LONDON METROPOLITAN UNIVERSITY

CS7081Enterprise Software Components and Systems Development

Coursework Assessment (2017-18)

Name: Khalid Ahmed Mohamed

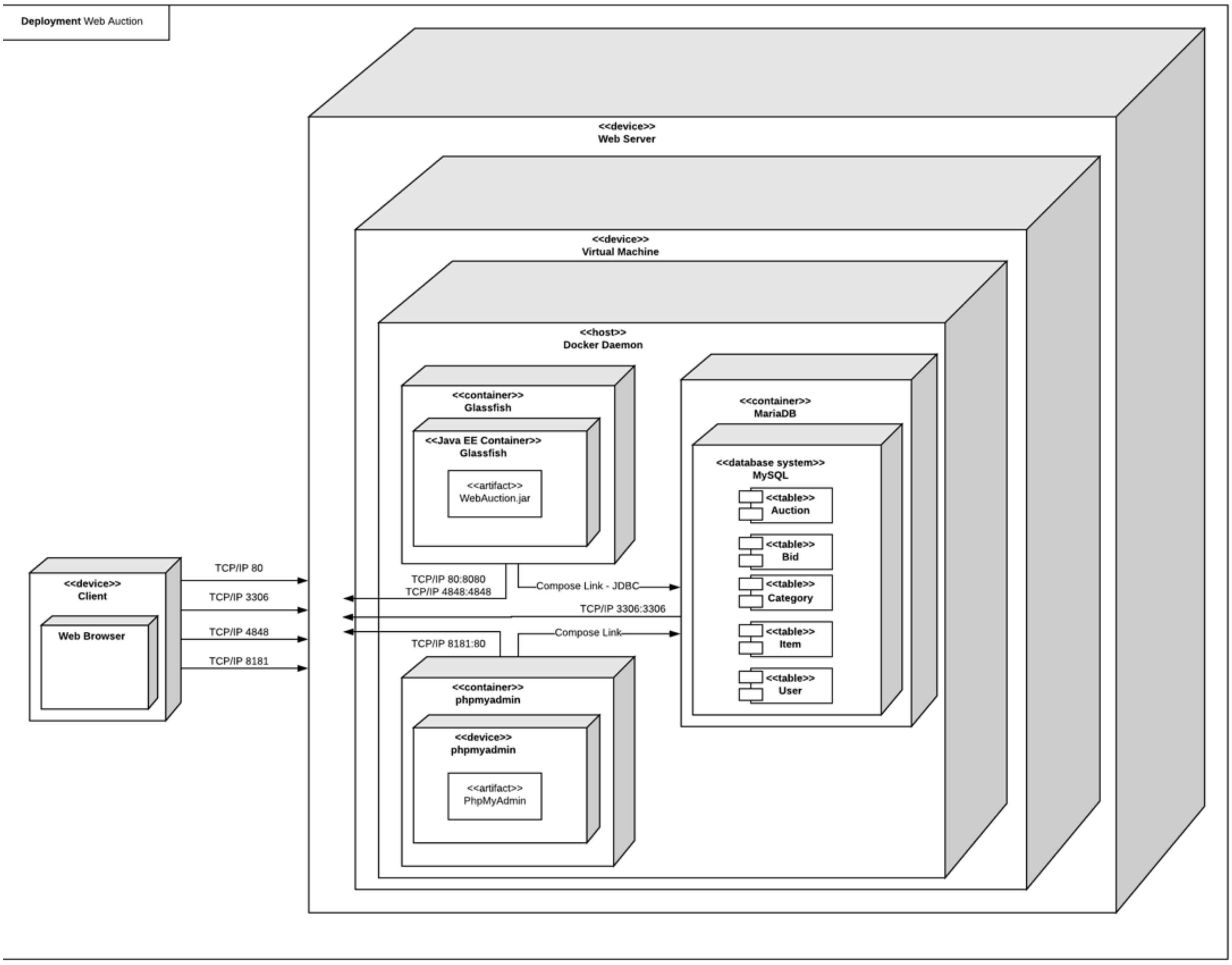
ID: 13036087

Table of Contents

1. Software Architecture

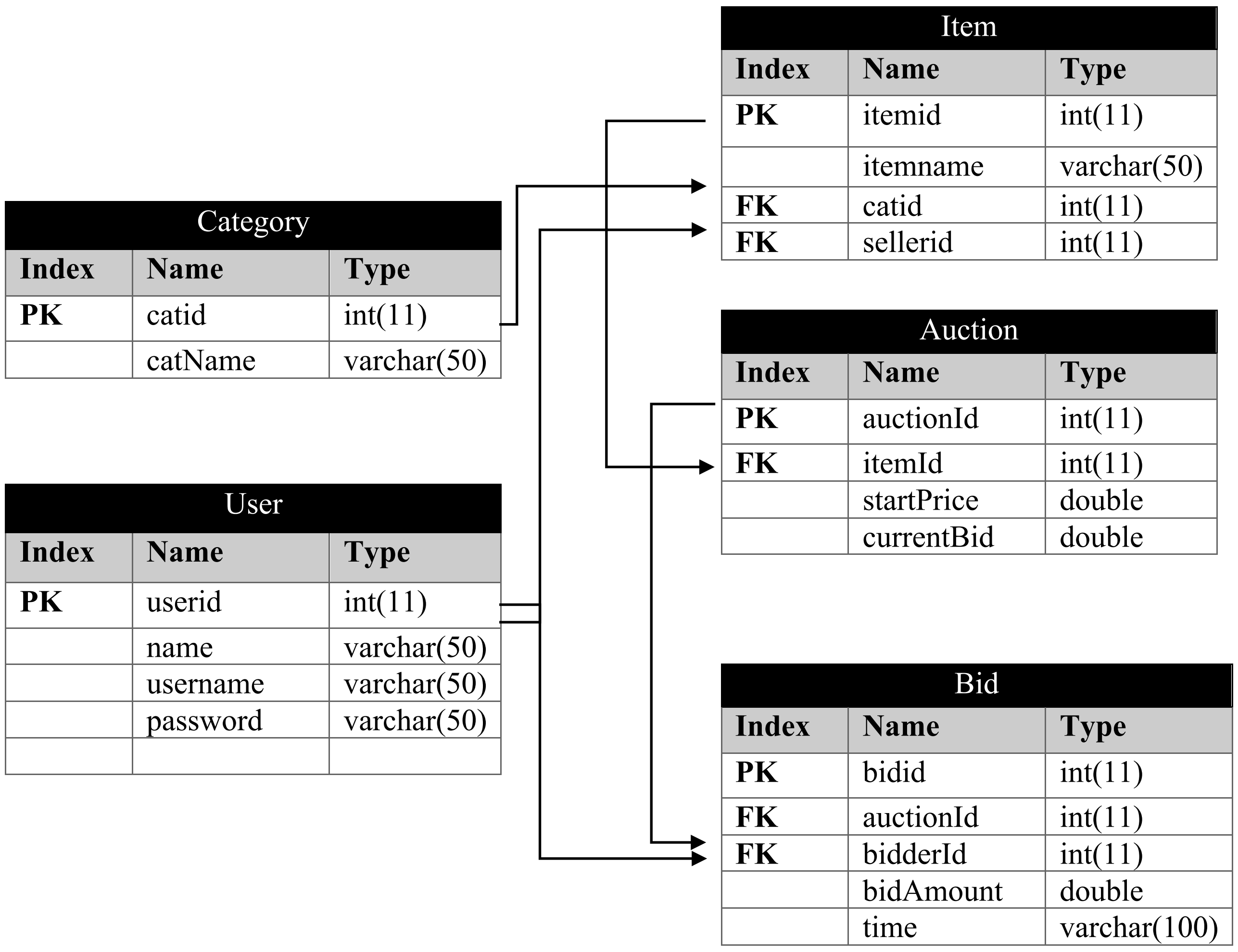
1.1 UML Deployment Diagram

Software Architecture is foundation or fundamental design of the software. One of the main reason why the Software Architecture is important is because without it, it could be difficult to maintain or change once built due to a lack of knowledge.



2. Database Design

Database Design is the process of creating the structure of the database, identifying the tables, column, input types and the relationship between the tables.



Item

Index	Name	Type
PK	itemid	int(11)
	itemname	varchar(50)
FK	catid	int(11)
FK	sellerid	int(11)

Category

Index	Name	Type
PK	catid	int(11)
	catName	varchar(50)

Auction

Index	Name	Type
PK	auctionId	int(11)
FK	itemId	int(11)
	startPrice	double
	currentBid	double

User

Index	Name	Type
PK	userid	int(11)
	name	varchar(50)
	username	varchar(50)
	password	varchar(50)

Bid

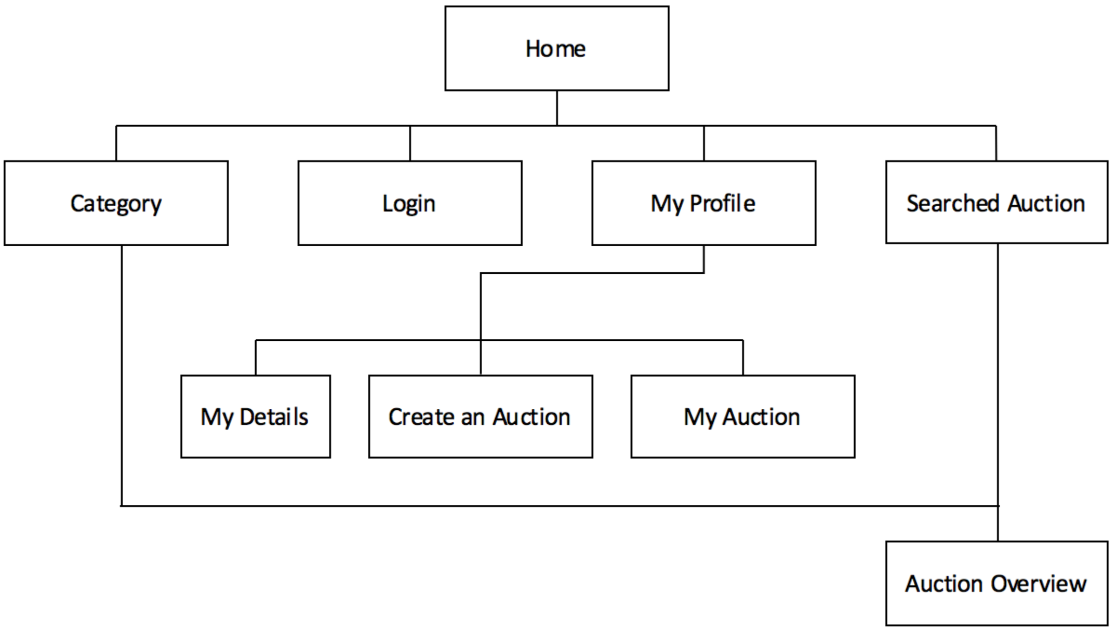
Index	Name	Type
PK	bidid	int(11)
FK	auctionId	int(11)
FK	bidderId	int(11)
	bidAmount	double
	time	varchar(100)

