

# JANGS

## Technical Documentation Version 4.0

**Project Manager:** *Ali Hassan* (*ali.x.hassan@abo.fi*)

**Project Members:**

*Arshid Nippon* (*arshid.nippon@abo.fi*)  
*Janani* (*janani.natarajan@abo.fi*)  
*Yan Gao* (*yan.gao@abo.fi*)  
*Sajjad Majumdar* ([md.majumdar@abo.fi](mailto:md.majumdar@abo.fi))

### Revision History

Date	Version	Description	Author
October 11, 2023	1.0	Technical Documentation V1	Ali, Janani, Gao, Sajjad, Nippon.
November 10, 2023	2.0	Technical Documentation V2	Ali, Janani, Gao, Sajjad, Nippon.
December 13, 2023	2.1	Technical Documentation V2.1	Ali, Janani, Gao, Sajjad, Nippon.
February 2, 2024	3.0	Technical Documentation V3.0	Ali, Janani, Gao, Sajjad, Nippon.
March 14, 2024	4.0	Technical Documentation V3.0	Ali, Janani, Gao, Sajjad, Nippon.

## Contents

<b>1 Product Requirements.....</b>	<b>3</b>
1.1 Requirements Specification.....	3
1.1.1 Functional Requirements.....	3
1.1.2 User Interface Requirements.....	9
1.1.3 Non-functional Requirements.....	10
<b>2 System Architecture and Design.....</b>	<b>11</b>
2.1 Architecture.....	11
2.2 Design.....	12
2.2.1 Static Design.....	12
2.2.2 Dynamic Design.....	13
2.2.3 Databases.....	15
2.2.4 UI Design Prototype.....	16
2.2.5 JANGS User Interfaces.....	17
<b>3 Implementation.....</b>	<b>22</b>
3.1 Technologies.....	22
3.1.1 React.js.....	23
3.1.2 Node.js.....	23
3.1.3 Express.js.....	23
3.1.4 MongoDB.....	23
3.1.5 Google Maps Api.....	24
3.2 Api Interfaces.....	25
3.2.1 Signup.....	25
3.2.2 LOGIN.....	26
3.2.3 Get User Profile.....	26
3.2.4 Upload Profile image.....	27
3.2.5 Get Products.....	28
3.2.6 Get Filter Options.....	30
3.2.7 Make Favorite.....	32
3.2.8 Get favorites.....	33
3.2.9 Remove Favorite.....	33
3.2.10 Make Reservations.....	34
3.2.11 Get Reservations.....	34
<b>4.0 References.....</b>	<b>35</b>

# 1 Product Requirements

Jangs is a web app designed to share information about clothes shopping in Turku. The system will serve as a preliminary, online guide for customers wanting to do physical shopping for their clothes.

Jangs will bridge the gap between online and physical shopping by connecting users with local stores nearby, helping them find in-store items quickly. Users can search for specific items, compare options from different shops, and visit the store for a seamless and more efficient shopping experience.

Our system would benefit those who want to have a fast yet pleasant shopping experience with the help of clear and comprehensive information. Jangs would also be very simple to use, no special technical skills or instructions are required. With just a few clicks, users can find all useful information in our system.

## 1.1 Requirements Specification

Requirements specification consist of functional requirements, non-functional requirements and user interface requirements. We mainly identify the requirements via technique analysis and communication with our customer.

### 1.1.1 Functional Requirements

<b>Requirement Id</b>	<i>REQ-1.1.1</i>
<b>Requirement description</b>	<i>All users should be able to log in and log out from the system.</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-

<b>Actors using the requirement</b>	<i>Users</i>
-------------------------------------	--------------

<b>Requirement Id</b>	<i>REQ-1.1.2</i>
<b>Requirement description</b>	<i>Consumers can type keywords (the type of outfits/brands) into the system to search for outfits</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	<i>-</i>
<b>Input</b>	<i>Keywords (the type of outfits/brands)</i>
<b>Output</b>	<i>All related items will be listed for consumer to view</i>
<b>Actors using the requirement</b>	<i>Users</i>

<b>Requirement Id</b>	<i>REQ-1.1.3</i>
<b>Requirement description</b>	<i>Consumers can select distance radius into the system to search for outfits</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	<i>-</i>
<b>Input</b>	<i>Consumers type the distance between their place to the shop</i>
<b>Output</b>	<i>Corresponding shops will be listed</i>
<b>Actors using the requirement</b>	<i>Users</i>

<b>Requirement Id</b>	<i>REQ-1.1.4</i>
<b>Requirement description</b>	<i>Consumers can register as a member and pay membership fee</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	<i>-</i>
<b>Input</b>	<i>Consumers pay as a member</i>
<b>Output</b>	<i>Members are registered and get their code</i>

<b>Actors using the requirement</b>	<i>Users</i>
-------------------------------------	--------------

<b>Requirement Id</b>	<i>REQ-1.1.5</i>
<b>Requirement description</b>	<i>Shops can scan membership code as completed.</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	<i>Merchants scan the code from the member</i>
<b>Output</b>	<i>The order completed</i>
<b>Actors using the requirement</b>	<i>Users</i>

<b>Requirement Id</b>	<i>REQ-1.1.6</i>
<b>Requirement description</b>	<i>Members can receive bonus once the shopping journey completed</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	<i>Depends on REQ 1.1.5</i>
<b>Input</b>	<i>The order completed</i>
<b>Output</b>	<i>Members receive bonus to their website account for next shopping</i>
<b>Actors using the requirement</b>	<i>Developer</i>

<b>Requirement Id</b>	<i>REQ-1.1.7</i>
<b>Requirement description</b>	<i>Members can mark products</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	<i>Members mark products</i>
<b>Output</b>	<i>The marked products will be collected in their profile page</i>

<b>Actors using the requirement</b>	<i>Users</i>
-------------------------------------	--------------

<b>Requirement Id</b>	<i>REQ-1.1.8</i>
<b>Requirement description</b>	<i>Consumers can access stores and look at products without logging in.</i>
<b>Requirement priority</b>	<i>2- medium</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Developer</i>

<b>Requirement Id</b>	<i>REQ-1.1.9</i>
<b>Requirement description</b>	<i>Consumers must sign in to mark items</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Developer</i>

<b>Requirement Id</b>	<i>REQ-1.1.10</i>
<b>Requirement description</b>	<i>Members can reserve the product for two hours</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	<i>Members choose to reserve the product</i>
<b>Output</b>	<i>Products will be reserved</i>

<b>Actors using the requirement</b>	<i>Members</i>
-------------------------------------	----------------

<b>Requirement Id</b>	<i>REQ-1.1.11</i>
<b>Requirement description</b>	<i>Consumers can check the shops name where sells outfits they want</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	<i>Consumers choose to check the place</i>
<b>Output</b>	<i>The name of the shop will be presented</i>
<b>Actors using the requirement</b>	<i>Users</i>

<b>Requirement Id</b>	<i>REQ-1.1.12</i>
<b>Requirement description</b>	<i>Members can check the location of shops in Google map</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	<i>Members choose to check the place</i>
<b>Output</b>	<i>The detailed address of the shop will be presented and members can check the location in Google maps</i>
<b>Actors using the requirement</b>	<i>Members</i>

<b>Requirement Id</b>	<i>REQ-1.1.13</i>
<b>Requirement description</b>	<i>Members can check the shelf number of products</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-

<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Members</i>

<b>Requirement Id</b>	<i>REQ-1.1.14</i>
<b>Requirement description</b>	<i>Members can check the discounted products which will be displayed in the sale area</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Members</i>

<b>Requirement Id</b>	<i>REQ-1.1.15</i>
<b>Requirement description</b>	<i>The user must be able to choose between a English, Finnish and Swedish interface</i>
<b>Requirement priority</b>	<i>3- low</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	<i>Choose language</i>
<b>Output</b>	<i>Contents on the website will be translated to corresponding language</i>
<b>Actors using the requirement</b>	<i>Users</i>



