in collaboration with

Softwarica College of IT & E-commerce | Coventry University


# Megaplex

*Real-World Project ~ S21*

**By: Chirag Simkhada | Nischal Bade | Ashok Shrestha | Salil Timalsina | Bimal Shrestha**

**To: Giri Raj Rawat**

## Abstract:

Most Nepali E-commerce is dominated by a primary marketplace where a customer has to succumb to the standards set by a distributor/mediator. Moreover, the platforms charge a considerable amount as commission for posting advertisements on their platforms. This, in turn, reduces the profit earned by the seller. To combat this, a neutral e-commerce platform is required which prioritizes people before profit. This is where Megaplex comes into play. To set the context, Megaplex is a secondary marketplace i.e., a customer-to-customer e-commerce platform.

The following document looks at the needs and the process of developing this platform. Initially, the aims and objectives of developing this application are stated along with the mission and vision of this application. Afterward, the scope of Megaplex is discussed which is followed by a careful stakeholder analysis. To improve the usability of this application, the literature review is done on similar applications which made us see the room for improvement in that application. This was followed by additional research on our platform which helped us to avoid the flaws of said applications. On an interesting note, this application explores the ethical issues surrounding E-commerce platforms and user data. Similarly, Risk analysis was done to understand riskthe s associated with this concept. PESTEL analysis is done to understand macro-environmental factors associated with this platform. Finally, the design along with the workflow of the system is included in this report.

