People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Abderrahmane Mira University of Béjaia
Faculty of Exact Sciences
Department of computer science

---


**This is Presented**

**For the Master's Degree**

**In Computer Science**

**Option: Advanced Information Systems**

**By: Mehdi Arslane BENBOUSSAD**

# Implementation of Combinatorial Auctions in a Dynamic Web Application

**Defended at Abderrahmane Mira University of Béjaia,**
**On 14/09/2023, before the jury composed of:**

| | | | |
|---|---|---|---|
| F. Mir | M.C. classe/ B | President | at l'UAMB - Bejaia. |
| L. Asli | M.C. classe A | Supervisor | at l'UAMB - Bejaia |
| M. Soufit | M.C. classe A | Examiner | at l'UAMB - Bejaia. |
| A. Zaidi | M.C. classe/ A | Examiner | at Centre of Research Tamazight - Bejaia. |

**Academic Year** 2022 – 2023

# Contents

# List of Figures

# Notations

**UML**: Unified Modeling Language

**PWA**: Progressive Web Application

**DBMS**: Database Management Systems

**NoSQL**: Not only SQL

**ER**: Entity-Relationship

**UI**: User Interface

**DBMS**: Database Management System

**UML**: Unified Modeling Language

**PWA**: Progressive Web Application

**SSL**: Secure Sockets Layer

**JIS**: Java Instrumentation Suite

**AFG**: Action Flow Graph

**PDG**: Problem Definition and Goal

**CA-GREEDY**: Combinatorial Auction - Greedy

**CA-LP**: Combinatorial Auction - Linear Programming

**CA-PROVISION**: Combinatorial Auction - Provision

**VMAP**: Virtual Machine Allocation Problem

**DVMPA**: Dynamic Virtual Machine Provisioning and Allocation Problem

**WDP**: Winner Determination Problem

**ER**: Entity-Relationship (used in the context of an ER diagram)

**LGC**: List of Winning Bidders in Conflict

**TWT**: Temporary Winner List

**EC**: Minimum Subset of Bidders

**TWL**: Temporary Winner List (again, mentioned in a different context)

**CWL**: Conflicting Winner List

**API**: Application Programming Interface

**SQL**: Structured Query Language

**IDE**: Integrated Development Environment

**JSON**: JavaScript Object Notation

**RESTful**: Representational State Transfer

**URL**: Uniform Resource Locator

# Thank You

I'd like to express my honest gratitude to the juries for their presence and their ability to review of my work at such a time. I'm also thankfull for my supervisor for his guidance and support throughout this journey.

To my dear friends, your presence and advice have been a great help. Thank you for always being there for me.

I'm deeply appreciative of my parents and family for their great emotional and financial support.
Lastly, I'd like to extend my apologies if I forgot anyone. Your contributions have not gone unnoticed. Thank you all for your support and encouragement.

# General Introduction

Negotiating over prices is a common human behavior that occurs in various contexts and cultures worldwide. It is a form of communication and interaction between buyers and sellers, where both parties aim to reach a mutually agreeable price for a product or service.

Auctions are a form of negotiation over goods and services through giving offers in form of an estimated price in exchange of the presented good or service, the winner shall be the buyer who gives the highest bid.

Traditional auctions have long been a popular way of negotioating over goods. However, they come with many limitations that can be overcome by online combinatorial auctions.

In the world of business, there are numerous ways to fine-tune the settings of an auction in order to achieve optimal performance and outcomes. One such mechanism involves utilizing various types of auctions.

Such as the English auction where the bidding price increases until there are no more bids, the Dutch auction where the bidding price of an item starts high and then gradually decreases until a buyer takes the offer, and the Vickrey auction in which the highest bidder wins but pays the amount of the second-highest bid[4].

Another type of auction is the online auction, where bidders can participate in the auction through an online auction platform[14].

Finally, there are combinatorial auctions where bidders can submit bids for a set of items and select a subset of items they wish to purchase along with the requested amount of each item. The auctioneer then determines the optimal combination of bids that will maximize revenue[4, 12].

One major issue of traditional auctions is their limitation to selling only one item or a set of items at a time. This can be problematic for buyers who are interested in bedding on multiple items, as it can be time-consuming to place separate bids for each individual item. As a result, many buyers may prefer to purchase these items from a marketplace where they can buy multiple items at once, rather than going through the bidding process for each individual item.

Another issue with traditional auctions, particularly those with multi-unit bids, is that buyers may end up with unwanted items if they purchase a full set at once. In

contrast, combinatorial auctions are designed to be mutually beneficial for both buyers and sellers, offering buyers greater flexibility in choosing which items they want to bid on and ultimately purchase. This increased freedom for buyers can lead to more profitable outcomes for both buyers and sellers.

On the other hand, online combinatorial auctions offer several advantages over traditional auctions. Buyers can bid on multiple items simultaneously, making the process faster and more efficient. This convenience is particularly notable since buyers can participate from the comfort of their own homes with just a click of a button.

Moreover, online combinatorial auctions can handle a larger scale than traditional auctions. With the right information system in place, online auctions such as eBay [14] can manage a larger number of bidders and items. This scalability is a significant advantage for sellers as it allows them to reach a broader audience and potentially increase their revenue.

This is why in this study, we are primarily focused on the last two types of auctions, namely online auctions and combinatorial auctions. We aim to integrate these auction formats by developing a dynamic website that will enable us to conduct combinatorial auctions.

In Chapter 1, we'll explore essential database concepts, distributed networks, and the selection of the right database type and schema. We'll use tools like Unified Modeling Language (UML) diagrams for clear system representation. Our dynamic platform combines React and Progressive Web Application (PWA) principles, impacting software performance. This chapter provides a foundation for understanding challenges in building a combinatorial auction web app, including scalability, real-time updates, UI design, security, and privacy.

Chapter 2 delves into auction mechanisms and mathematical models for our dynamic web app. Combinatorial auctions, allowing bids on sets of items, offer efficient resource allocation. We explore various auction types, including English, Dutch, and Vickrey auctions, and examine mathematical models for winner determination, forming the basis of our auction algorithms.

Chapter 3 establishes our Combinatorial Auction System. We create a robust database model using Entity-Relationship diagrams and introduce the dynamic Winner Determination Problem (WDP) model, emulating English combinatorial auctions. We focus on efficient bid handling, seamless updates, conflict resolution, and dynamic programming, shaping a responsive and future-ready auction platform.

In the final chapter, we present results, analysis, and recommendations for our web app. We explore core functionalities, database design, winner determination algorithms, user experience, performance, scalability, and security. This examination offers insights into our project's state and suggests enhancements for continued effectiveness and security in facilitating online auctions.

# 1

# Modeling and Computer Science Fundamentals

## Introduction

In the upcoming chapter, we will delve into essential database concepts and distributed networks, highlighting the importance of selecting the right database type and schema. We will explore modeling tools such as the Unified Modeling Language (UML) diagrams for clear system representation. Our dynamic implementation platform will combine React for user-friendly web apps and PWA (Progressive Web Application) principles for enhanced features. We will discuss the evolution of Database Management Systems (DBMS), UML's role in data representation, and the impact of dynamic platforms on software performance.

This chapter will provide a solid foundation for understanding the future challenges in building a combinatorial auction web app, including scalability, real-time updates, User Interface (UI) design, security, and privacy.

## 1.1 Databases

Databases are structured collections of items of information that are organized and stored for manipulation. They provide a way to store, manage, and retrieve large amounts of information. In the context of a combinatorial auction webapp, a database would be essential for storing various data related to the auctions, such as user information, item details, bids, and transaction records. here is an example of a data base network:

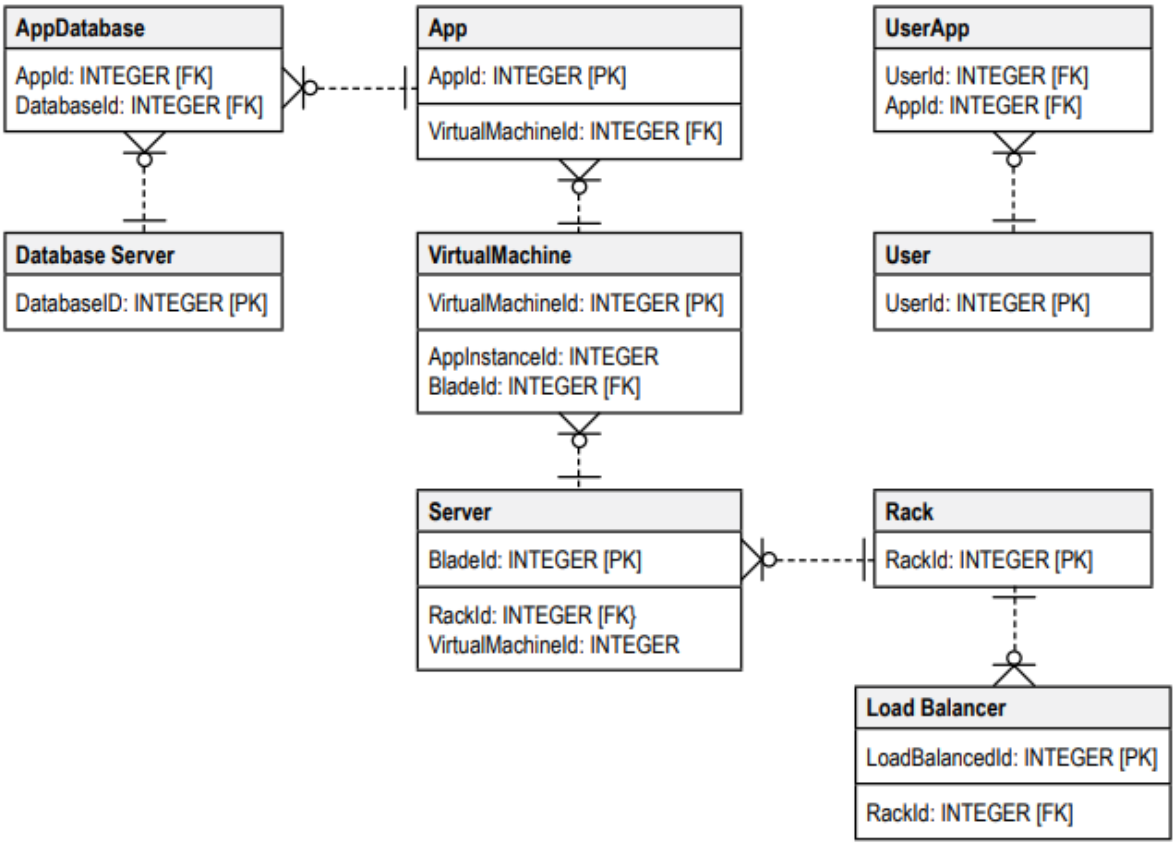



Figure 1.1: Example of a relational database network [20]

The importance of databases lies in their role in enhancing applications, offering enhanced scalability, availability, and performance. Distributed databases involve interconnected servers storing subsets of data, promoting load distribution and parallel processing. Different database types, such as relational. When designing an app, selecting the appropriate database type based on factors like data structure, scalability, and integration ease is crucial. Effective database schema design is also highlighted as essential for optimizing data storage and retrieval, ultimately playing a pivotal role in building a robust and efficient system.[3]

## 1.2 DB and Network DB

A file composed of records, each containing fields together with a set of operations for searching, sorting, recombining, and other functions. And in information management, a network database is a type of database in which data records can be related to one another in more than one way. A network database is similar to a hierarchical database in the sense that it contains a progression from one record to another. It differs in being less rigidly structured [15].

### 1.2.1 Relational Database

**Definition 1.** *A Relational Database is a database or database management system that stores information in tables—rows and columns of data and conducts searches by using*

*data in specified columns of one table to find additional data in another table.*

*In a relational database, the rows of a table represent records (collections of information about separate items) and the columns represent fields (particular attributes of a record).*

*In conducting searches, a relational database matches information from a field in one table with information in a corresponding field of another table to produce a third table that combines requested data from both tables.[15]*

### 1.2.2 Hierarchical Database

**Definition 2.** *A hierarchical database is A database in which records are grouped in such a way that their relationships form a branching, tree like structure. This type of database structure, most commonly used with databases for large computers, is well suited for organizing information that breaks down logically into successively greater levels of detail. The organization of records in a hierarchical database should reflect the most common or the most timecritical types of access expected [15].*

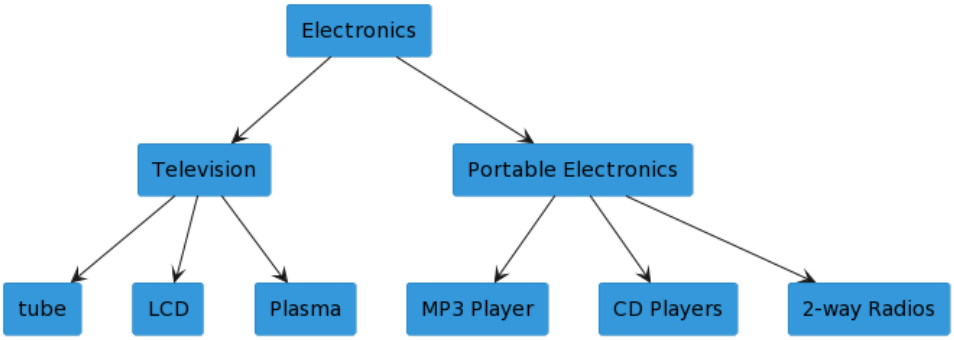



Figure 1.2: Example of a example of a Hierarchical Structure of a database

### 1.2.3 Data Warehouse

**Definition 3.** *A database, frequently very large, that can access all of a company's information. While the warehouse can be distributed over several computers and may contain several databases and information from numerous sources in a variety of formats, it should be accessible through a server. Thus, access to the warehouse is transparent to the user, who can use simple commands to retrieve and analyze all the information. The data warehouse also contains data about how the warehouse is organized, where the information can be found, and any connections between data. Frequently used for decision support within an organization, the data warehouse also allows the organization to organize its data, coordinate updates, and see relationships between information gathered from different parts of the organization.[15]*

## 1.3 Modeling and Implementation Tools

Modeling and implementation tools play a vital role in visualizing and designing various aspects of the system. This section explores the utilization of UML (Unified Modeling

Language) modeling techniques and diagrams to aid in the development process.

### 1.3.1 UML Diagrams

UML provides a standardized language for modeling software systems, offering a visual representation of different aspects and interactions within the system. Two key UML diagrams that will be utilized in this project are the Use Case diagram and the Entity-Relationship (ER) diagram.

#### Use Case Diagram

The Use Case diagram provides a high-level view of the system's functionality by illustrating the interactions between actors (users or external systems) and the system itself. It showcases the various use cases, their relationships, and the flow of events, helping to identify the system's behavior from a user's perspective.

**Actor:** An actor represents a user, an external system, or any entity that interacts with the system being modeled. Actors are depicted as stick figures in a use case diagram. They initiate use cases and receive the system's responses.

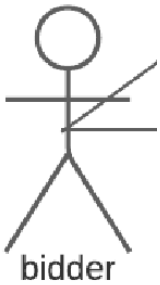



Figure 1.3: Example of an actor

**Use Case:** A use case represents a specific functionality or behavior of the system. It describes a set of actions or interactions between the system and actors to achieve a goal. Use cases define the system's functionality from a user's perspective and help capture the system's requirements.

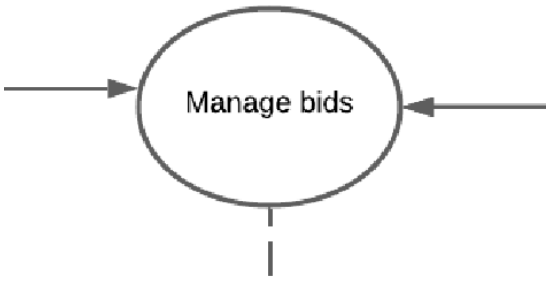



Figure 1.4: Example of a Use Cases

**Relationships:** Relationships in a use case diagram illustrate the connections between actors and use cases. The primary relationship in a use case diagram is the association relationship, which represents that an actor is associated with one or more use cases. Other relationships include generalization (inheritance), extends, and includes relationships.

- **Generalization (Inheritance) Relationship:** This relationship represents an "is-a" relationship between two use cases. It signifies that one use case inherits or specializes the behavior and attributes of another use case. The child use case adds more specific functionality to the parent use case.

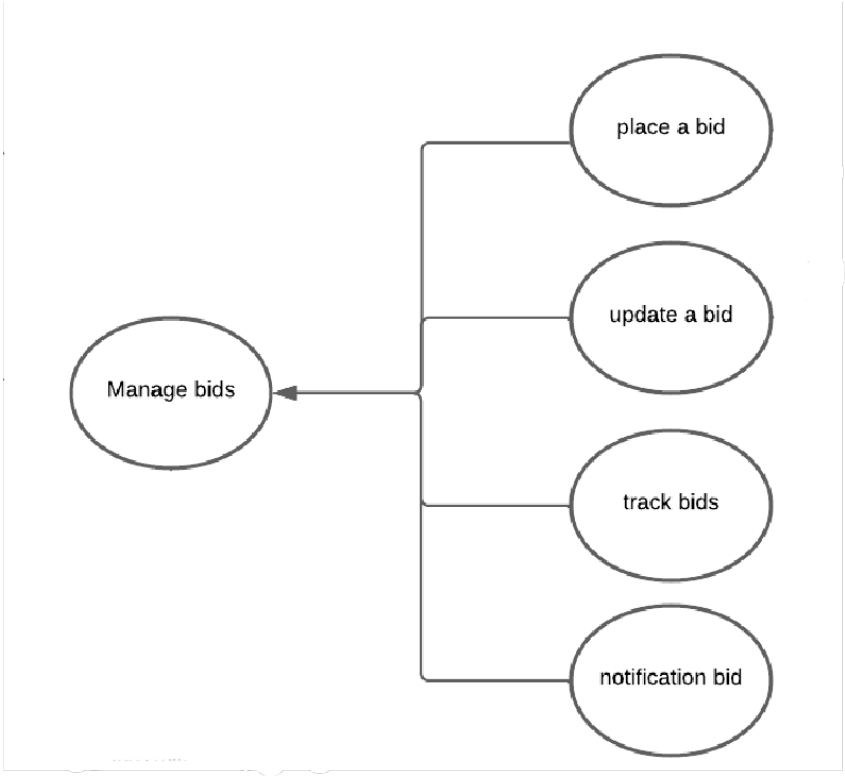



Figure 1.5: Example of an inheritance relationship

- **Extends Relationship:** The extends relationship shows optional or alternative behavior that can be added to a base use case. It signifies that the extended use case (extension) can be invoked under certain conditions to enhance the functionality of the base use case.

