
# COMP3900 Computer Science Project

# Final Report

## V-Gifts

### 3900-H16A-Victims

**Scrum Master:**

**Haoran Lyu** z5045262@ad.unsw.edu.au

**Developers:**

**Jie Mei** z5173405@ad.unsw.edu.au

**Yifan Wang** z5175023@ad.unsw.edu.au

**Chenkai Lyu** z5192710@ad.unsw.edu.au

**Pai Qu** z5299383@ad.unsw.edu.au

**23rd April 2020**

# Table of Contents

# 1. Overview

## 1.1 Introduction

E-commerce earned its first glance from the world at the end of last century (Cohen, 2003). Although it was much insignificant compared to its scale today, the pioneers, Amazon and Auction Web (predecessor of eBay until 1997), made their first attempt in 1995. Until today, the form of e-commerce has become much broader and adaptive. Almost every commercial good from dairy products to industrial materials, and even infrastructures have been covered by those famous enterprises and brands, including Amazon, eBay and Alibaba. However, for some of less populated products, small e-commerce companies could still earn trust from potential, but distant, customers.

Regardless the type of good is selling, working e-commerce companies still need websites to pair with their warehouses and delivery systems. And a good website for e-commerce provides the user with pages of different products available for purchase in the online store, as well as suitable recommendation systems, which extends the shopping window for each customer. In the long-run, extended shopping windows will not only raise profitability, but also helps in analysing customer personal preferences for further precise recommendations.

For e-commerce companies, the web operator, an admin-friendly website is also valuable for both web maintenance and development. Firstly, being able to import product information both individually and in batches from external files could massively improve web setup efficiency. Then, being able to batch editing existing product details allows a clear big picture view, together with convenient product management. Last but not least, order history from customers can be collected in the database, and ready for exporting or future analysis.

## 1.2 Our Product

Our Product e-commerce website provides various functions for user and admin. Standard utilities, including posting products and buying products, are polished for smoother experiences comparing to modern websites. Moreover, our product offers more than standard functionalities. Several additional features are provided for users and admins, to ensure higher management efficiency and better user-product match making.

For admins, a standard overview of existing product, past orders are collected for further analysis, together with product editing functions, which allows modification of product information without clearing past user ratings and past orders. Advanced functions, including batch product export and import makes database rebasing much less labour intensive.

For users, standard product searching, and traversing experiences are provided, while advanced user-product match making is specialized for individual user already logged in. Upon user login, personalized products recommendations are automatically pushed to individual user in homepage. Additionally, upon each successful purchase, future recommendations will shift towards dynamic interest of individual user. Such features would significantly increase the chance of purchasing corresponding products.

There is a chatbot AI names Victor for users. Victor could have small talk with users, like greeting and thanks. Victor is designed to recommend popular gifts and implement instructions for “forget password”. In detail, Victor will ask users who they want to send the gift for, after receiving the response from users, Victor will provide different types of gifts to users, and user can click the link to see detailed gifts information which also include a link to the product page.

To summarize, the bullet point objectives of our entire project are shown as below:

- User: Product Recommendation System
- User/Admin Register & Login
- Admin: Product Information Management

- Admin: Order History Readability and Analysis
- User: Product View & Selection
- User: Chatbot AI
- User: Product Purchase & Order Payment
- User: Order History Management

## 1.3 System Architecture

Our product e-commerce website has a following system architecture (Figure 1). Users and admins gain access to their web page from a computer, through a browser which supports JavaScript. Their actions, including searching for products and organizing orders, are passed to backend by components implemented in frontend. Then, the results requested by caller shall be passed back to and displayed by frontend.

Frontend is coding based on the React framework. Use third-party libraries to beautify the pages and make each page modular. For example, in the navigation bar, we can reuse different pages on the front end, and only need to transmit some parameters to this component to make changes to its internals. This is easy to maintain for the entire front-end structure.

The communication between frontend and backend is achieved through Flask API by binding a series of backend functions with a URL route. For example, an admin can receive all products list from http://localhost:${BackendPortNumber}/admin/all_order. Therefore, the load would be shift from frontend webpage to backend server. By only letting backend server to process large amount of data request can greatly reduce the response time of webpage and reduce memory usage of browser. Meanwhile, backend server can isolate individual requests to maintain the integrity of database at any moment, to make database access being exclusive while writing action happens.

The backend focuses on processing requests, manages access to database and communicate with online API services. Python is used for faster feature implementation, broader API access and better compatibility with flask. Several features utilize complex mathematical equation and

theory. Using APIs, such as Numpy, greatly reduces the difficulty in calculation. Other APIs, like DialogFlow, allow our server to use google cloud services for Chatbot AI. They ease the effort in building natural language processing algorithm, and help our product focusing on selling products to our customers. The backend server also has frequent access to the database, to reduce the complexity of database updating, a *.json file is used to store the changes we made.

Our products have been tested on Windows 10 and Mac OS, it has shown an excellent stability and a fast average response time. That is to say, our product can provide users with fabulous experiences, while advertising the most fit products to the right potential individuals.

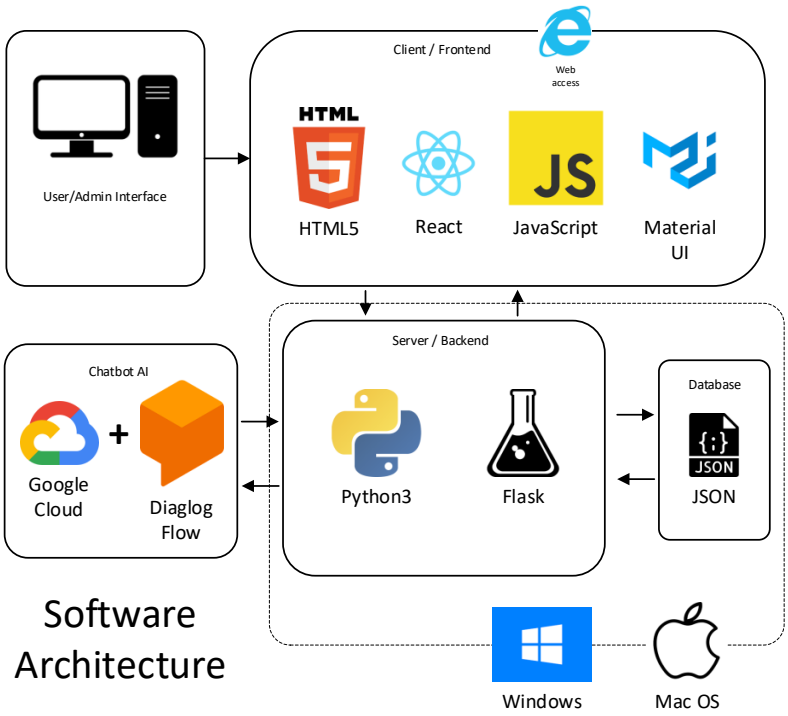



*Figure 1: System Architecture*

# 2. Third-Party APIs and Frameworks

Due to a very limited develop period allocated for our project, the development on some functionalities with high complexity would significantly delays the progress. Those functions comprise HTTP Exceptions, linear arithmetic from Numpy and DialogFlow Chatbot AI from google cloud. With APIs above, backend development became smooth and much less intensify. The same idea would be applied to frontend as well. Today, most frontends no longer simply rely on the html generated simply by backend. One reason would be the massive data on each single webpage, while another would be aesthetics need for eye-catching effects. Therefore, frontend frameworks, such as Material UI, React are used to provide basic webpage structure, and common webpage elements, including tick boxes, search bar, paging, scrolling and etc. In this part, features and usages of third-party APIs and frameworks will be discussed, with examples we met during product development.

