A Minor Project Proposal Report on

**Echo**

Submitted in Partial Fulfillment of the Requirements for

The Degree of **Bachelor of Engineering in Software Engineering**

Under Pokhara University

Submitted by:

**Pratik Bhandari, 181731**

**Prapti Koirala, 181730**

Date:

29 05 2022

**Department of Software Engineering**

**NEPAL COLLEGE OF INFORMATION TECHNOLOGY**

Balkumari, Lalitpur, Nepal

Abstract

Echo is a consumer-to-consumer (C2C) e-commerce application that is based on the web. It allows any user to buy as well as to sell products online, using the platform. Echo is designed to be decentralized into numerous virtual shops owned by individual users who wish to sell products where each shop defines its own rules and options regarding payment, delivery, product categories, etc. There is no additional registration other than signing up with the application, required for a user to start selling products. In Echo, each product a seller posts in the system is associated with a virtual shop owned by the seller, so a consumer can not only search for products but also search for particular shops and by extension, their owners.

Although a number of e-commerce platforms actively exist today, most of them have a business-to-consumer (B2C) model and systems that connect one consumer to another are limited in number. Echo aims to be a platform that is highly flexible and inclusive so that a large of spectrum of sellers can be accommodated. The platform is flexible enough for sellers ranging from an everyday user wishing to sell a one-off product to a user who wishes to sell hundreds of products with different stocks and product variations. Despite being designed with a C2C model in mind, Echo has the flexibility to allow a user having a physical shop or business to give it an online presence and not have to create a separate application for its advertisement or commerce.

Keywords: E-commerce, Consumer-to-Consumer, Decentralized, Virtual Shop

**Table of Contents**

[1. Problem Statement 2](#_Toc117614061)

[2. Project Objectives 3](#_Toc117614062)

[3. Significance of the Study 5](#_Toc117614063)

[4. Literature Review 6](#_Toc117614064)

[4.1 Existing Applications Similar to Echo 6](#_Toc117614065)

[4.2 Some Concepts in Theory 7](#_Toc117614066)

[5. Proposed Methodology 9](#_Toc117614067)

[5.1 Software Development Life Cycle 9](#_Toc117614068)

[5.1.1 Why Iterative Development Model 9](#_Toc117614069)

[5.1.2 Analysis Phase 9](#_Toc117614070)

[5.1.3 Design Phase 9](#_Toc117614071)

[5.1.4 Coding or Implementation Phase 10](#_Toc117614072)

[5.1.5 Testing Phase 10](#_Toc117614073)

[5.2 Programming Languages and Other Tools 10](#_Toc117614074)

[5.2.1 Programming Language 10](#_Toc117614075)

[5.2.3 Database 11](#_Toc117614076)

[5.2.4 Core Libraries and Frameworks 11](#_Toc117614077)

[5.2.4 Other Major Libraries 12](#_Toc117614078)

[5.2.5 Tools to be Used 13](#_Toc117614079)

[5.3 UML (Unified Modeling Language) Diagrams 14](#_Toc117614080)

[5.3.1 Use Case Diagrams 14](#_Toc117614081)

[5.3.2 Activity Diagrams 14](#_Toc117614082)

[5.3.3 Sequence Diagrams 14](#_Toc117614083)

[5.3.4 Entity Relationship (ER) diagram 14](#_Toc117614084)

[5.3.4 Class diagram 14](#_Toc117614085)

[5.4 API (Application Programming Interface) Endpoints 14](#_Toc117614086)

[5.4.1 Authentication 15](#_Toc117614087)

[5.4.2 User 15](#_Toc117614088)

[5.4.3 Shop 15](#_Toc117614089)

[5.4.4 Product 15](#_Toc117614090)

[5.4.5 Stock 16](#_Toc117614091)

[5.4.6 Rating 16](#_Toc117614092)

[5.4.7 Review 16](#_Toc117614093)

[5.4.8 Reply 16](#_Toc117614094)

[5.4.9 Subscription 16](#_Toc117614095)

[5.4.10 Category 17](#_Toc117614096)

[5.4.11 Notification 17](#_Toc117614097)

