Software Requirements Specification

for

store.it

Version 0.2

Prepared by

Group #: 3

Group Name: localghosts

Name	Roll No.	Email
Akanksha Singh	200070	akankshas20@iitk.ac.in
Antreev Singh Brar	190163	antreev@iitk.ac.in
Bhuvan Singla	180199	bhuvans@iitk.ac.in
Deepankur Kansal	180226	deepank@iitk.ac.in
Dipanshu Garg	190306	dipanshu@iitk.ac.in
Harshit Raj	200433	harshitr20@iitk.ac.in
Hitesh Anand	200449	ahitesh20@iitk.ac.in
Manas Gupta	200554	manasg20@iitk.ac.in
Priya Gole	200727	priyagole20@iitk.ac.in
Tushar	190915	tusharb@iitk.ac.in

Course: CS253A

Mentor TA: Mr. Swastik Maiti

Date: 30th January, 2022

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Akanksha Singh Antreev Singh Brar Bhuvan Singla Deepankur Kansal Dipanshu Garg Harshit Raj Hitesh Anand Manas Gupta Priya Gole Tushar	First draft	30/01/22
0.2	Akanksha Singh Antreev Singh Brar Bhuvan Singla Deepankur Kansal Dipanshu Garg Harshit Raj Hitesh Anand Manas Gupta Priya Gole Tushar	Final draft	27/04/2022

1 Introduction

1.1 Product Scope

The product, *store.it* aims to digitize various physical stores and service providers at the IIT Kanpur campus. The store owners and service providers will catalog their products and services online using the application. Any modern web browser can access the application. The application is helpful for the campus residents and students to place orders online. The store owners would manage the orders, inventory and gain insights into their sales.

The product's main goals are as follows:

- to provide improved access to products/services
- to provide a variety of buying options.
- to reduce physical traveling, especially during COVID.
- to reduce redundant interaction for the customers.
- to increase the customer base of businesses.
- to provide a platform for business owners to compete with other businesses.
- to provide logistical accounting data to businesses on sales and inventory
- to improve the sales of businesses

Broadly, the application will support the following:

- More Product Variety at One Place: Store owners and service providers can put a catalog of their products/services and update their inventory and prices.
- Improved Customer Comfort: Store owners also provide the option for order pickup or on-door delivery.
- Improved Customer Experience: Students can browse the catalogs of different stores, compare prices and order items all on platform and can obtain a digital bill of their order.
- Ease of Customer Interaction: Students can get their transaction history, order statuses, recommendations, promo code, and business owners can get their sales history(that can be easily exported to Excel or LibreOffice), create custom promotion offers.

There are several benefits associated with the product:

- Increased accessibility and variety of products and services.
- Maintainable records of previous sales and transactions.
- Reaching out to the stores physically can be avoided.
- Compare prices across various options available.
- Customers can see their current order statuses.

1.2 Intended Audience and Document Overview

Intended Viewers

As we start the design, developers would use this document to communicate and proceed. Then at the time of testing, testers would use this document. It can then be used to form a user manual using some subset of the SRS.

Sections and Organisation

The SRS starts with sections that describe its importance and how to use it, then it moves forward with the overview and functionality of the product. Then it covers a guide towards the interfaces which make it easier for a user to understand how they need to interact with the product, covering some use cases to wind up.

Sequence for Important Sections

- A Developer or a Tester can start reading the document with focus on overview (2.1), functionality (2.2), design and implementation (2.3), interfaces (3.1) and functional requirements (3.2).
- A user should focus on the scope (1.1), overview (2.1) and functionality (2.2), and then the specific requirements, going through the sections 3.1.1, 3.1.3 and 3.2.

1.3 Definitions, Acronyms and Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
AWS	Amazon Web Services
CRUD	Create, Read, Update, Delete
DB	Database
DDB	Distributed Database
ER	Entity Relationship
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JS	JavaScript
SOTA	State of the Arts
SRS	Software Requirements Specifications
TS	TypeScript

1.4 Document Conventions

Formatting Conventions:

- This document is written with Arial font of size 11 with single spacing and 1-inch margins.
- Words highlighted with bold in the same font space represent terms whose explanations are either given in footnotes or separately in the same section.
- Italics highlight has been used for comments.
- Underline has been used for headings in subsections.
- Bullet point ordering has been used as a listing typesetting tool.

Naming Conventions:

- Buyer: Any potential end customer for the sellers in the application.
- Seller: Any business owner who catalogs products in the application.
- Users: Buyers and Sellers
- Admin: Administrator who gives technological and product support to the users.

1.5 References and Acknowledgments

The following websites were referred to in the process of making this document:

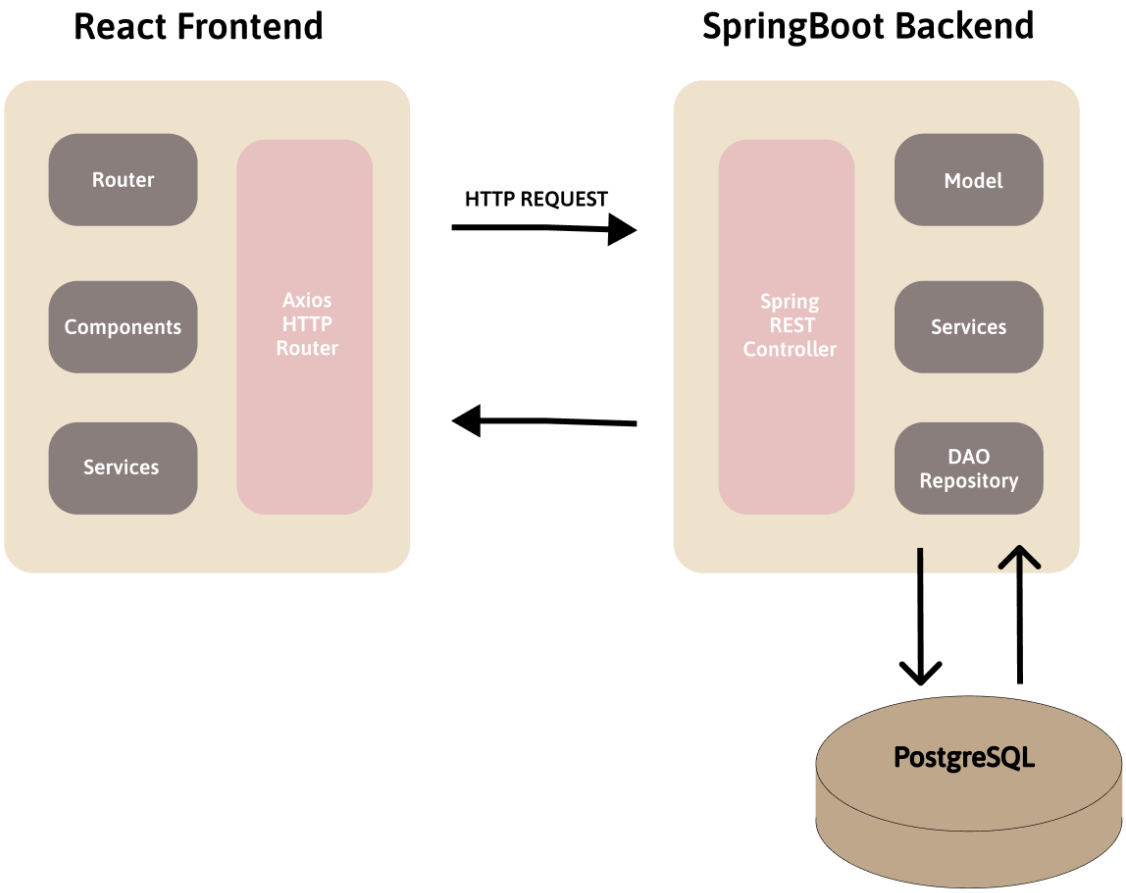
- Use-Case Model - Javatpoint
- Figma
- Creately

We'd also like to acknowledge the help of our TA, Mr. Swastik Maiti, for their valuable input in the creation of this document. We also would like to thank Prof. Indranil Saha for providing the SRS template and teaching the concepts.

2 Overall Description

2.1 Product Overview

"store.it" is a web-based application implementing a Model-View-Controller model.This system is the new self contained product that provides a simple mechanism to digitize all the stores and services on the IITK campus.



2.2 Product Functionality

The product would allow every user to classify as a *customer* or *seller*.

The product should have the following functionality for the seller:

- create an account and fill in the store details.
- details could contain store phone number, type of service provided, address, etc.
- manage the catalog of their product/services.
- add/remove/edit product/services and categorize in several categories.

- update the inventory and mark an item out of stock.
- view the list of orders placed/in-process/dispatched/delivered.
- export the entire list of orders in .csv format

The product should have the following functionality for the customer/buyer

- create an account and fill in the user details like name and delivery address.
- sign In to the account via email.
- check status of previous orders (Acknowledged/Dispatched/Delivered) listed by date.
- search and navigate across all the stores listed on the platform.
- browse the menu of a particular store and see all the listed products which are subdivided into categories along with their prices.
- add/remove products into the cart.
- order the products in the cart.
- obtain a digital bill for their purchase in .pdf format

2.3 Design and Implementation Constraints

Hardware Limitation:
We need a computer/mobile and an internet connection for this application to build and execute.

Software Specifications: We need a internet browser like Google Chrome, Brave, Mozilla Firefiox etc and an Operating system that runs on 64 bit architecture for this application and its dependencies to run smoothly.

2.4 Assumptions and Dependencies

We plan to use multiple components with an already prevalent technology stack. The assumptions that plan to have to keep the service working are:

- We rely on the email services for a user to log in, hence they need to be working at times.
- Our services would be best served on modern web browsers as we are designing with some of the modern web browsers in mind. If someone is using the older versions of web browsers, then the interface might not be efficient.
- The backend services would be hosted on ~~AWS~~ Heroku. Hence, Heroku Web Services should be working for smooth conduct.
- The same is true for the database, hence Heroku must be working
- Since the internet would play a major role in all of this, it is assumed that the users have a reliable internet connection.
- The frontend has also been hosted on Heroku, hence Heroku Web Services must be working

The efficient working of the project would depend on the contribution of the technology stack we are using, which means that the uptime of Heroku web services and other services such as mailing

servers would affect the time taken for the users to complete a transaction, hence affecting the efficiency of the program. The internet connection would also play some role in all of this, the better the internet connection, the better the things would work.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

