



# INTELLIGENT PHOTO SERVICE

Instructor Dr. JOHN JENQ

## ABSTRACT

Salesforce is a cloud-based company provides SaaS, PaaS services to many industries to accelerates the business relationships with customer. Salesforce easily available with minimum requirements without any installation. Intelligent Photo Service application build and developed in salesforce lightning environment by utilizing Salesforce point and click approach as well as by developing aura components. The Einstein Vision and its prediction builder used to make the application intelligent. The application divided into three views – Business administrator view, customer community view and Photographer Community view.

**Submitted by Shradha Mhaske**

CSIT697\_85SP20 MASTER'S PROJECT



## Table of Contents

<b>Purpose of System</b> .....	4
<b>Goals and Impact</b> .....	4
<b>System requirements Elicitation</b> .....	6
<b>Functional requirements</b> .....	6
<b>Nonfunctional Requirements</b> .....	7
<b>Software and Hardware Requirements</b> .....	8
<b>Cloud Computing</b> .....	9
<b>What Is Salesforce</b> .....	9
<b>Why Salesforce?</b> .....	9
<b>System Design</b> .....	10
<b>Use Case Diagram</b> .....	10
<b>Business Administrator Use Case</b> .....	10
<b>Client Use Case</b> .....	10
<b>Photographer Use Case</b> .....	11
<b>Flow Chart Diagram</b> .....	12
<b>Search Photographer Flow Chart</b> .....	12
<b>Image Prediction by Einstein Vision flow Chart</b> .....	13
<b>Database Design</b> .....	16
<b>Intelligent photo Service Architecture</b> .....	21
<b>Einstein Vision API</b> .....	23
<b>Machine Learning Life Cycle [MLOps]</b> .....	24
<b>Data Analysis and Visualization</b> .....	28
<b>Artificial intelligence</b> .....	28
<b>Einstein Platform</b> .....	28
<b>Data and Salesforce Einstein</b> .....	28
<b>Interface Design</b> .....	31
<b>Business Administrator Workspace</b> .....	31
<b>Photographer Community</b> .....	33
<b>Customer Community</b> .....	35
<b>Test design</b> .....	37
<b>Introduction</b> .....	37
<b>Types of Testing in Salesforce</b> .....	37

<b>Test Screens .....</b>	<b>38</b>
<b>Future Enhancement.....</b>	<b>40</b>
<b>User Manual .....</b>	<b>41</b>
<b>Contact Information.....</b>	<b>41</b>
<b>Getting Started .....</b>	<b>41</b>
<b>Business Administrator User Manual .....</b>	<b>41</b>
<b>Photographer User Manual.....</b>	<b>43</b>
<b>Client User Manual.....</b>	<b>44</b>
<b>Glossary .....</b>	<b>45</b>
<b>Bibliography .....</b>	<b>46</b>
<b>Appendix.....</b>	<b>47</b>
<b>Einstein Machine Learning Component .....</b>	<b>47</b>
<b>Photographer Ranking Component .....</b>	<b>53</b>
<b>Photographer Search Component.....</b>	<b>57</b>

### **Purpose of System**

There are large number of commercial applications for photographers. Many of the industries providing vast varieties of customer relationship management software. These software's and applications are useful for photographers to attend the events, creating large number of client contacts, making albums with different aspects and customizing the images based on client expectations. Photographers can easily utilize the current system by taking advantages of available applications. These applications or software's are crucial tools for photographers to expand their photography business.

Despite an abundance of data and modern technologies, the photography applications are running behind the market. This project aims to simplify some of many hassles that come with gathering information to make an informed decision prior moving to one client to another.

After reviewing some existing photography applications, Intelligent Photo Service application is built and developed to meet the customer requirements and satisfy their expectations. Intelligent photo Service is a cloud-based application developed in Salesforce. Salesforce is the leading Customer relationship Management provider in the market. This application will be helpful for photographers and their clients. This application let photographers and clients to login and use the application from any device, mobile from anywhere. Both users – Photographers and Clients - only need community link to login without any disturbance. As Intelligent Photo Service is cloud-based, users need Internet connection and Wi-Fi.

The motive of Intelligent Photo Service allows user to perform and utilize multiple functionalities to expand their business. This application also helpful tool for clients who are searching right fit for their event photography.

Additionally, Intelligent Photo Service built using cloud architecture where the total functionality utilizing multitenant, trusted cloud. The cloud architecture of this application made up of different services like data services, Einstein vision, Einstein Analytics and prediction services, and robust APIs for development. The application sits on top of force.com platform and integrated with all layers of cloud architecture. The cloud architecture is explained in detail in the further sections of this documentation.

### **Goals and Impact**

The many existing photography software's provides features to photographers for developing images and managing their clients allowing some restrictions. The Intelligent Photo Service provides different functionalities to photographers and their clients. This application can give easy access of all clients to specific photographers. The future clients who are wishing to join this application are added by business administrator. After few steps, Client will allow to login their own environment - Customer Community. Customer Community provides the platform for clients to explore their search of photographers and grasp highly skilled photographers using few clicks.

Photographers who wanted to join the photographer community, allow to contact business administrator of an application. After successful addition into application, Photographer will

## Intelligent Photo Service

allow to login their personal environment – Photographer community. This community offers users to communicate within the environment seamlessly.

Business administrator have permissions to rank the photographers after each event performed. Photographers need to upload event image to the community. The rank will be displayed at the platform level and automatically updated after ranking process is done.

Intelligent photo service aims to utilize the prediction service of salesforce to recognize the event images uploaded by photographers. The machine learning techniques used to train the Event image dataset and match the perform the image recognition functionality. This application also gives not but least the Einstein analytics feature to business administrator to explore and visualize the data and take further steps if necessary.

## **System requirements Elicitation**

### **Functional requirements**

Following are the snapshot of requirement elicitation that were used to create the Intelligent Photo Service:

- Intelligent Photo Service divides the user into three categories – Business Administrator, Photographer and Client.
- Every user needs to do registration once in a lifetime.

### **Administrative Requirements**

- Administrators should be able to create and assign profiles, permission sets to users.
- Administrators should be able to handle all basic administrative duties like, account creation, managing workflows, process builder and many other routines.
- Administrators should be able to integrate with external systems.
- Administrators should be able to work independently with members of user community to define and develop document requirements.
- Administrators should be able to generate reports and dashboards.
- Administrators should be able to utilize the developed functionality of machine learning techniques.
- Administrators should be able to rank the photographers based on their uploaded photos.
- An automatic email confirmation should be sent to user to reset the password for community.

### **Photographer Requirements**

- After login, photographers should be able to see home page as a landing page.
- Photographers should be able to create and manage albums and personal clients.
- Photographers personal clients should not be visible to salesforce business administrator.
- Photographers should be able to upload event images to the community.
- Photographers should be able to send messages to the same community members.
- Photographers should be able to create case and notify to business administrator.

### **Client Requirements**

- After login, client should be able to see home page as a landing page.
- Client should be able to send messages to the same community members.
- Client should be able to choose event type from drop down list.
- Client should be able to get highest skilled or ranked photographer for selected event type.
- Client should be able to send email or contact photographer.
- Client should be able to create case and notify to business administrator.

## **Nonfunctional Requirements**

### **Security**

- Application should follow salesforce security model which includes object level security, field level security and organizational level security.
- Login credentials should not disclose to external users.
- Users personal information must be intact with the Intelligent Photo Service application.
- Uploaded photos and event images and description should be private to users.

### **Performance**

- The application is accessed from salesforce cloud should be available within seconds.
- Application should be available regardless of geographical locations.
- Application should be able to access from chrome, safari browsers and supported android and iOS mobile devices too.

### **Reliability, Maintenance and Scalability**

It is the intent of this project to develop the application which is versatile and can be built and improved as needed. The system can potentially be updated to work on future technologies and able to integrate seamlessly.

### **Training**

Training of the Intelligent photo service application will not be required as the user interface of an application is friendly and welcoming to new users. Should a user be unable to learn the functionalities of the system without assistance, the user should reference the system's included user manual. Should other, more serious system malfunctions arise, the system administrator should be contacted. This contact information is also provided in the system's user manual.



### **Software and Hardware Requirements**

There is no such requirement to run and execute the Intelligent Photo Service. The Intelligent Photo Service application built on lightning platform of salesforce. The Internet and Wi-Fi connection is needed to communicate with this application. Almost all salesforce user type licenses and editions are supported to this application. The business administrator of this application needs to login to salesforce developer organization using valid login credentials.

The new users of this application required to hold the customer community user license and enable as a customer user permission to login to community. After enabling permission as a customer user to the user, Business administrator will share the URL of community to the user. The user will be able to access the community through URL.

Client Community URL - <https://intelligentphotoservic-developer-edition.na174.force.com/Customers>

Photographer Community URL - <https://intelligentphotoservic-developer-edition.na174.force.com/PhotographersIPS>

### **Supported Mobiles**

The application also only UI supported by salesforce for mobile devices. Salesforce for Android & iOS is available either via mobile web browser (as a mobile web experience) or as a mobile app, which can be downloaded from the App Store for iOS devices or Google Play for Android devices.

### **Supported Browsers**

- Microsoft Internet Explorer - for lightning experience and lightning communities [IE11]
- Google Chrome - for lightning experience and lightning communities [Latest]
- Mozilla Firefox - for lightning experience and lightning communities [Latest] – does not support private browsing
- Apple Safari - for lightning experience and lightning communities [Latest]
- Microsoft Edge - for lightning experience and lightning communities [Latest]

### **Cloud Computing**

Cloud computing is the latest trend in market which facilitates various functionalities such as outsourcing data, data storage and processing power. The growth of cloud computing has been supported by several foundation technologies that have allowed companies like Amazon, Salesforce, Google, and Microsoft to provide computing resources to consumers more efficiently and at larger scales than ever. Enterprises no longer need to invest in expensive server hardware or staff an IT department to manage that hardware. At the heart of cloud computing is the characteristic of resource pooling, which allows cloud service providers to use shared computing resources to provide a service to multiple customers at once.

### **What Is Salesforce?**

Salesforce is an American cloud-based company which provides CRM software and multiple cloud-based functionalities. This software supports businesses to connect with each other and facilitates interaction between them. As of 2017, Salesforce reportedly had 150,000 companies using their software - among which include Amazon (**AMZN**), Adidas (**ADDYY**), ADP (**ADP** - Get Report), American Express (**AXP**) and many more... Salesforce cloud based CRM software helps companies to track their business , analytics and support.

Salesforce Lightning At initial stage, salesforce has “Salesforce classic” user interface but in the course of time, Salesforce re-imagined the salesforce classic to a modern, revamped solution for CRM is “SALESFORCE LIGHTNING”.

Salesforce lightning and salesforce classic both shares the common features but the user interface of lightning have various outstanding functionalities like, efficient and smart interface, visual and dynamic view of information, Kanban view and settings, territory management, integration with multiple applications from AppExchange and many more...

### **Why Salesforce?**

Intelligent photo service application fully developed and functioning on salesforce lightning platform. Salesforce provides highly customizable CRM with multiple functionalities. These functionalities support complex business processes and gives the way to expand sales productivity, customer service and marketing strategy. The prominent use of salesforce is to integrates with third party applications with their CRM.

## System Design

### Use Case Diagram

#### Business Administrator Use Case

The Use case [Figure 1] of Intelligent Photo Service allow Administrator to login to the Salesforce Developer Organization. The Administrator use case allow to manage users, create and assign profiles to users, assign permission sets, create and maintain communities, rank the photographers using Einstein Vision Service, and able to utilize the salesforce platform point and click declarative approach.

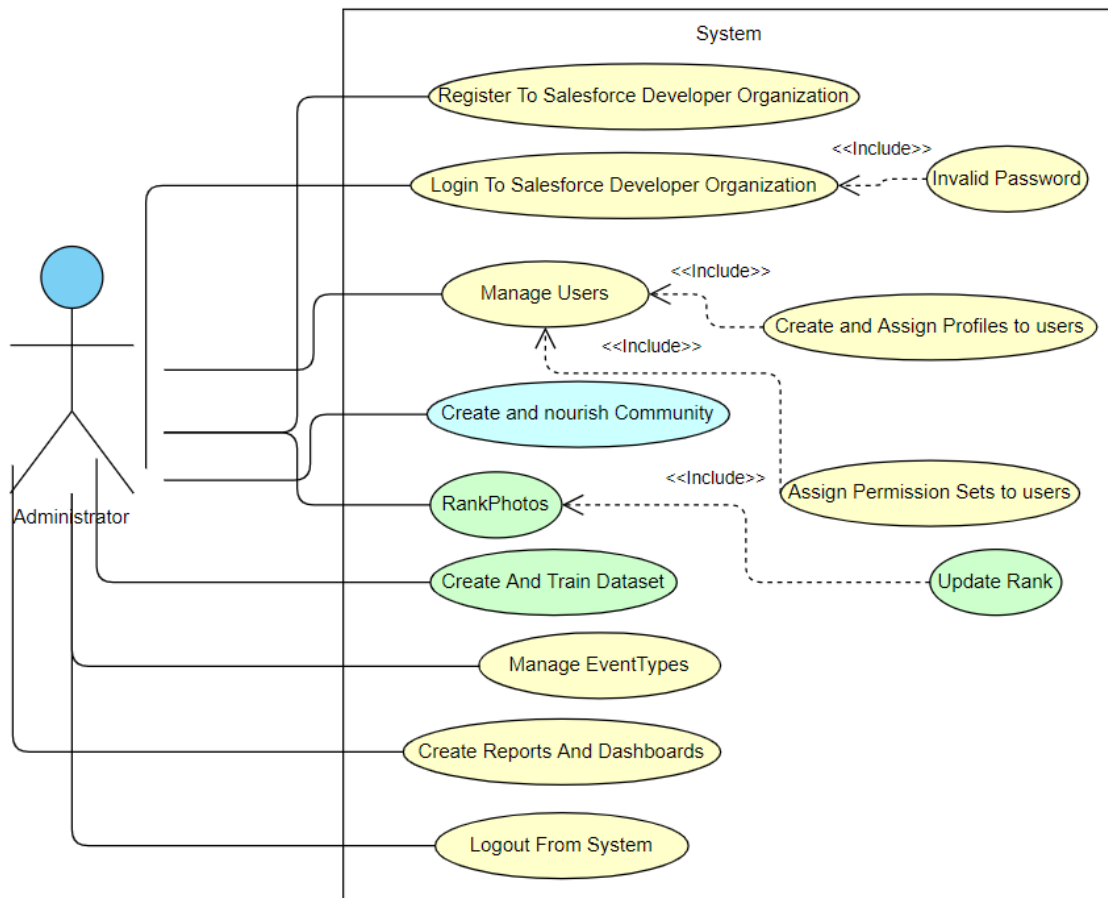


Figure-1 Business Administrator Use Case

#### Client Use Case

The Use case [Figure 2] Of Intelligent Photo Service allows client to Login and Logout to the Customer Community. Client also allow to create personal album, upload photos, send messages internally to community members, search photographers and email to them.

