

코드 해석

```
<body>
  <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-auth.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-firestore.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.6.5/firebase-storage.js"></script>
  <script src="https://code.jquery.com/jquery-3.7.1.min.js" integrity="sha256-/JqT3SQfawRcv/BIHPThkBVvs00EvtFFmqPF/lYI/Cxo=" crossorigin="anonymous"></script>
  <!--파이어 데이터 베이스를 위한 API-->
  <script>
    const firebaseConfig = {
      apiKey: "AIzaSyDuz4A-YN9-hZXemSXJ0MIvADk23zNLnGw",
      authDomain: "jsproject-9ebf7.firebaseio.com",
      databaseURL: "https://jsproject-9ebf7-default-rtdb.firebaseio.com",
      projectId: "jsproject-9ebf7",
      storageBucket: "jsproject-9ebf7.firebaseio.com",
      messagingSenderId: "595770505966",
      appId: "1:595770505966:web:033f9ab267afc4b27991ff"
    };
    // 데이터베이스 연결을 위한 정보
    firebase.initializeApp(firebaseConfig); //파이어 베이스 연결
  </script>
```

데이터베이스 연결 설정
: 모든 코드에 동일하게 적용

코드 해석

1. 회원가입

```
<script>
  //db 객체 생성
  var db = firebase.firestore();

  //유저 카운터 증가 함수(유저 별 식별)
  function getNextuserNumber() {
    return db.runTransaction((transaction) => {
      return transaction.get(db.collection('counterDB').doc('userCounter')).then((counterDoc) => {
        if (!counterDoc.exists) {
          throw "Counter document does not exist!";
        }
        var newUserNumber = counterDoc.data().userNumber + 1;
        transaction.update(db.collection('counterDB').doc('userCounter'), { userNumber: newUserNumber });
        return newUserNumber;
      });
    });
  }
}
```

코드 해석

1. 회원가입

```
//유저 추가 함수
function writeUser(name, id, password) {
  // Firestore에서 userId 중복 확인
  db.collection("user").where("id", "==", id).get()
    .then((querySnapshot) => {
      if (!querySnapshot.empty) {
        // userId가 이미 존재할 경우
        console.error("Error: User ID already exists.");
        alert("이미 사용 중인 ID입니다. 다른 ID를 선택해주세요.");
      } else {
        // userId가 중복되지 않으면 새 유저 추가(유저 번호, 이름, 아이디, 비밀번호)
        getNextuserNumber().then((userNumber) => {
          db.collection("user").add({
            userNumber: userNumber,
            name: name,
            id: id,
            password: password
          })
            .then((docRef) => {
              console.log("Document written with ID: ", docRef.id);
              setTimeout(() => {
                window.location.href = "D:/JavaScript/Source/JS_project/메인 홈페이지.html";
              }, 500);
            })
            .catch((error) => {
              console.error("Error adding document: ", error);
            });
        });
      }
    })
    .catch((error) => {
      console.error("Error checking user ID: ", error);
    });
}
```


코드 해석

1. 회원가입

```
// id가 userInput인 Form에 submit 버튼을 통해 이벤트 핸들러 추가
// submit을 통해 양식 제출
document.getElementById('userInput').addEventListener('submit', function(e) {
  e.preventDefault();

  //username, userId, password라는 아이디인 input 태그의 내용을 입력받아 각각 name, id, password로 저장
  const name = document.getElementById('username').value;
  const id = document.getElementById('userId').value;
  const password = document.getElementById('password').value;

  //writeUser함수에 저장한 name, id, password의 값을 넣어 실행
  writeUser(name, id, password);
});
</script>
```

코드 해석

2. 로그인

```
<script>
// Firestore 객체 생성
var db = firebase.firestore();

// 로그인 기능 함수
function loginUser(id, password) {
  // Firestore에서 입력한 id와 일치하는 사용자 검색
  db.collection("user").where("id", "==", id).get()
    .then((querySnapshot) => {
      if (querySnapshot.empty) {
        // id가 존재하지 않음
        alert("해당 ID를 사용하는 사용자가 없습니다.");
      } else {
        // id가 존재하는 경우
        let isPasswordCorrect = false;
        let userName = "";

        querySnapshot.forEach((doc) => {
          const userData = doc.data();
          //저장된 비밀번호와 입력한 비밀번호가 같을 경우
          if (userData.password === password) {
            isPasswordCorrect = true;
            userName = userData.name;
          }
        });
      }
    });
}
```

코드 해석

2. 로그인

```
        if (isPasswordCorrect) {
            // 로그인 성공
            alert("로그인 성공");
            // 로컬 스토리지 통해 사용자 정보 저장 (로그인 유지 기능)
            localStorage.setItem('loggedInUser', JSON.stringify({ id, name: userName }));
            setTimeout(() => {
                window.location.href = "../메인 홈페이지.html";
            }, 500);
        } else {
            // 비밀번호 불일치
            alert("비밀번호가 올바르지 않습니다.");
        }
    }
})
.catch((error) => {
    console.error("Error logging in: ", error);
    alert("로그인 중 문제가 발생했습니다. 다시 시도해주세요.");
});
}
```


코드 해석

2. 로그인

```
// id가 login인 Form에 submit 버튼을 통해 이벤트 핸들러 추가
// submit을 통해 양식 제출
document.getElementById('login').addEventListener('submit', function (e) {
  e.preventDefault();

  //userId, password라는 아이디인 input 태그의 내용을 입력받아 각각 id, password로 저장
  const id = document.getElementById('userId').value;
  const password = document.getElementById('password').value;

  //loginUser함수에 저장한 id, password의 값을 넣어 실행
  loginUser(id, password);
});
</script>
```

코드 해석

3. 메인 화면

//데이터베이스에서 게시글을 불러와 웹 페이지에 표시함

```
function loadPosts() {
  db.collection("posts").orderBy("createdAt", "desc").get().then((querySnapshot) => {
    querySnapshot.forEach((doc) => {
      var post = doc.data();
      //각 데이터를 표시할 요소를 만들
      var liElement = document.createElement("li");
      liElement.className = "content_2_li";

      var postHeader = document.createElement("div");
      postHeader.className = "post-header";

      var authorElement = document.createElement("b");
      authorElement.className = "author";
      authorElement.textContent = post.name;

      var dateElement = document.createElement("span");
      dateElement.className = "date";
      dateElement.textContent = post.datetime;

      postHeader.appendChild(authorElement);
      postHeader.appendChild(dateElement);
      liElement.appendChild(postHeader);

      var titleElement = document.createElement("a");
      titleElement.className = "post-title";
      //제목 요소에 이동할 링크 설정
      titleElement.href = `./게시글 화면.html?id=${doc.id}&title=${encodeURIComponent(post.title)}
      &name=${encodeURIComponent(post.name)}&school=${encodeURIComponent(post.school)}
      &major=${encodeURIComponent(post.major)}&entrance=${encodeURIComponent(post.entrance)}
      &datetime=${encodeURIComponent(post.datetime)}&content=${encodeURIComponent(post.content)}`;
      titleElement.textContent = post.title;
      liElement.appendChild(titleElement);
    });
  });
}
```

```
//학교가 연암공대일 경우만 게시글을 표시
if (post.school === "연암공대") {
```


코드 해석

3. 메인 화면

```
var contentElement = document.createElement("div");
contentElement.className = "post-content";
if (post.content.length > 100) { //게시글 내용이 100보다 긴 경우
    const truncatedContent = post.content.slice(0, 100) + '...';
    contentElement.innerHTML = truncatedContent + ` <a class="read-more" href="./게시글 화면.html?id=${doc.id}
    &title=${encodeURIComponent(post.title)}&name=${encodeURIComponent(post.name)}
    &school=${encodeURIComponent(post.school)}&major=${encodeURIComponent(post.major)}
    &entrance=${encodeURIComponent(post.entrance)}&datetime=${encodeURIComponent(post.datetime)}
    &content=${encodeURIComponent(post.content)}">더 보기</a>`;
} else { //게시글 내용이 100자 이하인 경우
    contentElement.textContent = post.content;
}
liElement.appendChild(contentElement);

var hrElement = document.createElement("hr");
hrElement.id = "content_2_hr";

document.getElementById("post-list").appendChild(hrElement);
document.getElementById("post-list").appendChild(liElement);
});
}).catch((error) => {
    console.error("Error loading posts: ", error);
});
}
```

코드 해석

4. 게시판 글쓰기 화면

```
<script>
  // Firebase Firestore 초기화
  var db = firebase.firestore();

  // 게시물 카운터 증가 함수 (게시글 별 식별)
  function getNextpostNumber() {
    return db.runTransaction((transaction) => {
      return transaction.get(db.collection('counterDB').doc('postCounter')).then((counterDoc) => {
        if (!counterDoc.exists) {
          throw "Counter document does not exist!";
        }
        var newpostNumber = counterDoc.data().postNumber + 1;
        transaction.update(db.collection('counterDB').doc('postCounter'), { postNumber: newpostNumber });
        return newpostNumber;
      });
    });
  }
}
```

코드 해석

4. 게시판 글쓰기 화면

```
// 게시물 작성 함수
function writePost(title, content, name, school, major, entrance, datetime) {
  getNextpostNumber().then((postNumber) => {
    //posts 라는 테이블에 아래 정보 추가
    db.collection("posts").add({
      postNumber: postNumber,
      title: title,
      content: content,
      name: name,
      school: school,
      major: major,
      entrance: entrance,
      datetime: datetime,
      createdAt: firebase.firestore.FieldValue.serverTimestamp()
    })
    .then((docRef) => {
      console.log("Document written with ID: ", docRef.id);
      // 게시물 작성 후 메인 홈페이지로 리다이렉션
      window.location.href = "메인 홈페이지.html"; // 메인 홈페이지로 이동
    })
    .catch((error) => {
      console.error("Error adding document: ", error);
    });
  }).catch((error) => {
    console.error("Transaction failed: ", error);
  });
}
```


