SWIFT SHOP

Project Report

https://github.com/stephenasuncionDEV/swift-shop

By Damandeep Singh, Ryuho Kikuchi, Cyrus Lum, and Stephen Asuncion

Table of Contents

Overview

Swift Shop is an ecommerce platform dedicated for selling tech products. Our goal is to provide the best electronics to consumers at the best prices and in an efficient way.

How we achieve these goals is by offering customers a variety of products to choose from as well as providing a UX design that will allow great accessibility throughout our website.

We wanted to make this project as collaborative as possible so we are going to take advantage of using various collaboration tools such as the following:

- Figma.com - for designing the website
- Notion.so - for project management and scheduling
- Discord - for team discussion/meeting
- Google Docs, Slides and Sheets - for documenting and planning
- Github - for version source control
- Whimsical - for creating data flow

The schedule and budget/costs plays a big role in this project. This project is estimated to be completed before the month of April. We intend to finish this project within a month. Furthermore, we don't have an initial budget to start this project but there should be no costs on building this project. Every service that we are planning to use is free.

During the development of this project, we have taken the time to gather the necessary resources such as SDLC model, User Stories, Technology Stack, APIs, Work Breakdown Structure, Schedules, Wireframes, and Prototype Elements.

Software Development Life Cycle (SDLC)

The SDLC model we chose for this project is the Agile methodology and we are using Scrum as our agile framework. We chose this methodology because we do expect that things may not go exactly as planned. As agile methodology offers the flexibility of having changes at any stage, we could analyze our project at the end of each sprint and could find the problems and fix them in the next sprint.

We are using notion.so as a project management tool because it offers flexibility and it comes with powerful tools that could help in increasing the productivity level. Though it takes time to set up, it makes it very easy to collaborate with the team and keep everyone up to date.

User Stories

Commerce.js

1. As a tech company owner, I want to expand my business by reaching more customers through an online platform. I want to attract more buyers by allowing them to have detailed information about products including description, price, images etc on this online website.

2. As a website owner, I want users to be able to buy more than one product and have flexibility in adding or removing things at any point before the payment is made.Users should be able to see all the products they wish to buy at a single place.

Stripe

1. I want my website to be available in multiple countries, so I want to have flexibility in making payment for the order. There should be different methods available for payment and a user should be able to make payment in their local currency.

2. To manage payment or order related conflicts, I want the user to have a complete record of his/her order, where the user can see the order number, products and their prices, total, amount paid and due etc.

Technology Stack

Our technology stack consists of Next.js for the front-end, and Chakra UI for the framework. We chose Next.js because it is very similar to React.js and it is easier to use. With Next.js, we are going to be able to have faster compilation time which results in faster development. It will also allow us to deploy faster because Next.js is compatible with Vercel. Vercel allows us to deploy our Github repository directly (any changes made from the main/master branch on Github will get deployed automatically).

We are going to use Chakra UI framework because it will also speed up our development as it is simple to use, and it is modular. Instead of coding everything in vanilla css, we will have read-made components that are easy to integrate and customize.

APIs

1) Commerce.js: The first reason we chose this API is that it includes almost everything that is required to build a successful ecommerce website. Secondly, it is free. Commerce.js includes products, checkout, cart which are what we exactly want. Every feature is properly documented which would make it easy for us to understand and implement in our projects.

2) Stripe: It is one of the popular API used by developers. It is very important to charge customers correctly to prevent issues. We chose this API as it is reliable and predictable. Moreover, most of the companies are making use of this API, which means it will be good for us to have some experience with this API to have a good chance of getting jobs in future. Their API is easy to use with a complete documentation of every feature it provides.

Planned API Features

Commerce.js

1. Store/Display product information - Images, price, description: We are going to implement an interface similar to amazon or best buy, where users could see the details about a particular product by clicking on it. So, we will be importing the images, product information- description, price etc to achieve this task.

2. Cart - We will be allowing users to buy more than one product by allowing them to add products in cart. The users could also remove products from the cart if they want to.
3. Search - To help users to find a specific product they want to buy easily, we will be implementing a search toolbar, where users could type the keywords or full product name to find it.

Stripe

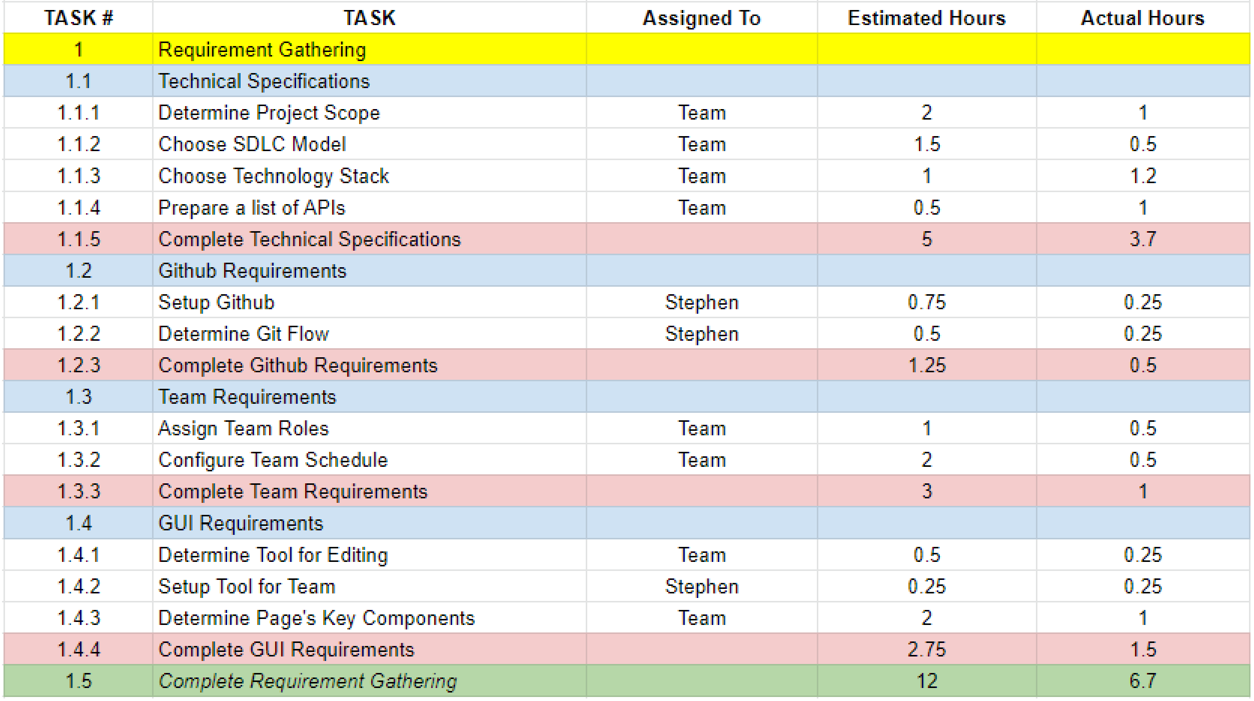
1. Payment Methods - We will allow users to pay with credit card, debit card, gift card (promo code)

2. Refund/Cancel Products - We will allow users to refund or cancel their products
3. Invoice Generation - After the payment is successful, we will generate an invoice containing all the products, their prices and their total sum and amount paid for the future reference of the customer.