# Table of Contents

# Table of Figures

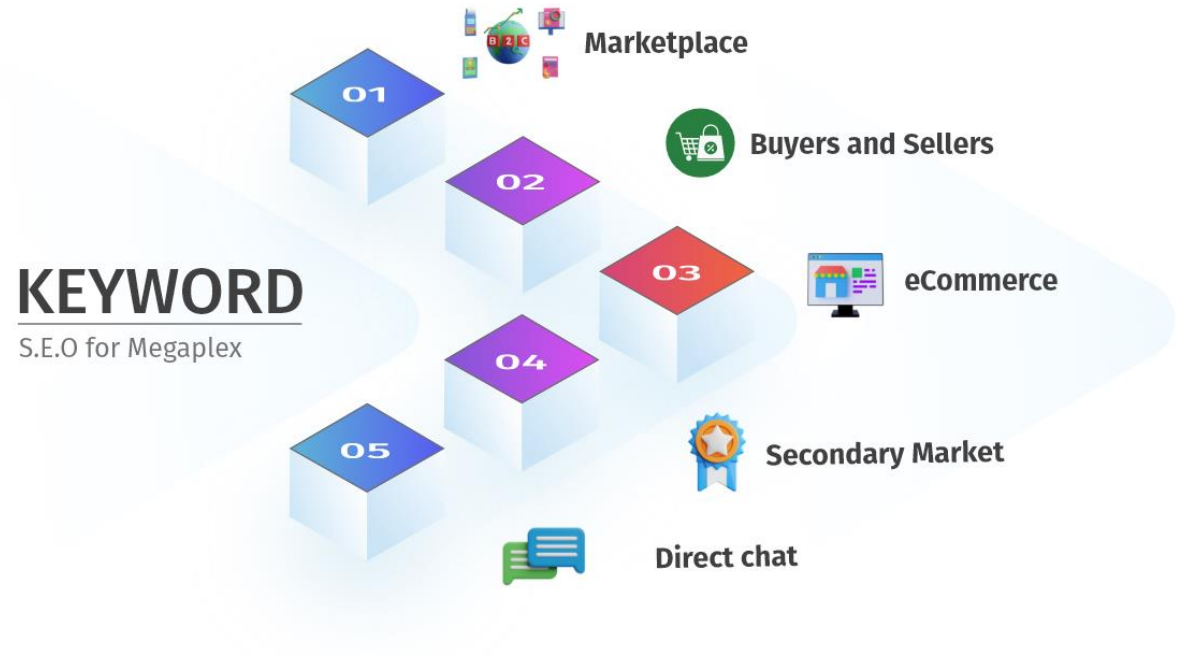



*Figure 1: Keywords*

# Introduction

Megaplex is a C2C (Customer to Customer) second-hand marketplace where customers can easily sell unused, old, or new products. The platform solves the problem of going to a store and buying things at ridiculous prices and encourages customers to purchase things at affordable prices. Nowadays selling different categories of products of individual and organization sellers have so many problems and faced confusion was to sell their products. Similarly, buyers also have problems with where to buy the old, unused, or new products. To point out these problems for both sellers and buyers megaplex website is one of the problem-solving online shopping websites for users focused on C2C. Megaplex provides the platform for sellers to display all their product categories with their rate, where the buyers can buy and place their orders for all kinds of the product by paying through a different e-banking system. Buyers can ask freely about the products through chat sessions between sellers and

## Aim

With the rising prices of products everywhere, customers want nothing more than to save money. This is where Megaplex comes in. The platform's primary aim is to remove the burden of visiting several stores and encourage the reuse of products at reasonable prices and connect buyers with potential sellers.

## Objectives

- ✓ Solve the current problem of Second-hand product buying and organizational in-stock product listing.
- ✓ Develop and establish an online channel and community between buyers and sellers
- ✓ Encouragement to Recycle products.
- ✓ Show the relevant product according to their need and interest.

