Bilkent University

Department of Computer Engineering

Senior Design Project

Kalas-Iris: Clothes recognition and rich attribute prediction using computer vision service for online clothing retail

Final Report

Olcay Akman, Doğaç Eldenk, Zeynep Korkunç, Hüseyin Ata Atasoy

Supervisor: Ayşegül Dündar

Final Report

30 April, 2021

1 Introduction

Online shopping is becoming more and more popular. There are around 7.1 million online retailers in 2020 and 1.8 billion people are shopping online each year[1]. The demand on online shopping is requiring more resources to be spent on customers. There are two main ways customers find the goods that they buy in those websites: Browsing and searching. According to Oberlo, 75% of the search queries of the customers are brand new[2]. Therefore, it is important for customers to find the good that they are searching for.

The ever-growing product range and amount in the e-commerce world has become overwhelming. Especially in clothes categories, the products are rather poorly labeled, which results in inaccurate or misleading search results via the search engines provided by the e-commerce websites. This problem stems from the fact that the products in those websites are poorly labeled. As an online retail company based in Turkey had explained to us, the labeling and categorization of the products available on their online store is done manually by humans[3]. This results in two main issues:

- Labels for products of the same category differ as not all product labeling is done by the same person.
- The online retail business owner pays too much money for the human workforce who work on the labeling of their products, and there is too much labor compared to when this process is completed via a computer.

Another problem present in this scheme of the online retail shopping world arises when a customer to e-commerce websites uses the search engine of the website to search for a product. The search among the products available on the e-commerce website is made via the labels of those products. Since the product labeling is made by humans, this leads to mismatching or half-complete product labels, which in turn results in poor search results for the customer.

Kalas-Iris service targets these problems. As explained, the core of such problems arises from poor labeling of products. Hence, the solutions to those problems lie in fixing the labeling and categorization of products. The purpose of our service is to automate the labeling process of retail products, produce consistent labels for products, and allow for an enhanced semantic search algorithm to run amongst the labels produced for each product in order to provide accurate search results.

2 Requirement Details

2.1 Functional Requirements

User Account

• The system should provide its users to create an account in order for them to benefit from its services.

• Customers can add their website’s end points to the system for automatic image labeling service.

• Kalas-iris customers should be able to obtain a private key from the web application.

• By using the key, customers should be able to make requests to the system in order for them to get the required results for their images.

Image Category and Attribute Prediction

• Customers can upload single images or use custom webhooks for uploading an image of the new item to the system.

• The system will automatically predict the category and the attributes of the given item.

• The system will send the results to the customer by making a http request to the customer’s website or showing a simple webpage.

Automatic Category and Attribute Tagging Service

• The system will predict the category and attributes of an item using its image.

• The system will benefit from a pre-trained model using modern Deep Learning techniques.

• The system will be available on an endpoint in the webapp.

Similar Item Retrieval Service

• The user can upload an image to the retrieval service.

• The retrieval service will find the closest match to the uploaded image inside a dataset.

Mobile Application

• Users can take photos using the mobile application.

• Users can edit and crop the photos using the mobile application.

• Users can see the annotation results of the taken photo.

• Users can see the similar products with respect to the taken photo.

2.2 Non-Functional Requirements

Scalability and Flexibility

The design of the web service provided should be flexible in the sense that it should have room to adapt to changes if they occur. The current system aims to provide two basic functionalities of product labeling and similiar image retrieval. After successful installation of these core functionalities, the service may be up for improvements, which will likely require the extension of the current one. Thus, the design is aimed to be scalable. Following the same principle, the service may get further and newer functionalities. The current functionalities should be flexible enough to provide a seamless integration of the new functionalities to the service, therefore the design is desired to be flexible.

Security

The products processed and kept by the Kalas-Iris service will be kept safe. That is, such data will not be accessible to third parties. The product images uploaded by the e-commerce business owners, and the data generated for those products, along with any other data related with the products in Kalas-Iris, will be solely available to the owners of the products and will be processed by the Kalas-Iris service only. This will ensure confidentiality.

Backup and Recovery and Synchronization

The users of the Kalas-Iris web application, the e-commerce website owners, should be able to process several products via a single account. The products they process and obtain results about are to be kept by the service. In this regard, we will use MongoDB to store the products and related data. The products added to the system by customers will be stored this way, and customers will be able to reach those whenever they want via Kalas-Iris website.

Availability

System should be accessible to its users all the time except for some possible errors caused by the services. The e-commerce website owners should be able to access the service to perform all the actions they are authorized to do, which initially include adding new products to their account, processing products for labeling and viewing their added products whenever they like via a web browser. In addition, all e-commerce website owners, wherever in the world they may be from, should be able to access the service and create accounts in order to benefit from our services.

Reliability

The application should be able to give accurate results, both classifying the categories and labels of the desired products. Also, the application should be able to handle possible errors immediately instead of giving mismatching results, and reporting them to the developers and the users wherever and whenever deemed necessary.

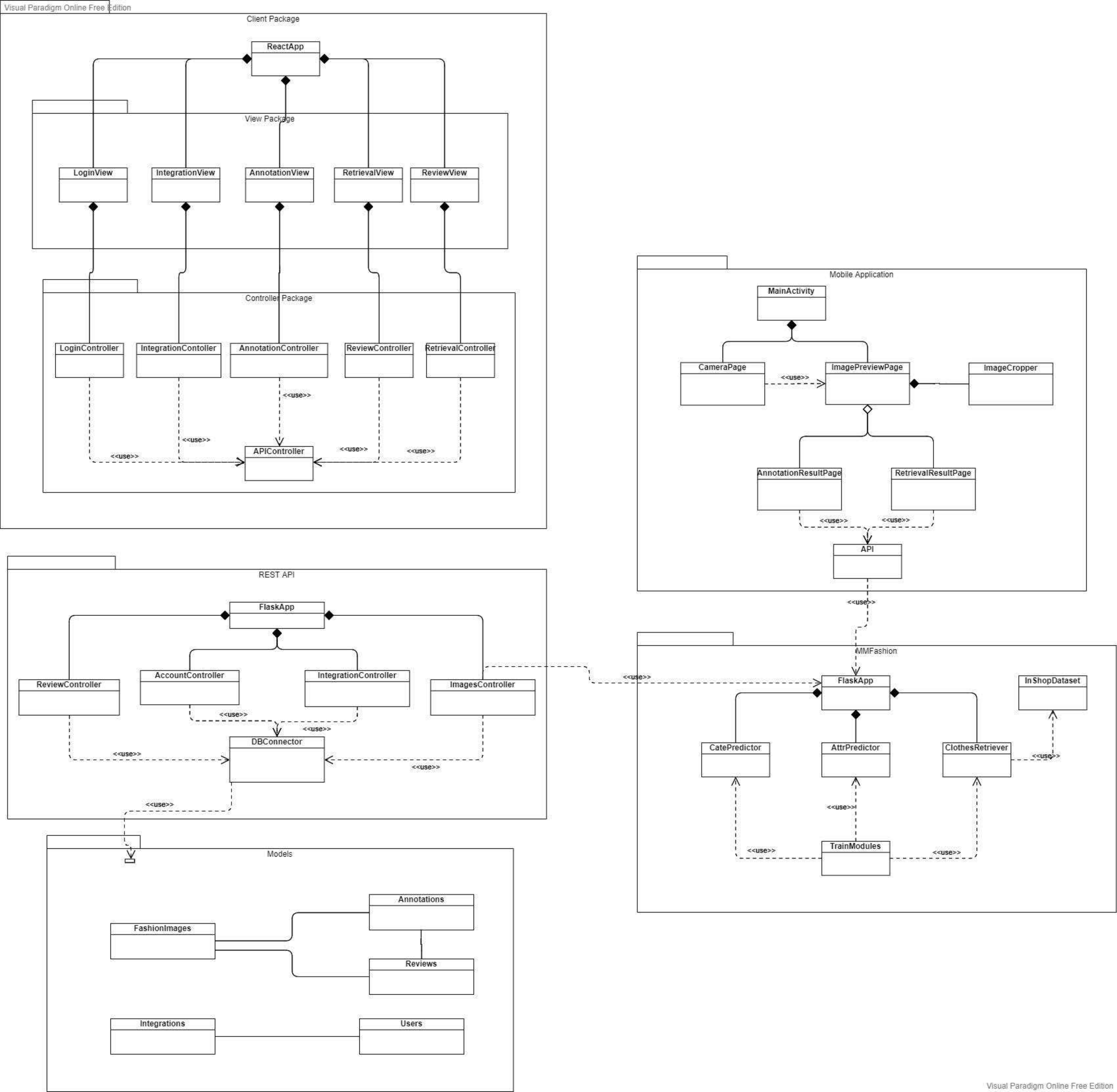
Usability

The application should be learnable, in order words, it should not take a long time to master for the users. Also, the application should provide a system to users where users can meet to their expectations and achieve their goals. In addition, in terms of efficiency, users should be able to achieve those goals in an optimal amount of time.

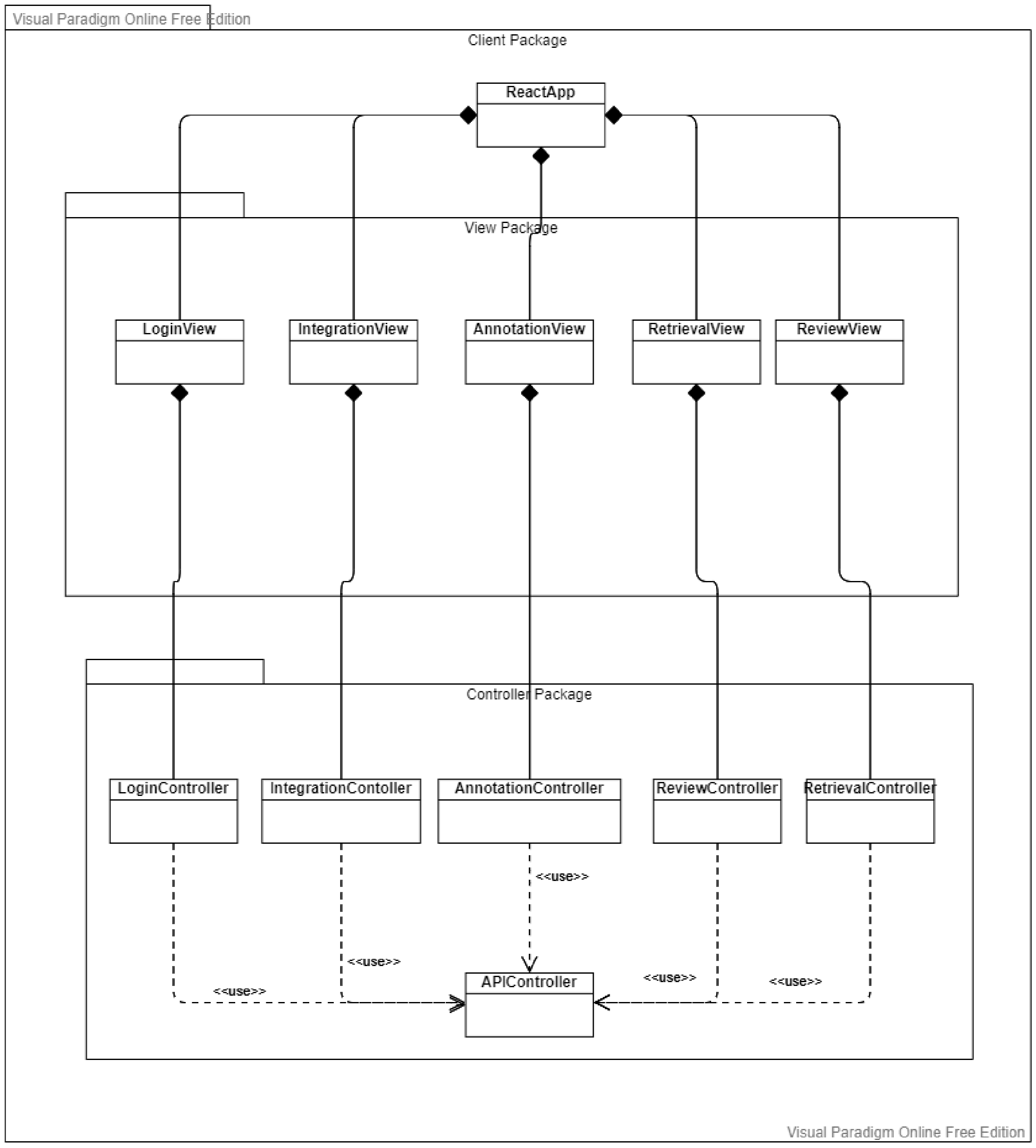
Deployment

The system should be able to deploy to the cloud systems in under a minute. This will ensure that the system will have a good uptime. The machine learning models are also required to be deployable on cloud systems. On demand, this system can work in a self hosted computer.

