NANYANG TECHNOLOGICAL UNIVERSITY

Individual Project-1
Airline Booking System

FUNCTIONAL SPECIFICATION

COURSE TITLE	**:** CI6225 – ENTERPRISE APPLICATION DEVELOPMENT
LECTURER	**:** Mr. HARIHARAN VAIDHYANATHAN
STUDENT	**:** Adam Myrén

1 Background of the system

The Airline Booking System developed is to be used as a tool for airlines that which want to handle their customer interactions, such as bookings and customer registrations, and manage their routes via a web interface. It provides admin functionally, such as being able to retrieve a list of all customers. Except being able to book new flights, customers need to be able to view and manage all their previous bookings. Airline personnel needs to be able to add new flights and modify existing flights.

2 System Scope

2.1 Functional Requirements

Based on the background presented functional requirements has been developed in order to specify what functions that are required from the system. Due to time constraints, all requirements are not included in the first deliverable. The requirements that has been implemented are: Registration of user, login of user, search for flights, book a flight, view booked flights, view booked flights, show report of passengers on a given flight, show report of customers, modify a route and add a new route as these were considered to be part of the core functionality. In the table below, all functional requirements of the system are stated.

Functional Requirements
Registration of user
Login of user
Search for flights
Book a flight
View booked flights
Cancel flights
Show report of passenger on a given flight
Show report of all customers
Show financial report for a period of time
Modify a booking
Modify a route
Add a new route
Create new admins
Check in on a flight
Delete user

2.2 Users

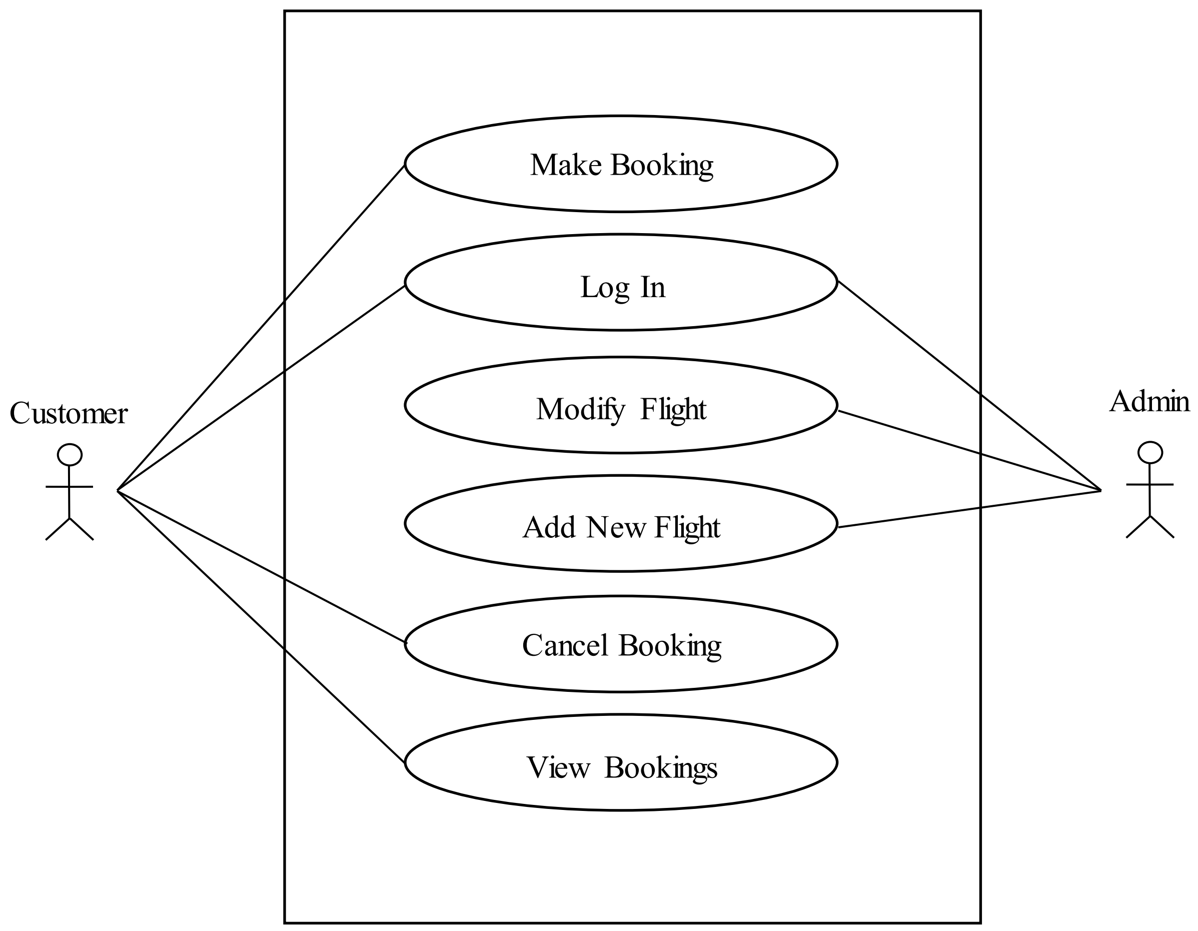
Both administrators from the airline company and the customers of the airline company should be able to use the system and its functionalities.

