***SRS FOR ONLINE AUCTION SYSTEM***

***PROJECT FOR –***

**SOFTWARE ENGINEERING (SEM IV)**

***SUBMITTED TO -***

**DR. POOJA JAIN**

***SUBMITTED BY-***

***AISHIT DUA BT17CSE078***

***SUPRIYA VENKATESH BT17CSE106***

***Introduction:***

An **auction** is a process of buying or selling goods or services by offering them up

for bid, taking bids, and then selling the item to the highest bidder. There are

several different types of auctions and certain rules exist for each auction. There

are variations for an auction which might include minimum price limit (least bid),

maximum price limit, time limitations etc. Depending upon the auction method,

the bidder can participate remotely or in person. Remote auction includes

participating through the telephone, mail, and the internet. Shopping online has

been increasing on an exponential scale. Online auction system is also increasing

rapidly. Online auction is becoming more and more popular in electronic

commerce and hence its system must increase its quality and security. The online

auction system is a model where we participate in a bid for products and service.

This auction is made easier by using online software which can regulate processes

involved. There are several different auction methods or types and one of the most popular methods is English auction system. This system has been designed to be

highly-scalable and capable of supporting large numbers of bidders in an active

auction. Online Auctioning System has several other names such as e-Auctions,

electronic auction etc. The requirement for online auction or online bidding can be

more accurately specified by the client. It should be healthy and will be a good

practice when it is made more transparent as a matter of fact. Online Bidding has

become more wide spread in all sorts of industrial usage. It not only includes the

product or goods to be sold, it also has services which can be provided. Due to

their low cost this expansion made the system to grow. Online bidding has

become a standard method for procurement process. Bidders can be maintained

in a single database according to the preference, and they can be monitored.

User’s data can be maintained in a confidential way for validity and integrity of

contractual documentation. In this project, there is a gist of a module of online

auctioning system.

***Purpose:***

The purpose of this document is to present a detailed description of the Online

Auction System. It will explain the purpose and features of the system, the

interfaces of the system, what the system will do, the constraints under which it

must operate and how the system will react to external stimuli. This document

also contains the functionalities and requirements of the system.

***Scope:***

This software system will provide the specifications of a module for the Online

Auction System for a number of customers. This system will be designed to

maximize the customer’s productivity by providing various options such as bid,

sell, etc. which or else had to be done manually. More specifically, this system is

designed to allow the individual users to login with their unique email ID and

password which is provided to the at the time of making a new account.

***Objectives:***

The objective is to develop an unadorned auctioning site where many kinds of

products can be listed, bidded and sold and to provide value added services to all

the customers.

* Secure registration of all users including a personal profile.
* Prior to each bid, the user’s bank or credit account must be authenticated for
* Available balance required for the bid.
* Complete Search/Site Map of the entire site for easy access.

***External Interfaces:***

* Requires a system with working internet connection.
* Requires a browser (preferably chrome) on the system to open the website.
* Requires a fast internet connection for the uninterrupted user experience.

***Performance Requirements:***

The system has been designed such that it will give the most optimized performance with the following:

* Works on a system with CHROME, MOZILLA, IE and all the other popular BROWSERS.
* Since it is a portable, it can be opened in any operating system and it will provide the same performance.
* Source Code – For working in dev mode and reading the code, the best IDE is INTELLIJ for JAVA, for working on the web dev and php documents we can use the ATOM IDE.

***Functional Requirements:***

**User side**:

Req 1: Login to a new session UC 1: login();

Req 2: Add a new user UC 2: Adduser();

Req 3: Bid on a specific item UC3: Bid();

Req 4: View listed item to buy UC 4: Buy();

Req 5: Sell an item UC 5: Sell();

Req 6: Auction items that were listed today UC 6: NewAuctionItemToday();

Req 7: Auction items ending today UC 7: NewAuctionItemsEndDay();

Req 9: View the list of all items UC 9: ViewBidItems();

**Admins side**:

Req 8: View the list of all present customers UC 8: ViewCustomer();

Req 9: View the list of all items UC 9: ViewBidItems();

Req10: Remove a particular listed item UC 10: RemoveItem();

Req11: View the profile of a particular customer UC 11: ViewProfile();

Req12: Ban a particular customer permanently UC 12: BanProfile();

Req 6: Auction items that were listed today UC 6: NewAuctionItemToday();

Req 7: Auction items ending today UC7: NewAuctionItemsEndDay();

***Design Constraints:***

* Though the website has been designed with optimized performance in

mind, the major constraint is that it is not mobile friendly.

* The site has been created with all basic web dev technologies, and even JAVA which is a really robust language, but due to the bandwidth of the server allotted, the website cannot handle more than 150 users at a time.
* Though when a customer logs in he can buy and sell on the same session, but simultaneous bidding and selling on different tabs cannot be performed.
* If a user has logged in from a device. he cannot login from another device until he logs out from that device itself i.e. A user can login only from one device at a time.

***Description of the individual use cases:***

* **Login()**:

USE CASE NUMBER: 1

ACTORS: CUSTOMER

This function is used to login to a new session. The user is expected to enter the user id and password. This finishes the successful authentication and redirects the user to the homepage. On unsuccessful login, the user is directed to the forgot password page where he has to reset his password. After that, consecutive login will take place with the new password.

* **AddUser()**:

USE CASE NUMBER: 2

ACTORS: NEW MEMBER: (NOT A CUSTOMER)

This function is a subset of the login(). In this, the new user is prompted to add his account in the website. This includes creating a new user id and password for the accessing of the individual account in the webite.

* **Bid():**

USE CASE NUMBER: 3

ACTORS: CUSTOMER

This function is for bidding of the items. The customer upon logging in, is directed to the homepage and then is expected to place his bid on the current items which are being bid upon at the moment. Upon successful bid, the item will be sold to the customer.

* **Sell():**

USE CASE NUMBER: 4

ACTORS: CUSTOMER

This function is for selling of a specific item. Here, the customer is selling his own individual item. The user is expected to set a basic amount called starting bid which is the minimum amount above which the item has to be bidded upon by the other customers who are interested in the product.

* **Buy():**

USE CASE NUMBER: 5

ACTORS: CUSTOMER

This function is for buying of an item. Here, upon successful login, the user goes through the list of all the items and then proceeds to the list of current items which are being sold. After deciding what to buy, the user is expected to place a bid which is more than the minimum bid for that particular item.

* **NewAuctionItemToday():**

USE CASE NUMBER: 6

ACTORS: CUSTOMER, ADMIN

This function is used to display all the items which upon which the bidding/selling is taking place on that particular day. The list of all the new items put up for the auction is displayed in this.

* **NewAuctionItemsEndDay():**

USE CASE NUMBER: 7

ACTORS: CUSTOMER, ADMIN

This function is also used to display all the items upon which the bidding/selling is supposed to be taking place on that particular day. The list of all the items whose scope ends of that particular day is displayed.

* **ViewCustomer():**

USE CASE NUMBER: 8

ACTOR: ADMIN

This function is from the admins side. This is for the admin to check the list of all the customers who have an account on the website. The admin has a choice to see customers through sorting also.

* **ViewBidItems():**

USE CASE NUMBER: 9

ACTORS: CUSTOMER,ADMIN

This function is from the side of the admin and the customer. This is used to view the list of all the bidded items for the admin at any point of time. All the items from the above mentioned modules will be accessible to the admin via the function.

* **RemoveItem():**

USE CASE NUMBER: 10

ACTORS: ADMIN

If the item is not within the set standards of the particular website, then the administrator can remove the item. This only available for the admin. The admin can remove items which are currently available.

* **ViewProfile():**

USE CASE NUMBER: 11

ACTOR: ADMIN

This function is used to view any particular profile of any particular customer. By this, the admin will have direct access to the customer’s profile.

This is used as a display function access.

