# 더치빗자루

Design and Planning

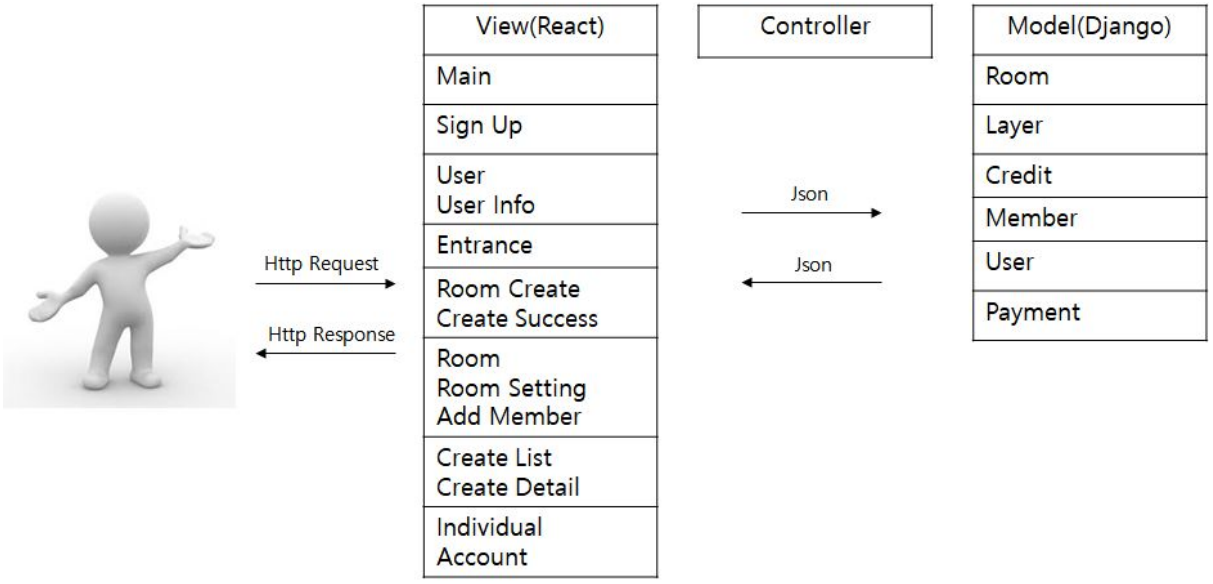
2019-04-16 - first version

## Members

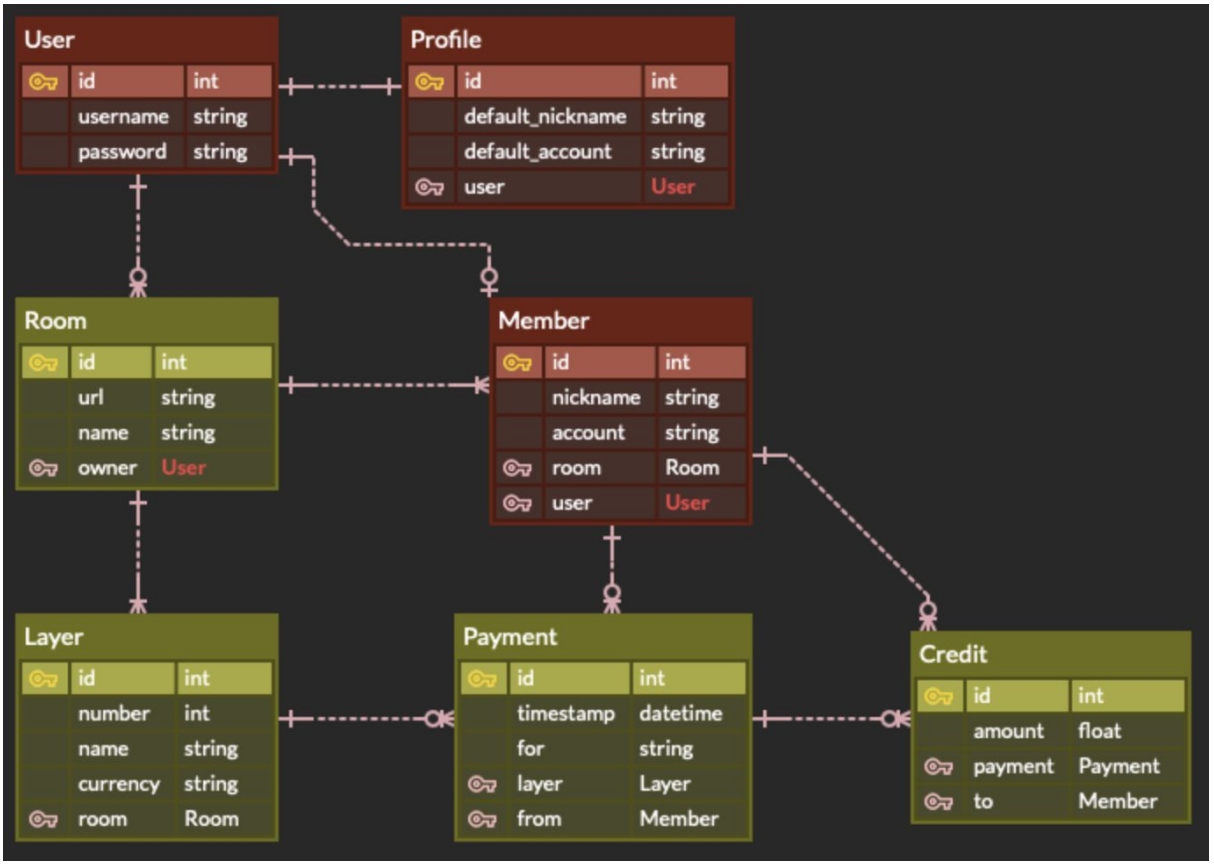
- 2017 - 10433 정유석
- 2017 - 10483 이정민
- 2017 - 17018 장태준

