Airline booking application overview

1   Running the application

1.1   Starting the server

Pull the image from GHCR:

```
docker pull ghcr.io/jarrowsm/flightapp:latest
```

then run the server with

```
docker run -p PORT:8000 ghcr.io/jarrowsm/flightapp:latest
```

The `dbprime.py` script has populated the database with random synthetic data ahead of building the Docker image. Schedule is filled for 100 days from today and 400 customers with 500 bookings are inserted. These values may be changed with arguments to dbprime.py after flushing the database with

```
python manage.py flush
```

You may also choose to exclude basic or synthetic data. For example, if wanting to first test without synthetic data, use `--no-synth`, and then with synthetic data, `--no-basic` should be used to avoid errors. Figure 1.1 shows the help information for dbprime.py.

```
Priming database
usage: dbprime.py [-h] [--no-basic] [--no-synth] [--schedule-days SCHEDULE_DAYS]
                  [--customers CUSTOMERS] [--bookings BOOKINGS]

Populate the database

options:
  -h, --help            show this help message and exit
  --no-basic            Exclude basic data (Aircraft, Airport, Schedule)
  --no-synth            Exclude synthetic data (Customer, Booking)
  --schedule-days SCHEDULE_DAYS
                        Number of days ahead of today to insert into Schedule
  --customers CUSTOMERS
                        Number of synthetic rows to insert into Customer
  --bookings BOOKINGS   Number of synthetic rows to insert into Bookings
```

