(2.Ado.Ok

UNDERGRADUATE THESIS - KI141502

DESIGN AND IMPLEMENTATION OF TRAVEL INFORMATION SYSTEM IN FIJI USING REST-API CONCEPT

NEMESIO R RAITUBU
NRP 05111640007006

Thesis Supervisor
Radityo Anggoro, S.Kom., M.Sc.

INFORMATICS DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

UNDERGRADUATE THESIS – KI141502

DESIGN AND IMPLEMENTATION OF TRAVEL INFORMATION SYSTEM IN FIJI USING REST-API CONCEPT

NEMESIO R RAITUBU
NRP 05111640007006

Thesis Supervisor
Radityo Anggoro, S.Kom., M.Sc.

INFORMATICS DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[This page has been intentionally left blank]

APPROVAL SHEET

DESIGN AND IMPLEMENTATION OF TRAVEL INFORMATION SYSTEM IN FIJI USING REST-API CONCEPT

UNDERGRADUATE THESIS

Submitted to Meet One Provision
Obtaining a Bachelor of Computer Degree
on
Undergraduate Program of Informatics Department
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By :
NEMESIO R RAITUBU
NRP : 05111640007006

Approved by Thesis's Supervisor :

Radityo Anggoro, S.Kom., M.Sc.
NIP: 19841016 2008121002

.....
(Supervisor)

SURABAYA
APRIL 2020

[This page has been intentionally left blank]

DESIGN AND IMPLEMENTATION OF TRAVEL INFORMATION SYSTEM IN FIJI USING REST-API CONCEPT

Name	: NEMESIO R RAITUBU
NRP	: 05111640007006
Major	: Informatics Department – ITS
Supervisor	: Radityo Anggoro, S.Kom., M.Sc.

Abstract

A Travel information system would play a vital role in planning the perfect trip while on business or personal tourist visit to Fiji. Fiji Island is one of the tourist destinations that in the list of travellers because of its friendly people and heavenly tropical islands, Fiji is the quintessential South Pacific paradise. The main purpose of this flight system is to help individual to manage customer domestic flights. The proposed system maintains centralized repository to make necessary travel arrangements and to retrieve information easily.

Fiji is one of the top tourist destinations in the world and the government has invested a lot of money into the tourism industry for the construction of many hotels to attract tourists to choose Fiji as their holiday destination. Therefore, Travel Information System in Fiji will be very useful for the tourists to use in booking flights around Fiji flying to different destinations

Keywords: Information System, Web, Travel, VueJs, REST-API, Laravel.

[This page has been intentionally left blank]

ACKNOWLEDGMENT

Writer gives Praise and gratitude to the Almighty God for the gracious mercy and tremendous blessing that enables the writer to accomplish the final project entitled ““Design and Implementation of Travel Information System in Fiji”.

On the process of doing this project, I am fully aware that there are many parties involved in supporting this project to be a success. Therefore. I would like to express special appreciation and gratitude to the followings:

1. Writer’s parents, brothers, sisters and relatives that always gives moral and spiritual support during my bachelor study.
2. Mr. Radityo Anggoro, S.Kom., M.Sc., as my supervisor who always give great help, advice. And taught me in order to accomplish this final project.
3. Bapak Rizky Januar Akbar, S.Kom., M.Eng. as a supportive lecturer who has helped, guided, and provided his knowledge to the completing of this Final Project.
4. Darlis Herumurti, S.Kom., M.Sc., as the Head of Informatics Department and also mu guardian lecturer which he helps me and guarded me through the whole of my studies and enabling the completion of this final project.
5. Adetiya Bagus class of 2014 was a great and supportive friend of mine in doing this final project.

6. Writer would like take this opportunity to give thanks and appreciation to all the hardworking lecturers of Informatics Engineering Department who have taught me, and support me in one way or the other during my bachelor program.
7. The writer also would like to say a special thanks to the Head of International Office ITS and all her hardworking staffs and volunteers who support me during the time of my bachelor program
8. Last but not the least, the writer would like to take this time to say thank you to ITS Rector, all the ITS staffs, and the Indonesian Government for giving me the opportunity to study in ITS.
9. Everyone else that cannot be mention one by one in making this research a success, may God repay all the good deeds.

The writer also fully aware that this report still misses a lot of part. Therefore, any critics and suggestion regarding this report is welcome. Hopefully this final project report can give benefit to the readers and please forgive me if there any unpleasant words and mistake during the process of making this final report.

Surabaya, April 2020

Writer

TABLE OF CONTENTS

LIST OF PICTURES

LIST OF TABLES

[This page has been intentionally left blank]

CHAPTER I
INTRODUCTION

This chapter discusses the outline of the preparation of the final project which includes the background, the purpose of making it, the formulation and boundaries of the problem, the methodology for the preparation of the final project, and the systematic writing.

1.1 Background

Nearly everyone goes on a vacation and a Travel information system would play a vital role in planning the perfect trip. The travel information system allows the user of the system access all the details such as hotels and flights. The main purpose is to help individual to manage customer, flights and hotels. The system can also be used for both professional and business trips. The proposed system maintains centralized repository to make necessary travel arrangements and to retrieve information easily.

With the rapid economic development in Fiji, the government has invested a lot of money into the tourism industry for the construction of many hotels to attract tourists to choose Fiji as their holiday destination. Actually there are some website that give informations about Fiji holiday destinations, for example : Tourism Fiji (https://www.fiji.travel/), U.S. News (https://travel.usnews.com/Fiji/), etc. But, this information only available on traditional web platform. In the last few years, most of all application is built on multiplatform devices. It will make easy for users to choose the platform that they want to be used. Web service architecture is the solution to create multiplatform apps. It can be accessed through certain protocol standards in the platform and interface of an independent programming language.
In software development, we also need to bear in mind specific modeling, deployment and marketing needs, and it also requires coping with a complex relationship among stakeholders.

This research will implement REST API (Representational State Transfer Application Programming Interface). One of the key advantages of REST APIs is that they provide a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia. This flexibility allows developers to build an API that meets your needs while also meeting the needs of very diverse customers. Unlike SOAP, REST is not constrained to XML, but instead can return XML, JSON, YAML or any other format depending on what the client requests. And unlike RPC, users aren't required to know procedure names or specific parameters in a specific order.

1.2 Problem Formulation

The formulation of the issues raised in this Final Project can be explained as follows.

1. How to design a traveling information system to be used in Fiji island for domestic flights booking.
2. How to implement this domestic traveling booking system using REST API
3. What are the requirements needed to make Fiji Travel Information System running well in Fiji

1.3 Problem Limitation

The problems discussed in the final project have several limitations, which are as follows.

1. Fiji travel Information System use for domestic flights booking
2. Fiji travel Information System will' be in web app.

1.4 Goals

The goals of this final project are as follow.

1. To implement a travel information system in Fiji.
2. The travel information system to make it easy for booking of domestic flights around Fiji island

1.5 Methodology

The steps taken to complete this Final Project are as follows:

a. **Completion of the final project**

   This final project proposal contains a preliminary description of the final project to be made. This introduction consists of the background of the proposed final project, the formulation of the issues raised, the problem boundaries for the final project, the purpose of making the final project, and the benefits of the results of the final project. In addition, it also describes the literature review which is used as a reference to support the final project. The methodology sub-section contains an explanation of the stages of preparing the final project starting from the preparation of proposals to the preparation of the final project. There is also a sub-chapter of the activity schedule that explains the final work schedule.

b. **Literature Review**

   At this stage the researcher collects information by conducting a study towards research that raises topics around development information systems with API concepts and REST architectural styles. Researchers to do direct communication and discuss with the employee section administration to one of the transport providers. This matter done to avoid possible mistakes in previous studies and strengthening the concepts used in research with similar cases.

c. Software Analysis and Design

At this stage an analysis and design of the academic information system design will be made. The analysis is carried out by determining the functional requirements of the system and designing the system design is done by making a database design, mock up the appearance of web pages and diagrams needed.

d. Implementation

The implementation phase in this research is carried out in several sub-processes, among others:

1. Building a system development environment.
2. Designing the structure of the table, functions and stored procedures on database.
3. Develop a back-end (coding) system and adjust it to database.
4. Design the rewrite structure on the web server.

e. Testing and Evaluation

To ensure that the system runs according to the development plan facilitated business systems and processes, a testing scenario is needed system. In this study, testing will be divide--ud into two test scenarios, including:

1. Web Apps Testing

Software testing is a test that focuses on the functional specifications of the web apps. This test is done to test whether the application is running well or not.

2. Mobile Apps Testing

Software testing is a test that focuses on the functional specifications of the web apps. This test is done to test whether the application is running well or not.

f. Completion of Thesis

At this stage, a book was prepared that made documentation regarding the design and implementation of travel information system in Fiji.

1.6 Preparation of Thesis

This thesis book consists of several chapters that are arranged systematically, as follows.

1. Chapter 1. Introduction
   The introductory chapter contains an explanation of the background of the problem, problem formulation, problem boundaries, objectives, benefits and systematic writing of the final project.
2. Chapter II. Literature review
   The literature review chapter contains an explanation of the theoretical basis that supports the completion of the final project.
3. Chapter III. Analysis and Design
   This chapter contains system design, database design, use case diagrams, activity diagrams and user interface designs.
4. Chapter IV. Implementation
   This chapter discusses the implementation of the designs made in the previous chapter. Explanation in the form of code used for the implementation process.
5. Chapter V. Trials and Evaluations
   This chapter explains the ability of the software by testing the truth and testing the performance of the system that has been made.
6. Chapter VI. Conclusions and recommendations
   This chapter is the last chapter that provides conclusions from the results of trials conducted and suggestions for future software development.

[This page has been intentionally left blank]

CHAPTER II
LITERATURE REVIEW

The literature review chapter contains explanations of theories related to software implementation. The explanation aims to provide an overview of the system to be built and is useful as a support in design and implementation of travel information system.

2.1 REST API

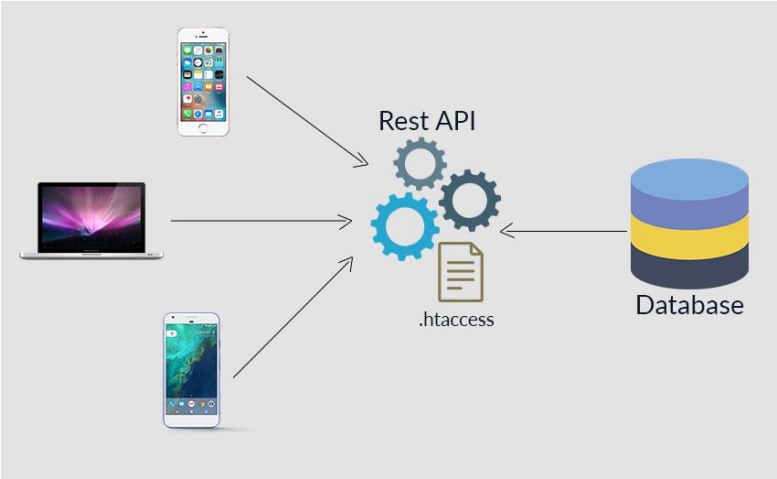


Figure 2-1 REST API Concept

A REST API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. A RESTful API is also referred to as a RESTful web service. It is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development. By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components

2.2 Laravel

Figure 2-2 Laravel Logo

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern and based on Symfony. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.Indeks

With the rise of mobile development and JavaScript frameworks, using a RESTful API is the best option to build a single interface between data. Laravel framework is very opinionated and strives to save developer time by favoring convention over configuration. The framework also aims to evolve with the web and has already incorporated several new features and ideas in the web development world—such as job queues, API authentication out of the box, real-time communication, and much more. In this thesis, Laravel is used for the backend of the system and also used for the web application user interface.

2.3 Vue JS

Figure 2-3 VueJS Logo

Vue is a simple and minimal progressive JavaScript framework that can be used to build powerful web applications incrementally. Vue is a lightweight alternative to other JavaScript frameworks like AngularJS. With an intermediate understanding of HTML, CSS and JS, you should be ready to get up and running with Vue.

Vue.js is an open-source Model–view–view model JavaScript framework for building user interfaces and single-page applications. It was created by Evan You, and is maintained by him.

Vue uses an HTML-based template syntax that allows binding the rendered DOM to the underlying Vue instance's data. All Vue templates are valid HTML that can be parsed by specification-compliant browsers and HTML parsers. Vue compiles the templates into virtual DOM render functions. A virtual Document Object Model (or “DOM”) allows Vue to render components in its memory before updating the browser. Combined with the reactivity system, Vue is able to calculate the minimal number of components to re-render and apply the minimal amount of DOM manipulations when the app state changes.

Vue users can use template syntax or choose to directly write render functions using JSX.Render functions allow application to

be built from software components. In this project, Vue JS is used for mobile interface for the mobile application.

.

2.4 MySQL Database

Figure 2-4 MySQL Logo

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modelling techniques.

MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. MySQL also provides connectors and drivers (ODBC, JDBC, etc.) that allow all forms of applications to make use of MySQL as a preferred data management server.

In this project, MySQL stores all the data regarding Fiji Travel Information System.

2.5 PHP

Figure 2-5 PHP Logo

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

PHP is a popular general-purpose scripting language that is especially suited to web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group.

During 2014 and 2015, a new major PHP version was developed, which was numbered PHP 7. The numbering of this version involved some debate. While the PHP 6 Unicode experiment had never been released, several articles and book titles referenced the PHP 6 name, which might have caused confusion if a new release were to reuse the name. After a vote, the name PHP 7 was chosen.

REST APIs are the backbone of modern web development. Most web applications these days are developed as single-page applications on the frontend, connected to backend APIs written in various languages. There are many great frameworks that can help you build REST APIs quickly. Laravel/Lumen and Symfony's API platform are the most often used examples in the PHP ecosystem. They provide great tools to process requests and generate JSON

responses with the correct HTTP status codes. They also make it easy to handle common issues like authentication/authorization, request validation, data transformation, pagination, filters, rate throttling, complex endpoints with sub-resources, and API documentation.

2.6 JSON

Figure 2-6 JSON Logo

In computing, JavaScript Object Notation is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types. It is used mostly in formatting each data that are requested in different programs.

2.7 MVC (Model-View-Controller)

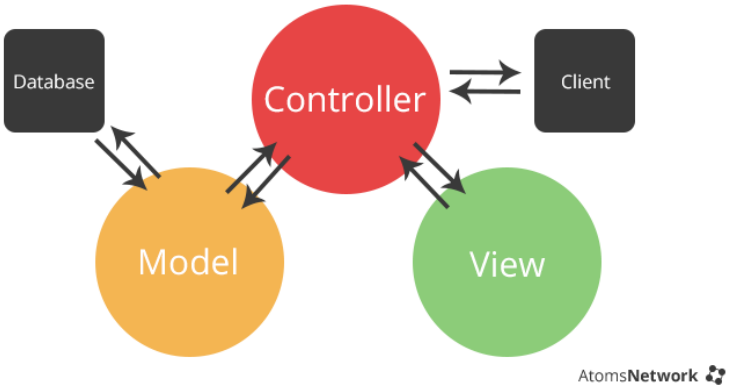


Figure 2-7 MVC Architecture

Model–view–controller (usually known as MVC) is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the way's information is presented to and accepted from the user. This kind of pattern is used for designing the layout of the page.

Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web applications' (Model-View-Controller). There are three main components of MVC:

- Model: The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
- View: Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

- Controller: Accepts input and converts it to commands for the model or view

2.8 MVVM (Model-View-View Model)

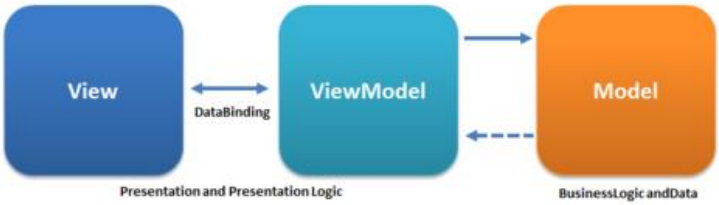


Figure 2-8 MVVM Architecture

Model–view–viewmodel (MVVM) is a software architectural pattern that facilitates a separation of development of the graphical user interface – be it via a markup language or GUI code – from development of the business logic or back-end logic (the data model). The view model of MVVM is a value converter, meaning the view model is responsible for exposing (converting) the data objects from the model in such a way that objects are easily managed and presented. In this respect, the view model is more model than view, and handles most if not all of the view's display logic. The view model may implement a mediator pattern, organizing access to the back-end logic around the set of use cases supported by the view.

MVVM is a variation of Martin Fowler's Presentation Model design pattern. MVVM abstracts a view's state and behavior in the same way,but a Presentation Model abstracts a view (creates a view model) in a manner not dependent on a specific user-interface platform.

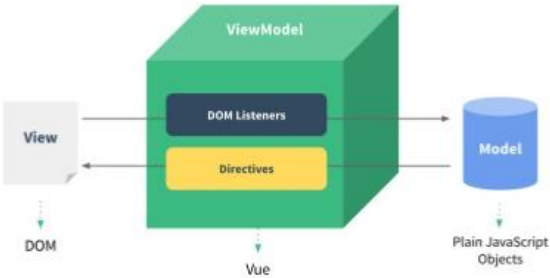


Figure 2-9 MVVM Pattern in VueJS

In MVVM, the Model simply represents the data access layer of the application. It holds the data/information which is to be presented to the user for manipulation or interaction. The Model has no behavior or logic defined on it in any way apart from data validation. The Model also has no means to access a backend or 3rd Party API to generate or save data — it simply serves as a container to hold the information/data which the VM retrieves and uses.

The View is used to render the information contained in the Model to the user. In MVVM, the View doesn't know about the Model and vice-versa. The View knows about the VM and is therefore known as an 'active' View. All user action e.g. user input Is intercepted by the View and passed to the VM for processing. The View doesn't maintain any state rather it simply represents 'state' as defined by the VM.

