Software Design Specification

for

Vaistreet

Version 1.0 approved

Prepared by Team

Submission Date – 28/10/16

Table of Contents

1. Introduction

The introduction gives a basic understanding of the entire SRS with purpose, scope, intended audiences and reading suggestions. The main aim of this document is to collect and analyze the online shopping software system by defining the problem statement in detail. All the important details and requirements for Vaistreet are stated in this document.

1.1. Purpose :

The main objective of this document is to illustrate the requirements of the project Vaistreet, a social e-commerce website. This document gives a detailed description of both functional and non-functional requirements proposed by the client. The purpose of this project is to provide a responsive web portal for online customers to shop and share products. The main objective is to provide a platform for two purposes: Advertising products and brands in grapevine, providing a catalog from where these products can be bought.

This is a web application which is a combination of an e-commerce as well as social media. Anyone who is socially active and likes doing shopping on web is a perfect customer for this web app. It solves the issue of revisiting different websites for doing shopping and being connected to a social media when you can do it at the same place. This project describes the hardware and software interface requirements using UML diagrams.

1.2. Product Scope :

The Vaistreet website is hosted online hence users can access the website over the internet. This project is specifically being developed for Safcodes Pvt. Ltd. This software bridges the gap between social media and e-commerce by providing a single interface for both functionalities. Users can use the chat inbox feature to chat with their friends and view other's timeline and on the other hand, navigate to the online store where they can buy online goodies and share it with their friends.

Strategically this is a great business idea of Vaistreet. People often look for these types of apps and Safcodes has rightly targeted it and looks promising with their idea. Hence this can gross high attention and revenues in the market.

Online customers are provided with a platform to shop and share, that is the essence of the whole product. We and the product owners envision that this product will be used by buyers to either buy a product or provide content about the product which is publically visible to other users on a newsfeed of the author. This content about a product is central for business. This is what Safcodes can use to drive revenue from the website.

1.3. Definitions, Acronyms and Abbreviations –

- API – Application Programming Interface
- IEEE – Institute of Electrical and Electronics Engineers
- JSON - JavaScript Object Notation
- OTP – One Time Password
- SDD – Software Design Document
- SDS – Software Design Specification
- UML – Unified Modeling Language

1.4. References :

- IEEE template for SDS
- Books
  - Software Engineering: A Practitioner's Approach Fifth Edition By Roger S. Pressman
- Website
  - http://safcodes.com/

1.5. Overview of the Document :

This document is intended for all users and customers who wish to use the software product efficiently and for developers as a guide to know the product design better for future reference. The rest of this SDD contains overall description of the project components, user interface, design decisions and component pseudo code.

- Suggested sequence to read this document for all viewpoints (especially users and product owner) is to read Introduction first and then System Architecture Description for clarity about the product.
- UI designers and testers can refer to Design Decisions and Tradeoffs.
- Developers and testers can refer to a detailed report about the project components in Detailed Description of Components and for the code implementation refer to Pseudo code for Components.

Document Conventions:

- Content inside tables should be Center aligned.
- Rest of the document should be justified.
- Conventional for Main title
  - Font face: Times New Roman
  - Font style: Bold and Underlined

- Font size: 18

- Convention for Sub title
  - Font face: Times New Roman
  - Font style: Bold
  - Font size: 14
- Convention for Body
  - Font face: Times New Roman
  - Font size: 12