Work Breakdown Structure (WBS)

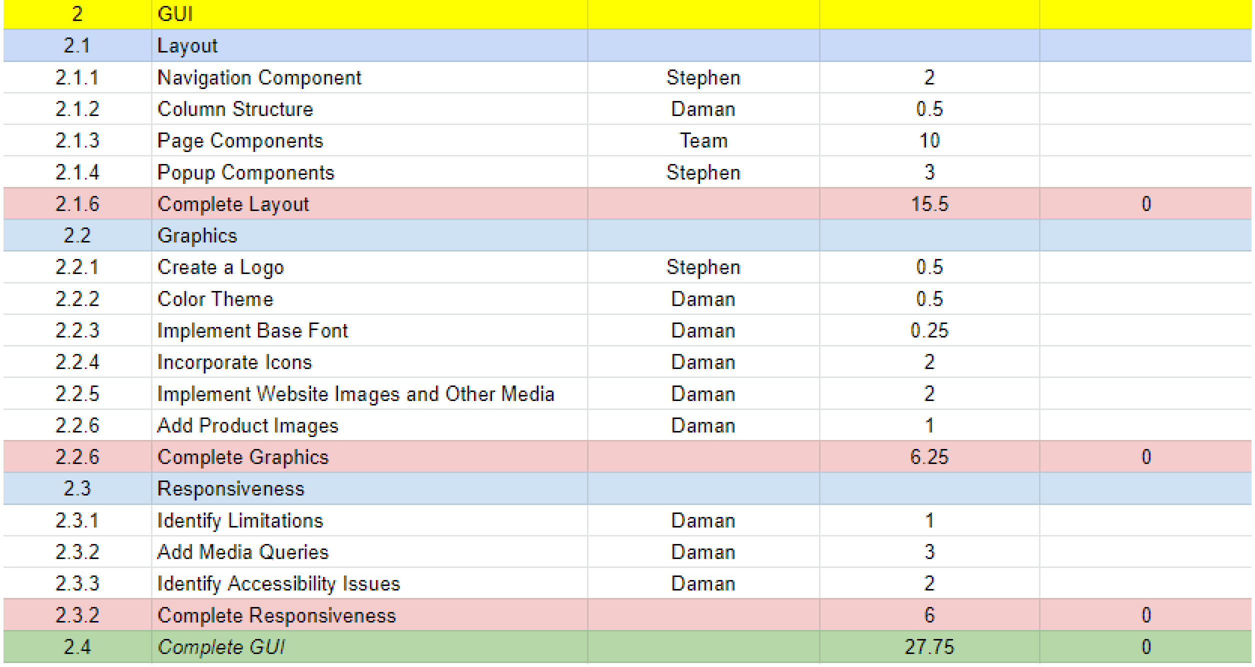
Each task was assigned based on our skills and abilities. In our project management, tasks were divided up by sprints and priority level. Each sprint has a timeframe of 1 week. We left a 1 week margin before the month of April to consider possible delays that may occur.

Requirement Gathering



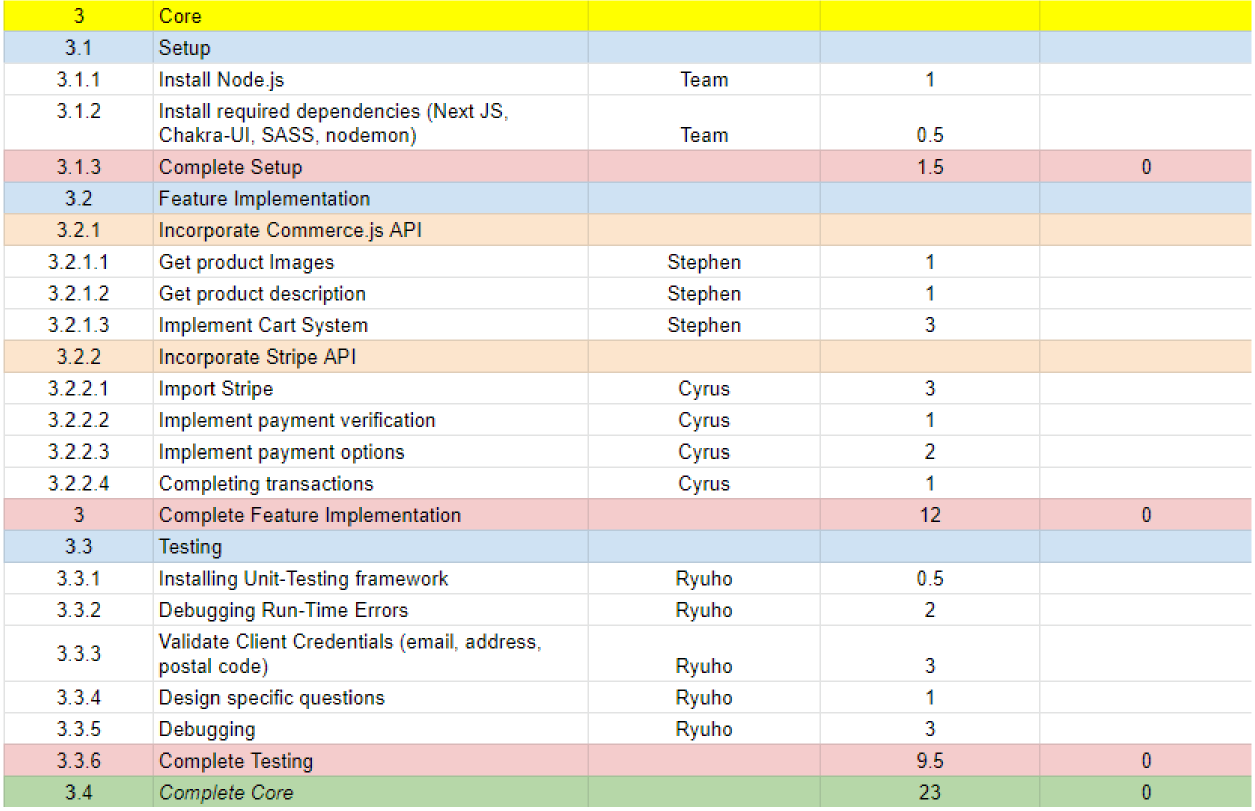
TASK #	TASK	Assigned To	Estimated Hours	Actual Hours
1	Requirement Gathering			
1.1	Technical Specifications			
1.1.1	Determine Project Scope	Team	2	1
1.1.2	Choose SDLC Model	Team	1.5	0.5
1.1.3	Choose Technology Stack	Team	1	1.2
1.1.4	Prepare a list of APIs	Team	0.5	1
1.1.5	Complete Technical Specifications		5	3.7
1.2	Github Requirements			
1.2.1	Setup Github	Stephen	0.75	0.25
1.2.2	Determine Git Flow	Stephen	0.5	0.25
1.2.3	Complete Github Requirements		1.25	0.5
1.3	Team Requirements			
1.3.1	Assign Team Roles	Team	1	0.5
1.3.2	Configure Team Schedule	Team	2	0.5
1.3.3	Complete Team Requirements		3	1
1.4	GUI Requirements			
1.4.1	Determine Tool for Editing	Team	0.5	0.25
1.4.2	Setup Tool for Team	Stephen	0.25	0.25
1.4.3	Determine Page's Key Components	Team	2	1
1.4.4	Complete GUI Requirements		2.75	1.5
1.5	*Complete Requirement Gathering*		12	6.7

