Members: Anthony Argel, EJ Brand, Michael Huang

Project: Hotel Reservation System (HRS)

Abstract

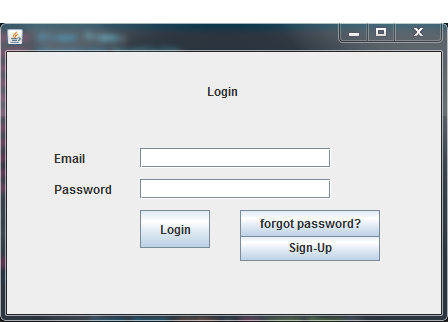
To accommodate tourists and businesspeople alike, a hotel must have a way for clients to book rooms easily. Our Hotel Reservation System (HRS) is an application that replicates a room reservation system for a given hotel. The customer will be able to use a GUI and choose their options and navigate through the layout of the reservation system. We have implemented various ideas and data structures we have learned throughout the course into our reservation system.

Introduction

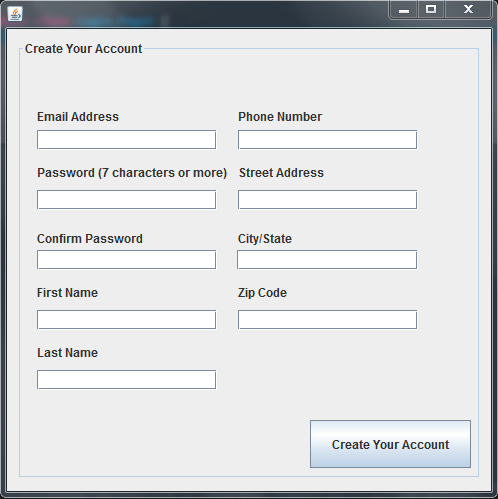
For the HRS, we utilized javax.swing and a GUI using JFrames for ease of navigation rather than a text based system. We start by having a log in screen with option to sign up for a new account. If one does not have an account, sign up for one. On the sign up screen, you are prompted with several fields, and some fields will have requirements. Once signed up, you can log in. In the hub, you are met with 2 choices, search for a room in the hotel, or view a reservation you have made. In the search room screen, you can choose a room to your liking. You can narrow down your preferences by displaying rooms of a certain type only, or sort them by certain criterion such as ascending or descending prices, room types, or room numbers by default. Before you click on the room you want, make sure you choose a date. The guest must have an overnight stay so the start date and end date must have a difference of 1. Once the dates are chosen, reserve it, and you can see which room you reserved for the dates chosen in the view reservations page. You can also choose to cancel the reservation if you click on the reserved room. Once a room is reserved, it will disappear until the reservation is cancelled.

Code Description

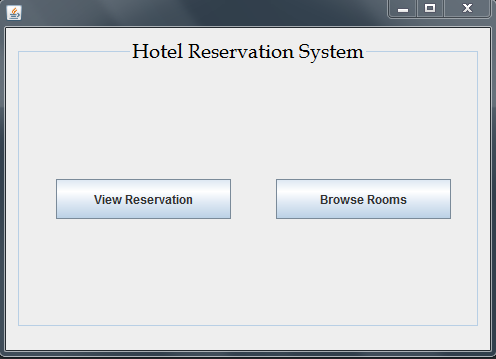
For the login page, the class employs a JFrame and has several text entry fields. There are JButtons that take you to various pages, the sign up page being the registration page for an account. The forgot password button is not really working as we did not know of a way to send an email in java.



The sign-up page is also another JFrame with various text entry fields, with the password fields hiding the actual password from plain sight. The create account button will check if the passwords match, then check if the phone number field contains only numbers, then the password field will check if the password meets the security requirement of 7 characters of more. Then the program checks for a zip code of all numbers with 5 digits or more. Once all these conditions are met, an account will be created and to log in, use the email associated with the account and the password. Accounts are saved in a text file which is read during log in. The password is encrypted using a HuffmanTree, and utilizes Huffman encryption, and is mentioned further in the report.