The VM is the link between the Model and the View. All logic required for manipulating the data contained in the Model is defined on the VM and all logic which the View uses to handle user interaction or format the data from the Model are provided to it from the VM. Unlike other MV* implementations, all Business Logic which the app requires to function are defined on the VM and not the Model.

[This page has been intentionally left blank]

CHAPTER III
ANALYSIS AND DESIGN

In this chapter, We will discused about the design of the system that will be developed. The system design will include the analysis of the requirements that needed in the system after getting a generic business process. The design of this system will be represented by various Unified Modeling Languages (called UML).

3.1 Analysis

The analysis phase is divided into several parts, including analysis of business process references, use cases of the system, and software requirements. In general, the steps above can be explained with **Error! Reference source not found.**

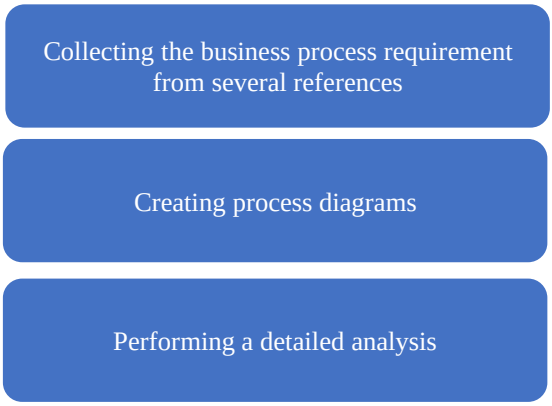


Figure 3-1 Software Requirements Analysis Phase

3.1.1 Software Requirements Specifications

Based on the description of scope of the software that will be developed, it is necessary to have a software specification in order to provide solutions to the problems given and be able to accommodate the needs. It is expected that this specification can

adjust to user needs. Software requirements specification in this thesis consists of functional requirements which can be seen in Table 3.1 and Table 3.2

3.1.2 Functional Requirements

Table 3.1 Functional Table of the Software

No	Functional Requirements	Description
1	Create travel destination	Create new travel destination
2	Update travel destination	Update existing travel destination
3	View travel destination	View the list and detail of travel destination
4	Manage Users	Manage all of registered users
5	View dashboard	View the informations on dashboard
6	Register	Register for a new user
7	Login	Login for registered user based on roles
8	Search travel destination	Search the travel destinations
9	Manage booking cart	Manage booking cart
10	Checkout	Checkout items on cart
11	View tickets	View detail of tickets that have been ordered
12	Manage user profile	Manage detail of user profile

3.1.3 Actors

Actors are people who interact with system. They have different roles and privileges. In this information system, there are four actors that explained on Table 3.2

Table 3.2 Table Actors of System

No	Actor	Description
1	Admin	• Create travel destination • Update travel destination • View travel destination • Manage Users • View dashboard
2	Customer	• Register • Login • Search travel destination • Manage booking cart • Payment • View tickets • Manage user profile

3.1.4 Use Case Diagram

This section explains the detail of the use cases contained in the software as shown in Figure 3.2. There are also table description and activity diagrams for each use case.

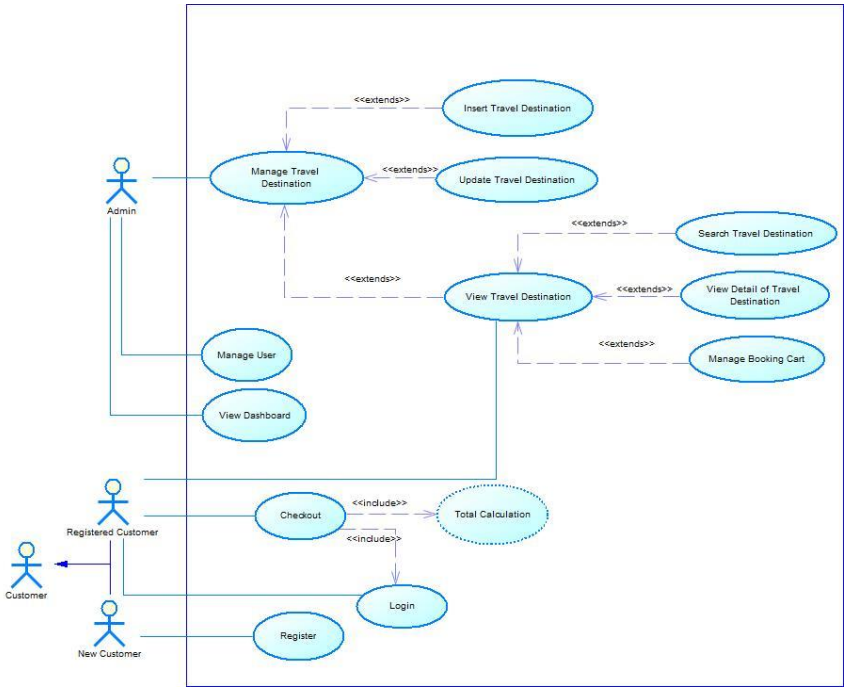


Figure 3.1 Fiji Travel Information System Use Case Diagram

3.1.4.1 Create Travel Destinations Use Case

In this use case, Admin can create new travel destinations that shown on the apps, Then, this travel destinations can be ordered by registered customer. Detail of the use case is shown on Table 3.5 and the activity diagram is shown on Figure 3.3

Table 3.3 Detail of Create Travel Destination's Use Case

Components	Description
Name	Create travel destination
Code	UC-001
Description	This usecase is used to create new travel destination.

Components	Description
Type	Functional
Actor	Admin
Initial Condition	Travel destination has not been added in the system.
End Condition	Travel destination has successfully added in the system.
Normal Flow	1. Actor choose destinations menu 2. System show the list of destinations 3. Actor click Create Destination Button 4. System show the form to create new destination. 5. Actor fill the form to create new travel destination 6. System save the new data
Alternate Flow	-

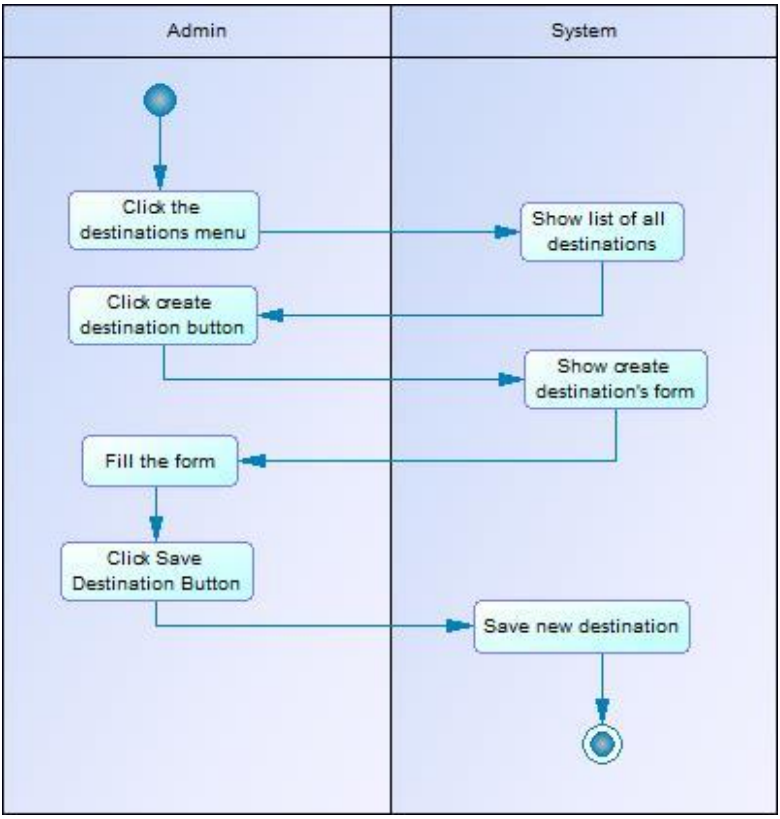


Figure 3.3 Create Travel Destination Activity Diagram

3.1.4.2 Update Travel Destination Use Case

In this use case, Admin can update the existing travel destinations that shown on the apps, Then, this travel destinations can be ordered by registered customer. Detail of the use case is shown on Table 3.6 and the activity diagram is shown on Figure 3.4.

Table 3.4 Detail of Update Travel Destination's Use Case

Components	Description
Name	Update travel destination
Code	UC-002
Description	This usecase is used to update travel destination
Type	Functional
Actor	Admin
Initial Condition	Travel destination has not been changed.
End Condition	Travel destination has successfully updated
Normal Flow	1. Actor choose destinations menu 2. System show the list of destinations 3. Actor choose the travel destination that want to be changed 4. Actor click update button 5. System show the form to update the destination. 6. Actor fill the form to update travel destination 7. System save the updated data
Alternate Flow	-

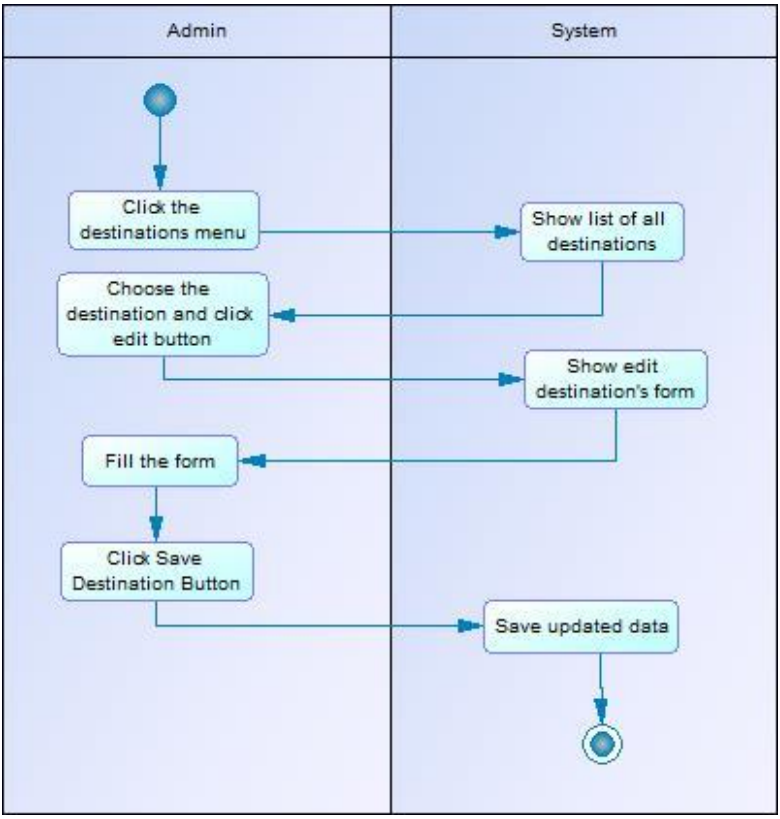


Figure 3.2 Update Travel Destination Activity Diagram

3.1.4.3 View Travel Destination Use Case

In this use case, Admin can view the detail of existing travel destinations that shown on the apps. Detail of the use case is shown on Table 3.7 and the activity diagram is shown on Figure 3.5.

Table 3.5 Detail of View Travel Destination's Use Case

Components	Description
Name	View travel destination
Code	UC-003
Description	This usecase is used to view the detail of existing travel destination
Type	Functional
Actor	Admin
Initial Condition	User choose the destination menu
End Condition	System successfully show the detail of destination
Normal Flow	1. Actor choose destinations menu 2. System show the list of destinations 3. Actor choose which travel destination that want to be viewed 4. Actor click show button 5. System show the detail of destination.
Alternate Flow	-

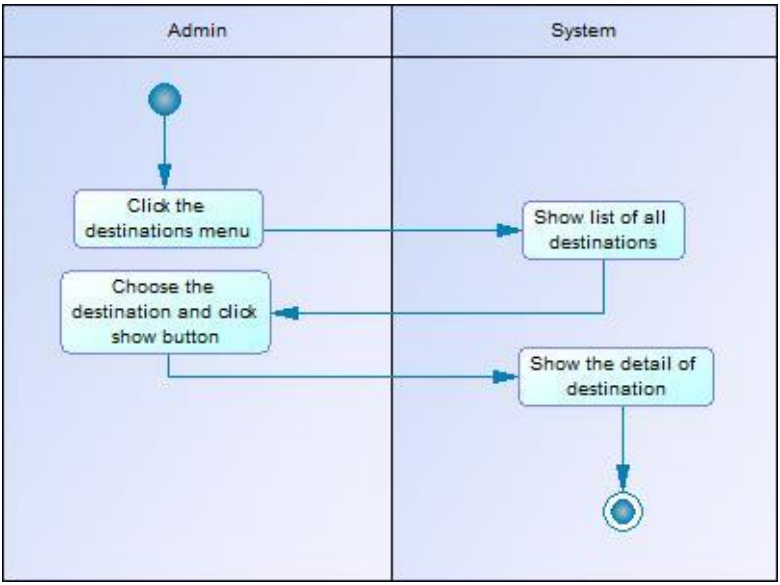


Figure 3.3 View Travel Destination Activity Diagram

3.1.4.4 Manage Users Use Case

In this use case, Admin can manage the user's data. Admin can update the data of all existing user. Detail of the use case is shown on Table 3.8 and the activity diagram is shown on Figure 3.6.

Table 3.6 Detail of Manage Users' Use Case

Components	Description
Name	Manage Users
Code	UC-004
Description	This use case is used to manage the users on system
Type	Functional
Actor	Admin
Initial Condition	User has registered on system

Components	Description
End Condition	Admin has successfully update the user's data
Normal Flow	1. Actor choose manage users menu 2. System show the list of users 3. Actor choose the user that want to be changed 4. System show the form to update the user. 5. Actor fill the form to update user's data 6. System save the updated data
Alternate Flow	-

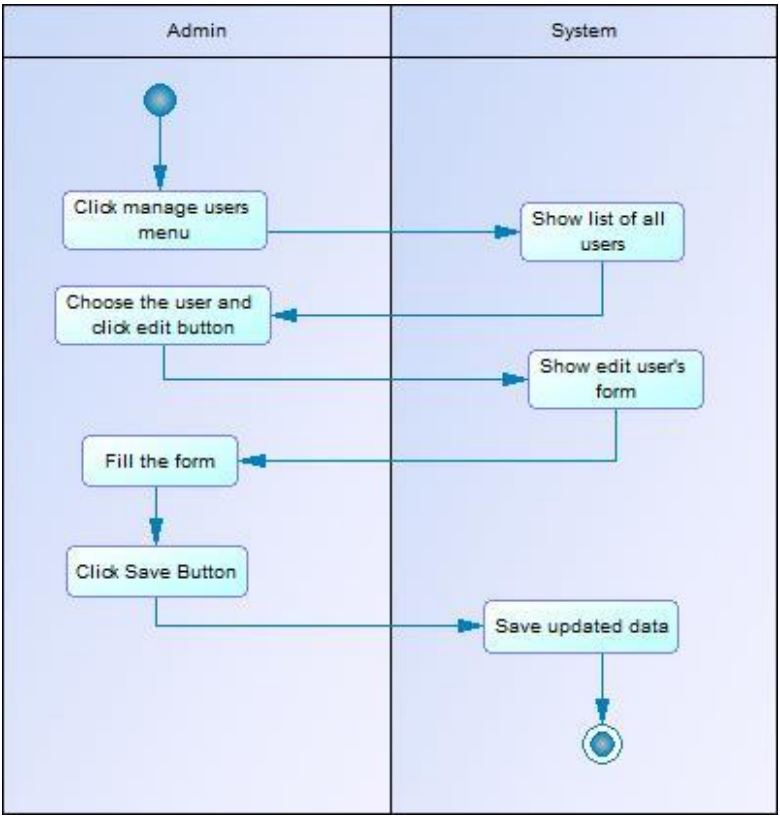


Figure 3.4 Manage Users Activity Diagram

3.1.4.5 View Dashboard Use Case

In this use case, Admin can view the dashboard after login. Detail of the use case is shown on Table 3.9 and the activity diagram is shown on Figure 3.7.

Table 3.7 Detail of View Dashboard use Case

Components	Description
Name	View Dashboard

Components	Description
Code	UC-005
Description	This use case is used to view the dashboard of the application
Type	Functional
Actor	Admin
Initial Condition	User is exist on the database and has role as admin
End Condition	System redirect the user to dashboard
Normal Flow	1. System shows the login page 2. User enter his credential as an admin 3. System redirect the user to the dashboard
Alternate Flow	1. System redirect user to login page

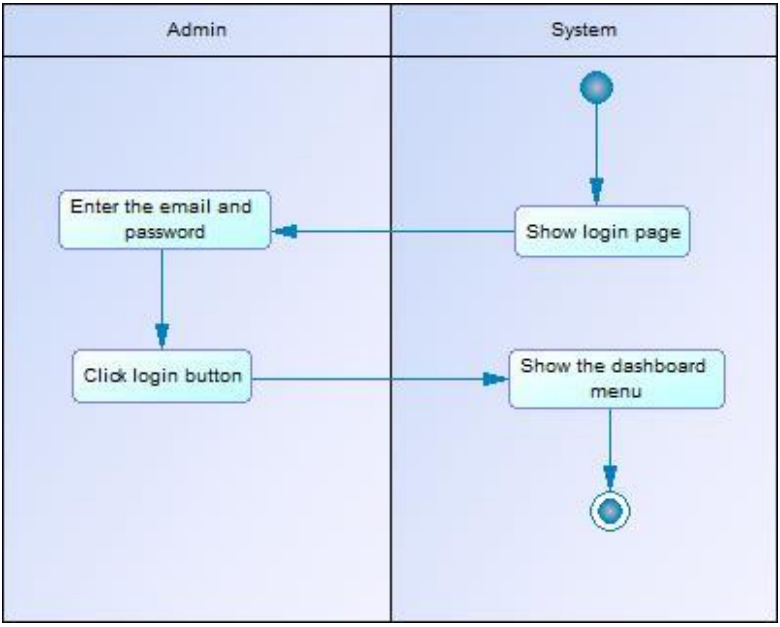


Figure 3.5 View Dashboard Activity Diagram

3.1.4.6 Register Use Case

In this use case, guest can register to system. User must register to system to book the travel destination. Detail of the use case is shown on Table 3.10 and the activity diagram is shown on Figure 3.8.

