유가네

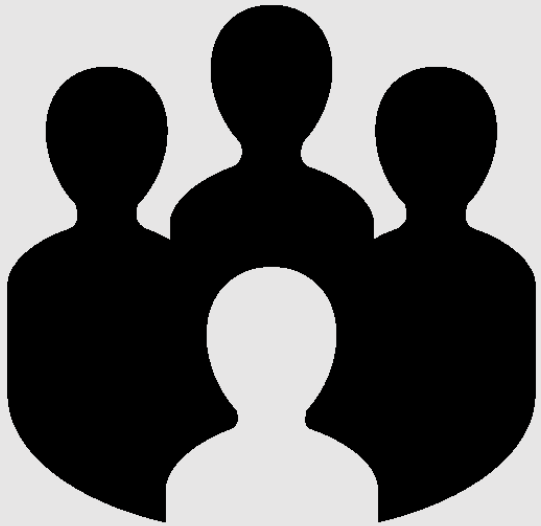
팀장: 유현진
팀원: 권현주, 나지혜, 한지예

THE CONTENTS

조원편성

이름	역할
유현진	프로젝트 총괄
나지혜	데이터베이스 구축 및 관리
권현주	메일시스템 및 인증서 구현
한지예	메일시스템 및 인증서 구현

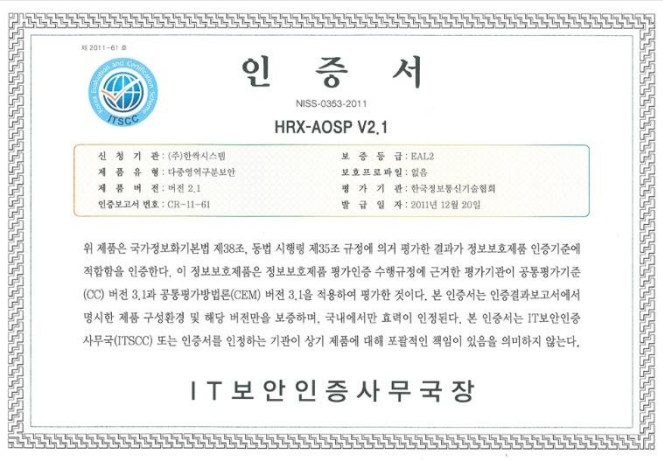
프로젝트목적



이름, 전화번호

프로젝트목적

사업자



제 2011-61 호

인 증 서

NISS-0353-2011

HRX-AOSP V2.1

신 청 기 관 : (주)한싹시스템	보 증 등 급 : EAL2
제 품 유 형 : 다중영역구분보안	보호프로파일 : 없음
제 품 버 전 : 버전 2.1	평 가 기 관 : 한국정보통신기술협회
인증보고서 번호 : CR-11-61	발 급 일 자 : 2011년 12월 20일

위 제품은 국가정보화기본법 제38조, 동법 시행령 제35조 규정에 의거 평가한 결과가 정보보호제품 인증기준에 적합함을 인증한다. 이 정보보호제품은 정보보호제품 평가인증 수행규정에 근거한 평가기관이 공통평가기준(CC) 버전 3.1과 공통평가방법론(CEM) 버전 3.1을 적용하여 평가한 것이다. 본 인증서는 인증결과보고서에서 명시한 제품 구성환경 및 해당 버전만을 보증하며, 국내에서만 효력이 인정된다. 본 인증서는 IT보안인증사무국(ITSCC) 또는 인증서를 인정하는 기관이 상기 제품에 대해 포괄적인 책임이 있음을 의미하지 않는다.

