**FlightPub**

**Design Model Document**

**Team No: 4**

**Team Members:**

**Tobias Colson**

**Samuel Brackenrig**

**Adam Fleming**

**Benjamin Collins**

**Jordan Cork**

**Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Authors** | **Summary of Changes** |
| 0.8 | 6 Jun 18 | Jordan Cork | Booking and Searching SubSystems |
| 0.9 | 7 Jun 18 | Jordan Cork | Searching and Review SubSystems |
| 1.0 | 8 Jun 18 | Jordan Cork | Review and User Management SubSystems, Formatting |
|  |  |  |  |

**Document Sign-Off**

|  |  |  |
| --- | --- | --- |
| **Name (Position)** | **Signature** | **Date** |
| Benjamin Collins | ~ | 08/05/18 |
| Tobias Colson | ~ | 08/05/18 |
| Adam Fleming | ~ | 08/05/18 |
| Jordan Cork | ~ | 08/05/18 |
| Samuel Brackenrig | ~ | 08/05/18 |

**Contents**

**SEARCHING SUBSYSTEM 6**

**1 SEARCHING SUBSYSTEM SPECIFICATION 6**

1.1 Role 6

1.2 Key Scenarios 6

1.3 Graphical Use Case Diagram 7

**2 SEARCHING SUBSYSTEM DESIGN 7**

2.1 Class Diagram 7

2.2 Class Descriptions 8

2.2.1 Class Name: <Search.jsp> 8

2.2.2 Class Name: <SearchController.java> 9

2.2.3 Class Name: <Results.java> 10

2.2.4 Class Name: <ReturnResults.java> 11

2.2.5 Class Name: <Flight.java> 11

2.3 Dynamics 11

**BOOKING SUBSYSTEM 12**

**1 BOOKING SUBSYSTEM SPECIFICATION 12**

1.1 Role 12

1.2 Key Scenarios 12

1.3 Graphical Use Case Diagram 12

**2 BOOKING SUBSYSTEM DESIGN 13**

2.1 Class Diagram 13

2.2 Class Descriptions 13

2.2.1 Class Name: <FlightSummary.jsp> 13

2.2.2 Class Name: <Booking.jsp> 14

2.2.3 Class Name: <BookingController.java> 15

2.2.4 Class Name: <Booking.java> 15

2.3 Dynamics 15

**REVIEW SUBSYSTEM 16**

**1 REVIEW SUBSYSTEM SPECIFICATION 16**

1.1 Role 16

1.2 Key Scenarios 16

1.3 Graphical Use Case Diagram 17

**2 REVIEW SUBSYSTEM DESIGN 17**

2.1 Class Diagram 17

2.2 Class Descriptions 17

2.2.1 Class Name: <Review.jsp> 17

2.2.2 Class Name: <ReviewController.java> 18

2.2.3 Class Name: <Review.java> 18

2.3 Dynamics 19

**USER MANAGEMENT SUBSYSTEM 19**

**1 USER MANAGEMENT SUBSYSTEM SPECIFICATION 19**

1.1 Role 19

1.2 Key Scenarios 19

1.3 Graphical Use Case Diagram 20

**2 USER MANAGEMENT SUBSYSTEM DESIGN 20**

2.1 Class Diagram 20

2.2 Class Descriptions 21

2.2.1 Class Name: <Login.jsp> 21

2.2.2 Class Name: <Register.jsp> 21

2.2.3 Class Name: <UserManagementController.java> 22

2.2.4 Class Name: <User.java> 22

2.3 Dynamics 23

# Searching Subsystem

# Searching Subsystem Specification

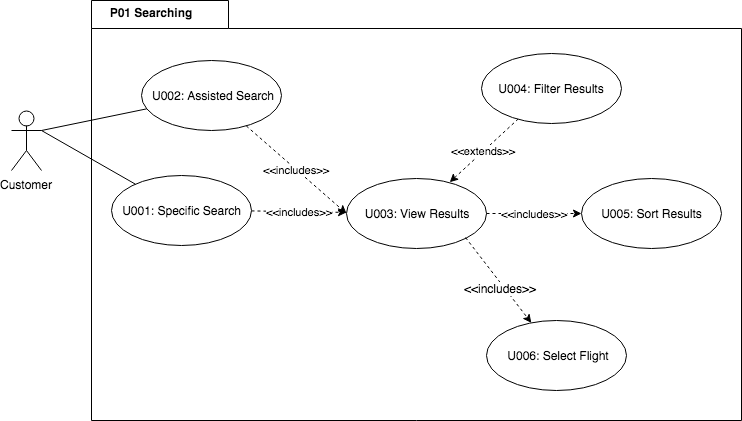
## Role

The Searching Subsystem is responsible for carrying out the entire flight search process from displaying the search menu, taking user input, retrieving results and displaying the results. Also handles filtering and sorting of those results.