GUI



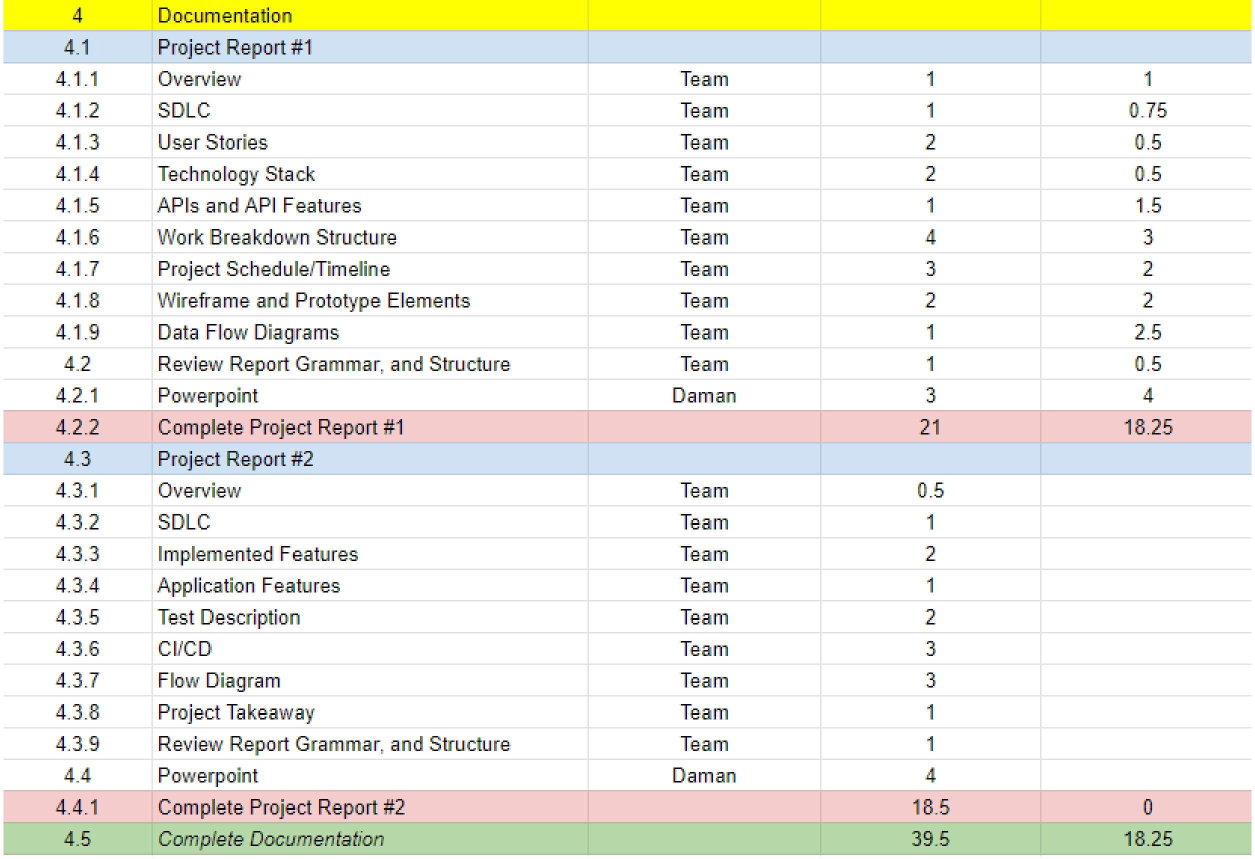
2	GUI			
2.1	Layout			
2.1.1	Navigation Component	Stephen	2	
2.1.2	Column Structure	Daman	0.5	
2.1.3	Page Components	Team	10	
2.1.4	Popup Components	Stephen	3	
2.1.6	Complete Layout		15.5	0
2.2	Graphics			
2.2.1	Create a Logo	Stephen	0.5	
2.2.2	Color Theme	Daman	0.5	
2.2.3	Implement Base Font	Daman	0.25	
2.2.4	Incorporate Icons	Daman	2	
2.2.5	Implement Website Images and Other Media	Daman	2	
2.2.6	Add Product Images	Daman	1	
2.2.6	Complete Graphics		6.25	0
2.3	Responsiveness			
2.3.1	Identify Limitations	Daman	1	
2.3.2	Add Media Queries	Daman	3	
2.3.3	Identify Accessibility Issues	Daman	2	
2.3.2	Complete Responsiveness		6	0
2.4	*Complete GUI*		27.75	0

Core



3	Core			
3.1	Setup			
3.1.1	Install Node.js	Team	1	
3.1.2	Install required dependencies (Next JS, Chakra-UI, SASS, nodemon)	Team	0.5	
3.1.3	Complete Setup		1.5	0
3.2	Feature Implementation			
3.2.1	Incorporate Commerce.js API			
3.2.1.1	Get product Images	Stephen	1	
3.2.1.2	Get product description	Stephen	1	
3.2.1.3	Implement Cart System	Stephen	3	
3.2.2	Incorporate Stripe API			
3.2.2.1	Import Stripe	Cyrus	3	
3.2.2.2	Implement payment verification	Cyrus	1	
3.2.2.3	Implement payment options	Cyrus	2	
3.2.2.4	Completing transactions	Cyrus	1	
3	Complete Feature Implementation		12	0
3.3	Testing			
3.3.1	Installing Unit-Testing framework	Ryuho	0.5	
3.3.2	Debugging Run-Time Errors	Ryuho	2	
3.3.3	Validate Client Credentials (email, address, postal code)	Ryuho	3	
3.3.4	Design specific questions	Ryuho	1	
3.3.5	Debugging	Ryuho	3	
3.3.6	Complete Testing		9.5	0
3.4	*Complete Core*		23	0