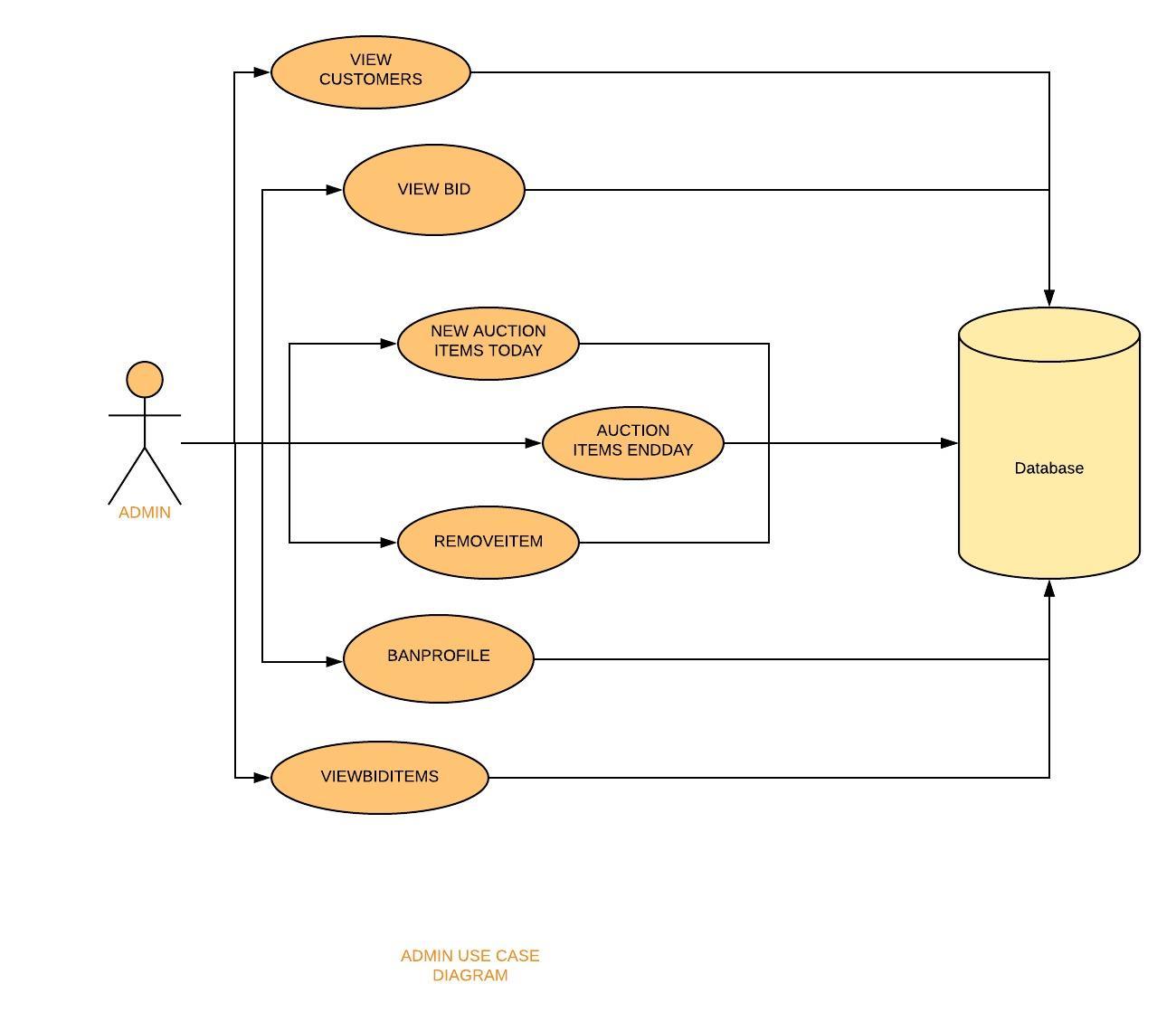
* **BanProfile():**

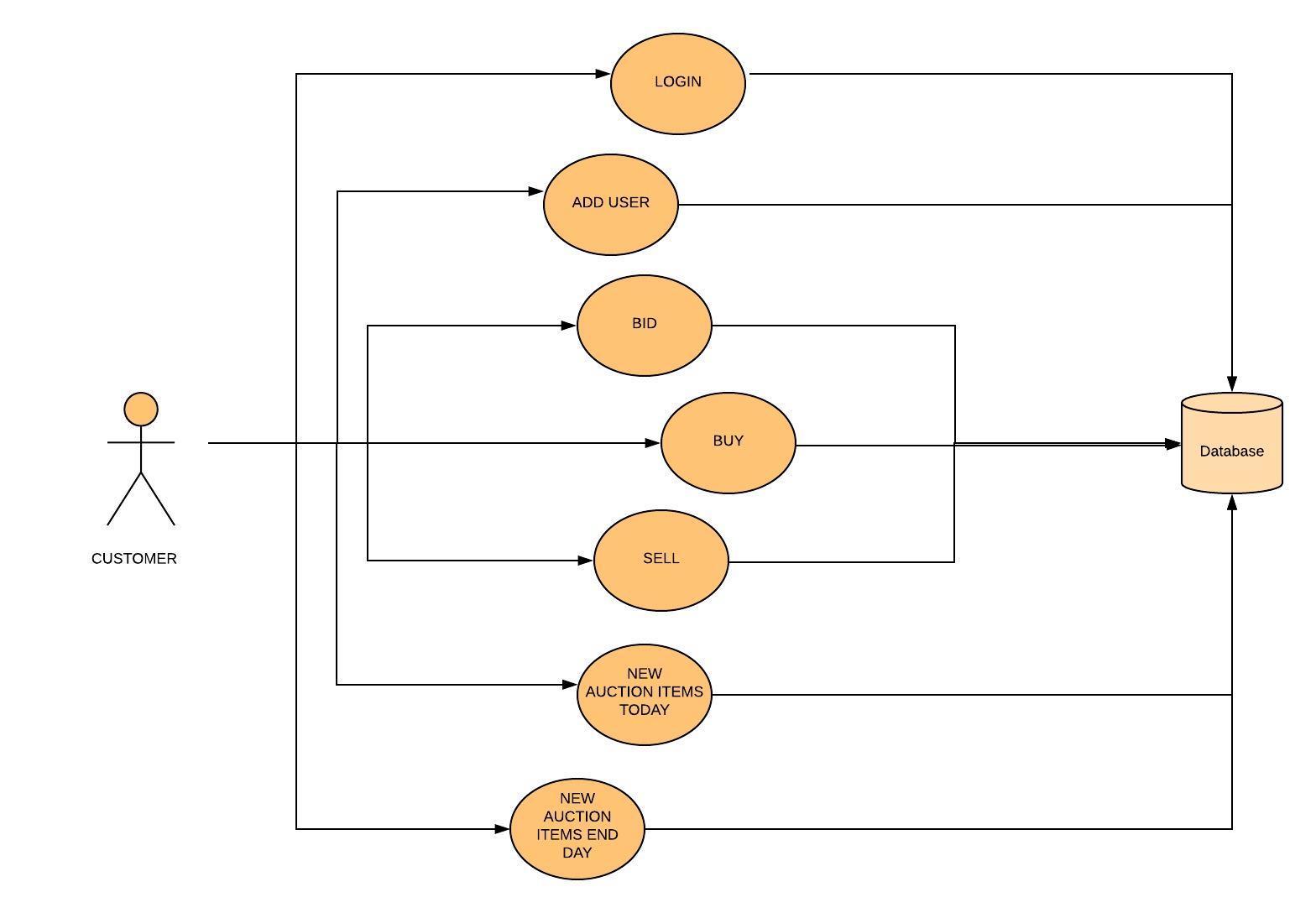
USE CASE NUMBER: 12

ACTORS: ADMIN

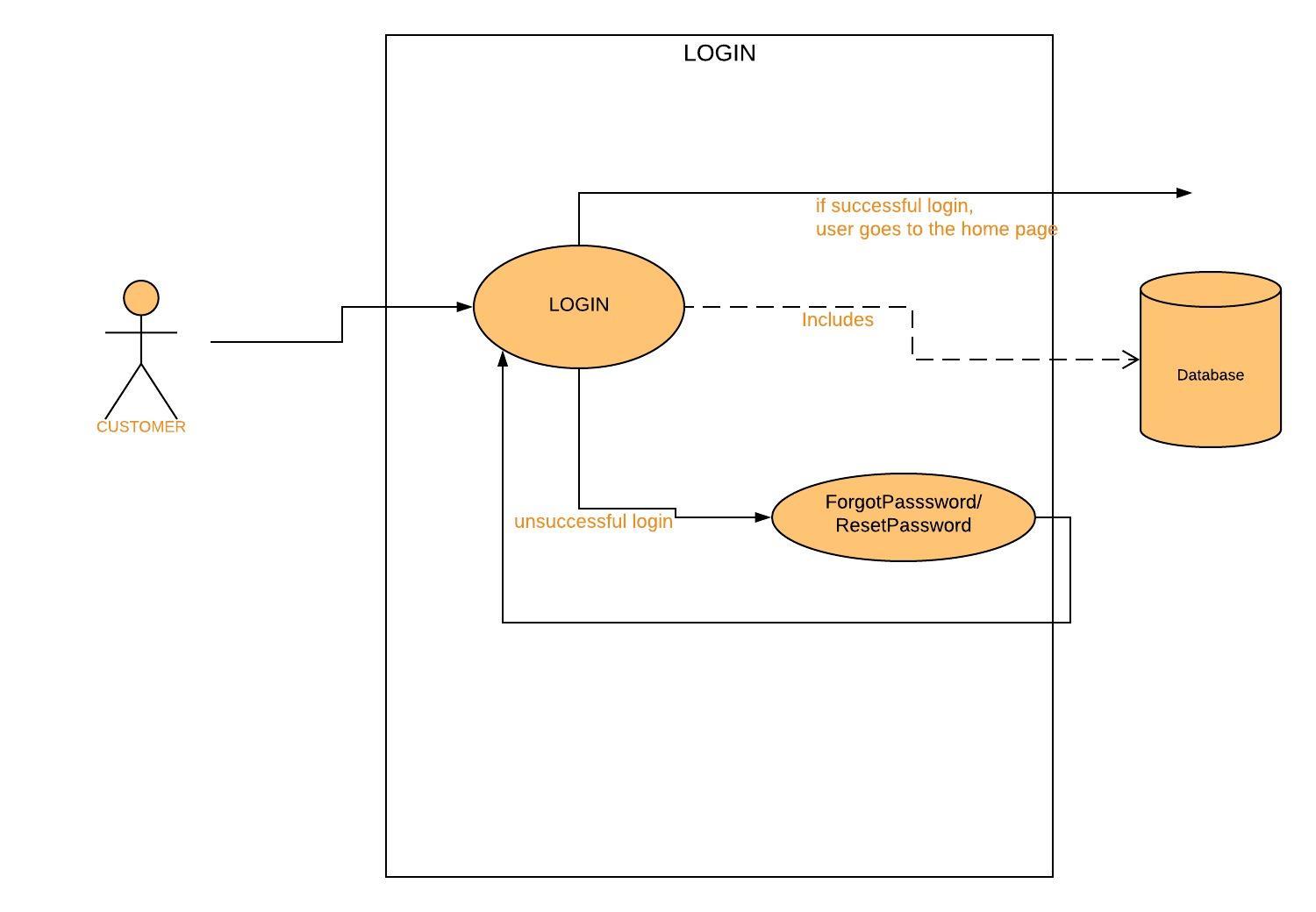
This is the function which gives the admin to authority to ban a profile when it fails to meet the necessary requirements. In this process, the ID is searched on the database and then it is banned.

