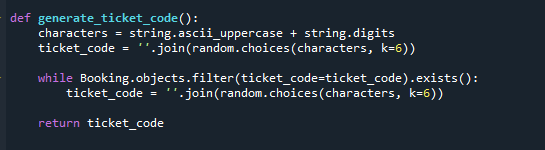
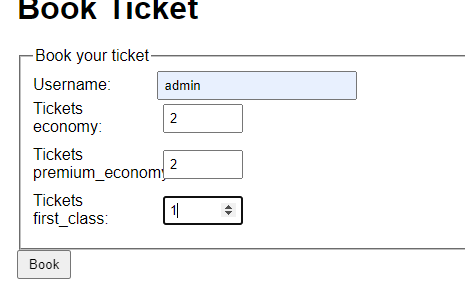
Test Plan for Flight Booking App:

1. Unit Tests:
   * Test 1: Test Ticket Code Generation
     + Description: Ensure that the ticket code is generated correctly.



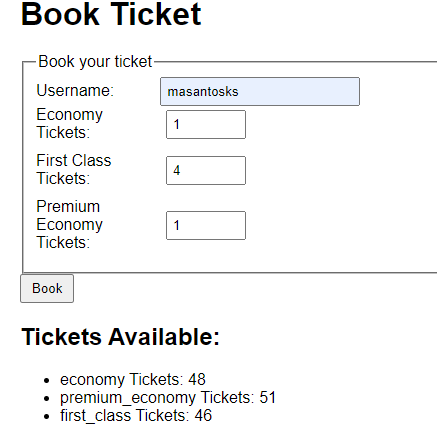
* + - Expected Outcome: The generated ticket code should be a string of length 6 consisting of alphanumeric characters.



* + - Success Criteria: The generated ticket code matches the expected format.



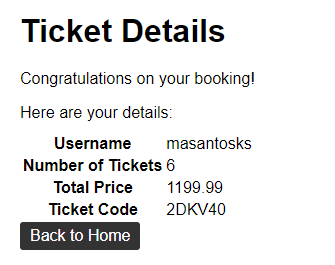
* + Test 2: Test Booking Creation
    - Description: Create a booking and check if all fields are populated correctly.



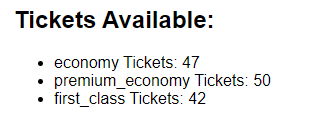
* + - Expected Outcome: The booking instance should have all the provided data accurately assigned to the respective fields.

Need be 6 tickets, and the total price need to be 1199.99.

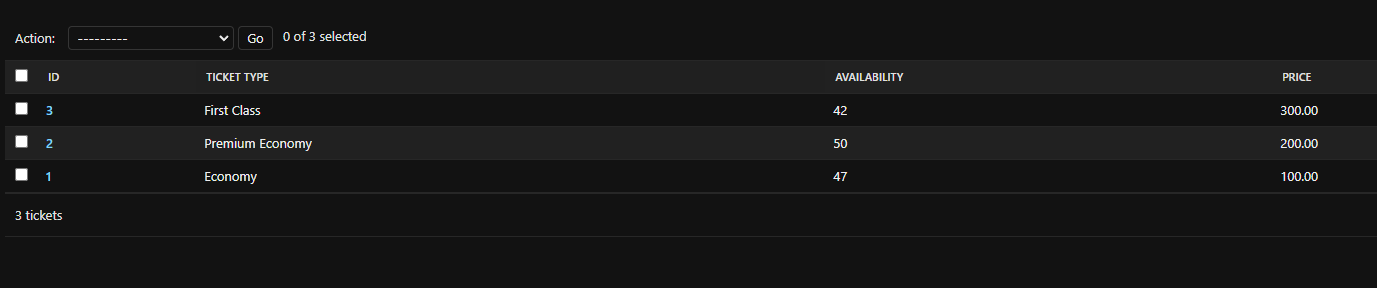
* + - Success Criteria: All fields in the booking instance match the provided data.



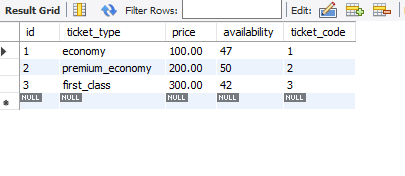
1. Integration Tests:
   * Test 3: Test Ticket Availability Check
     + Description: Check the availability of tickets for a specific ticket type.



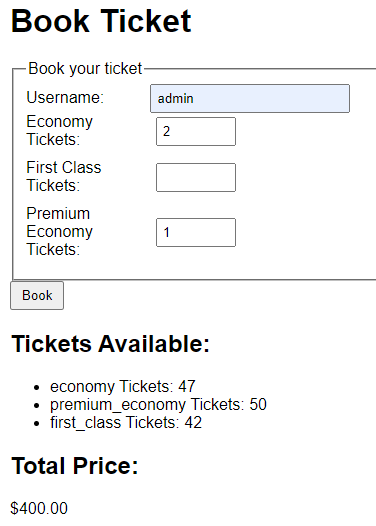
* + - Expected Outcome: The available ticket count for the specified ticket type should be accurate.



