HermanMiller

Team Herman Miller
Michigan State University
FIBRE: Fabric Identification Based Recommendation Engine
Project Plan
Fall 2018

Herman Miller Contacts:
Tom Holcomb
Andrea Haggerty

MSU Capstone Members:
Josh Bhattarai
Ritwik Biswas
Joseph Smith
Ted Stacy
David Xuan

Table of Contents

1. Executive Summary

Herman Miller is an industry-leading furniture manufacturer based in Zeeland, Michigan. With customers in over 100 countries, Herman Miller manufactures a wide array of furniture such as chairs, sofas, and also office equipment. Winning awards for their customer service and satisfaction, Herman Miller prides themselves on catering to and customizing their products to the customers exact needs.

As artificial intelligence has advanced, Herman Miller continues to embrace the use of this technology to provide the best possible customer experience. Customers can now submit an image of the design they desire for their furniture, and Herman Miller matches that fabric to an exact or closely related material in their inventory. Currently, this is an expensive and time-consuming venture, as a team of designers need to sift through thousands upon thousands of materials to find the most closely related fabric.

Herman Miller identified this process as inefficient and tasked The Herman Miller Capstone Team with solving this problem using computer vision and machine learning. Using these technologies, our system accepts a user submitted image, categorizes the fabric in that image, and queries the Herman Miller textile inventory to find materials with closely related designs. This automated process greatly minimizes the amount of time Herman Miller employees need to spend manually comparing fabrics.

2. Functional Specifications

Currently, Herman Miller does not have a process in place for determining aesthetic/subjective fabric categories such as color or pattern. We developed an AI system, FIBRE, that utilizes computer vision and deep learning to recognize the color and pattern category of a submitted image. Our system has a web and mobile user interface to upload an image of a fabric, run the image through our machine learning models, and display the categorization of the fabric. Our AI is trained on 11,000 labeled fabrics from Herman Miller's subsidiary company Maharam.

A secondary goal of this project is to recommend alternative Herman Miller fabrics when a user submits a fabric of their own. This functionality is beneficial for Herman Miller when they receive an order from a customer with a proposed fabric. Currently, when a customer specifies that Herman Miller use a similar fabric, a manual search is done to find a Herman Miller fabric similar to the customers proposed fabric. This process wastes a significant amount of internal time and money of designers and testers. Our solution to this issue uses deep learning. Through the training process, FIBRE has learned from thousands of images how to find similar fabrics to suggest to a user. Our recommendation engine utilizes classification output from our machine learning models and encoded vectors from an autoencoder neural network to rank and suggest similar fabrics in a matter of seconds.

The last goal of this project is to provide a comprehensive API that Herman Miller employees can use to update the recommendation engine, get quick classification results, and scale our system. Our FIBRE API has multiple endpoints that support getting classifications/recommendations on an image and CRUD operations on the recommendations engine. We designed and documented this API keeping the developer in mind, making sure each of the endpoints are intuitive to use and have quick response times.

An examples use case of the FIBRE system would help illustrate the power of our AI system for Herman Miller's customers. Shelly is in the market to buy a new sofa for her living room. In her office there is a blue and green geometric tapestry that she really likes the design of. With our mobile app she can snap a picture of the tapestry and FIBRE instantly recognizes that the color classification is 60% blue and 40% green and the pattern style is geometric. She can then tap the "See Recommendations" button, and FIBRE instantly compares her image across thousands of Herman Miller fabrics. Using deep learning, FIBRE identifies four close alternatives to the fabric she submitted. She can choose a fabric that she really likes and immediately order a sofa with the fabric design. In this scenario, FIBRE reduced to the turnaround time of evaluating what similar fabrics Herman Miller has from weeks to a couple seconds. FIBRE's AI has also saved the designers the work of manually sorting and comparing fabrics to find close matches.

3. Design Specifications

3.1 Overview

FIBRE's design is based on a web and mobile app frontend, where end users can access the project, and also an API backend. The API backend allows different Herman Miller subsidiaries and developers to interact with the system. The web app frontend allows the end user to upload multiple images of fabrics. After uploading the images, the user can see each fabric's color and pattern classification. The user also has the option to view recommendations for each fabric on a separate page. The API backend does not have any UI, just API endpoints to access programmatically.

