Django DBMS Project

Team members:
Chao Chen
Yihang Zhao
Saheed Adepoju

Outline

- Introduction
- Requirements Analysis
- Implementation
- Demo

Introduction

- Project commitment:

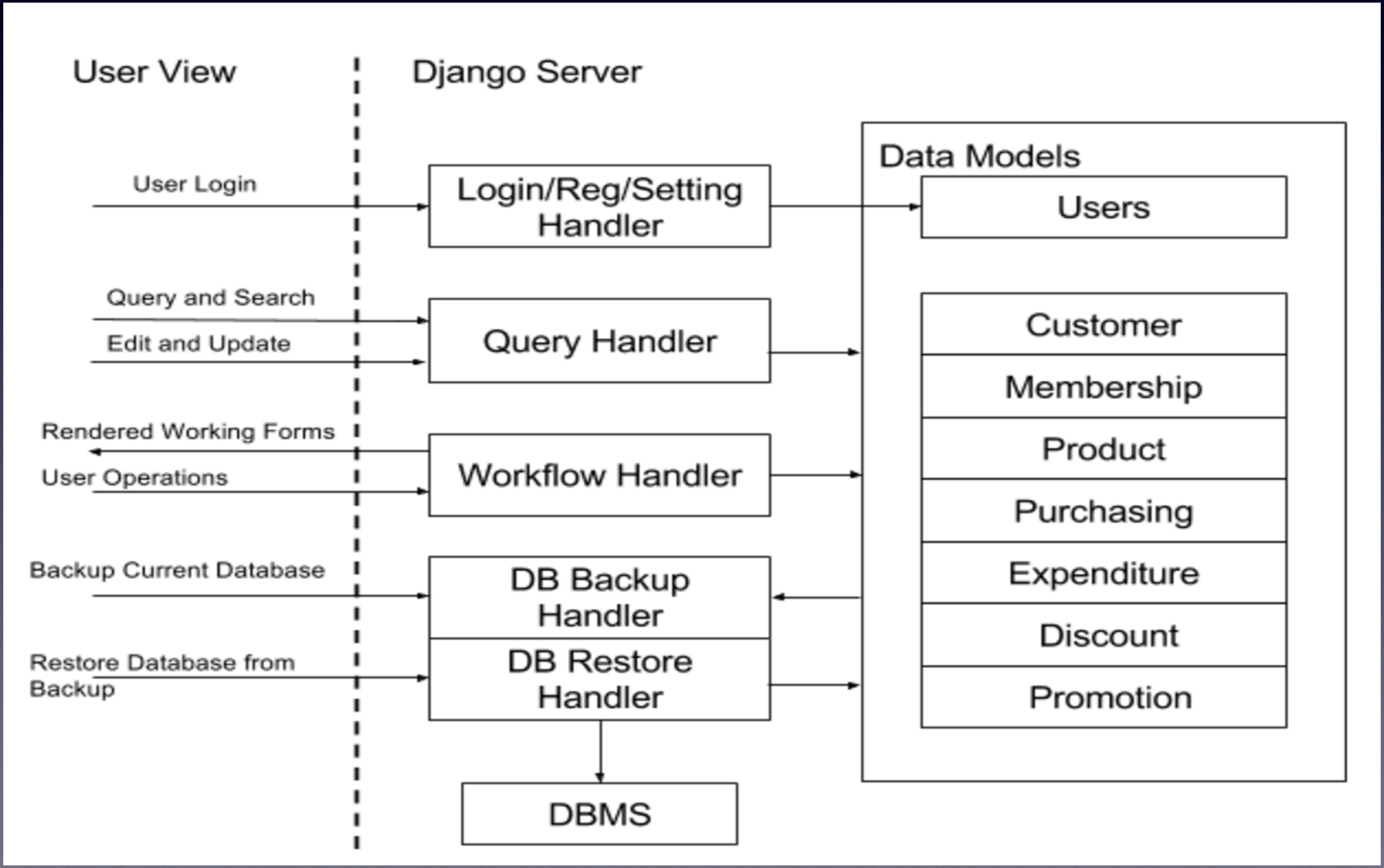
  Delivering a web-based application for resource management and regular works conduct of general purpose store

- Users:

  Store Manger, Staff

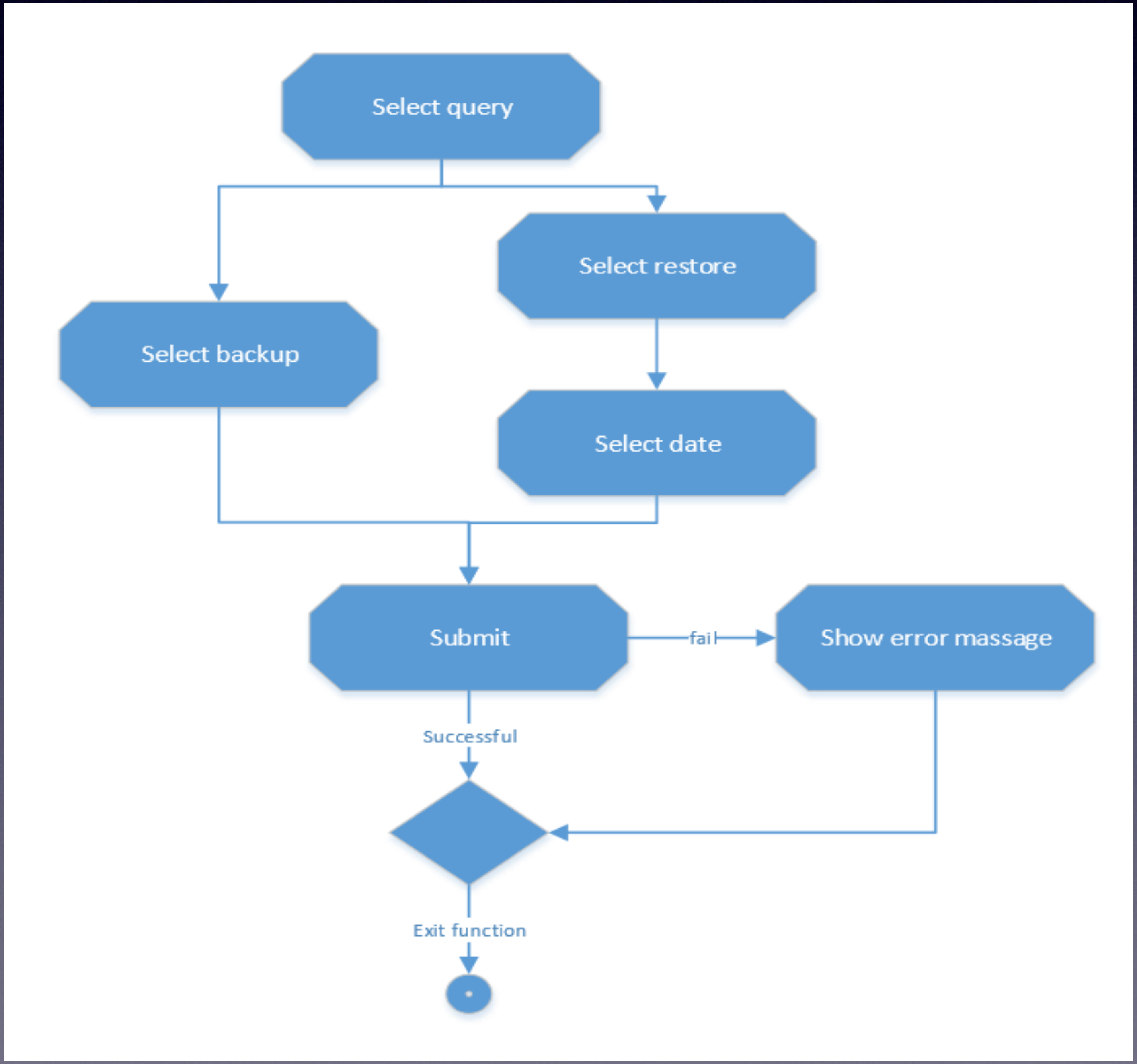
Design

- Architecture



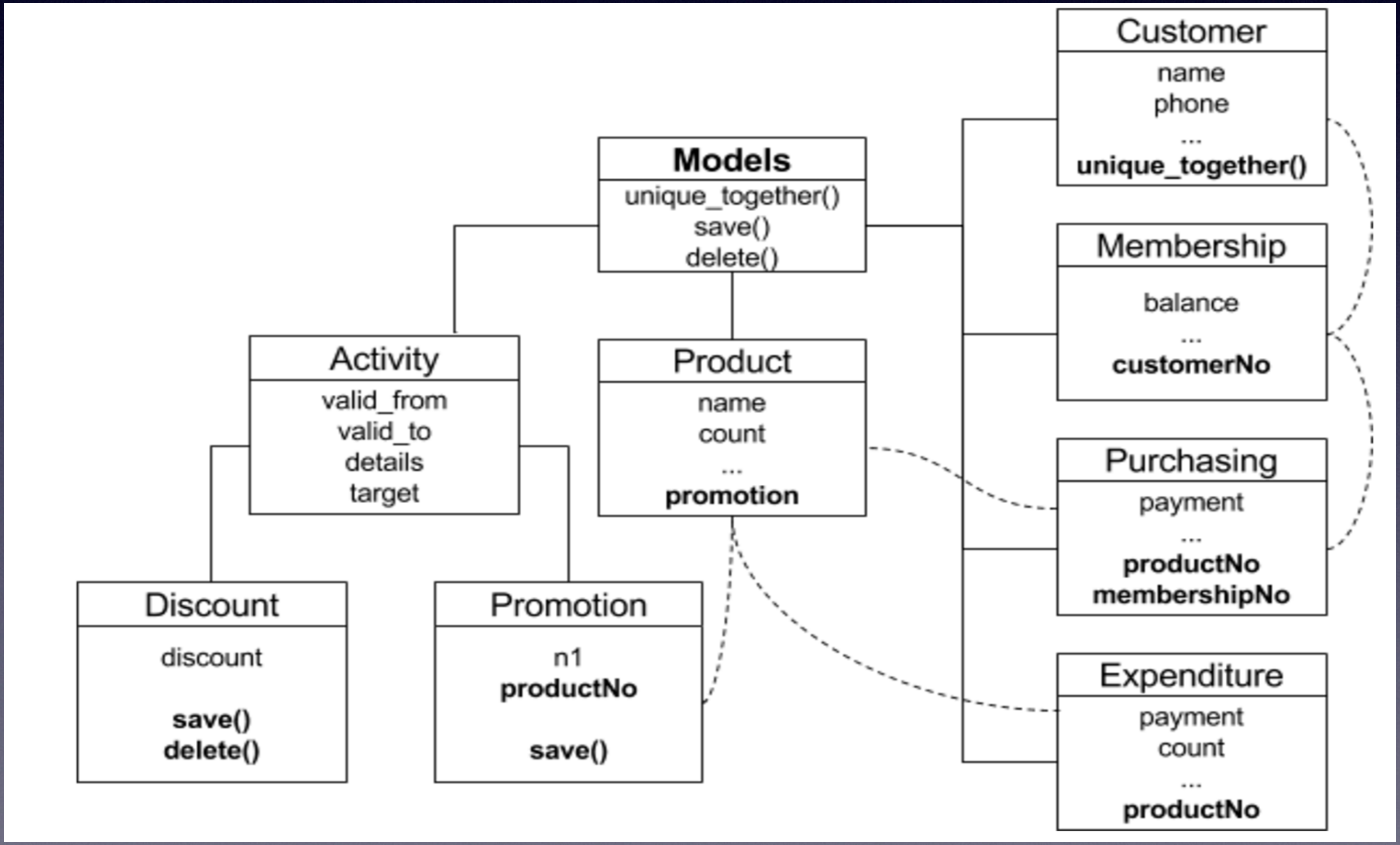
Design

- Activity Diagram

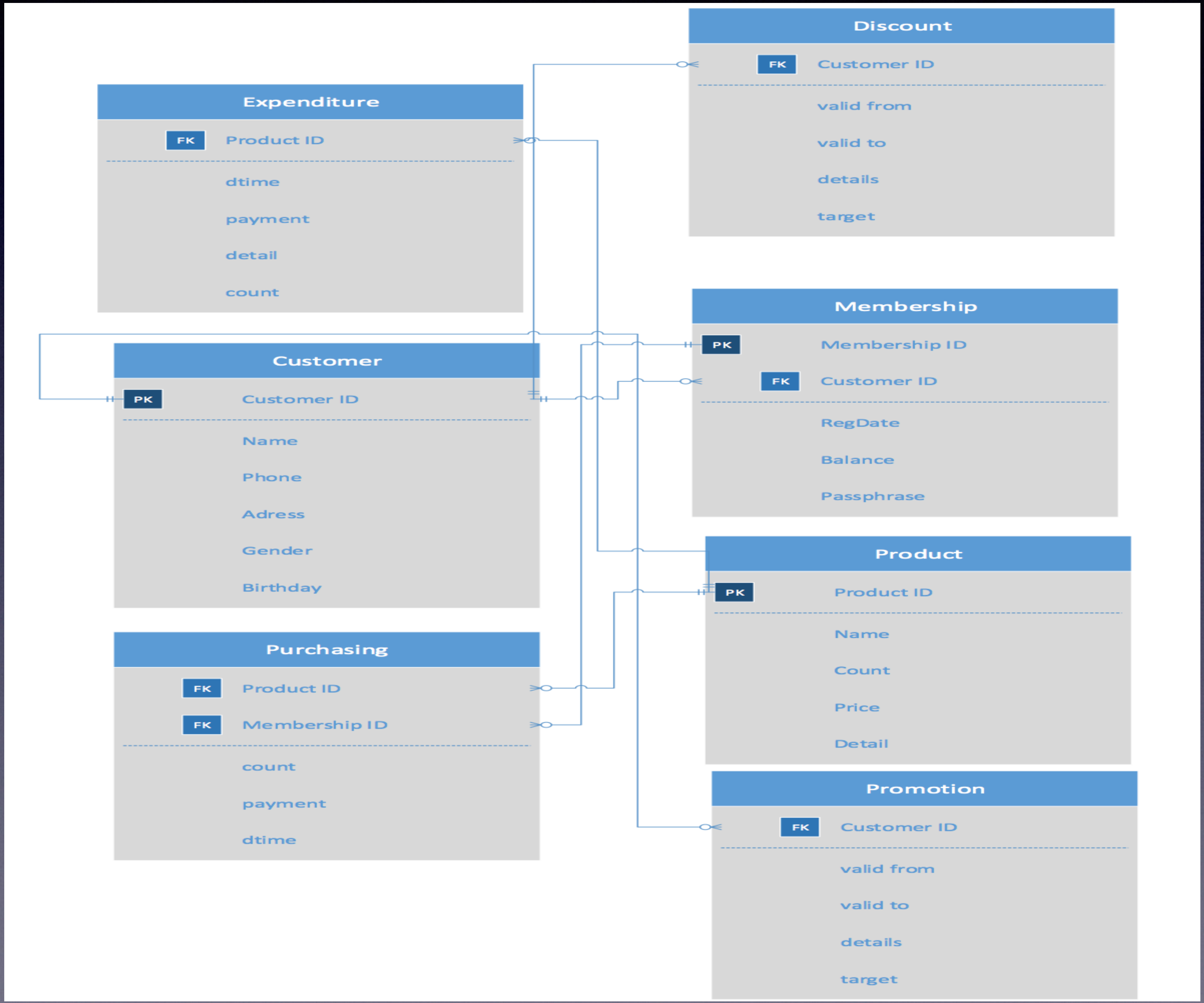


Design

