

**GROUP ASSIGNMENT**

**AAPP009-4-2-WDT**

**WEB DEVELOPMENT**

**UCDF1905ICT(SE)**

**HAND OUT DATE: 10 AUGUST 2020**

**HAND IN DATE: 25 OCTOBER 2020**

**WEIGHTAGE:** **60%**

**INSTRUCTIONS TO CANDIDATES:**

**1 Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).**

**2 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**

**3 Cases of plagiarism will be penalised.**

**4 The assignment should be bound in an appropriate style (comb bound or stapled).**

**5 Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

**Group Members:**

**LAI RONG XIANG TP: TP056196**

**HASSAN ABBAS TP: TP055184**

**MUHAMMAD FUAD BIN ABDULLAH (L) TP: TP055684**

Table of Contents

[1.0 Gantt Chart 4](#_Toc54518299)

[2.0 Introduction and Objectives 8](#_Toc54518300)

[2.1 Introduction 8](#_Toc54518301)

[2.2 Objective 8](#_Toc54518302)

[3.0 System Design 9](#_Toc54518303)

[3.1 Wireframe 9](#_Toc54518304)

[3.1.1 User Registration 9](#_Toc54518305)

[3.1.2 User Login 10](#_Toc54518306)

[3.1.3 Flight Search 11](#_Toc54518307)

[3.1.4 Search Result 12](#_Toc54518308)

[3.1.5 Booking Form 13](#_Toc54518309)

[3.1.6 Payment Selection 14](#_Toc54518310)

[3.1.7 Booking Review 15](#_Toc54518311)

[3.1.8 Airport Manager Add Flight 16](#_Toc54518312)

[3.1.8 Administrator Add Airport Manager 17](#_Toc54518313)

[3.2 Navigational Structure 18](#_Toc54518314)

[3.3 Activity Diagram 19](#_Toc54518315)

[3.3.1 User Activity 19](#_Toc54518316)

[3.3.2 Airport Manager Activity 21](#_Toc54518317)

[3.3.3 Administrator Activity 23](#_Toc54518318)

[3.4 Entity Relationship Diagram 25](#_Toc54518319)

[4.0 Implementation 26](#_Toc54518320)

[4.1 Create 26](#_Toc54518321)

[4.1.1 Booking a Flight 26](#_Toc54518322)

[4.1.2 Adding a Flight 27](#_Toc54518323)

[4.2 Read 28](#_Toc54518324)

[4.2.1 User Profile 28](#_Toc54518325)

[4.2.2 Flight Search Result 30](#_Toc54518326)

[4.3 Update 32](#_Toc54518327)

[4.3.1 User Password 32](#_Toc54518328)

[4.3.2 User Profile Picture 33](#_Toc54518329)

[4.4 Delete 34](#_Toc54518330)

[4.4.1 Booking History 34](#_Toc54518331)

[4.4.2 User Account 35](#_Toc54518332)

[4.5 Registration 36](#_Toc54518333)

[4.6 Login 38](#_Toc54518334)

[4.7 Self-Created CSS 40](#_Toc54518335)

[4.8 Self-Created JavaScript 41](#_Toc54518336)

[5.0 Main Section 42](#_Toc54518337)

[5.1 User Manual 42](#_Toc54518338)

[5.1.1 Create an Account 42](#_Toc54518339)

[5.1.2 Logging in 45](#_Toc54518340)

[5.1.3 Booking for a Flight 48](#_Toc54518341)

[5.1.4 Editing Profile 55](#_Toc54518342)

[5.1.5 Deleting Booking History 58](#_Toc54518343)

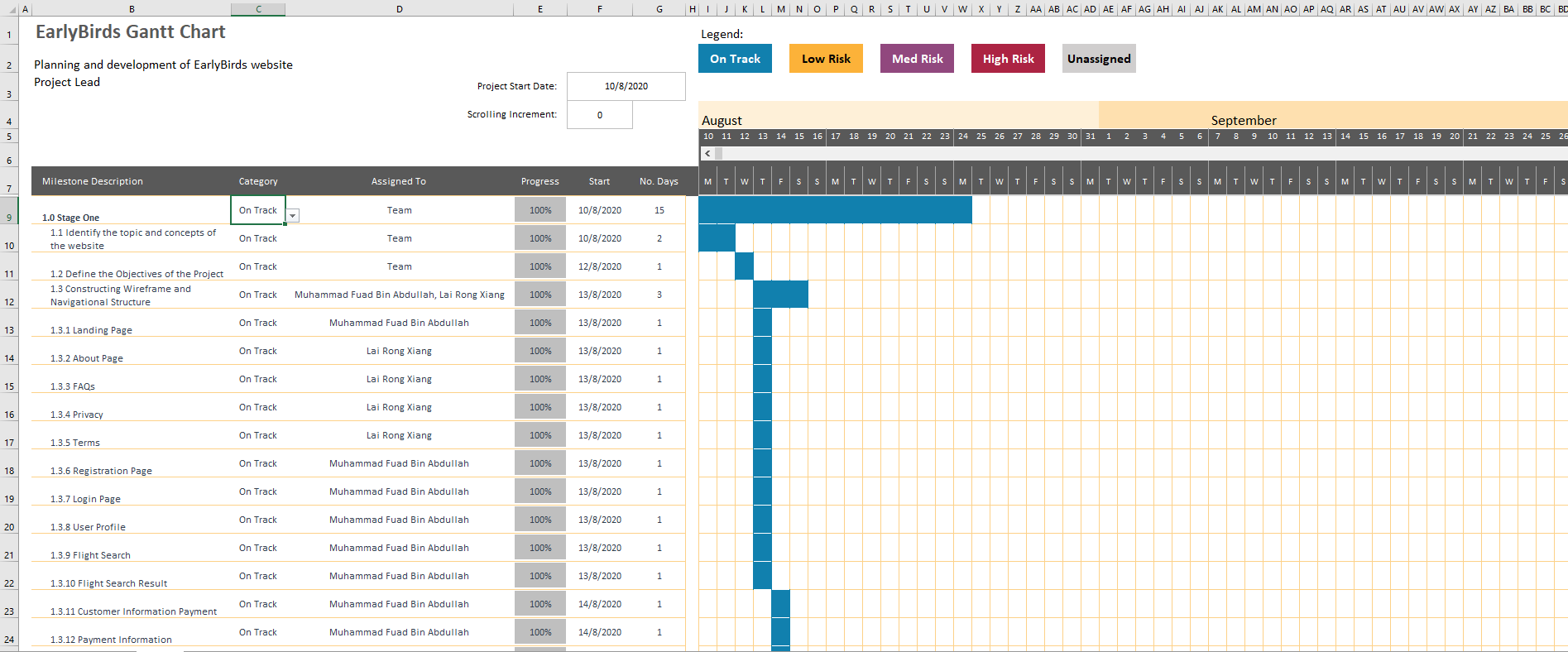
[6.0 References 60](#_Toc54518344)

[7.0 Appendices 61](#_Toc54518345)

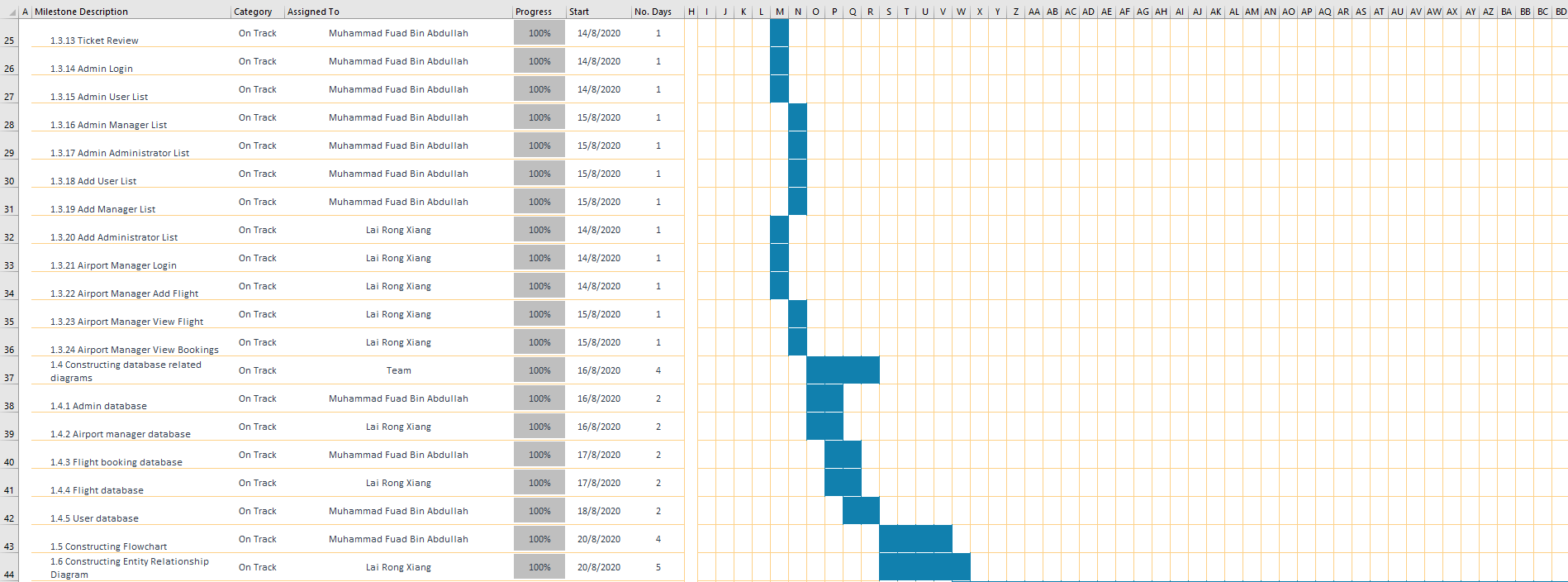
[7.1 Workload Matrix 61](#_Toc54518346)

[7.2 Peer-to-Peer Evaluation Form 63](#_Toc54518347)

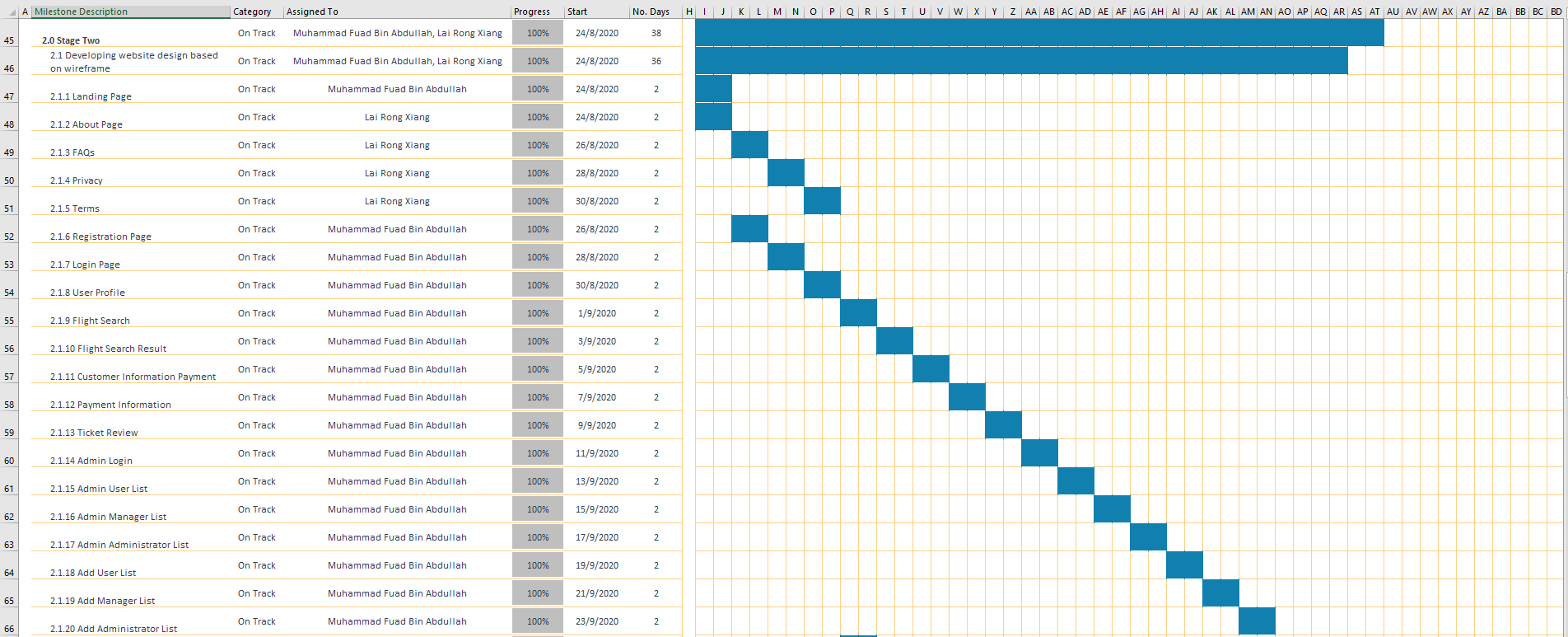
# Gantt Chart



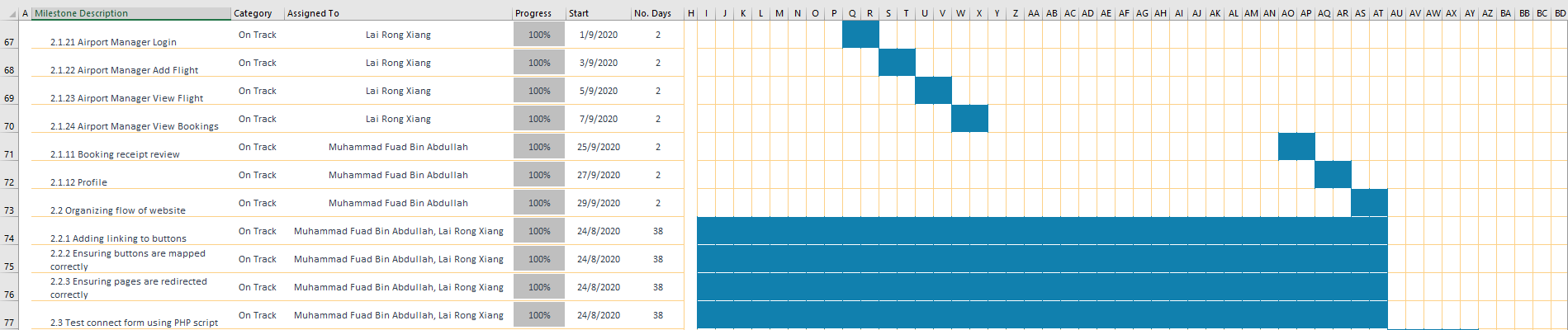
**Figure 1** shows the first half of stage 1 of EarlyBirds Gantt Chart



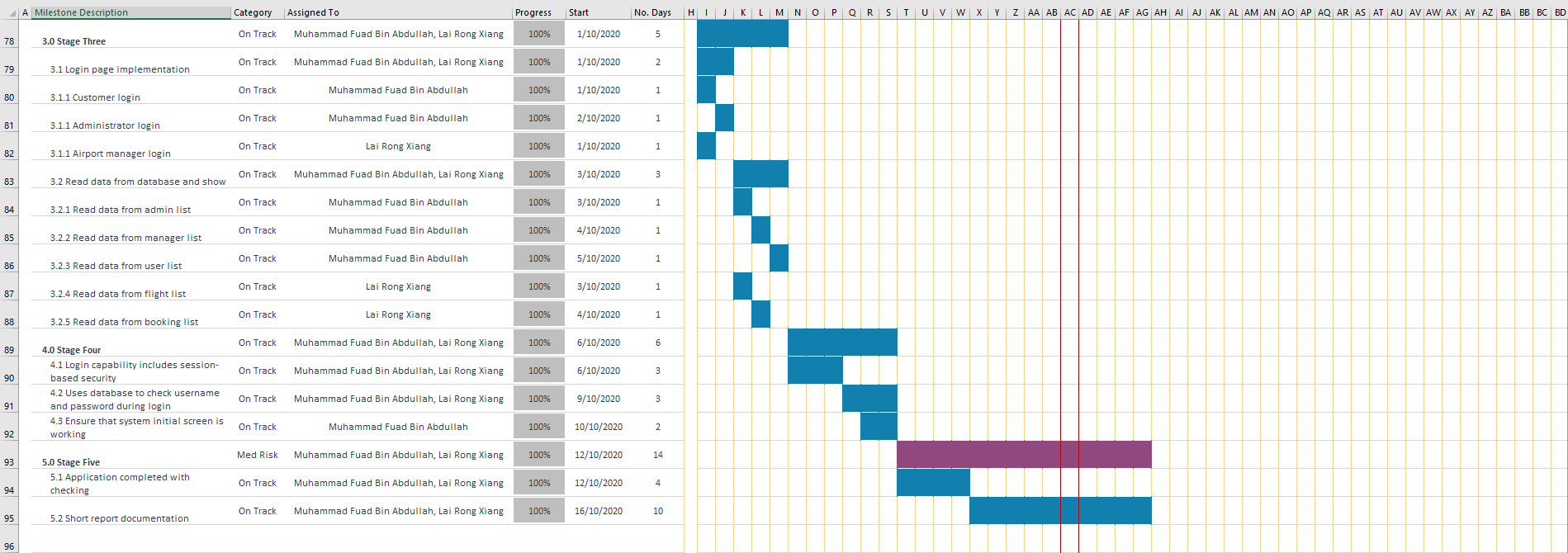
**Figure 2** shows the second half of stage 1 of EarlyBirds Gantt Chart

****

**Figure 3** shows the first half of stage 2 of EarlyBirds Gantt Chart



**Figure 4** shows the second half of stage 1 of EarlyBirds Gantt Chart



**Figure 5** shows stage 3, stage 4, and stage 5 of EarlyBirds Gantt Chart

# Introduction and Objectives

## 2.1 Introduction

The decision that has been made from the team is to develop a web application that assists with flight booking that can be done even if the flight happens on the same day. The website will be named “EarlyBirds”, which resembles the concept of the phrase “The early bird gets the worm”, where it focuses about higher success of flight booking as you are getting a quick reservation within a short period of time before departure of the flight. Rather than booking in few days, or few weeks in advance, customer can book a flight from various giant airlines within the same day too! The website has information filtered accordingly to show available flight time according to search results from users, especially with flights that will be ongoing within the same day, and booking information will also be stored in the user’s account for verification purposes upon arriving at the airport. The website usage is targeted to users who are in need of a flight with a short time due to business reasons, as well as users who are in need of a flight in a short time due to emergency reasons of arriving to a place immediately.

## 2.2 Objective

The objective of this project is to develop a website called “EarlyBirds”, which demonstrates the ability of webpage designing, where there is implementation usage of HTML, CSS, and Javascript to present visual elements in the webpage. The page will also be involving usage of PHP for the creation of dynamic pages, and the website will also be working with a database technology from the usage of MySQL, which helps with database functions such as the ability to create, read, update, and delete records for the database of the website. These databases will be used to store login credentials as users register from the website, which also involves validation purposes when logging in to the website. The database also serves as a method of storing data of flight booking from customers for record and verification purposes for customers as they arrive to the airport. EarlyBirds has a main objective in mind, which is to help making flight booking convenient for users, even if they are in a rush to get a flight booking within the same day. The website is also designed in a way which has a neat interface for users to be comfortable as they use the website.

# System Design

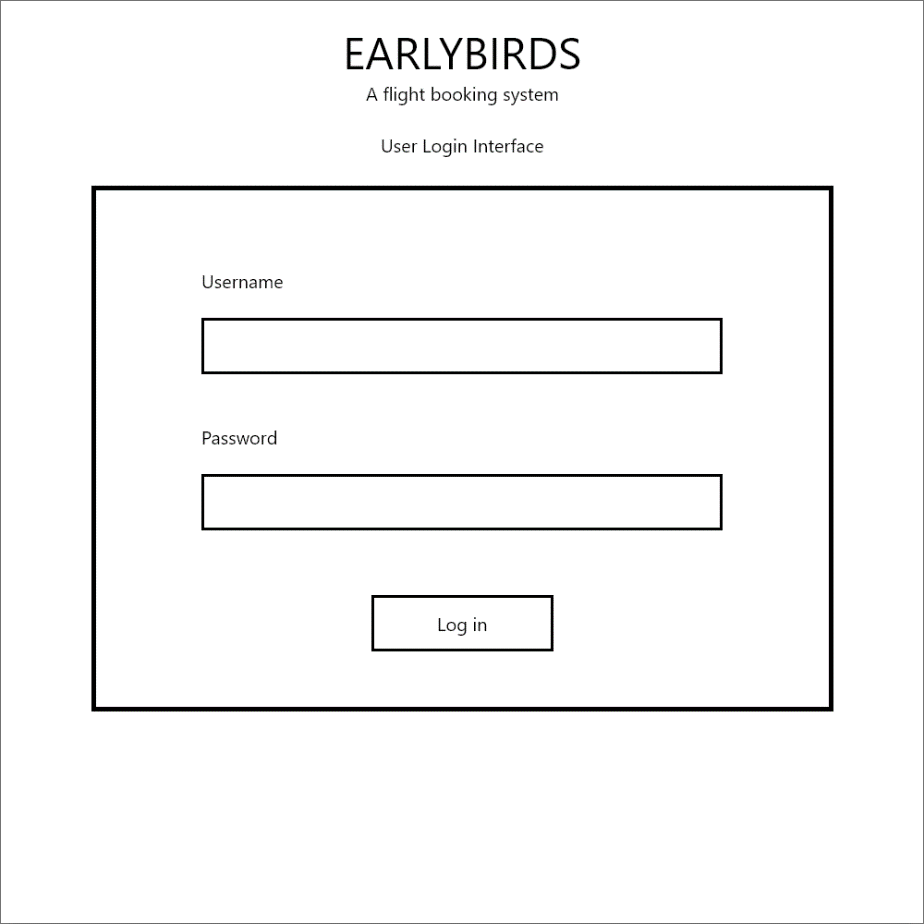
## 3.1 Wireframe

### 3.1.1 User Registration

**Figure 6** shows the proposed registration form for Earlybirds users

The registration wireframe displays several important elements that enable for user interaction to happen. In this case, the user can register by inserting their desired username and password into the prepared input fields. Besides that, the user is also required to provide their first and last name in order for Earlybirds to enhance user experience while using the site especially during booking process.