2. System Architecture Description

2.1. Overview of Components:

Below is a list of components listed according to the relevant functionality,

1. Catalog Functions

    1. View catalog
    2. Search a product (with advanced search options)
    3. View a product

2. Newsfeeds Functions

    1. View newsfeeds
    2. Browse newsfeeds

3. Login and Sign-Up Functions

    1. Login
    2. Signup
    3. OTP verification
    4. Forgot Password

4. User Profile Functions

    1. Manage credentials (password, email, phone no)
    2. Edit user profile info

5. User Social Functions

    1. Create a newsfeed post
    2. Chat
    3. Send email to customer care

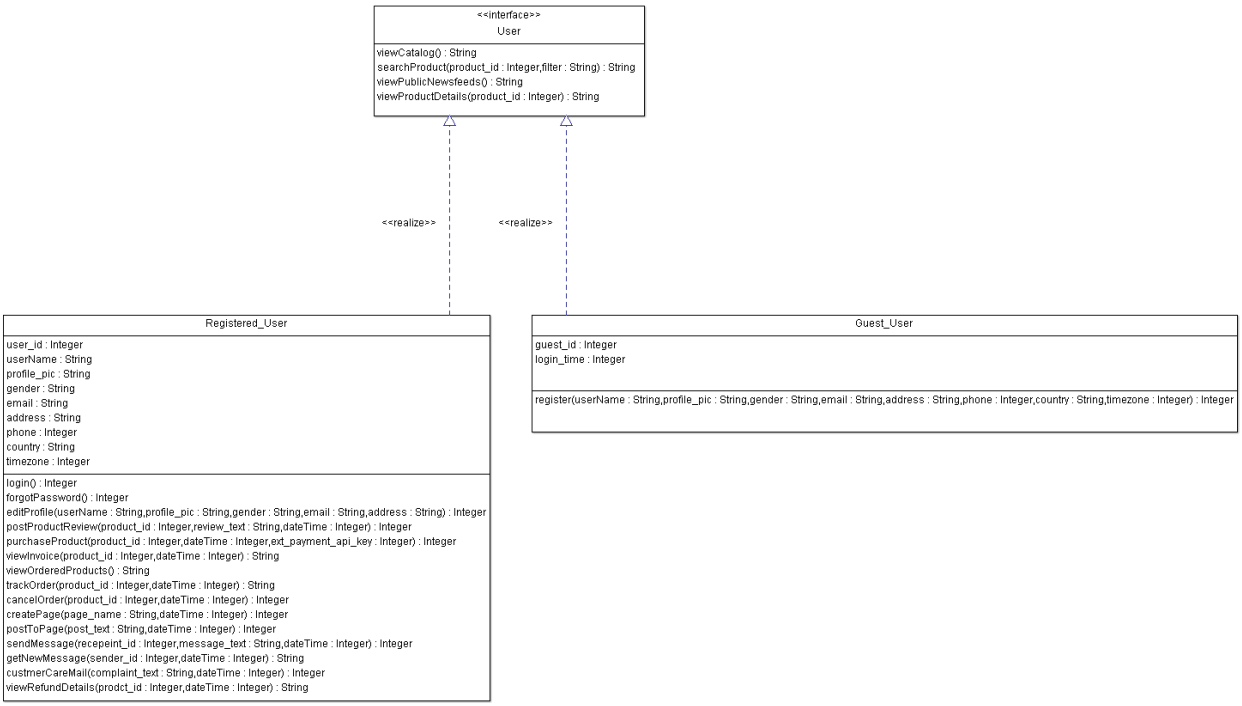
6. Product Functions

1. Show reviews
2. Post Reviews
3. Purchase
4. Checkout
5. View ordered products
6. Invoice Generation
7. Cancel order
8. Track Order
9. View refund details for canceled order

For a more detailed report on the components refer to section 3.2.

2.2. Structure and Relationships:

Here is a class diagram depicting our user classes and the functionality available to each one of them,



Since our project is not purely Object-Oriented we decided not to include our Catalog, Product, Page (Newsfeed) and User Posts as classes. We have decided to use a SQL database as our source of JSON data. The schemas related these tables are below,

Table – Catalog

Category_id (PRIMARY KEY)	Category_name	Creation_date	Poster_image

Table – Product

Product_id (PRIMARY KEY)	Product_name	Poster_image	Rating	Description	Stock	Price

Table – Category-Product

Product_id (PRIMARY KEY)	Category_id

Table – Product-Image

Product_id (PRIMARY KEY)	Product_image (PRIMARY KEY)