- Class an Components Diagram



Database



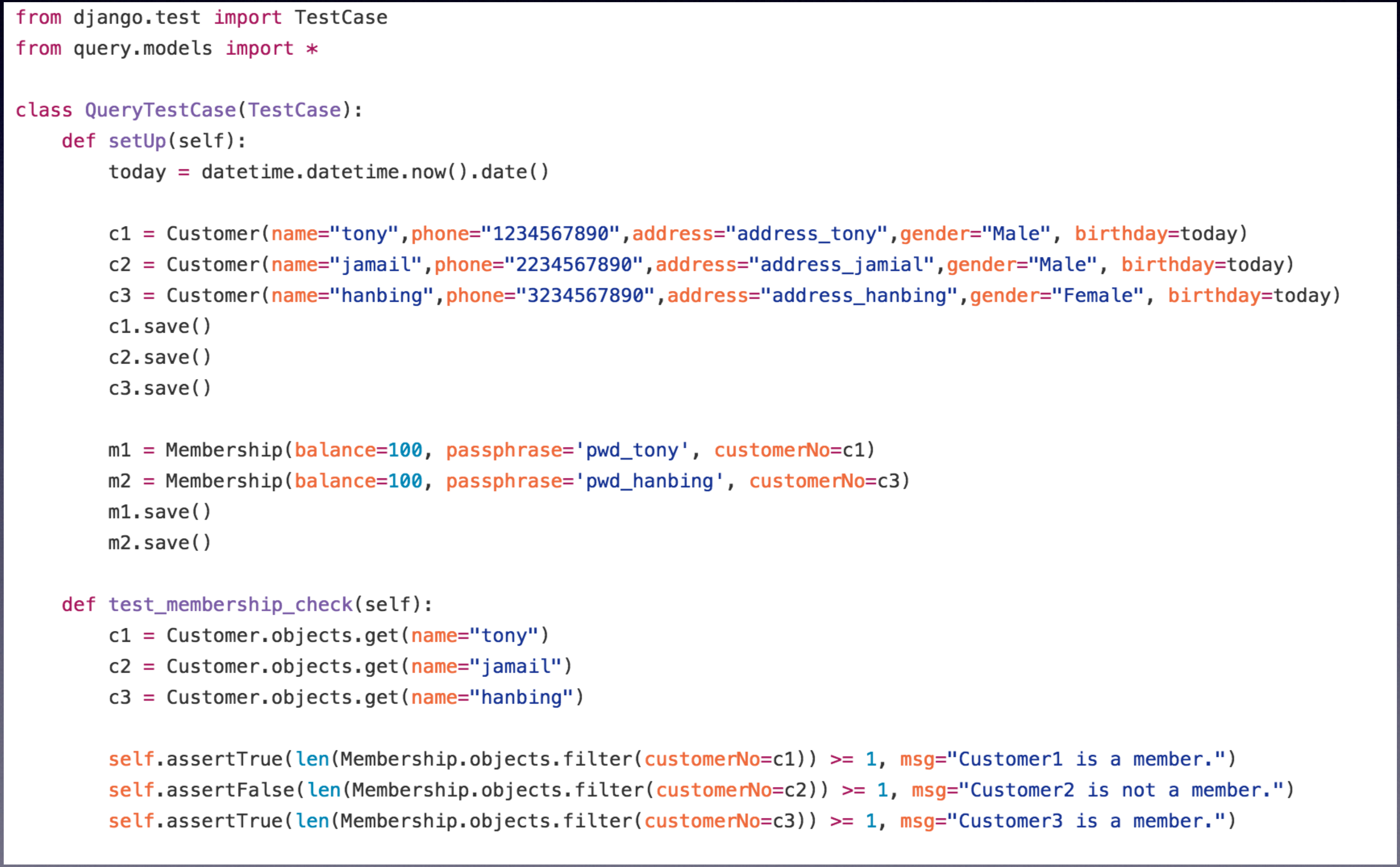
Discount
FK
Customer ID
valid from
valid to
details
target
Expenditure
FK
Product ID
dtime
payment
detail
count
Membership
PK
Membership ID
FK
Customer ID
RegDate
Balance
Passphrase
Customer
PK
Customer ID
Name
Phone
Adress
Gender
Birthday
Product
PK
Product ID
Name
Count
Price
Detail
Purchasing
FK
Product ID
FK
Membership ID
count
payment
dtime
Promotion
FK
Customer ID
valid from
valid to
details
target

Implementation

- Server Host:
  - Python2.7
  - Django 1.8.4
  - MySQL(Optional)
- UI and Interaction
  - JavaScript
  - jQuery/FancyBox/Bootstrap

