

# **TRAVABLE & EVENT MANAGEMENT WEB – APPLICATIONS**

*By*

**Kartik Katahre**  
**(2016118)**

**Internal Supervisor**  
Prof. Aparajita Ojha

**External Supervisor**  
Mr. Sankalp Kumar



**Computer Science and Engineering**

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND  
MANUFACTURING, JABALPUR**

**END TERM REPORT**  
(15 November 2019)

## **ACKNOWLEDGEMENT**

During the 6 months of my internship, I have received immense support and love at Gyaanify. I would like to extend my thanks to all the people that were a part of my life during the last 6 months.

I extend my heartiest thanks and gratitude to my managers, Mr. Sankalp Kumar for their exemplary guidance, monitoring and constant encouragement during the course of the internship.

I would also like to thank all the members of the team for helping me and answering all the questions regarding the Project. I would like to thank them for creating such a healthy workplace at Gyaanify.

This internship project would not have been possible without the support of the institute. I take this opportunity to thank my internal supervisor Prof. Aparajita Ojha for her active guidance, help motivation and encouragement.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way in order to attain desired career objectives.

Hope to continue cooperation with all of you in the future.

Kartik Katahre

2016118

B.Tech (CSE)

## Contents

<b>1. Introduction</b>	<b>5</b>
<b>1.1 Overview</b>	<b>5</b>
<b>1.2 Literature survey</b>	<b>6</b>
<b>1.2.1 Restful (Representational State Transfer) API's</b>	<b>6</b>
<b>1.2.2 Restlet Client</b>	<b>6</b>
<b>1.2.3 Ajax (Asynchronous JavaScript and XML)</b>	<b>6</b>
<b>1.2.4 JSON (JavaScript Object Notation)</b>	<b>6</b>
<b>1.2.5 mPDF Library</b>	<b>7</b>
<b>1.2.6 MVC Architecture</b>	<b>7</b>
<b>1.2.7 Middleware's in Laravel</b>	<b>7</b>
<b>1.2.8 JQuery/ JavaScript</b>	<b>7</b>
<b>1.3 Internship plan</b>	<b>8</b>
<b>2. Traversable web based Portal</b>	<b>10</b>
<b>2.1 Database Schema</b>	<b>10</b>
<b>2.2 Embedded portal with Vtiger CRM</b>	<b>11</b>
<b>2.3 Login Authentication</b>	<b>12</b>
<b>2.4 User Authorization</b>	<b>13</b>
<b>2.5 Portal's Modules</b>	<b>14</b>
<b>2.6 API's for the portal</b>	<b>17</b>
<b>2.7 Log files</b>	<b>18</b>
<b>3. Event Management Application</b>	<b>19</b>
<b>3.1 Software Design Document</b>	<b>19</b>
<b>3.2 Database Schema</b>	<b>19</b>
<b>3.3 Security and Authentication</b>	<b>20</b>
<b>3.4 JSON responses and HTTP status codes</b>	<b>21</b>
<b>3.5 Attendance System</b>	<b>21</b>
<b>3.6 JSON encoded Input parameters</b>	<b>23</b>
<b>3.7 RESTful API endpoint structure</b>	<b>23</b>
<b>3.8 Access log and error logs</b>	<b>25</b>
<b>3.9 Data validation</b>	<b>26</b>
<b>3.10 API's versions</b>	<b>27</b>
<b>3.11 API's consumer guide</b>	<b>27</b>
<b>3.12 API's testing in Local Environment</b>	<b>29</b>

3.13	API's in Staging Environment .....	29
4.	Result and Discussions .....	30
5.	Conclusions .....	31
6.	Literature Cited.....	32

# 1. Introduction

## 1.1 Overview

The aim of this report is to discuss the progress and elaborate on the project which was assigned to me during the course of my internship. This report covers the training I've gone through and the technologies I have learned during my internship at Gyaanify.

As a Full stack web developer at Gyaanify, I worked on the development of a web based application and a portal. I spent the majority of my internship in developing

After my training period, I was assigned the task of development Traversable web based portal which is a portal for the travel agents to automate the process of creating Quotes in PDF format for the clients and the second project was Event Management Application, I worked on developing this application using micro services architecture for that I have created web services using REST for each use cases defined in the system design document.

I have worked on the following projects:

### **AIM OF THE PROJECTS (PROJECT DEFINITION):**

- **Event Management Application**

An application using micro services architecture which enables us to build the application with different technologies and scale the application according to the need. All the micro services communicate with each other through a stateless server which is REST.

- **Traversable Web Portal**

Web-based Portal, which is designed to assist travel agents create Quotes and itineraries for their specific requirements. An agent had to create the quote manually with the help of existing CRM (Customer Relationship Management Software) and Word documents which often required a lot of time and effort and possessed a high risk of mistakes in the Quotes as they are manually prepared for each requirement. So the Desired state was to create a web-based portal that could

fetch client requirements from the existing CRM using the API's and enable the agents to prepare the Quotes in the PDF format in Traversable system.

## **1.2 Literature survey**

### **1.2.1 Restful (Representational State Transfer) API's**

A RESTful API - also referred to as a RESTful web service -- is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development.

### **1.2.2 Restlet Client**

Restlet client is a Chrome add-on which is used to fire requests to an API. It is very lightweight and fast. Requests can be organized in groups, also tests can be created with verifications for certain conditions on the response. With its features, it is very good and convenient API tool.

### **1.2.3 Ajax (Asynchronous JavaScript and XML)**

Learned Ajax (Asynchronous JavaScript and XML) for creating dynamic web pages which allows pages to update small amounts of data without reloading the whole page.

