



Design Document

Abhi Sharma
Arjun Harbhajanka
Bharat Sachdev
Mihir Abhyankar
Sanjith Pabbisetty
Shivam Bairoliya

Team 5

INDEX

● Purpose	3
○ Function Requirements	
○ Non-Functional Requirements	
● Design Outline	7
○ Description of the Components	
○ High-Level Overview	
○ Activity/State Diagram	
● Design Issues	12
○ Functional issues	
○ Non-Function issues	
● Design Details	15
○ Data Class Level - Design	
○ Description of Data Classes and their Interactions	
○ Sequence Diagram	
○ UI Mockup	

PURPOSE:

A post-Covid World has brought a lot of changes to how the world functions. One change is the exponential growth of the eCommerce segment of the market. So, every seller is trying to make his presence seen on the internet, either through a website or an app. While eCommerce website building and deployment has gotten significantly simpler over time through services such as Shopify, Squarespace, and Wix, this is not the case for mobile apps. Mobile app development remains extremely restrictive even though the demand for apps increases exponentially year after year and building an eCommerce app poses even more problems than a static mobile app.

SimpleSell provides a simple way for people to kick start their business on a mobile app with minimal hassle by building a template-based mobile application that can be modified for the businesses' needs. Our website will allow sellers to create an account and modify the template(s) available to suit their needs. Once they are happy with how the app looks, they can download an apk file of their app and deploy it to the android store. The app and website are synced to the same database to allow sellers to manage their inventory on the website and see changes in the app in real-time.

In other words, a seller gets to make his presence in the eCommerce sphere of the market through SimpleSell. SimpleSell will be the one-stop for every seller who wants to sell its products through its app and doesn't want to get clutched in the hands of different freelancers.

Functional Requirements:

1. Seller Account:

As a seller,

- a. I would like to be able to register for a SimpleSell account
- b. I would like to be able to log in using a username and password.
- c. I would like to modify my SimpleSell account details.
- d. I would like my password to be reset if I forget it.
- e. I would like Security Questions to make my account more secure.
- f. I would like to add my description.
- g. I would like to display my socials.

2. Seller Management:

As a seller,

- a. I would like to be able to add products to the inventory.
- b. I would like to be able to delete products from the inventory.
- c. I would like to add pictures to the products.
- d. I would like to be able to accept orders.
- e. I would like to be able to add product descriptions.
- f. I would like to mark products as sold out.
- g. I would like to give discounts on the marked price.
- h. I would like to be able to make custom discount codes.
- i. I would like to create collections of products.
- j. I would like to create tags for different collections.
- k. I would like to categorize each product.
- l. I would like to be able to change the price of items in my inventory.
- m. I would like to give products an SKU (Stock Keeping Unit) number.
- n. I would like to choose between a grid or list view for my product page.
- o. I would like to be able to specify subcategories of the same products.
- p. I would like to like to set the app icon.
- q. I would like to be able to deny or accept cancellation/return requests.

3. Seller Analysis:

As a seller,

- a. I would like to see total sales.
- b. I would like to see products sold by the unit.
- c. I would like to see the average order value.
- d. I would like to see the total number of orders.
- e. I would like to see the return customer rate.

- f. I would like to be able to view orders per customer.
- g. I would like to have the contact information of my customers.

4. Buyer Account:

As a Buyer,

- a. I would like to be able to create an account on the seller's app.
- b. I would like to be able to reset my login details if I forget them.
- c. I would like to be able to look at my previous orders and purchases.
- d. I would like to be able to request a cancellation of my purchases.
- e. I would like to be able to provide a reason for my cancellation request.
- f. I would like to get notified when the seller accepts/denies my order.
- g. I would like to add addresses to my address book.
- h. I would like to view my purchase history.
- i. I would like to delete addresses from my address book.

5. Buyer Usability:

As a Buyer,

- a. I would like to be able to order things from the seller's app.
- b. I would like to be able to use multiple payment interfaces.
- c. I would like to be able to search for specific items.
- d. I would like to be able to see the pricing for the product on the search page.
- e. I would like to keep track of products that I am interested in.
- f. I would like to be able to apply discount codes.
- g. I would like to view my cart.
- h. I would like to review my order before checking out.
- i. I would like to have access to the contact information of the seller.

6. Developer Management:

As a Developer,

- a. I would like to be able to update and manage the user database.
- b. I would like to be able to add or remove users.

7. Appearance Requirements:

As a Developer,

- a. I would like to be able to customize and make changes to the available templates.
- b. I would like to potentially add more templates. As a developer, I would like to add more templates.

Non- Functional Requirements:

1. Performance:

- a. We would like the application to run smoothly without crashing.
- b. We would like the application to start within 30 seconds.
- c. We would like the website to run smoothly with lagging.
- d. We would like to develop my frontend and backend independently.
- e. We would like to maintain a minimum of 5 sellers on the website
- f. We would like to maintain a minimum of 5 buyers per seller

2. Server:

- a. We would like to have a server that can support real-time communication.
- b. We would like to have a server that stores the application data in a database.

3. Appearance:

- a. (If time allows) We would like to create a website with a pleasing and unique interface.
- b. (If time allows) We would like to create an application with a unique, pleasing interface.

4. Security:

- a. We will not collect any user location data.
- b. We would like to store our data on a secure database.
- c. We would like to verify user input to prevent SQL injections.

5. Usability:

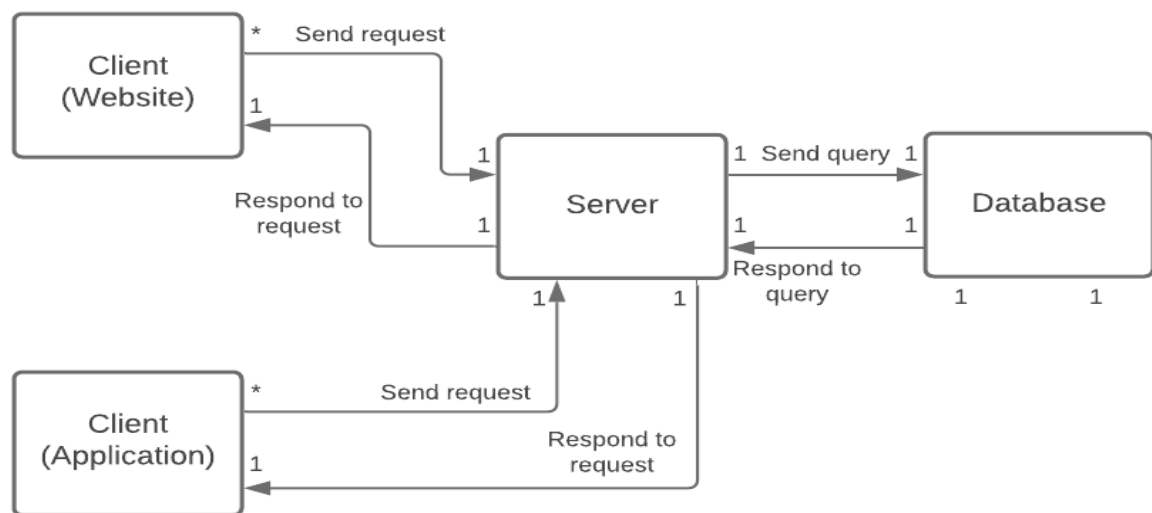
- a. The platform interface should be easy to navigate and intuitive from the user's perspective.
- b. The website should be designed with the user's ease and convenience in mind so that it can be navigated with ease.
- c. We would like the application to be friendly and easy to use.

6. Hosting and Deployment:

- a. We would like to host the front end on Github.
- b. We would like to host the backend on AWS.
- c. We would like to store our database using MySQL.
- d. We would like to update and deploy the frontend and backend separately.