IT보안인증사무국장

손님

설계 및 구축과정 사업자 페이지

Cacert.js

```
var forge = require("node-forge");
var fs = require("fs");
var pki = forge.pki;
// 1. CA 인증서 생성
// generate a keypair and create an X.509v3
certificate
var caKeys = pki.rsa.generateKeyPair(2048);
var caCert = pki.createCertificate();
// CA 개인키 파일 저장
console.log(pki.privateKeyToPem(caKeys.privateKey));
fs.writeFileSync("caPrivateKey.pem",
pki.privateKeyToPem(caKeys.privateKey));
console.log("CA개인키 저장 - caPrivateKey.pem \n");
caCert.publicKey = caKeys.publicKey;
caCert.serialNumber = "01";
caCert.validity.notBefore = new Date();
caCert.validity.notAfter = new Date();
caCert.validity.notAfter.setFullYear(
  caCert.validity.notBefore.getFullYear() + 1
);
var caAttrs = [
  {
    //name: 'commonName', // CN
    shortName: "CN",
    value: "youga"
  },
  {
    //name: 'countryName', // C
    shortName: "C",
    value: "KR"
  },
  {
    //name: 'stateOrProvinceName', // ST
    shortName: "ST",
    value: "Gyeonggi-do"
  },
  {
```

```
    shortName: 'O',
    value: 'Joongbu Univ.'
}, {
    //name: 'organizationalUnitName',
    shortName: 'OU',
    value: 'Dept. of Information Security'
}];
caCert.setSubject(caAttrs);
caCert.setIssuer(caAttrs);

caCert.setExtensions([{
    name: 'basicConstraints',
    cA: true
}, {
    name: 'keyUsage',
    keyCertSign: true,
    digitalSignature: true,
    nonRepudiation: true,
    keyEncipherment: true,
    dataEncipherment: true
}, {
    name: 'extKeyUsage',
    serverAuth: true,
    clientAuth: true,
    codeSigning: true,
    emailProtection: true,
    timeStamping: true
}, {
    name: 'nsCertType',
    client: true,
    server: true,
    email: true,
    objsign: true,
```

서버 측 자체서명인증서 생성

```
}, {
    name: 'subjectAltName',
    altNames: [{
        type: 6, // URI
        value: 'http://example.org/'
    }, {
        type: 7, // IP
        ip: '127.0.0.1'
    }]
}, {
    name: 'subjectKeyIdentifier'
}]);

// self-sign certificate
caCert.sign(caKeys.privateKey);
console.log('CA 자체서명인증서 생성');
console.log(pki.certificateToPem(caCert));
var verified = caCert.verify(caCert);
console.log('CA인증서 생성 후 검증: ' + verified);
console.log();
// CA 인증서 저장
fs.writeFileSync("caCert.pem", pki.certificateToPem
(caCert));
console.log('CA인증서 저장 - caCert.pem');
```

설계 및 구축과정 사업자 페이지

cert.ts

```
onCertRequest() {
  const request = {
    country: this.country,
    state: this.state,
    locality: this.locality,
    organization: this.organization,
    orgUnit: this.orgUnit,
    common: this.common
  }
  this.authService.certRequest(request).subscribe
  (data => {
    if (data.success) {
      this.authService.storeCert(data.cert,
      data.caCert);
      this.flashMessage.show
      ('인증서가 발급되었습니다.', {
        cssClass: 'alert-success',
        timeout: 3000
      });
      this.router.navigate(['dashboard']);
    } else {
      this.flashMessage.show('인증서 발급 실패', {
        cssClass: 'alert-danger',
        timeout: 3000
      });
      this.router.navigate(['cert']);
    }
  });
}
```

