TEAM E : CIRCLESPACE
MIDTERM PRESENTATION

날다람쥐랩

SEOKHYUN BAE
JIWOO CHUNG
GUHO KIM
JINU PARK

OBJECTIVE

GOAL 1

SYSTEMATIC CLASSIFICATION , CATEGORIZED INFORMATION

GOAL 2

STANDARDIZED CLUB INFORMATION

GOAL 3

REAL-TIME UPDATES

GOAL 4

UNIVERSITY ACCOUNT VERIFICATION

GOAL 5

NOTIFICATION FEATURE

MEET THE TEAM

SEOKHYUN BAE

TEAM LEADER
FRONTEND & BACKEND
DEPLOYMENT

GUHO KIM

BACKEND
DATABASE
WEB CRAWLING

JINWOO PARK

UI / UX DEGIGN
FRONTEND

JIWOO CHUNG

UI / UX DESIGN
FRONTEND

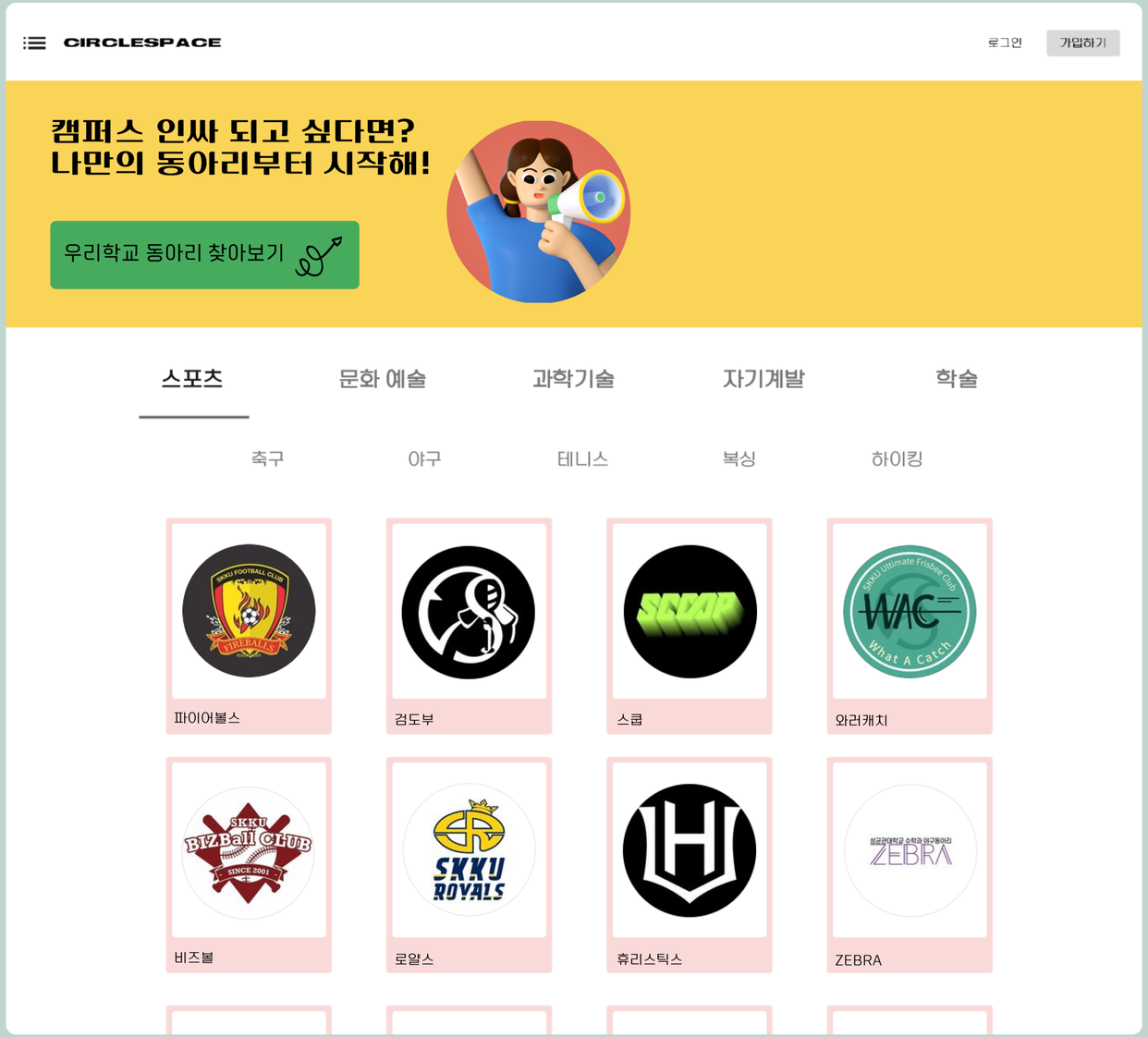
TIMELINE

TASK & ASSIGNE	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13
PROJECT SETTING ALL MEMBERS	■	■							
UI/UX DESIGN FRONTEND DEV.		■	■						
BUSINESS LOGIC DEVELOPMENT BACKEND DEV.		■	■						
WEB PAGE DEVELOPMENT FRONTEND DEV.				■	■	■			
API DEVELOPMENT BACKEND DEV.				■	■	■			
DATASOURCE DEVELOPMENT BACKEND DEV.						■	■		
TESTING & OPTIMIZATION ALL MEMBERS							■	■	
DEPLOYMENT BACKEND DEV.								■	■

- Basic UI / UX design : Done
- Business logic design : Done
- Web Page Development : In Progress
- API Development : In Progress
- Datasource Development : In Progress