xxx--- USE CASE DIAGRAM ON THE NEXT PAGES---xxx

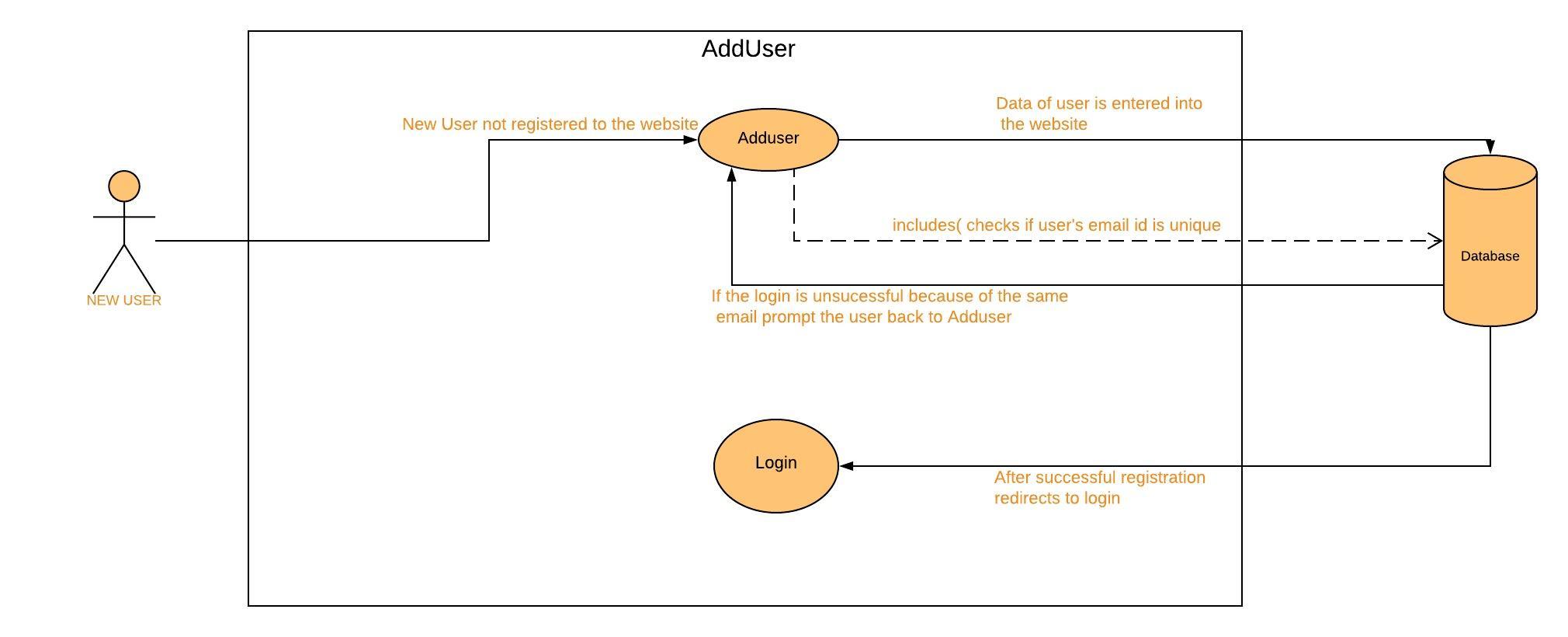




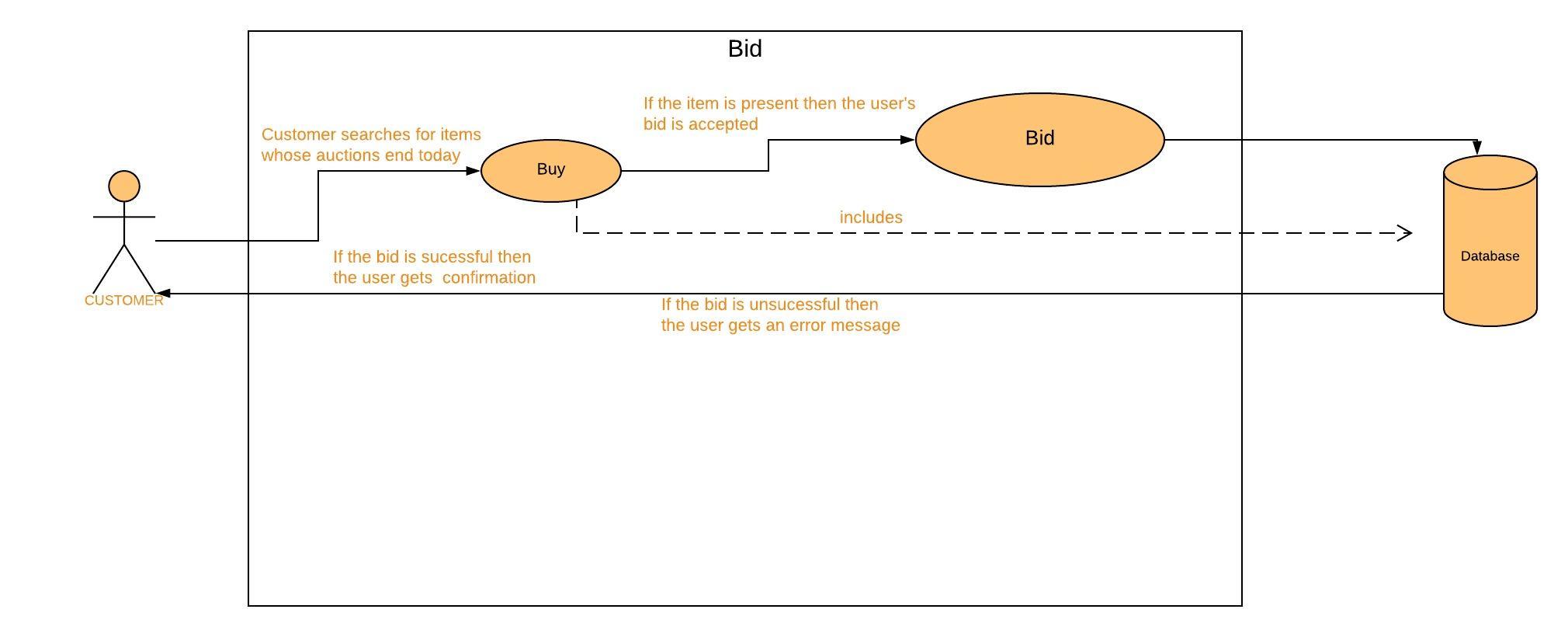
USER SIDE USE CASE DIAGRAM



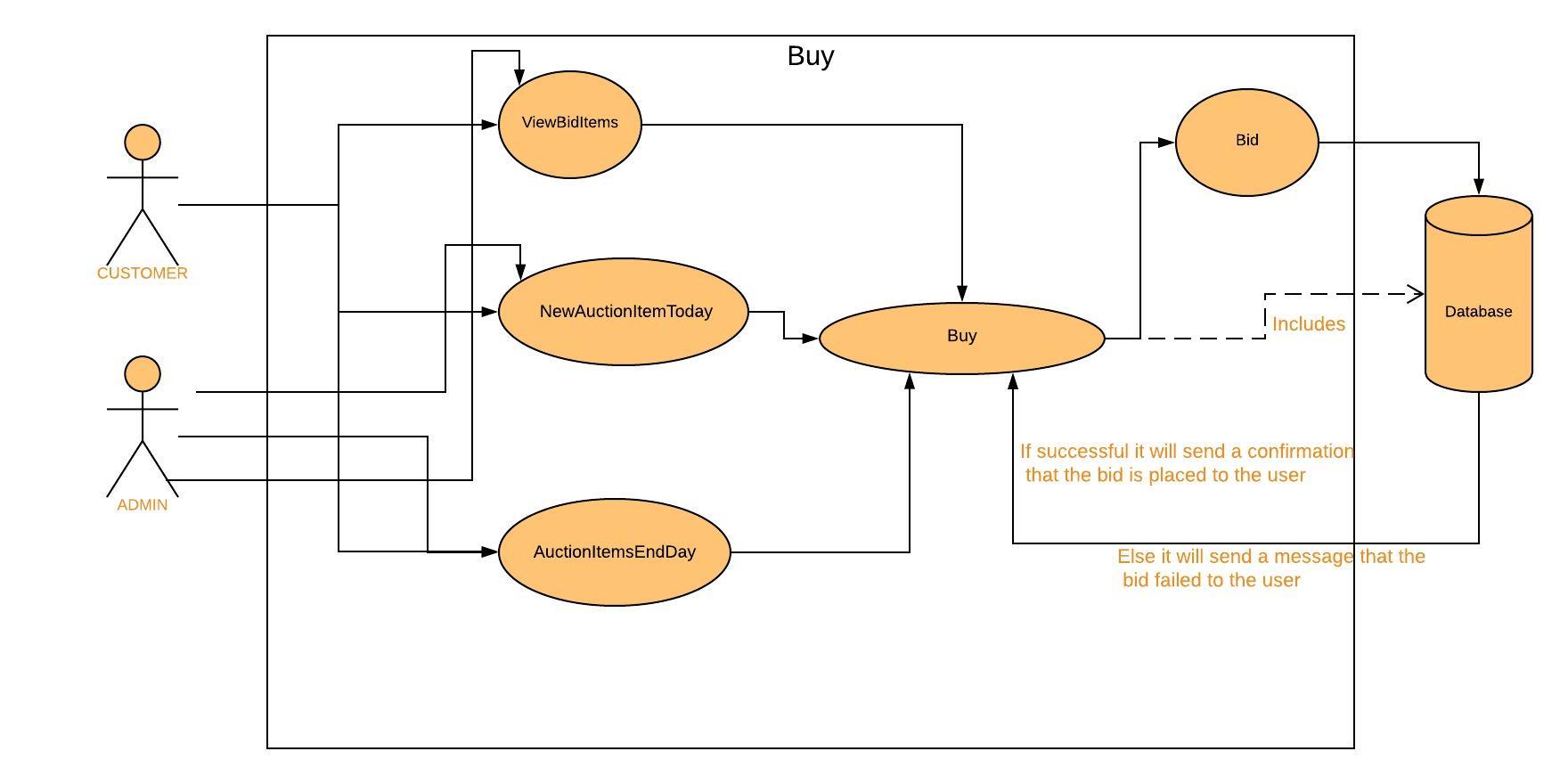
UC1: LOGIN



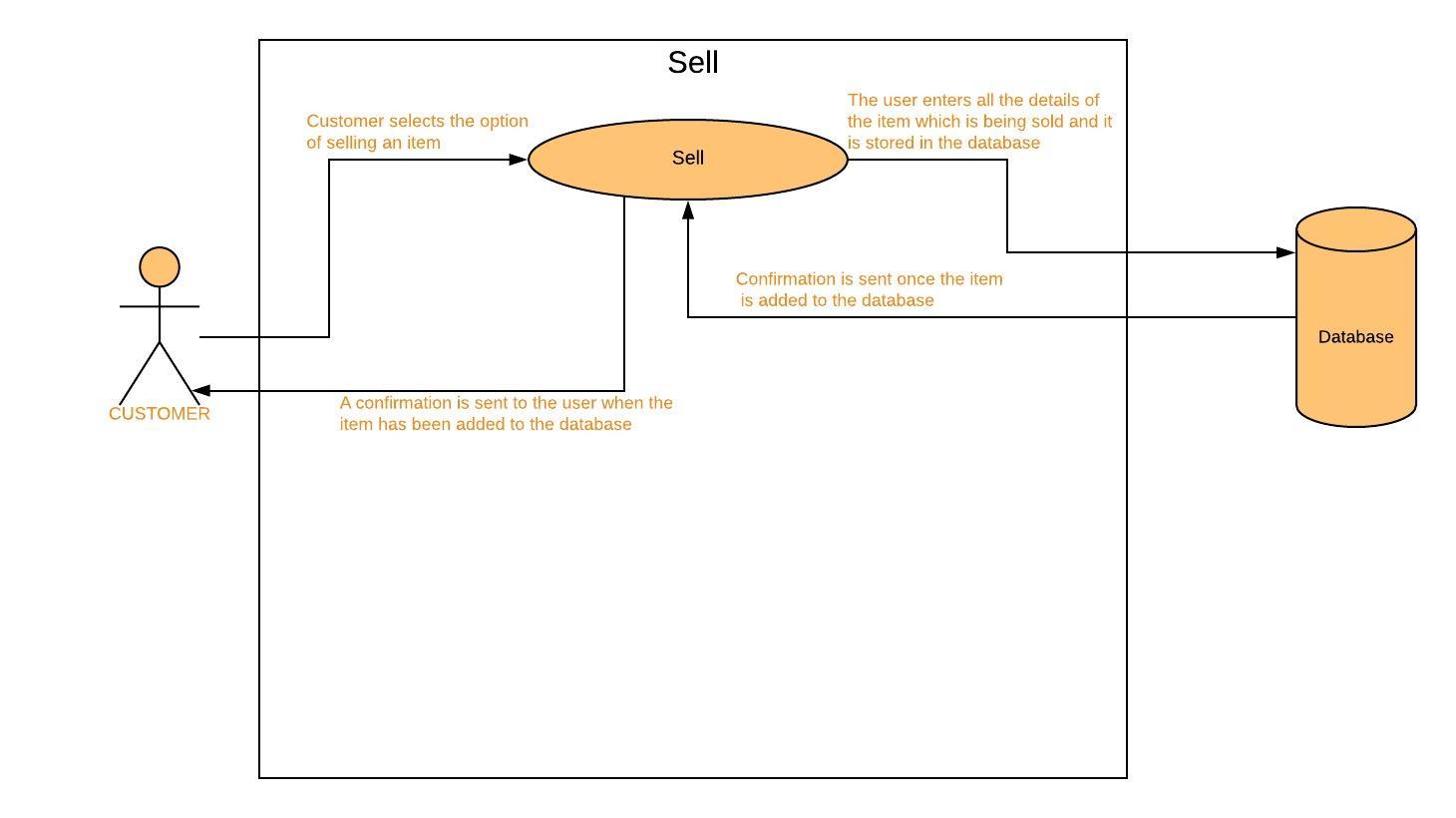
UC2: ADDUSER



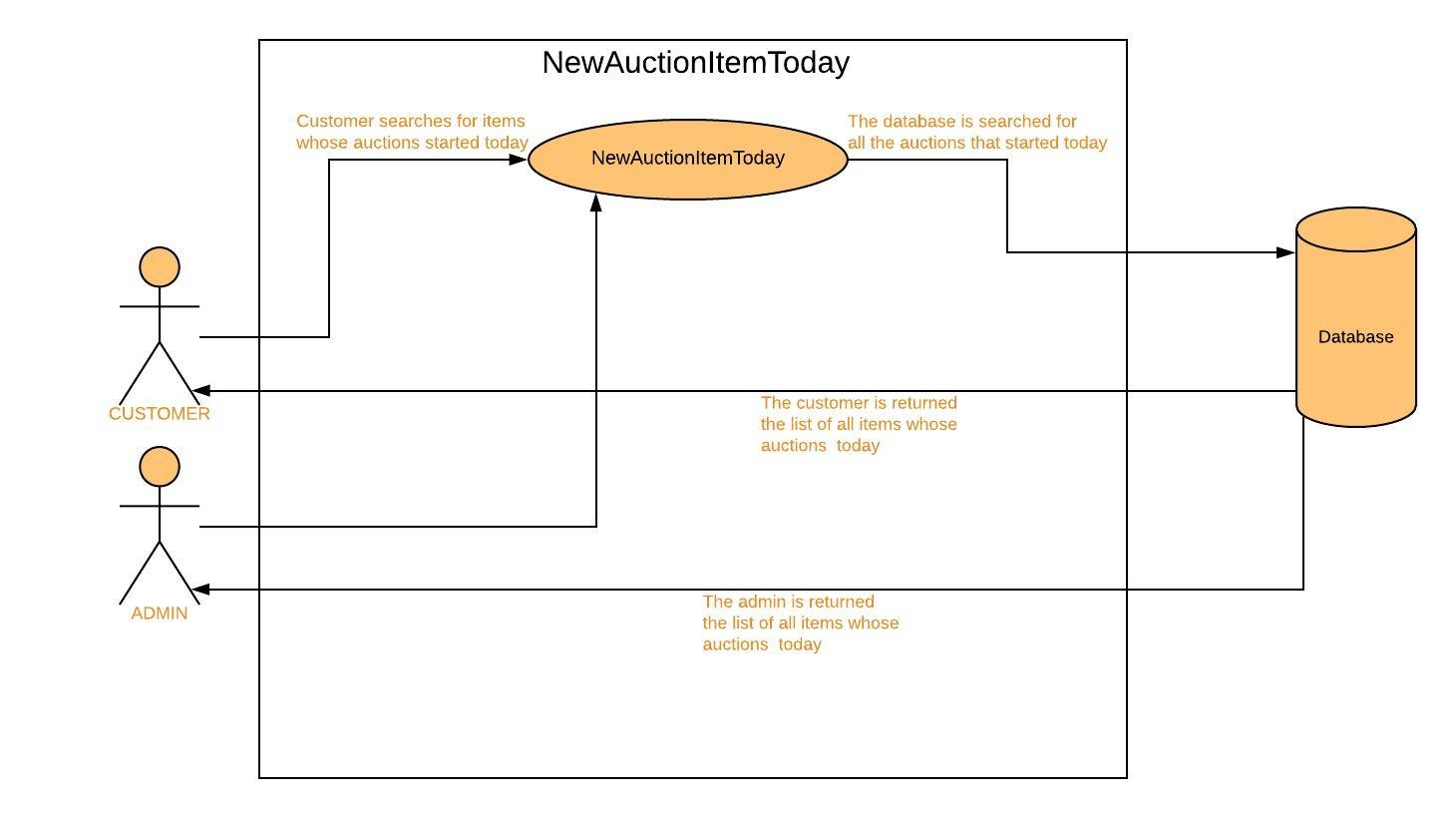
UC3: BID



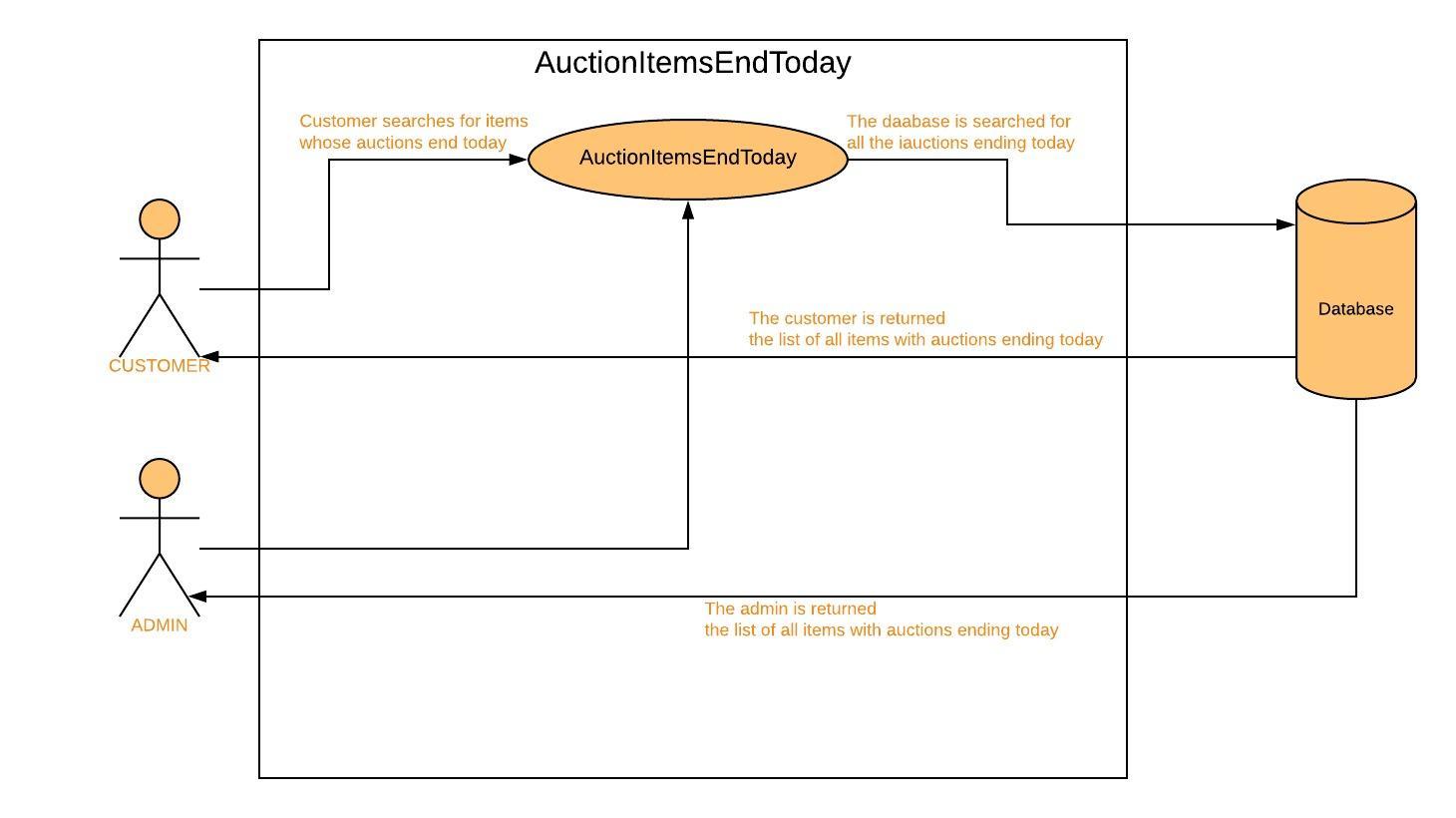
UC4: BUY



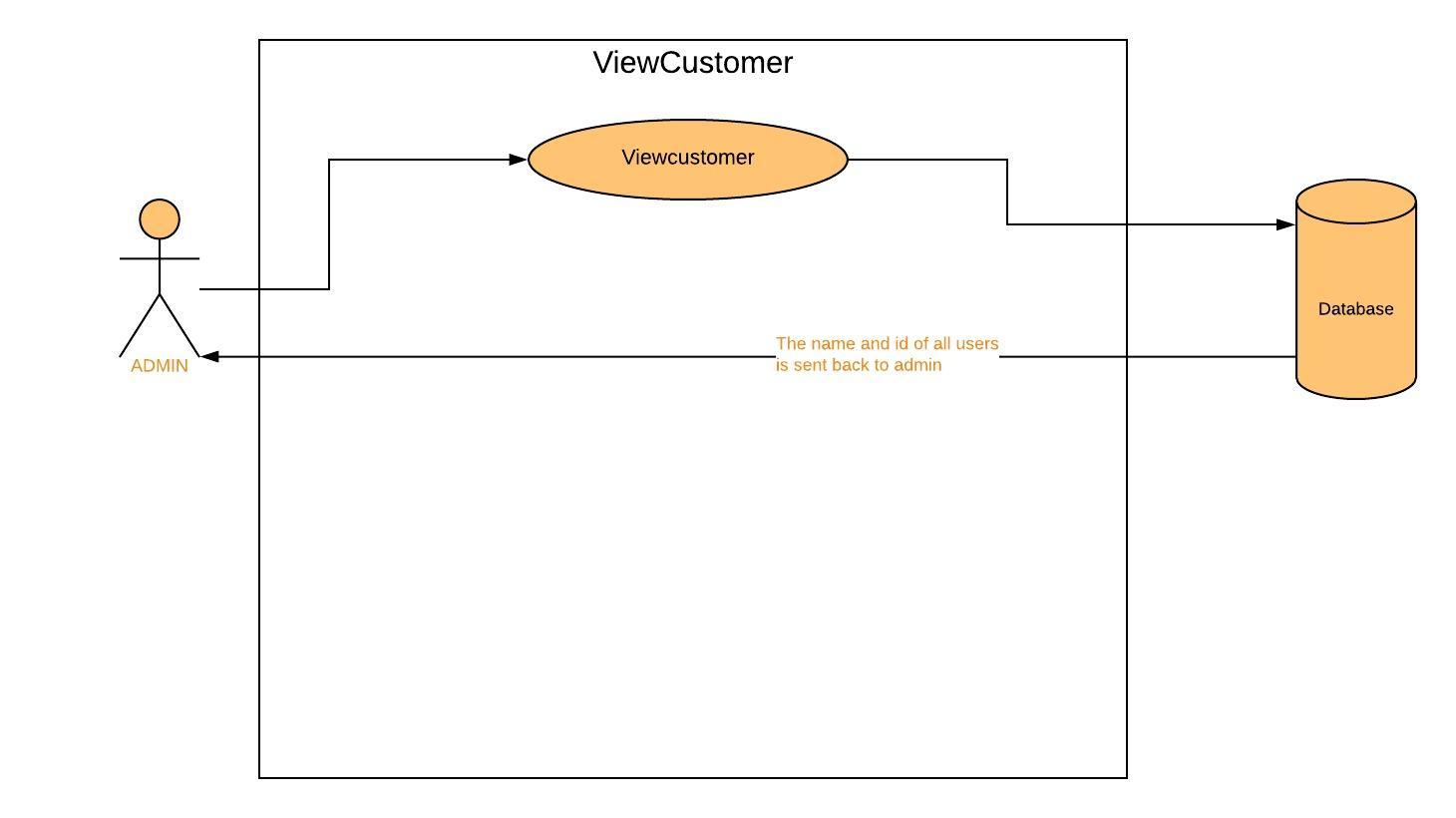
UC5: SELL



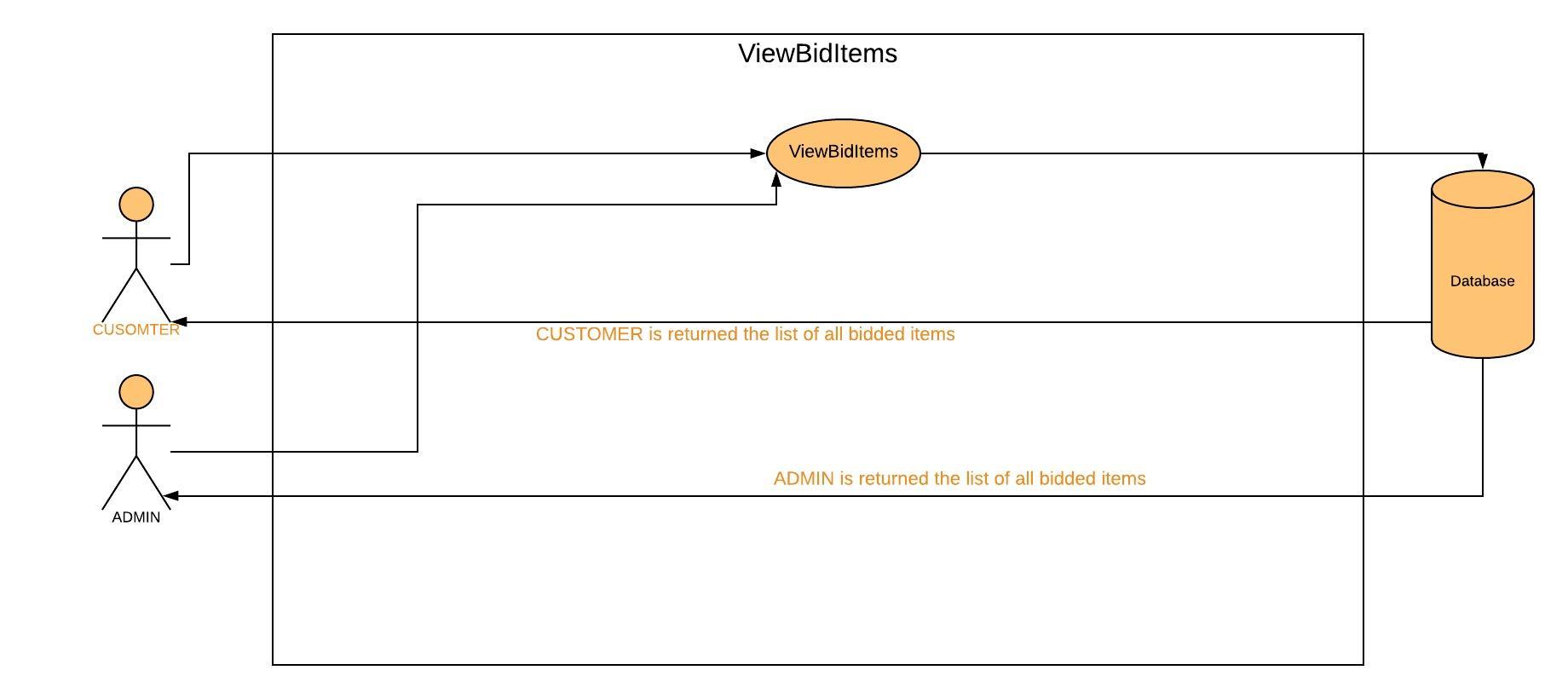
UC6: NEWAUCTIONITEMTODAY



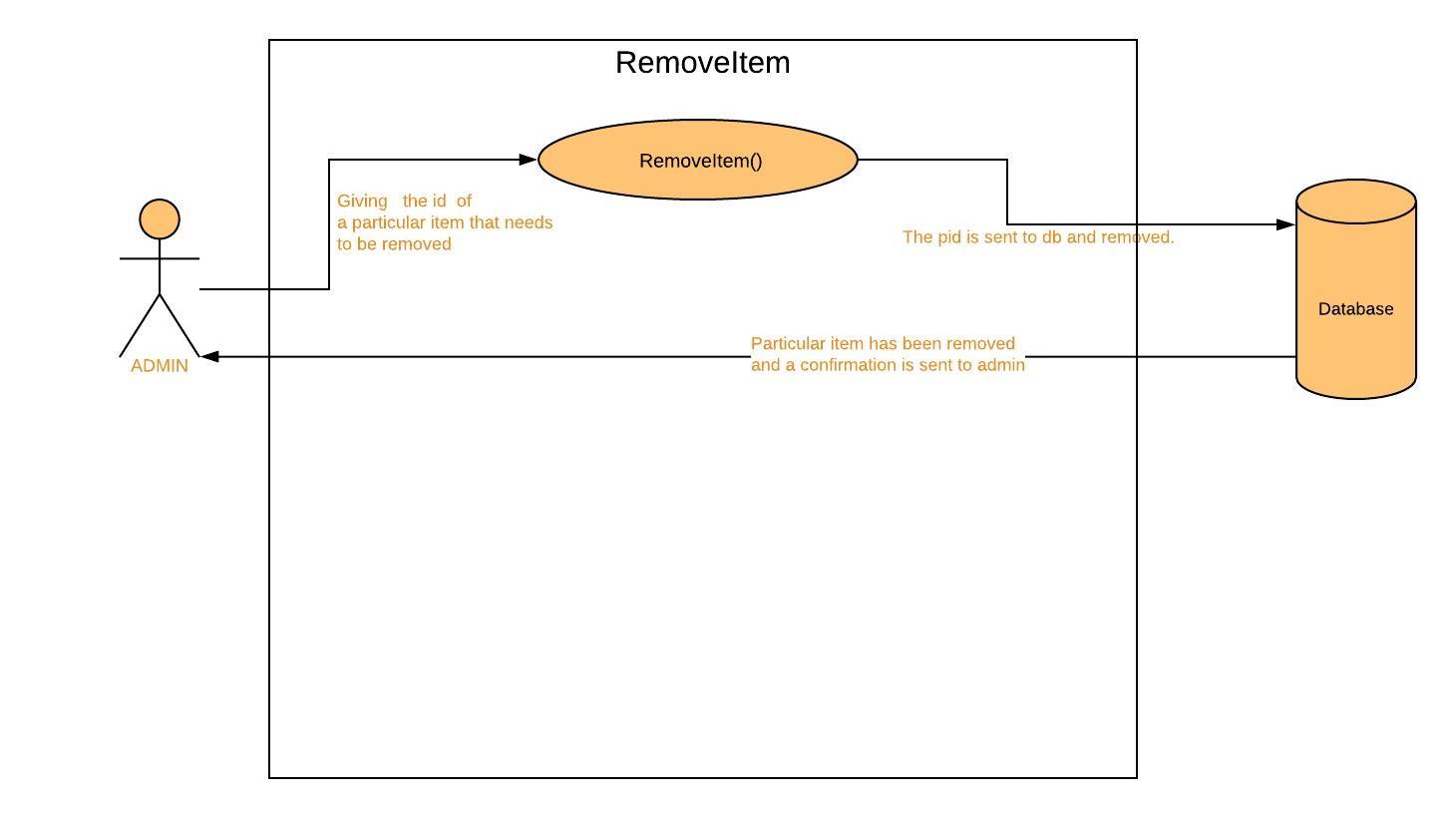
UC7: AUCTIONITEMSENDTODAY



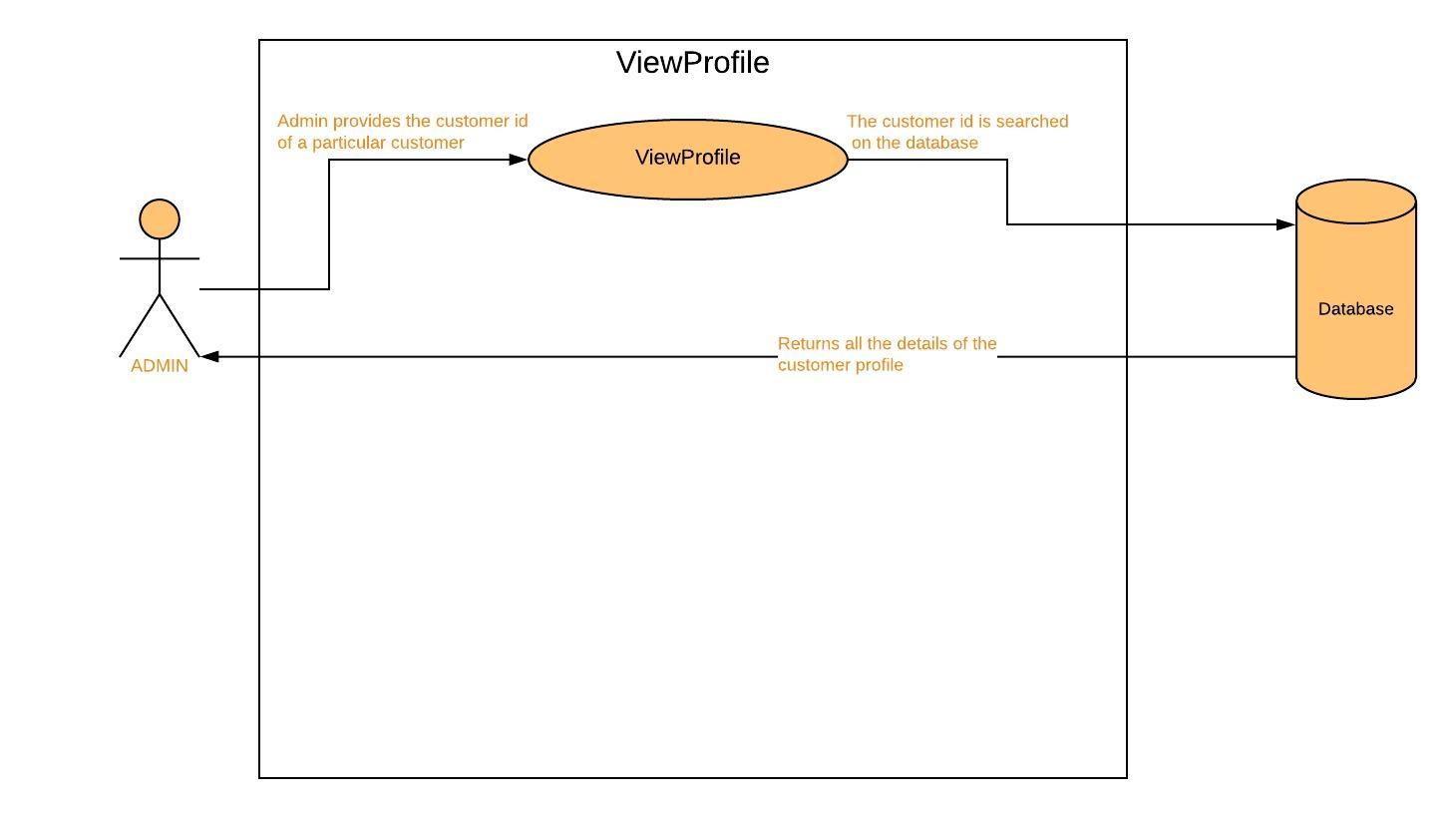
UC8: VIEWCUSTOMER



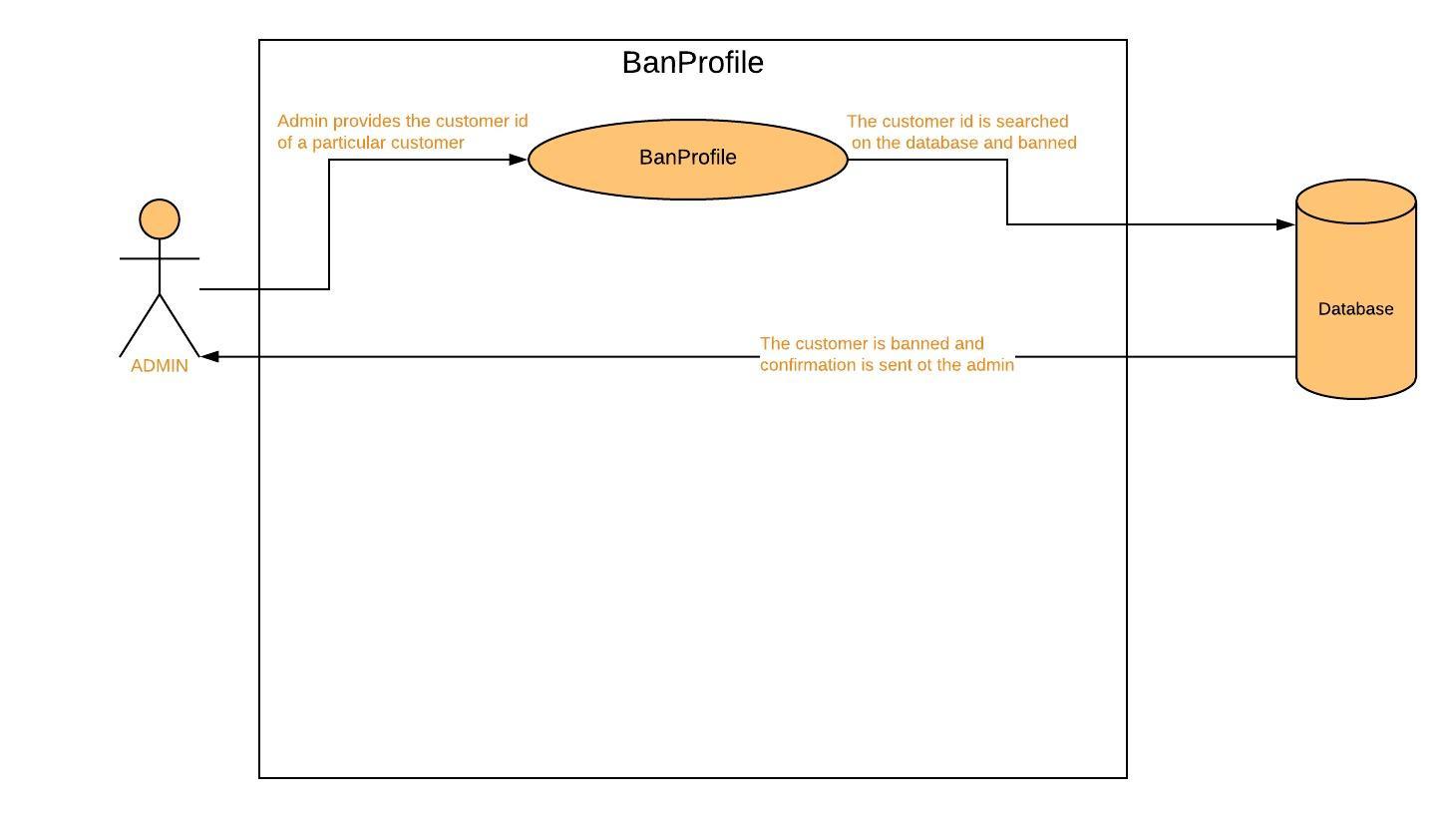
UC9: VIEWBIDITEMS



UC10: REMOVEITEM



UC11: VIEWPROFILE



UC12: BANPROFILE

***PROCESS MODEL – THE WATERFALL MODEL***

In this online auction system, the main process model we employ is the waterfall

mode. The waterfall model basically can be defined as- The waterfall model,

sometimes called the classic life cycle, suggests a systematic, sequential approach

to software development that begins with customer specification of requirements

and progresses through planning, modeling, construction, and deployment,

culminating in ongoing support of the completed software.

The main reason for choosing the waterfall process model would be because the

requirements for a problem are well understood—when work flows from

communication through deployment in a reasonably linear fashion.

The requirement of the system is already well defined as stated in our use case

diagrams etc. Also, the only disadvantage of using the disadvantage of the process

model is that the software won't be available till late in the development phase,

but it is okay since we are only delivering one increment of our software after

extensive testing.



***THE TYPE OF AGILE FRAMEWORK – SCRUM***

In our project which is Online Auction System, the Scrum framework can be applied

suitably. Scrum is a framework within which people can address complex adaptive

problems, while productively and creatively delivering products of the highest

possible value. In this, small working teams are organized to "maximize

communication, minimize overhead, and maximize the strength of sharing tacit and

informal knowledge. Often thought of as an agile project management framework,

Scrum describes a set of meetings, tools, and roles that work in concert to help

