# HOTEL DATA GATHERING

**A PROJECT REPORT**

*Submitted in partial fulfillment of the*
*requirements for the award of the degrees*

***of***
**BACHELOR OF TECHNOLOGY**
**in**

**COMPUTER SCIENCE AND ENGINEERING**

*Submitted by:*
**Surendra Chouhan**

*Guided by:*
**Dr. Bodhisatwa Mazumdar,**
**Associate Professor,**
**Discipline of Computer Science and Engineering**

**INDIAN INSTITUTE OF TECHNOLOGY INDORE**
**December, 2017**

## CANDIDATE’S DECLARATION

I hereby declare that the project entitled **“Hotel Data Gathering”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science and Engineering’ completed under the supervision of **Dr. Bodhisatwa Mazumdar, Associate Professor, Discipline of Computer Science and Engineering, IIT Indore** and Mr**. Prasad Kunte, Senior Manager at IDeaS-A SAS Company** is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

**Surendra Chouhan**

---

## CERTIFICATE BY BTP GUIDE

It is certified that the above statement made by the students is correct to the best of my knowledge.

**Dr. Bodhisatwa Mazumdar,**

**Associate Professor,**

**Discipline of CSE,**

**IIT Indore,**

## Preface

This report on "Hotel Data Gathering" is prepared under the guidance of Dr. Bodhisatwa Mazumdar, Associate Professor Discipline of Computer Science and Engineering, IIT Indore and Mr. Prasad Kunte, Senior Manager at IDeaS.

Through in this report I have tried to give a detailed design of a web application that I have created during my internship period at IDeaS.

I have tried to the best of my abilities and knowledge to explain the content in a lucid manner. I have also added screens and figures to make it more illustrative.

**Surendra Chouhan**
B.Tech. IV Year,
Discipline of Computer Science and Engineering,
IIT Indore

## Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to my BTP project supervisor, Dr. Bodhisatwa Mazumdar, Associate Professor ,Discipline of Computer Science and Engineering, IIT Indore.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the IDeaS , who gave me the opportunity to work for the company. Special thanks goes to my team mate, Shubham Burewar, who worked with me on this project. Last but not least, many thanks go to the manager of the project, Mr. Prasad Kunte, and my mentor Mr. Chaitanya Deshmuk who have invested his full effort in guiding the team in achieving the goal. I have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

**Surendra Chouhan**
B.Tech. IV Year,
Discipline of Computer Science and Engineering,
IIT Indore

## Contents

# CHAPTER 1: INTRODUCTION

**Gather Online Hotel Data-**

- **What?**
    - Hotels
        - Are Geographically located
        - Have Star Rating
        - Have Reputation
        - May have competitor hotel in its vicinity
    - Given that we know hotel's geographical location, we should be able to get top n (configurable) competitors and their room rates for next 60 days.
- **Explore**
    - How can we get hotel's master data?
    - How can we get hotel's daily room rate data?
- **Why?**
    - So that we can offer IDeaS solution to any hotel(small and medium size) which do not have revenue management culture or they can't afford one.
    - These hotel can start getting some meaningful pricing decisions by analyzing above data.

Develop a web application which can gather client hotels' data from different sources, these sources can be online websites/ APIs/ local database. And the gathered data will be in the form of hotels' master data (hotels' basic information), its competitor in a given range and their daily room rates which can be used for determining the room rates for clients' hotel.

## ABOUT COMPANY

### IDeaS - A SAS Company-

It is a private company founded in 1989, having headquarter in Minneapolis, MN. IDeaS Pune is the major development center for the company.

With more than one million rooms priced daily on its advanced systems, IDeaS Revenue Solutions leads the industry with the latest revenue management software solutions and advisory services.

Powered by SAS® and more than 25 years of experience, IDeaS proudly supports more than 9,000 clients in 94 countries and is relentless about providing hoteliers more insightful ways to manage the data behind hotel pricing. IDeaS empower its clients to build and maintain revenue management cultures by focusing on a simple promise: Driving Better Revenue.

IDeaS has the knowledge, expertise and maturity to build upon proven revenue management principles with next-generation analytics for more user-friendly, insightful and profitable revenue opportunities—not just for rooms, but across the entire hotel enterprise.

### Specialties:

- Revenue Management
- Hospitality Pricing
- Forecasting & Revenue Optimization
- Car park
- Travel
- Hospitality
- Big Data
- Computer Software
- SaaS Applications
- Lodging
- Hotels
- Smart Spaces

## Problem Explanation:

There are so many hotels in the world categorized by star ratings or reputations.  These hotels have some geographical location on the map. So using this geographical location I need to find their competitor hotels in a certain distance from it. And rate for each competitor hotel for next n days.

This data than can be used for analyzing the prices of the competitor hotels for a particular day and by analyzing this data this application should be able to provide optimal pricing rate for the hotel. So it can gain better revenue.

To do so I have to explore how I can get the data for the hotels in this entire world. This data about the hotel can be called as 'master data' for that hotel. Master data includes the name of the hotel, location of the hotel, star rating of the hotel, reputation/review of the hotel, address of the hotel etc. along with the master data I have to find a way to get the rate of the hotels for next 60 days at least.

To get the hotels' master data I have to explore what can be the possible sources and what type of data can be used for the application? Explore how it can be relatable to the daily room rates. What kind of relation between them needed for the better pricing prediction?

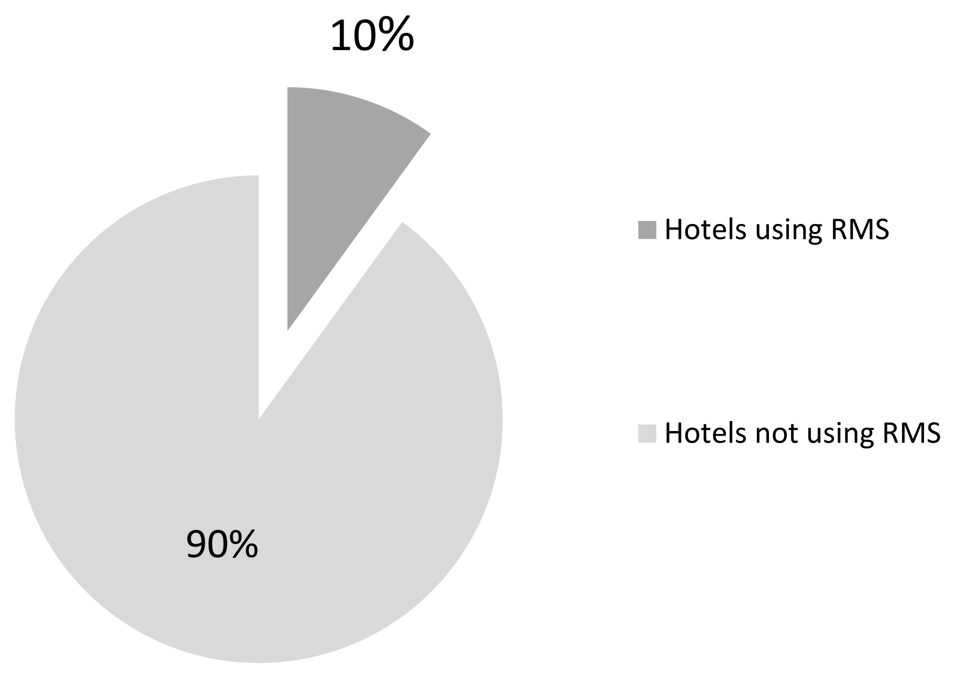
### Star rating:

Star ratings are often used to classify hotels according to their quality. There is a wide variety of rating schemes used by different organizations around the world. Many have a system involving stars, with a greater number of stars indicating greater luxury.

