4차 3기 실무 Project

|  |
| --- |
| 블록마블 |

**2018년 10월 31일**

**블록체인 기반 핀테크 전문가 양성과정**

**블록마블**

양흥영

윤기석

이남호

정용수

조상훈

함용재

허경환

**목 차**

**1. 프로젝트 개요**  **3**

1.1 프로젝트 기획 배경 및 목표 3

1.2 구성원 및 역할 4

1.3 프로젝트 추진 일정 4

**2. 프로젝트 현황**  **5**

2.1 시장 분석 5

2.2 경쟁 제품 장단점 분석 5

2.3 차별화 핵심 전략 기술 5

**3. 프로젝트 개발 결과**  **7**

3.1 주요 기능 / 주요 다이어그램 7

3.2 핵심 알고리즘 8

3.3 상세 동작 18

**4. 기대 효과 및 향후 개선사항**  **21**

**5. 개발 후기**  **22**

**6. 강사 의견**  **25**

**1. 프로젝트 개요:**

1.1. 프로젝트 기획 배경 및 목표

4차 산업혁명 선도인력 양성과정 블록체인 수업을 수강하며 go-ethereum, metamask, remix.ethereum 등의 툴을 활용한 스마트콘트랙트 생성과 배포에 대한 기술을 습득했다. 팀원들의 의견을 조율한 결과 블록체인과 게임을 접목한 프로젝트를 구상하게 되었다.

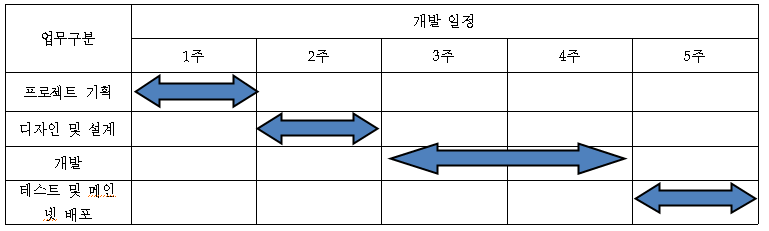
게임에 블록체인을 도입하게 된다면 크게 네 가지 이점이 있을 수 있다. 첫 번째로 게임 데이터 저장과 보존을 위한 안전한 환경을 유저들에게 제공할 수 있다. 두 번째로 소규모 게임 개발 회사들이 고객들을 접근하는 것이 더 쉬워지므로 더욱 다양한 게임개발 업체들이 생겨날 것으로 보인다. 세 번째로 게임 내 결제 시스템을 블록체인 상에 운영하게 되면 수금과정이 더욱 간편해지는 이점이 있다. 마지막으로 게임 내 가상 아이템들을 유동적으로 사고 파는 시장이 형성되어 가상세계의 물건이 현실세계에서도 가치를 지니게 될 수 있다.

지금껏 로직이 복잡한 디앱 기반 게임들은 TPS의 한계로 인해 개발된 사례가 적다. 그러나 '블루마블' 같은 류의 보드게임들은 로직의 복잡도를 통한 재미를 낮은 tps로도 구현이 가능하기 때문에 디앱 게임의 다양성을 증진시킬 수 있을 것이라고 판단했다. 서버의 역할을 최소화 하여 완전히 분산된 형태의 어플리케이션을 개발한다는 데에 의미가 있으며, 프로젝트에서 서버는 실제로 클라이언트 페이지를 배포하는 역할만 한다. 본 프로젝트의 서버에서 제공하지 않는 다른 클라이언트 페이지를 컨트랙트와 연동시켜 아무 문제가 없게끔 인터페이스 등을 github를 통해 모두에게 공개할 예정이다. 그래서 최종 프로젝트의 목표는 “블록체인에 호스팅 되며 creator의 임의로 커스터마이징 가능한 ‘브루마블’ 보드게임 플랫폼 서비스’를 제공하는 것 이다.

1.2. 구성원 및 역할

|  |  |  |  |
| --- | --- | --- | --- |
| 이름 | 전공 | 역할 | 구현 부분 |
| 윤기석 | 컴퓨터공학과 | 팀장 | 프로젝트 관리 |
| 양흥영 | 컴퓨터공학과 | 팀원 | Game Lobby UI 개발 |
| 조상훈 | 컴퓨터공학과 | 팀원 | Game Play UI 개발 |
| 정용수 | 전자공학과 | 팀원 | Create Game UI 개발 |
| 함용재 | 전자통신공학과 | 팀원 | Game Play UI 개발 |
| 허경환 | 경영학과 | 팀원 | Create game UI 개발 |
| 이남호 | 경영학과 | 팀원 | 핵심 알고리즘 개발 |

1.3. 프로젝트 추진일정



**2. 프로젝트 현황**

**2.1. 시장 분석**

Newzoo에서 출간한 보고서에 따르면 글로벌 게임산업을2016년부터 2020년까지 연평균 8.2%의 성장률을 보일 전망이다. 또한 탈중앙화 어플들의 현황을 관리하는 Dappradar의 정보에 따르면 가장 인기 있는 3개 분류는 gambling, game, exchange다. 그 밖에도 Forbes, WSJ, NYTimes등의 외신에서도 블록체인으로 인해 게임산업에 다양한 변화가 일어날 것이라는 예측을 하고 있으며, 업계에서는 다양한 시도들이 진행되고 있다.

