**Project Description**

Managing an apartment relates to a lot of data such as renting prices, utility fees, resident and room information, etc. This huge amount of data makes it difficult for the apartment owner to collect all data in hand without any tools. Thus, we created a program for an apartment’s owner in Samutprakarn to use as a tool for managing their apartments. The previous program that the apartment has been using does not cover all the information the owner needs, such as type of room and residents and all the little details about the renting is also not included in the program. Moreover, the interface is not easy to look for the availability of each room. Therefore, we decide to create the program that is suitable for that apartment. We plan to create the program that enables the owner to finish all the tasks in just a single program. This program will shorten their time when working on tasks for the apartment so they could have more time to develop other things further.

The software acts as database to record the essential information about the room and the resident according to the law. This program can also generate the invoice to the resident.

For the booking function, the program can show the availability for the rooms for each day (can be the date today or the day in the future) so the user can see clearly when they book to room for the new coming resident. Furthermore, the user can also click to each room to see the information of the room or the data of the resident of that room.

The program has 2 modes of the input for the resident; monthly resident, and daily resident. Both types of the resident’s data will be recorded to the database.

For both types of resident, the database contains the general data of the resident such as room number, type of the room (studio, 1 bedroom, 2 bedrooms), name, date of birth, permanent address, check in date, Thai national ID or passport number. The daily resident contains additional data which is the check-out date while the monthly resident contains another additional data such as the record of each month’s utilities fees.

Moreover, for the monthly resident, the program can calculate the utilities bill from the raw data from the utilities meter and it can connect to the printer and print the bills out for distributing to the residents.

**Requirements**

**User requirements**

**Functional:**

1. User must be able to make a room reservation.
2. User must be able to store data of tenants.
3. User must be able to see the calculated total water fee for each tenant.
4. User must be able to see the calculated total electricity fee for each tenant.
5. User must be able to see availability of the room conform with the calendar thoroughly.
6. User must be able to create bills of total expense for each tenant.
7. User must be able to see the separated tenant information from daily stay and monthly stay.
8. User must be able to issue bills and print from a printer.
9. User must be able to login to his corresponding account.
10. Unknown user (i.e., tenant) must not be able to access the program.

**Non-functional:**

1. Program should provide easy to use GUI (e.g., easy to read font).
2. Program should respond fast (e.g., take small amount of time to log in, calculate fee and fetch data).
3. Program should provide proper database for user to collect tenant data and user data.
4. Program should be robust.
5. Program should be easy to maintain.
6. Program should be able to display information of rooms and tenants correctly.
7. Program should prevent wrong type of input from the user.
8. Each button in the program should be as self-explanatory as possible to make program easy to use.
9. Program should be able to run effectively (i.e., consume small amount of resources)
10. User and tenant data should be able to update.

**System requirements**

**Functional:**

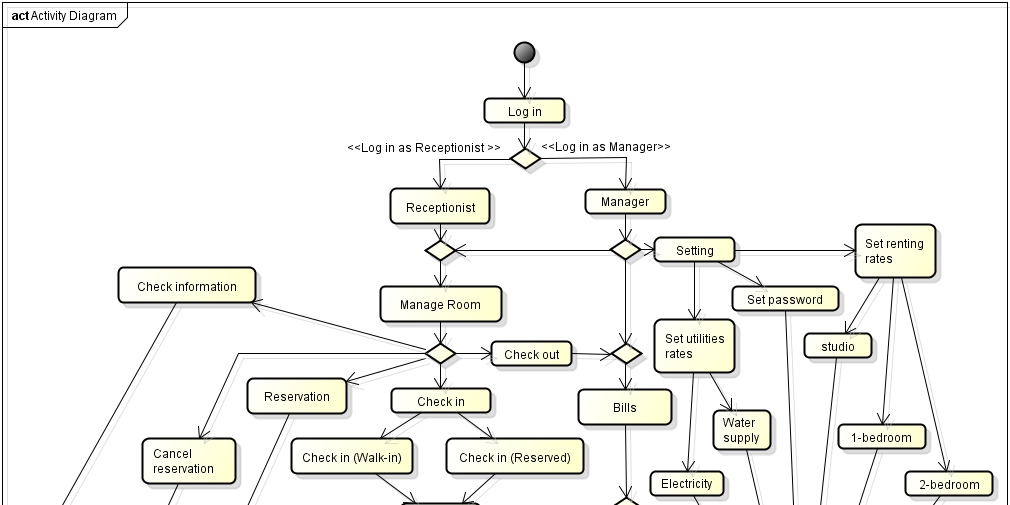
1. If room availability is false, the program must notify the user. Then program must not allow the user to book the room
2. Each tenant data must be allocated a unique identifier (TENANT\_ID), which must be able to store in the database.
3. Data of the tenant must contain: type of resident(TYPE\_OF\_RESIDENT), room number(ROOM\_NUMBER), type of the room(TYPE\_OF\_ROOM), name(NAME), birthdate(BIRTHDATE), permanent address(ADDRESS), check in date(CHECK\_IN\_DATE), Thai national id or passport number(ID).
4. Data of the daily resident must contain: checkout date (CHECK\_OUT\_DATE). While data of the monthly resident must contain: utility fees (UTILITY\_FEES).
5. The system must be able to work without network connection.
6. The system must be able to calculate total expense (TOTAL\_EXPANSE) for each tenant from UTILITY\_FEES, RENTING\_FEES.
7. If the user enters a wrong username or password, the program must display the error message.
8. The system must be able to save reservation information into the database if the user provides all required data.
9. The system must be able to cancel any reservation ordered by user.
10. The system must be able to open and operate normally in case of a system crash.

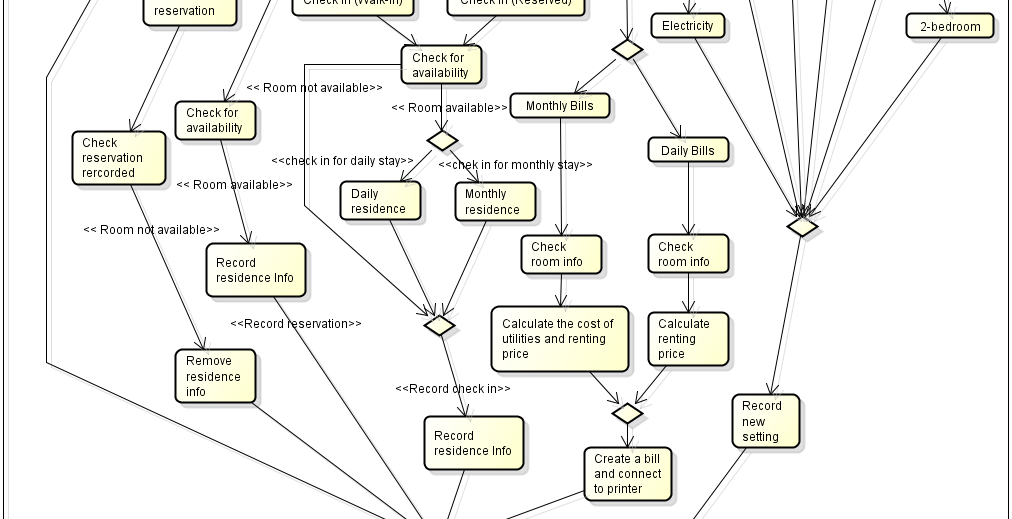
**Non-functional:**

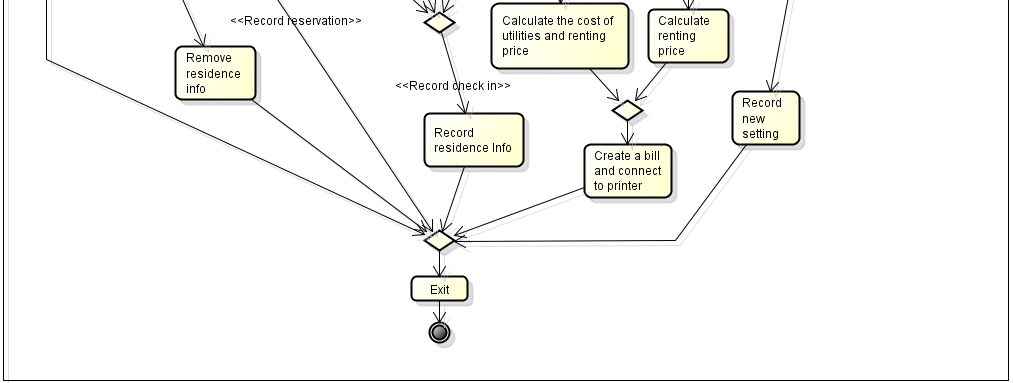
1. Size of the program must not be greater than 1 GB.
2. Modified data in database should be updated while accessing it within 2 second.
3. The user interface of the system shall be developed using PyQt.
4. The system shall be developed using python.
5. The system shall not disclose any personal information of the user apart from names, room number, and type of room. The system shall protect its data from the outsider.
6. The user shall receive a room status (available, not available) within 2 second after he requested for room availability.
7. The system run on Microsoft Windows 7 or newer versions of Microsoft Windows Operating System.
8. The system support only English language.
9. The system shall calculate the utilities bill within 10 seconds.
10. The system shall not lose any information in the database.

**UML**

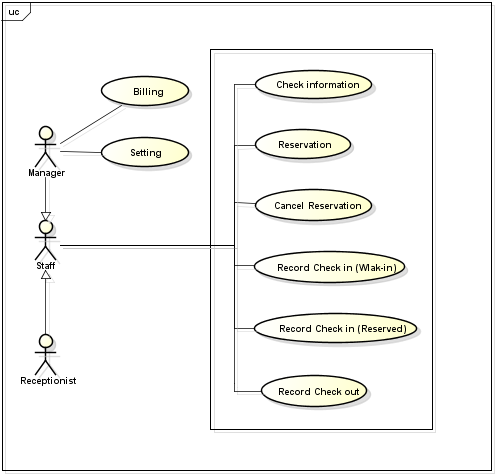
**Activity diagram**

****

****

****

**Use case diagram**



**Basic Course of Events**

**Check information**

1. The system allow receptionist or manager to click on the room button distinguish by room number and the color of button. Each color shows status of the room. Orange button is not available room or other residence is check in and check out within the correct reserved date, red button is not available room or other residence is check in and check out in the late day, and green button show an available room.
2. The receptionist and manager click the room button.
3. The system show information of room and residence. The information of room will always show but residence information will show when there is a check in for that room.
4. After the receptionist or manager click some room button, the system allow user to further click utility record button for seeing utility information for that room.

**Reservation**

1. The system allow receptionist or manager to search the available room. The user can search for all available room or specific room.
2. The system show information of available room in table.
3. If there are a rows show up in the table, the system allow receptionist or manager to click some row or column in the table.
4. If the receptionist or manager click reserve, the system will ask user to fill in their information the further do the reservation.
5. The receptionist or manager click reservation, then the system record their information in the database system and change status of the room to be reserved.

**Cancel Reservation**

1. The system show all information of all residence who reserved the room.
2. The system allow receptionist or manager to search the reservation of residence.
3. The system ask receptionist or manager to prompt first name and last name of residence if the user want to search their name for revered room.
4. The system show information of that revered room by that residence in table.
5. If there are a rows show up in the table, the system allow receptionist or manager to click some row or column in the table.
6. The receptionist or manager click cancel reservation.
7. The system will remove the reserved information from the database system and use case terminate.

**Record Check in (Walk-in)**

1. The system allow receptionist or manager to search the available room. The user can search for all available room or specific room.
2. The system show information of available room in table.
3. If there are a rows show up in the table, the system allow receptionist or manager to click some row or column in the table.
4. If the receptionist or manager click reserve, the system will ask user to fill in their information the further do the check in.
5. The receptionist or manager click check in, then the system record their information in the database system and change status of the room to be not available.

**Record Check in (Reserved)**

1. The system ask receptionist or manager to prompt their information of the already reserved user or allow them to search.
2. The system show their reserved information in the table.
3. The receptionist or manager click check in.
4. The system will record the check in information into the database system and change status of the room to be not available.

**Record Check out**

1. The system allow receptionist or manager to click on the room button distinguish by room number and the color of button. Each color shows status of the room. Orange button is not available room or other residence is check in and check out within the correct reserved date, red button is not available room or other residence is check in and check out in the late day, and green button show an available room.
2. The receptionist and manager click not available room button or the orange and red button.
3. The system show information of room and residence. The information of room will always show but residence information will show when there is a check in for that room.
4. The receptionist and manager click check out.
5. The system ask the receptionist and manager to confirm the checkout.
6. The room is marked as available. The system ask user to confirm the cancelation, if the answer is “yes”

**Billing**

1. The system asks the user for room number and water meter, electricity meter for that room.
2. The system calculates total cost by adding water cost, electricity cost and room cost for monthly resident type and using only room cost for daily resident type.
3. The system creates an invoice for the user.
4. The system save this invoice if the user choose to save it.

**Setting**

1. The system allow receptionist or manager to set the utilities rate for electricity or water, renting rate for each room type, or change their password for accessing into the program.
2. The system allow receptionist or manager to click update or default.
3. The receptionist or manager edit the information and click update.
4. The system updates the record into database system.

**Alternative Course of Events**

**Reservation:** There is no available room.

1. The system allow receptionist or manager to search the available room. The user can search for all available room or specific room.
2. The system show information of available room in table.
3. There is no information of available room.
4. The system not allow receptionist or manager to click reserve by disable the reserve button and use case terminate without reservation.

**Cancel Reservation:** No name in a searching list.

1. No information of revered room show in table.
2. The system not allow receptionist or manager to click cancel reservation by disable the cancel reservation button and use case terminate without cancel reservation.

**Record Check in (Walk-in):** There is no available room.

1. The system allow receptionist or manager to search the available room. The user can search for all available room or specific room.
2. The system show information of available room in table.
3. There is no information of available room.
4. The system not allow receptionist or manager to click reserve by disable the reserve button and cannot further do the check in operation.
5. Use case terminate without check in.

**Exceptional Course of Events**

**Reservation:** Residence not fill all needed information.

1. The system allow receptionist or manager to search the available room. The user can search for all available room or specific room.
2. The system show information of available room in table.
3. If there are a rows show up in the table, the system allow receptionist or manager to click some row or column in the table.
4. If the receptionist or manager click reserve, the system will ask user to fill in their information the further do the reservation.
5. The receptionist or manager not fill all needed information.
6. The receptionist or manager click reservation, then the system record their information in the database system and change status of the room to be reserved.
7. There is a null information recorded into the database system.

**Cancel Reservation:** User search without prompting their name.

1. The system ask manager to prompt first name and last name of residence for revered room.
2. The manager click search without entering first name and last name
3. The system display error, no reservation canceled and use case terminate.

**Record Check out:** User check out for available room.

1. The system allow receptionist or manager to click on the room button distinguish by room number and the color of button. Each color shows status of the room. Orange button is not available room or other residence is check in and check out within the correct reserved date, red button is not available room or other residence is check in and check out in the late day, and green button show an available room.
2. The receptionist and manager click available room button or the green button.
3. The system show information of room and residence. The information of room will always show but residence information will show when there is a check in for that room.
4. The receptionist and manager click check out.
5. The system show pop up cannot check out for available room.
6. Use case terminate without check out.

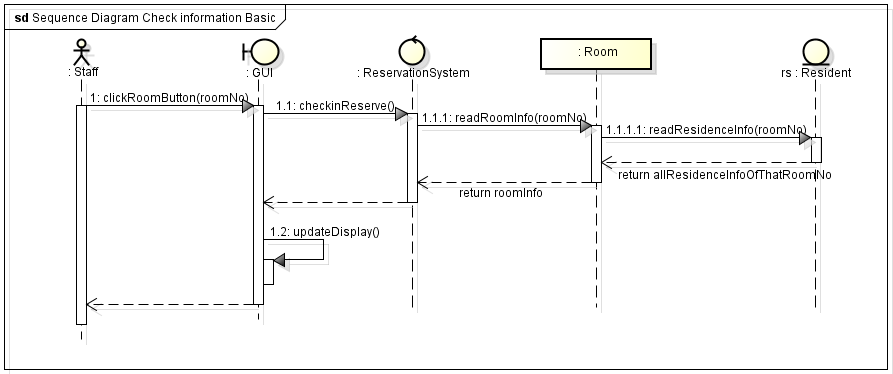
**Billing:** User want to print bill of available room.

1. The system asks the user for room number and water meter, electricity meter for that room.
2. The system shows the message that this room has no occupant at the moment and does not continue to next page.
3. The system does not continue to next page and shows the message that this room has no occupant at the moment.
4. The system asks the user to re-enter the room number.

**Sequence diagram**

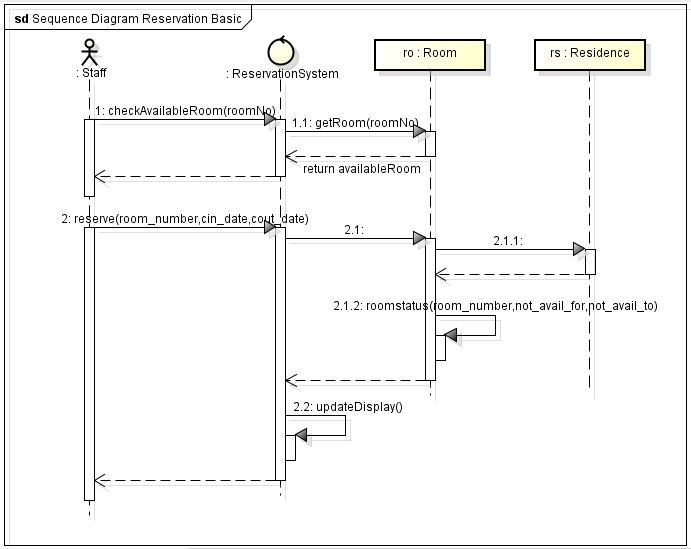
1. **Check information**

**Basic case:** The staff can view all information of the room and resident in the record, the user can only view information of their reservation recorded in the database if they check in.

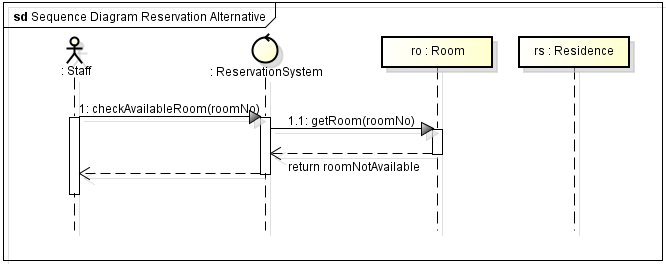
****

1. **Reservation**

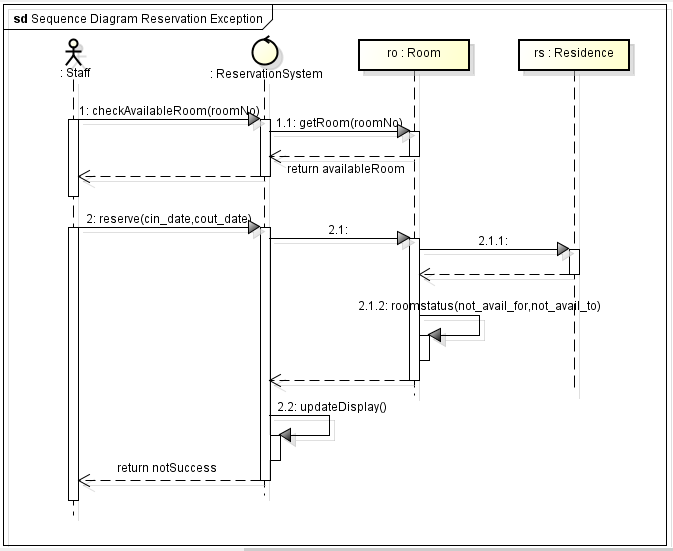
**Basic case:** The system check for availability of the room. If the room is available, staff can make their reservation.

****

**Alternative case:** The system check for availability of the room and there is no available room, then staff cannot make their reservation.

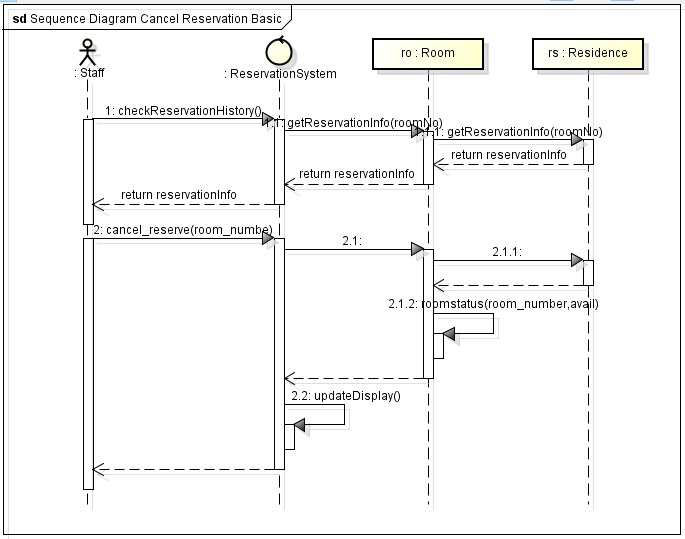
****

**Exceptional case:** The system check for availability of the room. If the room is available, staff can make their reservation. In the fill information field, staff not fill all needed information.

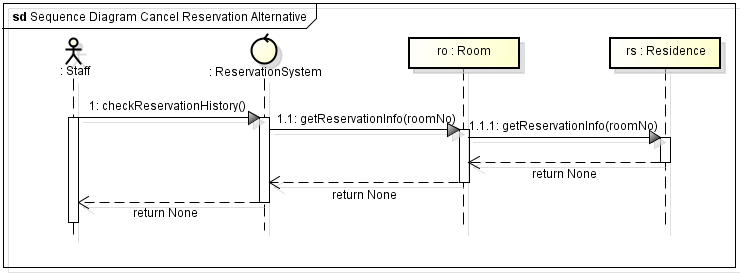
****

1. **Cancel Reservation**

**Basic case:** The system check for not availability of the room. If the room is not available, staff can make their cancel reservation.

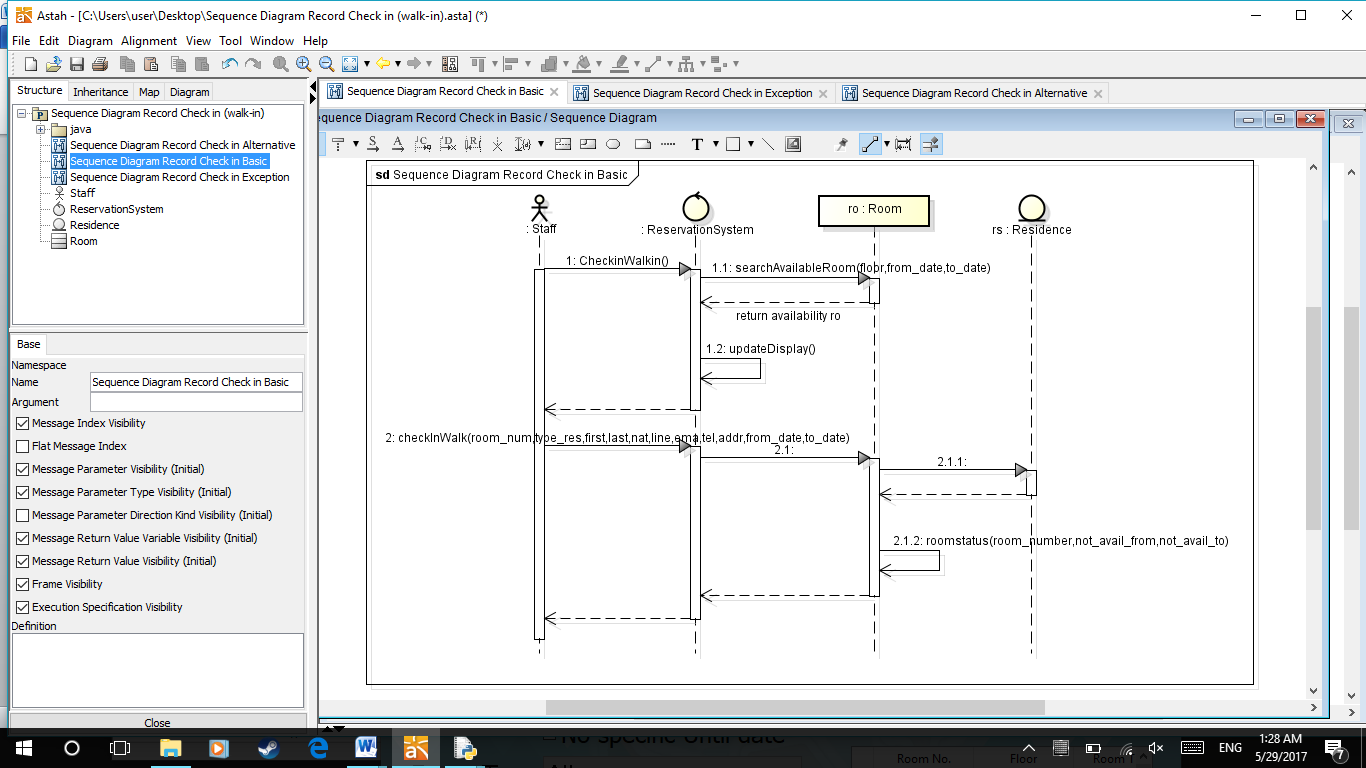


**Alternative case:** No information of revered room then staff cannot make their cancel reservation.

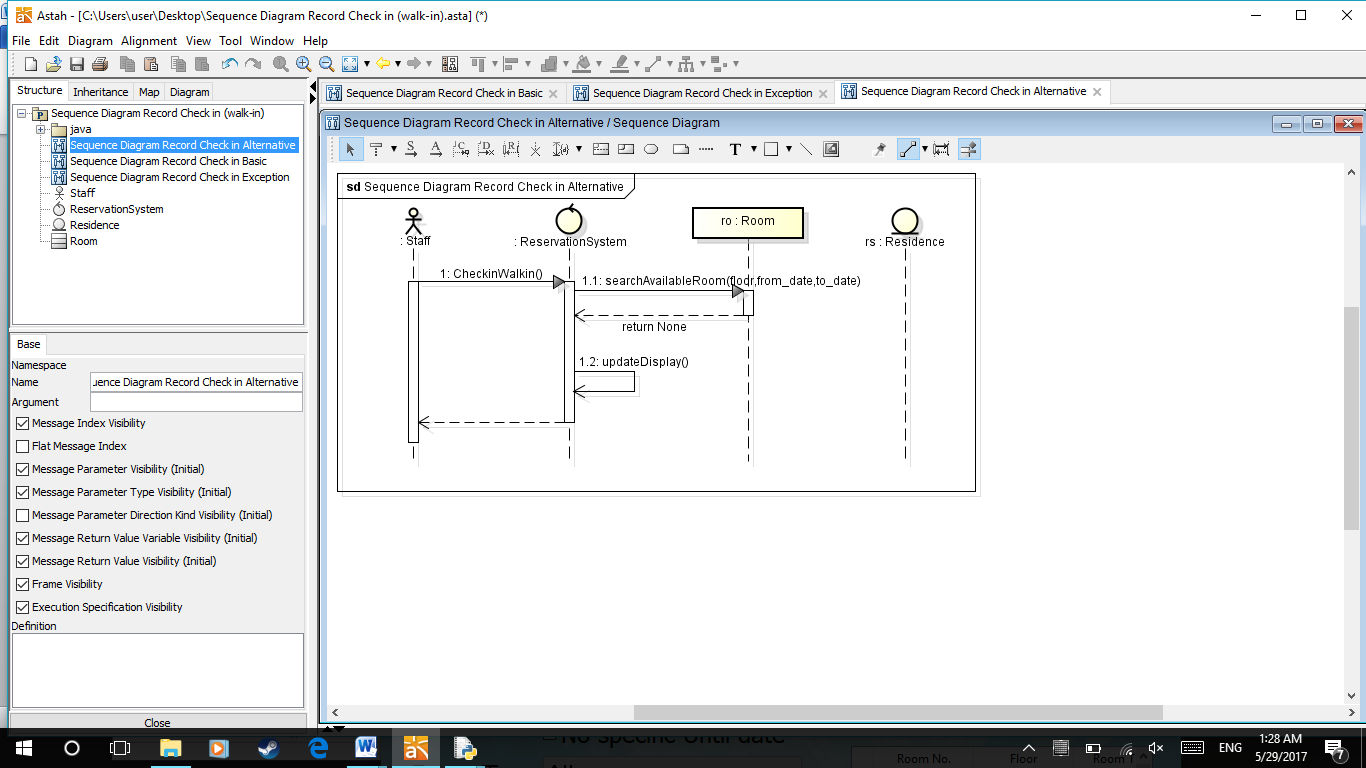
****

1. **Record Check in (walk-in)**

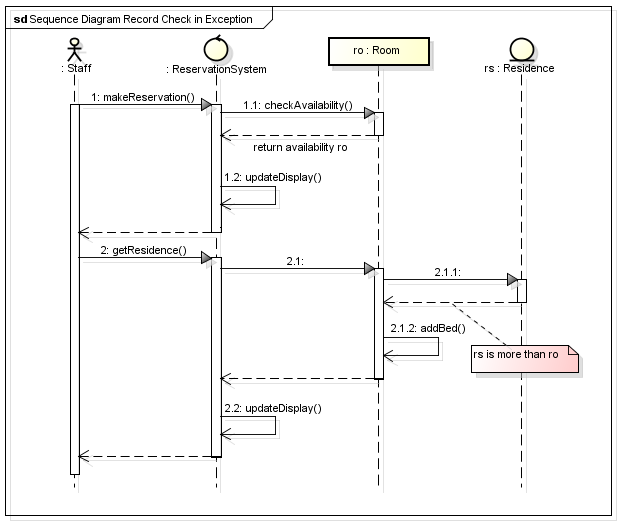
**Basic case:** The system check for availability of the room. There is room available, the system ask the user to choose the room number and get their information. The system add user to the room and mark the room as not available and record the booking into database.