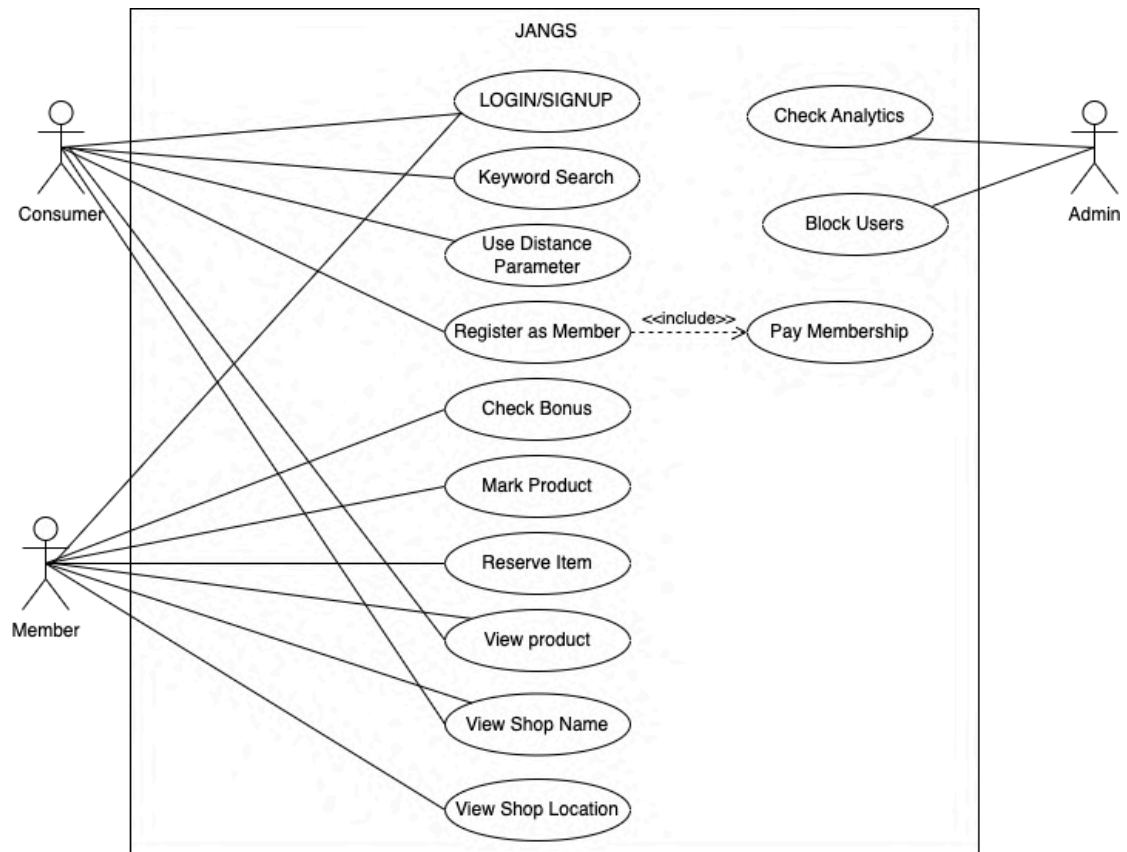


Figure 1.1 Use case Diagram

Figure 1.1 is a use case diagram for JANGS. JANGS has 3 types of actors: Consumer(non-member), Member, Admin. Each Actor has different rights in the system, some use cases are shared among two actors.

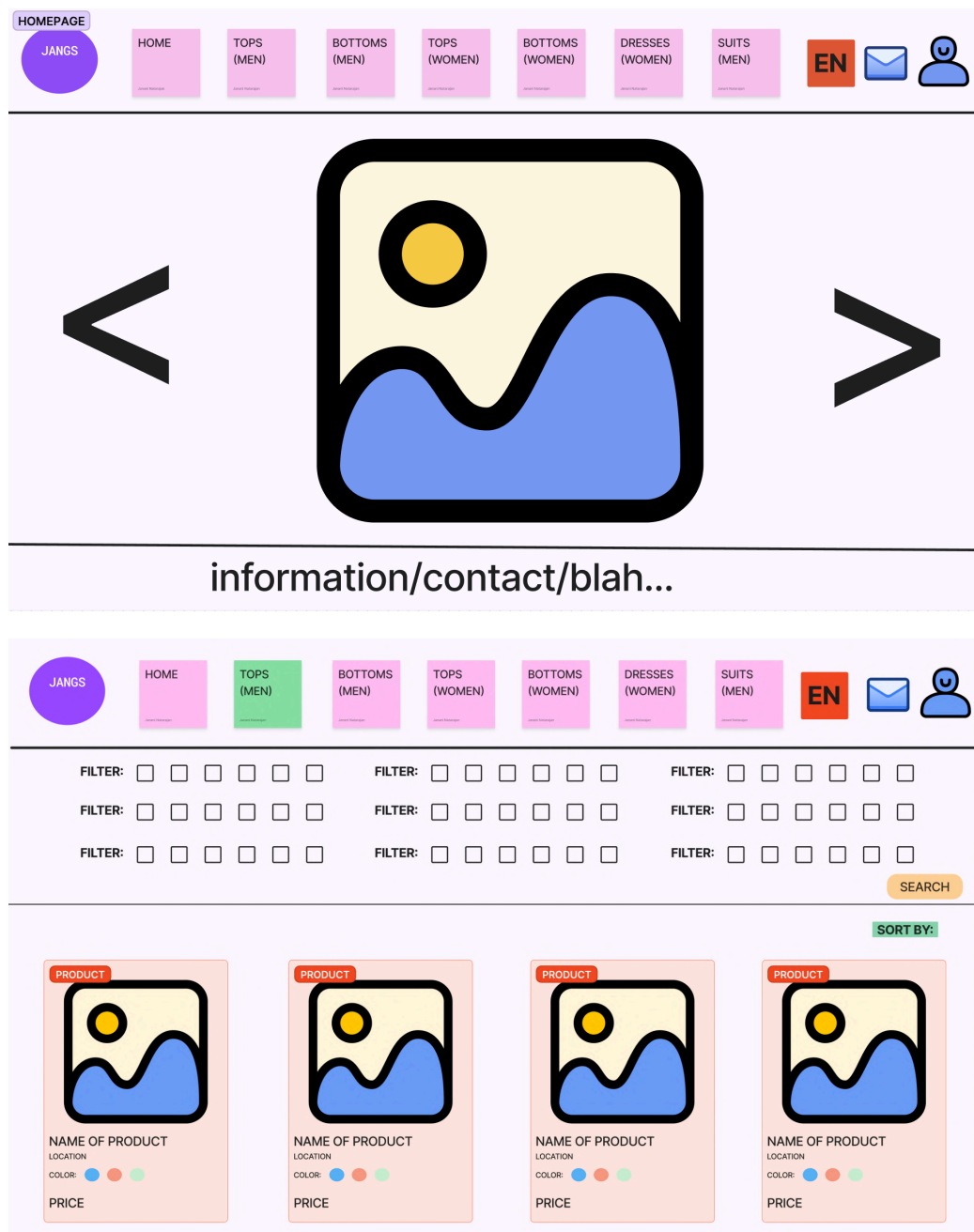
### 1.1.2 User Interface Requirements

The UI of the system has to ensure the following requirements-

- System should be simple, intuitive and efficient.
- Input controls such as checkboxes, radio buttons, dropdown lists, dropdown buttons, toggles, and text fields should ensure smooth experience.
- Navigational (filters, image carousels) & Informational (product status,stocks, message boxes) should be available.
- Visual elements such as images of the products should be included.
- Advertisements and Offers from partners can be added to the landing page.

It is also important to state what is correct and what is incorrect user behavior and how the system should react to the incorrect behavior. For example, if a

user enters incorrect login credentials, the system should display an error message and prompt the user to enter the correct information. Similarly, if a user tries booking an out-of-stock item, the system should display a message indicating that the item is out of stock and suggest similar items that are available.