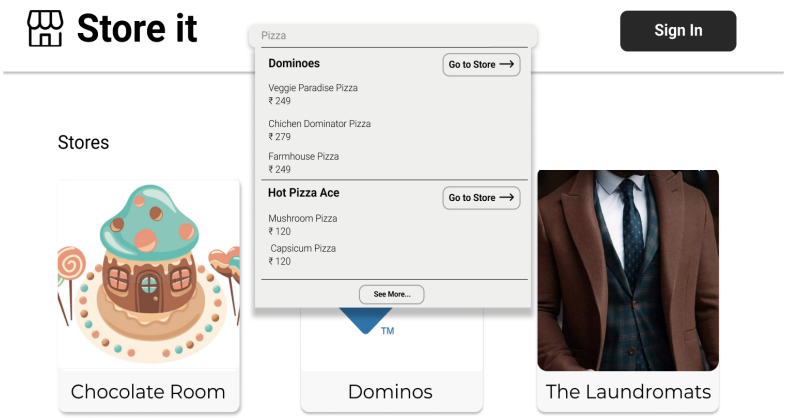


Fig-1: Home Page

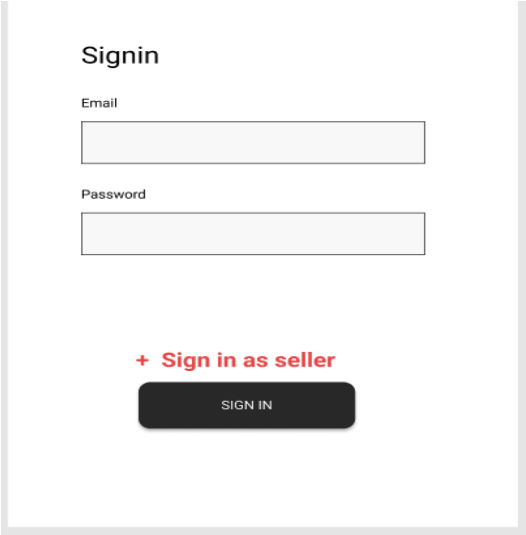


Fig-2 : Sign-in

1. Home Page

The home page will contain the application name and logo on the top left, and a sign-in button at the top right. The users can click on the button and proceed to the sign-in menu to sign in to their account. A user can also do a global search and find what all stores offer a particular item/service. The users can also see some cards displaying the links and icons of some of the stores present on StoreIt. For example, Chocolate Room, Dominos and The Laundromats in the figure attached. The users can navigate to the required store by clicking on the corresponding card.

2. Sign-in

The sign-in window will ask for the email and password of the user. The user can enter their email and password in the appropriate text fields and click on the Sign-in button at the bottom to sign in to their account.The user will be asked whether they want to sign-in as a customer or as a seller and they can select their roles and then proceed to sign-in.

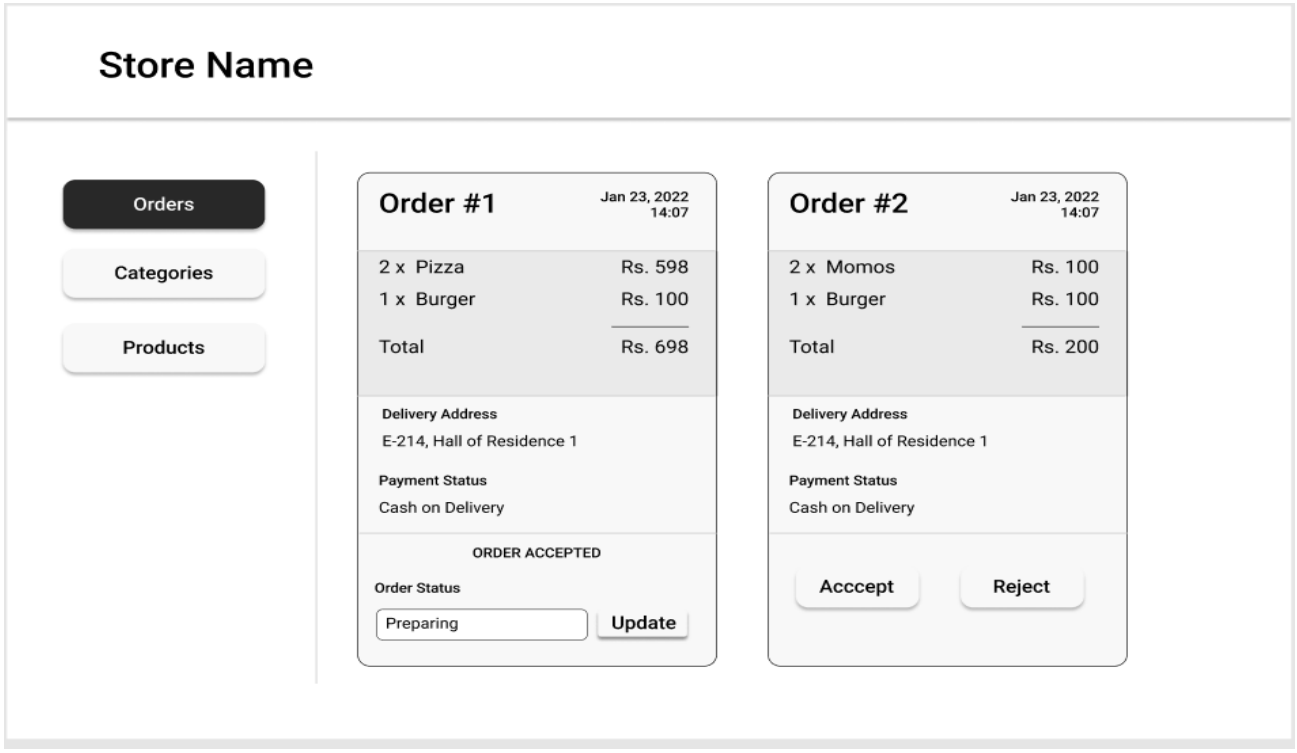


Fig-3 : Orders view (Seller's Dashboard)