### Reputation:

The review given by the customers indicates the reputation of the hotel. It shows how hotel is fulfilling the customer's requirements.

## Motivation for this project:



In this entire world only 10% of the hotels are using Revenue Management System (RMS). And other 90% are not using any Revenue Management Systems because:

- RMS needs lots of history data (minimum one years' booking history of the hotel) which most of the hotels' don't have.
- This RMS are very costly, most of the hotels can't afford to use them.
- Most of the hotels are not aware of RMS culture.

So IDeaS sees it as an opportunity to make business. Since 90% of the hotels are not using any RMS. So introducing new product in the market will be an easy task for the company. After analyzing the factors that are affecting the use of RMS by hotels, IDeaS came up with a solution to develop a light weighted RMS for these hotels which will be cost effective and can be afford by any size hotel. This application will work on the daily room rates of the hotels. By analyzing the competitors' room rate it will help the client hotel to determine its room rate. This application will not require any history data so it will be easy for those hotels who don't maintain history records.

This application will help the company to enter in the RMS market with greater reach to all types of the hotels.

# CHAPTER 2: DEVELOPMENT APPROACH

To develop this application I have used the '**Agile Methodology'**. And to support agile development I have used the '**Test Driven Development (TDD)**' practice.

**Agile Model:**

Agile is a software development life cycle model and is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

**What is Agile?**

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

The most popular agile methods include:

- Rational Unified Process (1994)
- Scrum (1995)

- Crystal Clear
- Extreme Programming (1996)
- Adaptive Software Development
- Feature Driven Development
- Dynamic Systems Development Method (DSDM) (1995).

These are now collectively referred to as **Agile Methodologies**, after the Agile Manifesto was published in 2001.

Following are the Agile Manifesto principles –

- **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

**Agile Vs Traditional Software Development Life Cycle Models**

- Agile is based on the adaptive software development methods, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.
- Predictive methods entirely depend on the requirement analysis and planning done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.
- Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.
- Customer Interaction is the backbone of this agile methodology, and open communication with minimum documentation are the typical features of agile

development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

**Pros and Cons**

Agile methods are being widely accepted in the software world recently. However, this method may not always be suitable for all products. The advantages of the Agile Model are as follows −

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

The disadvantages of the Agile Model are as follows −

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

To support the agile model to develop this application I have used the ‘Test Driven Development (TDD)’ practice.

**Test Driven Development:**

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: Requirements are turned into very specific test cases, and then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements.

Test-driven development is related to the test-first programming concepts of extreme programming, begun in 1999,[3] but more recently has created more general interest in its own right.

**Test Driven Development Cycle:**

1**. Add a test**

In test-driven development, each new feature begins with writing a test. Write a test that defines a function or improvements of a function, which should be very succinct. To write a test, the developer must clearly understand the feature's specification and requirements. The developer can accomplish this through use cases and user stories to cover the requirements and exception conditions, and can write the test in whatever testing framework is appropriate to the software environment. It could be a modified version of an existing test. This is a differentiating feature of test-driven development versus writing unit tests after the code is written: it makes the developer focus on the requirements before writing the code, a subtle but important difference.

**2. Run all tests and see if the new test fails**

This validates that the test harness is working correctly, shows that the new test does not pass without requiring new code because the required behaviour already exists, and it rules out the possibility that the new test is flawed and will always pass. The new test should fail for the expected reason. This step increases the developer's confidence in the new test.

**3. Write the code**

The next step is to write some code that causes the test to pass. The new code written at this stage is not perfect and may, for example, pass the test in an inelegant way. That is acceptable because it will be improved and honed in Step 5.

At this point, the only purpose of the written code is to pass the test. The programmer must not write code that is beyond the functionality that the test checks.

**4. Run tests**

If all test cases now pass, the programmer can be confident that the new code meets the test requirements, and does not break or degrade any existing features. If they do not, the new code must be adjusted until they do.

**5. Refractor code**

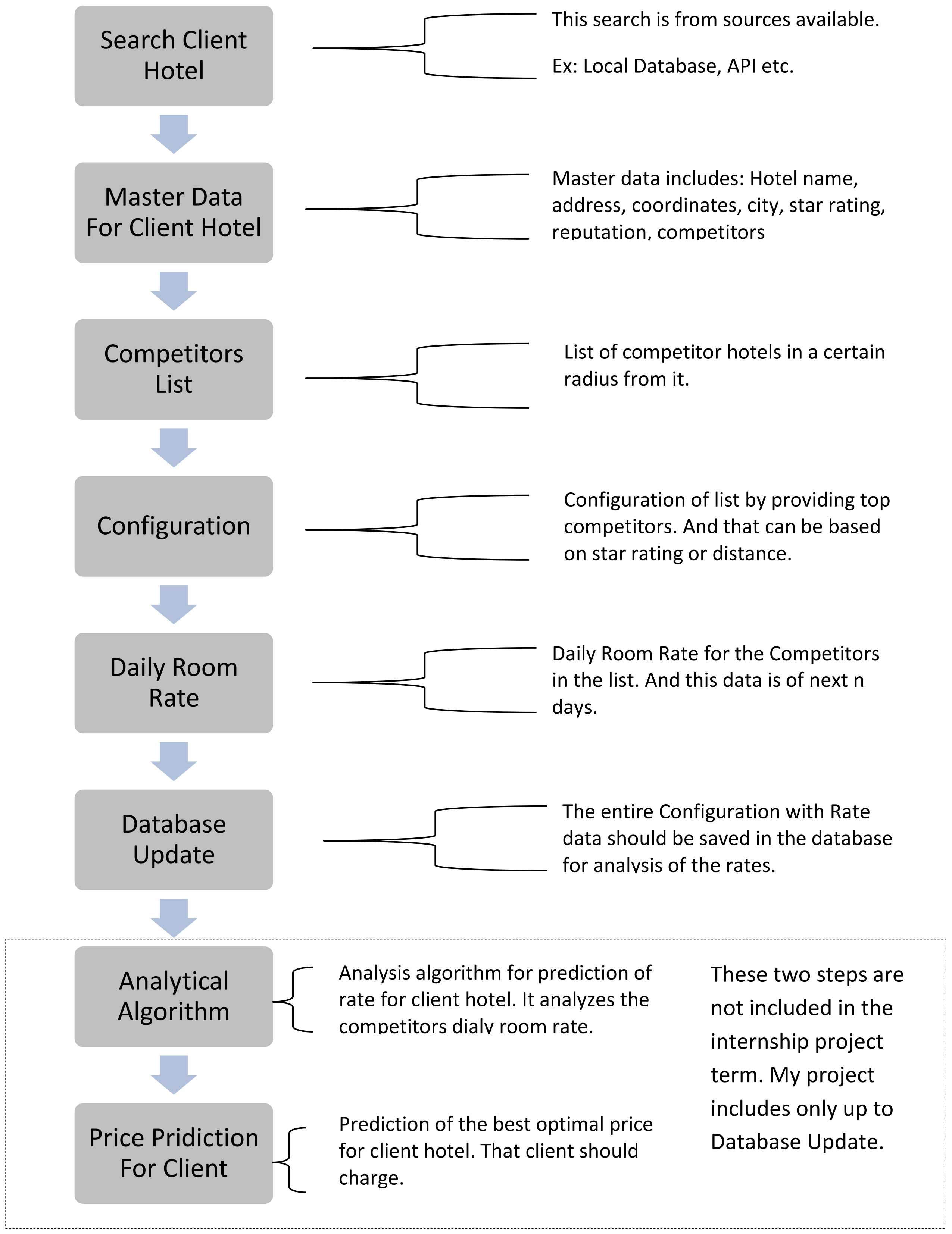
The growing code base must be cleaned up regularly during test-driven development. New code can be moved from where it was convenient for passing a test to where it more logically belongs. Duplication must be removed. Object, class, module, variable and method names should clearly represent their current purpose and use, as extra functionality is added. As features are added, method bodies can get longer and other objects larger. They benefit from being split and their parts carefully named to improve readability and maintainability, which will be increasingly valuable later in the software lifecycle. Inheritance hierarchies may be rearranged to be more logical and helpful, and perhaps to benefit from recognized design patterns. By continually re-running the test cases throughout each refactoring phase, the developer can be confident that process is not altering any existing functionality.

