COSC 4P02 – Final Report – Group 7

April 27, 2025

Instructor: Naser Ezzati-Jivan

TA: Madeline Janecek

Project Topic: PRJ3: Shop builder for social media sellers

GitHub: https://github.com/benCombe/Shopimy

Jira board: https://abishop.atlassian.net/jira/software/projects/SS/summary

Figma board: https://www.figma.com/design/fU1vUeeUaLm6gjVrEEEJGm/Shopimy?node-id=0-1&t=M80fOizwfvKM8yBG-1

If access is needed to view the Jira board, please contact Ashley

Team Members

• Ben Combe	5819446	bc14ik@brocku.ca
• Ashley Bishop	6693824	ab18yg@brocku.ca
• Adam Shariff	6768600	as19tq@brocku.ca
• Ben DeHooge	6567069	bd18rc@brocku.ca
• Spencer Ing	6756605	si19wd@brocku.ca
• Braden Lucas	6880462	bl19mj@brocku.ca
• Steven Putter	6966048	sp19cj@brocku.ca

Table of Contents

Outline of Documents

The following report outlines the development process of Shopimy, a light-weight shop builder for social media sellers. Below the following documents are outlined with links provided for easy access to them.

User Requirements Documentation

https://github.com/benCombe/Shopimy/blob/staging/Documents/Additional_documentation/User_requirements.pdf

This document introduces Shopimy and the main objectives of the project. It then discusses the functional dependencies/ key features of the project followed up with an outline of the user stories that are key to the project. This can be found in our GitHub repository at documents/Additonal_Documention/

https://github.com/benCombe/Shopimy/blob/staging/Documents/REQUIREMENTS.md

This document outlines the software requirements for the project. The documents include a System Overview, User Roles, Functional Requirements, Non-Functional Requirements, technology Stack, constraints and Assumptions and Appendices sections to form a complete view of the project's requirements. This can be found in our GitHub repository at Documents/ under the name REQUIREMENTS.md

Installation Guide

https://github.com/benCombe/Shopimy/blob/staging/Documents/INSTALLATION.md

This installation guide shows a user how to set up and run Shopimy for the developer. It outlines the perquisites needed, necessary steps, how to run the application, and development branching. This can be found in our GitHub repository at Documents/ under the file name INSTALLATION.md

User Manual

https://github.com/benCombe/Shopimy/blob/staging/Documents/Additional_documentation/User_manual.pdf

This user manual aims to provide an in-depth detailed description of how to navigate through the Shopimy application. It provides information on specific features, and how to work within a specific section of Shopimy. It can be found in documents/additional_documentation/

Introduction

E-commerce is a constantly growing and evolving sphere, one in which has boomed since the covid-19 pandemic. The rise of online shopping has completely changed how businesses interact with customers, it has become the norm for businesses to have an online component to their store, not just a physical location. Shopimy aims to help new and small businesses launch their online presence, it offers a lightweight and user-friendly experience to encourage connection between customers and store owners. Shopimy allows users to build their own stores and offers features such as personalization of the store, a dashboard that includes analytics, inventory management, and product logs. Shopimy allows users to interact with their stores as both a customer with or without an account and features unique URLs. This report outlines the development process of Shopimy including the design, front-end and back-end, it also includes testing documentation, GitHub and Jira logs of the entirety of the development process, a detailed description of the Shopimy teams' software engineering progress, challenges, and success we faced and a discussion of our use of AI.

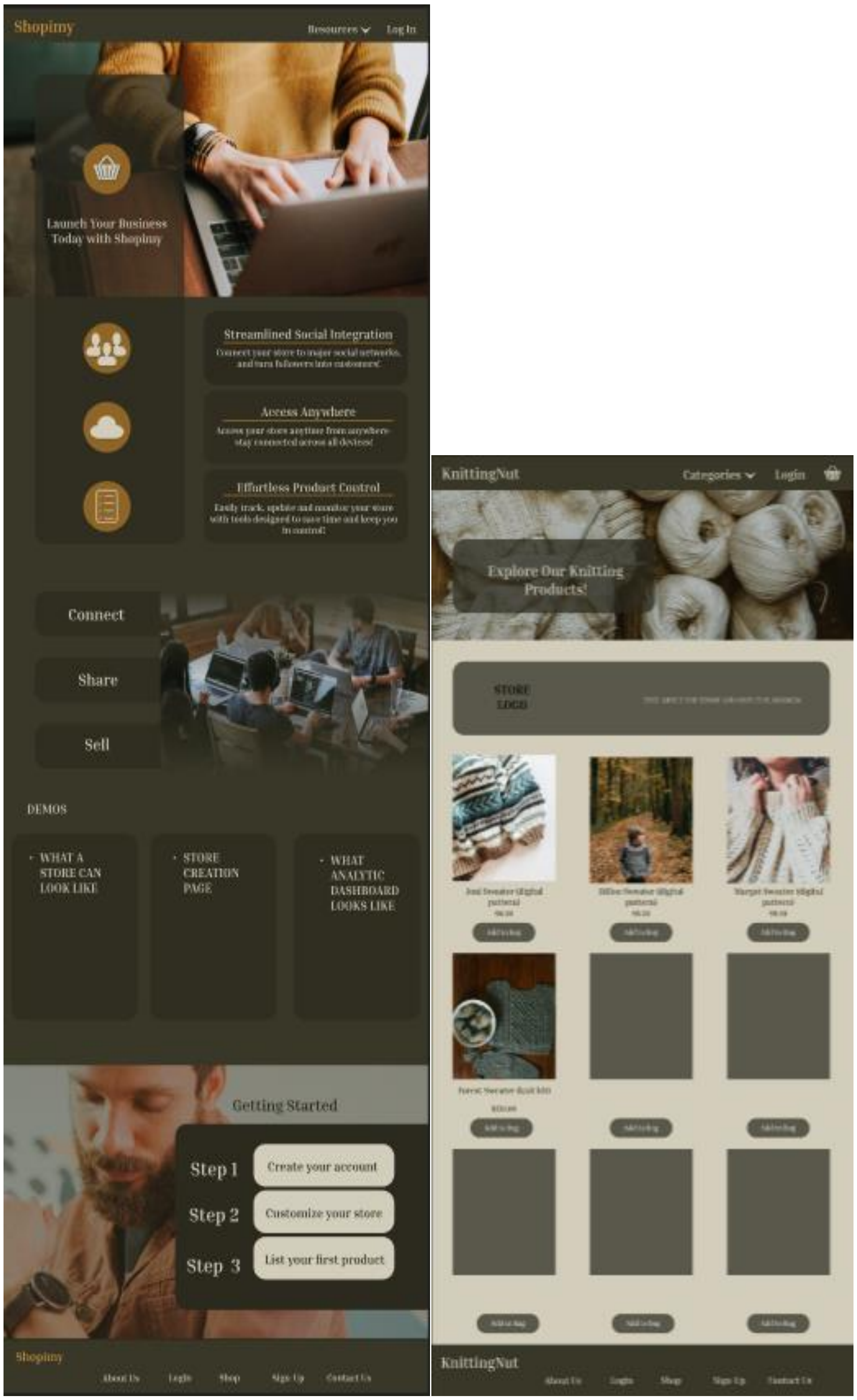
Development Process

Design Process

Over the course of the development process the design of the Shopimy application was very important to our team. Thought and time was put into our design process, consideration into the types of features, the layout and the feel of the application going forth to ensure consistency throughout the application. Shopimy was designed to ensure that new business owners would have a simple and intuitive application to fulfill their needs as an emerging e-commerce company.

Each sprint encompassed a design component, during our first initial sprints, considerable time was put into designing a large part of Shopimy. Considerations of colour, how specific aspects were laid out and specific features were all discussed within the team, with team members all approving initial designs before we moved on to implementing the designs. During our later sprints, as requirements and functionalities changed, our team revisited previous designs and edited and changed them to fit our specifications. Below is a compilation of all our designs created in Figma.

It should be noted that all designs created for the Shopimy application are included in the screenshots below. However, our team did not implement all the features that were designed, and many designs were changed during the implementation process to accommodate the changing needs of the application.

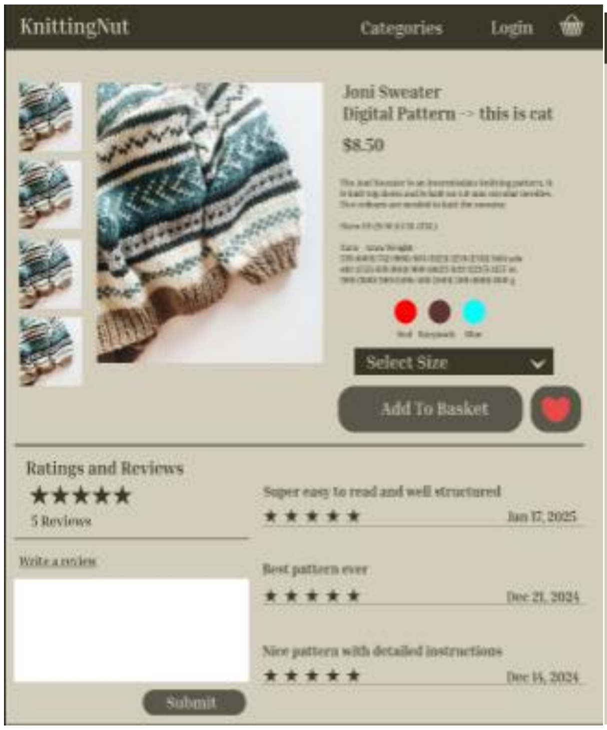


Connect your store to major social networks, and turn followers into customers!

Access your store anytime from anywhere- stay connected across all devices!

Easily track, update and monitor your store with tools designed to save time and keep you in control!

Figure 1 (left) homepage for Shopimy design, (right) General store layout design



Joni Sweater
Digital Pattern -> this is cat
$8.50

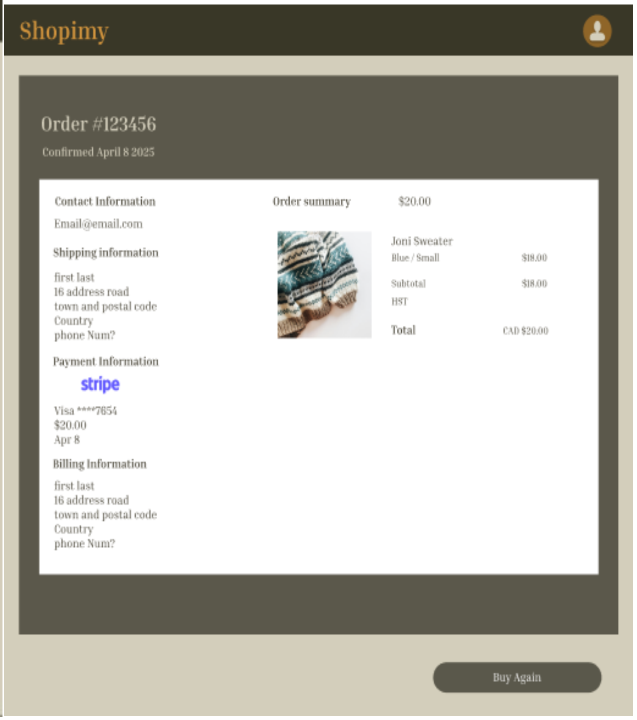
[illegible]

Ratings and Reviews
★★★★★
5 Reviews

Super easy to read and well structured
★ ★ ★ ★ ★ Jan 17, 2025

Best pattern ever
★ ★ ★ ★ ★ Dec 21, 2024

Nice pattern with detailed instructions
★ ★ ★ ★ ★ Dec 15, 2024



Order #123456
Confirmed April 8 2025

Contact Information
Email@email.com

Shipping information
first last
16 address road
town and postal code
Country
phone Num?

Payment Information
stripe
Visa ****7654
$20.00
Apr 8

Billing Information
first last
16 address road
town and postal code
Country
phone Num?

