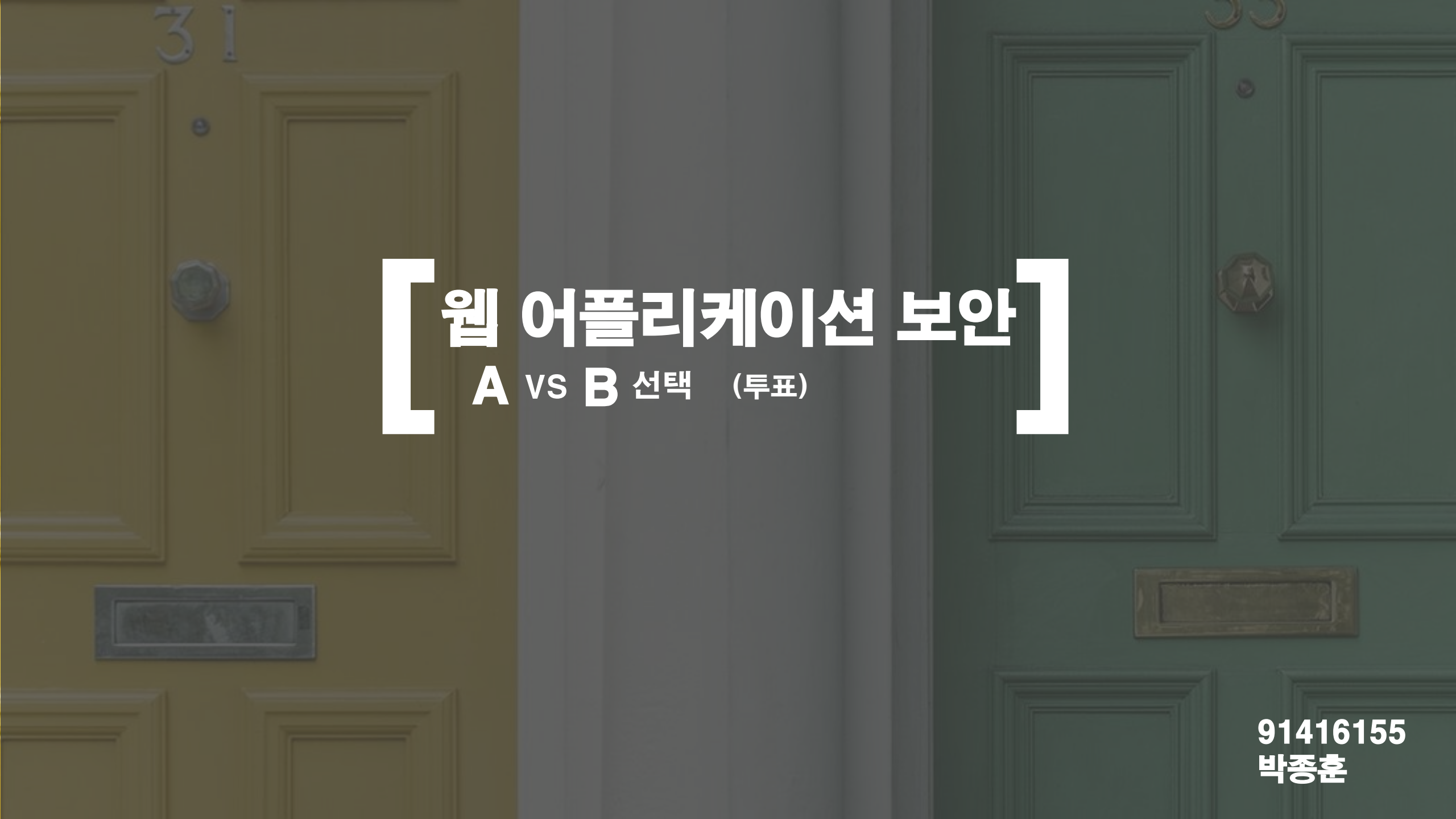
