Software Design Specification Document
(CS360)

GoBid

Group Number: 08

Saad Akbar
Syed Taimoor
Taimur Salman
Salman Masood

Course: Software Engineering CS360

Instructor: Suleman Shahid

University: Lahore University of Management Sciences (LUMS

Version: 1.0
Date: (8/03/2020)
Number of hours spent on this document: 26-32

Contents

1 Change Log

<TO DO: Please use this section to update us on the project scope, if it is different from the one you proposed in the SRS document. Highlight major changes or deviations>

1.1 Project Scope

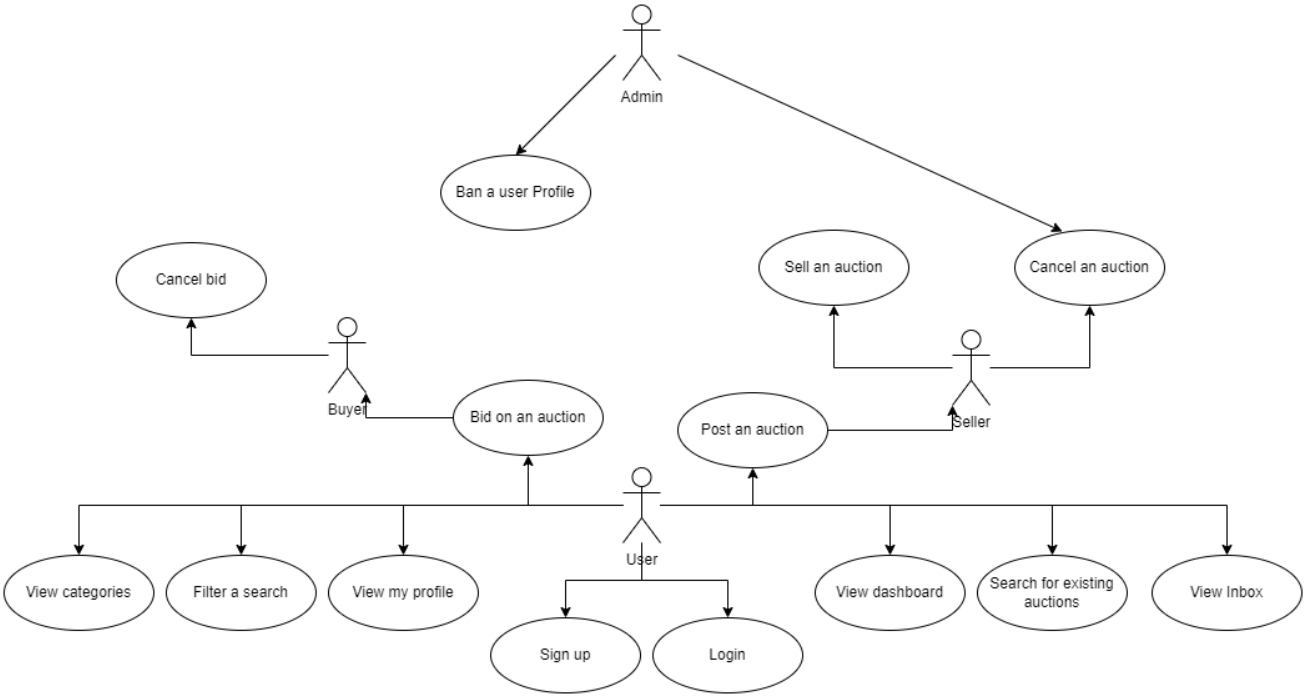
The Project aims to provide a working bidding app with a working database and deployment, that can be accessed from any web browser. The bidding app will allow consumer to consumer buying and selling using an auction-based method where the user will not need to go through the hassle of talking and negotiating with multiple clients and simply get the best price for their items by selling to the seller with the highest bid.

1.2 Change log

There is a new actor called “Admin”. The admin can view any auction and user, and delete any auction or inactivate any user’s account. This is so that any problematic user or content can be removed.

Primary Actor	Associated Use cases
admin	Ban a user’s account
admin	delete an auction
admin	can view any user profile (without email or contact info)

Updated Use Case Diagram:


Admin
Ban a user Profile
Cancel bid
Sell an auction
Cancel an auction
Buyer
Seller
Bid on an auction
Post an auction
User
View categories
Filter a search
View my profile
View dashboard
Search for existing auctions
View Inbox
Sign up
Login

2 Introduction

The reader will find the document purpose, project scope and the intended audience for the app

2.1 Document Purpose

The document is the design specification document and is meant to give a more fleshed out design for what the client should expect from the final app. This includes the user interface, the navigation of the site.
The document also goes into how the application will be designed on a lower level than the SRS. It describes the various classes, the database choice, the reasons for the given database choice and the different components that make up the whole application, and how these components are connected to each other. It also describes the information flow of the more important processes in the system.

2.2 Product Scope

GoBid is a bidding website which hopes to provide hassle free auctioning in Pakistan, with an easy to use interface. Current options such as OLX are not made with auctioning in mind. They function more as advertisements for people who wish to buy used products. We wish to introduce a website specifically designed for auctioning. Other websites often have auctions of niche, discontinued, and old items which may be used as parts for other products or as collector items. The market for online bidding websites in Pakistan is in its infancy. We hope that our app can drive this market forward and introduce this to the average person, which may then open other avenues for business.

2.3 Intended Audience and Document Overview

This document is intended for a course project, as such the intended audience is our client, the teaching assistants and the instructor grading us. The SDS contains the following sections:

- **Introduction:** An overview of **GoBid** and the SDS itself, along with a list of definitions, acronyms, abbreviations and all references used.
- **Overall Description:** A detailed description of **GoBid**, the functionality, the users and any assumptions that may affect the project at any point.
- **Specific Requirements:** The functional requirements, external interface requirements, and use cases.
- **Other Non-functional requirements:** Any non-functional requirements, performance, safety and quality. Includes user stories, and the group log.
- **Application Structure:** The system will provide some structural details of the program as well, such as class diagrams etc.

The suggested sequence to read this document is to read in order. For the client, the most useful sections would be the overall description and the functional requirements to gain an overall understanding of **GoBid**.

2.4 Definitions, Acronyms and Abbreviations

None.

2.5 References and Acknowledgments

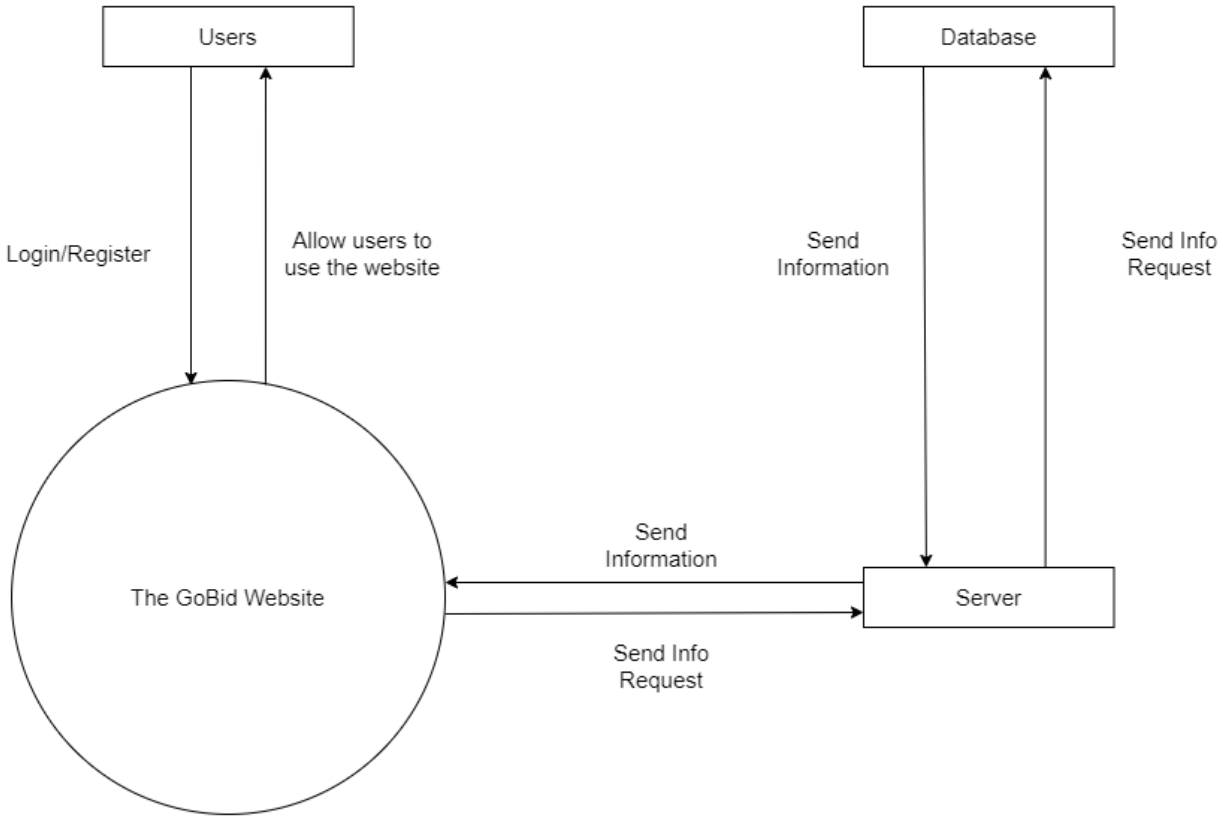
- Ebay
- Daraz
- OLX

3 Overall Description

3.1 System overview

The GoBid website will provide auction services to users in Pakistan. To do this it has an easy to use interface which is connected to a server and a database. The users are able to put up items for auction, bid on existing auctions. All data is stored in a MongoDB database, a no-SQL database. Our overall implementation uses the MERN stack.

The Context Diagram:



The context diagram:

3.2 System constraints

- There are security risks. If the system is successfully hacked, the information of all users can be leaked.
- Traffic has to be managed. Too many users using the website can crash the website.
- The interface can be too heavy, resulting in high load times.
- We cannot ensure the quality of the items being sold as there is no way to background check a user.
- There is no third party wallet system. This increases the chances of users to get scammed.

3.3 Architectural strategies

We will be implementing our application using the MERN stack. The MERN stack technologies are commonly used together and mesh together very well. They allow code to be easily written, read and understood. They allow for modular code. The technologies we will be using are:

- **MongoDB database:** MongoDB is an open source, NoSQL database. It stores data using JSON-like documents. We have made a schema, however this is optional. MongoDB is commonly used in the industry and has a lot of resources available. MongoDB is very quick and being NoSQL, it supports easy horizontal expansion.

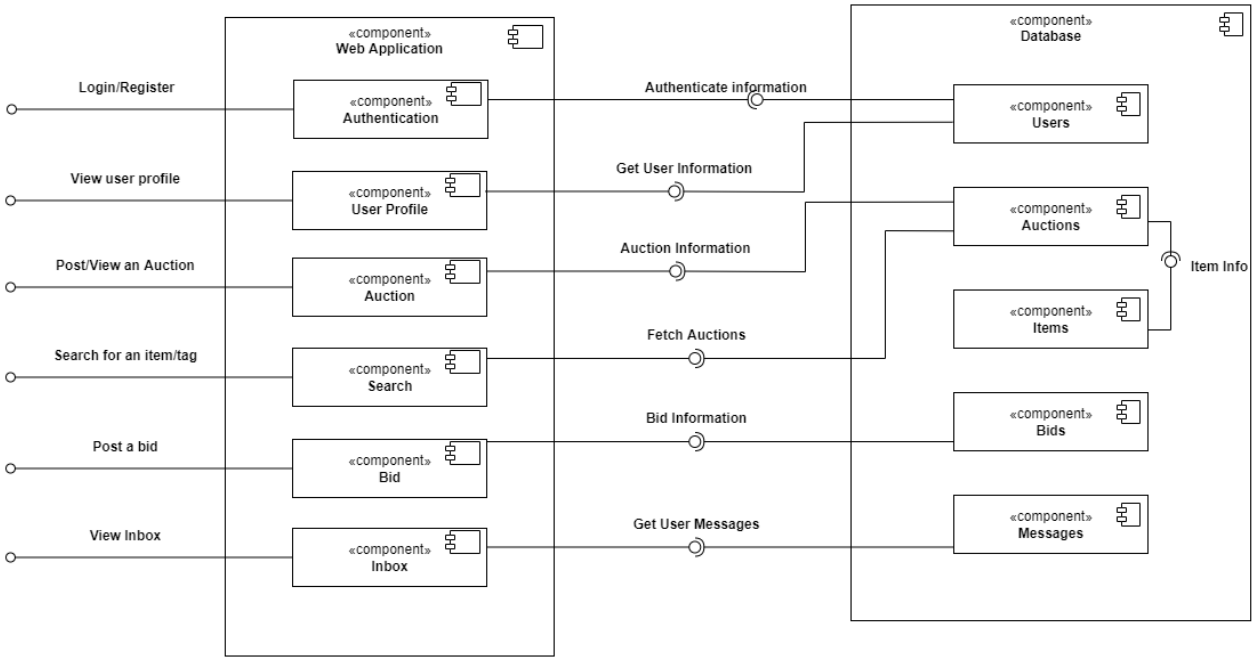
- **React.js library:** React.js is a free, open source library for Javascript developed by Meta. It is used for frontend development. React.js allows us to write easy to use code, and allows us to write modular code.
- **Express.js:** Express.js is a free, open source backend web application framework for Javascript under the MIT license. It will allow us to focus on other elements of our code since it will provide basic routing functionalities.
- **Node.js:** Node.js is an open source backend Javascript environment. It allows users to run Javascript outside of a web browser. Node.js has a dedicated http module.

