CC5003 Data Structures and Specialist Programming

Individual Coursework 2

Mateusz Pietryka

Student number: 20029204

Table of Contents

INTRODUCTION 3

WEBSITE STRUCTURE 3

WEBSITE DESIGN (PRESENTATION TIER) 6

DATABASE (DATA TIER) 7

JAVA BEANS (BUSINESS LOGIC TIER) 8

BookingBean **8**

CarBean **9**

TESTING 10

Rent form **10**

Manage Booking form **15**

Edit Booking form **17**

CONCLUSION 19

Introduction

For this assignment I have created a Three-Tier web application that mimics a car hire. This application allows the user to book a car and manage their booking. The user can retrieve the information about their booking using the booking id, once the booking is retrieved the user is also able to edit and cancel it.

In this report I am going to describe the structure and functionality of this application in detail.

Website Structure

My website consists of 9 pages:

- **Index** – The home page, I have created this page first as a general template for my website in terms of design. This page sets the theme for the website and lets the user know what it's all about. This page is a static page, and the only way user can interact with it is by using the links in the navigation bar.

- **Rent** – This page contains a form that allows user to rent a car by entering their details and choosing the car from the list. Their input will be validated, and a new entry will be added to a suitable table in the database. This page carries out the Create part of my CRUD functionality.

- **SuccessRent** – This page only appears after a successful execution of the rent function, it displays the "Congratulations" message to the user containing their booking number.

- **ManageBooking** – This page contains a form that allows user to manage their booking by entering their previously generated booking number (displayed to them in **SuccessRent** page) and hitting the “find” button. Their input will be validated, and the backing bean will search for the booking. This page carries out the Read part of my CRUD functionality.

- **NoBooking** – This page only appears after an unsuccessful execution of the manage booking function meaning that the booking with the id entered by the user was not found. This page displays a message stating that the booking doesn’t exist. This page is a static page, and the only way user can interact with it is by using the links in the navigation bar or clicking on the button provided.

- **Booking** – This page only appears after a successful execution of the manage booking function meaning that the booking with the id entered by the user was found. This page contains the information about the booking requested on the previous page. This page also allows the user to cancel and edit their booking. This page carries out the Delete part of my CRUD functionality.

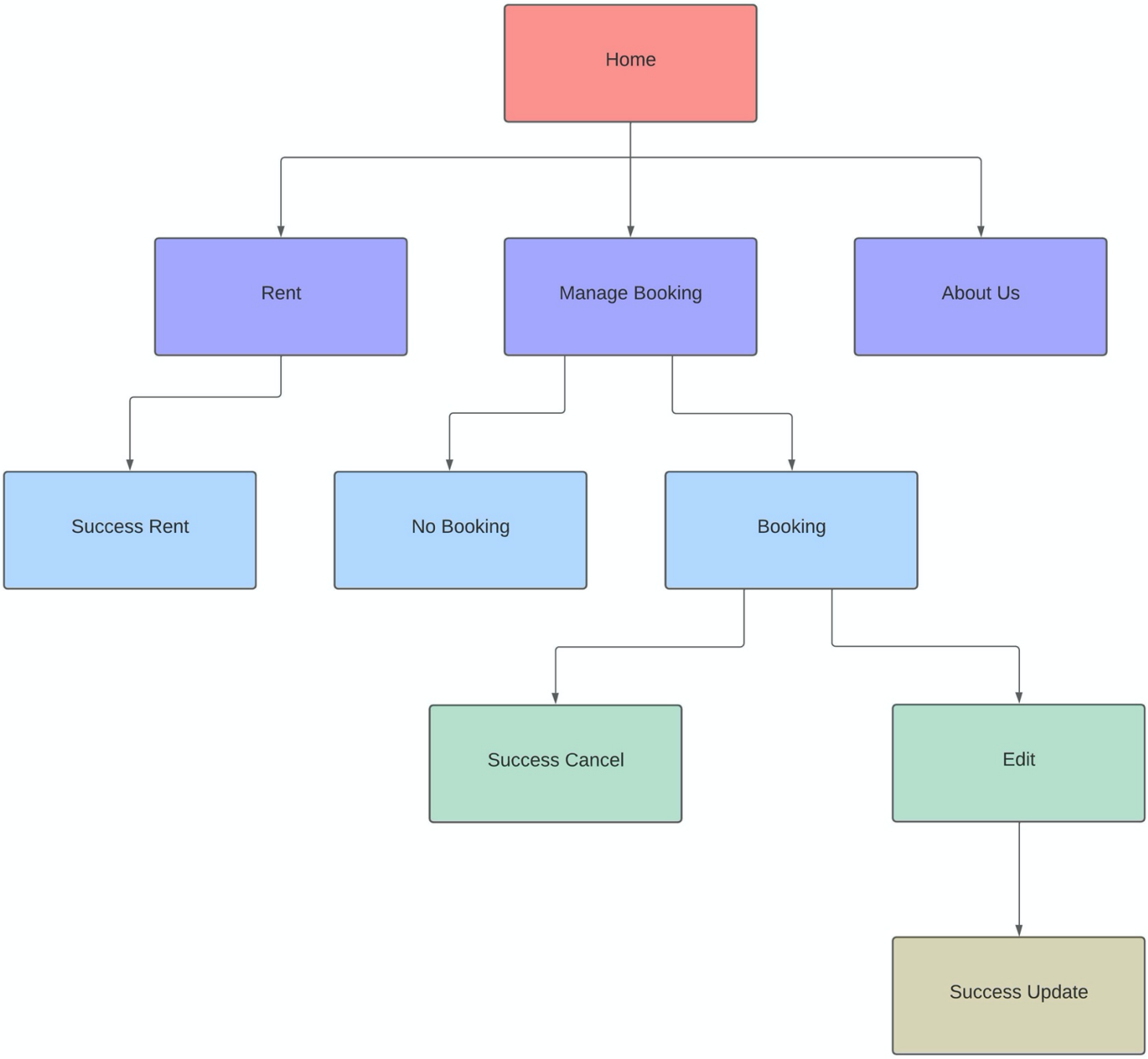
- **SuccessCancel** – This page only appears after a successful execution of the cancel function, it displays a message with a confirmation of the cancellation. This page is a static page, and the only way user can interact with it is by using the links in the navigation bar.

