

E-shop.io
DESIGN DOCUMENT



E-shop.io

TEAM 9:
-Eliel Vipata
-Viet Nguyen
-Ajay Kumar
-Julian Chen
-Saatvik Anumalasetty

Index:

- Purpose 3
- Design Outline 8
 - Design Decisions (Components of your system)
 - Sequence of Events
 - High Level Structure of System (UML DIAGRAM)
- Design Issues 9
 - Functional Issues
 - Non-Functional Issues
- Design Details: 13
 - Class Diagrams
 - Class Interactions
 - Sequence Diagrams
 - Activity Diagrams
 - Navigation Flow Map
- Mock User Interface25

Purpose:

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.

This problem can be solved by creating a website builder with live collaboration to allow teams and companies to build their services enabling them to market faster and more efficiently. The project will also implement a follow system, message network, and review area to strengthen the connectivity between buyers and sellers.

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.

In essence , E-shop.io is an E - Commerce website builder, which supports live collaboration, for customers who wish to sell their products online, and can organize, advertise, and make it extremely easy for their customers to fall in love with their products.

Functional Requirements:

1. User Account
 - a. As a user, I would like to auto login from the browser.
 - b. As a merchant I would like to have login credentials for my account.
2. Shop Viewing

- a. Preview
 - i. As a merchant I would like to have access to a preview of my site for different platforms (mobile/desktop)
 - b. After Publish
 - i. As a merchant I would like the products that are best rated / most bought to pop up on my front page
3. Website Builder
- a. General Builder Features
 - i. As a merchant, I would like to save my previous work and come back to it when I make a mistake.
 - ii. As a merchant, I would like to create a basic template and work on it if I have no specific idea.
 - iii. As a merchant, I would like to autosave my website creation when I forgot to save.
 - iv. As a merchant, I would like to be suggested a website template for my product type.
 - v. As a merchant I would like to be suggested the most used website format / template for my type of products
 - vi. As a merchant I would like to see some form of tutorial on how to create a shop using the software.
 - vii. As a user I would like to be able to change themes of my website with the click of a button
 - b. Teammate Features
 - i. As a merchant I would like to be able to make comments to changes made to my website by other teammates
 - ii. As a merchant, I would like to be able to view changes made by other teammates in real time.
 - iii. As a merchant, I would like to suggest changes or edits to the website.
 - c. Manager Features
 - i. As a manager, if I have group members, I would like to be able to give roles to them.
 - ii. As a manager, I would like to be able to review changes made by the team.
 - iii. As a manager, I would like to be able to accept or deny changes made to the website.
 - iv. 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
 - v. As a manager, I would like to be able change session duration so make my website more secure

- vi. As a manager I would like to be able to purchase a domain name
- d. User's Optional Features
 - i. As a merchant I would like to be able to add social links to my website
- e. Additional Requirements (if time)
 - i. As a merchant I would like to be able to add video links to reviews for my products to the website
 - ii. As a merchant, I would like to be able to sell my website
 - iii. As a merchant, I would like to be able to both sell and buy products from other stores from my account.
 - iv. As a manager I would like to have more than ten people be able to work concurrently on the website (If time allows)
 - v. As a user I would like to be able to see what tasks I've been assigned by my manager(If time allows)
 - vi. As a manager, I would like to have a calendar to put tasks for each day to do (if time allows)
 - vii. As user, I would be able to have a ability to press check if I am done with my task (if time allows)
- 4. Front Page
 - a. As a user, I would like to be able to have login option for shopper and seller
 - b. As a user, I would like to be able to have register option shopper and seller
 - c. As a customer, I would like to search for shops even if I am not registered
- 5. Customer Features
 - a. Product
 - i. As a customer I would like to be able to share a product with my friends
 - ii. As a customer I would like to be able to look at the reviews on the product
 - iii. As a customer, I would like to have products recommended to me based on my previous purchases.
 - iv. As a customer, I would like to be recommend a store that will suit my interests
 - v. As a customer, I would like to rate products
 - vi. As a customer, I would like to see the reviews and ratings from other customers that bought the same product from the shops

- vii. As a customer, I would like to upvote and downvote the ratings based on if I like the review or not
 - b. Store
 - i. As a customer I would like to be able to purchase a product from a store.
 - ii. As a customer, I would like to be able to see the review on store
 - iii. As a customer I would like to be able to search for products that are available from different shops.
 - iv. As a customer, I would like to be able to suggest and request permission to add changes to a shop.
 - v. As a customer, I would rate on stores
 - vi.
 - c. Payment
 - i. As a customer, I would like to be able to look at my purchase history
 - ii. As a customer, I would like to have products recommended to me based on my previous purchases.
 - iii. As a customer I would like to save my payment information to my account so that it is easy to make multiple payments.
 - iv. As a customer, I would like to be able to use a check as a payment option
 - v. As a user I would like to be able to support cryptocurrency as a payment option (if time allows)
 - d. Organization
 - i. As a customer, I would like to be able to create multiple shopping lists (checklist) to remember what I want to buy
- 6. Follow System
 - a. As a user, I would like to see users' shopping activity if I have added them.
 - b. As a user, I would like to follow other members
 - c. As user, I would like to see how many followers I have
 - d. As a user, I would like to share my followers with others in other apps (if time allows)

Non-Functional Requirements:

- 1. Architecture
 - a. As a developer I would like to allow groups of individuals to collaborate on the same shop at the same time through Operational Transformation.

- b. As a developer, I would like to separate the frontend of the website from the backend.
- 2. Security
 - a. As a developer, I would like to ensure that all transactions that take place through the website are secured by the Stripe API.
 - b. As a developer, I would like to implement a session manager that can limit the amount of time one person can stay connected to a group's session to 5 days.
- 3. Deployment
 - a. As a developer, I would like to have a continuous delivery pipeline, so that all changes can be deployed efficiently.
- 4. Usability
 - a. As a developer, I would like the platform to be intuitive and usable by anyone regardless of experience.
 - b. As a developer, I would like the websites created on the platform to be visually appealing to customers and easy to use.
- 5. Website Publishing
 - a. As a developer, I would like to host multiple domains on the platform with each having their own permissions and configurations.
 - b. As a developer, I would like to be able to easily access all the website domains I have with the help of a database.
- 6. Scalability
 - a. As a developer, I would like my platform to be able to handle an excess of users with the help of Parse.
 - b. As a developer, I would like to further increase my scalability of the platform by using services like Firebase and Djanga (Memcache)

Design Outline:



- Webclient
 - Website builder that the customer uses to build and edit their store(s)
 - The client will make changes to their website and a new file that replaces the old one will be sent to the web directory
 - Client will then provide the user with a preview of the site that they can use to view their changes and/or publish them
- Server
 - Server receives and handles requests in the form of file transfers.
 - Database requests are handled using json data for logging in, out, storing client data and configurations as well as website
 - Server will also handle API requests to stripe for payment processing for each transaction on the individual websites
- Database
 - All websites will have their own database; each database will handle the updating, insertion, deletion of website data based on user input

Design Issues:**Functional Issues:**

1) Is our website login default the only way the customer could use to login?

- Option 1: Only login with our default login
- Option 2: Also allow login with Google account, Facebook account, LinkedIn and other platforms.

Decision:

We will go with Option 2 because we want to allow our customers to sign in with whichever social media they think is most secure and convenient.