****

**Alternative case:** There is no available room left for new user.

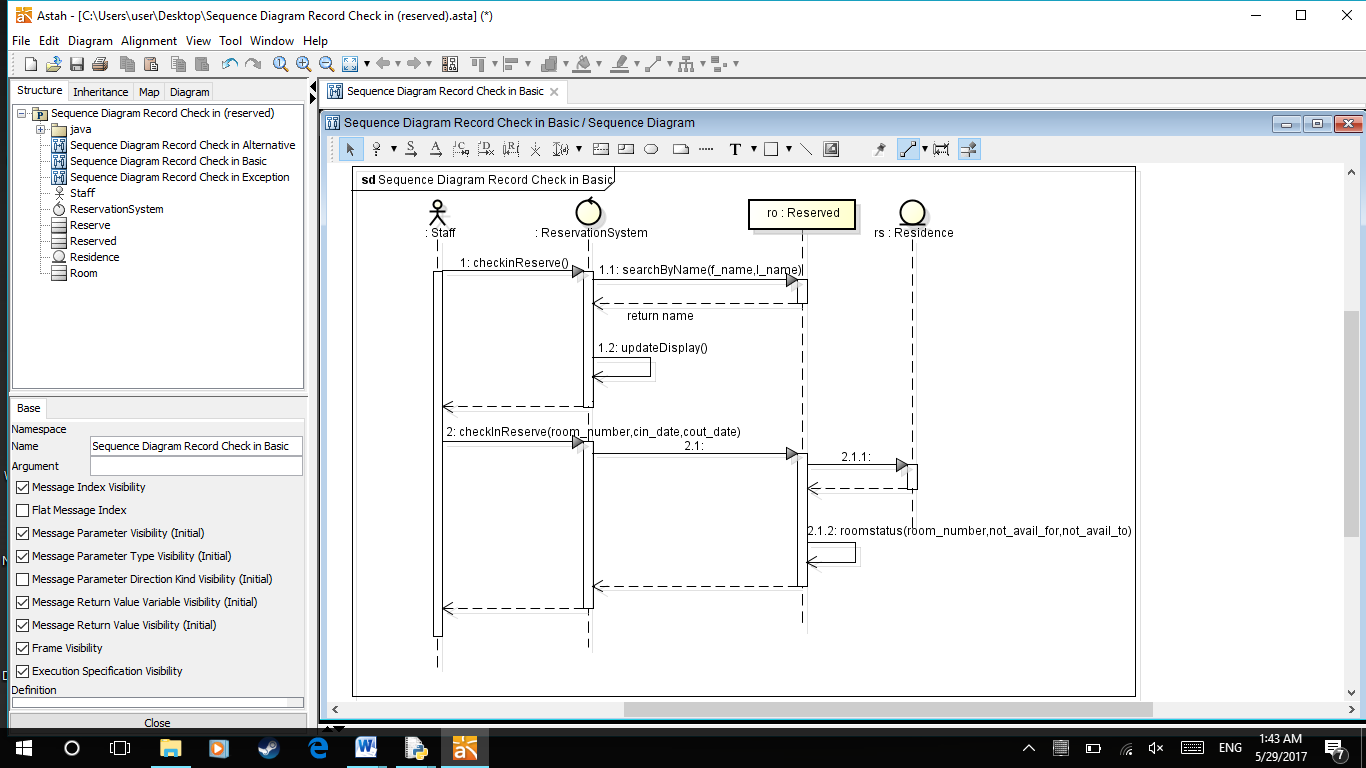


**Exceptional case:** The number of available room is not enough for the new user.

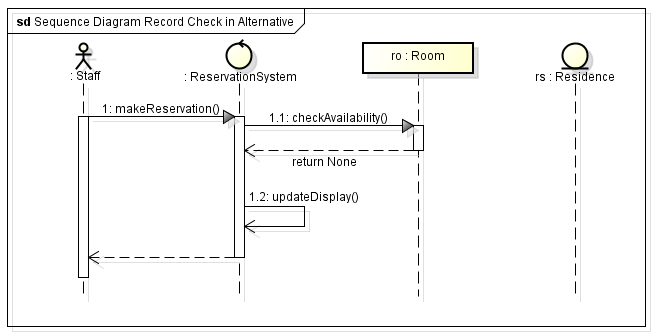


1. **Record Check in (reserved)**

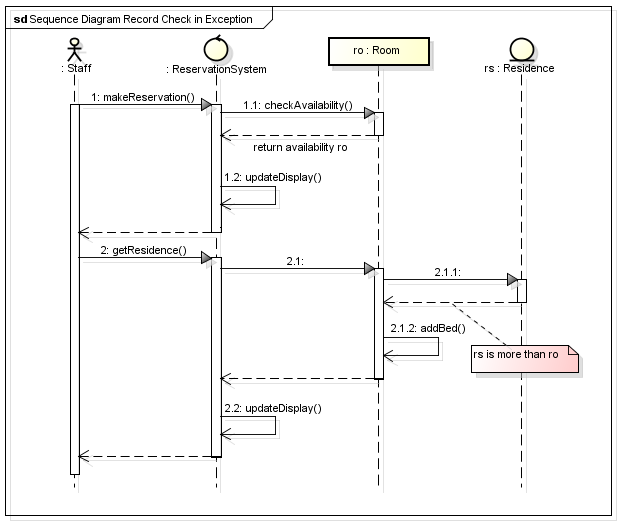
**Basic case:** User confirms to make their reservation. The user searches the reservation by name. The user chooses room number and gets their information. The system add user to the room and mark the room as not available and record the booking into database.

****

**Alternative case:** There is no available room left for new user.

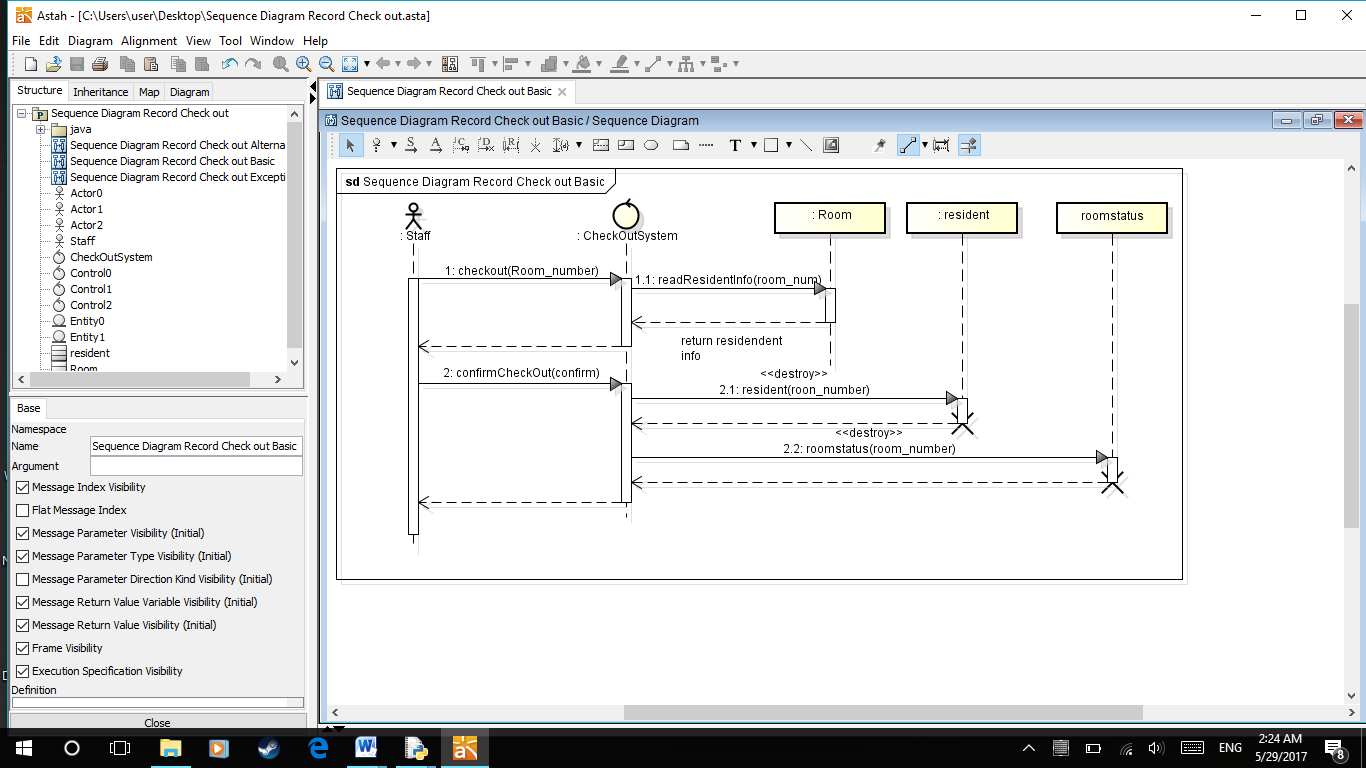


**Exceptional case:** The number of available room is not enough for the new user.

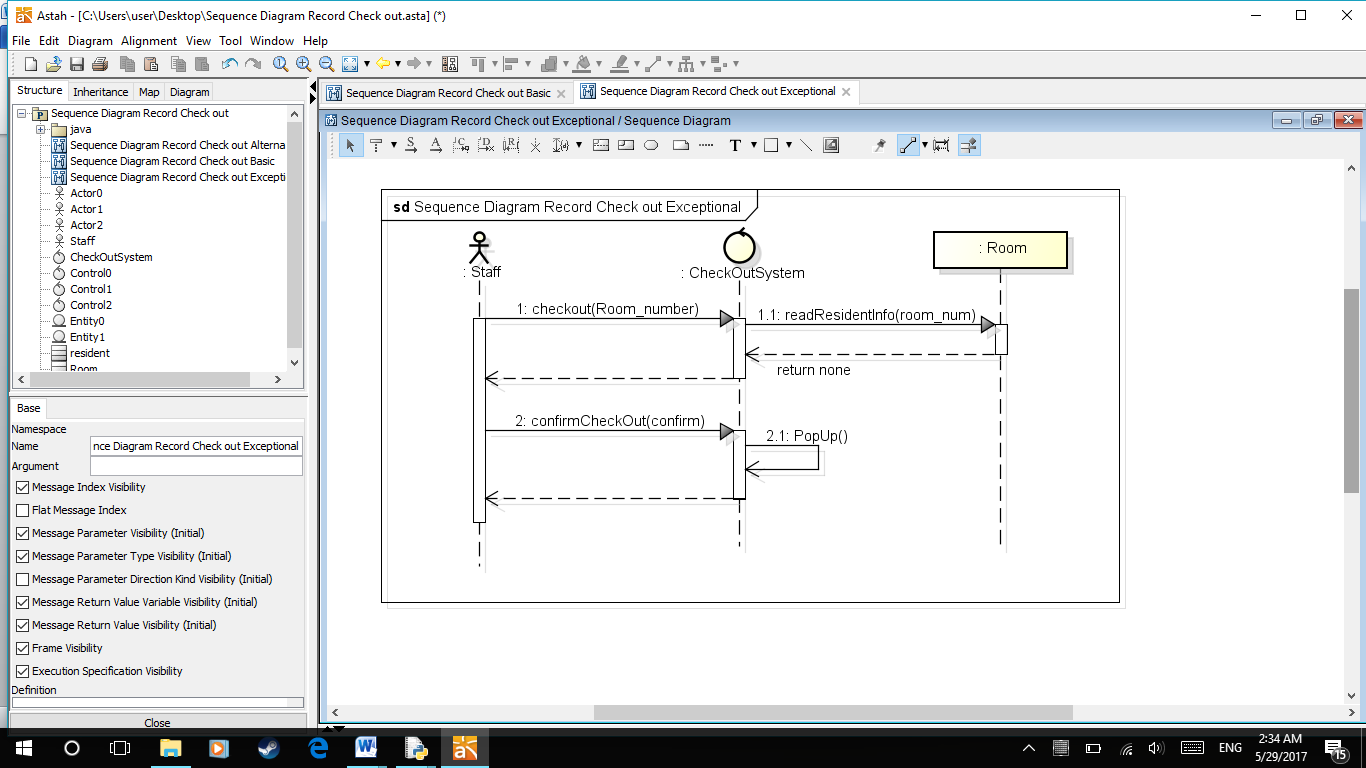


1. **Record Check out**

**Basic case:** User enters room number to retrieve information of residence in that room. The user, then, confirm the check out. After that, the system will change the room to be available again.

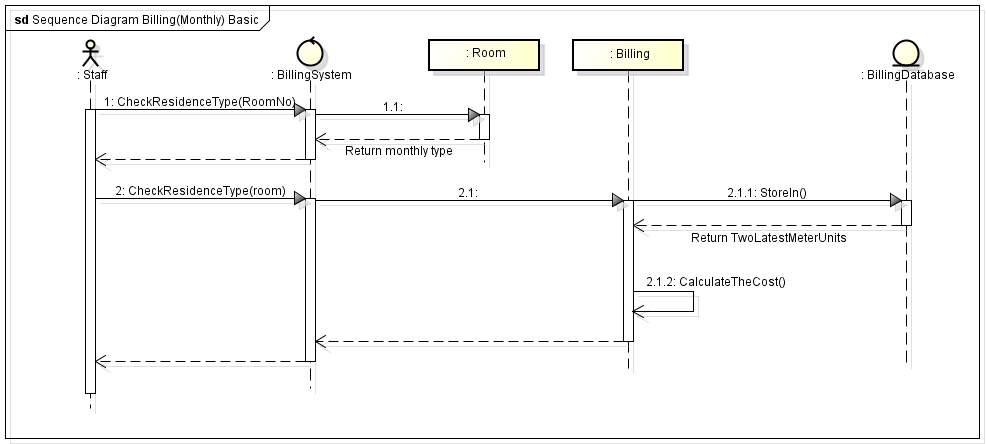
****

**Exceptional case:** If the room is vacant, the system will notify the user that the room is vacant.

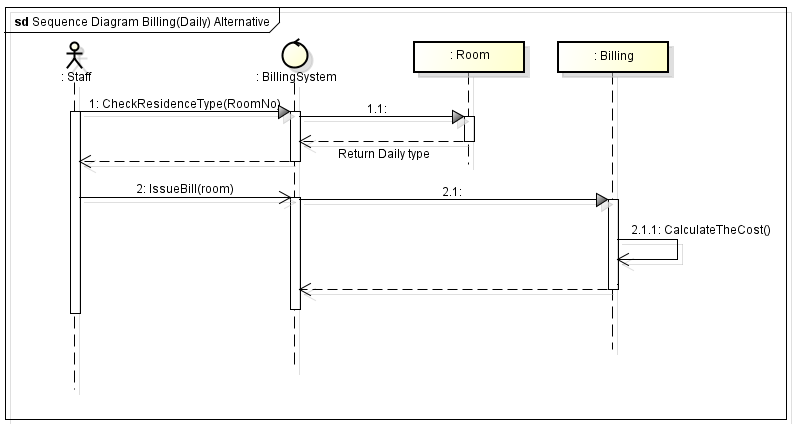


1. **Billing**

**Basic case:** Issuing a bill to a monthly residence.

****

**Alternative case:** Issuing a bill to a daily residence (no need to calculate for utilities).

****

**Exceptional case:** Invalid room number.

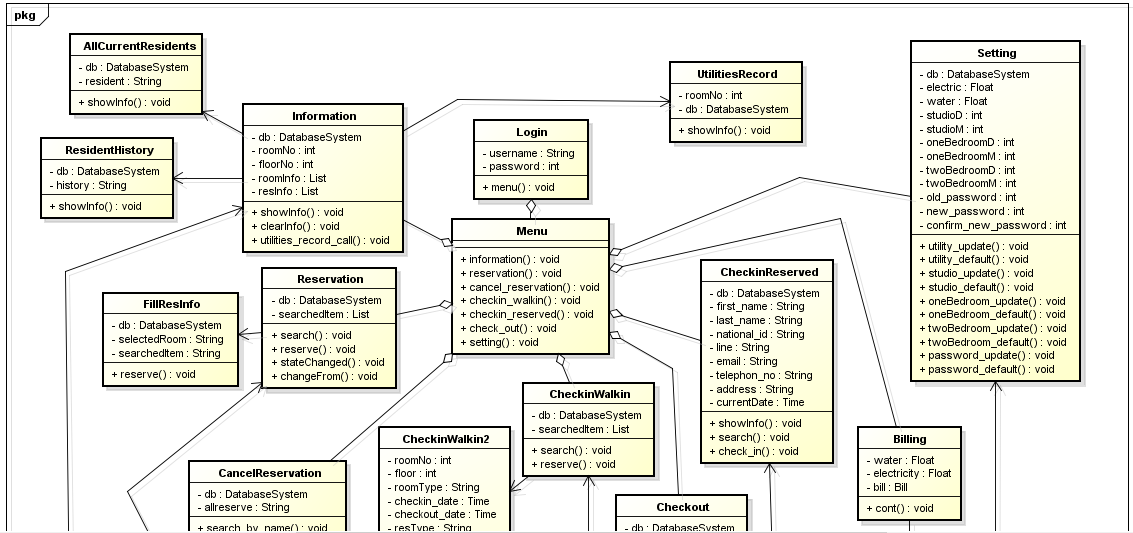
****

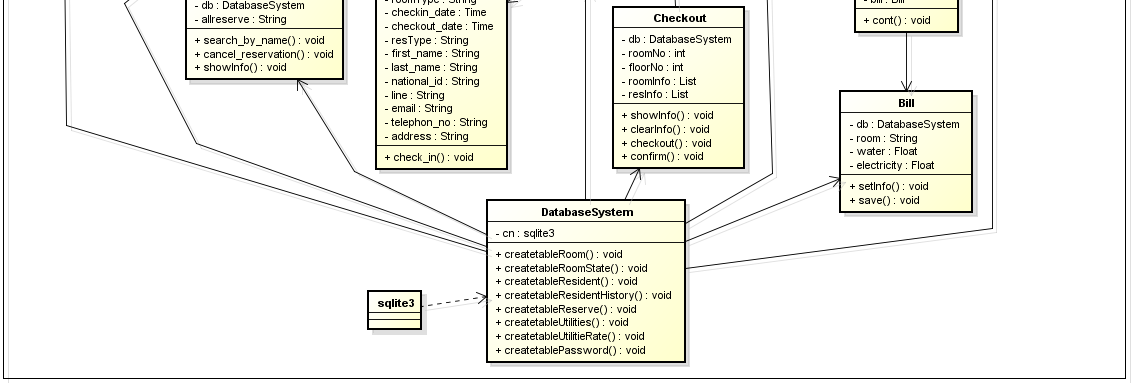
1. **Setting**

**Basic case:** The staff set the utilities rate for electricity or water, renting rate for each room type, or change their password.