2.1 SQL DDL Script

1) **-- Table structure for table `User`** -- CREATE TABLE User (userid int (11) NOT NULL AUTO_INCREMENT, name VARCHAR (50) NOT NULL, username VARCHAR (20) NOT NULL, password VARCHAR (20) NOT NULL, PRIMARY KEY (userid));	4) **-- Table structure for table `Auction`** -- CREATE TABLE Auction (auctionId int (11) NOT NULL AUTO_INCREMENT, itemId int (11) NOT NULL, startPrice DOUBLE NOT NULL, currentBid DOUBLE NOT NULL, PRIMARY KEY (auctionId), FOREIGN KEY (itemId) REFERENCES Item (itemid));
2) **-- Table structure for table `Category`** -- CREATE TABLE Category (catid int (11) NOT NULL AUTO_INCREMENT, catName VARCHAR (50) NOT NULL, PRIMARY KEY (catid));	5) **-- Table structure for table `Bid`** -- CREATE TABLE Bid (bidId int (11) NOT NULL AUTO_INCREMENT, auctionId int (11) NOT NULL, bidderId int (11) NOT NULL, bidAmount DOUBLE NOT NULL, time VARCHAR (50) NOT NULL, PRIMARY KEY (bidId), FOREIGN KEY (auctionId) REFERENCES Auction (auctionId), FOREIGN KEY (bidderId) REFERENCES User (userid));
3) **-- Table structure for table `Item`** -- CREATE TABLE Item (itemid int (11) NOT NULL AUTO_INCREMENT, itemname VARCHAR (50) NOT NULL, catid int (11) NOT NULL, sellerid int (11) NOT NULL, PRIMARY KEY (itemid), FOREIGN KEY (catid) REFERENCES Category (catid), FOREIGN KEY (sellerid) REFERENCES User (userid));	

3. Website Design

3.1 Page Navigation Map



4. Program Description

4.1 Use Case Model

4.1.1 High Level Use Case

A high-level use case is used to give a general description of every process within the system in plain simple English so that it is understandable to users.

---

1. **Log into Web Application**

   Actors:
   Users

   Type:
   Primary

   Description:
   The users will be able to log into the system with the credentials that will need to provide. The user is required to enter a valid username and password to gain access into the web application. Once submitted, the credential is validated. If the credential is valid, the user will move back to the home page of the web application with access to make bid and create auctions, otherwise a pop up will appear notifying the user of the invalid credential.

---

2. **Create an Auction**

   Actors:
   User

   Type:
   Primary

   Description:
   The user will have the ability to create an auction once logged into the web application. The web application will ask the user to fill out a form regarding the description of the item and the start price for the auction. Once submitted, the web application will add the item to the database and automatically start the auction for that specific item.

---

3. **Search for an Auction**

   Actors:
   Users

   Type:
   Primary

   Description:
   The users will have the ability to search for any auctions that are currently live. The user is required to enter what they are looking for in the search bar. The web application will retrieve any auctions that fit what the user has entered. The user can then click on any of the searched results to provide even more information about that auction.

4. **Make a bid**

   Actors:
   User

   Type:
   Primary

   Description:
   The user will have the ability to make a bid on any auction once they have logged into the web application. Once the user is on an auction, they can place a bid by entering a figure into the bid form. Once submitted, the system will first check whether the user is logged in. Once confirmed, the system will secondly check if the current bid that the user sees on the web application is consistence with the current bid in the database. Once confirmed, the system will thirdly check if the user's bid is greater than the current bid. Once all confirmed, the system will update the current bid.

5. **Remove an Auction**

   Actors:
   Users

   Type:
   Primary

   Description:
   The user will have the ability to remove an auction which they have created and no other. The user is required to go to the My Auction page which will display all of their live auction. The user will have an option beside every auction that will allow them to remove the auction from the database.

4.1.2 Expanded Use Case

1. **Log into Web Application**

Actors:
Users

Purpose:
To check the credential that was provided by the user and allocate them the appropriate permission.

Overview:
The user will have the ability to log into the system. The user will enter their credentials and once submitted, the system will check the credentials and either grant or deny the user access into the system.

Type:
Primary

Dialogue:

Actors	**System**
1. The actor will enter their credentials into the system and click the login button.	
	2. The system will retrieve the credentials provided and check it against the records in the database.
	3. If the credentials are valid, the user will be sent to the main screen, otherwise an error will appear.

2. Create an Auction

Actors:
Users

Purpose:
The user will have the ability to create an auction on the web application.

Overview:
The user will have the ability to create an auction once logged into the web application. The web application will ask the user to fill out a form regarding the description of the item and the start price for the auction. Once submitted, the web application will add the item to the database and automatically start the auction for that specific item.

Type:
Primary

Dialogue:

Actors	**System**
1. The actor will need to navigate to the create auction page via the menu.	
	2. The system will direct the actor to the create auction page.
3.The actor will need to fill in all of the fields and then click submit.	
	4.The system will check whether all of the required are filled in.
	5.The system will send the details to the database.

3. Search for an auction

Actors:
Users

Purpose:
To search for auctions in the system.

Overview:
The users will have the ability to search for any auctions that are currently live. The user is required to enter what they are looking for in the search bar. The web application will retrieve any auctions that fit what the user has entered. The user can then click on any of the searched results to provide even more information about that auction.

Type:
Primary

Dialogue:

Actors	System
1.The actor will need to enter the item into the field and click search.	
	2.The system will search for anything that is similar to what was search in the database. If found, the system will display the results.
3.The actor can click on an entry in the search result to provide more information.	
	4.The system will retrieve the extra data from the database and display it.

4. Make a bid

Actors:
Users

Purpose:
To make a bid on an auction.

Overview:
The user will have the ability to make a bid on any auction once they have logged into the web application. Once the user is on an auction, they can place a bid by entering a figure into the bid form. Once submitted, the system will first check whether the user is logged in. Once confirmed, the system will secondly check if the current bid that the user sees on the web application is consistence with the current bid in the database. Once confirmed, the system will thirdly check if the user's bid is greater than the current bid. Once all confirmed, the system will update the current bid.

Type:
Primary

Dialogue:

Actors	**System**
1.The actor will currently be on the auction overview page for a specific auction. 2.The actor will enter a bid amount in the bid field and click submit.	
	3.The system will check if the user is logged in. If not, the user will be directed back to the log in page. 4.The system will then check if the current bid on the user's web application is consistence with the current bid in the database. If not, the user will be notified that a bid has already been made by another user. 5.The system will then check if the user's bid is greater than the current bid. If not, the user will be notified. 6.Once all of the steps has been confirmed, the user's bid will be added and the user will be notified.

5. Remove an auction

Actors:
Admin

Purpose:
To remove an auction from the system.

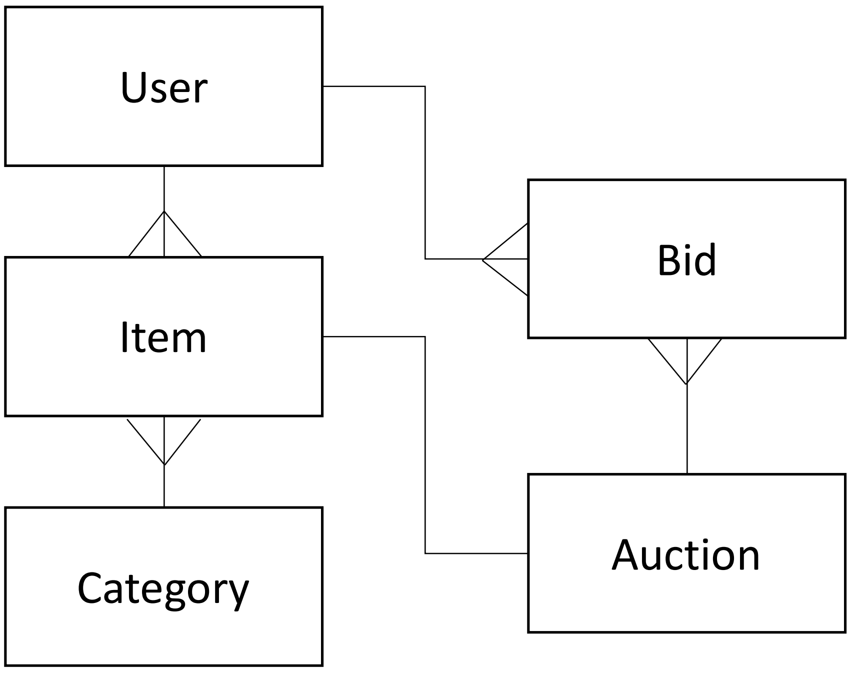
Overview:
The user will have the ability to remove an auction which they have created and no other. The user is required to go to the My Auction page which will display all of their live auction. The user will have an option beside every auction that will allow them to remove the auction from the database.

Type:
Primary

Dialogue:

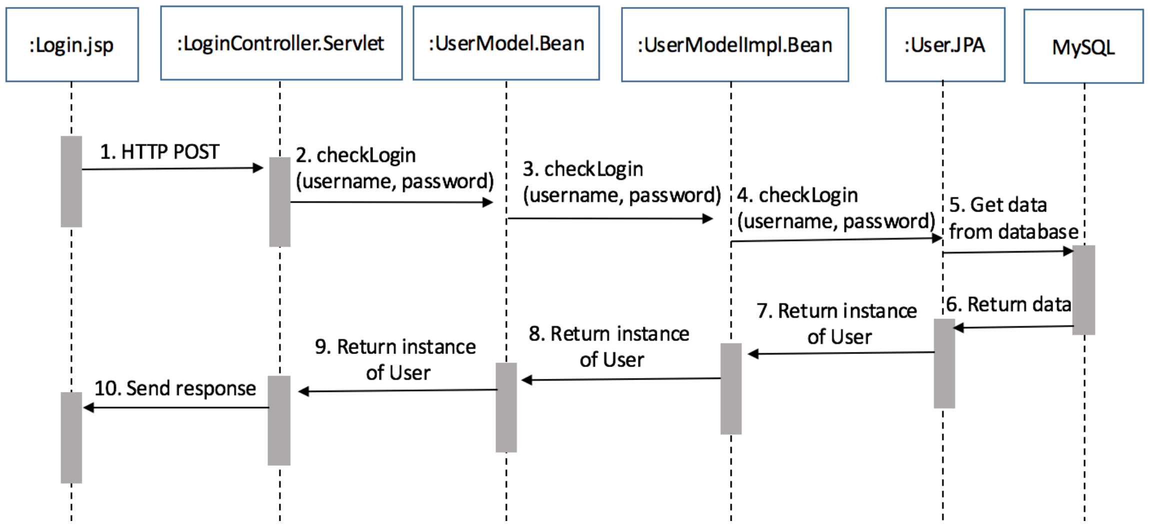
Actors	System
1.The actor will need to navigate to the My Auction page via the menu.	
	2.The system will direct the actor to the My Auction page.
	3.The system will retrieve all of the user's live auctions from the database.
4.The actor will select the auction they wish to remove and click on the remove option.	
	5.The system will get the id of the auction the user wants to remove and deletes it from the database.