## 2.1 Frontend

### 2.1.1 React

React is a JavaScript library for building user interfaces. Our frontend is a React app that imports several packages such as Material UI and Axios. Since every React component manages its own state and supports rich data passing, we were able to handle the data passed from the backend efficiently

### 2.1.2 Material UI

Most of our stylings are based on Material UI, a component library with numerous components. Material UI has a very informative and clear documentation for its usage. Using Material UI has helped us save a large amount of time from building every component from scratch. Because of it, we were able to create responsive components effectively and spend enough time on the UI&UX design.

### 2.1.3 Axios

We used Axios to handle different HTTP methods from our frontend to our backend. Axios is a promised based HTTP client. It is mainly used in two scenarios:

POST/GET requests: Send data to the backend in the correct format.

Error handling: By checking the status code of data passed from the server in the ".then" promise, we are able to catch the error and respond quickly.

## 2.2 Backend

### 2.2.1 Virtualenv

Virtualenv is a tool for creating isolated Python environments containing their own copy of python, pip, and their own place to keep libraries installed from PyPI (Matthews, 2013). It is an open-source framework and designed to allow developers to work on multiple projects with different dependencies at the same time on the same machine.

For our project, we use this to allow our team members and test users to quickly set up the packages needed in the backend (by using provided requirement.txt) without interfere system wide environment.

### 2.2.2 Werkzeug

Upon a successful response from backend server to frontend, a 200 code is returned. However, for requests failed to respond, a default 400 will be returned, if handled by backend. Therefore, to distinguish different illegal responses from frontend, a method of implementing a number of standard non-200 responses is required.

In our project, during regular operation, we needed to handle login failure, invalid authentication, illegal text input and etc. Our own custom HTTP status codes used are shown below (Table 1).

*Table 1: Custom HTTP status code*

| CODE | HTTP STATUS |
|---|---|
| **460** | InvalidToken |
| **461** | InvalidUsername |

| | |
|---|---|
| **462** | InvalidEmail |
| **463** | UsernameAlreadyExit |
| **464** | IncorrectUsername |
| **465** | InvalidPassword |
| **466** | NotEoughFund |
| **467** | InvalidInputID |
| **468** | EmailAlreadyExit |
| **469** | NoFile |

- InvalidToken exception is raised during authentication process, if user attempts to directly sending request to server and fails to input a valid token, which is bond with one's identity.
- InvalidUsername exception is raised during name text editing. It prevents user from inputting potential special character combinations, which can harm backend services.
- InvalidEmail exception is raised if the email entered doesn't match a standard email address format.
- UsernameAlreadyExit exception is raised while the account name entered by a user collides with existing account names. It can ensure uniqueness of a user, for better backend management.
- IncorrectUsername exception is raised if an invalid username is entered upon login. It prevents user from repetitively checking password, while actually the account name is invalid.
- InvalidPassword exception is raised if the password entered is not matching with the hashed password save in system, which prevents account leakage.
- NotEnoughFund exception is raised if the total price of products purchasing exceeds the total fund left of a user. It stops the action from completion and suggest user to top up before further purchasing.
- InvalidInputID exception is raised if the input ID is not found in database, and stop any changes with that invalid ID.

- EmailAlreadyExit exception is raised if an email is already taken by a user, to prevent email sharing between accounts.
- NoFile exception is raised if no file is provided upon new product creation (both import manually or from csv).

Werkzeug is an open-source and comprehensive WSGI web application library, with a BSD-3-Clause License. Therefore, redistribution and use of it are permitted in our project, as long as following criteria mentioned in BSD-3-Clause License are met:
Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
Neither the name of the NumPy Developers nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

### 2.2.3 Numpy

Numpy (Harris, C.R., Millman, K.J., van der Walt, S.J. et al., 2020), a math library of python, is famous of its computational power among scientific studies. In our project, some algorithms are implemented in a way which comprising massive linear algebra calculations. With basic python arithmetic, those calculation would cost significantly longer, and take much larger memory. With the help of Numpy library, the response speed and space complexity of our project would be improved enormously.
One of the most important features in our project is the automatic recommendation algorithm. It relies on Numpy to perform a fast matching between the portrait of a user and characteristics of all products. Firstly, we use information retrieval technics to extract the scalar of a product on different category dimensions. Then, we use Numpy to quickly perform an angle calculation

between two large vectors for hundreds of times. After obtaining all data necessary for ranking, we could finally find the best matching between one user and several products.

A simplified example is shown below (Figure 2). In a small database of one user and two distinct products, the category dimensions of each product are shown as vectors end with solid ball head, while the interest of a user is shown as a vector end with hollow ball head. The algorithm compares the angle between the product vector and user vector and finds the closest products to be put in daily recommendations. If the calculated angle between product B and user is the smallest, product B would be found at the top of entire recommendation list.

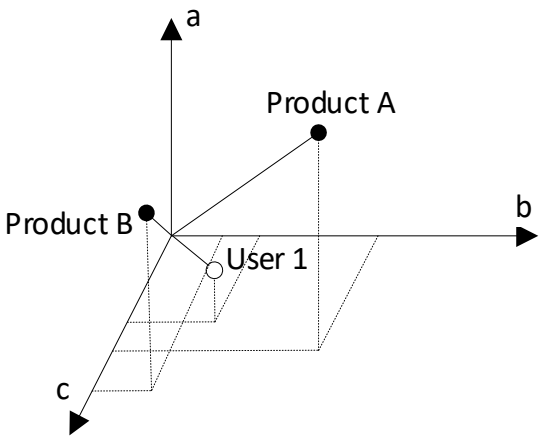



*Figure 2: Simplified recommendation algorithm example*

Numpy itself is an open-source library, with a BSD-3-Clause License. Therefore, redistribution and use of it are permitted in our project, as long as following criteria mentioned in BSD-3-Clause License are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the NumPy Developers nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

### 2.2.4 Json Library

JSON (JavaScript Object Notation) is a lightweight data-interchange format (Introducing JSON, 2021). It is especially human friendly and allows storage of hash table. Combining with Python, dictionaries can be easily dumped into JSON files without building SQL database. More importantly, JSON format supports a variety of types, especially array of elements. It allows dynamic length of lists to be stored. With the support of JSON library, our project is able to read and write database quickly and to provide a fast response time for every http request.

JSON format is open-source and allows modification and distribution. Therefore, the licensing will not become a problem in future product development.

### 2.2.5 Flask App

Flask is a micro WSGI web application framework written in Python (Grinberg, 2018). It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. It supports the communication between frontend and backend through HTTP request.

This is an open-source framework. Therefore, the licensing will not become a problem in future product development.

### 2.2.6 Dialogflow

DialogFlow is a natural language understanding platform that allows you to easily design and integrate conversational interfaces into your mobile applications, web, chatbot and so on. Dialogflow can analyze the input of various users and provide an optimal response (Figure 3). In our project, we use Dialogflow to support our chatbot. Intents and entities in Dialogflow give us the way to teach chatbot AI how to communicate with customs.