3.2 User Interface

3.2.1 Web Application

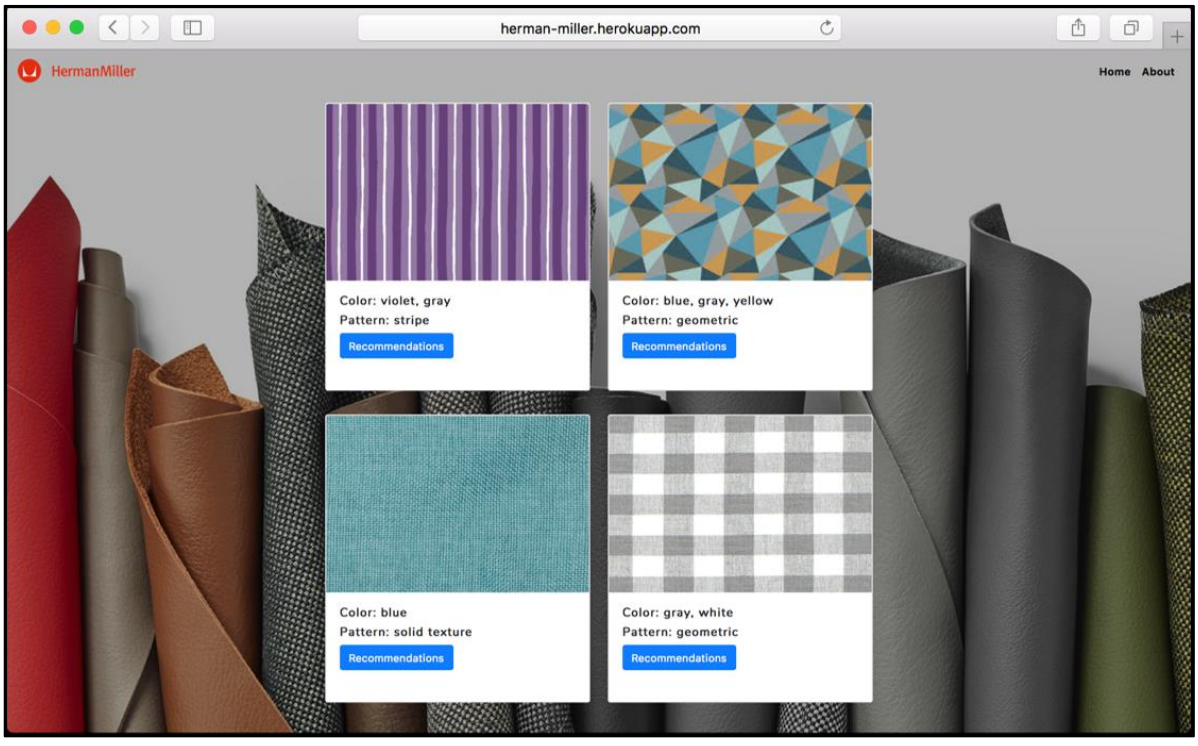

Figure 1.1: Classification Page

Figure 1.1 shows the classifications that the end user receives when they upload multiple fabrics. Below each fabric is the color and pattern classification our model assigned it. With the classifications is also a button that directs the user to the recommendation page for each fabric.

