

FlyUX: Booking a Flight

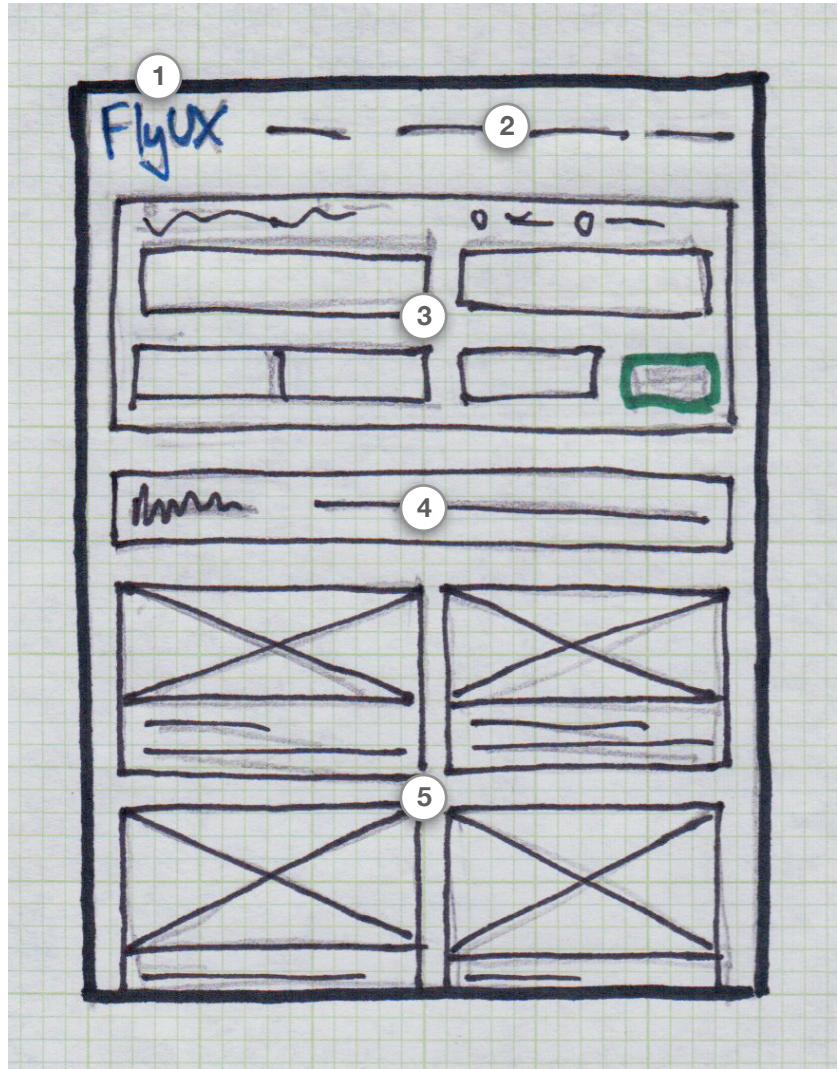
Wireframes (version 1.0)

This document presents some initial design ideas in the form of low-fidelity wireframes. The focus at this stage is on the booking process - up to and including the selection of seats.

The wireframes can be used to further consider the pages, interactions and micro tasks involved in the customer journey for booking a flight. It may help to look at this document in conjunction with the flow diagram and customer journey documents.

The FlyUX home page	2
The home page booking form	3
Selecting “from” and “to” airports	4
Selecting flight dates	5
Selecting passengers	9
The flight selection page	10
Refining the date and selecting a flight	11
The selected flight detail	12
(The booking progress stepper)	12
(The price/continue panel)	13
The add baggage page	14
The upgrade panel	15
Baggage options UI	16
Select seats page	17
Seat selection panel	18
Passenger selection panel	19

The FlyUX home page



The main parts of the FlyUX home page:

- 1 The FlyUX Branding
- 2 The main menu for the whole site
(e.g. Home, About, Book, FAQs)
- 3 [The flight booking form](#)
- 4 Announcements
- 5 Promotions

It is important to note that the booking form needs to be prominent and obvious. The primary user goal we are trying to solve is that of booking a flight (or researching to book later).

The home page booking form

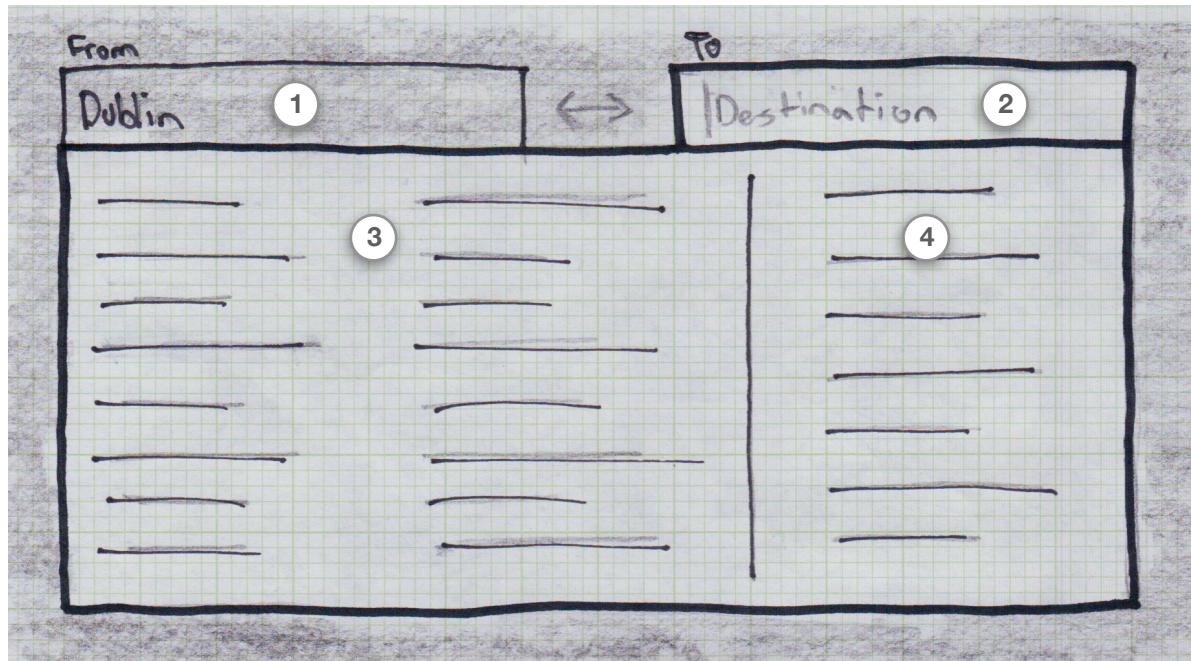
A hand-drawn wireframe of a flight booking form on lined paper. The form includes fields for departure and destination, flight dates, passengers, and a search button. Annotations with numbers 1 through 6 point to specific UI elements: 1 points to the heading 'Book your flight'; 2 points to the 'Return Trip' radio button; 3 points to the double-headed arrow between the 'From' and 'To' fields; 4 points to the 'Flight Dates' section; 5 points to the 'Passengers' field; and 6 points to the 'SEARCH' button.

