Hexaware Foundation Training Python Case Study Report Finance Management System

NAME: Aathirainathan P

DATE: 14-10-2024

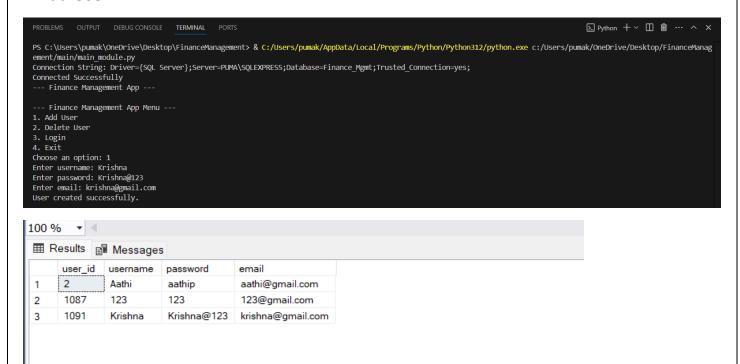
1)Sql Schema:

```
CREATE DATABASE Finance_Mgmt;
USE Finance_Mgmt;
CREATE TABLE Users (
 user_id INT PRIMARY KEY IDENTITY,
 username VARCHAR(50) UNIQUE NOT NULL,
 password VARCHAR(50) NOT NULL,
 email VARCHAR(100) UNIQUE NOT NULL
);
CREATE TABLE ExpenseCategories (
 category_id INT PRIMARY KEY IDENTITY,
 category name VARCHAR(50) NOT NULL
);
CREATE TABLE Expenses (
 expense_id INT PRIMARY KEY IDENTITY,
 user id INT,
 amount DECIMAL(10, 2) NOT NULL,
 category_id INT,
 date DATE NOT NULL,
 description VARCHAR(255),
  FOREIGN KEY (user_id) REFERENCES Users(user_id),
```

```
FOREIGN KEY (category_id) REFERENCES ExpenseCategories(category_id)
);
INSERT INTO ExpenseCategories (category_name)
VALUES ('Food'), ('Transportation'), ('Utilities'), ('Investments');
INSERT INTO ExpenseCategories (category_name)
VALUES
('Entertainment'),
('Healthcare'),
('Insurance'),
('Education'),
('Clothing'),
('Groceries'),
('Dining Out'),
('Subscriptions'),
('Travel'),
('Home Maintenance'),
('Personal Care'),
('Gifts'),
('Charity'),
('Pet Care'),
('Savings'),
('Debt Payments'),
('Fitness'),
('Electronics'),
('Rent'),
('Taxes');
```

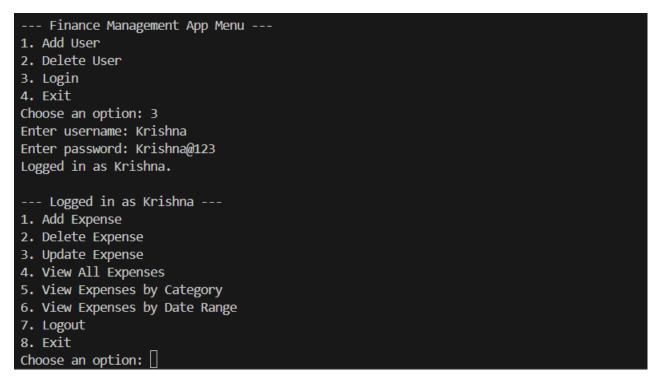
2)Output Screenshots:

1.Add User:



The user named 'Krishna' has been added to the database.

2. Login User:



The user got logged in and he can use the below displayed menu.

