**The League of Legends Store**

By Project\_X

### Members:

### Jiayi Liang, Naiqiang Lin, Berty Ruan, Shaohua Wang, Yusheng Wang

### Table of Contents

1. Introduction Pg.1

2. Conceptual Database Design Pg.2

2.1. Requirement Analysis Pg.2

2.2. Entity Relationship models Pg.15

3. Logical Database Design and Normalization Pg.19

3.1. Schema Refinement and Normalization Pg.19

3.2. SQL Statements Pg.20

3.3. Technology Survey Pg.26

3.4. Populate Data for Database Pg.28

4. Appendices Pg.33

5. Conclusion Pg.35

**List of Figures**

Figure 2.1 - Example of browsing function Pg. 8

Figure 2.2 - Example of searching function Pg. 9

Figure 2.3 - Example of gifting function Pg. 13

Figure 2.4 - Example of gifting function Pg. 13

Figure 2.5 - Example of sub-item combine system Pg. 14

Figure 2.6 - Example of skin combine system Pg. 14

Figure 2.7 - Exmaple of rune combine systerm Pg. 15

Figure 2.8 - Entity Relationship Model Pg. 16

Figure 3.1 - Schema Normalization Diagram Part 1 Pg. 19

Figure 3.2 - Schema Normalization Diagram Part 2 Pg. 19

Figure 4.1 - Github Commit History Pg. 33

Figure 4.2 - Asana Task History Pg.34

**List of Tables**

Table 1 - Required information for Item with listed price Pg. 5

Table 2 - Required information for Item sold by auction Pg. 5

Table 3 - Sample table for categories of summaries Pg. 11

Table 4 - Sample stable of sale information Pg. 11

Table 5 - Weapon sale rank Pg. 12

Table 6 - Entity's Attributes Pg. 18

Table 7 -Relationship's Attributes Pg. 19

Table 8 -User Pg. 28

Table 9 -Buyer Pg. 28

Table 10 -Supplier Pg. 29

Table 11 -Credit Card Pg. 29

Table 12 -Address Pg. 29

Table 13 -Zip Code Pg. 30

Table 14 -Rating User Pg. 30

Table 15 -Rating Supplier Pg. 30

Table 16 -Item Pg. 30

Table 17 -Property Pg. 31

Table 18 -Historyr Pg. 31

Table 19 -Analysis Pg. 31

Table 20 -Category Pg. 32

Table 21 -Bid on Pg. 32

Table 22 -Deliver to Pg. 32

Table 23 -Task Distribution Pg. 34

# 

# Introduction

The League of Legends Store is a startup company that has partnered with Riot Games to deliver the best virtual and real word gaming experiences to the millions of League of Legends fans worldwide. The opening day for this store has been anticipated for months, and it has been hard at work to fill its warehouse with quality League of Legend cosplay items, and notify potential suppliers to utilize its store. A month before the release date, the founders convened for an emergency meeting. They had met with an independent contractor and were informed that their current item database was too designed poorly to be unusable in production. The contractor informed them that they needed to redesign their database before the opening day.

As a result of the emergency meeting, the founders have contacted a group of Penn State’s CMPSC 431W students to design a better database in order to sell their League of Legends items and record all users, items and transactions. To speed up the student’s work process, the contractor had given the students a three step plan to deliver a quality product which is outlined below, and the rest of the project report will be dedicated to the student’s documentation on their work in designing the database.

First the students need to analyze the League of Legends Store business requirements and construct the preliminary database design using the Entity Relationship Model paradigm. Next the students need to decide which database management system to use, create a logical database design, refine and normalize the initial design and populate the database. Afterwards, the students need to write a set of representative transactions to access the database. When all three project phases are completed, then the students may begin implementing the League of Legends Store website, test their code, and finally demonstrate their product to the founding team. If all goes well, the CMPSC 431W students will finish on time and satisfy all the founding members’ database requirements. The success of the League of Legends Store is contingent on a well designed and implemented database.

# 2. Conceptual Database Design

## The League of Legends store needs multiple functions to work within the expectations of its users and suppliers. For our design, the store separates into thirteen core functions such as sale items, searching for items, biding on items and etc.. These functionalities will be translated into an ER model, which is a simplification of the League of Legends store’s database, but allows the designers to understand how each thirteen functionality relates to the other and what attributes are connected to an entity.

## 2.1. Requirement Analysis

***Sale Item -*** Items sold in the League of Legends (LoL) store must be usable in the LoL game environment or provide enhancements to the LoL gaming experience. Before an item can be uploaded onto the LoL database and store, suppliers must include basic information about their product and customize how they want to sell their product. Both requirements will be described fully in this section.

A valid LoL item must be either a physical object that incorporates LoL gaming themes, or compatible with the LoL gaming engine, and is usable either in a LoL user’s profile and account or in at least one of the four LoL’s gaming modes, Summoner’s Rift, the Twisted Treeline, Howling Abyss, or the Crystal Star. All items need not be endorsed by the LoL community. Both types of items may be made by third-properties manufactures and unendorsed by the LoL community.

When selling an item on the LoL store, we ask the suppliers to provide rudimentary descriptions of their product and other information. All items must list the name of the product and supplier. A source may be a company, individual or an LoL affiliated organization. The supplier needs to provide a short description and location of shipment. The type of item must be either listed as physical or virtual. When the item has been validated by the LoL store, it will be automatically assigned a unique identifier.

The seller must also specify to sell the item by listed price or by auction. If by listed price, the static price in dollars and the amount of stock available should be associated with the item. If there is no more stock for an item, it will be removed from the LoL store. If by auction, the seller must provide the reserve price, hidden from LoL customers, and the start and end date of the auction. LoL customer may bid on auctioned items between the start and end dates. Otherwise those items will not be visible on the LoL store. In addition the LoL store will automatically cancel auctions that pass their end dates or if the highest bid price is lower than the reserve price. All auctioned items must be sold individually.

An example of the required information needed from the supplier for items sold by listed price and auction is provided in the next page.

**Table 1 - Required Information for Items with Listed Prices**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item Name** | **Supplier name** | **Supplier’s Location** | **Descript.** | **Type** | **Price ($)** | **Stock** |
| Big Sword | Grapefruit | NYC, USA | Very Useful | Virtual | 9.99 | 7 |

**Table 2 - Required Information for Items Sold by Auction**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Item Name** | **Supplier Name** | **Supplier’s Location** | **Descript.** | **Type** | **Reserve Price ($)** | **Start Date** | **End Date** |
| Guardian Angel | LoL | Chicago, USA | Durable | Physical | 1,899 | Dec. 1, 2015 | Dec. 1, 2016 |

***Categories -*** All items listed in the LoL store will be categorized using a predefined classification tree, and can be found in the store through the tree. Assigning a category will be the responsibility of the seller. The properties of the classification tree will be described here.

Categories are described as nodes and represents a set of related items. Nodes are related to each other as parents or children. Parent nodes are strict supersets of the current node while child nodes are strict subsets of the current node. The root of the tree is labeled “All” to represent all items. The number of children for each node can reach up to no more than fifteen, and the height of the classification tree will be at least 10 nodes deep. Each node is given a descriptive name to help the suppliers and customers navigate through the store. Multiple nodes of the same descriptive name may exist, and items may be placed into multiple nodes. Items added by suppliers must be assigned to a leaf of the classification tree, or will be defaulted to the ‘General’ node in the subtree. Parent nodes of the leaf and the leaf itself will have access to the given item. An example classification tree is given below.

