**Phase 1 – Database Design & Technology Survey**

**Author:** Tejas C. Marimuthu

### ­­

### Table of Contents

1. Introduction

2. Requirement Analysis

2.1. Bidders

2.2. Sellers 4

2.3. HelpDesk Staff 6

2.4. Guests 6

2.4. Products 7

2.5. Features 7

2.6. Aesthetics 8

3. Conceptual Database Design 9

3.1. Bidders 9

3.2. Sellers 11

3.3. HelpDesk Staff 12

3.4. Products and Categories 13

3.5. Complete Overview of ER Diagram 15

4. Technology Survey 16

5. Logical Database Design and Normalization 21

5.1. Relational Schema of ER Diagram 21

5.2. Normalization 22

5. Project Plan 26

6. Conclusion 27

**List of Figures**

Figure 1 – Partial ER Diagram of Bidders 10

Figure 2 – Partial ER Diagram of Sellers 11

Figure 3 – Partial ER Diagram of HelpDesk Staff 13

Figure 4 – Partial ER Diagram of Products and Categories 14

Figure 5 – Overview of Entire ER Diagram 16

# 1. Introduction

LionAuction is a startup project that aims to revolutionize the way members of Lion State University (LSU) buy and sell used or unused goods via online auctions. This e-commerce platform is the brainchild of Hilbert Dude, who identified an untapped market for such a service in university campuses worldwide. Through a preliminary market investigation, Mr. Dude realized the potential value of his platform not only to the members of the university but also to the local economy in the campus and surrounding community. As a student of CMPSC 431W, I was tasked to assist Mr. Dude and Professor Lee to design a database-backed web application to buy/sell products on campus. This is the Phase 1 report that goes into details of how my LionAuction prototype will be designed, with a focus on requirement analysis, conceptual database design, technology survey, logical database design, normalization. The end goal of this report is to lay the foundation for Phase 2, which will be the building of the LionAuction prototype based on this Report. The objective of this report is to provide understanding of how the database management system will be implemented, what entities and relationships are present, the major features in the LionAuction prototype, and what technologies can be used to build the prototype.

# 2. Requirement Analysis

As part of Phase 1 for the LionAuction Prototype, a thorough analysis of the system requirements must be conducted. LionAuction aims to provide an online e-commerce platform for LSU university members and local businesses to buy and sell goods. In order to achieve this goal, researching various online e-commerce websites such as eBay, Amazon, and Facebook marketplace are important to gain a better understanding of the expected functions and requirements for LionAuction.

The purpose of this task is to envisage what the system should look like and how it should operate to enable campus auction business. In order to do so, I need to analyze and describe in detail the system requirements, including the system functions and operations, data needed to support those system functions and operations, the integrity constraints of LionAuction to be imposed on the data, and how and where they will be stored and accessed to support the system functionality. In the following, I describe the different components of the LionAuction prototype.

## 2.1. Bidders

As the client requests, users who attempt to operate on the LionAuction prototype must be registered in the system. That is, they must have an account to use the website. A user will be identified by a user ID, which will be their email, and be authenticated with a password. A suggestion here would be to allow the site to be viewed by unregistered users as well, similarly to Amazon and eBay. This would not only increase the people visiting the website, but it will also make it easy to search for products without having to login. When the user decides to either buy/manage a product/listing, they must log into the website with their credentials as those features are restricted to registered users. There are three types of users, and a possible fourth user as well. They are Bidders, Sellers, HelpDesk Staff, and Guests.

In the prototype implementation of LionAuction, Bidders will represent the students of LSU. To participate in the bidding process, a student needs to log in using their user ID (email address) and password. The system will maintain detailed information about the bidders, including their name, email address, password, address, phone number, and credit card information, which is used for payment. Address can be broken down into street, city, state, and zip code. Credit card information can be broken down into type of credit card, credit card number, and expiration date. Additionally, the system will collect data on the student's major, age, gender, and annual income, which will be used for marketing analysis purposes.

