Jambalaya.com Project Report

CMPSC 431w Project Report

Aakash Sham

Dan Connor

Chris Laplante

Dan Giannone

Michael Ross

Table of Contents

List of Figures

List of Tables

Introduction

Our goal here at Jambalaya.com is to provide the world the best method for obtaining goods. We were tasked with creating an interactive website to combine the best features of online shopping giants such as Himalaya, eBay, and Amazon. Our goal is to not only accomplish this task, but exceed it by creating the most innovative, dynamic, and easy to use marketplace on the web. We have come up with a slew of ideas for this new web store, and look forward to presenting it here.

As stated, the goal of Jambalaya.com is to combine the best of other online marketplaces on the web. This means the website features a variety of items for sale. In addition to items being listed at sale price, bidding is also available if the vendor so chooses. The website features a reviewing system where users can rate the product *and* the vendor, so that future customers know what they are buying.

All products are listed into categories, which makes items easy to find/search. Jambalaya.com features items in categories such as electronics, books, and office products. With such a wide variety of items, we aim to cater to shoppers of all types. Jambalaya.com will become their one stop shop for online shopping.

The website features two types of accounts: shoppers and vendors. The average consumer will have a shopper account. This account type allows users to purchase items, bid on items, and leave reviews. The vendor account will only have the ability to list items for sale. The vendor account will have the ability to give a list price, or set the item up for bidding, with a minimum sale price.

Lastly, Jambalaya.com features a shipping service, via UPS, to ship any items purchased on the site. The website will offer a variety of shipping types, including 2-day, standard shipping, and free shipping for orders over $100.

This report will delve deeper into each the specifics of the website. Each section will explain the mechanics of the website and show how it will operate. In addition, the technical aspects will be discussed, including the database schema, ER diagram, and backend technology.

Requirement Analysis

The website Jambalaya.com is be an online marketplace that features items for sale at a set price, and those for sale by auction. The site needs to include many features, which are outlined below. Also, some additional features will be included in the site, and are also outlined below. They will include a mobile version of the site, a recently viewed items feature, a commonly purchased together feature, and statistics about the items sold on the site.

Sale Items

Jumbalaya.com is centered around the items we have for sale. That being said, we try and make it as easy as possible for both companies, and the everyday consumer to sell items on our website. We are also giving sellers some leeway in terms of how they wish to sell their items. They may choose to either sell their item for a set price, or they can put their item up for auction where users can place bids on it. To make sure that the seller does not sell their item for less than their bare minimum, the seller is not required to ship their item if a price threshold has not been met.

Sellers are required to provide a short description of the item, where the item will be shipped from, and they may provide a link to a website that elaborates further on the item. Items will then be grouped into categories, which will be described in more detail down below. User generated ratings will also be associated with items to help give users a better idea of the quality of the item.

As more than one item can be sold of the same type, there is an important distinction between items and listings. For any one type of item, there will be one entry in the item table. Any time a supplier wants to sell an item, an entry for listing will be inserted for that type of item. This was a decision we chose to make in order to simplify the layout of our database, and to make it easier for ratings of a particular order to transcend listings.

Categories

Jambalaya.com will feature an extensive list of categories for which to organize items for sale on the website. This list will also give vendors and sellers the option of categorizing their posted items in the appropriate area, making it simple and intuitive for potential customers to find. Lastly, the category tree provides a solid backbone for searching the site.

Our category tree features 3 “generations” of categories. In the appendix, we have included 2 generations of item organizations, with the third to be included in a later report. We believe a

main category along with 2 sub categories provides the best balance of providing a refined search without bogging the user down into technicalities.

Sellers

We allow all users to potentially be sellers based on a checkbox that they would indicate either during the signup phase, or anytime thereafter. Mentioned previously, sellers will be able to sell items for a set price, or allow items to be put up for auction. Similar to the way ratings can be left on items, we will allow users to rate sellers as well as leave comments. This will ensure users know exactly what that suppliers track record is in terms of getting items out on time and in good condition.

When a user indicates they wish to be a seller as well, they will be asked for the address from which their item will be shipped, if it is different from the address they indicated for their user. In the same vain, they will be prompted for a different phone number, the company name, a point of contact name, and possibly some other additional fields.

Registered Users

In order to fully interact with Jambalaya.com, you will need to register to be a user. You will still be allowed to browse our website, but if you want to buy or sell anything on the website, you will need to register. You will be prompted to identify your phone number, email address, password, name, mailing address, and at least one credit card. If you wish to be a supplier, you will be prompted for additional information. You will then be able to buy items, possibly sell items, leave ratings/comments, and do just about everything Jambalaya.com has to offer.

Ratings

There are two types of ratings: item ratings and supplier ratings. Ratings are left by users on particular items or on vendors, and are represented in the ER diagram as an ISA relationship. Ratings are left in the form of a 1-5 star rating. The aggregated average of the ratings will be displayed right next to the item itself. Users can also leave comments if they so choose to elaborate on their rating. For a user to leave a rating on a supplier, they will have had to have ordered an item from that supplier. Supplier ratings will be visible to the user in the same way that item ratings are.

Browsing

Browsing will be extremely simple for the user. Not only will they be able to directly search any item they are looking for, there will be categories to narrow down their search. Traversing categories will be simplified by being able to reorganize searches based on such factors as average customer rating, price, and popularity.

Searching

As explained above, there will be a search box where users will be able to search directly for any item they want using keywords pertaining to that item. Once they have searched for something, they can filter the results as described in browsing.

Sale

Every time a purchase is made, which can consist of multiple items, it is recorded in our database for up to six months. These transactions will be recorded in our orders entity, consisting of a timestamp, the listings that were purchased, and the quantity of each listing. There is no distinction in the orders entity between items that were purchased from bidding of full price purchasing. The distinction can be derived through a relationship back to the listings.

Bidding

When suppliers create a listing, they will be able to decide whether or not their item will be put up for bidding or not. If they choose to put it up for auction, they will be prompted to list a minimum selling price that they are willing to sell their item for. If the auction does not reach this price, they are allowed to not ship the item to the winner, and the winner will not be billed. In this case, the transaction information stored in the orders table will be deleted.

Any item up for auction, a user will be able to bid on so long as their big exceeds the most recent bid's price by no less than $2. The seller of the item is not allowed to bid on their own item. All users will be allowed to bid on an item so long as the item's auction time has not run out. This will occur promptly after an item has been up for auction for two weeks.

When an item run out of time, the seller and all bidders for that listing will be notified of the winning bid and the user that place it. The seller will receive all of the contact information of the winner so that they will be able to ship the item to them.

Order and Sales Reports

As Jambalaya.com does not actually sell any products from Jambalaya.com, we will be reporting the order reports to the suppliers of items that are low in stock. Sales reports will be sent to suppliers based on what they sold in which categories. We will receive a sales report that details sales per category/sub-category in order to determine the most profitable categories and how much money we are making. These reports will be generated once a week every week.

Delivery

When an item is sold from an auction, there are a few things to consider. Think of Jambalaya.com as an intermediary. After a winner is decided, the item will be shipped to us. Once we have received an item, we will charge the credit card of the winner we have on file. Once we know the winner has been successfully billed, we will ship the package to the winner and we will mail a check to the seller. If the package never arrives, the auction will be declared void. If the payment method does not go through, the item will be returned to the seller and the auction will once again be voided. No matter whether the auction is voided or completed successfully, the proper information regarding the transaction will be sent to both parties. In order to do this properly, we will record appropriate information regarding the delivery process in both orders and order Events.

Additional Features

The first additional feature to be included for Jambalya.com is the mobile version. The front end of the site will be built on a dynamic HTML5 framework, and this allows the site to be fluid. On a regular desktop computer, the site can be resized in any way without affecting the content on the size. When viewed from a mobile device, the screen will auto adjust to fit the screen in an optimized fashion, to ensure the best buying experience for our customers

The second feature to be added is a recently viewed items list. Information about what each user looks at will be stored in our database, so that they can see what items they have recently viewed. This feature allows users to revisit items that they have looked at incase they have forgotten or just want to revisit an item's page faster.

Another extra feature to be implemented is a commonly bought together list. When users buy more than one item, data will be collected on these items. If enough people buy 2 or more items at the same time, they will be included in a list for items that are commonly bought together. Then, future customers will now have recommendations on what items go well together based on what past users have purchased.