## System Architecture

### MVC



## Model




https://www.erdcloud.com/d/Gj4s8JDgumLabZQZ5

위 그림은 모델의 구조를 E-R diagram으로 표현한 것이다. Entity 사이의 관계를 표현하는 데 까마귀발(Crow-feet) 표기법을 사용하였다.

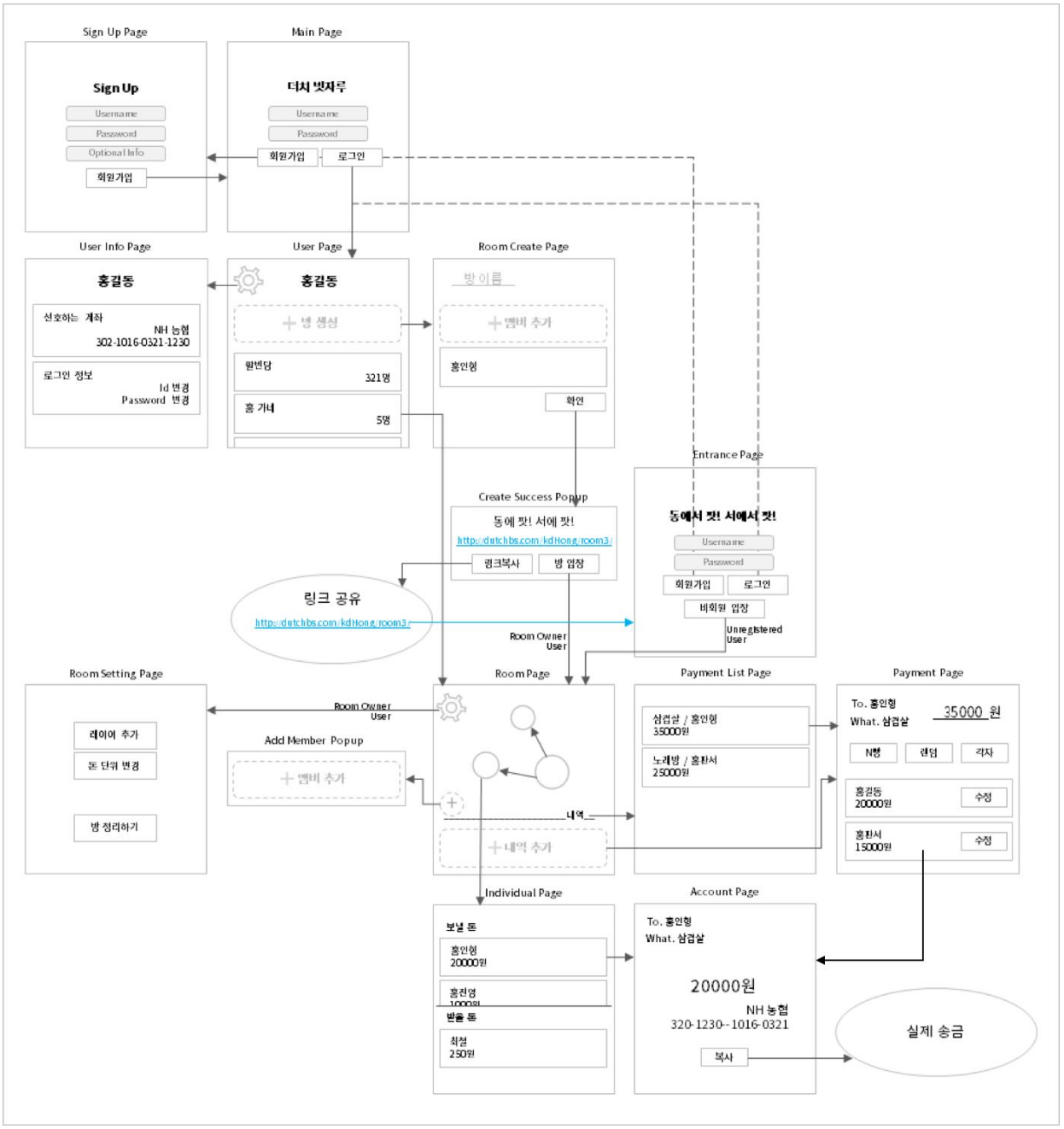
각각의 Entity는 본 서비스에서 다음 개념을 상징한다.

