<GROUP 03>

<SHOES PURCHASE>
Software Architecture Document

Version <1.5>

Revision History

Date	Version	Description	Author
<27/11/2022>	<1.0>	- Make basics: + Introduction + Architecture goals and constraints	 - Phan Thanh Lap - Nguyen Duc Minh
<30/11/2022>	<1.1>	- Logical view for Front-end - Logical view for Back-end	- Nguyen Minh Tuan - Ngo Van Trung Nguyen
<05/12/2022>	<1.2>	- Add Deployment diagram - Add Implementation View	- Vo Van Hoang - Nguyen Duc Minh
<08/12/2022>	<1.3>	- Add missing description of diagrams - Fix something in Use-case model	- Vo Van Hoang - Phan Thanh Lap
<22/12/2022>	<1.4>	- Do UI prototypes - Do deployment part - Do implemetation view	- Vo Van Hoang - Nguyen Minh Tuan - Ngo Van Trung Nguyen
<24.12/2022>	<1.5>	- Check and fix UI prototype file - Check and fix deployment and implementation view file	- Nguyen Duc Minh - Phan Thanh Lap

Table of Contents

Software Architecture Document

1. Introduction

1.1. Purpose:

This article provides a comprehensive overview of the system and illustrates various system components from various perspectives. Its goal is to document and communicate the important system decisions that have been made.

1.2. Scope:

This page provides an overview of the ‘Shoes Purchase’ design, including its most basic components and behaviors, to assist users in navigating the application and comprehending the project as a whole. Stakeholders who require a technical understanding of the Process Specification tools should begin by reading this paper, then proceed to the UML model, and then revisit the Process Specification Tools UML model's source code.

1.3. Definition:

This software development plan includes the following details: Project Overview: Describes the project's goals, scope, and purpose. It also describes the products that the project is expected to produce. Project Organization describes the organizational structure of the project team.

2. Architectural Goals and Constraints

Some key requirements and system constraints have a significant impact on the architecture. They are as follows:

- Websites are accessible via internet-connected
- The login system must ensure that data is completely protected from unauthorized access. All access is controlled by user identification and password.
- All performance and loading specifications specified in the Vision Document.

