DEVELOPMENT OF CERTAIN WEB APPLICATIONS

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:
Bandaru Harsha Vardhan

Guided by:
Dr. Surya Prakash

COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE December 2018

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Development of certain web applications"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering' completed under the supervision of **Mr. Prasad Kunte, Senior Manager, IDeaS A SAS Company, Pune and Dr. Surya Prakash, Assistant Professor, Computer Science and Engineering,** IIT Indore is an authentic work.

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

(BANDARU HARSHA VARDHAN)

DATE: _____

CERTIFICATE by BTP Guide(s)

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

Dr. Surya Prakash

Assistant Professor

Computer Science and Engineering

IIT Indore

Preface

This report on “Mars and CPRO Projects” is prepared under the guidance of Dr. Surya Prakash, Assistant professor, Discipline of Computer Science & Engineering, IIT Indore and Mr. Prasad Kunte, Senior Manager at “IDeaS A SAS Company”.

Through this report I have tried to give a detailed of work during my internship period at “IDeaS A SAS Company” from my start of day to end of my internship.

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.

BANDARU HARSHA VARDHAN
150001003
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

Acknowledgements

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. Surya Prakash, Assistant Professor, Discipline of Computer Science & Engineering, IIT Indore.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of "IDeaS A SAS Company ", who gave me the opportunity to work for the company. Special thanks to the manager of the project, Mr. Prasad Kunte, and my guide, Mr. Shivani Kaul 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.

BANDARU HARSHA VARDHAN

150001003

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

My Contributions

In Goa Picnic Web Application, I have planned, designed and developed Java based web development using Maven in Eclipse.I also normalized the database by applying 1st Normal Form,2nd Normal Form,3rd Normal Form,3.5 Normal Form during designing the database.I developed the website using TDD(Test Driven Development) method using JUnit 4.12.

In Mars, I also wrote test cases in JUnit for the data points which hold the base business logic and which gives the guarantee of correctness of data which is the most important factor of success of the product.I have wrote MongoDB queries to insert and extract the data from the database.

In CPRO, I have preprocessed the data and analysed the data using Python analytical tools and found few of the missing features in the data which will help the data scientist to analysis and predict more accurately.

Contents

List of Figures

Chapter 1

Introduction

1.1 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 reached the 10,000 customer benchmark in 2018. And It has been 10 years under SAS company.

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

1.2 Agile Software Development Model

Company follows 'Agile Methodology' of development, which comprises of small incremental additions and refactoring, with each refactoring building on previous functionalities.

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.

1.2.1 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.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.

1.3 Test Driven Development

To support Agile software development method, 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, but more recently has created more general interest in its own right. The cycle followed in Test Driven Development is called the Test Driven Development Cycle:

1.3.1 Test Driven Development Cycle

1. **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.

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 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.

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 (see step 5).

5. **Refactor 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.

6. **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.

1.4 Projects

As part of the internship, I worked on three separate projects:

- Goa Picnic Web Application
- Mars
- CPRO

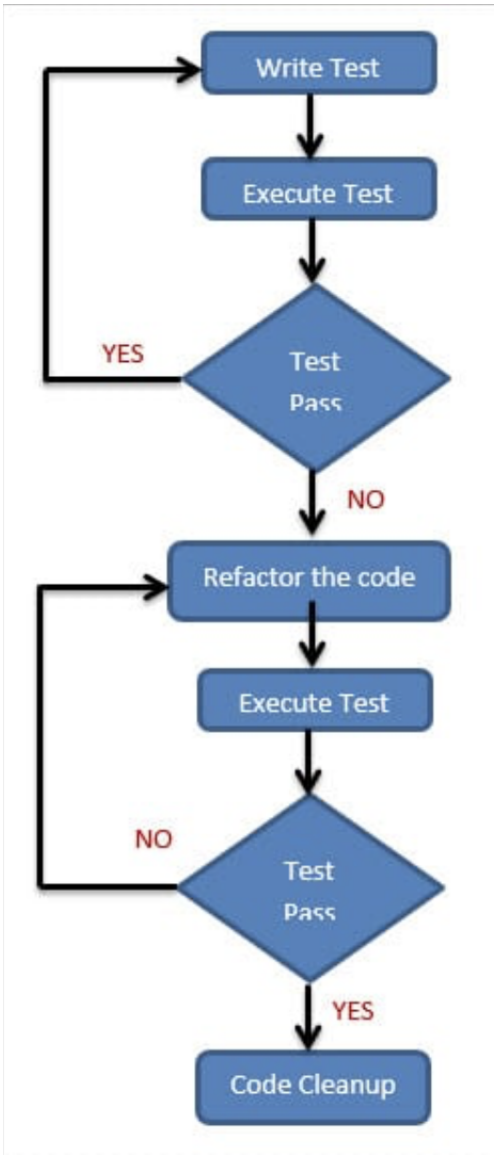

FIGURE 1.1: TDD Flowchart [Source: http://agiledata.org]

