**Phase 2**

**Conceptual Database Design**

**TEAM 3M: My Mongo Maria**

**TEAM MEMBER:**

**Eiji Iriyama**

**Kim Seong Kyu**

**Weidi Shen**

**Weitong Wang**

**Table of Contents**

**1. Project Description...…………………………………………………………………………..5**

**2. Figures and Tables…..…………………………………………………………………………6**

1. **2.1 Requirement Analysis……………………………………………………………….6**
2. **2.2 Conceptual Design…………………………………………………………………...8**
   1. **Figure 1 Member**
   2. **Figure 2 Address**
   3. **Figure 3 Reserve**
   4. **Figure 4 Rent**
   5. **Figure 5 Book\_Rating**
   6. **Figure 6 Book**
   7. **Figure 7 Admin**
   8. **Figure 8 User\_Rating**
   9. **Figure 9 Delete**
   10. **Figure 10 Deliver**
   11. **Figure 11 Have**
   12. **Figure 12 Leave**
   13. **Figure 13 Lives**
   14. **Figure 14 Make\_Contains**
   15. **Figure 15 Manages**
   16. **Figure 16 Place\_Order**
   17. **Figure 17 Overall ER diagram**
3. **2.3 Features Design Details………………………………………………………….…18**

**3. Associated Attribute……………………………………………………………………….…19**

**3.1 Baseline Features……………………………………………………………………..19**

1. **3.1.1 Books**
2. **3.1.2 Categories**
3. **3.1.3 Suppliers**
4. **3.1.4 Searching**
5. **3.1.5 Registered Users**
6. **3.1.6 Rating**
7. **3.1.7 User-rating**
8. **3.1.8 Transaction History**
9. **3.1.9 Reservation (Biding)**
10. **3.1.10 Rent (Sale)**
11. **3.1.11 Delivery**
12. **3.1.12Reports to Telemarketers**

**3.2 Extended Features………………………………………………………………………21**

**a. 3.2.1 Blacklist**

**4. Relation Schema………………………………………………………………………...…….22**

**5. Functional Dependency………………………………………………………………………25**

**a. 5.1 Functional Dependency……………………………………………………………25**

**b. 5.2 normalization………………………………………………………………………26**

**6. Summary………………………………………………………………………………………28**

**7. Appendices……………………………………………………………………………………29**

**Appendix A Design Approach…………………………………………………………29**

**Appendix B Project Plan……………………………………………………………….30**

**Appendix C Progress Report…………………………………………………………...31**

1. **Progress Report (3)**
2. **Progress Report (2)**
3. **Progress Report (1)**

**Section 1 Project Description**

The project is to implement an online rental system for a library. The core attribute we would like to strive for is user-friendliness. We would like to pattern the features from app stores – using categories to narrow the search down and use lineups in each section to increase the vividness of the application. A general searching feature would of course also be supported. Users can search by key words, author, ISBN, publisher, or any combinations of the search categories. We would also like to implement a ranking/rating system for the popularity of the books available for renting. In general, we want this implementation to be more of a casual manner in presentation, rather than a serious academic setting.

The biggest motivation for this project is because of the common characteristic of online library systems lacking vividness. As the development of electronic readings growing rapidly, the paper-based book reading is showing a trend of decreasing in popularity. One of the major reasons that people in general are reading fewer books is that paper based reading generally gives a feeling of having to be serious, that is why we want to break this stereotype by trying to make the library’s system more of a “fun” to use. Through observation and brainstorm amongst our team members, patterning the form of an app store is a practical way to achieve our goal. It is the simplest template that is both vivid and supports all the functionalities the system requires.

In this report, we are going to make some analysis for the requirements discussing what kind of functionality is supported and how to implement them. The requirement analysis is followed by the designs for entities, which will support the implementation. We would also have a deeper discussion on why the attributes are designed this way. Last but not least, a brief summary will be also included in the end of the report.