4 System Architecture

4.1 System Architecture

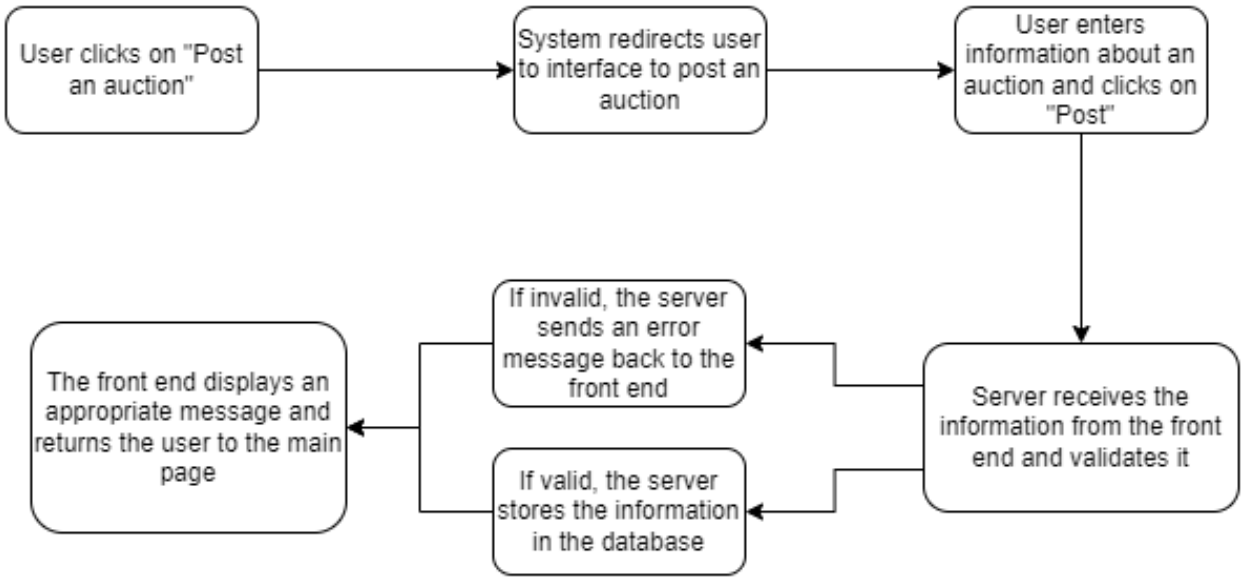
Our software will be managing the creation and maintenance of user accounts, management of auctions and their corresponding bids, notifications using system generated messages, and moderation using an administrator account.
For this purpose, we have divided our system into several components with subcomponents, each handling one of these tasks. On the top most level, we have the web application exchanging information with and interacting with the database to achieve the required functionalities. Each of these components have further subcomponents which interact with one another. For example, in order to post an auction, the Auction component of the web application will be interacting with the Auctions component of the database.
We have chosen this method of decomposition as our application is composed of many features, each depending on one another. For this reason, we have chosen a model which contains a wide array of subcomponents, each assigned a specific use case, working together to achieve the functionality which our application aims to provide.

The component diagram:

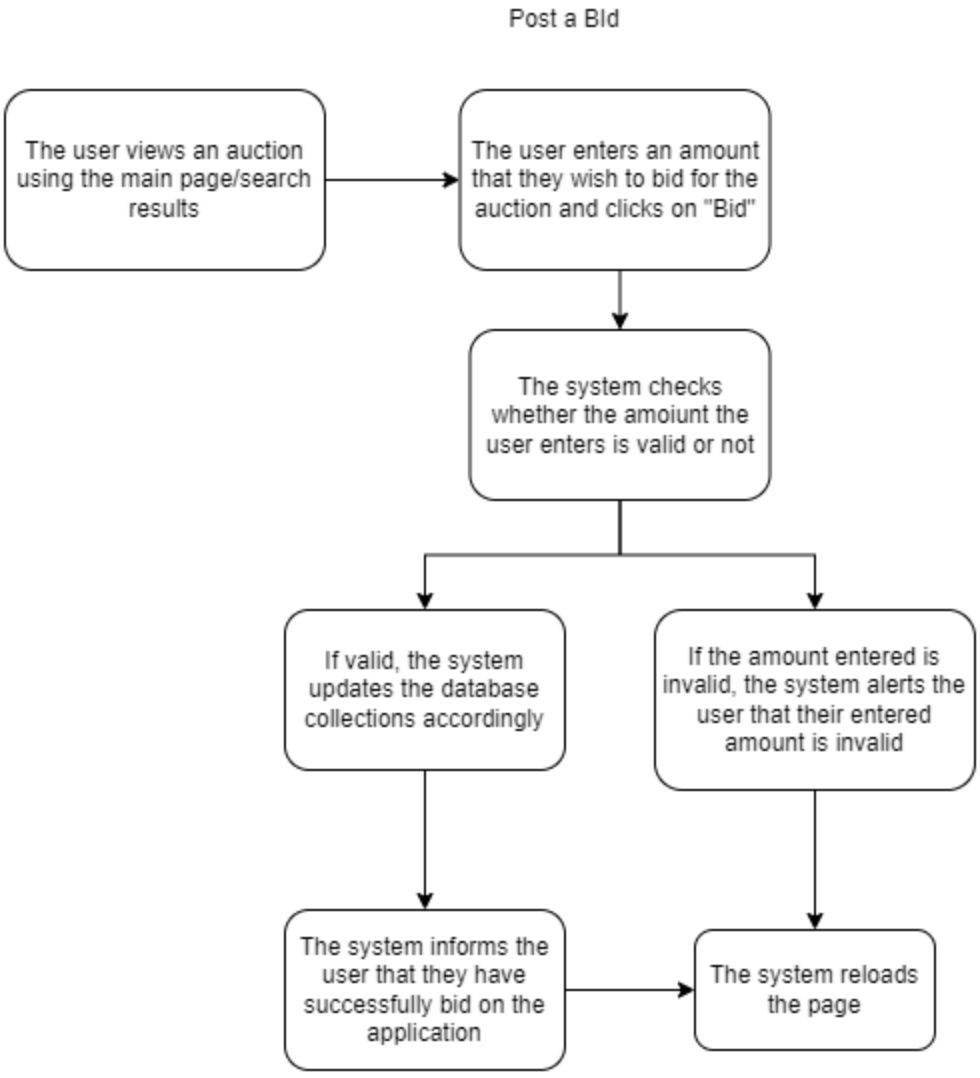


We are not considering “the server” as a component, as the only function it has is to fetch data from the web application and the database and to act as a communication method between the two.