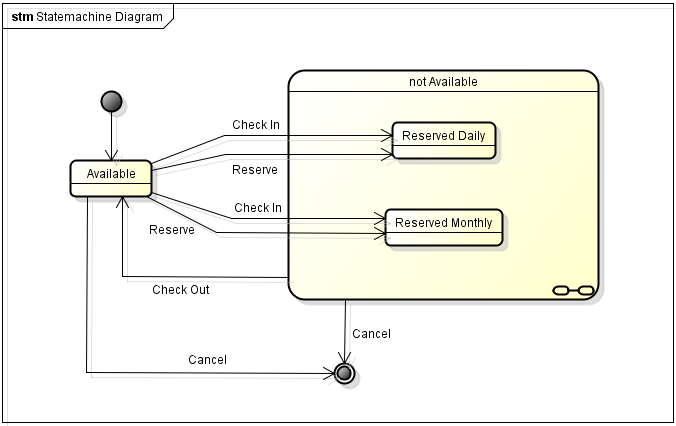
**Class diagram**



****

**State-chart diagram**

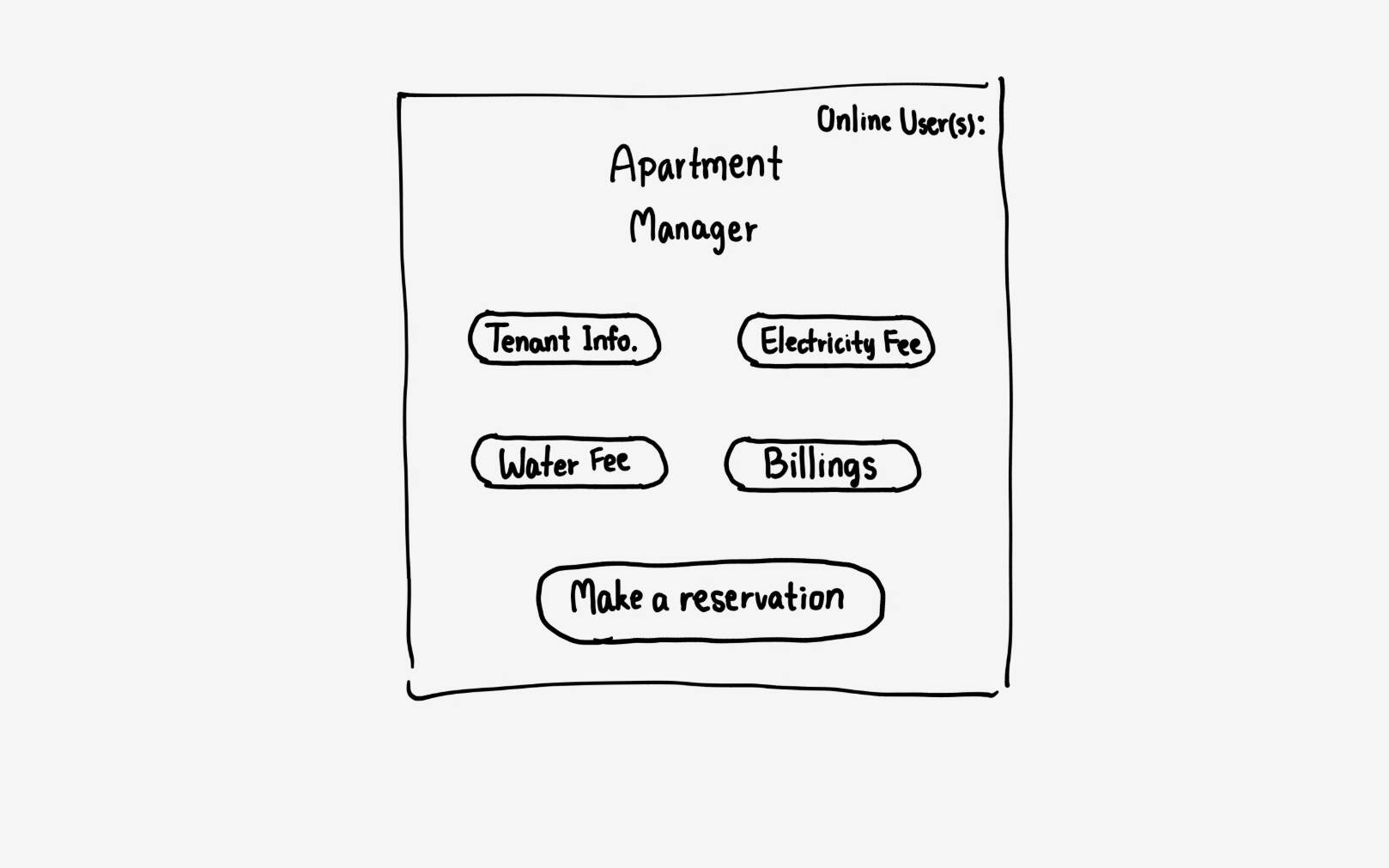
Reserve

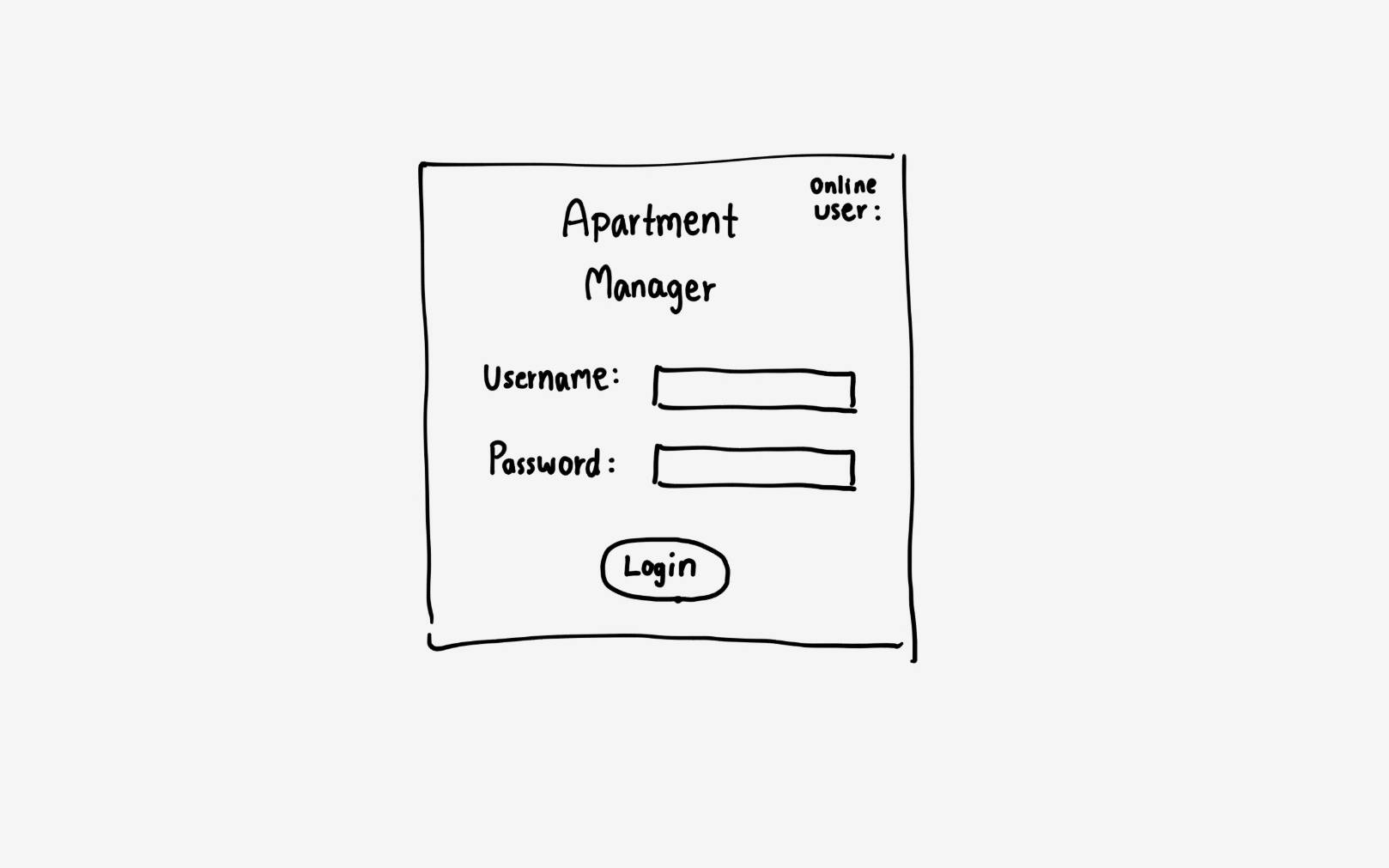


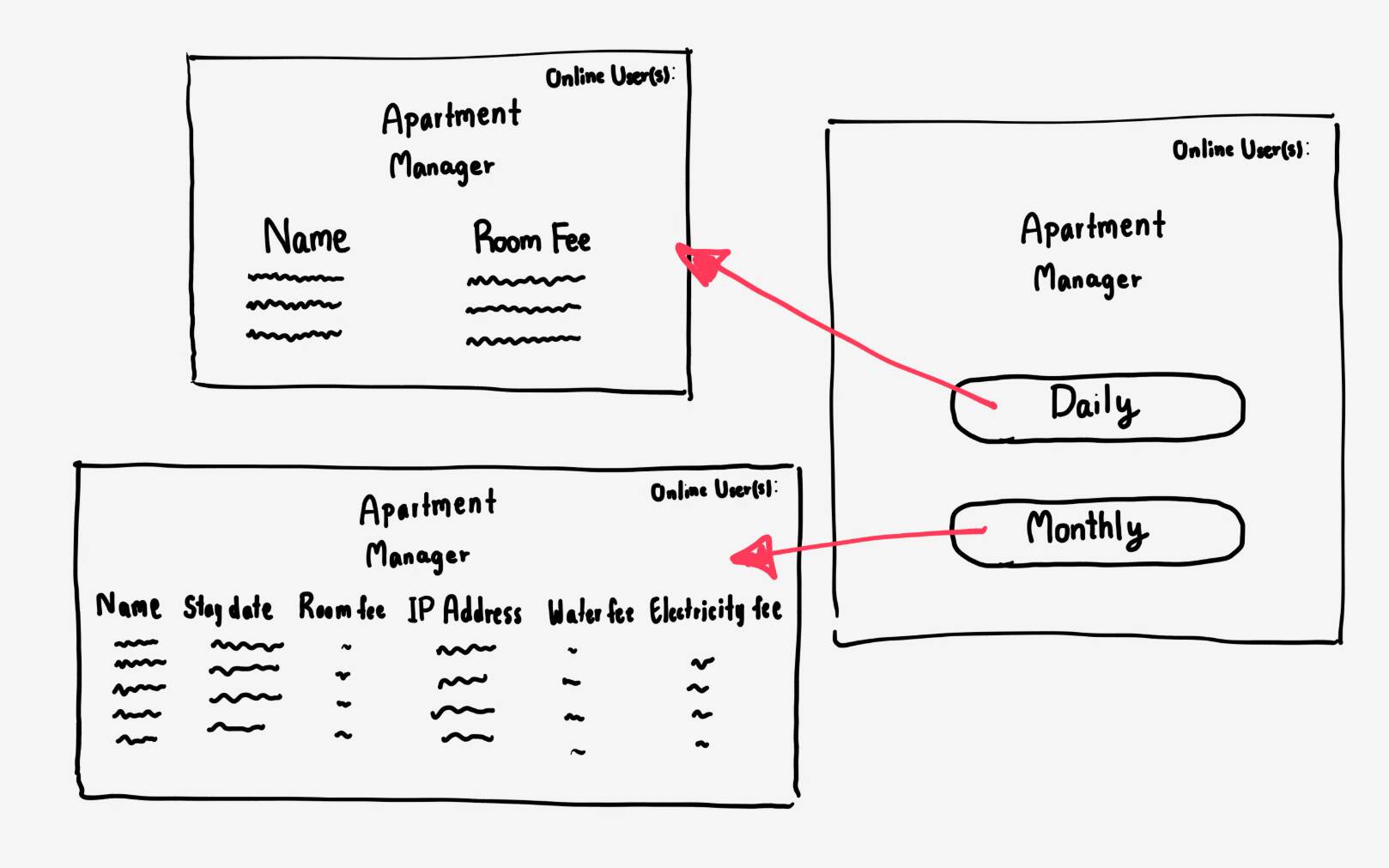


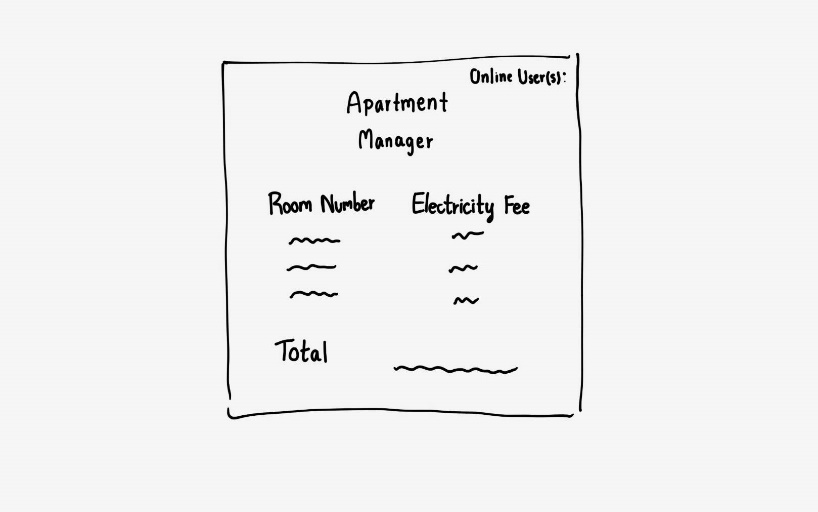
**Explanation:** The diagram shows states of room, Available and Not Available. Initially, the room will be in Available state. The stage of the room will change from Available to Not Available when the user check-In or reserve the room. Once the room is in Not Available state, the room can either be in reserved daily or reserved monthly state.

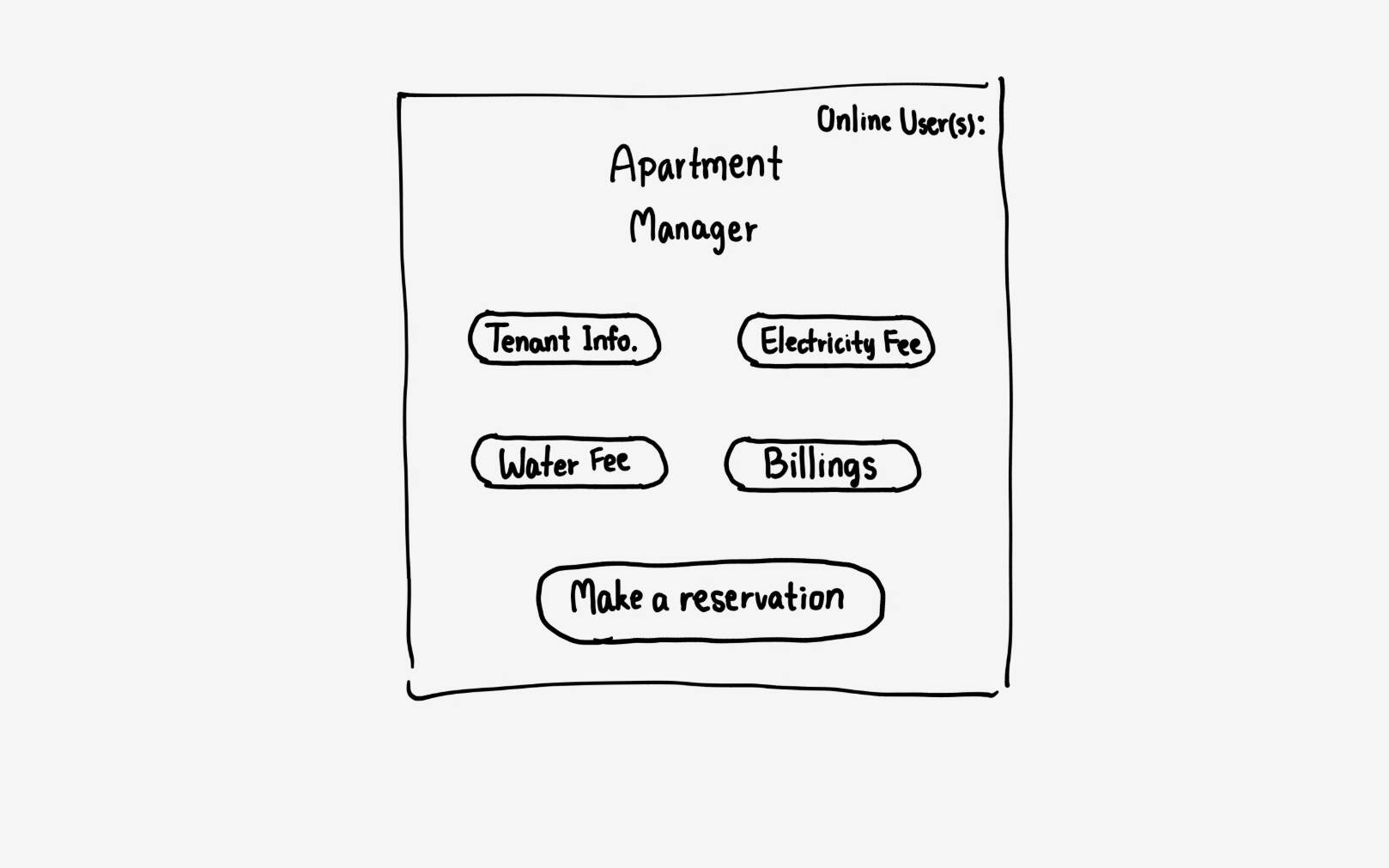
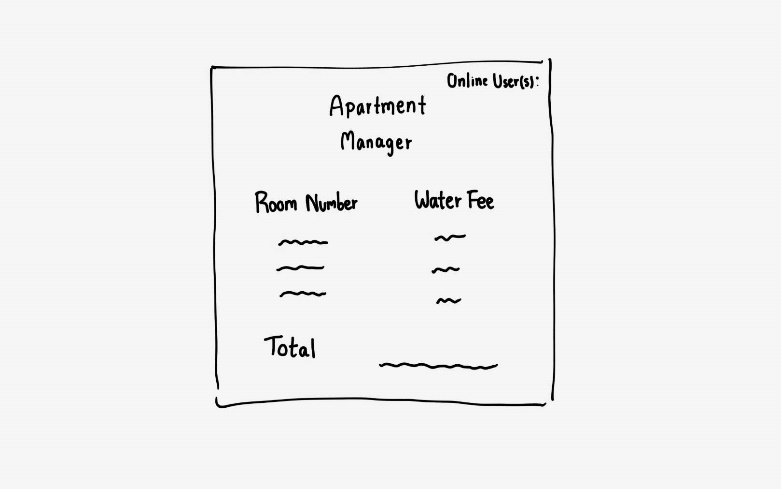
The room will change from Not Available to Available state when the room was checked-out.