**Section 2 Figures and Tables**

**2.1 Requirement Analysis**:

**User** Perspective:

|  |  |
| --- | --- |
| **User** | **Solution** |
| Can we easily check if a particular book can be rented? | Convenient web-based search system  The quantity of books can be rented or reserved? |
| The book I want is always on for rent! | Reserve System Construction  Number of people on the reservation list and schedule the arrival date of their reserved books. |
| I want to know the popular books | Top book listings according to most rented and user ratings  We will also create the Weekly/Monthly pop list based on the current browsing/searching record. |
| I want brief summary of the book | Plot, URL to publisher’s site. Also, you can jump into user’s rating. |
| How easily can I get the book? | You can pick it up at pickup station of library which hold the book by your reservation confirmation. Or user could select the delivery option to the user’s address. However, you have to pay for the delivery fee. |

**Admin Perspective:**

|  |  |
| --- | --- |
| **Admin** | **Solution** |
| Can we provide easy rental service to users? | Convenient web-based search system |
| Can we easily keep on track on rented books? | Configure rental period based on Date |
| Can we easily manage user information? | Blacklists, Good users, Browsing history |

**2.2 Conceptual Design**

|  |  |
| --- | --- |
| 1. **MEMBER ENTITY** | |
| Function | Shows users information |
| ER Model | * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **ADDRESS ENTITY** | |
| Function | Contains members address |
| ER Model | * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **RESERVE ENTITY** | |
| Function | Contains reservation information |
| ER Model | * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **RENT ENTITY** | |
| Function | Contains rental information |
| ER Model | * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **BOOK RATING ENTITY** | |
| Function | Contains information about ratings for a book |
| ER Model | **rating.png**   * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **BOOK ENTITY** | |
| Function | Contains all formation about the book |
| ER Model | * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **ADMIN ENTITY** | |
| Function | The administrator information |
| ER Model | **admin.png**   * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **USER RATING ENTITY** | |
| Function | Contains information for user’s rating |
| ER Model | * Strong Entity Type * Each Attribute is simple attributes * Each Attribute is one valued attributes |

|  |  |
| --- | --- |
| 1. **DELETE RELATIONSHIP** | |
| Function | Deletes user id from reservation list |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\delete.png   * Delete relationship * When reader succeeds on renting book, the user id is deleted from the reservation list. |

|  |  |
| --- | --- |
| 1. **DELIVER RELATIONSHIP** | |
| Function | Delivers book to the corresponding address |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deliver.png   * Deliver relationship * Each reservation has a corresponding address, which will deliver. |

|  |  |
| --- | --- |
| 1. **HAVE RELATIONSHIP** | |
| Function | Possessing relationship |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\have.png   * Have relationship * Ratings are related to rating. |

|  |  |
| --- | --- |
| 1. **LEAVE RELATIONSHIP** | |
| Function | For users to leave comment |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\leaves.png   * Leave relationship * User leaves comments and rates the book. |

|  |  |
| --- | --- |
| 1. **LIVES\_AT RELATIONSHIP** | |
| Function | Relates addresses to members |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lives_at.png   * Lives\_at relationship * A member lives at a corresponding address. |

|  |  |
| --- | --- |
| 1. **MAKES\_CONTAINS RELATIONSHIP** | |
| Function | Member rents a book |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\makes_contains.png   * Make\_Contains relationship * A member makes rents on the certain book |

|  |  |
| --- | --- |
| 1. **MANAGES RELATIONSHIP** | |
| Function | Admin manages the major entities |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\manage.png   * Manages relationship * Admin is the master who can manage and modify the major entities |

|  |  |
| --- | --- |
| 1. **PLACE\_ORDER RELATIONSHIP** | |
| Function | Member directly orders a book |
| ER Model | C:\Users\Weitong\AppData\Local\Microsoft\Windows\INetCache\Content.Word\place_order.png   * Place\_order relationship * If there is no member on reservation list, direct order is placed. |

|  |  |
| --- | --- |
| 1. **OVERALL ER DIAGRAM** | |
| Function | Overall ER diagram for our project |
| ER Model | * Admin will manage each Entity * Rating Entity has a total participation with Book Entity * User\_rating Entity has a total participation with Rent Entity |