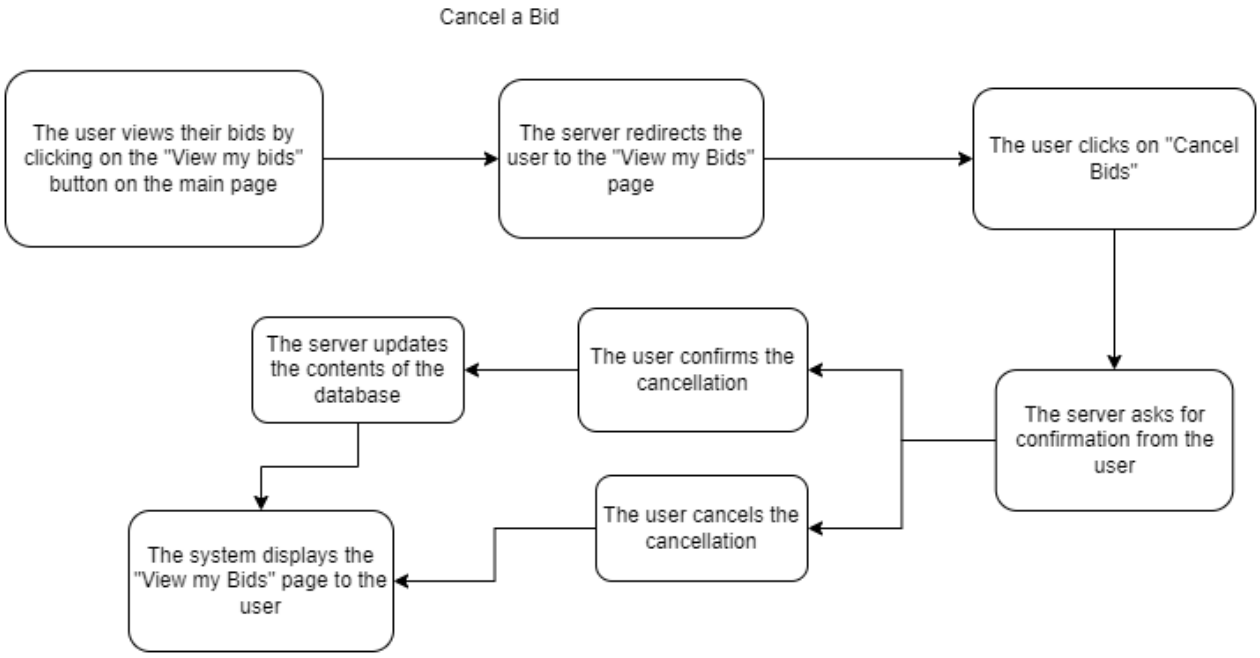
Activity Diagram 1: Post an auction



Activity Diagram 2: Post a Bid

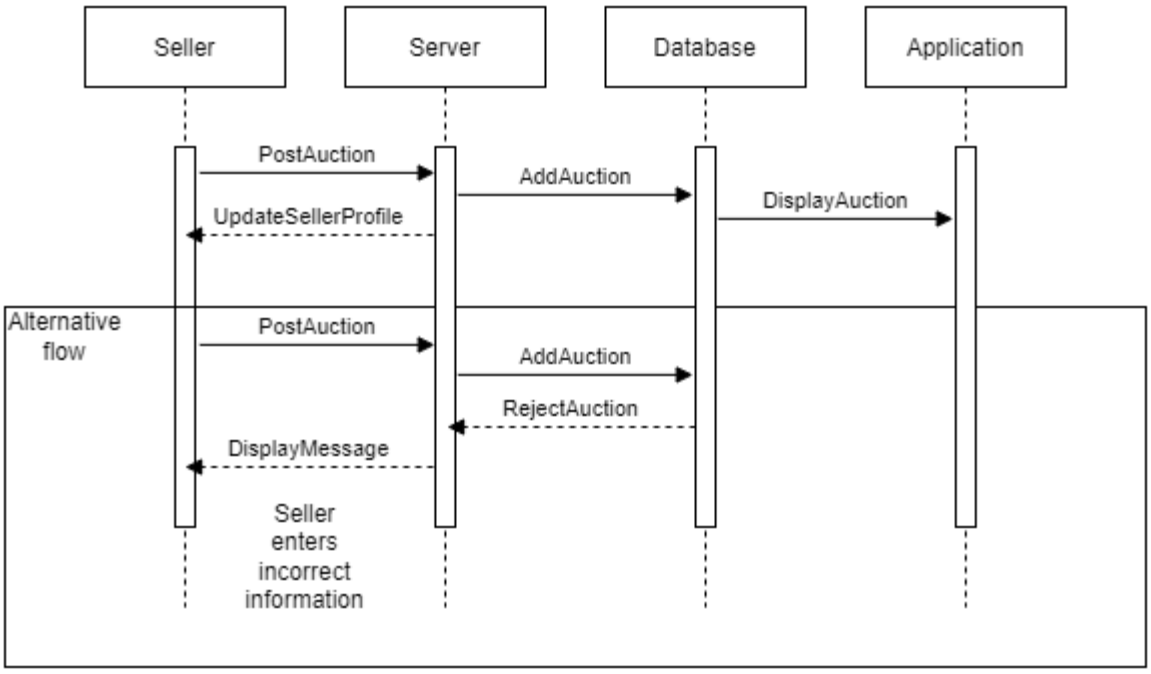


Activity Diagram 3: Cancel a bid

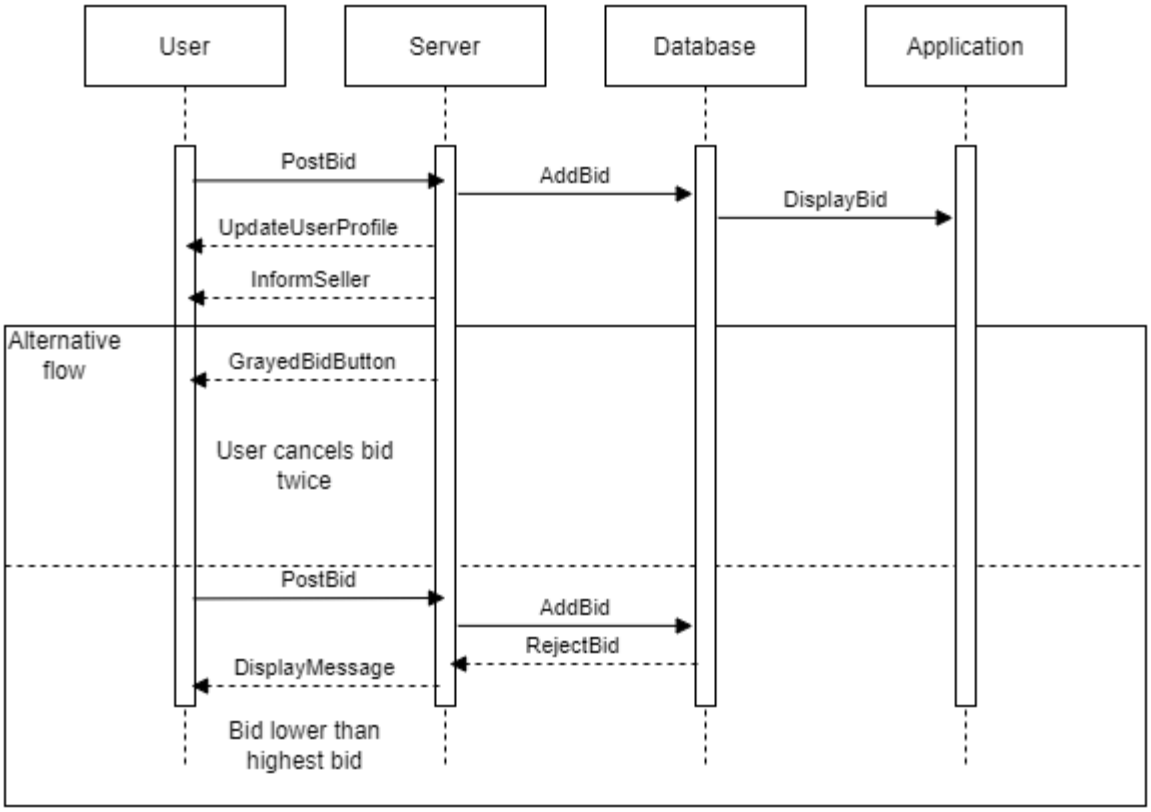


4.2 Subsystem Architecture

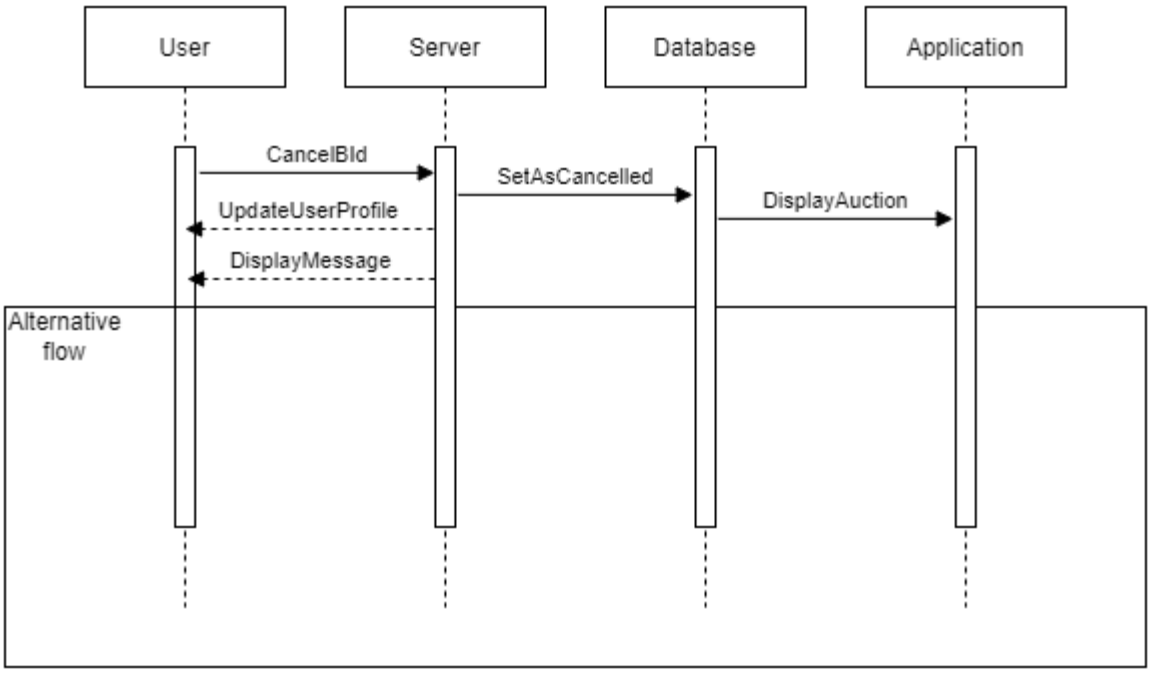
Sequence diagram 1: Post an auction



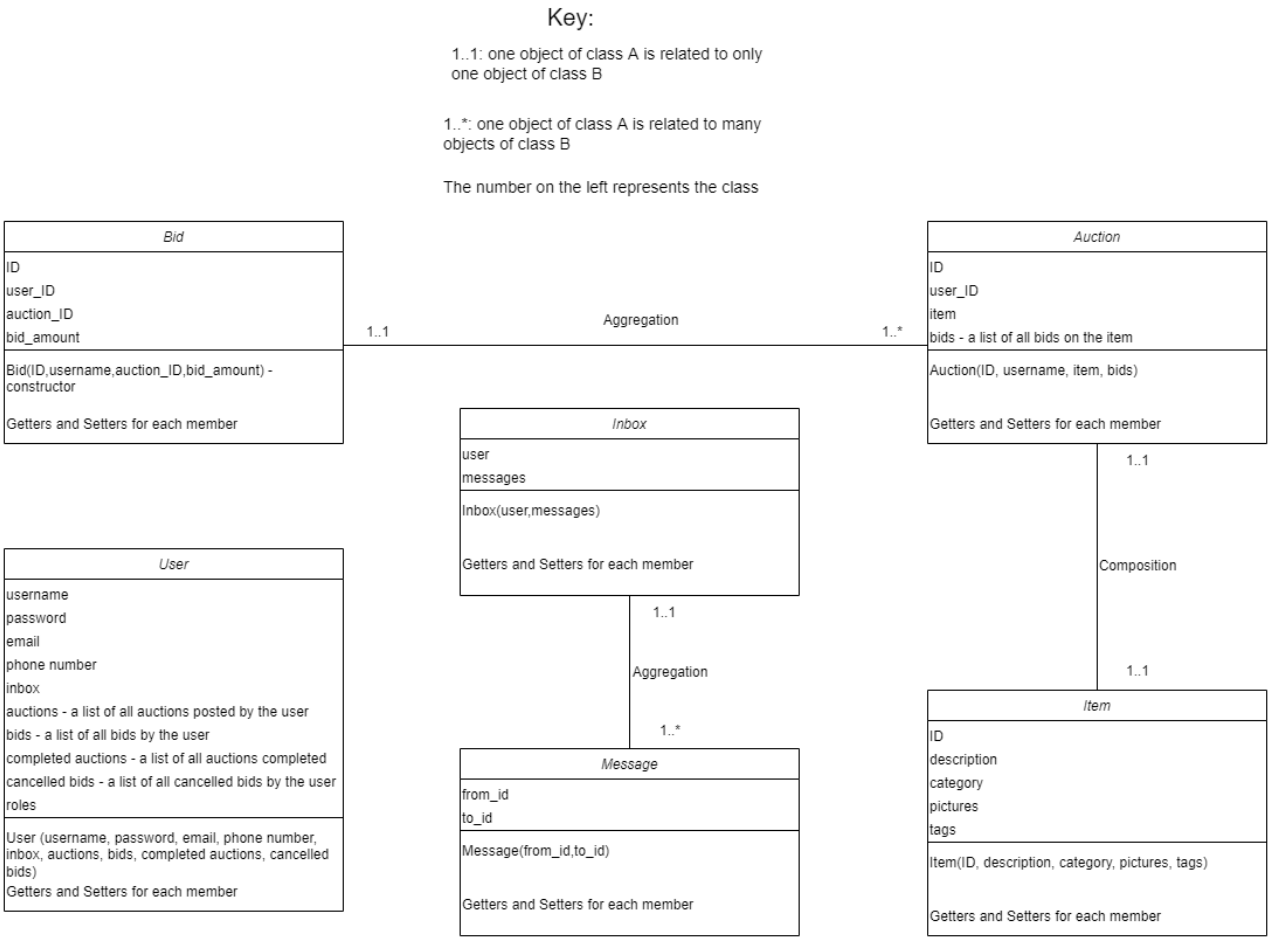
Sequence diagram 2: Post a Bid



Sequence diagram 3: Cancel a Bid



Class diagram:



4.3 Data Structure

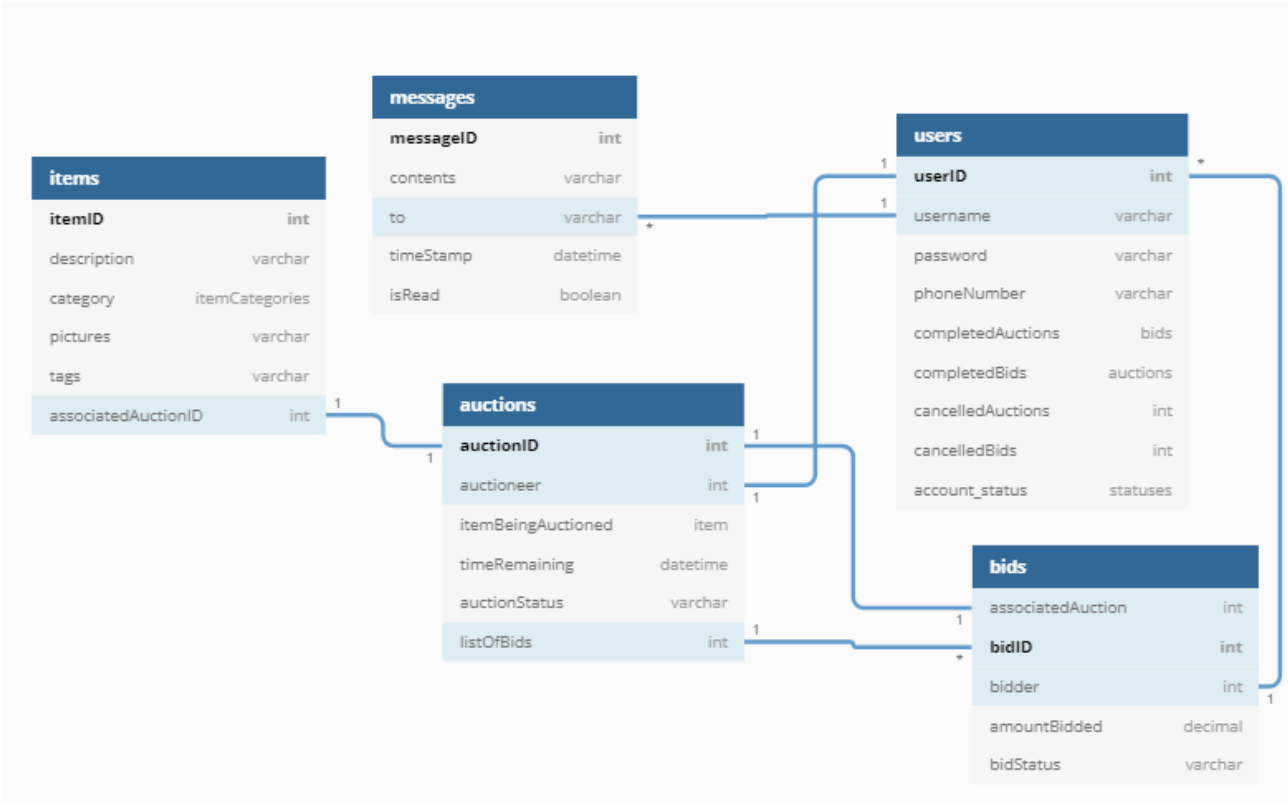
Since we will be implementing our application using the MERN stack, all data will be stored and passed in the form of .JSON/BSON objects. MongoDB stores data as .BSON (Binary JSON) objects. Complexities in data storage and retrieval will be handled by implementing embedding and references in MongoDB collections. When larger amounts of data is to be passed from one component to the other, it will be passed as a list of .JSON objects (or a .JSON object containing a list of the required .JSON objects)

4.4 Database Model

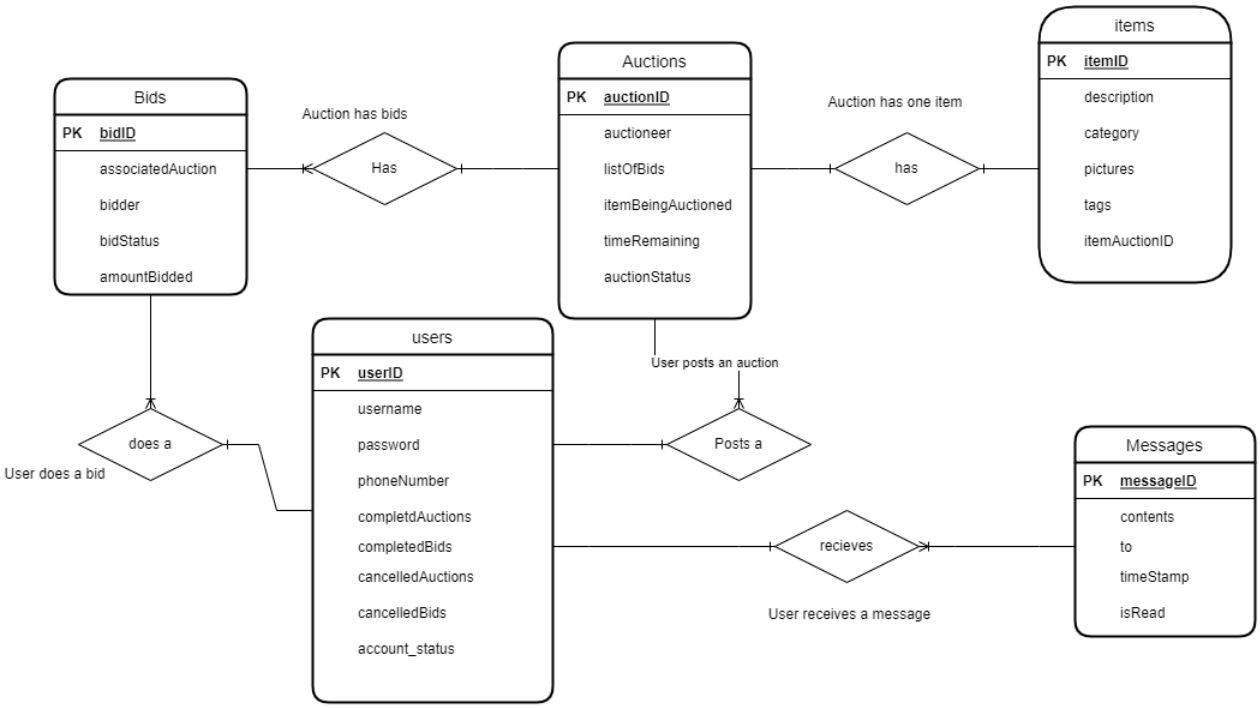
We will be using a single database containing multiple collections (tables) as described in section 4.4.1. The relations between these collections is described through the schema shown below, and an E-R diagram as well

4.4.1 Database scheme and detailed description

Visual presentation of DB Schema:



Entity-Relationship diagram:



Description of all tables and relationships:

- Users:

  Collection containing information about all of the users in the system

  - **(PK) UserID:** Unique ID of every user [type: int/ObjectID]
  - username: username of the user. [type: string]
  - password: the user's password stored as a hashed value [type: string]
  - phoneNumber: The user's phone number [type: string]
  - completedAuctions: A list of the auctions that the user has completed [type: list of auctions]
  - completedBids: A list of the bids that the user has completed [type: list of bids]
  - cancelledAuctions: The number of auctions that the user has canceled [type: int]
  - cancelledBids: The number of bids that the user has canceled [type: int]
  - account_status: The current status of the user's account [type: Enum{user, admin, banned, deleted}]
- Bids