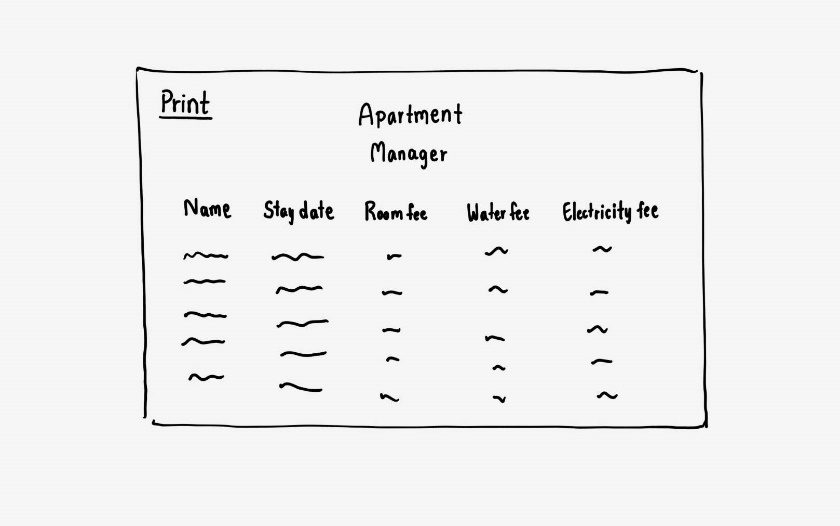
**UI design**

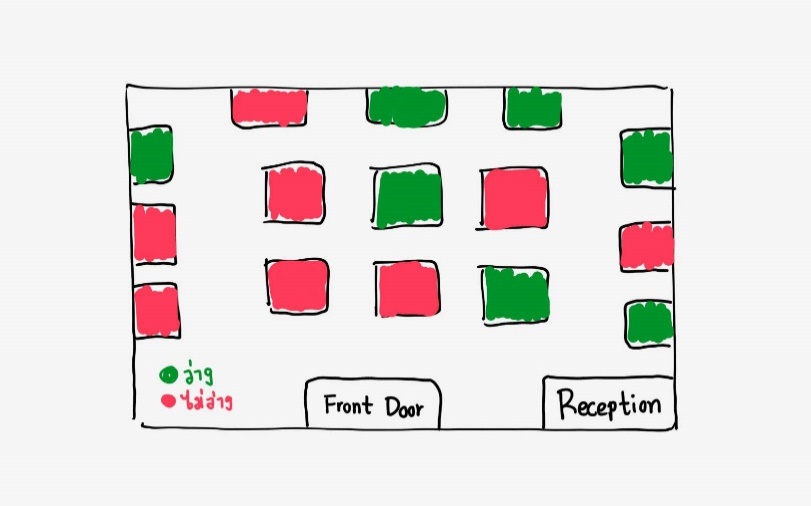
****

****

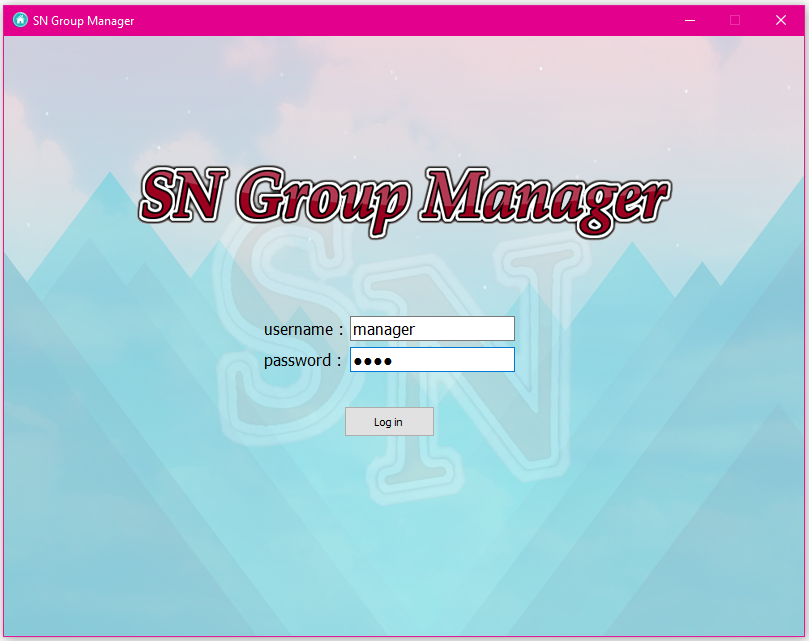
****

****

****

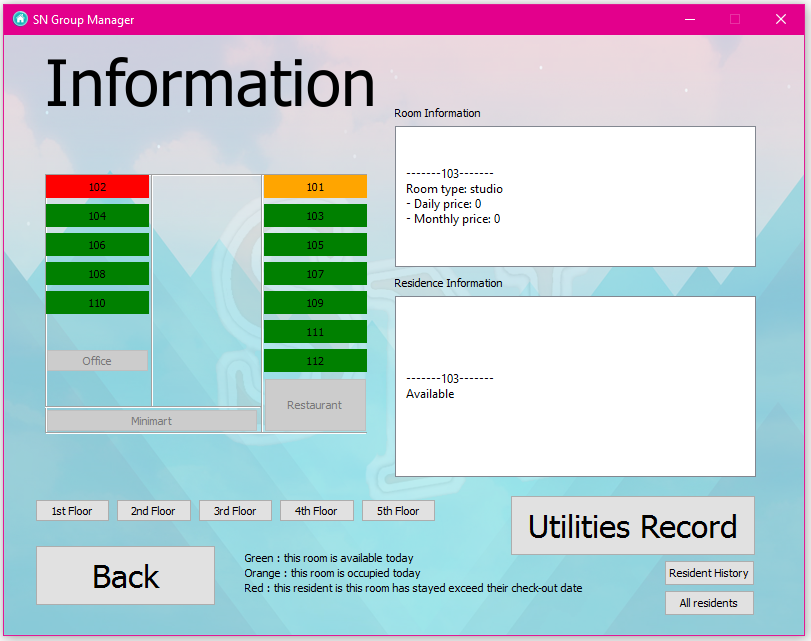
****

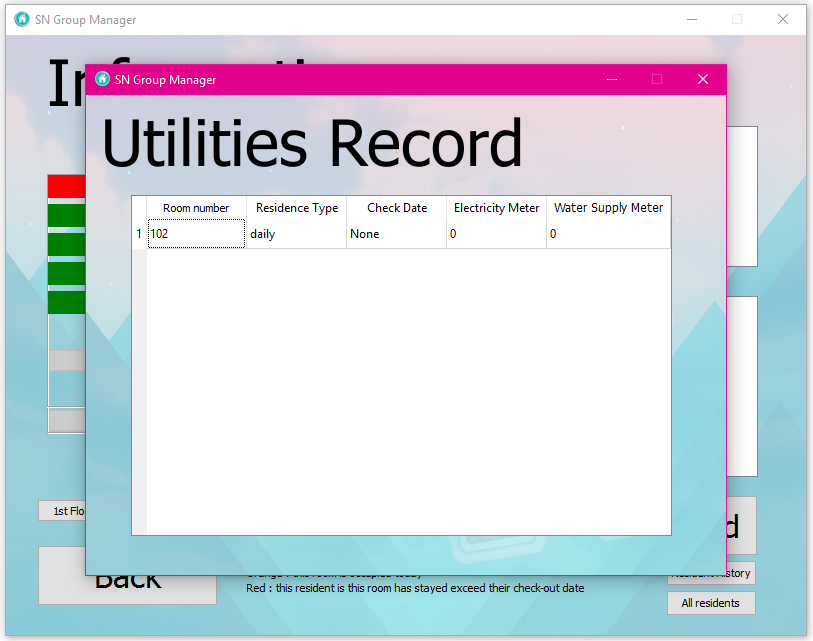
**Screen shots**

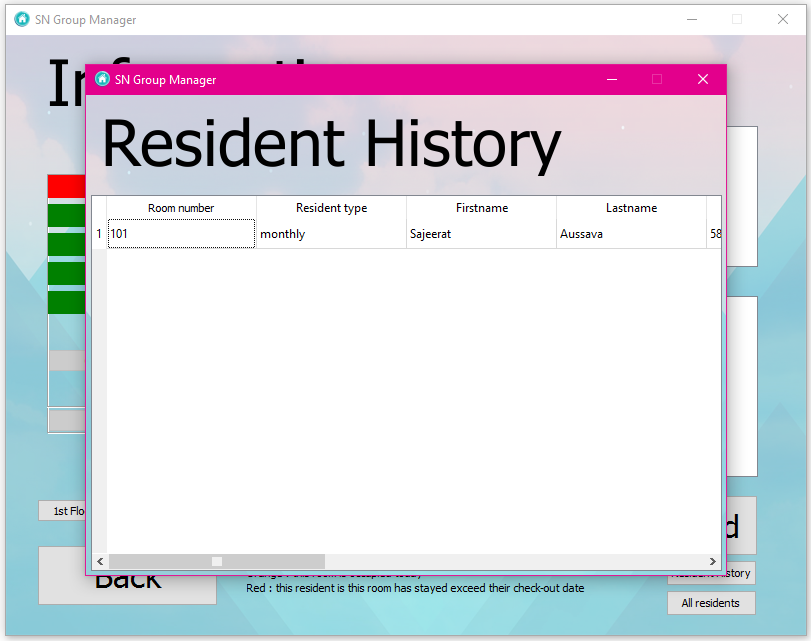
  
Main Login Screen

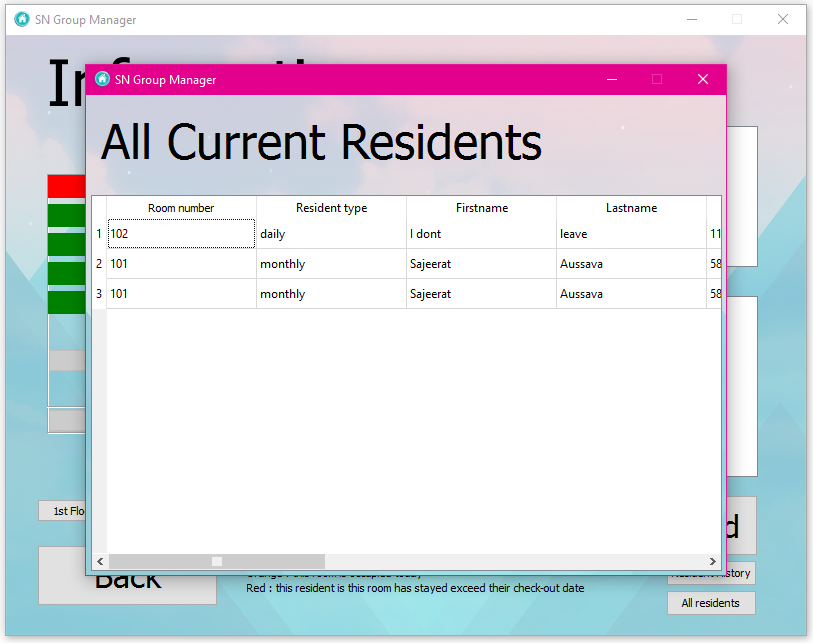
  
Main Manu Interface

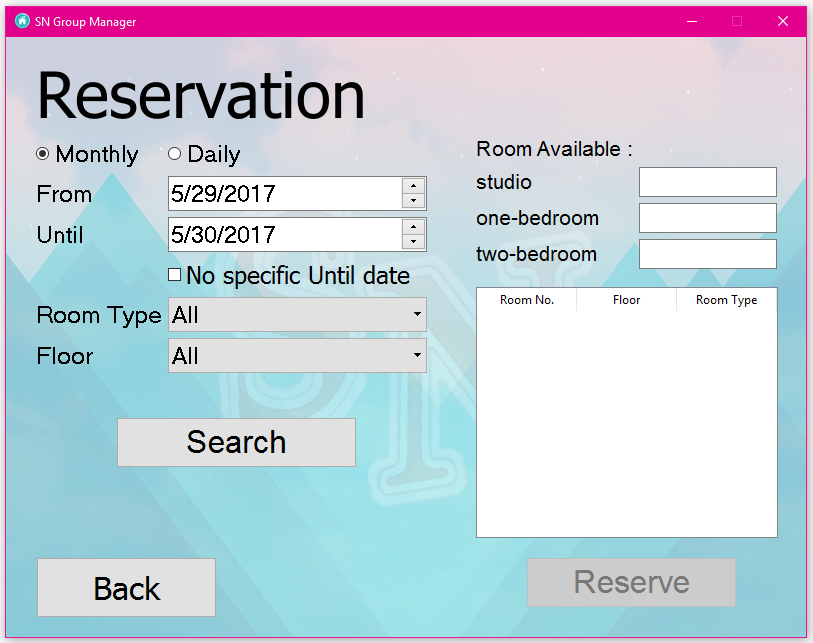
  
Check information Interface

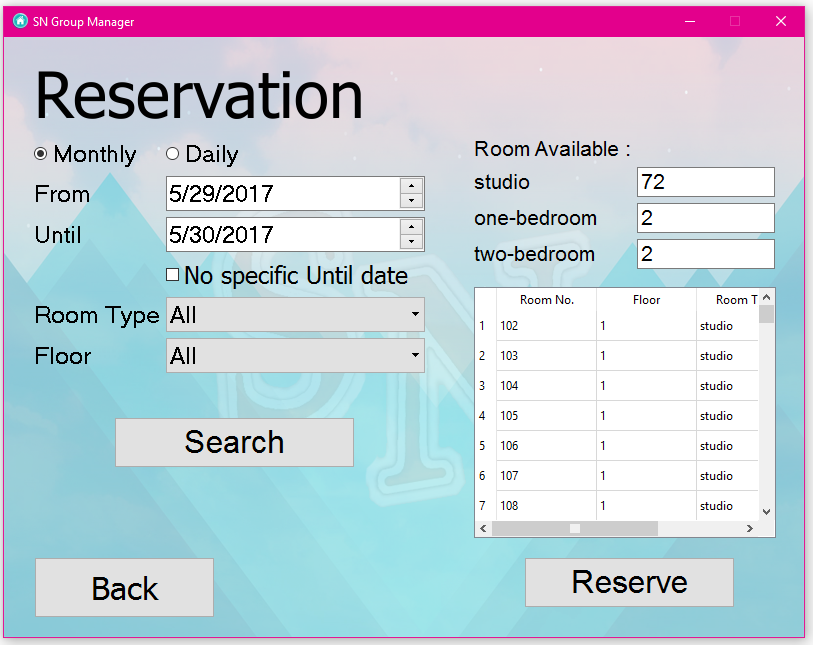


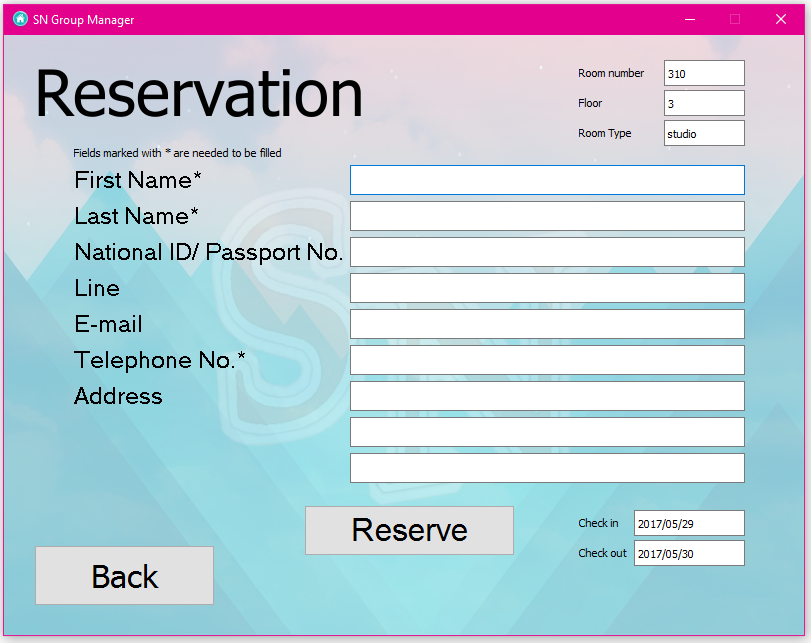


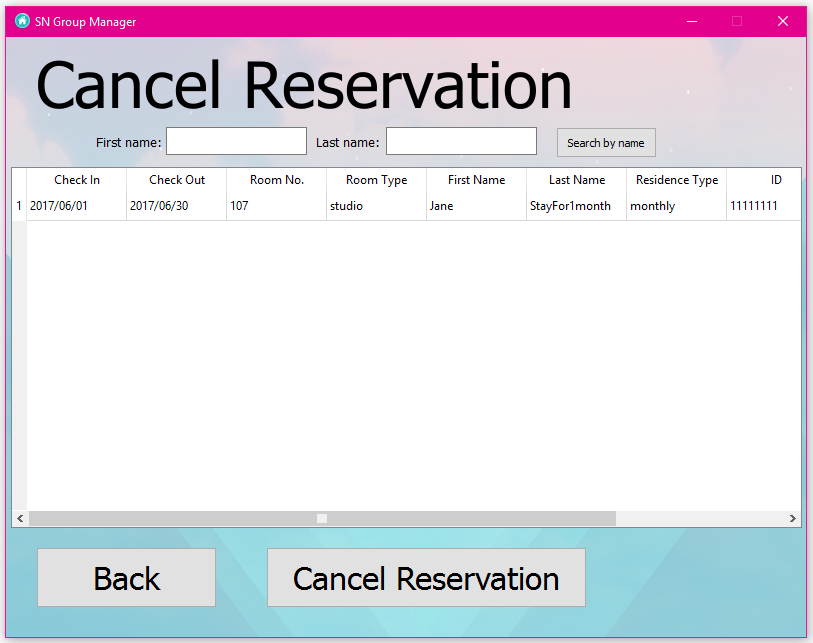


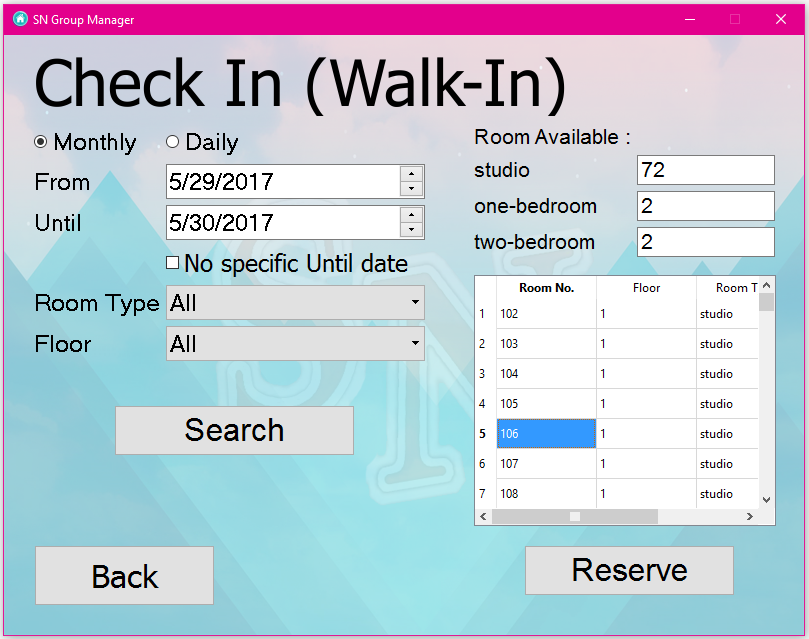


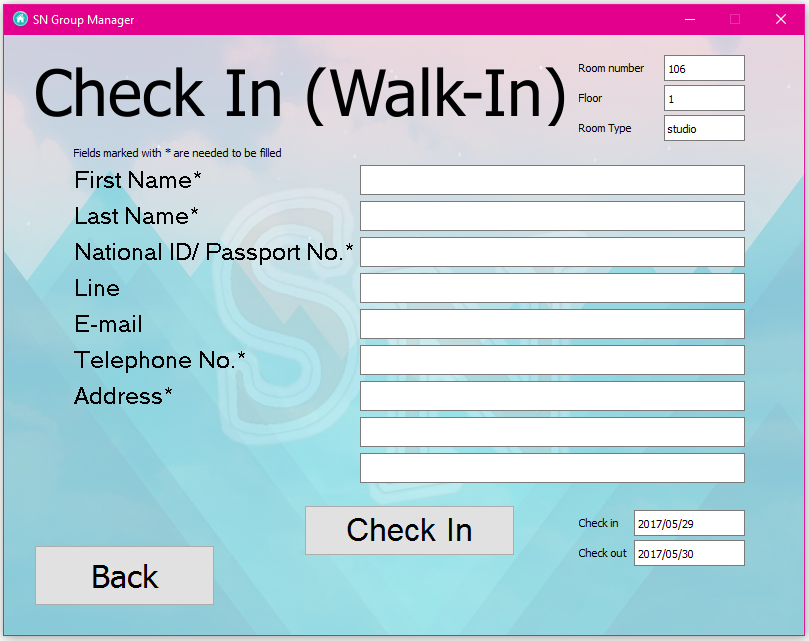
  
Reservation Interface

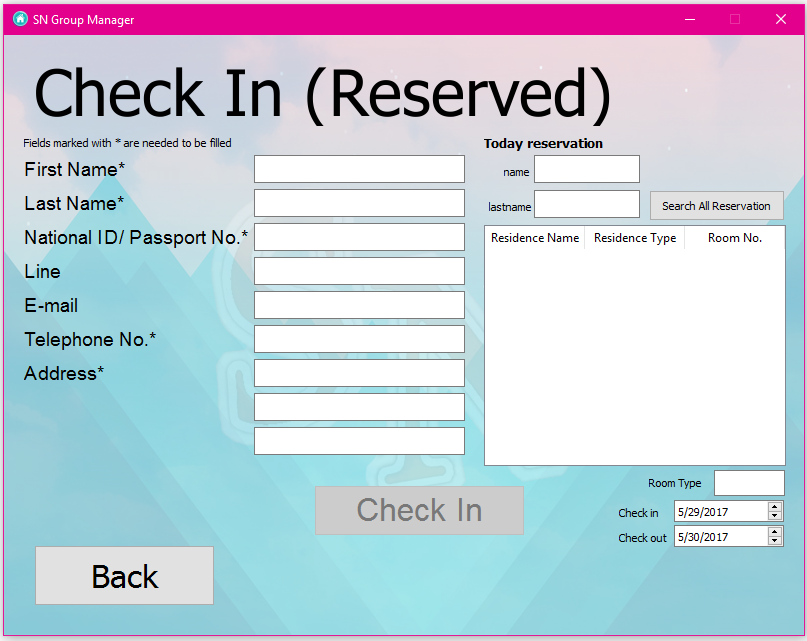


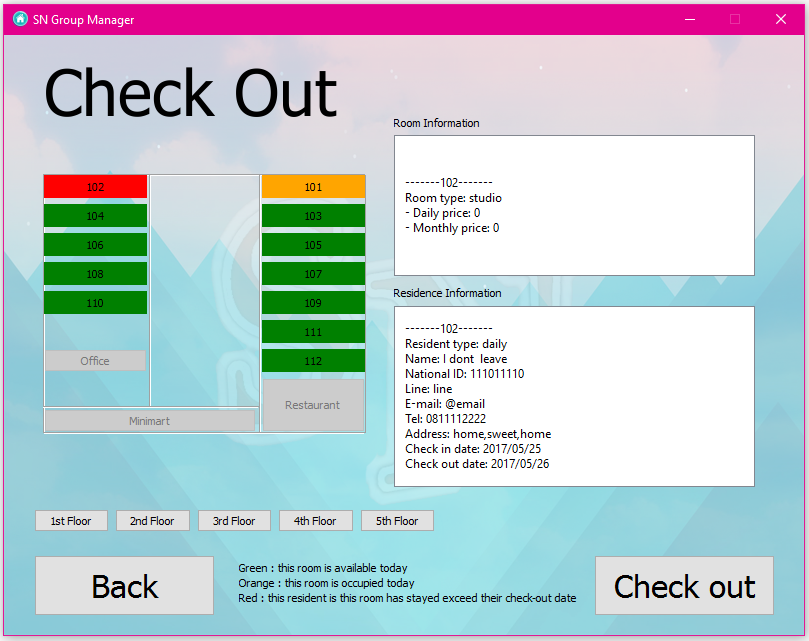


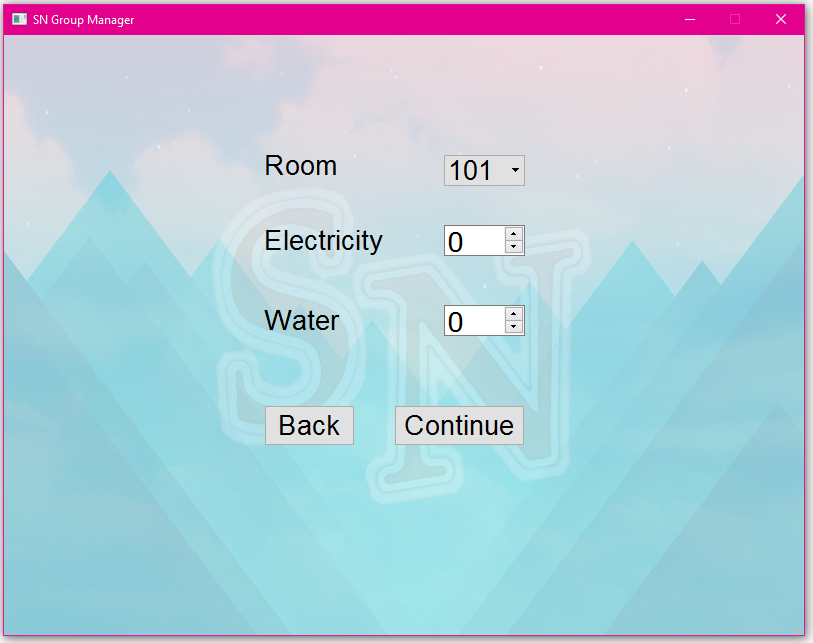
  
Cancel reservation Interface

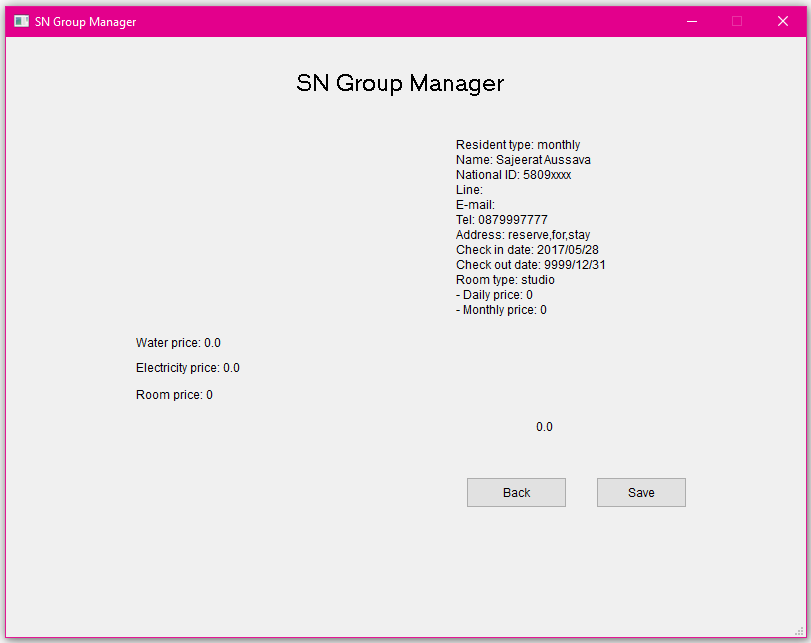
  
Check in (walk-in) Interface

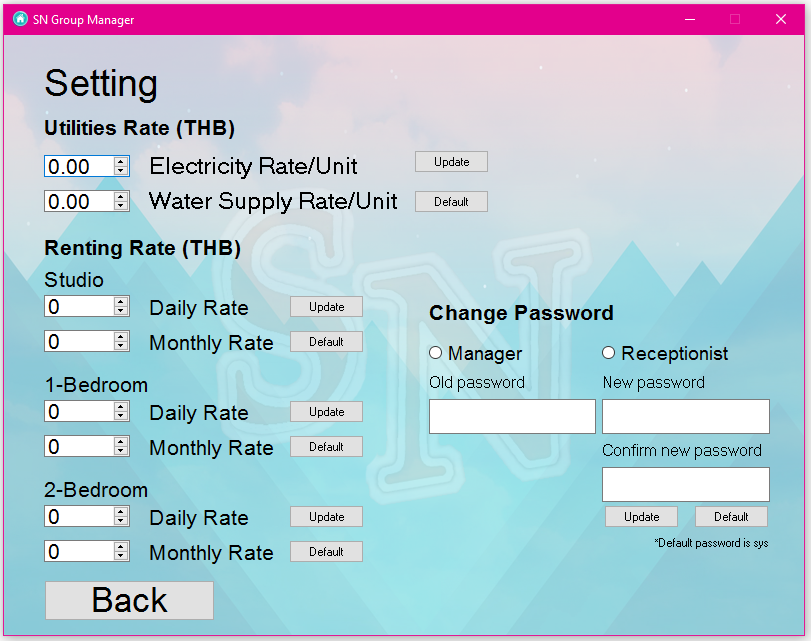


  
Check in (reserved) Interface

  
Check out Interface

  
Billing Interface



  
Setting Interface

**Source code**

DatabaseSystem.py

1. **import** sqlite3
2. **import** sys
3. **import** datetime
5. **class** DatabaseSys():
6. **def** \_\_init\_\_(self):
7. self.cn = sqlite3.connect(".\SNGroupSys.db")
8. self.cur = self.cn.cursor()
9. self.createtableResident()
10. self.createtableUtilities() #NEW
11. self.createtableUtilitieRate()#NEW
12. self.createtableRoom()
13. self.createtableReserve()
14. self.createtableRoomState()
15. self.createtableResidentHistory()
16. self.createtablePassword()#NEW
18. **def** createtableReserve(self):
19. self.cur.execute("""create table if not exists[reserve]([room\_number] varchar(20) ,[type\_of\_res] varchar(30),
20. [first\_name] varchar(20),[last\_name] varchar(20),
21. [national\_id] varchar(30),[line] varchar(30),[email] varchar(30),
22. [tel\_no] varchar(30),[permanent\_address] varchar(100),
23. [check\_in\_date] varchar(20),[check\_out\_date] varchar(20))""")
24. self.cn.commit()
26. **def** createtableRoomState(self):
27. self.cur.execute("""create table if not exists[roomstatus]([room\_number] varchar(20) ,
28. [not\_avail\_from] varchar(20),[not\_avail\_to] varchar(20))""")
29. self.cn.commit()
31. **def** createtableResident(self):
32. self.cur.execute("""create table if not exists[resident]([room\_number] varchar(20) ,[type\_of\_res] varchar(30),
33. [first\_name] varchar(20),[last\_name] varchar(20),
34. [national\_id] varchar(30),[line] varchar(30),[email] varchar(30),
35. [tel\_no] varchar(30),[permanent\_address] varchar(100),
36. [check\_in\_date] varchar(20),[check\_out\_date] varchar(20))""")
38. self.cn.commit()
40. **def** createtableUtilities(self):
41. self.cur.execute("""create table if not exists[utilities\_unit]([room\_number] varchar(20),[month\_year] varchar(20),[water\_meter\_unit] varcar(20),
42. [electricity\_meter\_unit] varcar(20))""")
43. self.cn.commit()
45. **def** createtableUtilitieRate(self):
46. s = """select count(\*) from sqlite\_master where type='table' and name='utilities\_rate'"""
47. table\_exist = self.cur.execute(s).fetchone()
48. **if**  table\_exist[0]:
49. **pass**
50. **else**:
51. self.cur.execute("""create table if not exists[utilities\_rate]([util\_type] varchar(20),[rate] decimal,[default\_val] decimal)""")
52. s = "insert into utilities\_rate values(?,?,?)"
53. self.cur.execute(s,("water",0.0,0.0))
54. self.cur.execute(s,("electricity",0.0,0.0))
55. self.cn.commit()
57. **def** createtableRoom(self):
58. s = """select count(\*) from sqlite\_master where type='table' and name='room'"""
59. table\_exist = self.cur.execute(s).fetchone()
60. **if**  table\_exist[0]:
61. **pass**
62. **else**:
63. self.cur.execute("""create table if not exists[room]([room\_number] varchar(20),[floor] varchar(30),[room\_type] varchar(30),[daily\_price] varchar(20),
64. [monthly\_price] varchar(20),[default\_d] varchar(20),[default\_m] varchar(20))""")
65. i = 101
66. floor = 1
67. type\_room = "studio"
69. **while**(True):
70. s = "insert into room values(?,?,?,?,?,?,?)"
71. self.cur.execute(s,(i,floor,type\_room,'0','0','0','0'))
72. i = i+1
73. **if** i == 113:
74. i = 201
75. floor = 2
76. **if** i == 218:
77. i = 301
78. floor = 3
79. **if** i == 318:
80. i = 401
81. floor = 4
82. **if** i == 418:
83. i = 501
84. floor = 5
85. **if** i == 514 **or** i == 515:
86. type\_room = "1-bedroom"
87. **if** i == 512 **or** i == 513:
88. type\_room = "2-bedroom"
89. **if** i == 516:
90. **break**
91. self.cn.commit()
93. **def** createtableResidentHistory(self):
94. self.cur.execute("""create table if not exists[residenthistory]([room\_number] varchar(20) ,[type\_of\_res] varchar(30),
95. [first\_name] varchar(20),[last\_name] varchar(20),
96. [national\_id] varchar(30),[line] varchar(30),[email] varchar(30),
97. [tel\_no] varchar(30),[permanent\_address] varchar(100),
98. [check\_in\_date] varchar(20),[check\_out\_date] varchar(20))""")
99. self.cn.commit()
101. **def** createtablePassword(self):
102. s = """select count(\*) from sqlite\_master where type='table' and name='password'"""
103. table\_exist = self.cur.execute(s).fetchone()
104. **if**  table\_exist[0]:
105. **pass**
106. **else**:
107. self.cur.execute("""create table if not exists[password]([manager\_password] varchar(20),
108. [receptionist\_password] varchar(20))""")
110. s = "insert into password values (?,?)"
111. self.cur.execute(s,("sys","sys"))
112. self.cn.commit()
114. **def** readRoomInfo(self,room\_num):
115. out = ""
116. self.cur.execute("select room\_type,daily\_price,monthly\_price from room where room\_number = ?",(room\_num,))
117. temp = self.cur.fetchall()
118. out = "Room type: "+ temp[0][0]+"\n- Daily price: "+ temp[0][1] + "\n- Monthly price: "+ temp[0][2]
119. **return** out

122. **def** readResidentInfo(self,room\_num):
123. out = ""
124. self.cur.execute("select \* from resident where room\_number = ?",(room\_num,))
125. temp = self.cur.fetchall()
126. **if** (len(temp) == 0):
127. out = "Available"
128. **else**:
129. out = "Resident type: "+temp[0][1]+ "\nName: " + temp[0][2]+ " " + temp[0][3] + "\nNational ID: " +\
130. temp[0][4] +"\nLine: "+ temp[0][5] +"\nE-mail: "+ temp[0][6] + "\nTel: " +\
131. temp[0][7] + "\nAddress: " +temp[0][8]+\
132. "\nCheck in date: " + temp[0][9] +"\nCheck out date: " + temp[0][10]
134. **return** out
136. **def** readResidentType(self,room\_num):
137. temp = ""
138. self.cur.execute("select type\_of\_res from resident where room\_number = ?",(room\_num,))
139. out = self.cur.fetchall()
140. **if** (len(out) == 0):
141. temp = "None"
142. **else**:
143. temp = out[0][0]
144. **return** temp
146. **def** readRoomType(self,room\_num):
147. self.cur.execute("select room\_type from room where room\_number = ?",(room\_num,))
148. out = self.cur.fetchone()
149. **return** out[0]
150. #and room\_number not in (select room\_number from resident)
152. **def** searchAllAvailableRoom(self,floor,from\_date,to\_date):
153. from\_date = from\_date.split("/")
154. from\_date = datetime.datetime(int(from\_date[0]),int(from\_date[1]),int(from\_date[2])).date()
156. to\_date = to\_date.split("/")
157. to\_date = datetime.datetime(int(to\_date[0]),int(to\_date[1]),int(to\_date[2])).date()
159. self.cur.execute("select \* from roomstatus")
160. allStatus = self.cur.fetchall()
161. roomstatus\_list = []
162. **for** i **in** range (len(allStatus)):
163. eachRecord = allStatus[i]
164. roomNo = eachRecord[0]
165. indate = eachRecord[1]
166. outdate = eachRecord[2]
168. indate = indate.split("/")
169. indate = datetime.datetime(int(indate[0]),int(indate[1]),int(indate[2])).date()
170. outdate = outdate.split("/")
171. outdate = datetime.datetime(int(outdate[0]),int(outdate[1]),int(outdate[2])).date()
172. roomstatus\_list.append([roomNo,indate,outdate])
173. not\_available\_list = []
175. **for** i **in** roomstatus\_list:
176. **if** from\_date == i[1] **or** to\_date==i[2] :
177. not\_available\_list.append(i[0])
178. **elif** (from\_date >= i[1] **and** from\_date<=i[2]): #indate <= from\_date <= outdate
179. not\_available\_list.append(i[0])
180. **elif** (to\_date >= i[1] **and** to\_date <=i[2]): #indate <= to\_date <= outdate
181. not\_available\_list.append(i[0])
182. **elif** (from\_date <= i[1] **and** to\_date>=i[2]): #from\_date <= indate AND to\_date >= out\_date
183. not\_available\_list.append(i[0])
185. s = "select room\_number,floor,room\_type from room where floor = ? "
186. self.cur.execute(s,(str(floor)))
187. allRoom = self.cur.fetchall()
188. **for** i **in** allRoom:
189. **if** i[0] **in** not\_available\_list:
190. allRoom.remove(i)
192. **return** allRoom
194. **def** searchAvailableRoom(self,floor,room\_type,from\_date,to\_date):
195. from\_date = from\_date.split("/")
196. from\_date = datetime.datetime(int(from\_date[0]),int(from\_date[1]),int(from\_date[2])).date()
198. to\_date = to\_date.split("/")
199. to\_date = datetime.datetime(int(to\_date[0]),int(to\_date[1]),int(to\_date[2])).date()
201. self.cur.execute("select \* from roomstatus")
202. allStatus = self.cur.fetchall()
203. roomstatus\_list = []
204. **for** i **in** range (len(allStatus)):
205. eachRecord = allStatus[i]
206. roomNo = eachRecord[0]
207. indate = eachRecord[1]
208. outdate = eachRecord[2]
210. indate = indate.split("/")
211. indate = datetime.datetime(int(indate[0]),int(indate[1]),int(indate[2])).date()
212. outdate = outdate.split("/")
213. outdate = datetime.datetime(int(outdate[0]),int(outdate[1]),int(outdate[2])).date()
214. roomstatus\_list.append([roomNo,indate,outdate])
215. not\_available\_list = []
217. **for** i **in** roomstatus\_list :
218. **if** from\_date == i[1] **or** to\_date==i[2] :
219. not\_available\_list.append(i[0])
220. **elif** (from\_date >= i[1] **and** from\_date<=i[2]): #indate <= from\_date <= outdate
221. not\_available\_list.append(i[0])
222. **elif** (to\_date >= i[1] **and** to\_date <=i[2]): #indate <= to\_date <= outdate
223. not\_available\_list.append(i[0])
224. **elif** (from\_date <= i[1] **and** to\_date>=i[2]): #from\_date <= indate AND to\_date >= out\_date
225. not\_available\_list.append(i[0])
227. s = "select room\_number,floor,room\_type from room where floor = ? "
228. self.cur.execute(s,(str(floor)))
229. allRoom = self.cur.fetchall()
230. **for** i **in** allRoom:
231. **if** i[0] **in** not\_available\_list:
232. allRoom.remove(i)
234. **return** allRoom

237. **def** countAvailableRoom(self,rtype,from\_date,to\_date):
238. from\_date = from\_date.split("/")
239. from\_date = datetime.datetime(int(from\_date[0]),int(from\_date[1]),int(from\_date[2])).date()
241. to\_date = to\_date.split("/")
242. to\_date = datetime.datetime(int(to\_date[0]),int(to\_date[1]),int(to\_date[2])).date()
244. self.cur.execute("select \* from roomstatus")
245. allStatus = self.cur.fetchall()
246. roomstatus\_list = []
247. **for** i **in** range (len(allStatus)):
248. eachRecord = allStatus[i]
249. roomNo = eachRecord[0]
250. indate = eachRecord[1]
251. outdate = eachRecord[2]
253. indate = indate.split("/")
254. indate = datetime.datetime(int(indate[0]),int(indate[1]),int(indate[2])).date()
255. outdate = outdate.split("/")
256. outdate = datetime.datetime(int(outdate[0]),int(outdate[1]),int(outdate[2])).date()
257. roomstatus\_list.append([roomNo,indate,outdate])
258. not\_available\_list = []
260. **for** i **in** roomstatus\_list:
261. **if** from\_date == i[1] **or** to\_date==i[2] :
262. not\_available\_list.append(i[0])
263. **elif** (from\_date >= i[1] **and** from\_date<=i[2]): #indate <= from\_date <= outdate
264. not\_available\_list.append(i[0])
265. **elif** (to\_date >= i[1] **and** to\_date <=i[2]): #indate <= to\_date <= outdate
266. not\_available\_list.append(i[0])
267. **elif** (from\_date <= i[1] **and** to\_date>=i[2]): #from\_date <= indate AND to\_date >= out\_date
268. not\_available\_list.append(i[0])
270. self.cur.execute("select count(\*) from room where room\_type = ? ",(rtype,))
271. numRoomType = self.cur.fetchall()
272. numRoomType = int(numRoomType[0][0])
273. count=0
275. **for** i **in** not\_available\_list:
276. **if** rtype == self.readRoomType(i) :
277. count +=1
278. numRoomType = numRoomType-count
280. **return** numRoomType

283. **def** reserve(self,room\_num,type\_res,first,last,nat,line,ema,tel,addr,from\_date,to\_date = "9999/12/31"):
284. s = "insert into reserve values(?,?,?,?,?,?,?,?,?,?,?)"
285. self.cur.execute(s,(room\_num,type\_res,first,last,nat,line,ema,tel,addr,from\_date,to\_date))
287. self.cur.execute("insert into roomstatus values(?,?,?)",(room\_num,from\_date,to\_date))
288. self.cn.commit()
290. **def** searchByName(self,f\_name,l\_name):
291. s = "select \* from reserve where first\_name = ? or last\_name = ?"
292. self.cur.execute(s,(f\_name,l\_name))
293. out = self.cur.fetchall()
294. **return** out
296. **def** cancelReserve(self,room\_number,cin\_date,cout\_date):
297. s = "delete from reserve where room\_number = ? and check\_in\_date = ? and check\_out\_date = ?"
298. self.cur.execute(s,(room\_number,cin\_date,cout\_date))
299. self.cn.commit()
301. s = "delete from roomstatus where room\_number = ? and not\_avail\_from = ? and not\_avail\_to = ?"
302. self.cur.execute(s,(room\_number,cin\_date,cout\_date))
303. self.cn.commit()
305. **def** checkInReserve(self,room\_num,type\_res,first,last,nat,line,ema,tel,addr,from\_date,to\_date = "9999/12/31"):
306. s = "insert into resident values(?,?,?,?,?,?,?,?,?,?,?)"
307. self.cur.execute(s,(room\_num,type\_res,first,last,nat,line,ema,tel,addr,from\_date,to\_date))
309. self.cur.execute("insert into roomstatus values(?,?,?)",(room\_num,from\_date,to\_date))
311. s = "delete from reserve where room\_number = ? and check\_in\_date = ? and check\_out\_date = ?"
312. self.cur.execute(s,(room\_num,from\_date,to\_date))
313. self.cn.commit()
315. **def** checkInWalk(self,room\_num,type\_res,first,last,nat,line,ema,tel,addr,from\_date,to\_date = "9999/12/31"):
316. s = "insert into resident values(?,?,?,?,?,?,?,?,?,?,?)"
317. self.cur.execute(s,(room\_num,type\_res,first,last,nat,line,ema,tel,addr,from\_date,to\_date))
319. self.cur.execute("insert into roomstatus values(?,?,?)",(room\_num,from\_date,to\_date))
320. self.cn.commit()
322. **def** checkOut(self,room\_number):
323. s = "select check\_in\_date from resident where room\_number = ?"
324. self.cur.execute(s,(room\_number,))
325. temp = self.cur.fetchone()
327. #FORCE NEW CHECKOUT DATE
328. today = str(datetime.date.today())
329. today = today.replace("-","/")
330. s = "update resident set check\_out\_date = ? where room\_number = ?"
331. self.cur.execute(s,(today,room\_number,))
332. self.cn.commit()
334. s = "insert into residenthistory select \* from resident where room\_number = ?"
335. self.cur.execute(s,(room\_number,))
336. self.cn.commit()
338. s = "delete from resident where room\_number = ?"
339. self.cur.execute(s,(room\_number,))
340. self.cn.commit()
342. s = "delete from roomstatus where room\_number = ? and not\_avail\_from = ?"
343. self.cur.execute(s,(room\_number,temp[0]))
344. self.cn.commit()

347. **def** deleteFronRoomstatus(self,room\_number):
348. s = "select check\_in\_date from resident where room\_number = ?"
349. self.cur.execute(s,(room\_number,))
350. temp = self.cur.fetchone()
352. s = "delete from roomstatus where room\_number = ? and not\_avail\_from = ?"
353. self.cur.execute(s,(room\_number,temp[0]))
354. self.cn.commit()
356. **def** setting\_daily(self,new\_daily,room\_type):
357. s = "select min(daily\_price) from room where room\_type = ?"
358. self.cur.execute(s,(room\_type,))
359. out = self.cur.fetchall()
361. s = "update room set default\_d = ? where room\_type = ?"
362. self.cur.execute(s,(out[0][0],room\_type))
363. self.cn.commit()
365. s = "update room set daily\_price = ? where room\_type = ?"
366. self.cur.execute(s,(new\_daily,room\_type))
367. self.cn.commit()
369. **def** setting\_monthly(self,new\_monthly,room\_type):
370. s = "select min(monthly\_price) from room where room\_type = ?"
371. self.cur.execute(s,(room\_type,))
372. out = self.cur.fetchall()
374. s = "update room set default\_m = ? where room\_type = ?"
375. self.cur.execute(s,(out[0][0],room\_type))
376. self.cn.commit()
378. s = "update room set monthly\_price = ? where room\_type = ?"
379. self.cur.execute(s,(new\_monthly,room\_type))
380. self.cn.commit()
382. **def** getDefault\_m(self,room\_type):
383. s = "select min(default\_m) from room where room\_type = ?"
384. self.cur.execute(s,(room\_type,))
385. out = self.cur.fetchall()
387. s = "update room set monthly\_price = ? where room\_type = ?"
388. self.cur.execute(s,(out[0][0],room\_type))
389. self.cn.commit()
390. **return** out[0][0]
392. **def** getDefault\_d(self,room\_type):
393. s = "select min(default\_d) from room where room\_type = ?"
394. self.cur.execute(s,(room\_type,))
395. out = self.cur.fetchall()
397. s = "update room set daily\_price = ? where room\_type = ?"
398. self.cur.execute(s,(out[0][0],room\_type))
399. self.cn.commit()
400. **return** out[0][0]
402. **def** setUtilRate(self,util\_type,new\_rate):
403. s = "select min(rate) from utilities\_rate where util\_type = ?"
404. self.cur.execute(s,(util\_type,))
405. out = self.cur.fetchall()
407. s = "update utilities\_rate set default\_val = ? where util\_type = ?"
408. self.cur.execute(s,(out[0][0],util\_type))
409. self.cn.commit()
411. s = "update utilities\_rate set rate = ? where util\_type = ?"
412. self.cur.execute(s,(new\_rate,util\_type))
413. self.cn.commit()

