HCI - Seminar 1 report

Luka Boljević

November 2022

1 Introduction

In this report, we will be explaining how we followed Nielsen's 10 principles, as well as explain our choices of widgets, colors, text, etc. for assignment 4 - application for flight bookings with company Polet.

1.1 Notes

FlatLaf light theme was used. It is a part of the gradle dependencies file, included with the project.

Unfortunately third party date pickers (like JCalendar, LGoodDatePicker, etc.) did not work in NetBeans 15 for some reason.

I didn't focus too much on "unimportant" things, like "this flight is available only every Wednesday in the morning" and stuff like that. We can select any dates (so long as the outbound is before return), any flight time, any combination of the 5 cities, the plane is "empty" i.e. all seats are available etc. This was done to simplify matters and focus on the important aspects.

Unfortunately, I couldn't change the window title color of JOptionPanes - I wanted to put a custom background color on everything, but sadly, this was the thing that stopped me from doing so, as it just didn't look appealing when the window title is one color, and the background is another. I had to stick with the default background color (242, 242, 242).

All images of the UI are given at the very bottom.

2 Nielsen's 10 principles

In this section, we will explain the way we followed **each** of Nielsen's 10 principles.

2.1 Match between system and the real world

This principle guides us not to use technical terms, but to use regular words that anyone can understand, and not to limit what the user can input.

We use terms that always occur in a flight booking scenario - outbound flight, return flight, flight class, flight time, luggage, seat, etc., and when reporting errors, they are explained in an understandable way (more on that later). The user is not limited to what

information he can input. Apart from entering a valid passport/email/card number/card CVV format, the user can enter whichever name and surname, address, city and country, and he can also use any email that fits the general email format (Figures 1, 3, 6, 8).

Likewise, this principle advises us to arrange all widgets in a logical manner, according to the user's expectations. Widgets are arranged from top to bottom, left to right, just like we are used to in any application, and the flow follows what we generally expect to see when booking a flight: (1) we select the flight type (one way or return), (2) we select where we are flying from (and to), (3) we select the dates of the outbound (and return) flight(s), (4) we select the flight time, (5) flight class, (6) number of child and adult passengers, (7) we input individual passenger details, meal and luggage choice, (8) we select a seat for each passengers, and finally (9) we input the payer details and debit/credit card information. After the final step, we are presented with a summary of the booking details, before finally submitting our order.

2.2 Consistency and standards

This principle is about keeping the same language and meaning throughout the system to avoid confusing the user, and keeping the behaviour of similar looking widgets, similar.

We achieved consistency with always having the bottom 3 buttons (back, clear, "forward" - for example Figure 1) be in the same place, with the same size. The alignment of input fields/buttons and their corresponding labels across panels is always the same - the label is always the same distance above the field/button, the label and field/button are always aligned to the left. Likewise, the positions of widgets never change, they are always in the same place. The window size also does not change, so as not to strain the user's eyes by making him look all over the screen. The font and font size across the application is the same, with the exception of titles of panels, or a particular group of input information (an example might be flight destinations and dates).

2.3 Help and documentation

This principle is about helping users understand how to perform their task properly.

Every input field/button in this application is preceded by a label, denoting what it corresponds to. Name means enter your name, surname means enter your surname, etc. - stuff the user has encountered before with about 99.9% chance. For fields that may be application specific (passport number) or for fields that require a specific format to be entered (passport number, email, debit/credit card number and CVV), a tool tip is available, that explains what the correct format is. An example tool tip can be seen on Figure 18.

If the user forgot to input/select some information, or if the user entered a wrong format of some fields, he/she will not be allowed to progress forward upon pressing the "forward" button, and will be shown a popup error message (one for each error).

2.4 User control and freedom

This principle tells us that a good UI should only suggest which paths the users can take, not force or make decision for the user. We should also consider that users may regret their decision, make an error or input wrong information.

The only path that is "forced" is the logical path of entering the required information for the booking. There are no "long" operations, so nothing needs to be given the option to be interrupted, but the application can be closed at any point of course.

The application allows the user, at any point, to go back to a previous panel and check/correct the information they entered. Any input or selection which is changed/updated will be visible on the upcoming panels (for example, if the flight class is changed, the flight price for each individual passenger will also change). There is a "clear" button visible at all times, which allows the user to clear all entered information on a given panel. When we are presented with the booking summary at the very end of the application, as seen in Figure 11, we have the option to cancel/close, and correct any wrongly entered information (which as said, will be automatically detected by the rest of the reservation process).