### 1.1.3 Non-functional Requirements

<b>Requirement Id</b>	<i>REQ-1.2.1</i>
<b>Requirement description</b>	<i>The application is simple in use</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Users</i>

<b>Requirement Id</b>	<i>REQ-1.2.2</i>
<b>Requirement description</b>	<i>The web app is secure and reliable to use</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Users, developers</i>

<b>Requirement Id</b>	<i>REQ-1.2.3</i>
<b>Requirement description</b>	<i>Consumers can have broad offerings to meet their needs</i>
<b>Requirement priority</b>	<i>1- high</i>
<b>Requirement dependencies</b>	-
<b>Input</b>	-
<b>Output</b>	-
<b>Actors using the requirement</b>	<i>Users</i>

## 2 System Architecture and Design

### 2.1 Architecture

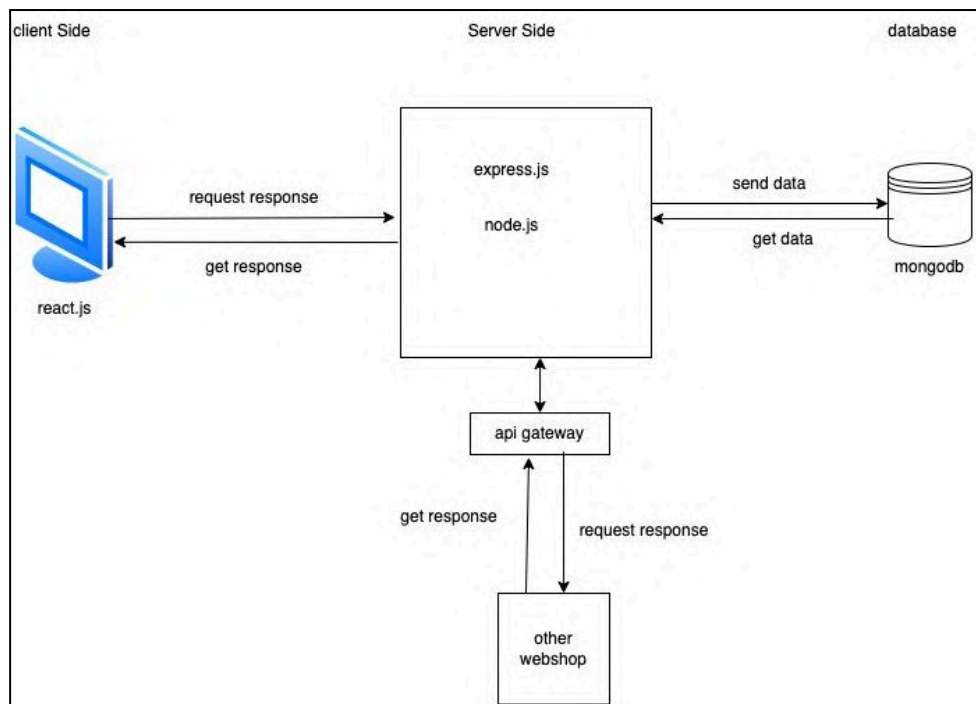


Figure 2.1 JANGS System Architecture

Figure 2.1 shows JANGS system architecture. We are using client-server architecture. react.js will be used to implement front end(client side), express.js and node.js will be used for backend server, runtime environment and for api development. We will use mongodb as a database. Our back-end will have an option to connect with shop api's(future implement). we will use an api gateway to communicate with all the 3rd party api's. Back-end will use google maps api for navigation related functionality.

## 2.2 Design

### 2.2.1 Static Design

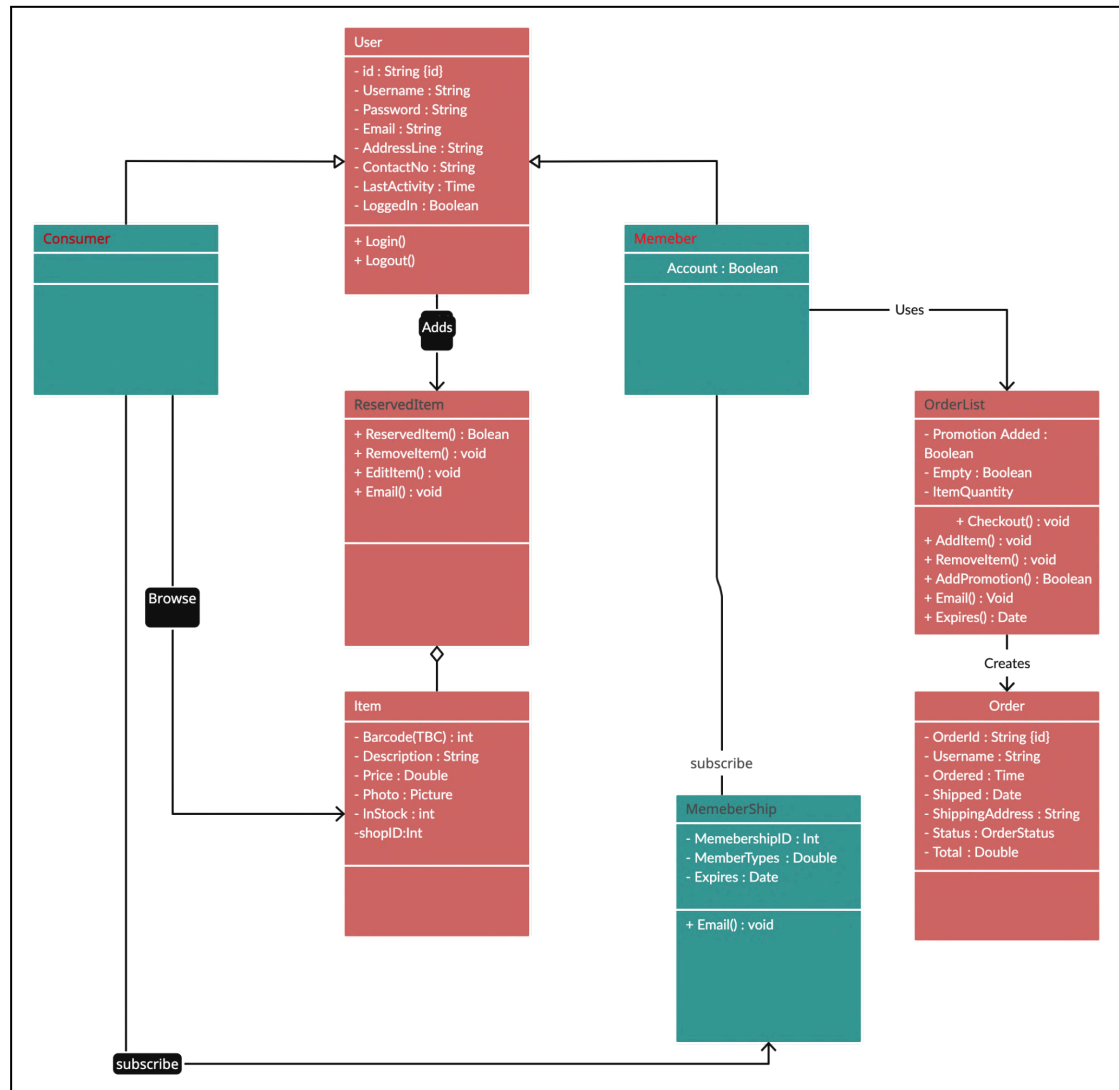


Figure 2.2.1 JANGS Class Diagram

Figure 2.2.1 shows the class diagram of JANGS . We have 8 classes. Some classes use another class to perform specific features.