2.3 Assumptions

3 Analysis of the system

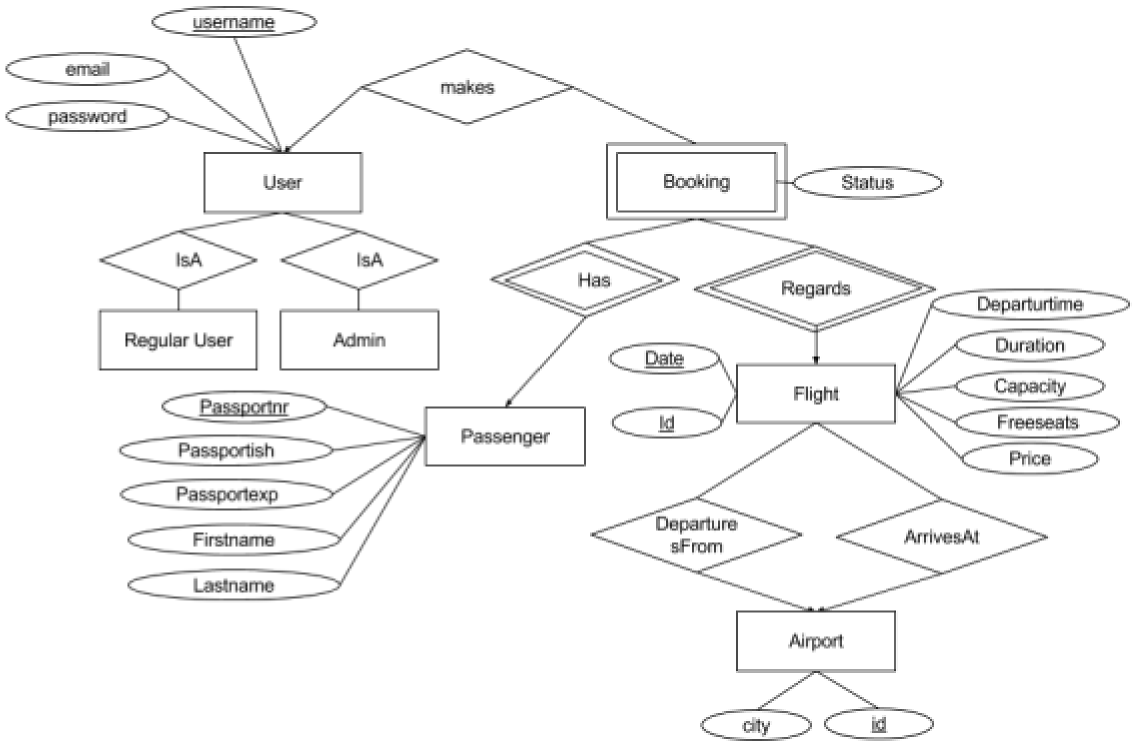
3.1 Use case diagram

Below is a use case diagram based on some of the requirements of the application.



