<PROJECT 최종보고서>

Two of Scorched Earth

탈 중앙 검색엔진 서비스를 위한 웹 게임 개발

2014-11217 공은채

2018-16371 문보설

2014-11476 차우진

엄현상 교수님

Table of Contents

1. Abstract

탈 중앙 검색엔진 서비스란 검색 정보 제공자와 검색 정보 구매자가 1대 1로 직접 거래하는 상황에서 구매자의 보복 가능성을 추가하여 거래 신용도를 높이는 서비스를 말한다. 본 프로젝트는 이더리움 재단에서 개발 중에 있는 탈 중앙 검색엔진 서비스의 핵심 아이디어를 쉽게 소개하고, 서비스 오픈소스 접근성을 높이며 오픈소스 응용 사례를 홍보하는 것을 목표로 한다. 게임의 핵심 모듈로는 인트로 모듈, 역할 체험 모듈, 샌드박스 모듈과 크레딧 모듈이 있다. 인트로 모듈에서 게임 사용자들은 탈 중앙 검색엔진 서비스의 등장배경과 문제의식, 간단한 개념 소개 정보를 받으면서 서비스의 핵심 아이디어를 이해할 수 있다. 이어서 나오는 역할 체험 모드에서는 게임 사용자들은 탈 중앙 검색 엔진 서비스 내의 컨텐츠 제공자와 컨텐츠 구매자의 역할을 각각 경험하며 서비스 매커니즘을 이해할 수 있다. 다음으로 샌드박스 모드에서 게임 사용자들은 컨텐츠 거래에 영향을 미치는 factor들을 조정해 가며 거래 결과값을 확인할 수 있으며, 이를 통해 핵심 아이디어를 심화하여 이해할 수 있다. 마지막으로, credit 화면에서 게임 사용자들은 이더리움이 현재 개발 중인 탈 중앙 검색엔진 오픈소스의 링크와, 이 오픈소스가 도입된 사례를 볼 수 있다.

2. Introduction

이더리움 재단에서는 탈 중앙 검색엔진 서비스를 개발하면서, 개인 간 거래의 낮은 거래 신용도라는 단점을 극복하기 위해 Two of Two Scorched Earth라는 개념을 도입했다. 이는 기존 1대 1 거래 매커니즘에 구매자의 보복 가능성을 추가한 것으로, 이 서비스에서는 구매자는 서비스에 불만족 했을 경우 제안자의 deposit을 태워버릴 수 있다. 이러한 처벌 가능성을 통해 구매자의 거래 영향력을 키울 수 있으므로, 제안자는 좀 더 책임감 있고 높은 질의 서비스를 제공할 수 있게 된다.

우리 D조는 이 서비스 오픈소스의 현실적 파급력 문제와 서비스 자체의 합리성 문제에 있어 몇 가지 Pain Point들을 짚어냈고, 이를 해결할 수 있는 웹 게임 프로젝트를 진행하였다.

첫 번째 Pain Point는 탈 중앙 검색엔진 서비스의 배경 기술을 잘 알지 못하는 사람의 경우에는, 서비스의 핵심 아이디어를 이해하기 어렵다는 것이다. 우선 탈 중앙 검색엔진 프로젝트에 대한 설명을 얻을 수 있는 곳은 오픈소스 스펙 문서 밖에 없는데, 이 문서는 영어로 쓰여 있으며 상당히 높은 수준의 기술적 관점에서 쓰여져 있다. 따라서 블록체인 또는 탈 중앙 패러다임에 대한 지식이 없는 사람은 이를 이해할 수 없다. 또한 해당 프로젝트가 ‘Resent Behavior Assumption’ 등의 여러 전문적인 전제를 한다는 점에서, 배경지식이 없는 경우 프로젝트 기본 작동 방식을 이해하기 어렵다. 우리의 게임은 이 첫 번째 Pain Point를 극복하기 위해 인트로 모듈과 역할체험 모듈, 샌드박스 모듈을 제공한다. 먼저 인트로 모듈에서 게임 설명의 목적으로 탈 중앙 검색엔진 오픈소스의 문제 의식, 배경 지식 등을 설명한다. 그리고 역할체험 모듈에서는 거래에 등장 하는 각 역할을 체험하면서 프로젝트의 매커니즘을 이해할 수 있다. 샌드박스 모듈에서는 전지적으로 거래에 영향을 미칠 factor들을 조정하고 그 결과를 확인하면서 해당 오픈소스에 대한 이해를 심화할 수 있다.

두 번째 Pain Point는 오픈소스의 접근성이 낮고, 응용 사례를 찾기 어렵다는 것이다. 해당 오픈소스는 깃허브에 올라와 있는데, 어떤 홍보도 하지 않아 찾아보지 않는 이상 접근할 수 없다. 또한 해당 오픈소스의 실제 사용 사례가 언급되어 있지 않아, 오픈소스를 도입하려고 하는 사람이 장단점을 구체적으로 파악하기 어렵다. 우리는 이를 해결하기 위해 크레딧 모듈을 제공한다. 크레딧 모듈에서는 오픈소스 링크와, 오픈소스가 적용된 레퍼런스를 보여준다. 이를 통해 오픈소스의 낮은 접근성을 해결하고 그 활용 가능성을 홍보할 수 있다.

