**CHAPTER 1:**

**INTRODUCTION**

Flight booking system will hold flight schedules and it fare tariffs, passenger reservations and ticket records. It saves time as it allows online procedure as user no longer to wait in a queue to book the flights. It automatically generated but the server. admin is the main authority who can do addition, deletion, and modifications of flights if required.

The project has planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of data base MYSQL and all the user interfaces have been designed using the adobe dream weaver technologies.

The flight booking system project is an implementation of a general flight booking website like Orbitz, which helps the customizers to search the availability and prices of various airline tickets, along with the different packages available with the booking.

This project also covers various features like online registration of the users, modifying the details of the website by the management staff or administrator of the website, by adding ,deleting, modifying the customer details ,flight or packages information. In general, this website would be designed to perform like any other plane ticketing website available online.

* 1. **Project Objective**
* Complex functions as done automatically
* Processing time can be minimized
* Simple and easy to manage
* Chances of errors reduced
* Faster and more accurate than existing system
* Easy for handling reports
  1. **Project Overview**

Filght booking system contains the details about flight schedules and its fare tariffs, passenger reservations and ticket records. Air alliance operates flights to 30 destinations in India namely

**CHAPTER 2:**

**EXISTING AND PROPOSED SYSTEM**

**2.1 Existing System**

In few countries if a person wants to book a flight ticket, he uses to follow one of these things:

**2.1.1 Disadvantages**

* Manually goes to the Airport and book his ticket.
* Downloading the ticket form as paper document, filling it manually and submitting

at Airport.

* Fill the Ticket form on system and get the print out as paper documents to submit it at Airport.
* Booking the Ticket at some particular registered ticket counters in online.
* Even above approaches make a ticket booking online, it was not completely done on online. Passenger may not have much freedom over this approach
* Hence the Passenger may or may not be satisfied with this approach as it includes manual intervention like travelling to Airport for booking his ticket.

**2.2 Proposed System**

The Proposed systems ensures the complete freedom for users, where user at his own system can login to this website and can book his ticket, our proposed system allows only registered users to book the tickets, view timings and cancel their tickets.

In this Proposal the entire work is done on online and ticket with id is also provided for passengers as a print document. Here passengers can send their queries and suggestions through a feedback form.

To debug the existing system, remove procedures those cause data redundancy, Make navigational sequence proper to build strong password mechanism.

**CHAPTER 3:**

**SOFTWARE AND HARDWARE SPECIFICATION**

Requirement analysis for web application encompasses the major tasks formulation, requirements gathering and analysis modelling. During formulation, motivation and goals for the web application are identified, and the categories of unsnarl defined. In the requirements gathering place, the comment and fictional require and interaction scenarios writes from end-user's point-of-view are developed. This establish a basic understanding of why the web application is built, who will use it, and what problems it will solve for its users.

**3.1 Software Requirements**

Operating System : Windows 10

User Interface : HTML, PHP

Database : MY SQL

Application Server : XAMPP Server

**3.2 Hardware Requirements**

Processor : Any processor above 500 MHz

Ram : 4GB

Hard Disk : 2GB free space

Input Device : Keyboard, Mouse

Output Device : Monitor

System type : 32-bit or 64-bit operating system

**CHAPTER 4**

**SYSTEM DESIGN**

The purpose of System Design is to create a technical solution. That satisfies the functional requirements for the system. At this Paint in the project lifecycle there should be Functional Specification, written primarily in business terminology, containing complete description of the operational needs of the various organizational entities that will use the new system

The Challenge is to translate all of this information into Technical Specifications that specifications describe the design of the system, and that can be used as input to System Construction. The Functional Specification produced during System Requirements Analysis is transformed into a physical architecture. System components are distributed across the physical architecture, usable interfaces are designed and prototyped, and technical Specifications are created for the Application Developers, enabling them to build and test the system. Many organizations look at System Design primarily as the Preparation of the system component specifications; however, constructing various system components is only one of a set off major step in successfully building system. The preparation of the environment seeded to build the system, the testing of the system, and the migration and preparation of the data that will ultimately be used by the system are equally important.

**4.1 Usecase Diagram**

**4.2 Schema Diagram**

It formulates all the constraints that are to be applied on the data. The database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by the means of schema diagram

**4.3 ER-Diagram**

The Entity-Relationship (ER) model was originally proposed by Peter in 1076 [Chen76] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Che wrote his paper the model has been extended and today it is commonly used for database design for the database designer, the utility of the ER model.

It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables. It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.

**Connectivity and Cardinality**

A one-to any(1:N) relationship is when for one instance of entity A, there are zero, one, or many instances of entity B, but for none instance of entity B, there is only one instance of entity A. An example of 1:N relationship is department has many employees each employee is assigned to one department.

A many-to many(M:N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instance of entity Band for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of relationship describes the mapping of associated.

**ER Notation**

There is no standard for representing data objects in ER diagrams. Each modelling methodology uses its own notations. The original notation used by when is widely used in academics texts and journals but rarely seen in either CASE tools or publications by non-academics. Today, where are a number of notations used, amo ng the more common are Bachman, crow's foot, and idefix.

All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in the document is from Martin. The symbols used for the basic ER constructs are: Entities are represented by a labelled rectangle. The label is the name of the entity. Entity names should be singular nouns.

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Front End**

PHP, which stands "PHP: Hypertext Preprocessor” is a widely used Open-Source general purpose scripting language the is especially suited to Web development and can be embedded into HTML. Its syntax draws upon C, Java, and ped, and very to lean. The main goal of language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP.

**5.2 Technology used for Implementation (PHP)**

**5.2.1 PHP:**

PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be a more seine language for writing Ca programs than Perles C. and with correct selection of compile time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need. As there are many different ways of utilizing PHP, there are many configuration options controlling its behavior. A large selection of options guarantees you can use PHP for a lot of purpose, but it also means there are combinations of these options and server configurations that result in an insecure setup. The configuration flexibility of PHP is equally rivaled by the code flexibility. PHP can be used to build complete server applications, with all the power of shell user, or it can be used for simple server side includes with little risk in a tightly controlled environment. How you build that environment, and how secure it is, is largely up to the PHP developer

**5.2.2 Back End**

**Database**:

A Database Management System (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise, DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems. Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office A DEMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database.

**SQL:**

Structured Query Language (SQL) in the language used to manipulate relational databases. SQL is tied very closely with the relational model.

* In the relational model, data is stored in structures called relations or tables.

SQL statements are issued for the purpose of

* Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes)
* Data manipulation: Used to manipulate the data within those schema objects (DML Inserting, Updating. Deleting the data, and Querying the Database)

**5.3 CODING:**

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

-- Database: `flight\_booking\_db`

-- Table structure for table `airlines\_list`

CREATE TABLE `airlines\_list` (

`id` int(30) NOT NULL,

`airlines` text NOT NULL,

`logo\_path` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `airlines\_list`

INSERT INTO `airlines\_list` (`id`, `airlines`, `logo\_path`) VALUES

(1, 'AirAsia', '1600999080\_kisspng-flight-indonesia-airasia-airasia-japan-airline-tic-asia-5abad146966736.8321896415221927106161.jpg'),

(2, 'Philippine Airlines', '1600999200\_Philippine-Airlines-Logo.jpg'),

(3, 'Cebu Pacific', '1600999200\_43cada0008538e3c1a1f4675e5a7aabe.jpeg');

-- Table structure for table `airport\_list`

CREATE TABLE `airport\_list` (

`id` int(30) NOT NULL,

`airport` text NOT NULL,

`location` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `airport\_list`

INSERT INTO `airport\_list` (`id`, `airport`, `location`) VALUES

(1, 'NAIA', 'Metro Manila'),

(2, 'Beijing Capital International Airport', 'Chaoyang-Shunyi, Beijing'),

(3, 'Los Angeles International Airport', 'Los Angeles, California'),

(4, 'Dubai International Airport', 'Garhoud, Dubai'),

(5, 'Mactan-Cebu Airport', 'Cebu');