- User : 서비스에 회원 가입한 사용자
- Room : 결제 내역을 공유하는 “방”
- Layer : 결제 내역을 분리해서 관리하는 한 단위
    - number :
    - 해외여행 등에서 다른 통화 단위를 사용할 때 등에 유효한 개념이다.
- Payment : 한 번의 결제 행위
    - for 필드는 결제 목적을 상징한다.
    - 모임이 끝나고 돈을 정산(송금)하는 행위 또한 Payment로 취급한다.
- Credit : 한 사람에게서 한 사람으로 보내는 결제 내역
    - 하나의 Payment에 여러 개의 Credit이 발생할 수 있다.
- Member : “방”에 소속된 구성원
    - 회원가입을 하지 않은 Member가 존재할 수 있다. 따라서 user 필드는 nullable하다.

모든 Entity에는 pk 필드가 있는데 이는 Django의 Model에는 pk 필드(autoincrement int)가 자동으로 생성됨을 반영한 것이다. 일부 Entity에는 id 필드가 있는데 이는 Django Model의 pk와는 별도로 필요하여 정의하였다.

- User.id는 로그인할 때 입력하는 ID 문자열을 상징한다.
- Room.id는 외부에서 접속 가능한 공유 링크를 생성할 때 사용한다.
- Layer.id는 레이어 번호를 상징한다. (Layer.room이 다를 경우 Layer.id는 중복될 수 있음)

## View




User Interface for View Design

1. Sign Up page (`/sign_up/`)
   -새로운 user의 회원가입
   -username, password 를 input으로 받아 계정생성
   -동일한 유저가 있는지 check

2. Main page(`/`)
   -Sign_Up
   -Sign_In

3. User page(`/user/`)
   -Setting : user info page로 이동
   -+방생성 버튼 으로 Create_Room page로 이동
   -user가 속해는 room의 list를 제공 (owner 인 것 표시)
   : get_room_list

4. User Info page (`/user/setting/`)
   -user의 정보를 나타냄
   -선호하는 계좌 정보를 수정할 수 있음
   -로그인 정보 (username, password) 수정
   : get_user_by_id

5. Room Create page (`/user/create_room/`)
   -방이름을 input으로 받음
   -멤버 추가를 통해 graph의 node 를 설정
   : graph_add_member
   -확인을 통해 room을 생성하고 그에 따라 link 생성됨

6. create success popup page (`/user/create_room/`)
   -방생성에 성공했을 시 띄우는 팝업 창
   -방에 들어갈 수 있는 link를 발급
   -링크를 복사할 수 있는 버튼 제공

7. Entrance page (`/owner/:room/entrance/`)
   -6.에서 발급된 link를 따라 들어갔을 시 가장 처음 나오는 창
   -로그인 되어 있는 상태로 위 링크를 누르면 룸 이름을 보여주고 입장 의사 확인
   -로그인이 안되어 있는 상태로 위 링크를 누르면 두 가지 옵션을 제공
      -> 로그인 or 회원가입 후 룸에 입장
      -> 비회원 상태로 룸에 입장

8. Room page (`/room/:room/`)
   -방에 입장하면 보이는 페이지
   -그래프를 통해 정산 관계를 보여줌
   -멤버를 추가하거나 삭제할 수 있음
   -본 방에 관련된 결제 내역을 확인 및 추가할 수 있음 (빠른 추가)
   -방에 들어온 사람은 member 중 하나와 대응되며 자신과 관련된 결제 내역을 확인 가능

9. Room Setting page (`/room/:room/setting/`)
   -방에 레이어를 추가 및 삭제하는 기능
   -레이어 별 세팅 (돈 단위 등)을 할 수 있는 기능
   -방 자체를 삭제할 수 있는 기능 (방 폭파)

10. add member popup page (`/user/:room/add_member/`)
    -멤버 추가 버튼을 누를 시 나타나는 팝업 창
    -이는 인터페이스 디자인에 따라 팝업 창이 아닌 Room page에 나타날 수도 있음
    -새로운 멤버의 이름을 input으로 받음
    -성공시 그래프에 새 노드가 생성됨