**2.3 Feature design Details:**

1. Browsing & searching are combined because we create the category tree and list of items (which is identifier in our case) as the attribute of our Book Entity.
2. We set all the entity as strong type, because we don’t have any entities which really depends on others. For our case, we used to consider the rating entity as a weak type. In common sense, as the certain book is no longer in our rental list, the comments for this book might be gone as well. However, by our search about relative website, we found that we could still read about the comments for a book even if the book is no more available to be rented. At last, we decide to keep rating entity as a strong type. Just in case, if users need those unavailable somewhere else, but they can also learn something about the book from others’ comments.
3. In order to gain fully control of our system, we also create an Admin Entity to manage other major entities, such as Member Entities, Book Entities, Reserve Entities, and Rent Entities.
4. In the ER diagram design, we create a Blacklist Entity for our extend feature. The Admin Entity will manager this entity depending on the rating of users. Once the user returns a damaged book. The Admin Entity will drop this user from Member Entity into the Blacklist Entity. Since we will apply this approach in the actual implementation, we will not include any blacklist-related ER diagram in our report.

**Section 3 Associated Attribute**

**3.1 Baseline features:**

**3.1.1 Books -** The focus here is the books for rental. A book is rented by users via web page. The sources of the book is the library. A unique identifier is assigned to a book when it is in available to be rented. This is used to identify the book. A short description (no more than 500 characters) is associated with a book (provided by the library). In addition, a URL may be provided to link to a more detailed description at publisher’s website.

**3.1.2 Categories -** The books available at the library are categorized using a predefined classification tree. Each node represents a set of items. The root of the tree is labeled ‘All’ and represents all items. Each, with a descriptive name, represents some subset of the items represented by the parent. An item can be specified by a path through this classification tree. For example, we may categorize a book as:

1.     Language > Novels > Science Fictions > **Hunger-Game**

**3.1.3 Suppliers -** The library supplies all the books and information.

**3.1.4 Searching & Browsing -** Users are able to search the books by entering some keywords or conditions. As a search result, a list of items that satisfy the search criteria is returned to the user. We will keep the browsing history of the users, and check on the current status of the certain book. When a page is loaded, the status of corresponding books will be updated. Based on the record of those update, the system will evaluate the current weekly or monthly popular book list.

**3.1.5 Registered Users -** To rent a book, a user must be registered. A registered user is identified by a user name and authenticated with a password. In addition, the maintained information includes, e-mail address, name, age, address (which consists of street, city, state, and zip), phone number, and etc. A user will be given a unique member number. Users are also rated. If the rating is extremely low, the user will be marked in our blacklist system.

**3.1.6 Rating -** We attempt to suggest most popular books by readers’ ratings. A user can rate the book on how much he/she liked it with some little comment (Max 100 words).These ratings and a short explanation (less than 300 characters) are then made available to other users.

**3.1.7 User\_Rating** – Since we determine the user blacklist by our Rating system, we decide to do a different rating system for users. As user are detected to have some problem with the returned books. We will be marked into blacklist.

**3.1.8 Transaction History -** We will keep track on the transaction history for users. The certain date, and book ID will be store in a table. Users can only access their own transaction history. The library will have access to every user’s transaction history, and check on the current status of a certain book (i.e. Book A is currently rented by someone, Book B has been returned by user S.)

**3.1.9 Reservation (bidding) -** A user could reserve at most 5 different books at the same time. Each book can be put at most 10 reservations. The user in blacklist cannot place any reservations on any books.

**3.1.10 Rent (Sale) -** The maximum rental date for each users is a week for each book. By default the maximum rental date is week. However, if necessary, the user could extend their rental date up to two weeks. This option is only for user with the good user mark.

**3.1.11 Delivery -** When users placed a rental order or reservation order, the library will check the users’ profile, and deliver the selected books to the address provided by the users. (Our library only delivers books to users in United State)

**3.1.12 Reports to Telemarketers -** We will monthly report to telemarketers that how many books are rented by a person, the person’s information, such as name, address, e-mail, phone, age, and gender.