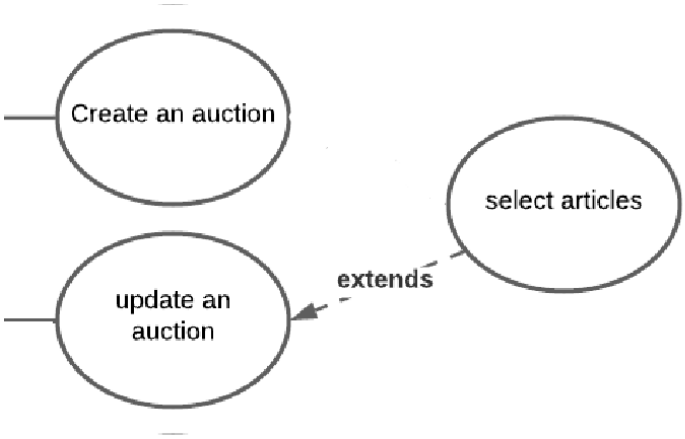



Figure 1.6: Example of an Extend Relationship

- **Includes Relationship:** The includes relationship indicates that a base use case includes the behavior of another use case. It represents a modular or reusable behavior that is common to multiple use cases.

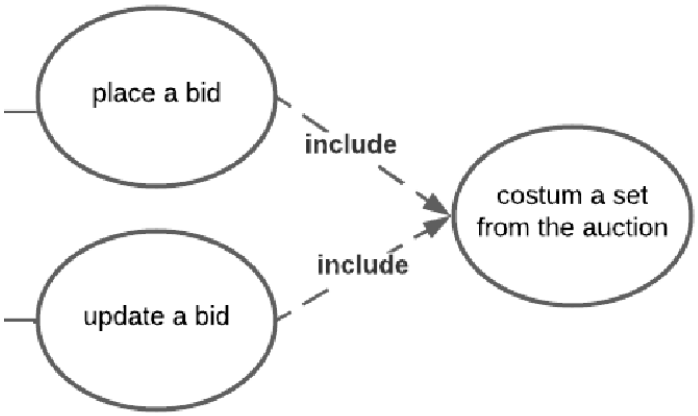



Figure 1.7: Example of an Include Relationship

### Sequence Diagram

The Sequence diagram depicts the dynamic behavior of the system by showcasing the interactions and message exchanges between various objects or components. It illustrates the flow of control and communication, helping in understanding the system's logic and behavior during run-time.

**Actor** An actor represents an external entity, such as a user or another system, that interacts with the system being modeled. It is depicted as a stick figure or a labeled box at the edge of the diagram.

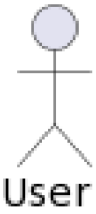



Figure 1.8: Example of an Actor

**Object** An object represents a specific instance of a class or component in the system. It represents a participant in the sequence of interactions. Objects are depicted as rectangles with the name of the object written inside.

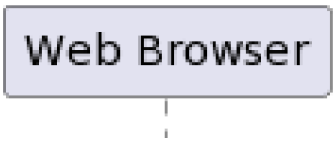



Figure 1.9: Example of an Object

**Lifeline** A lifeline represents the existence of an object over a period of time during the execution of a sequence diagram. It is depicted as a vertical dashed line extending from the object's box.

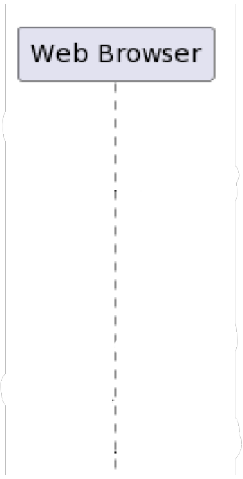



Figure 1.10: Example of a lifeline

**Message** A message represents a communication or interaction between objects. It signifies the flow of information or control between objects. Messages can be synchronous, asynchronous, or create or destroy messages, and they are depicted as arrows between lifelines.

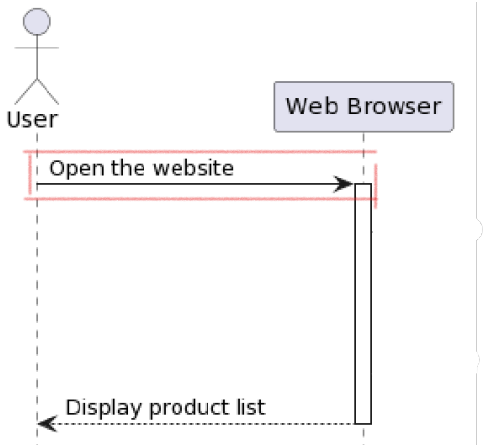



Figure 1.11: Example of a Message

**Activation** An activation represents the period of time when an object is executing a method or operation. It shows the duration of the method call and is represented by a thin rectangle vertically aligned with the lifeline.

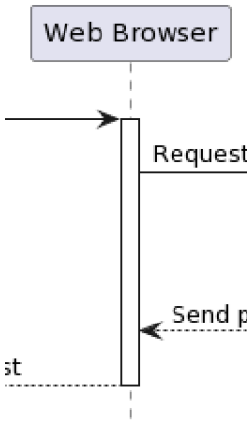



Figure 1.12: Example of an Activation

**Return Message** A return message represents the response or return value sent from the called object back to the calling object. It indicates the completion of the invoked method or operation and is depicted as an arrow with a dashed line.

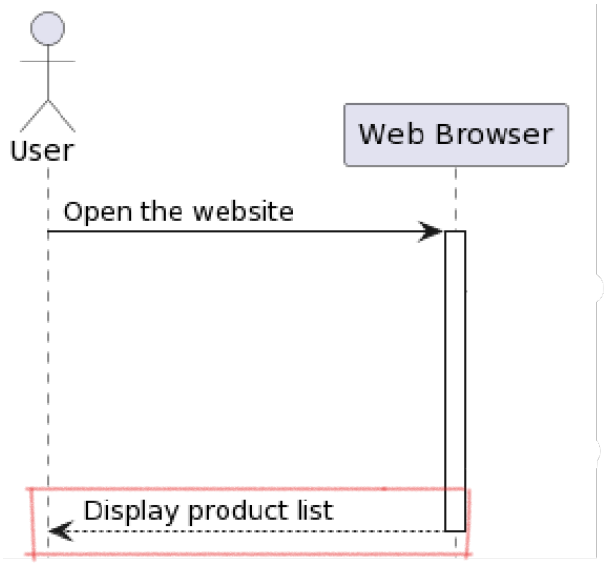



Figure 1.13: Example of a Return message

**Self Message** A self message represents a message sent from an object to itself. It denotes an internal operation or recursion within the object. It is depicted as a looped arrow or an arrow pointing back to the same lifeline.

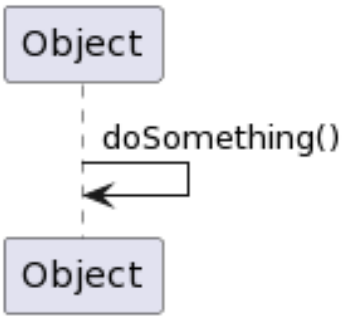



Figure 1.14: Example of a Self message

**Combined Fragment** A combined fragment represents a grouping of messages to express alternative or parallel flows in a sequence diagram. It helps depict conditional logic or looping behavior. Combined fragments are depicted as boxes with specific keywords, such as ”alt,” ”opt,” ”loop,” etc.

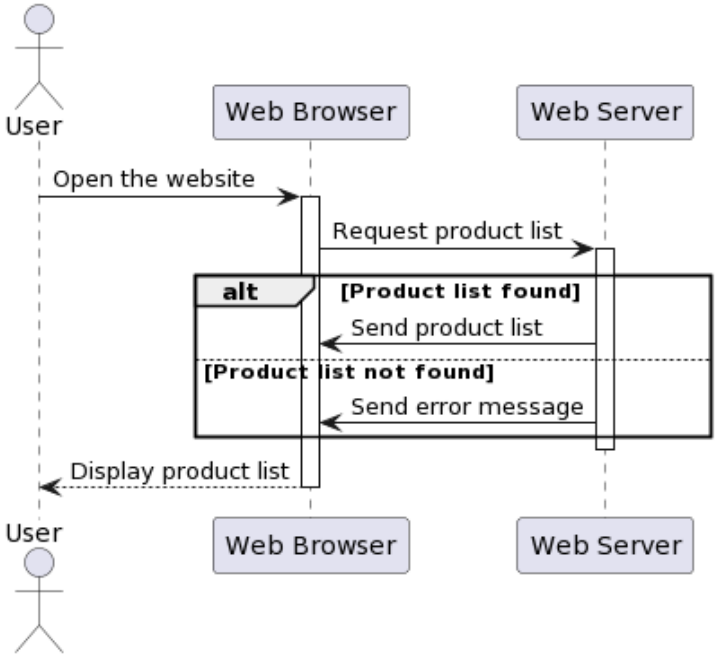



Figure 1.15: Example of a Combined fragement

**Interaction Operand** An interaction operand represents a condition or constraint applied to a combined fragment. It specifies when the combined fragment should be executed based on a certain condition or criteria.

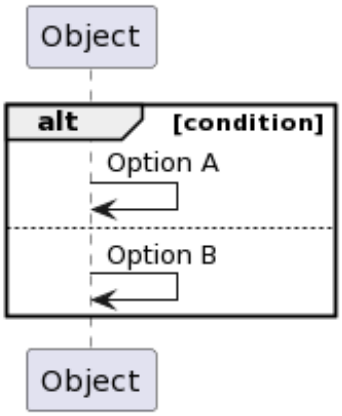



Figure 1.16: Example of an object

Sequence diagrams capture the dynamic behavior of the system, illustrating how objects interact and communicate during runtime. They are valuable for understanding the flow of control, the sequence of method invocations, and the overall logic and behavior of the system.

By utilizing these additional UML diagrams alongside the Use Case diagram and the sequence diagram, the development team and seniors can gain a good view of the system, capturing its structure, behavior, and deployment aspects. These modeling and implementation tools serve as a common language for communication among stakeholders and facilitate effective collaboration during the development process[6].

### 1.3.2 Entity-Relationship (ER) Diagram

The Entity-Relationship diagram represents the logical structure of the system's data by showcasing the entities (objects or concepts) and their relationships. It helps in understanding the data requirements and the associations between different entities, guiding the design of the system's database schema.
ER diagrams are not part of the UML model. it was developed as a separate modeling technique specifically for database design.

**Entity** An entity represents a real-world object, concept, or thing with independent existence. It is depicted as a rectangle in the ER diagram. Entities have attributes that describe their properties.

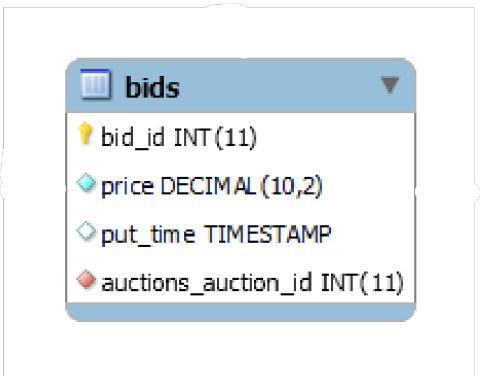



Figure 1.17: Example of an Entity

**Attribute** An attribute is a characteristic or property of an entity. It provides additional information about the entity. Attributes are represented as ovals connected to the respective entity.

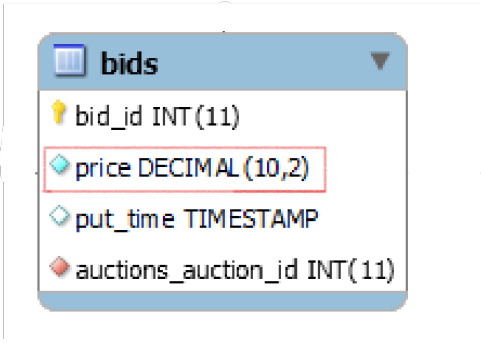



Figure 1.18: Example of an Attribute

**Relationship** A relationship represents an association or connection between two or more entities. It describes how entities interact with each other. Relationships are shown as diamond shapes and labeled to describe the nature of the association.

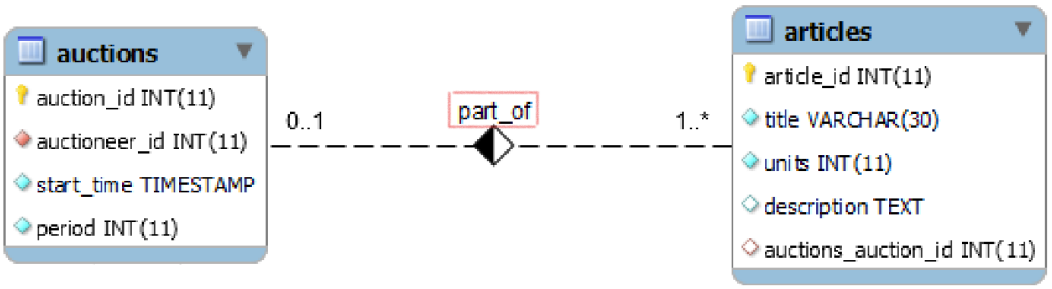



Figure 1.19: Example of a Relationship

**Cardinality** Cardinality specifies the number of instances of one entity that are related to the number of instances of another entity in a relationship. It defines the participation constraints of entities in a relationship, such as one-to-one, one-to-many, or many-to-many relationships, like the blow bids entity:

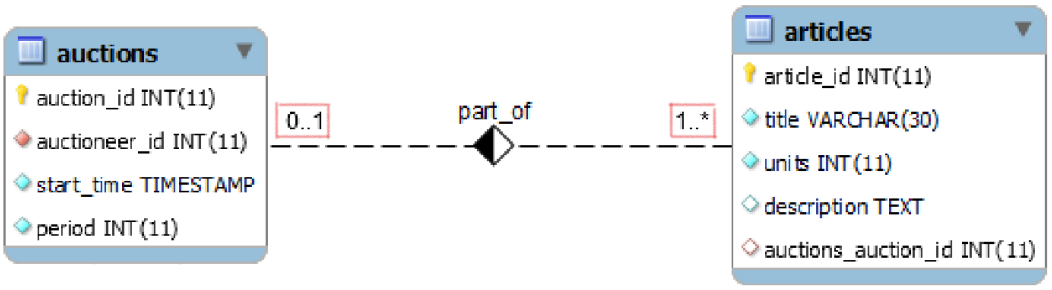



Figure 1.20: Example of Cardinalities

**Primary Key** A primary key is a unique identifier for an entity. It uniquely identifies each instance of an entity and is crucial for database operations. It is typically represented with an underline or bold font in the attribute list.

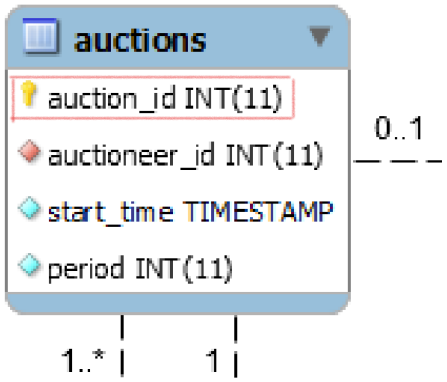



Figure 1.21: Example of Primary key

**Foreign Key** A foreign key is a reference to the primary key of another entity. It establishes a link between two entities and represents a many-to-one relationship. It

helps maintain data integrity and enforce referential integrity in the database.

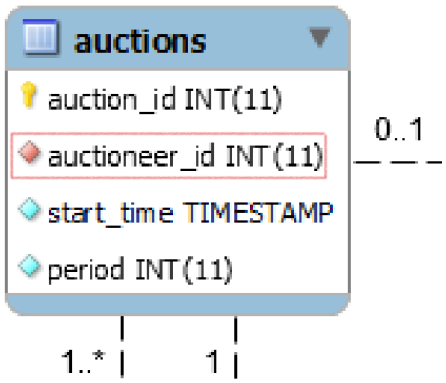



Figure 1.22: Example of Foreign key

These elements collectively provide a visual representation of the data structure, relationships, and constraints within the system. They help in understanding the data requirements, designing the database schema, and communicating the system's structure to stakeholders and developers.

## 1.4 Dynamic Implementation Platform: Progressive Web Applications

In the dynamic world of modern software development, Dynamic Implementation Platforms (DIPs) have gained prominence. DIPs go beyond technologies or frameworks; they encompass best practices adopted by the web community to mimic native app experiences. A key player in this realm is Progressive Web Applications (PWAs), which combine various elements like Service Workers, App Shell, Web App Manifest, and Push Notifications to deliver native-like capabilities.