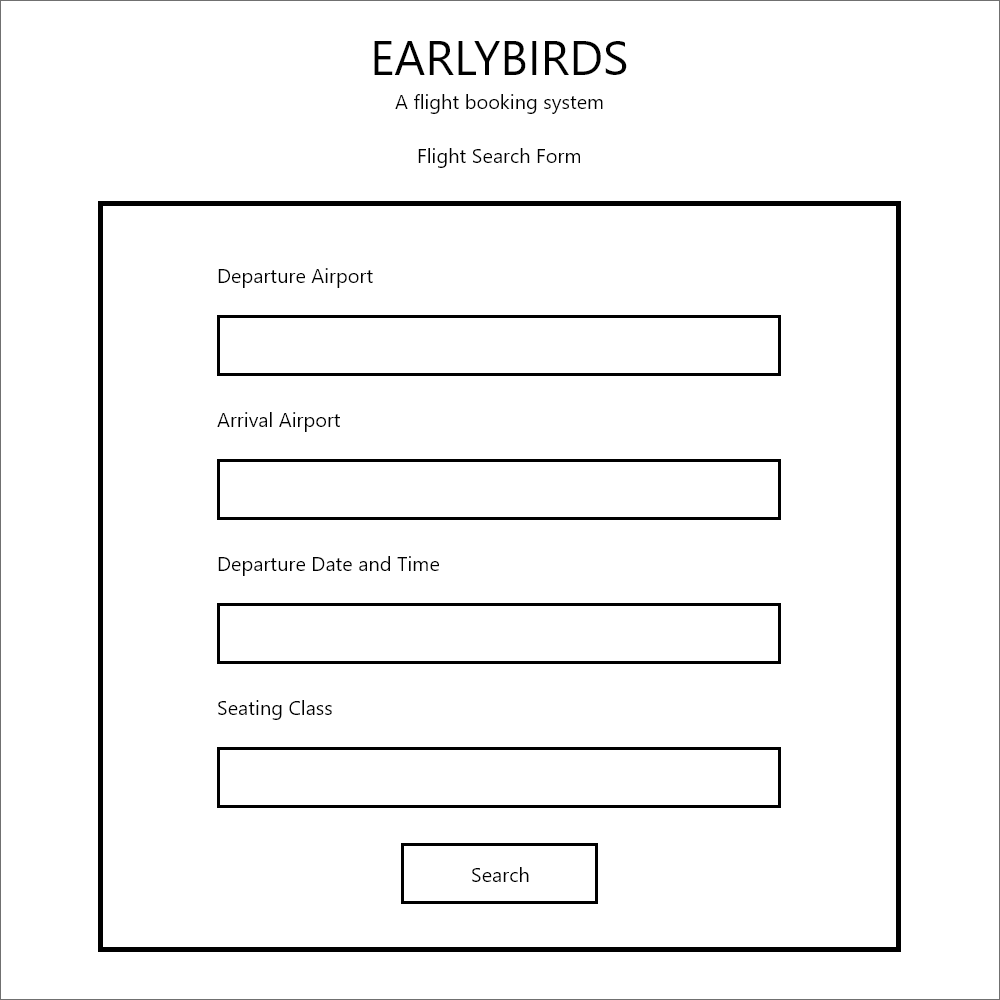
### 3.1.2 User Login



**Figure 7** is the main login page for Earlybirds registered users

The login wireframe shows the two most important input fields that require user insertion. The username and password input fields are prepared and will act as the container to the user inserted information during credential’s verification process. Only username and password registered with Earlybirds website can be used.

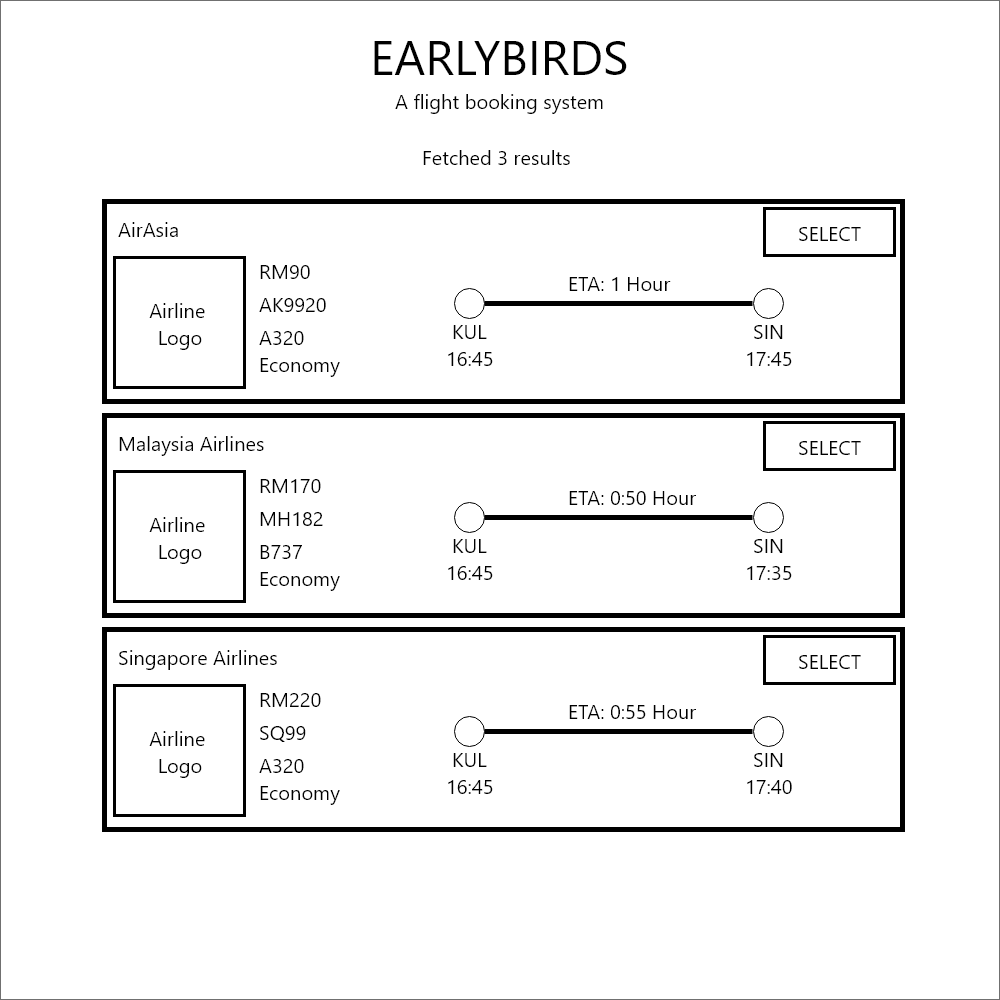
### 3.1.3 Flight Search

****

**Figure 8** is the search form used by registered Earlybirds users when looking for available flights

The search form wireframe details out four main inputs to be filled in by the user. The first and second input, Departure Airport and Arrival Airport respectively, specify current and destination location of the user. The third input is inserted by the user to inform the system of specific range of time of scheduled flights to be displayed. The last input would be to let the user choose for seating class while inside the aircraft. Search function will use the inputs to display results with the closest match to user’s constraints

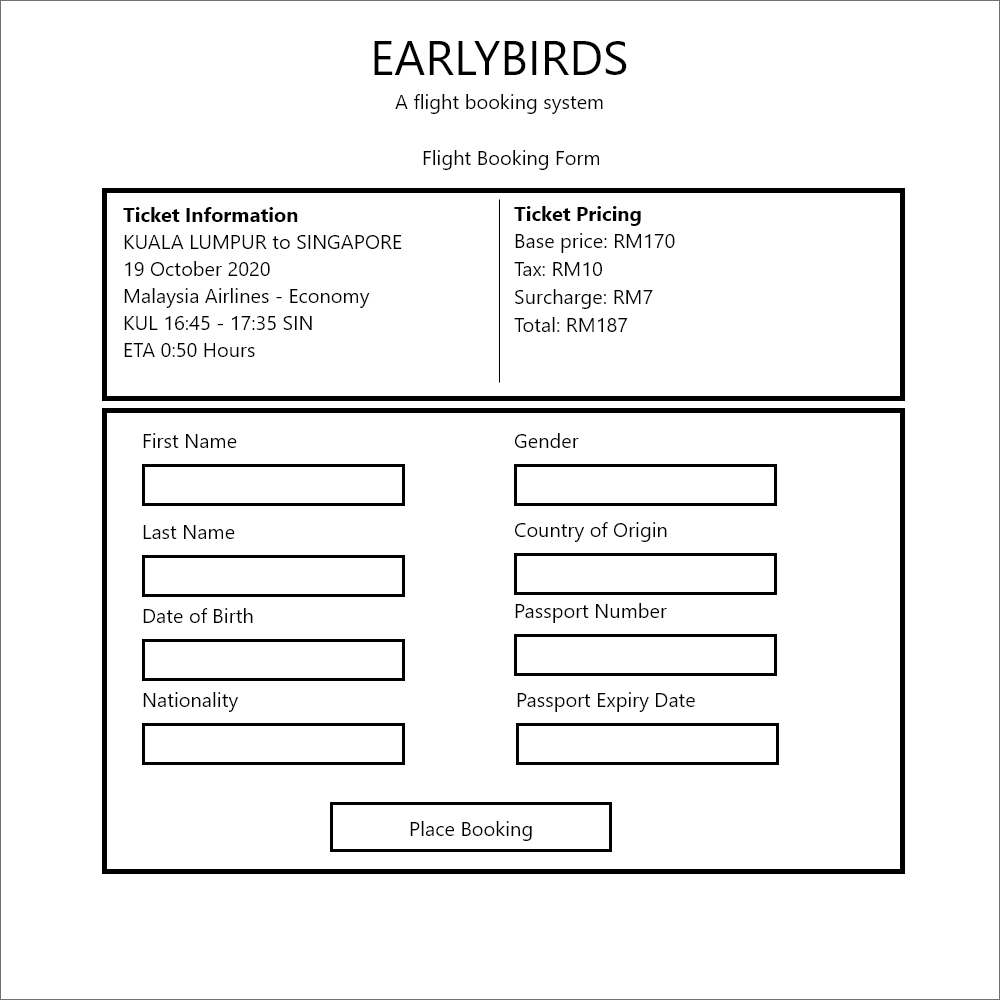
### 3.1.4 Search Result



**Figure 9** is the search result of available flights with constraint provided by the user

The search result wireframe will display fetched flight options that match closely to the user’s defined constraints in the previous search form. This page will only display information deemed as necessary for the user in their decision-making process when choosing their desired flight. The information provided are airline company, airline logo, ticket price, flight callsign, aircraft model, flight class, flight’s Estimated Time of Arrival (ETA), departure airport and time, and arrival airport and time. The selection button leads the user to the booking form with the flight information loaded once clicked.

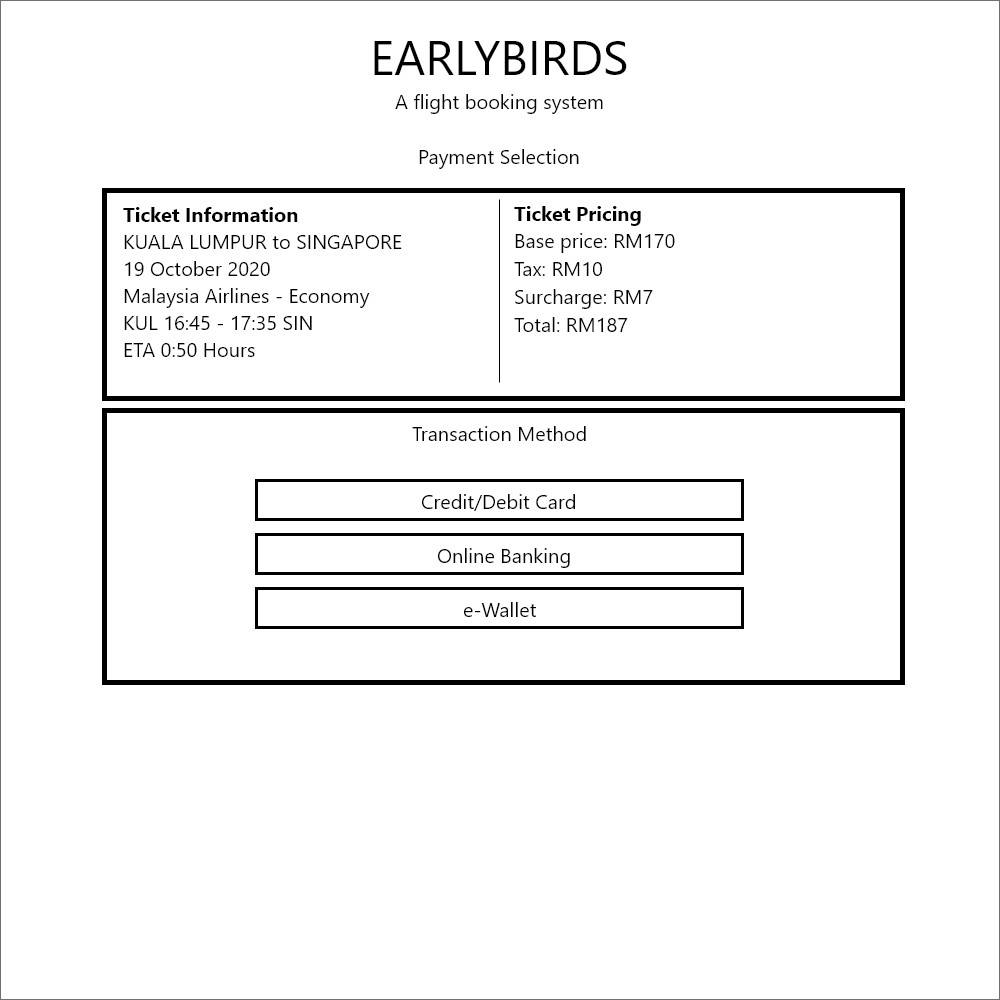
### 3.1.5 Booking Form



**Figure 10** is the booking form a user must fill-in to book for the flight

The booking form wireframe displays system-fetched information as well as to receive user inputs mainly for official flight booking record. The fetched information is the selected flight information as seen in the search result page and total payable after inclusive tax and surcharge calculation. On the input side, user is required to insert their first name, last name, date of birth, nationality, gender, country of origin, passport number and passport expiry date for official immigration documentation. The button at the bottom of the page will lead the user to the payment page.

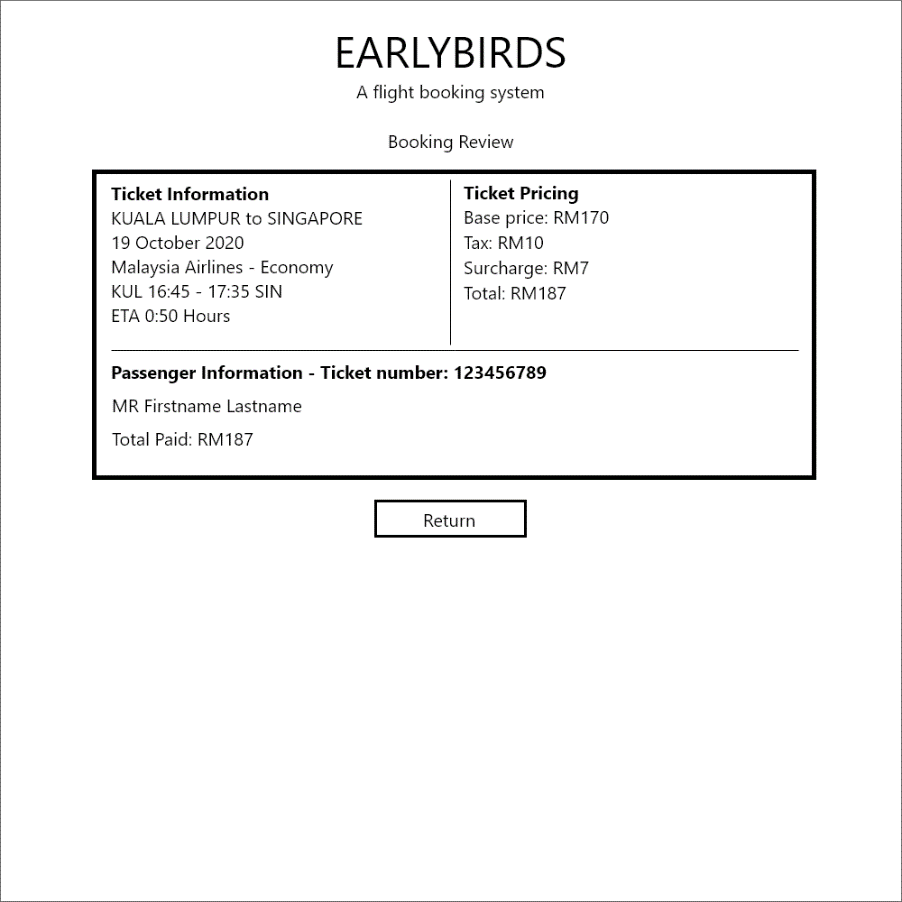
### 3.1.6 Payment Selection



**Figure 11** is the available payment method for user to select to pay for the booking

The payment selection wireframe will be displaying the same flight and pricing information from previous page with exception on the buttons that handles link to external transaction sites. The three buttons have label to help user differentiate which leads to which site. The three payment methods are credit/debit card, online banking and e-Wallet. Once the user presses on either of these three buttons, they will be redirected to the site via secured gateway. However, this proposed does not have a legitimate monetary handling system and therefore, it will not redirect the user to any external banking sites. Instead, it will recognise the booking as paid and sends the user to booking review page.

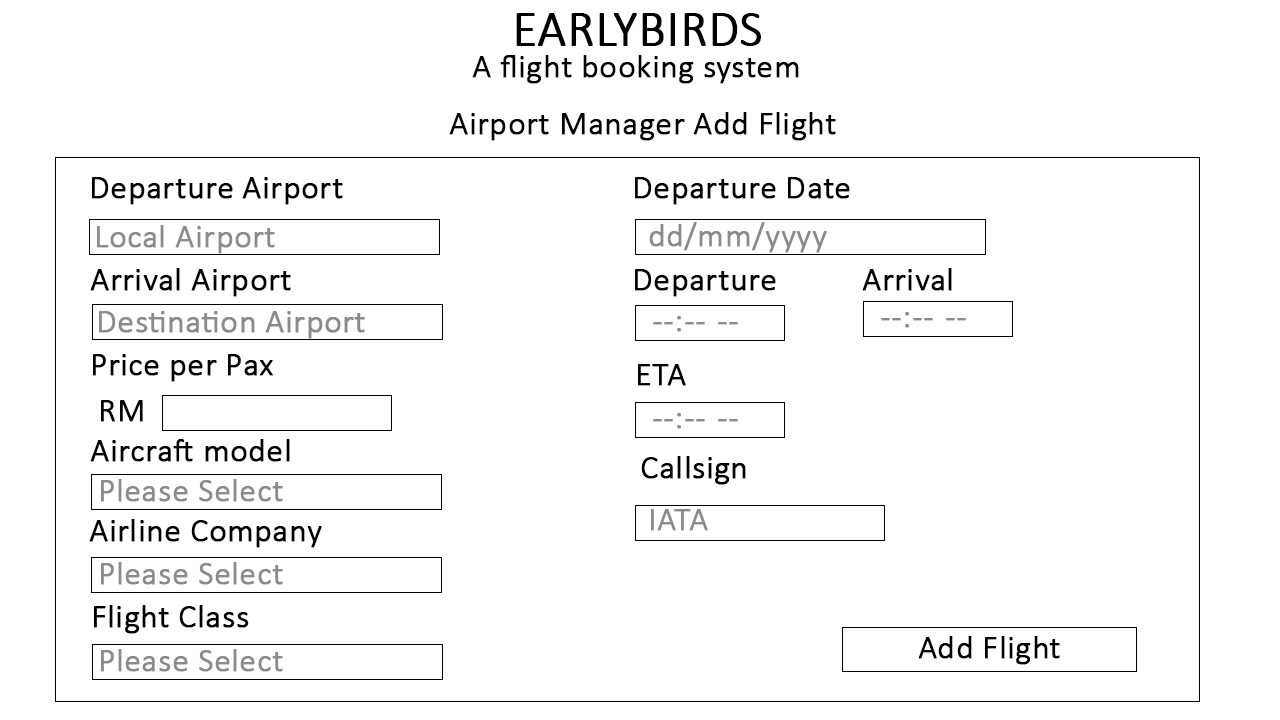
### 3.1.7 Booking Review



**Figure 12** is final page of the booking procedure where summary of the booking is displayed and saved as history

The booking review wireframe exhibits the only functionality this page has and that is to display the summary of the booking. It has no means nor reasons to accept any form of user input other than the user understands and acknowledges of the booking placement. The information displayed are the selected flight, pricing and passenger’s information with the total amount paid. The return button will take the user back to search page to perform another search or booking.