- **Edit** – This page can only be accessed from the **Booking** page, it contains a form that allows user to edit their booking. Their input will be validated, and the information about their booking will be updated in the database. This page carries out the Update part of my CRUD functionality.

- **SuccessUpdate** – This page only appears after a successful execution of the edit function, it displays a message with a confirmation of the update. This page is a static page, and the only way user can interact with it is by using the links in the navigation bar.

- **About us** – This page talks about a brief history of the company, I have created this page to add a more real feel to the website. This page is a static page, and the only way user can interact with it is by using the links in the navigation bar.

Fig.1. Site map. (This site map does not include the page displaying the current state of the database as it was created purely for the demo purposes and wouldn't be included in the final version of the website.)



Website Design (Presentation Tier)

The first thing I have done when designing this website was drawing the wire frame of the design I wanted on paper. I wanted to give my website a modern, intuitive feel where the user can easily find everything they need without going through too many pages.

After careful consideration and browsing a few real-world websites for inspiration I came up with this:

Fig.2. Website wireframe.

After creating the wireframe, I created and styled a template page that I later filled with relevant content. Using a template not only helped me save valuable time but also helped me make sure that all my pages look the same and the website has consistent aesthetics throughout.

Database (Data Tier)

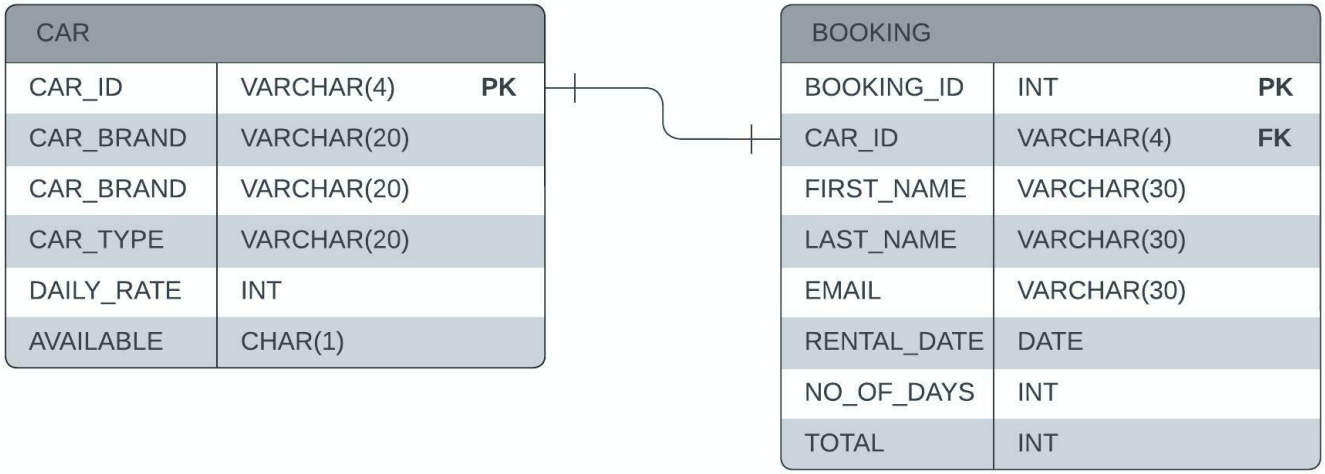
For this project I have created a relational database consisting of two tables:

- CAR – Stores all the information about the cars. This table is pre-populated with relevant data. **CAR_ID** attribute is a Primary Key of this table.

- BOOKING – Stores all information about bookings as well as customer details. The entries in this table are created by the user while interacting with the website. **BOOKING_ID** attribute is a Primary Key of this table, **CAR_ID** attribute is a Foreign Key in this table, and it references the **CAR_ID** attribute of the CAR table.

The relation between those entities is a One-to-One Relationship.

Fig.3. Entity relation diagram. (PK = Primary Key, FK = Foreign Key)



Full SQL code showing the constraints is available in ExpressCars.sql file included in this submission.