Table – Review

User_id (PRIMARY KEY)	Product_id (PRIMARY KEY)	Review_text	DateTime	Rating

Table – Newsfeed

Page_name (PRIMARY KEY)	DateTime	Poster_image	Description

Table – Users-Page

Page_name (PRIMARY KEY)	User_id

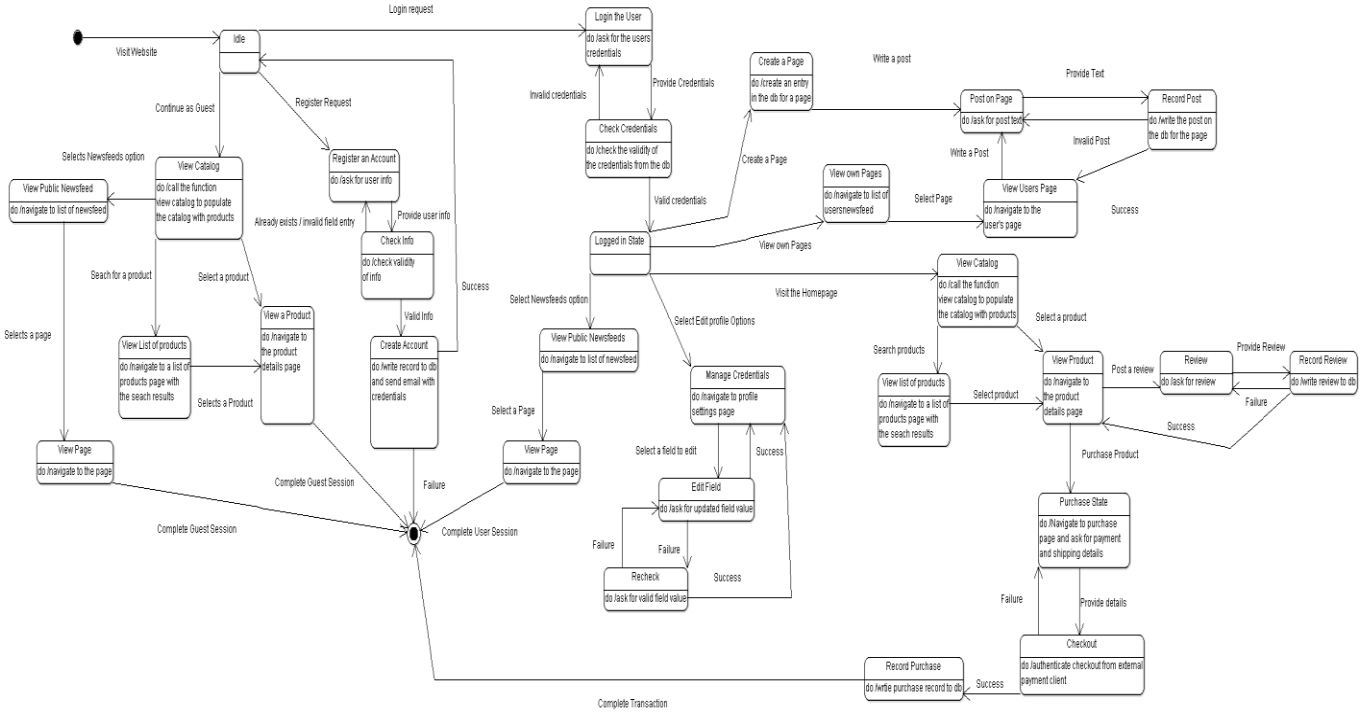
Table – Posts

User_id (PRIMARY KEY)	Page_name (PRIMARY KEY)	Post_text	DateTime

Table – Orders

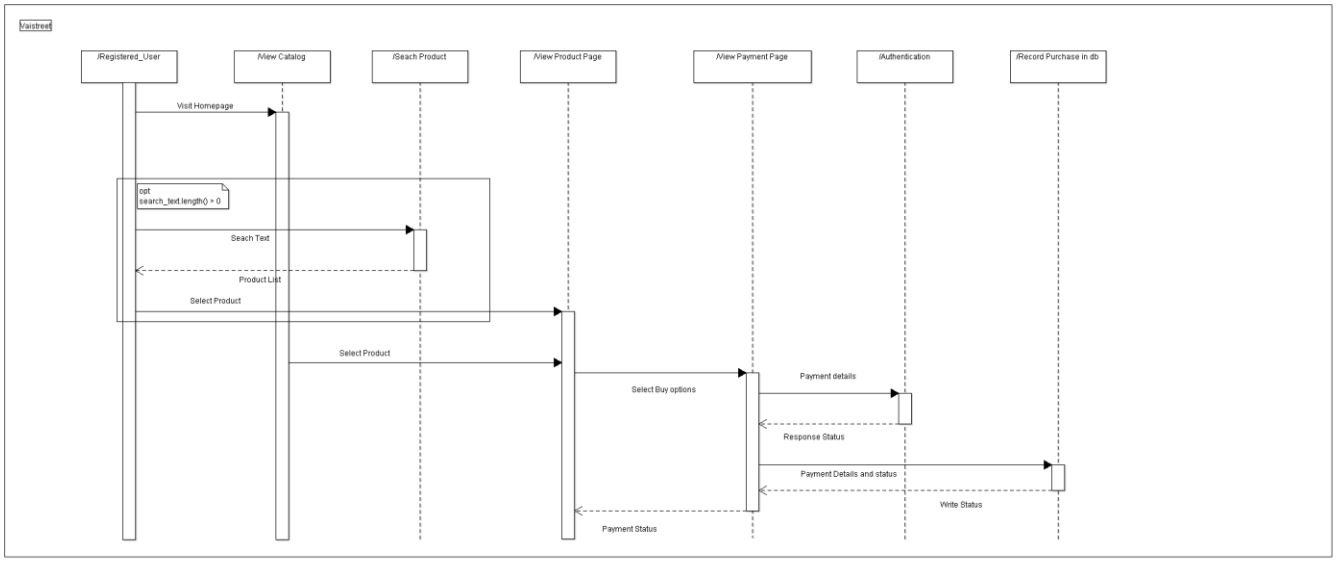
User_id (PRIMARY KEY)	Product_id (PRIMARY KEY)	DateTime (PRIMARY KEY)	Status	Quantity	Amount	Address	Payment_type

In order to demonstrate our product we have attached a State diagram and a Sequence Diagram below,



The state diagram can also be found in the Github repository as an image file named state_diagram.png. For the sake of readability we have removed certain transitions. The Guest user during any time in his session may choose to register. A User at any time may choose to leave the page.

