B. TECH. PROJECT REPORT

On

Hotel Data Gathering for a Simple RMS

By

Shubham Burewar

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2017

Hotel Data Gathering for a Simple RMS

A PROJECT REPORT

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

of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

Submitted by:
Shubham Burewar

Guided by:
Dr. Somnath Dey,
Assistant Professor,
Discipline of Computer Science & Engineering,

INDIAN INSTITUTE OF TECHNOLOGY INDORE
November 2017

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"Hotel Data Gathering for a simple RMS"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science & Engineering' completed under the supervision of **Dr. Somnath Dey, Assistant Professor, Discipline of Computer Science & Engineering, IIT Indore** and **Mr. Ashish Parkhi, 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.

Shubham Burewar

CERTIFICATE by BTP Guide

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

Dr. Somnath Dey,

Assistant Professor,

Discipline of Computer Science & Engineering,

IIT Indore

Preface

This report on “Hotel Data Gathering for a simple RMS" is prepared under the guidance of Dr. Somnath Dey, Assistant Professor, Discipline of Computer Science & Engineering, IIT Indore and Mr. Ashish Parkhi, Senior Manager at IDeaS.

Through this report I have tried to give a detailed design of the web application that was developed during my internship period at IDeaS to test the feasibility of a simpler Revenue Management System.

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.

Shubham Burewar
B.Tech. IV Year
Discipline of Computer Science & Engineering
IIT Indore

Acknowledgement

I would like to express my deepest appreciation to all those who provided me the opportunity to complete this project. I would like to give special gratitude to my BTP project supervisor, Dr. Somnath Dey, Assistant Professor, Discipline of Computer Science & Engineering, IIT Indore.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of IDeaS, who gave me the opportunity to work for the company. Special thanks goes to my team mate, Surendra Chouhan, who worked with me on this project. Last but not the least, many thanks to the manager of the project, Mr. Ashish Parkhi, and my guide, Mr. Chaitanya Deshmukh who invested his full effort in guiding the team in achieving this goal. I appreciate the guidance given by other supervisors as well as the comments and advice given by the panel during our presentation.

Shubham Burewar
B.Tech. IV Year
Discipline of Computer Science & Engineering
IIT Indore

My Contributions

In this project, I created the basic skeleton of the Spring Web application using Spring Boot and Gradle. I also created the complete UI for the main landing screens as well as the UI for the Dashboard and the Properties screen. I was also responsible for the switch from simple CSS to LESS preprocessor to get rid of duplication. The dynamic refresh on radius change for the Dashboard was also added by me. I created the Material Design style cards UI for the Properties page.

I also implemented the Spring Security module and added the registration feature along with the login feature using the BCrypt encoding for passwords which exploits the full 128-bit salt space. I implemented the REST controllers for all the request mappings within the application.

I, then added the hotel occupancy data feature on the Properties page. I also implemented Google Maps API for the Dashboard page and the integration of the competitor list with the map. In the end, I also created a dump file from the database so that application can be easily deployed and hosted on a server such as Jenkins.

List of figures

Contents

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Today, small and medium sized hotel businesses do not use any form of Revenue Management System for two main reasons –

Either they do not have the resources required for an RMS to give meaningful output, or they simply cannot afford one. So, if we were to offer IDeaS solution to said hotel businesses, we will need the client hotel's information such as geographic location, star rating, reputation etc. We can then use this to get appropriate competitor hotels in close vicinity of the client hotel. Now, 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.

Hence, we had to explore the probable sources of information described above which can provide us with the hotel's master data and the hotel's daily room rate data. This application can then enable small and medium sized hotels businesses to start getting pricing decisions by analyzing the data described above.

Our objective was to 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 competitors in a given range and their daily room rates which can be used for determining the room rates for clients' hotel.

1.2 ABOUT THE COMPANY

IDeaS – A SAS Company

Integrated Decisions and Systems, Inc. (IDeaS) is a private company founded in 1989, headquartered in Minneapolis, MN. IDeaS Pune is a major development centre 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 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
- SaaS Applications
- Lodging
- Hotels
- Smart Spaces

1.3 PROBLEM EXPLANATION

There are considerably many hotels in the world categorized by star ratings or reputations. A hotel and its competitors can be found using geographical details of the client hotel and then, for these competitors, we can find the room rates for the present day or the next n days.

This data can then be used by analysing the prices of the competitor hotels for a particular day using which, we should be able to provide optimal pricing rate for the client hotel. So that it can generate better revenue.

To do so, I have to explore how to get the data for these hotels. This basic data of the hotels is referred to as the 'master data' for that hotel. This 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, its coordinates etc. Along with the master data I also have to find a way to get the room rates of the hotels for next 60 days at least.

To get the hotels' master data I need to explore the possible sources and the type of data that will be used in the application. The same is true for the daily room rates.

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 in accordingly, greater price.

Reputation:

The review given by the customers indicates the reputation of the hotel. It shows how the hotel stands in terms of its public image.

1.4 CURRENT STATE-OF-THE-ART

The current products provided by the company are used by hotel chains like Hilton and Marriott, which is an extensive Revenue Management System that provides a whole array of facilities like occupancy predictions, room rates predictions, demand predictions etc. These results are provided by extensive analytics on the historical data provided by the client [1]. A major reason for this RMS not used by smaller scale hotels is that they do not have this historical data that must be provided for the system to work. The expensive nature of these systems is another reason why very few hotels currently use an RMS.

Small and Medium sized hotels make up a large part of the hospitality business. In order to introduce a Revenue Management System culture to these businesses, a new product needs to be devised, which is affordable, and does not require historical data in order to work. On top of this, it needs to be simpler to use than present RMS, which are quite complicated and require a few weeks to adapt.

Let's take an example. Consider a client hotel in Chicago. It can be located geographically which will enable us to find its competitors in a given radius and collect their data along with the client's. Once we locate the competitors then we should be able to collect their room rates for future 60 days. Using these rates we should be able to determine what the client should charge for today. And this will help them to get better revenue.