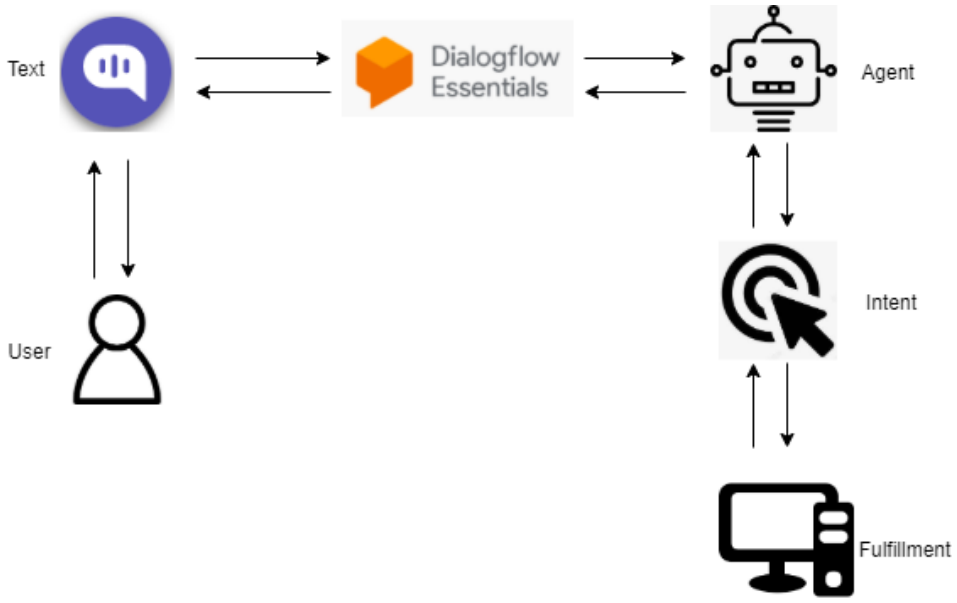



*Figure 3: Simplified chatbot system example*

For larger scale applications, DialogFlow free edition may restrict the performances due to limitation on response frequency set by Google Cloud. Therefore, either payment is required or our own DialogFlow must be developed to support future product development.

### 2.2.7 Kommunicate

Kommunicate is live-chat and chatbots powered customer support software (kommunicate, 2021). In our project we use google cloud platform to integrate our Dialogflow chatbot into Kommunicate. Kommunicate provide a beautiful, customized widget.

For larger scale applications, a static domain may be required for better performance and stability. Therefore, we have to pay for the static domain at its website. As a result, either payment is required or our own Kommunicate app must be developed to support future product development.

# 3. Functionalities

Functionalities in our project focuses on providing conveniency for roles involved in online trading. The two roles are usually the seller and buyer. However, in an e-commerce website, admins take the role of seller, while users are the potential buyers. The admins who operate from the back are supposed to manage the product user would see and to access all orders users have made. The users should be able to join the website as members, prior to making any purchase. They should also be recommended with products which matches their interest. They should be able to keep desired products within shopping carts, before pressing the pay button. To conclude, our project implemented many functionalities to provide convenience for both admins and users. In this part, different functionalities will be introduced with details together with examples met during product development.

## 3.1 Frontend

### 3.1.1 Admin Interfaces

The interface webpages written for admin provides admin with functionalities corresponding to the product design objectives below (Figure 4).

- Admin: Product Information Management
- Admin: Order History Readability and Analysis

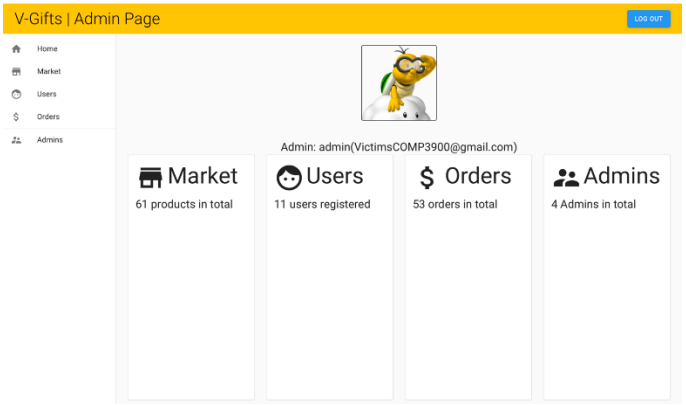



*Figure 4: Admin Interface*

Admin can manage the website by logging via a “secret button” located at the bottom-left corner of the login page. After logging in, admin will be directed to the Admin Home page where Admin can see the overview of the website: the number of products, users, orders and admins in the system. On the left side, there is a side menu of buttons: “Market”, “Users”, “Orders” and “Admins”. By clicking on any of these buttons, admin can manage the corresponding category in a page with more details.

- Market: All products will be shown to the admin. Admin can also add a new product, edit an existed product, and import several new products from a csv file.
- Users: The personal details of every users are shown to the admin in a table. Admin can sort the table by using any column as a key.
- Orders: All placed orders are shown to the admin. In this page, the admin can modify the current state of order to another.
- Admins: Admin can inspect all admins of the website, and add a new admin by entering its username, email and password.

Admin can also logout through the “log out” button at the top-right corner.

### 3.1.2 User Interfaces

The interfaces written for users provides users with functionalities mentioned in the product objectives below (Figure 5, Figure 6).

- User: Product Recommendation System
- User/Admin Register & Login
- User: Product View & Selection
- User: Chatbot AI
- User: Product Purchase & Order Payment
- User: Order History Management

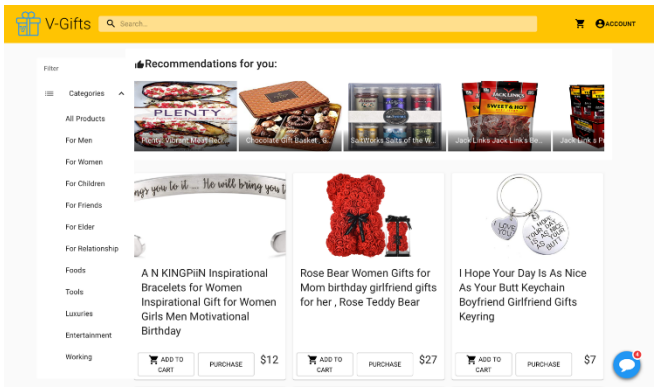



*Figure 5: User Homepage Interface*

User can register via multiple buttons on the website (home page, login page and market page), after filling up the requirement information, user can choose to select their interest so that they can receive better recommendations in the future. After registration, user will automatic login and redirect to market page, then the user is able to purchase products or add products into their cart. User can add multiple products into their cart and checkout selected products by clicking cart button located at top-right corner of market page. User can also chat with a little chatbot by clicking the chat icon at the button-right to start conversation.

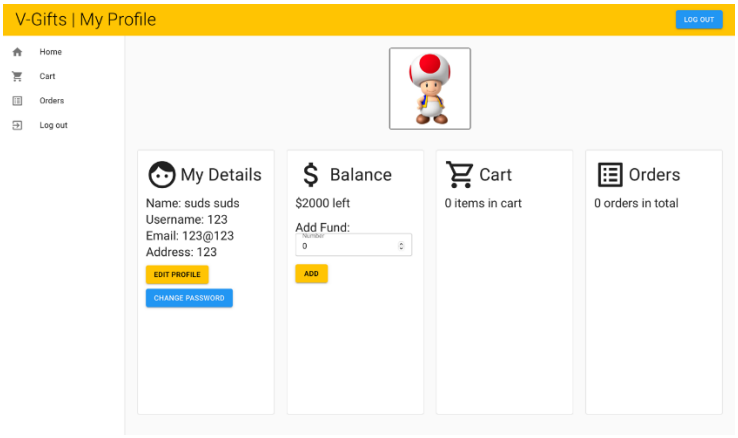



*Figure 6: User Profile Interface*

By clicking the “ACCOUNT” button located at the top-right corner of the market page, user will redirect to their profile page. In their own profile page, on the left side, there is a side menu of buttons: “Home”, “Cart”, “Orders”, “Log out”. By clicking on any of these buttons, the user can do the corresponding actions in a page with more details.

