더치 빗자루

Dutch Broomstick

이정민 2017-10483
장태준 2017-17018
정유석 2017-10433

Abstract

“얽히고 설킨 돈 관계를 쓸어담는 더치빗자루“

더치페이 상황에서 일어나는 복잡하고 번거로운 돈 계산을 도와주는 웹 서비스입니다.

+ 돈 거래 횟수의 최소화
+ 링크를 통한 편한 접근성
+ 수금자가 선호하는 수금 방법을 송금자에게 제공
+ 그룹에 속한 모두가 지출 정보 등록 가능
+ 송금자에게 수금자의 계좌 정보 복사 기능 제공
+ 1/N 은 물론이고 다양한 분할 계산 제공
+ 해외여행과 같은 특수한 상황에서 환율 계산도 지원
  (직접 환율 계산 기능 또는 레이어기능)

Motivation & Problem

토스 등 간편한 송금 앱의 사용이 활성화됨에 따라, 한 사람이 돈을 다 내고, 나머지 사람들이 송금하는 경우가 많다.
식사 한 번은 계산이 간단하지만, 여행처럼 지속적으로 지출이 있는 경우, 추후 복잡한 정산이 필요하다. 이때는 각자가 신경써야 할 것과 번거로운 점이 많다.

이러한 불편함을 해소하기 위하여, 지출 내역만 입력하면 가장 깔끔한 정산 과정을 제안하는 어플리케이션을 만들어 보기로 하였다.

Related work

현재 많은 카드사들이 제공하는 앱에 자동 더치페이 기능이 있고, 그 외에도 많은 더치페이 앱들이 존재합니다. 대표적으로 카카오페이, 우리은행, 신한은행 등 신용카드 업계들이 제공하는 서비스, 엔팡, A형총무, Mr.Bill 등의 앱이 존재합니다.

제공되고 있는 서비스들의 단점은 다음과 같습니다.

- 계좌 정보 연동의 번거로움
- 1인 지출이 아닌 다인 지출의 경우 늘어나는 송금 횟수의 번거로움
- (카드사의 경우) 양측 모두의 필수적인 어플리케이션 설치
- 지출 금액이 차이나는 경우를 지원하지 않는 경우

더치빗자루는 위와 같은 단점을 아래와 같은 방식으로 해결하고자 합니다.

- 계좌 정보 연동의 간편화, 최소화
  - 로그인 연동 시 자동 계좌 입력 방식
  - 지출이 없는 사람의 경우 계좌정보를 요구하지 않음
- 송금 횟수의 최소화
  - 내부 계산 절차를 거쳐, 송금 횟수의 최소화를 추구
- 지출 금액 설정 다양화
  - 1/N, 금액 직접 입력 등 다양한 더치페이 방식 지원

Functionality

Product Definition
"더치페이 상황에서 복잡한/번거로운 돈 계산을 도와주는 웹 서비스"
Product VIsion
"최대한 가볍게" & "최대한 효율적으로"
Persona
"지속적인 지출 관계가 있는 그룹"
Input
사람 목록 & 지출 내역
Output
최적 정산 결과

본 웹서비스는 다음과 같은 기능을 제공합니다.

- 사용자는 "방"을 개설할 수 있습니다. "방"을 개설한 방장은 구성원의 이름을 입력한 뒤 (나중에 수정할 수 있음), 링크를 발급받아 이를 구성원들에게 공유합니다.

- 링크를 가진 사람은 누구나 "방"에 접근하여, 결제 내역을 작성할 수 있습니다. 누가 / 얼마를 / 어디에 결제했는지를 입력해두면 다른 구성원들이 언제든 이를 확인할 수 있습니다.

- 결제 내역이 아무리 길어질지라도, 사용자는 가장 간단한 정산 결과를 확인할 수 있습니다. 더치빗자루는 매 결제마다 '결과적으로 누구한테 얼마를 보내면(받으면) 되는지'를 쉽게 계산해 줍니다.

- 모임이 끝나고, 실제로 돈을 보내는 단계에서는 외부 애플리케이션을 이용해야 합니다. 대신 더치 빗자루에서는 방 구성원이 어떤 결제 수단을 선호하는지 알려주고, 계좌번호를 쉽게 복사할 수 있는 기능을 제공함으로써 송금 과정을 돕습니다.

추가로, 특수한 결제 상황을 지원하기 위해 다음과 같은 기능을 준비하였습니다.