세 번째 Pain Point는 탈 중앙 검색엔진 프로젝트가 하고 있는 가정이 현실적이지 못한 경우가 많고, 현실적으로 적용가능한 범위가 제한되어 있다는 점이다. 탈 중앙 검색엔진 프로젝트는 기본적으로 'Resent Behavior Assumption'을 하고 있는데, 이는 경제학의 합리적 행위자 가정에 어긋난다. 'Resent Behavior Assumption'이란 거래 참여자가 자신의 손해를 감수하면서까지 상대방에게 보복을 하고자 할 것이라는 가정을 의미한다. 이는 거래 참여자가 경제적으로 합리적인 결정을 한다는 합리적 행위자 가정과 반대되는 것이다. 따라서 프로젝트와 같은 상황이 성립하는 경우는 매우 적을 것으로 생각된다. 예를 들어 거래가 이루어지는 재화가 구매자에게 필수불가결한 것이며, 거래 상황에서 제안자의 수가 극도로 적은 경우가 있을 수 있다. 하지만 이런 경우는 경험적으로 많지 않다. 또는 구매자의

정의감이 강한 경우에도 프로젝트와 같은 상황이 성립하나, 이런 경우는 매우 적다. 뿐만 아니라 해당 프로젝트가 효과적으로 작동하기 위해서는 구매자의 영향력이 상당히 커야 한다. 그렇지 않은 경우 정보 제공자에게 처벌의 영향이 크게 다가오지 않을 것이기 때문이다.

우리 프로젝트에서는 위와 같은 Pain Point를 해결하기 위해 샌드박스 모듈의 마지막에 Limitation 페이지를 추가하고, 해당 페이지에 프로젝트의 한계점들을 제시할 것이다.

3. Background Study

A. 관련 접근방법/기술 장단점 분석

이번 웹 서비스를 구축하는 과정에서는 페이스북에서 제공해주는 자바스크립트 라이브러리인 React를 활용하였다. Vue, Angular 등 다른 프론트엔드 프레임워크/라이브러리가 아닌 React를 사용한 이유는 다음과 같다. 첫 번째는 확장성이다. 최근 프론트엔드 개발 흐름의 주도권을 React가 가지고 있어서 인터넷 등에 가장 많은 정보가 있으며, 많은 이더리움 관련 웹 프로젝트가 React 기반의 개발을 하고 있다. 또한, 구현하는 게임이 가볍고 현대적이며 유저 친화적이어야 하기 때문에, Angular나 Vue가 아닌 React를 사용하게 되었다. 더 자세한 설명은 6-B에서 확인할 수 있다.

또한 컴포넌트를 재 사용하여 가볍고 효율적인 코드를 만들기 위하여 React를 선택한 부분도 존재한다. 해당 웹 게임 서비스에서는 기술적인 면 뿐만 아니라 비주얼 디자인을 프론트엔드 개발에 연결하는 측면에서도 난이도가 낮지 않은 부분들이 많았는데, React를 사용하면 UI Component를 구현하거나 유지/보수하는 과정에서 유리한 점이 많을 것이라 판단했다. 또한 9-B에 나온 Implmentation Issue 중 package 관련된 경험을 통하여, 외부 package를 거의 사용하지 않은 확장성 좋은 코드를 만들기 위하여 노력하였다.

B. 프로젝트 개발환경

웹 서비스를 완성도 높게 구축하는 점에서는 협업이 매우 중요하다고 판단하여 Git을 효율적으로 활용해 로컬저장소와 브랜치를 통한 효율적인 개발 프로세스를 진행하였다. 원격 저장소로는 Github을 사용하며, Github Issue 등의 기능을 활용하며 모든 프로젝트 참여자가 VSCode를 활용하여 효율적인 협업 과정을 진행하였다.

그리고 웹 게임 개발을 진행하는 과정에 있어 HTML, CSS, Javascript, 그리고 React.js를 활용했다. 또한 비주얼 UI 디자인 부분은 피그마와 어도비 일러스트레이터를 통해서 진행하였다. 그리고 기업 담당자와의 소통을 위하여 Discord를 활용했다.

4. Goal/Problem & Requirements

이번 프로젝트의 목표는 탈 중앙 검색엔진 서비스의 여러 Pain Point들을 보완할 수 있는 웹 기반 게임을 제작하는 것이다. 앞서 살펴본 바와 동일하게, 여기서 Pain Point는 이해 난이도가 높은 것, 오픈소스 접근성이 낮은 것, 합리성에 대한 고려가 필요한 것이다. 우리의 게임은 인트로 모듈, 역할 체험 모듈, 샌드박스 모듈을 통해 이해를 돕고, 아웃트로 모듈을 통해 오픈소스 접근성을 높이고, 크레딧 모듈에 Limitation 페이지를 추가하여 합리성에 대한 고려도 증진하는 것을 목표로 한다.

이를 위해서 우선 탈 중앙 검색엔진 서비스의 기본 매커니즘과 핵심 아이디어를 게임을 통해 쉽게 전달할 수 있어야 하며, 게임에 프로젝트에 대한 정확한 이해를 반영해야 한다. 또한 사용자가 쉽게 접근할 수 있는 UI 디자인도 중요하다.

