LionAuction Phase 1 Report

By: Matthew Kelleher

Table of Contents

List of Figures

List of Tables

1. Introduction

This is a phase 1 project report on the start-up project LionAuction. LionAuction is an online e-commerce platform that uses auctions to buy and sell goods between members of Lion State University and local vendors. The founder Hilbert Dude believes that the success of this project can lead to similar web applications launching on university campuses around the world and help boost the local economy on and around campuses. This report will go over four parts to lay a foundation for creating LionAuction: a requirement analysis, conceptual database designs, a technology survey, and logical database designs and normalization. The requirement analysis will go into the applications' data storage and expected functionality, the conceptual database design will lay out the application's data and constraints into entity-relationship models, the technology survey will drive into possible web and database application technologies to be used, and finally, the logical database designs and normalization will finalize the conceptual database designs from before into refined schema for the database in LionAuction.

2. Requirement Analysis

This section will go over the functions and data requirements for LionAuction that will be used to implement the web application. It is broken down into 6 categories, registered user types, products, bids, categories, reviews, and analytics.

2.1 Registered Users

There are three different types of users who can register with LionAuction, this includes, Bidders, Sellers, and HelpDesk. Each different type has its own unique information and functionality associated with them as well as information that is tracked for all users. All users have a userID attribute (which is the email that the user used to signup for LionAuction and a password attribute that is created on signup by the user that is used to log in to their respective accounts. This is stored in the registered_users database that is updated every time a new account is created, as seen in **Table 1**. userID is used as the key for getting the password to authenticate the user when they are logging in. Functionally when any user signs into their account they have access to change their password for their account updating their password attribute in the registered_users database. In addition to this, each type of user has additional information and functionality as follows:

Table 1 – registered_users Database Example

userID	**password**
johndoe@lsu.edu	abc123
larrysmith@gmail.com	cba456

Bidders –

Bidders can be any student at Lion State University. In addition to a userID and password, we hold additional information for bidders that they input when they signup and store this information in the bidders database, as seen in **Table 2**. This includes the following attributes: an email address (email_address), the users name (name), an address referenced by an id (address_id), a phone number (phone_number), a credit card referenced by an id (creditcard_id), the user's major (major), the user's age (age), the user's gender (gender), and finally the user's annual income (annual_income). The address_id attribute is a unique integer key to the address database that stores the street, city, state, and zipcode of the user (as seen in **Table 3**), and the creditcard_id attribute is a unique integer key to the creditcards database that stores the user's credit card type (type), number (number), and expiration date (expiration_date) (as seen in **Table 4**). This can all be accessed by giving the userID key to get a specific user's information.

Functionally a bidder user can view items up for auction and place bids on them this is further explained later in sections 2.2 and 2.3. If they win the bid, then the amount is taken out of their listed credit card. They are also able to send messages to the seller of the item and rate and review the seller after they win an auction. Users can then see when viewing the seller, their rating and reviews, this is further explained in section 2.5. Additionally, when a bidder is signed into their account, they can update all personal information attributes associated with their account, excluding their userID. If a user would like to update their userID they need to put in a form to the help desk to get it changed, as HelpDesk users are the only ones able to change the userID attribute.

If a bidder would like to put their own items up for action, they would need to submit an application to the help desk for approval to become a Seller.

Table 2 – bidders Database Example

userID	**email_address**	**name**	**address_ id**	**phone_ number**	**creditcard_ id**	**major**	**age**	**gender**	**annual_ income**
johndoe@lsu.edu	johndoe@lsu.edu	John Doe	1	(123)123-1234	1	Computer Science	20	male	5000
larrysmith@gmail.com	larrysmith@gmail.com	Larry Smith	2	(456)456-4567	2	Accounting	19	male	2000

Table 3 – addresses Database Example

address_id	street	city	state	zipcode
1	University Ave	Sunny Town	PA	12345
2	College St	Rainy Town	CA	67891

Table 4 – creditcards Database Example

creditcard_id	type	number	expiration_date
1	Discover	1745 2856 3869 3759	10.23
2	Visa	5826 4927 6549 3743	04.24

Sellers –

Sellers are students from Lion State University that applied to the help desk for approval to become a seller or an approved local business vendor that put up items on LionAuction to bid on. A student can be both a seller and a bidder, while local business vendors can only be a seller. In addition to a userID and password, we hold additional information for sellers that they put on their application and store this information in the sellers database, as seen in **Table 5**. This includes the following attributes: the seller's bank routing number (bankrouting_num), bank account number (backaccount_num), user rating (rating), and LionAuction account balance (balance). Additionally, if the seller is an approved business vendor, we also keep track of their business's name (business_name), an address referenced by an id (business_address_id), and the business's customer service phone number (customerservice_phonenumber), where the business_address_id

attribute is a unique integer key to the address database described above and seen in **Table 4**. If the seller is a student these additional attributes don't have a value.