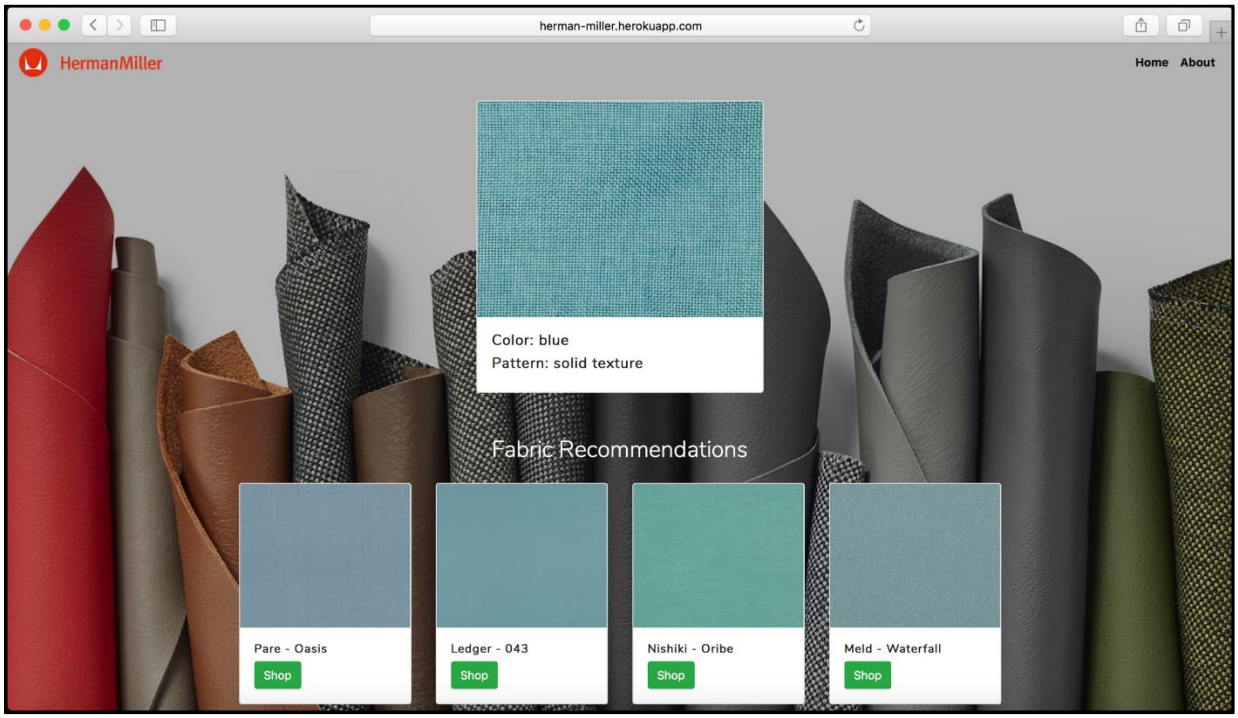

Figure 1.2: Recommendation Page

Figure 1.2 displays the recommendations for a single user uploaded image. The user receives up to four fabric recommendations that are all available for purchase from Herman Miller. Each recommendation provides the user with an image, name and a link directly to the shop page where they can purchase the recommended fabric.