This should work for all hotels in the world which have some star rating and geographical location. To do so, We need to explore how we can get the master data for the hotel and how we can get the daily room rates for the hotels for next 60 days.

1.5 OBJECTIVE

The aim of developing this application is to evaluate the feasibility of a simpler Revenue Management System that will work for hotels that do not have historical data which is necessary for a current state-of-the-art RMS. To develop this application, we'll first need to find a source from which we can gather client hotels' data. These sources can be online

websites/APIs/ local database. The gathered data will be in the form of the hotels' master data (basic information), their competitors in a given range and their daily room rates for up to the next 60 days which can then be used for determining the room rates for client's hotel.

1.6 MOTIVATION FOR THIS PROJECT

As we can see in Figure 1.1 given below which shows present-day RMS usage stats, less than 10% of the hotels in the world use any form of Revenue Management System (RMS) [2]. The other 90% do not have an RMS due to various reasons:

- They cannot afford an RMS.
- They do not have the necessary data for such RMS (at least one year of booking history of the hotel is required for some systems).
- They are not aware of the RMS culture and the impact it can have on their revenue.

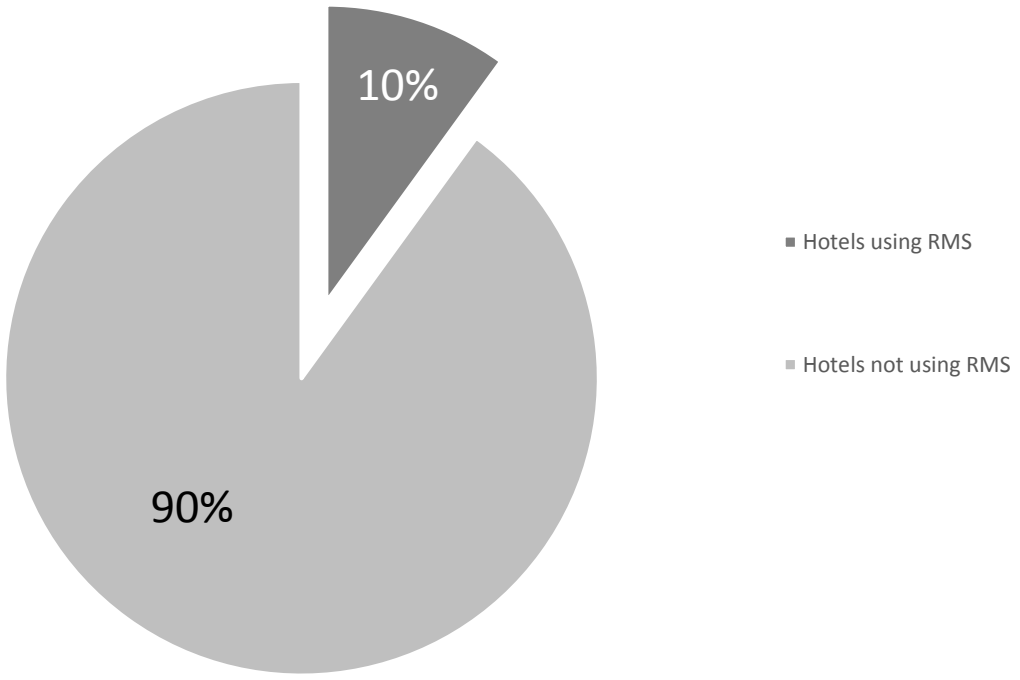

Figure 1.1 Usage of RMS in the hotel industry today

If we were to provide a solution to these hotels, then we need to come up with a product which has the following features that are crucial for an RMS for the aforementioned:

- It must be affordable.
- It should without the need of any historical data.
- It should be convenient and straightforward.
- It should have a complicated interface.

The most challenging factor here is to get rid of the need of data from the client hotel. In this case, we'll need to look at the hotels in close vicinity to the client, i.e. their competitors. By analyzing the pricing of the competitor hotels, the client hotel can be provided with better decisive prices to make better revenue, and introduce RMS culture to small and medium sized hotels across the globe.

CHAPTER 2
DEVELOPMENT APPROACH

To develop this application, I have used the ‘Agile Methodology’, which comprises of small incremental additions and refactoring, with each refactoring building on previous functionalities. To support this agile development model, I have used the ‘Test Driven Development’ (TDD) practice.

2.1 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 completed 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
- Acceptance Testing

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

2.1.1 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 begun early in the software development and became popular as time passed due to its flexibility and adaptability.

The principles of Agile Manifesto given below were followed and practiced throughout this project –

- **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. Since, this project was developed with an aim to release to the audience described in section 1.5, the managers in charge were asked to collaborate.
- **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

Agile methods are being widely accepted in the software world recently. However, this method may not always be suitable for all products. Still it has become the top choice as a

development model for newer businesses. 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.

2.2 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. The cycle followed in Test Driven Development is described in detail on the following page –

Test Driven Development Cycle:

i. Add a test

In test-driven development, each new feature begins with writing a test. We 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.

ii. 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 behavior 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.

iii. Write the code

The next step is to write some code required for 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.

iv. 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 (see step 5).

v. 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.

vi. Repeat

Starting with another new test, the cycle (steps i. to v.) 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

In the following sections, I have explained the design of the application and the overall flow of the application starting from the hotel search at the beginning to the predicted price at the end of the flow. The flow described below comprises of all stages of the application, some of which were not a part of this project. Also, the said flow of the application is elaborately illustrated in Figure 3.1 and described below.

3.1 FLOW OF THE APPLICATION

The complete flow of the application shown in Figure 3.1 is described in the following subsections –

3.1.1 Search Client Hotel

The first step in the application flow is to find out the hotel of the client from the database. In the first few iterations, we implemented 3 sources for this search –

i. Google Places API
ii. Expedia Static Data
iii. Travel Payouts Static Data

3.1.2 Master Data for Client Hotel

In the final iteration of the application, we used the Expedia Static Data stored in the MySQL database, which gives us the complete name of the client's hotel, its address, coordinates, rating etc. The database also has other data like City, Country, IATA codes etc. which might be useful in the future.