3.2 Database analysis

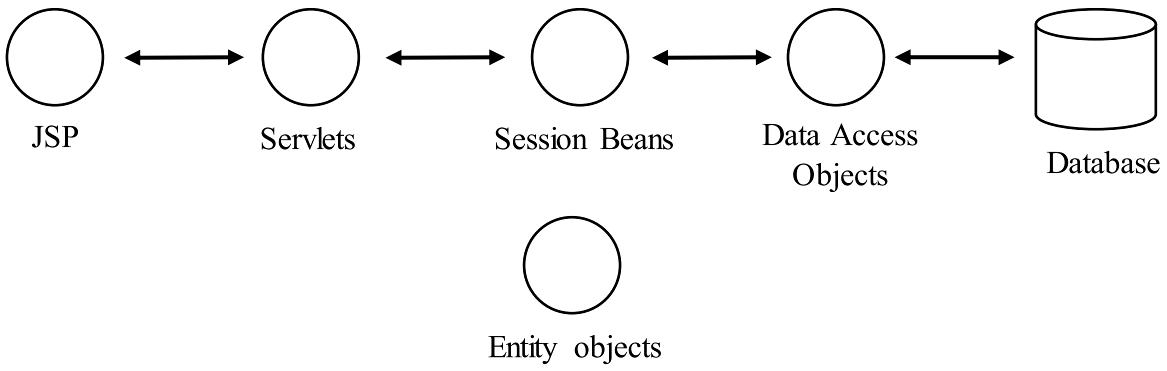
For persistence handling a relational database has been used. An entity relationship diagram has been developed and later translated into relational database tables. There are five tables in the database: Airports, Bookings, Flights, Passengers, and Users. They all have one primary key, except for flight that is uniquely defined by an id and a date. A booking is a week entity, that is uniquely defined by the flight it regards and the passenger that the passenger that that is registered on the booking.



4 System Design

4.1 Class diagram

There are five different types of main files in the system; JSP files, Servlets, java enterprise beans for business logic, java enterprise beans for data access, and entity objects. The system can be divided into three main parts, the view, the model and the controller. The view consists of JSP files and that are responsible for displaying information to the user. The controller consists of java servlets, responsible for handling user input and redirecting it to the appropriate session bean. The model consists of a number of session bean responsible for fetching and adding data in the database through use of the data access objects.



4.2 Entity Class

The entity classes are either representations of tables in the mysql database or elements in the jsp view. Their objective is to collect data related to one object so that it easily can be transferred between other classes.

4.2.1 Airport

Representation of the airports table in the database with getters and setters for all attributes.

Class Name	**Airport**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	Airport(String id, String city)		
Attribute	id	String	ID of the airport
	city	String	City of the airport
Method	getId(): String		Getter method for id
	setId(String id): void		Setter method for id
	getCity(): String		Getter method for city
	setCity(String city): void		Setter method for city

4.2.2 Booking

Representation of the bookings table in the database with getters and setters for all attributes.

Class Name	**Booking**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	Booking(String flight_id, String passenger, String user, String status, String date)		
Attribute	flight_id	String	Flight that the booking regards
	passenger	String	Passenger that the booking regards
	user	String	User that has created the booking
	status	String	Status of the booking
	date	String	Date that the booking concerns
Method	getFlight_id(): String		Getter method for id
	setFlight_id(String flight_id): void		Setter method for id
	getPassenger(): String		Getter method for city
	setPassenger(String passenger): void		Setter method for city
	getUser():String		Getter method for user
	setUser(String user):void		Setter method for user
	getStatus():String		Getter method for status
	setStatus(String status):void		Setter method for status
	getDate():String		Getter method for date
	setDate(String date):void		Setter method for date

	toString():String	Override toString

4.2.3 Flight

Representation of the airports table in the database with getters and setters for all attributes.

Class Name	**Flight**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	Flight(String id, String date, String departure_time, String duration, int capacity, String origin, String destination)		
Attribute	id	String	ID of the flight
	date	String	Date of the flight
	departure_time	String	Departure time of flight
	duration	String	Duration of the flight
	capacity	int	Capacity of the aircraft
	origin	String	ID of origin airport
	destination	String	ID of destination airport
	Price	String	Ticket price for the flight
Method	getId():String		Getter method for id
	setId(String _id): void		Setter method for id
	getDate():String		Getter method for date
	setDate(String date):void		Setter method for date
	getDeparture_time():String		Getter method for departure_time
	setDeparture_time(String departure_time): void		Setter method for departure_time
	getDuration():String		Getter method for duration
	setDuration(String duration):void		Setter method for duration
	getCapacity(): int		Getter method for capacity
	setCapacity(int capacity):void		Setter method for capacity
	getOrigin():String		Getter method for origin
	setOrigin(String origin):void		Setter method for origin
	getDestination():String		Getter method for destination
	setDestination(String destination):void		Setter method for destination
	getPrice():String		Getter method for price
	setPrice(String price):void		Setter method for price
	toString():String		Override toString

4.2.4 MyPagesBookingElement

A representation of a join of two tables in the database, used to display information regarding bookings to the user.