Order summary $20.00

Joni Sweater	
Blue / Small	$18.00
Subtotal	$18.00
HST	
Total	CAD $20.00

Figure 2 (left) item listing design, (right) receipt design

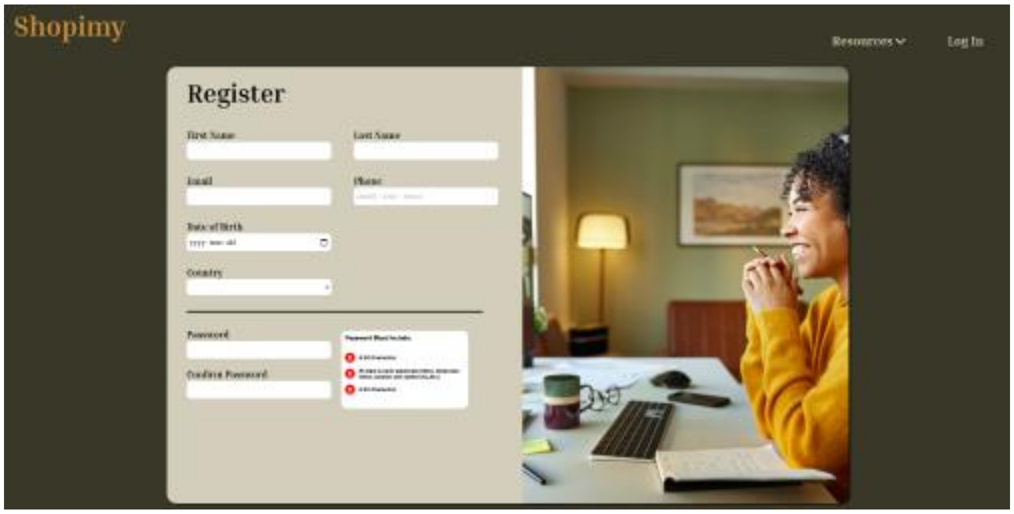


Figure 3 Register page design

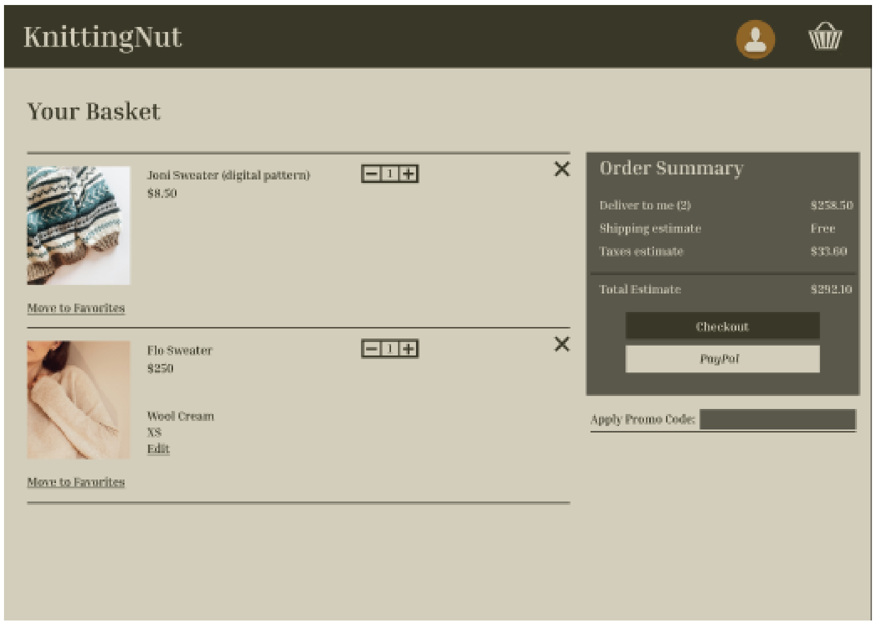


Figure 4 Shopping cart design

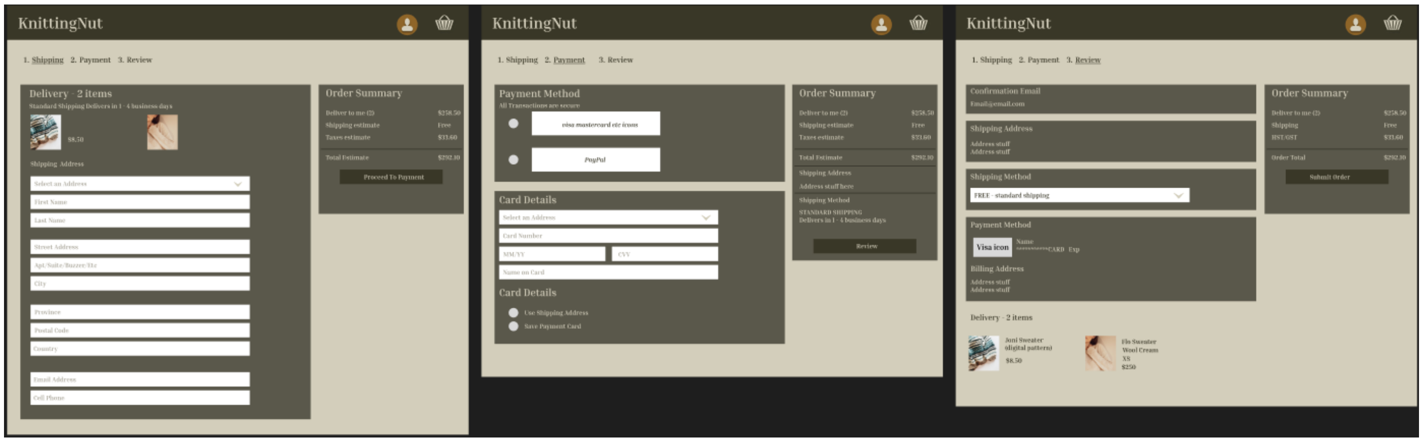


Figure 5 Checkout pages design (includes shipping, payment, and review information)

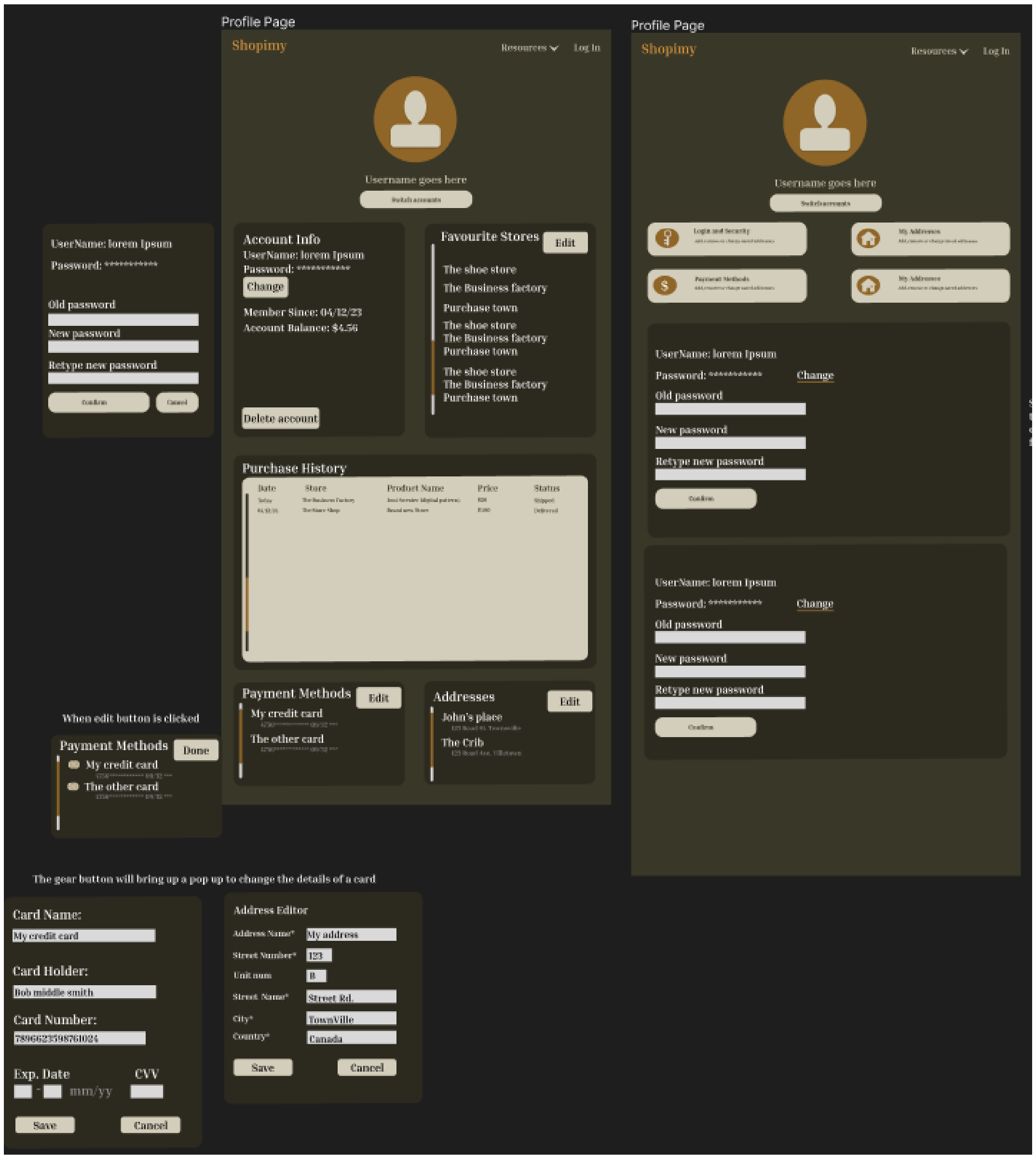
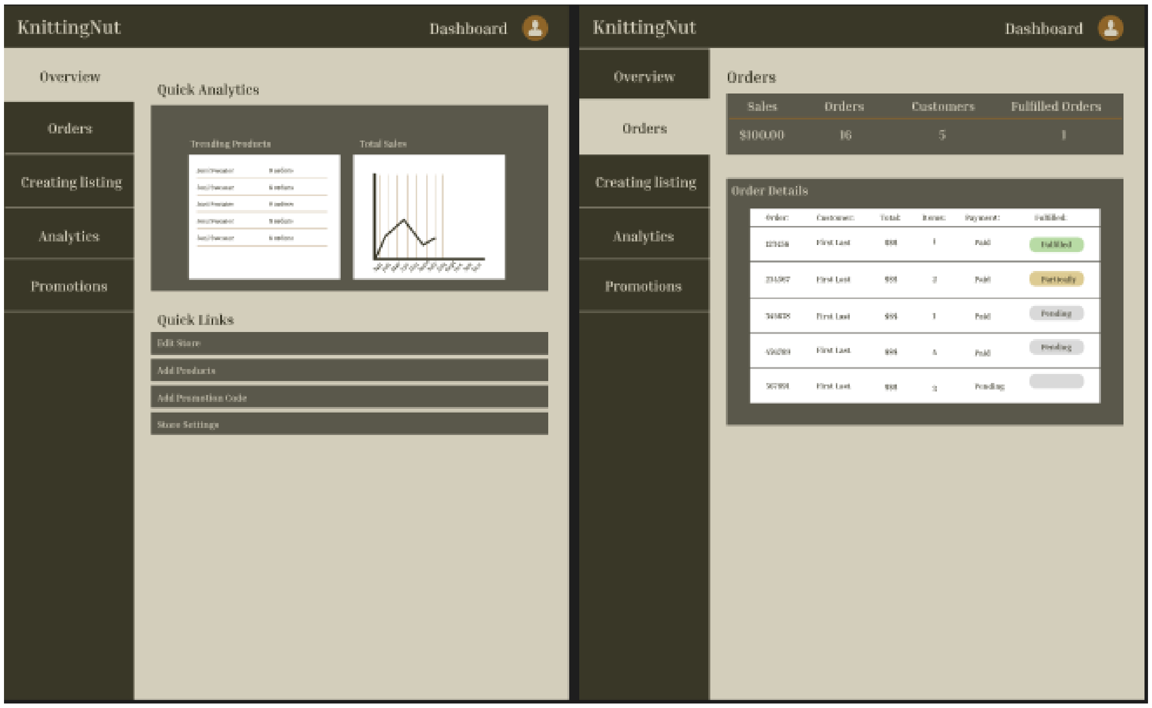


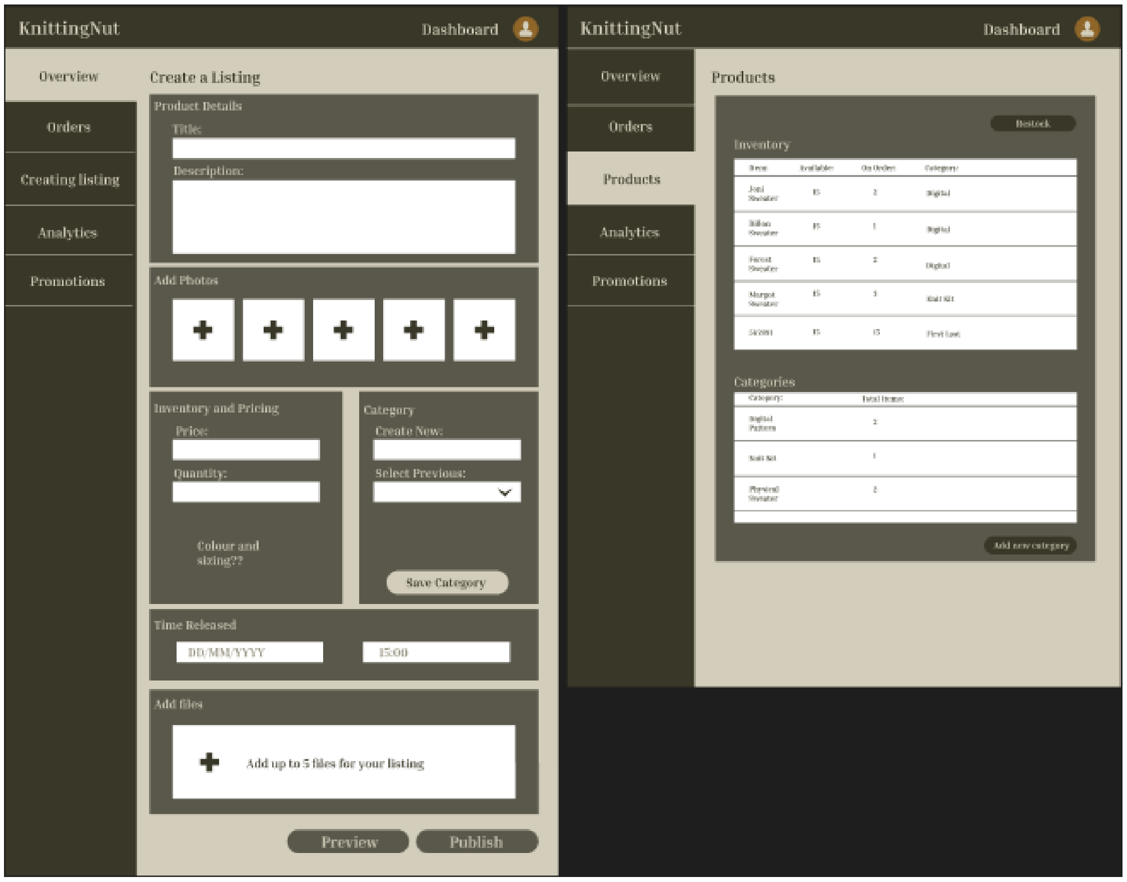
Figure 6 User Profile Design



Sales	Orders	Customers	Fulfilled Orders
$100.00	16	5	1

Order:	Customer:	Total:	Items:	Payment:	Fulfilled:
123456	First Last	$88	1	Paid	Fulfilled
234567	First Last	$55	2	Paid	Partially
345678	First Last	$55	1	Paid	Pending
456789	First Last	$88	4	Paid	Pending
567891	First Last	$88	3	Pending	

Figure 7 Store dashboard, (left) overview page, (right) orders page designs



Item:	Available:	On Order:	Category:
Joni Sweater	15	2	Digital
Billon Sweater	15	1	Digital
Forest Sweater	15	2	Digital
Margot Sweater	15	3	Knit Kit
567891	15	15	First Last

Category:	Total Items:
Digital Pattern	2
Knit Kit	1
Physical Sweater	2

Figure 8 Store Dashboard, (left) create a listing, (right) products listing with category section designs

