COM5940: NEW MEDIA BUSINESS MODEL & INNOVATION
INTEGRATE JS FRONT-END WITH FLASK REST API BACK-END SERVICES

Bernard Suen
Center for Entrepreneurship
Chinese University of Hong Kong

CUHK BUSINESS SCHOOL
Center for Entrepreneurship

Today's Agenda

1. Use FlaskAlchemy for MySQL data connection and CRUD support.
2. Use Airtable APIs for providing CRUD and authentication support.
3. Utilize JSON data in jQuery Datatable, Leaflet & C3.
4. Enable RESTful API by turning MySQL data into JSON for connection with mobile front-end.
5. Unit, integration and user acceptance tests.

Introducing Flask SQLAlchemy

CRUD Support in SQLAlchemy and Airtable for View Integration

Unit Testing
Integration Testing
User Acceptance Testing

Why FlaskSQLAlchemy if we already have MySQLdb or Pymysql?

1. Automatic SQL mapping to Python objects, so no need to know SQL. But one has to learn the object naming convention.
2. Database independent so once one learns how to use it on MySQL, it will work almost the same on Oracle, MSSQL, PostgresSQL, etc.
3. Practical and simple pagination support.
4. The reason why FlaskSQLAlchemy wasn't taught first instead of Pymysql and MySQLdb is because I think it's important for you to learn SQL.

SQLAlchemy

THE DATABASE TOOLKIT FOR PYTHON

home features blog library community download

SQLAlchemy 1.3 Documentation

Release: **1.3.13** CURRENT RELEASE | Release Date: January 22, 2020

SQLAlchemy 1.3 Documentation

CURRENT RELEASE

Contents | Index

Search terms: search...

SQLAlchemy 1.3 Documentation

Overview
SQLAlchemy ORM
SQLAlchemy Core

Dialects

The **dialect** is the system SQLAlchemy uses to communicate with various types of DBAPI implementations and databases. The sections that follow contain reference documentation and notes specific to the usage of each backend, as well as notes for the various DBAPIs.

All dialects require that an appropriate DBAPI driver is installed.

Included Dialects

https://docs.sqlalchemy.org/en/13/dialects/

- PostgreSQL
- MySQL
- SQLite
- Oracle
- Microsoft SQL Server

Included, but not currently supported dialects

The following dialects have implementations within SQLAlchemy, but they are not part of continuous integration testing nor are they actively developed. These dialects may be removed in future major releases.

- Firebird

https://docs.sqlalchemy.org/en/13/dialects/

Utilities

class `flask_sqlalchemy.`**`Pagination`**(*query*, *page*, *per_page*, *total*, *items*)

Internal helper class returned by **`BaseQuery.paginate()`**. You can also construct it from any other SQLAlchemy query object if you are working with other libraries. Additionally it is possible to pass *None* as query object in which case the **`prev()`** and **`next()`** will no longer work.

`has_next`

True if a next page exists.

`has_prev`

True if a previous page exists

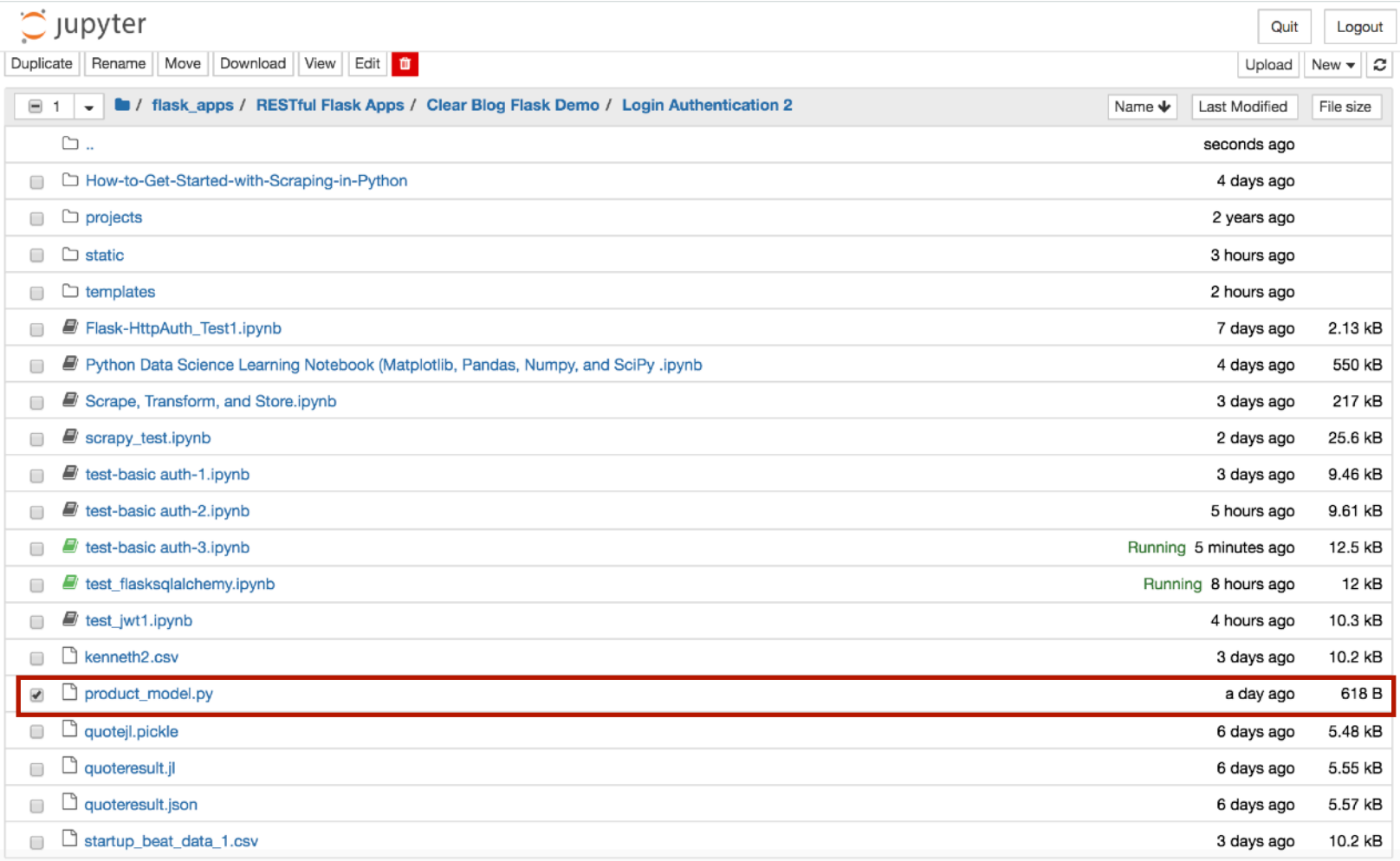
`items` = *None*

the items for the current page

`iter_pages`(*left_edge=2*, *left_current=2*, *right_current=5*, *right_edge=2*)

Iterates over the page numbers in the pagination. The four parameters control the thresholds how many numbers should be produced from the sides. Skipped page numbers are represented as *None*. This is how you could render such a pagination in the templates:

https://flask-sqlalchemy.palletsprojects.com/en/2.x/api/


jupyter
Quit
Logout
Duplicate
Rename
Move
Download
View
Edit
Upload
New
1
/ flask_apps / RESTful Flask Apps / Clear Blog Flask Demo / Login Authentication 2
Name
Last Modified
File size
..
seconds ago
How-to-Get-Started-with-Scraping-in-Python
4 days ago
projects
2 years ago
static
3 hours ago
templates
2 hours ago
Flask-HttpAuth_Test1.ipynb
7 days ago
2.13 kB
Python Data Science Learning Notebook (Matplotlib, Pandas, Numpy, and SciPy .ipynb
4 days ago
550 kB
Scrape, Transform, and Store.ipynb
3 days ago
217 kB
scrapy_test.ipynb
2 days ago
25.6 kB
test-basic auth-1.ipynb
3 days ago
9.46 kB
test-basic auth-2.ipynb
5 hours ago
9.61 kB
test-basic auth-3.ipynb
Running 5 minutes ago
12.5 kB
test_flasksqlalchemy.ipynb
Running 8 hours ago
12 kB
test_jwt1.ipynb
4 hours ago
10.3 kB
kenneth2.csv
3 days ago
10.2 kB
product_model.py
a day ago
618 B
quotejl.pickle
6 days ago
5.48 kB
quoteresult.jl
6 days ago
5.55 kB
quoteresult.json
6 days ago
5.57 kB
startup_beat_data_1.csv
3 days ago
10.2 kB

product_model.py

Similar to how we setup MySQL database using py mysql and MySQLdb, we use the following codes to set up Flask SQLAlchemy to connect to the MySQL database.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:root@localhost/southwind'
app.config['SECRET_KEY'] = "mysecret"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class Products(db.Model):
    __tablename__ = 'products'
    productID = db.Column(db.Integer, primary_key=True)
    productCode = db.Column(db.String(3))
    name = db.Column(db.String(30))
    quantity = db.Column(db.Integer)
    price = db.Column(db.Float)
    supplierID = db.Column(db.Integer)
    # description = db.Column(db.Text)
```

Since SQLAlchemy is a object-based (i.e. class) database mapper, it uses object notation to carry out its operations.

```python
from product_model import *
```

Include the product_model.py module in this program.