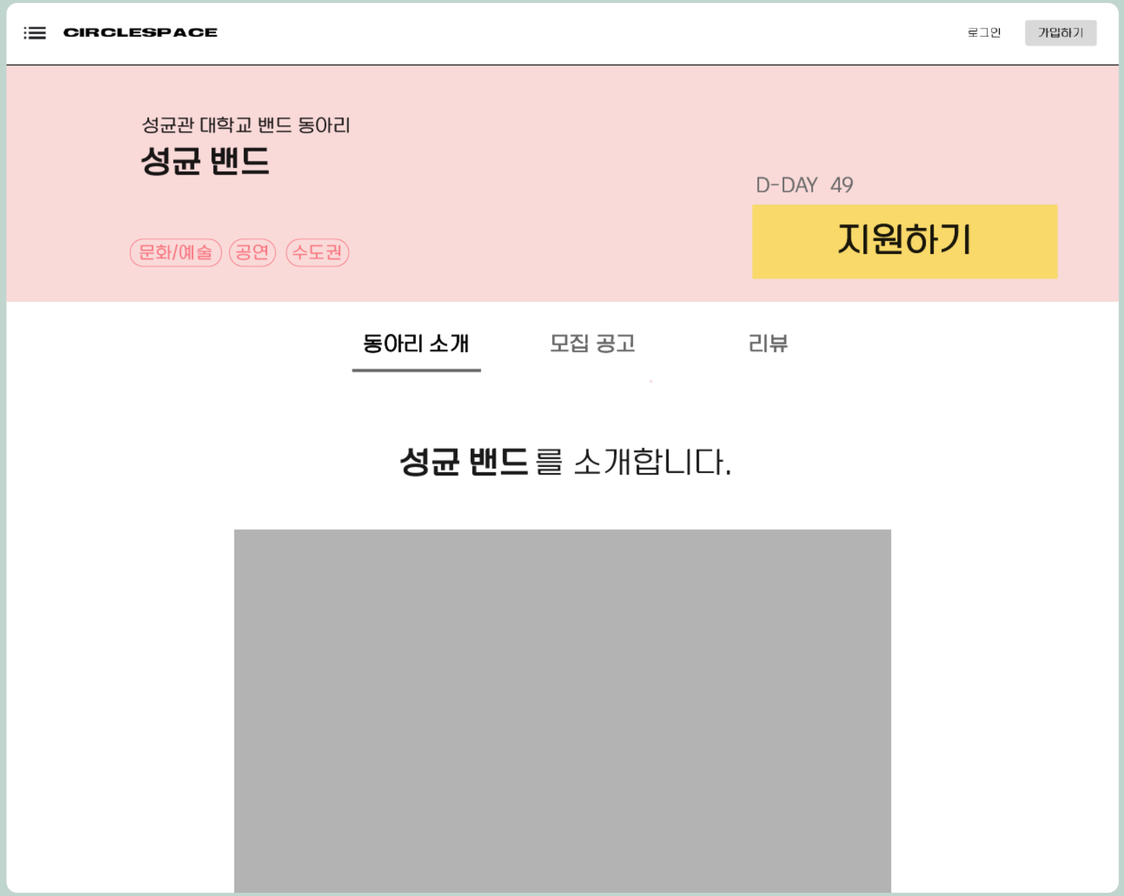
IMPLEMENTATION

CATEGORIZATION



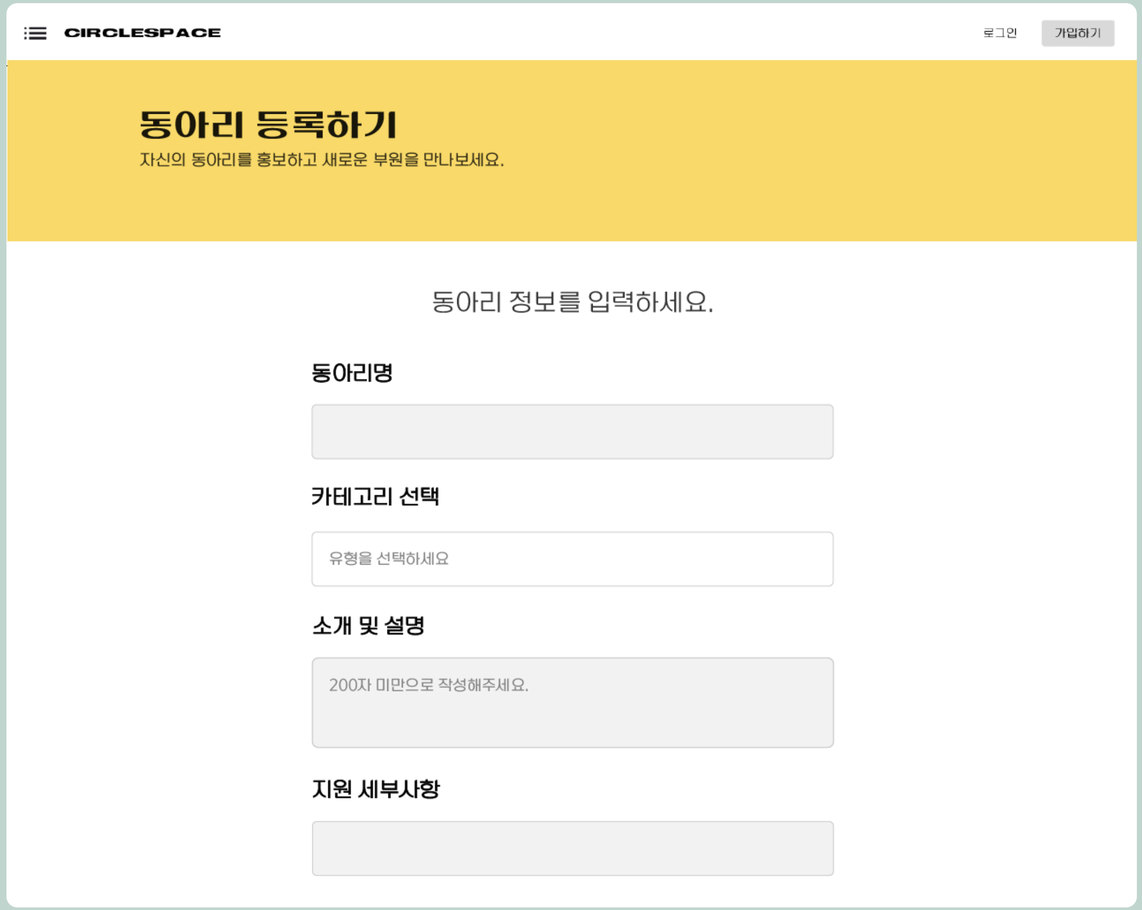

```java
public ResponseEntity<Void> deleteCategory(Long id) {
  try {
    Category category = categoryRepository.findById(id)
        .orElseThrow(() -> new IllegalArgumentException("카테고리를

    for (Club club : category.getClubs()) {
      club.getCategories().remove(category);
      clubRepository.save(club);
    }

    categoryRepository.delete(category);
    return ResponseEntity.noContent().build();
  } catch (Exception e) {
    throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_
  }
}

public ResponseEntity<?> getClubsByCategory(String category) {
  try {
    Category cat = categoryRepository.findByName(category)
        .orElseThrow(() -> new IllegalArgumentException("카테고리를

    Set<Club> clubs = cat.getClubs();
    return ResponseEntity.status(HttpStatus.OK).body(clubs.strea
        .map(clubService::setClubDTO)
        .collect(Collectors.toList()));
  } catch (Exception e) {
    return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERRO
  }
}
```