- 결제 내역을 작성하는 상황에서, 보통은 부담을 똑같은 액수로 나누는 이른바 'N빵' 방식을 사용합니다. 더치빗자루는 그 외에도 100원 단위 절삭, 메뉴값 별 계산, 랜덤 몰아주기 등 다양한 분할 계산 방법을 추가로 제공합니다.

- 해외 여행을 간 상황에서도 쉽게 돈 정산이 가능하도록 환율 계산 기능도 지원합니다. "방" 설정에서 언제든 통용되는 통화를 변경할 수 있습니다.

Algorithm for Simplifying Credits

송금 횟수를 최소화시키기 위해 결제 내역을 변형하는 알고리즘은 Client-side에서 JavaScript로 구현한다. 이를 위해서는 Layer 내에 존재하는 모든 Credit 정보를 백엔드로부터 읽어와야 한다.
(`GET /rooms/:room_pk/layers/:layer_pk/credits`)
백엔드 데이터를 다음과 같이 가공하면, 후술할 알고리즘을 통해 송금 횟수를 최적화 시킬 수 있다.

```
[ { from: 1, to: 2, amount: 1000.0 },
  { from: 2, to: 3, amount: 2000.0 }, ...]
```

다음 알고리즘은 n명의 사람이 있을 때 최대 n-1번의 결제로 정산을 끝낼 수 있다. 시간복잡도는 M이 총 결제 횟수, N이 사람일 때 O(M+N)이다. '최소'는 보장할 수 없지만 충분히 효율적인 휴리스틱이다.

1. `define total[M]   // total[i] : i가 받아야할 돈`
2. `let total[i] = sum(c.amount for c in credits if c.from === i)`
   `+ (-1) * sum(c.amount for c in credits if c.to === i)`
3. `j가 j+1에게 total[j+1]-total[j]를 보내주면 된다.`

Design & Implementation

<Backend Design>

Restful API

각 URL 밑에 필요한 parameters를 명시해두었다. (Params)

Optional한 필드의 경우 대괄호로 감싸 표시하였고, URL에 명시된 params는 제외하였다.

Users

- POST /users
  - Params: id, password, default_name, [default_account]
  - 회원가입 수행
- POST /users/:user_pk/login
  - Params: password
  - Returns: JWT
- PUT /users/:user_pk
  - Params: [password], [default_name], [default_account]
  - 회원정보 수정
- DELETE /users/:user_pk

Room

- POST /users/:user_pk/rooms
  - Params: id, name
  - 새로운 방 생성 (기본 Layer와 Member(owner)도 생성)
- GET /users/:user_pk/rooms
- PUT /rooms/:room_pk
  - Parmas: [id]
- DELETE /rooms/:room_pk

Member

- POST /rooms/:room_pk/members
  - Params: name, account
- GET /rooms/:room_pk/members
- GET /rooms/:room_pk/members/:member_pk
- PUT /rooms/:room_pk/members/:member_pk
  - Params: [name], [account], [user]
- DELETE /rooms/:room_pk/members/:member_pk

Layer

- POST /rooms/:room_pk/layers
  - Params: name, currency
- GET /rooms/:room_pk/layers
- GET /rooms/:room_pk/layers/:layer_pk
- PUT /rooms/:room_pk/layers/:layer_pk
  - Params: [name], [currency]
- DELETE /rooms/:room_pk/layers/:layer_pk