auth.service.ts

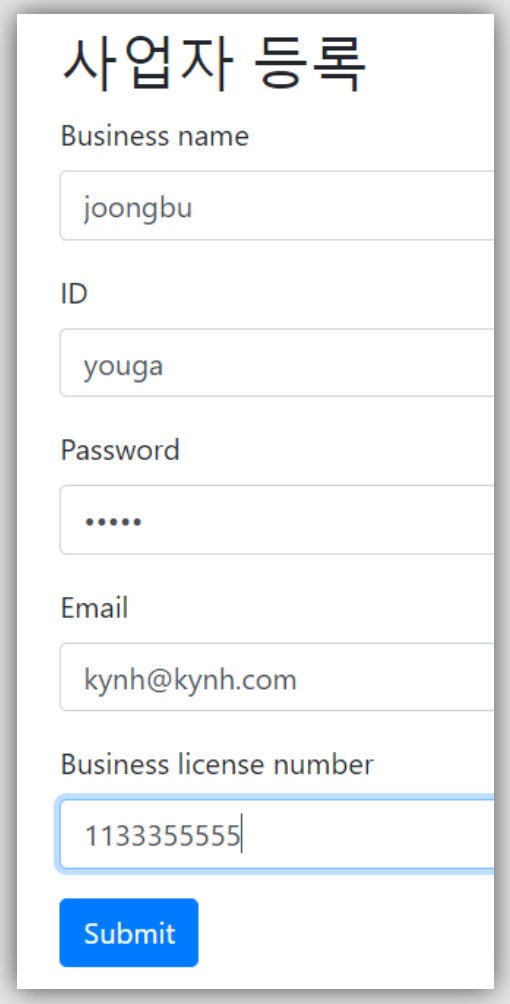
```
certRequest(request): Observable<any> {
  // 키쌍 생성
  let keyPair = pki.rsa.generateKeyPair(2048);
  let publicKey = keyPair.publicKey;
  let privateKey = keyPair.privateKey;
  let publicKeyPem = pki.publicKeyToPem(publicKey);
  let privateKeyPem = pki.privateKeyToPem
  (privateKey);
  // 개인키 저장
  localStorage.setItem("privateKey", privateKeyPem);
  // 인증서 발급 요청 req 생성
  const req = {
    country: request.country,
    state: request.state,
    locality: request.locality,
    organization: request.organization,
    orgUnit: request.orgUnit,
    common: request.common,
    publicKey: publicKeyPem
  };
  const certUrl = this.prepEndpoint("users/cert");
  return this.http.post(certUrl, req, httpOptions);
}
storeCert(cert, caCert) {
  localStorage.setItem("cert", cert);
  localStorage.setItem("caCert", caCert);
}
```