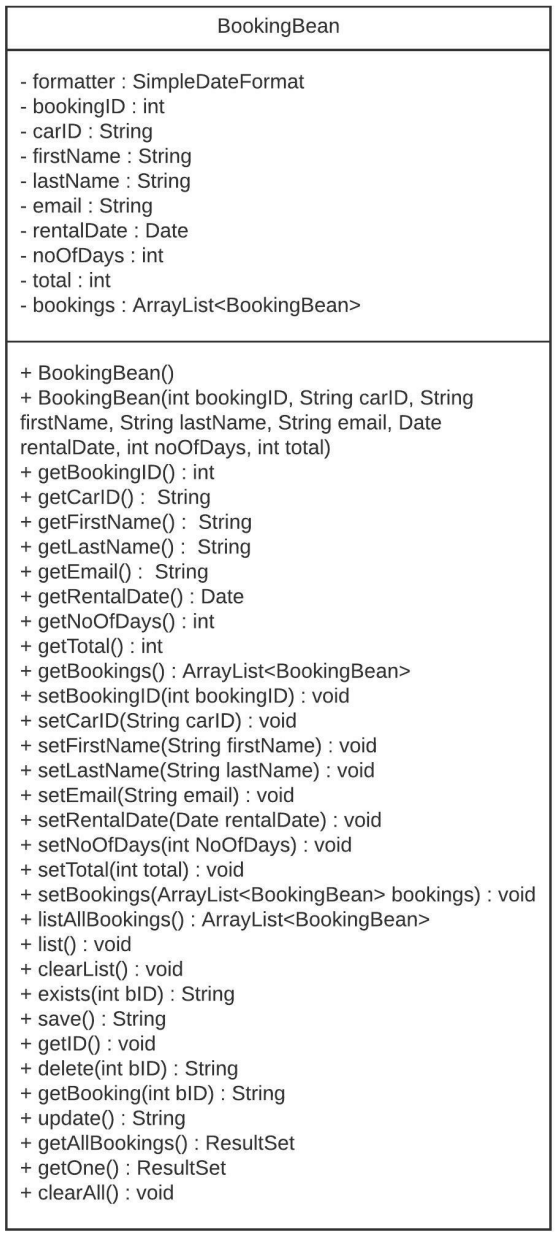
Java Beans (Business Logic Tier)

Business Logic tier for this project consists of 2 classes:

BookingBean

This class corresponds to BOOKING table in my database and manages all tasks related to it. This Class consists of 32 methods:

- Set of standard “get” and “set” methods for each attribute.
- **BookingBean()** – Default constructor.
- **BookingBean(int bookingID, String carID, String firstName, String lastName, String email, Date rentalDate, int noOfDays, int total)** – Constructor used for populating the ArrayList.
- **listAllBookings()** – Creates an ArrayList of all bookings using a SQL query to retrieve all bookings from database. This method returns an ArrayList.
- **clearList()** – Removes all items from the ArrayList of bookings.
- **list()** – Calls **clearList**() method, then calls **listAllBookings()** method and returns nothing, this way the method can be called inside a JSF page without displaying anything on the page itself.
- **exists(bID)** – Iterates over the ArrayList in search of a booking with id specified by the user. This method redirects the user to a suitable page depending on the result of the search.
- **save()** – Calculates the booking total and injects a new entry into a BOOKING table in the database, changes the *available* attribute of chosen car to ‘N’ to indicate that the car is not available.
- **getID()** – Retrieves a bookingID of the car based on its carID.
- **delete(bID)** – Deletes a row from the BOOKING table in the database based on bookingID and changes the *available* attribute of the car back to ‘Y’ to indicate that the car is available to rent again.
- **getBooking(bID)** – Retrieves all information about the booking from the database based on the bookingID passed by the user.
- **update()** – Updates the information about the booking in the BOOKING table of the database.
- **getAllBookings()** – Retrieves all rows from table BOOKING and returns them as rowSet.
- **getOne()** – Retrieves information about the booking and the car and returns them as rowSet.
- **clearAll()** – Sets all attributes to null.



CarBean

This class corresponds to CAR table in my database and manages all tasks related to it. This Class consists of 22 methods:

- Set of standard “get” and “set” methods for each attribute.
- **CarBean()** – Default constructor.
- **CarBean(String carID, String carBrand, String carModel, String carType, int dailyRate, char available**) – Constructor used for populating the ArrayList.
- **getCar(String id)** – Retrieves information about the specific car from CAR table.
- **getAvailableCars()** – Creates an ArrayList of all cars using a SQL query to retrieve all cars from database where *available* attribute is set to ‘Y’. This method returns an ArrayList.
- **clearList()** – Removes all items from ArrayList of cars.
- **description()** – Returns a nicely formatted string with information about the instance of CarBean. Used to display information about the car in the drop-down menu in the rent form.
- **Description2()** – Returns a nicely formatted string with information about the instance of a CarBean. Used to display information about the car in the edit form.

```
CarBean
- carID : String
- carBrand : String
- carModel : String
- carType : String
- dailyRate : int
- available : char
- cars : ArrayList<CarBean>

+ CarBean()
+ CarBean(String carID, String carBrand, String
carModel, String carType, int dailyRate, char available)
+ getCarID() : String
+ getCarBrand() : String
+ getCarModel() : String
+ getCarType() : String
+ getDailyRate() : int
+ getAvailable() : String
+ getCars() : ArrayList<CarBean>
+ setCarID(String carID) : void
+ setCarModel(String CarModel) : void
+ setCarBrand(String CarBrand) : void
+ getCarType(String CarType) : void
+ setDailyRate(int DailyRate) : void
+ setAvailable(char available) : void
+ setCarType(String carType) : void
+ setCars(ArrayList<CarBean> cars) : void
+ getCar(String id) : void
+ getAvailableCars() : ArrayList<CarBean>
+ clearList() : void
+ description() : String
+ description2() : String
```