Figure 1.1: `dbprime.py` help information

2   Landing page

Figure 1.2 depicts the landing page. It includes a 'hero banner' which provides login capabilities (see Section 4) and a feature to search flights. A featured destinations section is also included.
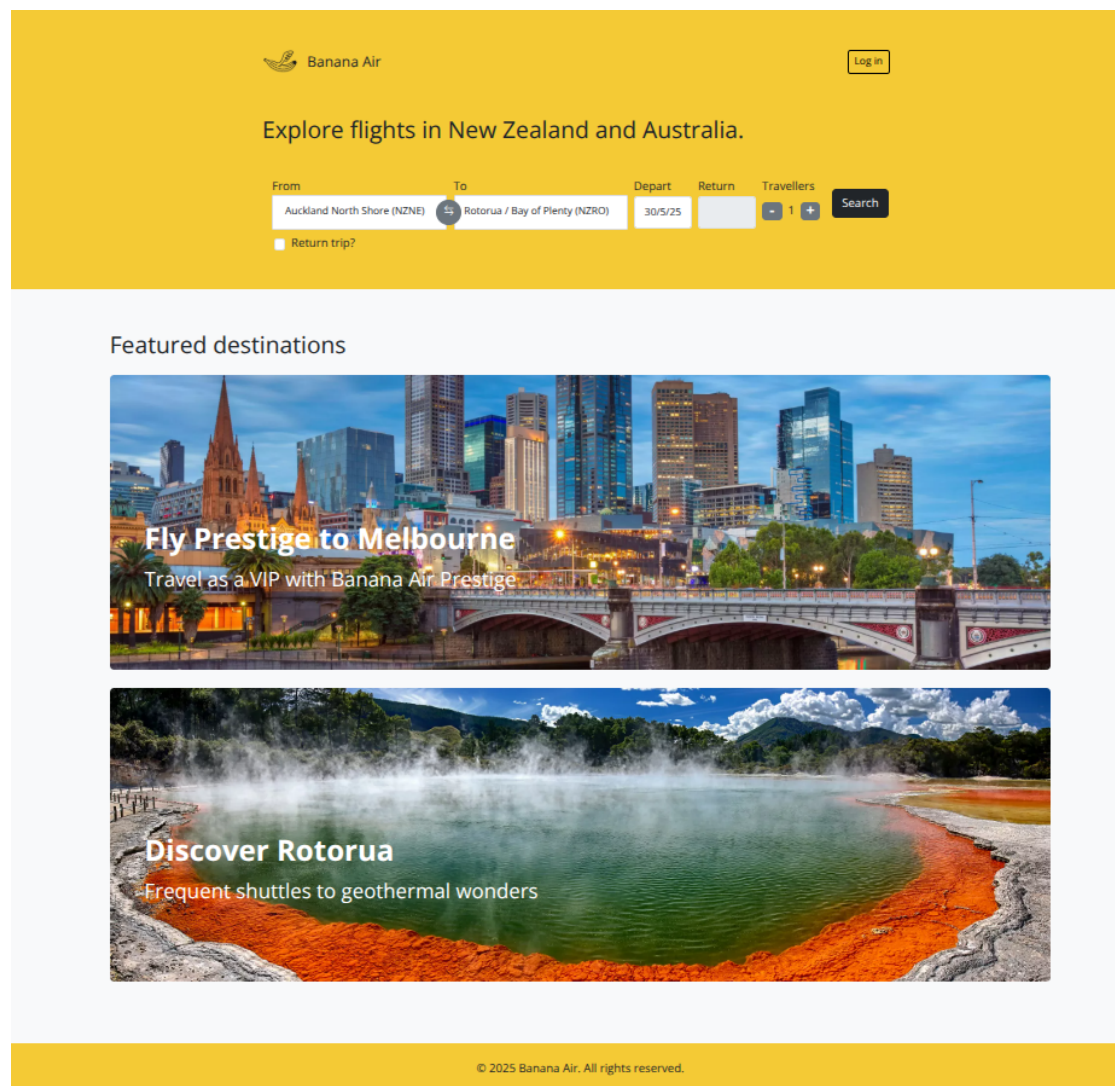
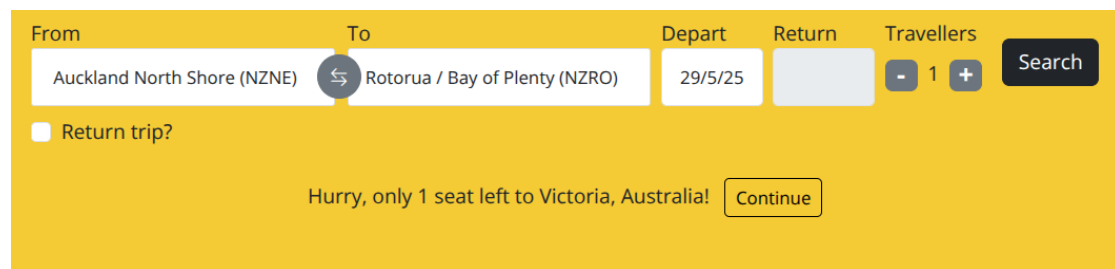Figure 1.2: Landing page

2.1 Search bar

Choices.js [2] provides searchable select inputs for the origin and destination as shown in Figure 1.3. Choices.js was chosen as it is lightweight, dependency-free and compatible with vanilla JS. Destinations are set dynamically according to origin via an AJAX request to the destinations/ URL. Between select inputs is a button to swap the origin and destination (refreshing the destinations and flight dates). Beneath these elements, a return checkbox toggles the return datetime input.

Figure 1.3: Searchable origin and destination select inputs

Another lightweight plugin, Flatpickr [4], was used for departure and return datetime inputs (Figure 1.4). Using dates attained from the flight_dates/ URL, grey marks are inserted to indicate dates with flights with seats remaining. The departure date is automatically set to the next available flight and, with the return textbox selected, the return date is set to the first available return flight after the departure. These dates are refreshed with changes to the origin and destination. Finally, the number of travellers—between one and six—may be chosen via two buttons which decrement and increment the value respectively.

Figure 1.4: Searchable origin and destination select inputs

Should the user select flights but leave at either the registration (Section 4) or confirmation (Section 5) steps, a call to action (CTA) displays the seats remaining and links to where the user left off, as shown in Figure 1.5.

Figure 1.5: Resume booking CTA

To complete the landing page, I include a featured destinations section with banners for Melbourne and Rotorua (see Figure 1.2). Clicking on these appropriately modifies the origin and destination and creates a fading shadow highlighting this (Figure 1.6).