**2.2. 경쟁제품 분석**

Etheremon:

‘포켓몬스터’와 같이 메타마스크 계정을 이용해 몬스터를 잡고, 이를 이용해 배틀을 진행할 수 있는 게임. 배틀 모드는 ‘Gym’, ‘Practice’, ‘Ranked 총 3개의 배틀모드가 존재한다. Gym은 몬스터를 훈련시켜 더욱 강하게 만들수 있으며, Practice는 랭크 게임 대비를 위해 연습할 수 있는 모드다. Ranked는 말 그대로 점수가 기록되어 플레이어 순위결정을 위한 포인트가 지급되는 시스템이다. 탈 중앙 어플리케이션이며, 스마트 컨트랙트에 기록되어 있는 프로그램만 실행한다. 소스코드는 모두가 조회할 수 있으며, EMONT라는 게임 내 화폐를 이용해 몬스터를 사고 팔 수 있다. ERC20토큰이며, 화폐의 70%는 플레이어들이 게임을 하는 도중 ‘mine’을 할 수 있고, 25%는 User Growth Pool을 위해 활용되며, 5%는 개발팀이 활용하는 EMONT다.

Gods Unchained TCG:

‘하스스톤’과 같은 트레이딩 카드 게임, 즉 카드를 모아서 다른 유저들과 배틀 하는 게임이다. 먼저 메타마스크에 연결하여 카드팩을 구입한 후 구입 내역이 이더리움 블록체인에 저장된다. 가스를 사용해 거래가 완료되면 카드를 개봉할 수 있으며, 구입한 카드들은 스마트 컨트랙트에 기록되어 배틀에 활용할 수 있게 된다.

PlayTable:

앞서 언급한 게임들과 달리 PlayTable은 하드웨어 본체를 구입해서 사용하는 블록체인 콘솔이다. 보드판 역할을 하는 태블릿이 있으며, RFID 태그가 부착된 카드 및 말들이 주어진다. 게임에서 사용되는 카드 및 말과 게임상에서 이루어진 내용들이 태블릿을 통해 모두 블록체인에 기록된다. 현재까지는 보드게임 위주의 게임들을 출시하고 있으며, 콘솔의 소비자 판매가는 599$이다.

**2.3. 차별화 핵심전략**

한때 선풍적인 인기를 끌었던 ‘모두의 마블’이라는 게임을 블록체인과 결합하고 유저에게 맵을 직접 커스터마이징 할 수 있는 권한을 줌으로써 여타 게임들과의 차별화를 추구한다. 또한 스마트 컨트랙트 게임 내에 발생하는 모든 것을 기록하게끔 함으로써 게임과 블록체인의 완전한 결합이라고 볼 수 있다. 스마트컨트랙트 호출 모듈을 제공함으로써 직접적으로 게임을 호스팅 할 필요성이 없다. 정리하자면 다음과 같은 이점들이 있다:

1. 굉장히 많은 유저를 보유한 게임과 블록체인을 완전하게 결합한 플랫폼임.
2. 유저에게 맵을 직접 커스터마이징 할 수 있는 권한을 주는 오픈소스 기반의 플랫폼 서비스임.
3. ‘모두의 마블’ 외에도 다른 보드게임에 적용시킬 수 있는 미래 성장 가능성이 충분함.

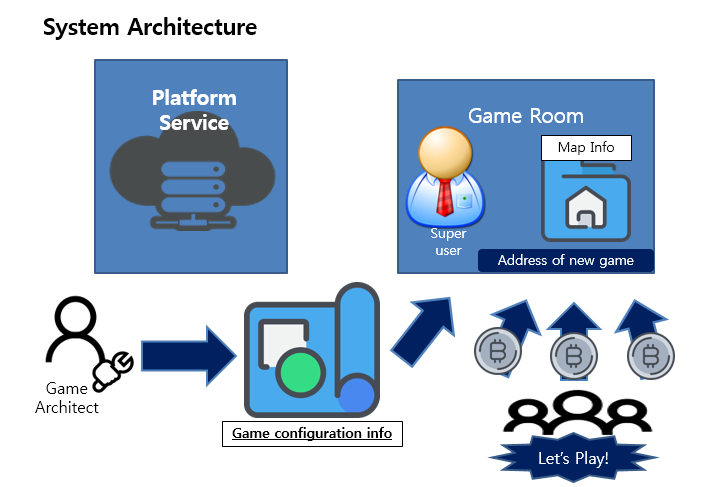
**3. 프로젝트 개발 결과**

3.1 주요 기능 / 주요 다이어그램

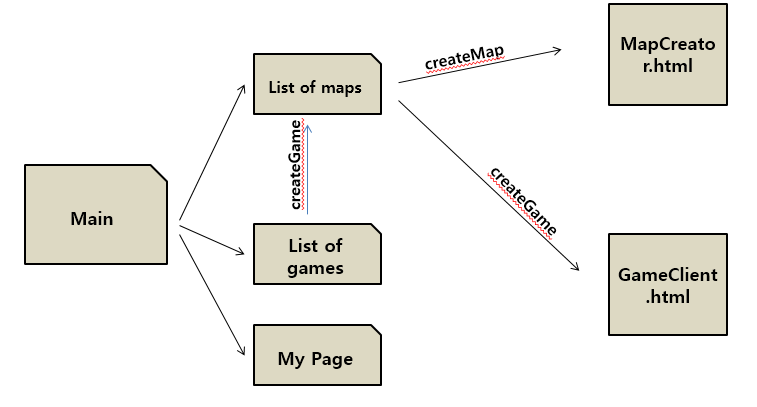
개발환경



시스템 아키텍쳐



서비스 순서도(workflow)



3.2 핵심 알고리즘

다음은 게임 맵 제작, 게임 방 생성 등과 관련된 contract 코드다.

