CS157A Team 2
Final Report
A Book Reading and Review Web Application — "OpenBooks"

Dispoto, Brett Kamel, Adham Cai, Feiyu

December 1, 2019

1 Part I — Project Requirements

1.1 Project Description

1.1.1 Application Overview

Our team will be developing a database application where users can find free books and are able to download them via multiple formats and leave reviews on those books for other customers to see. The books that the user can see will be ranging from textbooks to novels. A user will be given the option to create an account, or if they already registered, login to their account. After this, the user will be able to search for a specific book, whether that is by the title of the book or its ISBN number. If the user does not know the specific book they want to find, they can search for a specific author, or genre. Searches will include various filtering options such as date published, publisher, book length, user favorites, etc. Users can sort search results by the title alphabetically, the author alphabetically, or by book rating.

In our application, users will be able to leave reviews on books that they have read, which will be seen by other users and possibly influence their decision on their book. These reviews will contain both a comment subsection and a star rating system. Users will be able to comment how they either liked or disliked the book and give a star rating from one to five. The average star rating and total number of reviews will be displayed next to the book. Users will also be able to share the book they like with others with a shareable link that they can distribute how they please.

1.1.2 Project Motivation

The motivation behind this project is that as college students, we find ourselves paying several hundreds of dollars on college textbooks alone every semester, only to have them sitting on a shelf afterwards not being opened. Those hundreds of dollars spent on semester long textbooks could have been better utilized for students on other needs such as groceries or rent. We want to make a website to give students that option, giving them a website where they can download their desired textbooks for free.

1.1.3 Project Goal

The Goal of this project is to provide a free and open source infrastructure where users can read books from the public domain. We wish to provide an easy-to-use ecosystem, which allows the user to read or review books with ease.

1.1.4 Benefits to Users

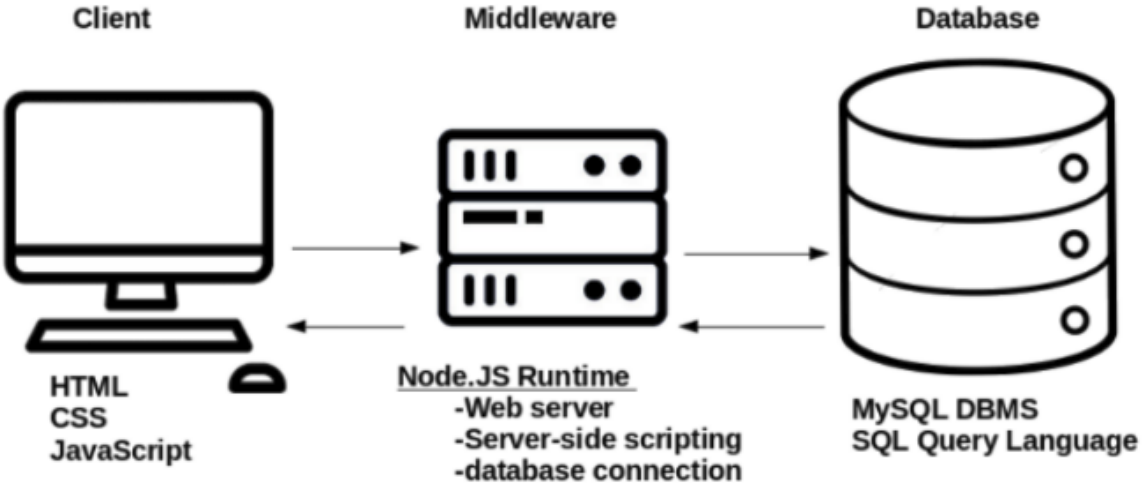
Users will have a community where they can **read and review books for free**. Right now, websites such as this exists, but they only serve one of the two desired functionalities. Websites like gutenberg.org give the users access to free book repositories, but do not allow for users to read and write reviews on books. Conversely, websites like bookpage.com provide reviews for books, but they do not provide free access to such books. Our web application

will take the best aspects of both these websites to provide the user an efficient solution for picking and reading online books for free.

1.1.5 Stakeholders and Importance

The stakeholders of our applications will be students who want to find free textbook alternative to the paid bookstore alternative, as well as casual and dedicated book readers who can find free books online and download them for instant reading. This application is important because it provides a streamlined way for book-readers to gain easy access to the books they want and get them in a timely manner.

