Final Report

for

ToonFinder

Daye Eun, Taesik Choi, Kyounga Woo

06/10/19

Table of Contents

1. General Description

1.1 Goals / Requirements

The main goal of this project is to build a well-organized database based on domain we chose and building web application called ***Toonfinder*** which has interaction with the database storing different types of information such as Webtoon, writer, and reader etc. The webpage requires to have function to response users' requests such as finding webtoon's title with given genre. It has to be robust enough to withstand users and furthermore, user friendly interface is required.

/*The goal of this document is to describe structure of database, function and manual of ***Toonfinder***. The intended readers of this document are users who want to acknowledge how ***Toonfinder*** is built and works in developer and user's view. After reading this document, readers can understand overall process of the result and way to run ***Toonfinder.*** */

1.2 Domain

Our main domain is Webtoon and its related domain. For this project the related domain is reader, company, cartoonist, team and related content. We implemented the relation between Webtoon, team and company. We also focused on the relation regarding webtoon and related contents such as movies and novels.

1.3 General Idea

Toonfider gives general information about "Webtoons" (Korean web-based cartoons) on different platforms (Naver, Daum, KakaoPage) and allows users to search webtoons based on their different tastes (title, genre, author, views and score etc). The information is webtoon title, author, genre, period, one-line description, rating, worth and webtoon related contents. The web

page requires authentication in order to save user information and do additional function such as subscribe.

1.4 Key Tool / Technology Used

We used Maria DB to build database and get connection between server and client's sides. Various scripting languages are used for page setup, functioning and style of ***Toonfinder***. PHP and html are mainly used for functioning and overall structure of pages. CSS is used for style of visualized features such as shape and color.

2. Use Cases

2.1 Search Webtoons

2.1.1 Search Webtoons

Goal	Search Webtoons that satisfy condition.
Precondition	N/A
Action	Type keyword in Title textbox. Select checkbox of company or genre or worth. Click Search to get the result.

2.2 Arrange Management

2.2.1 Arrange Cartoonists

Goal	Filter cartoonists and re-order them.
Precondition	N/A
Action	Navigate to ‘Cartoonists’ in the toolbox. Click the search box and select one of the options: ‘All’, ‘Both’, ‘Story’, or ‘Drawing’. Press ‘Click’ Button.

2.2.2 Arrange Webtoons

Goal	Re-arrange Webtoons with different standard.
Precondition	N/A
Action	Navigate to ‘Webtoons’ in the toolbox. Click the search box and select one of the options: ‘Views’, ‘Score’, ‘Alphabet’. Press ‘Click’ Button.

2.2.3 Arrange Related Contents

Goal	Filter related contents and re-order them.
Precondition	N/A
Action	Navigate to ‘Related Contents’ in the toolbox. Click the search box and select one of the options: ‘All Work’, ‘Underlying Work’, ‘Derived Work’, and ‘Series’. Press ‘Click’ Button.

2.3 View Information

2.3.1 Cartoonist Information

Goal	View individual cartoonist's information.
Precondition	N/A
Action	Navigate to 'Cartoonists' in the toolbox. Click 'View Info' button in one of the cartoonists.

2.3.2 Webtoon / Related Contents Information

Goal	View individual Webtoon's information.
Precondition	N/A
Action	① Navigate to 'Webtoons' in the toolbox. Click 'View Info' button in one of the Webtoons. ② Navigate to individual information of the cartoonist: 'Cartoonists' -> 'View Info'. Scroll down to 'Works' and press 'View Info' button.

2.4 Authentication

2.4.1 Create Account

Goal	Create account for the website.
Precondition	N/A
Action	Navigate to 'Login' in the toolbox. Press 'Create Account' button. Type the user name, password, age in the textbox. Select the gender among 'Female', 'Male', 'I don't want to Check'. Select country. Then press 'Create New Account' button.

2.4.2 Login

Goal	Login to the website
Precondition	N/A
Action	① Navigate to 'Login' in the toolbox. ② Without logged in, press the following buttons. From cartoonist page (see 2.3.1), press 'Favorite'. From 'Webtoons' page, press 'Like'. From Webtoon Information page (see 2.3.2), press 'Like' or 'Subscribe'. Then type the correct username and password. Press 'Login' button. If wrong username or password, the page redirects to login page.

2.3.2 Logout

Goal	Logout from the website.
Precondition	N/A
Action	Navigate to 'My Page' in the toolbox. Click 'Logout' link below the user information.

2.5 Preference Management

2.5.1 Subscribe Webtoon

Goal	Subscribe the Webtoon.
Precondition	The user should be logged in before clicking the button.
Action	Navigate to Webtoon page information (see 2.3.2) in the toolbox. Press 'Subscribe' button. When pop-up message appears, press 'Ok', The Webtoon will be added to 'Subscribed Webtoon' section in 'My Page'.

2.5.2 Like Webtoon

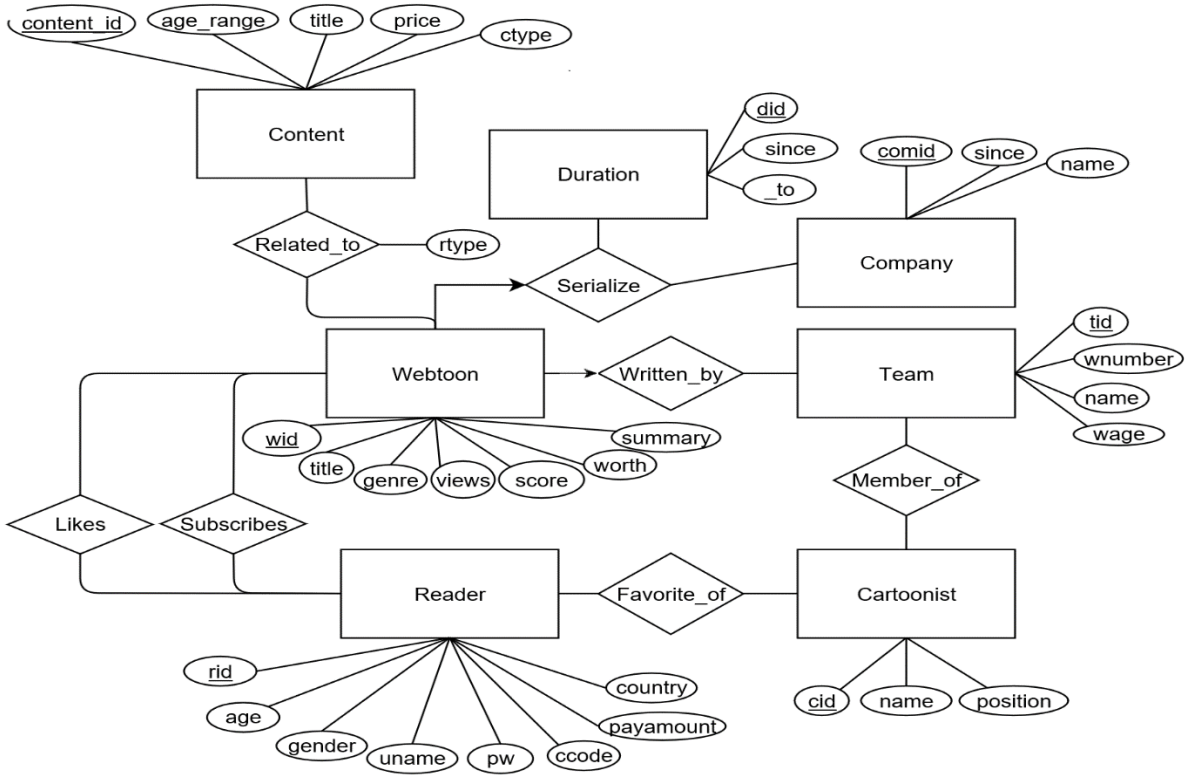
Goal	Add favorite Webtoon.
Precondition	The user should be logged in before clicking the button.
Action	① Navigate to 'Webtoons' page. ② Navigate to Webtoon page information (see 2.3.2) in the toolbox. Press 'Like' button. When pop-up message appears, press 'Ok', The Webtoon will be added to 'Favorite Webtoon' section in 'My Page'.