핵심 함수

createGame : 게임 방을 만들 때 map id의 정보를 불러와서 새 게임을 연다.

createMap : 새로운 맵을 만든다. 맵에는 게임판의 정보, 게임판에서의 이동방향, 게임 진행시 제공되는 주사위의 범위(눈) 등 등을 작성하고 해당 게임을 새로운 계약으로 등록한다. (emitCreatorEvent 호출)

|  |  |
| --- | --- |
| |  | | --- | | **Creator\_v0.16.sol** |   pragma solidity ^0.4.24;  import "browser/Game.sol";  contract Creator{  중략…    function fetchGame(uint i) public view returns (address gameAddr){  return games[i];  }    uint logid = 0;  function emitCreatorEvent(address sender,uint eventtyp,uint val) internal {  emit CreatorEvent(sender,eventtyp + (logid<<40), val);  logid++;  }    function createGame(uint mapid, uint wait) public {  uint current\_id = games.length;    uint8[] memory info\_ = infos[mapid];    games.push(address(new Game(current\_id,msg.sender,info\_,walls[mapid],prices[mapid])));  emitCreatorEvent(msg.sender,5,uint(games[games.length-1]));  }    function fetchMap(uint id) public view returns (uint8[] info\_,uint[] walls\_,uint32[] prices\_){  if(id>=0 && id < infos.length){  info\_ = infos[id];  prices\_ = prices[id];  walls\_ = walls[id];  }else{  info\_ = new uint8[](1);  prices\_ = new uint32[](0);  walls\_ = new uint[](0);  }  }    function createMap(uint8[] info,uint[] walls\_, uint32[] prices\_) public{  uint current\_id = infos.length;    infos.push(info);  walls.push(walls\_);  prices.push(prices\_);    emitCreatorEvent(msg.sender,1,current\_id);  }    function saveMap(uint8 width, uint8 height, uint8 playermin, uint8 playermax, uint8 dicemin, uint8 dicemax, uint[] walls\_, uint32[] prices\_) public{    uint current\_id = infos.length;    infos.push([width,height,playermin,playermax,dicemin,dicemax]);  walls.push(walls\_);  prices.push(prices\_);    emitCreatorEvent(msg.sender,1,current\_id);  }      function creatBasicGame() public{  uint current\_id = games.length;    uint8[] memory info\_ = new uint8[](6);  info\_[0] = 10;  info\_[1] = 10;  info\_[2] = 1;  info\_[3] = 10;  info\_[4] = 1;  info\_[5] = 10;    uint[] memory walls\_ = new uint[](1);  walls\_[0] = 115739253086608799473272446246676835709932397168287476801964517168783086845952;    uint numlands = 0;  for(uint i = 0 ; i < walls\_.length ; i ++){  uint wall = walls\_[i];  for(uint j = 0; j<256 ; j++){  if(wall%2 == 1){  numlands++;  }  wall = wall/2;  }  }    uint32[] memory prices\_ = new uint32[](numlands);  uint seed = 13;  for(i = 0 ; i< numlands; i++){  prices\_[i] = uint32(100000+((seed\*\*(7+i))%89)\*1000);  }      games.push(address(new Game(current\_id,msg.sender,info\_,walls\_,prices\_)));  emitCreatorEvent(msg.sender,5,uint(games[games.length-1]));  }  } |

다음은 게임 내에서 호출하는 contract 코드이다.

핵심 함수

join\_game : 참여자(플레이어) 들이 들어올 때 호출된다.

