Team Mixbowl

7th Weekly Progress

Eunbi Kang, Taehoon Kim,
Byeoungkyu Park, Youji Cho, Eunseo Choe

Index

#Part 1

This Week

Part 1.

Timeline (This Week)

Milestones

April 2023

Today

Sun	Mon	Tue	Wed	Thu	Fri	Sat
9	10	11	12	13	14	15

회원가입 / 로그인 / 로그아웃 / 회원탈퇴
Frontend Backend
April 10, 2023 → April 12, 2023
은서 최은서 E Elise 강 강은비 태훈 Goathoon 병규 병규 박

회원등급관리
Frontend Backend
April 13, 2023 → April 15, 2023
은서 최은서 E Elise 강 강은비 태훈 Goathoon 병규 병규 박

회원 관련
Frontend Backend
April 10, 2023 → April 16, 2023

#Part 2

Frontend

Part 2.

Page Routing Setting

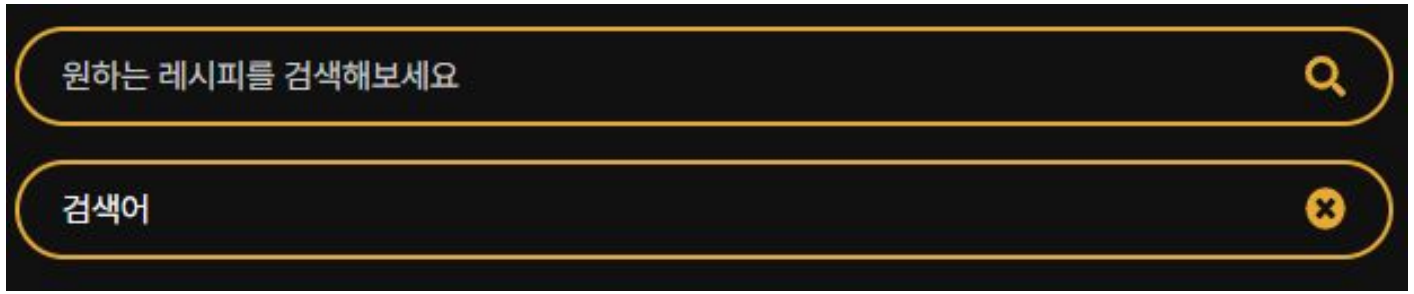
Routing Path	Page
/	Home page
/login	Login page
/register	Registration page
/recipe	Recipe page
/community	Community page
/cocktailbar	Cocktail Bar Map page
/mypage	My Page

Part 2. # Reusable Components(1)

Input Component

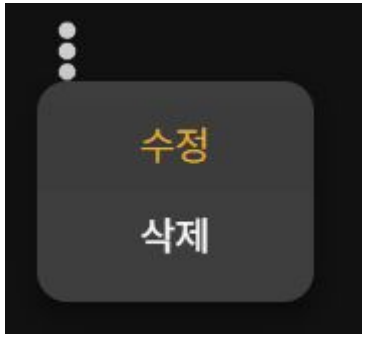
Part 2. # Reusable Components(2)

Search Bar Component



MemberBadge Component

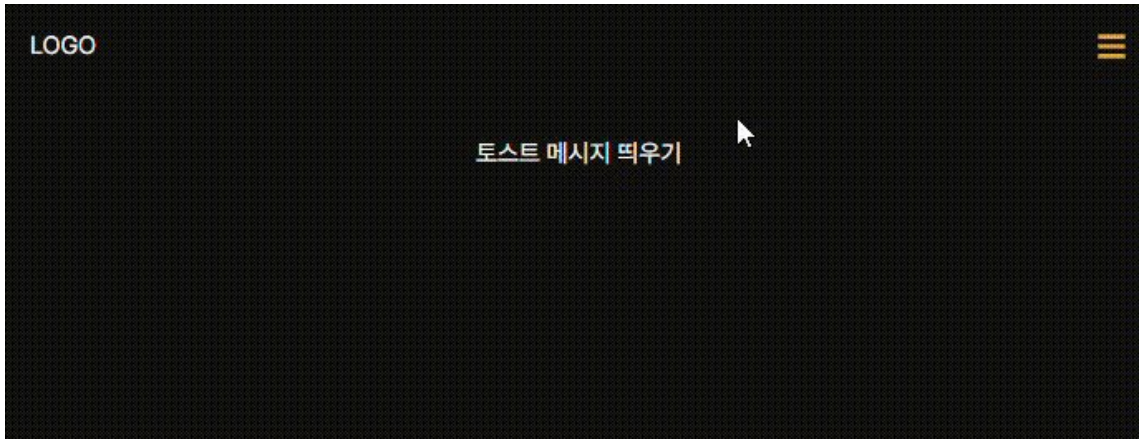
DropdownMenu Component



Part 2. # Reusable Components(3)

