Web Studio 2019
13.Authentication

김상근 / sisobus1@gmail.com

Contents

Authentication

인증이 왜 필요한가?

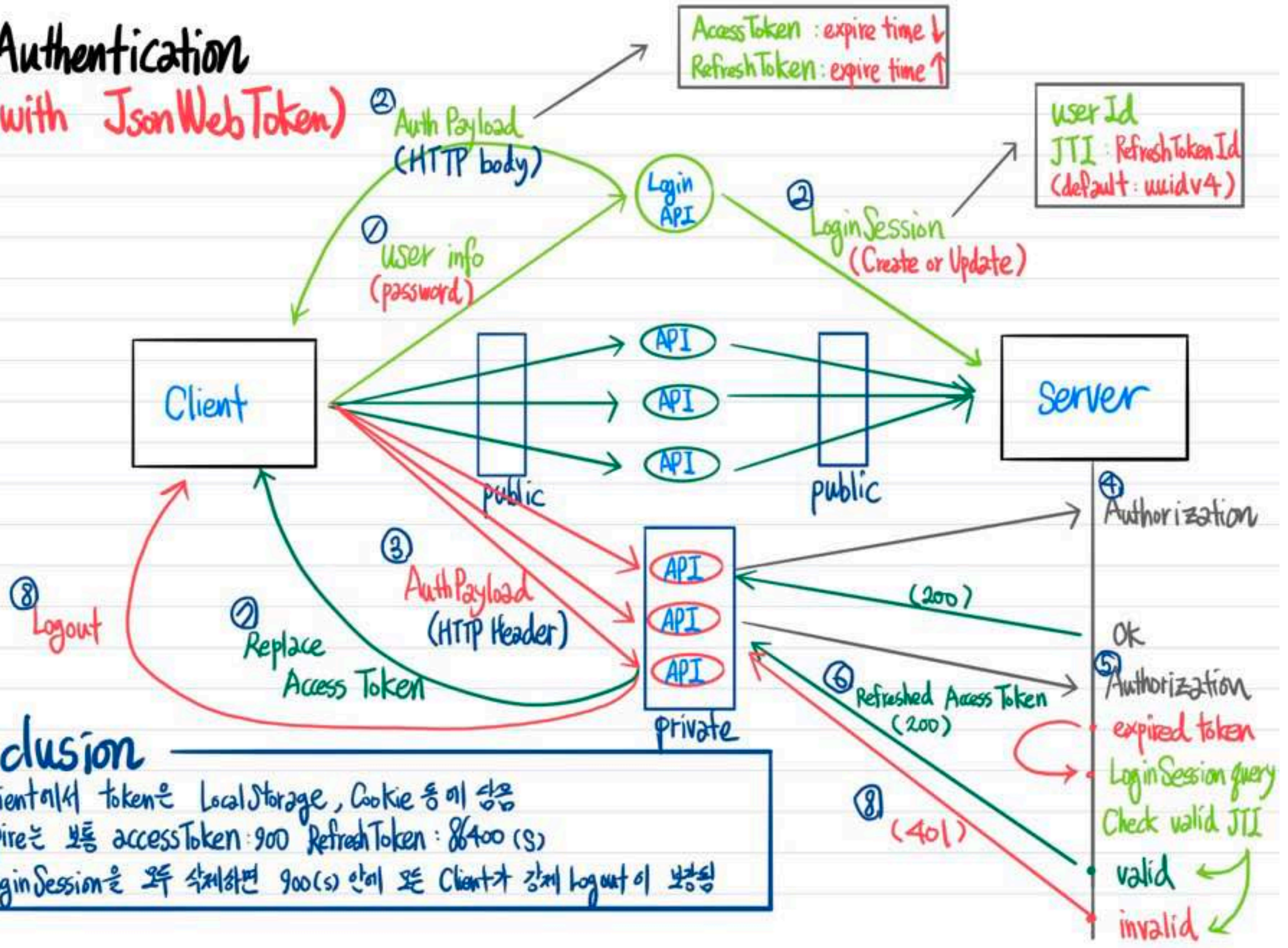
1. 로그인한 유저에게만 정보를 제공하고 싶을 때
2. 건당 과금 서비스를 만들 때
3. Private API 서비스를 구현해야 할 때
4. 유저의 Level이 존재할 때 (e.g. 관리자, 작성자, …)
5. 누가 뭘 했는지 남기고 싶어서

Authentication

인증 구현하기 쉬울 것 같은데?

1. 로그인하면 서버에서는 어떤 key같은 것을 돌려줘서
2. 클라이언트에서 이 key와 함께 API를 호출하도록 하면
3. 인증 완료!
4. 만약 이 key가 다른사람에 노출되었다면? 〉〉 key의 유효기간을 정해주자
5. key가 만료 될 때마다 로그인을 새로해야 한다면?
6. 모든 key를 폐기하고 싶다면? (강제 로그아웃)
7. 한 아이디로 돌려쓰는 것을 막고싶다면?