2.5 Visibility of system status

This principle is about keeping users informed about their actions and what's happening at a given interaction.

The time it takes to switch panels is practically immediate (if all entered information is correct and no required selections/fields are empty). The user is informed if there is an error in the entered data upon pressing the "forward" button. At all times, the user knows which panel he is on (based on the title), what data he needs to input/select, and as already stated, any changes made in the entered information will immediately be visible throughout the application.

The cursor changes according to the component that is hovered over, buttons slightly change color when they can be pressed, etc. As stated before, tool tips are available for input fields that require a specific format (passport, email, card number, card CVV).

2.6 Flexibility and efficiency of use

This principle tells us that our application should not be difficult to use, and it should allow for (keyboard) shortcut for common operations. It should suit experienced and non experienced users.

Pressing the TAB button on the keyboard moves us over to the next component in logical order (like we are used to). After TAB-ing to a (radio) button, we can press SPACE/ENTER to press it. Using the up/down arrow keys on the keyboard allows the user to choose the previous/next element in a combo box, or increase/decrease the amount of passengers in a spinner. Classic keyboard shortcuts are preserved (CTRL+C for copy, CTRL+V for paste, CTRL+A for select all, etc.). We did not set default values for the radio buttons (flight time, flight class, meal, luggage) because the user may oversee that a button is by default selected, but this is not what he/she wants (for example, maybe the user wants a vegi meal instead of a classic one, but may oversee that classic is by default selected).

2.7 Error prevention

This Nielsen principle proposes that a good design should always prevent problems from occurring.

For example, selecting dates in our application is done via 3 non-editable combo boxes - one of the day, one for the month, and one for the year. This way, we prevent the user from choosing dates which don't make sense. The user can still select, for example, February 31, but these scenarios are considered and reported as an error. Selecting the outbound and destination cities is also done via non-editable combo boxes, so as to avoid possible typing errors. Selecting the amount of adult and child passengers is done via spin boxes, whose value cannot go below 0. Selecting mutually exclusive values (one way or return flight, meal, luggage, ..) is done via radio buttons.

Likewise, all information necessary to fill in the fields is shown - either via labels above the widgets, or tool tips. The entered information is of course saved after switching to the next panel, so the user can go back at any time and correct any misentered information. If data is entered in the wrong format (passport, email, card number, card CVV), the user is informed about the format and that he/she needs to fix the entered information. Likewise, as mentioned, the user cannot leave the current panel unless he has entered all information. If the user pressed the "clear" button, he/she needs to confirm one more time that he/she actually wants to clear, as seen on Figure 13. On the last, payment panel, prior to "finalizing" the order, the user is shown a summary of their booking after clicking the "checkout" button, so they can double check all entered information in one place, and correct if necessary. Confirming the order presents us with a confirmation message, as seen on 12, to ensure us that everything went well.

2.8 Recognition rather than recall

This principle tells us that the user shouldn't have to memorize anything when using an application, because it's always easier to recognize rather than remember.

This principle is in a way similar to the aforementioned "visibility of system status" - all information should be (and in our case, is) visible and the user should immediately recognize what it is. We also believe the user doesn't have to memorize anything when using the application, apart from his personal information and where he wants to travel to/from, and when. In other, simpler words, we believe a user's brain can be totally turned off and they can still book a flight without any issues.

2.9 Help users recognize, diagnose, and recover from errors

The second to last principle is about letting the users know there was error, in an understandable and simple language, with the steps on how to fix it included.

We believe that error detection and recovering has been covered enough while explaining how we followed the previous principles. In essence, the error is explained in a simple way, and suggestions on how to fix it are shown. Example error messages are given on Figures 14, 15, 16, 17.

2.10 Aesthetic and minimalist design

The last Nielsen principle suggests that the UI should be as minimalistic as possible, with only the essential information being shown to the user.

There are no unnecessary visual elements, images or text that will distract the user. Widgets are likewise grouped together, so similar information is part of one small "box". We used as little text as possible (labels above widgets are just one or two words, errors

are explained as detailed as possible in as few words as possible, etc.). There are only a few colors used.

