요구사항 정의서

* 목차

1.1. 프로젝트 소개

저희 서비스는 사용자가 본 웹툰/웹소설에 대해 별점 및 리뷰를 남기고 별점에 따라 나의 취향에 맞는 작품을 추천해주는 서비스입니다.
또한 매주 새롭게 올라오는 회차에 대한 라이브톡을 진행하여 독자 모두가 모여 이야기하며 작품에 대한 감상을 나눌 수 있습니다.

1.2. 프로젝트 배경

문화체육관광부 보도자료

보도시점 2023. 9. 7.(목) 08:00 배포 2023. 9. 7.(목) 08:00

웹소설 산업 규모 1조 390억 원, 이용자 수 587만 명

- '2022 웹소설 산업 현황 실태조사' 결과 발표…웹소설 산업 관련 정부 첫 실태조사 실시
- 산업 성장세라는 긍정 신호와 더불어 불공정계약 등 지속성장 위한 개선 과제도 도출
- 9. 8. '민관 합동 웹소설 상생협의체' 출범, 표준계약서 제정 등 추진

문화체육관광부(장관 박보균, 이하 문체부)는 한국출판문화산업진흥원(원장 김준희)과 함께 계속 성장하고 있는 웹소설 시장의 현주소를 확인하고 향후 건강한 생태계 조성을 위한 제도개선의 기초 통계를 확보하고자 '2022 웹소설 산업 현황 실태('21년 기준)'를 조사하고 그 결과를 발표했다. 웹소설 분야에 있어서는 정부가 처음 실시한 실태조사로 앞으로 2년마다 조사가 이루어질 계획이다.

기업 매출액, 이용자 수 등 웹소설 산업 지속 성장세

이번 조사 결과, 웹소설 분야 산업 규모는 약 1조 390억 원으로, 2020년도 6,400억 원 대비 3,990억 원 증가(62% 성장)한 것으로 조사되었으며, 웹소설 이용자 수는 약 587만 명으로 파악되었다.

출처 : 닐슨코리아클릭 | 문화체육관광부 보도자료

현재 웹툰 및 웹소설을 드라마로 제작한 작품들이 성공을 거두며, 이로 인해 원작에 대한 관심도가 상승하고 있습니다. 닐슨코리아클릭의 통계에 따르면, 2020년 7월에 웹툰과 웹소설의 총 이용시간은 전년 동월 대비 24% 증가했습니다. 또한 문체부의 보고에 따르면, 웹소설 산업의 규모는 1조 390억 원으로 나타났습니다. 하지만 규모에 비해 작품 정보를 얻을 수 있는 충분한 곳이 부족한 상황입니다.

우리는 동일한 작품을 공유하는 사용자 간에는 서로 다른 시각과 해석을 나누며 다양한 관점을 탐험하여 더욱 만족스러운 작품 감상을 즐길 수 있습니다.

1.3. 프로젝트 시나리오

1. 사용자가 회원가입을 한다.
    - o 이메일를 입력합니다.
    - o 패스워드를 입력합니다.
    - o 회원가입을 완료합니다.
2. 사용자가 인증받은 이메일과 패스워드를 입력하여 로그인합니다.
3. 사용자는 마이페이지에 접속합니다.
    - o 마이페이지에서는 닉네임을 설정합니다.
    - o 마이페이지에서는 내가 평가한 작품, 나를 팔로우하는 회원 목록, 내가 팔로잉한 회원 목록을 확인 할 수 있습니다.
4. 작품을 선택하여 들어갑니다.
    - o 작품에 대한 코멘트와 별점을 남길 수 있습니다.

        코멘트 글자 수는 200자를 넘길 수 없습니다.