Functionally a seller user can put items up for auction this is further explained later in section 2.2. When the item is sold, then the amount of the highest bid is put in the seller's listed bank account using the bank account and routing number. They are also able to see all of their reviews and ratings from senders, this is further explained in section 2.5. Additionally, when a seller is signed into their account, they can update all personal information attributes associated with their account, excluding their userID. If a user would like to update their userID they need to put in a form to the help desk to get it changed. Also, if a seller wants a category added for a product listing they would need to put in a request to the help desk to get it added. If a seller leaves LionAuction then all their listed products are removed as well.

Table 5 – sellers Database Example

userID	**bankrouting_num**	**bankaccount_num**	**rating**	**balance**	**business_name**	**business_address_id**	**customerservice_phonenumber**
johndoe@lsu.edu	123456789	132435465768	5	500	NULL	NULL	NULL
marialane@gmail.com	987654321	978675645342	1	1500	cool computers inc	3	(832)287-7430

HelpDesk –

HelpDesk are hired members of the LionAuction IT staff that provide services to users. They do not have any additional data from a normal registered user. Functionally they can change any user's userID when a request is submitted, upgrade a bidder to a seller after a seller submits an application, and add additional categories to LionAuction if a seller requests one. HelpDesk is also in charge of marketing analytics which is further described in section 2.6.

2.2 Products

Products are any item that is put up for auction by a seller. Every item belongs to one category out of the list of categories that were approved by the HelpDesk. When a seller adds a product to LionAuction it is added to the product database as seen in **Table 6**. The database keeps track of a unique product id (pid), as well as the products name (name), description (description), category (category), reserve price (reserve_price), the date the auction stops (auction_stop_time), the value of the current max bid (max_bid), the userID of the seller of the product (seller_id), and the userID of the user with the highest bid (max_bidder_id). Functionally if a seller leaves LionAuction all products they are selling are removed from the product database. Once an item is sold it is moved into the archive database that holds the product information once it was sold, this is the same as **Table 6**.

Table 6 – product Database Example

pid	name	description	category	reserve_price	auction_ stop_ time	max_bid	seller_id	max_ bidder_ id
1	LSU Shirt	Men's medium LSU t-shirt	clothes	20	3/14/23	30	johndoe@lsu.edu	larrysmith@gmail.com
2	White Earbuds	White wired apple earbuds	electronics	50	4/1/23	70	marialane@gmail.com	johndoe@lsu.edu

2.3 Bids

For bids, we do not hold any additional data as the bidding amount is stored in the product database as seen above in **Table 6**. Functionally, any bidder can place a bid on a product put up by sellers for at least $1 more than the highest current bid. Once a product reaches the end of the auction the product goes to the bidder with the highest bid. However, if the highest bid is not at least equal to the reserve price the seller has the option to take the product off the auction. No matter if the product sells or not, the product goes into the archive database once it is no longer up for auction.

2.3 Categories

A category is a general group of similar products. When a seller posts a product they have to assign it a category from the available predefined list of categories by the HelpDesk. If the category the seller want is not listed then, they can send a request to HelpDesk for approval to get it added

to the list. The number of products within a category is unlimited. A category can also have subcategories.

2.4 Reviews

When a bidder buys an item from a seller, they can leave up to a 5-star rating and can add additional information in a written review. This information is then stored in a review database (as seen in **Table 7**). In the database, we store the userID of the seller being reviewed (userID), a unique review id (review_id), the rating out of 5 stars (rating), and the written review (review). Functionally we keep all reviewers anonymous, but all users can see all of a seller's ratings and reviews when they look at products the seller has up for auction. Additionally, the average of all the ratings of a seller is stored in the seller database as described above. When we want to get the reviews of the user we find all reviews with a matching userID to the seller.

Table 7 – review Database Example

userID	**review_id**	**rating**	**review**
johndoe@lsu.edu	1	5	The product was in excellent condition!
marialane@gmail.com	2	1	The seller sold me an empty box, there was no product inside!

2.6 Analytics

LionAuction stores analytical data for marketing, this is controlled by HelpDesk users. Various analytics are stored in the analytics database as seen in **Table 8**. These include the number of registered users (num_users), the total number of products sold all time (num_products_sold), the

total number of products posted all time (num_products_posted), the average age of all bidders (avg _age), and the average income of users (avg_income). These values are updated when a new user joins or a product is listed and sold.