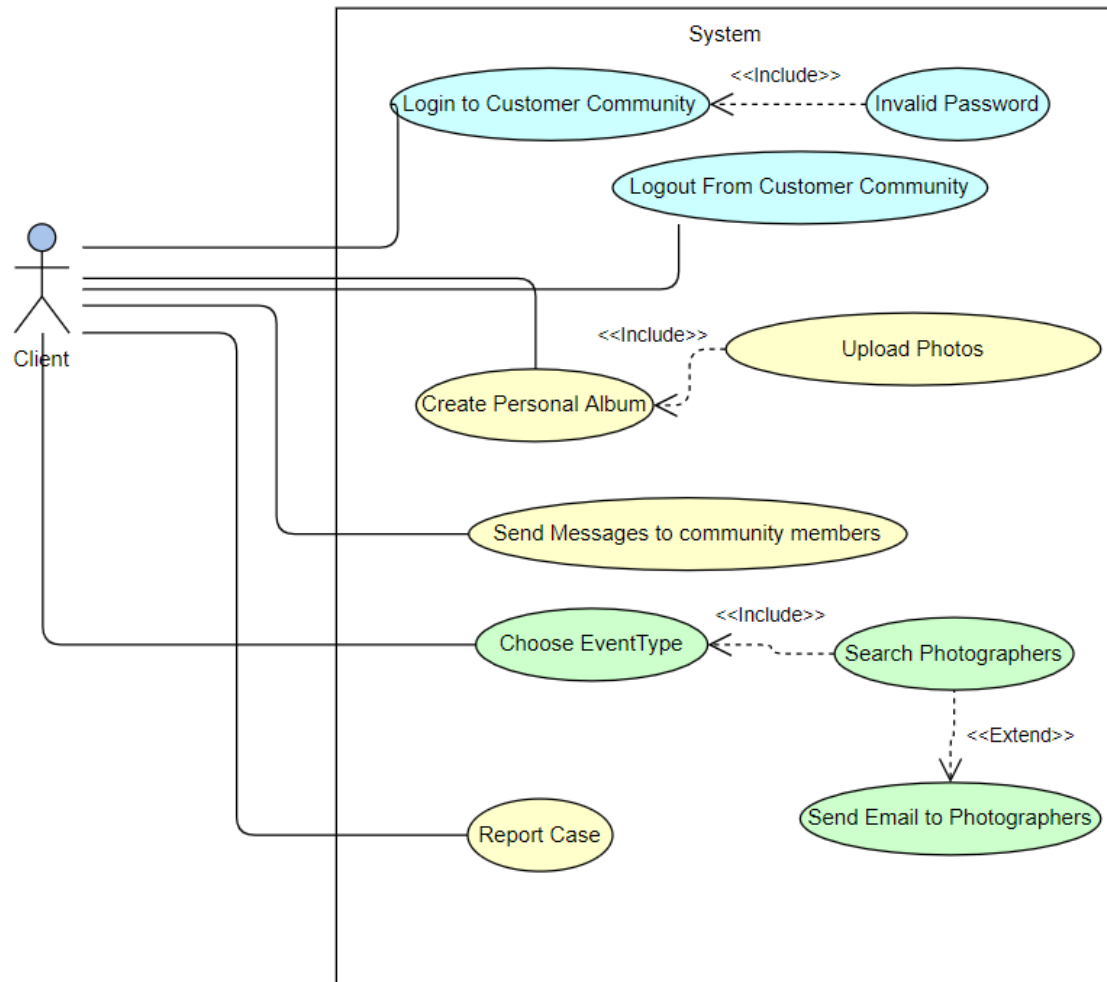


Figure 2 Client Use Case

### Photographer Use Case

The Use Case diagram of Photographer allows to Login and Logout to photographer community. Photographer also allow to create albums, send messages internally to community members, Create and manage hi personal clients, and report case if requires.

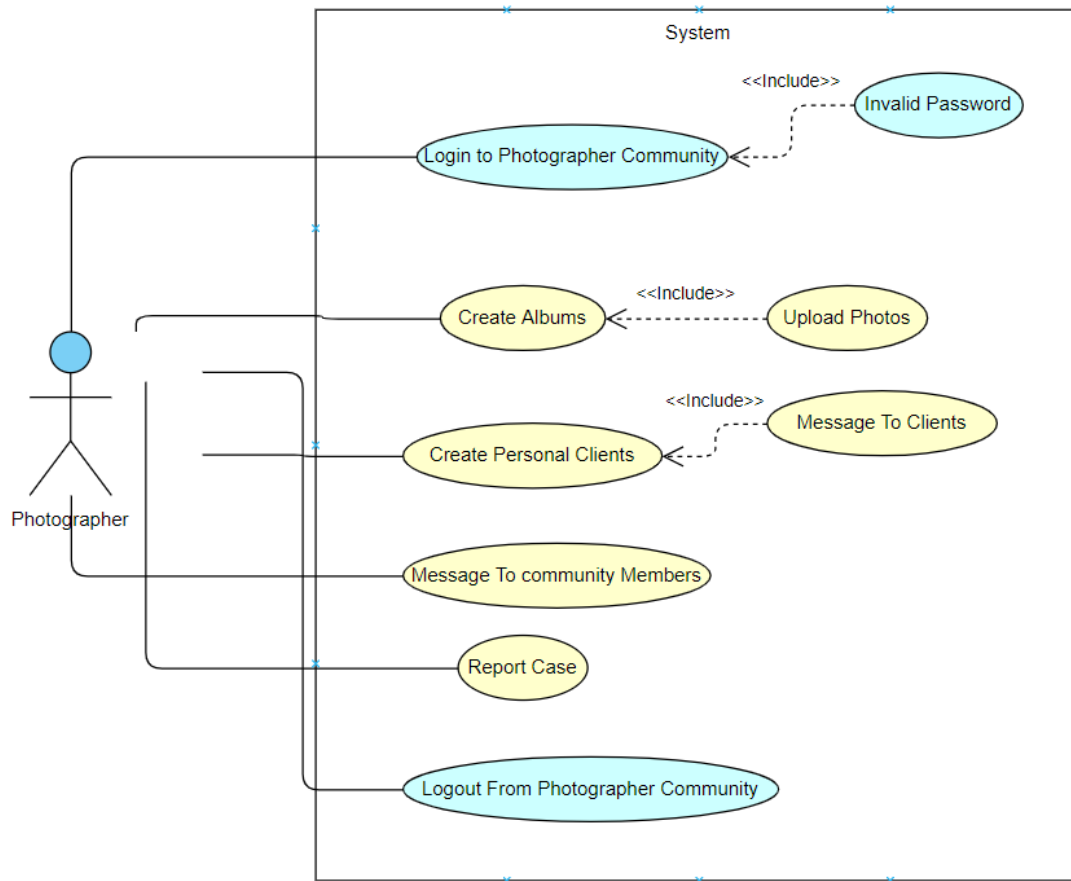


Figure 3 Photographer Use Case

## Flow Chart Diagram

### Search Photographer Flow Chart

The Flow Chart [Figure 4] provides glimpse of events performed by Clients to search a photographer which will best fit to their needs.

- Client needs to login to customer community with their valid login credentials.
- After successful login community allows client to choose the event for which he need photographer.
- If event type is not listed in drop down list, Client need to contact Intelligent Photo Service administrator to update the drop-down list.
- After successful choosing event and clicking get photographer button, community will show the highest ranked photographer for chosen event type.

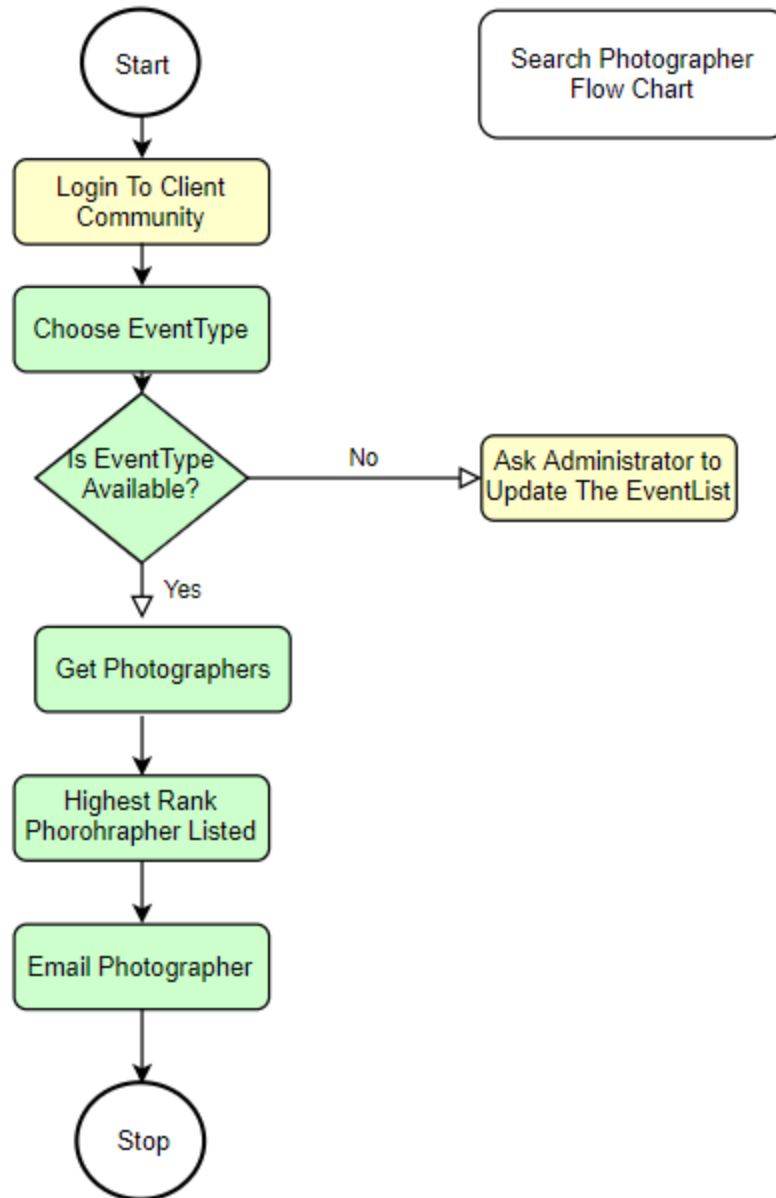


Figure 4 Search Photographer Flow Chart

### Image Prediction by Einstein Vision flow Chart

The flow Chart [Figure 3] provides representation of steps performed by photographer in Photographer community. The flow of events as follows:

- Intelligent photo service photographers allow to login to photographer's community by using their valid login credentials.
- If Photographer is not able to upload the images, he or she can immediately contact Intelligent Photo Service Business Administrator to give the corresponding permissions.

## Intelligent Photo Service

- After successful login, photographer can upload one or maximum ten photos at a time under Album tab by selecting related list of album tab.
- After uploading the event photos, the administrator side of application will run the Einstein Vision And prediction service which composed of trained dataset of event images will rank the photographer.
- After each photo upload the rank will increase by one.
- If uploaded photo failed to match with trained dataset, then the photographer rank will be same.

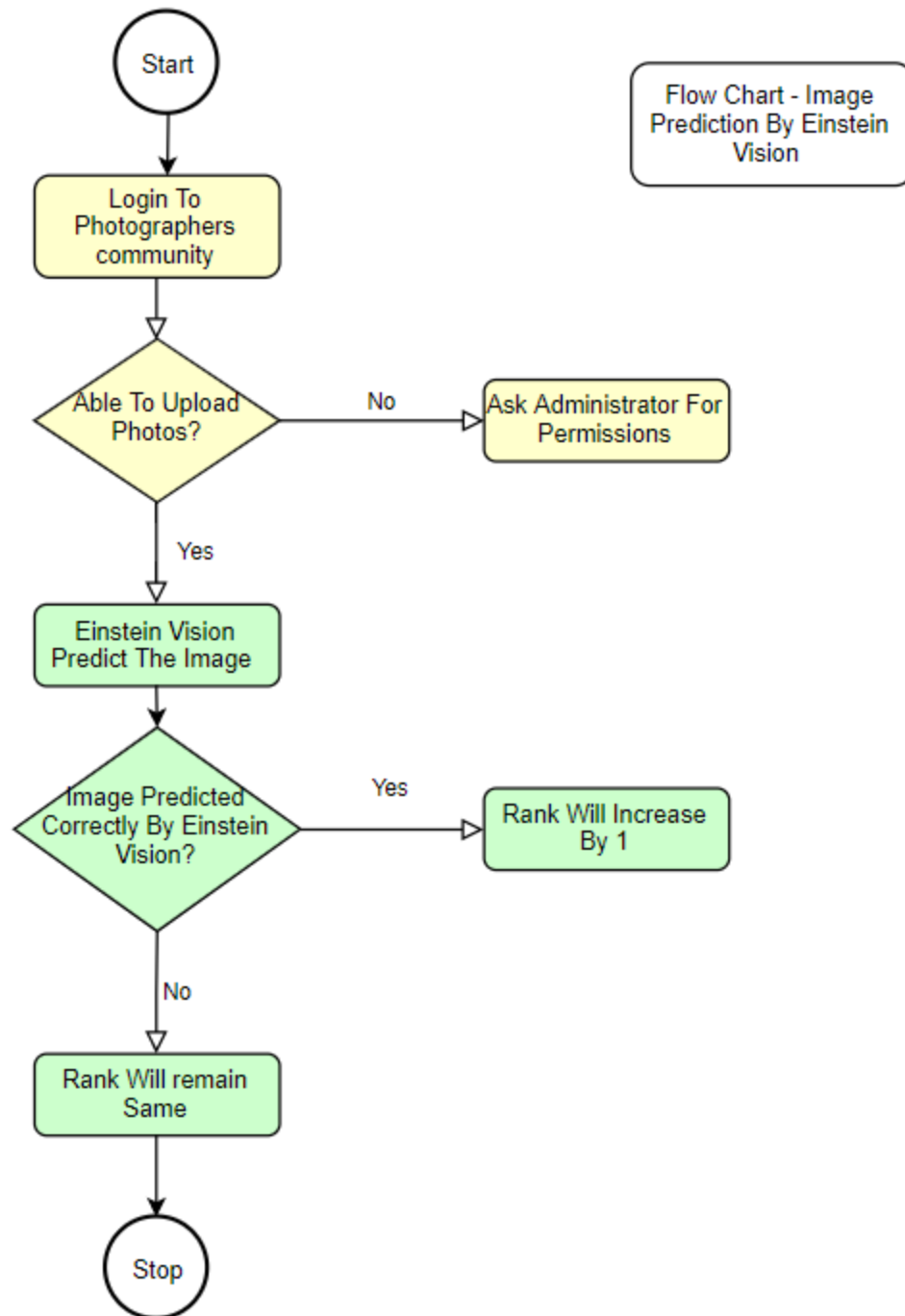


Figure 5 Einstein Vision Image Prediction Flow Chart



### Database Design

Lightning platform includes tightly coupled, strong cloud-based database which help to create powerful applications in a few clicks. By utilizing salesforce lightning point and click interface to create fields on standard object or store extra information in custom objects according to user needs.

The Lightning platform gives simple data storage. The Figure demonstrates the relationship and data structures between objects of Intelligent Photo Service. The Data model [Figure 8] concentrates on schema and relationships between objects. The project mainly uses Custom objects like Album, Clients, Event, Photo, Photographer as well as Standard objects namely, accounts, Contacts, Events, and User.

The many to many relationships created between photographer object and client object by using junction object EventName Object. Lookup relation is created in between Album object and Photographer object as well as album object and photo object. The Many to many relationships created between photographer and Event type object using junction object ranking.