```python
result = db.engine.execute('select * from products')
for i in result:
    print(i)
result = db.engine.execute('select * from products')
final_result = [list(i) for i in result]
dataset=[]
dict={}
for i in final_result:
    dict['productID'] = i[0]
    dict['productCode'] = i[1]
    dict['name'] = i[2]
    dict['quantity'] = i[3]
    dict['price'] = i[4]
    dict['supplierID'] = i[5]
    # print(i)
    # print(dict)
    dataset.append(dict.copy()) #markers.append(fld.copy())
print(dataset)
```

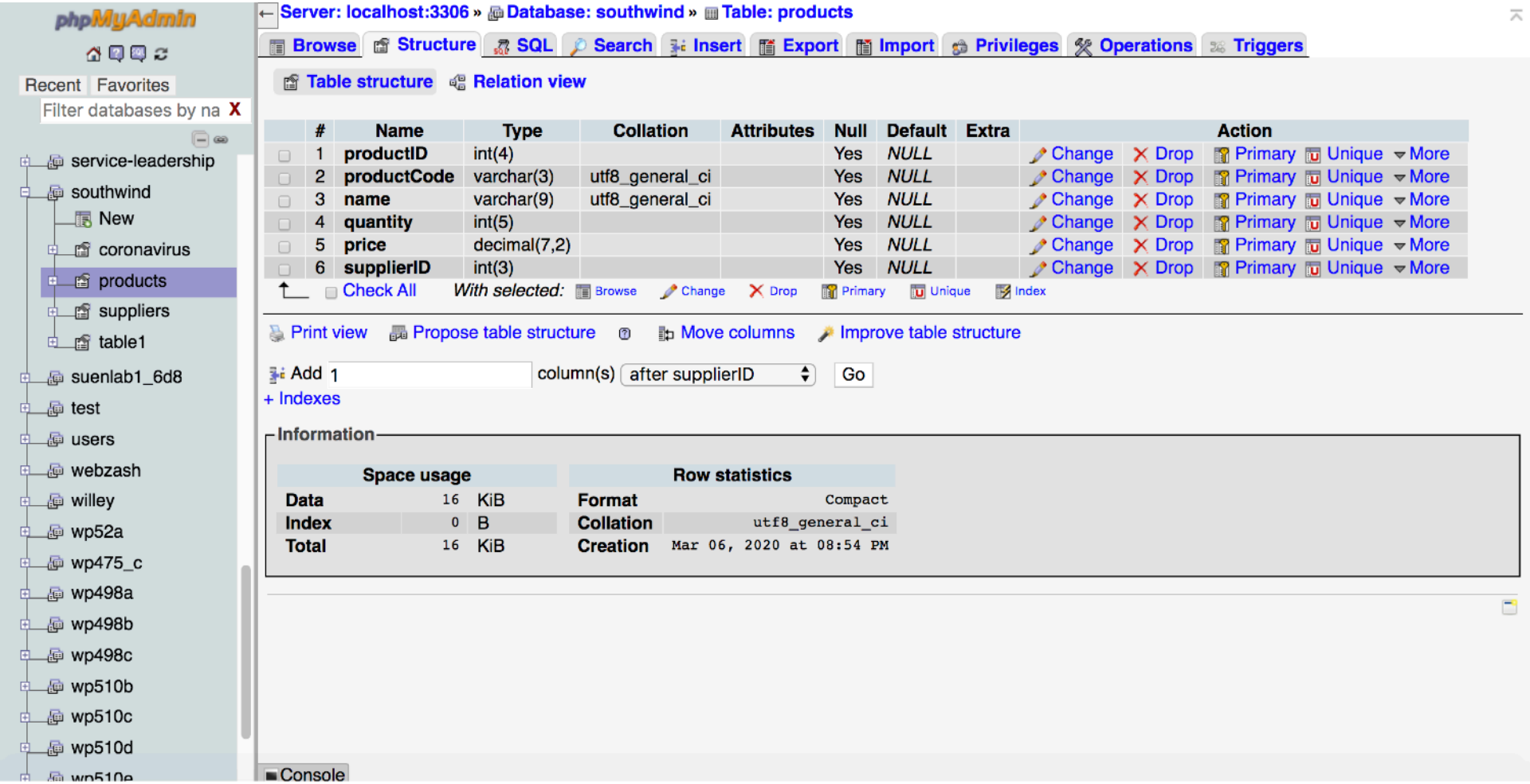
But we can still use SQL if we want. SQLAlchemy does support SQL statements.

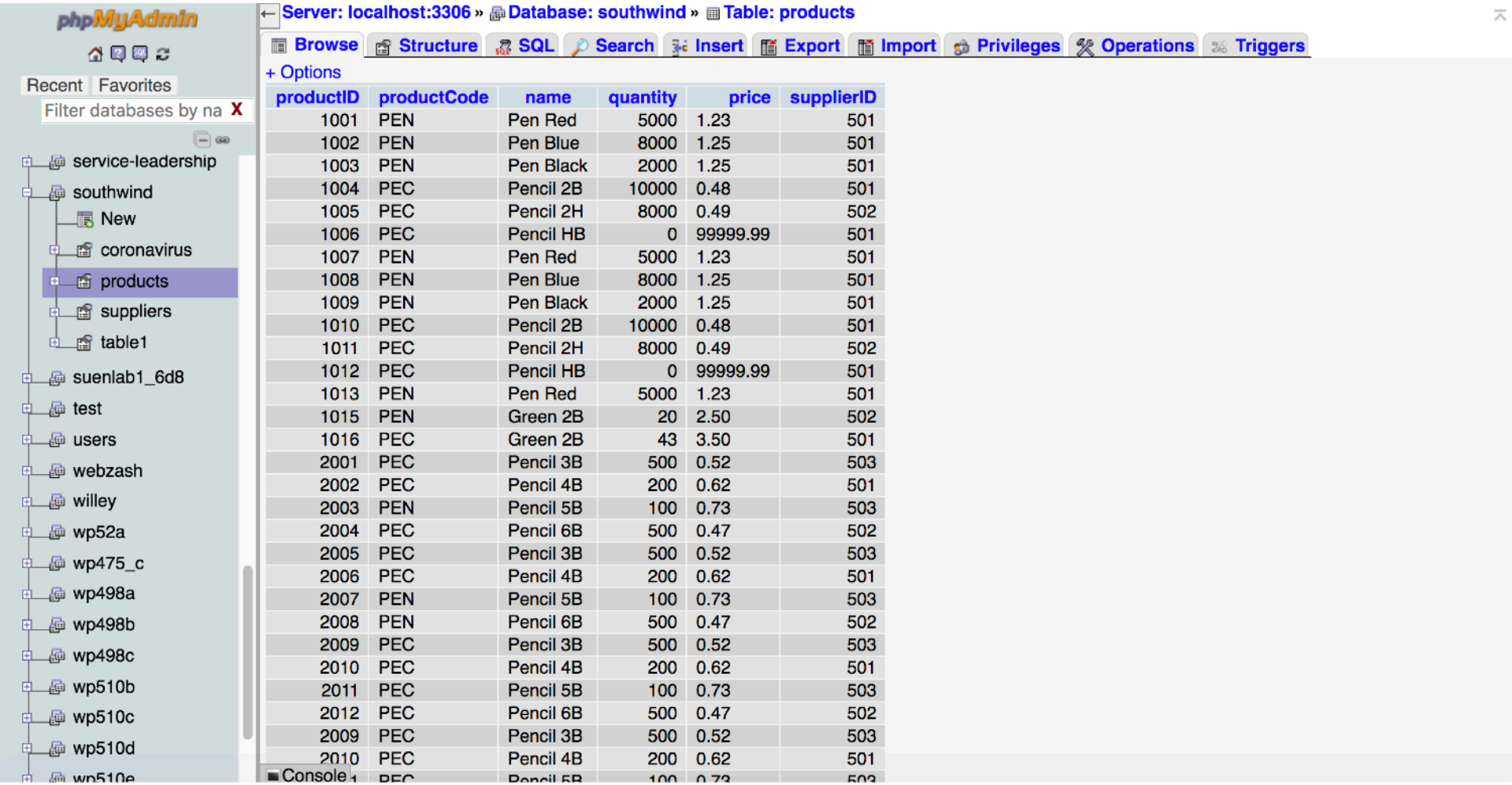
```python
# Update Example
pid = 1003
products = Products.query.filter_by(productID=pid).first()
products.productCode = "SAT"
db.session.commit()
```

```python
# Insert Example
product = Products(productCode='EGG',name="Bernard",quantity=100,price=1.5,supplierID=502)
db.session.add(product)
db.session.commit()
```

```python
# Delete Example
pid = 2044
product = Products.query.filter_by(productID=pid).first()
db.session.delete(product)
db.session.commit()
```

In order to support paging, we need to ensure that our database has a primary key and support automatic increment of key ID.


phpMyAdmin
Recent Favorites
Filter databases by na
service-leadership
southwind
New
coronavirus
products
suppliers
table1
suenlab1_6d8
test
users
webzash
willey
wp52a
wp475_c
wp498a
wp498b
wp498c
wp510b
wp510c
wp510d
Server: localhost:3306 » Database: southwind » Table: products
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Table structure Relation view
Name Type Collation Attributes Null Default Extra Action
1 productID int(4) Yes NULL Change Drop Primary Unique More
2 productCode varchar(3) utf8_general_ci Yes NULL Change Drop Primary Unique More
3 name varchar(9) utf8_general_ci Yes NULL Change Drop Primary Unique More
4 quantity int(5) Yes NULL Change Drop Primary Unique More
5 price decimal(7,2) Yes NULL Change Drop Primary Unique More
6 supplierID int(3) Yes NULL Change Drop Primary Unique More
Check All With selected: Browse Change Drop Primary Unique Index
Print view Propose table structure Move columns Improve table structure
Add 1 column(s) after supplierID Go
+ Indexes
Information
Space usage
Data 16 KiB
Index 0 B
Total 16 KiB
Row statistics
Format Compact
Collation utf8_general_ci
Creation Mar 06, 2020 at 08:54 PM
Console


phpMyAdmin
Recent
Favorites
Filter databases by na
X
service-leadership
southwind
New
coronavirus
products
suppliers
table1
suenlab1_6d8
test
users
webzash
willey
wp52a
wp475_c
wp498a
wp498b
wp498c
wp510b
wp510c
wp510d
Server: localhost:3306 » Database: southwind » Table: products
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
+ Options
productID | productCode | name | quantity | price | supplierID
1001 | PEN | Pen Red | 5000 | 1.23 | 501
1002 | PEN | Pen Blue | 8000 | 1.25 | 501
1003 | PEN | Pen Black | 2000 | 1.25 | 501
1004 | PEC | Pencil 2B | 10000 | 0.48 | 501
1005 | PEC | Pencil 2H | 8000 | 0.49 | 502
1006 | PEC | Pencil HB | 0 | 99999.99 | 501
1007 | PEN | Pen Red | 5000 | 1.23 | 501
1008 | PEN | Pen Blue | 8000 | 1.25 | 501
1009 | PEN | Pen Black | 2000 | 1.25 | 501
1010 | PEC | Pencil 2B | 10000 | 0.48 | 501
1011 | PEC | Pencil 2H | 8000 | 0.49 | 502
1012 | PEC | Pencil HB | 0 | 99999.99 | 501
1013 | PEN | Pen Red | 5000 | 1.23 | 501
1015 | PEN | Green 2B | 20 | 2.50 | 502
1016 | PEC | Green 2B | 43 | 3.50 | 501
2001 | PEC | Pencil 3B | 500 | 0.52 | 503
2002 | PEC | Pencil 4B | 200 | 0.62 | 501
2003 | PEN | Pencil 5B | 100 | 0.73 | 503
2004 | PEC | Pencil 6B | 500 | 0.47 | 502
2005 | PEC | Pencil 3B | 500 | 0.52 | 503
2006 | PEC | Pencil 4B | 200 | 0.62 | 501
2007 | PEN | Pencil 5B | 100 | 0.73 | 503
2008 | PEN | Pencil 6B | 500 | 0.47 | 502
2009 | PEC | Pencil 3B | 500 | 0.52 | 503
2010 | PEC | Pencil 4B | 200 | 0.62 | 501
2011 | PEC | Pencil 5B | 100 | 0.73 | 503
2012 | PEC | Pencil 6B | 500 | 0.47 | 502
2009 | PEC | Pencil 3B | 500 | 0.52 | 503
2010 | PEC | Pencil 4B | 200 | 0.62 | 501
Console

Paging through the Database Using Built-in Parameters