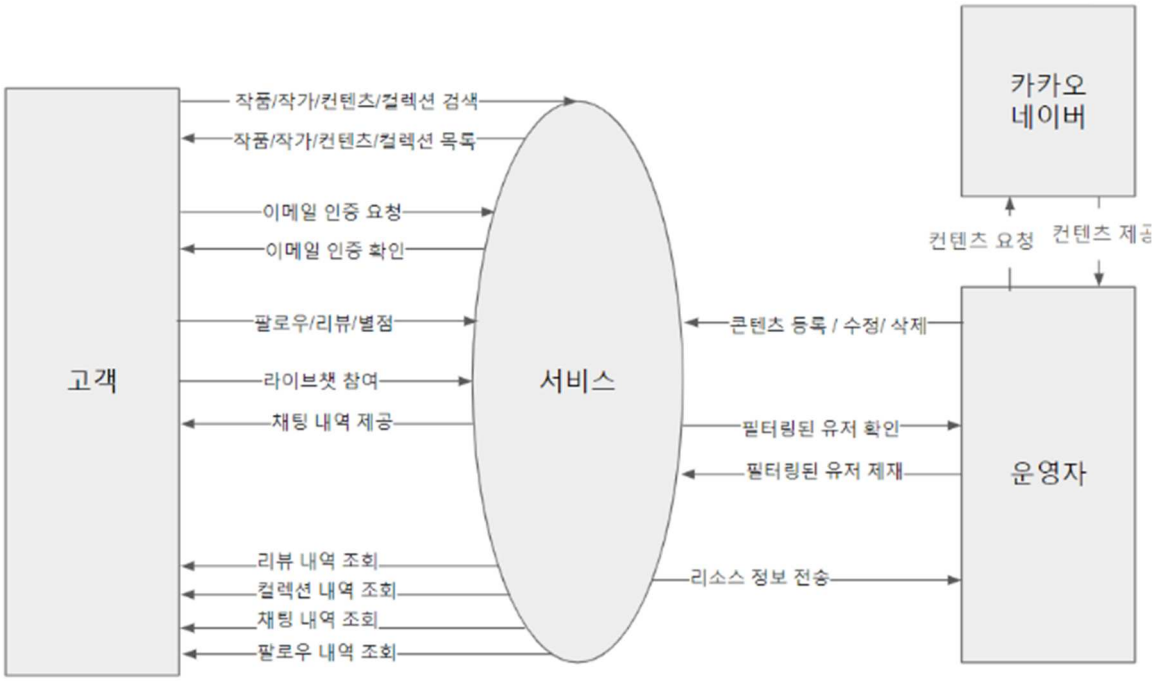
        욕설/비방어는 사용할 수 없습니다.
    - o 작품에 대한 회차별 또는 전체 작품에 대한 별점을 남길 수 있습니다.
    - o 전체 작품의 전체 평균 별점을 확인할 수 있습니다.
    - o 작품의 평가 상태를 확인할 수 있습니다.
    - o 작품명, 작가를 검색하여 작품을 찾을 수 있습니다.
5. 사용자는 자신이 보는 웹툰/웹소설이 업로드 되는 날에 라이브톡에 들어갑니다.

- o 라이브톡은 상위 n개의 작품들에서만 진행됩니다.
- o 라이브톡에는 회원들만 참여가 가능합니다.
- o 12시가 되면 이전의 채팅들이 사라집니다.
- o 채팅들이 사라지기 전에 사용자들은 채팅 내용을 백업할 수 있습니다.
- o 라이브톡에서 욕설/비방어를 입력할 경우 채팅을 보낼 수 없다는 메세지가 뜹니다.
- o 라이브톡은 상시로 참여할 수 있습니다.

6. 사용자는 자신이 좋아하는 작품을 클릭해 해당 작품이 담겨있는 컬렉션을 봅니다.