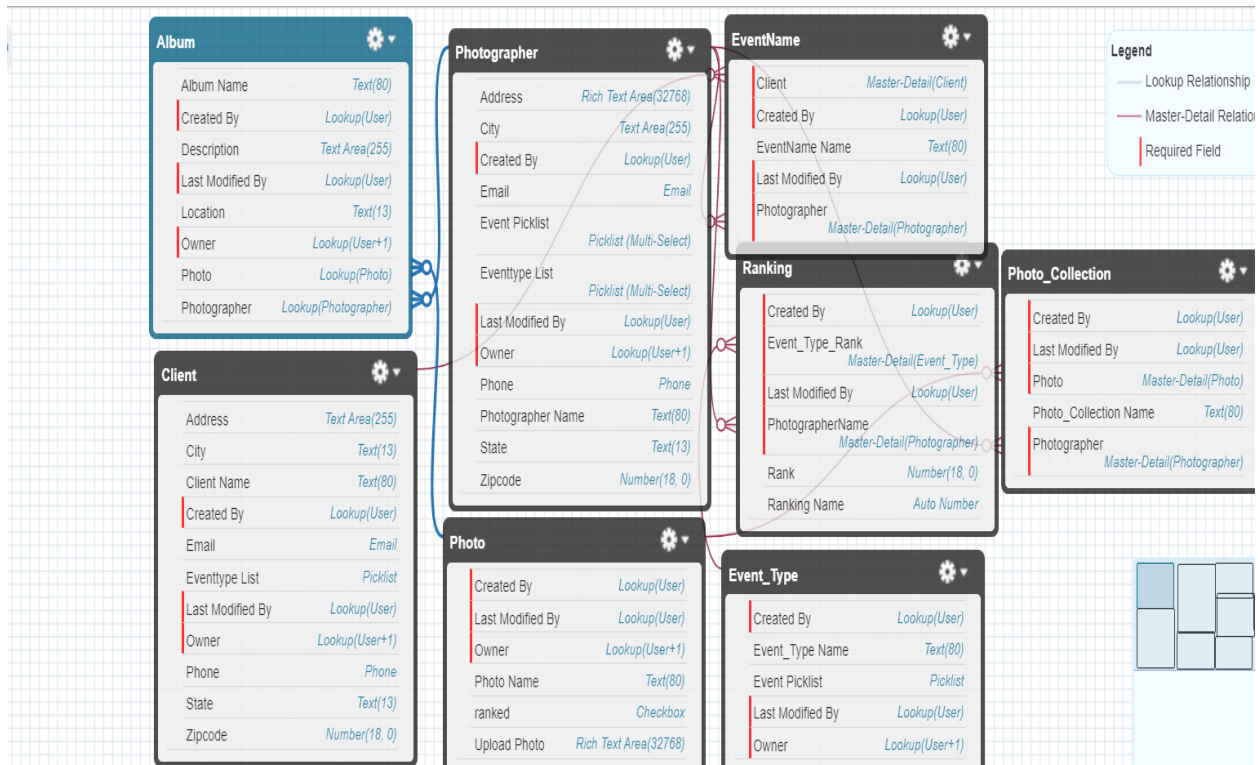


Figure 6 Data Model

### The Photographer Object Table

The Photographer object table includes Photographer Name, City, State, Address, Event type list custom fields.

*Table 1 Photographer Fields and Relationships*

## Fields & Relationships

12 Items, Sorted by Field Label

FIELD LABEL ▲	FIELD NAME	DATA TYPE
Address	Address__c	Rich Text Area(32768)
City	City__c	Text Area(255)
Created By	CreatedById	Lookup(User)
Email	Email__c	Email
Event Picklist	Event_Picklist__c	Picklist (Multi-Select)
Eventtype List	Eventtype_List__c	Picklist (Multi-Select)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Phone	Phone__c	Phone
Photographer Name	Name	Text(80)

## The Client Object Table

*Table 2 Client Fields and Relationships*

## Fields & Relationships

11 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Address	Address__c	Text Area(255)
City	City__c	Text(13)
Client Name	Name	Text(80)
Created By	CreatedById	Lookup(User)
Email	Email__c	Email
Eventtype List	Eventtype_List__c	Picklist
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Phone	Phone__c	Phone
State	State__c	Text(13)

## The Album Object Table

Table 3 Album Fields and Relationships

**Fields & Relationships**

8 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Album Name	Name	Text(80)
Created By	CreatedById	Lookup(User)
Description	Description__c	Text Area(255)
Last Modified By	LastModifiedById	Lookup(User)
Location	Location__c	Text(13)
Owner	OwnerId	Lookup(User,Group)
Photo	Photo__c	Lookup(Photo)
Photographer	Photographer__c	Lookup(Photographer)

**Event Type Table**

Table 4 Event Fields and Relationships

**Fields & Relationships**

5 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Event Picklist	Event_Picklist__c	Picklist
Event_Type Name	Name	Text(80)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)

## Ranking Table

*Table 5 Ranking Fields and Relationships*

### Fields & Relationships

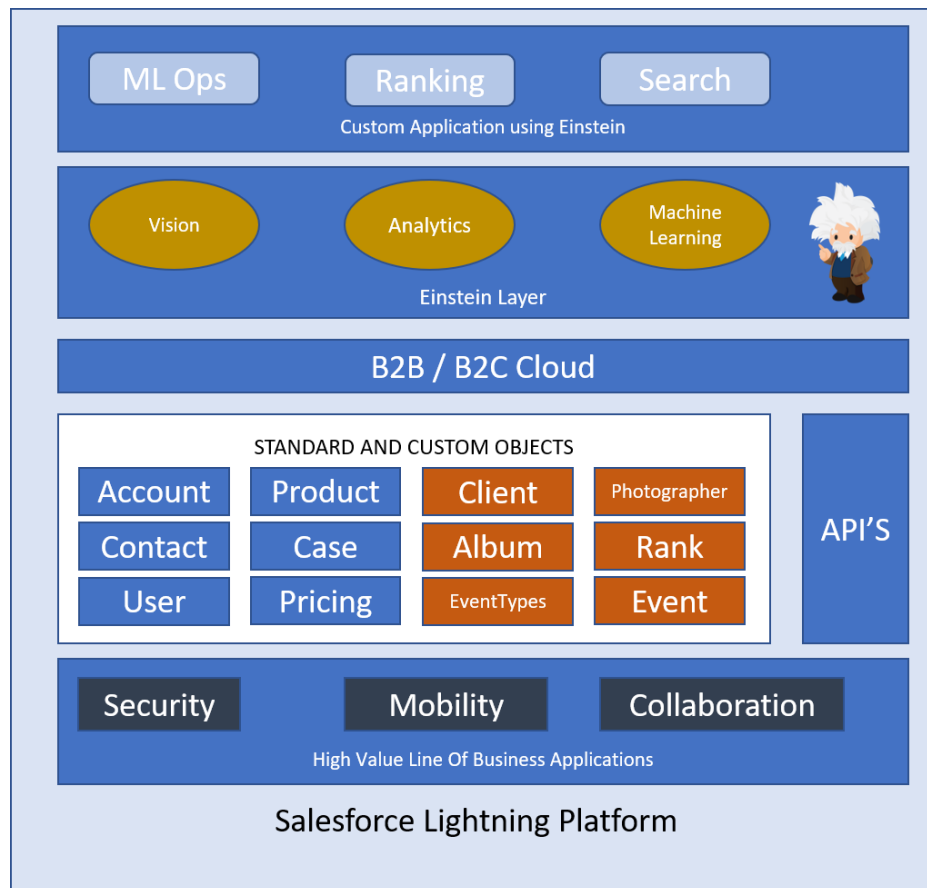
6 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Event_Type_Rank	Event_Type_Rank__c	Master-Detail(Event_Type)
Last Modified By	LastModifiedById	Lookup(User)
PhotographerName	PhotographerName__c	Master-Detail(Photographer)
Rank	Rank__c	Number(18, 0)
Ranking Name	Name	Auto Number

### Intelligent photo Service Architecture

Salesforce Is the Leading cloud-based company and provides SaaS and PaaS to many enterprises. Everything offered by Salesforce resides in multitenant trusted cloud. Salesforce stores customer data, provides process to nurture customers, gives ways to communicate and collaborate with people you work with.

Intelligent photo service application delivers highly customized user interface to customers, employees, and partners. IPS application architecture consists of series of layers that placed on top of each other's.



*Figure 7 Intelligent Photo Service Architecture*

The IPS sits on top of the platform. IPS offers two communities – Photographer community and client community both are resides in community cloud. Communities are online social platform for users of IPS. The important part is IPS Einstein model communicates with both communities to handle and manage user's data.

## Intelligent Photo Service

The most important layer is Data Model. IPS data model resides here. The data and metadata stored in Salesforce standard objects like Accounts, Contacts, cases, and custom objects namely, Photographer, Client, Album, Event type, Photo Collection.

The next layer is Platform microservices which includes many declarative approaches of an IPS like, Assignment rules, Flow Builder, validation rules, workflow processes, mobility and many more. These microservices are tiny and broadly decoupled little tasks to use in complex processes.

### **Einstein Vision API**

Einstein Vision is a crucial part of Einstein platform technology services. IPS utilizes the functionality of Einstein vision API to AI-enable the application. Einstein Vision provides many pre-built classifiers or users can build custom classifiers to concur the image recognition use cases. IPS incorporated the power of image recognition to IPS CRM and enhancing the functionality of an application.

The main components of EV - AI are Datasets, Label, Model, Training and Prediction.

#### **Datasets**

The trained data includes input and output and generate the model which will use to make predictions.

#### **Label**

A label references the output name which will predicted by the model and input name is the group of similar data.

#### **Model**

Machine Learning developers create and train the dataset to predict the respected outcome. When developer train the dataset, the system will determine the similarities and differences between the multiple labels to categorize and characterize to define the label.

#### **Training**

It is a process in which model is created and learns the classification rules from given dataset.

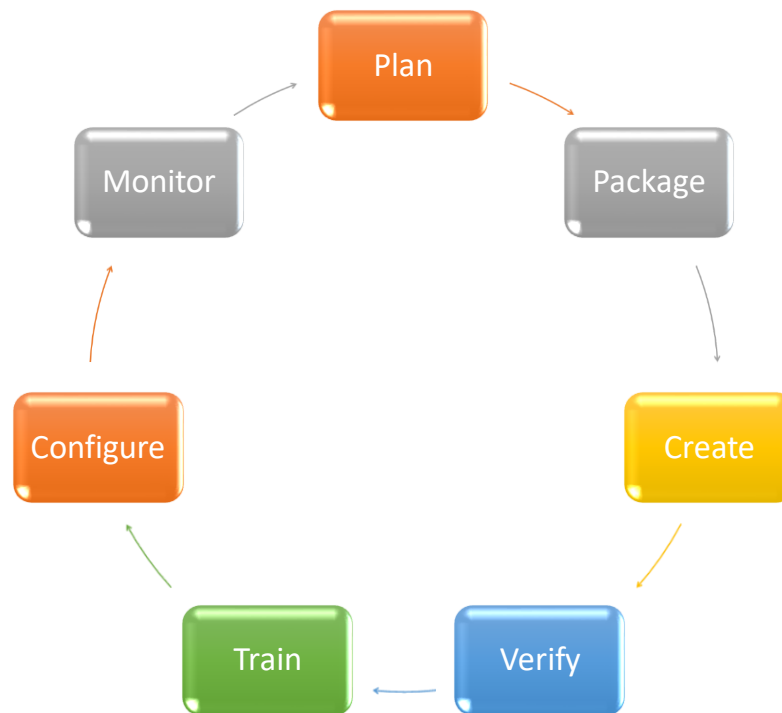
#### **Prediction**

The outcome generated by system is useful to identify how close the input data matches to the predicted result.



### Machine Learning Life Cycle [MLOps]

Intelligent Photo service application follows MLLC periodic or recurring process which involves Plan, Package, Create, Verify, Train, Configure and Monitor Phases.



*Figure 8 Machine Learning Life Cycle*

There are six phases of Machine Learning life cycle.

#### **Plan**

The plan phase includes the process of defining the event types which are used to rank the photographers. Its IPS responsibility to update the event type list regularly or based on requirements. Currently Event type list consists of Party Images, Wedding images and sports images.

#### **Package**

After finalizing Data or event images in plan phase, its required to create a package. This phase includes name the images according to standards, arranging images in proper format and finally convert folder or package into zip file. [Salesforce. (n.d.). *Einstein ai*. Retrieved from Salesforce Einstein ai: <https://einstein.ai/research>]

#### **Create**

## Intelligent Photo Service

This phase is crucial as it allows to generate the valid Zip URL of the dataset folder. It's necessary to provide the path to the zip file. To generate standard URL and provide the access from external hosted site into Salesforce is the most important stage of the life cycle.

IPS event list zip folder hosted on GitHub and used later in Einstein Vision Custom Component located at Lightning Welcome page of an application.

*Table 6 GitHub Hosted a Zip file Of Event List Images*

9 commits

2 branches

0 packages

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

mhaskeshradha1 medical dataset

Latest commit ae193b2 35 minutes ago

3Col-csv (8).csv	Add files via upload	12 months ago
Cats.zip	dataset files	23 days ago
EVENTNEW.ZIP	event new	22 days ago
EventLists.zip	dataset files	23 days ago
Eventnew github images.zip	Add files via upload	22 days ago
Eventphotos.zip	raw files	23 days ago
Events.zip	dataset files	23 days ago
Medical.zip	medical dataset	35 minutes ago
README.md	Update README.md	16 months ago
across.txt	Add files via upload	12 months ago
mountainvsbeach.zip	dataset files	23 days ago

## Verify

After Successfully finishing the Create phase, the administrator needs to click the Create Dataset button. If zip URL format meets the standard of salesforce, the dataset will be created with his

name.

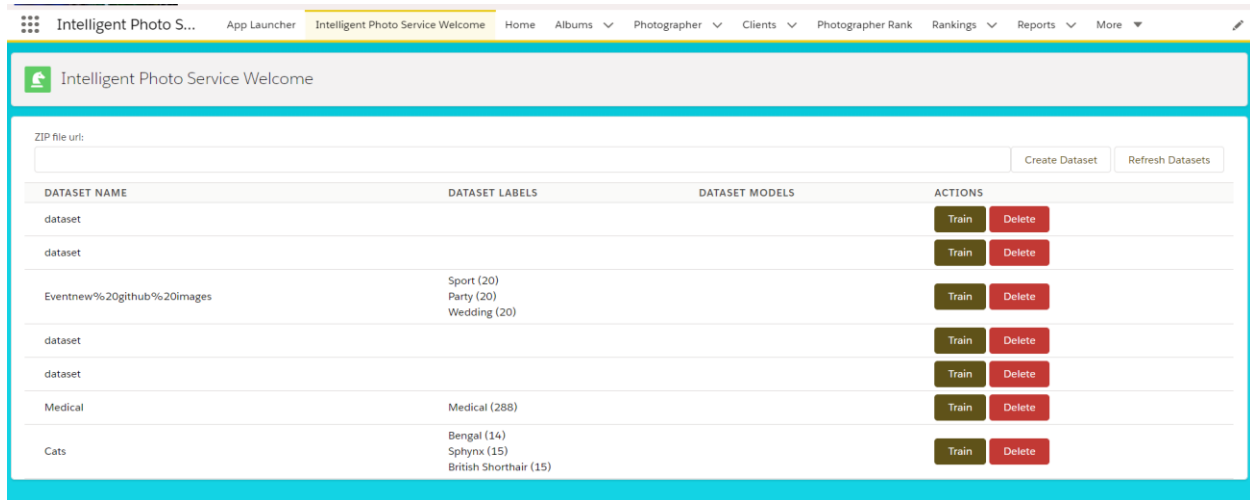


Figure 9 Verify Dataset

## Train

IPS trained the EventList dataset which includes twenty images of Wedding, twenty images of party and twenty images of Sports. After inserting zip URL and clicking create dataset button, the dataset will be created, and later business administrator will train the dataset by clicking Train button from Intelligent Photo Service Welcome lightning page. It is the responsibility of IPS to continuously create and update the dataset and trained them.