5. Approach

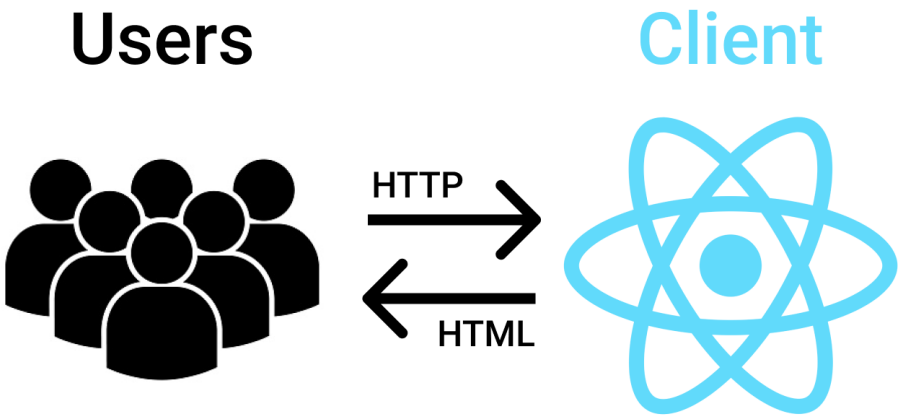
해당 웹 서비스의 목표는 탈 중앙 검색엔진 서비스에 대한 이해도 및 접근성을 높이는 게임을 만드는 것이다. 효율적으로 목표를 달성하기 위해 인트로 모듈, 역할 체험 모듈, 샌드박스 모듈, 아웃트로 모듈 네 가지 기능을 게임 내에서 제공하고자 한다. 네 가지 모듈에 대한 설명은 다음 문단에서 더 자세하게 설명할 것이다.

우선 역할 체험 모듈에서는 게임 유저가 역할 및 행동을 선택하는 과정에서 클릭하는 버튼들에 따라 적절한 페이지를 보여줘야 한다. 이 부분은 React의 Event Handler를 적절히 활용하여, 사용자가 화면에서 누른 버튼에 따라서 적절한 페이지로 라우팅한다. 샌드박스 모듈에서는 게임 유저의 Input을 받아 시뮬레이션을 수행하고, 그 결과값을 산출하는 시뮬레이션 코드가 필요하다. 시뮬레이션 코드를 통해서 실시간으로 전달되는 데이터 시각화 과정을 통해서 서비스를 이용하는 유저의 이해도를 증진시키기 위해 노력했다. 효율적인 UX 설계를 위하여 샌드박스 모듈은 그 역할에 따라 시뮬레이션을 수행하는 부분과 화면 출력 및 사용자 입력을 관리하는 부분으로 나누어 구성되었다.

프로젝트 구현에서 백엔드 없이 프론트엔드 React만 사용한다. 게임 시뮬레이션에 사용자 인증 등의 기능이 없기 때문에 백엔드 데이터베이스가 필요하지 않다고 판단하여, 프론트엔드만 활용한 SPA(단일 페이지 애플리케이션)를 만들기로 결정했다. 또한, 직관적이고 깔끔하면서도 유저 입장에서 심플하게 내용을 받아들일 수 있는 디자인이 중요한 요소라고 판단하였다. 그렇기 때문에, 프로젝트에 적합한 폰트나 레이아웃을 먼저 구성한 다음, 직접 손그림으로 귀여운 캐릭터들을 디자인하여 사용자들의 심리적 만족감을 높이고자 하였다. 이렇게 만들어진 게임을 최종 발표를 하는 시점에는 게임을 실시간으로 실행하는 모습을 보여주고자 한다.

6. Project Architecture

A. Architecture Diagram



B. Architecture Description

프론트엔드에서는 페이스북에서 제공해주는 프론트엔드 라이브러리인 React.js를 사용했다. Vue, Angular, Svelte 등 다른 프레임워크/라이브러리가 아닌 React.js를 선택한 이유는 다음과 같다. 이 게임 프로젝트는 짧은 기간 안에 가벼우면서도 현대적인 애플리케이션을 개발해야 하기 때문에, 기능이 다양하고 규모가 큰 애플리케이션을 만들 때 유리한 Angular보다는 React를 선택했다. Vue나 Svelte도 작고 가벼운 애플리케이션을 빠르게 개발하기에 유리한 면이 많지만, Ethereum Foundation의 Scorched Earth 프로젝트가 React를 기반으로 개발이 되었다는 점과 2021년 현재 웹 개발 커뮤니티에서 가장 많이 활용되는 javascript library라는 React의 특성 때문에 React를 선택하게 되었다.

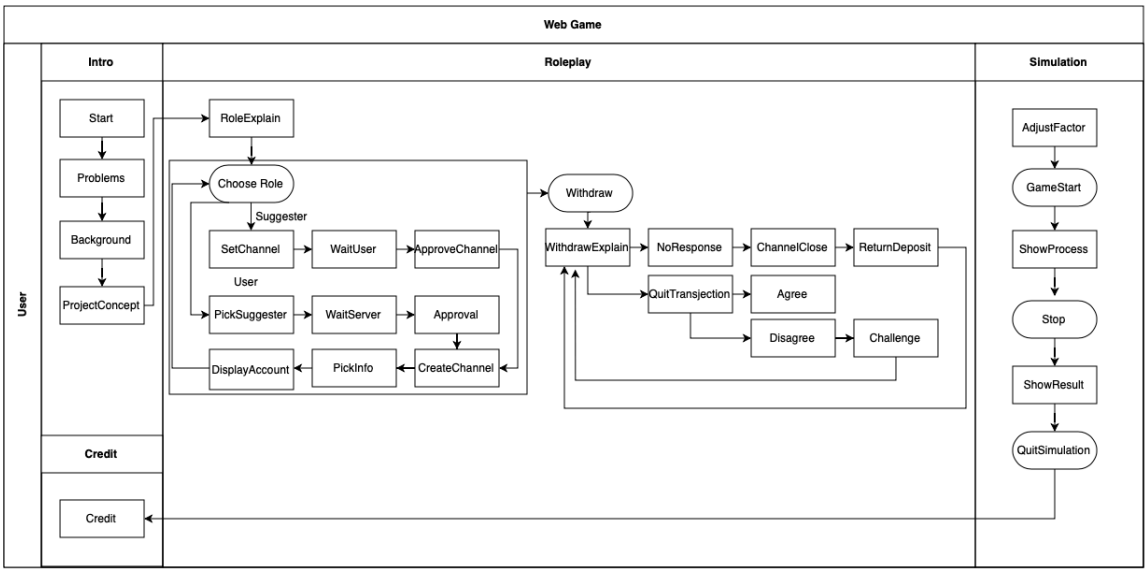
그리고 다양한 UI 디자인 요소가 해당 웹 게임 프로젝트에 들어가기 때문에, React.js를 활용하면 컴포넌트를 활용하여 간편하게 UI를 수정하고 재사용이 가능할 것이라고 판단을 하였다. React.js의 컴포넌트를 활용하면 헤더, 메인 콘텐츠, 버튼, 사이드바 메뉴, 캐릭터 등을 헤더 컴포넌트, 사이드바 컴포넌트와 같이 하나의 컴포넌트로 묶어서 관리할 수 있다. 그렇기 때문에, 컴포넌트를 활용하여 코드의 재사용성과 유지보수성을 증가시켜 줄 계획이다. 이러한 특성들 때문에, React를 사용하면 효율적으로 프로젝트 개발 및 관리를 진행할 수 있다고 판단하였다.