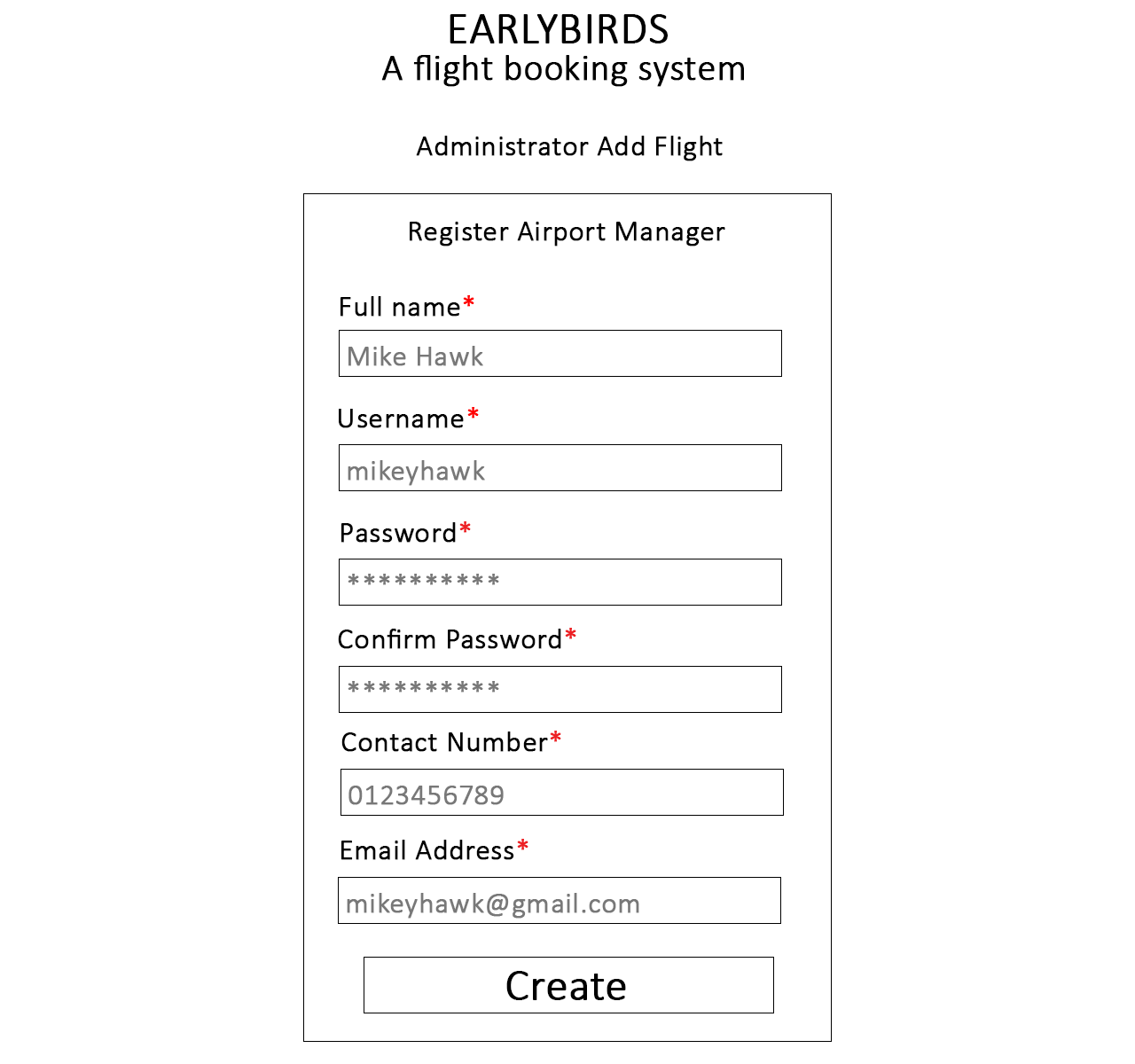
### 3.1.8 Airport Manager Add Flight



**Figure 13** is the system to add new flights to the schedule from airport manager

The add flight system is used for creating new flight schedule that helps with the creation of new flight, that adds to the flight schedule. The flight schedule added from here will be shown to customers when the customer starts making a search of flight schedule from the flight search page. The same data is also viewable and manageable by airport manager from view flight table in airport manager page.

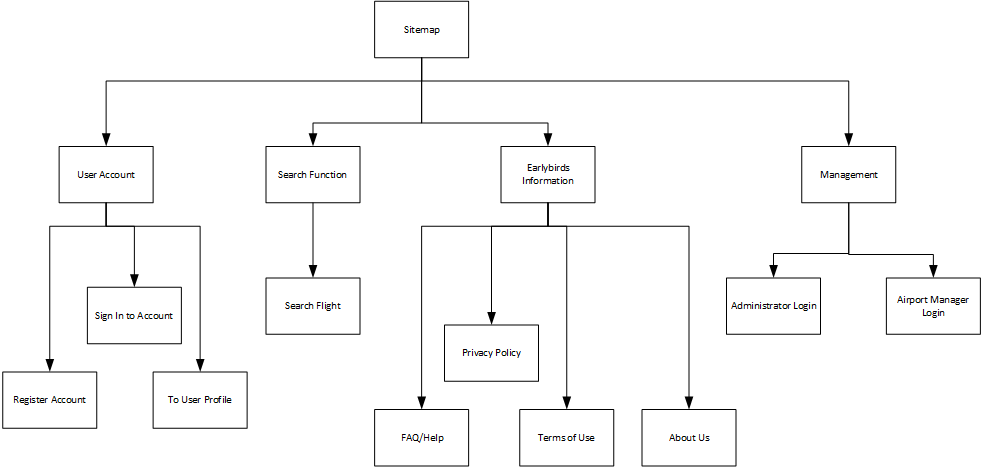
### 3.1.8 Administrator Add Airport Manager



**Figure 14** is the system to add new airport manager from admin page

In order for an airport manager to use the system, they will have to request from the admin to register an account for flight manager. The airport manager will contact the admin via email and pass necessary details to the admin such as the full name, desired username, contact number, and email address. The details that the admin obtains from the email will be used to register an account, and account details will be emailed back to the airport manager to be able to login.

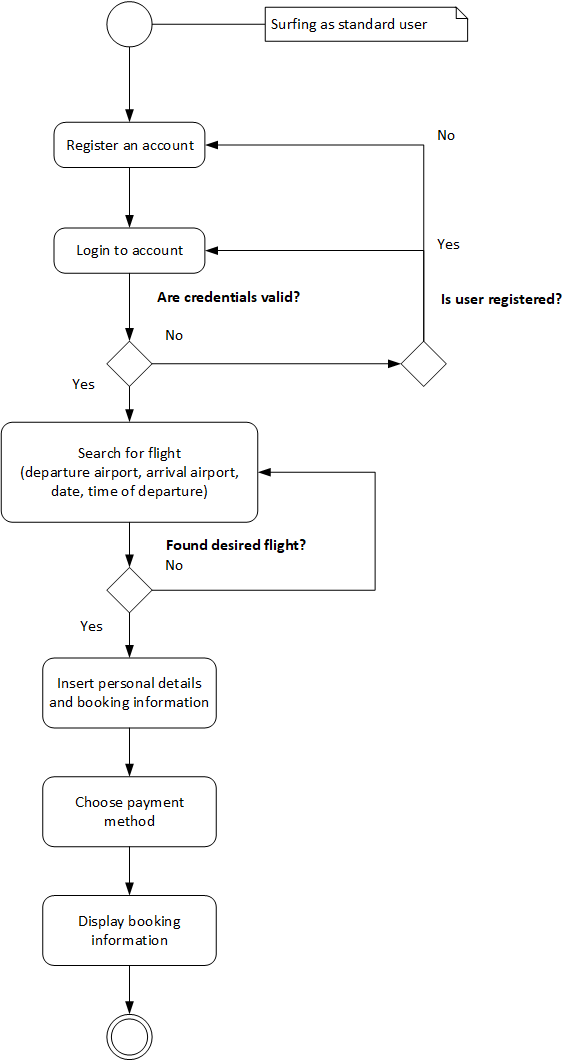
## 3.2 Navigational Structure



**Figure 15** is the navigational structure of the Earlybirds from sitemap page

## 3.3 Activity Diagram

### 3.3.1 User Activity



**Figure 16** shows user flow when using the system after registration and login

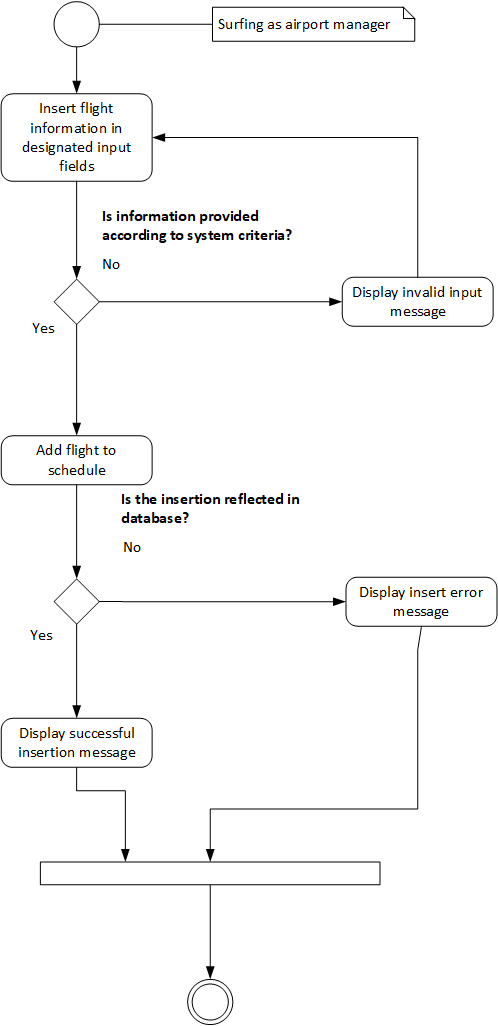
The activity diagram above depicts a user action while using the proposed website. First, the user starts by registering an account by providing required details such as username, password, first and last name. Once the user is registered, the user can log into the system using the username and password specified during the registration time. If the credentials inserted are invalid, the user will be asked to key-in their username and password again. However, if they somehow are unregistered when they got to the login page, they should register first.

After logging in, the user can search for available flights by specifying current airport, destination airport, date and time of departure into the input fields. If the search function returns no result, the user is recommended to retry by specifying different keywords or constraints into the input fields. If there are results from the search, the user can make their decision by comparing the individual flight pricing or flight duration before selecting.

Once they have selected a flight, they are required to fill in a booking form by specifying their personal details and passenger information like first name, last name, date of birth, gender, nationality, country of origin, passport number and passport expiry date. When they are satisfied with the information, they are allowed to proceed by choosing payment method. There will be three payment methods to choose from mainly credit/debit card, online banking and e-Wallet.

The user must choose either of these three before completing the booking. Right after the user chose their payment method and have it verified, they will be redirected to the booking review page to read and acknowledge the booking information they had placed earlier. The user has completed the linear flow of performing a booking.

### 3.3.2 Airport Manager Activity

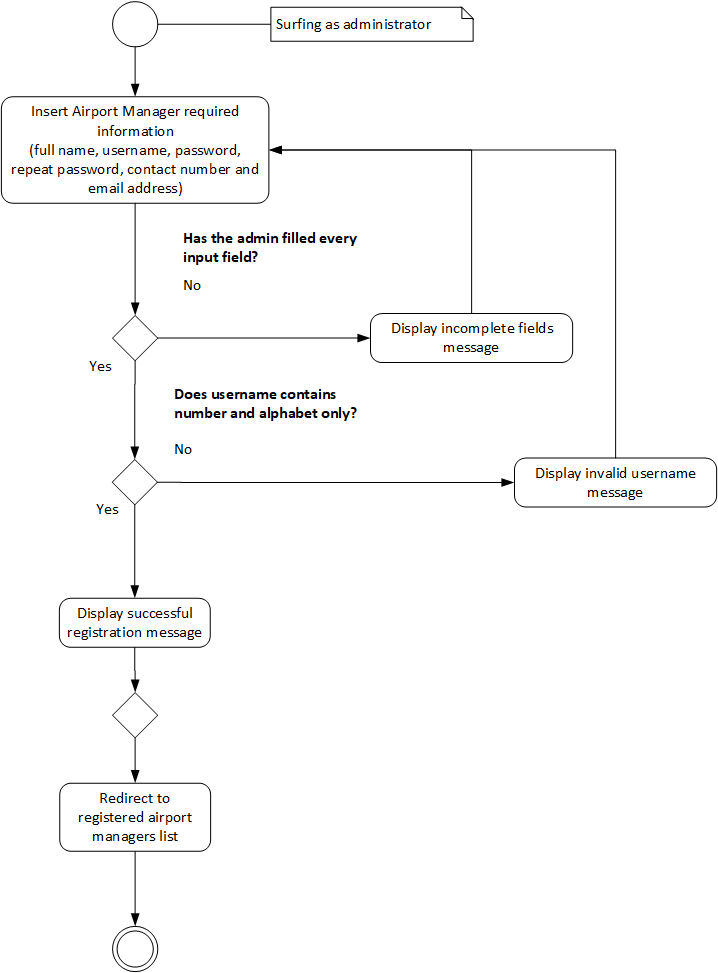


**Figure 17** shows airport manager flow when adding a flight to the airport schedule

The activity diagram above shows the activity of an airport manager primary feature which is to add a flight to the ongoing schedule. First, the airport manager will fill in the form with required information such as departure airport, destination airport, ticket price, aircraft model, airlines, flight class, departure date and time, arrival date and time, and flight callsign.

If the inserted information is not according to the system format, the airport manager will be notified and asked to address the issue immediately. If the information provided has met the system criteria, the system will perform database insertion with the specified inputs through SQL statement. If the insertion failed to appear inside the database, an error will appear notifying the airport manager of failed insertion. If no problem were encountered, a successful insertion notification will appear.

### 3.3.3 Administrator Activity

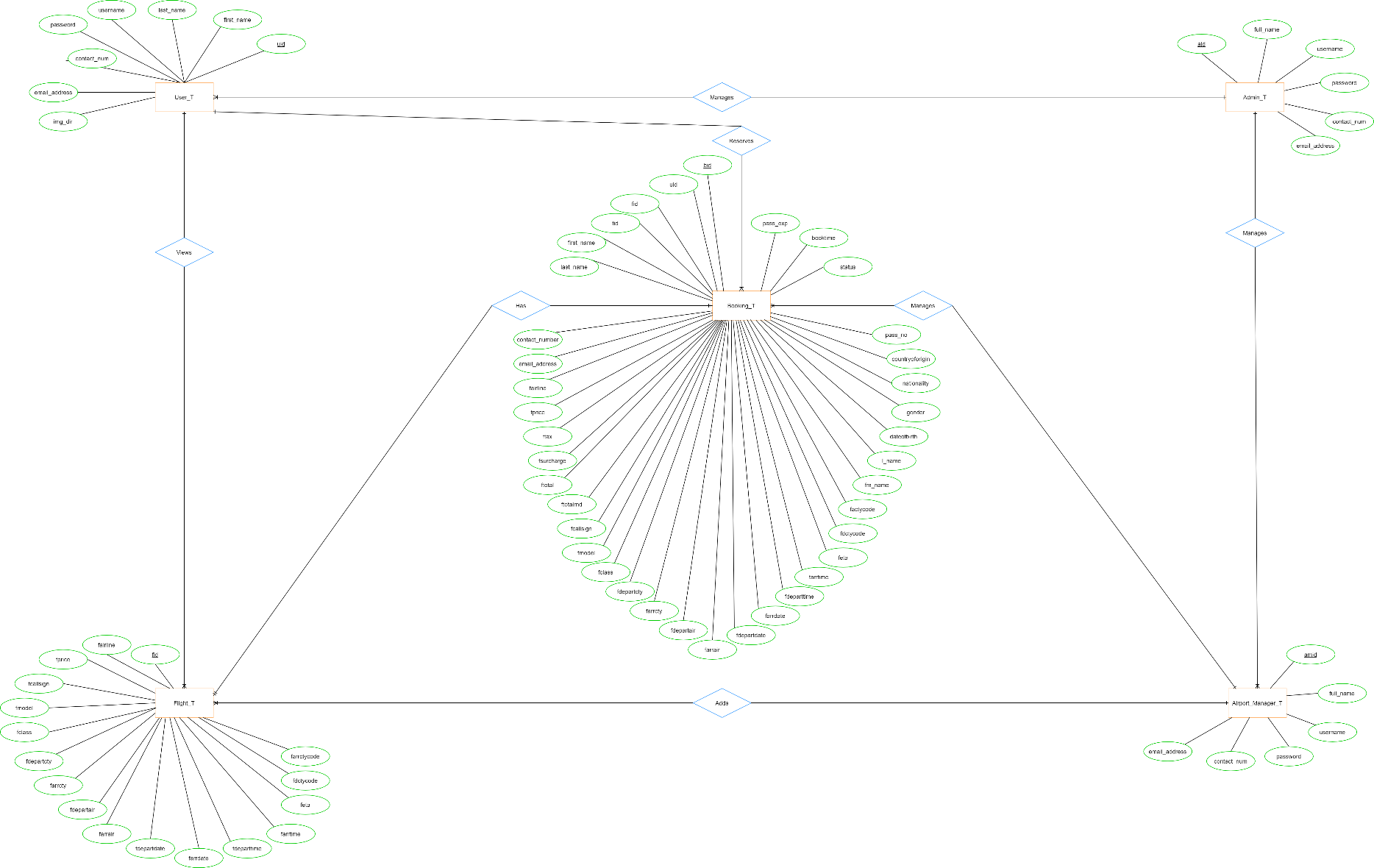


**Figure 18** shows administrator flow when registering an airport manager account

The figure above displays one of Administrator’s function in the system which is to register airport manager as they are unable to register themselves due to business policy. An administrator starts registering by inputting the airport manager’s full name, username, password, repeat password, contact number and email address into the required input fields. If the administrator left out any required input fields, a message will appear asking the administrator to fill in the empty field.

If there is no error, the inserted username will pass through a validation function to see whether the input contains only numerals and characters. If there are entity besides numerals and characters, a message will appear telling the administrator to remove disallowed entities from the username. When the username checks out, the registration shall proceed normally with the system querying the database for new record insertion. Once settled, a successful registration message appears, and the administrator will be redirected to list of airport manager view.

## 3.4 Entity Relationship Diagram



**Figure 19** shows the ERD Diagram for EarlyBirds

# Implementation

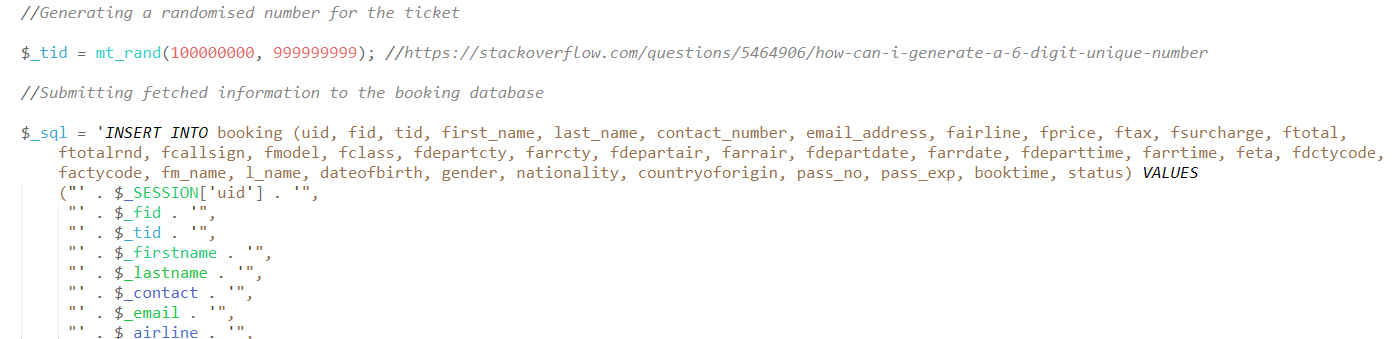
## Create

### Booking a Flight



**Figure 20** displays a snippet of flight booking form page for users

The user of the website can place booking by going through a linear procedure. At some point of the procedure, the user be required to insert passenger’s details and press a button to proceed with the booking.



**Figure 21** shows the PHP SQL statement to add completed booking into the system

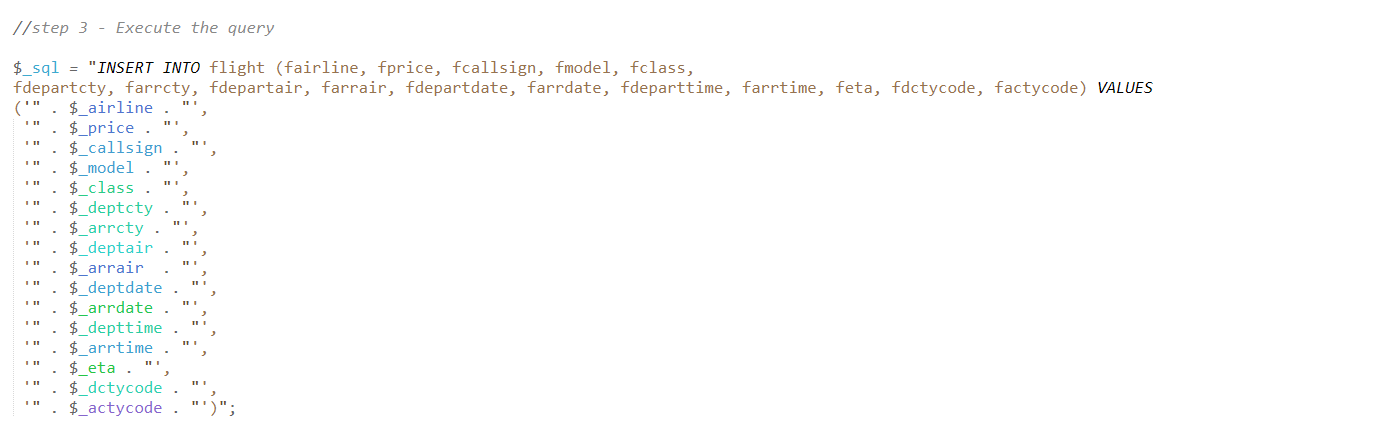
Once the button is pressed, the SQL statement depicted in the picture above will be queried to the database and the booking record is finally inserted and be fetchable by the system.

### Adding a Flight



**Figure 22** is about flight schedule insertion form used by Airport Managers

Airport managers are able to insert flight into the ongoing schedule by filling up the required input fields under the <form> tag. The inputs will be processed by PHP module into a usable record for the users.