[5.4.12 17](#_Toc117614098)

[8. References 18](#_Toc117614099)

**List of Figures**

[Figure 1: Incremental Development Model 10](#_Toc117613673)

List of Tables

[Table 1: Major Libraries to be Used in the Front End 12](#_Toc117613775)

[Table 2: Major Libraries to be used in the Back End 13](#_Toc117613776)

[Table 3: Tools to be Used 14](#_Toc117613777)

# 1. Problem Statement

Even though a large number of e-commerce platforms exist these days, a very few of them have a consumer-to-consumer model. Most of the platforms connect businesses to consumers through a central hub. We believe that anyone who has access to the internet should be able to buy and sell what they desire online. E-commerce should not only be for businesses but also for everyday users who have products of value. No seller should have to own a physical business in order to sell any number of products to anyone. Applications must exist that not only enable commerce between any type, scale and domain of sellers and buyers but also allow proper communication between these parties.

Even prominent C2C platforms do not have the concept of product stock and product variations. A seller is not able to define the stock of their products. They are unable to specify variations that their products may have, such as color, size, storage, etc. Because of this, if a seller has some quantity and different variations of the same product, they must post each variation of the same product multiple times which is extremely inconvenient in an e-commerce platform. Just because a seller is not or does not own a business should not mean that all products that they sell are one-off products, so a platform should be flexible enough to accommodate stock and product variations.

A buyer of a product should be allowed to rate and review the product itself, as well as the shop (which implies the shop owner) that sells the product because it is crucial that buyers be able to evaluate a shop itself, so that other potential buyers can know what type of person they are buying from. Similarly, e-commerce systems that we have today do not have the concept of a virtual shop. Since each product is a part of a global marketplace and not scoped to a particular shop, potential buyers can only search for products and not for shops themselves that may sell the type of products that the buyers want. Furthermore, a potential buyer does not get notified when a new product that they may like gets added on to the platform. Each time, they have to manually search for a desired product to see if it exists.

Echo solves these problems by incorporating and fulfilling above mentioned features and requirements respectively to make buying and selling convenient and engaging.

# 2. Project Objectives

Some of the objectives of our project have been mentioned before to some extent in the problem statement section. Echo is developed to eliminate some of the shortcomings of the applications that are currently in the consumer-to-consumer domain. The primary objective of Echo is to allow anyone with an access to the internet to sell and buy products online. Echo also becomes a highly flexible web application so that it can accommodate a wide range of sellers and large potential stocks and product variations. One who does not own a physical shop can easily create their own virtual shop in our platform and start selling products.

Breaking down the project objectives we get some of the points mentioned below:

1. To allow users to create their own virtual shops having specific categories and delivery schemes of their choice with no additional registration other than signing up with the application. Each product that a seller adds is associated with their own virtual shop.
2. To allow a buyer and a seller to communicate using a chat system so that the buyer can have queries regarding products and shops answered by the seller. The chat system can also be used to set up meetings for product delivery.
3. To provide users an option to be notified when a particular shop that they regularly buy products from adds a new product so that a user does not have to manually search for products each time.
4. To notify users in real-time regarding various activities and actions such as orders made by consumers, reviews, ratings on products and shop, etc. (for a seller) and order confirmation, order rejection, etc. (for a consumer) and new chats, chat messages (for both).
5. To allow all products to have a stock and their own variations based on any attributes making it convenient for a seller to add a single product with multiple variations at the same time and a consumer to know the quantity of each variation that is available at the moment.
6. To allow for easement in product search and access by associating each product that a seller posts to the virtual shop owned by them so that a consumer can not only search for products or products specific categories but also search for shops and shop owners.
7. To allow a consumer to rate and review not only a product but also a shop itself (which implies shop owner) so that other consumers understand the type of person they are buying from.
8. To make Echo flexible and accommodating so that all ranges of sellers and sell their products. From a user who wishes to sell a one-off product with a single quantity to a user who wishes to sell hundreds of products each with its own stock and variations, all can be accommodated. It is this flexibility of Echo that allows even users that own physical shops or businesses to give them an online presence.
9. To allow a buyer to track the status of their orders. This includes checking if an order has been viewed and acknowledged by the seller, whether an order has been confirmed or denied, whether a delivery date has been set (if a product is set to be delivered) or if a meet-up date and place has been provided (if a product is not set to be delivered) and so on.
10. To give users an option to choose between dark and light themes for the application.