Table 3.8 Detail of Register Use Case

Components	Description
Name	Register
Code	UC-006
Description	This usecase is used to register for a new user
Type	Functional
Actor	Guest
Initial Condition	User has not been registered on system
End Condition	User successful registerd on system
Normal Flow	1. Actor click register button 2. System show registration form 3. Actor fill the registration form 4. System save the registration data
Alternate Flow	3.1. Actor failed to fill the form 1. System redirect to registration form and give the warning message

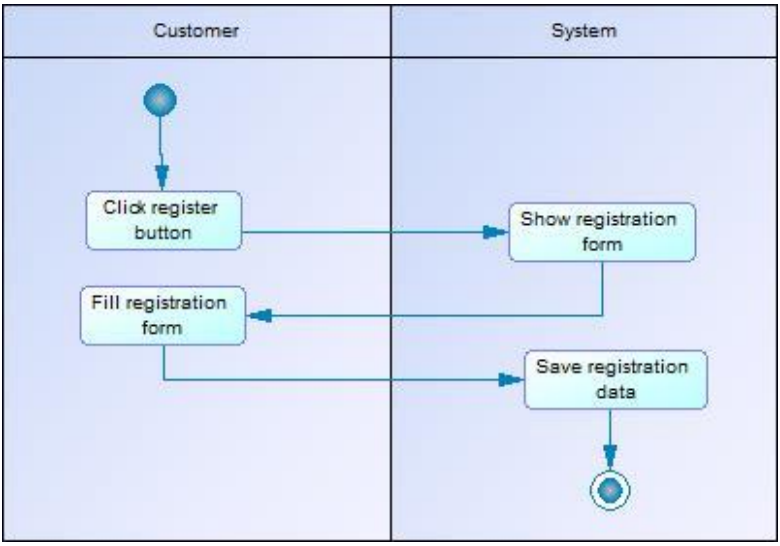


Figure 3.6 Register Activity Diagram

3.1.4.7 Login Use Case

In this use case, registered user can login to the system. There are two roles of users: admin and customer. Each of role has different capabilities and permissions. The detail of use case is shown on Table 3.11 and the activity diagram is shown on Figure 3.9.

Table 3.9 Detail of Login Use Case

Components	Description
Name	Login
Code	UC-007
Description	This usecase is used to login into system for registered users
Type	Functional

Components	Description
Actor	Registered User
Initial Condition	User has not logged in into system
End Condition	User has successfully login into system
Normal Flow	1. Actor go to login page 2. Actor enter his credential 3. System redirect the user into system based on role
Alternate Flow	3.2. Actor login as admin 1. System redirect into admin page 3.3 Actor login as customer 1. System redirect into customer page

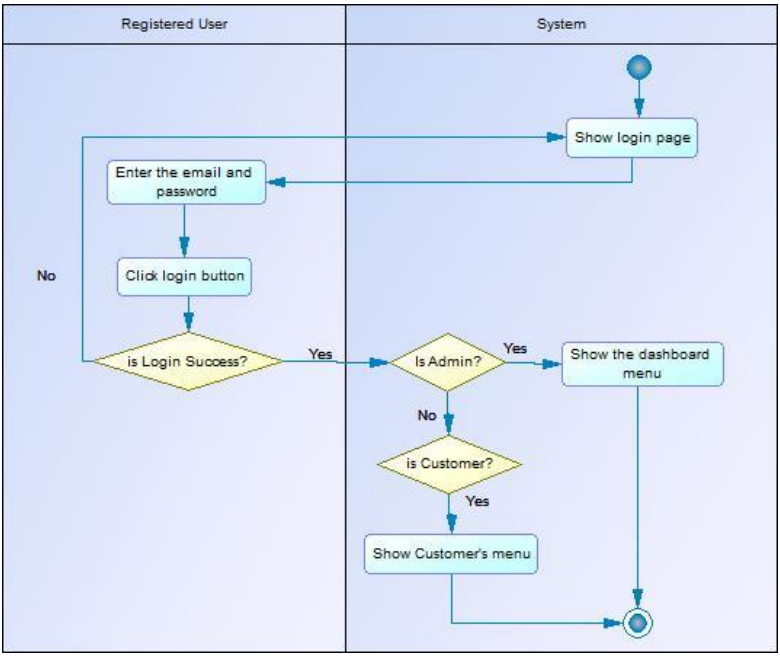


Figure 3.7 Login Activity Diagram

3.1.4.8 Search Travel Destinations Use Case

In this use case, Customer can search the travel destinations that they want, Then, this travel destinations can be ordered by registered customer. Detail of the use case is shown on Table 3.12 and the activity diagram is shown on Figure 3.10.

Table 3.10 Detail of Search Travel Destinations Use Case

Components	Description
Name	Search travel destination
Code	UC-008
Description	This use case is used to sear travel destinations on system
Type	Functional
Actor	Cutomer
Initial Condition	System show the homepage
End Condition	System show the destinations based on keyword entered by actor
Normal Flow	1. Actor go to search destination tab content 2. Actor enter the keyword of the destination 3. Actor click search button 4. System show the result
Alternate Flow	4.1 The entered keyword is match with the existing destination on system 1. System show the destinations 4.2 The entered keyword is not match with any existing destination on system 1. System show the warning text

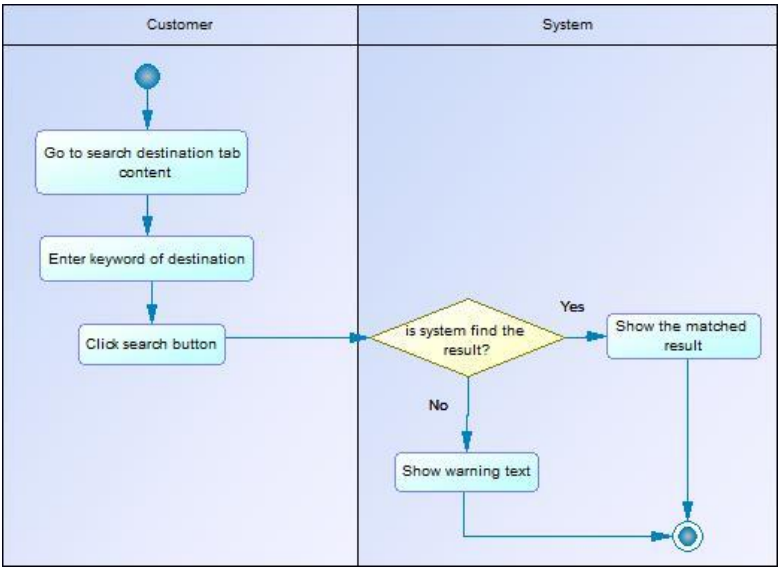


Figure 3.8 Search Destination Activity Diagram

3.1.4.9 Manage Booking Cart Use Case

In this use case, Customer can manage their cart Customer can add destination on the cart and can clear the cart before they can purchase it. Detail of the use case is shown on Table 3.13 and the activity diagram is shown on Figure 3.11.

Table 3.11 Detail of Manage Booking Cart Use Case

Components	Description
Name	Manage booking cart
Code	UC-009
Description	This use case is used to manage the customer's booking cart
Type	Functional
Actor	Customer
Initial Condition	The cart is empty

Components	Description
End Condition	Actor successfully add destination on cart
Normal Flow	1. Actor choose the destination 2. Actor click Add to Cart Button 3. The destination has added into cart 4. Actor click the cart icon 5. System show the actor's cart
Alternate Flow	4.1 User want to clear the cart 1. User click clear cart button 4.2 User want to purchase the cart 1. User click continue to payment button

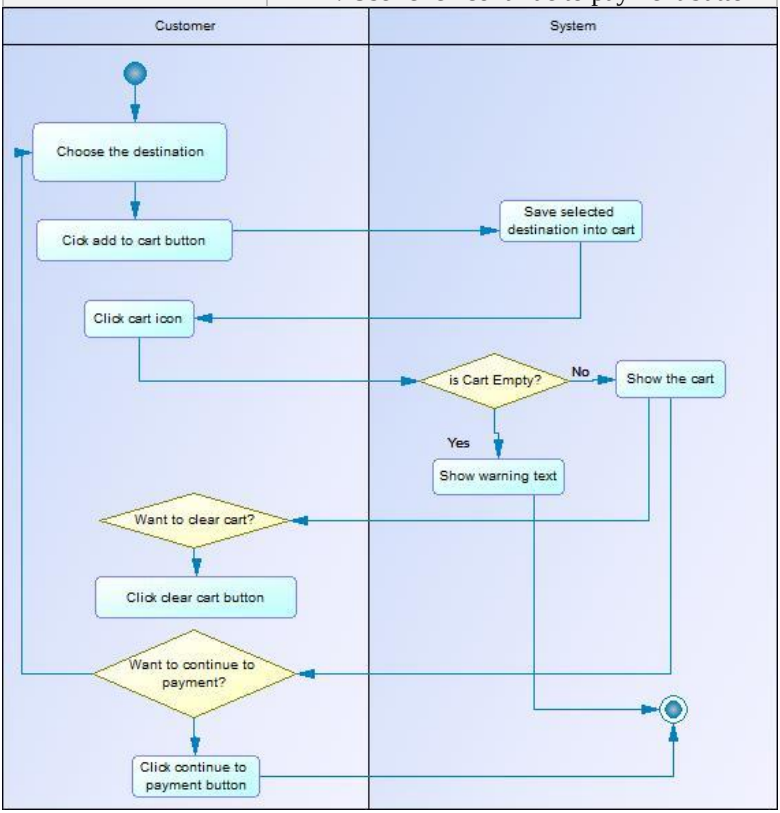


Figure 3.9 Manage Booking Cart Activity Diagram

3.1.4.10 Payment Use Case

In this use case, Customer can pay the existing destinations on his cart. Detail of the use case is shown on Table 3.14 and the activity diagram is shown on Figure 3.12.

Table 3.12 Detail of Payment Use Case

Components	Description
Name	Payment
Code	UC-010
Description	This use case is used to make customer's payment
Type	Functional
Actor	Customer
Initial Condition	The cart is not empty
End Condition	System show the detail of payment
Normal Flow	1. Actor click the cart icon 2. Actor click continue to pay 3. System show the detail of payment 4. Actor gets the ticket
Alternate Flow	-

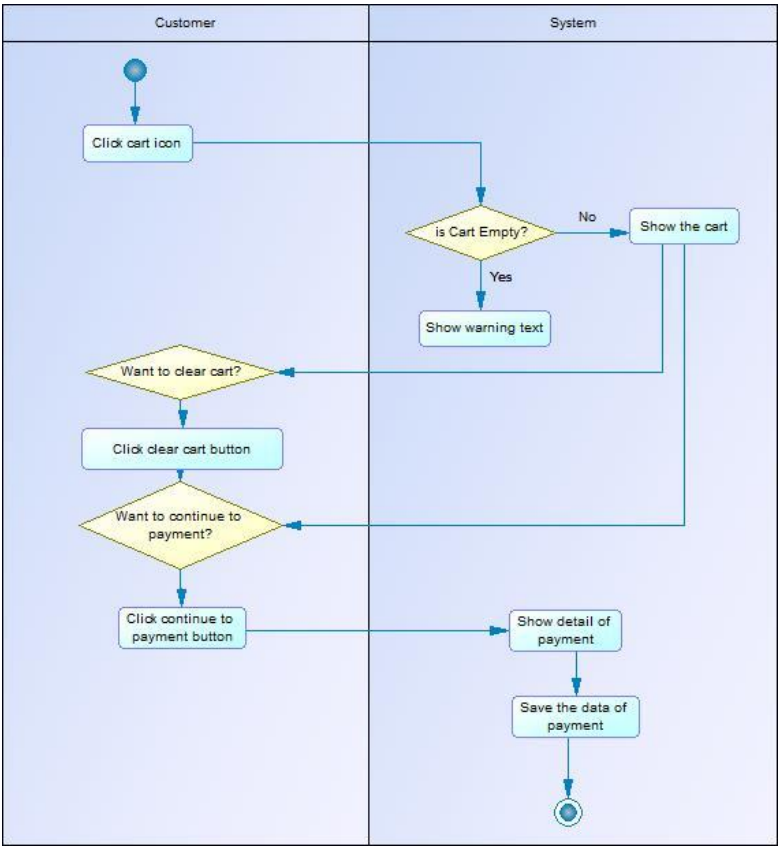


Figure 3.10 Payment Activity Diagram

3.1.4.11 View Tickets Use Case

In this use case, Customer can view their booking ticket Detail of the use case is shown on Table 3.15 and the activity diagram is shown on Figure 3.13.

Table 3.13 Detail of View Tickets Use Case

Components	Description
Name	View tickets
Code	UC-011
Description	This usecase is used by customers to view their tickets
Type	Functional
Actor	Customer
Initial Condition	Payment is complete
End Condition	Actor can view the ticket
Normal Flow	1. Actor click My Tickets menu 2. System show the list of actor's ticket 3. Actor click show ticket button 4. System show the detail of ticket
Alternate Flow	-

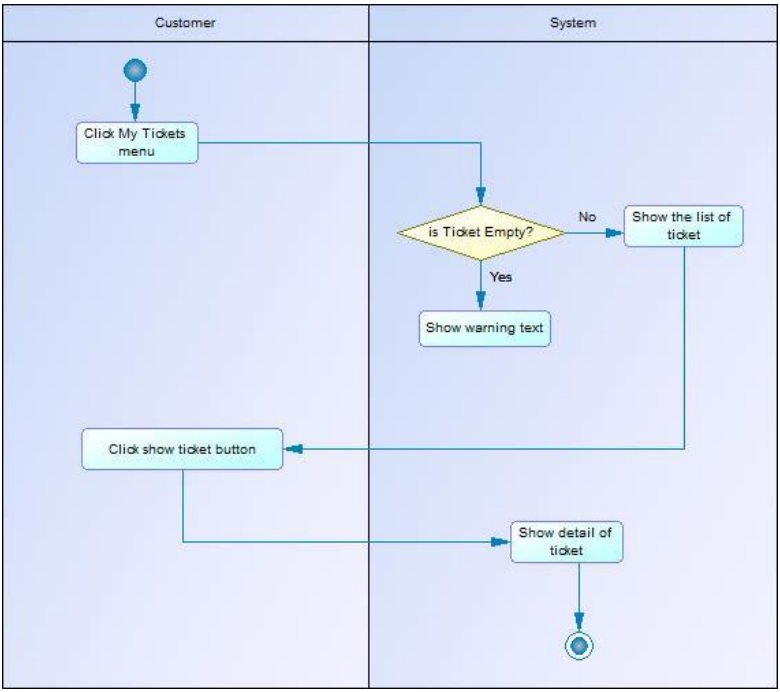


Figure 3.11 View Tickets Activity Diagram

3.1.4.12 Manage User Profile Use Case

In this use case, User can update their user profile. Detail of the use case is shown on Table 3.16 and the activity diagram is shown on Figure 3.14.

Table 3.14 Detail of Manage User Profile Use Case

Components	Description
Name	Manage user profile
Code	UC-012
Description	This use case is used by user to manage their profile
Type	Functional

Components	Description
Actor	Customer
Initial Condition	User's Profile has not been changed
End Condition	User's Profile has successfully updated
Normal Flow	1. Actor click My Profile menu 2. System show existing the actor's profile 3. Actor click update button 4. System show the update profile form 5. Actor click save button 6. System save the updated data
Alternate Flow	-

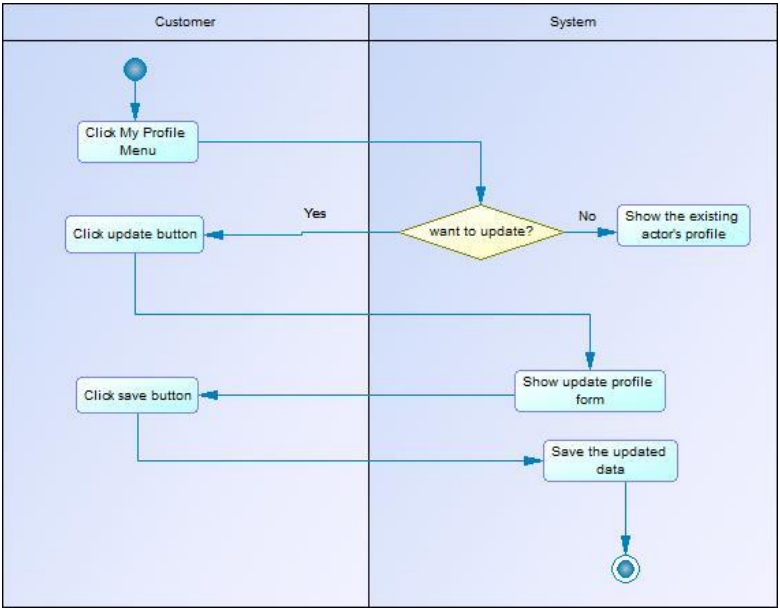


Figure 3.12 Manage User Profile Activity Diagram

3.2 Design

In the design section will be explain about the system architecture used, class diagram design, database design, and interface design.

3.2.1 Architectural Design and Design Pattern

The system architecture used in this Final Project is REST API. Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. An API is an application programming interface. It is a set of rules that allow programs to talk to each other. The developer creates the API on the server and allows the client to talk to it.

REST determines how the API looks like It is a set of rules that developers follow when they create their API. In this project, Laravel framework is used to manage the REST API (Backend) and Vue JS is used for User Interface (Frontend). The user interface is a layer that is directly related to the user.

The controller is the link between the interface and the service layer of the application. The service layer provides data processing from the repository layer. Then, the controller gets the data returned. After that, the controller will display in the user interface. The illustration in Figure 3.29 shows an REST API architecture diagram.