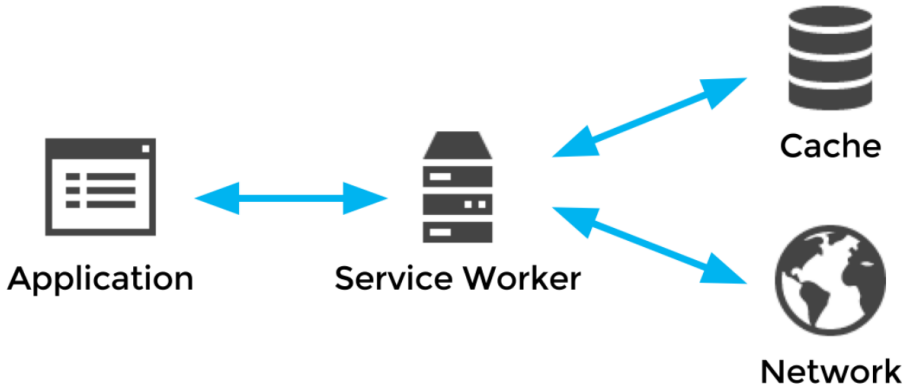



Figure 1.23: Service Worker caching strategy of react[21]

PWAs offer easy installation, swift loading even on slow networks, and user engagement through push notifications. Despite challenges like cross-browser compatibility and

hardware limitations, PWAs, powered by DIPs, are transforming digital experiences across industries such as e-commerce and online auctions, offering user-centered innovation [21].

## 1.5 Literature domain

### 1.5.1 The Basics of Database Management Systems (DBMS)

In the scholarly work "The Basics of Database Management Systems (DBMS)" by Anjard and Ronald P[3], the authors discuss the pivotal role of DBMS in efficient data organization. They trace the evolution of DBMS from early implementations, highlighting the transition from conventional file methods to modern design practices. The authors provide a systematic approach to database design, starting with the development of a conceptual data model and emphasizing the use of visualization tools like entity-relationship charts. They delve into the strategic determination of optimal data storage, considering factors like partitioning, centralization, or replication. The differentiation between conceptual and logical database design is explored, with analytical tools like the affinity matrix aiding in the examination of entities and attributes.

In the context of online environments, the authors stress the importance of security measures, including rapid response times and data safeguarding for external users. They underscore the multifaceted responsibilities of DBMS in averting concurrent updates, detecting deadlocks, and managing adverse scenarios through proactive measures such as backups and event logging. Notably, the authors elucidate that users anticipate rapid response times, typically ranging from 1 to 5 seconds, in interactions with online DBMS. The authors also highlight the streamlined structure definitions and data restructuring capabilities that these tools offer. Moreover, the article underscores the indispensable role of database systems in the context of PC systems, encompassing both IBM-compatible and MAC platforms. In conclusion, Anjard and Ronald P's work offers a comprehensive overview of DBMS fundamentals and their enduring impact on contemporary data management practices.

### 1.5.2 Graph Databases for Beginners

The article "Graph Databases for Beginners" by Sasaki[20] introduces graph databases and their relevance in today's data landscape. Sasaki emphasizes the importance of relationships in data and highlights how graph databases excel in managing meaningful connections, offering superior performance, flexibility, and agility compared to traditional relational databases. The article also discusses limitations of relational databases in handling data relationships.

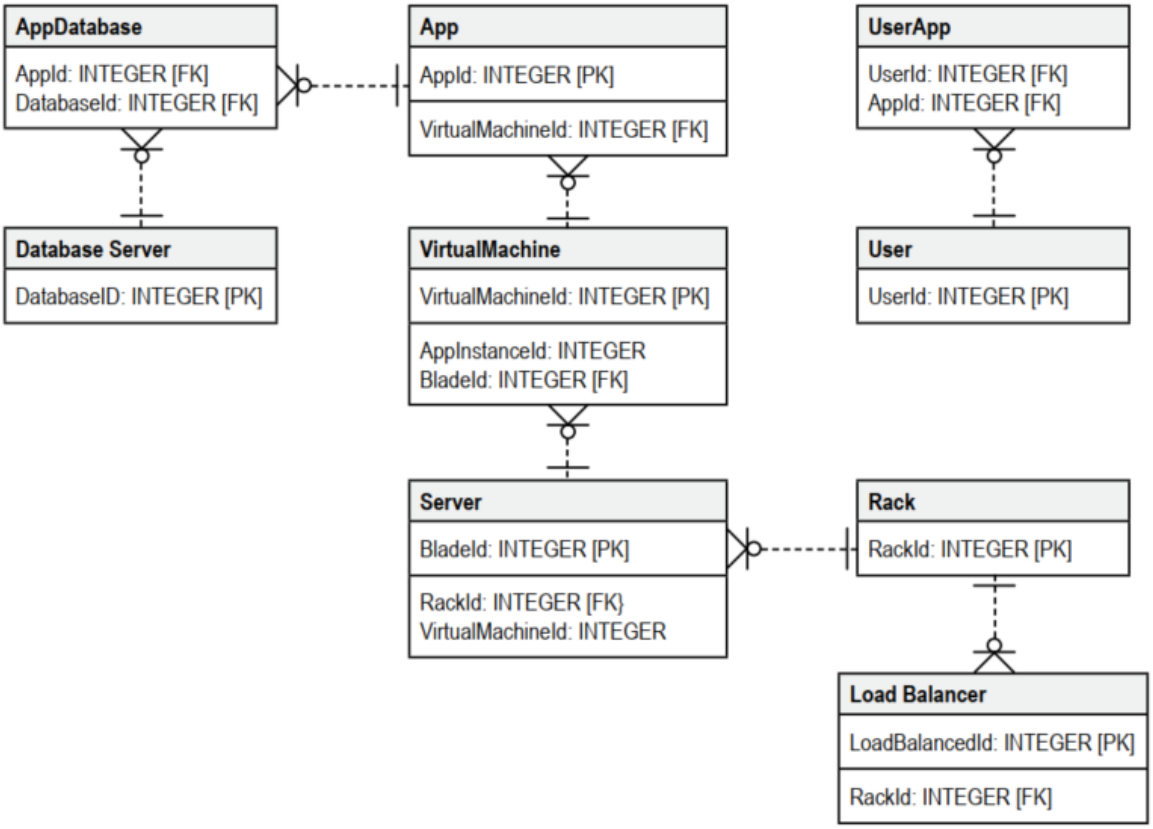



Figure 1.24: sasaki's rational database initial data model [20]

It explains data modeling basics, showcasing the simplicity of creating a graph data model compared to traditional relational databases. The article provides insights into common data modeling errors to avoid, using a fraud detection application as an example. It underscores the significance of a database query language, making querying more understandable and accessible for various stakeholders. Additionally, the article briefly touches upon graph theory's application in predictive modeling, emphasizing the benefits of graph technology in handling vast and varied data. In conclusion, the article serves as an introductory guide to graph databases, offering insights into their advantages, data modeling techniques, query languages, and practical applications. It encourages exploration of graph technology's potential benefits.

### 1.5.3 Understanding Database Performance Inefficiencies in Real-world Web Applications

In "Understanding Database Performance Inefficiencies in Real-world Web Applications" by Yan[23], the article explores performance optimization techniques in Rails-based web applications. It introduces the Action Flow Graph (AFG) to identify performance bottlenecks resulting from slow queries and rendering. Queries are categorized into those within loops and those outside, with a focus on optimizing the former to enhance scalability. Additionally, the article emphasizes the importance of caching query results, especially through cross-action caching, as a crucial step for improving performance.

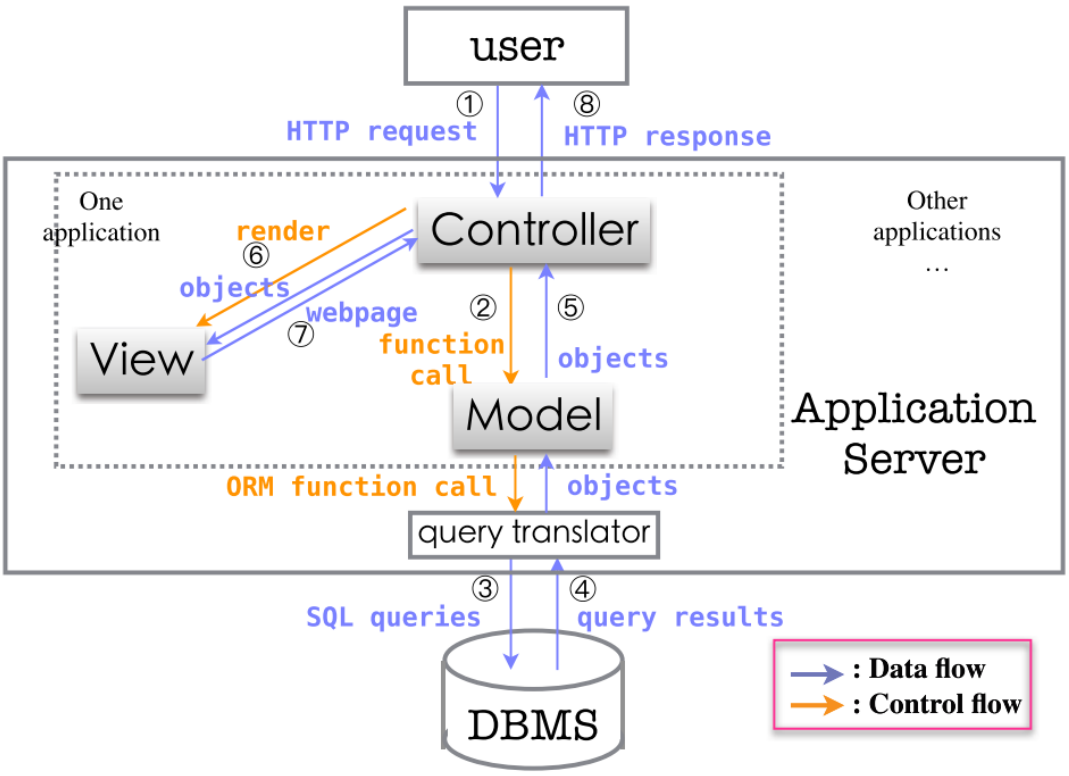



Figure 1.25: Architecture of a rails application [23]

The second part of Yan's article delves into optimizations related to the physical design of stored data in the database. It discusses how many queries use only a subset of stored object fields and suggests methods such as partial evaluation and customized layouts to expedite query execution. In conclusion, Yan's article highlights the significance of analyzing user actions, optimizing queries, and refining data design to enhance the performance and scalability of Rails web applications, offering valuable insights for developers seeking effective performance enhancements.

### 1.5.4 The Unified Modeling Language for Object-Oriented Development

In the documentation of the Unified Modeling Language (UML) by Booch[6], key challenges in distribution and concurrency are identified as significant areas of focus for future UML development. Distribution concerns encompass object allocation, migration, grouping, registration mechanisms, and remote communication. Concurrency involves processes, threads, class-object allocation, communication patterns, and resource allocation within a system. To tackle these issues, the UML creatively employs existing features, introducing properties like "location" to model object distribution and composite objects to represent distribution units. Interfaces play a pivotal role in modeling distributed systems by enabling clients to subscribe to interfaces and implement them, even if implementations are remote.

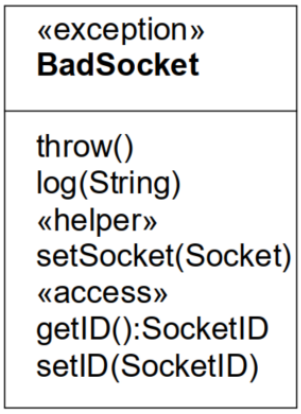



Figure 1.26: An example of the syntax of a stereotypes [6]

In addition, the UML addresses physical object distribution and migration by representing dependencies between objects and nodes, using the "location" property to specify node attachment, and designating classes to reside on particular nodes. Concurrency modeling introduces tasks as primary entities representing control threads, rooted in active objects and implemented either within a single process or distributed across address spaces. The documentation also explores proposals for UML improvements, including uniform type-instance notation, traceability relationships, and the integration of patterns as first-class modeling elements, aiming to enhance the UML's capabilities for modeling distribution, concurrency, and real-time systems, providing a more robust framework for software development.

### 1.5.5 Evaluation and Implementation of Progressive Web Application

In their article "Evaluation and Implementation of Progressive Web Application" [21], Thakur and Parbat delve into Progressive Web Applications (PWAs), highlighting their components and advantages, which bridge the gap between web and native applications. They emphasize key PWA constituents, including the Service Worker, Web App Manifest, App Shell Model, and Web Push Notifications, while highlighting the central role of React.js in creating dynamic and efficient user interfaces.


Figure 1.27: Combination of web site with manifest to give an app look[21]

The authors spotlight the Service Worker's significance in enabling offline functionality, caching, and push notifications by intercepting network requests and caching resources. The Web App Manifest, a JSON file containing app details, enriches the user experience by allowing PWAs to be added to home screens as standalone apps. They also discuss the App Shell Model, which separates static and dynamic content, enhancing user experiences with swift loading times and seamless navigation. Web Push Notifications are explored for timely updates and user engagement. Thakur and Parbat further provide insights into their PWA news app's development environment, tools used, and practical implementation, demonstrating the effectiveness of PWAs in delivering dynamic components and an improved user interface within web browsers.

### 1.5.6 Specification and implementation of dynamic web site benchmarks

In the dynamic content website domain, performance evaluation is vital. Amza's work, 'Specification and Implementation of Dynamic Web Site Benchmarks' [1], introduces tailored benchmarks for dynamic content websites, covering various applications. Amza's benchmarks and flexible workload generator tool enable performance exploration, highlighting distinct bottlenecks for different website types. Open-source benchmarking tools promote collaboration and standardized evaluation practices. This work is relevant to our project, helping identify performance bottlenecks as we develop a high-performance combinatorial auction web app.

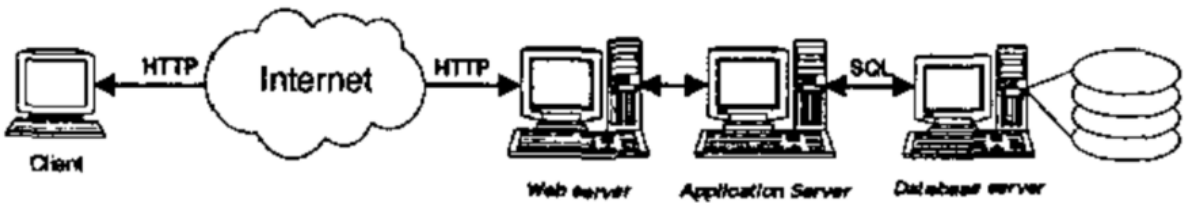



Figure 1.28: Typical Configuration of a Dynamic Content [1]

### 1.5.7 Characterizing Secure Dynamic Web Applications Scalability

Guitart's research, "Characterizing Secure Dynamic Web Applications Scalability" [8], explores the balance between security and scalability in Tomcat application servers with SSL. The study uses real-world simulations with Tomcat, the RUBiS benchmark, and Httperf. A performance analysis framework, combining Java Instrumentation Suite (JIS) and Paraver, scrutinizes the server's behavior, emphasizing SSL handshake impacts on scalability. The research uncovers the need for more processors to sustain performance due to SSL's computational demands and highlights the shift from resumed to full SSL handshakes, affecting server performance. Overall, Guitart's work provides insights into optimizing server configuration and resource allocation in secure web applications.

## 1.6 Problematic

The domain of combinatorial auction systems one of the largest domains both of research and business with various complexities that necessitate a strategic approach to exploit from and in our case develop a decent web application. Within this context, we encounter a range of challenges that require our attention and innovative solutions[1, 3, 8, 21, 23] :

1. **Scalability and Growth Management**: As the participant count swells and the auction items and bundles multiply, our web application must effortlessly manage the important data volumes and intricate calculations. The main goal here is to sustain a seamless user experience, regardless of the concurrent user count. To this end, we must architect a backend infrastructure that is elastic, capable of accommodating a growing number of participants without any compromise on system responsiveness.

2. **Real-Time Dynamics**: Operating within a dynamic auction environment implies that bid information faces continuous alterations. Providing real-time updates. Bidders demand immediate insights into auction statuses and bid progressions. To meet this requirement, we must establish a highly efficient synchronization mechanism that bridges the communication gap between the front-end and back-end systems. This mechanism should enable bid updates to be swiftly and accurately relayed to participants.

3. **User-Centric Interface Design**: The interface of our web application will be the bridge between participants and the complex auction landscape. Ensuring user-friendliness and simplicity. We must craft an intuitive interface that effortlessly

guides bidders through auction listings, item specifics, bid placements, and progress tracking. The presentation of information through visual aids and prompt feedback mechanisms will be crucial to enriching the overall user experience.

4. **Fortified Security and Data Privacy**: The terrain of combinatorial auctions harbors sensitive data, including bids, user information, and transaction particulars. Consequently, the implementation of robust security protocols is non-negotiable. Our application must uphold the integrity and confidentiality of this data through encryption, authentication mechanisms, and stringent access controls. This will assure participants that their interactions with the auction platform are shielded from unauthorized access and tampering.

To solve these difficult problems and make things work well, we need to use a variety of methods. We're starting a project that relies on advanced math rules, a structure that can grow when needed, instant communication rules, and a design that's easy to understand and use. This mix will set the base for a computer program that not only faces current problems directly, but also improves how auction systems work together. By making a place where people who want to buy things can comfortably take part in complicated auctions, our website aims to change how users feel and what results they get in this field.

# Conclusion

In Chapter 1, we comprehensively covered essential concepts and tools for building combinatorial auction web applications. We emphasized the significance of databases, distributed database networks, and effective schema design for system performance. We also explored modeling with UML diagrams, facilitating effective communication and design visualization. Additionally, dynamic implementation platforms like React and PWA principles were introduced to enhance user experiences.

Furthermore, in Chapter 1, we delved into Database Management Systems (DBMS) and Unified Modeling Language (UML) evolution, emphasizing data center management and UML advancements for modeling. In Chapter 2, we will lay the groundwork for understanding combinatorial auctions, defining key concepts and terminologies to prepare us for the project's challenges. This foundational knowledge will guide our technical implementations and project considerations in the future.