코드 해석

4. 게시판 글쓰기 화면

```
// id가 content_1 Form에 submit 버튼을 통해 이벤트 핸들러 추가
// submit을 통해 양식 제출
document.getElementById('content_1').addEventListener('submit', function(e) {
    e.preventDefault();

    // 각 필드 값 가져오기
    const title = document.getElementById('title').value;
    const content = document.getElementById('content').value;
    const isAnonymous = document.querySelector('input[name="name"]:checked');
    const school = document.querySelector('input[name="school"]:checked');
    const major = document.querySelector('select[name="major"]').value;
    const entrance = document.querySelector('select[name="entrance"]').value;
    const datetime = document.querySelector('select[name="datetime-local"]').value;

    // 필수 항목 검사
    if (!title || !content || !isAnonymous || !school || !major || !entrance || !datetime) {
        alert("모든 필드를 채워주세요!");
        return; // 필드가 비어 있으면 제출하지 않음
    }

    // 로그인된 사용자 이름 처리 (로컬 스토리지 이용해서 사용자 정보 불러옴)
    const loggedInUser = JSON.parse(localStorage.getItem('loggedInUser'));
    // 게시글 입력시 익명이면 익명 출력, 비익명일 시 로그인 된 사용자의 이름 출력
    const name = isAnonymous.value === '익명' ? '익명' : (loggedInUser ? loggedInUser.name : '익명');

    // 게시글 작성 함수 호출
    writePost(title, content, name, school.value, major, entrance, datetime);
});
```

코드 해석

4. 게시판 글쓰기 화면

```
// 로그인 상태 확인 함수
function checkLoginStatus() {
  const loggedInUser = JSON.parse(localStorage.getItem('loggedInUser'));
  if (loggedInUser) {
    document.getElementById('first_h1').textContent = loggedInUser.name;
    const authButtons = document.getElementById('auth-buttons');
    authButtons.innerHTML = ''; // 기존 로그인/회원가입 버튼 제거
    const logoutButton = document.createElement('button');
    logoutButton.className = 'sign-button';
    logoutButton.id = 'logout-button';
    logoutButton.textContent = '로그아웃';
    authButtons.appendChild(logoutButton);

    logoutButton.addEventListener('click', function () {
      localStorage.removeItem('loggedInUser');
      window.location.reload();
    });
  } else {
    document.getElementById('first_h1').textContent = "(User Name)";
  }
}

//로그인 검사
document.addEventListener("DOMContentLoaded", function() {
  checkLoginStatus();
});
</script>
```


코드 해석

5. 게시물 화면

```
//게시글에 관한 데이터를 가져오는 함수
function getPostDataFromUrl() {
  const urlParams = new URLSearchParams(window.location.search);
  return {
    id: urlParams.get('id'),
    title: decodeURIComponent(urlParams.get('title')),
    name: decodeURIComponent(urlParams.get('name')),
    school: decodeURIComponent(urlParams.get('school')),
    major: decodeURIComponent(urlParams.get('major')),
    entrance: decodeURIComponent(urlParams.get('entrance')),
    datetime: decodeURIComponent(urlParams.get('datetime')),
    content: decodeURIComponent(urlParams.get('content'))
  };
}
```


코드 해석

5. 게시물 화면

```
// 게시물 데이터를 가져와 화면에 나타내는 함수
function loadPost() {
    const postData = getPostDataFromUrl();
    document.getElementById('post-title').textContent = postData.title;
    document.getElementById('post-author').textContent = postData.name;
    document.getElementById('post-school').textContent = postData.school;
    document.getElementById('post-major').textContent = postData.major;
    document.getElementById('post-entrance').textContent = postData.entrance;
    document.getElementById('post-datetime').textContent = postData.datetime;
    document.getElementById('post-content').textContent = postData.content;
    // 게시물 내용 표시 추가
}
```

코드 해석

6. 게시물에 댓글 작성

```
//댓글 작성 폼과 텍스트 영역 요소 가져옴
const commentForm = document.getElementById('comment-form');
const commentText = document.getElementById('comment-text');

//댓글 작성 이벤트 리스너 추가
commentForm.addEventListener('submit', function(event) {
  event.preventDefault();
  const text = commentText.value.trim();

  if (text) {
    //firestore에 댓글 데이터 추가
    db.collection('comments').add({
      text: text,
      author: JSON.parse(localStorage.getItem('loggedInUser')).id,
      // 작성자 ID (로컬 스토리지에서 가져옴)
      postId: getPostDataFromUrl().id,
      timestamp: firebase.firestore.FieldValue.serverTimestamp()
    }).then(() => {
      commentText.value = '';
    }).catch(error => {
      console.error("댓글 작성 오류:", error);
    });
  }
});
```

코드 해석

6. 게시물에 댓글 작성

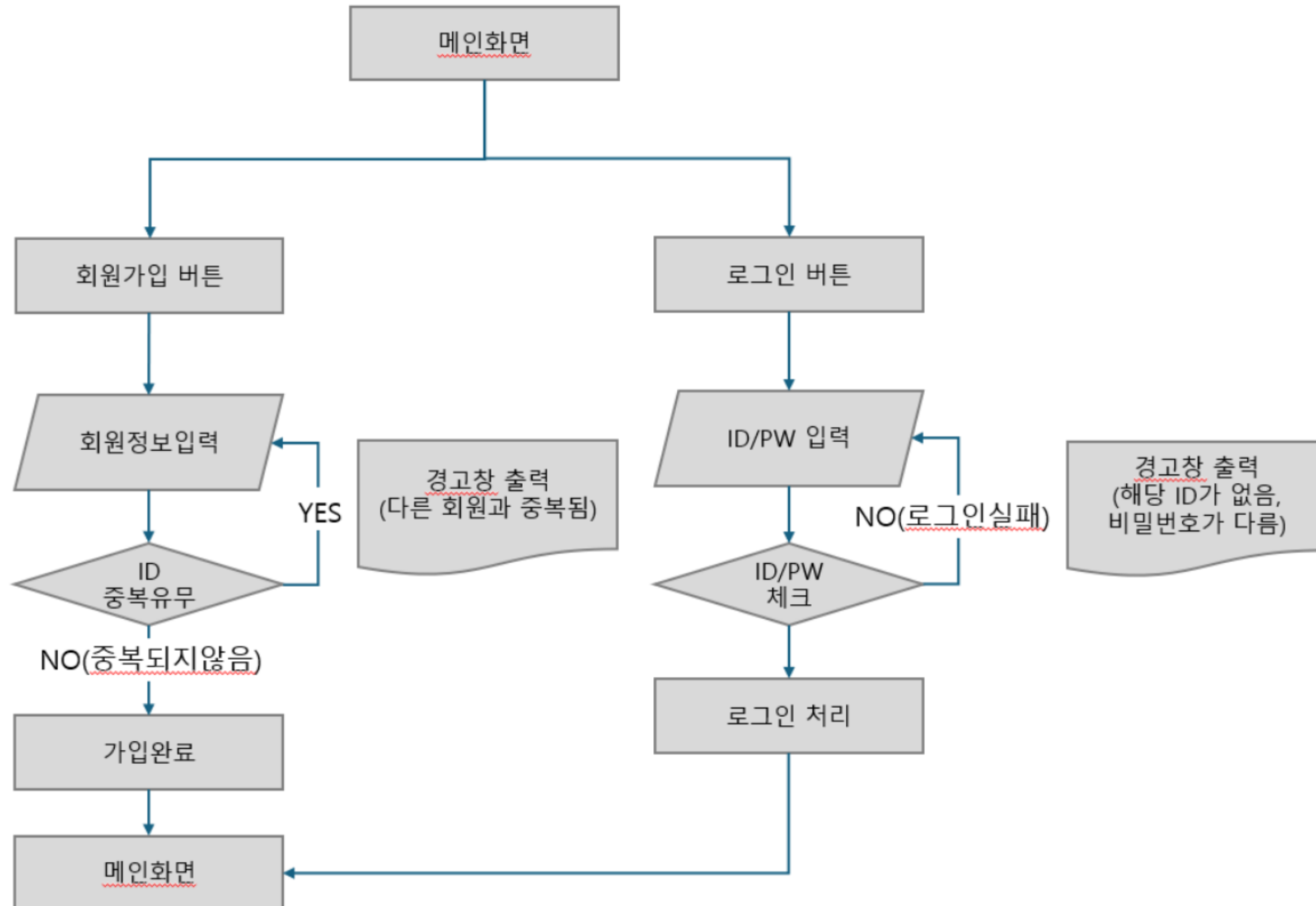
```
//댓글 목록 요소 가져옴
const commentsList = document.getElementById('comments-list');

//실시간으로 데이터 가져옴
db.collection('comments').orderBy('timestamp', 'desc').onSnapshot(function(snapshot) {
  commentsList.innerHTML = '';
  snapshot.forEach(doc => {
    const comment = doc.data(); //댓글 데이터 가져오기
    const commentElement = document.createElement('div');
    commentElement.className = 'comment';
    commentElement.innerHTML = `
      <strong>${comment.author}</strong>: ${comment.text}
    `;
    commentsList.appendChild(commentElement);
  });
});

//실시간으로 데이터 가져옴
db.collection('comments').orderBy('timestamp', 'asc').onSnapshot((querySnapshot) => {
  commentsList.innerHTML = '';
  querySnapshot.forEach((doc) => {
    const comment = doc.data();
    const divElement = document.createElement('div');
    if(comment.postId == getPostDataFromUrl().id) { //게시글 id와 일치하는 댓글만 표시
      divElement.classList.add('comment');
      divElement.textContent = `${comment.author}: ${comment.text}`; //댓글 내용 형식
    }
    commentsList.appendChild(divElement);
  });
});
```