2) If a store was edited, do the changes show up for the owner's customers to view, or will a message saying "server under maintenance" show up:

- Option 1: The new version of the store will be shown to its customers
- Option 2: A "server under maintenance" will show up and the website will only resume showing when the owner is ready to launch it again

Decision:

We will go with Option 2 because we want the customer to have time to properly test their website(s)' changes before launching them.

3) If a store was recently edited and the customers of the store have version compatibility issues accessing the store, how would we deal with it:

- Option 1: A file / message will be sent to the customer listing the configurations they need to update in order to access the website again
- Option 2: The customer would still be able to access the old version of the website, however when they try to access a functionality that is edited or no longer available in the new version, an error message will be sent to them.

Decision:

We will go with Option 1 because we want the customers' customers to have the best experience and updating the configurations will allow them to access the websites more smoothly and avoid inconveniences later on.

- If a customer uses his debit card to pay for the store's product, but there is not enough money in the card, how would we deal with it:
 - Option 1: If the event occurs, Payment gateway will not open, and the customer will get a message like 'Unable to complete transaction. Please use another form of payment.'

- Option 2: Transaction would still be allowed to process and the merchant(s) will get paid once there is enough money in the account(s) of the customers

Decision:

We will go for option 1 because we want to make sure the merchant(s) get paid for their work, since some customers might unregister their cards once they get the product(s) for free

- If two editors are changing the text at same time, how should we deal with it
 - Option 1: There will be a project master that manages which changes can be checked for merging with the main branch first.
 - Option 2: We will have an algorithm in place to put their changes in a queue and decide which changes will be checked for merging with the main branch first

Decision:

We will go with Option 1 since the developers will have a better insight into which changes they want to prioritize, which will be reflected through the person they choose to be their project master

Non Functional Issues:

1. The database scalability might be an issue as more users register on the website
 - a. Option 1: Since the Parse server is the backend that we are using, we can use MongoDB Parse or PostgreSQL to manage our databases
 - b. Option 2: Otherwise, we can deploy the Parse to service like Firebase

Decision:

We will go with Option 1 since Option 2 might take more time to deploy it to another application. Also, MongoDB is more robust and has better scalability compared to a database like Firebase. But if more database storage is needed, Firebase can be a good second option to consider.

2. The server can overload and crash because of the request limit of 400 requests per second
 - a. Option 1: We can use vertical scaling, which means we have to improve our software capabilities of our server by adding more CPUs, memory, or I/O resources to an existing server.
 - b. Option 2: We can use the horizontal scaling method, which means a new server is created every time the server limit is passed for the number of user requests.
 - c. Option 3: Use a mix of both vertical scaling and horizontal scaling

Decision:

Both horizontal and vertical scaling have their own drawbacks. The problems with this horizontal scaling can include an increased amount of timeouts. This can be an issue because the Parse server already has a bit of timeout issues. Second, Parse also has problems for vertical scaling methods because of the high cost of implementing multiple servers. Thus, we will mix both vertical scaling and horizontal scaling to get the combined benefits of both scaling methods while decreasing on the drawbacks.

3. Usability: Since websites will be designed using desktop browsers, it might lead to compatibility issues for the mobile devices because of screen resolutions and aspect ratios.
 - a. Option 1: Create mobile versions of the templates and add a mobile preview option for the user so they can see what their website would look like on a mobile browser.
 - b. Option 2: Allow them to design the mobile version completely from scratch.

Decision:

We will go with option 1. We can use viewport width and height values in css to make relative dimensions for the html elements and media. That way the user's website automatically adjusts to different screen dimensions.

4. Website Publishing: Users may have purchased domains from different websites or services like GoDaddy.com
 - a. Option 1: We could have them manually add their server ip address to them domain name for their site on the service they used to purchase the domain
 - b. Option 2: We could sell domain names on our service and help them migrate the domain over to our eshop

Decision:

We will go with option 1 as we are not experienced when it comes to selling or purchasing domain names. So we will let the users use their trusted services to buy their domains and point to the server ip addresses we will provide to them.

5. Inventory Management: Since our users will be building online stores, it would be helpful if we could offer an inventory or ledger system that would help them manage not only their stock but their sales.

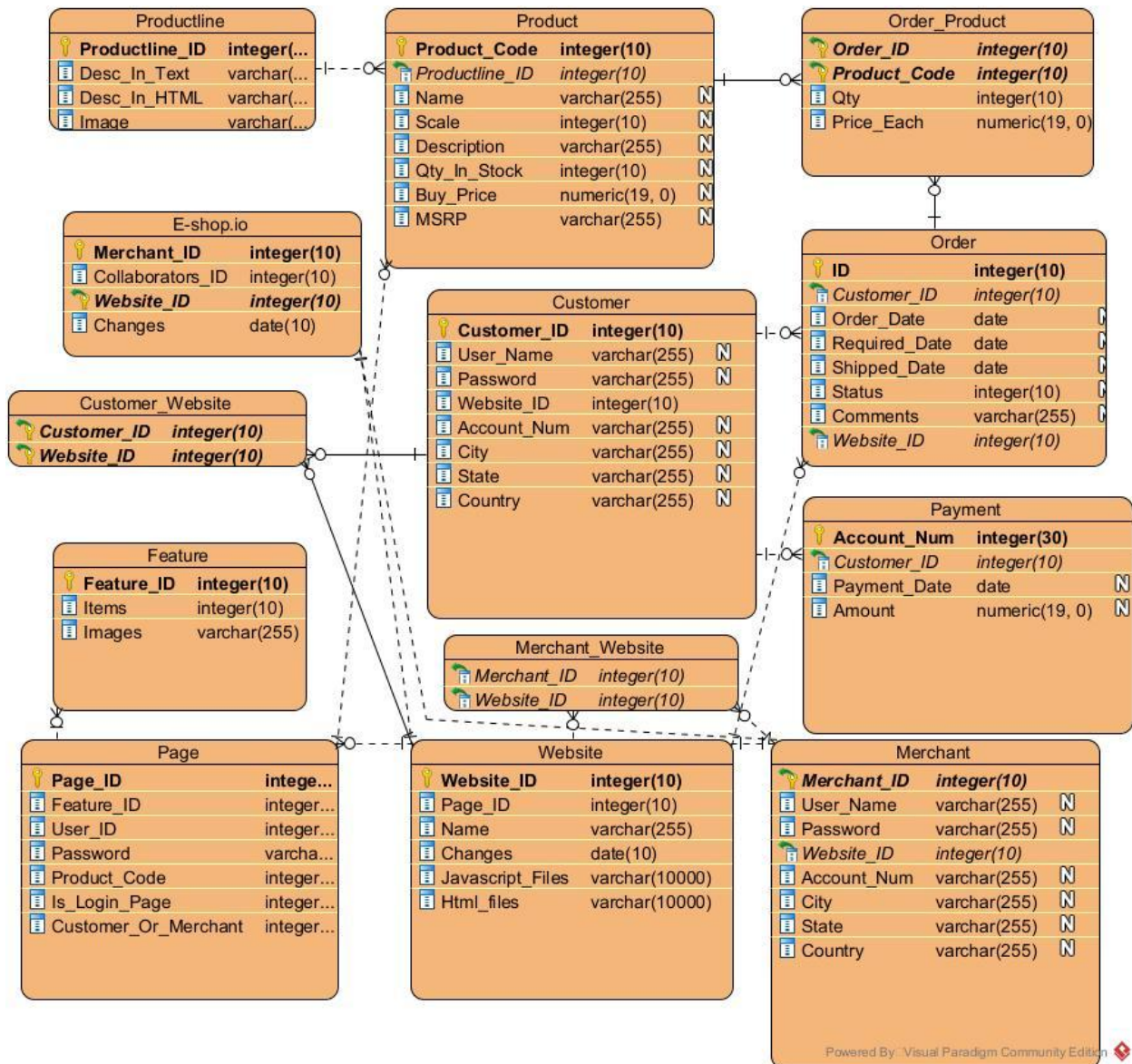
- a. Option 1: We could offer analytics and sales summaries then leave it up to the user to find a service to help them manage physical inventory and cash flow
- b. Option 2: If given time we can build a full inventory system to keep track of stock, shipping updates, cash inflow, outflow and analytics

Decision:

We will go with option 1 for the sake of time and slowly add features to build option 2 as it would be a more comprehensive solution for our users.