3.2.2 Mobile Application

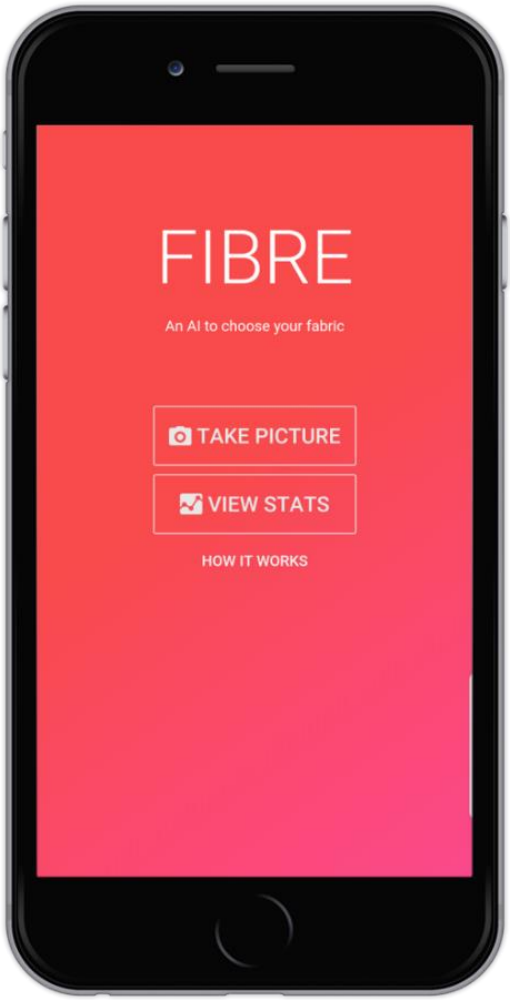

Figure 2.3: Mobile Home

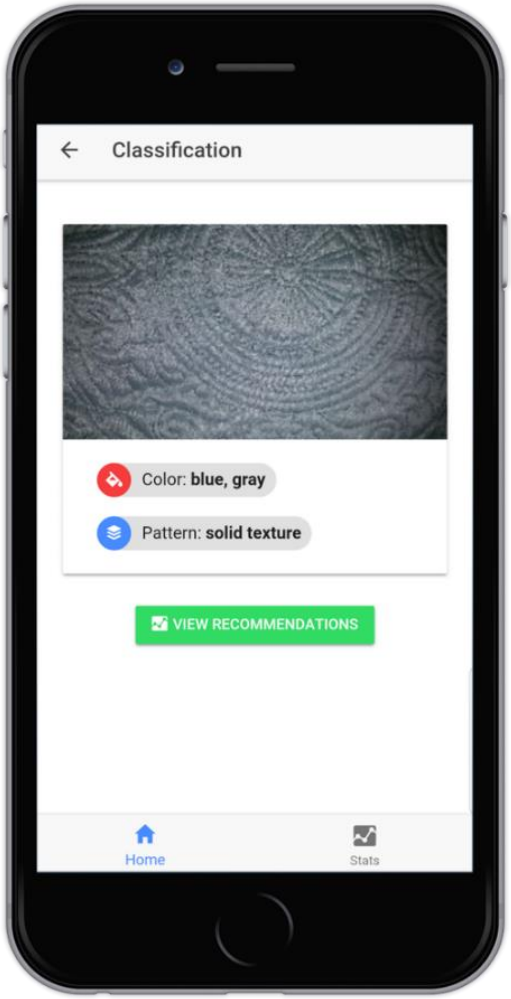

Figure 1.4: Mobile Classification

Figure 1.3 displays the mobile application home screen. The user can choose to take a picture of their desired fabric, view statistics about their previously submitted images, or gain more insight on how FIBRE works.

Figure 1.4 is the classification page the user sees after submitting an image. From here, the user can choose to return to the home screen, view similar fabrics on the recommendations page, or view statistics of their submission history.