1. All -> Physical -> Cosplay Item -> Summoner’s Rift -> Consumable -> Elixer of Wrath
2. All -> Virtual -> In Game -> Summoner’s Rift -> Illegal -> Over Powered -> Bonus Attack Speed -> Basic Stats -> Champion Selection -> Jinx
3. All -> Virtual -> General

***Suppliers -*** All suppliers must register themselves onto the LoL store before uploading their products. Supplier’s information will be stored in the store’s currently unsecured personal databases. For all new suppliers on the LoL store, their type of supplier, individual or a company, must be specified. If it’s an individual, the seller’s name, permanent home address, home phone number, email address, annual income, and credit card number must be provided. If the seller is a company, its organization’s name, main address, and annual revenue, point of contact’s name, phone number, email address, and the bank account’s routing number must all be provided.

***Registered Users -*** User will need to agree with the terms and conditions that are provide by the League of Legends store before creating an account. Only the email address, username and password are required to register.

For users who want to buy products, they will need to add their personal information which includes name, address (street, city, state, and zip), credit card information, and date of birth.

If users want to sell items they will need to give other information including company information as reflected in the *Suppliers* sectionabove.

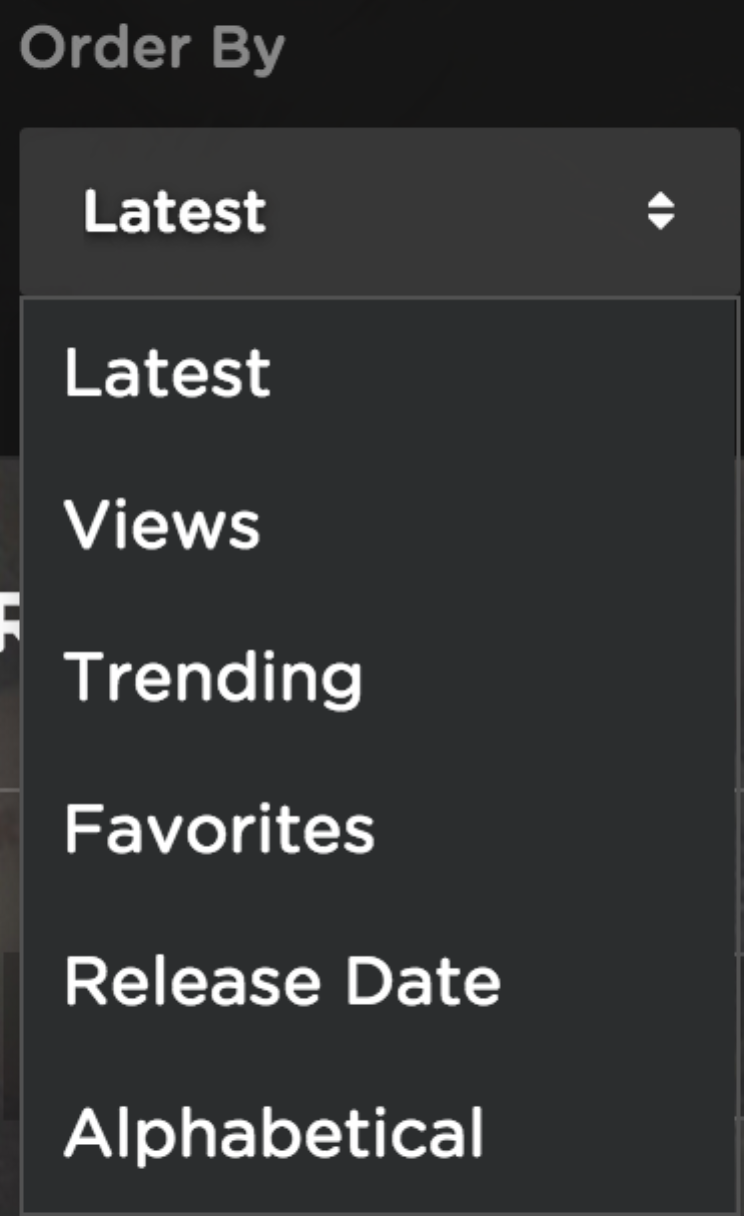
***Rating -*** We will be implementing a rating system for both customers and sellers. Once an item is paid and if applicable, delivered, the transaction will be kept in the system. During this period, the customer may rate and comment on and rate the quality of the product, his or her satisfaction, the trustworthiness of the seller, and the service of the delivery. The rating system will be quantified by a star from 1 star being the least satisfied to 5 stars being the most satisfied.

Once a comment and rating have been submitted, the seller will have two days to negotiate with the customer about the rating and improve their service or replace their product in order to receive a batter rating.

Customers may also comment and rate the League of Legends store’s quality, however this information will not be recorded internally by its databases and are not guaranteed to be considered or even read.

***Browsing*** - When a user clicks on a category, he will see all the items under that tree. The default view will have up to 30 items on each page. The user may adjust it among 15, 50, and 100 items per page. Each page will have a summary of the category and what products are listed there.

Additionally a user can also order the items on a page by using different keywords such as most sold, most comments, best ratings, trending items, most favorite items, selling date, sort alphabetically, most expensive, and least expensive. This mode of browsing is demonstrated in Figure 2.1 below. Once selected, the system will sort the items depending on that keyword.

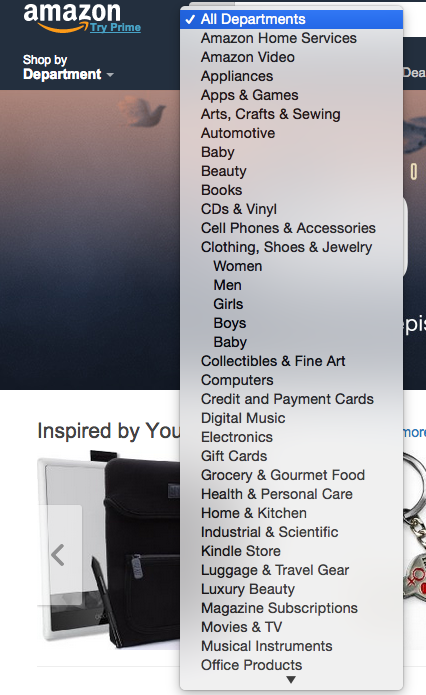


**Figure 2.1**

***Searching -*** Users can search for the item by keywords, if the user knows which category the item the user is looking for belongs to, the user can search by category, Such like the Figure 2.2 indicates.

We could use MapReduce to calculate the rank for each item: The initial rank is the same to every item, then we record how many visit each item gets in one day, and how the stream flows. Using MapReduce algorithm we can update the rank for each item every day and by doing this we can provide a better search result by presenting the item with higher rank in the front.

However, it is optional to implement the MapReduce algorithm, it is also feasible to arrange the order of the result by the frequency of an item is being visited in a day.



**Figure 2.2**

***Sale*** *-* We can use a table to record which credit card is charged when a certain item is sold, the table will have at least have three columns: item id, credit card number, andprice. Once the seller confirms that the credit card information is valid, a new row is created in the table meaning that the transaction is finished and this row will be kept in the table for at least six months.

The credit card information is retrieved from the buyer’s account profile table, and this information is processed in the backend to check if the card number length is correct, if the expiration date is later than half a month from the day the buyer purchases the item.