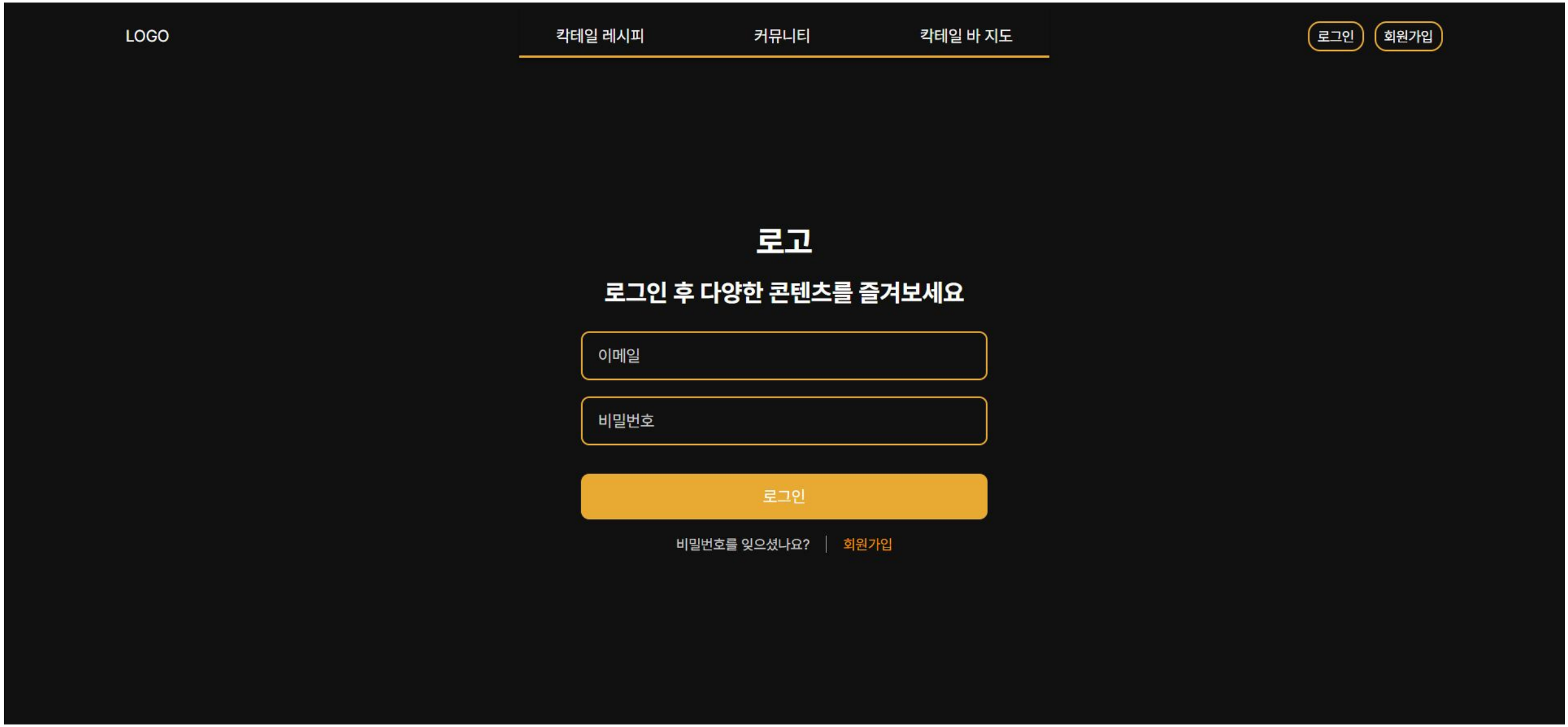
ToastMessage Component

Desktop Version

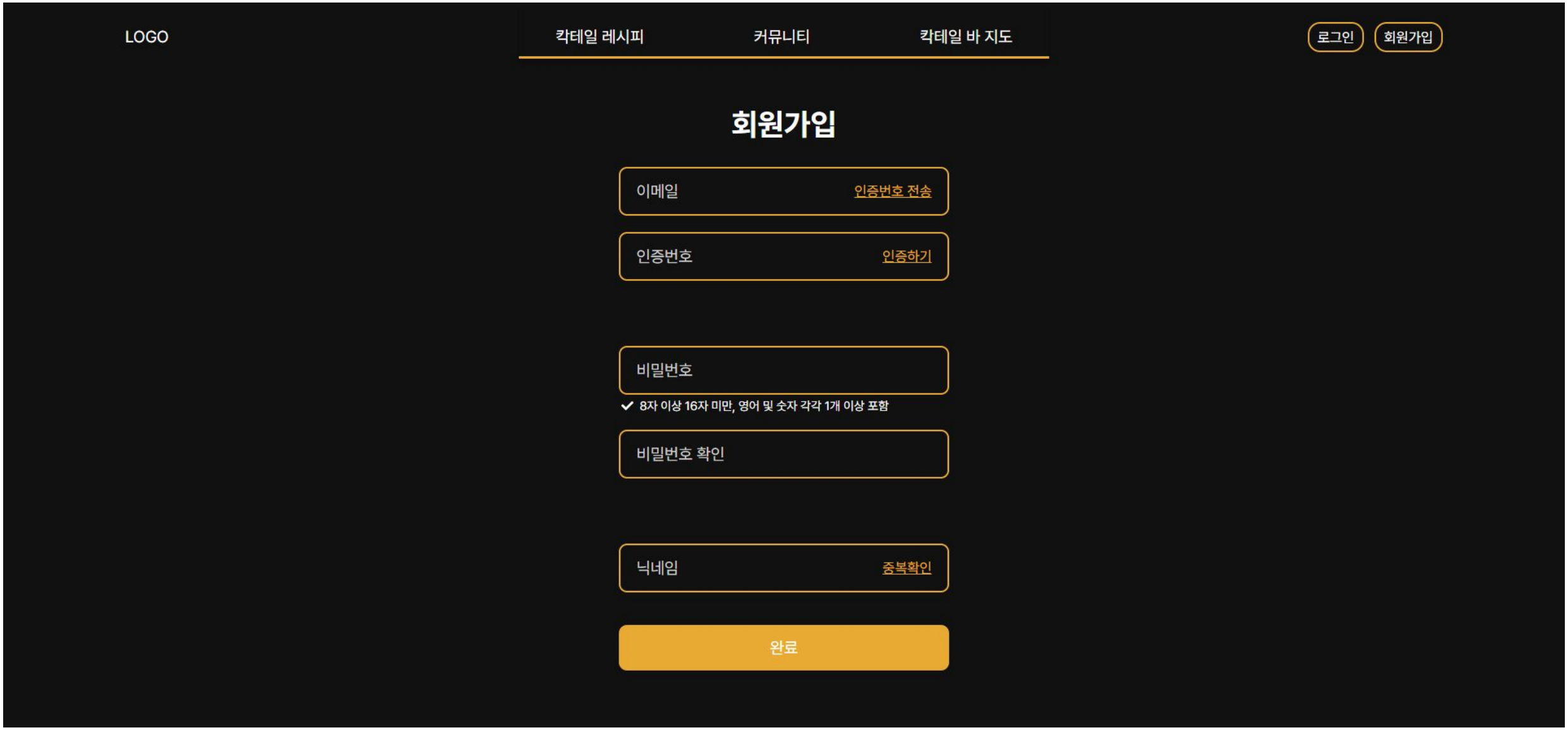


Mobile Version

Part 2. Login Page



Part 2. Registration Page



Part 2. # Proxy Setting

Frontend (Browser)

http://localhost:3000

Send HTTP Request

→

Backend (Server)

http://localhost:3030

ERROR

"Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at $somesite"

Part 2.

Proxy Setting

Frontend (Browser)

http://localhost:3000

→

Proxy Server

http://localhost:3000

→

Backend (Server)

http://localhost:3030

Part 2.

Token Management

Token	Expire Time	Storage Location
Access token	After 1 Hour	Local Storage
Refresh token	After 14 Days	Cookie

#Part 3

Backend / Data

Part 3. Setting - ORM(Objective Relational Mapping

'Sequelize'
&
'Sequelize-auto'
module

backend > src > models > USER.js > USER > init

```
import _sequelize from 'sequelize';
const { Model, Sequelize } = _sequelize;

export default class USER extends Model {
  static init(sequelize, DataTypes) {
  return super.init({
    UNO: {
      autoIncrement: true,
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true
    },
    NICKNAME: {
      type: DataTypes.STRING(45),
      allowNull: false
    },
    EMAIL: {
      type: DataTypes.STRING(45),
      allowNull: false
    },
    PASSWORD: {
      type: DataTypes.STRING(45),
      allowNull: false
    },
    LEVEL: {
      type: DataTypes.INTEGER,
      allowNull: false
    },
```

init-models.js > ...