**Repeat**

Starting with another new test, the cycle is then repeated to push forward the functionality. The size of the steps should always be small, with as few as 1 to 10 edits between each test run. If new code does not rapidly satisfy a new test, or other tests fail unexpectedly, the programmer should undo or revert in preference to excessive debugging. Continuous integration helps by providing revertible checkpoints. When using external libraries it is important not to make increments that are so small as to be effectively merely testing the library itself, unless there is some reason to believe that the library is buggy or is not sufficiently feature-complete to serve all the needs of the software under development.

# CHAPTER 3: APPLICATION DESIGN

## Basic flow Diagram for the application


Search Client Hotel
This search is from sources available.
Ex: Local Database, API etc.
Master Data For Client Hotel
Master data includes: Hotel name, address, coordinates, city, star rating, reputation, competitors
Competitors List
List of competitor hotels in a certain radius from it.
Configuration
Configuration of list by providing top competitors. And that can be based on star rating or distance.
Daily Room Rate
Daily Room Rate for the Competitors in the list. And this data is of next n days.
Database Update
The entire Configuration with Rate data should be saved in the database for analysis of the rates.
Analytical Algorithm
Analysis algorithm for prediction of rate for client hotel. It analyzes the competitors dialy room rate.
Price Pridiction For Client
Prediction of the best optimal price for client hotel. That client should charge.
These two steps are not included in the internship project term. My project includes only up to Database Update.

**Master data:**

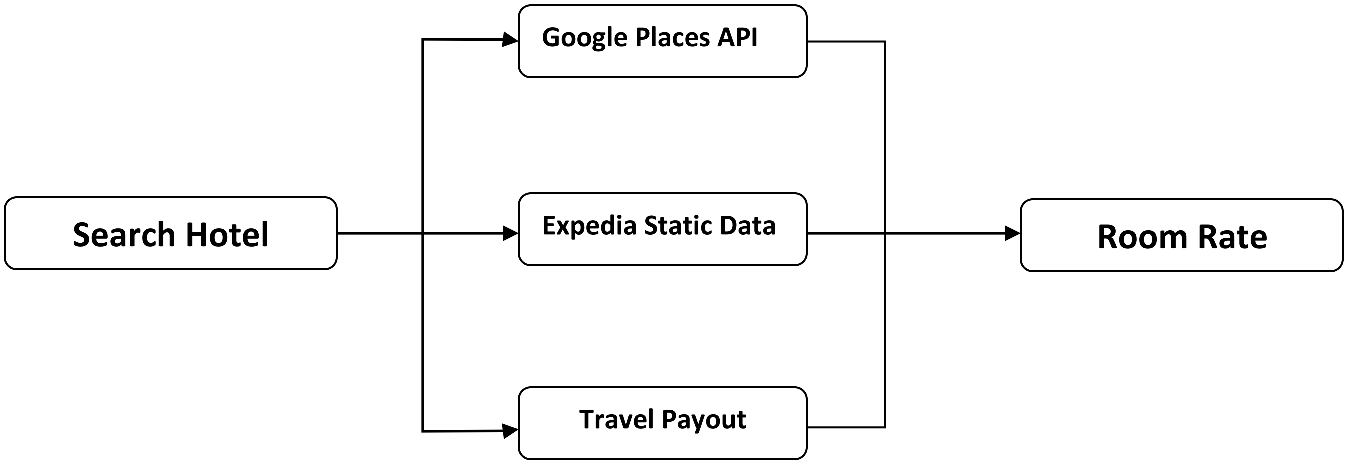
It includes the basic details about the hotels.
Ex: Name, Address, City, Country, Geo-Coordinates, Star rating, reviews, reputation etc.

**Sources for Master Data Collection:**
To collect the master data I had explored the different possible sources that can be used in this application. Some of them are listed below-

1. **Websites**
    - www.booking.com
    - in.hotels.com
    - www.goibibo.com
    - www.trivago.in
    - www.makemytrip.com
    - www.expedia.co.in/Hotels
    - www.hotwire.com
    - www.travelguru.com
    - www.agoda.com
    - www.priceline.com
    - http://hotel.yatra.com
2. **API**
    - Expedia
    - Google Places
    - Bookings.com
    - Travel Payouts

This all websites and API provides master data for the hotel, but the format and the content is different for all of them. Some of them have policies for using their data like: For every 5000(can be more or less for different websites) number of requests of the Hotel list request, expectation is at least 1 booking at minimum.

After understanding the working for all the sources I have decided to work with the static data provided by some of above API. Below listed sources are fulfilling the major requirement for the master data creation. These are easily available for the development purpose without any policy.



Initial flow for Master Data Collection

1. **Google Places API:**
   It is an API that provides access to information about more than 100 million places around the World. These places are, usually, public places like touristic attractions, hospitals, hotels and also stores, malls, companies etc. This API provides the details about the place and these are:
   - Name
   - Address
   - Latitude
   - Longitude
   - Rating
   - Type
   - Place id etc…

After getting these details, by using the coordinates (latitude and longitude) it allows us to locate the nearby places of certain type in a given range. So this helps in locating the competitor hotels for the client hotel and provides the details for all the competitor hotels.

2. **Expedia Static Data:**
It is an Online Travel Agency, which helps to book hotel reservation worldwide. So it has database for the hotels across the globe. And this database gets update in every seven day. Database has:
- Hotel Id
- Hotel name
- Address
- Star rating

- Latitude
- Longitude

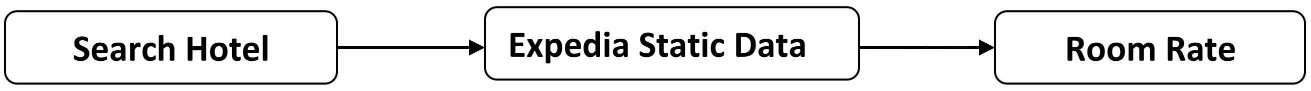
and so many other details but which is not relevant to this project. It has 233224 hotels' record and this helped in gathering master data for the client hotel.