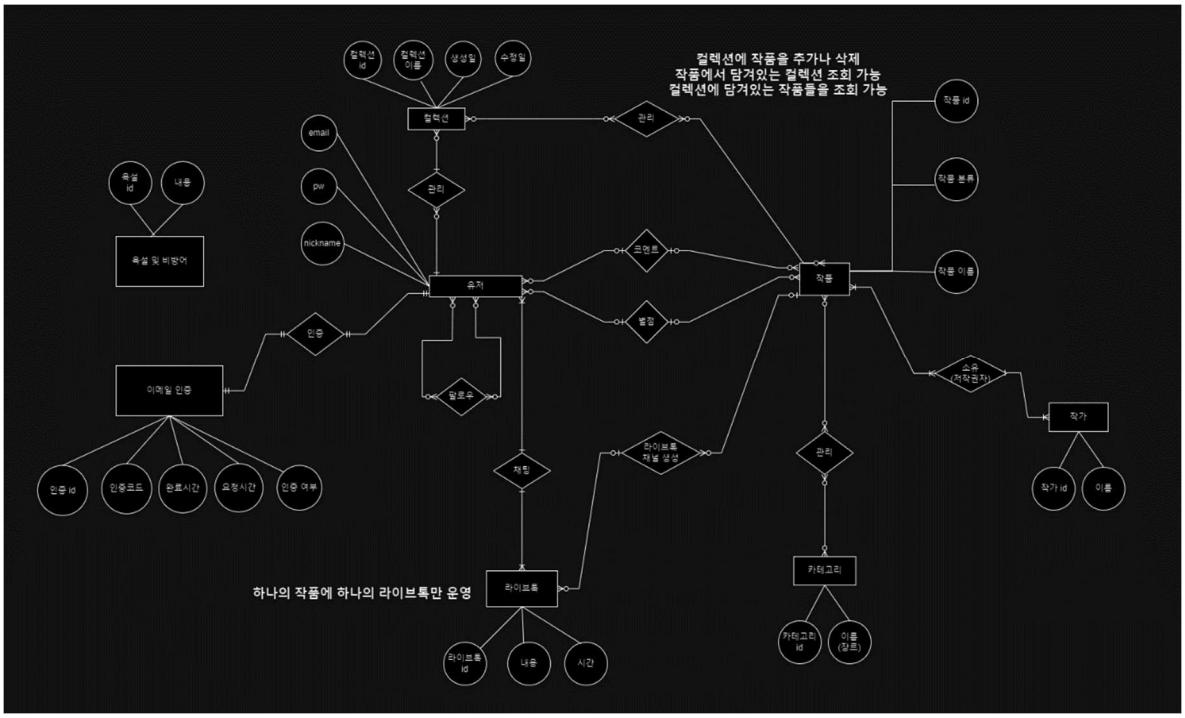
- o 사용자는 작품들을 담아서 컬렉션들을 만들 수 있습니다.
- o 회원 닉네임, 컬렉션 이름을 검색하여 다른 사용자가 만든 컬렉션을 볼 수 있습니다.
- o 작품이 담겨있는 다른 사용자들의 컬렉션을 볼 수 있습니다.