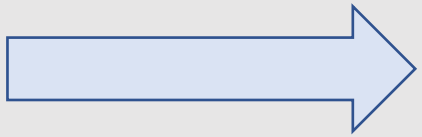
설계 및 구축과정 사업자 페이지

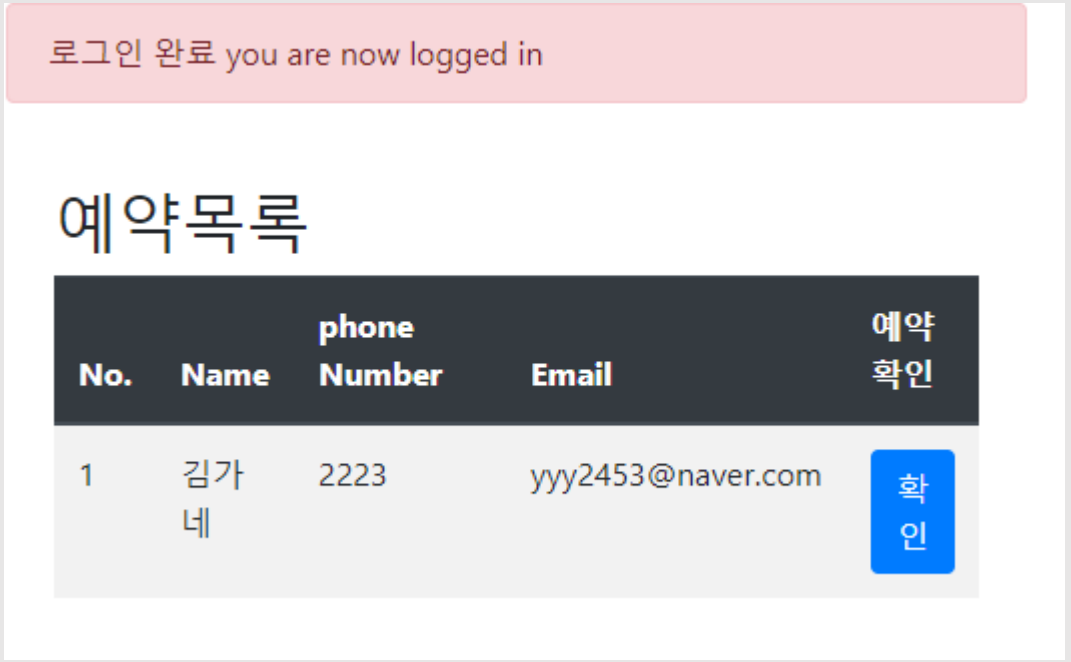
Routers/users.js

```
router.post('/cert', (req, res, next) => {
  let cert = pki.createCertificate();
  cert.publicKey = pki.publicKeyFromPem(req.body.publicKey);
  cert.serialNumber = '01';
  cert.validity.notBefore = new Date();
  cert.validity.notAfter.setFullYear(cert.validity.notBefore.getFullYear() + 1);
  var userAttrs = [{
    shortName: 'CN',
    value: req.body.common
  }, {
    shortName: 'C',
    value: req.body.country
  }, {
    shortName: 'ST',
    value: req.body.state
  }, {
    shortName: 'L',
    value: req.body.locality
  }, {
    shortName: 'O',
    value: req.body.organization
  }, {
    shortName: 'OU',
    value: req.body.orgUnit
  }];
  cert.setSubject(userAttrs);

  var caAttrs = [{
    shortName: 'CN',
    value: caCert.subject.getField('CN').value
  }, {
    shortName: 'C',
    value: caCert.subject.getField('C').value
  }, {
    shortName: 'ST',
    value: caCert.subject.getField('ST').value
  }, {
    shortName: 'L',
    value: caCert.subject.getField('L').value
  }, {
    shortName: 'O',
    value: caCert.subject.getField('O').value
  }, {
    shortName: 'OU',
    value: caCert.subject.getField('OU').value
  }];
  cert.setIssuer(caAttrs);
  cert.setExtensions([{
    name: 'basicConstraints',
    cA: true
  }, {
    name: 'keyUsage',
    keyCertSign: true,
    digitalSignature: true,
    nonRepudiation: true,
    keyEncipherment: true,
    dataEncipherment: true
  }, {
    name: 'extKeyUsage',
    serverAuth: true,
    clientAuth: true,
    codeSigning: true,
    emailProtection: true,
    timeStamping: true
  }, {
    name: 'nsCertType',
    client: true,
    server: true,
    email: true,
    objsign: true,
    sslCA: true,
    emailCA: true,
    objCA: true
  }, {
    name: 'subjectAltName',
    altNames: [{
      type: 6, // URI
      value: 'http://example.org/'
    }, {
      type: 7, // IP
      ip: '127.0.0.1'
    }]
  }, {
    name: 'subjectKeyIdentifier'
  }]);
  cert.sign(caPrivateKey);
  return res.json({
    success: true,
    cert: pki.certificateToPem(cert),
    caCert: caCertPem
  });
});

module.exports = router;
```

설계 및 구축과정 사업자 페이지







설계 및 구축과정 사업자 페이지

인증서 발급

Common Name (이름) - 이름은 use
다.

youga

Country (국가)

KR

State or Province Name (지역)

경기도

Locality Name (도시)

고양시

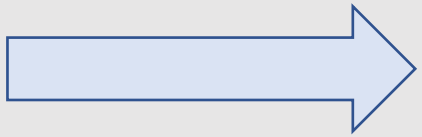
Organization Name (기관)

중부대

Organizational Unit Name (부서)

정보보호학과

인증서 발급 요청



인증서 발급

id_token	JWT eyJhbGciOiJIUzI1NiIsInR5c...
user	{"name":"joongbu","username":...
privateKey	-----BEGIN RSA PRIVATE KEY---...
cert	-----BEGIN CERTIFICATE----- MI...
caCert	-----BEGIN CERTIFICATE----- MI...