## Scenarios

* *Assisted Search*: Display Assisted Search form and take in user input. Validate input in Assisted search form then retrieve the search results based on the search parameters. If return flight was selected, then retrieve return flights from Database matching the search parameters with opposite airports and using return date
* *Specific Search*: Display Specific Search form and take in user input. Validate input in Specific search form then retrieve the search results based on the search parameters. If return flight was selected, then retrieve return flights from Database matching the search parameters with opposite airports and using return date
* *View Results*: Display the list of flights all relevant information as well as a ‘Select Flight’ button
* *Sort Results*: Sort the flight results based on lowest price, highest price, lowest flight time, highest flight time, rating, or User Categories: business, high class, family, budget, leisure or standard
* *Filter Results*: Filter flight results based on number of stops, price range, take off/arrival time, extras, cabin choices
* *Select Flights*: Allow Customer to select a departure flight and/or a return flight and store flight(s) in the system and send selected flights further for booking.

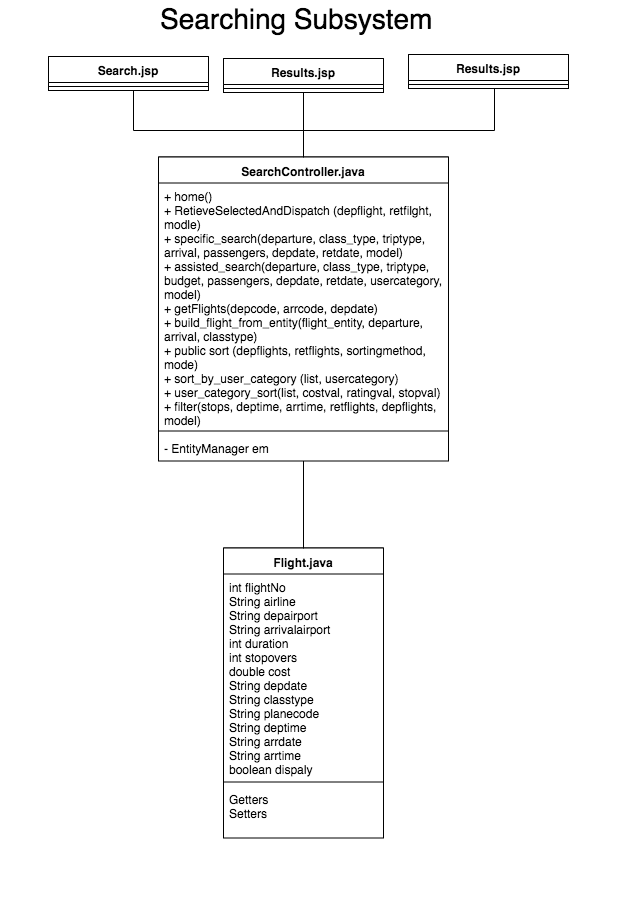
## Use Case Diagram



## Searching Subsystem Design

While the Searching Subsystem was not fully implemented in the prototype, the following design specifications provide a solid overview of how it will be implemented in the final delivery. The implementation has been written, but is not fully functional.

## 2.1 Class Diagram

**

## 2.2 Class Descriptions

### 2.2.1 Class Name: <Search.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Display Specific Search Form with Dep. Airport (txt), Arr. Airport (txt), Date(s) (txt), Return or One Way (radio selection), passengers (drop menu), class (drop menu) |  |
| Display Assisted Search Form with Dep. Airport (txt), Continent (drop menu), Date(s) (txt), Return or One Way (radio selection), passengers (drop menu), class (drop menu) user category (drop menu), budget (txt) |  |
| Form validation on Specific Search | flightpub.js |
| Form validation on Assisted Search | flightpub.js |
| Submit Specific Search Form parameters to Controller | SearchController.java |
| Submit Assisted Search Form parameters to Controller | SearchController.java |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp (UserManagement Subsystem) |