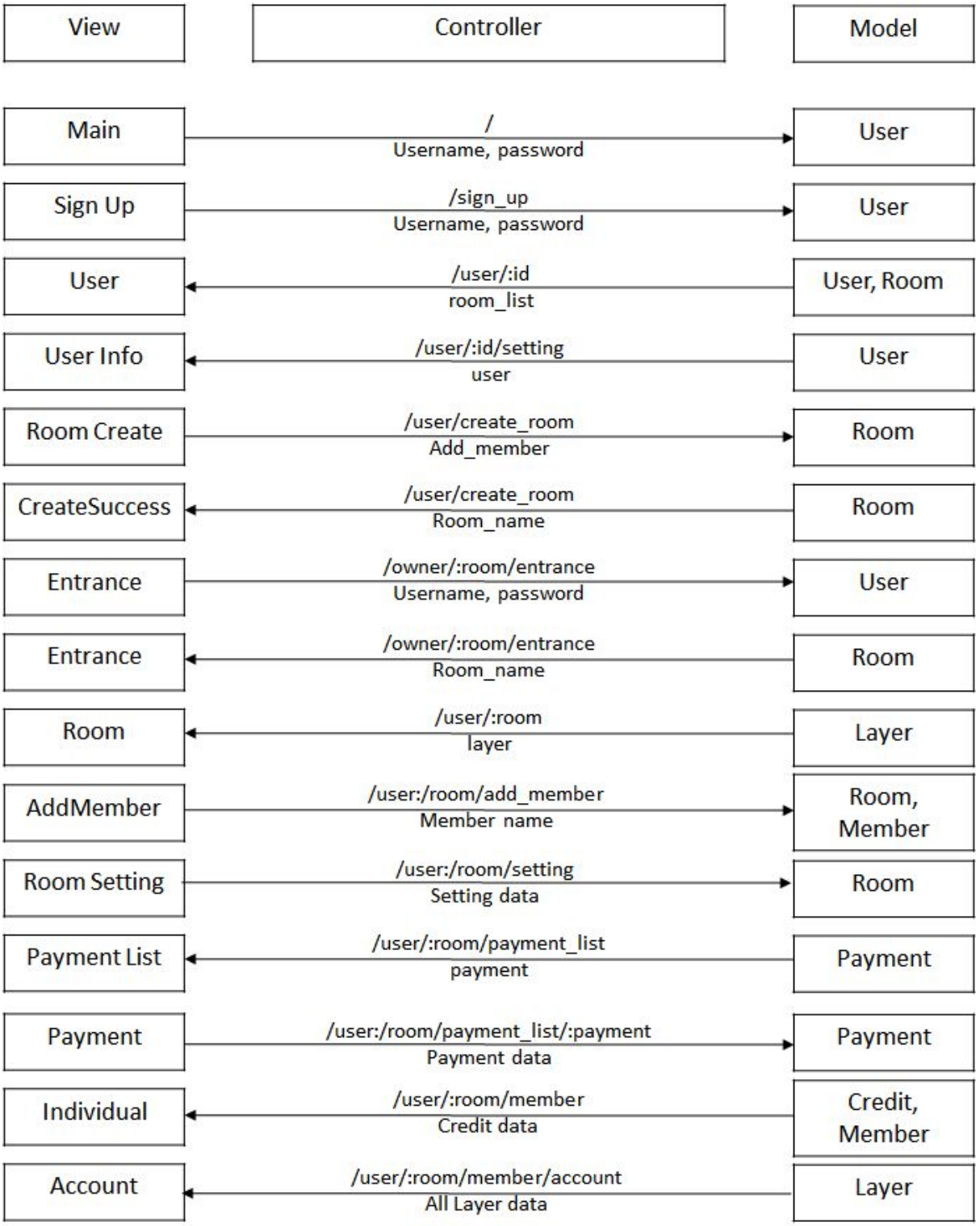
11. Payment List page (`/user/:room/payment_list/`)
    -본 룸에 관련된 결제 내역을 모두 나열하는 창
    -대략적인 정보만을 제공 : 무엇을 ? 얼마를 ? 누가 ?
    -결제 내역의 추가 및 삭제 가능
    (Room 에서의 결제 내역 추가는 빠른 접근만을 위함, 본 페이지에서의 접근을 기반)
    -백엔드와 통신하여 결제 내역을 불러옴
    -각 결제 내역을 클릭할 시  해당 내역과 연관된 Payment page로 연결

12. Payment page (`/user/:room/payment_list/:payment/`)
    -구체적인 결제 내역을 확인할 수 있는 페이지
    -새로운 내역 생성시 default 값이 들어가 있는 Payment 페이지가 나타남
    -무엇을 ? 얼마를 ? 누가 ? 와 함께
     누가 누구에게 얼마를 보내야할 지, 돈을 정산하는 방법은 무엇인지를 설정
                                    (N빵, 랜덤, 각자 비율에 따라)
    -해당 내역에 관헤 누가 누구에게 얼마를 보내야한다는 구체적인 account 리스트를 보여줌
    -account를 클릭하면 account page 로 넘어감
    -정산 된 account 는 따로 마크를 해줌
    -해당 payment에 관련된 모든 account가 정산 될 시, 본 payment가 정산 완료

13. Individual page (`/user/:room/member/`)
    -각 멤버가 자신과 관련된 결제 내역만을 확인할 수 있는 페이지
    -로그인 된 유저의 경우 member 와 자신의 계정을 연결할 수 있음 (송금 방식 연결)
    -보낼 돈, 받을 돈의 accounts를 백엔드와 통신하여 불러옴
    -account 를 누르면 해당  payment와 관련된 page로 연결

14. Account page (`/user/:room/member/account/`)
    -누가 누구에게 얼마를 무엇을 통해(송금방식)을 보여주는 페이지
    -링크를 복사해서 송금이 완료되면 해당 account가 정산 된 것으로 판단
    -본 페이지에는 송금 내역의 수정, 삭제 기능이 없음
    -송금 여부만 체크 할 수 있음
    -credit과 대응되며, 각 credit을 다루는 페이지