Payment

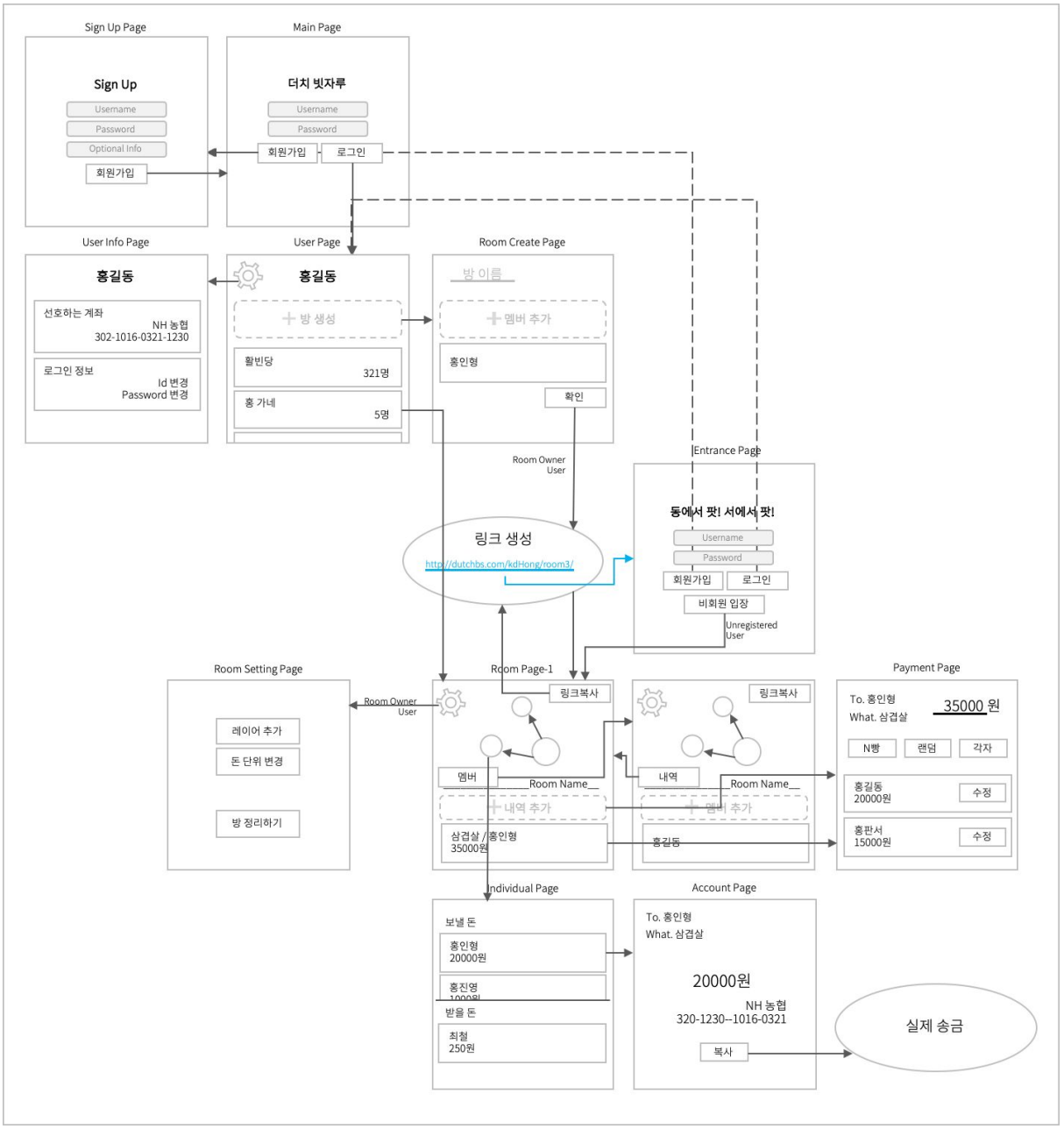
- POST /rooms/:room_pk/layers/:layer_pk/payments
  - Params: from, for, to_array, amount_array
  - 연관된 Credit까지 자동으로 생성됨
- GET /rooms/:room_pk/layers/:layer_pk/payments
- GET /rooms/:room_pk/layers/:layer_pk/payments/:payment_pk

Credit

- `GET /rooms/:room_pk/layers/:layer_pk/credits`
    - `Parmams: member`
    - `member` 필드를 지정하면 특정 멤버와 관련된 `Credit`만 필터링 됨
- `GET /rooms/:room_pk/layers/:layer_pk/credits/:credit_pk`

