[1. Overview 3](#_Toc69630070)

[1.1 Introduction 3](#_Toc69630071)

[1.2 Our Product 3](#_Toc69630072)

[1.3 System Architecture 4](#_Toc69630073)

[2. Third-Party APIs and Frameworks 7](#_Toc69630074)

[2.1 Frontend 7](#_Toc69630075)

[2.1.1 Material UI 7](#_Toc69630076)

[2.1.2 React 7](#_Toc69630077)

[2.2 Backend 7](#_Toc69630078)

[2.2.1 Werkzeug 7](#_Toc69630079)

[2.2.2 Numpy 7](#_Toc69630080)

[2.2.3 Dialogflow 8](#_Toc69630081)

[3. Functionalities 9](#_Toc69630082)

[3.1 Frontend 9](#_Toc69630083)

[3.1.2 Admin Interfaces 9](#_Toc69630084)

[3.1.3 User Interfaces 9](#_Toc69630085)

[3.2 Backend 9](#_Toc69630086)

[3.2.1 Database Management 9](#_Toc69630087)

[3.2.2 Admin Operations 9](#_Toc69630088)

[3.2.3 User Operations 9](#_Toc69630089)

[3.2.4 Product Promotion 10](#_Toc69630090)

[3.2.5 Chatbot AI 10](#_Toc69630091)

[4. Implementation Challenges 11](#_Toc69630092)

[4.1 Database Management – Atomic operations 11](#_Toc69630093)

[4.2 Chatbot AI implementation 11](#_Toc69630094)

[4.3 Frontend Framework 11](#_Toc69630095)

[5. User Documentation and Manual 12](#_Toc69630096)

[5.1 Windows 10 12](#_Toc69630097)

[5.2 MAC OS 12](#_Toc69630098)

# 1. Overview

## 1.1 Introduction

E-commerce earned its first glance from the world at the end of last century (Cohen, 2003; Stone, 2013). Although it was much insignificant compared to its scale today, the pioneers, Amazon and Auction Web (predecessor of eBay until 1997 (Mullen)), 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.

A chat bot AI is also implemented as the novelty feature of our product …… (ChAtBoT pArT)

## 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 balabala

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://. 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 reduced 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 uses 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

### 2.1.2 Material UI

HI QP 你们也留个备份 - ok

## 2.2 Backend

### 2.2.1 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). (前端加一下描述怎么handle每一个error的，能撑好几页！)

Table 1: Custom HTTP status code

|  |  |
| --- | --- |
| Code | Http Status |
| 460 | InvalidToken |
| 461 | InvalidUsername |
| 462 | InvalidEmail |
| 463 | UsernameAlreadyExit |
| 464 | IncorrectUsername |
| 465 | InvalidPassword |
| 466 | NotEoughFund |
| 468 | EmailAlreadyExit |
| 469 | NoImage |

* 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.
* EmailAlreadyExit exception is raised if an email is already taken by a user, to prevent email sharing between accounts.
* NoImage exception is raised if no image is provided upon new product creation.

### 2.2.2 Numpy

### 2.2.3 Dialogflow

# 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

### 3.1.2 Admin Interfaces

### 3.1.3 User Interfaces

## 3.2 Backend

### 3.2.1 Database Management

### 3.2.2 Admin Operations

### 3.2.3 User Operations

### 3.2.4 Product Promotion

### 3.2.5 Chatbot AI

# 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 inspecting, 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 Database Management – Atomic operations

## 4.2 Chatbot AI implementation

## 4.3 Frontend Framework

4.4 backend database structure changes a lot

# 5. User Documentation and Manual

## 5.1 Windows 10

## 5.2 MAC OS