2.5.3 Favorite Cartoonist

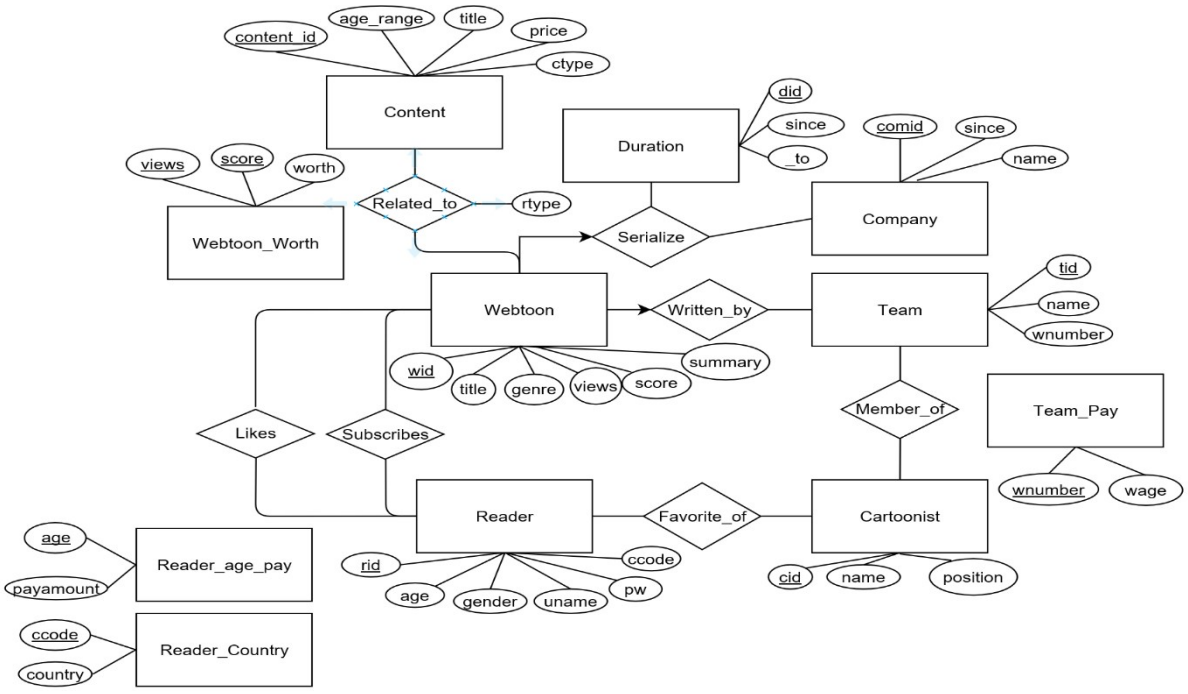
Goal	Add favorite cartoonist.
Precondition	The user should be logged in before clicking the button.
Action	Navigate to cartoonist page information (see 2.3.1) in the toolbox. Press 'Favorite' button. When pop-up message appears, press 'Ok', The cartoonist will be added to 'Favorite Cartoonists' section in 'My Page'.

3. ER Diagram

3.1 Whole ER Diagram



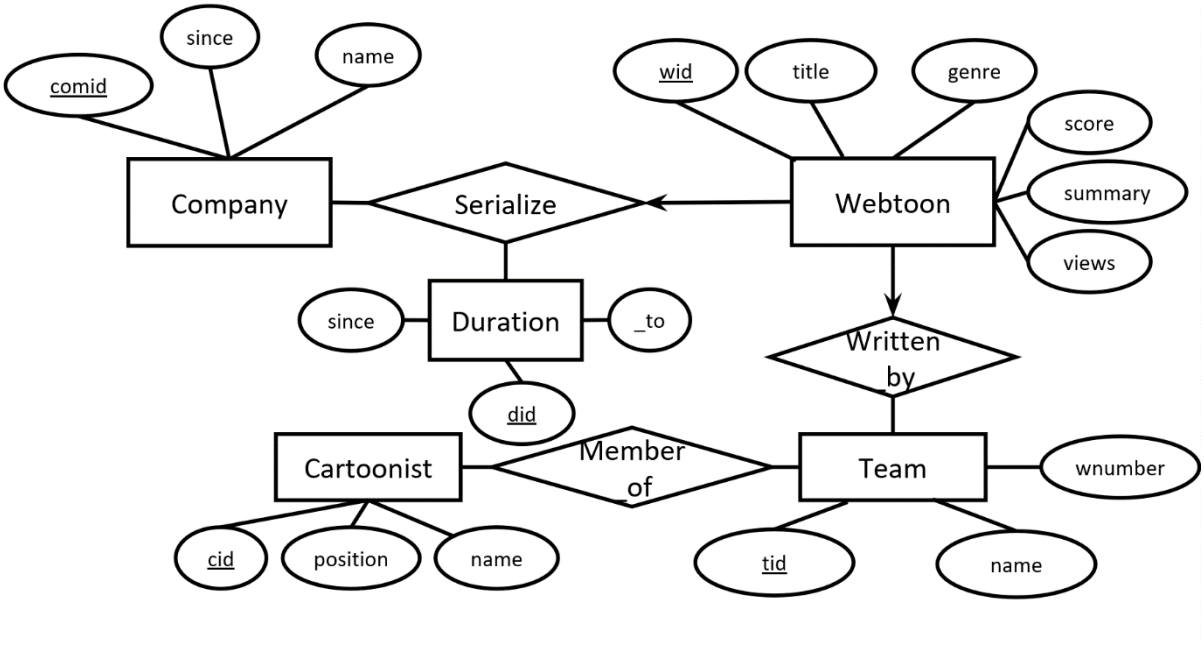
ER Diagram without Normalization



ER Diagram with Normalization

3.2 Partial ER Diagram

3.2.1 Company, Webtoon, Team, and Cartoonist

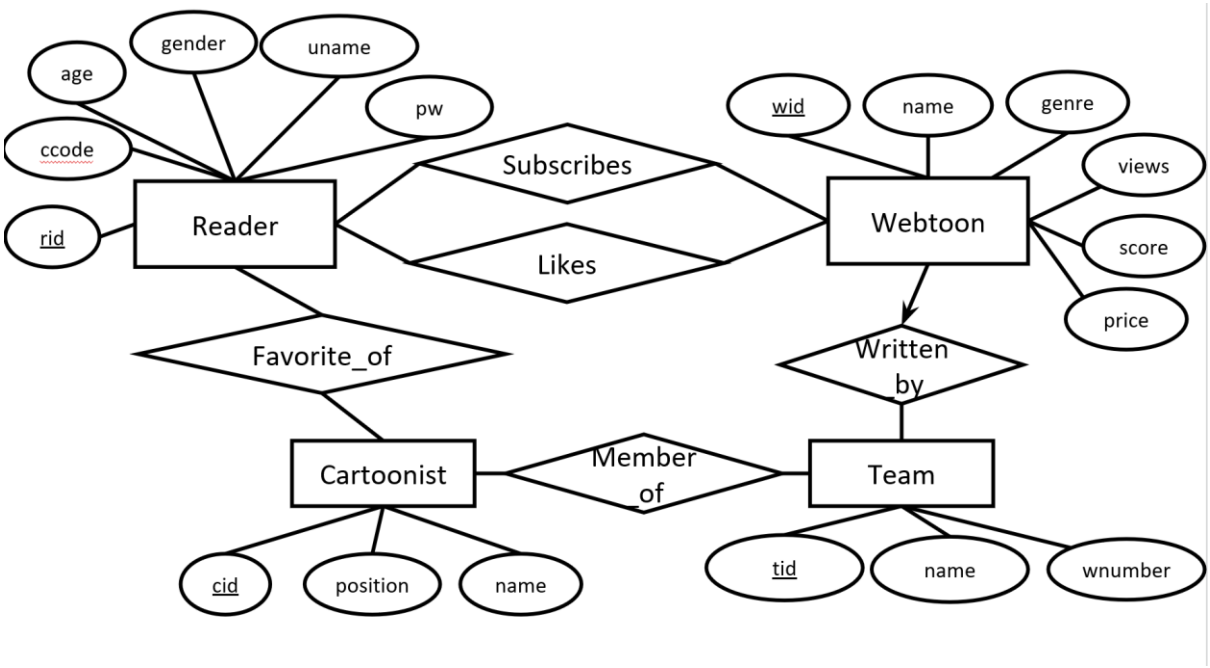


Main entity of our ER diagram is Webtoon and it has attributes of wid, title, genre, score, summary and views. wid is id and key. Every underscore attributes represents key in this diagram.