Authentication (with JsonWebToken)



Conclusion

* Client에서 token은 LocalStorage, Cookie 등에 심음
* expire는 보통 accessToken : 900 RefreshToken : 86400 (s)
* LoginSession을 모두 삭제하면 900(s) 안에 모든 Client가 강제 logout이 보장됨

Authentication

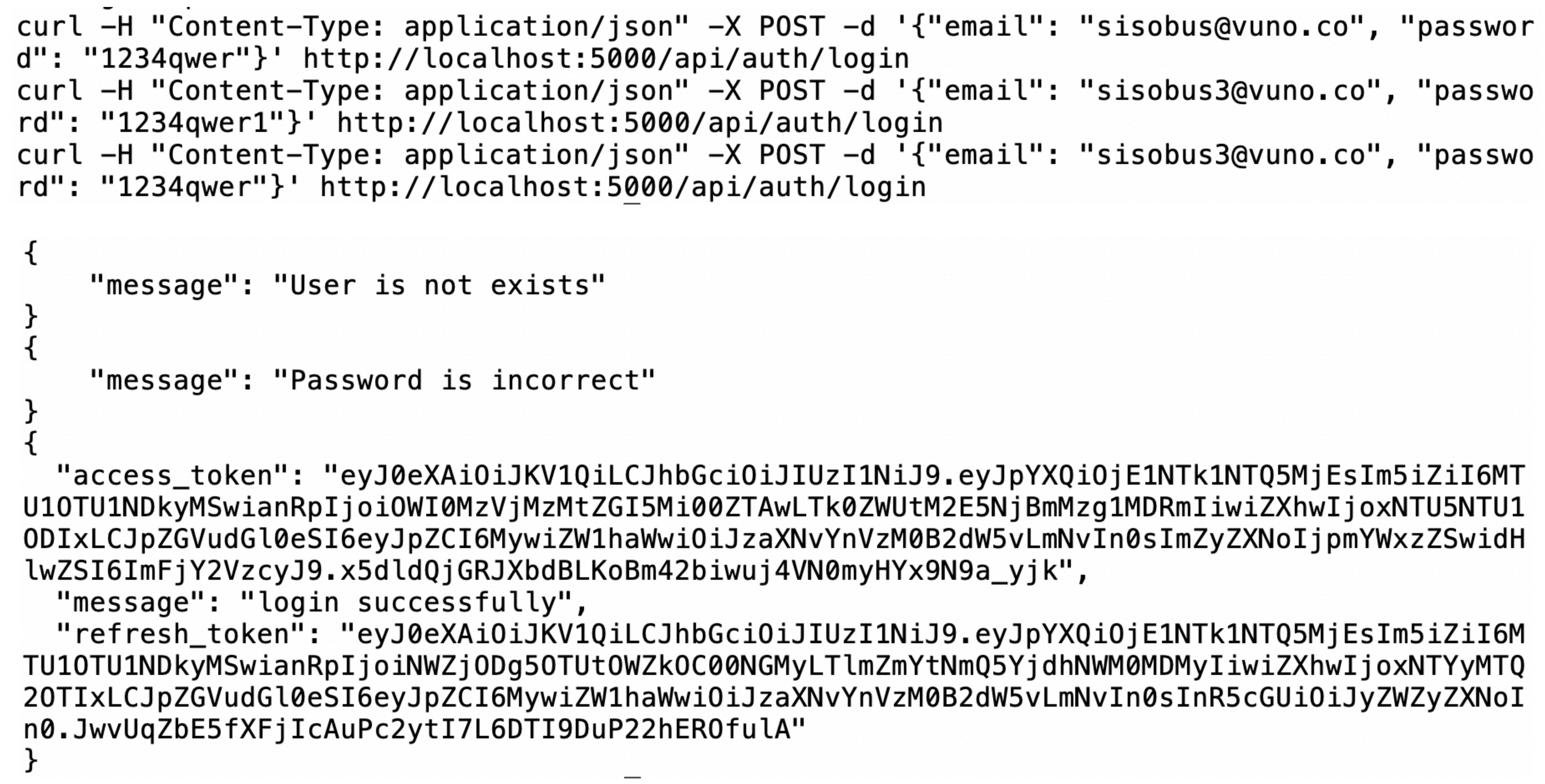
인증과정

1. Login 하면 accessToken, refreshToken을 발급해줌
   (accessToken은 유효기간을 짧게, refreshToken은 길게; default: 900s, 86400s)
2. Client에 tokens를 저장해두고, fetch할 때마다 request header에 담아서 보냄
   ({'Authorization': 'Bearer 〈accessToken〉', 'refreshToken': '〈refreshToken〉'}
3. Server에선 private api의 경우 request header를 보고 인증처리를 함
   1. accessToken이 만료되지 않았으면 인증처리
   2. 만약 accessToken이 만료되었으면 refreshToken을 보고, refreshToken도 만료되었으면 비인증 처리 (401 error return)
      1. refreshToken은 만료되지 않았으면 accessToken을 새로 발급해서 response header에 담아줌
4. Client에선 만약 response header에 새로운 token이 있다면 갱신해줌