## Mission and Vision


*Figure 2: Mission and Vision of Megaplex*

# Mind Maps

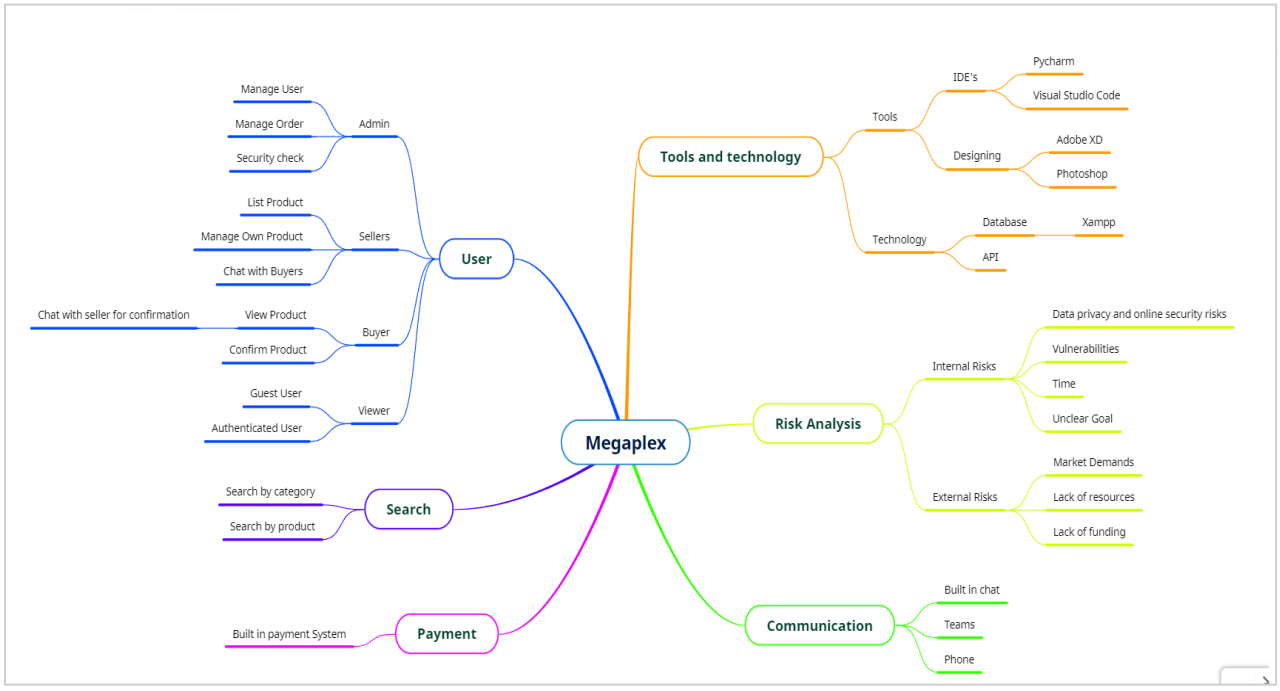



*Figure 3: Mind map of the project*

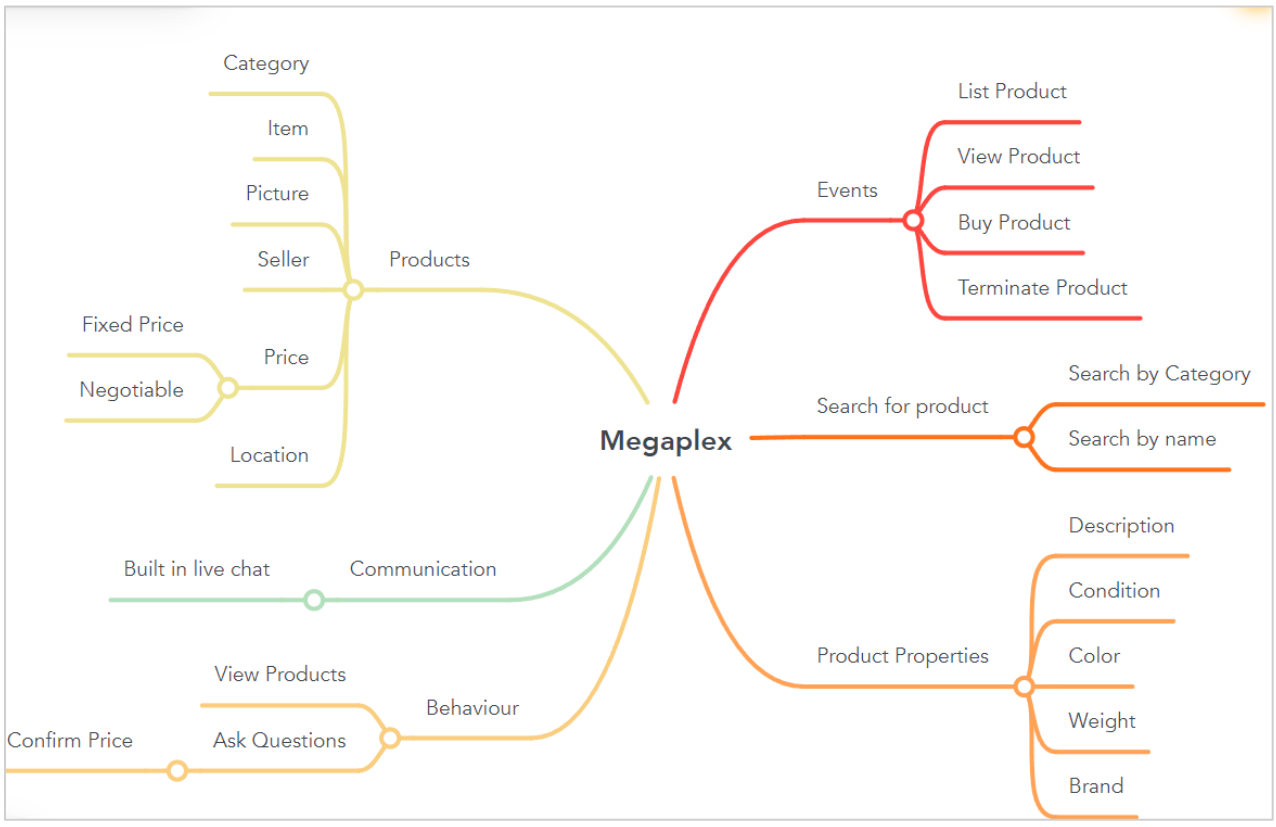
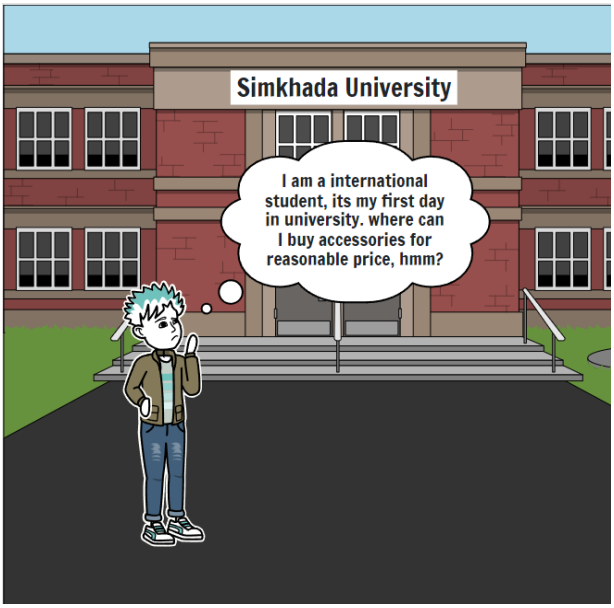