**3.2 Extended features:**

**3.2.1 Blacklist -** We will attached a blacklist to our registered users. It contains a rating ranks for corresponding user, and comments for why this user is added to the blacklist (Max 100 words). These ratings and a short explanation (less than 300 characters) are then made available to other users.

**Section 4 Relational Schema in SQL**

All the create table SQL statements for entities are listed below. From the final ER diagram, we developed the relational schema in SQL for all the entities and relationships. There are 8 entities and 15 relationships in our model. For the create table statements for entity, they are straight-forward conversion from the ER model we have shown above.

|  |
| --- |
| **create table member (member\_id int, gender varchar(10), date\_of\_birth date, reserve\_count int, name varchar(30), password varchar(20), email varchar(30), primary key(member\_id));** |
| **create table address(street\_no int, apt\_no int, zipcode int, state varchar(20), city varchar(30), member\_id int, primary key(member\_id));** |
| **create table reservations(reserve\_id int, member\_id int, isbn varchar(30), reserve\_date date, primary key(reserve\_id));** |
| **create table rent(member\_id int, transaction\_no int, rent\_date date, return\_date date, isbn verchar(30), return\_confirm boolean, primary key(member\_id));** |
| **create table rating(isbn varchar(30), isbn varvhar(30), rating\_id int, comment varchar(1500), star int, primary key(isbn), unique(rating\_id));** |
| **create table books(isbn varchar(30), stock int, avg\_rating int, title varchar(200), auther(200), description varchar(1500), cover blob, publisher varchar(100), url varchar(100), category varchar(20), primary key(isbn));** |
| **create table admin(admin\_id int, password varchar(30), primary key(admin\_id));** |
| **create table user\_rating(user\_rating\_id int, rating int, primary key(user\_rating\_id));** |

All the create table SQL statements for relationships are listed below. For the create table statements for relationships that we converted from our ER model, we have a cascade option in the rates table which deletes the tuple in user\_rating table when the member\_id in the rent table is deleted. When we tried to create tables in MySQL, we found that there is a constraint in MySQL that we cannot define more than 1 primary key in a table, so in the implementation, we will adjust our design based on MySQL constraints.

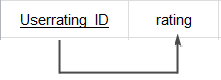
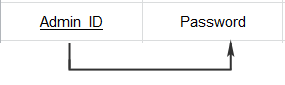
|  |
| --- |
| **create table lives\_at(member\_id int, primary key(member\_id) foreign key(member\_id) references member, foreign key(member\_id) references address);** |
| **create table delivers(member\_id int, reserve\_id int, primary key(member\_id, reserve\_id), foreign key(member\_id) references member, foreign key(reserve\_id) references reservations);** |
| **create table rent\_reserve(member\_id int, isbn varchar(30), primary key(member\_id, isbn), foreign key(member\_id) references member, foreign key(isbn) references books);** |
| **create table deletes(member\_id int, isbn varchar(30), primary key(member\_id, isbn), foreign key(member\_id) references rent, foreign key(isbn) references books);** |
| **create table place(member\_id int, reserve\_id int, primary key(member\_id, reserve\_id), foreign key(member\_id) references member, foreign key(reserve\_id) references reservation);** |
| **create table leave(member\_id int, isbn varchar(30), primary key(member\_id, isbn), foreign key(member\_id) references member, foreign key(isbn) references rating);** |
| **create table makes(member\_id int, primary key(member\_id), foreign key(member\_id) references member, foreign key(member\_id) references rent);** |
| **create table orders(reserve\_id int, isbn varchar(30), primary key(reserve\_id), foreign key(reserve\_id) references reservations, foreign key(isbn) references books);** |
| **create table has(isbn varchar(30), primary key(isbn), foreign key(isbn) references rating, foreign key(reserve\_id) references books);** |
| **create table rates(user\_rating\_id int, member\_id int, primary key(user\_rating\_id, member\_id), foreign key(member\_id) references rent, foreign key(user\_rating\_id) references user\_rating, on delete cascade);** |
| **create table contains(member\_id int, isbn varchar(30), primary key(member\_id, isbn), foreign key(member\_id) references rent, foreign key(isbn) references books);** |
| **create table manage1(member\_id int, admin\_id int, primary key(member\_id, admin\_id), foreign key(member\_id) references member, foreign key(admin\_id) references admin);** |
| **create table manage2(reserve\_id int, admin\_id int, primary key(reserve\_id, admin\_id), foreign key(reserve\_id) references reservations, foreign key(admin\_id) references admin);** |
| **create table manage3(isbn varchar(30), admin\_id int, primary key(isbn, admin\_id), foreign key(isbn) references books, foreign key(admin\_id) references admin);** |
| **create table manage4(member\_id int, admin\_id int, primary key(member\_id, admin\_id), foreign key(member\_id) references rent, foreign key(admin\_id) references admin\_id);** |

