***Seminar Hall Booking Application***

**Requirements**

**Feature 1 – Booking and Cancelling**

1. Server
2. Client
3. UI/Menu (user/admin) for Booking
   1. Ask Date (if too Far no Booking now)
      1. If date ok then go to next step
   2. Which Hall
   3. One day
      1. Full
      2. Slots(Single, More than one)
         1. 7 slots each day – 9AM to 4PM
   4. Multiple Days
      1. Same Slot
      2. Different Slot
   5. Details of the Client User
4. UI/Menu (user/admin) for cancelling
   1. Cancelling
   2. Rescheduling
      1. Take details and do the same

**Feature 2 – To know who has booked a Particular slots**

1. Ask Time and Date
2. Display if Booked or Not
   1. If Booked – Display the Details of the Booked Person
   2. To Display How many slots have been Booked

**Feature 3 – To give Maintenance review**

1. Ask Date and Time of the slot already booked
2. Give Review
3. Save in Database so that Admin can take care if any Problem has occurred during the event

Design For Feature 1

Database Design

1. Two different tables for two halls
   1. CREATE TABLE hall(  
      Date date,  
      Slot\_number Integer,  
      Name Text,  
      Contact Integers  
      )
   2. Can add more columns if needed
   3. Time slot will not be stored   
      Only slot number will be stored

|  |  |
| --- | --- |
| **Time** | **Slot Number** |
| 9AM – 10AM | 1 |
| 10AM – 11AM | 2 |
| 11AM – 12PM | 3 |
| 12PM – 1PM | 4 |
| 1PM – 2PM | 5 |
| 2PM – 3PM | 6 |
| 3PM – 4PM | 7 |

Structure Design

Request Message Structure

Struct request{

int type;/\*0 - booking

1 - cancelling

2 - Rescheduling\*/

int hall;/\* 0 – old seminar hall

1 – new seminar hall\*/

int B\_day;/\*0 - single day

* + 1. Multiple day

typedef union no\_of\_days{

struct s\_day one\_day;

struct m\_day multiple\_days;}booking\_days;

};

For Single Day

struct s\_day{

time\_t Date;//Date of Booking

int no\_of\_slots;

int first\_slot\_number;

/\*if more than one slot than booking can be done for continues slots only\*/

}

For multiple days

Struct m\_day{

time\_t dates[];

int no\_of\_slots\_each\_day;

int first\_slot\_number;}

Once the Request message is sent

We receive a response

Response structure

struct response{

strcut request required;//the status for this requirements is sent

int status;/0 – Available

1 – Not available

}

If the status is available

Details of the user is Taken

Struct details{

Char name[15];

Char contact[11];

}

Once these details are taken database is updated and successful message is printed