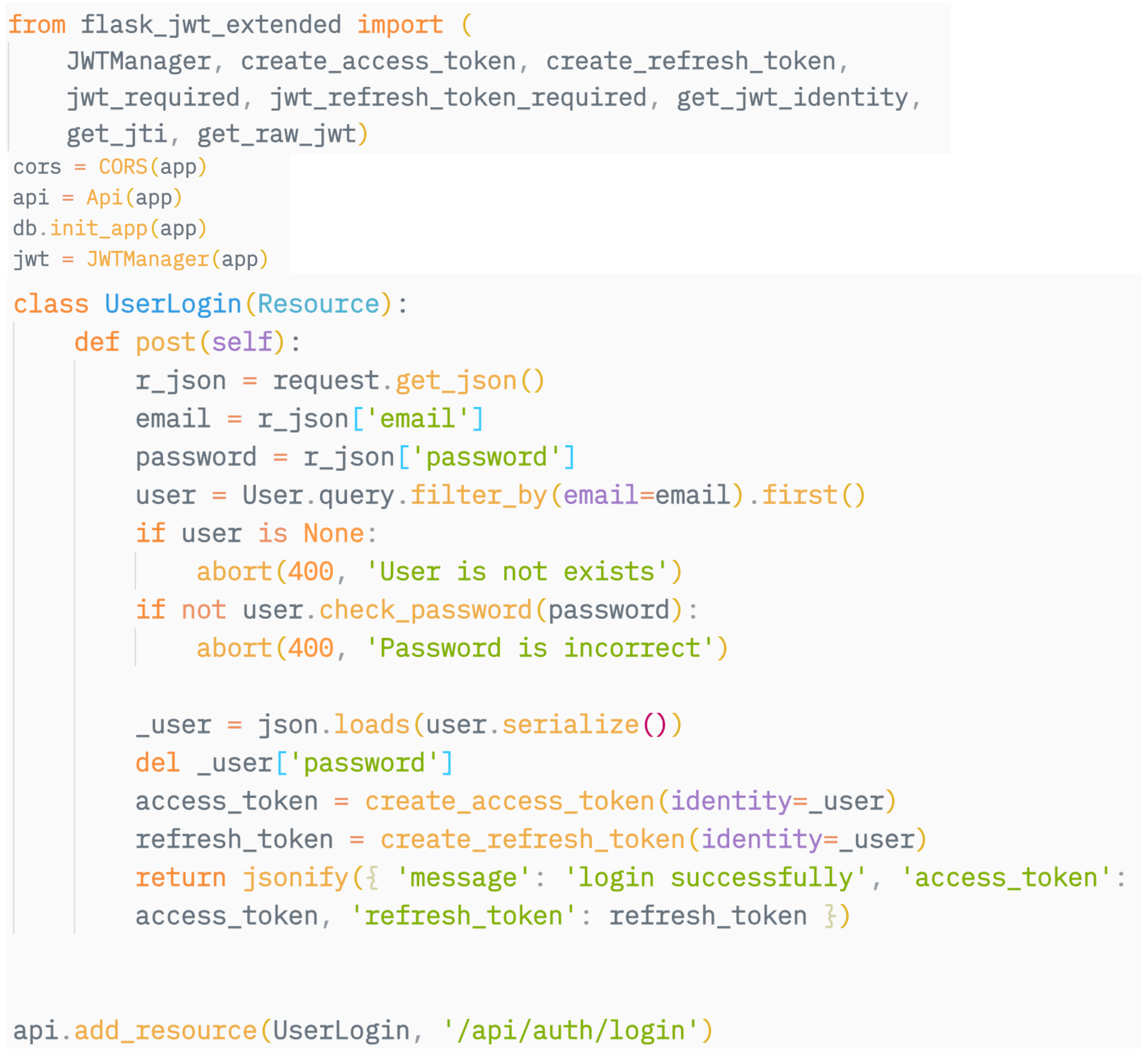
Authentication

API 서버측 구현

1. email, password를 받음
2. db에서 user를 email로 검색
3. 존재한다면 tokens 발급
4. Tokens 반환



```
curl -H "Content-Type: application/json" -X POST -d '{"email": "sisobus@vuno.co", "passwor
d": "1234qwer"}' http://localhost:5000/api/auth/login
curl -H "Content-Type: application/json" -X POST -d '{"email": "sisobus3@vuno.co", "passwo
rd": "1234qwer1"}' http://localhost:5000/api/auth/login
curl -H "Content-Type: application/json" -X POST -d '{"email": "sisobus3@vuno.co", "passwo
rd": "1234qwer"}' http://localhost:5000/api/auth/login

{
    "message": "User is not exists"
}
{
    "message": "Password is incorrect"
}
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1NTk1NTQ5MjEsIm5iZiI6MT
U1OTU1NDkyMSwianRpIjoiOWI0MzVjMzMtZGI5Mi00ZTAwLTk0ZWUtM2E5NjBmMzg1MDRmIiwiZXhwIjoxNTU5NTU1
ODIxLCJpZGVudGl0eSI6eyJpZCI6MywiZW1haWwiOiJzaXNvYnVzM0B2dW5vLmNvIn0sImZyZXNoIjpmYWxzZSwidH
lwZSI6ImFjY2VzcyJ9.x5dldQjGRJXbdBLKoBm42biwuj4VN0myHYx9N9a_yjk",
  "message": "login successfully",
  "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1NTk1NTQ5MjEsIm5iZiI6M
TU1OTU1NDkyMSwianRpIjoiNWZjODg5OTUtOWZkOC00NGMyLTlmZmYtNmQ5YjdhNWM0MDMyIiwiZXhwIjoxNTYyMTQ
2OTIxLCJpZGVudGl0eSI6eyJpZCI6MywiZW1haWwiOiJzaXNvYnVzM0B2dW5vLmNvIn0sInR5cGUiOiJyZWZyZXNoI
n0.JwvUqZbE5fXFjIcAuPc2ytI7L6DTI9DuP22hEROfulA"
}
```



```python
from flask_jwt_extended import (
    JWTManager, create_access_token, create_refresh_token,
    jwt_required, jwt_refresh_token_required, get_jwt_identity,
    get_jti, get_raw_jwt)
cors = CORS(app)
api = Api(app)
db.init_app(app)
jwt = JWTManager(app)
class UserLogin(Resource):
    def post(self):
        r_json = request.get_json()
        email = r_json['email']
        password = r_json['password']
        user = User.query.filter_by(email=email).first()
        if user is None:
            abort(400, 'User is not exists')
        if not user.check_password(password):
            abort(400, 'Password is incorrect')

        _user = json.loads(user.serialize())
        del _user['password']
        access_token = create_access_token(identity=_user)
        refresh_token = create_refresh_token(identity=_user)
        return jsonify({ 'message': 'login successfully', 'access_token':
        access_token, 'refresh_token': refresh_token })


api.add_resource(UserLogin, '/api/auth/login')
```