1.2 System Environment



1.2.1 Presentation Layer

- HTML
  - Purpose: HTML is the markup language supported by all major web-browsers. It allows content to be presented to a readable manner to the end-user. We will take advantage of HTML formatting tools such as hyperlinks, tables, and lists. Further, we will be using HTML forms for user input of email, username, password, search boxes, and comments.
  - Version: HTML 5
- Cascading Style Sheets (CSS)
  - Purpose: CSS will be used to improve the user experience of our web application. CSS gives us the ability to make animations, colored content, as well as more control of the appearance of our content.
  - Version: CSS 3
  - Framework: Bootstrap

- JavaScript (Client Side):
  - Purpose: Provide interactivity such as collection of user input, improve visual responsiveness, and sending alerts to users.
  - Version: ECMA2016

1.2.2 Application Layer

- Web Server: Node.js HTTP Module

  -Version: 10.6.3 LTS

- Server Side Application Language: JavaScript

  -Purpose: Provide commination between the presentation layer and the database layer
  -Version: EMCAScript 2016

- NPM Package Manger:

  -Purpose: Provide easy managment of external Node.js libraries such as express js, sql module, connect module, and http module.

  -Version: 6.9.0

1.2.3 Data Layer

- This web application will require the use of a relational database management system (RDBMS), the specific RDBMS we will use is MySQL.
- We will take advantage of the SQL programming language for tasks such as data definition, manipulation, query, control,and transaction control.
- MySQL RDBMS will take care of concurrency control, and will maintain the ACID principle for our database.
- MySQL Version: 5.7.27
- Query Language: SQL

1.2.4 Hardware/ Software Used

- Client Software Requirements
  - A web browser supporting the following is required to run the web application:
    - ECMAScript2016
    - HTML5
    - CSS3
  - Since the 3-tier architecture will only be virtual, (no remote web server or DMBS), the client will be required to install the proper versions of Node.js as well as MySQL as specified.

  - MySQL and Node.js are available on many operating systems such as Linux, MacOS, Microsoft Windows, FreeBSD, and OpenBSD

- Client Hardware Requirements:

  - Any hardware with support by the above software will be sufficient to run the web application.

1.3 Functional Requirements

Because this application is to be accessed from a web browser, the **input** is a keyboard or mouse, while the **output** is stylized HTML to be displayed on a web browser.

1.3.1 Comprehensive List of Features

- Search Book:

  - Users can search for books that on the system database
  - Input: ISBN, author, or title
  - Output: list of books that match with the input information

- Order Search Results:

  - User can order the book search results. Default search behavior is based on the number of favorites a book has received
  - Input: relese date or number of favorites
  - Output: ordered list of book that has characteristic from input

- Filter Search Results:

  - User can filter the search reslts
  - Input: release date or genre
  - Output: filtered list of book that matched with characteristic from input

- Select Book:

  - Once user finds the desired book, they can select the book and view its profile.
  - Input: selecting single book
  - Output: book profile that includes book information and comments

- View Book Profile:

  A book's profile will consist of the following:

  - The title of the book,
  - The author,

  - the relase date,
  - the publisher,
  - the ISBN,
  - the reviews/comments left for the book,
  - the number of "favorites" the book has received

- Read/Download Book:
  - User can download the book and view it in their web browser.
  - Input: selecting single book
  - Ouput: reading page or download to local

- Add/delete book to/from favorites:
  - Use can add the list to their "favorites", indicating that they enjoyed the book
  - Input: selecting single book
  - Output: add/delete book to/from the favorited book list that can be viewed on user's profile

- View Favorites **requires login**:
  - User can browse their list of favorited books, with the same sorting/filtering mechanisms as noted above
  - Input: Click view favorite book list in user profile
  - Ouput: list of books that users marked as favorite

- Leave Book Review **-requires login**:
  - Users can leave comments on the profiles of certain books
  - Input: comments that relating to the selected book
  - Output: comments added to the book's repository

- Register as User:
  - User can register for an account if they want to have the ability to comment and favorite books
  - Input: user information includes username, password, email
  - Output: user' account created

- Login as User:
  - Once registered, users will have the ability to login using the credentials they have provided during registration
  - Input: email and password

  - Ouput: login successfully give the access to login user's profile

- Go to home page:
  - The homepage will have information about the website and recent news regarding the website
  - Input: clicked website logo on navigator bar
  - Output: redirect user to homepage