설계 및 구축과정 사업자 페이지

Dashboard.html

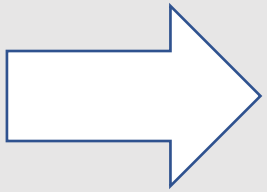
```
<button (click)="createQR()">createQR</button>
<ngx-qrcode [qrc-value]="confirmedQR">
  <p>{{ confirmedQR }}</p>
  {{ user.name }}
</ngx-qrcode>
```

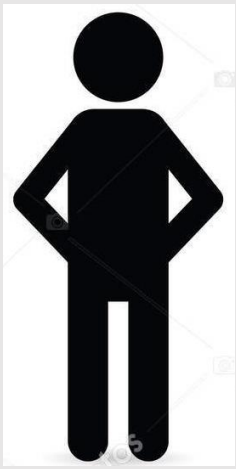
Dashboard.ts

```
createQR() {

  this.confirmedQR = "https://capstone2019-yooga-guest.herokuapp.com/";
}
```

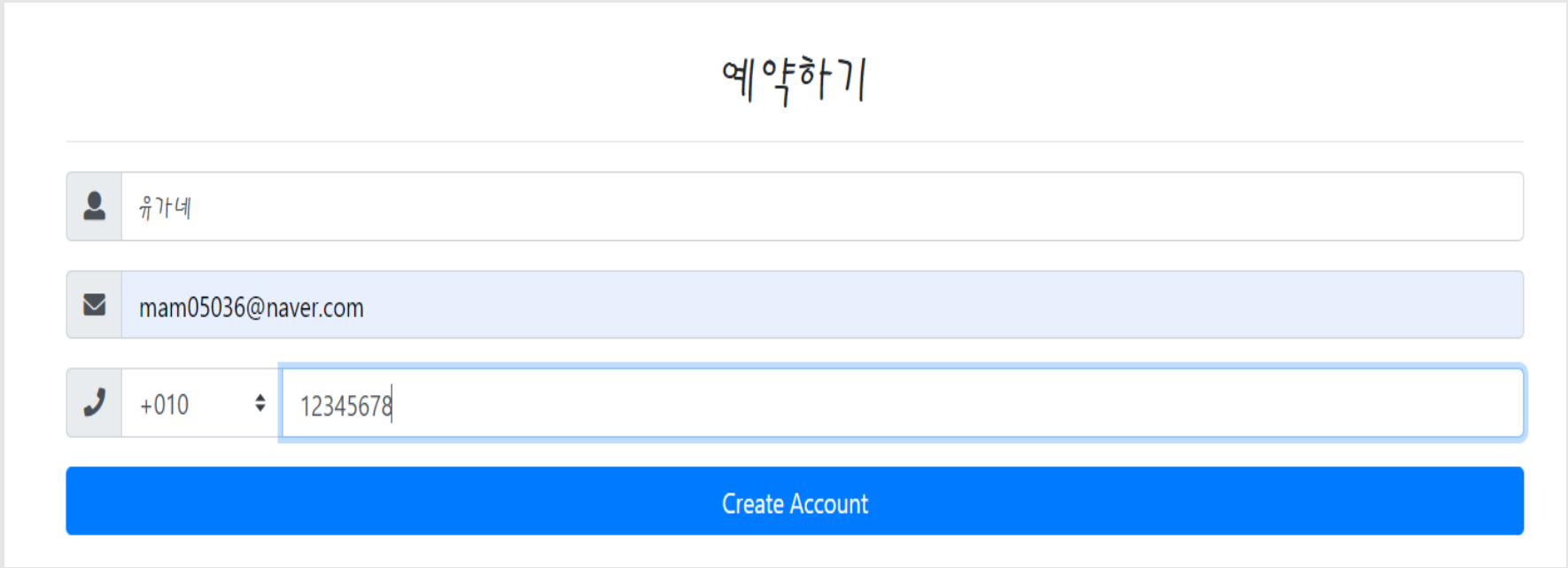
설계 및 구축과정

사업자 페이지





설계 및 구축과정 손님페이지



설계 및 구축과정 손님페이지

userpage.html

```
<form (ngSubmit)="onuserpageSubmit()">
  <div class="form-group input-group">
    <div class="input-group-prepend">
      <span class="input-group-text"> <i class="fa fa-user"></i> </span>
    </div>
    <input
      type="text"
      [(ngModel)]="rename"
      name="rename"
      class="form-control"
    />
  </div>

  <div class="form-group input-group">
    <div class="input-group-prepend">
      <span class="input-group-text">
        <i class="fa fa-envelope"></i>
      </span>
    </div>
    <input
      type="text"
      [(ngModel)]="reemail"
      name="reemail"
      class="form-control"
    />
  </div>
```

userpage.ts

```