Once the Bidder is authenticated from the log in page, they will have access to update their personal information (except user ID), reset password, and update payment information. An assumption to be made from the client’s requirement list is the information restricted to only be accessible by the Bidder is the payment information. Only Bidder can manage their own payment information. A bidder can also browse the products listed by sellers through a category hierarchy, select a product of interest, and start bidding on it. The system will display the seller's information and product details to the bidder. However, products that have been sold or terminated by the Seller should not be viewable to Bidders. While viewing a product, a Bidder can send questions to the Seller who is selling that product. Questions can either be conducted through direct contact to the seller using email or through a public posting where seller will respond publicly in the posting section.

If a bidder bids and wins an auction, they are required to pay the winning price immediately. After the transaction is completed, the listing will be archived, and the bidder can rate the seller based on a five-star rating system. This rating system will be beneficial to the public as they will know which Sellers are the most reliable. If a bidder wants to sell their own items on LionAuction, they need to fill out an application form and submit it to the HelpDesk for approval.

## 2.2. Sellers

Sellers are either students at LSU or approved local business vendors. Like Bidders, Sellers also must be authenticated in order to have access to the LionAuction website. This is done through user ID (email address) and password. Once logged in, they can update their account information exactly like how Bidders do. However, unlike Bidders, the data that is collected from Sellers is different type then what is collected on Bidders. Sellers will have bank routing number, bank account number, and balance. For a Seller that is an approved local business vendor, they will additionally have business name, business address, and customer service phone number stored in the database as well. Once a Seller’s account is established, they then need to enlist products that will be put up for auction on the website. Here they will be directed to a page on LionAuction where they can input the information of their product, such as product information, category of product, reserve price, and auction stop time. If the category does not exist for that specific product, a Seller may send a request with the name of the category of the product to the HelpDesk Staff. Once approved, HelpDesk Staff will create that category under the category hierarchy.

A Seller can view their listed product on its own distinct page, where all the product information will be present, similarly when viewing a specific product on Amazon. The forum with all the questions listed by Bidders will also be present on this page for a Seller to view and respond to. This page will also hold Seller ratings that are viewable by both the Seller and Bidder. The rating information will be crucial in determining the public’s trust of the Seller.

Seller is not allowed to bid as they are business accounts for the purpose of auctioning items. After an auction concludes, if no bid is higher than the reserve price, then the Seller is not required to sell the product. If the auction is either successful or not, it will be unlisted as an item not for sale. This means that the product will not be visible to the Bidder as the auction has concluded, however the information about the auction, such as bids placed, will be recorded in the database. An auction is successful when Bidder completes the payment and seller receives that payment in their account.

One rule to mention is that two sellers may not sell the exact same item. If two students that live together share a tv and decide to auction it, they both cannot list the same tv under their seller account. For the prototype implementation, I will assume based on the requirements given by client that a product list contains only one product item for ease of implementation. Any product sold on LionAuction shall be recorded in the LionAuction database along with bidding history for market research purposes.

## 2.3. HelpDesk Staff

The HelpDesk Staff will act as the administrators of LionAuction website. They will be the IT staff that will be in charge of managing all the functionality of the website. HelpDesk Staff have the power to change the Bidders and Sellers user IDs. They also have the function to upgrade a Bidder to Seller role and to add additional categories requested by Seller. They will also have the functionality to produce market analysis using the data collected from the LionAuction website.

The client never mentioned anything about how the HelpDesk Staff shall manage the website. I would implement a separate login page on LionAuction specifically for HelpDesk Staff Only. HelpDesk staff will need to provide a user ID (staff identification number) and password. Once logged in, they will have the ability to manage the LionAuction website and tend to Bidder and Seller requests.

## 2.4. Guests

This is a suggestion feature for LionAuction but allowing anyone to visit the website for the purposes of browsing/viewing items can greatly increase the online traffic to the website. Guests are users who only have the ability to browse/view. They cannot bid nor sell items on LionAuction without having an account. For a guest user to extend its abilities, it can either log into an existing account or create a new account.