- Home: User can edit their profile details and password, user can also add the fund in this page and view the brief information of their orders and cart.
- Cart: User can change the amount of products that already in the cart and purchase one item or all of them on their will.
- Orders: User can check their orders details and “refund” or “receive” their orders depends on different order state. User can also rate the order that in the order list.
- Logout: Just simple logout the user.

## 3.2 Backend

### 3.2.1 Database Management

In our project, permanent data storage, although doesn’t fulfill any design objectives directly, is the foundation of all design objectives mentioned in the beginning. All permanent data are stored within a readable and writable database along with the server. Permanent data comprises admin accounts, user accounts, products and orders made by users. They persist between server shutdown and server restart. This feature helps to reserve users’ information during system upgrade and maintenance.

Permanent data are stored in a way similar to SQL entity relationship diagram (Figure 7). Object tracks related objects, and saves space storing relations. Admins have access to the overview of entire databases, while users have access to product in shopping cart, their own orders created. And orders can be backtracked to the user and product purchased.

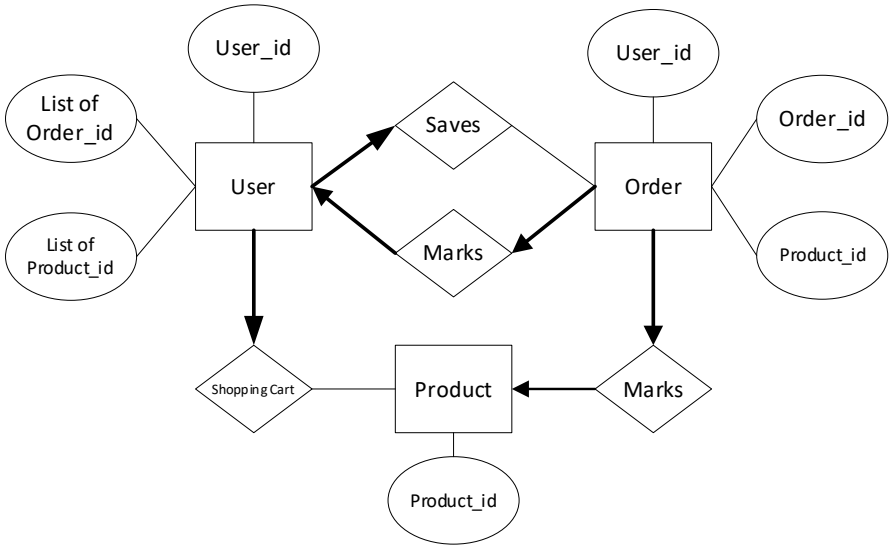



*Figure 7: Entity Relationship Diagram*

Users, products and orders have their own serial ids to distinguish between each other. Their relationships form the structure shown in Figure 7. This structure allows each user to keep a list of any number of order_id and a list of product_id in their own shopping cart. While each order keeps the user_id and the product_id, users can use queries to collect all past products they have bought.

In our database, different entities are separately stored in different hash table, and having their id as hash keys. Relations of each entity are stored by keeping a copy of id of other entity as one of its attributes. Therefore, it becomes efficient in looking for entities, without traversing entire database.

### 3.2.2 Admin Operations

In out project, admin roles are responsible for managing product information presented to users. Therefore, an admin should be able to login, import products, edit products, view order histories made by user and etc. to match following design objectives.

- Admin: Product Information Management
- Admin: Order History Readability and Analysis

Order histories, as discussed in previous part, has their own ids, user_id and product_id for better entity relationship maintenance. Besides that, every order keeps track the timestamp of when it is made. Also, a delivery state is kept along with the record (Table 2).

*Table 2: Delivery state*

| # Code | Delivery State |
|---|---|
| 0 | Just paid |
| 1 | Start delivery |
| 2 | Delivery Done |
| 3 | Cancelled |

Upon creation of each order, the delivery state is set to Just paid (#0) as default. User can still refund to cancel the delivery. After once admin triggers the deliver action, and set state to Start delivery (#1), refunding will be disabled. If the order is refunded by the buyer, the order state becomes Cancelled #3, otherwise, the order state will become Delivery Done (#2) in the end.

Besides having read and partial write access to orders, admins have almost full control to product information. For each product imported into database, admin should fill in the name, description, delivery days, price and sample picture. However, some product attributes are not available for direct editing, including user ratings and product categories. This mechanism restricts the admin permission and protect users from fake products and manipulated ratings.

For large scale backend operations, such as rebasing product and batch importing products, admin can import products from a CSV file, which contains hundreds and thousands of products. Upon detecting invalid product information, the entire operation will be cancelled, the error position will be spotted, and the database will also be rolled back. This mechanism protects database from mis-operations and helps operator to identify the position of error efficiently. Also, a mirror function, which exports database into CSV file, is also provided for product rebasing.

### 3.2.3 User Operations

In our project, user roles are defined to perform product purchasing actions. They should be able to view products, search products, put products into shopping cart, make payments and give ratings. Making above functionalities available fulfills following design objectives:

- User: Product View & Selection
- User: Product Purchase & Order Payment
- User: Order History Management

Upon registering, user have to enter their basic information, such as addresses, to make delivery possible. After finishing registering or logging in, users are redirected to the homepage. At homepage, users are provided with a normal search bar and tick boxes for advanced product searching. If users want to purchase a product, they can select the amount they want and choose between immediately paying and putting into shopping cart. Upon purchasing, users are supposed to have enough fund to create orders. If the fund is insufficient, users will be promoted to add fund to their account. Otherwise, orders will be created as described previously.

Users also have access to all orders they have made in the past. Users can give ratings to each product from the order list. And then, an average rating from all buys is shown at product detail page as a guide for other users. Users also have options to refund if a product is not delivered yet. If they do so, the currency will be added back to their fund. However, the order will not be deleted, so that user may purchase it again in the future.

### 3.2.4 Product Recommendation

Ideally, it is impossible to assume any user to go through the entire product list within a limited time, especially before a user loss his or her interest and patience. Therefore, our product designed a method to help users to quickly approach products they may potentially prefer. This method fulfills the following design objective:

- User: Product Recommendation System

This method is called auto product recommendation. It attempts to match users with the products have the closest characteristics, as discussed in Numpy usage and in Figure 2. However, this method requires additional steps to process products and users. Firstly, users are invited to a small survey to pick attractive categories. Then, upon product importation and editing, the characteristic dimensions of products are calculated. And finally, the calculated values will be compared and ranked to obtain the top-N best matches.

For users keeping using our product, their interest may vary, comparing to the initial setup. Therefore, our product uses an advanced strategy to keep our estimation and recommendation up to date with user. This strategy is to adjust the interest vector of user by averaging a number of most recent products with current interest vector. It would keep user feeling refreshed each time after login and purchasing. A visual example is shown in Figure 8. The shift of example user-vector shows the adjustment after the purchase made by user. After purchasing the Product-A, the user-vector becomes closer to it. The result recommendations in the future would likely to push more products closer to Product-A towards the top-N product in recommendation area at homepage.

