

# Mini Project

## CCS112 - Communication Models and Protocols

Submitted by the Members of **Group 20**

	<b>Name</b>	<b>Registration Number</b>
1	Mohomad Iffath Hana( <b>Group Leader</b> )	21UG0092
2	Janith Kavinda Hettiarachchi	21UG0228
3	Kesara Mahima Lakpriya	21UG0166
4	Oshan Wathsala	21UG0771
5		

Design the protocol in a **diagram of the client-server model** highlighting the bid timing mechanism through the diagram or otherwise.

```

175 print(time_extd)
176
177 if bid_end_check(expiry_date) > 60 or time_extd == "TRUE":
178
179     print(bid_user_id, item_id, bid_price, expiry_date)
180     sql_q5="INSERT INTO bid (bid_id, item_id, bid_user_id, bid_price) VALUES (NULL, %s, %s, %s);"
181     value = (item_id, bid_user_id, bid_price)
182     print(bid_price)
183     db_con.execute(sql_q5, value)
184     data_base.commit()
185
186     bid_noti_sql = "SELECT name, base_price, max(bid_price) FROM item LEFT JOIN bid ON item.item_id = bid.item_id WHERE item.item_id = %s;" % item_id
187     db_con.execute(bid_noti_sql)
188     bid_noti_arr = db_con.fetchall()
189     print(bid_noti_arr)
190
191     Title = bid_noti_arr[0][0]
192     Base_Price = bid_noti_arr[0][1]
193     Max_bid = bid_noti_arr[0][2]
194
195     new_bid_arr = {"Type": "New_Bid", "item_id": item_id, "name": Title, "Base_Price": Base_Price,
196                  "Max_bid": Max_bid}
197     Notification_Type = "New_Bid"
198     mu = threading.Thread(target=new_item_noti, args=(Notification_Type, new_bid_arr))
199     mu.start()
200
201 if bid_end_check(expiry_date) <= 60 and time_extd != "TRUE":
202     current_time = str(datetime.now())
203     current_time_expand = current_time[:19]
204     date_format_str = '%Y-%m-%d %H:%M:%S'

```

```

199 mu.start()
200
201 if bid_end_check(expiry_date) <= 60 and time_extd != "TRUE":
202     current_time = str(datetime.now())
203     current_time_expand = current_time[:19]
204     date_format_str = '%Y-%m-%d %H:%M:%S'
205     time_convert = datetime.strptime(current_time_expand, date_format_str)
206     print("time_convert", time_convert)
207     extended_time = 60 + bid_end_check(expiry_date)
208     final_time = time_convert + pandas.DateOffset(seconds=extended_time)
209     final_time_str = final_time.strftime('%Y-%m-%d %H:%M:%S')
210     print("final_time_str", final_time_str)
211
212     print(bid_user_id, item_id, bid_price, expiry_date)
213     sql_q5 = "INSERT INTO bid (bid_id, item_id, bid_user_id, bid_price) VALUES (NULL, %s, %s, %s);"
214     value = (item_id, bid_user_id, bid_price)
215     print(bid_price)
216     db_con.execute(sql_q5, value)
217     data_base.commit()
218
219     DateTime_sql = "UPDATE item SET expiry_date = %s, time_extd = 'TRUE' WHERE item_id = %s;"
220     val = (final_time_str, item_id,)
221     db_con.execute(DateTime_sql, val)
222     data_base.commit()
223
224     bid_noti_sql = "SELECT name, base_price, max(bid_price) FROM item LEFT JOIN bid ON item.item_id = bid.item_id WHERE item.item_id = %s;" % item_id
225     db_con.execute(bid_noti_sql)
226     bid_noti_arr = db_con.fetchall()
227     print(bid_noti_arr)
228

