bookDCU - CA236 3rd Year Project

Technical Specification

Nacé van Wyk 20467546

Scott Brady 20387871

04/02/2023

Supervisor: Dr Tracey Mehigan

Table of Contents

1. Introduction

1.1 Overview

BookDCU is a seat reservation system to be used by DCU students to reserve a seat in a Computing Lab guaranteeing students a seat and machine to complete projects. The system requires students to select a date and time. Once they have selected the date students are presented with the available seats, students then choose their preferred seat in their preferred lab. Students then follow steps to complete a booking, after the booking is complete students receive an email to the email address linked to the google account the student used to login. Students then have to verify the use of their booking at the time of their booking using a QR code. The QR code contains the details linked to the specific seat. The user uploads the image of the QR code to a QR code checker API to verify the booking. Once the booking is verified the user receives an email confirming the booking verification.

bookDCU makes use of the Imgur API as this is where the image of the QR code is uploaded to temporarily. From Imgur a link to the image is returned which is then passed to the goQR API which extracts the QR code contents from the image and returns this information to bookDCU.

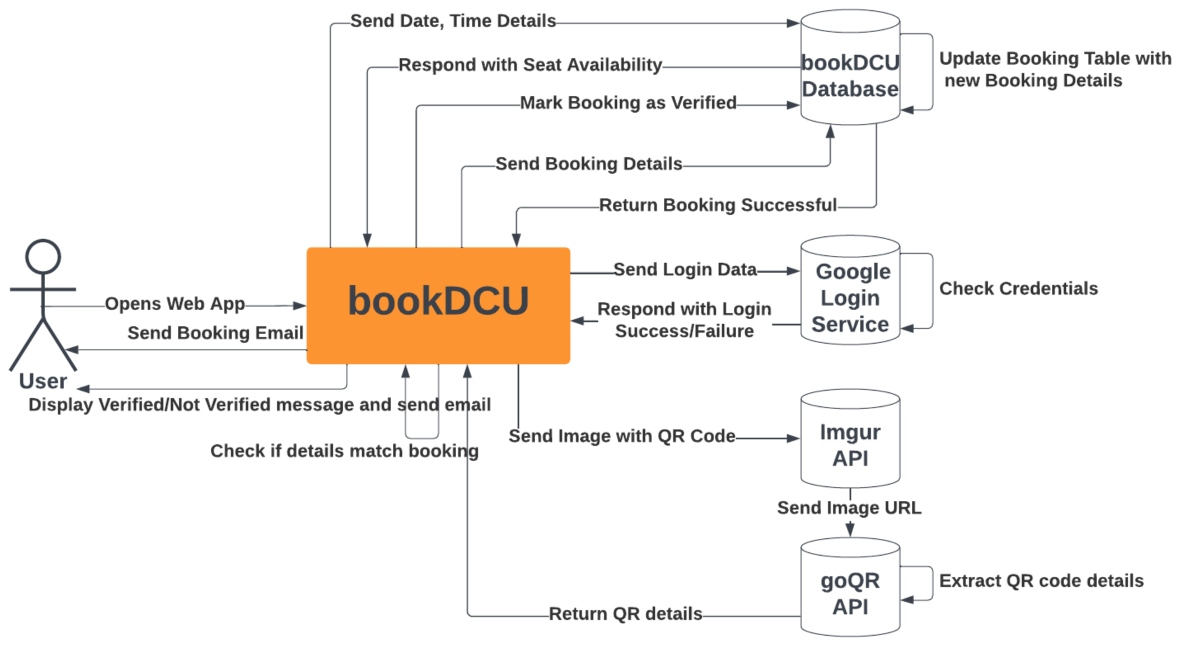
A Gmail account is used for sending out booking emails to users. This is directly connected to bookDCU through Django's emails feature.

1.2 Glossary

There are no explicitly complex technical terms or acronyms used that an average reader would not already know.

2. System Architecture

2.1 System Architecture Diagram



The system architecture is composed of five primary components: the bookDCU web application, the Google Login Service, the bookDCU database, the Imgur API and the goQR API. These components are interdependent, with two components continually interacting with each other.