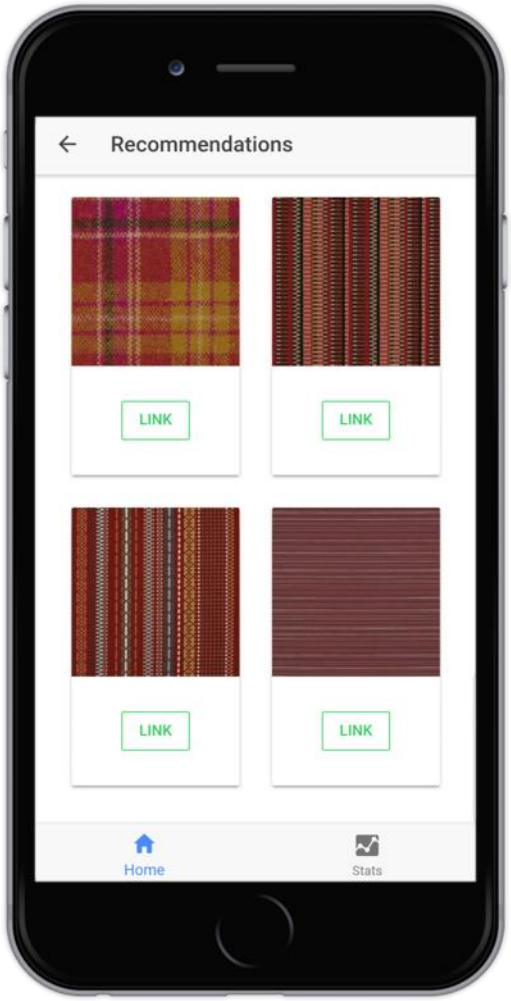

Figure 4.5: Mobile Recommendations

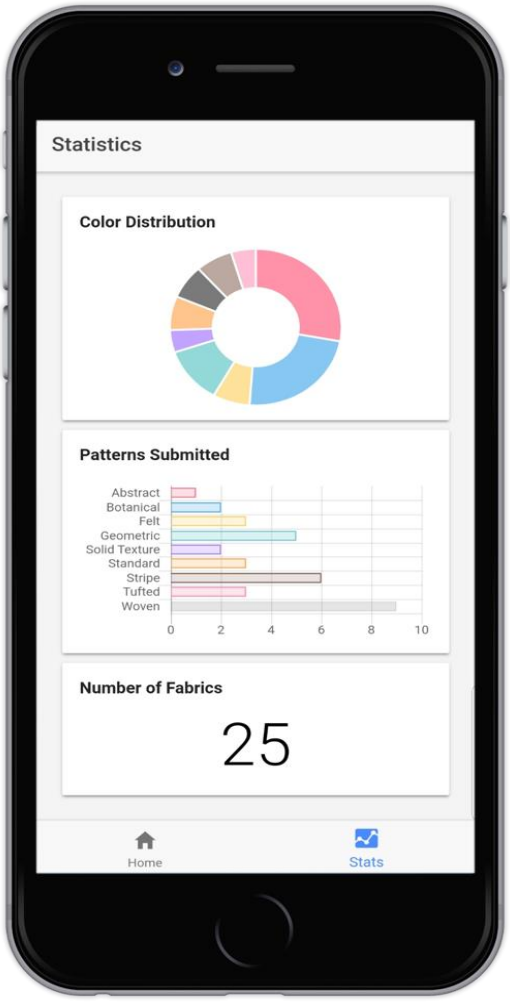

Figure 3.6: Mobile Statistics

Figure 1.5 displays the recommendation page, where users can view closely related fabrics to their original submitted image. The fabric's name is displayed to the user, as well as a link to purchase the textile directly from the vendor.

Figure 1.6 shows the statistics page, where users can view analytics about their previous submissions. Users can view the pattern and color distribution of their previously submitted images, as well as a count of how many times they have used FIBRE to categorize a textile.

4. Technical Specifications

4.1 System Architecture

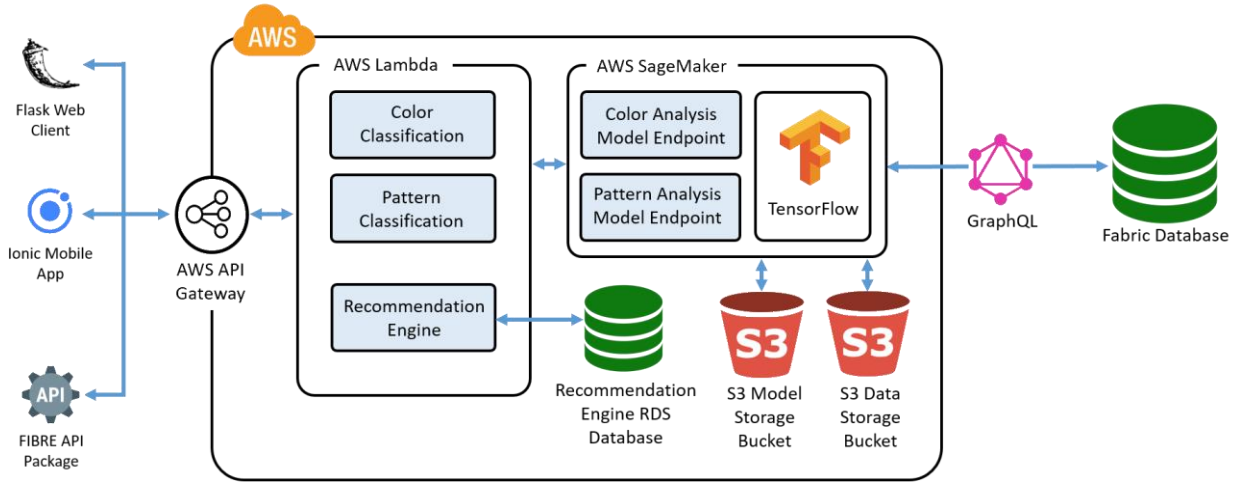

Figure 2.1: System Architecture Diagram

FIBRE contains three main components:

- Machine learning models to perform fabric identification
- A system that accepts an image of a fabric and returns the most similar fabrics
- An API that allows backend access to all components of FIBRE

A user can interact with FIBRE through a web or mobile app.