### **1.2.4 JSON (JavaScript Object Notation)**

JSON stands for JavaScript Object Notation. It is a very lightweight format so we can easily transport the data using JSON. Using JSON object we can store the data fetched from the Vtiger CRM and transport it to the Traversable portal.

### 1.2.5 mPDF Library

It is a PHP library using which we can generate PDF files. It has some predefined functions using which we can easily design the PDF layout.

### 1.2.6 MVC Architecture

Model View Controller is a software design pattern for developing web applications made up of the following three parts –

**Model** – Responsible for maintaining data.

**View** – Responsible for displaying all or a portion of the data to the user.

**Controller** – Controls the interactions between the Model and View.

### 1.2.7 Middleware's in Laravel

Learned middleware's and used there middleware's in login authentication for verifying the user. If the user is authenticated it will redirect the user to the Dashboard. If the user is not authenticated, it will redirect the user to the login screen.

### 1.2.8 JQuery/ JavaScript

JQuery is a fast, small, and JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation.

## 1.3 Internship plan

The first few weeks were allotted to me for Learning the technologies and understanding the project requirements. I got the online training of Laravel MVC framework on Udemy, Web services using REST and different architectures of developing the software's like monolithic and micro services architecture.

In my training period I learned and understood the CRM modules and creating the workflows in CRM. I learned various PHP libraries like PHPExcel PHPMailer and mPDF and their documentations so that I could use them according to the requirement.

Later I was introduced to the Front end team which was working on developing the frontend for Event management Application in web and Android app.

I was given training for few initial weeks which included the following:

- **REST architecture**

REST is used to build Web services that are lightweight, maintainable, and scalable. A service which is built on the REST architecture is called a RESTful service.

The key elements of a RESTful implementation are as follows:

- **Request Headers**

Additional instructions sent with the request like authorization details.

- **Request Body**

Data is sent with the request body. Data is normally sent in the request When a POST request is made to the REST web service.

- **Response Body**

When we made a request to the server it returns the response in response body either in JSON or XML.

- **Response status codes**



Using HTTP response status codes user can understand whether a specific HTTP request has been successfully completed. For example status code 201 is used when the data is created successfully using POST method, similarly status code 200 is used when we make the PUT request and it executes successfully.

- **Restful methods**

RESTful APIs enable us to develop any kind of web application having all possible CRUD (create, retrieve, update, delete) operation.

Following are the HTTP methods that are used for the API's

**POST:** Creates records

**GET:** Retrieves records

**PUT:** Updates records

**DELETE:** Deletes records

- **Laravel Framework**

The training/workshop was on Laravel Framework which included and covered the following:

- Artisan CLI ( command-line interface )
- Laravel Directory Structure
- Basic routing
- Call a controller method from a route
- Migrations
- Using Controllers and Routes for URLs and APIs
- Eloquent ORM ( Storing and Using Data )
- Using Ajax and jQuery
- Integrating with Bootstrap

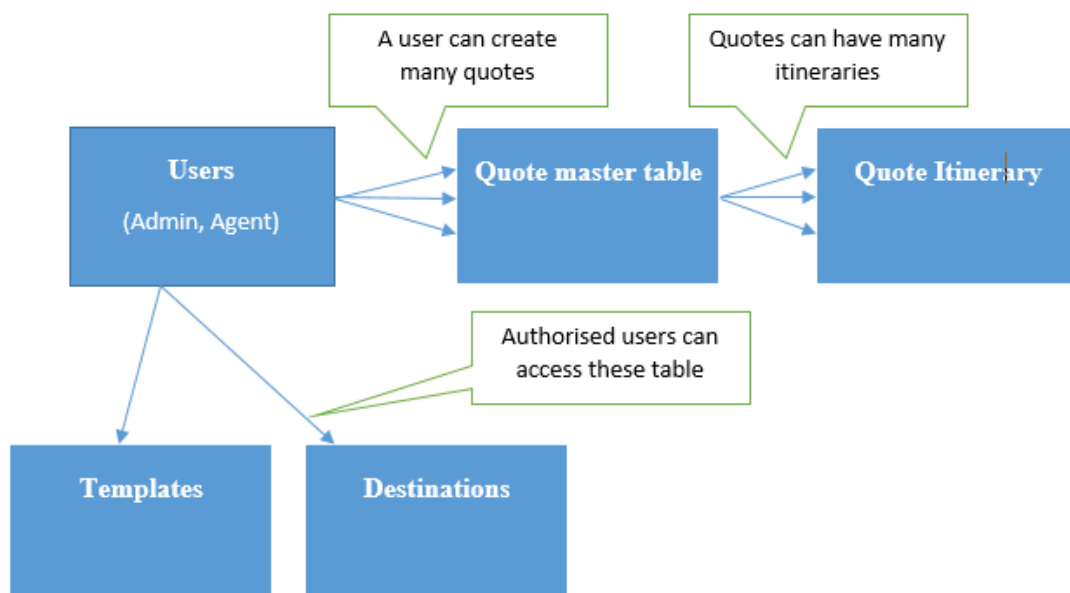
## 2. Traversable web based Portal

### 2.1 Database Schema

Traversable database schema uses database normalization which is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships.