****

**Figure 23** is an excerpt of the SQL statement used to insert the flight schedule information into flight database

The inputs provided from the previous form will be carried via POST superglobal and are contained into separate variables. After that, these variables are placed into the SQL statement before it is being sent to the database for insertion. The newly added flight will become available for users to book later.

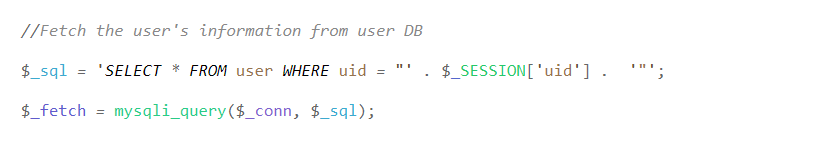
## Read

### User Profile



**Figure 24** is a snippet code of user profile page with PHP echo seen in some of the attributes

Users have the privilege to modify their own profile by going to the profile page. In this page, the user can read their personal details and credentials. These details are fetched upon page load with the help of PHP and SQL query. To simplify read and update operations, there is a <form> tag handling the inputs in case the user intends to update their existing details.



**Figure 25** is a PHP statement used to fetch user information from the database

The SQL statement is used to fetch the user’s information from database. The fetched information will be passed into variable containers and these containers will be echoed into the respective fields’ value attribute upon page load.



**Figure 26** is another part of the PHP with syntaxes used to contain the fetched results for display

As mentioned previously, once the SQL statement returns results from the database without any error, the variables through mysqli\_fetch\_assoc() function will contain the data of each column. Due to the special case of user registration where they are not required by default to specify contact number and email address, an if-else statement is written to check if the SQL statement fetches NULL data and replaces them with ‘No Information’ label.

### Flight Search Result



**Figure 27** is HTML code for user-performed flight search result page

Users can perform search function for available flights on this website on another page. However, there are input fields where the user inputs from previous page will be brought over into this page and those are the user search constraints. These constraints are the determinants to the fetched flight schedules from the database. Although this page carries over the preceding user inputs, the result is fetched as soon as this page loads.



**Figure 28** is a wall of PHP syntax and SQL statement combined to fetch desired result via filtration

As mentioned above, the result is fetched as soon as the page is loaded. This is due to the presence of PHP code inside the page itself. It can not only fetch result statically in a sense that the SQL statement is fixed, it can also accommodate to user’s filtration function by utilising an if-else statement which check for user filter keyword. The filter inputs are under HTML code but once the button is pressed, the page will refresh with a new POST value that is the filtration.

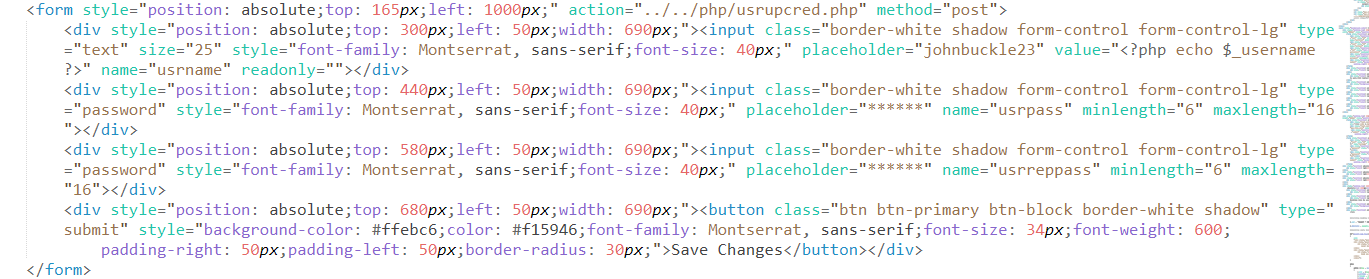


**Figure 29** is a PHP echo loop written to create a result container for every available fetch from database

As to how the fetched flight records are fed into the page display section, there is a PHP code placed inside the HTML <body> code which will initiate a loop for how many results are fetched. The loop will refresh the variable containers with different values so that the displayed results carry totally different information rather than repeating the same data from a single row over and over.

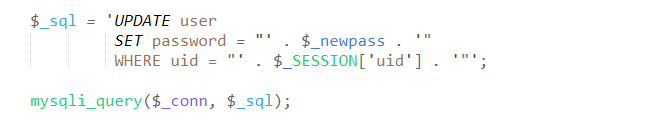
## Update

### User Password



**Figure 30** is HTML form code to handle user-inserted password and new password in profile page

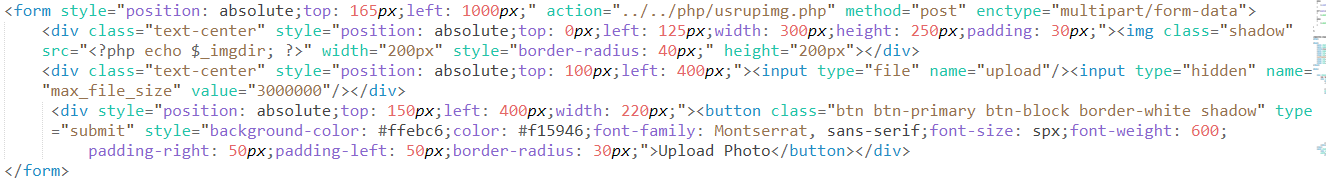
The users can update their credentials at any time through their profile. The input fields for the new password and repeat new password can be found under the <form> tag as it allows for the user-inserted inputs to be sent to the password update PHP file. The reason as to why there are two input fields for the same password is to ensure the insertion is identical with no character mistakes. The validation is done via the PHP file as well.



**Figure 31** is the Update SQL statement issued to amend the user account password in database

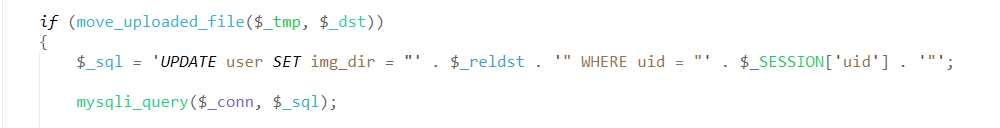
As the users press the ‘Save Changes’ button, another PHP file is opened and the inputs from profile page will be carried over through POST superglobal. It is stated that a password checking is conducted in this PHP file. After the passwords are verified to be similar, the SQL statement can be queried to the database. This SQL statement will update existing user record by amending only the row with specific user ID and that is the ID of the user that initiated the password update. Once the query is sent, the user will be redirected back to profile page after confirming the password change notification.

### User Profile Picture



**Figure 32** is a HTML form code to handle file upload for user profile picture

The users are able to upload their own profile picture into the account. A special input is used to allow for such procedure to happen. The tag <input> has its type attribute changed to “file” and some modification were done to the <form> tag to ensure file upload is possible. Although the input may accept any kind of files, the PHP file handling the upload sequence will reject any files with excluded filetypes in order to make sure only images are accepted and uploaded to the server.

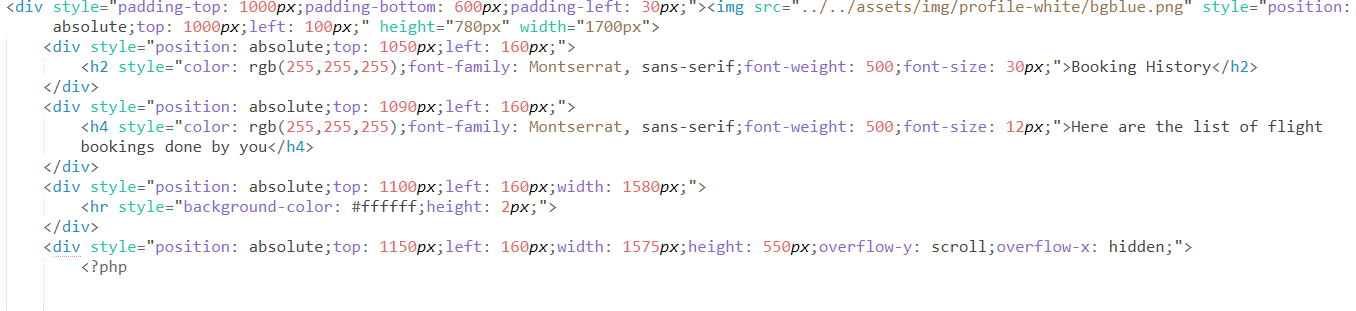


**Figure 33** is a snippet of PHP SQL query inserted to update user profile picture directory to the recently uploaded file

The snippet above shows the function used to perform upload of attached file. The function move\_uploaded\_file() is fed with two parameters that is the temporary directory of the file and the final destination for the file which is inside the server. The if statement is placed to check if the file has been successfully moved into the server or otherwise. However, the file being placed in the server is not enough to get the image displayed in the profile. Hence, the SQL statement is queried to update a special column under user table that points to the image file inside the server directory. Once the update query is passed and no error of file upload encountered, the user will be greeted with successful upload popup and be redirected back to profile page.

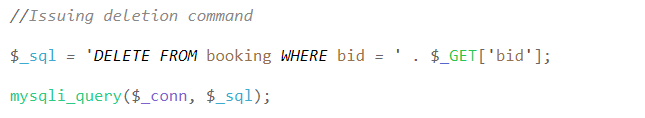
## Delete

### Booking History



**Figure 34** is an excerpt of HTML booking history code section under user profile page

After the user done booking a flight, a save of the booking will be stored and be displayed under the booking section of a user’s profile page. The users have the freedom to review the booking content at any time. Since they have the freedom to review, they also have the choice to delete the booking. Two pair of buttons are included in each booking history card which are either for review or deletion. The buttons are attached with ‘href’ attribute and a GET of selected booking ID which points to, in the case of deletion, PHP file handling deletion of booking.



**Figure 35** is a PHP SQL statement issued to delete the selected booking history

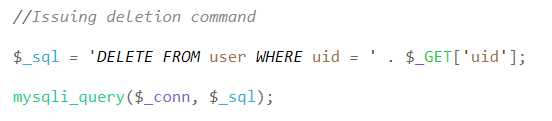
The excerpt of above SQL statement is sent to the database once the user presses on the delete button found on the booking history card. In order to correctly delete booking history, the statement is bundled with an extra GET superglobal set to the index ‘bid’ which is the booking ID information passed from the booking card previously. After the deletion command is issued, a notification of deletion will be shown to the user before redirecting them back to profile page with one less of a booking.

### User Account



**Figure 36** is HTML code to display button and text of deletion to the currently logged in account

As the user are given the ability to perform many functionalities inside the website, they can also terminate their own account for their own reasons. The HTML code is simple with only a text to inform the user of irreversible effect after deletion and the button to commit such task. Once the button is pressed, account deletion function commences and the user’s session will be terminated, as in the user will be redirected back to login page.



**Figure 37** is the SQL command written to delete the account

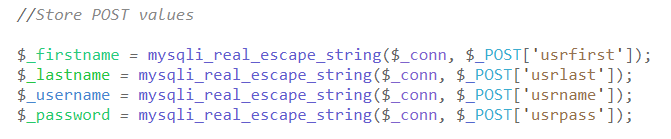
The SQL statement above is sent to the database immediately upon button press. Just like how deletion of booking history was handled, a GET superglobal pointed to ‘uid’ index is attached at the end of the statement to ensure that the account being deleted is the one that is logged in with the ID. Not included within the snippet, a function to destroy session, session\_destroy() is used to force the user to log out after the account is deleted. Once the account is no longer available, the user will be notified via a popup and be redirected to the login page.

## Registration



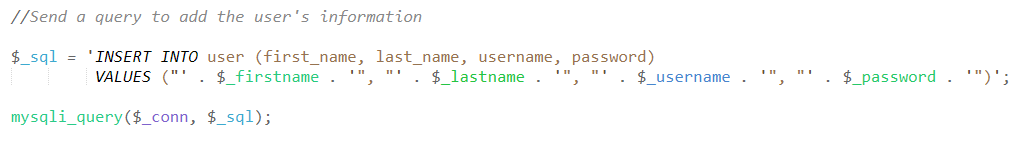
**Figure 38** is HTML form code to accept user details through POST superglobal

The users are able to register in this website through a sequence of HTML, PHP and SQL query. Firstly, the user must fill-in relevant information into the designated input fields that are under <form> tag. This information includes username, password, first and last name of the user. Once these fields are dealt with, the user may press the button with ‘Create Account’ label to proceed with the registration. The <form> tag is equipped with POST method which enables additional security measures on the transmitted inputs to the next file.



**Figure 39** is PHP variables to contain user details sent by the registration form

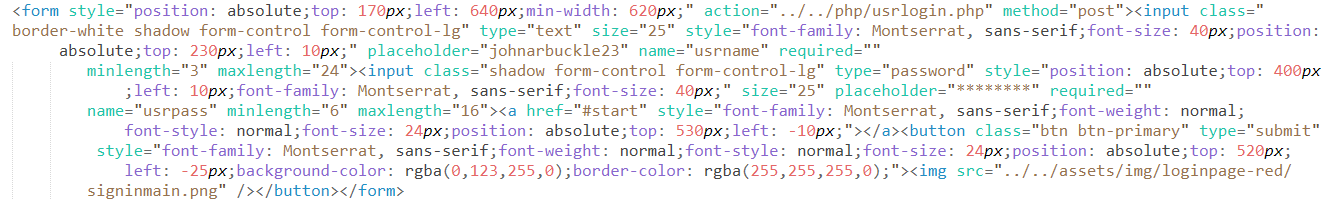
The inputs are then passed into several variables with clear naming convention to describe which variable contains which information. The function mysqli\_real\_escape\_string() is used to ensure special characters such as single quotation mark is excluded and be represented in a different form during SQL statement query to database.



**Figure 40** is the SQL statement to register the user

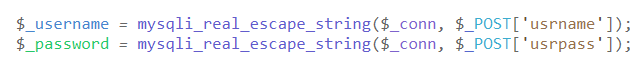
The actual registration takes place once the SQL statement is sent to the database. The SQL statement is made up of variables mapped to existing columns inside the user database table. Once the statement is passed to the database and created a new row, registration for the user is essentially complete. For the user to be notified of complete registration, a notification popup will appear inside the browser and redirects the user straight to the search page, ultimately forces log-in by fetching the recently registered user’s credentials and created a logon session automatically.

## Login



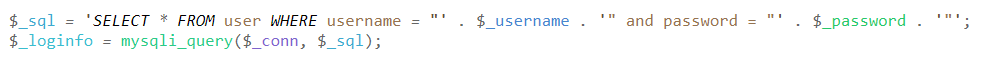
**Figure 41** is the HTML form code to allow for username and password insertion by user

Since a user is automatically logged in upon registration completion, a user can start logging in manually after they are logged out of their account. The HTML excerpt above shows the input fields for two user inputs namely username and password which are used for login. These inputs are included under a <form> tag to ensure for the user-inserted inputs to be passed to the next file for credentials validation. A button to submit the form is mandatory to signal the form to start transferring the data to the next file.



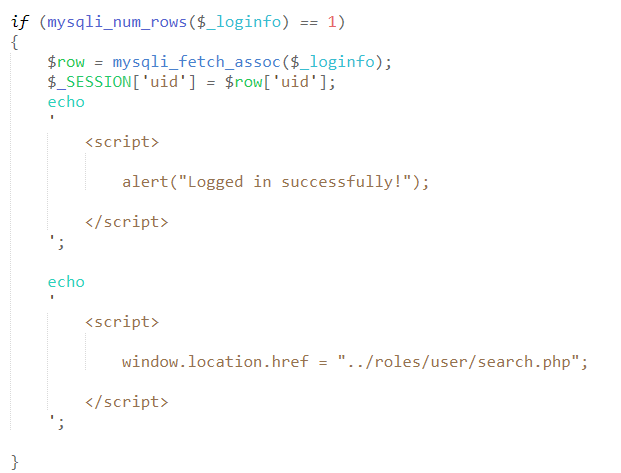
**Figure 42** is PHP variables to store the inserted username and password

After the user presses on login button, the inputs are then passed through POST superglobal to the PHP file that handles credentials validation. Firstly, the POST contents will be extracted and contained inside a variable container. In addition, mysqli\_real\_escape\_string() function is used to exclude any special characters which may cause SQL statement to fail once it is passed to the database. Now that the variables are ready, the inputs can now be verified of its presence inside the system.



**Figure 43** is the SQL statement used to fetch registered users’ credential with similar username and password

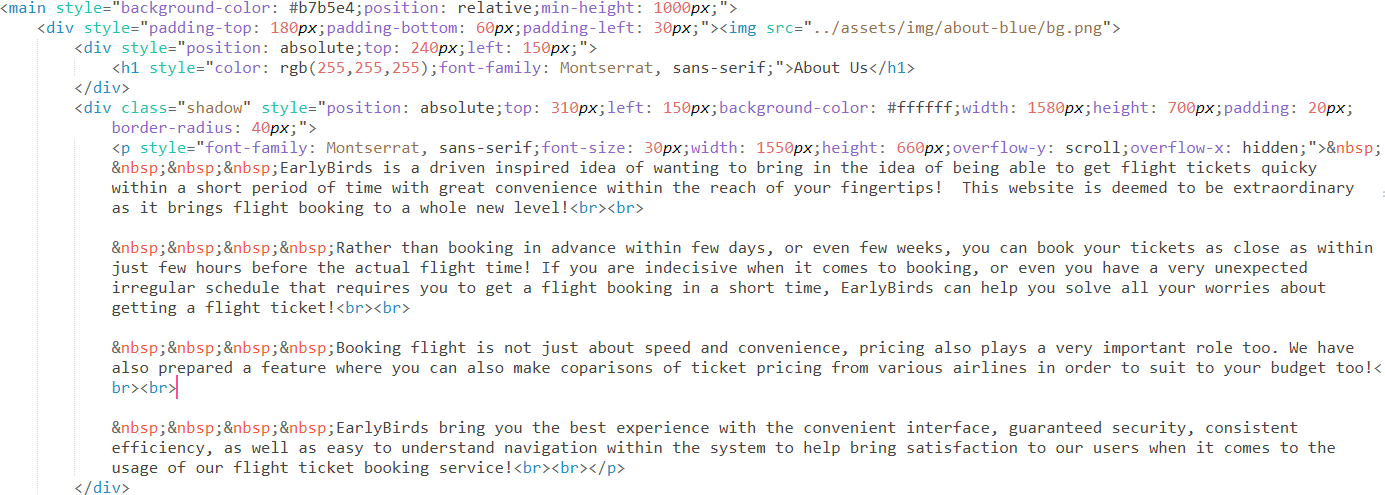
A registered user must always have a unique username. A function to check for username duplicate is available during registration. Due to the unique username setting, the SQL statement used to check for user account availability inside the database becomes a lot easier. By fetching the username and password from database, the system can conduct similarity check by comparison.



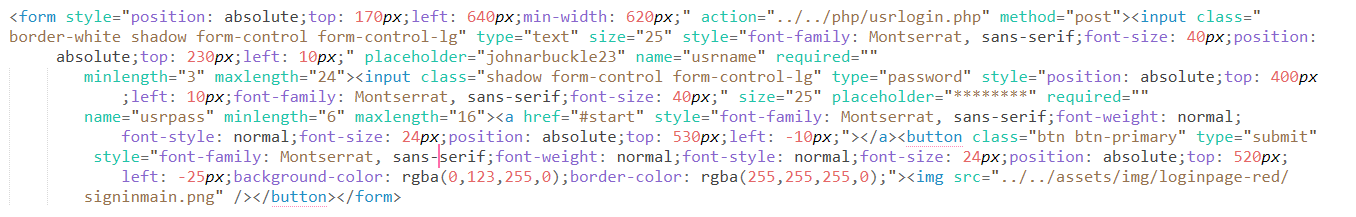
**Figure 44** is the use of JavaScript in redirecting successfully logged in user to the main page

As seen from the snippet above, an if statement is used to check if the fetched rows data is exactly singular. In a situation where there are more than one rows or none at all fetched, the login will fail, making it impossible for the user to login. Unless it is exactly one row, the user will never be allowed to use the website’s accessibilities. For when a user is allowed to login after going through the validation phase, a JavaScript code is invoked to redirect the user to the search page, completing the user’s login attempt.