  Collection containing information about all of the bids on the auctions in the system

  - **(PK) bidID:** Unique ID of every bid [type: int/ObjectID]
  - associatedAuction: The auctionID of the auction that this bid is bidded on [type: ObjectID/auctionID]
  - bidder: The userID of the user that has bidded [type: int/ObjectID, the same as userID]
  - amountBidded: The amount the user has bidded. [type: double]
  - bidStatus: If the bid is still ongoing or if it's been canceled [type: varchar/Enum]
- Messages:

  Collection containing information about all of the system generated messages sent to users

  - **(PK) messageID:** Unique ID of every system generated message [type:int/ObjectID]
  - contents: The contents of the message [type: String]
  - to: The userID of the user to whom this message was sent [type: int/ObjectID, the same as UserID
  - timeStamp: The timestamp at which the message was sent [type: datetime/String]
  - isRead: A boolean value which tells if the user has read the particular message yet or not [type: Boolean]
- Items:

  Collection containing information about all of the items being auctioned in the application

  - **(PK) itemID:** Unique ID of every item in the database [type: int/ObjectID]
  - description: A description of the item, given by the seller [type: String]
  - category: The category of the item [type: predefined Enum]
  - pictures: A picture about the item [type: Blob, or the NoSQL variant]

- tags: Tags which the seller may wish to add [type: list of Strings]
- associatedAuctionID: the auctionID where this item is being sold [type: int/ObjectID, the same as auctionID]

- Auctions:

  Collection containing information about all of the auctions in the application

  - **(PK) auctionID:** Unique ID of every auction in the database [type: int/ObjectID]
  - auctioneer: UserID of the user that is selling this auction [type: int/ObjectID, the same as UserID]
  - itemBeingAuctioned: itemID of the item being auctioned [type: int/ObjectID, the same as itemID]
  - timeRemaining: The time remaining until the auction expires [type: datetime/String]
  - auctionStatus: The current status of the auction [type: Enum]
  - listOfBids: A list of all the bids on the auction [type: List of bids]

4.4.2 Database

We will be using the NoSQL database, MongoDB. NoSQL databases are much more scalable and flexible in their design, and MongoDB stores data as BSON (a JSON-like object) objects, enabling us to store objects and lists of objects inside other objects as well, thereby adding a layer of complexity without being too complicated. NoSQL databases are also better for handling larger amounts of flexible data (such as auctions and items), and performs sufficiently well for less complex queries, which happen to be most of the queries that we will be employing for this application. MongoDB is also a more generalized NoSQL database, not specializing in a particular optimization, which is fine for our purposes.

4.5 External Interface Requirements

4.5.1 User Interfaces

The user interface would be compatible with all browsers such as Google Chrome, Mozilla etc. allowing the user to utilize the software easily and allow them to use the software's functionalities to the fullest. The software would interact with the users so that the interface is visually appealing and convenient. The GUI standards are kept in mind. The main page is kept simple and easily navigable so that the actual content is not affected. Our color scheme is easy on the eyes and the colors contrast well with each other. In addition we do not use colors that are indistinguishable for people with red-green color blindness (the most common kind of color blindness).

4.5.2 Hardware Interfaces

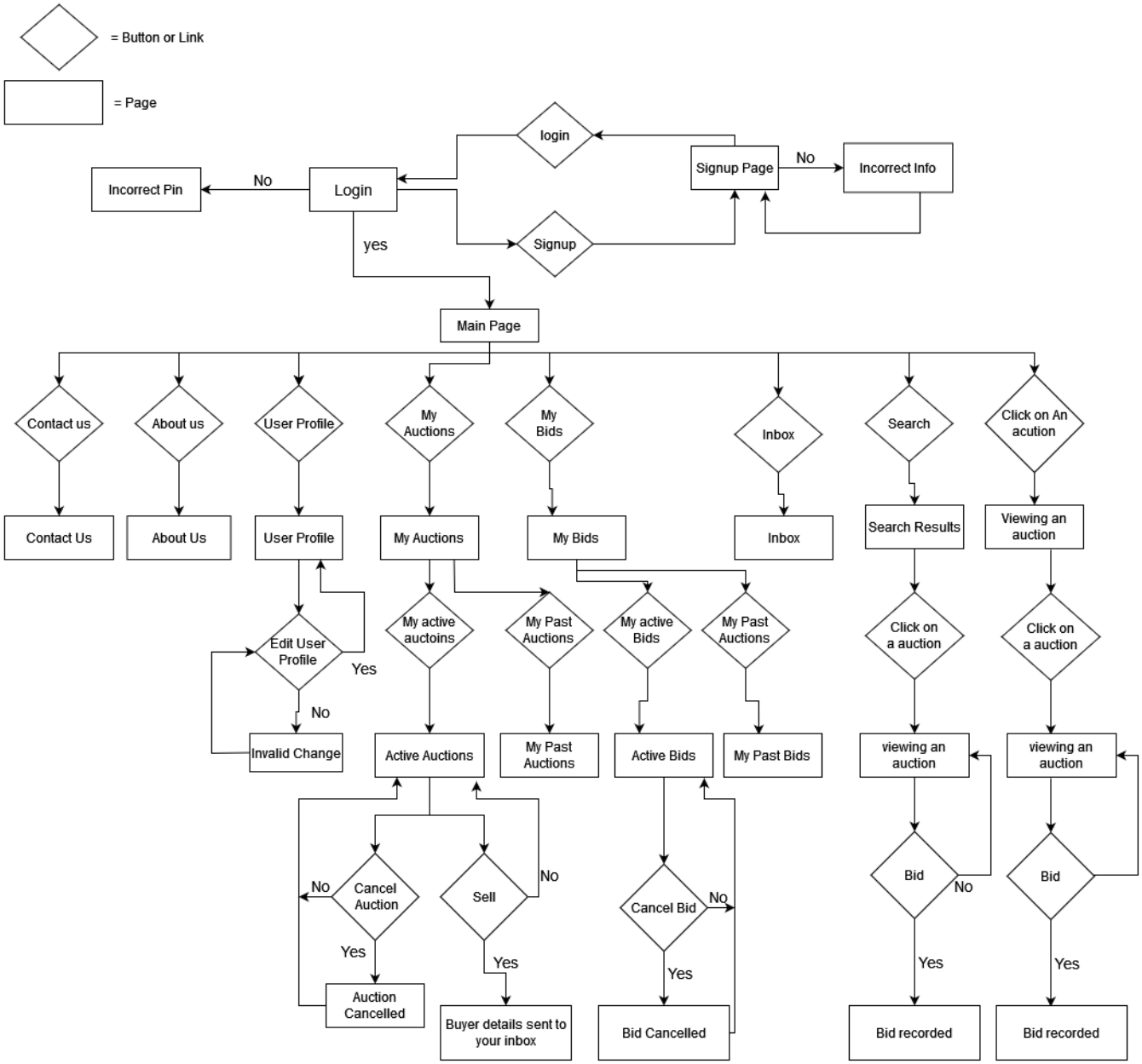
One of the major hardware interface requirements is the ability of the system to connect to the internet as our bidding website would only be accessed provided the internet connectivity. The hardware and software both should be compatible with each other for optimum efficiency.

5 User Interface Design

5.1 Description of the user interface

The front end technology that we will be using is React.js. We have chosen React as it is a well established library with a large array of developer tools and it fits well with the MERN stack. Since React.js employs the MVC architecture, our presentation layer will be comprised views and controllers, including but not limited to hyperlinks, forms, and buttons bound to onclick functions that execute appropriate routing and WebAPI calls to the business layer.

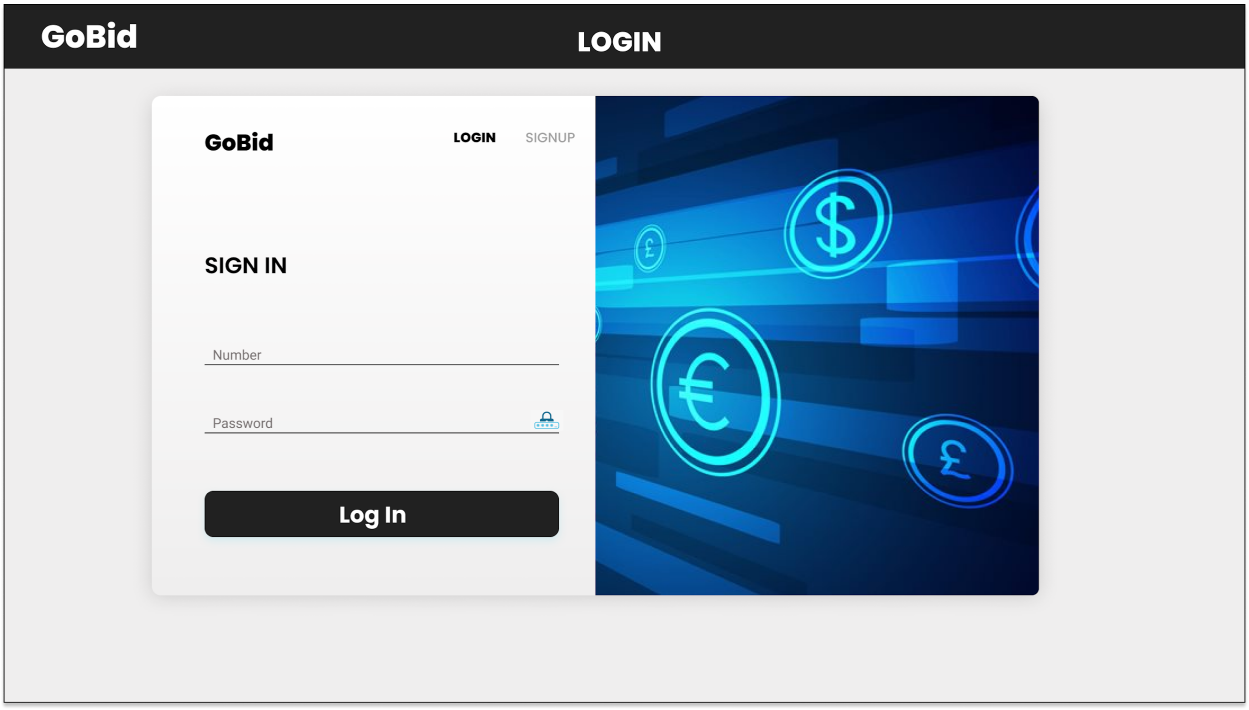
5.2 Information architecture



5.3 Screens

To do: Screen images with description – explain each item on the screen e.g. button, text field, etc. Explain how the screen is mapped on one of the user requirements/use cases

- **Login Page:**



Description:

This is the login screen of the website, and also serves as the onboarding page to the website. The goal of this screen is to allow the user to log into their account.

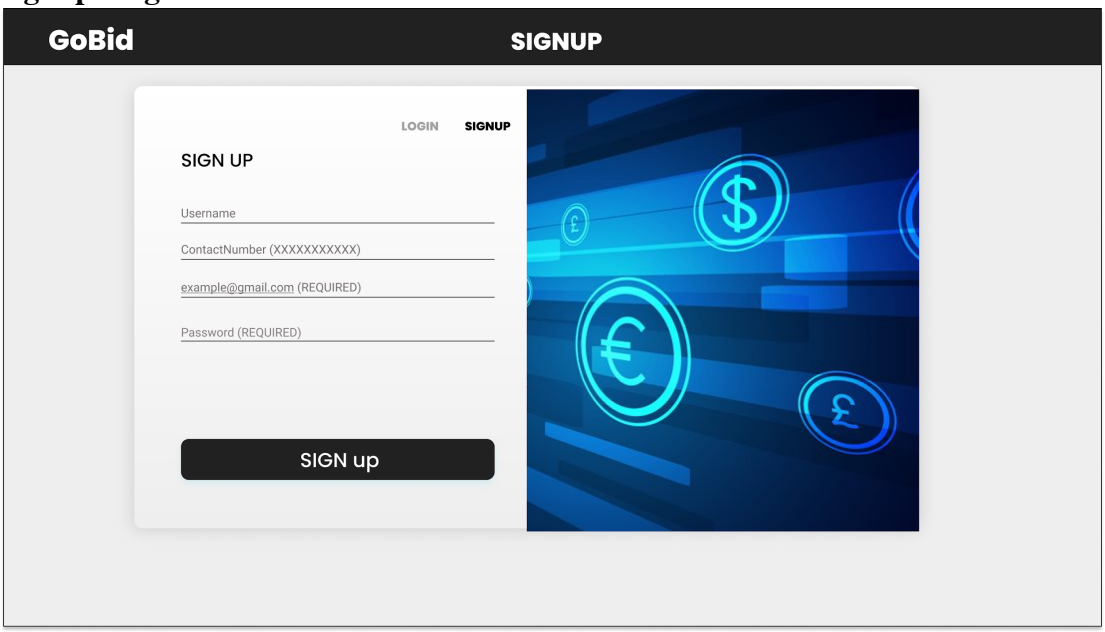
Components:

The screen has 2 text fields to allow the user to enter their phone number and password, and then the login button.

Login button redirects the user to the main page of the website after validating their info.

It has a link to the signup page to allow the user to make a new account.

