Colorado School of Mines

Field Session

AirBespoke Inventory Tracking System

Client: Kylen McClintock

Written by: Peter Palumbo, Kyle Thistlewood, Nhan Tran, Minh Vu

June 22, 2016

Contents

1 Introduction

1.1 Client description

AirBespoke is a Golden-based startup that provides access to world-class tailors in developing countries. AirBespoke's first-of-its-kind platform removes all language and digital connection barriers for their tailors to sell online; all while allowing customers in the developed world access to custom fitted suits that are affordable. Their second aim is to usher in a new era of smart clothing by integrating highly functional wearable technologies into AirBespoke suits that are immensely valuable to the wearer. At AirBespoke, they believe that by democratizing technology they can lift thousands of tailors out of poverty and 'Suit-up' thousands more in the process.

AirBespoke is currently developing an E-Commerce platform which supports transactions between suit tailors and international customers. The website provides tailors and customers with an easy way to achieve what they want (Tailors can sell their suits for an affordable price, and customers can get their bespoke suits with reasonable budgets). Prior to this project, AirBespoke already had several components completed, including a visual website, connections with tailors, and other necessities for the primary goal they are trying to achieve. They wanted our team to implement the functionalities of their website that supports the tailors to track their inventories, the customers to place and track their orders, and also the admin to manage the site in an user-friendly manner.

1.2 Product Vision

The ultimate goal of this summer field session project was to create a full-stack website that provided the AirBespoke tailors with an intuitive inventory and order tracking platform interface that was inherently valuable to their business. As the product was intended to be used by non-Computer Science (CS) clients, user-friendliness was a crucial component that determined our success. Our website must have full functionalities as an E-Commerce website, which allowed tailors to upload their information, to track their inventories, and to process the payment. Additionally, this system must be able to match a specific customer order to the tailors that currently possess the inventory to proceed the order. As well as implementing the tracking system, we needed to design a user interface for interacting with the database where this information would be stored. The target time frame for which we intended to complete this project was six weeks.

2 Requirements

2.1 Functional

There were three functional components that we needed to implement for this website, which we broke into smaller sub-problems:

2.1.1 Interface to support customer orders

- Enable customers to choose their favorite tailors in accordance with available fabrics
- Match a customer order with the selected tailor's inventory
- Customers' personal data must be secure and protected
- Customers can communicate with AirBespoke via a form
- Notifications and updates will be automatically sent to customers' emails

2.1.2 Provide inventory tracker for tailors

- Constantly update the tailors' inventories
- Show tailors' current inventories with graphs and/or lists
- Tailors can update their data (available fabrics, textile)
- Tailors must be provided a tailor view to manage their inventory
- Tailors can communicate with AirBespoke via a form
- Notifications and updates will be automatically sent to tailors' emails

2.1.3 User-friendly management system for admin

- Non-CS admins must be supported with an easy-to-use management page that allows them to keep track of all the transactions and invoices
- Admins must be able to modify the website without any requirements of coding knowledge
- Admins must be able to communicate with tailors and customers in the finest manner
- Multiple payment methods must be set up (Stripe, Paypal, Skrill, etc.)

2.2 Non-Functional

- Have to work with a simulated AirBespoke's website as the current website has to be on air all the time
- Use a database to store tailor inventories and information
- Use Django framework for the back-end of the website

- Use web programming languages (HTML5, CSS, JavaScripts) for the front-end of the website
- Implement an easy-to-use interface for customers and tailors who are not familiar with the web in particular and technology in general
- Use Authentication API for login purposes
- Use external API's (Stripe, Shopify) for the payment process
- Use a hosting site to keep the website working (Heroku)

2.3 Potential Risks

There are two types of potential risks that are likely to occur: technology risk and skill risk.

2.3.1 Technical Risks

- We may run into problems integrating our new code with the existing code on the website
- We have to set up an alternative way to test our code as opposed to running it on the website, since the current website has to stay up at all times
- Creating a way for tailors to input their data into the website, but ensure the security of the site so that other tailors can't access other data

2.3.2 Skills Risks

- Some team members have worked with databases before, but not in production which is what this project will take
- No one is an expert in web development so we will all have to learn more about HTML5, CSS, and other web programming languages to accomplish our goals
- We will also have to work with API's that we might not completely be familiar with

3 System Architecture

The construction of AirBespoke website includes many working parts. Below, in Figure 1, is a high level diagram of the website architecture. Following the diagram is a description of each part and how it interacts with other parts in the system