Authentication

API 서버측 구현 (+a)

1. email, password를 받음
2. db에서 user를 email로 검색
3. 존재한다면 tokens 발급
4. 이미 Login상태라면 jwt id를 갱신
5. jwt id를 LoginSession에 저장
6. Tokens 반환

```
{
  "data": {
    "email": "sisobus3@vuno.co",
    "id": 3,
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1NTk1NTUwNzMsIm5iZiI6MTU1O
TU1NTA3MywianRpIjoiMTUyYzQyZjItNDYwZi00OWIxLTkzYzItM2VhNGQ3MWE5YzUyIiwiZXhwIjoxNTYyMTQ3MDc
zLCJpZGVudGl0eSI6eyJpZCI6MywiZW1haWwiOiJzaXNvYnVzM0B2dW5vLmNvIn0sInR5cGUiOiJyZWZyZXNoIn0.O
XdQLZm2Sa03PlMr5j0H0AduJT5N1erFM_WE3t6eLbU",
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1NTk1NTUwNzMsIm5iZiI6MTU1OTU
1NTA3MywianRpIjoiZjVlYmExMTMtNmM3NC00NGQyLThmZjktN2JhZGY5ZWY2MGJjIiwiZXhwIjoxNTU5NTU1OTczL
CJpZGVudGl0eSI6eyJpZCI6MywiZW1haWwiOiJzaXNvYnVzM0B2dW5vLmNvIn0sImZyZXNoIjpmYWxzZSwidHlwZSI
6ImFjY2VzcyJ9._EEcGmS7QcKoQFNYHIWuzXDiEZXw2G7t4H-IjjH002o"
  },
  "message": "login successfully"
}
```

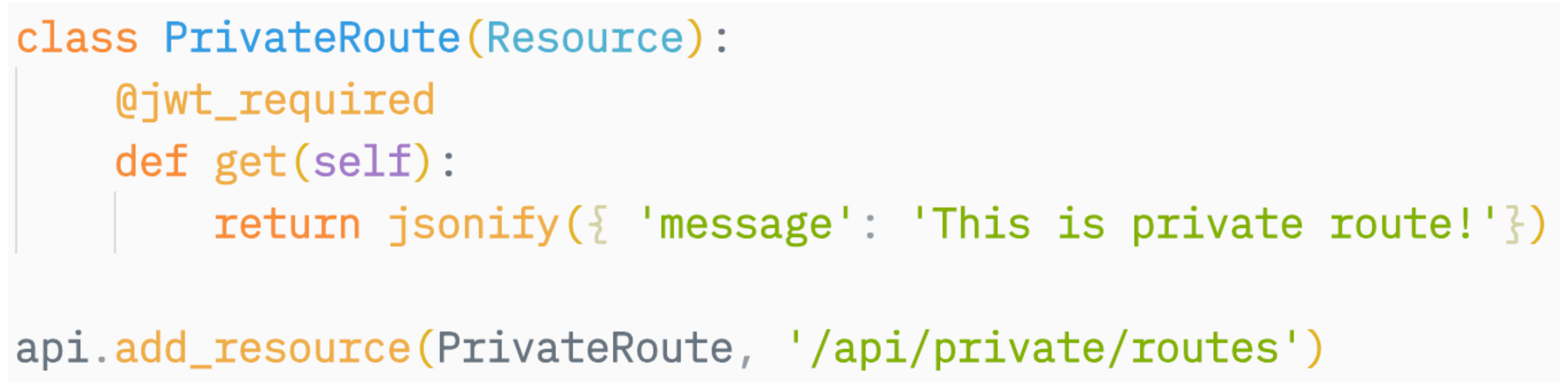
```python
class LoginSession(db.Model):
    __tablename__ = 'login_session'
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    jti = db.Column(db.Text)

    user = relationship('User')

    def __init__(self, user_id, jti):
        self.user_id = user_id
        self.jti = jti
```

```python
    jti = get_jti(refresh_token)
    _user['token'] = access_token
    _user['refresh'] = refresh_token
    login_session = LoginSession.query.filter_by(
        user_id=user.id).first()
    if login_session:
        login_session.jti = jti
    else:
        new_login_session = LoginSession(
            user.id, jti)
        db.session.add(new_login_session)
    try:
        db.session.commit()
    except Exception as e:
        print(e)
        abort(400, e)
    return jsonify({ 'message': 'login successfully', 'data': _user })
```