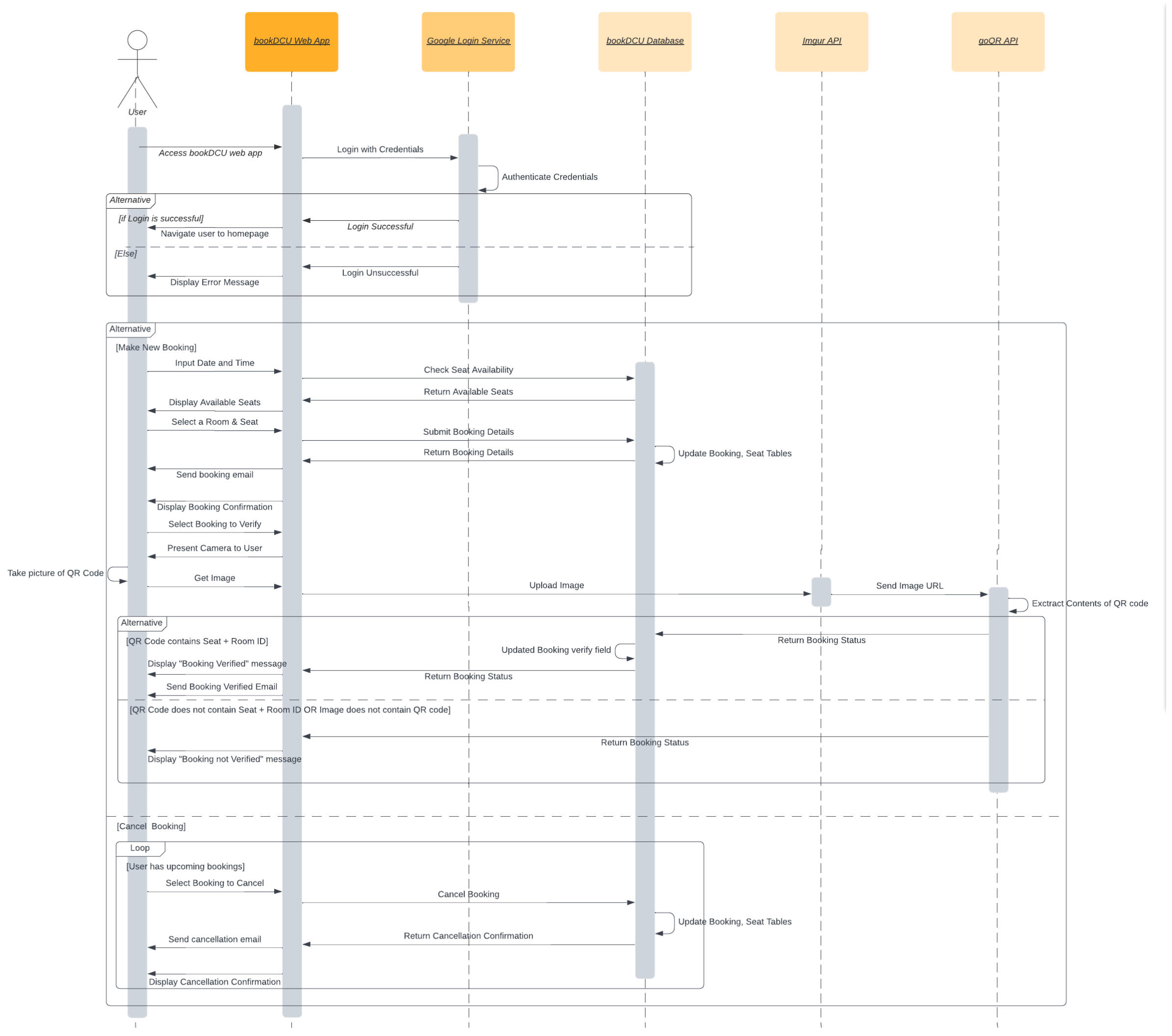
The user will open the bookDCU web application and request to login using their Google credentials, which will then send the user's login credentials to the Google Login Service. The Google Login Service will verify the details are correct and inform the bookDCU web application.

The user can then choose a date and time for their booking which the bookDCU web application will send to the bookDCU database. The bookDCU database which will check the availability for the specified date and time and return available seats to the user. When the user chooses a room and seat, the booking details are sent from the bookDCU web application to the bookDCU database and the bookings and seats tables are updated. Finally, the user is informed that their booking is successful via the bookDCU web application and an email sent to the user.

Following the user successfully making a booking they are required to verify the booking. The user selects a booking the verify and the bookDCU application opens their camera prompting them to upload a photo of the QR code linked to the seat that they have booked. This QR code is uploaded to the Imgur API which sends the image to the goQR API which extracts the QR code details and sends it back to bookDCU. If the details match the booking is verified message is displayed and the user receives an email confirming the verification. The booking's verified field in the bookDCU database is updated. Otherwise, a message displaying that the booking was not verified is delivered to the user.

3. High-Level Design

3.1 Sequence Diagram



In the above sequence diagram, the user logs into bookDCU and if successful is taken to the homepage. They can then make a new booking by inputting a date and time. The bookDCU system will then check the availability and show what seats are available to the user. The user can then choose a room and seat and submit the booking. Following this the bookings and seat tables in the bookDCU database will be updated to reflect the new booking. The user also then receives a booking confirmation email.