3 Designing the UI

3.1 Selecting appropriate widgets

- Main panel - Figures 1, 2
  - Flight/trip type (one way, return) - 2 mutually exclusive options that are commonly used, really the only logical choice are radio buttons. Choosing a one way flight enables the selection of return dates.
  - Origin and destination cities - typos are generally possible, so a list or combo box (drop down menu) are viable options, but since we're lacking on space, and there could easily be added cities in the future, a combo box is the better alternative.
  - Outbound and return dates - generally, we would choose a date picker, because it looks the nicest and feels easiest to use, however NetBeans 15 for some reason did not allow the use of third party date pickers (like LGoodDatePicker for example), so we went for the next best option, which are combo boxes. Lists don't make sense as they would take up too much space. A combo box is the most viable as we don't bother the user with "enter a date *in the following format*" or worry him/her that he/she might make a mistake while typing.
  - Flight time and class - similar argument as for flight type, so radio buttons are the only logical choice.
  - Number of adult and child passengers - a spinner/spin box feels the most viable, as the data is "ordered" (we can have 1, 2, 3, 4, ... adult passengers), typing in the number is sometimes desired, no mistakes can be made in this way, and in theory we don't limit the amount of passengers there can be (which would be one minus with using a combo box in this scenario, for instance).
- Individual passenger details - Figures 3, 4, 5. All personal data is entered via a text field, nothing to explain here, as there is simply no other choice (other than a text area, but that is used for when there's a lot of text). Meal and luggage choices are, again, groups of 2 mutually exclusive options that are commonly used, so radio buttons are the best choice again.
- Seat selection - Figures 6, 7. A seat selection is made by simply selecting which passenger we are selecting the seat for, and pressing the button in the corresponding row and column. This is the most "visual" and striking way to allow the user to choose a seat, as having combo boxes, lists, or something similar just doesn't look practical.
- Payment panel - Figures 8, 9, 10
  - Copy payer information from passenger - this only makes sense that it's a combo box. We don't have space for a list, values can easily change, typing the name out would be prone to errors and would take longer than 2 clicks. Usage can be seen on Figure 9.

  - Payer information - again, similar to passenger details, these are personal information, so we don't want to limit the user input in any way, and we give the user full freedom of what information he types in. Of course, we rely on the user to input correct information.
  - Card details - similar as payer information, with the exception that the card number and card CVV inputs are checked for the correct format (card number is a 16 digit number, CVV is a 3 digit number). We allow the user to use a space or a "-" between each group of 4 digits. Again, we choose the month and year of expiration with 2 combo boxes, as we don't have a date picker.

- Error messages - shown as a popup window, even though there are definitely multiple options for this. Example error messages are on Figures 14, 15, 16, 17

3.2 Arranging the widgets

First of all, similar information is grouped together into separate subpanels (flight type is one group, destinations and dates are one group, flight time and class are one group, payer details are a group, etc.), with enough horizontal and vertical whitespace between the components, to emphasize individual fields/selections and groups.

All components/widgets use the same font with the same size and thickness, apart from panel/group titles, which just have a bigger size. All components are arranged top to bottom, left to right, which is in line with how we are used to operate in any application. No panel is overcrowded with information, it shows the user exactly what he needs to see.

Labels are aligned with the left edge of their corresponding widget. Widgets that are one below another have their left and right borders aligned - for example, the combo box for selecting an origin city is aligned on the left and right with the combo boxes for selecting an outbound date. The tops and bottoms of widgets in each row are on the same level (i.e. there is nothing "sticking out"). If there are multiple text fields in one row, they are equally spaced apart.

3.3 Text, color and image choices

Here, we'll explain our choice of text, color and images.

The language and words used are short, understandable, not technical - which is line with Nielsen's principles. There are no images, as they are not necessary for anything. We stuck with simple colors, to follow Nielsen's minimalistic design principle - black for the foreground, gray for the background (sadly, had to be - look at notes at the beginning), and a lighter blue for the backward, clear and forward buttons.

3.4 Feedback

As explained before, all operations are fast and the time it takes to switch panels is practically immediate, so no feedback needs to be given here. Again, the user is informed of errors in his input, one by one, with a popup window. If the user clicks the "clear" button, a popup window to confirm the clear is shown. Similarly, when the user is at the payment panel, and wants to checkout, he/she is first presented with a summary of the booking details, before actually submitting his/her booking. When the booking is submitted, a

popup window tells the user that everything is successful, and that the tickets and receipt will be sent to his/her email.

3.5 User and their goal analysis

User analysis - the user is an employee at airline Polet, with good computer knowledge, communication skills, and friendly approach to customers. They should be fully capable of using an application like this, since it was created with the aim that anyone can use it, even people outside of th airline company.

User goal analysis - the end goal of the user is to book a flight, for him/her or someone else (which is what our employee will be doing). They need to be able to input any sort of personal information, and in the right format (passport, email, card number, card CVV).