기업 담당자와의 협의 후, 서비스의 Pain Point와 관련된 사용자 경험을 개선하는 프론트엔드 개발에 집중하기로 결정하였다. 우리가 발견한 Pain Point들은 기업에서 진행하는 서비스와는 독립된 속성들(접근성, 이해 난이도)이거나, 서비스 패러다임 자체를 바꾸어야 하는 속성들(아이디어 합리성)이었다. 그렇기 때문에 기업에서 진행하는 서비스에 직접적으로 참여하는 것보다는 자체적인 게임 서비스를 이번 과목 프로젝트에서 효율적으로 진행하고자 smart contract를 연결하지 않고, 웹 프론트엔드 위주의 개발에 집중하기로 담당자와 합의했다.

또한 이번 프로젝트에서 샌드박스 모듈을 구현하기 위해 필요한 시뮬레이션 코드는 데이터 축적을 필요로 하지 않으므로 백엔드 서버 없이 프론트엔드 단에서 구현을 진행하기로 결정했다. 샌드박스 모듈에서 사용자와 인터랙션이 진행되는 결과 화면은 매 시행의 독립적인 결과 치를 보여주기 때문에, 이전 시행에 대한 정보를 필요로 하지 않는다. 이러한 특성 때문에 시뮬레이션 코드는 프론트엔드 단에서 돌아가며 결과값을 관리하는 것이 더 적절하다고 판단하여, React 및 Javascript를 활용하여 모든 시뮬레이션 코드 개발을 진행했다.

7. Implementation Spec

A. Modules



해당 프로젝트는 인트로, 역할 체험, 샌드박스, 아웃트로의 총 네 가지 모듈로 구성되어 있다.

시작 페이지 이후 바로 진입하게 되는 인트로 모듈에서는 현재 문제점을 지적하고 프로젝트 배경 및 프로젝트가 어떠한 컨셉 하에 논의가 되는지 소개한다.

이후 진입하게 되는 역할 체험 모듈에서 게임 사용자는 User와 Suggester 역할을 체험해 볼 수 있다. 역할 별로 스마트 컨트랙트 채널을 형성하는 Phase0와 형성된 채널을 기반으로 거래를 진행하는 Phase1를 각각 경험할 수 있다.

User의 역할을 선택한 경우, Phase0에서 User는 Suggester를 선택하고 Deposit을 예치하는 과정을 통해 스마트 컨트랙트 채널을 형성한다. 이후 Phase1에서 User는 Suggester에게 이미지를 받고 보상할지 처벌할지 의사결정을 해야한다. 어떤 의사결정을 하였는지에 따라 자동으로 User와 Suggester의 Deposit 및 Suggester의 평판이 변화한다.

Suggester의 역할을 선택한 경우, Phase0에서 Suggester는 스마트 컨트랙트 채널에 예치할 Deposit을 결정한 뒤 특정 User로부터 요청이 올 때까지 기다리는 시간을 거쳐야한다. User에게 요청이 오면 Suggester는 User 정보를 기반으로 채널 승인 여부를 결정한다. Suggester가 채널 승인을 한 경우만 스마트 컨트랙트 채널이 실질적으로 형성되어 실제 거래를 수행하는 Phase1로 진입할 수 있다. Phase1에서 Suggester는 User에게 양질의 이미지를 보낼지, 좋지 않은 이미지를 보낼지 의사결정을 해야하며, 그 결과 User 내린 보상 혹은 처벌 결정에 따라 Deposit, Reputation이 자동으로 변화된다.

각 역할 체험 후, 게임은 어떠한 조건이 있을 때 거래가 중단이 되는지 간략히 소개한다.