Entity Webtoon is serialized by entity Company and it can be serialized by only one Company. Duration entity with attributes did, since and _to expresses specific serialization dates.

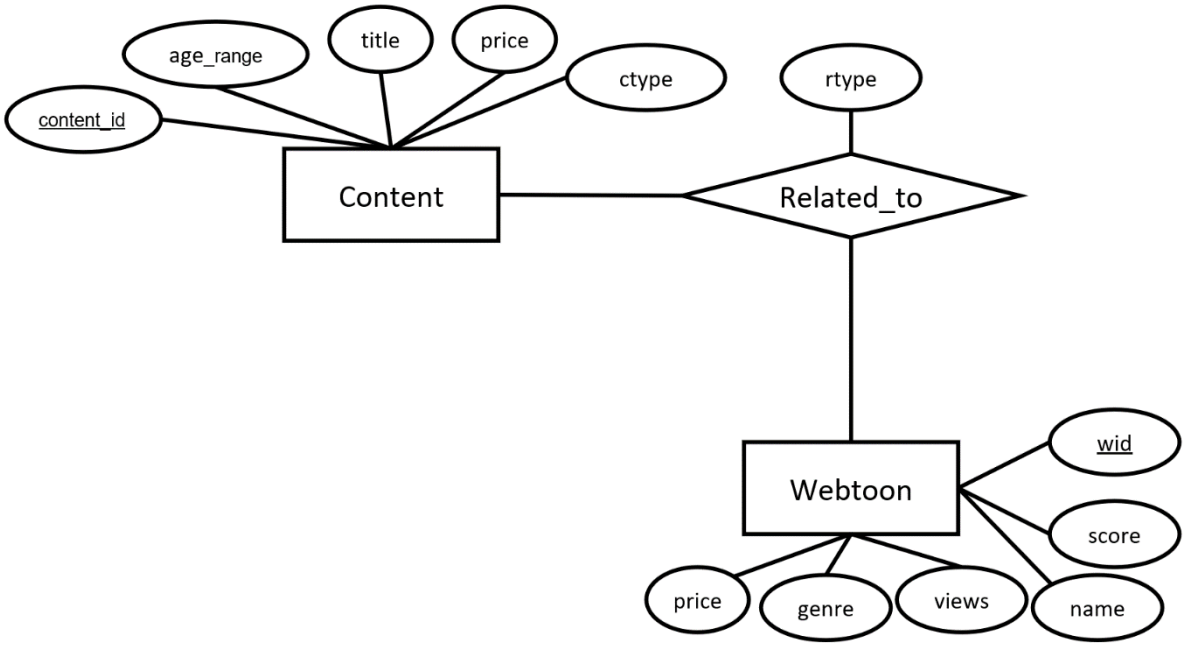
Cartoonist can grouped into Team to write Webtoon. Attribute position in entity Cartoonist can be classified into writer, printer and both. wnumber shows how many webtoons team serialize.

3.2.2 Reader Focused



Entity Reader has rid, ccode, age gender, uname and pw. ccode is for country code and uname, pw are username and password. Readers can subscribe or like specific webtoon by Subscribes and Likes relationship. Reader can also specify favorite cartoonists by Relationship Favorite_of.

3.2.3 Related Content Focused



Webtoon has its related content and entity Content has attributes of content_id, age_range, title, price, ctype. age_range is to distinguish users' access by their age. ctype is content's type such as movie and novel. rtype shows relationship between webtoon and novel.

4. Normalization

4.1 Functional Dependencies

Our database originally had 14 relations, and 3 of them violated BCNF. Following relations, Webtoon, Reader, Team have extra functional dependencies other than the ones generated by the key.

4.1.1 Webtoon

Schema of Webtoon is the following:

Webtoon (*wid, name, score, price, genre, views, worth, summary*)

We will denote this relation schema by listing the attributes: WNSPGVOU

Functional dependencies on Webtoons are:

W -> WNSPGVOU, since *wid* is the key,

SV->O, *score* and *views* determine *worth*.

This is because the worth is the evaluated value that we generated based on score and views. We need worth to categorize the webtoons more conveniently. For worth, there are 5 possible values, which is ‘A’, ‘B’, ‘C’, ‘D’ and ‘F’. Evaluation criteria is the following:

views > 10 or score > 9.5	A
8 <views <10 or 9<score<9.5	B
6<views<8 or 8<score<9	C
3<views<6 or 6<score<8	D
views<3 or score<6	F

4.1.2 Reader

Schema of Reader is the following:

Reader (*rid, uname, pw, age, payamount, gender, ccode, country*)

We will denote this relation schema by listing the attributes: RUWAPGOC

Functional dependencies on Reader are:

R -> RUWAPGOC, since *rid* is the key,

A->P, *age* determines *payamount*,

O->C, *ccode* determines *country*.

Age determines payamount because payamount is the restriction of pay per month that is given to the readers, to prevent them from overspending. Usually when the reader is younger, they have less money since they don’t work. Payamount goes higher for certain level and drops for seniors

since they usually spend less money on webtoons. Detailed dependency is the following:

1 – 4	0	Won
5 – 15	6000	Won
16 – 25	10000	Won
26 – 40	50000	Won
41 – 60	100000	Won
61 – 100	50000	Won

ccode determines country since ccode is the abbreviation of country name and it is unique to each country. We use ccode for the efficiency, since country is too long to reference.

4.1.3 Team

Schema of Team is the following:

Team (*tid, name, wnumber, wage*)

We will denote this relation schema by listing the attributes: TNUW

Functional dependencies on Team are:

T -> TNUW, since *tid* is the key,

U->W, *wnumber* determines *wage*.

This is because the team is making more webtoons, the wage tends to be higher, so we simplified the wage to be determined by wnumber, which is the number of webtoons that the team is creating. Detailed dependency is the following:

wnumber	wage (in Million Won)
1	300
2	500
3	700

4.2 Techniques Applied

To make the relations into BCNF, we used normalization, applying lossless join decomposition to the 3 relation above.

4.2.1 Webtoon

From Webtoon WNSPGVOU, we divide it into two table, WNSPGVOU and SVO since SV->O is violating the form. We named the new table SVO as Webtoon_Worth. After the decomposition, now the FDs are in BCNF and it is possible to obtain a lossless-join. Since no dependency is lost because SV->O was the only dependency in Webtoon, it is also dependency preserving. Therefore, there is a possible decomposition that meets BCNF.

4.2.2 Reader

From Reader RUWAPGOC, we first divide it into two table, RUAGOC and AP since A->P is violating the form. We named the new table AP as Reader_age_pay. RUAGOC still violates BCNF because of O->C. We divide RUAGOC into two table, RUAGO and OC. OC is named as Reader_Country. After the decompositions, now the FDs are in BCNF and it is possible to obtain a lossless-join. Since no dependency is lost, it is also dependency preserving. Therefore, there is a possible decomposition that meets BCNF.

4.2.3 Team

From Team TNUW, we divide it into two table, TNU and UW since U->W is violating the form. We named the new table UW as Team_Pay. After the decomposition, now the FDs are in BCNF and it is possible to obtain a lossless-join. Since no dependency is lost, it is also dependency preserving. Therefore, there is a possible decomposition that meets BCNF.

4.3 Result of Normalization

As a result, we modified Webtoon, Reader and Team, and got 4 additional relations: Webtoon_Worth, Reader_age_pay, Reader_Country and Team_Pay. Detailed schemas are the following.