1.4. 업무 배경도

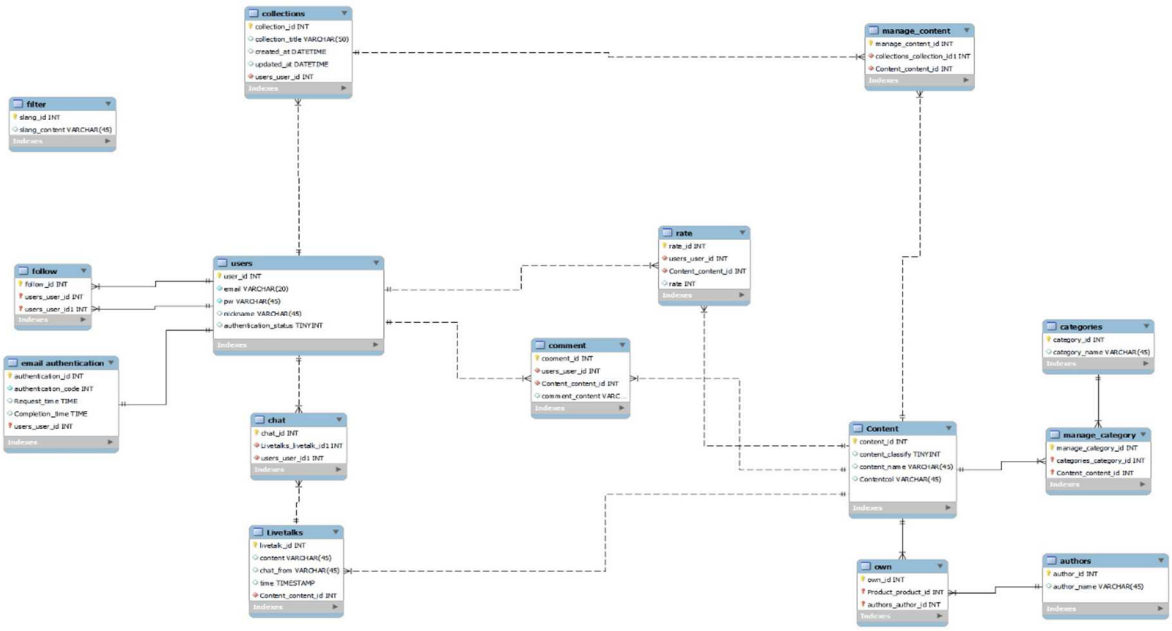

카카오
네이버
컨텐츠 요청
컨텐츠 제공
고객
작품/작가/컨텐츠/컬렉션 검색
작품/작가/컨텐츠/컬렉션 목록
이메일 인증 요청
이메일 인증 확인
팔로우/리뷰/별점
라이브챗 참여
채팅 내역 제공
서비스
콘텐츠 등록 / 수정/ 삭제
필터링된 유저 확인
필터링된 유저 제재
운영자
리소스 정보 전송
리뷰 내역 조회
컬렉션 내역 조회
채팅 내역 조회
팔로우 내역 조회

2. 설계

2-1. ERD



2.2 릴레이션 스키마



3. 구현

3.1 릴레이션

user

```
CREATE TABLE IF NOT EXISTS `mydb`.`users` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(20) NOT NULL,
  `pw` VARCHAR(45) NOT NULL,
  `nickname` VARCHAR(45) NULL,
  `authentication_status` TINYINT NULL,
  PRIMARY KEY (`user_id`))
ENGINE = InnoDB;
```

collection

```
CREATE TABLE IF NOT EXISTS `mydb`.`collections` (
  `collection_id` INT NOT NULL AUTO_INCREMENT,
  `collection_title` VARCHAR(50) NULL,
  `created_at` DATETIME NULL,
  `updated_at` DATETIME NULL,
  `users_user_id` INT NOT NULL,
  PRIMARY KEY (`collection_id`),
  INDEX `fk_collections_users1_idx` (`users_user_id` ASC) VISIBLE,
  CONSTRAINT `fk_collections_users1`
    FOREIGN KEY (`users_user_id`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

content

```
CREATE TABLE IF NOT EXISTS `mydb`.`Content` (
  `content_id` INT NOT NULL AUTO_INCREMENT,
  `content_classify` TINYINT NULL,
  `content_name` VARCHAR(45) NULL,
  `Contentcol` VARCHAR(45) NULL,
  PRIMARY KEY (`content_id`))
ENGINE = InnoDB;
```

category

```
CREATE TABLE IF NOT EXISTS `mydb`.`categories` (
  `category_id` INT NOT NULL AUTO_INCREMENT,
  `category_name` VARCHAR(45) NULL,
  PRIMARY KEY (`category_id`))
ENGINE = InnoDB;
```

Author

```
CREATE TABLE IF NOT EXISTS `mydb`.`authors` (
  `author_id` INT NOT NULL AUTO_INCREMENT,
  `author_name` VARCHAR(45) NULL,
  PRIMARY KEY (`author_id`))
ENGINE = InnoDB;
```

livetalk

```
CREATE TABLE IF NOT EXISTS `mydb`.`Livetalks` (
  `livetalk_id` INT NOT NULL AUTO_INCREMENT,
  `content` VARCHAR(45) NULL,
  `chat_from` VARCHAR(45) NULL,
  `time` TIMESTAMP NULL,
  `Content_content_id` INT NOT NULL,
  PRIMARY KEY (`livetalk_id`),
  INDEX `fk_Livetalks_Content1_idx` (`Content_content_id` ASC) VISIBLE,
  CONSTRAINT `fk_Livetalks_Content1`
    FOREIGN KEY (`Content_content_id`)
    REFERENCES `mydb`.`Content` (`content_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Chat

```
CREATE TABLE IF NOT EXISTS `mydb`.`chat` (
  `chat_id` INT NOT NULL AUTO_INCREMENT,
  `Livetalks_livetalk_id1` INT NOT NULL,
  `users_user_id1` INT NOT NULL,
  INDEX `fk_Livetalks_has_users_Livetalks2_idx` (`Livetalks_livetalk_id1` ASC) VISIBLE,
  INDEX `fk_Livetalks_has_users_users2_idx` (`users_user_id1` ASC) VISIBLE,
  PRIMARY KEY (`chat_id`),
  CONSTRAINT `fk_Livetalks_has_users_Livetalks2`
    FOREIGN KEY (`Livetalks_livetalk_id1`)
    REFERENCES `mydb`.`Livetalks` (`livetalk_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Livetalks_has_users_users2`
    FOREIGN KEY (`users_user_id1`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

follow

