Library Management Project

Group Members:
Xintong Zhan,
Feitong Zhu,
Joyce Qiao,
Jiafu Chen,
Qingwei Yang,
Haomiao Li

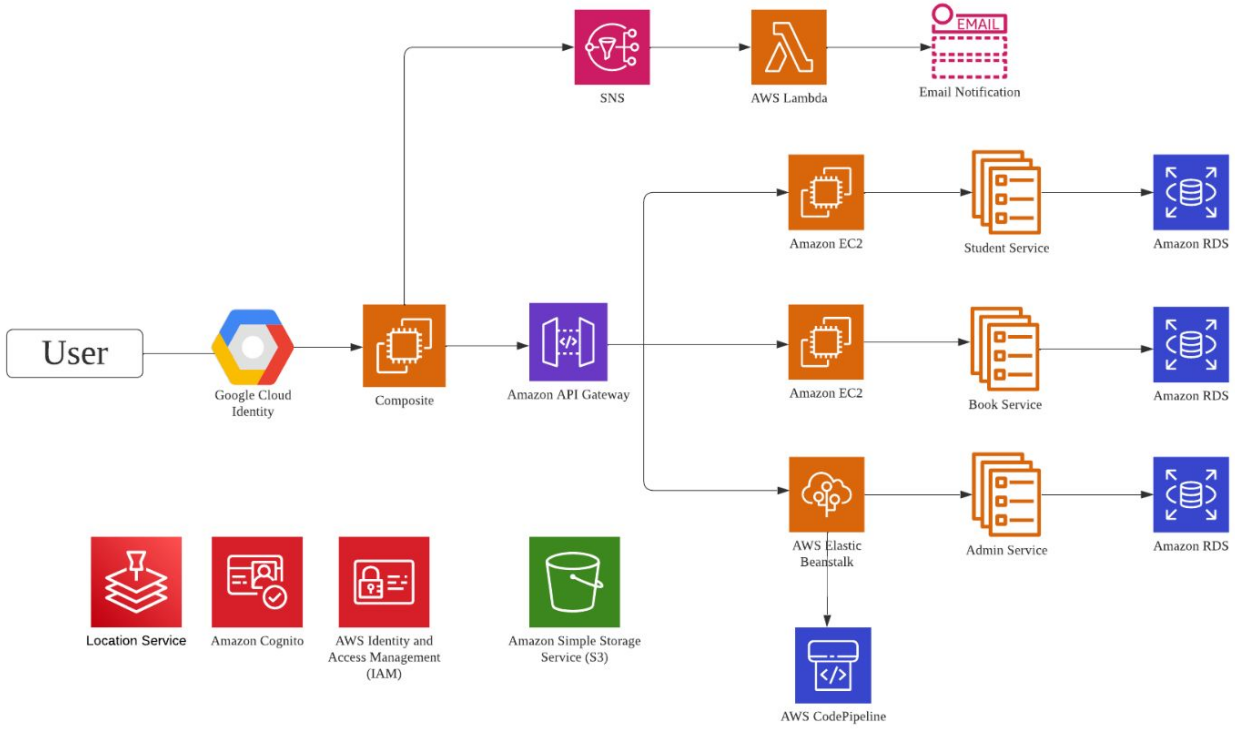
Our Project Services

- Student Service
- Admin Service
- Book Service
- Composition Service

Technical Deliverables:

- Microservice
- API Gateway
- EC2
- Elastic Beanstalk
- RDS
- Google Identify Service
- Simple Notification Service
- Lambda Function

Project Architecture



UI Demo

video...