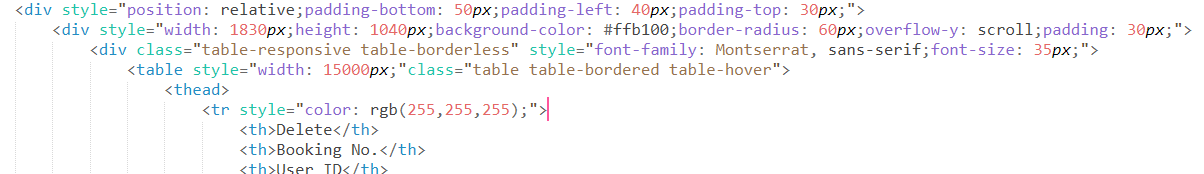
## Self-Created CSS



**Figure 45** demonstrate the use of CSS in-line positioning and font elements in About Us page

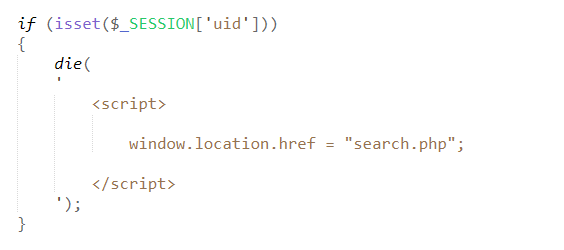


**Figure 46** is an excerpt of user login HTML with in-line CSS positioning and font styling elements



**Figure 47** is an example of CSS in-line sizing applied to user booking table

## Self-Created JavaScript



**Figure 48** is a JavaScript command used to redirect logged in users to the main page

****

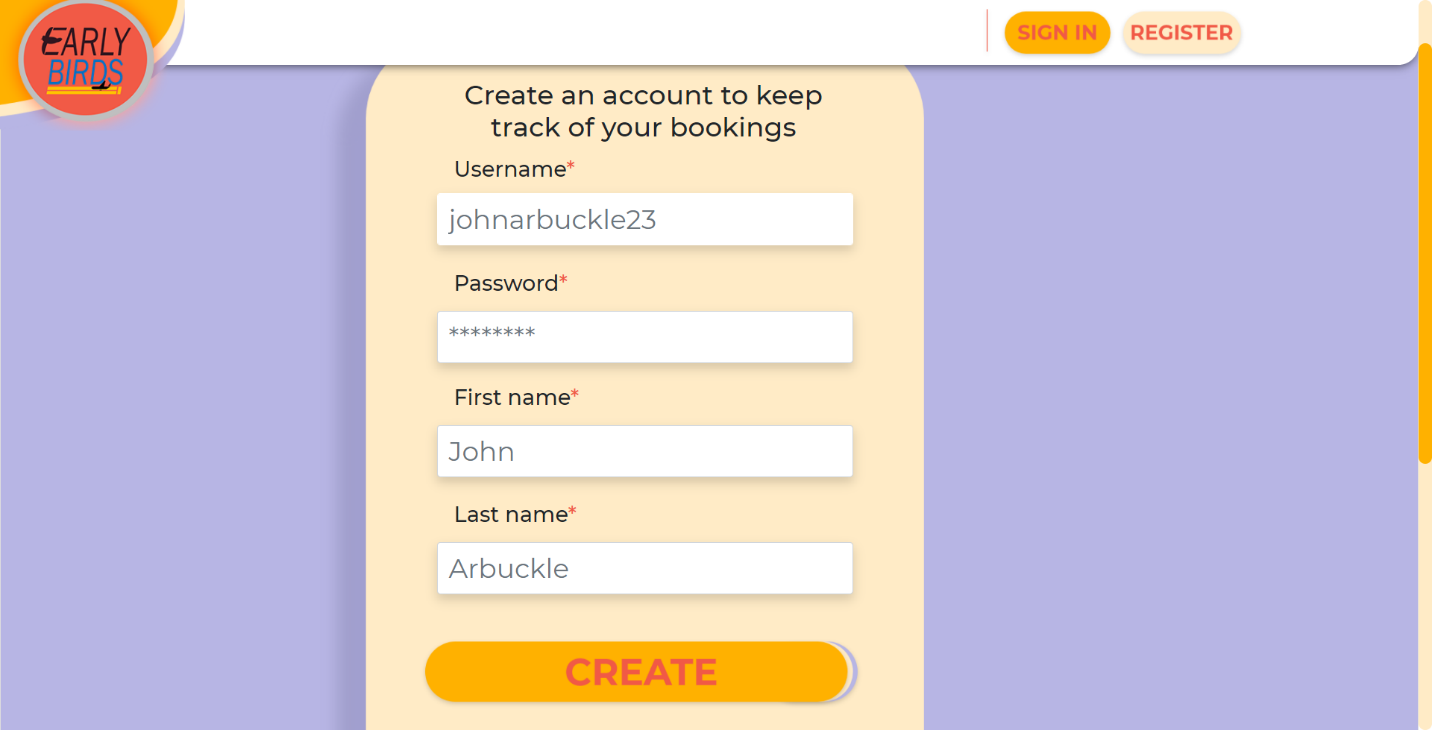
**Figure 49** is JavaScript command used to prevent unregistered or unlogged in users from accessing the functional pages

# Main Section

## User Manual

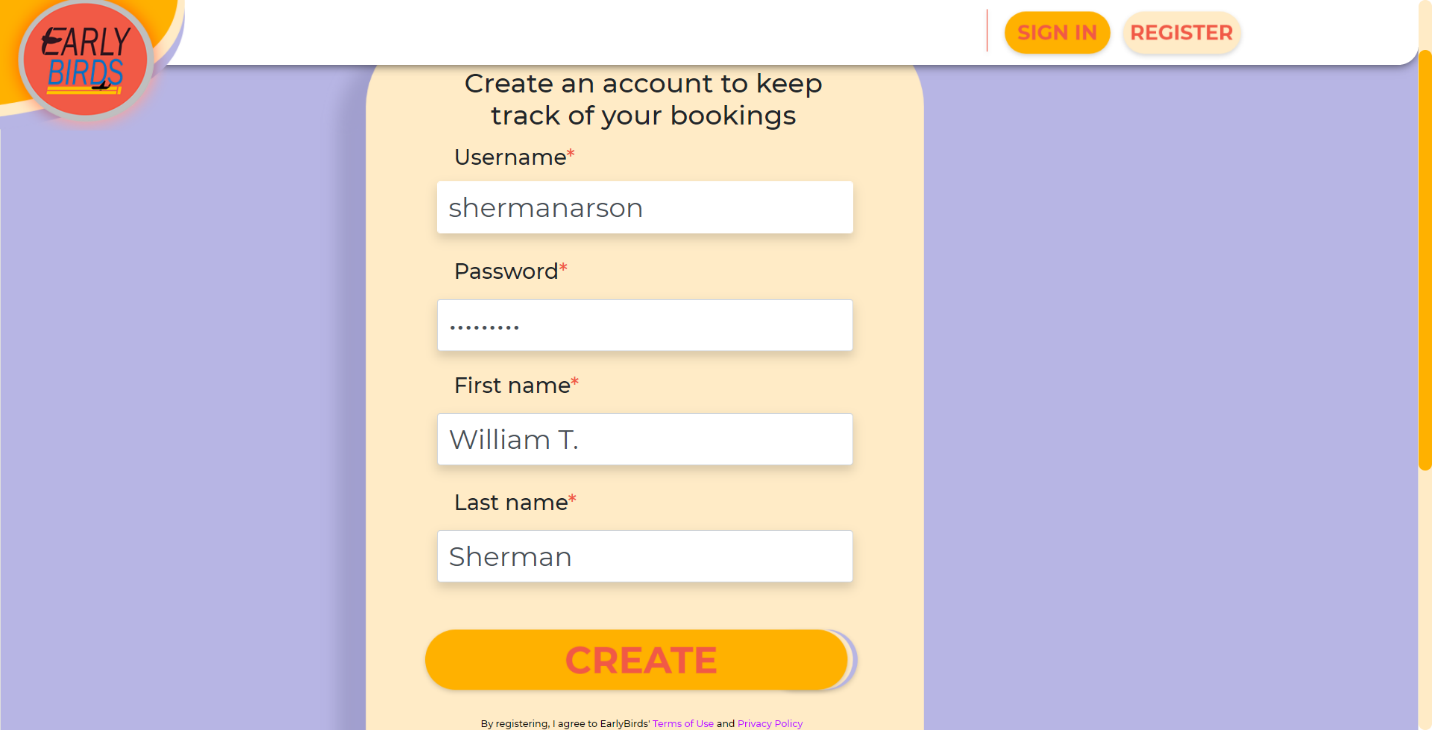
### Create an Account

Users start off with this website by registering for an account. This is a policy made by Earlybirds to ensure every user record are made available for the company to store before being able to use the website’s functionalities. Registration page can be accessed through the landing page or via sitemap.

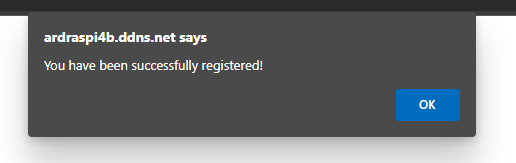


**Figure 50** shows the registration page initial look upon load

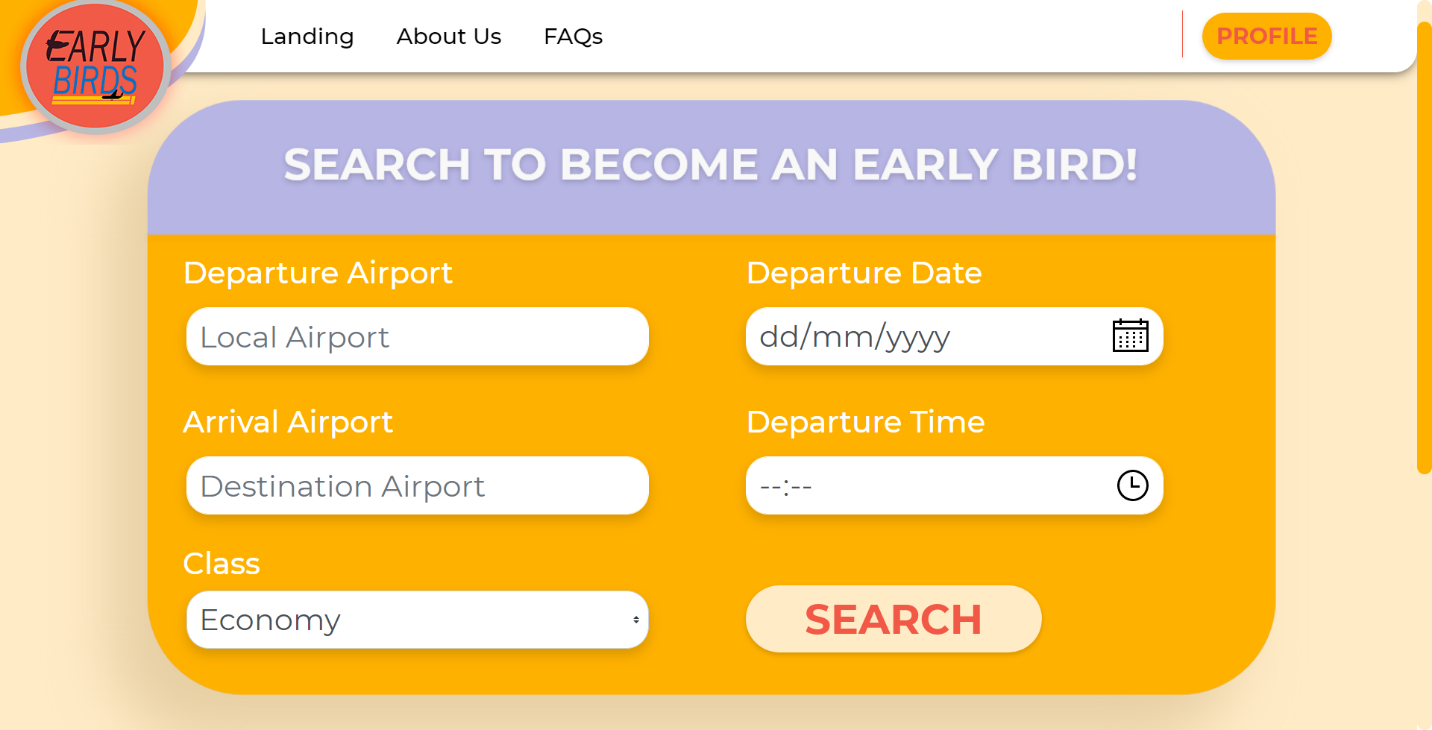
The grey-coloured words found inside the input fields are placeholders to hint registering users of the input format and structure. The form will not allow for any empty field and if there is any, there will be tooltips notifying of fields left empty and the user should fill it at once.



**Figure 51** displays the input fields with respective values inserted



**Figure 52** is a popup to notify the user of successful registration

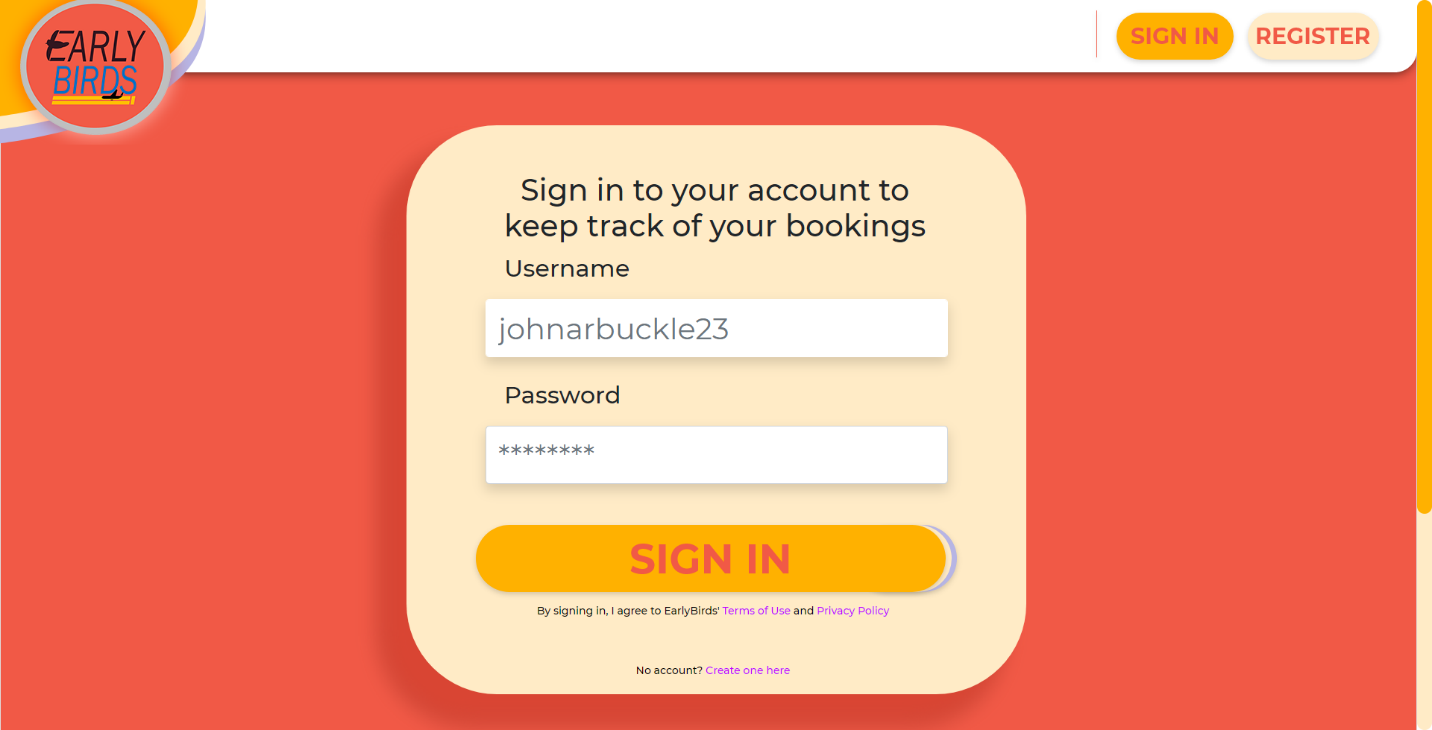


**Figure 53** is the post-registration page to which the recently registered user is redirected to

Once the user is successfully registered, they will automatically be logged in and get redirected to the main page which is the search page for flights.

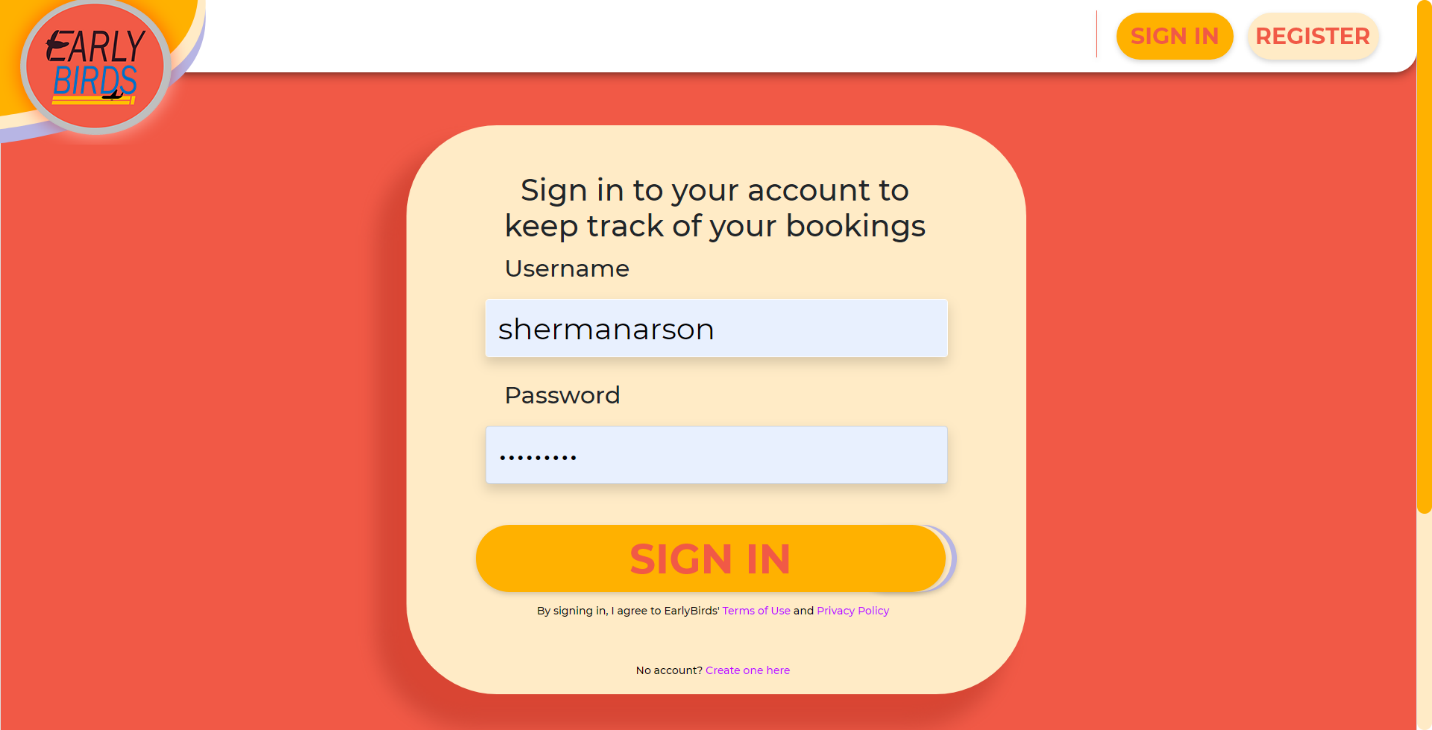
### Logging in

If a user logs out or unexpectedly terminated their session, the user will have to log in manually. Registration will have the users to login automatically, so this only applies to logged out users. Logging into the system is simple as it only requires username and password of the registered user.

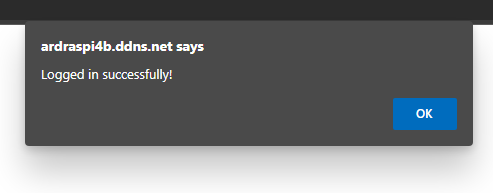


**Figure 54** is the initial look of user login form after load

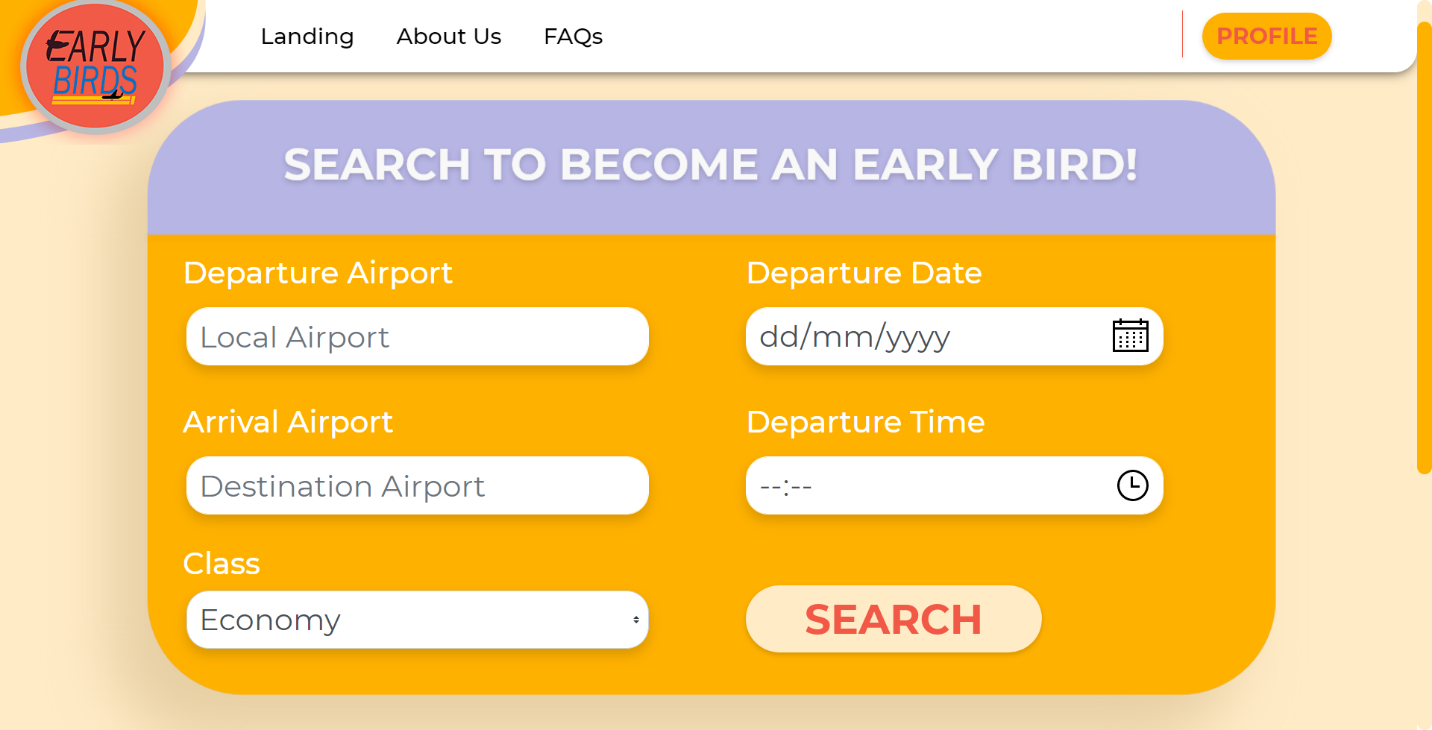
The grey-coloured words found inside the input fields are placeholders to hint users of the input format and structure. The form will not allow for any empty field and if there is any, there will be tooltips notifying of fields left empty and the user should fill it at once.