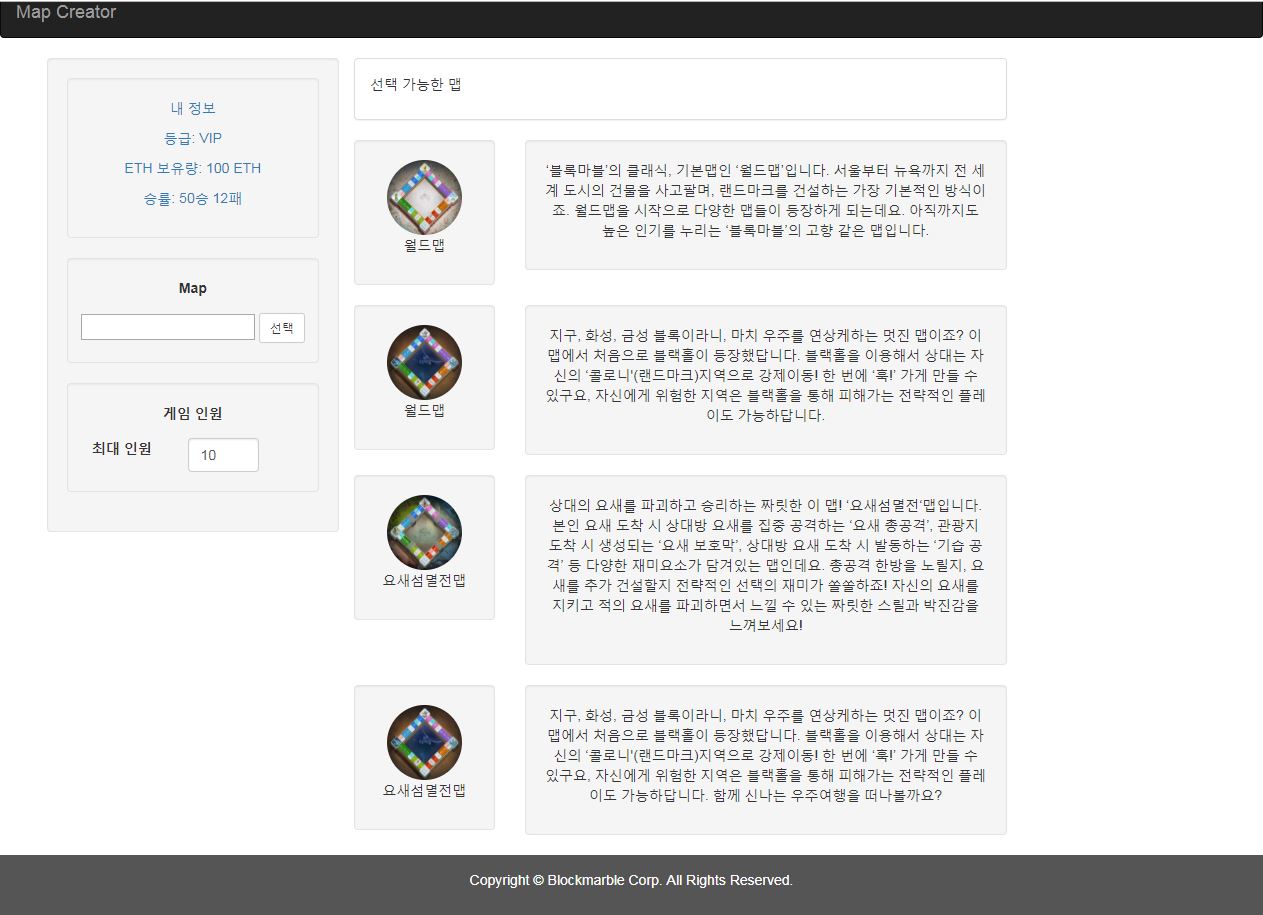
Gen\_dice : 주사위 눈을 만들어 모든 플레이어들에게 보낸다.

Action\_i : 게임이 진행되면서 특정 땅에 도착했을 때마다 행동하는 것을 액션이라 칭하여 실행되도록 하였다. 주요 액션으로는, 건물 구매, 땅 팔기, 땅 구매 참가 등이 있다.