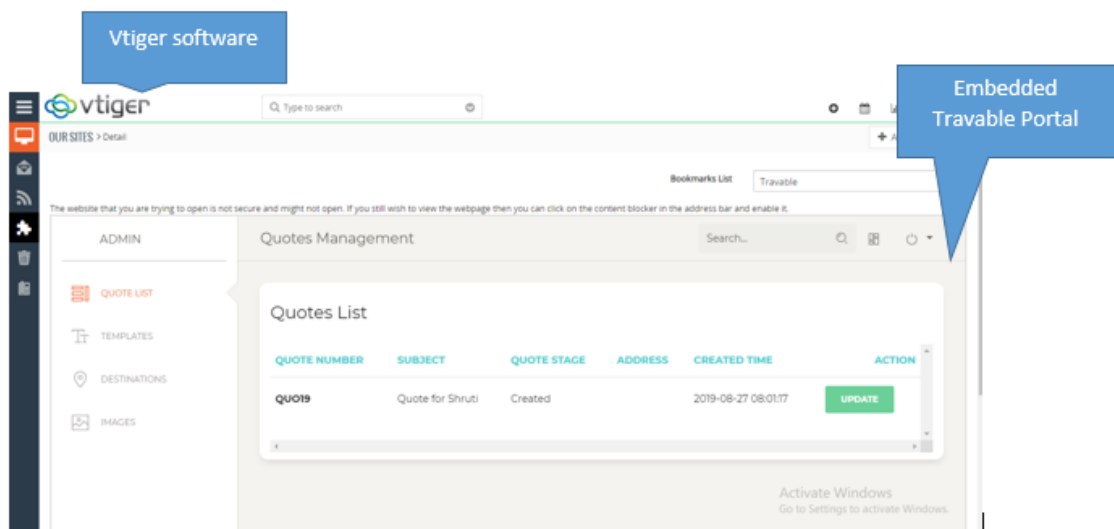
Following are the tables that we have used in this system.

- Users (id, username, password, user role )
- Quotes (id, quote no., subject, quote type assign to, validity etc.)
- Quote itineraries (quote no, day summary, day itinerary, image etc.)
- Destinations (id, destination, summary, package inclusions, exclusions, etc.)
- Templates (id, Template no., template heading name, template heading value)



## 2.2 Embedded portal with Vtiger CRM

Embedded Travable portal with Vtiger CRM using Iframe. A user has to login in the VTIGER CRM first after successfully logged in into the CRM user can get his access key in settings, using this key he can logged into the Travable portal and after successful login a he can access the Travable module according to the user roles and privileges.



## 2.3 Login Authentication

Once a user logged in into the Vtiger CRM he can get his access key in settings. User can use this access key as a password if he wants to log in into the Traversable portal although Laravel provides its predefined login authentication system, but as per the requirement, we have created a custom login authentication which is authenticating the users from users table from the Portal's database.

Username or Password Incorrect

Username

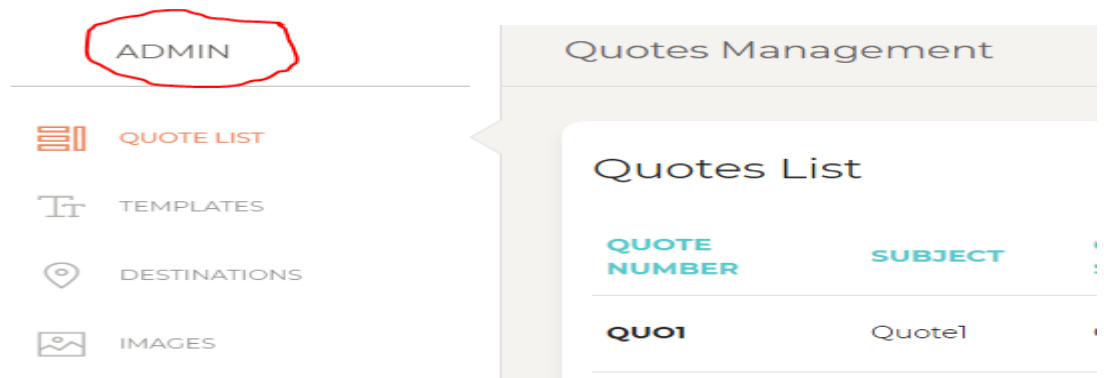
Password

Submit

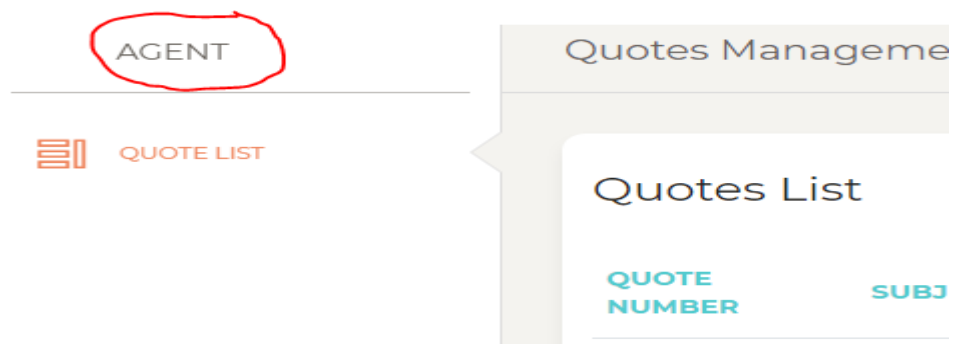
## 2.4 User Authorization

There are two users' roles in this portal Admin and Agent. Using Authorization, the system determines if the user has permission to use a resource or access a file. In this system an agent can access only quote module if a user is logged in as an admin he can access all the portals module.