Chapter 2

Goa Picnic Web Application

2.1 About

Every year IDeaS takes an annual trip, usually to the Goa.Usually, the process to collect the data of each employee who all are willing to come to Goa is done manually through excel sheet. This causes the issue of data inconsistency along with making the entire process tedious and error prone. The web application was proposed as a solution to the described problems.

2.2 Problem Statement

To make a web application in Java using Eclipse with efficient backend sql database and having two pages. First page contains the list of all employees who all are willing to come to Goa with search bar to search all the employees in the list, dynamic adding and deleting the employee. Second page contains the preference of respective employee with dynamic adding and deleting of preferences. TDD method of development is to be followed during the development.

2.3 Technologies Used

Technologies used during creating this website are following:

- Java 8
- Mysql

- Java Jdbc(Java DataBase Connectivity)
- JUnit(For Testing)
- HTML
- Bootstrap
- Css
- JQuery
- JavaScript
- Maven(For building the project)
- Eclipse(Platform)

2.4 File Structure

- **HTML**: HTML is a HyperText Markup Language file format used as the basis of a web page. HTML is a file extension used interchangeably with HTM. HTML is consists of tags surrounded by angle brackets. The HTML tags can be used to define headings, paragraphs, lists, links, quotes, and interactive forms.
- **Servlet**: A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.
- **Service**: A Service class/interface provides a way of a client to interact with some functionality in the application. This is typically public, with some business meaning. For example, a Ticketing Service interface might allow you to buy ticket, sell ticket and so on.
- **Entity Class**: In general, entity classes may be defined in any way desired by the application. The entity binding, which is also defined by the application, is responsible for mapping between key/value objects and entity objects.

2.5 Entity Diagram

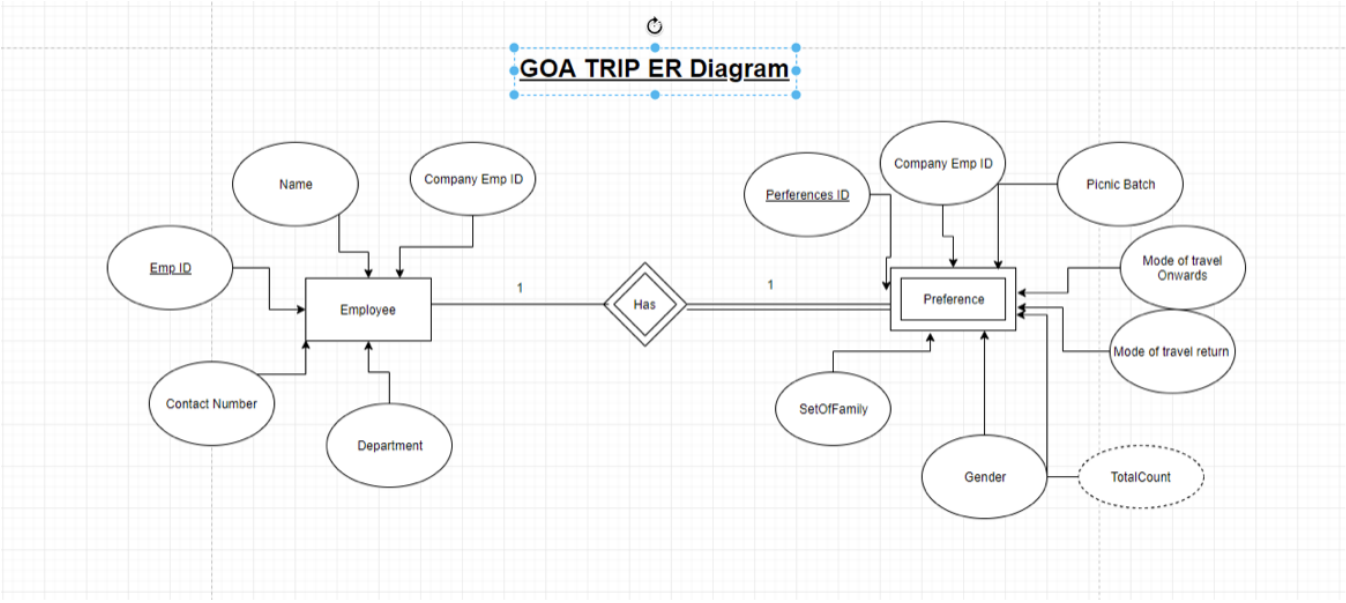

FIGURE 2.1: Entity Relationship diagram of the Goa Picnic Web Application

2.5.1 Description

Database consist of two tables. They are:

- Employee
- Preference

2.5.2 Employee Table

Attributes:

- Employee id
- Employee company Id
- Employee nameET:
- Employee contact
- Employee department

2.5.3 Preference Table

Attributes:

- Preference id
- Employee company id
- Picnic batch
- Set of family
- Gender
- Total count
- Mode of travel return
- Mode of travel onwards

Preference table is a weak entity. It takes employee company id as foreign key from the employee table. During the designing the database, I have normalized the data by careful application of 1st, 2nd, 3rd and 3.5 normal forms.

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

2.6 SQL Codes

2.6.1 Employee Table

```
CREATE TABLE Employee(
    Employee_ID INTEGER PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(100),
    Company_Employee_ID NUMERIC(5) not null,
    Contact_No char(10),
    Department ENUM('SD', 'QA', 'CARE', 'ROA', 'TECHOPS')
);
```

2.6.2 Preference Table

```
CREATE TABLE Preferences(
    Preferences_ID INTEGER PRIMARY KEY AUTO_INCREMENT,
    Employee_ID INTEGER,
    Picnic_Batch ENUM('FIRST_BATCH', 'SECOND_BATCH'),
    ModeOfTravel_Onward ENUM('CAR', 'AEROPLANE', 'TRAIN', 'BUS'),
    ModeOfTravel_Return ENUM('CAR', 'AEROPLANE', 'TRAIN', 'BUS'),
    Gender ENUM('FEMALE', 'MALE'),
    SetOfFamily ENUM('EMPLOYEE_SPOUSE_CHILDREN', 'EMPLOYEE_DEPENDENT_PARENTS',
    'EMPLOYEE_SPOUSE', 'EMPLOYEE'),
    TotalFamilyCount NUMERIC(5),
    FOREIGN KEY(Employee_ID) REFERENCES Employee(Employee_ID)
);
```

2.7 Basic Flow of the Application

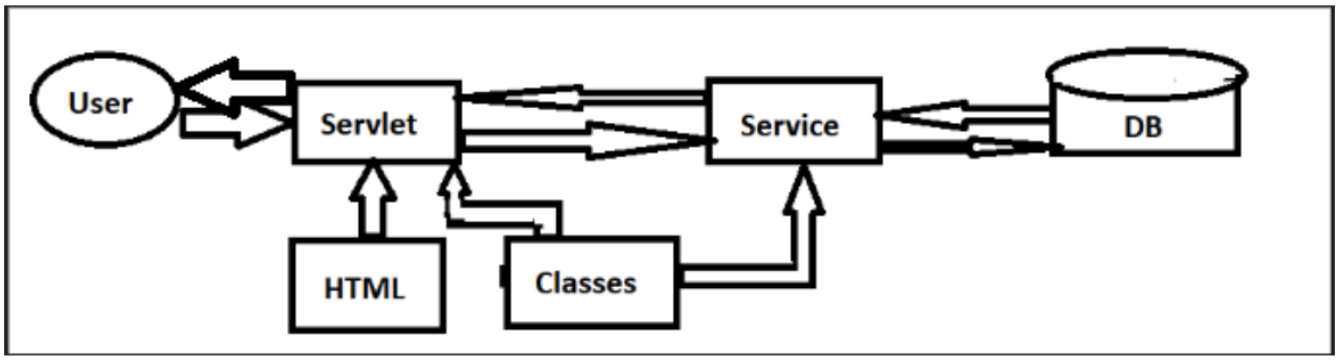

FIGURE 2.2: Flow of the Java based web application

1. User interacts with the website.Website sends the Http request to servlet.
2. Servlet sends the request to service for data from the database.
3. Servlet using Java JDBC connects to database.
4. Based on request, the database response, send the request to service
5. Service uses the classes as framework for the data to pass to servlet