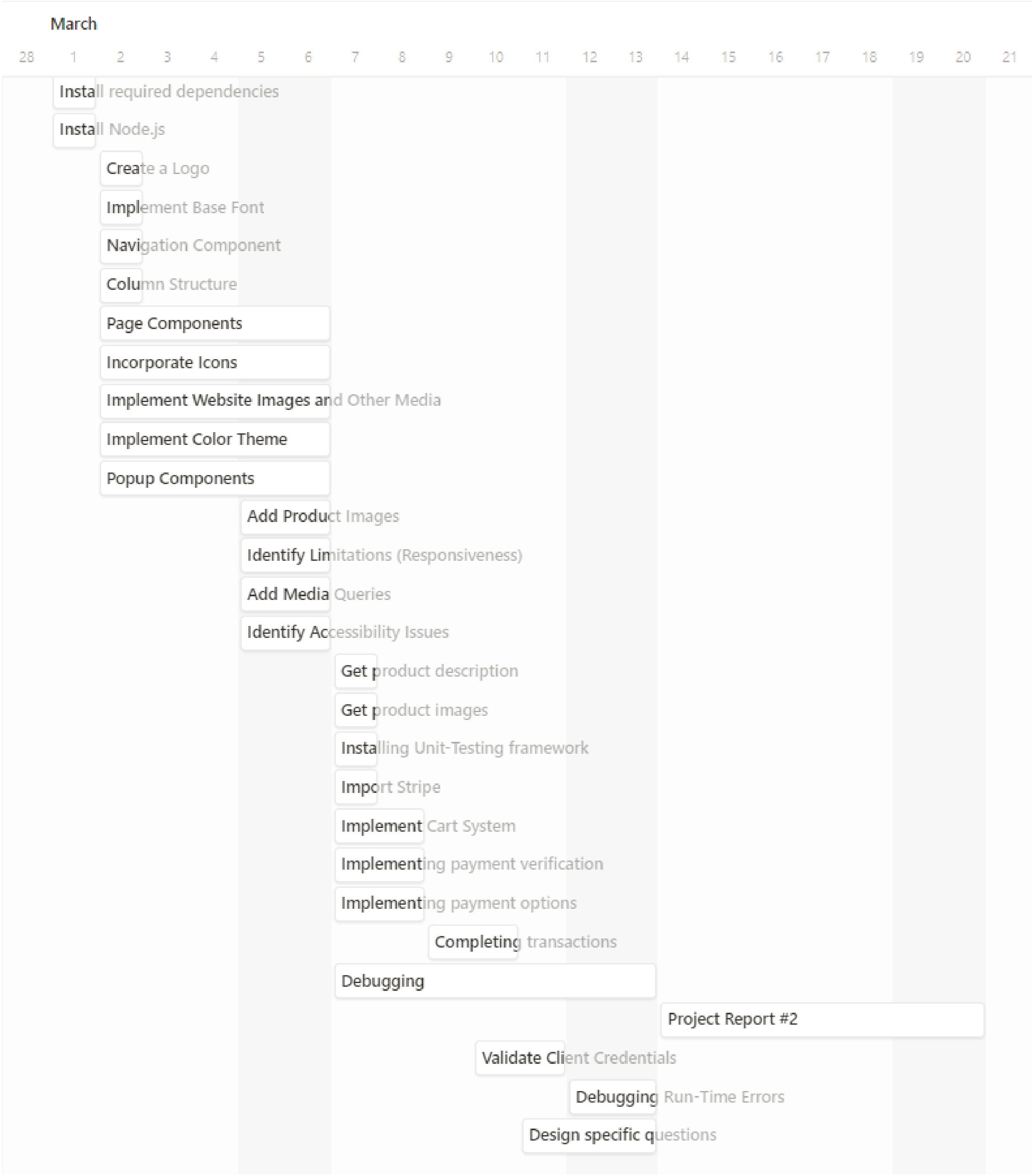
Documentation



4	Documentation			
4.1	Project Report #1			
4.1.1	Overview	Team	1	1
4.1.2	SDLC	Team	1	0.75
4.1.3	User Stories	Team	2	0.5
4.1.4	Technology Stack	Team	2	0.5
4.1.5	APIs and API Features	Team	1	1.5
4.1.6	Work Breakdown Structure	Team	4	3
4.1.7	Project Schedule/Timeline	Team	3	2
4.1.8	Wireframe and Prototype Elements	Team	2	2
4.1.9	Data Flow Diagrams	Team	1	2.5
4.2	Review Report Grammar, and Structure	Team	1	0.5
4.2.1	Powerpoint	Daman	3	4
4.2.2	Complete Project Report #1		21	18.25
4.3	Project Report #2			
4.3.1	Overview	Team	0.5	
4.3.2	SDLC	Team	1	
4.3.3	Implemented Features	Team	2	
4.3.4	Application Features	Team	1	
4.3.5	Test Description	Team	2	
4.3.6	CI/CD	Team	3	
4.3.7	Flow Diagram	Team	3	
4.3.8	Project Takeaway	Team	1	
4.3.9	Review Report Grammar, and Structure	Team	1	
4.4	Powerpoint	Daman	4	
4.4.1	Complete Project Report #2		18.5	0
4.5	*Complete Documentation*		39.5	18.25

Project Schedule/Timeline

March



Install required dependencies

Install Node.js

Create a Logo

Implement Base Font

Navigation Component

Column Structure

Page Components

Incorporate Icons

Implement Website Images and Other Media

Implement Color Theme

Popup Components

Add Product Images

Identify Limitations (Responsiveness)