-- Table structure for table `booked\_flight`

CREATE TABLE `booked\_flight` (

`id` int(30) NOT NULL,

`flight\_id` int(30) NOT NULL,

`name` text NOT NULL,

`address` text NOT NULL,

`contact` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `booked\_flight`

INSERT INTO `booked\_flight` (`id`, `flight\_id`, `name`, `address`, `contact`) VALUES

(2, 3, 'James Smith', 'Sample Address', '+4545 6456'),

(3, 4, 'John Smith', 'Sample Address', '+18456-5455-55');

-- Table structure for table `flight\_list`

CREATE TABLE `flight\_list` (

`id` int(30) NOT NULL,

`airline\_id` int(30) NOT NULL,

`plane\_no` text NOT NULL,

`departure\_airport\_id` int(30) NOT NULL,

`arrival\_airport\_id` int(30) NOT NULL,

`departure\_datetime` datetime NOT NULL,

`arrival\_datetime` datetime NOT NULL,

`seats` int(10) NOT NULL DEFAULT 0,

`price` double NOT NULL,

`date\_created` datetime NOT NULL DEFAULT current\_timestamp()) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `flight\_list`

INSERT INTO `flight\_list` (`id`, `airline\_id`, `plane\_no`, `departure\_airport\_id`, `arrival\_airport\_id`, `departure\_datetime`, `arrival\_datetime`, `seats`, `price`, `date\_created`) VALUES

(1, 1, 'GB623-14', 1, 3, '2020-10-07 04:00:00', '2020-10-21 10:00:00', 150, 7500, '2020-09-25 11:23:52'),

(2, 2, 'TIPS14-15', 1, 2, '2020-10-14 11:00:00', '2020-10-16 09:00:00', 100, 5000, '2020-09-25 11:46:12'),

(3, 3, 'CEB-1101', 5, 1, '2020-09-30 08:00:00', '2020-09-30 08:45:00', 100, 2500, '2020-09-25 11:57:31'),

(4, 3, 'CEB10023', 1, 5, '2020-10-07 01:00:00', '2020-10-07 01:45:00', 100, 2500, '2020-09-25 14:50:47');

-- Table structure for table `system\_settings`

CREATE TABLE `system\_settings` (

`id` int(30) NOT NULL,

`name` text NOT NULL,

`email` varchar(200) NOT NULL,

`contact` varchar(20) NOT NULL,

`cover\_img` text NOT NULL,

`about\_content` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `system\_settings`

INSERT INTO `system\_settings` (`id`, `name`, `email`, `contact`, `cover\_img`, `about\_content`) VALUES

(1, 'Online Flight Booking System', 'info@sample.comm', '+6948 8542 623', '1600998360\_travel-cover.jpg', '&lt;p

style=&quot;text-align:

center; background: transparent;

position: relative;&quot;&gt;&lt;span style=&quot;

background: transparent;

position: relative;

font-size: 14px;&quot;&gt;&lt;span style=&quot;

font-size:28px;

background: transparent;

position: relative;&quot;&gt;&lt;b style=&quot;

margin: 0px;

padding: 0px;

color: rgb(0, 0, 0);

font-family: &amp;quot;OpenSans&amp;quot;, Arial, sans-serif; text-align: justify;&quot;&gt;LoremIpsum&lt;/b&gt;&lt;span style=&quot;

color: rgb(0, 0, 0);

font-family: &amp

;quot;OpenSans&amp;

quot;, Arial, sans-serif;

font-weight: 400;

text-align: justify;&quot;&gt;&amp;

nbsp;is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry&rsquo;s standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was pop

ularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.&lt;/span&gt;&lt;br&gt;&lt;/span&gt;&lt;/p&gt;&lt;p style=&quot;text-align: center;

background: transparent;

position: relative

;&quot;&gt;&lt;br&gt;&lt;/p&gt;&lt;p&gt;&lt;/p&gt;');

-- Table structure for table `users`

CREATE TABLE `users` (

`id` int(30) NOT NULL,

`doctor\_id` int(30) NOT NULL,

`name` varchar(200) NOT NULL,

`address` text NOT NULL,

`contact` text NOT NULL,

`username` varchar(100) NOT NULL,

`password` varchar(200) NOT NULL,

`type` tinyint(1) NOT NULL DEFAULT 2 COMMENT '1=admin , 2 = doctor,3=patient'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `users`

INSERT INTO `users` (`id`, `doctor\_id`, `name`, `address`, `contact`, `username`, `password`, `type`) VALUES

(1, 0, 'Administrator', '', '', 'admin', 'admin123', 1),

(7, 0, 'George Wilson', 'Sample Only', '+18456-5455-55', 'gwilson@sample.com', 'd40242fb23c45206fadee4e2418f274f', 3),

(9, 2, 'DR.James Smith, M.D.', 'Sample Clinic Address', '+1456 554 55623', 'jsmith@sample.com', 'jsmith123', 2),

(10, 3, 'DR.Claire Blake, M.D.', 'Sample Only', '+5465 555 623', 'cblake@sample.com', 'blake123', 2),

(11, 0, 'Sample Only', 'Sample', '+5465 546 4556', 'sample@sample.com', '4e91b1cbe42b5c884de47d4c7fda0555', 3),

(15, 9, 'DR.Sample Doctor, M.D.', 'Sample Address', '+1235 456 623', 'sample2@sample.com', 'sample123', 2);

-- Indexes for dumped tables

-- Indexes for table `airlines\_list`

ALTER TABLE `airlines\_list`

ADD PRIMARY KEY (`id`);

-- Indexes for table `airport\_list`

ALTER TABLE `airport\_list`

ADD PRIMARY KEY (`id`);

-- Indexes for table `booked\_flight`

ALTER TABLE `booked\_flight`

ADD PRIMARY KEY (`id`);

-- Indexes for table `flight\_list`

ALTER TABLE `flight\_list`

ADD PRIMARY KEY (`id`);

-- Indexes for table `system\_settings`

ALTER TABLE `system\_settings`

ADD PRIMARY KEY (`id`);

-- Indexes for table `users`

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

-- AUTO\_INCREMENT for dumped tables

-- AUTO\_INCREMENT for table `airlines\_list`

ALTER TABLE `airlines\_list`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

-- AUTO\_INCREMENT for table `airport\_list`

ALTER TABLE `airport\_list`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

-- AUTO\_INCREMENT for table `booked\_flight`

ALTER TABLE `booked\_flight`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

-- AUTO\_INCREMENT for table `flight\_list`

ALTER TABLE `flight\_list`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

-- AUTO\_INCREMENT for table `system\_settings`

ALTER TABLE `system\_settings`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

-- AUTO\_INCREMENT for table `users`

ALTER TABLE `users`

MODIFY `id` int(30) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=16;

COMMIT;

**LOGIN PAGE CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta content="width=device-width, initial-scale=1.0" name="viewport">

<title>Admin | Online Flight Booking System</title>

<?php include('./header.php'); ?>

<?php include('./db\_connect.php'); ?>

<?php

session\_start();

if(isset($\_SESSION['login\_id']))

header("location:index.php?page=home");

?>

</head>

<style>

body{

width: 100%;

height: calc(100%);

/\*background: #007bff;\*/

}

main#main{

width:100%;

height: calc(100%);

background:white;

}

#login-right{

position: absolute;

right:0;

width:40%;

height: calc(100%);

background:white;

display: flex;

align-items: center;

}

#login-left{

position: absolute;

left:0;

width:60%;

height: calc(100%);

background:#59b6ec61;

display: flex;

align-items: center;

background: url(../assets/img/travel-cover.jpg);

background-repeat: no-repeat;

background-size: cover;

}

#login-right .card{

margin: auto;

z-index: 1

}

.logo {

margin: auto;

font-size: 8rem;

background: white;

padding: .5em 0.7em;

border-radius: 50% 50%;

color: #000000b3;

z-index: 10;

}

div#login-left::before,div#login-right::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: calc(100%);

height: calc(100%);

background: #00c4ff36;