**Section 5 Functional Dependency**

**5.1 Functional Dependency**

**User Rating**

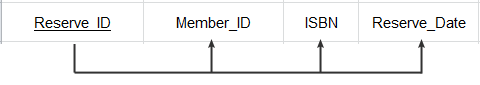
**Admin**

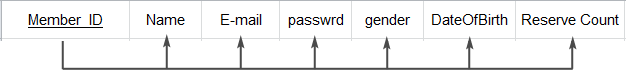


**Rating**

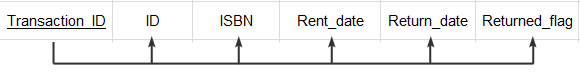


**Reservation**



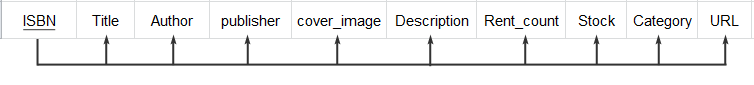


**Member**

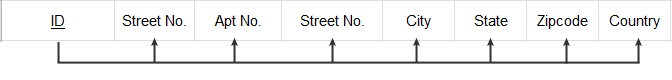


**Rent**

**Book**

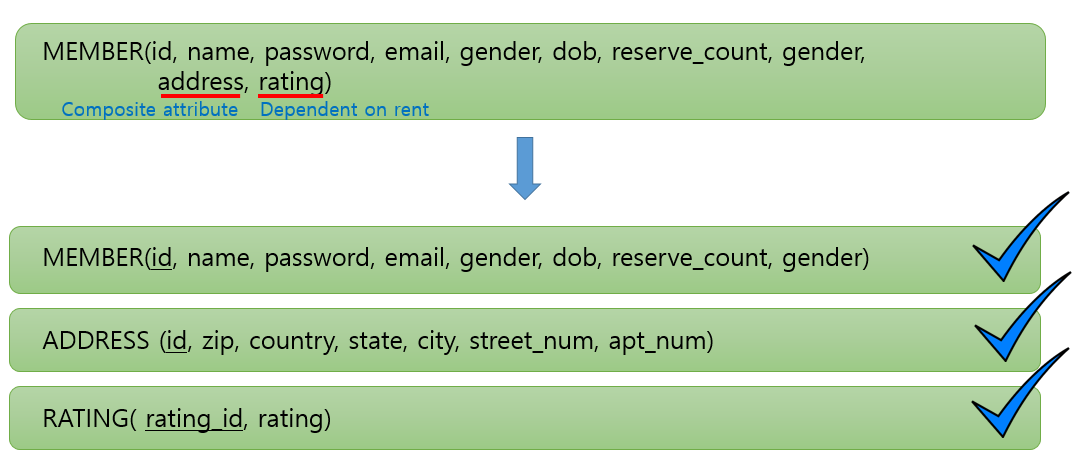


**Address**



**5.2 Normalization**

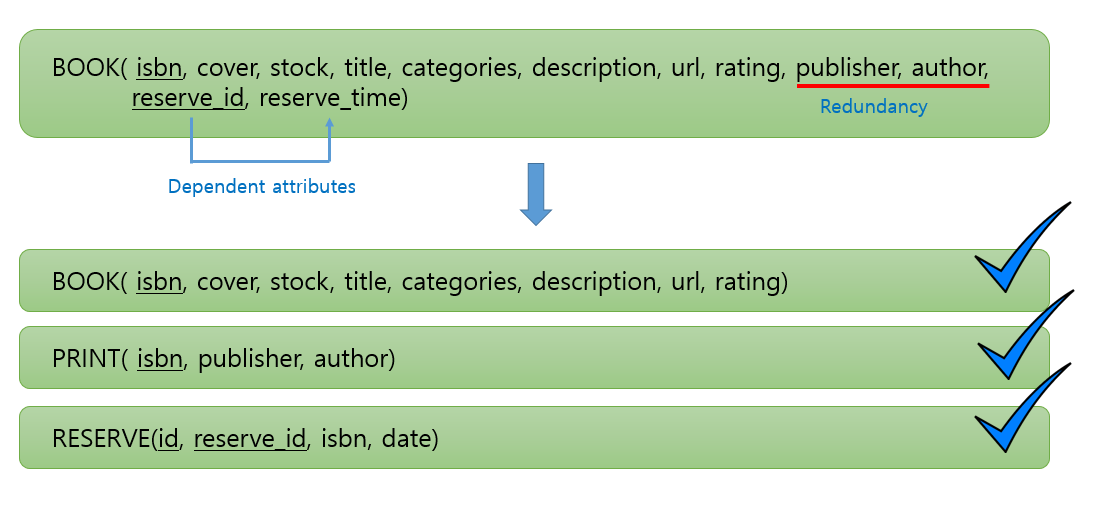
**5.2.1 Normalization 1**



**Address** is a composite attribute that contains various attributes, so we decided to make it entity.

**Rating** is mainly dependent on the Rent entity in our design, so we decided to make it an entity and relate to rent.

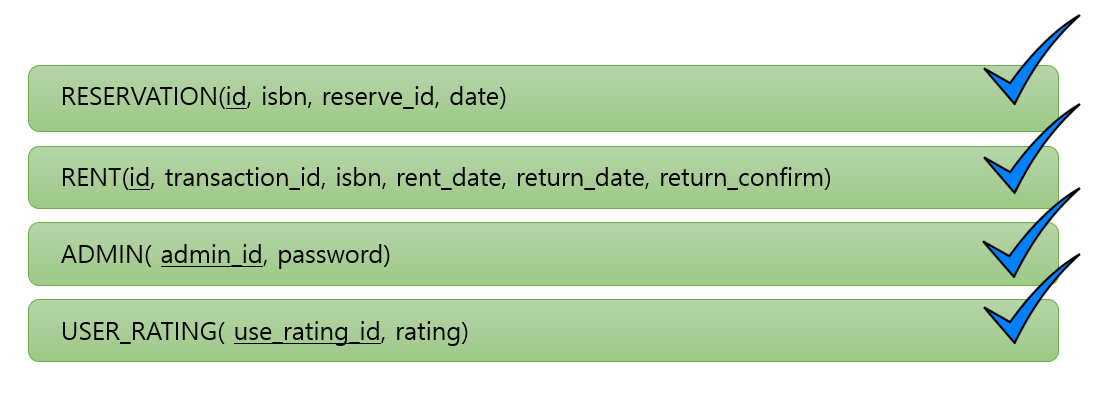
**5.2.2 Normalization 2**



**Publishers, and author** will be redundant in many cases, such as serious of books so we decided to move it out to reduce redundancy.

**Reserve\_time** is only dependent on **reserve\_id**, therefore we made an separate entity for it.

**5.2.3 Normalization 3**

Other entities are legitimately normalized.

By normalization we reduced redundancy, and made dependency more clear which helped to support our relational design,

**Section 6 Summary**

With a similar idea of Himalaya.com, we tried to design a book rental system for a library. Before we start it, we consider this project for two perspective: the user perspective and the admin perspective. We try to provide enough features to fulfill the requirement of our target potential user. At the same time, we also made the system free so almost everyone could be able to find what they in our system. In our design concept, we consider the Member Entity, Address Entity, Reserve Entity, Rent Entity, Rating Entity, Book Entity and Admin Entity as major component, and construct relationship for each entities in order to build our DBMS. During the actual brainstorming, we tried to relate our case to the real world problem. Then, we decided to have our major entity as strong type. With further discussion, we determine many necessary associated attributes for our entities, which most of them mapping to the baseline features, and the blacklist feature mapping to the extended feature.