API Gateway

Attach integrations to routes | Manage integrations

Routes for 6156_final

Search

- /admin_query
  - ANY HTTP ANY
- /book_add1
  - ANY HTTP ANY
- /book_in
  - ANY HTTP ANY
- /book_not
  - ANY HTTP ANY
- /book_query1
  - ANY HTTP GET
- /book_query2
  - GET HTTP GET
- /book_query3
  - ANY HTTP GET
- /book_query4
  - ANY HTTP GET
- /book_query5
  - GET HTTP GET
- /books_query
  - ANY HTTP ANY
- /booksquery
  - ANY HTTP ANY
- /find_function
  - GET HTTP GET

Integration details for route

Detach integration | Manage integration

ANY /book_in (tmuopxq)

HTTP URI
ANY http://18.216.35.187:5000/book_in

Integration ID
mkgnpeu

Description
-

Timeout
The number of milliseconds that API Gateway should wait for a response from the integration before timing out.
30000

Request parameter mapping
Not configured

Response parameter mappings
Not configured

Student Service

student.py

6156-admin_db > student.py

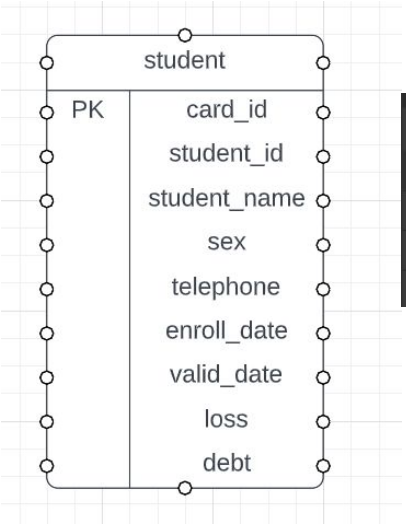
```
    return result

@app.get("/book_query9")
def book_query9():
    temp = copy.copy(request.args)
    user = Student.query.filter_by(card_id=temp['card_id']).first()
    if user is None:
        msg={'student_name':'Not found'}
    else:
        msg = {'card_id':user.card_id ,'student_id':user.student_id,'student_name':user.student_name,'sex':user.sex,'te
    result = Response(json.dumps(msg), status=200, content_type="application/json")
    print(msg)
    return result

@app.get("/book_query11")
def book_query11():

    temp = copy.copy(request.args)
    user = Student.query.filter_by(card_id=temp['card_id']).first()
    if user is None:
        msg={'student_name':'Not found'}
    else:
        msg = {'card_id':user.card_id ,'student_id':user.student_id,'student_name':user.student_name,'sex':user.sex,'te
    result = Response(json.dumps(msg), status=200, content_type="application/json")
    print(msg)
    return result
```

Student Database:schema



	card_id	student_id	student_name	sex	telephone	enroll_date	valid_date
1	fz2348	fz2348	Feitong Zhu	Female	18921902722	1544371200000	2644371200000
2	dq1232	dq1232	Dan Qing	Female	1593446400000	1544371200000	2564371200000
3	ka4634	ka4634	Kris Alex	Male	1593446400000	1544371200000	2644371200000
4	xz3165	xz3165	Xintong Zhan	Male	9291212291	1544371200000	2644371200000

Student Service:EC2

```
student_server_url="http://3.141.15.137:5000"
```

Instance summary for i-00a97cd38c67580f8 (6770) Info

Updated less than a minute ago

Instance ID i-00a97cd38c67580f8 (6770)	Public IPv4 address 3.141.15.137 | open address
IPv6 address –	Instance state Running
Hostname type IP name: ip-172-31-18-164.us-east-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-18-164.us-east-2.compute.internal
Answer private resource DNS name IPv4 (A)	Instance type t2.micro
Auto-assigned IP address 3.141.15.137 [Public IP]	VPC ID vpc-079b6cea9021d3724
IAM Role –	Subnet ID subnet-0fa1b3acd952c9f0b