DESIGN OUTLINE:

There are two major communication channels in this project: we have the seller communicating with the database through our website and we have the buyers communicating with the database through our android app. Essentially, we have two platforms: the mobile end and the website end, one for the buyers and one for the sellers.



Description of the Components:

1. Client (Website):

- a. The website client provides sellers with an interface to layout their app design.
- b. The website client sends HTTP requests to the server.
- c. The website client receives updates from the server and updates the interface accordingly.

2. Client (Application):

- a. The android application provides buyers with an interface to purchase.
- b. The application sends HTTP requests to the server with information regarding orders and purchases.
- c. The android client receives updates and responses from the server.

3. Server:

- a. The server receives HTTP requests from the website and application client.
- b. The server validates requests from both the client and sends queries to the database about the requests.
- c. The server generates responses and responds to the clients' requests.

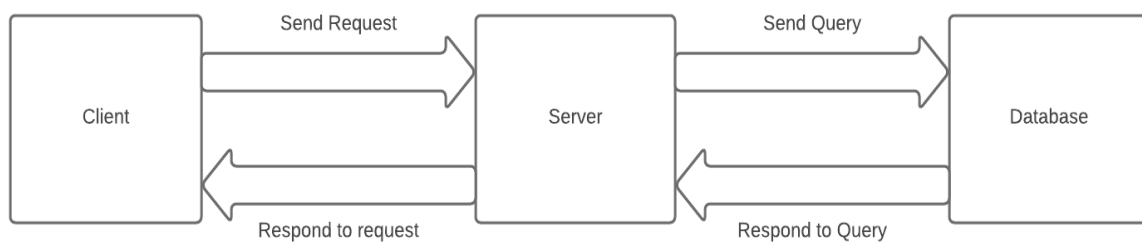
4. Database:

- a. The database stores buyer, seller, item, and order information
- b. The database receives queries from the server and responds with the appropriate data.

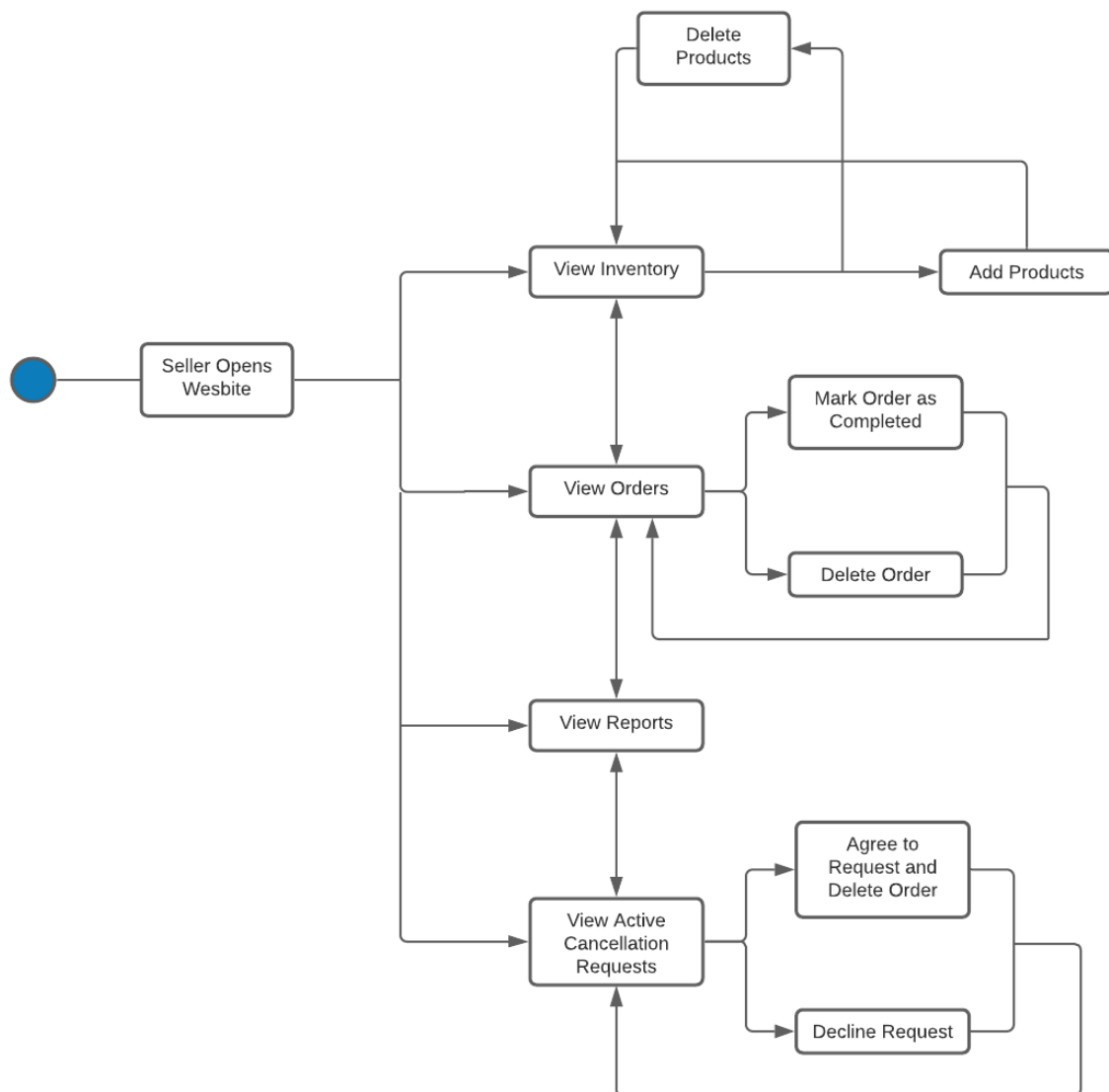
High-Level Overview

This project will involve a website, which allows the sellers to design an apk. This project will use a client-server model in which the server simultaneously handles access from multiple clients(sellers) using Java.

The client will first send requests to the server which need to be accepted by the server. Depending on the request, the server will either store some data in the database or extract some data from the database. And finally, it responds to the client accordingly.

