Test Suite
for
FRSS V1.0

Submitted by: Group 14

Suryam Arnav Kalra : 19CS30050

Kunal Singh : 19CS30025

Abhinandan De : 19CS10069

Contents

GUI Front End Testing

Instructions:

1. For each functionality the different scenarios, requirements and scopes of error have been explained in general.
2. To test the application, the tester has to follow and execute whatever has been written beside Action
3. The output is correct if it matches with whatever has been given beside Output:
4. The requirements/functionalities are divided into subsections and each section has been assigned alphabets (A, B, C, etc.). They are further divided into subparts as (A1, A2, etc.)
5. The serial number of a requirement/functionality is written after every action that tests it.
6. Some sample tests have been provided in this document. However, extra operations may also be done and more tests may be run and their outputs may be checked with the general guidelines.

Action: User compiles the code and loads the application.
Output: The home screen appears with two options

1. SignUp
2. Login

Action: User clicks on *"Sign Up"* button:
Output : A new window opens which asks us to enter the following inputs:

1. Name
2. UserName (has to be unique)
3. Enter the password.
4. Confirm the password.
5. Enter the address.
6. Enter the phone number.
7. Enter the email-id.
8. Enter the OTP.
9. Switch between Admin or Customer.

The following requirements are checked for a signup:

A1. No field is left empty.
A2. The username is unique.
A3. The password and verify password inputs match.
A4. The email-id filled in by the user is of a valid format.

If any of these is NOT satisfied, an error message informing the error is displayed. Else, an email is sent to the user with the OTP and a confirmation message is shown on the screen.

The user has to enter the OTP, click on "*Verify OTP*" and meet the following requirements.

B1. OTP entered should be the same as the one sent through the mail.
B2. It should be entered within 2 minutes of clicking on the "*Send email*" button.

If any of these is NOT satisfied, an error message informing the error is displayed in a popup. Else, the Complete signup option becomes active and the user can click on Complete SignUp to complete their sign up.

User enters the details to create an admin with the following attributes:

Admin 1

1. Name: Abhinandan
2. Username: Abhi1603
3. Password: Abhi@FRSS
4. Confirm password: Abhi@FRSS
5. Address: 21, Green Park Silchar, Assam
6. Mobile number: 8765456787
7. Email id: abhinandan0316@gmail.com

User tries to create three customer accounts with the following attributes: // TODO

Customer 1

- 1.1. Name: Suryam
- 1.2. Username: Suryam35
- 1.3. Password: Suryam@123
- 1.4. Confirm password: Suryam@123
- 1.5. Address: Dharamsala, Himachal Pradesh
- 1.6. Mobile number: 9764568847
- 1.7. Email id: suryamkalra35@gmail.com

Customer 2

- 2.1. Name: Kunal
- 2.2. Username: ksingh
- 2.3. Password: ksingh312
- 2.4. Confirm password: ksingh312
- 2.5. Address: 23 Sunshine Colony Varanasi
- 2.6. Mobile number: 89781274911
- 2.7. Email id: ksingh1916@gmail.com

Customer 3

- 3.1. Name: Kunal
- 3.2. Username: ksingh12
- 3.3. Password: Manoj_112
- 3.4. Confirm password: Manoj_112
- 3.5. Address: 21, Kalu Sarai, New Delhi
- 3.6. Mobile number: 9897123212
- 3.7. Email id: manojguddiya1@gmail.com

First, the admin account is created. Attributes are filled (according to specified conditions) and the outputs are checked after clicking on the "*Verify details and send OTP*" button after each action.

Action: Fill all the attributes apart from name/mobile number (A1)
Output: A messagebox showing *"Fill all the fields"* is displayed.

Action: Fill Abhi@FRSS for password and Abhi1@FRSS for confirm password (A3)
Output: A messagebox showing *"Passwords do not match"* is displayed.

Action: Fill all details correctly.
Output:A notification showing "*Successfully sent email. Enter OTP in 2 minutes*" is displayed.