- View profile **-requires login**:
  - All profiles are anonymous because this is **not a social network**. Users are not allowed to change their username.
  - Input: successfully login
  - Output: Profile page includes username and email information, favorites book list

- Log out:
  - login user exits login condition, become vistor to the website
  - Input: login user clicks "Log Out" button
  - Output: becomes vistor and no longer has access to profile page, return to homepage if logout successfully

1.4 Non-Functional Issues

1.4.1 Graphical User Interface

The Graphical interface of the system will have the following qualities: attractiveness, usability, and responsiveness

Attractiveness The GUI will have a color palate which makes the system attractive to users. We will use high quality fonts and images where applicable. The GUI as a whole will have a coherent and consistent theme.

Usability The GUI will be intuitive. Users will not have to read documentation on how to use the system in order to use it. Options available to the user will be kept to a minimum as to encourage simplicity.

Responsiveness Users will be aware that their requests have been recorded. For example, a user clicks a button, there will be an indication that the button has been successfully clicked, such as a change in color. We will keep bloat to a minimum, such as animations and videos, as to encourage responsiveness for older hardware.

1.4.2 Security

User Security To protect user data and privacy, users entities will have the following attributes:

- A unique username
- A password, required to follow the minimum standards set by the system: which is a minimum of 8 characters, 1 uppercase character, 1 lowercase character, 1 numerical character, and 1 special character

SQL Injection Protection To protect form SQL injections, we will take precautions to inspect all user input and escape any characters which could breach security.

Encryption Any sensitive information collected from users will be encrypted fields in the relational database. All passwords will be hashed and we will take advantage of MySQL's build in password functionality.

Anonymity When a user makes a comment, **only their username will show up**, not their real name. Anonymity protects users from being identified against their will, and prevents personal attacks or blackmail. Users on this web application should be allowed to enjoy and review any books they like, and not have to be worried about ramifications.

1.4.3 Concurrency Control

The Relational Database Management System (RDBMS), MySQL, will take care of all needs relating to concurrency control. From the user's perspective, all transactions can be made concurrently.

1.4.4 Access Control

Types of Users

- **User** this type of accounts are registered by user themselves

  Users will only have read access to the following:

  - Their own account profile
  - Book repositories with open access

  In addition, users will have write access to the following:

  - Their own account profile information
  - Their favorites list
  - Reviews left for books under restrictions