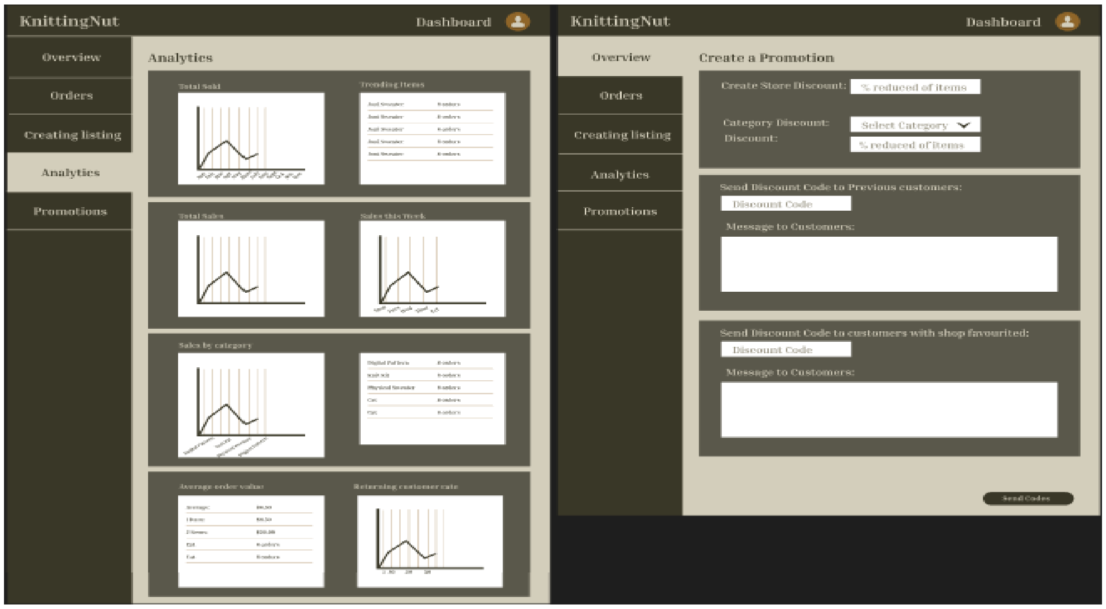


Figure 9 Store Dashboard, (left) analytics, (right) Promotions pages designs

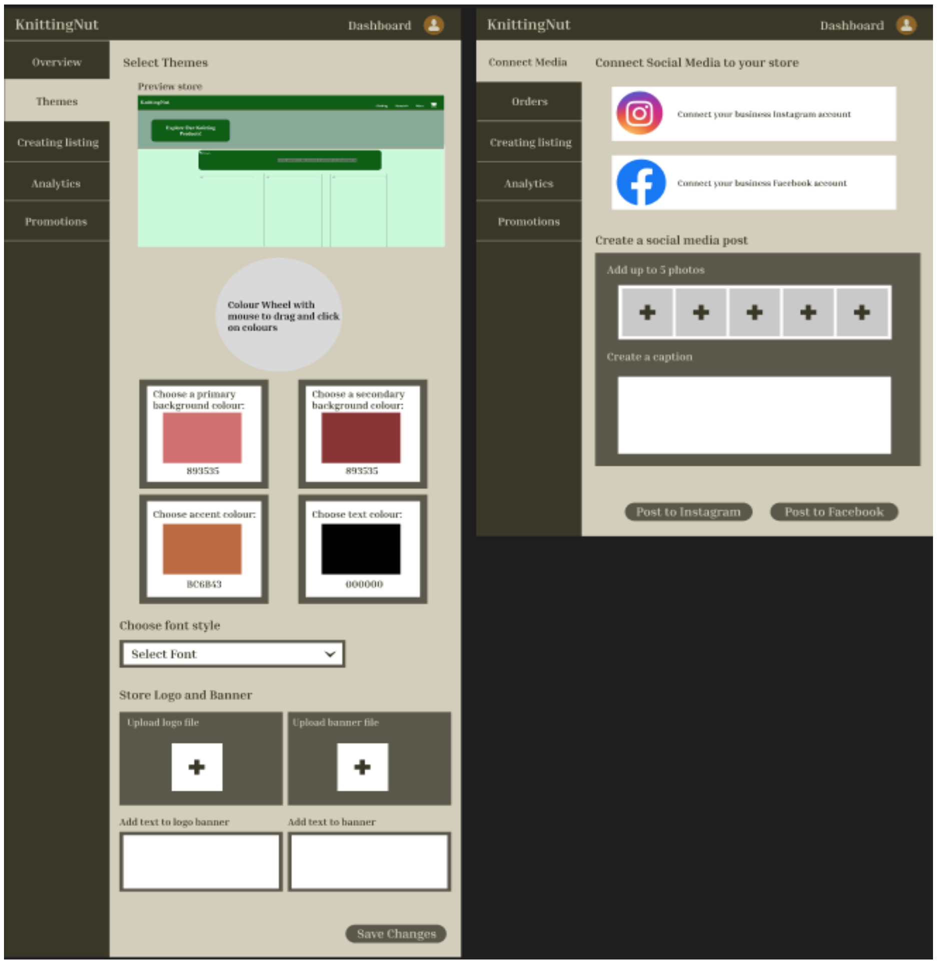
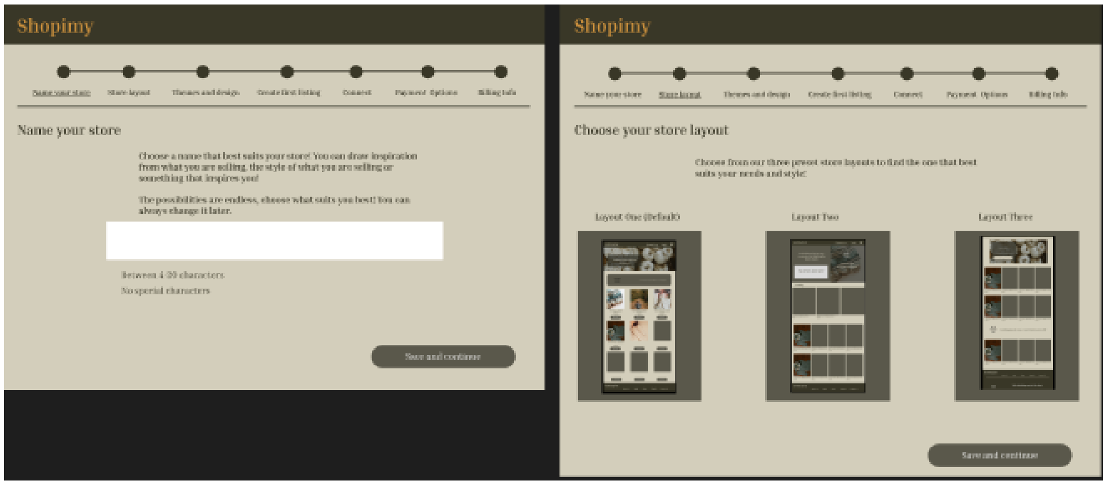


Figure 10 Store Dashboard, (left) Themes and Designs, (right) social media integration designs



Name your store

Choose a name that best suits your store! You can draw inspiration from what you are selling, the style of what you are selling or something that inspires you!

The possibilities are endless, choose what suits you best! You can always change it later.

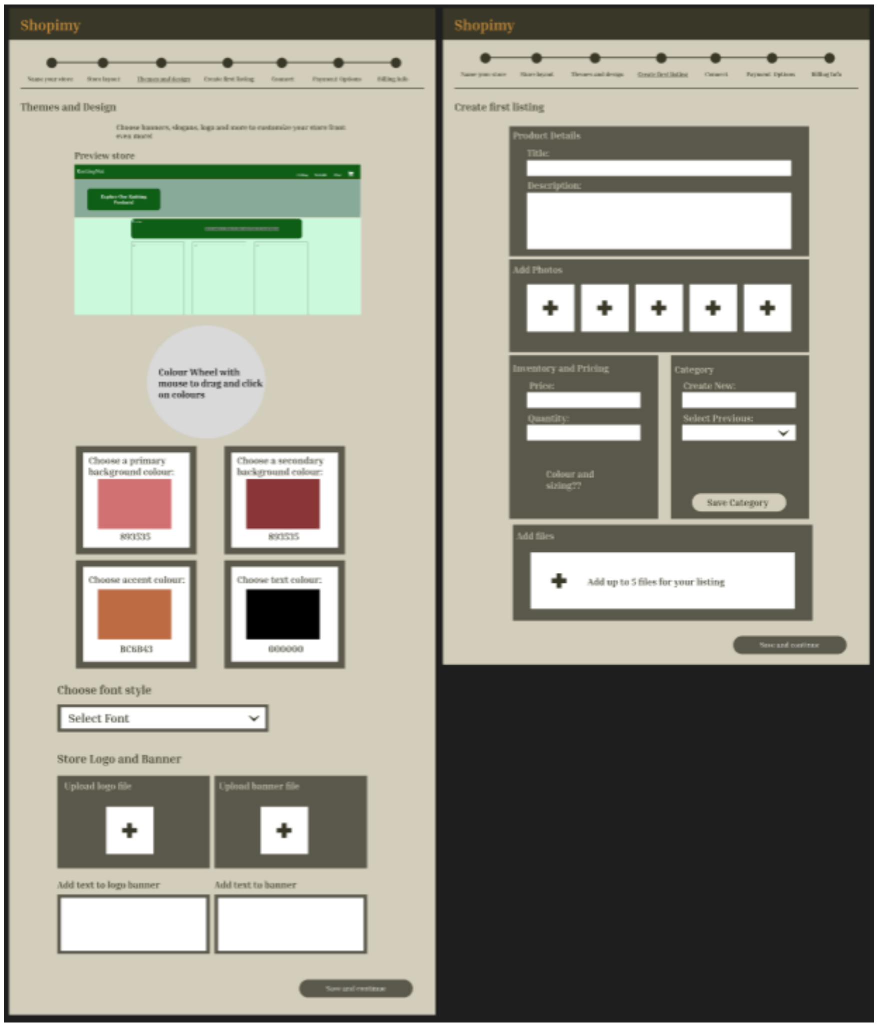
Between 4-20 characters

No special characters

Choose your store layout

Choose from our three preset store layouts to find the one that best suits your needs and style!

Figure 11 Store Dashboard, (left) store name, (right) store layout page designs



Themes and Design

Choose banners, slogans, logo and more to customize your store front even more!

Choose a primary background colour: 893535

Choose a secondary background colour: 893535

Choose accent colour: BC6B43

Choose text colour: 000000

Choose font style

Store Logo and Banner

Add text to logo banner

Add text to banner

Create first listing

Product Details

Title:

Description:

Add Photos

Inventory and Pricing

Price:

Quantity:

Colour and sizing??

Category

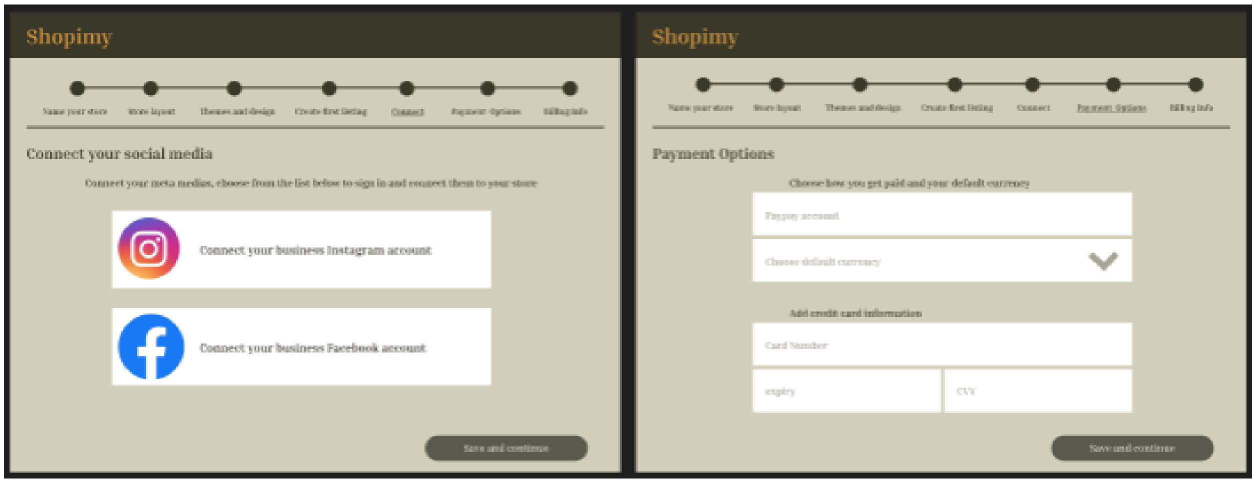
Create New:

Select Previous:

Add files

Add up to 5 files for your listing

Figure 12 Build your store, (left) themes and design pages design



Connect your social media

Connect your meta medias, choose from the list below to sign in and connect them to your store

Connect your business Instagram account

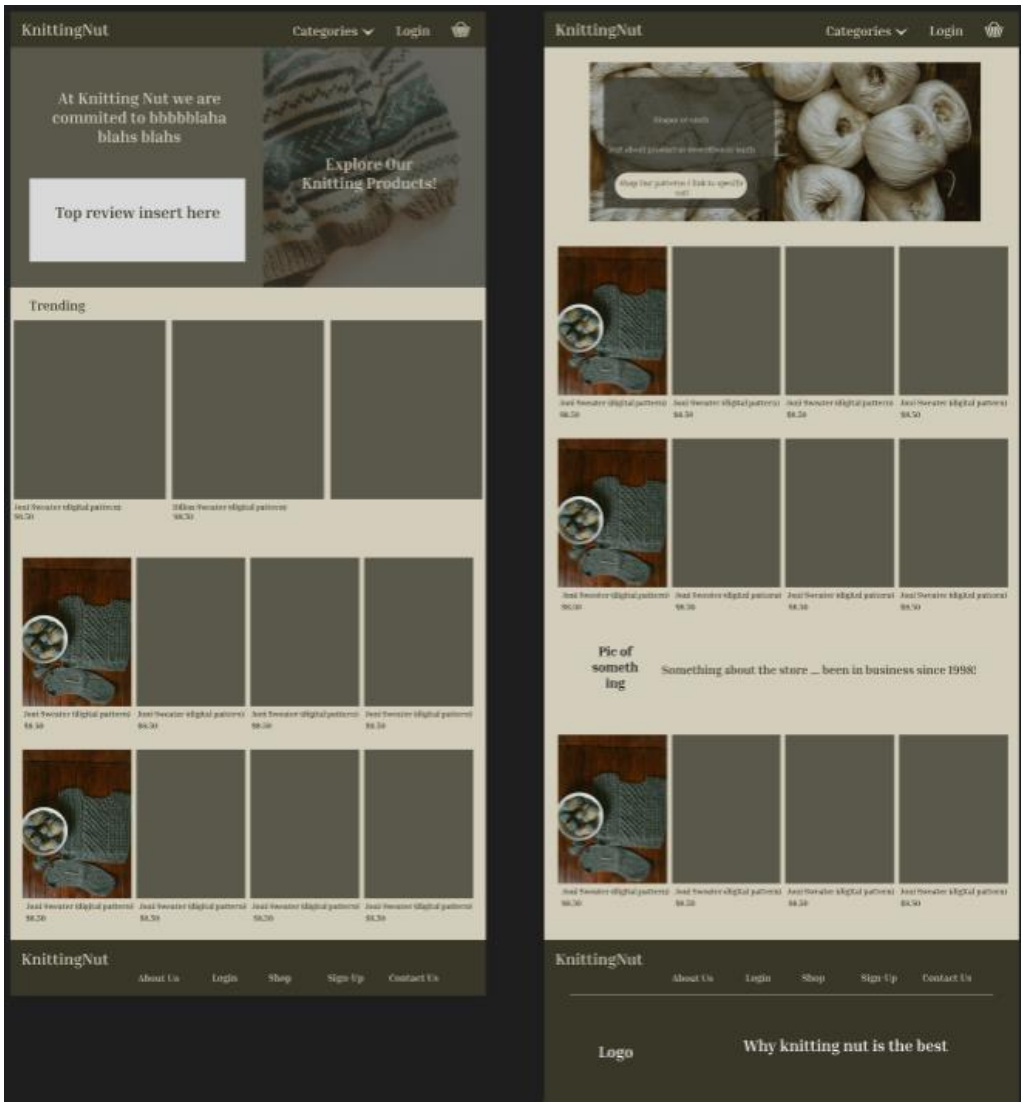
Connect your business Facebook account

Payment Options

Choose how you get paid and your default currency

Add credit card information

Figure 13 build your store, (left) connect your social media, (right) payment options designs



At Knitting Nut we are commited to bbbbblaha blahs blahs

Top review insert here

Something about the store ... been in business since 1998!

