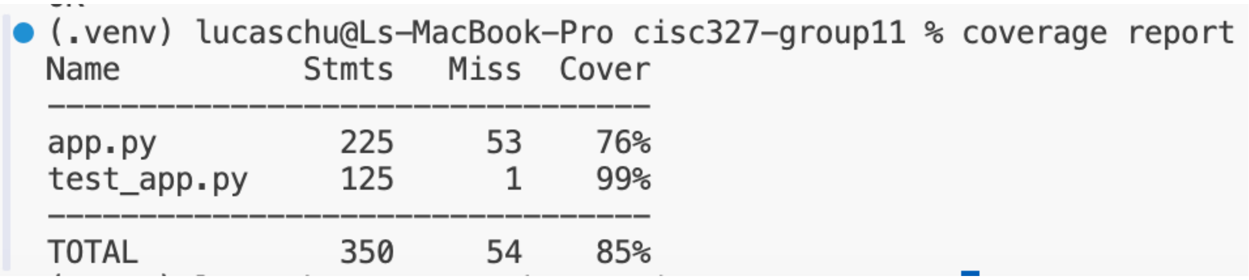
Original code from a3:



```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % coverage report
Name           Stmts   Miss  Cover
----------------------------------
app.py           225     53    76%
test_app.py      125      1    99%
----------------------------------
TOTAL            350     54    85%
```

1. **Flight Model __repr__ (Lines 89-91):**
   Missed Lines: return f'<Flight {self.flight_number}>' on line 91.

   Reason: This method is likely missed because it's not directly called in the tests, which would typically only invoke it when logging or printing Flight instances.

2. **Booking Model __repr__ (Lines 112-114):**
   Missed Lines: return f'<Booking {self.id} - Flight {self.flight_id} Seat {self.seats}>'on line 114.

   Reason: Similar to the Flight model, this method might not be tested directly, as it's generally used for debugging or logging.

3. **Login Route (Lines 133-143):**
   Missed Lines: flash('Invalid email or password', 'error') and subsequent render_template('login.html') on line 143.

   Reason: This line is executed when login credentials are invalid. If tests only check successful logins, this path might be missed.

4. **Signup Route (Lines 147-176):**
   Missed Lines: return render_template('signup.html') at line 176.

   Reason: This line is only reached for GET requests or if a POST fails certain conditions. If tests focus on successful POST requests, this line might not be executed.

5. **Forgot Password Route (Lines 177-191):**
   Missed Lines: return redirect(url_for('reset_password', email=email)) at line 188.

Reason: This redirect is for users whose email is found. If the tests don't include cases where the email exists, this line could be missed.

6. **Reset Password Route (Lines 192-216):**
   Missed Lines: return render_template('reset_password.html', email=email) at line 216.

   Reason: This line is used for GET requests or POST requests lacking a password. It could be missed if tests only check fully completed POST requests.

7. **Search Flights Route (Lines 232-241):**
   Missed Lines: flash('Please provide all required information', 'error') and subsequent redirect(url_for('book_flight')) on lines 240-241.

   Reason: These lines execute if any search criteria are missing. If tests always supply complete search data, this case might be skipped.

8. **Payment Method Route (Lines 255-292):**
   Missed Lines: flash('Payment failed. Please try again.', 'error') and subsequent redirect(url_for('book_flight')) on lines 289-292.

   Reason: This part handles payment failures. If tests assume successful payments only, this path might remain uncovered.

9. **Select Seat Route (Lines 293-336):**
   Missed Lines: The exception handling block starting with except Exception as e: on lines 330-335.

   Reason: This block is for unexpected issues like database errors. If tests don't simulate such errors, it won't be reached.

10. **Booking History Route (Lines 338-346):**
    Missed Lines: return redirect(url_for('login')) at line 342.

    Reason: This line is reached if there's no active session. If tests only run when a session is active, this case could be missed.

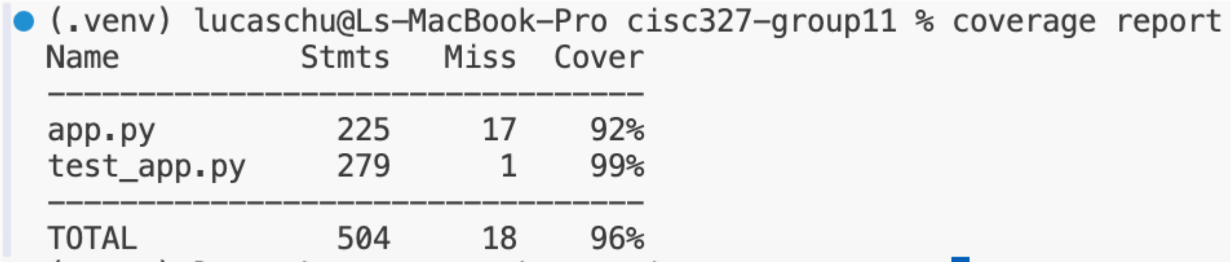
11. **Cancel Booking Route (Lines 349-371):**
    Missed Lines: flash('Unauthorized action', 'error') and redirect(url_for('booking_history')) at line 356.

    Reason: This part handles unauthorized booking cancellation attempts. If tests don't simulate unauthorized actions, it might be missed.

12. **Init DB and Main Block (Lines 410-414):**
    Missed Lines: The init_db() and app.run(debug=True) lines within the if __name__ == '__main__': block.

    Reason: These lines are typically not executed in a test environment, as tests import the module instead of running it directly.

New updated code:



```
● (.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % coverage report
  Name            Stmts   Miss  Cover
  -----------------------------------
  app.py            225     17    92%
  test_app.py       279      1    99%
  -----------------------------------
  TOTAL             504     18    96%
```

1. **Signup Route (Lines 147-176):**
   Missed Lines: return render_template('signup.html') at line 176.

   Reason: This line is only executed if the request method is GET, or if a POST request fails to pass through other conditions. If the tests only simulate successful POST requests (such as valid signups) or cases where errors are caught early (like missing fields or duplicate emails), this line may not have been reached.

2. **Forgot Password Route (Lines 177-191)**:
   Missed Lines: return redirect(url_for('reset_password', email=email)) at line 188.

   Reason: This line directs users to reset their password if their email exists. If the test only includes cases where the user is not found or doesn't test this specific redirect path, it could be missed.

3. **Reset Password Route (Lines 192-216)**:
   Missed Lines: render_template('reset_password.html', email=email) at line 216.

   Reason: This line is executed on a GET request or if the POST fails conditions like missing a password. Without a specific test for this GET case or the edge case with missing passwords, this line may remain uncovered.

4. **Payment Method Route (Lines 255-292)**:
   Missed Lines: return redirect(url_for('book_flight')) at line 292.

   Reason: The else block and redirect occur when payment_success is False. If the tests assume payment is always successful, this path might not be tested, leaving the failure case uncovered.

5. **Select Seat Route (Lines 293-336)**:
   Missed Lines: The exception handling block starting with except Exception as e: at line 330.

   Reason: This block handles unexpected exceptions. If tests don't simulate database errors or unexpected issues, this exception handling won't be covered.

6. **Booking History Route (Lines 338-346)**:
   Missed Lines: return redirect(url_for('login')) at line 342.

   Reason: This redirect occurs if there's no active session. Without testing access to this route without logging in, this condition remains uncovered.