```
CREATE TABLE IF NOT EXISTS `mydb`.`follow` (
  `follow_id` INT NOT NULL AUTO_INCREMENT,
  `users_user_id` INT NOT NULL,
  `users_user_id1` INT NOT NULL,
  PRIMARY KEY (`follow_id`, `users_user_id`, `users_user_id1`),
  INDEX `fk_users_has_users_users2_idx` (`users_user_id1` ASC) VISIBLE,
  INDEX `fk_users_has_users_users1_idx` (`users_user_id` ASC) VISIBLE,
  CONSTRAINT `fk_users_has_users_users1`
    FOREIGN KEY (`users_user_id`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_users_has_users_users2`
    FOREIGN KEY (`users_user_id1`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

filter

```
CREATE TABLE IF NOT EXISTS `mydb`.`filter` (
  `slang_id` INT NOT NULL AUTO_INCREMENT,
  `slang_content` VARCHAR(45) NULL,
  PRIMARY KEY (`slang_id`))
ENGINE = InnoDB;
```

Manage content

```
CREATE TABLE IF NOT EXISTS `mydb`.`manage_content` (
  `manage_content_id` INT NOT NULL AUTO_INCREMENT,
  `collections_collection_id1` INT NOT NULL,
  `Content_content_id` INT NOT NULL,
  INDEX `fk_collections_has_Product_collections2_idx` (`collections_collection_id1` ASC) VISIBLE,
  INDEX `fk_collections_has_Product_Product2_idx` (`Content_content_id` ASC) VISIBLE,
  PRIMARY KEY (`manage_content_id`),
  CONSTRAINT `fk_collections_has_Product_collections2`
    FOREIGN KEY (`collections_collection_id1`)
    REFERENCES `mydb`.`collections` (`collection_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_collections_has_Product_Product2`
    FOREIGN KEY (`Content_content_id`)
    REFERENCES `mydb`.`Content` (`content_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

email authentication

```
CREATE TABLE IF NOT EXISTS `mydb`.`email authentication` (
  `authentication_id` INT NOT NULL AUTO_INCREMENT,
  `authentication_code` INT NOT NULL,
  `Request_time` TIME NULL,
  `Completion_time` TIME NULL,
  `users_user_id` INT NOT NULL,
  PRIMARY KEY (`authentication_id`, `users_user_id`),
  INDEX `fk_email authentication_users1_idx` (`users_user_id` ASC) VISIBLE,
  CONSTRAINT `fk_email authentication_users1`
    FOREIGN KEY (`users_user_id`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Manage category

```
CREATE TABLE IF NOT EXISTS `mydb`.`manage_category` (
  `manage_category_id` INT NOT NULL AUTO_INCREMENT,
  `categories_category_id` INT NOT NULL,
  `Content_content_id` INT NOT NULL,
  PRIMARY KEY (`manage_category_id`, `categories_category_id`, `Content_content_id`),
  INDEX `fk_categories_has_Content_Content1_idx` (`Content_content_id` ASC) VISIBLE,
  INDEX `fk_categories_has_Content_categories1_idx` (`categories_category_id` ASC) VISIBLE,
  CONSTRAINT `fk_categories_has_Content_categories1`
    FOREIGN KEY (`categories_category_id`)
    REFERENCES `mydb`.`categories` (`category_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_categories_has_Content_Content1`
    FOREIGN KEY (`Content_content_id`)
    REFERENCES `mydb`.`Content` (`content_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Own