The last additional feature to be added will be statistics. Jambalaya.com will collect data on the common items bought together, the time (weekly and daily) when items are purchased, and what items have been recently purchased a great number of times (hot items of the week). Additional data to be collected is up for discussion, and will be included here in future reports.

ER Diagram

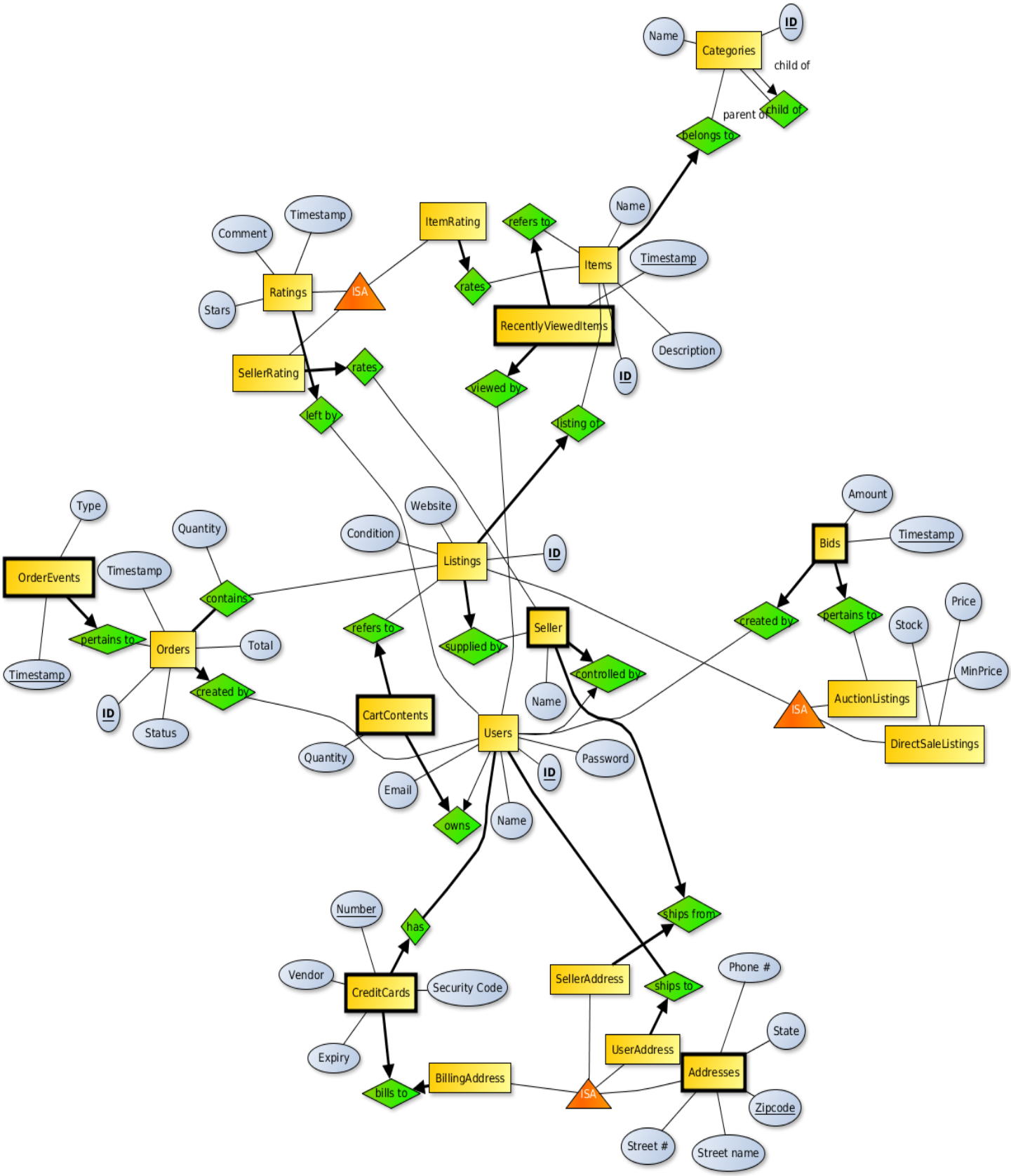
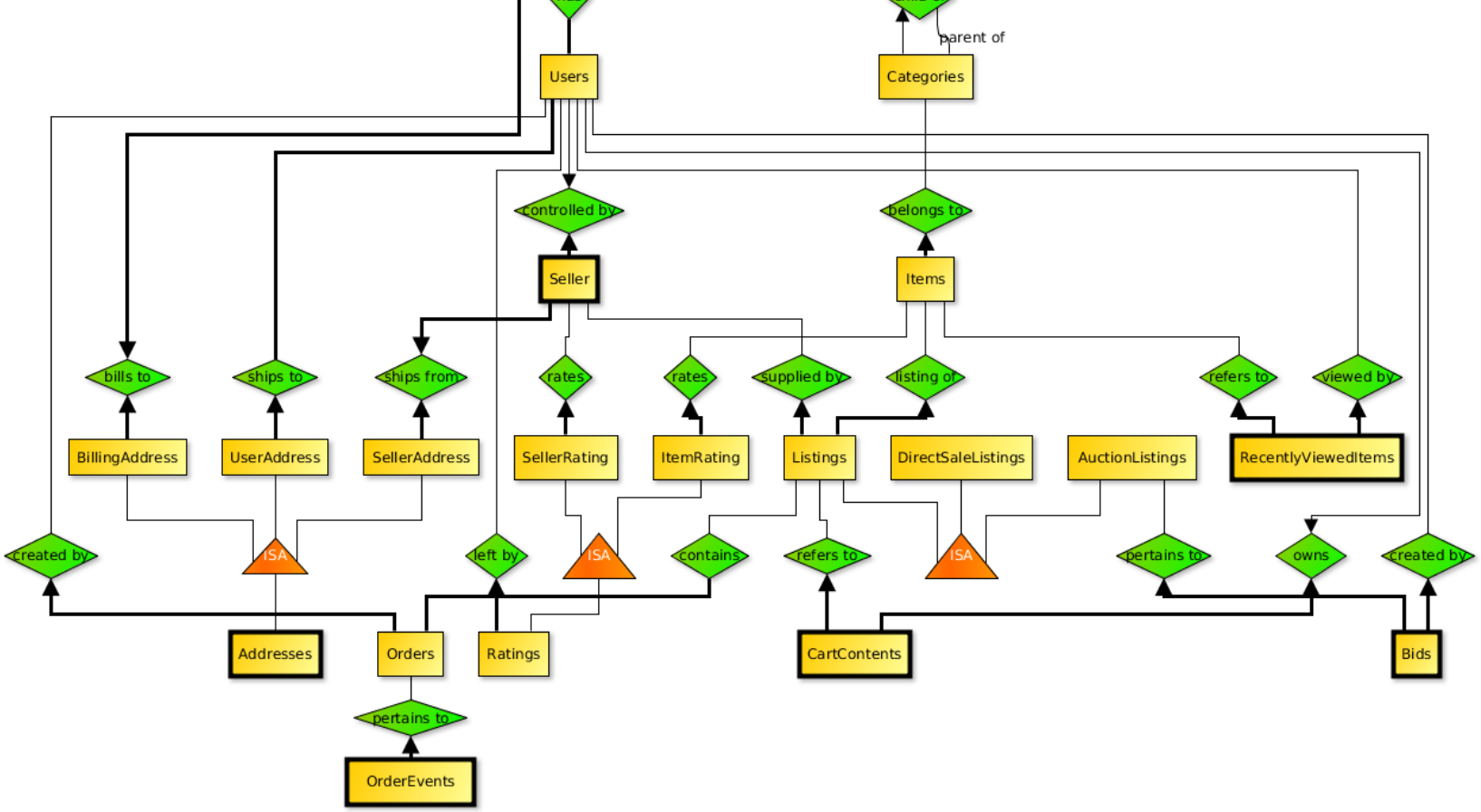


Figure 1 - ER Diagram


parent of
Users
Categories
controlled by
belongs to
Seller
Items
bills to
ships to
ships from
rates
rates
supplied by
listing of
refers to
viewed by
BillingAddress
UserAddress
SellerAddress
SellerRating
ItemRating
Listings
DirectSaleListings
AuctionListings
RecentlyViewedItems
created by
ISA
left by
ISA
contains
refers to
ISA
pertains to
owns
created by
Addresses
Orders
Ratings
CartContents
Bids
pertains to
OrderEvents