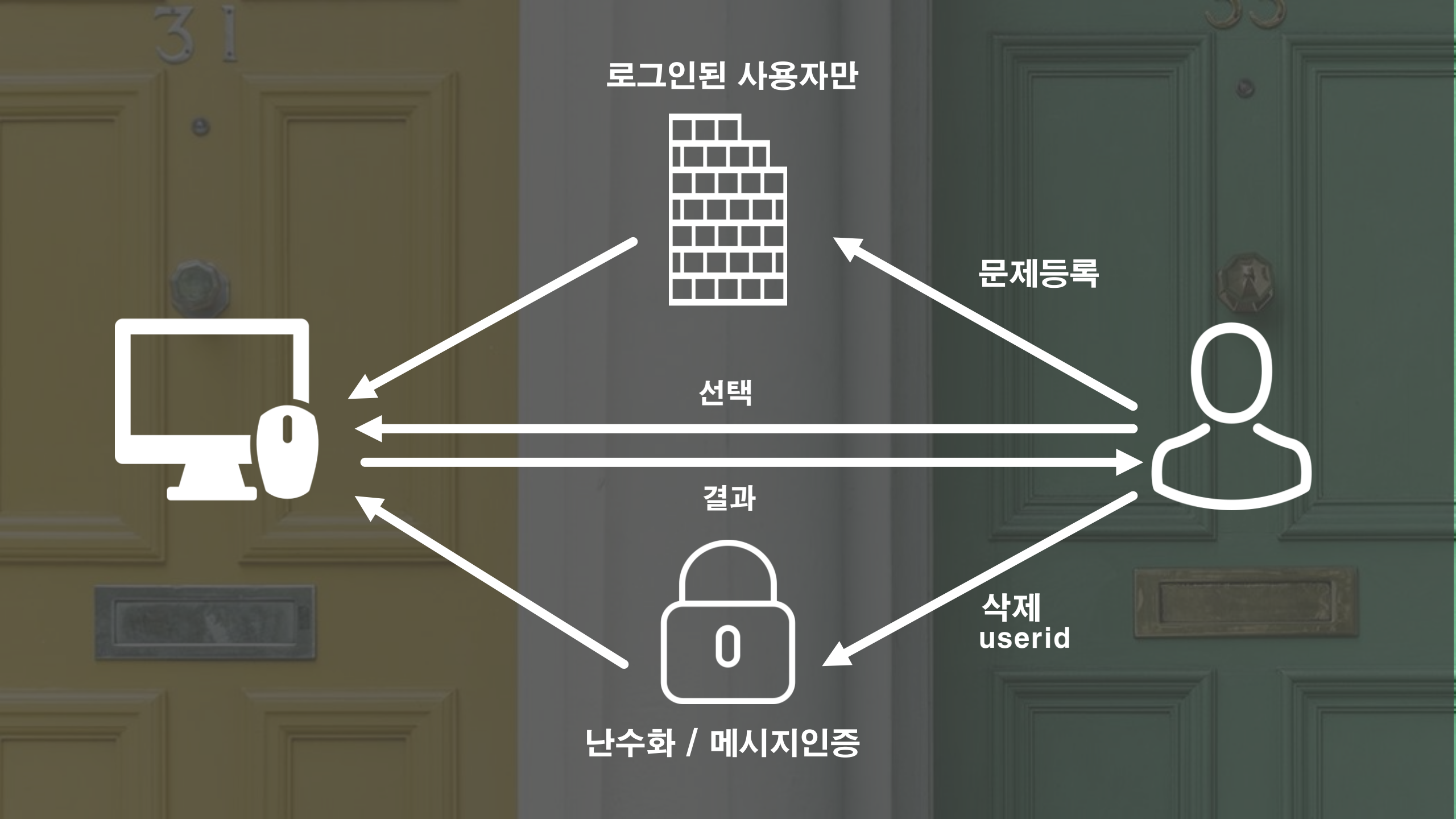