## 2.5. Products

Each product will have a title, product detail, category, reserve price, and product ID (Unique value to distinguish from other similar products). Every product is required to have exactly one category and one seller associated to it. After the conclusion of an auction, product is removed from Bidder’s view and continues to be stored in database. For a product that is sold by a local business, the data associated with that product is to be deleted completely from the database once the vendor leaves the market. Any other product is to remain in the database even after auction has concluded.

## 2.6. Features

***Bidding-*** An auction begins when a Seller posts a product and ends at the time specified by the Seller. A Bidder may place one or more bids on multiple products, as long as they are not the Seller. A minimum bid of $1 higher than the previous bid is required for a Bidder to bid. Once the auction is completed, the Seller notifies all Bidders except the winner of the auction that they have lost the auction. The Seller then notifies the winner that they have won the auction and must complete the payment transaction. Once payment is received, auction concludes, and product is removed off the site and maintained in historical database.

***Category Hierarchy-*** Categories are a way to group similar products together. Users are able to click on the category of products that they are interested in, and from there either click a subcategory or search for the product of interest in the current level. By using this category system, it will make searching for products much quicker and precise.

***Search Bar-*** On the home page of LionAuction, users have multiple ways of searching for products. They can either directly search for the product in the search bar, resulting in a list of similar products, or they can navigate the multiple categories in the header.

## 2.7. Aesthetics

When thinking about building the LionAuction prototype, examples of real-world businesses come to mind. Amazon, eBay, Facebook Marketplace, and Craigslist are four popular websites used to sell products. I want to ensure that the prototype maintains a modern, minimal, and aesthetic look that appeals to the user’s eye, as this will be used by college students. I also want to ensure that the website is also built to be accessibility friendly, expanding the range of users visiting the website.

# 3. Conceptual Database Design

To develop a conceptual database design for the LionAuction prototype, the key entities and their relationships must be identified.

## 3.1. Bidders

This entity represents all LSU students who are registered as bidders on LionAuction. It has attributes such as bid (primary key), name, email, phone\_number, password, major, age, gender, and annual\_income.

Another entity called Address stores the Bidder’s address, where it is broken down into street, city, state, zip\_code, and apt\_num (apartment number). There are two ways to implement this in the database system. I can either add all the attributes into the Bidder entity, or I can create a separate entity specifically for address and add a relationship between Bidder and Address. First reason is that there are multiple components of the address requiring to be a separate entity. The second reason is that some people might reside in the same address as another, so I needed a way to distinguish that by adding integrity constraints. Every address must be resided by at least one Bidder and every bidder resides in exactly one address. The address entity is also a weak entity, meaning that if Bidder is deleted, then so is the address associated with it. I decided to proceed with the route of adding a table since it will eliminate any two addresses that are the same, reducing the data stored.

I also implemented a separate entity to store the credit card information related to Bidder called Payment. This entity has similar constraints as the address entity. It stores the credit card information related to card number (partial key), expiration date, and card type. Like the Address entity, it is also a weak entity.

***Entity-Relationship Diagram of Bidders-*** See Figure 1 below for a partial view representing the Bidder ER Diagram. In this diagram, the rectangles represent the entities, the diamonds represent the relationships, and the ovals represent the attributes. The lines between the entities and the relationships represent the foreign keys.

Diagram

Description automatically generated**Figure 1 – Partial ER Diagram of Bidders**

## 3.2. Sellers

This entity represents the users who list their products for auction on LionAuction. It has attributes such as sid (primary key), name, email, password, rep\_score (reputation score). It also has the attributes associated to their bank information such as routing\_number, account number, and balance. A subclass of Sellers is Local\_Sellers entity. It has access to the same attributes that a Seller would because of the ISA hierarchy, and it also has business\_name, business\_address, and service\_number (customer service number).

***Entity-Relationship Diagram of Sellers-*** See Figure 2 below for a partial view representing the Seller ER Diagram. In this diagram, the rectangles represent the entities, the diamonds represent the relationships, and the ovals represent the attributes. The lines between the entities and the relationships represent the foreign keys.