6. Servlet also uses classes as framework for the data and along with Html page servlet send the response to user.

2.8 Application Overview

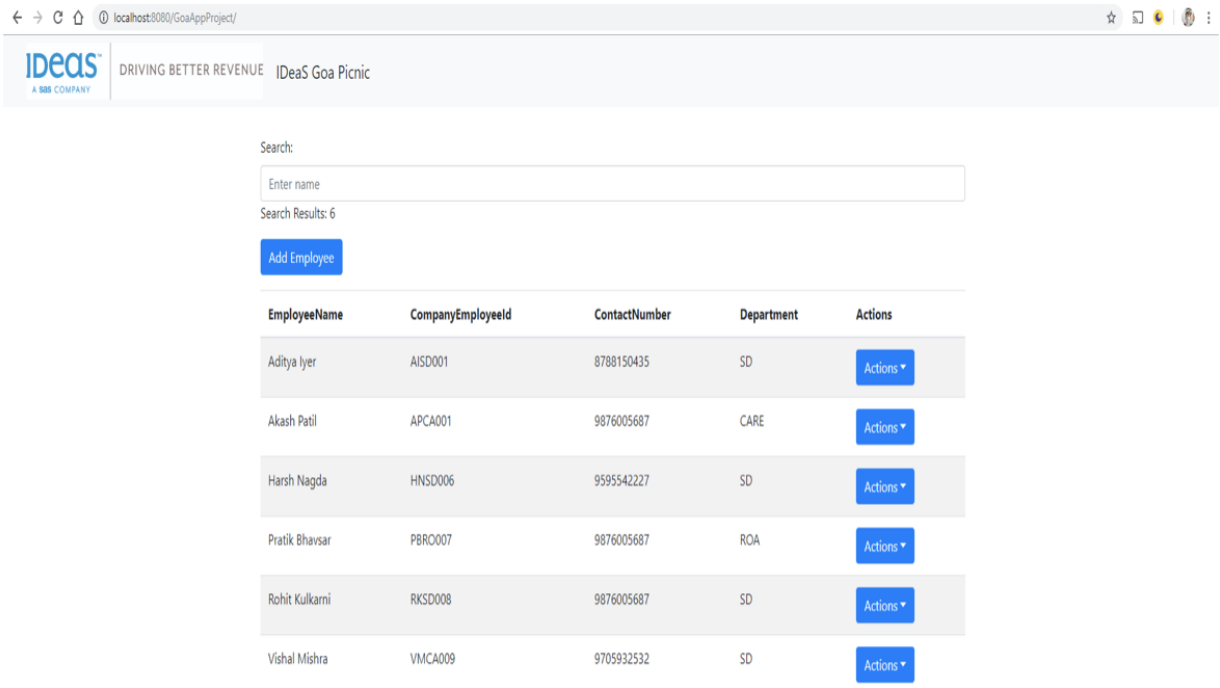

FIGURE 2.3: Screenshot of the first page of the Goa Picnic Web Application

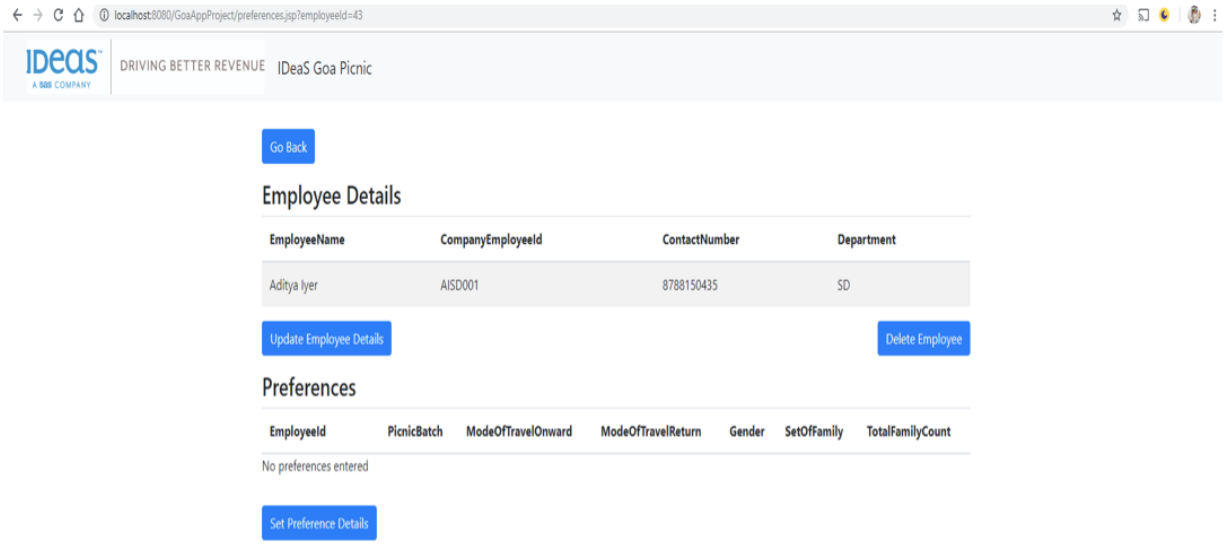

FIGURE 2.4: Screenshot of the second page of the Goa Picnic Web Application

Chapter 3

MARS

3.1 Overview

Every revenue management team has a revenue manager. This is a role that has developed over the years as the hospitality industry has seen the need to use revenue management. At one point the operational teams were in charge of revenue management before the creation of the revenue manager. Now there is a need for a revenue manager that is only focused on the revenue management of the business.