Admin Service

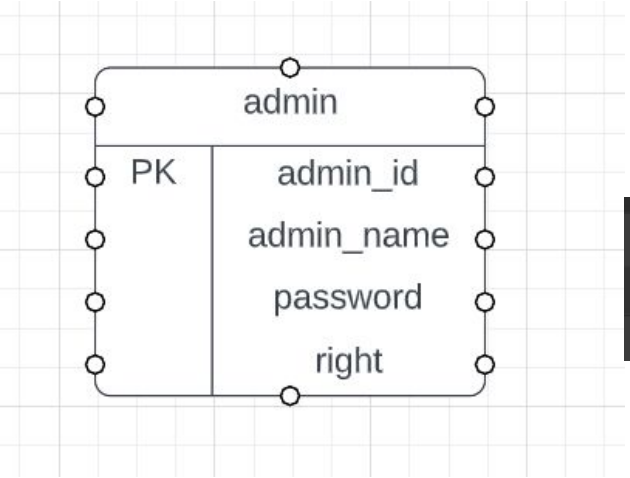
```python
    def get_id(self):
        return self.admin_id

    def verify_password(self, password):
        if password == self.password:
            return True
        else:
            return False

    def __repr__(self):
        return '<Admin %r>' % self.admin_name
```

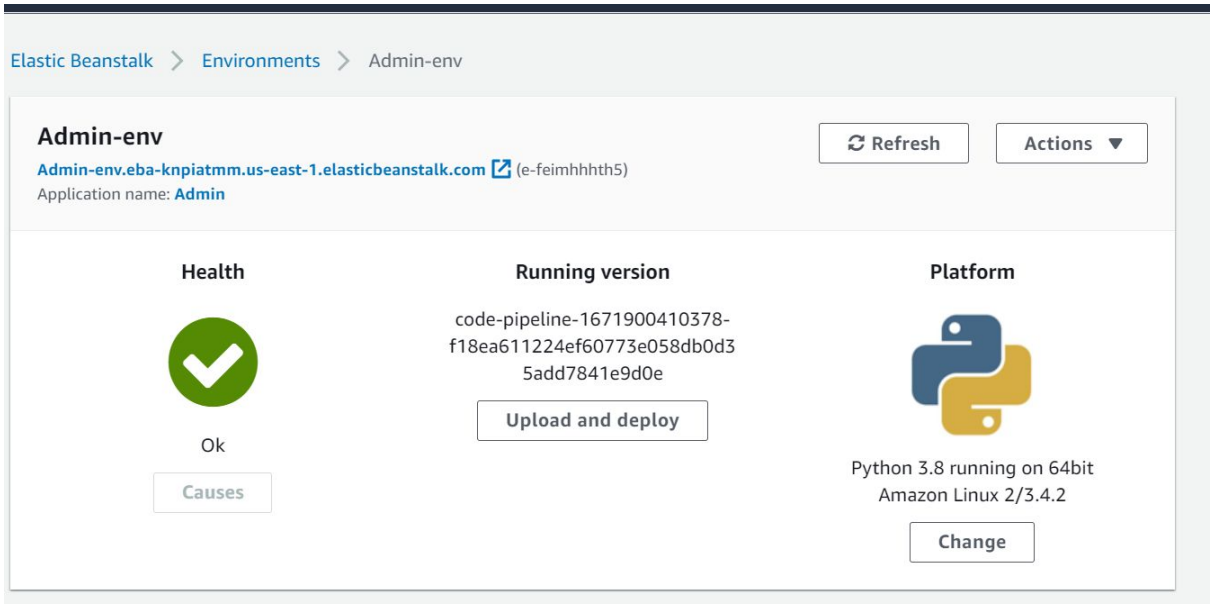
```python
@app.get("/admin_query")
def admin_query():
    temp=copy.copy(request.args)
    user = Admin.query.filter_by(admin_id=temp['admin_id'], password=temp['password']).first()
    print(temp['admin_id'])
    msg={'admin_id':user.admin_id,'admin_name':user.admin_name,'password':user.password,'right':user.right}
    result = Response(json.dumps(msg), status=200, content_type="application/json")
    print(result)
    return result
```