- 1 Heading needs to be prominent, clear and instructive.
- 2 Allow the user to toggle return vs one-way flight.
- 3 Departure airport is preselected (use preferences or location)
- 4 [Flight date selection](#) is very important UI.
- 5 [Passenger selection](#) is a micro-task.
- 6 Use a strong call to action. Consider using “Find Flights”.

Additional points to note:

- Selecting a “one way” flight will update the booking form such that “Flight Dates” will become a single “Departure date” field.
- Departure airport is pre-selected based on user preferences. If there is no preference we fall back to a best guess based on location.
- “1 Adult” has been pre-selected (but can easily be changed).
 - **Note:** We should check site stats to find what the most common passengers is (e.g. “2 Adults”).
- The next field’s micro-task should be activated whenever the current field’s micro-task is complete (reducing number of clicks and leading the user onwards through the form).

Selecting “from” and “to” airports



This is a departure *input field* with “Dublin” as the value.

- ① Clicking this field will switch the UI so that the user is now selecting a departure airport.

This is a destination field with “Destination” as placeholder text. It shows as currently active. Typing in this field will filter the lists below. Selecting an airport from the list (4) will populate this field.

- ② This is a list of countries that FlyUX services. Clicking a country will show the list of airports for that country.

③ This is a list of the airports for a selected country. Clicking an airport from this list will populate the field and move to the next micro-task in the booking form (selecting dates).

Additional points to note:

- All other areas of the booking form are greyed out when this UI is active - bringing the focus to this micro-task.
- By default, when no country is selected, we should show a list of the most popular airports in the right hand side (4).
- Individual country names on the left (3) should be greyed out if:
 - ... the departure airport does not fly to those destinations.
 - ... a country has already been selected (in this case we'll highlight the selected country and fade back the others).
- When a departure airport is selected the UI should automatically switch to destination airport selection. When a destination airport is selected we automatically switch to passenger selection.

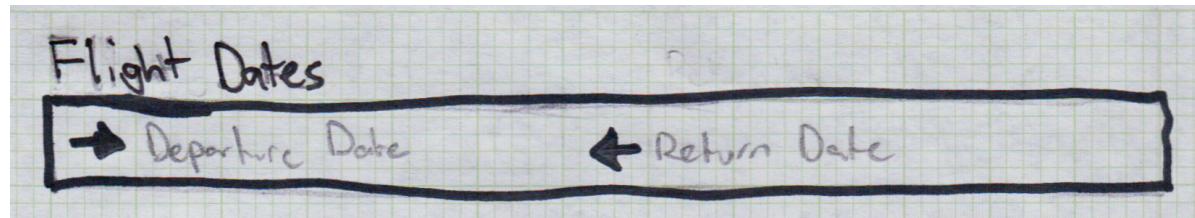
Selecting flight dates

Selecting flight dates is an important micro-task that caused some confusion during testing. It's important to get it right.

This reasonably complicated micro-task is broken down into the following sequential wireframes.

1. The default state (no dates selected)
2. Departure date is active
3. Departure date selected, return date active
4. Both departure date and return date selected

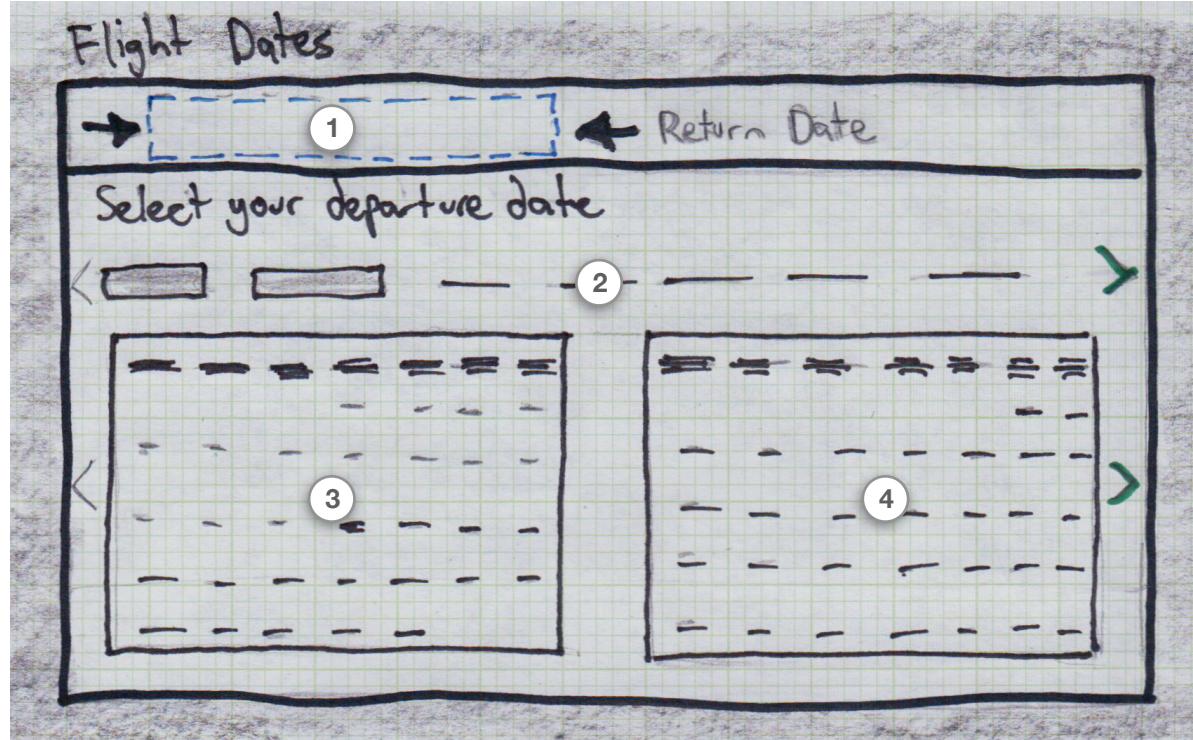
1. The default state (no dates selected)



"Departure Date" and "Return Date" are placeholder text for the date input fields.

Note: When the destination airport is selected, the departure date micro-task should automatically be activated. In other words, this default state will be skipped in the most common user flow (though it will show until the destination airport is selected).

