J2EE Project Proposal

By: Cup 'O Java (Katie Sullivan, Yuliia Shymbryk, Ujjawal Tivari)

Definition of Project

Our project is a hotel booking application, modeled after the site waringhousereservations.com. We chose to only model the booking functionality of the site, not the entire site.

In this site, users can select to stay at a large country inn with 4 different sites (hotels). They can select one of 5 different room types to stay in, and can select the duration of their stay. In order to sign up, the user needs to register themselves, and can save a pin number in order to access their registration information again. If they choose not to enter a pin, their booking will be saved, but they will have to re-enter information to make a new booking. They will still be able to modify or remove an existing booking. They can search for available rooms, add a booking, modify a booking, and remove a booking.

After making a booking, the user will be charged online, and will enter card information. We decided not to process payment, but our site will validate card information.

The administrator can access all bookings for all clients, and can add new bookings (such as when someone calls in to book), and can remove or modify all bookings.

Parts of Application

User:

1. User sees a page with option to create, change, or cancel a reservation.
2. To create a reservation, the user can search for availability by entering reservation details including: check in date, checkout date, rooms, number of adults, number of children and type of room. At this point, users may also enter a promotional or corporate code to receive discounts
3. After pressing enter, user is taken to a page displaying available rooms for the duration of their stay that match the room type requested. They can select a room, and begin making a reservation.
4. They user is taken to a reservation page, where either the user can sign in to access their account details, or they can enter their details, including credit information. All details will be validated. Once registered, user receives a confirmation page with a reservation number.
5. To change a reservation, the user needs to enter their reservation number, and credit card information. Once they press continue, the information will be validated in the database. If the reservation with matching details exists, user will see their reservation details populated in the table in the initial reservation page, and can modify it and save it as per above.

6. To cancel a reservation, user needs to enter reservation number, and credit card information. Once they press continue, the reservation will be removed.

Admin:

1. Admin will go to a separate page to login, and will be validated.
2. Once admin is logged in, they can search for a booking by reservation number, email address, or phone number and full name. After searching, it will return one or more reservations for that user.
3. Admin will then have the same ability to modify or delete bookings as the user.
4. Admin will also be able to create a reservation like the user does above.

Application Implementation:

We will be implementing our application using Spring MVC framework.

Please see class diagrams and use case diagram attached

Our model classes will represent our database tables, and include classes for:

- User
- Booking
- Hotel
- Room
- Database

Our controller classes will implement the functionality of the site, including:

- Initial – returns the index page
- Search
- SearchResults
- AddBooking
- ModifyBooking
- EditController
- DeleteBooking
- AdminSearch
- AdminLogin

Our views will match the controllers and include:

- index.jsp
- searchResults.jsp
- createBooking.jsp
- modifyBooking.jsp
- editBooking.jsp
- login.jsp
- adminSearch.jsp

Detailed Description in Text of MVC Relationships:

Models
For each table, except an admin one, the corresponding models have been created.

- User model has all the information for a specific guest who is making a booking as well as validation for these fields.
- Room model has the information about a specific room such as a price, image and description.
- Hotel model is composed of rooms and has attributes describing the number of each room available.
- Booking model contains the information about a specific booking such as dates, a total price. Booking also contains the information about a room type, a hotel type and the guest id.
- Database class contains all the methods to work with the database. This class will be used by the controllers to access the database.

Controllers and Views structure

All controllers use the database class

User and Admin Options

1. Initial.java controller renders the index page

2. Index.jsp page allows the user to enter the details to make a search. These details then will be sent to Search.java controller after submitting to extract the possible options from the database and pass them to the new view – searchResults.jsp. The booking model is used to validate some of the booking information. The room model is used to create a list of available rooms.

3. searchResults.jsp view contains the list of available rooms returned by the search controller. The user can select an option and then the information goes to the SearchResult.java controller that passes the information about the user choice and redirects the user to the createBooking.jsp

4. createBooking.jsp contains the form for a user to make a booking. Users have two options: to log in and then the data for the form will be auto-populated from the database, or the user can enter the details. Depending on the user action the corresponding method from the AddBooking.java controller will be called for either populating the data or validating the user data using validation specified in the User model. If there are some errors the user will get message, otherwise a new record will be inserted into a database.

Apart from creating a booking, a user can modify and delete booking. For both these options modifyBooking.jsp view will be used. After a user enters the information and submits the form, it goes into the corresponding controller – either ModifyBooking.java or DeleteBooking.java.

Both these controllers use the method from the database class to find a specific record depending on the user's information. In case no such record exists in the database, the user will get an error message. Otherwise,

1. If a user wants to modify a booking he/she will be redirected to editBooking.jsp which will be populated with the booking information. The user here can change some parameters of the booking and check whether the booking can be changed or not. This will be done by EditController.java controller. This controller will return a message in case a new option is not available. If a booking can be edited the changes will be made in the database and a user will be notified that the booking has been edited.

2. If a user wants to delete his booking, the record will be deleted from the database.

Admin Only Options

1. An admin has to go through login page login.jsp first to log in. Then the input will be checked in the AdminLogin.java controller, if it matches the information contained in the admin table, the session will be created and the admin will be redirected to the adminSearch.jsp view. If not - the corresponding error message will pop up.
2. adminSearch.jsp view allows an admin to make a search using different user's attributes (reservation number, email, name). The logic behind this search will be handled by AdminSearch.java controller.

The same functionality available for a user is also available for an admin. Thus, an admin has an option to go to the site, but as the session will be created, to go back to the adminSearch.jsp there will be a link in the bottom to open this page.

Responsibilities:

So far our group has been sharing responsibilities.

Each group member contributed 1+ tables (writing the database queries), and 1+ models corresponding to the tables. Each group member contributed text and/or diagrams to the project proposal. All 3 of us sat down together to define the functionality, and create the methods of each class (not all methods are implemented).

Ujjawal will be primarily in charge of creation of views, while Katie and Yuliia will be primarily in charge of the controller classes. All group members will meet weekly to discuss the implementation of the project and work together.

Group data will be stored on github, and file sharing will occur here.