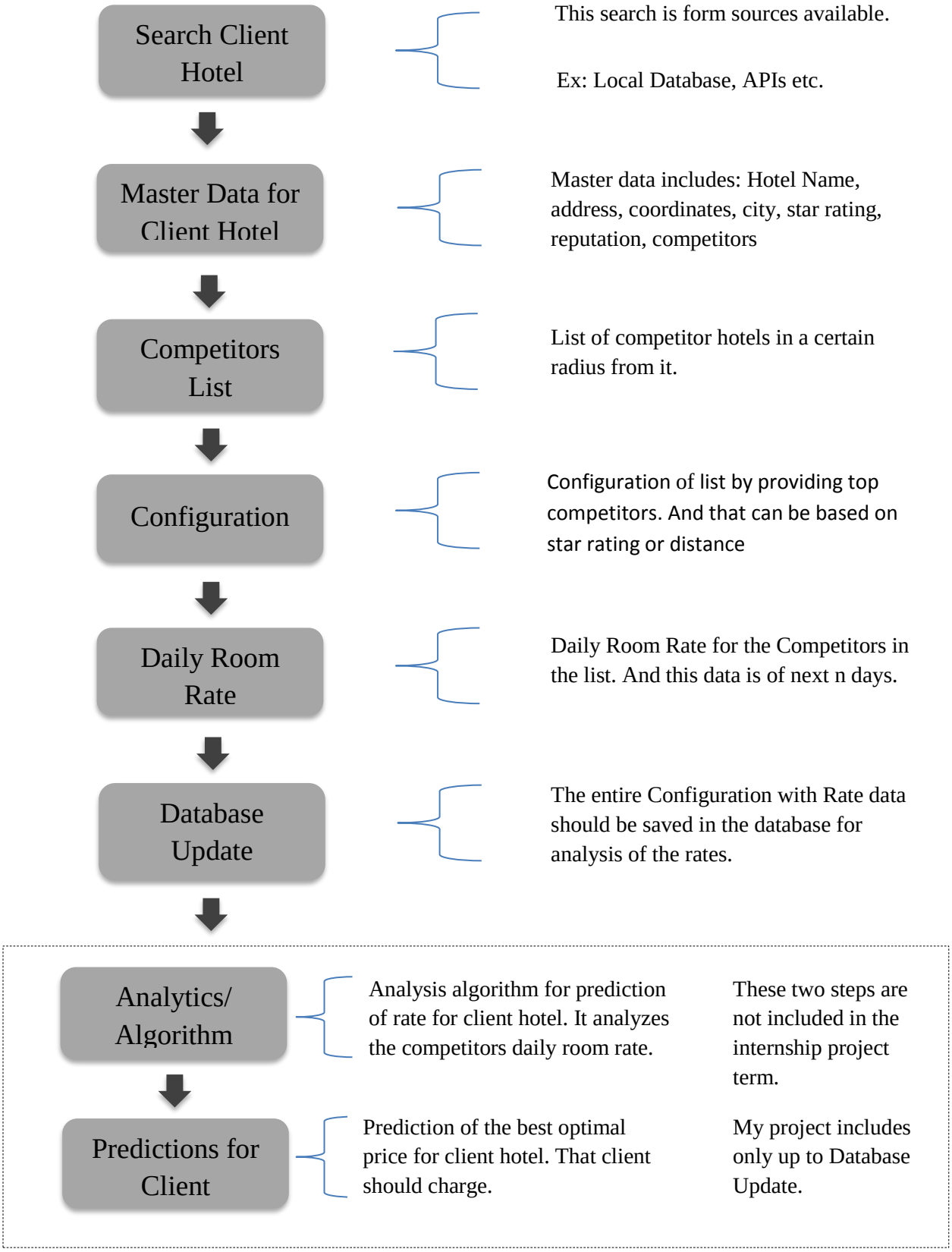

Figure 3.1 Basic flow of the application

3.1.3 Competitors List

Using the details of the client hotel accessed from the database, we obtain a list of competitor hotels in a given radius of the client hotel (default value is 1 km). These competitors are displayed as a list as well as marked on a Google Map.

3.1.4 Configuration

When the user has obtained a list of competitor hotels around the client hotel, the client can then configure the obtained list to get a more precise data set. For example, the user can constrain the list to a range of hotel rating such as hotels with rating between 3-5, or change the search radius such as hotel within 0.5 km only (The search radius can have a minimum value of 0.1 km). The user can select any number of hotels form this list to be configured for getting the rates.

3.1.5 Daily Room Rates

The rates for the hotels configured by user in the above step are then collected. We had planned to use a Rate-Shopping Vendor for getting the rates of the hotels but unfortunately, we weren't able to get a confirmation from the vendor, so we have implemented this step using dummy data stored in the local database.

3.1.6 Database Update

The configuration created by the user in step 4 can be saved to the database, in order to avoid configuring the list every time the user logs in. This saved configuration will be used directly to give the room rates, without having to go through step 4 again and again.

3.1.7 Analytics/Algorithm and Predictions

This part of the application was not a part of the project assigned to us during our internship. It will be added later on by the Analytics team. We were instructed to complete the application up to step 6 only, which is why the final iteration of the application is only up to here.

CHAPTER 4
TECHNOLOGY STACK

The different technologies and resources that we have used in this project have been categorically listed below followed by brief descriptions of each of them. The back end here consists of the database (which is a MySQL database in this case), the APIs used as well as the Spring controllers, whereas the front end consists of the client side source that mainly comprises of the HTML pages and JavaScript script files.

