 정보 API 를 받아서 Client 에게 제공한다.
기존	Push 알람을 위한 FCM	서버는 특정 상황에 FCM 에 알림을 요청하여 FCM 은 Client 에게 알림 메시지를 전달한다.
신규	버스 통합 정보를 위한 API	버스 데이터와 승하차 정보를 저장하여 Client 에게 제공한다.
신규	승객의 위치를 파악하기 위한 Beacon 알고리즘	버스 정류장과 버스 내부에 승객이 위치한 경우를 파악하기 위한 Beacon 알고리즘을 구현한다.

Table 7 Components

4.3 데이터 정의

4.3.1 Server (MySQL)

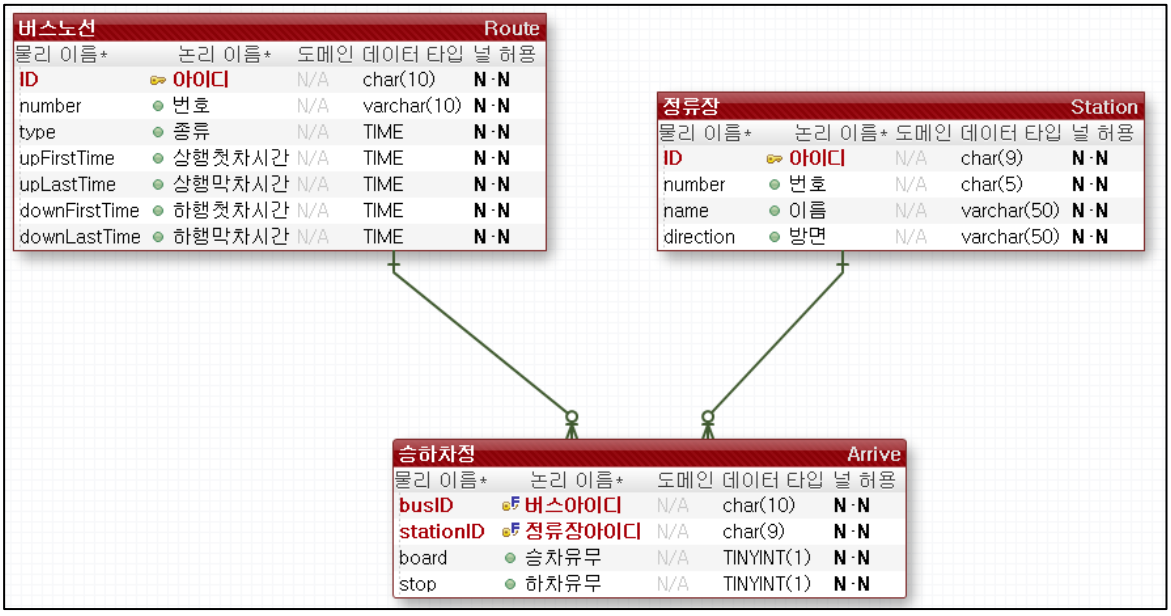

Figure 5 Server data schema

4.3.2 User application (Android)

각각의 User Android App 은 즐겨찾는 버스의 정보를 json 형태로 저장한다.

```
{
    “favouriteBusList”:[
                        {
                                “busNumber”:“720-2”
                        },
                        {
```

```
                    “busNumber”:“3007”
                },
                {
                    “busNumber”:“88”
                }
            ]
}
```

Table 8 User app data example

Name	Type	Description
busList	String	즐겨찾는 버스번호 리스트 busNumber: 버스 번호 (String) 으로 이루어진 배열이 저장되어있다.