# 2

# Foundations of Combinatorial Auctions

## Introduction

In this chapter, we delved into the fundamental concepts of auction mechanisms and mathematical models that supported the development of our dynamic web application for implementing combinatorial auctions.

Combinatorial auctions, where bidders could bid on sets of items, provided a flexible resource allocation approach that allowed participants to express their preferences more efficiently.

We explored various auction mechanisms and their suitability for our project, including well-known ones like English, Dutch, and Vickrey auctions. Additionally, we examined mathematical models used for winner determination problem, which served as the foundation for our auction algorithms. We also reviewed existing literature on solving combinatorial auction problems, equipping us with the knowledge needed to make informed design decisions for our dynamic auction web application.

## 2.1 General definition

Combinatorial auctions are unique auction formats that enable bidders to submit bids for sets of items rather than individual ones, enhancing flexibility in resource allocation. These auctions aim to maximize profit or social utility while adhering to constraints like fairness and efficiency, resulting in complex bidding spaces.

Designing and analyzing combinatorial auctions involve drawing from combinatorial optimization theories and algorithmic techniques. This encompasses creating models and algorithms to efficiently allocate goods among bidders, considering factors like preferences, budgets, and item inter-dependencies.

Combinatorial auctions find applications in diverse fields like e-commerce, procurement, and spectrum auctions, promoting cooperation and efficiency. In the context of our auction web app project, we'll implement mechanisms for set-based bidding and efficient allocation, offering users an intuitive interface for personalized collections from a predefined item set. By leveraging theoretical foundations and algorithmic techniques, our project aims to enhance bidder experiences and contribute to efficient online resource allocation[4].

## 2.2 Auction formalism (definition and basic concepts)

A combinatorial auction is a type of auction where bidders can bid on a set or combinations of items rather than bidding on individual items separately. This allows for more flexibility in expressing preferences and capturing the interaction among different items. The general definition of a combinatorial auction bound the following characteristics :

- Bidders: The participants in the auction who submit bids. Bidders are individuals or entities that express their interest in acquiring certain items or sets through the auction. They compete with each other by submitting bids, which represent the prices they are willing to pay for the items.
- Items: The goods or services being auctioned. Items can vary in nature and can include physical products, digital assets, services, or any other tradable entities. These items are made available for bidders to acquire through the auction process.
- A set: Combinations of items that bidders can bid on. In a combinatorial auction, bidders have the opportunity to bid on a mix of items rather than individual items. sets allow for more flexible bidding strategies as bidders can express their preferences for specific combinations of items.

## 2.3 Auction mechanism and types

In the realm of auctions, various mechanisms are employed to facilitate the buying and selling of goods or services. Understanding the different auction types and their characteristics is crucial for optimizing outcomes and achieving desired project objectives. In this discussion, we will explore several commonly used auction mechanisms, examining their features and suitability within the context of our project.

1. **English Auction:** This is the most well-known mode of auction. It is an open ascending-price auction. Paul Milgorm [Milgrom 1989] describes it as follows:

   The auction begins when the auctioneer announces the reserve price and proceeds to receive increasingly higher bids from participants until there are no more bidders willing to increase the current bid. The item is then awarded to the highest bidder.

   The reserve price may be kept secret in some cases, and the auctions start from zero. The main reason for this is to prevent the formation of coalitions among bidders to keep the competition open and avoid driving the winning price to the lowest possible threshold by the last bidder.

The English auction mechanism strongly encourages competition. It is not uncommon to witness enthusiastic but inexperienced bidders placing bids surpassing the actual value of the item. This competitive mechanism is sometimes humorously referred to as the "winner's curse."

2. **Dutch Auction:** This is an open descending-price auction. A high price is announced initially, and then the auctioneer gradually decreases it until a participant claims the item at the current price.

   The main difference from the English auction lies in the presumed advantage. In the Dutch auction, the winner, with the highest estimated price for the item, is more likely to claim the item as soon as the price becomes equal to or lower than their offering. In contrast, in an English auction, the winner could gradually increase their bid and potentially obtain the item for a much lower price than their maximum bid.

3. **Best Price Auction:** This is a secret first-price auction, where "sealed-bid" means that participants' bids are concealed from each other. The auction winner pays the exact amount of their bid.

   The process unfolds in two stages: the bid submission phase, followed by the winner determination phase after reviewing the bids. This auction type is static, with participants able to propose only one bid, emphasizing the importance of careful preparation in bid submission.

   One drawback of this type of auction is that participants' bids tend to be lower than their valuations.

4. **Vickrey Auction:** Similar to the sealed-bid first price auction, in a Vickrey auction, bids are hidden from other participants. The distinction lies in the fact that the winner, having submitted the best bid, pays not their bid amount but the second-highest bid offered for the item during the auction.

   Vickrey [Vickrey 1961], who introduced this method, demonstrated that the dominant strategy for a buyer is to submit their true valuation of the item. The participant's gain in participating in the auction is calculated as follows:

   If their bid is the highest, and another bid is the second highest, the surplus (gain) for the participant is: $Gi = vi - bj$. Otherwise, if the participant's bid is not the highest, they do not win, and their surplus (gain) is zero: $Gi = 0$.

   It is in the participant's interest to maximize their bid to increase their chances of winning the auction. This strategy leads to each participant's bid being equal to their valuation of the item: $bi = vi$.

5. **Oral Auction:** An auction format where bidding occurs verbally, usually in a physical setting with an auctioneer leading the process.

6. **Written Auction:** An auction format where bids are submitted in writing, often through sealed envelopes or online platforms.

7. **Single Item Auction:** An auction designed to sell or buy a single product or service at a time, with participants placing bids on that particular item.

8. **Combinatorial Auction:** A type of auction intended to sell or buy a combination or set of items simultaneously through a single submission. This approach allows participants to bid on a bundle of items rather than individual items.

These various auction mechanisms serve to evaluate and allocate prices for products and services, especially when determining fair valuations is challenging. The transition to online platforms, driven by the internet, has expanded the reach and popularity of auctions, enabling people to engage in bidding from various locations.[4]

## 2.4 Different mathematics WDP modeling

Based on the characteristics of the products being sold, three types of models can exist: auction of a single item in multiple units, auction of multiple items of different types in a single unit each, and auction of multiple items of different types in multiple units each. In the latter case, we can have the same number of units for each type of item, or different quantities.

### 2.4.1 Modèle I

Let $\beta$ be the quantity of the item being auctioned facing $n$ potential buyers, and let $S_j$ be the bid from bidder $E_j$, $\quad j \in \{1, \ldots, n\}$, This bid consists of the offered price $c_j$ and the desired quantity of items $a_j \in \mathbb{N}^*$ wanted.

The decision variables are defined by:

$$x_j = \left| \begin{array}{ll} 1, & \text{if the offer of } E_j \text{ is accepted,} \\ 0, & \text{otherwise.} \end{array} \right. \quad j = 1, \ldots, n.$$

the formulation of problen is as follow:

$$PDG = \left| \begin{array}{lll} F_(x) = & \max \sum_{j=1}^{n} c_j x_j & \\ & \sum_{j=1}^{n} a_j x_j & \leq \beta \\ & x_j & \in \{0,1\} \quad j = 1, \ldots, n. \end{array} \right. \tag{2.1}$$

Where the objective here is to maximize the total profit of the seller, calculated by the sum of the prices of winning bids, and the constraint limits the availability.

The sum of all bids must exceed. $\beta$, ie $\sum_{j=1}^{n} a_j \geq \beta$.
this modelisation is the same as that of the one-dimensional knapsack problem.

### 2.4.2 Modèle II

For this model, $m$ different types of items are being auctioned, each with only one unit. Let the binary vectors $a_j$ contain the preferences of the bidders, such that $a_{ij} = 1$ if the

article $i$ is desiered by the bidder $j$, and want 0 otherwise.

each offer comes with an evaluation $c_j$ representing the price allocated to this bid by the bidder $E_j$.

the mathematical model is written as follow:

$$PDG = \left| \begin{array}{lll} F(x) = & \max \sum_{j=1}^{n} c_j x_j & \\ & \sum_{j=1}^{n} a_{ij} x_j \leq 1 & i = 1, \ldots, m; \\ & x_j \in \{0,1\} & j = 1, \ldots, n. \end{array} \right. \tag{2.2}$$

in this model, the sum of demandes has to exceed 2, ie $\sum_{j=1}^{n} a_{ij} \geq 2$ for at least one index $i \in \{1, \ldots, m\}$.

This model consist of finding the winning offers, that maximise the profit of the seller under the constrainte, that each object cannot be related to more then one buyer.

### 2.4.3 Modèle III

This model is the generalisation of the precedent model, here the number of units for each type of articles is superior or equal 1, i.e. $\beta_i \in \mathbb{N}^* \ \ \forall i \in \{1, \ldots, m\}$. we can distanguish two cases for this model:

- The first case is $WDP$ of equal quantities $\beta_i$ (homogeneous) and
- the second is of different quantities $\beta_i$ (heterogeneous).

let $a_{ij}$ the number of entities of the article $i$ wanted by the bidder $E_j$, and let $c_j$ the price associated to the group of his offer.

the variables of decision :

$$x_j = \left| \begin{array}{ll} 1, & \text{if the offer of } E_j \text{ is accepted,} \\ 0, & \text{otherwise.} \end{array} \right. \quad j = 1, \ldots, n.$$

the mathimatical model associated to this problem is as follow :

$$PDG = \left| \begin{array}{lll} F(x) = & \max \sum_{j=1}^{n} c_j x_j & \\ & \sum_{j=1}^{n} a_{ij} x_j \leq \beta_i & i = 1, \ldots, m \\ & x_j \in \{0,1\} & j = 1, \ldots, n. \end{array} \right. \tag{2.3}$$

for the reasons For the model to be accurate, the sum of bids presented by all bidders must exceed the availability in quantities for each item $i$ (the demand exceed the offer) i.e. $\sum_{j=1}^{n} a_{ij} \geq \beta_i, \ \ \forall i = 1, \ldots, m.$

## 2.5 Literature on its resolution

### 2.5.1 Multiobjective optimization

Matthias Ehrgott's work is likely to provide an in-depth exploration of the principles and techniques involved in solving multiobjective optimization problems [7]. The author's expertise seems to encompass the theoretical foundations of multiobjective optimization, as well as practical algorithms and methods for solving complex problems with multiple conflicting objectives.

The excerpt discusses various approaches, such as scalarization techniques (including weighted-sum, e-constraint, and compromise programming methods), efficient solution concepts, and their relevance in different optimization scenarios. It also touches on the challenges posed by multiobjective combinatorial optimization problems and presents methods to address them.

Ehrgott's work appears to emphasize the intricate relationship between objectives, constraints, and decision variables in optimization problems with multiple conflicting goals. The content suggests a combination of theoretical insights and algorithmic approaches, potentially catering to both researchers interested in understanding the underlying mathematical principles and practitioners seeking effective methods to solve real-world multiobjective optimization problems.

### 2.5.2 Fuzzy programming for multiobjective

The research article "Fuzzy programming for multiobjective 0-1 programming problems through revised genetic algorithms" by Sakawa and Kato combines fuzzy logic, multiobjective optimization, and genetic algorithms to address complex 0-1 programming challenges. Fuzzy logic handles uncertainty with linguistic variables, multiobjective optimization optimizes conflicting goals, and genetic algorithms use evolutionary techniques.

Sakawa and Kato introduce revised genetic algorithms with double string representation, new crossover operators, and inversion techniques to tackle multiobjective 0-1 programming with fuzzy goals. This innovative approach overcomes traditional GA limitations in complex optimization with multiple objectives and constraints.

Their work provides valuable insights into solving real-world optimization problems with uncertain objectives, contributing to the optimization and evolutionary computation field by enhancing GAs for multiobjective and fuzzy scenarios[17].

### 2.5.3 The Winner Determination Model and Computation

The research article "Winner Determination Model and Computation for Linear Arrangement of Booth Auction[19]" by Sariddichainunta and Sinapiromsaran delves into the winner determination problem in booth auctions, addressing computational challenges. The study explores two methodologies, linear programming and dynamic programming, to efficiently solve this problem while considering bidder valuations and booth arrangements.

Experiments and simulations in both single-line and double-line booth scenarios consistently favor dynamic programming over linear programming as the preferred approach. This research's implications extend to resource allocation scenarios, including internet ad slot auctions, emphasizing the efficiency of dynamic programming in tackling complex winner determination problems across various domains.

### 2.5.4 Combinatorial Auction-Based Mechanisms

Sharrukh Zaman's article "Combinatorial Auction-Based Mechanisms for VM Provisioning and Allocation in Clouds[24]" explores cloud computing, mechanism design, and auction theory to optimize resource allocation. By addressing the Virtual Machine Allocation Problem (VMAP) and introducing innovative mechanisms like CA-GREEDY and CA-LP, it offers efficient solutions. Additionally, it tackles the Dynamic Virtual Machine Provisioning and Allocation Problem (DVMPA) with CA-PROVISION, extending its impact to real-time demands.

The article's multidisciplinary approach promises enhanced resource utilization and collaboration across domains, leaving a lasting mark on cloud computing and mechanism design research.

### 2.5.5 Integer Programming for Combinatorial Auction Winner Determination

Andersson and Tenhunen's article, "Integer Programming for Combinatorial Auction Winner Determination[2]" focuses on the computational challenges of solving the winner determination problem in combinatorial auctions. They explore the use of integer programming and compare it with specialized algorithms.

The paper highlights the importance of benchmarking and empirical analysis across various bid distribution scenarios. They evaluate two key algorithms: mixed-integer programming with CPLEX and specialized winner determination algorithms, considering performance, execution times, and optimality in real-world bid scenarios.

The article emphasizes the potential benefits of applying established operations research and combinatorial optimization techniques to address winner determination challenges. It also recognizes the difficulty of handling realistic bid distributions with specialized algorithms. Overall, it offers a comprehensive analysis of winner determination algorithms, considering benchmarking, bid distribution characteristics, and the application of optimization techniques in electronic commerce and auction design.

## Conclusion

In summary, this chapter has provided a solid foundation for our combinatorial auction web application. We've explored diverse auction mechanisms and mathematical models like Modèle I, Modèle II, and Modèle III, gaining insights into resource allocation and bidder preferences.

The review of existing literature underscores the role of algorithms in solving winner determination problems. This interdisciplinary approach spans economics, optimization, and computer science.

With this knowledge, we're ready to move forward with developing our dynamic auction web app. In the upcoming chapters, we'll transform these theories into a user-friendly digital platform that optimizes goods allocation in the online marketplace.

# 3

# Modeling of Bd and tools of WDP resolution

## Introduction

In this chapter, we will lay the foundation for our state-of-the-art Combinatorial Auction System. We commence by constructing a robust database model, utilizing an Entity-Relationship (ER) diagram, to meticulously organize essential data entities. Following this, we will introduce the dynamic Winner Determination Problem (WDP) model, providing a glimpse into the future of auctions. This model emulates English combinatorial auctions, where bidders will have continuous opportunities to refine their offers throughout the exercise period. Our forward-looking methodology encompasses handling new bids, seamless updates, sophisticated conflict resolution, and dynamic programming—a path toward efficient problem-solving. These components will constitute the core of our forward-looking system, ensuring structured data management and real-time WDP resolution, crafting a dynamic, responsive, and future-ready auction platform.

## 3.1 Database Model Associated with PDG

### 3.1.1 Entity-Relationship (ER) Diagram

The ER diagram depicts the logical structure of the database by illustrating the entities, attributes, and relationships between them. It serves as a blueprint for designing the database schema and defining the relationships between different data entities.

In the context of the combinatorial auction webapp, an ER diagram will be used to model the database schema, including entities like users, auctions, items, bids, and collections. The relationships between these entities, such as one-to-many or many-to-many associations, will be established and represented in the ER diagram.

Here is a simplified example of an ER diagram for the combinatorial auction webapp:

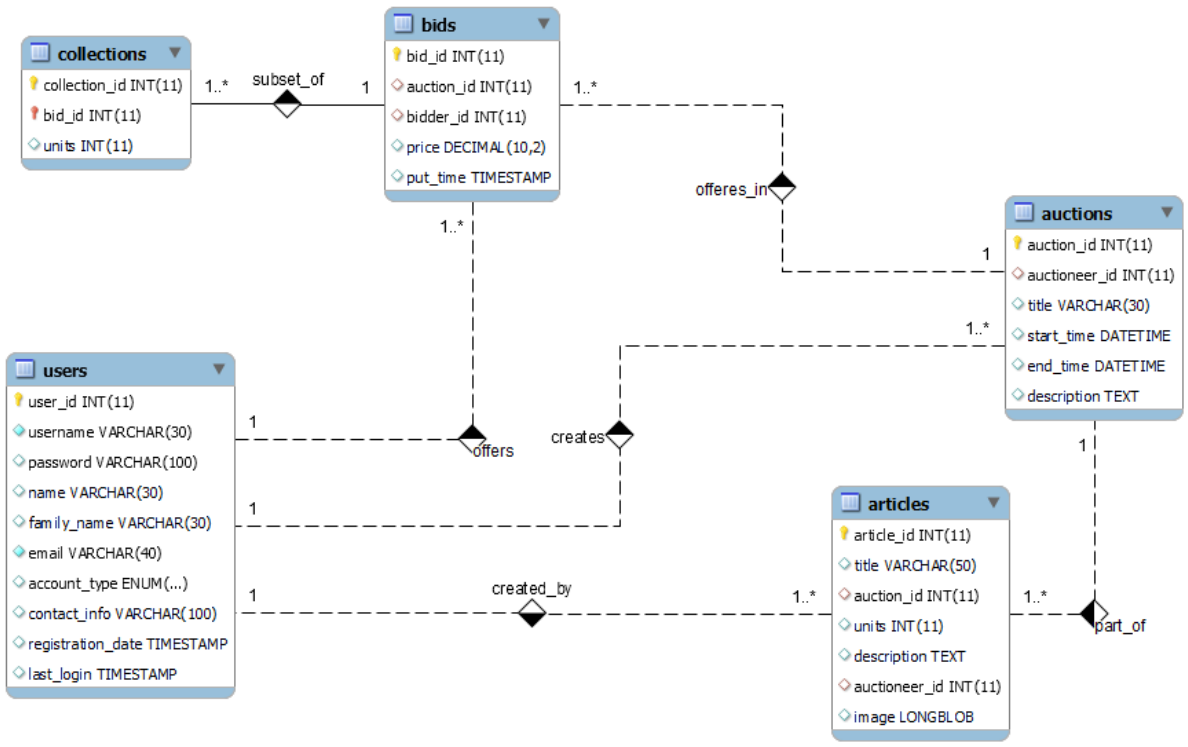



Figure 3.1: Entity-Relationship (ER) Diagram

The database structure provided represents the tables and their relationships in the Combinatorial Auction System. The structure is described as follows:

**Users:**

- Users are individuals or entities registered within the system.
- User details include a unique user ID, username, password, personal name, family name, email address, account type (bidder or auctioneer), contact information, registration date, and last login timestamp.