[웹 어플리케이션 보안]

A VS B 선택 (투표)

91416155
박종훈

목차


로그인된 사용자만
문제등록
선택
결과
삭제
userid
난수화 / 메시지인증

로그인 login.component.ts

```
onLoginSubmit(){
  const user = {
    username: this.username,
    password: this.password
  }

  this.authService.authenticateUser(user).subscribe(data => {
    if(data.success) {
      this.authService.storeUserData(data.ptoken, data.stoken, data.user);
      this.flashMessage.show('로그인 되었습니다.',
        {cssClass: 'alert-success', timeout: 5000});
       this.router.navigate(['/']);
  } else {
     this.flashMessage.show(data.msg, {cssClass: 'alert-danger', timeout:
      this.router.navigate(['login']);
    }
  });
  }
}
```

DB에 Username검색(models .user.js)

```
module.exports.getUserByUsername = function(username, callback){
  const query = {username: username};
  User.findOne(query, callback);
}
```

토큰생성(auth.service.ts)

```
router.post('/authenticate', (req, re
  const username = req.body.username
  const password = req.body.password
  User.getUserByUsername(username, (err, user) => {
    if(err) throw err;
    if(!user) {
      return res.json({success : false, msg: '유저를 찾을 수 없습니다.'});
    }
    User.comparePassword(password, user.password, (err, isMatch) => {
      if(err) throw err;
      if(isMatch) {
        const ptoken = 'JWT ' + jwt.sign({data: user}, config.secret, {
          expiresIn: 604800
        });
        const stoken = 'JWT ' + jwt.sign({data: ptoken}, config.secret, {
          noTimestamp: true
        });
        res.json({
          success: true,
          ptoken: ptoken,
          stoken: stoken,
          user: {
            id : user._id,
            name : user.name,
            username : user.username,
            email : user.email
          }
        });
      } else {
        return res.json({success: false, msg:'Wrong password'});
        }
      });
    });
});
```

DB와 Password 매칭(models .user.js)

```
module.exports.comparePassword = function(candidatePassword, hash, callback){
  bcrypt.compare(candidatePassword, hash, (err, isMatch) => {
    if(err) throw err;
    callback(null, isMatch);
  });
}
```

로그인 login.component.ts

```
onLoginSubmit(){
  const user = {
    username: this.username,
    password: this.password
  }

  this.authService.authenticateUser(user).subscribe(data => {
    if(data.success) {
      this.authService.storeUserData(data.ptoken, data.stoken, data.user);
      this.flashMessage.show('로그인 되었습니다.',
        {cssClass: 'alert-success', timeout: 5000});
        this.router.navigate(['/']);
  } else {
      this.flashMessage.show(data.msg, {cssClass: 'alert-danger', timeout:
      this.router.navigate(['login']);
    }
  });
  }
}
```

토큰저장(auth.service.ts)

```
storeUserData(ptoken, stoken, user){
  localStorage.setItem('id_ptoken', ptoken);
  localStorage.setItem('id_stoken', stoken);
  localStorage.setItem('user', JSON.stringify(user));
  this.pToken = ptoken;
  this.sToken = stoken;
  this.user = user;
}
```

문제등록 Choice.component.html/.ts

로그인된 사용자만

```
<body *ngIf="authService.loggedIn()">
  <div class="container center">
    <h2 class="page-header">AvsB 등록</h2>
    <div class="form-group">
      <label>문제</label>
      <input type="text" [(ngModel)]="question" name="question" class="form-control" >
    </div>
    <div class="form-group">
      <label>비교 A</label>
      <input type="text" [(ngModel)]="food_one" name="food_one" class="form-control" >
    </div>
    <div class="form-group">
      <label>비교 B</label>
      <input type="text" [(ngModel)]="food_two" name="food_two" class="form-control" >
    </div>
    <input type="submit" class="btn btn-primary" (click)="onContentSubmit(user.username)" value="등록하기">
  </div>
</body>
```

문제등록 Choice.component.ts

```
onContentSubmit(username){
  const content = {
    question: this.question,
    food_one: this.food_one,
    food_two: this.food_two,
    writer: username
  }

  if(!this.validateService.validateContent(content)){
    this.flashMessage.show('모든 필드를 입력하세요',
    {cssClass: 'alert-danger', timeout: 3000});
    return false;
  }

  this.authService.addContent(content).subscribe(data => {
    if(data.success) {
      this.flashMessage.show('문제 등록 성공', {cssClass: 'alert-success', timeout: 3000});
      location.reload();
    } else {
      this.flashMessage.show(data.msg + '문제 등록이 실패했습니다', {cssClass: 'alert-danger', timeout: 3000});
    }
  })
}
```

입력받은 내용

모든 필드 확인(validate.service.ts)

```
validateContent(content) {
  if(content.question == undefined || content.food_one == undefined || content.food_two == undefined) {
    return false;
  } else {
    return true;
  }
}
```

