|  |
| --- |
| Asia pasific university of technology & innovation |
| Project Report |
| Internet Application |
|  |
| **Muhammad Fayzan Siddiqui**  **Student ID: TP033089**  **Intake code: UCMP1308MCS**  **Lecturer: Naresh Kumar Appadurai** |
| **Date Assigned: 25th January 2014**  **Date Completed: 10th March 2014**  **E-Commerce Store Application for Travel & Tours** |



|  |
| --- |
|  |

Table of Contents

[**1.** **Introduction** 7](#_Toc382223465)

[**2.** **Requirement Specification** 10](#_Toc382223525)

[**3.** **System Design** 13](#_Toc382223584)

[**4.** **Implementation** 20](#_Toc382223585)

[**5.** **Conclusion** 30](#_Toc382223586)

[**6.** **User Manual** 31](#_Toc382223592)

1. **Introduction**
   1. **Company’s Business**

Tropica Travel Ltd is a leading travel service company in Malaysia, provides diversified range of services to both leisure and corporate travelers. They also provide tour packages for Malaysia and elsewhere in the world. Also provide customer support to their customers.

* 1. **Business Extension**

As in the current information technology era all the businesses are going online. The company also want to sale there services online and enable their customers to buy their tour packages online through internet from anywhere in the world. Going online will extend their business and increase the opportunity to facilitate more customers.

* 1. **Background**

This document defines the functionalities and features of the company’s e-commerce website, which is developed as an online store to sale their tour packages.

The website visitors can see the publically available features like tour packages, package details including price and package specific information. They can also search the available packages through the calendar, by providing single date or date range. After selecting the desired package they can add that package in the shopping cart. Visitors can add and remove items from the cart. But to purchase them, sign in is required.

There are some features which required sign in. To sign in, new users need to register first. After signing in the registered users can also view and do all the features and functionalities which a normal visitor can. Other than that, the register user can purchase the package or packages added in the cart (checkout cart), view the orders and can provide feedback and comments.

There are three types of users of this website.

Visitors – can access only publically available content.

Members – can access content available to registered users.

Admin – can access website management functions.

* 1. **Scope of Project**

The scope of this project is to provide an e-commerce website through which company can display their packages and there price to the potential customers and also enable them to purchase online. The administrator of the website can manage the website content (i.e. tour packages), manage the orders from the customers and also view there feedback and comments. The potential customers can register as a member and purchase the packages, manage their orders and also give feedback and comments.

User interface need to be simpler so that all type of users (i.e. visitor, member, admin) can easily understand and use. The website can be able to access from all five popular browsers (i.e. Mozilla Firefox, Google Chrome, Internet Explorer, Safari and Opera) and can also access from tablet and mobile devices.

* 1. **Target audience**

This document is for the stakeholders who required this application and also the team which is responsible for design, development, implementation and maintenance of the application.

* 1. **Project Schedule Summary**

The project should go through with the following phases.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Duration** | **Phase** |
| 1. | 02 man days | Requirement gathering |
| 2. | 02 man days | Analysis |
| 3. | 04 man days | System design |
| 4. | 15 man days | Development |
| 5. | 02 man days | Implementation |
| 6. | 02 man days | Testing |

Total man days required: 27

* 1. **Major Functions**

Functions of the website are divided by user types. There are three types of users Visitor, Member and Administrator.

1. Visitor
   1. View the home page displaying the packages, calendar to search, sign in panel and links.
   2. Search packages by providing single date or date range.
   3. Browse packages.
   4. View package details, including price, tour date, number of days nights.
   5. Add product into shopping cart.
   6. View shopping cart.
   7. Remove items and update shopping cart.
   8. Register as a member.
2. Member
   1. Sign in to website as member.
   2. Purchase the packages added in the shopping cart (cart checkout).
   3. View the status of orders (purchased packages). There are three status of orders Pending, Done, Cancel.
   4. Delete orders.
   5. Request an enquiry.
   6. Give feedback and comments.
   7. Change password
   8. Log out.
3. Administrator
   1. Sign in to website as admin.
   2. Manage products. (add new product, edit existing product, delete product)
   3. Manage orders (view order, delete order, change order status)
   4. View enquiries, feedback and comments.
   5. Change password
   6. Log out.
4. **Requirement Specification**

Following are the requirements from the various stakeholders.