Figure 1.6: Search bar after clicking the Melbourne banner

3 Searching for flights

Figure 1.7: Search page for a round-trip flight to Rotorua

Figure 1.7 depicts the search page for a round-trip flight to Rotorua. Starting with the date cards, each displays the lowest price out of flights with seats available (unless all flights are fully booked, in which case the lowest overall price is displayed). These link to the search with updated date parameters. Since the return comes after the outbound, the departure/return will also be updated if a departure after or return before is chosen. This is also validated at the server-side, but front-end logic also ensures any click entails a single GET request. Dates earlier than today (GMT) are greyed out and unclickable, while the price is struck on fully-booked dates. For dates with no flights, a simple message is displayed.

Beneath the dates, key information is provided for each flight: departure and arrival regions,

ICAO codes and times, flight duration, flight number, ticket price and seat availability. All times are in local times for the respective regions and a 'next day' tag is shown if the flight runs overnight. Also, a popover is shown on hovering over the (?) icon, as in Figure 1.8 for a Melbourne departure. Only flights after the current time are retrieved—the early outbound is absent from Figure 1.8 as it departed before the current time. Prices are dynamically calculated, increasing with fewer remaining seats and days until departure, with the base price restored three days in advance.
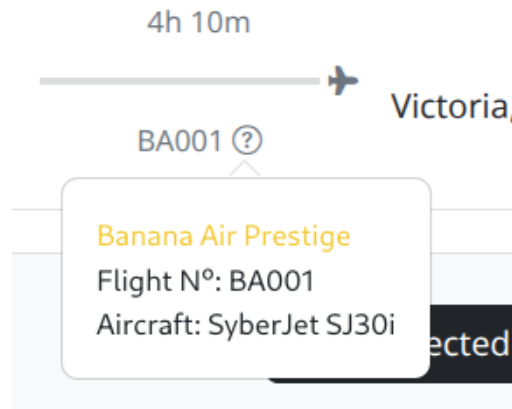
Figure 1.8: Popover for Melbourne departure

Selections are thoroughly validated at the server side with errors appearing as Bootstrap toasts. Figure 1.9 shows the error when selecting a full flight. Selecting this flight was not prevented on the front-end to make the error clearer to the user. Figure 1.10 shows the warning when a flight has too few seats for the number of travellers—the flight can still be booked with the available seats. Moreover, a return flight departing before the outbound flight arrives is not allowed (Figure 1.11). Additionally, the query parameters themselves are validated and errors are displayed with a link to the index page (Figure 1.12).
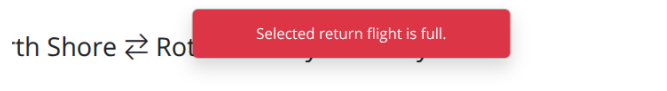
Figure 1.9: Flight full error
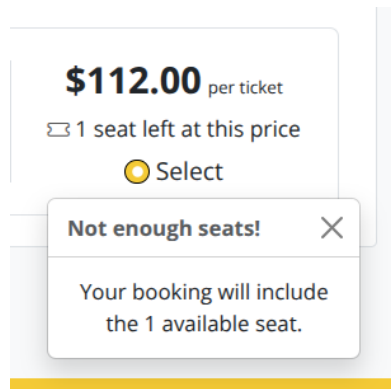
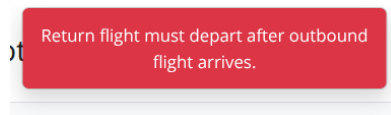Figure 1.10: Too few seats warning