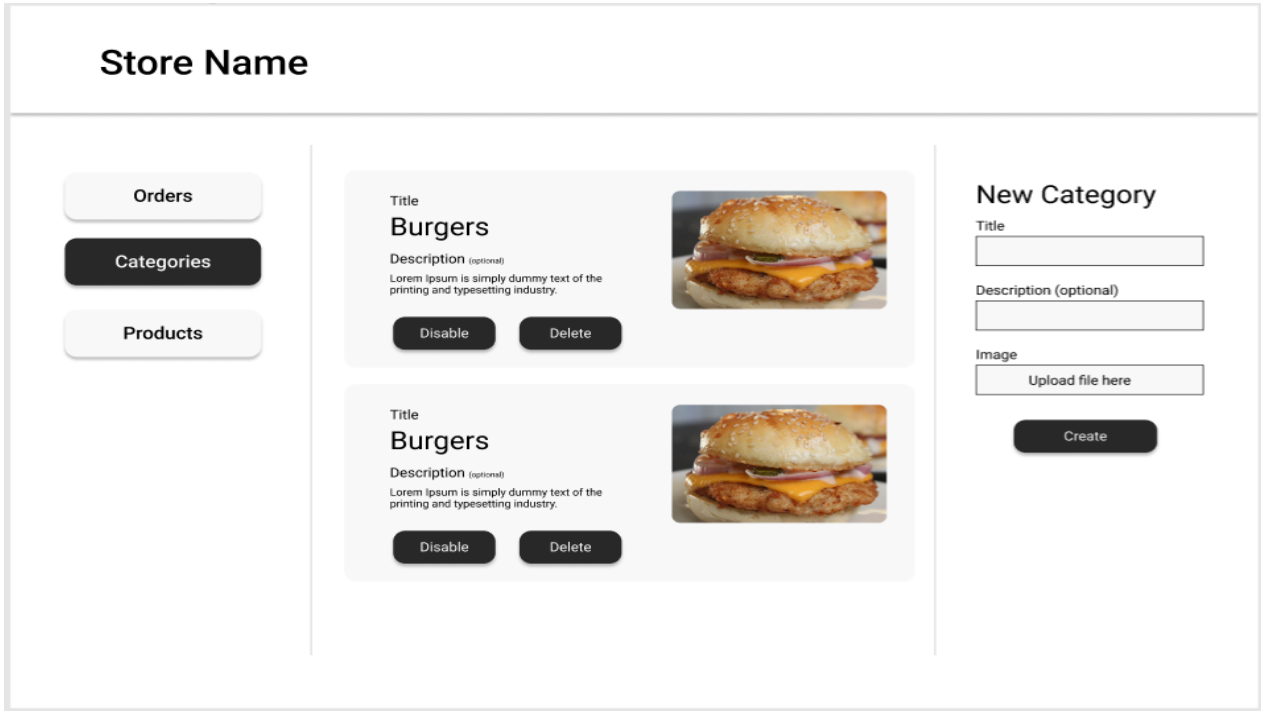


Fig-4 : Categories view (Seller's dashboard)

3. Orders view (in Seller's Dashboard)

Seller can select the orders view from the dashboard(left side menu in the graphic attached), the seller can view and manage the pending as well as non-accepted orders. For accepted orders(ex., Order #1), the seller can update the status(preparing, dispatched, delivered, etc.), as well as view order info such as payment method, delivery address and total bill amount. For new orders(ex., Order #2), the seller can view the total amount, payment method, delivery address and accept/reject the order by clicking on the accept/reject button.

4. Categories view (in Seller's dashboard)

Seller can display and manage various categories of products/services provided by his store. Cards containing details(title, description,.and image) of different categories will be displayed.
Sellers can disable/delete a particular category from the store by clicking on the disable/delete button present in the respective category's display.
Sellers can also add a new category to their store by entering the title, description and image of the category to be added in the right side menu. On clicking the *Create* button, the added category will also be displayed in the middle section.

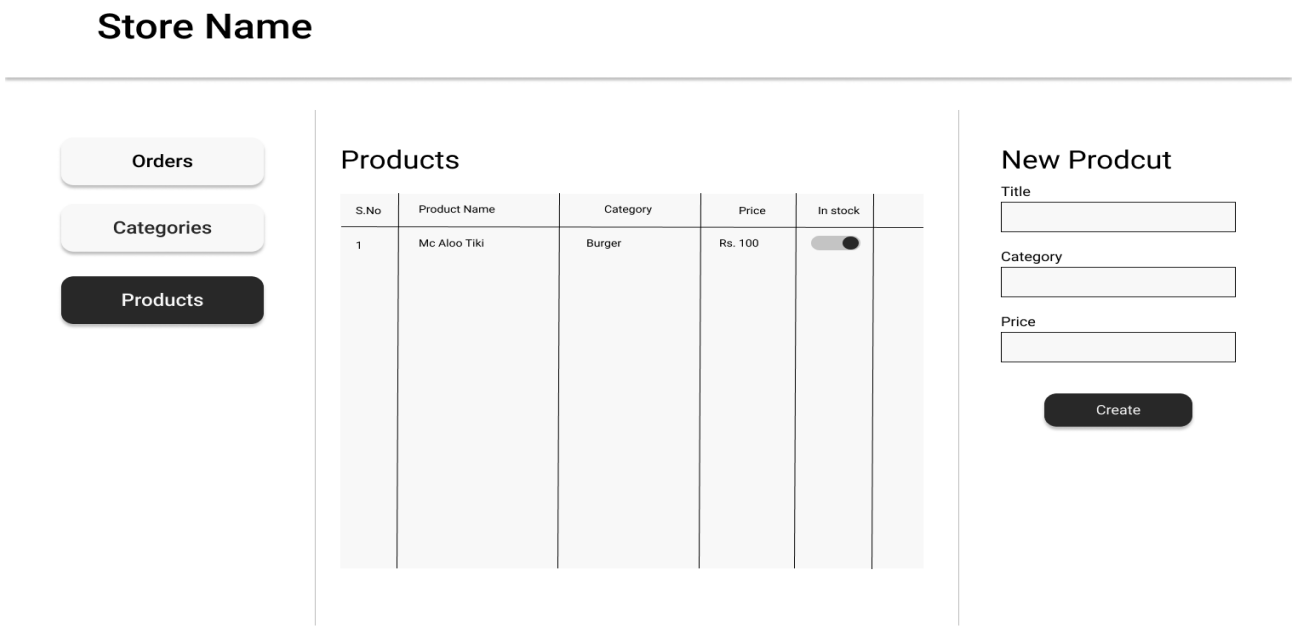


Fig-5 : Products view (Seller's Dashboard)