The primary role of the revenue manager is to maximize the businesses' opportunity for revenue and profits. In order to do that, the revenue manager is in charge of compiling and analyzing data to make decisions regarding pricing. The revenue manager compiles data on the business as well as the competition. They keep up with market changes and identify trends.

To do this, revenue managers spend 80% of their time in gathering the data and 20% of their time in doing the above things.To solve the above problem, a product from the IDeaS is initiated. That is Mars.

3.2 About the project

Mars project is mainly focused to aid the revenue managers.Mars project helps the revenue managers quickly gather the data and classify the data. So, revenue managers quick understand and apply complex formulas on the data for planning and budgeting. Simple picture explains what Mars do's:

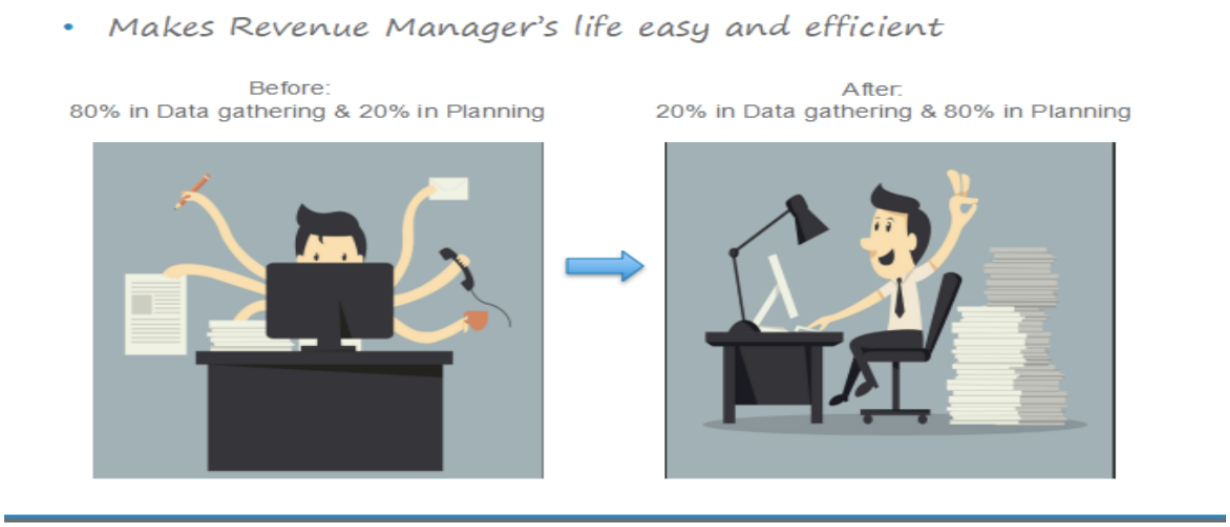

FIGURE 3.1: Life of revenue managers before and after Mars

3.3 Problem Statement

The problem statement contains the following:

- Automate the manual testing work by test cases
- Code the test cases for the data points that holding the base business logic
- Guarantee the data correctness by test cases

3.4 Technologies

Technologies, I used in this project are:

- Junit 4.12
- MongoDB
- Java 8
- Git commands
- Studio 3T
- Robo 3T

3.5 Role

3.5.1 Q.A (Quality Assurance)

Look up QA/QC in Wiktionary, the free dictionary. QA/QC is the combination of quality assurance, the process or set of processes used to measure and assure the quality of a product, and quality control, the process of ensuring products and services meet consumer expectations. Quality assurance is process oriented and focuses on defect prevention, while quality control is product oriented and focuses on defect identification.

3.5.2 Reason

- Deadlines are close.
- Manual testing is time consuming and error prone, as we are adding new features it is difficult to test existing manually.

3.6 Work

- Test cases
- Wrote test cases for the data point that holding the base business logic.
- MongoDB
- Wrote MongoDB queries

3.7 Testing Rules

During Testing, I followed few principles. They are as follows:

- Exhaustive testing is not possible
- Defect Clustering(80% defects in 20% code
- Pesticide Paradox

- Testing shows presence of defects don't show absence of defects
- Absence of Error is a problem
- Early Testing to fix it is easy
- Testing is context dependent

3.8 Data Points

A data point is a discrete unit of information. In a general sense, any single fact is a data point. In a statistical or analytical context, a data point is usually derived from a measurement or research and can be represented numerically and/or graphically. The term data point is roughly equivalent to datum, the singular form of data. I wrote test cases for the data points.