* + - Success Criteria: The available ticket count matches the expected value.



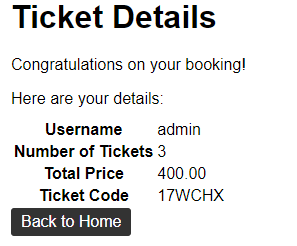
* + Test 4: Test Booking Calculation
    - Description: Create a booking and calculate the total price based on the ticket quantities.



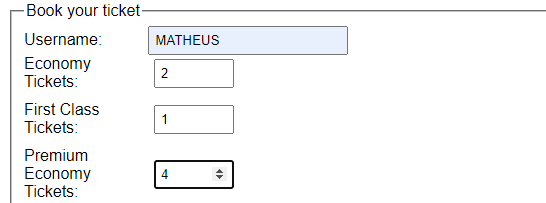
* + - Expected Outcome: The total price should be calculated correctly based on the ticket quantities and their respective prices.



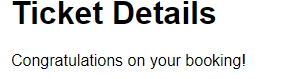
* + - Success Criteria: The calculated total price matches the expected value.



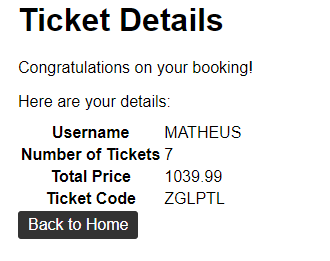
1. User Acceptance Tests:
   * Test 5: Test User Booking Process
     + Description: Simulate the user booking process.



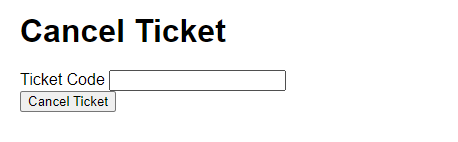
* + - Expected Outcome: The user should be able to successfully book tickets and receive a booking confirmation.



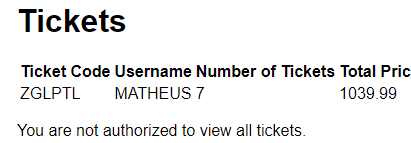
* + - Success Criteria: The user receives a valid booking confirmation with a unique ticket code.



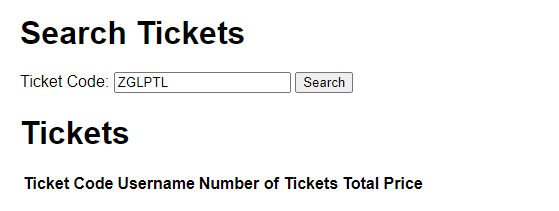
* + Test 6: Test User Ticket Cancellation
    - Description: Simulate the user ticket cancellation process.



* + - Expected Outcome: The user should be able to successfully cancel a booked ticket.



* + - Success Criteria: The canceled ticket is removed from the booking and the ticket becomes available again.



1. Code and user’s errors:
   * + Error 1:

File "C:\Users\Matheus\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11\_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\django\db\models\fields\related\_descriptors.py", line 266, in \_\_set\_\_

raise ValueError(

ValueError: Cannot assign "<User: admin>": "Booking.user" must be a "User" instance.

The error message encountered suggests that I’m trying to assign a value of "<User: admin>" to the "Booking.user" field, but it expects a "User" instance instead.

A picture containing text, screenshot, font

Description automatically generated

The way found to fix this error was changing user = User.objects.get(username='admin') to username=request.POST['username']. In this way, the database is saved using the POST method and can be consulted, changed or deleted at any time.

* + - Error 2:

TypeError: unsupported operand type(s) for \*: 'decimal.Decimal' and 'float'

2023-06-21 08:43:38,876 - ERROR - Error booking ticket for user admin

The error message encountered suggests that I’m trying to perform a multiplication operation between a decimal.Decimal object and a float value, which is not supported in Python.



A screen shot of a computer code

Description automatically generated with low confidence

The way I found was by importing the lib decimal import Decimal and the round\_down, I changed the function to operate in a decimal way and using the round\_down that thus allows me to use a multiplication operation between the decimal.Decimal object and a float value that is not supported in python.

* 5 Using Debugtool:

A screenshot of a computer program

Description automatically generated with medium confidence

5.1 The copy of debug text:  
Use debug tools.

* Attached Document Flight Booking System Test Plan

5.2 Develop a test plan that determines expected results from test data.

* Attached Document Flight Booking System Test Plan

5.3 Follow the test plan in 5.2 to carry out a test.

* Attached Document Flight Booking System Test Plan

5.4 Compare actual results against expected results and make recommendations and changes as appropriate.

* Attached Document Flight Booking System Test Plan

Test Environment and Dependencies:

* Programming Language: Python
* Framework: Django
* Testing Framework: Django Testing Framework
* Database: MySQL
* Required Dependencies: Django, MySQL connector