Admin Database: Schema

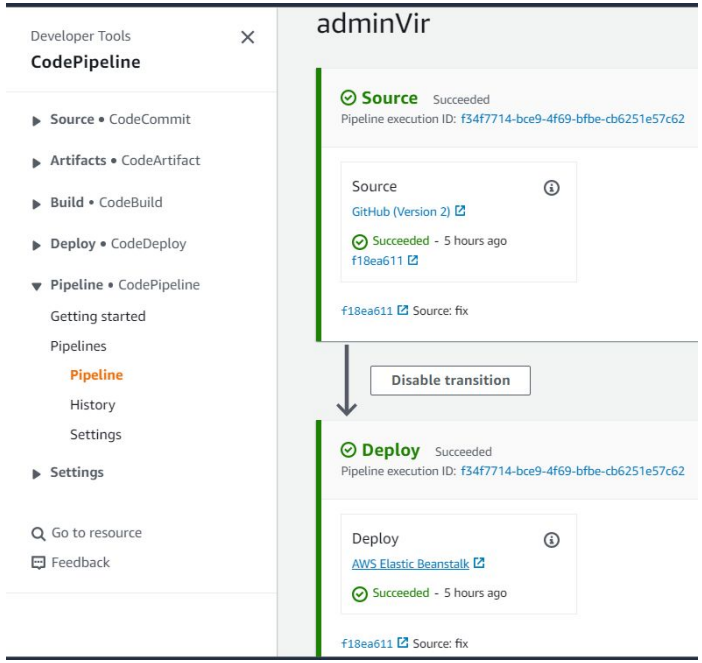


	admin_id	admin_name	password	right
1	xz3165	Xintong Zhan	123	root
2	fz2348	Feitong Zhu	123	root

Admin Service:Elastic Beanstalk Deployment



Admin Service



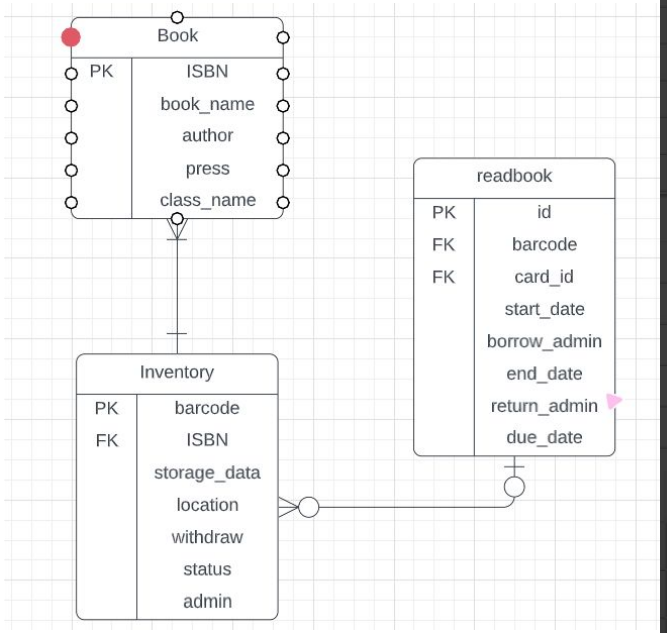
Book Service

```python
@app.get("/book_in")
def book_in():
    temp = copy.copy(request.args)
    record = ReadBook.query.filter(ReadBook.barcode == temp['barcode'], ReadBook.card_id == temp['card_id'],
                                   ReadBook.end_date.is_(None)).first()
    today_date = datetime.date.today()
    today_str = today_date.strftime("%Y-%m-%d")
    today_stamp = time.mktime(time.strptime(today_str + ' 00:00:00', '%Y-%m-%d %H:%M:%S'))
    record.end_date = int(today_stamp) * 1000
    record.return_admin = temp['admin_id']

    db.session.add(record)
    db.session.commit()

    book = Inventory.query.filter_by(barcode=temp['barcode']).first()
    book.status = True
    db.session.add(book)
    db.session.commit()
    bks = db.session.query(ReadBook).join(Inventory).join(Book).filter(ReadBook.card_id == temp['card_id'],
                                                                       ReadBook.end_date.is_(None)).with_entities(
        ReadBook.barcode, Book.isbn, Book.book_name, ReadBook.start_date,
        ReadBook.due_date).all()
```