CLUB DETAILS



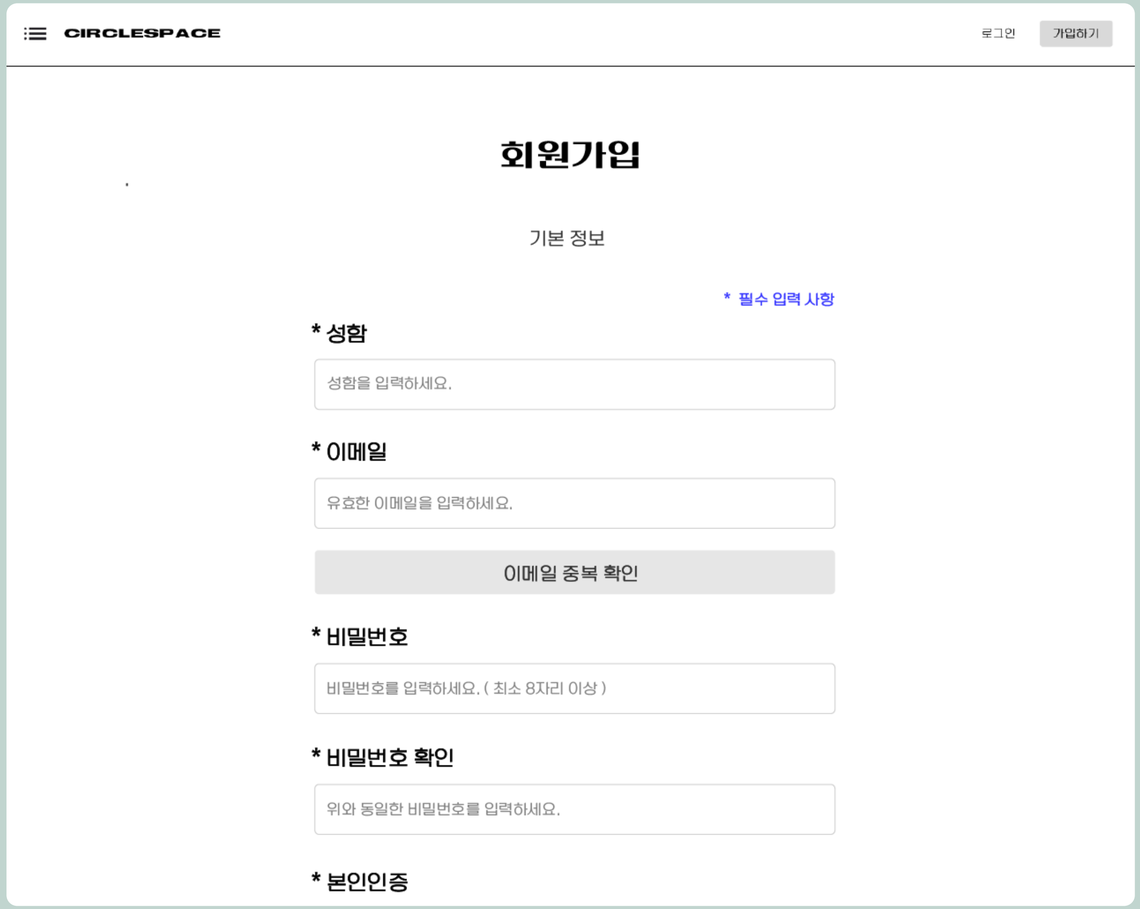

```java
@Entity
@Getter
@Setter
@ToString
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
public class Club {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  @Column(unique = true, nullable=false)
  private String title;

  private Long managerId;
  private String description;

  private String imageUrl;

  private Boolean featured = false;

  @ElementCollection
  @CollectionTable(name = "club_detail_image", joinColumns = @JoinColumn(name = "club_id"))
  private List<DetailImage> detailImages = new ArrayList<>();

  @CreationTimestamp
  @Column(updatable = false)
  private LocalDateTime createdAt;

  @UpdateTimestamp
  private LocalDateTime updatedAt;

  @ManyToMany(mappedBy = "clubs")
  @JsonBackReference
```