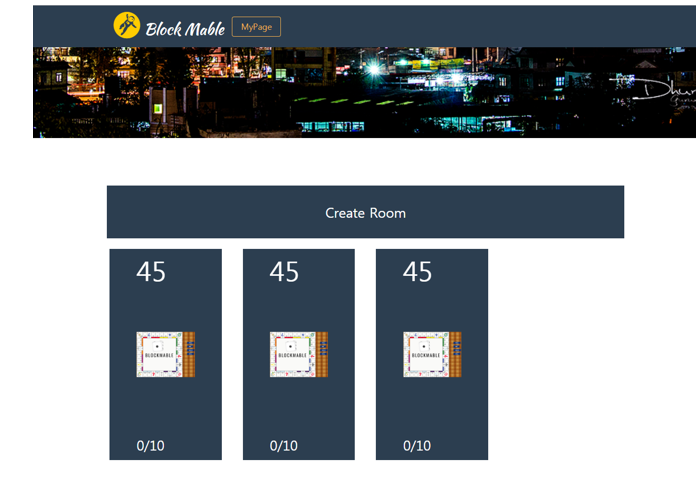
|  |
| --- |
| pragma solidity ^0.4.24;  // import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";  contract Game /\* is usingOraclize \*/ {  중략….  function join\_i(address sender) public{  if(players.length<playermax && cashes[sender] == 0){  cashes[sender] = 1000000;  players.push(sender);  emitGameEvent(sender,11);  }else if(cashes[sender] > 0){  emitGameEvent(sender,11);  }else{  emitGameEvent(sender,12);  }  }    function action(uint[] params) public{  action\_i(params,msg.sender);  }  function action\_i(uint[] params, address sender) public{  if(dices[sender] == 0){  emitGameEvent(sender,14);  }      uint i = 0;  while(i<params.length){  uint8 actionid = uint8(params[i]);  i++;    if(actionid>=1&&actionid<=4){  if(!ismovablelen(params[i])){  emitGameEvent(sender,178);    }    uint8 val = uint8(params[i]);    if(!move(sender,actionid,val)){  emitGameEvent(sender,1123);  }  }else if(actionid <= 7 && actionid >= 5){    if(dices[sender] != 0){  emitGameEvent(sender,1049);  return;  }    uint8[2] memory pos = poses[sender];  uint16 landid = posToLandId(pos);    if(actionid == 5){ // bid  if(landid != wall){  uint32 bid = uint32(params[i]);  if(bid > highest\_bid[landid]){  highest\_bidder[landid] = sender;  highest\_bid[landid] = bid;  }  }  }else if(actionid == 6){ // sell  landid = uint16(params[i]);  if(owners[landid] == sender){  cashes[sender] += sellvalue(landid);  owners[landid] = address(0);  }else{    return;  }  }else if(actionid == 7){ // build  landid = uint16(params[i]);  if(landid != wall && cashes[sender] > building\_price ){  built[landid] += 1;  cashes[sender] -= building\_price;  }  }  }else{  emitGameEvent(sender,125);  }  i++;  }    if(dices[sender] != 0){  emitGameEvent(sender,7462);  }    emitGameEvent(sender,1245);    numDone++;  if(numDone == players.length){  //sum\_moves();  }  }      function viewBoard() public view returns(address[] players\_,uint[] cashes\_, uint8[2][] poses\_, address[] owners\_ ,uint16[] buidings\_){  players\_ = players;  cashes\_ = new uint[](players.length);  poses\_ = new uint8[2][](players.length);    for(uint i = 0; i < players.length; i++){  cashes\_[i] = cashes[players[i]];  poses\_[i] = poses[players[i]];  }    owners\_ = owners;  buidings\_ = built;  }    function viewGame() public view returns(uint8,uint8,uint[],uint32[]){    uint8 \_n = N;  uint8 \_m = M;  uint[] memory \_boardRaw = boardRaw;  uint32[] memory \_prices = prices;    return (\_n, \_m, \_boardRaw, \_prices);  }    function gen\_dices() internal{  uint a = 37;  for(uint i = 0; i<players.length ; i++){  dices[players[i]] = uint8(a%(dicemax+1-dicemin)+dicemin);  a \*= 37;  }  emitGameEvent(address(0),7);  }      function sum\_moves() internal{  uint i = 0;  // default moves  for(i = 0; i<players.length ; i++){  if(dices[players[i]] > 0){  default\_action(players[i]);  }  }    // payments  for(i = 0; i< players.length ; i++){  address player = players[i];  uint16 landid = posToLandId(poses[player]);  if(owners[landid] != address(0) && owners[landid] != player){  uint32 fee = calcFee(landid);  if(cashes[player] >= fee||liquidate(player,fee-cashes[player])){  cashes[player] -= fee;  cashes[owners[landid]] += fee;  }else{  // broke  cashes[owners[landid]] += cashes[player];  cashes[player] = 0;  }  }  }      // decide bidding winners;  for(i = 0; i< prices.length ; i++){  if(owners[i] == address(0) && highest\_bidder[i] != address(0)){  if(cashes[highest\_bidder[i]]>highest\_bid[i]){  cashes[highest\_bidder[i]] -= highest\_bid[i];  owners[i] = highest\_bidder[i];  highest\_bidder[i] = address(0);  highest\_bid[i] = prices[i];  }  }  }    numDone = 0;  gen\_dices();  }        function liquidate(address addr,uint32 val) internal returns(bool){  uint32 cum = 0;  for(uint16 i = 0 ; i < prices.length ; i++){  if(owners[i] == addr){  uint32 sellval = sellvalue(i);  cum += sellval;  cashes[addr] += sellval;  owners[i] = address(0);  if(cum > val){  return true;  }  }  }  return false;  }      function move(address addr, uint8 moveid, uint8 val) internal returns (bool){  if(moveid == camefroms[addr]) return false;    uint8 dice = dices[addr];  uint8[2] memory pos = poses[addr];    if(dice < val) val = dice;    uint8 xVel = 0;  uint8 yVel = 0;    if(moveid == 1 || moveid ==3){  xVel = uint8(moveid-2);  }else{  yVel = uint8(moveid-3);  }    uint8 count = 0;    while(count<val){  count = count+1;  pos[0] += yVel;  pos[1] += xVel;    if(!inboard(pos)) return false;  if(posToLandId(pos)==wall) return false;  }    dice -= val;    dices[addr] = dice;  poses[addr] = pos;  camefroms[addr] = ((moveid-1)+2)%4 + 1;  return true;  }    function default\_action(address addr) internal{    while(dices[addr]>0){  for(uint8 moveid = 1; moveid<= 4; moveid++){  if( moveid==camefroms[addr] ) continue;    uint8 xVel = 0;  uint8 yVel = 0;    uint8[2] memory pos = poses[addr];    if(moveid == 1 || moveid ==3){  xVel = uint8(moveid-2);  }else{  yVel = uint8(moveid-3);  }    uint8 val = 0;    while(true){  pos[0] += yVel;  pos[1] += xVel;  if(!inboard(pos))break;  if(posToLandId(pos)==wall)break;  val++;  }    if(val>dices[addr])val = dices[addr];    if(val != 0){  if(move(addr,moveid,val)) break;  }  }  }  }  이하 생략.. |

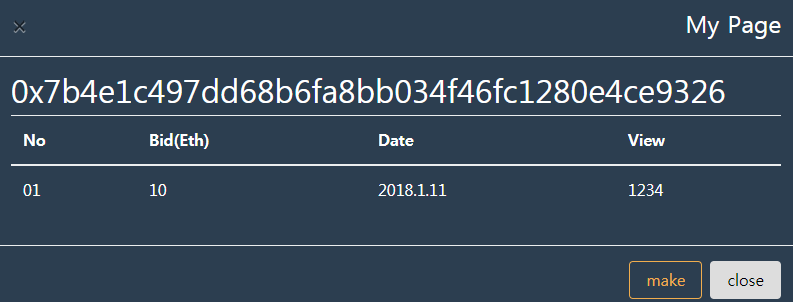
3.4 상세 동작

Createpage.html : 맵 제작자가 게임판을 만들 때 보는 페이지. 맵의 정보를 입력하여 자신이 원하는 새로운 맵과 기능을 만들 수 있다.

