Stripe Demo

Team Name: MicroManiacs

Team Members

1. Abhijeet Mehrotra (am4586)
2. Akshay Nagpal (an2756)
3. Kunal Baweja (kb2896)
4. Siddharth Shah (sas2387)

URLs

1. **S3 frontend**: http://s3-us-west-2.amazonaws.com/stripe6998/index.html
2. **Elastic Beanstalk (API URL)**: http://stripedeploy.pmi6pbp3mg.us-west-2.elasticbeanstalk.com/api/

API endpoints

Note: Calls autheticated by the token in case of invalid / expired token return 403.

Auth service

POST api/api-token-auth/
Request parameters

```
{
  "username": "dummy@user.com",
  "password": "password"
}
```

Response: 200 Success

```
{"token":"JWT_TOKEN_HERE"}
```

Response: 400 Failure

```
{"detail":"authorization failure"}
```

SignUp service

POST api/signup/

Request parameters

```
parameters = {
            "first_name": "foo",
            "last_name": "bar",
            "email": "foobar@gmail.com",
            "password": "password"
        };
```

Response: 201 Success

```
{"success":true}
```

Response: 400 Failure

```
{
   "success": false,
   "error": "failure message here"
}
```

Fetch product catalog service

GET api/product/

Response: 200 Success

```
[
{
  "id": 1,
  "price": 100,
  "description": "Brooklyn Bridge",
  "url": "https://c1.staticflickr.com/1/728/31226388014_5558604d0f_k.jpg"
},
{
  "id": 2,
  "price": 150,
```

```
      "description": "Singapore Grand Prix",
      "url": "https://c1.staticflickr.com/6/5763/20977162524_c8931fe2d3_k.jpg"
    }
  ]
```

Fetch orders of logged in user

GET api/order/

Response: 200 Success

```
[
  {
    "id": 14,
    "user": 5,
    "orderdate": "2017-02-11T02:40:24.333429Z",
    "paymentstatus": "PAID",
    "product": {
      "id": 1,
      "price": 100,
      "description": "Brooklyn Bridge",
      "url": "https://c1.staticflickr.com/1/728/31226388014_5558604d0f_k.jpg"
    }
  },
  {
    "id": 15,
    "user": 5,
    "orderdate": "2017-02-11T02:40:56.373584Z",
    "paymentstatus": "PAID",
    "product": {
      "id": 2,
      "price": 150,
      "description": "Singapore Grand Prix",
      "url": "https://c1.staticflickr.com/6/5763/20977162524_c8931fe2d3_k.jpg"
    }
  }
]
```

Submit order & stripe token to backend
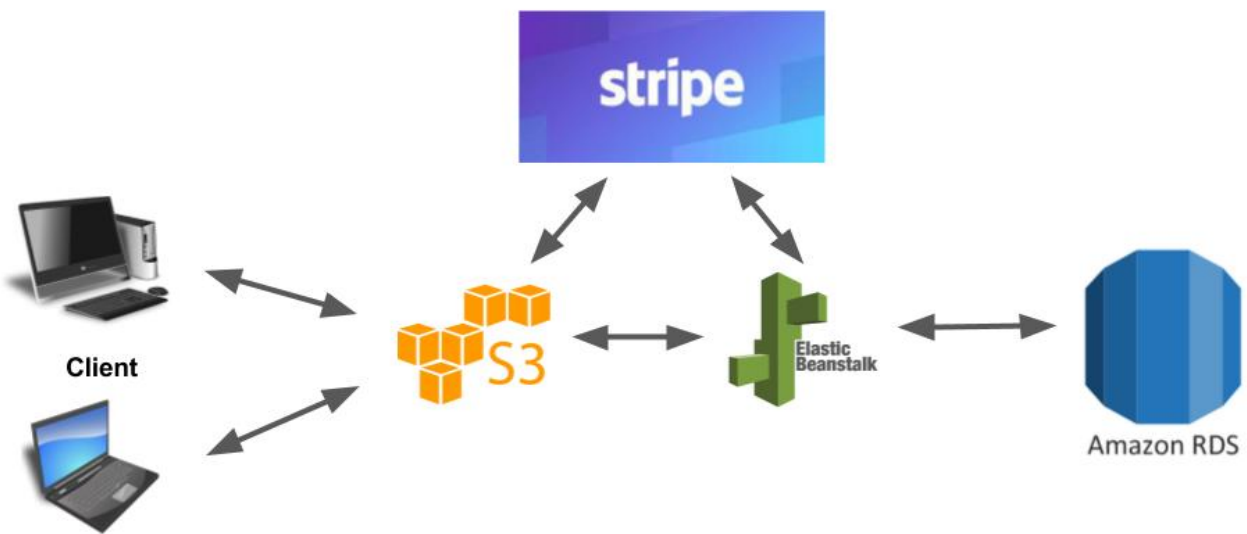
POST api/order/

Request parameters

```
{
    "token": "STRIPE_CLIENT_TOKEN",
    "product": "PRODUCT_ID"
}
```

Response: 201 Success

```
{
  "success": true
}
```

Response: 400 Bad request: In case of missing parameters.