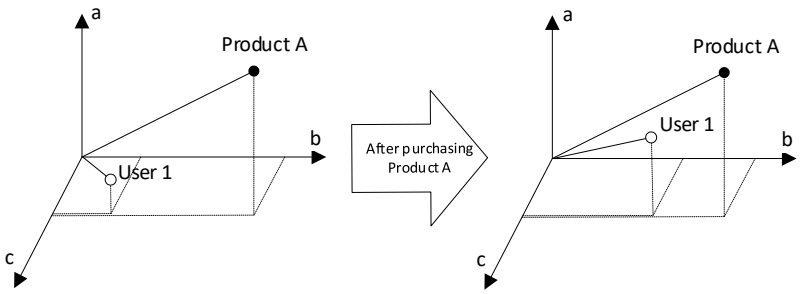



*Figure 8: Effect of purchasing product on user interest vector*

### 3.2.5 Chatbot AI

In our project, we chose to build a Chatbot AI as novelty. The Chatbot AI is designed to inspire the needs of a user and finally recommend our products based on user's needs. It fulfills the following project design objective:

- User: Chatbot AI

The flowchart of a Chatbot AI working flow is shown as below (Figure 9). Upon basic greeting, the Chatbot AI will attempt to acknowledge whom the user is gifting to. Then, the Chatbot AI can filter the product list and gives further suggestions on gift category, such as food, entertainment and etc. If the user clicks on an of them, further suggestions of products will be given to the user. In the progress, if the Chatbot AI fail to progress the conversation, it will go back to the initial greeting and restarting recommendation procedure.

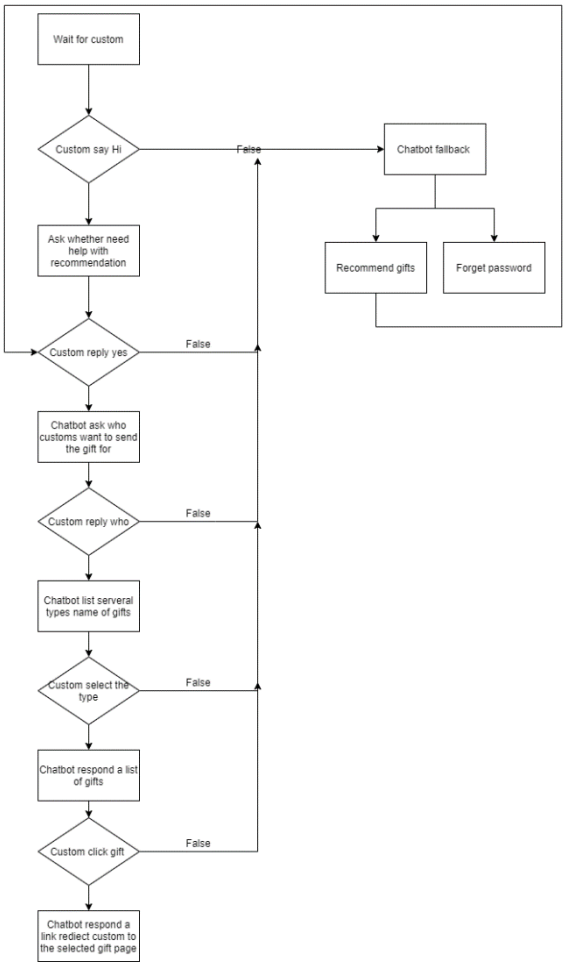



*Figure 9: Chatbot AI flow diagram*

# 4. Implementation Challenges

Our project has been completed in time and been polished well. However, our project didn't go as smooth as this report does. There were several segments and aspects making us inspect, researching and debugging for days and nights. In this part, challenges we met during our project development will be introduced together with our unique solutions.

## 4.1 Transforming Product Description into Category

One problem we were facing during matching product with specific user was that we have to define the category of every product manually. However, such way is neither applicable in large database, nor used anywhere in modern industry. Therefore, we came up with an ideal to find out the possible category of a product by analyzing its description and name. By setting up a large keyword list and category list, we applied information retrieval technic to find the combined category of every product.

However, we also found some products has a huge chunk of description with repeat keywords. So, we need a way to decrease the accumulated gain in keyword weighting for repeated keywords. Therefore, we chose to add a function to diminish the gain and smoothing the effect from keyword spamming.

## 4.2 Frontend Framework

One problem is that after receiving the backend data, we perform operations on the frontend, such as buying and adding products. We need to update our interface in real time. This is a difficult point for us. We set a variable to transmit between components. If an operation affects the display of a certain component, we change this variable to control useEffect() Read the data renderer interface again.

# 5. User Documentation and Manual

## 5.1 Installation of prerequisite environments and packages

Please not the if you need run our project in Windows system (at least Windows 10), you need to have a linux bash terminal to continue.

### 5.1.1 Backend

Go to https://www.python.org/downloads/ and install the latest python3 on your computer, then run command below after you successfully install python3.

```
python3 -m pip install virtualenv
```

Then go to "/backend" sub-folder and run command below.

```
cd where/the/project_code/project/backend
python3 -m venv env
```

After this stage, you should have a folder called "env" in your backend folder, then run the command below and active the virtual environment.

```
source env/bin/activate
```

At last, run command below to install the required packages in your virtual environment.

```
python3 -m pip install -r requirements.txt
```

### 5.1.2 Frontend

Go to https://nodejs.org/en/download/ and install the latest Node.js on your computer, then go to the "/frontend" folder and install all required dependencies

```
cd where/the/project_code/project/backend
npm install
```

## 5.2 Starting Server

### 5.2.1 Start the backend server

Go to the "/backend" folder and active the virtual environment if it is not been active yet.

```
cd where/the/project_code/project/backend

source env/bin/activate
```

To active the server in the backend, run command below in the backend folder.

```
python3 server.py # start the server in default port (5000)

# or

python3 server.py [PORT NUMBER] # example: python3 server 6000
```

### 5.2.2 Start the frontend server

Go to the "frontend" folder and start the frontend server use command below after the backend server has been running.

```
cd where/the/project_code/project/backend

sh run.sh [BACKEND PORT] [FRONTEND PORT]

# backend port number needs to be the same as what's used in the backen
d server and cannot be the same as the frontend port number
```