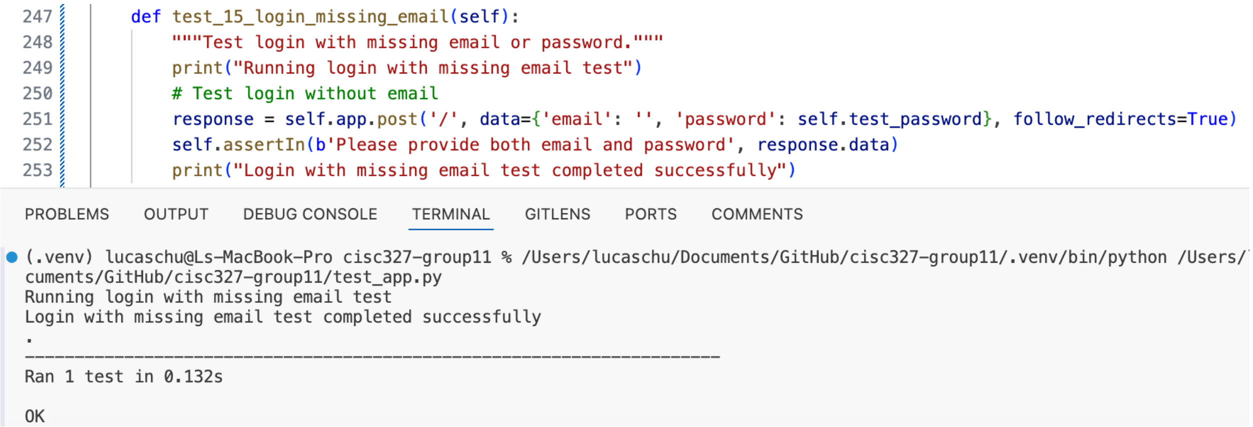
7. **Main Program Execution (Lines 410-412)**:
   Missed Lines: if __name__ == '__main__': block.

   Reason: This block initializes and runs the server. Test frameworks typically import the module rather than run it directly, so this part of the code isn't executed during testing.

Newly test scripts:

1. Login missing email

   Verify if a user cannot login without providing an email

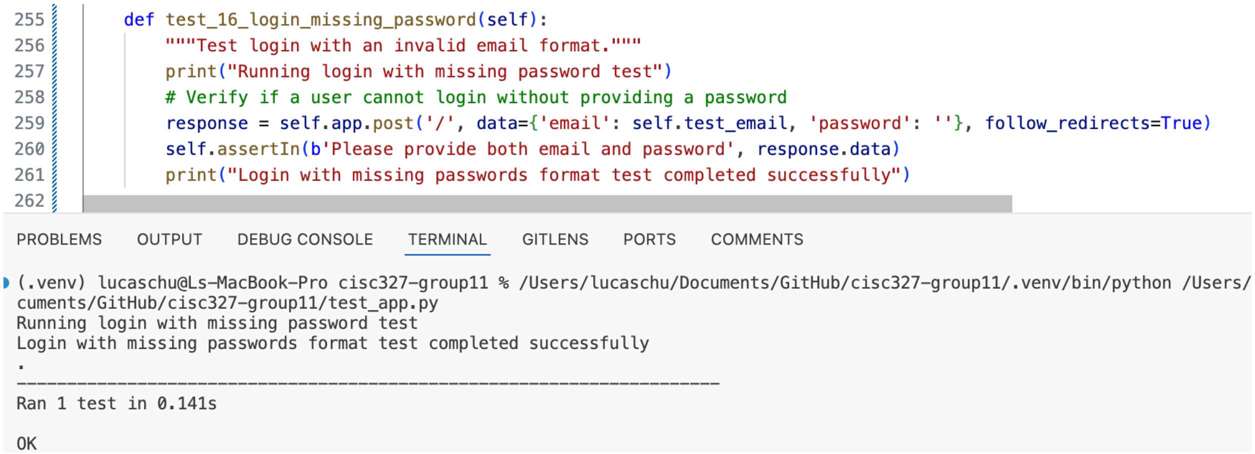


```
    def test_15_login_missing_email(self):
        """Test login with missing email or password."""
        print("Running login with missing email test")
        # Test login without email
        response = self.app.post('/', data={'email': '', 'password': self.test_password}, follow_redirects=True)
        self.assertIn(b'Please provide both email and password', response.data)
        print("Login with missing email test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running login with missing email test
Login with missing email test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.132s

OK
```

2. Login missing password

   Verify if a user cannot login without providing a password

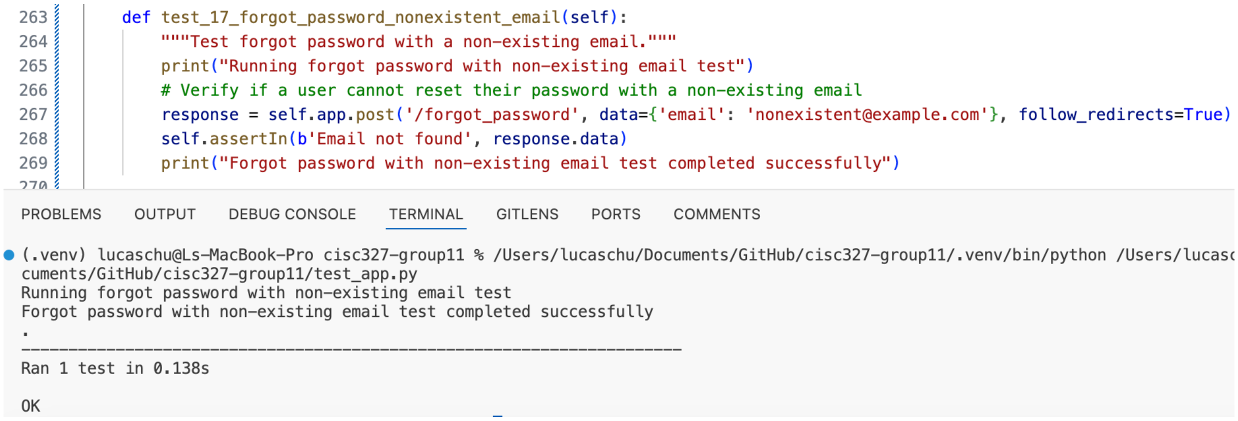


```
    def test_16_login_missing_password(self):
        """Test login with an invalid email format."""
        print("Running login with missing password test")
        # Verify if a user cannot login without providing a password
        response = self.app.post('/', data={'email': self.test_email, 'password': ''}, follow_redirects=True)
        self.assertIn(b'Please provide both email and password', response.data)
        print("Login with missing passwords format test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running login with missing password test
Login with missing passwords format test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.141s

OK
```

3. Forgot password non-existent email

   Verify if a user cannot reset their password with a non-existing email

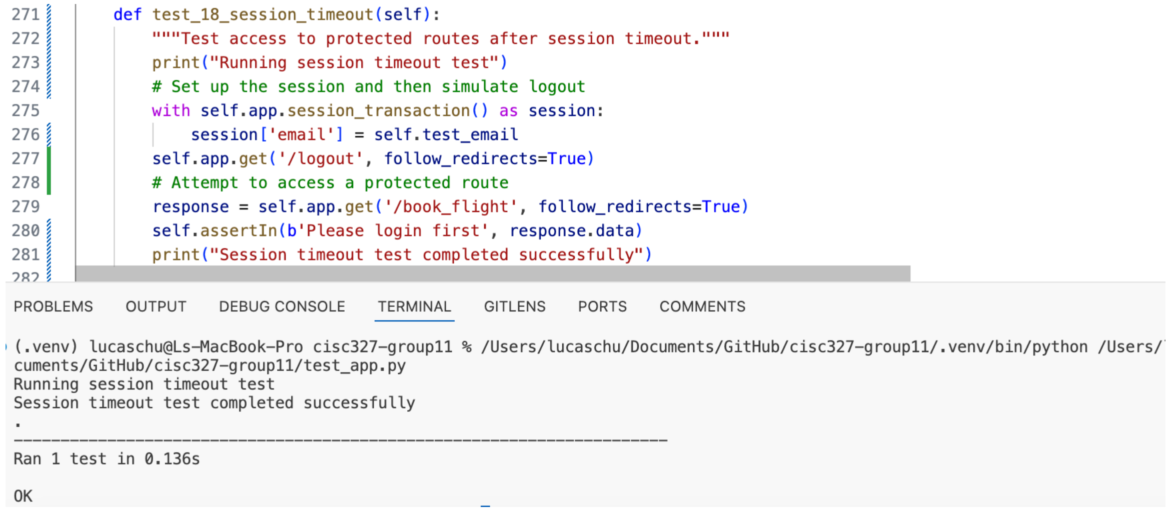


```
    def test_17_forgot_password_nonexistent_email(self):
        """Test forgot password with a non-existing email."""
        print("Running forgot password with non-existing email test")
        # Verify if a user cannot reset their password with a non-existing email
        response = self.app.post('/forgot_password', data={'email': 'nonexistent@example.com'}, follow_redirects=True)
        self.assertIn(b'Email not found', response.data)
        print("Forgot password with non-existing email test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/lucasc
cuments/GitHub/cisc327-group11/test_app.py
Running forgot password with non-existing email test
Forgot password with non-existing email test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.138s

OK
```