FIGURE 3.2: Some data points

Above highlighted is a data point. The base business logic formula:

$$ActiveForecastADR = \frac{ActiveForecastRevenue}{ActiveForecastRoomNights}$$

3.9 Contributions

In this project, I wrote the test cases for the data points. Few of them are as follows:

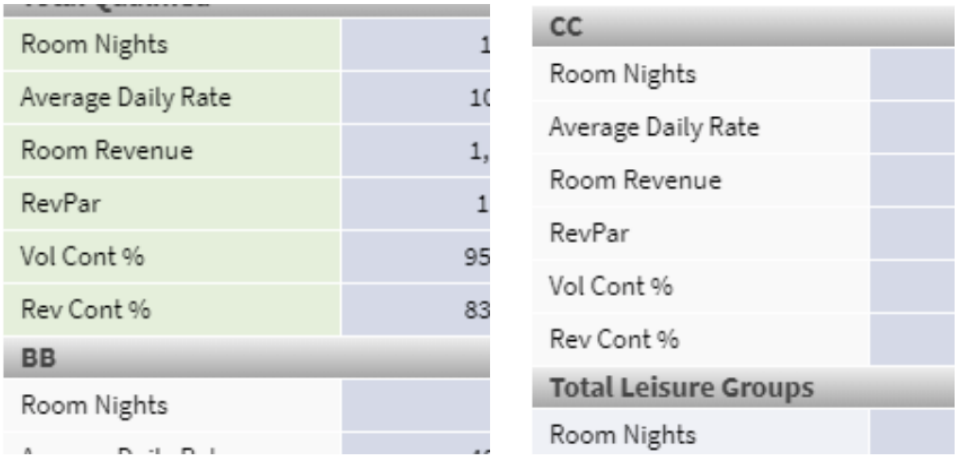

FIGURE 3.3: Example Datapoints

FIGURE 3.4: Example Datapoints

3.9.1 MongoDB

I also wrote MongoDB queries, for example:

```
Db.customers.aggregate ( [
    { $match: {"zip": "90210"}},
    {
      $group: {
         _id: null,
             count: {
           $sum: 1
         }
       }
    }
]);
```

Chapter 4

CPRO

4.1 Overview

According to the Airports Council International (ACI), car parking revenue brings about 20% of total operational (aeronautical and non-aeronautical) revenue for airport hubs of all sizes. In the meantime passenger airline landing fees bring about 18% of total operational revenue. The numbers shift a little bit if you look at Large (19% car parking and 16% airline fees), Medium (15% and 24%), and Small (10% and 26%) airports.

And even though car parking and car rental concessions are still a core revenue stream for airports, they can no longer be taken for granted as a consistent source of business.

4.2 About the project

IDeaS Car Park Revenue Management System dynamically forecasts with a clear view of competitor rate positions, enabling proactive pricing decisions across your locations and helping elevate overall business profitability.

The cloud-based solution equips you to:

- Accurately forecast business demand
- Determine the correct pricing for all car parks
- Assess performance and identify opportunity
- Make strategic decisions with an intelligent view of competitive insights

- Optimize demand and increase revenue across all locations
- Distribute products and prices for all car parks to sales channels in real time

4.3 Problem Statement

Given car park data of an airport, find in the data where revenue can be increase or where is lacking.

4.4 Technologies

- Python 3.7
- Spyder Anaconda navigator v3.3.2
- Scikit Library
- Pandas Library

4.5 About data

Data of the given car park is mainly divided into 3 categories:

- Free
- Drive up
- Booking

4.5.1 Free

These category of people don't pay to carpark.The people that belong to this category are:

- Airport employees
- Other officials

4.5.2 Drive up

These category of people don't buy the car park prior.They buy the car park then there itself.

- Receiver and sender of passenger
- Taxi's
- Immediate planned trips.