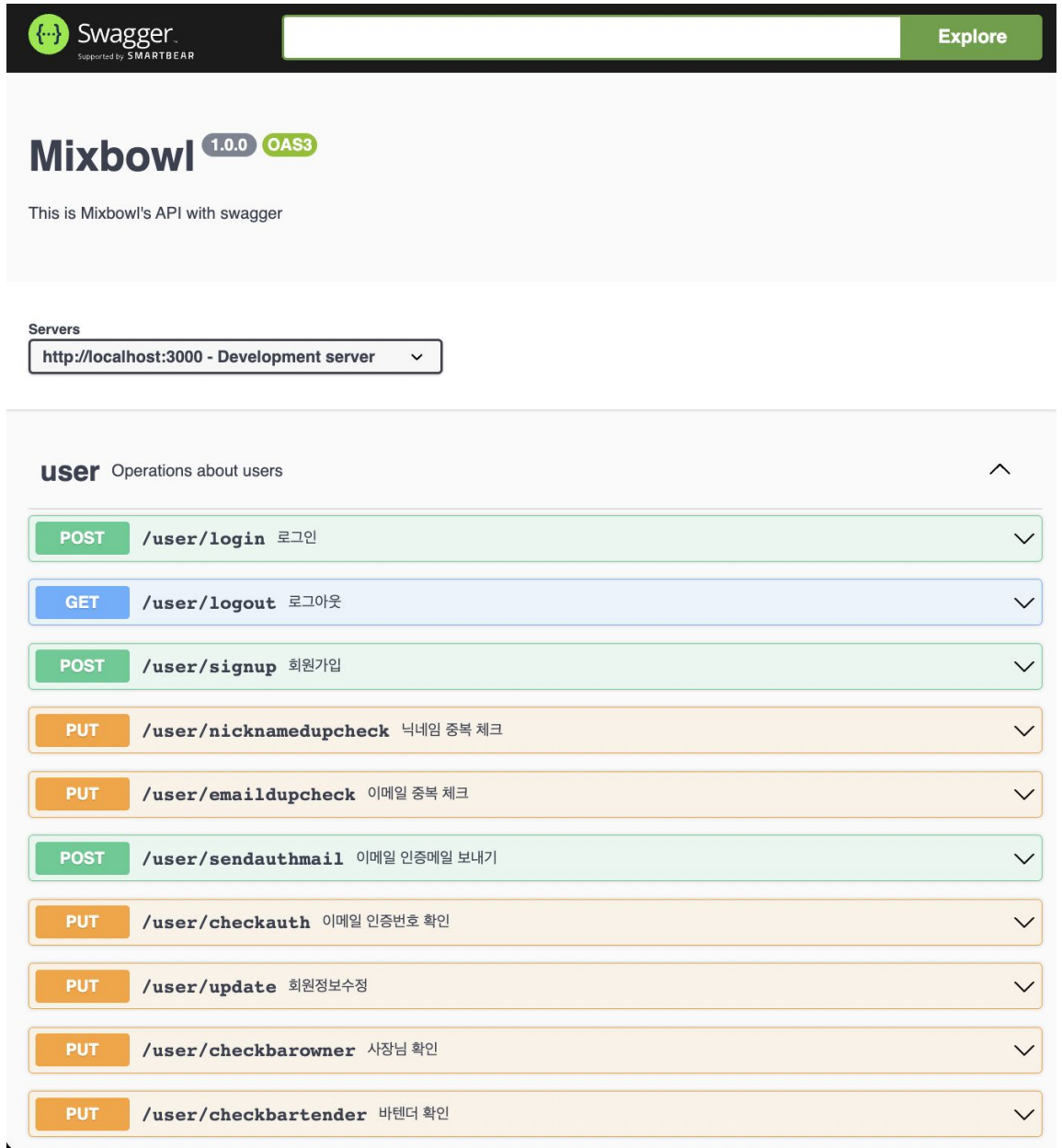
```
inction initModels(sequelize) {
POST.init(sequelize, DataTypes);
: = _POST_LIKE.init(sequelize, DataTypes);
. = _POST_REPL.init(sequelize, DataTypes);
_RECIPE.init(sequelize, DataTypes);
IKE = _RECIPE_LIKE.init(sequelize, DataTypes);
JSER.init(sequelize, DataTypes);

igsTo(POST, { as: "PNO_POST", foreignKey: "PNO"});
JST_LIKE, { as: "POST_LIKEs", foreignKey: "PNO"});
igsTo(POST, { as: "PNO_POST", foreignKey: "PNO"});
JST_REPL, { as: "POST_REPLs", foreignKey: "PNO"});
RECIPE, { as: "RNO_RECIPE", foreignKey: "RNO"});
POST, { as: "POSTs", foreignKey: "RNO"});
ongsTo(RECIPE, { as: "RNO_RECIPE", foreignKey: "RNO"});
RECIPE_LIKE, { as: "RECIPE_LIKEs", foreignKey: "RNO"});
USER, { as: "UNO_USER", foreignKey: "UNO"});
JST, { as: "POSTs", foreignKey: "UNO"});
igsTo(USER, { as: "UNO_USER", foreignKey: "UNO"});
JST_LIKE, { as: "POST_LIKEs", foreignKey: "UNO"});
igsTo(USER, { as: "UNO_USER", foreignKey: "UNO"});
JST_REPL, { as: "POST_REPLs", foreignKey: "UNO"});
To(USER, { as: "UNO_USER", foreignKey: "UNO"});
ECIPE, { as: "RECIPEs", foreignKey: "UNO"});
ongsTo(USER, { as: "UNO_USER", foreignKey: "UNO"});
ECIPE_LIKE, { as: "RECIPE_LIKEs", foreignKey: "UNO"});
```

```
const user = await USER.findOne({ where: { UNO: req.decoded.uno } });
```