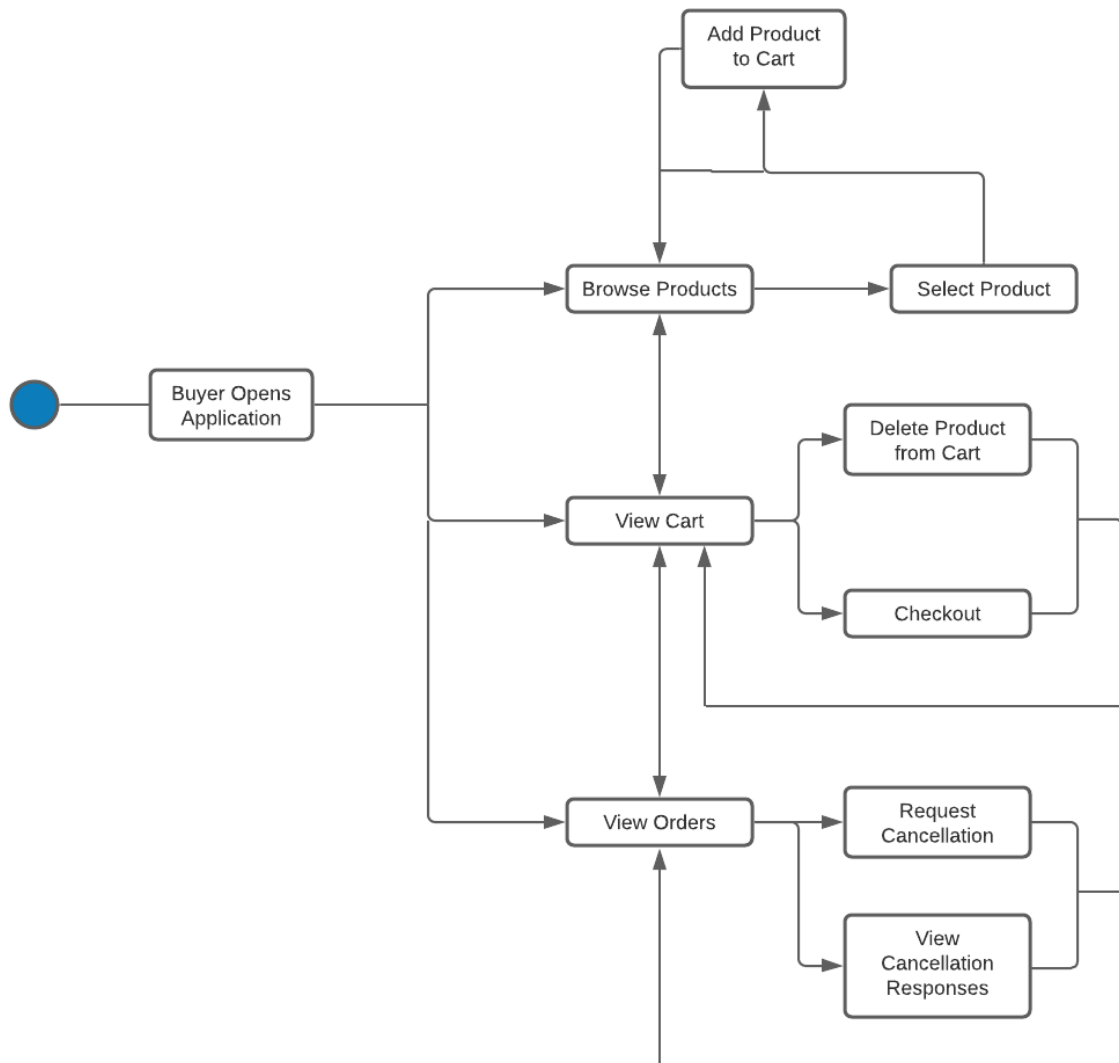


Activity/State Diagram For The Client(Website)



The above diagram shows the different things a seller can do once he opens his seller interface on the website. He can view his inventory and add or delete his products accordingly. At the same time, he can also view all his orders and can even mark them as completed. The seller can also view his financial report using this interface which will show the total sales, average order value, etc. Finally, the seller can also handle the cancellation requests by either declining or approving these requests.

Activity/State Diagram For The Client(Application)



The above diagram shows the different things a buyer can do once he opens an app. The buyer can start by viewing the different products and adding whatever he likes to add to the cart. Then he can view the cart and even make changes to the cart concerning removing items in the cart or changing the quantity of the item. Once he finalizes the cart, he can checkout and place the order.

DESIGN ISSUES:

Functional Issues:

1. What information is required for signing up for an account:

- a. Username and password
- b. Username, password, and email address.
- c. Username, password, email address, and phone number

Choice: (b)

Justification: To implement our forgot-password feature we will need to send a message to the user. This is why we will need the user's email address at the time of signing up. We want to make signing up for SimpleSell very easy and require minimum information. Any kind of SMS usage will incur charges which we want to avoid.

2. How can a buyer return a product:

- a. SimpleSell offers a feature that enables product returns.
- b. Returning a product is handled between the buyer and seller.

Choice: (b)

Justification: Reasons for returning a product can vary according to the product and SimpleSell cannot decide the terms of returning a product sold by an independent business. These details are entirely up to the seller and we think returns should be handled by the buyer contacting the seller.

3. Should we verify the seller's identity:

- a. Yes as it will establish a sense of trust between the buyer and the seller
- b. No, verifying the seller has a lot of complications.

Choice: (b)

Justification: Verifying the identity of the seller comes with a lot of limitations. The APIs available for identifying people using SSN require a lot of verification of the person integrating the API due to the guidelines implemented by the DHS. At the same time, the verification will be limited to people with an SSN which means this would not allow sellers from other countries to use this platform.

4. Should we integrate marketing tools with our platform?

- a. Yes, as it will provide sellers a way to market their app on different social media platforms.
- b. No, as marketing tools for an app are quite cumbersome to implement.

Choice: (b)

Justification: The way these marketing tools work are that they provide ads for the already published apps on the Play Store. So, to implement these marketing tools, we need to connect the Play store account with the SimpleSell account which is quite cumbersome to implement due to security reasons.

Non-Functional Issues:

1. What web service should we use?

- a. AWS
- b. Azure
- c. Google Cloud
- d. Heroku

Choice: (a)

Justification: AWS is the most widely used and the most sophisticated cloud computing platform that offers SaaS, PaaS, and IaaS services. At the same time, integrating AWS services with our other technology stack seemed to be the easiest and most efficient way. Most importantly, most of our team has already been exposed to AWS in the past. So, navigating through AWS services will be way easier for our team than any other cloud computing service.

2. What database should we use?

- a. MySQL
- b. MongoDB
- c. Oracle XE
- d. Redis

Choice: (a)

Justification: MySQL is a very secure and reliable database management system that allows even the most demanding applications to run while ensuring optimum speed, full-text indexes, and unique memory caches for enhanced performance. It is also a commonly used database and therefore there are many guides/tutorials available on the internet. MySQL is also free and open source.

3. What backend language/framework do we use?

- a. Java
- b. PHP
- c. Node.js(JavaScript)
- d. Django

Choice: Java

Justification: Since our project involves Android Development, Java seemed to be the obvious way to go. At the same time, our whole backend team is most efficient in Java which gives us all the more reason to choose Java.

4. How is the quantity of an item stored?

- a. Store the quantity of an item as an attribute in the item class.
- b. Make copies of the item as per the quantity in the item list.

Choice: (a)

Justification: We would like to make our data structures as small as possible to get the best performance. Making copies of the item as per the requested quantity in the item list will waste a lot of space and saving this information as an attribute will be the better option that suits our needs. Although we increase the size of the individual data structure, the overall number of data structures required is reduced.

5. What frontend language/framework should we use?

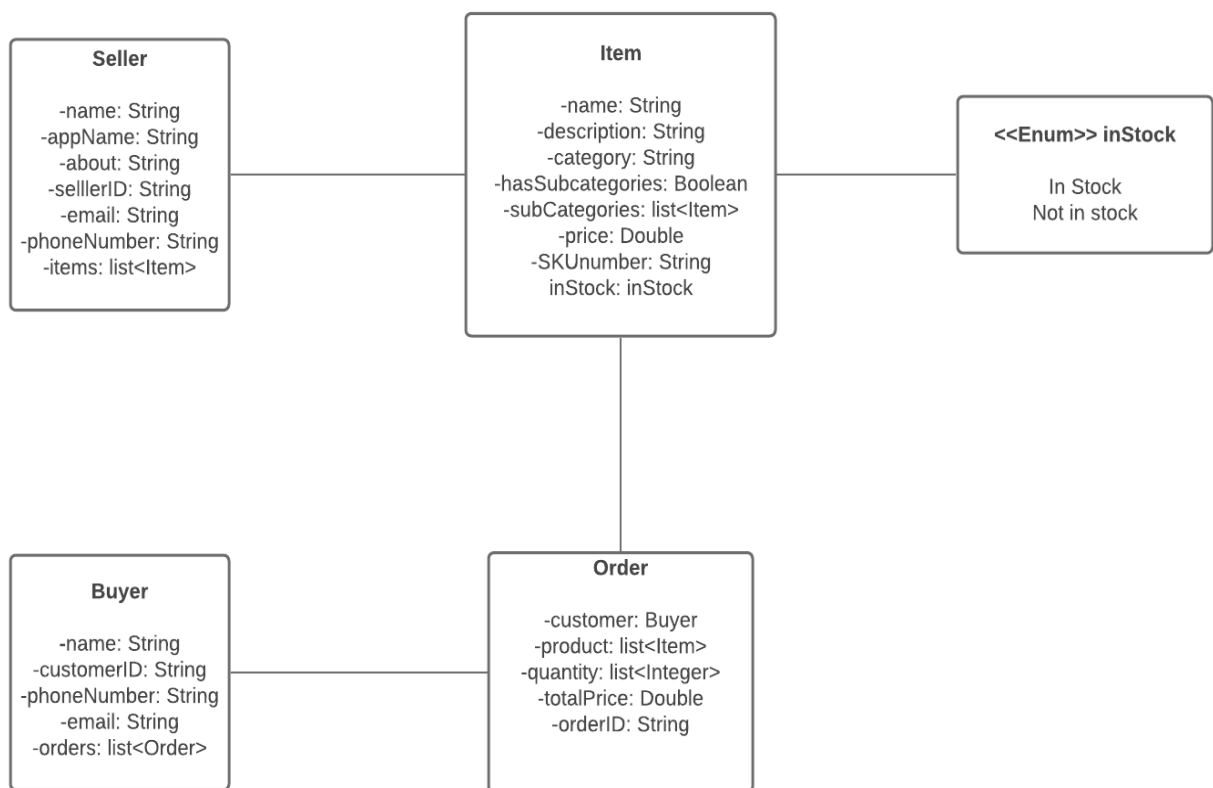
- a. HTML, CSS, Javascript
- b. TypeScript
- c. SASS
- d. React
- e. AngularJS