문제등록 routers.content.js

```
router.post('/addcontent', (req, res, next) => {
  let newContent = new Choice({
  question: req.body.question,
  food_one: req.body.food_one,
  food_two: req.body.food_two,
  count_one: req.body.count_one,
  count_two: req.body.count_two,
  writer: req.body.writer
  });

  Choice.getContentByContent(newContent.question, (err, que) => {/
    if(err) throw err;
    if(que) {
      return res.json({success : false, msg: '문제가 중복입니다. '});
    } else {
      Choice.addChoice(newContent, (err, content)=>{
        if(err) {
          res.json({success: false, msg:'문제 등록에 실패했습니다.'});
        } else {
          res.json({success: true, msg: '문제 등록에 성공했습니다.'});
        }
      });
    }
  });
});
```

중복검사(models.choice.js)

```
module.exports.getContentByContent = function(question, callback){
  const query = {question: question};
  Choice.findOne(query, callback);
}
```

추가(models.choice.js)

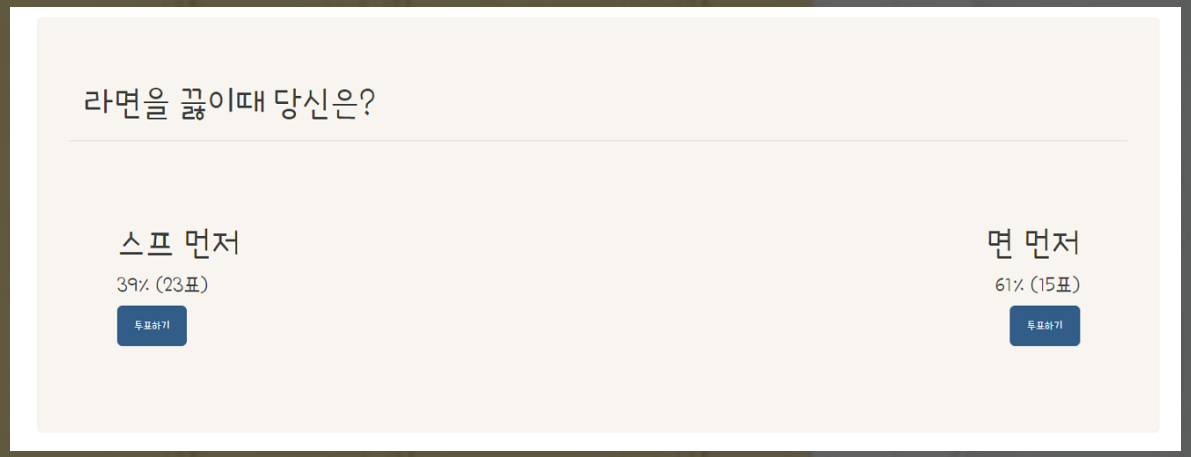
```
module.exports.addChoice = function(newChoice, callback){
  newChoice.save(callback);
}
```

문제등록 Choice.component.ts

DB에 저장되는 스키마

```
const ChoiceSchema = mongoose.Schema({
  question:{
    type: String,
    required: true
  },
  food_one: String,
  percent_one: {
    type: Number,
    default: 0
  },
  count_one: {
    type:Number,
    default: 0
  },
  food_two: String,
  percent_two: {
    type: Number,
    default: 0
  },
  count_two: {
    type:Number,
    default: 0
  },
  writer: String,
});
```

문제선택 resultat.component.html/.ts




```
<div *ngFor="let content of content">
  <div class="jumbotron jm"  >
    <div class="row">
      <div class="container ">
        <div class="container">
          <h1>{{content.question}}</h1>
        </div>
        <hr>
        <div class="row">
          <div class="col-md-6">
            <div class="jumbotron">
              <div class="container">
                <h1>{{content.food_one}}</h1>
                <h4 >{{content.percent_one}}% ({{content.count_one}}표)</h4>
                <button type="button" class="btn btn-primary btn-lg" (click)="onFoodChoice(content.food_one, content.count_one, content.count_two)"  >투표하기</button>
              </div>
            </div>
          </div>
          <div class="col-md-6">
            <div class="jumbotron">
              <div class="container two">
                <h1>{{content.food_two}}</h1 >
                <h4>{{content.percent_two}}% ({{content.count_two}}표)</h4>
                <button type="button"  class="btn btn-primary btn-lg" (click)="onFoodChoice(content.food_two, content.count_one, content.count_two)">투표하기</button>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
<button type="button" class="btn btn-primary btn-lg" (click)="onFoodChoice(content.food_one, content.count_one, content.count_two)"  >투표하기</button>
```

```
<button type="button"  class="btn btn-primary btn-lg" (click)="onFoodChoice(content.food_two, content.count_one, content.count_two)">투표하기</button>
```