From Webtoon,

Webtoon (*wid, name, score, price, genre, views, summary*)

Webtoon_Worth (*score, views, worth*)

From Reader,

Reader (*rid, uname, pw, age, gender, ccode*)

Reader_age_pay (*age, payamount*)

Reader_Country (*ccode, country*)

From Team,

Team (*tid, name, wnumber*)

Team (*wnumber, wage*)

In total, we have 18 relations in our database, and they are all in BCNF with lossless-join and dependency preserving. The advantages of normalization are mainly focused on efficiency. Memory-wise, it is more efficient because we removed redundancy. Also, it is efficient in a sense that the index for search is simpler. For example, in Reader, ccode is smaller value that consists of 2 characters while country is a long value. The disadvantages of normalization are that we have to join tables to display information. Since our application is informative app, we need to display all the relative information, which makes more joins.

5. Implementation

5.1 Search Engine

Our search engine allows users to enter title and select the company, genre, worth to search the webtoon that meets such selection.

```php
if (isset($_POST['search'])) {
    $title  = get_post($conn, 'title');
    $titleop = '';
    if($title != '') $titleop = "W.name LIKE '%$title%'";
    $companylist = get_options('S.comid', 'company');
    $genrelist = get_options('W.genre', 'genre');
    $worthlist = get_options('O.worth', 'worth');
    $queryop = array($titleop, $companylist, $genrelist, $worthlist);

    $query = "SELECT W.name, W.wid, W.genre FROM Webtoon W, Webtoon_Worth O, Serialize S
              WHERE S.wid = W.wid AND O.views = W.views AND O.score = W.score";
    $q = '';
    foreach($queryop as $selected) {
        if(!$selected == '') {
            $query = $query.' AND ('.$selected.')';
        }
    }
    $query = $query.';';
    $result  = $conn->query($query);
```

Figure 1. Query for Search

We joined Webtoon, Webtoon_Worth, Serialize table since worth, company and genre are from such tables. To change query depending on the checkbox the user check, we made a helper function called “get_options”.

```php
function get_options($type, $oplist) {
    $options = '';
    if(!isset($_POST[$oplist])) return $options;
    foreach($_POST[$oplist] as $selected) {

        if($options == '') {
            $options = $type." = '".$selected."'";
        }
        else {
            $options = $options." OR ".$type." = '".$selected."'";
        }
    }
    return $options;
}
```

Figure 2. Helper Function for creating Query

Helper function gets the list of selected options and creates string that contains all the options using OR operator. Concatenating all the values from get_options using AND operator, we get the desired query.

5.2 Login/Logout/Create Account

Our application is member-based, therefore we needed login, logout and create account.

```
if (isset($_POST['login']) && !empty($_POST['username'])
   && !empty($_POST['password'])) {
    $username   = get_post($conn, 'username');
    $password   = get_post($conn, 'password');
    $query  = "SELECT uname, pw FROM Reader WHERE uname = '$username' AND pw = '$password';";
    $result = $conn->query($query);
    if (!$result or mysqli_num_rows($result)==0) {
        echo "<script>if (confirm(\"Your information does not exist. Try different name or password\"))
    }
    else {
        $_SESSION['valid'] = true;
        $_SESSION['timeout'] = time();
        $_SESSION['username'] = $username;
        header('Refresh: 0; URL = userDisplay.php');
    }
    $result->close();
}
```

Figure 3. Query for Login

For login, we got the username and password using post method, and put those value in the SELECT query. If we found the right tuple with given uname and pw, we put the values to the SESSION variables. For logout, we cleared those SESSION variables.

```
if (    isset($_POST['username'])   &&
        isset($_POST['password']) &&
        isset($_POST['age']) &&
        isset($_POST['gender']) &&
        isset($_POST['country'])) {
    $uname  = get_post($conn, 'username');
    $pw = get_post($conn, 'password');
    $age    = intval(get_post($conn, 'age'));
    $gender     = trim($_POST['gender']);
    $ccode  =  trim($_POST['country']);
    $query = "SELECT * FROM Reader;";
    $result   = $conn->query($query);
    $rid = mysqli_num_rows($result);

    $query = "INSERT INTO Reader (rid, age, gender, ccode, uname, pw) VALUES ".
      "($rid, $age, '$gender', '$ccode', '$uname', '$pw');";
    $result   = $conn->query($query);
    if ($result) {
        echo "<script>alert(\"Create Account Success!\");</script>";
        $_SESSION['valid'] = true;
        $_SESSION['timeout'] = time();
        $_SESSION['username'] = $uname;
        header('Refresh: 0; URL = userDisplay.php');
    }
}
```

Figure 4. Query for Add Account

For creating new account, we got the basic information using post method, and put those value in INSERT query. rid is index in Reader, and we use them to search for liked, subscribed webtoons and other personal preference. If insert was successful, we automatically login using such information, setting the SESSION variables.

5.3 Personal Page

Our application has page that shows reader's basic information, along with the reader's preference based on Likes, Subscribes, and Favorite_of tables. We used simple SELECT query to find and display the basic information.

```
$query  = "SELECT W.name, W.wid, W.genre FROM Subscribes S, Webtoon W WHERE rid = '$rid' AND S.wid = W.wid;";
$result = $conn->query($query);
```

Figure 5. SELECT Query for displaying Subscribed Webtoons

For displaying Likes, Subscribes and Favorite_of, we had to join each table with Webtoons, using wid as an index. Since wid is a primary key, MySQL creates B-tree based index automatically.

```
$distinctGenre = array_unique($genre);
$genreCount = array_count_values($genre);
echo "<div>";
echo "<h3>Subscribed Webtoons Genre Stat</h3><table style=\"width:100%\"><tr><th>Genre</th><th>Count</th><tr>";
foreach ($distinctGenre as $c) {
    echo "<tr><td>$c</td><td>$genreCount[$c]</td></tr>";
}
echo "</table></div>";
```

Figure 6. Implementation for displaying Genre Statistics

Personal page also shows the genre statistics of the users liked or subscribed webtoons, and we used PHP functionalities to simply get the information. Detailed code is in Figure 6.

5.4 Webtoons/Cartoonists/Related Contents Page

We have pages that shows all the Webtoons, Cartoonists, and Related Contents. Each page has sort selection box that lets the user sort the elements by the criteria.

```
function scoreOrder() {
    $GLOBALS['sql'] = "SELECT * FROM Webtoon ORDER BY score DESC;";
    /*$GLOBALS['result'] = $conn->query($GLOBALS['sql']);*/
};

function viewOrder() {
    $GLOBALS['sql'] = "SELECT * FROM Webtoon ORDER BY views DESC;";
    //echo $GLOBALS['sql'];
    /*$GLOBALS['result'] = $conn->query($GLOBALS['sql']);*/
};

function alpaOrder() {
    $GLOBALS['sql'] = "SELECT * FROM Webtoon ORDER BY name ASC;";
    /*$GLOBALS['result'] = $conn->query($GLOBALS['sql']);*/
};

if(isset($_POST["key"])) {
    $value = trim($_POST["key"]);
    if($value === NULL or $value === "view") {
        viewOrder();
    }
    if($value === "score") {
        scoreOrder();
    }
    if($value === "alpa") {
        alpaOrder();
    }
}
$GLOBALS['result'] = $conn->query($GLOBALS['sql']);
$result = $conn->query($sql);
```

Figure 7. Query for Sorting

In Figure7, we change the SQL statement based on the selected key. Each SELECT Query has corresponding ORDER BY, with either descending or ascending order.