2. With the “Departure Date” field active

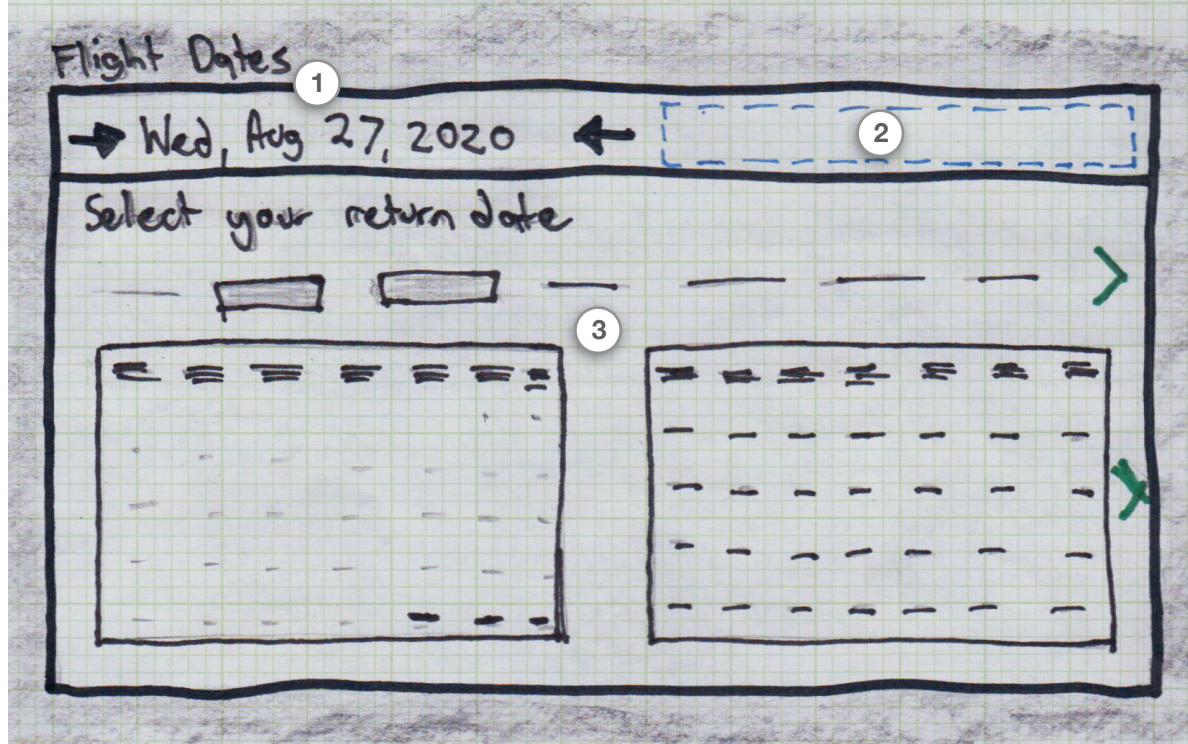


- ① It should look obvious that the departure date field is selected.
The current and next months are selected by default. The user can click a different month (which will show that month and the next month). The user can use the arrows to select months further into the future.
- ② The dates before the current date are disabled. If possible, dates on which there are no flights are disabled.
- ③ Showing two months at a time makes it clearer for trips up to a month or so in duration.

Additional points to note:

- Arrows for the calendars allow the user to easily scroll through to the next month.
 - The user can scroll back as well - but only to months that are not older than the current date.
- Arrows for the list of months will allow the user to scroll to months further into the future - but the calendars will only update when a month is actually clicked on.

3. Departure date selected, return date active

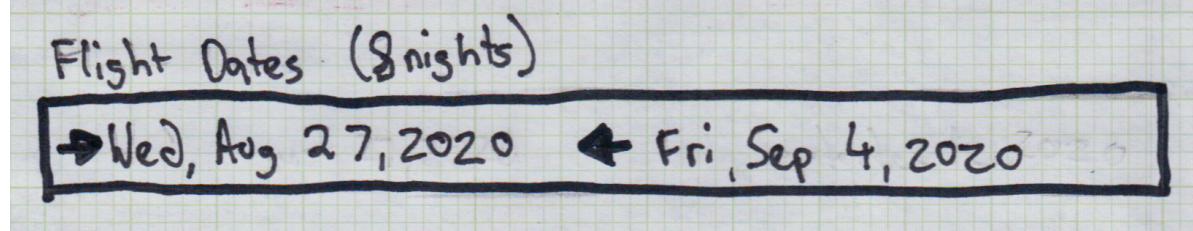


- 1 The date selected in step 2 (above) now populates the departure date field. Clicking into this field will revert back to step 2.
- 2 It should look obvious that the return date field is currently selected. This is reinforced by the instructive "Select your return date" text.
- 3 The dates before the departure date are disabled. If possible, dates on which there are no flights are disabled.

Additional points to note:

- As the user hovers over the dates (for the return date) we should highlight the range of days from the departure date to the currently hovered over date.
- If possible, it would be good to label the highlighted range of dates as "X nights".
 - We'll need to investigate the feasibility of this in terms of interaction design. Would it be too cluttered?
 - We show the duration (in number of nights) in step 4 anyway so maybe it's not necessary. Might be a good thing to test.

4. Both departure date and return date selected



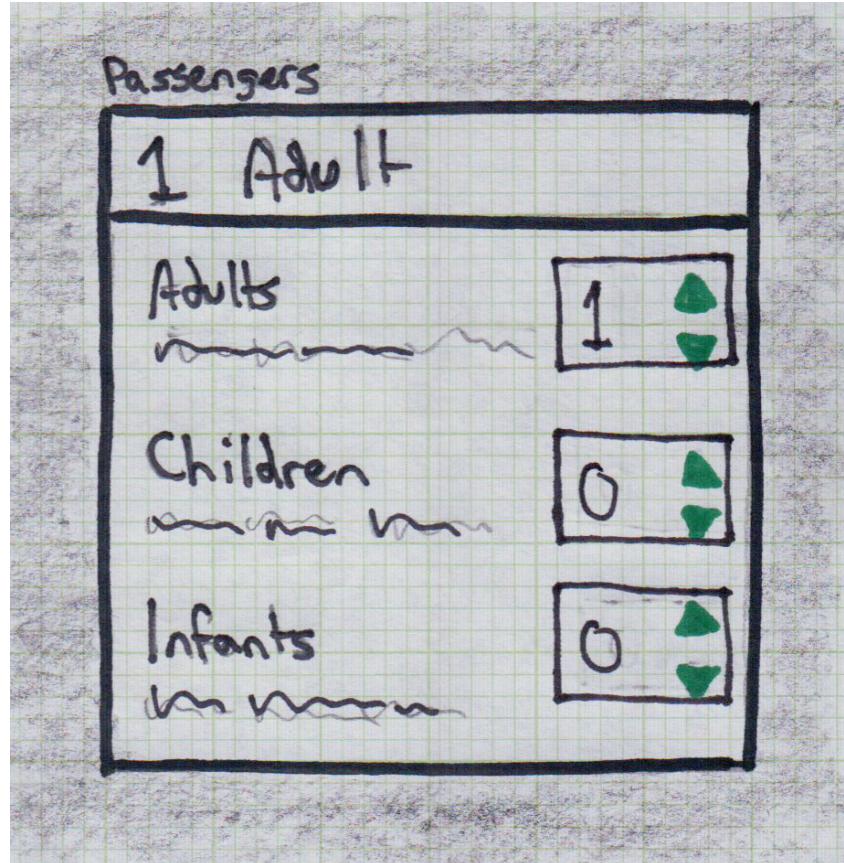
Note that both of the date fields are now populated. In addition we show the duration of the trip as the number of nights.