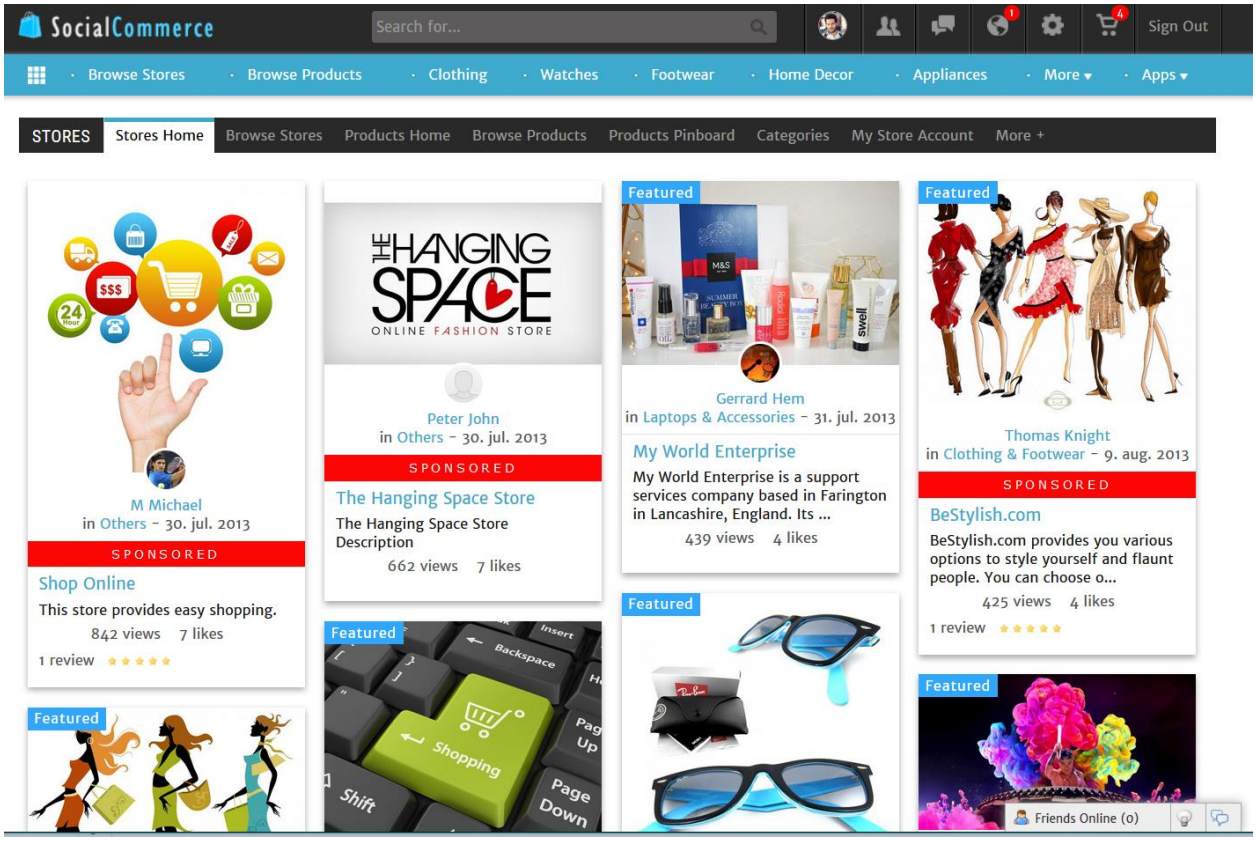
Since the State Diagram explains most of the flow in our website but we wanted to make the Purchase sequence clear to the readers, hence we have attached a sequence diagram for the Purchase flow