onuserpageSubmit() {
  const reuser = {
    rename: this.rename,
    rephone: this.rephone,
    reemail: this.reemail,
    re: this.re
  };
```

설계 및 구축과정 손님페이지

손님 예약목록

No.	성함
1	가나나
2	유현진
3	dddd
4	유가네

설계 및 구축과정 손님페이지

List.html

```
<tr *ngFor="let reuser of reusers">
  <td></td>
  <td>{{ reuser.rename }}</td>
</tr>
```

List.ts

```
export class ListComponent implements OnInit {
  constructor(private authService: AuthService) {}
  reusers: any;
  ngOnInit() {
    this.authService.getList().subscribe(reusers => {
      this.reusers = reusers;
    });
  }
}
```

설계 및 구축과정 손님페이지

userpage.html

```
<div class="form-group">
  <label>예약자 이메일</label>
  <input
    type="text"
    [(ngModel)]="reemail"
    name="reemail"
    class="form-control"
  />
</div>
```

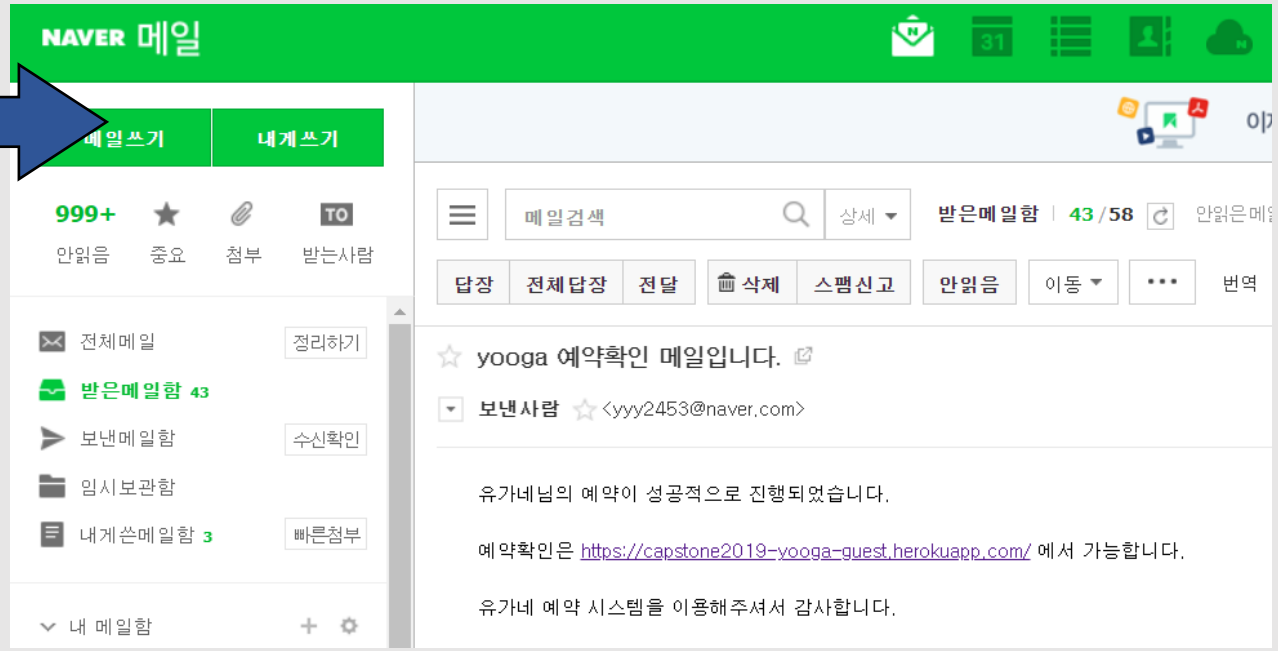
Routes/reusers.js

```
router.post("/nodemailerTest", (req, res, next) => {
  let newReuser = new Reuser({
    rename: req.body.rename,
    rephone: req.body.rephone,
    reemail: req.body.reemail
  });
```

```
let transporter = nodemailer.createTransport({
  service: "naver",
  auth: {
    user: "아이디@naver.com",
    pass: "계정의 비밀번호"
  }
});
```