Class Name	**MyPagesBookingElement**		
Type	**Plain Java Class**		
Package	**entities**		
Constructor	MyPagesBookingElement(String id, String date, String departure_time, String duration, String origin, String destination, String firstname, String lastname, String passportnr)		
Attribute	id	String	ID of the flight
	date	String	Date of the flight
	departure_time	String	Departure time of flight
	duration	String	Duration of the flight
	origin	String	ID of origin airport
	destination	String	ID of destination airport
	firstname	String	First name of passenger
	lastname	String	Last name of passenger
	passportnr	String	Passport number of passenger
Method	getId():String	Getter method for id	
	setId(String _id): void	Setter method for id	
	getDate():String	Getter method for date	
	setDate(String date):void	Setter method for date	
	getDeparture_time():String	Getter method for departure_time	
	setDeparture_time(String departure_time): void	Setter method for departure_time	
	getDuration():String	Getter method for duration	
	setDuration(String duration):void	Setter method for duration	
	getCapacity(): int	Getter method for capacity	
	setCapacity(int capacity):void	Setter method for capacity	
	getOrigin():String	Getter method for origin	
	setOrigin(String origin):void	Setter method for origin	
	getDestination():String	Getter method for destination	
	setDestination(String destination):void	Setter method for destination	
	getFirstname():String	Getter method for firstname	
	setFirstname(String firstname):void	Setter method for firstname	
	setLastname():String	Getter method for lastname	
	setLastname(String lastname):void	Setter method for lastname	
	getPassportnr():String	Getter method for passportnr	
	setPassportnr(String passportnr):void	Setter method for passportnr	
	toString():String	Override toString	

4.2.5 Passenger

Representation of the passengers table in the database with getters and setters for all attributes.

Class Name	**Passenger**

Type	Plain Java Class		
Package	entities		
Constructor	MyPagesBookingElement(String id, String date, String departure_time, String duration, String origin, String destination, String firstname, String lastname, String passportnr)		
Attribute	firstname	String	First name of passenger
	lastname	String	Last name of passenger
	passportnr	String	Passport number of passenger
	passportexp	String	Expiry date of passport
	passportish	String	Date of issue of passport
	birth	String	Date of birth
Method	getFirstname():String		Getter method for firstname
	setFirstname(String firstname):void		Setter method for firstname
	getLastname():String		Getter method for lastname
	setLastname(String lastname):void		Setter method for lastname
	getPassportnr():String		Getter method for passportnr
	setPassportnr(String passportnr):void		Setter method for passportnr
	getPassportexp():String		Getter method for passportexp
	setPassportexp(String passportexp):void		Setter method for passportexp
	getPassportish():String		Getter method for passportish
	setPassportish(String passportish):void		Setter method for passportish
	getBirth():String		Getter method for birth
	setBirth(String birth):void		Setter method for birth
	toString():String		Override toString

4.2.6 User

Representation of the users table in the database with getters and setters for all attributes.

Class Name	User		
Type	Plain Java Class		
Package	entities		
Constructor	User(String username, String password, String email, boolean admin)		
Attribute	username	String	Username of user
	password	String	Password of user
	email	String	Email of user
	admin	boolean	Indicates whether user has admin privileges
Method	getUsername():String		Getter method for username
	setUsername(String username):void		Setter method for username

	getPassword():String	Getter method for password
	setPassword(String password):void	Setter method for password
	getEmail():String	Getter method for email
	setEmail(String email):void	Setter method for email
	getAdmin():boolean	Getter method for admin
	setAdmin(boolean admin):void	Setter method for admin

4.3 Data Access Objects

The data access objects exist to separate the business layer from the database implementation, there is one DAO for each table in the database. All DAO's in this system are implemented towards a mysql database.