Table 8 – analytics Database Example

num_users	num_products_sold	num_products_posted	avg _age	avg_income
50	10	12	20	2000

3. Conceptual Database Design

The following will lay out the application's data and constraints into entity-relationship models. First, we will go over different entity sets, then we will connect them all into one entity-relationship model for all LionAuction.

3.1. Registered Users

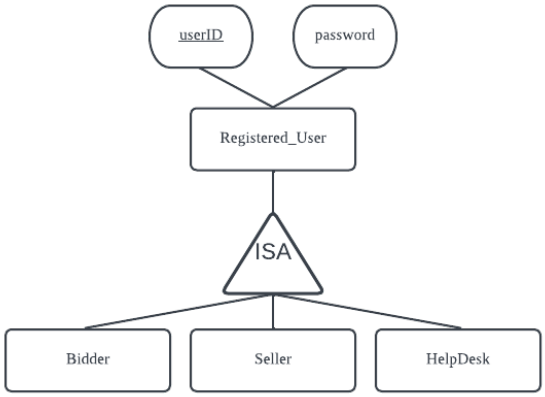

Figure 1 – Registered Users ER Model

The above figure models the registered user relationships and data. Starting with the Registered_User entity, it stores the attributes of the user's userID and password. The userID is underlined as the key as it's what's used to get the password of the user. We then branch off to the three different types of users, Bidder, Seller, and HelpDesk. We use a class ISA hierarchy to model that these three different types of users are all types of registered users with a userID and password.

3.2. Bidder Personal Data

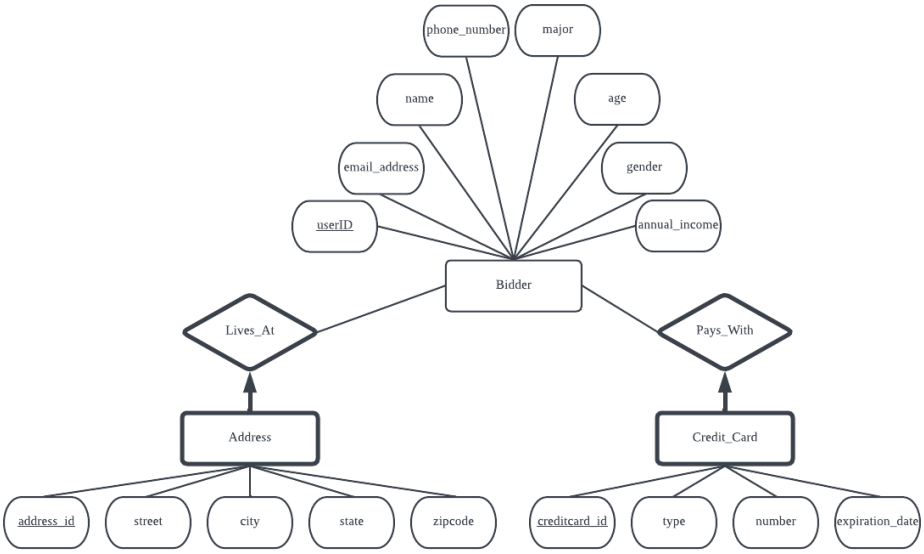

Figure 2 – Bidder Personal Data ER Model

The above figure models what is stored for a Bidder's personal data. There are eight different attributes that are stored as well as two weak entity sets. The eight attributes include the bidder's user ID (userID), email address (email_address), name (name), phone number (phone_number), major (major), age (age), gender (gender), and annual income (annual_income). In addition, the two weak entities are the user's address and credit card. The address entity stores a user's address with the attributes, street, city, state, and zip code. The credit card entity stores the bidder's credit card type, number, and expiration date. These being weak entries over attributes allows users to have multiple addresses and credit cards associated with their account each with their own attributes. Plus, when a Bidder deletes their account, all addresses and credit cards associated with their account will also be deleted along with all personal data. Both of these have total participation

requiring each user to have at least one address and credit card with their account. If you need to get a Bidder's data it can be accessed with the Bidder's userID as the key and the Bidder's addresses and credit cards can be accessed with the additional partial key of address_id and creditcard_id respectively.