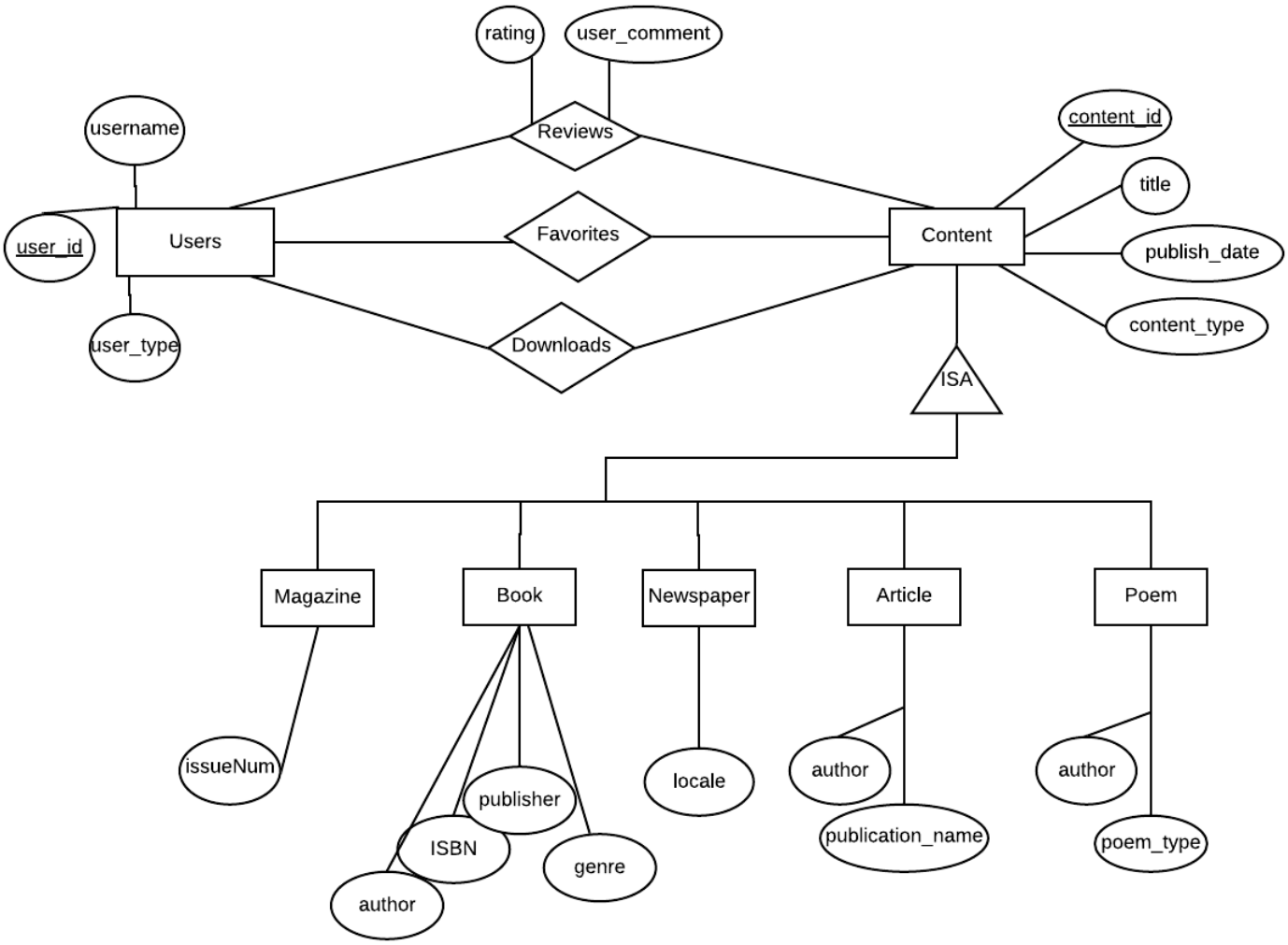
- **Database Administrator** this type of accounts are created assigned by the system owners

  Database Administrators have access **all user functionality** plus extra permissions to data which is not available to the general public. DB admins have read/write access to the following:

  – Read user profile
  – Modify user profile including password with authentication from user
  – Add and delete book repositories
  – Delete comments by users that are not under restrictions

2 Part II — Database Project Deisgn

2.1 Updated Entity-Relationship Diagram



2.2 Database Schema

- Users(user_id, username, use_type)
- Reviews(user_id, content_id, rating, user_comment)
- Favorites(user_id, content_id)
- Downloads(user_id, content_id)
- Content(content_id, title, publish_date, content_type)
- Article(content_id, author, publication_name)
- Magazine(content_id, issueNum)
- Book(content_id, ISBN, author, genre, publisher)
- Newspaper(content_id, locale)
- Poem(content_id, poem_type, author)

2.3 Entities

2.3.1 Users

Users are defined as those who have a valid account login for the website.
The attributes for Users are as follows:

- user_type: A user can be of two types, admin or regular. Admins have special privileges
- username: The unique username by which a user logs into the web application
- user_id: the unique identifier by which a user's information is tracked

2.3.2 Content

Content comes in many forms. There are four subclasses of Content, but it possible that there exist an item that is only classifiable as Content and none of its subclasses.
The attributes for Content are as follows:

- content_id: The unique content identifier
- title: The name/title of the content
- publish_date: The date the content was published, if available
- content_type: the type of content, if applicable.

2.3.3 Books

Books are a special case of Content. They have special attributes which sets them apart; however, books inherit the primary key from content. In addition, books have the following attributes:

- ISBN: the unique book ISBN number
- publisher: the publisher of the book
- author: The author of the book
- genre: the genre of the book

2.3.4 Magazines

Magazines are a special case of Content. They have special attributes which sets them apart; however, magazines inherit the primary key from content. In addition, magazines have the following attributes:

- issueNum: the issue number of the magazine.

2.3.5 Newspapers

Newspapers are a special case of Content. They have special attributes which sets them apart; however, Newspapers inherit the primary key from content. In addition, magazines have the following attributes:

- locale: the locality of a Newspaper, if available.