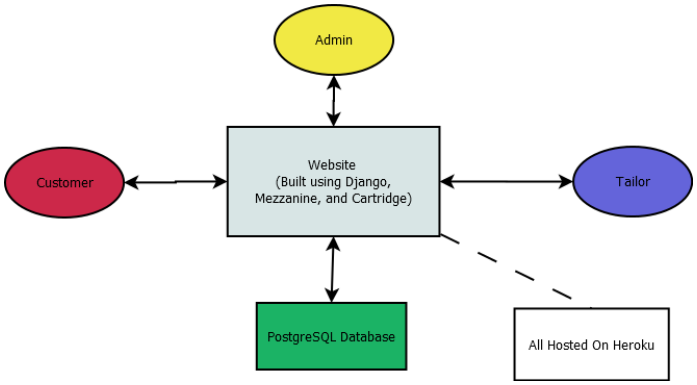

Figure 1: High Level View of System Architecture

3.1 Django/Mezzanine

We used the Django framework to quickly build our E-Commerce website. Django is an open source web framework written in Python. Django follows the model-view-controller design pattern and has an easy to create admin page. With the admin portal in Django it allowed our client to interact with the back-end without having to touch the codebase. Mezzanine, which is built on top of Django, enhances the admin site for our client to add pages and keep track of products with the press of a button. Mezzanine resembles other content management systems such as Wordpress or Wix, but allows backend support with the Django framework.

3.2 Cartridge/Stripe

Cartridge which extends the Mezzanine platform was used to develop the E-Commerce shop side of the website. Cartridge adds products, orders, and a cart which allowed us to build our store. We modified Cartridge to get custom products and orders for our client that tailored his exact needs. Cartridge included Stripe which allowed our client to add a payment method to their website.

3.3 Heroku

Heroku is a web application hosting service that allows for an easily deployable site. Heroku is easily scalable so if the client needs to expand the website it can be done quickly and requires no CS knowledge. Heroku also allowed us to

create backups of our data so when we had migration issues we were able to rollback without any problems.

3.4 PostgreSQL

For our project the database we used was PostgreSQL (Postgres). Postgres is an object relational database management system that stored all of our products, tailors, fabrics, users, and page information. Postgres also links with Heroku so we can have one database that is persistent with all of the data.

4 Technical Design

4.1 Database Hosting Design

As we were developing an E-Commerce website which required a database for orders, inventories, and information, database hosting was the most important design that we had to consider. Since our system could potentially have massive amounts of data in the future, we had to choose scalable technologies. Even though we currently have a few hundred queries to our database a day, this could easily rise to hundreds per minute when the scope of the company expanded to other developing countries besides Vietnam. Moreover, as we were working for a start-up company, our budget for the hosting site was limited. Therefore, we had to look for an inexpensive but fast-and-reliable database hosting service which could be scaled up if needed. After considering a couple of database hosting sites, we decided to use Heroku, a cloud Platform supporting several programming languages.

Heroku uses PostgreSQL to query the database, and it is accessible from any language with a PostgreSQL driver including many languages and frameworks supported by Heroku. As we were using Python as our primary language, Heroku was the most suitable for database hosting. It was linked with our local settings and was automatically updated whenever we made changes in the local site. Whenever a user requested to view a page by entering a URL in a browser, the Controller receives and decodes the request. Then it uses the Models we created to retrieve the data from the database, organizes them to send them to the View, where the data is used to render the final page to the customers. See Figure 2 on page 7.

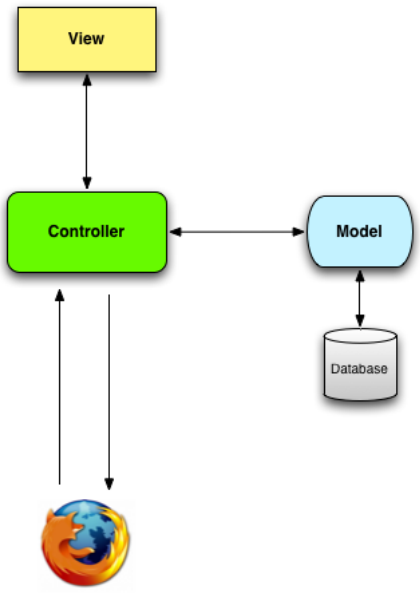

Figure 2: Database Architecture

4.2 Order Matching Design

One of our biggest goals for this project was to match a customer order with a tailor's inventory. When a customer is ordering their custom suit the first thing they have to do is choose a fabric that they want. Then they choose from a list of tailors that have that fabric, input their custom measurements, and choose how many suits they want of that kind of suit. Once the order is purchased we run a matching algorithm to assign the suit to the particular tailor that the customer wants.