Why knitting nut is the best

Figure 14 Store layout designs

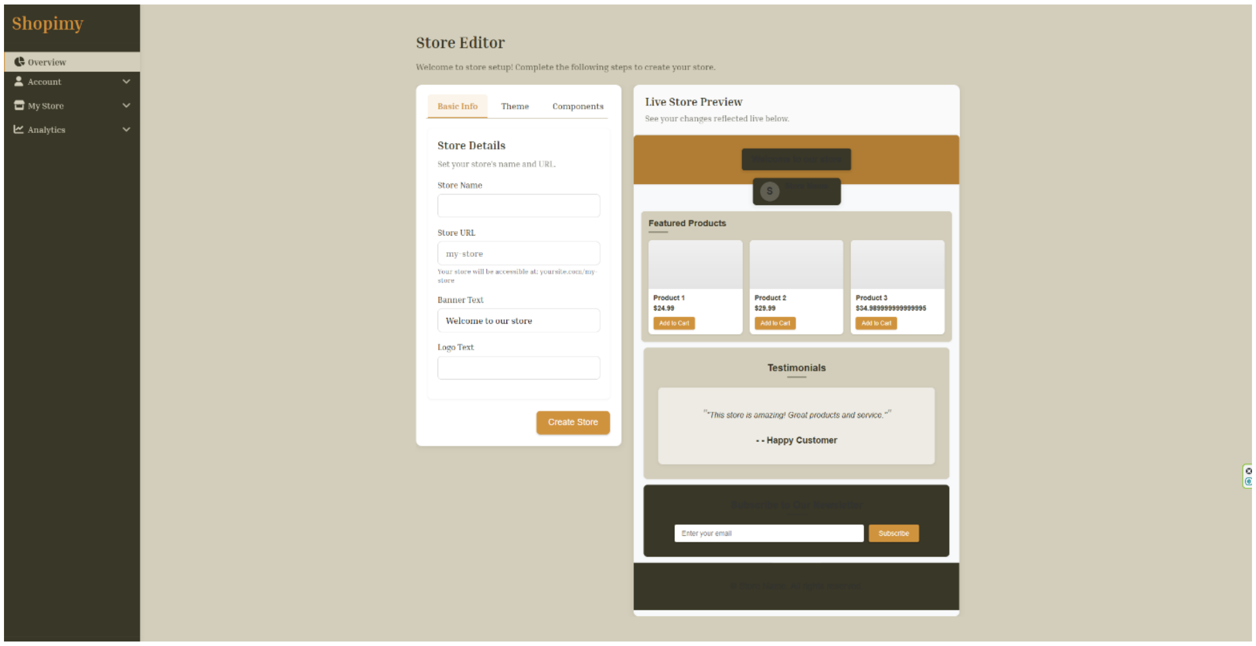
Front-End Process

Our Front-End team worked relentlessly to ensure that functionality and logic met with a functional design that served as a simple and easy to use application. In the beginning stages much of our team’s time was focused on designing an application that served our purposes and building a foundational base of the site to begin development.

The beginning sprints of development focused on the implementation of the landing page or the Shopimy homepage, the register and login pages and implementing the wireframe of what the store dashboard looked like. Additionally, work towards the shopping cart and the following checkout pages were also implemented. Our third and fourth sprint focused on implementing the store dashboard, a key component of the store. Our team also ensured that the product page, which shows product details like pictures, sizes, colours etc, had an implemented front-end and the store was able to load product data from the backend.

Towards the end of the development, considerable amounts of time were put in by members to ensure that the logic of the store, and the functionality of the store worked as anticipated. Additionally, time towards mobile/ responsive design, and ensuring that the store dashboard looked and felt like a usable feature. Much of what had been done regarding the dashboard was overhauled and rewritten to ensure consistency and functionality. Our later sprints, five and six, ensured that the store editor was a primary feature, and it allowed store owners to edit their store. Considerable amount of time by our team was put into ensuring that the store editor, had components such as basic information regarding, product title, banner text customization and logo text, with the user also being able to select themes (colours) for their store. Our team has ensured that there is a variety of preexisting themes for the user to choose from, with a user having the option to create their own colour theme and upload logo selection. This can be seen in *figure 15*. Additionally, our team allowed the store owner to toggle on or off features to ensure that customizability is a priority. All changes that the store owner makes can be seen in the live store preview section of the Store editor.

In our final sprint, the focus was ensuring that Shopimy behaved in the expected manner. To ensure this our team made countless adjustments regarding mobile fixes, code cleanup and testing manually the application.


Shopimy

Overview
Account
My Store
Analytics

Store Editor

Welcome to store setup! Complete the following steps to create your store.

Basic Info | Theme | Components

Store Details

Set your store's name and URL.

Store Name

Store URL

my-store

Your store will be accessible at: yoursite.com/my-store

Banner Text

Welcome to our store

Logo Text

Create Store

Live Store Preview

See your changes reflected live below.

Featured Products

Product 1
$24.99
Add to Cart

Product 2
$29.99
Add to Cart

Product 3
$34.989999999999995
Add to Cart

Testimonials

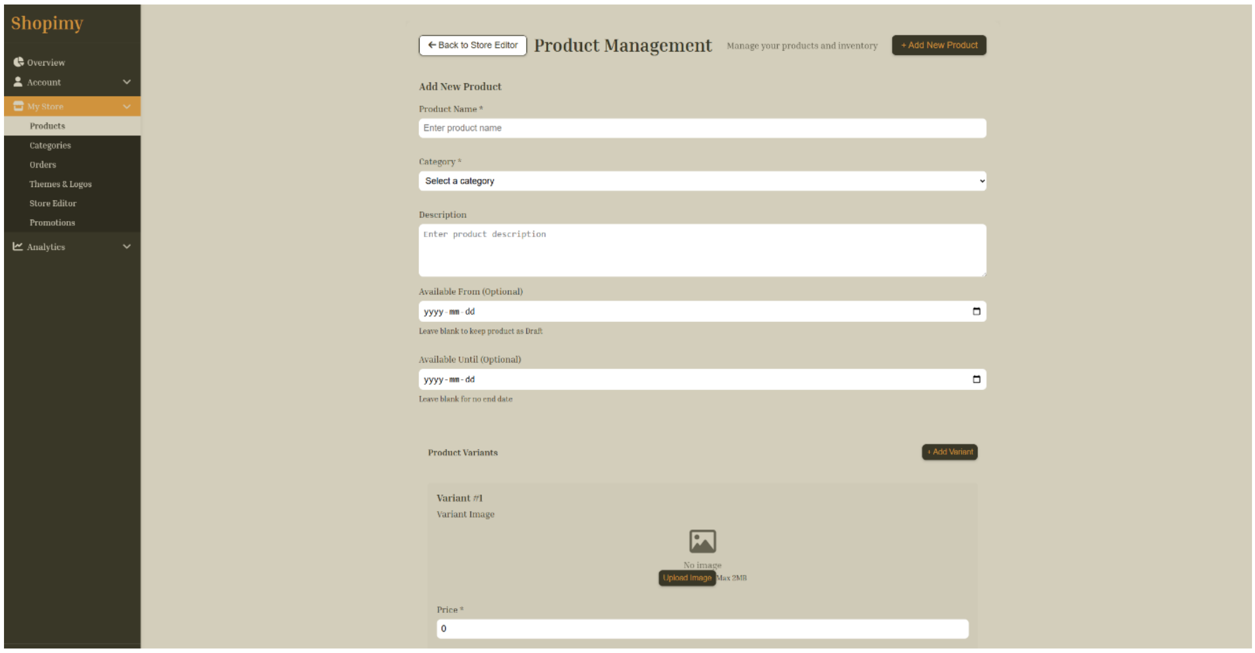
""This store is amazing! Great products and service.""

- - Happy Customer

Enter your email
Subscribe

Figure 15 Implementation of store editor

Product management (*figure 16*) and category management (*figure 17*) were also developed in our later sprints, these feature the ability to add products and their details and the ability to add, edit or delete categories, respectively.


Shopimy

Overview
Account
My Store
Products
Categories
Orders
Themes & Logos
Store Editor
Promotions
Analytics

← Back to Store Editor

Product Management

Manage your products and inventory

+ Add New Product

Add New Product

Product Name *

Enter product name

Category *

Select a category

Description

Enter product description

Available From (Optional)

yyyy-mm-dd

Leave blank to keep product as Draft

Available Until (Optional)

yyyy-mm-dd

Leave blank for no end date

Product Variants

+ Add Variant

Variant #1

Variant Image

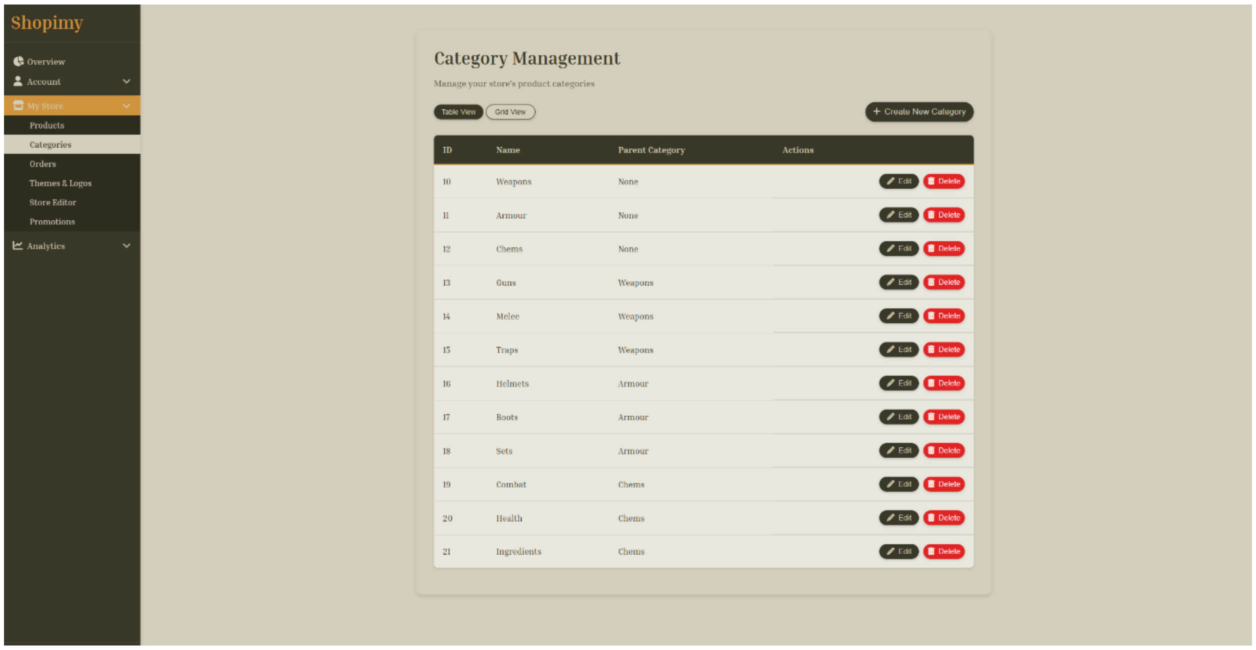
No image

Upload Image Max 2MB

Price *

0

Figure 16 Implementation of product management



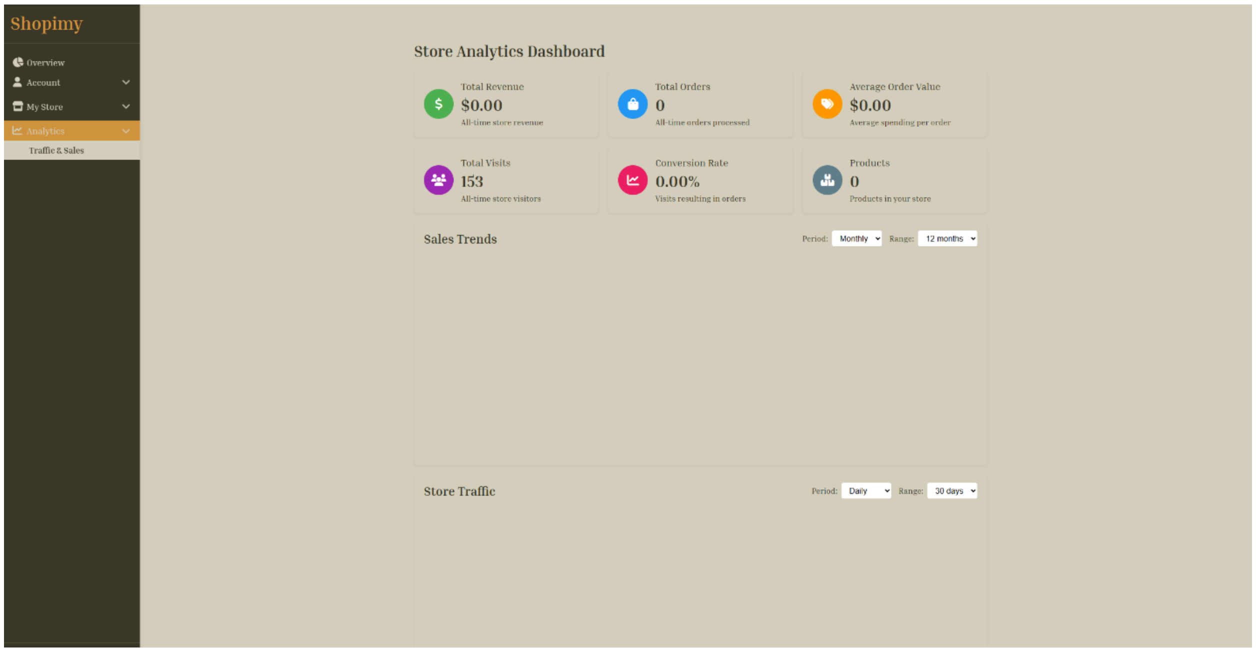
Category Management

Manage your store's product categories

ID	Name	Parent Category	Actions
10	Weapons	None	Edit Delete
11	Armour	None	Edit Delete
12	Chems	None	Edit Delete
13	Guns	Weapons	Edit Delete
14	Melee	Weapons	Edit Delete
15	Traps	Weapons	Edit Delete
16	Helmets	Armour	Edit Delete
17	Boots	Armour	Edit Delete
18	Sets	Armour	Edit Delete
19	Combat	Chems	Edit Delete
20	Health	Chems	Edit Delete
21	Ingredients	Chems	Edit Delete

Figure 17 Implementation of category management

Additionally, implementation of the analytics and promotions page was also done in the final sprints. These can be seen in *figures 18* and *19*, respectively. It should be noted that while the analytics page does not feature any figures, once the store is in use and has purchases being made would these be generated, it should be noted that we used chart.js to generate graphs once purchases have been made. The promotions page allows a store owner to create a promotion for their store.