*Figure 4: Mind map of Megaplex*

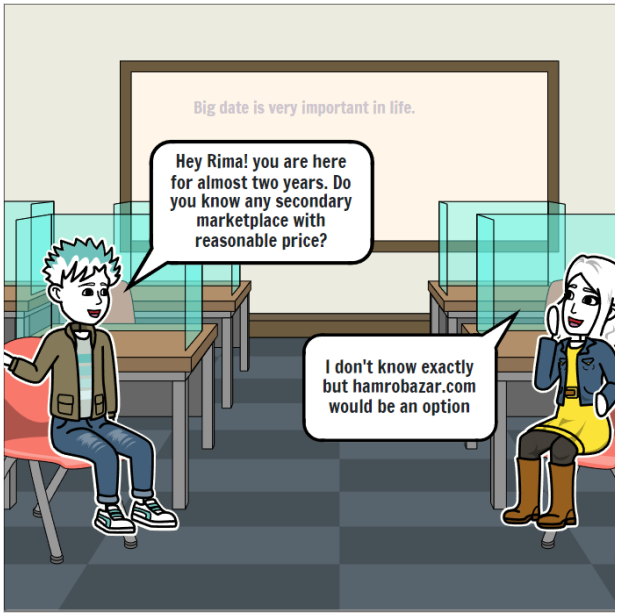
## Problem Definition

The smallest and largest online businesses that operate on a business-to-consumer (B2C), consumer-to-consumer (C2C), and buyer face these types of problems in their daily life. They face so much confusion about where to sell them and from where to buy their products like automobiles, real state,electronicsc, musicalinstrumentst, furnitu,re and many more products which we used in our daily life. Similarly, users have so many stock products, but users have no idea to sell them, and users need to buy a new product, but users have a limited budget so, users are searching for a nearly new product to buy.

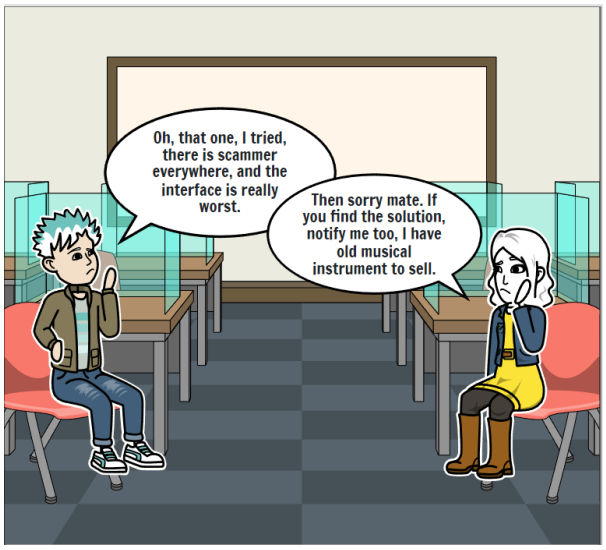
**Case ONE**




Salil is new in Nepal as an international student enrolled in Simkhada University. He wants to buy living items at a reasonable price than costly items.