4.2 System Components

4.2.1 Machine Learning and Deep Learning

4.2.1.1 Color Classification

For our color classifier we trained a machine-learning model to determine the color distribution of images submitted by the users. We used an Adaboost classifier of size n=10,000 with Decision Trees as the base model. Adaboost is an ensemble classification method that stacks multiple simpler models together constantly adjusting weights of the sub-classifiers to achieve best class separation in the feature space. Our model was trained using 11,000 images from Herman Miller's fabric database. After tuning the hyperparameters of our model training and cleaning up poorly labeled data, our color classifier is currently at 98% test accuracy.

In order to support multicolor classification, we developed an algorithm to convolve over the image by splitting it up into multiple 10x10 pixel windows. Our color model classifies each window and stores its color classification. The aggregate of these color classifications allows us to determine the color distribution of the image. Our color classifier makes approximately 900 classifications per image in around 500ms and returns the color distribution.

4.2.1.2 Pattern Classification

For our pattern classifier we used transfer learning to retrain a convolutional neural network (CNN) and teach it to recognize custom Herman Miller patterns. The CNN we used is an industry image classification standard, ImageNet Inception V3. As an image passes through the layers of the neural network, the network learns to abstract high-level features such as lines, shapes, etc. When the image reaches the last layer or the bottleneck layer, we can feed in our pattern data and teach it to use the high level features to learn pattern distinction. We trained our Pattern neural network using TensorFlow on AWS SageMaker, which allows access to distributed AWS servers to reduce training time and optimize the hyper-parameter tuning. Our model is currently at 95% test accuracy. Both our classification models are served to our front end via AWS APIs (Lambda and Elastic Beanstalk).

4.2.1.3 Recommendation Engine

Using our recommendation engine, users can find close alternatives to fabrics they submit. Our recommendation engine has two primary components we use to determine the best recommendations.

Component 1: Classification Based Recommendation

The first step for building our recommendation engine is creating the database of images to choose from. Each one of Herman Miller's production fabrics is passed through both our pattern classifier and color classifier. We deconstruct the image into a thirteen dimensional feature vector based on the output of the classifiers. These feature vectors are then stored in an RDS database on AWS. We then take a user submitted fabric and run the same process to deconstruct it into another feature vector. After closely talking to our client contacts at Herman Miller, we developed a custom algorithm to rank the similarity between a user submitted fabric's feature vector and the vectors in the database:

$$s = -\frac{q_n \cdot p}{2} + \sum_{i=1}^{n} q_i |e_i - q_i|$$

Figure 2.2: Match Score Algorithm

s = match score, **q** = query vector, **e** = entry in database, **p** = 1 if pattern match, 0 if no pattern match, **n** = number of color classes in query image

Component 2: Autoencoder Based Recommendation

The second component of our recommendation engine involves a deep denoising autoencoder. An autoencoder is a neural network that is trained to deconstruct an input image into an encoded vector and reconstruct a noiseless version of the input. We trained an 18-layer neural network on 11,000 fabrics using TensorFlow. Throughout the training process the network learns what the important features of a fabric are, so it can efficiently encode the image into a smaller vector set than the original image. We can then pass all our recommendation images into our autoencoder and store the encoded layer vector in a database to be used for recommendations. When the user submits a query image, it is passed through our neural network and a basic K-Nearest Neighbors algorithm is used to find the most similar images in our database.

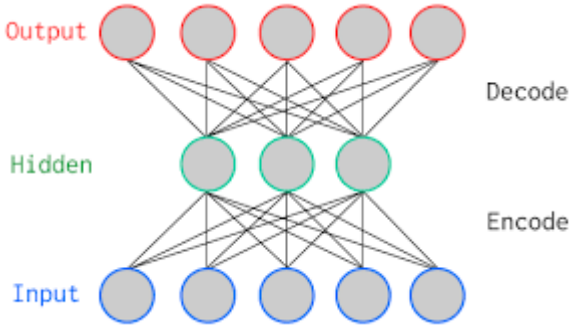

Figure 2.3: Basic Autoencoder Architecture

Through rigorous testing, we found that our classification based recommendation and autoencoder based recommendations tend to perform better in different scenarios. To provide the best results for our recommendation engine, we used a hybrid approach for recommendation. With certain pattern classifications, our recommendation provides the results from our autoencoder based recommendation and in other situations it provides our classification-based results. For both recommendation approaches, we designed our algorithms to maximize efficiency and ensure quick response times when a user submits a request.