- **If the user logged in as an Admin :** All the Portal's modules are visible to the user



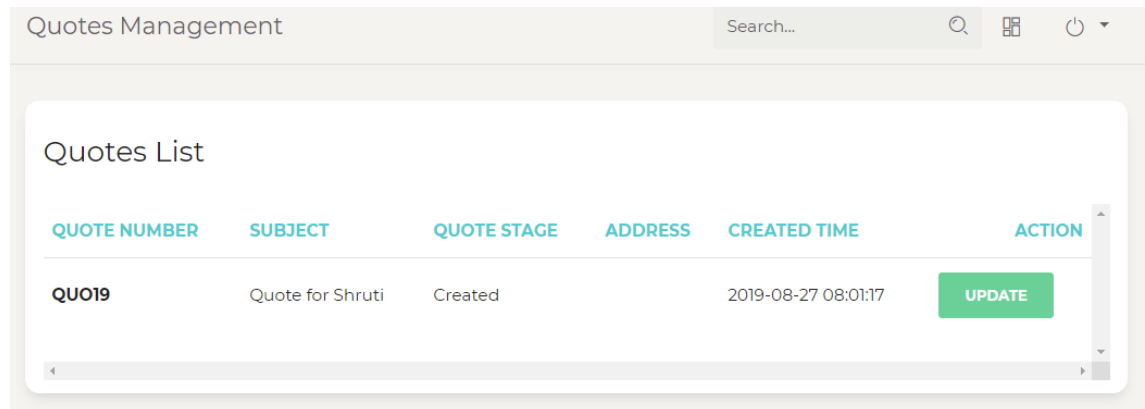
- **If the user is logged in as an Agent :** Only quote module is visible



## 2.5 Portal's Modules

Following are the Portal's modules

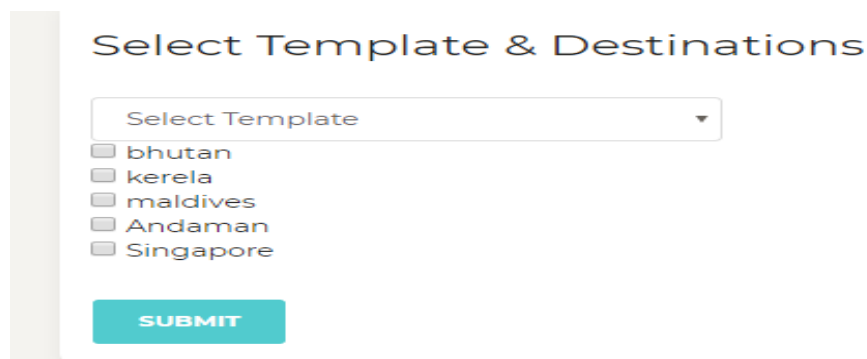
- **Quote module:** In this module, a user can view all the quotes that are assigned to the logged user. A user can create/ update the quote in this module. After selecting the quote from the list he can choose a quote template. After selecting the quote template system fetches all the details for a particular quote using web service and a user can use this information and enters the required additional information's for creating the quote in PDF format.



The screenshot shows a web interface titled "Quotes Management". It features a search bar and a table titled "Quotes List". The table has columns for "QUOTE NUMBER", "SUBJECT", "QUOTE STAGE", "ADDRESS", "CREATED TIME", and "ACTION". There is one row of data with the quote number "QU019", subject "Quote for Shruti", stage "Created", and created time "2019-08-27 08:01:17". An "UPDATE" button is visible in the "ACTION" column.

QUOTE NUMBER	SUBJECT	QUOTE STAGE	ADDRESS	CREATED TIME	ACTION
QU019	Quote for Shruti	Created		2019-08-27 08:01:17	<button>UPDATE</button>

User can select template and destinations for the quote after choosing create/ update quote.



The screenshot shows a form titled "Select Template & Destinations". It contains a dropdown menu labeled "Select Template" and a list of checkboxes for destinations: bhutan, kerela, maldives, Andaman, and Singapore. A "SUBMIT" button is at the bottom.

Select Template & Destinations

Select Template

☐ bhutan  
☐ kerela  
☐ maldives  
☐ Andaman  
☐ Singapore

SUBMIT

User can export the quote in PDF format after entering all the required details and system will generate the PDF file using the data received from the CRM (using web service) and Portal's data (Entered by the user).

PREVIOUS

FORWARD

EXPORT PDF

Data from CRM

QUO2

Quote for Shruti

Created

2019-08-27 08:01:17

Opp1

Delhi

Maldives

Administrator

- **Templates module:** In this module, a user can manage the templates. He can create new template and manage the template fields according to the requirement.

Templates

Select Template

SUBMIT

TEMPLATE NUMBER	HEADINGS	VALUES	SEQ	ACTION
tem_1	Hotel	hotel_details	90	DELETE
tem_3	Hotel Details	template3_hotel_details	15	DELETE

- **Destinations:** In this module, a user can manage the destination's information's.

Destinations

Enter Destination name.

ADD

maldives

SUBMIT

MALDIVES

- **Image Module:** In this module an Admin can save images into the Portal's server. An agent can use these image at the time of creating the quote for the client. Admin can select the image from his local system as an input and enter the relevant tags. Tags will be used to search the image while creating the quote.

## Images

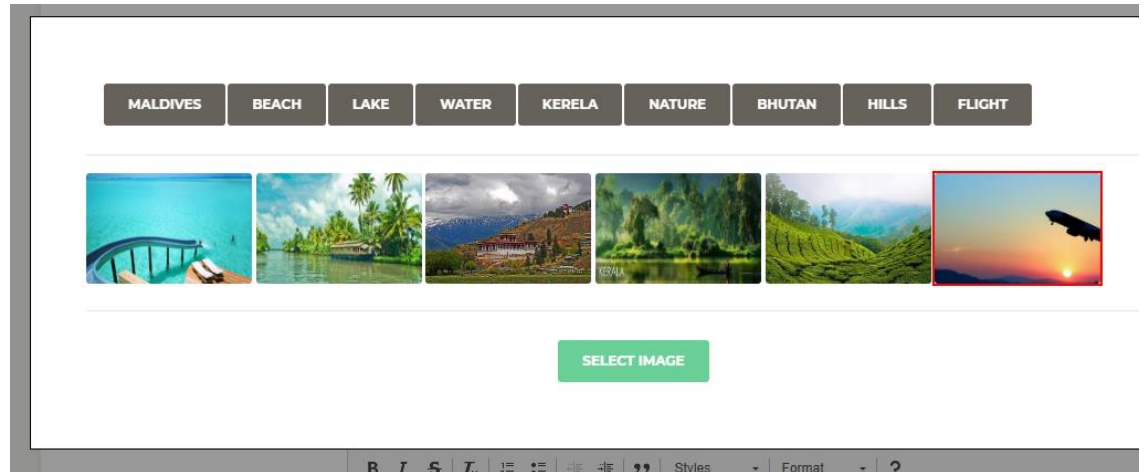
Choose File

No file chosen

ex: beach,nature,maldives....