**Figure 55** is the same login form with username and password of the registered user filled-in



**Figure 56** notifies the user of successful login

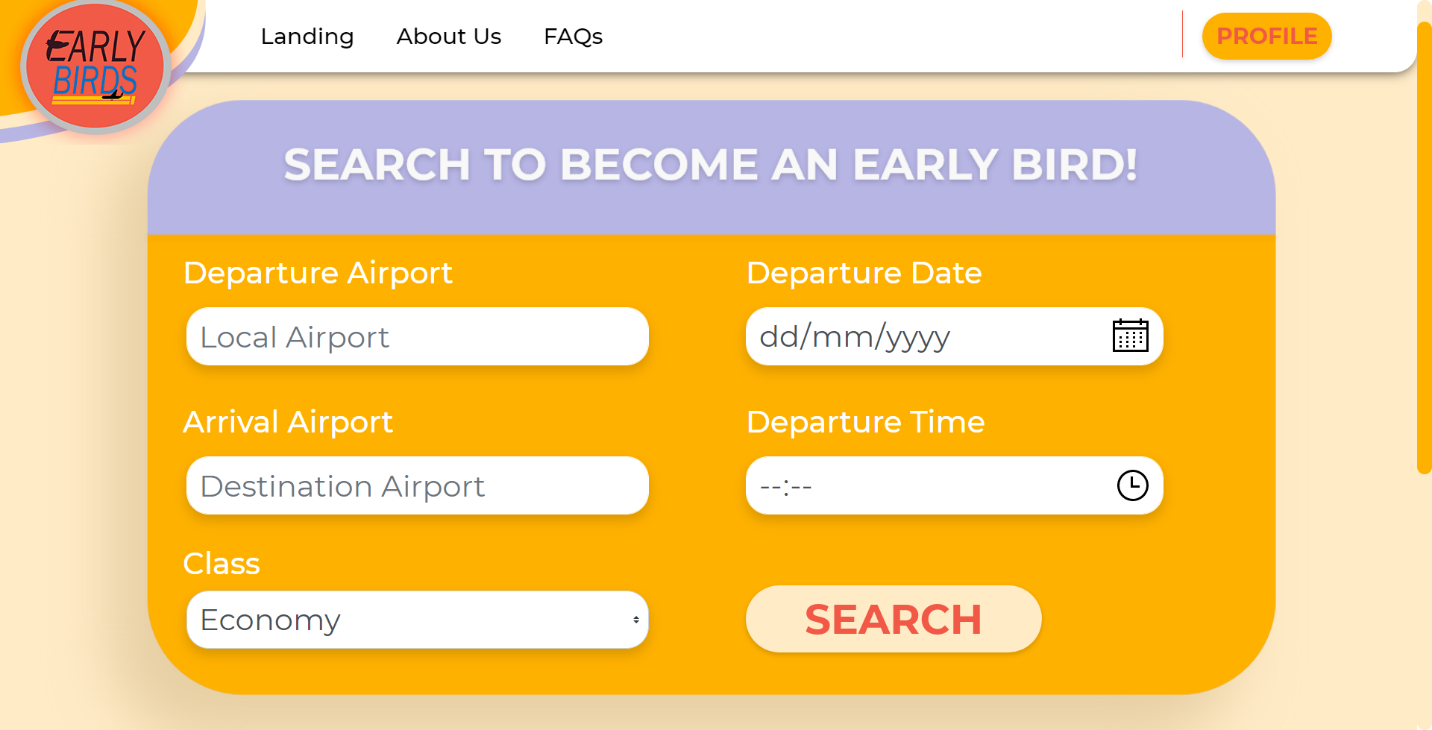


**Figure 57** is the search flight page loaded after the user has successfully logged in

The user will be redirected to the search page after logging in as this is the main feature of Earlybirds website.

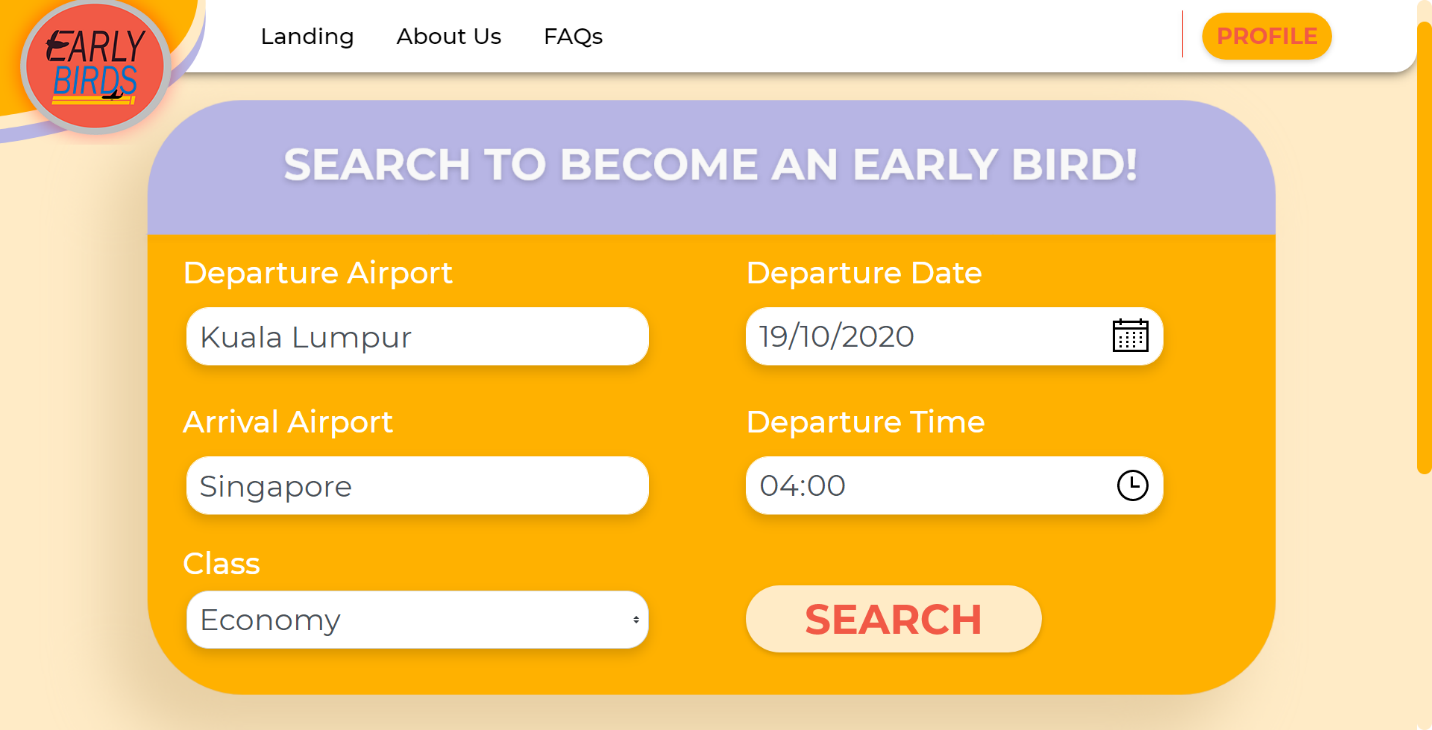
### Booking for a Flight

The users are able to conduct search once they are registered and logged into the system. Flights can be searched via the constraints presented in the form. These constraints will be used as keywords when fetching flight schedules from flight database.

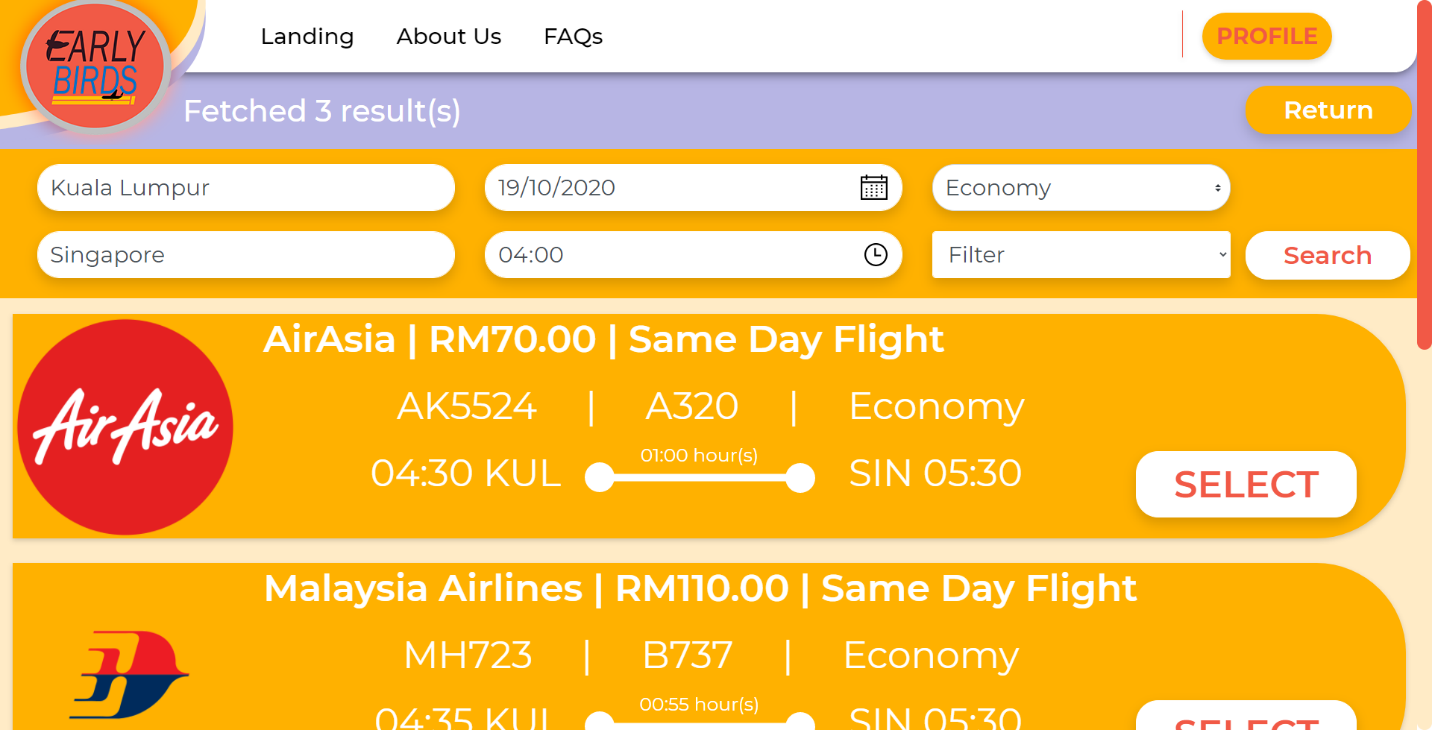


**Figure 58** is the search page in its initial form after load

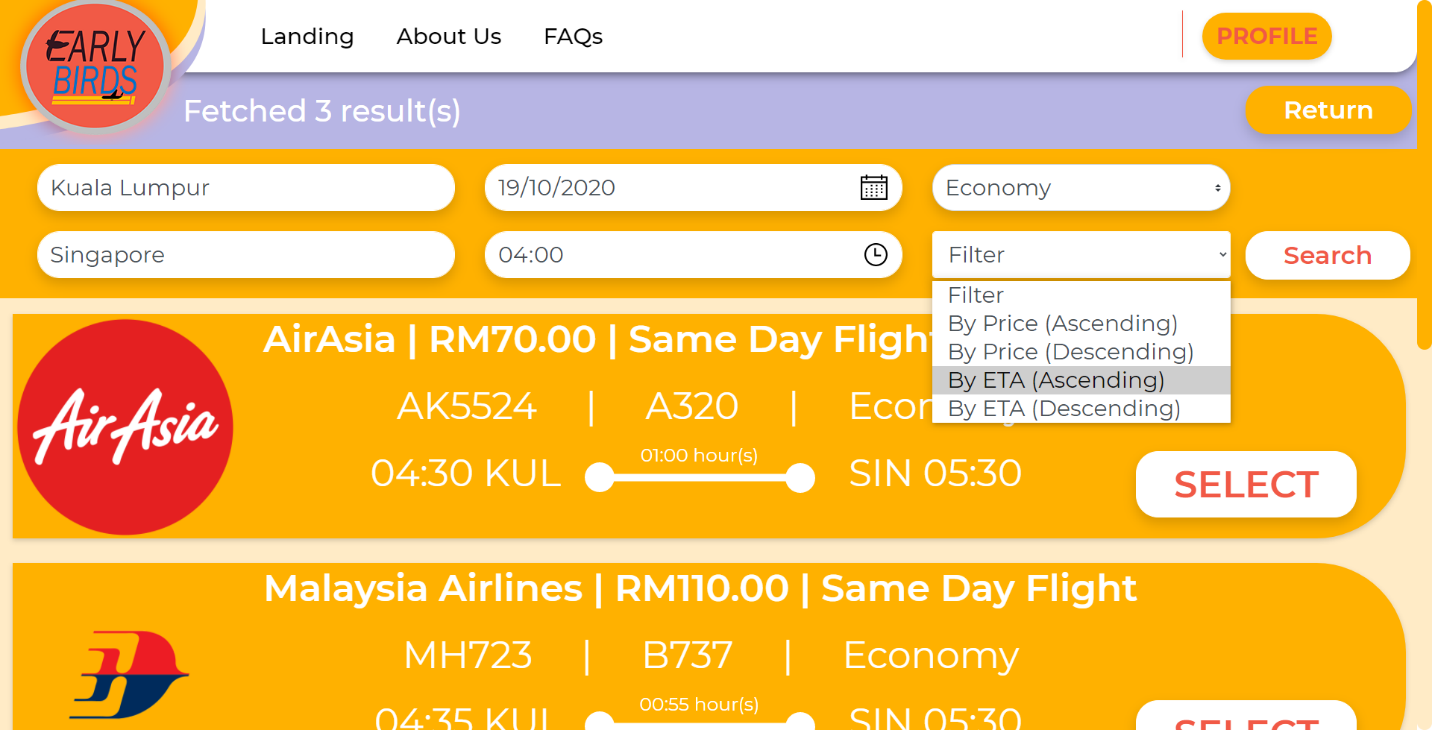
The grey-coloured words found inside the input fields are placeholders to hint users of the input format and structure. The form will not allow for any empty field and if there is any, there will be tooltips notifying of fields left empty and the user should fill it at once. There are sensitive inputs in the form which requires extra caution when inserting values for desired flight result to appear. The inputs are departure and arrival airport as the query used to fetch the flights only looks up for characters matching the input, making it inflexible to ambiguity.



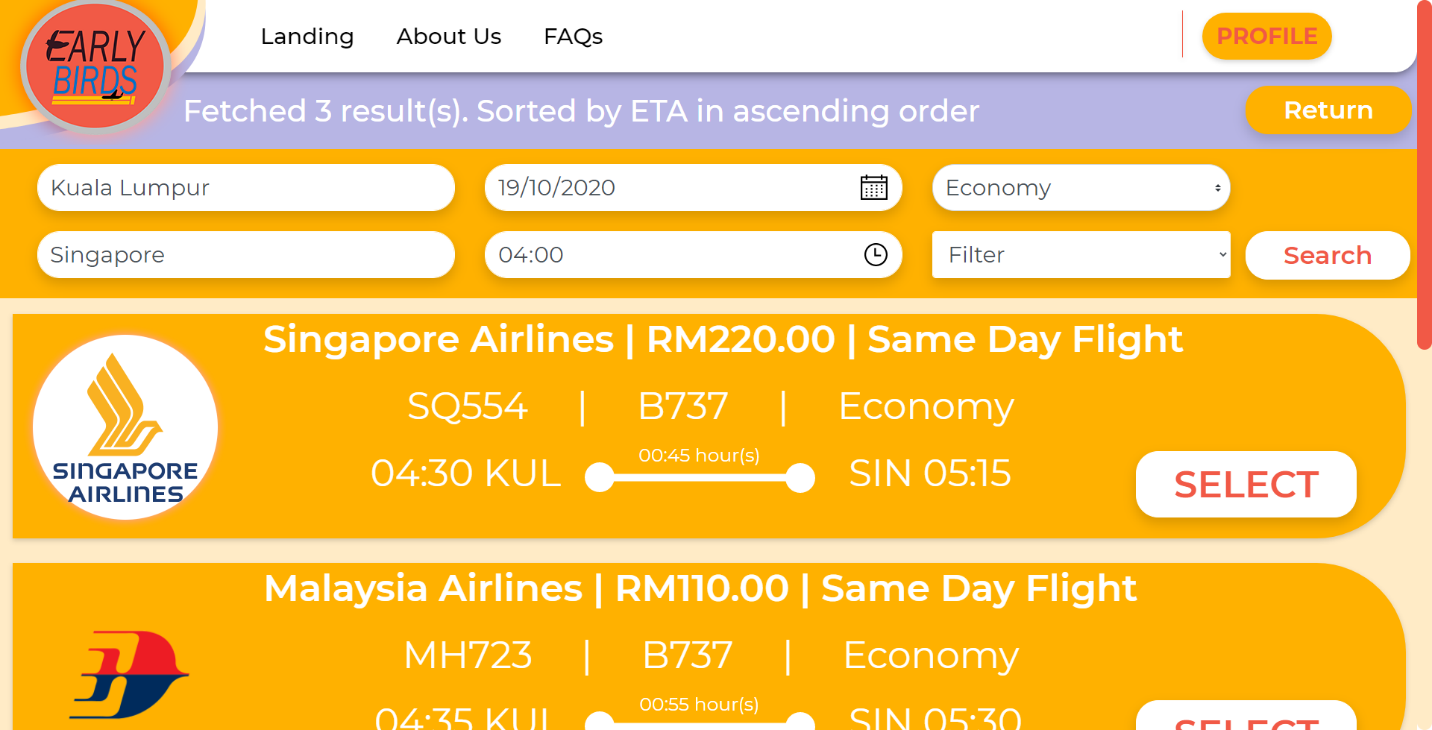
**Figure 59** with its constraint fields filled with respective values



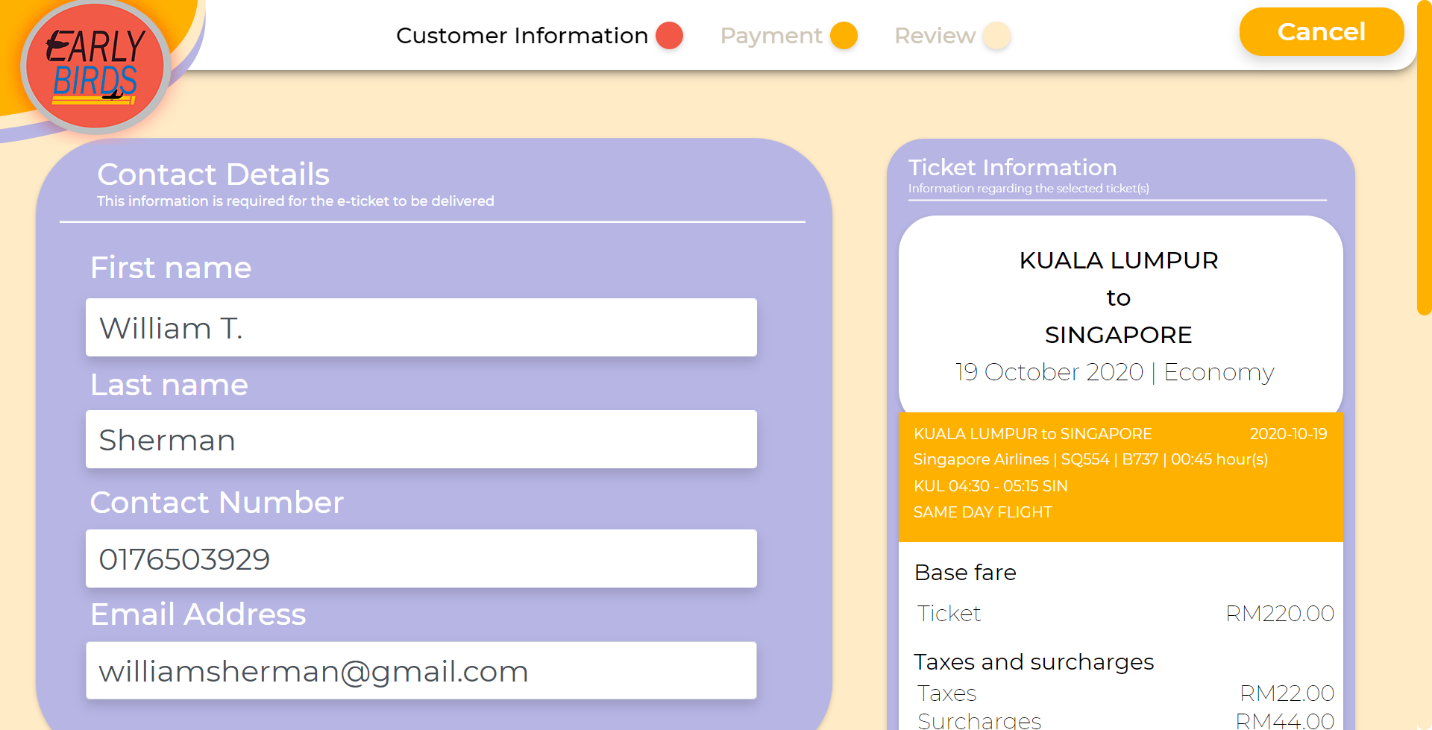
**Figure 60** shows the search result page with fetched flights based on user constraints