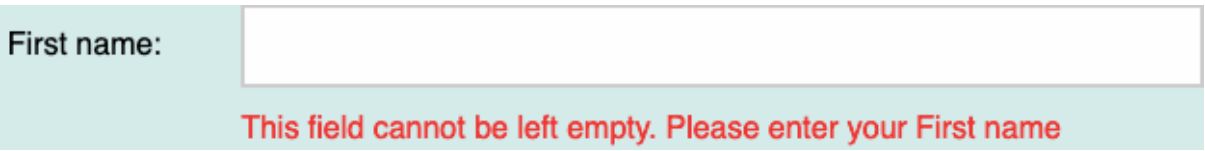
Testing

To make sure my application is working as intended I have conducted a thorough testing. I have spent some time trying to enter different values into my forms to ensure the validators are working correctly and catch all the errors that a regular user could encounter while using my application.
In this part of my report, I am going to share some of the possible test cases and their outcomes.

Rent form

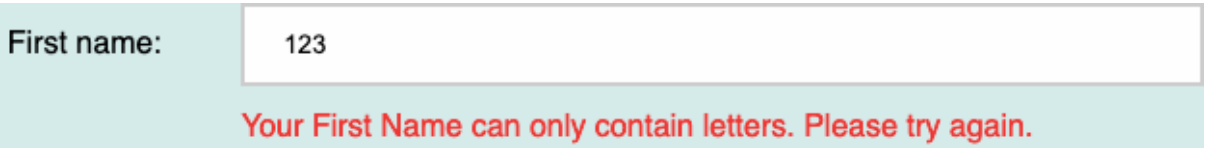
First name field within my rent form is validated using regular expressions, it will only accept alphabetical characters. This field cannot be left empty.

- **Test Case 1 –** Field left empty.



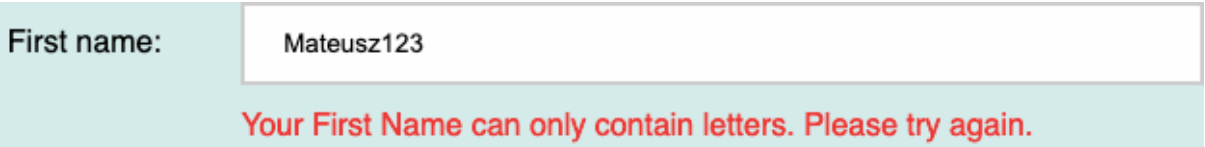
Outcome: Suitable error message displayed.

- **Test Case 2 –** Numeric characters entered.



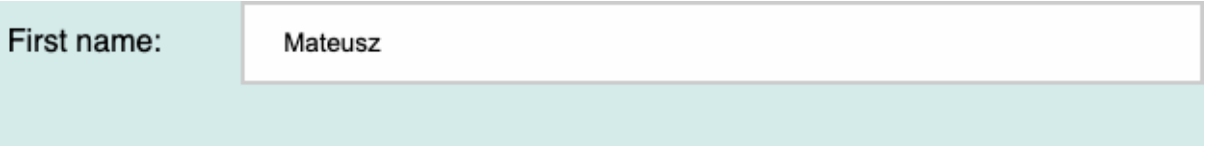
Outcome: Suitable error message displayed.

- **Test Case 3 –** Combination of letters and numbers entered.



Outcome: Suitable error message displayed.

- **Test Case 4 –** Only alphabetical characters entered.

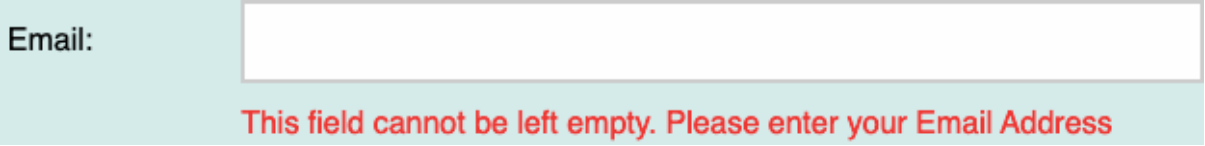


Outcome: Entry accepted, no error message was displayed.

Last name field is validated with the same regular expression therefore I am going to skip the testing for that field.

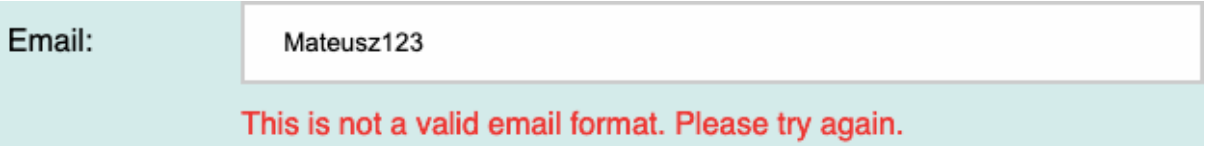
Email field within my rent form is validated using regular expressions, it will only accept email-like string of characters. This field cannot be left empty.

- **Test Case 5 –** Field left empty.



Outcome: Suitable error message displayed.

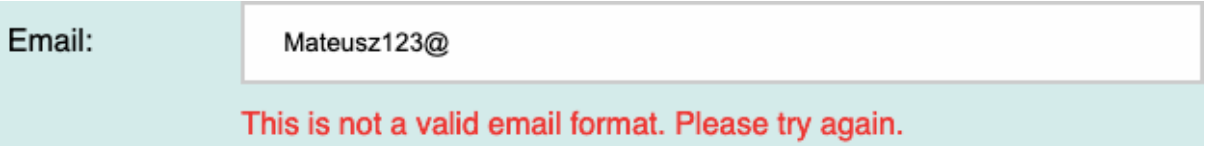
- **Test Case 6** – Invalid email format, no @ sign, no domain.