```
1  {% extends "base.html" %}
2  {% block content %}
3
4    {% for product in products.items %}
5         {% if loop.index == 1 %}{% set counter = ( counter | default(1) ) %}{% endif %}
6          {% if counter == 1 %}<div class="row">{% endif %}
7          <div class="col-md-4">
8            <div class="thumbnail">
9                <p>Col#:{{ loop.index }}<br>Product Code:{{ product.productID }}<br>Quantity:{{ product.quantity }}<br>Price:{{
   product.price }}</p>
10                 <div class="btn-group">
11                   <button type="button" class="btn btn-sm btn-outline-secondary">View</button>
12                 </div>
13           </div>
14         </div>
15         {% if counter == 3 %}</div><!--end row-->{%  set counter = ( counter | default(1) ) %}{% else %}{% set counter = ( counter
   | default(0) ) + 1 %}{% endif %}
16   {% endfor %}
17   <div class="container" style="float: left;">
18   {% for page in products.iter_pages() %}
19     {% if page %}
20          <a href="{{url_for('album')}}/{{ page }}">{{ page }}</a>
21     {% else %}
22          ...
23     {% endif %}
24   {% endfor %}
25   </div>
26 {% endblock %}
```

Two loops are involved here. The first loop goes through the rows and columns of the albums.

The second loop controls the paging of the album.

Class Demo | Home | Album | Dashboard | Products | Logout | Login

Col#:1
Product ID:1001
Product Code:PEN
Product NAME:Pen Red
Quantity:5000
Price:1.23
Supplier ID:501
View

Col#:2
Product ID:1002
Product Code:PEN
Product NAME:Pen Blue
Quantity:8000
Price:1.25
Supplier ID:503
View

Col#:3
Product ID:1003
Product Code:SAT
Product NAME:Pen God
Quantity:2000
Price:1.25
Supplier ID:501
View

Col#:4
Product ID:1004
Product Code:XYZ
Product NAME:Pen Pen
Quantity:10000
Price:9.75
Supplier ID:501
View

Col#:5
Product ID:1006
Product Code:PEC
Product NAME:Pencil HB
Quantity:222
Price:123.23
Supplier ID:501
View

Col#:6
Product ID:1007
Product Code:PEN
Product NAME:Pen Red
Quantity:5000
Price:1.23
Supplier ID:501
View

Col#:7
Product ID:1008
Product Code:PEN
Product NAME:Pen Blue
Quantity:8000
Price:1.25
Supplier ID:501
View

Col#:8
Product ID:1009
Product Code:PEN
Product NAME:Pen Black
Quantity:2000
Price:1.25
Supplier ID:501
View

Col#:9
Product ID:1010
Product Code:PEC
Product NAME:Pencil 2B
Quantity:10000
Price:0.48
Supplier ID:501
View

1 2 3 4 5 6

Introducing Flask SQLAlchemy

CRUD Support in SQLAlchemy and Airtable for View Integration

Unit Testing
Integration Testing
User Acceptance Testing

CRUD Operations in Action

Class Demo | Home | Album | Dashboard | Products | Logout | Login

Update Product | Delete Product | Add button

Search:

Product ID	Product Code	Name	Quantity	Price	Supplier ID
1001	PEN	Pen Red	5000	1.23	501
1002	PEN	Pen Blue	8000	1.25	503
1003	SAT	Pen God	2000	1.25	501
1004	XYZ	Pen Pen	10000	9.75	501
1006	PEC	Pencil HB	222	123.23	501
1007	PEN	Pen Red	5000	1.23	501
1008	PEN	Pen Blue	8000	1.25	501
1009	PEN	Pen Black	2000	1.25	501
1010	PEC	Pencil 2B	10000	0.48	501
1011	PEC	Pencil 2H	8000	0.49	502

Showing 1 to 10 of 50 entries

Previous | 1 | 2 | 3 | 4 | 5 | Next

Class Demo | Home | Album | Dashboard | Products | Logout | Login

Update Product | Delete Product | Add button

Search:

Product ID	Product Code	Name	Quantity	Price	Supplier ID
1001	PEN	Pen Red	5000	1.23	501
1002	PEN	Pen Blue	8000	1.25	503
1003	SAT	Pen God	2000	1.25	501
1004	XYZ	Pen Pen	10000	9.75	501
1006	PEC	Pencil HB	222	123.23	501
1007	PEN	Pen Red	5000	1.23	501
1008	PEN	Pen Blue	8000	1.25	501
1009	PEN	Pen Black	2000	1.25	501
1010	PEC	Pencil 2B	10000	0.48	501
1011	PEC	Pencil 2H	8000	0.49	502

Showing 1 to 10 of 50 entries 1 row selected

Previous | 1 | 2 | 3 | 4 | 5 | Next

Class Demo | Home | Album | Dashboard | Products | Logout | Login

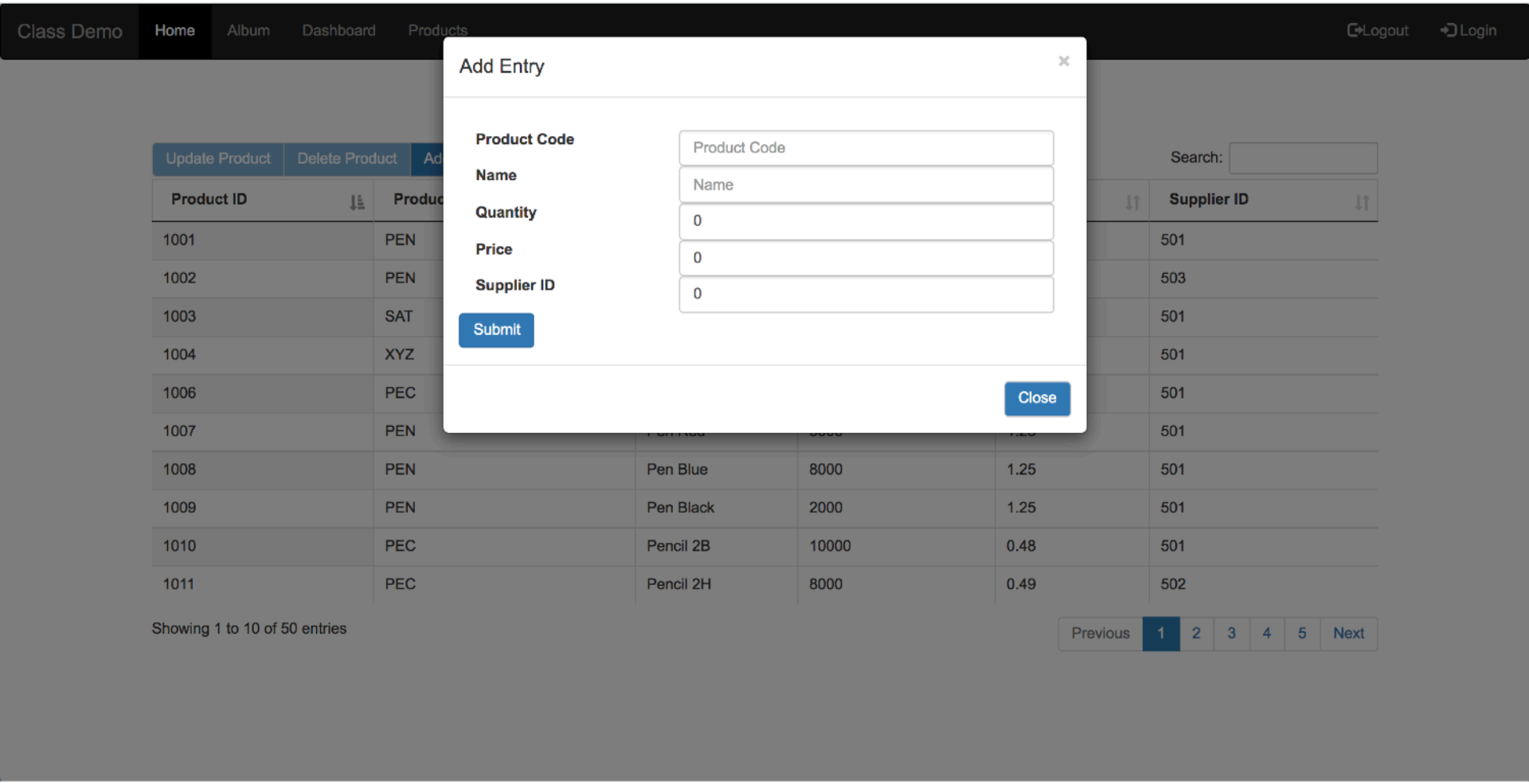
Update Product | Delete Product | Add button

Search:

Product ID	Product Code	Name	Quantity	Price	Supplier ID
1001	PEN	Pen Red	5000	1.23	501
1002	PEN	Pen Blue	8000	1.25	503
1003	SAT	Pen God	2000	1.25	501
1004	XYZ	Pen Pen	10000	9.75	501
1006	PEC	Pencil HB	222	123.23	501
1007	PEN	Pen Red	5000	1.23	501
1008	PEN	Pen Blue	8000	1.25	501
1009	PEN	Pen Black	2000	1.25	501
1010	PEC	Pencil 2B	10000	0.48	501
1011	PEC	Pencil 2H	8000	0.49	502

Showing 1 to 10 of 50 entries

Previous | 1 | 2 | 3 | 4 | 5 | Next


Class Demo
Home
Album
Dashboard
Products
Logout
Login
Add Entry
Product Code
Name
Quantity
Price
Supplier ID
Submit
Close
Update Product
Delete Product
Search:
Product ID
Supplier ID
1001 PEN 501
1002 PEN 503
1003 SAT 501
1004 XYZ 501
1006 PEC 501
1007 PEN 501
1008 PEN Pen Blue 8000 1.25 501
1009 PEN Pen Black 2000 1.25 501
1010 PEC Pencil 2B 10000 0.48 501
1011 PEC Pencil 2H 8000 0.49 502
Showing 1 to 10 of 50 entries
Previous
1
2
3
4
5
Next

Class Demo | Home | Album | Dashboard | Products | Logout | Login

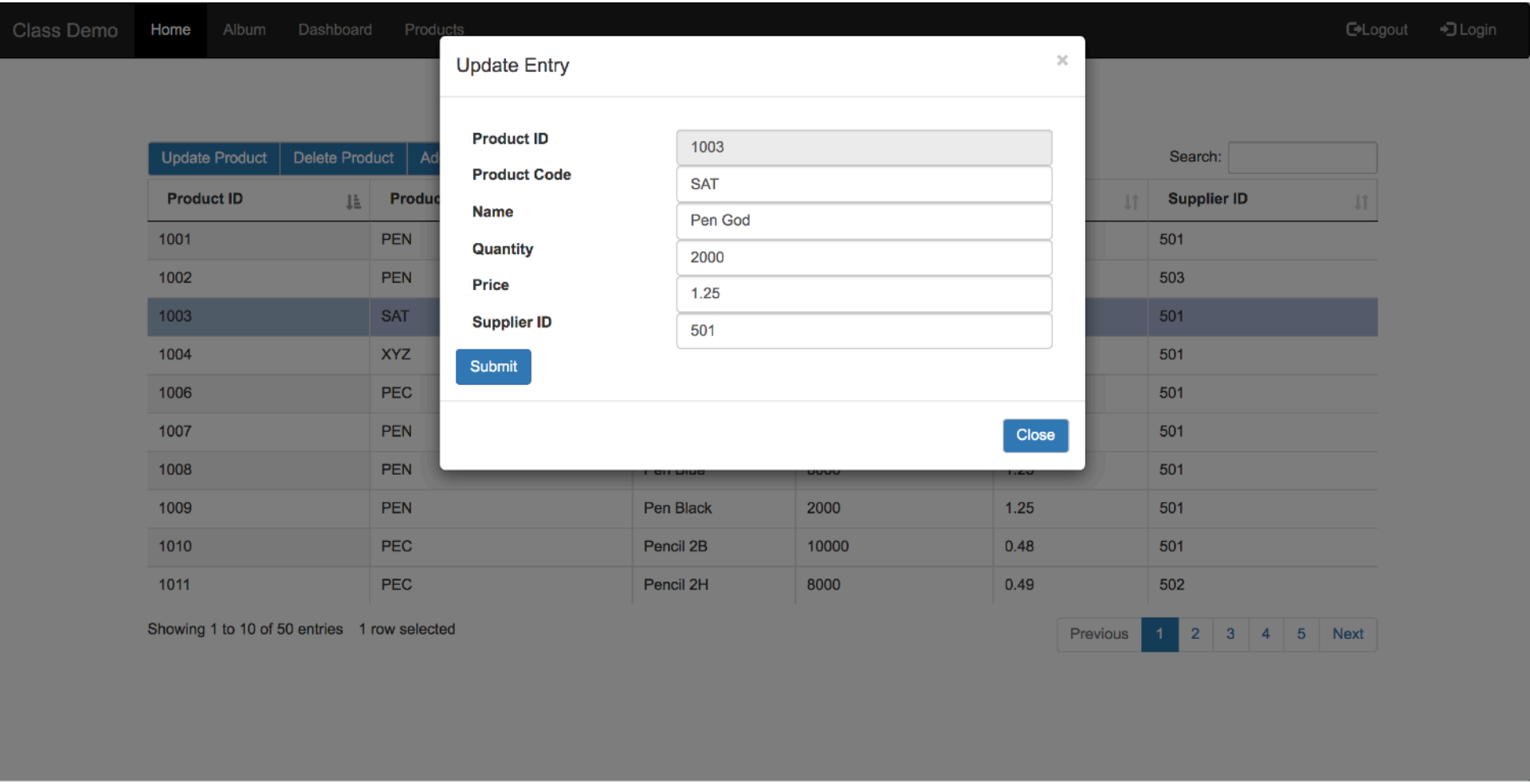
Update Product | Delete Product | Add button

Search:

Product ID	Product Code	Name	Quantity	Price	Supplier ID
1001	PEN	Pen Red	5000	1.23	501
1002	PEN	Pen Blue	8000	1.25	503
1003	SAT	Pen God	2000	1.25	501
1004	XYZ	Pen Pen	10000	9.75	501
1006	PEC	Pencil HB	222	123.23	501
1007	PEN	Pen Red	5000	1.23	501
1008	PEN	Pen Blue	8000	1.25	501
1009	PEN	Pen Black	2000	1.25	501
1010	PEC	Pencil 2B	10000	0.48	501
1011	PEC	Pencil 2H	8000	0.49	502

Showing 1 to 10 of 50 entries 1 row selected

Previous | 1 | 2 | 3 | 4 | 5 | Next


Class Demo
Home
Album
Dashboard
Products
Logout
Login
Update Entry
Product ID
1003
Product Code
SAT
Name
Pen God
Quantity
2000
Price
1.25
Supplier ID
501
Submit
Close
Update Product
Delete Product
Search:
Product ID
Supplier ID
1001 PEN 501
1002 PEN 503
1003 SAT 501
1004 XYZ 501
1006 PEC 501
1007 PEN 501
1008 PEN 501
1009 PEN Pen Black 2000 1.25 501
1010 PEC Pencil 2B 10000 0.48 501
1011 PEC Pencil 2H 8000 0.49 502
Showing 1 to 10 of 50 entries 1 row selected
Previous 1 2 3 4 5 Next

Class Demo | Home | Album | Dashboard | Products | Logout | Login

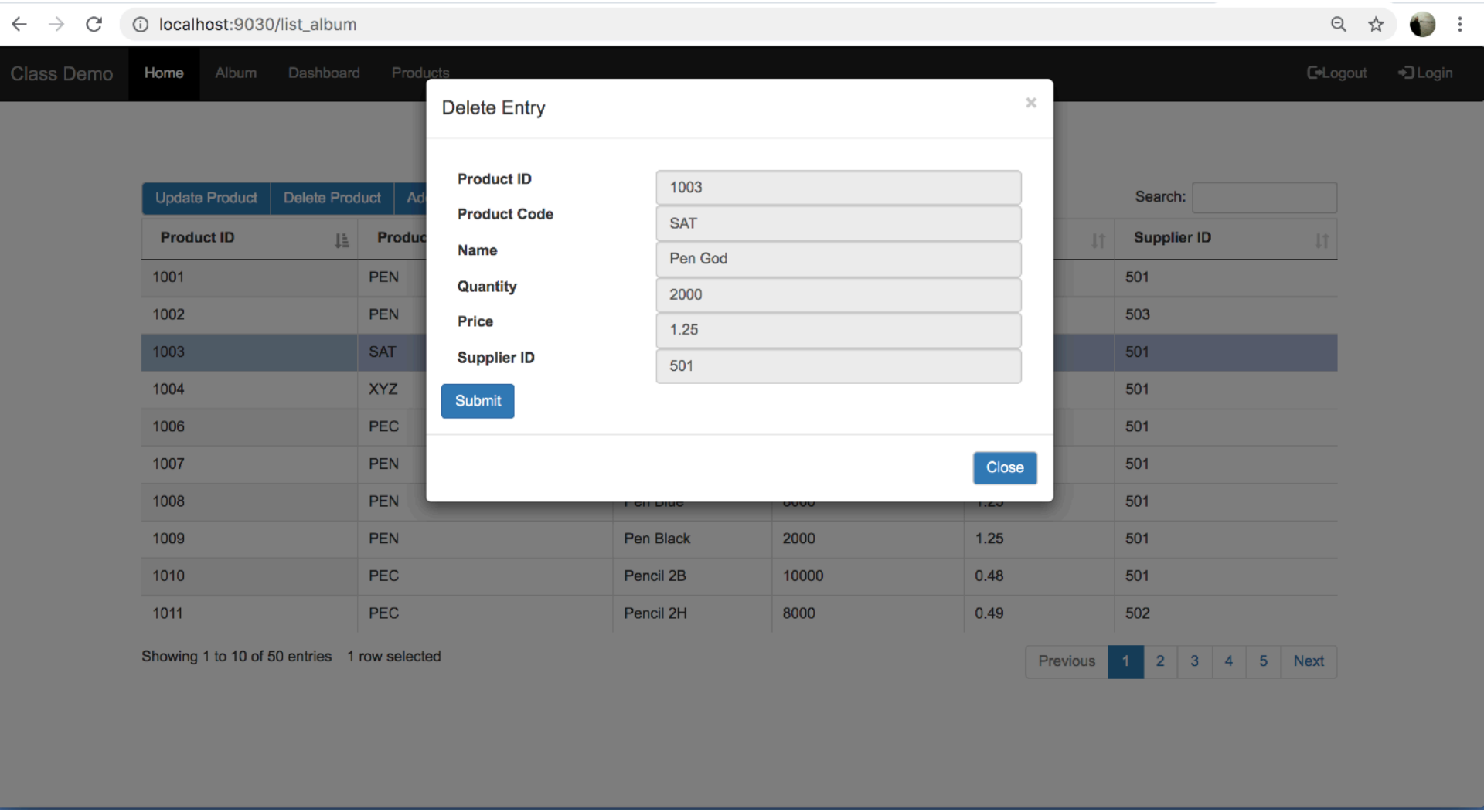
Update Product | Delete Product | Add button

Search:

Product ID	Product Code	Name	Quantity	Price	Supplier ID
1001	PEN	Pen Red	5000	1.23	501
1002	PEN	Pen Blue	8000	1.25	503
1003	SAT	Pen God	2000	1.25	501
1004	XYZ	Pen Pen	10000	9.75	501
1006	PEC	Pencil HB	222	123.23	501
1007	PEN	Pen Red	5000	1.23	501
1008	PEN	Pen Blue	8000	1.25	501
1009	PEN	Pen Black	2000	1.25	501
1010	PEC	Pencil 2B	10000	0.48	501
1011	PEC	Pencil 2H	8000	0.49	502

Showing 1 to 10 of 50 entries 1 row selected

Previous | 1 | 2 | 3 | 4 | 5 | Next


localhost:9030/list_album
Class Demo
Home
Album
Dashboard
Products
Logout
Login
Delete Entry
Product ID
1003
Product Code
SAT
Name
Pen God
Quantity
2000
Price
1.25
Supplier ID
501
Submit
Close
Update Product
Delete Product
Search:
Product ID
Supplier ID
1001 PEN 501
1002 PEN 503
1003 SAT 501
1004 XYZ 501
1006 PEC 501
1007 PEN 501
1008 PEN 501
1009 PEN Pen Black 2000 1.25 501
1010 PEC Pencil 2B 10000 0.48 501
1011 PEC Pencil 2H 8000 0.49 502
Showing 1 to 10 of 50 entries 1 row selected
Previous 1 2 3 4 5 Next