3. Use-Case Model

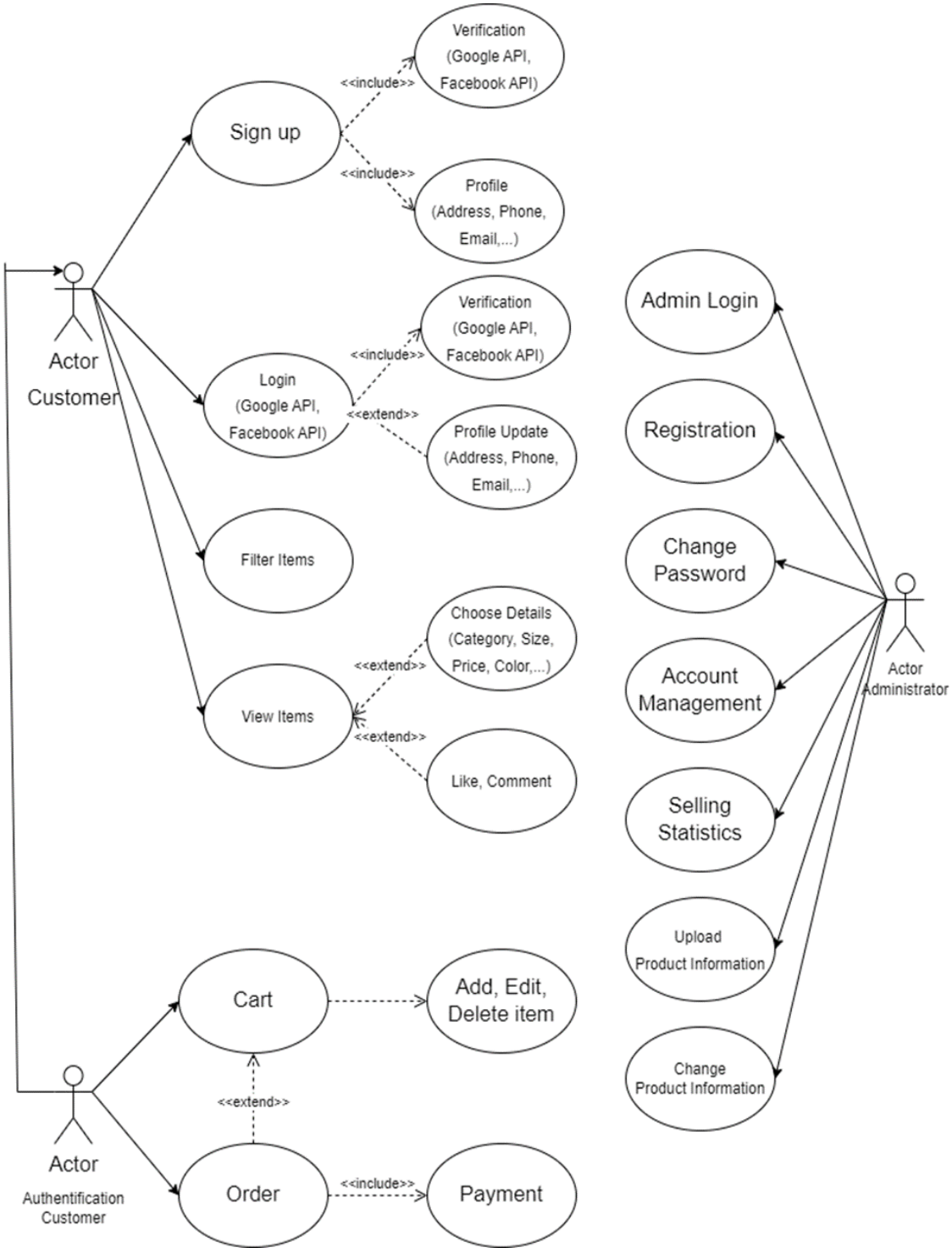
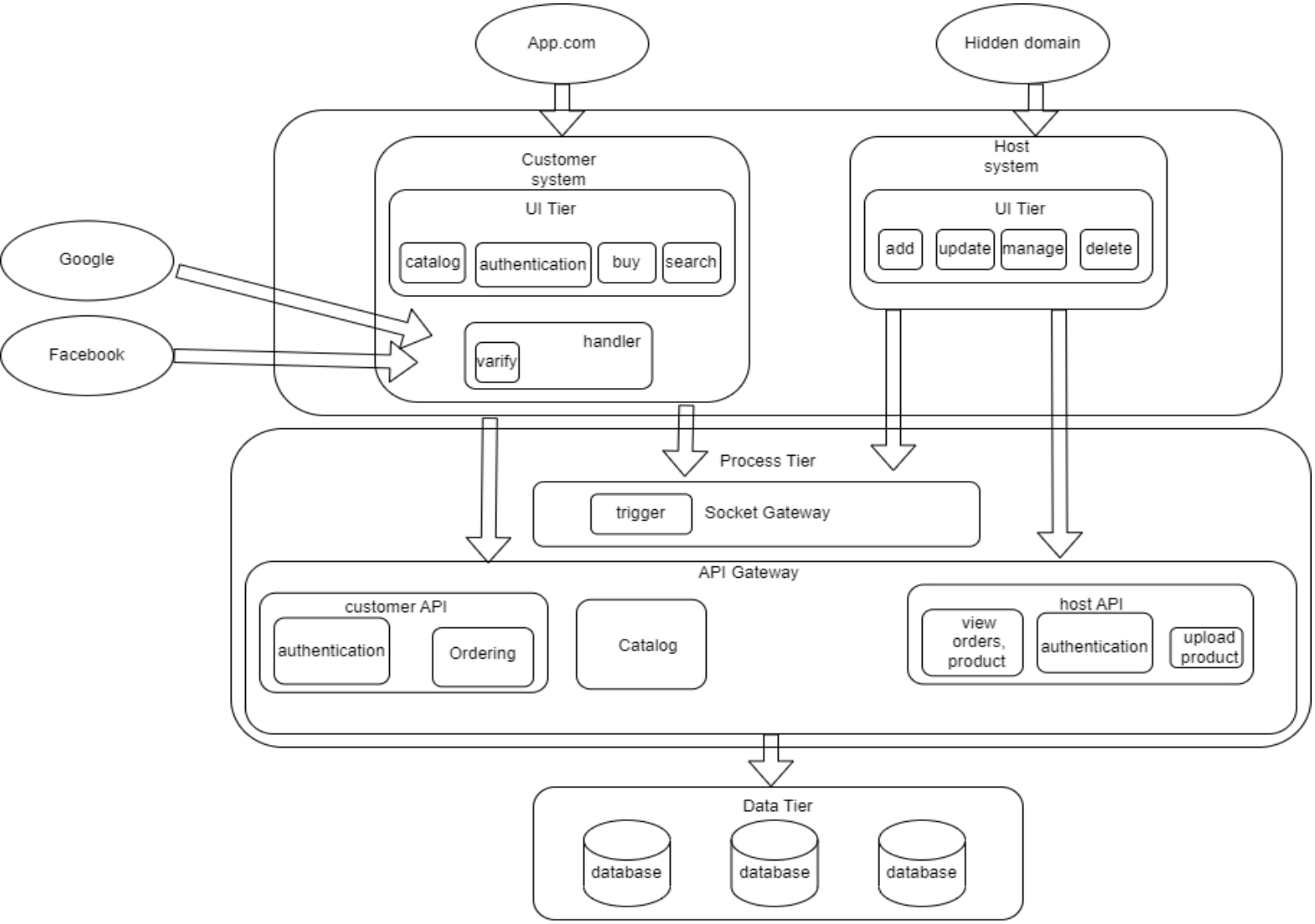


