Team Lambeosaurus Project Report

By: Janishjit Bedi, Allen Nguyen, Marc Talalayevsky, Zachary Choo

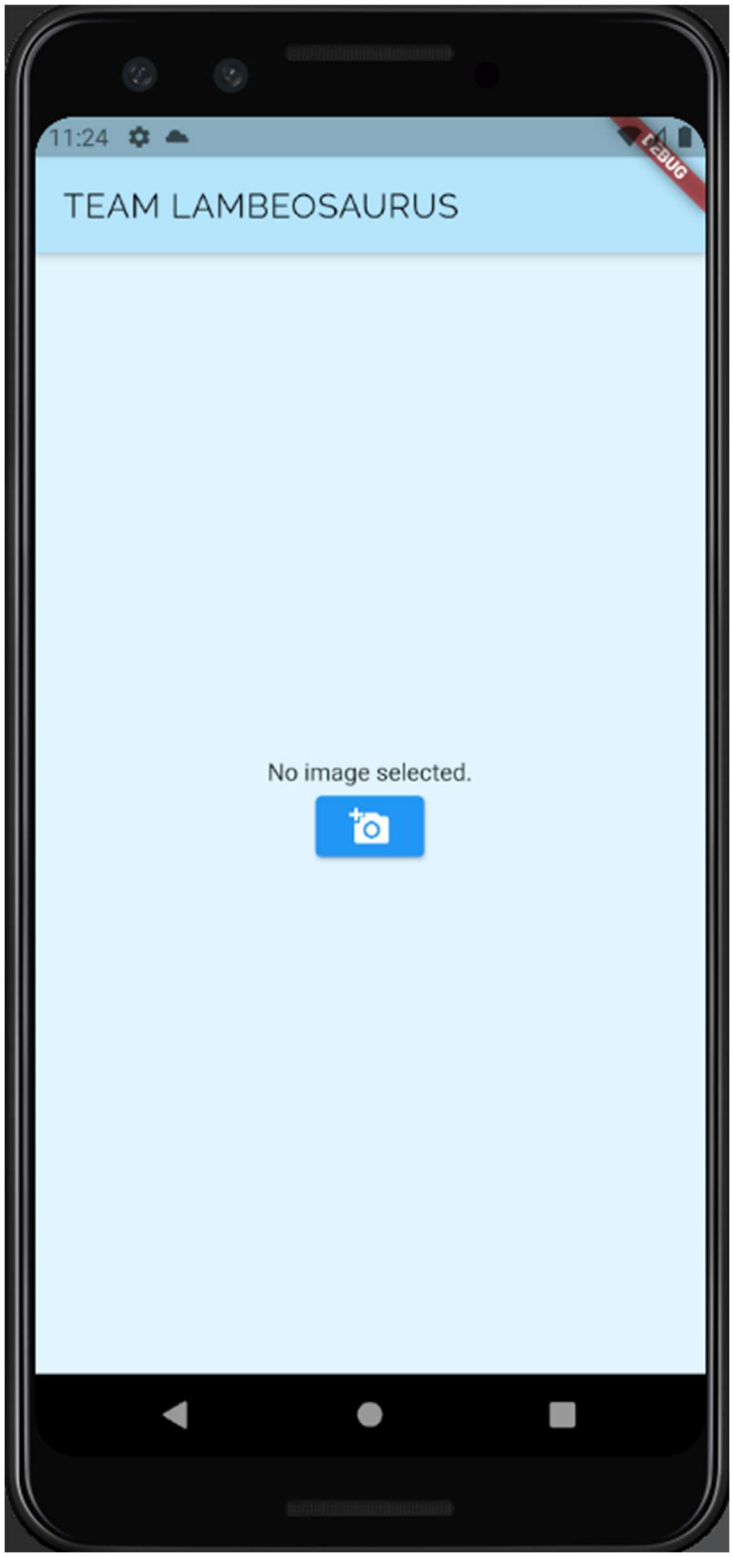
A brief description of the project.