Architecture

Architecture - Stripe Demo

Tech Stack

1. Python (Django REST Framework)
2. HTML5, CSS, Javascript
3. jQuery, Bootstrap
4. MySQL

Deployment

1. Front end static files hosted on S3 bucket
2. Database hosted on Amazon RDS

3. Backend server hosted on Elastic Beanstalk (Load Balancer + EC2 instance)

Communication with the Stripe Service

Client Side

Stripe.js was used to integrate payment popup on client side'

Server Side

Server end uses the Charge API to communicate with the Stripe service and store the order **meta data** on Stripe and the order status in the database

```
charge = stripe.Charge.create(
          amount=int(product.price*100),
          currency="usd",
          metadata={"order_id": order_id},
          source=stripe_token);
```

Screenshots

FrameSeller - Buy Amazing Photos!

Buy amazing photos online!

Existing User?

Login
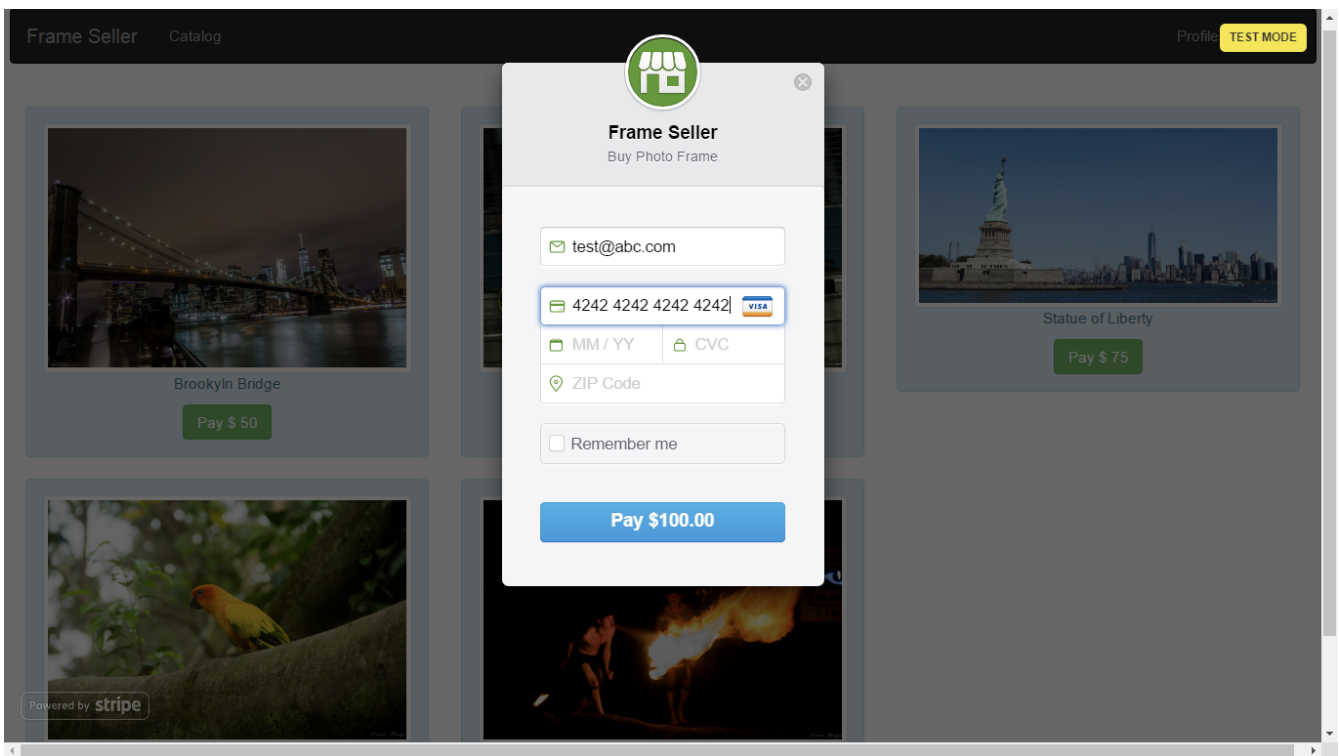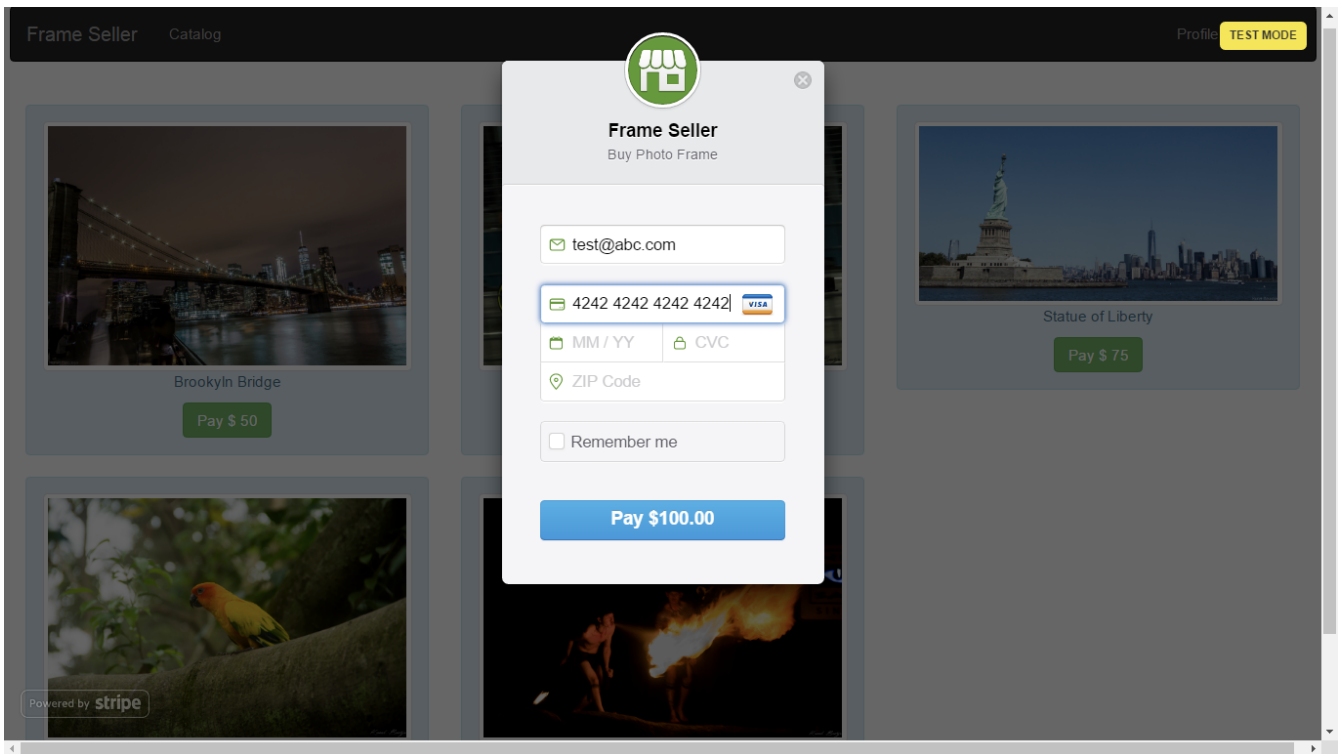
Email:

Password:

Login

New User?

SignUp

First Name:

Last Name:

Email:

Password:

Repeat Password:

Sign Up

Frame Seller   Catalog   Profile TEST MODE

Frame Seller
Buy Photo Frame

test@abc.com

4242 4242 4242 4242   VISA

MM / YY   CVC

ZIP Code

Remember me

Pay $100.00

Brookyln Bridge

Pay $ 50

Statue of Liberty

Pay $ 75

Powered by stripe

Frame Seller   Catalog   Profile TEST MODE

Frame Seller
Buy Photo Frame

test@abc.com

4242 4242 4242 4242   VISA

MM / YY   CVC

ZIP Code

Remember me

Pay $100.00

Brookyln Bridge

Pay $ 50

Statue of Liberty

Pay $ 75

Powered by stripe

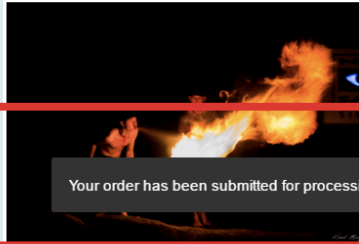Brookyln Bridge

Pay $ 50

F1 Racing

Pay $ 100

Statue of Liberty

Pay $ 75

Your order has been submitted for processing

Hello, Shahrukh Khan

Your orders

Parrot

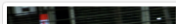**Date Placed on:** 2/10/2017 0:52

Price: $ 60

PAYMENT STATUS
PAID

F1 Racing

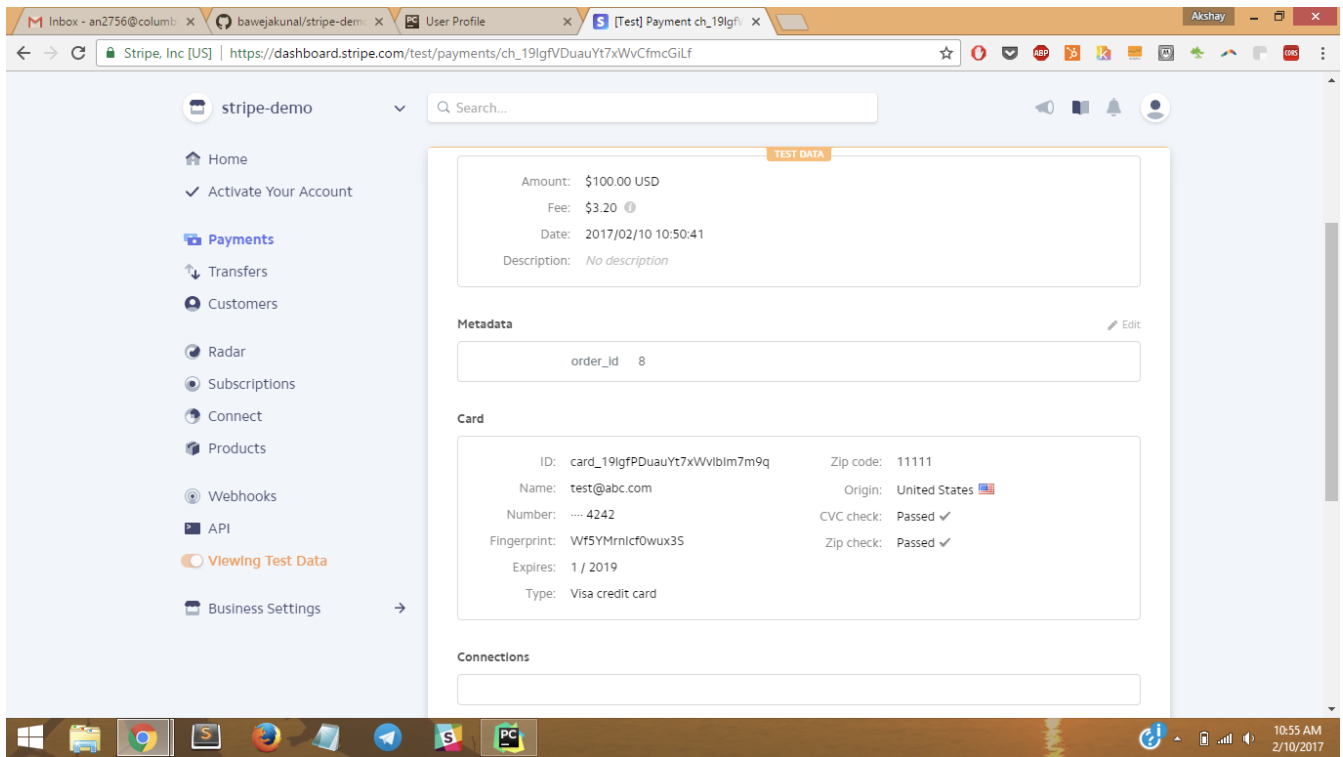**Date Placed on:** 2/10/2017 0:59

Price: $ 100

PAYMENT STATUS
PAID

Date Placed on: 2/10/2017 10:50

Further Improvements

1. Use [AngularJS](#) in future assignments
2. As suggested by Prof. Donald Ferguson, segregate the microservices further into Order, Payment and User services.
3. Add randomly generated `idempotency_key` in `stripe.Charge.create()` method call to implement [Stripe Idempotent Requests](#) for retrying payment requests that fail due to network errors.