* 1. **Functional Requirement**
     1. **Front-End**
* The front-end of the application allow users to view the tour packages offered by the company.
* There should be a calendar which allows users to search the packages by selecting different dates and can also search with date range.
* The main page shows the name of the package with an image, and also buttons to view the package and add to cart.
* When user clicks the view button, the package detail should be displayed including the description of package, tour date, price, number of days and nights, and also add to cart button.
* When add to cart button is clicked, either from the main page or from the package detail page, the package should be added in the shopping cart.
* All pages should also display the summary of shopping cart (i.e. number of items in cart and total price) as a link to the cart detail view page.
* When click on the cart summary, the cart detail page should be displayed.
* On card detail page, user can view cart items with their details including item name, quantity, price and also total price.
* On card detail page, user can increase or decrease the item quantity or can delete the item. A cart clear button should also be there which clears all the items from the cart.
* Cart detail page should have a checkout button which is used to checkout and purchase cart items. Also a ‘continue shopping’ button should be there which will redirect the user to the package listing page (main page) to continue shopping and add more items in the cart.
* To check out the cart, user need to sign in first otherwise user should be prompted.
* There should be a sign in panel on the sidebar which sticks with all the pages. Users already having member account can sign in using that panel anytime.
* If users didn’t have a member account they can register as a new member by the registration form.
* After signing in user can view change password, logout and order management link.
* Through change password link users can easily change their password. And logout link is used to logout the user.
* The order management link should display the list of orders. User can check the status of the order and also be able to delete order. A search box should also be there which is used to search the order from the order list.
* The side bar on main page should also have a link for feedback and enquiry page. A signed in user can only access this page.
* From feedback and enquiry page user can send feedbacks, enquiries and comments to the website administrator.
* When user logs out, if there is any item in the shopping cart it should be removed.
  + 1. **Checkout**
* After adding the packages in the shopping cart, signed in users (members) can be able to checkout at anytime through clicking the checkout button in the shopping cart detail view page.
* When user clicks on checkout they should be redirected to the PayPal checkout process.
* The shopping cart items with their prices should be sent to the PayPal and generate an invoice.
* If user has PayPal account they can proceed with payment through their account, otherwise they can chose other payment options provided by PayPal like credit card etc.
* PayPal payment process should be linked with the companies PayPal account and all the payments should be deposited there.
* Shopping cart checkout also generates an order which can be viewed by the administrator to approve. Users can also be able to view the orders after signing in.
  + 1. **Back-End**
* The back-end of the application can be only accessed by the administrator.
* When administrator signed in, they will be redirected to the application back-end where they can have access to the website management functionalities.
* After signing in as administrator, an admin panel will be displayed on the side bar of the main page.
* The admin panel should have product management, order management, feedback & enquiries link.
* Upon clicking on the product management link, all the packages will be displayed. Admin can add, edit or delete packages from there.
* Every package should have an edit and delete button in front of it.
* When delete button is clicked it will delete the package.
* When edit button is click it will show the edit product page, where admin can edit the parameters of product.
* There will be a add product button on the product management page. When clicked, it will show the add product page where admin can add new products.
* Upon clicking on the order management link from the admin panel on the main page, all orders by the members will be displayed. Admin can delete or change status from there.
* Every order in the order management page should have ‘done’, ‘cancel’ and ‘delete’ button.
* When ‘done’ button is clicked, it will change the status of the order to done (completed).
* When ‘cancel’ button is clicked, it will cancel the order.
* When ‘delete’ button is clicked, it will delete the order.
* Upon clicking on the feedback and enquiries link from the admin panel on the main page, all the feedbacks, enquiries and comments from the members will be displayed.
* Each feedback, enquiry and comment will have a detail and delete button.
* When delete button is clicked it will delete and if detail button is clicked it will show the details of that feedback/enquiry/comment.
  1. **Technical Requirements**

A cross platform web application is required for an e-commerce online store which can run on all major operating system.

1. Microsoft Windows
2. Mac OS
3. Linux

The web application should support all popular web browsers.

1. Mozilla Firefox
2. Google Chrome
3. Internet Explorer
4. Safari
5. Opera

The web application can be easily host on the Linux or UNIX based hosting servers.