```python
from flask import Flask, render_template, url_for, redirect, request, make_response, Response, jsonify
from flask_login import LoginManager, login_user, current_user, logout_user, login_required, UserMixin
from flask_cors import CORS
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
import requests
import jwt

############ Initialize Flask App #############

app = Flask(__name__)
CORS(app)

#### MySQL SQLAlchemy Object Relations Mapping #####

app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:root@localhost/southwind'
app.config['SECRET_KEY'] = "mysecret"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class Products(db.Model):
    __tablename__ = 'products'
    productID = db.Column(db.Integer, primary_key=True)
    productCode = db.Column(db.String(3))
    name = db.Column(db.String(30))
    quantity = db.Column(db.Integer)
    price = db.Column(db.Float)
    supplierID = db.Column(db.Integer)
    # description = db.Column(db.Text)

############## Login Manager Setup ##############

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'
app.config['SECRET_KEY'] = "lkkajdghdadkglajkgah"


@login_manager.user_loader
def load_user(user_id):
    return User(user_id)

class User(UserMixin):
  def __init__(self,id):
    self.id = id
```

Similar to how we setup MySQL database using py mysql and MySQLdb, we use the following codes to set up Flask SQLAlchemy to connect to the MySQL database.

Since SQLAlchemy is a object-based (i.e. class) database mapper, it uses object notation to carry out its operations.

```python
@app.route("/list_album")
@login_required
def list_album():
    dataset = []
    product_list = Products.query.all()
    for product in product_list:
        dataset.append({'productID': product.productID, 'productCode': product.productCode, 'name': product.name,
                        'quantity': product.quantity, 'price':product.price,'supplierID': product.supplierID})
    return render_template('list_album2.html', entries=dataset)

@app.route('/album')
def album():
    page_num = 1
    products = Products.query.paginate(per_page=9, page=page_num, error_out=True)
    return render_template('product_paging.html', products=products)

@app.route('/album/<int:page_num>')
def album_paging(page_num):
    products = Products.query.paginate(per_page=9, page=page_num, error_out=True)
    return render_template('product_paging.html', products=products)

@app.route("/add_product",methods=['POST'])
@login_required
def add_product():
    productCode = request.form['product_code']
    name = request.form['name']
    quantity = request.form['quantity']
    price = float(request.form['price'])
    supplierID = request.form['supplier_id']
    product = Products(productCode=productCode,name=name,quantity=quantity,price=price,supplierID=supplierID)
    db.session.add(product)
    db.session.commit()
    dataset = []
    product_list = Products.query.all()
    for product in product_list:
        dataset.append({'productID': product.productID, 'productCode': product.productCode, 'name': product.name,
                        'quantity': product.quantity, 'price':product.price,'supplierID': product.supplierID})
    return render_template('list_album2.html', entries=dataset)
```

Similar to the SQL “select * from products”

Passing how many entries per page (e.g. 9 entries).

Add operation

```python
@app.route("/update_product",methods=['POST','PUT'])
@login_required
def update_product():
    record_id = request.form['record_id']
    product = Products.query.filter_by(productID=record_id).first()
    product.productCode = request.form['product_code']
    product.name = request.form['name']
    product.quantity = request.form['quantity']
    product.price = float(request.form['price'])
    product.supplierID = request.form['supplier_id']
    db.session.commit()
    dataset = []
    product_list = Products.query.all()
    for product in product_list:
        dataset.append({'productID': product.productID, 'productCode': product.productCode, 'name': product.name,
                        'quantity': product.quantity, 'price':product.price,'supplierID': product.supplierID})
    return render_template('list_album2.html', entries=dataset)

@app.route("/delete_product",methods=['POST','DELETE'])
@login_required
def delete_product():
    #url = "https://api.airtable.com/v0/appM38HXlEVhxmnqx/flaskdemo/"
    record_id = request.form['record_id']
    product = Products.query.filter_by(productID=record_id).first()
    db.session.delete(product)
    db.session.commit()
    dataset = []
    product_list = Products.query.all()
    for product in product_list:
        dataset.append({'productID': product.productID, 'productCode': product.productCode, 'name': product.name,
                        'quantity': product.quantity, 'price':product.price,'supplierID': product.supplierID})
    return render_template('list_album2.html', entries=dataset)
```

Update operation

Delete operation

Airtable REST API CRUD Operations

In [33]:

```
#
# List Records
#
import requests
import pandas as pd

headers = {
        'Authorization': 'Bearer                ',
}

r = requests.get('https://api.airtable.com/v0/appM38HXlEVhxmnqx/flaskdemo?sortField=_createdTime&sortDirection=desc
print("Status Code:",r.status_code)
print(r.text)
dict = r.json()
a_list = dict['records']
df = pd.DataFrame.from_dict(a_list)
df.head()
```

```
Status Code: 200
{"records":[{"id":"recb1ZFR8ZL0HB9ti","fields":{"full_name":"Willam Lo","lname":"Lo","student_id":"5555","date_of_bir
th":"2002-02-21","record_id":"recb1ZFR8ZL0HB9ti","fname":"Willam"},"createdTime":"2019-09-21T02:11:26.000Z"},{"id":"r
eczlouWve6Oh1JPv","fields":{"full_name":"Winifred Wong","lname":"Wong","student_id":"010585","date_of_birth":"1958-01
-28","record_id":"reczlouWve6Oh1JPv","fname":"Winifred"},"createdTime":"2019-09-20T16:37:52.000Z"},{"id":"rect6erpDSP
Nty1TN","fields":{"full_name":"Bernard Suen","lname":"Suen","student_id":"11111","date_of_birth":"1958-05-23","record
_id":"rect6erpDSPNty1TN","fname":"Bernard"},"createdTime":"2019-09-19T15:18:41.000Z"},{"id":"rec8vF2VbIgOTdx3F","fiel
ds":{"full_name":"Mary Lu","lname":"Lu","student_id":"99999","date_of_birth":"2001-08-22","record_id":"rec8vF2VbIgOTd
x3F","fname":"Mary"},"createdTime":"2019-09-19T14:32:08.000Z"},{"id":"recASxs4aPaHLeJj3","fields":{"full_name":"Emily
Chan","lname":"Chan","student_id":"67891","date_of_birth":"2008-04-03","record_id":"recASxs4aPaHLeJj3","fname":"Emil
y"},"createdTime":"2019-09-19T14:04:24.000Z"}]}
```

Out[33]:

	createdTime	fields	id
0	2019-09-21T02:11:26.000Z	{'full_name': 'Willam Lo', 'lname': 'Lo', 'stu...	recb1ZFR8ZL0HB9ti
1	2019-09-20T16:37:52.000Z	{'full_name': 'Winifred Wong', 'lname': 'Wong'...	reczlouWve6Oh1JPv
2	2019-09-19T15:18:41.000Z	{'full_name': 'Bernard Suen', 'lname': 'Suen',...	rect6erpDSPNty1TN
3	2019-09-19T14:32:08.000Z	{'full_name': 'Mary Lu', 'lname': 'Lu', 'stude...	rec8vF2VbIgOTdx3F
4	2019-09-19T14:04:24.000Z	{'full_name': 'Emily Chan', 'lname': 'Chan', '...	recASxs4aPaHLeJj3

In [38]:

```
#
# Add Record
#

import requests
import pandas as pd

data = {
  "records": [
    {
      "fields": {
        "fname": "Emily",
        "lname": "Lau",
        "student_id": "686868",
        "date_of_birth": "2002-02-21"
      }
    }
  ]
}
headers = {'Authorization': 'Bearer _____', 'Content-Type': 'application/json; charset=utf-8'}
r = requests.post('https://api.airtable.com/v0/appM38HX1EVhxmnqx/flaskdemo',json=data,headers=headers)
print("Status Code:",r.status_code)
print(r.text)
dict = r.json()
a_list = dict['records']
df = pd.DataFrame.from_dict(a_list)
df.head()
```