3 Final Architecture and Design


Client Package
ReactApp
View Package
LoginView
IntegrationView
AnnotationView
RetrievalView
ReviewView
Controller Package
LoginController
IntegrationContoller
AnnotationController
ReviewController
RetrievalController
<<use>>
APIController
Mobile Application
MainActivity
CameraPage
ImagePreviewPage
ImageCropper
AnnotationResultPage
RetrievalResultPage
API
REST API
FlaskApp
ReviewController
AccountController
IntegrationController
ImagesController
DBConnector
MMFashion
FlaskApp
InShopDataset
CatePredictor
AttrPredictor
ClothesRetriever
TrainModules
Models
Annotations
FashionImages
Reviews
Integrations
Users

Client Package



Client subsystem of the Client-Server architecture is composed of two subsystems, named View and Controller.

View Package

The View subsystem consists of several views related to the front-end side of the API. They are all controlled by the ReactRouter. The views are:

- LoginView: The page the customers use to login or signup to Kalas-Iris service.
- IntegrationView: The page which aids the customer to integrate the Kalas-Iris service to their own e-commerce website.
- AnnotationView: The page which allows users to upload, crop and annotate a given image. They can also change the annotation results and send them back to the server as a correction.
- RetrievalView: The page which allows users to upload, crop and find similar products to the given image. The similar products are shown in a gallery fashion.
- ReviewView: The page which allows users to review the past annotations and the adjustments made to the annotation results from this page.

Controller Package

The controller subsystem of Client consists of the controllers for the views mentioned in View. All those controllers are managed by the ApiConnectorController. The controllers are:

- AuthenticationViewController: Connects the authentication API with the front-end. Manages the states of the views.
- IntegrationViewController: Connects the integration API with the front-end. Manages the states of the views.
- AnnotationViewController: Connects the annotationAPI to the frontend. Manages the states of the views.
- RetrievalViewController: Connects the annotationAPI to the frontend. Manages the states of the views.
- ReviewViewController: Connects the annotationAPI to the frontend. Manages the states of the views.
- APIController: Binds the methods on the frontend to the backend using REST clients.

Server Package

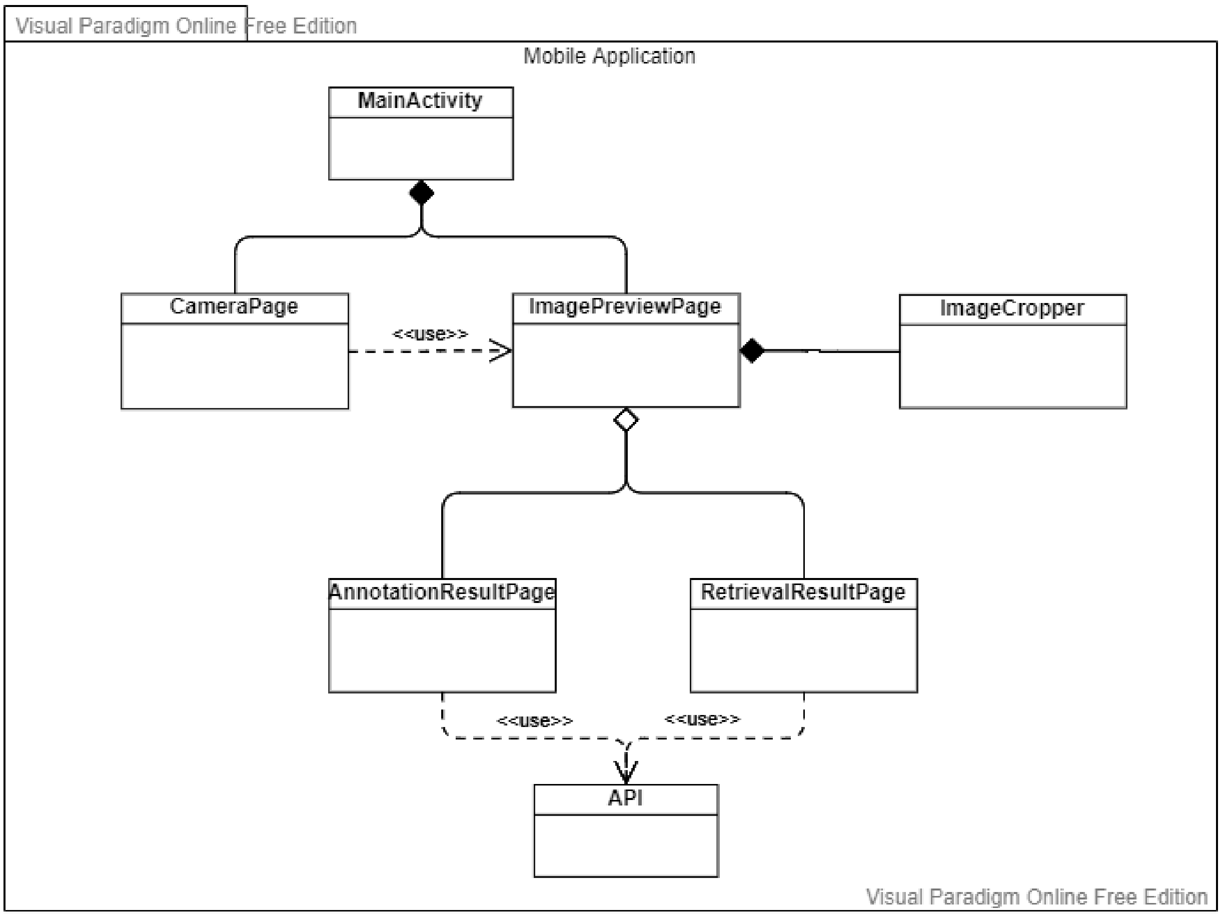
The server part of the Client-Server architecture deals with the back-end part of the Kalas-Iris service. It consists of the following subsystems:

- RestAPI, which controls the API's backend and communicates with the Front-end.

- Machine Learning, which controls the fashion annotation and semantic search part of the Kalas-Iris service.
- Models, which controls the models generated by Kalas-Iris and their representation in the Kalas-Iris website.

The REST API subsystem of the Client consists of the controllers for the Back-end service which will handle the data that is fetched from the machine learning and model subsystems and it communicates with the Server side in order to display the data. The controllers are managed by the ApiController as shown.

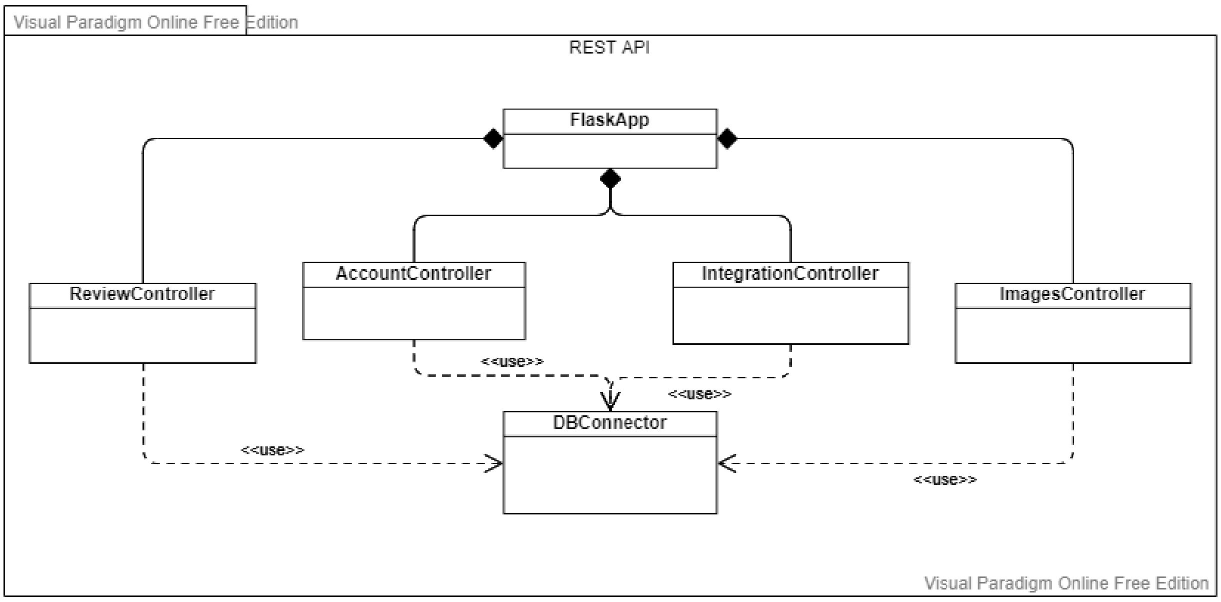
Mobile Application



- MainActivity: Entrypoint for the flutter application.
- CameraPage: Page with the camera view that the user can take photos of the fashion items.