**3. Travel Payouts Static Data:**
This is also an Online Travel Agency. Just like Expedia Static data it has similar database. It has 1674389 hotels' records.

These three are the major source for the master data of initial version of the project. But all of these have some issues with them. Like -
Google Places API gives the results based on the geo coordinate of the place and radius in which results should be shown but the result set is consist of the generalized places and not specific to the hotels only. Ex: lodging – this is one of the place categories which can be used to find places in Google Places API. This provides the hotels, hostels, restaurants, lodges etc. except hotels others are not required. So it is giving unnecessary data.
Expedia and Travel Payout static data is not a live data. This is provided by the respective agencies. But this data is updated once in every week. So to maintain the uniformity I need to update my database regularly in respect to get the updated data.

So, to overcome all of this I decided to work with only one source and that is **Expedias' static data**. Since it is more informative and has all the required data in it to work with.

Final flow for master data collection is:



## Competitors List:

To locate the competitor hotels in a given radius I have used the geo coordinates. Since the local database created by using Expedia Static data has the coordinates of the hotels. To find the list of the competitor hotels I have written a SQL query which finds the list of hotels satisfying the radius criteria.

```
SQL QUERRY : SELECT *,(6371* ACOS (COS ( RADIANS(:latitude) ) * COS( RADIANS( Latitude ) ) *
COS( RADIANS(Longitude ) - RADIANS(:longitude) ) SIN ( RADIANS(:latitude) ) * SIN( RADIANS(
Latitude ) ))) AS distance FROM hotels HAVING distance < :radius;
```

- Latitude: Client hotels latitude
- Longitude: Client hotels longitude

- latitude: latitude of the hotel for which distance is calculated
- longitude: longitude of the hotel for which distance is calculated
- radius : radius in which competitors hotel list is obtained
- hotels: name of the SQL table in which Expedia static data is stored.

Once list of competitor hotels is obtained user can configure it by selecting some of its major competitors and the selection criteria is dependent on user only, he can select the hotels by star rating , by distance or by review ratings. Once user configures the list the preference is saved in the local database against the hotel that user searched in the beginning. And this data than further will be used to get the daily room rates for the configured hotels only.

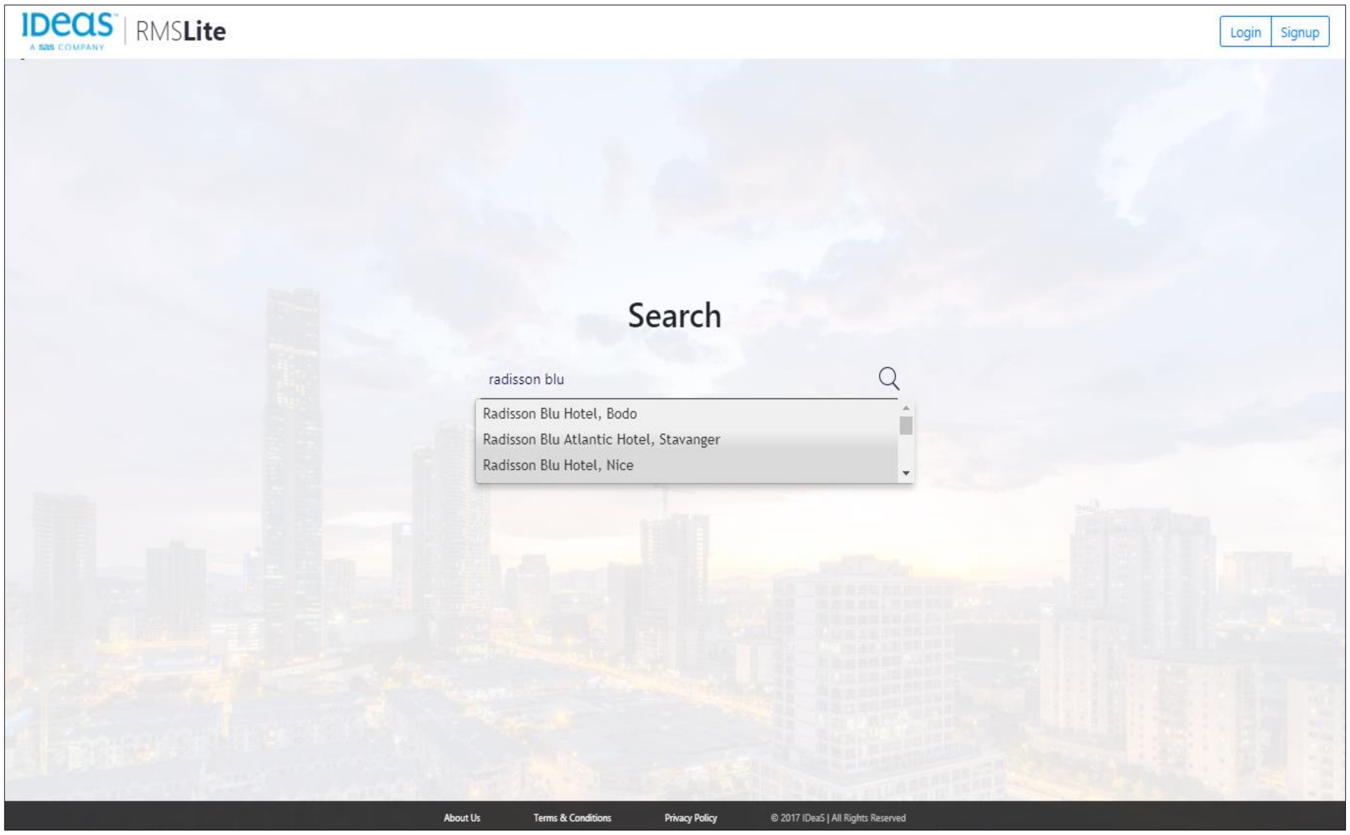
## Daily Room Rates:

Source for the Daily room rates is Rate Shopping Websites.

**Rate Shopping Websites**: These provide the daily room rates for the hotels in a particular city. And the rate data gets updated in every few hours. This makes it a better and reliable source for getting room rates. This will provide us the room rates for future days also, so by using these websites we can get the room rate for next 60 days or more. In this project I have created the local database schema for the rate data that can be stored by different dates. And the relationship between the rate data and hotel data is maintained. Each hotel will have the room rate data for next 60 or more day in rate data table. And this will help the analytical algorithm to predict the most optimal rate for the client hotel by comparing the rates of the competitor hotels.