}

</style>

<body>

<main id="main" class=" bg-dark">

<div id="login-left">

<div class="logo">

<span class="fa fa-plane-departure"></span>

</div>

</div>

<div id="login-right">

<div class="card col-md-8">

<div class="card-body">

<form id="login-form" >

<div class="form-group">

<label for="username" class="control-label">Username</label>

<input type="text" id="username" name="username" class="form-control">

</div>

<div class="form-group">

<label for="password" class="control-label">Password</label>

<input type="password" id="password" name="password" class="form-control">

</div>

<center><button class="btn-sm btn-block btn-wave col-md-4 btn-primary">Login</button></center>

</form>

</div>

</div>

</div>

</main>

<a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>

</body>

<script>

$('#login-form').submit(function(e){

e.preventDefault()

$('#login-form button[type="button"]').attr('disabled',true).html('Logging in...');

if($(this).find('.alert-danger').length > 0 )

$(this).find('.alert-danger').remove();

$.ajax({

url:'ajax.php?action=login',

method:'POST',

data:$(this).serialize(),

error:err=>{

console.log(err)

$('#login-form button[type="button"]').removeAttr('disabled').html('Login');

},

success:function(resp){

if(resp == 1){

location.href ='index.php?page=home';

}else if(resp == 2){

location.href ='voting.php';

}else{

$('#login-form').prepend('<div class="alert alert-danger">Username or password is incorrect.</div>')

$('#login-form button[type="button"]').removeAttr('disabled').html('Login');

}

}

})

})

</script>

</html>

**CHAPTER 6**

**TESTING**

The testing process focuses on the logical intervals of the software ensuring that all statements have been tested and on functional interval is conducting tests to uncover errors and ensure that defined input will produce actual results that agree with the required results Program level testing, modules level testing integrated and carried out.

**6.1 Unit testing**

Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use.

A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure.

Ideally, each test case is independent from the others: substitutes like method stubs, objects, fakes and test harnesses can be used to assist testing a module in isolation.

**6.2 Integration Testing**

This testing is sometimes called Integration and Testing. Integration testing is the phase in software testing in which individual software modules are combined and tested as a group It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.

**6.3 Validation Testing**

Validation Testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can reasonably expected by a customer. After validation test has been conducted, one of the following two possible conditions exists. The functions or performance characteristics confirm to specification and areaccepted.

* In the administrator and login modules, all the fields must be filled.
* In the student registration, contact number should contain exactly 10 numbers.

**6.4 User Acceptance Testing**

User acceptance of a system is a key factor of any system. The system under code ration is tested for the acceptance by constantly keeping in touch with the prospective system users at the same time of developing and marketing changes whomever required. This is done in regard to the following points:

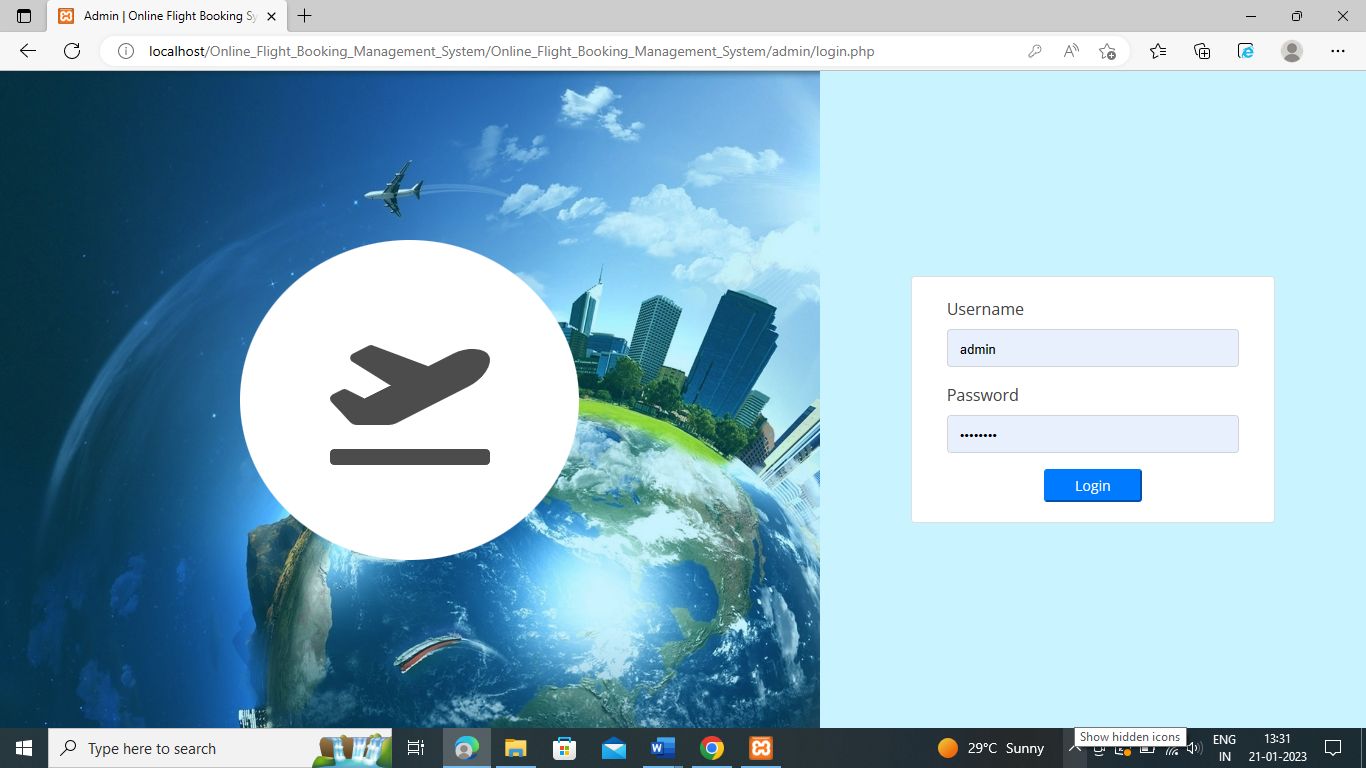
* Input Screen Design
* Output Screen Design