4.2 Entity Relationship Diagram

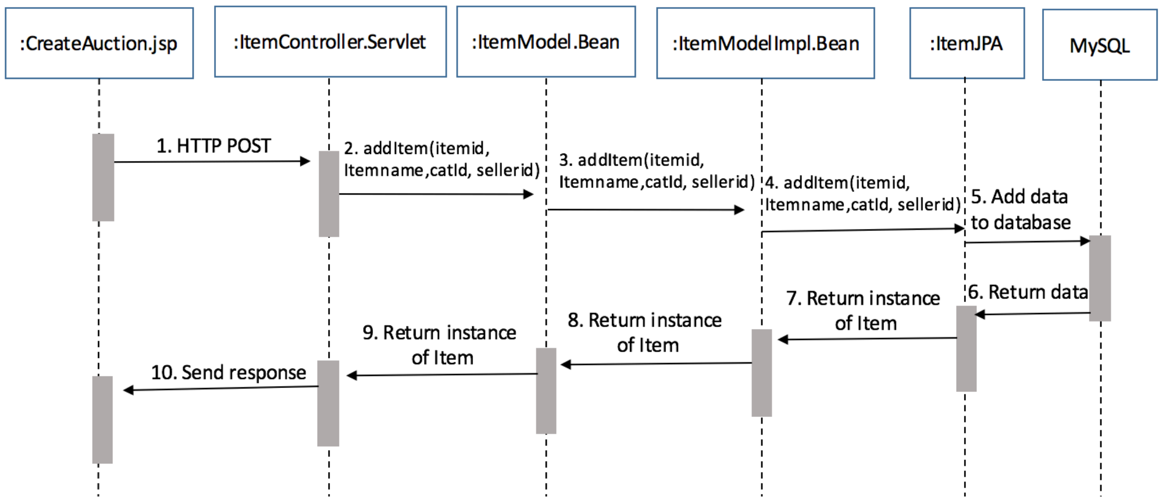


4.3 Sequence Diagram

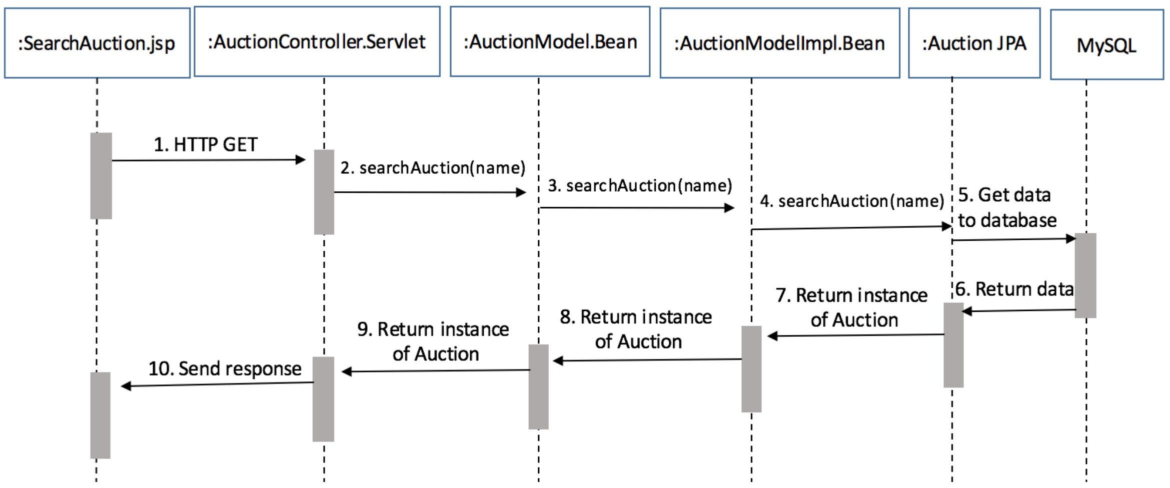
4.3.1 Log In


:Login.jsp
:LoginController.Servlet
:UserModel.Bean
:UserModelImpl.Bean
:UserJPA
MySQL
1. HTTP POST
2. checkLogin (username, password)
3. checkLogin (username, password)
4. checkLogin (username, password)
5. Get data from database
6. Return data
7. Return instance of User
8. Return instance of User
9. Return instance of User
10. Send response

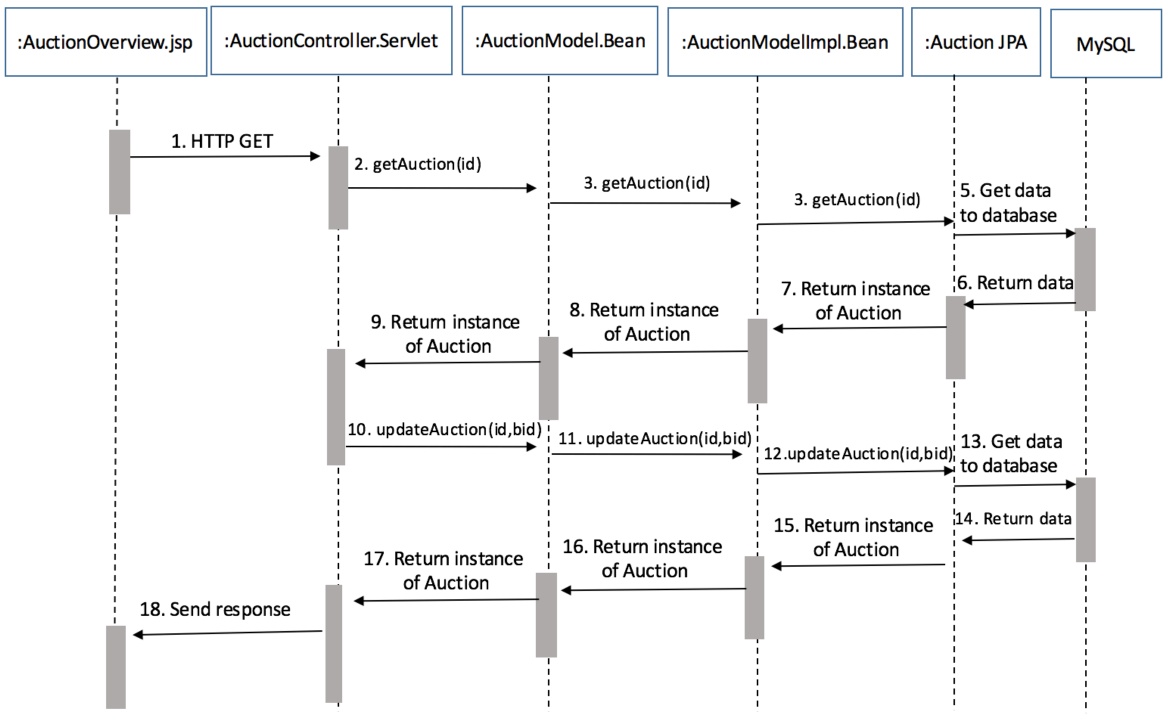
4.3.2 Create an Auction


:CreateAuction.jsp
:ItemController.Servlet
:ItemModel.Bean
:ItemModelImpl.Bean
:ItemJPA
MySQL
1. HTTP POST
2. addItem(itemid, Itemname,catId, sellerid)
3. addItem(itemid, Itemname,catId, sellerid)
4. addItem(itemid, Itemname,catId, sellerid)
5. Add data to database
6. Return data
7. Return instance of Item
8. Return instance of Item
9. Return instance of Item
10. Send response

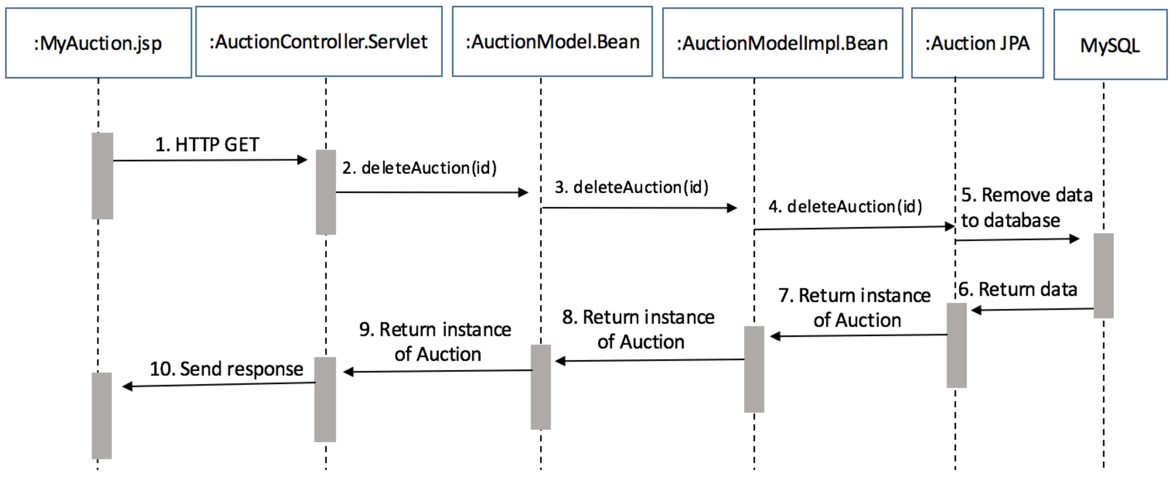
4.3.3 Search for an Auction



4.3.4 Make a bid



4.3.5 Remove an Auction



4.4 Analysis Class Diagram

4.4.1 Package: webAuctionException

4.4.1.1 Class: WebAuctionException

WebAuctionException

WebAuctionException()
WebAuctionException(String msg)

4.4.2 Package: model.user

4.4.2.1 Class: User

User
serialVersionUID Userid: String Name: String Username: String Password: String
User(String userid, String name, String username, String password) User() getName():String setName(String name):void getUsername():String setUsername(String username) :void getPassword():String setPassword(String password) :void getUserid():String toString() : String

4.4.2.2 Interface: UserModel

UserModel
checkLogin(String name, String password):User[] getUserDetails(String id):User[]

4.4.2.3 Class: UserModelImpl

UserModelImpl
emgr: EntityManager instance: UserModel
UserModelImpl() getInstance(): UserModel checkLogin(String name, String password):User[] getUserDetails(String id):User[]

4.4.3 Package: model.item

4.4.3.1 Class: Item

Item
serialVersionUID itemid: String itemname: String catid: String sellerid: String
Item(String itemid,String itemname, String catid,String sellerid) Item() getItemname ():String setItemname (String itemname):void getCatid ():String setCatid (String catid) :void getSellerid ():String setSellerid (String sellerid) :void getItemid ():String toString() : String

4.4.3.2 Interface: ItemModel

ItemModel
addItem(Item item):void getLastUserItemId(String userid):Item[]

4.4.3.3 Class: ItemModelImpl

ItemModelImpl
emgr: EntityManager instance: ItemModel
ItemModelImpl() getInstance(): UserModel addItem(Item item) :void getLastUserItemId(String userid) :Item[]

4.4.4 Package: model.category

4.4.4.1 Class: Category

Category
serialVersionUID catid: String catName: String
Category(String catid, String catName) Category() getCatid ():String setCatid (String catid):void getCatName ():String setCatName (String catName) :void toString() : String

4.4.4.2 Interface: CategoryModel

CategoryModel
getCategoryList():Category[]

4.4.4.3 Class: CategoryModelImpl

CategoryModelImpl
emgr: EntityManager instance: CategoryModel
getCategoryList():Category[]