Add Media Queries

Identify Accessibility Issues

Get product description

Get product images

Installing Unit-Testing framework

Import Stripe

Implement Cart System

Implementing payment verification

Implementing payment options

Completing transactions

Debugging

Project Report #2

Validate Client Credentials

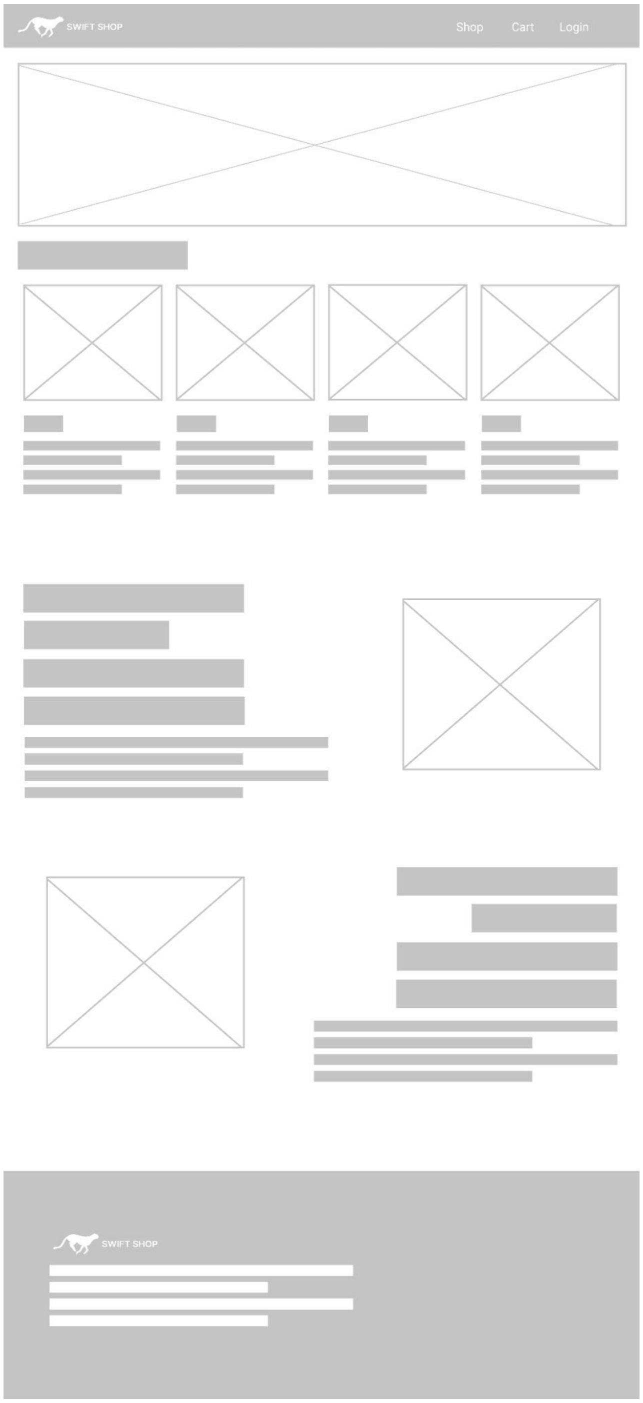