5. Products view (in Seller's Dashboard)

The sellers can manage the products available in the store here.
The products' details are displayed as a list. The seller can use the *In stock* toggle button to display whether the specified product is in stock or not. The user can click on the *dustbin* icon in the last column of each product to delete the product from the list. Sellers can also add a new product by entering product details such as title, category, and price in appropriate text fields in the right side menu and clicking the *Create button.*

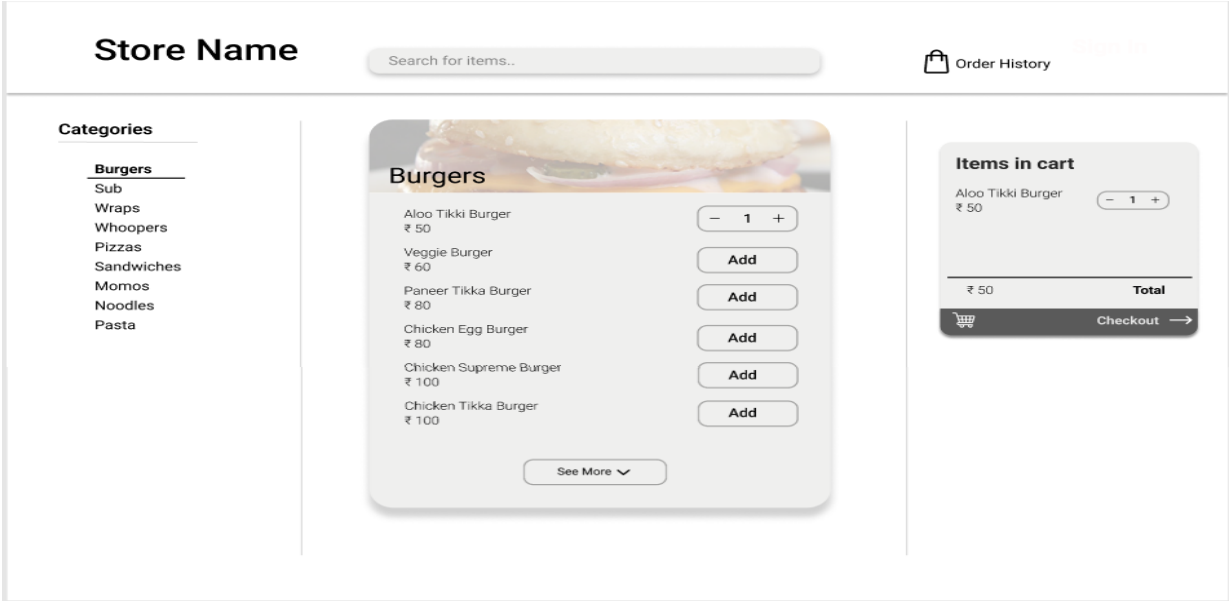


Fig-6 : Buyer's view

6. Buyer's view

In the top menu, the Store name, a search bar and a button to Order history view are displayed.
The user can search for different categories/products through the search bar.
The left side menu contains different categories available in the store and the user can navigate to the required category by clicking on the category name.The center of the view contains details of the products in the selected category. The user can add items to cart, click on '+'/'-' to change the quantity being added, and click on *See More* dropdown button to view more items in the list. The right side menu displays the current status of cart i.e., the current items in the cart, total amount, and a *checkout* button to place an order.

7. Order History view

The user can navigate back to the home page by clicking on the *Go back to Home* button in the left side menu. This view contains the Store name and a search bar to search the previous orders placed by the user. The center of the view displays the details(such as order number, date/time of order placed, items ordered, total among, delivery address, payment method, and order status) of

recent orders placed by the user. A button has been added on the bottom of each order to get a digital copy of their order in .pdf format

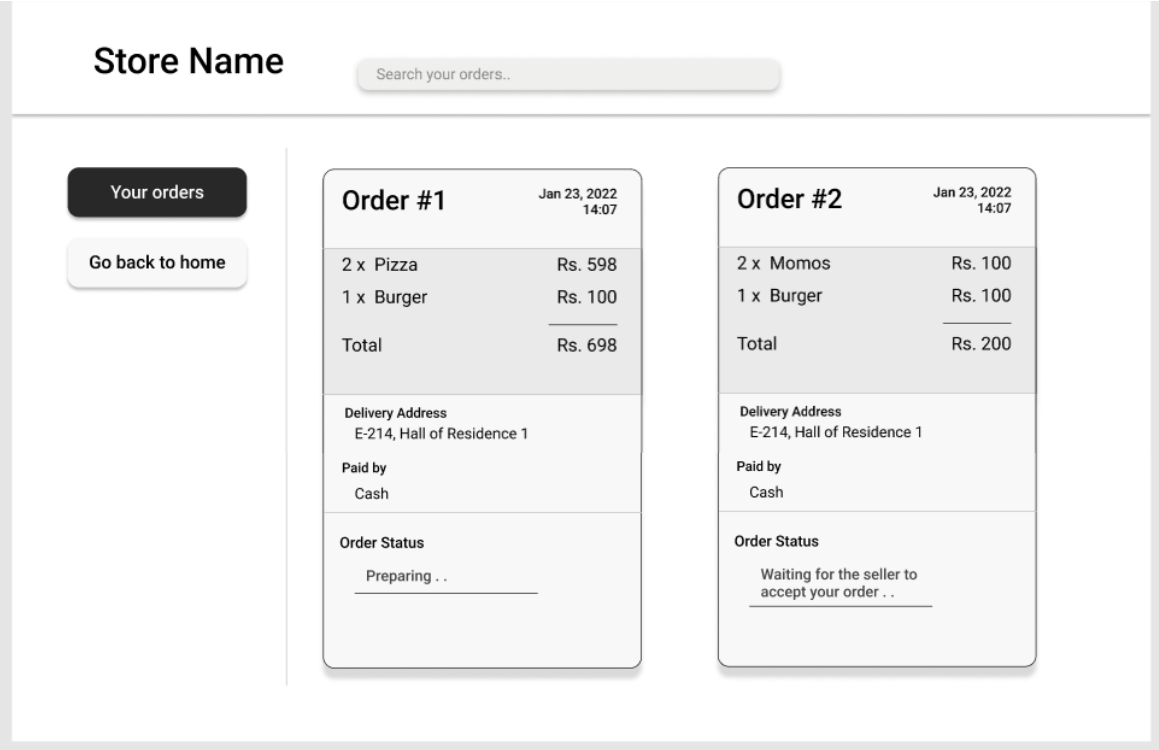


Fig-7 : Order History view

3.1.2 Hardware Interfaces

There are no hardware interfaces involved in the project. No additional hardware required.