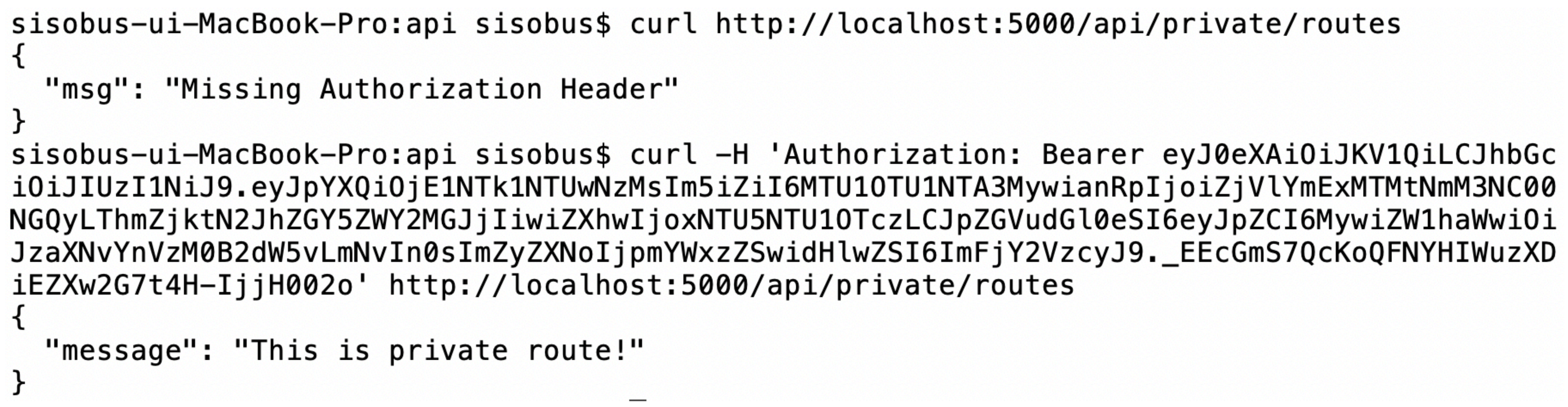
Authentication

Private route 구현

1. @jwt_required 를 이용해 쉽게 구현
2. HTTP Header에 valid한 Authorization이 있어야만 성공적으로 호출됨



```python
class PrivateRoute(Resource):
    @jwt_required
    def get(self):
        return jsonify({ 'message': 'This is private route!'})

api.add_resource(PrivateRoute, '/api/private/routes')
```



```
sisobus-ui-MacBook-Pro:api sisobus$ curl http://localhost:5000/api/private/routes
{
  "msg": "Missing Authorization Header"
}
sisobus-ui-MacBook-Pro:api sisobus$ curl -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc
iOiJIUzI1NiJ9.eyJpYXQiOjE1NTk1NTUwNzMsIm5iZiI6MTU1OTU1NTA3MywianRpIjoiZjVlYmExMTMtNmM3NC00
NGQyLThmZjktN2JhZGY5ZWY2MGJjIiwiZXhwIjoxNTU5NTU1OTczLCJpZGVudGl0eSI6eyJpZCI6MywiZW1haWwiOi
JzaXNvYnVzM0B2dW5vLmNvIn0sImZyZXNoIjpmYWxzZSwidHlwZSI6ImFjY2VzcyJ9._EEcGmS7QcKoQFNYHIWuzXD
iEZXw2G7t4H-IjjH002o' http://localhost:5000/api/private/routes
{
  "message": "This is private route!"
}
```

Authentication

Refresh 구현

1. @jwt_refresh_token_required 를 이용
2. LoginSession에서 refresh token jti와 비교
3. 존재하면 새로운 access_token 발급
4. 반환

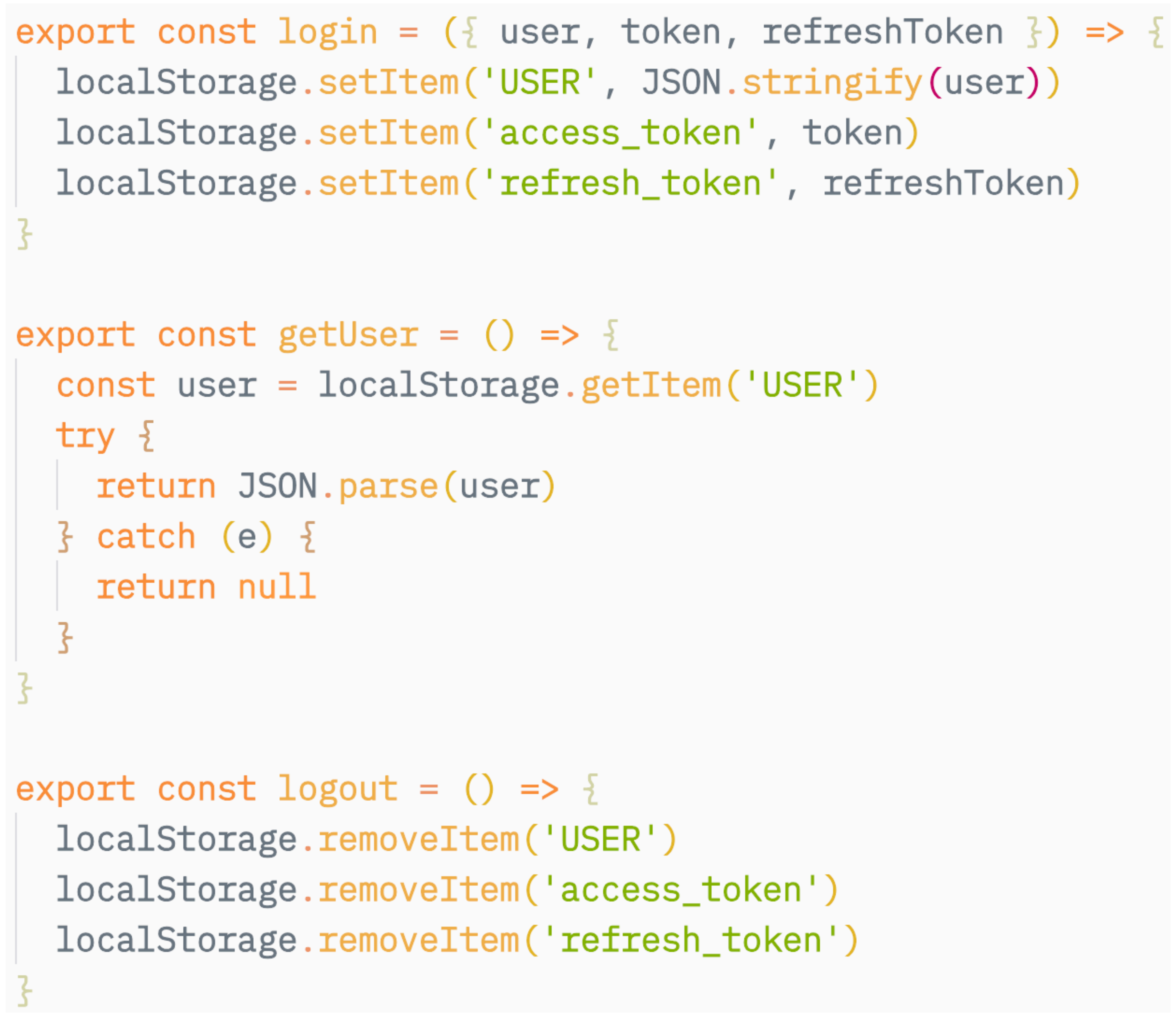
```python
app = Flask(__name__)
app.config.update({
    'SQLALCHEMY_TRACK_MODIFICATIONS': True,
    "SQLALCHEMY_DATABASE_URI": SQLALCHEMY_DATABASE_URI,
    'SECRET_KEY': 'THISISSECRETKEYOFTHISPROJECTHAHA',
    'JWT_ACCESS_TOKEN_EXPIRES': timedelta(minutes=15),
    'JWT_REFRESH_TOKEN_EXPIRES': timedelta(days=30)
})
cors = CORS(app)
api = Api(app)
db.init_app(app)
jwt = JWTManager(app)
```

```python
class UserRefresh(Resource):
    @jwt_refresh_token_required
    def post(self):
        current_user = get_jwt_identity()
        login_session = LoginSession.query.filter_by(
            user_id=current_user['id']).first()
        if login_session is None:
            abort(401)
        raw_jwt = get_raw_jwt()
        jti = raw_jwt['jti']
        if login_session.jti != jti:
            abort(401)
        ret = {
            'token': create_access_token(identity=current_user)
        }
        return jsonify({'ok': True, 'data': ret})

api.add_resource(UserRefresh, '/api/auth/refresh')
```

