Appraisal

Review of System objectives

In the analysis section I have outlined the objectives which need to be met. There are a total of 9 objectives which I have tried to achieve in the process of making the Room Hire System.

1. **The system should be able to store a minimum of 15 user accounts which can be logged into without any errors.**
   Initially the User file can hold a maximum of 50 unique users which have the ability to log into the system and make use of its resources. When the number of users in the file reaches the maximum then the program will automatically add in space for another 100 Users. This process will occur anytime the maximum number of users is reached. Since the User file could potentially store thousands of users, it is safe to assume that this objective has been met successfully.

2. **The system should allow the users to create and store information of a minimum of 5000 bookings and be able to display in an on-screen interface in approximately 40 seconds. The data should be stored for at least 2 years before it will be archived.**
   The system provides access to the Add New Booking form, which allows the users to a add new booking, and the Booking Calendar form which allows them to view the bookings taking place on a particular date. The process of adding a booking can be seen in screenshots 27 and 29 in the testing section, while the functionality of the Booking Calendar can be seen in screenshots 21 to 25. The time taken for the data to be displayed on the screen is, on average, less than 2 seconds; however, this will depend on the number of bookings. As the number of bookings in the file increases the time taken for the bookings to be loaded will decrease linearly.
   Much like the User file, the Booking file can be extended if the maximum number of bookings has been reached, meaning that the number of potential bookings can theoretically be infinite.
   Lastly, the system allows the user to create a back-up of the system which could easily be archived by the user if necessary. While the system does not provide an archiving system, my design allows for the addition of such a system thus I can safely conclude that this objective has been met.

3. **The system should be able to create a usage report detailing common statistics recorded by the system such as the most popular room or the most frequent customer within a specific time frame.**
   A noticeable shortcoming of the system is the fact that the system does not produce this report but instead it displays some of the system statistics on the “System Settings” form. This can be seen in screen shot 31 in the testing

section. In general, I do not believe that this objective has been met properly and would require more work in the future.

4. **Administrators should have the necessary facilities to manage their users. The system should allow them to add or delete users in less than 5 seconds.**
   System Administrators have access to the system settings menu which allows them to perform administrative tasks such as adding or removing users. This menu can be seen in screenshot 31 in the testing section. From the System settings form the Administrator can also access the Manage Users and Add New User forms which allow them to add and edit the users. This can be seen in screenshots 32 and 33. I possess enough evidence to suggest that this objective has been met.

5. **The system should grant users different access levels depending on what they will be using the system for. The access levels govern what parts of the system the users can and cannot access.**
   The Administrators have access to the Manage Users form which allows them to change a user’s access level if necessary. The access levels prevent certain users from accessing parts of the system they do not need to use. Evidence of this can be seen in screenshot 33. Since the access levels can be changed and are proven to prevent users from accessing parts of the system it seems logical to say that this objective has been met.

6. **The system should be able to create bookings, catering, and technical support reports in under 2 minutes.**
   The three reports mentioned in this objective have been merged into a single Booking report which can be generated by the system if the user requests this. This report is generated in the form of a Microsoft Word document which can be edited as needed by the user. Evidence of this can be seen in screenshots 25 and 26. This report is usually generated in about 40 seconds; therefore, the objective has been met.

7. **Users with the sufficient access levels should be able to view, edit, add, and delete customer details using an appropriate interface.**
   The Manage Customers and Add New Customer forms both provide an interface which allows the user to create, view, add and remove customers form the system if necessary. This can be seen in screenshots 7, 8, 9 and 10 in the testing section. From this evidence it can be inferred that this objective has been fulfilled.

8. **The system should include a facility which allows users to search for a particular customer or booking and retrieve their information within 10 seconds.**

At present there is no facility which would allow the customers to search through bookings and filter them depending on the search criteria, however, given the way in which the system was designed, it is entirely possible to implement such a system in the future. The entire procedure could easily be implemented on one of the existing forms in the form of a small text box which would allow the user to enter in their search criteria and press a button. While the search facility does not exist, it should still be noted that bookings can still be looked up using the booking calendar, however, the user can only view bookings by selecting a particular date.

9. **The system should allow for the deleting or amending of bookings. In the event of a cancellation the system should allow the deletion of a booking if the customer has given at least 48 hours' notice.**
   The ability to edit bookings is included in the Manage Bookings form. This includes being able to make a booking inactive. An inactive booking is not recognised as a valid booking by the system and hence if a booking occurs at the same time as an inactive booking an error will not take place as the booking does not exist in the eyes of the system. Even though the booking has been made inactive, it still exists within the system files. This could potentially lead to problems as the Booking file will eventually start to fill up with inactive bookings which will only take up space in the file and will slow down the system.

Overview of system and possible improvements

In general, the development of the system went very well and no major problems were encountered while coding. It should, however, be noted that the system underwent a series of revisions in order to optimise some of the processes and increase the ease of use. Changes were also made in order to make the code more readable and well structured.

While most of the major objectives have been met, I still believe that some improvements can be made. Validation is one of the biggest issues that I still think need to be sorted out. An example of a lack of validation is the fact that a user could easily add two items with the same name but a different price. This inconsistency could lead to severe problems and should definitely be addressed.

Another issue that I believe needs to be addressed is the inability to properly test the multi user capabilities of the system as I did not have access to multiple machines which would be capable of running the system. The code which deals with this has been looked over multiple times, but the only way to reveal any bugs would be to run it on multiple machines at the same time.

Apart from the log-in system and the access levels, the system lacks a proper security measures. The user passwords are stored in plaintext in the User file and could potentially be accessed by user who may want to use the system for malicious purposes. In order to improve this, the passwords should be stored as hashed strings which would be useless to a hacker as could not work out what the passwords actually are.

The last thing I would do to improve the system is to add SQL support to the system. This would make accessing the data much easier from the coder's perspective and also add more functionality to the program. With the use of SQL, adding a search facility to the program would be rather easy to do.