Part 3. # API Documentation

‘Swagger’ module

http://localhost:3030/api-docs/



Part 3. API Documentation

POST /user/signup 회원가입

Parameters

Try it out

No parameters

Request body required

Example Value | Schema

```
{
  "email": "john@email.com",
  "nickname": "mynickname",
  "password": "12345"
}
```

Responses

Code	Description	Links
200	Media type: application/json Controls Accept header. Example Value | Schema `{ "success": true, "message": "Operation success" }`	No links
400	Media type: application/json Example Value | Schema `{ "success": false, "message": "Operation Fail", "error": {} }`	No links

'yaml' file

backend > src > swagger > user.yaml > ...

```yaml
# user api for swagger

paths:
  /user/login:
    post:
      tags:
        - user
      summary: 로그인
      description: ''
      operationId: loginUser
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/User'
      responses:
        '200':
          description: ''
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Success'
        '400':
          description: ''
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Fail'
```

Part 3. **Implement**

user Operations about users

POST	/user/login	로그인
GET	/user/logout	로그아웃
POST	/user/signup	회원가입
PUT	/user/nicknamedupcheck	닉네임 중복 체크
PUT	/user/emaildupcheck	이메일 중복 체크
POST	/user/sendauthmail	이메일 인증메일 보내기
PUT	/user/checkauth	이메일 인증번호 확인
PUT	/user/update	회원정보수정
PUT	/user/checkbarowner	사장님 확인
PUT	/user/checkbartender	바텐더 확인

Part 3. # Implement - SQL module

```
export default sql;
```

- **getToken**
  To get Refresh Token on USER's DB
- **namedupcheck**
  To check USER's nickname duplication
- **emaildupcheck**
  To check USER's email duplication
- **signupUser**
  To sign up on DB
- **loginUser**
  To login by checking USER's table

```
loginUser: async (req) => {
  const { email, password } = req.body;
  try {
    // const [username] = await promisePool.query(`
    // SELECT NICKNAME FROM Mixbowl.USER WHERE '${email}' = EMAIL AND '${password}' = PASSWORD ;
    // `);
    const {dataValues} = await USER.findOne({ where: { email : `${email}`,password:`${password}` } });
    const username = dataValues["NICKNAME"];
    if (username.length === 0) {
      console.log("hi");
      throw new Error("Invalid Info User");
    }
    //UNO 도 같이 포함
    const accessToken = await jwt_module.sign(username[0]["NICKNAME"]);
    const refreshToken = await jwt_module.refresh();

    //refresh token sql 업데이트
    await promisePool.query(`
      UPDATE USER SET TOKEN = '${refreshToken}' WHERE NICKNAME = '${username}';
    `);
    return {
      code: 200,
      message: "토큰이 발급되었습니다.",
      token: {
        accessToken,
        refreshToken,
      },
    };
  } catch (error) {
    console.log(error.message);
    return {
      code: 401,
    };
```