4 Images

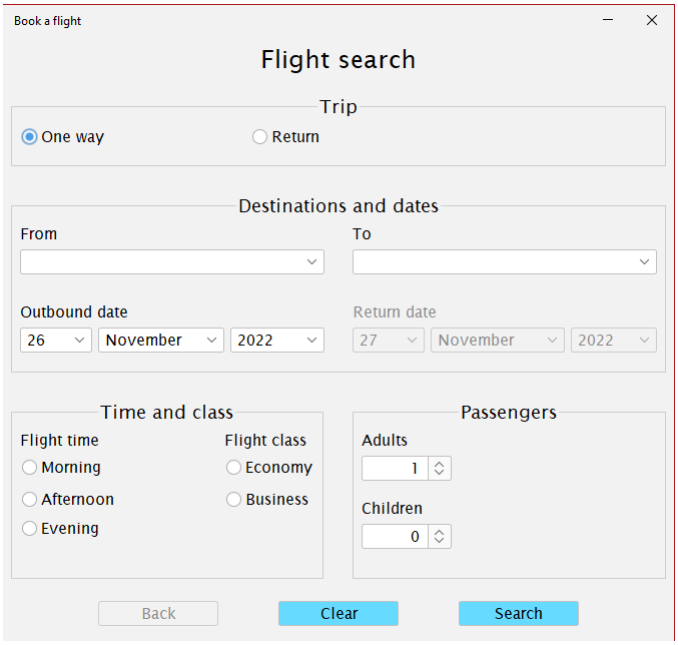

Figure 1: The first panel that is shown when the application is ran (so called main panel). We see that, since the one way radio button is selected, we cannot choose a return date. The back button is disabled since we can't go back to any panel.

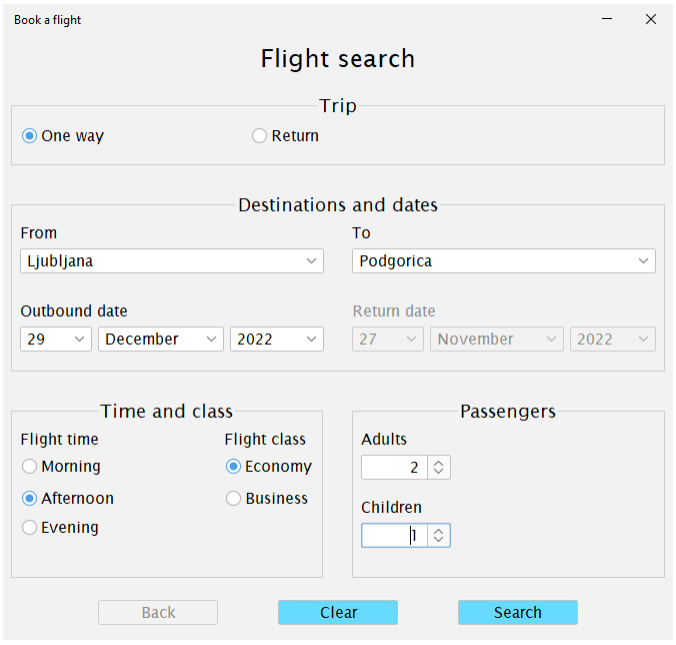

Figure 2: Main panel filled in with information.

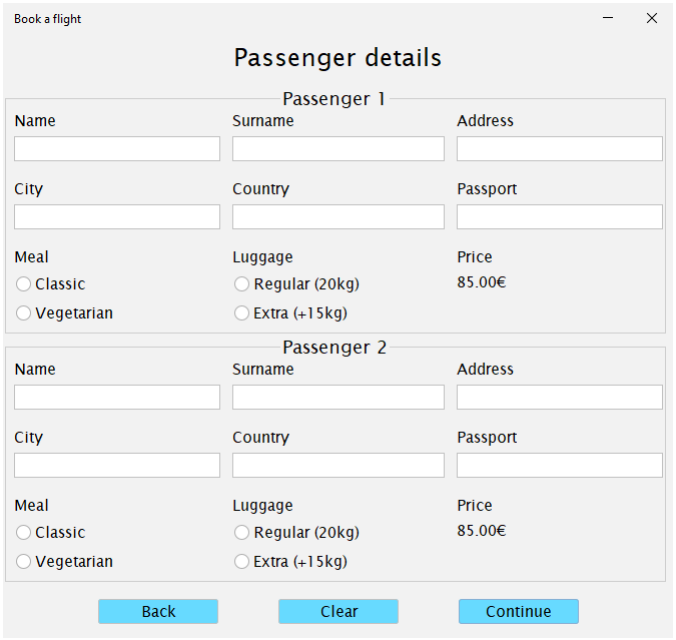

Figure 3: Passenger details panel. Each passenger can input their own details and choose their meal and luggage options. The flight price for each individual passenger is automatically calculated based on the selections from the main panel and luggage option (meal is included in the price). A tool tip is available for the allowed passport format.