# 3. Significance of the Study

As mentioned before, there are several existing e-commerce platforms that allow users to buy different kind of products and the problem is that most of these applications are business-to-consumer (B2C) which do not allow regular users that do not own businesses to sell products. As a result, any user who wishes to have their own virtual shop is not able to have one or even when it’s possible they must be registered and have their registration number which becomes highly inflexible. It must be realized that an application must be flexible enough to accommodate all kinds of features that a producer or consumer wishes to have. Echo is a consumer-to-consumer (C2C) platform that allows a user to not only post one-off products with no stock and variations but also to post hundreds of products with different variations and stock that the product might have. A user creates a virtual shop where they can post every product according to the category that they have specified and hence have their own virtual shop. The user gets to specify variations of the product that they have which is not available in other existing platform but is one of the important aspects to be considered. In regards to a buyer, they can search for their choice of products in most of the existing ecommerce application but in case of our project not only products user can search for the shop or the shop’s owner for their convenience.

Also, we have seen that reviewing and rating products is possible in most e-commerce applications so that buyers are aware of the condition of the product that other users have received or the service that other users have received. But Echo, being a platform for establishing a virtual shop, allows the consumer to review not only the product but also the shop from which they have purchased or the owner who owns the shop. This builds the trust of the user who are engaged in this platform as they know from other users about the shop, their service or their products.

The overall project and the system it intends to build are significant as the system aims to eliminate the shortcomings that exist in other C2C e-commerce platforms that exist today and incorporate new features and functionalities that are useful to both buyers and sellers.

# 4. Literature Review

## 4.1 Existing Applications Similar to Echo

Echo is a consumer-to-consumer (C2C) ecommerce platform but not only one to provide the services that it does. There are several other applications that are similar to Echo but Echo is somewhat different from each one of them. Some of the similar existing applications are mentioned below.

1. Hamrobazar

Hamrobazar is a consumer-to-consumer e-commerce platform and among existing applications, it is probably the most similar to Echo.

As compared to Echo, the disadvantages and advantages that Hamrobazar has are:

Disadvantages:

* There is no concept of stock, so each product that is sold is a one-off product with only one quantity.
* A seller cannot define variations of a product. Because of this, if the same product with different variations such as size, color, etc. is to be posted, a separate product has to be posted for each variation. Echo, being a flexible e-commerce platform, reduces the overhead of posting a product with different variations by allowing a user to post different products with their own variations and stocks at once and associating the product with the virtual shop owned by the user.

Advantages:

* Hamro bazar has a recommendation system that recommends different products to different users.

1. Gyapu

Gyapu is also an e-commerce platform where users have the option to sell their products. However, if a user wants to sell products in Gyapu, they must own a business and apply with PAN number or VAT number of the business.

Compared to this, we plan to design Echo such that a seller does not have to own a business and even if they do, they can give their business an online presence by simply signing up into the application.

1. Sasto Deal

Sasto Deal is also an e-commerce platform similar to Echo but here as well, any user who wishes to sell their products must have a business and they must register with their VAT number or PAN number. Because of this, if a user not owning a business wants to sell products they cannot do so. In case of Echo, as mentioned before, any user can sell their products by simply signing up into the application and creating a virtual shop.

However, Sasto Deal has an online payment system. Online payment will not be integrated into Echo as there might be some trust issue by not having a third party to contact.

1. Daraz

Daraz is the leading ecommerce application in Nepal, however being a business-to-consumer (B2C) platform, a user must own a business and register it with Daraz before they can sell products.