Authentication

Client 구현

1. LocalStorage에 정보를 담을예정
2. LocalStorage에 정보가 있다? 로그인 된 상태
3. 로그아웃 = LocalStorage의 정보 삭제



```
export const login = ({ user, token, refreshToken }) => {
  localStorage.setItem('USER', JSON.stringify(user))
  localStorage.setItem('access_token', token)
  localStorage.setItem('refresh_token', refreshToken)
}

export const getUser = () => {
  const user = localStorage.getItem('USER')
  try {
    return JSON.parse(user)
  } catch (e) {
    return null
  }
}

export const logout = () => {
  localStorage.removeItem('USER')
  localStorage.removeItem('access_token')
  localStorage.removeItem('refresh_token')
}
```

Authentication

Client 구현 (PrivateRoute)

1. PrivateRoute Component를 구현
2. LocalStorage에 존재하지 않다면 LoginPage로 Redirect

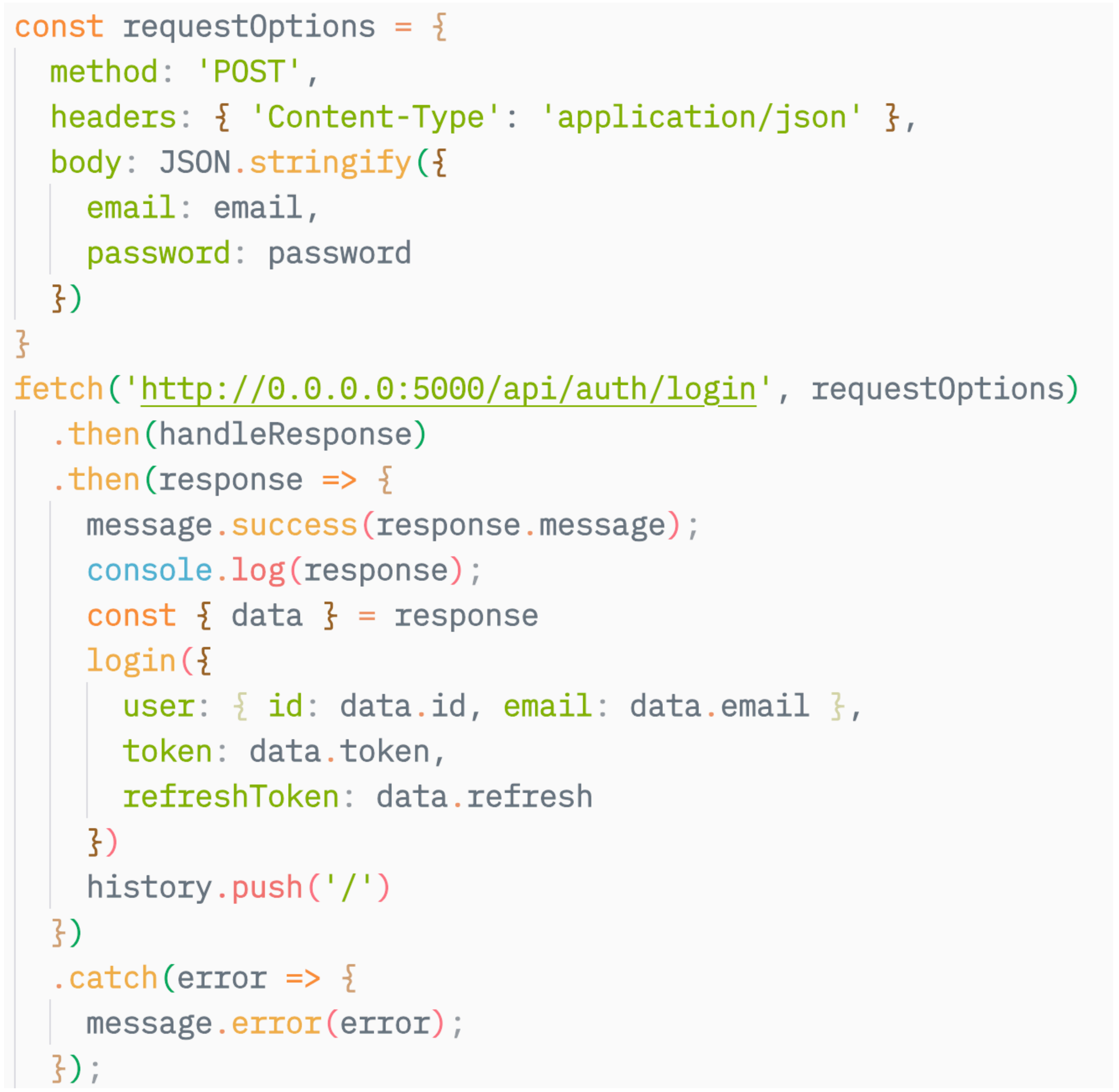
```
import React from 'react'
import { Route, Redirect } from 'react-router-dom'

export const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route
    {...rest}
    render={props =>
      localStorage.getItem('USER') &&
      localStorage.getItem('access_token') &&
      localStorage.getItem('refresh_token') ? (
        <Component {...props} />
      ) : (
        <Redirect
          to={{ pathname: '/login', state: { from: props.location } }}
        />
      )
    }
  />
)
```

```
import React from 'react';
import './App.css';
import MainPage from './_components/Main'
import BlankPage from './_components/BlankPage'
import { LoginPage } from './_components/LoginPage'
import { PrivateRoute } from './_components/PrivateRoute'
import { Router, Route } from "react-router-dom"
import { history } from './_components/history'

function App() {
  return (
    <div className="App">
      <Router history={history}>
        <PrivateRoute path="/" exact component={MainPage} />
        <Route path="/blank" exact component={BlankPage} />
        <Route path="/login" exact component={LoginPage} />
        <Route path="/register" exact component={LoginPage} />
        <Route path="/secret" exact component={MainPage} />
      </Router>
    </div>
  );
}


export default App;
```