CLUB REGISTRATION



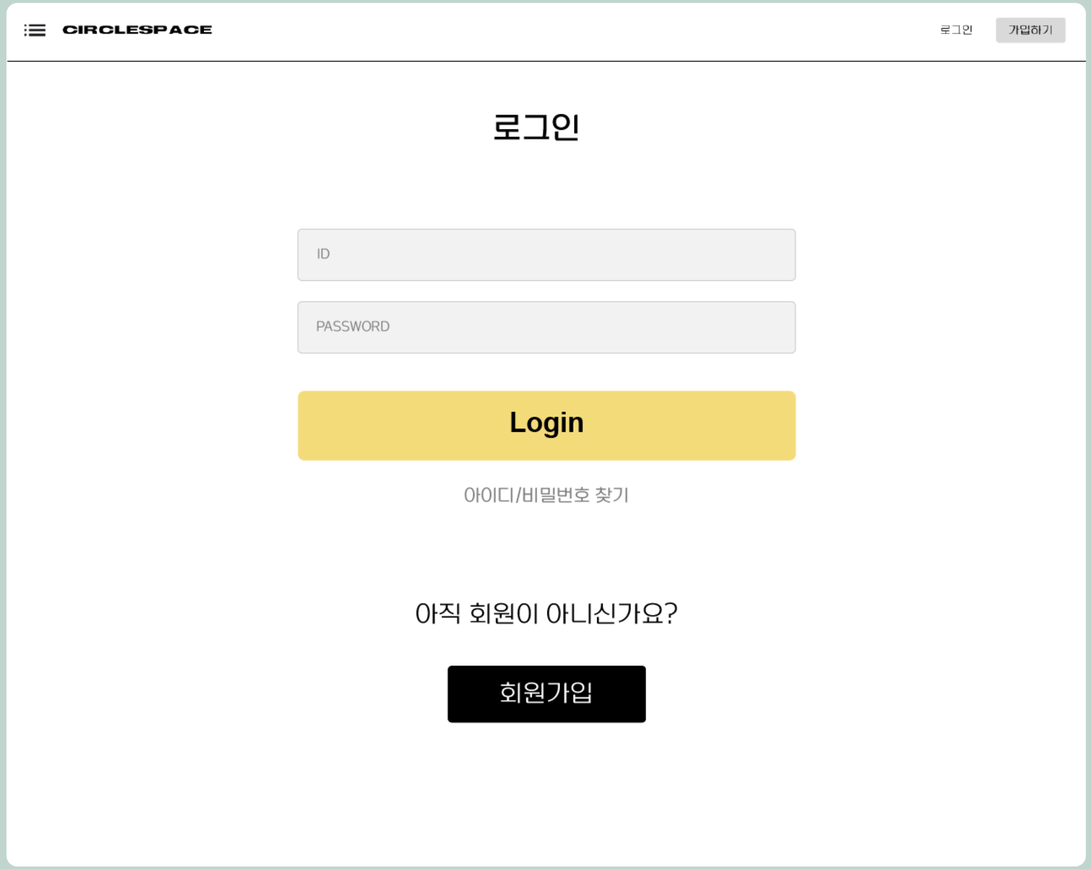

```java
public ResponseEntity<String> uploadClub(Map<String, Object> request) {
  try {
    Club club = new Club();
    String title = (String) request.get("title");
    String imageUrl = s3Service.getObjectUrl((String) request.get("imagePath"));
    String description = (String) request.get("description");
    List<String> categoryNames = (List<String>) request.get("categoryNames");

    club.setTitle(title);
    club.setImageUrl(imageUrl);
    club.setDescription(description);

    List<String> detailImagePaths = (List<String>) request.get("detailImagePaths");
    List<DetailImage> detailImages = detailImagePaths.stream()
        .map(path -> {
          DetailImage detailImage = new DetailImage();
          detailImage.setImageUrl(s3Service.getObjectUrl(path));
          return detailImage;
        })
        .collect(Collectors.toList());

    club.setDetailImages(detailImages);

    for (String categoryName : categoryNames) {
      Category category = categoryRepository.findByName(categoryName)
          .orElseThrow(() -> new IllegalArgumentException("Category not found: " + categoryName));
      club.addCategory(category);
    }

    clubRepository.save(club);
    log.info("동아리 등록 완료 {}", club.getTitle());
    return ResponseEntity.ok(club.getTitle());
  } catch (Exception e) {
```