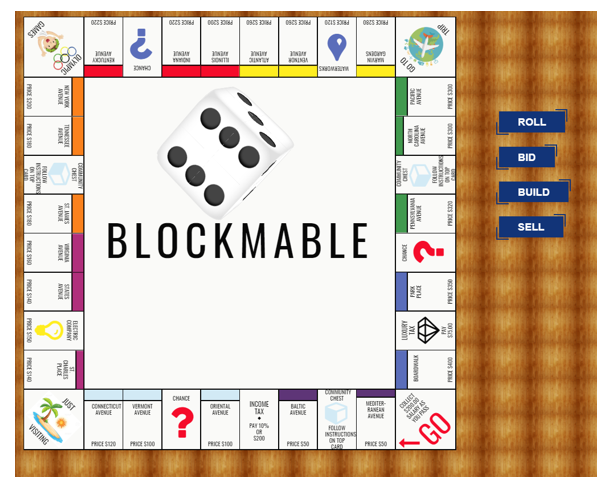


Index.html : 서비스 메인화면이다. 플레이어들은 방 목록에서 원하는 게임에 참가하거나, Crete Room으로 새로운 게임 방을 만들 수 있다. 또한 myPage에서 자신의 지갑주소와 남은 게임 머니등을 확인할 수 있다.





Map.html : 게임에 참가했을 때 플레이어는 해당 게임의 보드판을 볼 수 있고, 최소 게임 참가 인원이 모두 참가할 시 게임이 시작되며, 게임 진행 상황을 볼 수 있는 페이지다.



**4. 기대 효과 및 향후 개선사항**

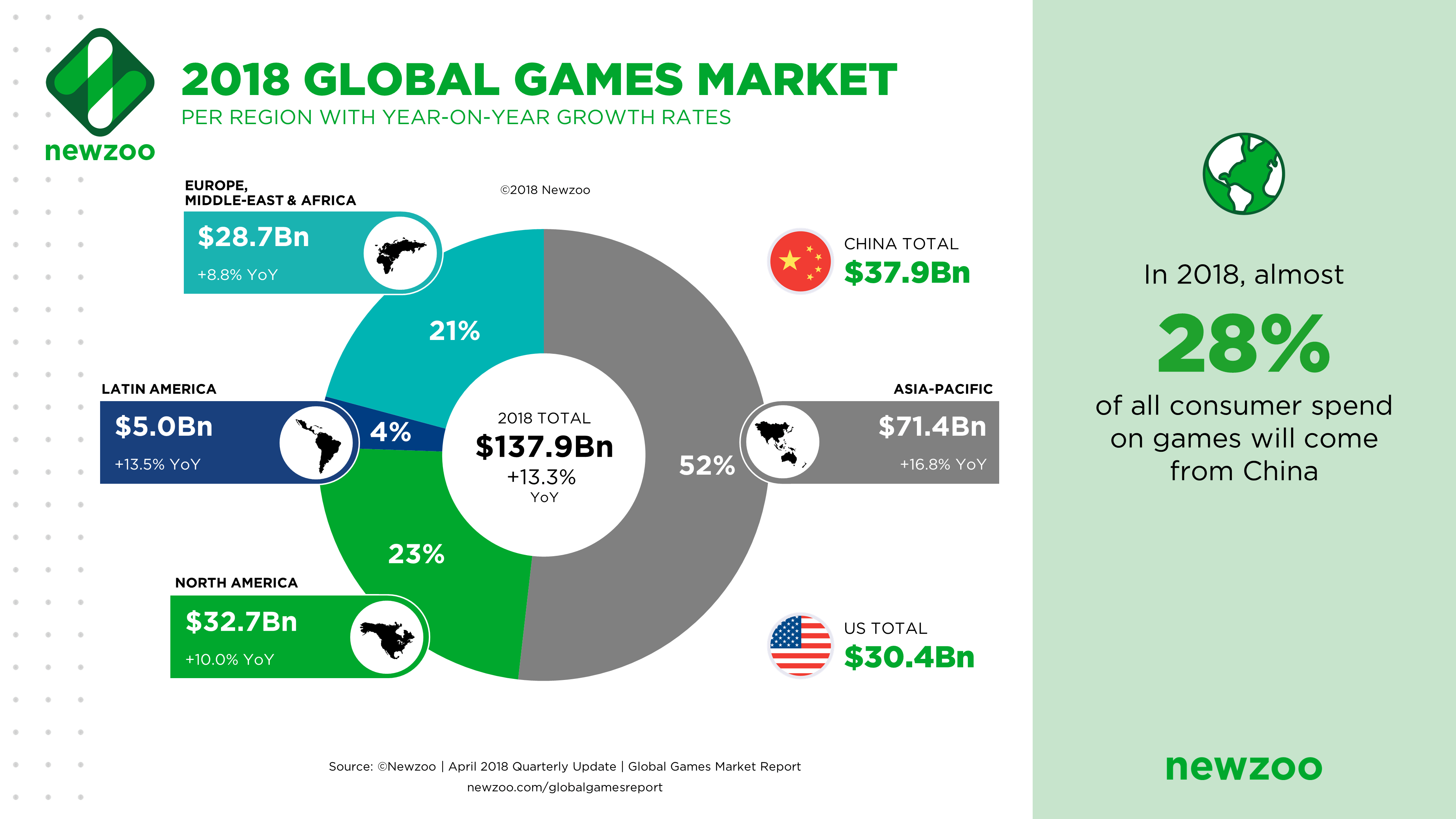
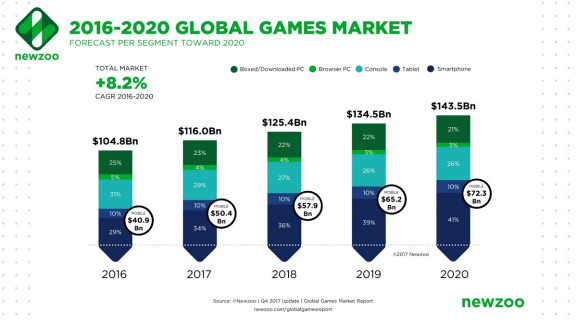
우선 첫 번째로 이더리움 Test net에서 Main net에 정식 배포가 가능하도록 더욱 완벽한 서비스를 구현해야 한다. 현재의 스마트 컨트랙트는 아직 알파테스트 단계 임으로 Ropsten Testnet에서 돌아가게끔 되어 있다.