4. Session timeout

Verify that a user cannot access protected routes after session timeout

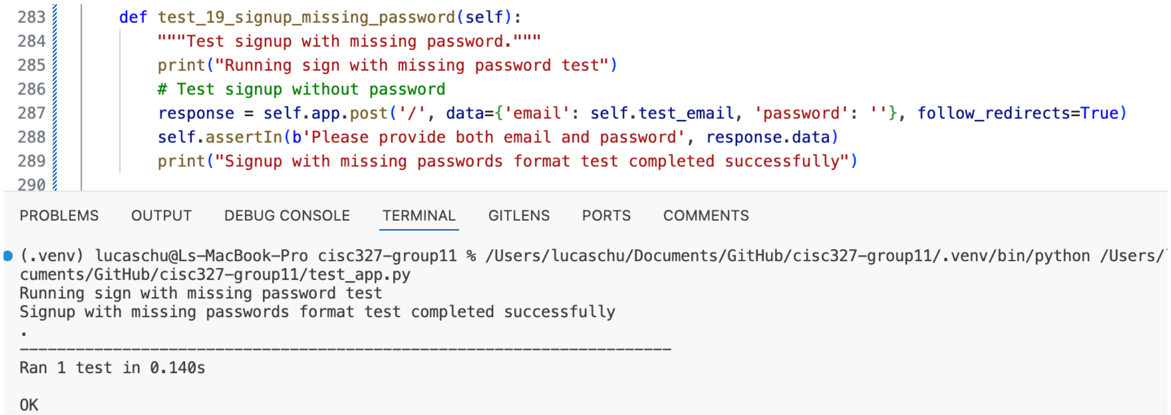


```
def test_18_session_timeout(self):
    """Test access to protected routes after session timeout."""
    print("Running session timeout test")
    # Set up the session and then simulate logout
    with self.app.session_transaction() as session:
        session['email'] = self.test_email
    self.app.get('/logout', follow_redirects=True)
    # Attempt to access a protected route
    response = self.app.get('/book_flight', follow_redirects=True)
    self.assertIn(b'Please login first', response.data)
    print("Session timeout test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running session timeout test
Session timeout test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.136s

OK
```

5. Signup missing password

Verify if a user cannot sign up without providing a password

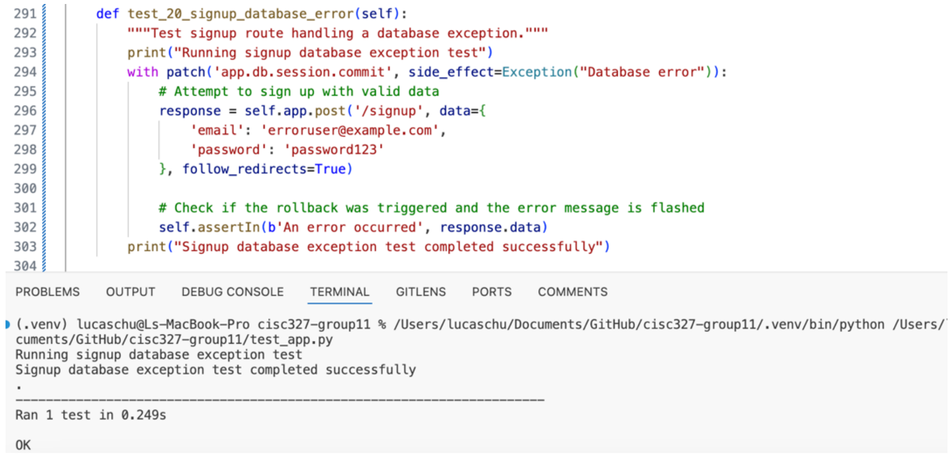


```
def test_19_signup_missing_password(self):
    """Test signup with missing password."""
    print("Running sign with missing password test")
    # Test signup without password
    response = self.app.post('/', data={'email': self.test_email, 'password': ''}, follow_redirects=True)
    self.assertIn(b'Please provide both email and password', response.data)
    print("Signup with missing passwords format test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running sign with missing password test
Signup with missing passwords format test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.140s

OK
```

6. Signup database error

Verify that a database error during signup triggers a rollback

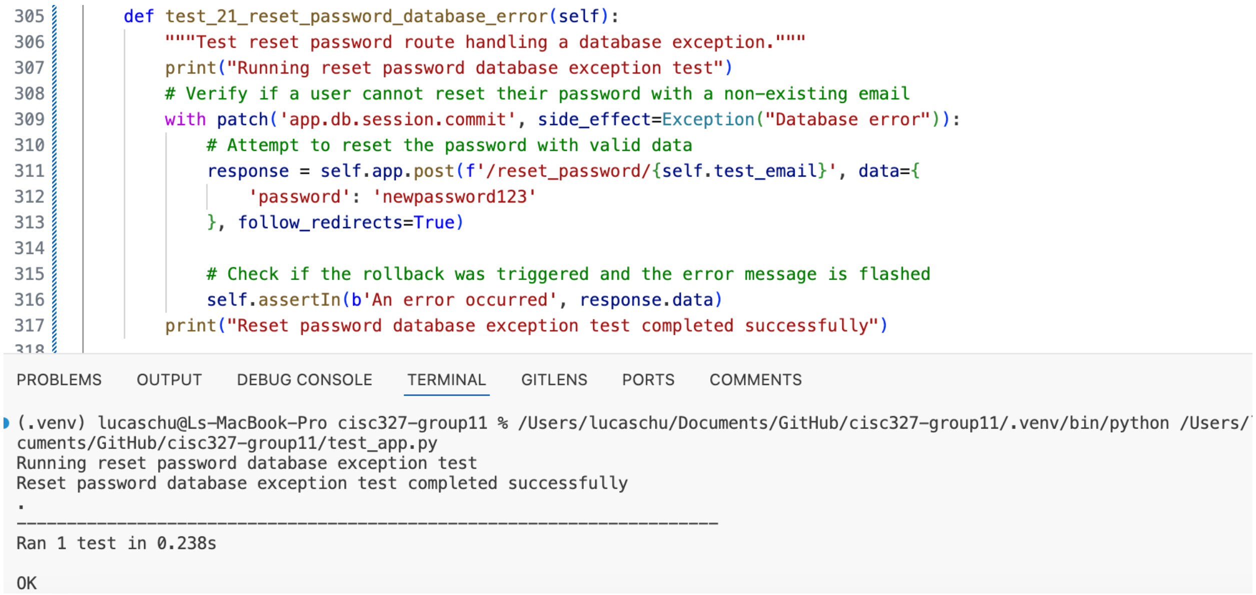


```
def test_20_signup_database_error(self):
    """Test signup route handling a database exception."""
    print("Running signup database exception test")
    with patch('app.db.session.commit', side_effect=Exception("Database error")):
        # Attempt to sign up with valid data
        response = self.app.post('/signup', data={
            'email': 'erroruser@example.com',
            'password': 'password123'
        }, follow_redirects=True)

        # Check if the rollback was triggered and the error message is flashed
        self.assertIn(b'An error occurred', response.data)
    print("Signup database exception test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running signup database exception test
Signup database exception test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.249s

OK
```