## Controller

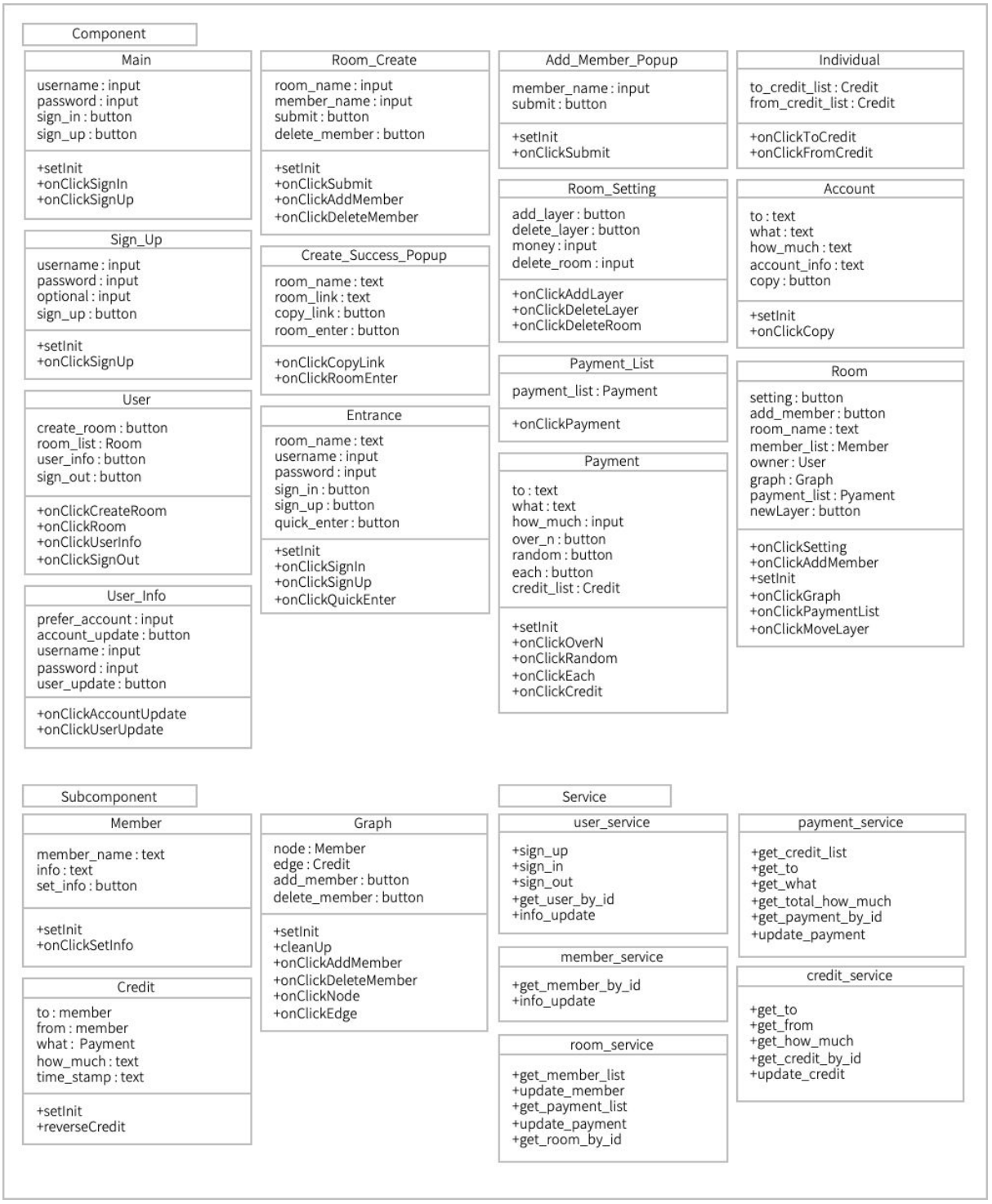

View
Controller
Model
Main
/
Username, password
User
Sign Up
/sign_up
Username, password
User
User
/user/:id
room_list
User, Room
User Info
/user/:id/setting
user
User
Room Create
/user/create_room
Add_member
Room
CreateSuccess
/user/create_room
Room_name
Room
Entrance
/owner/:room/entrance
Username, password
User
Entrance
/owner/:room/entrance
Room_name
Room
Room
/user/:room
layer
Layer
AddMember
/user:/room/add_member
Member name
Room, Member
Room Setting
/user:/room/setting
Setting data
Room
Payment List
/user/:room/payment_list
payment
Payment
Payment
/user:/room/payment_list/:payment
Payment data
Payment
Individual
/user/:room/member
Credit data
Credit, Member
Account
/user/:room/member/account
All Layer data
Layer

## Design Details

### Frontend Design

#### Frontend Components

frontend 의 구성은 다음과 같다. 각 component에서 수행가능한 method 들은 box 안에 있다.