teams structure and manage their work. There are various prerequisites for

choosing scrum. This employs a backlog in which there is a list of project

requirements that provide buisness value. Items can be added anytime. This makes

it customer friendly. The specifications are already taken care of in the functional

requirements and the use cases, but if there are any additional features, they can

always be added in the list of backlog. The heart of Scrum is a Sprint, a time-box of

two weeks or one month during which a potentially releasable product increment

is created. A new Sprint starts immediately after the conclusion of the previous

Sprint. During this time, the backlog items that the sprint work units address are

frozen. All the data from the use cases are implemented here. Meetings are

organized in which the modules of the system can be looked into. After the

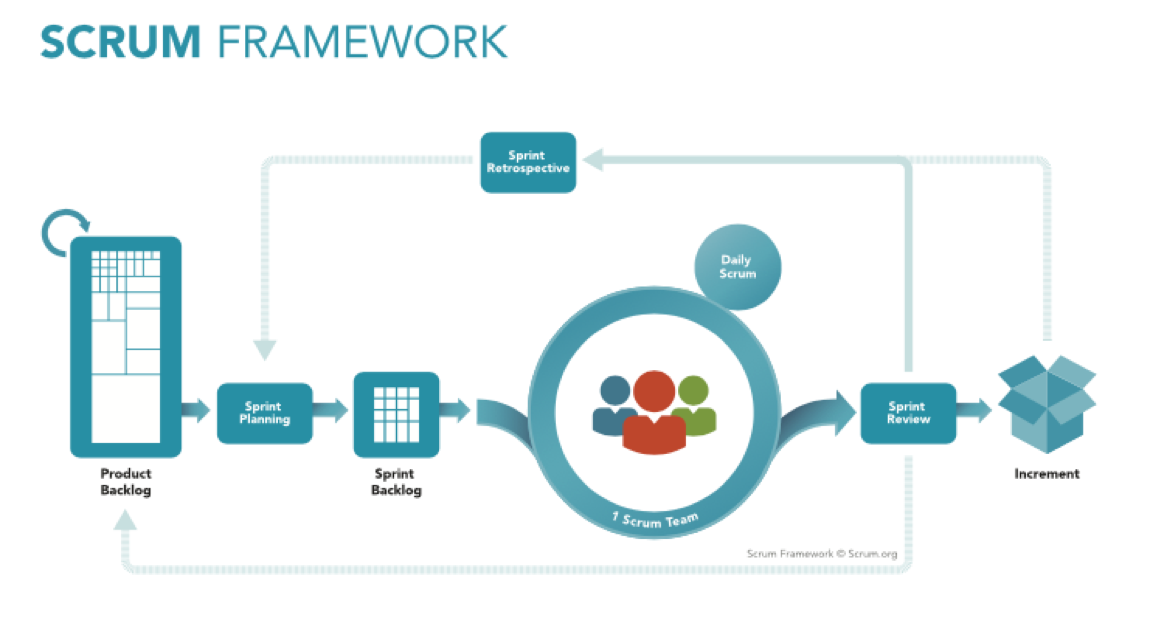
completion of every backlog, the next accomplishments of the backlog list is

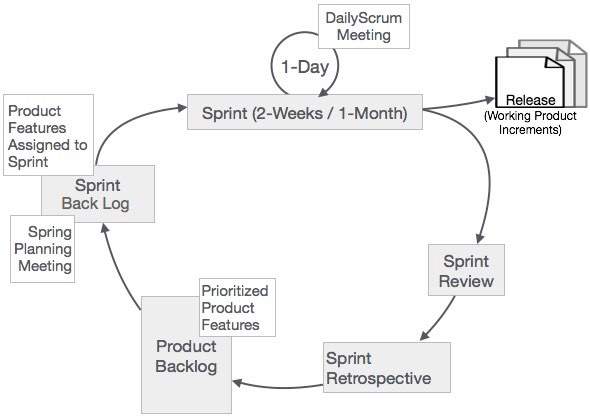
discussed. For example, after the completion of the login() module, a scrum

meeting can be held. The Sprint Retrospective occurs after the Sprint Review and

prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect itself

and create a plan for improvements to be enacted during the subsequent Sprint.





***UI DESIGN (FACTORS):***

|  |
| --- |
|  |
|  |
|  |
|  |

Design is very important for any project. In this project of online auction,

there are three golden rules which are followed to accomplish this.

•User in control:

If the project had to be exported commercially for a

particular user, then, as a designer, the goal is to simplify the

mode of interaction. Modes of interaction shouldn’t force a user

into unnecessary or undesired actions. The interaction should be

flexible and straight forward. Since the project requires the customer to

interact, the interactions should be undo-able.

•Reduced user’s memory load:

The project should remember pertinent user information and and assist the

user with an interaction scenario.

•Consistent interface:

In this project, customers can buy, sell and bid products. This implies that

the interface will acquire information in a consistent order. For example,

module name mentioned in every screen display of the project.

ANALYSIS:

***ACHIEVING SOFTWARE QUALITY:***

High quality of software is an important goal to be accomplished in the