Table 9 User app data description

4.3.3 Driver application (Android)

Bus 의 노선정보는 다음과 같이 json 형태로 관리한다.

```
{
    “busNumber”:“720-2”,
    “stationList”:  [
                    {
                        “stationName”:“아주대학교입구”,
                        “stationNumber”: 04238
                    },
                    {
                        “stationName”:
                        “아주대.아주대병원입구.한국자산관리공사”,
                        “stationNumber”: 04237
                    }
                ]
}
```

Table 10 Driver app data example

Name	Type	Description
busNumber	String	버스 번호
stationList	Array	버스가 지나가는 정류장을 지나가는 순서대로 나타낸다. {“stationName”: 버스정류장이름(String),

		“stationNumber”: 버스정류장번호(Number)} 형태의 객체가 배열로 저장되어있다.

Table 11 Driver app data description

4.3.4 Panel application (javaFX)

```
{
        “stationName” : “아주대학교입구”,
        “stationNumber’ : 04238
}
```

Table 12 Panel app data Example

Name	Type	Description
stationName	String	버스 정류장 이름
stationNumber	Number	버스 정류장 번호

Table 13 Panel app data description

4.4 API 및 Interface 정의

API Name	API 사용자	설명
busInfoAndTime	UserAPP Panel	정류소 번호를 보내면 해당 정류소에 정차하는 버스 목록과 도착정보 등을 제공한다.
busInfo		버스들의 상하행시간 등 간략한 정보를 제공한다.
specificBusInfo		한 버스의 차고지 등 상세한 정보들을 제공한다.
busTime		해당 정류소에 해당하는 버스들의 도착정보를 제공한다.
numberSearch		버스번호를 부분적으로 입력하면, 지원하는 버스에 대한 목록을 제공한다.
keywordSearch		키워드를 부분적으로 입력하면, 지원하는 버스에 대한 목록을 제공한다.
driveInfo	Driver	버스 운행에 필요한 정보로 앞뒤 버스와의 간격, 다음 정류장의 승하차 여부를 제공한다.

Table 14 API table

5. 수행 계획

5.1 개발 환경

Client	Server	Tool	OS
Android JavaFX	Go	Android Studio Eclipse Scene Builder Visual Studio Code XShell Postman MySQL WorkBench	Windows 10 Android 4.0.3 (Icecream sandwich) Ubuntu 16.04 LTS