앞선 역할 체험 모듈이 각각의 역할을 체험하면서 오픈소스의 기본적인 매커니즘을 이해하는 것을 그 목적으로 하고 있다면, 샌드박스 모듈은 각각의 변수를 조정하여 전지적 시점에서 오픈소스에 대한 이해를 심화하는 것을 목적으로 한다. 샌드박스 모듈에서는 시뮬레이션에 거래를 미치는 인구 변수와 환경 변수를 직접 조정하여 결과를 실시간으로 확인할 수 있다. 우리는 Suggester를 각각 Easygoing, Tactful 유형으로 User를 RiskTaker, Coward 유형으로 분리하여 이들의 인구수를 조정할 수 있는 인구변수 탭을 구현하였다. Easygoing은 User가 그들을 처벌하더라도 서비스 수준을 올리지 않는 Suggester이며, Tactful은 User가 그들을 처벌하면 서비스 수준을 올리는 Suggester다. Risktaker는 Suggester가 낮은 수준의 서비스를 제공했을 때 Suggester를 Punish하는 User이며, Coward는 Suggester가 낮은 수준의 서비스를 제공하더라도 Punish하지 않는 User이다. 그리고 환경 변수 탭에서는 스마트 컨트랙트 채널을 형성할 때 Suggester와 User가 어느 만큼의 금액을 예치할 것인지 Deposit을 각각 조정할 수 있도록 구현하였다.

아웃트로 모듈에서는 오픈 소스 프로젝트가 자체적으로 내재하고 있는 한계점에 대한 소개가 뒤따른다. 이후 이어지는 엔딩 페이지에서는 프로젝트 코드 및 오픈소스 코드가 담긴 Github 링크가 제공되어 Two of Scorched Earth 프로젝트의 확장성을 증대 시키려 했다.

B. Input/Output Interface

1) Input

이번 프로젝트에서는 게임사용자의 편의성 증대 및 페이지간 일관성을 부여하기 위해 모든 사용자 인풋을 클릭과 키보드 입력으로 통일하였다.

프로젝트에 관한 아이디어, 배경 및 개념에 대해 소개하는 페이지에서는 각 컴포넌트의 state에서 각 버튼 클릭 값을 여러 변수로 관리하고 있기 때문에 어떤 버튼을 클릭을 하느냐에 따라 페이지가 새로 렌더링이 되거나 다음 페이지로 넘어갈 수 있다.

역할 체험과 관련된 페이지에서 기존 페이지 내부에서 새로 렌더링하는 경우, 컴포넌트 내부의 state 값을 기준으로 visibility를 조정하였다. 게임 사용자가 페이지 내부 설명과 달리 인풋 필드에 정해진 실수 범위 외의 값을 입력하는 경우 alert 메시지를 통해 행동이 제한된다. 따라서 게임사용자는 게임에서 의도하는 일련의 과정을 따르지 않으면 다음 페이지로 가는 버튼이 활성화되지 않아 다음 페이지로 이동할 수 없다. 페이지를 이동하는 경우, 기존 컴포넌트 상태값이 소멸된다. 따라서 이전 페이지에서 기입한 사용자 인풋을 this.props.history.push() 함수 내부의 파라미터로 넘겨주는 방식을 취해 이를 해결하였다.

위 과정을 통해 게임 사용자가 프로젝트의 개념과 각 주체간 상호작용 과정에 대해 이해했다면, 그 후 사용자는 샌드박스 모듈 내에서 시뮬레이션 코드에 파라미터로 전달될 여섯 가지의 인풋값을 조정하게 된다. 해당 인풋값은 크게 인구 변수와 환경 변수로 분리할 수 있다.

- 인구 변수: RiskTaker 인구, Coward 인구, Easygoing 인구, Tactful 인구

- 환경 변수: 스마트 컨트랙트 시 User Deposit 양, Suggester Deposit 양

이 때 수치화 할 수 있는 인풋값은 게임 사용자가 input 필드에 입력하는 값으로 특정 범위로 규정되어 있어, 그 외의 값이 들어올 경우 alert을 통해 게임 사용자에게 올바른 인풋값을 입력 하기를 요구하고 있다. 샌드박스 페이지에서 시뮬레이션을 시작할 경우, 각 인풋값들이 시뮬레이션 코드에 정수값 형태로 변환되어 전달되고, 시뮬레이션 중 매 라운드 상태값들이 <Sandbox/> 컴포넌트 내부에서 변화한다.

각 페이지 하단의 <Footer/> 컴포넌트에서도 상태값과 history.push()를 통해 페이지 라우팅 기능을 제공한다. 일반 <Router/> 컴포넌트의 경우 페이지 이동 히스토리가 저장되지 않아 브라우저 내부 뒤로가기가 제대로 동작하지 않기 때문에, 맥락이 중요한 게임이라는 점에서 history 함수를 사용하였다.

2) Output

해당 프로젝트의 아웃풋은 샌드박스 모듈을 제외하고 모두 컴포넌트 내부에서 호출된 render() 함수의 결과값으로 통일되어있다. render() 함수는 DOM에 html을 내보낼 때 호출되거나 컴포넌트에서 setState() 함수가 실행되어 자동 렌더링 될 때마다 실행될 수 있다.

샌드박스 모듈에서는 시뮬레이션과 관련한 setEnvironment(), checkAlive(), callMatch(), setMatch() 함수의 수행 결과 변화되는 <Sandbox/> 컴포넌트 내부 SuggesterDeposit, UserDeposit, Connect, SuggesterAlive, UserAlive 배열을 기반으로 <RiskTaker/>, <Coward/>, <Easygoing/>, <Tactful/> 컴포넌트를 렌더링한다. SuggesterDeposit, UserDeposit, Connect, SuggesterAlive, UserAlive 배열에 대한 자세한 설명은 다음 문단에 기술되어 있다.