4.5.3 Booking

These category of people buy the car park prior.The people belong to this category are

- Business trips
- Passengers who's distance between airport and town.
- Passengers who time of travel is short

4.6 Attributes in the data

Data consist of following columns:

- Car arrival date and time
- Car departure date and time
- Car booking date and time
- Car pre booking arrival date and time
- Car pre booking departure date and time
- Car spend time in airport carpark
- Car owner spend money in airport carpark
- Car cancellation date and time
- Booking channel and its id

4.7 Approach

The following approach was used in the development of problem solution:

- Understand the data
- Find the important features in the data
- Method 1: Apply the Python data analytics tools
- Method 2: Convert the data into time series data
- Method 3: To classify the data into buckets based on elastic coefficient
- Method 4: To find whether any factors are missing or not.
- Analyse the results
- Results

4.8 Understanding the data

To understand the data, the following resources were used:

- Mentor classes about data
- Research papers about car park and factors affecting it.
- Websites

4.9 Features in the data

Features are the factors that affect the price.They are

4.9.1 Non derived factors

- Arrival date, day and time
- Departure date, day and time

- Booking date, day and time
- Pre booking arrival date, day and time
- Pre booking departure date, day and time
- Car spend time in the airport

4.9.2 Derived factors

- Length of stay of car in the airport
- Number of days in advance
- Booking day and time
- Arrival day and time
- Departure day and time
- Cancellation day and time

4.10 Methods

This section describes the four methods I worked tried.

4.10.1 Application of Python Data Analytics Tools

Process

- Preprocessing the data
- Applying the ML algorithms
- Analyzing the results

Preprocessing the data

1. **Cleaning the data**: Cleaning the rows and columns that are not useful.
2. **Applying Encoders**: Label and One-Hot encoders were used.
3. **Normalization**: Performed normalization and scaling of data.

Applying the ML algorithm

- **Linear Regression**: I applied this M.L. algorithm to find what are the features affecting the price and how much they are affecting the price by using there coefficient values.
- **Decision Tree**: I applied this M.L. algorithm to find the first few important factors affecting the price.

Analyzing the results

The results from the linear regression algorithm are not matching with results from the decision tree results. As I found something is wrong, I tried another method.

4.10.2 Time-Series Analysis

Converting the data into time series format to analysis the data. Such as:

- Week wise
- Month wise
- Year wise

Analysis of Result

Analysing the data and found few patterns. They are:

- Monday to wednesday incoming of cars into airport is high
- Saturday and Sunday cars leave the airport is high
- Summers car park are nearly full

4.10.3 Bucketing based on Elastic Coefficient

To classify the data into buckets using price elasticity.

Price Elasticity

Price elasticity of demand (PED or Ed) is a measure used in economics to show the responsiveness, or elasticity, of the quantity demanded of a good or service to a change in its price when nothing but the price changes. More precisely, it gives the percentage change in quantity demanded in response to a one percent change in price.

$$Ed = \frac{\frac{dQ}{Q}}{\frac{dP}{P}}$$

where P is the price of the demanded good and Q is the quantity of the demanded good.

Based on e there are three categories:

- $e > 1$ then **Plastic Elastic**: When a change in demand is greater than the change in price, the demand for the product or good is said to be elastic. When a product is elastic, slight changes in price lead to huge changes in the demand for the product. Many goods and services that not necessity items are usually highly elastic. To determine the elasticity of the demand for a product, the percent change in quantity is divided by the percent change in price. When this equation is calculated, the answer reveals a product's elasticity. If the answer to the equation is equal to or greater than one, the product is considered elastic.

- $e < 1$ then **Price Inelastic**: Inelastic refers to the change in demand being less than the change in price on the product or good. Inelastic products are typically those people consider necessities. Changes in price do not change the demand for the product very much. When the elasticity equation is calculated, goods that are considered inelastic have an answer that is less than one.

- $e = 1$ then **Unitary Elastic**: Goods that are considered unitary in terms of elasticity are goods that have no change in demand when prices change. There are few goods ever considered unitary, but products such as medicine or utilities can

sometimes reach this point. No matter the prices charged, people find a way to purchase the goods, regardless. Companies selling goods that are unitary often make large profits because people consider these goods a necessity above all other goods.

Analysis of Result

Some of the bucketed data is not the pattern or trend. So, I tried another method

4.10.4 Applying the Neural Network

To check, whether all the feature present here are sufficient to predict the data. So, I made one layer neural network, trained.

Analyzing the results

After analysing the results,I went through some research papers and other websites to find the missing features. Some of them are:

- Weather
- Distance between airport and the city
- Fuel price
- Toll prices
- Traffic
- Public transportation
- Price transparency
- Convenience of car park

4.10.5 Results and scope

Analysis on the given airport car park. Helps the data scientist to predict and analysis more accurately.

Chapter 5

Some Problems

5.1 Problems

During internship, few of the problems, I solved are:

- Find the STLY(Same Time Last Year)?
- Fixed sum rounding problem?