```
CREATE TABLE IF NOT EXISTS `mydb`.`own` (
  `own_id` INT NOT NULL AUTO_INCREMENT,
  `Product_product_id` INT NOT NULL,
  `authors_author_id` INT NOT NULL,
  PRIMARY KEY (`own_id`, `Product_product_id`, `authors_author_id`),
  INDEX `fk_Product_has_authors_authors1_idx` (`authors_author_id` ASC) VISIBLE,
  INDEX `fk_Product_has_authors_Product1_idx` (`Product_product_id` ASC) VISIBLE,
  CONSTRAINT `fk_Product_has_authors_Product1`
    FOREIGN KEY (`Product_product_id`)
```

```
    REFERENCES `mydb`.`Content` (`content_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Product_has_authors_authors1`
    FOREIGN KEY (`authors_author_id`)
    REFERENCES `mydb`.`authors` (`author_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

comment

```
CREATE TABLE IF NOT EXISTS `mydb`.`comment` (
  `cooment_id` INT NOT NULL AUTO_INCREMENT,
  `users_user_id` INT NOT NULL,
  `Content_content_id` INT NOT NULL,
  `comment_content` VARCHAR(200) NULL,
  INDEX `fk_users_has_Content_users1_idx` (`users_user_id` ASC) VISIBLE,
  INDEX `fk_users_has_Content_Content1_idx` (`Content_content_id` ASC) VISIBLE,
  PRIMARY KEY (`cooment_id`),
  CONSTRAINT `fk_users_has_Content_users1`
    FOREIGN KEY (`users_user_id`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_users_has_Content_Content1`
    FOREIGN KEY (`Content_content_id`)
    REFERENCES `mydb`.`Content` (`content_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

rate

```
CREATE TABLE IF NOT EXISTS `mydb`.`rate` (
  `rate_id` INT NOT NULL AUTO_INCREMENT,
  `users_user_id` INT NOT NULL,
  `Content_content_id` INT NOT NULL,
  `rate` INT NULL DEFAULT NULL,
  INDEX `fk_users_has_Content1_users1_idx` (`users_user_id` ASC) VISIBLE,
  INDEX `fk_users_has_Content1_Content1_idx` (`Content_content_id` ASC) VISIBLE,
  PRIMARY KEY (`rate_id`),
  CONSTRAINT `fk_users_has_Content1_users1`
    FOREIGN KEY (`users_user_id`)
    REFERENCES `mydb`.`users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_users_has_Content1_Content1`
    FOREIGN KEY (`Content_content_id`)
    REFERENCES `mydb`.`Content` (`content_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

3.2 기능별 쿼리

이메일과 패스워드를 입력합니다.

```
INSERT INTO mydb.users (email, pw)
VALUES ('qwer123@naver.com', '12345');
```

회원가입을 완료합니다.

```
UPDATE mydb.users
SET authentication_status =1
WHERE email ='qwer123@naver.com';
```

사용자가 인증받은 이메일과 패스워드를 입력하여 로그인합니다.

```
SELECT *
FROM mydb.users
WHERE email ='qwer123@naver.com'
AND pw ='12345' AND authentication_status =1;
```

마이페이지에서는 닉네임을 설정합니다.

```
Update users
Sent ickname ='asdf'
Where users.email='aaa@gmail.com';
```

마이페이지에서는 내가 평가한 작품을 확인할 수 있습니다.

```
Select collection_title
From collections
Left join users
On collections.users_user_id =users.user_id
Where nickname ='bbb' and users.user_id =13;
```

마이페이지에서는 다른 사용자를 팔로우 할 수 있습니다

```
Insert into follow (users_user_id, Users_user_id1) values(
(6 #내 유저 아이디
(select user_id
From users
Where email ='bbb@gmail.com') #팔로우할 사람의 유저 아이디
);
```

마이페이지에서는 나를 팔로우하는 회원 목록을 확인 할 수 있습니다

```
Select nickname, email
From users
left join follow
on users.user_id =follow.users_user_id
where user_id = 6;
```

마이페이지에서는 내가 팔로잉한 회원 목록을 확인 할 수 있습니다.

```
Select nickname, email
From users
left join follow
on users.user_id =follow.users_user_id
where nickname ='bbb' and email ='bbb@gmail.com';
```