```
Status Code: 200
{"records":[{"id":"recggGoKr4zQhqj3l","fields":{"lname":"Lau","student_id":"686868","date_of_birth":"2002-02-21","fna
me":"Emily","full_name":"Emily Lau","record_id":"recggGoKr4zQhqj3l"},"createdTime":"2019-09-21T02:53:23.000Z"}]}
```

Out[38]:

	createdTime	fields	id
0	2019-09-21T02:53:23.000Z	{'lname': 'Lau', 'student_id': '686868', 'date...	recggGoKr4zQhqj3l

In [40]:

```
#
# Update Records
#
import requests
import pandas as pd

data = {
  "records": [
    {
      "id": "recggGoKr4zQhqj3l",
      "fields": {
        "fname": "Emily",
        "lname": "Chan",
        "student_id": "67891",
        "date_of_birth": "2008-04-03"
      }
    }
  ]
}
headers = {'Authorization': 'Bearer                    ', 'Content-Type': 'application/json; charset=utf-8'}
r = requests.put('https://api.airtable.com/v0/appM38HXlEVhxmnqx/flaskdemo',json=data,headers=headers)
print("Status Code:",r.status_code)
print(r.text)
dict = r.json()
a_list = dict['records']
df = pd.DataFrame.from_dict(a_list)
df.head()
```

```
Status Code: 200
{"records":[{"id":"recggGoKr4zQhqj3l","fields":{"lname":"Chan","student_id":"67891","date_of_birth":"2008-04-03","fna
me":"Emily","full_name":"Emily Chan","record_id":"recggGoKr4zQhqj3l"},"createdTime":"2019-09-21T02:53:23.000Z"}]}
```

Out[40]:

	createdTime	fields	id
0	2019-09-21T02:53:23.000Z	{'lname': 'Chan', 'student_id': '67891', 'date...	recggGoKr4zQhqj3l

In [41]:

```
#
# Delete Record
#
import requests
import pandas as pd

headers = {'Authorization': 'Bearer                    ', 'Content-Type': 'application/x-www-form-urlencoded'}
r = requests.delete("https://api.airtable.com/v0/appM38HXlEVhxmnqx/flaskdemo/recggGoKr4zQhqj3l",headers=headers)
print("Status Code:",r.status_code)
print(r.text)

headers = {
        'Authorization': 'Bearer                    ',
}
r = requests.get('https://api.airtable.com/v0/appM38HXlEVhxmnqx/flaskdemo?
                 sortField =_createdTime&sortDirection=desc', headers=headers)
dict = r.json()
a_list = dict['records']
df = pd.DataFrame.from_dict(a_list)
df.head()
```

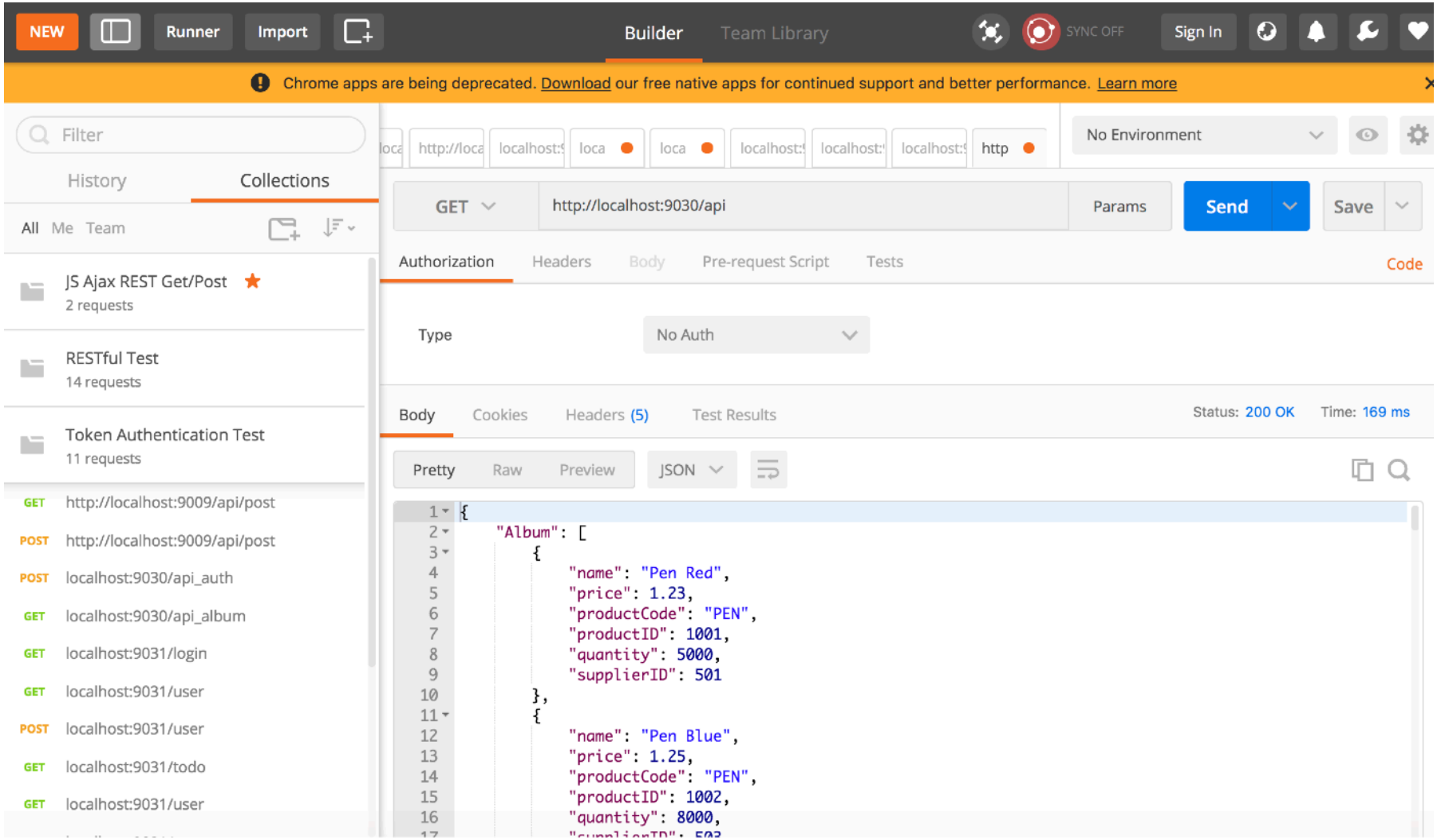
```
Status Code: 200
{"deleted":true,"id":"recggGoKr4zQhqj3l"}
```

Out[41]:

	createdTime	fields	id
0	2019-09-21T02:11:26.000Z	{'full_name': 'Willam Lo', 'lname': 'Lo', 'stu...	recb1ZFR8ZL0HB9ti
1	2019-09-20T16:37:52.000Z	{'full_name': 'Winifred Wong', 'lname': 'Wong'...	reczlouWve6Oh1JPv
2	2019-09-19T15:18:41.000Z	{'full_name': 'Bernard Suen', 'lname': 'Suen',...	rect6erpDSPNty1TN
3	2019-09-19T14:32:08.000Z	{'full_name': 'Mary Lu', 'lname': 'Lu', 'stude...	rec8vF2VbIgOTdx3F

API End-points for Front-end Access Using JSONIFY

```python
@app.route('/api')
def api():
    result = db.engine.execute('select * from products')
    final_result = [list(i) for i in result]
    dataset=[]
    dict={}
    for i in final_result:
        dict['productID'] = i[0]
        dict['productCode'] = i[1]
        dict['name'] = i[2]
        dict['quantity'] = i[3]
        dict['price'] = i[4]
        dict['supplierID'] = i[5]
        # print(i)
        # print(dict)
        dataset.append(dict.copy()) #markers.append(fld.copy())
    return jsonify({'Album': dataset})
```


NEW
Runner
Import
Builder
Team Library
SYNC OFF
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
JS Ajax REST Get/Post
2 requests
RESTful Test
14 requests
Token Authentication Test
11 requests
GET http://localhost:9009/api/post
POST http://localhost:9009/api/post
POST localhost:9030/api_auth
GET localhost:9030/api_album
GET localhost:9031/login
GET localhost:9031/user
POST localhost:9031/user
GET localhost:9031/todo
GET localhost:9031/user
No Environment
GET
http://localhost:9030/api
Params
Send
Save
Authorization
Headers
Body
Pre-request Script
Tests
Code
Type
No Auth
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 169 ms
Pretty
Raw
Preview
JSON
{
"Album": [
{
"name": "Pen Red",
"price": 1.23,
"productCode": "PEN",
"productID": 1001,
"quantity": 5000,
"supplierID": 501
},
{
"name": "Pen Blue",
"price": 1.25,
"productCode": "PEN",
"productID": 1002,
"quantity": 8000,

```
ocument).ready(function(){
 var counter = 1;
 var at_getapi = "https://api.airtable.com/v0/appKIU0zkdHt3AVTL/Venues?api_key=keycj6dRwXwYLEjiv";
 $.getJSON(at_getapi, function(result) {
     $.each(result.records, function(key,value) {
             var name = value.fields.Name;
             var image = value.fields.img_url;
             var org_desc = value.fields.desc;
             var desc = org_desc.substring(0, 300);
             var n = desc.lastIndexOf(" ");
             var desc = desc.substring(0, n)+"...";
             // alert(value.fields.Name);
             if (counter == 1) {$("#album_items").append(`<div class="row">`)};
             $("#album_items").append(`<div class="col-md-4">
         <div class="thumbnail">
             <img height="auto" width="100%" src="${image}"
             class="img-responsive" style="max-height:200px;">
             <div class="caption">
                 <h3>${name}</h3>
                 <p><!-- ${desc} --></p>
                 <button class="btn btn-lg btn-primary" onclick="modal_handle('${name}','${org_des
                  data-id="107" data-body="testing...">View</button>
             </div>
         </div>
 </div>`);
```

```
<link href="css/font-awesome.min.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" href="css/style.css">
<script>
        function modal_handle(name, org_desc, image) {
                    $("h3.modal-title").html(`${name}`);
                    $(".modal-body").html(`<img src="${image}"
                        class="img-responsive"><p>${org_desc}</p>`);
                    $('#popup-photo').modal('show');
        }

        function topFunction() {
            $('html, body').animate({scrollTop : 0},800);
            return false;
        }
```

```
$.getJSON(at_getapi, function(result) {
    $.each(result.Album, function(key,value) {
            var productID = value.productID;
            var productCode = value.productCode;
            var name = value.name;
            // var quantity = value.quantity;
            // var price = value.price;
            // var supplierID = value.supplierID;
            // alert(value.fields.Name);
            if (counter == 1) {$("#album_items").append(`<div class="row">`)};
            $("#album_items").append(`<div class="col-md-4">
        <div class="thumbnail">
            <div class="caption">
                <h3>${productID}</h3>
                <p><!-- ${productCode} --></p>
                <button class="btn btn-lg btn-primary" onclick="modal_handle('${productID}','${product
                 data-id="107" data-body="testing...">View</button>
            </div>
        </div>
    </div>`);
            counter = counter +1; // counte
```

```
<script>
        function modal_handle(productID, productCode, name) {
                    $("h3.modal-title").html(`${productID}`);
                    $(".modal-body").html(`<p>"${productCode}</p>
                        <p>${name}</p>`);
                    $('#popup-photo').modal('show');
        }

        function topFunction() {
            $('html, body').animate({scrollTop : 0},800);
            return false;
        }