SuggesterDeposit[]은 각 Index에 대응되는 Suggester의 남은 Deposit을 기록한다. Suggester의 Index는 0부터 시작해서 전체 Suggester 수까지 Easygoing부터 Tactful 순서로 주어진다. UserDeposit[]은 각 Index에 대응되는 User의 남은 Deposit을 기록한다. User의 Index는 0부터 시작해서 전체 User 수까지 Risktaker부터 Coward 순서로 주어진다. Connect[]는 각 Index에 대응되는 User와 연결된 Suggester의 Index를 기록한다. Suggester는 앞서 주어진 Index로 서로 구분된다. 만약 해당 index의 User가 어느 Suggester와도 연결되지 않은 경우 -1을 기록한다. SuggesterAlive[]는 각 Index에 대응하는 Suggester의 파산 여부를 기록한다. 파산한 경우 1이 기록되고 파산하지 않은 경우 0이 기록한다. UserAlive[]는 각 Index에 대응하는 User의 파산 여부를 기록한다. 파산한 경우 1이 기록되고 파산하지 않은 경우 0이 기록한다. Punish[]는 각 Index에 대응하는 User가 거래 상대방을 처벌 했는지를 나타낸다. 만약 User가 현재 라운드에서 거래 상대방을 처벌을 했다면 1로 기록하고, 아닌 경우 0으로 기록한다.

샌드박스 모듈 내부에서 변화한 아웃풋은 실시간으로 샌드박스 페이지에 렌더링이 되어 게임 이용자는 각 라운드 별 Users, Suggesters에게 어떠한 변화가 생기는지 확인할 수 있다. 매번 setState()를 통해 페이지가 렌더링 된 이후 기존에 있던 상태값은 사라진다는 문제를 해결하기 위해 우리는 매 라운드의 결과를 console 창에 기록하였다. 따라서 게임 이용자는 브라우저의 Inspect>console에 들어간다면 매 라운드 별 배열의 변화값을 모두 확인 할 수 있다.

C. Inter Module Communication Interface

해당 프로젝트 내에서 모듈간 커뮤니케이션은 페이지 전환이라는 가장 간단한 형태로 발생한다. 각 모듈 내에서 요구되는 프로세스를 모두 거치면 컴포넌트 내부의 state 값이 변화하면서 다음 모듈로 갈 수 있는 next 버튼이 활성화되며 모듈 전환이 발생한다. 단, <Footer/> 컴포넌트의 버튼을 사용할 경우, 프로세스를 거치지 않더라도 어떤 모듈에서건 다른 모듈로 이동할 수 있다.

중간보고서에서는 시뮬레이션 코드와 샌드박스 모듈을 분리 구현하여 모듈간 커뮤니케이션을 구현하고자 하였으나, 시뮬레이션 코드에서 다루는 변수들이 React 모듈 내부에서 충분히 처리할 수 있는 규모로 간결화되었다. 시뮬레이

션 코드가 샌드박스 모듈 내부의 함수 형태로 변환되며 모듈간 상호 커뮤니케이션이 불필요해졌기 때문이다.

8. Solution

A. Implementations Details

1) 렌더 관련 함수들

clickNextHandler : 다음 페이지로 가기 위한 버튼을 클릭하였을 때 페이지 이동 히스토리를 추가, 이동하는 함수

clickUserHandler : 역할 체험 단계에서 유저 페이즈로 이동하는 함수

clickSuggesterHandler: UserPhase0에서 선택된 Suggester를 기반으로 state를 변화시키는 함수

loadingHandler : 스마트 컨트랙트 채널을 형성 시간동안 User가 대기하도록 하는 함수

startTransferHandler : 채널 형성 및 거래 시작 조건을 User가 만족하지 못했을 경우를 판단하고 alert 메시지를 띄우는 함수 / SuggesterPhase0에서 요구되는 모든 프로세스를 수행한 경우를 판단하여 Phase1으로 넘어갈 수 있도록 하는 함수

initState : UserPhase0의 인풋값을 url에서 추출하여 채널 초기값을 세팅하는 함수/ SuggesterPhase0의 인풋값을 url에서 추출하여 채널 초기값을 세팅하는 함수

clickSatisfyHandler : User가 Suggester로부터 받은 이미지에 만족하는지 불만족하는지에 따라 User Deposit, Suggester Deposit, Suggester Reputation을 변화시키는 함수

experienceSuggesterHandler : Userphase1에서 수행해야하는 모든 프로세스를 완료한 경우 Suggester 체험 페이지로 이동시키는 함수

settingHandler : SuggesterPhase0에서 채널이 성공적으로 형성된 경우 채널 형성 당시의 인풋값을 확인하는 메시지를 출력하는 함수

clickSendHandler : Suggester가 어떤 품질의 이미지를 보냈느냐에 따라 User의 응답값을 조정하여 User Deposit, Suggester Deposit, Suggester Reputation을 변하시키는 함수

clickStartSimulationHandler : 샌드박스 모듈에서 게임 사용자가 조정할 수 있는 인구 변수와 환경 변수가 주어진 제한 범위를 넘어서지 않았는지 체크하고 시뮬레이션 코드를 시행하는 함수

final : 시뮬레이션이 종료되는 조건을 확인하고 종료시 최종 상태를 렌더링하는 함수

checkResetHandler : 게임 사용자가 시뮬레이션을 다시 시작할 수 있도록 샌드박스 모듈의 모든 state를 초기값으로 세팅하는 함수

2) 시뮬레이션 내부 함수

setEnviroment : 게임 이용자가 조정한 population, deposit 값을 이용해 Suggester들과 User들의 기본값을 설정하는 함수