2.3.6 Articles

Articles are standalone pieces of writing which may, in some cases, be significant enough to publish on the web application as it's own piece of media. Articles inherit the primary key from Content but have additional relationships, as described in the relationships subsection. Articles also have the following additional attributes:

- author: the author of the article
- publication_name: the name of the publication where the article is from.

2.3.7 Poems

Poems are a specialized form of content, usually artistic in nature. Poems inherit the primary key from Content but have additional attributes, as described in the relationships subsection. Articles also have the following additional attributes:

- author: the author of the poem (if available)
- poem_type: the type of poem (ballad, epic, prose, haiku, etc.)

2.4 Relationships

2.4.1 User Reviews Content

A user (admin or regular) is the author of zero or many reviews on zero or many pieces of content.
The attributes for Reviews are as follows:

- rating: a star rating between 1 and 5
- user_comment: a comment about the piece of content (optional)

2.4.2 User Favorites Content

A user can favorite content, which allows them to keep track of their favorite items. Content can be favorited by zero or many users. Users can favorite zero or many pieces of content.

2.4.3 User Downloads Content

We would like to keep track of user's download history. A user can download zero or many pieces of content, and a piece of content may be downloaded by zero or many users.

2.5 Boyce Codd Normal Form (BCNF) Proofs

2.5.1 Users — BCNF Proof

Claim: the Users relation is in BCNF

Proof. The relational schema for for the **Users** table is:
$Users(user\_id, user\_type, username)$
The minimal cover of the functional dependencies for Users is:
$user\_id \rightarrow user\_type, username$
Now, we have to check if $user\_id$ is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $user\_id$, we find that...
$user\_id+ = \{user\_id, username, user\_type\}$
Which means that $user\_id$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Users \in BCNF$ □

2.5.2 Reviews — BCNF Proof

Claim: the Reviews relation is in BCNF

Proof. The relational schema for for the **Reviews** table is:
$Reviews(user\_id, content\_id, user\_comment, rating)$
The minimal cover of the functional dependencies for Reviews is:
$user\_id, content\_id \rightarrow user\_comment, rating$

Now, we have to check if the set of attributes $\{user\_id, content\_id\}$ is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{user\_id, content\_id\}$, we find that...
$\{user\_id, content\_id\}+ = \{user\_id, content\_id, user\_comment, rating\}$
Which means that $\{user\_id, content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Reviews \in BCNF$ □

2.5.3 Favorites — BCNF Proof

Claim: the Favorites relation is in BCNF

Proof. The relational schema for for the **Favorites** table is:
$Favorites(user\_id, content\_id)$
There are no non-trival Functional Dependencies in this relation, so it is true that for all non-trivial FD's ∈ Favorites (none), the left side of the FD is a superkey.
$\therefore Favorites \in BCNF$

□

2.5.4 Downloads — BCNF Proof

Claim: the Downloads relation is in BCNF

Proof. The relational schema for for the **Downloads** table is:
$Downloads(user\_id, content\_id)$
There are no non-trival Functional Dependencies in this relation, so it is true that for all non-trivial FD's ∈ Downloads (none), the left side of the FD is a superkey.
$\therefore Downloads \in BCNF$

□

2.5.5 Poem — BCNF Proof

Proof. The relational schema for for the **Poem** table is:
$Poem(content\_id, poem\_type, author)$
The minimal cover of the functional dependencies is:
$content\_id \rightarrow poem\_type, rating$
Now, we have to check if the attribute *content_id* is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{content\_id\}$, we find that...
$\{content\_id\}+ = \{content\_id, author, poem\_type\}$
Which means that $\{content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Poem \in BCNF$ □

2.5.6 Content — BCNF Proof

Proof. The relational schema for for the **Content** table is:
$Content(content\_id, content\_type, title, date\_published)$
The minimal cover of the functional dependencies is:
$content\_id \rightarrow content\_type, title, date\_published$
Now, we have to check if the attribute *content_id* is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{content\_id\}$, we find that...
$\{content\_id\}+ = \{content\_id, content\_type, title, date\_published\}$
Which means that $\{content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Content \in BCNF$ □

2.5.7 Magazine — BCNF Proof

Proof. The relational schema for for the **Magazine** table is:
$Magazine(content\_id, issueNum)$
The minimal cover of the functional dependencies is:
$content\_id \rightarrow issueNum$
Now, we have to check if the attribute *content_id* is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{content\_id\}$, we find that...
$\{content\_id\}+ = \{content\_id, issueNum\}$
Which means that $\{content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Magazine \in BCNF$