Once a booking has been made the user then verifies their booking. The user selects a booking to be verified. They are presented with a camera to take a picture of the QR code linked to their seat. This image is uploaded to Imgur and the link to the image on Imgur is passed to a QR Code checker API, which passes the information in the QR code to bookDCU. If the QR code is the correct QR code for that room and seat, the verification field is then updated and the booking status is the passed to bookDCU and a booking verified message is then displayed. The user then also receives an email confirming their booking verification.

Cancelling a booking works in a similar fashion, except we must first check to make sure the user has any upcoming bookings before we allow them to cancel. Once the user chooses to cancel a booking, the bookings table is updated and confirmation of the cancellation is shown to the user.

4. Problems and Resolution

OpenTimetable Integration

- During the development process we aimed to incorporate the OpenTimetable into the system to allow the user to only book a room if it is available, however we found that we were not exactly sure how to go about this. We contacted various members of staff to put us in touch with someone who can guide us regarding the OpenTimetable API, however it seemed impossible to find anyone able to help us. In addition, having looked ourselves at how OpenTimetable operates itself, it seemed quite complicated, with room IDs being long, illogical strings that did not bear any representation to the room they were supposed to be. In addition, we found that these room IDs frequently changed which would make it even more difficult to incorporate the service in bookDCU. For our final submission, we opted not to include OpenTimetable in bookDCU.

Google Login

- One of our first steps was implementing logging in using your Google account rather than having users creating an account, because this will ensure that we are not responsible for storing the user’s data. Implementing this proved to be easier than expected as Django had built in features for social account authentication. However it was not strictly smooth sailing since the initial fazes of the project is being completed by running a server on your machine and visiting your webapp at localhost. This has caused for a number of redirects when a user attempts to log in using their google account before login is deemed successful. After conducting research, we found reason the user is redirected 2-3 times before seeing the homepage was because on the Google backend, the redirect URL was set as localhost (127.0.0.1). If the URL was something different, like bookdcu.ie, this issue would not occur.

To solve this, we decided we would try to use GitHub pages to host bookDCU as this was the same place we were hosting our blog for this project from. However, we soon discovered that it is not possible to host a Django webapp on GitHub pages as it is designed for hosting static pages only.

Some attempts were made to deploy bookDCU using PythonAnywhere, however this proved more difficult than we expected and given the short time remaining, we decided to abandon this idea and spend the time we had left perfecting what we already had.

Dynamic seat maps

- We initially set up the seat map of each room manually using hard-coded HTML. However, given the wide array of rooms and their different sizes, we knew that it would be preferable from an administrator point-of-view to generate the seat maps automatically.

  Our solution requires the administrator to enter the number of rows in a room and the number of seats in each row. The map is then generated automatically.

  To do this we first tried using nested for loops in Python, but soon realised this wouldn’t work. We switched to using JavaScript with one for loop creating each row, and the inner for loop populating the rows with the appropriate amount of seats.

Booking cancellation

- An issue was encountered wherein when a booking was cancelled, the seat associated with that booking was still marked as occupied on the seat map, meaning another user would not be able to select it. This was obviously not ideal, so a fix had to be found. In this case, we initially tried to get the seat number from the booking and remove it from the rooms’s booked_seats field. However, this did not work and it turned out that we needed to first get the seat *object* that was associated with both that seat number and that room. Following this, we could remove the seat object from the room's booked_seats field, and then save the room and delete the seat object itself.

- Several issues were encountered in relation to cancelling bookings.
    - One issue that we encountered was when a user cancelled their booking, the seat associated with that booking was still marked as occupied on the seat map, meaning another user would not be able to select it. This was obviously not ideal, so a fix had to be found. In this case, we initially tried to get the seat number from the booking and remove it from the rooms's booked_seats field. However, this did not work and it turned out that we needed to first get the seat *object* that was associated with both that seat number and that room. Following this, we could remove the seat object from the room's booked_seats field, and then save the room and delete the seat object itself.
    - Another issue that occurred was that if I had two bookings on different dates for the same seat and cancelled the first booking, I would not be able to cancel the second. This was because we were actually deleting the entire seat object that was associated with the first booking, and because this seat object was also associated with the second booking, it could not be found when we tried to cancel the second booking.

      To fix this, we had to avoid deleting the seat, and instead remove the dates of the booking we were cancelling from that seat's booked_dates field in the database.