Choice: (a)

Justification: After careful consideration, the members of our team working on the front end are most familiar with HTML, CSS, and JavaScript. Nevertheless, these languages go very well together and we believe we can make an elegant web page without the hassle of learning new technologies such as SASS and TypeScript. We also found that there are a lot more resources that will be useful to us based on making a retail app in HTML, CSS, and JavaScript.

DESIGN DETAILS:

Data Class Level - Design



Description of Data Classes and their Interactions:

Our class design is carefully crafted to suit our needs for the project. The instances of our classes will be stored in a Relational Database Management System. We plan to use the MySQL backend as our database. Our Java client will use the following classes and the classes will be passed over as JSON data via HTTP requests. The class descriptions may change in the long run. They can be adjusted to suit the needs of the program and the customer.

- **Seller**

- Represents a seller
- It stores the data about the seller including their name, sellerID, email, and phone number
- It also stores the data about the seller. A seller description that represents data about him.
- It stores a list of Items which is of type of item. These are items that the seller is selling.

- **Item**

- Represents a single item that is bought by a buyer or sold by a seller.
- Stores all details about the item including the name of the item and its description
- It stores the category the item belongs to as a string.
- The item object also contains an option to have subcategories. If the instance has subcategories it stores a list of type Items that are its subcategories.
- It stores the price as a Double variable while also storing a unique identification number for each item.
- It is linked with an Enumeration that checks whether the seller has the item in stock

- **inStock**

- An enumeration that keeps track of whether an item is in stock or not
- It has two states In Stock and Not in Stock

- **Order**

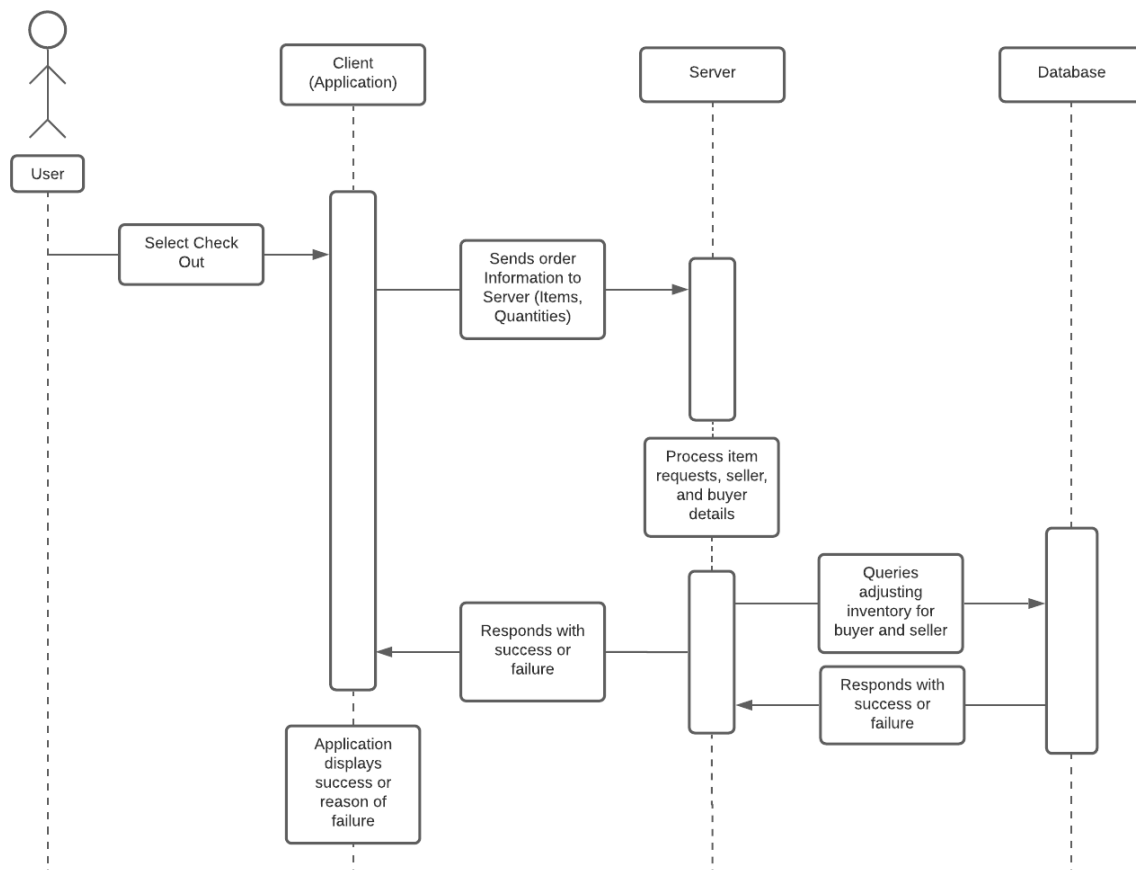
- A class that stores the details of an order placed by the buyer for a particular product or products sold by the seller
- It stores a product list and quantity list. The product list represents the product or products for which the order has been placed
- The quantity list stores the quantity for each product in the list corresponding to the index of the products in the list
- It also stores the price as a Double variable and a unique order ID for every cart order placed

- **Buyer**

- The buyer class represents a Buyer who buys the sellers products
- It stores a unique customerID for every buyer
- It also stores the contact details of the customer including phone number and email
- There is also a list attached that keeps tracks of all the orders made by the customer