□

2.5.8 Article — BCNF Proof

Proof. The relational schema for for the **Artcile** table is:
$Article(content\_id, author, publication\_name)$
The minimal cover of the functional dependencies is:
$content\_id \rightarrow author, publication\_name$
Now, we have to check if the attribute *content_id* is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{content\_id\}$, we find that...
$\{content\_id\}+ = \{content\_id, author, publication\_name\}$
Which means that $\{content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Article \in BCNF$

□

2.5.9 Newspaper — BCNF Proof

Proof. The relational schema for for the **Article** table is:
$Newspaper(content\_id, locale)$
The minimal cover of the functional dependencies is:
$content\_id \rightarrow locale$
Now, we have to check if the attribute *content_id* is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{content\_id\}$, we find that...
$\{content\_id\}+ = \{content\_id, locale\}$
Which means that $\{content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Newspaper \in BCNF$

□

2.5.10 Book — BCNF Proof

Proof. The relational schema for for the **Book** table is:
$Book(content\_id, ISBN)$
The minimal cover of the functional dependencies is:
$content\_id \rightarrow ISBN, author, publisher, genre$
Now, we have to check if the attribute *content_id* is a superkey, because it is the left side of our only functional dependency.
Computing the closure of $\{content\_id\}$, we find that...
$\{content\_id\}+ = \{content\_id, ISBN, author, publisher, genre\}$
Which means that $\{content\_id\}$ is a superkey, so the only non-trivial functional dependency in the minimal cover of FDs for this relation does **not** violate BCNF
$\therefore Book \in BCNF$

□

2.6 Database Table Instances

2.6.1 Users

user_id	username	user_type
1	bobby12	regular
10	AI	regular
11	boot	regular
12	goolger	regular
13	password	regular
14	enterpirseUser	regular
15	dispo341	regular
2	tim12	admin
3	joe01	admin
4	tim	admin
5	joseph	regular
6	mikeWU	regular
7	timard	regular
8	WUmiek	regular
9	jame	regular

2.6.2 Reviews

user_id	content_id	rating	user_comment
1	23	3	fun
1	28	1	good book
1	46	3	nicee
1	49	5	decent
1	59	1	NULL
1	60	2	NULL
1	61	1	NULL
1	66	1	NULL
10	18	1	NULL
2	52	4	bad book
5	22	2	boring
5	24	1	bad
5	73	4	NULL
7	61	2	NULL
8	55	7	I love this book so much it is so good best book ever!
8	62	2	NULL

2.6.3 Favorites

user_id	content_id
1	1
1	4
1	7
1	12
1	17
10	4
11	4
12	4
12	6
13	14
14	4
19	4
5	2
6	4
8	4

2.6.4 Downloads

user_id	content_id
1	1
1	5
1	7
1	12
1	17
10	4
11	4
12	4
12	6
13	14
14	4
19	4
5	2
6	4
8	4

2.6.5 Poem

content_id	author	poem_type
61	Georgre Bush	Haiku
62	James Jones	Haiku
63	James Reds	Ballad
64	John Renyolds	Ballad
65	Patrick Renyolds	Ballad
66	Harry Renyolds	Ballad
67	Daisy Renyolds	Haiku
68	Patrick James	Haiku
69	Patrick Bush	Ode
70	Patrick Renyolds	Ballad
71	James Bush	Haiku
72	Tomard Renyolds	Love
73	Gustavo Renyolds	Love
74	Frank Renyolds	Ballad
75	J. Renyolds	Ballad