4.3.1 AirportDAO

Data access object handling all requests towards the Airports table in the database.

Class Name	**AirportDAO**		
Type	**Session Bean**		
Package	**dao**		
Constructor	AirportDAO()		
Attribute	DBusername	String	Final String that holds DB username
	DBpassword	String	Final String that holds DB password
	DBurl	String	Final String that holds DB url
Method	getAirportId(String city):List<Airport>		Return all airport id's for a given city

4.3.2 BookingDAO

Data access object handling all requests towards the Bookings table in the database.

Class Name	**BookingDAO**		
Type	**Session Bean**		
Package	**dao**		
Constructor	BookingDAO()		
Attribute	DBusername	String	Final String that holds DB username
	DBpassword	String	Final String that holds DB password
	DBurl	String	Final String that holds DB url
Method	addBooking(Passenger passenger, Flight flight, User user):boolean		Adds the given passenger to the given flight with the given user as reference
	getBookings(User user):List<Booking>		Returns a list of all bookings related to a given user

	getBookings(String flight_id, String date):List<Booking>	Returns a list of all bookings on a given flight
	deleteBooking(String flight_id, String date, String passportnr):boolean	Deletes a given booking

4.3.3

4.3.4 FlightDAO

Data access object handling all requests towards the Flights table in the database.

Class Name	**FlightDAO**		
Type	**Session Bean**		
Package	**dao**		
Constructor	FlighDAO()		
Attribute	DBusername	String	Final String that holds DB username
	DBpassword	String	Final String that holds DB password
	DBurl	String	Final String that holds DB url
Method	getFlights(String origin, String destination, String date, int nr_of_tickets):List<Flight>		Returns a list of flights between two airports on a given day that has more or equal free seats as nr_of_tickets
	getFlight(String id, String date):Flight		Returns a given flight from the database based on the primary keys
	exists(String flight_id, String date):boolean		Returns true if the flight exists, else false
	add(Flight flight):boolean		Adds a flight to the database
	Update(String flight_id, String date, Flight new_flight):boolean		Updates an existing flight with new data

4.3.5 PassengerDAO

Data access object handling all requests towards the Passengers table in the database.

Class Name	**PassengerDAO**		
Type	**Session Bean**		
Package	**dao**		
Constructor	PassengerDAO()		
Attribute	DBusername	String	Final String that holds DB username
	DBpassword	String	Final String that holds DB password
	DBurl	String	Final String that holds DB url
Method	getPassenger(String passportnr):Passenger		Returns a passenger based on the primary key
	addPassenger(Passenger passenger):boolean		Adds a new passenger to the database
	exists(Passenger passenger):boolean		Returns true if the passenger exists, else false

4.3.6 UserDAO

Data access object handling all requests towards the Users table in the database.

Class Name	**UserDAO**		
Type	**Session Bean**		
Package	**dao**		
Constructor	UserDAO()		
Attribute	DBusername	String	Final String that holds DB username
	DBpassword	String	Final String that holds DB password
	DBurl	String	Final String that holds DB url
Method	getUser(String username):User		Returns a user based on the primary key
	add(User user):boolean		Adds a new user to the database
	exists(String username, String password):boolean		Returns true if the user exists, else false
	getAll():List<User>		Returns a list of all users

4.4 Business Logic Beans

These java enterprise beans handle all business logic in the system. The collect and edit persistent data through the use of DAO's and they are invoked by the different servlets.

4.4.1 AdminBean

Handles all business logic related to a user with admin privileges.

Class Name	**AdminBean**		
Type	**Session Bean**		
Package	**model**		
Constructor	Adminbean()		
Attribute	flightdao	FlightDAO	DAO to handle flights
	passengerda o	PassengerDAO	DAO to handle passengers
	bookingdao	BookingDAO	DAO to handle bookings
	userdao	UserDAO	DAO to handle users
Method	addNewFlight(String flight_id, String date, String departure_time, String duration, int capacity, String origin, String destination):boolean		Checks if the given a flight with the given values already exists, if not it adds it to the database, else it returns false
	getFlight(String id, String date):Flight		Returns a given flight from the database based on the primary keys
	Update(String flight_id, String date, Flight new_flight):boolean		Updates an existing flight with new data
	getPassengers(String flight_id, String date):ArrayList<Passenger>		Returns a list of all passengers on a given flight

4.4.2 Booking Bean

Handles all business logic involved when making a booking, from search of flights to registration of the booking.