작품에 대한 코멘트와 별점을 남길 수 있습니다.(수정 가능)

```
별점이나 코멘트 남기기
INSERT INTO mydb.rate (users_user_id,Content_content_id,rate)
VALUES(6, 6, 3)
INSERT INTO mydb.comment (users_user_id,Content_content_id,comment_content)
VALUES(6, 6, '작화가 너무 멋져요')
수정하기 –
Update mydb.rate set rate ='5' whereusers_user_id =20;
Update mydb.comment set comment_content ='끝까지 봐야하는 명작!'
where users_user_id =20;
```

욕설/비방어는 사용할 수 없습니다.

```
SELECT comment_content FROM comment;
```

전체 작품의 전체 평균 별점을 확인할 수 있습니다.

```
Select content_name, round(avg(rate),1)
From mydb.rate
join mydb.Content
on mydb.rate.Content_content_id =mydb.Content.content_id
where Content_content_id =6;
```

작품의 평가 상태를 확인할 수 있습니다.

```
SELECTcontent_name,
CASE
WHEN mydb.rate.rate ISNULL THEN '미평가'
ELSE mydb.rate.rate
END AS '평가상태'
FROM mydb.Content
LEFT JOIN mydb.rate ON mydb.Content.content_id =mydb.rate.Content_content_id
WHERE users_user_id =6;
```

작품명, 작가를 검색하여 작품을 찾을 수 있습니다.

뷰 생성

```
Use mydb;
CREATE VIEW search as
Select Content.content_name, categories.category_name, authors.author_name
From mydb.manage_category
join mydb.categories
on manage_category.categories_category_id =categories.category_id
join mydb.Content
on manage_category.Content_content_id =Content.content_id
join mydb.own
on manage_category.Content_content_id =own.Content_Content_id
join mydb.authors
on own.authors_author_id =authors.author_id;
```

작품명으로 검색

```
Select *
From search
Where content_name ='화산귀환';
```

작가이름으로 검색

```
Select *
From search
Where author_name ='조석';
```

라이브톡에는 회원들만 참여가 가능합니다.

```
SELECT *
FROM users
WHERE email IS NOT NULL AND authentication_status = 1;
```

12시가 되면 이전의 채팅들이 사라집니다.

12시가 되면 이전의 채팅들이 사라집니다.

```
# 스케줄 상태 확인

Show global variables like'%schedule%';

# OFF 라면 ON

Set global event_scheduler=On;

Se t@@global.event_scheduler=On;

# 진행중인 스케줄 목록

SELECT * FROMi nformation_schema.events;

# 초기화

Drop event ifexists check_expected_end_date;

# 실행

Create event check_expected_end_date

# 하루마다

On schedule every 1day

# 기준시간

starts '2023-11-01 12:00:00'

enable

do

DELETE FROM Livetalks;
```

채팅들이 사라지기 전에 사용자들은 채팅 내용을 백업할 수 있습니다.

채팅 내역 조회

```
SELECT users.user_id, users.nickname , content, time FROM Livetalks
JOIN chat
ON Livetalks.livetalk_id =chat.Livetalks_livetalk_id1
JOIN users
ON chat.users_user_id1 =users.user_id;
```

백업 테이블 3개

mysqldump -u [사용자 계정] -p [패스워드] [원본 데이터베이스명] [테이블 명] >[생성할 백업 DB명].sql

mysqldump -u test_user -p Livetalks chat users >backup_db.sql

~# 비번 :

라이브톡에서 욕설/비방어를 입력할 경우 채팅을 보낼 수 없다는 메세지가 뜹니다.

```
Show global variables like '%schedule%';
Set global event_scheduler=On;
Set @@global.event_scheduler=On;
SELECT * FROMinformation_schema.events;
Drop event ifexists check_filter;
Create event check_filter
On schedule every 1second
starts current_timestamp
enable
do
SELECT * FROM FILTER WHERE (SELECT content FROM Livetalks)
LIKE '%'OR slang_content OR '%';
```