2.6.6 Content

content_id	title	publish_date	content_type
1	Snowbaording Magazine	2018-08-01	magazine
2	Snowbaording Magazine	2018-09-01	magazine
3	Snowbaording Magazine	2018-10-01	magazine
4	Snowbaording Magazine	2018-11-01	magazine
5	Snowbaording Magazine	2018-12-01	magazine
6	Snowbaording Magazine	2019-01-01	magazine
7	Snowbaording Magazine	2019-02-01	magazine
8	Snowbaording Magazine	2018-03-01	magazine
9	Snowbaording Magazine	2019-04-01	magazine
10	Snowbaording Magazine	2019-05-01	magazine
11	Snowbaording Magazine	2019-06-01	magazine
12	Snowbaording Magazine	2019-07-01	magazine
13	Snowbaording Magazine	2019-08-01	magazine
14	Snowbaording Magazine	2019-09-01	magazine
15	Snowbaording Magazine	2019-10-01	magazine
16	In the Electric Mist	2019-02-01	book
17	Babylon 5	2018-04-01	book
18	Mortuary	2017-11-11	book
19	Counterfeit Coin	2018-06-05	book
20	Ah, Wilderness!	2018-10-13	book
21	Independencia	2018-10-24	book
22	Lost Son, The	2018-11-20	book
23	Wild Child	2019-07-14	book
24	Little Big Soldier	2018-11-18	book
25	Medea	2017-12-16	book
26	One Fine Spring Day	2019-07-05	book
27	Galaxina	2018-12-18	book
28	Godzilla 2000	2017-11-09	book
29	Free Soul, A	2019-08-21	book
30	Attack on the Iron Coast	2019-04-30	book
31	New York Time	2018-06-28	newspaper
32	New York Time	2018-04-10	newspaper
33	New York Time	2018-10-04	newspaper
34	New York Time	2018-08-30	newspaper
35	New York Time	2018-05-29	newspaper
36	New York Time	2019-09-05	newspaper
37	New York Time	2019-08-16	newspaper
38	New York Time	2018-12-01	newspaper
39	New York Time	2018-07-24	newspaper
40	New York Time	2018-09-07	newspaper
41	New York Time	2018-09-17	newspaper
42	New York Time	2019-01-18	newspaper
43	New York Time	2019-05-13	newspaper
44	New York Time	2019-10-02	newspaper
45	New York Time	2018-04-30	newspaper
46	Why WWII was Bad	1994-12-11	article
47	WWI -- The Good War	1994-12-10	article
48	Vietnam War: 20 Year Later	1994-12-09	article
49	Cold War	1994-12-01	article
50	The Good War	1994-12-01	article
51	The Bad War	1994-12-01	article
52	10 Reasons Why Poems are Good	1994-12-05	article
53	5 Reasons to use NoSQL	1994-12-05	article
54	7 Reaons to use MySQL	1994-12-05	article
55	12 Reasons to Normalize Database	1994-12-05	article
56	Neural Nets for Preschoolers	1994-12-11	article
57	How to Perform Surgery	1994-12-11	article
58	How to write ER Diagrams	1994-12-01	article
59	SJSU Announces New Gym	1994-12-01	article
60	SJSU Power Outage	1994-12-12	article
61	A Walk	1666-01-11	poem
62	A Run	1266-01-11	poem
63	A Jog	1631-12-10	poem
64	A Dog	1435-10-10	poem
65	A CAT	1123-10-10	poem
66	A Harry Potter	1823-01-09	poem
67	Table	1923-01-09	poem
68	I love Databases	1941-11-09	poem
69	Best Poem Ever	1921-01-09	poem
70	Consumerism	1941-01-09	poem
71	Capitalism	1952-01-09	poem
72	Canda	2035-11-09	poem
73	Food	1635-11-09	poem
74	A Long Run in the Park	1835-04-12	poem
75	A Ballad to James	1935-11-12	poem

2.6.7 Magazine

content_id	issueNum
1	291
2	292
3	293
4	294
5	295
6	296
7	297
8	298
9	299
10	300
11	301
12	302
13	303
14	304
15	305

2.6.8 Article

content_id	author	publication_name
46	Patrick	New York Times
47	Jones	New York Times
48	Tim Cook	New York Times
49	Steve Job	New York Times
50	Bill Gate	New York Times
51	Tom Gate	New York Times
52	Michael Gate	New York Times
53	John Lennon	New York Times
54	John Lennon	New York Times
55	Timard Jones	New York Times
56	Mike Wu	New York Times
57	Mike Wu	New York Times
58	Harry Potter	New York Times
59	Harry Potter	New York Times
60	James Potter	New York Times

2.6.9 Newspaper

content_id	locale
31	New York City, NY
32	New York City, NY
33	New York City, NY
34	New York City, NY
35	New York City, NY
36	New York City, NY
37	New York City, NY
38	New York City, NY
39	New York City, NY
40	New York City, NY
41	New York City, NY
42	New York City, NY
43	New York City, NY
44	New York City, NY
45	New York City, NY