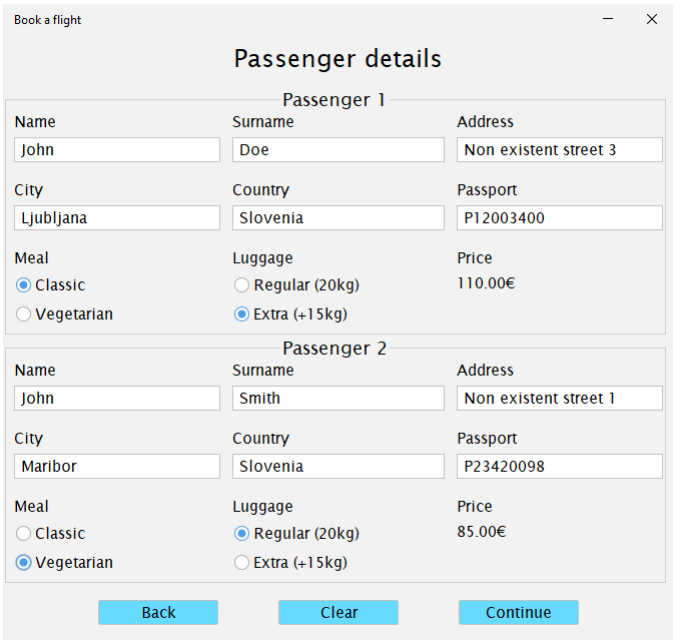

Figure 4: Passengers entered their information.

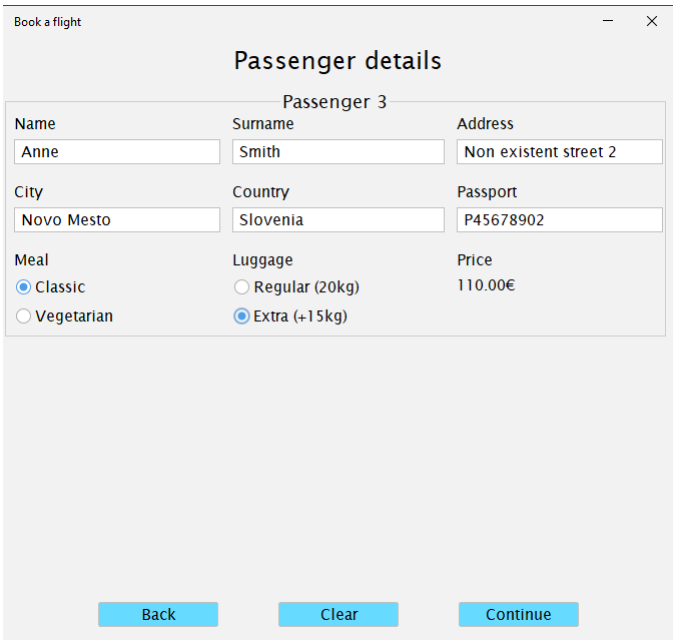

Figure 5: If there are more than 2 passengers, the input fields for them are shown on the next page, so as not to have too high of a window.

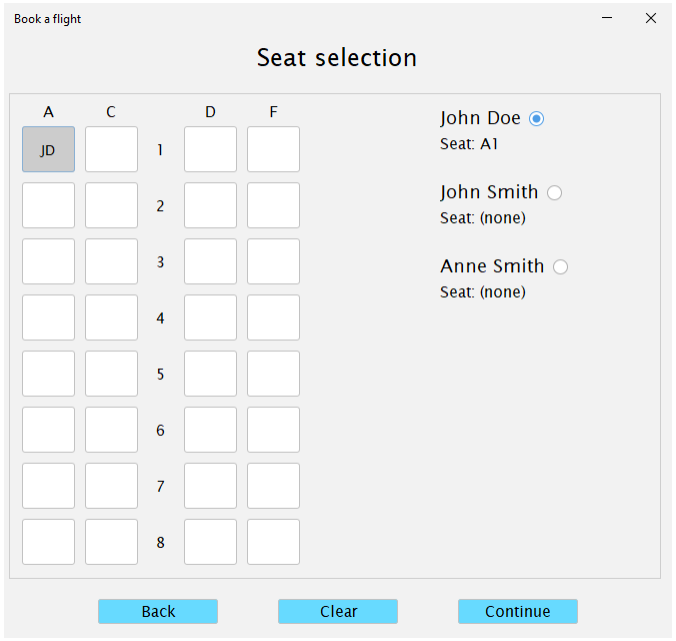

Figure 6: The seat selection panel. On this panel, all passengers choose their seats. In this figure, John Doe has already selected seat A1. A passenger can choose a seat for him/herself by clicking the button next to his/her name first, and then selecting a seat.