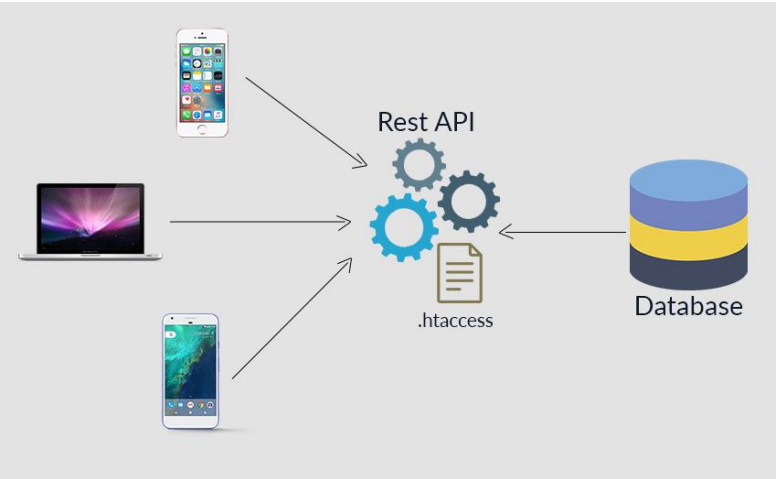


Figure 3.13 REST API Architecture

3.2.2 Class Diagram Design

In the Appendix chapter, the architecture model has classes in the form of controller, service, and repository. The use of system architecture like the picture above is used to make maintenance easier and easier to implement.

The controller class depends on the service class that is the place of data processing and the service class sends data requests to the database through the repository class. The repository class will send requests to the database and send data back from the database to the service class.

.

3.2.3 Database Design

Analysis of database design is needed to make an information system. MySQL was chosen as a database application because it can hold data on a large scale and free.

The database design is displayed in the form of Conceptual Data Model (CDM) and Physical Data Model (PDM). For a more detailed explanation, PDM will be explained in the Appendix chapter.

3.2.4 User Interface Design

This section will discuss about the design of interface for users based on use cases that have been designed.

3.2.4.1 Landing Page

Landing page is page where website visitors land when they first reach the website. A landing page is also an important term and concept for inbound marketing and lead generation. This page contains some important information of the information system. The design of user interface is shown on Figure 3.30. and Figure 3.31.

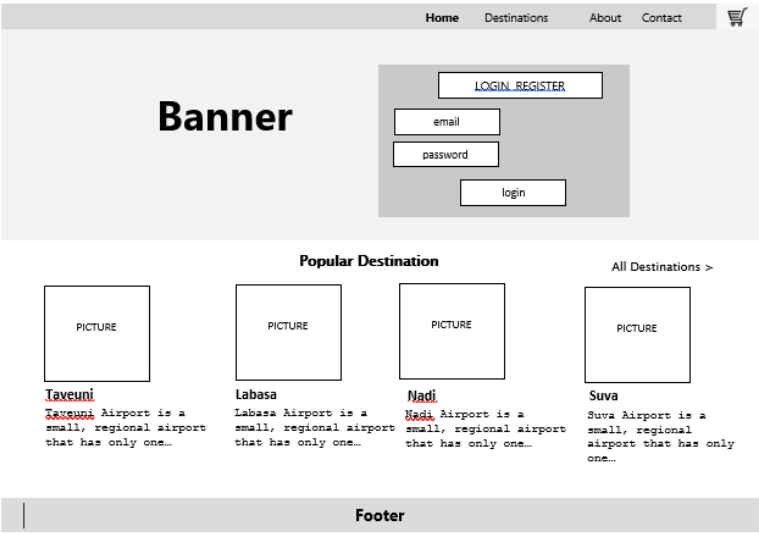


Figure 3.14 Desktop Landing Page User Interface Design

Table 3.15 Explanation of Landing Page User Interface

No	Attribute Name	Type	Function	Input/ Output
1	*banner*	*Banner*	Banner is either a graphic image that announces the name or identity of a site	*Image*
2	*loginTab*	*Tab*	Action Tab to move to the login form	*ButtonClick*
3	*registerTab*	*Button*	Action Tab to move to the register form	*ButtonClick*
4	*emailInput*	*Button*	Input text to fill the user's email	*String*
5	*passwordInput*	*Text*	Input text to fill the user's password	*String*
6	*loginButton*	*Button*	Action Button to Login into System	*ButtonClick*

3.2.4.2 Dashboard Page

Dashboard page is used to display the most important and useful information in the apps. Only admin can see this page. This page contains the information of total users registered, latest orders, total revenues of orders. The design of user interface is shown on Figure 3.30. and Figure 3.31.

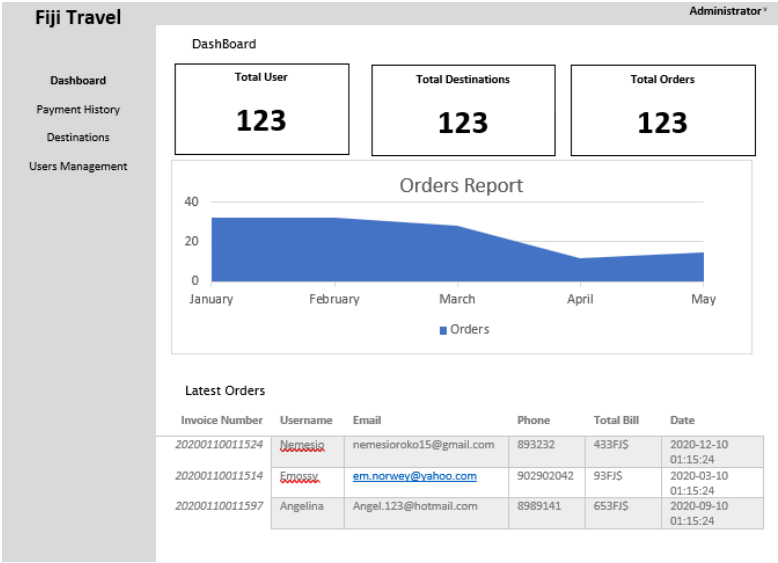


Figure 3.15 Dashboard User Interface Design

Table 3.16 Explanation of Dashboard User Interface

No	Attribute Name	Type	Function	Input/ Output
1	*totalUserInformation*	*Text*	Show the total users registered in system	*String*
2	*totalDestinationInformation*	*Text*	Show the total destinations in system	*String*
3	*totalOrdersInformation*	*Text*	Show the total orders in system	*String*
4	*orderReportGraph*	*Graph*	Show the Order Report graph	*Graph*
5	*listLatestOrders*	*List*	Show the list of latest order in system	*List*

3.2.4.3 Destinations Management Page

This page is used to manage the destinations data of the apps. On this page there is a list of destinations that has been made. Users can also create, search and update the data. The design of user interface is shown on Figure 3.30. and Figure 3.31.

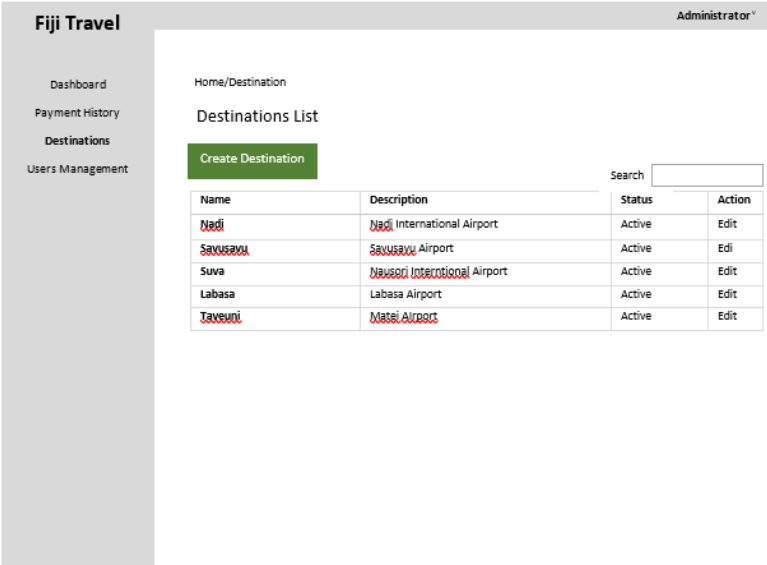


Figure 3.16 Destination List Page User Interface Design

Table 3.17 Explanation of Destination List Page User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*crateDestination Button*	*Button*	Action Button to create new destination	*ButtonClick*
2	*editButton*	*Button*	Action Button to Edit the existing travel destination	*ButtonClick*

No	Attribute Name	Type	Function	Input/ Output
5	*listDestinationsDatatable*	*Table*	Table to show the list of destinations	*Table*

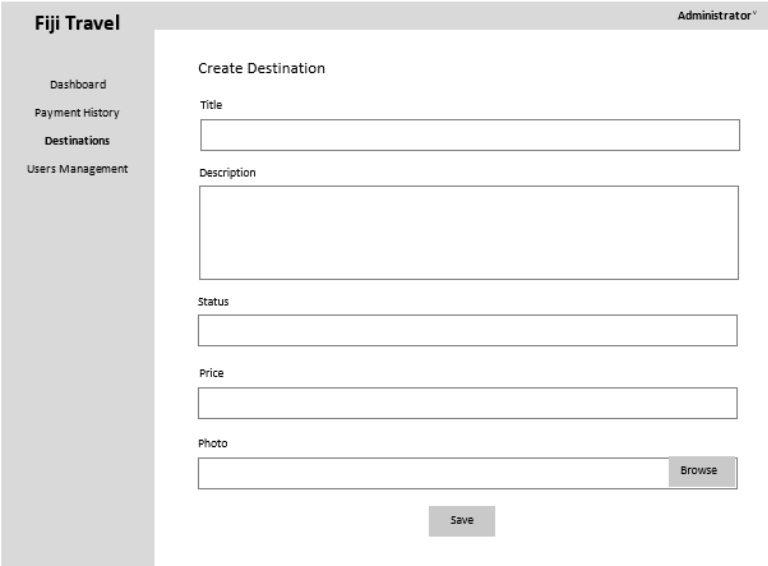


Figure 3.17 Create Destination Form User Interface Design

Table 3.18 Explanation of Create Destination Form User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*Title*	*Text*	Input the title / name of the destination	*String*
2	*Description*	*Text*	Input the description of destination	*String*

No	Attribute Name	Type	Function	Input/ Output
3	*Status*	*selectForm*	Input the status of the destination	*String*
4	*Price*	*Number*	Input the price of destination	*Number*
5	*Photo*	*Image*	Input the image of the destination	*Image*

3.2.4.4 Users Management Page

This page is used to manage the existing users of the apps. On this page there is a list of registered users. Users can also create, search and update the data. The design of user interface is shown on Figure 3.30. and Figure 3.31.

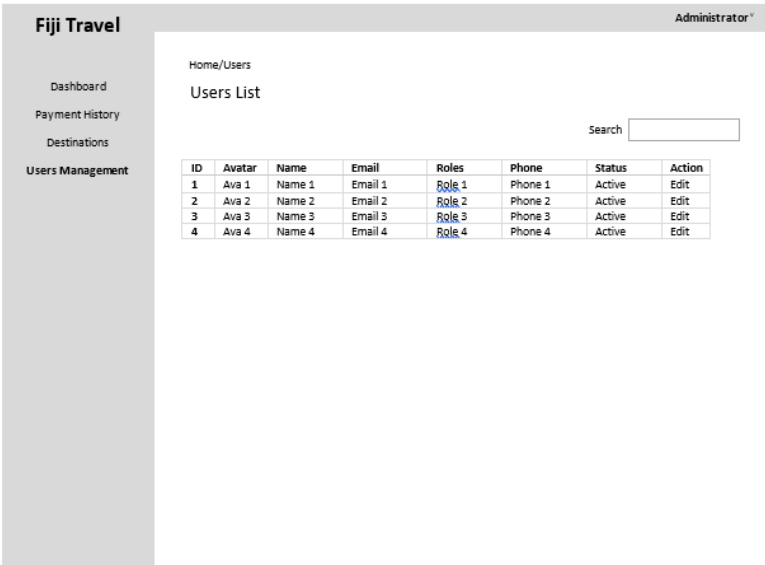


Figure 3.18 User List User Interface Design

Table 3.19 Explanation of User List User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*editButton*	*Button*	Action Button to edit the user's data	*Button*
2	*lisUsersDatatable*	*Table*	Table to show the list of all users	*Table*

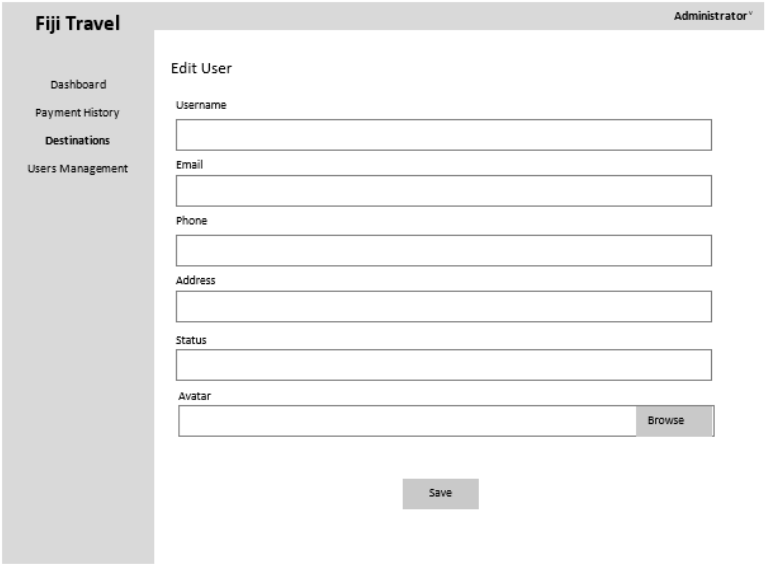


Figure 3.19 Edit User Form Interface Design

Table 3.20 Explanation of User List User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*Username*	*Text*	Input the username of the user	*String*
2	*Email*	*Text*	Input the email of the user	*String*

No	Attribute Name	Type	Function	Input/ Output
3	*Status*	*selectForm*	Input the status of the user	*String*
4	*Phone*	*Number*	Input the user's number phone	*Number*
5	*Address*	*Text*	Input the address of the user	*String*
6	*Avatar*	*Image*	Input the image of the user	*Image*

3.2.4.5 Payment History Page

This page is used to show the histories of all transactions of the apps. On this page there is a list of orders that has been made. Users can view the detail of data. The design of user interface is shown on Figure 3.30. and Figure 3.31.

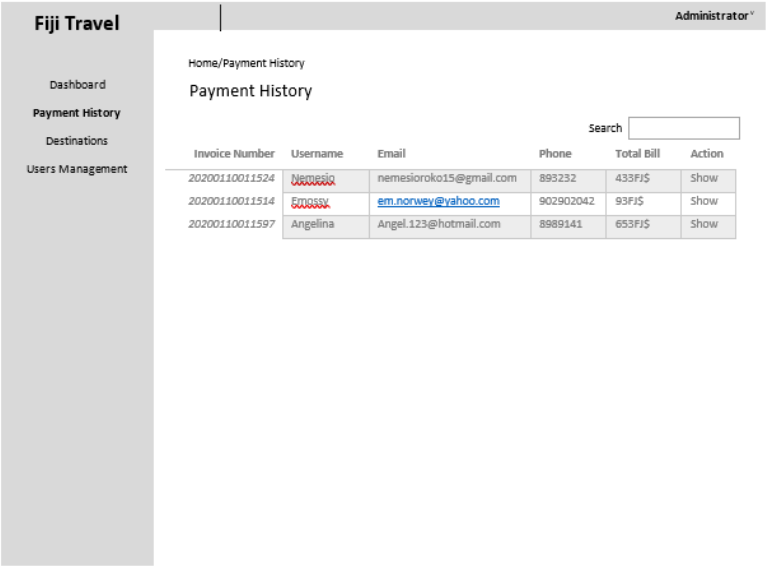


Figure 3.20 Payment History Page User Interface Design

Table 3.21 Explanation of Payment History Page User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*showButton*	*Button*	Action button to show the detail of invoice	*Button*
2	*paymentListData table*	*Table*	Table to show the list of the payment	*Table*

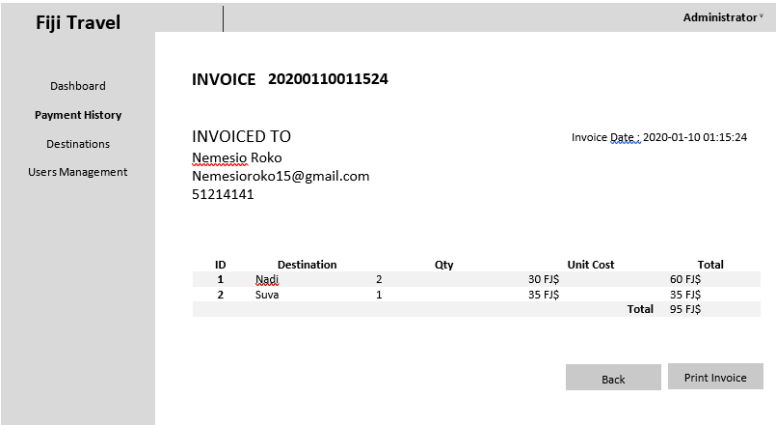


Figure 3.21 Invoice Page User Interface Design

3.2.4.6 Search Destination Page

This page is used for customers to search the destinations that they want. In this page, user can show the detail of the destination and can add the destination into cart. The design of user interface is shown on Figure 3.30. and Figure 3.31.