UPLOAD IMAGE

User can select the images while creating the quote for the client and also filter the images using tags.



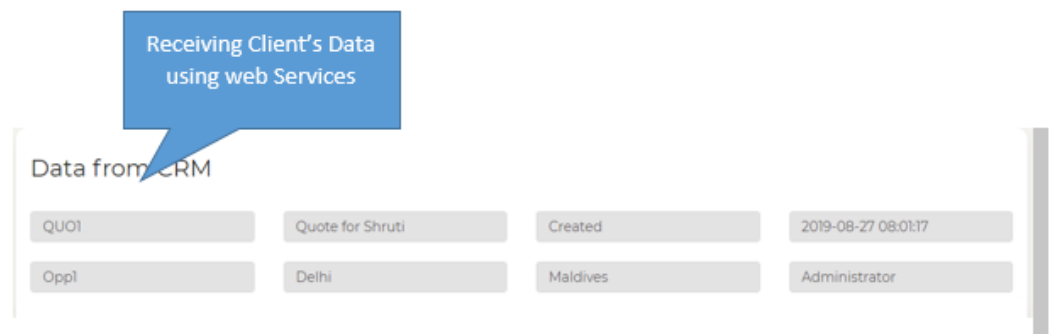


## 2.6 API's for the portal

Used Vtiger CRM libraries for creating the web services. These web services provides the required data from Vtiger CRM to Traversable portal.

Portal uses the following API's

- **For Quotes data:** This web service retrieves all the quotes assigned to a particular agent.
- **For Opportunity data:** Retrieves necessary information required by the client.
- **For Contacts data:** Retrieves the contact information of a client.



## 2.7 Log files

- **Access log files:**

Log files that contain the information about requests coming into the web server and record each and every event that occurred in the portal.

```
-----
11-09-19 08:41:42 admin Created Quote Package with Quote number QUO2
11-09-19 08:41:43 admin Updated Quote Package with Quote number QUO2
11-09-19 08:43:23 admin Saved Package itinerary for Quote number QUO2
11-09-19 08:44:18 admin Downloaded PDF for QUOTE Number QUO2 with tem_3 template
11-09-19 08:44:31 admin Downloaded PDF for QUOTE Number QUO2 with tem_3 template
11-09-19 08:44:41 admin Updated Quote Package with Quote number QUO2
11-09-19 08:44:42 admin Downloaded PDF for QUOTE Number QUO2 with tem_3 template
11-09-19 08:46:01 admin Downloaded PDF for QUOTE Number QUO2 with tem_3 template
11-09-19 08:51:40 admin Updated Quote Package with Quote number QUO2
11-09-19 08:51:46 admin Downloaded PDF for QUOTE Number QUO2 with tem_3 template
11-09-19 08:52:24 admin Downloaded PDF for QUOTE Number QUO2 with tem_3 template
11-09-19 09:08:30 admin Updated Quote Package with Quote number QUO1
11-09-19 09:08:32 admin Downloaded PDF for QUOTE Number QUO1 with tem_3 template
-----
```

## 3. Event Management Application

### 3.1 Software Design Document

Before start working on this project I prepared the software design document for the Event management application which is a written description of a software product, and provides the overall guidance to the architecture of the project.

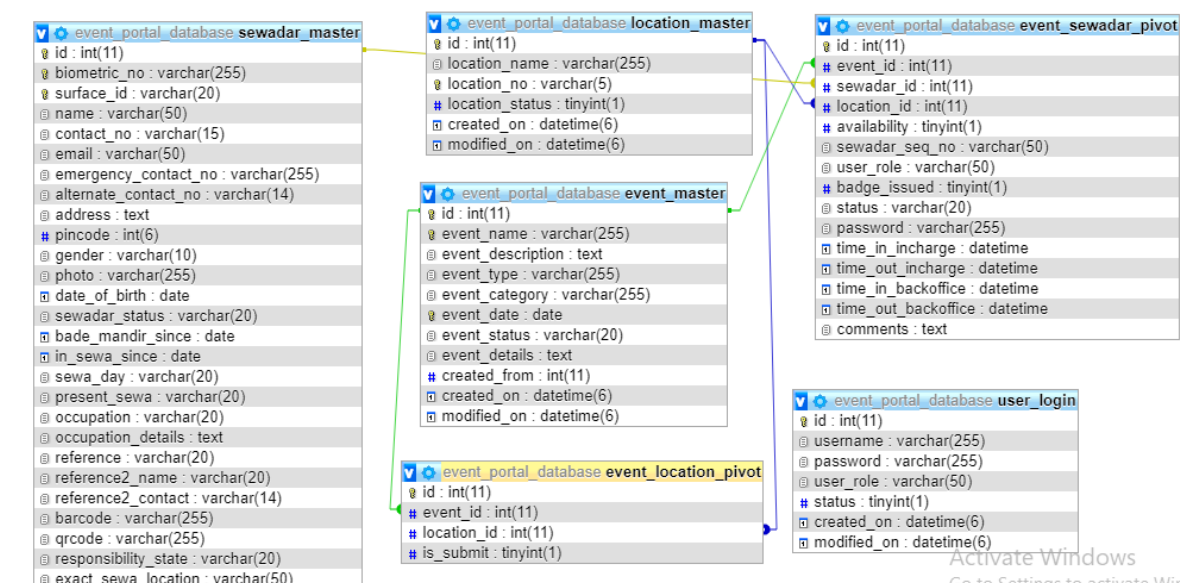
Following are the points that are covered the Document.