4.4.5 Package: model.bid

4.4.5.1 Class: Bid

Bid
serialVersionUID bidId: String auctionId: String bidderId: String bidAmount: String time: String
Bid(String bidId, String auctionId, String bidderId, String bidAmount, String time) Bid() getBidId ():String setBidId (String bidId):void getAuctionId ():String setAuctionId (String auctionId) :void getBidderId ():String setBidderId (String bidderId) :void getBidAmount ():String setBidAmount (String bidAmount) :void getTime ():String setTime (String time) :void toString() : String

4.4.5.2 Class: BidHistory

Bid
serialVersionUID bidId: String auctionId: String name: String bidAmount: String time: String
Bid(String bidId, String auctionId, String name, String bidAmount, String time) Bid() getBidId ():String setBidId (String bidId):void getAuctionId ():String setAuctionId (String auctionId) :void getName ():String setName (String name) :void getBidAmount ():String setBidAmount (String bidAmount) :void getTime ():String setTime (String time) :void toString() : String

4.4.5.3 Interface: BidModel

BidModel
addBid(Bid bid):void getBidHistory(String id): BidHistory[] deleteAuction(String auctionid):void

4.4.5.4 Class: BidModelImpl

BidModelImpl
emgr: EntityManager instance: BidModelImpl
addBid(Bid bid):void getBidHistory(String id): BidHistory[] deleteAuction(String auctionid):void

4.4.6 Package: model.auction

4.4.6.1 Class: Auction

Auction
serialVersionUID auctionid: String itemid: String startprice: String currentbid: String
Auction(String auctionid, String itemid, String startprice, String currentbid) Auction () getItemid ():String setItemid (String itemid):void getStartprice ():String setStartprice (String startprice) :void getCurrentbid ():String setCurrentbid (String currentbid) :void getAuctionid ():String toString() : String

4.4.6.2 Class: AuctionItem

AuctionItem
serialVersionUID auctionid: String itemname: String catName: String startprice: String currentbid: String
public AuctionItem(String itemid, String auctionid, String itemname, String catName, String startprice, String currentbid) AuctionItem () getItemid ():String setItemid (String itemid):void getStartprice ():String setStartprice (String startprice) :void getCurrentbid ():String setCatName(String catName) :void getCatName ():String setItemname(String itemname) :void getItemname ():String setCurrentbid (String currentbid) :void getAuctionid ():String toString() : String

4.4.6.2 Interface: AuctionModel

AuctionModel
addAuction(Auction auction):void getAuction(String id): Auction getUserAuction(String userid): AuctionItem[] deleteAuction(String itemid):void searchAuction(String name):AuctionItem[] getSearchedAuction(String id):AuctionItem[] updateAuction(String id, String currentbid):void getCategorisedAuction(String id):AuctionItem[]

4.4.6.3 Class: AuctionModelImpl

AuctionModelImpl
emgr: EntityManager instance: AuctionModelImpl
addAuction(Auction auction):void getAuction(String id): Auction getUserAuction(String userid): AuctionItem[] deleteAuction(String itemid):void searchAuction(String name):AuctionItem[] getSearchedAuction(String id):AuctionItem[] updateAuction(String id, String currentbid):void getCategorisedAuction(String id):AuctionItem[]

4.4.7 Package: controller.user

4.4.7.1 Class: LoginController

LoginController
serialVersionUID Model:UserModel catModel: CategoryModel

LoginController()
doGet(HttpServletRequest request, HttpServletResponse response)
doPost(HttpServletRequest request, HttpServletResponse response)
processRequest(HttpServletRequest request, HttpServletResponse response)

4.4.8 Package: controller.item

4.4.8.1 Class: ItemController

ItemController
serialVersionUID itemModel: ItemModel auctionModel: AuctionModel

ItemController ()
doGet(HttpServletRequest request, HttpServletResponse response)
doPost(HttpServletRequest request, HttpServletResponse response)
processRequest(HttpServletRequest request, HttpServletResponse response)

4.4.9 Package: controller.category

4.4.9.1 Class: CategoryController

CategoryController
serialVersionUID catModel: CategoryModel model: AuctionModel

CategoryController ()
doGet(HttpServletRequest request, HttpServletResponse response)
doPost(HttpServletRequest request, HttpServletResponse response)
processRequest(HttpServletRequest request, HttpServletResponse response)

4.4.10 Package: controller.bid

4.4.10.1 Class: BidController

BidController
serialVersionUID bidModel: BidModel auctionModel: AuctionModel

BidController ()
doGet(HttpServletRequest request, HttpServletResponse response)
doPost(HttpServletRequest request, HttpServletResponse response)
processRequest(HttpServletRequest request, HttpServletResponse response)

4.4.11 Package: controller.auction

4.4.11.1 Class: AuctionController

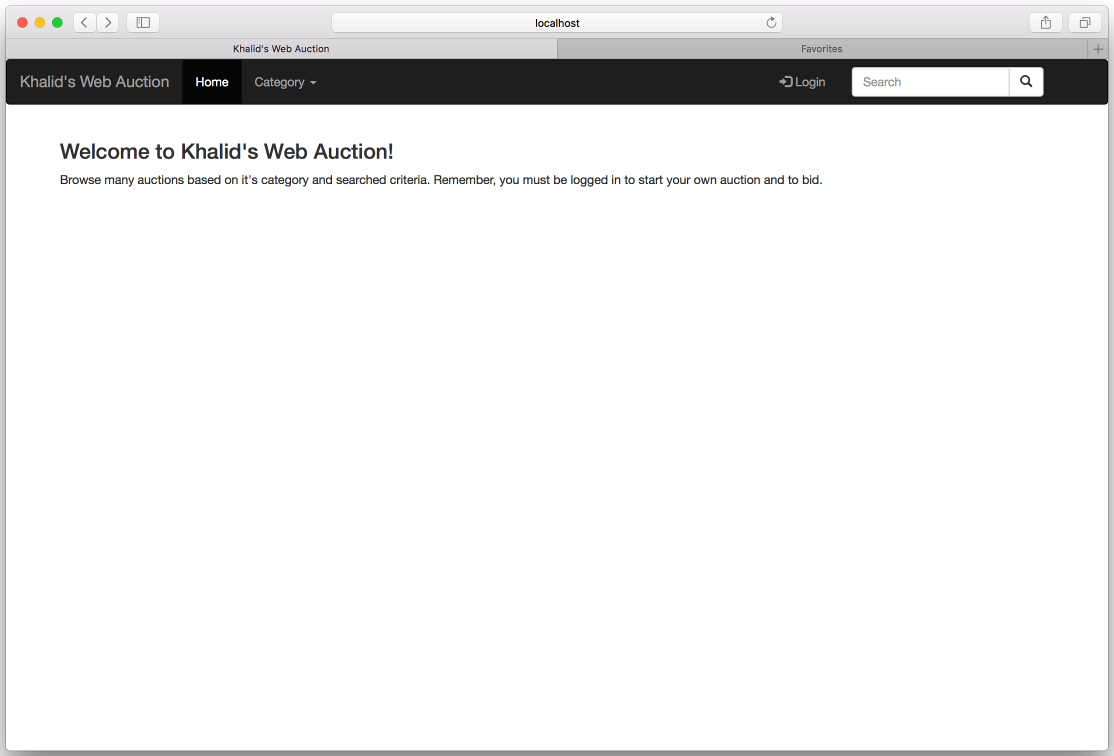
AuctionController
serialVersionUID bidModel: BidModel model: AuctionModel catModel: CategoryModel

AuctionController ()
doGet(HttpServletRequest request, HttpServletResponse response)
doPost(HttpServletRequest request, HttpServletResponse response)
processRequest(HttpServletRequest request, HttpServletResponse response)

5. Screen Dump Walkthrough

5.1 Home Page for a Non-Logged in User

When the user first accesses the web application, they will be directed to the home page. From here, the user has a variety of options. These are Category, Login and Search for an auction.

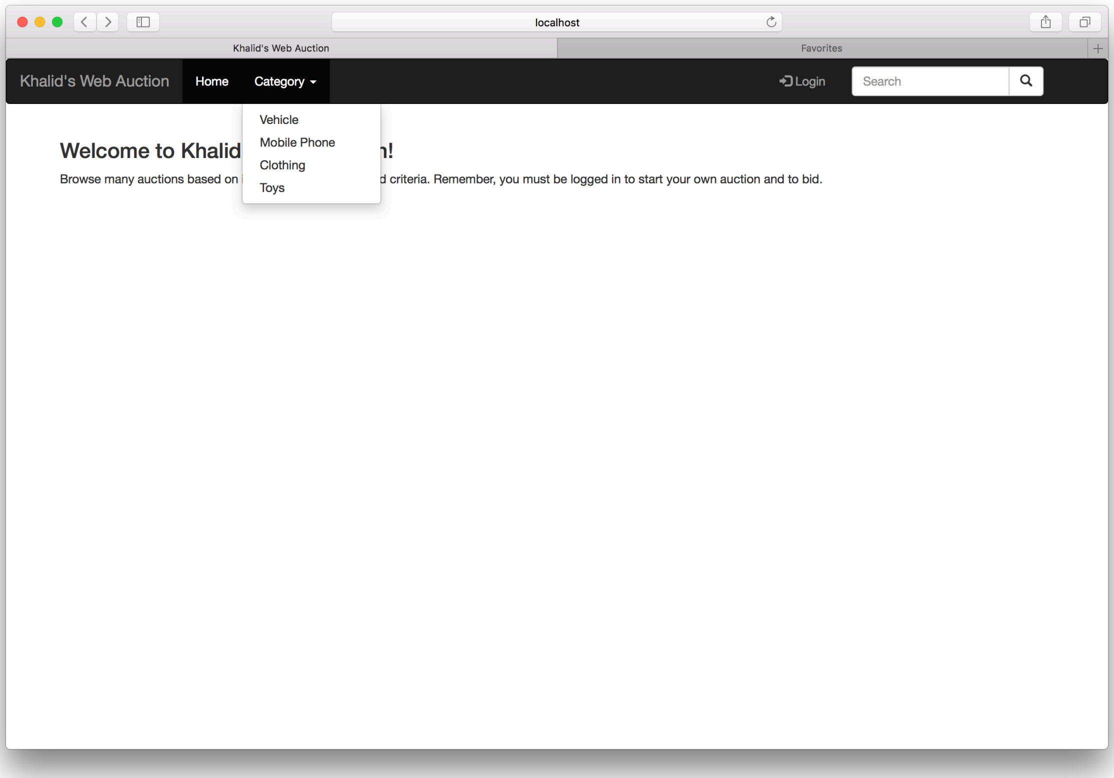


Welcome to Khalid's Web Auction!

Browse many auctions based on it's category and searched criteria. Remember, you must be logged in to start your own auction and to bid.

5.2 Category Option

If the user clicks on the Category option, they will be presented with a dropdown of additional options. If the user clicks on any of the option from the dropdown, they will be taken to a result list of auction, relating to the category that was chosen.