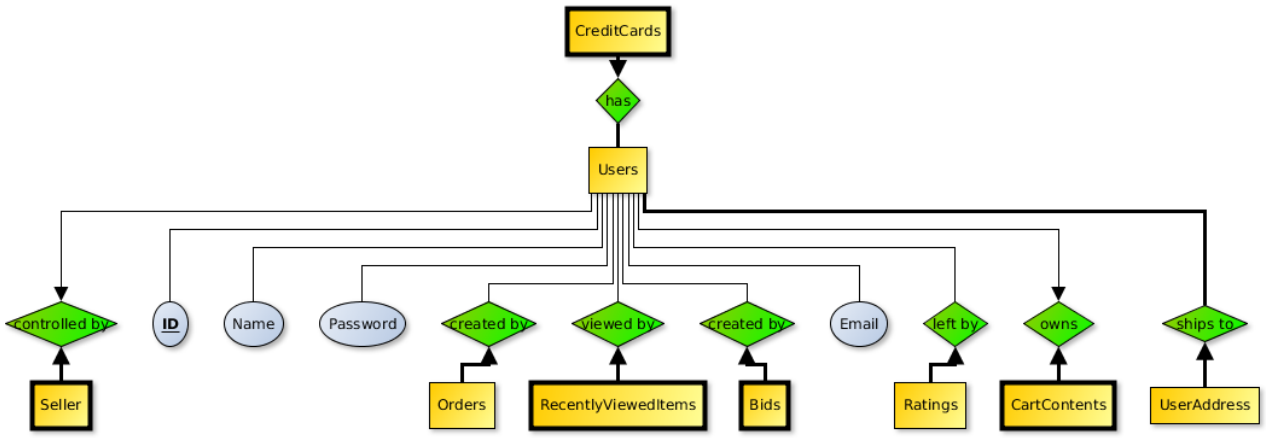
Description of ER Diagram

Seller:

- Summary: Every item on Jambalaya.com is sold by a seller, which might be a company or an individual.
- Attributes:
    - Name: Every supplier needs to have a name that is viewable to potential buyers
- Relations:
    - “supplied by” - Every item is supplied by a single seller (the creator).
    - “controlled by” - Every seller must be controlled by a single user (i.e. the user that created the seller). A user can only control up to a single seller.
    - “ships from” - Every seller has a sellerAddress entity that they ship from

Users:

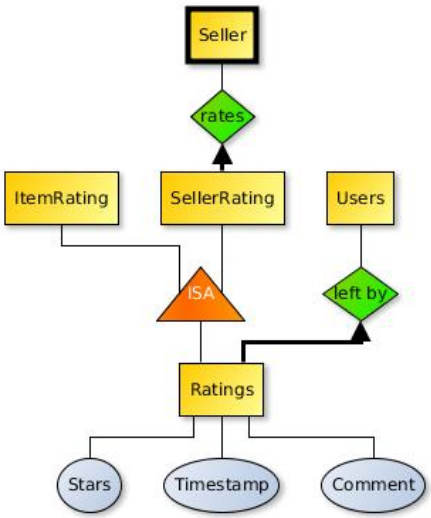
- Summary: A customer of Jambalaya.com, who will be viewing and purchasing items
- Attributes:
    - ID: A numeric identifier (automatically assigned)
    - Password: the password for the user (encrypted)
    - Name: the user’s name
    - Email: the user’s email address
- Relations:
    - “controlled by” - Every seller must be controlled by a single user (i.e. the user that created the seller). A user can only control up to a single supplier.
    - “created by” - Every user can create a bid
    - “viewed by” - A user can view their recently viewed items



    - “owns” - users own the contents of their cart

Ratings:

- Summary: Ratings can be assigned to both items and users (by users)
- Attributes:
    - Stars - The rating left by the user on a (integer) scale of 1 to 5
    - Comment - text comment is an optional part of a rating entity
    - Timestamp - time and date the rating is left
- Relations:
    - “left by” - Each rating is left by a user



Item Rating:

- Summary: ISA Rating, so inherits all of the attributes and relations associated with Ratings.
- Attributes:
    - Stars - The rating left by the user on a (integer) scale of 1 to 5
    - Comment - text comment is an optional part of a rating entity
    - Timestamp - time and date the rating is left
- Relations:
    - “left by” - Each rating is left by a user
    - “rates” - each ItemRating rates an item

Seller Rating:

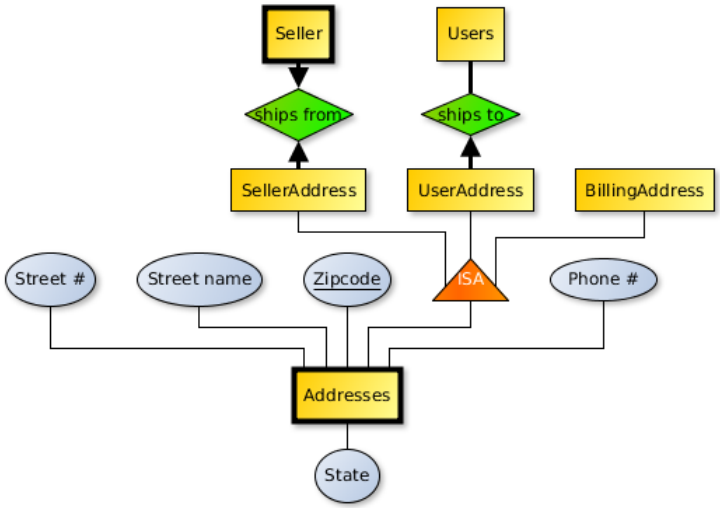
- Summary: ISA Rating, so inherits all of the attributes and relations associated with Ratings.
- Attributes:
    - Stars - The rating left by the user on a (integer) scale of 1 to 5
    - Comment - text comment is an optional part of a rating entity
    - Timestamp - time and date the rating is left
- Relations:
    - “left by” - Each rating is left by a user
    - “rates” - each sellerRating rates a seller

Addresses:

- Summary: Basic address entity that holds street, city, state etc.
- Attributes:
    - Street # - house/apartment number
    - streetName - name of the street
    - zipCode - postal code
    - State - 2 character abbreviation of the state
    - Phone # - phone number associated with an address

User Address:

- Summary: ISA Address, so inherits all of the attributes and relations associated with Addresses
- Attributes:
    - Street # - house/apartment number
    - streetName - name of the street
    - zipCode - postal code
    - State - 2 character abbreviation of the state
    - Phone # - phone number associated with an address



- Relations:
    - “ships to” - userAddresses ship to users
- ISA: A UserAddress isa Addresses

Seller Address:

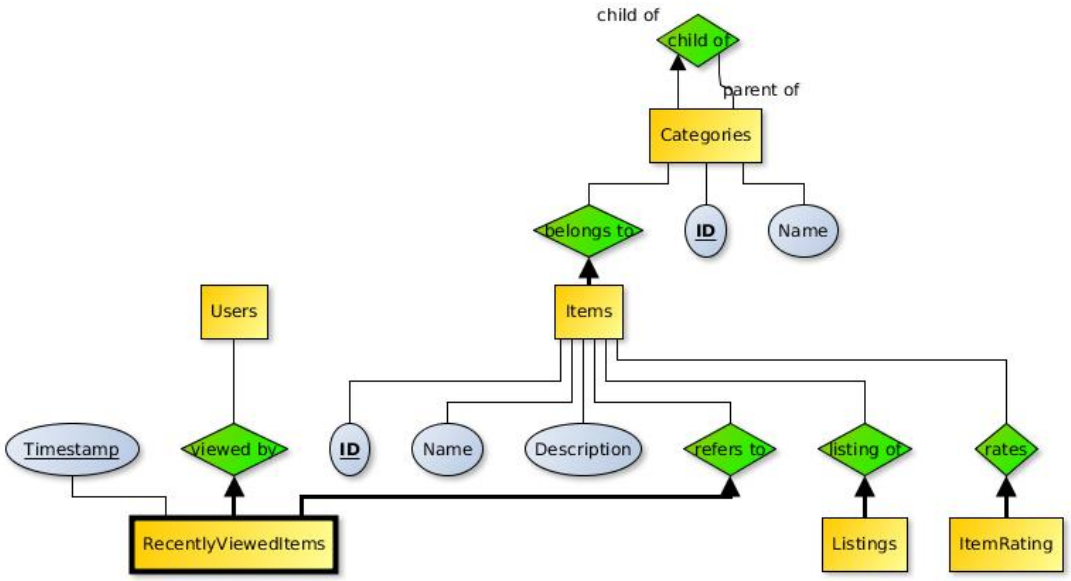
- Summary: ISA Address, so inherits all of the attributes and relations associated with Addresses
- Attributes:
    - Street # - house/apartment number
    - streetName - name of the street
    - zipCode - postal code
    - State - 2 character abbreviation of the state
    - Phone # - phone number associated with an address
- Relations:
    - “ships from” - sellerAddresses ships from seller
- ISA: A SellerAddress isa Addresses