- **Database**
    - MySQL
- **APIs**
    - Google Geocode API
    - Google Places API
    - OpenWeatherAPI
- **Java**
    - Spring Web
    - Spring Security
    - Spring JPA
    - JDBC
- **Front End**
    - HTML
    - jQuery (JavaScript)
    - LESS preprocessor (for CSS)

- **Testing**
    - Junit
    - Mockito
- **Build Automation**
    - Gradle

The project has been developed using Spring framework for Java EE applications and Gradle as the build automation system to simplify the development of the web application.

4.1 BACK END

The backend of the application is written in Java using Spring Boot module [4] for Java EE. The MySQL database which store all the user and hotel data is accessed using Spring JDBC (Java DataBase Connectivity) module. The user authentication is done using the Spring Security module using JDBC authentication and BCrypt encoding in order to provide practicable safety from exploitation with minimum code. Passwords have been encrypted using the BCrypt which is one of the more secure hashing functions used today (There are vulnerabilities found in the MD5 and SHA hashing functions which why they are deprecated).

4.1.1 APIs

Using APIs provided by OTAs (Online Travel Agencies) for hotel and competitor data was the initial goal of the project. Unfortunately, we were not able to get required permissions for this since the OTAs require details of the application such as expected revenue generation which is why we had to look at alternate sources like Static Data stored in a database. Regardless, we have used a number of APIs in the application nonetheless, such as the Google Maps API for a graphical representation of the competitor list, the Google Geocode API for marking the hotels on the map, OpenWeatherAPI for showing the current weather data of the hotel vicinity, and the jQuery UI API to effectively use jQuery to execute features such as search autocomplete.

4.2 FRONT END

The frontend of the application consists of the code that runs on the client side and mainly consists of code written in HTML and jQuery (JavaScript), the Bootstrap framework, and the LESS pre-processor for CSS.

4.3 SOURCES

We have tried and tested three sources to gather the master data for the client hotels (which will be reduced to only one source in the end).

i. **Google Places API:** It is an API that provides access to information about more than 100 million places around the World. It provides details about a place such as:

- Name
- Address
- Latitude
- Longitude
- Rating
- Type
- Unique Id

Upon entering the coordinates of the client which are found using the Google Geocode API. This allows us to locate the nearby places of similar type in a given range. This helps in locating the competitor hotels for the client hotel and provides the details for all the competitor hotels.

ii. **Expedia Static Data:** This Online Travel Agency (OTA), helps aids in booking hotel reservations worldwide. It has database for the hotels across the globe, which is regularly updated. The hotel data obtained from here is elaborate and

abundant. Some of the more important and useful details that are available here are:

- Hotel name
- Address
- Star Rating
- Latitude
- Longitude

We also get other details which are not relevant to this project. This database has 2,33,224 hotels record and this helped in gathering master data for the client hotel. **This is the source that we decided to use in the final iteration of the application.**

iii. **Travel Payouts Static Data:** This is also an Online Travel Agency just like Expedia and it has similar database. It has 16,74,389 hotels records.

Since Expedia and Travel Payouts data is static, so these two databases are stored locally on the machine using MongoDB (non-relational database) (This was changed to MySQL to facilitate relational schema). The search can be done using either of these sources.

4.4 COMPETITORS

List of the competitor hotels in a given radius can be generated from the source which was used to gather the master data for client hotel. The list will contain up to 20 hotels around the client hotel. The client can then configure his/her competitors list by selecting the hotels from the list. Google Places API provides the 'nearby search' facility which helps in locating the competitor hotels. For Expedia and Travel Payouts database, we calculated a circle with the coordinates of the hotel as the center and the radius as given by the user (1 km is the default value).

This query gives a list of hotels that are within a given radius in meters from the given latitude and longitude.

4.5 ROOM RATES FOR COMPETITORS

For getting the room rates of the competitors, we were hoping to get a Rate Shopping Vendor to provide a reliable source of rates which is collected city wise. Unfortunately, we weren't able to get an answer in time so we decided to run the application with dummy data stored on our MySQL database for the time being. A Rate Shopping Vendor can be added in the future to the application.

4.6 ANALYTICS/ALGORITHMS

The analytics/algorithms involved in this application are not a part of this project, mainly because the company has an analytics team that works that domain.

In a nutshell, once the room rates for competitor hotels are gathered then these can be run through an algorithm which will give some meaningful pricing prediction that can be used by client hotel to make better revenue [5]. In other words, these are the minimum rates that client hotel should charge.

CHAPTER 5
RESULTS

5.1 HOME SCREEN

The home page for the application is as shown in Figure 5.1. 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. We have tried to keep the home page as simple and as minimalistic as possible.