4.2.2 Server Components

Our system runs on Amazon Web Services. Our system uses S3 for data storage, SageMaker to host the machine learning models, Lambda to interface to the models, and an API Gateway to deploy the machine learning models.

4.2.2.1 S3

S3 Buckets function as data storage buckets, and model storage buckets. The machine learning models are stored into S3 buckets by SageMaker and are easily accessible to SageMaker when running the model.

4.2.2.2 SageMaker

SageMaker functions as a hosting environment for the machine learning model scripts. SageMaker runs the machine learning model scripts, creates a model, stores the model in an S3 bucket, and publishes the model as an endpoint, accessible to Lambda.

4.2.2.3 Lambda

Lambda functions interface to the SageMaker model endpoints and provide an environment for combining and returning the results of multiple model endpoints to the frontend. Using Lambda functions provides us with a "serverless" infrastructure. The benefit of building FIBRE on a serverless infrastructure is that we do not need to configure any physical equipment. Further, when we deliver FIBRE to Herman Miller developers, they will be able to leverage the exact same serverless configuration that the system was built upon.

4.2.2.4 API Gateway

The API Gateway functions as the link between our frontend and our machine learning model endpoints. The web client can send requests to our deployed APIs through the API Gateway and receive the results of the machine learning models.

4.2.3 FIBRE API

The FIBRE API is a RESTful API system that delivers the full functionality of FIBRE within six endpoints. FIBRE API is built using AWS API Gateway and runs code through serverless AWS Lambda functions.

4.2.3.1 Classify

The /classify endpoint accepts a base 64 encoded image and returns the pattern match and the color distribution of the image.

4.2.3.2 Recommend

The /recommend endpoint accepts a base 64 encoded image and returns the pattern match, color distribution and the closes fabric matches that exist in our fabric database.

4.2.3.3 Insert

The /insert endpoint accepts a base 64 encoded image, a product name and a product URL. This endpoint calls the /classify endpoint, decodes the image, and stores the decoded image in an S3 bucket, returning a link to access the image by URL. The image URL, classification, product name, and product URL are inserted into the recommendation engine database. This endpoint returns with information on whether the insertion was successful.

4.2.3.4 List

The /list endpoint accepts no parameters and returns the contents of the entire recommendation engine database.

4.2.3.5 Update

The /update endpoint accepts no parameters and updates the classification values for every fabric within the recommendation engine database. Returns with information on whether the update was successful.

4.2.3.6 Delete

The /delete endpoint accepts the product URL of a fabric within the recommendation engine database as a parameter and removes that fabric from the database. Returns with information on whether the delete was successful.

4.2.4 Front End

The user interface component of our project is available as a web application and a mobile application.

4.2.4.1 Web Application

Our web application is created using the Flask client-side framework. The interface has the ability to accept user uploaded images and run the images against the machine learning models. This generates the color and pattern classification for the image and displays those classifications on the web application. Users can also elect to display similar fabrics, which are sent to the web application from the recommendation engine.

4.2.4.2 Mobile Application

Our mobile application was created using the Ionic SDK for both iOS and Android. The FIBRE mobile app provides the exact same functionality as the web application. An additional feature included in the mobile app is submission statistics. FIBRE users are able to view color and pattern analysis of their previous submitted fabrics.

4.2.5 External Data Systems

Herman Miller has many fabric subsidiaries. For our project, we access data from the Herman Miller subsidiary, “Maharam”. Maharam uses GraphQL, a schema definition language used for querying their fabric database. The Maharam database contains fabrics with metadata SKU and image location. The Maharam fabric data set is used to train the machine learning models.

4.3 Testing Plan

One major point of stress on our system is the requests calls made to our AWS API Gateway. We mitigate this concern through rigorous load testing. We created stress on the endpoints to collect metrics on their maximal efficiency load. We then load balance the system as necessary.

Another area of the system we tested is making sure only JPG, PNG, and BMP file formats can be uploaded to our user interface. If any other file formats are uploaded to our system, the system does not send the request to the API’s and the contents of that file are not opened. This ensures that no files with malicious intent are accepted into our system.