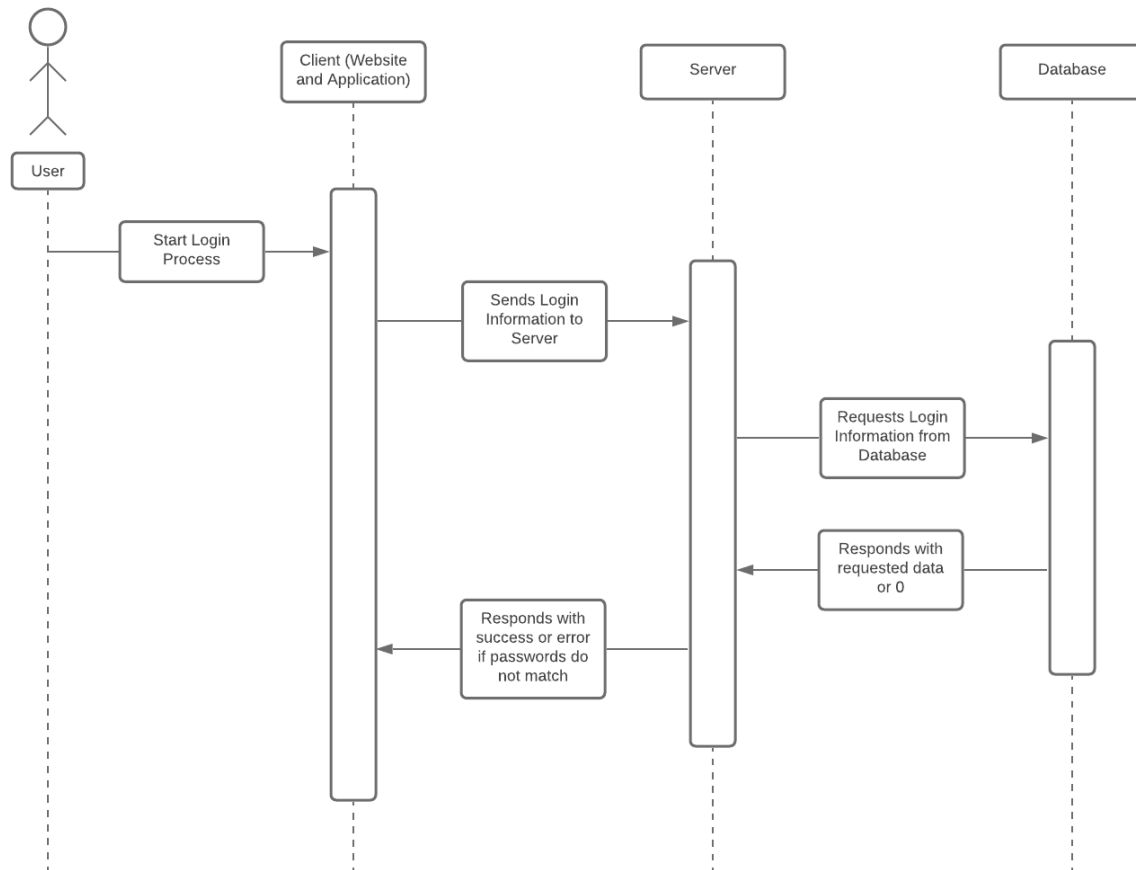
Sequence Diagrams:

Check-Out Process:



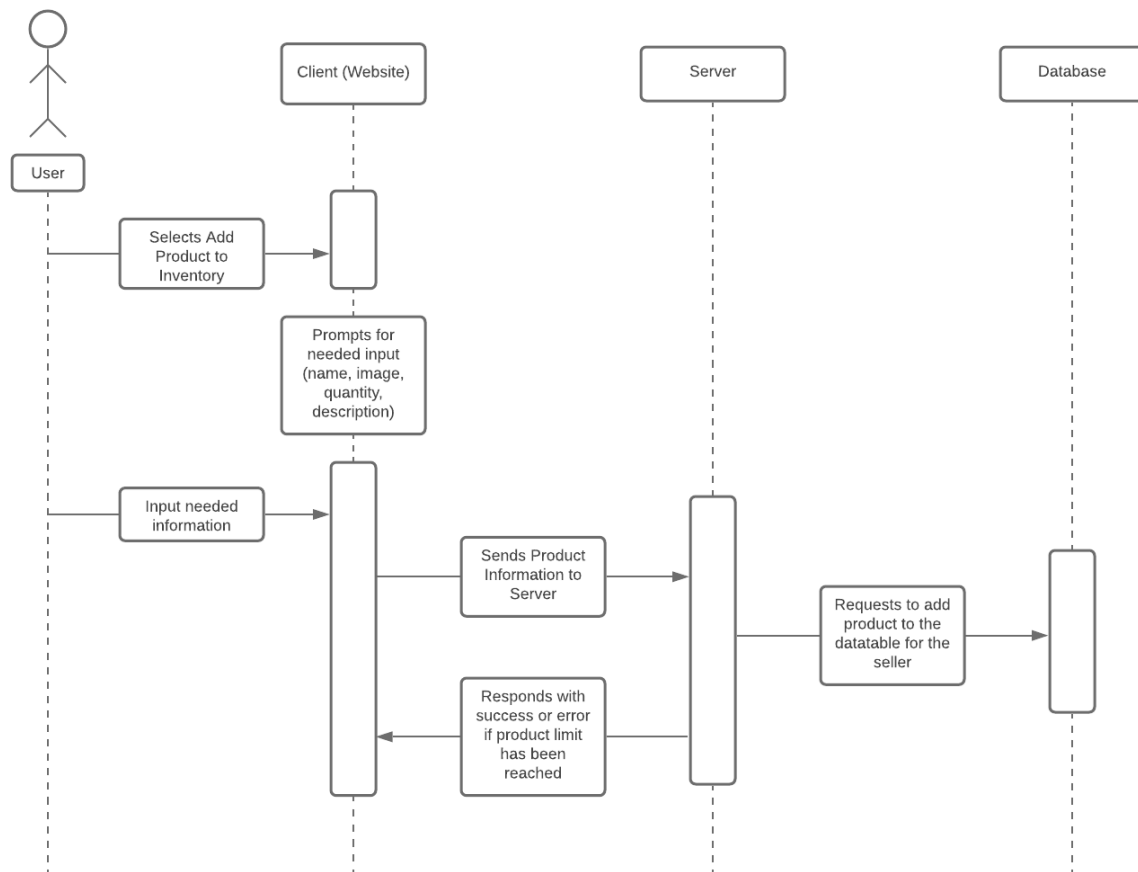
The above diagram shows the sequence of events that take place when an order is placed on the app. So, once the buyer presses the place order button, the server receives a request containing the order information. The server then processes the request by retrieving information from the database. Once the data is processed, the client is sent a success or failure response depending on whether the items are in stock or not.

User Login Process:



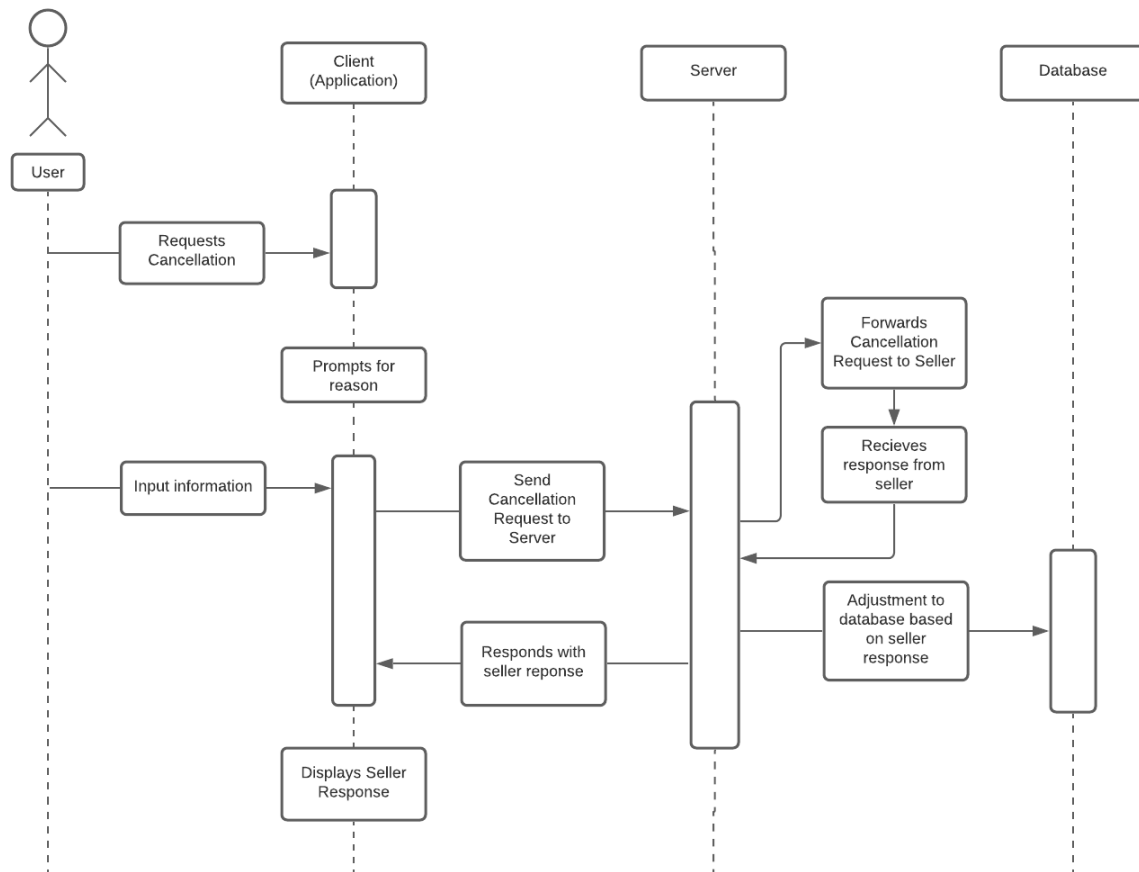
The above diagram shows the process of logging in. The user becomes the client by starting the login process. The client sends login information to the server. Based on this information the server will query the database. Once the query is complete, the database responds to the server with the requested information. The server will respond to the client with a success or a failure. For a successful login, the password entered by the user should be correct.