## 4.2 Some Concepts in Theory

1. Redux for State Management

React being a declarative library, one of the most important aspects of our project will be its state. A state is essentially a set of properties that control the behavior of components that make up Echo. Therefore, proper management of the state as well as deciding how it will be share between components is very important. Keeping these points in mind, we have decided to use Redux.It is a library which can be used with react to manage the state. Without redux, sharing of state from a parent component to its nested child components as well as vice versa can be extremely difficult as the project grows. Redux takes the state as stores it in what it calls a storewhich lies above all components, so if a component no matter where it lies in the component hierarchy wishes to access the state or makes changes to the state, it can use methods provided by redux.

1. ORM

ORM stands for Object Relational Mapping and lets us query and manipulate data from a database using an object oriented paradigm. ORM is a library that encapsulates the code needed to manipulate the data so we don’t have to use SQL anymore and interact with the object directly. Using ORM saves a lot of time as data model is written only in one place and is easier to update, maintain, and reuse the code. Also, a lot of stuff is done automatically using ORM. ORM library is more flexible to use as it fits naturally in our way of coding and abstracts the database system that allows us to change database whenever we want.

1. Server-side Rendering

It is basically the process of rendering a page in a server rather than on a browser. Server-side rendering can make an application fast and performant is used properly. A server sends a fully rendered page to the client that is handled by the client-side code. Server-side rendering also helps with search engine optimization as page contents are rendered on the server where web crawlers look. Server-side rendering improves user experiences by allowing pages to load even in slow internet connections.

1. API

API stands for Application Programming Interface that makes it possible for two independent systems to interface/communicate with each other. We can use APIs in many cases like to get data for a web app or to connect to a remote server that has data that keeps on changing or to enable two applications to exchange data among each other. It not only provides reusability of code but also uses the concept of abstraction by hiding complexity and showing functionality only. In web development, we create an API to enable the front end of an application to communicate with a server. In a REST (Restful) API, there are routes that the front end can make HTTP (Hypertext Transfer Protocol) requests with different HTTP methods to get, post, update, delete data on a database.

# 5. Proposed Methodology

In order to successfully plan, design, implement, test and deploy our application on the web, we have implemented, used or adopted following methodologies and technologies:

## 5.1 Software Development Life Cycle

We followed the principles of iterative development model for the overall tasks and activities that we have to carry out during the development of this system.

### 5.1.1 Why Iterative Development Model

Iterative development is a way or process of developing software in small and numerous increments. Each increment corresponds to one or many features or functionalities being incorporated into the system being developed. With successive increments, the system to develop becomes more refined and complete version of itself. Rather than developing an entire application or software in one go which can become exceedingly difficult as the size of the project increases and requirements change, following the principles of helps to cope with changing requirements of the system as new requirements can be dealt with in new increments of development. Iterative development will also allow us to conceptualize the whole system in terms of features that get added as the development process continues.

Each increment of development process has following phases:

### 5.1.2 Analysis Phase

Analysis is performed prior to anything else during in an increment. Analysis phase consists of activities such as identifying and analyzing requirements, studying the application domain, figuring out the scope of the increment and so on. If changes are being made, the effects of these changes on the overall system are also studied.

### 5.1.3 Design Phase

After a thorough analysis of the requirements is done, the overall architecture of the increment is designed based on the analysis. The architecture shows the relationships between the components that make up the increment as well as the overall system and flow of data and information in the system.

### 5.1.4 Coding or Implementation Phase

In this phase, the planned increment is implemented by writing code using programming languages, libraries and all the tools available. The previous two phases have significant role in aiding in the implementation process.

### 5.1.5 Testing Phase

After a planned increment has been implemented, it is time to test whether the functional and non-functional requirements of the increment have been fulfilled. Various tools can be used for testing an increment before it is incorporated with the rest of the application or deployed. Testing may come in various forms and types.