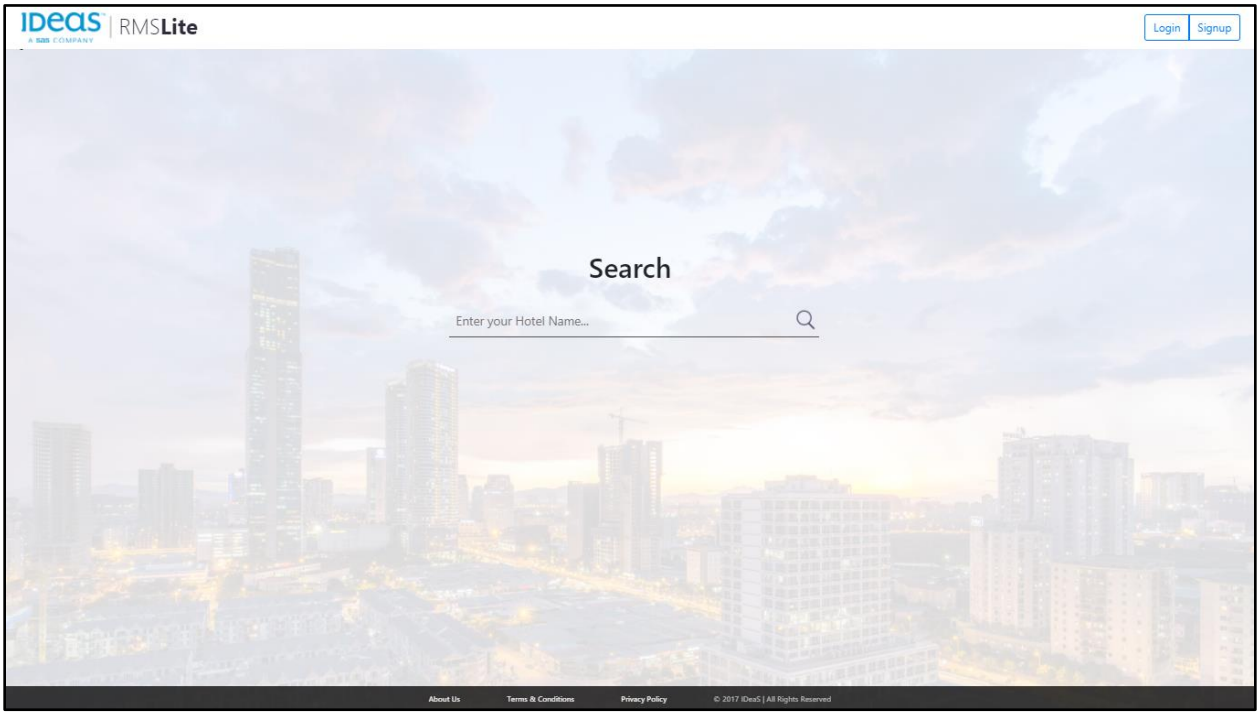

Figure 5.1 Screenshot of the home page

5.2 LOGIN AND REGISTRATION PAGE

The login and registration pages seen below in Figure 5.2 and Figure 5.3 respectively, are also created similar to the home page, i.e. as simple as possible. In case, of any loss of connection or unintentional timeout, the Spring Security module will redirect the user to this login screen. The Login screen and the registration screen also contain links to each other for easier navigation.

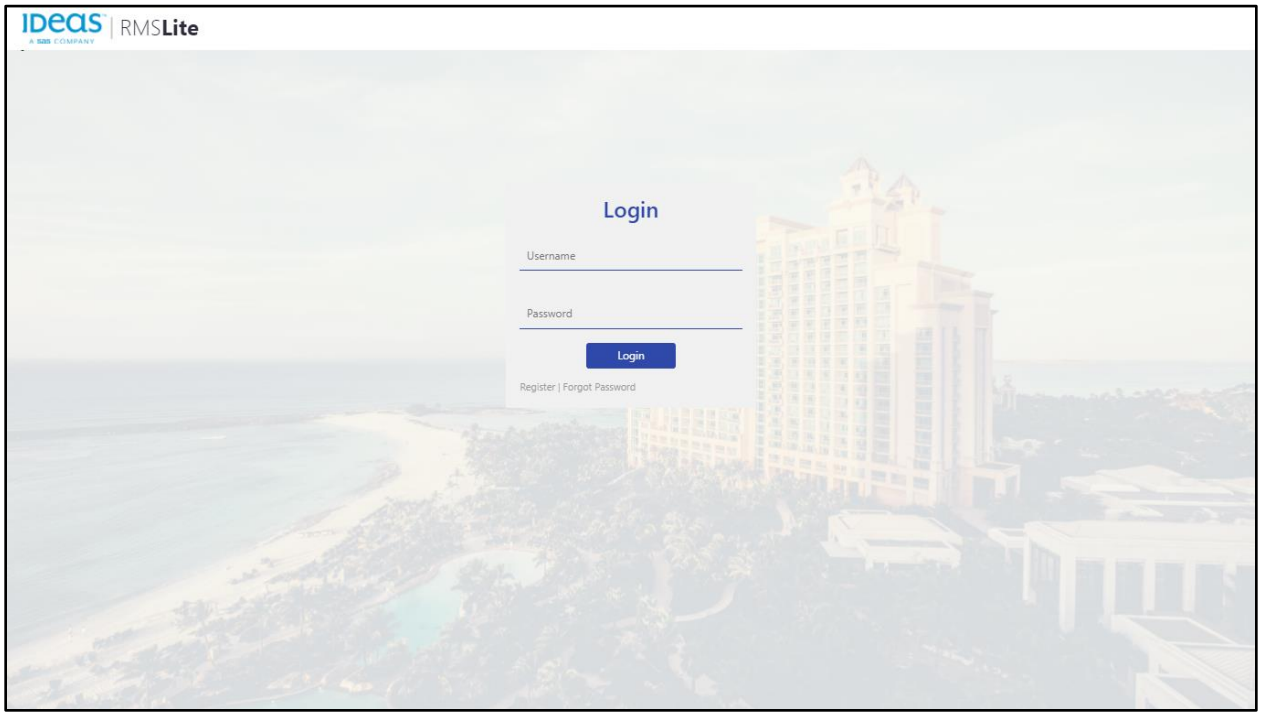

Figure 5.2 Screenshot of the login page

These pages use the features of Spring Security module of the Spring Boot framework. Spring Boot has predefined security settings for the web application. It also includes authentication, registration, login, logout and sessions.