416. **def** getDefault\_util(self,util\_type):
417. s = "select min(default\_val) from utilities\_rate where util\_type = ?"
418. self.cur.execute(s,(util\_type,))
419. out = self.cur.fetchall()
421. s = "update utilities\_rate set rate = ? where util\_type = ?"
422. self.cur.execute(s,(out[0][0],util\_type))
423. self.cn.commit()
424. **return** out[0][0]
426. **def** getUtil(self,room\_number,util\_type):
427. **if** (util\_type == "all"):
428. s = "select \* from utilities\_unit where room\_number = ?"
429. self.cur.execute(s,(room\_number,))
430. out = self.cur.fetchall()
432. **elif** (util\_type == "water"):
433. s = "select water\_meter\_unit from utilities\_unit where room\_number = ?"
434. self.cur.execute(s,(room\_number,))
435. out = self.cur.fetchall()
437. **elif** (util\_type == "electricity"):
438. s = "select electricity\_meter\_unit from utilities\_unit where room\_number = ?"
439. self.cur.execute(s,(room\_number,))
440. out = self.cur.fetchall()
441. **if** (len(out) == 0):
442. **return** "0"
443. **else**:
444. **return** out
446. **def** getUtilDate(self,room\_number):
447. s = "select month\_year from utilities\_unit where room\_number = ?"
448. self.cur.execute(s,(room\_number,))
449. out = self.cur.fetchall()
450. **if** (len(out) == 0):
451. **return** "None"
452. **else**:
453. **return** out
455. **def** insertUtil(self,room\_number,month\_year,water\_meter\_unit,electricity\_meter\_unit):
456. s = "insert into utilities\_unit values(?,?,?,?)"
457. self.cur.execute(s,(room\_number,month\_year,water\_meter\_unit,electricity\_meter\_unit))
458. self.cn.commit()
460. **def** editUtil(self,room\_number,month\_year,new\_water\_meter\_unit,new\_electricity\_meter\_unit):
461. s = "update utilities\_unit set water\_meter\_unit = ?,electricity\_meter\_unit = ? where room\_number = ? and month\_year = ?"
462. self.cur.execute(s,(new\_water\_meter\_unit,new\_electricity\_meter\_unit,room\_number,month\_year))
463. self.cn.commit()
465. **def** setPassword(self,mode,new\_pass):
466. **if**  mode == "manager":
467. s = "update password set manager\_password = ?"
468. **else**:
469. s = "update password set receptionist\_password = ?"
470. self.cur.execute(s,(new\_pass,))
471. self.cn.commit()
473. **def** getPassword(self,mode):
474. **if**  mode == "manager":
475. s = "select manager\_password from password"
477. **else**:
478. s = "select receptionist\_password from password"
480. self.cur.execute(s)
481. out = self.cur.fetchone()
482. **return** out[0]
484. **def** getRoomPrice(self,room\_type):
485. s = "select daily\_price,monthly\_price from room where room\_type = ?"
486. self.cur.execute(s,(room\_type,))
487. out = self.cur.fetchall() ##[0][0] =daily [0][1] = monthly
488. **return** out
490. **def** getCheckinoutDate(self, room\_num):
491. s = ("select check\_in\_date, check\_out\_date from resident where room\_number = ?")
492. self.cur.execute(s, (room\_num,))
493. out = self.cur.fetchall()
494. **return** out[0][0]
496. **def** getCheckinDate(self,room\_num):
497. s = ("select check\_in\_date from resident where room\_number = ?")
498. self.cur.execute(s,(room\_num,))
499. out = self.cur.fetchall()
500. **return** out[0][0]
502. **def** setDefaultPasswordForManager(self):
503. i = "sys"
504. self.cur.execute("update password set manager\_password = ?",(i,))
505. self.cn.commit()
507. **def** setDefaultPasswordForReceptionist(self):
508. i = "sys"
509. self.cur.execute("update password set receptionist\_password = ?",(i,))
510. self.cn.commit()
512. **def** checkExceedCheckoutDate(self,room\_number):
513. s = "select check\_out\_date from resident where room\_number = ?"
514. self.cur.execute(s,(room\_number,))
515. chkout = self.cur.fetchone()
516. chkout = chkout[0]
517. chkout = chkout.split('/')
518. y = int(chkout[0])
519. m = int(chkout[1])
520. d = int(chkout[2])
522. chkout = datetime.datetime(y,m,d).date()
524. **if** datetime.date.today() > chkout :
525. **return** True
526. **else** :
527. **return** False
529. **def** checkStatusIftheroomisCheckedIn(self, room\_number):
530. self.cur.execute("select room\_number from roomstatus where room\_number not in (select room\_number from reserve)")
531. out = self.cur.fetchall()
532. temp = []
533. l = len(out)+1
534. **for** i **in** range(l):
535. **if**(i == len(out)):
536. **continue**
537. **else**:
538. temp.append(out[i][0])
540. **if** room\_number **in** temp:
541. **return** True
542. **else**:
543. **return** False
545. **def** checkStatusIftheroomisReserved(self, room\_number):
546. self.cur.execute("select room\_number from roomstatus where room\_number not in (select room\_number from resident)")
547. out = self.cur.fetchall()
548. temp = []
549. l = len(out)+1
550. **for** i **in** range(l):
551. **if**(i == len(out)):
552. **continue**
553. **else**:
554. temp.append(out[i][0])
556. **if** room\_number **in** temp:
557. **return** True
558. **else**:
559. **return** False
561. **def** checkReserveByCheckInDay(self,checkin\_day):
562. s = "select \* from reserve where ? = check\_in\_date"
563. self.cur.execute(s,(checkin\_day,))
564. out = self.cur.fetchall()
565. **return** out
567. **def** checkReservedResidentInfoByRoomNo(self,room\_number):
568. s = "select \* from reserve where ? = room\_number"
569. self.cur.execute(s,(room\_number,))
570. out = self.cur.fetchall()
571. **return** out[0]
573. **def** allpassword(self):
574. self.cur.execute("select \* from password")
575. out = self.cur.fetchall()
576. **return** out
578. **def** allutilities\_unit(self):
579. self.cur.execute("select \* from utilities\_unit order by room\_number, month\_year")
580. out = self.cur.fetchall()
581. **return** out
582. **def** allutilities\_rate(self):
583. self.cur.execute("select \* from utilities\_rate")
584. out = self.cur.fetchall()
585. **return** out
586. **def** allroomstatus(self):
587. self.cur.execute("select \* from roomstatus")
588. out = self.cur.fetchall()
589. **return** out
590. **def** allreserve(self):
591. self.cur.execute("select \* from reserve")
593. out = self.cur.fetchall()
594. **return** out
595. **def** allresident(self):
596. self.cur.execute("select \* from resident")
597. out = self.cur.fetchall()
598. **return** out
599. **def** allresidenthistory(self):
600. self.cur.execute("select \* from residenthistory")
601. out = self.cur.fetchall()
602. **return** out
603. **def** allroom(self):
604. self.cur.execute("select \* from room")
605. out = self.cur.fetchall()
606. **return** out
607. **def** test(self,from\_date,to\_date):
608. s = "select room\_number from roomstatus where ? between not\_avail\_from and not\_avail\_to and ? between not\_avail\_from and not\_avail\_to"
609. self.cur.execute(s,(from\_date,to\_date))
610. out = self.cur.fetchall()
611. **return** out
613. **def** IsAvailable(self,room\_number):
614. s = "select room\_number from resident where room\_number = ?"
615. self.cur.execute(s,(room\_number,))
616. out = self.cur.fetchall()
617. **if** len(out) == 0:
618. **return** True
619. **else**:
620. **return** False
621. '''''
622. def allutilities\_unit\_by\_room\_number(self,room\_number):
623. self.cur.execute(s,"select \* from utilities\_unit order by room\_number, month\_year where room\_number = ?")
624. out = self.cur.fetchall()
625. return out'''


629. '''''
630. def main():
631. w = DatabaseSys()
633. w.createtableResident()
634. w.createtableUtilities() #NEW
635. w.createtableUtilitieRate()#NEW
636. w.createtableRoom()
637. w.createtableReserve()
638. w.createtableRoomState()
639. w.createtableResidentHistory()
640. w.createtablePassword()#NEW
642. w.setDefaultPasswordForManager()
643. w.setDefaultPasswordForReceptionist()
644. w.setting\_daily(500,"studio") #pass
645. w.setting\_daily(800,"1-bedroom") #pass
646. w.setting\_daily(1000,"2-bedroom") #pass
648. w.setting\_monthly(4500,"studio") #pass
649. w.setting\_monthly(6500,"1-bedroom") #pass
650. w.setting\_monthly(8500,"2-bedroom") #pass
652. w.setUtilRate("water","23.23") #pass
653. w.setUtilRate("electricity","50.23") #pass
655. w.checkInWalk("102","daily","I dont ","leave","111011110","line","@email","0811112222","home,sweet,home","2017/05/25","2017/05/26")
656. w.reserve('107', 'monthly', 'Jane', 'StayFor1month', '11111111', '', '', '081111111', 'a1,a2,a3', '2017/06/01', '2017/06/30')
657. w.checkInWalk('101', 'monthly', 'Sajeerat', 'Aussava', '5809xxxx', '', '', '0879997777', 'reserve,for,stay', '2017/05/28', '9999/12/31')
659. #w.deleteFronRoomstatus("101")
660. print("status")
661. for i in w.allroomstatus():
662. print(i)
663. print("resident")
664. for i in w.allresident():
665. print(i)
666. print("reserve")
667. for i in w.allreserve():
668. print(i)
670. main()'''
672. '''''
673. w = DatabaseSys()
674. print("status")
675. for i in w.allroomstatus():
676. print(i)
677. print("resident")
678. for i in w.allresident():
679. print(i)
680. print("reserve")
681. for i in w.allreserve():
682. print(i)
683. '''
684. '''''
685. w = DatabaseSys()
686. w.checkInWalk("102","daily","I dont ","leave","111011110","line","@email","0811112222","home,sweet,home","2017/05/25","2017/05/26")
687. w.reserve('107', 'monthly', 'Jane', 'StayFor1month', '11111111', '', '', '081111111', 'a1,a2,a3', '2017/06/01', '2017/06/30')
688. w.checkInWalk('101', 'monthly', 'Sajeerat', 'Aussava', '5809xxxx', '', '', '0879997777', 'reserve,for,stay', '2017/05/28', '9999/12/31')
689. '''
691. #w = DatabaseSys()
692. #w.checkInWalk("202","daily","I dont ","leave","111011110","line","@email","0811112222","home,sweet,home","2017/05/25","2017/05/26")
693. '''''
694. w = DatabaseSys()
695. print(w.allresident())'''
696. '''''
697. w = DatabaseSys()
698. w.checkInWalk('101', 'monthly', 'Sajeerat', 'Aussava', '5809xxxx', '', '', '0879997777', 'reserve,for,stay', '2017/05/28', '9999/12/31')
699. t =w.searchAllAvailableRoom("1","2017/05/26","2017/6/5")
700. for i in t:
701. print(i)'''

Login\_ui.py

1. **from** Menu\_ui **import** \*
2. **from** constant\_initialization **import** \*
3. **from** DatabaseSystem **import** \*
5. **class** Login(QMainWindow):
6. **def** \_\_init\_\_(self):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiLogin.ui", None)
12. self.db = DatabaseSys()
14. self.setWindowTitle("SN Group Manager")
15. self.setWindowIcon(QIcon('images/icon.png'))
16. self.setCentralWidget(form)
17. self.setFixedSize(display\_width,display\_height)
19. self.login\_btn = form.findChild(QPushButton, "login\_btn")
20. self.login\_btn.clicked.connect(self.menu)
22. self.username = form.findChild(QLineEdit, "username\_lineEdit")
23. self.password = form.findChild(QLineEdit, "password\_lineEdit")
25. self.password.setEchoMode(QLineEdit.Password)
27. **def** menu(self):
28. rec\_password = self.db.getPassword("receptionist")
29. manager\_password = "manager"
31. **if** self.username.text() == "receptionist" **and** self.password.text() == rec\_password:
32. self.menu = Menu("receptionist")
33. self.menu.show()
34. self.hide()
35. **elif** self.username.text() == "manager" **or** self.password.text() == manager\_password:
36. self.menu = Menu("manager")
37. self.menu.show()
38. self.hide()
39. **else**:
40. self.popUP("invalid password or username")
42. **def** popUP(self, txt):
43. dialog = QDialog(self)
44. layout = QVBoxLayout()
45. label = QLabel(str(txt))
46. layout.addWidget(label)
47. close\_button = QPushButton("Close")
48. close\_button.clicked.connect(dialog.close)
49. layout.addWidget(close\_button)
50. dialog.setLayout(layout)
51. dialog.show()
53. **def** paintEvent(self, e): #for adding an image
54. p = QPainter()
55. p.begin(self)
56. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
57. p.drawImage(QRect((display\_width-title\_font\_width)/2, 125, 543, 84), title\_font\_img)
58. p.end()

61. **def** main():
62. app = QApplication(sys.argv)
63. w = Login()
65. w.show()
67. **return** app.exec\_()
69. **if** \_\_name\_\_ == "\_\_main\_\_":
70. sys.exit(main())

Menu\_ui.py

1. **from** Login\_ui **import** \*
2. **from** Information\_ui **import** \*
3. **from** Reservation\_ui **import** \*
4. **from** CancelReservation\_ui **import** \*
5. **from** CheckinWalkin\_ui **import** \*
6. **from** CheckinReserved\_ui **import** \*
7. **from** Checkout\_ui **import** \*
8. **from** Setting\_ui **import** \*
9. **from** Billing\_ui **import** \*
11. **class** Menu(QMainWindow):
12. **def** \_\_init\_\_(self,mode):
13. QMainWindow.\_\_init\_\_(self, None)
15. loader = QUiLoader()
16. form = loader.load("UI/uiMenu.ui", None)
18. self.setWindowTitle("SN Group Manager")
19. self.setWindowIcon(QIcon('images/icon.png'))
20. self.setCentralWidget(form)
21. self.setFixedSize(display\_width,display\_height)
23. self.information\_btn = form.findChild(QPushButton, "information\_btn")
24. self.information\_btn.clicked.connect(self.information\_ui)
26. self.reservation\_btn = form.findChild(QPushButton, "reservation\_btn")
27. self.reservation\_btn.clicked.connect(self.reservation\_ui)
29. self.reservation\_btn = form.findChild(QPushButton, "cancel\_reservation\_btn")
30. self.reservation\_btn.clicked.connect(self.cancel\_reservation\_ui)
32. self.checkin\_walk\_in\_btn = form.findChild(QPushButton, "check\_in\_walk\_in\_btn")
33. self.checkin\_walk\_in\_btn.clicked.connect(self.checkin\_walkin\_ui)
35. self.checkin\_reserved\_btn = form.findChild(QPushButton, "check\_in\_reserved\_btn")
36. self.checkin\_reserved\_btn.clicked.connect(self.checkin\_reserved\_ui)
38. self.check\_out\_btn = form.findChild(QPushButton, "check\_out\_btn")
39. self.check\_out\_btn.clicked.connect(self.check\_out\_ui)
41. self.billing\_btn = form.findChild(QPushButton, "billing\_btn")
42. self.billing\_btn.clicked.connect(self.billing\_ui)
44. self.setting\_btn = form.findChild(QPushButton, "setting\_btn")
45. self.setting\_btn.clicked.connect(self.setting\_ui)
47. self.status = form.findChild(QLabel, "Status")
48. self.mode = mode
49. **if** self.mode == "manager":
50. self.status.setText("Manager")
51. **else** :
52. self.status.setText("Receptionist")
53. self.billing\_btn.setEnabled(False)
54. self.setting\_btn.setEnabled(False)
56. **def** information\_ui(self):
57. self.information = Information(self)
58. self.information.show()
59. self.hide()
61. **def** reservation\_ui(self):
62. self.reservation = Reservation(self)
63. self.reservation.show()
64. self.hide()
66. **def** cancel\_reservation\_ui(self):
67. self.cancel\_reservation = CancelReservation(self)
68. self.cancel\_reservation.show()
69. self.hide()
71. **def** checkin\_walkin\_ui(self):
72. self.checkin\_walkin = CheckinWalkin(self)
73. self.checkin\_walkin.show()
74. self.hide()
76. **def** checkin\_reserved\_ui(self):
77. self.checkin\_reserved = CheckinReserved(self)
78. self.checkin\_reserved.show()
79. self.hide()
81. **def** check\_out\_ui(self):
82. self.check\_out = Checkout(self)
83. self.check\_out.show()
84. self.hide()
86. **def** billing\_ui (self):
87. self.billing = Billing(self)
88. self.billing.show()
90. **def** setting\_ui(self):
91. self.setting = Setting(self)
92. self.setting.show()
93. self.hide()
95. **def** paintEvent(self, e): #for adding an image
96. p = QPainter()
97. p.begin(self)
98. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
99. p.drawImage(QRect((display\_width-title\_font\_width)/2, 45, 543, 84), title\_font\_img)
100. p.drawImage(QRect(40, 150, 399, 365), building\_img)
101. p.end()

Information\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** UtilitiesRecord\_ui **import** \*
3. **from** DatabaseSystem **import** \*
4. **from** ResidentHistory\_ui **import** \*
5. **from** AllCurrentResidents\_ui **import** \*
7. **class** Information(QMainWindow):
8. **def** \_\_init\_\_(self,menu):
9. QMainWindow.\_\_init\_\_(self, None)
11. loader = QUiLoader()
12. form = loader.load("UI/uiInformation.ui", None)
14. self.setWindowTitle("SN Group Manager")
15. self.setWindowIcon(QIcon('images/icon.png'))
16. self.setCentralWidget(form)
17. self.setFixedSize(display\_width,display\_height)
19. self.menu = menu
21. self.building\_sw = form.findChild(QStackedWidget, "Building\_sw")
23. self.back\_btn = form.findChild(QPushButton, "back\_btn")
24. self.back\_btn.clicked.connect(self.back)
26. self.F1\_btn = form.findChild(QPushButton, "F1\_btn")
27. self.F1\_btn.clicked.connect(**lambda**: self.floor(0))
29. self.F2\_btn = form.findChild(QPushButton, "F2\_btn")
30. self.F2\_btn.clicked.connect(**lambda**: self.floor(1))
32. self.F3\_btn = form.findChild(QPushButton, "F3\_btn")
33. self.F3\_btn.clicked.connect(**lambda**: self.floor(2))
35. self.F4\_btn = form.findChild(QPushButton, "F4\_btn")
36. self.F4\_btn.clicked.connect(**lambda**: self.floor(3))
38. self.F5\_btn = form.findChild(QPushButton, "F5\_btn")
39. self.F5\_btn.clicked.connect(**lambda**: self.floor(4))
41. self.db = DatabaseSys()
43. #Room
44. self.room101 = form.findChild(QPushButton, "pb101")
45. self.room101.clicked.connect(**lambda**: self.showInfo(101))
46. self.btnColor(self.room101, "101")
47. self.room102 = form.findChild(QPushButton, "pb102")
48. self.room102.clicked.connect(**lambda**: self.showInfo(102))
49. self.btnColor(self.room102, "102")
50. self.room103 = form.findChild(QPushButton, "pb103")
51. self.room103.clicked.connect(**lambda**: self.showInfo(103))
52. self.btnColor(self.room103, "103")
53. self.room104 = form.findChild(QPushButton, "pb104")
54. self.room104.clicked.connect(**lambda**: self.showInfo(104))
55. self.btnColor(self.room104, "104")
56. self.room105 = form.findChild(QPushButton, "pb105")
57. self.room105.clicked.connect(**lambda**: self.showInfo(105))
58. self.btnColor(self.room105, "105")
59. self.room106 = form.findChild(QPushButton, "pb106")
60. self.room106.clicked.connect(**lambda**: self.showInfo(106))
61. self.btnColor(self.room106, "106")
62. self.room107 = form.findChild(QPushButton, "pb107")
63. self.room107.clicked.connect(**lambda**: self.showInfo(107))
64. self.btnColor(self.room107, "107")
65. self.room108 = form.findChild(QPushButton, "pb108")
66. self.room108.clicked.connect(**lambda**: self.showInfo(108))
67. self.btnColor(self.room108, "108")
68. self.room109 = form.findChild(QPushButton, "pb109")
69. self.room109.clicked.connect(**lambda**: self.showInfo(109))
70. self.btnColor(self.room109, "109")
71. self.room110 = form.findChild(QPushButton, "pb110")
72. self.room110.clicked.connect(**lambda**: self.showInfo(110))
73. self.btnColor(self.room110, "110")
74. self.room111 = form.findChild(QPushButton, "pb111")
75. self.room111.clicked.connect(**lambda**: self.showInfo(111))
76. self.btnColor(self.room111, "111")
77. self.room112 = form.findChild(QPushButton, "pb112")
78. self.room112.clicked.connect(**lambda**: self.showInfo(112))
79. self.btnColor(self.room112, "112")
81. self.room201 = form.findChild(QPushButton, "pb201")
82. self.room201.clicked.connect(**lambda**: self.showInfo(201))
83. self.btnColor(self.room201, "201")
84. self.room202 = form.findChild(QPushButton, "pb202")
85. self.room202.clicked.connect(**lambda**: self.showInfo(202))
86. self.btnColor(self.room202, "202")
87. self.room203 = form.findChild(QPushButton, "pb203")
88. self.room203.clicked.connect(**lambda**: self.showInfo(203))
89. self.btnColor(self.room203, "203")
90. self.room204 = form.findChild(QPushButton, "pb204")
91. self.room204.clicked.connect(**lambda**: self.showInfo(204))
92. self.btnColor(self.room204, "204")
93. self.room205 = form.findChild(QPushButton, "pb205")
94. self.room205.clicked.connect(**lambda**: self.showInfo(205))
95. self.btnColor(self.room205, "205")
96. self.room206 = form.findChild(QPushButton, "pb206")
97. self.room206.clicked.connect(**lambda**: self.showInfo(206))
98. self.btnColor(self.room206, "206")
99. self.room207 = form.findChild(QPushButton, "pb207")
100. self.room207.clicked.connect(**lambda**: self.showInfo(207))
101. self.btnColor(self.room207, "207")
102. self.room208 = form.findChild(QPushButton, "pb208")
103. self.room208.clicked.connect(**lambda**: self.showInfo(208))
104. self.btnColor(self.room208, "208")
105. self.room209 = form.findChild(QPushButton, "pb209")
106. self.room209.clicked.connect(**lambda**: self.showInfo(209))
107. self.btnColor(self.room209, "209")
108. self.room210 = form.findChild(QPushButton, "pb210")
109. self.room210.clicked.connect(**lambda**: self.showInfo(210))
110. self.btnColor(self.room210, "210")
111. self.room211 = form.findChild(QPushButton, "pb211")
112. self.room211.clicked.connect(**lambda**: self.showInfo(211))
113. self.btnColor(self.room211, "211")
114. self.room212 = form.findChild(QPushButton, "pb212")
115. self.room212.clicked.connect(**lambda**: self.showInfo(212))
116. self.btnColor(self.room212, "212")
117. self.room213 = form.findChild(QPushButton, "pb213")
118. self.room213.clicked.connect(**lambda**: self.showInfo(213))
119. self.btnColor(self.room213, "213")
120. self.room214 = form.findChild(QPushButton, "pb214")
121. self.room214.clicked.connect(**lambda**: self.showInfo(214))
122. self.btnColor(self.room214, "214")
123. self.room215 = form.findChild(QPushButton, "pb215")
124. self.room215.clicked.connect(**lambda**: self.showInfo(215))
125. self.btnColor(self.room215, "215")
126. self.room216 = form.findChild(QPushButton, "pb216")
127. self.room216.clicked.connect(**lambda**: self.showInfo(216))
128. self.btnColor(self.room216, "216")
129. self.room217 = form.findChild(QPushButton, "pb217")
130. self.room217.clicked.connect(**lambda**: self.showInfo(217))
131. self.btnColor(self.room217, "217")
133. self.room301 = form.findChild(QPushButton, "pb301")
134. self.room301.clicked.connect(**lambda**: self.showInfo(301))
135. self.btnColor(self.room301, "301")
136. self.room302 = form.findChild(QPushButton, "pb302")
137. self.room302.clicked.connect(**lambda**: self.showInfo(302))
138. self.btnColor(self.room302, "302")
139. self.room303 = form.findChild(QPushButton, "pb303")
140. self.room303.clicked.connect(**lambda**: self.showInfo(303))
141. self.btnColor(self.room303, "303")
142. self.room304 = form.findChild(QPushButton, "pb304")
143. self.room304.clicked.connect(**lambda**: self.showInfo(304))
144. self.btnColor(self.room304, "304")
145. self.room305 = form.findChild(QPushButton, "pb305")
146. self.room305.clicked.connect(**lambda**: self.showInfo(305))
147. self.btnColor(self.room305, "305")
148. self.room306 = form.findChild(QPushButton, "pb306")
149. self.room306.clicked.connect(**lambda**: self.showInfo(306))
150. self.btnColor(self.room306, "306")
151. self.room307 = form.findChild(QPushButton, "pb307")
152. self.room307.clicked.connect(**lambda**: self.showInfo(307))
153. self.btnColor(self.room307, "307")
154. self.room308 = form.findChild(QPushButton, "pb308")
155. self.room308.clicked.connect(**lambda**: self.showInfo(308))
156. self.btnColor(self.room308, "308")
157. self.room309 = form.findChild(QPushButton, "pb309")
158. self.room309.clicked.connect(**lambda**: self.showInfo(309))
159. self.btnColor(self.room309, "309")
160. self.room310 = form.findChild(QPushButton, "pb310")
161. self.room310.clicked.connect(**lambda**: self.showInfo(310))
162. self.btnColor(self.room310, "310")
163. self.room311 = form.findChild(QPushButton, "pb311")
164. self.room311.clicked.connect(**lambda**: self.showInfo(311))
165. self.btnColor(self.room311, "311")
166. self.room312 = form.findChild(QPushButton, "pb312")
167. self.room312.clicked.connect(**lambda**: self.showInfo(312))
168. self.btnColor(self.room312, "312")
169. self.room313 = form.findChild(QPushButton, "pb313")
170. self.room313.clicked.connect(**lambda**: self.showInfo(313))
171. self.btnColor(self.room313, "313")
172. self.room314 = form.findChild(QPushButton, "pb314")
173. self.room314.clicked.connect(**lambda**: self.showInfo(314))
174. self.btnColor(self.room314, "314")
175. self.room315 = form.findChild(QPushButton, "pb315")
176. self.room315.clicked.connect(**lambda**: self.showInfo(315))
177. self.btnColor(self.room315, "315")
178. self.room316 = form.findChild(QPushButton, "pb316")
179. self.room316.clicked.connect(**lambda**: self.showInfo(316))
180. self.btnColor(self.room316, "316")
181. self.room317 = form.findChild(QPushButton, "pb317")
182. self.room317.clicked.connect(**lambda**: self.showInfo(317))
183. self.btnColor(self.room317, "317")
185. self.room401 = form.findChild(QPushButton, "pb401")
186. self.room401.clicked.connect(**lambda**: self.showInfo(401))
187. self.btnColor(self.room401, "401")
188. self.room402 = form.findChild(QPushButton, "pb402")
189. self.room402.clicked.connect(**lambda**: self.showInfo(402))
190. self.btnColor(self.room402, "402")
191. self.room403 = form.findChild(QPushButton, "pb403")
192. self.room403.clicked.connect(**lambda**: self.showInfo(403))
193. self.btnColor(self.room403, "403")
194. self.room404 = form.findChild(QPushButton, "pb404")
195. self.room404.clicked.connect(**lambda**: self.showInfo(404))
196. self.btnColor(self.room404, "404")
197. self.room405 = form.findChild(QPushButton, "pb405")
198. self.room405.clicked.connect(**lambda**: self.showInfo(405))
199. self.btnColor(self.room405, "405")
200. self.room406 = form.findChild(QPushButton, "pb406")
201. self.room406.clicked.connect(**lambda**: self.showInfo(406))
202. self.btnColor(self.room406, "406")
203. self.room407 = form.findChild(QPushButton, "pb407")
204. self.room407.clicked.connect(**lambda**: self.showInfo(407))
205. self.btnColor(self.room407, "407")
206. self.room408 = form.findChild(QPushButton, "pb408")
207. self.room408.clicked.connect(**lambda**: self.showInfo(408))
208. self.btnColor(self.room408, "408")
209. self.room409 = form.findChild(QPushButton, "pb409")
210. self.room409.clicked.connect(**lambda**: self.showInfo(409))
211. self.btnColor(self.room409, "409")
212. self.room410 = form.findChild(QPushButton, "pb410")
213. self.room410.clicked.connect(**lambda**: self.showInfo(410))
214. self.btnColor(self.room410, "410")
215. self.room411 = form.findChild(QPushButton, "pb411")
216. self.room411.clicked.connect(**lambda**: self.showInfo(411))
217. self.btnColor(self.room411, "411")
218. self.room412 = form.findChild(QPushButton, "pb412")
219. self.room412.clicked.connect(**lambda**: self.showInfo(412))
220. self.btnColor(self.room412, "412")
221. self.room413 = form.findChild(QPushButton, "pb413")
222. self.room413.clicked.connect(**lambda**: self.showInfo(413))
223. self.btnColor(self.room413, "413")
224. self.room414 = form.findChild(QPushButton, "pb414")
225. self.room414.clicked.connect(**lambda**: self.showInfo(414))
226. self.btnColor(self.room414, "414")
227. self.room415 = form.findChild(QPushButton, "pb415")
228. self.room415.clicked.connect(**lambda**: self.showInfo(415))
229. self.btnColor(self.room415, "415")
230. self.room416 = form.findChild(QPushButton, "pb416")
231. self.room416.clicked.connect(**lambda**: self.showInfo(416))
232. self.btnColor(self.room416, "416")
233. self.room417 = form.findChild(QPushButton, "pb417")
234. self.room417.clicked.connect(**lambda**: self.showInfo(417))
235. self.btnColor(self.room417, "417")
237. self.room501 = form.findChild(QPushButton, "pb501")
238. self.room501.clicked.connect(**lambda**: self.showInfo(501))
239. self.btnColor(self.room501, "501")
240. self.room502 = form.findChild(QPushButton, "pb502")
241. self.room502.clicked.connect(**lambda**: self.showInfo(502))
242. self.btnColor(self.room502, "502")
243. self.room503 = form.findChild(QPushButton, "pb503")
244. self.room503.clicked.connect(**lambda**: self.showInfo(503))
245. self.btnColor(self.room503, "503")
246. self.room504 = form.findChild(QPushButton, "pb504")
247. self.room504.clicked.connect(**lambda**: self.showInfo(504))
248. self.btnColor(self.room504, "504")
249. self.room505 = form.findChild(QPushButton, "pb505")
250. self.room505.clicked.connect(**lambda**: self.showInfo(505))
251. self.btnColor(self.room505, "505")
252. self.room506 = form.findChild(QPushButton, "pb506")
253. self.room506.clicked.connect(**lambda**: self.showInfo(506))
254. self.btnColor(self.room506, "506")
255. self.room507 = form.findChild(QPushButton, "pb507")
256. self.room507.clicked.connect(**lambda**: self.showInfo(507))
257. self.btnColor(self.room507, "507")
258. self.room508 = form.findChild(QPushButton, "pb508")
259. self.room508.clicked.connect(**lambda**: self.showInfo(508))
260. self.btnColor(self.room508, "508")
261. self.room509 = form.findChild(QPushButton, "pb509")
262. self.room509.clicked.connect(**lambda**: self.showInfo(509))
263. self.btnColor(self.room509, "509")
264. self.room510 = form.findChild(QPushButton, "pb510")
265. self.room510.clicked.connect(**lambda**: self.showInfo(510))
266. self.btnColor(self.room510, "510")
267. self.room511 = form.findChild(QPushButton, "pb511")
268. self.room511.clicked.connect(**lambda**: self.showInfo(511))
269. self.btnColor(self.room511, "511")
270. self.room512 = form.findChild(QPushButton, "pb512")
271. self.room512.clicked.connect(**lambda**: self.showInfo(512))
272. self.btnColor(self.room512, "512")
273. self.room513 = form.findChild(QPushButton, "pb513")
274. self.room513.clicked.connect(**lambda**: self.showInfo(513))
275. self.btnColor(self.room513, "513")
276. self.room514 = form.findChild(QPushButton, "pb514")
277. self.room514.clicked.connect(**lambda**: self.showInfo(514))
278. self.btnColor(self.room514, "514")
279. self.room515 = form.findChild(QPushButton, "pb515")
280. self.room515.clicked.connect(**lambda**: self.showInfo(515))
281. self.btnColor(self.room515, "515")
283. self.roomNo = 101
284. self.floorNo = 1
286. self.utilities\_record\_btn = form.findChild(QPushButton, "utilities\_record\_btn")
287. self.utilities\_record\_btn.clicked.connect(**lambda**: self.utilities\_record\_call(self.roomNo))
288. #self.utilities\_record\_btn.clicked.connect(self.utilities\_record\_call)
289. self.utilities\_record\_btn.setEnabled(False)
291. self.resHis\_btn = form.findChild(QPushButton, "resHis\_btn")
292. self.resHis\_btn.clicked.connect(self.resident\_history)
294. self.allRes\_btn = form.findChild(QPushButton, "allRes\_btn")
295. self.allRes\_btn.clicked.connect(self.all\_current\_residents)
297. #listWidget
298. self.roomInfo\_listWidget = form.findChild(QListWidget, "roomInfo\_listWidget")
299. self.resInfo\_listWidget = form.findChild(QListWidget, "resInfo\_listWidget")
301. #Layout
302. self.room\_layout = QVBoxLayout()
303. self.res\_layout = QVBoxLayout()
305. **def** btnColor(self, btn, rN):
306. roomNo = rN
307. **if**(self.db.checkStatusIftheroomisCheckedIn(str(roomNo))==True) **and** (self.db.checkExceedCheckoutDate(str(roomNo)) == False):
308. self.setButtonColor(btn, 'orange')
309. **elif**(self.db.checkStatusIftheroomisCheckedIn(str(roomNo))==True) **and** (self.db.checkExceedCheckoutDate(str(roomNo)) == True):
310. self.setButtonColor(btn, 'red')
311. **else**:
312. self.setButtonColor(btn, 'green')
314. **def** setButtonColor(self, btn, color):
315. btn.setFlat(True)
316. palette = QPalette(btn.palette())
317. palette.setColor(QPalette.Button, QColor(color))
318. btn.setAutoFillBackground(True)
319. btn.setPalette(palette)
320. btn.update()
322. **def** getRoomNo(self):
323. **return** self.roomNo
325. **def** showInfo(self, roomNo):
326. self.roomNo = roomNo
327. #print(self.getRoomNo())
328. self.utilities\_record\_btn.setEnabled(True)
330. #room\_layout
331. self.clearInfo(self.room\_layout)
332. self.room\_layout.addWidget(QLabel(str("-------")+str(roomNo)+str("-------\n") + self.db.readRoomInfo(roomNo)))
333. self.roomInfo\_listWidget.setLayout(self.room\_layout)
334. self.roomInfo\_listWidget.show()
336. #res\_layout
337. self.clearInfo(self.res\_layout)
338. self.res\_layout.addWidget(QLabel(str("-------")+str(roomNo)+str("-------\n") + self.db.readResidentInfo(roomNo)))
339. self.resInfo\_listWidget.setLayout(self.res\_layout)
340. self.resInfo\_listWidget.show()