Implementation

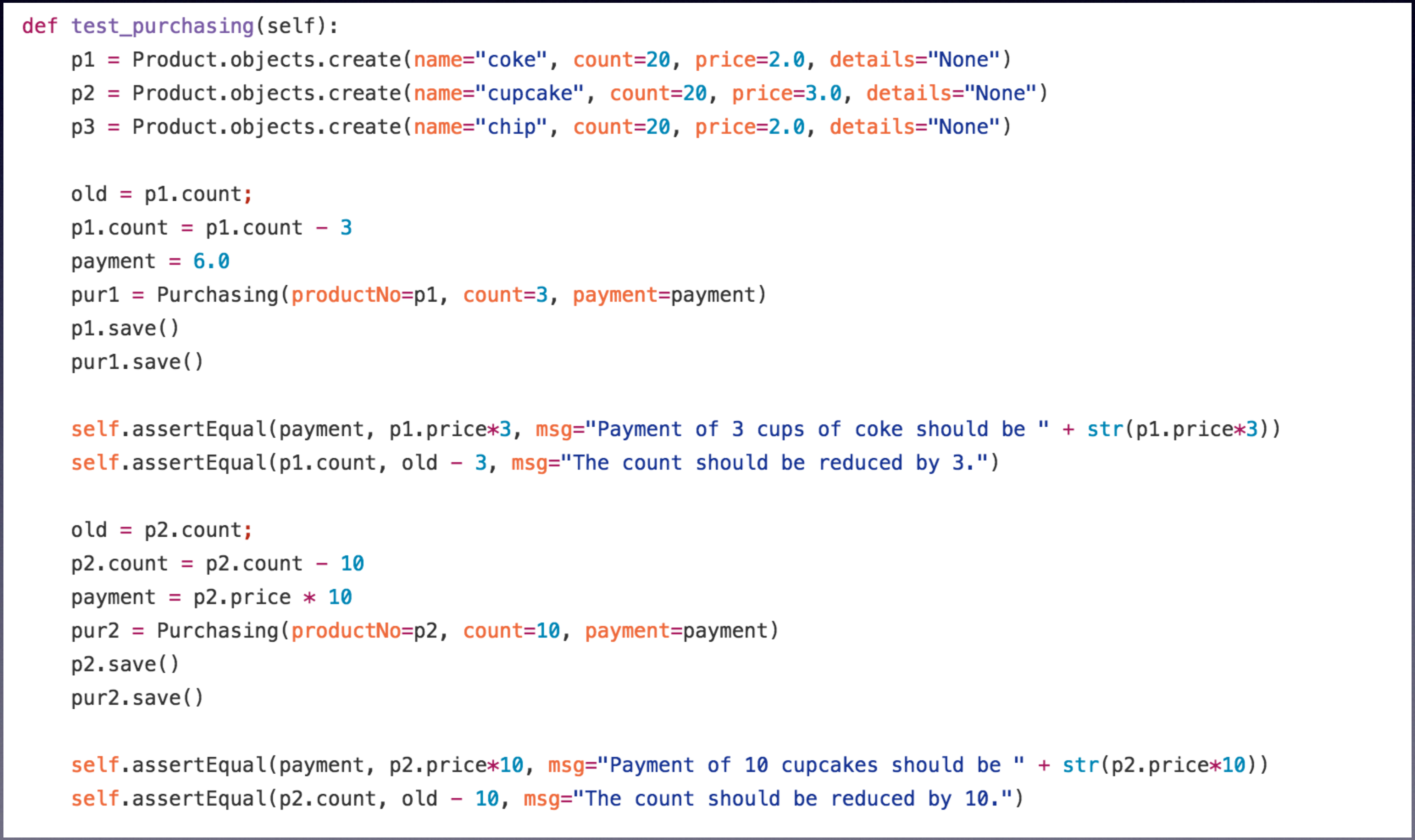
- UnitTests for Data Model



```python
from django.test import TestCase
from query.models import *

class QueryTestCase(TestCase):
    def setUp(self):
        today = datetime.datetime.now().date()

        c1 = Customer(name="tony",phone="1234567890",address="address_tony",gender="Male", birthday=today)
        c2 = Customer(name="jamail",phone="2234567890",address="address_jamial",gender="Male", birthday=today)
        c3 = Customer(name="hanbing",phone="3234567890",address="address_hanbing",gender="Female", birthday=today)
        c1.save()
        c2.save()
        c3.save()

        m1 = Membership(balance=100, passphrase='pwd_tony', customerNo=c1)
        m2 = Membership(balance=100, passphrase='pwd_hanbing', customerNo=c3)
        m1.save()
        m2.save()

    def test_membership_check(self):
        c1 = Customer.objects.get(name="tony")
        c2 = Customer.objects.get(name="jamail")
        c3 = Customer.objects.get(name="hanbing")

        self.assertTrue(len(Membership.objects.filter(customerNo=c1)) >= 1, msg="Customer1 is a member.")
        self.assertFalse(len(Membership.objects.filter(customerNo=c2)) >= 1, msg="Customer2 is not a member.")
        self.assertTrue(len(Membership.objects.filter(customerNo=c3)) >= 1, msg="Customer3 is a member.")
```