5.5 Webtoon/Cartoonist Individual Page

In Webtoons, Cartoonists page, we have buttons that goes to individual page. Individual display page gets the wid or cid and display the related information. Figure 8 shows the code that links individual page to webtoons/Cartoonists page.

```
<form action =\"webtoon.php\" method=\"post\">
    <input type='hidden' name='webtoon' value=".$row['wid']."><input type=\"submit\" class = \'
</div></div></div>";
```

Figure 8. Code for Connecting Individual Page

In Figure8, we post the wid when we click the button, “View Info”, which sends the webtoon id to the individual webtoon page.

```
if (isset($_POST["webtoon"])) {
    $_SESSION['wid'] = intval(get_post($conn, "webtoon"));
}
if(isset($_SESSION['wid'])) {
    $wid = $_SESSION['wid'];
    $query = "SELECT * FROM Webtoon WHERE wid = $wid";

    $result = $conn->query($query);
    if (!$result) die ("SELECT failed: " . $conn->error);

    $row = mysqli_fetch_assoc($result);
    $name = $row['name'];
    $genre = $row['genre'];
    $score = $row['score'];
    $views = $row['views'];
    $price = $row['price'];
    $summary = $row['summary'];
```

Figure 9. Code in Individual Page

In Figure9, we get the posted wid and use the value to find the webtoon and their information. We print the information on the page to display and give information.

```
if (isset($_POST['like'])) {
    if (isset($_SESSION['username'])) {
        $username = $_SESSION['username'];
        $query  = "SELECT* FROM Reader WHERE uname = '$username';";
        $result2 = $conn->query($query);
        $result2->data_seek(0);
        $row2 = $result2->fetch_array(MYSQLI_NUM);
        $rid = $row2[0];
        $query  = "INSERT INTO Likes (rid, wid) VALUES ($rid, $wid);";
        $result2 = $conn->query($query);
        if($result2) echo "<script>alert(\"Liked!\");</script>";
    }
    else {
        echo "<script>if (confirm(\"Please Login\")) {location.replace(\"userlogin.php\");}</script>";
    }

}
```

Figure 10. Subscribe/Like Webtoon

The user can subscribe or like the webtoon from the individual page, and we used INSERT query in order to add the webtoon. We get the rid from the SESSION variable, and use wid in the individual page.

6. Result

6.1 Achievements

Through this project, we made some achievements. Our original goal to build Webtoon Search and View site were met, but additional goals such as recommendation function was deleted due to time. We were able to build a functioning web app with php based on our database about webtoon and its users, and learn how to use PHP and MySQL, while being able to apply technics learned in class such as normalization. We were also able to learn how to collaborate with peers.

6.1.1 Project

We have built the web application that displays information about webtoon and its related contents. It shows webtoons cartoonists and related contents, sorted by some categories.

Our application has more function, allowing the users to create account and login as a reader and let them view, like, subscribe the webtoons and cartoonists. Readers have their personal page to view their favorites and subscription.

We also have a webtoon search engine which is the main page, that lets the user to check the category such as company, genre and worth to find the webtoon they want. Webtoons are also searchable by name.

6.1.2 Technics

From this project, we were able to apply techniques learned in class. We had to choose the domain in our life that can be represented to database and built the design. We choose webtoon and was able to find the relations and draw ER diagram to make it clear.

After designing, we went through the process of normalization, making our DB to follow BCNF. We were able to find dependency preserving lossless join of our database and applied it to our DB.

We were also able to learn how to use PHP and MySQL. We used numerous SQL statements to get desired information from our DB, and used PHP to display the information that will help the user experience.

6.1.3 Teamwork

Our team consists of three members, and we learned a lot working together by learning each one’s specialty and looking after each other. We distributed our works evenly for each part to give opportunity to learn each part of the process, but at the same time we tried to distribute the work amount based on our skills.

We had to make time to meet and discuss, and our team was able to meet every week to process the work. If we didn’t consider each other’s situation, we would have trouble finishing the project.

6.2 Extendibility

6.2.1 Scalability

Our current database has information of 60 selected webtoons and its cartoonists and related contents. We choose 3 big companies that hosts webtoons, which are NAVER, DAUM and KAKAO and selected 20 webtoons for each. To make this project bigger, we can include all webtoons and include small companies that also hosts webtoons. This scaling will need some atomization of collecting the data. Building the WebCrawler that collects the information can scale the size of this project.

6.2.2 Additional Features

Our current functionality offers the maintenance of subscribed, liked webtoons and favorite cartoonists. Later we can add functionality that recommends the new webtoon based on this preference, and give notification if the new webtoon or episode is uploaded.

Using the pay amount and showing the webtoon pay plan to the reader is another possible function. It shows the reader how much they are paying to read webtoon and how much is left for possible subscription.

We can also add functions such as giving discount based on ages and their possible pay amount. Having special promotion will get more users to use the application.

6.2.3 Customer

Our current customer who uses the application is readers who likes webtoons or reads them. However, we can get Companies as our target also, when we have enough users to gain their preference. We can sell the information to the company and suggest the right price for the webtoon they are offering. Also, we can get special arrangements to give readers discount when they use our web application.

6.3 Contributions

For each category of work, we tried to distribute the work evenly, but for some work such as normalization, we gathered together to discuss and apply, so it is hard to tell the individual contributions for such cases. Those cases are: discussing use cases, drawing ER diagram and applying normalization, and documentation.

6.3.1 Daye Eun

For basic research, Daye collected webtoons that are from Kakaopage. For making database, she created some readers, subscribes, likes, and favorite_of data to insert to the table. She also worked on creating data for related contents. Implementing PHP, she was on charge of creating My Page (userDisplay.php), and Search(searchEngine.php). She worked on login, login and create account while contributed on connecting individual page to the view info button.

6.3.2 Kyounga Woo

For basic research, Kyounga collected webtoons that are from DAUM. For making database, she created some readers, subscribes, likes, and favorite_of data to insert to the table. She also worked on creating data for webtoons and collected image for the webtoons. Implementing PHP, she was on charge of creating overall view pages for webtoons, cartoonists, and related works that has selection box for filtering. She also worked on changing design.

6.3.3 Taesik Choi

For basic research, Taesik collected webtoons that are from NAVER. For making database, he created cartoonist, team, and company data to insert to the table. He collected images of related contents and added summary for the webtoon instances. Implementing PHP, he was on charge of creating individual view pages for webtoons, cartoonists. He was also in charge of changing design, including button alignment and color scheme.

7. References

7.1 HTML/CSS/JavaScript Template License

We downloaded HTML5 template and modified it in our use. Used some JavaScript and CSS for the effect, and used some HTML structure for the PHP page. PHP codes are all original and not from the template.

License Information for Free Template

https://themewagon.com/license/

Download link

https://themewagon.com/themes/free-bootstrap-4-html5-real-estate-website-template-homeland/

7.2 Webtoon and Content Images and Information

Webtoon images and information are all from the websites that are hosting the webtoons.

Hosting Websites

NAVER Webtoon	https://comic.naver.com/webtoon/weekday.nhn
DAUM Webtoon	http://webtoon.daum.net/
Kakao Page	https://page.kakao.com/main

Appendix A: User's Manual

This section introduces key features of the ***ToonFinder*** website, along with major functions. For the ease of reading this user manual, specialized terminologies are described. Then, it helps users to get started with the application.

A.1 Key Features

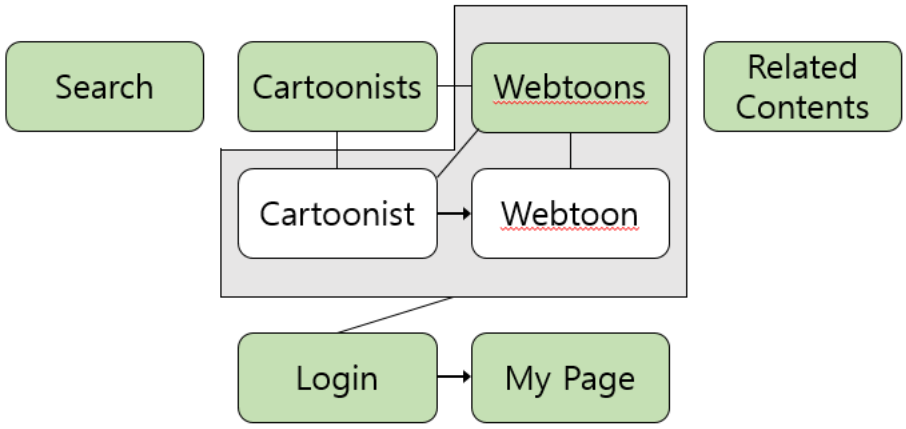


Figure 1. ToonFinder Mapping

There are six main pages – Search, Cartoonists, Webtoons, Related Contents, Login, and My Page – on ***ToonFinder***. Also, there are two sub-pages, Cartoonist and Webtoon page, and each of them shows individual cartoonist and Webtoon. All green-colored pages in **Figure 1** can be directed from all the other pages through toolbar. Features of main pages are described below.