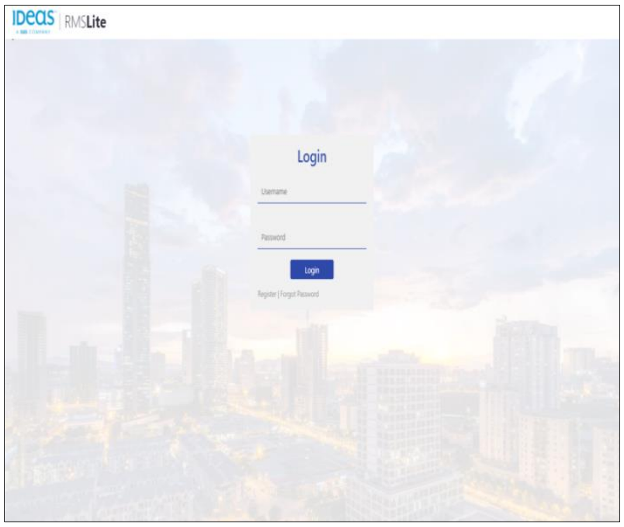
# CHAPTER 4: RESULT SCREENS

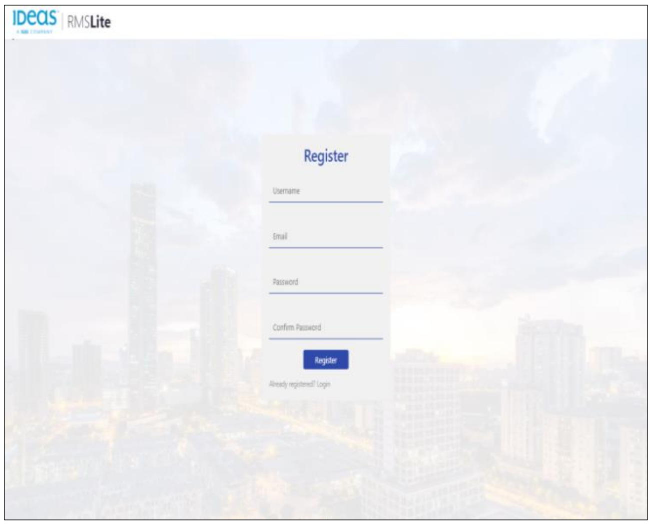
- **Home Screen:**



This is the Home page for the application. It consists of the search option with the auto completion feature. And these auto complete results are from local database that is created using Expedia Static Data. Home page also consist the options for the registration and login for the users. Only registered users are allowed to visit the application.

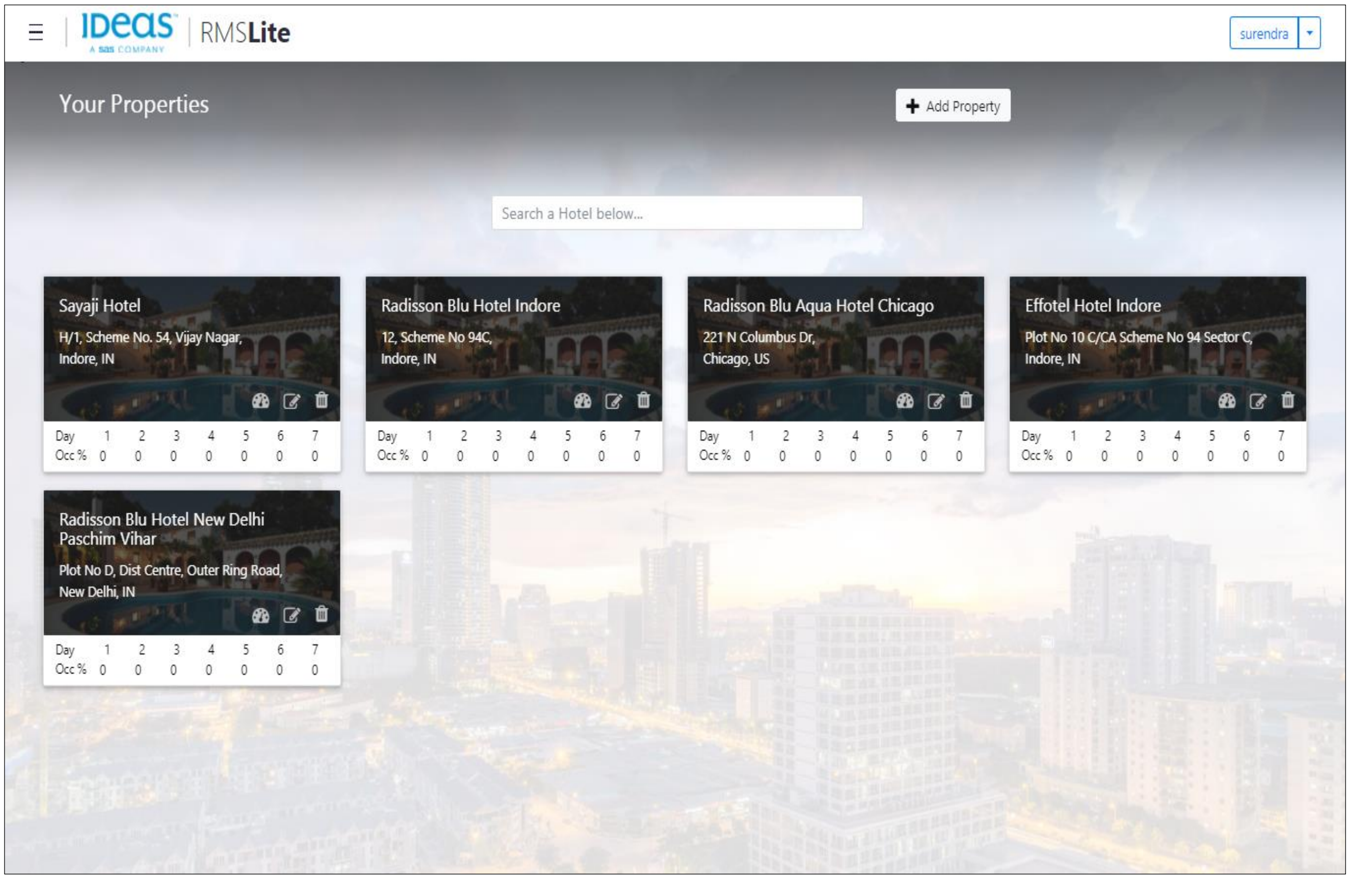
- **Login and Registration Page:**





These pages use the security feature of the Spring Boot framework. Spring Boot has the predefined security for the web app. And it includes the authentication, registration, login, logout, and sessions.

- **Users Property page:**



Sayaji Hotel
H/1, Scheme No. 54, Vijay Nagar,
Indore, IN

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Occ % | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Radisson Blu Hotel Indore
12, Scheme No 94C,
Indore, IN

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Occ % | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Radisson Blu Aqua Hotel Chicago
221 N Columbus Dr,
Chicago, US

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Occ % | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Effotel Hotel Indore
Plot No 10 C/CA Scheme No 94 Sector C,
Indore, IN