Part 3. Implement - JWT module

```javascript
export function sign(username) {
  // Access 토큰 생성 코드
  const payload = {
    type: "JWT",
    nickname: username,
  };

  return jwt.sign(payload, process.env.SECRET_KEY, {
    expiresIn: "1h",
    issuer: "MixBowl",
  });
}

export function accessVerify(token) {
  //Access 토큰 확인 코드
  let decoded = null;
  try {
    decoded = jwt.verify(token, process.env.SECRET_KEY);
    return {
      ok: true,
      nickname: decoded.nickname[0]["NICKNAME"],
    };
  } catch (error) {
    return {
      ok: false,
      message: error.message,
    };
  }
}
```

```javascript
export function refresh() {
  // Refresh 토큰 생성 코드
  return jwt.sign({}, process.env.SECRET_KEY, {
    expiresIn: "14d",
    issuer: "MixBowl",
  });
}

//토큰 header에 주고, db 내 refresh 토큰으로 확인
export async function refreshVerify(token, username) {
  //Refresh 토큰 확인 코드
  //redis 도입하면 좋을듯
  try {
    const refToken = await sql.getToken(username);
    if (token === refToken) {
      try {
        jwt.verify(token, process.env.SECRET_KEY);
        return true;
      } catch (error) {
        return false;
      }
    } else {
      return false;
    }
  } catch (error) {
    return false;
  }
}
```

Part 3.

Implement - JWT module

```
export const refresh_new = async (req, res) => {
  if (req.headers.authorization && req.headers.refresh) {
    const access = req.headers.authorization;
    const refresh = req.headers.refresh;

    const accessResult = accessVerify(access);
    const decodeAccess = jwt.decode(access);

    if (decodeAccess === null) {
      res.status(401).send({
        ok: false,
        message: "No Authorization for Access Token",
      });
    }

    const refreshResult = refreshVerify(refresh, decodeAccess.nickname);

    if (accessResult.ok === false && accessResult.message === "jwt expired") {
      if (refreshResult.ok === false) {
        res.status(401).send({
          ok: false,
          message: "No Authorization, MAKE A NEW LOGIN",
        });
      } else {
        //refresh tokenOl 유효하므로, 새로운 access token 발급
        const newAccessToken = sign(req.body.nickname);

        res.stauts(200).send({
          ok: true,
          nickname: req.body.nickname,
        });
      }
```

Part 3. Implement - Login

```
const {dataValues} = await USER.findOne({ where: { email : `${email}`,password:`${password}` } });
const username = dataValues["NICKNAME"];
if (username.length === 0) {
  console.log("hi");
  throw new Error("Invalid Info User");
}
//UNO 도 같이 포함
const accessToken = await jwt_module.sign(username[0]["NICKNAME"]);
const refreshToken = await jwt_module.refresh();
```

return Token

CHECK USER's INFORMATION

```
// 로그인
router.post("/login", async (req, res) => {
  try {
    const tokens = await sql.loginUser(req, res);
    if (tokens.code !== 200) {
      throw new Error();
    }
    const { email } = req.body;
    //이메일 유효성 검사 함수 정의 필요
    if (email.length === 0) {
      throw new Error();
    }
    return res.status(200).send({
      success: true,
      // nickname: nickname[0]["NICKNAME"],
      tokens,
    });
  } catch (error) {
    return res.send({ success: false });
  }
});
```

#Part 4

NEXT WEEK

Part 4. # Timeline (Next Week)

칵테일 바 지도 페이지 구현 및 API 연결
강 강은비
2023년 4월 17일 → 2023년 4월 30일
Not started
Frontend

칵테일바 지도 기능 구현
2023년 4월 17일 → 2023년 4월 30일
Not started
Frontend Backend

칵테일 바 리뷰 데이터 ...
은서 최은서 태훈 Goathoon
2023년 4월 17일
Not started
Backend

지도 및 리뷰 관련 REST API 작성
은서 최은서 태훈 Goathoon
2023년 4월 18일 → 2023년 4월 19일
Not started
Backend

칵테일 바 리뷰 작성 기능 구현
은서 최은서 태훈 Goathoon
2023년 4월 20일 → 2023년 4월 23일
Not started
Backend

칵테일바 지도 관련 데...
태훈 Goathoon 은서 최은서
2023년 4월 17일
Not started
Backend

Kakao Map API 숙지
은서 최은서 태훈 Goathoon
2023년 4월 18일 → 2023년 4월 19일
Not started
Backend

Thank you