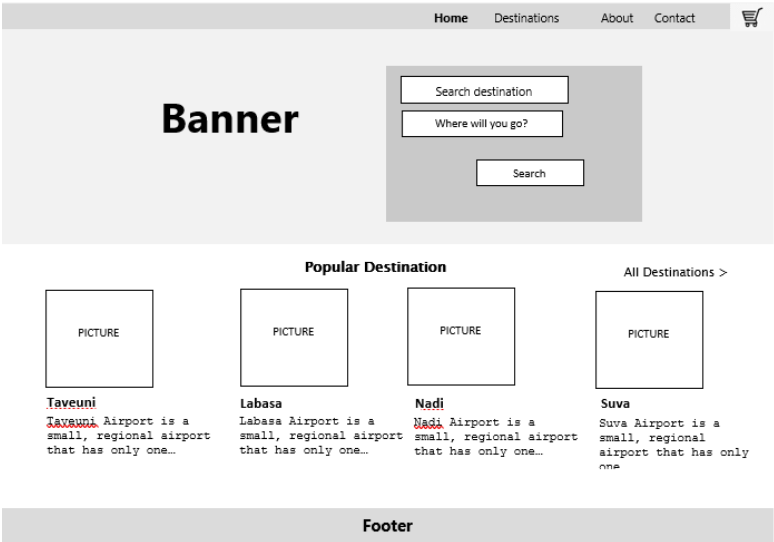


Figure 3.22 Search Destination Page User Interface Design

Table 3.22 Explanation of Search Destination Page User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*banner*	*Banner*	Banner is either a graphic image that announces the name or identity of a site	*Image*
2	*keywordInput*	*Text*	Input text to search the destination	*String*
3	*searchButton*	*Button*	Action Button to search the destination	*ButtonClick*

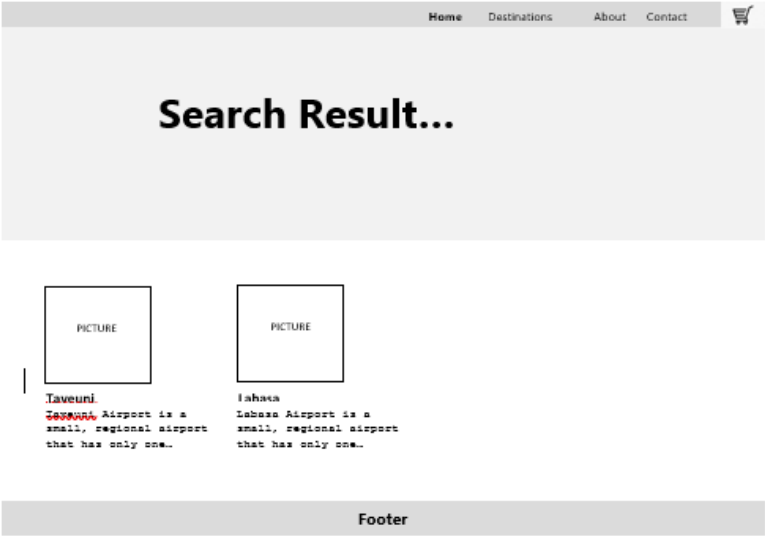


Figure 3.23 Search Result User Interface Design

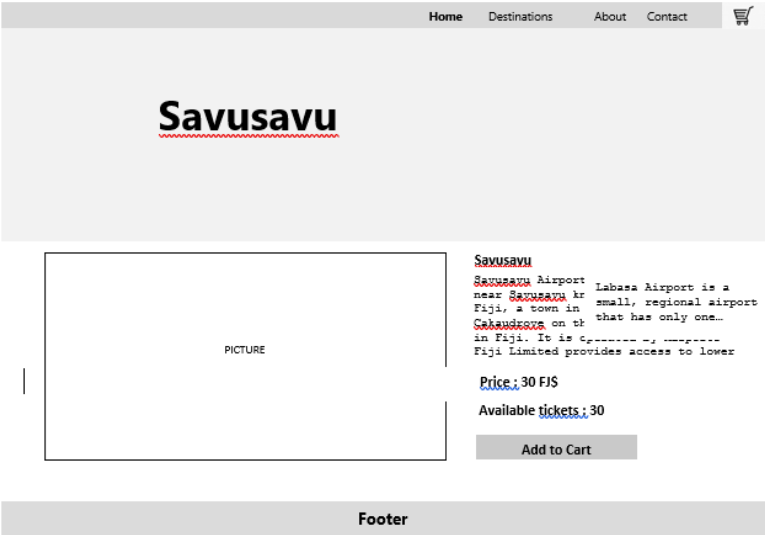


Figure 3-2 Detail Destination User Interface Design

Table 3.23 Explanation of Detail Destination User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*Picture*	*Image*	Image of the destination	*Image*
2	*Description*	*Text*	Description of Destination	*String*
3	*addToCartButton*	*Button*	Action Button to add the destination into the cart	*Button*

3.2.4.7 Customer's Cart Page

This page is used for customers to show their carts before they make a payment. In this page, user also can clear the cart. The design of user interface is shown on Figure 3.30. and Figure 3.31.

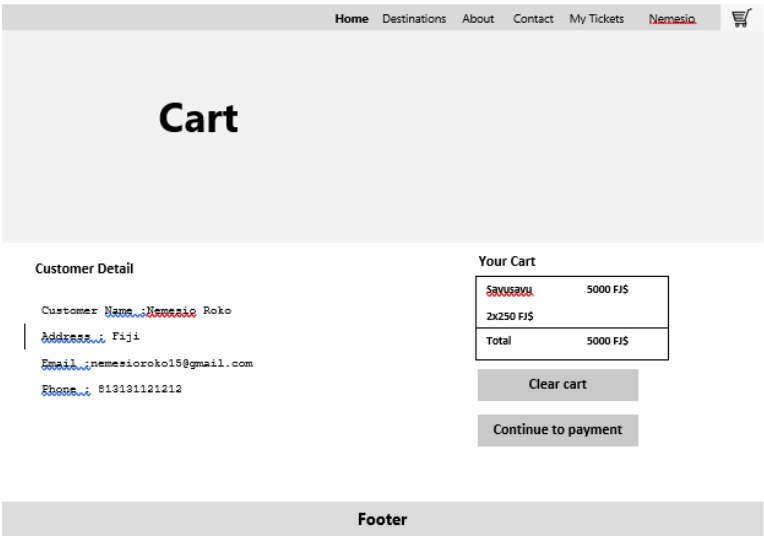


Figure 3.24 Customer's Cart User Interface Design

Table 3.24 Explanation of Customer's Cart User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*customerDetail*	*Text*	Detail information of the customer	*String*
2	*cartDetail*	*Text*	Detail of the customer's cart	*String*
3	*clearCartButton*	*Button*	Action button to clear the cart	*Button*
4	*continueToPaymentButton*	*Button*	Action button to continue t the payment	*Button*

3.2.4.8 Payment Page

This page contains the information of the payment after the customer complete to check out the cart. After complete the payment, customer will get the tickets. The design of user interface is shown on Figure 3.30. and Figure 3.31.

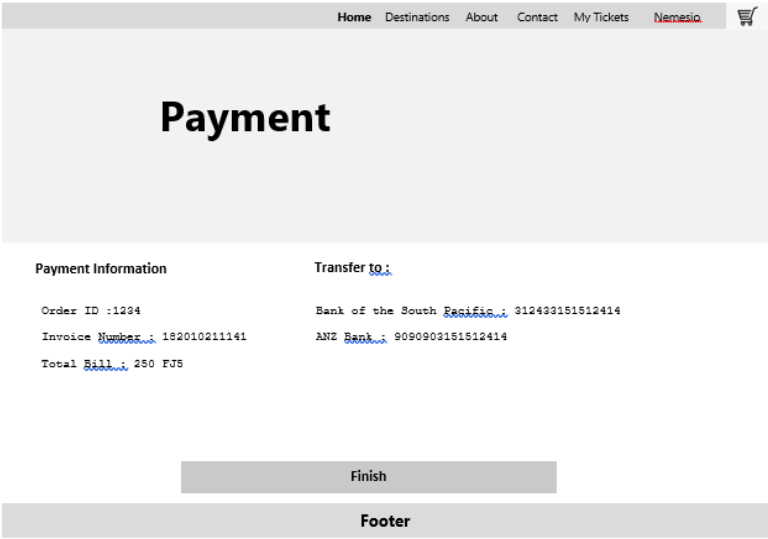


Figure 3.25 Payment Page User Interface Design

Table 3.25 Explanation of Payment Pag User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*paymentInformation*	*Text*	Detail information of payment	*String*
2	*finishButton*	*Button*	Action button to finish the payment	*Button*

3.2.4.9 Customer's Ticket Page

This page contains the list of the customer's tickets. Customer also can show the detail of the tickets. The design of user interface is shown on Figure 3.30. and Figure 3.31.

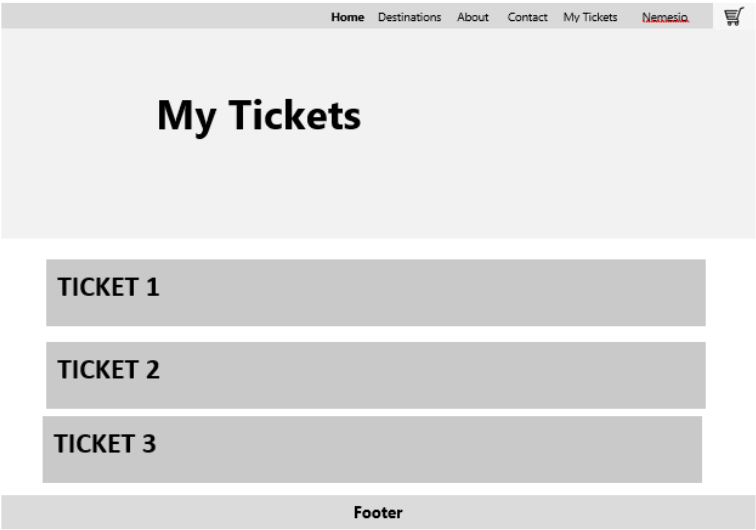


Figure 3.26 Customer's Ticket Page User Interface Design

Table 3.26 Customer's Ticket Page User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*listTickets*	*List*	List of cutomer's ticket	*list*

3.2.4.10 Customer's Profile Page

This page contains the information of the user's profile. Users can update their data on this page. The design of user interface is shown on Figure 3.30. and Figure 3.31.

Figure 3.27 Customer's Profile Page User Interface Design

Table 3.27 Explanation of Customer's Profile Page User Interface Design

No	Attribute Name	Type	Function	Input/ Output
1	*Username*	*Text*	Input the username of the user	*String*
2	*Email*	*Text*	Input the email of the user	*String*
3	*Status*	*selectForm*	Input the status of the user	*String*
4	*Phone*	*Number*	Input the user's number phone	*Number*
5	*Address*	*Text*	Input the address of the user	*String*
6	*Avatar*	*Image*	Input the image of the user	*Image*
7	*updateButton*	*Button*	Action button to update the profile	*Button*

3.2.5 Business Process

This section will show business processes for the Information System. The business process module is started by an admin posts the destination on the apps. Then, the customer creates an account and login to order the destination on the apps. After order destination and complete the payment, customer gets the ticket. An admin can views the order of all customers and can manage users accounts too. The business process flow is shown in Figure 3.53.

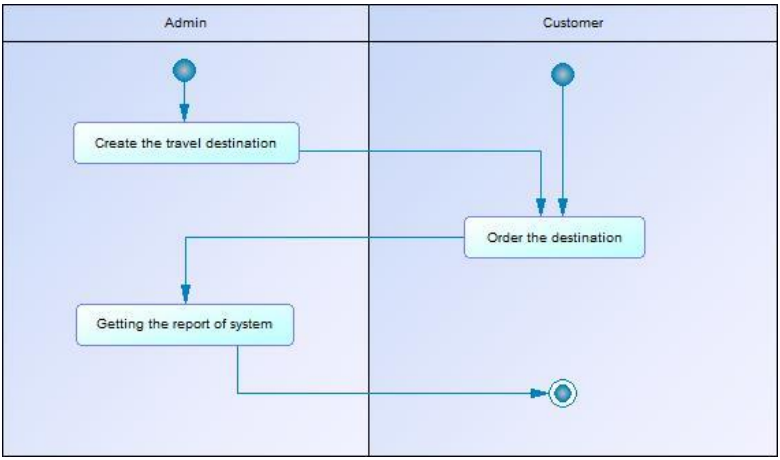


Figure 3.28 Business Process

CHAPTER IV
IMPLEMENTATION

This chapter will discuss the implementation of the system based on business process that has been explained before.

4.1 Implementation Environment

The system implementation environment that used to develop the final project has specifications as shown on Table 4.1.

Table 4.1 Implementation System Environment

Device	Specifications
Hardware	Prosesor: Intel® Core™ i5-7400 CPU @ 3.00GHz (4 CPUs) , ~3.0GHz Memory: 8192 MB
Software	Operating System: Microsoft Windows 10 Pro 64-bit Framework: Laravel, Vue Diagram Designer: Sybase Power Designer 16 Database: MySQL

4.2 Implementation of User Inteface

Implementation of the user interface is using a vue file and blade file for each page. The following will be discussed regarding the implementation of the system that has been realized.

4.2.1 Desktop User Interface

4.2.1.1 Landing Page

Landing page is page where website visitors land when they first reach the website. A landing page is also an important term and concept for inbound marketing and lead generation. This page contains some important information of the information system. The design implementation of user interface is shown on Figure 4.1.

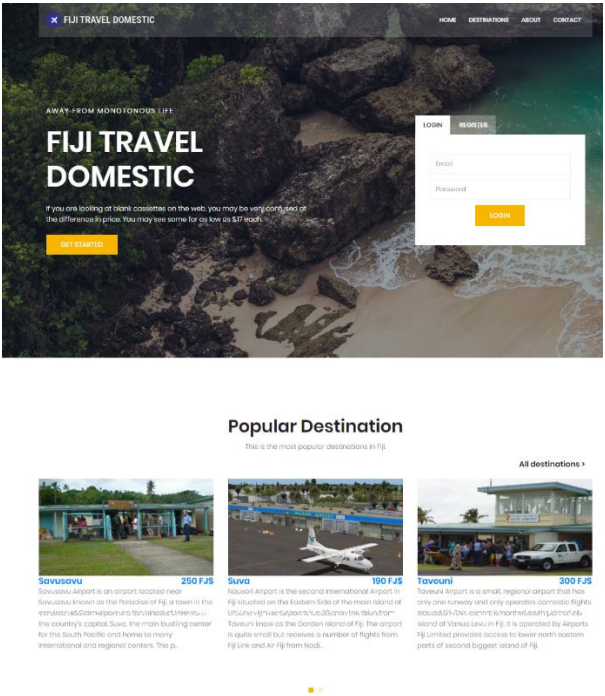


Figure 4-1 Landing Page User Interface

4.2.1.2 Dashboard Page

Dashboard page is used to display the most important and useful information in the apps. Only admin can see this page. This page contains the information of total users registered, latest orders, total revenues of orders. The design implementation of user interface is shown on Figure 4.2.

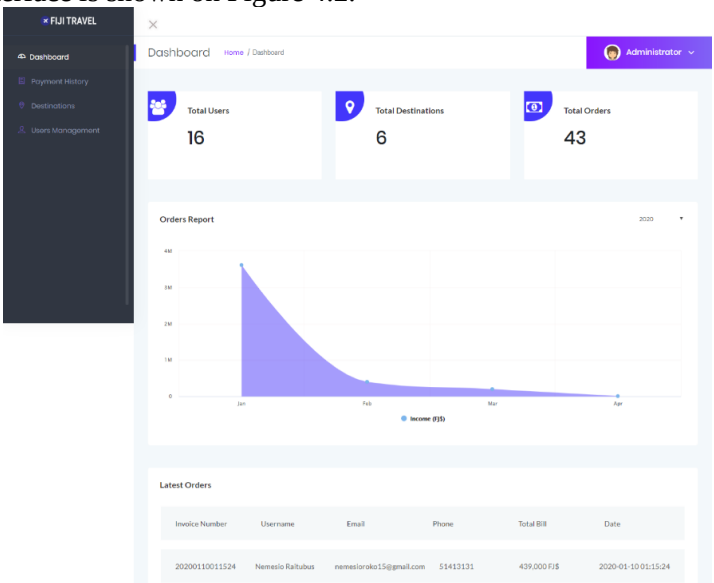


Figure 4-2 Dashboard User Interface

4.2.1.3 Payment History Page

This page is used to show the histories of all transactions of the apps. On this page there is a list of orders that has been made. Users can view the detail of data. The design implementation of user interface is shown on Figure 4.3 and 4.4

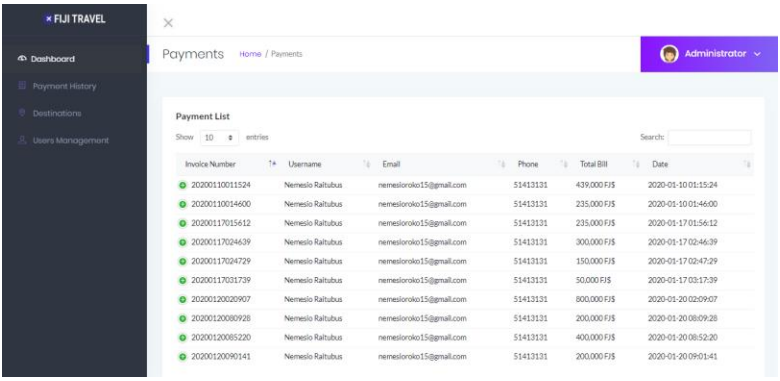


Figure 4-3 List of Payment History User Interface

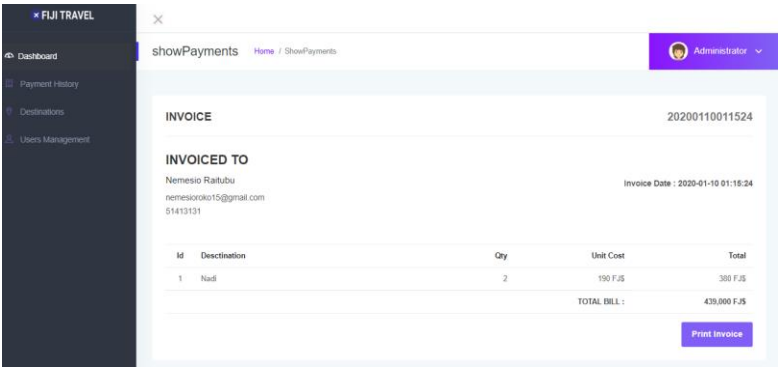


Figure 4-4 Invoice User Interface

4.2.1.4 Manage Destinations Page

This page is used to manage the destinations data of the apps. On this page there is a list of destinations that has been made. Users can also create, search and update the data. The design implementation of user interface is shown on Figure 4.5, Figure 4.6, and Figure 4.7.