### 

### 2.2.2 Class Name: <SearchController.java>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve Specific Search parameters from Specific Search Form | Search.jsp |
| Perform Query on FlightEntity using the specific search parameters | FlightEntity.java |
| For Specific. If return was selected perform Query with return date and switched destination and arrival airports | Flight.java |
| Retrieve Assisted Search parameters from Assisted Search Form | Search.jsp |
| Perform Query on FlightEntity using the assisted search parameters | FlightEntity.java |
| For Assisted. If return was selected perform query on FlightEntity.java with return date and switched destination and continent | FlightEntity.java |
| Perform initial sort of search results based on selected User Category or default Standard Category |  |
| Sort Flight Results list based on sort parameter received from View | Results.jsp/ ReturnResults.jsp |
| Filter Flight Results list based on sort parameter received from View. Use display boolean attribute in Flight.java for filtering | Results.jsp/ ReturnResults.jsp |
| Send sorted list of flights to Results page | Results.jsp/ ReturnResults.jsp |
| Send filtered list of flights to Results page | Results.jsp/ ReturnResults.jsp |
| Retrieve selected flight(s) from Results and send them to the Flight Summary page for displaying | Results.jsp/ ReturnResults.jsp  Booking.jsp (Booking Subsystem) |

### 

### 2.2.3 Class Name: <Results.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve Flight list from Controller | SearchController.java |
| Display Departure Flight list (initially) |  |
| Display Return Flight list (after departure flight selected) |  |
| Display top 50 search results (list) in current order with Airline, airports, duration, stops, cost and ‘Select’ button |  |
| Display Filter options as checkboxes and an ‘Update Results’ button. Options are number of stops, price range, take off/arrival time, extras, cabin choices |  |
| Send Filter options and flight results list to Controller once update button is pressed | SearchController.java |
| Display a Sort drop down menu with the following options: lowest price, highest price, lowest flight time, highest flight time, rating, or User Categories: business, high class, family, budget, leisure or standard |  |
| Send selected Sort parameter to Controller with flight list when new option is selected | SearchController.java |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp (UserManagement Subsystem) |
| When flight is selected, send back to Controller | SearchController.java |
| Display ‘Back’ button and return to previous page when selected |  |

### 

### 2.2.4 Class Name: <Return.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve Flight list from Controller | SearchController.java |
| Display Departure Flight list (initially) |  |
| Display Return Flight list (after departure flight selected) |  |
| Display top 50 search results (list) in current order with Airline, airports, duration, stops, cost and ‘Select’ button |  |
| Display Filter options as checkboxes and an ‘Update Results’ button. Options are number of stops, price range, take off/arrival time, extras, cabin choices |  |
| Send Filter options and flight results list to Controller once update button is pressed | SearchController.java |
| Display a Sort drop down menu with the following options: lowest price, highest price, lowest flight time, highest flight time, rating, or User Categories: business, high class, family, budget, leisure or standard |  |
| Send selected Sort parameter to Controller with flight list when new option is selected | SearchController.java |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp (UserManagement Subsystem) |
| When flight is selected, send back to Controller | SearchController.java |
| Display ‘Back’ button and return to previous page when selected |  |

