130102T –

130113D – W.W.A De Silva

130

130

Warunika

**Contents**

1. Introduction and Background

1.1 Problem Scenario

1.2 Purpose

1.3 Background

1.4 Essay Map

2. Design of the System

2.1 Requirements

2.2 ER Diagram

2.3 Database Schema

2.4 Design Decisions

3. Implementation

3.1 Frameworks and Standards

3.2 SQL Injection and Prepared Statements

3.3 Triggers

3.4 Transactions

3.5 Indexes

3.6 Integrity Constraints

3.7 Users

3.8 Project Management

4. Deployment

5. Functionality

6. Discussion

7. References

**1. Introduction and Background**

**1.1 Problem Scenario**

The objective of the project is to design a database application for MoraSpirit system. Mora spirit requires a database system for managing their sports equipment. The system also needs to manage the student details who are involved in sports within the university. In addition it should be capable of managing the supplies of sports equipment to the students, managing the student resource allocation schedule and managing the practice schedule.

**1.2 Purpose**

The purpose of this report is to describe the work flow of the project, how the difficulties were overcome and to identify any further improvements that can be implemented with regard to the system.

**1.3 Background**

Since this application is to be used by university students and the admin, it should be able to comply with various tasks. For example verifying incorrect records entered throughout the process, validating the data that are entered into the forms in the user interface and updating the related data with new additions. Many of these kinds of functions are handled by the database system itself while some are required to be implemented by the developers. And also the database is designed in such a way to meet these requirements.

**1.4 Essay Map**

As the first step of project, a requirement analysis was done. During that step possible requirements were identified and acceptance criteria was decided, which helped to identify the required functionalities and features of the database and the application. According to these decisions ER diagram and a relational schema were designed.

The database management system used in this project is MySQL, the world’s most used open source relational database management system. It runs as a server allowing multiple user access to several databases. The reasons to choose MySQL over other available options were; scalability and flexibility, high performance, robust transactional support, high availability, strong data protection and most importantly simplicity. PHP was used to write the database application. The framework that we used is Symfony. Adhering to an MVC pattern and strict object oriented language Symfony framework allows creating more testable code which can be reused in future and accordingly saves the developers’ time.

**2. Design of the System**

**2.1 Requirements specification**

|  |  |
| --- | --- |
| User Story | Acceptance criteria |
| Users with student privileges should be able to login to the system by proving their access credentials. | * The username must be university ID and password is set a default value so that users can change it later. |
| User should be able to enter their details and edit them. | * The user should be able to enter their own details to the system and edit them with valid data. * System should return already existing details of user and save the edited details. |
| Users with student privileges should be able to make reservation from their view of equipment details | * The user should be able to reserve an equipment that belongs to a certain sport. * System should update the status of the equipment as reserved and after lending the item state should be changed to lend. |
| Users should be able to view sport and equipment details, events and achievements. | * The system should display details about sports, equipment, events and achievements. |
| Users with administrative privileges should be able to login to the system by proving their access credentials so that they can view and update system information. | * There can be more than one user with the administrative privileges. * The username has to be unique. |
| Users with administrative privileges should be able to edit the practice schedule, add achievements and event details. | * The system should provide facilities for admin to edit the practice schedule, add achievements and event details. |

**2.2 ER Design**

**2.3 Database Schema**

**2.4 Important Design Decisions**

Designing the ER model from the requirements and converting it to the database schema deals with several design decisions considering redundancy in relations and making sure the application can support the intended functionality of the system.

2.4.1 Removing Redundant Relations

When converting our ER design to the database schema it may contain redundant relations where same details are repeated. For example consider the relationship between the flight and the fare. When converting this relationship to database schema we get three tables flight, flight fare and fare.

* flight (flight\_id,airline\_name,fly\_mon, fly\_tue , fly\_wed , fly\_thu , fly\_fri ,fly\_sun )
* flight\_fare (flight\_id,type)
* fare (flight\_id, type, max\_weight, amount)

Considering flight\_fare and fare relations they both have the primary keys flight\_id and type and flight\_fare relation only contains those two attributes. Hence the relation flight\_fare is not necessarily needed as the information stored in that relation is also stored in the fare relation. So the flight\_fare is a redundant relation and we can remove it from the database.

Similarly in the relations flight\_specific\_flight and specific\_flight , we can remove the redundant relation flight\_specific\_flight.