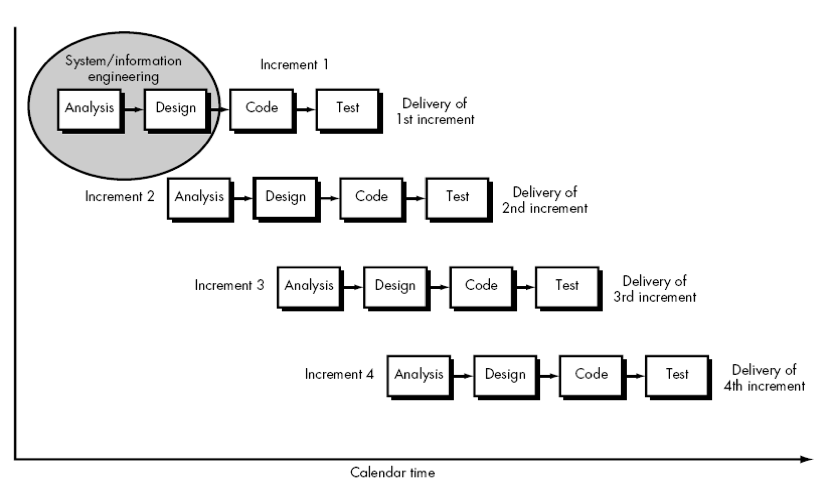


Figure 1: Incremental Development Model

## 5.2 Programming Languages and Other Tools

### 5.2.1 Programming Language

We have used JavaScript as the only programming language for the development of Echo. It is the language of the web and it is powerful enough to implement both the client side and the server side of the proposed system.

5.2.2 Back-end Runtime

We used Node Js as the run-time environment in the back end. We decided to use it because it is a run-time environment for JavaScript and also is extremely performant. The community is huge around Node Js and with it in the back end and Next Js and React Js on the front end, the entire code can be written in JavaScript.

### 5.2.3 Database

We have used PostgreSQL database. It is a relational database system which is suitable for Echo as multiple relationships between data models have to be addressed as the application is developed. It is an open source database system which has been in development by hundreds of developers for more than 30 years. It is also one of the database systems supported by Prisma.

### 5.2.4 Core Libraries and Frameworks

The core libraries and frameworks used on the front end and the back end of the application are:

5.2.3.1 Front End

5.2.3.1.1 Next Js

Next Js is a library built on top of React Js that is mostly used for server-side rendering. However, it allows both server-side and client-side rendering of pages, which makes it flexible to use. We expect to make Echo a hybrid of client-side and server-side rendered pages, so Next Js is suitable. Next Js also greatly optimized images and supports lazy loading out of the box and also helps with search engine optimization.

5.2.3.1.2 React Js

React Js is a library or framework used to build interactive user interfaces (UIs) having highly reusable and performant components. A UI created with React Js is component-based, meaning that it is made up of several entities called components having their own state which is a set of properties that describes the current appearance or behavior of the component that are integrated together to make larger parts of the UI.

5.2.3.2 Back End

5.2.3.2.1 Express

Express is a framework for Node Js which is a javascript run-time environment. It greatly simplifies the process of developing APIs (Application Programming Interfaces). Express also helps with error handling and supporting the overall process of creating routes and controllers on the back end.

5.2.3.2.2 Prisma

Prisma is an ORM (Object Relational Mapper). It is a library that enables an application to safely read and write data to and from a database. With the abstraction and syntax provided by Prisma, developers avoid having to write low-level SQL (Structured Query Language) or NoSQL queries which can source of errors and complications if not used properly.

### 5.2.4 Other Major Libraries

Major libraries (not mentioned in the core libraries and frameworks section) that we expect to use throughout the development of Echo on the front end and the back end are tabulated below. By the time the final system is completed, other libraries may also be used.

5.2.4.1 Front End

Table 1: Major Libraries to be Used in the Front End

|  |  |
| --- | --- |
| Library | Application |
| Redux Toolkit | It will be used in managing the state of the entire application. As the size of a system increases, state management with only React becomes highly complex and tedious. |
| React Redux | It will be used to connect Redux with our application that will be written in React. So, React Redux is a bridge between Redux and React. |
| Tailwind CSS | It is a CSS library having multiple useful and customizable utility classes. Tailwind CSS will be used to increase productivity and simplify the process of styling components. |
| React Query | It provides hooks for fetching, caching and updating data from a back end. With React Query, data displayed on a component or a page remains fresh and is sync with the data that is on a database. |