```

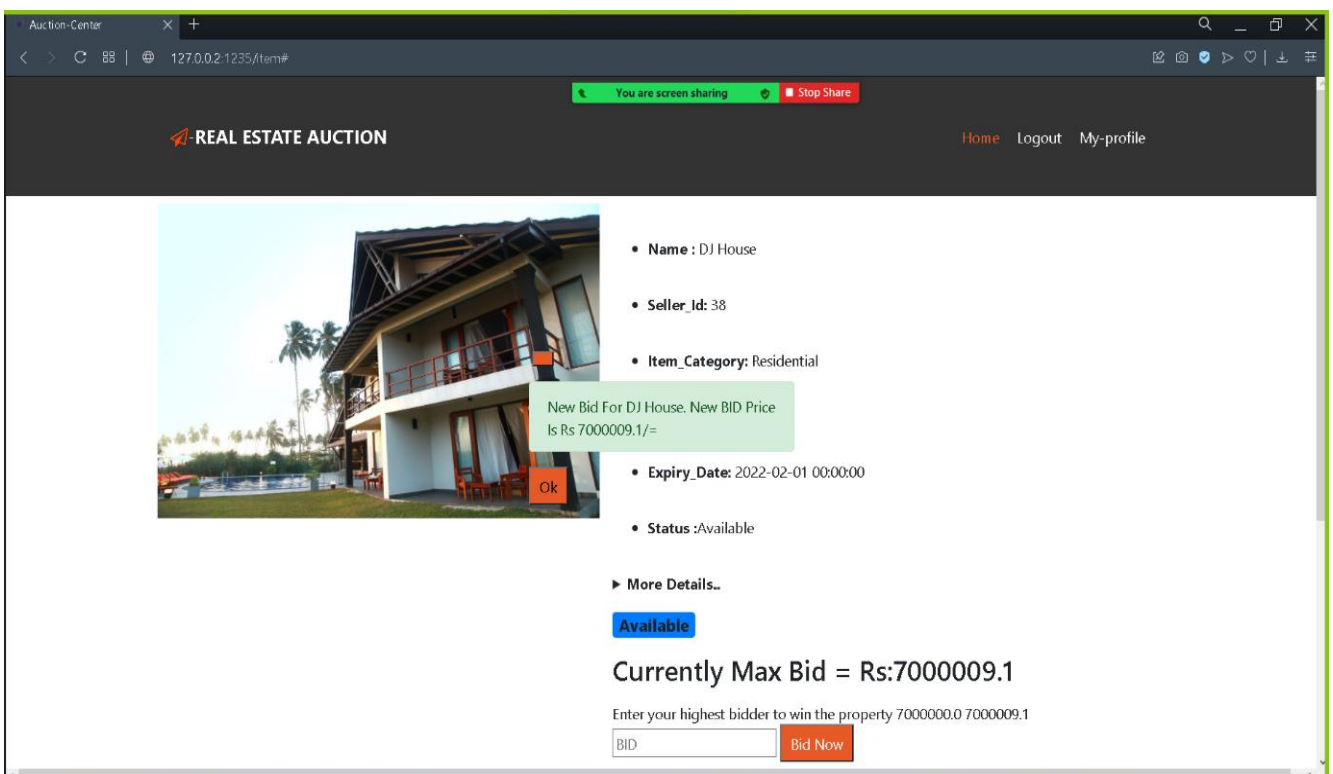
```

407         if sub_ip == (len(broadcast_ip_arr)-1):
408             sub_ip=0
409             break
410
411     def bid_end_check(Expiry_Date):
412         while True:
413             time_str = Expiry_Date
414             date_format_str = '%Y-%m-%d %H:%M:%S'
415             expiry_time = datetime.strptime(time_str, date_format_str)
416             now_time = datetime.now()
417             duration = expiry_time - now_time
418             duration_in_sec = duration.total_seconds()
419             print("bid_end_check", duration_in_sec)
420
421         return duration_in_sec
422
423     def bid_end():

```

Design the protocol in a **diagram of the publisher-subscriber model** showing the information exchange between the publisher-subscriber and the auctioning server.

- The auction server sends continuous notifications to the subscriber till he or she confirms that the notification has been received. This is the way information interchange in the system we have developed. Screenshots of the above process are attached below.



Describe a mechanism to ensure that the client **ID can be kept unique** for each user.

- When a client logs in to the system, an IP Address is generated for the client. This is used as the Client's ID throughout the session. Messages will be sent to the client based on the generated IP Address. This is performed by assigning each client with a new Python client file.

Clearly **justify why a publisher-subscriber model is suitable** for bid and profit updates (as opposed to client-server).

A publisher subscriber model is suitable for bid and profit update because publisher subscriber model, the broker notifies the subscriber about any changes on their subscriptions. In this method end points don't connect directly. Publishers connect with brokers. Subscribers subscribe to different things available as they interested in such as products and services. Brokers usually don't store data. They simply move data from publishers to subscribers.

In client server model nothing like this can be done. Neither the client can subscribe to the things they interested in nor he or she receives any updates about the things they interested in. Notifications won't be appeared unless the client logged in. Furthermore, client has to request in order to find out about updates regarding the products/ services. In every aspects client may find it annoying and difficult.

Therefore, publisher subscriber model is suitable for bid and profit updates.

Clearly state **functional and non-functional requirements** of the publisher-subscriber model.

Functional Requirement

Server notifies the subscribers about new biddings and new items.

Non - Functional Requirement

Subscribers are notified without any delay.

Notifications are popped up without any interference.

Explain how you use transport layer protocol to ensure these requirements.

UDP (User Datagram Protocol) is the transport layer protocol which used to accomplish above stated functional as well as non-functional requirements. The server sends a notification to the subscriber which is considered to be a functional requirement using UDP protocol

State the additional functionality you add in the application layer to fulfill the requirements not guaranteed in the transport layer.

The application layer is involved in communication handling. The server sends a notification to the subscriber using UDP. There is no guarantee that the notification has reached the subscriber. Therefore, we keep sending the notification until the subscriber confirms that he or she received the notification. This is the additional functionality we have added.