설계 및 구축과정

```
let mailOptions = {
  from: "yyy2453@naver.com",
  to: newReuser.reemail,
  subject: "유가네 예약확인 메일입니다.",
  html:
    newReuser.rename +
    "님의 예약이 성공적으로 진행되었습니다.\n
    예약확인은 https://
    capstone2019-yooga-guest.herokuapp.com/ 에서
    가능합니다.\n\n " +
    '<h4> 유가네 예약시스템을 이용해주셔서
    감사합니다.</h4><a herf= "https://
    capstone2019-yooga-guest.herokuapp.com/"> ' +
    ' <img src="http://pds.joins.com/news/component/
    htmlphoto_mmdata/201910/26/
    ce877ed2-0800-457f-b9a6-a86044718d40.jpg /></
    p></a>'
};
```

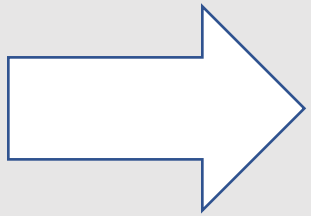


설계 및 구축과정

사업자페이지

예약목록

No.	Name	phone Number	Email	예약확인
1	가나나	2234	yyy2453@naver.com	확인
2	유현진	123	yyy2453@naver.com	확인
3	dddd	234	d@c.com	확인
4	유가네	12345678	mam05036@naver.com	확인



예약목록

No.	Name	phone Number	Email	예약확인
1	가나나	2234	yyy2453@naver.com	확인
2	유현진	123	yyy2453@naver.com	확인
3	dddd	234	d@c.com	확인

설계 및 구축과정

사업자페이지

html

```
<td>
  <button class="btn btn-primary" (click)="remove(reuser)">
    확인
  </button>
</td>
```

ts

```
remove(data) {
  const deluser = {
    rename: data.rename,
    rephone: data.rephone,
    reemail: data.reemail
  };

  this.authService.deleteReuser(deluser).subscribe(row => {
    console.log(row);
    console.log(data);
  });
}
```

Auth.service.ts

```
deleteReuser(deluser): Observable<any> {
  const deleteUrl = this.prepEndpoint("reusers/delete");
  return this.http.post<Reuser>(deleteUrl, deluser, httpOptions);
}
```

성능분석 및 기대효과

예약 시스템을 사용함으로써 개인정보 유출을 최소화 할 수 있는 효과를 기대합니다. 은행 등 대기표를 뽑는 실생활에서의 예약 시스템을 QR코드를 사용함으로써 개인정보 유출을 최소화 하고, 보다 간편하게 다양한 분야에서 활용 가능합니다.