- *Search Page:* A user can search Webtoons using search box for title, and check boxes for company, genre, and worth.
- *Cartoonists Page:* This page shows a list of cartoonists. Users can sort cartoonists by using a search box, and can access to individual cartoonist (Cartoonist page) for more informations.
- *Webtoons Page:* This page shows a list of Webtoons. Users can sort Webtoons by using a search box, access to individual Webtoon (Webtoon page), and add to 'Favorite Webtoons' in 'My Page'.
- *Related Contents Page:* This page shows related contents regarding Webtoons. Users can sort them using a search box, and access to the content for more information.
- *Login Page:* Users can create account or login. After login (or account has created), it leads to 'My page'. If users want to add favorite cartoonists, like and subscribe Webtoons without authentication, the page shows pop-up message that leads to the login page.

- *My Page:* This page shows information about the user. Users can also see their 'Subscribed Webtoons', 'Liked Webtoons' and 'Favorite Authors'. They can direct to individual cartoonists / webtoon or delete them. Logging out is available here.

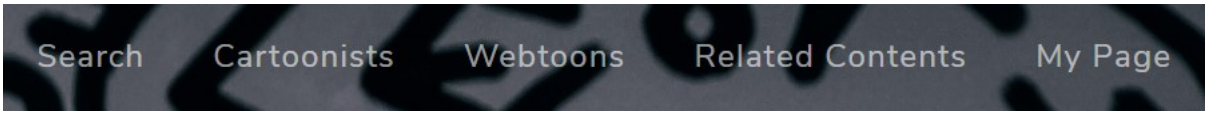


Figure 2. Toolbar Before Login

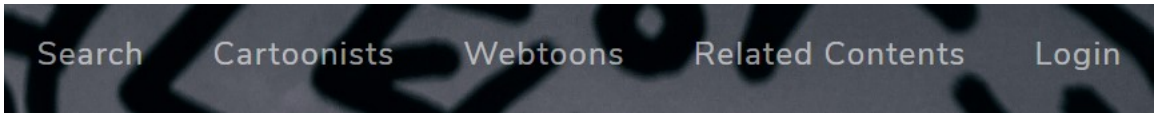


Figure 3. Toolbar After Login

If the user is not logged in, Login link is shown on toolbar (**Figure 2)**, and it changes to My Page link when user is logged in (**Figure 3**).

A.2 Major Functions

Overall, eight different major operations that can be done while doing the task.

- *Search Webtoons:* A user can search Webtoons by typing part of the Webtoon name or/and check boxes that satisfy the conditions (Company, Genre, and Worth).
- *Arrange Management:* A user can manage Cartoonists, Webtoons, and Related Contents to specific standard using a search box.
- *View Information:* The information for individual Webtoon or Cartoonist can be viewed.
- *Authentication:* A user can login and out, or create an account.
- *Preference Management:* Users preference information regarding subscribed / liked Webtoons and favorite cartoonists can be viewed, modified.

A.3 Definitions

These are the terms that are used often and unfamiliar to the users.

Webtoon: It is a cartoon that can be accessed through Web.

Webtoon Worth (in the main page): It is a value combined of 'Views' and 'Score' for the webtoon. If the Webtoon has higher views and scores, then the worth will be higher (Close to 'A').

Underlying Work: It is a content which Webtoon is based off.

Derived Work: It is a content which are made based off of a Webtoon.

A.4 Procedures

This is a tutorial for overall workflow: Installation / De-installation, accessing the system, data operations, and finishing the program.

A.4.1 Installation / De-installation

Installation: The access for repository is available from https://github.com/ after invitation. If you do not have an access, please download the Zip file from blackboard in C:\xampp\htdocs folder then unzip.

If the user wants to save the Zip file other folder then unzip, they can choose to create symbolic link in htdocs folder.

De-installation: Close the website, mysql and delete the project from where user initially installed.

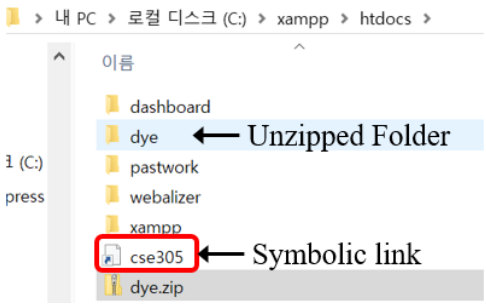


Figure 4. Installation

A.4.2 Accessing the System

1. Turn on XAMPP/MAMP Control Panel. If you do not have one, please refer to below site and download.

Downloading XAMPP for Windows:
https://www3.cs.stonybrook.edu/~alee/cse305/penv/xamppWindows.html

Downloading MAMP for Mac:
https://www3.cs.stonybrook.edu/~alee/cse305/penv/mampMac.html

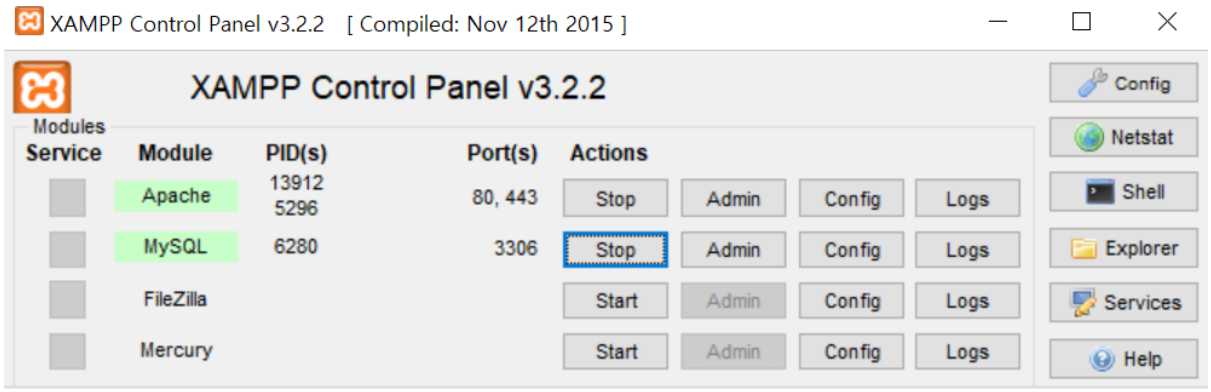


Figure 5. Starting XAMPP on Windows

2. Press ‘Start’ for Apache and MySQL. Then press Shell.

3. On the opened shell, type ‘mysql –u root’ to run the MariaDB.

4. Type ‘source PATH TO DATABASE/WebtoonDatabase.sql’ on the shell.
If you saved in htdocs, command will be ‘source C:/xampp/htdocs/dye/cse305-final-project-master/ToonFinder/WebtoonDatabase.sql’.

5. Open any browser and type the path for PHP website: ‘http://localhost/ PATH TO UNZIPPED FILE FROM HTDOCS OR SYMBOLIC PATH/dye/cse305-final-project-master/ToonFinder/searchEngine.php’.

If you saved in htdoes, the path will be ‘http://localhost/dye/cse305-final-project-master/ToonFinder/searchEngine.php’.