Authentication

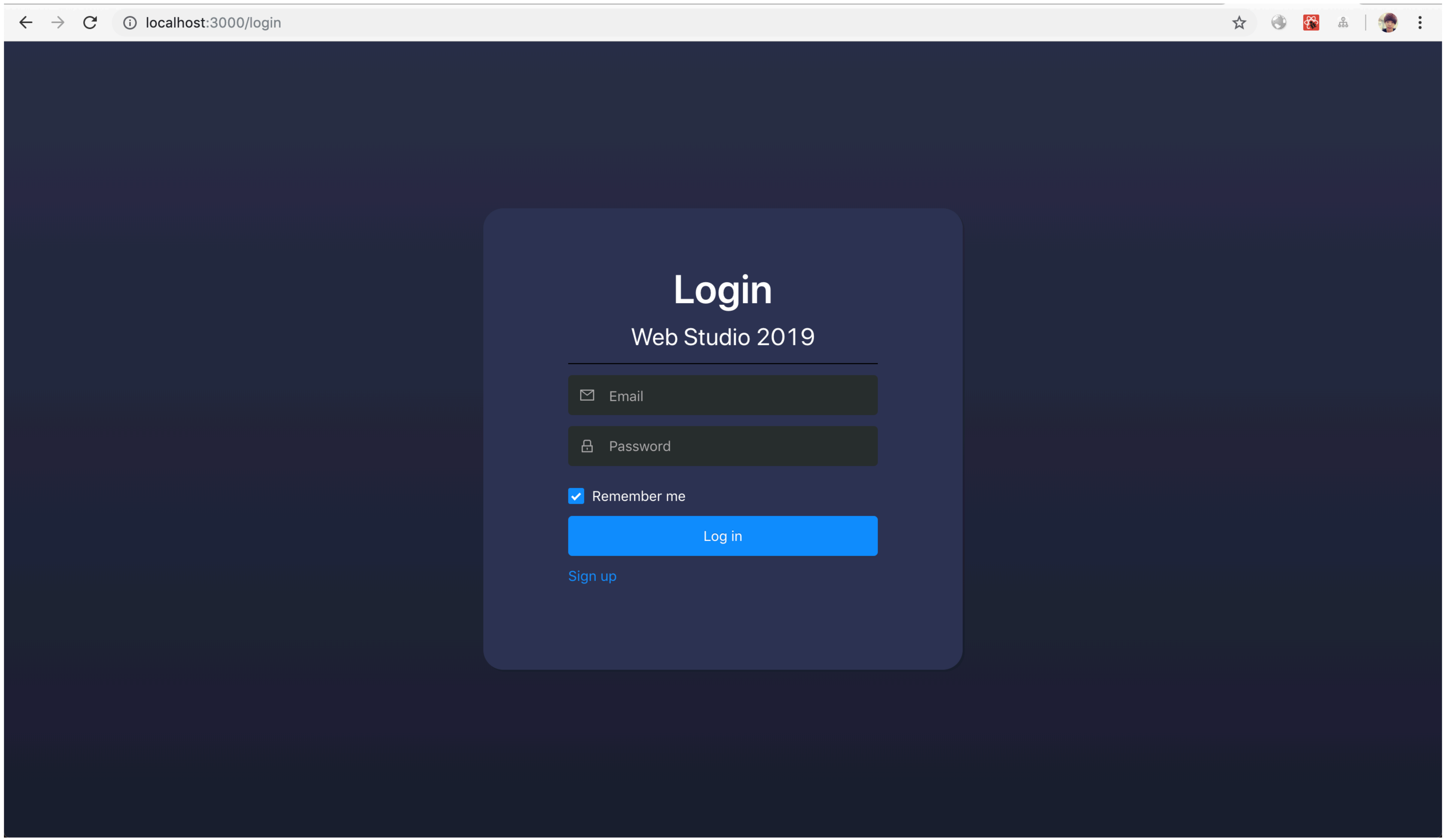
Client 구현 (LoginPage)