Billing Address:

- Summary: ISA Address, so inherits all of the attributes and relations associated with Addresses
- Attributes:
    - Street # - house/apartment number
    - streetName - name of the street
    - zipCode - postal code
    - State - 2 character abbreviation of the state
    - Phone # - phone number associated with an address
- Relations:
    - “ships to” - billingAddress bills to a credit card
- ISA: A BillingAddress isa Addresses

Categories:

- Summary: Categories will be provided in which items may be listed under. This will help organize Jambalaya.com in order to make items more easily accessable.
- Attributes:
    - ID - A numeric identifier (automatically assigned)
    - name - name of category
- Relations:
    - “child of” - This relation allows us to relate categories to other categories in order to create a hierarchy. This will allow us to know both the “parent” and “child” category/categories of a particular category where applicable.
    - “parent of” - Allows a category to be the parent of another category (a subcategory)
    - “belongs to” - each item belongs to a category



Items:

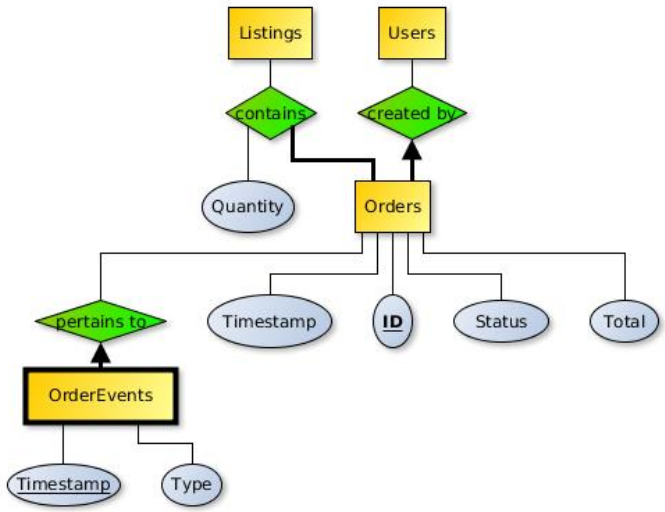
- Summary: Items are what listings contain. Multiple listings can reference the same item.
- Attributes:
    - ID - A numeric identifier (automatically assigned)
    - Description - a text description of the item (255)
    - Name - The name of the item
- Relations:
    - “rates” - ratings are associated with items. Items can have many ratings, but ratings cannot pertain to multiple items.
    - “listing of” - listings refer to items. Several listings can refer to the same item but not the other way around.
    - “refers to” - recently viewed items refer to items. several recently viewed items can refer to the same item, but not the other way around.

- “belongs to” - An item is a part of a category. Many items can be part of the same category, but not the other way around.

Listings:

- Summary: A listing is associated with an item, and is contained within an order and supplied by a seller
- Attributes:
  - ID - A numeric identifier (auto assigned)
  - Website - URL to secondary description/listing of an item
  - Condition - 1 word descriptor to describe the item in the listing (new/used)
- Relations:
  - “supplied by” - Each listing is supplied by a seller
  - “refers to” - cart contents refer to a listing
  - “owns” - a user owns the contents of a cart

Orders:



- Summary: An *order* is created by a single user, and it contains one or more items.
- Attributes:
  - ID: A numeric identifier (automatically assigned)
  - Timestamp: The date and time when the order was submitted.
  - Status: a 1 word descriptor for the current status of the order
  - Total: total price of the order
- Relations:
  - “created by” - Each order is created by exactly one user.
  - “contains” - Each order contains one or more items, with an associated quantity for each instance of the relationship.
  - “pertains to” - Every OrderEvent is related to one order

Order Events:

- Summary: An *orderevent* stems from an order
- Attributes:
  - Type: type of the order event (shipped, delivered, etc.)
  - Timestamp: time and date of the particular order event
- Relations:
  - “pertains to” - Each orderEvent pertains to one order

Cart Contents:

- Summary: each user has exactly one cart (if there is something in it). The cart can contain multiple listings.
- Attributes:

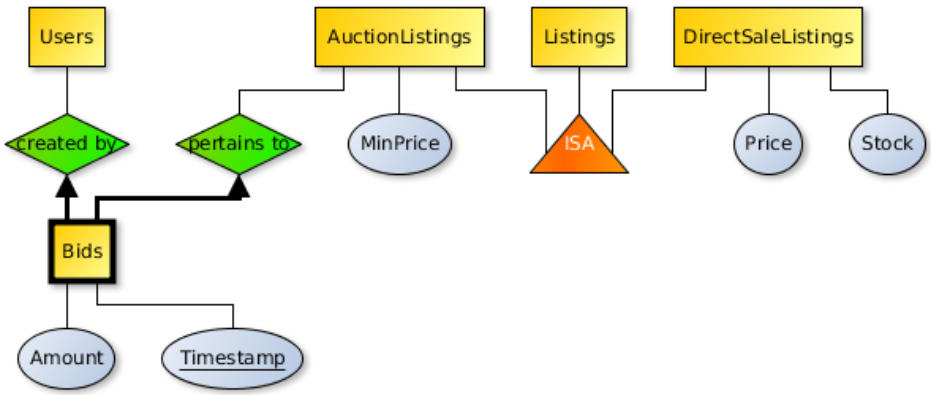
    - Quantity- a user can have several of one listing (denoted by quantity) in their cart.
- Relations:
    - “Owns”- A user owns a cart
    - “Refers to”- an entry in cart contents refers to an item.

Recently Viewed Items:

- Summary: This is a list of the last 5 items that a user has previously viewed.
- Attributes:
    - Timestamp - timestamp of the last view of the item
- Relations:
    - “Viewed by”: This relation indicates the user that viewed the item
    - “Refers to”: This relationship shows what item the user viewed

Auction Listings:

- Summary: A specialization of Listings; a Listing that is for auction.
- Attributes:
    - MinPrice: The minimum price the seller will allow the item to be sold for.
- Relations:
    - “pertains to”: Bids can be left on AuctionListings.
- ISA: An AuctionListings isa Listings



Bids:

- Summary: Users can place bids on listings that are eligible for bidding
- Attributes:
    - Amount - A float value representing the amount of the bid
    - Timestamp - The exact time the bid was submitted
- Relations:
    - ”pertains to” - This relation indicates the AuctionListing the bid is for
    - “created by” - This relation indicates the user who submitted the bid

Direct Sale Listings:

- Summary: A specialization of Listing: a listing that is available for instant purchase (i.e. not for auction).
- Attributes:
    - Price: The price of the item
    - Stock: The number of items the supplier has in stock, available for sale
- ISA: A DirectSaleListings isa Listings

Credit Cards:

- Summary: This is the credit card information of the users of the website. Multiple credit cards can be associated with one user, but not the other way around.
- Attributes:
    - Vendor: the vendor of that credit card.
    - Expiry: the expiration date of the card
    - Number: the number on the card
    - Security Code: the ccv on the back of the card
- Relations:
    - “has”: each credit card only has one user (users can have multiple cards)
    - “bills to”: each credit card will have a billing address.

Additional constraints

(not captured in ER diagram)

- Every Listing must either be an AuctionListing or a DirectSaleListing
- AuctionListings have no Stock attribute because items available for auction have an implied stock of 1.
- An ItemRating or SellerRating must contain a Stars value, but the Comment can be null.
- A user cannot leave a SellerRating on a Seller they control.
- A Category does not necessarily have to have a parent.
- Only 5 recently viewed items are maintained for each user.

Functional Dependencies

A functional dependency occurs when one attribute in a relation uniquely determines another attribute. There are many examples of this in our ER diagram:

- A user’s ID uniquely determines their name, email, orders, bids, and ratings.
- A credit card’s number uniquely determines its security code, expiry, and billing address.
- An item’s ID uniquely determines its name, description, rating, and category
- An address zipcode uniquely determines the State which the address resides in.
- An order’s ID uniquely determines the status, total, timestamp, and user of the order.