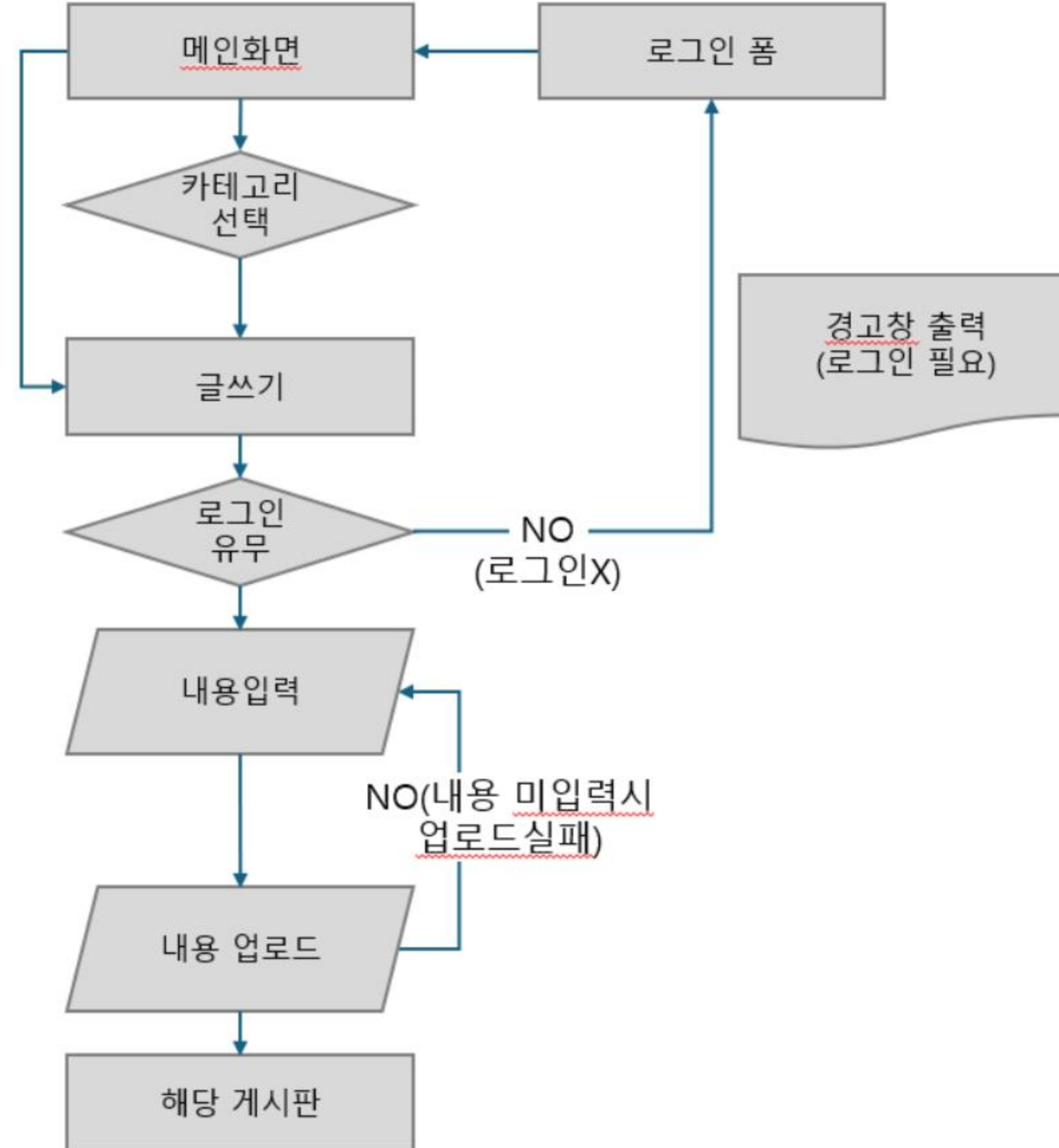

플로우 차트

<회원가입 · 로그인>



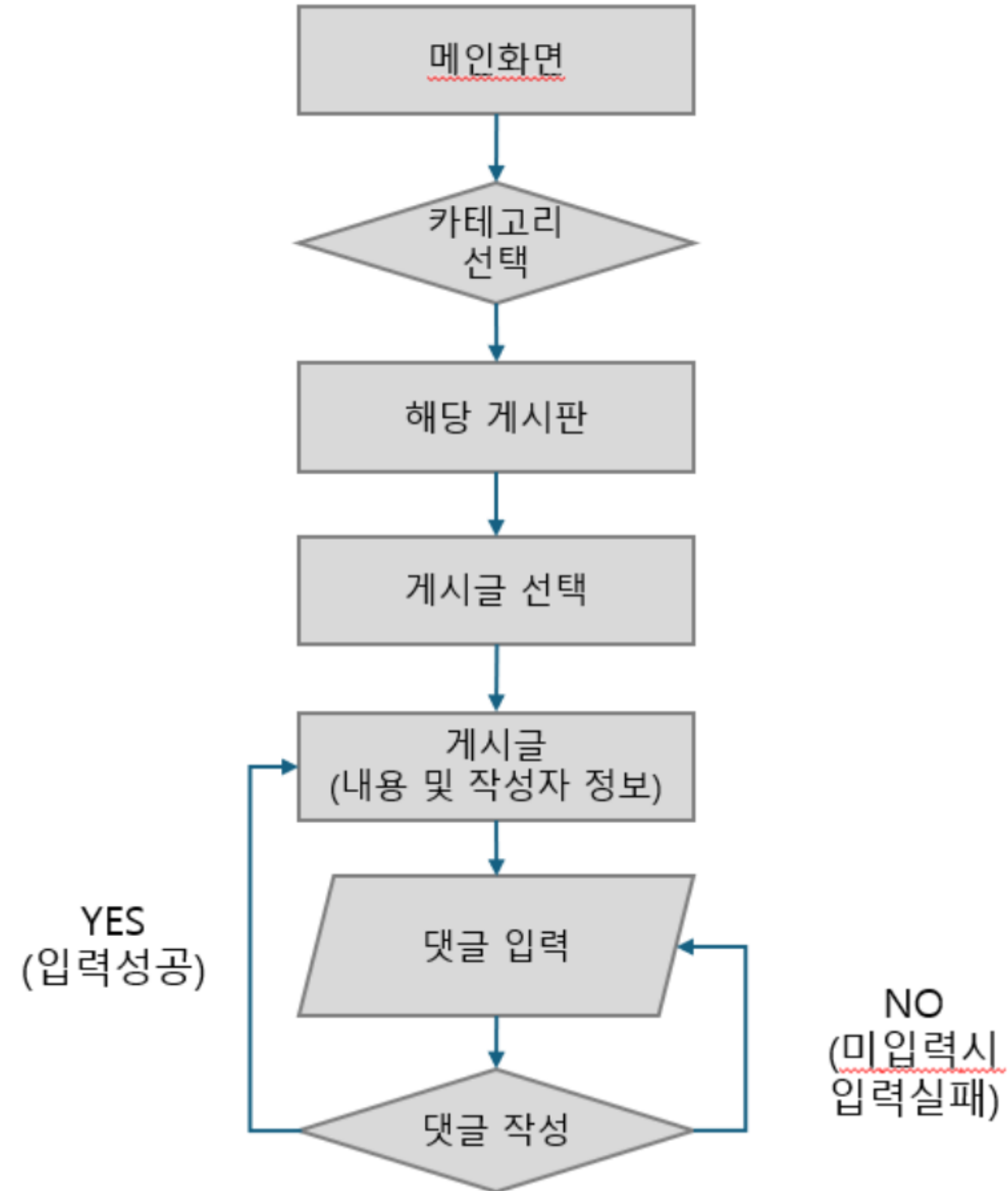
플로우 차트

<게시글 쓰기>



플로우 차트

<게시판 찾기 및 댓글 입력>



결과



(User Name)