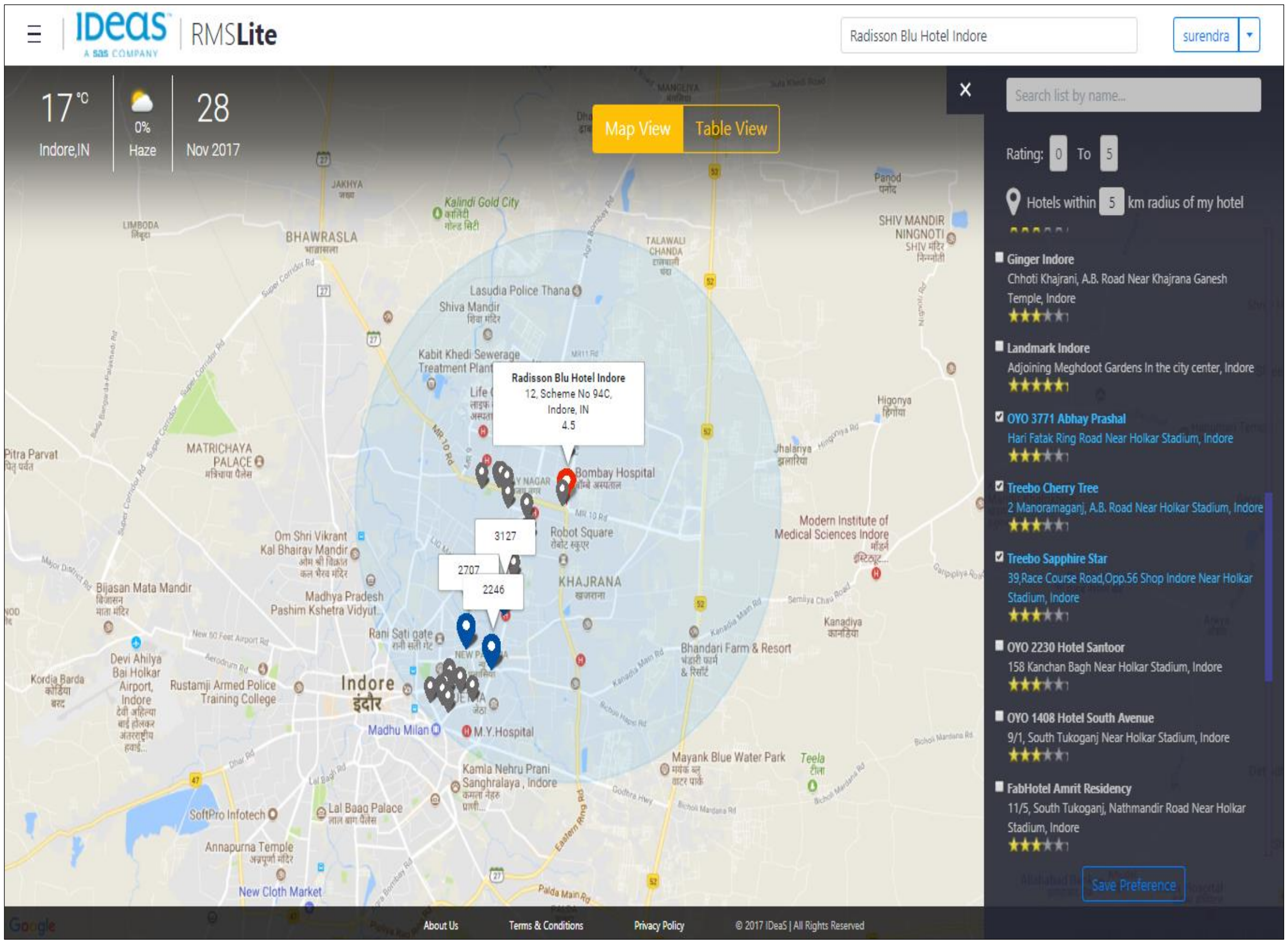
| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Occ % | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Radisson Blu Hotel New Delhi Paschim Vihar
Plot No D, Dist Centre, Outer Ring Road,
New Delhi, IN

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Occ % | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This is the page where a user can configure his/her properties. One user can have multiple properties and each property will have different configurations so this page is all about organizing these properties.
In this page user can add or remove properties, user can set the occupancy % for a particular property (this factor is not used in this project but later it will be used by the analytical algorithm).
Add option provides the hotel search function with the auto complete feature. It will be helpful for the user to identify his/her property.
From each property card user can navigate to the Dashboard or can remove the property from list or can set the occupancy.

- **Dashboard**



Radisson Blu Hotel Indore

surendra

Search list by name...

Rating: 0 To 5

Hotels within 5 km radius of my hotel

- [ ] **Ginger Indore** — Chhoti Khajrani, A.B. Road Near Khajrana Ganesh Temple, Indore
- [ ] **Landmark Indore** — Adjoining Meghdoot Gardens In the city center, Indore
- [x] **OYO 3771 Abhay Prashal** — Hari Fatak Ring Road Near Holkar Stadium, Indore
- [x] **Treebo Cherry Tree** — 2 Manoramaganj, A.B. Road Near Holkar Stadium, Indore
- [x] **Treebo Sapphire Star** — 39,Race Course Road,Opp.56 Shop Indore Near Holkar Stadium, Indore
- [ ] **OYO 2230 Hotel Santoor** — 158 Kanchan Bagh Near Holkar Stadium, Indore
- [ ] **OYO 1408 Hotel South Avenue** — 9/1, South Tukoganj Near Holkar Stadium, Indore
- [ ] **FabHotel Amrit Residency** — 11/5, South Tukoganj, Nathmandir Road Near Holkar Stadium, Indore

Save Preference

This is the dashboard for a particular property. It consists of a map in which the property is highlighted with an information window and a red marker. Sidebar contains the list of the possible competitor hotels in given radius i.e.5 km (in this screenshot). The list can be filtered by using the rating filters and text search which can only show the filtered hotels only. List is sorted on the basis of the distance from the client hotel. User can change the search radius according to his/her needs. And map consists of the markers of the hotels present in the list. There is weather data information is also available on the map, which is gathered using the OpenWeatherMap API. On the header bar there is information about the logged in user and a drop down list of properties from there user can switch to the dashboard of the other property.
User can select the desired competitor hotel from the list by checking the checkbox against each hotel in the list. Selected hotel information will be highlighted and the information about the rate will be shown on the marker of the respective hotel on the map.

User can save the selected hotels as preference list in the database by using save preference option. And this preference will help to get the daily room data for the selected hotels and will help in the prediction of the rate for client hotel.