5.2 Find the STLY

STLY(Same Time Last Year) is week-day adjusted same time last year.

For example: STLY of 01/12/2018 (Saturday) is 02/12/2017(Saturday)

Simple formula from the above example is 1 year minus 1 day is the STLY but problem comes If any one of the present or past year is leap year.I solved this problem.

```
public Date getSTLYDateForAGivenDate(String DateString){
    String yearString = DateString.substring(0,4);
    String monthString=DateString.substring(5,7);
    String dayString =DateString.substring(8,10);

    Integer yearVal = toInteger(yearString);
    Integer monthVal = toInteger(monthString);
    Integer dayVal = toInteger(dayString);
    Integer isItLeapYear = (((((yearVal % 4 == 0) && (yearVal % 100 != 0)) ||(yearVal % 400 == 0))&& (monthVal>2 || (monthVal==2 && dayVal==29) ))||
            (((((yearVal-1) % 4 == 0) && ((yearVal-1) % 100 != 0)) ||((yearVal-1) % 400 == 0))&& monthVal<=2)) ? 1:0;
    Integer noofDays = -1-isItLeapYear;

    return DateUtil.subtractDays( DateUtil.subtractYears(DateUtil.getUTCStartOfDayFromString(DateString), years: 1),noofDays);
}
```

FIGURE 5.1: STLY Solution Code

5.3 Fixed Sum Rounding Problem

Given a decimal array and there sum. Convert into array of integers by rounding with sum constant (as before) As it is believed that normal rounding sum differ at max by one.

5.3.1 Proving the difference is more than one

```
#include <bits/stdc++.h>

using namespace std;

int main()
{
    double sum = 0.0;
    for(int i =1;i<=10000;i++){
        sum+=i;
        for(int j=1;j<=100;j++){
            for(int k = 1;k<=j;k++){
                double ratio = (k)/(1.0*j);
                long int newSum = round(ratio * sum);
                long newS = 0;
                for(int m=1;m<=i;m++){
                    newS+= (round(newSum*(m/(sum))));
                }
                if(abs(newS-newSum)>1){
                    cout<<"Yes"<<endl;
                    break;
                }
            }
        }
    }

    return 0;
}
```

FIGURE 5.2: Code for proving that difference is more than 1

5.3.2 Solution to the above problem

Using array

Time Complexity: $O(N^2)$ and Space Complexity: $O(N)$

```
int fixedSumSolution(vector<double>v){
    vector<int>flooredVector(toFloor(v));
    double sum = sumOfVector(v);
    int sum1 = sumOfVector(flooredVector);
    int dif = round(sum -(double)sum1);
    //cout<<"DIFF - "<<dif<<" "<<sum<<" "<<sum1<<endl;
    while(dif){
        int index = maxDif(flooredVector,v);
        flooredVector[index]++;
        dif--;
    }
   // print(flooredVector);
    return sumOfVector(flooredVector);
}
```

FIGURE 5.3: Solution using Array for Fixed Sum Problem

Optimized Solution using Heap

Time Complexity: $O(NlogN)$ and Space Complexity: $O(N)$

```
int fixedSumSolution2(vector<double>v){
    vector<int>flooredVector(toFloor(v));
    double sum = sumOfVector(v);
    int sum1 = sumOfVector(flooredVector);
    int dif = round(sum -(double)sum1);
    priority_queue<Node,vector<Node>,CompareHeight >q;
    for(int i =0;i<v.size();i++){
        Node temp = {v[i],i};
        q.push(temp);
    }
    while(dif){
        int index =q.top().index;
        flooredVector[index]++;
        q.pop();
        Node temp = {(double)flooredVector[index],index};
        q.push(Node(temp));
        dif--;
    }
   // print(flooredVector);
    return sumOfVector(flooredVector);
}
```

FIGURE 5.4: Solution using Heap for Fixed Sum Problem

Bibliography

[1] Panou, Konstantinos, *Factors influencing car users propensity to shift to other modes and their impacts on demand for airport parking facilities*, Journal of Airline and Airport Management. 4(1), pp. 26-47, 2014, DOI: 10.3926/jairm.19.

[2] Guido Ferilli, *An analysis of the city centre car parking market: The supply side point of view (Case study), Edinburgh Napier University*

[3] Java web development "https://www.journaldev.com/1854/java-web-application-tutorial-for-beginners" [DOA- 1 July 2018]

[4] JUnit tutorials "https://www.tutorialspoint.com/junit/" [DOA-10 Aug 2018]

[5] Scikit-Learn tutorials "*https://scikit-learn.org/stable/tutorial/index.html*" [DOA- 25 Sept 2018]