문제선택 resultat.component.ts

```
onFoodChoice(food, count1, count2){
  const choice = {
    food_one: food,
    count_one: count1,
    count_two: count2
  }
  this.authService.choUser(choice).subscribe(data => {
    if(data.success) {
      this.flashMessage.show(data.msg, {cssClass: 'alert-success', timeout: 5000});
      location.reload();
    } else {
      this.flashMessage.show(data.msg, {cssClass: 'alert-danger', timeout: 5000});
    }
  });
}
```

문제선택 routers.content.js

```
router.post('/cho', (req, res, next) => {
  let newChoice = new Choice({
    food_one: req.body.food_one,
    count_one: req.body.count_one,
    count_two: req.body.count_two
  });
  Choice.getFoodByChoice_one(newChoice, (err, food) => {
    if(err) throw err;
    if(!food) {
      Choice.getFoodByChoice_two(newChoice, (err, food) => {
        if(err) throw err;
        if(!food) {
          res.json({success: false, msg: '값이 없습니다.'});
        } else {
```

대상확인(models.choice.js)

```
module.exports.getFoodByChoice_one = function(newChoice, callback){
  const query = {food_one: newChoice.food_one};
  Choice.findOne(query, callback);
}

module.exports.getFoodByChoice_two = function(newChoice, callback){
  const query = {food_two: newChoice.food_one};
  Choice.findOne(query, callback);
}
```

문제선택 routers.content.js

```
Choice.updateOne({food_two : newChoice.food_one}, {$inc: {count_two: +1}}, (err, choice)=>{
  if(err) throw err;
  if(choice) {
    var one = (newChoice.count_one)+0;
    var two = (newChoice.count_two)+1;
    var sum = one+two;
    var per2 = (two/sum) * 100;
    var per1 = 100 - per2;
    per1 = Math.floor(per1);
    per2 = Math.floor(per2);
    let setChoice = new Choice({
      food_one: req.body.food_one,
      percent_one: per1,
      percent_two: per2
    });
      Choice.updateOne({ food_two : newChoice.food_one }, { $set: {percent_one: setChoice.pe
      if(err) throw err;
      res.json({success: true, msg: '성공적으로 통계가 집계되었습니다.'});
    });
  }
});
```

퍼센트 계산

카운터 증가

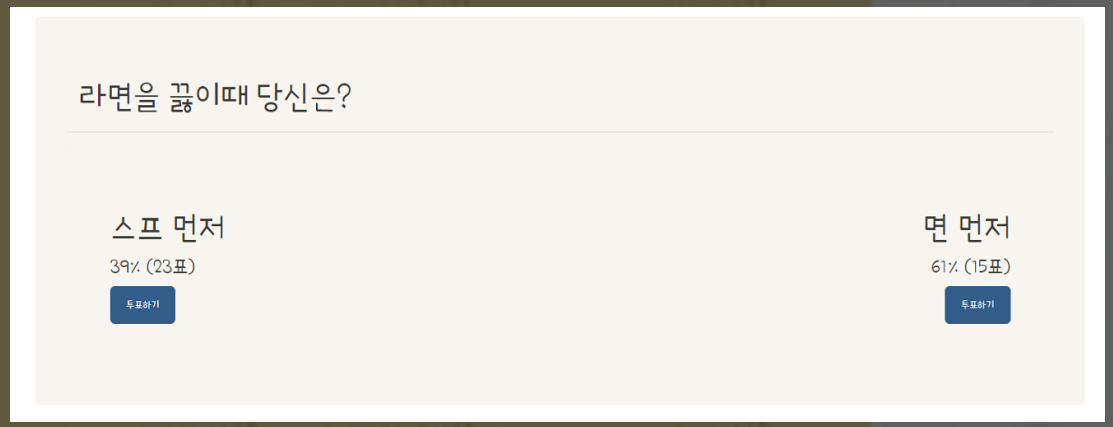
문제선택 routers.content.js

```
} else {
  Choice.updateOne({ food_one : newChoice.food_one }, { $inc: { count_one: +1 } },(err, choice)=>{
    if(err) throw err;
    if(choice) {
      var one = (newChoice.count_one)+1;
      var two = (newChoice.count_two)+0;
      var sum = one+two;
      var per1 = (one/sum) * 100;
      var per2 = 100 - per1;
      per1 = Math.floor(per1);
      per2 = Math.floor(per2);
      let setChoice = new Choice({
        food_one: req.body.food_one,
        percent_one: per1,
        percent_two: per2
      });
      Choice.updateOne({ food_one : newChoice.food_one }, { $set: { percent_one: setChoice.percent_one, percent_two: setChoice.percent_two  }
        if(err) throw err;
        res.json({success: true, msg: '성공적으로 통계가 집계되었습니다.'});
      });
    }
```