The user can click on either of the date fields to go back to step 2 or 3.

Additional notes about selecting dates

- Steps 2 and 3 above should grey out the rest of the booking form UI - to help focus on the date selection micro task.
- When the dates are selected, the focus will automatically switch to the select passengers micro-task.

Selecting passengers

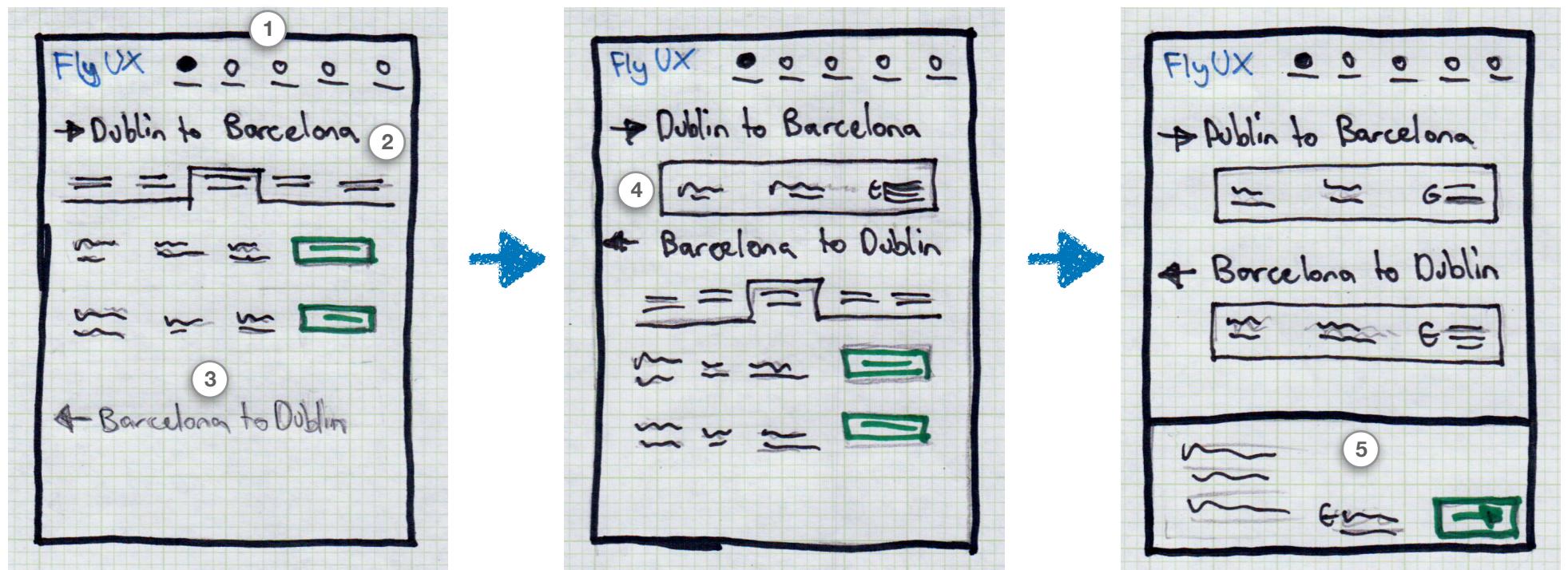


As with the other micro-tasks, the rest of the booking form should be greyed out while this micro-task is being completed.

The user is able to type directly into a field or they can use arrow buttons to increase or decrease the number of passengers in each group.

Note: We should determine what the best default value should be for this field (based on researching site usage/booking data).