Implementation

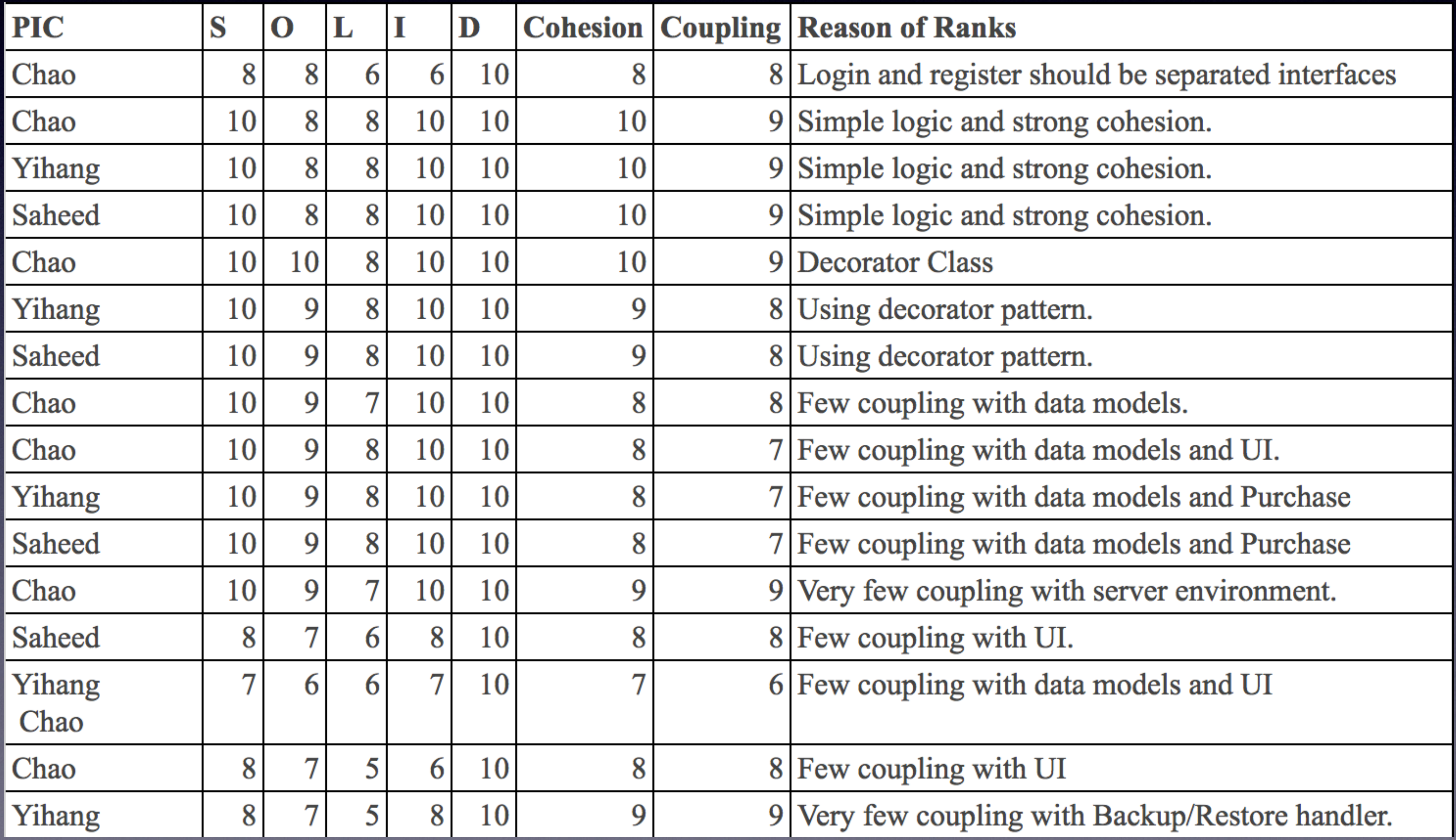
- UnitTests for Data Model



```python
def test_purchasing(self):
    p1 = Product.objects.create(name="coke", count=20, price=2.0, details="None")
    p2 = Product.objects.create(name="cupcake", count=20, price=3.0, details="None")
    p3 = Product.objects.create(name="chip", count=20, price=2.0, details="None")

    old = p1.count;
    p1.count = p1.count - 3
    payment = 6.0
    pur1 = Purchasing(productNo=p1, count=3, payment=payment)
    p1.save()
    pur1.save()

    self.assertEqual(payment, p1.price*3, msg="Payment of 3 cups of coke should be " + str(p1.price*3))
    self.assertEqual(p1.count, old - 3, msg="The count should be reduced by 3.")

    old = p2.count;
    p2.count = p2.count - 10
    payment = p2.price * 10
    pur2 = Purchasing(productNo=p2, count=10, payment=payment)
    p2.save()
    pur2.save()

    self.assertEqual(payment, p2.price*10, msg="Payment of 10 cupcakes should be " + str(p2.price*10))
    self.assertEqual(p2.count, old - 10, msg="The count should be reduced by 10.")
```