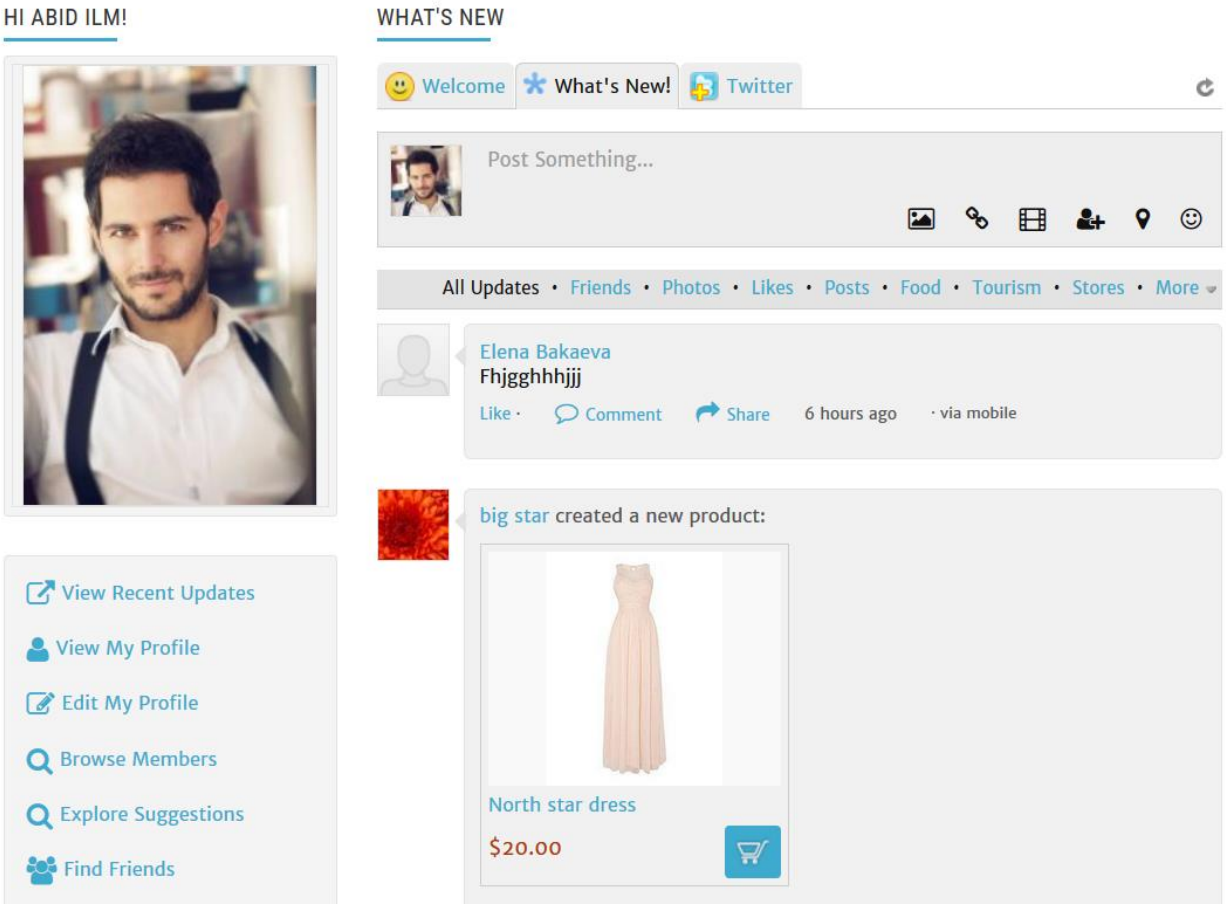
The sequence diagram image can also be found in the Github repository as purchase_sequence.png

2.3. User Interface Issues

The product has a common interface for both registered and unregistered users. This interface includes the product catalog, product details page, product reviews and public newsfeeds. The image below depicts this common interface.



Features available only for registered users will have a separate interface in addition to the common interface. This interface includes all the added functionalities specified under the registered user class in User Classes and Characteristics. Below is an image that depicts this interface.



3. Detailed Description of Components

3.1. Component Template Description

Identification	The unique name for the component and the location of the component in the system.
Type	A module, a subprogram, a data file, a control procedure, a class, etc
Purpose	Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements.
Function	What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.
Subordinates	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
Dependencies	How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.
Interfaces	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.
Processing	The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic.
Data	For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.

3.2. Components

View Catalogue:-

Identification	View catalogue
Type	Module
Purpose	To show the users a list of all products available
Function	It reads from a database a list of relevant products
Subordinates	A database read operation
Dependencies	Depends on the read operation
Interfaces	A return value (list of all the products in the form of a catalog)
Resources	No resources except online connectivity to trigger the read operation
Processing	Processes the data to be read from the database
Data	JSON object containing products across all categories

Search:-

Identification	Search
Type	Module
Purpose	To help the user search for a specified product
Function	It takes the search query as argument and reads the db, then returns a list of suitable results
Subordinates	A database read operation
Dependencies	Depends on the read operation
Interfaces	One arguments and one return value (list of relevant products)
Resources	No resources except online connectivity to trigger the read operation
Processing	Processes the data to read from the db and returns relevant results
Data	One string, search query

View Product:-

Identification	View Product
Type	Module
Purpose	To show the users a product and it's details
Function	It takes the product id as argument and reads from a database the details of the product
Subordinates	A database read operation
Dependencies	Depends on the read operation
Interfaces	One arguments and one return values
Resources	No resources except online connectivity to trigger the read operation
Processing	Processes the data to be read from the database
Data	JSON object for the product and an integer (product id).

View Feed:-

Identification	View Feed
Type	Module
Purpose	To show the users a list of relevant data
Function	It reads from a database the details of relevant data
Subordinates	A database read operation
Dependencies	Depends on the read operation
Interfaces	A return value
Resources	No resources except online connectivity to trigger the read operation
Processing	Processes the data to be read from the database
Data	JSON array containing list of all pages.

Browse feed:-

Identification	Browse feed
Type	Module
Purpose	To show the users a list of posts based on search query
Function	It takes the search as argument and reads from a database the list of relevant posts
Subordinates	A database read operation
Dependencies	Depends on the read operation
Interfaces	One arguments and multiple return values
Resources	no resources except online connectivity to trigger the read operation
Processing	processes the data to be read from the database
Data	JSON array of list of feeds

Login:-

Identification	Login
Type	module
Purpose	To help the user to gain access to his account
Function	It takes the userid and password as an argument and reads from a database if it matches
Subordinates	a database read operation
Dependencies	depends on the read operation
Interfaces	two arguments and one return value (success/fail)
Resources	no resources except online connectivity to trigger the read operation
Processing	processes match if the password entered matches as the one in the database
Data	Two strings as arguments and one integer return value.