SQL Tables and Design Approach

Our team has produced a database that is in third-normal form (3NF). To accomplish this, we first converted several of our entities into weak entities, by removing the extraneous ID primary key. Next, we extracted several new entities (such as the various types of Address). Address became a new entity since users can have multiple addresses, and we should not store the Address as a field in the User table.

We started by implementing the entities involved in ISA relationships; Address, Listings, and Ratings. Logically, Address has three subtypes - BillingAddress, SellerAddress, and UserAddress. In the database however, it is represented as only two tables. That is because the subtypes differ only in relationships; their attributes are all the same. To relate a UserAddress to a User, for example, we use a helper table, UserAddress. That table has an entry for each User - Address pair. The other two types, BillingAddress and SellerAddress, are only involved in one-to-one relationships and so don't need helper tables. CreditCard and Seller both have an AddressID column, which points to an entry in Address.

The Listings relation is expressed as three tables, Listing, DirectSaleListing, and AutionListing. Listing is the parent table, containing shared attributes of both types of listings. The other two tables are children containing attributes specific to each type of listing. A foreign key in each of the child table relate back to the parent entry in the Listing table.

Ratings is implemented as two tables, SellerRating and ItemRating. Like Address, these entities differ only in relationships. Unlike Address, each contains foreign keys referring to the owning seller or item, whereas Address using foreign keys in the other direction.

We now turn our attention to representing weak entities. To implement these, we choose a compound primary key based on a partial key from the entity itself, and some identifying attributes from the parent. For example, a CreditCard can be uniquely identified by the card number (Number) and owning user (User ID). So the primary key for CreditCard is that compound key. We also enforce a foreign key on CreditCard's UserID column to ensure integrity. Deletions and updates are set to cascade, as is required for weak entities. All the other weak entities are implemented in a similar manner.

Technology Stack

Jambalaya.com is built on a few different, but powerful technologies. When designing and creating a website that needs to handle a variety of features, it is important to choose widely used technologies with a lot of features and support. Jambalaya.com is primarily built with Python using the Django web framework, with MySQL for the database.

Django is a very common framework in industry to build powerful web applications. It gives a variety of options for implementing features that make good web applications. For the front end, our group is using HTML5, CSS, and Javascript. Using these languages, we can make an easy to use web application that also looks great. Next, we will be utilizing the model-view-controller (MVC) design pattern, which is designed around separating code segments for optimal use. This is a very important design pattern for us to use, because it will allow us to use different views for our web application. Our plan is to implement a web and a mobile version of the site, so having the ability to switch views on the fly with ease is essential. Django is built in Python, and therefore our website will be too. The language is power, it offers many useful libraries, and has copious support available.

The next technology we are using is MySQL. MySQL is an free technology that will allow our team to easily design and create an effective database. It also works in tandem with Python, so adding it to the web application will be seamless. The fact that it is free also makes it accessible to all our team members.

Project Management Methodology

To work on this project, the team met on certain days of the week, for a predetermined amount of time, agreed upon by everyone. Everyone worked with everyone else helping as the project progressed along. In order to keep track of overall tasks and features, we implemented a Trello board, and marked off things that were completed, and set goals for the next meeting. Lastly, a private git repository was implemented on BitBucket.org, so that version control and code management could be implemented.

Implemented Feature List

Authentication/Logon System

Jambalaya features a logon system that is tied to vital aspects of the site's functionality. For instance, a user cannot purchase items, or view his or her recently viewed items without being logged on. The log on button is located on the top right of each page, and is easily accessible at any time. Users just type in their username and password, and then they can be on their way.

User Registration

Like the logon feature, this feature is vital for utilization of other features on the site. When visiting the site for the first time, users should, but is not required to, make an account. This process allows them to set up a username, password, sign up to be a seller, become a cajun member (our version of Amazon Prime), and input credit card information for purchases. In order to make full use of the site, users should do make use of this process, but, as stated, it is not necessary if the user just wants to have a look around at the items available on the site.

Sale Items

This, along with auction items is the core of the site. Direct sale items, as they are often referred as, are items listed on the site for the purchase by other users. Any user can become a seller and post things to the website for sale. All other users can then search/find these items and purchase them and/or leave ratings and reviews.

Auction Items

This items, along with direct sale items, are the most important aspect of Jambalaya. If a user wishes, instead of listing items for direct sale, they can list items for auction. This allows them to garner the highest possible price for an item that other users really want. They can also set a minimum price, so that they do not end up losing money for a item that did not receive many bids

Terminate Auction

This feature is tied to the auction items. When an auction is over, at the end of a certain time period, the listing for the item will come down, and other users will no longer be able to bid on that particular item.

Category Trees

Category trees are used to classify item listings. There are many different categories (as enumerated below and in the appendix) which are used. The pane for them is located on the left side of most pages on Jambalaya, making it easy for users to see and access. Just click on any category and a list of sub-categories will show. After, select a sub-category and all items listed in that section will be displayed for a user to browse

Search

Much like the category trees, search functionality make it easy for a user to find various items by typing in the name of that item. It matches full words and partial words, so even if the user does not complete the entire name of the item, he or she will still get the results that should be displayed. The search bar is located near the top right of all pages of Jambalaya, making it accessible to users whenever they need it.

Item Reviews and Ratings

In order to ensure users are purchasing quality items, other users can leave reviews on the item listings page. Each user must leave a rating of between 1 and 5 stars (in our case they are chili peppers), and optionally, can also leave a text comment review detailing their experience with the product and what they thought about it's various aspects. This feature is crucial to Jambalaya and online shopping in general.

Cart and Checkout System

The cart system is another vital process in the online shopping experience. Users can items to their cart and continue to shop, or proceed to check out. When they have all the items they want to purchase, they can click on the shopping cart icon (located in the top right of every page, so it is very easily accessible) and review their order. After, they pick a shipping address, and Jambalaya automatically coordinates with the item vendor to ship it and to send the appropriate funds. Users will not need to input a credit card at this stage, since it is required at the time of signing up for the site.

Seller Ratings

Like item reviews, users can also leave reviews on specific sellers. This helps other users see what kind of distributor the vendor is, and what they can expect in terms of customer service, shipping time, packaging, etc. when they order an item listed by that particular vendor.

User Profile Page

Each user, after signing up for the site, has a landing page on which to view various items. It shows their recently viewed items for instance (among other things) and is just a convenient method to view the site and to navigate to other pages, so that they begin to shop. It is also the best way to change any user specific settings they may want, such as addresses or credit card information.

Order Status

The order status is the method by which the users can track their packaging and status of any of their current outstanding orders. A fairly standard feature in online shopping

Telemarketer Reports

Telemarketer reports generate useful information for telemarketers, so they may target specific markets and optimize their efforts. All the information is tracked in the database, and a Jambalaya administrator can generate these reports upon vendor request.

Additional/Bonus Features

Hot Items

Hot Items are items that users have purchased more than a certain threshold each week. This list is updated anytime a user checks the list, and displays a list from the last week. The current threshold is set to 5, meaning that if more than 5 users have purchased a particular items in the last week, it will be displayed on this page.

Recently Viewed Items

Recently viewed items are items that a particular user has looked at in the last week. An account is required, since that is the method by which the items viewed are tracked. The list is

updated each time a user wishes to view it, and it automatically displayed on a user's home page.

Frequently Bought Together

Frequently bought together is a feature that shows which items are purchased at the same time. This is accomplished by taking a look at each user's cart during the checkout process, and aggregating a list of things that often appear together in the same purchase order. This information can then be displayed to all users, informing them of potential purchases that they may want or need with the current item that they are viewing.

Dynamic Mobile Layout

Built in HTML5, the frontend display can automatically shift the layout for optimal viewing, no matter the screen size or device. This ensures that users get the best possible experience when shopping with Jambalaya. We worked with the code and tested various layouts to make sure the website was functional on every screen

Statistics

The statistics the site takes are in the form of other features. Hot items, frequently bought together, recently viewed items, and the information in the seller reports are all vital to looking at trends in purchases and making informed predictions about the future of certain products. This valuable feature will ensure that Jambalaya stays ahead of its competitors.