2.4.2 Normalization

2.4.2.1 First Normal Form

All the tables should be in the first normal form therefore all the domains should be atomic. In the flight relation flight\_id contains two parts. airline\_id and flight\_number. This is not an atomic domain. So we have to divide these two attributes to airline\_id and flight\_no.

2.4.2.2 Boyce Codd Normal Form

Taken all the functional dependencies of F+ such that α 🡪 β, α and β are subsets of F+, the relation schema is in BCNF if one of the following holds:

* α 🡪 β is a trivial functional dependency
* α is a super key for R

Since we developed our database schema from the ER design where we created entities with certain attributes such that the functional dependencies occur only from the primary key of a relation, the database schema is already in the BCNF form. For example consider the relation flight:

flight (flight\_id, airline,fly\_mon, fly\_tue, fly\_wed, fly\_thu, fly\_fri,fly\_sat, fly\_sun)

In this relation the only functional dependency is from flight\_id to all the other attributes. Since the flight\_id is the primary key it satisfies the BCNF.

**3. Implementation**

**3.1 Frameworks and standards**

We decided to use a framework because it is better and faster. Better, because a framework provides you with the certainty that you are developing an application that is in full compliance with the business rules, that is structured, and that is both maintainable and upgradable. Faster, because it allows developers to save time by re-using generic modules in order to focus on other areas. Without, however, ever being tied to the framework itself.

The framework used in implementing this project is Symfony, a PHP framework which uses clean MVC conventions which guided us to the things need to be done through the development. Using Symfony framework helped us in many aspects. It helped to develop faster and better software than with flat PHP and it rescued us from mundane tasks and let us take back control of the code. Since this has many in-built tools such as input validation, form tampering protection, XSS prevention, and authentication which made our lives easier.

Moreover symfony uses code generation and scaffolding for rapidly built prototypes. Scaffolding is a technique which allows the developer to obtain a basic CRUD database up and going. This creates a loose structure with full flexibility. At the initial stages the database schemas are subjected to changes with establishing relations and eliminating the existing relations. This scaffold supports these operations to be done on the database.

**3.2 SQL injection and prepared statements**

At this point of the implementation process we have an application up and running. Then the program execution is written from scratch. Prepared statements are usually used to,

1. Save on query parsing
2. Save on data conversion and copying
3. Avoid SQL Injection
4. Save memory on handling blobs

In our case we tend to use prepared statements mainly to avoid SQL Injection and thereby to secure the data that is being passed through the forms in the application. SQL injection is a technique where malicious users can inject SQL commands into an SQL statement via user input. Injected SQL commands can alter SQL statements and compromise the security of a web application. By using prepared statement we the query and the data are sent to the SQL server separately which removes the root of the SQL injection problem, **mixing of the code and the data.** Thus, the passed data does not interfere with program code and alter it.

**3.3 Triggers**

Whenever a record is changed after a CRUD operation, triggers are used to monitor, control and manage a group of tables. In the equipment lending system we have implemented two triggers. First one is implemented such that whenever an equipment is lent or reserved, that particular equipment’s status will be updated to ‘lent’ or ‘reserved’ respectively. Similarly we have implemented another trigger such that once an equipment is handed over, the status of that particular equipment will be updated to ‘available’. Triggers have reduced the work of the developer and have made the code simpler and shorter.

**3.4 Transactions**

When a customer makes a reservation the procedure includes updating several relations in the schema. If a failure happens before all the tables are updated the database may become inconsistent. So we have to include a transaction for that procedure. if the transaction was not successful after checking the status of each step the database is called to roll back to the previous state //do we have this????

**3.5 Indexes And Views**

Using indexes improve the performance in the select operation. But when considering update operation it adds an unnecessary work amount. Adding indexes to the tables which are not updates frequently and used in select operations frequently increase the performance. In our scenario the coach table is indexed by the sport\_id.

Views are used when all the users need not to see the entire logical model. In our scenario we have used one view such that student users can view only the name, faculty and department of the student players not the entire details. Another view has been used to make the SQL query simpler. A view has been created only with the needed data and in the query to fetch details we have used simply “SELECT \* FROM <view\_name>...”

**3.6 Integrity Constraints**

Certain attributes in the database should not be null values at any instance. To prevent that when we create the table we add the constraint not null. The attributes added as primary keys or unique keys can’t be duplicated anywhere in that table. Foreign keys are added to preserve relationship between tables.