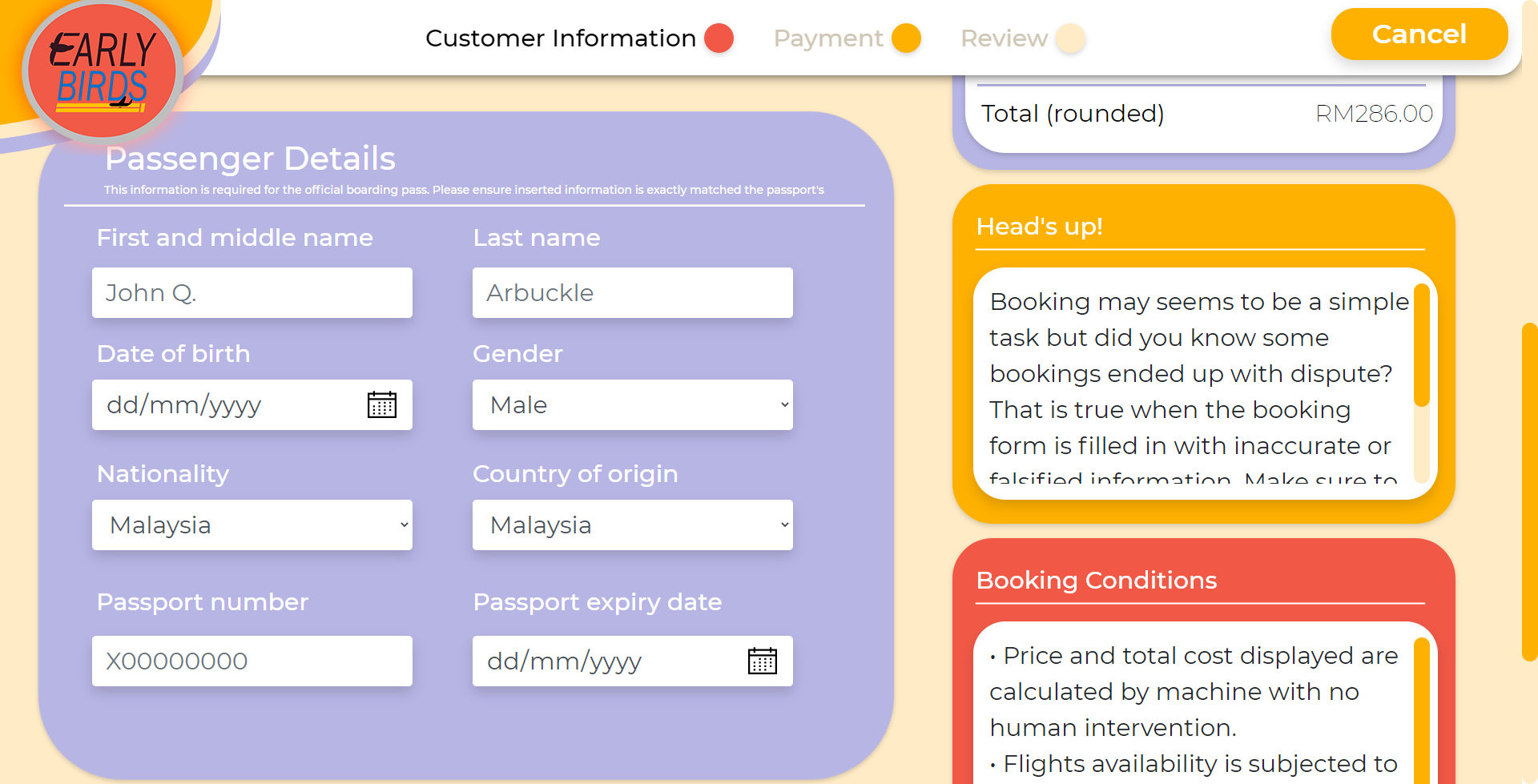
**Figure 61** demonstrates the use of filter function to sort the result



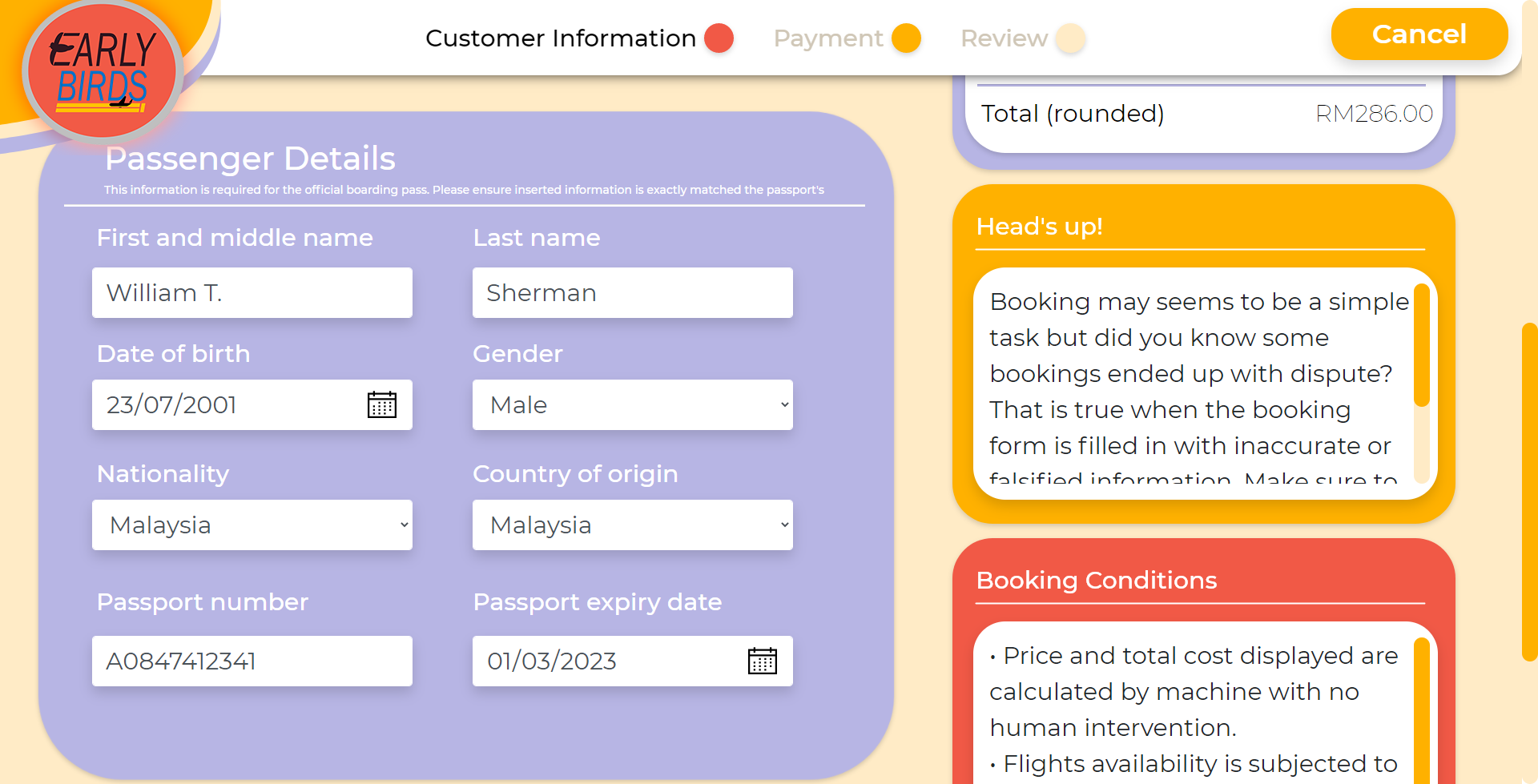
**Figure 62** displays the search result after filter function is applied



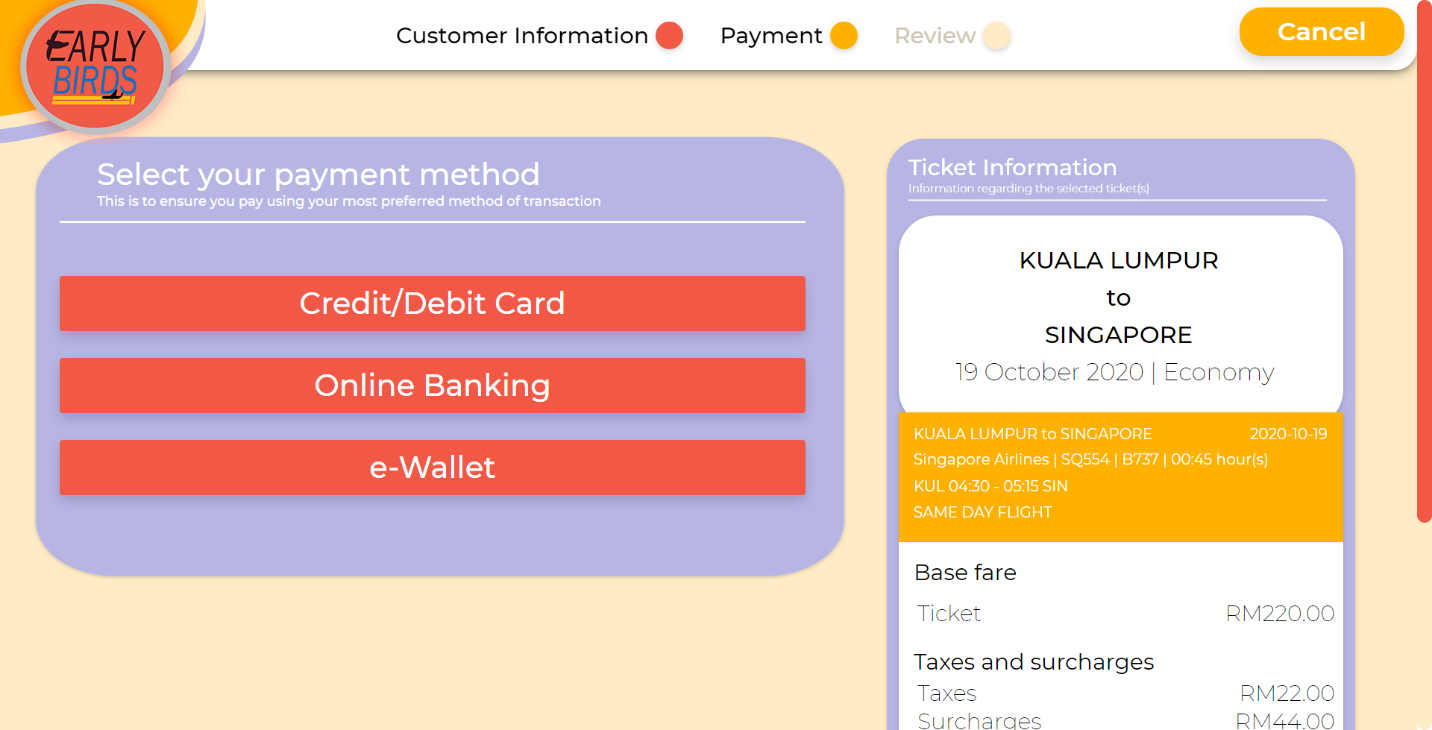
**Figure 63** is the booking form along with flight information displayed



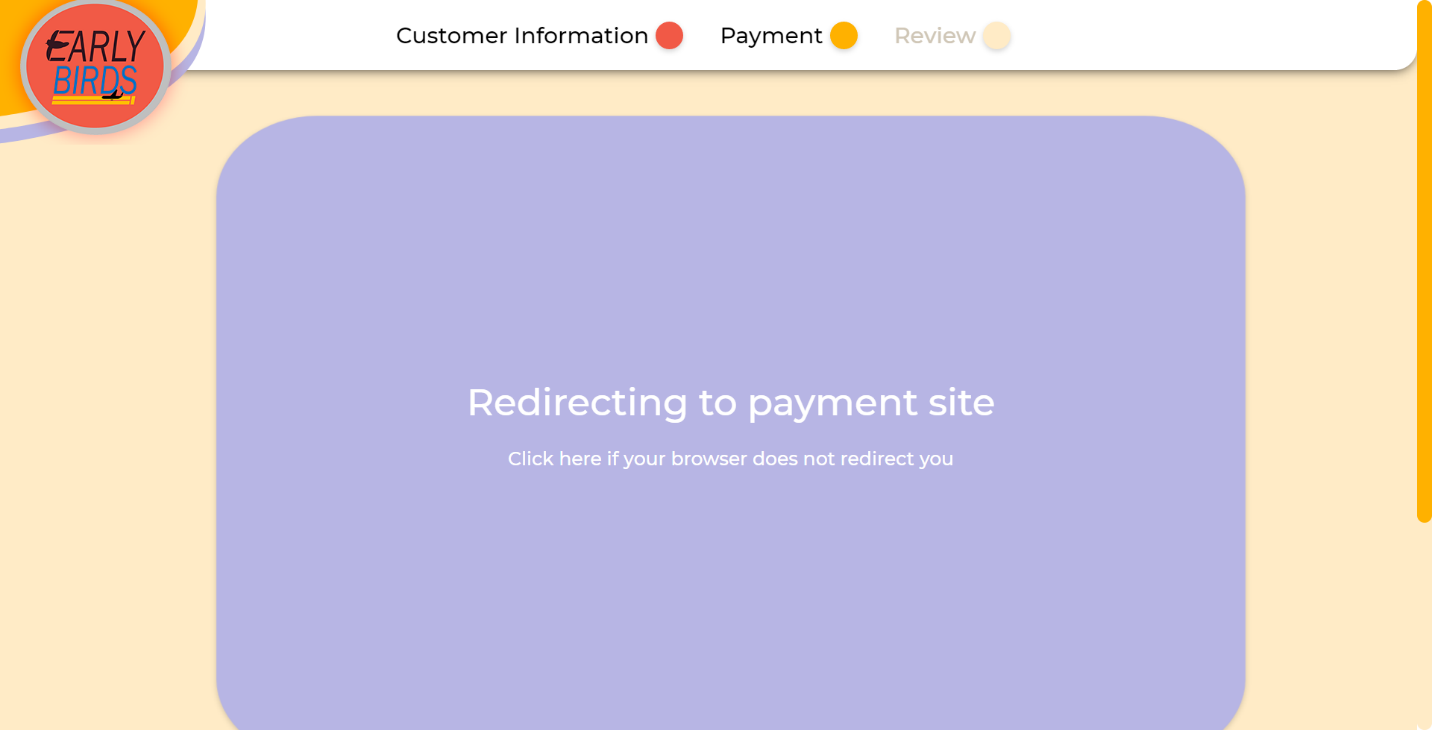
**Figure 64** is the initial look of the passenger details form a user must fill in before proceeding



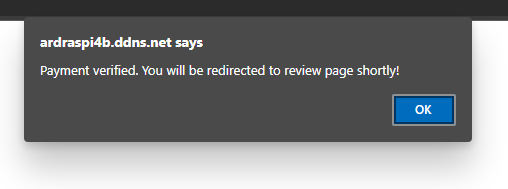
**Figure 65** is the passenger details form after being filled with required information



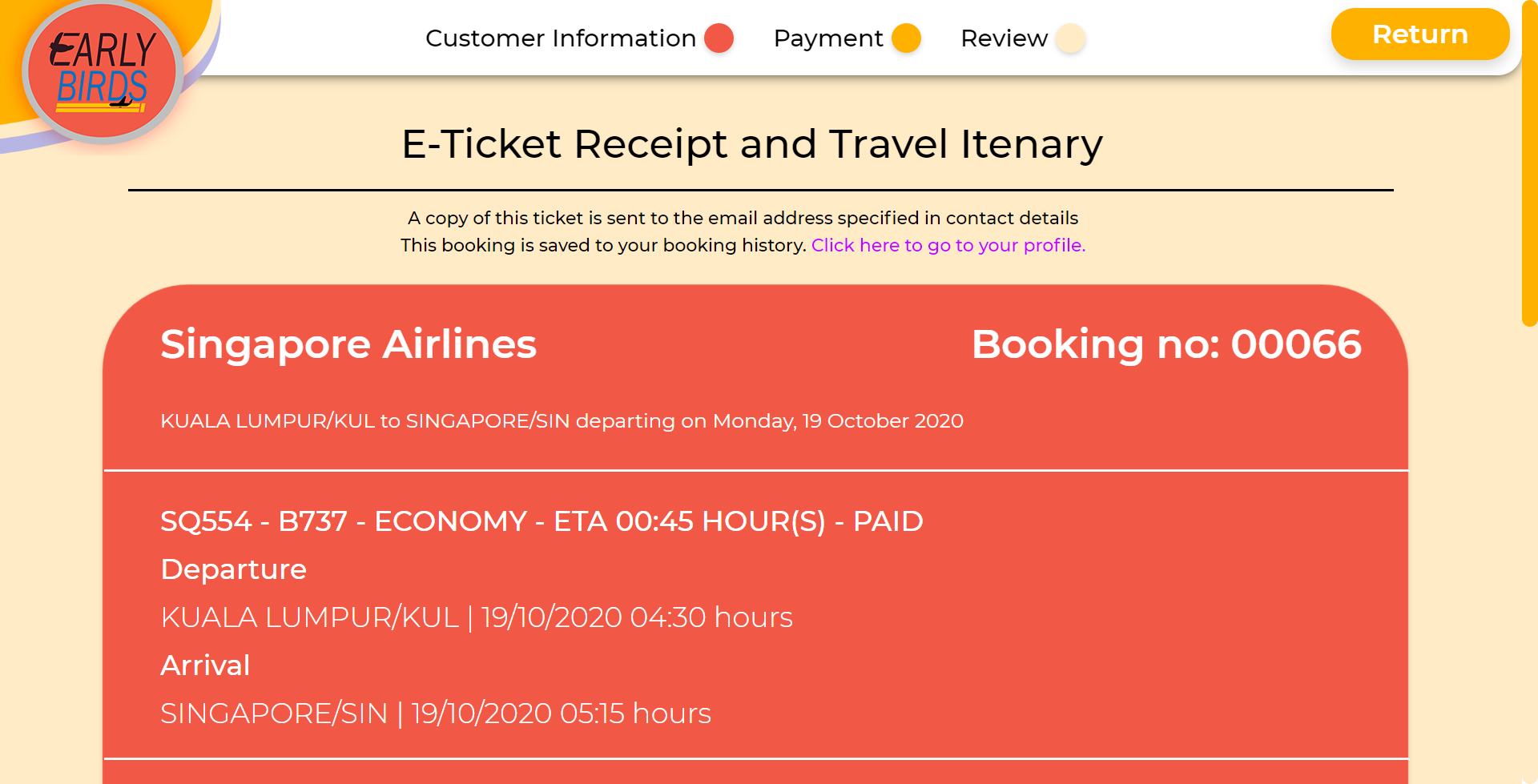
**Figure 66** shows the transaction method available for the user to choose



**Figure 67** is a page handling redirection of user to secured payment gateway



**Figure 68** shows popup notification of a successful payment

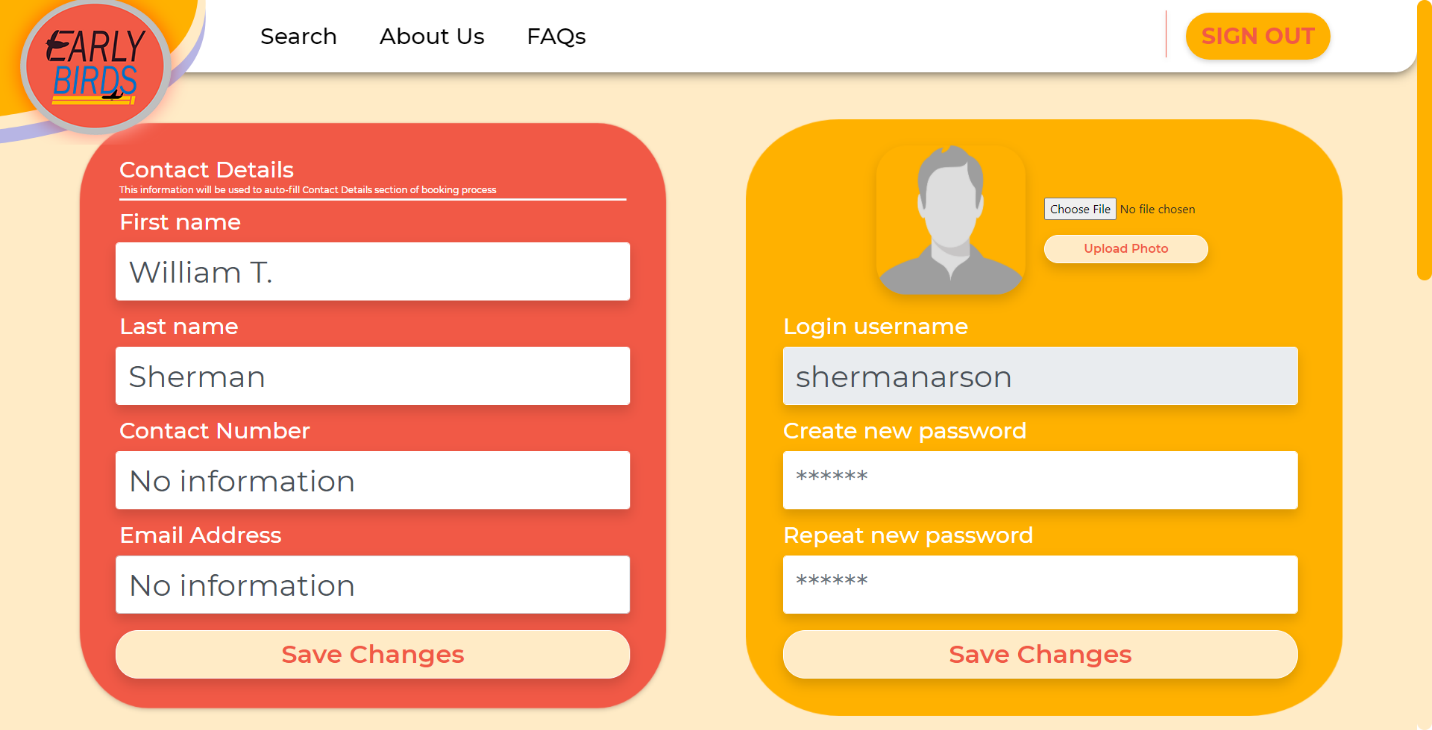


**Figure 69** is the user review page with the booking information listed

After the payment is complete, the user will be redirected to this review page to read and acknowledge the booking that has been placed. A copy of this booking information is saved inside the user’s booking history section under profile.