5. Risk Analysis

TensorFlow Integration to SageMaker

Difficulty: Medium

Importance: Medium

Description: As a group, we have a lack of knowledge on the individual parts in Amazon Web Service. The AWS technologies that we need to learn are EC2, S3, and SageMaker. We are not sure how to create AWS Instances that are accessible by everyone, deploy code that runs as intended, and connect these services together. Since these are vital technologies needed to complete our project, this is a major priority item that needs to be completed early.

Mitigation: Each member of the group is completing tutorials on each service and also creating prototypes to test out each one. We are also attempting to connect these prototypes together as to replicate the environment we will be using in our final project.

Image Tags are not Consistent

Difficulty: High

Importance: Medium

Description: Maharam is providing our image training set of over 10,000 images. Each image is tagged with category information that is used to train the machine learning model. However, some tags are inconsistent which results in unpredictable training and subpar models.

Mitigation: Discussed with client, given permission to modify the data set and remove misleading data.

Minimizing Image Size

Difficulty: Easy

Importance: Medium

Description: The average size for each image used in our training set is roughly 1.1 MB but with thousands of images this quickly adds up to a very large amount, reaching over 30 GBs! We needed to reduce the average size of each image so that we could store more images in both our local memory storage and reduce the cost of our AWS S3 service.

Mitigation: We wrote a Python script that automatically resizes each image in our dataset by an average of 95%. This allowed us to store over ten-fold the amount of images in the same amount of memory.

Pattern Scale Category definition is unclear

Difficulty: High

Importance: Low

Description: Each image is defined by three categories, color, pattern and pattern scale. Color and pattern have relatively straight forward definitions but pattern scale does not.

Mitigation: Look into EXIF or other image metadata that will determine scale or request scale in API, there are also other computer vision theories

6. Schedule

Week 1 (8/29 – 9/2)

- Initial meeting with group members
- Created team Slack, GroupMe and GitHub
- First Herman Miller Visit
- Began research on TensorFlow/Keras, GraphQL, and AWS Services
- Setup AWS EC2 Instance and Tom created our AWS accounts

Week 2 (9/3 – 9/9)

- Continued research on above technologies
- Created first AWS S3 instance, uploaded test images and connected to EC2
- Ran initial TensorFlow tests on images scrapped from Maharam website
- Completed Status Report Presentation

Week 3 (9/10 – 9/16)

- 9/12: Status Report Presentation

Week 4 (9/17 – 9/23)

- Finish Project Plan Document and Project Plan Presentation

Week 5 (9/24 – 9/30)

- Project Plan Presentation
- Build web application front-end
- Wrap ML Models in Flask APIs
- Configure SQL Database
- Prototype Recommendation Engine

Week 6 (10/1 – 10/7)

- Connect web application to ML Models and Recommendation Engine
- Classify all Herman Miller images with ML Models and generate metadata
- Configure Recommendation Engine to handle client requests

Week 7 (10/8 – 10/14)

- Finalize the user interface design
- Increase accuracy, efficiency, and retraining process of ML Models
- Deploy Recommendation Engine version 1.0

Week 8 (10/15 – 10/21)

- Alpha Presentation
- Created Autoencoder architecture

Week 9 (10/22 – 10/28)

- AWS Migration
- Optimize Machine Learning Models
- Mobile Development
- Autoencoder Training

Week 10 (10/29 – 11/4)

- AWS Migration
- Interface web client to AWS API Gateway
- Optimize Machine Learning Models
- Improve UI/UX on front-end
- Mobile Development
- Autoencoder Training

Week 11 (11/5 – 11/11)

- AWS Migration
- Interface web client to AWS API Gateway
- Optimize Machine Learning Models
- Improve UI/UX on front-end
- Mobile Development
- Autoencoder Training

Week 12 (11/12 – 11/18)

- Beta Presentation
- Create Project Video
- Design Day Presentation

Week 13 (11/19 – 11/25)

- Create Project Video
- Design Day Presentation
- Software Testing

Week 14 (11/26 – 12/2)

- Create Project Video
- Design Day Presentation
- Software Testing

Week 15 (12/3 – 12/9)

- 12/3: Submitted Project Video
- 12/5: Submitted All Deliverables
- 12/7: Design Day