callMatch : 0.5초 이후 match를 수행하는 함수

match : 각 User와 그 User에 배정된 Suggester 간의 거래를 수행하는 함수

checkAlive : User가 모두 파산했거나 Suggester가 모두 파산한 경우를 체크하는 함수

final : 시뮬레이션을 종료시키는 함수

B. Implementations Issues

1) 샌드박스 모드 factor 조정 범위 설정 이슈

시뮬레이션에서는 사용자가 조정할 수 있는 파라미터의 범위를 설정하는 것이 중요하다. 먼저 Population Tab에서는 화면 디자인을 고려하여 User의 수가 도합 10명을 넘어가지 않도록 각각 5명을 max 값으로 설정했다. 그리고 Suggester의 총 합이 Risktaker보다 적을 수 있도록 Suggester의 수가 3을 넘어가지 않게 설정 했는데, 이에 대한 이유는 10.B에 나와 있다. Deposit에 관해서는 min 값을 20으로 설정 했을 때, User의 Deposit이 Suggester의 deposit보다 많지 않은 경우 Easygoing 제안자가 파산되지 않는다는 문제가 있었다. 이는 기본적으로 User가 자신의 돈을 희생하여 Suggester를 처벌하는 것이기 때문에 당연히 발생하는 일이고, 오픈소스 개념의 한계에 해당한다. 따라서 이 부분은 User의 deposit 범위의 min을 Suggester의 deposit 범위의 max값으로 설정하고, 그 이유를 Limitation에 기입하는 방식으로 해결하였다.

2) React 외부 패키지를 사용하여 문제가 발생한 이슈

React 개발 환경에서 효율적인 데이터 시각화 작업을 통하여 외부 패키지를 다운로드하는 과정에서 문제가 발생했다. 변화 속도가 빠른 웹 프론트엔드 개발 환경 특성상, 외부 패키지가 변화할 때마다 버전 문제가 생기는 경우가 많은데, 해당 프로젝트에서도 lineto라는 외부 npm package에서 문제가 발생하였다. 이러한 과정은 지속적으로 관리하기 편리한 코드를 작성하고, 서비스의 안정성을 높이고자 외부 패키지를 사용하지 않고 개발 과정을 진행하게 되는 계기가 되었다.

3) 시뮬레이션 결과값을 중간에 받아올 수 없어서 발생한 이슈

중간 보고서 당시 계획은 시뮬레이션 코드를 바닐라 js로 구현하고, 이를 리액트로 구현된 샌드박스 모듈에서 불러와 시뮬레이션을 수행하고 결과값을 랜더링하는 것이었다. 그러나 시뮬레이션의 각 라운드마다 중간 결과값을 보여주는 것이 더 직관적이기 때문에, 시뮬레이션 중간의 값을 받아오기 위해 바닐라 Js 코드를 리액트에서 사용 가능한 Jsx 코드로 변환하기로 했다. 이를 위해 시뮬레이션에서 사용하는 Suggester, User 등의 클래스 변수를 어레이 변수로 바꾸었고, 리액트에서 각 클래스에 대응되는 컴포넌트를 구현하여 값을 업데이트하는 방식을 택했다. 그리고 function으로 정의된 여러 함수들을 화살표 함수로 변경했다.

9. Results

A. Experiments

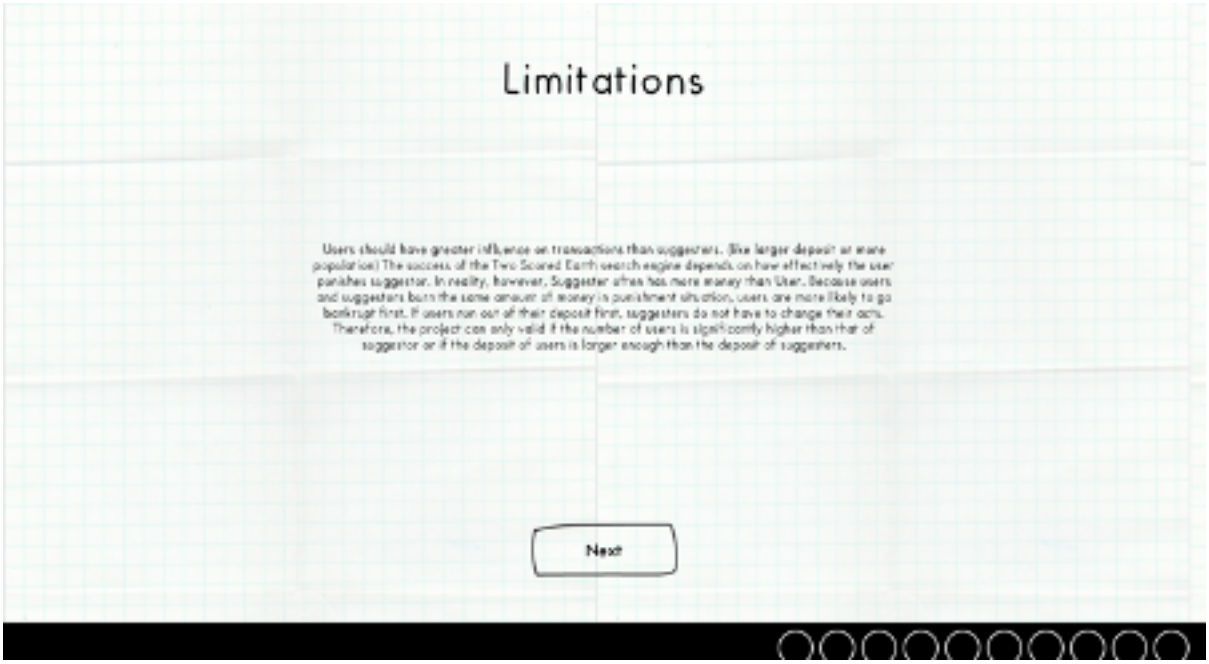
```
Match:: #Round: 7                                    Sandbox.js:266
        Suggester Deposits : 4,46
        User Deposits : 52,62,64,66,66
        Connect : 0,1,1,0,1
        Punish : 1,0,0,0,0
        Suggester Alive : 1,1
        User Alive : 1,1,1,1,1

Match:: #Round: 8                                    Sandbox.js:266
        Suggester Deposits : 0,54
        User Deposits : 48,60,62,64,64
        Connect : 0,1,1,1,1
        Punish : 1,0,0,0,0
        Suggester Alive : 1,1
        User Alive : 1,1,1,1,1

Match:: #Round: 9                                    Sandbox.js:266
        Suggester Deposits : -4,62
        User Deposits : 44,58,60,62,62
        Connect : 0,1,1,1,1
```