Action: Fill a wrong OTP. (B1)
Output: A messagebox showing "*Invalid OTP entered*" is shown.

Action: Fill the correct OTP (within 2 minutes).
Output: A notification showing "*Successfully verified OTP! Click on the Complete SignUp button to log in*" is displayed.

Action: Click on the Complete SignUp button.
Output: A notification showing "*Successfully created account*" is displayed.

Action: Create the first and second customer accounts.
Output: *(successful creation of accounts)*.

Action : While creating the third account enter username "ksingh" (same as second account). (A2)
Output: A messagebox showing "Choose a different username" is displayed.

Action: Fill in the correct username. But enter "*manojguddiya1@*" for email id. (A4)
Output: A notification showing "*Failed to send email! Check username!*" is displayed

Action: Fill everything correctly.
Output: A notification showing "*Successfully sent email. Enter OTP in 2 minutes*" is displayed.

Action: Fill the OTP after 2 minutes. (B2)
Output: A messagebox showing *"Invalid OTP"* is displayed.

Action: Try to verify the data again to resend OTP. Fill the correct OTP, verify and signup.
Output: A notification showing "*Successfully created account*" is displayed.

Action: Close the SignUp Window.
Output: The main screen is displayed once again.

Login

Action: User Clicks on the “Login” button.
Output: A new window opens which asks the user to enter the following inputs:

1. Username.
2. Password.
3. Switch between admin and customer.

The following things are checked once the user clicks the login button:

C1. All fields are filled.
C2. Both the username and password are found in the database.

Action: User keeps password or username field empty and tries to login. (C1)
Output: A messagebox showing “*Fill all the fields*” is displayed.

Action: User enters “Abhi1603” as username and enters “Abhi@frss” as password. (C2)
Output: A messagebox showing “*Invalid username or password*” is displayed.

Action: User enters “Abhi16031” as username and enters “Abhi@FRSS” as password. (C2)
Output: A messagebox showing “*Invalid username or password*” is displayed.

Action: User enters correct details for admin 1 and logs in.
Output: A new window for Admin opens. (will be described later).

Action: User logs out of the admin account and logs in through customer 1.
Output. A new window for Customer opens. (will be described later).

Action: User logs out of the customer account.
Output: The login window opens up once again.

If the details are found in the database, 2 different scenarios arise:

D1. The user is a customer.
D2. The user is an admin.

Case 1. The user is an admin (D1).
A new window opens which is divided into two parts.

I. The left part contains **buttons** for carrying out the following functionalities.

E1 *Create a customer account.*

- The Signup window window opens.
- The admin can create an account for a customer
  Requirement:
  1. Same as for signup (already tested).

E2 *Delete a customer account.*

- A new window opens in which the admin is prompted to enter the username.
- The admin should enter the username and click on the "*delete*" button.
- If the username is not valid an error message confirming the invalidity is displayed in a popup. Else the account is deleted with a confirmation message.
  Requirement:
  1. There should be no associated amount due for the customer account.
  2. The username should be valid.

E3 *See the notifications.*

- A new window opens which contains a list of all the notification messages.
- Whenever any item falls below a threshold, notification is sent to the admin.

E4 *Investment and profit.*

- A new window opens which shows the total investment and the total profit.
- A plot of investment v/s profit is also shown.

E5 *Change price.*

- A new window opens in which the admin is prompted to alter the price of a given type of item.
- The admin should enter the type and the new price and click on the "Alter" button.
- If the record is found, the changes are made and a success popup is displayed.
- If not, a pop up with an error message is displayed.

  Requirement:
  1. The furniture entry should be valid.

E6 *Check returns.*

- A new window opens which prompts the admin to enter the following details:
  - a: Furniture id
  - b: Username (of the customer)
- After entering this, the admin should verify the status of the return i.e. damaged or undamaged.
- After this, the admin should click on the "*Complete the return*" button.
- If damaged,the price of the item gets added to the amount due of the customer.
- Else the item is added back to the inventory with a slight reduction in the interest rate.