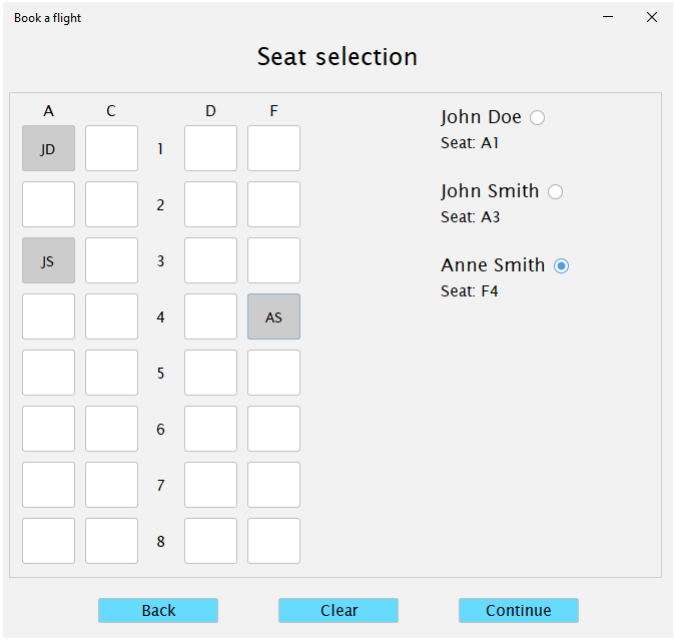

Figure 7: Everybody selected their seat.

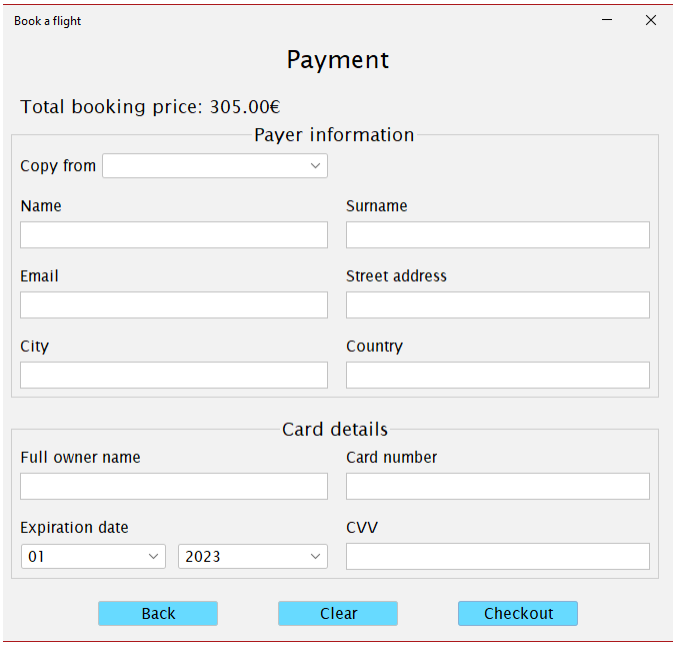

Figure 8: The last panel - the payment panel. We can see that the total booking price is listed at the top, so the payer immediately knows how much he needs to pay. The payer needs to enter his personal information, and his/her/someone else's card details. A tooltip is available for the email, card number, and card CVV input fields.