Sign up:-

Identification	Sign up
Type	module
Purpose	To help the user to create his account
Function	It takes the name, email id, password & dob as an argument and writes it on the database
Subordinates	a database write operation
Dependencies	depends on the write operation
Interfaces	four arguments and one return value (success/fail)
Resources	no resources except online connectivity to trigger the read operation
Processing	processes match if the email id matches as the one in the database
Data	The JSON object to write to the database

OTP:-

Identification	OTP
Type	module
Purpose	To help the authenticate the user
Function	It takes the otp password as an argument and reads from a database if it matches
Subordinates	a database write operation
Dependencies	depends on the write operation
Interfaces	one arguments and one return value (success/fail)
Resources	no resources except online connectivity to trigger the read operation
Processing	processes match if the otp password entered matches as the one in the database
Data	

Forgot Password:-

Identification	Forgot Password
Type	module
Purpose	To help the user to reset the password to his account
Function	It takes the userid and as an argument and reads from a database if matches, if it does sends reset link to corresponding email id
Subordinates	a database write operation
Dependencies	depends on the write operation
Interfaces	one arguments and one return value (success/fail)
Resources	no resources except online connectivity to trigger the read operation
Processing	processes match if the user id entered matches as the one in the database, if it does sends reset link to corresponding email id
Data	A string

Manage Credentials:-

Identification	Name-Manage credentials Location – Profile info
Type	A data file
Purpose	To store info of the user
Function	Security and authentication. It stores the details of the customer (e-mail , phone no. ,address etc)
Subordinates	Email, password, phone no. etc. Security and credentials
Dependencies	Used for login and depends on read operation
Interfaces	Return data file including all the information
Resources	no resources except internet connectivity
Processing	Processes the data to be read from database
Data	A string

Edit user profile info:-

Identification	Name- Edit Profile Location-Profile info
Type	A Sub program
Purpose	To allow user to change credentials
Function	Flexibility and maintenance of correct user information. User can change phone no. ,email or any other information
Subordinates	Email, password, phone no. etc. Security and credentials
Dependencies	User details for delivery, login and other functions, depends on read and write operation
Interfaces	Return data file which can be edited and saved
Resources	no resources except internet connectivity
Processing	Processes the data to be read and written on database
Data	Two strings. One to read and second one to add or change data

Show Reviews:-

Identification	Name-Show reviews Location – on product page
Type	A module
Purpose	Transparency better website operation
Function	Better UI and Customer satisfaction. Customer can see product quality based on previous buyers so it is verified.
Subordinates	Performance and review
Dependencies	depends on the read operation
Interfaces	return data file consisting of all previous entries
Resources	no resources except internet connectivity

Processing	Processes the data to be read from database
Data	A string

Post reviews:-

Identification	Name-Post reviews Location – ordered products page
Type	Sub-program
Purpose	User experience details and improvement
Function	Better UIand customer experience review for improvement. user can share his/her experience about product which will help for other buyers as well as seller
Subordinates	Performance and review
Dependencies	Depends on write operation
Interfaces	Add data to an existing file
Resources	no resources except internet connectivity
Processing	Processes the data to be written to database
Data	A string

Create Newsfeed Post:-

Identification	Name-Newsfeed post Location – on website home page
Type	A module
Purpose	News and updates of website
Function	Better UI and updates of latest changes. User can see latest updates about products or any new trends or changes
Subordinates	Newsfeed and performance
Dependencies	Depends on write operation
Interfaces	Add data to an existing file
Resources	no resources except internet connectivity
Processing	Processes the data to be written to database
Data	A string

Chat Box:-

Identification	Name-Chat Box Location – On website page
Type	A sub program
Purpose	Customer satisfaction and innovation
Function	Better UI and innovation to help customer buy required product. User can talk to each other and discuss their queries
Subordinates	Chat box and performance
Dependencies	-
Interfaces	an independent sub program which lets to different users talk

Resources	no resources except internet connectivity
Processing	-
Data	-

Send E-mail to customer care:-

Identification	Name-Customer care Location – under help page
Type	A sub program
Purpose	To solve problems and errors
Function	To solve problems and errors. User can report problems by sending mail.
Subordinates	Email, password, phone no. etc. Security and credentials
Dependencies	Used for login and depends on read operation
Interfaces	Return data file including all the information
Resources	no resources except internet connectivity
Processing	Processes the data to be read from database
Data	A string

Manage Credentials:-

Identification	Name-Manage credentials Location – Profile info
Type	A data file
Purpose	To store info of the user
Function	Security and authentication. It stores the details of the customer(e-mail , phone no. ,address etc)
Subordinates	Availability and maintainability
Dependencies	Depends on write operation
Interfaces	Creates and sends data file generated by customer
Resources	no resources except internet connectivity
Processing	Processes the data to be written and send to customer care
Data	A string