development of a new software project. It is the result of good project

management and solid software engineering practice. Management and practice

are applied within the context of four broad activities that help a software team

achieve high software quality: software engineering methods, project management

techniques, quality control actions, and software quality assurance.

•Software Engineering Method:

An effective software process applied in a manner that

creates a useful product that provides measurable value for those who produce it

and those who use it. An effective software process establishes the infrastructure

that supports any effort at building a high-quality software product. A useful

product delivers the content, functions, and features that the end user desires, but

as important, it delivers these assets in a reliable, error-free way. There are many

factors a software must abide to achieve software quality:

1.Garvin’s Quality Dimensions: Our project of “The online auction system”

abides by these dimensions by following performance, feature, reliability, and

conformance qualities.

2.McCall’s Quality Factors: The auction system focuses on operational

qualities, its ability to undergo changes and to adapt to new environments. This is

achieved by correctness, efficiency, usability and integrity. The interface can be

specified to the needs of the user as per required.

3.ISO 9126 Quality Factors: Our project identifies six key attributes of

software, functionality, usability, efficiency, reliability, maintainability and

portability. The software is usable for a considerable amount of audience.

4.Targeted Quality Factors: The interface layout of the project is conductive to

easy understanding. The operations such as logging in, getting the price of a

particular article, etc. are easy to locate and initiate. The inputs are taken through

the mouse clicks.

• Project Management Techniques: This comprises of three parts, estimation

decisions, scheduling decisions and risk-oriented decisions. A particular date is set

to deliver the project. The tasks to be accomplished in the auction(project) are

sequenced based on dependencies.

•Quality Control: It encompasses a set of software engineering actions that help

to ensure work meets quality goals. The code for the online auction system

is reviewed and inspected to cover errors. A series of testing steps are applied.