라이브톡은 상시로 참여할 수 있습니다.

```
SELECT * FROM Livetalks;
```

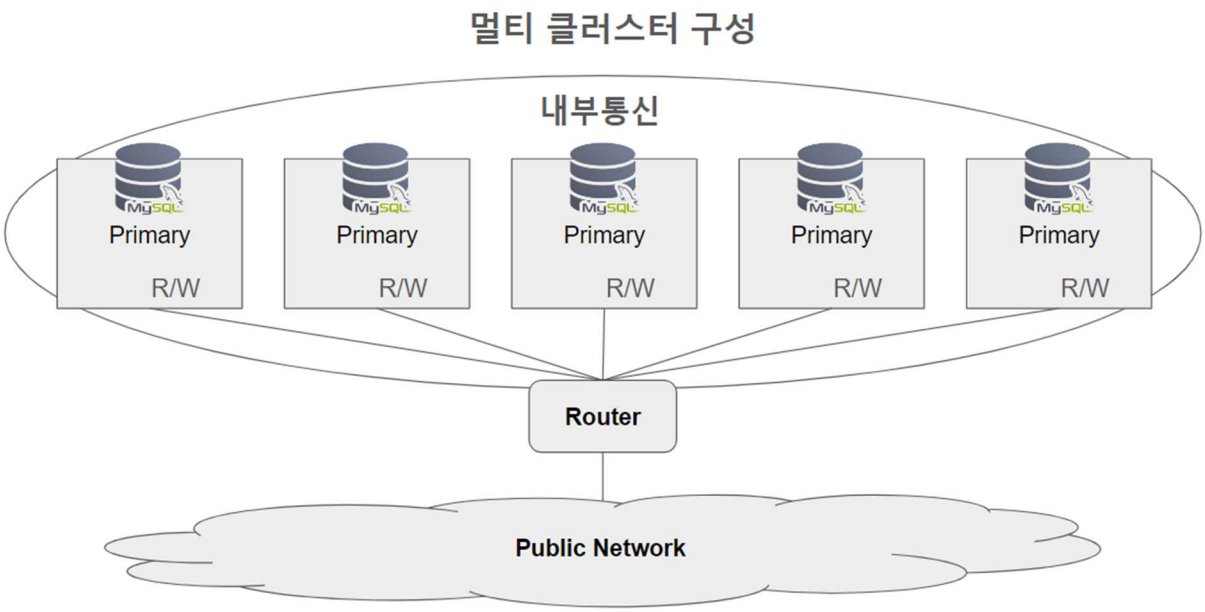
사용자는 작품들을 담아서 컬렉션들을 만들 수 있습니다.

```
insert into mydb.collections (collection_title, users_user_id)
value ('강추 웹소설', 20);
Insert i manage_content (collections_collection_id1, Content_content_id)
Values ( 20, 13 ) ;
```

회원 닉네임, 컬렉션 이름을 검색하여 다른 사용자가 만든 컬렉션을 볼 수 있습니다.

```
# 닉네임으로 컬렉션 조회
Select mydb.collections.collection_title, mydb.Content.content_name
From collections
join manage_content
on mydb.collections.collection_id =mydb.manage_content.collections_collection_id1
join Content
on manage_content.Content_content_id =mydb.Content.content_id
join users
on mydb.collections.users_user_id =mydb.users.user_id
where nickname='bbb';
# 작품명으로 컬렉션 조회
Select mydb.users.nickname, mydb.collections.collection_title, mydb.Content.content_name
From collections
join manage_content
on mydb.collections.collection_id =mydb.manage_content.collections_collection_id1
join Content
on manage_content.Content_content_id =mydb.Content.content_id
join users
on mydb.collections.users_user_id =mydb.users.user_id
where mydb.Content.content_name like '%아홉%';
```

3-3. 시스템 아키텍쳐



저희 서비스는 리뷰 작성과 리스트 만들기 그리고 채팅 등의 서비스가 활발하게 이용될 수 있도록 운영할 계획입니다

이러한 서비스를 운영하게 되면 읽기와 쓰기 기능을 모두 자주 사용해야 될것이라 예상됩니다

그래서 비용 절감에도 유리하고 부하 분산과 높은 가용성으로 인해 안정적인 서버 상태를 유지할 수 있는 멀티 클러스터 방식으로 서버를 구성했습니다

[서버 구성]

PC 6대 준비

> PC 5대를 클러스터 노드로 만들어 DB 서버를 구성

> PC 1대는 라우터로 구성해서 라우터를 통해 DB에 접근

*6646 포트는 읽기/쓰기 포트이다. (9999 포트로 포트 포워딩)