E-SHOP.IO

Team 9: Product Backlog

Julian Chen, Eliel Vipata, Viet Nguyen, Ajay Kumar, Saatvik Anumalasetty

Problem Statement:

E-commerce websites have become the goto solution for most people who intend to start a business online yet most of the tools that enable users to build these sites such as Shopify, NetSuite, Squarespace and HackMD do not support live collaboration and do not have a convenient interface for sellers and customers to communicate, which makes creating a website in a team very tedious and slow. The problem can be solved by creating a website builder with access to live collaboration, follow system, message network, and review area to allow teams and companies to build their services enabling them to market faster and more efficiently. In brief, E-shop.IO is an E-Commerce website builder for customers who wish to sell their products online, can organize, advertise, and make it extremely easy for their customers to fall in love with their products.

Background Information:

In the wake of the pandemic, platforms like shopify saw significant growth as people realized their own ability to make and sell products without needing a physical store. E-commerce software empowers individuals in this way and a lot of the software that already exists gives power to individuals. E-shop.io expands on e-commerce software that already exists by allowing for groups of people to work together easily on the same shop and collaborate live to build the shop and manage the products. Other solutions like shopify or hackMD are clunky in their implementation of live collaboration or require third party software to allow for people to work together in their shops. Our product is targeted towards groups of people who are interested in creating their own shops and desire the ability to do it with other people.

Functional requirements:

1. As a merchant I would like the products that are best rated / most bought to pop up on my front page
2. As a merchant I would like to be suggested the most used website format / template for my type of products
3. **As a merchant I would like to be able to communicate with other merchants and collaborate with them, which would allow us to easily combine our pages**
4. As a merchant I would like to have the ability to create a unique shop for my product.'
5. As a merchant I would like the ability to make transactions through the platform.
6. As a merchant I would like to emphasize new products.
7. As a merchant I would like to see the reviews on my products.
8. As a merchant, I would like to save my previous work and come back to it when I make a mistake.
9. As a merchant, I would like to create a basic template and work on it if I have no specific idea.
10. As a merchant, I would like to autosave my website creation when I forgot to save.

11. As a merchant, I would like to be suggested a website template for my product type.
12. As a merchant I would like to be able to edit my website simultaneously with teammates
13. As a merchant I would like to be able to add video links to reviews for my products to the website
14. As a merchant I would like to be able to add social links to my website
15. As a merchant I would like to have access to a preview of my site for different platforms (mobile/desktop)
16. As a merchant I would like to be able to make comments to changes made to my website by other teammates
17. As a merchant, I would like to be able to both sell and buy products from other stores from my account.
18. As a merchant, I would like to be able to view changes made by other teammates in real time.
19. As a merchant, I would like to suggest changes or edits to the website.
20. As a merchant, I would like to be able to sell my website
21. As a manager I would like to be able to purchase a domain name
22. As a merchant I would like to see some form of tutorial on how to create a shop using the software.
23. As a merchant I would like to have login credentials for my account.
24.
25. As a customer I would like to be able to purchase a product from a store.
26. As a customer I would like to be able to share a product with my friends
27. As a customer I would like to be able to look at the reviews on the product.
28. As a customer I would like to be able to search for products that are available from different shops.
29. As a customer, I would like to be able to look at my purchase history
30. As a customer, I would like to have products recommended to me based on my previous purchases.
31. As a customer, I would like to be recommend a store that will suit my interests
32. As a customer I would like to save my payment information to my account so that it is easy to make multiple payments.
33. As a customer, I would like to be able to suggest and request permission to add changes to a shop.
34. As a user, I would like to see users' shopping activity if I have added them.
35. As a user, I would like to auto login from the browser.
36. As a manager, if I have group members, I would like to be able to give roles to them.
37. As a manager, I would like to be able to review changes made by the team.
38. As a manager, I would like to be able to accept or deny changes made to the website.
39. As a manager, I would like to be able to manage who had access to my website and what permissions they have to edit or view
40. As a manager I would like to have more than ten people be able to work concurrently on the website (If time allows)
41. As a user I would like to be able to change themes of my website with the click of a button