Purchase –

Identification	Name - purchase_product Location – Product Information Page
Type	A Module
Purpose	To facilitate the purchase of a product for the user
Function	It takes the user id as argument and writes to the database that a purchase has been made
Subordinates	A database write operation
Dependencies	Depends on the write operation
Interfaces	Two arguments and one return value (success or fail)
Resources	No resources except online connectivity to trigger the write operation

Processing	Processes the data to be written in the database to record a purchase
Data	Two strings – user_id and product_id along with a date variable

Checkout –

Identification	Name – checkout_cart Location – Product Information Page
Type	A Module
Purpose	To facilitate the checkout of the items added in the cart by the user
Function	It takes the user id as argument and writes to the database that a checkout has been made
Subordinates	A directed payment gateway portal provided by authorised partner PayU
Dependencies	Depends on the external server of the authorized payment portal
Interfaces	One argument and one return value (final list of all the produts in cart)
Resources	No resources except online connectivity to trigger the write operation
Processing	Processes the data to be transferred to the payment portal and the authorized partner to accept and verify it
Data	Two strings – user_id and product_id along with a date variable

View Ordered Products –

Identification	Name – ordered_products Location – My Orders Page
Type	A Module
Purpose	To help the user view all the products that the user has ordered
Function	It takes the user id as argument and reads from the database for the ordered products
Subordinates	A database read operation
Dependencies	Depends on the read operation
Interfaces	One return value (list of the ordered products)
Resources	No resources except online connectivity to trigger the write operation
Processing	Processes the data to be read from the database to show the ordered product list
Data	Two strings – user_id and product_id along

Invoice –

Identification	Name - invoice Location – My Orders Page
Type	A Module
Purpose	To help the user get the reciept of the recent purhase
Function	It takes the user_id and order_id as argument and writes to the database for the invoice of the purchased products
Subordinates	A database write operation
Dependencies	Depends on the write operation
Interfaces	Two arguments and one return value (success or fail)
Resources	No resources except online connectivity to trigger the write operation
Processing	Processes the data from the cart checkout domain and then writes it in the

	form of a bill
Data	Two strings – user_id and product_id along with a date variable

Cancel Order –

Identification	Name – cancel_order Location – My Orders Page
Type	A Module
Purpose	To help the user cancel the order that they have placed
Function	It takes user id and order id and writes to the database for cancelling the order
Subordinates	A database write operation
Dependencies	Depends on the write operation
Interfaces	Two arguments and one return value (success or fail)
Resources	No resources except online connectivity to trigger the write operation
Processing	Processes the data from the purchased list and then writes to the database as to which product to be cancelled
Data	Two strings – user_id and product_id along with a date variable

Track Order -

Identification	Name – track_order Location – My Orders Page
Type	A Module
Purpose	To help the user track the order that they have placed
Function	It takes user id and order id and writes to the database for tracking the order
Subordinates	A database write operation
Dependencies	Depends on the write operation
Interfaces	Two arguments and one return value (success or fail)
Resources	No resources except online connectivity to trigger the write operation
Processing	Processes the data from the purchased list and then writes to the database as to which product to be tracked
Data	Two strings – user_id and product_id along with a date variable

View Refund Details for Cancelled Order –

Identification	Name – refund_details Location – My Orders Page
Type	A Module
Purpose	To help the user get the refund details of the order that they have cancelled
Function	It takes user id and order id and writes to the database for getting refund details
Subordinates	A database write operation
Dependencies	Depends on the write operation
Interfaces	One return value(Refund information)
Resources	No resources except online connectivity to trigger the write operation

Processing	Processes the data from the cancelled list and notes the time at which the product is cancelled from the date of purchase and then reads from the database as to what amount to be refunded
Data	Two strings – user_id and product_id along with a date variable

4. Reuse and Relationships to other Products

Our team is familiar with the concept of not re-inventing the wheel when something is already available and usable. Hence, we have made use of a number of components both internal and external to our project. The internal components are login components and other modules that we have created and reused throughout our code. Some of the external components are as follows,

- Payment Authentication Client – PayU
- Bootstrap Framework
- Angular Framework
- Express Framework
- Open Source Framework from PrestaShop
- Auth0 authentication SDK
- Social-CMS module

5. Design Decisions and Tradeoffs

Our team decided to use Auth0's Lock for the login model in our website. We believe that Auth0 provides security and enables diversity as it accepts users from several platforms (Facebook, Google, etc.). PayU is also a reliable and widely used payment client. We decided not to use a NoSql database as we wanted our schema to be well defined and realized that we did not require a NoSql database. Our team has more experience with Bootstrap and Angular hence we decided to use these technologies to get an early start.

6. Pseudo Code for components

The Pseudo Code for each of the Components is a Javascript Pseudo Code and the $scope variables are variables that have document scope of our Webpage.