A.4.3 Website Operations

1. Search Page

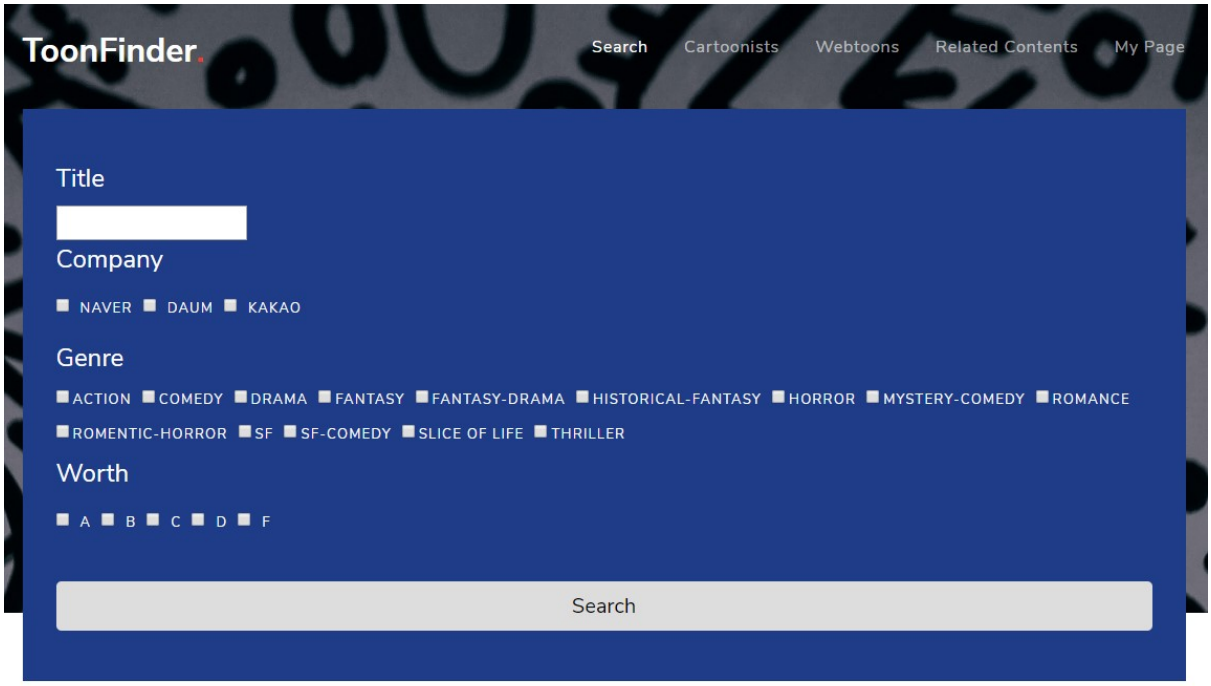


Figure 6. Start Page of Website

This will be the page you will see when you access the website.

In here, a user can search for Webtoons. Type ‘god’ and press ‘Search’ button.

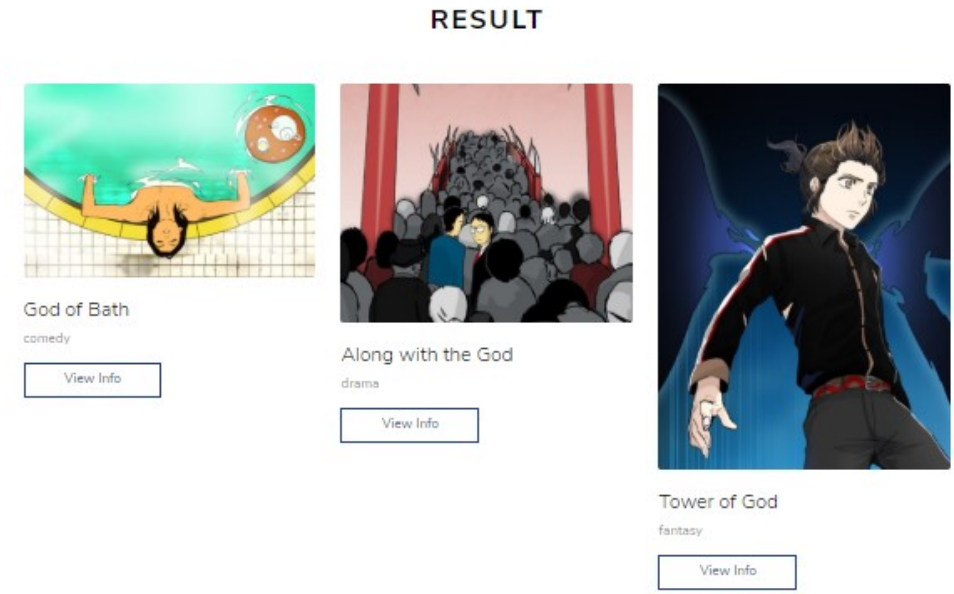


Figure 7. Result of searching by keyword ‘god’

Then, the ‘Result’ section shows Webtoons that has ‘god’ in the title. The user can also select check boxes for ‘Company’, ‘Genre’, and ‘Worth’ along with keyword from the title.

Ex) ‘Title: type life, Company: select naver and daum, Genre: select Slice of life, Worth: select none’

When the user click ‘View Info’ button, user will be able to see the information about the Webtoon.

2. Cartoonists Page

If the user clicks ‘Cartoonists’ on the toolbar, rows of cartoonists are shown. A user can click the selection box like **Figure 9** to filter cartoonists by ‘Both’: writer and drawer, ‘Story’: writer, and ‘Drawing’: drawer.

When the user clicks View Info, user can watch cartoonists information.

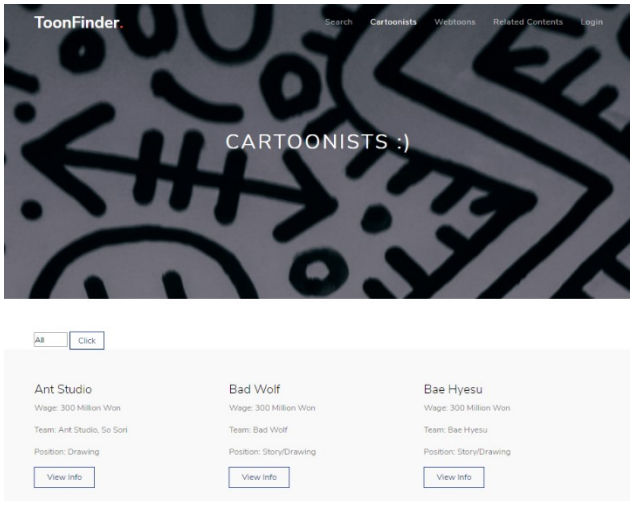


Figure 8. Cartoonists Page

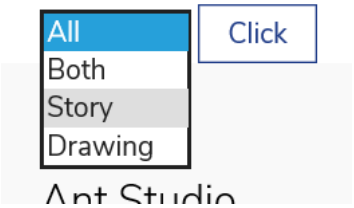


Figure 9. Selection Box

3. Webtoons Page

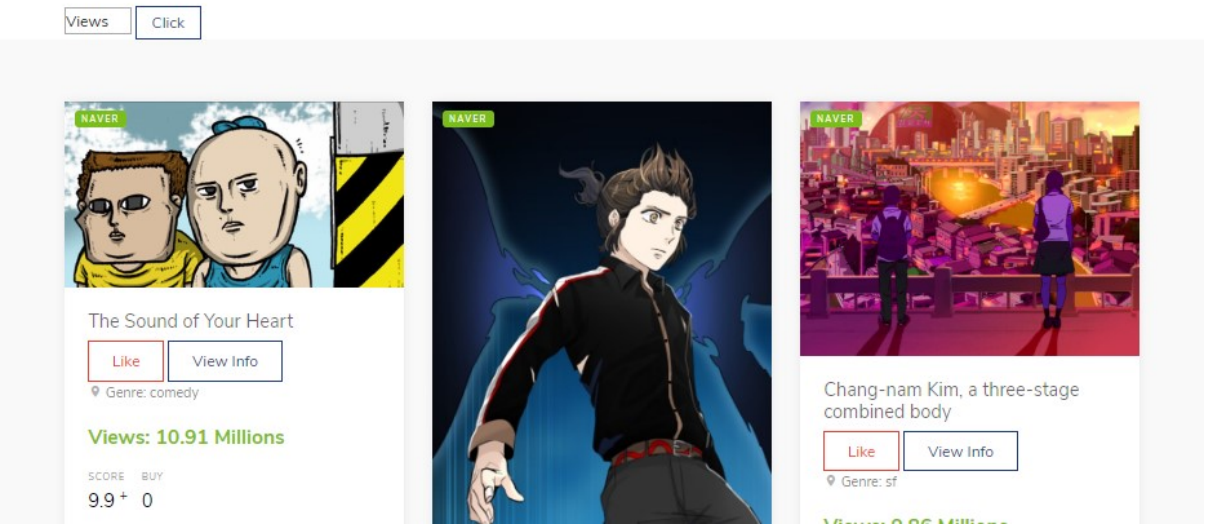


Figure 9. Part of Webtoons Page

A user can also navigate to Webtoons page from the toolbar. Similar to cartoonists page, the user will see entire Webtoons ordered by views (default). Using a search box, user can arrange Webtoons in alphabetic order, descending order of score and views.