Design Details:

Data class model design:



Descriptions of Classes and Interaction between Classes:

The classes are designed based on the objects in our application. Each class has a list of attributes which is the characteristics owned by each object.

• Merchant

- Merchant object is created when someone signs up to be a merchant.
- Each merchant will have a unique user ID.
- Each merchant will have a username, password.
- Each merchant will put in their account number for payment processing, their city, state and country
- Each merchant will put in their account number for payment processing, their city, state and country
- Each merchant will have a list of website ids that they (and their collaborators) are working on

• Website

- Website objects are created when a merchant creates a website.
- Each website will have a unique website ID.
- Each website will have a list of page IDs in the website.
- Each website will have a name.
- Each website will have a list of changes made by the merchant(s) (time the changes are made).
- Each website will have a list of Javascript files and Html files that made up the website.

• Merchant_Website

- Website objects and Merchant objects are connected in a many-to-many relationship through Merchant_Website table
- A Merchant_Website object holds the keys for connecting the two tables, which are Merchant_ID and Website_ID

• Page

- Page objects are the objects associated with Page_IDs in Website table.
- A Page object will contain its ID, the IDs of its Feature's, a value that is essentially a boolean to tell if the page is a log in page, the User IDs (merchant or customer) and passwords of users in case it is a log in page, another boolean to tell if the user is a merchant or a customer if it is a log in page, the product codes of the product(s) in the page

- The page design will depend on the Feature_IDs that it has and the Product_Code(s) will help track the product(s) a page and a website has

• **Feature**

- Features are objects with Feature_IDs corresponding to those in the Page table.
- A feature object would have an ID, a list of items and a list of images
- Items could be whatever the merchant decides for their websites, be it pages design styles, or for instance, a product preference menu used for matching users preferences with products in the website for suggesting products to users.

• **Customer_Website**

- Customers connect to Websites in a many-to-many relationship through the Customer_Website table
- A Customer Website object holds the keys for connecting the two tables, which are Customer_ID and Website_ID

• **Order**

- An order a customer has on a website
- An Order object will have an ID, the Customer ID and Website ID used to uniquely identify the order
- An Order object will have the following dates: the date the order was made, the date the customer requires the order to be shipped, the date the order is expected to be shipped.
- An Order object will have a boolean to tell the status of the order: shipped or not shipped, and the comment the customer made for the order

• **Product**

- A Product object will have a Product_Code, that is found by the Page table and the Order table
- An Product object will have a Productline_ID that identifies the version of the product, a name, the scale of the product, the description, the quantity of the product in stock, the buy price of the product and the price the manufacturers suggest selling the product at

• Order_Product

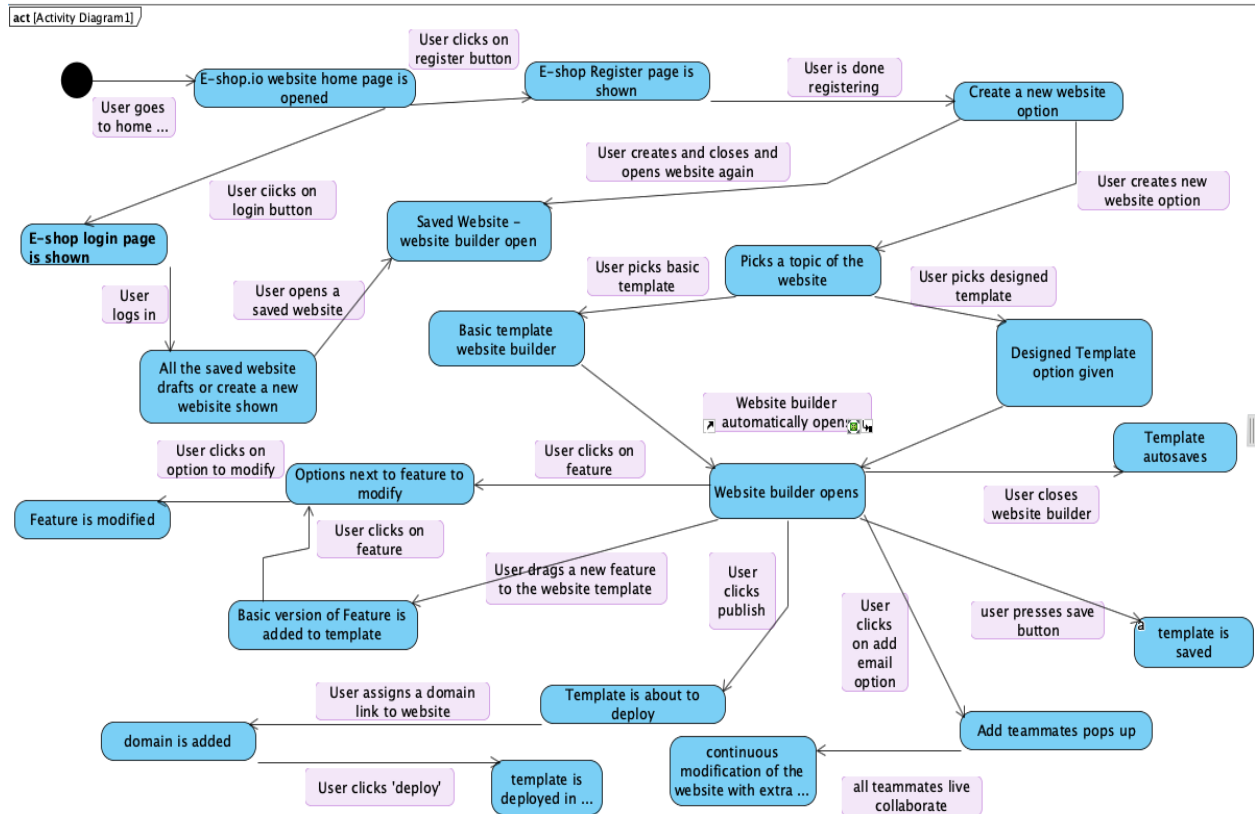
- The Order table connects to the Product table in a many-to-many relationship through the Order_Product table
- The Order_Product table will have the Order_ID and Product_ID to uniquely identify the product order
- The Order_Product table will have the quantity of a product in a order and the price for each product