Adding item to an inventory:



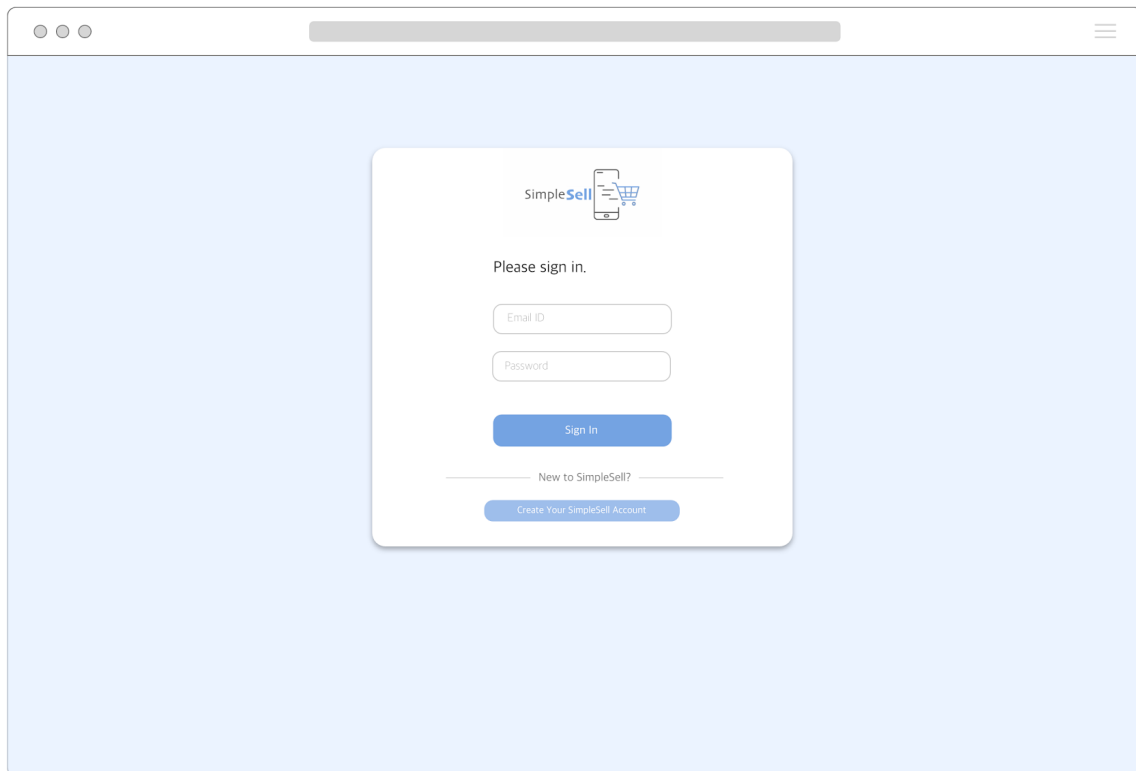
The above diagram shows the process of adding products to the user's inventory. The user becomes the client when the user selects the add to inventory button. Along with this, the user will provide other information. The client sends information such as name, input, quantity, and description to the server. The server requests the database to be added to the appropriate table associated with the seller. The server sends a success or error to the client. A failure could be caused if the product limit is reached.

Requesting a Cancellation:

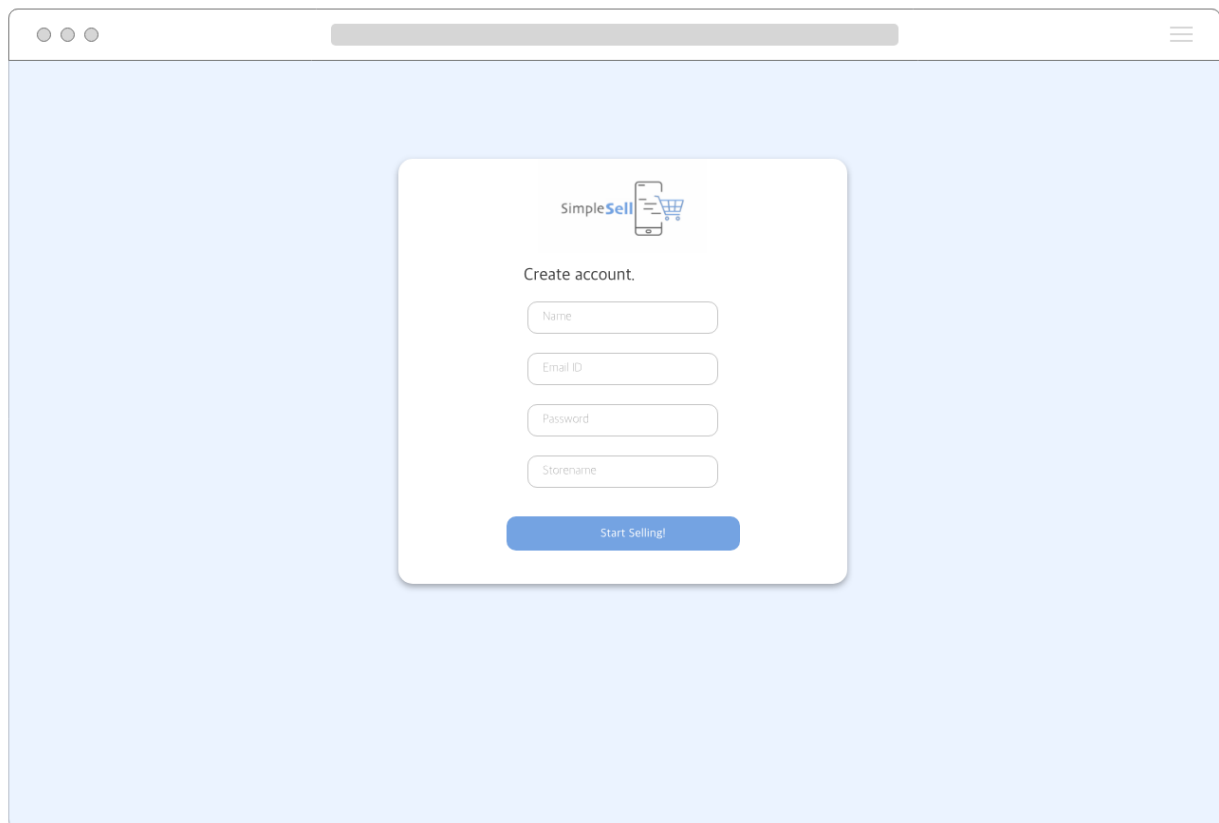


The above diagram shows the process of canceling an order. The user requests a cancellation as the client. The application will prompt the user for the reason for cancellation and the user can input information. After collecting this information the client will send a cancellation request to the server. The server will send this request to the seller the order pertains to. The seller can approve or deny the cancellation request and send this information to the server. If the seller has approved the cancellation request appropriate changes are made to the database. The server sends the seller's response to the client. The client displays the seller's response to the user.