</script>
```

HKU

View

HK Science Musuem

View

Victoria Peak

View

^

Home Album

1001
View

1002
View

1003
View

1004
View

1006
View

1007
View

1008
View

1009
View

1010
View

1011
View

1012
View

1013
View

^

Introducing Flask SQLAlchemy

CRUD Support in SQLAlchemy and Airtable for View Integration

Unit Testing
Integration Testing
User Acceptance Testing

Unit Testing, Integration Testing, and User Acceptance Testing (UAT)

敏捷开发中用户故事地图(User Story Mapping)学习

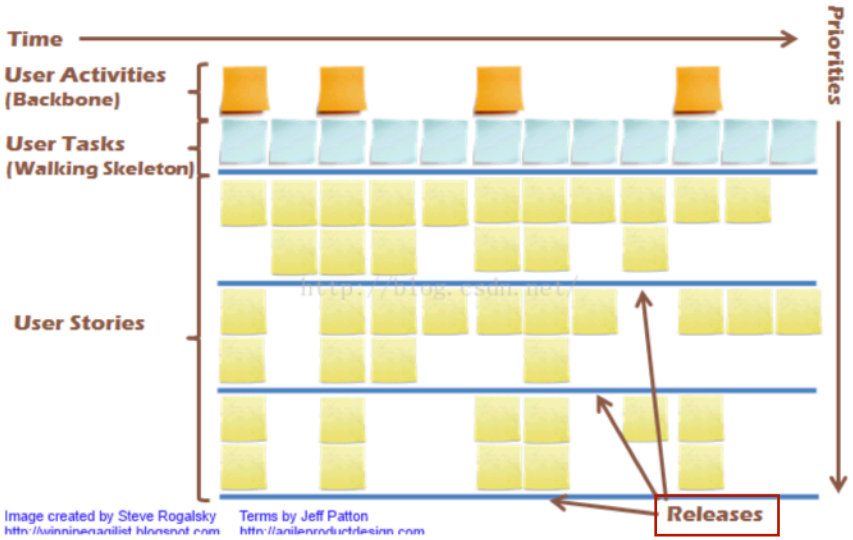
2016年09月25日 20:07:11 _佚名-2018_ 阅读数：3485

1.用户故事地图

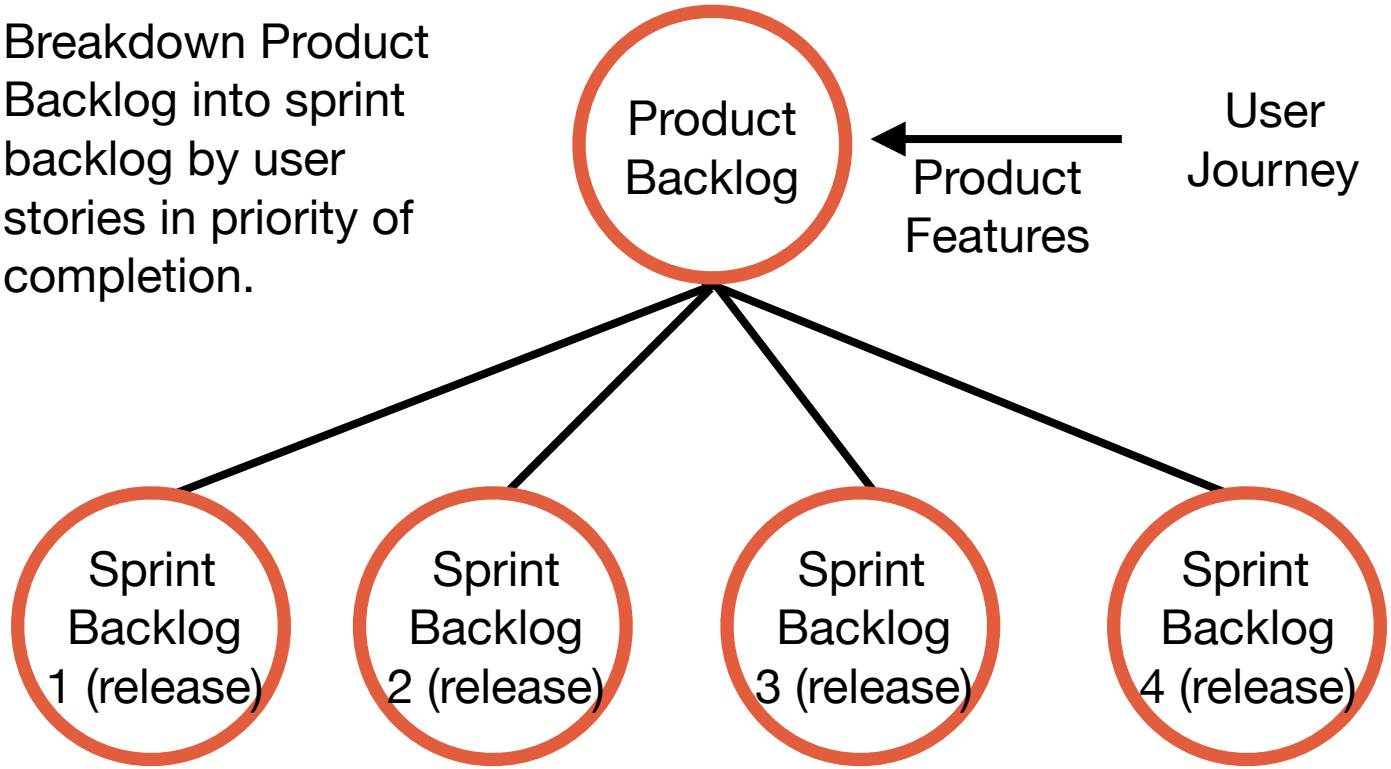
User Story是敏捷开发和管理的核心，User Story是从用户的角度对系统的某个功能模块所作的简短描述。一个User Story描述了项目中的一个小功能，以及这个功能完成之后将会产生什么效果，或者说能为客户创造什么价值。

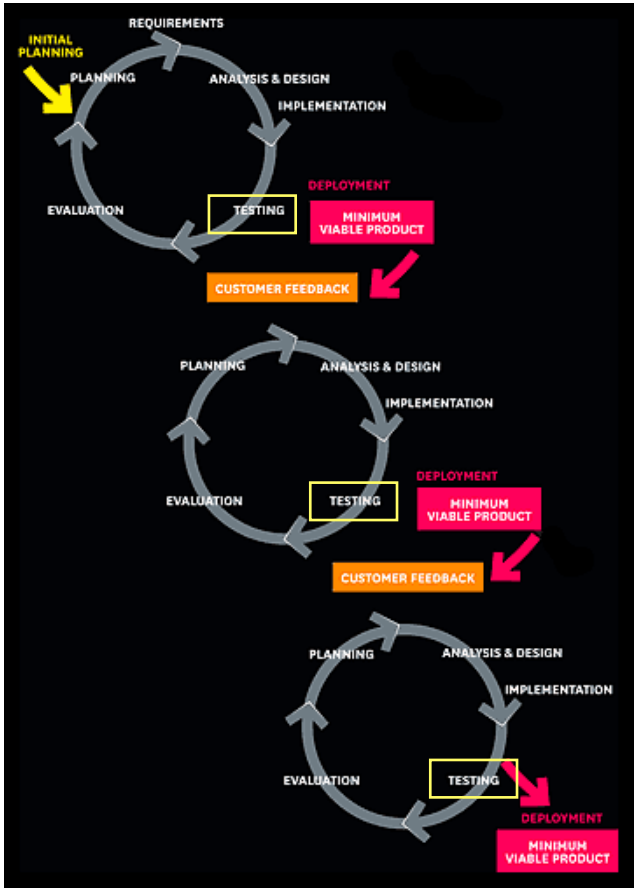
用户故事地图，英文名为User Story Mapping。用户故事地图已经成为敏捷需求规划中的一个流行方法。用户故事地图可以将你的backlog（待办事项）变成一张二维地图，而不是传统的简单列表。

2.用户故事地图编写模板



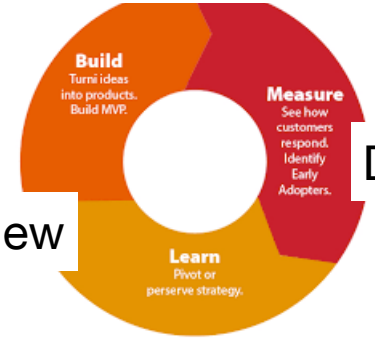
Breakdown Product Backlog into sprint backlog by user stories in priority of completion.





Source: “Why Lean Start-up Changes Everything” by Steve Blank

Sprint Planning (Release Planning)



Daily Scrum (Daily Meeting)

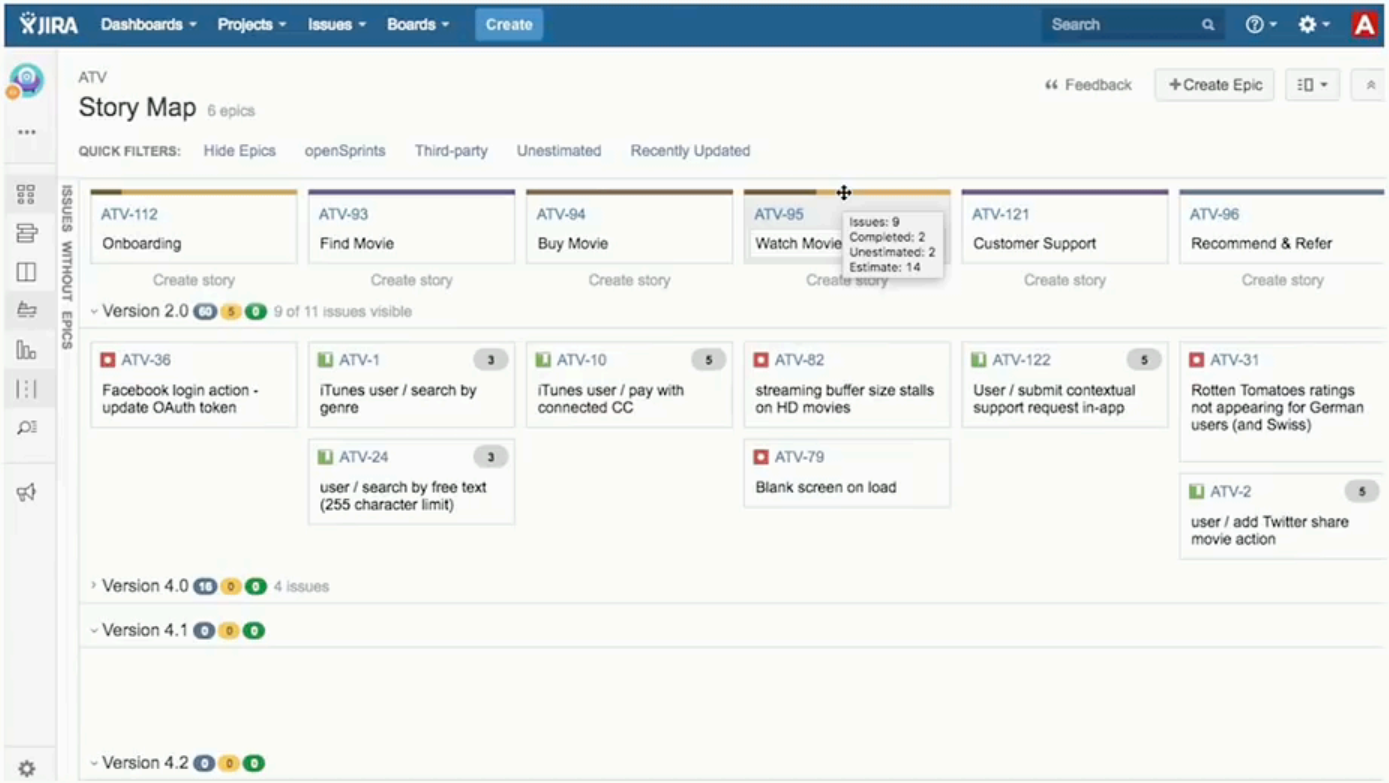
Sprint Review (Release Review)

Each build-measure-learn cycle (ends with a **review**) is called a “**sprint**” usually measured in 1 to 3 weeks.

Testing is an integral part of this process.

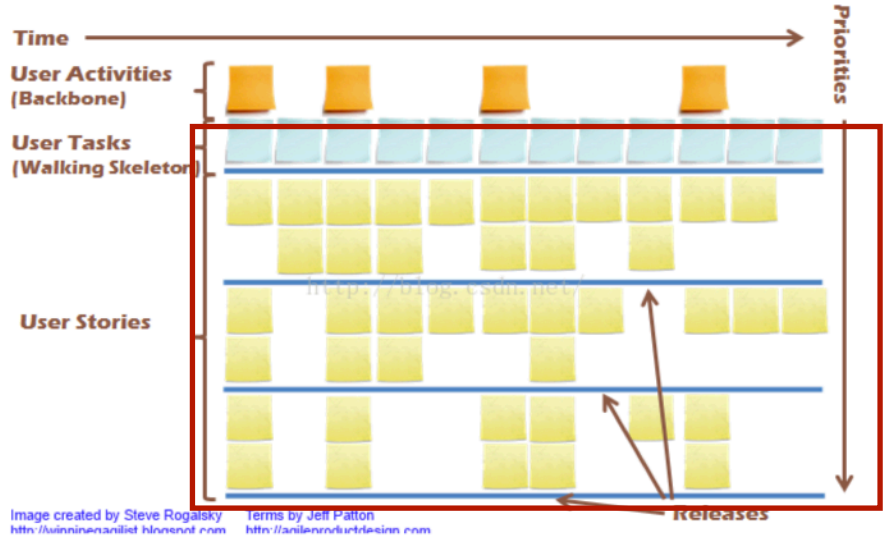
Using Stories to Drive the Process

Easy Agile User Story Maps Demo

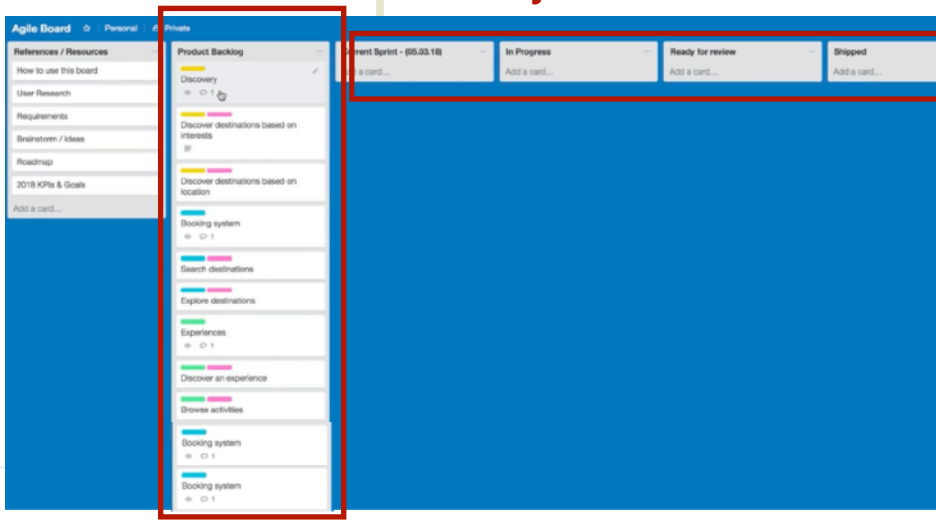


[Download t.co/getusm](t.co/getusm)

2.用户故事地图编写模板



Journey/Epic/Backbone or Project Status

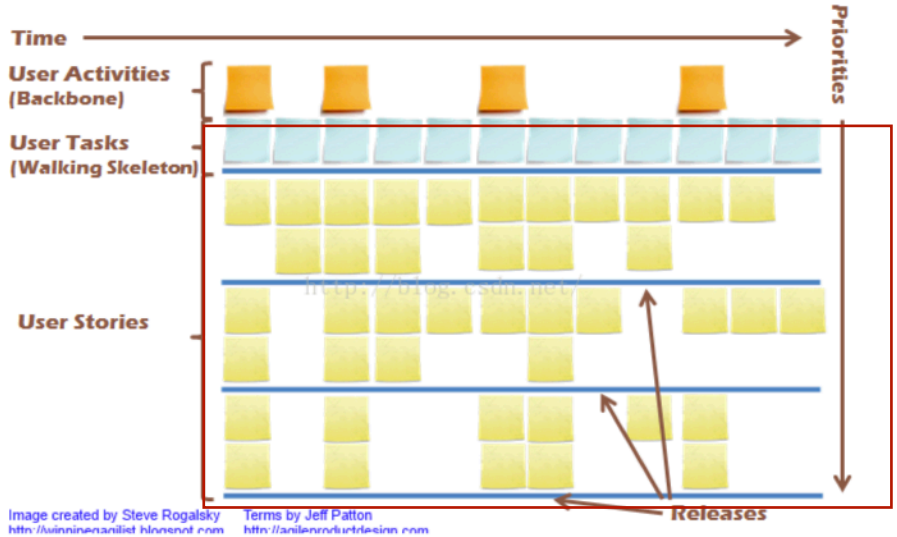


Mapping the backlog and sprinting tasks to the Kanban board to see how the project status changes over time.

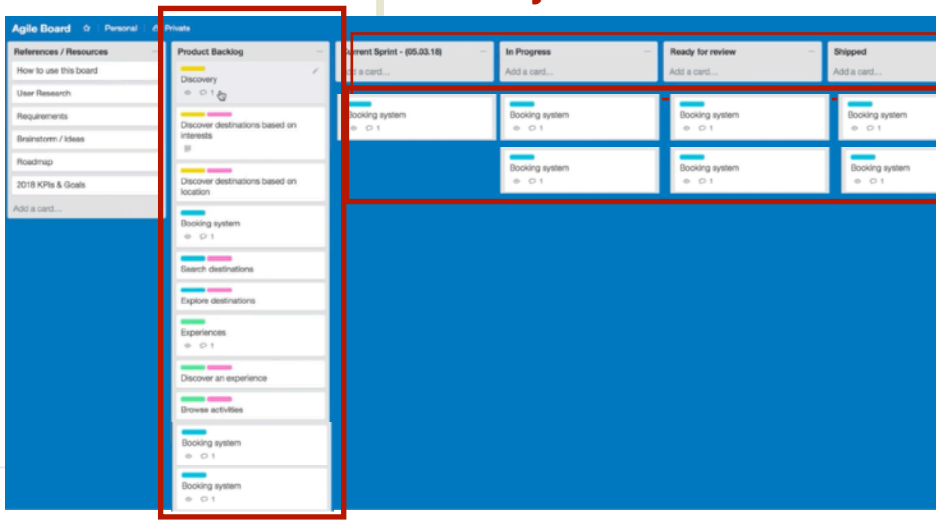
Backlog Made Up of User Stories

Kanban View by Journey Stage/Epic/Backbone

2.用户故事地图编写模板



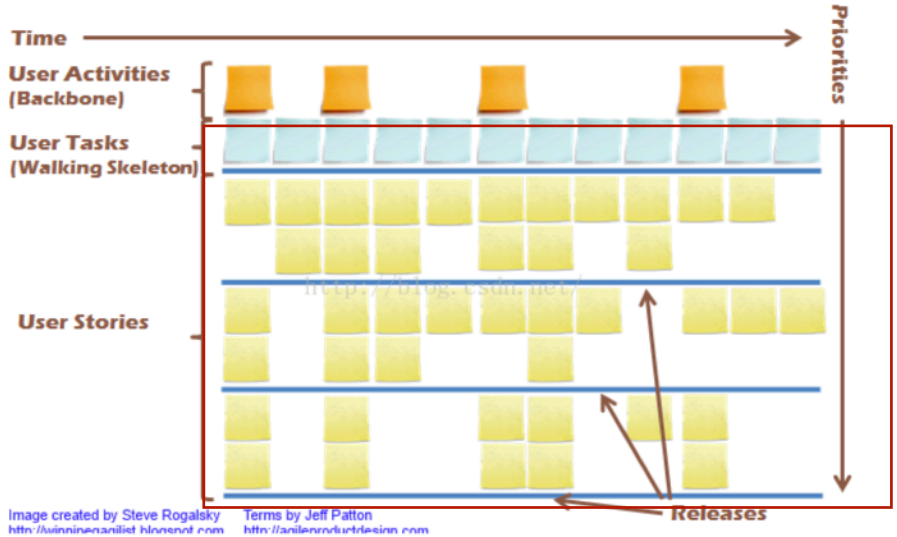
Journey/Epic/Backbone or Project Status



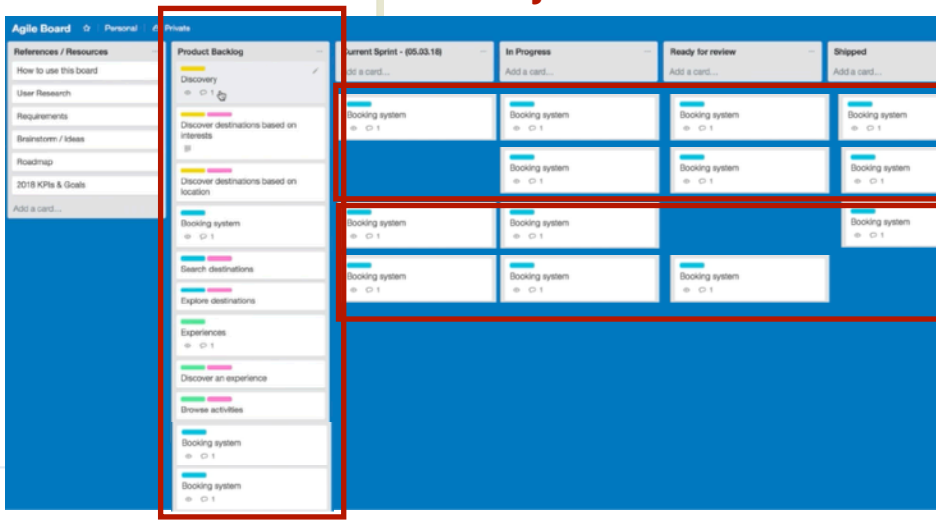
Mapping the backlog and sprinting tasks to the Kanban board to see how the project status changes over time.

Backlog Made Up of User Stories

2.用户故事地图编写模板

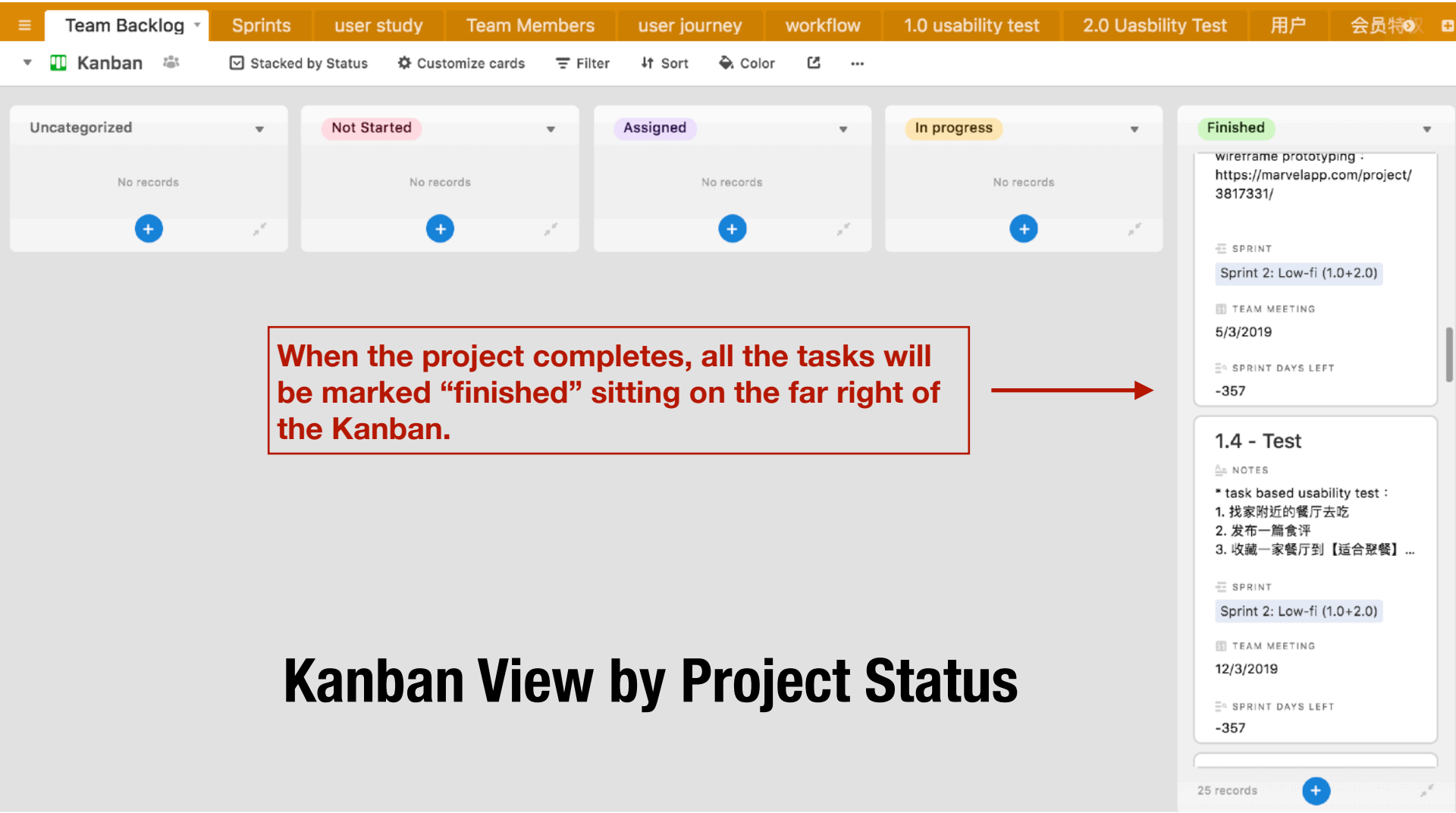


Journey/Epic/Backbone or Project Status



Mapping the backlog and sprinting tasks to the Kanban board to see how the project status changes over time.

Backlog Made Up of User Stories



When the project completes, all the tasks will be marked “finished” sitting on the far right of the Kanban.

Kanban View by Project Status

Building Different Views to Monitor the Project

Airtable Bases Templates Universe HELP

Templates

Use these starter bases to get a jump start on your project. For inspiration from community-published bases, check out Universe.

Search templates