- **Table view representation for rates of seven days**

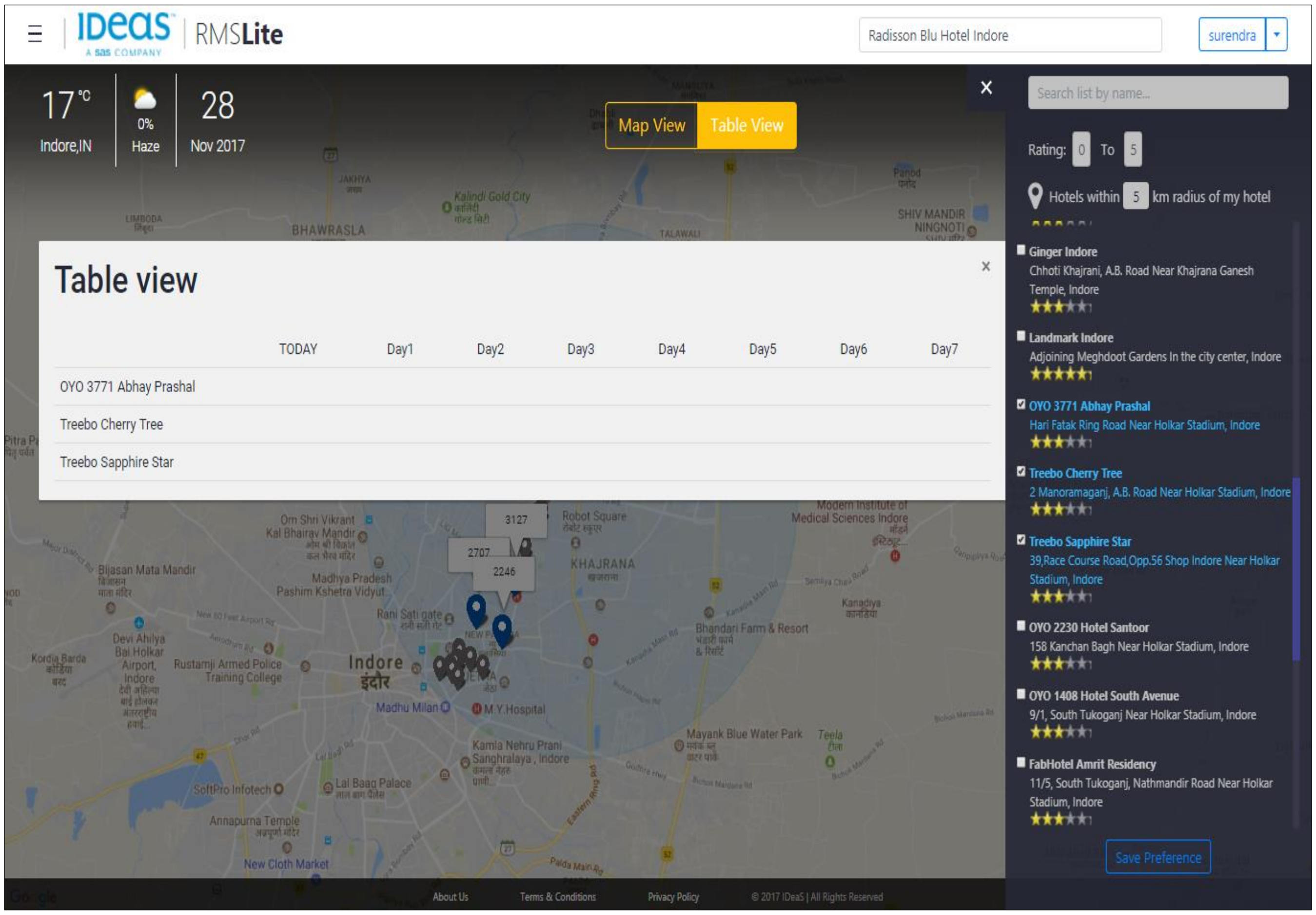


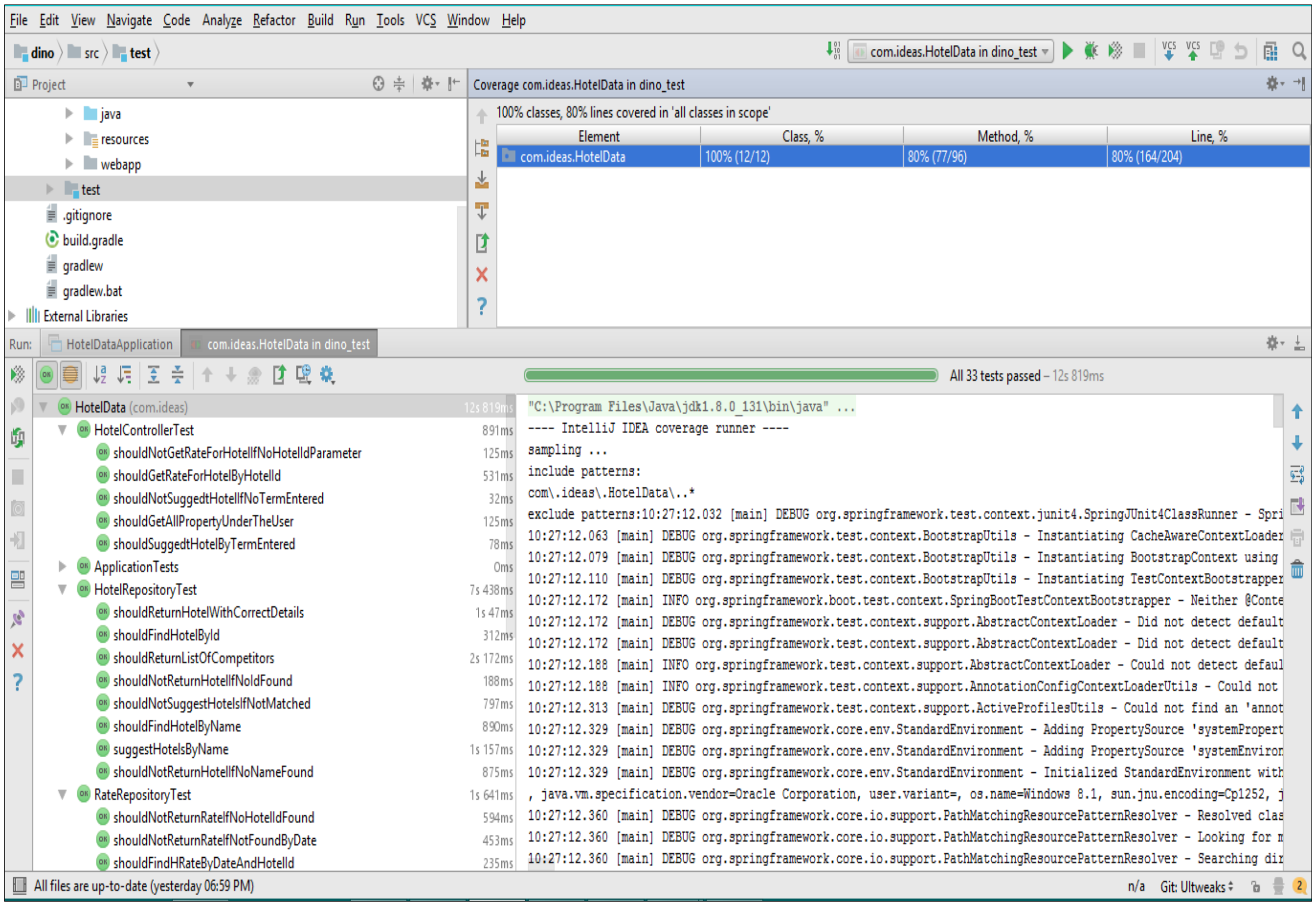
Table view

| | TODAY | Day1 | Day2 | Day3 | Day4 | Day5 | Day6 | Day7 |
|---|---|---|---|---|---|---|---|---|
| OYO 3771 Abhay Prashal | | | | | | | | |
| Treebo Cherry Tree | | | | | | | | |
| Treebo Sapphire Star | | | | | | | | |

- [ ] Ginger Indore
  Chhoti Khajrani, A.B. Road Near Khajrana Ganesh Temple, Indore
- [ ] Landmark Indore
  Adjoining Meghdoot Gardens In the city center, Indore
- [x] OYO 3771 Abhay Prashal
  Hari Fatak Ring Road Near Holkar Stadium, Indore
- [x] Treebo Cherry Tree
  2 Manoramaganj, A.B. Road Near Holkar Stadium, Indore
- [x] Treebo Sapphire Star
  39,Race Course Road,Opp.56 Shop Indore Near Holkar Stadium, Indore
- [ ] OYO 2230 Hotel Santoor
  158 Kanchan Bagh Near Holkar Stadium, Indore
- [ ] OYO 1408 Hotel South Avenue
  9/1, South Tukoganj Near Holkar Stadium, Indore
- [ ] FabHotel Amrit Residency
  11/5, South Tukoganj, Nathmandir Road Near Holkar Stadium, Indore

In this screen the selected hotels from the list also get added in the table and this table will show the seven days rate for a particular hotel. These seven day count start from the present day. User can switch between map view and table view by using toggle buttons on the top.