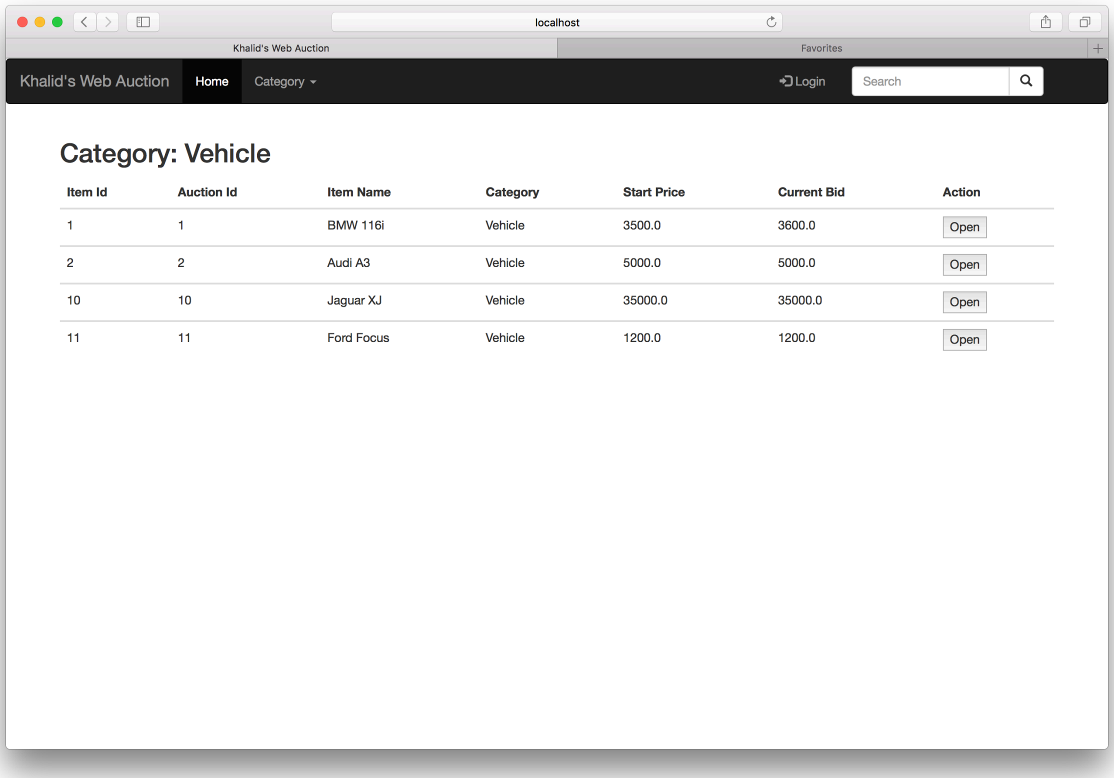


Welcome to Khalid ...n!

Browse many auctions based on ...d criteria. Remember, you must be logged in to start your own auction and to bid.

5.3 Category Result List

Once the user clicks the on category option, they will be taken to a result list of their chosen category. From here, the user can see the current status of returned auction. The user can then open any of the auctions which will provide even more detail about the auction.

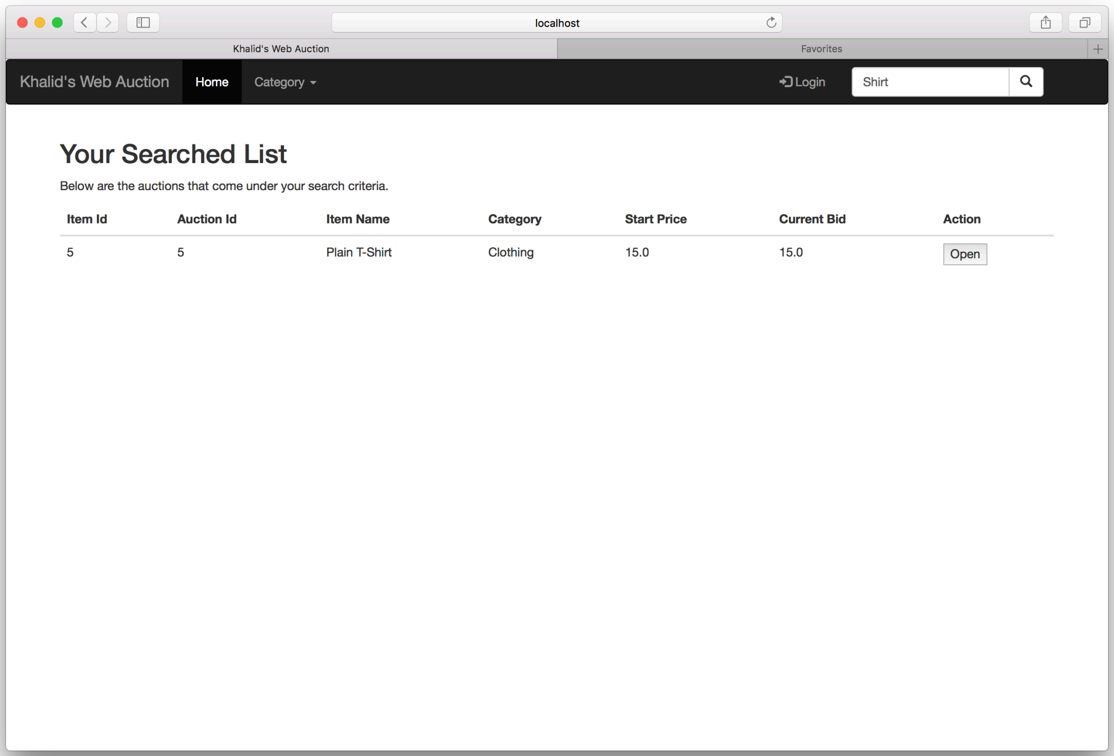


Category: Vehicle

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid	Action
1	1	BMW 116i	Vehicle	3500.0	3600.0	Open
2	2	Audi A3	Vehicle	5000.0	5000.0	Open
10	10	Jaguar XJ	Vehicle	35000.0	35000.0	Open
11	11	Ford Focus	Vehicle	1200.0	1200.0	Open

5.4 Searched Result List

Rather than looking for auctions by selecting the category, the user is also able to find auctions by using a search bar. The search bar is located on the top right. The user can enter the name of an item that they are looking for and once submitted, a list of results will be display, very similar to the Category Result List (Section 5.3).



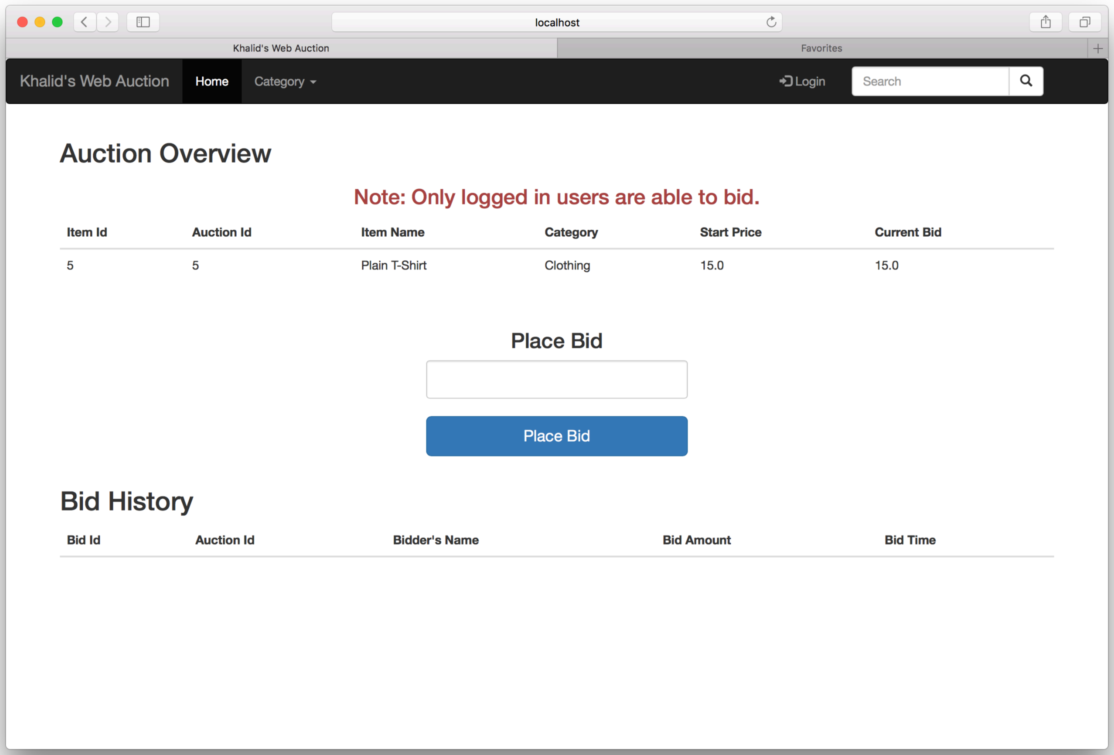
Your Searched List

Below are the auctions that come under your search criteria.

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid	Action
5	5	Plain T-Shirt	Clothing	15.0	15.0	Open

5.5 Auction Overview for a Non-Logged in User

Once a user has selected to open an auction, they will be directed to the Auction Overview. Here, the user can see updated information about the auction, make a bid and see the bid history. However, the page will detect whether the user is logged in or not as only logged in users can make a bid. If a user does attempt to make a bid without logging into the system, they will be directed to the login page.



Auction Overview

Note: Only logged in users are able to bid.

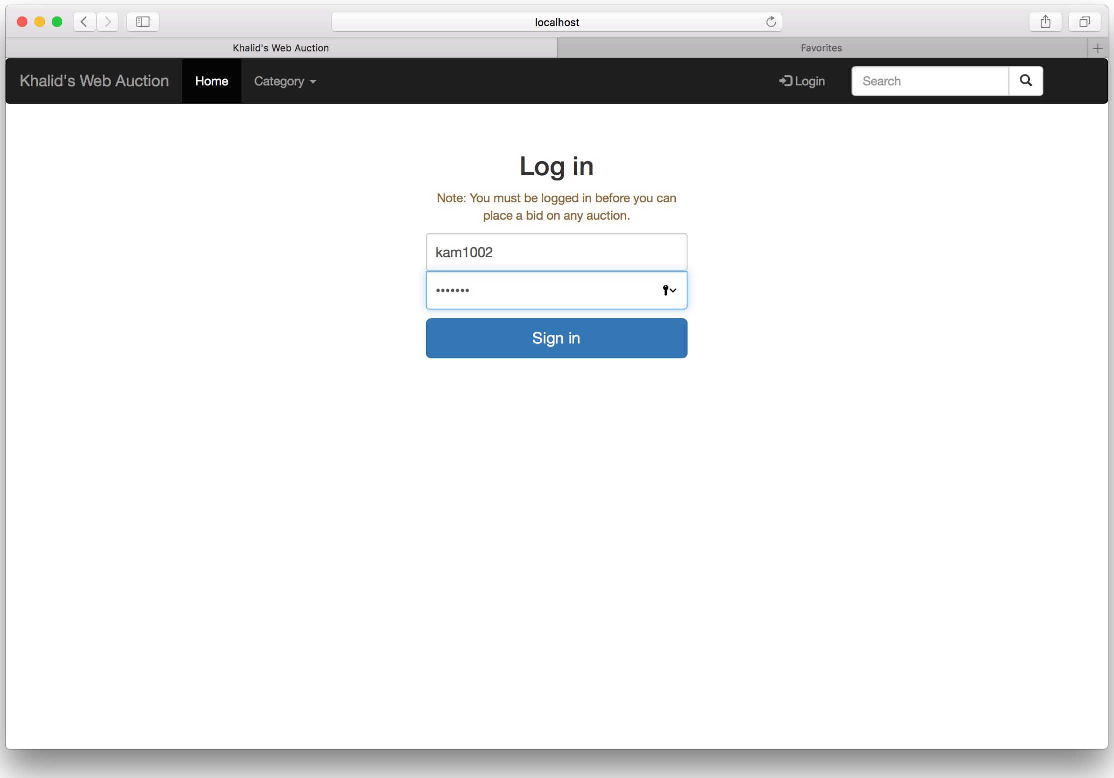
Item Id	Auction Id	Item Name	Category	Start Price	Current Bid
5	5	Plain T-Shirt	Clothing	15.0	15.0

Bid History

Bid Id	Auction Id	Bidder's Name	Bid Amount	Bid Time

5.6 Logging In

The user can log into the web application with a valid credential which is stored in the database. If the credential is invalid, the user will be notified and asked to try again. However, if the credential is valid, the user will be directed back to the home page.