UI Mock-Ups:



Log In



The image shows a mobile application interface for 'SimpleSell'. The app is displayed within a browser window. The background is a solid light blue. In the center, there is a white rounded rectangle containing the sign-up form. At the top of this white box is the 'SimpleSell' logo, which consists of the text 'SimpleSell' in a sans-serif font, followed by a blue icon of a smartphone with a shopping cart on its screen. Below the logo, the text 'Create account.' is displayed. There are four input fields stacked vertically, each with a light gray border and rounded corners. The labels 'Name', 'Email ID', 'Password', and 'Storename' are positioned to the left of each respective input field. At the bottom of the white box is a solid blue button with the text 'Start Selling!' in white.

SimpleSell

Create account.

Name

Email ID

Password

Storename

Start Selling!

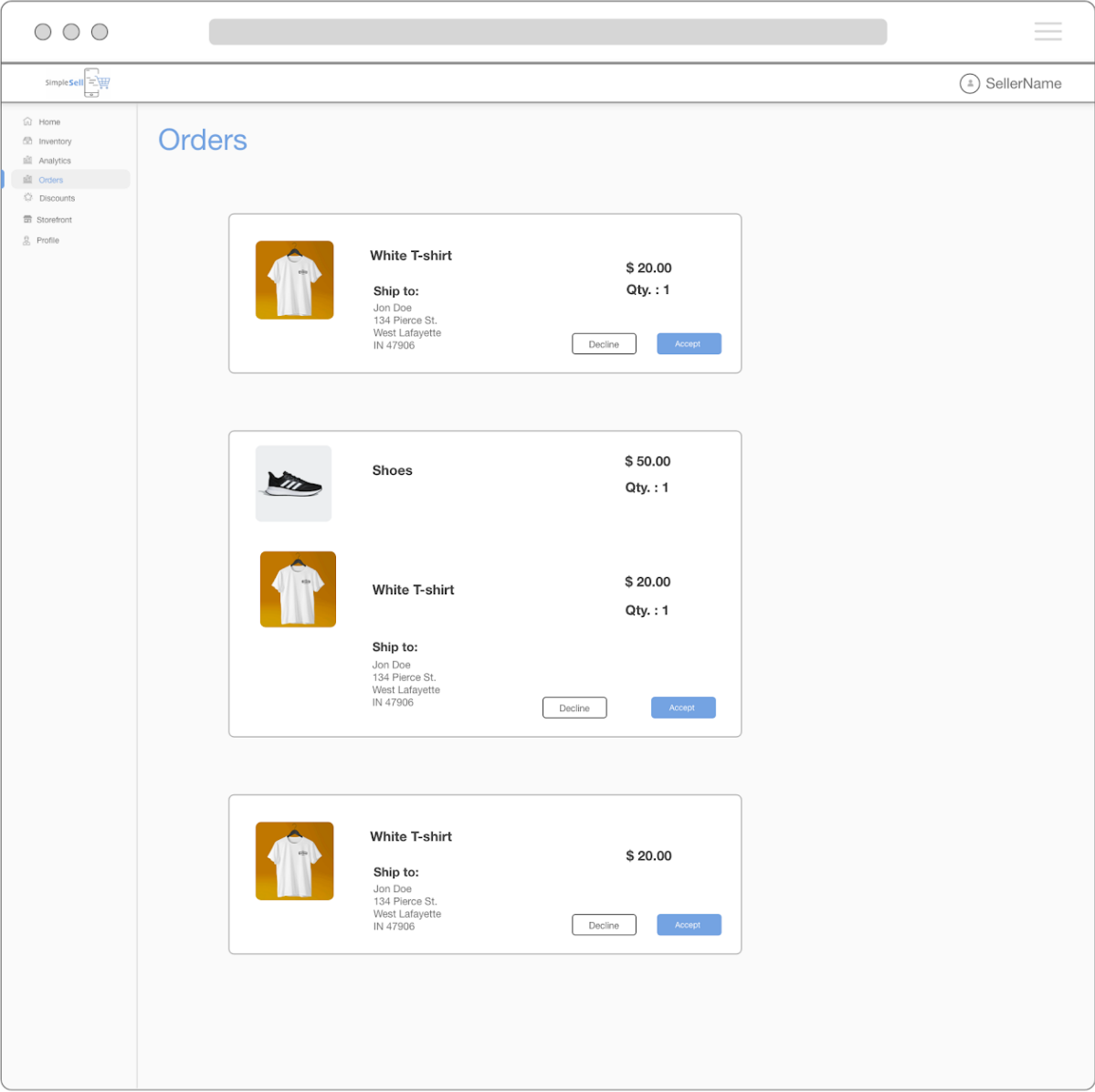
Sign Up

The screenshot shows a web application window titled 'SimpleSell' with a user profile 'SellerName'. The left sidebar contains navigation links: Home, Inventory (selected), Analytics, Discounts, Storefront, and Profile. The main content area is titled 'Add Products' and contains the following form fields:

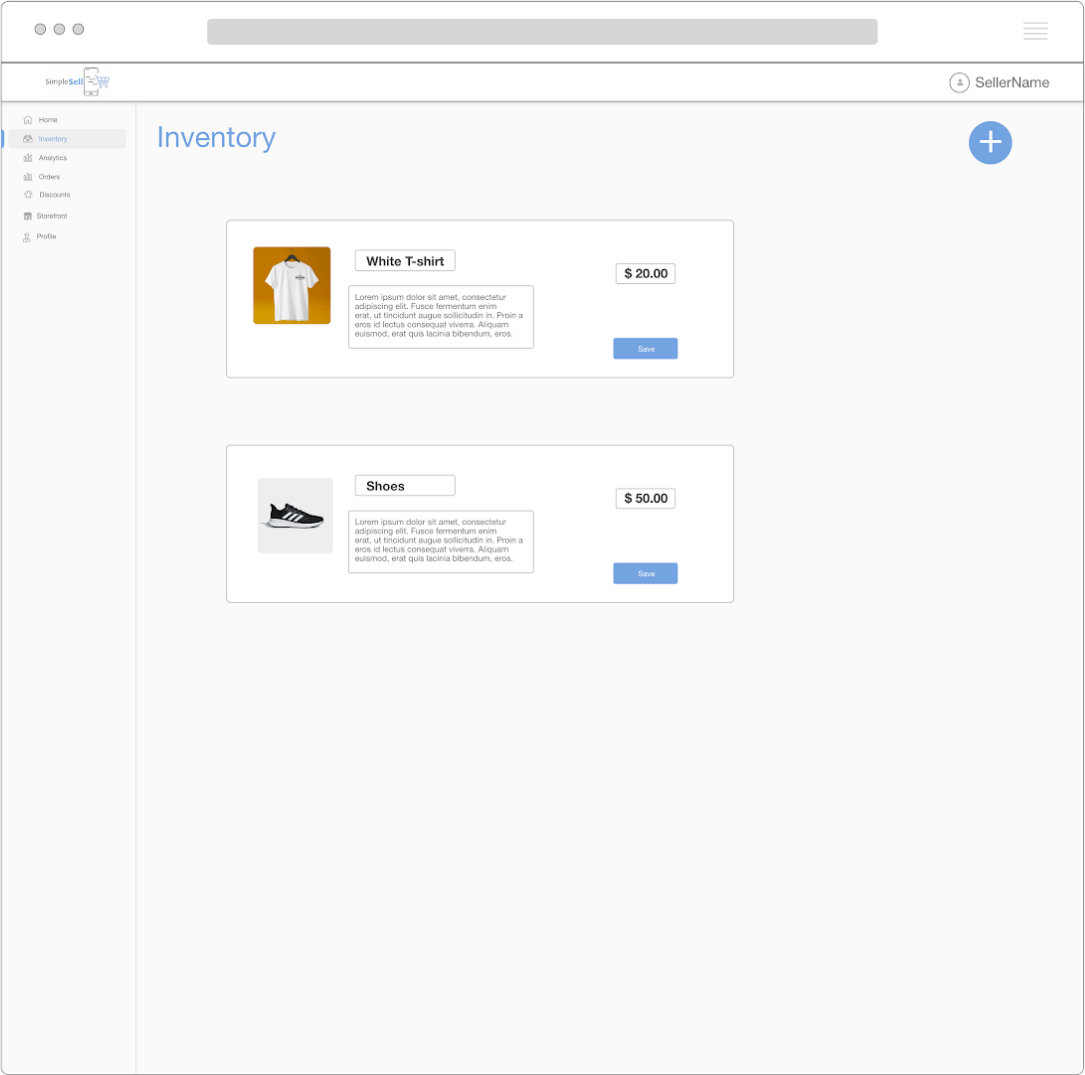
- Title:** A single-line text input field.
- Description:** A multi-line text area.
- Pictures:** A large square area with a central upload icon (an upward arrow on a document) and a small 'Add Image' button below it.
- Price:** A single-line text input field.
- Discount:** A single-line text input field.