Table Talk 로그인

Table Talk 회원가입

Category

학급

- 연암공과대학교
- 국립경상대학교

학과별

- 기계공학과
- 전기전자공학과
- 스마트 기계공학과
- 스마트 전기전자공학과
- 스마트 소프트웨어학과

학도

- 20학번
- 21학번
- 22학번
- 23학번
- 24학번
- 25학번

날짜/시간별

- 평일 점심
- 평일 저녁
- 주말 점심
- 주말 저녁

Table Talk

전체 게시판

글쓰기

test5

경상대 / 기계공학과 / 21학번 / 평일 저녁

경상대 / 기계공학과 / 21학번 / 평일 저녁

평일 저녁

testUser2

연암공대 / 스마트 소프트웨어학과 / 25학번 / 평일 저녁

연암공대 / 스마트 소프트웨어학과 / 25학번 / 평일 저녁

평일 저녁

응답

연암공대 / 스마트 소프트웨어학과 / 24학번 / 평일 점심

연암공대 / 스마트 소프트웨어학과 / 24학번 / 평일 점심

평일 점심

의무

연암공대 / 스마트 전기전자공학과 / 23학번 / 주말 저녁

연암공대 / 스마트 전기전자공학과 / 23학번 / 주말 저녁

주말 저녁

의명

연암공대 / 스마트 기계공학과 / 22학번 / 주말 점심

연암공대 / 스마트 기계공학과 / 22학번 / 주말 점심

주말 점심

testUser1

연암공대 / 전기전자공학과 / 21학번 / 평일저녁

연암공대 / 전기전자공학과 / 21학번 / 평일저녁

평일 저녁

응답

연아고대 / 기계고하과 / 20하하 / 페이지 1

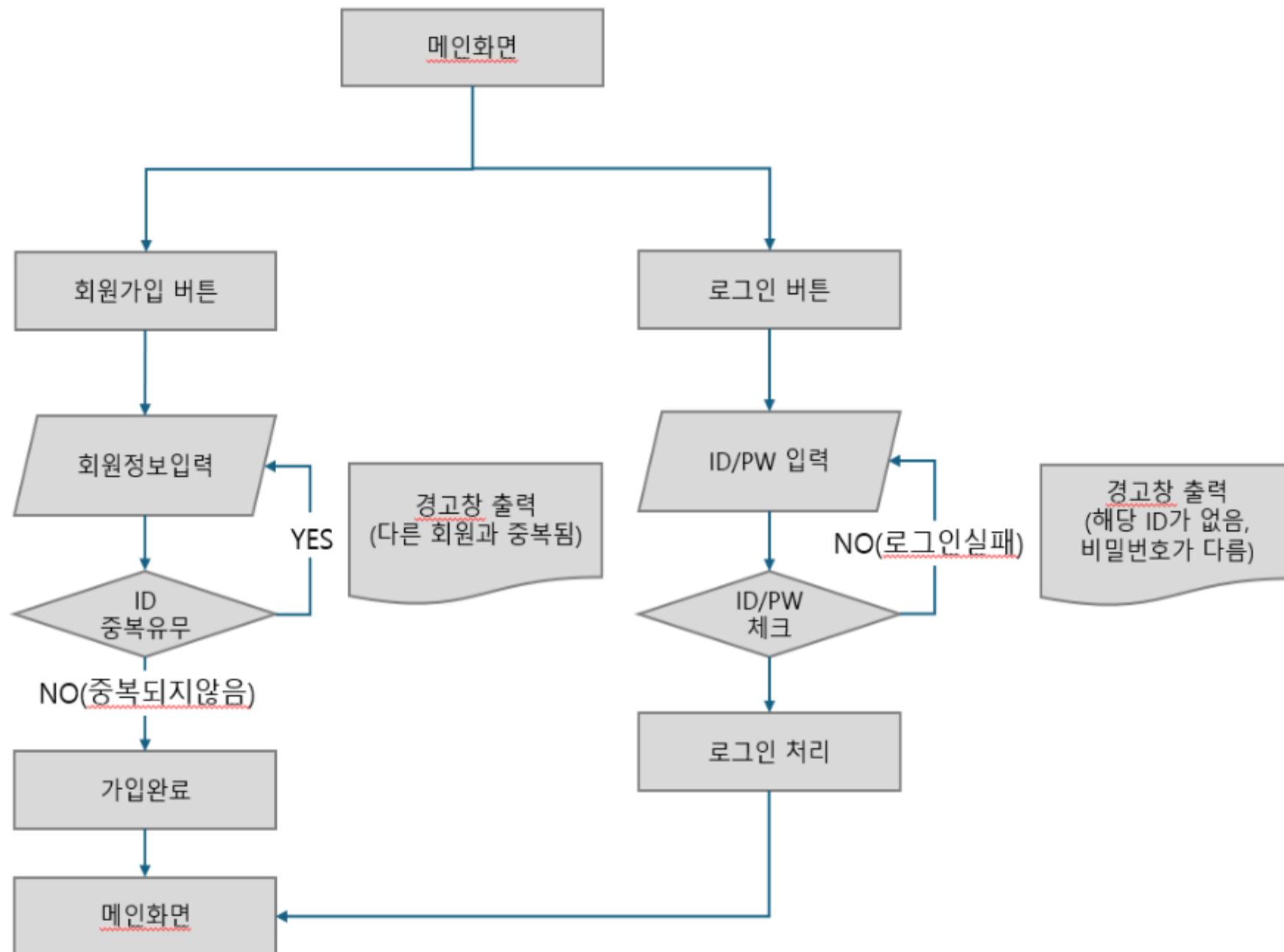
평일 점심

메인 화면

결과

1. 회원가입 및 로그인

<회원가입 · 로그인>



(User Name)

Table Talk 로그인

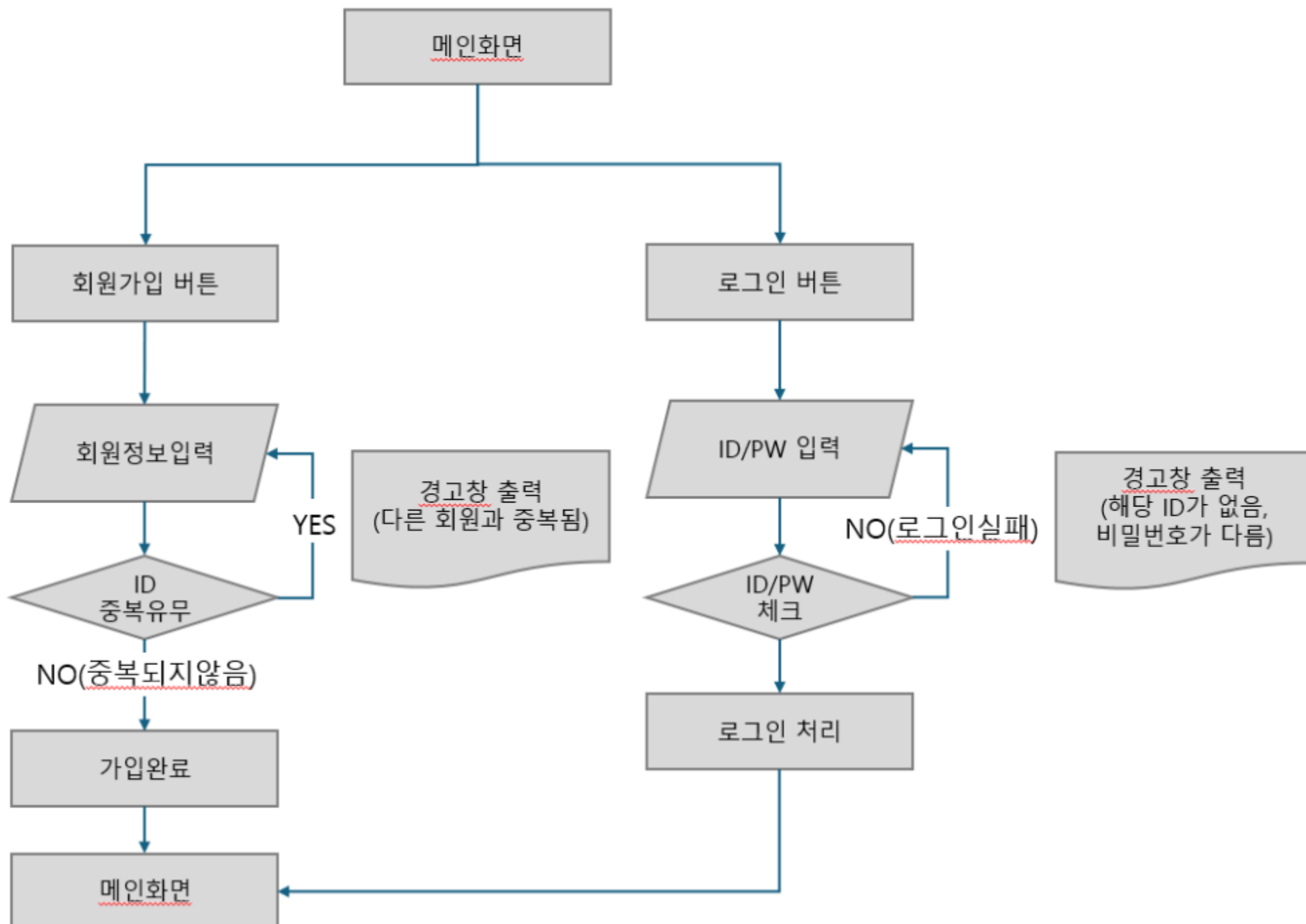
Table Talk 회원가입

회원가입

결과

1. 회원가입 및 로그인

<회원가입 · 로그인>



Sign up

Enter Your Name

StartCode

Create A ID

StartCode

Create A Password

.....



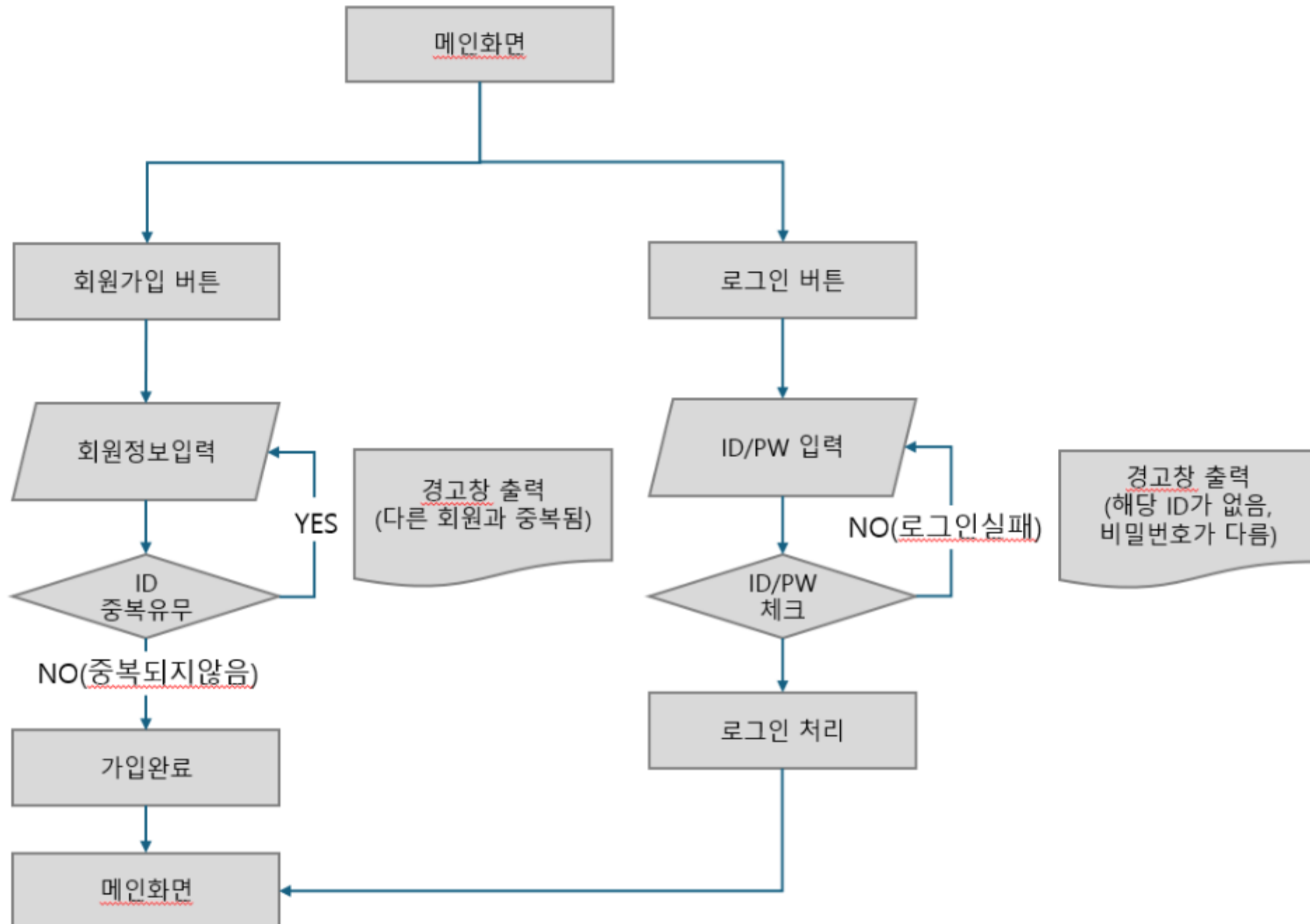
Sign up

회원가입

결과

1. 회원가입 및 로그인

<회원가입 · 로그인>



(User Name)