Design Approach

The Jambalaya design team worked tirelessly to create a website with a simple, yet functional layout. They focused on making the page easy to use, but still with a masterful flow that ensures maximum effectiveness with the users. The website features a timeless look that harkens back to the golden days of ecommerce and online shopping. This look is sure to resonate with all users, making them feel at home, and ensuring that the customers know that Jambalaya is name they can trust. Below we have several figures that illustrate the website and its flawless design.

Miscellaneous

There are some other interesting bits of information that should be documented about Jambalaya.com. The database for the project contains a little over 100 unique items. Each of these items are either listed for direct sale or are up for auction. There around 50 items shown as an auction item, and the rest are direct sale listings. Currently, there are 30 users for the website, and all of them have made at least one comment or review on an item or seller. Most of them have multiple reviews up for multiple items. Each item, direct sale or auction, has at least 3 unique reviews and ratings.

Next, the website has an extensive category tree from which to search items. The first layer contains 10 different main categories. Underneath each of these is between 2 and 5 sub-categories, for a total of 32 categories. Each category contains 3 or 4 items.

Lastly, there are a few other features worth mentioning. The first is password hashing. When users input their passwords, it is not displayed. Additionally, when passwords are stored, they are not stored in plain text, but in hashed form. In case of a database breach, this will keep each user's passwords secure. Next, all fields are validated. For instance, when a user enters an email that is in an invalid format, the website will through a red flag, and indicate to the user that they may have made a mistake. Lastly, we care about protecting our users, so as a feature, we have implemented that the user will automatically be logged off after 15 minutes of inactivity.

Conclusion

Jambalaya.com is an ambitious project to say the least. It features an online marketplace where consumers can purchase and bid on items. They are able to leave reviews about the goods they purchase to help guide future customers. Users of the site are also be able sell items.

The items on the site are categorized into a vast array of sections, allowing the user to easily narrow down their search for specific items. The categorization of items also synergizes with the search feature, which customers can use to find items. The site features a “shopping cart” so that users may bulk buy certain items, or just buy multiple items of different types.

When companies sell items, reports are generated by the site admin to inform the seller on what has been sold, so that they may prepare their items to be sent to the consumer. Consumers will also get a notification indicating that their sale went through. All of these features make Jambalaya.com a reliable marketplace that consumers can trust.

Jambalaya.com also features of slew of various features that make it stand out. The site will be available in a different form for our mobile consumers. This dynamic re-rendering of the site to fit the mobile screen will make it easier for users to buy and sell items on the go. Next, Jambalaya features a commonly bought together section. This section will inform users of items that potentially go well together, based on what previous customers have bought. In addition to this section, users will also have access to a recently viewed section. This section makes it convenient for visitors to quickly get back to items they were previously considering to purchase, and reconsider. Lastly, the site will take in some basic statistics. These kind of stats can power other unique sections like “hot items of the week.”

With the plan set here, Jambalaya.com is poised to come out ahead of the competition. The technology in use provides a great backbone for the site, and will be able to scale to meet consumer demand. Lastly, our database design is insightful and well-thought out. Jambalaya.com is sure to take the competition by storm.

Appendix

Categories

Category	Subcategories
Books	Books and Calendars
Camera & Photo	Cameras, Camcorders, Telescopes
Cell Phones	Phones and Smart Phones
Consumer Electronics	TVs, Car Audio, GPS
Electronics Accessories	Audio, Video, Car Electronics, Computer, and Office Accessories
Music	CDs and Vinyl
Musical Instruments	Instruments and Recording Equipment
Office Products	Supplies, Printers, Calculators
Personal Computers	Desktops, Laptops, Tablets
Sports	Exercise & Fitness, Hunting Accessories, Team Sports, Licensed/Fan Shop, and Game Room
Tools & Home Improvement	Hand & Power Tools and Plumbing
Toys & Games	Action Figures, Dolls, Board Games, Arts, Crafts, Hobbies,
Video Games & Video Game Consoles	Game Consoles, Console Games, Accessories