1. **System Design**
   1. **Database design**

The application will have the following database tables.

1. User
2. Product
3. Order
4. Comment
   * 1. **User Table**

The user table will store the information of the customers which are the member of the website, and also the information of admin.

Following are the attributes of the user table.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Comment** |
| 1 | Id | Integer | Unique Id of user |
| 2 | Email | String | Email address of user also used as username to sign in. |
| 3 | Password | String | Password of user |
| 4 | Role | String | Role of user (admin, member) |

* + 1. **Product Table**

The product table will store the information of the tour packages offered by the company to their customers.

Following are the attributes of the product table.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Comment** |
| 1 | Id | Integer | Unique Id of package |
| 2 | Name | String | Name of the package |
| 3 | Description | String | Detail information of the package and tour |
| 4 | Price | Integer | Price of the package |
| 5 | Image | String | Image of the package |
| 6 | Date | Date | Date when the tour is offered |
| 7 | Days | Integer | Number of days in the tour |
| 8 | Nights | Integer | Number of nights in the tour |

* + 1. **Order Table**

The order table will store the information of the orders placed by the customers upon checkout from the shopping cart.

Following are the attributes of the order table.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Comment** |
| 1 | Id | Integer | Unique Id of order |
| 2 | Date | Date | Date when order is placed |
| 3 | Product Name | String | Name of the package which is ordered |
| 4 | Product Price | Integer | Price of the package |
| 5 | Product Quantity | Integer | Quantity of the package ordered |
| 6 | User Id | Integer | Id of the user who have placed the order |
| 7 | Status | Integer | Status of the order (0 - pending, 1 - done, 2 - cancel) |

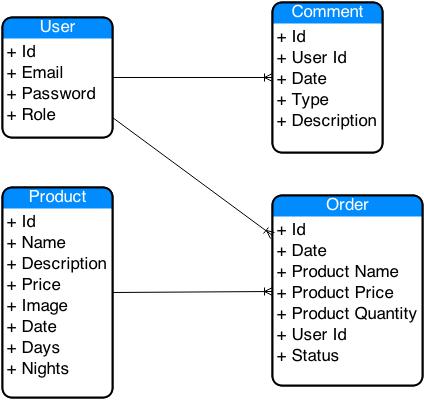
* + 1. **Comment Table**

The comment table will store the information of the feedback, enquiries and comments send from the registered users.

Following are the attributes of the comment table.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Name** | **Data Type** | **Comment** |
| 1 | Id | Integer | Unique Id of comment |
| 2 | User id | Integer | Id of user who send this comment |
| 3 | Date | Date | Date when the comment is sent |
| 4 | Type | Integer | Type of comment (0 - enquiry, 1 - feedback, 2 - comment) |
| 5 | Description | String | Message body of the comment |

* + 1. **Database diagram**



* 1. **Application Architecture**

The application is divided into three components. Following are the components.

1. Database (as described in section 3.1.)
2. REST API
3. User Interface



* + 1. **REST API**

The REST API exposes methods to access database, and to create and manipulate data through HTTP methods. Following HTTP methods are used for various actions on data.

1. POST (create and update data)
2. GET (fetch and delete data)

When a REST method is called, it establishes connection with the database and perform the required action (insert, update, select, delete). The REST API methods will be then implemented in the user interface. The REST API component works as a data source for the user interface.

The REST API uses JSON (JavaScript Object Notation) format to send data to the user interface from where it’s called.

* + 1. **User Interface**

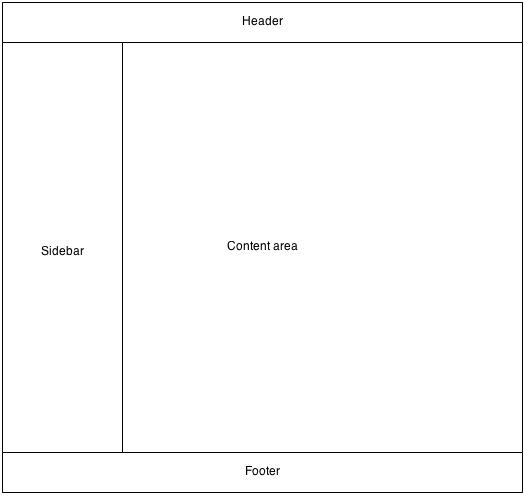
The user interface is the collection of dynamic web pages which are displayed to the user and provide interface to interact with the application. The web pages display and manipulate data with the help of REST API, used as a data source.