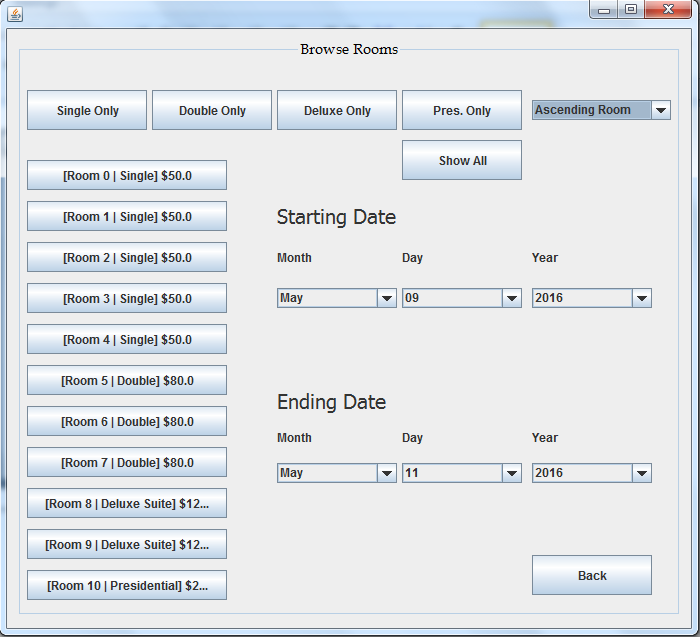


The main menu page and class is a JFrame with 2 buttons, view reservations, and browse rooms.



The browse rooms page is the Search\_Rooms class, and displays the available rooms as JButtons. Clicking on the button will reserve that room, given the starting and end dates are not equal and the end date is not before the start date. The dates are compared using a local Date class that we have written, which utilizes the java Calendar class. The default java Date class is very poorly written, in addition our own class provides sharper stability and accuracy and ease of use. Once proper dates are set, and then click a room to reserve it. There are many options to narrow down a search. One can use the display certain type of rooms only feature to suit your preferences. One can also use the sorting feature, which sorts the rooms in 6 criterias: ascending/descending room number, ascending/descending room type, and ascending/descending room price. Once a room is chosen, it will disappear from the browse rooms page as it is no longer available and is reserved.

Our project utilizes a Hotel object, which has a default of 11 rooms: 5 singles, 3 doubles, 2 deluxe suites, and 1 presidential suite. One can change the Hotel construction in the Hotel class. The Hotel class holds HotelRoom objects, which can be sorted. The sorting algorithm of choice is quicksort, which is mentioned further in the report. Our sorting can take in different criteria and has the ability to perform descending sort, which reverses comparison results.



Finally, the view reservation class can view and cancel your reservations. When view reservations, reservations are read from a text file and interpreted as a reservation button. By clicking on the button, one can cancel the reservation for that room, which will allow that room to appear in the browse room pages again.

Techniques

For sorting, we employed the efficient sorting algorithm of quicksort. The hotel room display in the “search for a room” frame is sorted using quicksort. We used the median of 3 implementation, and if a partition’s size is less than or equal 3, then it is sorted using insertion sort, just as we learned in the course. The sorting can take many criterions, such as sorting by ascending/descending room number, ascending/descending room types, and ascending/descending room price, with ascending room number being the default. This displays the rooms as buttons in the desired sorted order.

For trees, we employed a HuffmanTree using Huffman encoding for passwords. The Huffman tree takes a string, in this case the password, and it will go through a HashMap and split the keys and values, which will be the character array and the frequency of those characters in the array. Then it is passed into the HuffmanTree and is encoded, giving a binary string. When logging in, it will compare the entered password and the saved password’s encoded string.

For maps, we employed a HashMap for our reservation system. HashMaps were used in holding the reservations, and for determining the character array of the password string and the frequency of appearance of each letter. A Map sorting method is utilized to sort the values (the frequency) and key (character) mappings.

Conclusion

This project was fairly intuitive and straightforward. However, there was great difficulty and challenge in linking the separate frame together. Anthony managed most of the UI to data structure linking, Michael (the author of this report) wrote most of the data structures, and Ej worked on a majority of the frames and GUI. Meshing together individual parts were fairly difficult since we wrote different parts of the code, since we do not completely understand how the other components were written or its implementation. Overall, this project is a great learning experience, and we learned that team work is important and debugging is essential to make sure all features in out app are functional. Although, we wish we knew of a way to actually reset a password with the forgot password option, and we originally included other features as well, such as a payment system. We had it written, but it was scrapped due to time constraints and bugging linkage from frame to data structures.