343. **def** clearInfo(self, layout):
344. **if** layout **is** **not** None:
345. **while** layout.count():
346. item = layout.takeAt(0)
347. widget = item.widget()
348. **if** widget **is** **not** None:
349. widget.deleteLater()
350. **else**:
351. self.clearInfo(item.layout)
353. **def** floor(self, idx):
354. self.floorNo = idx
355. self.building\_sw.setCurrentIndex(idx)
357. **def** utilities\_record\_call(self,roomNo):
358. self.utilities\_record = UtilitiesRecord(roomNo)
359. self.utilities\_record.show()
361. **def** resident\_history(self):
362. self.rh = ResidentHistory()
363. self.rh.show()
365. **def** all\_current\_residents(self):
366. self.rc = AllCurrentResidents()
367. self.rc.show()
369. **def** back(self):
370. self.clearInfo(self.room\_layout)
371. self.clearInfo(self.res\_layout)
372. self.utilities\_record\_btn.setEnabled(False)
373. self.menu.show()
374. self.hide()
376. **def** paintEvent(self, e): #for adding an image
377. p = QPainter()
378. p.begin(self)
379. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
380. p.end()

UtilitiesRecord\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** Information\_ui **import** \*
4. **from** DatabaseSystem **import** \*
6. **class** UtilitiesRecord(QMainWindow):
7. **def** \_\_init\_\_(self, roomNo=None):
8. QMainWindow.\_\_init\_\_(self, None)
10. loader = QUiLoader()
11. form = loader.load("UI/uiUtilitiesRecord.ui", None)
13. self.setWindowTitle("SN Group Manager")
14. self.setWindowIcon(QIcon('images/icon.png'))
15. self.setCentralWidget(form)
16. self.setFixedSize(display\_width\*0.8,display\_height\*0.8)
18. self.db = DatabaseSys()
20. self.tableWidget = form.findChild(QTableWidget,"uti\_tableWidget")
21. self.layout = QGridLayout()
23. self.roomNo = roomNo
25. self.showInfo(self.roomNo)
27. **def** showInfo(self, rN):
28. roomNo = rN
29. self.tableWidget.setRowCount(1)
30. self.tableWidget.setColumnCount(5)
31. self.layout.addWidget(self.tableWidget, 0,0)
32. self.tableWidget.setItem(0, 0, QTableWidgetItem(str(roomNo)))
33. self.tableWidget.setItem(0, 1, QTableWidgetItem(str(self.db.readResidentType(roomNo))))
34. self.tableWidget.setItem(0, 2, QTableWidgetItem(str(self.db.getUtilDate((roomNo)))))
35. self.tableWidget.setItem(0, 3, QTableWidgetItem(str(self.db.getUtil(roomNo,"electricity"))))
36. self.tableWidget.setItem(0, 4, QTableWidgetItem(str(self.db.getUtil(roomNo,"water"))))
37. self.tableWidget.show()
39. **def** paintEvent(self, e): #for adding an image
40. p = QPainter()
41. p.begin(self)
42. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
43. p.end()

ResidentHistory\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** Information\_ui **import** \*
3. **from** DatabaseSystem **import** \*
5. **class** ResidentHistory(QMainWindow):
6. **def** \_\_init\_\_(self):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiResidentHistory.ui", None)
12. self.setWindowTitle("SN Group Manager")
13. self.setWindowIcon(QIcon('images/icon.png'))
14. self.setCentralWidget(form)
15. self.setFixedSize(display\_width\*0.8,display\_height\*0.8)
17. self.db = DatabaseSys()
19. self.tableWidget = form.findChild(QTableWidget,"tableWidget")
20. self.layout = QGridLayout()
22. self.history = self.db.allresidenthistory()
24. self.showInfo(self.history)
26. **def** showInfo(self, History):
27. self.tableWidget.setRowCount(len(self.history))
28. self.tableWidget.setColumnCount(11)
29. self.layout.addWidget(self.tableWidget, 0,0)
30. row = 0
31. **for** eachRes **in** History:
32. self.tableWidget.setItem(row, 0, QTableWidgetItem(str(eachRes[0])))
33. self.tableWidget.setItem(row, 1, QTableWidgetItem(str(eachRes[1])))
34. self.tableWidget.setItem(row, 2, QTableWidgetItem(str(eachRes[2])))
35. self.tableWidget.setItem(row, 3, QTableWidgetItem(str(eachRes[3])))
36. self.tableWidget.setItem(row, 4, QTableWidgetItem(str(eachRes[4])))
37. self.tableWidget.setItem(row, 5, QTableWidgetItem(str(eachRes[5])))
38. self.tableWidget.setItem(row, 6, QTableWidgetItem(str(eachRes[6])))
39. self.tableWidget.setItem(row, 7, QTableWidgetItem(str(eachRes[7])))
40. self.tableWidget.setItem(row, 8, QTableWidgetItem(str(eachRes[8])))
41. self.tableWidget.setItem(row, 9, QTableWidgetItem(str(eachRes[9])))
42. self.tableWidget.setItem(row, 10, QTableWidgetItem(str(eachRes[10])))
43. row = row+1
45. **def** paintEvent(self, e): #for adding an image
46. p = QPainter()
47. p.begin(self)
48. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
49. p.end()

AllCurrentResidents\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** Information\_ui **import** \*
3. **from** DatabaseSystem **import** \*
5. **class** AllCurrentResidents(QMainWindow):
6. **def** \_\_init\_\_(self):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiAllCurrentResidents.ui", None)
12. self.setWindowTitle("SN Group Manager")
13. self.setWindowIcon(QIcon('images/icon.png'))
14. self.setCentralWidget(form)
15. self.setFixedSize(display\_width\*0.8,display\_height\*0.8)
17. self.db = DatabaseSys()
19. self.tableWidget = form.findChild(QTableWidget,"tableWidget")
20. self.layout = QGridLayout()
22. self.resident = self.db.allresident()

25. self.showInfo(self.resident)
27. **def** showInfo(self, Residents):
28. self.tableWidget.setRowCount(len(self.resident))
29. self.tableWidget.setColumnCount(11)
30. self.layout.addWidget(self.tableWidget, 0,0)
31. row = 0
32. **for** eachRes **in** Residents:
33. self.tableWidget.setItem(row, 0, QTableWidgetItem(str(eachRes[0])))
34. self.tableWidget.setItem(row, 1, QTableWidgetItem(str(eachRes[1])))
35. self.tableWidget.setItem(row, 2, QTableWidgetItem(str(eachRes[2])))
36. self.tableWidget.setItem(row, 3, QTableWidgetItem(str(eachRes[3])))
37. self.tableWidget.setItem(row, 4, QTableWidgetItem(str(eachRes[4])))
38. self.tableWidget.setItem(row, 5, QTableWidgetItem(str(eachRes[5])))
39. self.tableWidget.setItem(row, 6, QTableWidgetItem(str(eachRes[6])))
40. self.tableWidget.setItem(row, 7, QTableWidgetItem(str(eachRes[7])))
41. self.tableWidget.setItem(row, 8, QTableWidgetItem(str(eachRes[8])))
42. self.tableWidget.setItem(row, 9, QTableWidgetItem(str(eachRes[9])))
43. self.tableWidget.setItem(row, 10, QTableWidgetItem(str(eachRes[10])))
44. self.tableWidget.show()
45. row = row+1
47. **def** paintEvent(self, e): #for adding an image
48. p = QPainter()
49. p.begin(self)
50. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
51. p.end()

Reservation\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** FillResInfo\_ui **import** \*
3. **from** DatabaseSystem **import** \*
5. **class** Reservation(QMainWindow):
6. **def** \_\_init\_\_(self,menu):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiReservation.ui", None)
12. self.setWindowTitle("SN Group Manager")
13. self.setWindowIcon(QIcon('images/icon.png'))
14. self.setCentralWidget(form)
15. self.setFixedSize(display\_width,display\_height)
17. self.db = DatabaseSys()
19. self.layout = QVBoxLayout()
21. self.tableWidget = form.findChild(QTableWidget, "tableWidget")
23. self.searchedDate = []
25. self.studio\_count = form.findChild(QLineEdit,"studio\_lineEdit")
26. self.oneBedroom\_count = form.findChild(QLineEdit,"one\_bedroom\_lineEdit")
27. self.twoBedroom\_count = form.findChild(QLineEdit,"two\_bedroom\_lineEdit")
29. self.back\_btn = form.findChild(QPushButton, "back\_btn")
30. self.back\_btn.clicked.connect(self.back)
32. self.reserve\_btn = form.findChild(QPushButton, "reserve\_btn")
33. self.reserve\_btn.clicked.connect(self.reserve)
34. self.reserve\_btn.setEnabled(False)
36. self.daily = form.findChild(QRadioButton, "daily\_radioButton")
37. self.daily.clicked.connect(self.dailyChecked)
39. self.monthly = form.findChild(QRadioButton, "monthly\_radioButton")
40. self.monthly.clicked.connect(self.monthlyChecked)
41. self.monthly.setChecked(True)
43. self.fromDate = form.findChild(QDateEdit,"fromDate")
44. self.fromDate.setDate(QDate.currentDate())
45. self.fromDate.setMinimumDate(QDate.currentDate())
46. self.fromDate.dateChanged.connect(self.changeFrom)
48. self.toDate = form.findChild(QDateEdit,"toDate")
49. self.toDate.setDate(QDate.currentDate().addDays(1))
50. self.toDate.setMinimumDate(QDate.currentDate().addDays(1))
52. self.notDecide = form.findChild(QCheckBox,"not\_decide\_checkBox")
53. self.notDecide.stateChanged.connect(self.stateChanged)
55. self.roomType = form.findChild(QComboBox,"room\_type\_comboBox")
56. self.floor = form.findChild(QComboBox,"floor\_comboBox")
58. self.search\_btn = form.findChild(QPushButton,"search\_btn")
59. self.search\_btn.clicked.connect(self.search)

62. self.menu = menu
64. **def** stateChanged(self):
65. **if** self.notDecide.isChecked():
66. self.toDate.setEnabled(False)
67. **else**:
68. self.toDate.setEnabled(True)
70. **def** dailyChecked(self):
71. self.notDecide.setEnabled(False)
72. self.notDecide.setChecked(False)
74. **def** monthlyChecked(self):
75. self.notDecide.setEnabled(True)
77. **def** changeFrom(self, fromDay):
78. self.toDate.setMinimumDate(fromDay.addDays(1))
80. **def** search(self):

83. startDate = self.fromDate.date().toString("yyyy/MM/dd")
85. **if** self.notDecide.isChecked() == False:
86. outDate = self.toDate.date().toString("yyyy/MM/dd")
87. **else** :
88. outDate = "9999/12/31"
90. self.studio\_count.setText(str(self.db.countAvailableRoom("studio",startDate,outDate)))
91. self.oneBedroom\_count.setText(str(self.db.countAvailableRoom("1-bedroom",startDate,outDate)))
92. self.twoBedroom\_count.setText(str(self.db.countAvailableRoom("2-bedroom",startDate,outDate)))
94. **if** self.roomType.currentIndex()==1:
95. roomT = "studio"
96. **elif** self.roomType.currentIndex()==2:
97. roomT = "1-bedroom"
98. **elif** self.roomType.currentIndex()==3:
99. roomT = "2-bedroom"
100. **else** :
101. roomT = "All"
103. **if** self.floor.currentIndex() == 1:
104. f = "1"
105. **elif** self.floor.currentIndex() == 2:
106. f = "2"
107. **elif** self.floor.currentIndex() == 3:
108. f = "3"
109. **elif** self.floor.currentIndex() == 4:
110. f = "4"
111. **elif** self.floor.currentIndex() == 5:
112. f = "5"
113. **else** :
114. f = ["1","2","3","4","5"]
116. **if** roomT == "All":
117. **if** f!= ["1","2","3","4","5"]:
118. Avai = self.db.searchAllAvailableRoom(f,str(startDate),str(outDate))
119. **else** :
120. Avai = (self.db.searchAllAvailableRoom(1,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(2,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(3,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(4,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(5,str(startDate),str(outDate)))
121. **else** :
122. **if** f!= ["1","2","3","4","5"]:
123. Avai = self.db.searchAvailableRoom(f,roomT,str(startDate),str(outDate))
124. **else** :
125. Avai = (self.db.searchAvailableRoom(1,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(2,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(3,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(4,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(5,roomT,str(startDate),str(outDate)))

128. self.tableWidget.setRowCount(len(Avai))
129. self.tableWidget.setColumnCount(3)
130. self.layout.addWidget(self.tableWidget, 0,0)
131. row=0
132. **for** i **in** Avai:
133. self.tableWidget.setItem(row,0,QTableWidgetItem(str(i[0])))
134. self.tableWidget.setItem(row,1,QTableWidgetItem(str(i[1])))
135. self.tableWidget.setItem(row,2,QTableWidgetItem(str(i[2])))
136. row = row+1
138. **if** self.monthly.isChecked()==True:
139. restype = "monthly"
140. **else** :
141. restype = "daily"
143. self.searchedItem = [restype,startDate,outDate]
144. self.reserve\_btn.setEnabled(True)
146. **def** reserve(self):
147. **if** (len(self.tableWidget.selectedRanges())  >0):
148. row = self.tableWidget.currentRow()
149. selectedRoomno = self.tableWidget.item(row, 0).text()
150. selectedRoom = [self.tableWidget.item(row, 0).text(),self.tableWidget.item(row, 1).text(),self.tableWidget.item(row, 2).text()]
152. self.FillResInfo = FillResInfo(self,selectedRoom,self.searchedItem)
154. self.FillResInfo.show()
155. self.hide()
157. **def** back(self):
158. self.menu.show()
159. self.hide()
161. **def** paintEvent(self, e): #for adding an image
162. p = QPainter()
163. p.begin(self)
164. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
165. p.end()

FillResInfo\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** DatabaseSystem **import** \*
3. **from** Reservation\_ui **import** \*
5. **class** FillResInfo(QMainWindow):
6. **def** \_\_init\_\_(self,previous,selectedRoom,searchedItem):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiFillResInfo.ui", None)
12. self.setWindowTitle("SN Group Manager")
13. self.setWindowIcon(QIcon('images/icon.png'))
14. self.setCentralWidget(form)
15. self.setFixedSize(display\_width,display\_height)
17. self.db = DatabaseSys()
18. self.selectedRoom = selectedRoom
19. self.searchedItem = searchedItem
21. self.roomNo\_l = form.findChild(QLineEdit, "roomNo\_lineEdit")
22. self.floor\_l = form.findChild(QLineEdit, "floor\_lineEdit")
23. self.roomType\_l = form.findChild(QLineEdit, "roomType\_lineEdit")
24. self.checkin\_date\_l = form.findChild(QLineEdit, "checkinDate\_lineEdit")
25. self.checkout\_date\_l = form.findChild(QLineEdit, "checkoutDate\_lineEdit")
27. self.roomNo\_l.setText(self.selectedRoom[0]) #roomNo
28. self.floor\_l.setText(self.selectedRoom[1]) #floor
29. self.roomType\_l.setText(self.selectedRoom[2]) #roomType
30. self.checkin\_date\_l.setText(str(self.searchedItem[1]))
31. **if** self.searchedItem[2]!="9999/99/99":
32. self.checkout\_date\_l.setText(str(self.searchedItem[2]))
33. **else**:
34. self.checkout\_date\_l.setText("No checkout date")
36. #filled info
37. self.roomNo = self.selectedRoom[0]
38. self.floor = self.selectedRoom[1]
39. self.roomType = self.selectedRoom[2]
40. self.checkin\_date = str(self.searchedItem[1])
41. self.checkout\_date= str(self.searchedItem[2])
42. self.resType = self.searchedItem[0]
44. self.first\_name = form.findChild(QLineEdit, "first\_name\_lineEdit")
45. self.last\_name = form.findChild(QLineEdit, "last\_name\_lineEdit")
46. self.national\_id = form.findChild(QLineEdit, "national\_id\_lineEdit")
47. self.line = form.findChild(QLineEdit, "line\_lineEdit")
48. self.email = form.findChild(QLineEdit, "email\_lineEdit")
49. self.telephon\_no = form.findChild(QLineEdit, "telephon\_no\_lineEdit")
50. self.address1 = form.findChild(QLineEdit, "address1\_lineEdit")
51. self.address2 = form.findChild(QLineEdit, "address2\_lineEdit")
52. self.address3 = form.findChild(QLineEdit, "address3\_lineEdit")
54. #button
55. self.previous = previous
56. self.back\_btn = form.findChild(QPushButton, "back\_btn")
57. self.back\_btn.clicked.connect(self.back)
59. self.reserve\_btn = form.findChild(QPushButton, "reserve\_btn")
60. self.reserve\_btn.clicked.connect(self.reserve)
62. **def** reserve(self):
63. #filled info
64. firstname = self.first\_name.text()
65. lastname = self.last\_name.text()
66. tel = self.telephon\_no.text()
67. natID = self.national\_id.text()
68. line = self.line.text()
69. email = self.email.text()
70. addr = self.address1.text()+","+self.address2.text()+","+self.address3.text()
72. whitespaceFisrtname = firstname.count(" ")
73. whitespaceLastname = lastname.count(" ")
74. whitespaceTel = tel.count(" ")
76. #check if not empty
77. **if** (len(firstname)-whitespaceFisrtname)>0 **and** (len(lastname)-whitespaceLastname)>0 **and** (len(tel)-whitespaceTel) > 0:
78. self.db.reserve(self.roomNo,self.resType,firstname,lastname,natID,line,email,tel,addr,self.checkin\_date,self.checkout\_date)
79. self.reserve\_btn.setEnabled(False)
80. self.previous.reserve\_btn.setEnabled(False)
81. self.popup("residence reserved!")
82. **else** :
83. self.popup("please fill all \* values")
85. **def** back(self):
86. self.clearText()
87. self.previous.show()
88. self.hide()
90. **def** clearText(self):
91. self.first\_name.clear()
92. self.last\_name.clear()
93. self.national\_id.clear()
94. self.line.clear()
95. self.email.clear()
96. self.telephon\_no.clear()
97. self.address1.clear()
98. self.address2.clear()
99. self.address3.clear()
101. **def** popup(self,text):
102. dialog = QDialog(self)
103. layout = QVBoxLayout()
104. label = QLabel(text)
105. layout.addWidget(label)
106. close\_button = QPushButton("Close")
107. close\_button.clicked.connect(dialog.close)
108. layout.addWidget(close\_button)
109. dialog.setLayout(layout)
110. dialog.show()
112. **def** paintEvent(self, e): #for adding an image
113. p = QPainter()
114. p.begin(self)
115. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
116. p.end()

CancelReservation\_ui .py

1. **from** constant\_initialization **import** \*
2. **from** DatabaseSystem **import** \*
4. **class** CancelReservation(QMainWindow):
5. **def** \_\_init\_\_(self,menu):
6. QMainWindow.\_\_init\_\_(self, None)
8. loader = QUiLoader()
9. form = loader.load("UI/uiCancelReservation.ui", None)
11. self.setWindowTitle("SN Group Manager")
12. self.setWindowIcon(QIcon('images/icon.png'))
13. self.setCentralWidget(form)
14. self.setFixedSize(display\_width,display\_height)
15. self.menu = menu
17. self.first\_name = form.findChild(QLineEdit, "first\_name\_lineEdit")
18. self.last\_name = form.findChild(QLineEdit, "last\_name\_lineEdit")
20. self.search\_by\_name\_btn = form.findChild(QPushButton, "search\_by\_name\_btn")
21. self.search\_by\_name\_btn.clicked.connect(self.search\_by\_name)
23. self.tableWidget = form.findChild(QTableWidget, "tableWidget")
24. self.layout = QGridLayout()
26. self.cancel\_reservation\_btn = form.findChild(QPushButton, "cancel\_reservation\_btn")
27. self.cancel\_reservation\_btn.clicked.connect(self.cancel\_reservation)
29. self.back\_btn = form.findChild(QPushButton, "back\_btn")
30. self.back\_btn.clicked.connect(self.back)
32. self.db = DatabaseSys()
34. self.allreserve = self.db.allreserve()
36. self.tableWidget.setRowCount(len(self.allreserve))
37. self.tableWidget.setColumnCount(9)
38. self.layout.addWidget(self.tableWidget, 0,0)
40. self.showInfo(self.allreserve)
42. **def** search\_by\_name(self):
43. self.tableWidget.clearContents()
45. f\_name = self.first\_name.text()
46. l\_name = self.last\_name.text()
47. out = self.db.searchByName(f\_name, l\_name)
49. **if**(out **is** **not** "None"):
50. self.showInfo(out)
51. **else**:
52. self.popUP("Name not found!!!")
54. **def** cancel\_reservation(self):
55. **if** (len(self.tableWidget.selectedRanges())  >0):
56. row = self.tableWidget.currentRow()
57. roomNo = self.tableWidget.item(row, 2).text()
58. indate = self.tableWidget.item(row, 0).text()
59. outdate = self.tableWidget.item(row, 1).text()
60. self.db.cancelReserve(roomNo,indate,outdate)
61. self.tableWidget.removeRow(self.tableWidget.currentRow())
63. **def** showInfo(self,allReserve):
64. row = 0
65. **for** item **in** allReserve:
66. self.tableWidget.setItem(row, 0, QTableWidgetItem(item[9])) #checkinDate
67. self.tableWidget.setItem(row, 1, QTableWidgetItem(item[10])) #checkoutDate
68. self.tableWidget.setItem(row, 2, QTableWidgetItem(item[0])) #roomNumber
69. self.tableWidget.setItem(row, 3, QTableWidgetItem(str(self.db.readRoomType(item[0])))) #roomType
70. self.tableWidget.setItem(row, 4, QTableWidgetItem(item[2])) #firstname
71. self.tableWidget.setItem(row, 5, QTableWidgetItem(item[3])) #lastname
72. self.tableWidget.setItem(row, 6, QTableWidgetItem(item[1])) #residence type
73. self.tableWidget.setItem(row, 7, QTableWidgetItem(item[4])) #ID
74. self.tableWidget.setItem(row, 8, QTableWidgetItem(item[7])) #Tel No
75. row += 1
77. **def** popUP(self, txt):
78. dialog = QDialog(self)
79. layout = QVBoxLayout()
80. label = QLabel(str(txt))
81. layout.addWidget(label)
82. yes\_button1 = QPushButton("Yes")
83. yes\_button1.clicked.connect(dialog.close)
84. layout.addWidget(yes\_button1)
85. dialog.setLayout(layout)
86. dialog.show()

89. **def** back(self):
90. self.menu.show()
91. self.hide()
93. **def** paintEvent(self, e): #for adding an image
94. p = QPainter()
95. p.begin(self)
96. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
97. p.end()