3.3. Seller Personal Data

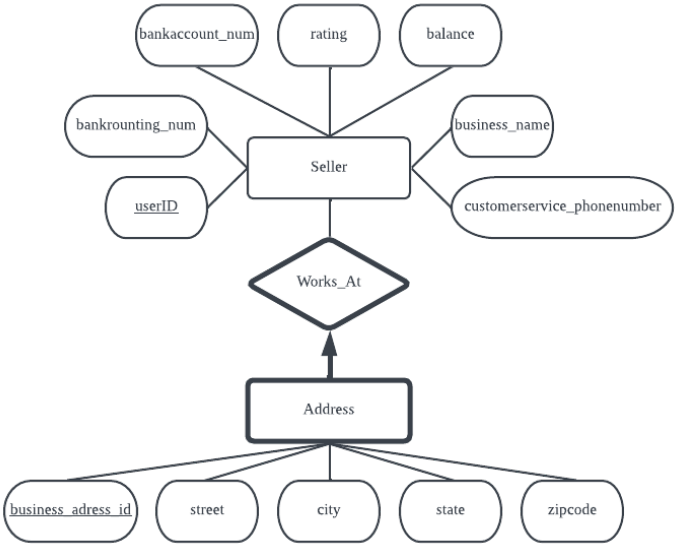

Figure 3 – Seller Personal Data ER Model

The above figure models what is stored for a Seller's personal data. There are seven different attributes that are stored as well as a weak entity set. The seven attributes include the seller's user ID (userID), bank routing number (bankrouting_num), bank account number (bankaccount_num), average rating (rating), LionAuction balance (balance), name of their business (business_name),

and the businesses customer service phone number (cuntomerservice_phonenumber). In addition, there is a weak entity storing the seller's business address with the attributes, street, city, state, and zip code. This being a weak entity over an attribute allows users to have multiple business addresses with their account each with their own attributes. Plus, when a Bidder deletes their account, all addresses associated with their account will also be deleted along with all personal data. This weak entity has total participation requiring each user to have at least one business address with their account. If you need to get a Seller’s data it can be accessed with the Seller’s userID as the key and the Seller’s business addresses can be accessed with the additional partial key of business_address_id.

3.4. Product Data

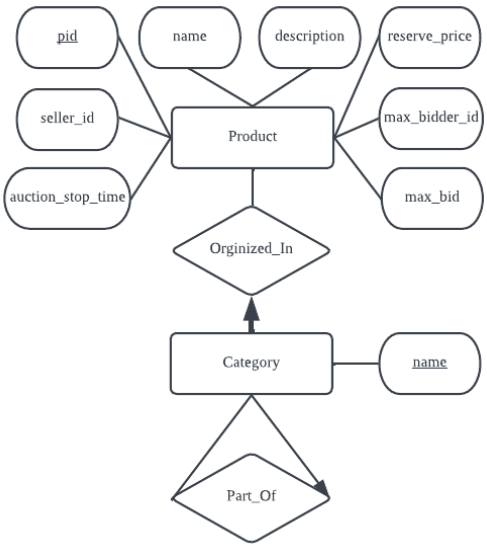

Figure 4 – Product Data ER Model

The above figure models what is stored for product data. There are 8 attributes associated with the product entity. This includes the product's unique id number (pid) which is used as a key for the product, name (name), description of what is it (description), its reserve price (reverve_price), the current max bid (max_bid), the userID of the current max bidder (max_bidder_id), the userID of the seller (seller_id), and when the product it on auction to (auction_stop_time). Additionally, products have the relation of being organized into categories that have a name attribute. This has a total participation key constraint because every product is in exactly one category but there are unlimited products in a category. There is also a key contain for categories being organized in themselves as a category can recursively be in exactly one other category infinitely.

3.5. Analytics

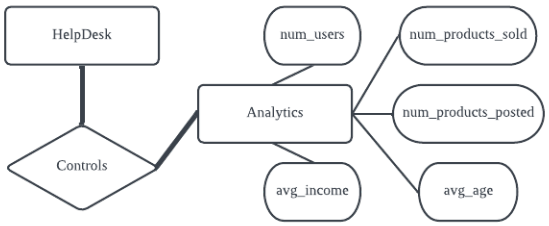

Figure 5 –Analytics ER Model

The above figure models how the analytics of LionAuction are controlled by the HelpDesk. HelpDesk is shown as having total participation with the LionAuction analytics as every HelpDesk user can control the application's analytics. The analytic entity has its own attributes that it keeps

track of for all of the LionAuction. These attributes include the number of total users signed up for LionAuction (num_users), the total number of products that have been sold all-time (num_products_sold), the total number of products that have been posted all-time (num_products_posted), the average age of all users (avg_age), and the average annual income of all users (avg_income).