- Introduction
- Requirements
- Business design
- Use cases
- Component details

### 3.2 Database Schema

Database for the system so that the CRUD (CREATE, Read, update, delete) operation can be achievable by the API's. Created some master tables as well as pivot tables for mapping two or more tables as well as summarize the data in the database.

Database schema for the application



### 3.3 Security and Authentication

For security and authentication of the application we are generating the API Keys instead of username and password authentication. User has to provide login credentials (username and password) only at the time of login which uses the calls the login API. If the user is authenticated by the system, it will return API key (token ID) to the logged-in user which will be only available for a particular period of time and will be destroyed at the time of logout.

An API key is a long series of random characters that is difficult to guess. Username/password are typically much smaller in length, use common words that are generally insecure.

After successfully receiving the API key user can call the other API's. In every API system will check the user API key and its validity. API will respond only if the user is authenticated successfully by the system.

**If login credential are valid and the user is authenticated by the system return API Key (Token ID) with status code 200 and status message "ok"**

```
{
  status : 200,
  status_message : "Ok",
  data : {
    tokenID : "u1HxcHJAM0nSFaqFCYddA7q91iJPND6HZew51evSkvJIGgcbb4gzBYeNWeO",
    username : "admin",
    user_role : "admin"
  }
}
```

**If user is not authenticated by the system return null with status code 401 and status message "Unauthorized"**

```
{
  status : 401,
  status_message : "Unauthorized",
  data : null
}
```

### 3.4 JSON responses and HTTP status codes

Using HTTP response status codes user can understand whether a specific HTTP request has been successfully completed or not.

Following are some status codes that we have in this application.

Status Code	Status Message	Code Type
200	Ok	Success
201	Created	Success
500	Internal Server Error	Server Error
440	Login Time Out	Client Error
401	Unauthorized	Client Error

### 3.5 Attendance System

Implemented attendance system in which a QR code need to be send using android app. Every member has a unique QR code in the database when client uses attendance API, the API accepts the QR code as an input than this web service searches for the particular member in the database and if he is available for the event system update his attendance as IN and return the member name and other details in JSON format, web service uses the same procedure for the attendance OUT.

## For attendance IN

▶ BODY ⓘ

pretty ▼

```
▼ {
  status : 200,
  status_message : "Ok",
  data : ▼ {
    Attendance : "IN",
    Name : "name27",
    photo : null,
    location_no : "03",
    sewadar_seq_no : "01",
    user_role : "incharge"
  }
}
```

lines nums

length: 158 bytes

## For attendance OUT

◀ ▶ BODY ⓘ

pretty ▼

```
▼ {
  status : 200,
  status_message : "Ok",
  data : ▼ {
    Attendance : "OUT",
    Name : "name27",
    photo : null,
    location_no : "03",
    sewadar_seq_no : "01",
    user_role : "incharge"
  }
}
```

lines nums

length: 159 bytes

## 3.6 JSON encoded Input parameters

Every web service accepts JSON encoded parameters so that any of the client application (Android/ IOS/ Web application) can send the parameters for the API and API's receives the input in desired format like 2D arrays or associative arrays.

JSON encoded form parameters



## 3.7 RESTful API endpoint structure

An API endpoint is a URL that performs some action against the database when it is called. Uses the design pattern for the URL that restful API recommends.

- **Descriptive URL:**

Using a Descriptive URL the user can relate the action to be performed by using a particular web service. Included the API version number in the path so that it will enable us to update the API in the future.

The screenshot shows a REST client interface. At the top, the endpoint is 'create\_event\_scratch'. To its right is a 'Save' button. Below this, the 'METHOD' is set to 'POST'. The URL is 'http://13.232.30.207/bmap/api/create\_event\_scratch.php', which is highlighted in yellow. Above the URL, a schema template is shown: 'SCHEME://HOST[:PORT]/[PATH["?"]QUERY]'. To the right of the URL is a 'Send' button. Below the URL, it says 'length: 55 byte(s)'. Underneath the URL field, there is a section for 'QUERY PARAMETERS' with a downward arrow.

- **One or more HTTP methods:**

RESTful APIs enable us to develop any kind of web application having all possible CRUD (create, retrieve, update, delete) operation.

Following are the HTTP methods that we have used for the API's development.

- **POST:** Creates records
- **GET:** Retrieves records
- **PUT:** Updates records
- **DELETE:** Deletes records

- **Return data:**

API's returns the data in JSON (Java Script Object Notation) format which is a lightweight data interchange format and we can use it for sharing data among applications and interfaces.

### **Data in JSON Format**



```

{
  status : 200,
  status_message : "Ok",
  data : {
    event_types : [ { id : "2", event_type : "New Year" }, { id : "3",
    event_categories : [ { id : "2", event_category : "Function" } ],
    event_status : [ { id : "1", event_status : "Upcoming" }, { id :
  }
}

```

### 3.8 Access log and error logs

Access logs and error log files contain the information about requests coming into the web server and record each and every event that occurred in the portal and also implemented a use case.