• Productline

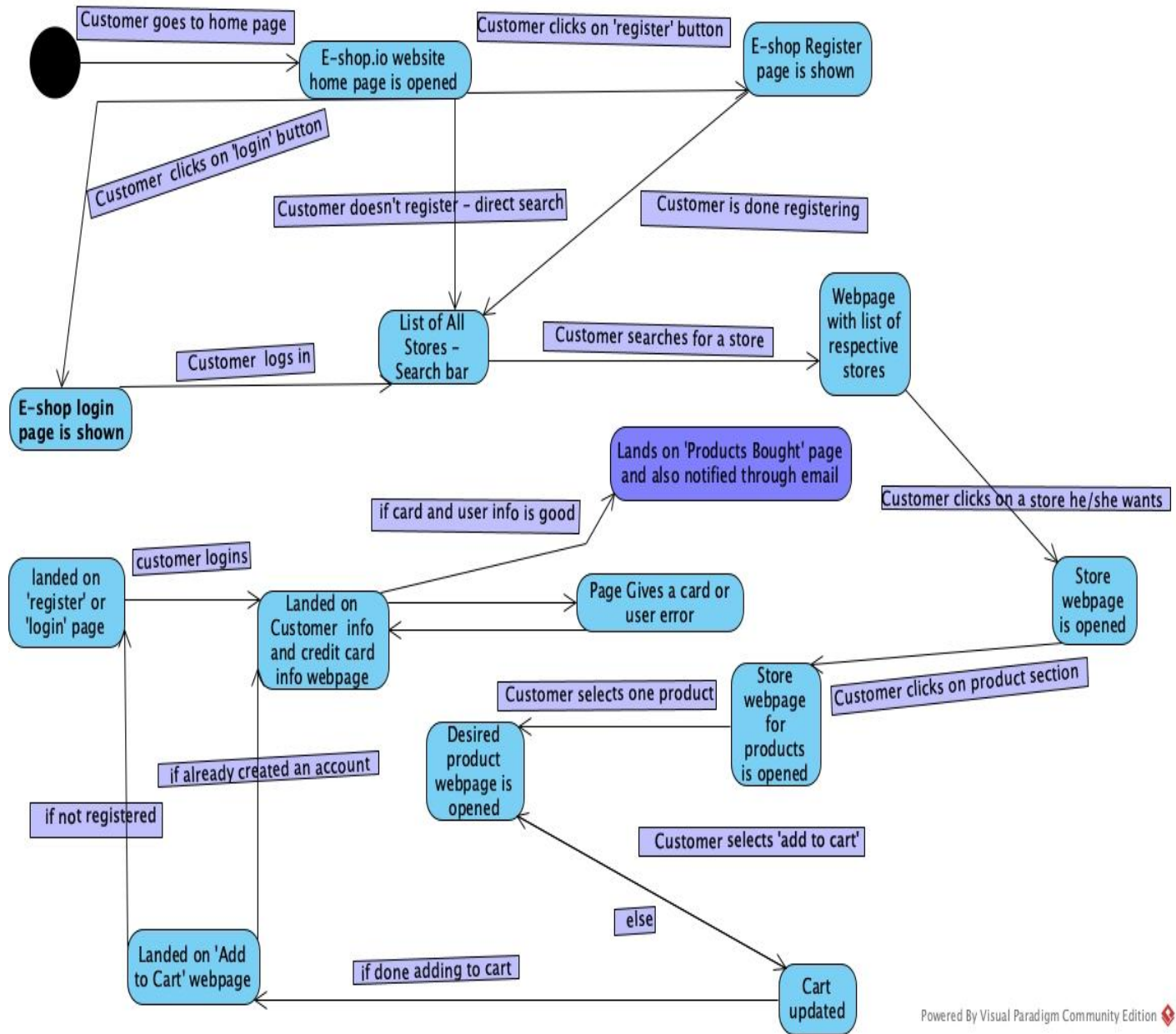
- A Productline object is a line of a product, which could have multiple products
- An Productline object will have a Productline_ID that corresponds to that in the Product table
- An Productline object will have its description in text, html and an image of the product line

Activity Diagrams

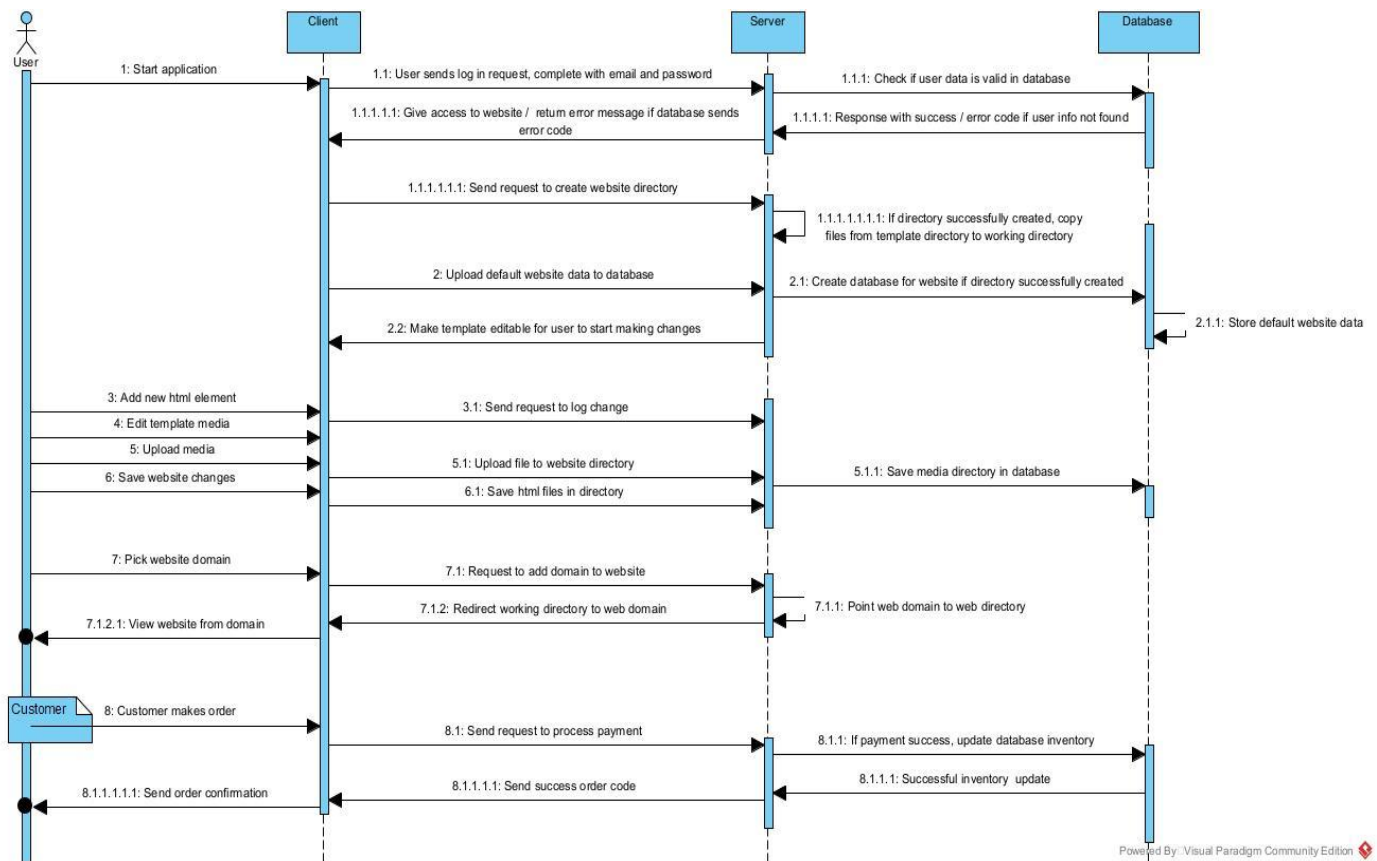
1. High level Activity Diagram of User Creating a Website