Store Analytics Dashboard

Total Revenue
$0.00
All-time store revenue

Total Orders
0
All-time orders processed

Average Order Value
$0.00
Average spending per order

Total Visits
153
All-time store visitors

Conversion Rate
0.00%
Visits resulting in orders

Products
0
Products in your store

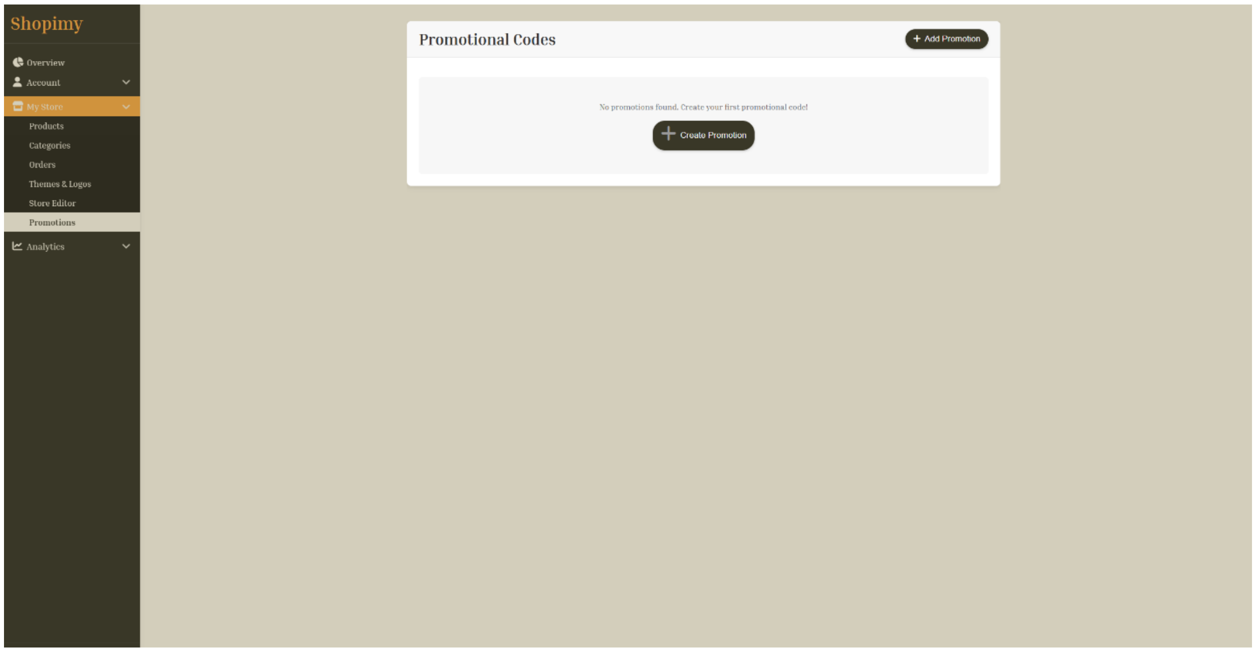
Sales Trends

Period: Monthly Range: 12 months

Store Traffic

Period: Daily Range: 30 days

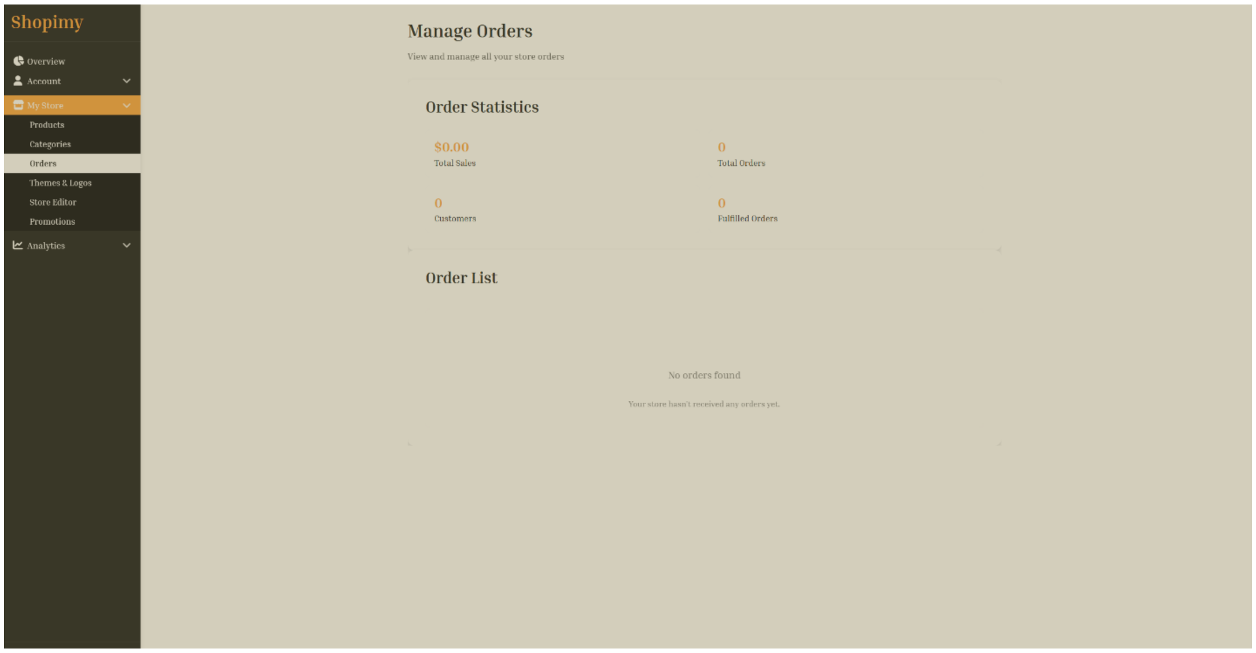
Figure 18 Store dashboard, analytics



No promotions found. Create your first promotional code!

Figure 19 Store dashboard, implemented promotions page

The orders page was also implemented in our final sprints, with the user being able to view quick statistics and view the order list that is currently in demand. It should be noted as there are no current purchases, there is not data to pull from, however as purchases are made data will appear in the order list.



No orders found

Your store hasn't received any orders yet.

Figure 20 Store Dashboard, implemented orders page

The above outlines all recent implementations that have been made to Shopimy. Much of which is seen was redesigned (original designs allowed for a base line of the features of each page) and some functionalities that were initially proposed were scrapped due to time constraints and the distribution of work.

Back-End Process

Throughout our team's development process, our back-end team has been working to ensure that our database is working as efficiently as possible.

Within our first couple sprints there was much time that was put into designing our database and implementing our initial design. This was done to ensure we had a strong foundational base of how the database will work going forward. One of the first functionalities that our team implemented in the beginning of the project was the user login and sign in functions with a salted hash. Additionally, work towards setting up the database and hosting it. Ensuring that routing and navigation between pages was also done early. Throughout all our sprints, considerable time and consideration was put into creating triggers and tables that will ensure that our application can work as effectively as possible, in addition to controllers for various functions.

In our later sprints, our database was always evolving and changing with the needs of our growing application. Below in *Figure 21* is the updated and final ER diagram our team produced for our database. In our middle sprints and final sprints, our team put time into functionalities regarding our shopping cart, calculating prices, and ensuring that our item can display variations it. Photos and photo storage were implemented with blobs. Our last and final couple sprints were spent implementing our item detail and implementing graphs and analytics for our store dashboard. By adding in various controllers that were necessary to the system. Additionally, we spent time implementing Stripe, a payment system. Stripe is set up to work however because we are not hosting Shopimy publicly, it is not accessible to use, but time was spent ensuring that we have the proper procedures in place. Furthermore, in our last sprint, considerable time was put into establishing the store dashboard in the backend. This includes making numerous controllers, APIs, and tables for all pages within the dashboard.

Our team also developed numerous API's for Shopimy, in *figure 22,* and the following two images show the amount that we implemented. *Figure 22* and the two were screenshots from Swagger, a tool for developing and describing RESTful APIs.

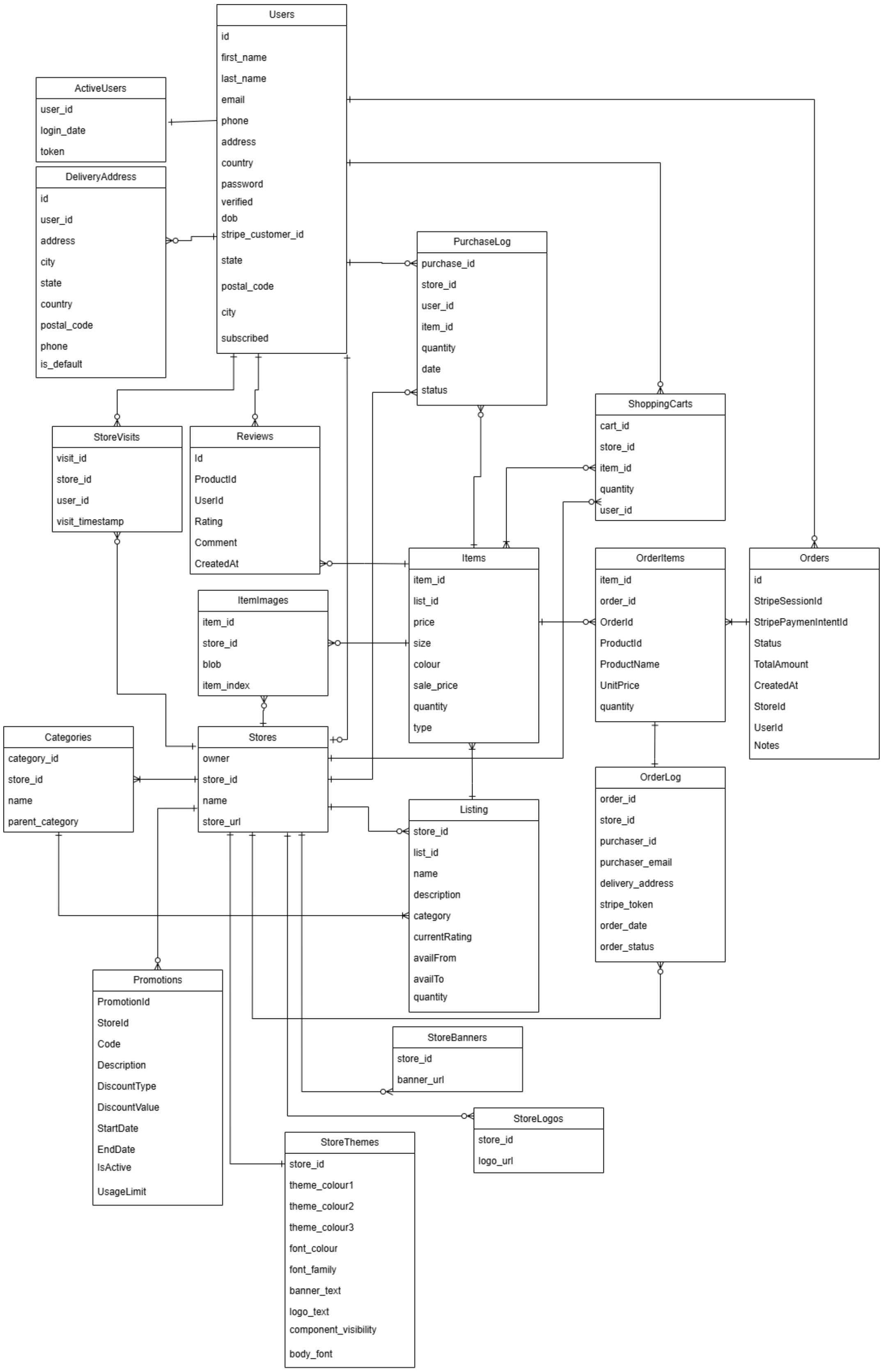
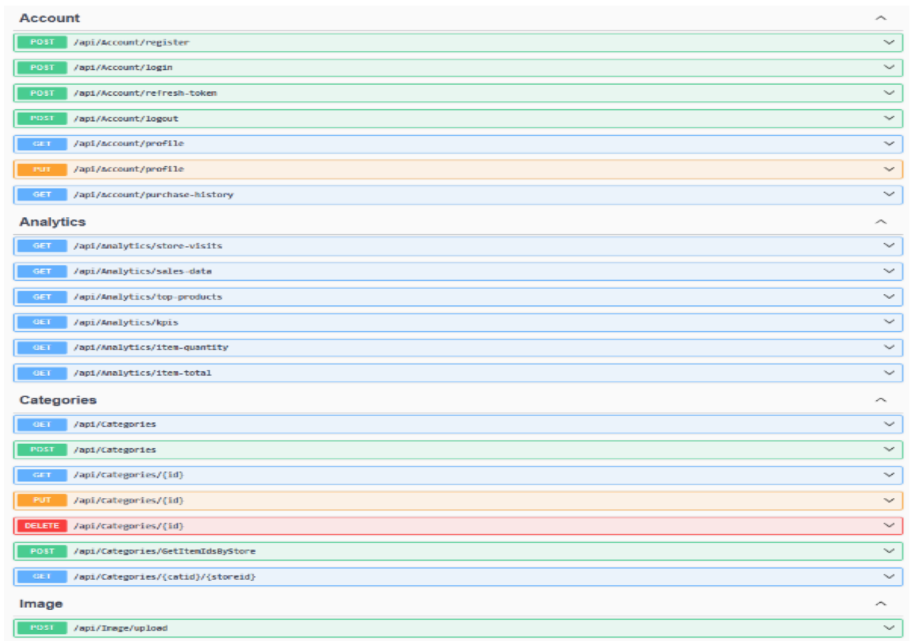


Figure 21 Final ERD for database


Account

POST /api/Account/register
POST /api/Account/login
POST /api/Account/refresh-token
POST /api/Account/logout
GET /api/Account/profile
PUT /api/Account/profile
GET /api/Account/purchase-history

Analytics

GET /api/Analytics/store-visits
GET /api/Analytics/sales-data
GET /api/Analytics/top-products
GET /api/Analytics/kpis
GET /api/Analytics/item-quantity
GET /api/Analytics/item-total