B. Result Analysis and Discussion

시뮬레이션을 구현하면서 오픈소스 프로젝트의 유효성을 알아보기 위해 다양한 거래 당사자를 투입했다. 그 종류는 다음과 같다. 첫 번째로 Suggester는 전통적 제안자인 Easygoing과 프로젝트 상황에서 가정하는 제안자인 Tactful로 나뉜다. Easygoing은 전통적 전자 상거래에서의 Suggester과 마찬가지로 구매자의 반응과 관련없이 좋지 않은 서비스를 제공한다. 반면, Tactful은 프로젝트 상황에서 가정한 것과 마찬가지로 구매자가 punish를 할 경우에 더 좋은 서비스를 제공한다. 이 둘의 비교를 통해 오픈소스가 도입된 decentralized content recommendation 모델에서는 Tactful이 Easygoing보다 더 잘 살아남는다는 것을 보여주고자 했다. 그리고 User는 전통적 구매자인 Coward와 프로젝트 상황이 가정한 구매자인 Risktaker로 나뉜다. Coward는 전통적 구매자와 같이 나쁜 서비스를 받아도 punish하지 않는 User다. 반면 Risktaker는 오픈소스 프로젝트의 가정에 부합하는 구매자로, 나쁜 서비스를 받으면 반드시 punish하는 구매자이다. 이 둘의 비교를 통해 오픈소스가 도입된 상황에서는 Risktaker가 Cow-

ard보다 영향력이 클 때 Easygoing을 파산시킬 수 있고, 결론적으로 더 좋은 서비스를 제공하려고 하는 Suggester가 살아남는다는 사실을 보여줄 수 있다.

구현 이후 위와 같은 로그를 확인할 수 있었다. 예상했던 것과 같이, 오픈소스 상황과 같은 상황에서는 Tactful이 Easygoing 보다 오래 살아남는다는 것을 알 수 있었다. Coward와 Risktaker 간의 차이는 Tactful과 Easygoing 간의 차이로 인해 간섭을 받아 의도한 결과를 보여주지 않을 가능성이 있었다. 예를 들어 Risktaker가 Coward보다 적을 때, Easygoing이 Tactful보다 너무 적어서 Easygoing이 전부 파산되는 결과가 나온다면 충분히 시뮬레이션을 돌리지 않은 게임 이용자의 경우에는 Risktaker의 수가 많아야 한다는 점을 알아채지 못할 수 있다. 따라서 이 부분은 Enviroment 설정에서 반드시 Risktaker가 전체 Suggester보다 많아야 한다는 식으로 제한을 주고, 이후 이런 제한을 왜 도입했는지 Limitation 페이지에서 설명하는 방식으로 해결하였다.



10.Division & Assignment of Work

항목	담당자
게임 설계	공은채, 차우진, 문보설
시뮬레이션 코드 구현	문보설
CSS	차우진
UX 설계 및 UI 구성	차우진
캐릭터 및 화면 디자인	차우진
인트로 모듈 구현	공은채
크레딧 모듈 구현	공은채
역할체험 모듈 구현	공은채
샌드박스 모듈 구현	공은채
아웃트로 모듈 구현	공은채
샌드박스 모듈과 시뮬레이션 코드 연결	공은채, 문보설
시뮬레이션 관련 리서치	문보설
프로젝트 한계점 정리	문보설

11.Conclusion

프로젝트 결과 오픈소스의 Pain Point들을 해결할 수 있는 웹 게임을 개발할 수 있었다. 우선 인트로 모듈과 역할 체험 모듈, 샌드박스 모듈은 오픈소스 매커니즘을 잘 이해할 수 있게 구현되었다. 또한 그 과정에서 발견할 수 있는 오픈소스 한계점을 따로 정리하여 그 합리성에 대한 의미있는 논의를 할 수 있었다. 그리고 마지막에 오픈소스와 우리 프로젝트 github 링크를 걸어서 접근성 문제도 해결할 수 있었다.

◆ [Appendix] User Manual

프로젝트의 코드를 확인할 수 있는 github link는 다음과 같다.

https://github.com/ericagong/SNU-project

프로젝트를 다운로드 받아서 구동하기 위해서는 하단과 같은 명령어들을 터미널에서 순서대로 실행하면 된다. 그 전에 npm과 yarn이 로컬 환경에 설치가 되어 있어야 한다.

git clone https://github.com/ericagong/SNU-project.git

cd SNU-project

yarn

yarn start