2. High level Activity Diagram of Customer Buying a Product from a Shop



Sequence Diagrams

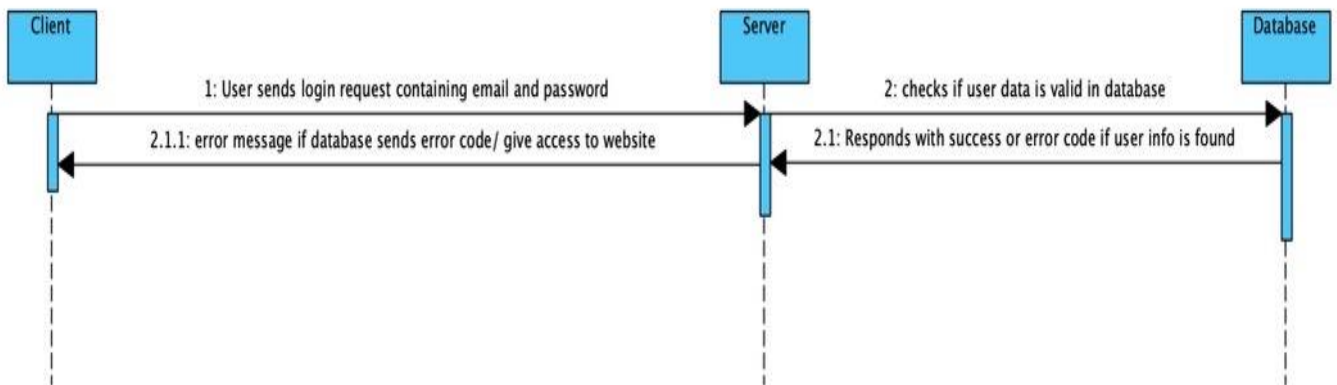


The sequence diagram is initiated by the user start E-shop.io. The user logs in, the Client sends request to the Server, which sends query to and receive data from the Database if the log in credentials are valid. When the user is logged in, they can send a variety of requests to the Client to perform the actions they desire, such as creating an website, editing (adding) their html elements, uploading their files to the website, saving changes and picking a domain. The client will make sure these requests reach the server and the server will make sure to edit (pull from) the database accordingly and send changes (confirmations) back to the client so the user can view. In addition, the customers of a website could also make orders from that website through the client, which will send requests to the server, will process payments, pull data from database so that the client could send the order confirmations to its users.

Overview

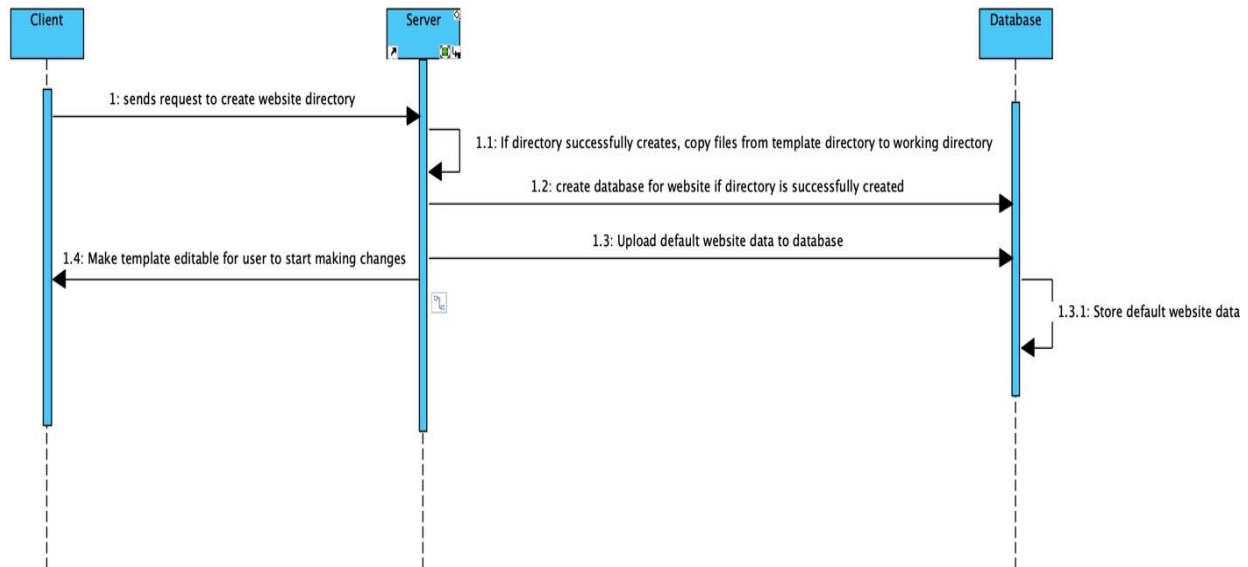
The following sequence diagrams show the typical interactions between the client, server and database throughout different processes.

1. User Attempts to Login
 - a. Every time the user attempts to login, a request containing the user's login credentials. If the credentials valid according to the database, the user is granted to the service.



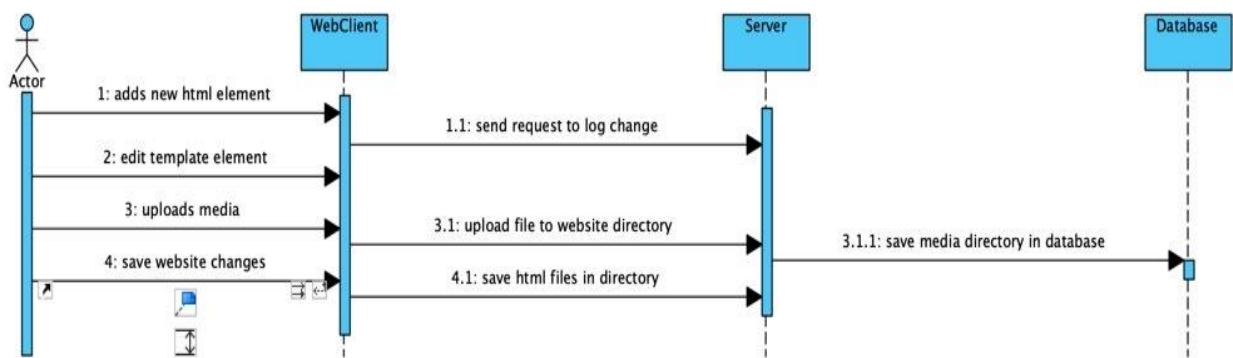
2. User picks template

- a. When a user selects a template, a request is sent to the server to create a new directory, and database for the website. This is where all the data and media will be uploaded and stored.