Log in

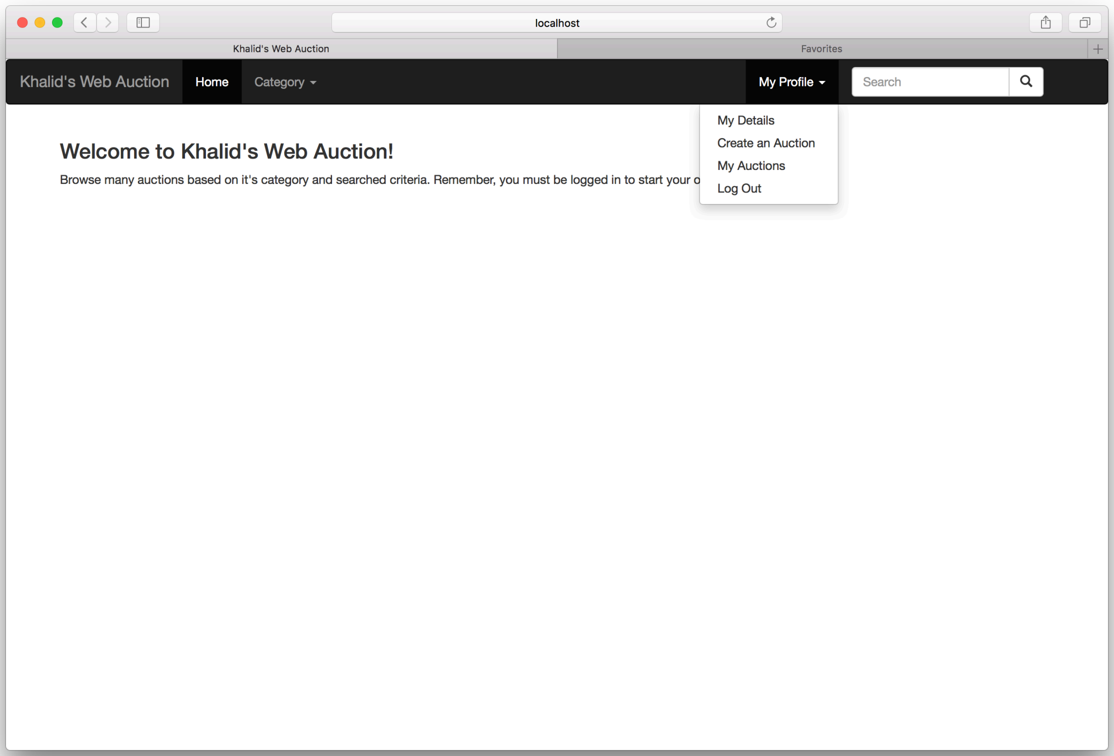
Note: You must be logged in before you can place a bid on any auction.

kam1002

•••••••

5.7 Home Page for a Logged in User

This page is very similar to the Home Page for a Non-Logged in user (Section 5.1) however once logged in, the user will be provided with an additional option; My Profile. Once My Profile is clicked, a dropdown will be shown with additional options; My Details, Create an Auction, My Auctions and Log Out.

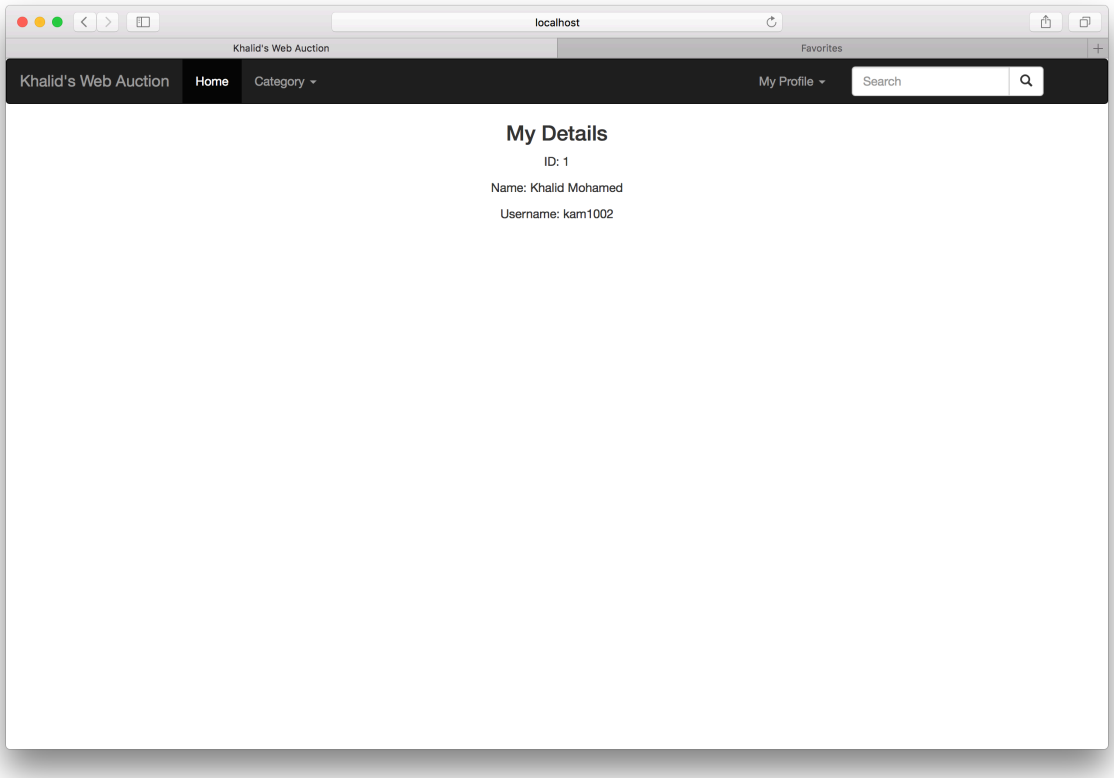


Welcome to Khalid's Web Auction!

Browse many auctions based on it's category and searched criteria. Remember, you must be logged in to start your o

5.8 My Details

If the user clicked on the My Details option from the My Profile dropdown, the user will be taken to a page that displays all of the personal information stored in the database about the user.



My Details

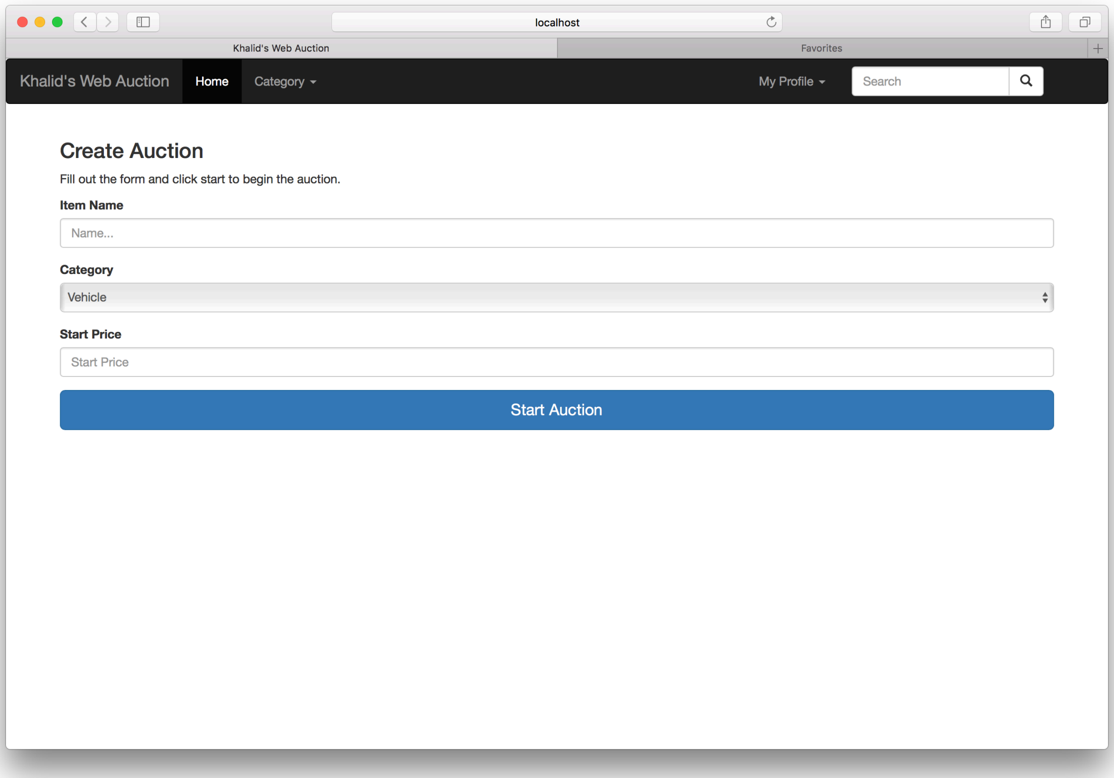
ID: 1

Name: Khalid Mohamed

Username: kam1002

5.9 Create an Auction

If the user clicked on the Create an Auction option from the My Profile dropdown, the user will be taken to a page that allows the user to create an auction. Here, all of the fields are required.



Create Auction

Fill out the form and click start to begin the auction.

Item Name

Name...