The REST APIs are implemented in the user interface. The web pages of the user interface component call the REST API asynchronously through HTTP request and receive data as HTTP response. The received data will be in JSON (JavaScript Object Notation) format. The web page will parse the data and display it in the HTML format.

* 1. **Interface Design**
     1. **Page Layout**

All the pages of the application have same layout. The layout consists of following elements.

1. Header
2. Footer
3. Sidebar
4. Content area



**Header –** the header of the page is fixed and displayed on all pages, it consist of website name. **Footer –** the footer of the page is fixed and displayed on all pages, it consists of company’s copyright information and web developer’s signature.

**Sidebar** – the sidebar of the page is also fixed and displayed on all pages, it consists of sign in panel, cart summary and feedback & enquiries link and package search control. If user signed in as member, then the sign in panel will replaced with member panel along with account management links. If user signed in as admin, then it will be replaced with admin panel along with account management links.

**Content area** – this area is not fixed, it displays the actual content of the page and have different views.

* + 1. **Web pages**

Following are the main web pages of the application, which will be displayed in the content area.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Page Name** | **Access level** | **Description** |
| 1 | Home | Public | Display the list of products with thumbnail image and view and add to cart button. |
| 2 | Product detail | Public | Display the detail of products including package name, tour description, number of days and nights, tour date and price, along with add to cart button. |
| 3 | Shopping cart | Public | Displays cart details including item name, quantity, unit price, total price and also grant total of all products in the cart. Also have clear cart, checkout button and continue shopping button |
| 4 | Order Management | Member/Admin | If sign in as member it will display the orders of only that account, with an option to delete any order. If sign in as admin then it will display all orders from the customers, with option to change the order status and delete. |
| 5 | Feedback & Enquiries | Member/Admin | If sign in as member it will show a form to send feedback/enquiry/comment. If sign in as admin it will show all the feedbacks, enquiries and comments from the members, with a detail view option and delete option. |
| 6 | Product Management | Admin | Display the list of products with information including package name, description, price, image, date, days, nights. Every product have edit and delete button with it, and the page will also have an add product button |
| 7 | Add Product | Admin | Display the form to add product |
| 8 | Edit Product | Admin | Display the form to edit product |

* + 1. **Design Decisions**

The interface of the application is designed according to the following factors in mind.

1. Easy to understand the content of website and what they offer, by the user.
2. Easy to use by basic level of internet users.
3. Load and run fast on browsers with intermediate level of internet connection.
4. Easy navigation between web pages.
5. Design according to a shopping cart specific application.
6. Same interface design used for website administration pages, so that there will be no inconsistency.
7. Displayed only required content is simpler way, so that the website will not be looked overloaded.
8. Web page header, footer and sidebar are fixed so that it can be displayed on every web page. It enables the user to use menus on these sections from every web page.
9. The summary of shopping cart is displayed on the sidebar which is fixed, so that the user can review its cart any time while viewing any page.
10. Package search control is placed on the top header which is fixed, so that the user can easily perform search on packages using calendar.
11. The sign in panel is placed in the fixed sidebar which will enable the user to sign in anytime when they need it while viewing any page, and also can logout anytime.
12. All list view data like product list, order list and comment list are displayed in a grid format with action buttons with each row, so that user can perform specific functions on each row.
13. **Implementation**
    1. **Development Environment**

For the development of the web application, WAMP (**W**indows **A**pache **M**ySql **P**HP) environment is used.

* **Operating System:** Microsoft Windows
* **Web Server:** Apache
* **Database:** MySql
* **Programming language:** PHP (server-side), JavaScript (client-side)
  1. **Components to Implement**

Following components are required to implement according to the designed architecture of the application.

1. Database
2. REST API
3. User Interface
   * 1. **Database**

The database model of the application is implemented in MySql database. For creation and management of database, phpMyAdmin tool is used.

The details of data models are defined in section 3.1. These models will be implemented here.

* + 1. **REST API**

PHP is used as a programming language to implement the REST API. Slim framework is used to create REST interface and handle requests and responses. PHP extension php\_mysql is used to connect with the database and perform actions on it.