Book Database: schema



isbn	book_name	author	press
978704015109X	ERP Principle and Application Training	Qingming Wang	Higher Education Press
9787040273243	Management Information System	Tiyun Huang	Higher Education Press
9787115335500	node.js Tutorial	Ling Pak	Posts and Telecommunic
9787121204869	Mobile Design	Xiaozhen Fu	Electronic Industry Pr
9787302292609	Experiment Course of ERP Production Mana	Lili Zhang	Tsinghua University Pr
978710800982x	Fifteen Years of Wanli	Renyu Huang	Joint Publishing

	barcode	isbn	storage_date	location	withdraw
1	102341	9787302423287	1514736000000	1 floor,02 sh	0
2	102342	9787302423287	1514736000000	1 floor,02 sh	0
3	102343	9787302423287	1514736000000	1 floor,02 sh	0
4	102344	9787302423287	1514736000000	1 floor,02 sh	0
5	211411	9787302292609	1514736000000	2 floor,11 sh	0
6	211412	9787302292609	1514736000000	2 floor,11 sh	0

id	barcode	card_id	start_date	borrow_admin	e
1	102341	16000001	1544371200000	xz3165	154
2	102342	16000002	1545926400000	xz3165	154
3	310321	16000001	1546012800000	xz3165	154
4	203773	16000001	1546012800000	xz3165	154

Book Service: EC2

```
book_server_url="http://18.216.35.187:5000"
```

Instance summary for i-0c652184d80366951 (6770_part2.1) Info

Updated less than a minute ago

Instance ID i-0c652184d80366951 (6770_part2.1)	Public IPv4 address 18.216.35.187 | open address
IPv6 address –	Instance state Running
Hostname type IP name: ip-172-31-38-184.us-east-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-38-184.us-east-2.compute.internal
Answer private resource DNS name IPv4 (A)	Instance type t2.micro
Auto-assigned IP address –	VPC ID vpc-079b6cea9021d3724
IAM Role –	Subnet ID subnet-03310858477cc1a0f

Composite Service

Flask+CSS+HTML+JS

```python
@app.route('/change_password', methods=['GET', 'POST'])
@login_required
def change_password():
    form = ChangePasswordForm()
    if form.password2.data != form.password.data:
        flash(u'Inconsistent with the password above!')
    if form.validate_on_submit():
        if current_user.verify_password(form.old_password.data):
            current_user.password = form.password.data
            db.session.add(current_user)
            db.session.commit()
            flash(u'Change password successfully!')
            return redirect(url_for('index'))
        else:
            flash(u'Wrong original password. Change Failed!')
    return render_template("change-password.html", form=form)
```

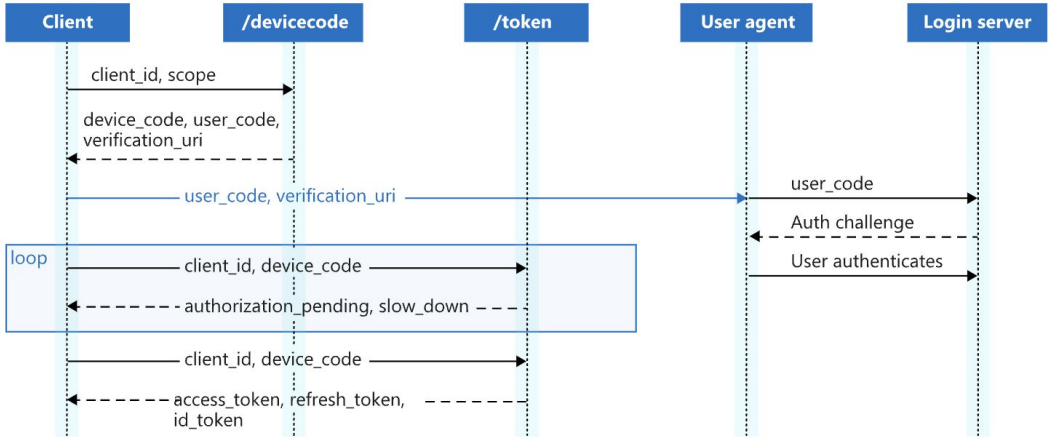
```python
@app.route('/change_info', methods=['GET', 'POST'])
@login_required
def change_info():
    form = EditInfoForm()
    if form.validate_on_submit():
        current_user.admin_name = form.name.data
        db.session.add(current_user)
```