Our unique features which distinguish our project to against other competitions is the rating system. Based on the rating, we will determine a blacklist feature. The purpose is that we want to increase the communications among our users. We will be glad to see that people using our system to discuss what books they like and what books they don’t like. However, our administrators will also gain control over these comments.

**Section 5 Appendices**

**Appendix A** **Design Approach**

In our design, we have an Admin as the system manager that gains the control over all entities. We expand our list of the entities based on the real world cases and we also did researches online about similar web applications such as Himalaya.com to identify our approach. Since we are trying to implement the book online rental system, we want to have similar features as Himalaya.com. We want to do deliver, online book browsing or searching, create account for users, and ratings for both users and the books. In our research, we found that there is a perfect rating system for those existing application. So we decided to enhance the rating system by creating the blacklist.

In our project, we will focus on the admin entity first. Because we want this entity have fully control over our application. For example, the admin entity could manage the rating for users. Once the user returned a damaged book, the administrator marks a low rating for that particular renting transaction (which points to the user), and will eventually cause that user being put into blacklist.

That is mainly the reason we split rent and reservation into separate entities, which seems intuitively should be relationships. We want the admin to be able to have relationships with these entities.

The other action we did to refine our schema is to separate the user rating entity. We find that every renting transaction should result in a rating, as mentioned above. We also made it in full participation constraint to fulfil the requirement.

**Appendix B Project Plan**

Schedule:

|  |  |
| --- | --- |
| Deadlines | Tasks |
| Week1 |  |
| Week2 | Team assigned |
| Week3 | Brainstorming the topic |
| Week4 | Brainstorming the details and design of the project |
| Week5 | Define project details and tools |
| Week6 | Complete conceptual design |
| Week7 | Implement database(1) |
| Week8 | Implement database(2) |
| Week9 | Spring break |
| Week10 | Implement website(1) |
| Week11 | Implement website(2) |
| Week12 | Integrate the website with the database(1) |
| Week13 | Integrate the website with the database(2) |
| Week14 | Additional implementation |
| Week15 | Testing |
| Week16 | Final review |

**Appendix C Progress Report**

**Progress Report (3) for CMPSC431W**

Summary

For our phase 2 report, we first fix some issues that we had from phase 1 report. As we kept keeping on our missing items. We re-design our whole overall ER diagram. Once we finished those basic works, we wrote the relation schema based on our ER diagram. Then, we figure out the all functional dependencies for all the entities and relationship. At last, we convert these functional dependencies into third normal form.

Individual Activities

Eiji Iriyama

* Assignments - Writing relation schema by using mysql: 2/27/2015 – 3/5/2015 (8hr)
* Background - studied basic commands for MySQL(3hr)
* Meetings - Group meeting: 2/27/2015 – 3/5/2015 (26hr)
* Design –Relation schema
* Implementation – Implement relation schema through mysql
* Testing – testing relation schema through mysql

Jackson(Weitong) Wang

* Assignments - Writing Project Report, Design ER diagram: 2/27/2015 – 3/5/2015 (8hr)
* Background - Student basic MySQL knowledge(3hr)
* Meetings - Group meeting: 2/27/2015 – 3/5/2015 (26hr)
* Design – Normalization, ER diagram
* Implementation - none
* Testing - none

WeiDi Shen

* Assignments – Design ER diagram, Functional Dependency, and figure out normalization: 2/27/2015 – 3/5/2015 (8hr)
* Background - Studied web-based programming (3hr)
* Meetings - Group meeting: 2/27/2015 – 3/5/2015 (26hr)
* Design – Normalization, Functional Dependency, ER diagram
* Implementation - none
* Testing -none