7. Reset password database error

Verify if a user cannot reset their password with a non-existing email

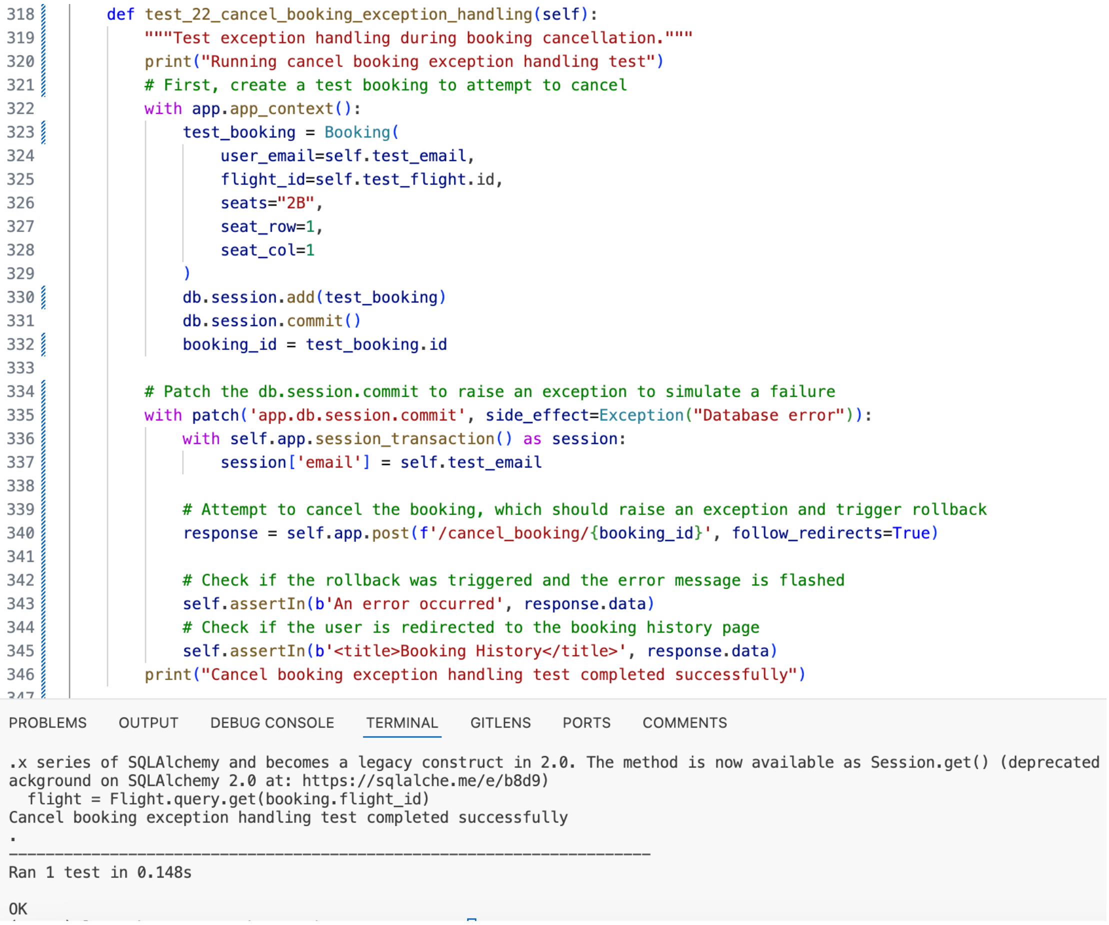


```
    def test_21_reset_password_database_error(self):
        """Test reset password route handling a database exception."""
        print("Running reset password database exception test")
        # Verify if a user cannot reset their password with a non-existing email
        with patch('app.db.session.commit', side_effect=Exception("Database error")):
            # Attempt to reset the password with valid data
            response = self.app.post(f'/reset_password/{self.test_email}', data={
                'password': 'newpassword123'
            }, follow_redirects=True)

            # Check if the rollback was triggered and the error message is flashed
            self.assertIn(b'An error occurred', response.data)
        print("Reset password database exception test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running reset password database exception test
Reset password database exception test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.238s

OK
```

8. Cancel booking exception handling

Verify that the app can handle a database exception during booking cancellation

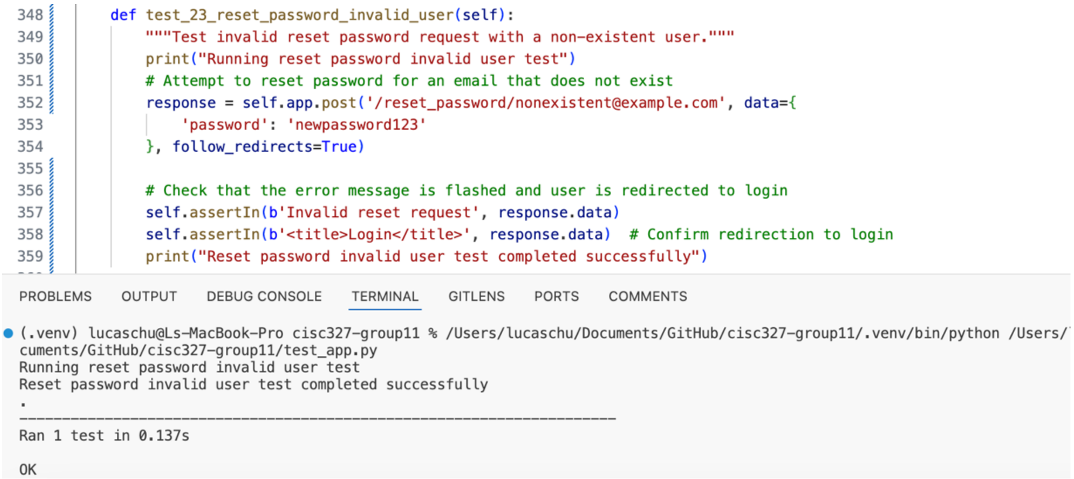


```
    def test_22_cancel_booking_exception_handling(self):
        """Test exception handling during booking cancellation."""
        print("Running cancel booking exception handling test")
        # First, create a test booking to attempt to cancel
        with app.app_context():
            test_booking = Booking(
                user_email=self.test_email,
                flight_id=self.test_flight.id,
                seats="2B",
                seat_row=1,
                seat_col=1
            )
            db.session.add(test_booking)
            db.session.commit()
            booking_id = test_booking.id

        # Patch the db.session.commit to raise an exception to simulate a failure
        with patch('app.db.session.commit', side_effect=Exception("Database error")):
            with self.app.session_transaction() as session:
                session['email'] = self.test_email

            # Attempt to cancel the booking, which should raise an exception and trigger rollback
            response = self.app.post(f'/cancel_booking/{booking_id}', follow_redirects=True)

            # Check if the rollback was triggered and the error message is flashed
            self.assertIn(b'An error occurred', response.data)
            # Check if the user is redirected to the booking history page
            self.assertIn(b'<title>Booking History</title>', response.data)
        print("Cancel booking exception handling test completed successfully")
```

```
.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated
ackground on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  flight = Flight.query.get(booking.flight_id)
Cancel booking exception handling test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.148s

OK
```