**Auctions:**

- Auctions are events hosted by auctioneers to sell articles.
- Auctions are identified by a unique auction ID and contain information such as the auctioneer's ID, title, start and end times, and a textual description.

**Bids:**

- Bids represent offers made by users on specific auctions.
- Each bid is associated with a unique bid ID and includes details such as the auction ID it pertains to, the bidder's ID, bid price, and the timestamp of when the bid was placed.

**Articles:**

- Articles are items or assets that are part of auctions.

- They are characterized by a unique article ID, a title, the auctioneer's ID, the auction they belong to, the number of units available, and a textual description.

**Collections:**

- Collections represent groupings of bids that share common characteristics.
- Each collection has a unique collection ID, references a bid, and specifies the number of units within the collection.

This schema also outlines the relationships between these entities, serving to connect and organize data effectively:

- Users can create auctions and make bids in auctions.
- Bids are associated with both auctions and users.
- Articles could be part of auctions and created by users.
- Collections represent a costume subsets (articles) from an auction original subset.

These tables and their relationships form the backbone of the Combinatorial Auction System's database, enabling the storage and retrieval of essential data for the webapplication.

## 3.2 Database Implementation

### 3.2.1 Use Case Diagram

The Use Case diagram illustrates the functional requirements and interactions between actors (users) and the system. It presents a high-level view of the system's functionality, showcasing the different use cases and how they relate to the actors.

Here is the Use Case diagram for the combinatorial auction webapp:

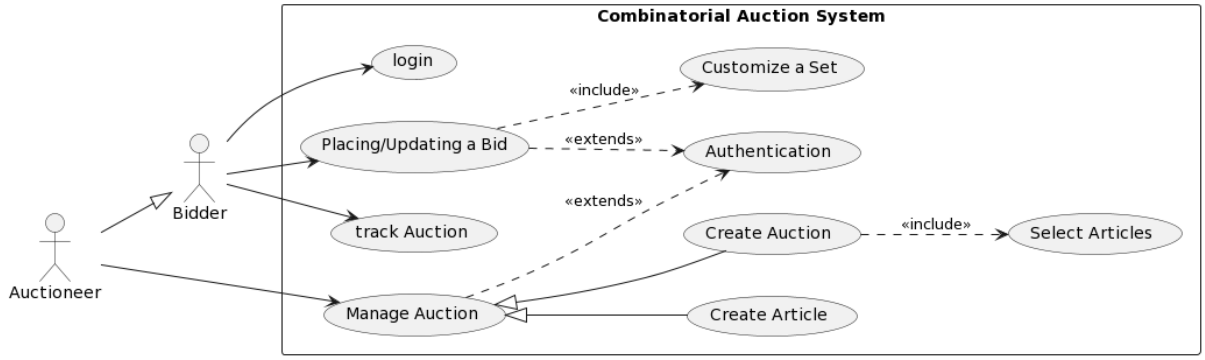



Figure 3.2: Use Case Diagram of the Combinatorial Auction webapp

The diagram provided is a Use Case diagram representing the interactions and relationships between actors and functionalities in the Combinatorial Auction System. Here is a description of the diagram:

**Actors:**

- Bidder: A user who participates in auctions by placing bids and customizing sets of items.
- Auctioneer: A user who manages auctions, creates auctions, and selects articles for auctions.

**Combinatorial Auction System:**

- Login: Both bidders and auctioneers can log in to the system using this use case.

**For Bidders:**

- Placing/Updating a Bid: Bidders can place bids on items and update their existing bids.
  - Customize a Set: This is an included use case within "Placing/Updating a Bid." Bidders can customize sets of items when placing or updating their bids.
- Track Auction: Bidders can track auctions to monitor the progress and status of auctions they are interested in.

**For Auctioneers:**

- Create Auction: Auctioneers can create new auctions.
  - Select Articles: This is an included use case within "Create Auction." Auctioneers can select articles to include in the newly created auction.
- Manage Auction: Auctioneers can manage existing auctions, which includes tasks such as updating auction details.
- Create Article: Auctioneers can create new articles to be used in auctions.

**Authentication:**

- Both bidders and auctioneers must be authenticated when performing actions such as placing bids, creating auctions, or managing auctions.

**General Relationships:**

- Bidders can log in to the system.
- Auctioneers can manage auctions.
- When creating an auction, auctioneers select articles to include.
- Bidders can track auctions.
- Managing auctions may include creating articles.
- Placing/Updating a Bid may require authentication.
- Managing Auctions may require authentication.
- Both actors, Auctioneers, and Bidders interact with the system.

This Use Case diagram provides an overview of the functional requirements and interactions within the Combinatorial Auction System, showcasing the roles of different actors and their interactions with various functionalities. It serves as a blueprint for the development and implementation of the system.

### 3.2.2 Sequence Diagrams

#### Create/Update Bid Sequence diagram

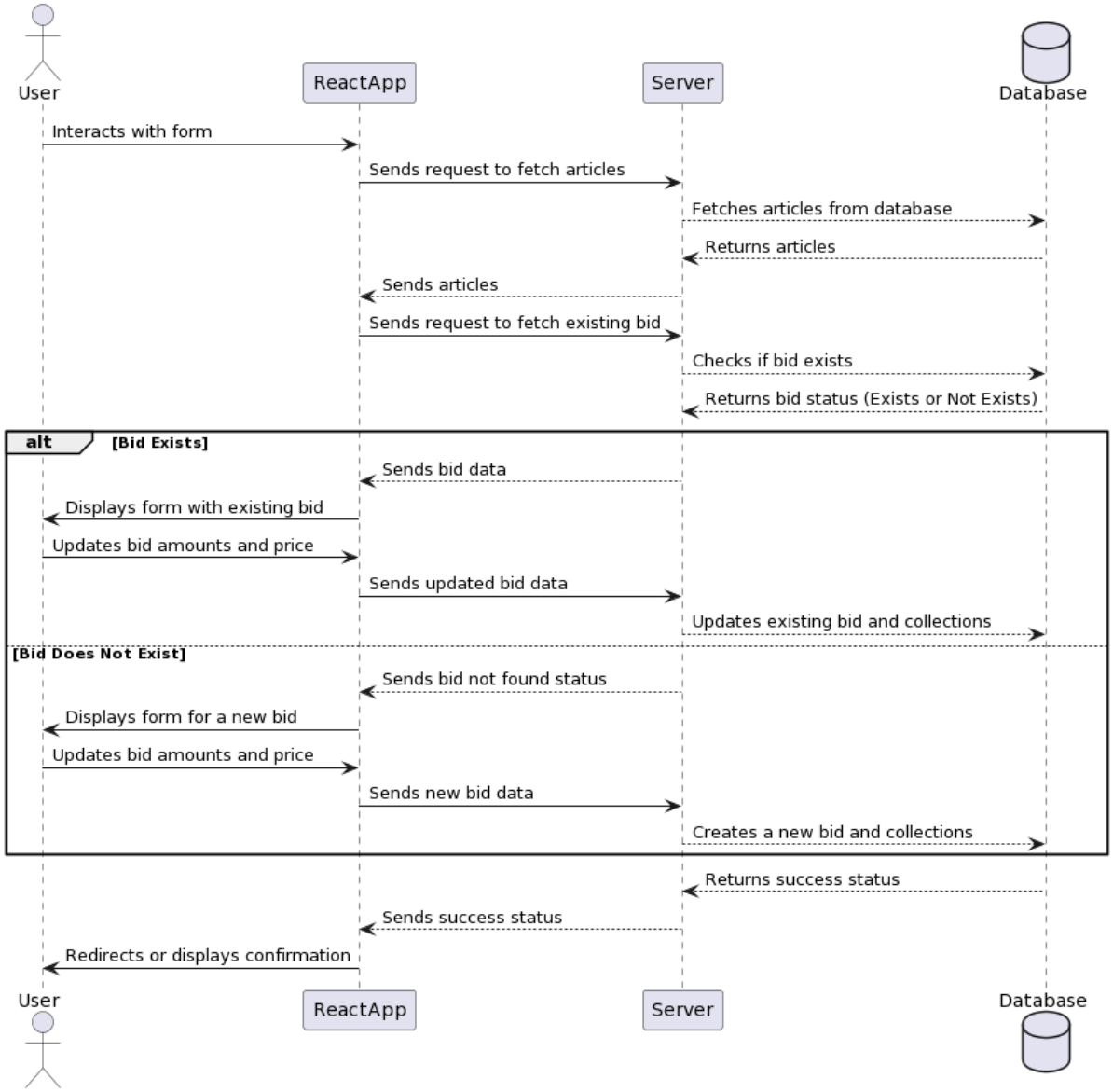



Figure 3.3: Sequence diagram of DB interaction with the webapp through an api (updating/creating a bid)

**Actor**: User

**Participants**:

- ReactApp: The React application serving as the user interface.
- Server: The backend server handling requests and data processing.
- Database: The database storing bid and article data.

1. **Actors and Participants**:
   - **User**: The user interacts with the system by providing input and receiving feedback.

- **ReactApp**: Represents the React application, which serves as the user interface and communicates with the server.
- **Server**: The backend server that handles HTTP requests, processes data, and interacts with the database.
- **Database**: The database where bid and article data are stored.

2. **User Interaction**:
   - The user interacts with a form, presumably through the React application, to update a bid.

3. **Fetching Articles**:
   - The ReactApp sends a request to the Server to fetch articles related to the bid.
   - The Server queries the Database to retrieve articles associated with the bid.
   - The Database returns the article data to the Server.
   - The Server sends the retrieved articles to the ReactApp.

4. **Checking Bid Existence**:
   - The ReactApp sends a request to the Server to check if a bid already exists.
   - The Server queries the Database to determine if the bid exists.
   - The Database returns the status of the bid (Exists or Not Exists) to the Server.

5. **Bid Exists (Alternative Path)**:
   - If the bid exists, the Server sends the bid data to the ReactApp.
   - The ReactApp displays the form with the existing bid data to the user.
   - The user updates the bid amounts and price in the form.
   - The ReactApp sends the updated bid data back to the Server.
   - The Server updates the existing bid and associated collections in the Database.

6. **Bid Does Not Exist (Alternative Path)**:
   - If the bid does not exist, the Server notifies the ReactApp that the bid was not found.
   - The ReactApp displays the form for creating a new bid.
   - The user updates the bid amounts and price in the form.
   - The ReactApp sends the new bid data to the Server.
   - The Server creates a new bid and associated collections in the Database.

7. **Database Update**:
   - The Database updates or creates the bid and collections as necessary.
   - The Database returns a success status to the Server.
   - The Server sends a success status to the ReactApp.

8. **User Feedback**:
   - The ReactApp provides feedback to the user, which may involve redirection or displaying a confirmation message.

**Create Auction Sequence diagram**

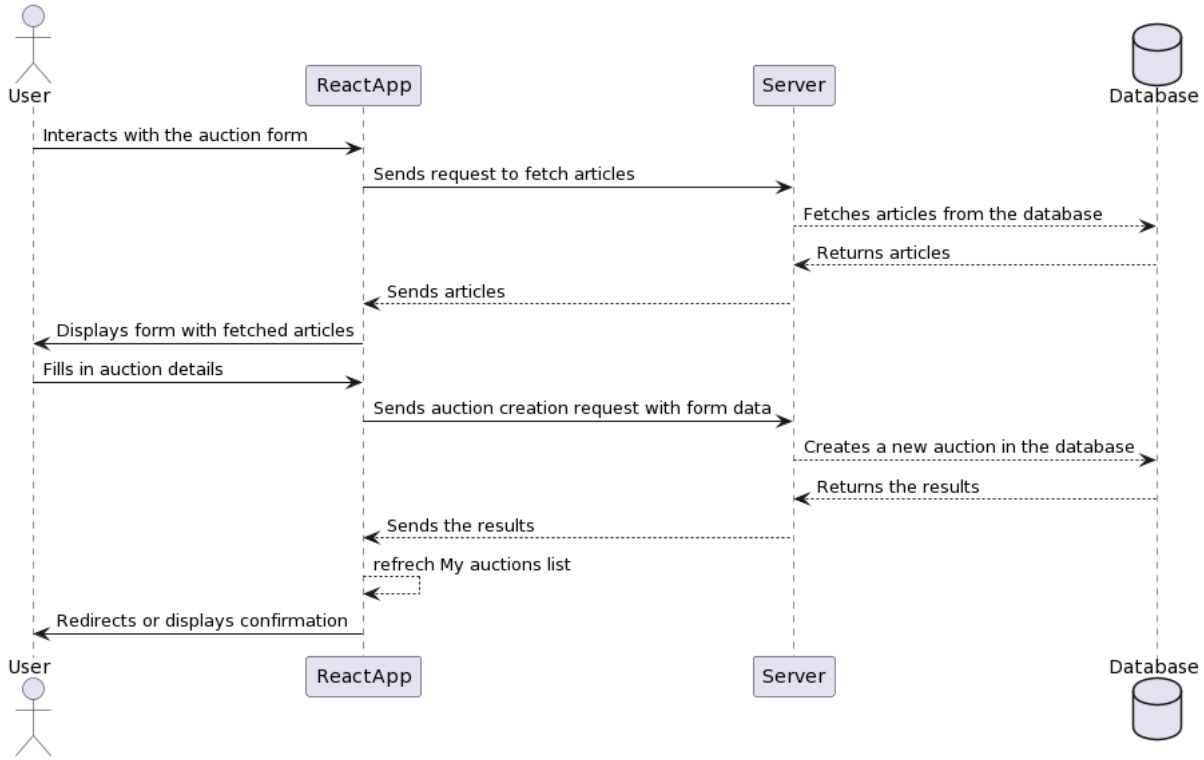



Figure 3.4: Sequence diagram of DB interaction with the webapp through an api (creating an auction)

**Actor:**

**User:** The individual interacting with the system.

**Participants:**

**ReactApp:** Represents the frontend of the application, typically built with React.js.

**Server:** Represents the backend server, which handles HTTP requests and interacts with the database.

**Database:** Refers to the database where auction data is stored.

**Sequence Description:**

1. **User Interaction:**
   The process starts with the user interacting with the auction form provided by the React application.

2. **Fetching Articles:**
   The ReactApp sends a request to the Server to fetch articles that might be associated with the auction. These articles could be items the user wants to auction.

3. **Fetching Articles from the Database:**
   The Server, upon receiving the request, interacts with the Database to fetch the relevant articles. These articles are retrieved from the database.

4. **Sending Articles to ReactApp:**
   The Server sends the fetched articles to the ReactApp.

5. **Displaying Articles in the Form:**
   The ReactApp displays the auction form to the user, including the fetched articles. This allows the user to choose items from the articles as part of the auction.

6. **User Filling Auction Details:**
   The User interacts with the form, filling in details for the auction, including choosing items from the fetched articles and specifying auction-specific information.

7. **Sending Auction Creation Request:**
   Once the user has completed the form, the ReactApp sends a request to the Server to create a new auction. This request includes all the form data.

8. **Creating a New Auction in the Database:**
   The Server, upon receiving the auction creation request, interacts with the Database to create a new auction entry. This involves storing all the provided details in the database.

9. **Database Processing:**
   The Database processes the request and creates the new auction. It may perform various checks and validations based on the provided data.

10. **Server Receives Results:**
    The Server receives the results of the database operation, which include information about whether the auction creation was successful or if any errors occurred during the process.

11. **Sending Results to ReactApp:**
    The Server communicates the results back to the ReactApp.

12. **Refreshing My Auctions List:**
    The ReactApp, upon receiving the results, may update the list of auctions, possibly by refreshing the list of auctions in the user's dashboard.

13. **User Feedback:**
    Finally, the User receives feedback from the ReactApp, which might include a redirection to a relevant page or a confirmation message indicating the success or failure of the auction creation process.

## 3.3 Dynamic Winner Determination Problem (WDP) Model

The Winner Determination Problem (WDP) in combinatorial auctions (CAP) is a crucial area where buyers bid for sets of items rather than individual items. However, many existing CAP models in the literature are static and do not allow bidders to update their offers, which does not reflect the reality of auctions where competition continues as long as time remains[4].

This part focuses on creating a dynamic formulation to replicate the English auction process. In an English combinatorial auction, a certain number of items in limited quantities are offered for sale to potential buyers. Buyers have a limited period to update

their bids. Each bid is a combination of items, formulated as a vector of dimension m+1, containing m quantities of items desired by the buyer and the price offered for that bid. An algorithm developed based on the bids of the buyers lists temporary winners at each moment, giving bidders the opportunity to respond accordingly.

The exercise period, denoted as T, is defined as the time interval [0; T], associated with the auction during which bidders can submit or update their offers. Temporary bids can be modified during this period[4].