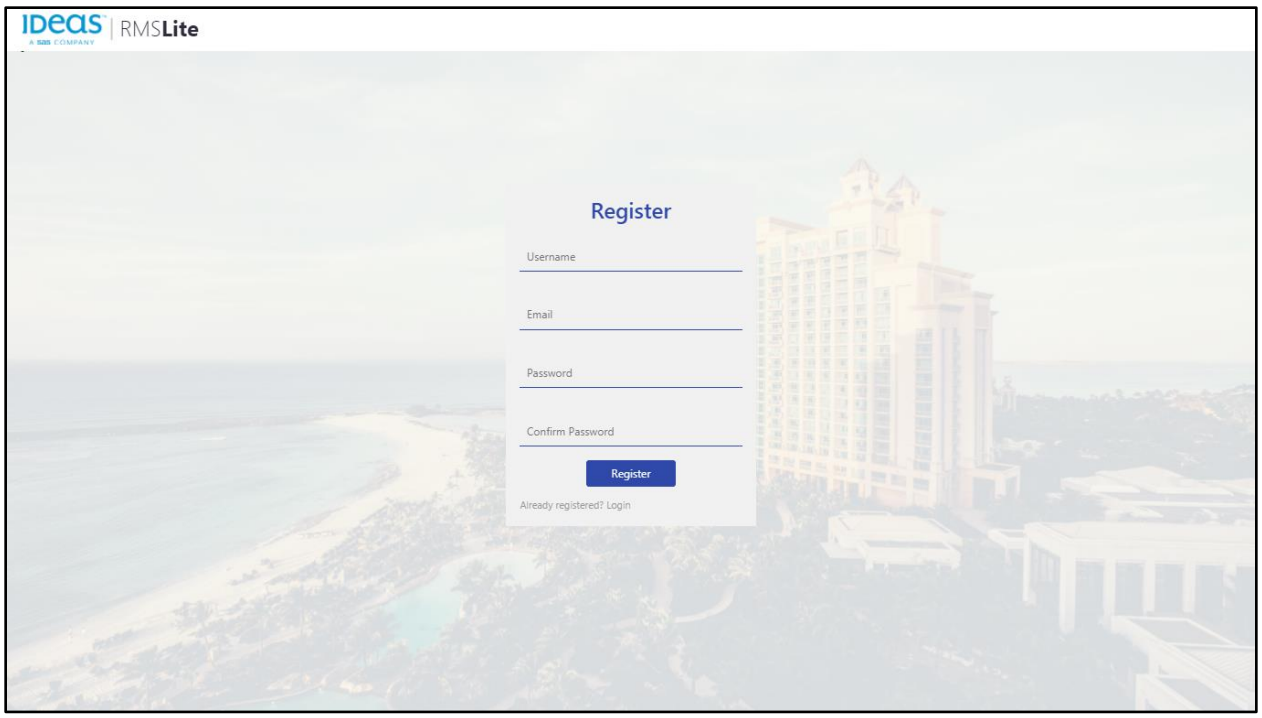

Figure 5.3 Screenshot of the register page

5.3 USER PROPERTIES SCREEN

The user properties page, as seen in Figure 5.4, is 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). The add option provides the hotel search function with the auto complete feature similar to the one on the home page. 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.

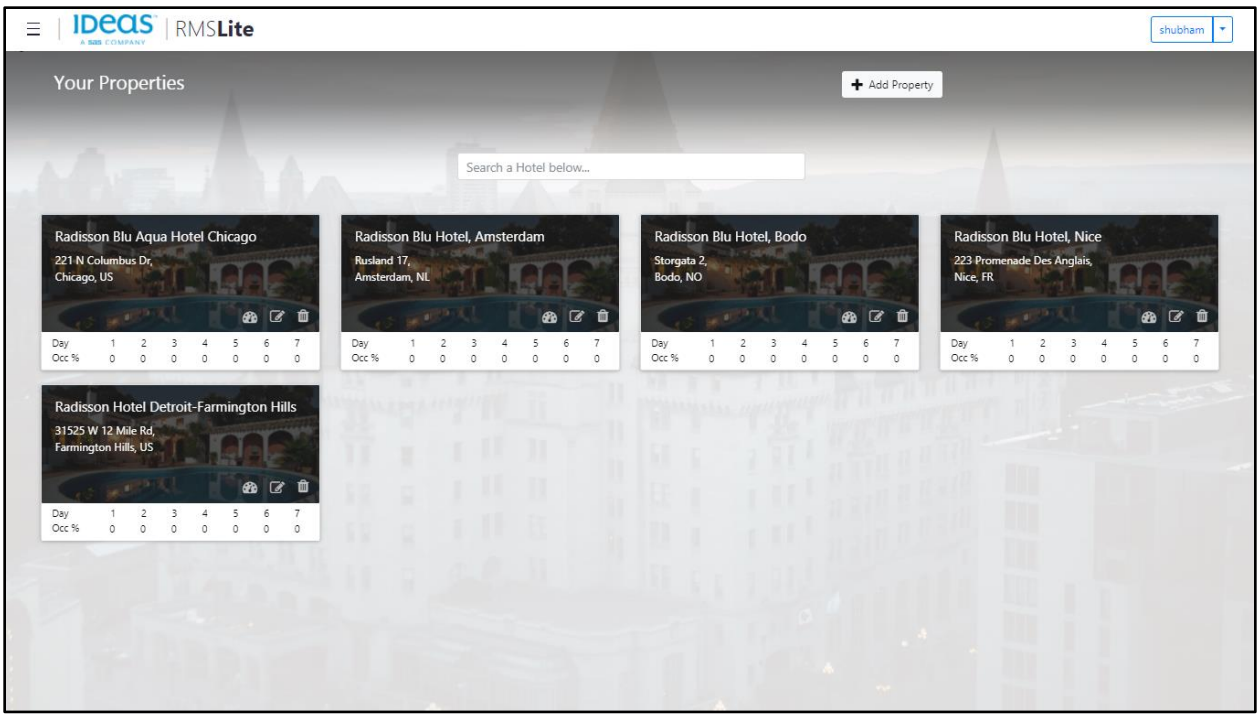

Figure 5.4 Screenshot of the user properties page

Here, each hotel is shown as a Material Design style cards, with a table for entering occupancy data, and three buttons which are for going to the hotel dashboard, saving the entered occupancy values, or deleting the hotel from the user page. We have also added a search bar for the user to search for a hotel among the cards in case if the number of hotels for one user is very large in which case, it could become difficult to find a particular hotel in that card stack.

5.4 DASHBOARD SCREEN

Figure 5.5 shows 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 dropdown 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.