Debugging Run-Time Errors

Design specific questions

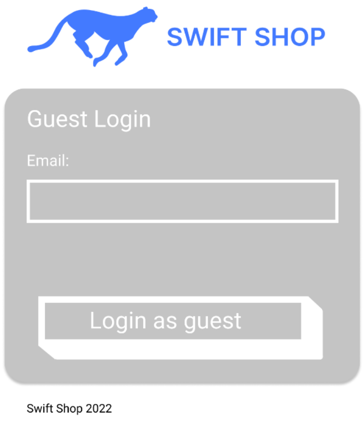
Wireframes and Prototype Elements

Wireframe

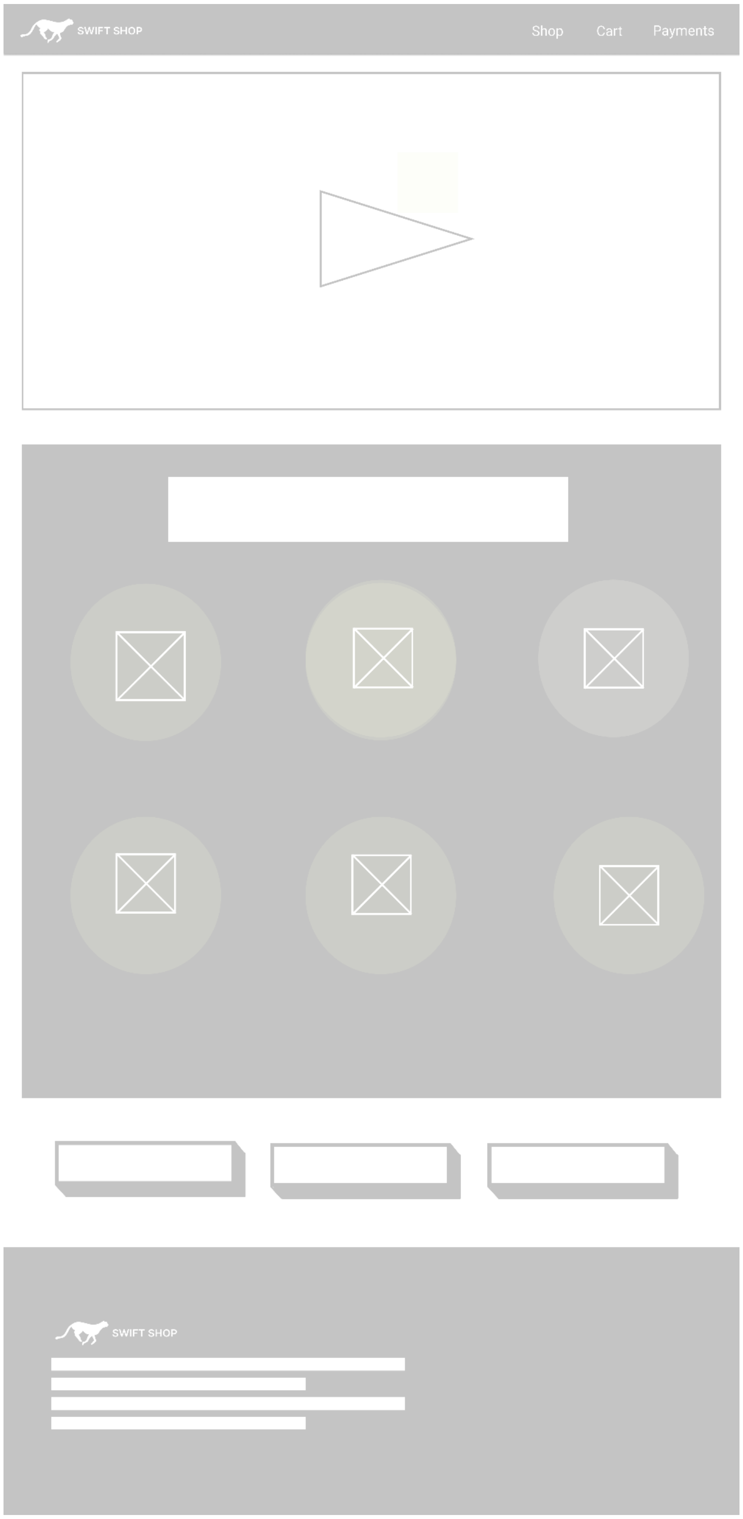
Landing Page



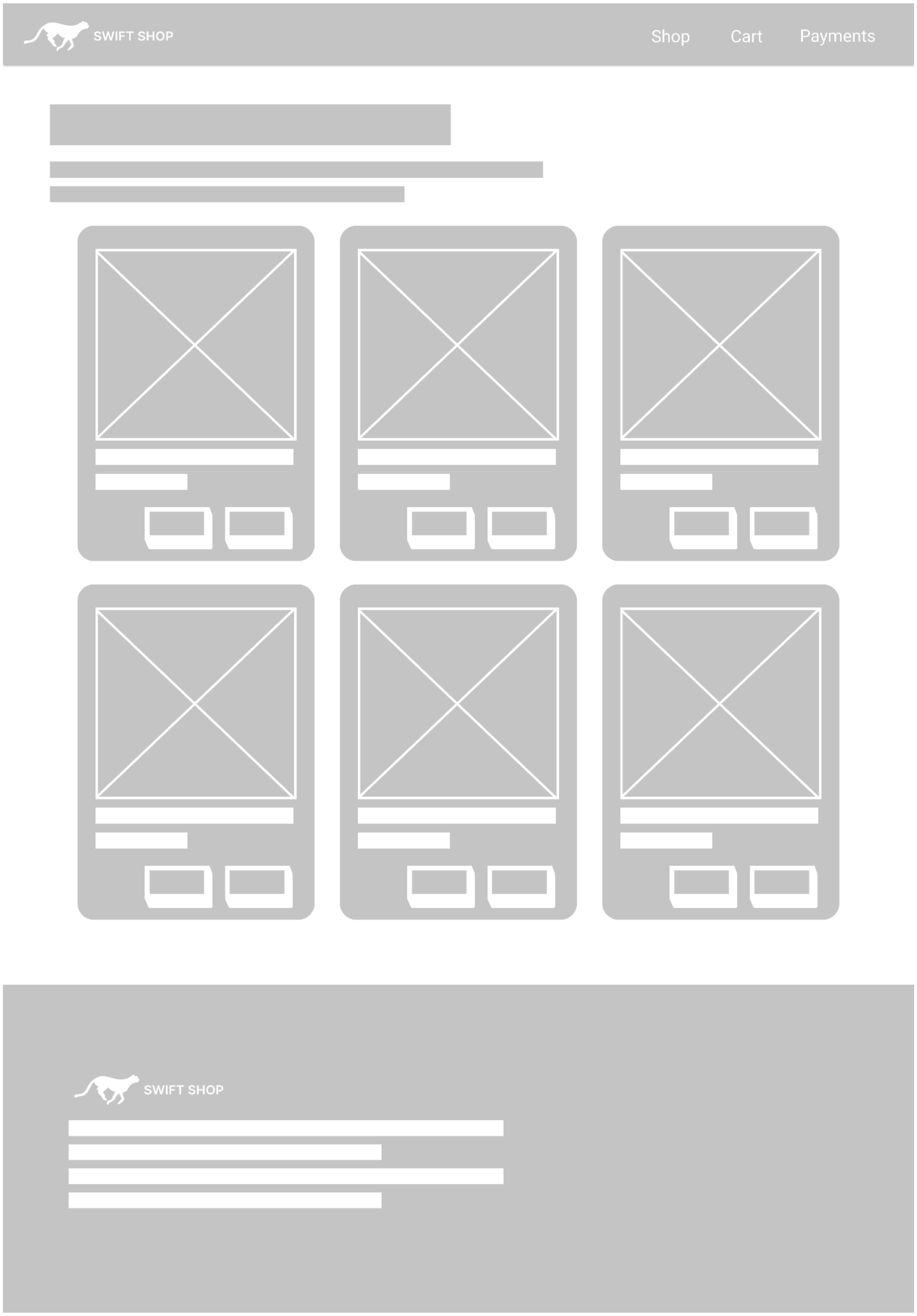
Login Page



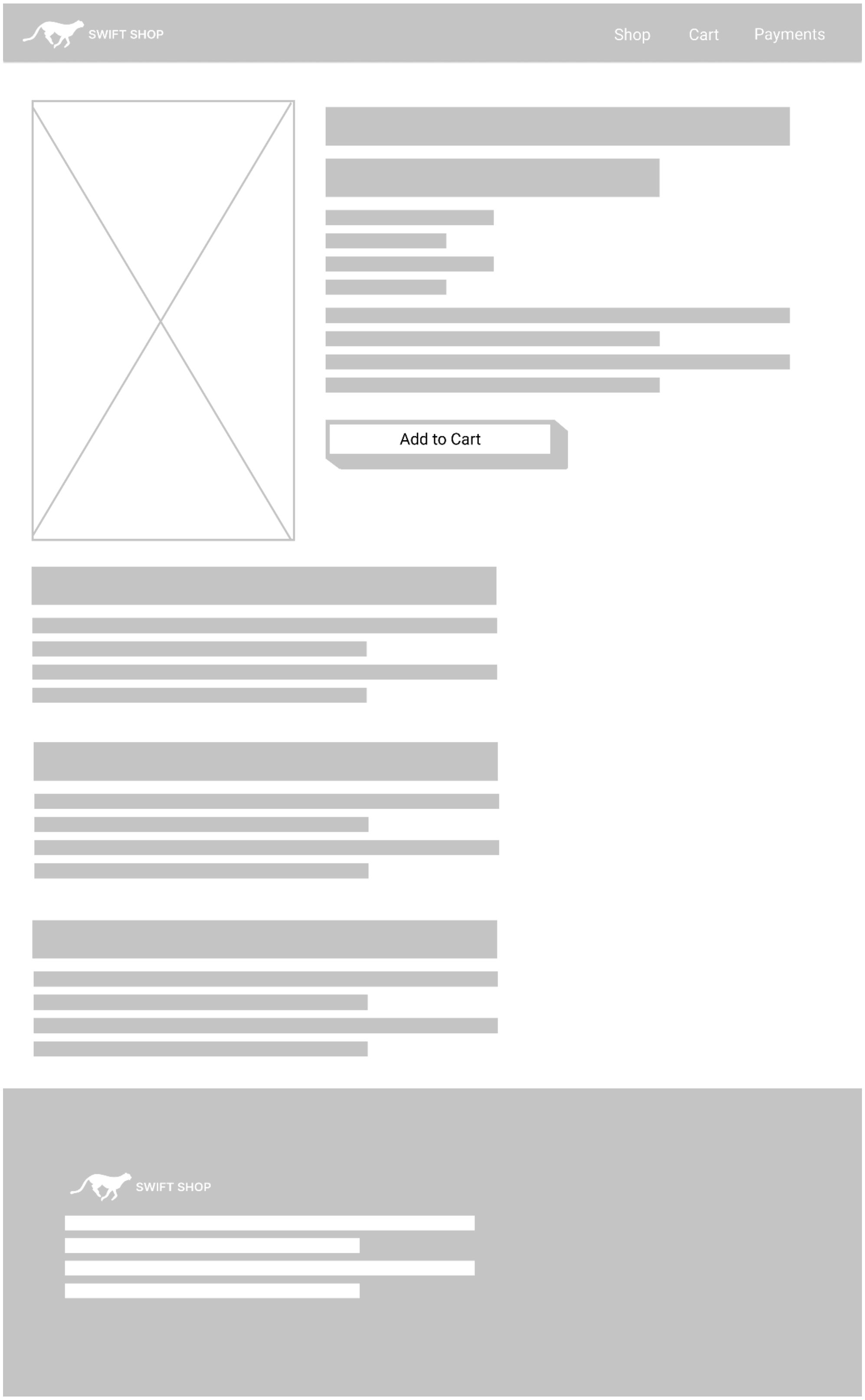
Shop Page



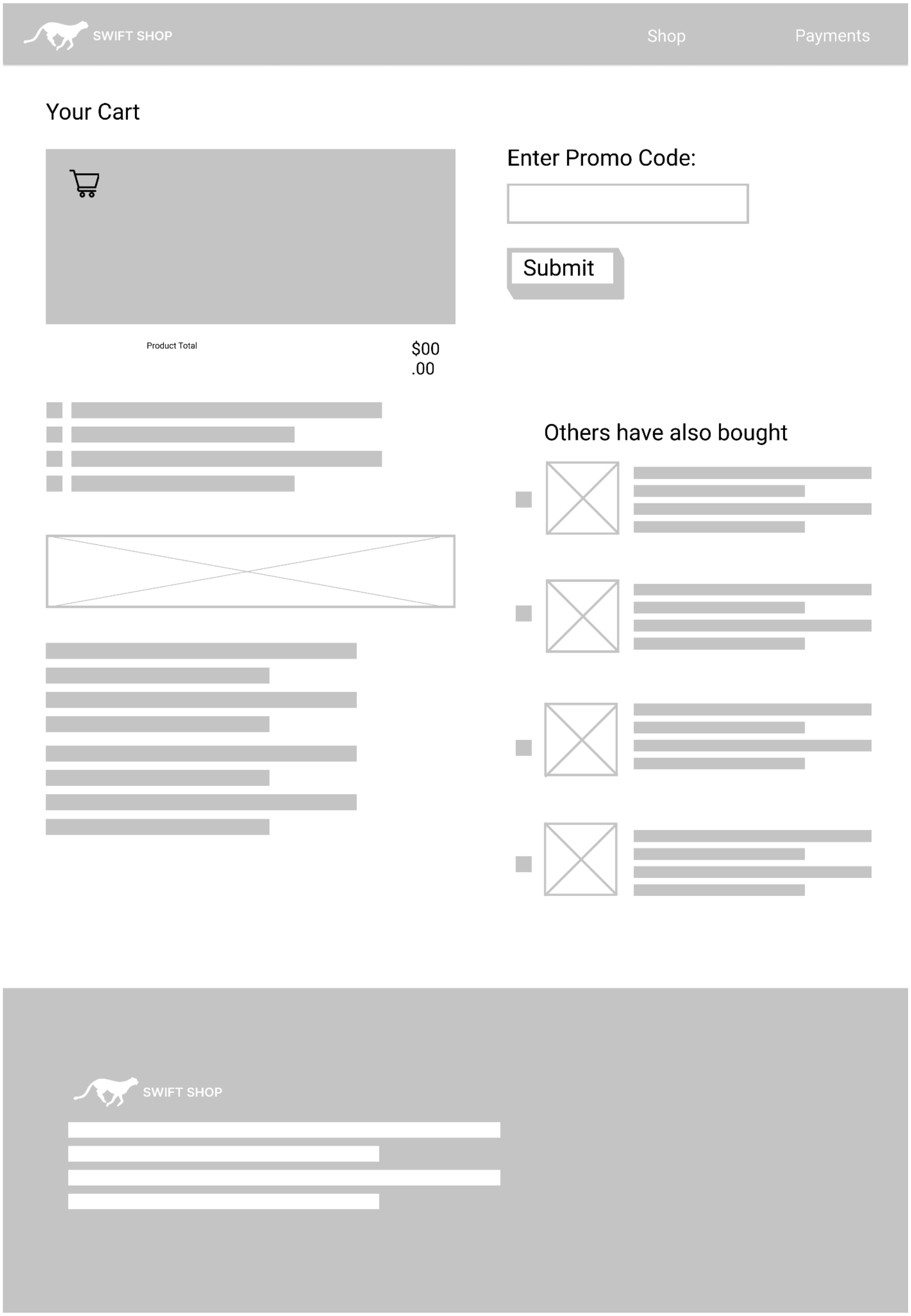
Category Page