## Configure

In IPS, salesforce developer and administrators can create and trained multiple datasets but only one dataset will be active at a time. Currently, four datasets are generated and trained successfully. For this IPS application, EventList Dataset activated from Custom Settings of salesforce setup. It means currently, default organizational level value is name of EventList dataset.

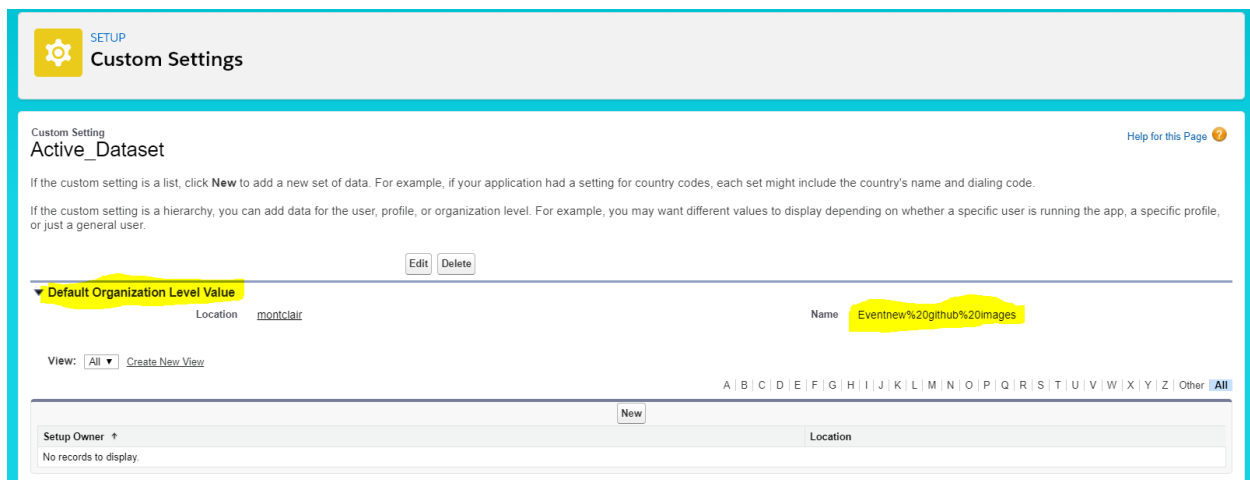
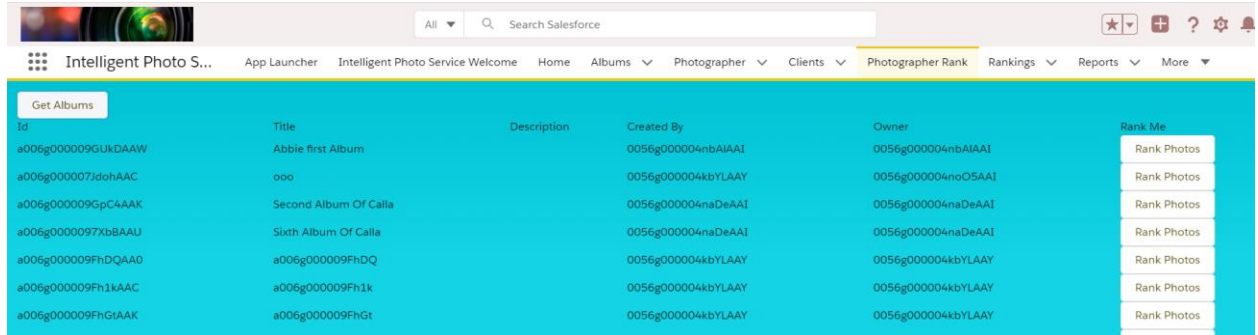


Figure 10 Custom Settings - one default dataset allowed to activate in Salesforce

## Monitor

Its IPS administrator and developer's responsibility to continuously monitor dataset and MLLC. If client or photographer requested to update EventList, it will be updated in picklist of data model and the respected dataset folder. Business administrator will rank the photographers daily to update the rank as well as to monitor that whether the ranking is updated for specific event type.



The screenshot shows the Salesforce interface for the Intelligent Photo Service. The top navigation bar includes the Salesforce logo, a search bar, and various utility icons. The main navigation menu is visible, with the 'Photographer Rank' tab selected. Below the navigation bar, there is a table with columns: Id, Title, Description, Created By, Owner, and Rank Me. The table contains several rows of data, each with a 'Rank Photos' button in the 'Rank Me' column.

Id	Title	Description	Created By	Owner	Rank Me
a006g000009GUKDAAW	Abbie first Album		0056g000004nbAIAAI	0056g000004nbAIAAI	Rank Photos
a006g000007IdohAAC	ooo		0056g000004kbYLAAY	0056g000004noO5AAI	Rank Photos
a006g000009GpC4AAK	Second Album Of Calla		0056g000004naDeAAI	0056g000004naDeAAI	Rank Photos
a006g0000097XtbBAAU	Sixth Album Of Calla		0056g000004naDeAAI	0056g000004naDeAAI	Rank Photos
a006g000009FhDQAA0	a006g000009FhDQ		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009Fh1kAAC	a006g000009Fh1k		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009FhGIAAK	a006g000009FhGt		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos

*Figure 11 monitor the MLLC and rank the Photographer*

### **Data Analysis and Visualization**

Data analysis and data visualization both terms come side by side in the data community. Data analysis is an exploratory process which always initializes questions. Data visualization is the visual or graphical representation of data. It involves visualizing a data using single chart or complex dashboards.

In this project I have used Salesforce Einstein analytics for data analysis and visualization. Einstein is the smart CRM assistant for salesforce professionals. Using Einstein salesforce professionals able to change and apply AI technology to interact with customers.

### **Artificial intelligence**

Artificial intelligence gives various functionalities like personal recommendations, intelligent search, and results, and performs task automation. It tremendously affects companies to sell, service and customers. For example, in salesforce AI can automatically log customer data, sales activities, can suggest email responses and next best actions.

### **Einstein Platform**

Salesforce Einstein platform provides tools to salesforce professionals to create and build a custom assistant for their business. Einstein platform includes powerful tools like voice input, voice output, natural language processing, intelligent prediction which helps to better interact with clients.

Most important part is that Einstein allows every salesforce user to discover, predict, recommend, and automate functionalities based on their needs.

### **Data and Salesforce Einstein**

External Data and datasets can be imported into salesforce analytics using .csv files and analytics will automatically create metadata, or you can upload data using user interface or external data API to create datasets.

In this project I have extracted data from a salesforce developer organization using data loader and then imported into salesforce Einstein analytics platform through .csv files.

Photographer object and client object mock datasets are also tested against real data.

Some of the predictions built by using salesforce Einstein analytics:

Clients per States donut chart gives visual representation of total clients count per state. As per the column chart California and Texas states having highest client's number- 123. It also illustrates the total client's number in selected states. As in California states client's number is increasing while in Kentucky states client number is minimal. Intelligent photo Service administrators can put a lot of effort into growing their business in Kansas, Kentucky after observing figure 14. Furthermore, It also provides glance view of the total number of photographers per state. As per the column chart currently Maximum photographers are serving in California. As client numbers are also increasing in California intelligent Photo service needs

## Intelligent Photo Service

more photographers in California to fulfill needs. Also, IPS salesforce professionals also observe the ratio of clients and photographers in Texas state. Client total number is increasing but photographers' total number is less than client total number.

By taking look at following, Intelligent photo service administrators can easily predict that their photo service business expanding rapidly in California and Texas.

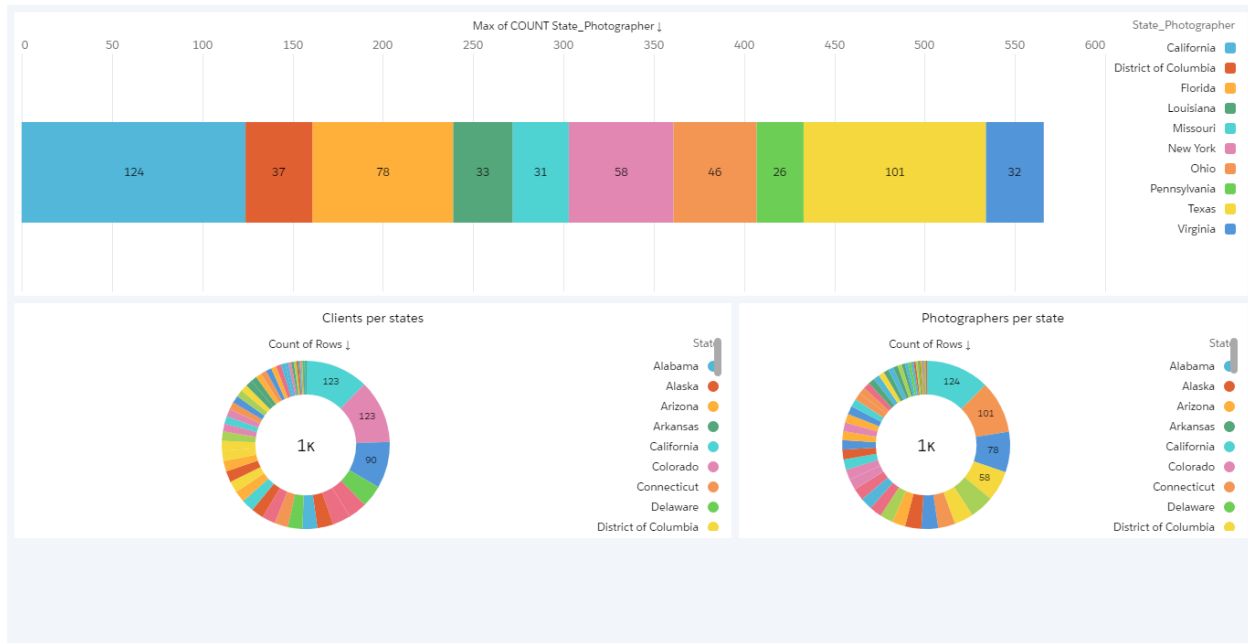


Figure 12 Dashboard- Data Visualization and analysis

## Intelligent Photo Service

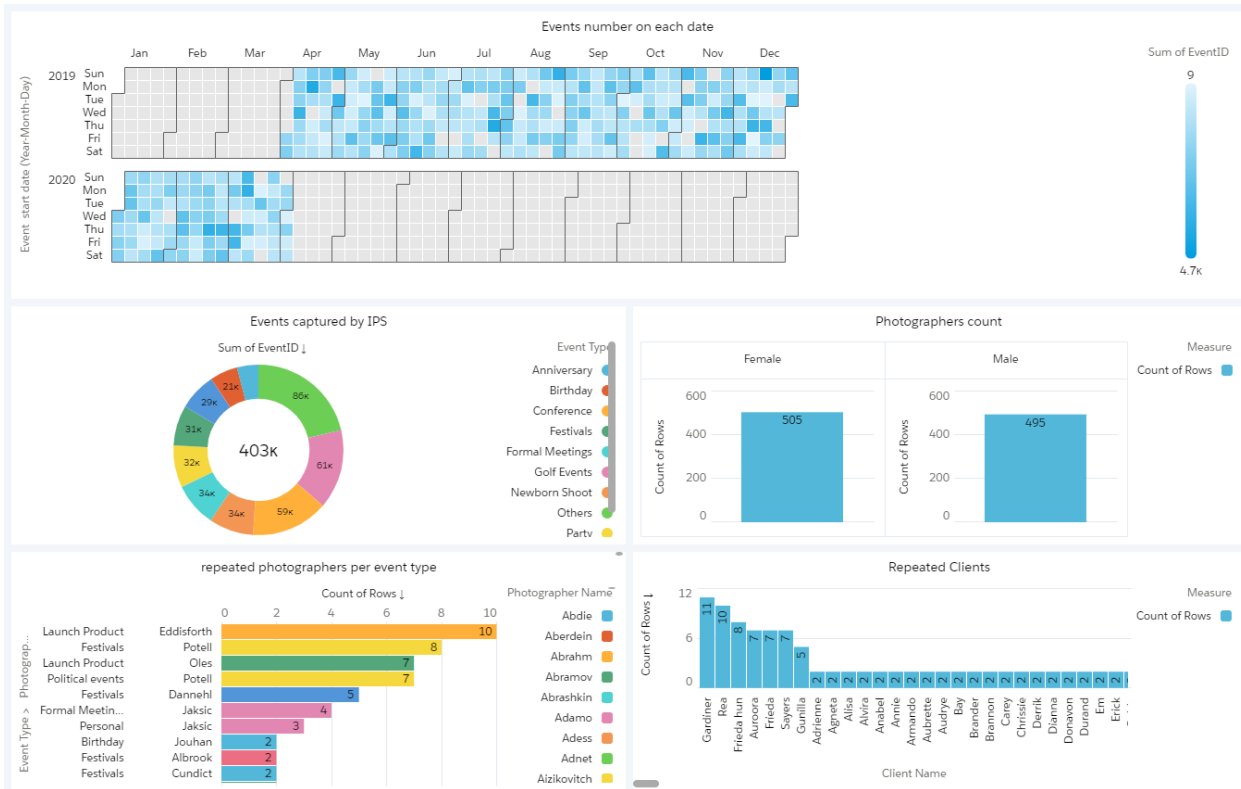


Figure 13 Data Visualization and Analysis Continued...

## Interface Design

The Intelligent photo Service provides enhanced user interface to their customers – Photographers and Clients. The business administrator created and manages two communities namely Photographer Community and Client community.

### Business Administrator Workspace

Business Administrator is the main key of Intelligent photo service application. Firstly, one user needs to register and login to salesforce developer organization. After login, the user will automatically be assigned as a system administrator. Administrator allow utilizing the System administrator profile and Salesforce user license. He / she allow to perform all administrative task namely, create, edit, delete, update all applications, tabs, custom objects, few standard objects, manage and maintain profiles and permission sets, create and nurture salesforce communities.

### Home Page

The administrator home page includes the Machine learning data model aura component. The Data model consists of create dataset button, zip file url textbox, train and delete button. Administrator uploaded and trained event image dataset and continuously updating the trained dataset.