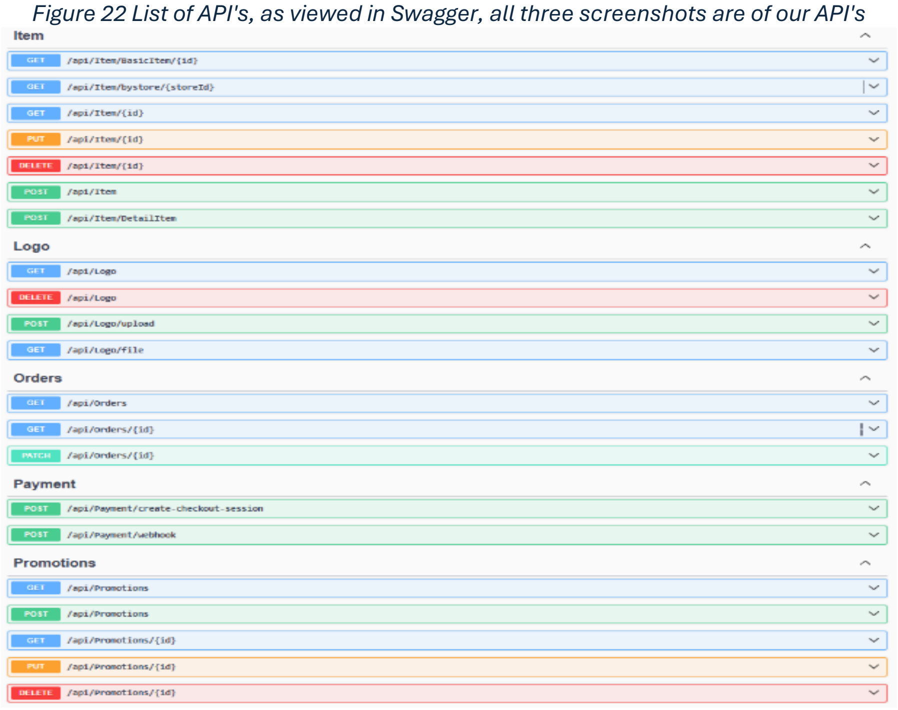
Categories

GET /api/Categories
POST /api/Categories
GET /api/Categories/{id}
PUT /api/Categories/{id}
DELETE /api/Categories/{id}
POST /api/Categories/GetItemIdsByStore
GET /api/Categories/{catid}/{storeid}

Image

POST /api/Image/upload

Figure 22 List of API's, as viewed in Swagger, all three screenshots are of our API's


Item

GET /api/Item/BasicItem/{id}
GET /api/Item/bystore/{storeId}
GET /api/Item/{id}
PUT /api/Item/{id}
DELETE /api/Item/{id}
POST /api/Item
POST /api/Item/DetailItem

Logo

GET /api/Logo
DELETE /api/Logo
POST /api/Logo/upload
GET /api/Logo/file

Orders

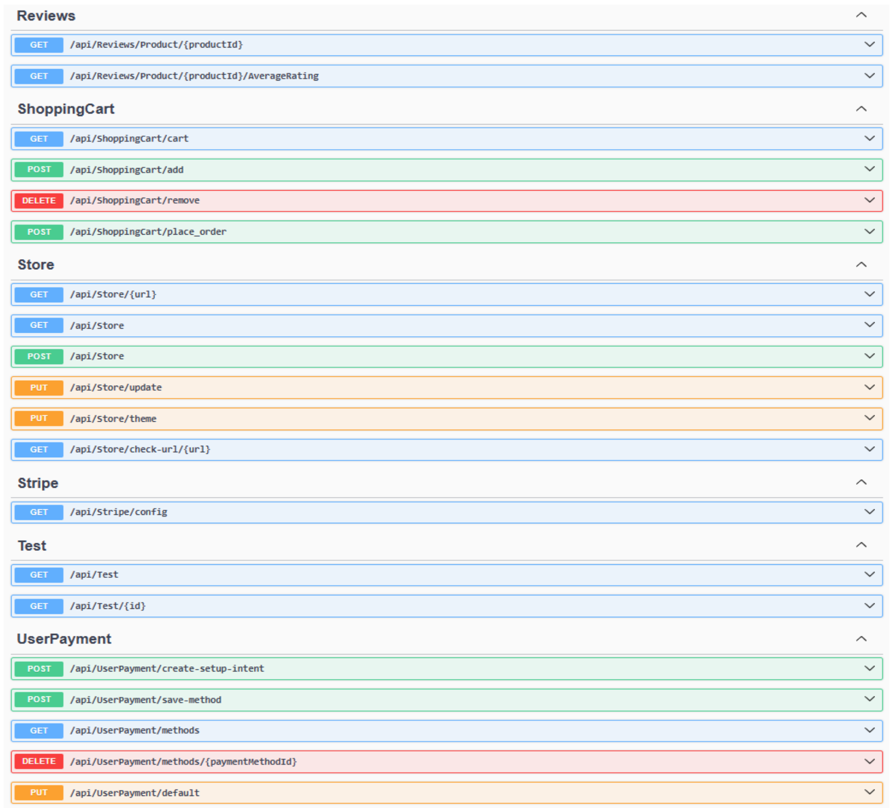
GET /api/Orders
GET /api/Orders/{id}
PATCH /api/Orders/{id}

Payment

POST /api/Payment/create-checkout-session
POST /api/Payment/webhook

Promotions

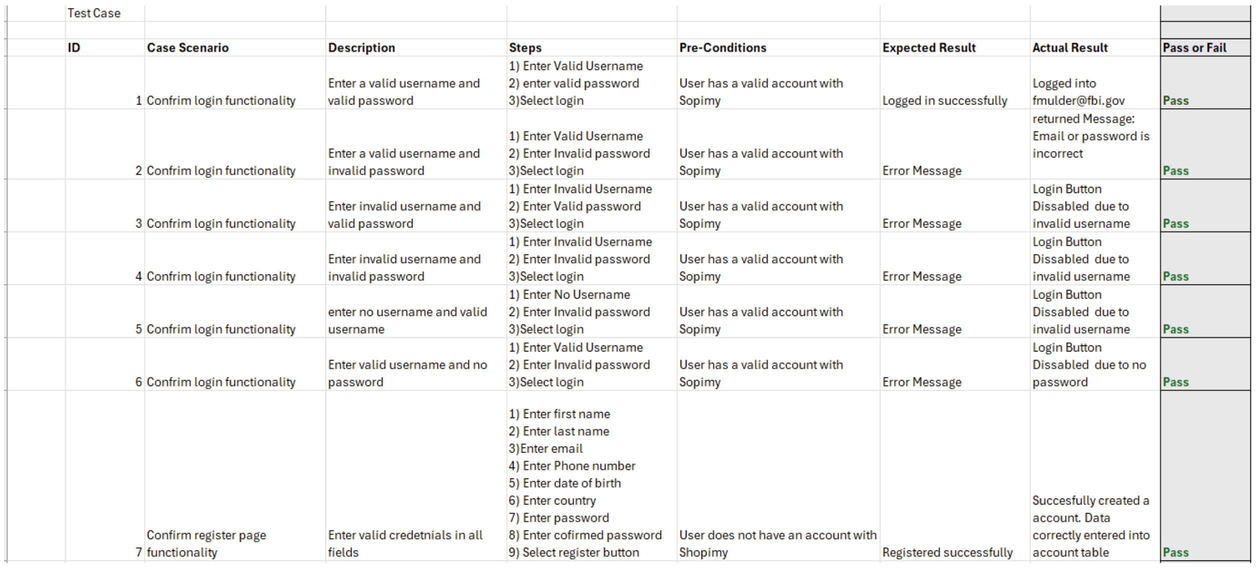
GET /api/Promotions
POST /api/Promotions
GET /api/Promotions/{id}
PUT /api/Promotions/{id}
DELETE /api/Promotions/{id}



```
Reviews
GET     /api/Reviews/Product/{productId}
GET     /api/Reviews/Product/{productId}/AverageRating
ShoppingCart
GET     /api/ShoppingCart/cart
POST    /api/ShoppingCart/add
DELETE  /api/ShoppingCart/remove
POST    /api/ShoppingCart/place_order
Store
GET     /api/Store/{url}
GET     /api/Store
POST    /api/Store
PUT     /api/Store/update
PUT     /api/Store/theme
GET     /api/Store/check-url/{url}
Stripe
GET     /api/Stripe/config
Test
GET     /api/Test
GET     /api/Test/{id}
UserPayment
POST    /api/UserPayment/create-setup-intent
POST    /api/UserPayment/save-method
GET     /api/UserPayment/methods
DELETE  /api/UserPayment/methods/{paymentMethodId}
PUT     /api/UserPayment/default
```

Testing

Testing was an important aspect of our project. Manual testing was conducted by all members of staff on their contribution, to ensure we had the best working product. Towards the end of our project, our team conducted further manual testing on our application to ensure that all functionality was working as anticipated. In the screenshot below (*figure 23*), we can see some of the manual testing that was conducted. The spreadsheet lays out the test case ID, the case scenario, the description of the problem, the steps needed to perform the test, any pre-conditions required, the expected result from our application and the actual result, followed up with a pass/fail label to determine if it meets our requirements. Our team carried out a total of 39 tests, all of which can be found in our repository under the folder Documents/Additional_documents/Manual_testing_spreadsheet.xlsx.



Test Case							
ID	**Case Scenario**	**Description**	**Steps**	**Pre-Conditions**	**Expected Result**	**Actual Result**	**Pass or Fail**
1	Confrim login functionality	Enter a valid username and valid password	1) Enter Valid Username 2) enter valid password 3)Select login	User has a valid account with Sopimy	Logged in successfully	Logged into fmulder@fbi.gov	Pass
2	Confrim login functionality	Enter a valid username and invalid password	1) Enter Valid Username 2) Enter Invalid password 3)Select login	User has a valid account with Sopimy	Error Message	returned Message: Email or password is incorrect	Pass
3	Confrim login functionality	Enter invalid username and valid password	1) Enter Invalid Username 2) Enter Valid password 3)Select login	User has a valid account with Sopimy	Error Message	Login Button Dissabled due to invalid username	Pass
4	Confrim login functionality	Enter invalid username and invalid password	1) Enter Invalid Username 2) Enter Invalid password 3)Select login	User has a valid account with Sopimy	Error Message	Login Button Dissabled due to invalid username	Pass
5	Confrim login functionality	enter no username and valid username	1) Enter No Username 2) Enter Invalid password 3)Select login	User has a valid account with Sopimy	Error Message	Login Button Dissabled due to invalid username	Pass
6	Confrim login functionality	Enter valid username and no password	1) Enter Valid Username 2) Enter Invalid password 3)Select login	User has a valid account with Sopimy	Error Message	Login Button Dissabled due to no password	Pass
7	Confirm register page functionality	Enter valid credetnials in all fields	1) Enter first name 2) Enter last name 3)Enter email 4) Enter Phone number 5) Enter date of birth 6) Enter country 7) Enter password 8) Enter cofirmed password 9) Select register button	User does not have an account with Shopimy	Registered successfully	Succesfully created a account. Data correctly entered into account table	Pass

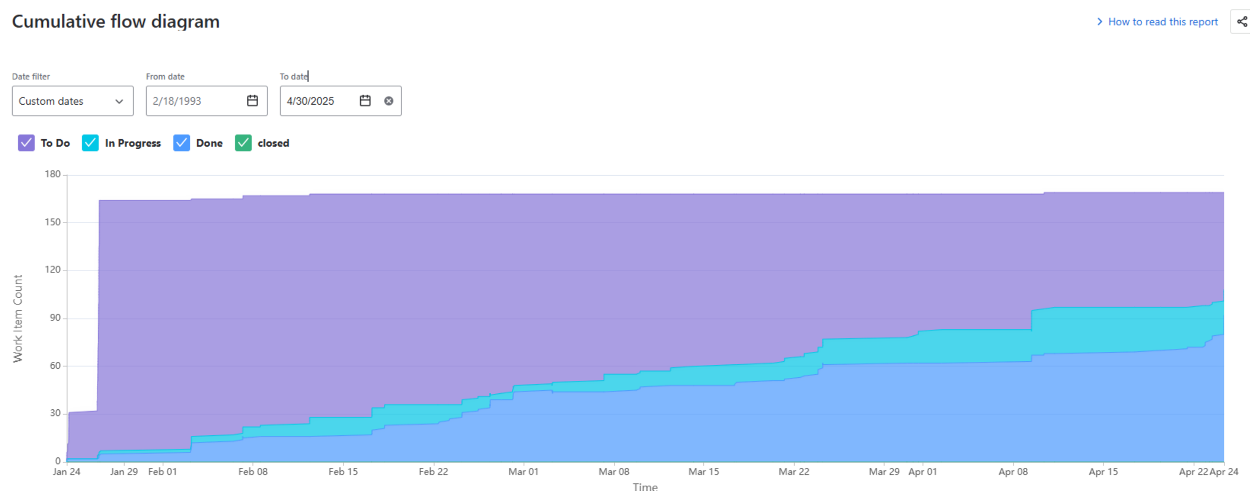
Figure 23 Manual Testing Spreadsheet

Testing our APIs was also important to our team, however due to technical difficulties we are unable to produce any results on coverage. Our team attempted to make use of xUnit.net, a testing tool for the .NET framework. To view the full amount of API's that were to be tested please look to *figure 22*, and the two images after. To view tests written by our API tester, please view our repository and navigate to the folder Server/Tests, to view all written tests.

Jira and GitHub Logs

It should be noted that, while our team was completing tasks, it was due to our want to conduct testing. We are unable to produce fully accurate sprint burn up/down reports as tasks were not marked as done until all subtasks were completed, the same can be said for our cumulative flow diagram. For a complete view of all the user stories and tasks that our team completed please visit our repository and go to Documents/Sprint_Logs, to view a complete list broken down by each sprint of what was completed and what was not.

Additionally, while we feel the insights regarding our GitHub are reflective of most of the work done by the team, we encourage you to look at the contributions section of this report, to see a full scope of what was done by each member.