Category

Vehicle

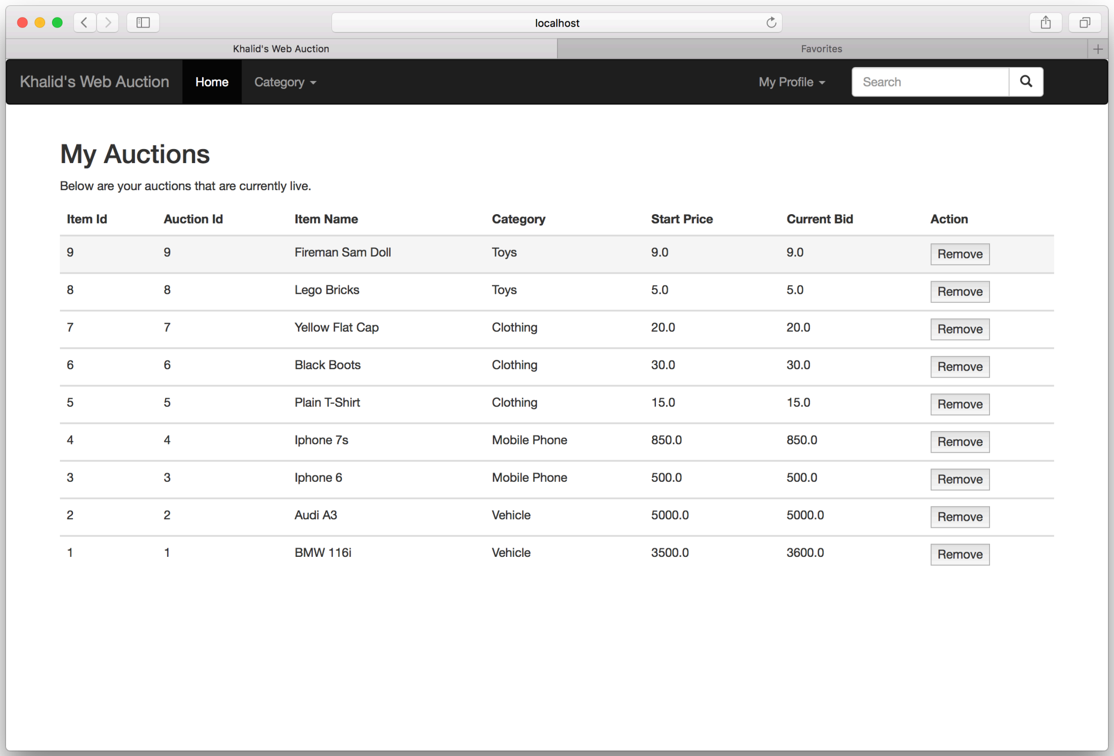
Start Price

Start Price

Start Auction

5.10 My Auctions

If the user clicked on the My Auctions option from the My Profile dropdown, the user will be taken to a page that will display all of the user's currently live auctions. The My Auctions page will display updated information about the auction. The user will also have the option to remove an auction.



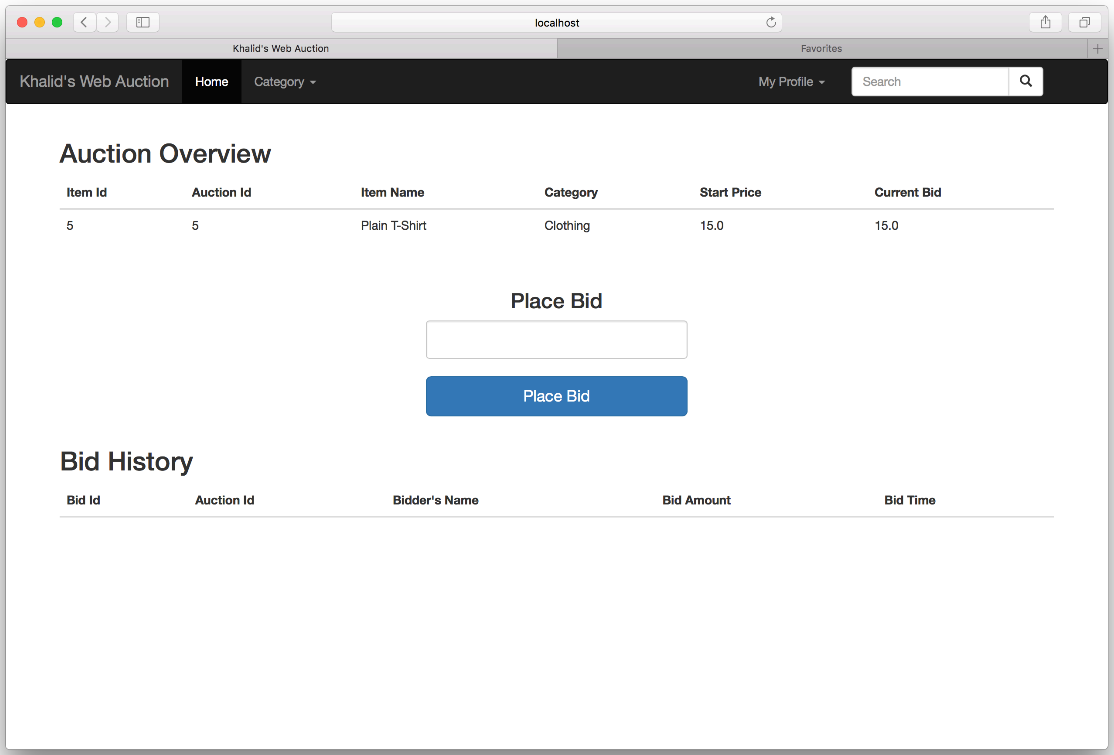
My Auctions

Below are your auctions that are currently live.

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid	Action
9	9	Fireman Sam Doll	Toys	9.0	9.0	Remove
8	8	Lego Bricks	Toys	5.0	5.0	Remove
7	7	Yellow Flat Cap	Clothing	20.0	20.0	Remove
6	6	Black Boots	Clothing	30.0	30.0	Remove
5	5	Plain T-Shirt	Clothing	15.0	15.0	Remove
4	4	Iphone 7s	Mobile Phone	850.0	850.0	Remove
3	3	Iphone 6	Mobile Phone	500.0	500.0	Remove
2	2	Audi A3	Vehicle	5000.0	5000.0	Remove
1	1	BMW 116i	Vehicle	3500.0	3600.0	Remove

5.11 Auction Overview for a Logged-in User

Once a user has selected to open an auction, they will be directed to the Auction Overview. Here, the user can see updated information about the auction, make a bid and see the bid history. Once a user has logged in, they will have the ability to make a bid.



Auction Overview

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid
5	5	Plain T-Shirt	Clothing	15.0	15.0

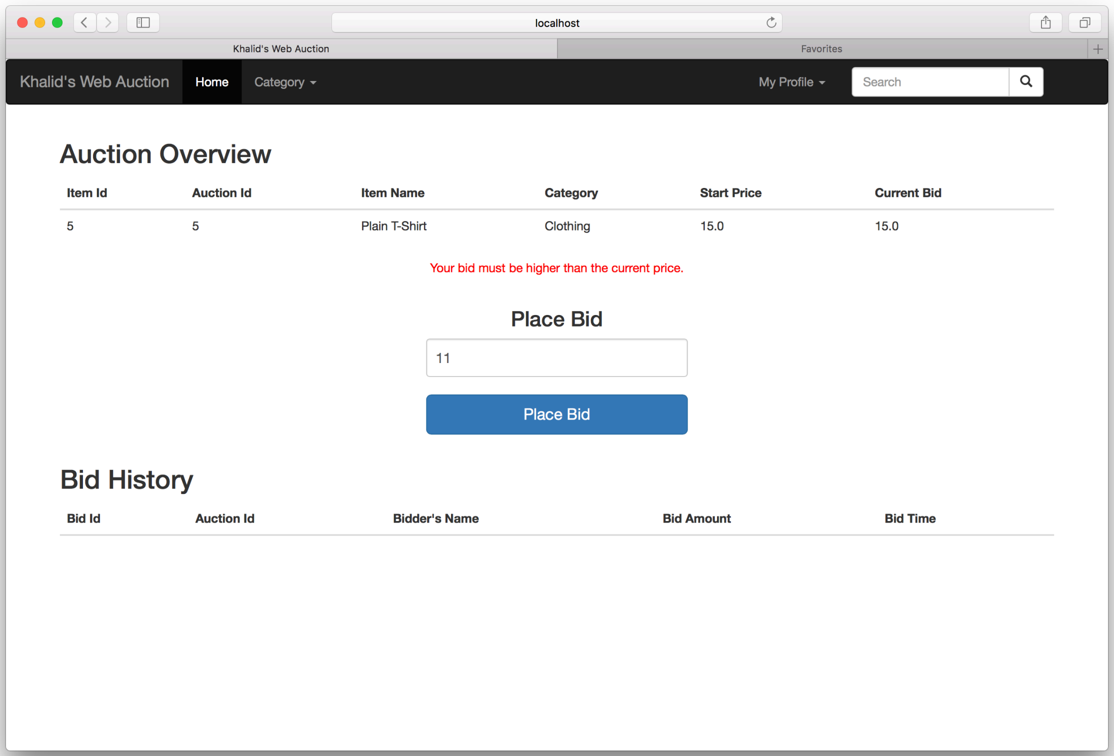
Place Bid

Bid History

Bid Id	Auction Id	Bidder's Name	Bid Amount	Bid Time

5.12 Bidding Below the Current Bid

If the user attempts to bid below the current bid, the user will be notified that they must bid higher than the current bid.



Auction Overview

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid
5	5	Plain T-Shirt	Clothing	15.0	15.0

Your bid must be higher than the current price.

Place Bid

11

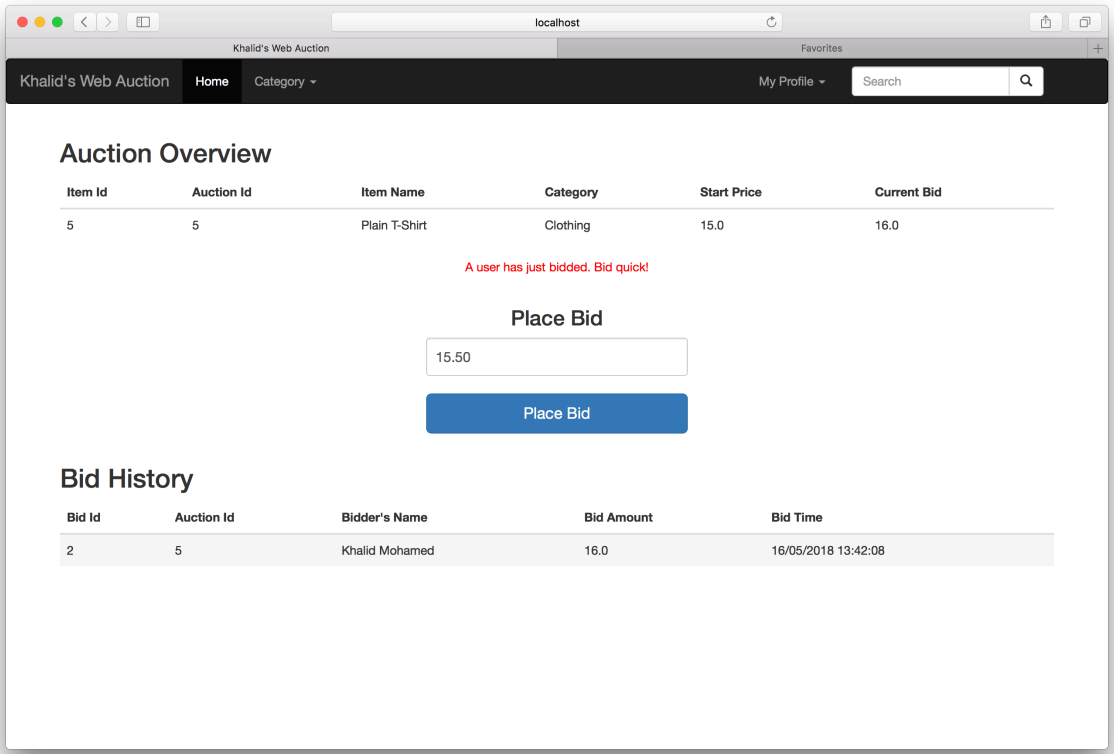
Place Bid

Bid History

Bid Id	Auction Id	Bidder's Name	Bid Amount	Bid Time

5.13 Out-Bided by another User

As the web application does not update the page in real-time, it is possible for another user to bid while this user is in the process of bidding. When this happens, once the user submits their bid, the page will automatically be updated with the new current bid and the user will be notified that a user already made a bid. The user can check the bid history by scrolling to the bottom of the page.