CATEGORIES

- Featured
- Content production
- Creative
- Event Planning
- Everyday Life
- Groups, Clubs & Hobbies
- HR & Recruiting
- Legal
- Local Business
- Marketing
- Nonprofit
- Personal
- PR & Communications
- Product, design, and UX
- Project Management
- Publishing
- Real Estate

Agile Workflow

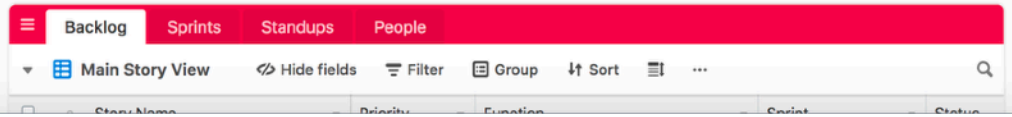
Use template

#PRODUCT, DESIGN, AND UX

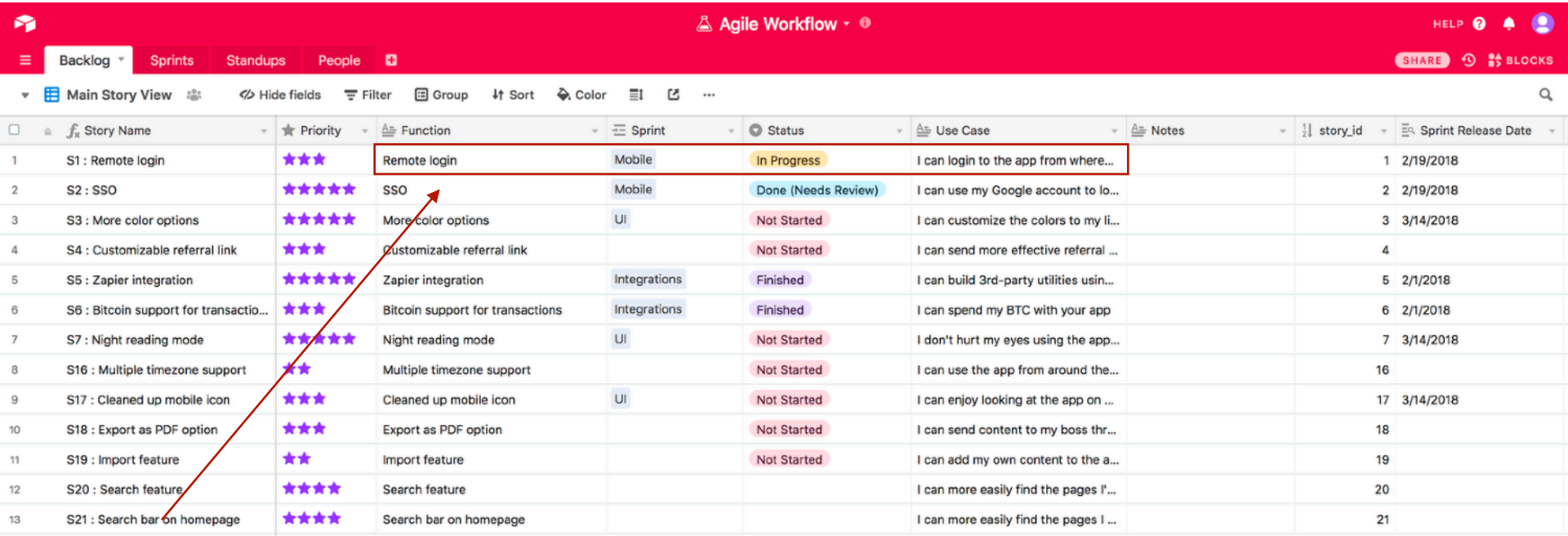
Even if you don't want to adopt the entire Agile manifesto, shifting your workflow to incorporate aspects of the Agile methodology—a small-a "agile" workflow—is a good idea for virtually any software development team.

By taking an agile approach to the development process, you let customer needs direct the features and improvements you're working on. By tailoring your product development to focus on the user, you waste less effort on irrelevant or unnecessary features than you do with traditional approaches.

Our blog post here walks you through how to use this template to apply agile processes to suit your team's needs. Hand it off to the scrum master to build your next sprint, schedule standups for your team, and make a note of stories for future sprints in one database that is easily shareable between your entire team. By tagging every potential feature or improvement in a project with the function it serves, the utility it provides to the customer, and its prioritization in the product, this base helps you gather steam and keep rolling into the next sprint after each completed project.



Adding Testing into the Picture

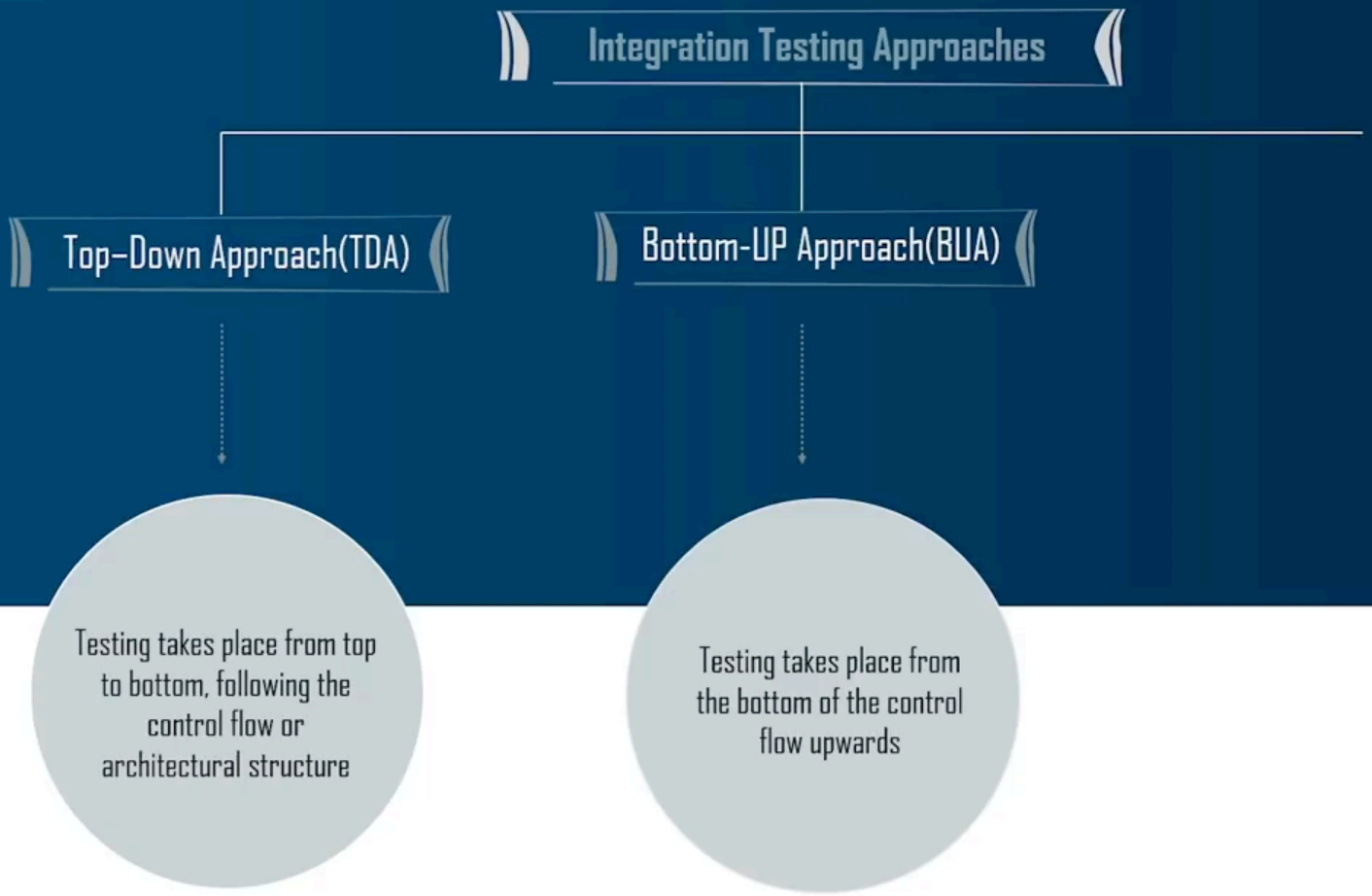


	Story Name	Priority	Function	Sprint	Status	Use Case	Notes	story_id	Sprint Release Date
1	S1 : Remote login	★★★	Remote login	Mobile	In Progress	I can login to the app from where...		1	2/19/2018
2	S2 : SSO	★★★★★	SSO	Mobile	Done (Needs Review)	I can use my Google account to lo...		2	2/19/2018
3	S3 : More color options	★★★★★	More color options	UI	Not Started	I can customize the colors to my li...		3	3/14/2018
4	S4 : Customizable referral link	★★★	Customizable referral link		Not Started	I can send more effective referral ...		4	
5	S5 : Zapier integration	★★★★★	Zapier integration	Integrations	Finished	I can build 3rd-party utilities usin...		5	2/1/2018
6	S6 : Bitcoin support for transactio...	★★★	Bitcoin support for transactions	Integrations	Finished	I can spend my BTC with your app		6	2/1/2018
7	S7 : Night reading mode	★★★★★	Night reading mode	UI	Not Started	I don't hurt my eyes using the app...		7	3/14/2018
8	S16 : Multiple timezone support	★★	Multiple timezone support		Not Started	I can use the app from around the...		16	
9	S17 : Cleaned up mobile icon	★★★	Cleaned up mobile icon	UI	Not Started	I can enjoy looking at the app on ...		17	3/14/2018