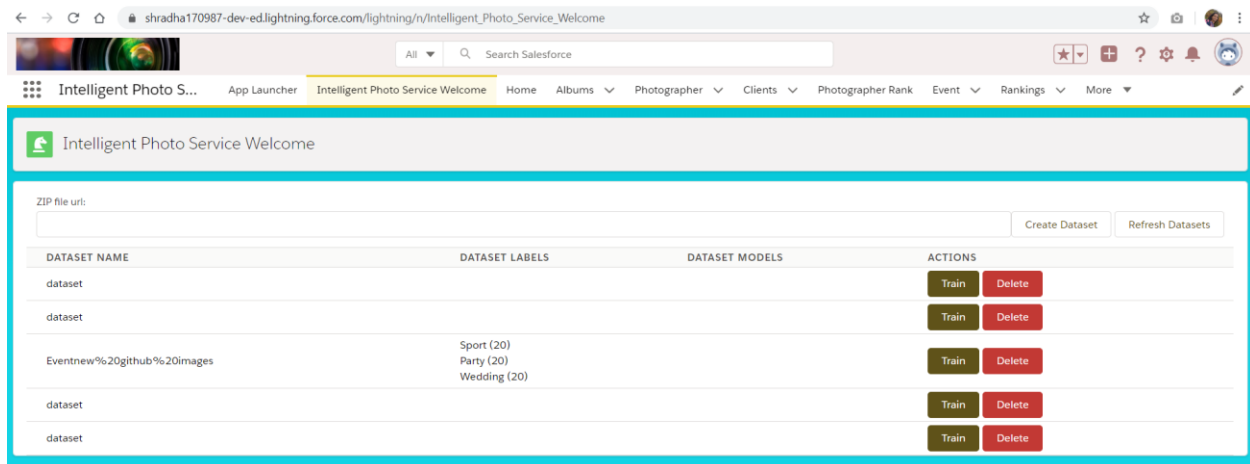


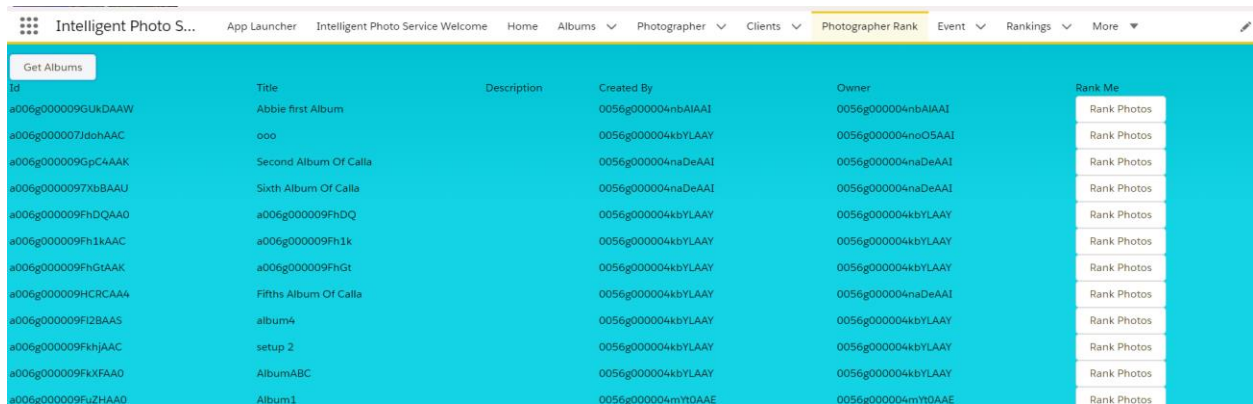
Figure 14 Business Administrator lightning welcome page

### Photographer Rank Page

Photographer rank page maintained by administrator. Administrator should be ranked the photographer daily by clicking Rank Photo button. The rank will be updated of all photographers who uploaded event images. The rank will be maintained in Photographer as well as Ranking object.



## Intelligent Photo Service

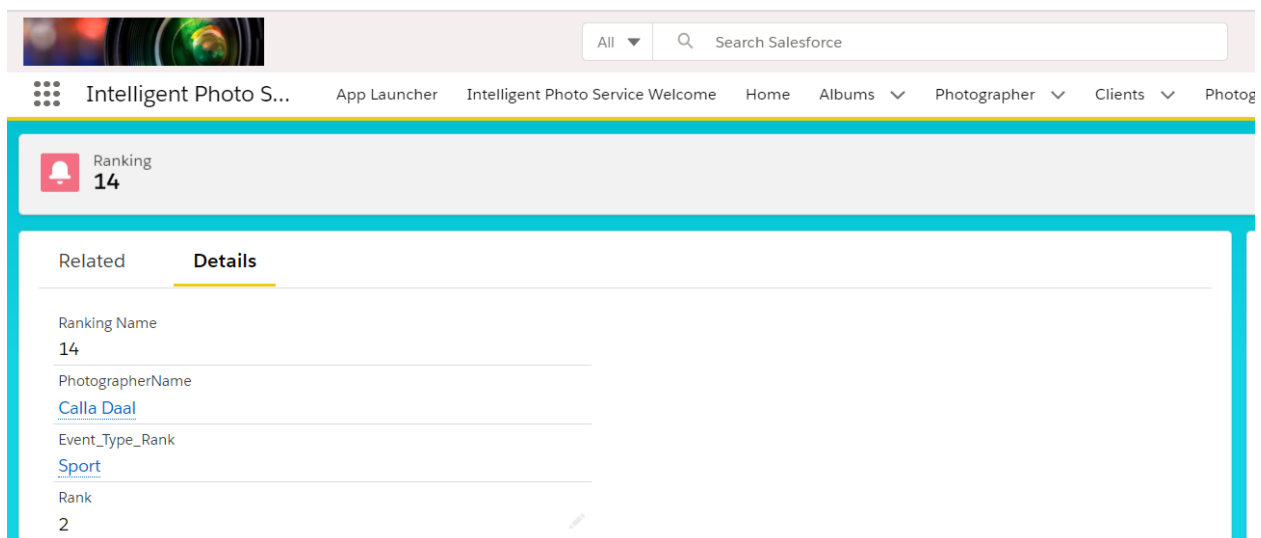


Id	Title	Description	Created By	Owner	Rank Me
a006g000009GUKDAAW	Abbie first Album		0056g000004nbAIAAI	0056g000004nbAIAAI	Rank Photos
a006g000007JdsHAAc	ooo		0056g000004kbYLAAY	0056g000004noQ5AAI	Rank Photos
a006g000009GpCAAAK	Second Album Of Calla		0056g000004naDeAAI	0056g000004naDeAAI	Rank Photos
a006g0000097Yb8AAU	Sixth Album Of Calla		0056g000004naDeAAI	0056g000004naDeAAI	Rank Photos
a006g000009FhDQA0	a006g000009FhDQ		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009Fh1kAAC	a006g000009Fh1k		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009FhGIAAK	a006g000009FhGt		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009HCRCAA4	Fifths Album Of Calla		0056g000004kbYLAAY	0056g000004naDeAAI	Rank Photos
a006g000009Fi2BAAS	album4		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009FkhJAAC	setup 2		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009FkIFAAD	AlbumABC		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009FuZHAAD	Album1		0056g000004mY0AAE	0056g000004mY0AAE	Rank Photos

Figure 15 Rank Photographer Component Page

## Ranking Page

The ranking page displays the rank of photographer along with general details like, photographer name, event type rank.



Related	Details
Ranking Name	14
PhotographerName	<a href="#">Calla Daal</a>
Event_Type_Rank	<a href="#">Sport</a>
Rank	2

Figure 16 Rank Detail Page

## Mobile Application

Administrator also gets mobile application from where he can access major functionalities.

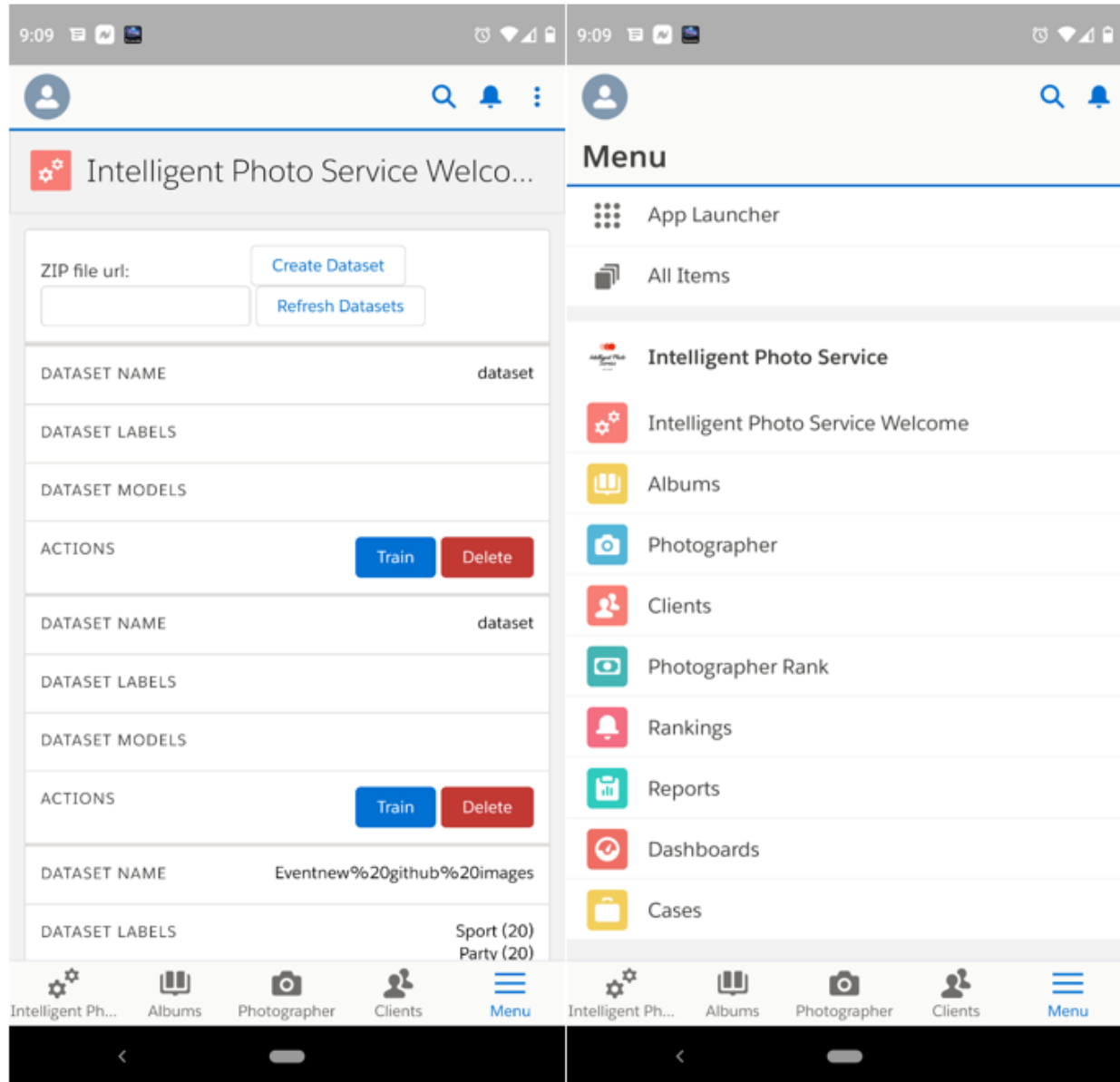
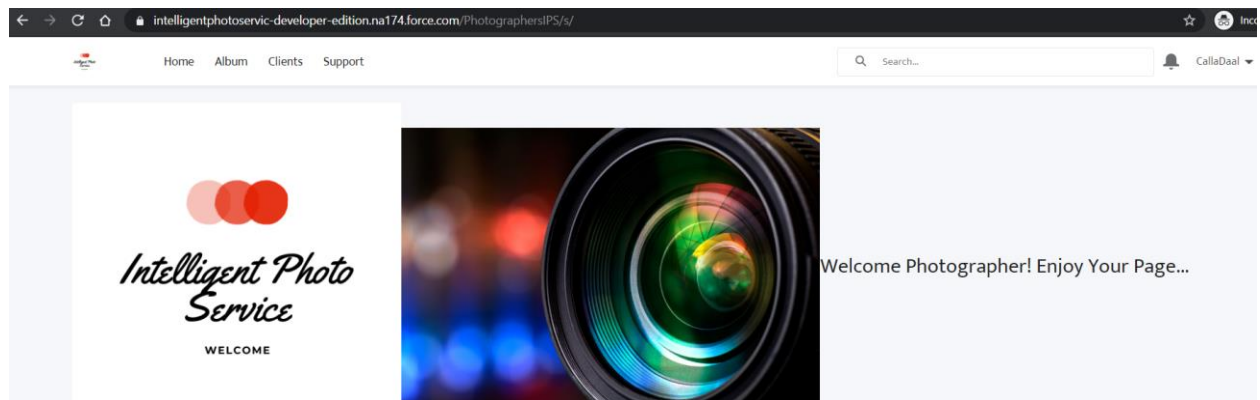


Figure 17 Mobile Application for Administrator

## Photographer Community

The Photographers allows to register and login to the photographer community using their login credentials provided by Intelligent photo service business administrator. Photographer allow to create a new album or update the existing album under Album tab. The event images can be uploaded by Photographer under related list tab of album.

## Home Page



*Figure 18 Home Page – Photographer Community*

### **Album Page**

This page allows Photographer to create new album. It facilitates photographer to list and maintain their personal albums too. Each newly created album has two pages namely, Detail Record page and Related list page. Detail record page displays the detailed information of an album while Related list page allows to upload new event images and helps to creates new collections of photos.