Component – **View Catalog**
Pseudo Code –

```
Function () {

    //Query the database for all products in every category
    //Retrieve all the items from the Catalog and Product Table
    //Retrieve all the ids from the Catalog-Product table
    //Populate an object (catalog) which stores each product with its category details

    //Store result in the object catalog
```

```
    $scope.catalog = catalog

}
```

Component – **Search Product**
Pseudo Code –

```
Function (string product_name) {

    //Query the Product table for the product_name
    //Retrieve the category details for each product
    //Create an object (list) that stores each product with its category details

    //Store result in the object product_list
    $scope.product_list = list

}
```

Component **– View a Product**
Pseudo Code -

```
Function (int product_id) {

    //Query the Product table for the product_id
    //Retrieve the images from the Product-Images Table for the product
    //Retrieve the reviews from the Reviews Table
    //Create an object (product) that stores the product with its images and reviews

    //Store result in the object product
    $scope.product = product

}
```

Component – **View Newsfeed**
Pseudo Code –

```
Function () {

    //Query the Newsfeed table for all the pages
    //Create an object (page_list) that stores the pages

    //Store result in the object page_list
    $scope.page_list = page_list

}
```

Component – **Login**
Pseudo Code –

Handled by the Auth-0 Lock module from the Auth-0 SDK.

Component – **SignUp**
Pseudo Code -

Handled by the Auth-0 Lock module from the Auth-0 SDK.

Component – **Forgot Password**
Pseudo Code –

Handled by the Auth-0 Lock module from the Auth-0 SDK.

Component – **OTP verification**
Pseudo Code –

Handled by the Auth-0 Lock module from the Auth-0 SDK.

Component – **Manage Credentials**
Pseudo Code -

```
Function () {

    $scope.profile_field_id contains the integer tag for the field in the profile page

    If( $scope.profile_field_id == 1 )// where 1 is the tag for Name
    {
            //accept new value
            //write new value to the Database
            If( write is successful )
            {
                    //modify the object of the webpage to reflect the change
                    Return;
            }
            Else
                    Return error;
    }
}
```

Component – **Create Newsfeed**
Pseudo Code –

```
Function () {

    //Accept page name, image and description from user
    //Record the Page name, Date, Image and description in the Newsfeed Table
    //Update the Users-Page Table with user_id and Page_name
    //if successful writes occur return 1 else 0

}
```

Component – **Post on newsfeed**
Pseudo Code –

```
Function () {

    //accept the post text from user
    //write the post to the Posts Table
    //if successful return 1 else 0

}
```

Component – **Send Message**
Pseudo Code –

Handled by the Socket.io framework and the Pubnub javascipt SDK, but basic design approach taken is that when a user sends a message it is published across all clients. Each client is subscribed to these posts using pubnub channels

```
Function () {

    //accept user message and recipient user name
    //query for the user_id using the recipient user name from the Users Table

    //publish the message using the user_id, the message and the channel is the recipient's user_id

}
```

Component – **Receive Message**
Pseudo Code –

Agasin handled by the Socket.io framework and the Pubnub javascipt SDK, but basic design approach taken is that each client is subscribed to the posts using pubnub channels.

```
Function () {

    //subcribe to a user_id
    //listen for messages from this channel
    //for every message append to chat

}
```

Component – **Notify customer care**
Pseudo Code –

Publish the complaint to the customer care channel via pubnub. The response from the customer care is also from pubnub.

```
Function () {

    //accept user message

    //publish the message using the user_id, the message and the channel is the customer care's channel

}
```

Component – **Post review**
Pseudo Code -

```
Function (int product_id) {

    //accept review text, rating from user
    //write the review to the Reviews Table
    If( write successful )
    {
            //reflect the change on the $scope object for the webpage
            Return 1;
    }
    Else
            Return 0;
```

```
}
```

Component – **Purchase**
Pseudo Code -

```
Function (int product_id) {

    //accept quantity, payment type from user activity (clicking on button)
    //send the payment form to the PayU API using a post request
    If( transaction is successful )
    {
            //Write to Orders table with Status Ordered
            If ( write successful )
                    Return 1;
            Else
            {
                    Return 0;
            }
    }
    Else
    {
            Return 0;
    }
}
```

Component – **View Ordered products**
Pseudo Code -

```
Function () {

    //retrieve all the orders from the Orders Table
    //Store the object in the scope variable

    $scope.order_list = order_list

}
```

Component – **Cancel Order**
Pseudo Code -

```
Function (int product_id, int dateTime) {

    //update the Status column in the Orders table for the product_id, user_id and dateTime to Canceled
    If( successful )
            Return 1;
    Else
```

```
        Return 0;
}
```

Component – **View Canceled Order**
Pseudo Code -

```
Function () {

    //retrieve all the orders from the Orders Table with Status as Canceled
    //Store the object in the scope variable

    $scope.order_list = order_list
}
```