Table Talk 로그인

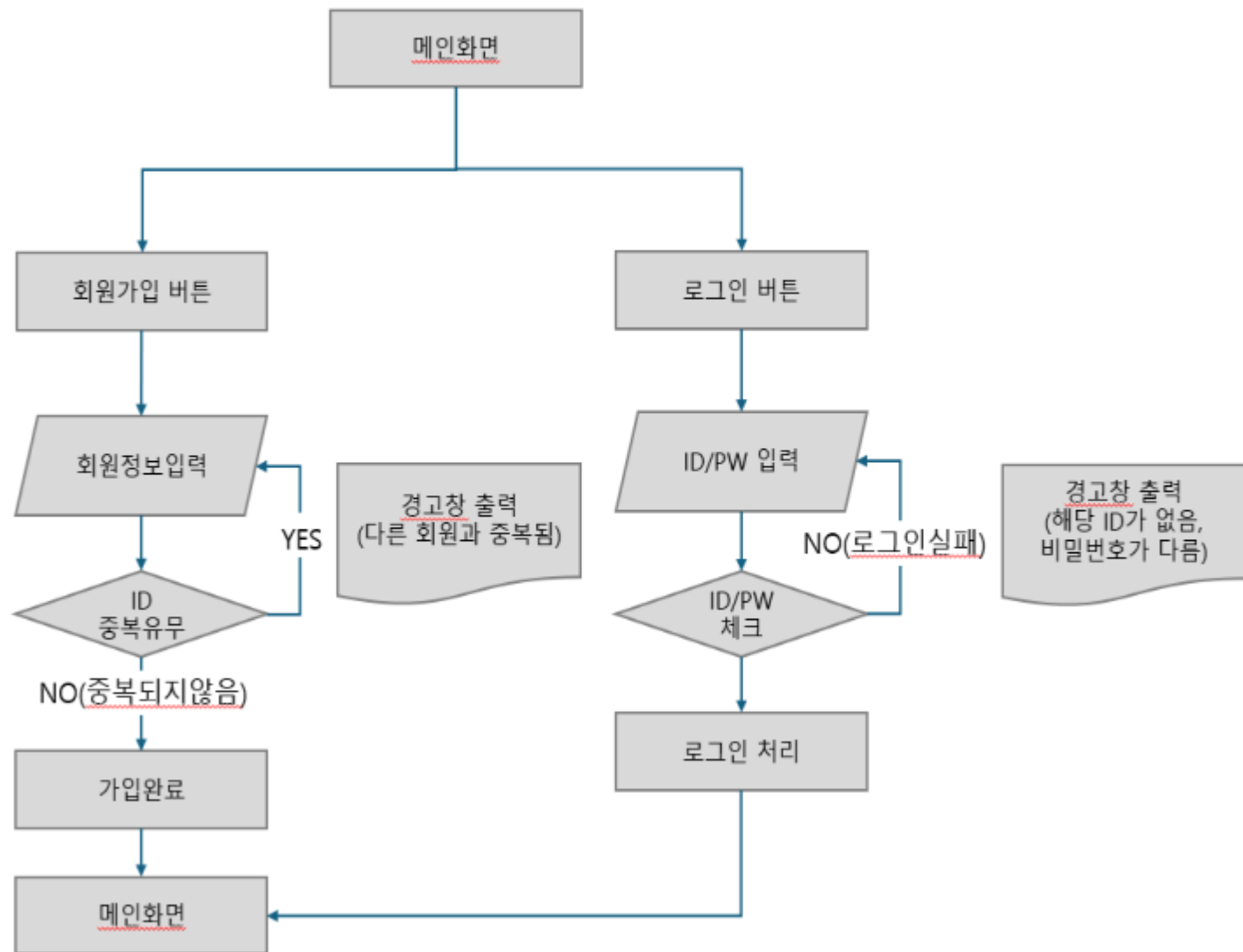
Table Talk 회원가입

로그인

결과

1. 회원가입 및 로그인

<회원가입 · 로그인>



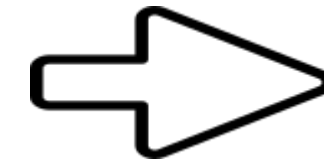
Sign in

User ID

StartCode

Password

.....