The supplier has the option whether to put the item in auction function of in the selling list. If the item is in both auction list and selling list, then the sale price is the reserve price.

***Bidding -*** To make every bid valid, we need a function to check that the bidder is not seller and the bid happens between the time the item is registered for auction and the end of the auction. When one wants to bid, he or she can bid at any amount higher or equal to $2. Only meeting these criterion can a bid be valid.

After the auction ends, the system sends message to every bidder and seller that who is the winner of the auction and the how much he or she bid. The winner will also get a message that confirms the credit card information, if the user confirms the message, the auction succeeds, the status of the item changes to “sold” and the user is charged. However, if the user refused to pay the price, the auction fails, the bidder gets a negative comment for the failure of the auction and the supplier can choose register the item for another auction.

***Order and sale reports:*** The Summary report will provide an overview of sale information. Every week, a report is generated to summaries the ordering and sales information based on categories of items just like Table 3 shown below.

**Table 3 - Sample table for category of summaries**

|  |  |  |  |
| --- | --- | --- | --- |
| category | average | total amount | total price |
| champions | 23 | 2 | 46 |
| rune pages | XXXX | XXXX | XXXX |

Also, for individuals, a similar report of sale information is generated every day, shown Table 4.

**Table 4 - Sample table of Sale information**

|  |  |
| --- | --- |
| category | cost |
| champions | 32 |
| rune pages | 43 |

In addition, for different category, ranking reports of sale items are provided for clients to see which one is the most popular in the latest week, as shown Table 5.

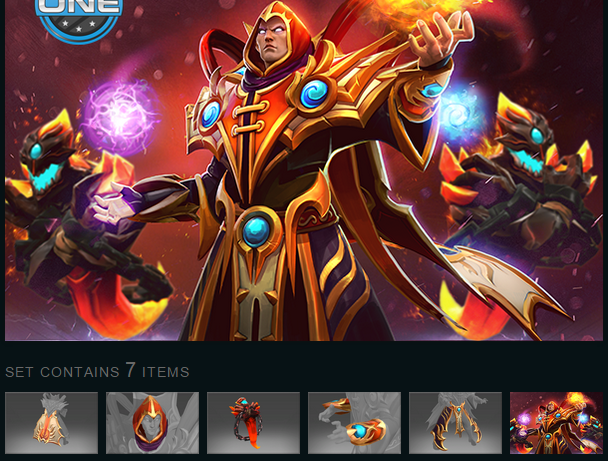
**Table5 - weapon sale rank**

|  |  |  |
| --- | --- | --- |
| rank | item | amount sold |
| 1 | weapon 1 | 4543 |
| 2 | weapon 2 | 2333 |

From the reports, the developer will know what the favors of players are and based on the reports, the store can improve its business.

***Delivery:*** All virtual items are implicitly granted access to the virtual item by our supplier after purchase and are not delivered. Delivery is only valid for physical items. The delivery system shows the order id and the tracking number. The system should record the delivery status of each item to ensure the success of the order.

***Recommendation and Gift:*** Every week, based on information of the League of Legends champions played in the last seven days, the items of heroes that LoL users play the most frequently are recommended for you as shown in Figure 2.3. Users can also select a timeline to see which heroes were popular in the past.



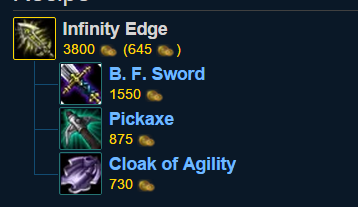
**Figure 2.3**

In the gift system, anyone can send an item as a gift to others. Highly liquid market can attract more and more players to join in. Content Gifting includes skins, champions, rune pages, icons, ward skins, and mystery gifts as seen in Figure 2.4.



**Figure 2.4**

***Combination System:*** Since we are doing the League of Legends Store, it is necessary to have a combination system that allows users to combine multiple sub-item into another item. For example, to make a weapon named infinity edge, it first needs a BF Sword, a Cloak of Agility and a Pickaxe (Figure 2.5). The latter three items must be acquired before purchasing the ability to combine the three into a super item.



**Figure 2.5**

Not every weapon item can be combined such as cosmetics and physical items. However, it is possible to form a customizable virtual cosmetic that can be used for different LoL champions and create a combination system for them (Figured 2.6). In this functionality, we can design a combination system by giving the League of Legends store users the option to allow current items to form a new complete item that is being sold by the user.



**Figure 2.6**

A rune system is another important parts in League of Legends (Figure 2.7) and also utilizes the combination system. For this system, it will ask if buyer would like to randomly or directly combine their inventory of runes. Direct combination will give more space for buyer to choose, but random combination will cost them less money.

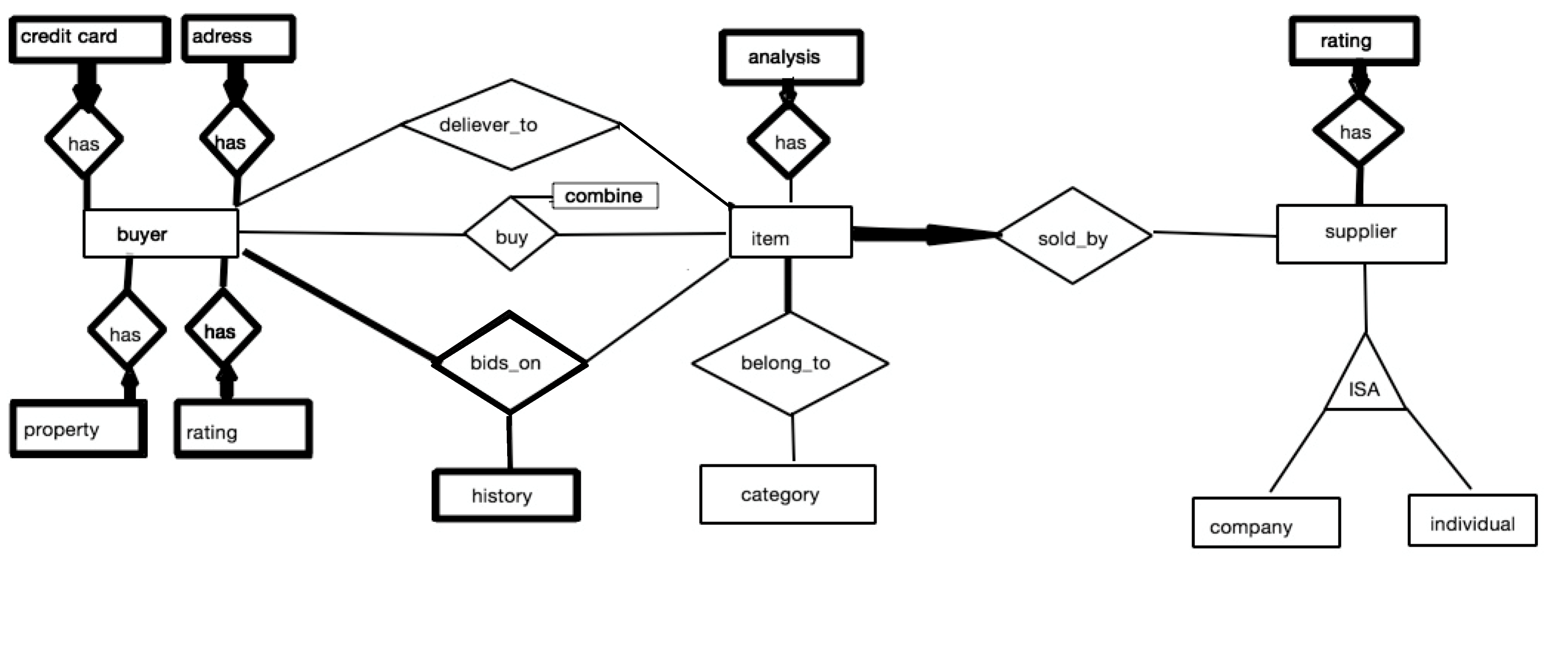
Regardless the type of combination, this functionality will always look into the buyer’s account and check the prerequisites to build a complete item. When the combine request is successfully made, the data of this account and selling history will be recorded, and the purchaser will receive a message for the successful combination.