Outcome: Suitable error message displayed.

- **Test Case 7**

Input: Invalid email format, no domain.



Outcome: Suitable error message displayed.

- **Test Case 8** – Invalid email format, domain incomplete.

Email:	Mateusz123@gmail
	This is not a valid email format. Please try again.

Outcome: Suitable error message displayed.

- **Test Case 9** – Valid email address entered.

Email:	Mateusz123@gmail.com

Outcome: Entry accepted, no error message was displayed.

Rental Date field within my rent form uses a DateTimeConverter, it will only accept dates in dd/MM/yyyy format. This field cannot be left empty.

- **Test Case 10 –** Field left empty.

Rental Date:	
	This field cannot be left empty. Please enter the Rental Date

Outcome: Suitable error message displayed.

- **Test Case 11 –** random word entered.

Rental Date:	tomorrow
	This is not a valid date format (dd/mm/yyyy). Please try again.

Outcome: Suitable error message displayed.

- **Test Case 12 –** random number entered.

Rental Date:	123
	This is not a valid date format (dd/mm/yyyy). Please try again.

Outcome: Suitable error message displayed.

- **Test Case 13 –** invalid date format dd.MM.yyyy instead of dd/MM/yyyy.

Rental Date:	03.05.2022
	This is not a valid date format (dd/mm/yyyy). Please try again.

Outcome: Suitable error message displayed.

- **Test Case 14 –** invalid date format yyyy/MM/dd instead of dd/MM/yyyy.

Rental Date:	2022/03/05
	This is not a valid date format (dd/mm/yyyy). Please try again.

Outcome: Suitable error message displayed.

- **Test Case 15 –** Valid date entered.

Rental Date:	03/05/2022

Outcome: Entry accepted, no error message was displayed.

Number of days field within my rent form is validated using validateDoubleRange tag, it will only accept numbers between 1 and 30. This field will only accept 2 digits and cannot be left empty.

- **Test Case 16 –** Field left empty.

Number of days:	
	Please enter Number of Days

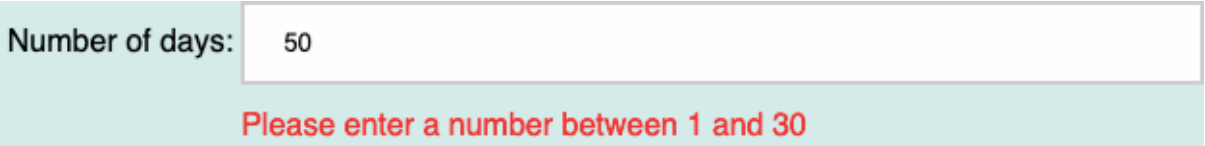
Outcome: Suitable error message displayed.

- **Test Case 17 –** 0 entered.

Number of days:	0
	Please enter a number between 1 and 30

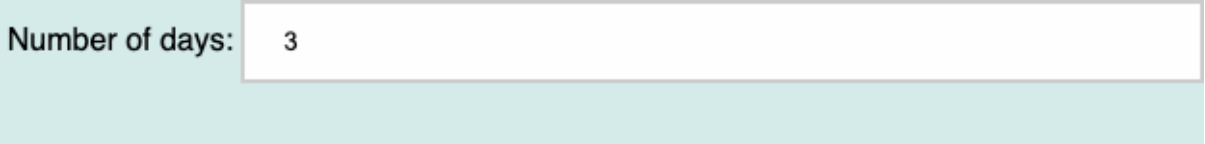
Outcome: Suitable error message displayed.

- **Test Case 18 –** 50 entered.



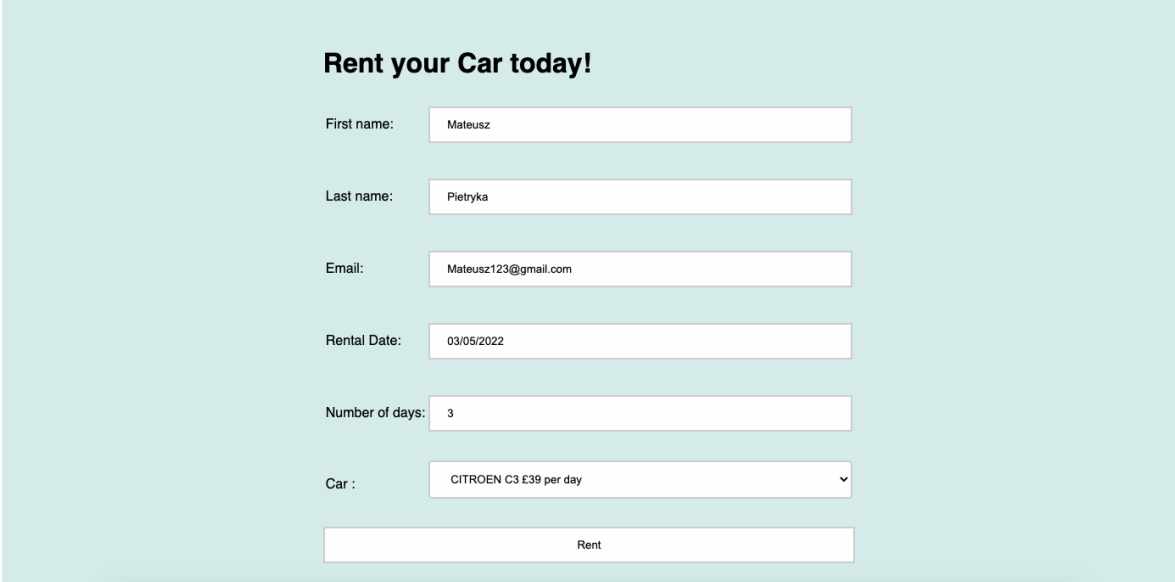
Outcome: Suitable error message displayed.