## 2.2.2 Dynamic Design

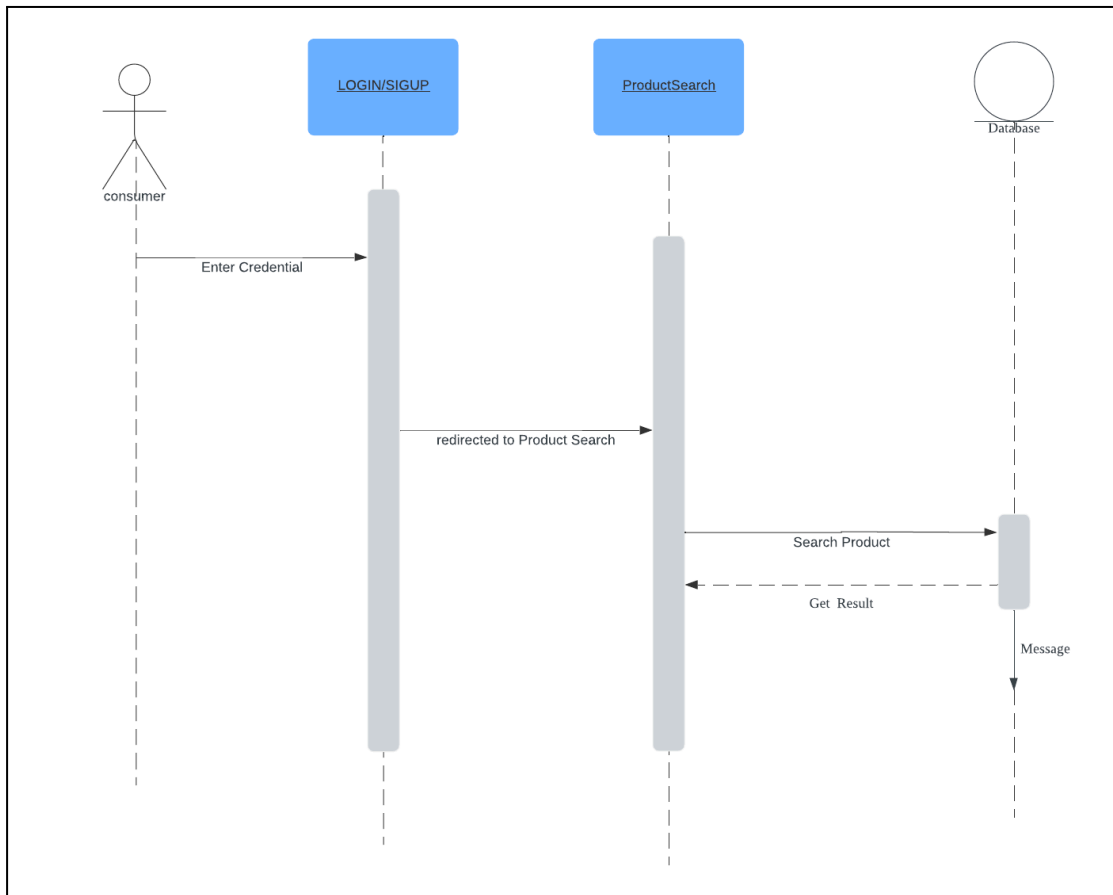


Figure 2.2.2.1 JANGS Consumer (Non-Member) Sequence diagram

Figure 2.2.2.1 is JANGS Consumer(non-member) sequence diagram. Consumers will login in the system to search for products. Consumers will use different parameters to search products in the system, the system will check in the database and will show results in the product Search page.

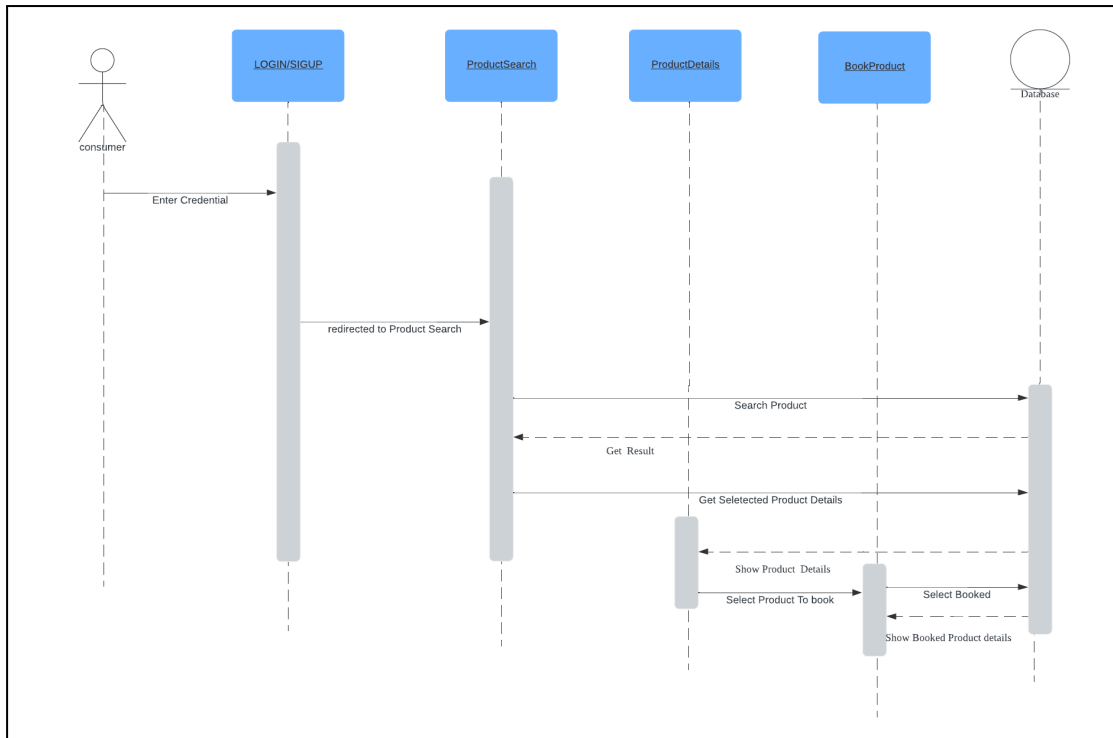


Figure 2.2.2.2 JANGS Member Sequence diagram

Figure 2.2.2.2 is JANGS Member sequence diagram. Members will login in the system to search for products. Members will use different parameters to search products in the system, the system will check in the database and will show results in the product Search page. Members are also able to see product details and book the product for certain times to collect from the shop.

### 2.2.3 Databases

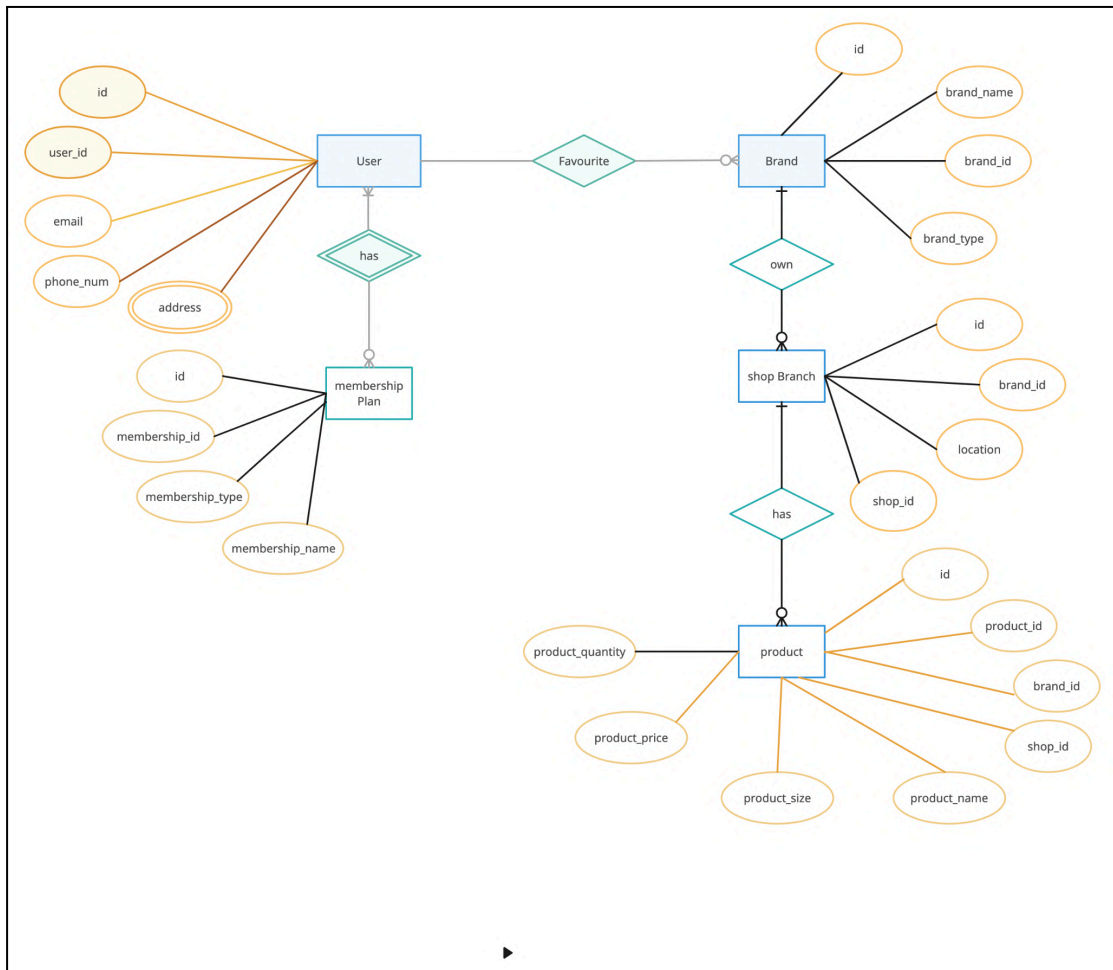


Figure 2.2.3 ERD of JANGS

Figure 2.2.3 is Entity Relationship diagram(ERD) of JANGS. It shows the relationship between entities and their attributes.