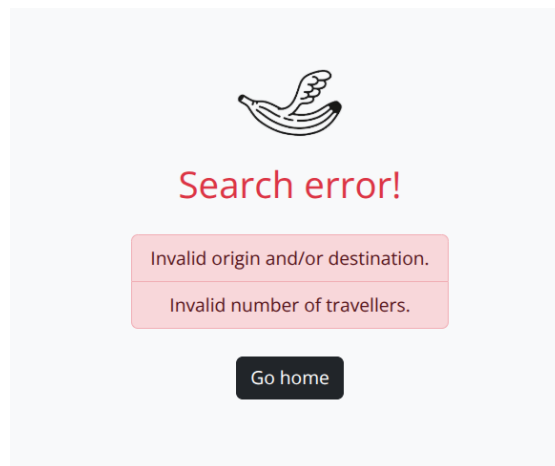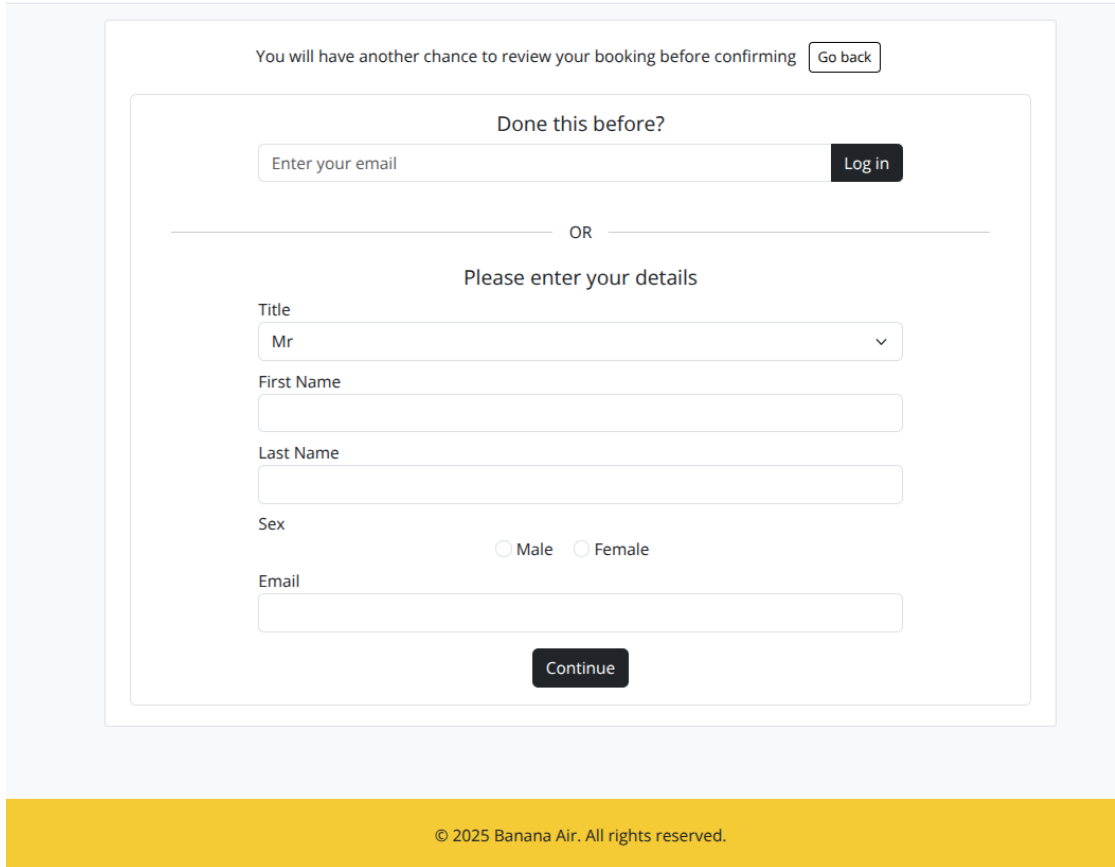Figure 1.11: Return flight departing before outbound arrives error