MEMBER REGISTER & LOGIN



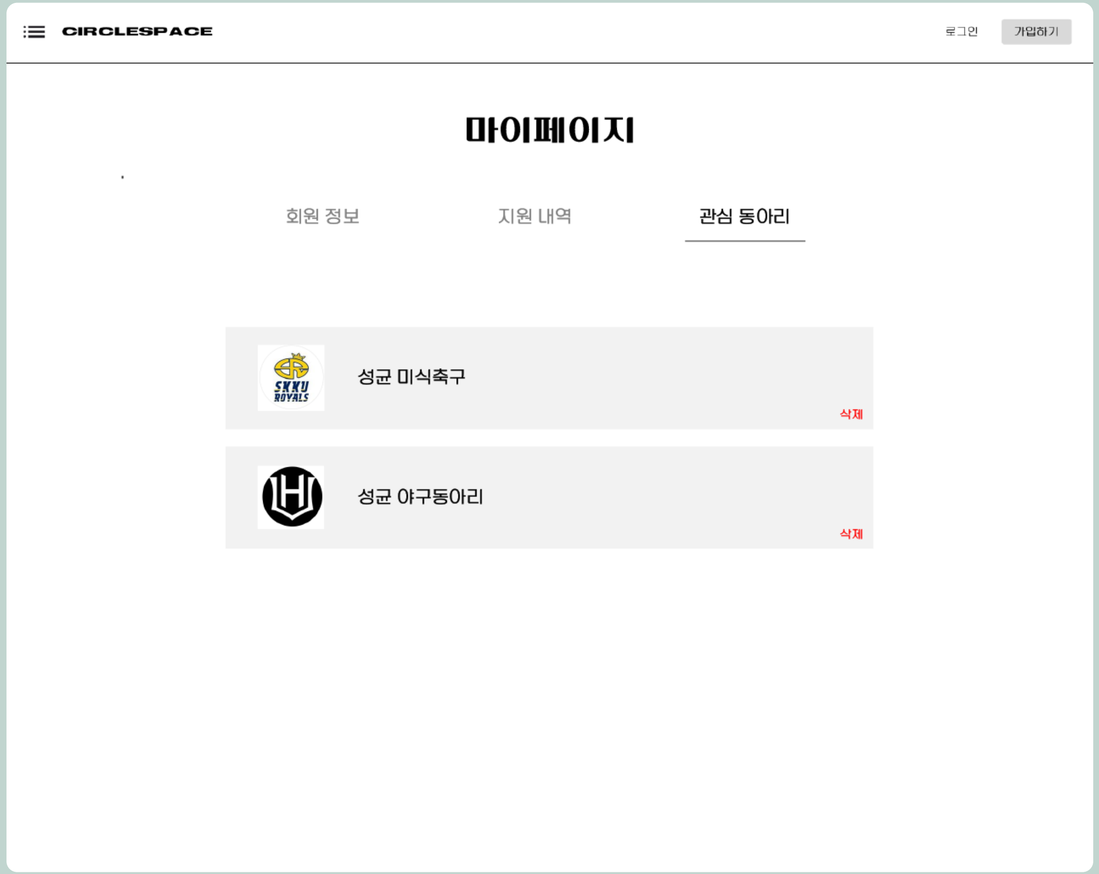

```java
@Entity
@ToString
@Getter
@Setter
public class Member {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  @Column(unique = true, nullable=false)
  private String username;
  private String displayName;
  private String realName;

  @Column(nullable=false)
  private String password;

  private String phoneNumber;
  private String role;
  private String schoolCode;

  @CreationTimestamp
  @Column(updatable = false)
  private LocalDateTime createdAt;

  @UpdateTimestamp
  private LocalDateTime updatedAt;

  @OneToMany(mappedBy = "member", cascade = CascadeType.ALL, orphanRemoval = true)
  private List<Like> likes = new ArrayList<>();

  @ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE})
```

MEMBER REGISTER & LOGIN




```java
@Log4j2
public class LoginFilter extends UsernamePasswordAuthenticationFilter {

  private final AuthenticationManager authenticationManager;
  private final JWTUtil jwtUtil;

  public LoginFilter(AuthenticationManager authenticationManager,JWTUtil jwtUtil) {
    this.authenticationManager = authenticationManager;
    this.jwtUtil = jwtUtil;
    setFilterProcessesUrl("/api/loginProc");
  }


  @Override
  public Authentication attemptAuthentication(HttpServletRequest request,
                                              HttpServletResponse response)
      throws AuthenticationException {
    String username = obtainUsername(request);
    String password = obtainPassword(request);

    log.info(username);

    UsernamePasswordAuthenticationToken authToken =
        new UsernamePasswordAuthenticationToken(username, password, null);

    return authenticationManager.authenticate(authToken);
  }

  @Override
  protected void successfulAuthentication(HttpServletRequest request,
                                          HttpServletResponse response,
                                          FilterChain chain,
                                          Authentication authentication)
      throws IOException {
```