3.1.3 Software Interfaces

There are no software interfaces involved in the project. No third party installations required.

3.2 Functional Requirements

3.2.1 Sellers could create an account and fill in the store details:

store_id=create_store(store_name, password)

3.2.2 Seller could add details like type of service provided, store logo, store banner etc:

add_details(store_id, type, store_logo, store_banner)

3.2.3 Sellers could add/remove/edit product:

product_id=add_product(store_id, name, description, image, price)

remove_product(store_id, product_id)

change_product(store_id,product_id, name, description, image, price)

toggle_visibility(store_id, product_id)

3.2.4 Sellers could manage the catalogue of their product/services:

toggle_visibility(store_id,product_id)

3.2.5 Sellers could also update the inventory and mark an item out of stock:

mark_out_of_stock(store_id, product_id)

3.2.6 Sellers could view the list of orders placed/in-process/dispatched/delivered:

order_list = view_order_list(store_id)

3.2.7 Sellers can update the status of their orders:

order_update(store_id,order_id,action_type)

3.2.8 Sellers can Get transaction history csv:

print_transaction_csv(user_id)

3.2.9 Buyer can create an account and fill in the user details:

user_id = create_account(name, created_password, OTP)

3.2.10 Buyers can search for all the products in all the stores in a global search:

search_results=global_search(keywords)

3.2.11 Buyers can check status of previous orders (Acknowledged / Dispatched /Delivered):

status = check_status(user_id, order_id)

3.2.12 Buyers can check all previous transactions listed by date:

prev_transactions = check_transactions(user_id)

3.2.13 Buyers can search and navigate across all the stores listed on the Storeit:

store_list = get_store_list()

3.2.14 Buyers can browse the menu of a particular store and see all the listed products which are subdivided into categories along with their prices:

menu = get_menu(store_id)

3.2.15 Buyers can Add/Remove products into the cart:

add_into_cart(user_id, cart_id, store_id, product_id)
remove_from_cart(user_id, cart_id, store_id, product_id)

3.2.16 Buyers can checkout the entire cart:

checkout_cart(user_id, cart_id)

3.2.17 Buyers can add phone number and delivery address at the time of checking out the cart:

checkout_cart_add_details(user_id, cart_id, phone_number, delivery_address)

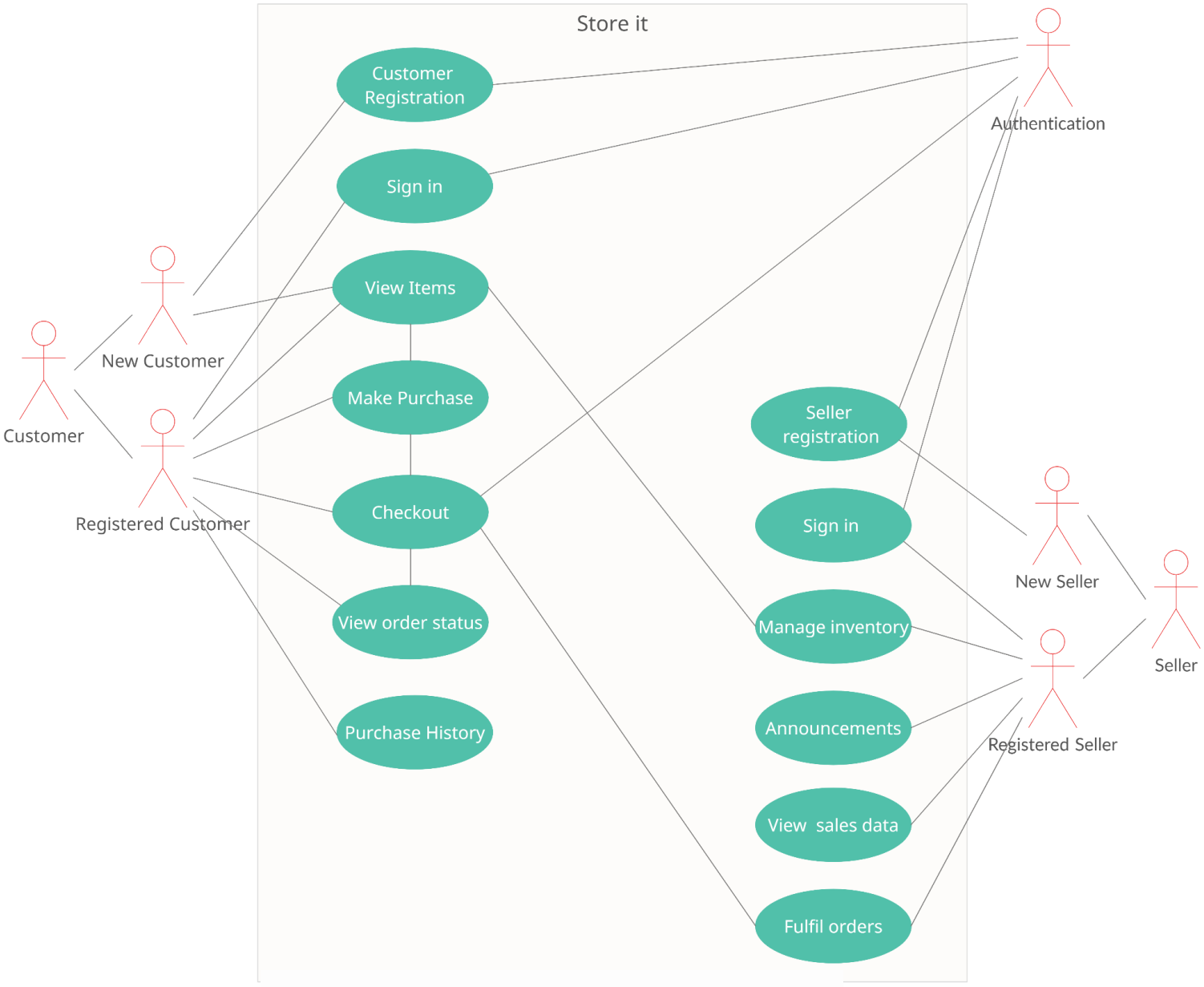
3.2.18 Buyers can place an order:

order_id_list = place_order(user_id, cart_id)

3.2.19 Buyers can get order print:

print_order (user_id, order_id)

3.3 Use Case Model



3.3.1 Use Case #1

Use Case S.No.	Use Case #1
Author	Akanksha Singh
Purpose	Shopkeeper registration and catalogue creation
Requirements Traceability	Shop Name, Shop ID, List of all items available at the shop with brief descriptions and prices.
Priority	High
Preconditions	None
Post Conditions	Shop is registered in the database and has an ID Catalogue of the shop is visible to users
Actors	Shopkeeper, Shop's catalogue
Exceptions	Creating a catalogue for which shop does not exist in the database.
Includes	None.
Notes/Issues	Make sure that the shop owner profile is logged in or created before catalogue modification.

Use Case S.No.	Use Case #2
Author	Akanksha Singh
Purpose	Browse catalogue of a particular shop and see all items being sold by the shopkeeper along with short descriptions and listed prices.
Requirements Traceability	ID of shop, item details and price list of shop products.
Priority	High
Preconditions	Shop must be registered and have an ID Item details of a product should be complete.
Post Conditions	Access to shop's catalogue
Actors	User's device, Shop's catalogue

Exceptions	Shop name requested is not registered. Item details of a product are not complete.
Includes	Use case #1
Notes/Issues	Items list has to be visible. While adding a product to the shop's item list, the completeness had to be made sure.

Use Case S.No.	Use Case #3
Author	Akanksha Singh
Purpose	Add item to cart
Requirements Traceability	Item chosen, Shop ID, Product ID
Priority	Medium
Preconditions	Shop must be registered and have an ID Item chosen must be available
Post Conditions	Cart now has the required item listed in it.
Actors	User's device Shop's catalogue
Exceptions	Shop name requested is not registered. Product is not available.
Includes	Use Case #1, Use Case #2, Use Case #4
Notes/Issues	Item availability might change during processing the order

Use Case S.No.	Use Case #4
Author	Akanksha Singh
Purpose	Register as a buyer using Phone number and OTP verification.
Requirements Traceability	Name and Phone number of buyer
Priority	High

Preconditions	None
Post Conditions	User is now registered in the database with their phone number
Actors	User's device
Exceptions	Phone number has already been registered. Phone number is invalid.
Includes	Nothing
Notes/Issues	OTP could not be sent for any reason. Phone number validity to be checked. If already registered then just log in through password.

Use Case S.No.	Use Case #5
Author	Akanksha Singh
Purpose	Checkout and place order
Requirements Traceability	Buyer's name, registered number and address. Cart with items IDs and names of shops from which buyer is ordering
Priority	Medium
Preconditions	Buyer and seller have to be registered. Items must be available.
Post Conditions	Order is listed in the "orders" list of buyer Shop receives order details with address, items and total
Actors	User's device Shop's catalogue
Exceptions	One or more of the items is no longer available
Includes	Use case #1 Use case #2 Use case #3 Use case #4
Notes/Issues	Check had to be there at the checkout stage.

Use Case S.No.	Use Case #6
Author	Akanksha Singh
Purpose	Shopkeeper adds/removes items from shop inventory
Requirements Traceability	ID of shop Item and price list of shop
Priority	Medium
Preconditions	Shop must be registered and have an ID Item list of Shop must be available
Post Conditions	Item is added/removed from item list
Actors	Shopkeeper Shop's catalogue
Exceptions	Item to be removed isn't in the items list
Includes	Use case #1
Notes/Issues	Removing an item present in the carts

Use Case S.No.	Use Case #7
Author	Antreev
Purpose	Customer downloads the digital copy of order purchase in .pdf format
Requirements Traceability	ID of Customer Order ID
Priority	Low
Preconditions	Customer must be registered and have an ID
Post Conditions	None

Actors	Student Order List
Exceptions	None

4 Other Non-functional Requirements

4.1 Performance Requirements

Performance of the application depends on some factors like

- APIs: The application is expected to have a traffic of over 5K users. The APIs should be scalable enough to support this traffic. The IITK community is about 10K. We expect 25% of this to be our target audience.
- Concurrency: The database read/write operations should respect the ACID properties as there would be multiple users accessing the same API at the same time so there shouldn't be any concurrency issues
- Latency: It is preferable that API requests take less than 200ms to fulfill, so that the App runs smoothly. This is quite close to the standard expected latency in industry.

4.2 Safety and Security Requirements

Safety Requirements:

- Privacy: The email id provided by the user should not be disclosed in any form without the consent of the user. Same applies for other personally identifiable user data.
- Mandatory login: All users must create a login account before placing orders (the buyer) and before setting a store (the seller). This ensures the authenticity of users.