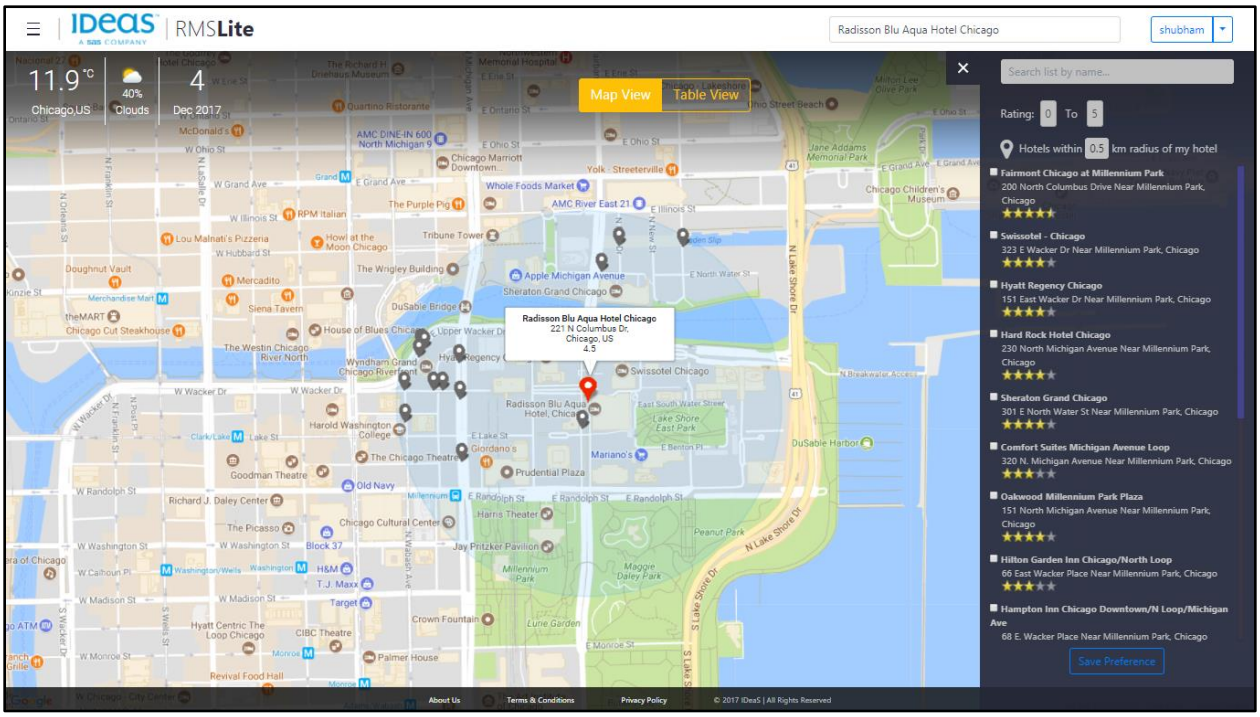

Figure 5.5 Screenshot of the dashboard page

5.5 TABLE VIEW REPRESENTATION ON DASHBOARD

As seen in Figure 5.6, We have added table view which shows the rate of the client hotel along with the competitor hotels selected in the list. 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 days count start from the present day. User can switch between map view and table view by using toggle buttons on the top.

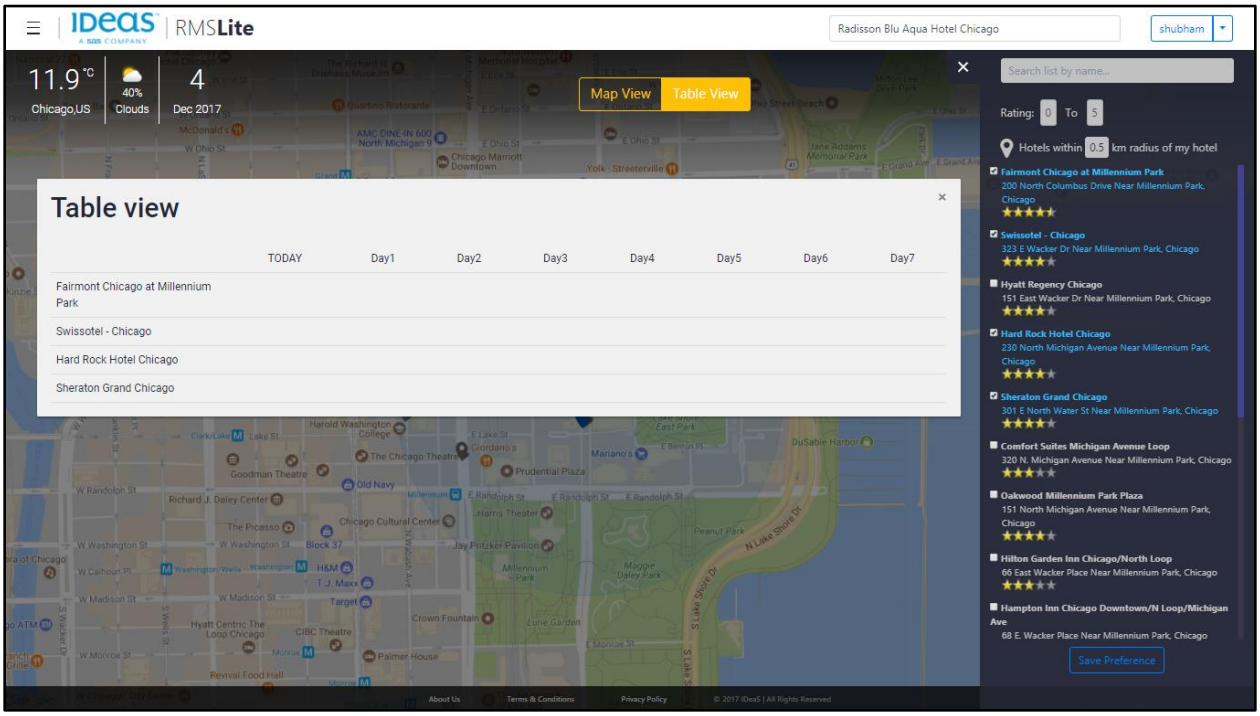

Figure 5.6 Screenshot of the table view on the dashboard

5.6 CODE COVERAGE RESULTS FOR TEST CASES

Figure 5.7, shown below is a screenshot of the IDE showing that all the test cases that were run had all passed. It also shows the result for the code coverage. Code coverage shows how much of the total code is tested using the test cases that have been created. We can see in the screenshot that there are total of 33 test cases. There are no test cases for User Interface Testing. **These 33 test cases have covered 100% classes, 80% of the methods and 80% of the code lines.**