- **Signup Page:**



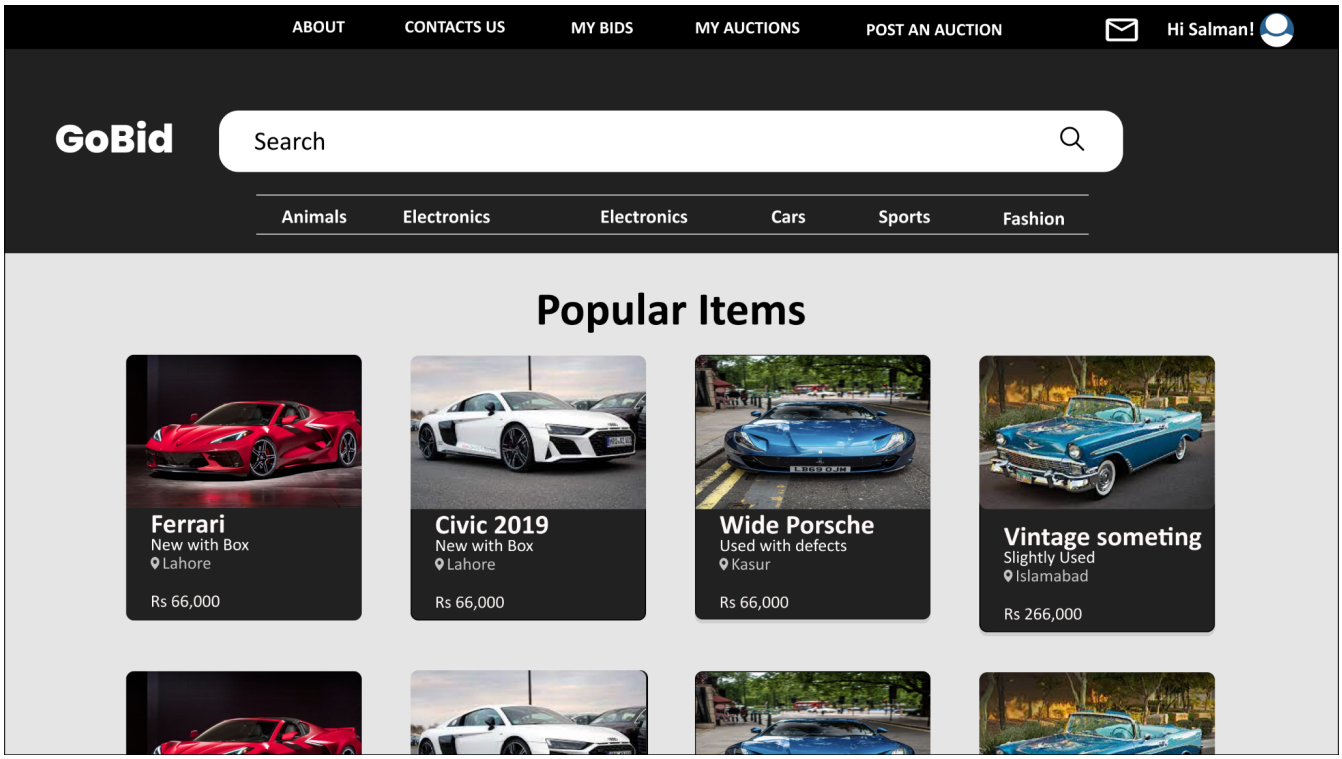
Description:

This is the signup screen of the website. If the user is new to the website, they can make an account in order to use the website.

Components:

- The screen has some text fields to get the details of the user and a link to go back to the login page.
- The signup button checks the info of the user and redirects to the login page.

- **Main Page:**



Description:

The main page is the area shown to the user when they login. It has a list of popular auctions (ones with the most bids) displayed. It has links to different categories and services, as well as the inbox and the search bar. The theme selected is a dark gray and black.

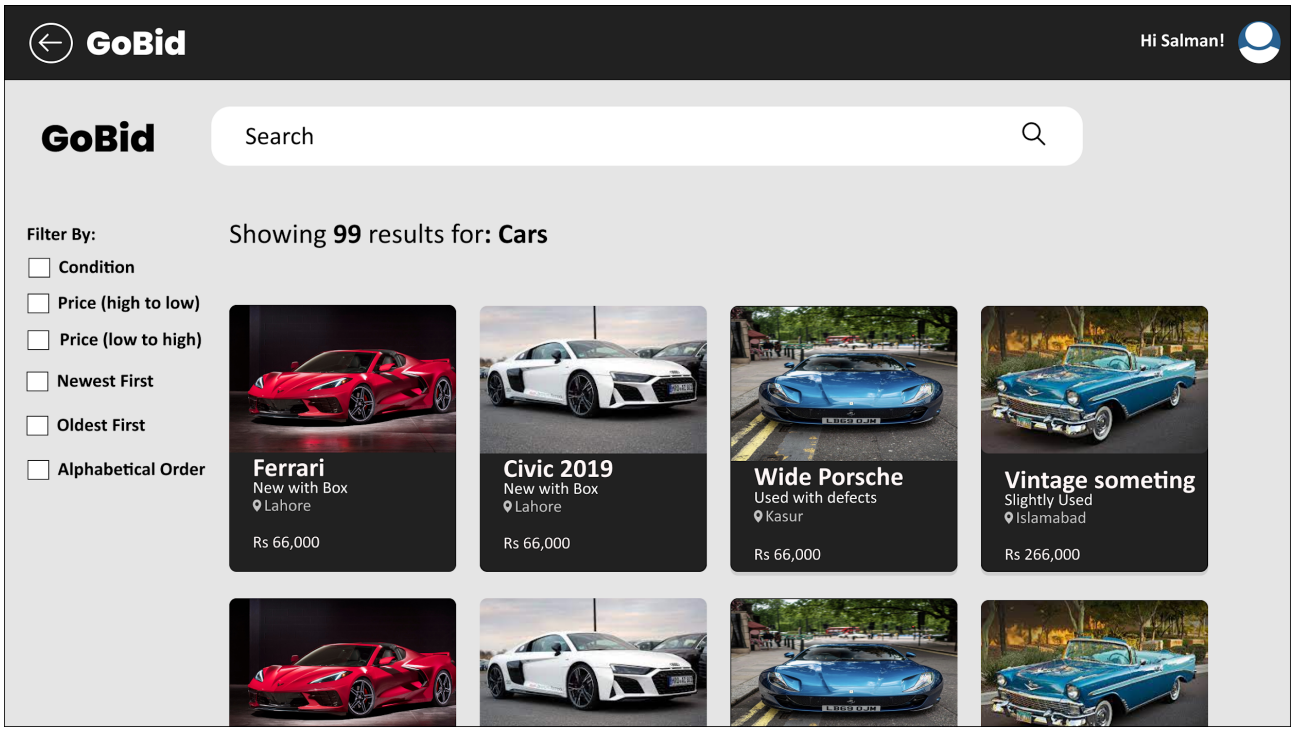
The goal of this page is to act as a homepage where the user can browse popular items and search for items, as well as access the full functionality of the website.

Components:

- On the top are links to the about us an contact us pages, as well as:
- MY BIDS: redirects the user to My bids page.
- MY AUCTIONS: redirects the user to the My Auctions page.
- POST AN AUCTION: redirects the user to Post an Auction page where they can list an item for auction.
- Inbox Icon: redirects the user to their inbox, where they view their messages.
- Profile: This is displayed in the form of the username. The user can click this to access their user profile.
- SEARCH: Used to search the website for relevant auctions. redirects the user to the search results page.
- Categories: The categories listed below the search bar can be clicked to filter out popular items according to the category clicked.

- auctions: Clicking on any auction (the cards) will redirect the user to that auction's view page.

- **Search Results:**



Description:

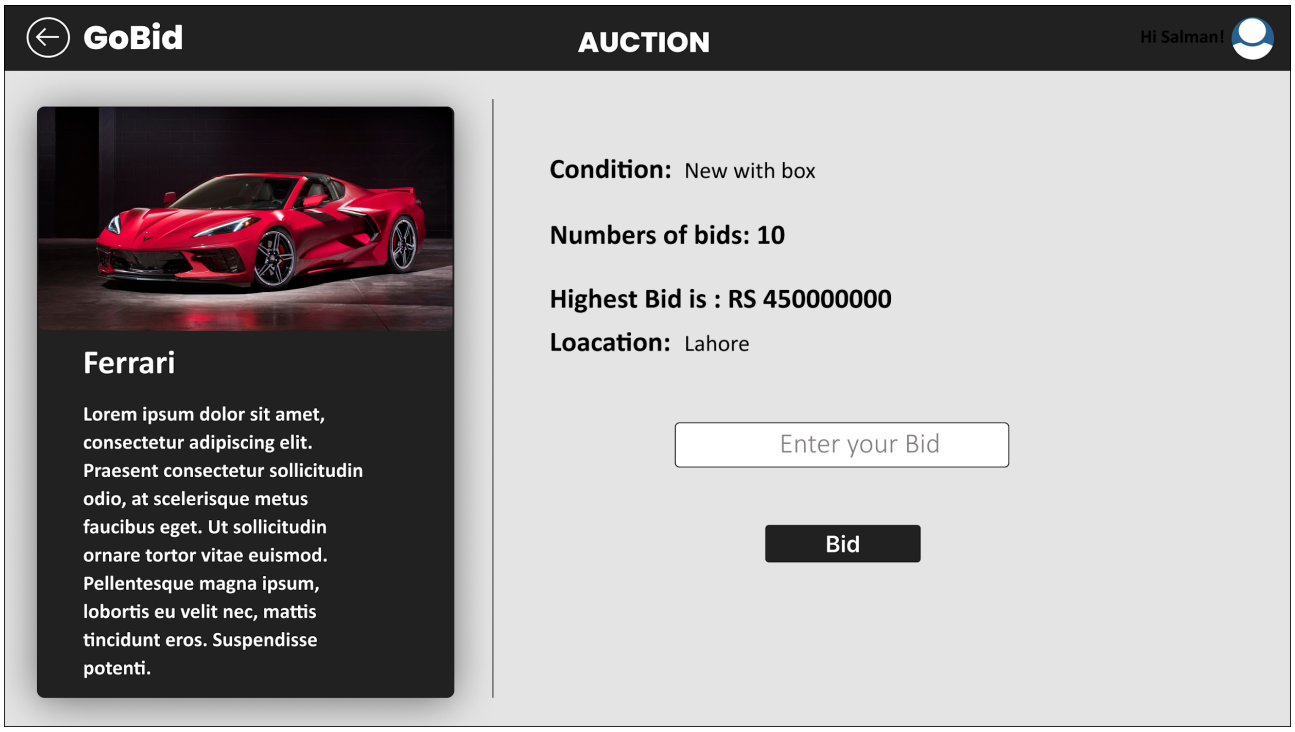
The search results page shows a list of auctions that are a result of a particular search based on the name of the auction. It also allows the user to filter by some parameters.

Components:

the parameters are:

- a search Bar: allows the user to search again.
- auction placards: shows a preview of the auction. clicking on it redirects the user to that auction's page.
- Filter By: these are checkboxes used to filter the result based on Condition, Price, time, or alphabetical order.
- Back Button: redirects user back to the main page.

- **Auction Page:**



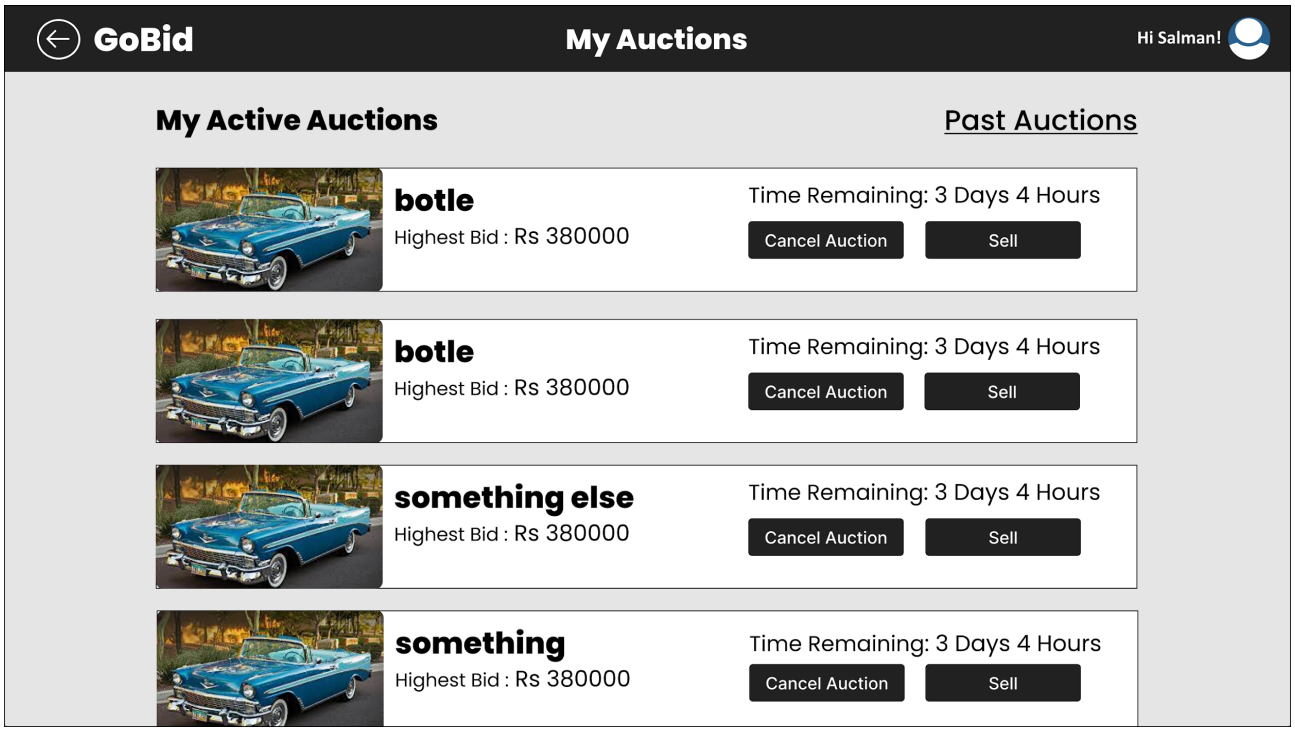
Description:

This page shows the details regarding an auction. it shows the number of bids on that auction, the description, the highest bid on that auction, and the location.
The goal is to provide details about the auction and let the user bid on the auction.

Components:

- Text Field: so that the user can enter the amount they are willing to pay.
- Bid button: This button posts the bid to the auction if it is greater than the current maximum.
- back Button: takes the user back to the search results/main page, depending on where the user came from.