3.6. Reviews

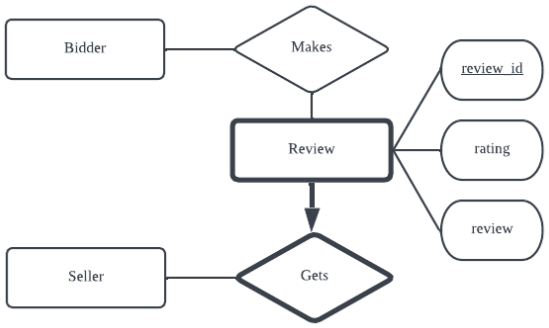

Figure 6 – Review ER Model

The above figure models how the Seller reviews are made and stored. Sellers have reviews as a weak entity as this allows a Seller to have mutable reviews written about them, plus when a Seller leaves LionAuction all of their reviews are deleted as well. The review entity has its own attributes of a partial key to get a particular review (review_id), a rating out of 5 stars (rating), and a descriptive review (review). A bidder also has a relationship of making reviews for Sellers. This has no constraints as a Bidder can write as many to as few reviews as they would like.

3.7. Auctions

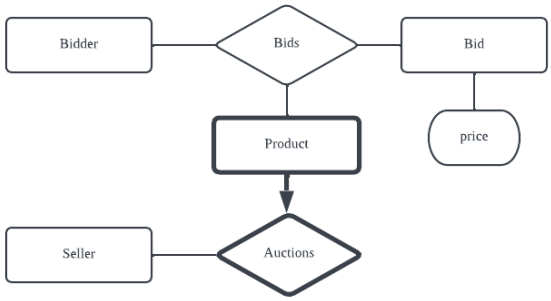

Figure 7 – Auction ER Model

The above figure models how the Auctions take place in LionAuction. Sellers have Products as a weak entity as this allows a Seller to have mutable products up for auction, plus when a Seller leaves LionAuction all of their products listed are deleted as well. The product entity and its data are already explained above in section 3.4. Additionally, a bidder has a relationship of placing bids on a listed product up for auction. This is connected to a bid entity that has a price attribute so a Bidder can place as many bids on a product as they would like.