Every sneakerhead has been in the situation where they see a unique shoe out in public and they desperately want to know what the name of the shoe is. The only method to find out what shoe it is is to type our observations on Google, but with over hundreds of brands and thousands of shoe models it is likely we do not end up with what we were searching for. To solve this problem, we have come up with the idea to create a mobile app called SeeKicks. SeeKicks takes a photo of the unidentified sneakers and uses machine learning classification to identify up to 899 different sneaker models. SeeKicks can identify sneakers from brands like Nike, Jordan, Adidas, New Balance, Asics, Vans, and many more. Once the shoe is classified, the app will open up its corresponding StockX link, a popular marketplace that shows the current price and details of sneakers.

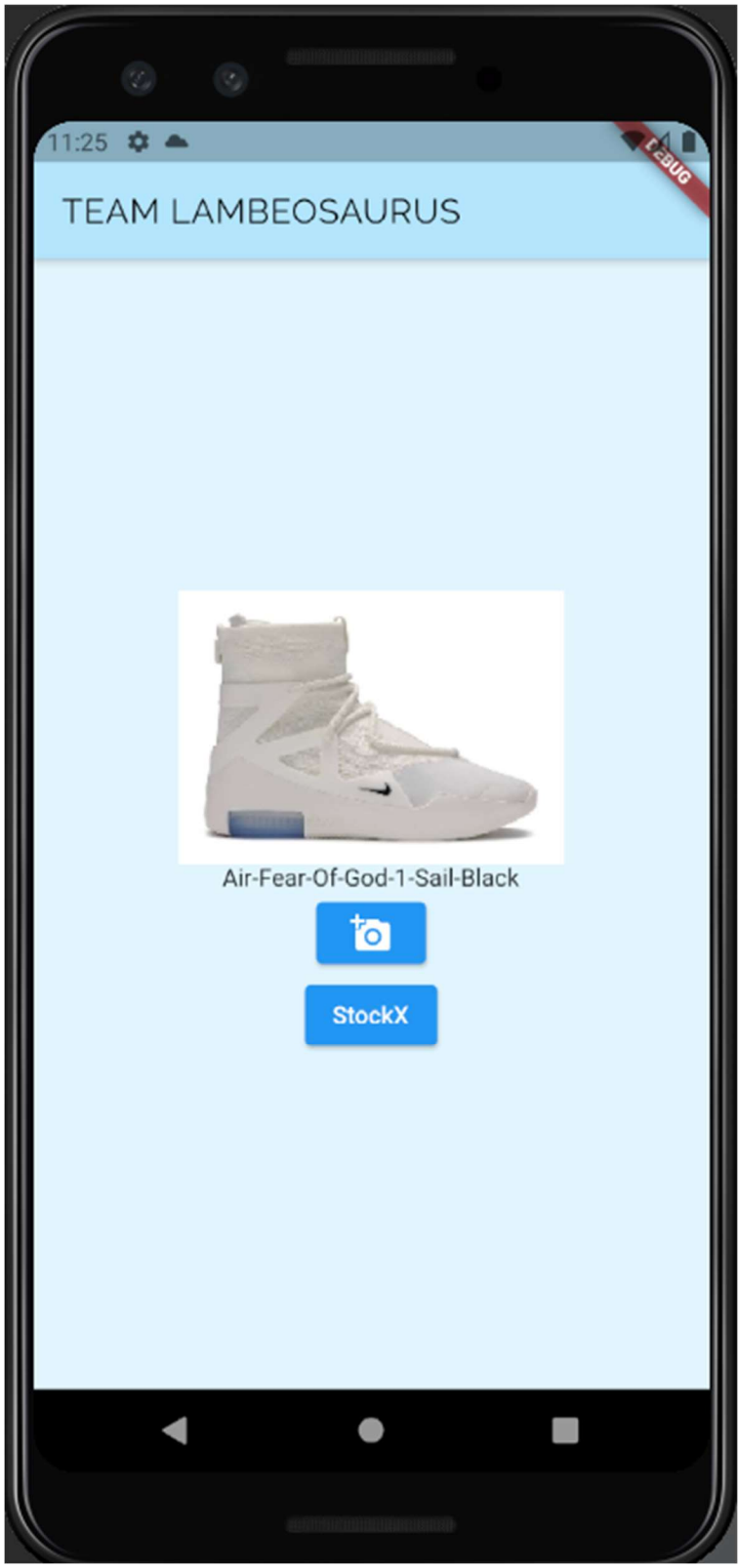
A walkthrough example of your project in operation, including pictures and/or measurements as appropriate. Imagine you're trying to convince an investor to fund your shiny new company.

