**Reservation Service**

This service is used to reserve rooms in the hotel , this service is common to both traveller and hotel admin(employee).

|  |  |
| --- | --- |
| **Interface** | **Description** |
| reservation/save | Traveller and Employee can reserve a room.  **Request**  Method : POST  Body : Request body should contain the following fields.   * roomId * noOfAdults * noOfChildren   **Response**  An email and an SMS will be sent to the relevant user. |
| reservation /delete/:id | Traveller and Employee can cancel a reservation  **Request**  Method : DELETE  Body : Request body should contain the following fields.   * objectID of reservation   **Response**  Frontend will receive a “Delete succesful” message. |
| reservation /getAll/:username | Used to get all reservations by username. This is used by the traveller.  **Request**  Method : GET  Parameter : Username is passed as the parameter  **Response**  Frontend will receive the response data set as json |
| reservation /get/:id | Used to get a particular reservation.  **Request**  Method : GET  Parameter : Object ID is passed as parameter  **Response**  Frontend will receive the reservation details as json |
| reservation /update/:id | Used to update a particular reservation  **Request**  Method : PUT  Parameter : Object ID is passed as a parameter  **Response**  Frontend will receive a confirmation message as response. |

An SMS and an Email is sent to the user upon successful reservation.

For Email sending “nodemailer” library has been used. For SMS sending “vonage” library has been used

**A screenshot of a computer

Description automatically generated with low confidenceSequence diagram**

**Calendar

Description automatically generated**

All other routes show same behavior as the “Get All Reservations Route” above.

**Login Service**

The Login service has been implemented as to be common for all the users. User in the frotend will select the respective user type , then they will be redirected to a common web page (User type variable will be passed upon user type selection) . Then based on the user type , the relevant request will be sent.

**Authentication in Pages :** User levels have been defined. Relevant users can only access their pages. For example ; a traveller won’t be able to access the hotel-add-reservation page , they will be redirected to the login in case they input the link in the URL.

|  |  |
| --- | --- |
| **Interface** | **Description** |
| login/check/:username | Traveller can login  **Request**  Method : Get  Parameter : username  **Response**  Validity of username whether it exists or not |
| login/get/:username | After checking username , the password is checked  **Request**  Method : GET  Parameter : username  **Response**  Username |
| **Similarly , this has services for Hotel admin and System Admin** | |

**Calendar

Description automatically generated with medium confidenceSequence diagram**

**Registration Service**

This service is used by traveller , users can register to the system using this service

|  |  |
| --- | --- |
| **Interface** | **Description** |
| user/save | Traveller can register to the system.  **Request**  Method : Get  Body : Traveller Details , username and pasword are required.  **Response**  Succesful message upon valid registration. |

**Sequence diagram**

**Calendar

Description automatically generated**