•Quality Assurance: In this, a set of auditing and reporting functions that assess the

completeness of the project are encompassed.

***AGILE FRAMEWORK:***

In our project which is Online Auction System, the Scrum framework can be applied

suitably. Scrum is a framework within which people can address complex adaptive

problems, while productively and creatively delivering products of the highest

possible value. In this, small working teams are organized to "maximize

communication, minimize overhead, and maximize the strength of sharing tacit and

informal knowledge. Often thought of as an agile project management framework,

Scrum describes a set of meetings, tools, and roles that work in concert to help

teams structure and manage their work. There are various prerequisites for

choosing scrum. This employs a backlog in which there is a list of project

requirements that provide buisness value. Items can be added anytime. This makes

it customer friendly. The specifications are already taken care of in the functional

requirements and the use cases, but if there are any additional features, they can

always be added in the list of backlog. The heart of Scrum is a Sprint, a time-box of

two weeks or one month during which a potentially releasable product increment

is created. A new Sprint starts immediately after the conclusion of the previous

Sprint. During this time, the backlog items that the sprint work units address are

frozen. All the data from the use cases are implemented here. Meetings are

organized in which the modules of the system can be looked into. After the

completion of every backlog, the next accomplishments of the backlog list is

discussed. For example, after the completion of the login() module, a scrum

meeting can be held. The Sprint Retrospective occurs after the Sprint Review and

prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect

itself and create a plan for improvements to be enacted during the subsequent

Sprint.

***ARCHITECTURAL MODEL:***

In this project for the online auction system, we try to find an appropriate

architectural style, one which suits the requirements of this model. Now in our

system, we want that as soon as user gets one item (i.e. buys or bids on a

particular item), then the details of the same should be visible as changes to the

database as soon as possible and also visible to all the other potential buyers and

the seller himself. For this major requirement, we think that the data centered

architectural style fits well as it is a blackboard mechanism for each of the users

(read customers) be it a buyer or a seller.

Data Centered Architecture- A data store (e.g., a file or database) resides at the

center of this architecture and is accessed frequently by other components that

update, add, delete, or otherwise modify data within the store. Figure 9.1

illustrates a typical data-centered style. Client software accesses a central

repository. In some cases, the data repository is passive. That is, client software

accesses the data independent of any changes to the data or the actions of other

client software. Any one of them can update the value of the price of bid or

purchase the item at once, at no time can two users buy the same at once which

is exactly what we need.

Organization and Refinement-

Control-

In our particular module, the bidding module, the control

hierarchy exists as admin is at the top of the hierarchy and then it is followed by

the buyer and then the seller. It implies that the hierarchy looks somewhat like this-

accept the bid in the admin

^

|

check in database for availability

^

|

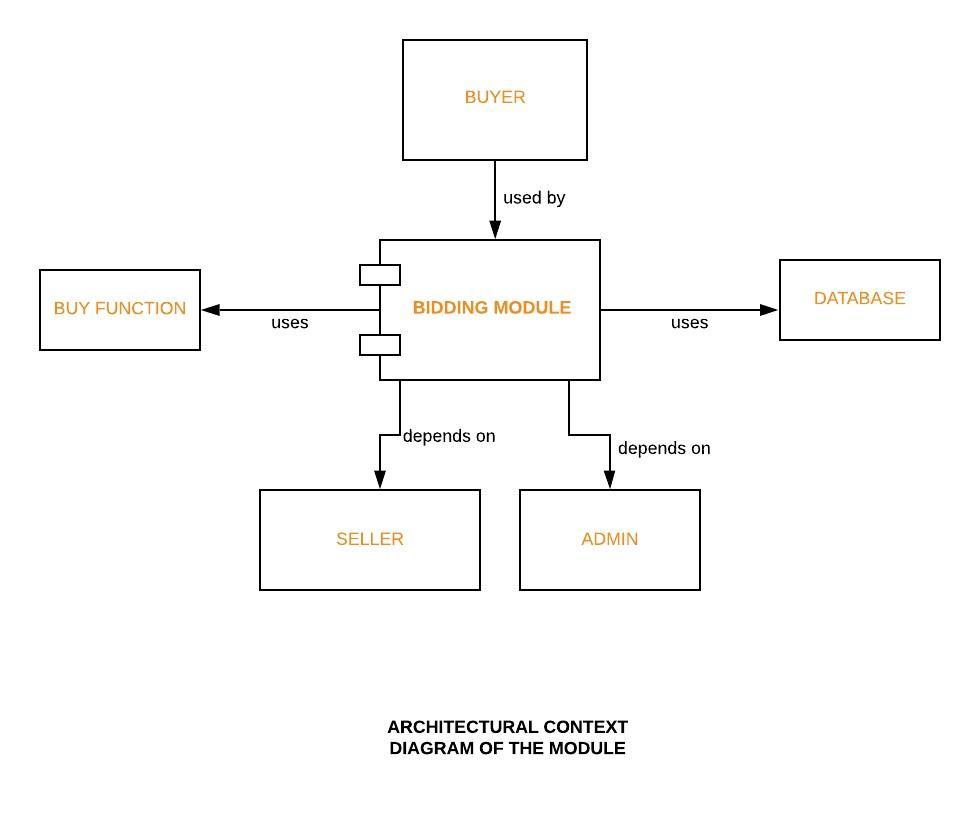
bid/buy

Data-

The communication of data is done through

the blackboard mechanism as stated in the above architectural style.

***ARCHITECTURAL CONTEXT DIAGRAM:***

******

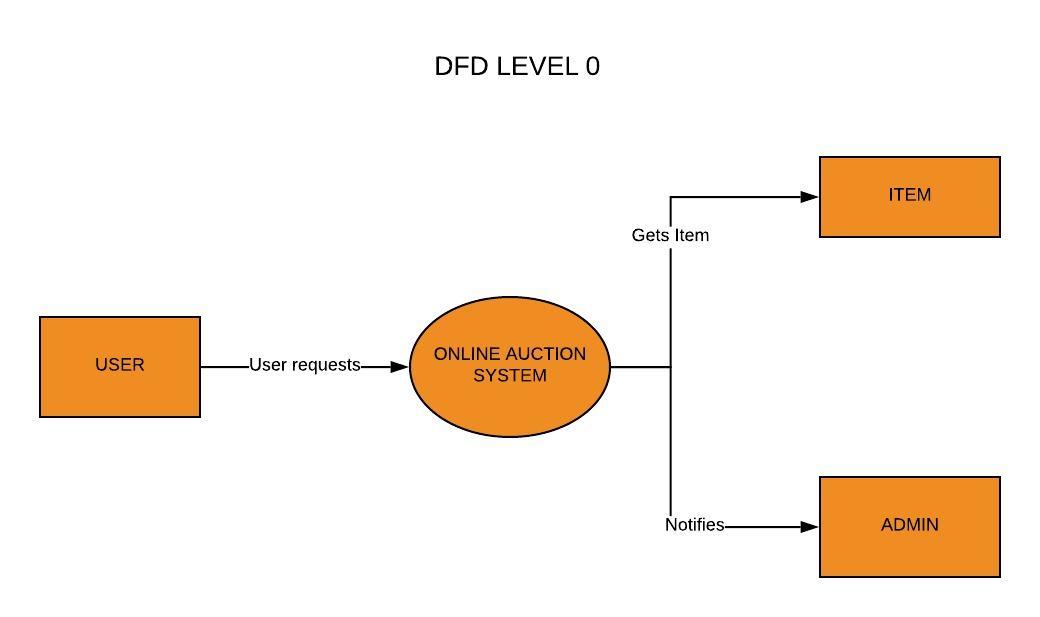
DESIGN:

DFD:

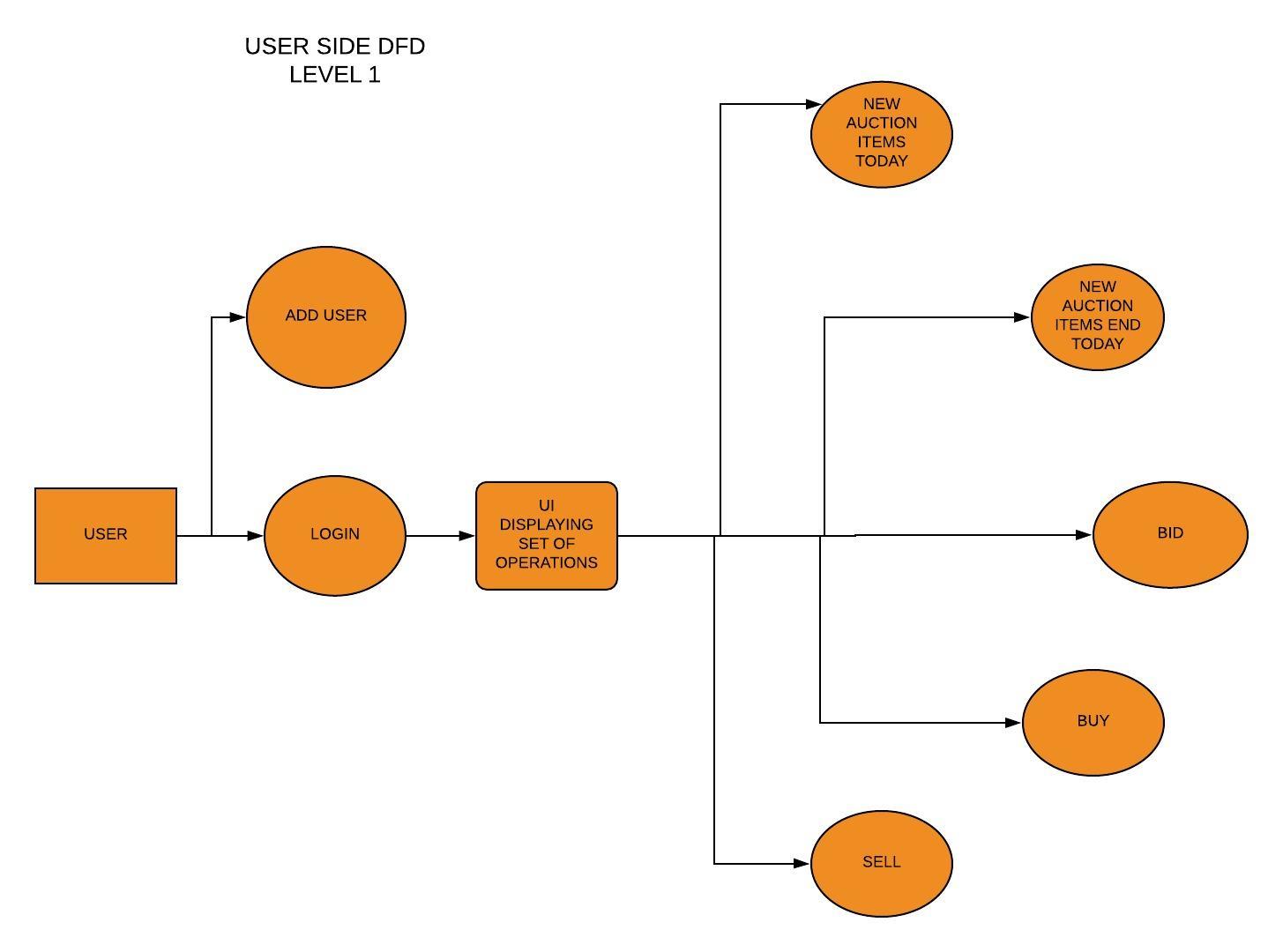
Also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

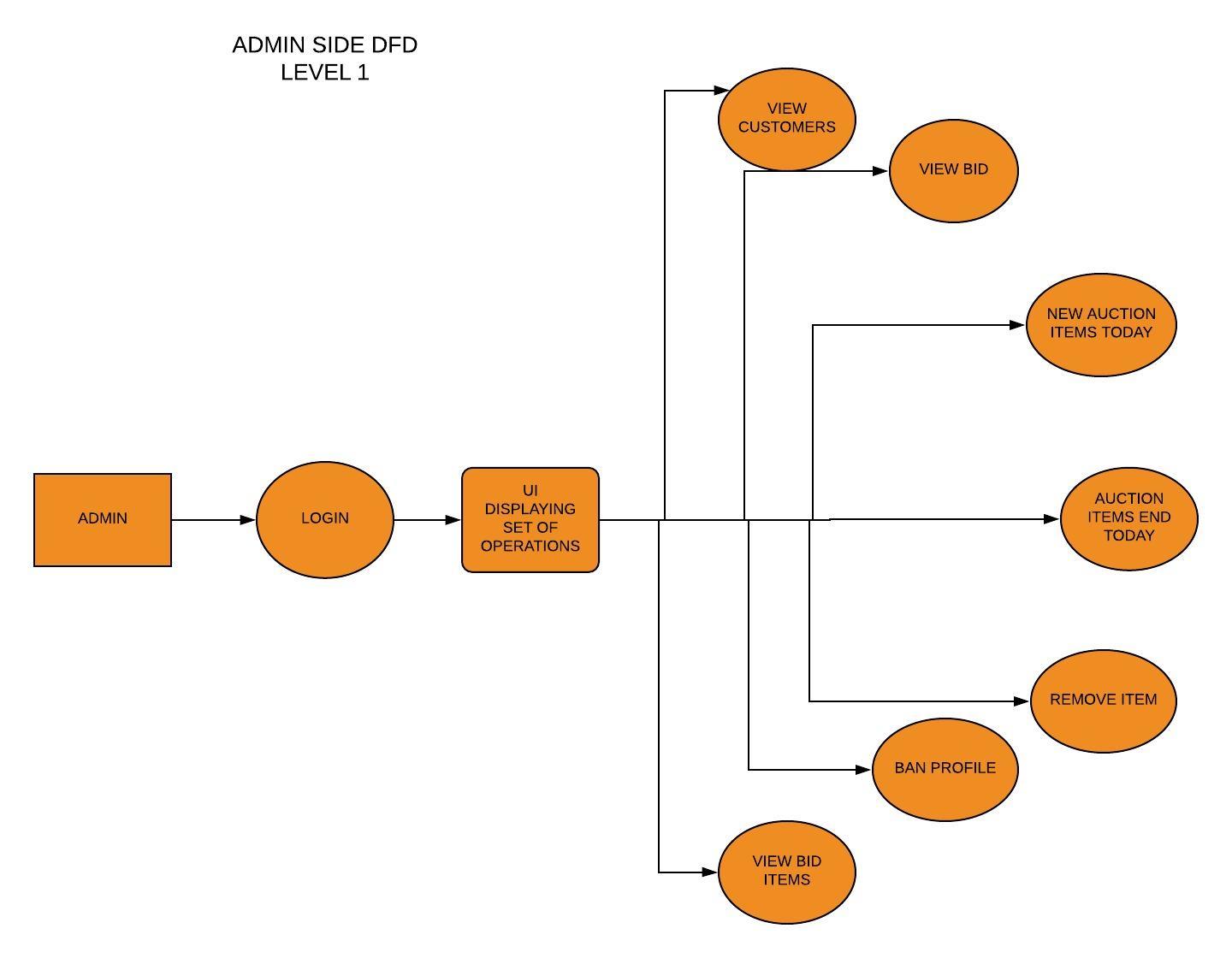
DFD (LEVEL 0):