퍼센트 계산

카운터 증가

문제선택 resultat.component.html/.ts




```
ngOnInit() {
  this.authService.getContent().subscribe(content => {
    this.content = content.content;
  }, err => {
    console.log(err);
    return false;
  });
}
```

```
<div *ngFor="let content of content">
    <div class="row">
      <div class="container ">
        <div class="container">
          <h1>{{content.question}}</h1>
        </div>
        <hr>
        <div class="row">
          <div class="col-md-6">
<h1>{{content.food_one}}</h1>
<h4 >{{content.percent_one}}% ({{content.count_one}}표)</h4>
            </div>
          </div>
        </div>
        <div class="col-md-6">
<h1>{{content.food_two}}</h1 >
<h4>{{content.percent_two}}% ({{content.count_two}}표)</h4>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

프로필

profile.component.html/.ts

인증정보 계산

```
this.currT = this.authService.getCurrTime();
this.ptoken = this.authService.getPubToken();
this.stoken = this.authService.getSecToken();
this.auth = this.authService.computeAuth(this.currT, this.stoken);
this.authService.getProfile(this.ptoken, this.currT, this.auth).subscribe(profile => {
  this.user = profile.user;
},
err => {
  console.log(err);
  this.authService.logout();
  this.router.navigate(['login']);
  return false;
});
}
```

혜시화(auth.service.ts)

```
getCurrTime() {
  return new Date().getTime();
}

getPubToken() {
  return localStorage.getItem('id_ptoken');
}

getSecToken() {
  return localStorage.getItem('id_stoken');
}

computeAuth(currT, stoken){
  var md = forge.md.sha256.create();
  md.update(currT+stoken);
  return md.digest().toHex();
}
```

프로필

profile.component.html/.ts

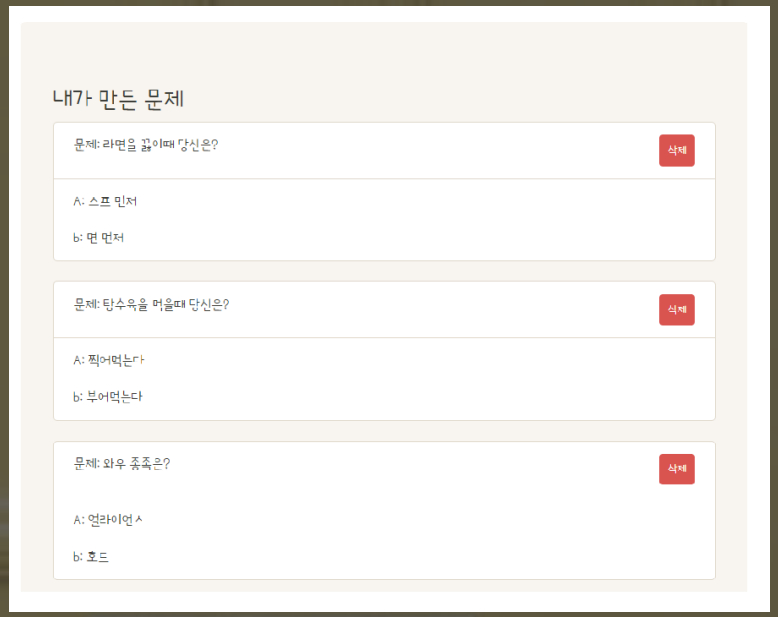
```
getProfile(ptoken, currT, auth){
  let headers = new Headers();
  headers.append('Authorization', ptoken);
  headers.append('Ctime', currT);
  headers.append('Auth', auth);
  headers.append('Content-Type', 'application/json');
  let ep = this.prepEndpoint('users/profile');
  return this.http.get(ep, {headers: headers})
  .pipe(map(res => res.json()));
}
```

검증