- **Test Case 19 –** Valid number entered.

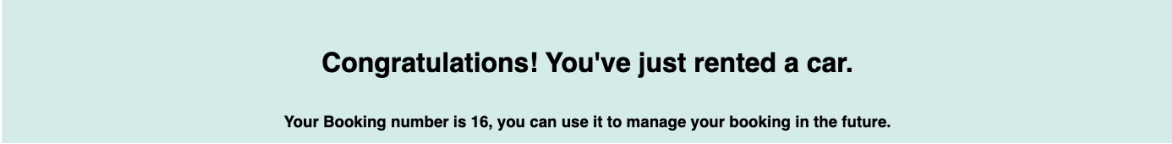


Outcome: Entry accepted, no error message was displayed.

After filling up the fields with valid test data the form is ready to submit.



After hitting the “Rent” button, user is taken to another page where their uniquely generated booking ID is displayed.



New database entry was added.

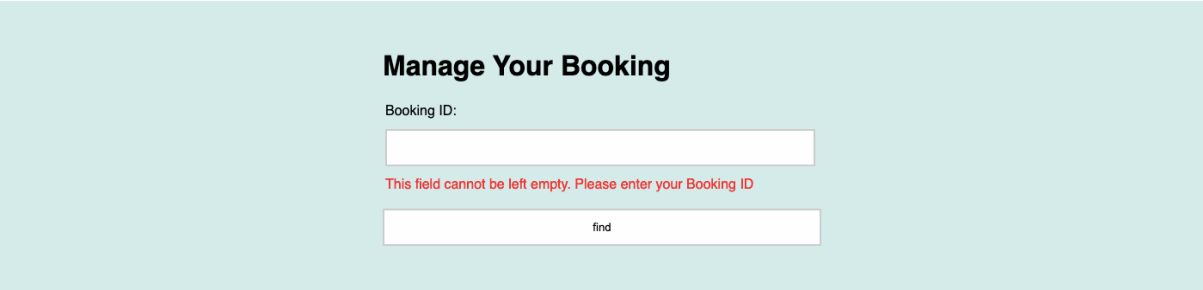
#	BOOKING_ID	CAR_ID	FIRST_NAME	LAST_NAME	EMAIL
1	1	0119	Liam	Manning	lmanning@gmail.com
2	3	0610	Mateusz	Pietryka	mat.pietryka@gmail.com
3	4	0619	Barbara	Dias	babad@gmail.com
4	14	0149	Barbara	Dias	babadias@gmail.com
5	16	0666	Mateusz	Pietryka	Mateusz123@gmail.com

Manage Booking form

After successfully booking a car, we can now use the “Manage Booking” page.

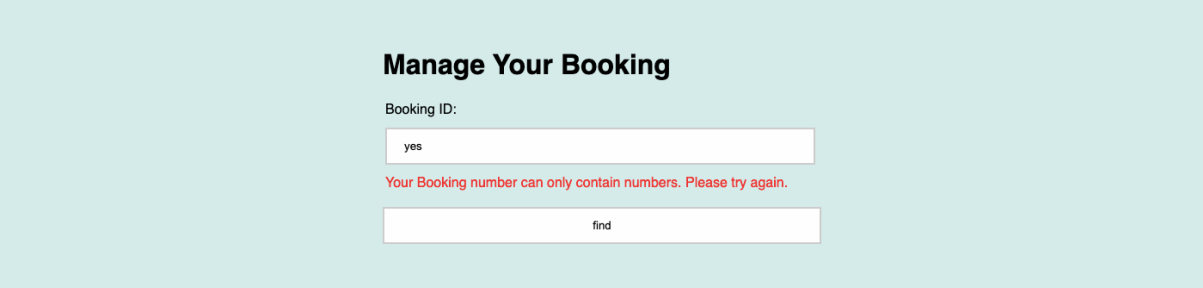
Booking ID field within my Manage Booking form will only accept numerical characters. This field cannot be left empty.

- **Test Case 20 –** Field left empty.



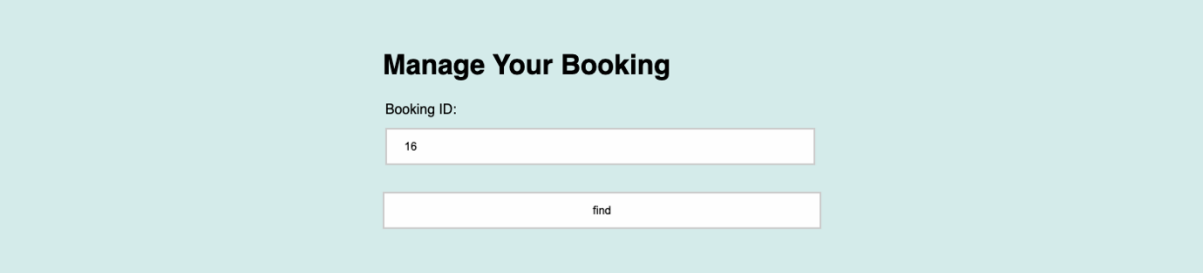
Outcome: Suitable error message displayed.