The hardest thing about getting the orders to match to the tailors was getting the tailor information to be shown to the customer when purchasing the suit. So after the customer selects a fabric we run a filter query to only show tailors that have that fabric in stock. Then the customer chooses the tailor and we store that choice with the suit. The problem was that the suit can't be sent to the tailor immediately once the customer selects the tailor that they want because they may choose not to purchase the suit.

Once the customer purchases the order we ran a method to take all of the suits in the order and send them to the tailor. This was accomplished by taking each suit and getting the tailor that the customer selected and matching the id of the selected tailor to the id of the tailor that uses the site. Then we added that suit to the tailor's list of orders. See Figure 2 on page 8.

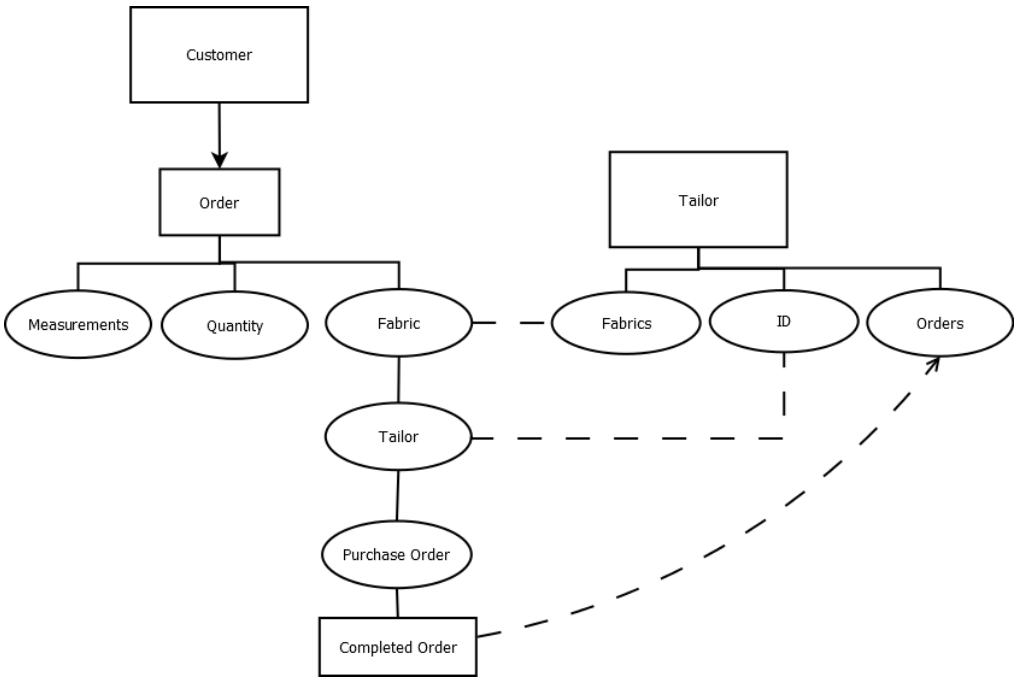

Figure 3: Order Matching Flow Diagram

5 Decisions and Implementations

There were three main aspects that we had to consider: Coding language, working framework, and the hosting site.

5.1 Language Decision

Initially, our client had a fully-polished website built with WIX, a drag-and-drop site builder for people who don't have coding knowledge. We did some research into WIX and ultimately realized that it would not be possible to accomplish the task of creating an inventory tracking system for the current website because WIX had no backend customization. Therefore, we researched into languages that we could use to build our simulated website to support the project. We looked into other E-Commerce sites and what they used to figure out which language suited the best. Finally, we narrowed it down to three languages: Ruby, PHP, and Python. As our team had no experience with Ruby, we would need to learn Rails if we were to build a website from scratch. For PHP, only one member of our group had experience in working with the language. While a majority of the group had at least worked with Python before. Therefore, we decided to proceed with Python as most of us had experienced the language. Moreover, Python is a very popular language which is supported by many good frameworks, and it has multiple applications in the programming industry.