로그인 성공

확인



Sign in

로그인

결과

로그인 동작 데이터베이스

결과

로그인 후 메인화면



StartCode

로그아웃

Category

학교별

- 연암공과대학교
- 국립경상대학교

학과별

- 기계공학과
- 전기전자공학과
- 스마트 기계공학과
- 스마트 전기전자공학과
- 스마트 소프트웨어학과

학번별

- 20학번
- 21학번
- 22학번
- 23학번
- 24학번
- 25학번

날짜/시간별

- 평일 점심
- 평일 저녁
- 주말 점심
- 주말 저녁

Table Talk

전체 게시판

글쓰기

test5

경상대 / 기계공학과 / 21학번 / 평일 저녁

경상대 / 기계공학과 / 21학번 / 평일 저녁

평일 저녁

testUser2

연암공대 / 스마트 소프트웨어학과 / 25학번 / 평일 저녁

연암공대 / 스마트 소프트웨어학과 / 25학번 / 평일 저녁

평일 저녁

익명

연암공대 / 스마트 소프트웨어학과 / 24학번 / 평일 점심

연암공대 / 스마트 소프트웨어학과 / 24학번 / 평일 점심

평일 점심

익명

연암공대 / 스마트 전기전자공학과 / 23학번 / 주말 저녁

연암공대 / 스마트 전기전자공학과 / 23학번 / 주말 저녁

주말 저녁

익명

연암공대 / 스마트 기계공학과 / 22학번 / 주말 점심

연암공대 / 스마트 기계공학과 / 22학번 / 주말 점심

주말 점심

testUser1

연암공대 / 전기전자공학과 / 21학번 / 평일저녁

연암공대 / 전기전자공학과 / 21학번 / 평일저녁

평일 저녁

익명

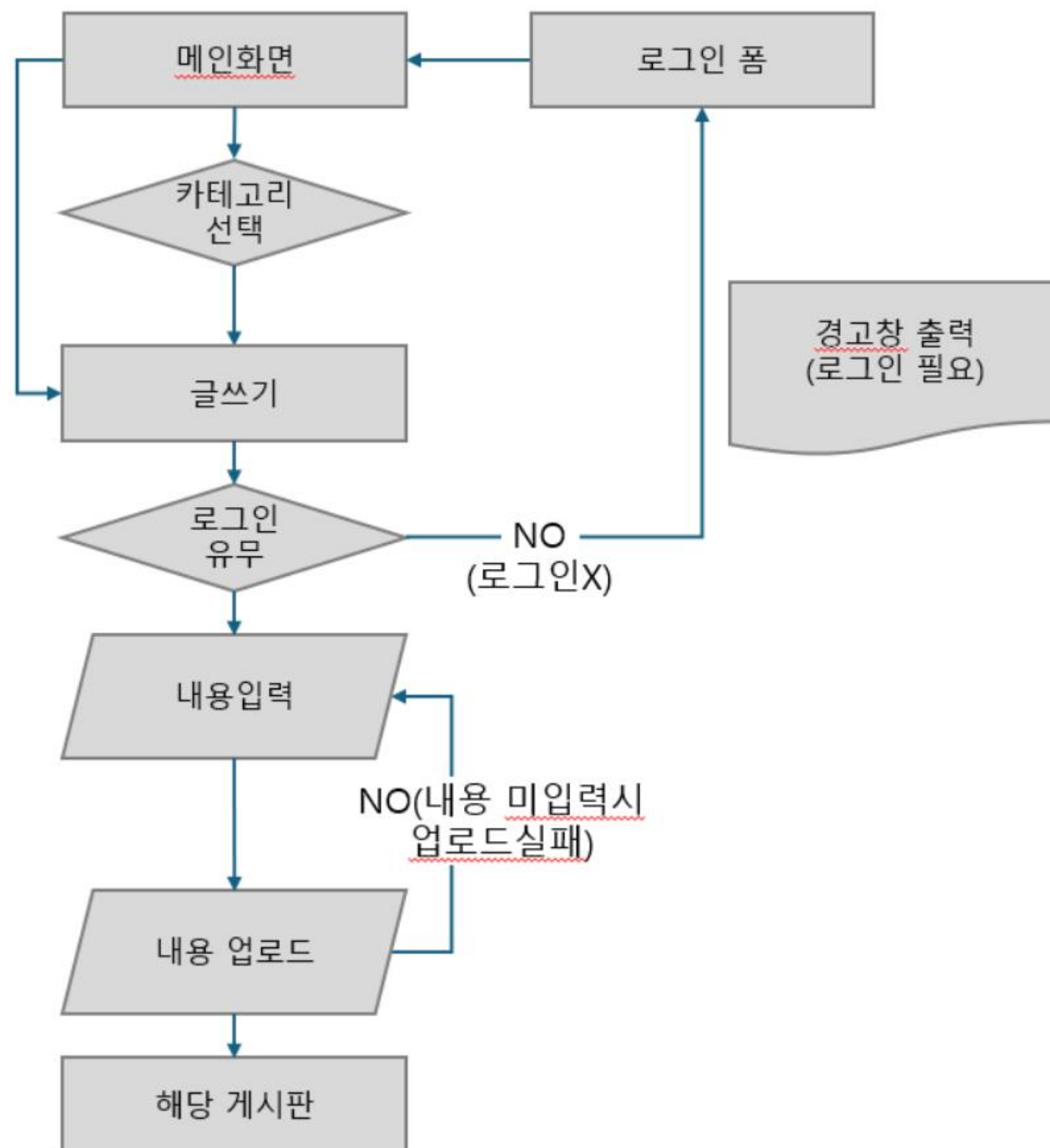
연암공대 / 기계공학과 / 22학번 / 평일점심

평일 점심

플로우 차트

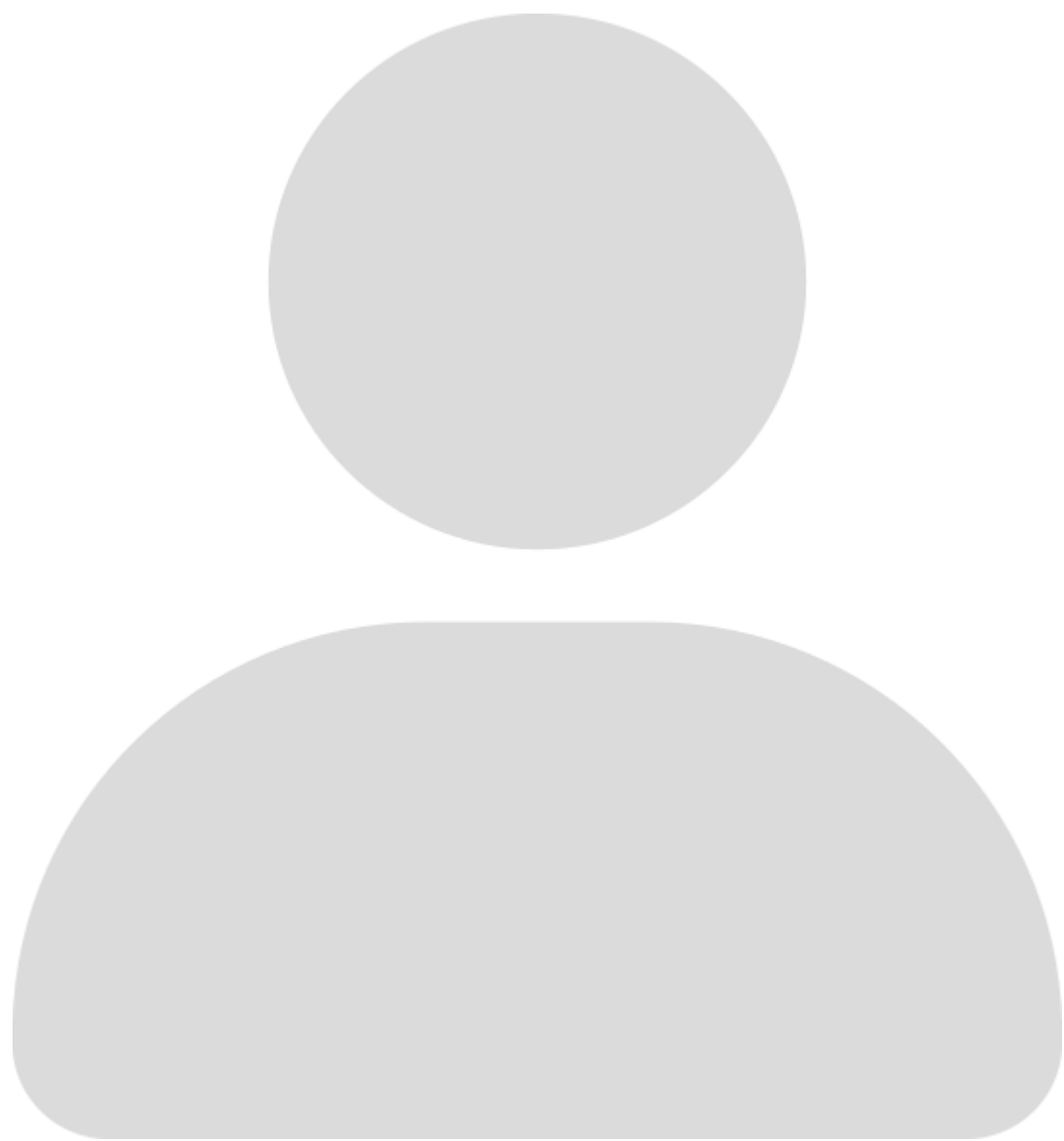
<게시글 쓰기>

2. 게시물 쓰기



플로우 차트

2. 게시물 쓰기



이름 : StartCode

학과 : 스마트소프트웨어학과

학번 : 24학번

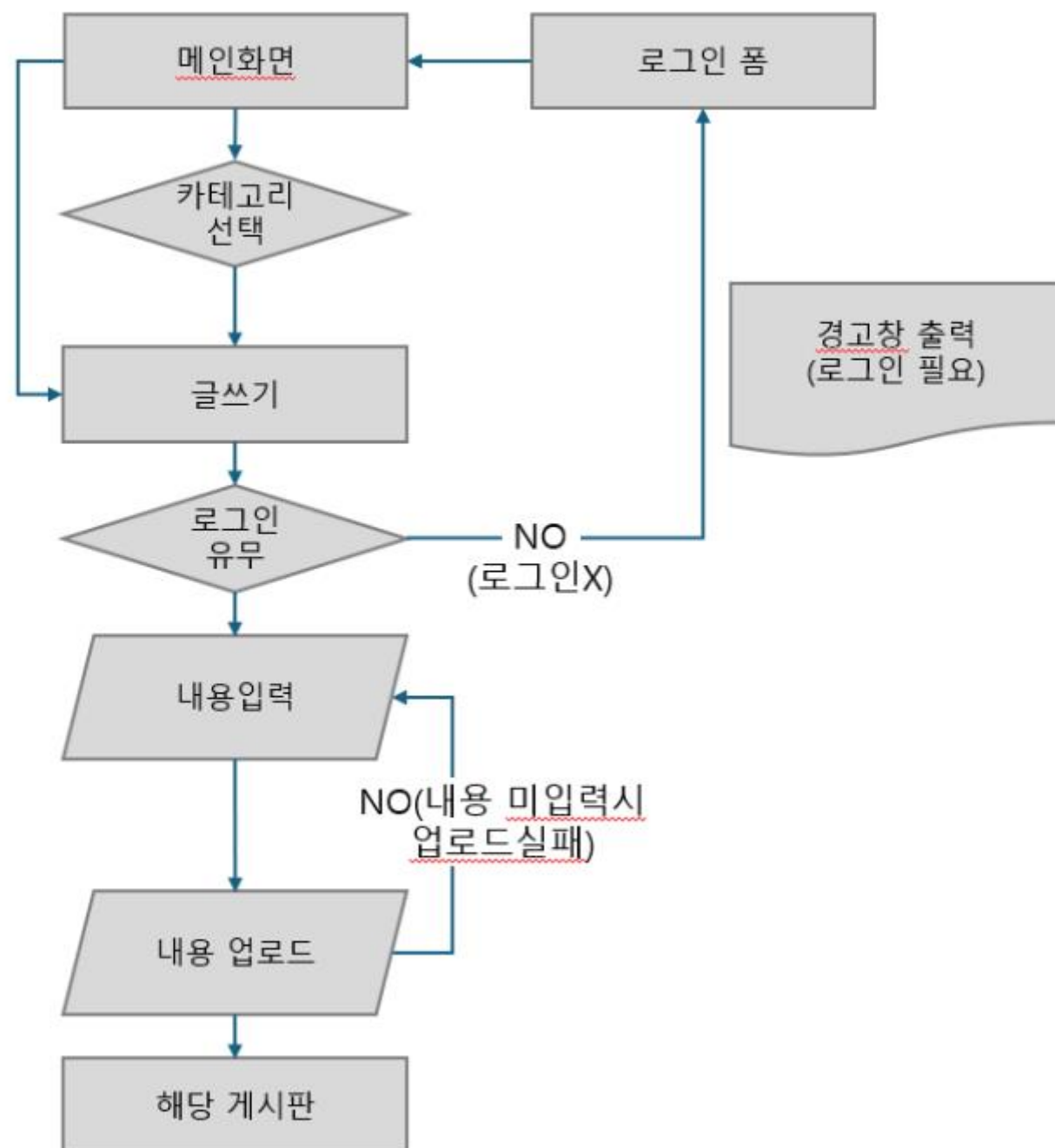
희망 날짜 : 평일 점심

이 사용자가 게시글을 올리는 과정은 ?

결과

2. 게시물 쓰기

<게시글 쓰기>



글쓰기 선택

Table Talk

✓ 글쓰기

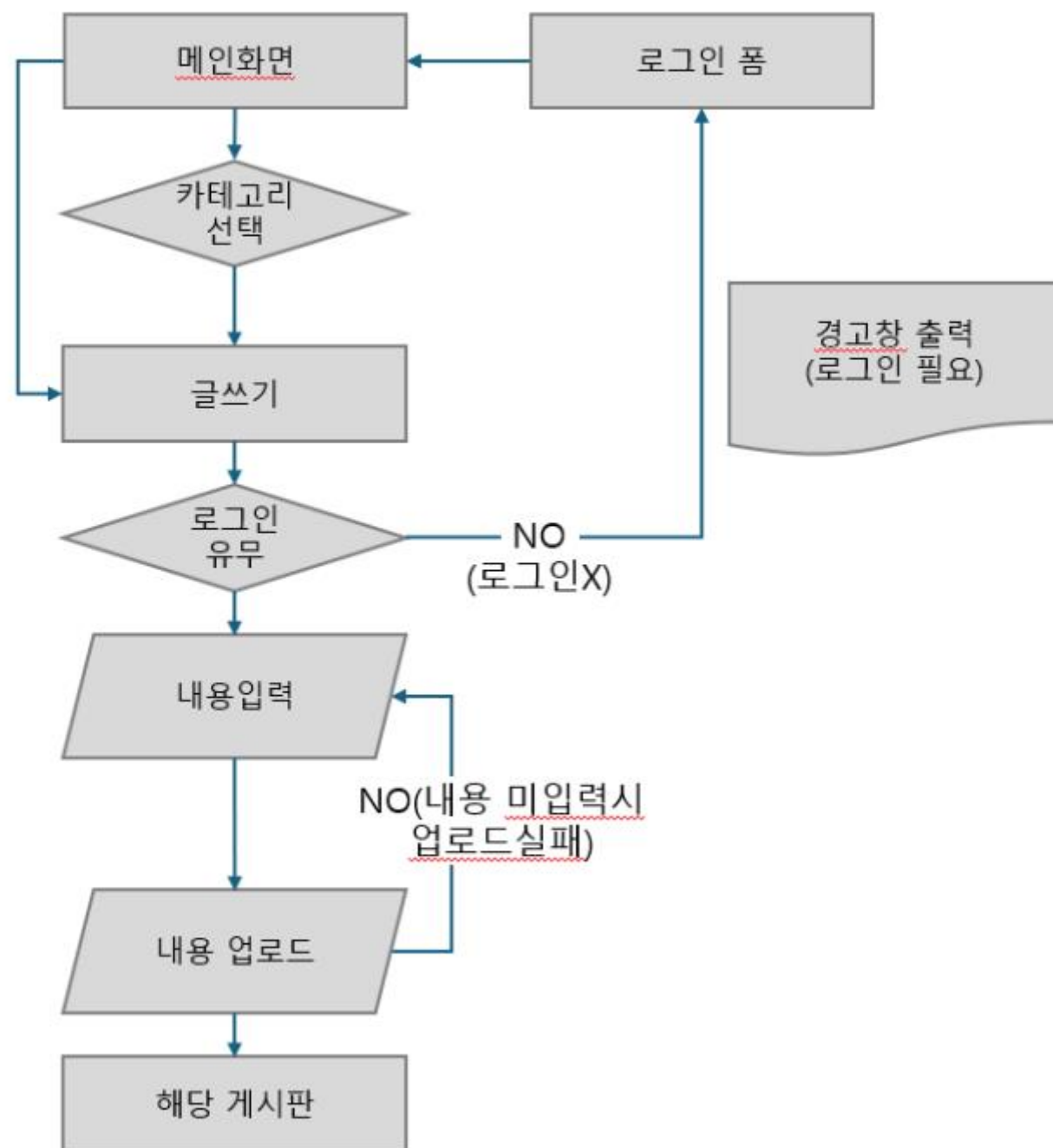
평일 저녁

평일 점심

결과

2. 게시물 쓰기

<게시글 쓰기>



게시글 작성

금요일 12시 점심 같이 먹으실 분 있나요 ~?