조금 더 완벽한 ‘블록마블’ 플랫폼 서비스가 가능하게 된다면 다른 보드게임에도 도입을 시도할 수 있다. 현재의 TPS로 소화가 가능한 다른 게임에도 도입함으로써 블록체인과 게임을 결합한 플랫폼 모델을 더욱 보편화 시킬 수 있을 것이다.

하단의 좌측 그래프를 참조하면 글로벌 게임 시장은 현재도 그렇고 앞으로도 스마트폰의 비중이 약 40% 정도로 가장 클 것으로 전망된다. 그렇기 때문에 모바일 어플리케이션에서도 호환이 가능하도록 개발을 하는 것이 더욱 중요할 것이다.

또한 게임사와의 전략적 제휴를 통해 블록체인 및 스마트컨트랙트를 게임에 도입하는 프로세스를 더욱 촉진시킬 수 있다고 생각한다. 넷마블, 넥슨, 엔씨소프트와의 협력을 통해 더욱 안전한 게임 환경을 유저들에게 제공하는 것을 선도할 수 있을 것이다.

마지막으로 게임 산업의 성장 가능성이 높은 국가들을 상대로 스마트컨트랙트 플랫폼을 확장시켜 나가야 할 것이다. 특히 중국이 게임 산업 소비자 지출이 가장 많은 나라로 손꼽히고 있으며, 미국 역시 무시하지 못할 지출 규모를 가지고 있다. 우리나라에서 저변을 확보한 후 다른 나라에 블록체인 게임 플랫폼을 확산시키는데에 전념한다면 블록체인 게임산업을 선도할 수 있을 것이라고 사료된다.