Auction Overview

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid
5	5	Plain T-Shirt	Clothing	15.0	16.0

A user has just bidded. Bid quick!

Place Bid

15.50

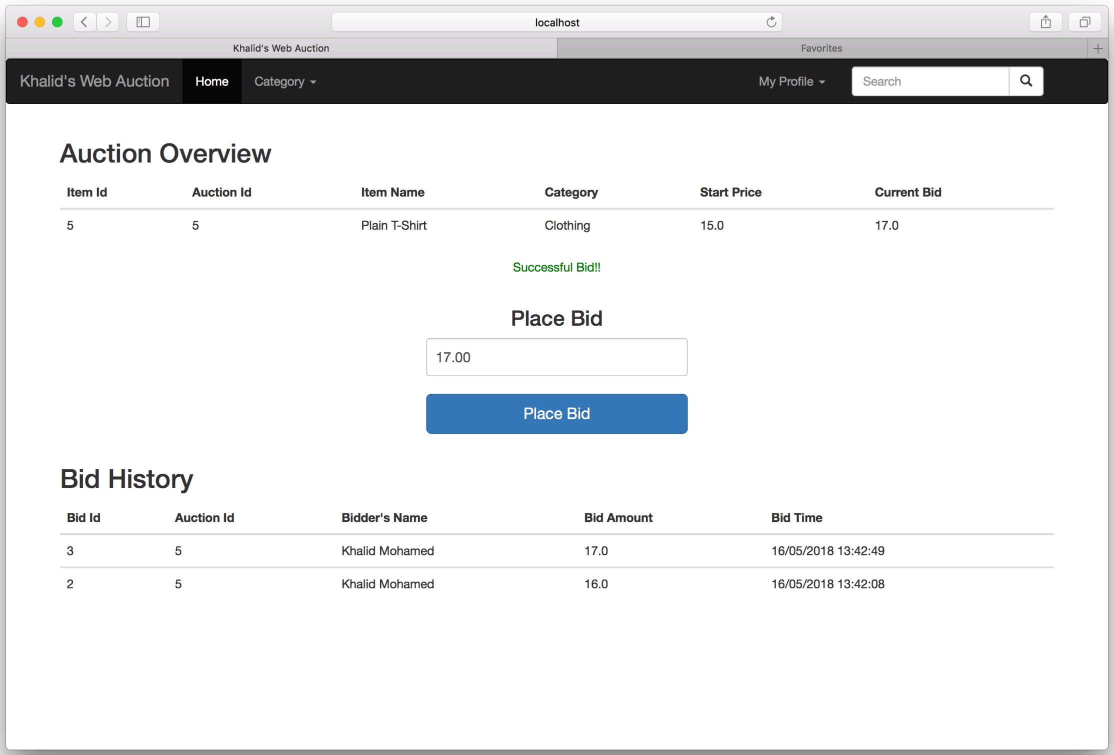
Place Bid

Bid History

Bid Id	Auction Id	Bidder's Name	Bid Amount	Bid Time
2	5	Khalid Mohamed	16.0	16/05/2018 13:42:08

5.14 Successful Bid

If the user bid satisfies all of the requirements based on Sections 5.12 and 5.13, the page will update, notifying the user that the bid was successful. The user can then check the Bid History and will be able to see their bid top of the table.



Auction Overview

Item Id	Auction Id	Item Name	Category	Start Price	Current Bid
5	5	Plain T-Shirt	Clothing	15.0	17.0

Successful Bid!!

Place Bid

17.00

Place Bid

Bid History

Bid Id	Auction Id	Bidder's Name	Bid Amount	Bid Time
3	5	Khalid Mohamed	17.0	16/05/2018 13:42:49
2	5	Khalid Mohamed	16.0	16/05/2018 13:42:08

6. Deployment

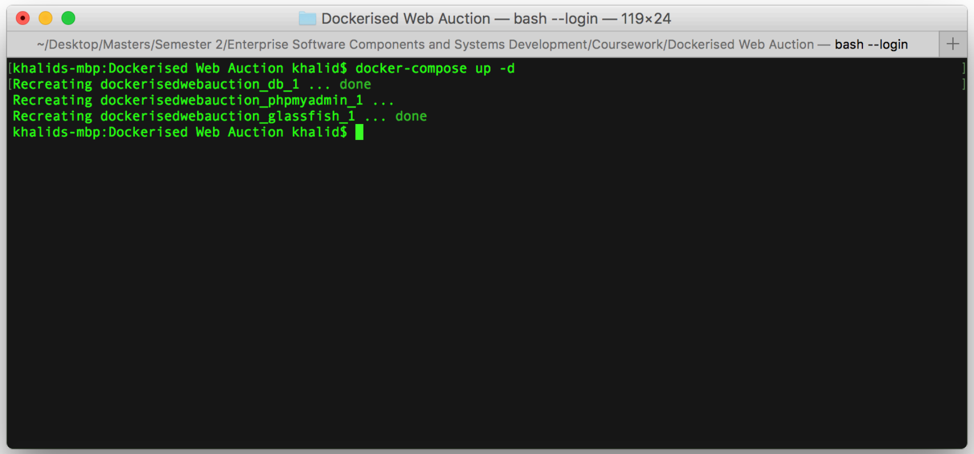
6.1 Deployment and Configuration Instructions

This project uses many components for it to function correctly; Glassfish, JDK, MySQL Connector, MariaDB. Therefore, I decided to use a platform called Docker. Docker allows you to run this web application with very little configuration and on any system provided that firstly Docker is install onto the system and that the custom dockerfile and docker-compose.yml for this web application is on your system.

NOTE: This requires prior knowledge of Docker and GitHub.

1. To obtain all of the necessary files required to run this project, it can be cloned from this GitHub repository: https://github.com/Khalid145/WebAuction.

2. Once you have cloned the repository to your system, navigate to the "Dockerised Web Auction" folder and run the "**docker-compose up –-build –d**" command in your terminal. This command will start and configure Glassfish and database, ready for it to be used by the web application.



Dockerised Web Auction — bash --login — 119×24

~/Desktop/Masters/Semester 2/Enterprise Software Components and Systems Development/Coursework/Dockerised Web Auction — bash --login

```
khalids-mbp:Dockerised Web Auction khalid$ docker-compose up -d
Recreating dockerisedwebauction_db_1 ... done
Recreating dockerisedwebauction_phpmyadmin_1 ...
Recreating dockerisedwebauction_glassfish_1 ... done
khalids-mbp:Dockerised Web Auction khalid$ 
```

3. Next step is to create the database and all of the necessary tables. This has been provided to you, which can be found in the "Glassfish" folder via sql script. To access the database interface, you can use PHPMyAdmin. This can be found on **localhost:8181** when typed into your browser.

4. Once the database has been created, it is time to deploy the web application on Glassfish. All of the JDBC connection and JDBC resource has already been created for you. To deploy the application, a .war file has been provided in the Glassfish folder from the GitHub repository. To access Glassfish via the browser, enter **localhost:4848**. You will be prompted to enter a username and password to access Glassfish. This has already been created for you:

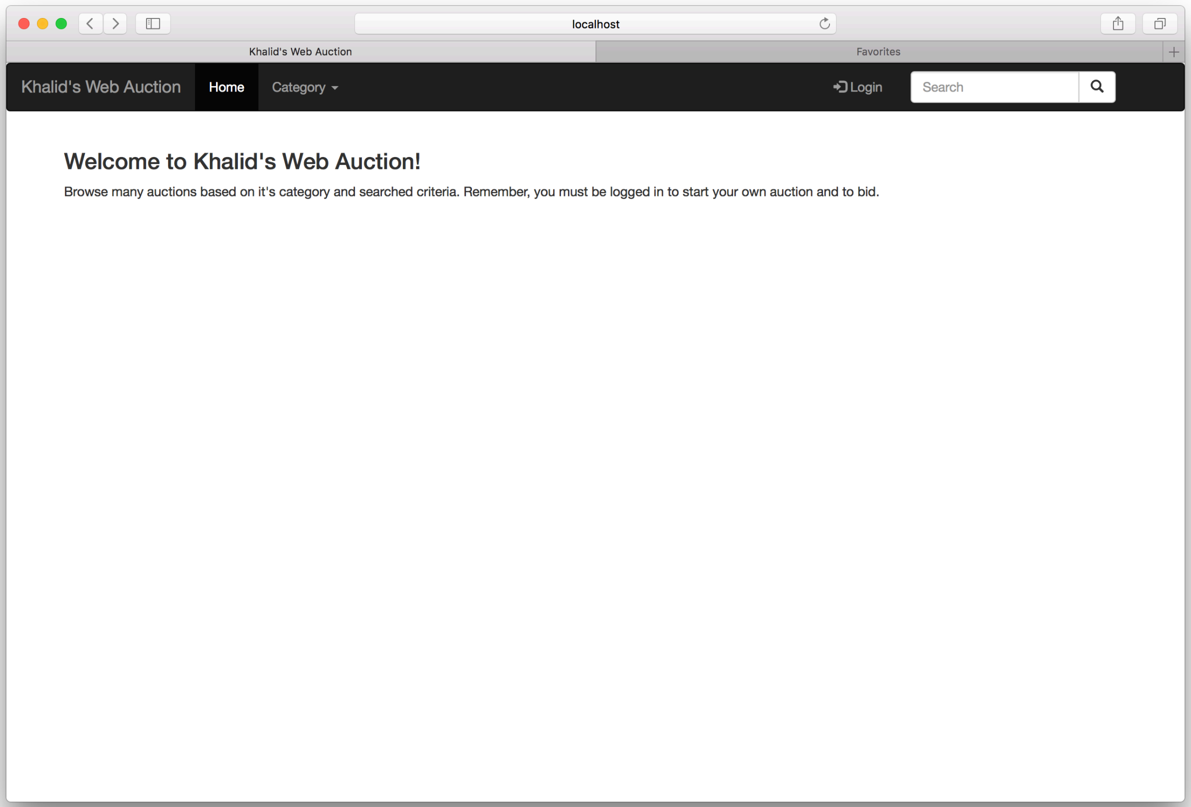
Username: admin

Password: password

5. Once on the Glassfish home page, there will be a side menu bar on the left hand side of the page. Click on Application. Once the Application page appears, click on the Deploy button.

6. The user will be required to first upload the .war file to glassfish. Once uploaded, the user will need to change the application type from **"Web Application" to "Enterprise Application"** and click OK.

7. Once the application has successfully deployed and no errors were displayed to the user, the user can access the web application by entering **localhost/WebAuction** into the browser.



Welcome to Khalid's Web Auction!

Browse many auctions based on it's category and searched criteria. Remember, you must be logged in to start your own auction and to bid.