## 2.2.4 UI Design Prototype

The [UI design prototype](#), meticulously crafted on Figma, serves as a preview of the forthcoming web application. Our navigation bar intuitively organizes product categories for both women and men, further subcategorized into dresses, suits, tops, and bottoms. Upon entering a specific category, exemplified in Figure 2.2.2, users will encounter a consistent interface across all categories.

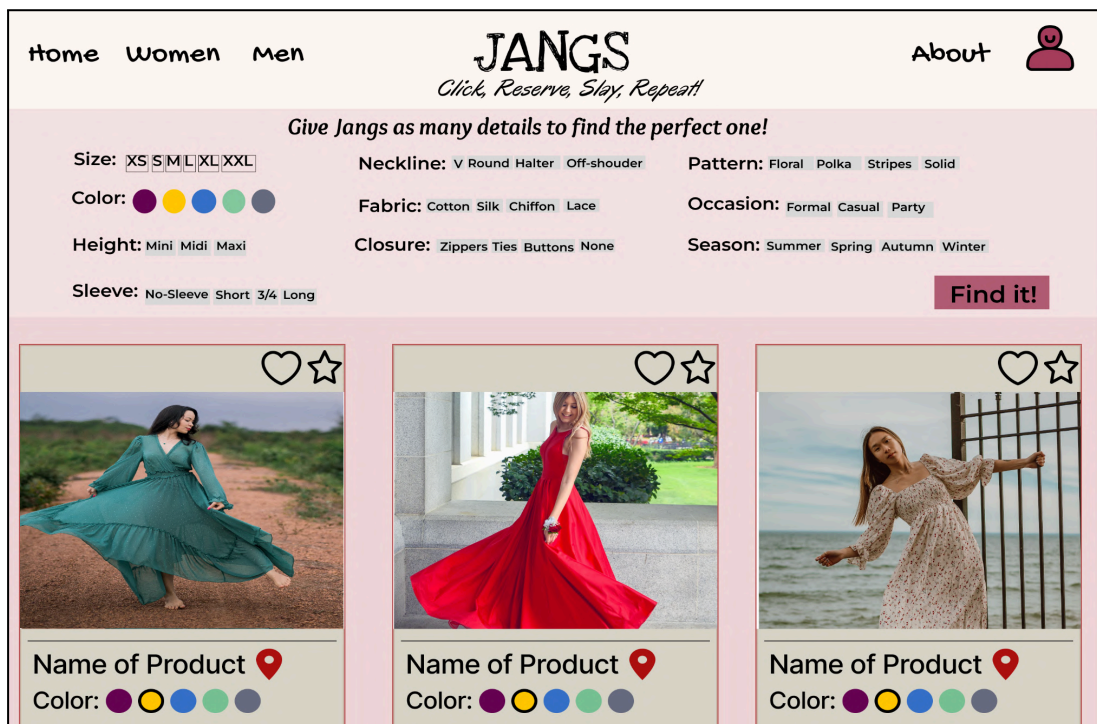


Figure 2.2.2 Screenshot of Figma Prototype

Notably, our advanced filter option empowers users to provide intricate details about the desired product, enhancing search precision. The more details users provide, the more refined and tailored their search results become, ensuring an optimal shopping experience. Additionally, the prototype includes sample pages for login, signup, user profiles, and an about section, contributing to a comprehensive and detailed representation of the future web application. This design underscores our commitment to delivering a sophisticated, user-centric, and seamlessly navigable platform for an enhanced shopping journey.

## 2.2.5 JANGS User Interfaces

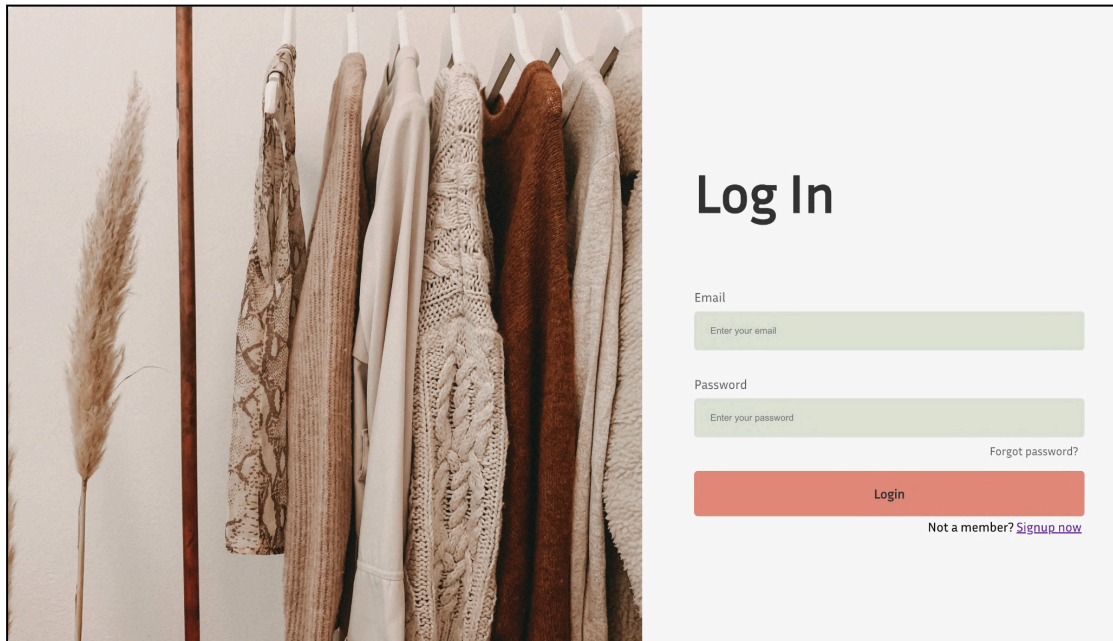
The login interface is displayed on a light gray background. On the left, there is a vertical image of a pampas grass plant and a row of various knitted garments hanging on a wooden hanger. The text 'Log In' is prominently displayed in a large, bold, black font. Below this, there are two input fields: 'Email' with the placeholder text 'Enter your email' and 'Password' with the placeholder text 'Enter your password'. A link 'Forgot password?' is located to the right of the password field. A red 'Login' button is positioned below the input fields. At the bottom right, there is a link 'Not a member? [Signup now](#)'.

Figure 2.2.3 JANGS User login interface.

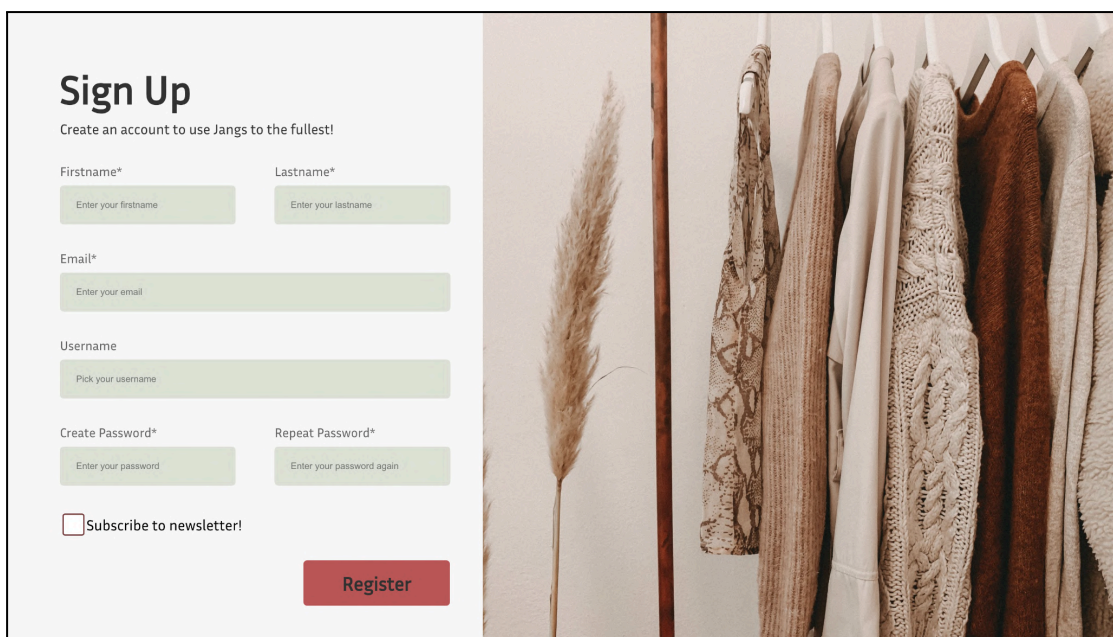
The signup interface is displayed on a light gray background. On the right, there is a vertical image of a pampas grass plant and a row of various knitted garments hanging on a wooden hanger. The text 'Sign Up' is prominently displayed in a large, bold, black font. Below this, there is a sub-header 'Create an account to use Jangs to the fullest!'. The form includes several input fields: 'Firstname\*' with placeholder 'Enter your firstname', 'Lastname\*' with placeholder 'Enter your lastname', 'Email\*' with placeholder 'Enter your email', 'Username' with placeholder 'Pick your username', 'Create Password\*' with placeholder 'Enter your password', and 'Repeat Password\*' with placeholder 'Enter your password again'. There is a checkbox labeled 'Subscribe to newsletter!'. A red 'Register' button is located at the bottom right of the form.