Dates

- Dates were a source of constant issues throughout the project, mainly due to them coming in a wide variety of formats and locales.
- One major issue that we encountered caused by dates was the inability to make bookings for the months of March, April, May, June, July and September. After much testing and debugging, we discovered that the cause of this issue was due to how the month names were being shortened in our JavaScript. Some months were behaving differently due to the lengths of their names, for example March was being shortened to "Marc" instead of "Mar". Carefully formatting the month names correctly meant that when the date was passed to our booking function, this issue disappeared.

Changing Rooms on the Seatmap

- We ran into an issue that is specific to the Safari browser in relation to changing the room from one to another on the seatmap page. In other browsers, changing the room simply causes the page to reload and the seatmap appears for the new room. However, in Safari, the user is returned to the homepage and must enter their date and times again. When they do so, the next time they are taken to the seatmap page, it will display the new room they selected previously.

  We recognise that this does not provide the optimal user experience, and carefully researched ways to mitigate it. We disocvered that the reason this issue occurs is due to how Safari handles `window.location.reload()` JavaScript requests and POST data. It is not possible to reload a page using JavsScript in this way while still carrying over data received by the page from a POST request. We tried a few other ways, such as `window.location.href = window.location.href`, but this made no difference.

  In the end, we were unfortunately unable to rectify this Safari-specific issue.

5. Installation guide

Prerequisites

- Up-to-date browser and internet connection
- Text editor (e.g. VScode) and a terminal
- DCU credentials and Microsoft authenticators
- Mobile phone with a camera

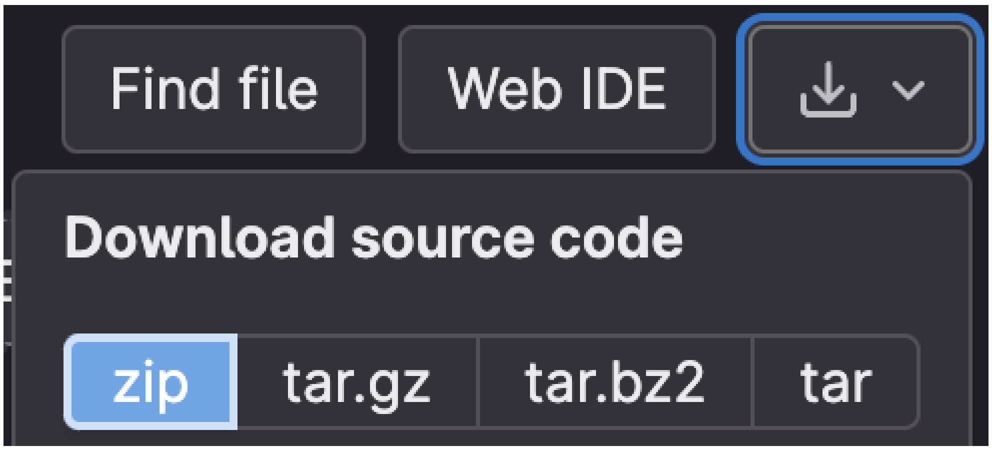
Installing bookDCU on Your Computer

The following are required to run bookDCU:

- Python v3.8 or later
  python.org/downloads
- pip (for installing Django)
  sudo apt install python3-pip
- Django (latest version available)
  pip3 install django

- Django Allauth (latest version available)
  pip3 install django-allauth

1. Install Python 3 according to the installation instructions for your system on https://python.org/downloads
2. Install pip with the command `sudo apt install python3-pip`.
3. Install Django with the command `pip3 install django`.
4. Install django-allauth with the command `pip3 install django-allauth`.
5. Download bookDCU from https://gitlab.computing.dcu.ie/bradys76/2023-ca326-bradys76-vanwykn2. Click on the download button as shown and choose 'zip'.



6. Extract the downloaded zip file to your desired location.
7. Navigate to the `src/backend/` directory using the terminal.
8. To start bookDCU, run `python3 manage.py runserver 0.0.0.0:8000`.
9. It is recommended to run the following commands to ensure bookDCU performs optimally:
   a. `python3 manage.py makemigrations bookDCU`
   b. `python3 manage.py migrate`
   c. `python3 manage.py seed`

bookDCU is now installed and running. You can access it by visiting localhost (127.0.0.1) in your browser, or from a mobile device on the same network as the host machine by appending the port 8000 to that machine's IP address (e.g. if the machine's IP is 192.168.0.6, you would need to enter 192.168.0.6:8000 in your browser). Note that you will also need to add your machine's IP in the `ALLOWED_HOSTS` list in `settings.py`, as shown below:

```
ALLOWED_HOSTS = ["192.168.0.88", "127.0.0.1", "172.20.10.9", "192.168.0.6"]
```

- bookDCU is available at the following url https://vanwykn2.pythonanywhere.com/
  - Functionality is limited at this url so running locally is recommended.