  Requirement:
  1. All entries should be valid while verification (furniture id and username).

II. The right part of the main window is used to Add new Furniture to the inventory.

F1 The following details are expected to be entered about the furniture:

1.1 Name
1.2 Type
1.3 Price
1.4 Interest Rate
1.5 Description
1.6 Company

There is a button ("*Choose Image*") which prompts the admin to choose an image.
On clicking this button, a new window opens from which an image of the furniture may be selected. The furniture is added only if all the details are filled.
Else an error message is displayed in a popup.

Requirement:

G1. All fields should be filled.

Case 2. The user is a customer (D2).

A new window opens and is divided into two parts.

A. The right part which contains the catalog.
Here the images of the available products is shown with the following info:

2.1 Id
2.2 Name
2.3 company
2.4 Price
2.5 Description
2.6 Interest Rate

B. The left part which contains buttons for carrying out the following functionalities:

H1 *Buy Now on Loan*:

- On clicking this button a new window opens in which the customer is prompted to enter the id of the furniture and the number of days to loan.
- If the furniture is not found, a suitable error message is displayed in a pop up.
- After entering this, the "*Check Price*" button shows the interest rate and the total price to be paid.
- The customer can then click on the "*Buy Now*" button to complete the loan.

Requirement:

1. Furniture id entered should be correct

H2 *Buy Now on Rent*:

- On clicking this button a new window opens in which the customer is prompted to enter the id of the furniture and the number of days to rent.
- If the furniture is not found, a suitable error message is displayed in a pop up.
- After entering this, the "*Check Price*" button shows the interest rate and the total price to be paid.
- The customer can then click on the "*Buy Now*" button to complete the rent.

Requirement:

1. Furniture id entered should be correct

H3 *Return* :

- On clicking this button, a new window opens in which we can enter the id of the furniture that we want to return.
- The id must be valid else a suitable error message is displayed.
- Then the customer has to click on the "*Add to return*" button.
- This then goes for pending verification by the admin.

Requirement:

1. Furniture id entered should be correct

H4 *Amount due*:

- On clicking this button, a new window opens in which the customer is shown the amount that is due.
- Then he/she is prompted to pay back the desired amount.
- If the amount paid is greater than the amount due, only the amount due is subtracted.
- The entered amount has to be paid back by clicking on the "*Pay Now*" button.

H5 *Past Order History*:

- On clicking this button, a new window opens in which the customer can see their past order history.
- For every order, the order details are shown.

H6 *Search Furniture* :

- On clicking this button, a new window opens where the customer is prompted to enter the name of the furniture.
- If the name matches any of the (substrings of) names of the furniture present in the inventory, it is displayed in the new catalog.
- The customer can note down the id of the furniture for reference.

The following furniture is added with the following specifications:

Furniture 1:

1.1 Gaming chair
1.2 Chair
1.3 1000
1.4 10%
1.5 Modern gaming chair comes with adjustable human-curved lumbar support backrest & flip-up armrest to ease pressure & pain on the body.
1.6 Woody

Furniture 2:

2.1 Gaming chair
2.2 Chair
2.3 1000
2.4 10%
2.5 Modern gaming chair comes with adjustable human-curved lumbar support backrest & flip-up armrest to ease pressure & pain on the body.
2.6 Woody

Furniture 3:

3.1 Deluxe Bed
3.2 Bed
3.3 20000
3.4 12%
3.5 Sturdy double bed made out of good quality pine wood.
3.6 Sleepwell

‘Furniture 4

4.1 Single Bed
4.2 Bed
4.3 10000
4.4 15%
4.5 Good quality single bed with a great look.
4.6 Godrej

Furniture 5

5.1 Deluxe Sofa Set
5.2 Sofa
5.3 1500
5.4 18%
5.5 Comfortable sofa with great cushions!
5.6 Push

A customer account is also added (by the admin).