**Figure 2.7**

***News, Deal and Follow Ups:*** It is a common scene that every company will provide follow up news or sale deals to the buyer who bought products and provided their emails and we believe this will be a good way to help promote our supplier’s products as well. This follow-up function will sent the information of our newest deals and our activists to the buyers regardless of their privacy preferences.

## 2.2 Entity Relationship models



**Figure 2.8**

***ER Model Narrative –*** To begin our narrative of our ER model in Figure 2.8, let’s talk only of the buyer, item and supplier entities and their relationships that are independent of the other two mentioned entities. This will make our analysis simpler. Later, we will consider the relationship sets that span across these entities.

Then let us begin first with the left-most entity, the buyer. While this entity has many attributes, it also has attributes that must be listed multiple times. They are the credit card and address entities, and must be described in separate relationship sets. In addition, the buyer may also have a list of personal properties and ratings given by the seller after a purchase. Personal properties denote the buyer’s purchasing history and how much was bought. The ratings are given by the suppliers to rank the buyer’s rating and comment quality. Notice that all four entities (on the left) related to the buyer are weak entities. This is because if a buyer is removed from the system, then these four relationships must be removed also.

Next we have the item. Besides its attributes which contains the item’s name and price, it belongs to two relationships. The first one is has\_analysis. This relationship gives each item a financial statement that includes how many items were sold and revenue collected. Consequently this relationship is related to the weak entity analysis. For the belongs\_to category, all items must be sorted into a category or else they will not appear in our League of Legends store view.

Lastly, the supplier can also have a rating like the buyer. The rating is given by many buyers, but exists only if the supplier chooses to sell in the League of Legends store. If the company decides to leave or gets removed, the ratings should be removed also. Notice that we also enforce that all suppliers must have ratings. The supplier entity also belongs to an ISA relationship. This helps distinguish the two types of merchants in our store by focusing on independent suppliers to corporate suppliers. Of course, these two groups also share common attributes such as names.

Now let us proceed to the inner relationship sets. First we have the deliver\_to entity. Not all buyer or items have to be deliver\_to each other. However when they are, the order id and tracking number for the delivery must be recorded. For the bids\_on relationship, it’s also a weak entity because its existence depends on an item’s existence. Also all buyers must have a bidding history whether it’s nothing or something. For the buy relationship set, a buyer may buy an item or choose a superitem from the combination entity and choose to buy an item from there. The reason why the combine entity is not connected to a supplier is because the League of Legends store enforces all combination items and does not allow a supplier to identify or profit from them. Finally, we have the sold\_by relationship. This relationship specifies that all items must be related to one supplier. This makes sense because items are sold by suppliers and no more than one person may sell and profit from an item.

***Attribute Table -*** Because the ER model contains many attributes, we decided to simplify it by recording all attributes in a table. The left most column represents the entity or relationship and the following fields are the attributes that correspond to the representative model.

**Table 6 – Entity’s Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User** | Id | Email address | Phone number | Username | password |
| **Buyer** | Name | DOB | income |  |  |
| **Supplier** | Company name | Company category | revenue | Bank routing number | Company Address |
| **Credit Card** | Credit card number | Name | Expiration date | brand | CVV number |
| **Address** | Address id | Street | Apartment number | State | City |
|  | Zip code | Is Permanent |  |  |  |
| **Rating for User** | Rating id | Comment | Star rank | Supplier id | Product id |
| **Rating for Supplier** | Rating id | User id | Comment | Product id | Star rank |
| **Item** | Item id | Item name | Category | Producer | Price |
|  | Description | Item Type | quantity (-1 denotes bidding) | Combine id |  |
| **Property (items purchased)** | Product id | Quantity | Date purchased |  |  |
| **History (for bidding)** | Bid id | Item id (f) | User id (f) | Bidding Price | Time stamp |
| **Combine (max 3 subitems)** | Combine id | SubitemID1 | subitemID2 | subitemID3 | price |
| **Analysis (financial statement)** | Analysis id | Item id | Quantity sold | Total revenue |  |
| **Category** | Category id | Category name | Super category id (f) |  |  |

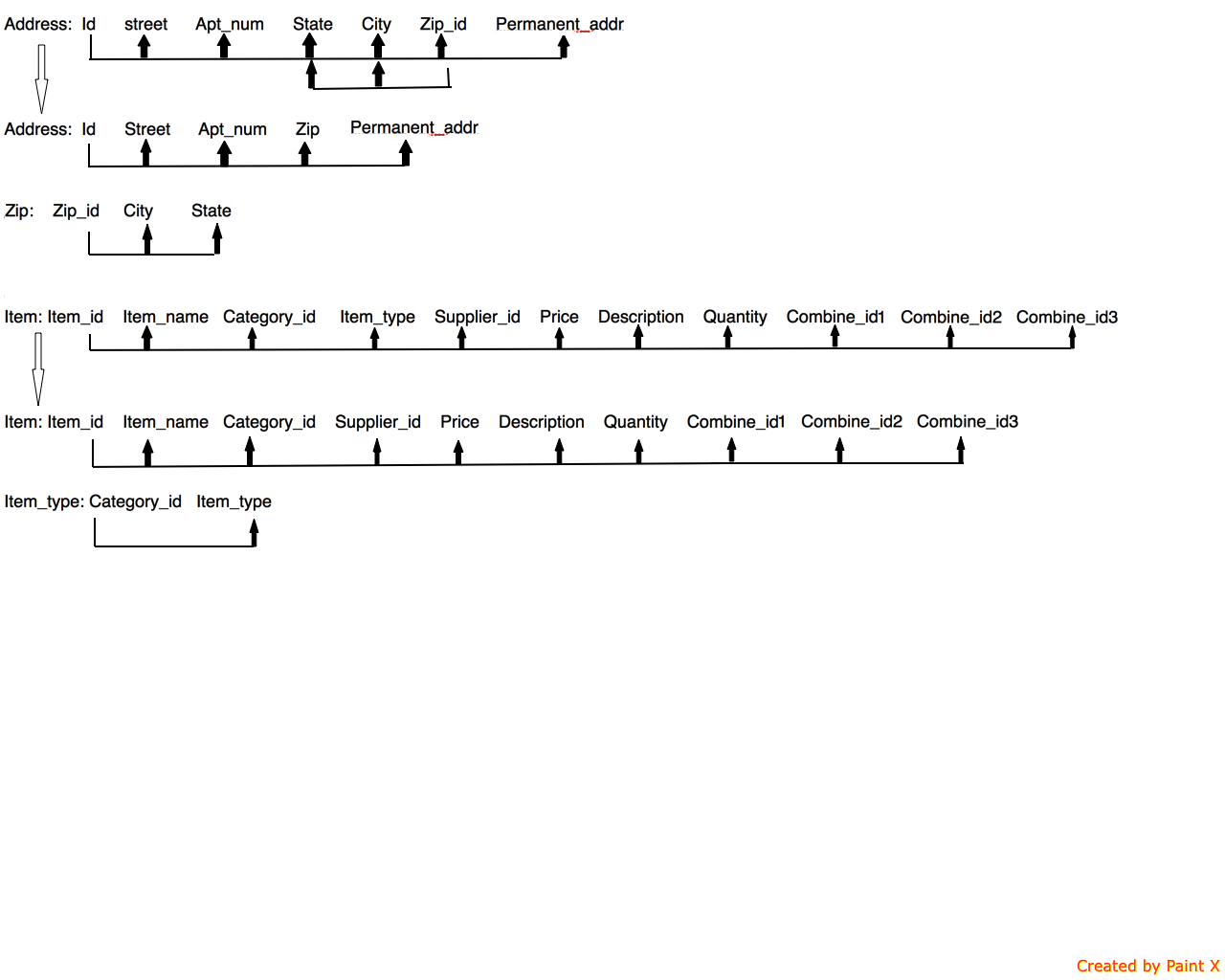
**Table 7 – Relationship’s Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bids\_on** | Start bid date | End bid date | Reserve price |
| **Deliver\_to** | Tracking number | Order Id |  |