- **My Auctions:**



Description:

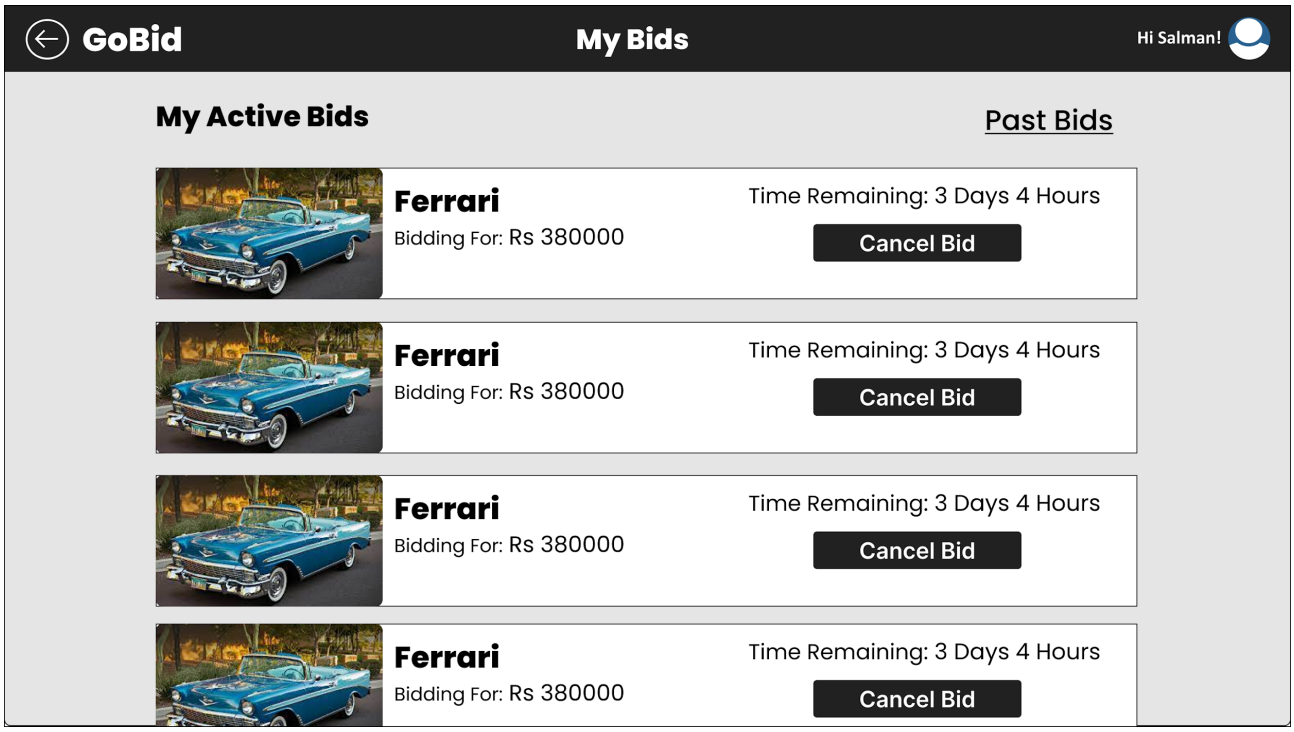
The My auction page has a list of user's active auctions and the current highest bid on it. It also provides additional functionality of concluding the auction or cancelling it.

Components:

The components are:

- Auctions: the list of auctions has 2 buttons, sell and cancel auctions. It shows the name of the auction and the highest bid on that auction so far.
- Sell Button: Sells the item. It will send the seller and buyer's contact info to each other.
- Cancel Auction: cancels the auction.
- Back Button: takes the user back to the main page.

- **My Bids:**



Description:

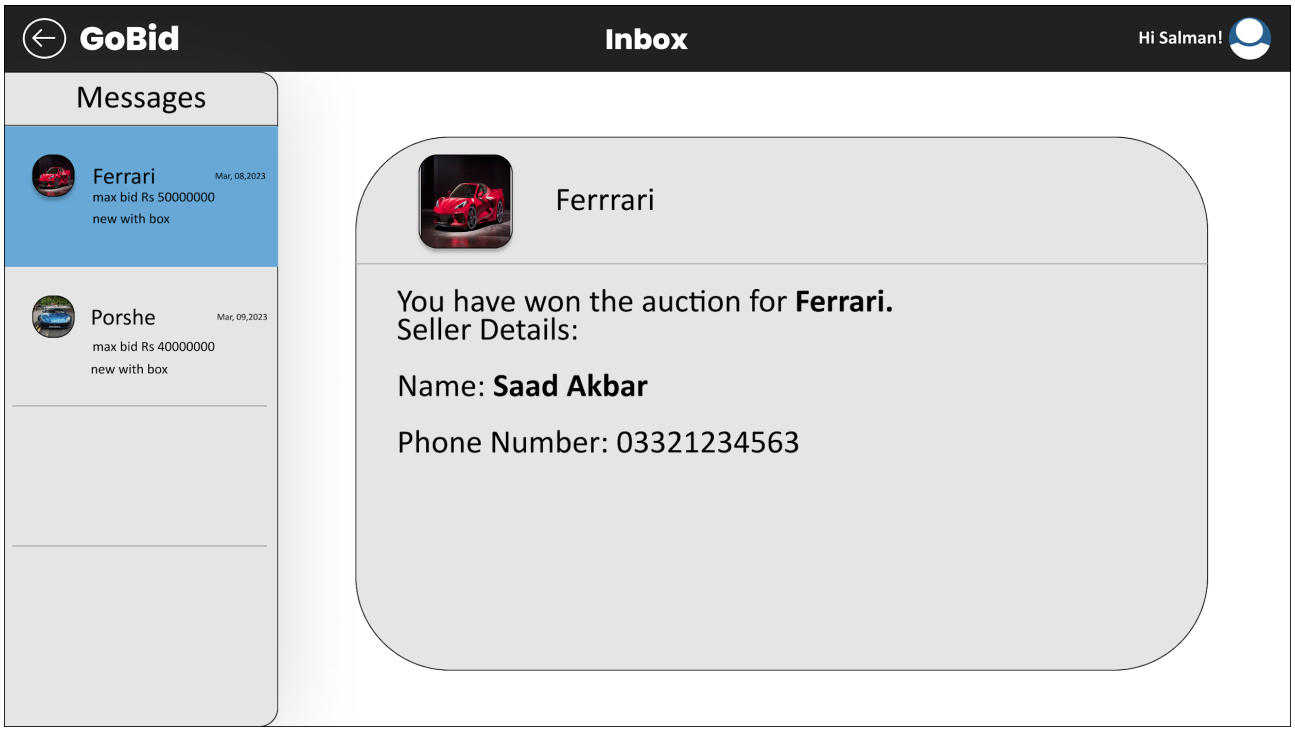
The My auction page has a list of user's active bids and the current highest bid on it. If the user gets outbid on any auction, it is removed from this screen and a message is sent to the user's inbox.
It also provides additional functionality of canceling the bid.

Components:

The components are:

- Bids: the list of Bids. Each has the cancel button. It shows the name of the auction and the highest bid on that auction so far.
- Cancel Auction: cancels the Bid.
- Back Button: takes the user back to the main page.

- **Inbox:**



Description:

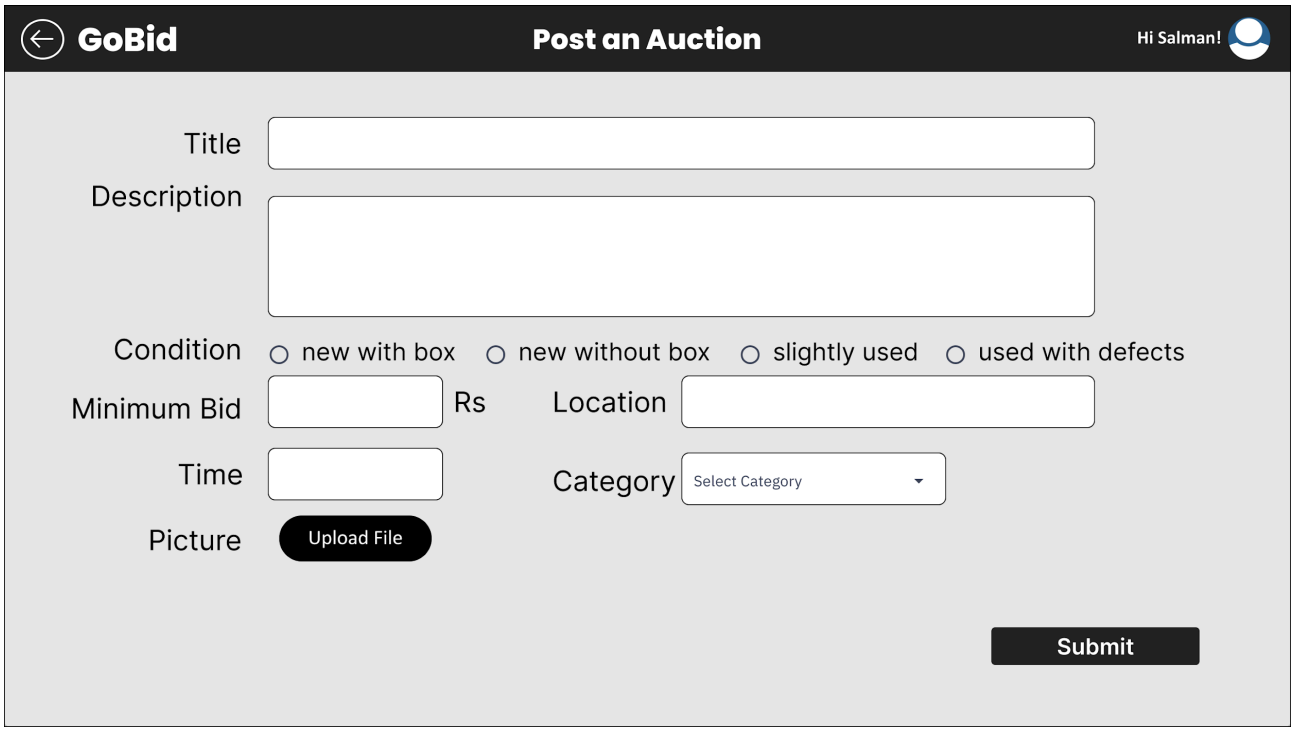
The Inbox is a way for the system to communicate with the user. Any messages regarding the user's auctions, or their bids are sent to this inbox.

Components:

The components are:

- Messages: the messages are from the system, such as if they have been outbid, or if they have won an auction etc.
- back button: Takes the user back to the main page.

- **Post An Auction:**



Description:

This page is a form for details of an auction that the user wants to post an auction. the page validates the information and redirects to the main page. A message is sent to the user's inbox on successful post.

Components:

The components are:

- Text fields: There are text fields for item Name, description, Condition, minimum bid, location, time and category.
- There is a upload file button for uploading pictures.
- Submit button to post an auction.

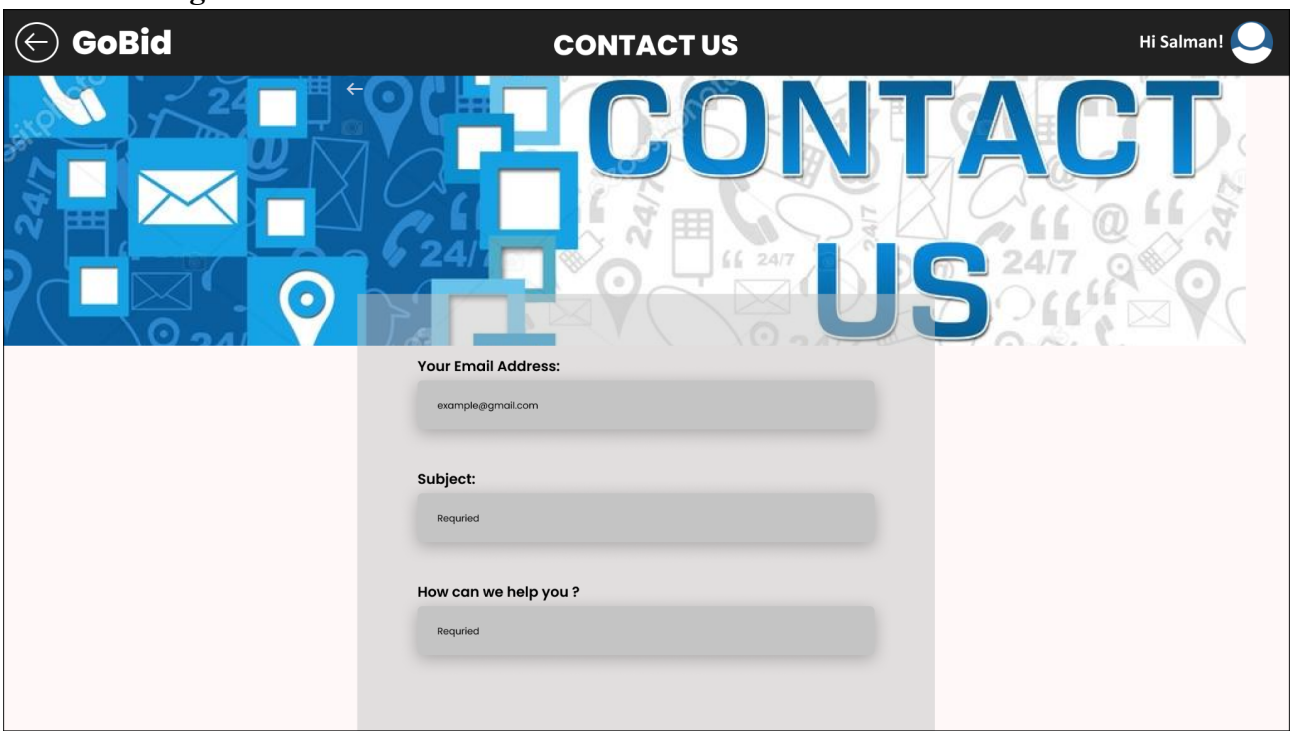
- **About us Page**

Description:
After clicking the About us button, the user will be directed to this page which provides the details of our website and its purpose of development. The goal of this page is to inform the user about out application and its operation

Components:

- On the top are is the back button which would redirect us to the main page
- It also has a profile icon which when pressed allows the user to access their user profile