Customer 4
1.1 Name: Jacky
1.2 Username: CoolJack
1.3 Password: JackIsCool
1.4 Confirm Password: JackIsCool
1.5 Address: South Ex, New Delhi
1.6 Mobile number: 9089218736
1.7 Email-id: jackychan12@gmail.com

State: The admin logs in to his account and tries to enter the details of furniture 1.

Action: All details are specified correctly but image is not chosen! (F1)
Output: A messagebox showing "*Image not chosen*" is displayed

Action: All details are filled apart from Description/Company. (G1)
Output:A messagebox showing "*All fields must be filled*" is displayed.

Action: All the furniture items are added one by one correctly.
Output: The furniture gets added to the inventory.

Action: Admin clicks on "*Create a customer account*" button. (E1)
Output:The signup window gets opened and the process becomes the same as signup.

Action: Admin creates account for customer 4 following the signup process.
Output: Account gets created.

Action: Admin Logs out.
Output: The login page opens up again.

Action: Customer 1 logs in.
Output: Customer 1 gets logged in.

Action: Customer 1 clicks on the "*Search furniture*" button. (H6)
Output A new window opens in which the user is prompted to enter the name of the furniture.

Action: Customer 1 searches "*bat*".
Output Nothing is displayed in the inventory.

Action: Customer 1 searches "*bed*".
Output: Both single bed and double bed are displayed along with all the details.

Action: Customer 1 closes the search window and clicks on "*Buy now on loan*". (H1)
Output: A new window opens in which the id of the furniture is asked along with the number of days.

Action: Customer 1 enters a wrong id / keeps a field empty.
Output: A messagebox displaying the error pops up.

Action: Customer 1 enters the id of the Deluxe bed (2) and rents for 100 days. He clicks on the "*Check Price*" button.
Output: The price is displayed according to the calculated formula.

Action: Customer1 clicks on the "*Buy Now*" button.
Output: The item is bought.

Action: Customer 1 searches "*chair*".
Output : One gaming chair is displayed in the inventory.

Action: Customer 1 now clicks on the "*Buy now on rent*" button. (H2)
Output: A new window opens in which the furniture id and duration is asked.

Action: Customer 1 enters a wrong id / keeps a field empty.
Output: A messagebox displaying the error pops up.

Action: Customer 1 enters the id of the Gaming Chair (2) and rents for 50days. He clicks on the "*Check Price*" button..
Output: The price is displayed according to the calculated formula.

Action: Customer1 clicks on the "*Buy Now*" button.
Output: The item is bought.

Action: Customer 1 checks the items in inventory.
Output: The Deluxe bed is no more to be found. However the gaming chair is still there as there initially were 2 pieces.

Action: Customer 1 logs out.
Output: The login screen opens once again.

Action: Customer 2 logs in to the system.
Output: Customer 2 gets logged in.

Action: Customer 2 rents the gaming chair (following the same process).(H1)
Output: All the actions happen smoothly.

Action: Customer 2 then checks his amount due by clicking on the "*Amount due*" button.(H4)
Output: A non zero amount due is shown.

Action: Customer logs out.
Output: The login screen opens once again.

Action: The admin logs in again and clicks on the "*See Notifications*" button.(E3)
Output: A new window opens in which notifications regarding shortage of Chair and Bed are found.

Action: The admin clicks on the "*Change Price*" button.(E5)
Output: A new window where the price of a furniture may be changed is opened.

Action: Admin tries to change the price of a wrong furniture id (100).
Output: A messagebox showing "*Invalid furniture id*" pops up again.

Action: Admin puts in a correct id (of the gaming chair).
Output: A notification showing "*Price of item updated*" is shown.

Action: Admin logs out.
Output: The login screen opens once again.

Action: Customer 1 logs in and clicks on the "*Return*" button. (H3)
Output: A new window opens where he is prompted to enter the id of the furniture.