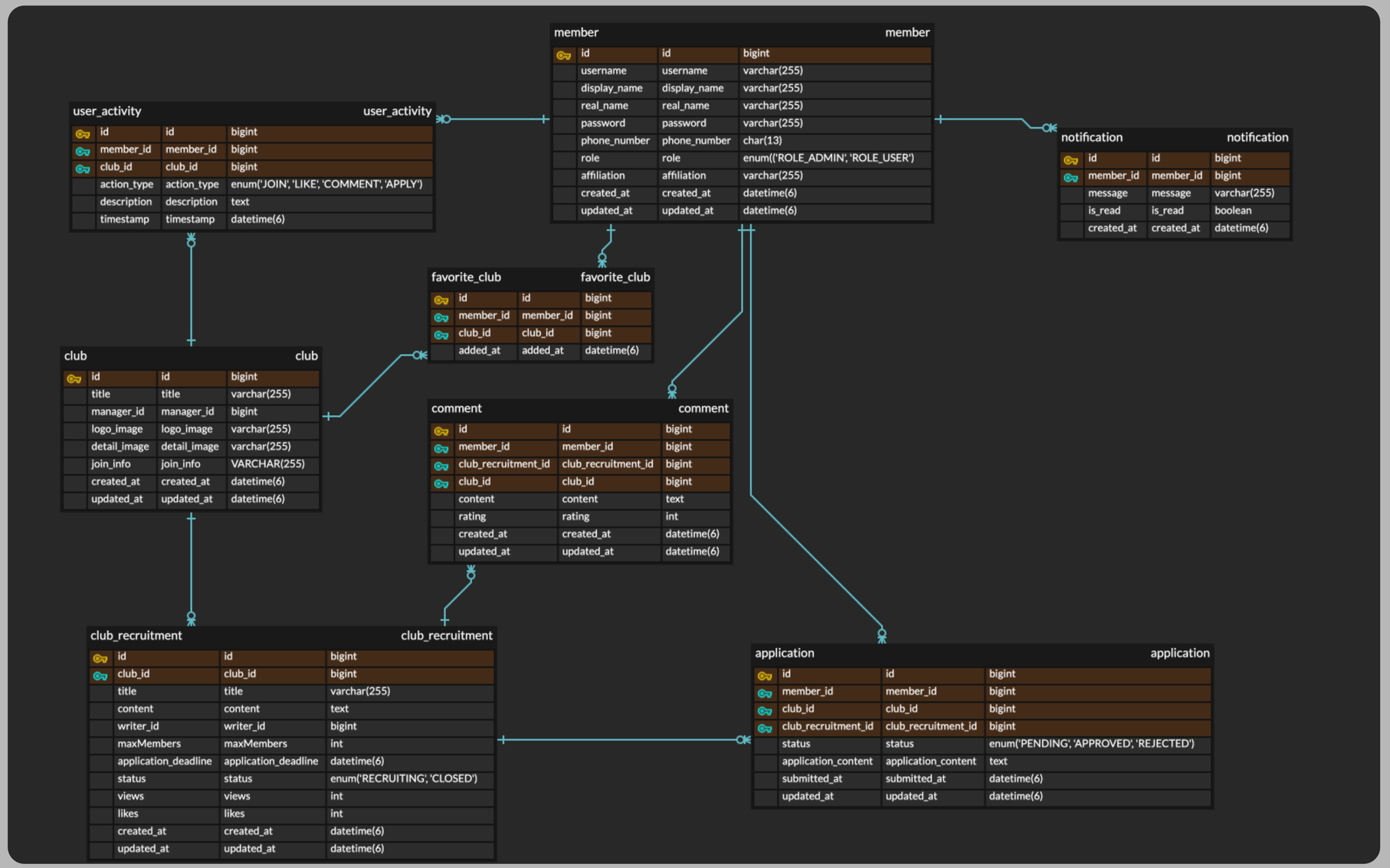
MYPAGE FEATURES



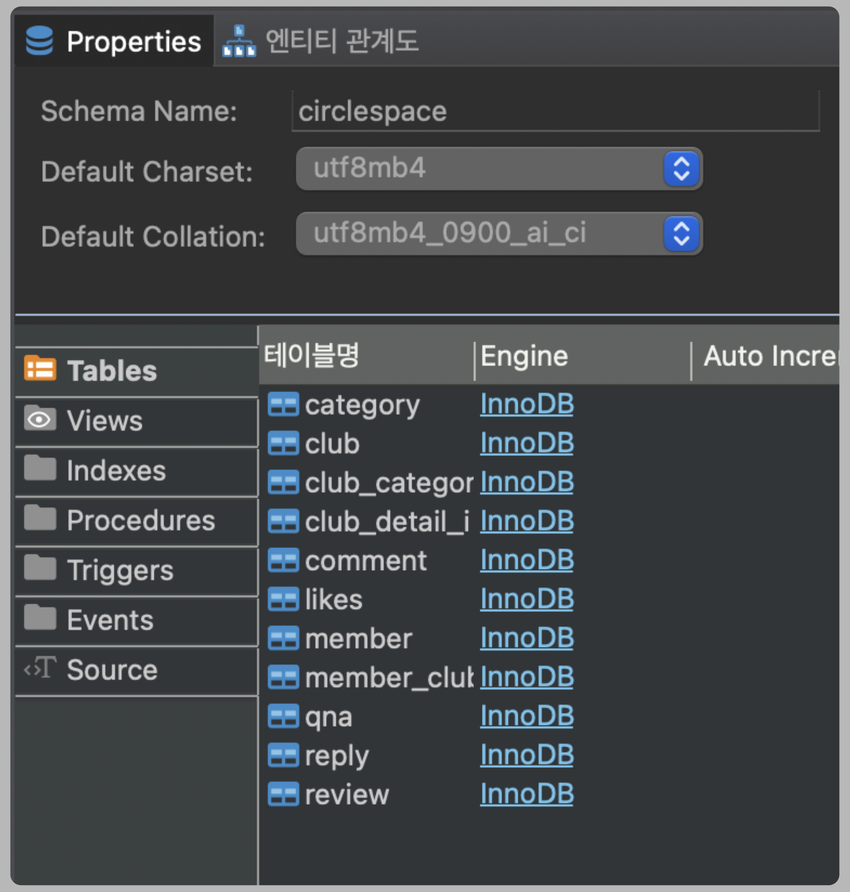

```java
@Transactional
public ResponseEntity<String> addToLike(String username, String title) {

  Member member = memberRepository.findByUsername(username);
  if (member == null) {
    return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("회원정보를 찾을 수 없습니다.");
  }

  Club club = clubRepository.findByTitle(title)
      .orElseThrow(() -> new IllegalArgumentException("동아리를 찾을 수 없습니다: " + title));

  Like exsistingLike = likeRepository.findByMemberAndClub(member, club);

  if (exsistingLike != null) {
    return ResponseEntity.status(HttpStatus.OK).body("Already Liked");
  } else {
    Like like = new Like();
    like.setMember(member);
    like.setClub(club);
    likeRepository.save(like);
  }

  return ResponseEntity.ok("Added to Like");
}
```