3.8. LionAuction Full ER Model

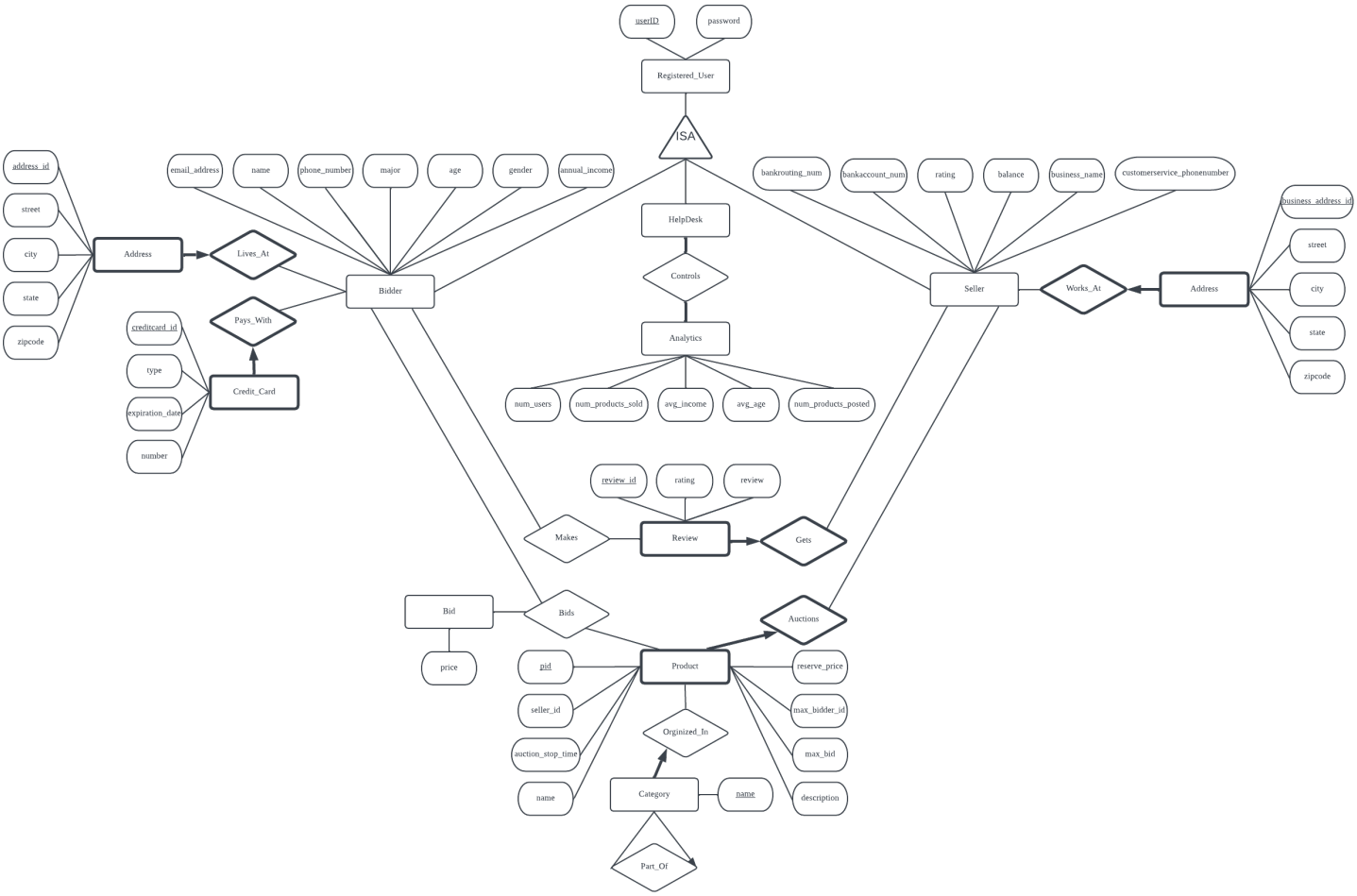

Figure 8 – LionAuction Full ER Model

The above figure models the entirety of the LionAuction data and constraints that have been shown and described in sections 3.1 through 3.7 into one ER model by combining like entities.

4. Technology Survey

In the IT industry, it is always good to keep up to date on current market trends and technologies. Because of this, we will be looking at some alternatives to using Flask, Python, PyCharm, and SQLite for creating LionAuction and their pros and cons. We will also discuss the impact and relevance of advancing technology trends to the computer science fields and society.

4.1. Web Programming Frameworks

Alternatively to Flask, Node.js is an excellent choice if you need to build high-performance applications with complex user interfaces. Where Flask integrates HTML and Python for web programming, Node.js uses JavaScript for the front and back end of an application. This has the added benefit of having all your code in one language to avoid confusion especially working in a group and can allow for more complex designs. JavaScript also tends to be faster and renders HTML templates faster as well, allowing us to optimize the speed of our web application. However, Flask is a lightweight and flexible framework that is easy to learn and use. This is perfect for programmers without much web programming experience, who know Python and are creating small web applications. In our case, we will be building a prototype of LionAuction and want to keep it as simple as possible, making Flask more beneficial to us than Node.js. We won't care about the speed of the application and will not be doing anything too complex making Flasks simplicity more what we are looking for in building our LionAuction prototype.

4.2. Programming Languages

Alternatively to Python, Java is an excellent choice for building web applications, as it offers a range of libraries and frameworks that make it easy to create complex applications. Java is known for its scalability and reliability, making it a popular choice for large-scale applications that require a lot of testing. It also tends to be faster and more portable to a wide range of devices than Python. However, Python is an easy-to-learn. an easy-to-read, and versatile language that is widely used in the web development industry. Python's simplicity allows developers to quickly develop and test code, making it a good choice for rapid application development. For the LionAuction prototype, we won't care about the speed and optimization that Java gives as much as the simplicity and ease of use that Python gives. Therefore, Python will be a better language for us to use for our web application development.

4.3. Code Editors

Alternatively to PyCharm, Visual Studio Code is an excellent choice, as it provides a range of features that make it easy to write and debug code. Visual Studio Code is a lightweight and versatile code editor that supports a wide range of programming languages and frameworks. It has many features similar to PyCharm, such as code completion, debugging, and Git integration. It also has a large and active community that creates and maintains extensions for almost any language or framework. However, PyCharm is specifically designed for Python development. It

has many features that make Python development more efficient and productive, including support for web development with popular web frameworks like Flask, and built-in database management functionality. For LionAuction, PyCharm would be a better option than Visual Studio Code. Both PyCharm and Visual Studio Code can be used effectively, but PyCharm is more suitable as we are mainly working with Python, Flask, and databases. It also has advanced debugging and profiling tools that are specifically designed for Python development, where Visual Studio Code we need to find various extensions to get the same functionality we have when we first open PyCharm.

4.4. Database Management Systems

Alternatively to SQLite, MySQL is a popular choice for building web applications that require scalability and security. MySQL offers advanced security features, including encryption and access control, as well as scalability and high availability through features like replication and clustering. MySQL also supports a wide range of programming languages and frameworks, making it flexible for whatever programming language you are using. However, setting up and configuring a MySQL server can be more complex than using SQLite, and it requires more resources to run. SQLite is a lightweight and flexible database management system that is an excellent choice for small to medium-sized applications that do not require complex queries. However, SQLite does not offer the scalability or security features of MySQL, making it less suitable for large-scale applications or applications that handle sensitive data. For our prototype, SQLite will be fine as it will allow us to easily manage our database, but for the final product, we

may want to look into using MySQL because LionAuction will be managing a lot of bank transactions and credit cards which the security features could help protect.