SeeKicks mobile app

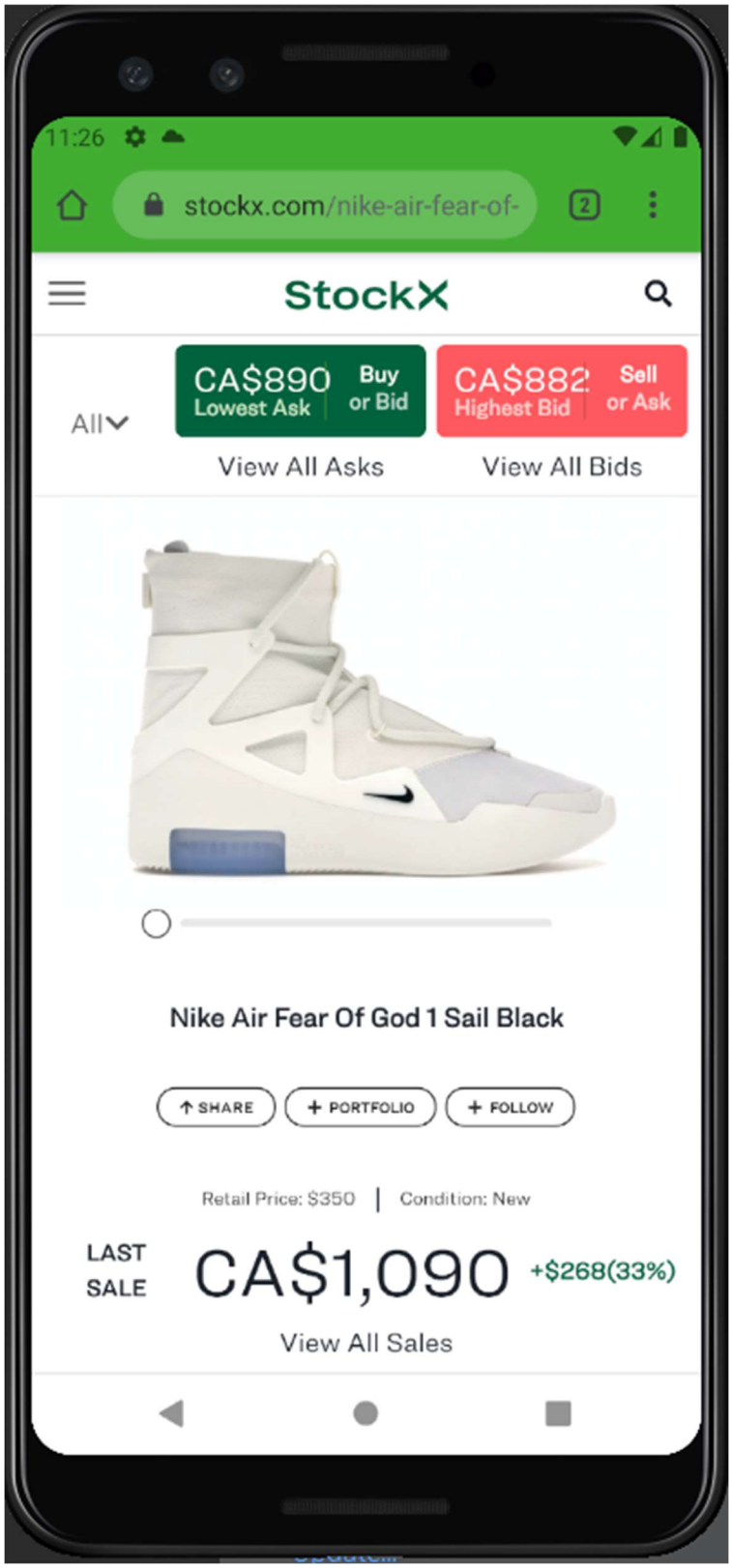
1. Open the SeeKicks mobile app on your Android device.