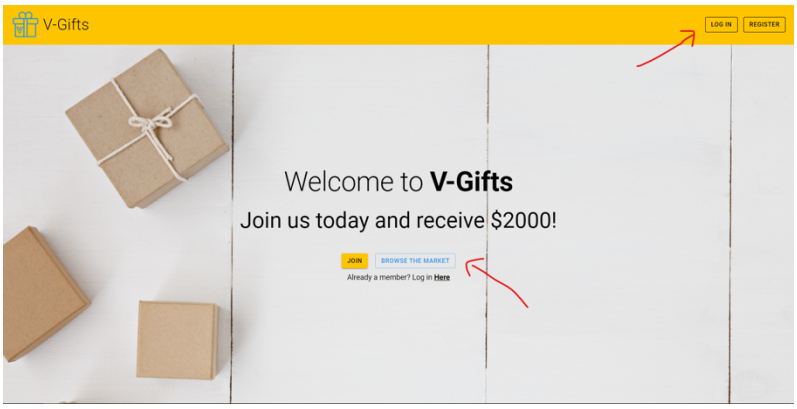
example:

```
sh run.sh 5000 3000
```

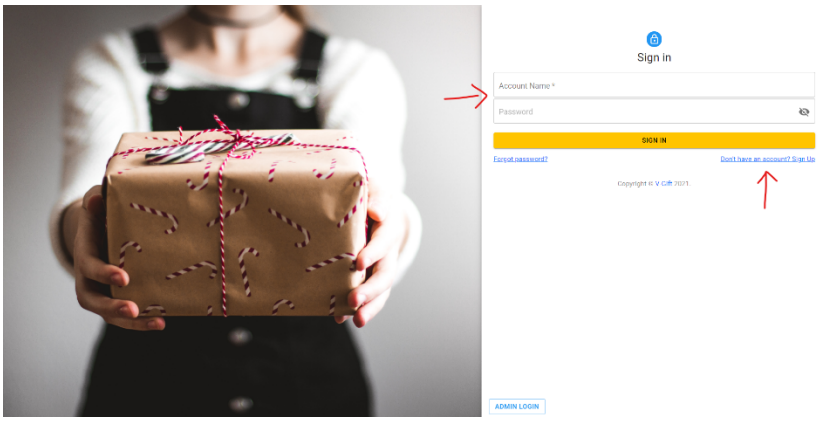
## 5.3 Using Product as User

### 5.3.1 Login / Register/ Forget Password

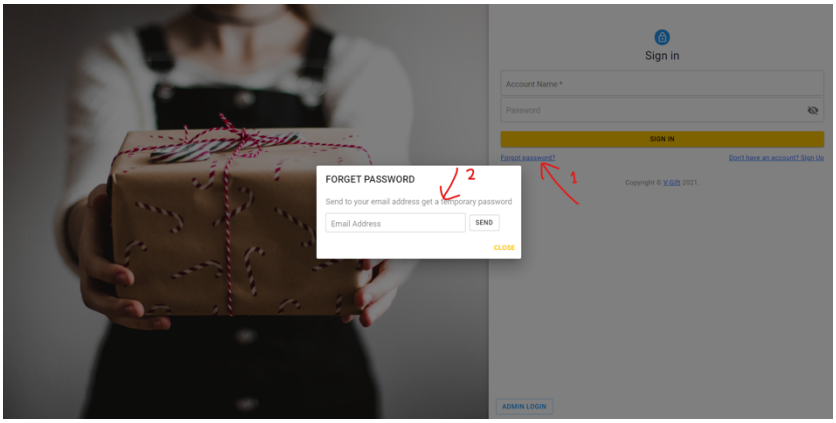
Upon server startup, a welcoming page is shown, and click buttons to enter information to login / Register/ Forget Password.



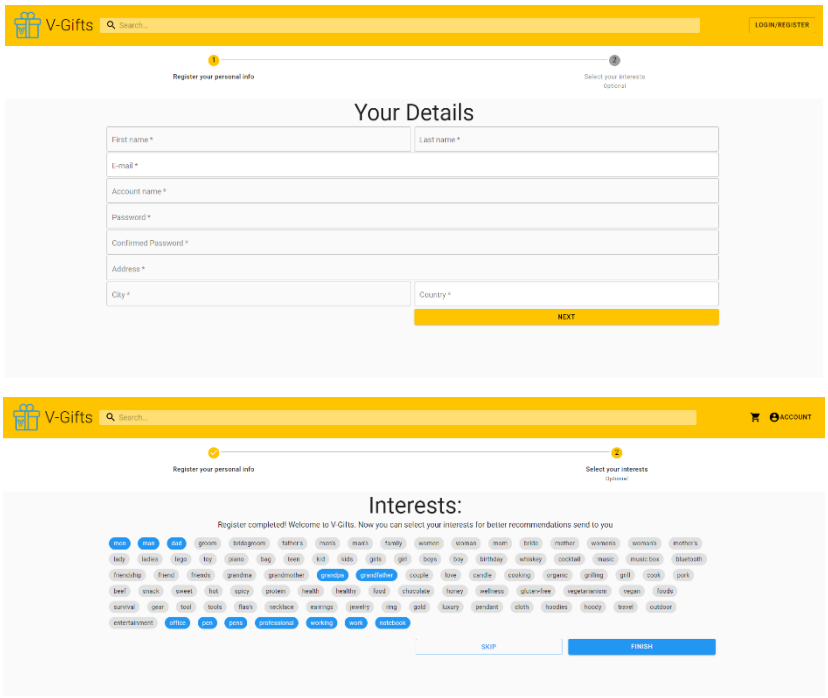

Then, fill in account name and password to login; Or to click Blue link at right to register.




Or click Forget Password Link to send email to obtain temporary password from your registered email.



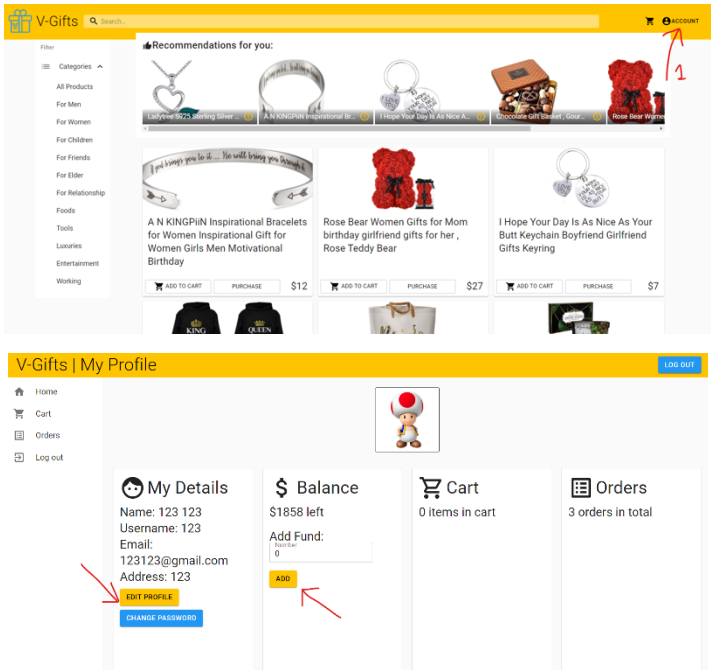

In the register page, fill in basic information to continue and setup your favorite categories.




Then, you will have access to our main product page.

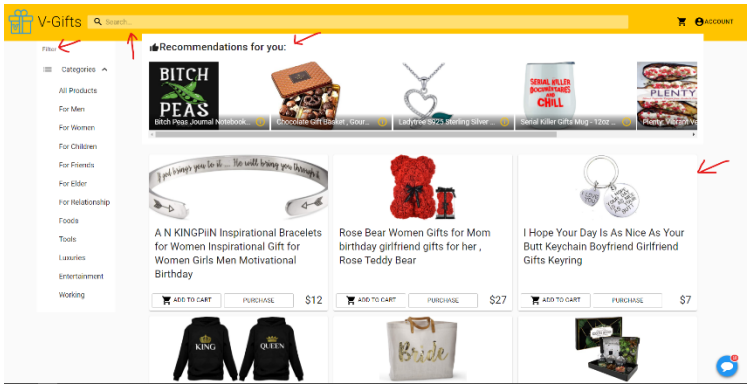
### 5.3.2 Editing User Information and Adding Fund

User can edit their non-unique information any time they want. They can also add more credit if they need to buy products.



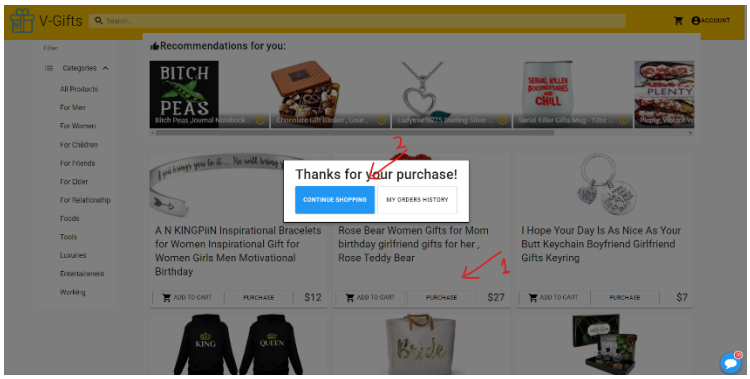
### 5.3.3 Searching Product

In the home page, a search bar is provided for keyword searching; a filter is provided at left hand side; a row of recommended products is provided based on user's interest; and other product are shown below available for browsing.