5.2.4.2 Back End

*Table 2: Major Libraries to be used in the Back End*

|  |  |
| --- | --- |
| Library | Application |
| Joi | It is used for data validation. It is a powerful library that describes a schema for data to be inserted into a database. |
| JSON Web Token | It generates and validates web tokens. It will be used in Echo for user authentication and authorization. |
| Bcrypt | It will be used to hash and check hashed passwords before they are inserted into a database. |
| Multer | It is a library or middleware for handling multipart form data. In Echo, it will be used to work with images. |

### 5.2.5 Tools to be Used

The tools that we expect to use during the overall development process of Echo are:

Table 3: Tools to be Used

|  |  |
| --- | --- |
| Tool | Application |
| Visual Studio Code | It is a text editor, so it will be used to write code for the entire application. |
| Google Chrome | It is a web browser and will be used to debug and preview the client side of the application as it is being developed. |
| Git and GitHub | Git is a version control software and GitHub is a site to host git repositories. They will be used accordingly. |
| Figma | Figma is a powerful, free design tool and will be used to design UI components. |
| Postman | It will be used to use and test various routes of the API that will be developed. Postman is a great tool to be used while developing APIs. |
| Heroku | Heroku will be used to host the application and the database. |

## 5.3 UML (Unified Modeling Language) Diagrams

### 5.3.1 Use Case Diagrams

### 5.3.2 Activity Diagrams

### 5.3.3 Sequence Diagrams

### 5.3.4 Entity Relationship (ER) diagram

### 5.3.4 Class diagram

## 5.4 API (Application Programming Interface) Endpoints

The server-side of Echo has an API build using the Express framework on Node Js run-time environment. Any HTTP request that resembles <hostname>:<port\_number>/api/\* is forwarded to the associated router and an appropriate controller is called.

### 5.4.1 Authentication

1. POST /auth/signup – Sign a new user up with Echo.
2. POST /auth/signin – Sign a user in to the application and provide a token.

### 5.4.2 User

1. GET /users/:userId – Get details about the user with id – userId.
2. POST /users/address – Set the address of the requesting user.
3. PATCH /users – Update the requesting user.
4. PATCH /users/reset-password – Reset the password of the requesting user.
5. DELETE /users/avatar – Delete the avatar of the requesting user.

### 5.4.3 Shop

1. POST /shops – Set or update the shop of the requesting user.
2. GET /shops – Get shops created by sellers except by the requesting user.
3. GET /:shopId – Get the details of the shop with id – shopId.
4. DELETE /shops – Delete the shop of the requesting user.
5. DELETE /shops/image – Delete the image of the shop of the requesting user.

### 5.4.4 Product

1. POST /products – Create a new product and associate it with the shop of the requesting user.
2. GET /products – Get products posted to the application.
3. GET /products/:productId – Get the details of the product with id – productId.
4. PATCH /products/:productId – Update the product with id – productId.
5. DELETE /products/:productId – Delete the product with id – productId.
6. DELETE /products/:productId/images – Delete an image of the product with id – productId
7. PATCH /products/:productId/images – Add images to the product with id – productId
8. POST /products/:productId/variation-types – Set the variation types of the product with id – productId and having a varied stock.

### 5.4.5 Stock

1. POST /stocks/:productId – Set or update the stock of the product with id – productId.

### 5.4.6 Rating

1. POST /ratings/:contentType/:contentId – Give a rating to the content of type – contentType and id – contentId. contentType may be “product” or “shop”.
2. DELETE /ratings/:contentType/:contentId – Delete the rating of the requesting user to the content of type – contentType and id – contentId.
3. GET /ratings/:contentType/:contentId – Get the ratings of the content of type – contentType and id – contentId.

### 5.4.7 Review

1. POST /reviews/:contentType/:contentId – Provide a review to a content (product or shop).
2. GET /reviews/:contentType/:contentId – Get the ratings of a content.
3. DELETE /reviews/:commentId – Delete the comment (review) of the id – commentId.