Seong Kyu Kim

* Assignments - Writing Project Report, Power Point: 2/27/2015 – 3/5/2015 (8hr)
* Background - Web development hands on: Node.js, Ajax, JQuery
* Meetings - 2/27/2015 – 3/5/2015 (26hr)
* Design – Normalization
* Implementation - none
* Testing – none

Group Activities

* Group meeting(26hr) 2/27/2015 – 3/5/2015
* Brainstorming and discussion about the project
* Figure out relation schema, functional dependency, and normalization.
* Writing project report, progress report, and power point for phase 2 presentation

Future Task

* As we finish all the basic requirement for the actual implementation. We will try our best to develop the actual system through web.

**Progress Report(2) for CMPSC431W**

Summary

We decided to change our topic of the project from Pokemon website to an online book rental system for a library. The core attribute we would like to strive for is user-friendliness. We would like to pattern the features from app stores – using categories to narrow the search down and use lineups in each section to increase the vividness of the application. A general searching feature would of course also be supported. Users can search by key words, author, ISBN, publisher, or any combinations of the search categories. Also, we have developed several diagrams for ER-model to our project.

Individual Activities

Eiji Iriyama

* Assignments - Writing Progress Report: 2/8/2015(1hr)
* Background - studied basic commands for MySQL(3hr)
* Meetings - Group meeting: 2/8/2015(3hr)
* Design - none
* Implementation - none
* Testing - none

Jackson(Weitong) Wang

* Assignments - Writing Project Report, Power Point: 2/8/2015(2hr)
* Background - Student basic MySQL knowledge(3hr)
* Meetings - Group meeting: 2/8/2015(3hr)
* Design - none
* Implementation - none
* Testing - none

WeiDi Shen

* Assignments - Writing Progress Report:  2/8/2015(1hr)
* Background - Studied web-based programming (3hr)
* Meetings - Group meeting: 2/8/2015(3hr)
* Design - none
* Implementation - none
* Testing -none

Seong Kyu Kim

* Assignments - Writing Project Report, Power Point: 2/8/2015(2hr)
* Background - Web development hands on: Node.js, Ajax, JQuery
* Meetings - Group meeting: 2/8/2015(2hr)
* Design - none
* Implementation - none
* Testing – none

Group Activities

* Group meeting(3hr) 2/8/2015
* Brainstorming and discussion about the project
* We have developed the first phase ER-model
* Writing project report, progress report, and power point for phase 1 presentation

Future Task

* Since we only have one teammate who has experience with web-programming, we are going to keep studying about web-programming, such as jsp.

**Progress Report(1) for CMPSC431W**

Summary

1/25/2015 – We decided the topic for the project:

Pokemon Open Dictionary where people can prompt the attributes and characteristics of a certain pokemon on web which is going to be saved in a DBMS. The saved data is to be interpreted to allow look up(search), communications between the users, and provide with team combination recommendations according to the user requirements.

Individual Activities

Eiji Iriyama

* Assignments - Writing progress report: 1/25/2015(1hr)
* Background - studied basic commands for MySQL(4hr)
* Meetings - Group meeting: 1/25/2015(2hr)
* Design - none
* Implementation - none
* Testing - none

Jackson(Weitong) Wang

* Assignments - Assist writing progress report: 1/25/2015(1hr)
* Background - Have experience in software application development with use of database
* Meetings - Group meeting: 1/25/2015(2hr)
* Design - none
* Implementation - none
* Testing - none

WeiDi Shen

* Assignments - Idea brainstorming, web implementation possibilities: 1/25/2015(1hr)
* Background - Studied basic concepts of jsp(2hr)
* Meetings - Group meeting: 1/25/2015(2hr)
* Design - none
* Implementation - none
* Testing - none

Seong Kyu Kim

* Assignments - Idea brainstorming, web implementation possibilities: 1/25/2014(1hr)
* Background - Web development experience with HTML, CSS, and JavaScript
* Meetings - Group meeting: 1/25/2015(2hr)
* Design - none
* Implementation - none
* Testing - none

Group Activities

* Group meeting(2hr) 1/25/2015