제목 작성

익명 / 비익명 : ☐ 익명 ☒ 비익명 익명 유무 선택

학교 : ☒ 연암공대 ☐ 경상대 학교 선택

학과

학번

약속 날짜 및 시간

금요일 12시에 점심식사 하실 분 구합니다.
댓글 남겨주세요 ~ ^^

게시글 내용

결과

2. 게시물 쓰기

스마트 소프트웨어학과 ▼

기계공학과

전기전자공학과

스마트 기계공학과

스마트 전기전자공학과

✓ 스마트 소프트웨어학과

학과 선택

24학번 ▼

20학번

21학번

22학번

23학번

✓ 24학번

25학번

학번 선택

평일 점심 ▼

✓ 평일 점심

평일 저녁

주말 점심

주말 저녁

약속 날짜 / 시간 선택

결과

2. 게시물 쓰기

업로드 클릭 !

게시글 작성



금요일 12시 점심 같이 먹으실 분 있나요 ~?

익명 / 비익명 : ☐ 익명 ☒ 비익명

학교 : ☒ 연암공대 ☐ 경상대










학과 ▼

학번 ▼

약속 날짜 및 시간 ▼

금요일 12시에 점심식사 하실 분 구합니다.
댓글 남겨주세요 ~ ^^

결과

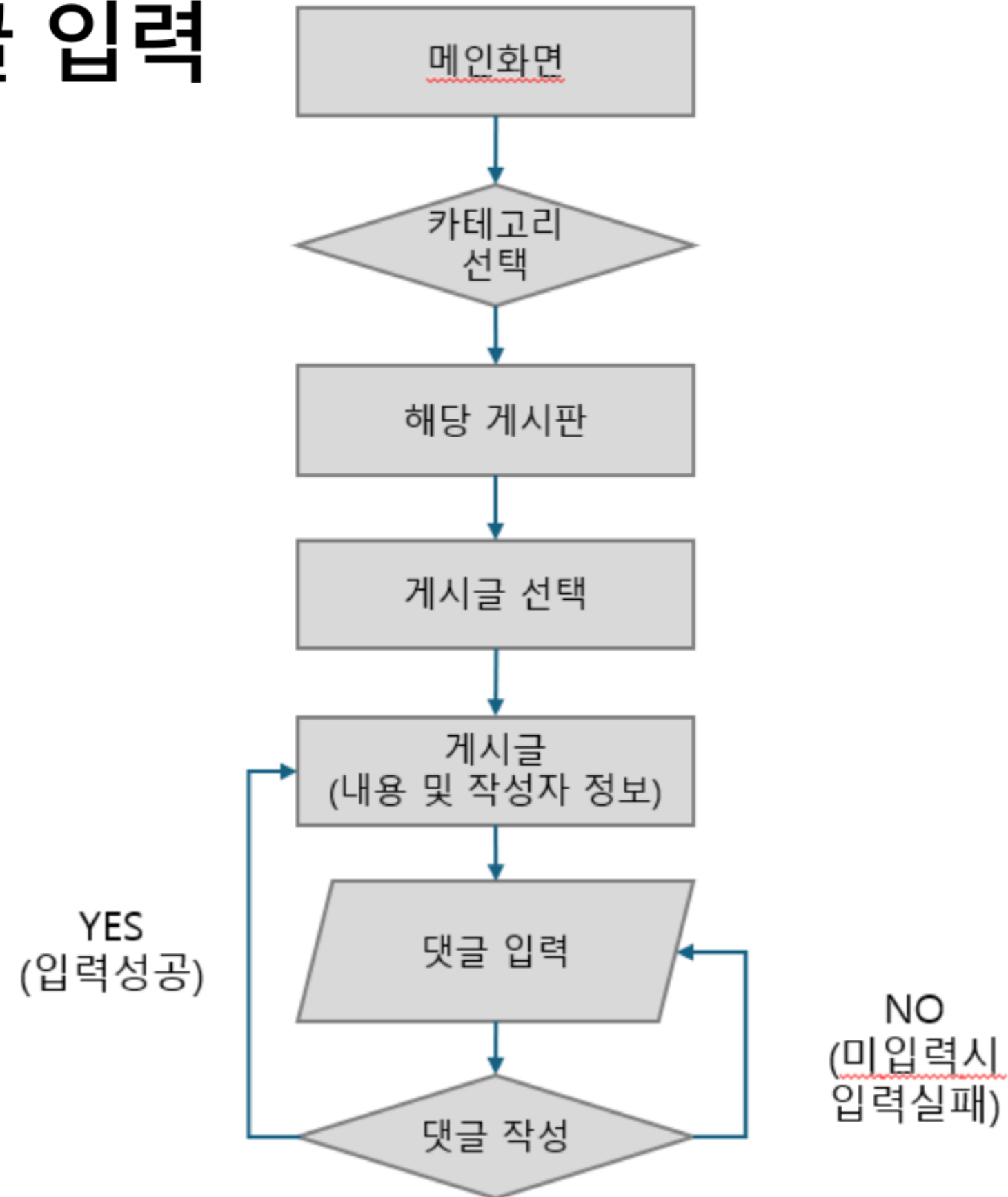
<div><div> > posts > kOIP0br3iJApLS.</div><div>Google Cloud의 추가 기능 </div></div>		
<div><div> (default)</div></div>	<div><div> posts  </div></div>	<div><div> kOIP0br3iJApLSmxsJ68 </div></div>
<div><div>+ 컬렉션 시작</div><div>comments</div><div>counterDB</div><div>posts ></div><div>user</div></div>	<div><div>+ 문서 추가</div><div>Cef4h5RI1juCJ7CMjILJ</div><div>EjPk2h41hFuhm573icvR</div><div>GGCFXISGeB4XX7b9Txjs</div><div>Uih6ohzoPIM8Wb0AC0xR</div><div>VjXNonC66AoN1cH0BCB7</div><div>aZPTxLzhFLqIv6TaYyCy</div><div>h9irEQx8tZFWKc4b1Dan</div><div>hqsFHkTILyn1bZK0PlmF</div><div><div> kOIP0br3iJApLSmxsJ68 ></div><div>1XgEkBKfoKkIgM2N8rNT</div></div></div>	<div><div>+ 컬렉션 시작</div><div>+ 필드 추가</div><div>content: "금요일 12시에 점심식사 하실 분 구합니다. 댓글 남겨주세요 ~ ^^"</div><div>createdAt: 2024년 12월 2일 오전 3시 58분 44초 UTC+9</div><div>datetime: "평일 점심"</div><div>entrance: "24학번"</div><div>major: "스마트 소프트웨어학과"</div><div>name: "StartCode"</div><div>postNumber: 10</div><div>school: "연암공대"</div><div>title: "금요일 12시 점심 같이 먹으실 분 있나요 ~?"</div></div>