9. Reset password invalid user

Verify if a user cannot reset their password with a non-existing email

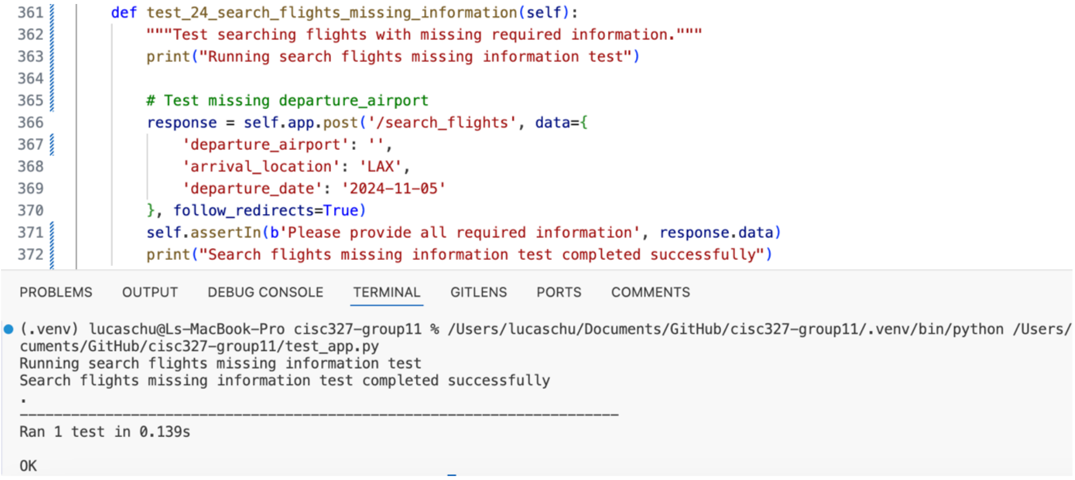


```python
    def test_23_reset_password_invalid_user(self):
        """Test invalid reset password request with a non-existent user."""
        print("Running reset password invalid user test")
        # Attempt to reset password for an email that does not exist
        response = self.app.post('/reset_password/nonexistent@example.com', data={
            'password': 'newpassword123'
        }, follow_redirects=True)

        # Check that the error message is flashed and user is redirected to login
        self.assertIn(b'Invalid reset request', response.data)
        self.assertIn(b'<title>Login</title>', response.data)  # Confirm redirection to login
        print("Reset password invalid user test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running reset password invalid user test
Reset password invalid user test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.137s

OK
```

10. Search flights missing information

Verify that the user is prompted to provide all required information

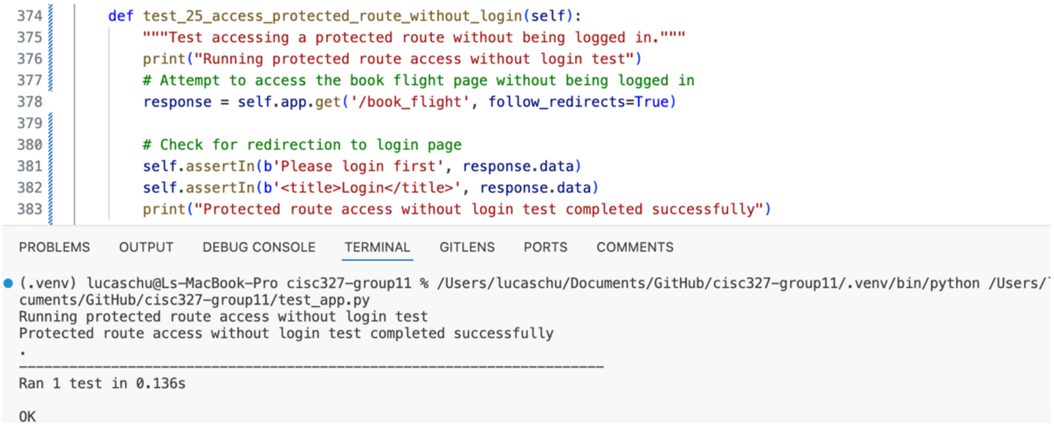


```python
    def test_24_search_flights_missing_information(self):
        """Test searching flights with missing required information."""
        print("Running search flights missing information test")

        # Test missing departure_airport
        response = self.app.post('/search_flights', data={
            'departure_airport': '',
            'arrival_location': 'LAX',
            'departure_date': '2024-11-05'
        }, follow_redirects=True)
        self.assertIn(b'Please provide all required information', response.data)
        print("Search flights missing information test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running search flights missing information test
Search flights missing information test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.139s

OK
```

11. Access protected route without login

Verify if a user cannot access a protected route without being logged in

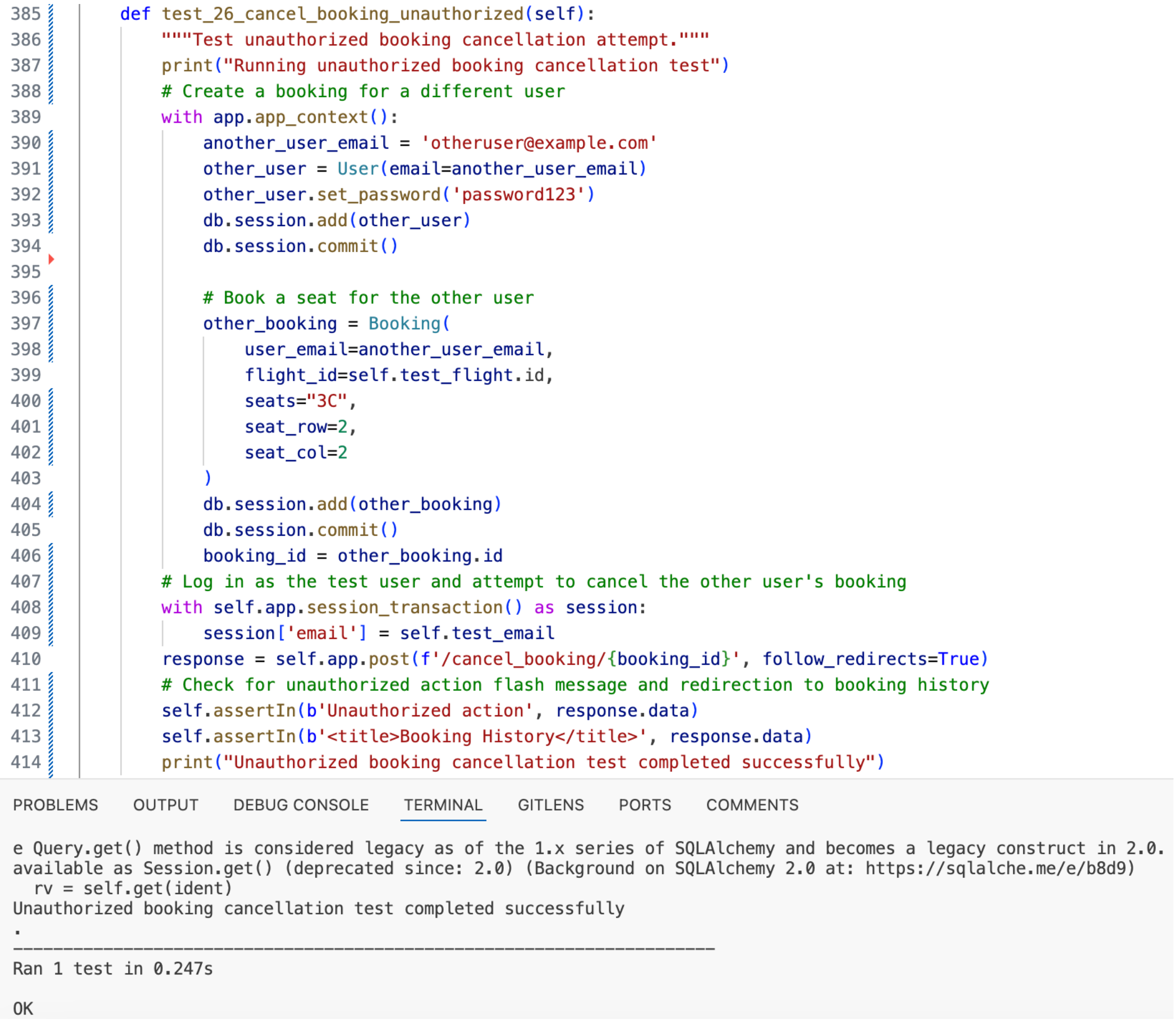


```python
    def test_25_access_protected_route_without_login(self):
        """Test accessing a protected route without being logged in."""
        print("Running protected route access without login test")
        # Attempt to access the book flight page without being logged in
        response = self.app.get('/book_flight', follow_redirects=True)

        # Check for redirection to login page
        self.assertIn(b'Please login first', response.data)
        self.assertIn(b'<title>Login</title>', response.data)
        print("Protected route access without login test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running protected route access without login test
Protected route access without login test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.136s

OK
```