Table 15 Development tools

5.2 리스크 분석

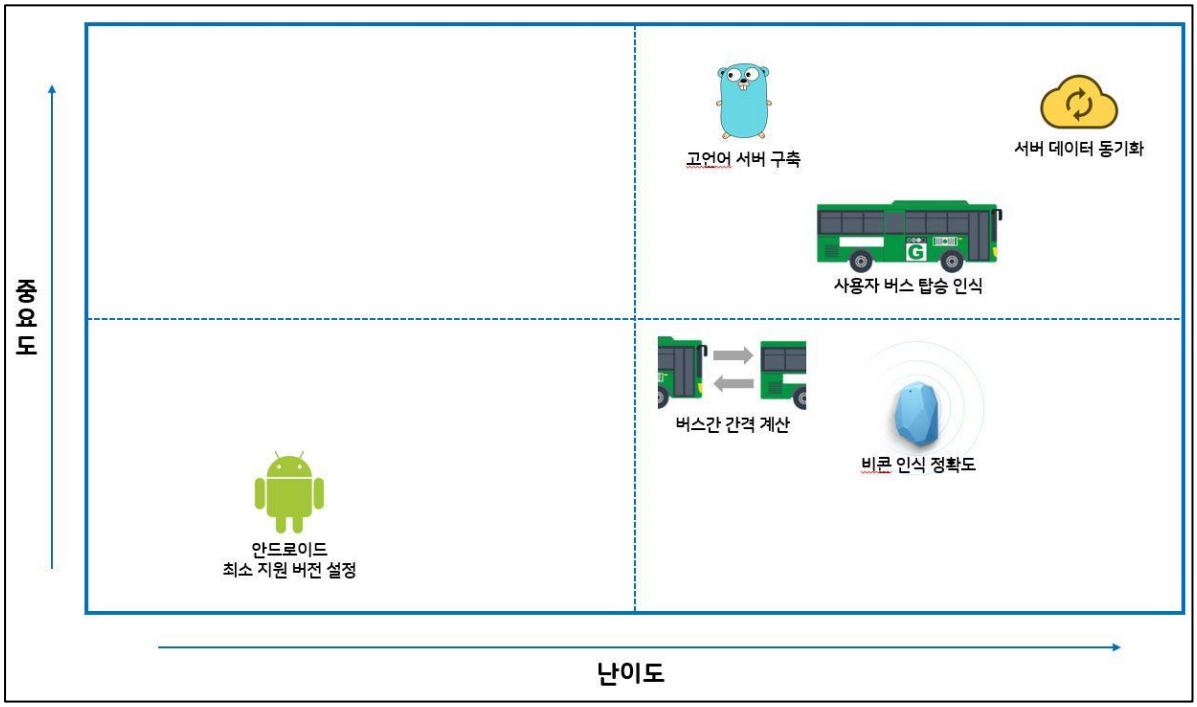

Figure 6 Risk graph

Risk Type	Possible Risk	Solution
Technology	비콘이 여러 개 인식되는 경우가 있을 수 있다.	상대적인 거리를 확인하여 구현할 것이다.
	Go 언어 서버를 구축한 경험이 없기 때문에 서버 구축에 어려움이 따른다.	Node.js 기반의 서버를 구축할 것이다.

	서버는 여러 단말기를 통해 얻는 데이터 값들이 많고 알림을 보내야하는 일이 많기 때문에 서버의 동기화 문제가 있을 수 있다.	단말과 서버간의 데이터 전송 규약을 만들어서 서버가 정확한 타이밍에 알림을 보낼 수 있도록 할 것이다.
	최대한 많은 사람들이 사용자용 어플리케이션을 사용하게 해야 하기 위해 낮은 버전의 안드로이드부터 사용 가능 하기 때문에 최신 안드로이드 기능을 사용 하지 못하는 문제가 있을 수 있다.	가능한 4.0.3 이상의 버전에서는 모두 사용이 가능하게 최하 버전에서 제공하는 기능을 이용하여 어플리케이션을 만들 것이다.
	사용자가 어플을 켰을 때, 버스에 탑승했는지 여부를 판단할 수 있어야 한다.	버스 1 대당 비콘 1 개가 아닌 2 개를 사용하여 사용자의 위치를 계산하여 판단할 것이다.
	GBUS API 로부터 정보를 요청한 버스의 앞 버스와 뒷 버스 위치 데이터를 가져올 수 있어야 한다.	제공받을 수 있는 데이터가 없다면 정류소의 도착예정 시간을 계산하여 버스간 간격을 구한다.
Testing	실제 버스를 타면서 서비스를 테스트 해봐야 하기 때문에 테스트에 어려움이 따른다.	가상의 버스와 버스 노선 그리고 정류장을 만들어서 테스트를 해보고, 실제 버스정보에 대한 테스트는 정류장에서 진행할 것이다.
Teamwork	개발 도중 기획서와 다른 방향으로 개발이 진행 될 수 있다.	개발은 기획서를 기반으로 하되, 변경해야 할 사항이 있으면 Slack 을 활용한 내부 회의를 거친뒤 진행한다.
	각자의 진행사항이 달라서 개발에 진척이 생길 수 있다.	모든 팀원은 Trello 에 자신의 개발 상황을 꾸준히 업로드 하면서 항상 서로의 개발 진척도를 알고 문제가 생기면 서로 도울 것이다. 그렇게 하므로써 팀원의 뒤쳐짐을 원천적으로 막아서 개발에 진척이 생기지 않게 한다.