### **Event Images Uploaded by Photographer**

## Intelligent Photo Service

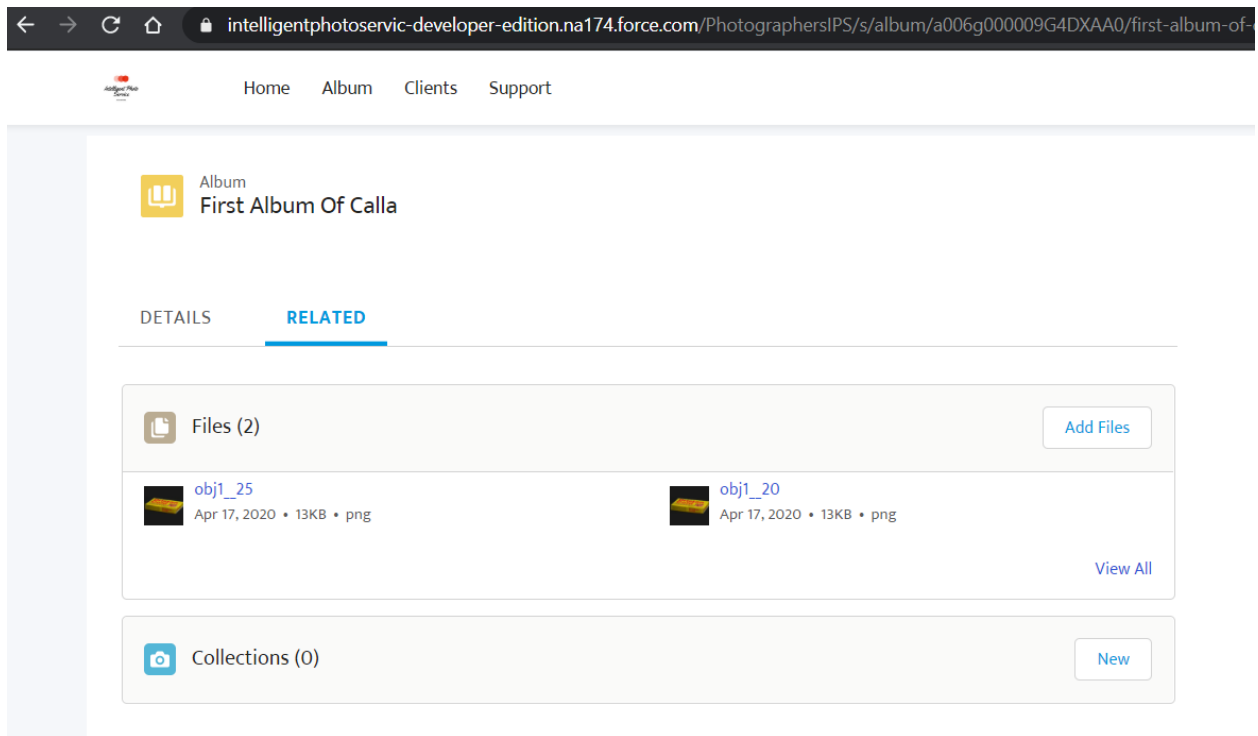


Figure 19 Upload Images

## Case / Support Page

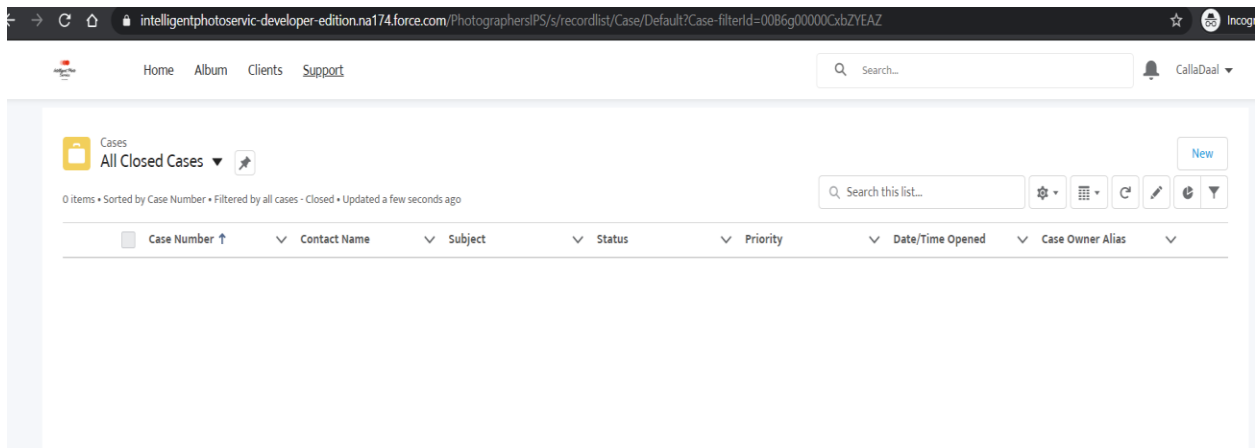


Figure 20 Raise Case

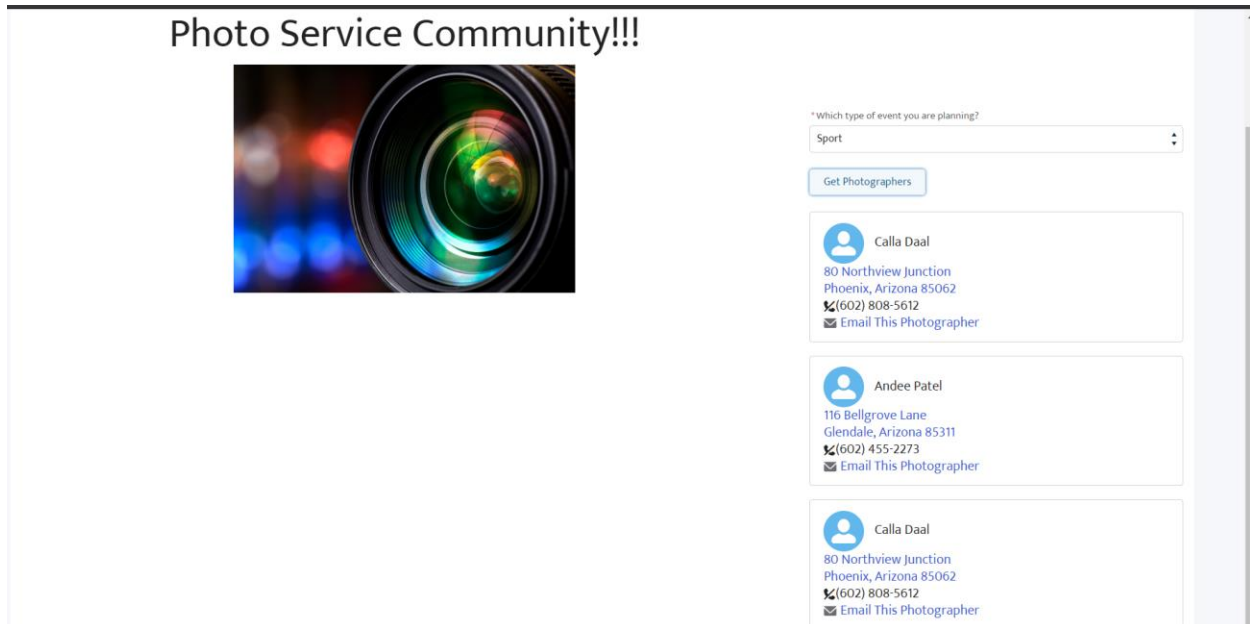
## Customer Community

The client allows to register and login to the Customer community using their login credentials provided by Intelligent photo service business administrator. Client allow to create a new album or update the existing album under Album tab. Client also able to find the highest rank photographer for specific event.

## Home Page

The home page of customer community allows clients to choose the event. After selecting event the community will automatically provide highest ranked photographers from Intelligent photo service application.

The highest ranked photographers listed for Sports event. The client able to call or email the photographers for further process.



*Figure 21 Get Highest Skilled Photographer*

## Case / Support Page

From Case or support page client allow to ask about any support or generate case/help to the Intelligent Photo Service administrator. It is similar functionality as mentioned and demonstrated in Figure 22.

## **Test design**

### **Introduction**

The lightning platform requires minimum 78% test coverage of apex class from unit test to deploy the code from developer or any organization to production environment. As a tester you should not rely on 75% test coverage instead need to strike more percentage of unit tests. Salesforce testing is a validation and customization of Salesforce Vanilla SDLC. Salesforce platform functionalities are mainly built on Apex and it also has many pre-built test cases to perform testing on salesforce components, lightning components, and point-click declarative interface. Developer and tester should use the sandbox environment to test the business processes. Later it will be migrated to the production environment.

### **Types of Testing in Salesforce**

#### **Unit Testing**

Unit testing performed by APEX developers. It consists of writing clauses in apex code which automatically calculate testing coverage. From test coverage, Apex developers can easily evaluate how many data records are affected and further decide whether to push the code in the production environment or not.

#### **System Testing**

System testing performed by salesforce consultants to test the technical processes from initial point till end. It involves many processes like workflow rules, approval processes, flow, or process builder to troubleshoot the flaws present in the system.

#### **User Acceptance Testing**

It is conducted by end users of the system. It allows us to test the business processes of the system. The end user needs to confirm that business processes fit to the organization according to their needs.

#### **Production Testing**

Production testing is like system testing in production environments. It allows you to test whether the code and configuration have been correctly deployed in the production environment.

## Test Screens

The screenshot shows the 'Intelligent Photo Service' web application. At the top, there is a navigation bar with a logo, the text 'Intelligent Photo S...', and a dropdown menu set to 'All'. Below the navigation bar is a 'Welcome' message. The main content area features a 'ZIP file url:' input field and a 'Create Dataset' button. Below this is a table with the following data:

DATASET NAME	DATASET LABELS	DATASET MODELS	ACTIONS
dataset			<button>Train</button> <button>Delete</button>
dataset			<button>Train</button> <button>Delete</button>
Eventnew%20github%20images	Sport (20) Party (20) Wedding (20)		<button>Train</button> <button>Delete</button>
dataset			<button>Train</button> <button>Delete</button>
dataset			<button>Train</button> <button>Delete</button>

Figure 22 Page validation - Insert URL to create dataset

## New Photographer

Review the errors on this page.

These required fields must be completed: Photographer Name

## Information

\* Photographer Name

Complete this field

Owner

Shradha Mhaske

Address

The screenshot shows the SLDS toolbar with the following elements:

- Font family: Salesforce Sans
- Font size: 12
- Text formatting: Bold (B), Italic (I), Underline (U), Strikethrough (ABC)
- List creation: Bulleted list (≡), Numbered list (1≡), Table of contents (≡➔)
- Link insertion (🔗)
- Image insertion (🖼️)
- Text color selection (I<sub>x</sub>)

Cancel

Save &amp; New

Save

Figure 23 Form Validation - Insert Photographer name to save New Photographer

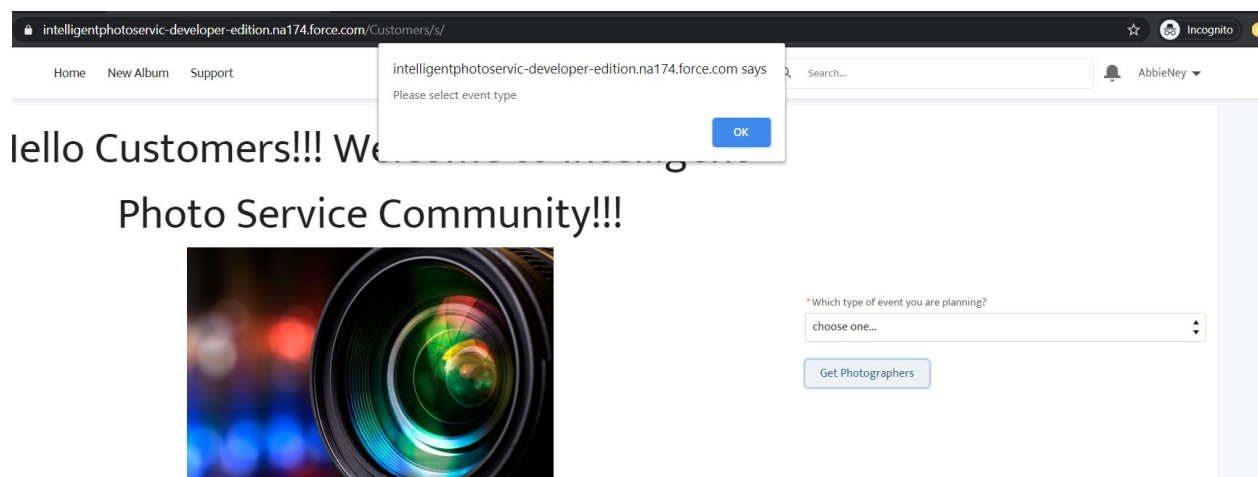


Figure 24 Page Validation - Choose event type to get photographer list



### **Future Enhancement**

Intelligent photo service application is the cloud based and developed on lightning platform of salesforce. This application is the combination of a pre-existing and newly created application which covers and gives solutions to basic as well as latest requirements. There are many features and functionalities are integrated in current application, but project scope has been constrained to resolve the difficulties as identified in problem areas.

The project objective is to provide Photography services like enhanced User Interface, store client and photographers data, updated event list to photographers as well as clients. The main aim is to find highest skilled photographer for event or occasion using artificial Intelligence. The client and photographer have their personal communities where both users need to login and maintain their data.

Both communities are currently handled by business administrator of Intelligent photo service. Business administrator also able to analyze the business needs and can track business growth by utilizing the functionality of dashboards. The client able to find highest skilled photographer from customer community. In Present system the main aim is fulfilled with some limitations.

The main limitation is the photographer rank will be increased only by clicking the rank photos button by administrator. Business administrator need to accomplish this task manually. In future, this ranking process will be performed to schedule automatically.

Next, the photographer rank would increase only after uploading the event images example – if photographer attended and clicked Party event then he should be uploaded only standard party image provided by Business administrator. In future, the development will be more focused on security and authenticity of photographer whether he truly attended the event or not. Furthermore, later in development process administrator will be able to track the total count of events attended by individual photographer.

The current development of Intelligent photo service is completed in developer edition of salesforce. In later process, the full application will be migrated to the enterprise level edition to overcome the restrictions of user licenses and application access usage like currently, an application only store 20MB size images, and in enterprise level edition will support up to 10GB.

Currently, Intelligent Photo service utilizing Einstein Vision API which is freely available in market. The limit of API call is only 2000 times per month. In future the application will capture the paid version of an Einstein Vision API to generate unlimited calls.

In future, an application will give more focus on attracting clients and photographers to increase the business of an application by using external systems such as creating and organizing campaigns, website invites, and salesforce marketing cloud implementation.

## User Manual

Intelligent Photo Service is a salesforce cloud-based application will effectively run on different devices including Mobile devices and tablet devices. There are basically three users – Business administrator, Photographer and Client.

### Contact Information

Application developed by – Shradha Mhaske

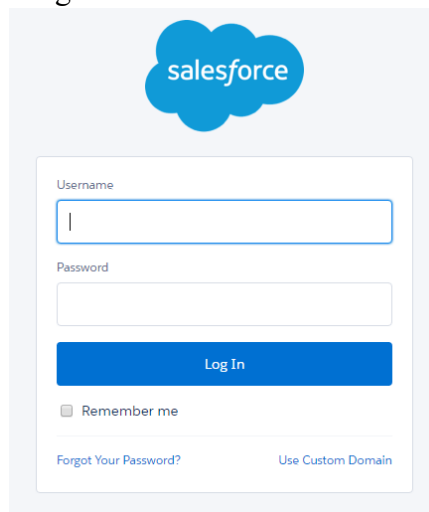
Email [mhaskes1@montclair.edu](mailto:mhaskes1@montclair.edu)

Instructor Dr. John Jenq.

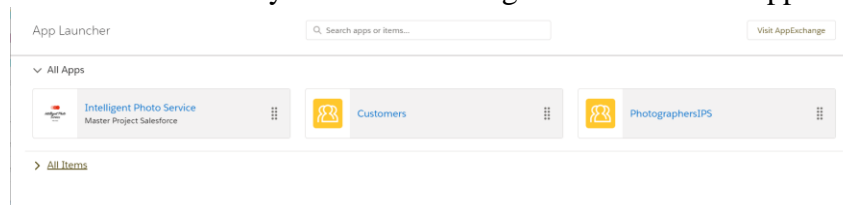
### Getting Started

#### Business Administrator User Manual

- Login to Salesforce Developer Organization by using link - <https://login.salesforce.com/>
- Using credentials given by Intelligent Photo Service Application Owner “Shradha Mhaske”, login to Salesforce organization.

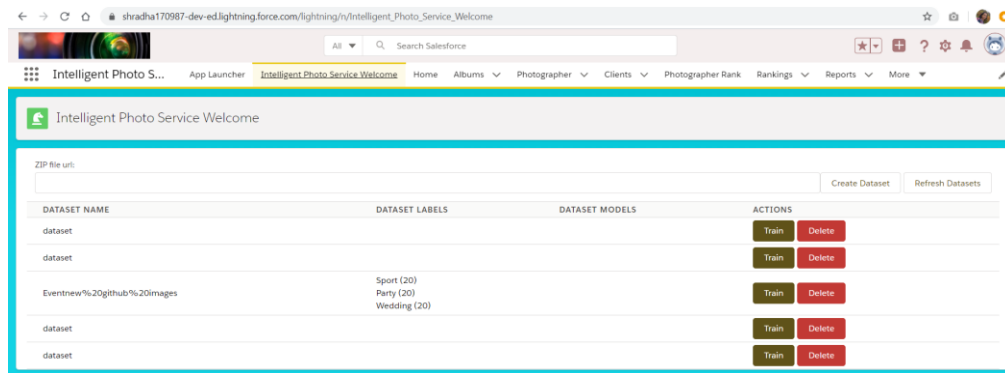


- Enter Username and Password. Click Log In button to enter sIntelligent Photo service application.
- Click App Launcher icon. Now you have in Intelligent Photo Service application wizard.