### 2.2.5 Class Name: <Flight.java>

**Superclass Name:**  None

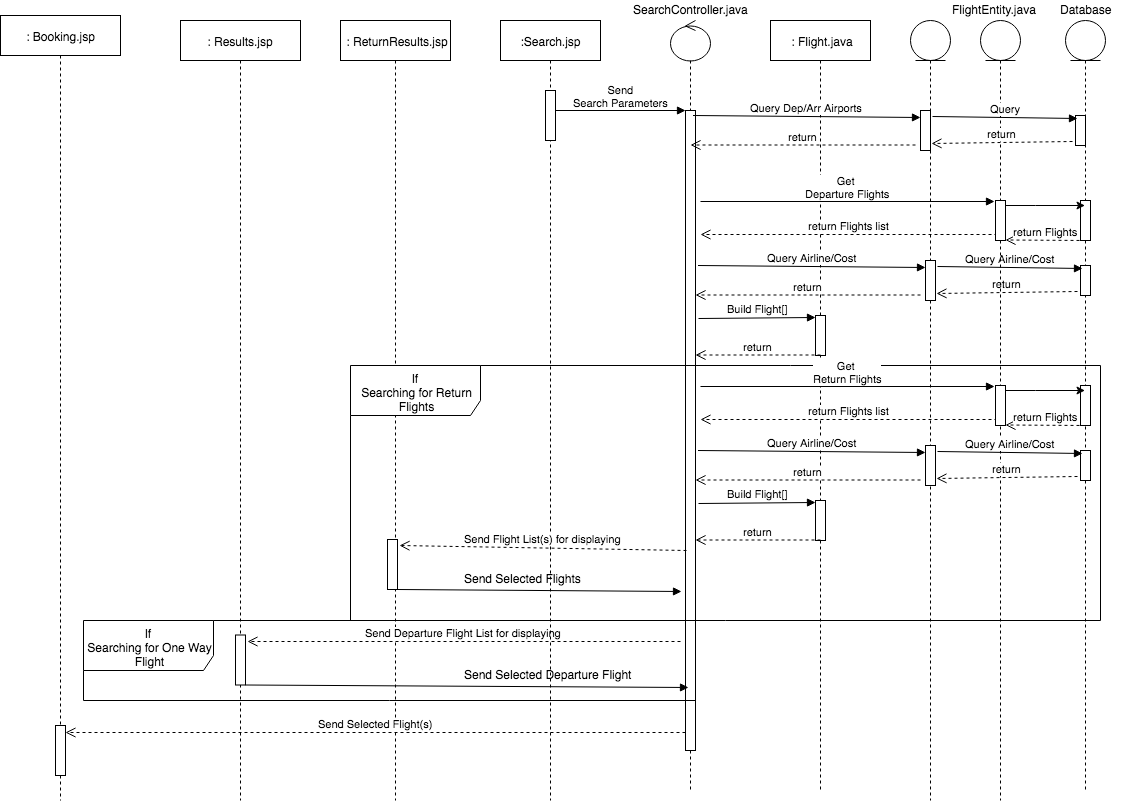
**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Store all relevent parameters for use in displaying flight data and for booking |  |
| When departing flight is selected, send to controller and display return flights | SearchController.java |
| When return flight is selected, progress to Booking.jsp | SearchController.java  BookingController.java |

## 2.3 Dynamics

* *Form Validation*: Specific or Assisted Search form (depending on which is selected) will be sent to flightpub.js for validation and flightpub.js will send back a validation confirmation or failure
* *Search Process*: Once validation is completed, the form data (Specific or Assisted) is sent to SearchController.java. Controller then Queries database through Entities for Airport Codes. Controller then performs Query through the FlightEntity Bean. Query retrieves flights based on the search parameters. Controller then retrieves addtional information on each flight. This is done through querying the Aircraft and Price Entities which access the database and return the relevent information. This infomation is used to create a list Flight.java objects. If Return was selected in the search then the process is repeated for the return flights. Each of theses lists is then sorted based on User Category. If only one way then send to Results.jsp to be displayed. If return flights then send the two lists to ReturnResults.jsp. Once the flight or flights have been selected, the selected flight(s) are sent back to the Controller. Once the flight(s) has/have been selected the flight(s) is/are then sent to the FlightSummary.jsp page for viewing.

*Search Process*



* *Sort*: When a Sorting method is selected on the Results.jsp/ReturnResults.jsp pages, the method (lowest price, highest price, lowest flight time, highest flight time, or User Categories: business, high class, family, budget, leisure or standard) and the flight list (or lists if return flights are being selected) is/are sent to the Controller. The controller then sorts the list and returns it back to the Results.jsp/ReturnResults.jsp pages for displaying.

* *Filter*: When ‘Update Results’ is selected on Results.jsp/ReturnResult.jsp pages the Filter parameters and the list(s) is/are sent to the Controller. The Controller then filters the list(s) based on the parameters and returns filtered list to the Results.jsp page for displaying Booking SubSystem

# Subsystem Specification

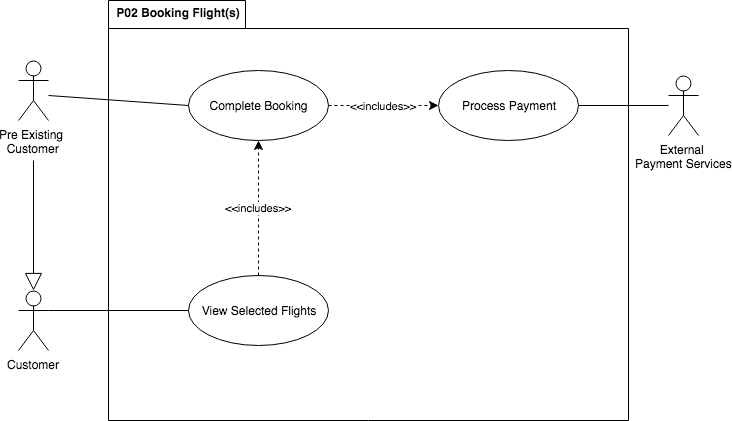
## Role

The booking subsystem is responsible for handling the entire process of booking a flight or flights, from viewing Search results to booking confirmation.