Table 16 Risk table

5.3 개발 일정

Iteration	User Stories	Task Description	Estimation(days)
1 (4/16~4/22)	개발 환경 구축	User APP Android 개발환경 구축	2
		User Panel JavaFX 개발환경 구축	2
		Driver APP Android 개발환경 구축	2
		StopBus Server Go 개발환경 구축	2
	Beacon 분석	Beacon 테스트	2
	GBUS API 사용 환경 구축	GBUS API 신청	0.5
		GBUS API 테스트	3
	Stop Bus API 구성	단말기 별 필요 API 설계	3
2 (4/23~5/6)	User APP 기능 구현	버스/정류장 검색 기능 추가	3
		즐겨찾기 등록 기능 추가	3
		즐겨찾기 삭제 기능 추가	3
	User Panel 기능 구현	버스 정보 검색 기능 추가	2.5
		전체 버스 정보 조회 기능 추가	2
	Driver APP 기능 구현	현재 위치 표시 기능 추가	2
		앞 차량 위치 표시 기능 추가	3
		뒷 차량 위치 표시 기능 추가	3
	DB 구현	DB Schema 생성	2.5
		각 단말기에 필요한 데이터 입력	3

3 (5/7~5/20)	User APP 기능 구현	즐겨 찾기를 통한 승차 예약 기능 추가	2
		정류장 정보 표시 기능 추가	2.5
		정류장에서 승차 예약 기능 추가	3
	User Panel 기능 구현	승차 예약 기능 추가	3
		선착순 버스 도착 정보 기능 추가	2.5
	Driver APP 기능 구현	탑승 여부 표시 기능 추가	1.5
		하차 여부 표시 기능 추가	2
	Restful API 구현	각 단말기별 API 구현	3
4 (5/21~6/3)	User APP 기능 구현	정류장 하차 예약 기능 추가	2.5
		하차 알림 및 자동 하차벨 눌림 기능 추가	2
	데모 제작	버스 모형 제작	3
	Beacon 기능 구현	버스 내 Beacon 알고리즘 구현	1.5
	Restful API 구현	각 단말기별 API 수정 및 보완	2.5
5 (6/4~6/11)	데모 제작	정류장 모형 제작	2
		데모 시나리오에 따른 데모 제작	2.5
	기능별 테스트	User APP 종합 Test	3
		User Panel 종합 Test	3
		Driver APP 종합 Test	3
		Server Log 모니터링 및 종합 Test	3

Table 17 Development schedule

5.4 비용 분석

비용 분석 대상	설명	비용
Beacon	정확한 거리 계산을 위한 물품	23,900 * 8 개
Amazon Web Service	서버 제공 서비스	Free Tier 이용
데모 환경 구현	데모를 위한 정류장/ 도로 제작	약 10 만원
합계	약 29 만원	

Table 18 Cost table

5.5 업무 분장 계획

5.5.1 FEATURE / 컴포넌트 기반

Member	Feature 기반 업무 분장
박명진	서버 구현 • AWS EC2 환경 구축 • 데이터베이스 구축 • 서버 API 설계 • GBUS API 분석 • FCM PUSH 알림 구현
김범근	User Application UI 디자인 User Application 구현 - 스마트폰 • 정류장 정보 표시 • 승차 예약 • 버스/정류장 검색 • 즐겨찾기 관리 • 즐겨찾기를 통한 승차 예약 • 정류장 하차 예약 • 하차 알림 및 자동 하차벨 눌림
김지선	User Panel UI 디자인 User Panel 구현 - 정류장 • 승차예약 • 선착순 도착 버스 정보 • 전체 버스 정보 조회 • 버스 정보 검색 Beacon 센서를 이용하여 차량 내부 인식 알고리즘 구현 정류장 모형 제작
김희연	Driver Application UI 디자인

	Driver Application 구현 - 버스 기사 태블릿 • 현재 위치 표시 • 앞,뒷 버스 위치 표시 • 정류장 정차 여부 표시 Beacon 센서 분석 및 설계 버스 모형 제작