4.5. Impact and Relevance

When it comes to impact and relevance, the computer science field and society benefit from the continued development and improvement of web programming frameworks, programming languages, tools, and database management systems. These advancements allow for the creation of increasingly sophisticated and complex web applications that improve business efficiency, streamline workflows, and enhance user experience across the world. As technology advances and new ways to implement technology become available, the development of new and innovative web programming technologies is crucial for the continued growth and success of computing, our society, and the world.

5. Logical Database Design and Schema Normalization

The following section will use the requirement analysis in section 2 and the conceptual database designs in section 3 to translate the data requirements into a logical data model that can be implemented in a database management system and eliminate data redundancy within them.

5.1. Database Schema

The following is the representation of the schema from the above ER Model for LionAuction. Every entity in the ER modal has its table where the keys of the entity are underlined and all rest are attributes. To show the relationship between the two tables they have a like key between them. Also, arrows show attribute dependencies on each other. When it comes to analytics it is static for the whole application so doesn't have instances and keys. **Figures 9-17** depict these schemas.

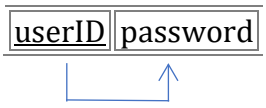

Figure 9 – Registered Users Schema

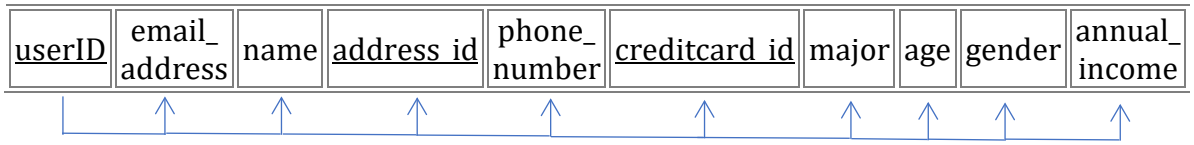

Figure 10 – Bidders Schema

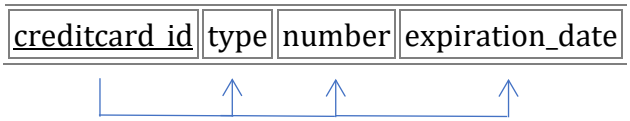

Figure 11 – Credit Cards Schema

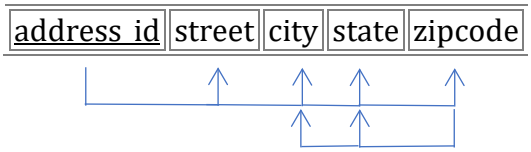

Figure 12 – Addresses Schema

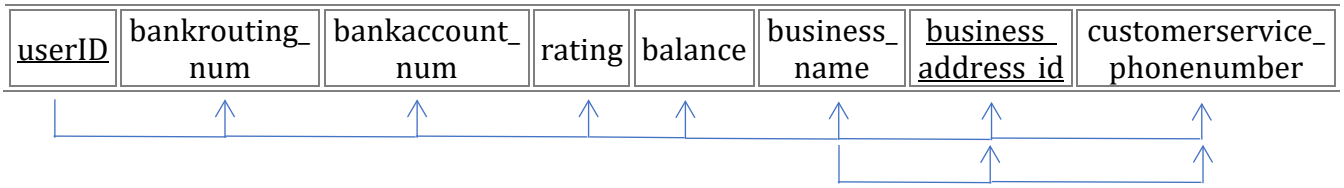

Figure 13 – Sellers Schema

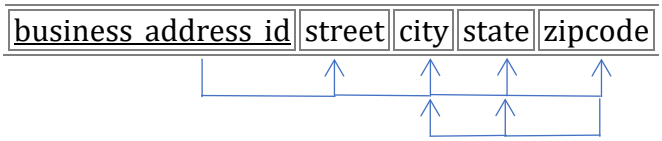

Figure 14 – Business Addresses Schema

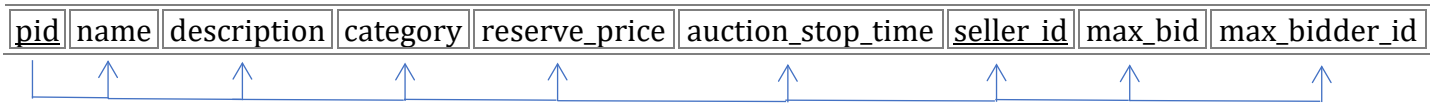

Figure 15 –Product Schema

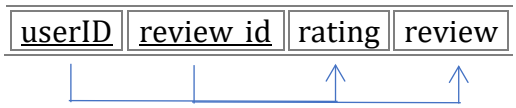

Figure 16 – Reviews Schema

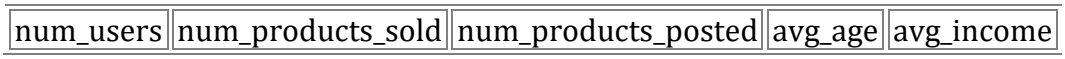

Figure 17 – Analytics Schema

5.2. Schema Normalization

Now that we have the databases schemas for LionAuction we need to refine the schema so that it reduces data redundancy to an acceptable level, we will do this by putting them into the 3rd Normal Form, while not unduly affecting performance. To reduce them to the 3rd Normal Form we need to make sure that we do not have a non-key attribute functionally determined by non-key attributes. We will fix this by decomposing and creating a new schema that includes the non-key