- templates
  - base-user.html
  - base.html
  - base2.html
  - borrow.html
  - change-info.html
  - change-password.html
  - index.html
  - library_map.html
  - login.html
  - new-store.html
  - return.html
  - search-book.html
  - search-student.html
  - storage.html
  - user-book.html
  - user-info.html
  - user-student.html

- layui
  - css
    - modules
    - layui.css
    - layui.mobile.css
  - font
  - images
  - lay
  - layui.all.js
  - layui.js
- icon.jpg
- login.css
- map.css

Google Identify Service

The entire device code flow is shown in the following diagram. Each step is explained throughout this article.



Google Identify Service

Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more

DISMISS ACTIVATE

Google Cloud | 6156final | Search (/) for resources, docs, products, and more | Search

API APIs & Services

- Enabled APIs & services
- Library
- Credentials
- OAuth consent screen
- Page usage agreements

Client ID for Web application

DOWNLOAD JSON | RESET SECRET | DELETE

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

URIs 1 *
http://ec2-3-87-44-167.compute-1.amazonaws.com

+ ADD URI

Authorized redirect URIs

For use with requests from a web server

URIs 1 *
http://ec2-3-22-168-24.us-east-2.compute.amazonaws.com:5000/

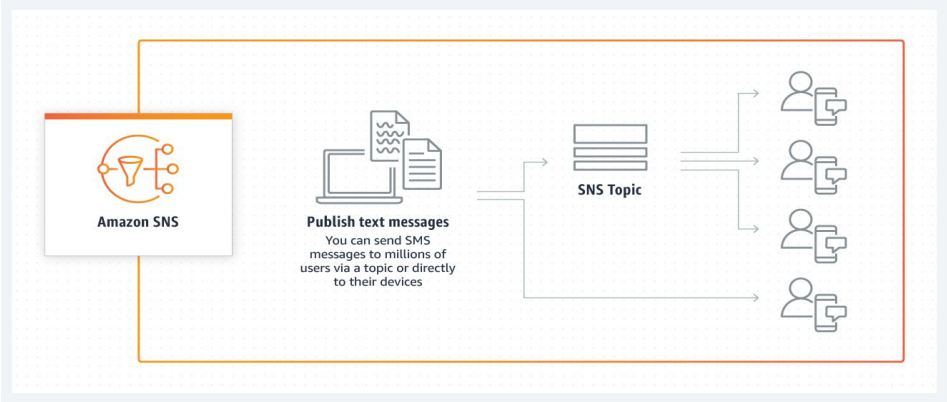
+ ADD URI

Note: It may take 5 minutes to a few hours for settings to take effect

SAVE CANCEL

Simple Notification Service (SNS)

- Connect with amazon account using aws access key and secret key
- Create Topic
- Create Subscription with the users



Topics (3)

Edit | Delete | Publish

Search

Name	Type	ARN
Borrow	Standard	arn:aws:sns:us-east-1:314437740289:Borr
Return	Standard	arn:aws:sns:us-east-1:314437740289:Retu

Lambda Function

- Implement a Lambda function that subscribes to the event

- SNS event triggers the Lambda function

- Send a Slack message to the specific channel using a webhook