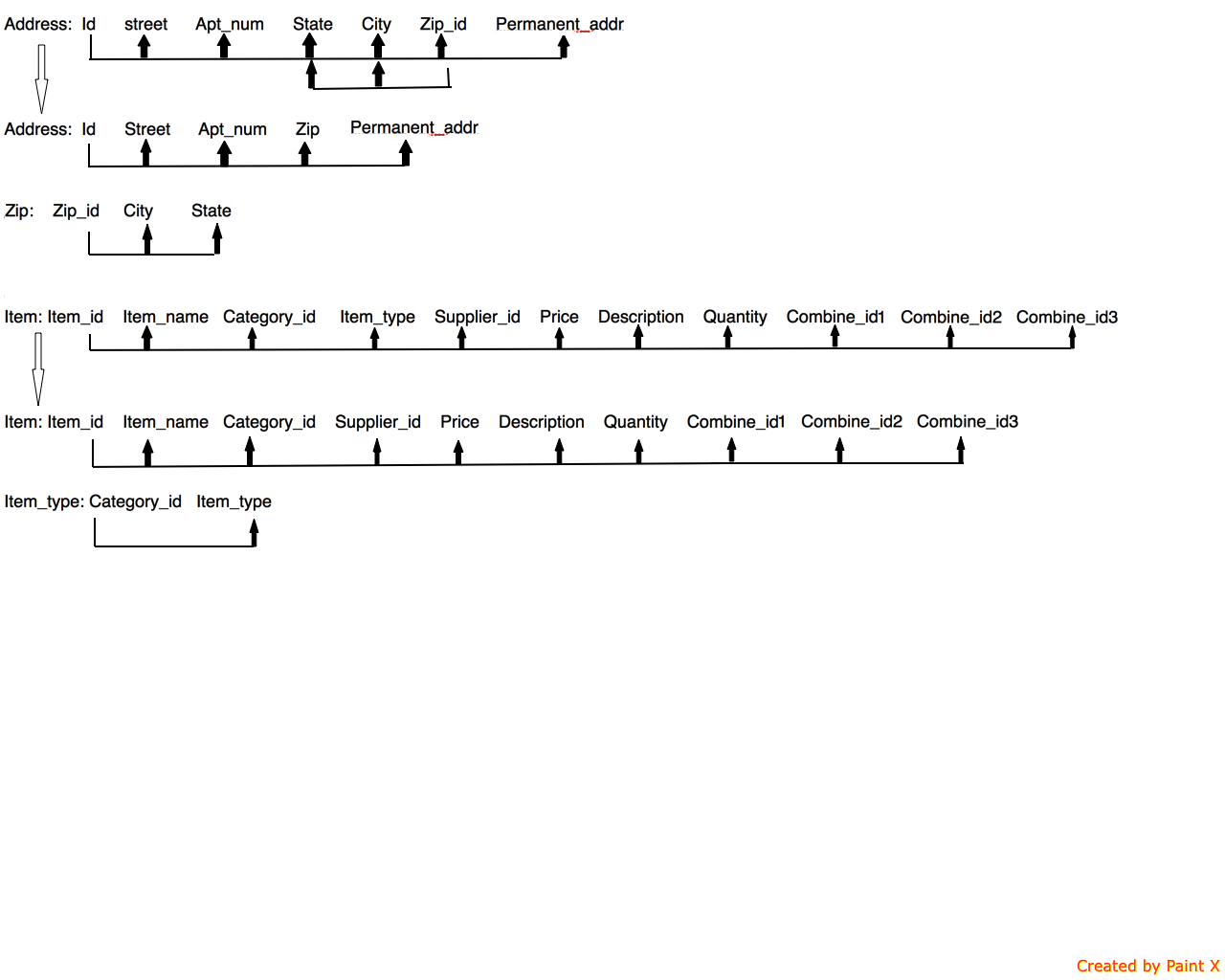
# 3. Logical Database Design and Normalization

**3.1 Schema Refinement and Normalization**

The figure below is our refined schema reduced to the third normal form. The team identified three entities - address, items, and combine - that needed to be normalized. Note that entities items and combine were joined before beginning the normalization analysis. For Figure 3.1, the motivation for normalizing it was because zip codes can uniquely identify cities and states. For items, a category\_id is a string of numbers that references to the item types, and specifies if the object is either virtual and physical.



**Figure 3.1**



**Figure 3.2**

## 3.2 SQL Statements

The followings are SQL statements used to create relations that we are going to use:

CREATE TABLE User(

Id int NOT NULL AUTO\_INCREMENT,

Email\_address CHAR(50),

Phone\_num CHAR(10),

Username CHAR(50),

Password CHAR(50),

PRIMARY KEY (Id)

);

CREATE TABLE Buyer(

Buyer\_id int NOT NULL AUTO\_INCREMENT,

Id int,

DOB DATE,

Income int,

PRIMARY KEY (Buyer\_id),

FOREIGN KEY (Id) REFERENCES User (Id)

);

CREATE TABLE Supplier(

Supplier\_id int NOT NULL AUTO\_INCREMENT,

Id int,

Company\_name CHAR(50),

Company\_category CHAR(50),

Revenue int,

Routing\_num int,

Address int,

PRIMARY KEY (Supplier\_id),

FOREIGN KEY (Id) REFERENCES User (Id),

FOREIGN KEY (Address) REFERENCES Address (Id)

);

CREATE TABLE Credit\_Card(

Card\_num int NOT NULL,

Name CHAR(50),

Expir\_date DATE,

CVV CHAR(3),

Card\_brand CHAR(50),

PRIMARY KEY (Card\_num)

);

CREATE TABLE Address(

Id int NOT NULL AUTO\_INCREMENT,

Street CHAR(50),

Apt\_num int,

Zip CHAR(5),

Permanet\_addr BOOLEAN,

PRIMARY KEY (Id),

FOREIGN KEY (Zip) REFERENCES Zip (Zip\_id)

);