Table 19 Feature job assignment

5.5.2 기타과제관리/DEPOLY/발표 등 부가적 업무 분장

Member	기타 업무 분담
박명진	컨셉, 기획, 중간, 최종 발표 외부 공모전 참가 신청
김범근	예산 관리 회의록 작성 및 관리
김지선	프로젝트 일정 관리 발표 자료 구성 및 제작
김희연	Deploy 작성 멘토 미팅 일정 관리

Table 20 ETC job assignment

5.6 협업 방안

5.6.1소스 코드 관리

도구	운영방안
GitHub	Server, User, Driver, Panel 로 4 개의 Repository 를 사용한다. 각 소스코드의 버전관리를 위해 사용하며, 필요한 경우 Issue 기능을 사용하여 수정해야할 부분들에 대한 언급을 이용한다.

Table 21 Cooperation GitHub

5.6.2 CI 관리

도구	운영방안
Travis	Travis 를 통해 지속적인 빌드 상태를 확인한다. GitHub 내 빌드 상태를 README.md 에 표시하여 상호간 확인할 수 있도록 한다.

Table 22 Cooperation Travis

5.6.3이슈 관리

도구	운영방안
Trello	매번 모임마다 회의 내용을 기록하거나 일정, 개인별 개발 현황, 비용발생내역 등에 대한 소스코드 이외의 정보들을 모두 기록한다. 그리고 API 의 변경시에는 Dev Board 에 수정된 이력을 첨부하여 다른 사람들이 확인할 수 있도록 게시한다.
Slack	GitHub 와 Trello 에 대한 변경 알림을 알려준다. 그리고 개발에 필요한 채팅을 나누는 과정에서 사용하고, reminder 기능을 사용하여 모임에 대한 알림을 준다.

Table 23 Cooperation Trello & Slack

6. 출시 계획

6.1 성과 측정 방안

6.1.1성과 측정 방안

목표	평가지표(KPI)	측정방법	가중치
버스 내/외부 위치 판단	버스 내/외부 위치 판단	버스를 타지 않고, 정류장에만 있을 때에도 탑승으로 인식되지 않는지	30%
	버스 내 인식 범위	실제 버스에서 테스트를 할 때, 버스의 구석 부분까지 움직여서 위치가 정확하게 인식되는지	
	위치에 따라 알맞은 서비스 화면 제공	모의 버스를 제작하여 핸드폰의 위치에 따라서 특정 버스의 경로 화면을 출력하는지	
승차벨 기능의 정확도 개선	버스 기사 알림	승차벨을 패널이나 APP에서 눌렀을 때, 특정 버스에만 승차 알림이 가는지	30%
	알림 초기화	승차 예약된 버스가 정류장을 지나갔을 때, 패널의 승차벨이 비활성화 되는지	
앱과 물리적 하차 버튼의 동기화	하차 버튼 활성화	APP으로 하차 예약을 한 정류장에 진입 전, 물리적 하차 버튼이 활성화 되는지	20%
	하차 버튼 비활성화	하차 예약을 한 정류장을 지날 때, 물리적 하차 버튼이 비활성화 되는지	
정류장 위치 판단	정류장 위치 판단	정류장 근처에 있을 시 가장 가까운 정류장을 정확하게 판단하는지	20%

	위치에 따라 알맞은 서비스 화면 제공	정류장에 도착했을 때, 핸드폰의 위치에 따라서 정류장 승차 화면을 출력하는지	

Table 24 KPI (Key Performance Indicator)