**5. 개발 후기**

|  |
| --- |
| 팀 사진 첨부 |

|  |  |
| --- | --- |
| 성명 | 후기 |
| 정용수 | 실무 프로젝트에서 웹 파트를 맡아 프론트 단을 개발할 수 있는 좋은 경험을 하였습니다. 특히 부트스트랩으로 프론트 단을 개발하기 위해 W3School과 구글링을 하며 독학하였던 경험이 웹 개발 뿐 아니라 개발자로써의 역량을 향상시킬 수 있던 좋은 경험이었다고 생각합니다. 18년도 하반기 공채 원서접수 및 면접 기간에서 많이 바쁜 와중에도 프로젝트 마무리를 위해 힘써준 팀원들에게 감사합니다. |
| 함용재 | 블록체인이라는 신기술을 배워 실전에 응용할 수 있는 좋은 경험이 되었습니다. 기존에 배웠던 Web개발에서 스마트 컨트렉트의 기술로 확장시켜서 블록마블을 구현함으로써 제 역량에 큰 보탬이 되었습니다. 바쁘고 열악한 상황이었지만, 팀원들이 각자 제 역할을 책임감 있게 끝까지 맡아주어서 프로젝트가 성공적으로 잘 마무리 되었던 것 같고, 공채 일정이 많이 겹쳐서 전력을 다하지 못했던 점에서 한편으로는 미안한 마음도 큽니다. 그래도 끝까지 완성할 수 있어서 큰 보람이 있었던 프로젝트 였습니다. |
| 조상훈 | 이번 프로젝트에서는 In Game부분을 담당했습니다. 개발하면서 사용자를 위한 UI&UX를 구성하기 위해 생각하는 과정을 처음부터 끝까지 해본 경험이 적어서 진행 과정에서 오랜 시간이 걸렸던 것 같습니다. 그렇지만 수업을 통해 배웠던 내용들과 그 과정을 활용하면서 더 효율적이고 의미 있는 개발을 할 수 있었던 것 같습니다. 실제 현업에서도 이와 같은 일을 하게 된다면 보다 능동적으로 수행해 나갈 수 있을 것이라 생각합니다. 마지막으로 프로젝트 기획부터 구현을 완료하기까지 수많은 어려움이 있었지만 팀원들과 함께하여 극복할 수 있었던 것 같습니다. 모두 수고하셨습니다. |
| 윤기석 | 약 한달 간의 개발 기간동안 우여곡절이 많았습니다. 개발기간동안 10일에 달하는 추석 연휴가 겹쳐 쉬는 날에 코딩을 하는 일도 있었고 개인 사정상 프로젝트에 참여 못하는 팀원들도 있었습니다. 그래도 끝까지 책임감과 함께 하고자 하는 마음으로 최선을 다해 참가해준 팀원들 모두에게 고맙다는 생각이 먼저 듭니다. 개발을 진행하며 담당한 웹페이지 개발부분에서는 bootstrap을 사용해 원하는 모습을 기획하고 만들어가면서 이를 활용하는 역량을 키울 수 있었습니다. 앞으로 현업에서는 블록체인기반 서비스와 플랫폼이 더욱 더 생겨날 것이라 생각합니다. 그 때에 이번 프로젝트를 떠올리며 도움이 되고 참고할 수 있는 그런 의미로 남게 되었으면 좋겠습니다. |
| 양흥영 | 이번 프로젝트 통해 블록체인의 활용에 대한 부분을 고민해보고 적용해보는 좋은 경험을 할 수 있었습니다. 개인적인 취업 일정으로 인해 많은 부분 기여하지 못하여 아쉬움과 미안한 마음이 큽니다. 바쁜 일정이 있음에도 팀원들 모두 마지막까지 맡은바 역할을 잘 수행해주어서 다행히도 프로젝트를 잘 마무리 할 수 있었던 것 같습니다. |
| 허경환 | 멀티캠퍼스에서 진행한 ‘블록마블’ 프로젝트는 디앱 개발에 참여할 수 있는 유익한 경험이었습니다. 다른 팀원들에 비해 개발 역량이 부족했지만, 협력하는 과정에서 많이 배우고 더욱 성장할 수 있었습니다. 각자 맡은 부분을 수행하느라 바쁜 와중에도 많이 가르쳐주고 이끌어준 팀원들에게 감사의 표현을 하고 싶습니다. 빠듯한 일정이었지만 모두가 주어진 시간 내에 최선을 다한 만큼 성황리에 잘 마무리 지을 수 있었던 것 같습니다. |
| 이남호 | 이번 프로젝트는 진행과정에서 많은 어려움과 미숙함이 존재했습니다. 이러한 몇가지 점들을 잘 정리하여 추후 다른 프로젝트의 진행에도 도움이 되었으면 좋겠습니다.  1. 체인과는 별개로 동작하는 인터페이스를 먼저 개발하지 않았던 점.  - 이번 프로젝트의 개발에서 가장 어려운 파트는 스마트 컨트랙트의 개발에 있었습니다. 체인의 특성상 컴파일하고 체인에 올린 뒤에 테스트하는 것이 매우 더디기 때문입니다. 때문에 저희 조에서는 저를 체인개발 담당으로 지정하고 컨트랙트 개발에 처음부터 힘을 쏟았습니다. 하지만 이러한 방식은 여러가지 문제를 낳았습니다. 그중 가장 큰 문제는 클라이언트 단의 개발이 방향성 없이 더디게 진행 되었다는 것입니다. 서비스의 핵심에는 컨트랙트가 있었고 이 컨트랙트가 어떤 식으로 동작할 지 어떤 식으로 인터렉션 될지 예상이 안갔기 때문에 클라이언트 개발자들은 여러 가능성을 염두에 두고 더디게, 그리고 조심스럽게 개발에 임할 수 밖에 없었습니다.  2. 게임 로직에 대한 완전한 이해와 커뮤니케이션이 부족했던 점.  - 저희 조는 블록체인 위에서 안전하게 가상화페를 걸고 블루마블 (혹은 모노폴리) 게임을 즐길 수 있는 플랫폼을 만들고자 했습니다. 그러나 블루마블이란 게임은 그 룰의 종류나 로직이 정해진 것이 없고 다양하게 구현될 수 있는 게임이었습니다. 각종 찬스와 특별 칸들 그리고 매매방식 등등 세부적으로 합의를 보고 개발을 진행해야할 사항들이 매우 많았습니다. 더군다나 저희는 추가적인 재미와 차별화를 위해 플레이어 수를 거의 무제한으로 늘리기 위해 동시 턴제를 도입하고자 했고, 맵의 구조도 단순한 정사각형 외의 다양한 형태로 맵의 개발과 게임이 가능하게끔 하고자 했습니다. 그러나 이러한 과대하고 불명확한 목표는 개발 과정에서 많은 문제를 낳았습니다. 서로 개발중인 룰을 다르게 이해하고, 게임 진행 방식을 정확하게 이해하지 않은 상태에서 개발하다보니 계속해서 코드를 뒤엎고 다시 개발해야하는 문제가 생겼습니다.  결론적으로 여러 사람이 참여하는 프로젝트를 위해서는 기능들간의 연결점이 되는 인터페이스를 우선적으로 개발해야하며, 그 개발의 범위와 서비스의 로직에 대한 완전한 상호 이해가 기본이 되어야 한다는 어찌보면 당연한 사실을 다시금 깨닫는 경험이었습니다. |

**6. 강사 의견**

|  |  |  |
| --- | --- | --- |
| 평 가 요 소 | 배점 | 평 |
| 아이디어 :  유사한 서비스 존재 유무 및  체계성 | /20 |  |
| 2. 개발 : 실제 구현 정도 및 배포 유무,  코드의 무결성 및 난이도, 현업적용도, 실무기술 반영정도 | /30 |  |
| 3. PJT 수행력 : 일정관리 및 역할분담, 목표 일정 달성도, 팀내 참여도 등 | /30 |  |
| 4. 준비도 : 프리젠테이션 및 프로젝트 준비 정도 | /20 |  |
| 계 | /100 |  |