12. Cancel booking unauthorized

Verify that a user cannot cancel another user's booking

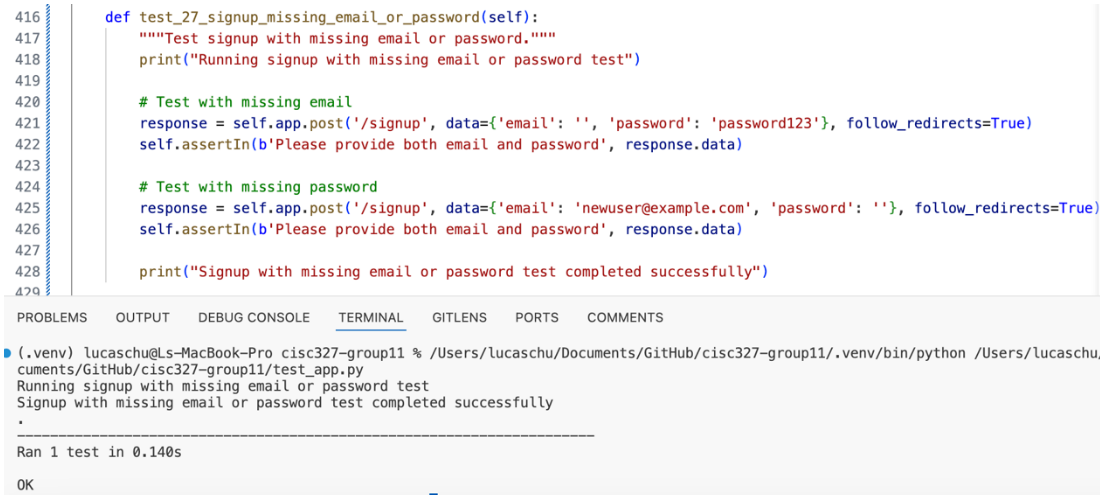


```python
    def test_26_cancel_booking_unauthorized(self):
        """Test unauthorized booking cancellation attempt."""
        print("Running unauthorized booking cancellation test")
        # Create a booking for a different user
        with app.app_context():
            another_user_email = 'otheruser@example.com'
            other_user = User(email=another_user_email)
            other_user.set_password('password123')
            db.session.add(other_user)
            db.session.commit()

            # Book a seat for the other user
            other_booking = Booking(
                user_email=another_user_email,
                flight_id=self.test_flight.id,
                seats="3C",
                seat_row=2,
                seat_col=2
            )
            db.session.add(other_booking)
            db.session.commit()
            booking_id = other_booking.id
        # Log in as the test user and attempt to cancel the other user's booking
        with self.app.session_transaction() as session:
            session['email'] = self.test_email
        response = self.app.post(f'/cancel_booking/{booking_id}', follow_redirects=True)
        # Check for unauthorized action flash message and redirection to booking history
        self.assertIn(b'Unauthorized action', response.data)
        self.assertIn(b'<title>Booking History</title>', response.data)
        print("Unauthorized booking cancellation test completed successfully")
```

```
e Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0.
available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  rv = self.get(ident)
Unauthorized booking cancellation test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.247s

OK
```

13. Signup missing email or password

Verify if a user cannot sign up without providing an email or password

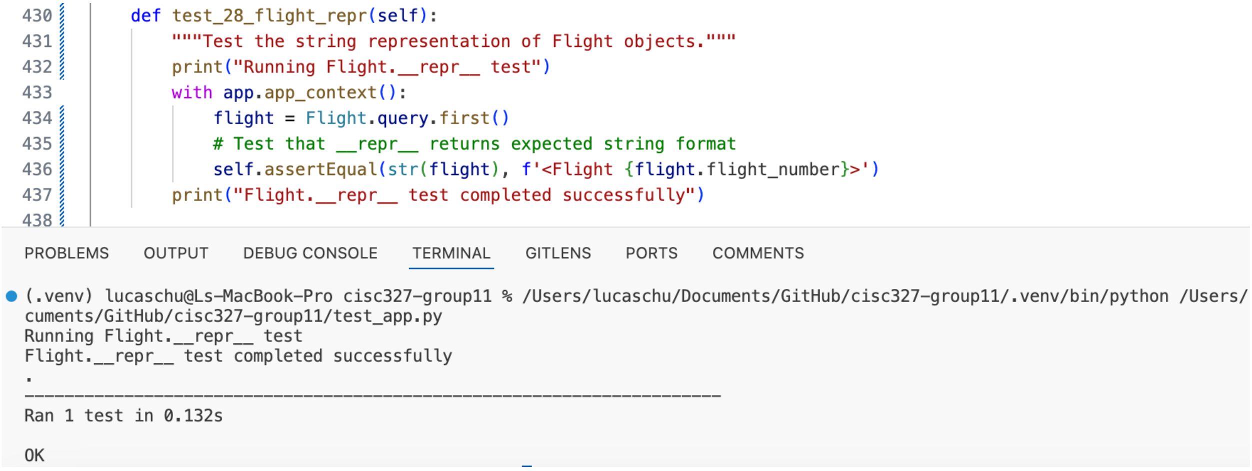


```python
    def test_27_signup_missing_email_or_password(self):
        """Test signup with missing email or password."""
        print("Running signup with missing email or password test")

        # Test with missing email
        response = self.app.post('/signup', data={'email': '', 'password': 'password123'}, follow_redirects=True)
        self.assertIn(b'Please provide both email and password', response.data)

        # Test with missing password
        response = self.app.post('/signup', data={'email': 'newuser@example.com', 'password': ''}, follow_redirects=True)
        self.assertIn(b'Please provide both email and password', response.data)

        print("Signup with missing email or password test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/lucaschu,
cuments/GitHub/cisc327-group11/test_app.py
Running signup with missing email or password test
Signup with missing email or password test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.140s

OK
```

14. Flight representation

Verify that a flight object can be created

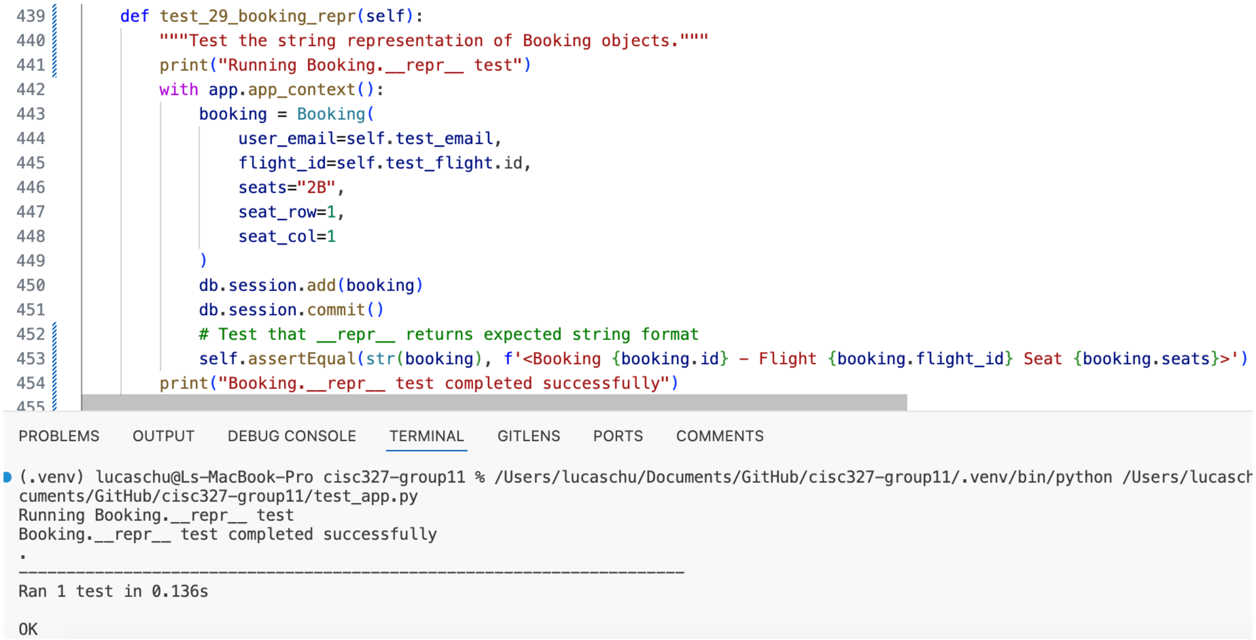


```python
    def test_28_flight_repr(self):
        """Test the string representation of Flight objects."""
        print("Running Flight.__repr__ test")
        with app.app_context():
            flight = Flight.query.first()
            # Test that __repr__ returns expected string format
            self.assertEqual(str(flight), f'<Flight {flight.flight_number}>')
        print("Flight.__repr__ test completed successfully")