1. 잘 구현하고
2. Form submit 시 authentication.login 호출
   〉〉 LocalStorage에 tokens 정보 저장



```javascript
const requestOptions = {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    email: email,
    password: password
  })
}
fetch('http://0.0.0.0:5000/api/auth/login', requestOptions)
  .then(handleResponse)
  .then(response => {
    message.success(response.message);
    console.log(response);
    const { data } = response
    login({
      user: { id: data.id, email: data.email },
      token: data.token,
      refreshToken: data.refresh
    })
    history.push('/')
  })
  .catch(error => {
    message.error(error);
  });
```

Authentication



localhost:3000/login

Login

Web Studio 2019

Email

Password

✓ Remember me

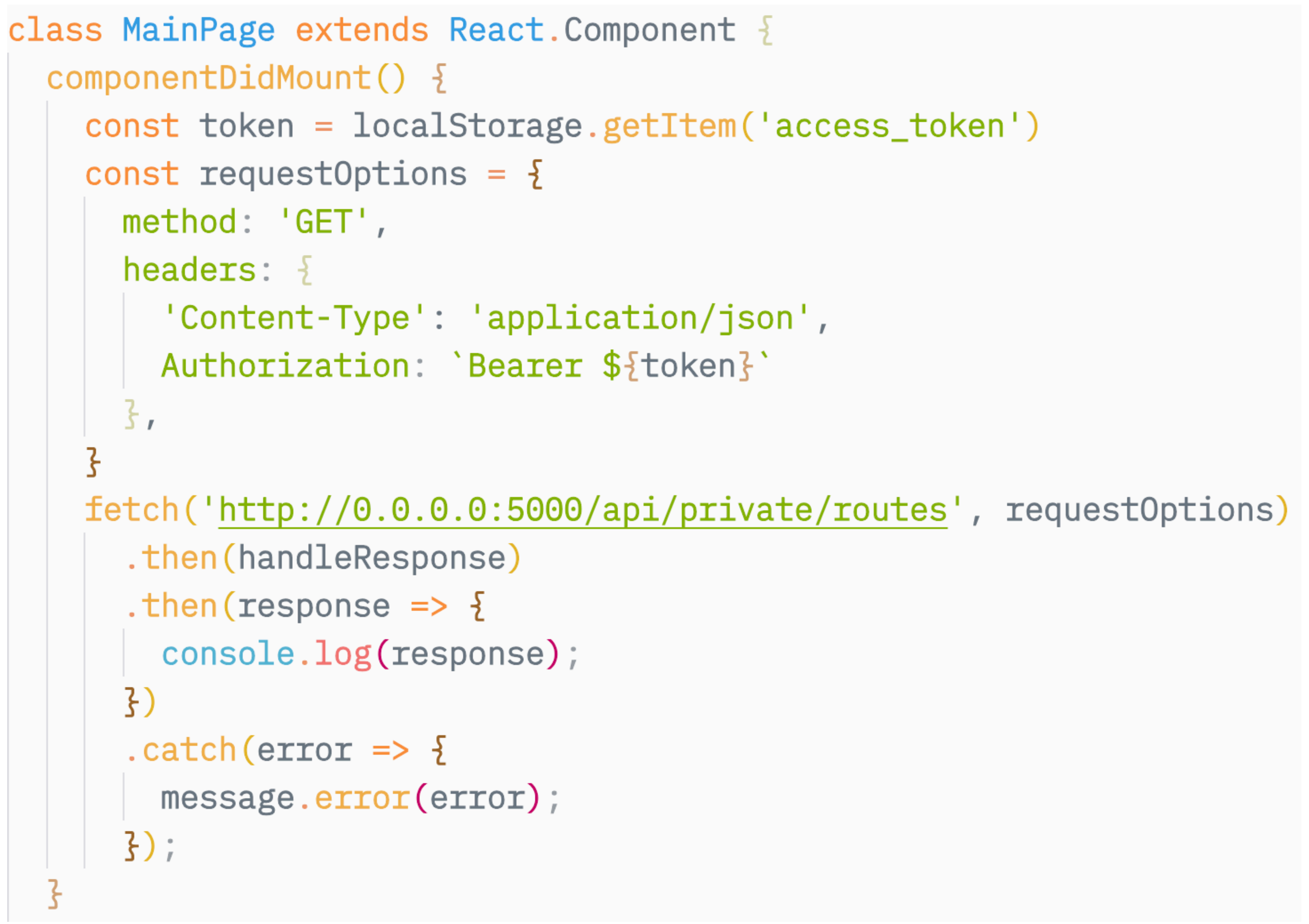
Log in

Sign up

Authentication

Client 구현 (Private route 호출)

1. fetch시 headers에 Authorization 정보 담아서 보냄
2. 이 부분을 MiddleWare로 감싸면 편해짐



```
class MainPage extends React.Component {
  componentDidMount() {
    const token = localStorage.getItem('access_token')
    const requestOptions = {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${token}`
      },
    }
    fetch('http://0.0.0.0:5000/api/private/routes', requestOptions)
      .then(handleResponse)
      .then(response => {
        console.log(response);
      })
      .catch(error => {
        message.error(error);
      });
  }
```

Q & A