### 3.3.1 Modeling the Winner Determination Problem (WDP)

The Winner Determination Problem (WDP) in combinatorial auctions is defined as a set of items being offered for sale to numerous buyers. Each buyer wishes to purchase a subset of items for which they provide an estimate. However, conflicts can arise among buyers due to intersections between these subsets.

The seller's objective is to maximize the total profit from the sale, which involves solving an NP-hard combinatorial optimization problem. We focus on a monobjective formulation of WDP that aims to maximize the total auction profit[4].

### 3.3.2 Approach and Algorithm

We develop a dynamic monobjective model to replicate the English auction process, where bidders have the opportunity to update their offers during the exercise period. To efficiently solve this complex problem, we utilize an algorithm referred to as D-CAP.

## 3.4 Resolution Organigram

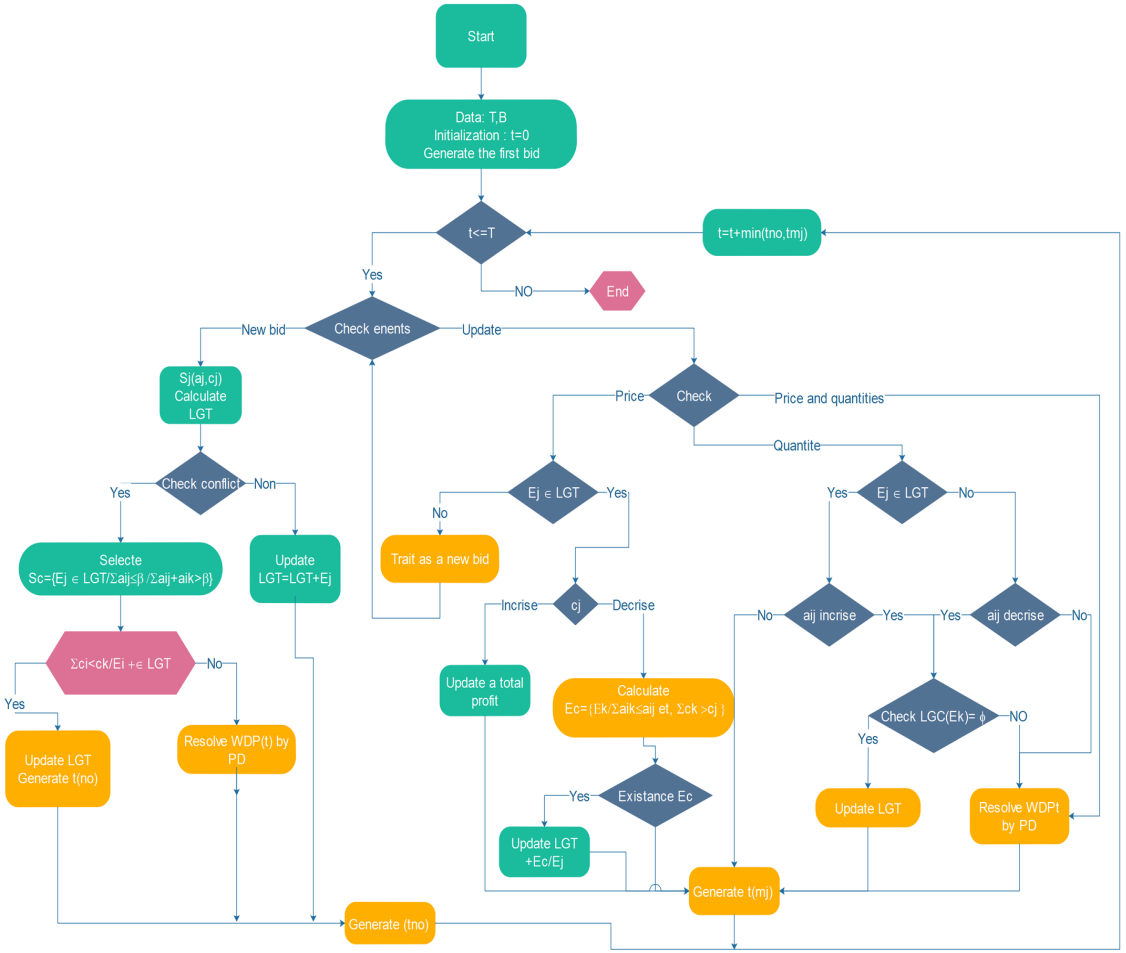



Figure 3.5: Winning Determination Problem Diagram[4]

The Winning Determination Problem diagram provides a visual representation of the algorithm or approach used to determine the winning bids and allocate items to the bidders. This diagram helps in understanding the logic behind the winning determination process and assists in the implementation of the algorithm in the webapp.

## 3.5 Combinatorial Resolution Methodology

In the context of dynamic combinatorial auctions, resolving the Winner Determination Problem (WDP) involves managing the constantly changing bids and determining the temporary list of winning bidders at each moment during the exercise period $T$. This methodology outlines the steps and algorithms for effectively addressing these challenges.

### 3.5.1 Handling New Bids

When a new bid, denoted as $E_j = (a_{ij}(t), c_j(t))$, is submitted, the following steps are taken:

1. Calculate the Temporary List of Winning Bidders in Conflict (LGC) with $E_j$.
2. If $E_j$'s offer does not conflict with LGC, it becomes a temporary winner.
3. Update the Temporary Winner List (TWT) by adding $E_j$.
4. In the case of a conflict, sort LGC in ascending order of prices ($c_i$).
5. Select the minimum subset of bidders (EC) from LGC such that their removal resolves conflicts with $E_j$.
6. Calculate the sum of their prices ($s_c$).
7. If $s_c$ is less than $E_j$'s price ($c_j$), update TWT by removing EC from it.
8. If not, the problem is resolved to find the best solution.

### 3.5.2 Updating Existing Bids

Bidders may update their existing bids, either by changing the price ($c_j$) or both the price and quantities ($a_{ij}$). Different cases arise for updates:

1. If only the price ($c_j$) is changed, and the bidder is already a temporary winner, the total temporary profit is updated accordingly.
2. If the price ($c_j$) decreases and there is a maximum set of non-winning bidders (Ek) whose combined bid quantities ($a_{ij}$) are less than or equal to $E_j$'s quantities for all items and their combined prices ($c_i$) exceed $E_j$'s price, then update TWT by removing Ek.
3. If $E_j$ is not in the temporary winner list, treat it as a new bid according to the steps mentioned earlier.
4. When both price ($c_j$) and quantities ($a_{ij}$) are updated, the situation depends on whether $E_j$ is a winner or not.

5. If $E_j$ is a winner, calculate LGC, and if it's not empty, resolve the problem using resolution approuch.

6. If $E_j$ is not a winner, treat it as a new bid.

### 3.5.3 Resolving Conflicts

Conflicts arise when two or more bids cannot be retained simultaneously because the sum of the quantities for at least one item exceeds the available quantity. Conflicting bids are handled by considering their prices ($c_i$).

1. If the number of conflicting bids in the Temporary Winner List (TWL) is equal to the total number of temporary winners (TWL), compare their prices ($c_i$). If $E_j$ has a lower price ($c_j$), it replaces the other conflicting bids in TWL.

2. If the number of conflicting bids ($|CWL|$) is less than the total number of temporary winners ($|TWL|$), and the sum of prices ($c_i$) for the conflicting bids is greater than $E_j$'s price, a subset of bidders is selected to update TWT by removing them.

3. If neither of these conditions apply, the original TWL remains unchanged.

## Conclusion

In conclusion, this chapter has set up the foundation for our innovative Combinatorial Auction System, which is ready to make waves in the world of online auctions. We've created a strong database model using a clear diagram to organize our data. Additionally, we've introduced a dynamic Winner Determination Problem (WDP) model that represents the future of auctions. It's like an online auction where bids can be updated in real time, making it adaptable to changing market conditions. We've also developed smart ways to handle conflicts and solve problems quickly.

These elements will be the core of our advanced system, providing organized data management and real-time WDP resolution. With these foundations, we're all set to build a responsive and future-proof auction platform that will change the way online auctions work.

# 4

# Results, Analysis, and Recommendations

## Introduction

In this chapter, we present the results, analysis, and recommendations for our web application. We'll explore its core functionalities, database design, winner determination algorithm, user experience, performance, scalability, and security. This comprehensive examination aims to provide insights into the current state of our project and offer suggestions for future enhancements, ensuring the platform's continued effectiveness and security in facilitating online auctions.

## 4.1 Database Implementation Software

In the context of my project, we utilized two essential tools for working with MySQL databases: HeidiSQL and MySQL Workbench. These software applications played a significant role in facilitating the development and implementation of a web application auction with a winning determination problem. we would like to provide a more detailed explanation of how each tool contributed to the project's success.

### 4.1.1 Database softwares

we utilized both HeidiSQL, an open-source database management tool, and MySQL Workbench, a comprehensive database development and administration tool, to efficiently manage and design the MySQL database for the web application auction. HeidiSQL offered a user-friendly interface, simplifying interactions with the MySQL server, enabling schema design, query execution, and data transfer.

MySQL Workbench provided visual modeling, intuitive schema design, and powerful SQL query support, enhancing my understanding of the database structure and streamlining development. These combined tools significantly contributed to the project's success by simplifying database management and enhancing the development workflow[13, 11].

### 4.1.2 Web App Database Interaction

The web application interacts with the MySQL database through server.js APIs. These APIs handle various tasks, including user registration and login, data retrieval, bid creation and updates, and more. The server.js script sets up an Express.js server and connects to the MySQL database. It also utilizes packages like bcrypt for password hashing, express for routing, and axios for making HTTP requests from the frontend. For instance, when a user registers or logs in, the web app sends POST requests to the appropriate API endpoints, such as '/api/register' and '/api/login'. These endpoints handle user authentication and database operations, ensuring secure user management.

To fetch data from the database, the web app uses GET requests to endpoints like '/api/auctions', '/api/data/users/:userid', and '/api/data/articles/:auctionId'. These endpoints retrieve auction information, user details, and articles, respectively. Additionally, the server.js script includes endpoints for creating and updating bids, managing collections, and fetching data about bidders and their collections. These APIs facilitate real-time bidding functionality within the web app.

In React code examples, you can see how the web app makes HTTP requests using the fetch() method and the axios library to communicate with the server.js APIs. These requests enable seamless interaction between the web app's frontend and the MySQL database, enabling users to perform various actions, including creating articles, viewing collections, and participating in auctions.

## 4.2 Programmatic Winner Determination Problem (WDP) Solver Software

### 4.2.1 Using Python with Spyder for WDP

In the Programmatic Winner Determination Problem (WDP) Solver Software section of the project, we chose Python, along with the Spyder integrated development environment (IDE), as the primary tool for solving the WDP. Python's versatility, extensive libraries, and ease of use made it an excellent choice for this task.

**Python Advantages:**

1. **Rich Ecosystem:** Python offers a vast ecosystem of libraries and tools for data manipulation, optimization, and mathematical modeling. This makes it well-suited for solving complex optimization problems like the Winner Determination Problem.

2. **Spyder IDE:** Spyder provides a user-friendly environment for developing and running Python code. Its interactive features and debugging capabilities make it ideal for working on mathematical models and algorithms.

3. **Ease of Integration:** Python's simplicity and versatility allowed for easy integration with other components of the project, such as the React web application and the MySQL database.

### 4.2.2 How the Python WDP Solver Works

The Python-based WDP solver works as follows:

1. **Flask Server:** The core of the solver is a Flask server, which provides a RESTful API for interacting with the solver. The server listens for incoming requests from the React web application.

2. **API Endpoint for Fetching Winners:** The `/api/winners/<int:auction_id>` endpoint is responsible for solving the WDP for a specific auction. It uses auction-specific data and bidder information retrieved from the MySQL database.

3. **Fetching Bidders and Collections:** The solver starts by making requests to the MySQL database using endpoints like `/api/bidders/<auction_id>` and `/api/collections/<user_id>/<auction_id>`. It fetches data about bidders and their collections for the given auction.

4. **Optimization Algorithm:** The solver then utilizes an optimization algorithm to determine the winners of the auction. It considers factors such as bidder prices, collection units, and stock limits to select the winning bidders. The algorithm handles conflicts, ensuring that stock limits are not exceeded.

5. **JSON Response:** Finally, the solver prepares the list of winners and their relevant information, such as names, prices, and timestamps, in JSON format. This data is returned as a response to the React web application.

### 4.2.3 Interaction with React Web App and MySQL Database

The Python WDP solver acts as a bridge between the React web application and the MySQL database:

- **React Web App:** The web application initiates the WDP solving process by making HTTP requests to the Flask server's API endpoint. It sends the `auction_id` as a parameter to specify the auction for which it wants to determine the winners.

- **MySQL Database:** The solver fetches necessary data from the MySQL database, including bidder details and collections, which are crucial for solving the WDP. It leverages the MySQL data to perform the optimization and select the winners.

By employing Python and Spyder within a Flask server, we were able to create a robust and efficient solver for the Winner Determination Problem, seamlessly integrating it with the React web application and the MySQL database. This approach ensures the accurate and reliable determination of winners in the combinatorial auctions.

## 4.3 Implementation and Results

### 4.3.1 Database handeling for testing

#### Testing and Inserting Queries

Before diving into the specifics of the selected database software, it's essential to highlight a fundamental aspect of our project—testing. To ensure the functionality and reliability

of our web application, we conducted extensive testing, including the insertion of various types of data into the database.

One illustrative example of this testing process is shown below:

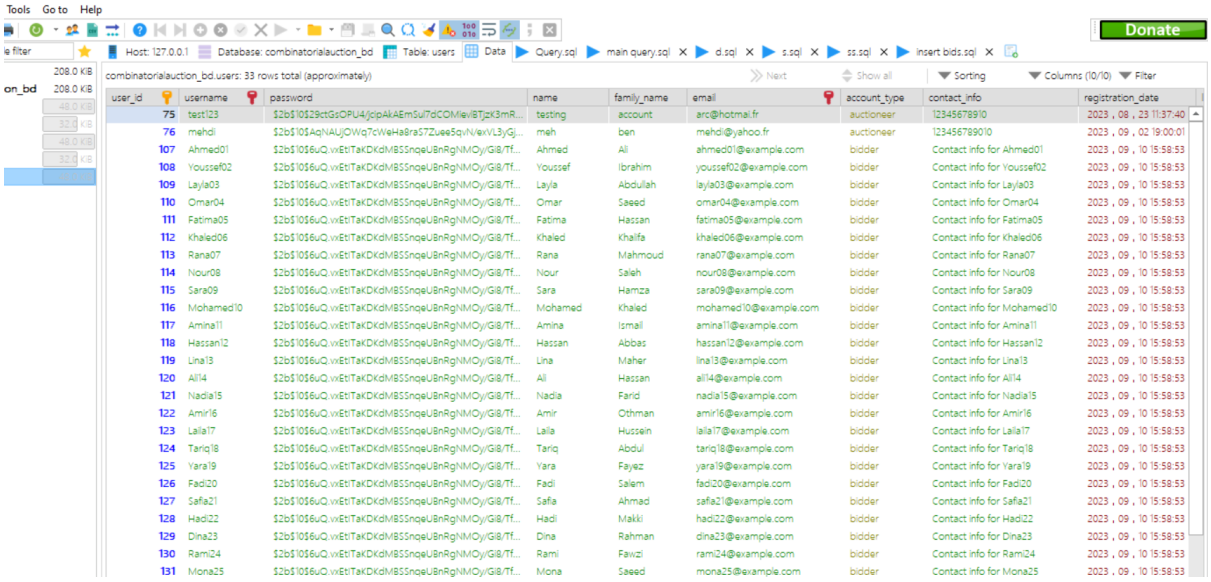
```
-- Create bids and collections
INSERT INTO bids (auction_id, bidder_id, price, put_time)
SELECT
    36 AS auction_id,
    user_id AS bidder_id,
    ROUND(RAND() * 100 + 1, 2) AS price,
    NOW() AS put_time
FROM users
WHERE user_id BETWEEN 130 AND 135;

-- Retrieve the saved bid_ids
SET @first_bid_id = LAST_INSERT_ID();
SET @last_bid_id = @first_bid_id + 5; -- Assuming 6 bids were inserted

-- Insert collections for each bid
INSERT INTO collections (collection_id, bid_id, units)
SELECT
    a.article_id AS collection_id,
    b.bid_id AS bid_id,
    ROUND(RAND() * a.units) AS units
FROM articles a
JOIN bids b ON b.auction_id = a.auction_id
WHERE a.auction_id = 36
AND b.bid_id BETWEEN @first_bid_id AND @last_bid_id
AND ROUND(RAND() * a.units) <= a.units;
```

In the above SQL query, we insert a bid into the database, specifying the auction, bidder, article, bid price, and timestamp. This operation simulates a user placing a bid on a specific item during an auction.