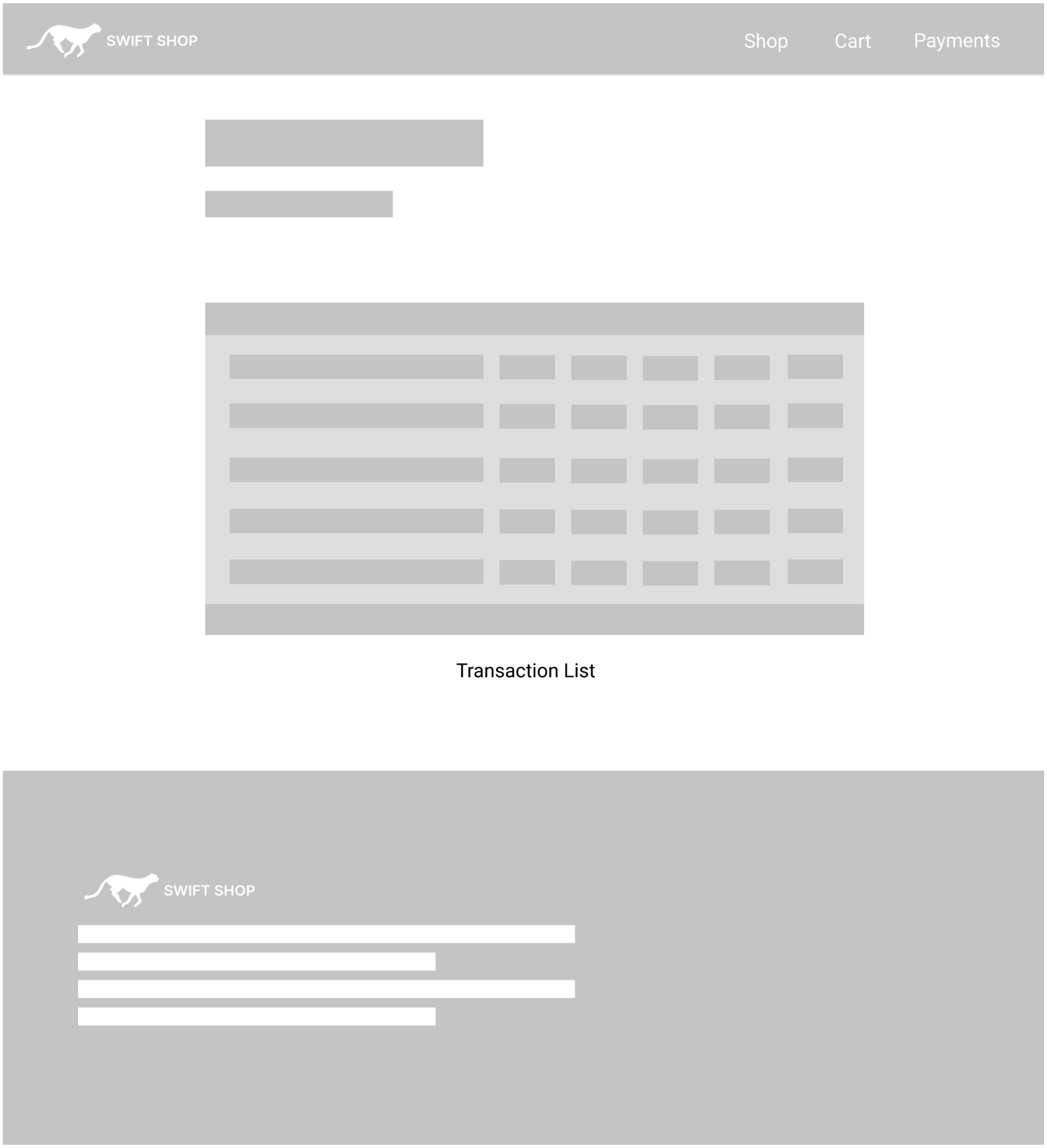
Product Page



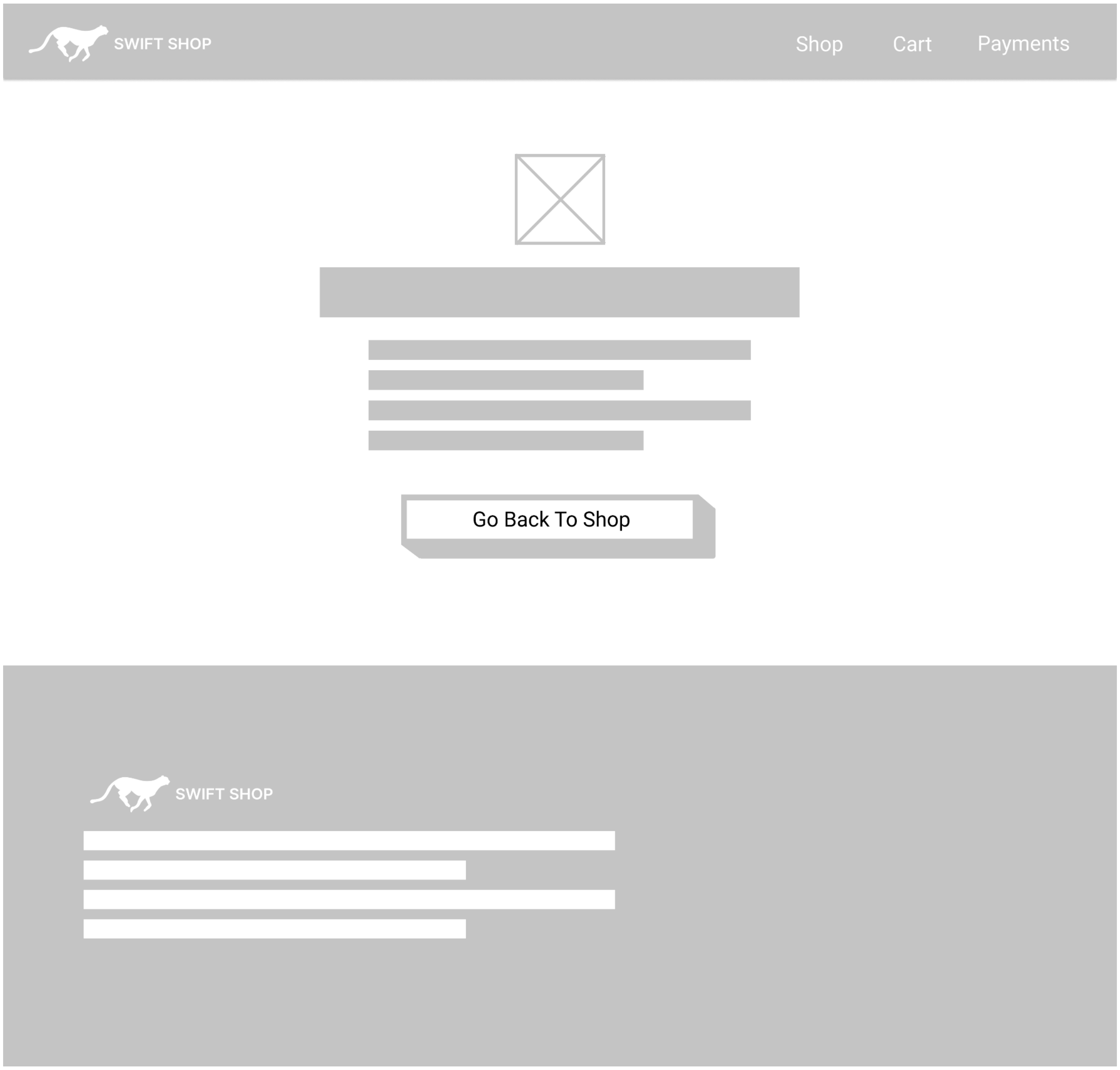
Cart Page


SWIFT SHOP
Shop
Payments
Your Cart
Enter Promo Code:
Submit
Product Total
$00
.00
Others have also bought
SWIFT SHOP

Payments Page

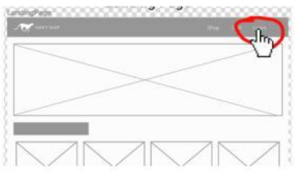


Successful Page



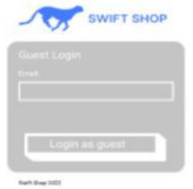
Prototype Element (Storyboard)

1.