```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running Flight.__repr__ test
Flight.__repr__ test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.132s

OK
```

15. Booking representation

Verify that the __repr__ method returns the expected string format

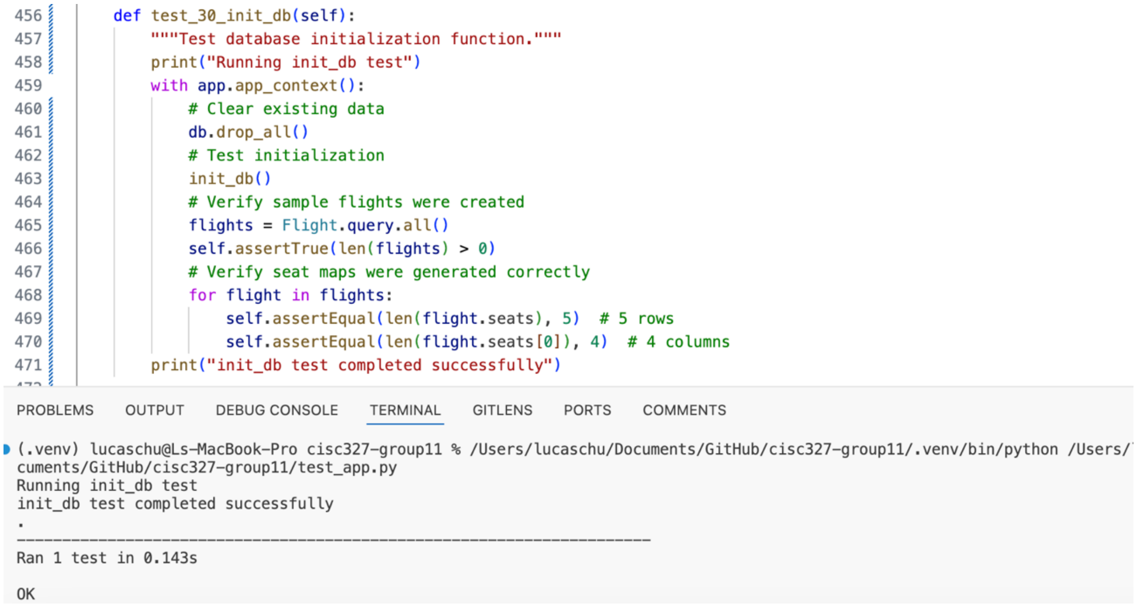


```python
    def test_29_booking_repr(self):
        """Test the string representation of Booking objects."""
        print("Running Booking.__repr__ test")
        with app.app_context():
            booking = Booking(
                user_email=self.test_email,
                flight_id=self.test_flight.id,
                seats="2B",
                seat_row=1,
                seat_col=1
            )
            db.session.add(booking)
            db.session.commit()
            # Test that __repr__ returns expected string format
            self.assertEqual(str(booking), f'<Booking {booking.id} - Flight {booking.flight_id} Seat {booking.seats}>')
        print("Booking.__repr__ test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/lucasch
cuments/GitHub/cisc327-group11/test_app.py
Running Booking.__repr__ test
Booking.__repr__ test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.136s

OK
```

16. initialized db

Verify that the database is initialized correctly

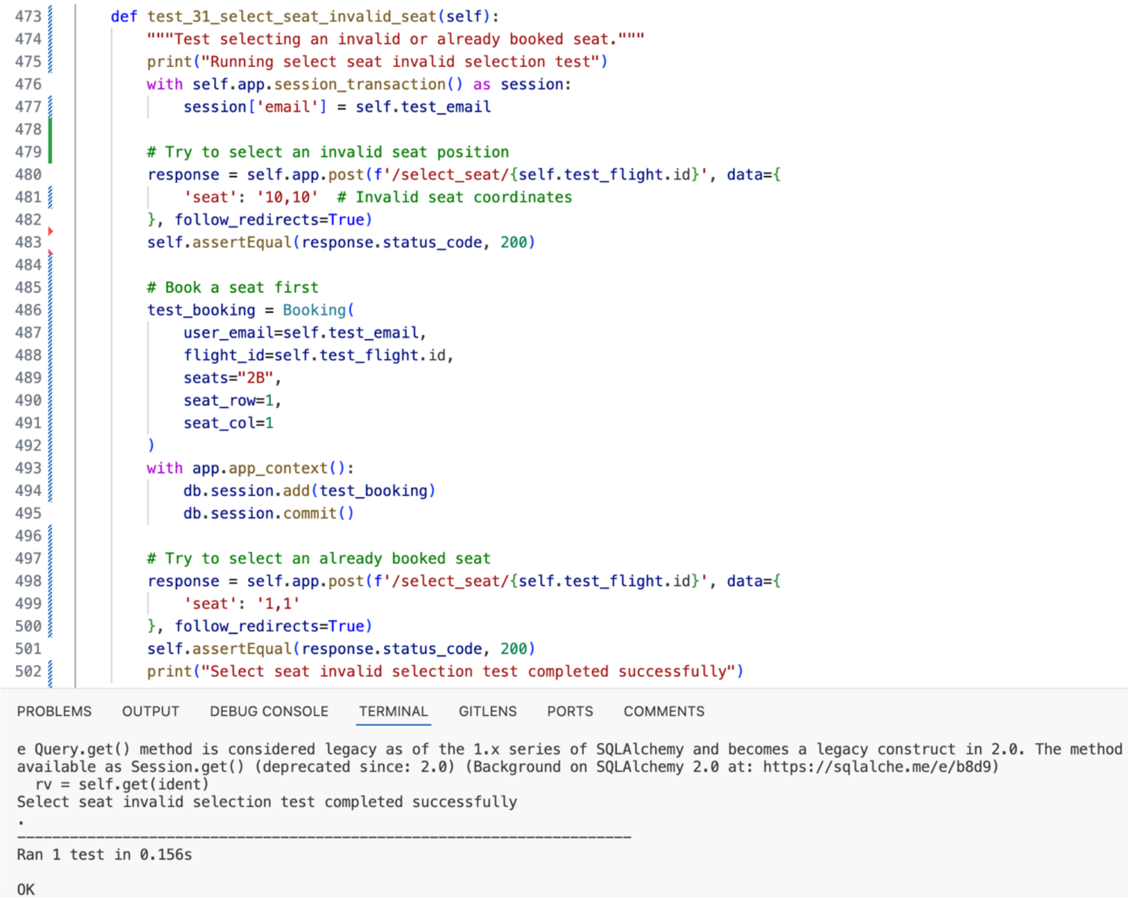


```
    def test_30_init_db(self):
        """Test database initialization function."""
        print("Running init_db test")
        with app.app_context():
            # Clear existing data
            db.drop_all()
            # Test initialization
            init_db()
            # Verify sample flights were created
            flights = Flight.query.all()
            self.assertTrue(len(flights) > 0)
            # Verify seat maps were generated correctly
            for flight in flights:
                self.assertEqual(len(flight.seats), 5)  # 5 rows
                self.assertEqual(len(flight.seats[0]), 4)  # 4 columns
        print("init_db test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running init_db test
init_db test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.143s