**3.7 Users**

This system has three types of users. They are the students, coaches and administrators. They are allowed different levels of access to the database with authorization. Once a student is added to the system, his/her details are recorded in the database and an account is created for the student in the system with his/her index as the username. It is same with the coaches. Once a new coach started working he will also get an account in the system. System users are allowed the CRUD operations and only the administrators can drop and create tables with all global privileges.

**3.8 Project Management**

Since this is a group project during the implementation phase we used Git online source code management system for make it easier to for all the team members to contribute for the project in parallel. Members can commit the changes they did for the code anytime. So any member can know what the other members of the team have done.

**4. Deployment**

In deploying the ticketing system WAMP, MySQL 5.5, Apache 2:2.72 and PHP 5.3.5 are used.

WAMP is an acronym formed from the initials of Microsoft Windows and principal components of the package Apache, MySQL and one of PHP, Perl or Python.

MySQL 5.5, open source relational database management is used with improved scalability, performance, usability, recovery performance and availability. Apache 2:2.72 is the web server used. PHP 5.3.5, the sever-side scripting language is used to develop the web application.

**5. Functionality**

The online MoraSpirit system includes a web based front end and a database. The front end is used by two types of users; customers who want to reserve their tickets via the system and users with administrative privileges who can edit and update the database.

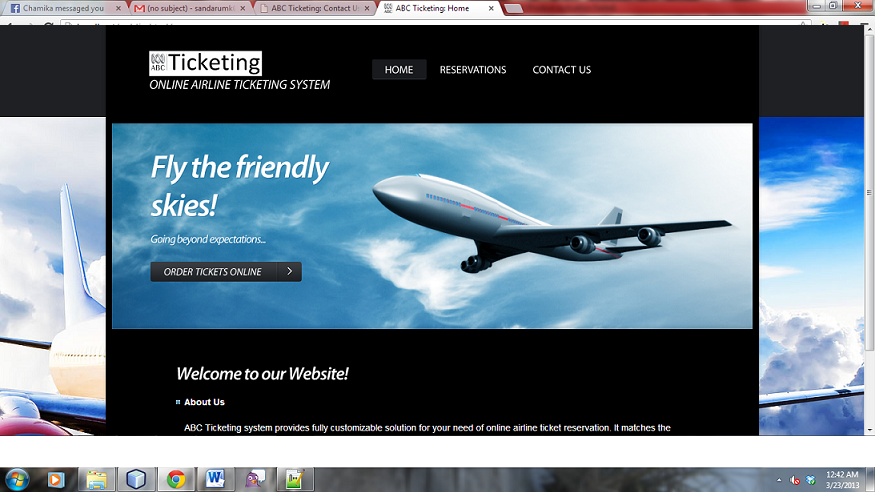


Figure 5.1 - Home page of the Ticketing System

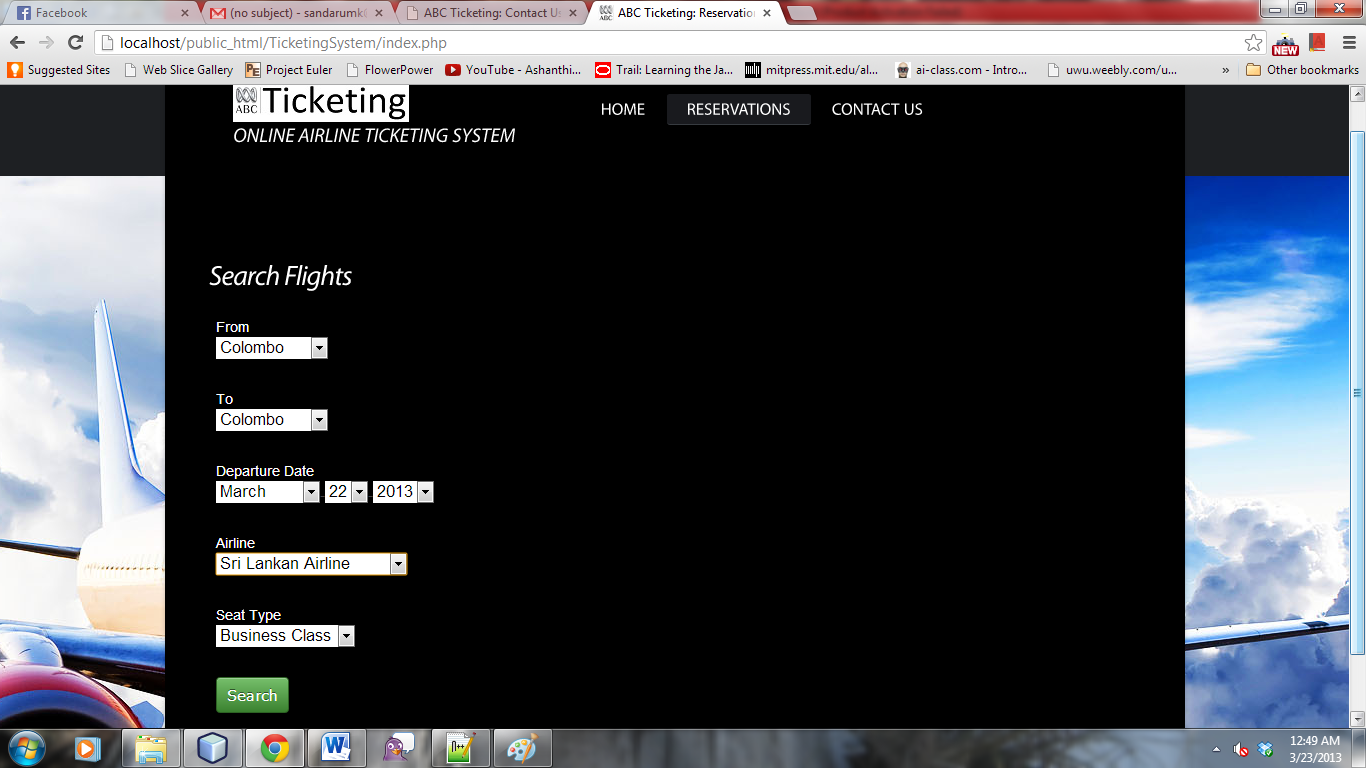
When a client comes to our website he can enter the departure airport, departure date and time, arrival airport, airline of his preference, and ticket type ( whether business class or economy class) and search for a flight matching his details. The system provides the search results containing the flight code, departure and arrival airports, departure and arrival dates and time.