The flight selection page

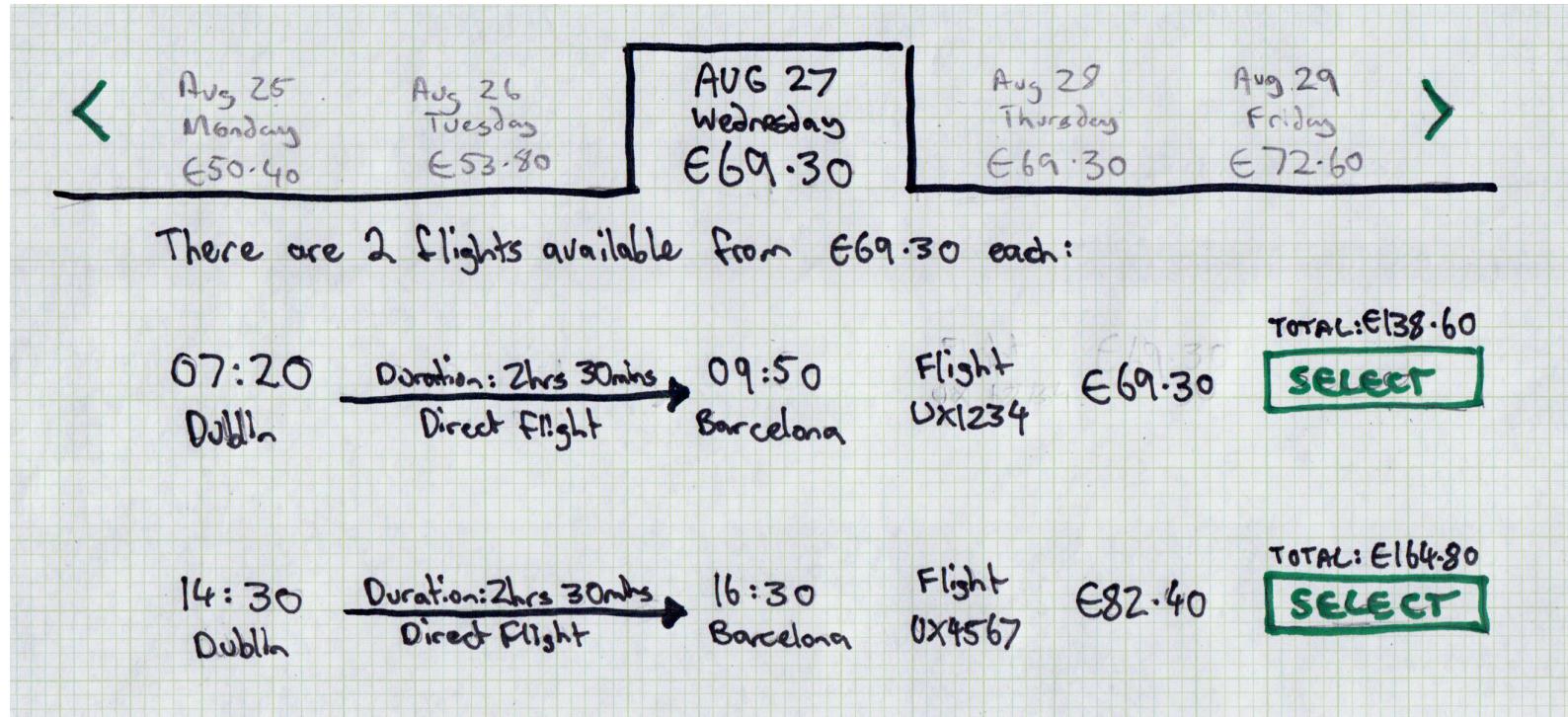


- 1 The main navigation is replaced with a [progress stepper](#).
- 2 [Flight selection](#) for outgoing flight.
- 3 The return flight selection is disabled until the outward journey has been selected.
- 4 When the flight has been selected we show [the flight details in a selected state](#).
- 5 When both flights have been selected, we show a [price and continue panel](#) that will be anchored to the bottom of the screen throughout most of the rest of the booking process.

Additional points to note:

- As per the flow chart document, the process is as follows:
 1. User refines their date selection (optional)
 2. User selects an outward flight
 3. User refines their return date selection (optional)
 4. User selects a return flight
 5. User clicks continue (in the bottom panel)
- Note that we do not show fare types in this UI. Instead we will show an upgrade package on the next screen (when the user adds baggage).

Refining the date and selecting a flight



Points to note:

- The user can select a different date
 - The rules for disabling dates in the calendar of the booking form will apply here as well (e.g. cannot select a date before today).
- The currently selected date is prominently indicated. The other dates are clickable but less prominent.
- We show a summary of the number of flights available for any given date (to be clear and avoid confusion when only one flight is available for the given date and destination).
- We associate a clear total price (for the number of passengers) with the “Select” button.
- When the “select” button is clicked, the selected flight will be shown in a “selected” state. And additionally...
 - The return date UI will be activated in the case of a selected outward flight.
 - The price/continue panel will be shown in the case of a selected return flight.

The selected flight detail



When a flight is selected, we change the layout of the flight details slightly such that:

- We now show the date of the flight
- We remove the “Select” button and replace it with a summary of the flight price (being sure to show the total price for the number of passengers).

- We add a “Change flight” link so that the user can change their flight (reverting to the refining the date and selecting a flight UI).

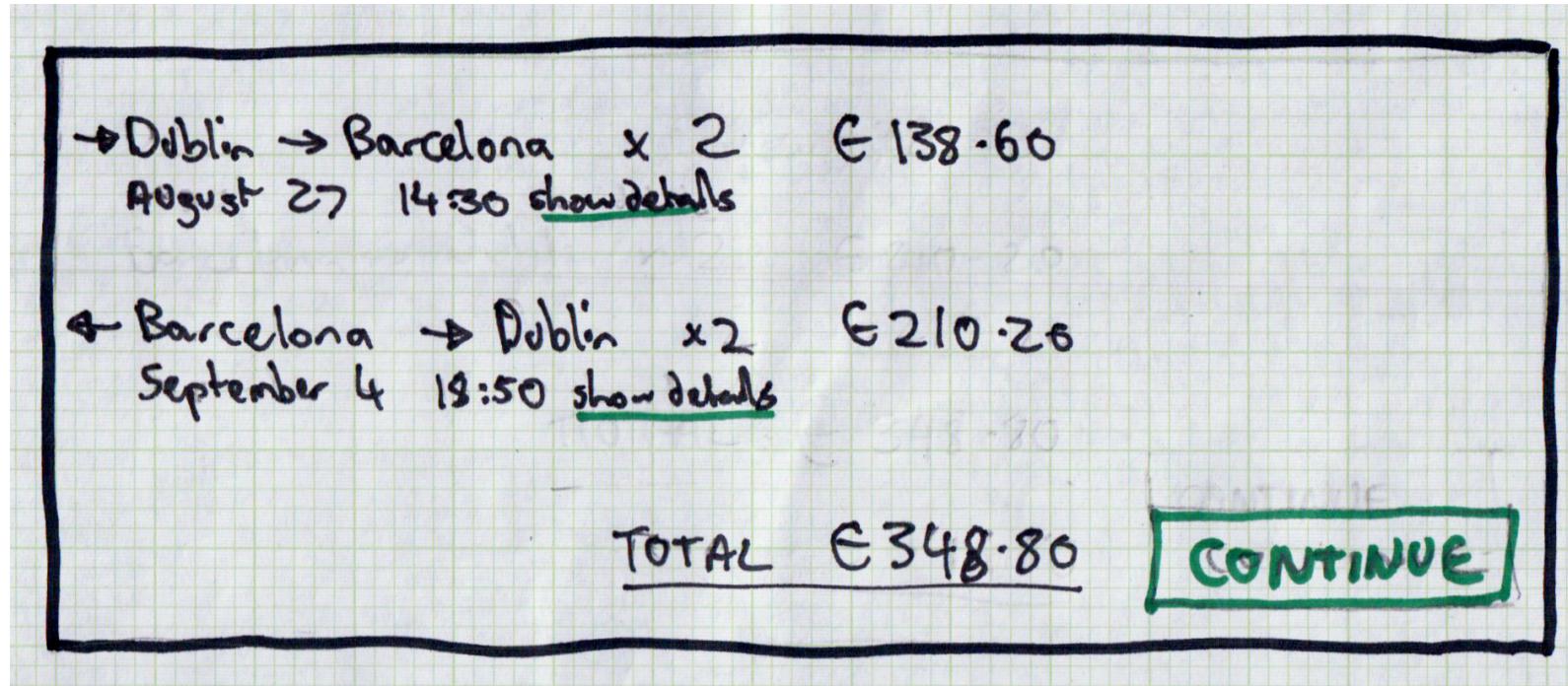
(The booking progress stepper)



This stepper is shown at the top of each page throughout the booking process. The current page (or step) of the process is highlighted to show the user where they are in the process and how far they have yet to go.

Note: The steps in the stepper are not links - e.g. you can't skip ahead to payment.