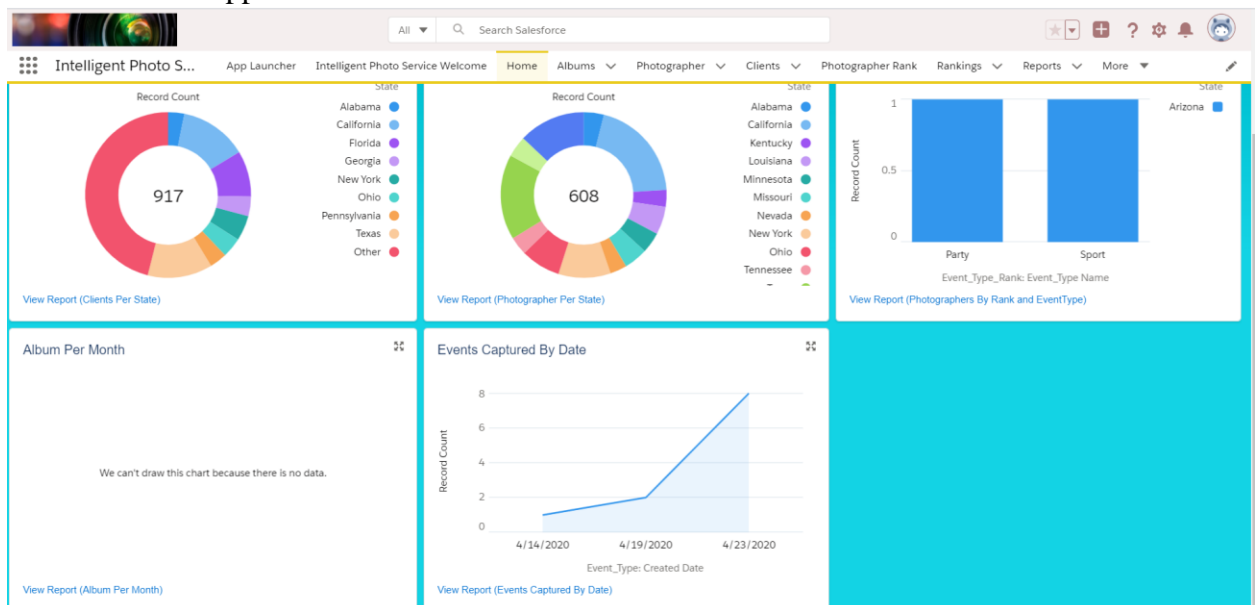


- Click Intelligent Photo Service.
- After Clicking Intelligent Photo Service application, you will see the landing page of ana application.

## Intelligent Photo Service

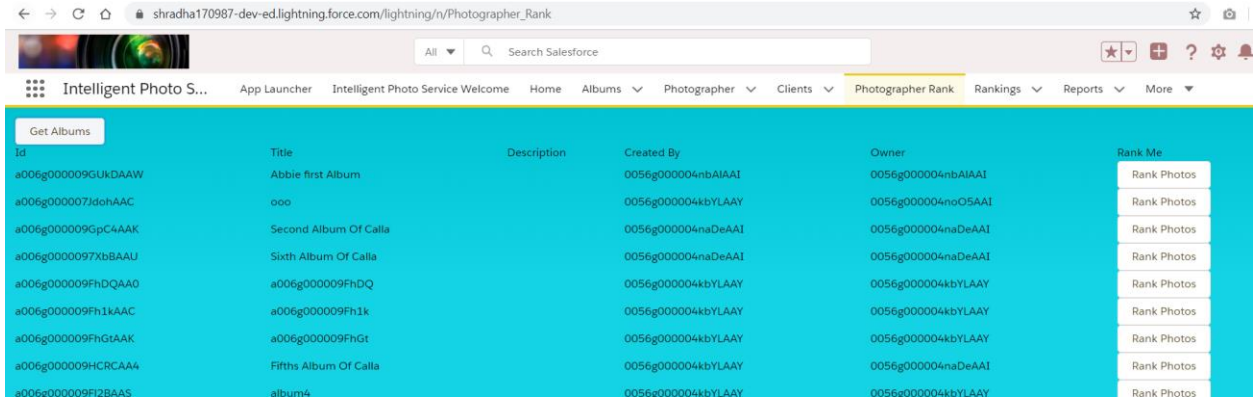


- To train the dataset, first create dataset. Then Make URL and paste it in ZIP file URL box. Click Create dataset.
- Train dataset by clicking Train Button.
- If you want to delete dataset, click delete button.
- To see weekly progress of an Intelligent photo service business, go to Home Tab. Dashboard will appear.



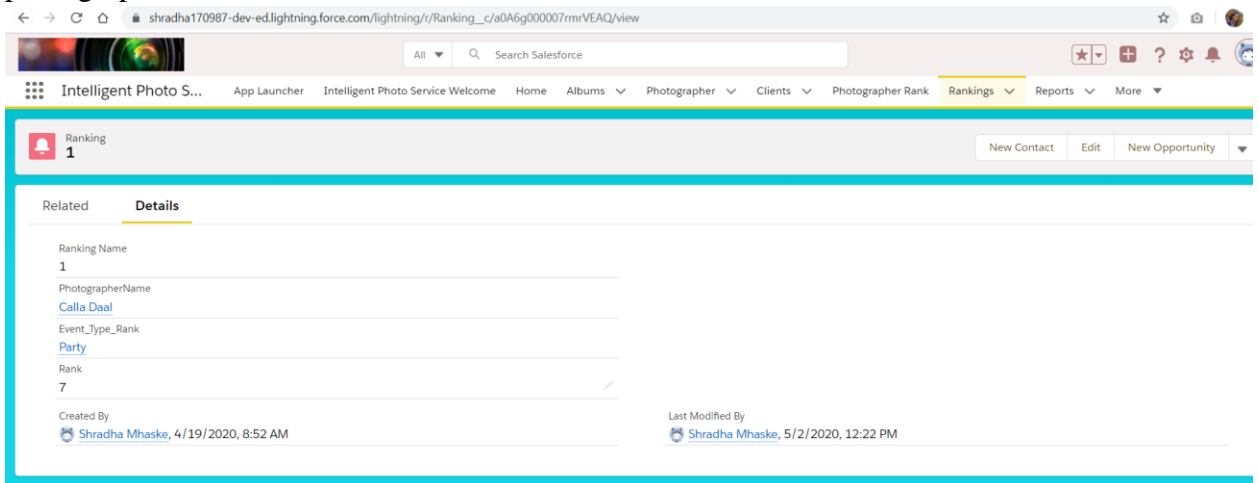
- Every week administrator will get email about latest dashboard updates.
- Go to photographer Rank tab to rank the photographers who have uploaded the event images.
- Click get albumm. List of albums appear. Click on an rank phtoots button.

## Intelligent Photo Service



Id	Title	Description	Created By	Owner	Rank Me
a006g000009GUKDAAW	Abbie first Album		0056g000004nbAIAAI	0056g000004nbAIAAI	Rank Photos
a006g000007JdehAAC	ooo		0056g000004kbYLAAY	0056g000004noO5AAI	Rank Photos
a006g000009GpC4AAK	Second Album Of Calla		0056g000004naDeAAI	0056g000004naDeAAI	Rank Photos
a006g0000097XbBAAU	Sixth Album Of Calla		0056g000004naDeAAI	0056g000004naDeAAI	Rank Photos
a006g000009FhDQAA0	a006g000009FhDQ		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009Fh1kAAC	a006g000009Fh1k		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009FhGtAAK	a006g000009FhGt		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos
a006g000009HCRCAA4	Fifths Album Of Calla		0056g000004kbYLAAY	0056g000004naDeAAI	Rank Photos
a006g000009FI2BAAS	album4		0056g000004kbYLAAY	0056g000004kbYLAAY	Rank Photos

- After clicking rank button toast message will appear.
- Now check the rank of Photographer by selecting corresponding photographer from photographer tab.



Ranking 1

New Contact Edit New Opportunity

Related Details

Ranking Name  
1

Photographer Name  
[Calla Daal](#)

Event\_Type\_Rank  
[Party](#)

Rank  
7

Created By  
[Shradha Mhaske](#), 4/19/2020, 8:52 AM

Last Modified By  
[Shradha Mhaske](#), 5/2/2020, 12:22 PM

## Photographer User Manual

- First contact business administrator to access the salesforce photographer community. Contact email – mhaskes1@montclair.edu
- Using login credentials, login to Photographer Community.  
<https://intelligentphotoservice-developer-edition.na174.force.com/PhotographersIPS>
- Access the home page by clicking on Home tab on navigation menu.
- Access Album tab by clicking Album from Navigation Menu.
- Create your own album by clicking New button located at right side of Album detail page.
- Create your personal clients by clicking client tab located on navigation menu by pressing new button located at right hand side of client detail page.
- Support tab on navigation menu help to raise issue of system or useful to ask help or information from business administrator.

### **Client User Manual**

- First contact business administrator to access the salesforce client community.  
Contact email – mhaskes1@montclair.edu
- Using login credentials, login to client Community.
- <https://intelligentphotoservic-developer-edition.na174.force.com/Customers>
- Access the home page by clicking on Home tab on navigation menu.
- Access Album tab by clicking Album from Navigation Menu.
- Create your own album by clicking New button located at right side of Album detail page.
- Choose highest rank photographer by choosing your event type and clicking get photographer button.
- Support tab on navigation menu help to raise issue of system or useful to ask help or information from business administrator.

## **Glossary**

**SaaS** Software as a Service

**PaaS** Platform as a Service

**IPS** Intelligent Photo Service

**URL** Unified Resource Language

**SOSL** Salesforce Object search Language

**SOQL** Salesforce Object Query Language

**LWC** Lightning Web Components

**VS** Visual Studio

**CLI** Command Line Interface

**HTTP** Hyper Text Markup Language

**API** Application Programming Interface

**WWW** World Wide Web

**MLLC** Machine Learning Life Cycle

**MLOps** Machine Learning Operations

**CRM** Customer Relationship Management

**AI** Artificial Intelligence

**EV – AI** Einstein Vision – Artificial Intelligence

### **Bibliography**

- Salesforce. (n.d.). *connected customer state*. Retrieved from Salesforce Research paper:  
[https://www.salesforce.com/form/conf/state-of-the-connected-customer-3rd-edition/?leadcreated=true&redirect=true&chapter=&DriverCampaignId=cta-header-1&player=&FormCampaignId=7010M000002E3dbQAC&videoid=&playlistId=&mcloudHandlingInstructions=&landing\\_page](https://www.salesforce.com/form/conf/state-of-the-connected-customer-3rd-edition/?leadcreated=true&redirect=true&chapter=&DriverCampaignId=cta-header-1&player=&FormCampaignId=7010M000002E3dbQAC&videoid=&playlistId=&mcloudHandlingInstructions=&landing_page)
- Salesforce. (n.d.). *Einstein ai*. Retrieved from Salesforce Einstein ai: <https://einstein.ai/research>
- Salesforce. (n.d.). *Einstein Prediction Builder*. Retrieved from Salesforce Trailblazer Community:  
[https://help.salesforce.com/articleView?id=custom\\_ai\\_prediction\\_builder.htm&type=5](https://help.salesforce.com/articleView?id=custom_ai_prediction_builder.htm&type=5)
- Salesforce. (n.d.). *Lightning Component Library*. Retrieved from Salesforce Developers:  
<https://developer.salesforce.com/docs/component-library/overview/components>
- Salesforce. (n.d.). *Lightning Web Components*. Retrieved from Salesforce Lightning Web Components:  
<https://developer.salesforce.com/docs/component-library/documentation/en/lwc>
- Salesforce. (n.d.). *Salesforce appExchange*. Retrieved from Appexchange :  
<https://appexchange.salesforce.com/>
- Salesforce. (n.d.). *Salesforce einstein vision api*. Retrieved from  
<https://metamind.readme.io/v1/reference>
- Salesforce. (n.d.). *salesforce help*. Retrieved from help.salesforce:  
[https://help.salesforce.com/articleView?id=salesforce\\_help\\_map.htm&type=5](https://help.salesforce.com/articleView?id=salesforce_help_map.htm&type=5)
- Salesforce. (n.d.). *Salesforce Lightning Design System*. Retrieved from Salesforce Lightning Design System: <https://www.lightningdesignsystem.com/>

## Appendix

### Einstein Machine Learning Component

Einstein vision is used for Visualization of Events

#### EinsteinVision\_Admin\_UI.cmp

```
<aura:component implements="flexipage:availableForAllPageTypes" access="global" controller="EinsteinVision_Admin">
<aura:attribute name="datasets" type="EinsteinVision_Dataset[]"></aura:attribute>
<aura:attribute name="models" type="EinsteinVision_Model[]"></aura:attribute>
<aura:attribute name="spinnerWaiting" type="Boolean" default="false"/>
<aura:handler name="init" value="{!this}" action="{!c.onLoadDatasets}" />
<div class="slds-card">
<div class="slds-p-left_medium slds-p-right_medium">
<lightning:layout verticalAlign="end" class="slds-m-around--small">
<lightning:layoutitem flexibility="grow">
<lightning:input type="URL" label="ZIP file url:" aura:id="zipUrl" value="" />
</lightning:layoutitem>
<lightning:layoutitem>
<lightning:button onclick="{!c.onCreateDataset}">Create Dataset</lightning:button>
<lightning:button onclick="{!c.onLoadDatasets}">Refresh Datasets</lightning:button>
</lightning:layoutitem>
</lightning:layout>
<table class="slds-table slds-table--bordered slds-table--cell-buffer"><thead>
<tr class="slds-text-title--caps">
<th scope="col"><div class="slds-truncate" title="Dataset Name">Dataset Name</div></th>
<th scope="col"><div class="slds-truncate" title="Dataset Labels">Dataset Labels</div></th>
<th scope="col"><div class="slds-truncate" title="Dataset Models">Dataset Models</div></th>
<th scope="col"><div class="slds-truncate" title="Actions">Actions</div></th>
</tr>
</thead><tbody>
<aura:iteration items="{!v.datasets}" var="dataset">
<tr>
<td scope="row" data-label="Dataset Name">
<div class="slds-truncate" title="{!dataset.name}">{!dataset.name}</div>
</td>
<td scope="row" data-label="Dataset Labels">
<aura:iteration items="{!dataset.labelSummary.labels}" var="label">
{!label.name} ({!label.numExamples})<br></br>
</aura:iteration>
</td>
<td scope="row" data-label="Dataset Models">
<aura:iteration items="{!v.models}" var="model">
{!model.modelId} ({!model.status} - {!model.progress*100}%)<br></br>
</aura:iteration>