Frontend Algorithms

## Component

### 1) Main

- onClickSignIn(username: string, password: string)
    - 로그인 API 호출
        - 요청 성공 시 User(3)으로 이동 / token을 저장
        - 요청 실패 시 경고 메시지 표시
- onClickSignUp()
    - Sign_Up(2)로 이동

### 2) Sign_Up

- onClickSignUp(username: string, password: string)
    - 회원 가입 API 호출
        - 요청 성공 시 Main(1)로 이동
        - 요청 실패 시 (유저네임 중복 등) 경고 메시지 표시

### 3) User

- setInit()
    - 방 목록 GET API 호출
        - 요청 성공시 해당 정보를 room_list에 저장
        - 요청 실패 시 Main(1)로 이동
- onClickCreateRoom()
    - Sprint 3에서 구현 예정
- onClickRoom(key: int)
    - Sprint 3에서 구현 예정
- onClickUserInfo()
    - User_Info(4)로 이동
- onClickSignOut()
    - Main(1)로 이동

### 4) User_Info

- onClickAccountUpdate(prefer_account)
    - 사용자 정보 수정(PUT) API 호출
        - 요청 성공 시 화면에 업데이트
- onClickUserUpdate(username, password)
    - 사용자 정보 수정(PUT) API 호출
        - 요청 성공 시 화면에 업데이트

### 5) Room_Create

- onClickSubmit(roomName, members)
    - 방 생성 API 호출
        - 요청 성공시 (6)으로 이동
        - 요청 실패시 경고 메시지 표시
- onClickAddMember(member)
    - 인자로 넘길 members에 특정 멤버를 추가
- onClickDelteMember(member)
    - 인자로 넘길 members에 있는 특정 멤버를 지움

## 6) Create_Success_Popup

- onClickCopyLink()
    - 해당 room에 접근할 수 있는 link를 clip board에 저장
- onClickRoomEnter()
    - 해당 room의 페이지에 들어감
        - onClickCopyLink()에서 저장되는 링크에 접속하는 것과 동일

## 7) Entrance

- onClickSignIn(username, password)
    - 로그인 API 호출
        - 요청 성공 시 Room(8)으로 이동
        - 요청 실패 시 (유저네임 중복 등) 경고 메시지 표시
- onClickSignUp(username, password)
    - 회원 가입 API 호출
        - 요청 성공 시 Room(8)으로 이동
        - 요청 실패 시 (유저네임 중복 등) 경고 메시지 표시
- onClickQuickEnter()
    - Room(8)로 이동 : 로그인 또는 회원가입 없이 서비스 이용

## 8) Room

- onClickSetting()
    - Room_Setting(10)으로 이동
- onClickAddMember()
    - Add_Member_Popup(9)을 띄움
- onClickGraph()
    - Subcomponent - (2) Graph에 접근 가능
        - onClickNode, onClickEdge 등 Graph 관련 기능 사용가능
        - 직관적인 접근을 제공
- onClickPaymentList()
    - Payment 목록 GET API 호출
        - 요청 성공시 해당 정보를 payment_list에 저장
        - 요청 실패시 오류 메시지 출력
- onClickMoveLayer()
    - Layer GET API 호출
        - 요청 성공시 다른 Layer로 화면 이동
        - 요청 실패시 오류 메시지 출력

## 9) Add_Member_Popup

- onClickSubmit(member_name)
    - members에 member를 추가함
    - member 생성 POST API 호출
        - 요청 성공시 새로운 member를 room 에 저장
        - 요청 실패시 오류 메시지 출력

## 10)Room_Setting

- onClickAddLayer()
    - Layer 생성 POST API 호출
        - 요청 성공시 새로운 Layer를 room 에 저장
        - 요청 실패시 오류 메시지 출력

- onClickDeleteLayer()
    - Layer 삭제 DELETE API 호출
        - 요청 성공시 해당 layer를 room 에서 삭제
        - 요청 실패시 오류 메시지 출력
- onClickDeleteRoom()
    - Room 삭제 DELETE API 호출
        - 요청 성공시 해당 Room을 삭제 (link 비활성화)
        - 요청 실패시 오류 메시지 출력

### 11)individual

- onClickToCredit(member)
    - toCredit GET API 호출 (해당 member가 보낼 돈)
        - 요청 성공시 해당 credit의 account page 연결
        - 요청 실패시 오류 메시지 출력
- onClickFromCredit(member)
    - fromCredit GET API 호출 (해당 member가 받을 돈)
        - 요청 성공시 해당 credit의 account page 연결
        - 요청 실패시 오류 메시지 출력

### 12)Account

- onClickCopy()
    - “누가 누구에게 얼마를 어디로 보낼지”의 내용을 clip board에 저장
    - 어떻게 최종 결과가 나왔는지는 화면에만 출력.