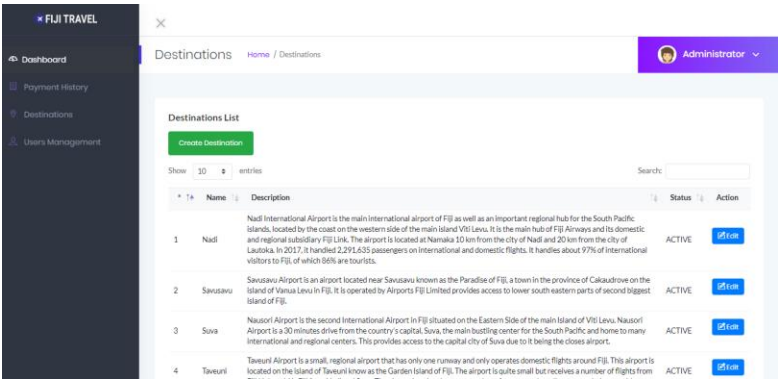


Figure 4-5 List of Destination User Interface

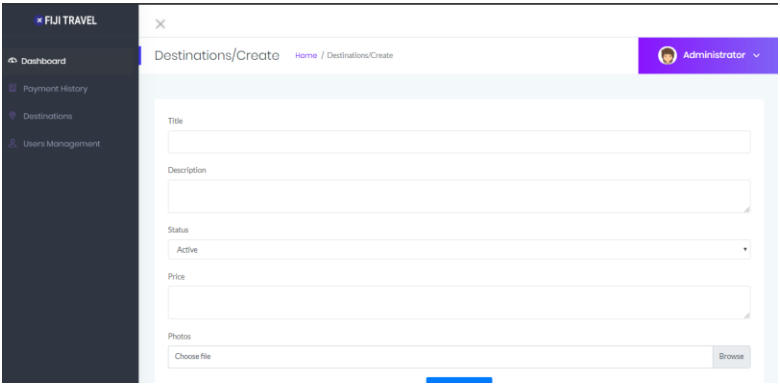


Figure 4-6 Create Destination User Interface

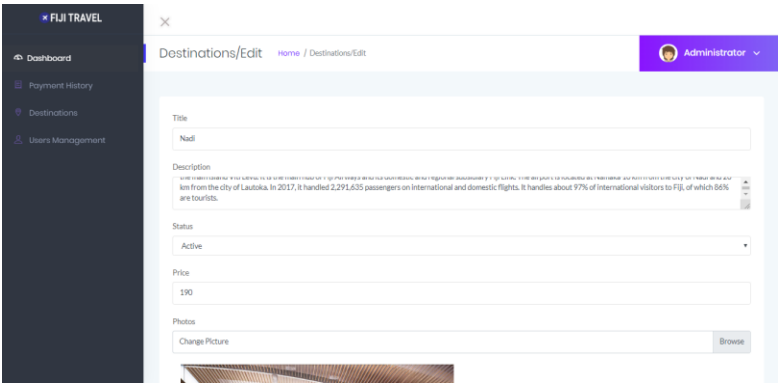


Figure 4-7 Edit Destination User Interface

4.2.1.2 Manage Users Page

This page is used to manage the existing users of the apps. On this page there is a list of registered users. Users can also create, search and update the data. The design implementation of user interface is shown on Figure 4.8 and 4.9.

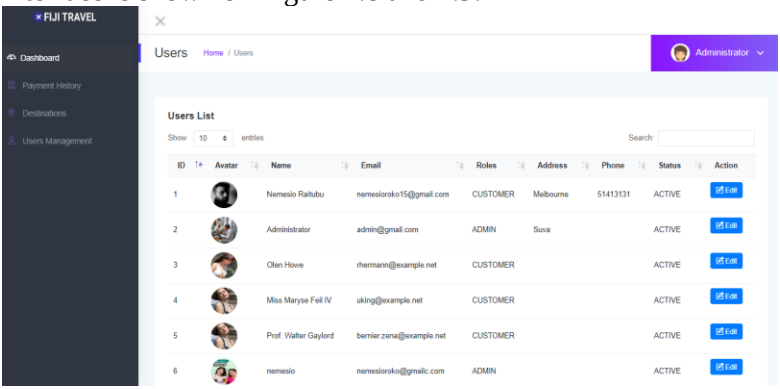


Figure 4-8 List Users User Interface

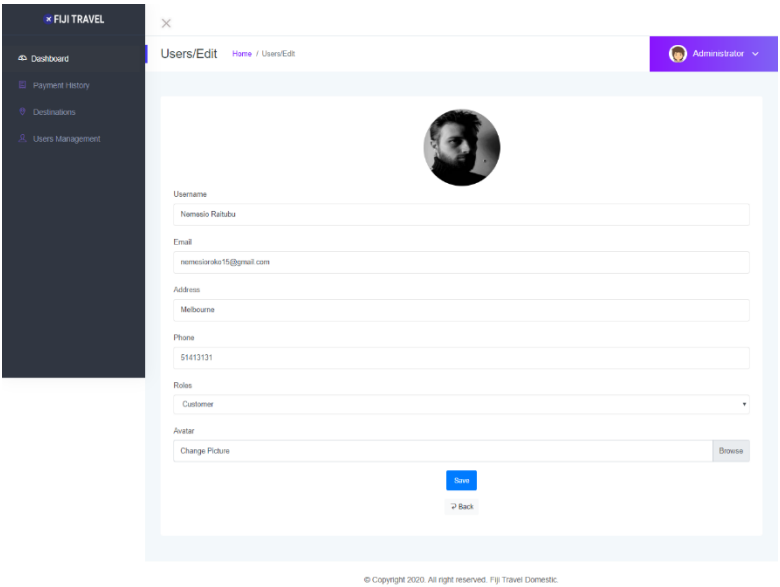


Figure 4-9 Edit User User Interface

4.2.1.2 Search Destination

This page is used for customers to search the destinations that they want. In this page, user can show the detail of the destination and can add the destination into cart. The design implementation of user interface is shown on Figure 4.10, Figure 4.11 and Figure 4.12

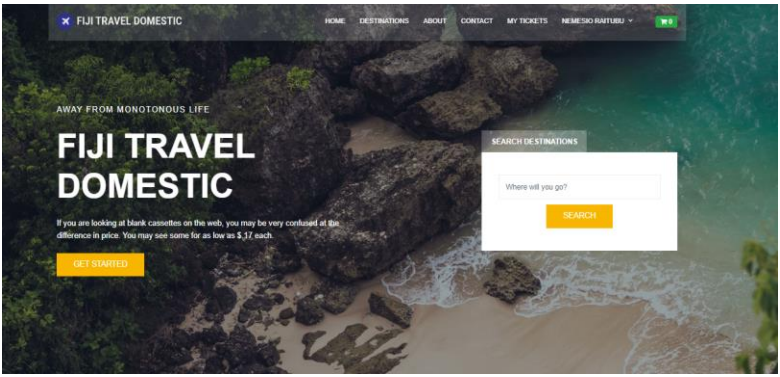


Figure 4-10 Search Destination User Interface

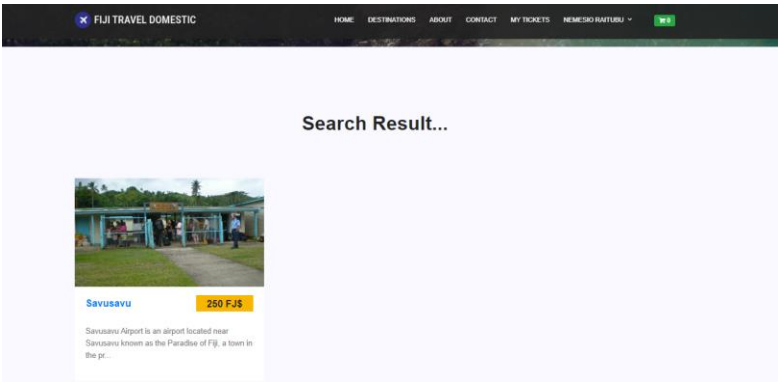


Figure 4-11 Search Result User Interface

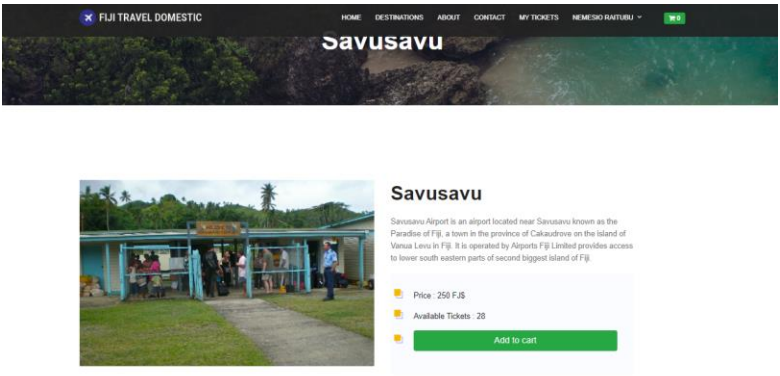


Figure 4-12 Detail of Destination User Interface

4.2.1.2 Customer's Cart Page

This page is used for customers to show their carts before they make a payment. In this page, user also can clear the cart.The design implementation of user interface is shown on Figure 4.13.

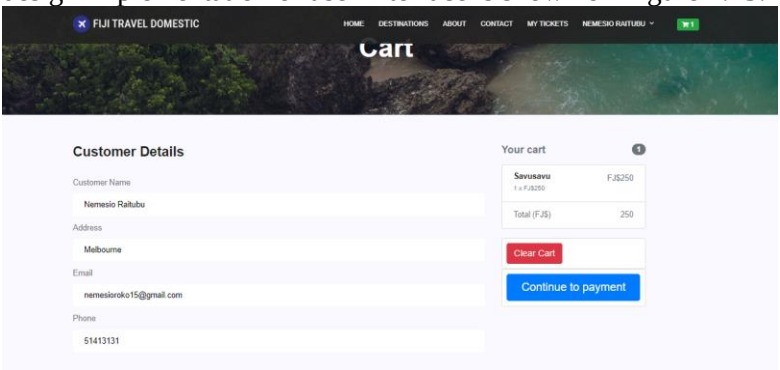


Figure 4-13 Customer's Cart User Interface

4.2.1.2 Payment Page

This page contains the information of the payment after the customer complete to check out the cart. After complete the payment, customer will get the tickets. The design implementation of user interface is shown on Figure 4.14.

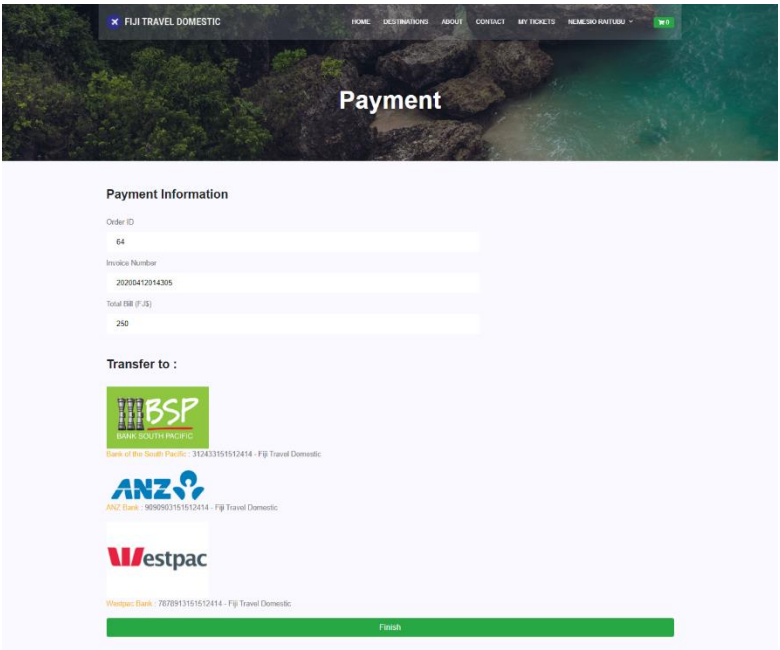


Figure 4-14 Payment Page User Interface

4.2.1.2 Customer's Ticket Page

This page contains the list of the customer's tickets. Customer also can show the detail of the tickets. The design implementation of user interface is shown on Figure 4.15 and Figure 4.16.

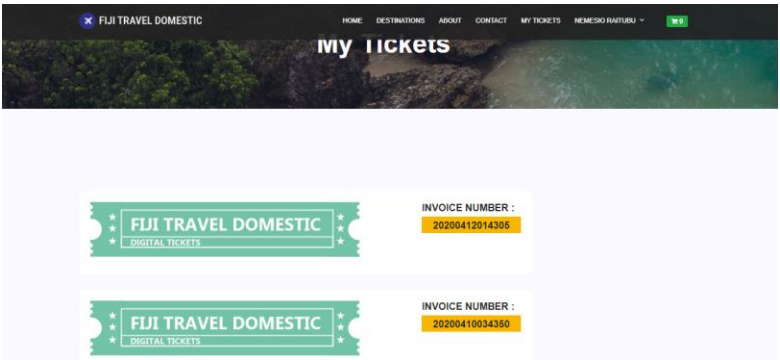


Figure 4-15 List Customer's Ticket User Interface

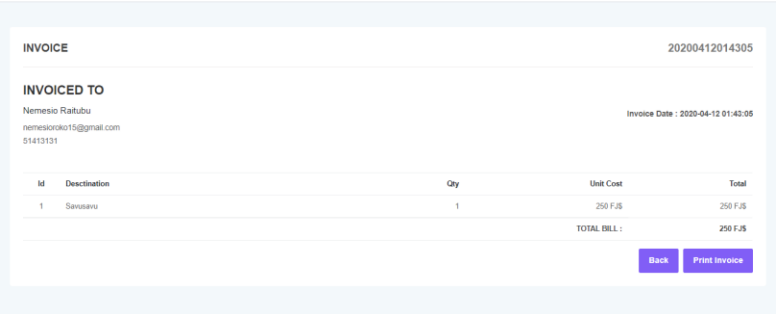


Figure 4-16 Detail of Ticket User Interface

4.2.1.2 Customer's Profile Page

This page contains the information of the user's profile. Users can update their data on this page. The design implementation of user interface is shown on Figure 4.17.

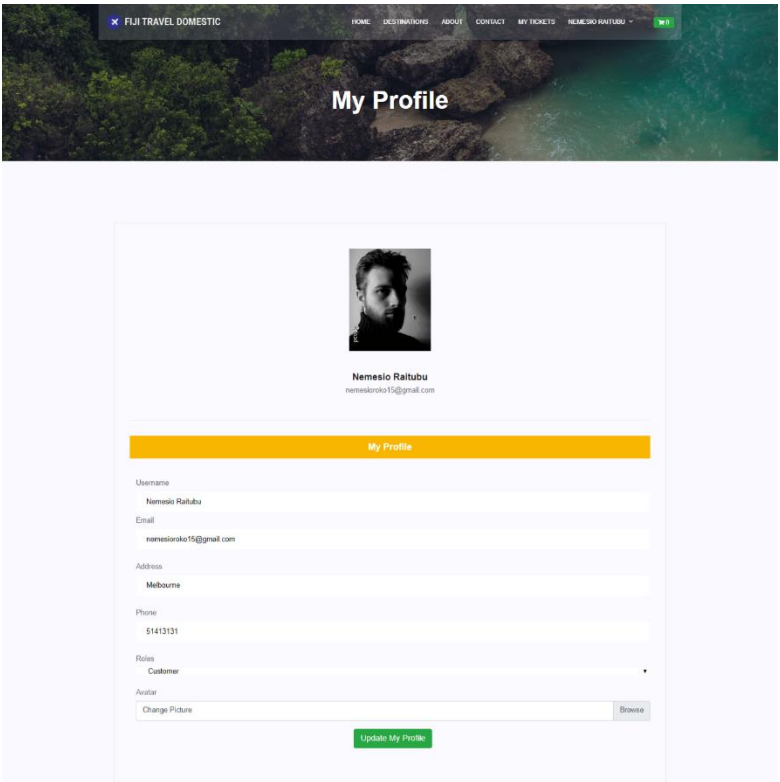


Figure 4-17 Customer's Profile User Interface

4.2.2 Mobile User Interface

4.2.1.1 Homepage

Homepage is page where visitors land when they first reach the apps. This page contains some important information of the apps. The design implementation of user interface is shown on Figure 4.18.

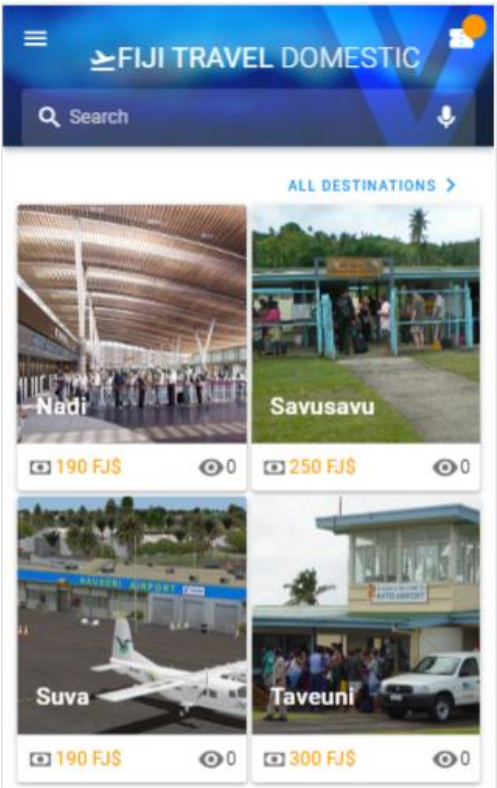


Figure 4-18 Homepage User Interface

4.2.1.2 Register Page

This page is used for a guest to register account into system before they can make the destination order. The design implementation of user interface is shown on Figure 4.19.