SQL Statements

```
CREATE DATABASE  IF NOT EXISTS `Jambalaya`
USE `Jambalaya`;

CREATE TABLE `Address` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `StreetName` varchar(255) NOT NULL,
  `StreetNumber` int(11) NOT NULL,
  `State` varchar(45) NOT NULL,
  `Zipcode` int(11) NOT NULL,
  `PhoneNumber` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `AuctionListing` (
  `ListingParentID` int(11) NOT NULL,
  `MinPrice` float NOT NULL,
  PRIMARY KEY (`ListingParentID`),
 CONSTRAINT `fk_AuctionListing_1` FOREIGN KEY (`ListingParentID`) REFERENCES
`Listing` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Bid` (
  `Amount` float NOT NULL,
  `Timestamp` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `UserID` int(11) NOT NULL,
  `ListingID` int(11) NOT NULL,
  PRIMARY KEY (`Timestamp`,`UserID`,`ListingID`),
  KEY `fk_user_idx` (`UserID`),
  KEY `fk_listing_bid_idx` (`ListingID`),
  CONSTRAINT `fk_bid_listing` FOREIGN KEY (`ListingID`) REFERENCES `Listing` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_bid_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `CartListing` (
  `UserID` int(11) NOT NULL,
  `ListingID` int(11) NOT NULL,
  `Quantity` int(11) NOT NULL,
  PRIMARY KEY (`UserID`,`ListingID`),
  KEY `fk_cart_listing_idx` (`ListingID`),
  CONSTRAINT `fk_cart_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
```

```
  CONSTRAINT `fk_cart_listing` FOREIGN KEY (`ListingID`) REFERENCES `Listing` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Category` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL,
  `ParentCategoryID` int(11) DEFAULT NULL COMMENT 'if subcategory, this is the category_id
of the parent category',
  PRIMARY KEY (`ID`),
  KEY `fk_parent_idx` (`ParentCategoryID`),
  CONSTRAINT `fk_category_parent` FOREIGN KEY (`ParentCategoryID`) REFERENCES
`Category` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `CreditCardInfo` (
  `Number` int(11) NOT NULL,
  `UserID` int(11) NOT NULL,
  `Vendor` varchar(255) NOT NULL,
  `ExpirationDate` date NOT NULL,
  `SecurityCode` int(11) NOT NULL,
  `AddressID` int(11) NOT NULL,
  PRIMARY KEY (`Number`,`UserID`),
  KEY `fk_user_idx` (`UserID`),
  KEY `fk_cc_address_idx` (`AddressID`),
  CONSTRAINT `fk_cc_address` FOREIGN KEY (`AddressID`) REFERENCES `Address` (`ID`)
ON UPDATE CASCADE,
  CONSTRAINT `fk_cc_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `DirectSaleListing` (
  `ListingParentID` int(11) NOT NULL,
  `Price` float NOT NULL,
  `Quantity` int(11) NOT NULL,
  PRIMARY KEY (`ListingParentID`),
  CONSTRAINT `fk_DirectSaleListing_1` FOREIGN KEY (`ListingParentID`) REFERENCES
`Listing` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Item` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL,
  `Description` varchar(255) NOT NULL,
  `CategoryID` int(11) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `fk_item_category_idx` (`CategoryID`),
```

```
  CONSTRAINT `fk_item_category` FOREIGN KEY (`CategoryID`) REFERENCES `Category`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `ItemRating` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Comment` varchar(140) DEFAULT NULL,
  `Rating` int(11) NOT NULL,
  `ItemID` int(11) NOT NULL,
  `UserID` int(11) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `fk_rating_item_idx` (`ItemID`),
  KEY `fk_item_rating_user_idx` (`UserID`),
  CONSTRAINT `fk_item_rating_item` FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ID`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_item_rating_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `Listing` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Website` varchar(255) NOT NULL,
  `Condition` varchar(255) NOT NULL,
  `ItemID` int(11) NOT NULL,
  `SellerID` int(11) NOT NULL,
  `IsAuction` bit(1) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `fk_ListingItem_idx` (`ItemID`),
  KEY `fk_listing_seller_idx` (`SellerID`),
  CONSTRAINT `fk_listing_seller` FOREIGN KEY (`SellerID`) REFERENCES `Seller` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_ListingItem` FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ID`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Order` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `Total` float NOT NULL,
  `UserID` int(11) NOT NULL,
  `Status` int(11) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `fk_order_user_idx` (`UserID`),
  CONSTRAINT `fk_order_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `OrderEvent` (
  `OrderID` int(11) NOT NULL,
  `Timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `Type` int(11) NOT NULL,
  PRIMARY KEY (`OrderID`,`Timestamp`),
  CONSTRAINT `fk_oe_order` FOREIGN KEY (`OrderID`) REFERENCES `Order` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `OrderListing` (
  `OrderID` int(11) NOT NULL,
  `ListingID` int(11) NOT NULL,
  `Quantitiy` int(11) NOT NULL,
  PRIMARY KEY (`OrderID`,`ListingID`),
  KEY `fk_oi_item_idx` (`ListingID`),
  CONSTRAINT `fk_oi_listing` FOREIGN KEY (`ListingID`) REFERENCES `Listing` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_oi_order` FOREIGN KEY (`OrderID`) REFERENCES `Order` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `RecentlyViewedItem` (
  `UserID` int(11) NOT NULL,
  `ItemID` int(11) NOT NULL,
  `Timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`UserID`,`ItemID`),
  KEY `fk_rv_item_idx` (`ItemID`),
  CONSTRAINT `fk_rv_item` FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_rv_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `Seller` (
  `ID` int(11) NOT NULL,
  `UserID` int(11) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Phone` varchar(45) NOT NULL,
  `AddressID` int(11) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `fk_seller_address_idx` (`AddressID`),
  KEY `fk_seller_user_idx` (`UserID`),
  CONSTRAINT `fk_seller_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_seller_address` FOREIGN KEY (`AddressID`) REFERENCES `Address`
(`ID`) ON UPDATE CASCADE
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `SellerRating` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Comment` varchar(140) DEFAULT NULL,
  `Rating` int(11) NOT NULL,
  `SellerID` int(11) NOT NULL,
  `UserID` int(11) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `fk_sr_user_idx` (`UserID`),
  KEY `fk_sr_supplier_idx` (`SellerID`),
  CONSTRAINT `fk_sr_seller` FOREIGN KEY (`SellerID`) REFERENCES `Seller` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_sr_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `User` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Email` varchar(255) NOT NULL,
  `FirstName` varchar(255) NOT NULL,
  `LastName` varchar(255) NOT NULL,
  `Password` varchar(25) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `UserAddress` (
  `UserID` int(11) NOT NULL,
  `AddressID` int(11) NOT NULL,
  PRIMARY KEY (`UserID`,`AddressID`),
  KEY `fk_ua_address_idx` (`AddressID`),
  CONSTRAINT `fk_ua_user` FOREIGN KEY (`UserID`) REFERENCES `User` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_ua_address` FOREIGN KEY (`AddressID`) REFERENCES `Address` (`ID`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Progress Reports

26 January 2015

Group: JambalayaDotCom

Summary:

We met as a team for the first time. Discussed the basic outline and functions of the website. We created a repository for storing the project. A feature list was created, and we also discussed the additional features that we would like to add to the project. We agreed on a framework and languages to use for the project. Lastly, we brainstormed about the database design and worked on the ER diagram.

Individual Report

- Group Name: JambalayaDotCom
- Name: Aakash Sham
- Reporting Period: 1/26/15

Date	Hours	Activities
1/22/15	**1**	Met with group, discussed categories for site
1/25/15	**1**	Talked about features and additional features
1/25/15	**3**	Worked on ER diagram and database design

- Group Name:JambalayaDotCom
- Name: Dan Connor
- Reporting Period: 1/26/15

Date	Hours	Activities
1/18	**0.5**	Came up with a baseline schema for the database
1/22	**1**	Group meeting. came up with categories for inventory.
1/25	**1**	Met with group and came up with basic feature list.
1/25/15	**3**	Worked on ER diagram and database design

- Group Name: JambalayaDotCom
- Name: Michael Ross
- Reporting Period: 1/26/15

Date	Hours	Activities
1/22	**1**	Met with group and agreed on categories for the project.
1/25	**1**	Met with group to discuss additional features.
1/25/15	**3**	Worked on ER diagram and database design

- Group Name: JambalayaDotCom
- Name: Chris Laplante
- Reporting Period: 1/26/2015

Date	Hours	Activities
1/15	**0.5**	Created private Git repository
1/16	**0.5**	Created website logo
1/22	**1**	Met with group to discuss features
1/25	**1**	Met with group
1/25/15	**3**	Worked on ER diagram and database design

- Group Name: JambalayaDotCom
- Name: Dan Giannone
- Reporting Period: 1/26/2015

Date	Hours	Activities
1/22	**1**	Met with group and agreed on categories for the project.
1/25	**1**	Met with group to discuss additional features.
1/25/15	**3**	Worked on ER diagram and database design

Group Report

Group Name: JambalayaDotCom
Reporting Period: 1/26/15

Name	Total Hours (period)	Total Hours (cumulative)

Aakash Sham	5	5
Dan Connor	5.5	5.5
Michael Ross	5	5
Dan Giannone	5	5
Chris Laplante	6	6
Group Totals	23.5	23.5

9 February 2015

Group: JumbalayaDotCom

Summary:

We discussed the table structure, created the ER diagram, created the database based on the diagram, wrote the phase 1 project report, and met 3 more times since the last progress report.

Individual Report

- Group Name: JumbalayaDotCom
- Name: Aakash Sham
- Reporting Period: 2/9/15

Date	Hours	Activities
1/29/15	**2**	Met with group, discussed started on phase 1 report
1/31/15	**2**	Met with group, continued working on ER diagram and phase 1 report
2/4/15	**1**	Worked on phase 1 report
2/5/15	**3**	Worked on ER diagram and database design
2/8/15	**2**	Met with group to finalize group report and finish presentation preparation

- Group Name: JumbalayaDotCom
- Name: Dan Connor
- Reporting Period: 1/26/15 - 2/9/15

Date	Hours	Activities
1/29	**2**	Met with group to work on phase 1 report
1/31	**2**	Met with group to work on phase 1 report
2/5	**2**	Created the database in phpmyadmin based on ER diagram
2/5	**2**	Met with group to continue project report
2/8	**2**	Met with group to work on presentation and finalize report

- Group Name: JumbalayaDotCom
- Name: Michael Ross
- Reporting Period: 1/26/15

Date	Hours	Activities
1/29/15	**2**	Met with group, discussed started on phase 1 report
