**Test Steps:**

1. Open the following url: http://localhost:9292/payment
2. Fill the sections “Payment Information” and “Sender Information” with proper data as following (amount < 1000 USD: 5% fees)

* Payment Information

Your School -> Stanford

Tuition amount -> 500

Amount received -> 510

Currency -> USD

* Sender Information

Student ID -> 123456abc (value which does not exist in the system)  
Your name -> Test1  
Email -> test1@gmail.com  
Address -> Calle Ortiga 3, Madrid  
Country -> Spain

1. Click on Book a Payment
2. Go to Bookings page, URL: <http://localhost:9191/bookings>
3. Go to the last line and click on Refresh button
4. Repeat step 1
5. Repeat step 2 changing only the following values (1000 =< amount < 10000: 3% fee):

Tuition amount -> 1100

Amount received -> 1100

1. Repeat steps 3-4
2. Go to the last line and click on Refresh button
3. Repeat step 1
4. Repeat step 2 changing only the following values (amount >= 10.000 USD: 2% fees):

Tuition amount -> 11000

Amount received -> 10999

Student ID -> 123456abd (does not exist in the system)

1. Repeat steps 3-4
2. Go to the last line and click on Refresh button
3. Repeat step 1
4. Repeat step 2 changing only the following values (Threshold):

Tuition amount -> 1000099

Amount received -> 1000100

1. Repeat steps 3-4
2. Go to the last line and click on Refresh button
3. Repeat step 1
4. Repeat step 2 changing only the following values (email):
   * Email -> “test1@gmail.”,” test1@gm”, “test1”
5. Repeat steps 3-4
6. Go to the last line and click on Refresh button
7. Repeat steps 14-17 selecting Currency equal to EUR and CAD
8. Repeat steps 1-5 selecting Country equal to China and India

Negative scenarios:

1. Repeat step 1
2. Repeat step 2 adding negative values:
   * Tuition amount -> -500.10
   * Amount received -> -510.10
   * Student ID -> -1235.01
3. Repeat steps 3-4
4. Repeat step 1
5. Repeat step 2 changing the following values
   * Tuition amount -> 500.10
   * Amount received -> 510.10
6. Click on Book a Payment
7. Repeat step 1
8. Repeat step 2 changing only the following value):
   * Value lenght more than 10 in StudentID (i.e StudentID = 5678978910).
9. Repeat step 3

**Test Results:**

1. Verify you are redirected to Flywire Booking Process page
2. Verify fields are properly filled with the data
3. Verify you are redirected to a page getting the following message: Thanks for booking a payment with Flywire, your payment will be processed shortly.
4. Verify you are redirected to Bookings checking panel and you have the following columns: Reference, Amount, Amount with fees, Amount received, Quality check, Over Payment, Under Payment
5. Verify your payment is there and values are equal than the ones entered:
   1. 1st column: random ID assigned to your booking
   2. 2nd column: 500 Amount of the tuition
   3. 3th column: 525 amount of the tuition + 5% fee
   4. 4th column: 510 amount received
   5. Quality check: blank as everything is correct
   6. Over payment checked
   7. Under payment unchecked
6. Same result than step 1
7. Verify the fields are properly filed with the entered values
8. Same results than steps 3-4
9. Verify your payment is there and check the following:
   1. 1st column: random ID assigned to your booking
   2. 2nd column: 1100 Amount of the tuition
   3. 3th column: 1133 amount of the tuition + 3% fee
   4. 4th column: 1100 amount received
   5. Quality check: DuplicatedPayment should appears as StudentID already exist
   6. Over payment unchecked
   7. Under payment unchecked
10. Same result than step 1
11. Verify the fields are properly filed with the entered values
12. Same results than steps 3-4
13. Verify your payment is there) and check the following:
    1. 1st column: random ID assigned to your booking
    2. 2nd column: 11000 - Amount of the tuition
    3. 3th column: 11220 - amount of the tuition + 2% fee
    4. 4th column: 10999 - amount received
    5. Quality check: in blank
    6. Over payment unchecked
    7. Under payment checked
14. Same result than step 1
15. Verify the fields are properly filed with the entered values
16. Same results than steps 3-4
17. Verify your payment is there and check the following:
    1. 1st column: random ID assigned to your booking
    2. 2nd column: 1000099.00 - Amount of the tuition
    3. 3th column: 1020100.98- amount of the tuition + 2% fee
    4. 4th column: 1000100.00 - amount received
    5. Quality check:
       1. DuplicatedPayment should appears as user with same Name already did a booking
       2. AmountThreshold should appear as amount is higher than 1000000
    6. Over payment checked
    7. Under payment unchecked
18. Same result than step 1
19. Verify the fields are properly filed with the entered values
20. Same results than steps 3-4
21. Verify your payment is there) and check the following:
    1. 1st column: random ID assigned to your booking
    2. 2nd column: 500 Amount of the tuition
    3. 3th column: 525 amount of the tuition + 5% fee
    4. 4th column: 510 amount received
    5. Quality check should have:
       1. Duplicate Payment
       2. invalidEmail
    6. Over payment checked
    7. Under payment unchecked
22. Same results than steps 14-17
23. Same results than steps 1-5

Negative scenarios:

1. Same result than step 1
2. Verify the fields are properly filed with the entered values
3. In my opinion here we have a **BUG: an error should appear when clicking on Book a Payment as amount should be always greater than zero.** But, payment registers are introduced in the backend

Apart of that, field Student ID is not validated in this case.

1. Same result than step 1
2. Verify Tuition amounts and Amount received text boxes are marked in red
3. Verify you receive the following error:

Text

Description automatically generated with medium confidence

1. Same result than step1
2. Verify the fields are properly filed with the entered values
3. Verify you’ll receive the following error: **#<RangeError: 5678978910 is out of range for ActiveRecord::Type::Integer with limit 4>**

**Automated testing:**

1. The frontend has been automated (Book a payment). The main goal is to book a payment and verify the payment is properly integrated in the backend (Check Bookings page). The following unit cases have been automated:
   1. Book a payment
   2. Check bookings page
      1. Check fee amount
         1. Verify if amount < 1000 then 5% fee is applied
         2. Verify if 1000 <= amount < 10000 then 3% fee is applied
         3. Verify if amount >= 10000 then 2% fee is applied
      2. Check under-payment and over-payment
      3. Check email and payment status (duplicated or new)
      4. Check threshold

**Performance testing:**

**Scenario 1**: check the delay of the Frontend

FRONTEND

(Form)

TEST

A

FRONTEND

(Form)

A’

A – A’ = Frontend delay

TEST

**Scenario 2**: check the delay of the Backend

BACKEND

(List)

User1

B

BACKEND

(List)

B’

B – B’ = Backend delay

User2

**Scenario 3**: test both operations at the same time and compare response time with number of orders (payments)

FRONTEND

(Form)

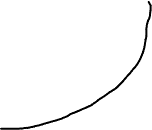
User1

C

BACKEND

(List)

User2



T

R

E

S

P

Nº Orders