At the bottom right of the form are two buttons: 'Discard' and 'Save'.

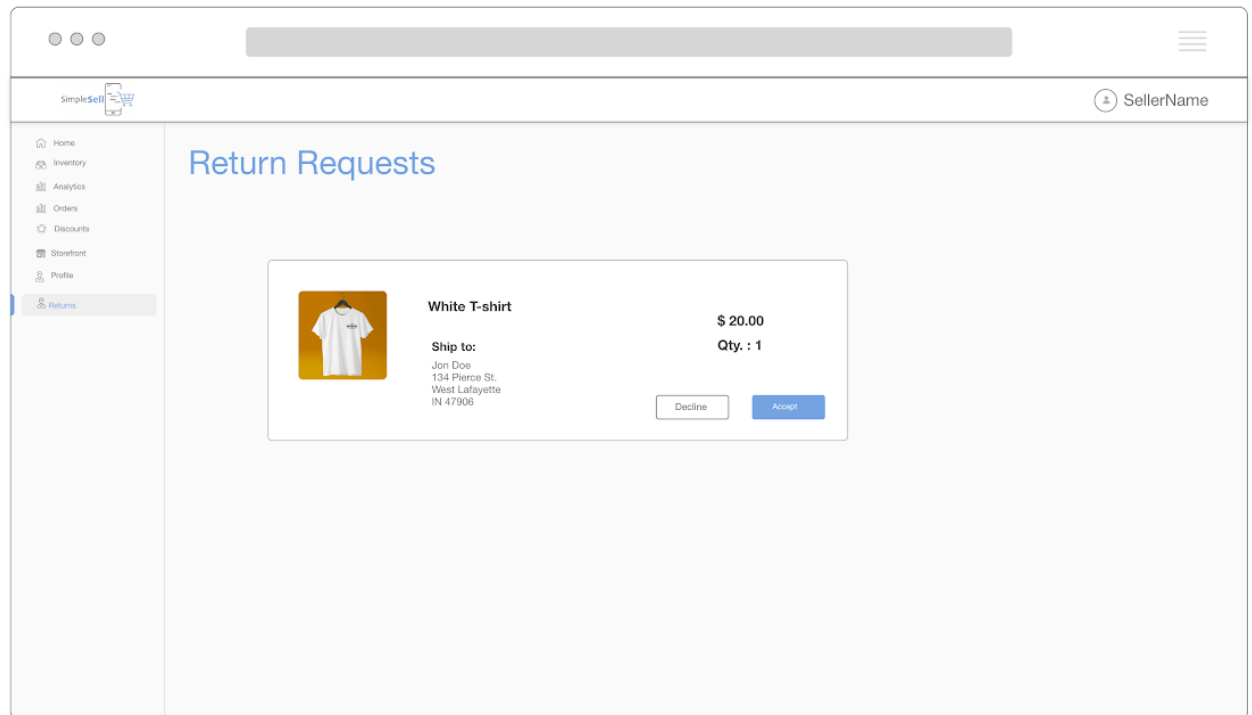
Add new Product



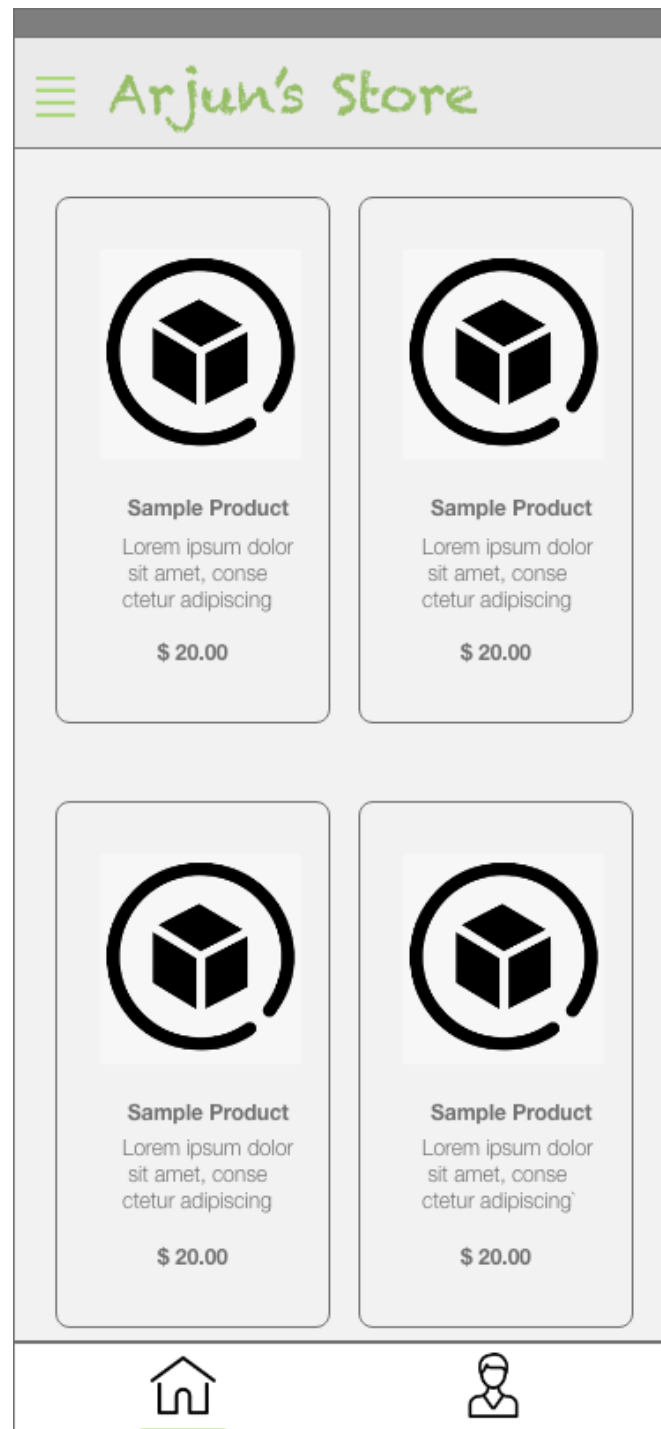
Orders



Edit Product



Approve or Decline Return Requests



Application Product View