OK
```

17. Select seat invalid seat

Verify that a user can select a seat for a flight

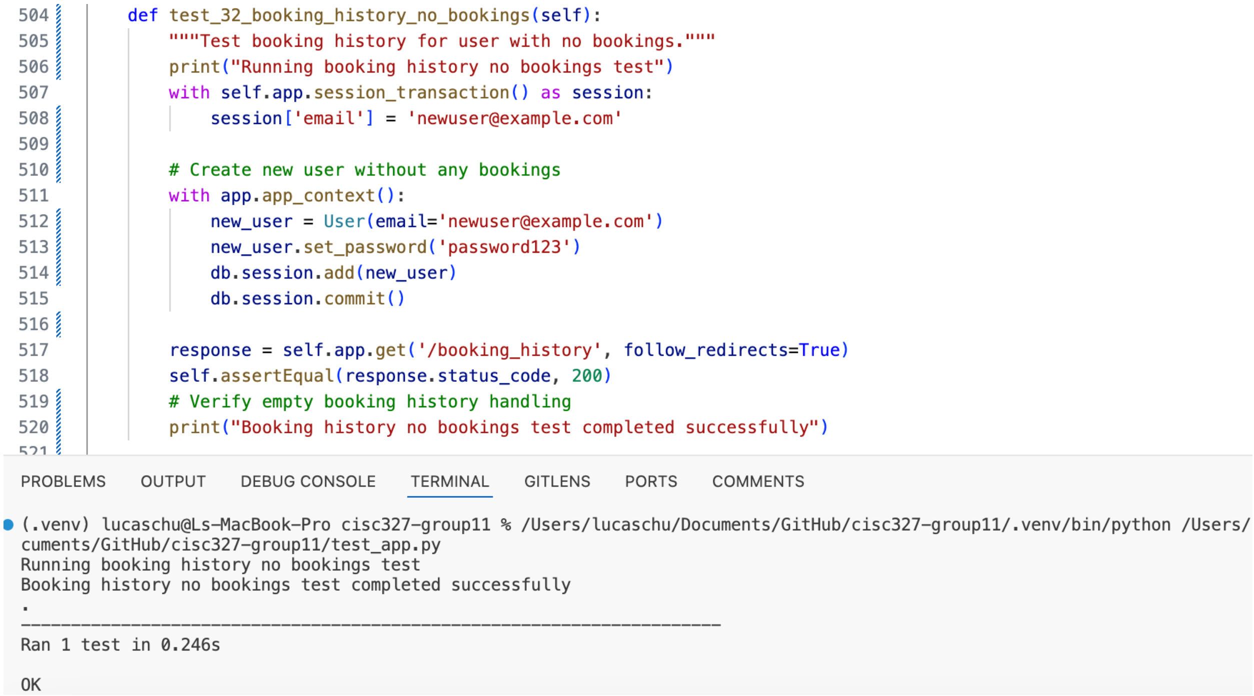


```
    def test_31_select_seat_invalid_seat(self):
        """Test selecting an invalid or already booked seat."""
        print("Running select seat invalid selection test")
        with self.app.session_transaction() as session:
            session['email'] = self.test_email

        # Try to select an invalid seat position
        response = self.app.post(f'/select_seat/{self.test_flight.id}', data={
            'seat': '10,10'  # Invalid seat coordinates
        }, follow_redirects=True)
        self.assertEqual(response.status_code, 200)

        # Book a seat first
        test_booking = Booking(
            user_email=self.test_email,
            flight_id=self.test_flight.id,
            seats="2B",
            seat_row=1,
            seat_col=1
        )
        with app.app_context():
            db.session.add(test_booking)
            db.session.commit()

        # Try to select an already booked seat
        response = self.app.post(f'/select_seat/{self.test_flight.id}', data={
            'seat': '1,1'
        }, follow_redirects=True)
        self.assertEqual(response.status_code, 200)
        print("Select seat invalid selection test completed successfully")
```

```
e Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method
available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  rv = self.get(ident)
Select seat invalid selection test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.156s

OK
```

18. Booking history no bookings

Verify that a user with no bookings has an empty booking history

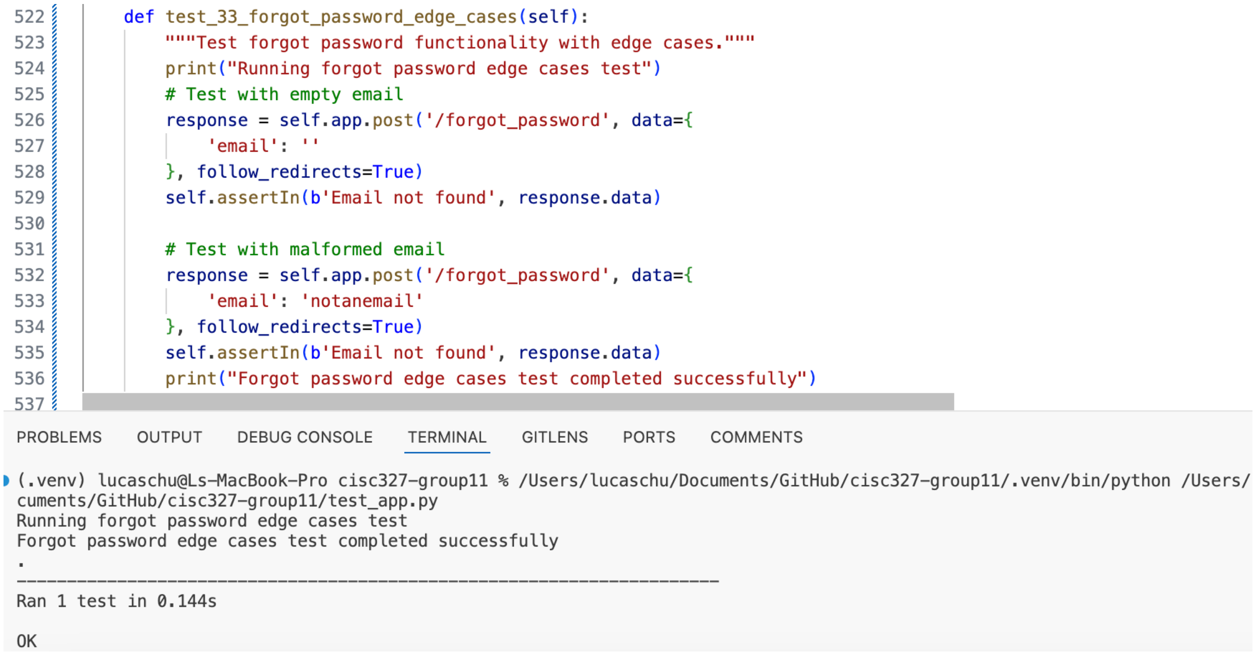


```python
def test_32_booking_history_no_bookings(self):
    """Test booking history for user with no bookings."""
    print("Running booking history no bookings test")
    with self.app.session_transaction() as session:
        session['email'] = 'newuser@example.com'

    # Create new user without any bookings
    with app.app_context():
        new_user = User(email='newuser@example.com')
        new_user.set_password('password123')
        db.session.add(new_user)
        db.session.commit()

    response = self.app.get('/booking_history', follow_redirects=True)
    self.assertEqual(response.status_code, 200)
    # Verify empty booking history handling
    print("Booking history no bookings test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running booking history no bookings test
Booking history no bookings test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.246s

OK
```

19. Forgot password edge cases

Verify if a user cannot reset their password without providing an email



```python
def test_33_forgot_password_edge_cases(self):
    """Test forgot password functionality with edge cases."""
    print("Running forgot password edge cases test")
    # Test with empty email
    response = self.app.post('/forgot_password', data={
        'email': ''
    }, follow_redirects=True)
    self.assertIn(b'Email not found', response.data)

    # Test with malformed email
    response = self.app.post('/forgot_password', data={
        'email': 'notanemail'
    }, follow_redirects=True)
    self.assertIn(b'Email not found', response.data)
    print("Forgot password edge cases test completed successfully")
```

```
(.venv) lucaschu@Ls-MacBook-Pro cisc327-group11 % /Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/bin/python /Users/
cuments/GitHub/cisc327-group11/test_app.py
Running forgot password edge cases test
Forgot password edge cases test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.144s

OK
```