attributes that functionally determine other non-key attributes. From the above schema, the addresses, business addresses, and sellers schema all don't fit the definition of 3rd normal form so we will remedy this. In both the addresses and business addresses schema, since zip code determines city and state we need to create a new schema where zipcode is a key and city and states are attributes, removing these from their original schema. The new schema can be seen in **Figure 18** and **Figure 19** below. Additionally, with the seller schema, we can take similar steps. We see that business_name determines the business_adress_id and the

customerservice_phonenumber. We can take these and move them to a new table where business_name is the key and the business_adress_id and the customerservice_phonenumber are attributes. This can be seen in **Figure 20**. At this point, all given schemes from above should now be in 3rd normal form.

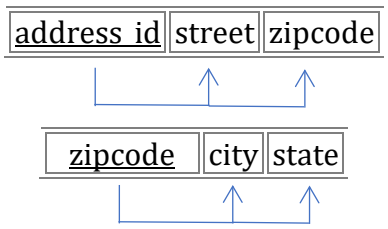

Figure 18 – 3rd Normal Form Addresses Schema

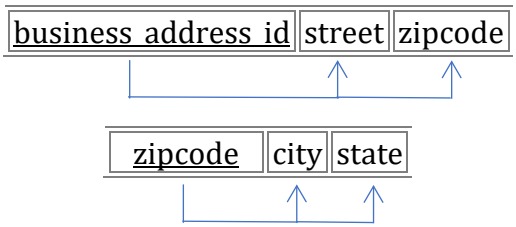

Figure 19 – 3rd Normal Form Business Addresses Schema

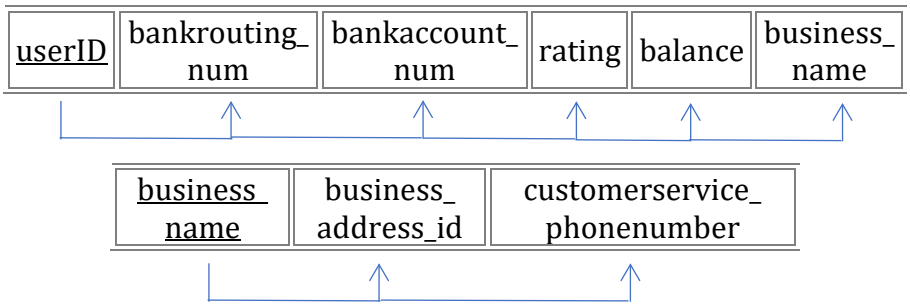

Figure 20 – 3rd Normal Form Sellers Schema

6. Conclusion

This phase 1 project report has provided a comprehensive overview of the startup project LionAuction. The report covers four essential components necessary for creating LionAuction: requirement analysis, conceptual database designs, technology survey, and logical database designs and normalization. The requirement analysis explores the data storage and expected functionality of the application. The conceptual database design lays out the application's data and constraints into entity-relationship models. The technology survey explores potential web and database application technologies that can be used for LionAuction. Finally, the logical database designs and normalization refine the conceptual database designs into a schema suitable for the database in LionAuction. Overall, this report provides a solid foundation for the development of LionAuction going into phase 2 and building the prototype, which has the potential to be a game-changer in the e-commerce industry.

Appendix A. Project Plan

- **March 3, 2023 –**
    - Finish Project Phase 1: Database Design Report
- **March 6, 2023 –**
    - Start Project Phase 2
- **March 20, 2023 –**
    - Project Phase 2 Progress Review
- **April 21, 2023 –**
    - Finish Project Phase 2
- **April 22, 2023 – April 23, 2023 –**
    - Make Project Video
- **April 24, 2023 – April 25, 2023 –**
    - Project Final Demonstration and Review
- **April 26, 2023 –**
    - Final Project Video Presentation