Action: He enters a wrong id (200) and clicks on the "*Add to return*" button.
Output: A messagebox showing "*Invalid id entered*" is shown.

Action : He enters the correct id of his rented furniture and returns them one by one by clicking on the "*Add to Return*" button.
Output: A notification showing "*Pending verification by admin*" is displayed for both cases.

Action: He clicks on the "*Amount due*" button.(H4)
Output: A new window opens which shows the pending amount is displayed.
It also prompts him to pay the amount.

Action: He enters half of the displayed amount, clicks on the "*Pay amount due*" button and closes the window. He then reopens it..
Output: The amount due has now been updated.

Action: He now enters an amount which is greater than the amount due and clicks on the "*Pay amount due*" button. He closes the window and reopens it.
Output: The amount due becomes 0 and not negative since it can't be so.

Action: He then logs out.
Output: The login window reopens.

Action: Admin logs in and clicks on the "*Check returns*" button. (E6)
Output: A new window opens which shows the pending returns on the right side.On the left side the admin is prompted to enter the furniture id and the username for a particular order.

Action: The admin at first enters a wrong combination of username and furniture id.
Output: A messagebox showing the mismatch is displayed.

Action: The admin then enters the furniture id of the gaming chair and the username of Customer 1. He marks it as not damaged. (E6)
Output: It gets added to the inventory (can be verified later).

Action: The admin then enters the furniture id of the deluxe bed and the username of Customer 1. He marks it as DAMAGED.
Output: It DOES NOT get added to the inventory (can be verified later). The amount gets added to the amount due of customer 1.

Action: The admin clicks on the "*Delete a customer account*" button.
Output: A new window opens in which the admin is prompted to enter the username of the customer.

Action: The admin enters the username of customer 1 (Suryam) and clicks on the "*Delete*" button. (E2)
Output: A messagebox showing "*Account can't be deleted since dues not cleared*" is displayed.

Action: The admin now tries to delete customer 3 by entering his username and clicking on the "*Delete*" button.
Output: A messagebox confirming the successful deletion is displayed.

Action: The admin clicks on the “*Investment and profit*” button. (E4)
Output: A new window opens which shows total profit and total investment on the left.
It also plots a graph between these two parameters on the right.

Action: The admin closes this and logs out.
Output: The login page gets reopened.

Action: Customer 1 logs in to his account and clicks on the “*Amount due*” button.(H4)
Output: There is a non zero amount due.

Action: He then fills the amount and clicks on the “*Pay amount due*” button and closes the window. He then reopens it.
Output: The amount due has now been updated.

Action: He then looks at the catalog.
Output: The deluxe bed is no longer found in the catalog.

Action: He clicks on the “*Past order history*” button.(H5)
Output: A new window showing his past order history is opened with his two past orders.

Action: He logs out.
Output: The login page reopens.

Action: Customer 2 logs in and clicks on the “*Return*” button.
Output: The return window opens.

Action: He then returns his rented furniture by entering the id and clicking on the “*Add to return*” button.
Output: A notification showing “*Pending verification by admin*” is displayed.

Action: He then logs out.
Output: The login page reopens.

Action: Admin logs in and clicks on the “*Check returns*” button.(E6)
Output: A new window opens which shows the pending returns on the right side.
On the left side the admin is prompted to enter the furniture id and the username for a particular order.

Action: The admin then enters the furniture id of the gaming chair and the username of Customer 2. He marks it as not damaged.
Output: It gets added to the inventory (can be verified later).

Action: Admin logs out.
Output: Login screen reopens.

Action: Customer 4 logs in and views the catalog.
Output: All the initial furniture apart from the deluxe bed (damaged) is shown confirming everything was added back to the inventory.
Also there is a reduction in the interest rate of selected furniture.

Backend Testing

This contains all the unit tests for all the classes and for all the functionalities which have been implemented (eg: insertion into database, deletion from database, etc.). A separate file test.py has also been created for convenience.