DATABASE

DATABASE


member
member
id id bigint
username username varchar(255)
display_name display_name varchar(255)
real_name real_name varchar(255)
password password varchar(255)
phone_number phone_number char(13)
role role enum('ROLE_ADMIN','ROLE_USER')
affiliation affiliation varchar(255)
created_at created_at datetime(6)
updated_at updated_at datetime(6)
user_activity
user_activity
id id bigint
member_id member_id bigint
club_id club_id bigint
action_type action_type enum('JOIN','LIKE','COMMENT','APPLY')
description description text
timestamp timestamp datetime(6)
notification
notification
id id bigint
member_id member_id bigint
message message varchar(255)
is_read is_read boolean
created_at created_at datetime(6)
favorite_club
favorite_club
id id bigint
member_id member_id bigint
club_id club_id bigint
added_at added_at datetime(6)
club
club
id id bigint
title title varchar(255)
manager_id manager_id bigint
logo_image logo_image varchar(255)
detail_image detail_image varchar(255)
join_info join_info VARCHAR(255)
created_at created_at datetime(6)
updated_at updated_at datetime(6)
comment
comment
id id bigint
member_id member_id bigint
club_recruitment_id club_recruitment_id bigint
club_id club_id bigint
content content text
rating rating int
created_at created_at datetime(6)
updated_at updated_at datetime(6)
club_recruitment
club_recruitment
id id bigint
club_id club_id bigint
title title varchar(255)
content content text
writer_id writer_id bigint
maxMembers maxMembers int
application_deadline application_deadline datetime(6)
status status enum('RECRUITING','CLOSED')
views views int
likes likes int
created_at created_at datetime(6)
updated_at updated_at datetime(6)
application
application
id id bigint
member_id member_id bigint
club_id club_id bigint
club_recruitment_id club_recruitment_id bigint
status status enum('PENDING','APPROVED','REJECTED')
application_content application_content text
submitted_at submitted_at datetime(6)
updated_at updated_at datetime(6)

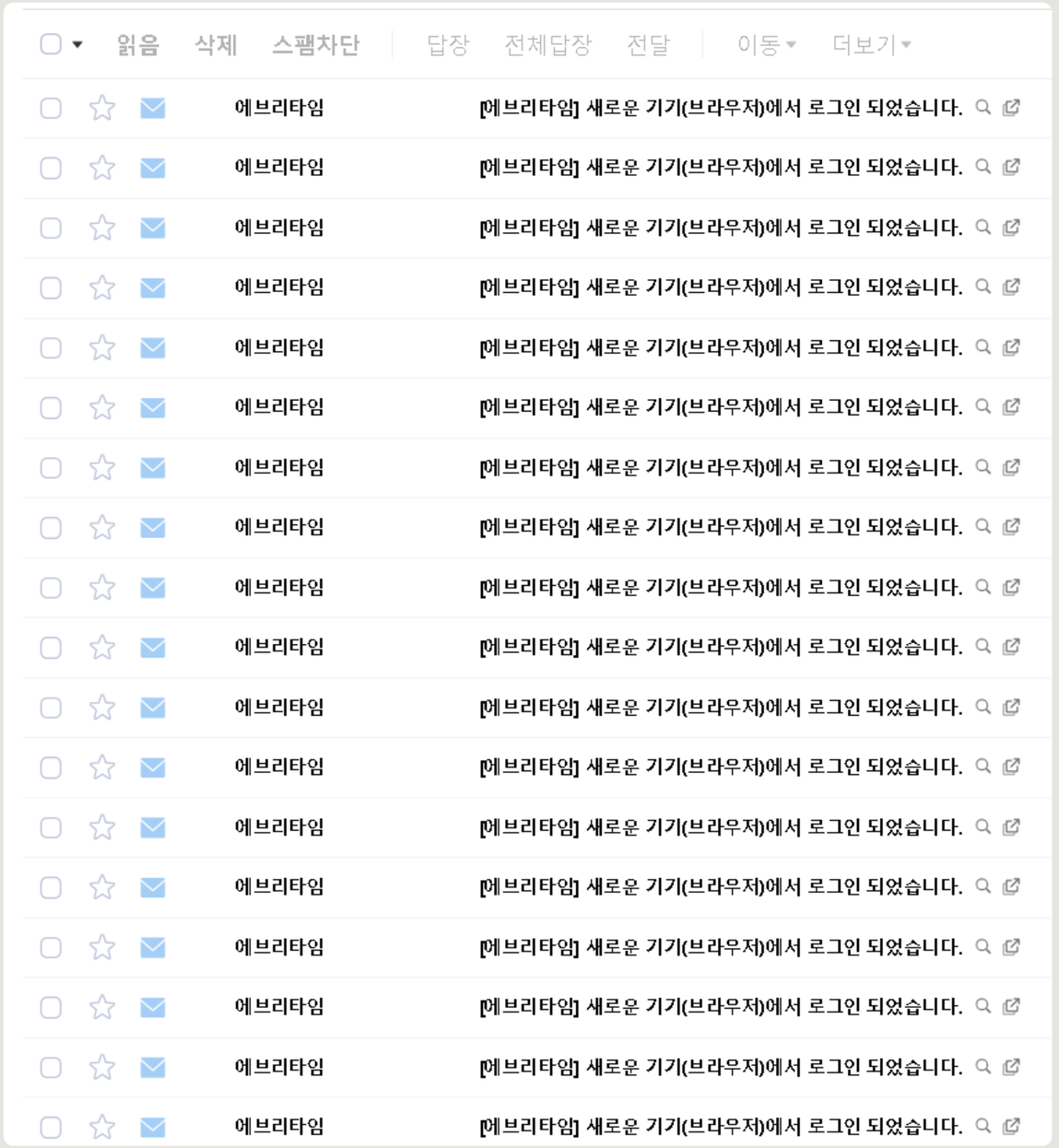
DATABASE




```sql
CREATE TABLE `club` (
    `club_id`   bigint  NOT NULL    COMMENT '동아리',
    `club_name` varchar(255)    NULL    COMMENT '동아리 이름',
    `manager_id`    bigint  NULL    COMMENT '동아리 대표자 id',
    `logo_image`    varchar(255)    NULL    COMMENT '로고 이미지',
    `detail_image`  varchar(255)    NULL    COMMENT '상세 이미지',
    `join_info` VARCHAR(255)    NULL,
    `created_at`    datetime(6) NULL    COMMENT '생성 일시',
    `updated_at`    datetime(6) NULL    COMMENT '정보 수정 일시'
);

CREATE TABLE `club_recruitment` (
    `recruitment_id`    bigint  NOT NULL,
    `club_id`   bigint  NOT NULL,
    `title` varchar(255)    NULL,
    `content`   text    NULL,
    `writer_id` bigint  NULL,
    `maxMembers`    int NULL,
    `application_deadline`  datetime(6) NULL,
    `status`    enum('RECRUITING', 'CLOSED')    NULL,
    `views` int NULL,
    `created_at`    datetime(6) NULL,
    `updated_at`    datetime(6) NULL
);

CREATE TABLE `comment` (
    `comment_id`    bigint  NOT NULL,
    `recruitment_id`    bigint  NOT NULL,
    `user_id`   bigint  NOT NULL    COMMENT '멤버',
    `content`   text    NULL,
    `created_at`    datetime(6) NULL,
    `updated_at`    datetime(6) NULL
);
```