(The price/continue panel)

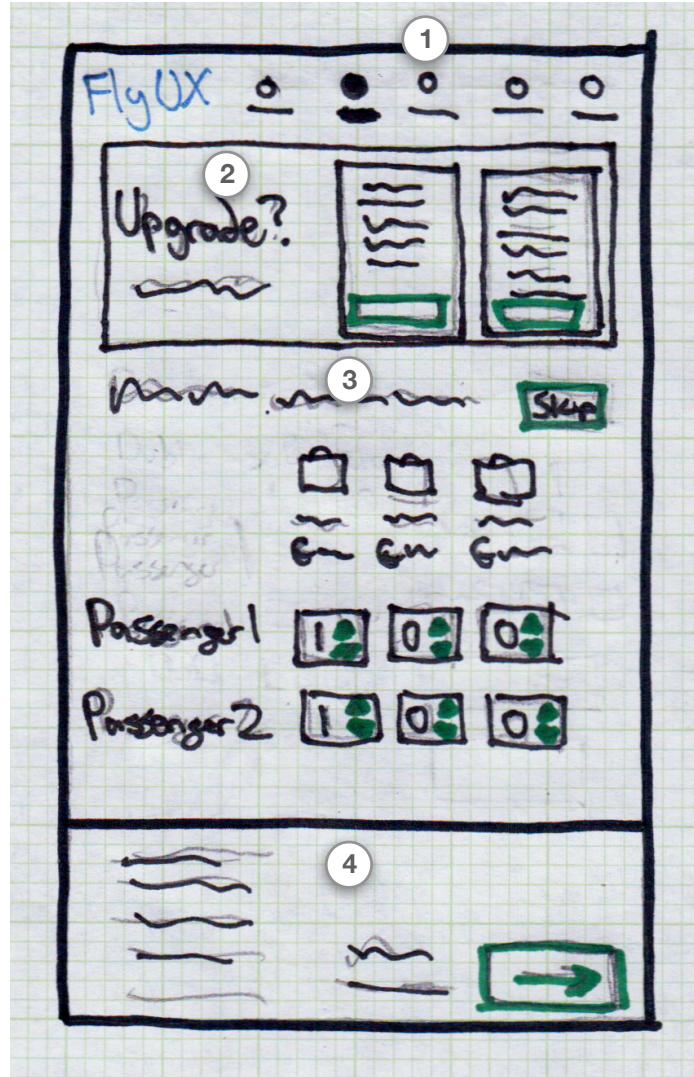


This panel is shown at the bottom of the booking pages after the flights have been selected and for the rest of the duration of the booking process (until the payment screen).

- The price inventory will update as the user makes adjustments to their booking (e.g. if they choose an upgrade package or pay for baggage or seats etc).
- The show details links will show the flight information etc for the corresponding flight.

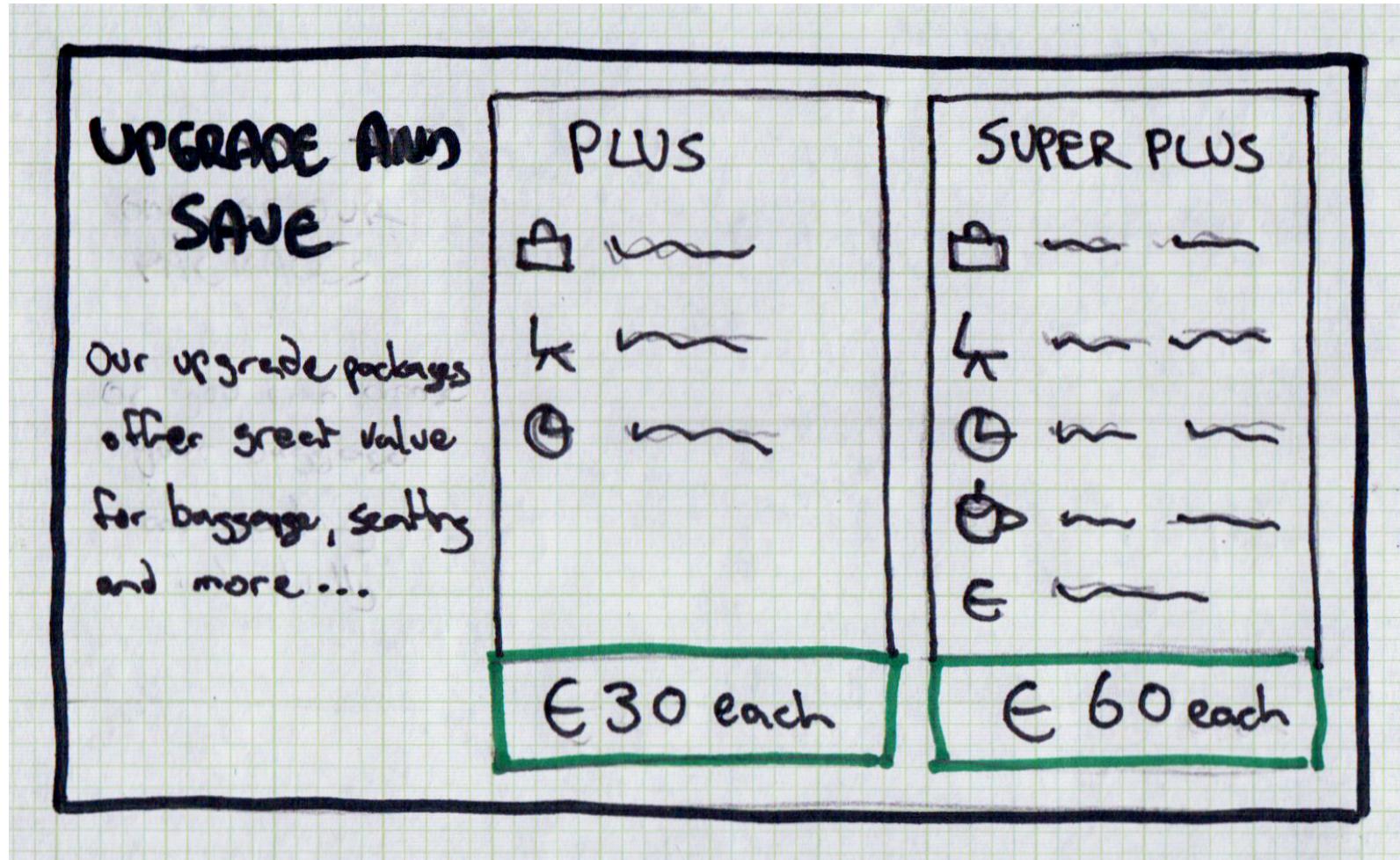
- The “continue” button will be disabled on pages where user input is required (e.g. on the “your details” page).
- This panel should be anchored to the bottom of the page so that it is visible at all times.

The add baggage page



- ① The progress stepper is updated to reflect that we are now in the second step of the booking process.
- ② [The upgrade panel](#) allows a user to select an upgrade package - which may include additional baggage or seating options.
- ③ The [baggage options UI](#) is elaborated on below.
- ④ The [price summary](#) will update as selections are made.