‘.htaccess’ file is used to redirect all the incoming requests to the ‘index.php’ page which acts as a REST interface and handle’s all HTTP GET and POST requests.

Following are the REST API methods.

1. **Get User**

Method: /user/get

Type: POST

Response: User data in JSON format

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Email | String | Email of the user |
| Password | String | Password of user |

1. **Create User**

Method: /user/create

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Email | String | Email of the user |
| Password | String | Password of user |
| Name | String | Name of user |
| Role | String | User role |

1. **Edit User**

Method: /user/edit

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of user |
| Name | String | Name of user |
| Role | String | User role |

1. **Change Password**

Method: /user/change/password

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Email | String | Email of the user |
| Password | String | Password of user |

1. **Get Products**

Method: /products

Type: GET

Response: Product data in JSON format.

No request parameter.

1. **Get Product**

Method: /product/:id

Type: GET

Response: Product data in JSON format.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of product |

1. **Get Product by Date**

Method: /product/date/:start/:end

Type: GET

Response: Product data in JSON format.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Start date | Date | Package tour date start |
| End date | Date | Package tour date end |

1. **Delete Product by Id**

Method: /product/delete/id/:id

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of product |

1. **Edit Product**

Method: /product/edit

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of package |
| Name | String | Name of the package |
| Desc | String | Description of package |
| Price | Integer | Price of package |
| Image | String | Image location of package |
| Date | Date | Date of package |
| Days | Integer | Days of tour |
| Nights | Integer | Nights of tour |

1. **Create Product**

Method: /product/create

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Name | String | Name of the package |
| Desc | String | Description of package |
| Price | Integer | Price of package |
| Image | String | Image location of package |
| Date | Date | Date of package |
| Days | Integer | Days of tour |
| Nights | Integer | Nights of tour |

1. **Get Orders**

Method: /orders

Type: GET

Response: Order data in JSON format.

1. **Get Order by id**

Method: /order/:id

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of package |

1. **Get Order by Status**

Method: /order/status/:status

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Status | Integer | Status of the order  0-pending, 1-done, 2-cancel |

1. **Get Order by User**

Method: /order/user/:userId

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| User Id | Integer | Id of user |

1. **Delete Order by Id**

Method: /order/delete/id/:id

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Order Id |

1. **Create Order**

Method: /order/create

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Date | Date | Date when order is placed |
| P\_name | String | Name of package |
| P\_price | Integer | Price of package |
| P\_quantity | Integer | Quantity of package |
| U\_id | Integer | Id of user |
| Status | Integer | Status of the order  0-pending, 1-done, 2-cancel |

1. **Edit Order**

Method: /order/edit

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Order Id |
| Date | Date | Date when order is placed |
| P\_name | String | Name of package |
| P\_price | Integer | Price of package |
| P\_quantity | Integer | Quantity of package |
| U\_id | Integer | Id of user |
| Status | Integer | Status of the order  0-pending, 1-done, 2-cancel |

1. **Change Order Status**

Method: /order/change/status

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of order |
| Status | Integer | Status of the order  0-pending, 1-done, 2-cancel |

1. **Get Comments**

Method: /comments

Type: GET

Response: Comment data in JSON format.

1. **Get Comment by id**

Method: /comment/:id

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of comment |

1. **Get Comment by User**

Method: /comment/user/:userId

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| User Id | Integer | Id of user |

1. **Delete Comment by Id**

Method: /comment/delete/id/:id

Type: GET

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Comment Id |

1. **Create Comment**

Method: /comment/create

Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Date | Date | Date when comment is send |
| U\_id | Integer | Id of user |
| Type | Integer | Type of comment  0-enquiry, 1-feedback, 2-comment |
| Desc | String | Message body of comment |