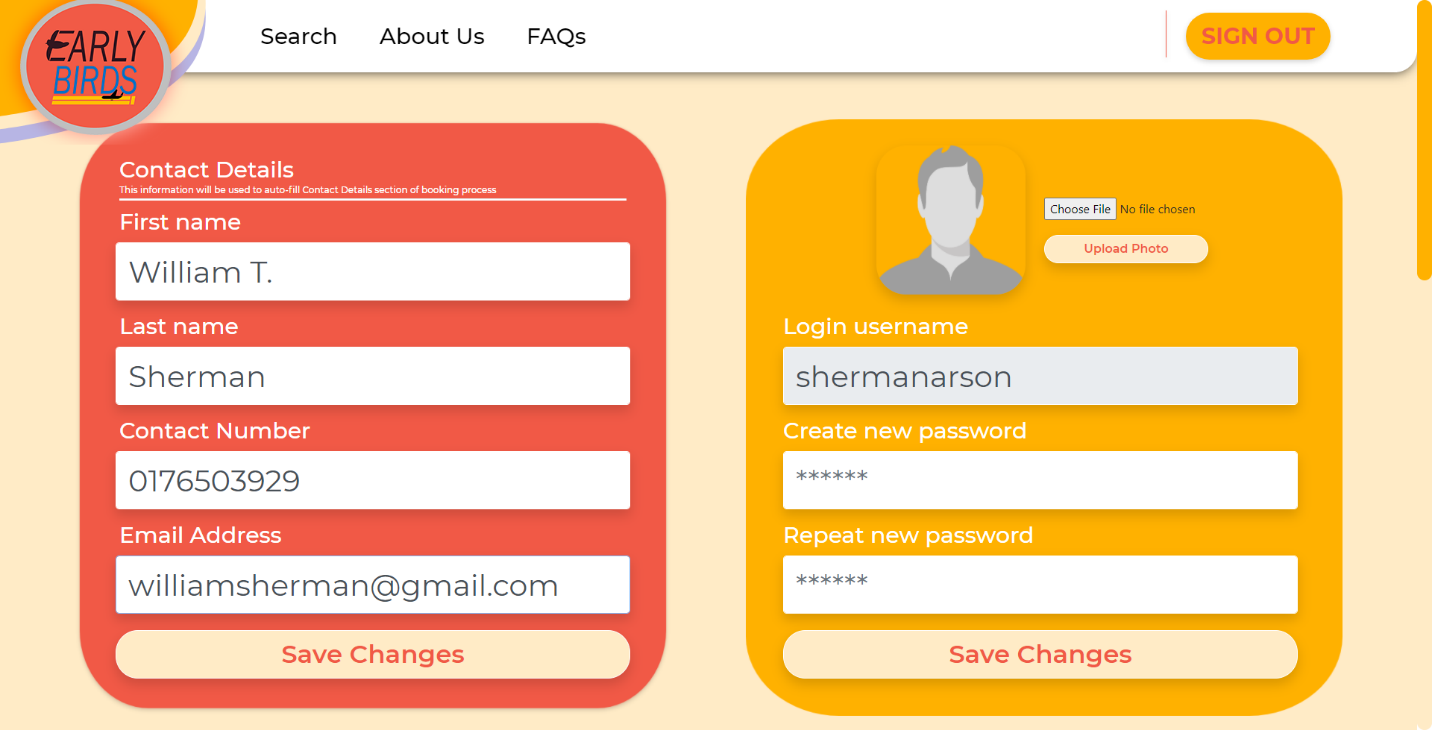
### Editing Profile

One of the many features Earlybirds provides to the users is to allow contact details and credentials update as well as the upload of their own profile picture. Contact details are used to feed information automatically when the user is booking for a flight. For credentials, password is easily forgotten or becoming less secure and therefore, it is a handy feature to implement that allows for the user to set once they feel like so. Profile picture, on the other hand, is an aesthetic addition to the whole point of having a profile so that the user can present themselves in a different way.

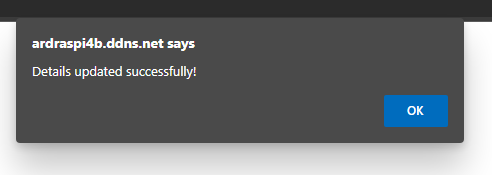


**Figure 70** is the user profile page in its initial form without contact number nor email address set

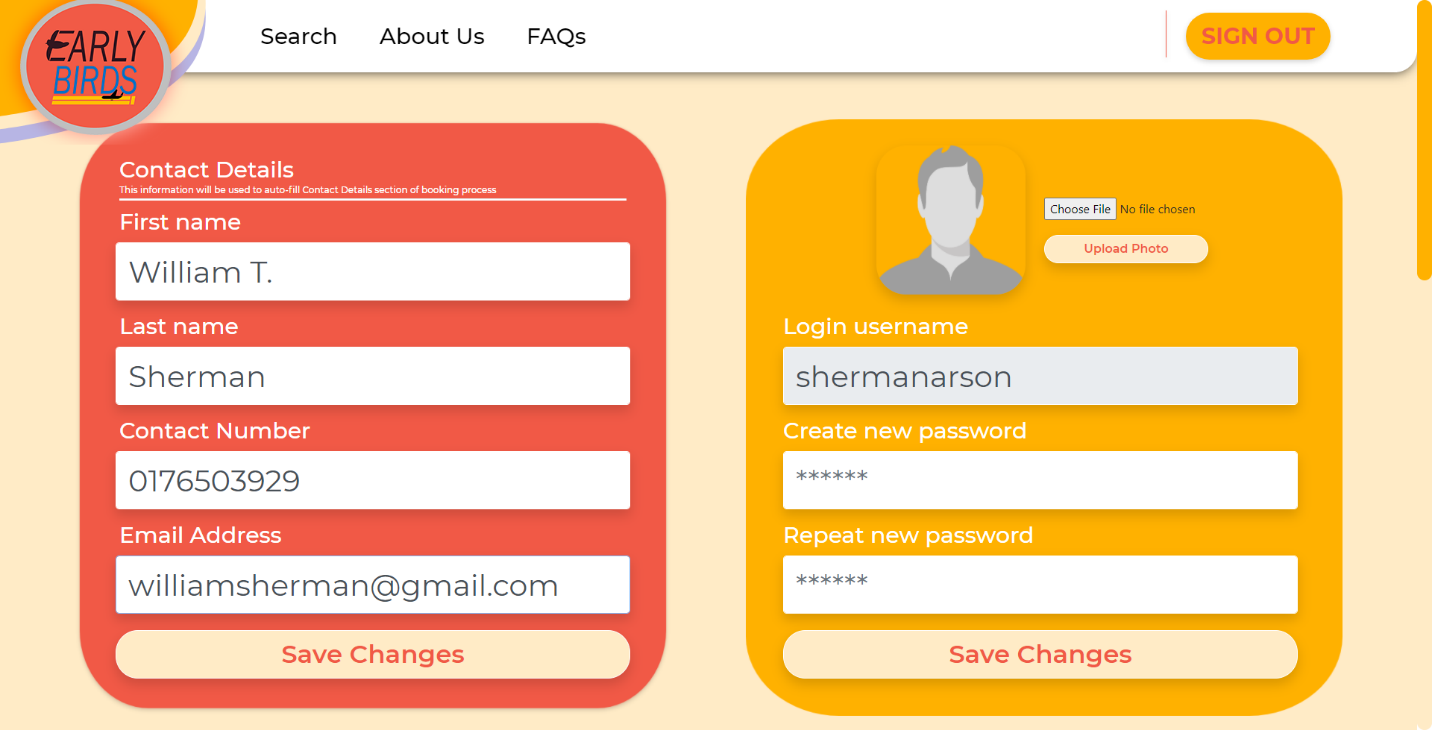
The grey-coloured words found inside the input fields are placeholders to hint users of the input format and structure. The form will not allow for any empty field and if there is any, there will be tooltips notifying of fields left empty and the user should fill it at once. Black-coloured fonts, however, are not placeholders but actual data fetched from the user record inside the database. No information in the columns signify the unavailability of such data inside the database and it is high suggested for users to complete it.



**Figure 71** shows the insertion of contact number and email address by the user



**Figure 72** is a popup notification pushed to notify the user of successful details update

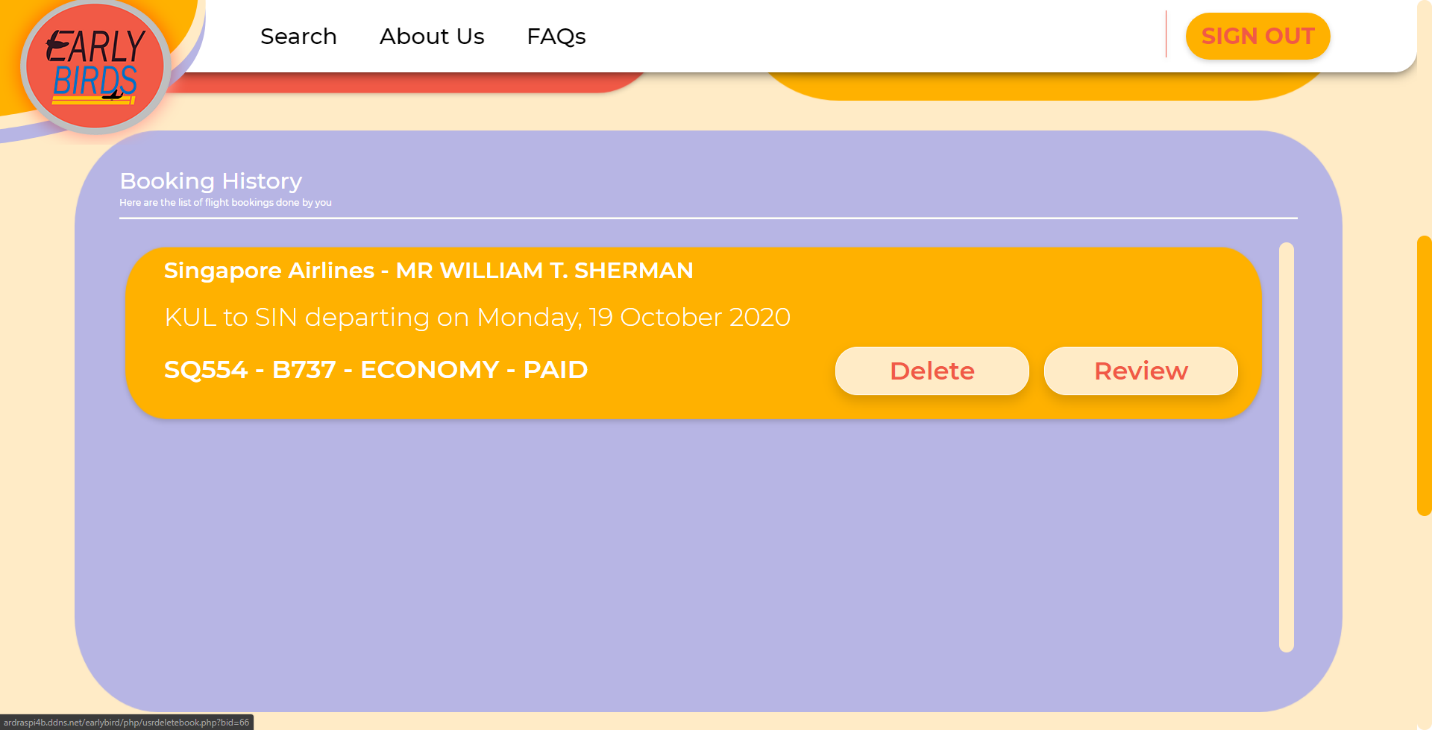


**Figure 73** is the profile page with updated contact number and email address displaying

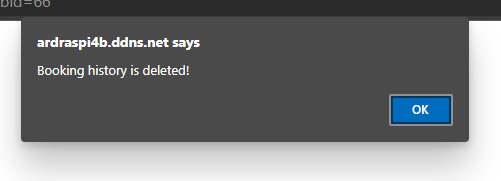
As seen in the screenshot above, the contact number and email address fields are filled in with new data fetched from the recently updated user record. The same action is applied to when updating credentials and profile picture.

### Deleting Booking History

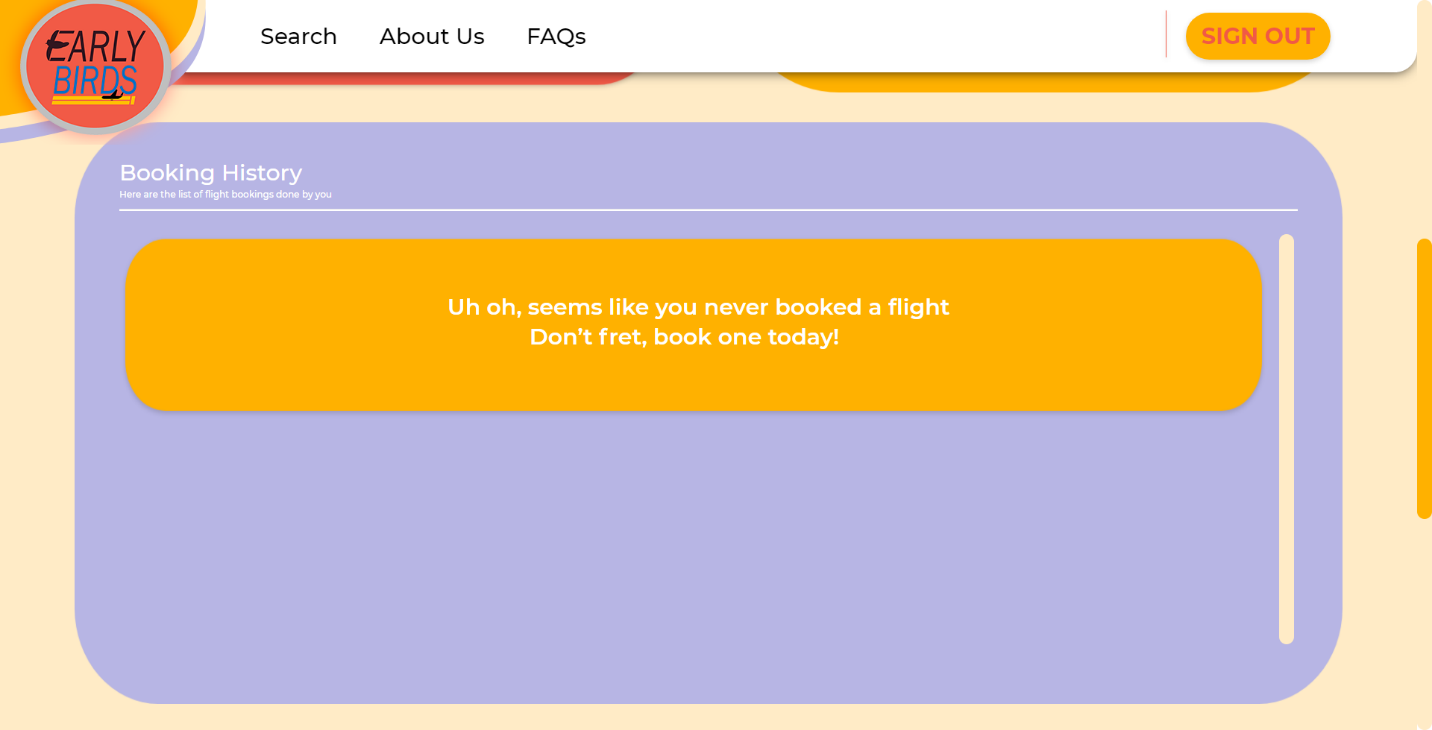
Sometimes, sentimental values are endearing and should be treasured the most. However, if these sentimental values start to pile up and clog the view, it is better to remove or throw some of them away for cleanliness and hygiene purpose. For that reason, users of Earlybirds are able to delete their booked flights history. This feature is very straightforward as the user is only required to press the delete button and the particular record will be gone both from the user profile and database.



**Figure 74** is the initial look of booking history section found inside user profile page



**Figure 75** is a popup notification of successful booking history deletion



**Figure 76** is the booking history section of the user after the last booking history deletion

Since this account does not have any more bookings after the deleted one, an empty booking placeholder appears and encouraging the user to book for a flight again. Now the basic functionalities of the website are covered, and users can use the system without a problem.

# References

Bootstrap Studio, *A powerful desktop app for creating responsive websites using the Bootstrap framework.* [Online] Available at: <https://bootstrapstudio.io/> [Accessed 23 October 2020].

Charles, 2011. *How can i generate a 6 digit unique number?* [Online] Available at: <https://stackoverflow.com/questions/5464906/how-can-i-generate-a-6-digit-unique-number>

[Accessed 23 October 2020].

W3Schools, *PHP preg\_match() Function.* [Online] Available at: <https://www.w3schools.com/php/func_regex_preg_match.asp> [Accessed 23 October 2020].

php.net, *date\_default\_timezone\_set.* [Online] Available at: <https://www.php.net/manual/en/function.date-default-timezone-set.php> [Accessed 23 October 2020].

W3Schools, *PHP switch Statement.* [Online] Available at: <https://www.w3schools.com/php/php_switch.asp> [Accessed 23 October 2020].

php.net, *The DateTime class.* [Online] Available at: <https://www.php.net/manual/en/class.datetime.php> [Accessed 23 October 2020].

W3Schools, *PHP mysqli real\_escape\_string() Function.* [Online] Available at: <https://www.w3schools.com/php/func_mysqli_real_escape_string.asp> [Accessed 23 October 2020].

W3Schools, *HTML <meta> http-equiv Attribute.* [Online] Available at: <https://www.w3schools.com/tags/att_meta_http_equiv.asp> [Accessed 23 October 2020].

W3Schools, *PHP date() Function.* [Online] Available at: <https://www.w3schools.com/php/func_date_date.asp> [Accessed 23 October 2020].

W3Schools, *PHP strtotime() Function.* [Online] Available at: <https://www.w3schools.com/php/func_date_strtotime.asp> [Accessed 23 October 2020].

# Appendices

## Workload Matrix

**INTAKE CODE: UCDF1905ICT(SE)**

**GROUP NUMBER: 4**

**LECTURER: MR SYED MOHD ZAHID BIN SYED ZAINAL ARRIFIN**

**WEBSITE NAME: EARLYBIRDS**

|  |  |  |
| --- | --- | --- |
| **Name** | **Task / Description / Responsibility** | **Signature** |
| **MUHAMMAD FUAD BIN ABDULLAH, TP055684**  (**Group Leader**) | **Documentation Components:**  *Wireframe - User, Navigational Structure, Activity Diagram, Implementation (All), User Manual (All)*  **Coding Components:**  *User Login, User Registration, User Profile, User Search, Filter Function, User Booking, Administrator Login, Administrator Registration, Administrator Register Airport Manager, Administrator Edit User and Airport Manager, Administrator Delete User and Airport Manager,* |  |
| **LAI RONG XIANG, TP056196** | **Documentation Components:**  *Gantt Chart, Introduction, Wireframe – Airport Manager and Administrator, Entity Relationship Diagram*  **Coding Components:**  *Airport Manager Login, Airport Manager Add Flight to Schedule, Airport Manager Edit Flight from Schedule, Airport Manager Delete Flight from Schedule.* |  |
| **HASSAN ABBAS, TP055184** | **Documentation Components:**  *N/A*  **Coding Components:**  *N/A* | **A picture containing snow, covered, group, standing  Description automatically generated** |

## Peer-to-Peer Evaluation Form

**WDT Peer-to-Peer Evaluation Form**

Directions: In the space below, honestly evaluate the work of other students in your group by answering yes or no and by using a scale from 1 to 3, 1 = poor, 2 = average, 3 = good.

**Evaluator’s Name: MUHAMMAD FUAD BIN ABDULLAH** **Date: 24/10/2020**

|  |  |  |
| --- | --- | --- |
|  | ***LAI RONG XIANG*** | ***HASSAN ABBAS*** |
| ***Did this group member complete his/her assigned tasks for the group?*** | **YES NO** | **YES NO** |
| ***How would you rate the quality of this person’s work?*** | **1 2 3** | **1 2 3** |
| ***How would you rate the timeliness of the completion of the work?*** | **1 2 3** | **1 2 3** |
| ***How would you rate the accuracy of the work?*** | **1 2 3** | **1 2 3** |
| ***Overall, how would you rank this group member’s performance in the group?*** | **1 2 3** | **1 2 3** |

**Would you want to work with your current group members again? YES NO**

**Explain why in the space given below.**

*Due to several circumstances, one of our teammates was unable to contribute to the assignment which forces the other two members including myself to toil ourselves to the max as to ensure the completion of this assignment. I personally would like to have a teammate with better communication skills and are able to work without relying on my commands or instructions too much as that will obstruct growth and independence*

**This peer evaluation is private and confidential.**

**WDT Peer-to-Peer Evaluation Form**

Directions: In the space below, honestly evaluate the work of other students in your group by answering yes or no and by using a scale from 1 to 3, 1 = poor, 2 = average, 3 = good.

**Evaluator’s Name: LAI RONG XIANG** **Date: 24/10/2020**

|  |  |  |
| --- | --- | --- |
|  | ***MUHAMMAD FUAD BIN ABDULLAH*** | ***HASSAN ABBAS*** |
| ***Did this group member complete his/her assigned tasks for the group?*** | **YES NO** | **YES NO** |
| ***How would you rate the quality of this person’s work?*** | **1 2 3** | **1 2 3** |
| ***How would you rate the timeliness of the completion of the work?*** | **1 2 3** | **1 2 3** |
| ***How would you rate the accuracy of the work?*** | **1 2 3** | **1 2 3** |
| ***Overall, how would you rank this group member’s performance in the group?*** | **1 2 3** | **1 2 3** |

**Would you want to work with your current group members again? YES NO**

**Explain why in the space given below.**

*There has been some unexpected inconvenience along the way in the assignment as one of the members within our group is not able to contribute for the assignment, causing the remaining members, including me to carry on more workload in order to complete the assignment in time. I also would prefer to have a team member with good communication etiquette to ensure that there is progress for the assignment,*

**This peer evaluation is private and confidential.**