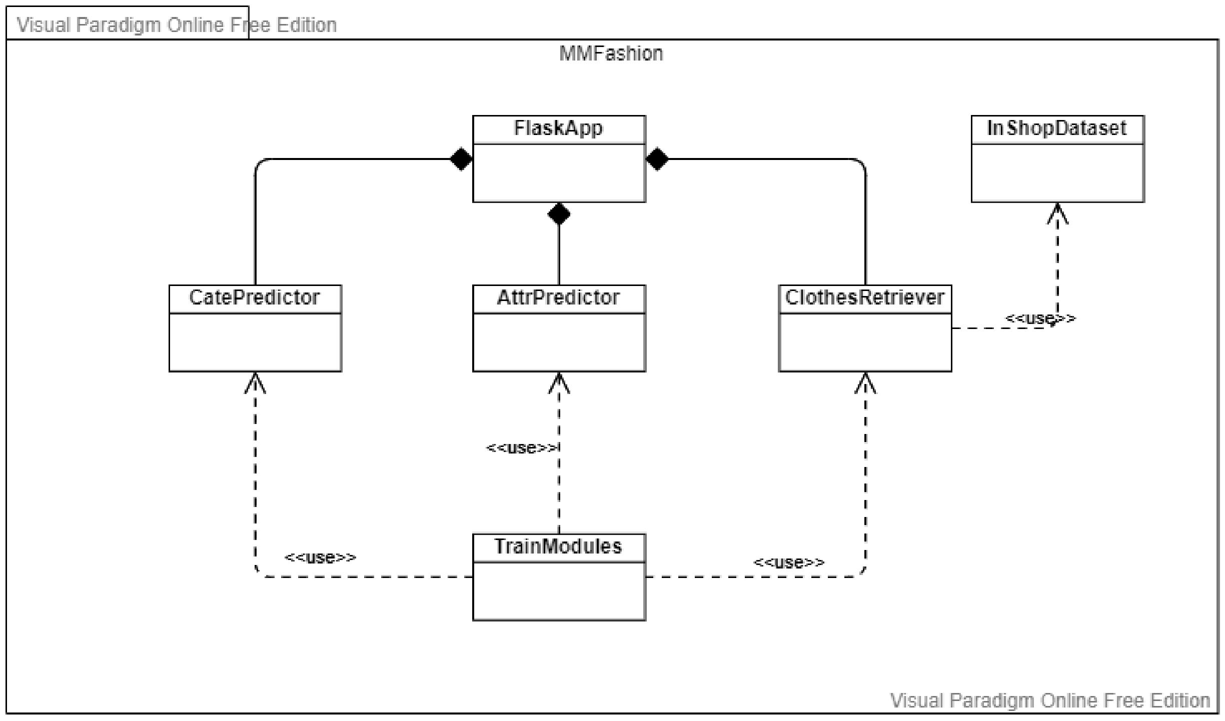
- ImagePreviewPage: The taken and edited photos are displayed under this page can be annotated or queried for the similar products.
- ImageCropper: A flutter widget used to crop the images.
- AnnotationResultPage: The page where the results of the annotations are shown.
- RetrievalResultPage: The page where the retrieved similar images are shown.
- API: Connector for the Machine Learning service.

REST API



- ImagesController: Contains the controllers for retrieval and annotations. Connect the machine learning models from machine learning servers to the backend.
- AccountController: Fetches the data related to the customers account from the model subsystem and sends the obtained results to the ApiController.
- ReviewController: Manages the reviewed products and their annotations. Those models will be reused by the machine learning models for improvement.
- IntegrationController: Controls the integration of the Kalas-Iris service to the customer's e-commerce website.
- DBConnector: Maintains the database connections of the REST API.

Machine Learning



MMFashion

The class in the Annotation package is:

- AttrPredictor: Attribute Predictor machine learning model with its configuration and helper functions.
- CatePredictor: Category Predictor machine learning model with its configuration and helper functions.
- ClothesRetriever: In-shop clothes retriever machine learning model with its configuration and helper functions.
- TrainModules: Module used for training the Neural Network models.
- InShopDataset: Dataset which clothes retriever uses to query the similar images.

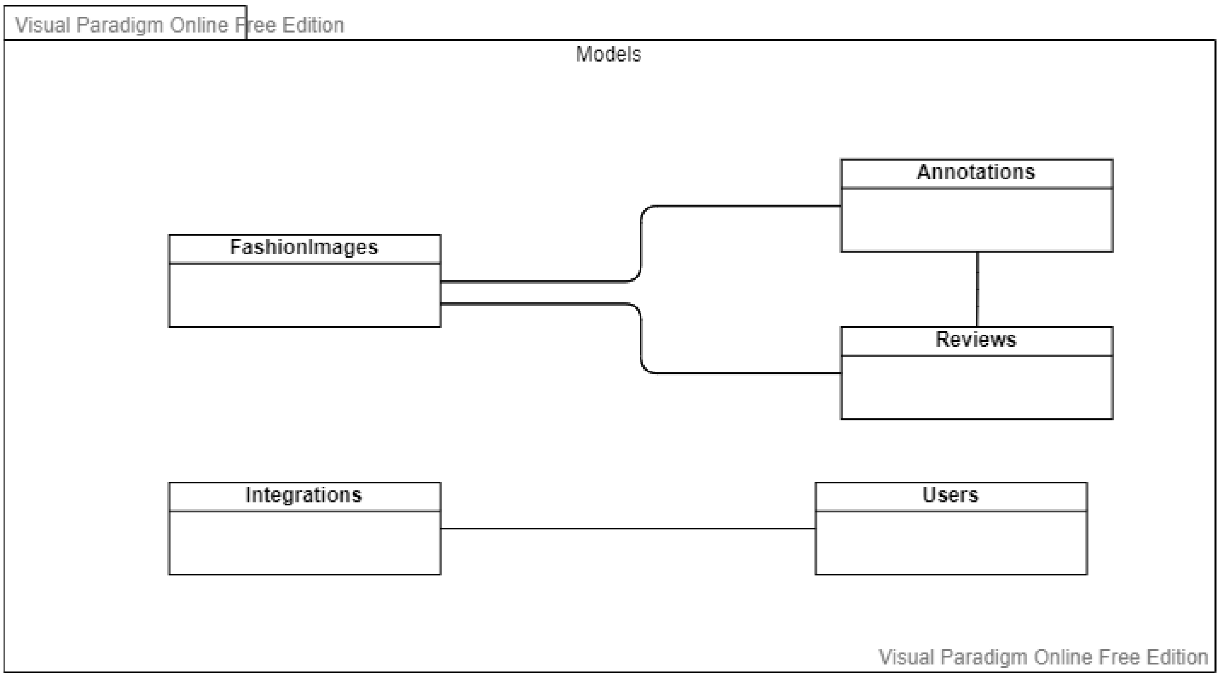
Core

Core of our system is the MMFashion module. This module is highly complex. You can check out our source for better understanding of the Neural Network we are using.

- Data: Modules for preparing the data for processing or training.
- MMDetection: Submodule used for detection of the clothing bounding boxes.

- Models: Contains the neural network models. We are currently using, retriever, attr_cate_predictor, global_pool, backbones and losses to construct our neural networks.
- Dataset: Contains the training or retrieval dataset for the models.
- Utils: Contains the image and training utilities.

Models


Visual Paradigm Online Free Edition
Models
Annotations
FashionImages
Reviews
Integrations
Users
Visual Paradigm Online Free Edition

The classes in the Models package are:

- FashionImages: Contains the images of Fashion Items uploaded to the system.
- Annotations: Stores the annotations for the Fashion Items.
- Reviews: Stores the user reviews of the annotation results.
- Integrations: A collection of e-commerce integration credentials.
- Users: Contains the user credentials for authentication.

4 Development and Implementation Details

Client - Server Implementation

The client and the front-end is implemented in React JS. We chose React over other front-end frameworks as it's well documented and we had previous hands-on experience with it. The server back-end is implemented in Python Flask. We built a REST API based on Flask since it was easier to learn and flexible compared to other Python Frameworks such as Django.

NN Model Implementation

Kalas-Iris uses a modification of the open source MMFashion [5] visual fashion analysis toolkit provided by the MultimediaLab CHUK. MMFashion uses a Backbone VGG16 or ResNet-50 to extract the features. It uses Cross-Entropy loss on the category attribute prediction task. MMFashion currently uses a global pooling mechanism but it also supports a landmark pooling based on the landmarks generated by the landmark generator model. The predictors are basically linear layers connected to a final probability layer. However the image retriever uses only the attribute predictor and the embedding extractor. The embedding extractor uses a TripletLoss with an linear embed layer, identity layer and batch norm.

This model is very similar to the model proposed on the DeepFashion paper [7]. It is also trained on the DeepFashion dataset. This dataset contains over 220,000 images of fashion items on 1000 different attributes and 50 categories [6].

Persistent Data Management

Kalas-Iris uses MongoDB Atlas to store annotations, user credentials and integration credentials. MongoDB was chosen as its NoSQL feature allows for more flexible storage, compared to a relational database. It's also deployed on the cloud independent from Kalas-Iris, thus we did not have to worry about its availability.

Services Integration

Our service uses webhooks to communicate with the integrations. A Webhook was created on the WooCommerce API to trigger when a new product was created. The Webhook sent the product information to the Kalas-Iris REST API, which then forwards the images to the MMFashion service for annotation. Finally, the annotation results are used to set the product tags and category.

5 Testing Details

Fashion Annotation and In-Store Retrieval features were tested manually. Fashion Annotation takes about 2 seconds and In-Store Retrieval takes about 20 seconds to finish. During testing, it took 30 seconds for the Webhook to receive the payload from WooCommerce[4] API and update the product with the annotation results. We are expecting our customers to integrate with our services asynchronously using the webhook architecture. Therefore those response times are not a big deal and they should not affect customers directly.

Performance results for the MMFashion models are given in the table below.

Model	**Top-5 Recall %**	**Top-5 Precision %**
Coarse Attribute Predictor	23.52	99.29

Model	**Top-5 Category Recall %**	**Top-5 Attribute Recall %**
Fine Category and Attribute Predictor	35.91	25.44

Model	**Top-5 Accuracy %**
In-Shop Clothes Retriever	38.76

6 Maintenance Plan and Details

Operating costs of our services are very high at the moment. Hourly cost of the service is around $0.5 because neural networks require CUDA environment to work at full capacity. Also we were not able to train the model on our own because it costs us around $500 and a couple of weeks to replicate the environment the pre-trained model has been trained in. Assuming that we are not going to train the model for once and perfect it at the first run, it can cost us more than $5000 dollars to fully train, test and evaluate the model. Therefore we require funding mostly from our customers in order to provide them seamless integration and the best performance. But still, we can narrow our scope down and use the pretrained models as it is until we get the funding and we can concentrate more on the integration side of the project.

7 Other Project Details

Consideration of Various Factors in Engineering Design

	Effect Level (Out of 10)	Effect
Public Health	1	The ease of search for products in online retailing does not have any immediate effects on public health.
Public Safety	1	Similarly, the ease of search in online retailing does not have any effects on public safety either.
Public Welfare	4	Online retailing is very popular and possible if overall welfare is low.
Global Factors	5	Changes in the global fashion industry will affect the accuracy and reliability of our system. Our system should be able to adapt.
Cultural Factors	7	Fashion industry has differences in every culture. The language changes over time so searching algorithms should adapt to the new language.
Social Factors	3	Customers' websites should be protected and not exposed to the other people.

Ethics and Professional Responsibilities

In terms of the scope of this project, Kalas Iris uses product images and related data, which are usually confidential to the company who has the rights for those products. By using our services, the e-commerce business owners confide in Kalas Iris to keep their data safe from attackers and third parties.

Judgements and Impacts to Various Contexts

Initially we believed that we could have developed our neural network for detecting categories and attributes. Moreover we have seen that our hardware is not sufficient to train such a network. Even if we did, it might take months to train a single model. Because of this constraint, we have decided to work with the pretrained models provided by the MultiMedia Laboratory, authors of the MMFashion paper. Also we were hosting our models and machine learning backends on the GCP with high costs. Therefore we had to shut down our system regularly when we were not using it. Also we did not have any budget for the senior project therefore we had to keep our servers as simple as possible. We could have tried training the models if we had some budget or hardware given by the department.