Figure 5.2 - Search page

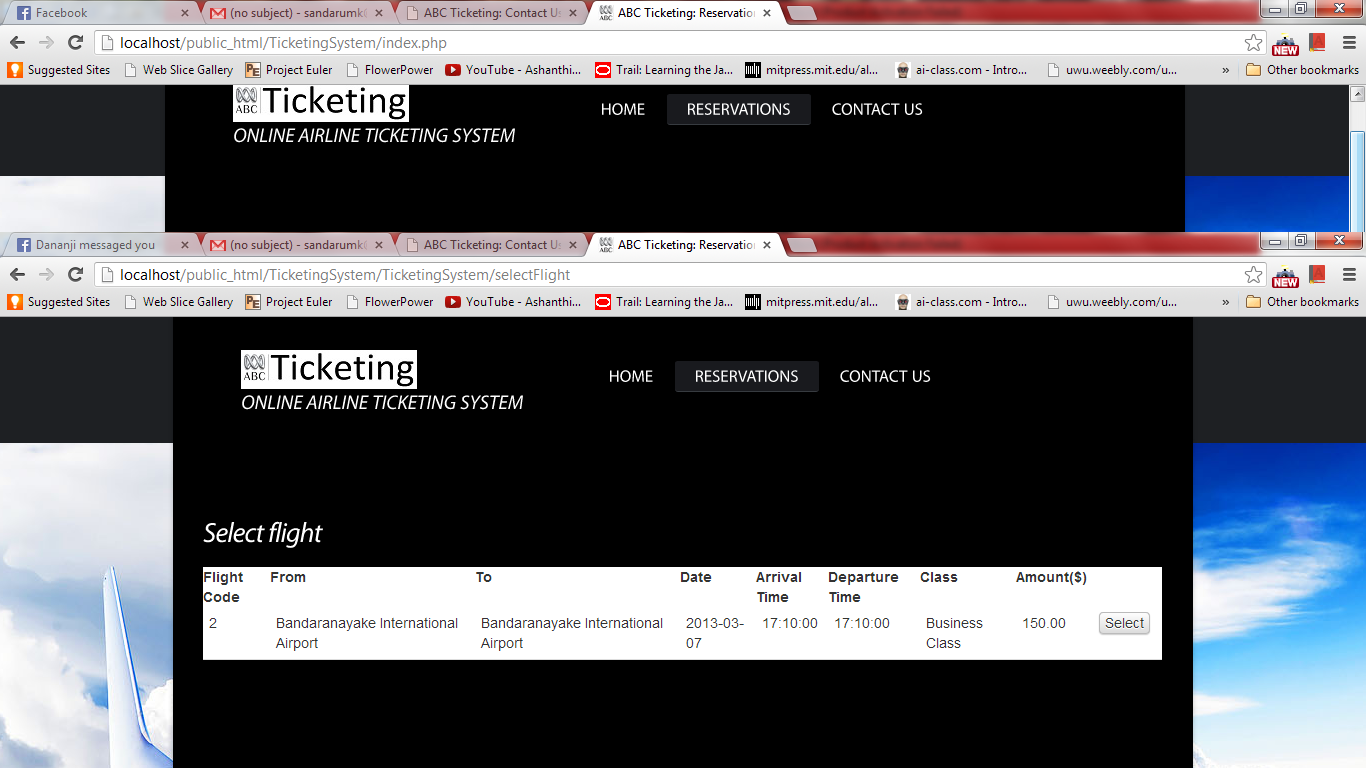


Figure 5.3- Search Result Page

Then the customer can select a flight and he can make a reservation and pay for the reservation using his credit card. If the transaction is successful he gets a summary of his reservation and he can print it and give it to the airport counter to receive the boarding pass.