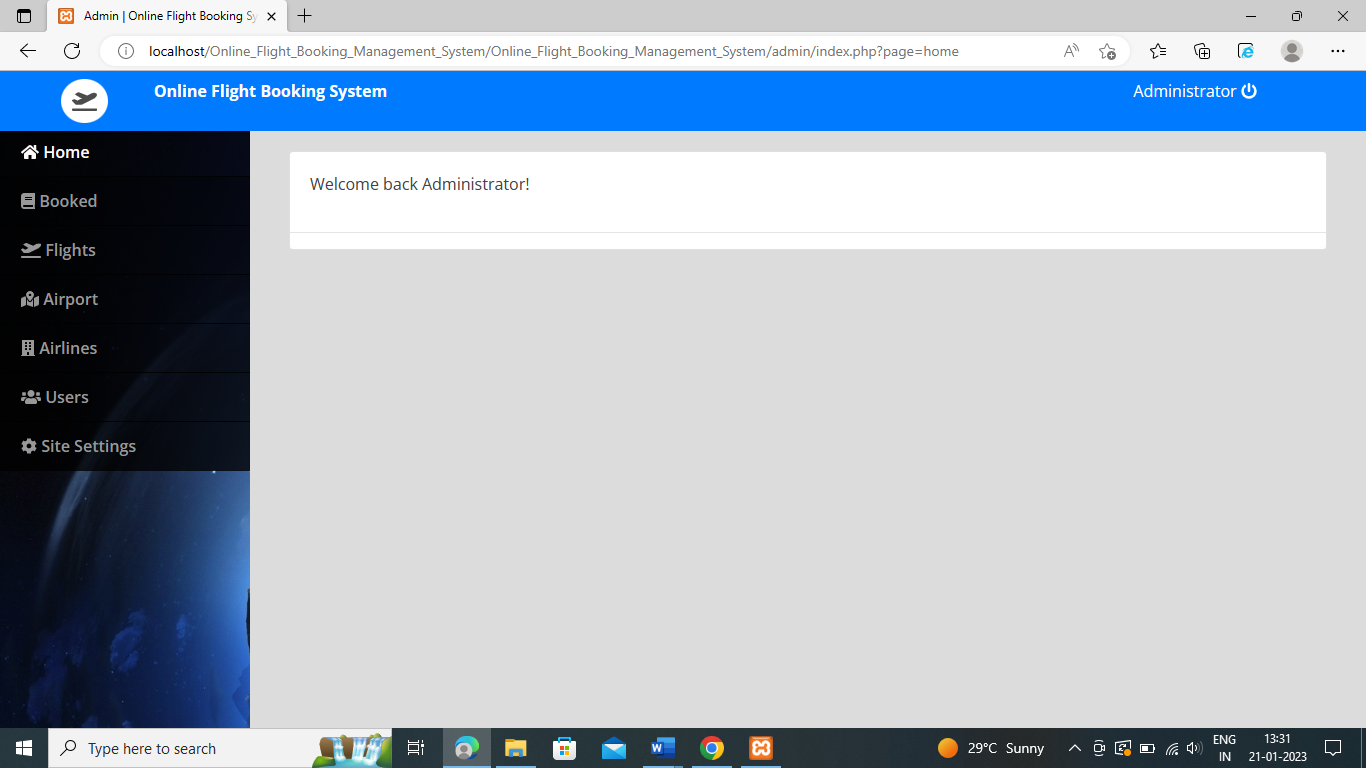
**CHAPTER 7**

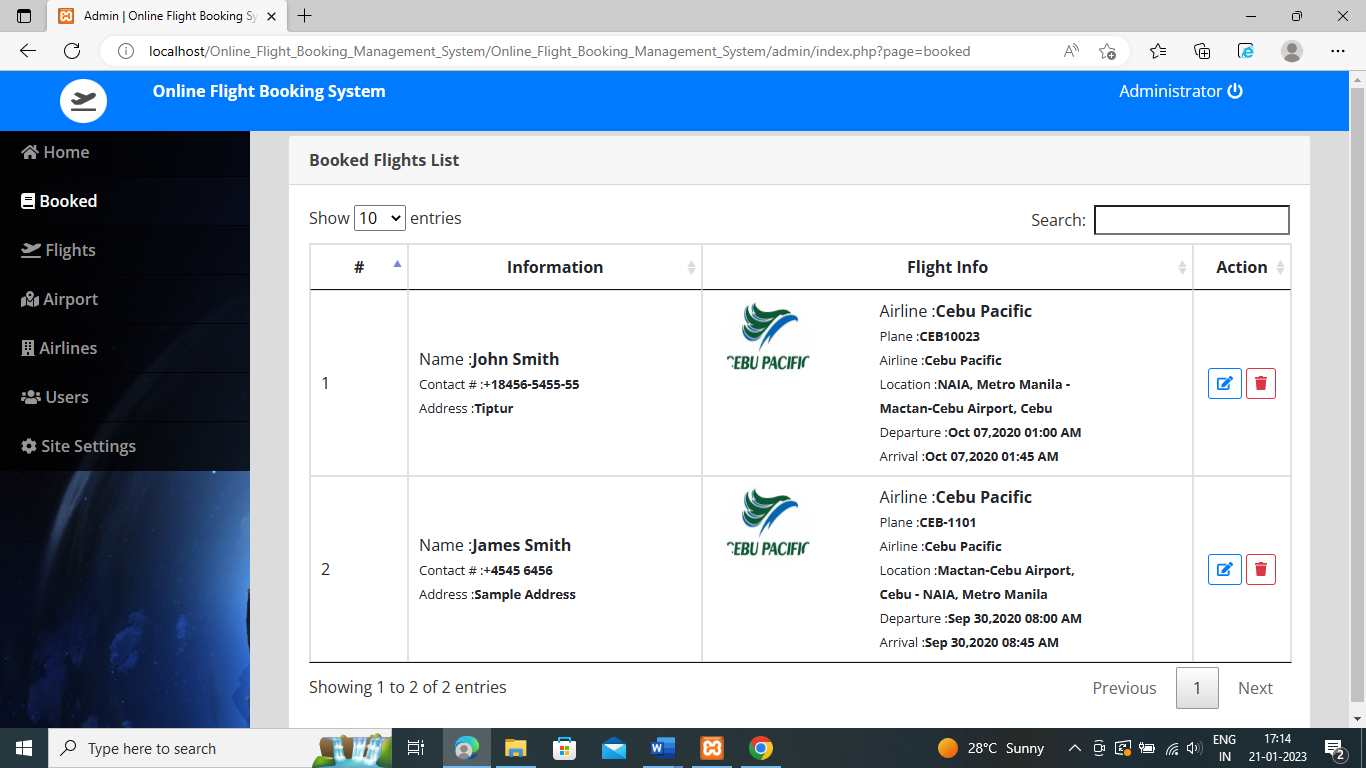
**SNAPSHOTS**

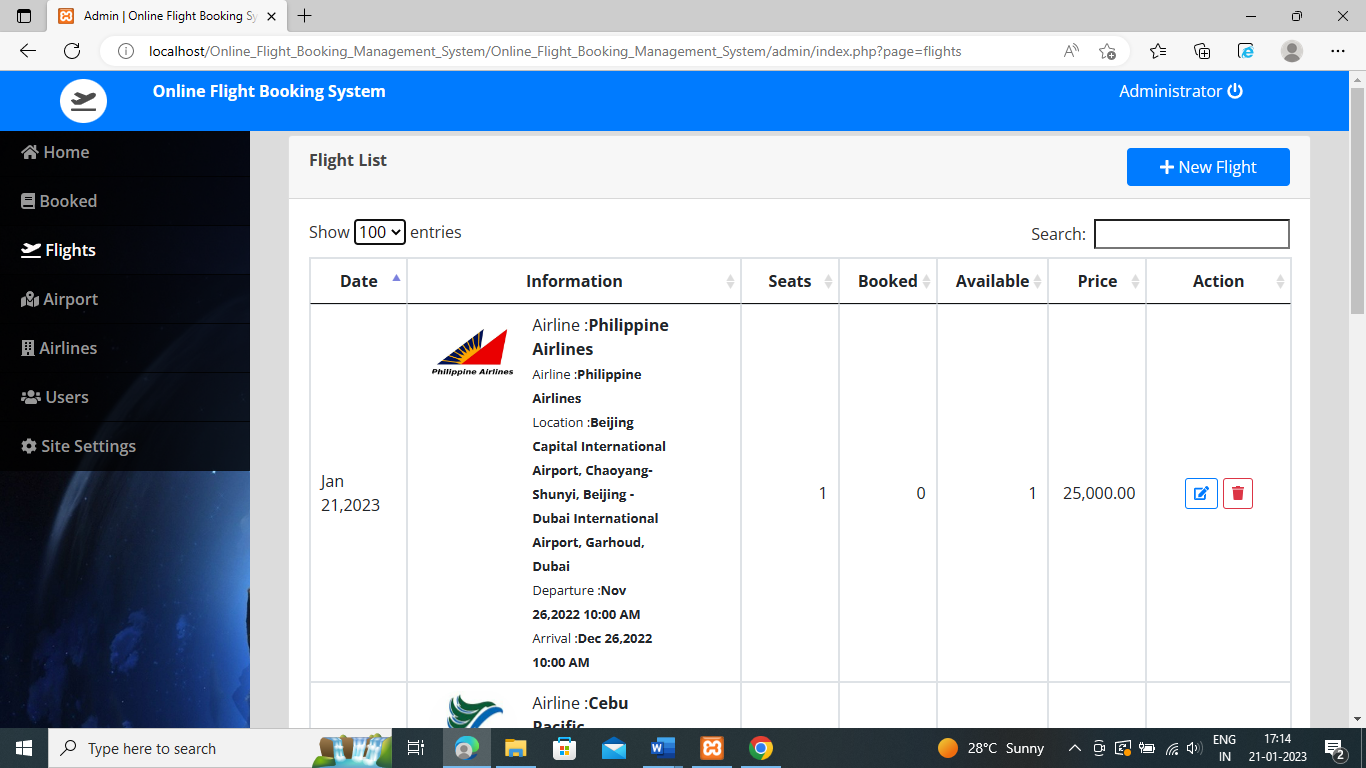
The section describes the screed of the "FLIGHT TICKET BOOKING. The snapshots shown below for each module.