Teamwork Details

Peer Contribution

Doğaç Eldenk:

- Project management, task assignment, sprint planning and development environment documentation.
- Workflow automations such as automated issue creation, CI/CD pipelines, github project board automations, github discord bot notification integration.
- Setting up machine learning codebase and dataset for image category predictor, attribute predictor and in-shop retriever models.
- Designing a REST API for the machine learning service and connecting it with FE and BE and deploying it to the google cloud platform.
- Implementing the mobile application.
- Regular code reviews, pairing with Ata Atasoy on bugs, solving small issues on the backend and frontend.

Hüseyin Ata Atasoy:

- Implementing the user authentication and registration system (both front-end and back-end).

- Implementing the WooCommerce integration (both front-end and back-end and Webhook).
- Implementing Image Annotation and Image Retrieval services (both front-end and back-end).
- Implementing integration controller on back-end.
- Worked on Past Reviews View on front-end.
- Worked on Color Recognition modules for Fashion Images.
- Setting up and connecting MongoDB Atlas to the back-end.
- Code reviews for Doğaç Eldenk's pull requests.

Olcay Akman:

- Designing the Kalas Iris webpage design and considering UX matters for the website
- Creating menu items on the navigation bar for the website
- Adding explanatory content for the explanation of the tool to the website
- Working in collaboration with Zeynep Korkunç for UI design and implementation

Zeynep Korkunç:

- Designed the UI for the Kalas-Iris Webpage and conduct research in UX
- Working with Olcay Akman during the implementation of UI design
- Creating landing page images and information display for the Kalas Iris webpage

Helping creating a collaborative and inclusive environment

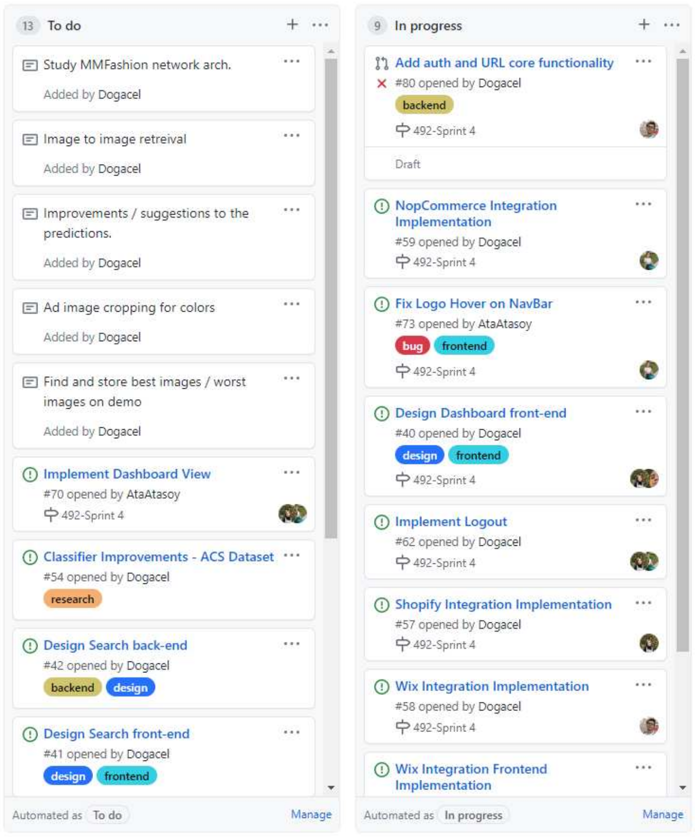
In order to speed up the development process, pair programming technique was used to help group members whenever they were about to with a codebase they were not familiar with. During these sessions, one group member shared their screen and acted as the "driver", where they gave a brief walkthrough of the code and continued working on the task, while the other group member acted as the "passenger", observing and reviewing the driver's work along the process. These roles were swapped through the pair programming session.

Taking lead role and sharing leadership on the team

Doğaç Eldenk was the group leader as he was responsible for implementing the core features and the general architecture of the codebase. We also tried to take turns in leading the Sprint Reviews to make sure every group member was aware of the current state of the project.

Meeting objectives

We tried to work similar to an Agile development team and perform Sprint reviews and plannings on every saturday of the week. Github Actions were used to create issues and assign group members for each issue and track their progress automatically under the To do and In progress columns.



13 To do

- Study MMFashion network arch. — Added by Dogacel
- Image to image retreival — Added by Dogacel
- Improvements / suggestions to the predictions. — Added by Dogacel
- Ad image cropping for colors — Added by Dogacel
- Find and store best images / worst images on demo — Added by Dogacel
- Implement Dashboard View — #70 opened by AtaAtasoy — 492-Sprint 4
- Classifier Improvements - ACS Dataset — #54 opened by Dogacel — research
- Design Search back-end — #42 opened by Dogacel — backend, design
- Design Search front-end — #41 opened by Dogacel — design, frontend

Automated as To do — Manage

9 In progress

- Add auth and URL core functionality — #80 opened by Dogacel — backend — 492-Sprint 4 — Draft
- NopCommerce Integration Implementation — #59 opened by Dogacel — 492-Sprint 4
- Fix Logo Hover on NavBar — #73 opened by AtaAtasoy — bug, frontend — 492-Sprint 4
- Design Dashboard front-end — #40 opened by Dogacel — design, frontend — 492-Sprint 4
- Implement Logout — #62 opened by Dogacel — 492-Sprint 4
- Shopify Integration Implementation — #57 opened by Dogacel — 492-Sprint 4
- Wix Integration Implementation — #58 opened by Dogacel — 492-Sprint 4
- Wix Integration Frontend Implementation

Automated as In progress — Manage

Figure: Github Actions

Above you can see the To Do and In Progress columns. When a group member links a pull request to an issue, the issue is automatically moved to the “In Progress” column.

New Knowledge Acquired and Applied

Most of the group had previous experience with web development using React and Flask frameworks. On the other hand, we did not have any previous experience on Neural Networks with PyTorch. Therefore, before trying to use MMFashion, we spent one week getting acquainted with PyTorch and Neural Networks through online tutorials. We read documentation about Google

Compute Engine to deploy the MMFashion module to Google Cloud Platform and integrate it with our back-end.

Risks and Alternatives

	Likelihood	**Effect on the project**	**B Plan Summary**
Wrong Category Prediction	Moderate	Wrongly categorized items will make a hard time for customers to find the items that they have searched for.	Human labelers can double check if the confidence of the prediction is low.
High Costs	Moderate	If the costs are too high, customers might not choose to work with us.	Find partners who are interested in investigating new technologies rather than small businesses with limited budget.
Outlier Inaccuracies	Low	Outlier items can be placed in a wrong category with wrong attributes.	Manually label outliers using human labelers.

8 Conclusion and Future Work

Also we would like to add improved search capabilities, database and inventory management systems per customer and feedback loop to our machine learning model. We are currently improving the search capabilities of the e-commerce websites by providing them more metadata and attributes per clothing item. But we want to improve that and include NLP techniques to improve the semantic search capabilities of the customer's website and further improve the customer experience.

Currently we provide a feedback mechanism to our machine learning model but the feedback is not being reused inside the neural network for improving its performance. This is due to fact we are not training the model ourselves because of the operational expenses. Once we can train our model, we can also use this feedback loop to further improve our model's capabilities. Even though the DeepFashion dataset contains over 220,000 images we believe that we can surpass that image count way faster by collaborating with our customers by involving the annotation reviewal process. Also customers can focus on their specific items and categories besides training a more generic model.

Finally we want to have an extended database and inventory management system for our customers to customize their similar image retrieval process. This platform can enable them to manage their inventories from our intuitive UI and further improve our neural network's learning process by observing the dataset our customers are exporting to our platform.

9 Glossary

- **Fashion Item:** Any type of clothing item that is sold in stores.
- **MMFashion:** The fashion annotation library used to annotate fashion images. [2][3]
- **Landmark Detection:** The detection of the landmarks of a clothing item from its image (i.e., its collar, hemline, etc.)
- **Attribute:** Attributes of a clothing product, detected via MMFashion (i.e., floral, long sleeve, striped, etc.)
- **Category:** Category of the fashion item, detected via MMFashion (i.e., summer dress, wedding dress, sneakers etc.)
- **Customer:** The e-commerce website owner who uses the Kalas-Iris services, unless otherwise stated (in some cases, those who purchase retail goods from e-commerce websites are also called customers, but in such cases it is explicitly stated. Otherwise, 'customer' has the meaning explained here).

10 References

[1]“THE 2020 BREAKDOWN OF THE ECOMMERCE INDUSTRY”, October 2020, https://www.etailinsights.com/online-retailer-market-size
[2] M. Moshin, ”10 ONLINE SHOPPING STATISTICS YOU NEED TO KNOW IN 2020”, 23 March 2020. [Online]. Available: https://www.oberlo.com/blog/online-shopping-statistics.

[3] Interview with Yağız Demirsoy, Hepsiburada Product Engineering Manager, 10 October 2020.
[4] “WooCommerce - Sell Online With The eCommerce Platform for WordPress”. *WooCommerce.* https://woocommerce.com. Accessed: October 2020.
[5] X. Liu, J. Li, J. Wang and Z. Liu, "MMFashion: An Open-Source Toolbox for Visual Fashion Analysis," Hong Kong, 2020.
[6] Z. Liu, P. Luo, S. Qiu, X. Wang and X. Tang, "DeepFashion Database," June 2016. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion.html.
[7] Z. Liu, P. Luo, S. Qiu, X. Wang and X. Tang, "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations," CVPR, Hong Kong, 2016.

Kalas-Iris User Manual

1) Build From Source

In this section we will explain how to build our project from source for exploration, testing or development.

Prefered development environment is Linux with 30GB+ disk space available. A CUDA supported GPU is required for running Machine Learning services such as image annotation and image retrieval.

Before starting you should fetch the repository from it’s source:

```
$ git clone git@github.com:Dogacel/Kalas-Iris.git
$ cd Kalas-Iris && git pull --recurse-submodules
```

Required Tools

Python3 (3.8 prefered)

```
$ sudo add-apt-repository ppa:deadsnakes/ppa
$ sudo apt update
$ sudo apt install python3.8
$ pip install virtualenv # Highly Suggested
```

NVIDIA CUDA drivers

Please install NVIDIA CUDA 11.0 drivers on your machine. Link

Node (14.4) and NPM

More info

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | b
$ nvm install 14.4 # Install the required npm version
```

Flutter 2.0.5

Install flutter from source and follow the instructions given.

Spin up an Android emulator or connect your Android device and enable USB debugging.

Run the application using the following command:

```
$ cd mobile
$ flutter pub get
$ flutter run
```

Ngrok

Install and setup Ngrok by following the instructions.

After following the instructions for installing and connecting your account. Use the following command to expose the port where the Flask App is running (5000 by default).