6.2 데모시나리오

6.2.1 데모 시나리오1

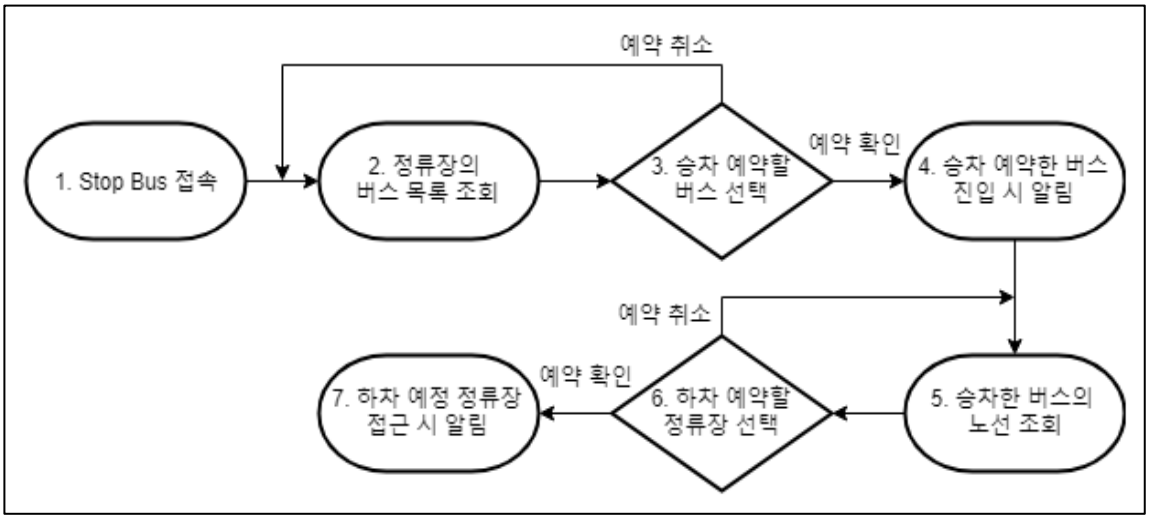

Figure 7 Demo scenario 1

Demo Scenario 1	
Actor	Application User
데모 시나리오	1. Stop Bus 접속 • 사용자가 Stop Bus 어플리케이션에 접속한다.
	2. 정류장의 버스 목록 조회 • 사용자는 Stop Bus 에 표시된 정류장의 버스 목록을 보고 승차 예약을 원하는 버스 정보를 조회한다.
	3. 승차 예약할 버스 선택 • 버스 목록 중 승차 예약을 하고자 하는 버스를 선택한다. 버스를 선택하면 승차 예약 정보가 담긴 확인 메시지가 뜬다. 버스를 잘못 선택한 경우 '아니오' 버튼을, 버스를 맞게 선택한 경우 '예' 버튼을 누를 수 있다. '아니오' 버튼을 누르면 버스 목록이 표시된 화면으로 돌아가고, '예' 버튼을 누르면 해당 버스에 대한 승차 예약이 완료된다.
	4. 승차 예약한 버스 진입 시 알림 • 승차 예약한 버스가 정류장으로 진입하면 Stop Bus 에서는 알림이 뜨게 된다.

	5. 승차한 버스의 노선 조회 • 사용자가 버스에 승차하게 되면 Stop Bus 에서는 승차한 버스의 노선도가 표시된다. 사용자는 Stop Bus 에 표시된 노선도를 보고 하차 예약을 원하는 정류장 정보를 조회한다.
	6. 하차 예약할 정류장 선택 • 노선도에서 하차 예약을 원하는 정류장을 선택한다. 정류장을 선택하면 하차 예약 정보가 담긴 확인 메시지가 뜬다. 정류장을 잘못 선택한 경우 '아니오' 버튼을, 정류장을 맞게 선택한 경우 '예' 버튼을 누를 수 있다. '아니오' 버튼을 누르면 노선도가 표시된 화면으로 돌아가고, '예' 버튼을 누르면 해당 정류장에 대한 하차 예약이 완료된다.
	7. 하차 예정 정류장 접근 시 알림 • 하차 예약한 정류장 근처로 접근할 때, Stop 버스에서는 알림이 뜨게 된다. 또한 버스에서는 자동으로 하차벨이 눌린다.