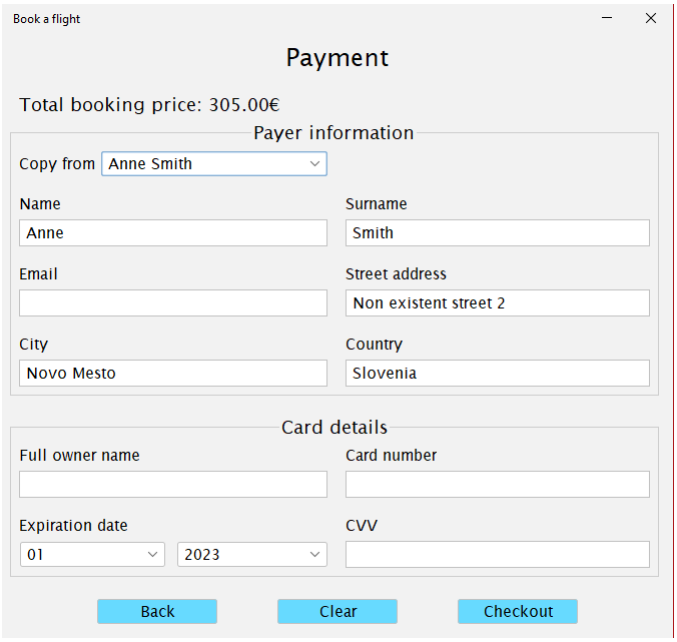

Figure 9: The payer can of course be one of the passengers, so we allow the user to copy the entered details from an existing passenger (excluding the passport number).

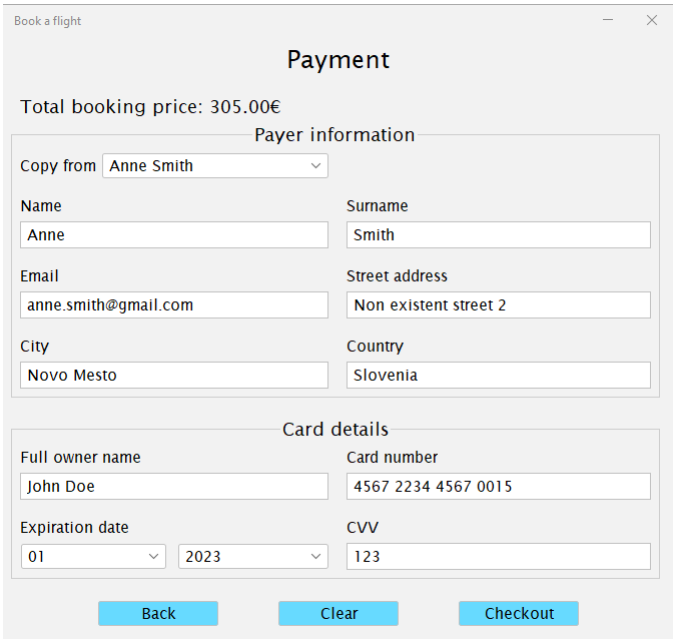

Figure 10: Payment panel filled with details. The user can proceed to checkout.

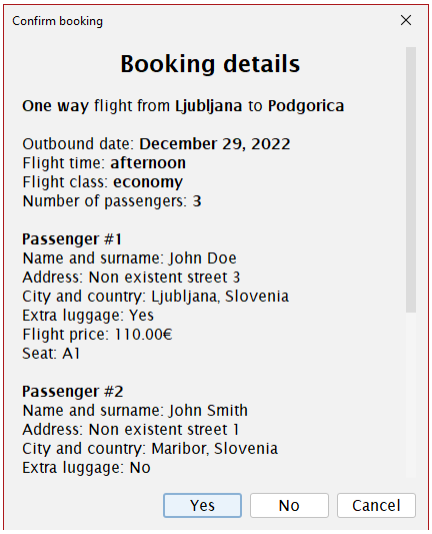

Figure 11: Before checking out, a summary of the entire booking is given to the user, so they can double check all entered information. Should some info be wrong, the user can cancel, go back and correct the misentered information. If there is a lot of information, the user can scroll down to check all of it.

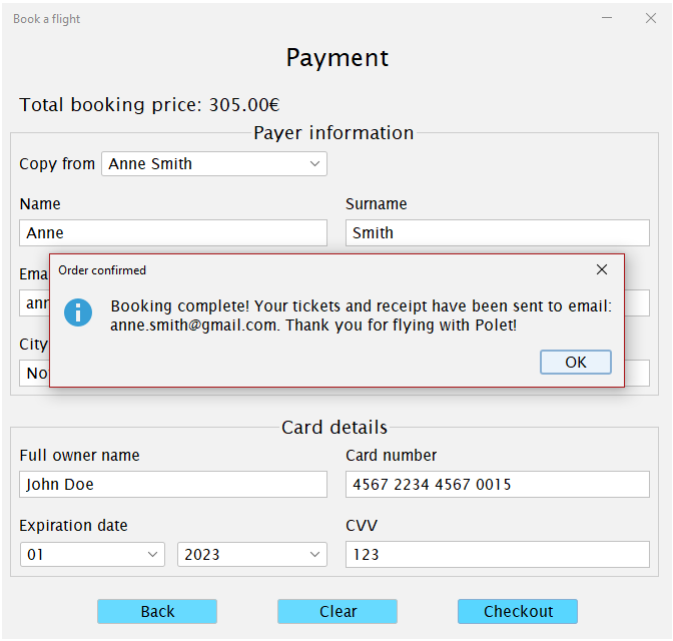

Figure 12: Booking is complete. The user is told he will receive all necessary information via the entered email.