CREATE TABLE Zip(

Zip\_id CHAR(5),

State CHAR(20),

City CHAR(30),

PRIMARY KEY (Zip\_id)

);

CREATE TABLE Rating\_user(

Rating\_id int NOT NULL AUTO\_INCREMENT,

Comment CHAR(140),

Star\_rank int,

Supplier\_id int,

Item\_id int,

PRIMARY KEY (Rating\_id),

FOREIGN KEY (Supplier\_id) REFERENCES Supplier(Supplier\_id),

FOREIGN KEY (Item\_id) REFERENCES Item (Item\_id)

);

CREATE TABLE Rating\_Supplier(

Rating\_id int NOT NULL AUTO\_INCREMENT,

Comment CHAR(140),

Star\_rank int,

User\_id int,

Item\_id int,

PRIMARY KEY (Rating\_id),

FOREIGN KEY (User\_id) REFERENCES User(Id),

FOREIGN KEY (Item\_id) REFERENCES Item (Item\_id)

);

CREATE TABLE Item(

Item\_id int NOT NULL AUTO\_INCREMENT,

Item\_name CHAR(50),

Category\_id int,

Supplier\_id int,

Price int,

Description CHAR(140),

Quantity int,

Combine\_id1 int,

Combine\_id2 int,

Combine\_id3 int,

PRIMARY KEY(Item\_id),

FOREIGN KEY(Combine\_id1) REFERENCES Item (Item\_id),

FOREIGN KEY(Combine\_id2) REFERENCES Item (Item\_id),

FOREIGN KEY(Combine\_id3) REFERENCES Item (Item\_id),

FOREIGN KEY (Supplier\_id) REFERENCES Supplier (Supplier\_id),

FOREIGN KEY (Category\_id) REFERENCES Category (Category\_id)

);

CREATE TABLE Property(

Item\_id int,

Quantity int,

Date\_purchased DATE,

PRIMARY KEY (Item\_id),

FOREIGN KEY (Item\_id) REFERENCES Item (Item\_id)

);

CREATE TABLE History(

Bid int NOT NULL AUTO\_INCREMENT,

Item\_id int,

Buyer\_id int,

Bidding\_price int,

Bid\_on int,

Time\_stamp TIMESTAMP,

PRIMARY KEY (Bid),

FOREIGN KEY (Item\_id) REFERENCES Item (Item\_id),

FOREIGN KEY (Bid\_on) REFERENCES Bid\_on (Bidon\_id),

FOREIGN KEY (Buyer\_id) REFERENCES Buyer (Buyer\_id)

);

CREATE TABLE Analysis(

Analysis\_id int NOT NULL AUTO\_INCREMENT,

Item\_id int,

Quantity\_sold int,

Total\_revenue int,

PRIMARY KEY(Analysis\_id),

FOREIGN KEY(Item\_id) REFERENCES Item(Item\_id)

);

CREATE TABLE Category(

Category\_id int NOT NULL AUTO\_INCREMENT,

Category\_name CHAR(50),

Super\_category int,

PRIMARY KEY (Category\_id),

FOREIGN KEY (Super\_category) REFERENCES Category (Category\_id)

);

CREATE TABLE Bid\_on(

Bidon\_id int NOT NULL AUTO\_INCREMENT,

Bid\_start\_time TIMESTAMP,

Bid\_end\_time TIMESTAMP,

Price int,

PRIMARY KEY (Bidon\_id)

);

CREATE TABLE Deliver\_to(

Deliver\_id int NOT NULL AUTO\_INCREMENT,

Tracking\_num int NOT NULL,

Order\_id int NOT NULL,

PRIMARY KEY (Deliver\_id)

);

## 3.3 Technology Survey: For the front-end, we would like to use the common techniques such as HTML/CSS and Javascript to develop our web, and we may use Bootstrap and AngularJS to help us easily design our dynamic and scalable web pages.

Bootstrap is the most popular front end framework in the recent time. It is sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript. Using bootstrap will allow the team to quickly mockup a simple yet beautiful looking website quickly, and allow the team to focus its effort in designing the logic and control flow of the League of Legends store site. The Bootstrap's CSS library is extensive and reliable, and many of the team members have used it before. Thus, Bootstrap was an obvious choice to use.

AngularJS is a popular JavaScript framework that uses the MV\* paradigm to design scalable web apps. It is a powerful web tool that helps developers reduce the amount of code for a project and offers powerful functions that help developers organize and design a website's codebase. The team choose AngularJS because it is one of the most popular MV\* frameworks available and members of the team have used it before.

For the back-end, we would like to use the open source software bundle XAMPP, which stands for cross-platformed, Apache, MySQL, PHP and Perl. Using the XAMPP stack will help us create a dynamic ecommerce web site and scale it in the future. Other technologies exist to create dynamic web content such as ASP.NET, and to store data such as PostgreSQL, but most members in the team have used or heard of, and expressed more interest in developing in PHP and MySQL than in other technologies. In addition no major trade offs were found between different technologies that would effect our implementation. Minor differences such as access methods, partitioning and replicating data will not affect the performance of the League of Legends store noticeably.

For this project we will be exclusively using PHP and not Perl or Python. We chose PHP because it's a general-purpose language that is especially suited to server-side web development, and generally runs on a web server. Any PHP code in a requested file is executed by the PHP run time, usually to create dynamic web page content or dynamic images used on websites or elsewhere. It can also be used for command-line scripting and client-side graphical user interface (GUI) applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). Most web hosting providers support PHP for use by their clients. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

The down side for PHP is that it lacks support for multithreading at the core language level. However, it doesn't matter to us since in our project we won't be using multithread, so it is our best choice to use PHP.

## 3.4 Populate Data for Database

While it was not required to populate the database, we have provided below a series of mock data that follow the format that will be used on the imported data.

**Table 8 - User**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | Email | Phone | UserName | Password |
| 1 | Abc123 | 123-456-7891 | David | A123456 |
| 2 | Bcd234 | 234-567-8901 | Mike | B234567 |
| 3 | Cdf345 | 345-678-9012 | John | C345678 |

**Table 9 - Buyer**

|  |  |  |  |
| --- | --- | --- | --- |
| UserID | BuyerID | DOB | income |
| 1 | 1001 | 06/21/1994 | $2341.00 |
| 2 | 1002 | 05/18/1992 | $6789.01 |

**Table 10 - Supplier**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| UserID | SupplierID | CompanyName | CompanyCatogory | Revenue | Address | RoutingNumber |
| 1 | 10001 | Arm Seller | Physical Item | $123,456.00 | Martin St. | 10084 |
| 3 | 10002 | Earth Protector | Virtual Item | $234567.00 | Mario Valley | 12354 |

**Table 11 - Credit Card**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CardNumber | Name | ExpDate | CVV | Brand |
| 4257-8975-6895-4712 | Corry Newer | 06/18 | 012 | VISA |
| 2358-8546-8974-8975 | Mew Chow | 03/17 | 755 | VISA |
| 4875-8965-4134-9653 | Susan Logen | 11/16 | 436 | VISA |

**Table 12 - Address**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AddressID | Street | AptNum | ZipCode | Permanent |
| 1 | Martin St | 1234 | 16803 | No |

**Table 13 - Zip Code**

|  |  |  |
| --- | --- | --- |
| ZipID | State | City |
| 1 | PA | State College |

**Table 14 - Rating User**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rate ID | Comment | Star Rank | SupplierID | ItemID |
| 1 | It’s really good quality | 5 | 10001 | 1 |
| 2 | Terrible | 2 | 10002 | 2 |