```
router.get('/profile', passport.authenticate('jwt', {session:false}), (req, res, next) => {
    const ptoken = req.headers.authorization;
    const currT = req.headers.ctime;
    const auth = req.headers.auth;
    const stoken = 'JWT '+jwt.sign({data: ptoken}, config.secret, {
      noTimestamp: true
    });
    var md = forge.md.sha256.create();
    md.update(currT+stoken);
    const auth2 = md.digest().toHex();
    const serverTime = new Date().getTime();
    const diff = serverTime - currT;
    if(auth==auth2 && diff<100000) {
      res.json({user: req.user});
    }
});
```

문제삭제

profile.component.html/.ts




```
<div *ngIf="user">
  <div class="row">
    <div class="col-sm-6 col-md-4">
      <div class="jumbotron">
        <table class="table">
          <tbody>
            <tr>
              <h1>{{user.username}}</h1>
            </tr>
            <tr>
              <td>이름</td>
              <td>{{user.name}}</td>
            </tr>
            <tr>
              <td>이메일</td>
              <td>{{user.email}}</td>
            </tr>
          </tbody>
        </table>
        <p><a class="btn btn-primary btn-lg" (click)="onLogoutClick()" href="#" role="button">로그아웃</a></p>
      </div>
    </div>
    <div class="col-sm-6 col-md-8">
      <div class="jumbotron ">
        <h3>내가 만든 문제</h3>
        <div *ngFor="let content of content">
          <div *ngIf="user.username == content.writer">
            <ul class="list-group que">
              <li class="list-group-item">문제: {{content.question}} <button  type
              <li class="list-group-item">A: {{content.food_one}}<p></p>b: {{conte
            </ul>
            </div>
          </div>
        </div>
      </div>
  </div>
</div>
```

```
(click)="ondelques(content._id)">삭제</button></li>
```

문제삭제

profile.component.ts

난수화 / 메시지인증

```
ondelques(delid){
  this.currT = this.authService.getCurrTime();
  this.key = this.authService.computeAuth(this.currT, this.stoken);
  this.mac = this.authService.getMac(delid, this.key);
  const encrypted = this.authService.getEncrypt(delid, this.key);
  this.authService.delMacEncrypt(encrypted, this.currT, this.ptoken, this.mac).subscribe(data => {
    if(data.success) {
      this.flashMessage.show(data.msg, {cssClass: 'alert-success', timeout: 5000});
        location.reload();
    } else {
        this.flashMessage.show(data.msg, {cssClass: 'alert-danger', timeout: 5000});
      }
  });
}
```

난수생성(auth.service.ts)

```
getMac(plantext, key){
    var md = forge.md.sha256.create();
    md.update(plantext+key);
    return md.digest().toHex();
}
```

AES암호화(auth.service.ts)