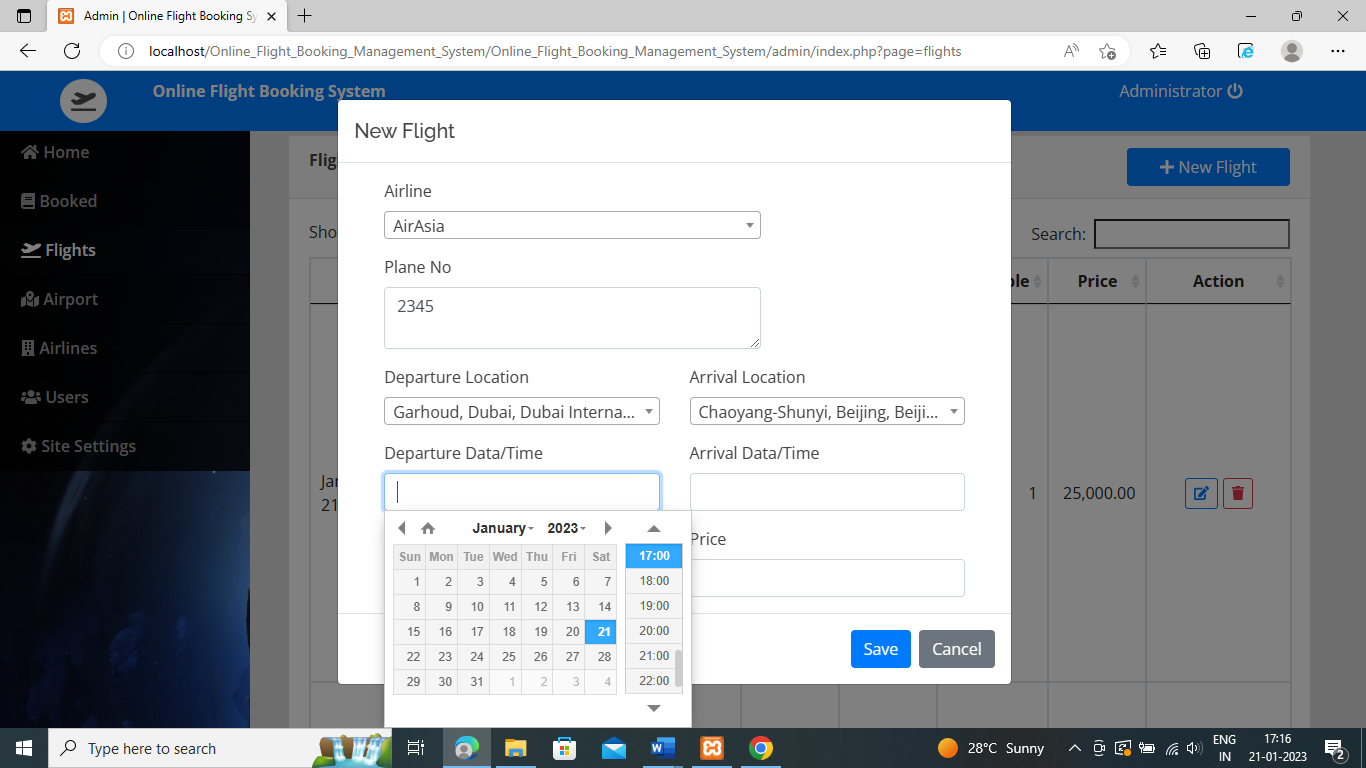
**7.1 Admin login page:**

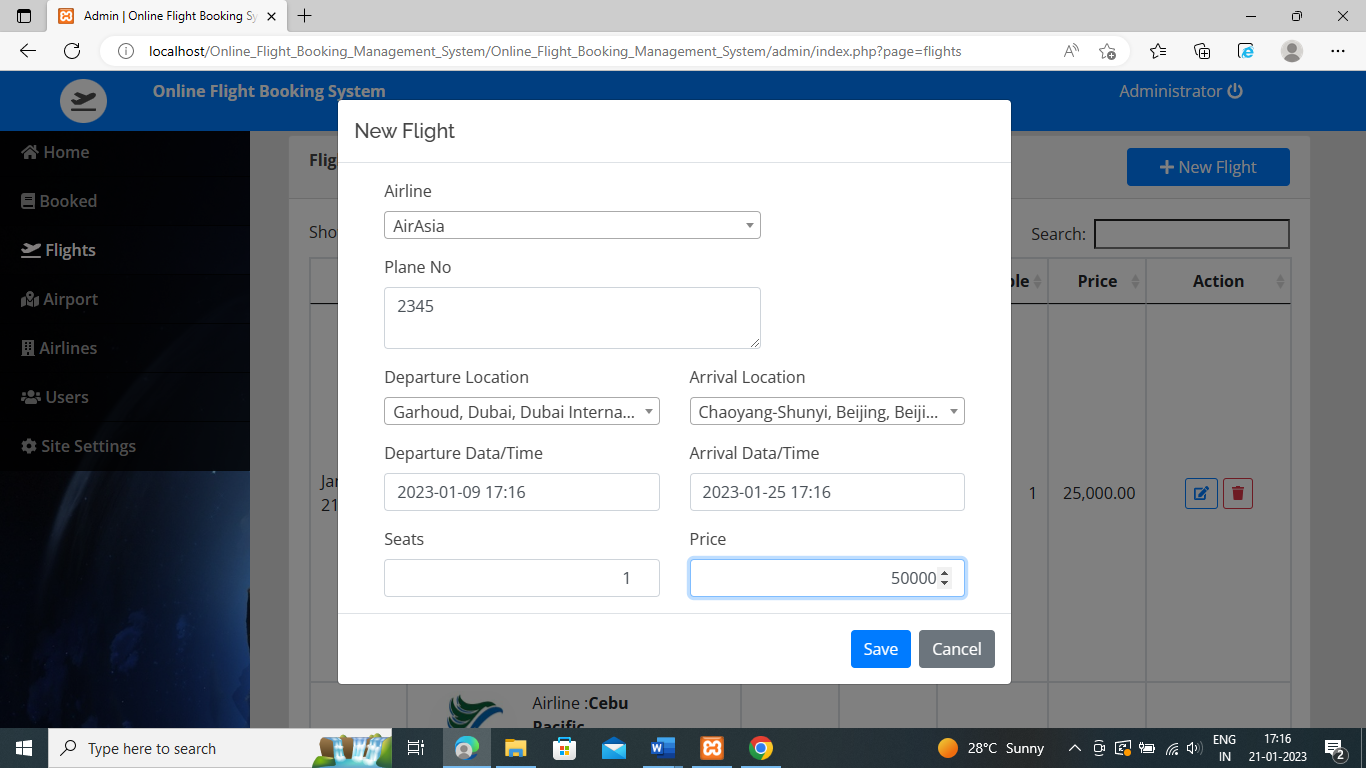


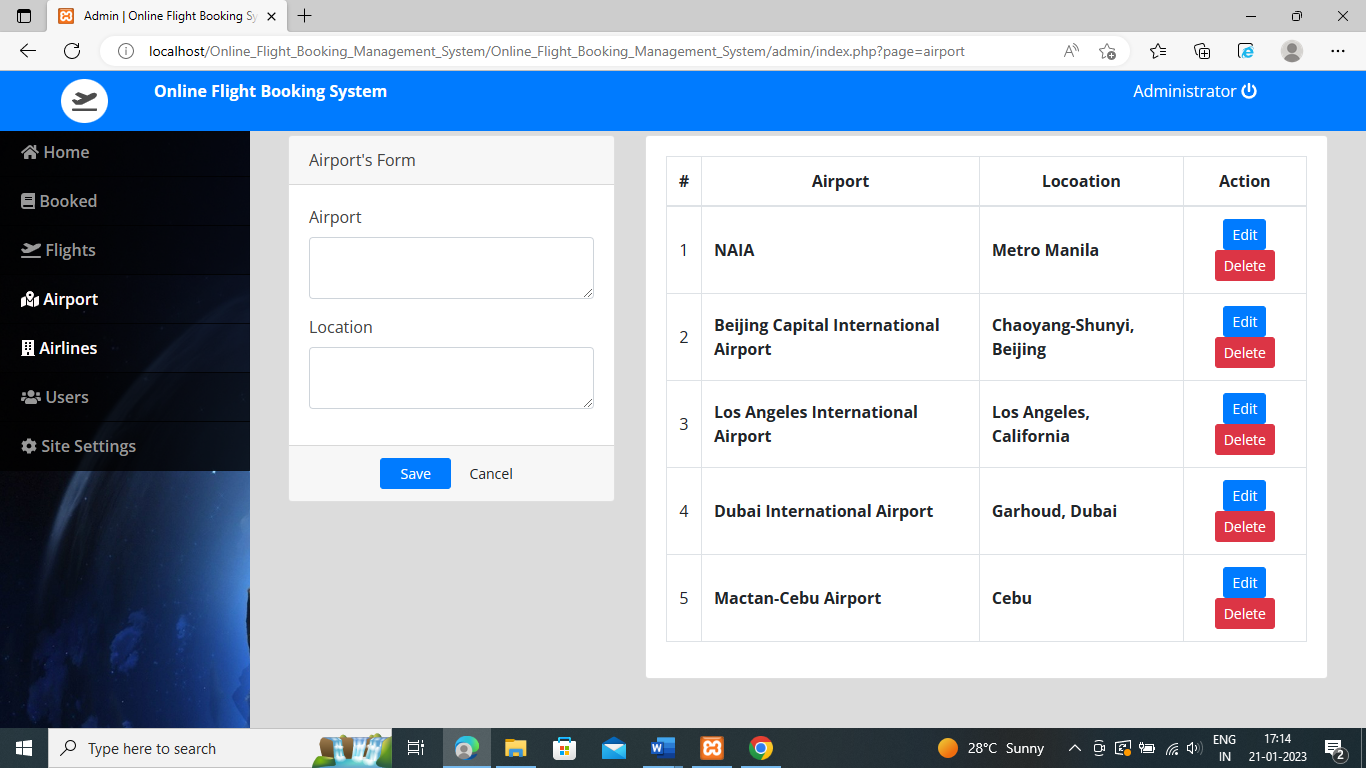


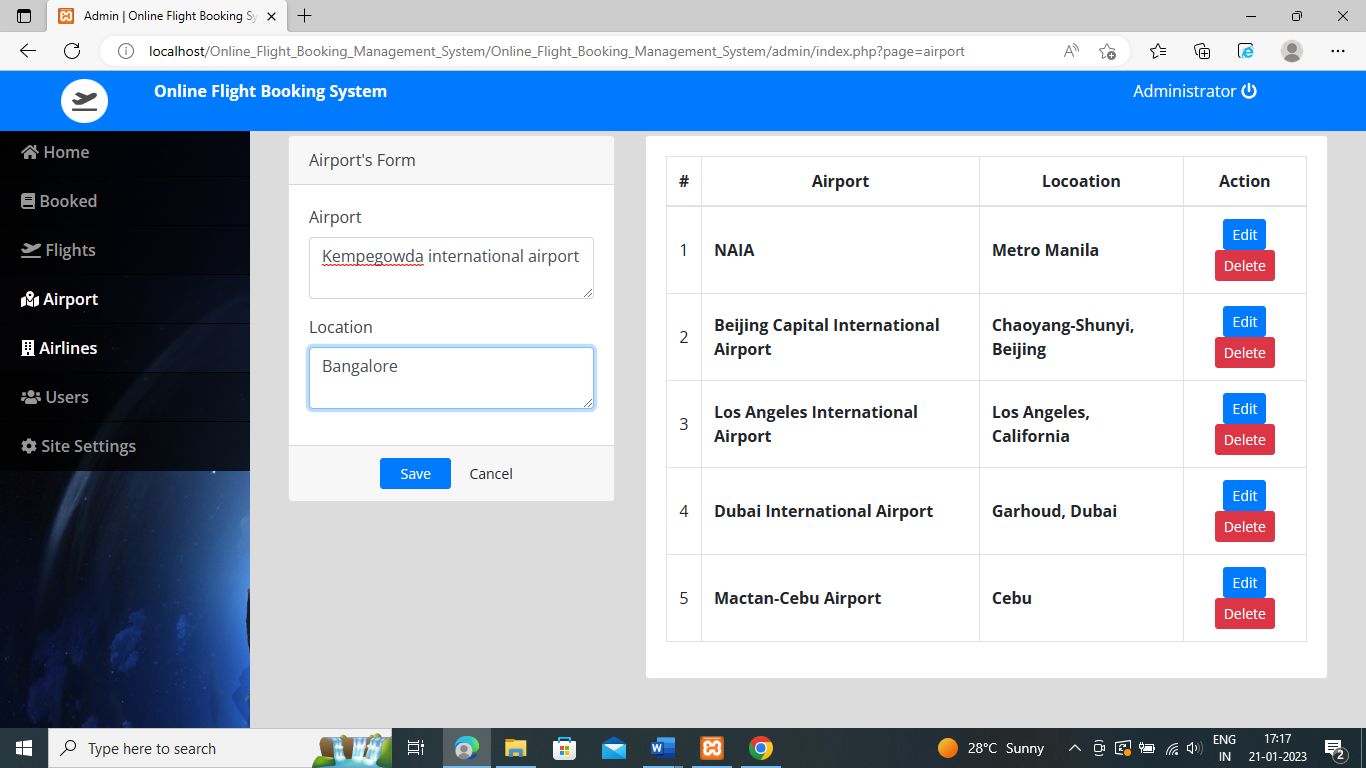


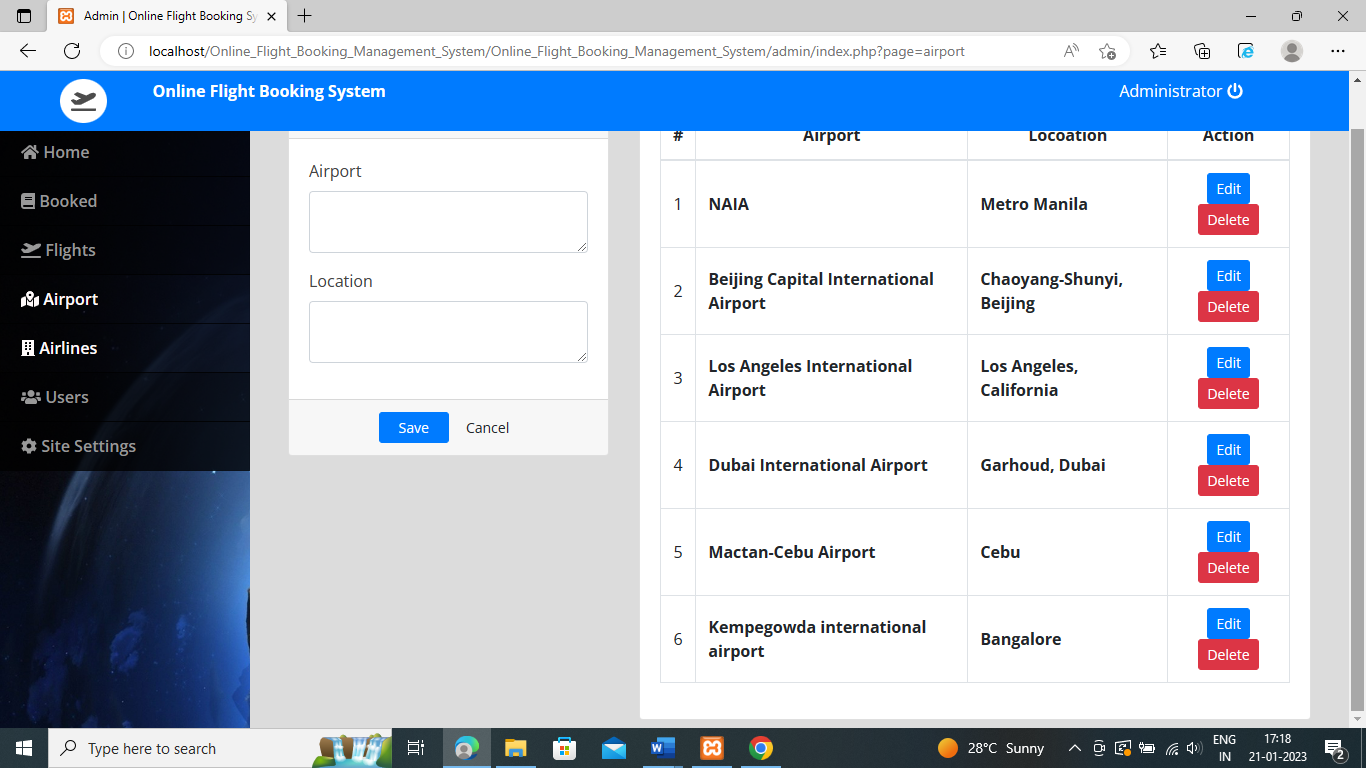


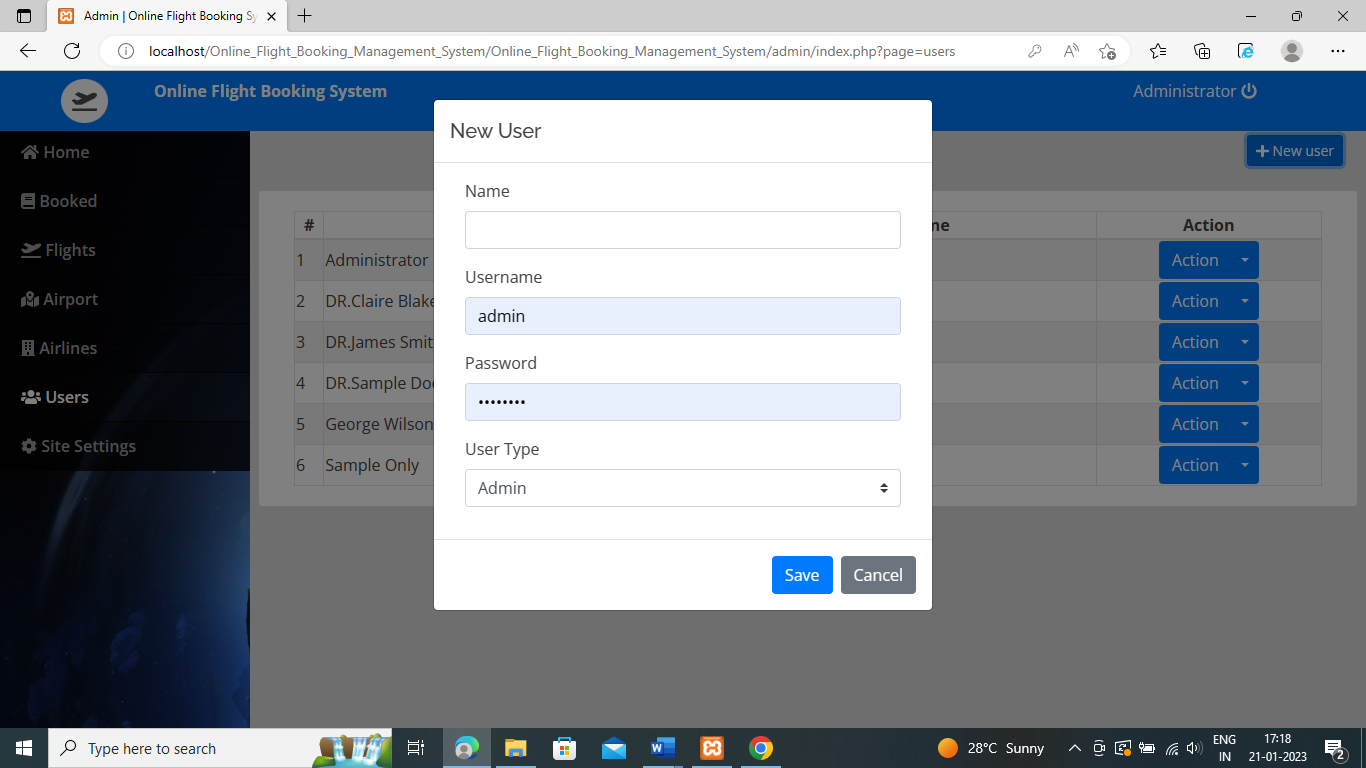


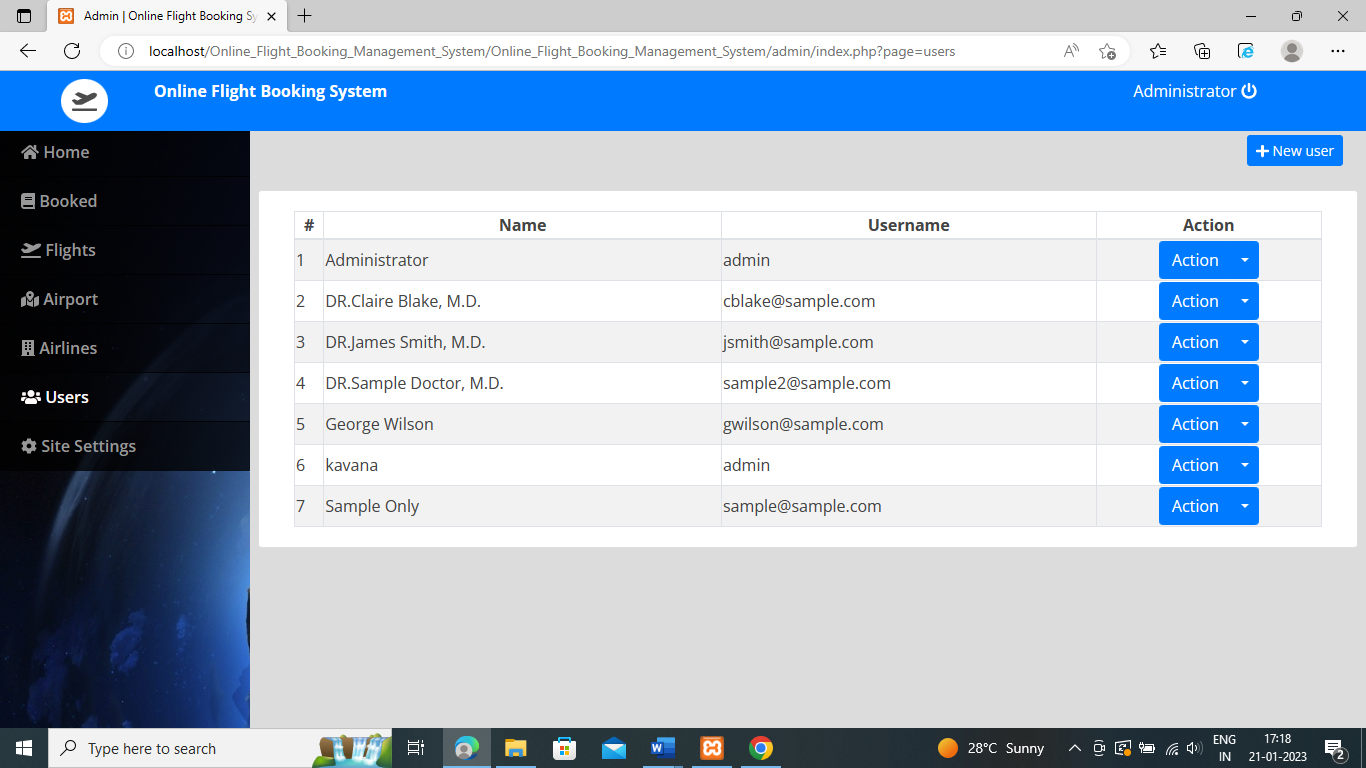


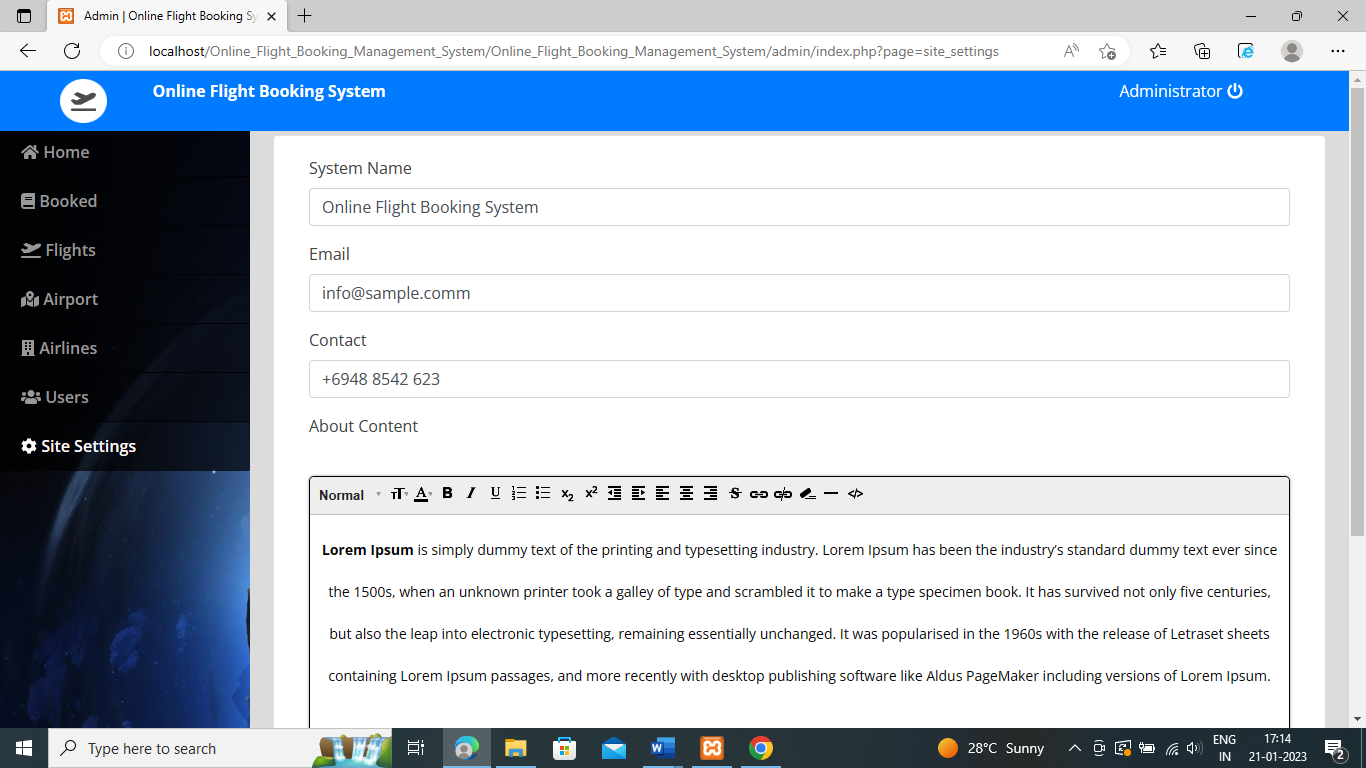


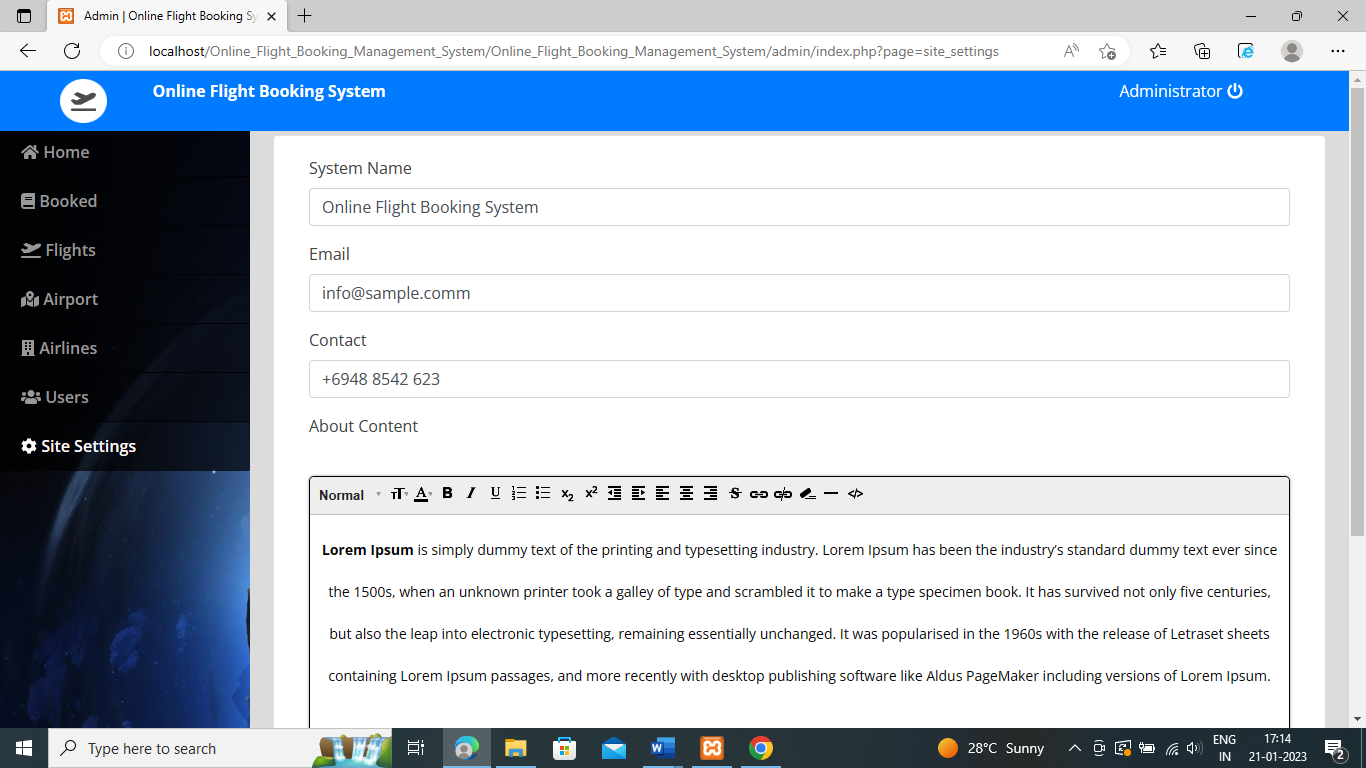












**CONCLUSION**

Online Flight Booking system has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not not only programming in Adobe Dreamweaver web-based application and no some extent Windows Application and SQL Server, but also about all handling procedure related with"Alliances". It also proves knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently. The purpose of object-oriented programming is to have code that is structured and organized. For the most part, you'll find yourself in situations where you'll be reusing code. Instead of having to recreate that same logic for each project, you can have a class that does most of the work for you.

Another reason for having such strict is because you may up share your class. For example, let's say you had a database class. A lot of people connect to databases. Some people may be looking for code that can help them connect to a database easier and run certain functions. By creating your database connector as a class, you help prevent collisions in their own program because all code wrapped in a class is unique to the class only.

**REFERENCES**

[1] Software Requirement Specifications from Internet.

[2] Software Requirement Specifications, Airline Reservation System IEEE (Institute of Electrical and Electronics Engineers)

[3] Specifications, [www.slideshare.com](http://www.slideshare.com)Guide to Software Requirement

[4] www.google.com

[5] www.wikipedia.com

[6] www.project-airline-system-basics.com

[7] HTML for the World Wide Web with XIITML and CSS Guide, Fifth Edition

[8] IEEE Computer Society ,1998