Testing the functionalities (using test.py):

```
import mysql.connector

def testsignup(username):
    ex = "SELECT * FROM customers WHERE username = %s"
    va = (username,)
    my_cursor1.execute(ex, va)
    res = my_cursor1.fetchall()
    ex = "SELECT * FROM admins WHERE username = %s"
    va = (username,)
    my_cursor1.execute(ex, va)
    re = my_cursor1.fetchall()
    if len(re) > 0 or len(res) > 0:
        print("User SIGN UP : PASSED")
    else:
        print("USER SIGN UP : FAIL")

def testdeleteuser(username):
    ex = "SELECT * FROM customers WHERE username = %s"
    va = (username,)
    my_cursor1.execute(ex, va)
    res = my_cursor1.fetchall()
    if len(res) == 0:
        print("User DELETE : PASSED")
    else:
        print("USER DELETE : FAILED")

def testnotdeleteduser(username):
    ex = "SELECT * FROM customers WHERE username = %s"
    va = (username,)
    my_cursor1.execute(ex, va)
    res = my_cursor1.fetchall()
    if len(res) == 0:
        print("User DELETE WITH AMOUNT DUE : FAILED")
    else:
        print("USER DELETE WITH AMOUNT DUE : PASSED")
```

```
def testaddfuniture(type, file):
    ex = "SELECT * FROM furnitures WHERE type = %s AND photo = %s"
    print("type here" + type)
    print("photo here" + file)
    va = (type , file)
    print("vaaaa ", va)
    my_cursor1.execute(ex, va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    print(res)
    if len(res) < 1:
        print("ADD FURNITURE : FAILED here")
        return
    else:
        if res[0][5] == type:
            print("ADD FURNITURE : PASSED")
        else:
            print("ADD FURNITURE : FAILED")
        if res[0][7] == file:
            print("CORRECT PHOTO ADDED : PASSED")
        else:
            print("CORRECT PHOTO ADDED : FAILED")

def testinvestment(inv):
    exe = "SELECT SUM(investment) FROM admins"
    my_cursor1.execute(exe)
    investment = my_cursor1.fetchall()
    mydb1.commit()
    if float(investment[0][0]) == inv:
        print("INVESTMENT TEST : PASSED")
        pass
    else:
        print("INVESTMENT TEST : FAILED")

def testprofit(pro):
    exe = "SELECT SUM(profit) FROM admins"
    my_cursor1.execute(exe)
    profit = my_cursor1.fetchall()
    mydb1.commit()
    if float(profit[0][0]) == pro:
        print("PROFIT TEST : PASSED")
        pass
    else:
        print("PROFIT TEST : FAILED")
    pass
```

```
def testchangeprice(type, newprice):
    exe = "SELECT price FROM furnitures WHERE type = %s"
    va = (type,)
    my_cursor1.execute(exe, va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    if(float(res[0][0]) == newprice):
        print("CHANGE PRICE TEST : PASSED")
    else:
        print("CHANGE PRICE TEST : FAILED")


def testfurnituredelete(fur_id):
    exe = "SELECT * FROM furnitures WHERE id = %s"
    va = (fur_id,)
    my_cursor1.execute(exe, va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    if len(res) > 0:
        print("DELETE DAMAGED FURNITURE : FAILED")
    else:
        print("DELETE DAMAGED FURNITURE : PASSED")

def testreadditionfurniture(fur_id):
    exe = "SELECT rented FROM furnitures WHERE id = %s"
    va = (fur_id,)
    my_cursor1.execute(exe, va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    if (res[0][0]) != 0:
        print("READDITION FURNITURE : FAILED")
    else:
        print("READDITION FURNITURE : PASSED")

def testcustomeramountdue(username, amt):
    ex = "SELECT amountdue FROM customers WHERE username = %s"
    va = (username , )
    my_cursor1.execute(ex , va)
    res = my_cursor1.fetchall()[0][0]
    mydb1.commit()
    if amt != res:
        print("AMOUNT DUE TEST : FAILED")
    else:
        print("AMOUNT DUE TEST : PASSED")
```

```
def testpaymentdone(username, amount):
    ex = "SELECT amountdue FROM  customers WHERE username = %s"
    va = (username, )
    my_cursor1.execute(ex , va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    if res[0][0] != amount:
        print("PAYMENT DONE : PASSED")
    else:
        print("PAYMENT DONE : FAILED")

def testfeedbackinsert(typ, feed):
    ex = "SELECT review FROM feedbacks WHERE type = %s"
    va = (typ ,)
    my_cursor1.execute(ex , va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    if res[0][0] != feed:
        print("FEEDBACK INSERTED : FAILED")
    else:
        print("FEEDBACK INSERTED : PASSED")

def testcustomerlogin(username, user_id):
    if username == user_id:
        print("CUSTOMER LOGIN : PASSED")
    else:
        print("CUSTOMER LOGIN : FAILED")

def testadminlogin(username, user_id):
    if username == user_id:
        print("ADMIN LOGIN : PASSED")
    else:
        print("ADMIN LOGIN : FAILED")

def testreturnadded(fur_id):
    ex = "SELECT * FROM current_returns WHERE furniture_id = %s"
    va = (int(fur_id),)
    my_cursor1.execute(ex , va)
    res = my_cursor1.fetchall()
    mydb1.commit()
    if len(res) < 1:
        print("RETURN INITIATED : FAILED")
    else:
        print("RETURN INTIATED : PASSED")
```

```
mydb1 = mysql.connector.connect(host = "localhost",
                                     user = "root",
                                     passwd = "",
                          database = "frss",)

global my_cursor1
my_cursor1 = mydb1.cursor()
```