42. As a user I would like to be able to support cryptocurrency as a payment option (if time allows)
43. As a customer, I would like to be able to use a check as a payment option
44. As a customer, I would like to be able to create multiple shopping lists (checklist) to remember what I want to buy
45. As a customer, I would like to rate products and shops
46. As a customer, I would like to see the reviews and ratings from other customers that bought the same product from the shops
47. As a customer, I would like to upvote and downvote the ratings based on if I like the review or not
48. As a manager, I would like to be able change session duration so make my website more secure
49. As a manager I want to be able to get a summary of my website analytics
50. As a user I would like to be able to see what tasks I've been assigned by my manager(If time allows)
51. As user, I would be able to have a ability to press check if I am done with my task (if time allows)
52. As a manager, I would like to have a calendar to put tasks for each day to do (if time allows)

Non-Functional Requirements:

Architecture:
We would like to have completely separate backend and frontend. By having separate frontend and backend, our customers would find it a lot easier to manage their databases and their pages without these actions interfering with each other. The backend will model RESTful API since Parse automatically creates end points when a table is created, modelling the REST API. Due to the way Parse is set up, every request is limited to 1000 objects in a single request and 1800 requests per minute. So the server should be able to handle multiple edits happening at the same time.

The project is also going to be able to support live collaboration, the framework of which is going to be implemented through the Operational Transformation control algorithm COT, which allows for Operational Transformation - base dos and undos, adding to the convenience. The caching system is going to be done through Django, which is extremely robust and useful in terms of avoiding redo of expensive calculations.
The frontend is going to be implemented by the most popular language, html, for obvious reasons such as convenience and clearness. We also plan on using javascript to implement the backend since it is a very convenient language yet very powerful with its libraries.

Security:
Because transactions will be happening frequently through the software, it is important to ensure that the information of customers and merchants is secure. The Stripe API to handle these transactions is secure as all card numbers are encrypted and Stripe itself is certified to PCI service provider level 1 which is the strictest certification relating to payment processing. Stripe can also handle account creation to ensure that user credentials are kept secure. For extra security, we plan to use a session manager in our database to keep track of which users are logged in and what devices are being used. Each session will have a max of 5 days until it expires and the user is required to login again. Managers of projects will be able to edit this session duration according to how they see fit.

Deployment:
All our application data and media will be stored on a Parse server instance. The frontend code will be placed in a public directory for user access and the backend code including javascript libraries will be placed in a cloud directory. The front end will send http requests to the cloud code functions in the cloud directories. Cloud code functions will include tasks like sending emails to a website's entire customer base, live queries for periodically checking for changes in the databases. All the website's front end code will be edited using Django since it is easier for file management and all website changes logged in a private file for version control and reverting if the user wishes to.

Usability:
Creating a shop needs to be a streamlined process that should not take longer than two days for even the most inexperienced users of the software. The use of adjustable templates will allow people to shop quickly, while also being able to customize it to make their own shop unique.

The interface for the site needs to be as appealing to customers, which will in turn promote merchants to create shops on the site. To accomplish this, there are key elements on a shop page that need to be emphasized.

Website publishing:
In order for our service to host multiple domains, each website created will have its own directory, permissions and config file. Since we plan on using Django, the django hosts package allows for multiple domain names to be registered to one application and have the requests to the domains routed to their corresponding URLconf. All website and media data will be stored in their respective directories and their data indexed in their own databases.

Scalability:
As more data is stored, the software needs to adapt and make sure the server doesn’t crash. The software should be able to run smoothly to meet the user requirements, and the recovery speed of the website needs to be quick. The examples of scalability in our software include database, cache, and system.

Parse is able to do database scalability. But, if more storage is needed, then Parse is deployed to other services like Firebase. Django (Memcache) manages the cache by using an interface that can add, retrieve, and delete data fast, so there is less traffic and quick recovery rate for the website. Memcache can also share cache over multiple servers, which can be useful when we do horizontal scaling. We plan

on using an 8 core cpu and limiting the docker container for our server to 4 cores. Since the average request time for a parse request is 10ms, the max number of the server will be able to handle is 400/sec. Horizontal scaling will keep creating a new server every time we pass the number of required requests per server, to decrease the system load. We will have a max of 2 containers per server which will allow us to have 800 requests/sec before creating a new one.