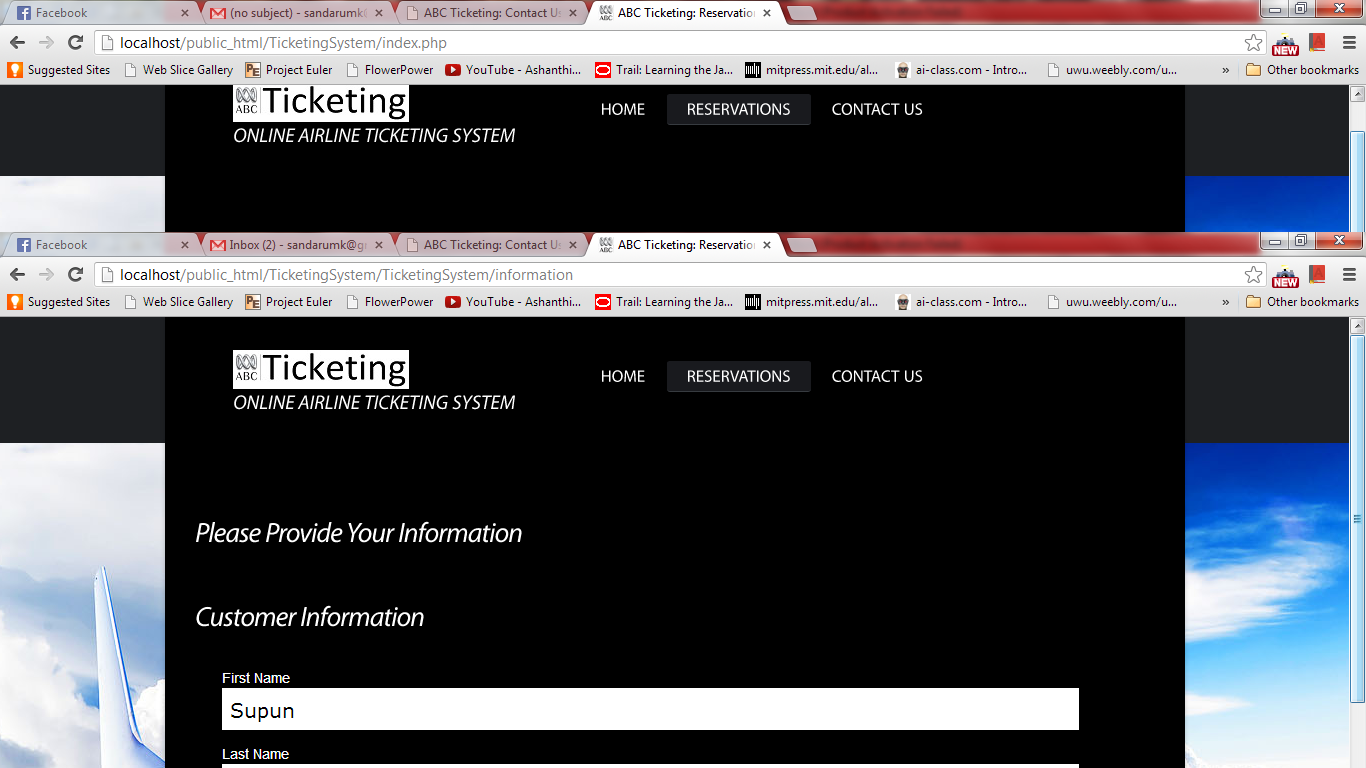


Figure 5.3 – Reservation Page

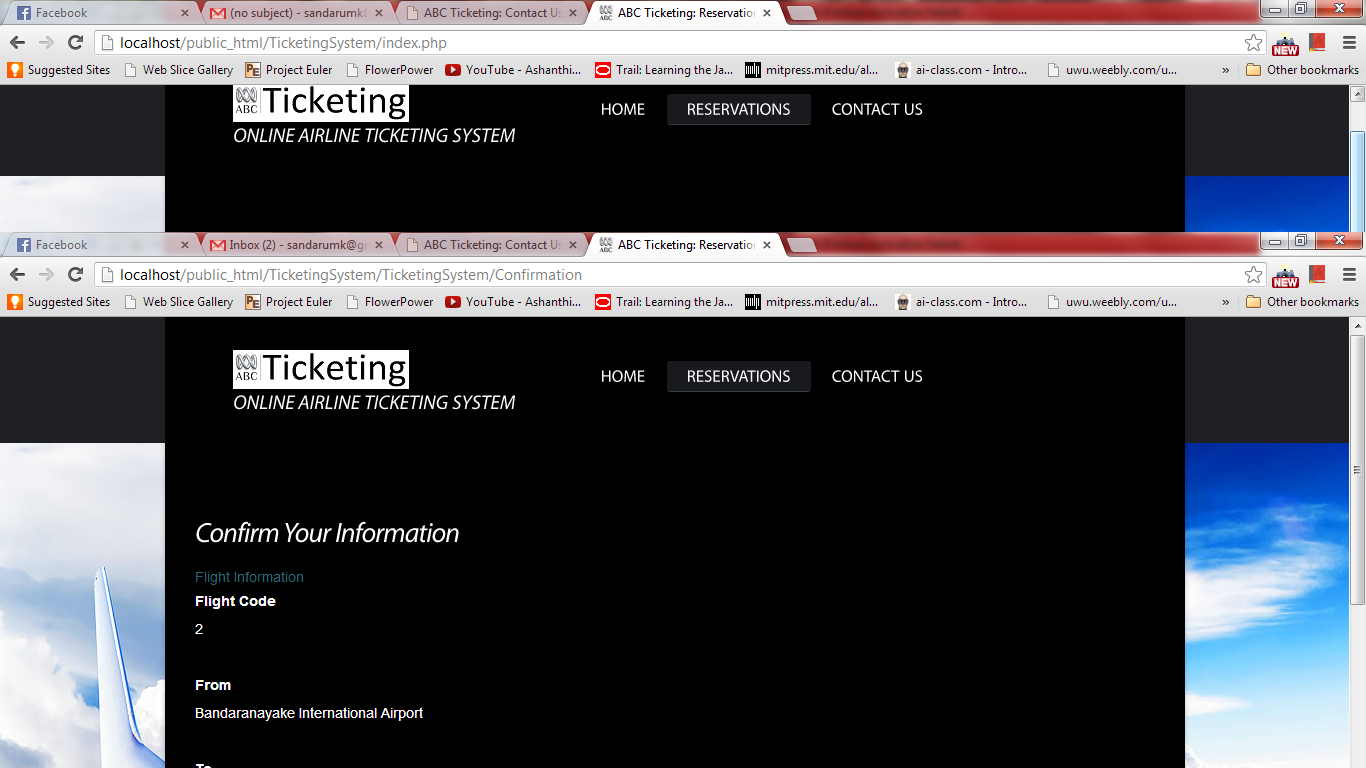


Figure 5.4 – Confirm the information

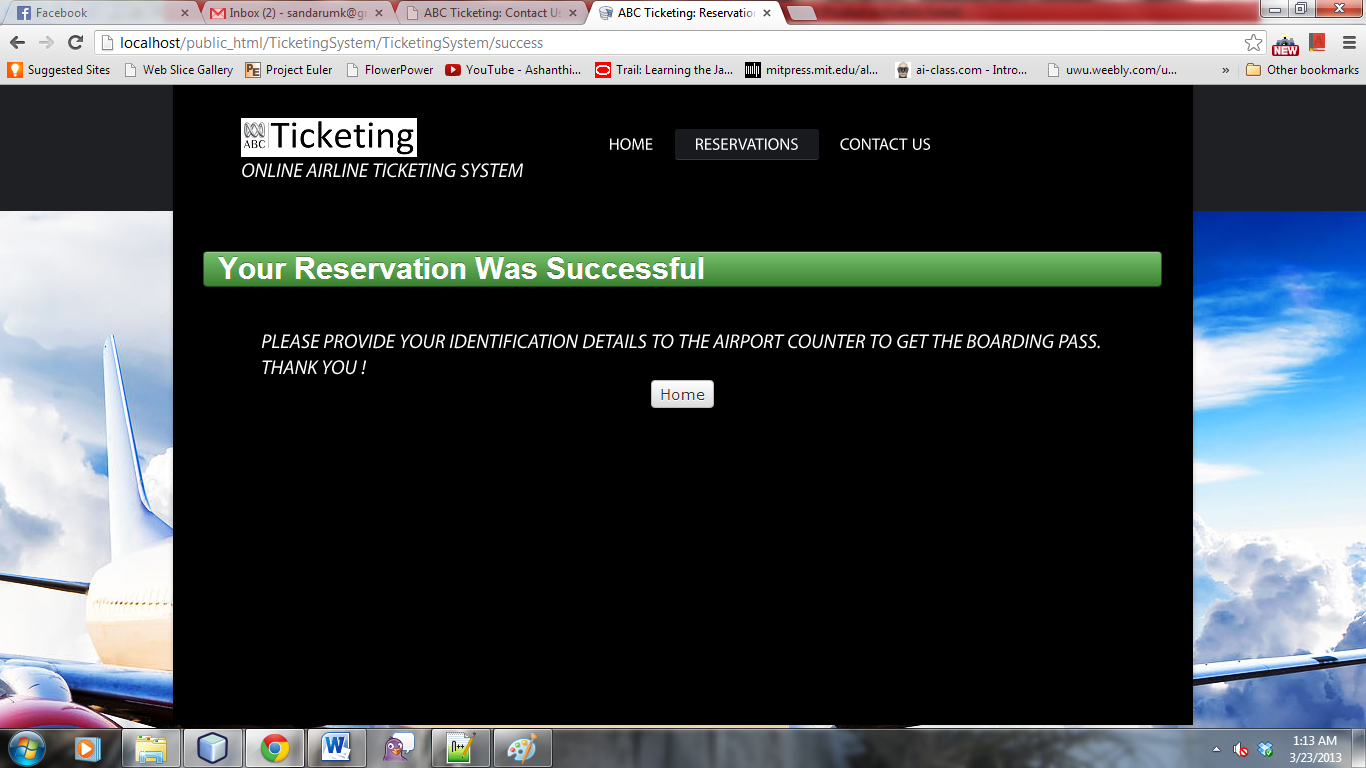


Figure 5.5- Successful ReservationUsers with administrative privileges can login to this system using their credentials. There two types of users apart from customers; administrators and system users. System users are allowed to do the CRUD operations. They can edit, update and delete data. The administrators can drop and create tables with all other global privileges. The level of privileges of system users can be altered by administrators if needed.

6. Discussion

We can improve the system further by adding a search option for round trip tickets. Using the current system the user has to search it twice to get a ticket to and from the same airports. We can develop the system to select an option in the search and adding a time period, a combination of flights to go to a certain destination and come back may appear in the search result.

The current system search result only includes the direct flights from the departure and arrival airport. We can add an option to search flight combinations from the departure and to arrival airport through more than one direct flight. If the user want to go straight from Colombo to London and if there are no direct flights to Colombo and London and there are two direct flights from Colombo to Dubai and Dubai to London the system should be able to give that combination of flights to the user in the search result.