**Access log can be analysed to tell us -**

- Information regarding the incoming requests to a web server.
- Provides additional data that is useful for debugging purposes, informational purposes, and more.
- The number of times a user visited the Portal.

```

13-11-19 06:54:42 : SUCCESS : Checking for data in user_login Columns & Values: username="admin" API Name /Event_management/api/login.php
13-11-19 06:54:42 : SUCCESS : Inserted in usersessions Columns: username,event_id,location_id,user_role,login_type,tokenCreatedOn,tokenExpiry,tokenID Values: 'admin'
07:24:42', API Name /Event_management/api/login.php
13-11-19 06:54:42 : SUCCESS : admin Logged In API Name /Event_management/api/login.php
13-11-19 06:55:48 : SUCCESS : Checking for data in usersessions Columns & Values: username="admin" API Name /Event_management/api/add_sewadar_in_waitlist.php
13-11-19 06:55:48 : SUCCESS : Checking for data in usersessions Columns & Values: username="admin"sessionID="" API Name /Event_management/api/add_sewadar_in
13-11-19 06:55:49 : SUCCESS : Updated Table Name: usersessions Columns&Values: tokenExpiry="2019-11-13 07:34:42" Where: username="admin"AND sessionID=""AN
tokenID="jFCeAhuRZ4to7lX8rDDi5MDfUEneBUFY9A191wtakXtdxt1h5hO3YDSs2cs76pez" API Name /Event_management/api/add_sewadar_in_waitlist.php
13-11-19 06:55:49 : SUCCESS : Checking for data in usersessions Columns & Values: username="admin" API Name /Event_management/api/add_sewadar_in_waitlist.php
13-11-19 06:56:58 : SUCCESS : Checking for data in usersessions Columns & Values: username="admin" API Name /Event_management/api/add_sewadar_in_waitlist.php
13-11-19 06:56:58 : SUCCESS : Checking for data in usersessions Columns & Values: username="admin"sessionID="" API Name /Event_management/api/add_sewadar_in
13-11-19 06:56:58 : SUCCESS : Updated Table Name: usersessions Columns&Values: tokenExpiry="2019-11-13 07:44:42" Where: username="admin"AND sessionID=""AN

```

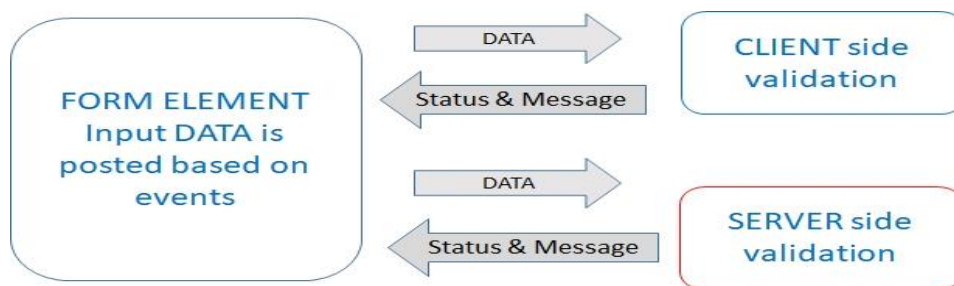
### 3.9 Data validation

When we enter data, the web application checks it to see that the data is correct. If the information is correct, the application allows the data to be submitted to the server and saved in a database and if the information is not correct, it gives us an error message explaining what needs to be corrected.

Some validation tasks are:

- User has to fill all the required fields.
- User has to enter a valid email address and mobile number.
- Only alphabets have to be filled in the Text field.

Validation can be defined by many different methods and deployed in many different ways.



- **Server side validation:** It is performed by a web server, after input has sent to the server.

Sewadar Status	<input type="text" value="Waitlist"/>	Sewa Location	<input type="text" value="04 Security"/>
Biometric number	<input type="text" value="test234"/>	Sewadar Name	<input type="text" value="Test Name"/>
Mobile Number	<input type="text" value="8889598183"/>	Sewadar Email	<input type="text" value="test@gmail.com"/>
Gender	<input checked="" type="radio"/> Male <input type="radio"/> Female	Date of Birth	<input type="text"/>
Referred by	<input type="text" value=""/> <div>*Required</div>		
<input type="button" value="Submit"/>			

Active  
Go to Se

- **Client side validation:** It is performed by a web server, before input is sent to a web server.

<b>Sewadar Status</b>	<input type="text" value="Waitlist"/>	<b>Sewa Location</b>	<input type="text" value="04 Security"/>
<b>Biometric number</b>	<input type="text" value="232323212312"/> <small>Max 8 digits are allowed!</small>	<b>Sewadar Name</b>	<input type="text" value="Test Name3"/> <small>Only letters and white apaces are allowed!</small>
<b>Mobile Number</b>	<input type="text" value="08889598183"/> <small>Must contain only digits and length must be 10!</small>	<b>Sewadar Email</b>	<input type="text" value="test@gmail.com"/>
<b>Gender</b>	<input checked="" type="radio"/> Male <input type="radio"/> Female		
<b>Referred by</b>	<input type="text" value="Team 1"/>		

Activate Winc  
Go to Settings to i

### 3.10 API's versions

Divided the API's into versions each version contains three or four API's so that it can be easy for us to testing the API's as well as for the Front end team. We have shared the API's as well as API's consumer guide (document which contains the instructions for using an API) version wise with the Front end team.

Different versions will manage the changes and configuration of an application and it also allows us to easily share the API's with the frontend team.

### 3.11 API's consumer guide

An instruction Guide contains all the necessary information about the API's such as:

- **URL:** URL of the web service using which they can make request
- **Methods:** Method of the web service (GET, POST, PUT, DELETE).
- **Input Fields:** Input parameters, if required for the particular web service.