```
$ ./ngrok http 5000
```

You can also install and use the Visual Studio Code Extension of Ngrok. Follow the listed instructions and make sure to give the correct port number for the Flask App.

Pre-Build configuration

Download python packages

```
$ cd api # Locate to the api/ folder
$ python -m virtualenv venv # Create a virtual env for python
$ . venv/bin/activate # Activate the virtual env use don't forget to use 'deac
$ source venv/bin/activate # Mac users should use this to activate the virtua
$ pip install -r requirements.txt # Install python dependencies
```

Repeat the steps above again for the `mmfashion` folder as well.

Download MMFashion module dependencies

MMFashion setup is highly complex, you can always visit MMFashion Docs when you need.

Download Pre-Trained Models

First, visit the Model Zoo Page and download the following models:

- VGG-16 Backbone => `checkpoint/vgg16.pth`
- Attribute Prediction Coarse / ResNet-50 Global Pooling => `checkpoint/resnet_coarse_global.pth`
- Category Attribute Prediction Fine / VGG-16 Global Pooling => `checkpoint/vgg16_fine_global.pth`
- In-Shop Clothes Retrieval / VGG-16 Global Pooling => `checkpoint/Retrieve/vgg/global/epoch_100.pth`

Prepare Data

MMFashion Data Preperation Docs

Download DeepFashion Dataset and put it under `mmfashion/data` .

Set your file structure as the following:

```
In-shop
├── Anno
│   ├── segmentation
│   |   ├── DeepFashion_segmentation_train.json
│   |   ├── DeepFashion_segmentation_query.json
│   |   ├── DeepFashion_segmentation_gallery.json
│   ├── list_bbox_inshop.txt
│   ├── list_description_inshop.json
│   ├── list_item_inshop.txt
│   └── list_landmarks_inshop.txt
├── Eval
│   └── list_eval_partition.txt
└── Img
    ├── img
    |   ├──XXX.jpg
    ├── img_highres
    └── ├──XXX.jpg
```

And run `python prepare_in_shop.py` to arrange the dataset. For more information check MMFashion Dataset Docs.

Finally update your backend IP address to point your local under `web/src/api/api.js` and `api/flaskr/routes/image.py` .

MongoDB

Create a MongoDB instance on your local or on the Mongo Atlas cloud. Update your server paths under `api/flaskr/db.py`

After creating a database, create a .env file containing your username and password. It should have the following format.

```
$ DATABASE_USERNAME = "yourusername"
$ DATABASE_PASSWORD = "yourpassword"
```

For more on .env files, you can visit here and here

2) Running

API

```
$ cd api
$ sh run.sh # Or ./run.sh
```

Website

```
$ cd web
$ npm start # This might take a while on the first run. It will install depenc
```

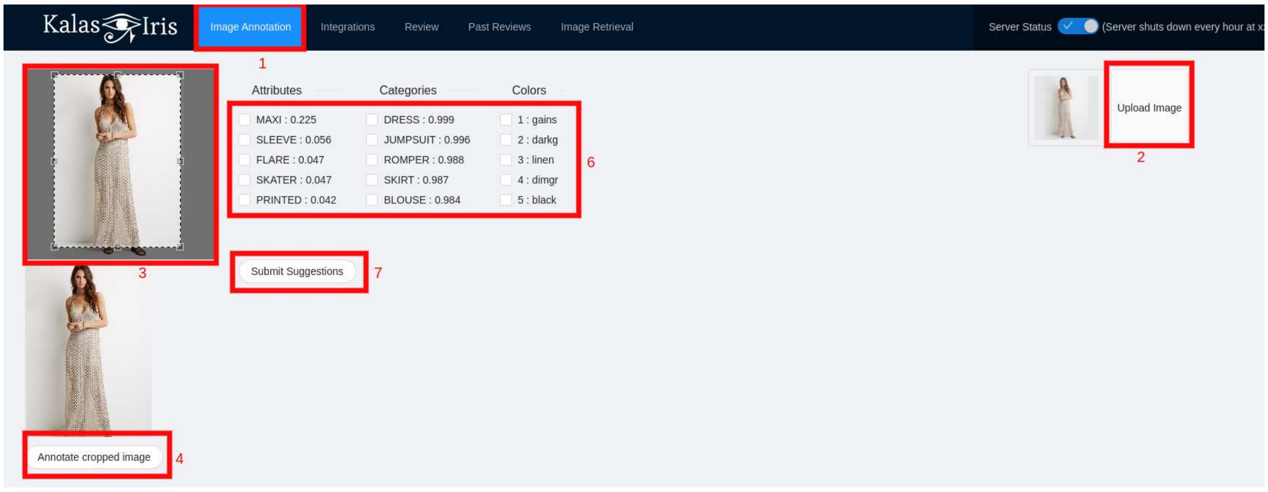
MMFashion