CheckinWalkin\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** CheckinWalkin2\_ui **import** \*
3. **from** DatabaseSystem **import** \*
5. **class** CheckinWalkin(QMainWindow):
6. **def** \_\_init\_\_(self,menu):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiCheckinWalkin.ui", None)
12. self.menu = menu
14. self.db = DatabaseSys()
15. self.layout = QVBoxLayout()
16. self.tableWidget = form.findChild(QTableWidget, "tableWidget")
18. self.setWindowTitle("SN Group Manager")
19. self.setWindowIcon(QIcon('images/icon.png'))
20. self.setCentralWidget(form)
21. self.setFixedSize(display\_width,display\_height)
23. self.searchedItem = []
25. self.studio\_count = form.findChild(QLineEdit,"studio\_lineEdit")
26. self.oneBedroom\_count = form.findChild(QLineEdit,"one\_bedroom\_lineEdit")
27. self.twoBedroom\_count = form.findChild(QLineEdit,"two\_bedroom\_lineEdit")
29. self.back\_btn = form.findChild(QPushButton, "back\_btn")
30. self.back\_btn.clicked.connect(self.back)
32. self.reserve\_btn = form.findChild(QPushButton, "reserve\_btn")
33. self.reserve\_btn.clicked.connect(self.reserve)
34. self.reserve\_btn.setEnabled(False)
36. self.search\_btn = form.findChild(QPushButton, "search\_btn")
37. self.search\_btn.clicked.connect(self.search)
39. self.daily = form.findChild(QRadioButton, "daily\_radioButton")
40. self.daily.clicked.connect(self.dailyChecked)
42. self.monthly = form.findChild(QRadioButton, "monthly\_radioButton")
43. self.monthly.clicked.connect(self.monthlyChecked)
44. self.monthly.setChecked(True)
46. self.fromDate = form.findChild(QDateEdit,"fromDate")
47. self.fromDate.setDate(QDate.currentDate())
49. self.toDate = form.findChild(QDateEdit,"toDate")
50. self.toDate.setDate(QDate.currentDate().addDays(1))
52. self.fromDate = form.findChild(QDateEdit,"fromDate")
53. self.fromDate.setDate(QDate.currentDate())
54. self.fromDate.setMinimumDate(QDate.currentDate())
55. self.fromDate.dateChanged.connect(self.changeFrom)
57. self.toDate = form.findChild(QDateEdit,"toDate")
58. self.toDate.setDate(QDate.currentDate().addDays(1))
59. self.toDate.setMinimumDate(QDate.currentDate().addDays(1))
61. self.notDecide = form.findChild(QCheckBox,"not\_decide\_checkBox")
62. self.notDecide.stateChanged.connect(self.stateChanged)
64. self.roomType = form.findChild(QComboBox,"room\_type\_comboBox")
65. self.floor = form.findChild(QComboBox,"floor\_comboBox")


69. **def** stateChanged(self):
70. **if** self.notDecide.isChecked():
71. self.toDate.setEnabled(False)
72. **else**:
73. self.toDate.setEnabled(True)
75. **def** dailyChecked(self):
76. self.notDecide.setEnabled(False)
77. self.notDecide.setChecked(False)
79. **def** monthlyChecked(self):
80. self.notDecide.setEnabled(True)
82. **def** changeFrom(self, fromDay):
83. self.toDate.setMinimumDate(fromDay.addDays(1))
85. **def** search(self):
86. startDate = self.fromDate.date().toString("yyyy/MM/dd")
88. **if** self.notDecide.isChecked() == False:
89. outDate = self.toDate.date().toString("yyyy/MM/dd")
90. **else** :
91. outDate = "9999/12/31"
93. self.studio\_count.setText(str(self.db.countAvailableRoom("studio",startDate,outDate)))
94. self.oneBedroom\_count.setText(str(self.db.countAvailableRoom("1-bedroom",startDate,outDate)))
95. self.twoBedroom\_count.setText(str(self.db.countAvailableRoom("2-bedroom",startDate,outDate)))
97. **if** self.roomType.currentIndex()==1:
98. roomT = "studio"
99. **elif** self.roomType.currentIndex()==2:
100. roomT = "1-bedroom"
101. **elif** self.roomType.currentIndex()==3:
102. roomT = "2-bedroom"
103. **else** :
104. roomT = "All"
106. **if** self.floor.currentIndex() == 1:
107. f = "1"
108. **elif** self.floor.currentIndex() == 2:
109. f = "2"
110. **elif** self.floor.currentIndex() == 3:
111. f = "3"
112. **elif** self.floor.currentIndex() == 4:
113. f = "4"
114. **elif** self.floor.currentIndex() == 5:
115. f = "5"
116. **else** :
117. f = ["1","2","3","4","5"]
119. **if** roomT == "All":
120. **if** f!= ["1","2","3","4","5"]:
121. Avai = self.db.searchAllAvailableRoom(f,str(startDate),str(outDate))
122. **else** :
123. Avai = (self.db.searchAllAvailableRoom(1,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(2,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(3,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(4,str(startDate),str(outDate)))+(self.db.searchAllAvailableRoom(5,str(startDate),str(outDate)))
124. **else** :
125. **if** f!= ["1","2","3","4","5"]:
126. Avai = self.db.searchAvailableRoom(f,roomT,str(startDate),str(outDate))
127. **else** :
128. Avai = (self.db.searchAvailableRoom(1,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(2,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(3,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(4,roomT,str(startDate),str(outDate)))+(self.db.searchAvailableRoom(5,roomT,str(startDate),str(outDate)))

131. self.tableWidget.setRowCount(len(Avai))
132. self.tableWidget.setColumnCount(3)
133. self.layout.addWidget(self.tableWidget, 0,0)
134. row=0
135. **for** i **in** Avai:
136. self.tableWidget.setItem(row,0,QTableWidgetItem(str(i[0])))
137. self.tableWidget.setItem(row,1,QTableWidgetItem(str(i[1])))
138. self.tableWidget.setItem(row,2,QTableWidgetItem(str(i[2])))
139. row = row+1
141. **if** self.monthly.isChecked()==True:
142. restype = "monthly"
143. **else** :
144. restype = "daily"
146. self.searchedItem = [restype,startDate,outDate]
147. self.reserve\_btn.setEnabled(True)
149. **def** reserve(self):
150. **if** (len(self.tableWidget.selectedRanges())  >0):
151. row = self.tableWidget.currentRow()
152. selectedRoomno = self.tableWidget.item(row, 0).text()
153. selectedRoom = [self.tableWidget.item(row, 0).text(),self.tableWidget.item(row, 1).text(),self.tableWidget.item(row, 2).text()]
154. self.CheckinWalkin2 = CheckinWalkin2(self,selectedRoom,self.searchedItem)
155. self.CheckinWalkin2.show()
156. self.hide()
158. **def** back(self):
159. self.menu.show()
160. self.hide()
162. **def** paintEvent(self, e): #for adding an image
163. p = QPainter()
164. p.begin(self)
165. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
166. p.end()

CheckinWalkin2\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** DatabaseSystem **import** \*
4. **class** CheckinWalkin2(QMainWindow):
5. **def** \_\_init\_\_(self,chkin1,selectedRoom,searchedItem):
6. QMainWindow.\_\_init\_\_(self, None)
8. loader = QUiLoader()
9. form = loader.load("UI/uiCheckinWalkin2.ui", None)
11. self.db = DatabaseSys()
12. self.selectedRoom = selectedRoom
13. self.searchedItem = searchedItem
15. self.roomNo\_l = form.findChild(QLineEdit, "roomNo\_lineEdit")
16. self.floor\_l = form.findChild(QLineEdit, "floor\_lineEdit")
17. self.roomType\_l = form.findChild(QLineEdit, "roomType\_lineEdit")
18. self.checkin\_date\_l = form.findChild(QLineEdit, "checkinDate\_lineEdit")
19. self.checkout\_date\_l = form.findChild(QLineEdit, "checkoutDate\_lineEdit")
21. self.roomNo\_l.setText(self.selectedRoom[0]) #roomNo
22. self.floor\_l.setText(self.selectedRoom[1]) #floor
23. self.roomType\_l.setText(self.selectedRoom[2]) #roomType
24. self.checkin\_date\_l.setText(str(self.searchedItem[1]))
25. **if** self.searchedItem[2]!="9999/13/31":
26. self.checkout\_date\_l.setText(str(self.searchedItem[2]))
27. **else**:
28. self.checkout\_date\_l.setText("No checkout date")
30. #filled info
31. self.roomNo = self.selectedRoom[0]
32. self.floor = self.selectedRoom[1]
33. self.roomType = self.selectedRoom[2]
34. self.checkin\_date = str(self.searchedItem[1])
35. self.checkout\_date= str(self.searchedItem[2])
36. self.resType = self.searchedItem[0]
38. self.first\_name = form.findChild(QLineEdit, "first\_name\_lineEdit")
39. self.last\_name = form.findChild(QLineEdit, "last\_name\_lineEdit")
40. self.national\_id = form.findChild(QLineEdit, "national\_id\_lineEdit")
41. self.line = form.findChild(QLineEdit, "line\_lineEdit")
42. self.email = form.findChild(QLineEdit, "email\_lineEdit")
43. self.telephon\_no = form.findChild(QLineEdit, "telephon\_no\_lineEdit")
44. self.address1 = form.findChild(QLineEdit, "address1\_lineEdit")
45. self.address2 = form.findChild(QLineEdit, "address2\_lineEdit")
46. self.address3 = form.findChild(QLineEdit, "address3\_lineEdit")

49. #GUI
50. self.previous = chkin1
52. self.setWindowTitle("SN Group Manager")
53. self.setWindowIcon(QIcon('images/icon.png'))
54. self.setCentralWidget(form)
55. self.setFixedSize(display\_width,display\_height)
57. self.check\_in\_btn = form.findChild(QPushButton, "check\_in\_btn")
58. self.check\_in\_btn.clicked.connect(self.check\_in)
60. self.first\_name = form.findChild(QLineEdit, "first\_name\_lineEdit")
61. self.last\_name = form.findChild(QLineEdit, "last\_name\_lineEdit")
62. self.national\_id = form.findChild(QLineEdit, "national\_id\_lineEdit")
63. self.line = form.findChild(QLineEdit, "line\_lineEdit")
64. self.email = form.findChild(QLineEdit, "email\_lineEdit")
65. self.telephon\_no = form.findChild(QLineEdit, "telephon\_no\_lineEdit")
66. self.address1 = form.findChild(QLineEdit, "address1\_lineEdit")
67. self.address2 = form.findChild(QLineEdit, "address2\_lineEdit")
68. self.address3 = form.findChild(QLineEdit, "address3\_lineEdit")
70. #button
71. self.back\_btn = form.findChild(QPushButton, "back\_btn")
72. self.back\_btn.clicked.connect(self.back)

75. **def** check\_in(self):
76. #filled info
77. firstname = self.first\_name.text()
78. lastname = self.last\_name.text()
79. tel = self.telephon\_no.text()
80. natID = self.national\_id.text()
81. line = self.line.text()
82. email = self.email.text()
83. addr = self.address1.text()+","+self.address2.text()+","+self.address3.text()
85. whitespaceFisrtname = firstname.count(" ")
86. whitespaceLastname = lastname.count(" ")
87. whitespaceTel = tel.count(" ")
88. whitespacNatID = natID.count(" ")
89. whitespaceAddr = addr.count(" ")
91. #check if not empty
92. **if** (len(firstname)-whitespaceFisrtname)>0 **and** (len(lastname)-whitespaceLastname)>0 **and** (len(tel)-whitespaceTel) > 0 **and** (len(natID)-whitespaceFisrtname)>0 **and** (len(addr)-whitespaceAddr)>2:
93. self.db.checkInWalk(self.roomNo,self.resType,firstname,lastname,natID,line,email,tel,addr,self.checkin\_date,self.checkout\_date)
94. self.popup("residence checkd in!")
95. self.check\_in\_btn.setEnabled(False)
96. self.previous.reserve\_btn.setEnabled(False)
97. **else** :
98. self.popup("please fill all \* values")
100. **def** back(self):
101. self.clearText()
102. self.previous.show()
103. self.hide()
105. **def** clearText(self):
106. self.first\_name.clear()
107. self.last\_name.clear()
108. self.national\_id.clear()
109. self.line.clear()
110. self.email.clear()
111. self.telephon\_no.clear()
112. self.address1.clear()
113. self.address2.clear()
114. self.address3.clear()
116. **def** popup(self,text):
117. dialog = QDialog(self)
118. layout = QVBoxLayout()
119. label = QLabel(text)
120. layout.addWidget(label)
121. close\_button = QPushButton("Close")
122. close\_button.clicked.connect(dialog.close)
123. layout.addWidget(close\_button)
124. dialog.setLayout(layout)
125. dialog.show()
127. **def** paintEvent(self, e): #for adding an image
128. p = QPainter()
129. p.begin(self)
130. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
131. p.end()

CheckinReserved\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** DatabaseSystem **import** \*
3. **import** datetime
5. **class** CheckinReserved(QMainWindow):
6. **def** \_\_init\_\_(self,menu):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiCheckinReserved.ui", None)
12. self.db = DatabaseSys()
13. self.menu = menu
15. self.setWindowTitle("SN Group Manager")
16. self.setWindowIcon(QIcon('images/icon.png'))
17. self.setCentralWidget(form)
18. self.setFixedSize(display\_width,display\_height)
20. self.roomtype\_l = form.findChild(QLineEdit,"roomType\_lineEdit")
21. self.checkin\_date\_de = form.findChild(QDateEdit, "checkin\_dateEdit")
22. self.checkout\_date\_de = form.findChild(QDateEdit, "checkout\_dateEdit")
23. self.checkin\_date\_de.setDate(QDate.currentDate())
24. self.checkout\_date\_de.setDate(QDate.currentDate().addDays(1))
26. self.search\_btn = form.findChild(QPushButton, "search\_btn")
27. self.search\_btn.clicked.connect(self.search)
29. self.search\_name\_lineEdit = form.findChild(QLineEdit, "search\_by\_name\_lineEdit")
30. self.search\_lastname\_lineEdit = form.findChild(QLineEdit, "search\_by\_lastname\_lineEdit")
32. self.check\_in\_btn = form.findChild(QPushButton, "check\_in\_btn")
33. self.check\_in\_btn.clicked.connect(self.check\_in)
34. self.check\_in\_btn.setEnabled(False)
36. self.first\_name = form.findChild(QLineEdit, "first\_name\_lineEdit")
37. self.last\_name = form.findChild(QLineEdit, "last\_name\_lineEdit")
38. self.national\_id = form.findChild(QLineEdit, "national\_id\_lineEdit")
39. self.line = form.findChild(QLineEdit, "line\_lineEdit")
40. self.email = form.findChild(QLineEdit, "email\_lineEdit")
41. self.telephon\_no = form.findChild(QLineEdit, "telephon\_no\_lineEdit")
42. self.address1 = form.findChild(QLineEdit, "address1\_lineEdit")
43. self.address2 = form.findChild(QLineEdit, "address2\_lineEdit")
44. self.address3 = form.findChild(QLineEdit, "address3\_lineEdit")
46. currentDate = str(datetime.date.today())
47. currentDate = currentDate.replace("-","/")
49. self.tableWidget = form.findChild(QTableWidget, "tableWidget")
51. self.todayReservedRoom = self.db.checkReserveByCheckInDay(currentDate)
52. self.showInfo(self.todayReservedRoom)
54. self.back\_btn = form.findChild(QPushButton, "back\_btn")
55. self.back\_btn.clicked.connect(self.back)
57. #tablewidget
58. self.tableWidget.cellClicked.connect(self.itemisSelected)
60. **def** showInfo(self,item):
61. self.tableWidget.clearContents()
63. self.layout = QVBoxLayout()
65. self.tableWidget.setRowCount(len(item))
66. self.tableWidget.setColumnCount(3)
67. self.layout.addWidget(self.tableWidget, 0,0)
68. row=0
69. **for** i **in** item:
70. self.tableWidget.setItem(row,0,QTableWidgetItem(str(i[2])))
71. self.tableWidget.setItem(row,1,QTableWidgetItem(str(i[1])))
72. self.tableWidget.setItem(row,2,QTableWidgetItem(str(i[0])))
73. row = row+1
75. **def** itemisSelected(self):
76. self.check\_in\_btn.setEnabled(True)
77. row = self.tableWidget.currentRow()
78. selectedRoom = [self.tableWidget.item(row, 0).text(),self.tableWidget.item(row, 1).text(),self.tableWidget.item(row, 2).text()]
79. self.roomNo = str(self.tableWidget.item(row, 2).text())
80. res = self.db.checkReservedResidentInfoByRoomNo(self.roomNo)
81. self.resType = self.tableWidget.item(row, 1).text()
82. name = res[2]
83. lastname = res[3]
84. natID = res[4]
85. line = res[5]
86. email = res[6]
87. tel = res[7]
88. addr = res[8]
89. checkinD = res[9]
90. checkoutD = res[10]
92. checkinD = checkinD.split('/')
93. checkinD = datetime.datetime(int(checkinD[0]),int(checkinD[1]),int(checkinD[2]))
94. checkoutD = checkoutD.split('/')
95. checkoutD = datetime.datetime(int(checkoutD[0]),int(checkoutD[1]),int(checkoutD[2]))
97. addr = addr.split(",")
98. self.first\_name.setText(name)
99. self.last\_name.setText(lastname)
100. self.national\_id.setText(natID)
101. self.line.setText(line)
102. self.email.setText(email)
103. self.telephon\_no.setText(tel)
104. self.address1.setText(addr[0])
105. self.address2.setText(addr[1])
106. self.address3.setText(addr[2])
108. self.roomtype\_l.setText(self.db.readRoomType(self.roomNo))
109. self.checkin\_date\_de.setDate(checkinD)
110. self.checkout\_date\_de.setDate(checkoutD)

113. **def** search(self):
114. self.tableWidget.clearContents()
115. allReserve = self.db.searchByName(self.search\_name\_lineEdit.text(),self.search\_lastname\_lineEdit.text())
117. **if**(allReserve **is** **not** None):
118. self.showInfo(allReserve)
119. **else**:
120. self.popUP("Name not found!!!")
122. **def** check\_in(self):
123. #filled info
125. firstname = self.first\_name.text()
126. lastname = self.last\_name.text()
127. tel = self.telephon\_no.text()
128. natID = self.national\_id.text()
129. line = self.line.text()
130. email = self.email.text()
131. addr = self.address1.text()+","+self.address2.text()+","+self.address3.text()
132. chkin = str(self.checkin\_date\_de.date())
133. chkout = str(self.checkout\_date\_de.date())
134. **print**(chkin)
136. whitespaceFisrtname = firstname.count(" ")
137. whitespaceLastname = lastname.count(" ")
138. whitespaceTel = tel.count(" ")
139. whitespacNatID = natID.count(" ")
140. whitespaceAddr = addr.count(" ")
142. #check if not empty
143. **if** (len(firstname)-whitespaceFisrtname)>0 **and** (len(lastname)-whitespaceLastname)>0 **and** (len(tel)-whitespaceTel) > 0 **and** (len(natID)-whitespaceFisrtname)>0 **and** (len(addr)-whitespaceAddr)>2:
144. self.db.checkInReserve(self.roomNo,self.resType,firstname,lastname,natID,line,email,tel,addr,chkin,chkout)
145. self.popup("residence checkd in!")
146. self.tableWidget.removeRow(self.tableWidget.currentRow())
147. self.check\_in\_btn.setEnabled(False)
148. **else** :
149. self.popup("please fill all \* values")


153. **def** back(self):
154. self.clearText()
155. self.menu.show()
156. self.hide()
158. **def** popup(self,text):
159. dialog = QDialog(self)
160. layout = QVBoxLayout()
161. label = QLabel(text)
162. layout.addWidget(label)
163. close\_button = QPushButton("Close")
164. close\_button.clicked.connect(dialog.close)
165. layout.addWidget(close\_button)
166. dialog.setLayout(layout)
167. dialog.show()

170. **def** clearText(self):
171. self.first\_name.clear()
172. self.last\_name.clear()
173. self.national\_id.clear()
174. self.line.clear()
175. self.email.clear()
176. self.telephon\_no.clear()
177. self.address1.clear()
178. self.address2.clear()
179. self.address3.clear()
181. **def** paintEvent(self, e): #for adding an image
182. p = QPainter()
183. p.begin(self)
184. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
185. p.end()