- **Success:** Success output with status code, status message and data, if the request made successfully and the user authenticated by the system.
- **Error:** Error output with status code, status message and data received, if internal server error, Login Time out and the user is not authenticated

API consumer guide for login web service that we have shared with the frontend team:

Name of Web Service	1. LOGIN
URL	http://13.232.30.207/bmapi/api/login.php
Access Level	For all user roles
Inputs:	<ul style="list-style-type: none"> <li>• <b>username*</b> <ul style="list-style-type: none"> <li>○ A string of maximum 255 characters</li> </ul> </li> <li>• <b>password*</b> <ul style="list-style-type: none"> <li>○ A string of maximum 255 characters</li> </ul> </li> <li>• <b>login_type</b> <ul style="list-style-type: none"> <li>○ should be event or admin</li> </ul> </li> <li>• <b>event_id</b> <ul style="list-style-type: none"> <li>○ only if login type is event</li> </ul> </li> </ul>
Request Method	POST
Return Data String Format	JSON
Action	<ul style="list-style-type: none"> <li>• Accept username &amp; password</li> <li>• Authenticate the credentials</li> <li>• If authenticated, return "SUCCESS" with: <ul style="list-style-type: none"> <li>○ tokenID: (To be used for every successive API calls for the session. System will reject any API call without a valid tokenID. Tokens expire on 30 minutes of inactivity)</li> <li>○ username: (The logged-in username against whom the token is tagged)</li> <li>○ User role</li> </ul> </li> <li>• If authentication fails, return "ERROR" with the error message "Unauthorized"</li> </ul>
Success	<b>Login type: Admin</b> <pre>{   "status": 200,   "status_message": "Ok",   "data": {     "tokenID": "ANYiQ05OysR2off37umVA123G1n81txEaEGLhy77731JtdSrqM0pe3UK71jLV04",     "username": "testUser",     "user_role": "admin"   } }</pre> <b>Login Type : Event</b> <pre>{</pre>

### 3.12 API's testing in Local Environment

Restlet client for testing the API's locally on the system.

Testing of API's in localhost using Restlet client

The screenshot shows the Restlet client interface for testing a local API endpoint. The title is "create\_event+scratch". The METHOD is set to "POST". The URL is "http://localhost/Event\_management/api/create\_event\_scratch.php?". The QUERY PARAMETERS section shows a parameter "name" with a value "value". The HEADERS section shows "Content-Type" as "application/x-www-form-urlencoded", "tokenID" as "hzNIJWXHIGAKoIAXYZ4F", and "username" as "admin". The BODY section shows "event\_name" as "testEvent3", "event\_type" as "sdf", and "event\_description" as "dfsdf". There is an "Activate Windows" watermark in the bottom right corner.

### 3.13 API's in Staging Environment

After testing the API's locally on the system, created a staging environment on Linux Server and hosted the project on the server so that the frontend team can test the API's.

Testing of API's in staging environment using Restlet client

The screenshot shows the Restlet client interface for testing a staging API endpoint. The title is "create\_event\_scratch". The METHOD is set to "POST". The URL is "http://13.232.30.207/bmapi/api/create\_event\_scratch.php". The QUERY PARAMETERS section is empty. The HEADERS section shows "username" as "adminuser", "tokenID" as "xyWh6JTWsDkF6J0KI4rd", and "Content-Type" as "application/json". The BODY section shows a single line "1".

## 4. Result and Discussions

During my six months internship course at Gyaanify, as a full stack developer, I have learned a lot. In the first few months of my internship, I had some technical training.

After the training, I was assigned the task of developing a Travable web-based portal which is a portal for the travel agents to automate the process of creating Quotes in PDF format for their clients. For this Project, I have created web services that are retrieving the essential data from the CRM and providing to the Portal. Completed Login authentication in which Admin and Agent will have different interfaces according to their privileges also user authorization so that users can only access the authorized files. The system is generating session variables after a successful login and destroying when the user logout from the system. The system is generating the access logs and error logs for each and every user action in the system.

The second project is Event management Application, I have worked on this application with micro services architecture for that I have created web services using REST for each use cases defined in the system design document. I also tested the APIs in the local system using Restlet client which is a lightweight chrome extension and after tested the APIs locally we created a staging environment on AWS and tested the API on a staging server. We divided the API into a version where each version contains some API files and prepared the API Consumer guide for the Front end team.

## 5. Conclusions

This report discussed in detail the tasks accomplished during the six months long internship period at Gyaanify as a full stack developer. During the internship, I have gone through certain technical training that helped in my development.

I learned about the different architecture of developing applications like monolithic and microservice architectures as well as working in a team. During my internship, I got familiar with the various HTTP methods (GET, POST, PUT and DELETE) and their uses. Learned the JOINS in MySQL. Learned to test the web service using the RESTLET Client tool.

## 6. Literature Cited

- <https://www.udemy.com/course/php-with-laravel-for-beginners-become-a-master-in-laravel/>
- <https://hackernoon.com/a-beginners-guide-to-building-a-rest-api-with-laravel-5c717afd77fe>
- <https://laravel.com/docs/5.8/migrations#creating-tables>
- <https://www.sitepoint.com/role-based-access-control-in-php/>
- <https://community.vtiger.com/help/>
- <https://restfulapi.net/>
- [https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)
- [https://www.w3schools.com/js/js\\_json\\_intro.asp](https://www.w3schools.com/js/js_json_intro.asp)
- [https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form\\_validation](https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation)
- <https://www.codexworld.com/store-retrieve-image-from-database-mysql-php/>
- <https://www.youtube.com/watch?v=eoGITOPpBfU&t=228s>
- <https://stackoverflow.com/questions/27941207/http-protocols-put-and-delete-and-their-usage-in-php>