Figure 2.2.3 JANGS User Signup interface.

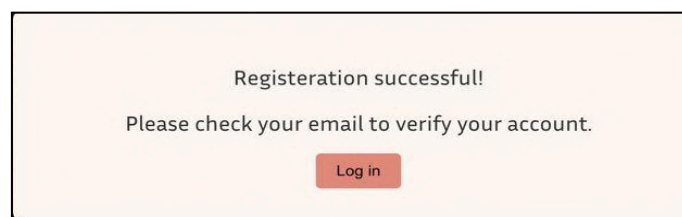
The confirmation interface is a light orange rectangular box. It contains the text 'Registration successful!' in a bold black font, followed by 'Please check your email to verify your account.' in a regular black font. At the bottom center, there is a red 'Log in' button.

Figure 2.2.4 JANGS User Signup Confirmation interface.

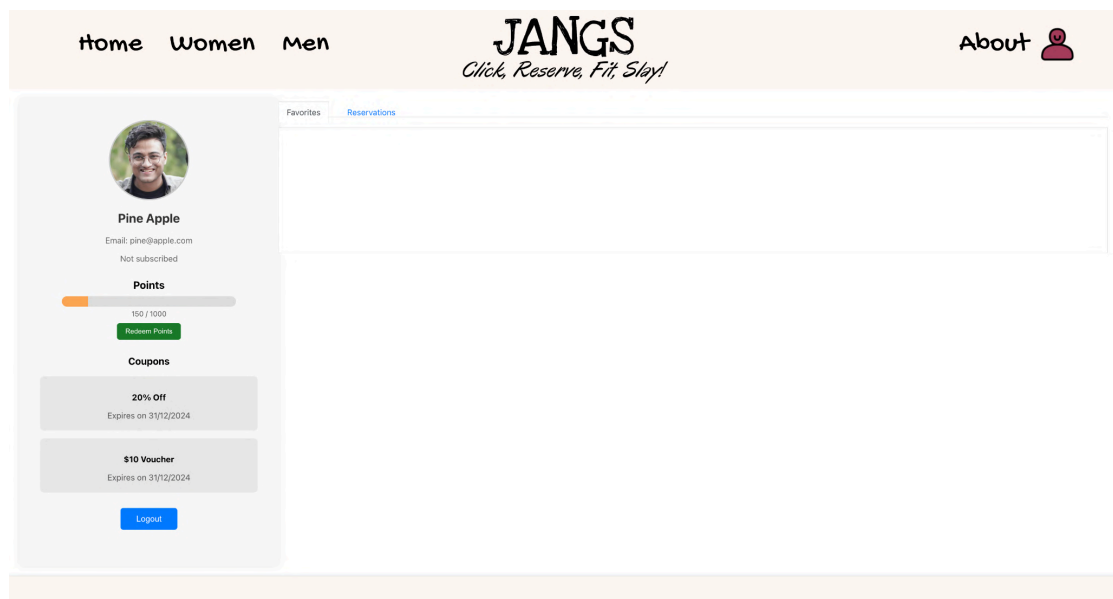


Figure 2.2.5 JANGS User Profile interface.

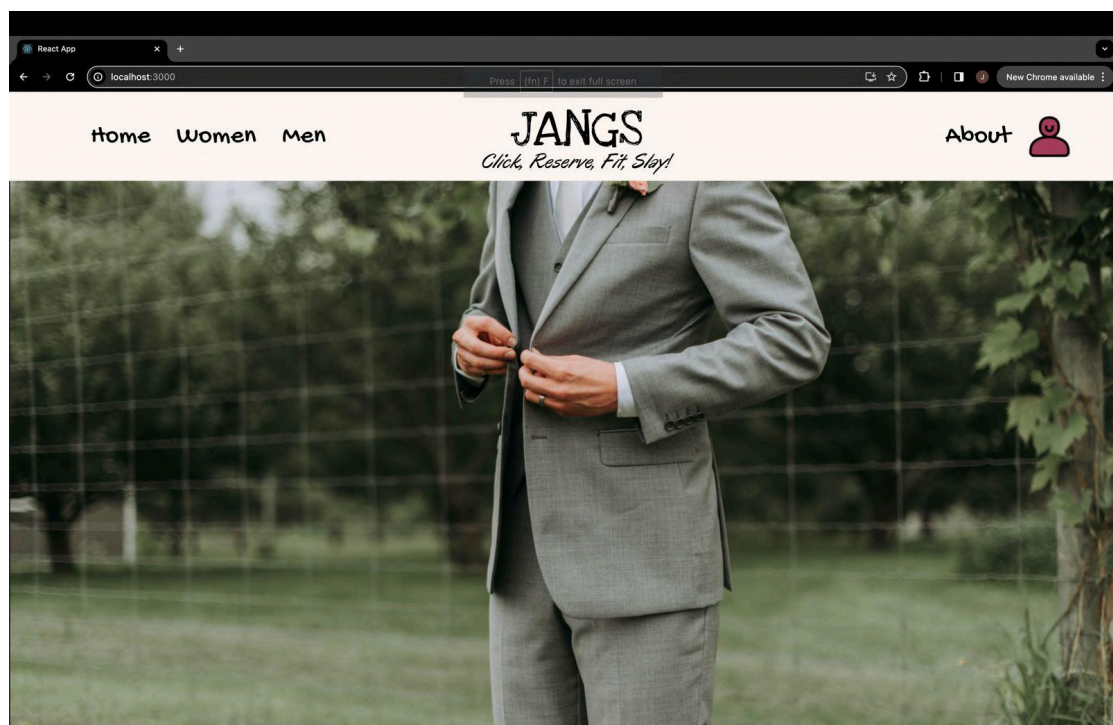


Figure 2.2.6 JANGS Landing interface.



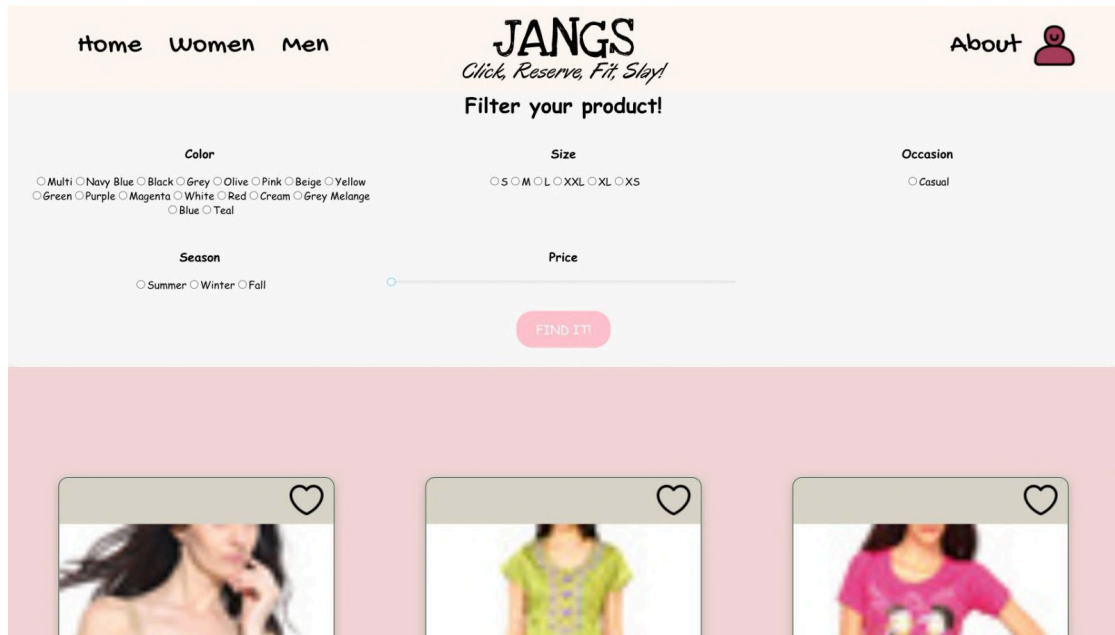


Figure 2.2.7 JANGS Products Page with Filters.