**Table 15 - Rating Supplier**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rate ID | Comment | Star Rank | UserID | ItemID |
| 1 | Thanks for buying our product | 5 | 1001 | 1 |
| 2 | Our pleasure | 2 | 1002 | 2 |

**Table 16 - Item**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ItemID | ItemName | CatID | SupID | Price | Description | Quantity | Picture  address | Combind ID |
| 1 | Infinity Edge | 1 | 10001 | $38 | Weapon | 1 | Infinity Edge.jpg | 203 |
| 2 | Cape | 2 | 10001 | $8 | weapon | 1 | Cape.jpg | 2031 |
| 3 | Sword | 3 | 10001 | $10 | weapon | 1 | Sword.jpg | 2032 |
| 4 | Ham | 4 | 10001 | $14 | weapon | 1 | Ham.jpg | 2033 |

**Table 17 - Property**

|  |  |  |
| --- | --- | --- |
| ItemID | Quantity | PurchaseDate |
| 1 | 1 | 10/12/2015 |
| 2 | 1 | 10/12/2015 |

**Table 18 - History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BidID | ItemID | BuyerID | BiddingPrice | TimeStamp |
| 1 | 1 | 1001 | $13 | 2013-10-13 13:00:12 |
| 2 | 1 | 1002 | $15 | 2013-10-13 13:55:12 |

**Table 19 - Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| AnalysisID | ItemID | QuantitySold | TotalRevenue |
| 1 | 1 | 3 | 114 |
| 2 | 2 | 1 | 8 |

**Table 20 - Category**

|  |  |  |
| --- | --- | --- |
| CategoryID | CategoryName | SuperCategory |
| 1 | Physical | 0 |
| 2 | Virtual | 0 |

**Table 21 - Bid On**

|  |  |  |  |
| --- | --- | --- | --- |
| BidOnID | BitStartTime | BidEndTime | Price |
| 1 | 2013-10-13 13:00:12 | 2013-10-13 13:01:12 | $13 |
| 2 | 2013-10-13 13:00:12 | 2013-10-13 13:02:12 | $18 |

**Table 22 - Deliver To**

|  |  |  |
| --- | --- | --- |
| DeliverID | TrackingNumber | OrderID |
| 1 | 123541 | 1 |
| 2 | 156358 | 2 |

**4.1 Transactions**

AddUser: In this function, we actually divided it into 2 parts, Sign up and Log in. The sign up function actually insert the information of new user to our databases, which allow this new user to shop and pay. The log in function are the function that keep user in his own shopping car, and record its data, whether history or bidding. By using of the section. Here are some images shown below (figured).

BrowsingItems: We made the categories into 3 layers, each item has a unique category ID belonging to one of these categories. When click on one of the categories, you can find all the items under that category and for more detailed category you can find more precise item related to your need.

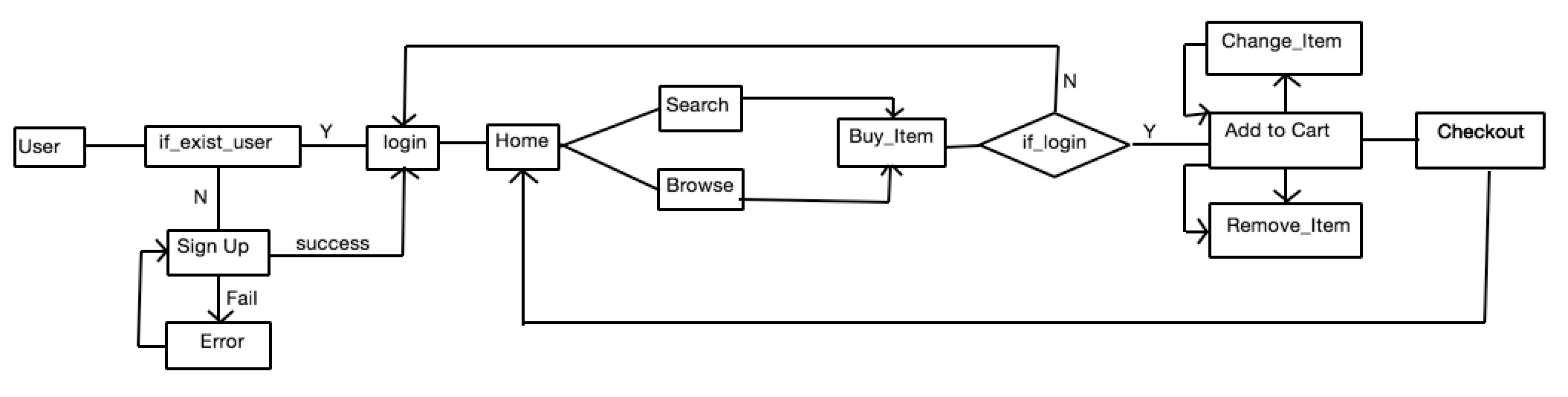
SearchingItems: Since we only sell some actual product, and the product are facing to Legue of Legent player, thus, searching function is only able to search by item names or by catogory.(figure )

buyItem: We have all price listed on each item and user able to buy multiple item by using the shopping cart. Also user should fill out the form of information of their payment methods and additional information inorder to purchase. (figure )

AuctionItem: We are doing the actual product for sale, we are not supporting the user to sell their own product. It is more likely a store than "amazon"

BidItem: All of the Item support bidding function. As designed, each bidding will has its record until end-time. And will sent a notice to the bidder who bidded it. (figure)

**4.2 Illustration**

****

We implemented HTML/CSS/JavaScript for the front end, at first we used Bootstrap to decorate the webpage, then we found it would be more interesting to mimic the layout of Amazon.com , so now our pages look like fake Amazon :)

We used PHP/MySQL for the back-end, and the apache in XAMPP as our local server. Each tuple in the User table is the a piece of user information. In SignUp component, we write PHP functions to check if the user’s input information is valid and then insert this information into the database. While in Login section, we first look up in the table to find the corresponding username and then check if its password matches the input password.

In the home page we display all the recommended items from the item’s table. While for the search function on our website, we implement SQL ‘like’ query to search the matching item name in the items table, and return and display all the corresponding results.

Since http is a stateless protocol, we used session to keep track the state and information of a user. Once the user adds an item to the shopping cart, we can know which user this shopping cart belongs to.