- **Code  coverage results for test cases:**


Coverage com.ideas.HotelData in dino_test

100% classes, 80% lines covered in 'all classes in scope'

| Element | Class, % | Method, % | Line, % |
|---|---|---|---|
| com.ideas.HotelData | 100% (12/12) | 80% (77/96) | 80% (164/204) |

All 33 tests passed – 12s 819ms

```
HotelData (com.ideas)                                12s 819ms
  HotelControllerTest                                    891ms
    shouldNotGetRateForHotelIfNoHotelIdParameter         125ms
    shouldGetRateForHotelByHotelId                       531ms
    shouldNotSuggedtHotelIfNoTermEntered                  32ms
    shouldGetAllPropertyUnderTheUser                     125ms
    shouldSuggedtHotelByTermEntered                       78ms
  ApplicationTests                                         0ms
  HotelRepositoryTest                                 7s 438ms
    shouldReturnHotelWithCorrectDetails                1s 47ms
    shouldFindHotelById                                  312ms
    shouldReturnListOfCompetitors                     2s 172ms
    shouldNotReturnHotelIfNoIdFound                      188ms
    shouldNotSuggestHotelsIfNotMatched                   797ms
    shouldFindHotelByName                                890ms
    suggestHotelsByName                               1s 157ms
    shouldNotReturnHotelIfNoNameFound                    875ms
  RateRepositoryTest                                  1s 641ms
    shouldNotReturnRateIfNoHotelIdFound                  594ms
    shouldNotReturnRateIfNotFoundByDate                  453ms
    shouldFindHRateByDateAndHotelId                      235ms
```

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
com\.ideas\.HotelData\..*
exclude patterns:10:27:12.032 [main] DEBUG org.springframework.test.context.junit4.SpringJUnit4ClassRunner - Spri
10:27:12.063 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoader
10:27:12.079 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using
10:27:12.110 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper
10:27:12.172 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @Conte
10:27:12.172 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
10:27:12.172 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
10:27:12.188 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect defaul
10:27:12.188 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not
10:27:12.313 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find an 'annot
10:27:12.329 [main] DEBUG org.springframework.core.env.StandardEnvironment - Adding PropertySource 'systemPropert
10:27:12.329 [main] DEBUG org.springframework.core.env.StandardEnvironment - Adding PropertySource 'systemEnviron
10:27:12.329 [main] DEBUG org.springframework.core.env.StandardEnvironment - Initialized StandardEnvironment with
, java.vm.specification.vendor=Oracle Corporation, user.variant=, os.name=Windows 8.1, sun.jnu.encoding=Cp1252, j
10:27:12.360 [main] DEBUG org.springframework.core.io.support.PathMatchingResourcePatternResolver - Resolved clas
10:27:12.360 [main] DEBUG org.springframework.core.io.support.PathMatchingResourcePatternResolver - Looking for m
10:27:12.360 [main] DEBUG org.springframework.core.io.support.PathMatchingResourcePatternResolver - Searching dir
```

All files are up-to-date (yesterday 06:59 PM)

This screen shows the result for the code coverage. Code coverage shows how much of the total code is tested using the test case that has been created.

It shows there are total of 33 test cases. There is no test case for User Interface Testing. **These 33 test cases have covered 100% classes, 80% of the methods and 80% of the code lines**.

# CHAPTER 5: TECHNOLOGIES USED

1. **Software Development Life Cycle Modal**
    - Agile Modal

2. **Agile Modal Technique**
    - Test Driven Development(TDD)

3. **API**
    - Google Places API
    - Google Maps API
    - Google Geocode API
    - OpenWeatherMap API
    - Jquery UI API

4. **Database**
    - MySQL

5. **Testing Tool**
    - JUNIT
    - Mockito

6. **Framework**
    - Spring Boot

7. **IDE**
    - IntelliJ IDEA

# CHAPTER 6: FUTURE SCOPE

Since this report is about the project to collect the master data for the hotel and rates of the hotels. This is a part of a larger project. So after this project, integration with the analytical algorithm will help this project to get the predicted prices for the client hotel. This will complete the product.

When this project is integrated with the analytical part, then the hotel of medium or small sizes will be having some affordable RMS option, which will help them to get the better pricing for them and also help them to gain better revenue. It will introduce the culture of RMS to the medium or small sized hotels.

Rates that are used in this project are of only one type of room in the hotel which is general room rate. In future it can be based on the type of rooms available in the hotel. Ex: suits, deluxe room, presidential room etc.

In this project testing was done for java classes and methods only, in future test cases for the user interface can be added and also the code coverage can be improved.

Performance can be improved in terms of load for the application. As it will be on the production site so the performance will play major role in there.

# CHAPTER 7: MY CONTRIBUTION TO PROJECT

- Created MySQL database for master data of the hotel.
- Created JavaScript file for Dashboard screen.
- Created JavaScript file for Property screen.
- Integration of the Dashboard with Google APIs.
- Added test cases for each Java class.
- Function for finding the competitor hotels from MySQL database.
- Implementation of Auto-complete feature.
- Created database schema for storing the preferences.
- Created the database schema for the user properties.
- Created database schema for rate shopping data.
- Implementation of the OpenWeatherMap API in the Dashboard.

# CHAPTER 8: CONCLUSION

The objective for this report is to develop a web application for the product of the company applying the standard J2EE development approach. This project was developed by following the agile methodology. During the development of this project I had to give daily progress report to my mentor. And he daily discussed the plan for the rest of the day with me. We followed the “scrum methodology” in which every day we have to conclude what we had done and what was the progress from previous day. Along with the mentor I had to present the progress in the project to my manager weekly. It’s like delivering the product to the client in small intervals and then manager suggested some changes or introduced some new features for the application. And that changes and feature would be the part of the next delivery. This all process concluded the agile development of the project.

I used the Test Driven Development approach for the agile development of the project. So the results for the code coverage are as follows:

- Classes : 100% tested
- Methods : 80% tested
- Lines: 80% tested

This above results shows that the all the classes present in the project are well tested. 80% of the method in the whole project is tested. Remaining other methods comes across the UI testing which is not done in this project. The most significant figures that matters a lot is number of lines tested in the whole project and here almost all the lines are tested only except the code line for the UI and Integration. So these figures concluded that the TDD for this project was a successful approach.

## REFERENCES:

- *Google Maps: https://developers.google.com/maps/documentation/javascript/tutorial*
- *Google Places: https://developers.google.com/maps/documentation/javascript/places*
- *Google GeoCoordinates: https://developers.google.com/maps/documentation/geocoding/start*
- *Autocomplete: https://jqueryui.com/*
- *Weather: https://openweathermap.org/api*
- *Markers: https://developers.google.com/maps/documentation/javascript/markers*
- *Info Window: https://developers.google.com/maps/documentation/javascript/infowindows*
- *MySQL Documentaion: https://dev.mysql.com/doc/refman/5.7/en/*
- *JUNIT: http://junit.org/junit4/javadoc/latest/*
- *Spring Boot Initializer: https://start.spring.io/*
- *Spring Boot Documentation: https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/*