1/31/15	**2**	Met with group, continued working on ER diagram and phase 1 report
2/5/15	**3**	Worked on ER diagram and database design
2/8/15	**2**	Met with group to finalize group report and finish presentation preparation

- Group Name: JumbalayaDotCom
- Name: Chris Laplante
- Reporting Period: 1/26/2015

Date	Hours	Activities
1/29/15	**2**	Met with group, discussed started on phase 1 report
1/31/15	**2**	Met with group, continued working on ER diagram and phase 1 report
2/4/15	**4**	Set up the ASP.NET project file and connected to database
2/5/15	**3**	Worked on ER diagram and database design
2/8/15	**2**	Met with group to finalize group report and finish presentation preparation

- Group Name: JumbalayaDotCom
- Name: Dan Giannone
- Reporting Period: 1/26/2015

Date	Hours	Activities
1/29/15	**2**	Met with group, discussed started on phase 1 report
1/31/15	**2**	Met with group, continued working on ER diagram and phase 1 report
2/5/15	**3**	Worked on ER diagram and database design

2/8/15	**2**	Met with group to finalize group report and finish presentation preparation

Group Report

Group Name: JumbalayaDotCom
Reporting Period: 1/26/15

Name	Total Hours (period)	Total Hours (cumulative)
Aakash Sham	10	15
Dan Connor	10	15.5
Michael Ross	9	14
Dan Giannone	9	14
Chris Laplante	13	19
Group Totals	51	74.5

6 March 2015

Group: JumbalayaDotCom

Summary:

We worked on phase 2 parts of the report/project. We generated SQL statements for our ER diagram. We ensured that the relationships were in 3rd normal form, and we also worked functional dependencies. In addition, we refined our ER Diagram from the previous report, and added/fixed things that were did incorrectly for portion one of the project. Lastly, we settled on how to host our final project.

Individual Report

- Group Name: JumbalayaDotCom
- Name: Aakash Sham
- Reporting Period: 3/6/15

Date	Hours	Activities
2/19/15	1	Met with group, discussed what needed to be accomplished for phase 2
2/22/15	3	Went back through report and added in requirements analysis.
2/26/15	2	Worked on fixing other issues with phase 1 report, and setting up server for hosting final project
3/1/15	3	Worked on functional dependencies and phase 2 presentation slides
3/5/15	2	Finalized phase 2 report and presentation slides

- Group Name: JumbalayaDotCom
- Name: Dan Connor
- Reporting Period: 3/6/15

Date	Hours	Activities
2/26/15	3.5	refined entity list based on new ER diagram. Also worked on SQL statements.
3/1/15	3	Worked on functional dependencies and phase 2

		presentation slides
3/5/15	2	Finalized phase 2 report and presentation slides

- Group Name: JumbalayaDotCom
- Name: Michael Ross
- Reporting Period: 3/6/15

Date	Hours	Activities
2/19/15	**1**	Met with group, discussed what needed to be accomplished for phase 2.
2/22/15	**4**	Added in requirements analysis. Also fixed other issues with phase 1 report. Worked on refining ER diagram
2/26/15	**2**	Worked on fixing other issues with phase 1 report, and refining entity list from phase 1 based on new ER diagram
3/1/15	**3**	Worked on functional dependencies and phase 2 presentation slides
3/5/15	**2**	Finalized phase 2 report and presentation slides

- Group Name: JumbalayaDotCom
- Name: Chris Laplante
- Reporting Period: 3/6/15

Date	Hours	Activities
2/22/15	**3**	Refined ER diagram and relationships. Ensured all relationships were in 3rd normal form. Worked on SQL statements.
2/26/15	**4**	Finished all SQL Statements and refined errors from phase 1 report
3/1/15	**3**	Worked on functional dependencies and phase 2 presentation slides
3/5/15	**2**	Finalized phase 2 report and presentation slides

- Group Name: JumbalayaDotCom
- Name: Dan Giannone
- Reporting Period: 3/6/15

Date	Hours	Activities
3/1/15	**3**	Worked on functional dependencies and phase 2 presentation slides
3/5/15	**2**	Finalized phase 2 report and presentation slides

Group Report

Group Name: JumbalayaDotCom
Reporting Period: 3/6/15

Name	Total Hours (period)	Total Hours (cumulative)
Aakash Sham	11	26
Dan Connor	8.5	24
Michael Ross	12	26
Dan Giannone	6	20
Chris Laplante	17	36
Group Totals	52	136

03 April 2015

Group: JumbalayaDotCom

Summary:

We worked on building the website, populating the database, and finishing the report.

Individual Report

- Group Name: JumbalayaDotCom
- Name: Aakash Sham
- Reporting Period: 4/30/15

Date	Hours	Activities
3/19/15	4	Setup neccessary software on all computers to work on project. Including IDEs, Python, and Django.
3/22/15	3	Continued setting up project, and started working on Database, Front end, and connecting database to source code.
4/2/15	3	Worked on filling database with dummy data. Worked On drop down menus, styling of main page layout, and finished authentication system.

- Group Name: JumbalayaDotCom
- Name: Dan Connor
- Reporting Period: 4/30/15

Date	Hours	Activities
3/19/15	4	Setup neccessary software on all computers to work on project. Including IDEs, Python, and Django.
3/22/15	3	Continued setting up project, and started working on Database, Front end, and connecting database to source code.
4/2/15	3	Worked on filling database with dummy data. Worked On drop down menus, styling of main page layout, and finished authentication system.

- Group Name: JumbalayaDotCom
- Name: Michael Ross
- Reporting Period: 4/30/15

Date	Hours	Activities
3/19/15	4	Setup neccessary software on all computers to work on project. Including IDEs, Python, and Django.
3/22/15	3	Continued setting up project, and started working on Database, Front end, and connecting database to source code.
4/2/15	3	Worked on filling database with dummy data. Worked On drop down menus, styling of main page layout, and finished authentication system.

- Group Name: JumbalayaDotCom
- Name: Chris Laplante
- Reporting Period: 4/30/15

Date	Hours	Activities
3/19/15	4	Setup neccessary software on all computers to work on project. Including IDEs, Python, and Django. Also starting on base files and set-up project.
3/22/15	3	Continued setting up project, and started working on Database, Front end, and connecting database to source code.
4/2/15	3	Worked on filling database with dummy data. Worked On drop down menus, styling of main page layout, and finished authentication system.

- Group Name: JumbalayaDotCom
- Name: Dan Giannone
- Reporting Period: 4/30/15

Date	Hours	Activities
3/19/15	4	Setup neccessary software on all computers to work on project. Including IDEs, Python, and Django.

Group Report

Group Name: JumbalayaDotCom

Reporting Period: 4/30/15

Name	Total Hours (period)	Total Hours (cumulative)
Aakash Sham	10	36
Dan Connor	10	34
Michael Ross	10	36
Dan Giannone	4	24
Chris Laplante	10	46
Group Totals	44	180

30 April 2015

Group: JumbalayaDotCom

Summary:

We worked on building the website, populating the database, and finishing the report.

Individual Report

- Group Name: JumbalayaDotCom
- Name: Aakash Sham
- Reporting Period: 4/30/15

Date	Hours	Activities
4/5/15	5	Worked on various features/parts of the project
4/7/15	3	Worked on various features/parts of the project
4/9/15	3	Worked on various features/parts of the project
4/12/15	5	Worked on various features/parts of the project
4/14/15	3	Worked on various features/parts of the project
4/16/15	3	Worked on various features/parts of the project
4/19/15	5	Worked on various features/parts of the project
4/21/15	3	Worked on various features/parts of the project
4/23/15	3	Worked on various features/parts of the project
4/26/15	5	Worked on various features/parts of the project
4/28/15	3	Worked on various features/parts of the project
4/29/15	2	Worked on various features/parts of the project
4/30/15	2	Worked on various features/parts of the project

- Group Name: JumbalayaDotCom
- Name: Dan Connor
- Reporting Period: 4/30/15

Date	Hours	Activities
4/5/15	5	Worked on various features/parts of the project
4/7/15	3	Worked on various features/parts of the project
4/9/15	3	Worked on various features/parts of the project
4/12/15	5	Worked on various features/parts of the project
4/14/15	3	Worked on various features/parts of the project
4/16/15	3	Worked on various features/parts of the project
4/19/15	5	Worked on various features/parts of the project
4/21/15	3	Worked on various features/parts of the project
4/23/15	3	Worked on various features/parts of the project
4/26/15	5	Worked on various features/parts of the project
4/28/15	3	Worked on various features/parts of the project
4/29/15	2	Worked on various features/parts of the project
4/30/15	2	Worked on various features/parts of the project

- Group Name: JumbalayaDotCom
- Name: Michael Ross
- Reporting Period: 4/30/15

Date	Hours	Activities
4/5/15	5	Worked on various features/parts of the project
4/7/15	3	Worked on various features/parts of the project
4/9/15	3	Worked on various features/parts of the project
4/12/15	5	Worked on various features/parts of the project
4/14/15	3	Worked on various features/parts of the project
4/16/15	3	Worked on various features/parts of the project

4/19/15	5	Worked on various features/parts of the project
4/21/15	3	Worked on various features/parts of the project
4/23/15	3	Worked on various features/parts of the project
4/26/15	5	Worked on various features/parts of the project
4/28/15	3	Worked on various features/parts of the project
4/29/15	2	Worked on various features/parts of the project
4/30/15	2	Worked on various features/parts of the project

- Group Name: JumbalayaDotCom
- Name: Chris Laplante
- Reporting Period: 4/30/15

Date	Hours	Activities
4/5/15	5	Worked on various features/parts of the project
4/7/15	3	Worked on various features/parts of the project
4/9/15	3	Worked on various features/parts of the project
4/12/15	5	Worked on various features/parts of the project
4/14/15	3	Worked on various features/parts of the project
4/16/15	3	Worked on various features/parts of the project
4/19/15	5	Worked on various features/parts of the project
4/21/15	3	Worked on various features/parts of the project
4/23/15	3	Worked on various features/parts of the project
4/26/15	5	Worked on various features/parts of the project
4/28/15	3	Worked on various features/parts of the project
4/29/15	2	Worked on various features/parts of the project
4/30/15	2	Worked on various features/parts of the project

- Group Name: JumbalayaDotCom

- Name: Dan Giannone
- Reporting Period: 4/30/15

Group Report

Group Name: JumbalayaDotCom
Reporting Period: 4/30/15

Name	Total Hours (period)	Total Hours (cumulative)
Aakash Sham	45	81
Dan Connor	45	79
Michael Ross	45	81
Dan Giannone	45	69
Chris Laplante	45	91
Group Totals	225	401