- **Test Case 21 –** Alphabetical characters entered.



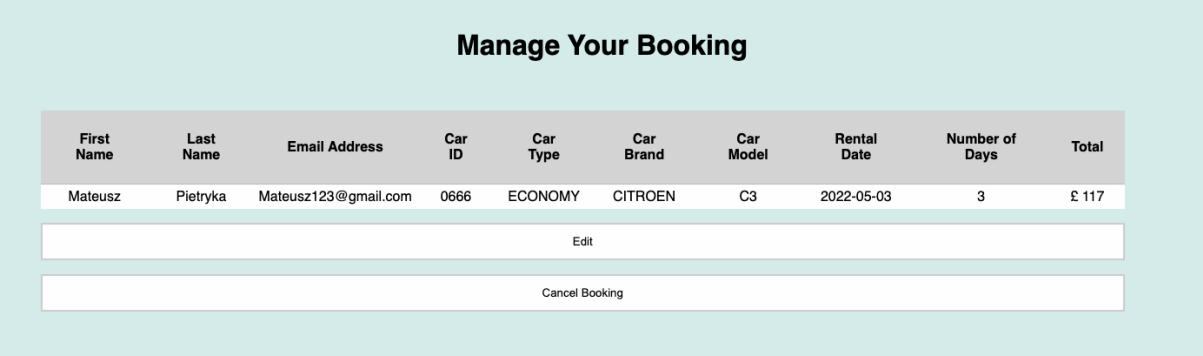
Outcome: Suitable error message displayed.

- **Test Case 22 –** Valid Booking ID entered.

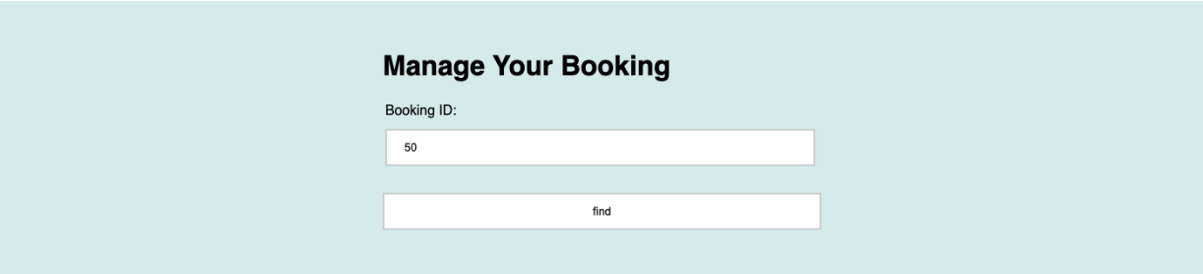


Outcome: Entry accepted, no error message was displayed.

After hitting the “find” button, user is taken to another page that allows them to edit and cancel their booking.

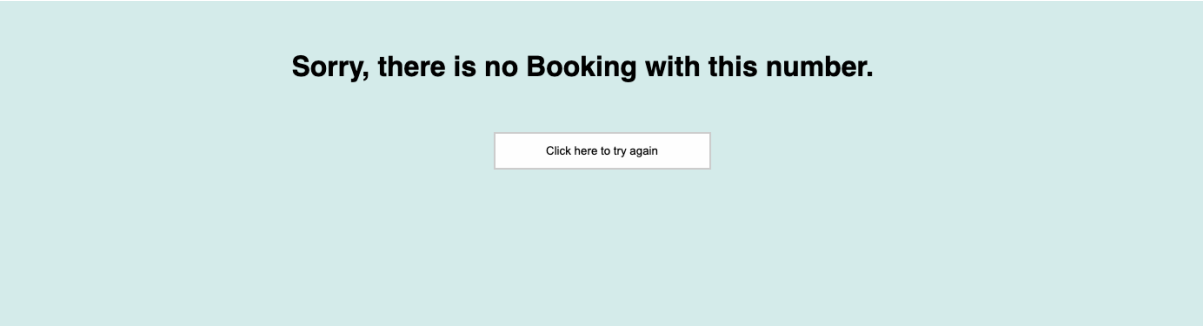


- **Test Case 23 –** Valid Booking ID of a non-existing booking entered.



Outcome: Entry accepted, no error message was displayed.