Similar insertion queries were executed to populate the database with sample data, including bidders as users, auction details, and articles for sale. This extensive testing helped us validate the functionality of our web application and ensured that the database could handle the dynamic nature of online auctions effectively.



| user_id | username | password | name | family_name | email | account_type | contact_info | registration_date |
|---|---|---|---|---|---|---|---|---|
| 75 | test123 | $2b$10$29ctGsOPU4/jcipAkAEmSuI7dCOMievBTjzK3mR... | testing | account | arc@hotmail.fr | auctioneer | 12345678910 | 2023 , 08 , 23 11:37:40 |
| 76 | mehdi | $2b$10$AqNAUjOWq7cWeHa8ra57Zuee5qvN/exVL3yGj... | meh | ben | mehdi@yahoo.fr | auctioneer | 123456789010 | 2023 , 09 , 02 19:00:01 |
| 107 | Ahmed01 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Ahmed | Ali | ahmed01@example.com | bidder | Contact info for Ahmed01 | 2023 , 09 , 10 15:58:53 |
| 108 | Youssef02 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Youssef | Ibrahim | youssef02@example.com | bidder | Contact info for Youssef02 | 2023 , 09 , 10 15:58:53 |
| 109 | Layla03 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Layla | Abdullah | layla03@example.com | bidder | Contact info for Layla03 | 2023 , 09 , 10 15:58:53 |
| 110 | Omar04 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Omar | Saeed | omar04@example.com | bidder | Contact info for Omar04 | 2023 , 09 , 10 15:58:53 |
| 111 | Fatima05 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Fatima | Hassan | fatima05@example.com | bidder | Contact info for Fatima05 | 2023 , 09 , 10 15:58:53 |
| 112 | Khaled06 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Khaled | Khalifa | khaled06@example.com | bidder | Contact info for Khaled06 | 2023 , 09 , 10 15:58:53 |
| 113 | Rana07 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Rana | Mahmoud | rana07@example.com | bidder | Contact info for Rana07 | 2023 , 09 , 10 15:58:53 |
| 114 | Nour08 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Nour | Saleh | nour08@example.com | bidder | Contact info for Nour08 | 2023 , 09 , 10 15:58:53 |
| 115 | Sara09 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Sara | Hamza | sara09@example.com | bidder | Contact info for Sara09 | 2023 , 09 , 10 15:58:53 |
| 116 | Mohamed10 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Mohamed | Khaled | mohamed10@example.com | bidder | Contact info for Mohamed10 | 2023 , 09 , 10 15:58:53 |
| 117 | Amina11 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Amina | Ismail | amina11@example.com | bidder | Contact info for Amina11 | 2023 , 09 , 10 15:58:53 |
| 118 | Hassan12 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Hassan | Abbas | hassan12@example.com | bidder | Contact info for Hassan12 | 2023 , 09 , 10 15:58:53 |
| 119 | Lina13 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Lina | Maher | lina13@example.com | bidder | Contact info for Lina13 | 2023 , 09 , 10 15:58:53 |
| 120 | Ali14 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Ali | Hassan | ali14@example.com | bidder | Contact info for Ali14 | 2023 , 09 , 10 15:58:53 |
| 121 | Nadia15 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Nadia | Farid | nadia15@example.com | bidder | Contact info for Nadia15 | 2023 , 09 , 10 15:58:53 |
| 122 | Amir16 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Amir | Othman | amir16@example.com | bidder | Contact info for Amir16 | 2023 , 09 , 10 15:58:53 |
| 123 | Laila17 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Laila | Hussein | laila17@example.com | bidder | Contact info for Laila17 | 2023 , 09 , 10 15:58:53 |
| 124 | Tariq18 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Tariq | Abdul | tariq18@example.com | bidder | Contact info for Tariq18 | 2023 , 09 , 10 15:58:53 |
| 125 | Yara19 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Yara | Fayez | yara19@example.com | bidder | Contact info for Yara19 | 2023 , 09 , 10 15:58:53 |
| 126 | Fadi20 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Fadi | Salem | fadi20@example.com | bidder | Contact info for Fadi20 | 2023 , 09 , 10 15:58:53 |
| 127 | Safa21 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Safa | Ahmad | safa21@example.com | bidder | Contact info for Safa21 | 2023 , 09 , 10 15:58:53 |
| 128 | Hadi22 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Hadi | Makki | hadi22@example.com | bidder | Contact info for Hadi22 | 2023 , 09 , 10 15:58:53 |
| 129 | Dina23 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Dina | Rahman | dina23@example.com | bidder | Contact info for Dina23 | 2023 , 09 , 10 15:58:53 |
| 130 | Rami24 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Rami | Fawzi | rami24@example.com | bidder | Contact info for Rami24 | 2023 , 09 , 10 15:58:53 |
| 131 | Mona25 | $2b$10$6uQ.vxEtiTaKDKdMBSSnqeUBnRgNMOy/GiB/Tf... | Mona | Saeed | mona25@example.com | bidder | Contact info for Mona25 | 2023 , 09 , 10 15:58:53 |

Figure 4.1: The data set made for testing

The ability to conduct these tests seamlessly and observe how the database handled various data interactions was a key criterion in our database software selection process. We needed a solution that offered both reliability and performance to support our web application's real-time bidding functionality.

### 4.3.2 Auction Details Page

The "Auction Details Page" is a central component of the web application, providing users with a comprehensive view of the ongoing auction. This page combines various elements and functionalities to create an engaging and informative experience for users.

#### Auction Overview

At the core of the "Auction Details Page" is the presentation of essential auction information, including the auction's title and a list of articles available for bidding. Users can access this page to gain insights into the current auction and participate in the bidding process.

#### Bidders List Container

The "Bidders List Container" is a critical component that serves several key functions:

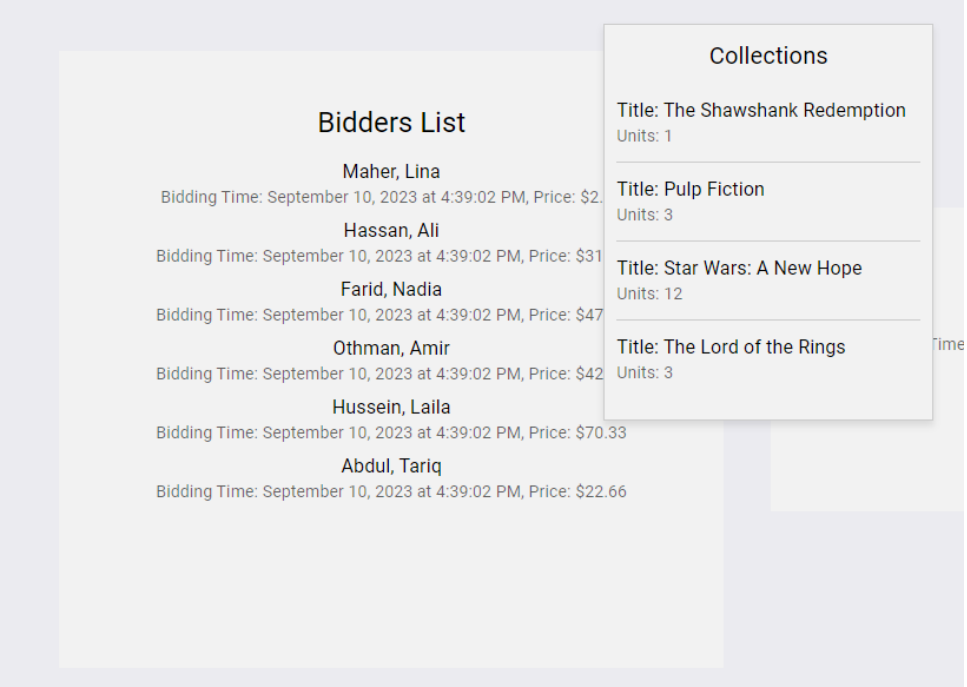
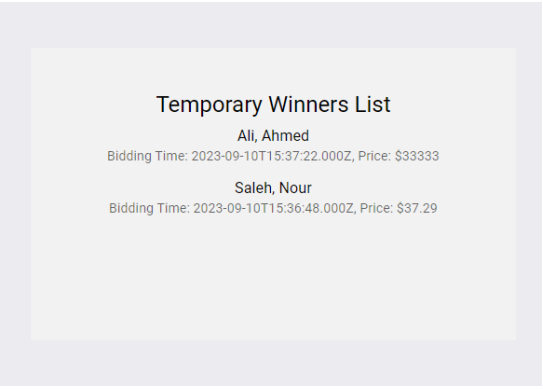



Figure 4.2: Bidder's list

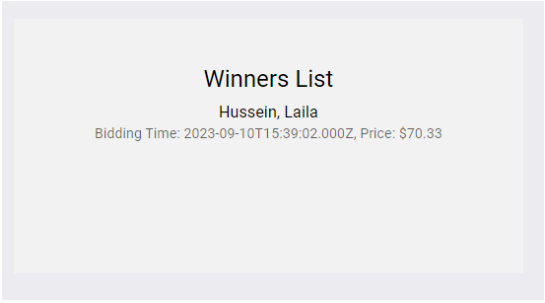
- **Bidders List Display:** It presents a list of all participating bidders in the auction, enhancing transparency and competition awareness.
- **Collection Information:** When a user hovers over a bidder's name, a "Collections" section pops up, providing insight into the bidder's collections within the auction.
- **Real-time Updates:** The component fetches data at regular intervals, ensuring that the list of bidders and their collections is always up to date.

### Winners List Container

Complementing the "Bidders List Container," the "Winners List Container" offers valuable insights:




(a) Temporary winner's list



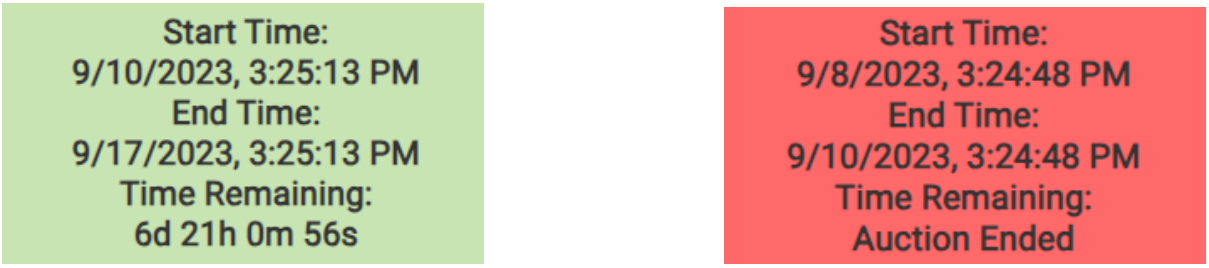

(b) Winner's list

Figure 4.3: Comparison of Temporary and Winner's Lists

- **Temporary Winners vs. Final Winners:** Depending on the auction's status (active or concluded), the component displays either the "Temporary Winners List" or the "Winners List," providing clarity about the current state of the auction.
- **Winner Information:** It lists the winners of the auction, including their names, bidding times, and prices. Users can quickly see who has secured items.
- **Real-time Updates:** Similar to the "Bidders List Container," this component fetches data at regular intervals, ensuring that the list of winners is continuously updated during the auction.

### Auction Timer

The "Auction Timer" component adds excitement and urgency to the "Auction Details Page":




(a) Auction timer (Green)

(b) Auction timer (Red)

Figure 4.4: Auction timers

- **Countdown Timer:** It displays a countdown timer, indicating the time remaining until the auction's end.
- **Start and End Times:** Users can see the exact start and end times of the auction, ensuring transparency.
- **Real-time Updates:** The timer updates in real-time, providing users with an accurate view of the time remaining in the auction.

These components collectively contribute to an engaging and informative "Auction Details Page." Users can access vital information, place bids, view other participants, and stay updated on the auction's progress, all in one place. This enhances the overall auction experience and encourages active participation.

### Article set Component

The "Article set Component" complements the page by displaying the articles available in the auction:

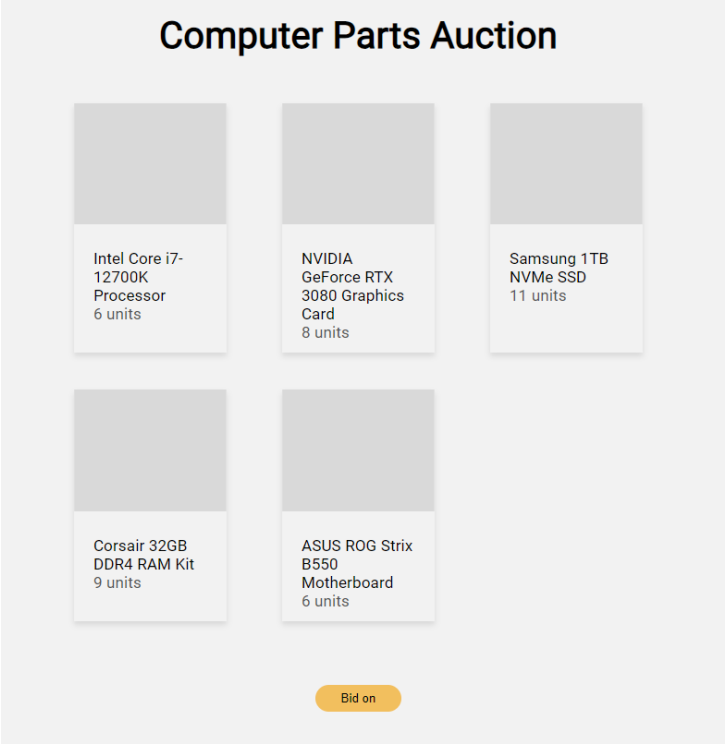



Figure 4.5: Articles in an Auction page

- **Article Overview:** It presents a grid of articles available for bidding within the auction. Each article is represented with its title and details.
- **Dynamic Data Loading:** The component dynamically loads data from the server, ensuring that users can see the latest information about the articles within the auction.

### Bidding Form

The "Bidding Form" plays a crucial role in facilitating the bidding process:

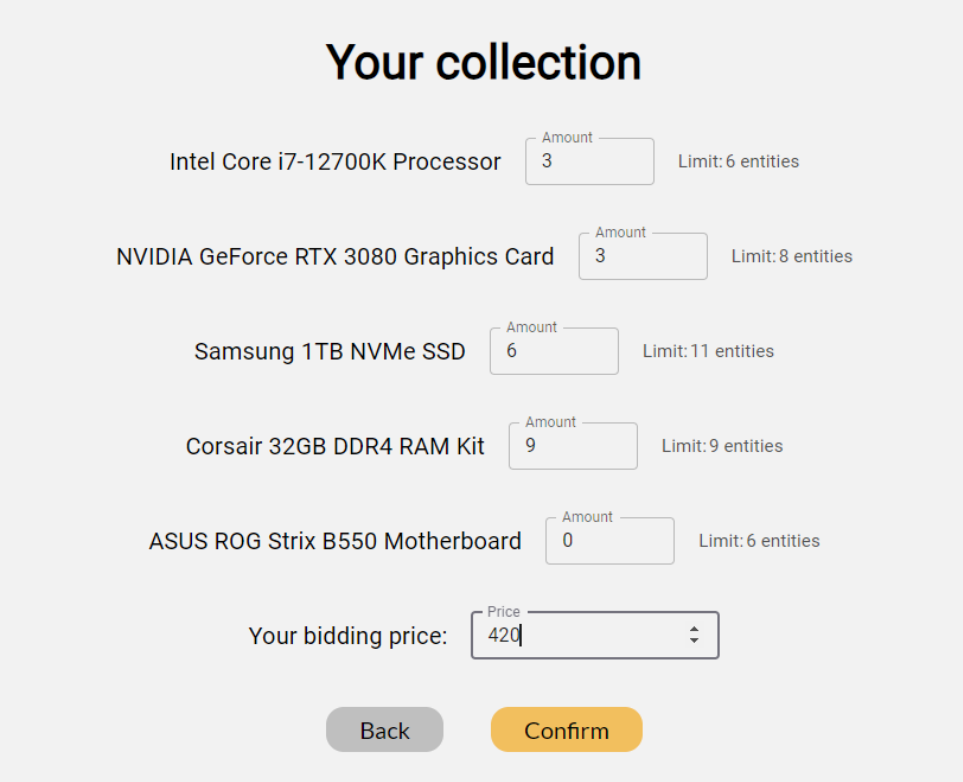



Figure 4.6: Bidding Form

- **User Interaction:** It allows users to place bids on articles within the auction. Users can specify the number of articles they want to bid on and the price they are willing to pay.
- **Data Fetching and Submission:** The component fetches data about articles and existing bids, populating the bidding form with relevant information. After submission, it sends bid data to the server, updating the user's bid status.
- **User-friendly Design:** The bidding form is designed for ease of use, with clear input fields and buttons for confirmation.

### 4.3.3 My Auctions, Articles, and Bids

This subsection provides an overview of three essential pages within the web application: "My Auctions," "My Articles," and "My Bids." Each of these pages serves a distinct purpose in helping users manage their activities and interactions within the platform.

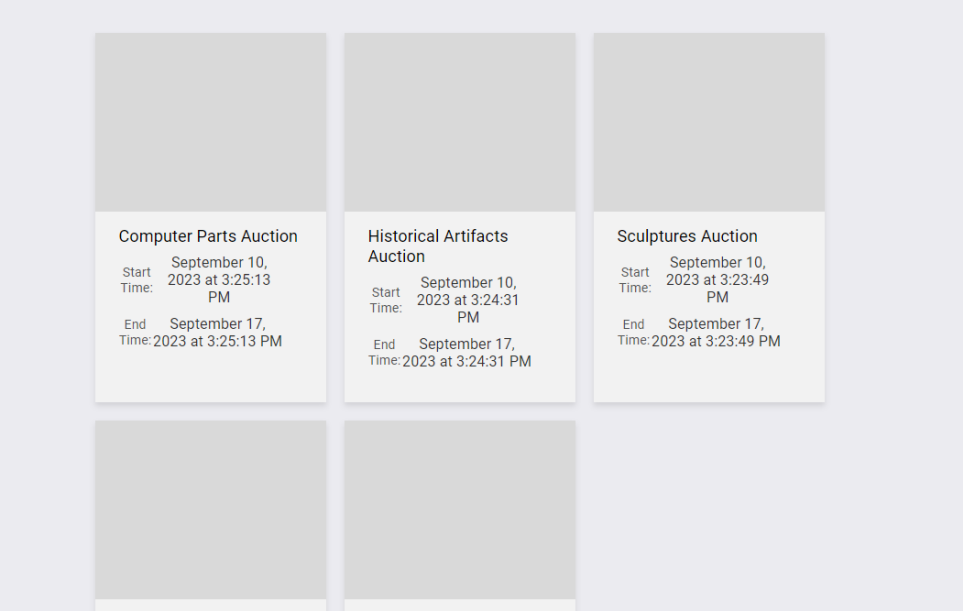



Figure 4.7: Example of how the instance show in a page

### My Auctions Page

The "My Auctions Page" is a dedicated space for users to oversee and manage auctions they have initiated as auctioneers. Key features and functionalities of this page include:

- **Auction Overview:** Users can view a list of auctions they have created, including details such as titles, descriptions, and current status.
- **Auction Creation:** The page offers an intuitive interface for users to initiate new auctions, setting the stage for new bidding opportunities.
- **Real-time Updates:** Auction information is updated in real-time, ensuring users have access to the latest data.