CHALLENGES : TRIAL AND ERROR

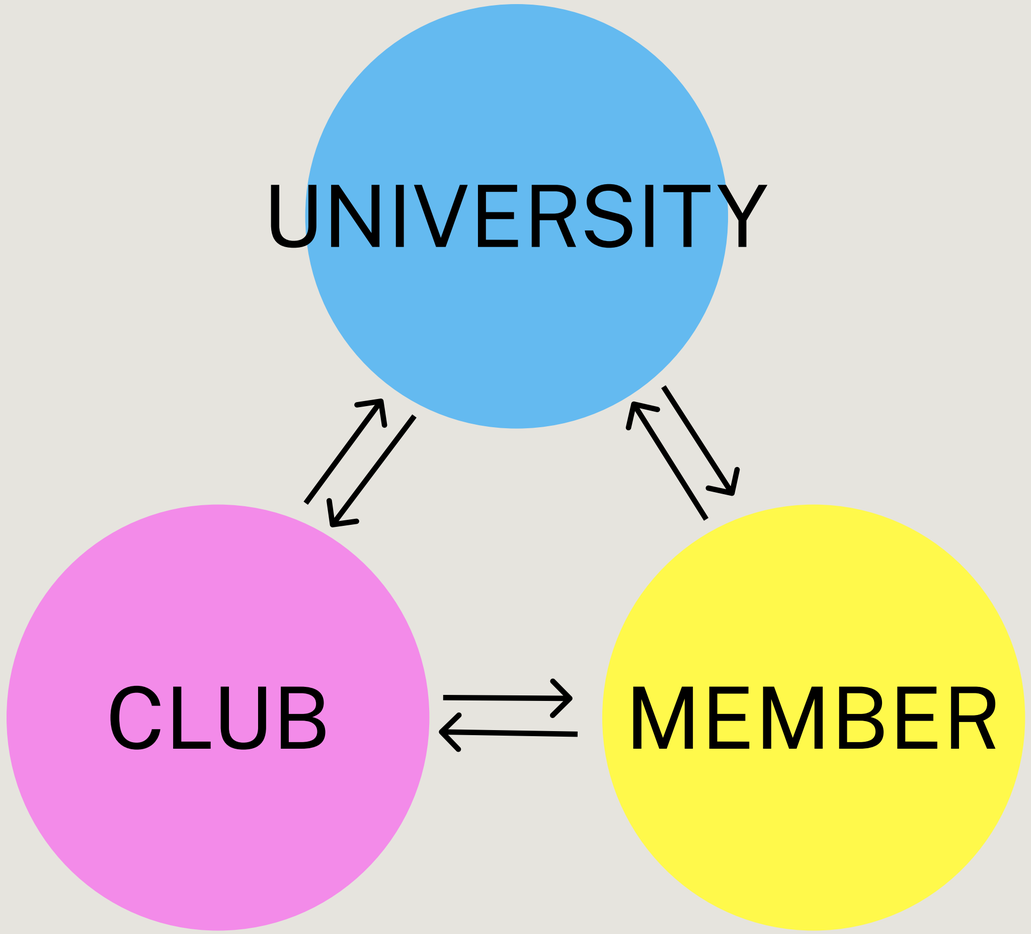
SECURE WEB CRAWLING



SECURE WEB CRAWLING

```java
// 쿠키 저장 메서드
private static void saveCookies(WebDriver driver, File file) throws IOException {  1 usage  GK
    FileWriter fileWriter = new FileWriter(file);
    BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
    Set<Cookie> cookies = driver.manage().getCookies();
    for (Cookie cookie : cookies) {
        bufferedWriter.write( str: cookie.getName() + ";" + cookie.getValue() + ";" + cookie.getDomain() + ";"
                + cookie.getPath() + ";" + cookie.getExpiry() + ";" + cookie.isSecure());
        bufferedWriter.newLine();
    }
    bufferedWriter.close();
}

// 쿠키 로드 메서드
private static void loadCookies(WebDriver driver, File file) throws IOException {  1 usage  GK
    FileReader fileReader = new FileReader(file);
    BufferedReader bufferedReader = new BufferedReader(fileReader);
    String line;
    while ((line = bufferedReader.readLine()) != null) {
        String[] cookieDetails = line.split( regex: ";");
        Cookie cookie = new Cookie.Builder(cookieDetails[0], cookieDetails[1])
                .domain(cookieDetails[2])
                .path(cookieDetails[3])
                .expiresOn( expiry: null) // 만료 날짜를 처리할 필요가 있음
                .isSecure(Boolean.parseBoolean(cookieDetails[5]))
                .build();
        driver.manage().addCookie(cookie);
    }
    bufferedReader.close();
}
```

DATABASE SCHEMA COMPLEXITY



THANK YOU

날다람쥐랩

SEOKHYUN BAE
JIWOO CHUNG
GUHO KIM
JINU PARK