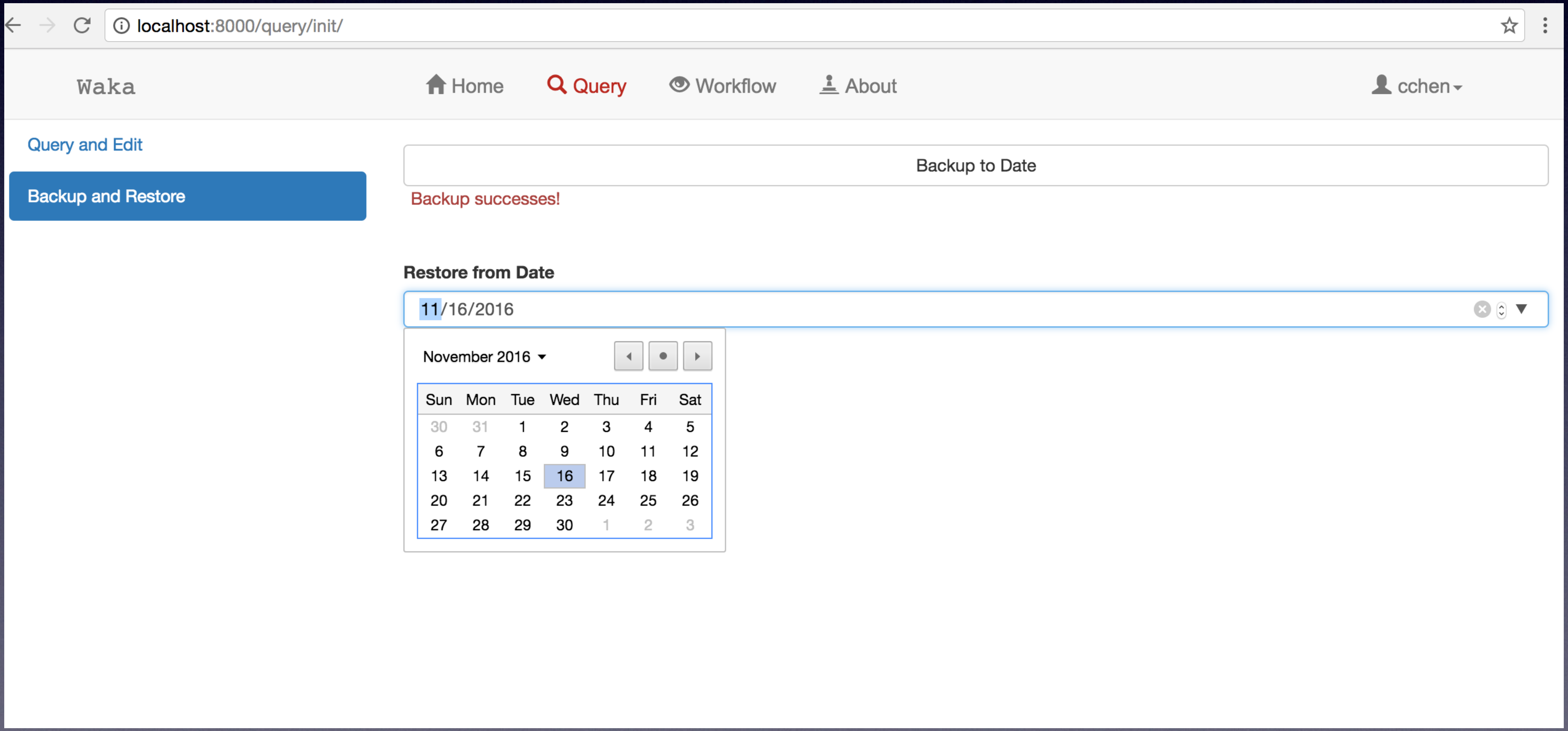
Implementation

- Design Module Matrix



PIC	S	O	L	I	D	Cohesion	Coupling	Reason of Ranks
Chao	8	8	6	6	10	8	8	Login and register should be separated interfaces
Chao	10	8	8	10	10	10	9	Simple logic and strong cohesion.
Yihang	10	8	8	10	10	10	9	Simple logic and strong cohesion.
Saheed	10	8	8	10	10	10	9	Simple logic and strong cohesion.
Chao	10	10	8	10	10	10	9	Decorator Class
Yihang	10	9	8	10	10	9	8	Using decorator pattern.
Saheed	10	9	8	10	10	9	8	Using decorator pattern.
Chao	10	9	7	10	10	8	8	Few coupling with data models.
Chao	10	9	8	10	10	8	7	Few coupling with data models and UI.
Yihang	10	9	8	10	10	8	7	Few coupling with data models and Purchase
Saheed	10	9	8	10	10	8	7	Few coupling with data models and Purchase
Chao	10	9	7	10	10	9	9	Very few coupling with server environment.
Saheed	8	7	6	8	10	8	8	Few coupling with UI.
Yihang Chao	7	6	6	7	10	7	6	Few coupling with data models and UI
Chao	8	7	5	6	10	8	8	Few coupling with UI
Yihang	8	7	5	8	10	9	9	Very few coupling with Backup/Restore handler.

Demo


localhost:8000/query/init/

Backup successes!

Restore from Date

11/16/2016

November 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3