```
$ cd mmfashion
$ sudp python app.py
```

3) Using Web Interface

The website can be accesed at `http://localhost:5000` on your browser.

Annotating Clothing Images



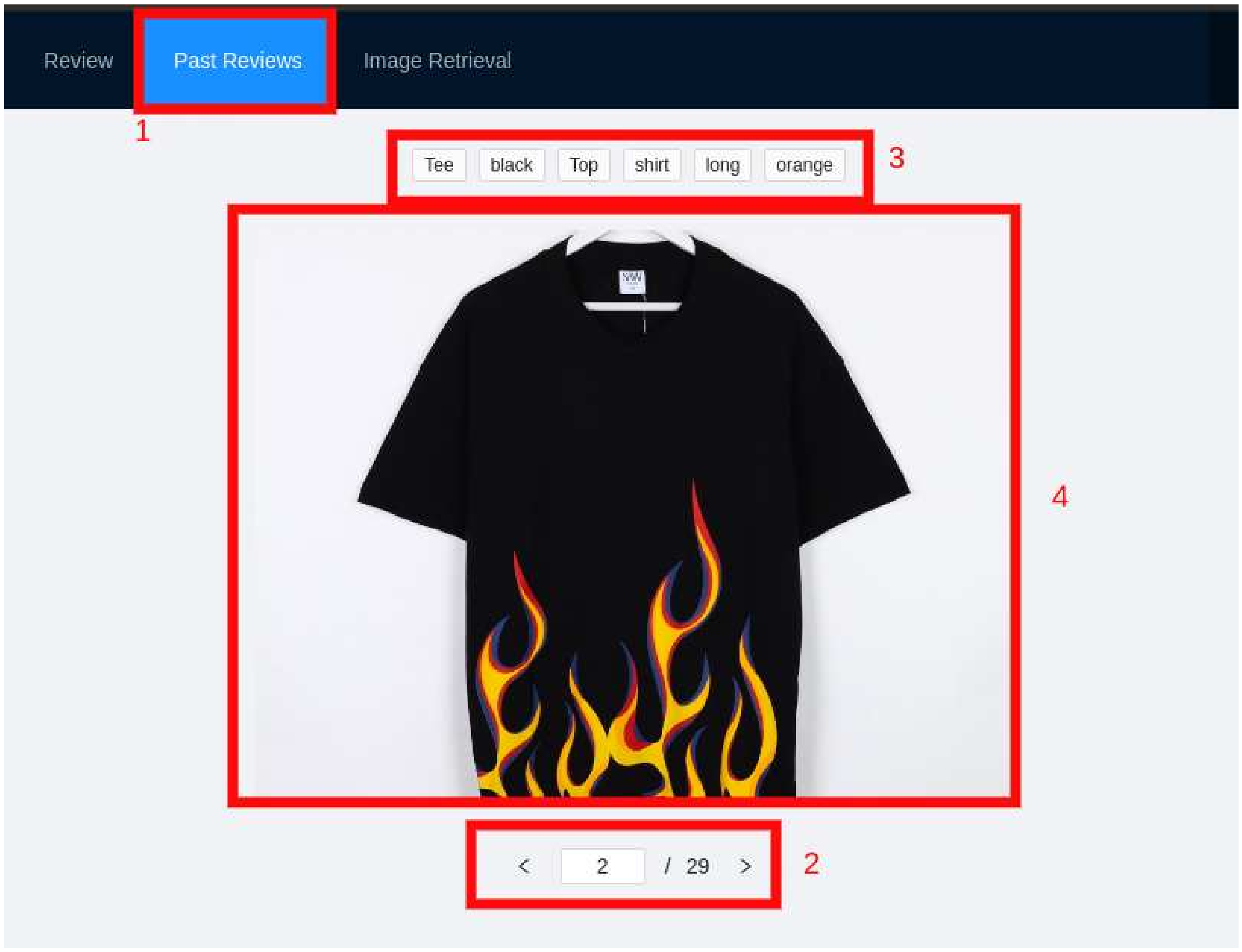
Attributes	Categories	Colors
MAXI : 0.225	DRESS : 0.999	1 : gains
SLEEVE : 0.056	JUMPSUIT : 0.996	2 : darkg
FLARE : 0.047	ROMPER : 0.988	3 : linen
SKATER : 0.047	SKIRT : 0.987	4 : dimgr
PRINTED : 0.042	BLOUSE : 0.984	5 : black

1. Click on the Image Annotation tab to access the image annotation page.
2. Click Upload image to upload a new image. Upload image will be automatically annotated. You can also hover over the images and delete or preview them.
3. Drag your mouse around the picture to draw a bounding box to the clothing item you want to annotate.
4. Click on the annotate cropped image button to re-annotate the image. This time only the cropped image will be annotated.

Suggesting Better Annotations

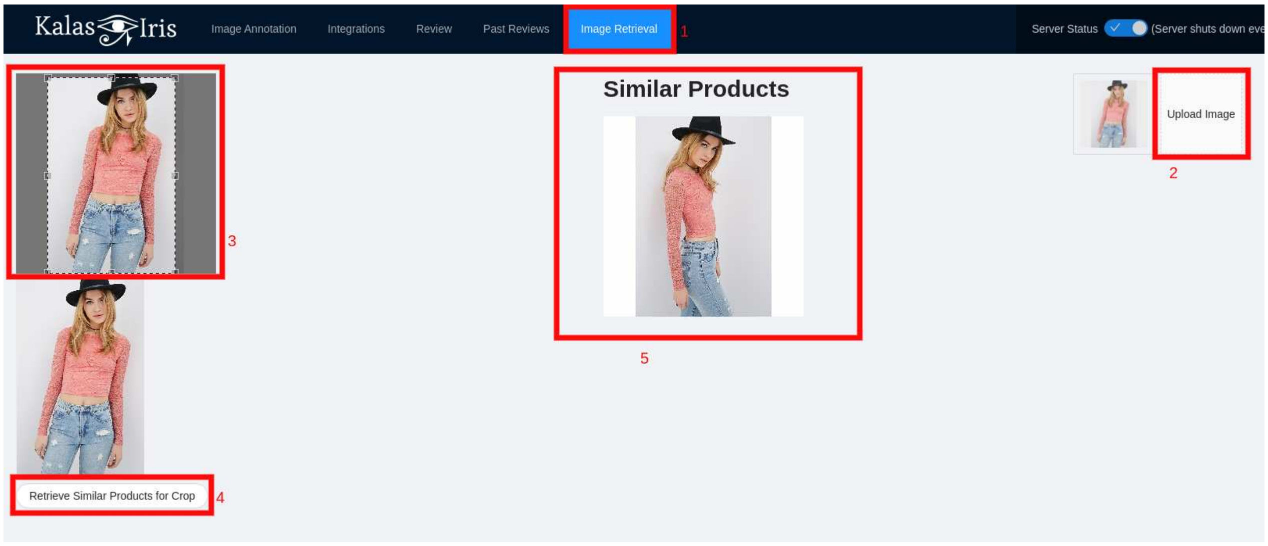
6. You can see the suggested Attributes, Categories and Colors from these colums. You select the correct annotations via the checkbox corresponding to it. The selected annotations can be saved.
7. Click the button in order to save the suggested annotations for the image. Those annotations are saved in our backend services for improving the performance of the model.

Reviewing Suggestions



1. Click on the ‘Past Reviews’ tab to see which suggestions has been made to the automatically annotated products.
2. Type the index of the item or paginate using the left and right arrows to see other suggested annotations.
3. The suggested annotations are shown on this top row.
4. The image of the annotation is shown here.

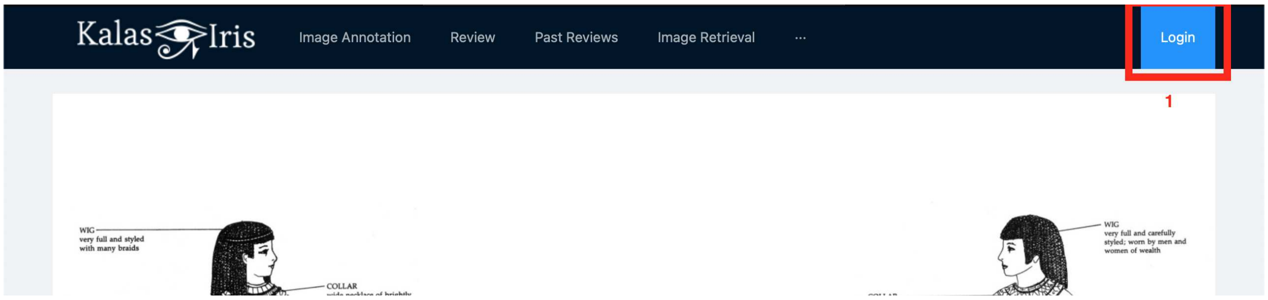
Retrieving Similar Products



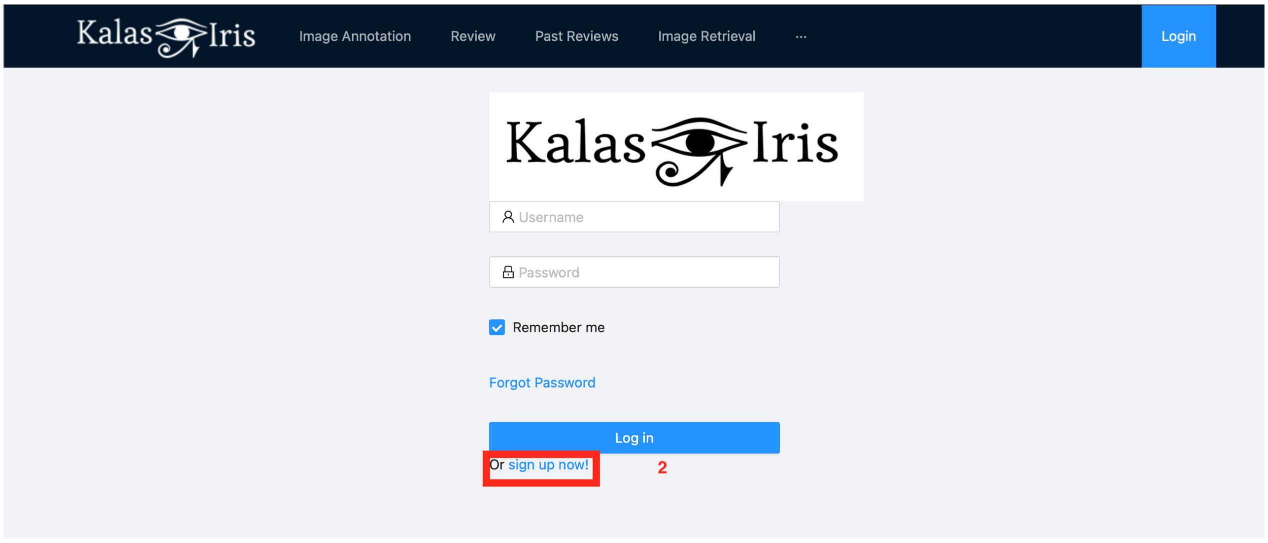
1. Click on the ‘Image Retrieval’ tab for searching the similar items by a given image.
2. Click Upload image to upload a new image. Upload image will be automatically used to find similar products. This process might take a while depending on your gallery size. You can also hover over the images and delete or preview them.
3. Drag your mouse around the picture to draw a bounding box to the clothing item you want to search for.
4. Click on the retrieve similar products for crop button to re-search the image. This time only the cropped image will be searched.
5. This slider shows you the retrieved similar items. You can use arrow keys to go right or left. Also the slider automatically changes images periodically.

4) Enabling Auto Annotation For E-Commerce Websites

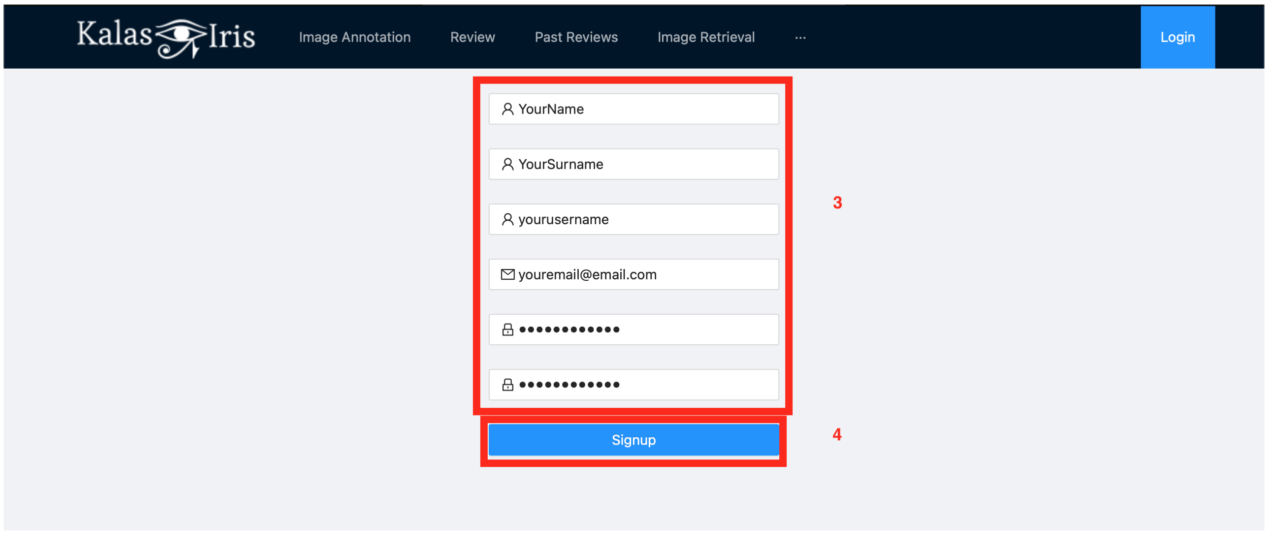
Registering To Kalas-Iris Services



1. Click on the “Login” tab.

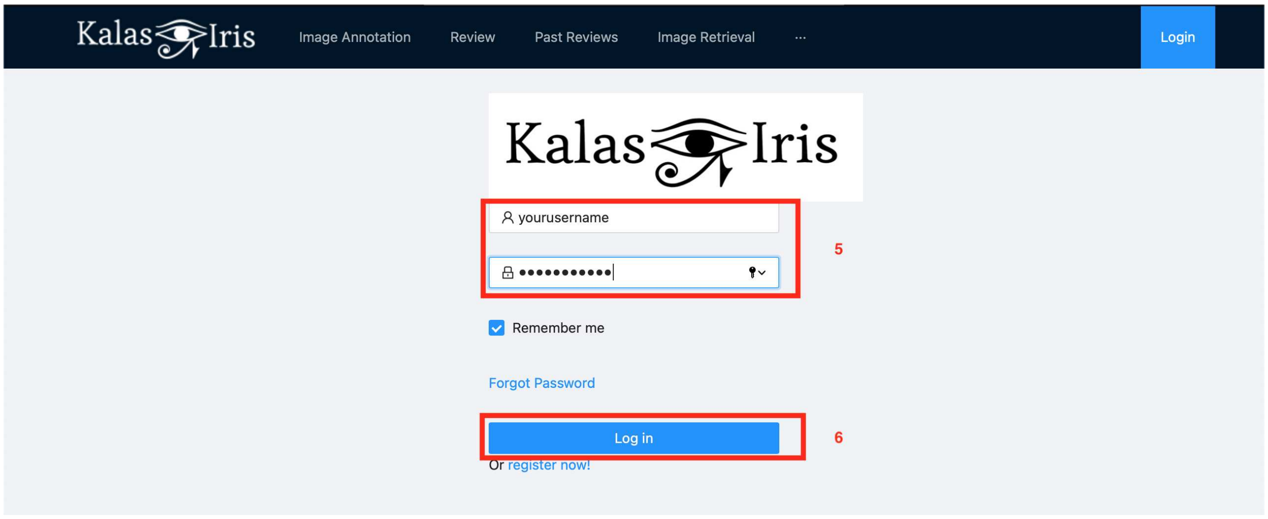


2. Click to “Signup Now”. It will redirect you to the signup page.



3. Fill the user credentials.

4. Click “Signup”. Upon success, you will be redirected to the login screen.



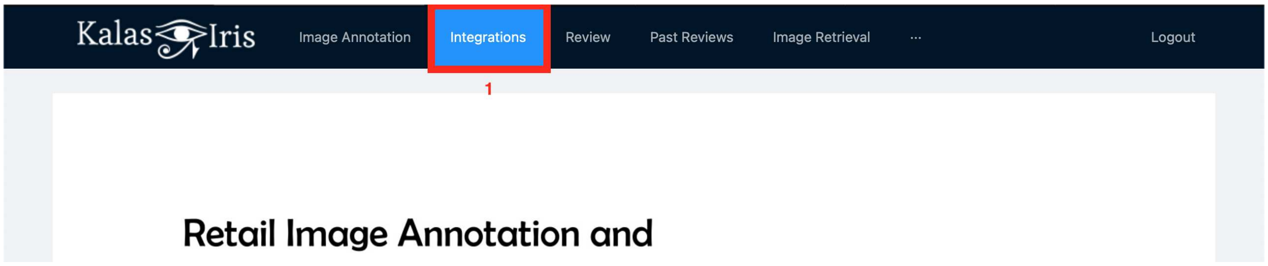
5. Enter your username and password.

6. Click login to enter to the system.

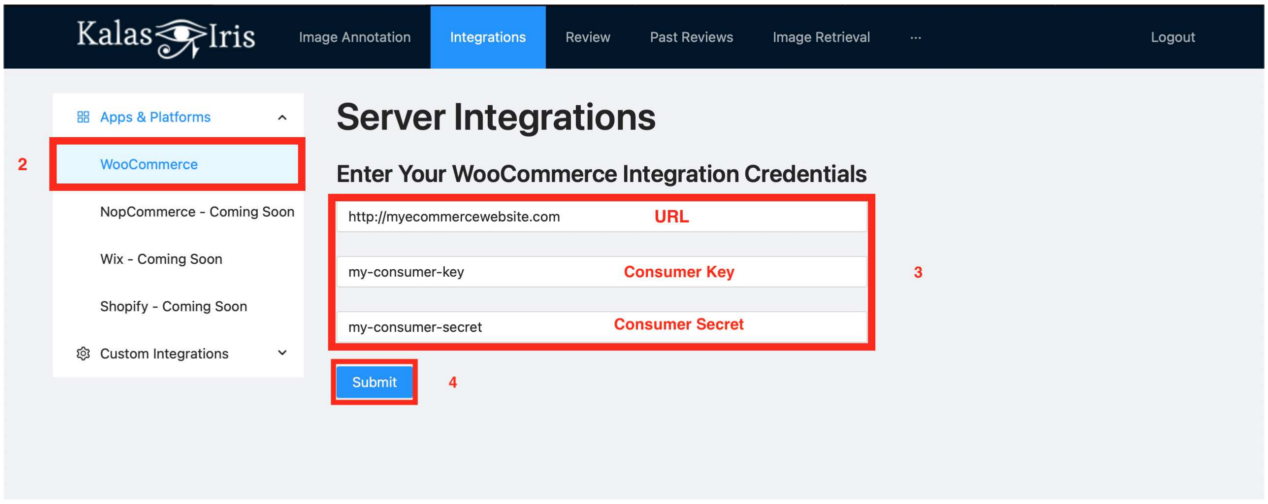
WooCommerce API Key setup

Refer to the WooCommerce Documentation to setup your WooCommerce API.

Adding API Key to Kalas-Iris



1. Click on the “Integrations” tab.



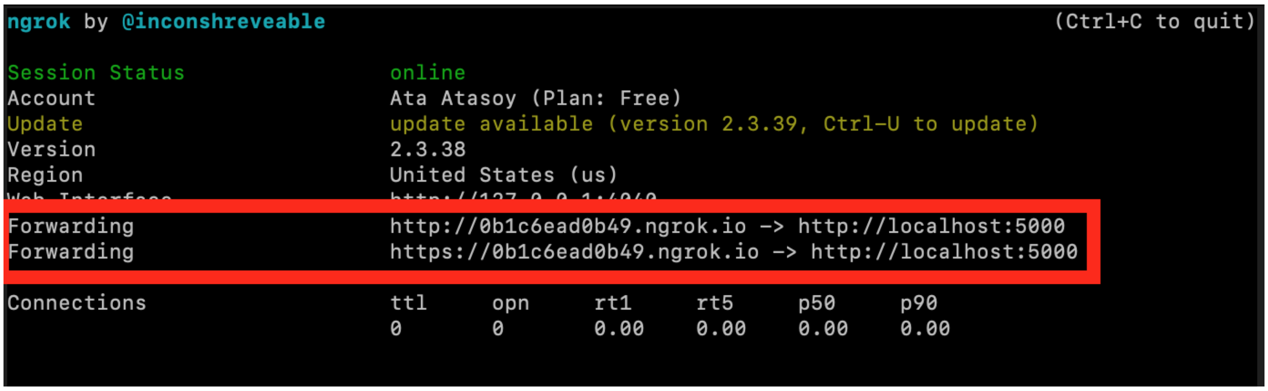
2. Click to “WooCommerce”.

3. Enter the url of your website, and the consumer key and consumer secret you receieved from WooCommerce.

4. Click submit to add your integration information to Kalas-Iris.

WooCommerce Demo Flow

1. Create a Webhook on WooCommerce by following these steps

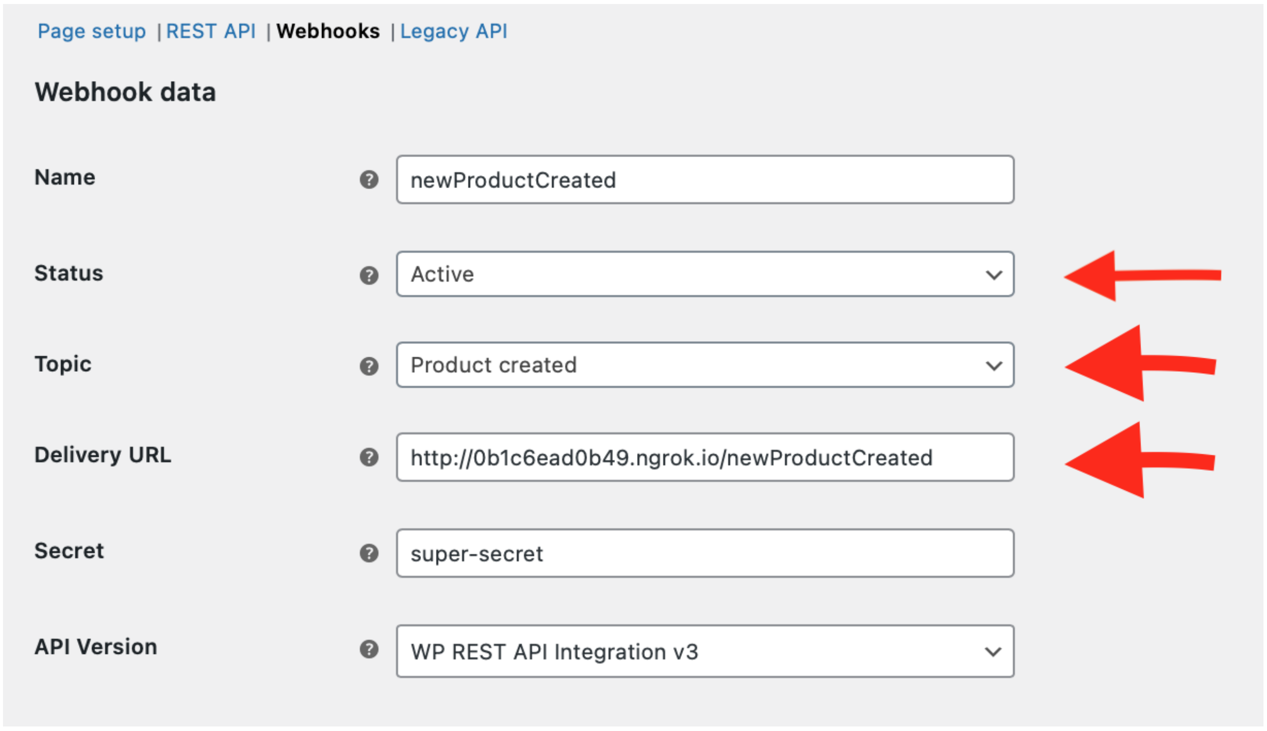