## Scenarios

* *View Selected Flights*: Retrieve selected flights from Searching subsystem and show customer the list of their selected flights or flight as well as total cost
* *Complete Booking:* Process involves retrieving personal information from a customer. If user is logged in, retrieve stored personal information from logged in user and populate personal information form. Validate personal information form and display error message for incorrect form details
* *Process Payment*: Involves carrying out dummy payment procedure (i.e. third party payment) then storing Booking information in database (Flight details and customer information). Finally display a confirmation message to customer after flight booking has been confirmed

## Use Case Diagram

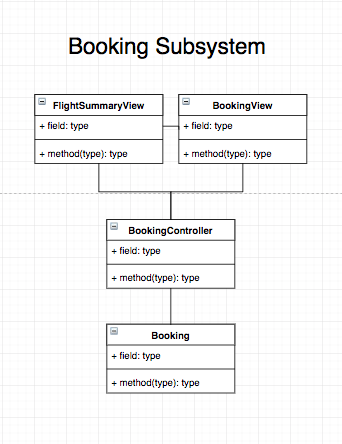
*.*

## Booking Subsystem Design

While the Booking Subsystem was not fully implemented in the prototype, the following design specifications provide a solid overview of how it will be implemented in the final delivery.

## 2.1 Class Diagram

*Unfinished*

**

## 2.2 Class Descriptions

### 2.2.1 Class Name: <FlightSummary.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve flight(s) selected by customer in the flight results page | SearchController.java  (Searching Subsystem) |
| Display Departure Flight information i.e. Date, Airline, Dep. Airport, Arr. Airport, Stopovers, Duration |  |
| If a return flight was selected display Flight information i.e. Date, Airline, Dep. Airport, Arr. Airport, Stopovers, Duration |  |
| Display total cost of flight(s) |  |
| Display ‘Login’ button and forward to ‘Login’ page when selected | UserManagementController.java (UserManagement Subsystem) |
| Display ‘Confirm Booking’ button and pass flight(s) information to Booking View selected | Booking.jsp |
| Display ‘Choose Different Flight(s)’ and return to Results View when selected | Results.jsp (Searching Subsystem) |
| Display ‘Home’ button and forward to “Search” page when selected | Search.jsp (Searching Subsystem) |

### 2.2.2 Class Name: <Booking.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Display Personal Details Form incl. Title, FName, MName, LName, Phone No, DOB, Address |  |
| Retrieve Personal Details from User Management Subsystem if user is logged in | UserManagementController.java (User Management SubSystem) |
| If personal details have been retrieved, populate Form |  |
| Display Payment Radio Selection, Credit Card and Paypal |  |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp (UserManagement Subsystem) |
| Display ‘Choose Different Flight(s)’ and return to Results View when selected | Results.jsp (Searching Subsystem) |
| Display ‘Proceed to Payment’ Button and validate form data when selected | flightpub.js |
| Display error message if form validation fails |  |
| Pass Flight and Customer details to Controller for payment processing | BookingController.java |
| Display Confirmation message when payment confirmation has been received | BookingController.java |
| Display ‘Return’ button. Return to Search Page when selected | Search.jsp  (Searching Subsystem) |
| Display ‘Home’ button and forward to Searching page when selected | Search.jsp (Searching Subsystem) |

### 

### 2.2.3 Class Name: <BookingController.java>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve booking information from Booking View | Booking.jsp |
| Dummy Payment confirmation |  |
| Store Booking Information in Booking bean | Booking.java |
| Update Flight(s) with number of seats taken by booking | Flight.java |
| Send confirmation and Review email to customers email address |  |
| Send confirmation back to Booking View | Booking.jsp |

## 

### 2.2.4 Class Name: <Booking.java>

**Superclass Name:**  None

**Subclass Name:**  None

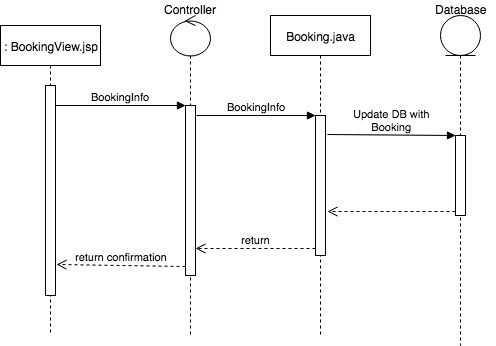
|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Update Bookings Table in DB with new booking information | Database |
| Retrieve Booking information based on User account information and pass back to UserManagement Controller | UserManagementController.java |