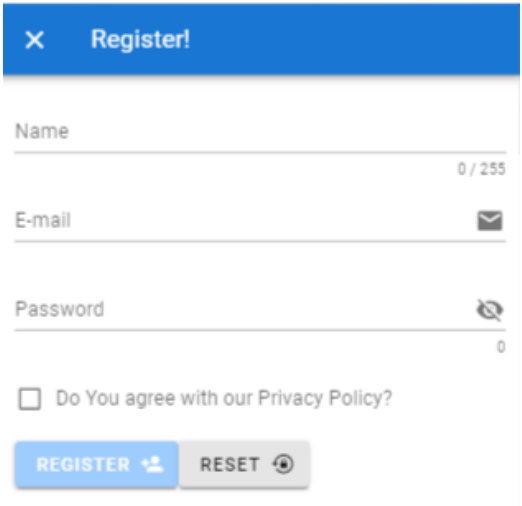


Figure 4-19 Register User Interface

4.2.1.3 Login Page

This page is used for a registered customer to login into their account into system. The design implementation of user interface is shown on Figure 4.20.

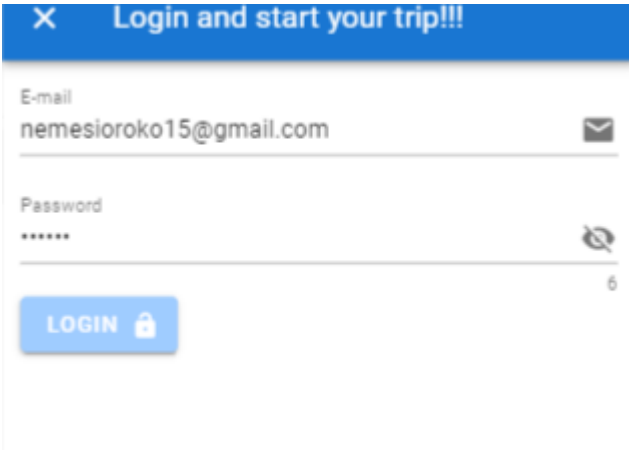


Figure 4-20 Login User Interface

4.2.1.4 Search Destination Page

This page is used for customers to search the destinations that they want. In this page, user can show the detail of the destination and can add the destination into cart .The design implementation of user interface is shown on Figure 4.21.

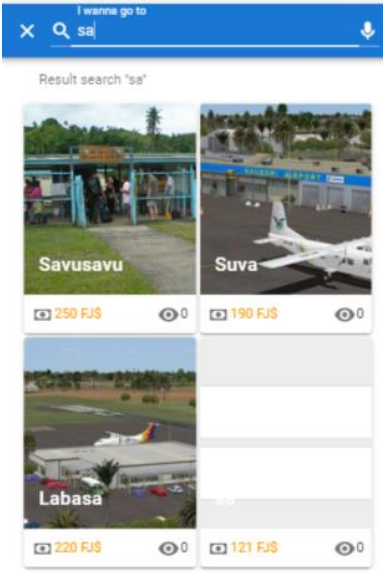


Figure 4-21 Search Destination User Interface

4.2.1.5 Detail Destination Page

This page is used for customers to show the detail of destination that In this page, user can show the detail of the destination and can add the destination into cart. The design implementation of user interface is shown on Figure 4.22.

Figure 4-22 Detail Destination User Interface

4.2.1.6 Customer's Cart Page

his page is used for customers to show their carts before they make a payment. In this page, user also can clear the cart. The design implementation of user interface is shown on Figure 4.23.

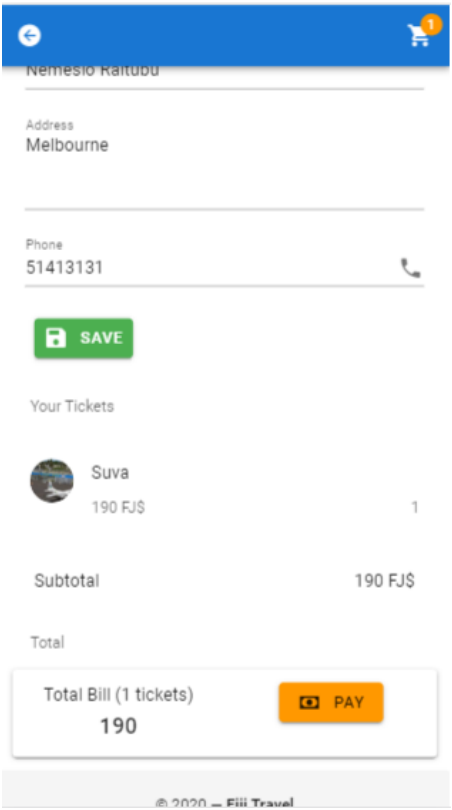


Figure 4-23 Customer's Cart User Interface

4.2.1.7 Payment Page

This page contains the information of the payment after the customer complete to check out the cart. After complete the payment, customer will get the tickets. The design implementation of user interface is shown on Figure 4.24.

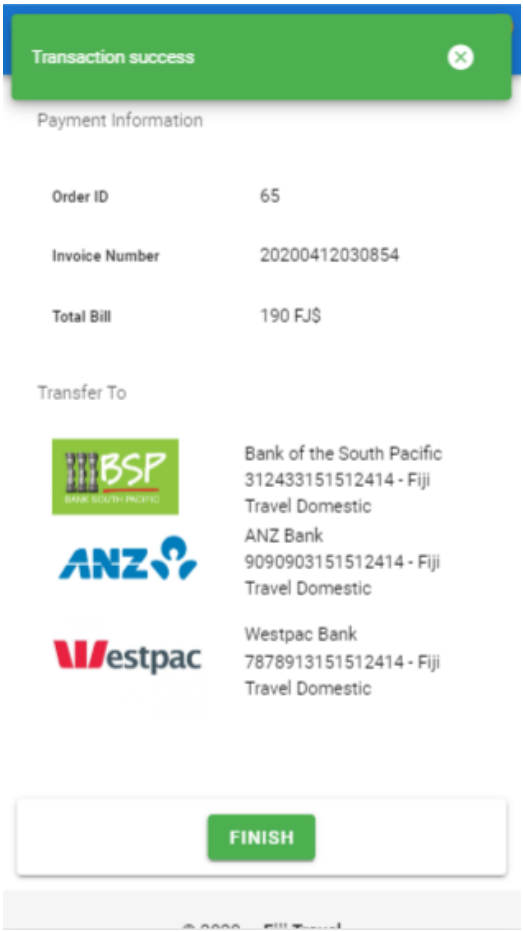


Figure 4-24 Payment Page User Interface

4.2.1.8 Customer's Ticket Page

his page contains the list of the customer's tickets. Customer also can show the detail of the tickets. The design implementation of user interface is shown on Figure 4.25.

Figure 4-25 Customer's Ticket User Interface

4.3 Implementation of REST API Architecture on MVC (Model-View-Controller Pattern) and MVVM (Model-View-ViewModel Pattern)

REST APIs are the defined interfaces through which interactions happen between an enterprise and applications that use its assets, which also is a Service Level Agreement (SLA) to specify the functional provider and expose the service path or URL for its API users. An API approach is an architectural approach that revolves around providing a program interface to a set of services to different applications serving different types of consumers.

When used in the context of web development, an API is typically defined as a set of specifications, such as Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages that represented on JavaScript Object Notation (JSON) format. In this thesis, the API is built with Laravel. Below the list of API is shown on Figure 4.26.

```
C:\laragon\www\travel-backend
λ php artisan route:list
+--------+----------+--------------------------------+------+-----------------------------------------------+-----
--------+
| Domain | Method   | URI                            | Name | Action                                        | Midd
leware   |
+--------+----------+--------------------------------+------+-----------------------------------------------+-----
--------+
|        | GET|HEAD | v1/categories                  |      | App\Http\Controllers\CategoryController@index     | api
|        |
|        | GET|HEAD | v1/categories/random/{count}   |      | App\Http\Controllers\CategoryController@random    | api
|        |
|        | GET|HEAD | v1/categories/slug/{slug}      |      | App\Http\Controllers\CategoryController@slug      | api
|        |
|        | GET|HEAD | v1/destinations                |      | App\Http\Controllers\DestinationController@index  | api
|        |
|        | POST     | v1/destinations/cart           |      | App\Http\Controllers\DestinationController@cart   | api
|        |
|        | GET|HEAD | v1/destinations/search/{keyword} |      | App\Http\Controllers\DestinationController@search | api
|        |
|        | GET|HEAD | v1/destinations/slug/{slug}    |      | App\Http\Controllers\DestinationController@slug   | api
|        |
|        | POST     | v1/destinations/store          |      | App\Http\Controllers\DestinationController@store  | api
|        |
|        | GET|HEAD | v1/destinations/top/{count}    |      | App\Http\Controllers\DestinationController@top    | api
|        |
|        | POST     | v1/login                       |      | App\Http\Controllers\AuthController@login         | api
|        |
|        | POST     | v1/logout                      |      | App\Http\Controllers\AuthController@logout        | api,
auth:api |
|        | GET|HEAD | v1/my-order                    |      | App\Http\Controllers\ShopController@myOrder       | api,
auth:api |
|        | POST     | v1/payment                     |      | App\Http\Controllers\ShopController@payment       | api,
auth:api |
|        | POST     | v1/register                    |      | App\Http\Controllers\AuthController@register      | api
|        |
|        | POST     | v1/services                    |      | App\Http\Controllers\ShopController@services      | api,
auth:api |
|        | POST     | v1/shipping                    |      | App\Http\Controllers\ShopController@shipping      | api,
auth:api |
```

Figure 4-26 List of API

In the frontend layer, we use blade and vue format file to represent the User Interface. The user interface that using blade file is created using MVC pattern. Below is the structure of the project.

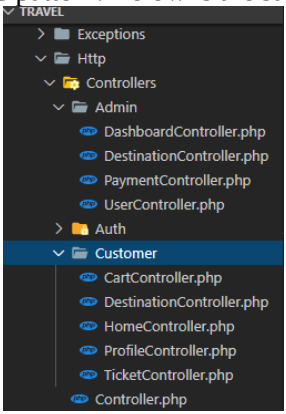


Figure 4-27 Projects Controller Directory Structure

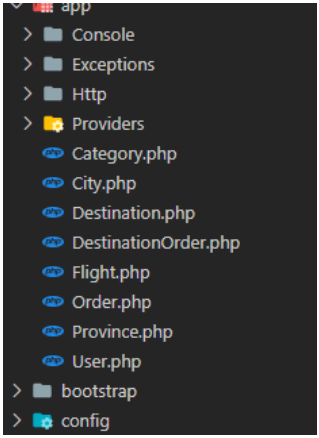


Figure 4-28 Project Models Directory Structure

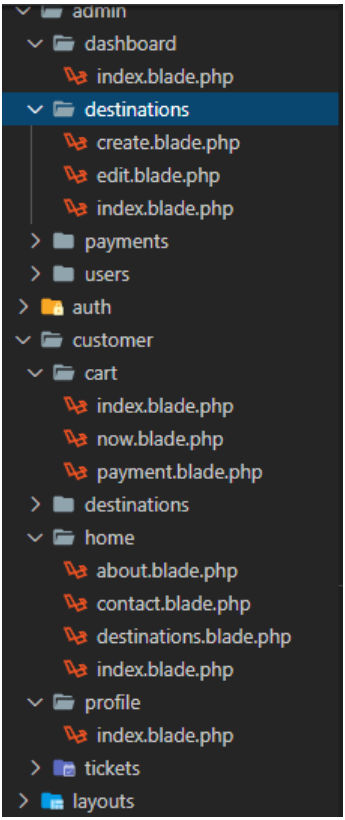


Figure 4-29 Project Views Directory Structure

This project is using api_url constructor to call the API from the backend. Below is the implementation

```
public function __construct()
{
    $this->middleware('auth');
    $this->api_url = 'travel-backend.local/v1/';
}
```

Figure 4-30 API URL Constructor

These are the implementation of GET method and POST Method in REST API.

```
public function getDestinations()
{
    $client = new Client();
    $url =  $this->api_url . 'destinations';
    $request = $client->get($url);
    $response = $request->getBody()->getContents();
    $destinations = json_decode($response, true);

    return Datatables::of($destinations)
        ->addColumn('action', function ($destinations) {
            return '<a href="destinations/edit/' . $destinations->slug . '"
        })
        ->removeColumn('views')
        ->make(true);
```

Figure 4-31 REST API GET Method

```
public function store(Request $request)
{
    $client = new Client();

    $response = $client->request('POST', $this->api_url . 'destinations/store', [
        'form_params' => [
            'title' => $request->title,
            'slug' => str_slug($request->title),
            'description' => $request->description,
            'status' => $request->status,
            'price' => $request->price,
            'cover' => $request->cover->getClientOriginalName(),
        ]
    ]);
    $file = $request->file('cover');
    $path = '../../travel-backend/public/images/destinations';
    $file->move($path,$file->getClientOriginalName());

    return redirect()->route('Destinations');
```

Figure 4-32 REST API POST Method

In this project, the mobile frontend is using Vue JS. In VueJS, the design pattern that used is using MVVM (Model View View-Model). MVVM stands for Model-View-View Model. The key difference is the existence of the VM which is a construct which provides linkage/interface between the Model and the View.

In MVVM, the Model simply represents the data access layer of the application. It holds the data/information which is to be

presented to the user for manipulation or interaction. The Model has no behavior or logic defined on it in any way apart from data validation. The Model also has no means to access a backend or 3rd Party API to generate or save data — it simply serves as a container to hold the information/data which the VM retrieves and uses.

The View is used to render the information contained in the Model to the user. In MVVM, the View doesn't know about the Model and vice-versa. The View knows about the VM and is therefore known as an 'active' View. All user action e.g. user input Is intercepted by the View and passed to the VM for processing. The View doesn't maintain any state rather it simply represents 'state' as defined by the VM.

The VM is the link between the Model and the View. All logic required for manipulating the data contained in the Model is defined on the VM and all logic which the View uses to handle user interaction or format the data from the Model are provided to it from the VM.

[This page has been intentionally left blank]

CHAPTER V
TESTING AND EVALUATION

This chapter discusses about testing and Evaluation of Fiji Travel Domestic Information System. The test that used in this chapter is functionality testing.

5.2 Testing Environment

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured. In this case, the testing environment is run on :

Table 5.1 Testing Environment

Specification	Description
CPU	Intel ® Core ™ i3-2120 CPU @ 3.30 GHz
RAM	12.0 GB
Operating System	Windows 10 Enterprise 64-bit

5.3 Trial Scenarios

This section will discuss about the testing process used. Testing is done by black box method to test each functionality that has been designed on the system. Black box method is a software testing method that checks the functionality of a software regardless of its internal structure.

In the trial process, each trial participant is asked to carry out a series of commands to the system which will henceforth be called a test case. This test case correlates with use cases and functional requirements that have been previously designed and explained in Chapter III.

5.3.1 Create Travel Destination Test Case

This test case is used to test whether the actor with the role of Admin can create the travel destination. Detail of the test cases are shown in Table 5.2.