10	S18 : Export as PDF option	★★★	Export as PDF option		Not Started	I can send content to my boss thr...		18	
11	S19 : Import feature	★★	Import feature		Not Started	I can add my own content to the a...		19	
12	S20 : Search feature	★★★★	Search feature			I can more easily find the pages I'...		20	
13	S21 : Search bar on homepage	★★★★	Search bar on homepage			I can more easily find the pages I ...		21	

Each unit of work can be separately performed by one to two team members. Using the MVC framework to further break down the works is a common approach with some people working on the views (UI) while others working on the back-end preparing the data model, routes, API endpoints, and processing logic.

In some cases, there will be a content team busy acquiring and validating data to be ready for the database. All these works require testing, documentation and integration support. During more mature releases, users will be brought in for UAT.

edureka!


Integration Testing Approaches
Top–Down Approach(TDA)
Bottom-UP Approach(BUA)
Testing takes place from top to bottom, following the control flow or architectural structure
Testing takes place from the bottom of the control flow upwards

ASSIGNMENT #5

1. Change the **MySQL** connection module to **Flask-SQLAlchemy**.
2. Add **pagination** to the album.
3. **Add CRUD** support to the **“album” table** so new album entries can be added and old ones be updated and even deleted if needed.
4. Develop a mobile front-end based on the **REST API** for user to view the album (optional).


Reload bssuen.pythonanywhere.com
Best before date:
We're happy to host your free website – and keep it free – for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. See here for more details.
This site will be disabled on Monday 18 May 2020
Run until 3 months from today
Paying users' sites stay up forever without any need to log in to keep them running.
Traffic:
How busy is your site?
This month (previous month) 357 (26)
Today (yesterday) 0 (0)
Hour (previous hour) 0 (0)
Want some more data? Paying accounts get pretty charts ;-)
Code:
What your site is running.
Source code: /home/bssuen/mysite Go to directory
Working directory: /home/bssuen/ Go to directory
WSGI configuration file: /var/www/bssuen_pythonanywhere_com_wsgi.py
Python version: 3.7
Virtualenv:
Use a virtualenv to get different versions of flask, django etc from our default system ones. More info here. You need to Reload your web app to activate it; NB - will do nothing if the virtualenv does not exist.
Enter path to a virtualenv, if desired
Log files:
The first place to look if something goes wrong.
Access log: bssuen.pythonanywhere.com.access.log
Error log: bssuen.pythonanywhere.com.error.log
Server log: bssuen.pythonanywhere.com.server.log
Log files are periodically rotated. You can find old logs here: /var/log

THANK YOU FOR YOUR TIME!