Class Name	**BookingBean**		
Type	**Session Bean**		
Package	**model**		
Constructor	BookingBean()		
Attribute	flightdao	FlightDAO	DAO to handle flights
	passengerdao	PassengerDAO	DAO to handle passengers
	bookingdao	BookingDAO	DAO to handle bookings
Method	getFlights(String origin, String destination, String date, int nr_of_tickets):List<Flight>		Returns a list of flights between two airports on a given day that has more or equal free seats as nr_of_tickets
	bookFlights(List<Passenger> passengers, List<Flight> flights, User user):boolean		Books all passengers on all flights and associates them with the given user
	deleteBooking(String flight_id, String date, String passportnr):boolean		Deletes a given booking

4.4.3 UserBean

Handles all type of user administration, such as adding users and administrating information related to a specific user.

Class Name	**UserBean**		
Type	**Session Bean**		
Package	**model**		
Constructor	UserBean()		
Attribute	flightdao	FlightDAO	DAO to handle flights
	passengerdao	PassengerDAO	DAO to handle passengers
	bookingdao	BookingDAO	DAO to handle bookings
	userdao	UserDAO	DAO to handle users
Method	addNewUser(User user):boolean		Adds a new user to the database
	getUser(String username):User		Returns a user based on the primary key
	getBookingElements(User):List<MyPagesBookingElement>		Returns all booking elements related to a given user

4.5 Servlets

12 Servlets have been developed to handle request from the view and communicate with the business logic through instantiation of Session Beans. All Servlets have two methods, doGet and doPost.

Servlet Name	Functionality

AddFlightServlet	Adds a new flight to the system, if the add is not successful it attaches an error and returns to previous jsp
BookingServlet	Books passengers on a given flight, if the booking is unsuccessful it attaches an error and returns the user to the page where the information is entered
DeleteBookingServlet	Deletes a booking from the database
EditFlightServlet	Updates a flight in the database, if the update is unsuccessful, it attaches an error and returns the user to the page where the invalid input was entered
FlightInformationServlet	Gets a flight from the database based on the primary keys
IndexServlet	Generates a background image for the session, then redirects to the right start jsp, StartPage and MyPages respectively, depending on if a user is logged in or not
LoginServlet	Can create or login a user depending on parameters, if either is unsuccessful it attaces an error and redirects back to Login.jsp
LogoutServlet	Invalidates the current session and sends the user to the StartPage
MyPagesServlet	Loads necessary information (booked flights) to start MyPages.jsp
ReportServlet	Creates a list of entities to display to the administator
SearchServlet	Attatches a list of flights on a given date between two given airports to the request then forwards the request to SelectFlight.jsp
SelectFlightServlet	Intepretates what flight the user has picked and then attaches that flight to the request and forwards it to either Login.jsp or PutInformation.jsp depending on if a user is already logged in

4.6 JSP Files

A number of JSP page have been used to display information to the user and to collect various inputs.

Name	Functionality
AdminPage.jsp	StartPage for admin users
ConfirmAndPay.jsp	Page where the user can review the booking details and enter payment information
DisplayReport.jsp	A page to display various reports generated by an admin user
Login.jsp	Page where one can login or create a new user
MyPages.jsp	StartPage for a logged in user
PutInformation.jsp	Page where a user that is about to make a booking can enter passenger information

SelectFlight.jsp	Page where a user that is about to make a booking can select among available flights
StartPage	Start page if no one is logged in

4.7 Supporting Files

Besides files that support the main functionalities of the system, a number of other files are needed to style the jsp pages and make them interactive for the user. Therefore, css and Javascript have been used to achieve this.

Type	Name	Description
CSS	main.css	Overall style
	login.css	Style Login.jsp
	adminpage.css	Style AdmiPage.jsp
	confirmandpay.css	Style ConfirmAndPay.jsp
	displayreport.css	Style DisplayReport.jsp
	mypages.css	Style MyPages.jsp
	putinformation.css	Style PutInformation.jsp
	selectflight.css	Style SelectFlight.jsp
Javascript	script.js	Client side script