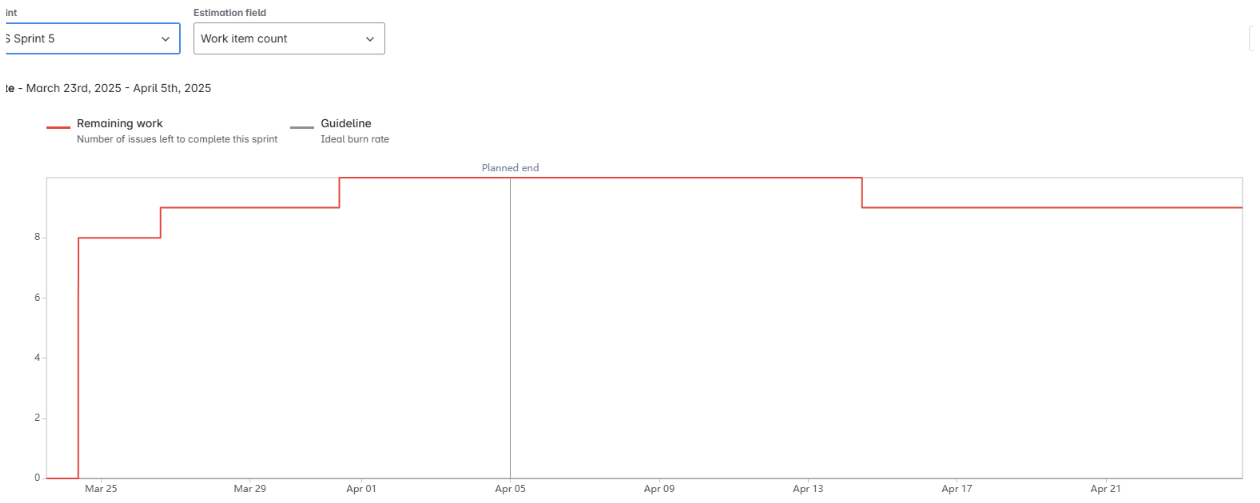
Cumulative flow diagram

Date filter: Custom dates

From date: 2/18/1993

To date: 4/30/2025

Figure 24 Jira Cumulative flow diagram for full development process

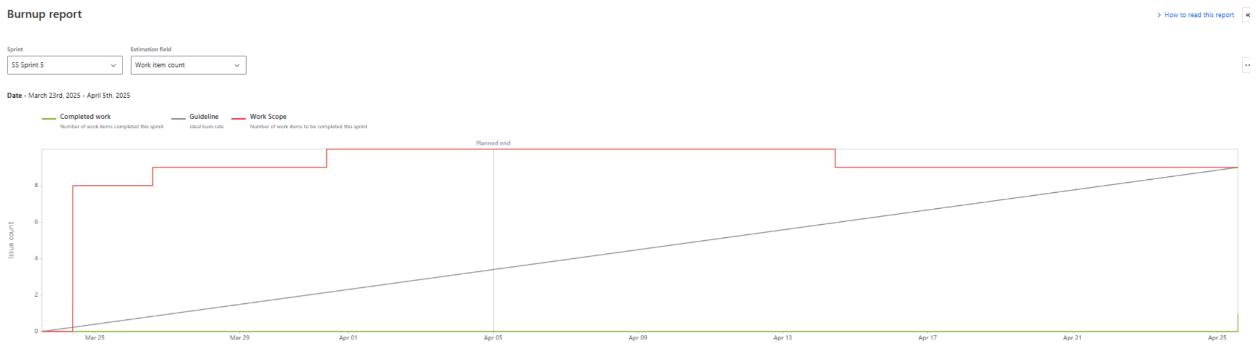


int: 5 Sprint 5

Estimation field: Work item count

te - March 23rd, 2025 - April 5th, 2025

Figure 25 Jira burndown chart of our sprint 5 progress



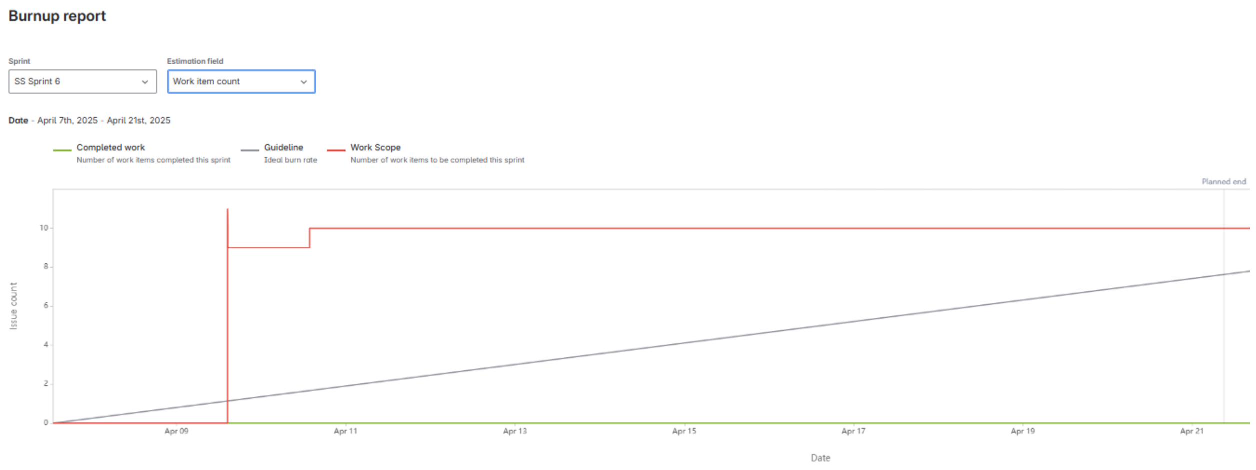
Burnup report

Sprint: SS Sprint 5

Estimation field: Work item count

Date - March 23rd, 2025 - April 5th, 2025

Figure 26 Jira Sprint 5 burnup report

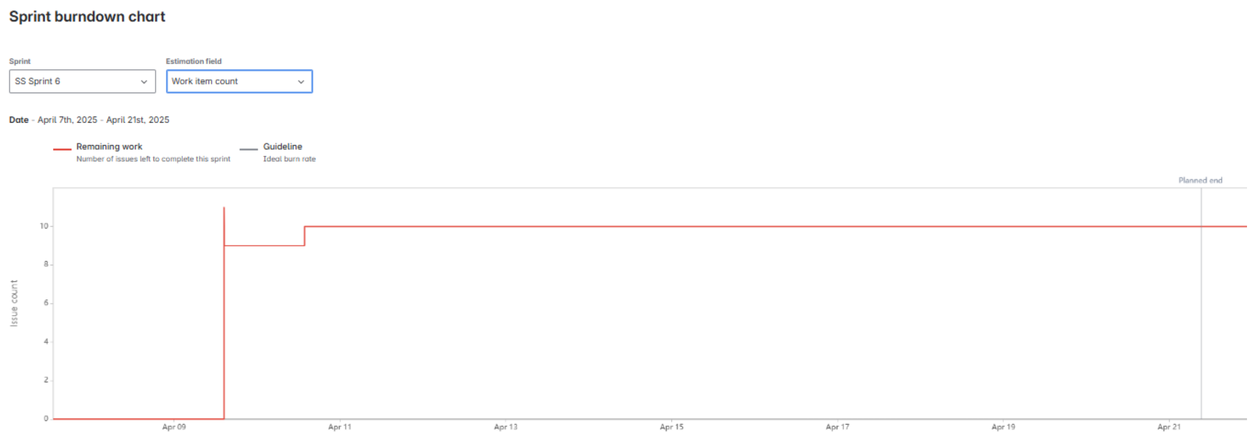

Burnup report

Sprint: SS Sprint 6

Estimation field: Work item count

Date - April 7th, 2025 - April 21st, 2025

Figure 27 Jira Sprint 6 burnup report


Sprint burndown chart

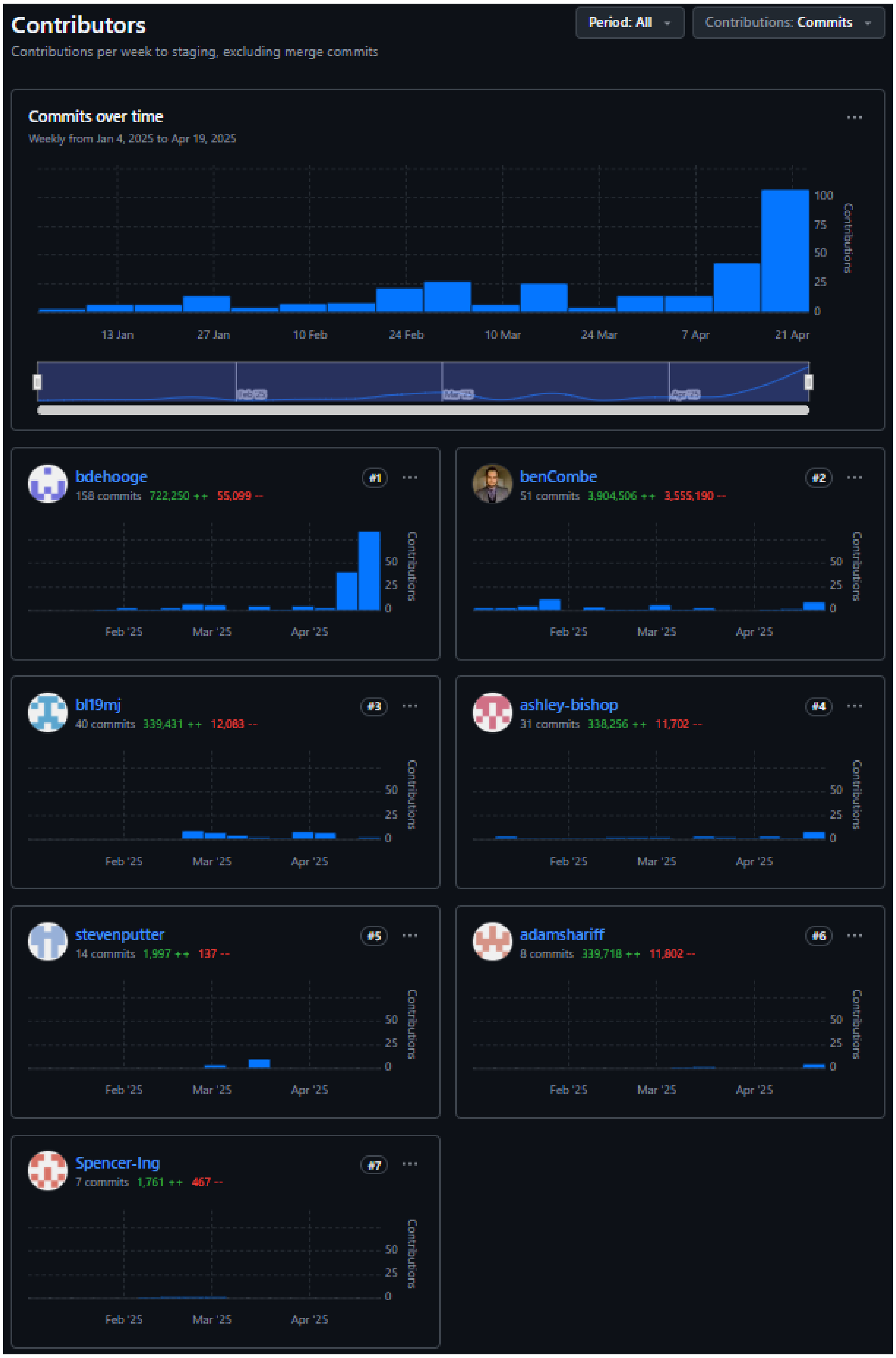
Sprint: SS Sprint 6

Estimation field: Work item count

Date - April 7th, 2025 - April 21st, 2025

Figure 28 Jira Sprint 6 burndown report

To view previous sprints Jira reports, please visit our repository and navigate to the folder Documents/Reports/ then view both our progress reports 1 & 2.



Contributors

Contributions per week to staging, excluding merge commits

Period: All

Contributions: Commits

Commits over time

Weekly from Jan 4, 2025 to Apr 19, 2025

bdehooge #1
158 commits 722,250 ++ 55,099 --

benCombe #2
51 commits 3,904,506 ++ 3,555,190 --

bl19mj #3
40 commits 339,431 ++ 12,083 --

ashley-bishop #4
31 commits 338,256 ++ 11,702 --

stevenputter #5
14 commits 1,997 ++ 137 --

adamshariff #6
8 commits 339,718 ++ 11,802 --

Spencer-Ing #7
7 commits 1,761 ++ 467 --

Figure 29 GitHub insights regarding contributors to the project

Software Engineering Process

Our team completed six sprints over 4 months, each with a varying number of tasks. Our team had an original sprint schedule that got changed due to time constraints, accommodating members. The new updated schedule that we followed is as follows, Sprint One: January 20th – February 2nd, 2025, Sprint Two: February 3rd – February 16th, 2025, Sprint Three: February 24th – March 9th, 2025, Sprint Four: March 12th – March 22nd, 2025, Sprint Five: March 23rd – April 5th, 2025, Sprint Six: April 7th – April 21st, 2025.

Our team met consistently throughout the development process, with three weekly scrum meetings. In which members detailed the tasks they were working on, the problems they were having and discussed solutions. Additionally, we had tutorials on how to set up our technology stack and how the site was functioning with new features that had been implemented.

Our team also had planned sprint review and retrospective meetings in which we discussed the tasks we completed for the sprint and highlighted them for the group. We also reflected on the scrum process, the tools we were using and how we can improve our productivity going forward in our development process. All notes regarding meeting minutes can be found in our repository under the folder Documents/meeting_minutes, notes are broken down by sprint, and the folder Sprint_review_notes houses all notes regarding our reviews and retrospectives.

Additionally, our team made use of Discord to communicate between members about various bugs with Shopimy, tasks and important updates and it is where we held all our scrum meetings. Jira was also used by our team; we ensured it was kept up to date and accurate to do the work. It held all our user stories and tasks that needed to be done and served as a way for us to organize our sprints and assign tasks to members. While it should be noted that due to time constraints, we were unable to complete all the user stories and goals we originally set out to meet, we did do our best to ensure that the core functionalities of Shopimy were implemented. For a complete breakdown of tasks, and who was assigned to them please visit our sprint logs folder in our repository, under the Documents folder.

Challenges

Over the course of developing Shopimy, our team encountered a myriad of problems. While we did our best to mitigate them, there were some persistent problems. One of the first and largest problems we continually encountered was time management. Due to members varies other commitments, and dedication to other classes, we fell behind on our sprint progress, while we did our best to meet this concern during our retrospective meetings and discussed ways, we could be more productive it was a continuous problem. Another challenge that goes hand and hand with this is the distribution of tasks, with some members taking on more work than others to ensure we meet the requirements of our application.

Another large problem our team faced was with our technology stack. Throughout our development, members set up the stack at various points, making it difficult to progress forward. Once the stack was installed most members were unfamiliar with the tools used and required time to try to understand how everything worked together. There were also problems regarding the software xUnit regarding our testing, it also took considerable time for our team to try and get to work, with it being unsuccessful.

Also, regarding software, using Jira was a big hangup for our team. Most members had not used it in the capacity that we were, and working in an environment where it needed to be constantly updated was a challenge. While our team improved its usage of Jira over the project, it could be improved on.

While our team dedicated itself to the scrum process, keeping to our original dates for sprint deadlines was another challenge. To accommodate members of the team over the reading week we changed our sprint schedule. This in turn changed our schedule as we were no longer ending a sprint at the end of the reading week, but rather the week after. This caused somewhat of a domino effect, pushing all our sprints back, and to catch up with our original plans we had a couple sprints that had fewer tasks, and a brief period. Such as a week and a half, with 4 tasks as opposed to a full two weeks with more tasks. In the end we completed six sprints and completed most tasks we originally set out to do.

The last challenge our team encountered was working within a collaborative team environment. Most members only have experience of working individually. Coupled with the new stack and Jira usage, getting used to the scrum meetings, and ensuring members were not doing the same tasks was something that required acclamation.

Successes

Our team is immensely proud of the work that we accomplished over the course of development, while we did encounter problems and setbacks, we feel that Shopimy was developed to the best of our abilities.

Our team feels that our dedication to the software engineering process is a large contributor to our success in developing Shopimy. Our team followed an agile style of development, with three weekly scrum meetings in which members attended frequently and had minimal absences. Additionally, our up-to-date documentation found in our repository shows the amount of effort our team put into Shopimy. Additionally, we feel that our review and retrospective meetings ensured that we were reflecting on our work and considering how to increase our productivity to improve Shopimy.