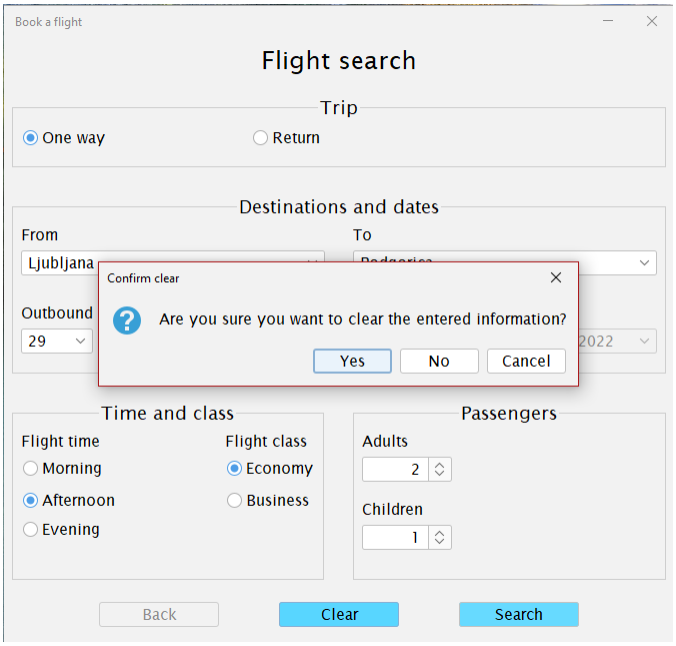

Figure 13: After clicking the "Clear" button, the user is asked to confirm his choice, so as to avoid accidental clicks of the "Clear" button.

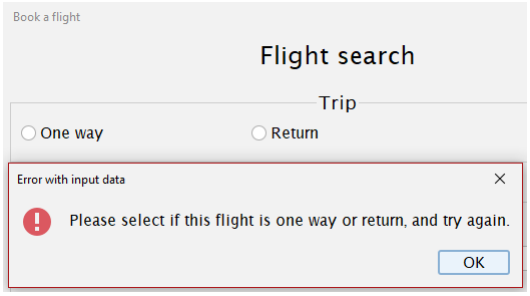

Figure 14: After clicking the "Search" button, if no flight type is selected, an appropriate message is shown, and the user is not allowed to proceed.

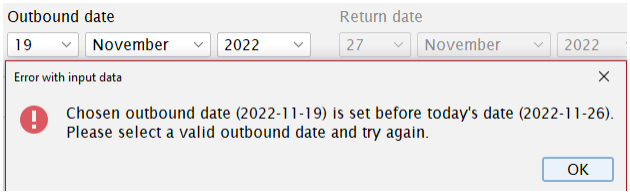

Figure 15: If the outbound date is in the past, an appropriate message is shown.

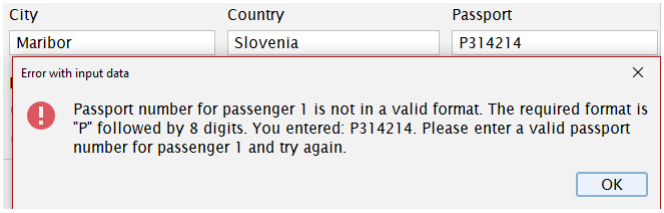

Figure 16: If the passport format is not correct, the user is informed, and is told what the correct format is, so he does not have to guess what it is.

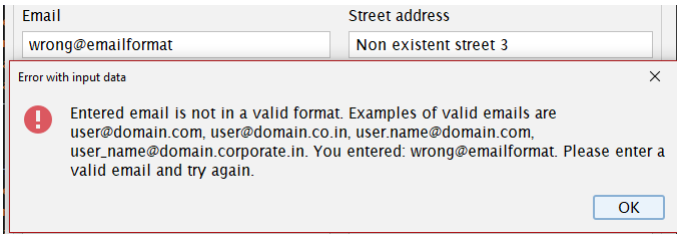

Figure 17: If the email format is not correct, the user is informed, and is told what the allowed formats are.

Figure 18: One of the available tool tips (in this instance, the email tool tip. Tool tips are available for the passport format, card number format, and card CVV format).