3.Add Expense:

- --- Logged in as Krishna --
 1. Add Expense

 2. Delete Expense

 3. Update Expense

 4. View All Expenses

 5. View Expenses by Category

 6. View Expenses by Date Range

 7. Logout

 8. Exit

 Choose an option: 1

 Enter amount: 2500
- Enter category name: Food Enter date (YYYY-MM-DD): 2024-10-10 Enter description: Family Dinner Expense created successfully.
- --- Logged in as Krishna ---
- 1. Add Expense
- 2. Delete Expense
- 3. Update Expense
- 4. View All Expenses
- 5. View Expenses by Category
- 6. View Expenses by Date Range
- 7. Logout
- 8. Exit

Choose an option: 1 Enter amount: 500

Enter category name: Food

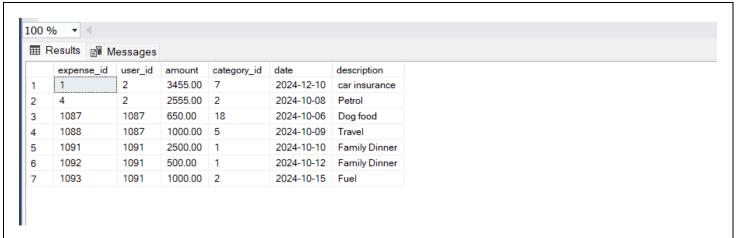
Enter date (YYYY-MM-DD): 2024-10-12 Enter description: Family Dinner Expense created successfully.

- --- Logged in as Krishna ---
- 1. Add Expense
- 2. Delete Expense
- 3. Update Expense
- 4. View All Expenses
- 5. View Expenses by Category
- 6. View Expenses by Date Range
- 7. Logout
- 8. Exit

Choose an option: 1 Enter amount: 1000

Enter category name: Transportation Enter date (YYYY-MM-DD): 2024-10-15

Enter description: Fuel Expense created successfully.



All three expenses have been added to the Expenses table.

4. View All Expenses:

```
--- Logged in as Krishna ---
1. Add Expense
2. Delete Expense
3. Update Expense
4. View All Expenses
5. View Expenses by Category
6. View Expenses by Date Range
7. Logout
8. Exit
Choose an option: 4
Expense ID: 1091, Amount: 2500.00, Category: Food, Date: 2024-10-10, Description: Family Dinner
Expense ID: 1092, Amount: 500.00, Category: Food, Date: 2024-10-12, Description: Family Dinner
Expense ID: 1093, Amount: 1000.00, Category: Transportation, Date: 2024-10-15, Description: Fuel
```

The user gets to view all the expenses he has made.

5. View Expenses by Category:

```
--- Logged in as Krishna ---

1. Add Expense

2. Delete Expense

3. Update Expense

4. View All Expenses

5. View Expenses by Category

6. View Expenses by Date Range

7. Logout

8. Exit

Choose an option: 5

Enter category name: Food

Expense ID: 1091, Amount: 2500.00, Date: 2024-10-10, Description: Family Dinner

Expense ID: 1092, Amount: 500.00, Date: 2024-10-12, Description: Family Dinner
```

Expenses under the category 'Food' is displayed to the user here.

6. View Expenses by Date Range:

```
--- Logged in as Krishna ---

1. Add Expense

2. Delete Expense

3. Update Expense

4. View All Expenses

5. View Expenses by Category

6. View Expenses by Date Range

7. Logout

8. Exit

Choose an option: 6

Enter start date (YYYY-MM-DD): 2024-10-10

Enter end date (YYYY-MM-DD): 2024-10-12

Expense ID: 1091, Amount: 2500.00, Category: Food, Date: 2024-10-12, Description: Family Dinner

Expense ID: 1092, Amount: 500.00, Category: Food, Date: 2024-10-12, Description: Family Dinner
```

Here the user enters dates from **2024-10-10 to 2024-10-12** to get the expense made between those two dates and the user gets the expenses made.

7. Update Expense:

```
--- Logged in as Krishna ---
1. Add Expense
2. Delete Expense
3. Update Expense
4. View All Expenses
5. View Expenses by Category
6. View Expenses by Date Range
7. Logout
8. Exit
Choose an option: 3
Enter expense ID to update: 1092
Enter new amount: 1500
Enter new category name: Food
Enter new date (YYYY-MM-DD): 2024-10-09
Enter new description: Dinner with Friends
Expense updated successfully.
```

Here the updated Expense of the user is successful and can be verified here.