1. **Edit Comment**

Method: /comment/edit

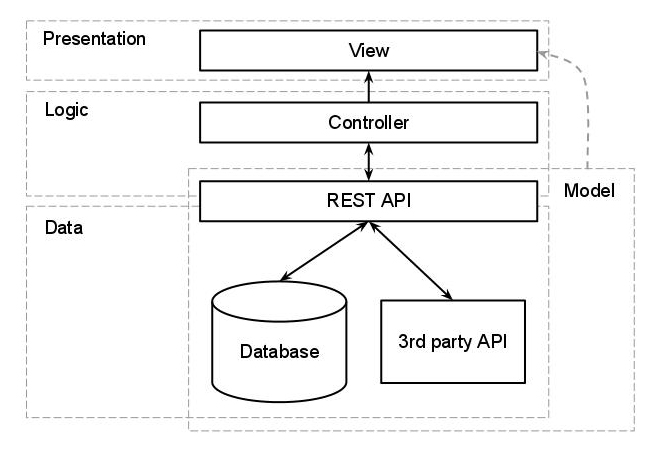
Type: POST

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Data type** | **Description** |
| Id | Integer | Id of comment |
| Date | Date | Date when comment is send |
| U\_id | Integer | Id of user |
| Type | Integer | Type of comment  0-enquiry, 1-feedback, 2-comment |
| Desc | String | Message of body |

* + 1. **User Interface**

The user interface is the dynamic web pages of the application which are implemented by using HTML5, JavaScript and CSS. The user interface is implemented as a single page application for which an open source JavaScript framework AngularJS is used.

AngularJS is a framework managed by Google. The library of AngularJS provides additional tag attributes to the HTML which makes it dynamic.

The user interface is developed using MVC pattern with the help of AngularJS. The Views are in the form of HTML pages, where as controllers are written in JavaScript. The REST API is used as a data source for Model.

The template of the application uses Bootstrap front-end framework for responsive design and also jQuery (an open source JavaScript library). The application design is responsive and can also be viewed on tablet and mobile devices.

Following frameworks and libraries are used to develop user interface.

* AngularJS (http://angularjs.org/)
* Bootstrap (http://getbootstrap.com/2.3.2/)
* jQuery (<http://jquery.com/>)
  1. **Platforms & Tools Used in Development**

**1. Programming Languages**

* PHP
* JavaScript

**2. Database**

* MySql

**3. Tools**

* NetBeans IDE for PHP (REST API)
* WebStrom for JavaScript (User Interface)
* phpMyAdmin (Database)

**4. Frameworks**

* Slim framework for REST API
* AngularJS for User Interface

**5. Webserver**

* Apache
  1. **Production Environment**

The production environment of the application will be a hosted server having Linux or UNIX based server operating system with Apache web server running on it. The operating system also have MySql database where the database of the application will be deployed. The application files and all resources will be deployed on the Apache webserver.

The production environment can be a LAMP (**L**inux **A**pache **M**ySql **P**HP) or WAMP based server.

1. **Conclusion**

The application is developed to full fill the requirements of a travel company who wants to sale there tour packages online through an e-commerce store. The application is developed specifically to sale the travel tour packages. The tour packages are used as a product to sale.

The application is well designed for simple tour packages but it can’t handle complex packages.

The checkout process of the application uses PayPal which provide easy integration and secure payment process.

The architecture of application is based on REST APIs which can be easily accessed from any type of user interface. Currently the user interface is provided as web pages but it can be a desktop application or mobile application.

The web pages are developed as a single page application. The whole application runs as a single page in the browser which minimizes the delay of page loads and run faster and provides easy navigation. The pages call REST API asynchronously through HTTP.

* 1. **Future Work**
* The application is developed for relatively small store and for simple tour packages. For complex tour packages some changes in all the components will be required.
* The REST API needs some more security features.
* 3rd party user authentication will be added so that users having Facebook/Google/Hotmail etc. account can also be signed in.
* Email based verification will be added upon new user registration.

1. **User Manual**

Following are the steps to deploy the application.

* Web application folder in the CD contains the application.
* Install WAMP package on the system.
* Copy the folders ‘TravelCart’ and ‘travelcart-api’ and paste it on the www directory of the WAMP server.
* ‘travelcart.sql’ file contain the database script.
* Using phpMyAdmin of create a new database with name ‘travelcart’ and run all the sql scripts in the travelcart.sql file on the database. It will create the required table and data.
* Configure the database parameters by editing the file ‘travelcart-api/cfg/config.php’.
* In Apache modules enable the ‘rewrite\_module’
* Now run the application on the browser with the link <http://localhost/TravelCart/> (if the WAMP server is deployed on localhost)

Following accounts can be used for testing.

User: admin (administrator account)   
Password: 123

User: customer (member account)   
Password: 123