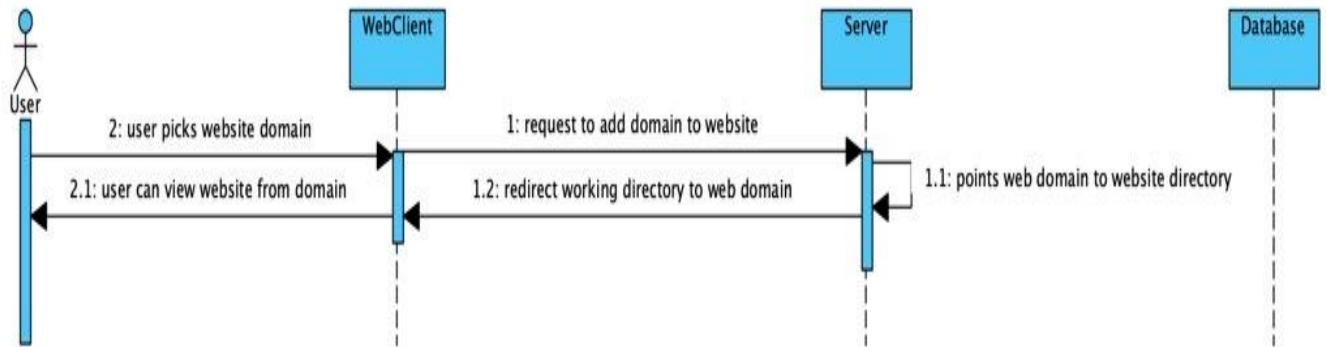
3. User makes changes to website

- a. When a user makes a change to their website, a log file containing the changes gets updated. This log contains a list of commands that will be passed when the user saves or uploads data to be written to the server and stored.

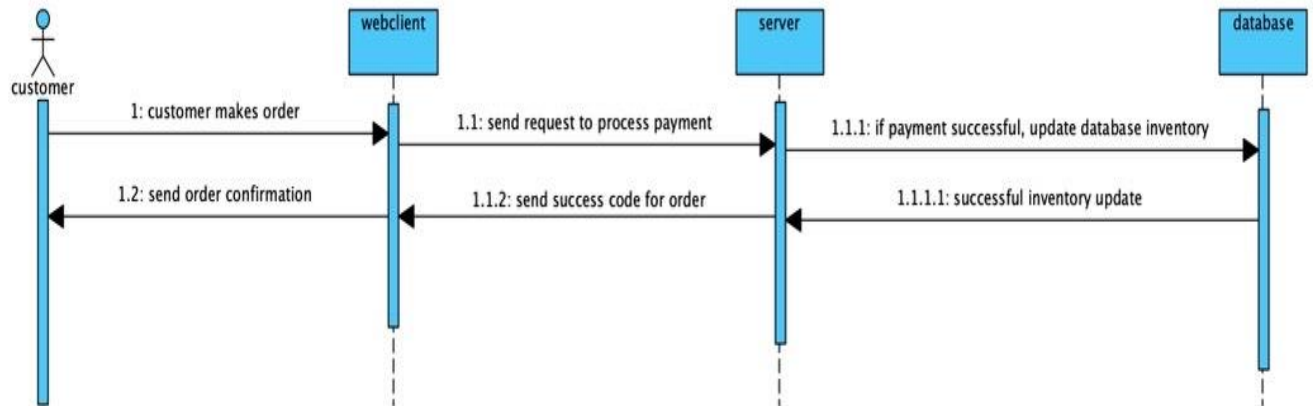


4. User saves and published website

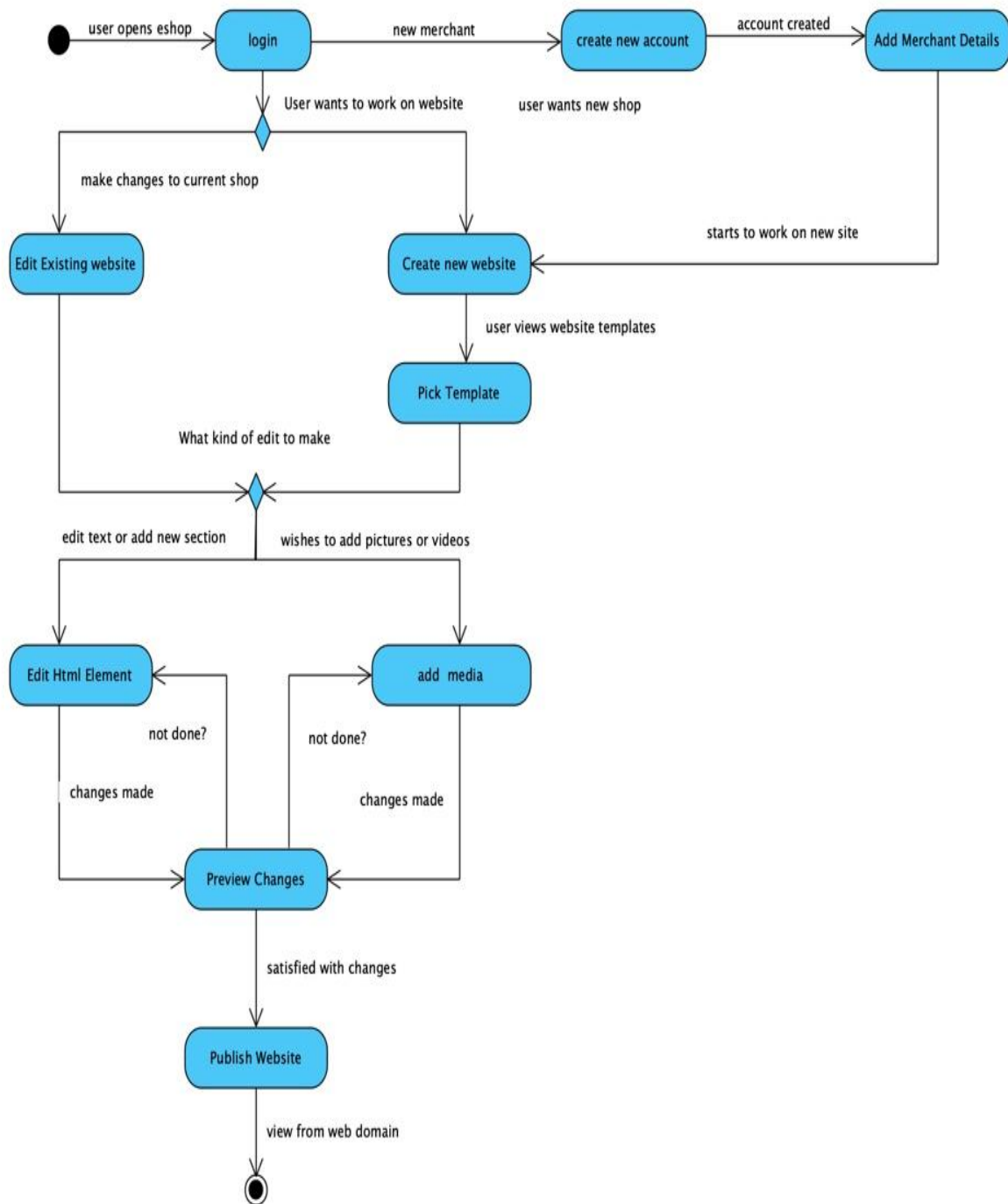
- a. When a user publishes a website, they will have to point their domain to the ip address and directory of the new website. The user is then granted a preview of what the published site looks like.



5. Customer buys from user's website
 - a. When a customer buys from a user's site. They make a request to process a payment, which then updates the user's inventory and database for their stock. If this is successful the customer gets an order confirmation.