```
getEncrypt(plantext, key){
  var key1 = forge.util.hexToBytes(key);
  var cipher = forge.cipher.createCipher('AES-ECB', key1);
  cipher.start();
  cipher.update(forge.util.createBuffer(plantext, 'binary'));
  cipher.finish();
  var ciphertext = cipher.output;
  return forge.util.bytesToHex(ciphertext);
}
```

문제삭제

routers.content.js

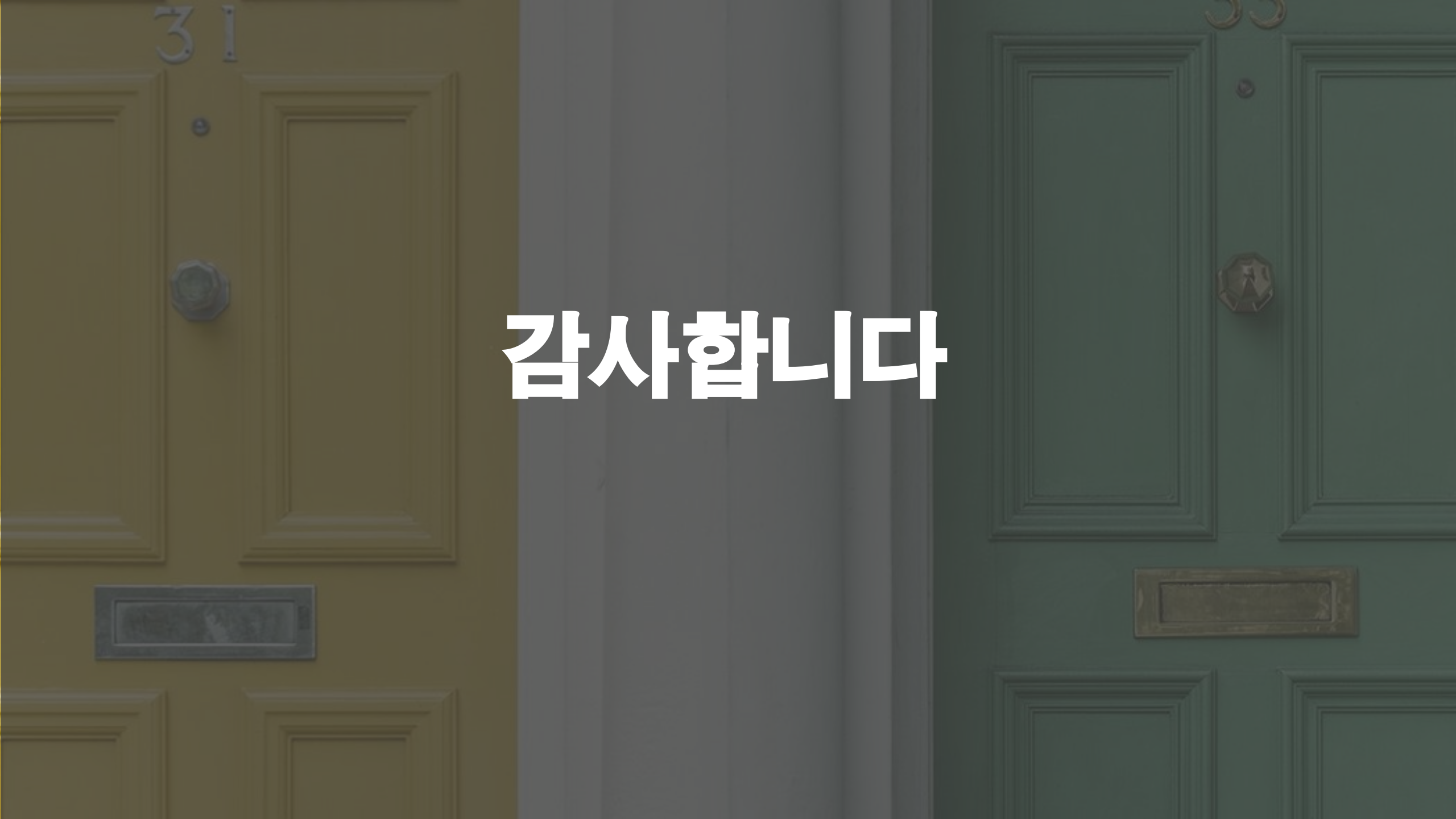
```
router.get('/del', (req, res, next) => {
  const ptoken = req.headers.authorization;
  const currT = req.headers.ctime;
  const encrypted = req.headers.enc;
  const mac1 = req.headers.mac;
  const stoken = 'JWT ' + jwt.sign({data: ptoken}, config.secret, {
    noTimestamp: true
  });
  var md = forge.md.sha256.create();
  md.update(currT+stoken);
  var
  var
  var
  var
  decipher.start();
  decipher.update(forge.util.createBuffer(encrypted1, 'binary'));
  decipher.finish();
  var decrypted = decipher.output;
  var md1 = forge.md.sha256.create();
  md1.update(decrypted+auth);
  var mac2 = forge.util.bytesToHex(md1.digest());
  if(mac1==mac2) {
```

```
송신된 난수 : 39655e62ce596153017dbd491fd98a41fbb0be473c973de089734e9b5823bb12
복호화된 난수 : 5c1668661e226130f4debab6
송신된 메시지 인증 코드 : 61b2f8653bda2d6bec9426862e69e8bab5a92e1d3fa4de970bc59e1bc94992a9
복호화된 메시지 인증 코드 : 61b2f8653bda2d6bec9426862e69e8bab5a92e1d3fa4de970bc59e1bc94992a9
```

```
    let id = new Choice({
      _id: decrypted
    });
    Choice.deletByContent(id, (err, data) =>{
      if(err) {
        return res.json({success : false, msg:'삭제에 실패하였습니다.'});
      } else {
        return res.json({success : true, msg: '삭제되었습니다.'});
      }
});
```

삭제(models.choice.js)

```
module.exports.deletByContent = function(delcontent, callback){
  Choice.deleteOne({_id: delcontent}, callback);
}
```


감사합니다