Our team also feels that the amount of communication between members was also a strength of ours. Our team utilized Discord to discuss various bugs and problems or questions that members had between meetings to ensure that we were using our time most effectively.

We also feel that the number of designs created ensured that our team understood how the application should be implemented. While designs did change as we moved with implementation, the designs provided a foundational basis on the features and schemes that should be implemented within the system.

Our team also feels that an enormous success was the skills that we learned from working on Shopimy. Developing within a collaborative team environment was a new experience for most of our members and gaining that insight we feel will help us move forward in our careers. Additionally, working with new tools and software stacks has ensured that we are more used to the scrum process and requirements needed outside of a school environment. The experience and knowledge we gained we feel was invaluable.

Use of AI

One of the first pieces of advice we received in our feedback meeting for our first progress report was to integrate and use AI to aid us as we move forward with the development of Shopimy.

Our team used a variety of AI aids to help us with both coding assistance and research into how to develop aspects of Shopimy. Our team made use of Chat GPT, Cursor AI, Claude, and Copilot, we feel that our use of a variety of aids helped us to efficiently program Shopimy. These tools helped us to identify bugs and aid us with setting up our stack to an extent which was new to most members.

Full Breakdown:

Anthropic: claude-3.5-sonnet, claude-3.7-sonnet, claude-3.7-sonnet-thinking

Cursor: cursor-small

Google: gemini-2.5-3-25-pro-exp, gemini-2.5-4-17-flash

OpenAI: gpt-4.5-preview, ChatGPT o3-mini-high, o4-mini-high

Contributions

Team Member	Contributions
Ben Combe, 5819446 Scrum Master	• Designed and coded register and login pages, coded backend functionality for register/login/logout • Created user/cookie services for frontend • Configured pages routes/navigation bars • Database setup and hosting • Coded user dashboard layout and profile page, coded shopping cart/checkout pages and coded landing page • Designed and wrote theme service for frontend, and maintained database • Implemented dynamic store, category, and item URL addressing • Implemented shopping cart functionality on frontend and backend • Modified store page and item card for loading item data in frontend • Added frontend UI responsiveness • Setup azure blob storage for images, setup image fetching through API • Oversaw scrum meetings and work of other group members • Led tutorial on how to use GitHub and the stack • Managed and maintained teams GitHub repository, including merging and debugging and handling conflicts • Modified responsiveness for mobile display. • Added order log tables in database. • Prepared demo for presentation • Implemented testing and final fixes. • Started code cleanup
Ashley Bishop, 6693824 Product Owner	• Lead designer for the project, designs include landing page, general shop layout (both customer and shop owner view) products page, product listing page, shopping cart page (followed with payment, shipping and review pages), store dashboard (includes overview, products, orders, analytics, promotions pages), social media integration pages, and the receipt design page • Wrote and complied teams work for progress report one • Managed Jira board • Edited initial designs of item and shop pages, updated listing design • Designed optional preset store layouts for a shop owner to choose from, designed “build your store” pages • Wrote and compiled teams work for the second progress report • Created sprint logs spreadsheets • Wrote and maintained all meeting minutes • Wrote both the user requirements and user manual documentation • Created the Spreadsheet for the manual_testing_spreadsheet, and created and listed out all tests needed to be performed • Created final presentation slideshow • Wrote and complied teams work for the final report

	• Ran all sprint review and retrospective meetings
Adam Shariff, 6768600 Development Team	• Create tables and triggers in backend for items • Created controller for database to add items • Created and updated ER diagram • Added additional functionalities for displaying items and categories in backend controllers • Coded backend calls for getting item data • Added trigger for calculating item functionality • Coded backend calls for getting item variation • Coded backend item detail. • Coded backend and frontend graphs and analytics for store dashboard
Ben DeHooge, 6567069 Development Team	**Landing Page Styling** • **Frontend:** `LandingPageComponent.css`, `styles.css` (global), `README-STYLES.md` • **Backend:** N/A **Landing Page TopNav** • **Frontend:** `TopNavComponent`, `UserService` (for login status), `PublicLayoutComponent` (includes TopNav) • **Backend:** `AccountController` (implicitly via `UserService` for login state) **Landing Page Hamburger/Dropdown** • **Frontend:** `TopNavComponent` (handles mobile menu logic and display) • **Backend:** N/A **Footer** • **Frontend:** `FooterComponent`, `PublicLayoutComponent` (includes Footer) • **Backend:** N/A **Entirety of About page** • **Frontend:** `AboutUsComponent`, `PublicLayoutComponent` • **Backend:** N/A (Static content) **Entirety of Contact page** • **Frontend:** `ContactComponent`, `PublicLayoutComponent`

- **Backend:** N/A (Currently static, might use a future `ContactController`)

Entirety of Support page

- **Frontend:** `SupportComponent`, `PublicLayoutComponent`
- **Backend:** N/A (Static content)

Entirety of Privacy Policy page

- **Frontend:** `PrivacyPolicyComponent`, `PublicLayoutComponent`
- **Backend:** N/A (Static content)

Entirety of Terms of Service page

- **Frontend:** `TermsOfServiceComponent`, `PublicLayoutComponent`
- **Backend:** N/A (Static content)

Entirety of Documentation page

- **Frontend:** `DocsComponent`, `PublicLayoutComponent`, `HttpClient` (to load markdown)
- **Backend:** N/A (Serves static .md files)

Entirety of Blog page

- **Frontend:** `BlogComponent`, `PublicLayoutComponent`
- **Backend:** N/A (Static content, could have a future `BlogController`)

Login form styling and functionality

- **Frontend:** `LoginComponent`, `UserService`, `CookieService`
- **Backend:** `AccountController` (handles login POST request), `Users` table, `ActiveUsers` table

Register form styling and functionality

- **Frontend:** `RegisterComponent`, `UserService`, `CookieService`
- **Backend:** `AccountController` (handles register POST request), `Users` table

Dashboard:

- **Overview Styling, Recent Sales Analytic, Visits to Store Analytic, functionality and styling of Quick Links**
    - **Frontend:** `StoreOwnerDashboardComponent`, `OverviewComponent`, `AnalyticsService`, `OrderService`, Chart.js
    - **Backend:** `AnalyticsController`, `OrdersController`, `AnalyticsService`, `Orders` table, `StoreVisits` table

- **Account: Profile: Account Info, Payment Methods, Delivery Addresses, Purchase history styling and functionality**
  - **Frontend:** `StoreOwnerDashboardComponent`, `ProfileComponent`, `UserService`, `PaymentService`, `DeliveryService`, `PurchaseService`, Stripe.js
  - **Backend:** `AccountController` (profile info, purchase history), `UserPaymentController` (Stripe payment methods), `UserDeliveryController`, `Users` table, `Orders` table, `OrderItems` table, `DeliveryAddresses` table, Stripe API

- **Account: Settings: Account Information, Payment Information styling and functionality**
  - **Frontend:** `StoreOwnerDashboardComponent`, `SettingsComponent`, `UserService`, `PaymentService`
  - **Backend:** `AccountController` (profile update), `UserPaymentController` (payment settings), `Users` table

- **My Store: Products: Your Products list, Add New Product form, Variant form Styling and functionality**
  - **Frontend:** `StoreOwnerDashboardComponent`, `ProductManagementComponent`, `ItemService`, `ImageService` (for uploads)
  - **Backend:** `ItemController` (CRUD), `ImageController` (uploads), `CategoriesController` (for category selection), `Listing` table, `Items` table, `ItemImages` table

- **My Store: Categories: Create, Edit, Delete, Table View, Grid View styling and functionality**
  - **Frontend:** `StoreOwnerDashboardComponent`, `CategoryListComponent`, `CategoryFormComponent`, `CategoryService`
  - **Backend:** `CategoriesController`, `CategoryService`, `CategoryRepository`, `Categories` table

- **My Store: Orders: Order Statistics, Order List styling and functionality**
  - **Frontend:** `StoreOwnerDashboardComponent`, `OrdersComponent`, `OrderService`, `AnalyticsService`
  - **Backend:** `OrdersController`, `AnalyticsController` (stats), `AnalyticsService`, `Orders` table, `OrderItems` table

- **My Store: Themes & Logos: Store Appearance: Select Themes, Colours & Typography, Brand Logo all styling and functionality**

    - **Frontend:** `StoreOwnerDashboardComponent`, `ThemesComponent`, `LogoSelectorComponent`, `ThemeService`, `LogoService`, `StoreService`
    - **Backend:** `StoreController` (update theme/text), `LogoController` (upload/delete logo), `LogoService`, `StoreThemes` table, `StoreLogos` table
- **My Store: Store Editor: Basic Info/ Theme/ Components/ Products form styling and functionality, Live Store Preview styling and functionality**
    - **Frontend:** `StoreOwnerDashboardComponent`, `StoreEditorComponent`, `ThemesComponent`, `ComponentsSettingsComponent`, `StorePreviewComponent`, `StoreService`, `ThemeService`, `ItemService` (if product quick add)
    - **Backend:** `StoreController` (create/update store), `ItemController` (if products managed), `Stores` table, `StoreThemes` table

- **My Store: Promotions: Create, edit, delete promotion styling and functionality.**
    - **Frontend:** `StoreOwnerDashboardComponent`, `PromotionsComponent`, `PromotionsService`
    - **Backend:** `PromotionsController`, `PromotionsService`, `Promotions` table

- **Analytics: Dashboard Component (Total Revenue, Total Orders, etc) styling and functionality.**
    - **Frontend:** `StoreOwnerDashboardComponent`, `AnalyticsComponent`, `AnalyticsService`, Chart.js
    - **Backend:** `AnalyticsController`, `AnalyticsService`, `Orders` table, `StoreVisits` table

- **Analytics: Sales Trends, Store Traffic, Top Selling Products component styles and functions.**
    - **Frontend:** `AnalyticsComponent` (contains logic/sub-components), `AnalyticsService`, Chart.js
    - **Backend:** `AnalyticsController`, `AnalyticsService`, `Orders` table, `OrderItems` table, `StoreVisits` table

Store:

- **Styling and functionality of all components** (Public store view)
    - **Frontend:** `StorePageComponent`, `StoreHeaderComponent`, `HeroBannerComponent`, `FeaturedProductsComponent`, `CategoriesComponent`, `TestimonialsComponent`, `NewsletterComponent`,

`StoreFooterComponent`, `ItemCardComponent`, `CategoryPageComponent`, `ItemDetailComponent`, `ProductReviewsComponent`, `StoreService`, `ItemService`, `CategoryService`, `ReviewService`, `ThemeService`, `StoreNavComponent`, `ShoppingCartComponent`, `OrderSummaryComponent`
  - **Backend:** `StoreController`, `ItemController`, `CategoriesController`, `ReviewsController`

- **Theme and Logo injection**
  - **Frontend:** `StorePageComponent`, `ThemeService`, `StoreHeaderComponent`
  - **Backend:** `StoreController`, `StoreThemes` table, `StoreLogos` table

- **Store Nav bar (different than the Main/Dashboard ones) styling and functionality**
  - **Frontend:** `StoreNavComponent`, `StoreNavService`, `ShoppingService` (cart count), `UserService` (login status)
  - **Backend:** `CategoriesController` (for category dropdown)

- **Cart List, Order Summary, Promo Code components styling and functionality**
  - **Frontend:** `ShoppingCartComponent`, `OrderSummaryComponent`, `ShoppingService`, `PromotionsService`
  - **Backend:** `ShoppingCartController`, `PromotionsController`, `ShoppingCarts` table, `Promotions` table

- **Checkout form styling and functionality**
  - **Frontend:** `CheckoutComponent`, `PaymentService`, `ShoppingService`, Stripe.js
  - **Backend:** `PaymentController`, `WebhookController` (logic within PaymentController), `Orders` table, `OrderItems` table, Stripe API

- **Supporting Changes & Fixes:**
- **Backend:**
  - `AccountController.cs`: Added `GetPurchaseHistory` endpoint and DTOs.

	- `StoreController.cs`: Updated to handle `componentVisibility` saving/loading. Refinements to store creation/update logic. Added `/check-url` endpoint. - `ItemController.cs`: Updated for full product lifecycle management (Draft/Publish using `availFrom`, variant image URL handling). - `ImageController.cs`: Added/used for handling image uploads (URLs). - `OrdersController.cs`: Updated/added endpoints for viewing orders. - `UserPaymentController.cs`: Centralized Stripe key configuration potentially moved to `Program.cs`. - `AppDbContext.cs`: Updated with `DbSet`s for new tables. - `Migrations/`: Reflect various schema changes including those for orders, component visibility, and analytics/promotions. - `Program.cs`: Added dependency injection for new services (Analytics, Promotions). Added `StoreClaimMiddleware`. **Frontend:** - `ProfileComponent`: Updated to display purchase history. - `PurchaseService`: Added/updated to fetch purchase history. - `ProductManagementComponent`: Significantly updated UI and logic for variant management, image URLs, and status handling. - `ItemService`: Updated interfaces/methods to support the product management changes. - `TopNavComponent`: Added link to Blog. Updated user dropdown/mobile menu logic (e.g., conditionally showing "My Store"). - `StoreNavComponent`: Updated navigation logic. - `SideNavComponent`: Updated navigation logic (e.g., added Analytics). - `StoreOwnerDashboardComponent`: Logic updated to handle routing/display of new components (Analytics, Promotions, potentially updated Products/Categories views). Checks if user `hasStore`. - `DocsComponent`: Added to display project documentation fetched from assets. - `copy-docs.js`: Added script to facilitate copying documentation files. - `package.json` / `package-lock.json`: Added `chart.js`. Updated Angular and other dependencies to specified versions. - Styling (`styles.css` and component CSS): General consistency improvements and additions for new components.

	• **Database:** ◦ `TableCreation.sql`: Includes schema for `StoreVisits`, `Promotions`, `Orders`, `OrderItems`, `Listing`, `Items`, `ItemImages`. Updates to `Users` (address fields), `StoreThemes` (`component_visibility`). • **Documentation:** Extensive updates across many `.md` files to reflect the new architecture, features, status, and development processes. • **Bug Fixes:** ◦ Fixed user information disappearing on dashboard reload (`status.md`, potentially related to `AppComponent` and `UserService` initialization). ◦ Fixed analytics API query/error handling. ◦ Resolved Chart.js rendering issues.
Spencer Ing, 6756605 Development Team	• Began coding frontend store owner dashboard, worked on implementing front-End designs for user dashboards to view account and store information • worked on implementing backend API calls for user dashboard to request and upload data from the database • Tested and reported if front-end components successfully loaded and properly made calls to back-end API, • tested website responsiveness and performed according to design documentation.
Braden Lucas, 6880462 Development Team	• Designed Profile page • Built desktop view layout for the general store layout • Created the item page and reviews component • Designed and created mobile layout for item page and reviews component • Attempted creation of scrolling banners for store page • Final review and touch ups on various pages
Steven Putter, 6966048 API Tester	• Research regarding API testing • Created Api testing for store controller and shopping cart controller. • Troubleshooted API testing • Tested and found bugs with the final product, reviewed responsiveness of some pages