Diagram

Description automatically generated**Figure 2 – Partial ER Diagram of Sellers**

## 3.3. HelpDesk Staff

This entity represents the HelpDesk Staff, who have attributes such as staff\_id (staff identification number), name, and password. In the front end, the HelpDesk Staff will be able to act as administrators for the website, ensuring it is run smoothly. The main relationship that the HelpDesk Staff has is that it resolves requests from the Bidders or Sellers. To implement this into the database, I created another entity called Requests that utilizes an ISA hierarchy to distinguish Bidder requests from Seller requests. A request has three attributes: rid (Primary key and request ID), details, and status. Seller\_Request has sid and Bidder\_Request has bid. This way it is known which bidder and which seller made a certain request. I did not place any constraints here as there were no requirements that suggested so. The two Send relationships branching down on both sides of the Request entity lead to the Bidder entity and the Seller entity as Bidder and Seller can send a request. Every request must have exactly one bidder or seller. This relationship might have to be revised on the diagram but for the purposes of the prototype, I will leave it be.

***Entity-Relationship Diagram of HelpDesk Staff-*** See Figure 3 below for a partial view representing the HelpDesk Staff ER Diagram. In this diagram, the rectangles represent the entities, the diamonds represent the relationships, and the ovals represent the attributes. The rectangles with a black bolded rectangular outline suggest weak entities. The lines between the entities and the relationships represent the foreign keys.

**Figure 3 – Partial ER Diagram of HelpDesk Staff**

Diagram

Description automatically generated

## 3.4. Products and Categories

This entity represents the products listed by the sellers on LionAuction. It has attributes such as pid (primary key), title, details, cid (category id), reserve\_price (start price). I added a Category entity to hold all the categories in it. It contains cid (Primary key), name, and description. Because every product is required to have at least one category, I added a total participation constraint on Products to Belongs to relationship.

***Entity-Relationship Diagram of Products-*** See Figure 4 below for a partial view representing the Product ER Diagram. In this diagram, the rectangles represent the entities, the diamonds represent the relationships, and the ovals represent the attributes. The rectangles with a black bolded rectangular outline suggest weak entities. The lines between the entities and the relationships represent the foreign keys.

**Figure 4 – Partial ER Diagram of Products and Categories**

**Diagram

Description automatically generated**

## 3.5. Complete Overview of ER Diagram

In this section, here is a complete overview of the entire ER Diagram showcasing all the entities and relationships present in our database.

***Overview of Entire Entity-Relationship Diagram-*** See Figure 5 below for a complete view of the proposed ER diagram for the LionAuction Prototype. In this diagram, the rectangles represent the entities, the diamonds represent the relationships, and the ovals represent the attributes. The lines between the entities and the relationships represent the foreign keys.

**Diagram

Description automatically generated with medium confidenceFigure 5 – Overview of Entire ER Diagram**

# 4. Technology Survey

There are several web development frameworks, programming languages, and database management systems available today. Selecting the optimal technology stack for building the LionAuction prototype can be difficult. Therefore, in this technology survey, here are three potential technology stacks that may be appropriate for the prototype.

1. **Flask, Python, PyCharm IDE, SQLite:** Flask is a lightweight web application framework written in Python that is widely used in small to medium-sized applications. Flask is a flexible framework that can manage any complexity of the application. Python is a high-level programming language known for its simplicity, readability, and vast libraries. PyCharm IDE is a widely used Python IDE that offers many features such as code completion, debugging, and refactoring. SQLite is a relational database management system (RDBMS) that is widely used for small to medium-sized applications. It's easy to set up and doesn't require a separate server.

**List of Pros:**

* Flask is easy to set up and use, making it a good choice for small to medium-sized applications.
* Python is a widely used language with vast libraries and is easy to learn.
* PyCharm IDE provides features such as code completion, debugging, and refactoring, which helps developers write code faster and more efficiently.
* SQLite is easy to set up, making it ideal for small to medium-sized applications.