Figure 2.2.8 Product Card.

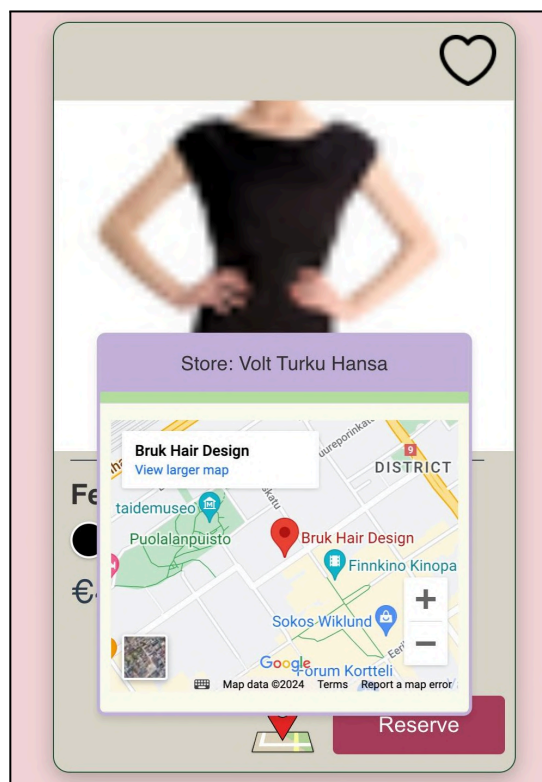


Figure 2.2.9 Store Location.



Figure 2.2.10 Reservation Pop Up

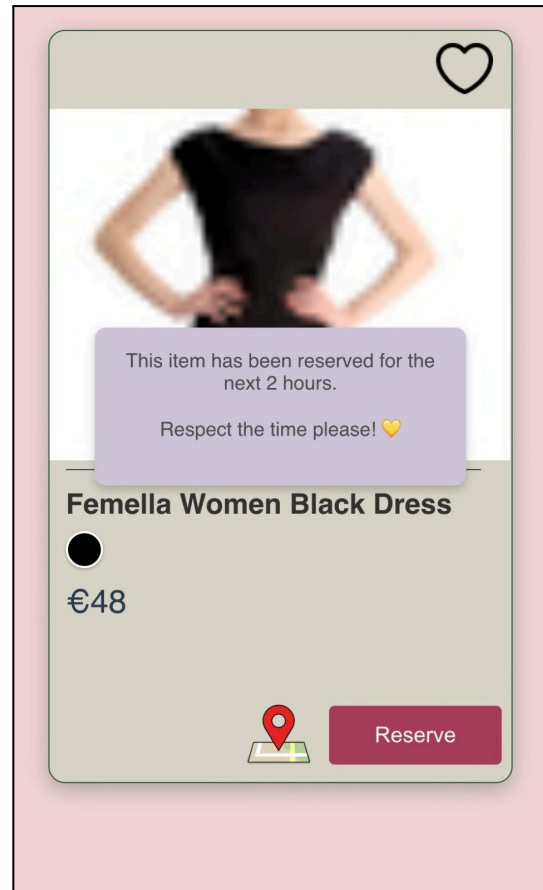


Figure 2.2.11 Confirmation

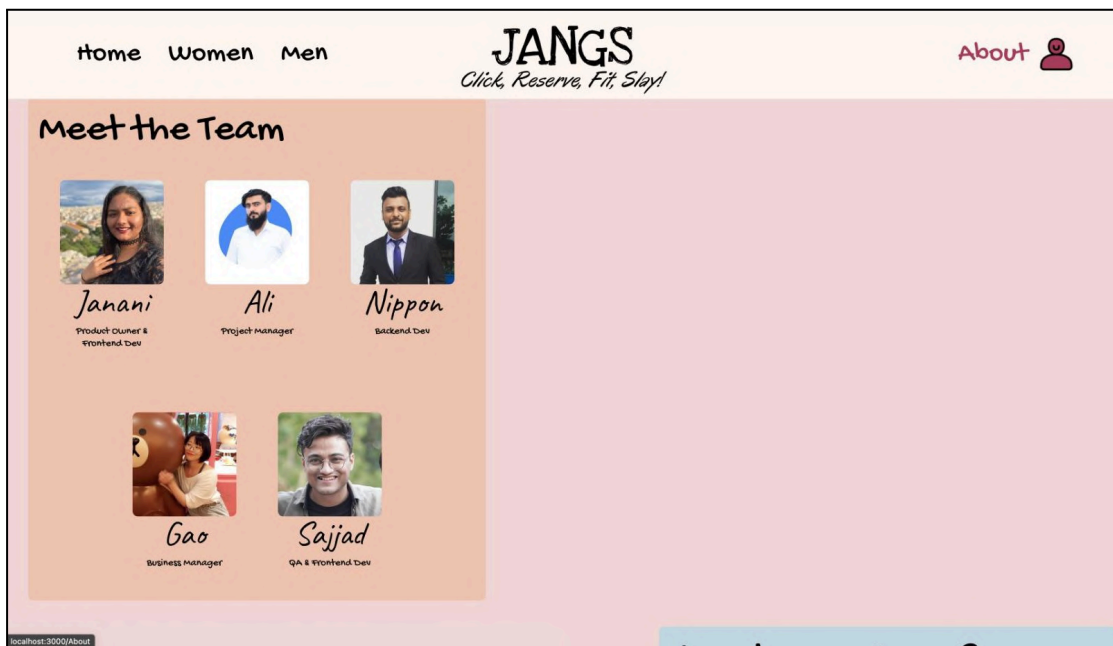


Figure 2.2.12 JANGS About us

## 3 Implementation

### 3.1 Technologies

This system will be implemented using react.js, mongodb, node.js, express.js , google maps api. Our system is a web based application. We are using client-server architecture. react.js will be used to implement front end(client side), express.js and node.js will be used for backend server, runtime environment and for api development. We will use mongodb as a database. Our back-end will have an option to connect with shop api's(future implement). we will use an api gateway to communicate with all the 3rd party api's. Back-end will use google maps api for navigation related functionality.

Name	Purpose
React.js	Front End Framework
Node.js	Back-end runtime environment and api development.
Express.js	For the backend https server.
Mongodb compass	Development Database.
Visual studio Code	IDE
Google Maps Api	Maps and geolocation.

*Table 1 Development tools and framework*

#### 3.1.1 React.js

React.js, also known as React, is a popular JavaScript library for building user interfaces (UIs). It was developed by Facebook and released as an open-source project in 2013. React allows developers to create reusable UI components and build complex UIs by composing these components.

The main concept behind React is the Virtual DOM (Document Object Model). Instead of directly manipulating the browser's DOM, react creates a virtual representation of the UI in memory. This virtual representation, called the Virtual DOM, allows React to efficiently update and render only the necessary parts of the UI when changes occur, resulting in improved performance.

### **3.1.2 Node.js**

Node.js is an open-source, cross-platform runtime environment that allows developers to run JavaScript code outside of a web browser. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient for building scalable network applications. Several node packages are used in our system to reach project objectives.