### 5.4.8 Reply

1. POST /replies/:reviewId – Provide a reply to the review of id – reviewId.
2. GET /replies/:reviewId – Get the replies of the review of id – reviewId.
3. PATCH /replies/:replyId – Update the reply of id – replyId.
4. DELETE /replies/:replyId – Delete the reply of id – replyId.

### 5.4.9 Subscription

1. POST /subscriptions/:shopId – Subscribe or unsubscribe to the shop of id – shopId.
2. GET /subscriptions – Get the subscribed shops of the requesting user.
3. GET /subscriptions/users – Get the subscribers to the shop of the requesting user.

### 5.4.10 Category

1. GET /categories – Get overall product categories or product categories of a particular shop.

### 5.4.11 Notification

1. POST /notifications – Send a notification from the requesting user to a destination user.
2. GET /notifications – Get the notifications of the requesting user.
3. DELETE /notifications/:notificationId – Delete the notification with id – notificationId.
4. DELETE /notifications – Delete all notifications of the requesting user.
5. PATCH /notifications – Set notifications of the requesting user as seen.

### 6.4.12 Order

1. POST /orders/:productId – Create an order of the product with id – productId.
2. GET /orders – Get the orders made the requesting user or made to their shop.
3. PATCH /orders – Set orders of the requesting user or their shop as acknowledged.
4. DELETE /orders/:orderId – Delete the order of id – orderId.
5. POST /orders/:orderId/completion – Request the completion of the order with id – orderId.
6. PATCH /orders/:orderId/completion – Accept or reject the completion request of the order with id – orderId.

### 6.4.13 Chat

1. POST /chats/:userId – Start chat with the user of id – userId.
2. GET /chats – Get the chats of the requesting user.
3. PATCH /chats/:chatId – Reset the unseen messages count of the chat of id – chatId.

### 6.4.14 Message

1. POST /messages/:chatId – Send a message to the chat with id – chatId.
2. GET /messages/:chatId – Get the messages of the chat with id – chatId.
3. DELETE /messages/:messageId – Delete the message with id – messageId.
4. PATCH /messages/:messageId – Set the message with id – messageId as seen.

### 6.4.15 Account

1. GET /accounts/verification – Request the verification code to verify the account of the requesting user.
2. POST /accounts/verification – Verify the account of the requesting user.
3. POST /accounts/account-recovery – Request the verification code to recover the account of the requesting user.
4. PATCH /accounts/account-recovery – Recover the account of the requesting user by changing the password.

### 6.4.16 Transaction

1. GET /transactions – Get the transactions of the requesting user or their shop.
2. DELETE /transactions/:transactionId – Delete the transaction with the id – transactionId.
3. PATCH /transactions/ - Set the transactions of the requesting user or their shop as acknowledged.

# 

# 8. References

Various articles, documentations, books and courses have been and are to be using for and during the development of the proposed system. The references to these materials are given below:

1. Vercel (date-not-available). Data Fetching Overview [Online]. Available <https://nextjs.org/docs/basic-features/data-fetching/overview>
2. Vercel (date-not-available). Routing [Online]. Available <https://nextjs.org/docs/routing/introduction>
3. Will Sentance (2020, January 7). JavaScript: The Hard Parts, v2 [Online]. Available <https://frontendmasters.com/courses/javascript-hard-parts-v2>
4. Scott Moss (2022, March 15). Build a Fullstack App from Scratch (feat Next.js) [Online]. Available <https://frontendmasters.com/courses/fullstack-app-next>
5. Brian Holt (2022, April 14). Intermediate React, v4 [Online]. Available <https://frontendmasters.com/courses/intermediate-react-v4/>
6. Prisma (date-not-available). Prisma Documentation [Online]. Available <https://www.prisma.io/docs/>
7. Prisma (date-not-available). PostgreSQL [Online]. Available <https://www.prisma.io/docs/concepts/database-connectors/postgresql>
8. Maximilian Schwarzmuller (2022, April). Adding File Upload [Online]. Available <https://www.udemy.com/course/react-nodejs-express-mongodb-the-mern-fullstack-guide/>
9. Tailwind CSS (date-not-available). Utility-First Fundamentals [Online]. Available <https://tailwindcss.com/docs/utility-first>