4. Related Contents Page

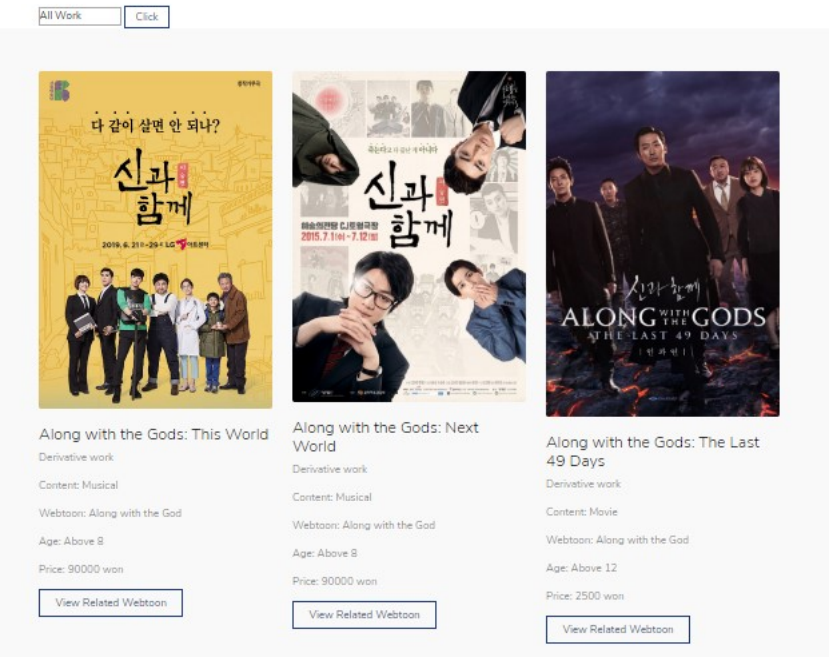


Figure 10. Part of Related Contents Page

When the user click ‘Related Contents’ on toolbar, user will be able to see all the related works of Webtoons. Like two previous pages, there is a search box that filters the work by ‘Underlying Work’, ‘Derivative Work’ and ‘Series’. When ‘View Related Webtoon’ is pressed, it shows ‘Webtoon page’ with related contents.

5. Login Page

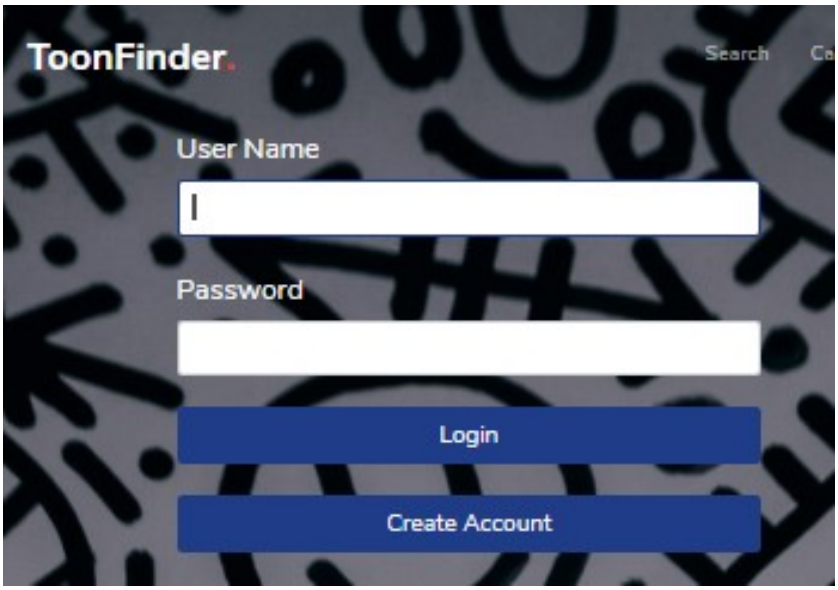


Figure 11. Login Page

There are two ways to direct to the login page. One way is to click ‘Login’ on toolbar, and another way is to click ‘Favorite’, ‘Like’, and ‘Subscribe’ on cartoonist or Webtoon page.

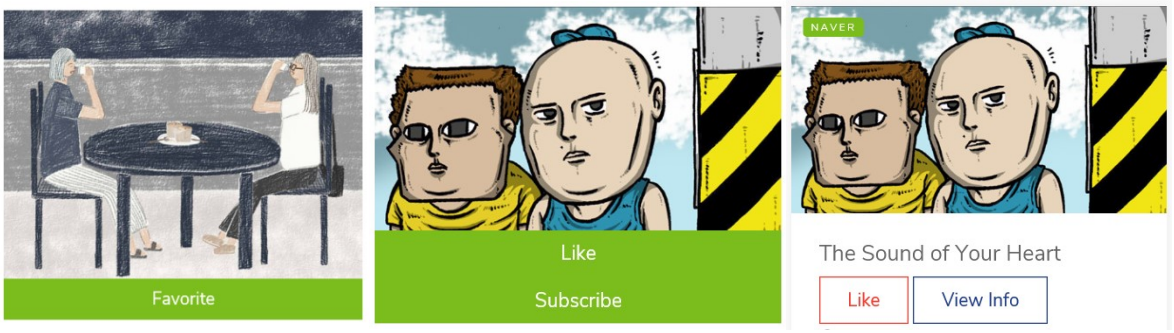


Figure 12. 'Favorite', 'Like', and 'Subscribe' buttons

The user can either create account or login using 'Reader' information on WebtoonDatabase.sql. If the user puts wrong id or password when login, there will be a pop-up message asking for retrial.

6. My Page

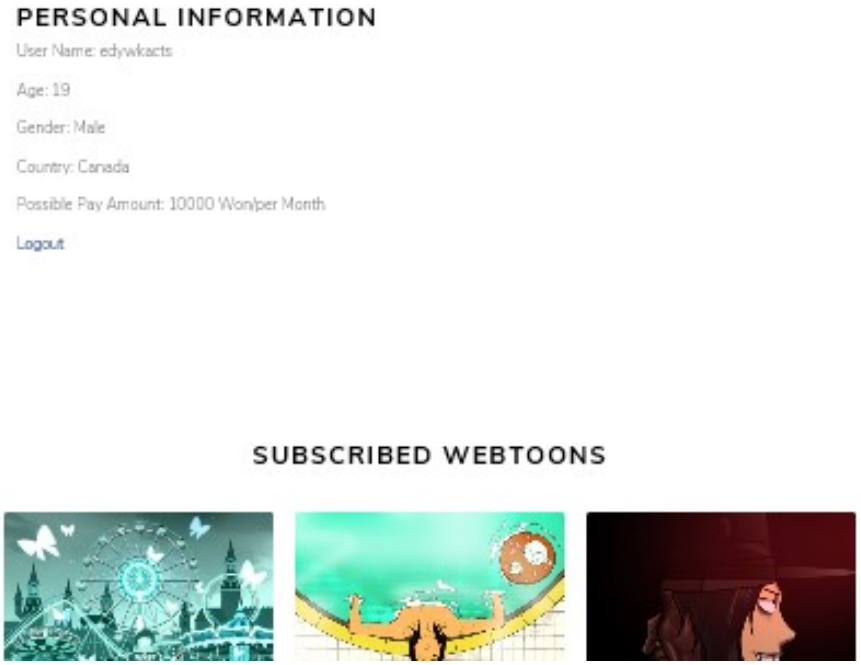


Figure 13. My Page

After logging in, the website automatically directs to MyPage. The user can view personal information, logout, subscribed and liked Webtoon and favorite cartoonists. Also, statistics for the subscription is shown.

A.4.4 Finishing the Program

A user can exit the website by clicking X button on the upper right corner.