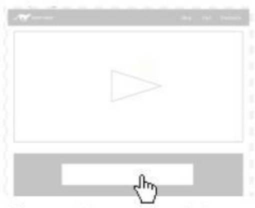
Customer clicks on the login button

2.



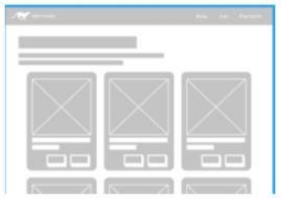
Customer is prompted to the login page where they will enter their email

3.



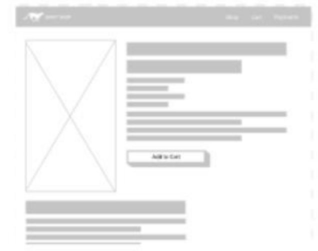
After entering a valid email, the customer is prompted to the shop page

4.



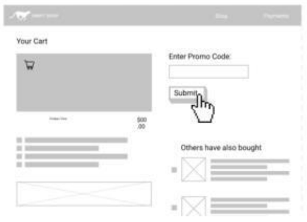
When clicking on the category button, the user can find the products they are looking for sorted by the category group

5.



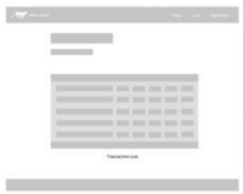
After selecting the product of their choice, the customer can click on the item to bring it to the product page.

6.



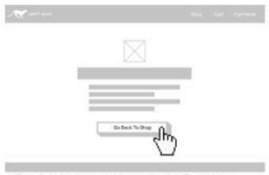
Once the item(s) are added to the cart, the customer may enter a valid promo code if applicable and can review their cart before payment

7.



Once cart is reviewed, the customer may proceed to the payment method. They will be prompted with our payment API stripe where they will enter their payment informnation

8.

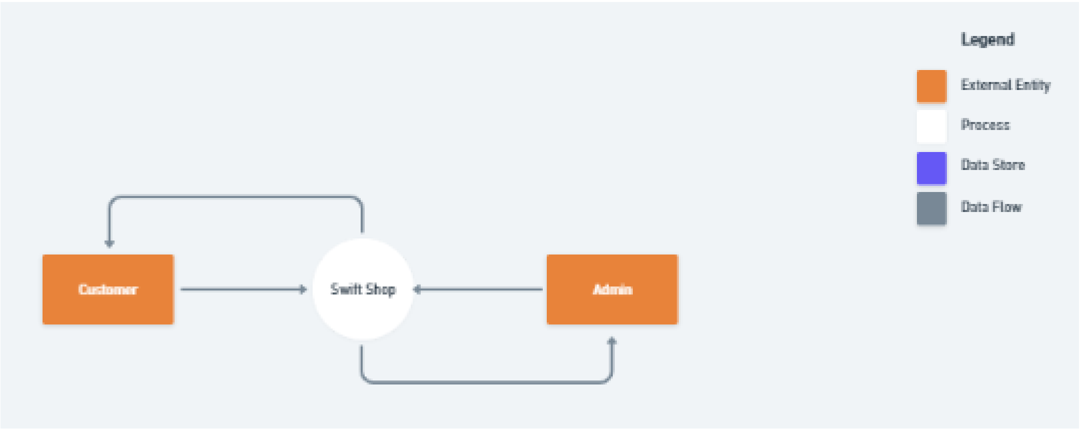


Once payment is sucessful, the customer will be prompted with a success page that will have their order confrimation and infromation, they will have a choice if they want to go back to the shop page with the back to shop button

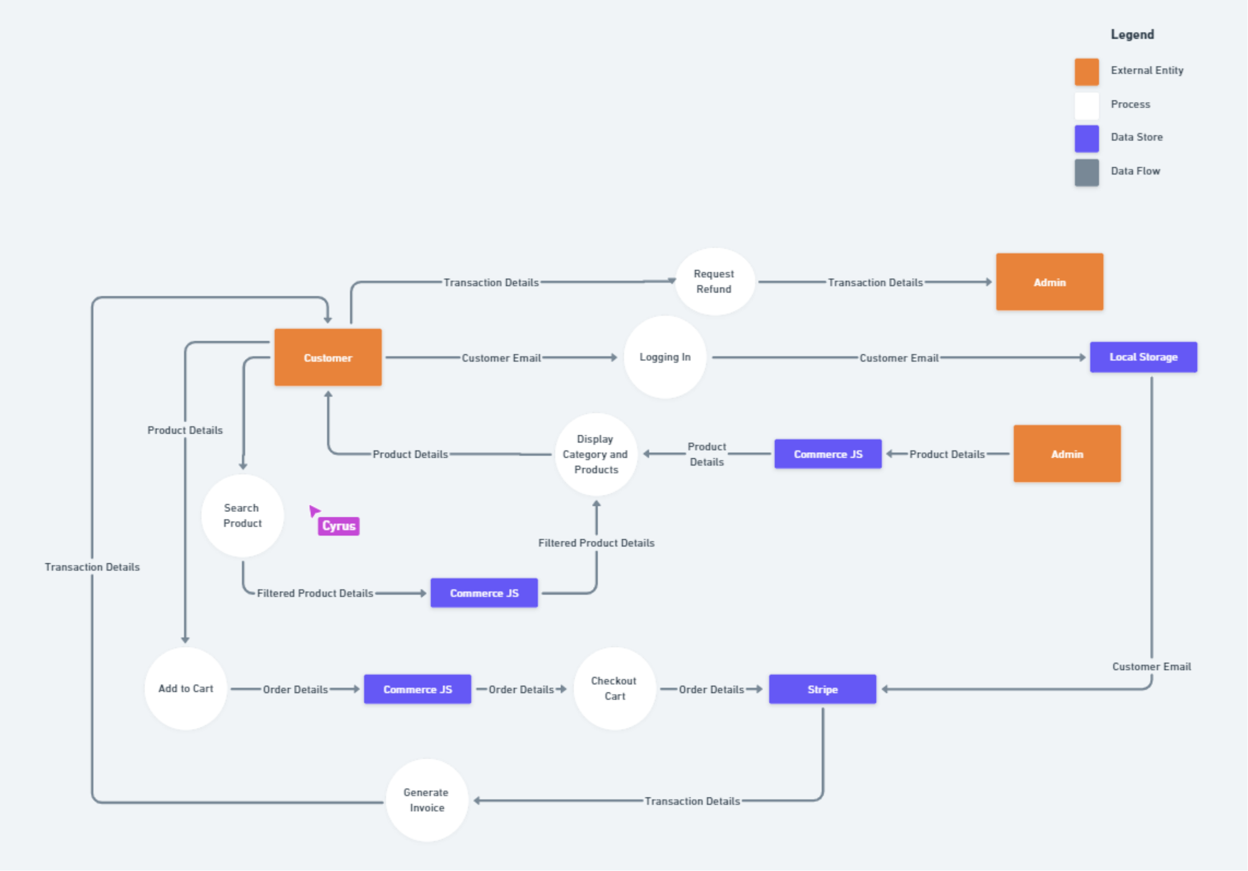
Data Flow

We are using Yourdon and Coad notation for our data flow diagrams.

Level 0 - High Level



Level 1


Legend
External Entity
Process
Data Store
Data Flow
Transaction Details
Request Refund
Transaction Details
Admin
Customer
Customer Email
Logging In
Customer Email
Local Storage
Product Details
Product Details
Display Category and Products
Product Details
Commerce JS
Product Details
Admin
Search Product
Cyrus
Filtered Product Details
Transaction Details
Filtered Product Details
Commerce JS
Customer Email
Add to Cart
Order Details
Commerce JS
Order Details
Checkout Cart
Order Details
Stripe
Generate Invoice
Transaction Details