<Frontend Design>

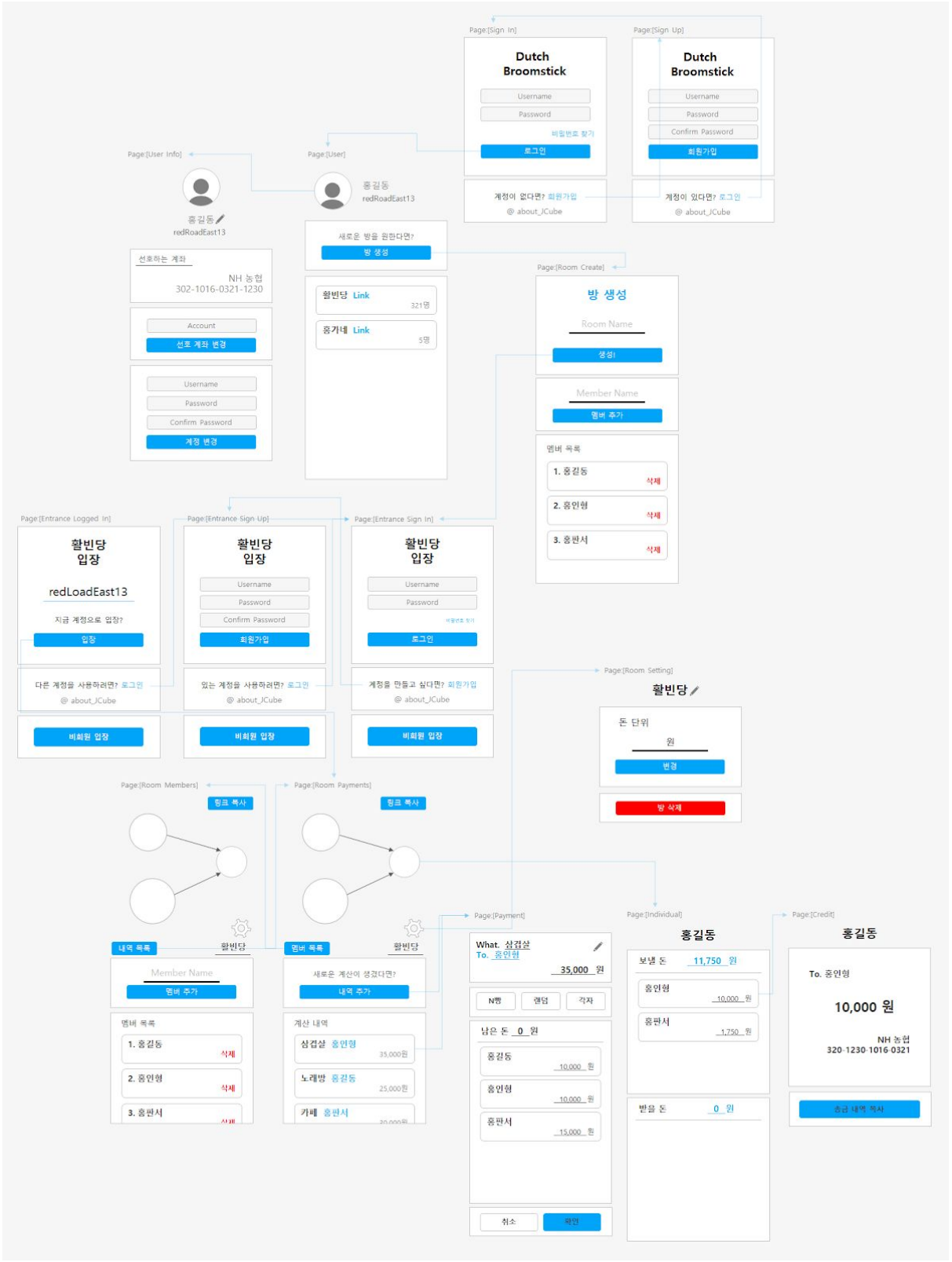
Atomic React



UI/UX

사용자 흐름도

로그인 -> 방생성 -> 방 입장 -> 내역 추가 -> 개인 송금 확인 -> 송금



Conclusion
<결론>

http://skile.me:3000/

한 학기가 채 되지 않는 개발 기간 동안 구현한 결과이다.
처음 계획한 것, 그리고 개발 기간 동안 욕심이 난 부분들을 모두 구현하지는 못하였다.
(ex. Layer 개념)
그러나 핵심적인 송금 횟수 최소화 알고리즘을 고안하였으며,이를 그래프를 통해 시각화하였다.

이는 단순히 하나의 웹페이지를 만들어본다는 경험을 넘어,
최적화를 위한 알고리즘을 구상하고 그를 실제로 서비스하는 구체화 단계까지 구현한 것이라고 생각한다.

또한, 완성도 측면에서도 부수적인 부분을 제외한 핵심적인 부분들은 모두 구현을 완료하였다.
따라서 짧은 기간 동안 꽤 만족할만한 성과를 이루었다고 생각한다.

<느낀점>
-이정민
힘들지만, 보람있는 프로젝트였다.

-장태준
힘들지만, 보람있는 프로젝트였다. 팀원들과 함께 서비스를 기획하고, 세부적인 구현을 회의하면서 많은 것을 배울 수 있었다.

-정유석
힘들지만, 보람있는 프로젝트였다.