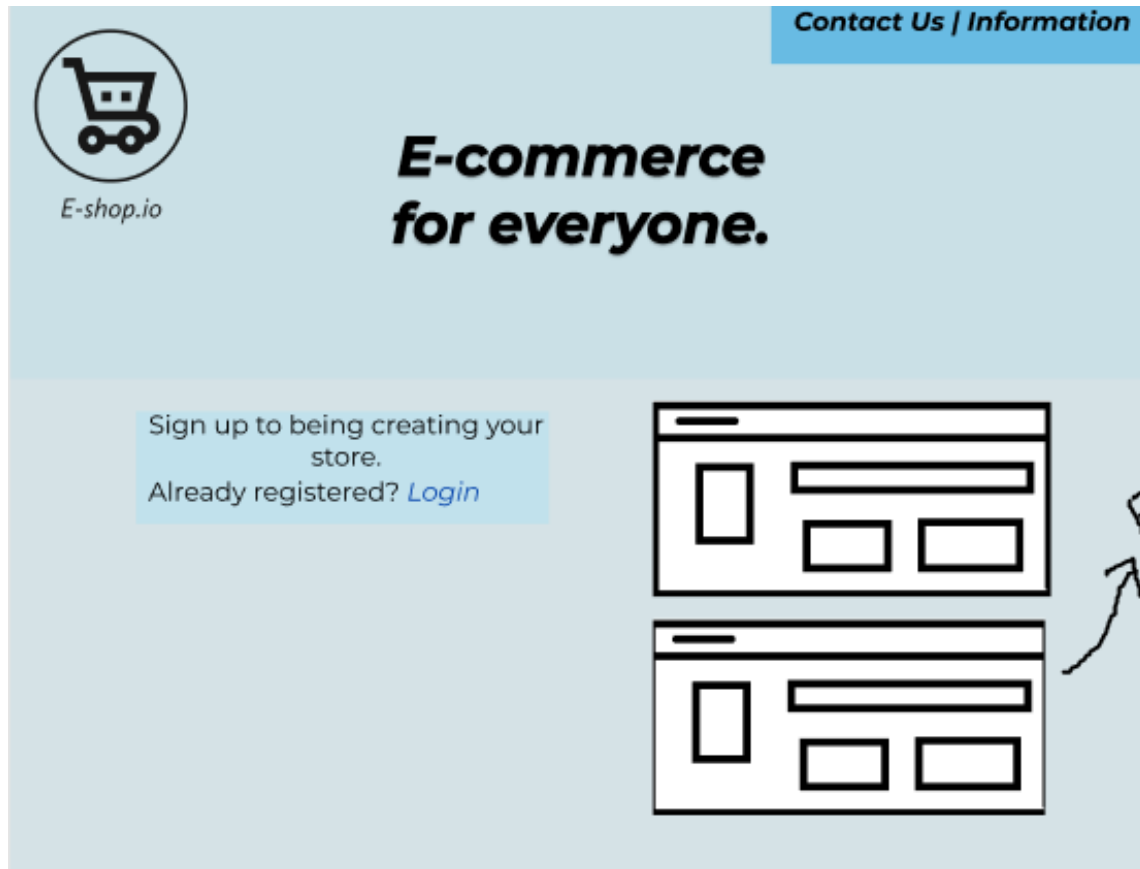


Navigation Flow Map

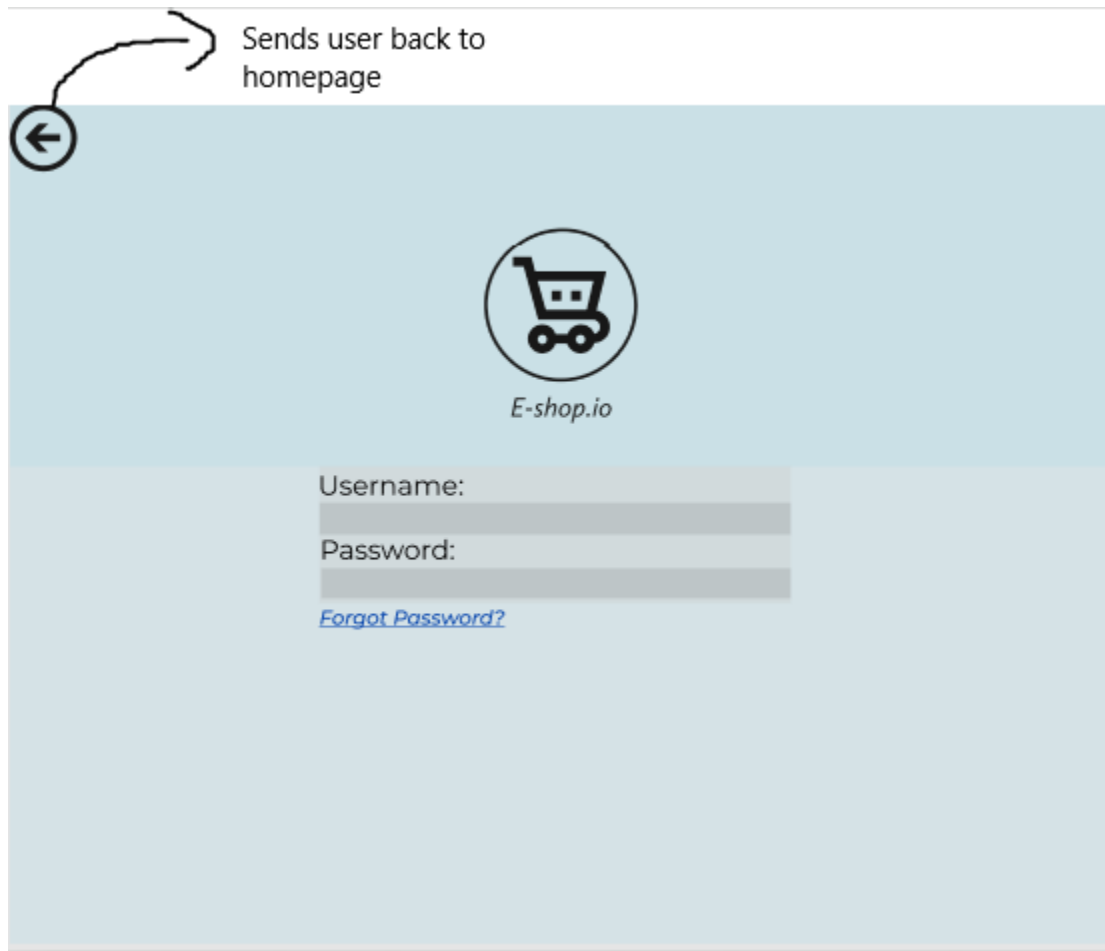


UI Mock-Up

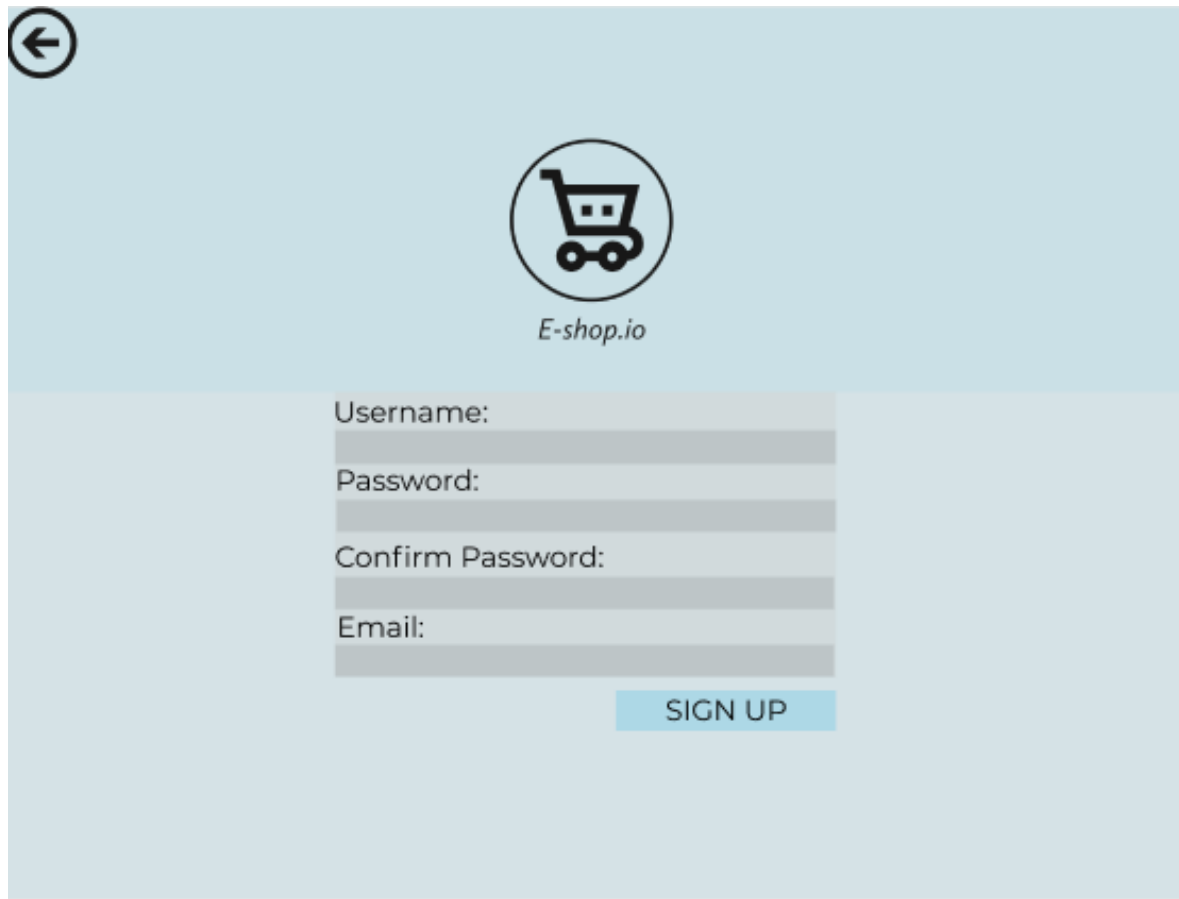
Preview of the homepage for the website.



The first page the user sees when they arrive on the site. There will be an information section where people can figure out exactly what our platform is for. There will also be a place for users to login to their accounts or register an account.

Preview of user login.

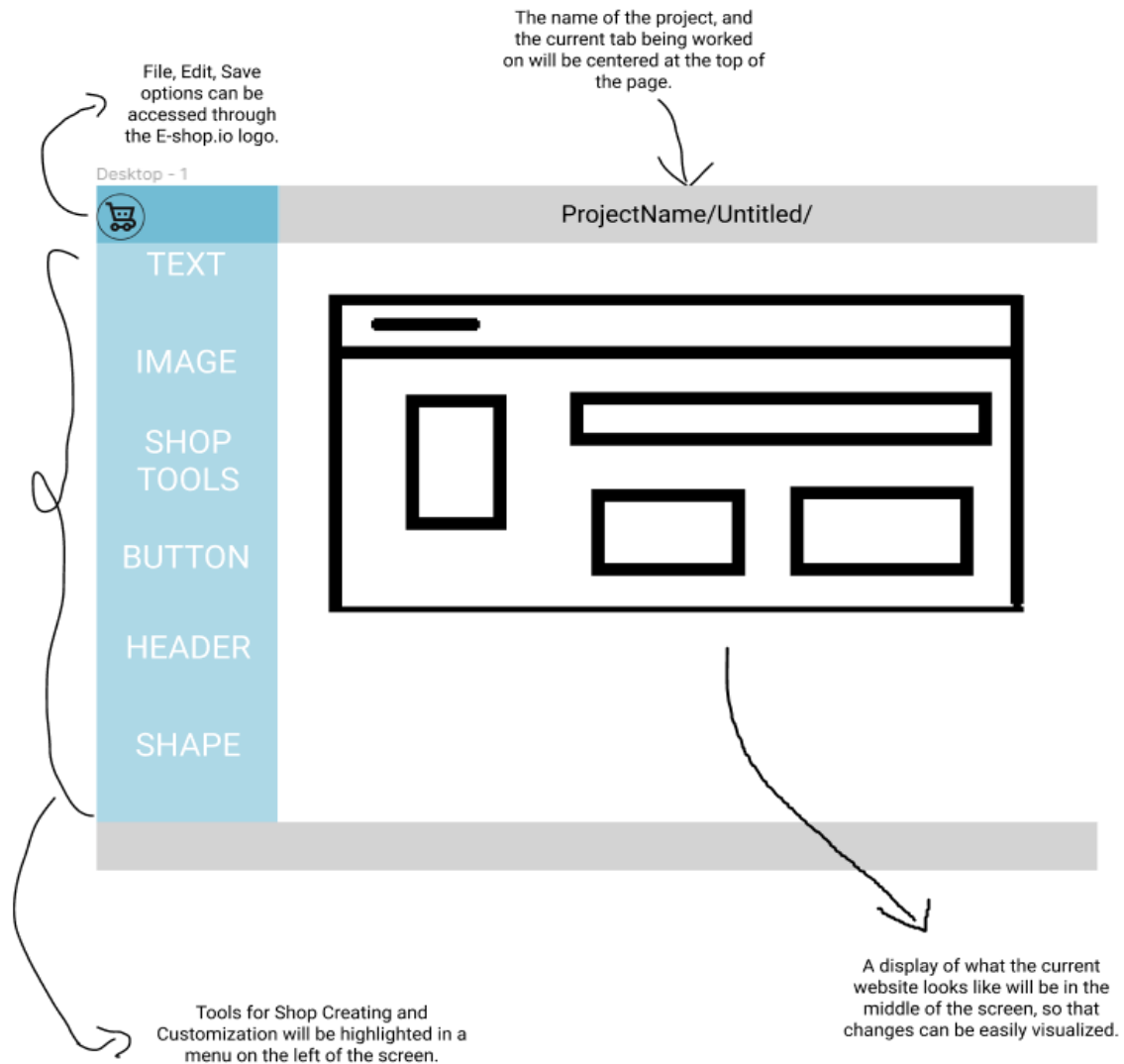
Here is a preview of the page where the user can enter their login information. There is a button to go back to the homepage.

Preview of User Account Creation:

A preview of a user account creation form. The form is centered on a light blue background. At the top left, there is a circular icon with a left-pointing arrow. In the center, there is a circular icon with a shopping cart, and below it, the text "E-shop.io". The form fields are arranged vertically: "Username:", "Password:", "Confirm Password:", and "Email:". Each label is followed by a light gray input field. At the bottom right, there is a blue button with the text "SIGN UP".

Here is a preview of where the user can register an account. It is similar to the login page but there is more information required.

Preview of the shop creation tool.



Here is an example of what the actual shop creation tool will look like. There will be sections where you can access different design tools. A section to see different design templates as well as a live display of what you and your team are working on.