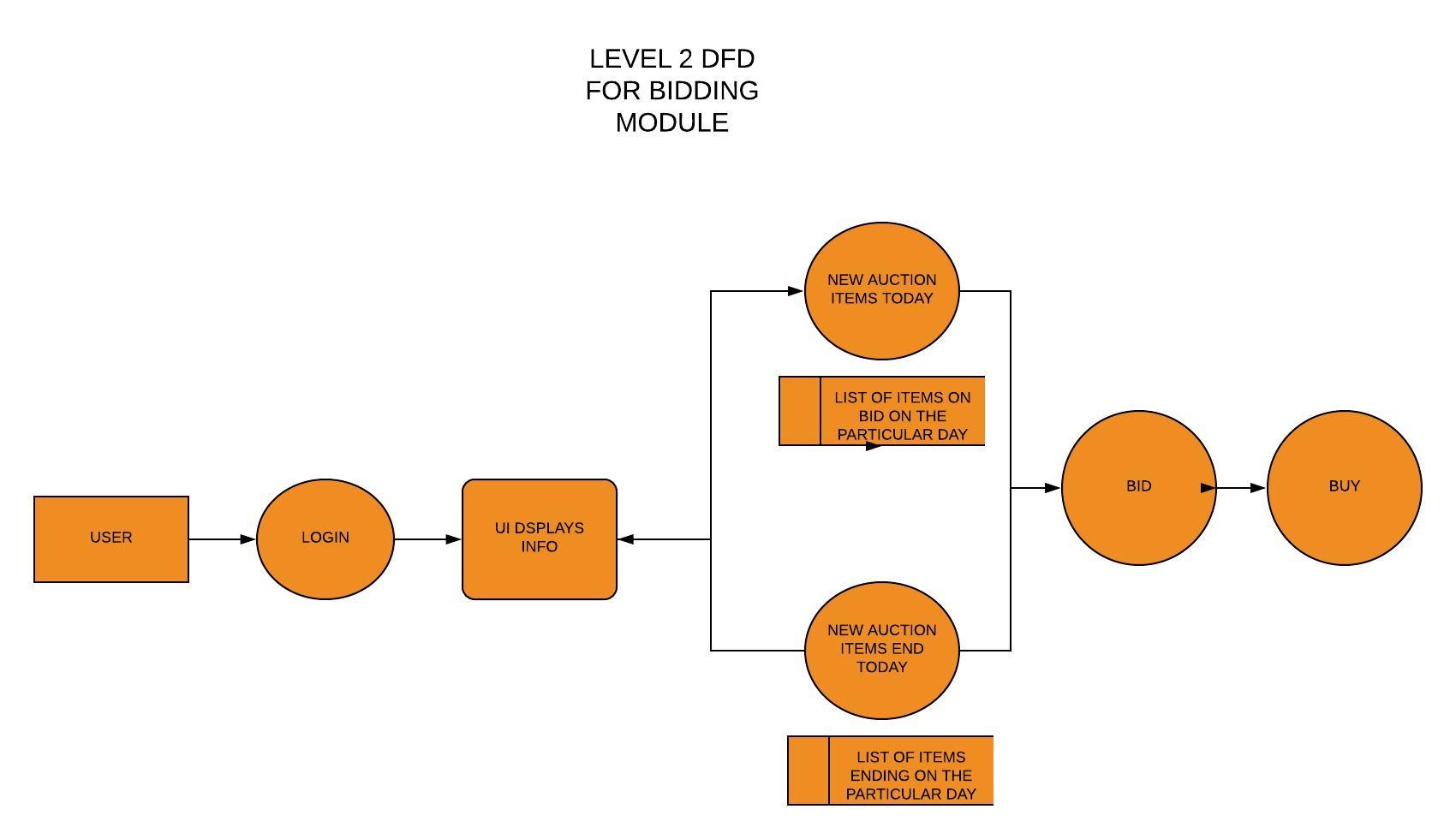
DFD- USER SIDE (LEVEL 1):



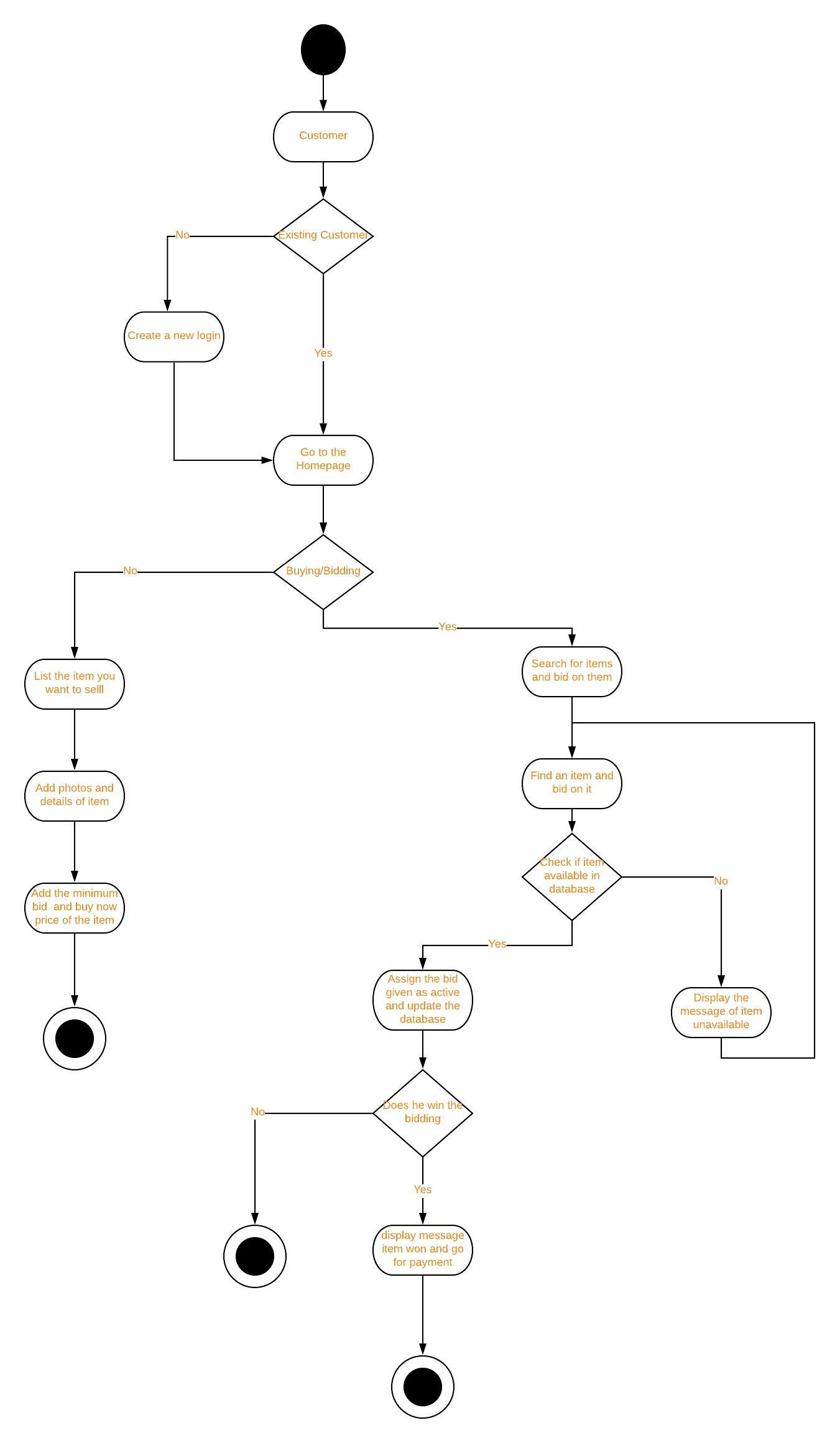
DFD- ADMIN SIDE (LEVEL 1):



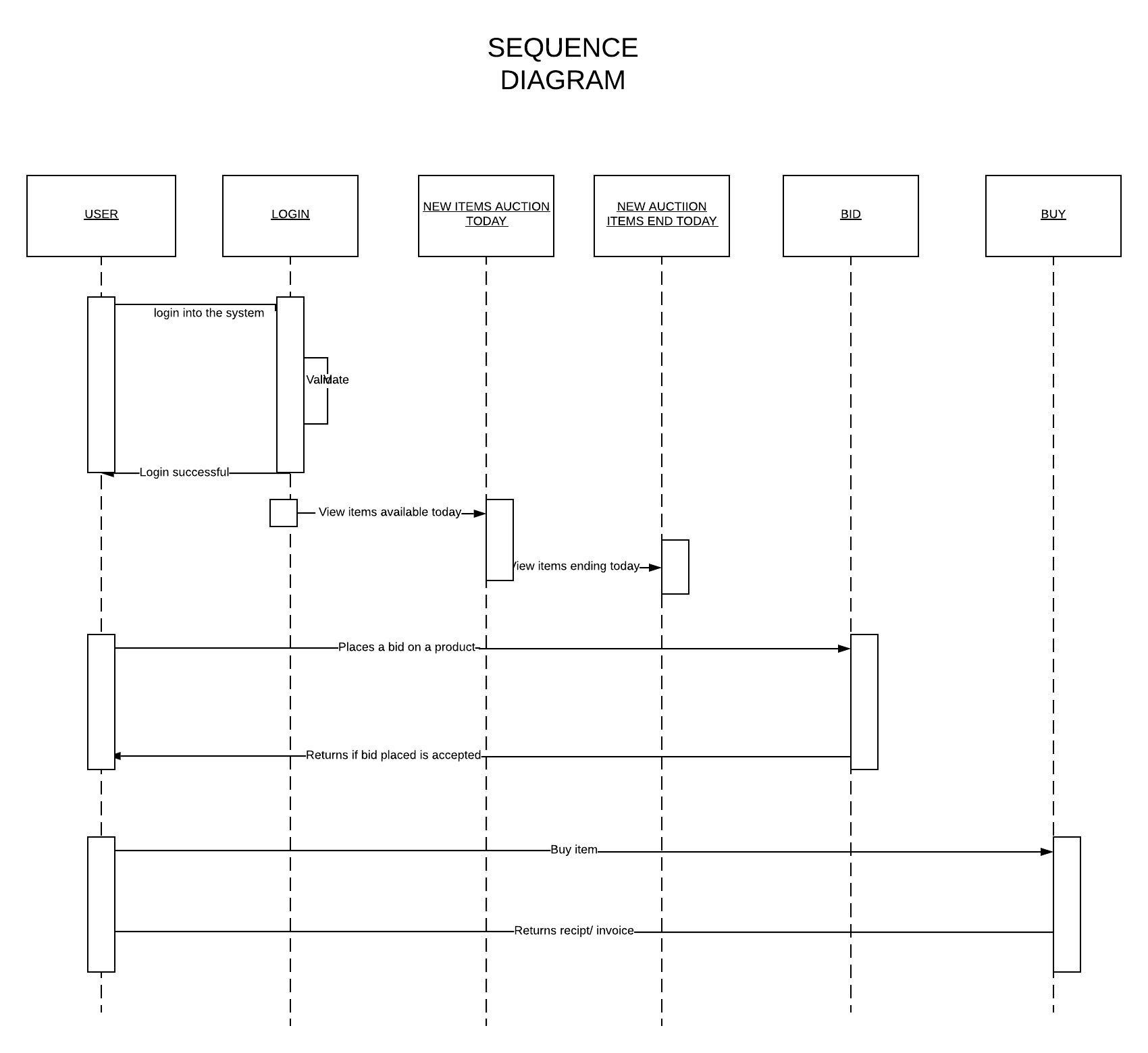
DFD FOR BIDDING MODULE (LEVEL 2):



ACTIVITY DIAGRAM:



SEQUENCE DIAGRAM:



CODE:

//1

package com.example.auction.bidding.api;

import com.fasterxml.jackson.annotation.JsonCreator;

import com.fasterxml.jackson.annotation.JsonProperty;

import lombok.Value;

import java.time.Instant;

import java.util.UUID;

/\*\*

\* A bid value object.