## 

## 2.3 Dynamics

* *Selection*: Selected Flights are passed from the Searching Controller (Searching Subsystem) to FlightSummary.jsp in order to be displayed and potentially passed on further
* *Retrieve Personal Information*: Before displaying the personal information form, Booking.jsp will check if user is logged in. If so it will retrieve the user bean from the UserManagementController.java which contains the Personal information
* *Form Validation*: Personal information form will be sent to FormValidation.js for validation and FormValidation.js will send back a validation confirmation or failure
* *Store Booking*: Booking.jsp will send the booking information i.e. flight info and customer info to the BookingController.java. The controller will then send the information to the Booking.java bean. This bean will then store in the database. Once this is done, the Controller will return a confirmation message to Booking.jsp which will then display the confirmation

*Store Booking*



# Review Subsystem

# Review Subsystem Specification

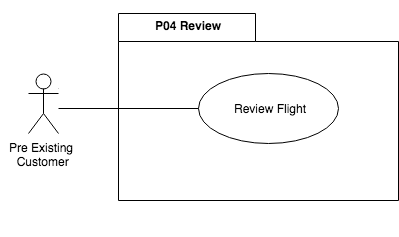
## Role

The Review Subsystem is responsible for handling Customer reviews of previously booked flights. These reviews are accessed from links (one per flight) in the confirmation email sent to the Customer.

## Scenarios

* *Review Flight:*
  + *Review Form:* Display the review form for the flight including Star Rating and Additional Feedback
  + *Validate Form:* Validate the User input for the Review form and return validation confirmation or failure
  + *Store Review:* Store the customers review in the database to be used in calculating the final Rating for a flight
  + *Update Rating:* Update the User Rating attributed to Airline / Aircraft combination, used to provide recommendations to users

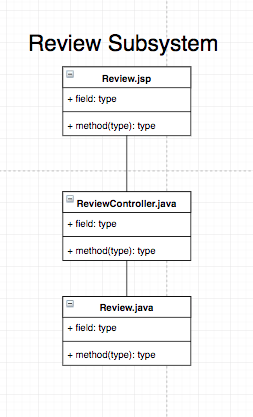
## Use Case Diagram

**

## 

## 2. Subsystem Design

## 2.1 Class Diagram

**

## 2.2 Class Descriptions

### 2.2.1 Class Name: <Review.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve Flight Data necessary for storing Review from Email link | Email |
| Display Review Form including Rating out of 5 Radio selection, and Comment Box |  |
| Submit form for Validation | flightpub.js |
| Display Error message for failed form validation |  |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp (UserManagement Subsystem) |
| If logged in Display ‘Account’ button and go to ‘Account’ page when selected | Account.jsp  UserManagementController.java |
| Display ‘Home’ button and forward to Searching page when selected | Search.jsp (Searching Subsystem) |
| Send form data to Controller | ReviewController.java |
| Retrieve confirmation or failure from Controller | ReviewController.java |
| Display confirmation or failure message |  |

### 2.2.2 Class Name: <ReviewController.java>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve Form data and flight details from Review.jsp | Review.jsp |
| Process form data and send to Review.java to store in database | Review.java |
| Update Avg review score for the Aircraft/Airline combination | Review.java |
| Return confirmation message to Review.jsp | Review.jsp |

## 

### 2.2.3 Class Name: <Review.java>

**Superclass Name:**  None

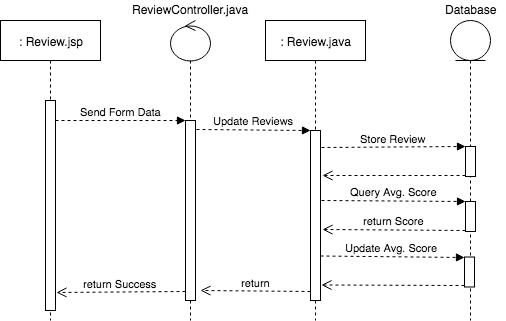
**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Store Customer Review in database | Database |
| Retrieve Avg review score for the Aircraft/Airline combination | Database |
| Update Avg review score for the Aircraft/Airline combination with new avg. | Database |

## 2.3 Dynamics

* *Form Validation*: Jsp page sends form data to FormValidaion.js. Then FormValidaion.js returns either confirmation or failure with necessary messages
* *Storing Reviews*: Validated form is sent to Controller which sends information to entity which stores the Review Information in the database. Bean then retrieves Avg. Rating Score for the Airline/Aircraft combination from the database. The new Score is calculated and returned to the database. The Bean returns to Controller which passes back a confirmation message to the JSP. The JSP then displays confirmation message along with link to home page

*Storing Reviews*



# 

# User Management Subsystem

# User Management Subsystem Specification

## Role

The User Management Subsystem is responsible for dealing with User Accounts i.e. Logging in and out, registering and viewing account information.