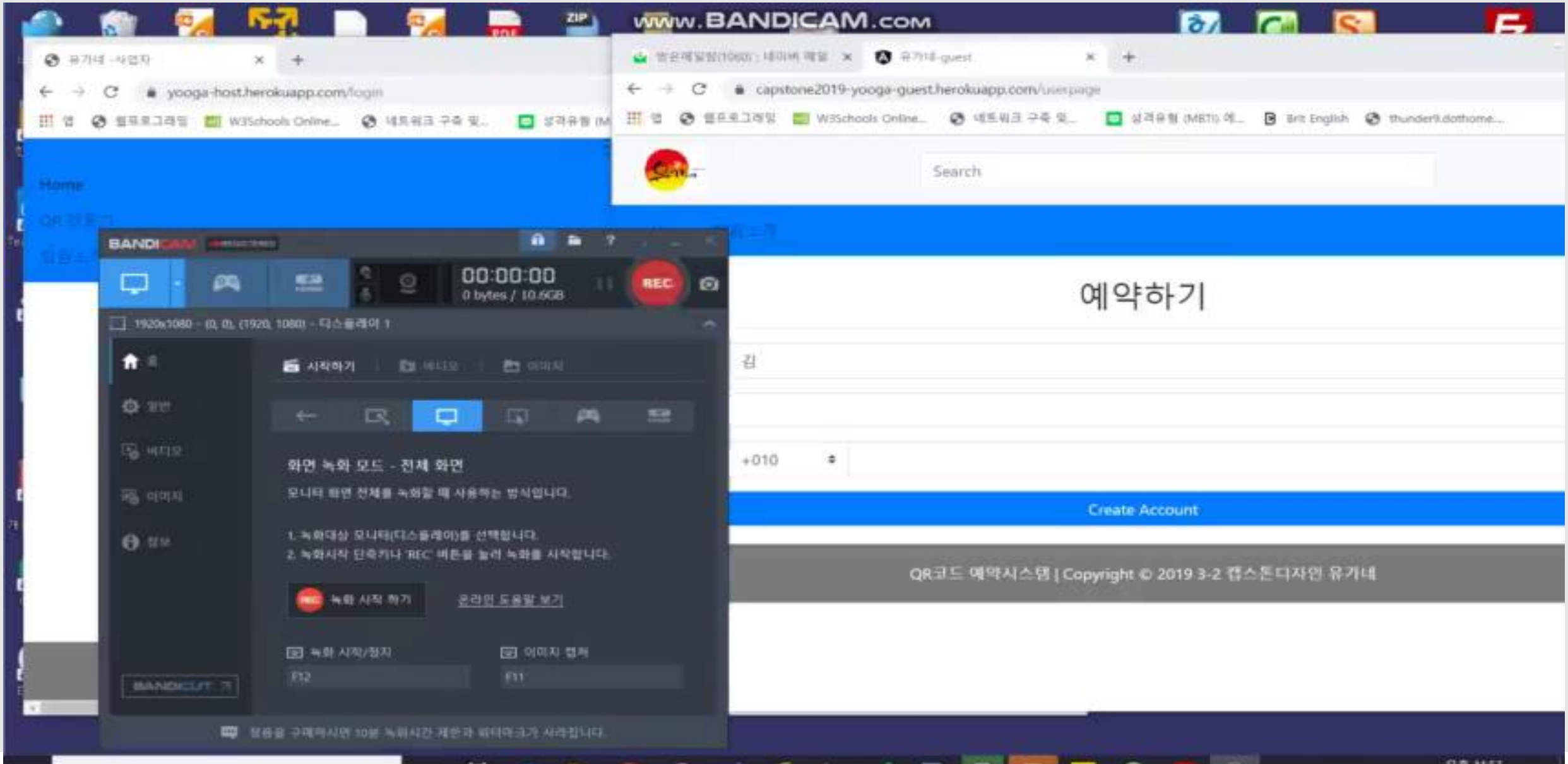
향후계획

카카오톡 플러스 친구 알림 서비스를 이용하여 예약된 정보를 실시간으로 확인할 수 있고 대기 시간 등의 서비스를 제공합니다. 또한 주변 식당 또는 편의시설 위치를 알려주는 위치기반 서비스를 넣을 계획입니다.

카카오톡플러스친구

실습영상

https://yooga-host.herokuapp.com/



Q&A

감사합니다