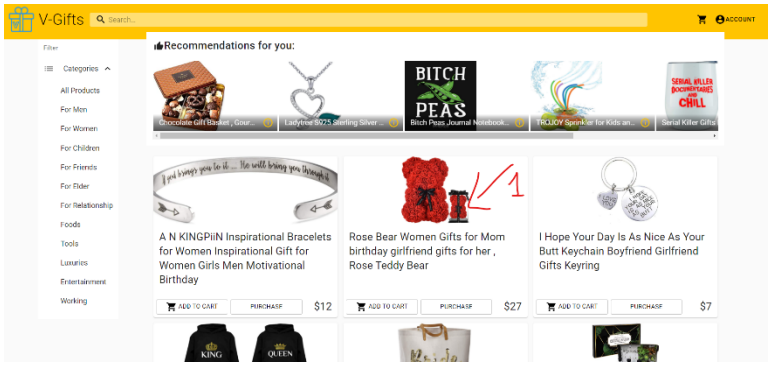


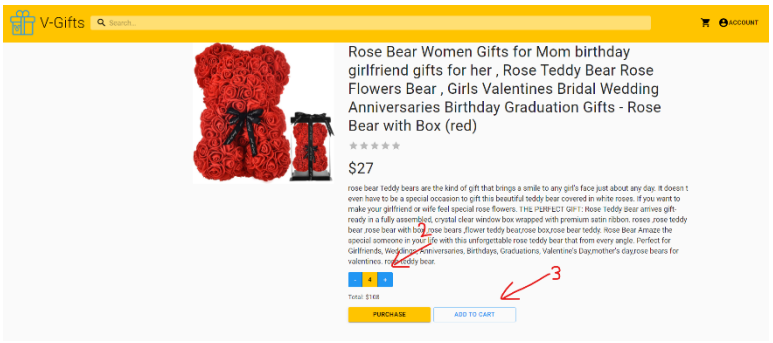

### 5.3.4 Purchasing Product

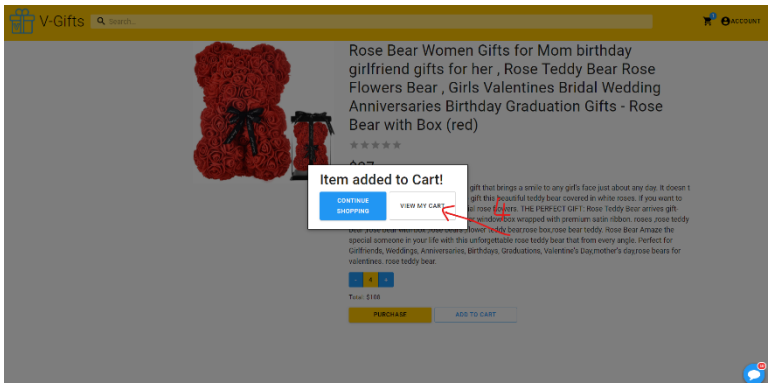
You have two ways to purchase a product. One is to direct purchase from homepage.

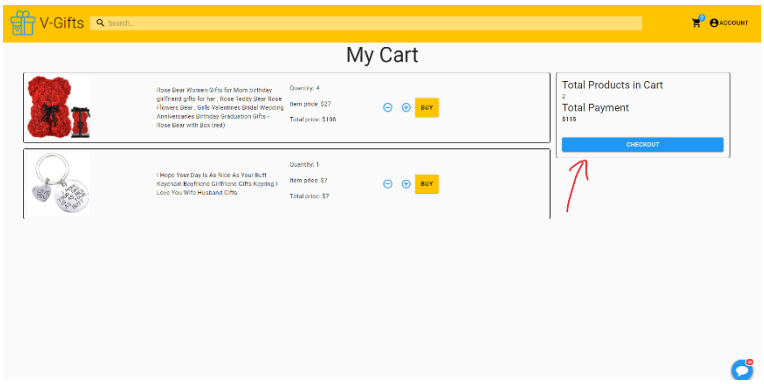



Another way is to add product into shopping cart, and then purchase together.



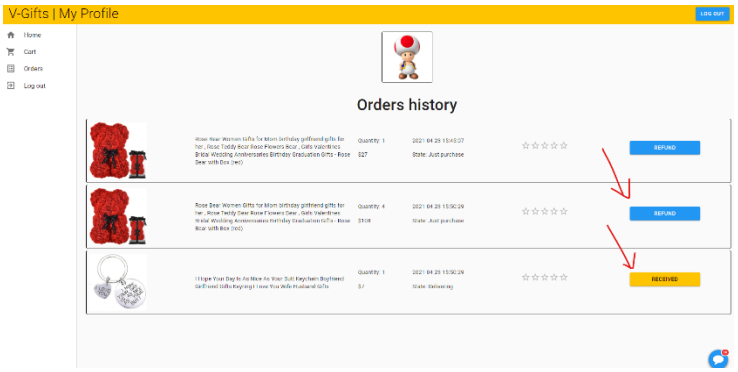


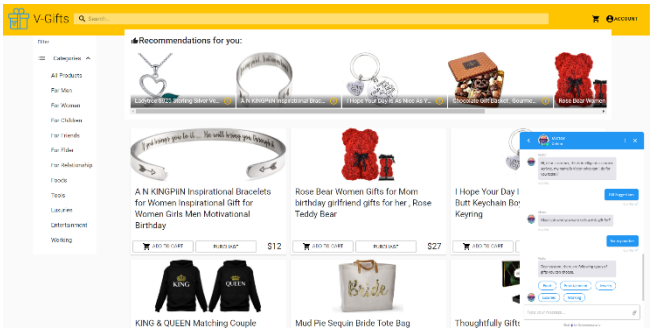

### 5.3.5 Refund and Receive Order

Right after purchase, before admin starts to deliver an order, users can refund an order to get credit back. Users can also click receive order to confirm the delivery.




### 5.3.6 Chatbot

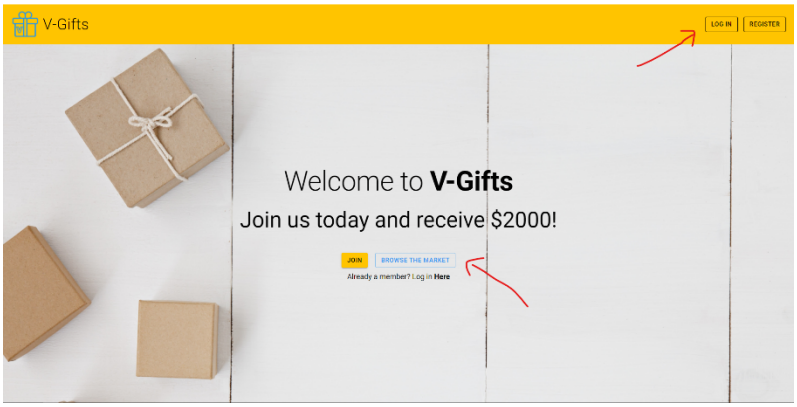
A chatbot AI is provided to inspire the need of a user. To click the bubble at bottom-right corner to talk.