## Subcomponent

### 1) Member

- onClickSetInfo(member_name, prefer_account)
    - member 정보 수정 PUT API 호출
        - 요청 성공시 정보 update
        - 요청 실패시 오류 메시지 출력

### 2) Graph

- cleanUp()
    - graph 내의 모든 edge를 삭제함
- onClickAddMember(member_name, prefer_account)
    - member 추가 POST API 호출
        - 요청 성공시 새로운 member를 members에 추가, 해당 member에 대응되는 새로운 node 추가
        - 요청 실패시 오류 메시지 출력
- onClickDeleteMember(member)
    - member 삭제 DELETE API 호출
        - 요청 성공시 해당 member 정보를 삭제 단, prev_edge 는 정보만 남겨둠
        - 요청 실패시 오류 메시지 출력
          (아직 활성화 된 edge 남아 있는 경우 등)
- onClickNode()
    - 해당 Node에 대응되는 member의 individual(11) 페이지로 연결
- onClickEdge()
    - 해당 edge에 대응되는 Account(12) 페이지로 연결

### 3) Credit

- reverseCredit()
    - 해당 credit이 정산되었을 경우 삭제가 아니라 reverse credit을 생성

## Service

### 1) user_service

- sign_up(username:string, password:string)
    - 회원가입 Backend API 를 호출하여 그 결과를 return
    - error 시 error 메시지를 띄운다.
- sign_in(username:string, password:string)
    - 로그인 Backend API 를 호출하여 그 결과를 return
    - error 시 error 메시지를 띄운다.
- sign_out()
    - 로그아웃 Backend API를 호출하여 그 결과를 return
- get_user_by_id(id)
    - user를 id를 통해 GET Backend API 호출하여 가져옴
- info_update()
    - user의 정보를 update 하는 PUT Backend API 호출

### 2) member_service

- get_member_by_id(id)
    - member를 id를 통해 GET Backend API 호출하여 가져옴
- info_update()
    - member의 정보를 update 하는 PUT Backend API 호출

### 3) room_service

- get_member_list()
    - GET Backend API를 호출하여 room에 소속된 member들을 가져옴
- update_member()
    - member 추가나 삭제시 자동으로 호출되어 member_list 동기시킴
- get_payment_list()
    - GET Backend API를 호출하여 room에 연관된 payment를 가져옴
- update_payment()
    - payment 추가나 삭제시 자동으로 호출되어 payment_list 동기시킴
- get_room_by_id(id)
    - room을 id를 통해 GET Backend API를 호출하여 가져옴

### 4) payment_service

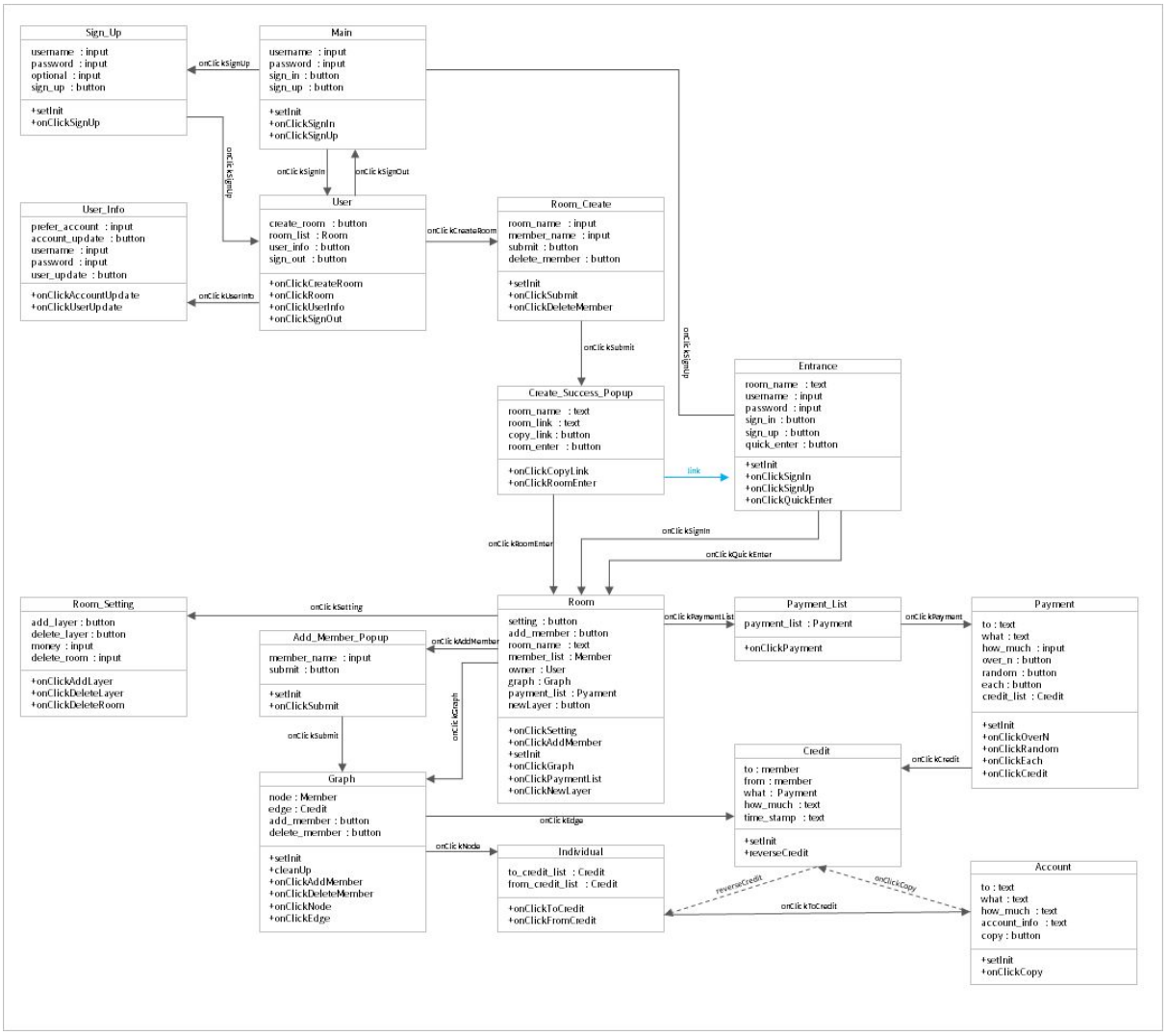
- get_credit_list()
    - GET Backend API를 호출하여 payment에 관련된 credit들을 가져옴
- get_to()
    - fromWho로 부터 누구에게 돈을 보내야할지 정보를 가져옴
- get_what()
    - forWhat으로 부터 무엇에 돈을 썼는지 정보를 가져옴
- get_total_how_much()
    - total로 부터 현재 Payment에 얼만큼 돈을 썼는지 가져옴
- update_payment()
    - PUT Backend API를 호출하여 변경 사항을 적용함