\*/

@Value

public final class Bid {

/\*\*

\* The user that placed the bid.

\*/

private final UUID bidder;

/\*\*

\* The time that the bid was placed.

\*/

private final Instant bidTime;

/\*\*

\* The bid price.

\*/

private final int price;

/\*\*

\* The maximum bid price;

\*/

private final int maximumPrice;

@JsonCreator

// parameter annotations needed until

https://github.com/lagom/lagom/issues/172 is fixed.

public Bid(@JsonProperty("bidder") UUID bidder, @JsonProperty("bidTime")

Instant bidTime,

@JsonProperty("price") int price, @JsonProperty("maximumPrice") int

maximumPrice) {

this.bidder = bidder;

this.bidTime = bidTime;

this.price = price;

this.maximumPrice = maximumPrice;

}

}

//2

package com.example.auction.bidding.api;

import com.fasterxml.jackson.annotation.JsonCreator;

import com.fasterxml.jackson.annotation.JsonSubTypes;

import com.fasterxml.jackson.annotation.JsonTypeInfo;

import com.fasterxml.jackson.annotation.JsonTypeName;

import lombok.Value;

import java.time.Instant;

import java.util.Optional;

import java.util.UUID;

/\*\*

\* A bid event.

\*/

@JsonTypeInfo(use = JsonTypeInfo.Id.NAME, property = "type", defaultImpl =

Void.class)

@JsonSubTypes({

@JsonSubTypes.Type(BidEvent.BidPlaced.class),

@JsonSubTypes.Type(BidEvent.BiddingFinished.class)

})

public interface BidEvent {

UUID getItemId();

/\*\*

\* A bid was placed.

\*/

@JsonTypeName("bid-placed")

@Value

final class BidPlaced implements BidEvent {

/\*\*

\* The item the bid was placed on.

\*/

private final UUID itemId;

/\*\*

\* The bid itself.

\*/

private final Bid bid;

@JsonCreator

public BidPlaced(UUID itemId, Bid bid) {

this.itemId = itemId;

this.bid = bid;

}

}

/\*\*

\* Bidding finished.

\*/

@JsonTypeName("bidding-finished")

@Value

final class BiddingFinished implements BidEvent {

/\*\*

\* The item that finished bidding.

\*/

private final UUID itemId;

/\*\*

\* The winning bid, if there was one.

\*/

private final Optional<Bid> winningBid;

@JsonCreator

public BiddingFinished(UUID itemId, Optional<Bid> winningBid) {

this.itemId = itemId;

this.winningBid = winningBid;

}

}

}

//3

package com.example.auction.bidding.api;

import com.fasterxml.jackson.annotation.JsonCreator;

import lombok.Value;

import java.util.UUID;

/\*\*

\* The result of placing a bid.

\*/

@Value

public final class BidResult {

/\*\*

\* The current bid price.

\*/

private final int currentPrice;

/\*\*

\* The status of the result.

\*/

private final BidResultStatus status;

/\*\*

\* The current winning bidder.

\*/

private final UUID currentBidder;

@JsonCreator

public BidResult(int currentPrice, BidResultStatus status, UUID currentBidder) {

this.currentPrice = currentPrice;

this.status = status;

this.currentBidder = currentBidder;

}

}

//4

package com.example.auction.bidding.api;

/\*\*

\* The status of the result of placing a bid.

\*/

public enum BidResultStatus {

/\*\*

\* The bid was accepted, and is the current highest bid.

\*/

ACCEPTED,

/\*\*

\* The bid was accepted, but was outbidded by the maximum bid of the current

highest bidder.

\*/

ACCEPTED\_OUTBID,

/\*\*

\* The bid was accepted, but is below the reserve.

\*/

ACCEPTED\_BELOW\_RESERVE,

/\*\*

\* The bid was not at least the current bid plus the increment.

\*/

TOO\_LOW,

/\*\*

\* The auction hasn't started.

\*/

NOT\_STARTED,

/\*\*

\* The auction has already finished.

\*/

FINISHED,

/\*\*

\* The auction has been cancelled.

\*/

CANCELLED

}

//5

package com.example.auction.bidding.api;

import static com.lightbend.lagom.javadsl.api.Service.named;

import static com.lightbend.lagom.javadsl.api.Service.pathCall;

import static com.lightbend.lagom.javadsl.api.Service.topic;

import akka.NotUsed;

import com.example.auction.security.SecurityHeaderFilter;

import com.lightbend.lagom.javadsl.api.Descriptor;

import com.lightbend.lagom.javadsl.api.Service;

import com.lightbend.lagom.javadsl.api.ServiceCall;

import com.lightbend.lagom.javadsl.api.broker.Topic;

import com.lightbend.lagom.javadsl.api.deser.PathParamSerializers;

import org.pcollections.PSequence;

import java.util.UUID;

/\*\*

\* The bidding services.

\*

\* This services manages all bids and lifecycle events associated with them.

\*

\* An auction is created when an AuctionStarted event is received, then bids can

be placed, and when the end date

\* specified in AuctionStarted is reached, this service will published a

BiddingFinished event with the winning

\* bidder (if there was one).

\*/

public interface BiddingService extends Service {

String TOPIC\_ID = "bidding-BidEvent";

/\*\*

\* A place a bid.

\*

\* @param itemId The item to bid on.

\*/

ServiceCall<PlaceBid, BidResult> placeBid(UUID itemId);

/\*\*

\* Get the bids for an item.

\*

\* @param itemId The item to get the bids for.

\*/

ServiceCall<NotUsed, PSequence<Bid>> getBids(UUID itemId);

/\*\*

\* The bid events topic.

\*/

Topic<BidEvent> bidEvents();

@Override

default Descriptor descriptor() {

return named("bidding").withCalls(

pathCall("/api/item/:id/bids", this::placeBid),

pathCall("/api/item/:id/bids", this::getBids)

).withTopics(

topic(TOPIC\_ID, this::bidEvents)

).withPathParamSerializer(UUID.class, PathParamSerializers.required("UUID",

UUID::fromString, UUID::toString))

.withHeaderFilter(SecurityHeaderFilter.INSTANCE);

}

//6

package com.example.auction.bidding.api;

import com.fasterxml.jackson.annotation.JsonCreator;

import lombok.Value;

/\*\*

\* A request to place a bid.

\*/

@Value

public final class PlaceBid {

/\*\*

\* The maximum bid price.

\*/

private final int maximumBidPrice;

@JsonCreator

public PlaceBid(int maximumBidPrice) {

this.maximumBidPrice = maximumBidPrice;

}

}

TESTING:

Software Testing is the process used to help identify the correctness,

completeness, security, and quality of developed computer software. Testing is a

process of technical investigation, performed on behalf of stakeholders, that is

intended to reveal quality-related information about the product with respect to

the context in which it is intended to operate. This includes, but is not limited to,

the process of executing a program or application with the intent of finding

errors. Quality is not an absolute; it is value to some person. With that in mind,

testing can never completely establish the correctness of arbitrary computer

software; testing furnishes a criticism or comparison that compares the state and

behavior of the product against a specification. An important point is that

software testing should be distinguished from the separate discipline of Software

Quality Assurance (SQA), which encompasses all business process areas, not just

testing. There are many approaches to software testing, but effective testing of

complex products is essentially a process of investigation, not merely a matter of

creating and following routine procedure. One definition of testing is "the process

of questioning a product in order to evaluate it", where the "questions" are

operations the tester attempts to execute with the product, and the product

answers with its behavior in reaction to the probing of the tester. Although

most of the intellectual processes of testing are nearly identical to that of review

or inspection, the word testing is connoted to mean the dynamic analysis of the

product—putting the product through its paces. Some of the common quality

attributes include capability, reliability, efficiency, portability, maintainability,

compatibility and usability. A good test is sometimes described as one which

reveals an error; however, more recent thinking suggests that a good test is one

which reveals information of interest to someone who matters within the project

community. In general, software engineers distinguish software faults from

software failures. In case of a failure, the software does not do what the user

expects. A fault is a programming error that may or may not actually manifest as a

failure. A fault can also be described as an error in the correctness of the semantic

of a computer program. A fault will become a failure if the exact computation

conditions are met, one of them being that the faulty portion of computer

software executes on the CPU. A fault can also turn into a failure when the

software is ported to a different hardware platform or a different compiler, or

when the software gets extended. Software testing is the technical investigation

of the product under test to provide stakeholders with quality related

information. Software testing may be viewed as a sub-field of Software Quality

Assurance but typically exists independently (and there may be no SQA areas in

some companies). In SQA, software process specialists and auditors take a

broader view on software and its development. They examine and change the

software engineering process itself to reduce the amount of faults that end up in

the code or deliver faster

***BLACK BOX TESTING:***

Black-box testing, also called behavioral testing, focuses on the functional

requirements of the software. That is, black-box testing techniques enable you to

derive sets of input conditions that will fully exercise all functional requirements

for a program. Black-box testing is not an alternative to white-box techniques.

Rather, it is a complementary approach that is likely to uncover a different class of

errors than white-box methods.

Black-box testing attempts to find errors in the following categories:

(1) incorrect or missing functions

(2) interface errors

(3) errors in data structures or external database access

(4) behavior or performance errors

(5) initialization and termination errors.

Unlike white-box testing, which is performed early in the testing process, black-

box testing tends to be applied during later stages of testing. Because black-box

testing purposely disregards control structure, attention is focused on the

information domain.

The type of black-box testing we are focusing is:

* Equivalence Partitioning: It is a software test design technique that

involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.

In our bidding module we can have many paths as already seen in the code along

with the activity diagram, the sequence diagram, the level 2 DFD and the use

cases.

|  |  |  |  |
| --- | --- | --- | --- |
| TEST CASES | VARIABLE NAME | EXPECTED OUTPUT | RESULT |
| Char value in int | price | throw an error | executed successfully |
| Int value in int | maximumPrice | no error | executed successfully |
| Float value in int | bidTime | throw an error | executed successfully |
| Int value in float | bid | no error | executed successfully |
| Char value in float | bid | throw an error | executed successfully |
| Float value in float | bid | no error | executed successfully |
| Int value in char | status | throw an error | executed successfully |
| Float value in char | status | throw an error | executed successfully |
| Char value in char | status | no error | executed successfully |

Above are all the possible test cases are in the given module.

Each of these paths were checked and there is no error found in the specific module.

Test Cases 9

Successful Test Cases 9

Success % 100

The black box testing for the bidding module has been completed...

***WHITE BOX TESTING:***

White Box Testing also known as Clear Box Testing, Open Box Testing, Glass Box

Testing, Transparent Box Testing, Code-Based Testing or Structural Testing is a

software testing method in which the internal structure /design/implementation

of the item being tested is known to the tester. The tester chooses inputs to

exercise paths through the code and determines the appropriate outputs.

Programming know-how and the implementation knowledge is essential. White-

box testing is testing beyond the user interface and into the nitty-gritty of a

system.

Applying the white box testing to the Bidding module-

In our bidding module we can have many

paths as already seen in the code along with the activity diagram, the sequence

diagram, the level 2 DFD and the use cases.

|  |  |  |
| --- | --- | --- |
| EXPLORING ALL THE POSSIBLE PATHS OF THE MODULE | EXPECTED OUTPUT | RESULT |
| User enters the module but doesn't like anything and exits | no message displayed | Executed successfully |
| User enters the module and bids on a specific item and exits | message depends on result | Executed successfully |
| Bid is countered by a higher bid and the user loses | lost bid message | Executed successfully |
| User comes online and bids again rather than losing | bid completed message | Executed successfully |
| User wins straightaway from the first bid itself | bid won message | Executed successfully |
| User wins but exits from the third-party payment app | payment failed message | Executed successfully |
| User pays the money and gets the payment deducted | payment successful message | Executed successfully |

Above are all the possible paths the user can take in the given module.

Each of these paths were checked and there is no error found in the specific module.

Test Cases 7

Successful Test Cases 7

Success % 100

The white box testing for the Bidding module has been completed...

***SOFTWARE REVIEW:***

After completion of the online auction system, this is how we propose

someone on a higher level than us should review our project-

What is a software review?

In the terms of Roger S. Pressman it is a group of diverse people-

1. Trying to improve the software by giving suggestions of additions in the

software that appeal to them more.

2. Confirm those parts of the software that are not needed or can be removed.

3. Achieve technical work of more uniform, or at least more predictable, quality than can be achieved without reviews, in order to make technical work more manageable.

Reviewing the defect amplification will be the best way, because it not only tells

us about the errors but also about the things that affected by the changes in our

code and how adversely they get affected is also important.

Thus, the best way to review our model would be the defect amplification.

***CONCLUSION:***

This paper described a case study highlighting the best practices in designing and building a

web-based auction system. We designed the auction system using UML. The Use Case Diagram,

Sequence Diagram, Class Diagram and Component Diagram offered by UML were used

successfully during the process. Our implementation, with its basis in component-based

programming enabled us to develop a highly maintainable system with a number of re-usable

components. Further, the system used intelligent agents that permitted fair help to bidders

participating in auctions, and at the same time, achieved maximum profit for the seller. Again,

the implementation environment and the tools used, provided excellent support for the

successful development of the system. The approach outlined here was more effective in

implementing our auction system than the existing Information Engineering (data-oriented),

Structured Development (function-oriented), or Object-oriented (data-oriented and function-

oriented) methodology. Although we only made a few specific changes to the components,

these changes indicate that subsequent changes to other system components will be

straightforward. Consequently, the reusability of the system was facilitated and, as a direct

result, we expect that the system will be easily able to suitably evolve in the fast-changing

Internet environment.