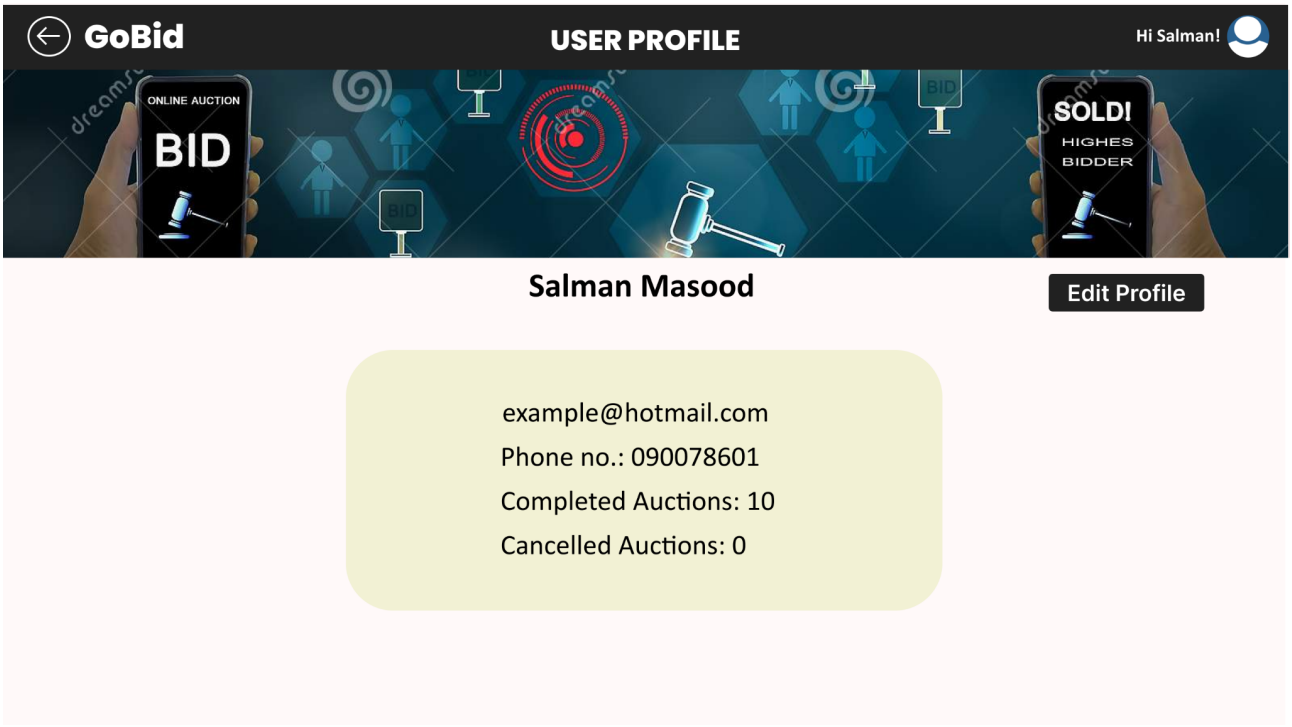
- **Contact us Page**



Description:
After clicking the Contact us button from the main page, the user is directed to this above shown screen which asks the user to provide their email address and subject for contacting us along with the concern that they want to deliver in text form. The goal of this page is to help users get in touch with GoBid and resolve users' concerns and make the application improve and more user friendly.
Components:

- On the top are is the back button which would redirect us to the main page
- It also has a profile icon which when pressed allows the user to access their user profile
- Text areas are provided for the user to enter their information
- To submit their concern, a send button is their which on pressing would let the developers know the user's concern

- **User Profile Page**

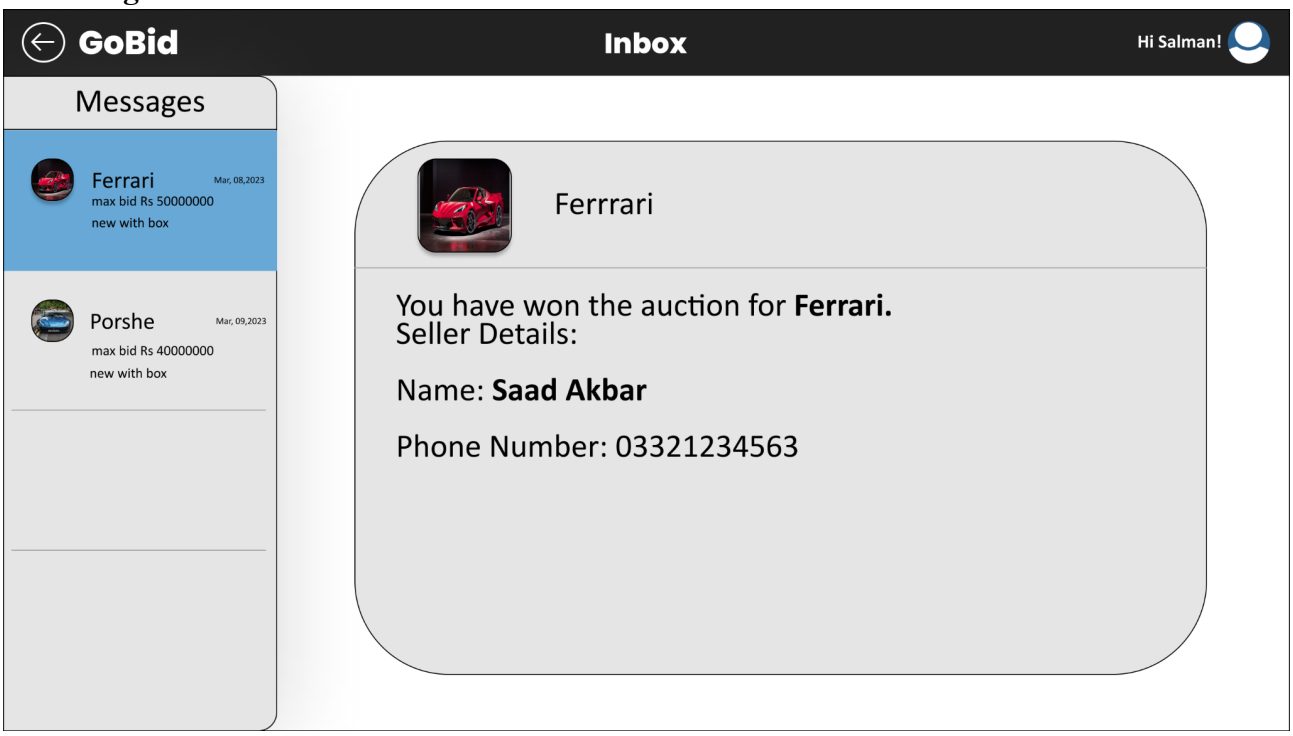


Description:
When the user presses the profile icon shown top right on the bar, they are directed to this screen which allows the user to access their profile details (for example email address, phone no. , total completed auctions and their total canceled auctions etc.). The user can also edit their profile details if required from the "Edit Profile button". The goal of this page is to allow the user to check their associated details by one-click away.

Components:

- On the top are is the back button which would redirect us to the main page
- It also has a profile icon which when pressed allows the user to access their user profile
- There is an Edit profile button that allows the user to make changes to their profile.

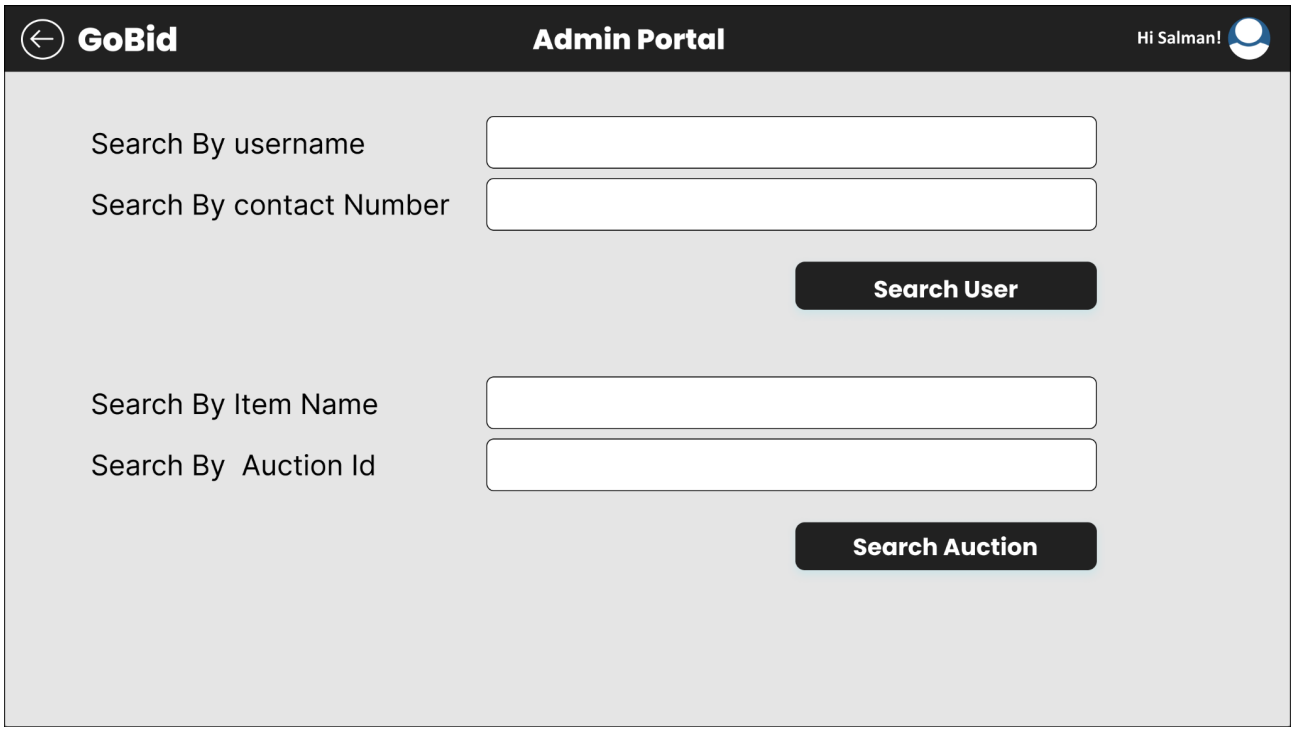
- **Inbox Page**



Description:
This page shows the user all the auctions that they have bid for. It shows the user whether they were successful in winning the auction or not. It also shows the seller's details. The goal of this page is to make the user aware of their bids and how many auctions they have winned.
Components:

- On the top are is the back button which would redirect us to the main page
- It also has a profile icon which when pressed allows the user to access their user profile
- On the left side it has a panel which shows all the bids of a particular user. These playcards are clickable and when clicked pops out the description and status of the bid.

- **Admin Portal**



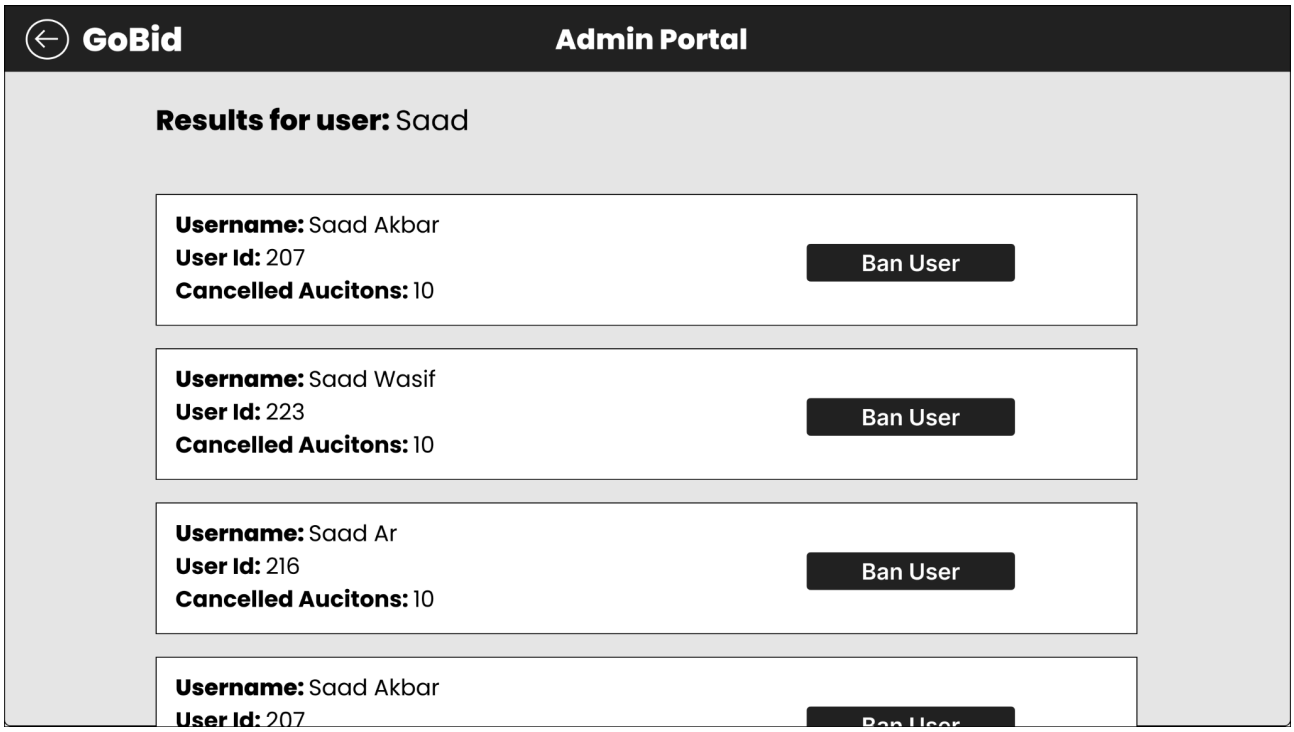
Description:

If the username and password are of an admin account, then the admin portal is logged in. it provides the admin to search for any user or auction.

Components:

- Text fields: The text fields are for username, and contact number. The admin can search either by username or contact number or both.
- Search User button: Redirects the admin to the display users page.
- Text fields: The bottom two text fields are for item name and auction ID. The admin can search either by item name or auction ID or both.
- Search Auction button: Redirects the admin to the display auctions page.