Diagram of Use-case Model

4. Logical View

In general, our website consists of frontend and backend part, they communicate with each other through APIs, database is stored and processed on , we use Google drive to store images of the web, as well as socket which is used to trigger special events.

4.1 General:



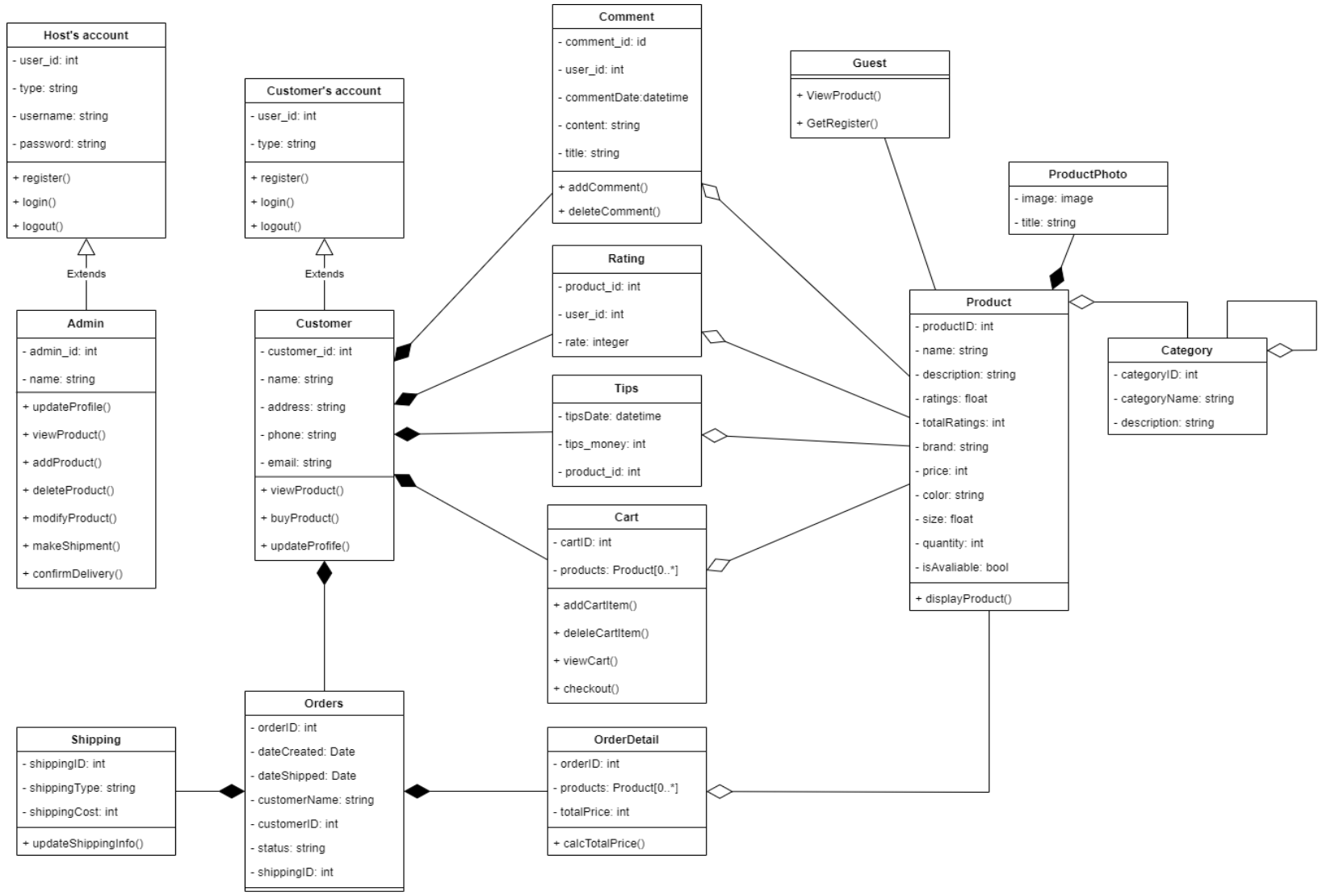
Note:

Catalog has the feature to use for both user and host

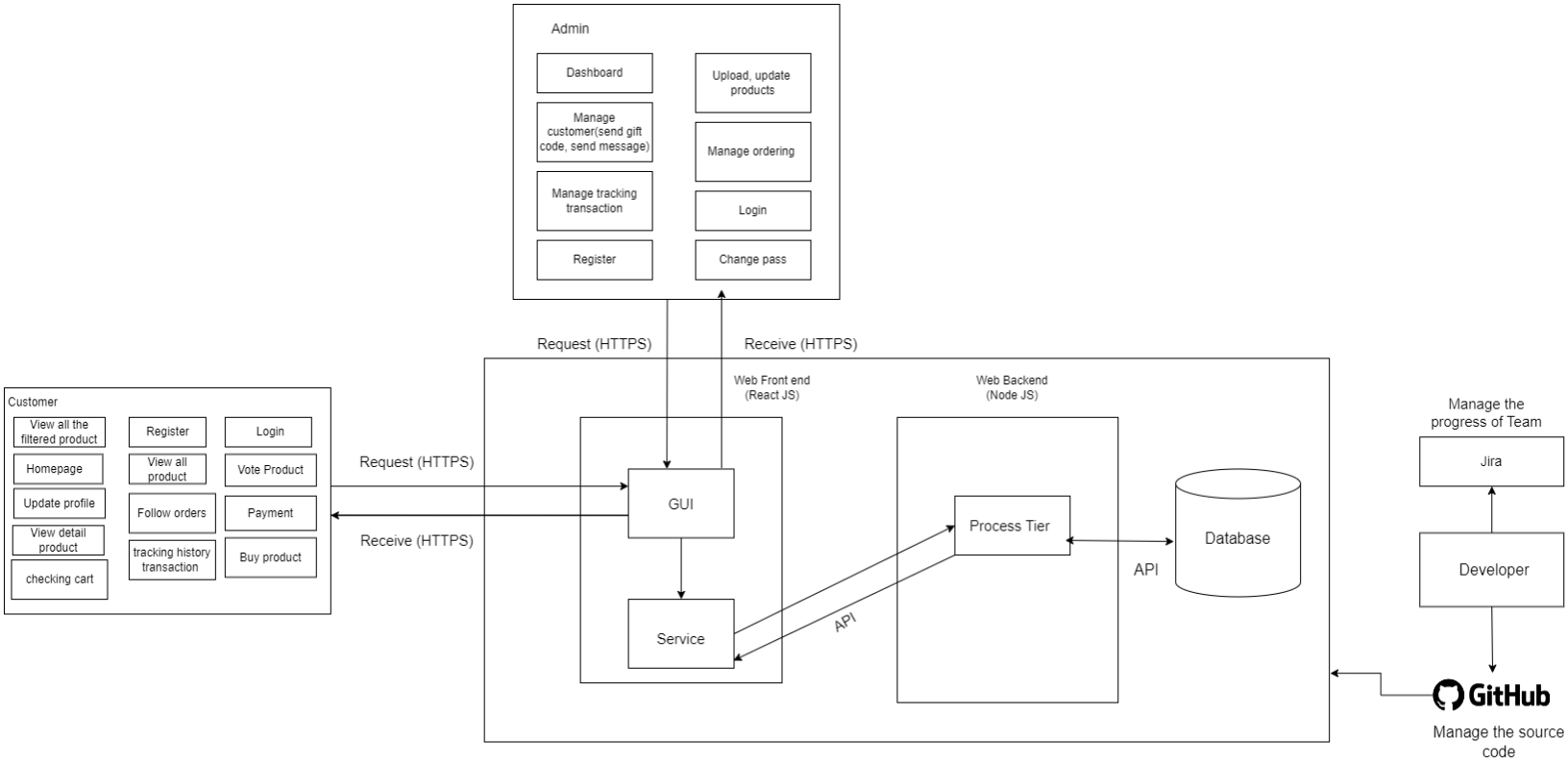
Sub-class handler has the feature when Google or Facebook verify successfully, the data will be saved in our database

Socket Gateway: listen any events from customer when they buy products or comment,...then update state's host (e.x: notice customer had bought products). Otherwise, listen from the host when the host change or update something in order to update data of customer.