5) credit_service

- get_to()
    - 연관된 payment의 fromWho로 부터 정보를 가져옴
- get_from()
    - toWho로 부터 누가 돈을 보내야하는지 정보를 가져옴
- get_how_much()
    - amount로 부터 얼만큼 돈을 보내야하는지 정보를 가져옴
- get_credit_by_id(id)
    - GET Backend API로부터 credit을 가져옴
- update_credit
    - PUT Backend API를 호출하여 변경 사항을 적용함

## Frontend Relations

frontend의 component 간 관계는 다음 그림과 같다.




## 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.`

## 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`

## Authentication

User가 관여되는 행동(방 생성 등)에 대해서는 Authentication이 필요하다.
Django REST Framework에서 제공하는 Token-based Authentication를 사용할 계획이다.

https://www.django-rest-framework.org/api-guide/authentication/#tokenauthentication

## Implementation Plan

### Plan for Sprint 2 (23 Apr ~ 6 May)

회원가입 등 User 모델과 관련된 기능을 개발하는 것을 Sprint 2의 주요 목표로 둔다.
Sprint 2에서는 아래 다섯개의 User story를 구현할 예정이다.

- #1 Sign Up Page [장태준]
- #2 Main Page (#1 required) [장태준]
- #3 Login / User Page (#2 required) [이정민]
    - 회원가입 기능 구현 (POST user)
- #4 User Page (#3 required) [이정민]
    - 로그인 기능 구현 (POST user login)
- #7 User Info Page (#4 required) [정유석]
    - 유저 정보 수정 (PUT user login)

위 계획의 주요 리스크로는 사용자 인증(Authentication) 방식을 어떻게 할 것인지 결정하는 것에 있다. HW 2에서는 Basic Header을 이용한 Authentication을 이용하였으나 password를 Redux state로써 저장하는 것이 보안 상 우려되며 구조도 비효율적이라는 생각이 들어 앞서 Backend Design에서 명시한 대로 DRF 프레임워크에서 제공하는 Authentication 기능을 사용할 계획이다.

### Plan for Sprint 3

Room, Member 모델과 관련한 기능을 구현하는 것을 주요 목표로 두되, Sprint 2의 계획을 얼마나 이행했는지에 따라 추후 구체화한다.

### Plan for Sprint 4

Layer, Payment, Credit 모델과 관련한 기능을 구현하는 것을 주요 목표로 두되, Sprint 2의 계획을 얼마나 이행했는지에 따라 추후 구체화한다.

### Plan for Sprint 5

Frontend에 기술한 결제 상황 간소화 알고리즘 구현을 완료하고 정산 기능을 추가하는 것을 주요 목표로 두되, Sprint 2의 계획을 얼마나 이행했는지에 따라 추후 구체화한다.

## Testing Plan

Sprint 1 기준 현재 프로젝트에서 Boilerplating이 갓 끝난 때문에 구체적인 Testing Plan을 기술하기에는 어려움이 있어, 어떤 프레임워크를 사용할 것인지만 간단히 명시하였다.

### Backend

Django에서는 Python unittest 기반의 테스트 프레임워크를 제공하므로 이를 활용하고자 한다.

### Frontend

React/Redux 기반의 프론트엔드 소스 코드는 Jest라는 테스트 프레임워크를 활용하여 Unit Test를 수행한다. 현재 프로젝트에서 사용하고 있는 Atomic React boilerplate는 Jest를 이용한 unit test 예시가 준비되어 있어 이를 참고하여 테스트를 설계하고자 한다.

### Continuous Integration

지속적인 통합(Continuous Integration)을 실현하기 위해 Travis CI라는 도구를 사용한다.