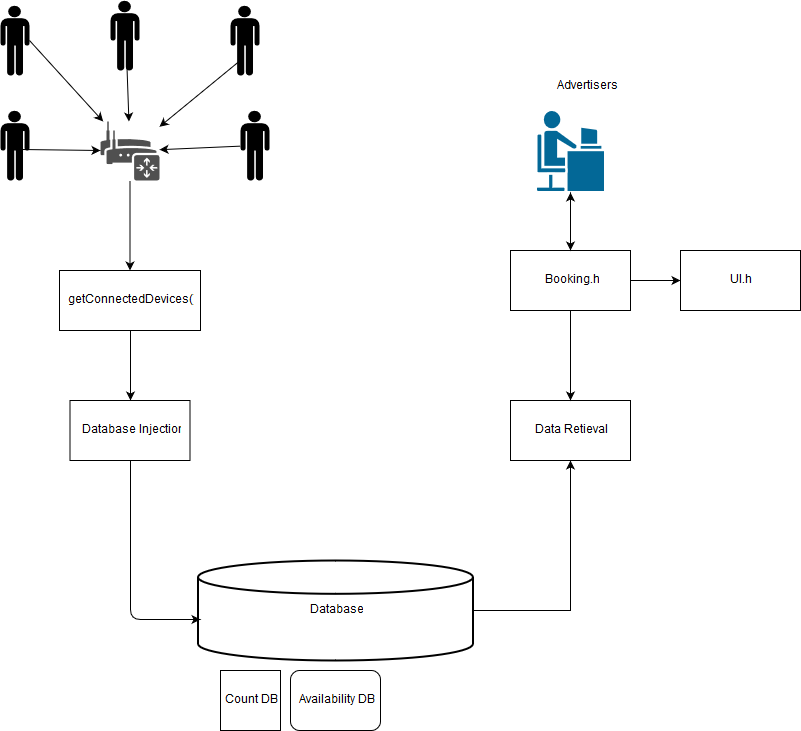
**Application Overview:**

Our application makes use of Wifi to detect the number of people visiting a place (Mall, Airport, etc). These statistics can be used for calculating prices of Advertisements for a given time frame.

The prices are calculated using the user count for the given time frame. This application is implemented using multi threadingwhere one thread updates the database at regular intervals and the other for managing the user interactions.



**Modules:**

**Module -1**:

1. int getCount():

This function returns the current count of users connected to a router. Function also clears the arp cache so the next time a fresh count is created (not containing previous )

**Module – 2:**

1. void updateDB():

This function updates the database by making use of the function getCount() to get the count of connected users at a given time . The time is calculated in this function itself.

1. int\* getData():

This function returns an array containing the count of users at each interval (of 1 hour).

1. Void addLog(char \*name, int price)

This function appends the data of the advertiser in a log file

1. Void showLog()

This function displays the log file.

**Module -2:**

1. int getAvailability(int time):

This function returns the availability status at a given time slot.

0 - > empty

1 -> full

1. void setAvailability(int time, int available):

This function sets the availability of a given time slot to a given value.

1. void resetAvailability(int time):

This functions sets the availability of a given time slot to 0 (empty).

1. int\* getAvailableSlots():

This function returns an array containing the availability statuses for each time slot.

**Module-3 :**

1. void gotoxy(int x, int y):

This function changes the cursor position to the given x, y location.

1. void rect(int x, int y, int l, int b):

This function prints a rectangle at given x,y with length l and bredth b

1. int max\_val(int \*values):

This function returns the maximum value in given array

1. int \*normalize\_data(int \*values):

This function normalizes the values in array based on maximum value in that array.

1. void \_print\_graph(int width, int height, int \*normalized\_values, int \*actual\_values):

This function depicts the values in a graph and the graph is printed with given width and height.

1. void print\_graph(int \*values):

This is a wrapper function of \_print\_graph()

1. void \_print\_data(int \*data, void (print\_method(int a))):

This function prints the given data using the passed function pointer.

1. void print\_free\_slot(int \*boolean\_data):

This function prints the free time slots using the Boolean\_data

1. void print\_24hr\_data(int \*values):

This function prints the count of users for the given time slots.

1. int main\_menu(int option\_count, ...):

This function prints and returns the option selected from the given variable main menu arguments.

1. char \*\*payment\_page(int price):

This function prints the payment page which asks the client for his/her name and credit card details and returns them. It also prints the price to be paid.

**Module-3:**

1. int validateDebitCard(char\* debitCard, char\* name) :

This function is used for validating the debit card details.

1. void book():

This function takes the start and end booking times, checks if those time slots are available.

If available it takes the user to advertisement registration page.

1. Int isFreeSlot(int startTime, int endTime):

This takes the requested time interval as input and checks whether the specified interval can be allocated or not by checking the interval with the data in the availability.db and returns if the slot can be allocated or not

**Source.cpp**:

1. void multithreading():

This function creates two threads, one (DBside) for updating the counts of the people connected to the wifi in the DB and the other thread(ClintSide) for client side.

1. void DBinitialization():

This is a function which sleeps(keeps the thread in a waiting state) for an hour and updates the count in the data for every hour.

1. void dataFetching():

This is a thread runs the UI and the server of the application.