Table 5.2 Detail of Create Travel Destination Test Case

Testing Scenario Name	The functionality of Create Travel Destination
Code	TC-001
Testing Purpose	Testing Create Travel Destination Function
Scenario 1	*Admin create new destination*
Initial Condition	Travel destination has not been added in the system.
Testing Procedure	1. Actor choose destinations menu 2. System show the list of destinations 3. Actor click Create Destination Button 4. System show the form to create new destination. 5. Actor fill the form to create new travel destination 6. System save the new data
Input	1. Title 2. Description 3. Status 4. Price 5. Photo
Expected Result	Travel destination has successfully added in the system.
Testing Result	Travel destination has successfully added in the system.
Status	Success

5.3.2 Update travel destination Test Case

This test case is used to test whether the actor with the role of Admin can create the travel destination. Detail of the test cases are shown in Table 5.3

Table 5.3 Detail of Update Travel Destination Test Case

Testing Scenario Name	The functionality of Update Travel Destination
Code	TC-002
Testing Purpose	Testing Update Travel Destination Function
Scenario 1	*Admin update existing destination*
Initial Condition	Travel destination has not been changed.
Testing Procedure	1. Actor choose destinations menu 2. System show the list of destinations 3. Actor click edit button 4. System show the form to edit destination. 5. Actor fill the form to edit travel destination 6. System save the updated data
Input	1. Title 2. Description 3. Status 4. Price 5. Photo
Expected Result	Travel destination has successfully updated
Testing Result	Travel destination has successfully updated
Status	Success

5.3.3 View travel destination Test Case

This test case is used to test whether the actor with the role of Admin can view the travel destination. Detail of the test cases are shown in Table 5.4

Table 5.4 Detail of View travel destination Test Case

Testing Scenario Name	The functionality of View Travel Destination
Code	TC-003
Testing Purpose	Testing View Travel Destination Function
Scenario 1	*Admin view the destination*
Initial Condition	User choose the destination menu
Testing Procedure	Actor choose destinations menu System show the list of destinations Actor click show button System show the detail of destination.
Input	-
Expected Result	User successfully see the detail of travel destination
Testing Result	User successfully see the detail of travel destination
Status	Success

5.3.4 Manage Users Test Case

This test case is used to test manage users function. Detail of the test cases are shown in Table 5.5

Table 5.5 Detail of Manage Users Test Case

Testing Scenario Name	The functionality of manage users
Code	TC-004
Testing Purpose	Testing manage users function
Scenario 1	*Admin edit the user's data*
Initial Condition	User has registered on system
Testing Procedure	1. Actor choose manage users menu 2. System show the list of users 3. Actor choose the user that want to be changed 4. System show the form to update the user. 5. Actor fill the form to update user's data

	6. System save the updated data
Input	1. Username 2. Email 3. Status 4. Phone 5. Address 6. Avatar
Expected Result	Admin has successfully updated the user's data
Testing Result	Admin has successfully updated the user's data
Status	Success

5.3.5 View dashboard Test case

This test case is used to view dashboard function. Detail of the test cases are shown in Table 5.6

Table 5.6 Detail of View Dashboard Test Case

Testing Scenario Name	The functionality of view dashboard
Code	TC-005
Testing Purpose	Testing of view dashboard function
Scenario 1	*User logged in as admin*
Initial Condition	User is exist on the database and has role as admin
Testing Procedure	1. System shows the login page 2. User enter his credential as an admin 3. System redirect the user to the dashboard
Input	1. Email 2. Password
Expected Result	System redirect the user to dashboard
Testing Result	System redirect the user to dashboard
Status	Success

5.3.6 Register Test Case

This test case is used to test register function. Detail of the test cases are shown in Table 5.7

Table 5.7 Detail of Register Test Case

Testing Scenario Name	The functionality of register
Code	TC-006
Testing Purpose	Testing of register function
Scenario 1	*User register into system*
Initial Condition	User has not been registered on system
Testing Procedure	1. Actor click register button 2. System show registration form 3. Actor fill the registration form 4. System save the registration data
Input	1. Username 2. Password 3. Email
Expected Result	User successful registered on system
Testing Result	User successful registered on system
Status	Success

5.3.7 Login Test Case

This test case is used to test login function. Detail of the test cases are shown in Table 5.8

Table 5.8 Detail of Login Test Case

Testing Scenario Name	The functionality of login
Code	TC-007
Testing Purpose	Testing of login function
Scenario 1	*User logged in as customer*
Initial Condition	User has not logged in into system
Testing Procedure	1. Actor go to login page 2. Actor enter his credential 3. System redirect the user into customer homepage
Input	1. Email 2. Password
Expected Result	User has successfully login into system
Testing Result	User has successfully login into system
Status	Success
Scenario 1	*User logged in as admin*
Initial Condition	User has not logged in into system
Testing Procedure	1. Actor go to login page 2. Actor enter his credential 3. System redirect the user into dashboard
Input	1. Email 2. Password
Expected Result	User has successfully login into system
Testing Result	User has successfully login into system
Status	Success

5.3.8 Search Travel Destination Test Case

This test case is used to test search travel destination function. Detail of the test cases are shown in Table 5.9

Table 5.9 Search travel Destination Test Case

Testing Scenario Name	The functionality of search travel destination
Code	TC-008
Testing Purpose	Testing of search travel destination function
Scenario 1	User search the destination
Initial Condition	System show the homepage
Testing Procedure	1. Actor go to search destination tab content 2. Actor enter the keyword of the destination 3. Actor click search button 4. System show the result
Input	1. Keyword
Expected Result	System show the destinations based on keyword entered by actor
Testing Result	System show the destinations based on keyword entered by actor
Status	Success

5.3.9 Manage Booking Cart Test Case

This test case is used to test manage booking cart function. Detail of the test cases are shown in Table 5.10

Table 5.10 Detail of Manage Booking Cart Test Case

Testing Scenario Name	The functionality of manage booking cart
Code	TC-009
Testing Purpose	Testing of manage booking cart function
Scenario 1	User want to add destination into cart
Initial Condition	The cart is empty

Testing Procedure	1. Actor choose the destination 2. Actor click Add to Cart Button 3. The destination has added into cart 4. Actor click the cart icon 5. System show the actor's cart
Input	-
Expected Result	Actor successfully add destination on cart
Testing Result	Actor successfully add destination on cart
Status	Success
Scenario 2	User want to clear cart
Initial Condition	The cart is not empty
Testing Procedure	1. Actor click the cart icon 2. System show the actor's cart 3. Actor click clear cart 4. The cart is empty
Input	-
Expected Result	The cart is empty
Testing Result	The cart is empty
Status	Success

5.3.10 Payment Test Case

This test case is used to test payment function. Detail of the test cases are shown in Table 5.11

Table 5.11 Detail of Payment Test Case

Testing Scenario Name	The functionality of payment
Code	TC-010
Testing Purpose	Testing of payment function
Scenario 1	Actor want to make a payment
Initial Condition	The cart is not empty
Testing Procedure	1. Actor click the cart icon 2. Actor click continue to pay 3. System show the detail of payment

	4. Actor gets the ticket
Input	-
Expected Result	System show the detail of payment
Testing Result	System show the detail of payment
Status	Success

5.3.11 View Tickets Test Case

This test case is used to test view tickets function. Detail of the test cases are shown in Table 5.12

Table 5.12 Detail of View Tickets

Testing Scenario Name	The functionality of view ticket
Code	TC-011
Testing Purpose	Testing of view tickets function
Scenario 1	*User want to view the ticket*
Initial Condition	Payment is complete
Testing Procedure	1. Actor click My Tickets menu 2. System show the list of actor's tickets 3. Actor click show ticket button 4. System show the detail of ticket
Input	-
Expected Result	Actor can view the ticket
Testing Result	Actor can view the ticket
Status	Success

5.3.12 Manage User's Profile Test Case

This test case is used to test manage user's profile function. Detail of the test cases are shown in Table 5.13

Table 5.13 Detail of Manage User's Profile Test Case

Testing Scenario Name	The functionality of manage user's profile
Code	TC-012
Testing Purpose	Testing of manage user's profile function
Scenario 1	*User want to update the profile*
Initial Condition	User's Profile has not been changed
Testing Procedure	1. Actor click My Profile menu 2. System show existing the actor's profile 3. Actor click update button 4. System show the update profile form 5. Actor click save button 6. System save the updated data
Input	
Expected Result	User's Profile has successfully updated
Testing Result	User's Profile has successfully updated
Status	Success

5.4 Test Case Recapitulation

Below is a summary of the test cases conducted above. The recapitulation results of the test cases are shown on Table 5.21

Table 5.14 Recapitulation of Test Case

No	Functionality	Success	
		Yes	No
1	Create travel destination	✓	
2	Update travel destination	✓	
3	View travel destination	✓	
4	Manage Users	✓	
5	View dashboard	✓	
6	Register	✓	
7	Login	✓	
8	Search travel destination	✓	

9	Manage booking cart	✓	
10	Checkout	✓	
11	View tickets	✓	
12	Manage user profile	✓	

5.5 Performance Test

In this section we will explain about performance test. This test is done in local environment using ReadyAPI 3.2.7

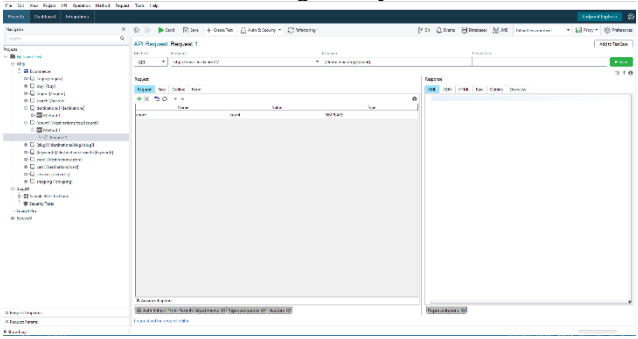


Figure 33 Screenshoot of ReadyAPI Application

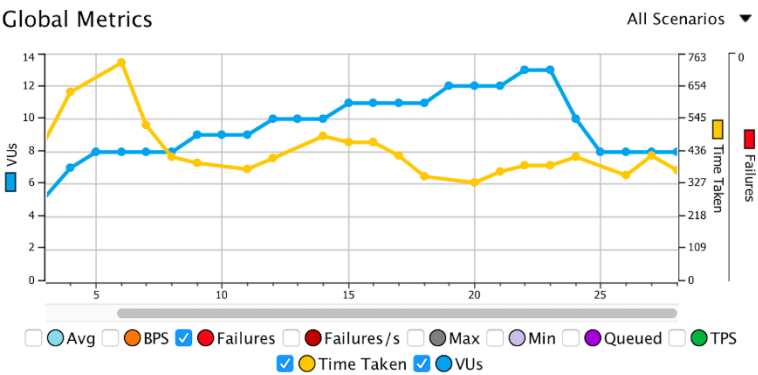


Figure 34 Performance Test Result

The test result, indicate that the functional requirement is running well in ideal conditions. But in non ideal condition, maybe we will find some errors in application. So, these are some of non functional requirement that need to be improved.

1. Caching Mechanism
   Caching is a mechanism for a temporary storage of web pages in order to reduce bandwidth and improve performance. When a visitor arrives at your site the cached version will be served up unless it has changed since the last cache. This saves server time and makes things altogether faster. This application still doesn't support this. So, in the next development, we can use Redis. Redis essentially is a cache - a place to store run time artifacts for a highly scalable distributed system.
2. Server Infrastructure
   Infrastructure is the foundation or framework that supports a system or organization. In computing, information technology server infrastructure is composed of physical and virtual resources that support the flow, storage and processing time. There are some points that must be considered : budget, the number of users and the amount of data. In the beginning of development, we can deploy shared hosting or VPS (Virtual Private Server). When the apps is grow up, we can build a dedicated server or use third party Cloud Service Provider like : Amazon Web Service (AWS), Microsoft Azure, etc.

[This page has been intentionally left blank]

CHAPTER VI
CONCLUSIONS AND RECOMMENDATIONS

This chapter consists of conclusion and the suggestion of the research. The conclusion can be used as a guide and reference to the to web and application programmers, while suggestion is for the better future research.

6.2 Conclusions

The conclusion after conducting this research are:

1. In order completing this Final project few important steps were taken into consideration including: Project Proposal, Literature review, Software Analysis and Design, Implementation, Testing and evaluation and Final project booklet.
2. Web service uses two method to process data which is: GET (get data from database) and POST (post data on database).
3. This final project is using REST API, and consisting of Frontend and Backend. The Frontend usually shows the view (web) using Vue Js while the backend is the data from database are transferred (text only) using MYSQL database.
4. It can be concluded that Travel Information System if Fiji conceptualization promises to provide a firmer base for Web applications development at the initial stages of design. It is possible to integrate this system approach with current methods of software development processes.
5. The travel Information System in Fiji is user-friendly and still allowing researchers for improvements in the future.

6.3 Recommendations

The suggestions for the better future researchers are:

1. This final project was completed in a short time. It would be highly recommended for a good project more time was given.
2. The future researchers would be able to complete the payment process by producing flight tickets and Itinerary.
3. Need to explore further needs in coding with relevant users for future system development.
4. And lastly, I would strongly recommend that future researchers to explore more on this project to develop a good International Information System.

BIBLIOGRAPHY

[1] "Laravel (Framework)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Laravel. [Accessed 2019 July 4].

[2] "VueJS," [Online]. Available: Vue is a simple and minimal progressive JavaScript framework that can be used to build powerful . [Accessed 2019 July 4].

[3] "REST API," [Online]. Available: https://searchapparchitecture.techtarget.com/definition/RESTful-API.

[4] "MySQL," 2017 Januari 26. [Online]. Available: https://www.123-reg.co.uk/support/servers/what-is-mysql-and-why-do-i-need-it/. [Accessed 2019 Desember 26].

[5] "PHP," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/PHP.

[6] "JSON," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/JSON.

[7] "MVC (Model-View-Controller)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Accessed 15 July 2019].

[8] "MVVM(Model-Vew-ViewModel)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel. [Accessed 2019 December 27].

APPENDIX

Figure A. 1 Class Diagram

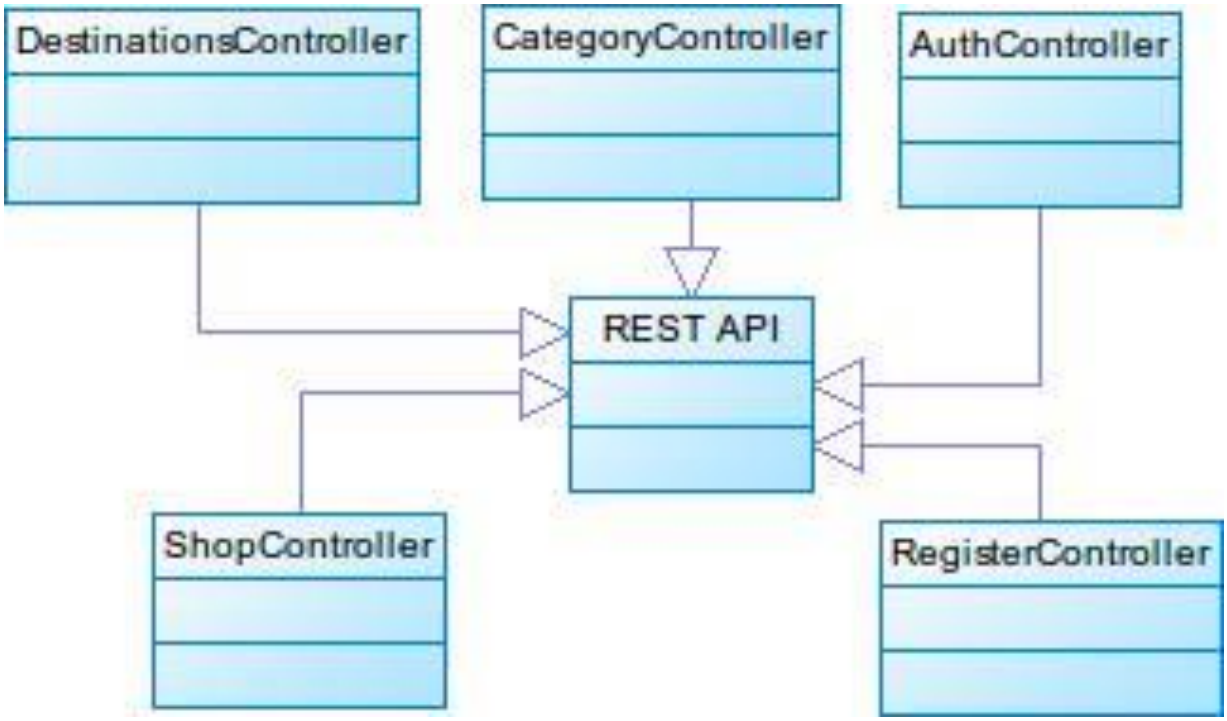
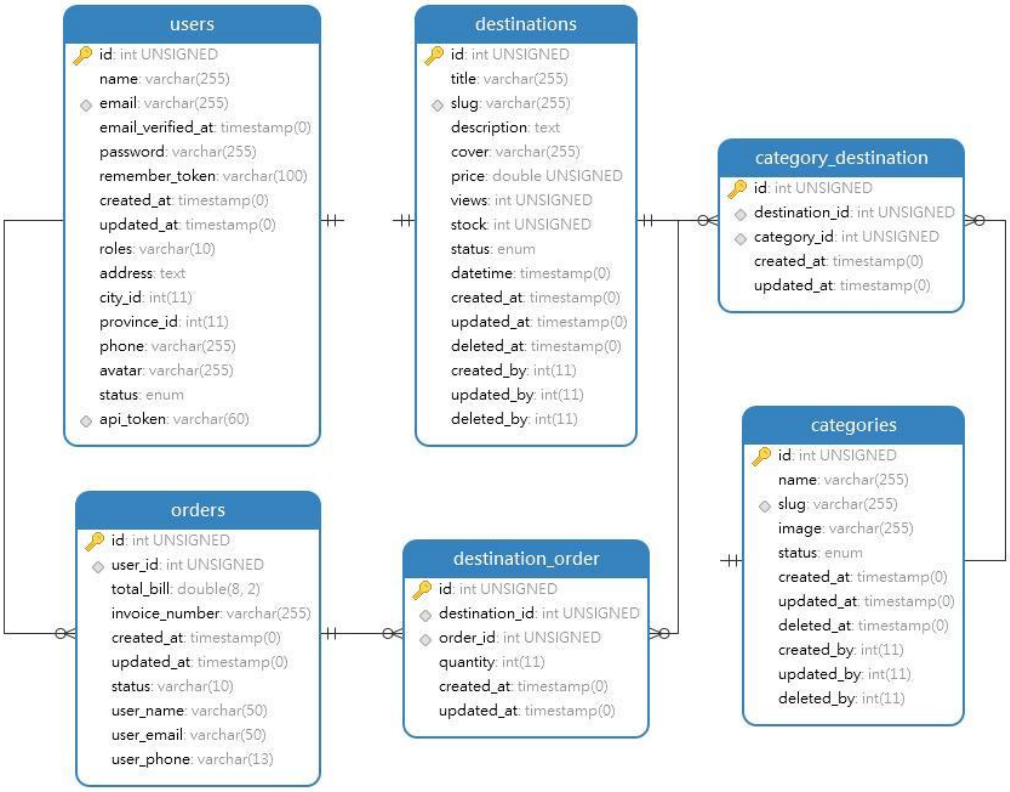


Figure A. 2 Conceptual Data Model



AUTHOR'S BIODATA

The author, Nemesio Rokoqera Raitubu was born in Taveuni, Fiji Island on January 2nd 1989. The author attended Niusawa Primary School in Taveuni (1994- 2000) and to Sila Central High School in Nausori for High School (2001 – 2007). Since childhood, the author has a great interest in the field of computers so the author decided to take a Diploma in Business Studies Majoring in Computing at Fiji National University with a certificate in teaching (2008-2010), In 2011 – 2015 the author was teaching in various schools in Fiji before applying for a scholarship to continue his studies.

In Mid-2015, author applied for Indonesian Government Scholarship and was accepted as one of the KNB scholarship recipient in September 2015. The author was accepted at the Faculty of Information and Communication Technology, Sepuluh November Institute of Technology Surabaya to pursue his Bachelor in Informatics Engineering.

As a college student, the author had actively participated in several activities held in campus and outside from campus. Author join several international programs that was organized by the ITS International office. Lastly, author is a kind and open minded person, if there is any queries, please do not hesitate to contact him via email: nemesioroko15@gmail.com