2. Take a photo of the unidentified sneakers using the built-in camera feature.

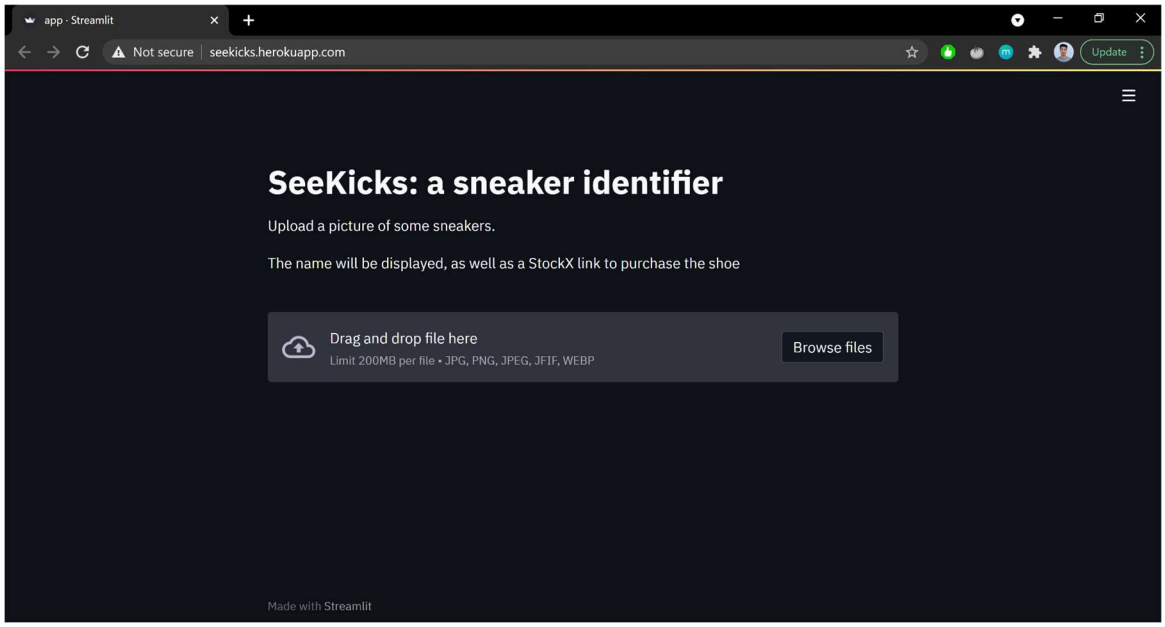


3. SeeKicks will tell you the name of the sneaker and a StockX link will be provided.

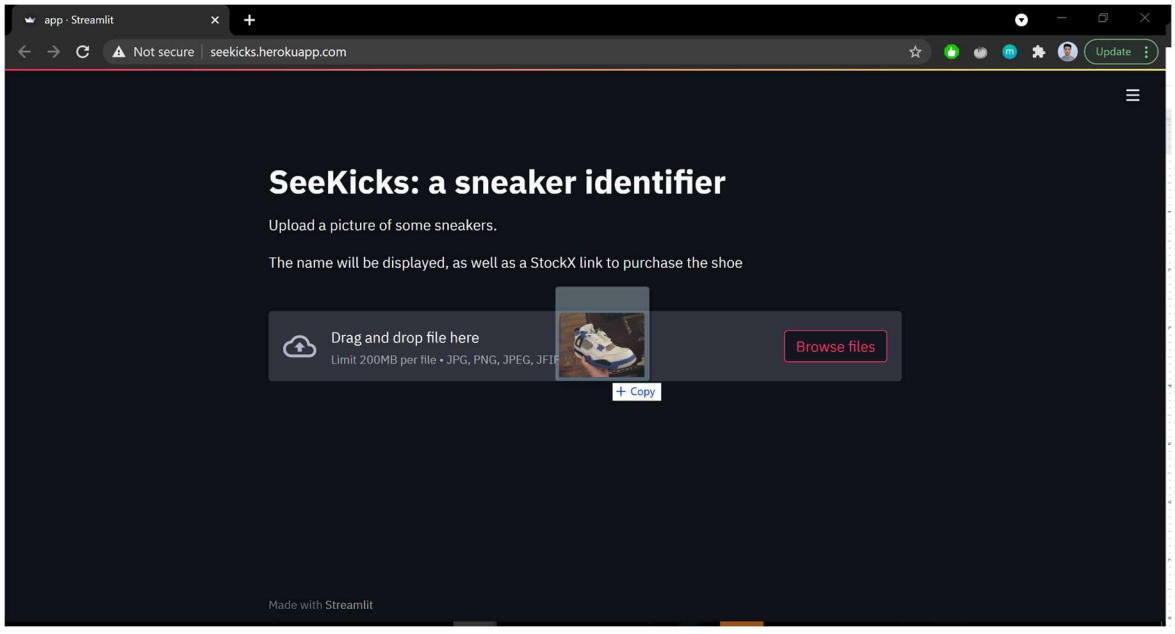


SeeKicks web app on a computer

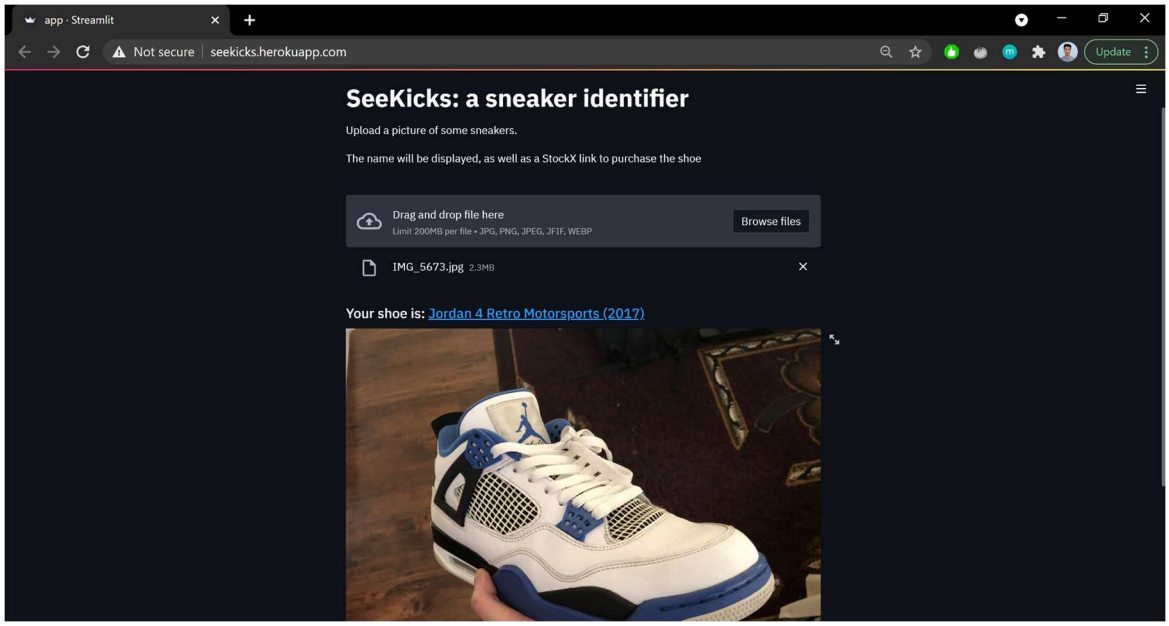
1. Open the SeeKicks web page on your computer by visiting the following URL, http://seekicks.herokuapp.com/.



2. Upload or drag and drop a photo of the unidentified sneaker.



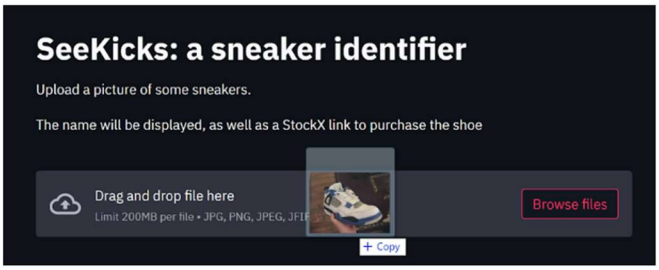
3. After just a few seconds, SeeKicks will provide you with the name of the sneaker and a StockX hyperlink.