2.6.10 Book

content_id	ISBN	author	publisher	genre
16	304555343-5	Murdoch Cave	Leela Buterton	Comedy
17	659196532-7	Goldy Aimeric	Ali Gaitskell	Comedy
18	491743522-6	Jelene Fancet	Linzy Shelf	Comedy
19	346181073-8	Paulie Ouldcott	Adrianna Gresch	Adventure
20	868282781-6	Taber Deguara	Shanta Daouze	Horror
21	109670642-3	Clareta O'Carran	Winifield	Sci-Fi
22	239189700-6	Osbourne Kingsly	Antonella Fancett	Crime
23	671528801-1	Stevena Gotling	Mabelle MacFaul	Drama
24	583092606-7	Debi Vala	Madelon Croux	Drama
25	674056408-6	Merilee Blackall	Oates Raspison	Crime
26	273454549-7	Wanda Morbey	Shawn Kosel	Crime
27	544491903-6	Clyde Marginson	Jocko Connors	Comedy
28	469558624-X	Magdaia Killelay	Prudence Rudsdell	Crime
29	541188048-3	Philippa Frays	Noel Doerrling	Action
30	133956813-6	Sharron Earley	Roze Sterzaker	Comedy

3 Implementation

3.1 Implementation Overview

On the server, our project uses the Node.js runtime environment and express.js framework for routing and http requests/ responses. Each HTML page is implemented using EJS (Embedded JavaScript) as a templating engine.
Here is a general overview of how the application works:

- The server is initialized, this includes:
  - Claiming the 8080 port of local host using express.js — app.listen()
  - Establishing a database connection, using the Node.js MySQL module — mysql.createConnection(JSON obj)
  - Defining where all static content is located (files, photos, etc), using express.js — express.static(/path/to/static/content)
  - Initializing all routes/ modules, such as content, auth, and search
- HTTP requests from the client trigger various database operations such as INSERT and SELECT, which are executing using the MySQL module function database.query(string sql_code, function callback). Database.query() is an asynchronous function, meaning that there is no guarantee that it will be executed immediately upon calling it. For this reason, we often must wrap our server HTTP responses in the callback function, assuming we want to use data in our webpages which we have just queried.
- Dynamic content queried from the database is then served using EJS templates. EJS allows us to embed server-like JavaScript code into our HTML code. The templating engine then truncates the js/HTML hybrid into proper HTML code for the browser to parse.

4 Project Conclusion

4.1 Lessons from Project — Teammate Statements

4.1.1 Brett Dispoto

Prior to working on OpenBooks, I had no knowledge of practical database design/ application development. I now understand the following:

- The importance of an E/R design and analysis **prior** to implementing the relational schemas in the DBMS. Because our group put a large amount of work into our relational schema design, the normalization process into BCNF was extremely minimal.
- How to perform database integration on a virtualised 3-tier architecture. This includes
  - How to install and configure a local MySQL server, allowing remote access.
  - How to use Node.js to perform MySQL database queries, then parsing the JSON output of those queries.
  - How to install and configure the MySQL Node.js module.
- How to use embedded javascript (EJS) to display dynamically generated content using data from a MySQL database.
- How to write SQL (DDL/DML) scripts for automated database setup.
- How to effectively work as a team, where the responsibilities include writing functional specifications, designing database schemas, and implementing a database application.

4.1.2 Feiyu Cai

4.1.3 Adham Kamel

4.2 Future Improvements — Discussion

A *wishlist* of future improvements for this project would be as follows:

1. Deploying the database/ middleware on a remote server, eliminating the virtualised 3-tier architecture we are currently running on in favor of a proper client-server-database architecture. This would allow OpenBooks to be accessed on the web, rather than having to manually install, configure, and deploy on the client's local machine.
2. Adding advertisements — the data stored in our tables would be valuable to advertising firms running targeted advertisements. Information such as *"John Doe likes to read the new york times on Tuesday afternoons"* is very useful to such firms, and the revenue gained could be used to help OpenBooks achieve goal (1) above.
3. Add more tables to track user data such as average session time.
4. Add more features / customizability user profiles — such as profile photos, following users, view other user's profile.

5. Email notification system — When a new book is released by an author of which a user has favorited their book, the user should receive an email notification.
6. Look into the possibility of having tables be 4NF/5NF.
7. A terms of service, copyright licence, and privacy policy.