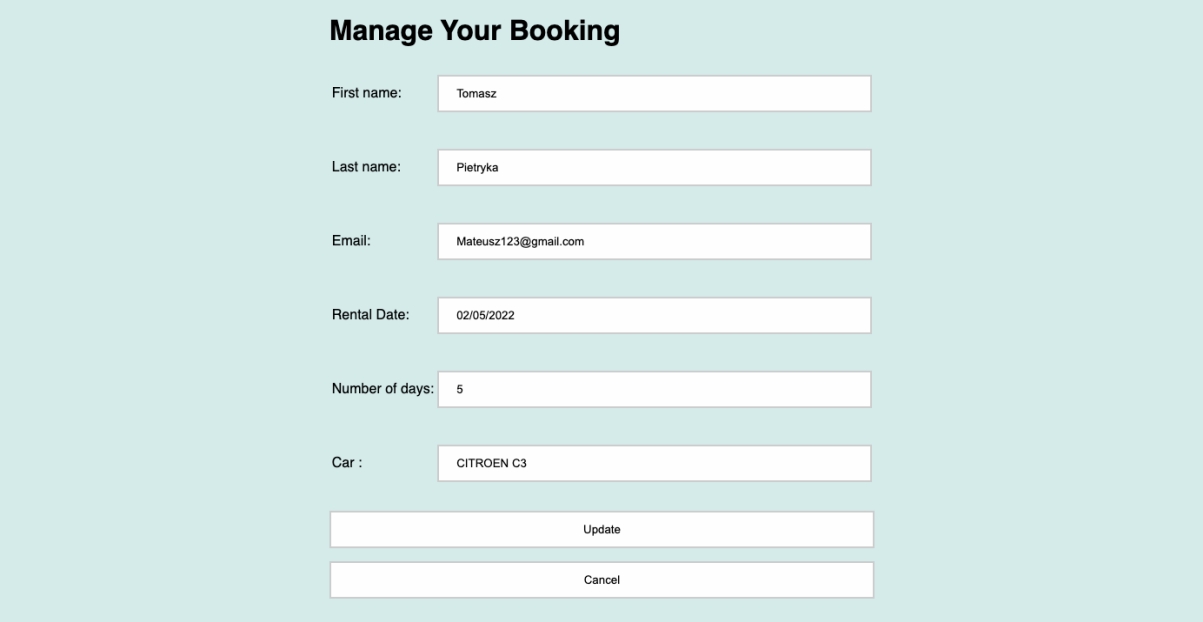
After hitting the “find” button, user is taken to another page that informs them that the booking they’re looking for does not exist. The user will be offered to try again, clicking on “Click here to try again” button will take the user back to “Manage Booking” page.



Edit Booking form

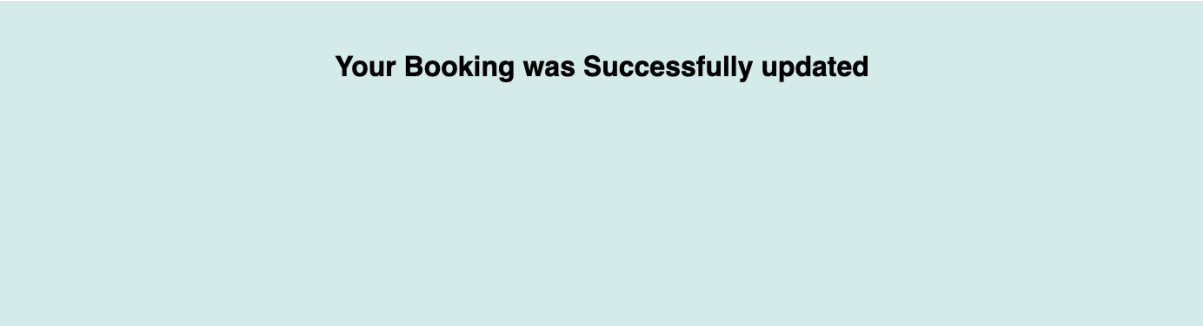
This form is nearly identical to my “Rent” form, and it’s validated in the same way. I am going to skip the testing of the validators.

- **Test Case 24 –** New values entered into the form.



Outcome: Entry accepted, no error message was displayed.

After hitting the “Update” button, user is taken to another page that informs them that the booking had been updated.



Database entry had been updated.

#	BOOKING_ID	CAR_ID	FIRST_NAME	LAST_NAME	EMAIL
1	1	0119	Liam	Manning	lmanning@gmail.com
2	3	0610	Mateusz	Pietryka	mat.pietryka@gmail.com
3	4	0619	Barbara	Dias	babad@gmail.com
4	14	0149	Barbara	Dias	babadias@gmail.com
5	16	0666	Tomasz	Pietryka	Mateusz123@gmail.com

Manage Your Booking

First Name	Last Name	Email Address	Car ID	Car Type	Car Brand	Car Model	Rental Date	Number of Days	Total
Tomasz	Pietryka	Mateusz123@gmail.com	0666	ECONOMY	CITROEN	C3	2022-05-02	5	£ 195

Edit

Cancel Booking

- **Test Case 25 –** Cancel Booking button clicked.

Your Booking was Cancelled.

After hitting the “Cancel Booking” button, user is taken to another page that informs them that the booking had been cancelled.

Database entry had been removed.

#	BOOKING_ID	CAR_ID	FIRST_NAME	LAST_NAME	EMAIL
1	1	0119	Liam	Manning	lmanning@gmail.com
2	3	0610	Mateusz	Pietryka	mat.pietryka@gmail.com
3	4	0619	Barbara	Dias	babad@gmail.com
4	14	0149	Barbara	Dias	babadias@gmail.com

Conclusion

This assignment was my first time building a Three-Tier web application, it turned out to be much more challenging and overwhelming than I initially expected. Luckily, I have approached it in an organised manner by creating a Kanban board for myself and breaking down the task at hand into a smaller tasks.

Having previous experience with SQL definitely helped me design a working database as well as the queries to implement the CRUD functionality.

I feel like I developed a strong interest in Web Development and I'm going to explore it further in the future but most likely not using the JSF technology as it is a bit outdated according to the journals I came across while researching for this project,