**List of Cons:**

* Flask is not suitable for large-scale applications.
* SQLite has limited scalability compared to other database management systems.

1. **React, Node.js, VS Code, MongoDB:** React is a popular JavaScript library for building user interfaces. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to build scalable and high-performance applications. Visual Studio Code (VS Code) is a widely used code editor that supports many programming languages and has a variety of features such as code highlighting, debugging, and refactoring. MongoDB is a NoSQL database management system that uses a document-based data model.

**List of Pros:**

* React is an excellent choice for building dynamic user interfaces.
* Node.js allows developers to use JavaScript on both the front-end and back-end, which makes development faster and more efficient.
* VS Code is a lightweight code editor with many features that can be customized according to developer's needs.
* MongoDB has a flexible document-based data model that makes it easy to store and retrieve data.

**List of Cons:**

* React can be complex to set up and learn.
* Node.js has a steep learning curve and may not be suitable for beginners.
* MongoDB may not be suitable for applications that require complex relationships between data.

1. **Ruby on Rails, Ruby, Atom, PostgreSQL:** Ruby on Rails is a popular web application framework written in Ruby. Ruby is a high-level, dynamic, and object-oriented programming language known for its simplicity and readability. Atom is a free and open-source code editor that provides a customizable user interface and many features such as code highlighting and auto-completion. PostgreSQL is a powerful and open-source relational database management system.

**List of Pros:**

* Ruby on Rails is a full-stack web application framework that is easy to learn and use.
* Ruby is a high-level, easy-to-learn programming language that allows developers to write clean and concise code.
* Atom is a lightweight and customizable code editor that can be tailored to developers' needs.
* PostgreSQL is a powerful and open-source database management system that can handle complex data relationships.

**List of Cons:**

* Ruby on Rails might not be suitable for large-scale applications and I am unfamiliar with it.
* Ruby might not be as fast as other programming languages and I will have to take time to learn this language, increasing build time of application.
* Atom may not have as much features as other code editors and I am not familiar with using it.
* PostgreSQL might not be as easy to set up when compared to other database management systems.

I will be building a prototype for LionAuction, which means that this will be a small-scale application that will be mainly used to test the idea of LionAuction. Because of that, I have decided to select the first stack. This is because the prototype will be a small-scale application, SQLite is easy to set up, and Python is a language that I am well versed in. The only change I will make to using this stack is replacing PyCharm IDE with VS Code. As a developer, I want the coding process to be very smooth by utilizing much customization and extensions. VS Code will allow me to do so, and it is a popular IDE for development in current times. The second stack is appealing as well since I do have experience with utilizing those software/tools, however because this is a simple prototype to demonstrate the possible functionality of LionAuction, I will use stack one as it will allow me to build it faster for the client.

# 5. Logical Database Design and Normalization

Here we have developed relational schemas based on the first two sections, requirement analysis and the conceptual database. After translating the ER Diagram to relations, we will then Normalize it, essentially organizing the data, reduce redundancy, and improve data integrity. The goal of this section is to produce a refined schema that is in 3rd Normal Form. Here we will assume two functional dependencies:

* 1. Zip Codes in the address can determine the state and cities.
  2. Student email can determine the password associated with that Student.

## 5.1. Relational Schema of ER Diagram

This is the translated ER Diagram into Relations:

1. Bidder(bid: integer, name: string, email: string, password: string, phone\_number: integer, major: string, age: integer, gender: string, annual\_income: real)
2. Address(aid: integer, street: string, city: string, state: string, zip\_code: integer, apt\_num: integer)
3. Payment(credit\_card\_type: string, card\_number: integer, expiration\_date: string)
4. Seller(sid: integer, name: string, email: string, password: string, routing\_number: integer, account\_number: integer, balance: real, rep\_score: real)
5. Local\_Seller(sid: integer, name: string, email: string, password: string, routing\_number: integer, account\_number: integer, balance: real, rep\_score: real, business\_name: string, business\_address: string, service\_number: integer)
6. HelpDesk\_Staff(staff\_id: integer, name: string, password: string)
7. Products(pid: integer, title: string, details: string, reserve\_price: real, cid: integer)
8. Categories(cid: integer, name: string, description: string)
9. Bid(bidID: integer, status: string, time: string, amount: real)
10. Seller\_Request(sid: integer, rid: integer, details: string, status: string)
11. Bidder\_Request(bid: integer, rid: integer, details: string, status: string)
12. Questions(qid: integer, question: string, response: string, pid: integer, bid: integer)