Checkout\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** DatabaseSystem **import** \*
4. **class** Checkout(QMainWindow):
5. **def** \_\_init\_\_(self,menu):
6. QMainWindow.\_\_init\_\_(self, None)
8. loader = QUiLoader()
9. form = loader.load("UI/uiCheckout.ui", None)
11. self.setWindowTitle("SN Group Manager")
12. self.setWindowIcon(QIcon('images/icon.png'))
13. self.setCentralWidget(form)
14. self.setFixedSize(display\_width,display\_height)
15. self.menu = menu
17. self.building\_sw = form.findChild(QStackedWidget, "Building\_sw")
19. self.back\_btn = form.findChild(QPushButton, "back\_btn")
20. self.back\_btn.clicked.connect(self.back)
22. self.checkout\_btn = form.findChild(QPushButton, "checkout\_btn")
23. self.checkout\_btn.clicked.connect(self.checkout)
24. self.checkout\_btn.setEnabled(False)
26. self.F1\_btn = form.findChild(QPushButton, "F1\_btn")
27. self.F1\_btn.clicked.connect(**lambda**: self.floor(0))
29. self.F2\_btn = form.findChild(QPushButton, "F2\_btn")
30. self.F2\_btn.clicked.connect(**lambda**: self.floor(1))
32. self.F3\_btn = form.findChild(QPushButton, "F3\_btn")
33. self.F3\_btn.clicked.connect(**lambda**: self.floor(2))
35. self.F4\_btn = form.findChild(QPushButton, "F4\_btn")
36. self.F4\_btn.clicked.connect(**lambda**: self.floor(3))
38. self.F5\_btn = form.findChild(QPushButton, "F5\_btn")
39. self.F5\_btn.clicked.connect(**lambda**: self.floor(4))
41. self.db = DatabaseSys()
43. #Room
44. self.room101 = form.findChild(QPushButton, "pb101")
45. self.room101.clicked.connect(**lambda**: self.showInfo(101))
46. self.btnColor(self.room101, "101")
47. self.room102 = form.findChild(QPushButton, "pb102")
48. self.room102.clicked.connect(**lambda**: self.showInfo(102))
49. self.btnColor(self.room102, "102")
50. self.room103 = form.findChild(QPushButton, "pb103")
51. self.room103.clicked.connect(**lambda**: self.showInfo(103))
52. self.btnColor(self.room103, "103")
53. self.room104 = form.findChild(QPushButton, "pb104")
54. self.room104.clicked.connect(**lambda**: self.showInfo(104))
55. self.btnColor(self.room104, "104")
56. self.room105 = form.findChild(QPushButton, "pb105")
57. self.room105.clicked.connect(**lambda**: self.showInfo(105))
58. self.btnColor(self.room105, "105")
59. self.room106 = form.findChild(QPushButton, "pb106")
60. self.room106.clicked.connect(**lambda**: self.showInfo(106))
61. self.btnColor(self.room106, "106")
62. self.room107 = form.findChild(QPushButton, "pb107")
63. self.room107.clicked.connect(**lambda**: self.showInfo(107))
64. self.btnColor(self.room107, "107")
65. self.room108 = form.findChild(QPushButton, "pb108")
66. self.room108.clicked.connect(**lambda**: self.showInfo(108))
67. self.btnColor(self.room108, "108")
68. self.room109 = form.findChild(QPushButton, "pb109")
69. self.room109.clicked.connect(**lambda**: self.showInfo(109))
70. self.btnColor(self.room109, "109")
71. self.room110 = form.findChild(QPushButton, "pb110")
72. self.room110.clicked.connect(**lambda**: self.showInfo(110))
73. self.btnColor(self.room110, "110")
74. self.room111 = form.findChild(QPushButton, "pb111")
75. self.room111.clicked.connect(**lambda**: self.showInfo(111))
76. self.btnColor(self.room111, "111")
77. self.room112 = form.findChild(QPushButton, "pb112")
78. self.room112.clicked.connect(**lambda**: self.showInfo(112))
79. self.btnColor(self.room112, "112")
81. self.room201 = form.findChild(QPushButton, "pb201")
82. self.room201.clicked.connect(**lambda**: self.showInfo(201))
83. self.btnColor(self.room201, "201")
84. self.room202 = form.findChild(QPushButton, "pb202")
85. self.room202.clicked.connect(**lambda**: self.showInfo(202))
86. self.btnColor(self.room202, "202")
87. self.room203 = form.findChild(QPushButton, "pb203")
88. self.room203.clicked.connect(**lambda**: self.showInfo(203))
89. self.btnColor(self.room203, "203")
90. self.room204 = form.findChild(QPushButton, "pb204")
91. self.room204.clicked.connect(**lambda**: self.showInfo(204))
92. self.btnColor(self.room204, "204")
93. self.room205 = form.findChild(QPushButton, "pb205")
94. self.room205.clicked.connect(**lambda**: self.showInfo(205))
95. self.btnColor(self.room205, "205")
96. self.room206 = form.findChild(QPushButton, "pb206")
97. self.room206.clicked.connect(**lambda**: self.showInfo(206))
98. self.btnColor(self.room206, "206")
99. self.room207 = form.findChild(QPushButton, "pb207")
100. self.room207.clicked.connect(**lambda**: self.showInfo(207))
101. self.btnColor(self.room207, "207")
102. self.room208 = form.findChild(QPushButton, "pb208")
103. self.room208.clicked.connect(**lambda**: self.showInfo(208))
104. self.btnColor(self.room208, "208")
105. self.room209 = form.findChild(QPushButton, "pb209")
106. self.room209.clicked.connect(**lambda**: self.showInfo(209))
107. self.btnColor(self.room209, "209")
108. self.room210 = form.findChild(QPushButton, "pb210")
109. self.room210.clicked.connect(**lambda**: self.showInfo(210))
110. self.btnColor(self.room210, "210")
111. self.room211 = form.findChild(QPushButton, "pb211")
112. self.room211.clicked.connect(**lambda**: self.showInfo(211))
113. self.btnColor(self.room211, "211")
114. self.room212 = form.findChild(QPushButton, "pb212")
115. self.room212.clicked.connect(**lambda**: self.showInfo(212))
116. self.btnColor(self.room212, "212")
117. self.room213 = form.findChild(QPushButton, "pb213")
118. self.room213.clicked.connect(**lambda**: self.showInfo(213))
119. self.btnColor(self.room213, "213")
120. self.room214 = form.findChild(QPushButton, "pb214")
121. self.room214.clicked.connect(**lambda**: self.showInfo(214))
122. self.btnColor(self.room214, "214")
123. self.room215 = form.findChild(QPushButton, "pb215")
124. self.room215.clicked.connect(**lambda**: self.showInfo(215))
125. self.btnColor(self.room215, "215")
126. self.room216 = form.findChild(QPushButton, "pb216")
127. self.room216.clicked.connect(**lambda**: self.showInfo(216))
128. self.btnColor(self.room216, "216")
129. self.room217 = form.findChild(QPushButton, "pb217")
130. self.room217.clicked.connect(**lambda**: self.showInfo(217))
131. self.btnColor(self.room217, "217")
133. self.room301 = form.findChild(QPushButton, "pb301")
134. self.room301.clicked.connect(**lambda**: self.showInfo(301))
135. self.btnColor(self.room301, "301")
136. self.room302 = form.findChild(QPushButton, "pb302")
137. self.room302.clicked.connect(**lambda**: self.showInfo(302))
138. self.btnColor(self.room302, "302")
139. self.room303 = form.findChild(QPushButton, "pb303")
140. self.room303.clicked.connect(**lambda**: self.showInfo(303))
141. self.btnColor(self.room303, "303")
142. self.room304 = form.findChild(QPushButton, "pb304")
143. self.room304.clicked.connect(**lambda**: self.showInfo(304))
144. self.btnColor(self.room304, "304")
145. self.room305 = form.findChild(QPushButton, "pb305")
146. self.room305.clicked.connect(**lambda**: self.showInfo(305))
147. self.btnColor(self.room305, "305")
148. self.room306 = form.findChild(QPushButton, "pb306")
149. self.room306.clicked.connect(**lambda**: self.showInfo(306))
150. self.btnColor(self.room306, "306")
151. self.room307 = form.findChild(QPushButton, "pb307")
152. self.room307.clicked.connect(**lambda**: self.showInfo(307))
153. self.btnColor(self.room307, "307")
154. self.room308 = form.findChild(QPushButton, "pb308")
155. self.room308.clicked.connect(**lambda**: self.showInfo(308))
156. self.btnColor(self.room308, "308")
157. self.room309 = form.findChild(QPushButton, "pb309")
158. self.room309.clicked.connect(**lambda**: self.showInfo(309))
159. self.btnColor(self.room309, "309")
160. self.room310 = form.findChild(QPushButton, "pb310")
161. self.room310.clicked.connect(**lambda**: self.showInfo(310))
162. self.btnColor(self.room310, "310")
163. self.room311 = form.findChild(QPushButton, "pb311")
164. self.room311.clicked.connect(**lambda**: self.showInfo(311))
165. self.btnColor(self.room311, "311")
166. self.room312 = form.findChild(QPushButton, "pb312")
167. self.room312.clicked.connect(**lambda**: self.showInfo(312))
168. self.btnColor(self.room312, "312")
169. self.room313 = form.findChild(QPushButton, "pb313")
170. self.room313.clicked.connect(**lambda**: self.showInfo(313))
171. self.btnColor(self.room313, "313")
172. self.room314 = form.findChild(QPushButton, "pb314")
173. self.room314.clicked.connect(**lambda**: self.showInfo(314))
174. self.btnColor(self.room314, "314")
175. self.room315 = form.findChild(QPushButton, "pb315")
176. self.room315.clicked.connect(**lambda**: self.showInfo(315))
177. self.btnColor(self.room315, "315")
178. self.room316 = form.findChild(QPushButton, "pb316")
179. self.room316.clicked.connect(**lambda**: self.showInfo(316))
180. self.btnColor(self.room316, "316")
181. self.room317 = form.findChild(QPushButton, "pb317")
182. self.room317.clicked.connect(**lambda**: self.showInfo(317))
183. self.btnColor(self.room317, "317")
185. self.room401 = form.findChild(QPushButton, "pb401")
186. self.room401.clicked.connect(**lambda**: self.showInfo(401))
187. self.btnColor(self.room401, "401")
188. self.room402 = form.findChild(QPushButton, "pb402")
189. self.room402.clicked.connect(**lambda**: self.showInfo(402))
190. self.btnColor(self.room402, "402")
191. self.room403 = form.findChild(QPushButton, "pb403")
192. self.room403.clicked.connect(**lambda**: self.showInfo(403))
193. self.btnColor(self.room403, "403")
194. self.room404 = form.findChild(QPushButton, "pb404")
195. self.room404.clicked.connect(**lambda**: self.showInfo(404))
196. self.btnColor(self.room404, "404")
197. self.room405 = form.findChild(QPushButton, "pb405")
198. self.room405.clicked.connect(**lambda**: self.showInfo(405))
199. self.btnColor(self.room405, "405")
200. self.room406 = form.findChild(QPushButton, "pb406")
201. self.room406.clicked.connect(**lambda**: self.showInfo(406))
202. self.btnColor(self.room406, "406")
203. self.room407 = form.findChild(QPushButton, "pb407")
204. self.room407.clicked.connect(**lambda**: self.showInfo(407))
205. self.btnColor(self.room407, "407")
206. self.room408 = form.findChild(QPushButton, "pb408")
207. self.room408.clicked.connect(**lambda**: self.showInfo(408))
208. self.btnColor(self.room408, "408")
209. self.room409 = form.findChild(QPushButton, "pb409")
210. self.room409.clicked.connect(**lambda**: self.showInfo(409))
211. self.btnColor(self.room409, "409")
212. self.room410 = form.findChild(QPushButton, "pb410")
213. self.room410.clicked.connect(**lambda**: self.showInfo(410))
214. self.btnColor(self.room410, "410")
215. self.room411 = form.findChild(QPushButton, "pb411")
216. self.room411.clicked.connect(**lambda**: self.showInfo(411))
217. self.btnColor(self.room411, "411")
218. self.room412 = form.findChild(QPushButton, "pb412")
219. self.room412.clicked.connect(**lambda**: self.showInfo(412))
220. self.btnColor(self.room412, "412")
221. self.room413 = form.findChild(QPushButton, "pb413")
222. self.room413.clicked.connect(**lambda**: self.showInfo(413))
223. self.btnColor(self.room413, "413")
224. self.room414 = form.findChild(QPushButton, "pb414")
225. self.room414.clicked.connect(**lambda**: self.showInfo(414))
226. self.btnColor(self.room414, "414")
227. self.room415 = form.findChild(QPushButton, "pb415")
228. self.room415.clicked.connect(**lambda**: self.showInfo(415))
229. self.btnColor(self.room415, "415")
230. self.room416 = form.findChild(QPushButton, "pb416")
231. self.room416.clicked.connect(**lambda**: self.showInfo(416))
232. self.btnColor(self.room416, "416")
233. self.room417 = form.findChild(QPushButton, "pb417")
234. self.room417.clicked.connect(**lambda**: self.showInfo(417))
235. self.btnColor(self.room417, "417")
237. self.room501 = form.findChild(QPushButton, "pb501")
238. self.room501.clicked.connect(**lambda**: self.showInfo(501))
239. self.btnColor(self.room501, "501")
240. self.room502 = form.findChild(QPushButton, "pb502")
241. self.room502.clicked.connect(**lambda**: self.showInfo(502))
242. self.btnColor(self.room502, "502")
243. self.room503 = form.findChild(QPushButton, "pb503")
244. self.room503.clicked.connect(**lambda**: self.showInfo(503))
245. self.btnColor(self.room503, "503")
246. self.room504 = form.findChild(QPushButton, "pb504")
247. self.room504.clicked.connect(**lambda**: self.showInfo(504))
248. self.btnColor(self.room504, "504")
249. self.room505 = form.findChild(QPushButton, "pb505")
250. self.room505.clicked.connect(**lambda**: self.showInfo(505))
251. self.btnColor(self.room505, "505")
252. self.room506 = form.findChild(QPushButton, "pb506")
253. self.room506.clicked.connect(**lambda**: self.showInfo(506))
254. self.btnColor(self.room506, "506")
255. self.room507 = form.findChild(QPushButton, "pb507")
256. self.room507.clicked.connect(**lambda**: self.showInfo(507))
257. self.btnColor(self.room507, "507")
258. self.room508 = form.findChild(QPushButton, "pb508")
259. self.room508.clicked.connect(**lambda**: self.showInfo(508))
260. self.btnColor(self.room508, "508")
261. self.room509 = form.findChild(QPushButton, "pb509")
262. self.room509.clicked.connect(**lambda**: self.showInfo(509))
263. self.btnColor(self.room509, "509")
264. self.room510 = form.findChild(QPushButton, "pb510")
265. self.room510.clicked.connect(**lambda**: self.showInfo(510))
266. self.btnColor(self.room510, "510")
267. self.room511 = form.findChild(QPushButton, "pb511")
268. self.room511.clicked.connect(**lambda**: self.showInfo(511))
269. self.btnColor(self.room511, "511")
270. self.room512 = form.findChild(QPushButton, "pb512")
271. self.room512.clicked.connect(**lambda**: self.showInfo(512))
272. self.btnColor(self.room512, "512")
273. self.room513 = form.findChild(QPushButton, "pb513")
274. self.room513.clicked.connect(**lambda**: self.showInfo(513))
275. self.btnColor(self.room513, "513")
276. self.room514 = form.findChild(QPushButton, "pb514")
277. self.room514.clicked.connect(**lambda**: self.showInfo(514))
278. self.btnColor(self.room514, "514")
279. self.room515 = form.findChild(QPushButton, "pb515")
280. self.room515.clicked.connect(**lambda**: self.showInfo(515))
281. self.btnColor(self.room515, "515")
283. #listWidget
284. self.roomInfo\_listWidget = form.findChild(QListWidget, "roomInfo\_listWidget")
285. self.resInfo\_listWidget = form.findChild(QListWidget, "resInfo\_listWidget")
287. #Layout
288. self.room\_layout = QVBoxLayout()
289. self.res\_layout = QVBoxLayout()
291. self.confirm\_dialog = QDialog(self)
293. **def** btnColor(self, btn, rN):
294. roomNo = rN
295. **if**(self.db.checkStatusIftheroomisCheckedIn(str(roomNo))==True) **and** (self.db.checkExceedCheckoutDate(str(roomNo)) == False):
296. self.setButtonColor(btn, 'orange')
297. **elif**(self.db.checkStatusIftheroomisCheckedIn(str(roomNo))==True) **and** (self.db.checkExceedCheckoutDate(str(roomNo)) == True):
298. self.setButtonColor(btn, 'red')
299. **else**:
300. self.setButtonColor(btn, 'green')
302. **def** setButtonColor(self, btn, color):
303. btn.setFlat(True)
304. palette = QPalette(btn.palette())
305. palette.setColor(QPalette.Button, QColor(color))
306. btn.setAutoFillBackground(True)
307. btn.setPalette(palette)
308. btn.update()
310. **def** showInfo(self, roomNo):
311. self.roomNo = roomNo
312. self.checkout\_btn.setEnabled(True)
314. #room\_layout
315. self.clearInfo(self.room\_layout)
316. self.room\_layout.addWidget(QLabel(str("-------")+str(roomNo)+str("-------\n") + self.db.readRoomInfo(roomNo)))
317. self.roomInfo\_listWidget.setLayout(self.room\_layout)
318. self.roomInfo\_listWidget.show()
320. #res\_layout
321. self.clearInfo(self.res\_layout)
322. self.res\_layout.addWidget(QLabel(str("-------")+str(roomNo)+str("-------\n") + self.db.readResidentInfo(roomNo)))
323. self.resInfo\_listWidget.setLayout(self.res\_layout)
324. self.resInfo\_listWidget.show()

327. **def** clearInfo(self, layout):
328. **if** layout **is** **not** None:
329. **while** layout.count():
330. item = layout.takeAt(0)
331. widget = item.widget()
332. **if** widget **is** **not** None:
333. widget.deleteLater()
334. **else**:
335. self.clearInfo(item.layout)
337. **def** floor(self, idx):
338. self.building\_sw.setCurrentIndex(idx)
340. **def** back(self):
341. self.clearInfo(self.room\_layout)
342. self.clearInfo(self.res\_layout)
343. self.checkout\_btn.setEnabled(False)
344. self.menu.show()
345. self.hide()
347. **def** checkout(self):
348. **if**(self.db.IsAvailable(self.roomNo)):
349. self.popUPRoomAvailable()
350. **else**:
351. self.popUPRoomNotAvailable()
353. **def** popUPRoomNotAvailable(self):
354. layout = QVBoxLayout()
355. label = QLabel("Are you sure")
356. layout.addWidget(label)
358. no\_button = QPushButton("No")
359. no\_button.clicked.connect(**lambda**: self.confirm("No"))
361. yes\_button = QPushButton("Yes")
362. yes\_button.clicked.connect(**lambda**: self.confirm("Yes"))
364. layout.addWidget(no\_button)
365. layout.addWidget(yes\_button)
366. self.confirm\_dialog.setLayout(layout)
367. self.confirm\_dialog.show()
369. **def** confirm(self, ans):
370. **if**(ans == "Yes"):
371. self.db.checkOut(self.roomNo)
372. self.checkout\_btn.setEnabled(False)
373. self.confirm\_dialog.close()
374. **elif**(ans == "No"):
375. self.confirm\_dialog.close()
376. **return**
378. **def** popUPRoomAvailable(self):
379. dialog = QDialog(self)
380. layout = QVBoxLayout()
381. label = QLabel("Room availabel, you can not check out!")
382. layout.addWidget(label)
383. yes\_button1 = QPushButton("Yes")
384. yes\_button1.clicked.connect(dialog.close)
385. layout.addWidget(yes\_button1)
386. dialog.setLayout(layout)
387. dialog.show()
389. **def** paintEvent(self, e): #for adding an image
390. p = QPainter()
391. p.begin(self)
392. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
393. p.end()

Billing\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** Bill\_ui **import** \*
3. **from** DatabaseSystem **import** \*
5. **class** Billing(QMainWindow):
6. **def** \_\_init\_\_(self, menu):
7. QMainWindow.\_\_init\_\_(self, None)
9. loader = QUiLoader()
10. form = loader.load("UI/uiBilling.ui", None)
12. self.setWindowTitle("SN Group Manager")
13. self.setCentralWidget(form)
14. self.setFixedSize(display\_width, display\_height)
15. self.menu = menu
16. self.db = DatabaseSys()
17. self.w = None
18. self.e = None
19. self.r = None
20. self.cb = form.findChild(QComboBox, "roomno")
21. self.cb.addItems(["101", "102", "103", "104", "105", "106", "107", "108",
22. "109", "110", "111", "112", "113", "201", "202", "203",
23. "204", "205", "206", "207", "208", "209", "210", "211",
24. "212", "213", "214", "215", "216", "217", "301", "302",
25. "303", "304", "305", "306", "307", "308", "309", "310",
26. "311", "312", "313", "314", "315", "316", "317", "401",
27. "402", "403", "404", "405", "406", "407", "408", "409",
28. "410", "411", "412", "413", "414", "415", "416", "417",
29. "501", "502", "503", "504", "505", "506", "507", "508",
30. "509", "510", "511", "512", "513", "514", "515"])
32. self.water = form.findChild(QSpinBox, "winput")
33. self.water.setMaximum(9999)
35. self.electricity = form.findChild(QSpinBox, "einput")
36. self.electricity.setMaximum(9999)
38. self.back\_btn = form.findChild(QPushButton, "back")
39. self.back\_btn.clicked.connect(self.back)
41. self.cont\_btn = form.findChild(QPushButton, "cont")
42. self.cont\_btn.clicked.connect(self.cont)
44. **def** back(self):
45. self.menu.show()
46. self.hide()
48. **def** cont(self):
49. self.w = self.water.text()
50. self.e = self.electricity.text()
51. self.r = self.cb.currentText()
52. **if** self.db.IsAvailable(self.r) == False:
53. self.bill = Bill(self, self.r, self.w, self.e)
54. self.bill.show()
55. self.hide()
56. **else**:
57. self.popUP("This room has no occupant")
59. **def** popUP(self, txt):
60. dialog = QDialog(self)
61. layout = QVBoxLayout()
62. label = QLabel(str(txt))
63. layout.addWidget(label)
64. close\_button = QPushButton("Close")
65. close\_button.clicked.connect(dialog.close)
66. layout.addWidget(close\_button)
67. dialog.setLayout(layout)
68. dialog.show()
70. **def** paintEvent(self, e): #for adding an image
71. p = QPainter()
72. p.begin(self)
73. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
74. p.end()

Bill\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** Billing\_ui **import** \*
3. **from** DatabaseSystem **import** \*
4. **import** datetime
6. **class** Bill(QMainWindow):
7. **def** \_\_init\_\_(self, billing, room, water, electric):
8. QMainWindow.\_\_init\_\_(self, None)
10. loader = QUiLoader()
11. form = loader.load("UI/uiBill.ui", None)
13. self.billing = billing
15. self.setWindowTitle("SN Group Manager")
16. self.setCentralWidget(form)
17. self.setFixedSize(display\_width,display\_height)
18. self.setStyleSheet('white')
20. self.db = DatabaseSys()
21. self.room = room
22. self.w = int(water)
23. self.e = int(electric)
24. self.t = 0
26. self.info = form.findChild(QLabel, "info")
28. self.water = form.findChild(QLabel, "water")
30. self.electricity = form.findChild(QLabel, "elect")
32. self.r = form.findChild(QLabel, "room")
34. self.total = form.findChild(QLabel, "total")
36. self.back\_btn = form.findChild(QPushButton, "back")
37. self.back\_btn.clicked.connect(self.back)
39. self.save\_btn = form.findChild(QPushButton, "save")
40. self.save\_btn.clicked.connect(self.save)
42. self.setInfo()
43. self.getElectricityPrice()
44. self.getWaterPrice()
45. self.getRoomPrice()
46. self.setTotal()
47. self.updateUnit()
49. **def** setInfo(self):
50. self.result = self.db.readResidentInfo(self.room) + "\n" + self.db.readRoomInfo(self.room)
51. self.info.setText(self.result)
53. **def** getWaterPrice(self):
54. w = self.db.getUtil(self.room, "water")
55. **if** type(w) == list:
56. w = w[0][0]
57. w = int(w)
59. **if** self.w < w: #meter resets after it exceeds 9999
60. total = (9999 - w) + self.w
61. **elif** self.w >= w:
62. total = self.w - w
64. **if** self.db.readResidentType(self.room) == 'monthly':
65. price = total \* float(self.db.getDefault\_util("water"))
66. self.t = self.t + price
67. result = "Water cost: " + str(price)
68. self.water.setText(result)
69. **elif** self.db.readResidentType(self.room) == 'daily':
70. result = 'Water used: ' + str(total)
71. self.water.setText(result)

74. **def** getElectricityPrice(self):
75. e = self.db.getUtil(self.room, "electricity")
76. **if** type(e) == list:
77. e = e[0][0]
78. e = int(e)
80. **if** self.e < e: #meter resets after it exceeds 9999
81. total = (9999 - e) + self.e
82. **elif** self.e >= e:
83. total = self.e - e
85. **if** self.db.readResidentType(self.room) == 'monthly':
86. price = total \* float(self.db.getDefault\_util("electricity"))
87. self.t = self.t + price
88. result = "Electricity cost: " + str(price)
89. self.electricity.setText(result)
90. **elif** self.db.readResidentType(self.room) == 'daily':
91. result = 'Electricity used: ' + str(total)
92. self.electricity.setText(result)
94. **def** getRoomPrice(self):
95. **if** self.db.readResidentType(self.room) == "daily":
96. price = self.db.getRoomPrice(self.db.readRoomType(self.room))
97. price = int(price[0][0])
98. inDate = self.db.getCheckinoutDate(self.room)[0][0]
99. outDate = self.db.getCheckinoutDate(self.room)[0][1]
100. days = 0
101. inyear, inmonth, inday = inDate.split('/')
102. outyear, outmonth, outday = outDate.split('/')
103. checkin = datetime.date(int(inyear), int(inmonth), int(inday))
104. checkout = datetime.date(int(outyear), int(outmonth), int(outday))
105. duration = checkout - checkin
106. duration = duration.days
107. **if** duration == 0:
108. duration = 1
109. price = duration\*price
110. self.t = self.t + (price)
111. result = "Room cost: " + str(price)
112. self.r.setText(result)
114. **elif** self.db.readResidentType(self.room) == "monthly":
115. price = self.db.getRoomPrice(self.db.readRoomType(self.room))
116. price = int(price[0][1])
117. self.t = self.t + price
118. result = "Room cost: " + str(price)
119. self.r.setText(result)
121. **def** updateUnit(self):
122. **if** self.db.getUtil(self.room, "water") == '0' **or** self.db.getUtil(self.room, "electricity") == '0':
123. self.db.insertUtil(self.room, self.db.getCheckinoutDate(self.room)[0][1], str(self.w), str(self.e))
125. **else**:
126. self.db.editUtil(self.room, self.db.getCheckinoutDate(self.room)[0][1], str(self.w), str(self.e))
128. **def** setTotal(self):
129. result = "Total: " + str(self.t)
130. self.total.setText(result)
132. **def** back(self):
133. self.billing.show()
134. self.hide()
136. **def** save(self):
137. filename = 'Bills/billshot.jpg'
138. p = QPixmap.grabWindow(self.winId(), 0, 0, 800, 400)
139. p.save(filename, 'jpg')
141. image = QImage("Bills/billshot.jpg")
142. printer = QPrinter()
143. printer.setOutputFormat(QPrinter.PdfFormat)
144. time = datetime.datetime.now().strftime('%H:%M:%S')
145. time = time.replace(':', '-')
146. f = "Bills/bill\_" + str(self.room) + "\_" + str(datetime.date.today()) + "\_" + time + ".pdf"
147. printer.setOutputFileName(f)
149. painter = QPainter()
150. painter.begin(printer)
151. painter.drawImage(QRect(0,0,display\_width,display\_height), image)
152. painter.end()
154. **def** paintEvent(self, e): #for adding an image
155. p = QPainter()
156. p.begin(self)
157. p.drawImage(QRect(0,0,display\_width,display\_height), bill\_bg)
159. p.end()

Setting\_ui.py

1. **from** constant\_initialization **import** \*
2. **from** DatabaseSystem **import** \*
4. **class** Setting(QMainWindow):
5. **def** \_\_init\_\_(self,menu):
6. QMainWindow.\_\_init\_\_(self, None)
8. loader = QUiLoader()
9. form = loader.load("UI/uiSetting.ui", None)
11. self.setWindowTitle("SN Group Manager")
12. self.setWindowIcon(QIcon('images/icon.png'))
13. self.setCentralWidget(form)
14. self.setFixedSize(display\_width,display\_height)
16. self.db = DatabaseSys()
18. self.menu = menu
20. self.back\_btn = form.findChild(QPushButton,"back\_btn")
21. self.back\_btn.clicked.connect(self.back)
23. #Utilities Section
24. self.electric = form.findChild(QDoubleSpinBox,"electric\_doubleSpinBox")
25. self.water = form.findChild(QDoubleSpinBox,"water\_doubleSpinBox")
27. self.electric.setMaximum(999.99)
28. self.water.setMaximum(999.99)
30. self.water.setValue(float(self.db.getDefault\_util("water")))
31. self.electric.setValue(float(self.db.getDefault\_util("electricity")))
33. self.utility\_update\_btn = form.findChild(QPushButton,"utility\_update\_btn")
34. self.utility\_update\_btn.clicked.connect(self.utility\_update)
36. self.utility\_default\_btn = form.findChild(QPushButton,"utility\_default\_btn")
37. self.utility\_default\_btn.clicked.connect(self.utility\_default)

40. #Renting Fees Section
41. self.studioD = form.findChild(QSpinBox,"studio\_daily\_spinBox")
42. self.studioM = form.findChild(QSpinBox,"studio\_monthly\_spinBox")
43. self.oneBedroomD = form.findChild(QSpinBox,"one\_bedroom\_daily\_spinBox")
44. self.oneBedroomM = form.findChild(QSpinBox,"one\_bedroom\_monthly\_spinBox")
45. self.twoBedroomD = form.findChild(QSpinBox,"two\_bedroom\_daily\_spinBox")
46. self.twoBedroomM = form.findChild(QSpinBox,"two\_bedroom\_monthly\_spinBox")
48. self.studioD.setMaximum(99999)
49. self.studioM.setMaximum(99999)
50. self.oneBedroomD.setMaximum(99999)
51. self.oneBedroomM.setMaximum(99999)
52. self.twoBedroomD.setMaximum(99999)
53. self.twoBedroomM.setMaximum(99999)
55. self.studioD.setValue(int(self.db.getDefault\_d("studio")))
56. self.studioM.setValue(int(self.db.getDefault\_m("studio")))
57. self.oneBedroomD.setValue(int(self.db.getDefault\_d("1-bedroom")))
58. self.oneBedroomM.setValue(int(self.db.getDefault\_m("1-bedroom")))
59. self.twoBedroomD.setValue(int(self.db.getDefault\_d("2-bedroom")))
60. self.twoBedroomM.setValue(int(self.db.getDefault\_m("2-bedroom")))
62. self.studio\_update\_btn = form.findChild(QPushButton,"studio\_update\_btn")
63. self.studio\_update\_btn.clicked.connect(self.studio\_update)
65. self.studio\_default\_btn = form.findChild(QPushButton,"studio\_default\_btn")
66. self.studio\_default\_btn.clicked.connect(self.studio\_default)
68. self.oneBedroom\_update\_btn = form.findChild(QPushButton,"one\_bedroom\_update\_btn")
69. self.oneBedroom\_update\_btn.clicked.connect(self.oneBedroom\_update)
71. self.oneBedroom\_default\_btn = form.findChild(QPushButton,"one\_bedroom\_default\_btn")
72. self.oneBedroom\_default\_btn.clicked.connect(self.oneBedroom\_default)
74. self.twoBedroom\_update\_btn = form.findChild(QPushButton,"two\_bedroom\_update\_btn")
75. self.twoBedroom\_update\_btn.clicked.connect(self.twoBedroom\_update)
77. self.twoBedroom\_default\_btn = form.findChild(QPushButton,"two\_bedroom\_default\_btn")
78. self.twoBedroom\_default\_btn.clicked.connect(self.twoBedroom\_default)
80. #Password Section
81. self.managerPass = form.findChild(QRadioButton,"manager\_radioButton")
82. self.receptionistPass = form.findChild(QRadioButton,"receptionist\_radioButton")
84. self.old\_password = form.findChild(QLineEdit,"old\_password\_lineEdit")
85. self.new\_password = form.findChild(QLineEdit,"new\_password\_lineEdit")
86. self.confirm\_new\_password = form.findChild(QLineEdit,"confirm\_new\_password\_lineEdit")
88. self.old\_password.setEchoMode(QLineEdit.Password)
89. self.new\_password.setEchoMode(QLineEdit.Password)
90. self.confirm\_new\_password.setEchoMode(QLineEdit.Password)
92. self.password\_update\_btn = form.findChild(QPushButton,"password\_update\_btn")
93. self.password\_update\_btn.clicked.connect(self.password\_update)
94. self.password\_default\_btn = form.findChild(QPushButton,"password\_default\_btn")
95. self.password\_default\_btn.clicked.connect(self.password\_default)
97. **def** utility\_update(self):
98. self.db.setUtilRate("water",float(self.water.value()))
99. self.db.setUtilRate("electricity",float(self.electric.value()))
101. **def** utility\_default(self):
102. self.water.setValue(float(self.db.getDefault\_util("water")))
103. self.electric.setValue(float(self.db.getDefault\_util("electricity")))
105. **def** studio\_update(self):
106. self.db.setting\_daily(str(self.studioD.value()),"studio")
107. self.db.setting\_monthly(str(self.studioM.value()),"studio")
109. **def** studio\_default(self):
110. self.studioD.setValue(int(self.db.getDefault\_d("studio")))
111. self.studioM.setValue(int(self.db.getDefault\_m("studio")))
113. **def** oneBedroom\_update(self):
114. self.db.setting\_daily(str(self.oneBedroomD.value()),"1-bedroom")
115. self.db.setting\_monthly(str(self.oneBedroomM.value()),"1-bedroom")
117. **def** oneBedroom\_default(self):
118. self.oneBedroomD.setValue(int(self.db.getDefault\_d("1-bedroom")))
119. self.oneBedroomM.setValue(int(self.db.getDefault\_m("1-bedroom")))
121. **def** twoBedroom\_update(self):
122. self.db.setting\_daily(str(self.twoBedroomD.value()),"2-bedroom")
123. self.db.setting\_monthly(str(self.twoBedroomM.value()),"2-bedroom")
125. **def** twoBedroom\_default(self):
126. self.twoBedroomD.setValue(int(self.db.getDefault\_d("2-bedroom")))
127. self.twoBedroomM.setValue(int(self.db.getDefault\_m("2-bedroom")))
129. **def** password\_update(self):
130. newpass = self.new\_password.text()
132. **if** self.managerPass.isChecked() == True :
133. **if** newpass == self.confirm\_new\_password.text():
134. **if** self.db.getPassword("manager") == self.old\_password.text():
135. self.db.setPassword("manager",newpass)
136. self.old\_password.clear()
137. self.new\_password.clear()
138. self.confirm\_new\_password.clear()
139. self.popup("Manager password is changed")
140. **else**:
141. self.popup("Password is not correct")
142. **else** :
143. self.popup("Confirm password is not match!")
145. **elif** self.receptionistPass.isChecked() == True :
146. **if** newpass == self.confirm\_new\_password.text():
147. **if** self.db.getPassword("receptionist") == self.old\_password.text():
148. self.db.setPassword("receptionist",newpass)
149. self.old\_password.clear()
150. self.new\_password.clear()
151. self.confirm\_new\_password.clear()
152. self.popup("Receptionist password is changed")
153. **else** :
154. self.popup("Password is not correct")
155. **else**:
156. self.popup("Confirm password is not match!")
158. **def** password\_default(self):
159. **if** self.managerPass.isChecked() == True :
160. self.db.setDefaultPasswordForManager()
161. self.popup("Manager password is set to default")
162. **elif** self.receptionistPass.isChecked() == True :
163. self.db.setDefaultPasswordForReceptionist()
164. self.popup("Receptionist password is set to default")
166. **def** back(self):
167. self.menu.show()
168. self.hide()
170. **def** popup(self,text):
171. dialog = QDialog(self)
172. layout = QVBoxLayout()
173. label = QLabel(text)
174. layout.addWidget(label)
175. close\_button = QPushButton("Close")
176. close\_button.clicked.connect(dialog.close)
177. layout.addWidget(close\_button)
178. dialog.setLayout(layout)
179. dialog.show()
181. **def** paintEvent(self, e): #for adding an image
182. p = QPainter()
183. p.begin(self)
184. p.drawImage(QRect(0,0,display\_width,display\_height), bg)
185. p.end()

constant\_initialization.py

1. **import** sys
2. **from** PySide.QtCore **import**\*
3. **from** PySide.QtGui **import**\*
4. **from** PySide.QtUiTools **import**\*
5. **import** os
6. **import** time
8. display\_width = 800
9. display\_height = 600
10. center = ((1366-display\_width)/2,(768-display\_height)/2)
11. os.environ['SDL\_VIDEO\_WINDOW\_POS'] = "%d,%d" % center
13. #Pictures
14. bg = QImage('images/bg.png')
15. title\_font\_img = QImage('images/title\_font.png')
16. building\_img = QImage('images/SNGroupBuilding.jpg')