**4.3 Reflection**

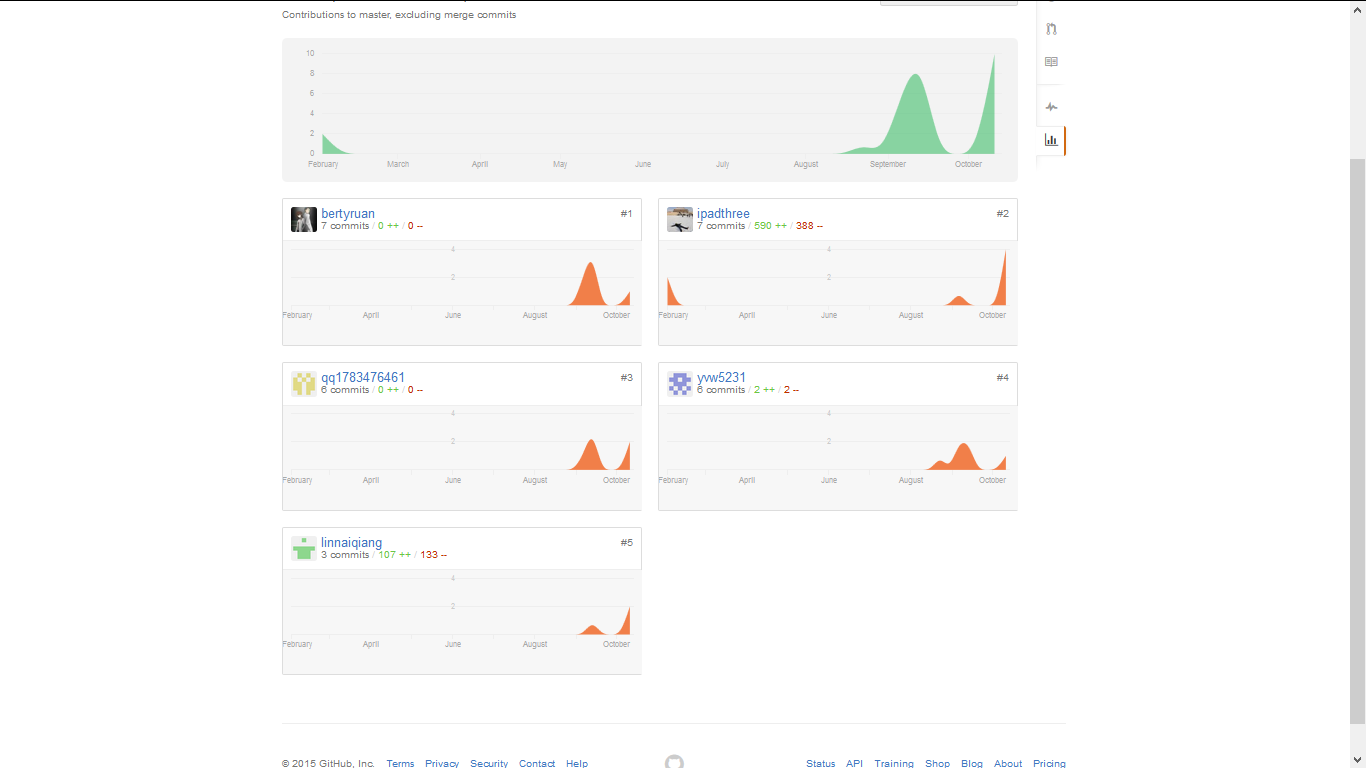
# Appendices

1. **Appendix A**

*Note*: We used an existing but empty GitHub repository which explains our extended time scale.

*Username Key:* yvw5231 - Yusheng Wang, ipadthree - Jiayi Liang, bertyruan - Berty Ruan

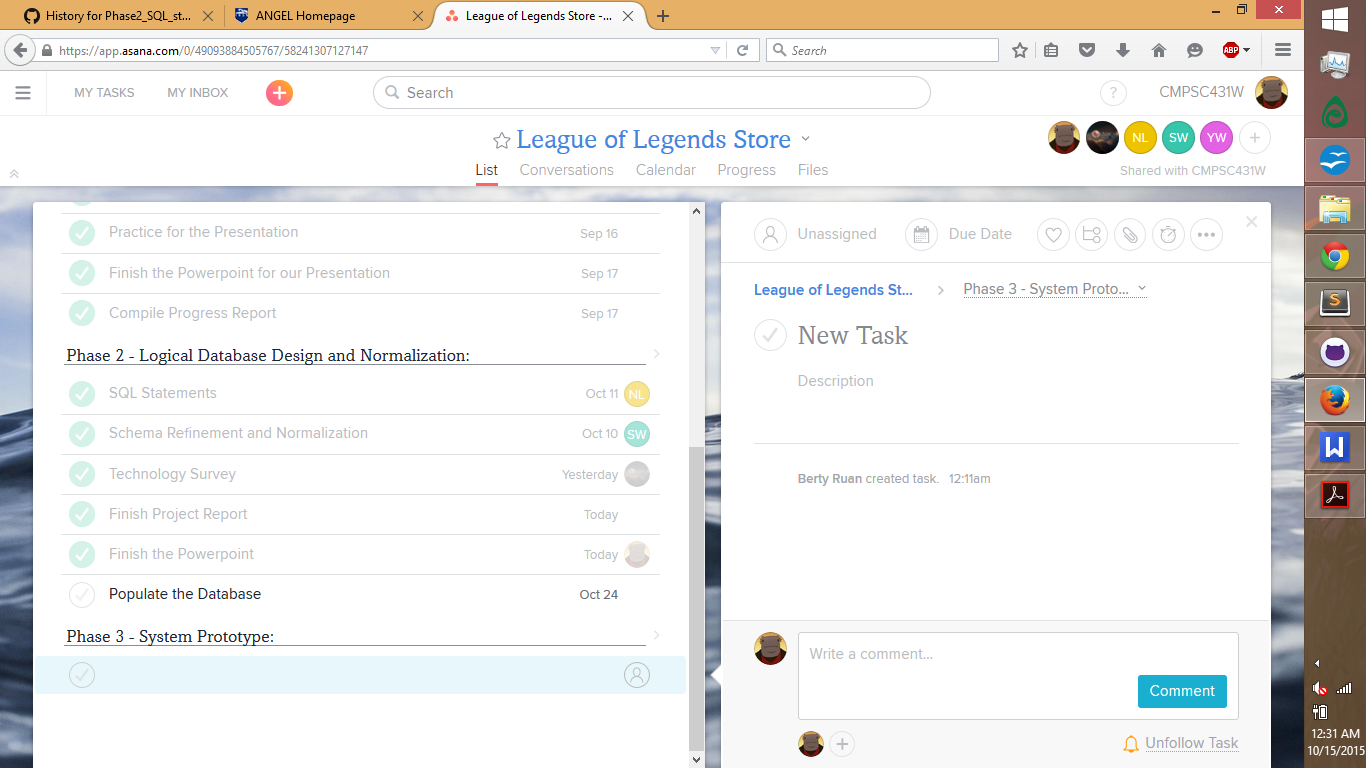
linnaiqiang - Naiqiang Lin, qq1783476461 - Shaohua Wang



**Figure 4.1Appendix B**

**Table 23 – Task Distribution**

|  |  |
| --- | --- |
| *Berty Ruan*: Technology Survey, Finish Project Report and Powerpoint | *Naiqiang Lin*: SQL Statement and Schema Refinement |
| *Shaohua*: SQL Statement and Schema Refinement | *Yusheng Wang*: SQL Statement and Schema Refinement |
| *Jiayi Liang*: SQL Statement and Schema Refinement and Technology Survey |  |



**Figure 4.2**

# 5. Conclusion

This concludes the League of Legends store requirements for Phase 2 of the project. The CMPSC 431W team has completed the three main tasks for this milestone. They included refining and normalizing the schemas, translating the schemas into SQL statements, and deciding on what technologies to use to develop the League of Legends store.

In the previous phase, the team analyzed the requirements for implementing a database for the store, and translated the requirements to an Entity Relationship model. By completing these steps first, we have set a solid foundation to create a versatile database and meet our client’s product requirements.

In the next phase, the team will be completing its inital mock up of the League of Legends store, and will focus much time to development and testing. In addition, the team will finish populating the data with users and items to prepare for the store’s initial launch date. In the final phase, the team will be describing the database transactions and completeting the system framework.