Security Requirements:

- Confidentiality: Only the buyer and the corresponding seller of the store should be able to see the order details.
- Integrity: The software should not corrupt the order history for the buyer and the store. The data associated with the stores should not be damaged.
- Encryption: All sensitive information such as passwords should be stored in an encrypted format using SOTA encryption schemes.

4.3 Software Quality Attributes

4.3.1 Availability

The application will be available and accessible to every user inside the campus. This application can be operated by any resident/business owner inside the IITK campus having an account, without any restrictions.

4.3.2 Consumability

A user-friendly interface will make the use of the application quite intuitive for the customers using it for the first time. The catalog can be used to compare the prices of the products and the "add to

cart" feature makes it easier for the customers to place orders. The naming of the various menus/windows/options inside the application will be done in a manner that they are most likely self-explanatory.

4.3.3 Correctness

Correctness can be achieved by asking the store owners to maintain the catalog by updating the correct prices of the items/services provided by them. Further, these prices will be checked by the store owners after each month to ensure correctness. These prices can be verified by the user by comparing the prices with other options available online or by directly contacting the store.

4.3.4 Interoperability

The operation of this application will not interfere with any other application operating simultaneously on the device. While purchasing products, users can simply add the product to the cart, operate any other application simultaneously, and resume purchasing from the Recent screen/browser tabs. This will ensure the interoperability of the application.

4.3.5 Scalability

The software should be able to accommodate any future stores that may be opened in IITK campus. The software services should be designed such that it is easy to scale both horizontally and vertically.

5 Other Requirements

Legal data requirements

User data collected via the website should be stored in the country's regional data center. This is to comply with data protection and privacy rules that may be enforced by the Government of the nation.

Internationalization requirements

All strings and locale-specific references from the code should be maintained in a resources file which creates the possibility of using other languages in the future.

Appendix A – Data Dictionary

- **User class**

Element Name	User
Description	The user of the software. Each user registers using an email and a password which is later used to login into their account.
Attributes	1. **email** : string 2. **password** : string
Operations	1. **register()** : registration of a user using the software for the first time. 2. **login()** : called while user wants to login
Relationships	Each user can have one account corresponding to a particular email used while registration. There are two types of users : Buyer and Owner.

- **Buyer class**

Element Name	Buyer
Description	The buyer can buy the different products available in the stores.
Attributes	1. **customerName** : string 2. **phoneNumber** : string
Operations	1. **getShoppingCartByStoreID()** : used to access the shopping cart in a particular store.
Relationships	Buyer is one of the two types of User and hence, related to the User class. It has a one-to-many relationship with the

	ShoppingCart class and Order class.

- **Shopping Cart class**

Element Name	Shopping Cart
Description	The shopping cart contains the items added by the buyer that they want to purchase. Each cart has a customer ID and a store ID.
Attributes	1. **customerID** : string 2. **storeID** : string
Operations	1. **checkout()** : for checkout 2. **calculateCartValue()**: calculate the total value of the items in the cart 3. **addCartItem()** : add an item to the cart 4. **getItems()** : get the items in the cart
Relationships	The Shopping Cart class has a many-to-one relationship with the Buyer class as well as the ShoppingCartItem class.

- **Order class**

Element Name	Order
Description	Order placed by the sellers through the software. Each order has the ID of the customer, store ID where the order is being placed, amount for the transaction, and its payment status and shipping info.
Attributes	1. **customerID** : string 2. **storeID** : string 3. **shippingInfo** : string 4. **currentStatus** : string

	5. **paymentStatus** : string 6. **amount** : int
Operations	1. **updateOrderStatus()** : used by the seller to update the status of an order
Relationships	The Order class has a many to one relationship with Buyer, Seller and the OrderItem classes.

- **Store class**

Element Name	**Store**
Description	A store offers different products to the buyers. Each store has its name, its address, contact, and an imageURL.
Attributes	1. **name** : string 2. **address** : string 3. **contact** : string 4. **imageURL** : string
Operations	1. **addCategory()** : used by the seller to add a new category to the store
Relationships	The Store class has one-to-one relationship with the seller and a one-to-many relationship with the categories.

- **Category class**

Element Name	**Category**
Description	The categories present in a store. Each category contains different products related to that category.
Attributes	1. **title** : string

	2. **description** : string 3. **imageURL** : string
Operations	1. **addProduct()** : add a new product under the category
Relationships	The Category class has a one-to-many relationship with Product class and a many-to-one relationship with a store.

- **Product class**

Element Name	Product
Description	The products available in the store. Each product has a title, description, price and a boolean value denoting its availability.
Attributes	1. **title** : string 2. **description** : string 3. **price** : int 4. **available** : bool
Operations	1. **updatePrice()** : used by the seller to update the price of the product 2. **updateAvailability()** : used by the seller to mark the product as available/not available.
Relationships	The Product class has a many-to-one relationship with the Category class since a category can include more than one product.

Appendix B - Group Log

Since the beginning of the project, our entire team has been very enthusiastic.

We have set up a separate Discord server as well as formed a Whatsapp group for effective communication.

We have created a private organisation on GitHub for collaborative software development.

Meeting minutes	Agenda
12 Jan 2022 7:30 - 9:00 pm	Brainstormed different project ideas and finalised a web application for Mess Management.
	Reworked on a new project idea because of conflict with other teams. Finalised a web application for digitising all kinds of stores for IITK campus.
23 Jan 2022 1:00 - 6:00 pm	Worked on a draft of Software Requirement Specification document.
23 Jan 2022 6:30 - 7:30 pm	Meet with TA. Updated TA with our current progress. Discussed any issues we were facing. Planned the further steps.
27 Jan 2022 1:20 - 5:00 pm	Made changes suggested by TA. Fine tuned the SRS document to make it ready for submission.
27 April 2022 5:00-7:00 pm	Made final changes to the Document