8. Delete Expense:

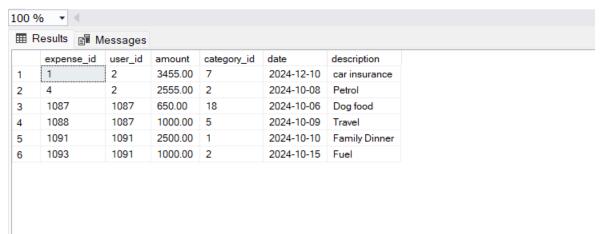
```
Logged in as Krishna ---
1. Add Expense
2. Delete Expense
3. Update Expense
4. View All Expenses

    View Expenses by Category
    View Expenses by Date Range

7. Logout
Choose an option: 2
Enter expense ID to delete: 1092
Expense deleted successfully.
 --- Logged in as Krishna ---
1. Add Expense
2. Delete Expense
3. Update Expense
4. View All Expenses

    View Expenses by Category
    View Expenses by Date Range

7. Logout
8. Exit
Choose an option: 4
Expense ID: 1091, Amount: 2500.00, Category: Food, Date: 2024-10-10, Description: Family Dinner
Expense ID: 1093, Amount: 1000.00, Category: Transportation, Date: 2024-10-15, Description: Fuel
```



The expense has been successfully deleted here and is verified.

9. Logout:

```
--- Logged in as Krishna ---
1. Add Expense
2. Delete Expense
3. Update Expense
4. View All Expenses
5. View Expenses by Category
6. View Expenses by Date Range
7. Logout
8. Exit
Choose an option: 7
User Krishna logged out.
--- Finance Management App Menu ---
1. Add User
2. Delete User
3. Login
4. Exit
Choose an option:
```

The user has been logged out successfully.

10.Delete User:

```
--- Finance Management App Menu ---

1. Add User

2. Delete User

3. Login

4. Exit

Choose an option: 2

Enter username to delete: Krishna

Are you sure you want to delete the user 'Krishna'? (yes/no): yes

User 'Krishna' deleted successfully.
```



The user has been deleted successfully.

3) Exception Handling:

1. ExpenseNotFoundException:

```
class ExpenseNotFoundException(Exception):
    def __init__(self, message="Expense not found"):
        self.message = message
        super().__init__(self.message)
```

```
--- Logged in as Krishna ---

1. Add Expense

2. Delete Expense

3. Update Expense

4. View All Expenses

5. View Expenses by Category

6. View Expenses by Date Range

7. Logout

8. Exit

Choose an option: 2

Enter expense ID to delete: 99999

Expense with ID 99999 not found.
```

Expection Id is not found.

2.UserNotFoundException:

```
class UserNotFoundException(Exception):
    def __init__(self, message="User not found"):
        self.message = message
        super().__init__(self.message)
```

```
--- Finance Management App Menu ---

1. Add User

2. Delete User

3. Login

4. Exit

Choose an option: 2

Enter username to delete: NonexistentUser

User with username 'NonexistentUser' not found.
```

NonExistentUser is not found.

3) Testing:

1.Test User Creation:

```
import sys
import os
base_dir = os.path.abspath(os.path.join(os.path.dirname(_ file__), ".."))
sys.path.append(base_dir)
import unittest
from dao.FinanceRepositoryImpl import FinanceRepositoryImpl
from entity.User import User
class TestUserCreation(unittest.TestCase):
    def setUp(self):
        self.repo = FinanceRepositoryImpl()
    def test_user_creation(self):
        new_user = User(username="test_user", password="password123",
email="test user@example.com")
        success = self.repo.create_user(new_user)
        fetched_user = self.repo.get_user_by_username("test_user")
        self.assertTrue(success)
        self.assertIsNotNone(fetched user)
        self.assertEqual(fetched_user.get_username(), "test_user")
    def tearDown(self):
        test_user = self.repo.get_user_by_username("test_user")
            self.repo.delete user(test user.get user id())
if __name__ == '__main__':
    unittest.main()
```

Output:

2.Test Search Expense:

```
import sys
import os
base dir = os.path.abspath(os.path.join(os.path.dirname( file ), ".."))
sys.path.append(base dir)
import unittest
from entity. Expense import Expense
from entity.User import User
from dao.FinanceRepositoryImpl import FinanceRepositoryImpl
class TestSearchExpense(unittest.TestCase):
    def setUp(self):
        self.repo = FinanceRepositoryImpl()
        self.test_user = User(username="search_user", password="password123",
email="search user@example.com")
        self.repo.create user(self.test user)
        self.test user = self.repo.get user by username("search user")
        category_id = self.repo.get_category_id_by_name("Utilities")
        self.expense = Expense(user id=self.test user.get user id(), amount=50,
category id=category id, date="2024-10-11", description="Electricity Bill")
        self.repo.create_expense(self.expense)
    def test search expense(self):
        expenses = self.repo.get_all_expenses(self.test_user.get_user_id())
        self.assertEqual(len(expenses), 1)
        self.assertEqual(expenses[0].get_description(), "Electricity Bill")
    def tearDown(self):
        expenses = self.repo.get all expenses(self.test user.get user id())
        for expense in expenses:
            self.repo.delete expense(expense.get expense id())
        self.repo.delete user(self.test user.get user id())
if __name__ == '__main__':
    unittest.main()
```

3.Test Expense Creation:

```
import sys
import os
base_dir = os.path.abspath(os.path.join(os.path.dirname(_ file__), ".."))
sys.path.append(base dir)
import unittest
from entity. Expense import Expense
from entity.User import User
from dao.FinanceRepositoryImpl import FinanceRepositoryImpl
class TestExpenseCreation(unittest.TestCase):
    def setUp(self):
        self.repo = FinanceRepositoryImpl()
        self.test_user = User(username="expense_user", password="password123",
email="expense_user@example.com")
        self.repo.create user(self.test user)
        self.test_user = self.repo.get_user_by_username("expense_user")
    def test_expense_creation(self):
        category_id = self.repo.get_category_id_by_name("Food")
        new_expense = Expense(user_id=self.test_user.get_user_id(), amount=1000,
category_id=category_id, date="2024-10-11", description="Dinner")
        success = self.repo.create_expense(new_expense)
        expenses = self.repo.get_all_expenses(self.test_user.get_user_id())
        self.assertTrue(success)
        self.assertGreater(len(expenses), 0)
        self.assertEqual(expenses[0].get_description(), "Dinner")
    def tearDown(self):
        expenses = self.repo.get_all_expenses(self.test_user.get_user_id())
        for expense in expenses:
            self.repo.delete_expense(expense.get_expense_id())
        self.repo.delete_user(self.test_user.get_user_id())
if __name__ == " main ":
    unittest.main()
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\pumak\OneDrive\Desktop\FinanceManagement> & C:\Users/pumak\AppData/Local/Programs/Python/Python312/python.exe c:\Users/pumak\OneDrive\Desktop\FinanceManagement> & C:\Users/pumak\OneDrive\Desktop\FinanceManagement> & C:\Users/pumak\OneDrive\Desktop\FinanceManagement> & C:\Users/pumak\OneDrive\Desktop\FinanceManagement> & C:\Users\pumak\OneDrive\Desktop\FinanceManagement> & C:\Users\pumak\OneDrive
```

4.Test Exceptions:

```
import sys
import os
base_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
sys.path.append(base_dir)
import unittest
from dao.FinanceRepositoryImpl import FinanceRepositoryImpl
from exception. UserNotFoundException import UserNotFoundException
from exception.ExpenseNotFoundException import ExpenseNotFoundException
base_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
sys.path.append(base_dir)
class TestExceptions(unittest.TestCase):
    def setUp(self):
        self.repo = FinanceRepositoryImpl()
    def test_user_not_found_exception(self):
        with self.assertRaises(UserNotFoundException):
            self.repo.get_user_by_username("non_existent_user")
    def test_expense_not_found_exception(self):
        with self.assertRaises(ExpenseNotFoundException):
            self.repo.delete_expense(99999)
if __name__ == '__main__':
    unittest.main()
```