```



```

        </td>
        <td scope="row" data-label="Actions">
        <div class="slds-truncate">
        <lightning:button onclick="{!c.onTrainDataset}" value="{!dataset.id}" variant="brand
        ">Train</lightning:button>
        <lightning:button onclick="{!c.onDeleteDataset}" value="{!dataset.id}" variant="destructive">Delete</lightning:button>
        </div>
        </td>
        </tr>
    </aura:iteration>
</tbody></table>
<aura:if isTrue="{!v.spinnerWaiting}">
<lightning:spinner size="medium" alternativeText="Loading data..." />
</aura:if>
</div>
</div>
</aura:component>

```

#### EinsteinVision\_Admin\_UIController.js

```

({
  onCreateDataset : function(component, event, helper) {
    helper.onCreateDataset(component);
  },
  onLoadDatasets : function(component, event, helper) {
    helper.onLoadDatasets(component);
  },
  onTrainDataset : function(component, event, helper) {
    helper.onTrainDataset(component, event);
  },
  onDeleteDataset : function(component, event, helper) {
    helper.onDeleteDataset(component, event);
  }
})

```

#### EinsteinVision\_Admin\_UIHelper.js

```

({
  onCreateDataset: function(component) {
    var action = component.get("c.createDatasetFromUrl");
    var zipUrl = component.find("zipUrl").get("v.value");
    console.log(": Zipurl ", zipUrl);
    var self = this;
    action.setParams({
      zipUrl: zipUrl
    });
    action.setCallback(this, function(response) {
      component.set("v.waiting", false);
    });
  }
})

```

```

var state = response.getState();
if (state === 'ERROR') {
    var errors = response.getError();
    if (errors) {
        if (errors[0] && errors[0].message) {
            return alert(errors[0].message);
        }
    } else {
        return console.log("Unknown error");
    }
}
var result = response.getReturnValue();
self.onLoadDatasets(component);
});
component.set("v.waiting", true);
$A.enqueueAction(action);
},
onLoadDatasets : function(component) {
var self = this;
var action = component.get("c.getDatasets");
action.setCallback(this, function(response) {
    var state = response.getState();
    if (state === 'ERROR') {
        var errors = response.getError();
        if (errors) {
            if (errors[0] && errors[0].message) {
                return alert(errors[0].message);
            }
        } else {
            return console.log("Unknown error");
        }
    }
    component.set("v.datasets", response.getReturnValue());
    var dataset = response.getReturnValue();
    if (dataset && dataset.length>0) {
        self.onModelStatus(component, dataset);
    } else {
        component.set("v.spinnerWaiting", false);
    }
});
component.set("v.spinnerWaiting", true);
$A.enqueueAction(action);
},
onModelStatus : function(component, datasets) {
var action = component.get("c.getModels");
action.setParams({

```

```

        datasetId: datasets[0].id
    });
    action.setCallback(this, function(response) {
        component.set("v.spinnerWaiting", false);
        var state = response.getState();
        if (state === 'ERROR') {
            var errors = response.getError();
            if (errors) {
                if (errors[0] && errors[0].message) {
                    return alert(errors[0].message);
                }
            } else {
                return console.log("Unknown error");
            }
        } else {
            component.set("v.models", response.getReturnValue());
        }
    });
    component.set("v.spinnerWaiting", true);
    $A.enqueueAction(action);
},
onDeleteDataset : function(component, event) {
    var action = component.get("c.deleteDataset");
    var datasetId = event.getSource().get("v.value");
    var self = this;
    action.setParams({
        datasetId: datasetId
    });
    action.setCallback(this, function(response) {
        component.set("v.spinnerWaiting", false);
        var state = response.getState();
        if (state === 'ERROR') {
            var errors = response.getError();
            if (errors) {
                if (errors[0] && errors[0].message) {
                    return alert(errors[0].message);
                }
            } else {
                return console.log("Unknown error");
            }
        }
        self.onLoadDatasets(component);
    });
    component.set("v.spinnerWaiting", true);
    $A.enqueueAction(action);
},

```

```

onTrainDataset : function(component, event) {
var action = component.get("c.trainDataset");
var datasetId = event.getSource().get("v.value");
var self = this;
action.setParams({
    datasetId: datasetId
});
action.setCallback(this, function(response) {
    component.set("v.spinnerWaiting", false);
    console.log("Componentet ", component);
    var state = response.getState();

    console.log("response ", response);
    if (state === 'ERROR') {
        var errors = response.getError();
        if (errors) {
            if (errors[0] && errors[0].message) {
                return alert(" ABC " + errors[0].message);
            }
        } else {
            return console.log("Unknown error");
        }
    } else {
        var toastEvent = $A.get("e.force:showToast");
        toastEvent.setParams({
            "title": "Success!",
            "type": "success",
            "message": "The model id for the training is " + response.getReturnValue() + ". Refres
h the dataset for seeing the training progress."
        });
        toastEvent.fire();
    }
});
component.set("v.spinnerWaiting", true);
$A.enqueueAction(action);
}
})

```

EinsteinVision\_Admin.apxc

```

public class EinsteinVision_Admin {
    @AuraEnabled
    public static void createDatasetFromUrl(String zipUrl) {
        EinsteinVision_PredictionService service = new EinsteinVision_PredictionService();
        EinsteinVision_Dataset dataset = service.createDatasetFromUrlAsync(zipUrl);

        System.debug('Hi this is my dataset ' + dataset.name);
    }
}

```

```
@AuraEnabled
public static List<EinsteinVision_Dataset> getDatasets() {
    EinsteinVision_PredictionService service = new EinsteinVision_PredictionService();
    EinsteinVision_Dataset[] datasets = service.getDatasets();
    return datasets;
}

@AuraEnabled
public static String trainDataset(Decimal datasetId) {
    EinsteinVision_PredictionService service = new EinsteinVision_PredictionService();
    EinsteinVision_Model model = service.trainDataset(Long.valueOf(String.valueOf(dataset
Id)), 'Training', 0, 0, "");
    return model.modelId;
}

@AuraEnabled
public static void deleteDataset(Long datasetId) {
    EinsteinVision_PredictionService service = new EinsteinVision_PredictionService();
    service.deleteDataset(datasetId);
}

@AuraEnabled
public static List<EinsteinVision_Model> getModels(Long datasetId) {
    EinsteinVision_PredictionService service = new EinsteinVision_PredictionService();
    EinsteinVision_Model[] models = service.getModels(datasetId);
    return models;
}
}
```

**Photographer Ranking Component**

Ranking photographers based on uploaded photos in Album.

**RankMe.cmp**

```
<aura:component implements="force:appHostable,flexipage:availableForAllPageTypes,force
Community:availableForAllPageTypes" controller="Rankmefirst" access="global" >
<aura:attribute name="reg" type="ContentDocument[]"/>
<ui:button label="Get Albums" press="{!c.myAction}"/>
<table>
<tr>
<td> Id </td>
<td> Title </td>
<td> Description </td>
<td> Created By </td>
<td> Owner </td>
<td> Rank Me </td>
</tr>
<aura:iteration var="r" items="{!v.reg}" >
<tr>
<td>{!r.Id}</td>
<td>{!r.Name}</td>
<td>{!r.Description__c}</td>
<td>{!r.LastModifiedById}</td>
<td>{!r.OwnerId}</td>
<td>
<lightning:button label="Rank Photos" onclick="{!c.rankAlbum}" value="{!r.Id}"/>
</td>
</tr>
</aura:iteration>
</table>
</aura:component>
```

**RankMeController.js**

```
((
myAction : function(component, event, helper) {
var action = component.get("c.getAllAlbums");
action.setCallback(this, function(response){
var name = response.getState();
if (name === "SUCCESS") {
component.set("v.reg", response.getReturnValue());
}
});
$A.enqueueAction(action);
},
rankAlbum : function(component, event, helper) {
var idAlbum = event.getSource().get("v.value");
var action = component.get("c.getFilesForAlbum");
action.setParams({ idAlbum: idAlbum });
```

```

action.setCallback(this, function(response){
var name = response.getState();
if (name === "SUCCESS") {
    alert("All updated ranks for album :: " , response.getReturnValue());
}
});
$A.enqueueAction(action);
}
})

```

## Rankmefirst.apxc

```

public class Rankmefirst {
@auraEnabled
public static List<Album__c> getAllAlbums()
{
List<Album__c> reg = [select id, Name, Description__c, LastModifiedById, OwnerId from Album__c];
return reg;
}

@auraEnabled
public static List<Ranking__c> getFilesForAlbum(String idAlbum)
{
List<ContentDocumentLink> cdLists = [SELECT ContentDocumentId FROM ContentDocumentLink WHERE LinkedEntityId = :idAlbum];
Set<Id> contentDocumentIds = new Set<Id>();
for(ContentDocumentLink c : cdLists) {
    contentDocumentIds.add(c.ContentDocumentId);
}
List<ContentDocument> cds = [SELECT Id, Title, CreatedById, OwnerId FROM ContentDocument WHERE Id IN :contentDocumentIds];
List<ContentVersion> cvs = [SELECT VersionData, CreatedById FROM ContentVersion WHERE ContentDocumentId IN :contentDocumentIds AND IsLatest = true];
Integer rankedItems = 0;
Map<String, Ranking__c> mapRanks = new Map<String, Ranking__c>();

for(ContentVersion c : cvs) {
    String eventName = getPhotoPrediction(c);
    String userId = c.CreatedById;
    if('NO_RESULT' != eventName ) {
        String key = userId + '_' + eventName;
        Ranking__c rank = null;
        if(mapRanks.containsKey(key)){
            rank = mapRanks.get(key);
            rank.Rank__c = rank.Rank__c + 1;
        } else {
            rank = getRankByUserNameAndEventTypeName(userId, eventName);

```

```

        mapRanks.put(key, rank);
    }
}

Savepoint sp = Database.setSavepoint();
try{
    Database.upsert(mapRanks.values());
    rankedItems = mapRanks.values().size();
} catch(Exception e){
    Database.rollback(sp);
}
return mapRanks.values();
}

public static Ranking__c getRankByUserNameAndEventTypeName(String userId, String eventTypeName){
try {
    User user = [Select Username From User where Id= :userId];
    String email = user.Username;
    Photographer__c photographer = [Select Id From Photographer__c where email__c = :user.Username];
    Event_Type__c event = [Select Id, Name From Event_Type__c where Name = :eventTypeName];
    String etName = event.Name;
    Ranking__c[] ranking = [Select Name, Rank__c, PhotographerName__c, Event_Type_Rank__c
                                from Ranking__c
                                where
                                    Event_Type_Rank__c = :event.Id
                                    and PhotographerName__c = :photographer.Id];
    if(ranking.size() > 0){
        Ranking__c rank = ranking[0];
        rank.Rank__c = rank.Rank__c + 1;
        return rank;
    } else {
        Ranking__c rank = new Ranking__c();
        rank.Event_Type_Rank__c = event.Id;
        rank.PhotographerName__c = photographer.Id;
        rank.Rank__c = 1;
        return rank;
    }
} catch(Exception e){
    return null;
}
}

```



```
public static String getActiveDataset() {
    Active_Dataset__c ad = Active_Dataset__c.getOrgDefaults();
    return ad.Name__c;
}

public static String getPhotoPrediction(ContentVersion c) {
    Blob fileBlob = c.VersionData;
    EinsteinVision_PredictionService service = new EinsteinVision_PredictionService();
    EinsteinVision_Dataset[] datasets = service.getDatasets();
    String activeDataset = getActiveDataset();
    for (EinsteinVision_Dataset dataset : datasets) {
        if (dataset.Name.equals(activeDataset)) {
            EinsteinVision_Model[] models = service.getModels(dataset);
            EinsteinVision_Model model = models.get(0);
            EinsteinVision_PredictionResult result = service.predictBlob(model.modelId, fileBlob, "");
            ;
            if(result != null){
                return result.proBABILITIES.get(0).label;
            } else {
                return 'NO_RESULT';
            }
        }
    }
    return 'NO_RESULT';
}
}
```

**Photographer Search Component**

Searching and presenting ranked photographers.

**RankedPhotographersByEventType.cmp**

```

<aura:component implements="force:appHostable,flexipage:availableForAllPageTypes,force
Community:availableForAllPageTypes"
  controller="rankedPhotographersByEventType"
  access="global" >
<aura:attribute name="reg" type="ContentDocument[]"/>
<lightning:select name="eventType"
  aura:id="selectOne"
  label="Which type of event you are planning?"
  messageWhenValueMissing="Choose one!"
  onChange="alert('hi')"
  required="true">
<option value="">choose one...</option>
<option value="a076g000002XIIMAA0">Sport</option>
<option value="a076g000002XjrxAAC">Party</option>
<option value="a076g000002XIIMAA0">Festivals</option>
<option value="a076g000002Xr4GAAS">Meeting</option>
<option value="a076g000002Xr4KAAS">Personal</option>
<option value="a076g000002Xr4PAAS">Thanksgiving</option>

</lightning:select>
<ui:button label="Get Photographers" press="{!c.onEventTypeChange}"/>
<aura:iteration var="r" items="{!v.reg}" >
<lightning:card>
<aura:set attribute="title">
  <lightning:avatar size="large"
    variant="circle"
    src="https://developer.salesforce.com/docs/component-
library/app/images/examples/avatar1.jpg"
    fallbackIconName="standard:avatar"
    alternativeText="{!r.Name}"
    class="slds-m-right_small"/>
    {!r.Name}
  <lightning:formattedAddress
    street="{!r.Address__c}"
    city="{!r.City__c}"
    province="{!r.State__c}"
    postalCode="{!r.Zipcode__c}"
  >
</lightning:formattedAddress>
<p><lightning:clickToDial value="{!r.Phone__c}"/></p>

```

```

<p><lightning:formattedEmail value="{!r.Email__c + '?subject=Looking%20for%20photog
raphe%20' }"
    label="Email This Photographer" />
</p>
</aura:set>
</lightning:card>
</aura:iteration>
</aura:component>

```

#### RankedPhotographersByEventTypeController.js

```

({
onEventTypeChange : function(component, event, helper) {
var idEventType = component.find("selectOne").get("v.value");
if(idEventType === ""){
    alert("Please select event type");
    return;
}
var action = component.get("c.getPhotographerByEventType");
action.setParams({
    eventTypeId: idEventType
});
action.setCallback(this, function(response){
    var name = response.getState();
    if (name === "SUCCESS") {
        component.set("v.reg", response.getReturnValue());
    }
});
$A.enqueueAction(action);
}
})

```

#### RankedPhotographersByEventType.apxc

```

public class rankedPhotographersByEventType {
@auraEnabled
public static List<Photographer__c> getPhotographerByEventType(String eventTypeId)
{
List<Ranking__c> rankings = [Select PhotographerName__c, rank__c
                        from Ranking__c
                        where
                        Event_Type_Rank__c = :eventTypeId
                        ORDER BY rank__c DESC];
List<Id> photographerIDs = new List<Id>();
for(Ranking__c c : rankings){
    photographerIDs.add(c.PhotographerName__c);
}

List<Photographer__c> photographerList = [Select Name, State__c, City__c, Email__c, Addr
ess__c, Phone__c, Zipcode__c from Photographer__c where id IN :photographerIDs];

```

```
System.debug(photographerList.size() + (' photographers(s) returned.'));

List<Photographer__c> rankPGMap = new List<Photographer__c>();
Integer i = 1;
for(Ranking__c c : rankings){
    for(Photographer__c p : photographerList){
        if(p.id == c.PhotographerName__c){
            rankPGMap.add(p);
            i++;
        }
    }
}
return rankPGMap;
}
```