## Scenarios

* *Login*: Allow user to enter their username and password to login to their account
* *Logout*: Allow user to logout of their account
* *Register*: Allow a user to register an account with FlightPub. This includes creating a username and password and inputting all personal information
* *View Account*: Allow a user to view their account information and flight history
* *Update Account*: Allow a user to update their account information

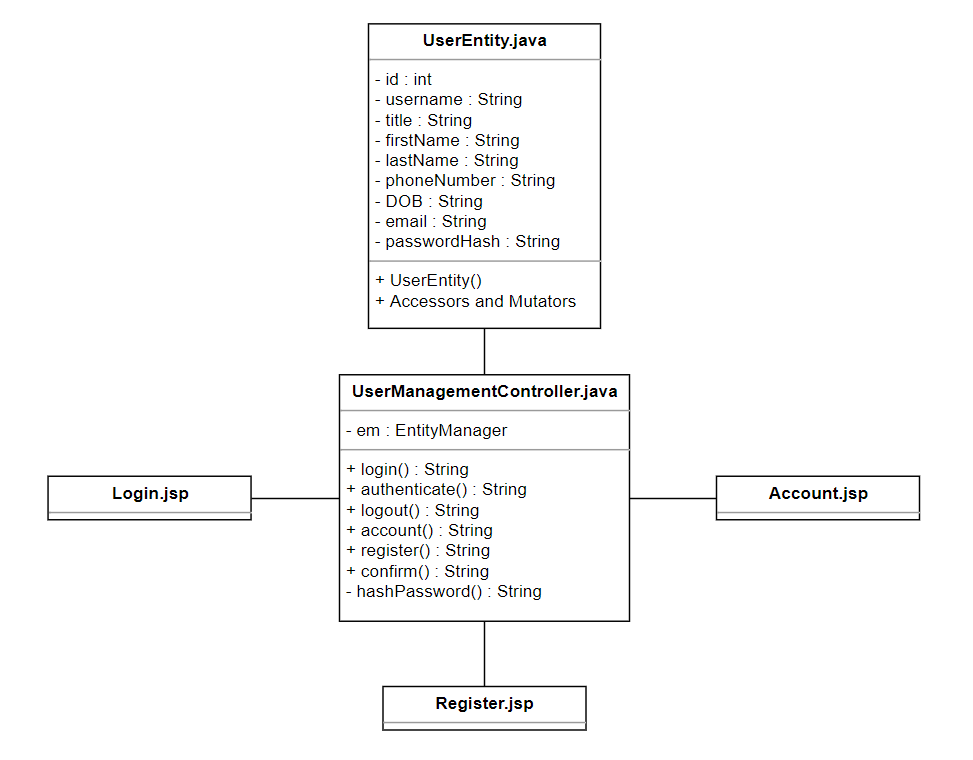
## Use Case Diagram

## 

## 

## 2. User Management Subsystem Design

## 2.1 Class Diagram



## 2.2 Class Descriptions

### 2.2.1 Class Name: <Login.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Display login form including Username and Password |  |
| Send login info to Controller | UserManagementController.java |
| Display ‘Register’ button and go to Register.jsp when selected | Register.jsp |
| Display ‘Home’ button and forward to Searching page when selected | Search.jsp (Searching Subsystem) |

### 2.2.2 Class Name: <Register.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Display Register form with username, password, and all personal information |  |
| Send Form to Controller for validation and user creation | UserManagementController.java |
| Retrieve Failed Registration message from Controller due to taken username or email. Display error message | UserManagementController.java |
| Forward to Login page on successful registration | Login.jsp |
| Display ‘Home’ button and forward to Searching page when selected | Search.jsp (Searching Subsystem) |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp |

### 2.2.3 Class Name: <Account.jsp>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Display personal information stored on account in session |  |
| Display flight history for account |  |
| Send Form to Controller for updating personal information | UserManagementController.java |
| Display ‘Home’ button and forward to Searching page when selected | Search.jsp (Searching Subsystem) |
| Display ‘Login’ button and forward to ‘Login’ page when selected | Login.jsp (UserManagement Subsystem) |