게시글 데이터베이스

플로우 차트

<게시판 찾기 및 댓글 입력>

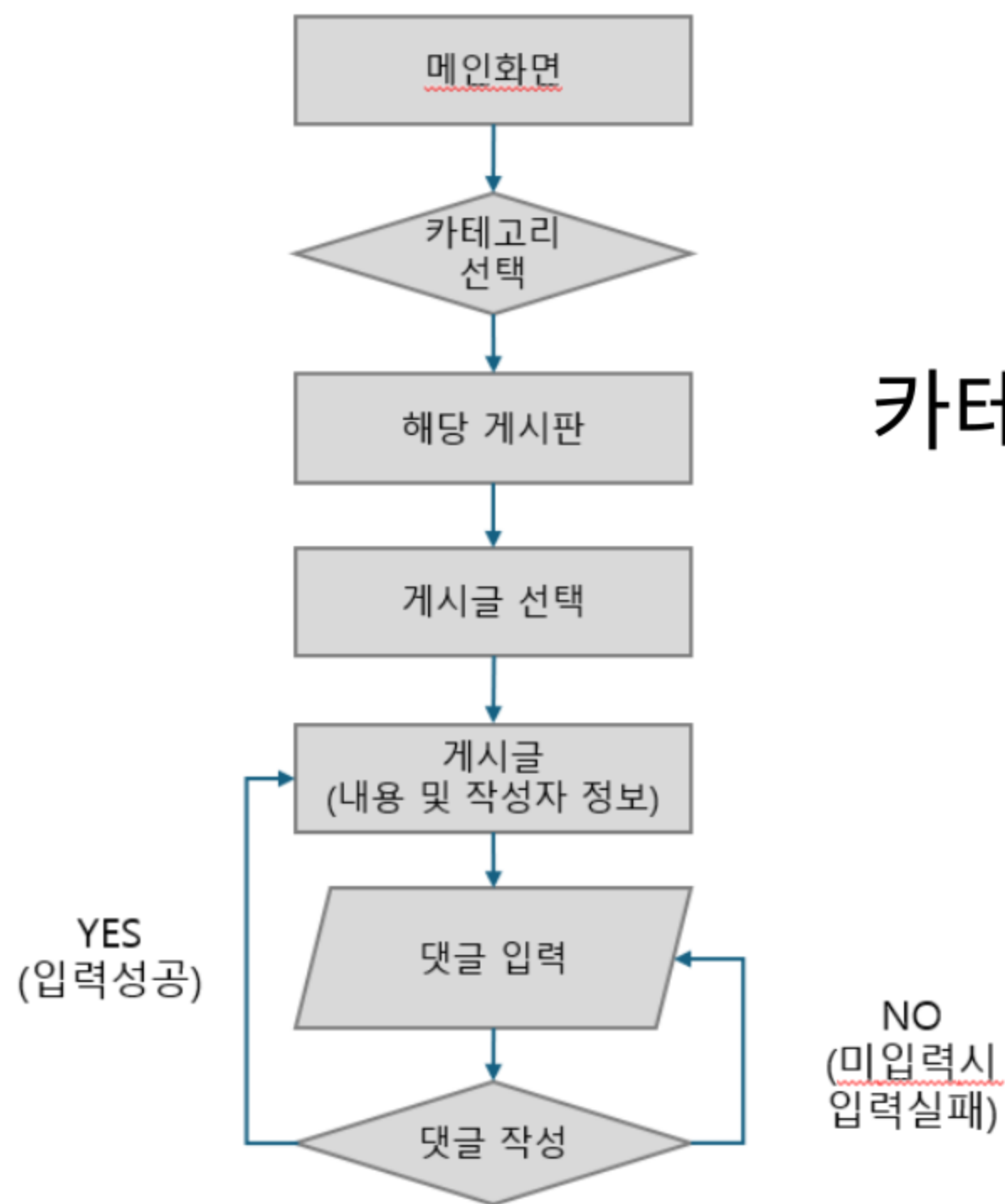
3.게시판 찾기 및 댓글 입력



결과

3.게시판 찾기 및 댓글 입력

<게시판 찾기 및 댓글 입력>



카테고리 선택

학교별
학과별
학번별
날짜/시간별

StartCode

로그아웃

Category

학교별

✓ 연암공과대학교
- 국립경상대학교

학과별

- 기계공학과
- 전기전자공학과
- 스마트 기계공학과
- 스마트 전기전자공학과
✓ 스마트 소프트웨어학과

학번별

- 20학번
- 21학번
- 22학번
- 23학번
✓ 24학번
- 25학번

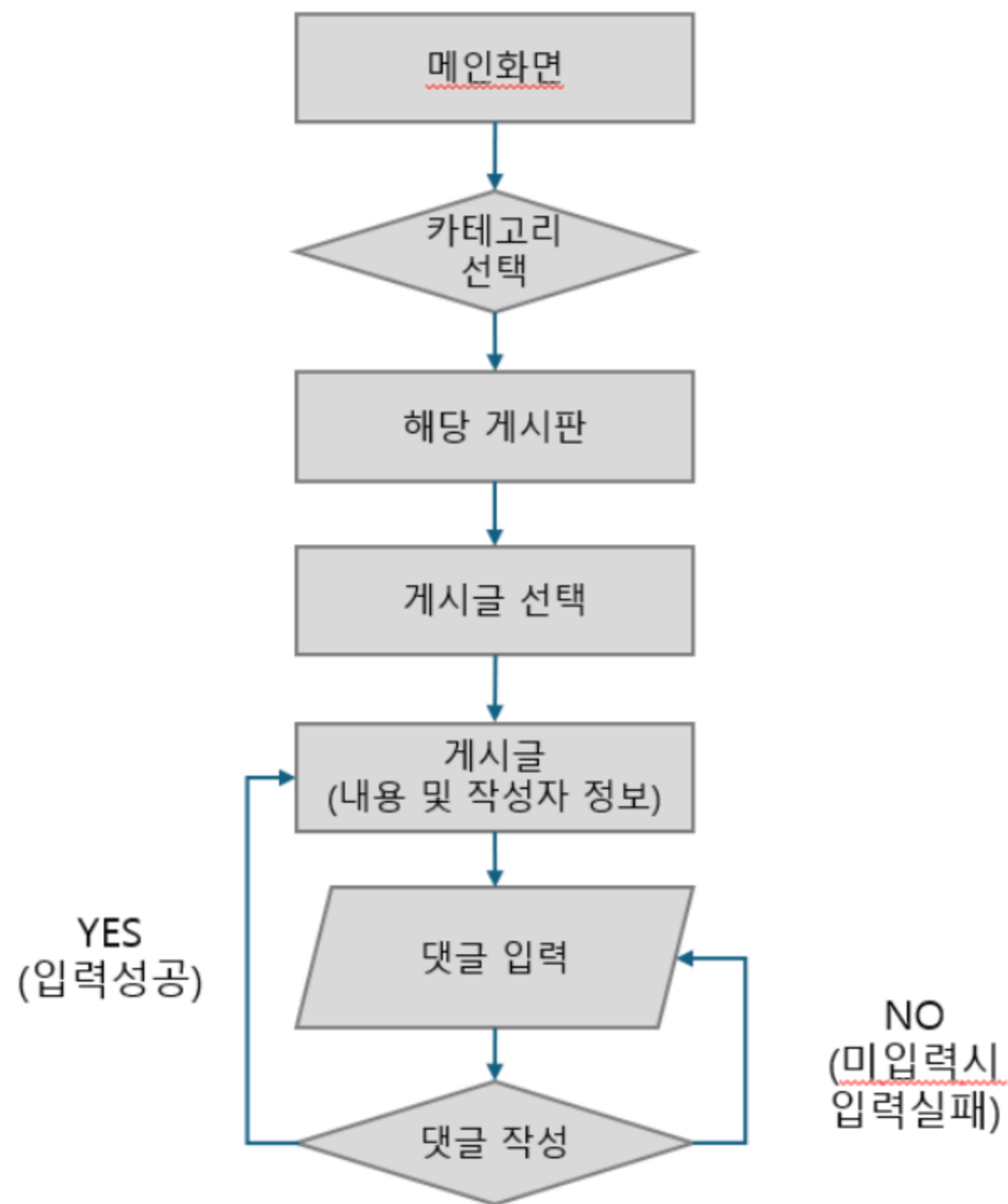
날짜/시간별

✓ 평일 점심
- 평일 저녁
- 주말 점심
- 주말 저녁

결과

3.게시판 찾기 및 댓글 입력

<게시판 찾기 및 댓글 입력>



스마트 소프트웨어학과 게시판



StartCode

금요일 12시 점심 같이 먹으실 분 있나요 ~?

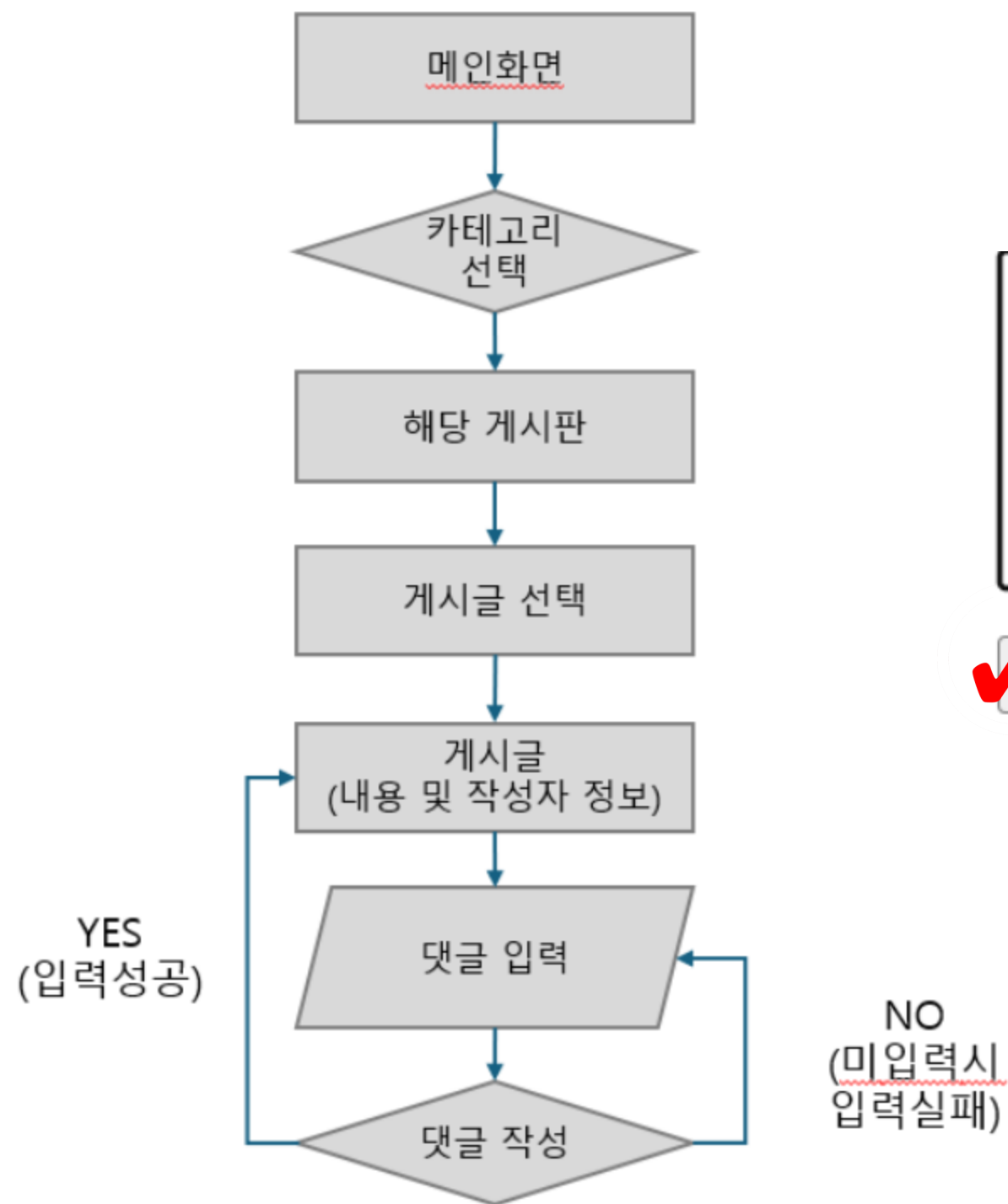
금요일 12시에 점심식사 하실 분 구합니다. 댓글 남겨주세요 ~ ^^

게시글 확인

결과

3.게시판 찾기 및 댓글 입력

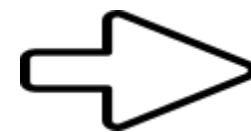
<게시판 찾기 및 댓글 입력>



12시에 시간 가능해요! 저랑 먹어요~

✓ 댓글 작성

댓글 작성



댓글










Meow: 12시에 시간 가능해요! 저랑 먹어요~

댓글을 작성하세요

댓글 작성

댓글 확인

결과

<div><div> > comments > uw8pcC56DFW...</div><div>Google Cloud의 추가 기능 </div></div>		
<div><div> (default)</div></div>	<div><div> comments  </div></div>	<div><div> uw8pcC56DFWaNpPMuP5X </div></div>
<div>+ 컬렉션 시작</div>	<div>+ 문서 추가</div>	<div>+ 컬렉션 시작</div>
<div>comments ></div>	<div>01DaRzbUqYvB3511uLs3</div>	<div>+ 필드 추가</div>
<div>counterDB</div>	<div>ZjivMetFgM4WnZ5sFaFv</div>	<div>author: "Meow"</div>
<div>posts</div>	<div>pWoofYWzg44G3fayQEb8</div>	<div>postId: "kOIP0br3iJApLSmxsJ68"</div>
<div>user</div>	<div><div> uw8pcC56DFWaNpPMuP5X ></div></div>	<div>text: "12시에 시간 가능해요! 저랑 먹어요~"</div>
	<div>vFJi1W1JBDD2gWaR4C1k</div>	<div>timestamp: 2024년 12월 2일 오전 4시 5분 4초 UTC+9</div>

댓글 데이터베이스