Table 25 Demo scenario 1 description

6.2.2 데모 시나리오2

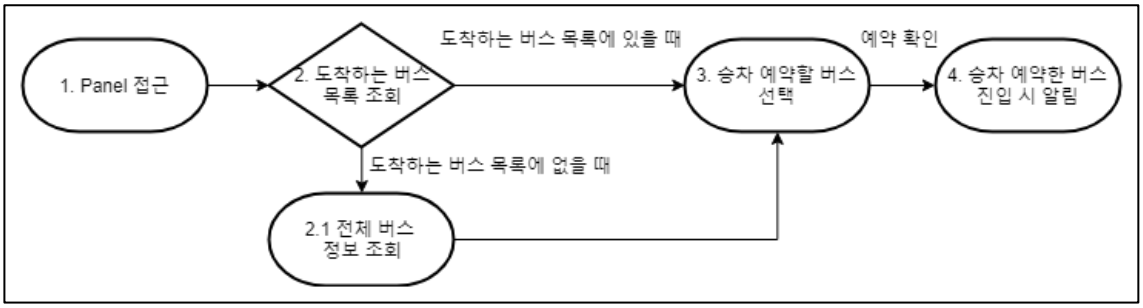

Figure 8 Demo scenario 2

Demo Scenario 2	
Actor	Panel User
데모 시나리오	1. Panel 접근 • 사용자가 정류장에 설치된 Panel 에 접근한다.
	2.a 도착하는 버스 목록 조회 • 사용자는 Panel 화면의 '도착하는 버스'에 표시된 버스 목록을 조회할 수 있다. '도착하는 버스'에서는 선착순으로 도착하는 버스의 목록이 표시된다. 2.b 전체 버스 정보 조회 • '도착하는 버스'에 사용자가 찾고자 하는 버스가 없을 때, 사용자는 '버스 정보 조회'에서 전체 버스 목록을 조회할 수 있다.
	3. 승차 예약할 버스 선택 • 사용자는 버스 목록 중 승차 예약을 하고자 하는 버스를 선택한다. Panel 에서는 승차 예약이 완료되었음을 표시한다.

	4. 승차 예약한 버스 진입 시 알림 • 승차 예약한 버스가 정류장으로 진입하면 Panel 에서는 음성으로 버스가 진입중임을 안내한다.

Table 26 Demo scenario 2 description

6.3 향후 발전 방향

Goal	Description
Universal Design 을 이용한 접근성 개선	버스는 다양한 연령층의 사용자가 이용한다. 따라서 여러 사람들이 쉽게 접근하여 사용할 수 있는 디자인을 고안해야 한다.
결제 시스템 도입	버스를 탑승, 하차할 때 NFC 를 태그해야 한다. 비콘을 이용하여 승객이 버스에 탑승했는지 아닌지를 판단할 수 있기 때문에, 굳이 NFC 태그를 하지 않고도 결제가 가능해지므로, 자동 결제 시스템을 제공한다.
교통약자들을 위한 기능 추가	교통약자를 위해 universal design 을 추가하여, 교통약자가 application 을 쉽게 이용할 수 있게 한다. 또한, 교통약자가 승 하차 예약을 한다면, Bus Driver 에게 교통약자가 승차 또는 하차를 한다는 특별한 알림을 주어, 도움을 줄 수 있게 한다. 마지막으로, 교통약자에게 예약한 버스가 정류소에 들어올 때, 휴대폰에서 어떠한 버스가 들어오는지를 알려주며, 해당 핸드폰으로부터 버스까지의 거리를 알려주는 기능을 추가할 것이다.

Table 27 Goals

7. 참고문헌

1. 최병삼(2013), 스마트폰 또는 이동통신 단말기를 이용한 하차벨 시스템 및 그 방법법, Google patents

2. 김준호, 이성원(2015), “비콘 기반의 버스 자동 승하차 시스템 구현”, 한국통신학회 하계종합 학술발표회

3. 경기도 버스 운송 사업 조합 사이트, 교통불편신고, http://www.gbus.or.kr/2006/program/board/board_lst.php?inja=unkind&sm=5_1#, (2018.03.16)