5.2 Framework Decision

When choosing Python as our working language, one of the main factors was the ability to use available frameworks. One Web framework, Django, enabled us to create a full-stack web page using less time, thanks to their built-in functionality. We chose Django because we realized that it would be quicker for us to use this framework in order to build a complex website rather than to write all of the site from scratch. Also, the Django framework led us to Mezzanine, a content management platform built on top of Django. Eventually, Mezzanine led us to Cartridge, a third party shopping cart module. These platforms allow the client to easily create products, orders, and users for their site, which is a crucial component of this project. Finally, Django, Mezzanine and Cartridge had a significant amount of documentation that supported us with learning about what we needed to proceed with the project.

5.3 Hosting Decision

One of the non-functional requirements for our project was to enable our website to run in production. With our Django project, we can run it on our local machine, but this will not work in production since there could be hundreds of users on the site at one time. Therefore, we had to look for a reliable hosting server. Moreover, as the budget was limited and our client was not CS-majored, we tried to look into hosting servers that were inexpensive, had a short learning curve, and would be easily deployable for non-CS personnel. We decided to go with Heroku because it was free to start with and could be easily scaled upwards if needed for the client. Also, it was easy to adapt to, and the documentation had many step-by-step tutorials, from how to get your project up to how to

manage your database. Heroku also links with a PostgreSQL database which can be scaled accordingly for the client when the company grows bigger. The final thing that was great about Heroku was that the dashboard had a very user-friendly UI, from which our client could run/configure the server conveniently without having to use the command prompt.

6 Results

6.1 Overview

Our project has been thoroughly tested and is currently working beyond what was initially required. Besides the tracking functionality of the website, we extended to implement some feasible features for our client (Page management, registration, and some custom designs). For the majority of the project, our testing was done manually. Since we were building a website, a lot of our code was checked on whether the feature showed up and worked accurately or not. If the page didn't show up or look correct, we could easily change the code and test it again. For our more extensive code such as the linking algorithm, we ran tests to ensure that each order got matched up to an available tailor, which could easily be checked in the admin page.

6.2 Implemented Features

- Customers are able to choose their favorite tailors in accordance with available fabrics
- Match a customer order with the selected tailor's inventory
- Order notification will be automatically sent to the customer's email when the order is placed
- Tailors' inventories will always be updated
- Tailors' inventories will be shown in the list format
- Tailors are able to update their data (available fabrics, textile)
- Tailor accounts are set up and tailor profile page is provided
- Admins are able to manage the website via a user-friendly setting page (set tax, change the display, track orders, manage users)
- Multiple payment methods have already been set up (Stripe, Paypal, Skrill, etc . . .)

6.3 Future Work

- Having a web designer come in and decorate the frontend of the website using CSS, HTML, and JavaScript to make it look more professional
- Creating a fully-customizable shop interface as the client wants something similar to WIX-based system, which is user-friendly as it provides many drag-and-drop tools
- Implementing more features into the tailor input page, as currently it is kept as simple as possible for the tracking functionality
- Setting up the shipping system which allows customers to track their orders
- Implementing a more detailed customer profile page

Appendices

A Product Installation Instructions

A.1 Django

https://docs.djangoproject.com/en/1.9/intro/install/
https://docs.djangoproject.com/en/1.9/intro/overview/

A.2 Mezzanine

http://mezzanine.jupo.org/docs/overview.html#installation

A.3 Cartridge

http://cartridge.jupo.org/overview.html#installation

B Development Environment Description

Getting our production app onto your computer

- Go to your project's empty src folder / delete everything in the src folder
  git init
- Then run the command git remote add origin https://github.com/kthistle/AirBespoke1.0.git
- Then run git pull origin master (You should have the entire project now)
- Then run pip install -r requirements.txt (Make sure your virtualenv is on, this step gives you all the dependencies from the project)
- Whenever you upload to master it will automatically be uploaded to Heroku also so make sure any changes you make are stable
- I suggest that you should pull the master, branch off from there, add content, pull again, merge branches, the push to the git hub to ensure that we have as little conflicts as possible and that the production app doesn't have any errors.
- You can view the app at: https://airbespoke-1.herokuapp.com/

C Modeling Technique

C.1 Django

https://docs.djangoproject.com/en/1.9/intro/tutorial02/
https://docs.djangoproject.com/en/1.9/intro/tutorial03/

C.2 Mezzanine

http://mezzanine.jupo.org/docs/model-customization.html

C.3 Cartridge

http://cartridge.jupo.org/overview.html#features