### My Articles Page

The "My Articles Page" empowers users to manage articles they have posted for auction. This page provides:

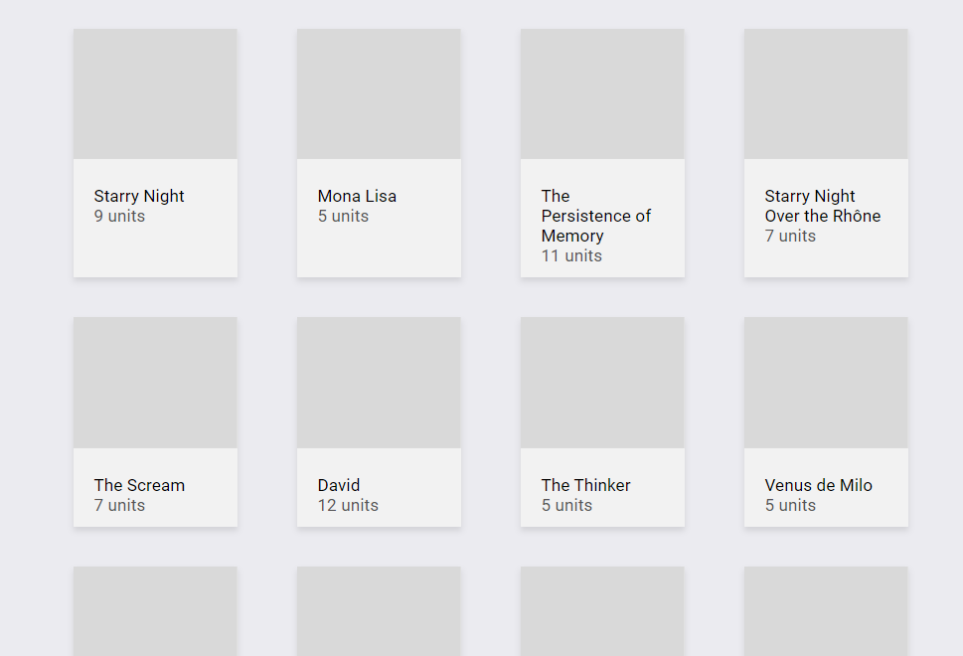



Figure 4.8: Example of how the instance show in a page

- **Article Listing:** Users can access a comprehensive list of articles they have listed for auction, complete with titles, descriptions, and current status.
- **Article Management:** Users can easily edit, update, or remove articles from the platform, maintaining control over their listings.
- **Streamlined Posting:** The page includes a user-friendly interface for adding new articles to the platform, streamlining the listing process.

### My Bids Page

The "My Bids Page" serves as a centralized hub for users to monitor their bidding activities. Key functionalities include:

- **Bid Tracking:** Users can review a comprehensive list of their bids, providing insights into their bidding history.
- **Bid Details:** Detailed bid information is available, including bid amounts, timestamps, and the status of each bid.
- **Real-time Updates:** The page dynamically updates bid information, ensuring users have the latest data on their bids.

Together, these three pages empower users to take full control of their auction-related activities. Whether creating new auctions, managing articles, or tracking bids, users can efficiently navigate and engage with the platform.

## 4.4 Discussion and Recommendations

In this section, we delve into a comprehensive discussion of the key aspects of our web application, including its functionality, user experience, and performance. We also provide recommendations for improvements and enhancements.

### 4.4.1 Web Application Functionality

Our web application successfully achieves its primary objectives, including:

- **Auction Creation and Management:** Users can easily create and manage auctions, facilitating the process of selling items.
- **Article Management:** The "My Articles" page allows users to manage their articles efficiently, streamlining the listing and tracking of items.
- **Bidding System:** The bidding system enables users to place bids on auctions, creating a dynamic and competitive environment.

However, there are areas where we can further improve the functionality:

#### Recommendations for Functionality

- **Real-Time Updates:** Implement real-time updates to provide users with immediate notifications of bid changes and auction status.
- **Search and Filtering:** Enhance the search and filtering capabilities to help users find auctions and articles more efficiently.

### 4.4.2 Database Design

Our web application relies on a well-structured database to store and manage various entities, including articles, auctions, bids, collections, and user data. The database design follows best practices for data integrity and relationships.

#### Tables and Relationships

- **Users Table:** This table stores user information, including usernames, passwords, names, email addresses, account types, and registration dates. It establishes a relationship with other tables through foreign keys.
- **Auctions Table:** The auctions table contains data related to auctions, such as titles, auctioneer IDs, start times, end times, and descriptions. It is linked to the users table to associate auctions with auctioneers.
- **Articles Table:** In this table, information about articles is stored, including titles, auction IDs, available units, descriptions, auctioneer IDs, and images. It establishes a relationship with the auctions and users tables.
- **Bids Table:** The bids table records bid details, such as auction IDs, bidder IDs, bid prices, and timestamps. It is linked to both auctions and users, associating bids with specific auctions and bidders.

- **Collections Table:** This table represents collections associated with bids. It contains collection IDs, bid IDs, and units. It maintains a relationship with the bids table.

#### Recommendations for Database

Our current database structure is well-suited for the application's requirements. However, to ensure its optimal performance and scalability, we recommend the following:

- **Regular Backups:** Implement a robust backup strategy to protect against data loss.
- **Index Optimization:** Continuously monitor and optimize database indexes to enhance query performance.
- **Data Archiving:** Implement a data archiving mechanism to manage historical data efficiently.

### 4.4.3 Winner Determination Problem

One of the critical aspects of our combinatorial auction system is the determination of winners among the bidders. We have implemented an algorithm in Python to address this problem efficiently. Below, we discuss the approach used and provide recommendations for further improvements.

#### Algorithm Overview

Our winner determination algorithm takes into account both the price bid by each bidder and the availability of articles in their collections. The key steps of the algorithm are as follows:

1. **Fetching Bidder Data:** We start by retrieving bidder data, including user IDs and bidding prices, from the server based on the auction ID.
2. **Fetching Collections:** For each bidder, we fetch their collections, which include articles they have bid on and the quantities they bid for.
3. **Conflict Resolution:** To handle conflicts where a bidder's bid exceeds the available stock limit, we prioritize bidders based on their bidding prices and the total units bid in ascending order.
4. **Determining Winners:** We iterate through the sorted list of bidders and select winners by ensuring that their bids do not conflict with stock limits.
5. **Result Presentation:** Finally, we present the winners' data, including names, family names, bidding prices, and timestamps, as a JSON response.

**Recommendations for Improvement**

While our current algorithm effectively determines winners, we recognize the potential for enhancements and optimizations:

- **Efficiency:** Investigate methods to improve the algorithm's efficiency, especially for large-scale auctions, by considering more advanced data structures and algorithms.
- **Real-time Updates:** Implement real-time updates to provide instant feedback to users when new bids are placed or auctions end.
- **Scalability:** Ensure that the winner determination process remains scalable as the number of users and auctions grows.
- **User Interface:** Enhance the user interface to clearly display the winners and the articles they have won.

Our winner determination algorithm is a crucial component of our combinatorial auction system, and ongoing refinements will contribute to a smoother and more efficient bidding experience for our users.

### 4.4.4 User Experience

The user experience of our web application plays a pivotal role in its success. We have designed an intuitive and visually appealing interface, ensuring ease of use. Users have provided positive feedback regarding the following aspects:

- **Intuitive Navigation:** The layout and navigation are user-friendly, allowing users to explore and interact with ease.
- **Responsive Design:** The web application is responsive, adapting to various screen sizes and devices.

However, there are areas for improvement in enhancing the overall user experience:

**Recommendations for User Experience**

- **Mobile Optimization:** Further optimize the application for mobile devices to provide a seamless experience on smaller screens.
- **User Onboarding:** Implement an onboarding process or tutorial for new users to familiarize them with the platform's features.

### 4.4.5 Performance and Scalability

Ensuring optimal performance and scalability is vital for accommodating a growing user base. Our web application currently demonstrates:

- **Stable Performance:** The application performs well under typical loads, with fast response times.
- **Scalability Potential:** The architecture allows for potential scalability to handle increased traffic.

To further enhance performance and scalability:

**Recommendations for Performance and Scalability**

- **Load Testing:** Conduct load testing to identify potential bottlenecks and optimize database queries and server performance.
- **Caching Strategies:** Implement caching strategies to reduce database queries and enhance response times.

### 4.4.6 Security

Security is paramount in any web application. We have implemented basic security measures, including authentication and authorization.However, additional security measures should be considered:

**Recommendations for Security Enhancements**

- **Data Encryption:** Implement data encryption for sensitive user information to protect against data breaches.
- **Regular Security Audits:** Conduct regular security audits and vulnerability assessments to identify and address potential threats.

# Conclusion

Our web application offers an intuitive and responsive platform for online auctions, ensuring a user-friendly experience. The winner determination algorithm is effective, providing fairness and transparency.

To further enhance our project, we recommend real-time updates, advanced algorithms for efficiency, mobile optimization, and improved security measures. Load testing and caching strategies are vital for performance and scalability.

In conclusion, our web application brings combinatorial auctions to users, and with these enhancements, we aim to make it a robust and secure solution for online bidding. We're committed to continuous improvement to meet user needs in the dynamic world of online auctions.

# General conclusion

In conclusion, the practice of negotiating prices is a universally recognized aspect of human interaction. It serves as a means of communication and interaction between buyers and sellers, with the mutual goal of reaching agreeable price terms for goods and services. Auctions, as a specialized form of negotiation, involve participants presenting offers, typically in the form of estimated prices, in competition for a specific item or service. The ultimate victor in these auctions is the bidder who submits the highest bid.

Traditional auctions have been a popular method for negotiating goods for an extended period. However, they are not without their limitations, which can be effectively addressed through online combinatorial auctions. To optimize auction outcomes in the business world, various auction types have been employed. These include the English auction, where bidding prices increase until there are no more bids, the Dutch auction, where the initial price decreases until a buyer accepts, and the Vickrey auction, where the highest bidder wins but pays the second-highest bid amount.

Online auctions, conducted through digital platforms, have gained prominence, and among them, combinatorial auctions stand out. Combinatorial auctions allow bidders to submit bids for sets of items, specifying the desired quantity of each item. The auctioneer then determines the optimal combination of bids to maximize revenue. This approach eliminates the constraints of traditional auctions, which often limit sales to single items or predefined sets.

One of the significant limitations of traditional auctions is their inability to handle multiple item bidding efficiently. Buyers interested in acquiring multiple items may find it cumbersome to place individual bids for each item. In contrast, combinatorial auctions offer greater flexibility to both buyers and sellers. Buyers can select precisely which items they wish to bid on and purchase, enhancing the overall auction experience and potentially leading to more profitable outcomes.

Online combinatorial auctions offer several advantages over their traditional counterparts. They enable buyers to bid on multiple items simultaneously, streamlining the process and offering the convenience of participation from anywhere. Furthermore, online platforms can handle larger scales of bidders and items, enhancing reach and revenue potential.

This study's primary focus has been on online and combinatorial auctions. The aim is to integrate these auction formats by developing a dynamic website capable of conducting combinatorial auctions. Throughout the upcoming chapters, we have explored

the foundational concepts of auction negotiation, delved into the various auction mechanisms, and emphasized the importance of online and combinatorial auctions. We've also covered database management, frontend development using React, and the technical implementation of auction algorithms. These chapters collectively lay the groundwork for our project, ensuring it is poised to address the evolving landscape of online auctions effectively.

# Bibliography


[1] Amza, Chanda, Cox, Elnikety, Gil, Rajamani, Zwaenepoel, Cecchet, and Marguerite. Specification and implementation of dynamic web site benchmarks. In *2002 IEEE international workshop on workload characterization* (2002), IEEE, pp. 3–13.

[2] Andersson, A., Tenhunen, M., and Ygge, F. Integer programming for combinatorial auction winner determination. In *Proceedings Fourth International Conference on MultiAgent Systems* (2000), IEEE, pp. 39–46.

[3] Anjard, R. P. The basics of database management systems (dbms). *Industrial Management & Data Systems 94*, 5 (1994), 11–15.

[4] ASLI, L. *Les encheres combinatoires multiobjectif dynamiques*. PhD thesis, Thése de doctorat en sciences, Université de USTHB, Alger, 2019. Chapitre 2: 31-44, Chapitre 4: 62-70.

[5] Bichler, M., and Kalagnanam, J. Configurable offers and winner determination in multi-attribute auctions. *European Journal of Operational Research 160*, 2 (2005), 380–394.

[6] Booch, G., Rumbaugh, J., and Jackobson, I. Uml: unified modeling language. *Versão* (1997).

[7] Ehrgott, M. *Multicriteria optimization*, vol. 491. Springer Science & Business Media, 2005. [47-57].

[8] Guitart, J., Beltran, V., Carrera, D., Torres, J., and Ayguadé, E. Characterizing secure dynamic web applications scalability. In *19th IEEE International Parallel and Distributed Processing Symposium* (2005), IEEE, pp. 10–pp.

[9] Harrington, J. L. *Relational database design clearly explained*. Elsevier, 2002.

[10] Kagel, J. H., and Levin, D. Auctions: A survey of experimental research, 1995-2008. *Handbook of experimental economics 2*, 2 (2008).

[11] Letkowski, J. Doing database design with mysql. *Journal of Technology Research 6* (2015), 1.

[12] McAfee, R. P., McMillan, J., and Wilkie, S. The greatest auction in history. *Better living through economics* (2010), 168–184.

[13] MySQL, A. Mysql, 2001. Interval: Pages 3 to 7 and from 231 to 234.

[14] Ockenfels, A., Reiley Jr, D. H., and Sadrieh, A. Online auctions, 2006. Interval: Pages 48 to 53.

[15] Press, M. Microsoft computer dictionary, 2002. Individual pages: 141, 446, 252, 145.

[16] Roughgarden, T. Algorithmic game theory. *Communications of the ACM 53*, 7 (2010), 78–86.

[17] Sakawa, M., Kato, K., Sunada, H., and Shibano, T. Fuzzy programming for multiobjective 0–1 programming problems through revised genetic algorithms. *European Journal of Operational Research 97*, 1 (1997), 149–158.

[18] Sandholm, T. Approaches to winner determination in combinatorial auctions. *Decision Support Systems 28*, 1-2 (2000), 165–176.

[19] Sariddichainunta, P., and Sinapiromsaran, K. The winner determination model and computation for linear arrangement of booth auction. *Information Technology Journal 7*, 2 (2011), 46–51.

[20] Sasaki, B. M., Chao, J., and Howard, R. Graph databases for beginners. *Neo4j* (2018).

[21] Thakur, P. Evaluation and implementation of progressive web application. 3–12.

[22] Wolfstetter, E. Auctions: an introduction. *Journal of economic surveys 10*, 4 (1996), 367–420.

[23] Yan, C., Cheung, A., Yang, J., and Lu, S. Understanding database performance inefficiencies in real-world web applications. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 1299–1308.

[24] Zaman, S., and Grosu, D. Combinatorial auction-based mechanisms for vm provisioning and allocation in clouds. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)* (2012), IEEE, pp. 729–734.

# Abstract


## Abstract:

This project focuses on the Implementation of Combinatorial Auctions within a Dynamic Web Application, leveraging key technologies such as MySQL, React, and Python for addressing the Winner Determination Problem. It addresses the universal human behavior of price negotiation, highlighting the limitations of traditional auctions and the potential of online combinatorial auctions.

Through comprehensive chapters, it delves into the core concepts of auction mechanisms, robust database management, intuitive frontend development, and the technical implementation of dynamic bidding algorithms. This endeavor aims to bridge the gap between traditional and innovative auction formats, offering a dynamic platform to conduct efficient and flexible combinatorial auctions in a user-friendly online environment.



**Keywords :** Combinatorial Auctions, Dynamic Web Application, MySQL, robust database management, Winner Determination Problem.



## Résumé :

Ce projet se concentre sur la mise en œuvre des enchères combinatoires au sein d'une application Web dynamique, en utilisant des technologies clés telles que MySQL, React et Python pour résoudre le problème de détermination du gagnant. Il aborde le comportement humain universel de la négociation des prix, mettant en évidence les limites des enchères traditionnelles et le potentiel des enchères combinatoires en ligne.

À travers des chapitres complets, il explore les concepts fondamentaux des mécanismes d'enchères, de la gestion robuste de bases de données, du développement intuitif de l'interface utilisateur et de la mise en œuvre technique d'algorithmes d'enchères dynamiques. Cette initiative vise à combler l'écart entre les formats d'enchères traditionnels et innovants, en proposant une plateforme dynamique pour mener des enchères combinatoires efficaces et flexibles dans un environnement convivial en ligne.



**Mots-clés :** Enchères Combinatoires, Application Web Dynamique, MySQL, Gestion robuste de bases de données, Problème de détermination du gagnant.

Modifications:

1. Liste des abréviations alignées.
2. Diagramme de base de données corrigé (ER-Diagramme).
3. Ordre des titres : Dans le Chapitre 1 : Database (DB) est avant Network Database (Network DB).
4. Renommer les titres des chapitres :
    - Chapitre 1 : Fondements de la Modélisation et de l'Informatique
    - Chapitre 2 : Fondements des Ventes Combinatoires
5. Supprimer une figure.
6. Supprimer le type NoSQL.
7. Clarification du diagramme WDP (Diagramme de Programmation Web).
8. Ajouter les intervalles de page dans la bibliographie.
9. Ajouter un résumé en français.