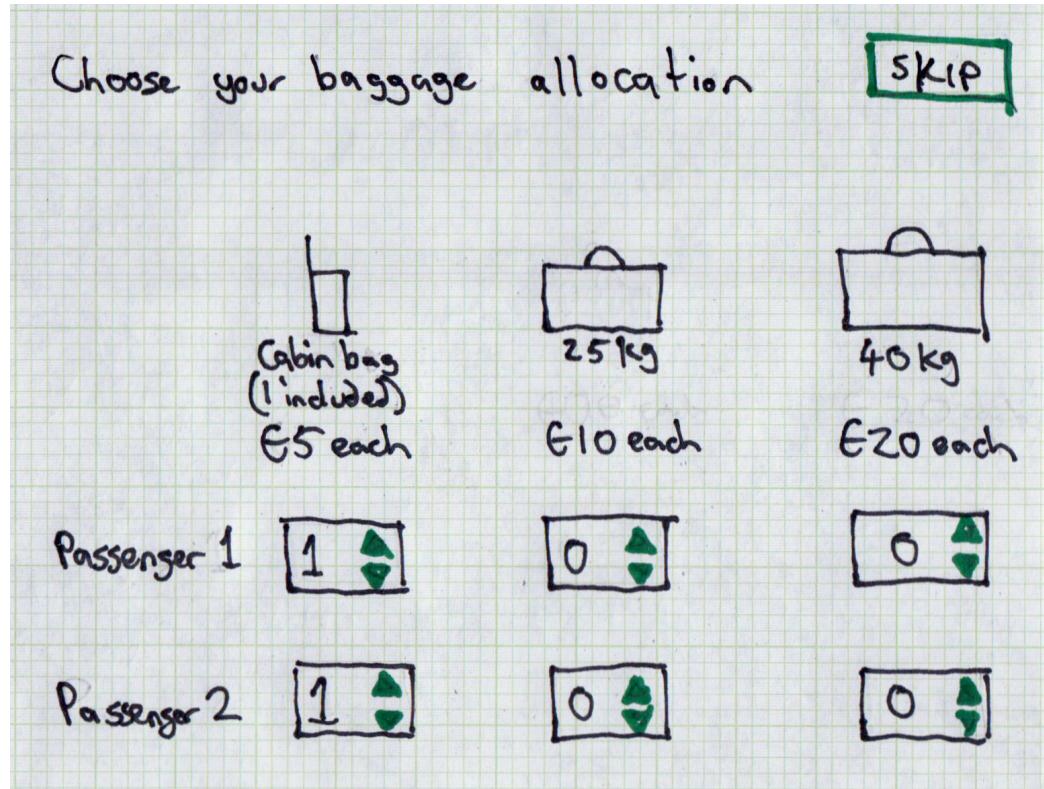
The upgrade panel



The user is able to upgrade their flight to receive an additional baggage allowance and/or allocated seating etc.

If the user selects one of these options, the price panel will update accordingly. In addition, the defaults for baggage allowance (and seating) will be updated to reflect that the upgrade package has been applied.

Baggage options UI



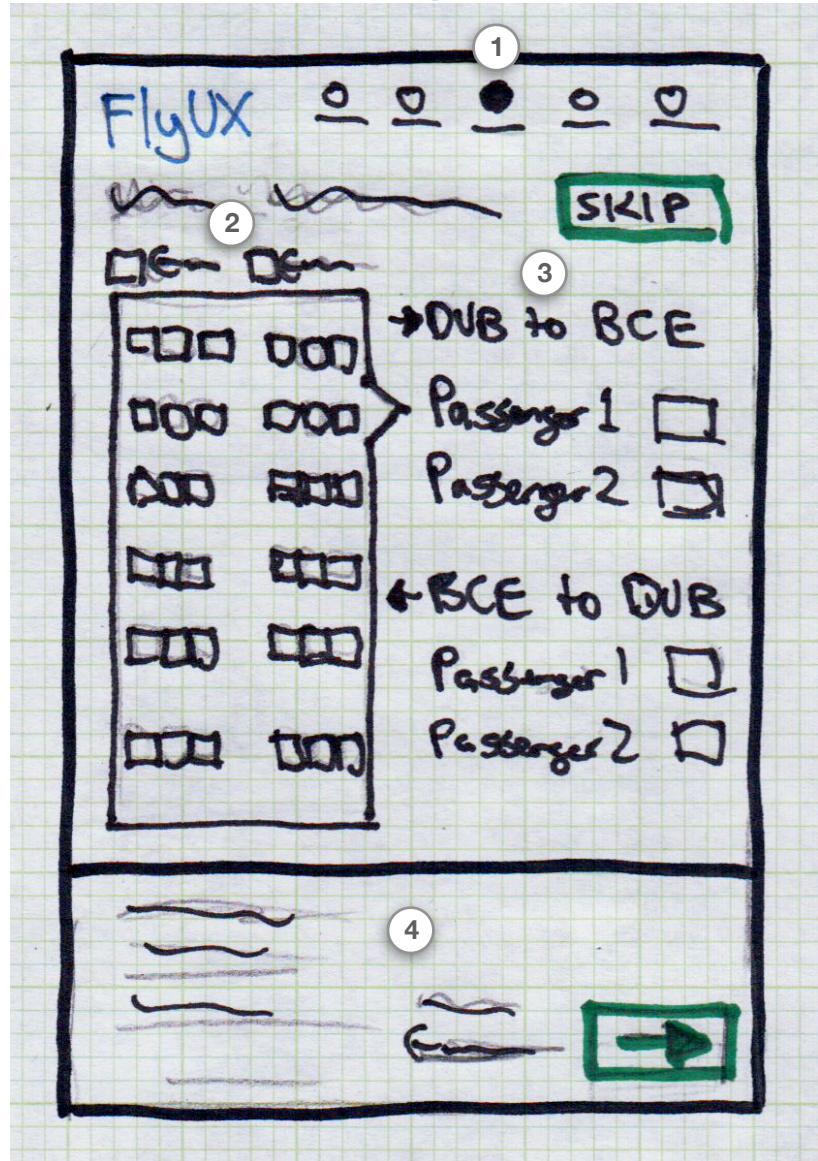
As the user makes selections, the price panel (at the bottom of the page) will update accordingly.