- **Admin Portal: User Search**



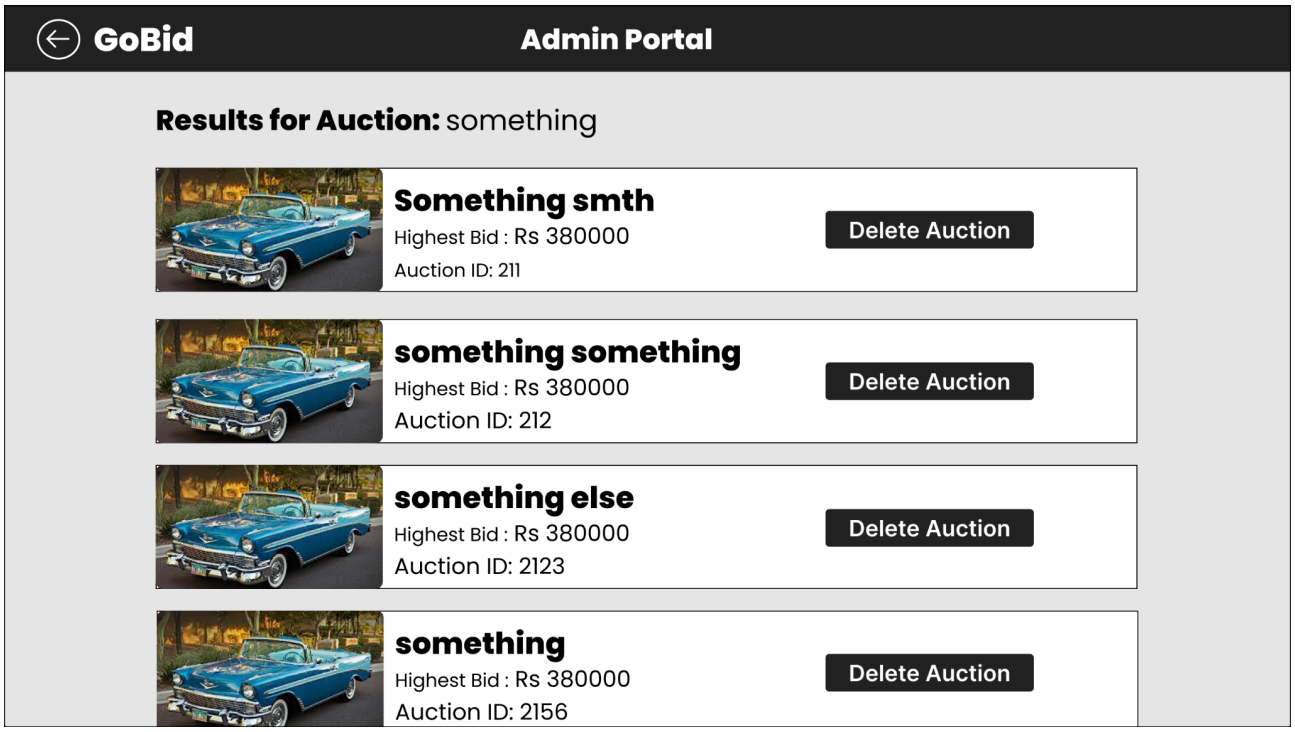
Description:

This is the user list that the admin can search through. they can ban any user here.

Components:

- Ban User Button: bans the user.
- Back Button: redirects the user back to the admin portal.

- **Admin Portal: Search Auctions:**



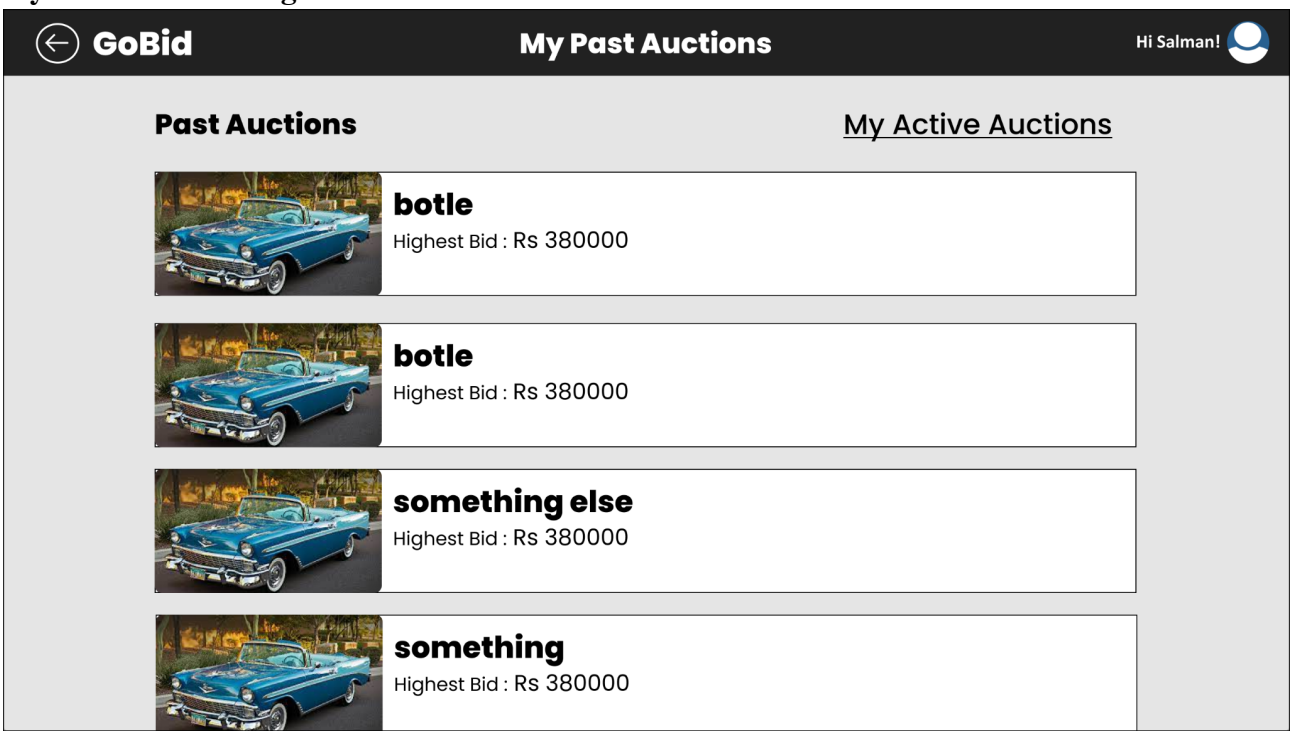
Description:

This is the Auction list that the admin can search through. they can delete any auction here.

Components:

- Delete User Button: deletes the auction.
- Back Button: redirects the user back to the admin portal.
- Clicking on an auction: Redirects the admin to the auction page.

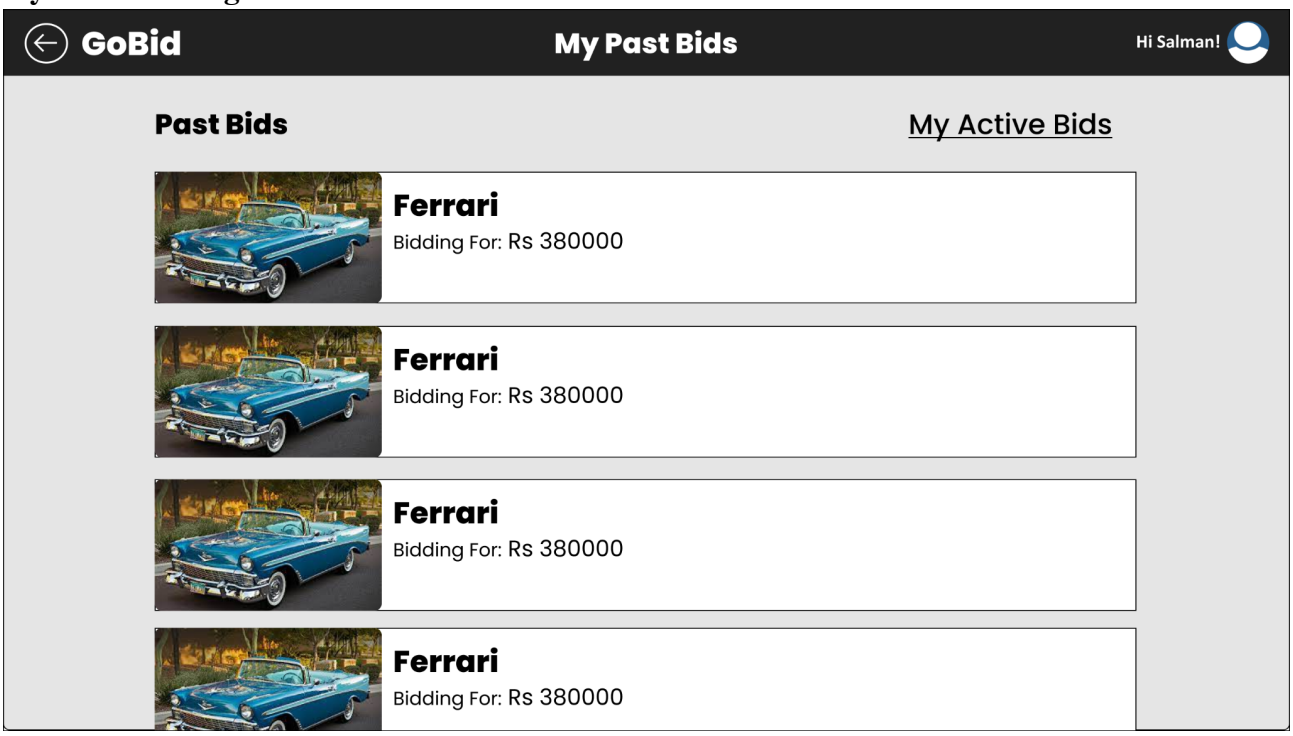
- **My Past Auctions Page**



Description:
When the user clicks the link on the my auction list (active), they are directed to this screen that shows the user all the past auctions that they have participated in. Their past auctions are displayed in the form of a list. If the user wants to go back to the "My active Auctions" screens they can click the link provided on the top right of "My active Auctions". These playcards displaying the items are not clickable. The goal of this page is to let the user know all their past auctions made.
Components:

- On the top are is the back button which would redirect us to the main page
- It also has a profile icon which when pressed allows the user to access their user profile
- The "My Active Auctions" clickable link will redirect the user to their active auction page.

- **My Past Bids Page**



Description:
When the user clicks the Past Bid link on the active Bid page, the user is directed to this page. This page provides the user all their past Bids that they have made in a form of a list. The user can go back to their active bids page by clicking on the “My active Bids” link. These playcards displaying the items are not clickable. The goal of this page is to let the user know all the past bids made in an efficient way.

Components:

- On the top are is the back button which would redirect us to the main page
- It also has a profile icon which when pressed allows the user to access their user profile
- The “My Active Bids” link redirects the user to their active bids page

5.4 User interface design rules

Color scheme:

The colors selected for the website are:

- Primary: gray
- secondary: black
- white for text in black backgrounds
- a lighter shade of gray for text in certain places
- another shade of gray for background.

The colors are consistent with Google's material design guidelines. The colors work well with each other. The background is gray, as to make the contrast not too sharp against the cards, and it is not too bright, but not too dim, so it does not look jarring and hurt the user's eyes.
The use of such colors also accommodates the color blind people, as no colors have been used that they cannot see.
User Interface:

The UI, consisting of symbols, back buttons, the layout of the main page and auction page, are inspired from retail websites like eBay, Daraz, and amazon. But we have also made some improvements which we think will make the overall user experience better (background is less white to reduce eye strain, and the layout is less cluttered). These elements are also very common among the web, and the navigation tools are in very conventional places like the back button being in top left, etc. the buttons used are uniform throughout the website so as not to cause confusion.

6 Other Non-functional Requirements

6.1 Performance Requirements

- The software should be secure enough to prevent security breaches.
- Should be easily navigable
- Consistent and easy to use interface
- Quick response time
- Ensure maintainability and scalability without compromising usability
- The system should be able to handle large amounts of traffic without crashing
- In case of a crash, recovery with minimal loss should be ensured

6.2 Safety and Security Requirements

- Any user data, passwords, emails, contact numbers should be safely protected from security attacks.
- The software should have measures against people trying to exploit the system. Once identified they should be permanently banned.
- Scammers, once identified, should be permanently banned.

6.3 Software Quality Attributes

Adaptability:
The software should be adaptable and ensure that it provides the users products of their interests based on their searches and it should also adjust advertisement of products in accordance with the user's location. This could be achieved by analyzing the user's search preferences and interest and from frequent bidding activities from a particular location. When the user searches for a certain bit their searches would be stored in cookies which would help the software to customize and adapt based on user need.

Robustness:
The software should be robust enough to deal with security breaches and system crashes and this would be achieved with continuous and rigorous testing of the software and ensure that all the corner cases have been properly dealt with. The software would be passed from component testing every time a new component has been developed and then would be integrated to the software and then a complete test would be performed on the entire software to ensure that every component and module is aligned with each other.

Usability:
The software would be designed in such a way that it is easy to navigate, have minimum gulfs and would be intuitive so that it could be used by anyone who accesses the website without any mistake or hurdle. It can be used without any need of training for the software. It would have visual affordance and to ensure its usability it would follow all the norman's principles of usability such as the principle of affordance, mapping, feedback etc..

Portability:
The software would be able to run on any machine that has access to the internet and would be compatible with all the web browsers and to ensure portability we would make the software light with minimal hardware limitations and would develop the website on a platform that is compatible with all the requirements of a machine. Since it would be a website, there is no need for the installation of this software as it would be readily available once the user enters the URL of the website.

Appendix A - Group Log

*<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist **ME and the Teaching Assistants** to determine the effort put forth to produce this document>*

Appendix B – Contribution Statement

Name	*Contributions in this phase*	*Approx. Number of hours*	*Remarks*
Saad Akbar	*Screens, Architecture, User Interface*	*26*	
Syed Taimoor	*Overall design, non-functional requirements, subsystem architecture*	*26*	
Taimur Salman	*Database, System Architecture*	*28*	
Salman Masood	*Screens, User Interface*	*27*	