## 5.2. Normalization

The goal of this section is to convert our relational schema found in section 5.1 to 3rd Normal form. 3rd normal form is achieved if it is in 2nd normal form and no non-key attributes are dependent on the primary key. Essentially, I went through each relational schema and viewed the dependencies/redundancies in each one. If one attribute was dependent on another and it was not the primary key, then I created a new table where one attribute acted as a key and is in both schemas, and the other attribute/attributes resided in the new table.

1. Bidder(bid: integer, name: string, email: string, phone\_number: integer, major: string, age: integer, gender: string, annual\_income: real, aid: integer, card\_number: integer)
2. Login(email: string, password: string)
3. Address(aid: integer, street: string, zip\_code: integer, apt\_num: integer)
4. State(zip\_code: integer, state: string, city: string)
5. Payment(credit\_card\_type: string, card\_number: integer, expiration\_date: string)
6. Seller(sid: integer, name: string, email: string, password: string, account\_number: integer, rep\_score: real)
7. Bank(account\_number: integer, routing\_number: integer, balance: real)
8. Local\_Seller(sid: integer, name: string, email: string, password: string, account\_number: integer, rep\_score: real, business\_name: string, business\_address: string, service\_number: integer)
9. HelpDesk\_Staff(staff\_id: integer, name: string, password: string)
10. Products(pid: integer, title: string, details: string, reserve\_price: real, cid: integer)
11. Categories(cid: integer, name: string, description: string)
12. Bid(bidID: integer, status: string, time: string, amount: real)
13. Seller\_Request(sid: integer, rid: integer, details: string, status: string)
14. Bidder\_Request(bid: integer, rid: integer, details: string, status: string)
15. Questions(qid: integer, question: string, response: string, pid: integer, bid: integer)

# 6. Project Plan

The timeline for developing this prototype can be broken down in 3 weeks.

The first week is the initial week where setting up the development environment is important. The next task once setup is to develop the SQLite scripts to create the database schema and tables. Once this is completed, then enforcing the integrity constraints and testing the database are important. The last task of this week is to create a basic user interface for the LionAuction prototype.

The second week will be about developing the user registration, login functionality, and authentication. Once this is completed, adding features to the website is the next step, such as bidding, auctioning, product listing, and more. After this is completed, payment is the last step. The goal is to establish all the main requirements for the website.

The third and final week is about designing the website to look more aesthetic and creating different user views for seller, bidder, helpdesk staff, and guest. It will also focus on implementing the questions asked by bidders and requests sent to helpdesk staff.

By the end of the three weeks, we should have a fully functional LionAuction prototype that allows LSU students to buy and sell items through online auctions.

# 7. Conclusion

In conclusion, the LionAuction prototype is a comprehensive system that is designed to meet the needs of LSU students who want to buy and sell items through an online auction platform. In this project, I have conducted a thorough analysis of the system requirements, developed a conceptual database design, mapped it to a logical database design, and normalized the schema to ensure data redundancy is reduced to an acceptable level while maintaining data integrity.

The LionAuction database schema includes several tables such as bidders, sellers, products, categories, bids, payments, and more. Each table has a specific purpose and is related to other tables through primary and foreign keys. Normalization was applied to the schema to ensure that the data is stored in an efficient and logical manner and to reduce data redundancy.

Overall, the LionAuction prototype aims to provide a user-friendly platform for LSU students to buy and sell items through online auctions while maintaining data integrity and reducing data redundancy.

# Citations

Chapter 1,2,3,4,19 Slides

My own knowledge Notes