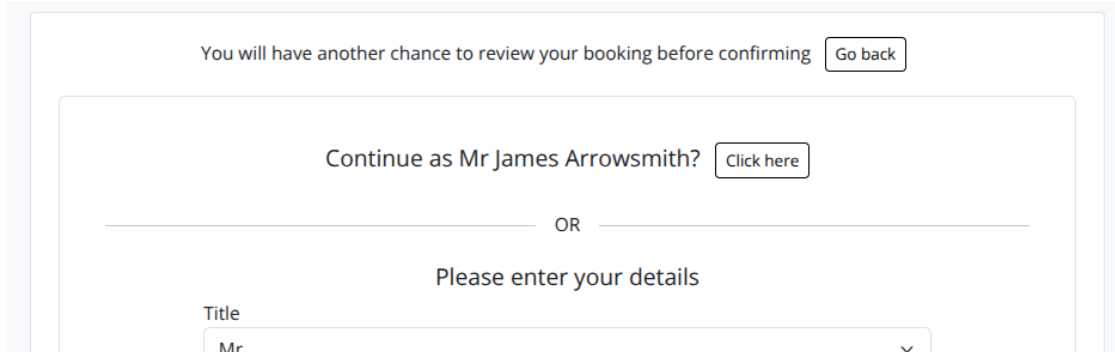Figure 1.12: Page shown after search query errors

4 Registration

Figure 1.13: Registration page when logged out

Figure 1.14: Registration page when logged in

After selecting flights, the user is asked to register their details. Alternatively, they may continue with an email used for a previous booking. Sessions track the booking process and prevent users from accessing this or the bookings page (Section 6) too early. Figure 1.14 shows the register page when logged in. The login system, while not secure, reduces the time to make multiple bookings. Login buttons appear on the index page and navbar of some other pages. These present a Bootstrap modal with an email or confirm form (Figure 1.15). JS is used to avoid an unnecessary refresh.
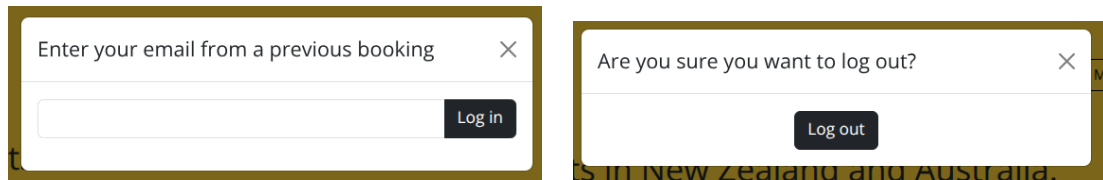
(a) Modal shown when logging in   (b) Modal shown when logging out

Figure 1.15: Login system modals

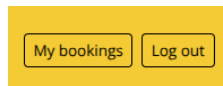When logged in, a button is displayed that links to the bookings page as shown in Figure 1.16.

Figure 1.16: Index page detail when logged in

5 Confirmation

Figure 1.17: Confirmation page

The confirmation page, shown in Figure 1.17, provides another opportunity to check the flights before booking. It highlights the flight information with the number of tickets and prices. The user may modify their customer details or change flights, which takes them back to their search. Clicking show route shows a relevant map from Great Circle Mapper [5].

6 Booking management

Figure 1.18: Bookings page

Figure 1.18 shows the page after a successful booking. The customer can make multiple bookings and these appear in descending order. Each booking can be cancelled, presenting a confirmation modal followed by a toast, removing the booking and adjusting the schedule. Route maps are also included. This page can be accessed while logged out and displays a message directing the user to log in. Navigating back and rebooking raises the error shown in Figure 1.19.

Figure 1.19: Booking error

7 Invoice

Figure 1.20: Invoice page

Figure 1.20 shows the index page for a return booking. Accessible only when logged in, it includes the customer and company information, booking reference, the date the booking was made (GMT), the schedule information, and the prices and overall balance, with an option to print this page. Appropriate errors are shown with a link to bookings for a missing or invalid reference.

Bibliography

[1] Bootstrap: Responsive front-end framework. `https://getbootstrap.com/`.

[2] Choices.js: A vanilla js customisable select box/text input plugin. `https://github.com/Choices-js/Choices`.

[3] Django. `https://www.djangoproject.com/`.

[4] Flatpickr: a lightweight and powerful datetime picker. `https://flatpickr.js.org`.

[5] Great circle mapper. `https://www.gcmap.com`.