If the user has selected an upgrade package, the defaults will update accordingly. For example, if the user has selected the “Plus” upgrade package:

- The 25kg back will show “(1 included)” like it does for the cabin bag
- The input fields for the 25kg bag will both default to 1

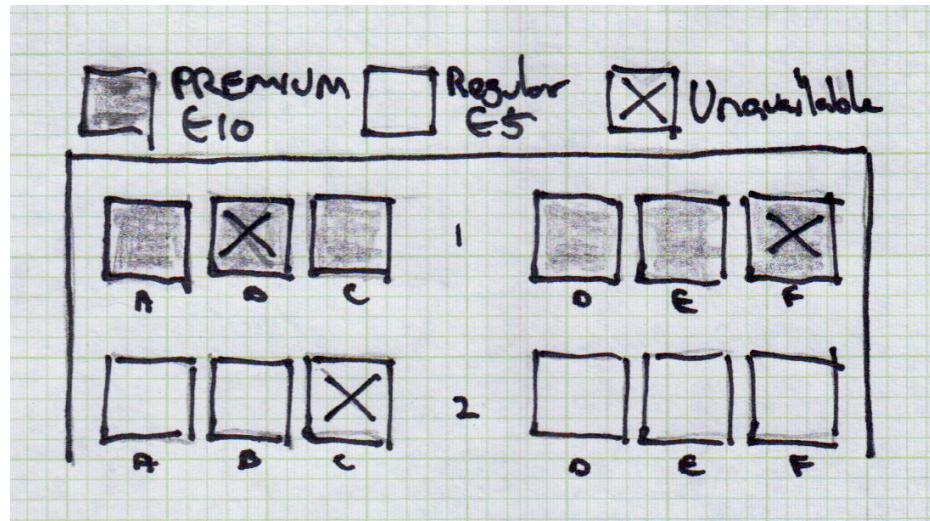
Assuming that there are upper limits on the number of bags you are able to take on board (regardless of cost), such limits will need to be enforced via client-side validation.

Select seats page



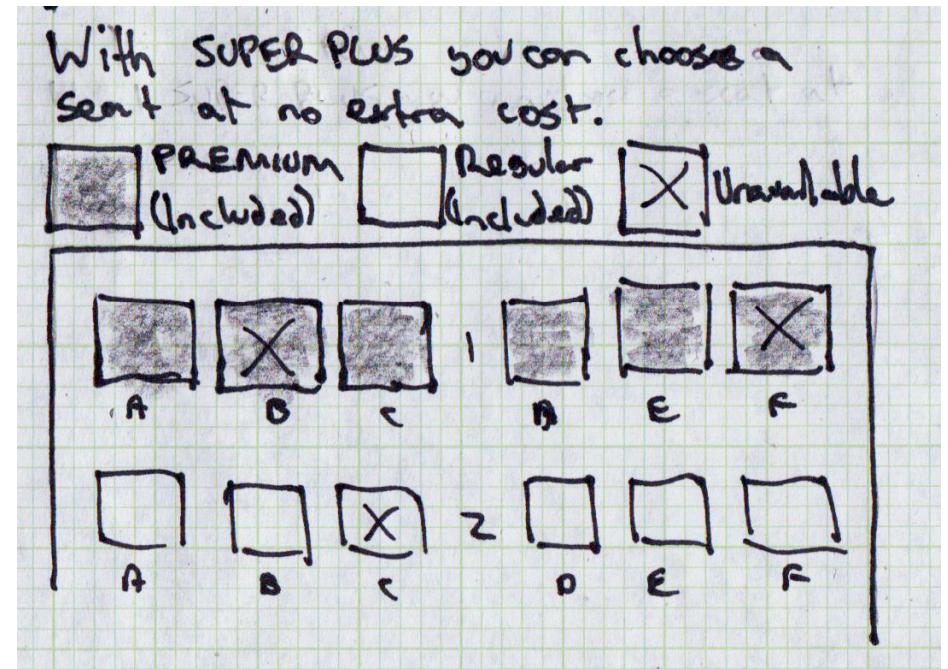
- 1 The progress stepper is updated to reflect that we are now in the third step of the booking process.
- 2 The [seat selection panel](#) allows available seats to be selected for each passenger on each leg of the journey.
- 3 The [passenger selection panel](#) shows which seats (if any) each passenger has selected.
- 4 The [price summary](#) will update as selections are made.

Seat selection panel

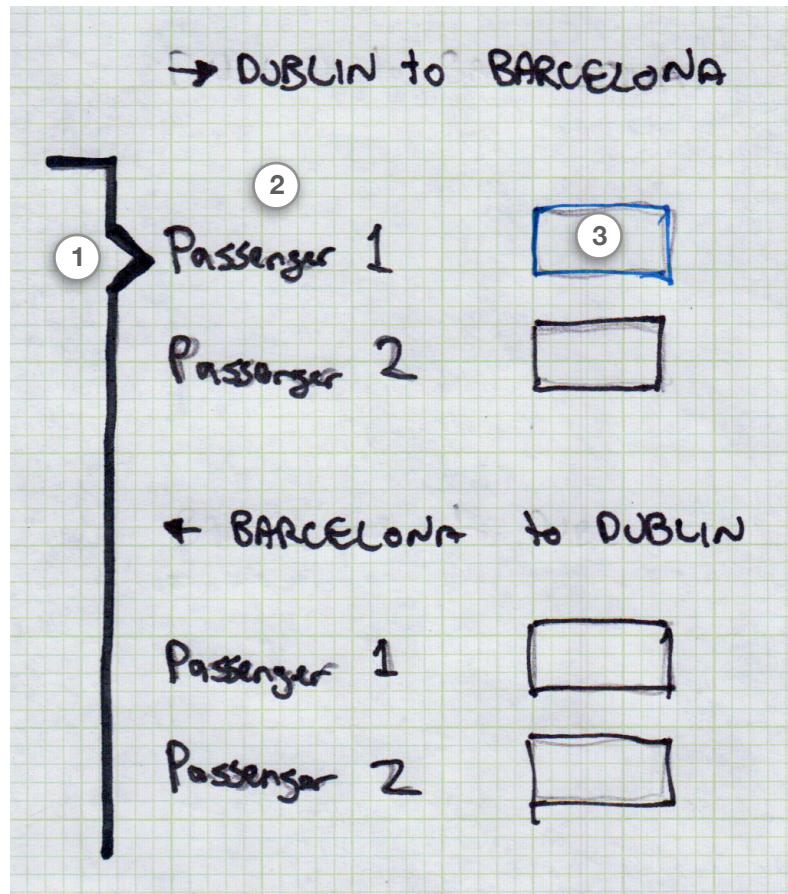


- Shading/colour is used to indicate the premium (more expensive) seating options.
- Unavailable (already selected) seats are marked with an X.
- The user is able to click on a seat to populate the corresponding field in the passenger selection panel.
- This panel will need to be scrollable.

If the user has selected an upgrade package, the key at the top of the seat selection panel should reflect that no additional cost will be incurred as they make a selection.



Passenger selection panel



- 1 The seat selection looks like it's attached or pointing to the currently selected passenger.
- 2 The passengers are clickable.
- 3 This looks a little bit like an input field and is in a highlighted state but it only receives a value when the user clicks on a seat in the [selection panel](#).

When a seat is selected:

- The input field receives the value of the seat that was selected AND it should be coloured according to the type of seat that was selected.
- The next passenger (or next leg of the journey) will move to the focussed state (as per below).
- As always, the price panel will also be updated.