Unit Testing

```
import unittest

class TestFurnitureMethods(unittest.TestCase):
   myfurniture = Furniture(1, "Center Table", "neelkamal", 4000, "Durable!", "Made of
pure wood", "Table", 0, "Imagepath", 2, 365)

   def test_id(self):
      self.assertEqual(self.myfurniture.getId(), 1)

   def test_name(self):
      self.assertEqual(self.myfurniture.getName(), "Center Table")

   def test_Company(self):
      self.assertEqual(self.myfurniture.getCompany(), "neelkamal")

   def test_Price(self):
      self.assertEqual(self.myfurniture.getPrice(), 4000)

   def test_Feedback(self):
      self.assertEqual(self.myfurniture.getFeedback(), "Durable!")

   def test_Description(self):
      self.assertEqual(self.myfurniture.getDescription(), "Made of pure wood")

   def test_Type(self):
      self.assertEqual(self.myfurniture.getType(), "Table")

   def test_Photo(self):
      self.assertEqual(self.myfurniture.getPhoto(), "Imagepath")

   def test_Interestrate(self):
      self.assertEqual(self.myfurniture.getInterestRate(), 2)
```

```
    def test_Timeframe(self):
        self.assertEqual(self.myfurniture.getTimeFrame(), 365)

class TestAdminMethods(unittest.TestCase):
    my_admin = Admin("Saurav", "12345667", "india", "saurav123", "saurav")

    def test_name(self):
        self.assertEqual(self.my_admin.getName(), "Saurav")

    def test_PhoneNumber(self):
        self.assertEqual(self.my_admin.getPhoneNumber(), "12345667")

    def test_Adress(self):
        self.assertEqual(self.my_admin.getAddress(), "india")

class TestCustomerMethods(unittest.TestCase):
    my_customer = Customer("Saurav Likhar", "15667", "india", "saurav13", "saurav",
1200, 2)

    def test_name(self):
        self.assertEqual(self.my_customer.getName(), "Saurav Likhar")

    def test_PhoneNumber(self):
        self.assertEqual(self.my_customer.getPhoneNumber(), "15667")

    def test_Adress(self):
        self.assertEqual(self.my_customer.getAddress(), "india")

If __name__ == “__main__”:
        unittest.main()
```