### 2.2.4 Class Name: <UserManagementController.java>

**Superclass Name:**  None

**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Retrieve login information from Login.jsp | Login.jsp |
| Validate user information through User.java bean | UserEntity.java |
| Go back to Login.jsp and send ‘Login Failure’ message. (if login failed) | Login.jsp |
| Set User as logged in if login success and return to Home page | Search.jsp |
| Retrieve Registration form from Register.jsp | Register.jsp |
| Create New User through UserEntity Bean | UserEntity.java |
| Receive Failed registration due to taken username or email error and submit Error to Register.jsp | UserEntity.java  Register.jsp |
| Receive Failed registration due to incorrect username or password error and submit Error to Login.jsp | UserEntity.java  Login.jsp |
| Receive Registration success and forward success message to Register.jsp | User.java  Register.jsp |
| Receive Logout Request from any page and drop logged in user | **All subsytems** |

## 

### 2.2.5 Class Name: <UserEntity.java>

**Superclass Name:**  None

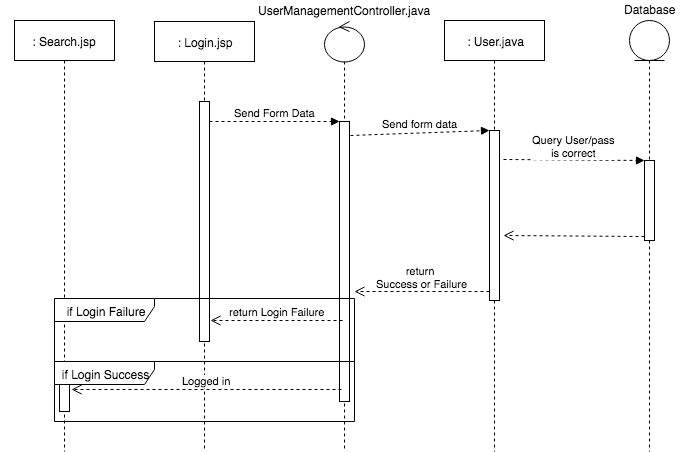
**Subclass Name:**  None

|  |  |
| --- | --- |
| **Responsibilities** | **Collaborators (classes or another subsystem)** |
| Receive Login information from Controller and query database for correct information | UserManagementController.java  Database |
| Return successful login or failure to Controller | UserManagementController.java |
| Receive Register information from Controller and query database to check if Username has been taken | UserManagementController.java  Database |
| Create new User entry in Database (if username or email not taken) | Database  UserManagementController.java |
| Return successful Registration or failure to Controller | UserManagementController.java |

## 2.3 Dynamics

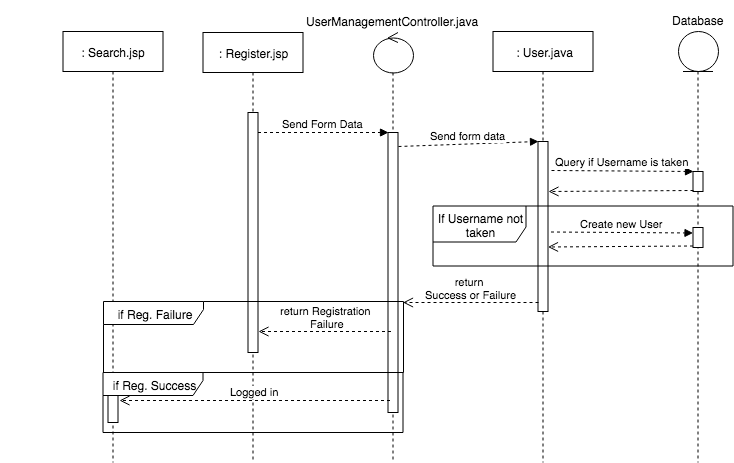
* *Login:* Login form is submitted to the Controller. Controller then access database through UserEntity bean. Bean then checks login information with User table to validate username and password. Bean then returns success or failure to Controller. If failure, controller sends failure message to Login.jsp. If success, Controller stores User as logged-in and goes to Home Page (Search.jsp)

*Login*



* *Register*: Register form is submitted to the Controller. Controller then access database through User bean. Bean then checks if username or email has been taken. If not, then bean creates new User in Database. If registration fails at any time, controller sends failure message to Register.jsp. If success, Controller stores new User as logged and goes to Home Page (Search.jsp)

*Register*

**