```
ngrok by @inconshreveable                                                    (Ctrl+C to quit)

Session Status                online
Account                       Ata Atasoy (Plan: Free)
Update                        update available (version 2.3.39, Ctrl-U to update)
Version                       2.3.38
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://0b1c6ead0b49.ngrok.io -> http://localhost:5000
Forwarding                    https://0b1c6ead0b49.ngrok.io -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

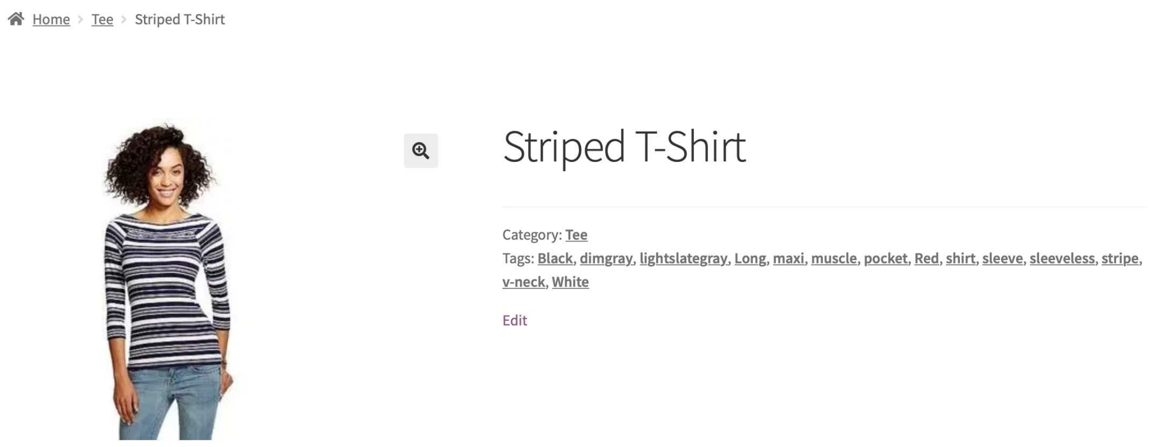
2. Launch Ngrok with the port number used by the Flask App. Ngrok will then give you the exposed URL for the back-end.


Page setup | REST API | Webhooks | Legacy API

Webhook data

Name	newProductCreated
Status	Active
Topic	Product created
Delivery URL	http://0b1c6ead0b49.ngrok.io/newProductCreated
Secret	super-secret
API Version	WP REST API Integration v3

3. Make sure you choose "Product created" for Topic and enter `http://$LINK_FROM_NGROK/newProductCreated` for the "Delivery URL" and "Active" for Status.


Home › Tee › Striped T-Shirt

Striped T-Shirt

Category: Tee
Tags: Black, dimgray, lightslategray, Long, maxi, muscle, pocket, Red, shirt, sleeve, sleeveless, stripe, v-neck, White

Edit

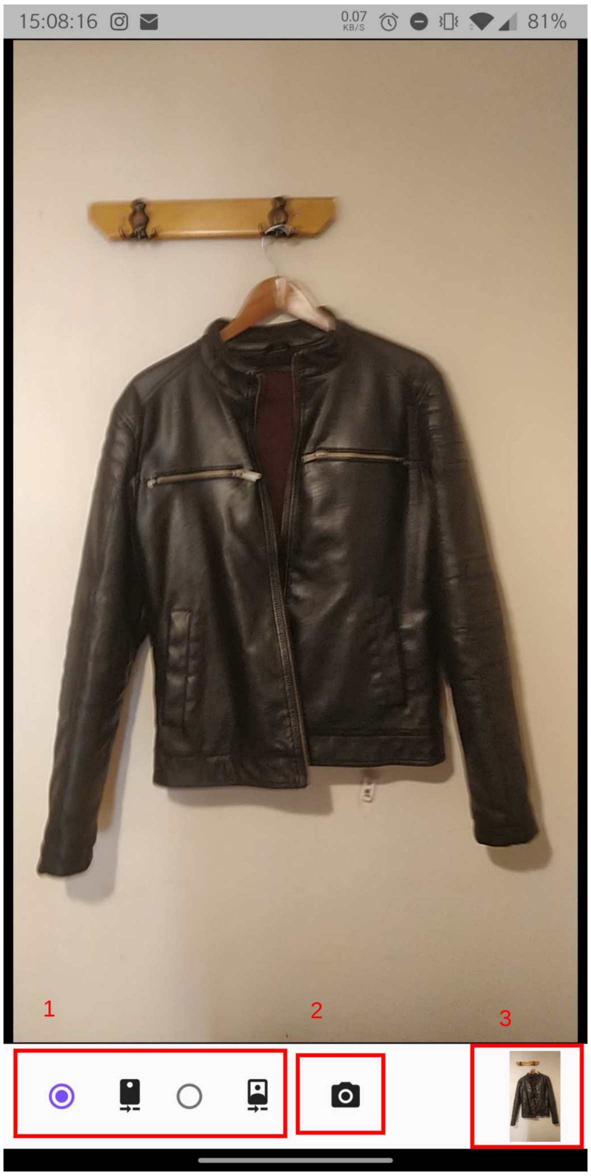
4. Congrats, now when you create a product with its name and image, it will be annotated automatically.