### **3.1.3 Express.js**

Express.js, commonly referred to as Express, is a minimalistic and flexible web application framework for Node.js. It is designed to provide a simple and intuitive way to build web applications and APIs. Express.js is built on top of the core HTTP module of Node.js, adding a layer of abstraction and additional features to simplify web development.

### **3.1.4 MongoDB**

MongoDB is a popular open-source document-oriented database management system (DBMS) that falls under the category of NoSQL databases. It was first released in 2009 and has gained significant popularity due to its flexibility, scalability, and ease of use.

Key features and concepts of MongoDB include:

- Document-Oriented Data Model
- NoSQL and Non-Relational
- Scalability and High Availability
- Querying and Indexing
- Flexible Data Model
- Integration and Ecosystem

### **3.1.5 Google Maps Api**

To achieve our project objectives we are using several google maps apis. google maps apis will give us the option to locate stores based on user search, we will also use google maps apis to show the exact location of physical stores in our system.

**Maps JavaScript API:** This API enables us to embed interactive maps into our web applications. It provides features such as map display, zooming, panning, adding markers, customizing map styles, and handling user interactions.

**Geocoding API:** The Geocoding API allows developers to convert addresses into geographic coordinates (latitude and longitude) and vice versa. It helps in geolocation, address validation, and various location-based services.

**Places API:** The Places API allows developers to integrate place-related information into their applications. It provides access to details about millions of places, including businesses, landmarks, points of interest, and more. Developers can search for places, retrieve details, and display them on maps.

**Directions API:** This API enables developers to calculate and display directions between different locations. It provides routes, step-by-step instructions, estimated travel times, and distance calculations based on various transportation modes like driving, walking, or public transit.

**Distance Matrix API:** The Distance Matrix API allows developers to calculate distances and travel times between multiple origins and destinations. It helps in determining the optimal routes, estimating delivery times, and providing distance-based services.

### ***3.2 Api Interfaces***

We deployed our backend to aws ec2 instance and our database (mongodb) on other ec2 instances. Connection between backend server and db was done with ec2 security bound rules, we made sure that our back end server only got access to our db instance. Our front end communicates with the back end server using AWS API gateway.



Below are the details of our api's.

BASE-API-Url:

<https://gceh50045i.execute-api.eu-north-1.amazonaws.com/api>

### 3.2.1 Signup

Table 3.2.1 Signup Api

Module	Login/Registration
Method	POST
URL	{baseurl}/auth/signup
Headers	Content-Type: application/json,
SampleBody	<pre>{   "firstName": "test",   "lastName": "Nippon",   "email": "john.doe@example.com",   "username": "testNippon",   "password": "password123",   "repeatPassword": "password123",   "isSubscribe": true }</pre>
Sample O/P	

### 3.2.2 LOGIN

Table 3.2.2 Login Api

Module	Login/Registration
Method	POST
URL	{baseurl}/auth/login
Headers	Content-Type: application/json,
SampleBody	{ "email": "john.doe@example.com", "password": "password123" }
Sample O/P	

### 3.2.3 Get User Profile

Table 3.2.3 User Profile Api

Module	Profile
Method	GET
URL	{baseurl}/secured/user-profile
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParam s/Body	N/A

Sample O/P	<pre>{   "_id": "65e6f4b506148ad53b4bdb17",   "firstName": "test",   "lastName": "Nippon",   "email": "arshidnippontech@gmail.com",   "username": "testNippon",   "isSubscribe": true,   "favourites": [],   "__v": 0,   "profileImageUrl":   "https://jangsimages.s3.eu-north-1.amazonaws.com/1709636856037-Screenshot%202024-02-28%20at%2015.54.18.png" }</pre>
------------	---

### 3.2.4 Upload Profile image

Table 3.2.4 User Profile Image Upload Api

Module	Profile
Method	POST
URL	{baseurl}/secured/profileImage
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParam s/Body	Form-data {profileImage: file}
Sample O/P	<pre>{   "message": "Profile image uploaded successfully.", }</pre>

	<pre> "profileImageUrl": "https://jangsimages.s3.eu-north-1.amazonaws.com/1709636856037- Screenshot%202024-02-28%20at%2015.54.18.png" } </pre>
--	--

### 3.2.5 Get Products

Table 3.2.5 Get Products Api

Module	Filter
Method	GET
URL	{baseurl}/products/getProducts?gender=Men&subCategory=Topwear&baseColour=Navy Blue&usage=Casual&minPrice=90&maxPrice=99
Headers	Content-Type: application/json,
QueryParams/Body	N/A
Sample O/P	<pre> {   "page": 1,   "limit": 10,   "results": [     {       "_id": "65e6f8b35d29c883707ce0ad",       "id": "7964",       "gender": "Men",       "masterCategory": "Apparel",       "subCategory": "Topwear",       "articleType": "Tshirts",       "baseColour": "Black", </pre>

```
"year": 2011,  
"usage": "Sports",  
  "productDisplayName": "Puma Men's Large Logo Graphic  
Black T-shirt",  
  "price": 91,  
  "store_id": "6",  
  "store_size": "M",  
  "season": "Fall",  
  "SKU": "50",  
  "imageUrls": [],  
  "__v": 0,  
  "storeDetails": {  
    "_id": "65e6fc25f1fccceb204c61c2",  
    "StoreName": "Marimekko Turku",  
    "store_id": "6",  
    "Latitude": "60.4509008",  
    "Longitude": "22.1127748",  
    "__v": 0  
  }  
}  
]  
}
```

### 3.2.6 Get Filter Options

Table 3.2.6 Get Filter Options Api

Module	Filter
Method	GET
URL	{baseurl}/products/product-attributes?gender=Men&subCategory=Top wear
Headers	Content-Type: application/json,
QueryParams/Body	N/A
Sample O/P	<pre>{   "baseColours": {     "isAvailable": true,     "data": [       "Green",       "Grey Melange",       "Cream",       "Pink",       "Maroon",       "Off White",       "Burgundy",       "White",       "Blue",       "Rust",       "Olive",       "Brown",       "Purple",       "Orange",       "Navy Blue",       "Mustard",       "Yellow",     ]   } }</pre>

	<pre>         "Beige",         "Grey",         "Black",         "Lavender",         "Red",         "Charcoal",         "Multi",         "Peach"       ]     },     "usages": {       "isAvailable": true,       "data": [         "Formal",         "Ethnic",         "Sports",         "Casual"       ]     },     "prices": {       "isAvailable": true,       "data": [         49,         74,         78       ]     },     "storeSizes": {       "isAvailable": true,       "data": [         "L",         "XS",         "M", </pre>
--	--

	<pre>         "S",         "XXL",         "XL"       ]     },     "seasons": {       "isAvailable": true,       "data": [         "Spring",         "Winter",         "Fall",         "Summer"       ]     }   } }</pre>
--	--

### 3.2.7 Make Favorite

Table 3.2.7 Make Favorite Api

Module	Favorite
Method	PUT
URL	{baseurl}/secured/favourite/:ProductID
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParam s/Body	NA
Sample O/P	<pre> {   "message": "", }</pre>



### 3.2.8 Get favorites

Table 3.2.8 Get favorites Api

Module	Favorite
Method	GET
URL	{baseurl}/secured/favourites
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParams/Body	NA
Sample O/P	{ }

### 3.2.9 Remove Favorite

Table 3.2.9 Remove favorite Api

Module	Favorite
Method	DELETE
URL	{baseurl}/secured/favourite/:productID
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParams/Body	
Sample O/P	{ "message": "Product Successfully removed from favorite", }

### 3.2.10 Make Reservations

Table 3.2.10 Make Reservations Api

Module	Reservations
Method	POST
URL	{baseurl}/secured/reserve/: ProductID
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParams/Body	Form-data
Sample O/P	{ "message": "Product Successfully Reserved", }

### 3.2.11 Get Reservations

Table 3.2.11 Get Reservations Api

Module	Reservations
Method	GET
URL	{baseurl}/secured/reservations
Headers	Content-Type: application/json, Authorization: Bearer {{user-token}}
QueryParams/Body	N/A
Sample O/P	{ }

## 4.0 References

1. <https://expressjs.com/en/guide/routing.html>
2. <https://nodejs.org/docs/latest/api/>
3. <https://www.mongodb.com/developer/>
4. <https://aws.amazon.com/ec2/>
5. <https://aws.amazon.com/api-gateway/>
6. <https://aws.amazon.com/s3/>