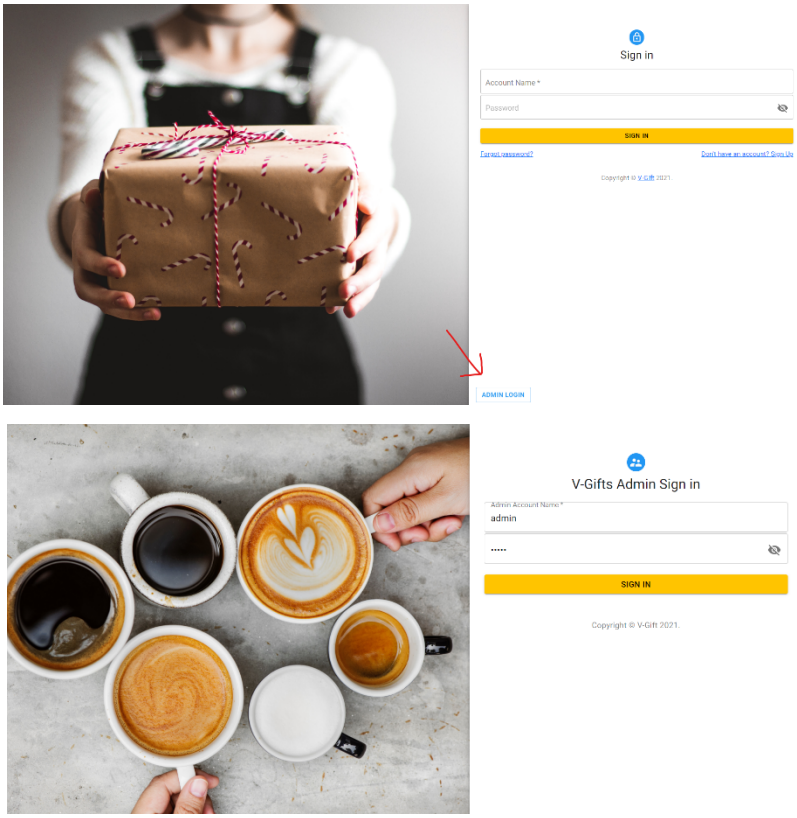



## 5.4 Using Product as Admin

### 5.4.1 Login

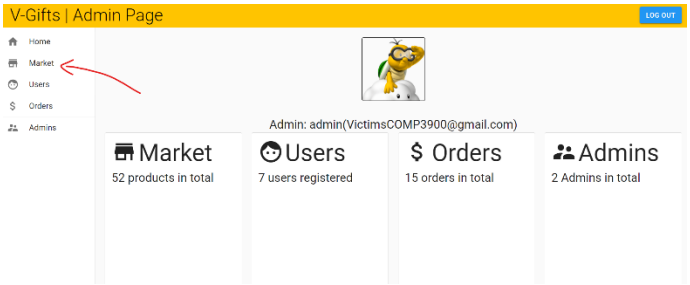
Admin can login from a secret button. A default admin account is already inside database. Its account name is **admin**, and password is **admin**.

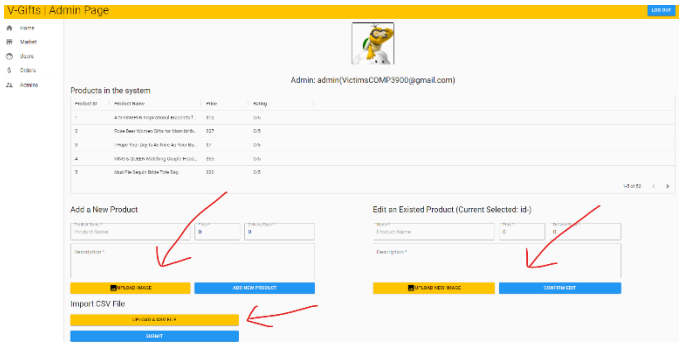





### 5.4.2 Managing Product

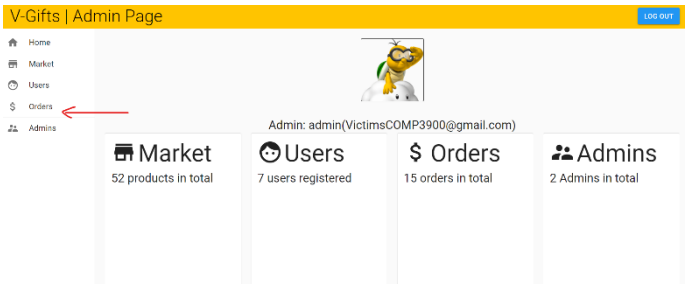
Admin can access the list of products from market label. Admin can add new product and can click on any product to edit them.

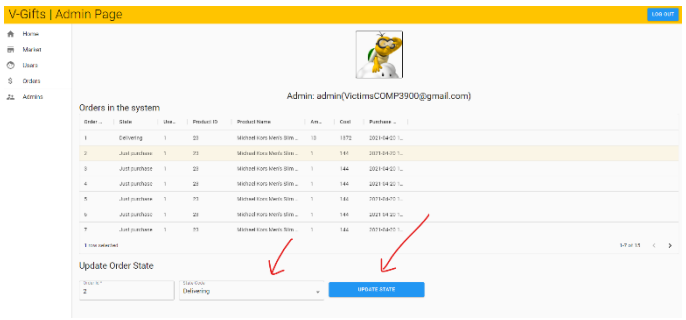





## 5.4.3 Managing Order

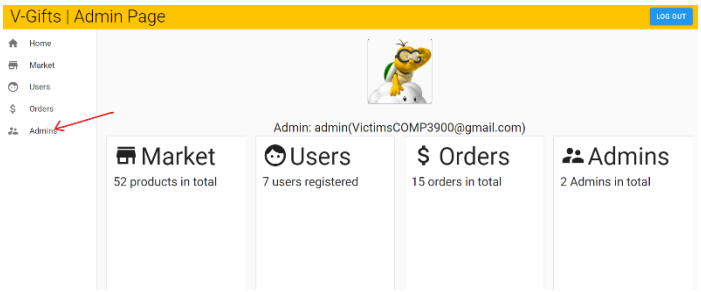
Admins can also change states of an order to start order delivering or cancel an order.

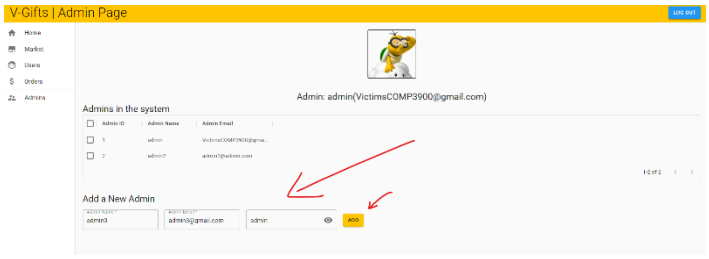



## 5.4.4 Create New Admin

Finally, an admin can create more admins to allow other operator to manage the website.




V-Gifts | Admin Page
LOG OUT
Home
Market
Users
Orders
Admins
Admin: admin(VictimsCOMP3900@gmail.com)
Admins in the system
Admin ID
Admin Name
Admin Email
1-2 of 2
Add a New Admin
admin3
admin3@gmail.com
admin
ADD

# 6. References


Cohen, A. (2003). *The Perfect Store.* Boston: Back Bay Books.

Grinberg, M. (2018). *Flask web development: developing web applications with python.* Reilly Media, Inc.

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. (2020). Array programming with NumPy. *Nature 585*, pp. 357–362.

*Introducing JSON*. (2021, 4 15). Retrieved from Introducing JSON: https://www.json.org/json-en.html

*kommunicate*. (2021, 4 10). Retrieved from kommunicate: https://www.kommunicate.io/

Matthews, J. (2013, April 18th). *dabapps*. Retrieved from https://www.dabapps.com/blog/introduction-to-pip-and-virtualenv-python/