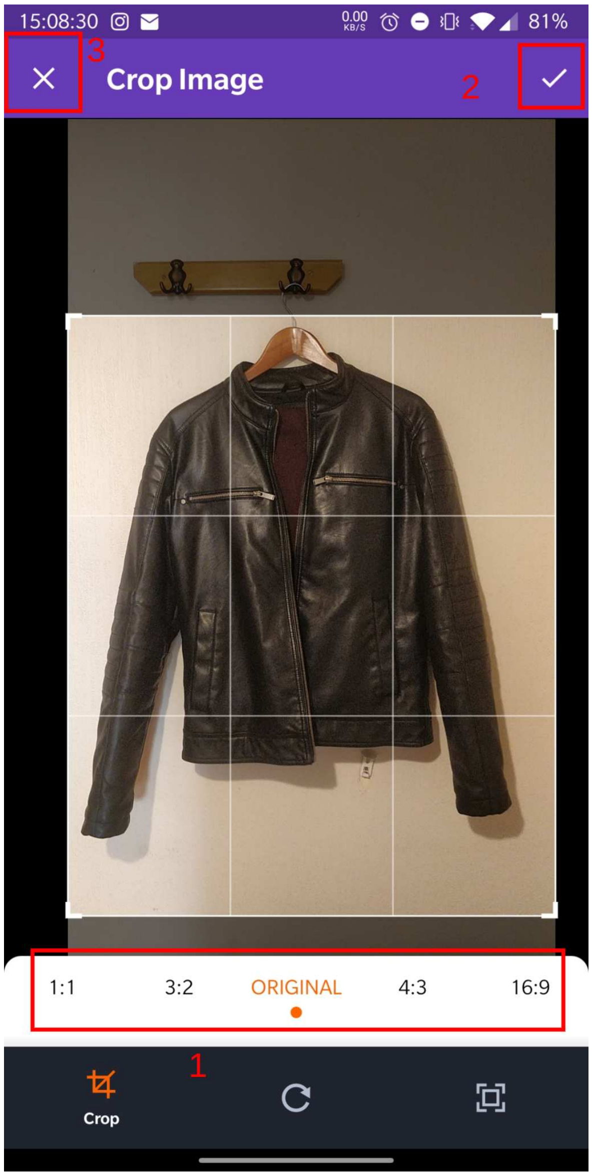
5) Using Mobile Interface

Installing the APK

Head over to http://kalas-iris.com/app.apk to download the APK for your device.

Taking and Editing a Photo





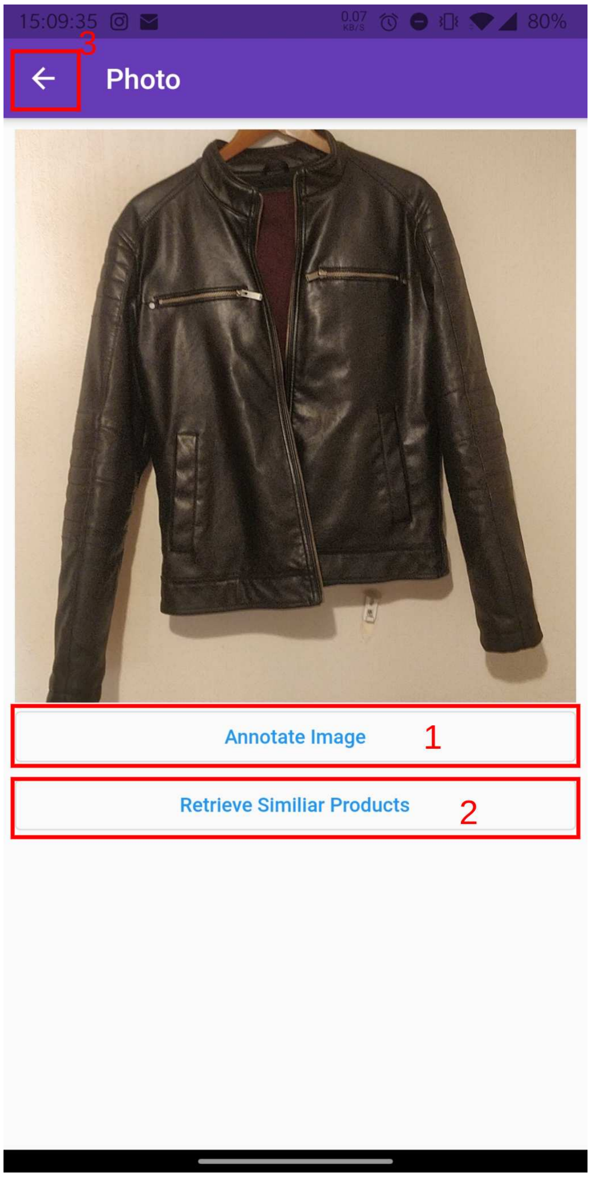
Once you open the app, you will be welcomed with a camera view.

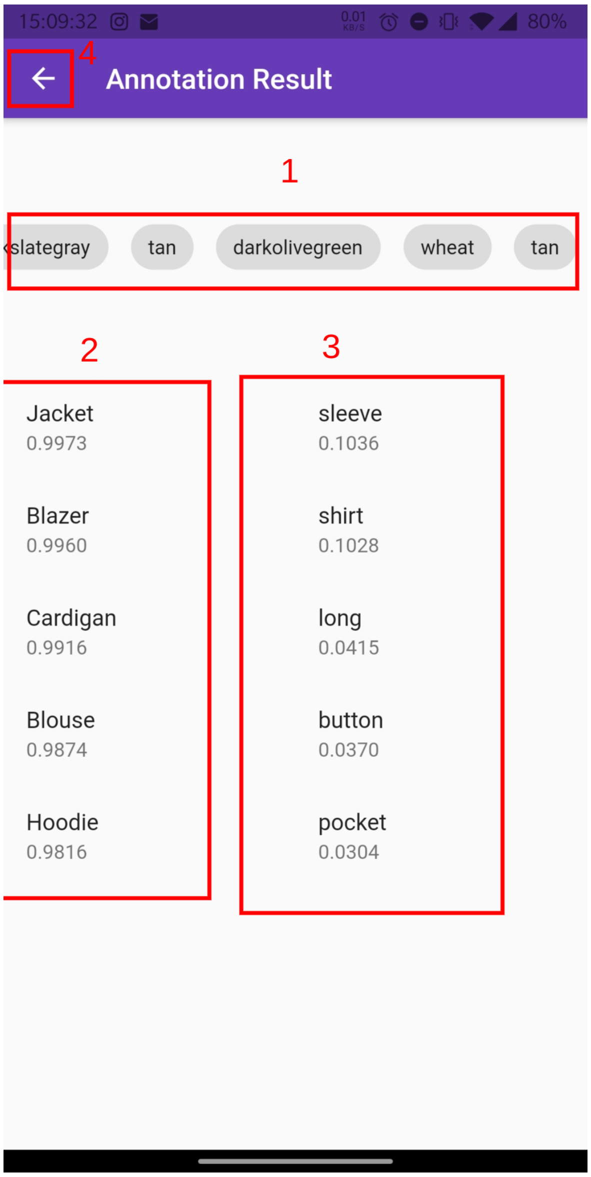
1. Select whether you want to use the front-facing camera or the back-facing camera.
2. Click the camera icon to take a picture. Taking a picture will automatically forward you to the cropping view.
3. You can see the last taken picture under this gallery view. Click on the picture to show image gallery view.

After you take a picture, you will see an image editor.

1. You can select the aspect ratio, cropped area and rotation using the tools provided on the bottom bar.
2. Click on the approval icon to continue with the cropped image.
3. Click on the cancel icon to use the uncropped image.

Using the Annotation Service





slategray · tan · darkolivegreen · wheat · tan

Jacket 0.9973	sleeve 0.1036
Blazer 0.9960	shirt 0.1028
Cardigan 0.9916	long 0.0415
Blouse 0.9874	button 0.0370
Hoodie 0.9816	pocket 0.0304

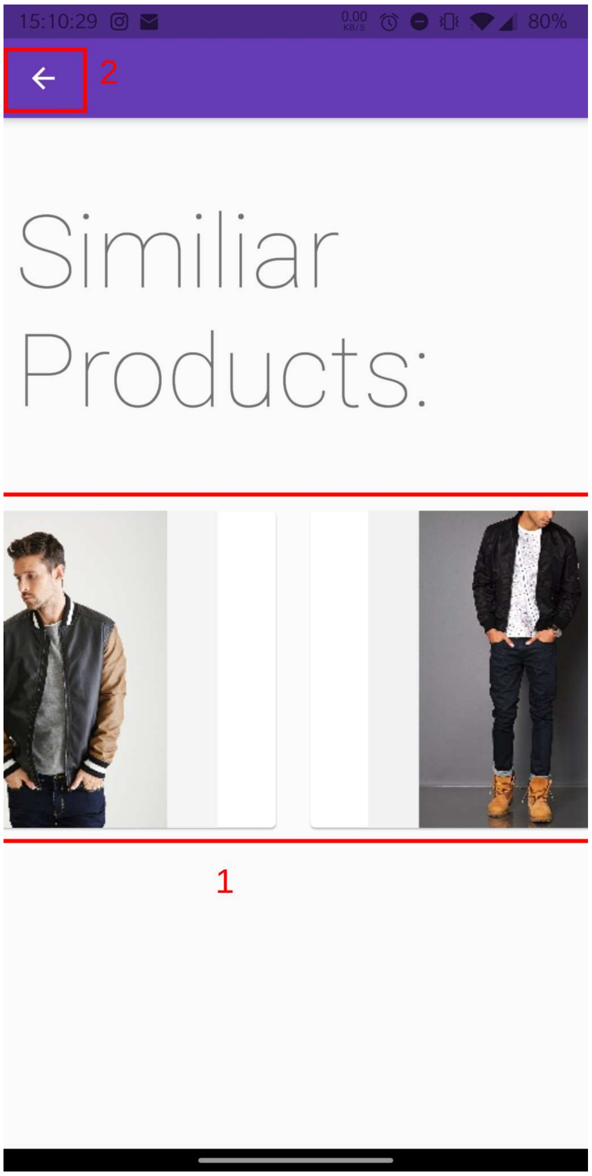
Once you crop your image, or click the button on the gallery view button on your camera view, you will see the image information page. In this page you can preview your image or use one of our services using the previewed image.

1. Click the Annotate Image button to annotate the image.
2. Click the Retrieve Similar Products button to search for similar products.

After you annotate your image you will see the annotation result page.

1. Found colors can be seen on the top row slider.
2. Predicted categories can be seen on the left column.
3. Predicted attributes can be seen on the right column.
4. Click to go back to the image preview view.

Using the Retrieval Service



1. A sliding widget shows you the similar images the service found.
2. Click to go back to the image preview view.