4.2 Online shopping system:



5. Deployment



Deployment diagram for SHOES PURCHASE

General process:

- First of all, developer team will make the changes to the websites and make sure no errors founded, after that push it to github, at that time, all the team can see and use the code together.

Nodes:

- Node 'Developer': Represent developers of this website, in this case is all the member of group 3
- Node 'Process Tier': Take responsibility for handle the requests from customers as well as admin, such as query data from database and return the responses to them
- Node 'GUI': It is a group of UI components where the users can access all the web pages. In my project, users can access directly to the screens without needing the responses from the backend, my backend returns data only.
- Node 'Service': Where listen the events on the UI from users and then make the requests into the backend and receive the response from it later.
- Node 'Jira': a place to describe the progress and phases of each member in a team which can help team control the work more effectively.

- 'API':
- 'Database':

6. Implementation View

customer: customer UI

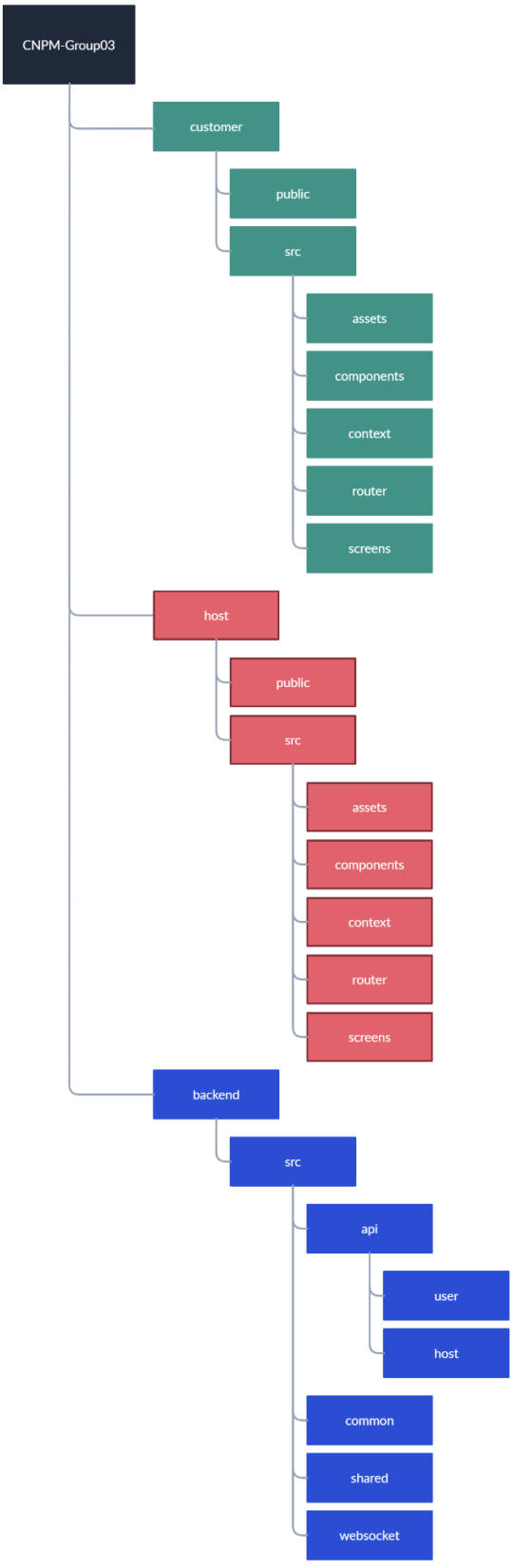
- src: contains source code.
    - assets: contains image file.
    - components: contains customer folder's components (header, footer, navigation bar, …).
    - context: save global data.
    - router: contains endpoint.
    - screens: contains customer's screens (cart, Product_details, notifications, …).

host: host UI

- src: contains source code.
    - assets: contains image file.
    - components: contains users folder's components (addProduct, updateProduct, checkList, …).
    - context: save global data.
    - router: contains endpoint.
    - screens: contains host's screens (manage, statistic, …).

backend: backend code

- src: contains source code.
    - api: contains user and host file.
        - user: manage customer's endpoint, handle api, interact with database (query).
        - host: manage host's endpoint, handle api, interact with database (query).
    - common: contains middleware and env file.
    - websocket: manage socket channel.
    - shared: contains typeORM config.


CNPM-Group03
customer
public
src
assets
components
context
router
screens
host
public
src
assets
components
context
router
screens
backend
src
api
user
host
common
shared
websocket