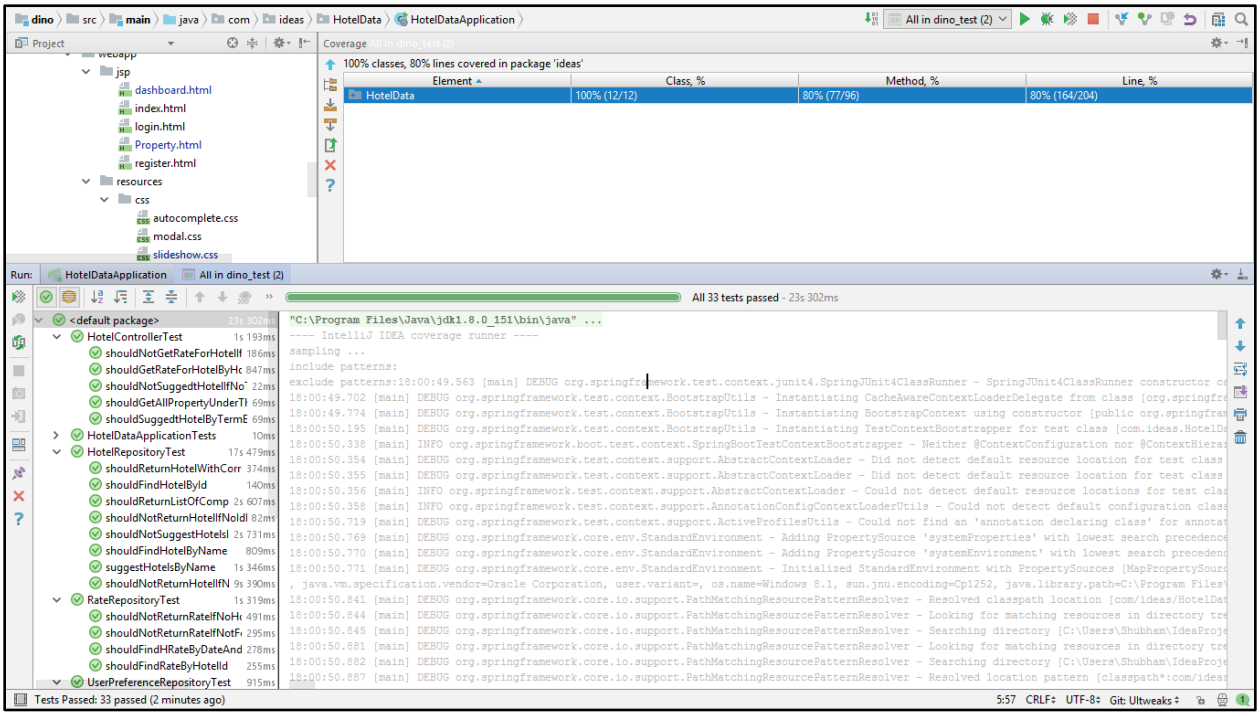

Figure 5.7 Screenshot of the IDE showing all test cases passed

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

We have developed a simpler Revenue Management system that works without historical data using Spring Web framework for Java Enterprise development. It is a multi-tenant system which means one user/client can have multiple properties in this system. The project was developed using agile methodology, wherein we participated in scrums, where we concluded what we did the previous day and planned on future work. In other words, this project was an excellent aid in understanding and acclimate ourselves with software development as it is practiced in the software industry today.

The entire application was developed using the Agile methodology of software development to aid in faster development and refactoring cycles and using the Test-Driven Development cycle to ensure maximum code coverage and minimum code duplication. The results for code coverage are as follows:

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

These results show that the all the classes present in the project backend are well tested. 80% of the methods in the project are tested. Remaining other methods consists of UI for which, testing has not been done. The number of lines tested in the project is a significant number here and as we can see above 80% of the lines are tested, the exceptions being the UI and Integration code. Hence, Test-Driven Development was a successful approach for this project.

This project was regarding the development of a part of application. So, after our work is done, this project will be integrated with the appropriate analytical algorithm which will help the application to give accurate predictions based on the data that we have gathered.

The elimination of the need of historical data will enable small and medium scale hotel businesses to have an affordable option to embrace the Revenue Management culture, which is becoming increasingly common in the hotel industry.

Since, this application is meant for a large market, we need to make sure that the application can handle a lot of traffic. Hence, performance needs to be optimized in order to smoothly run the application on a great number of users simultaneously.

This web application requires a Rate Shopping Vendor to provide authentic rates for the application to run in production. Also, the registration procedure has not been implemented in the application, because we were not yet sure which form of Spring registration needs to be implemented here that will work with JDBC authentication used in the application.

References

[1] A. Sulistio, K. Kim and R. Buyya, "Using Revenue Management to Determine Pricing of Reservations - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2017.

[2] Kimes, S. E. (2010). The future of hotel revenue management [*http://scholarship.sha.cornell.edu/cgi/viewcontent.cgi?article=1068&context=chrpubs*]. *Cornell Hospitality Report, 10*(14), 6-15.

[3] Experiment about Test-first programming**,** *Matthias M. Muller and Oliver Hagner, University of Karlsruhe, Germany*

[4] Spring Boot Documentation: *https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#getting-started-introducing-spring-boot*

[5] M. Lee, Y. Fu and K. Lai, "A Study on Revenue Management of a Service Industry - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2017.