A diagram showing all significant components in your system, plus how (and what) data flows among them. A component is something that does a well-defined high-level function and has a well-defined interface; for example, a single ML classifier will be one component (so if you classify two things you will have two), the web frontend / backend / database would likely each all be a module (maybe more than one), etc.

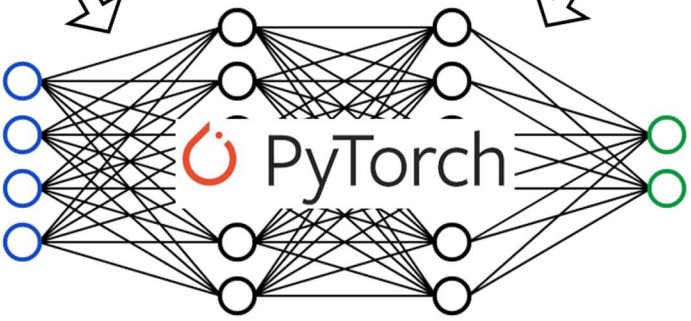
1. Photo of an unclassified sneaker is taken on the SeeKicks mobile app or uploaded to the web app.
2. SeeKicks feeds the photo into a machine learning algorithm through Flutter or Streamlit.
3. PyTorch machine learning classifier model analyzes and classifies the sneaker.
4. SeeKicks gives the user the name and a StockX link of the classified sneaker.


PORTRAIT
PHOTO
VIDEO


SeeKicks: a sneaker identifier
Upload a picture of some sneakers.
The name will be displayed, as well as a StockX link to purchase the shoe
Drag and drop file here
Limit 200MB per file • JPG, PNG, JPEG, JFI
Browse files
+ Copy


Flutter


Streamlit


PyTorch


stockX

A description of each component and how it was implemented (algorithms, abstractions, data structures, etc.)

PyTorch ML classifier:
The machine learning image recognition model was implemented using a pretrained ResNet-18 model using PyTorch. ResNet-18 is a convolutional neural network with 18 layers. However, for our purposes we replaced the 18 layers with five layers. While testing the model, we found that these five layers produced the highest accuracy. These five layers consist of three linear layers and two ReLU layers in the following order: Linear, ReLU, Linear, ReLU, Linear. The model was trained on a Kaggle dataset with over 50 thousand photos of sneakers. Furthermore, the dataset was expanded with three different data augmentations. These data augmentations include a rotation transform, perspective transform, and a horizontal flip transform. For the training process, we used a SGD optimizer, StepLR scheduler, and the DataLoader class with its optimized parameters. These optimized parameters were found by testing and experimenting with the model on a dataset with only 12 sneaker models. As a result, our final model reached an accuracy of 96.2% over 15 epochs with the entire dataset.

Flutter app frontend:
The mobile app frontend was implemented using Flutter. The mobile app has a built-in camera feature that allows the user to easily take a photo on the app itself on android devices or upload photos from the user's "Photos" app on iOS devices. The camera feature was implemented using a image_picker Flutter plugin. We used the pytorch_mobile plugin of flutter to implement our model onto the application, and the url_launcher plugin to access stockX websites. On providing an image the model predicts the shoe, and the application then displays the name and provides a stockX site where the shoe can be purchased.

Streamlit webpage:
The web app was implemented using Streamlit. The user inputs an image, and using the pytorch library, the image is turned into a tensor and fed into the model. The maximum number in the output tensor is taken as the model's prediction, and is matched with its corresponding label and stockX link. The web app displays the label, which is linked to the stockX site where the shoe can be purchased. We used Heroku to deploy the app on their servers.

The overall contributions of each team member to the project. This needs less detail than the milestone reports, we just want to know who was responsible for which parts.

Allen: Coded and tested parameters of the model for high accuracy, responsible for writing the final report.

Janishjit: Flutter application and research and implementation of various plugins. Also responsible for debugging.

Marc: Worked on the application and connected the model into it. Worked largely with Pandas and Selenium to automatically create resources for the application.

Zachary: Made and debugged the model, tested and trained the model, implemented and deployed the web app.