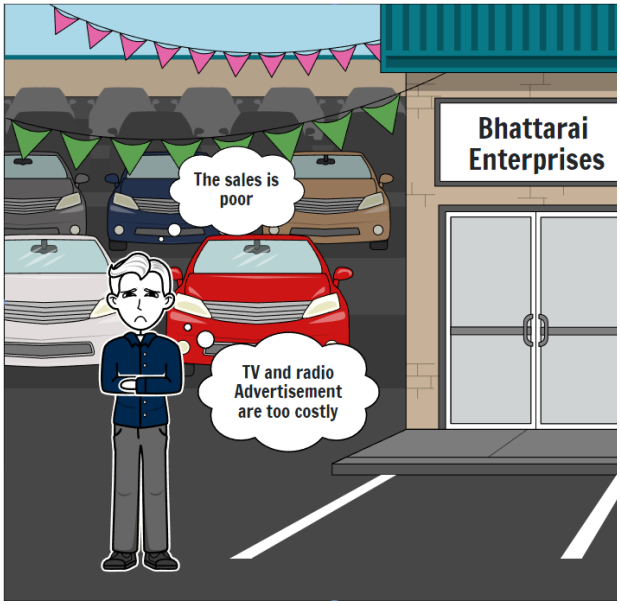

He asked for help from the senior batch girl, Rima. She recommended a local C2C market to Salil.



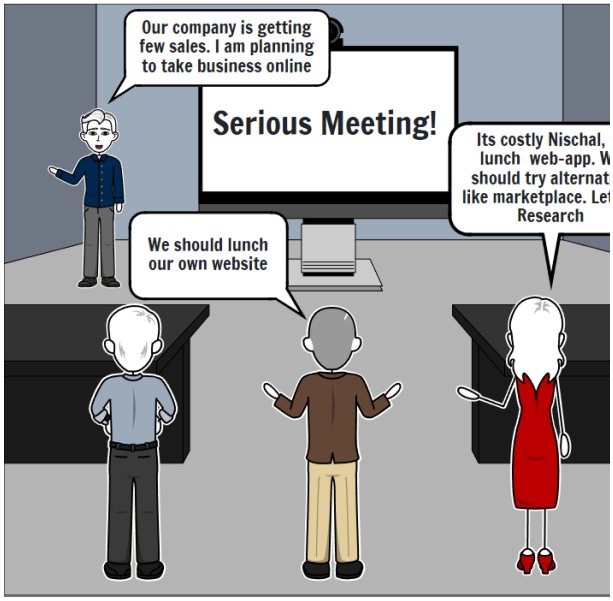

Salil already tried it and found it untrustworthy.


He then searched around a local store but was unable to find a quality good

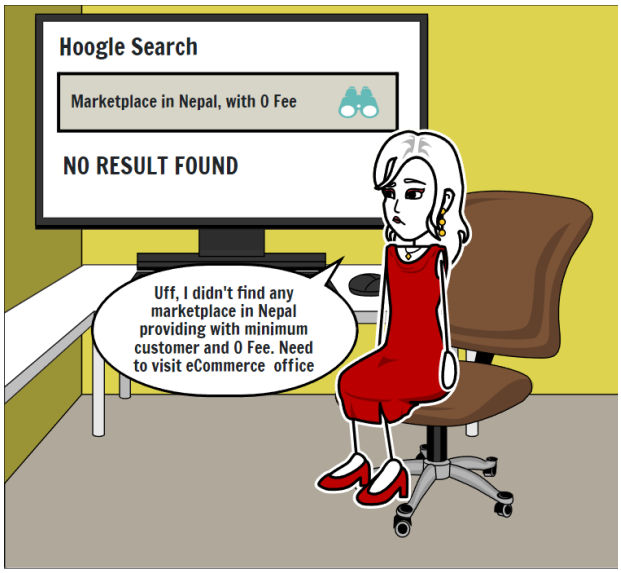
**Case TWO**



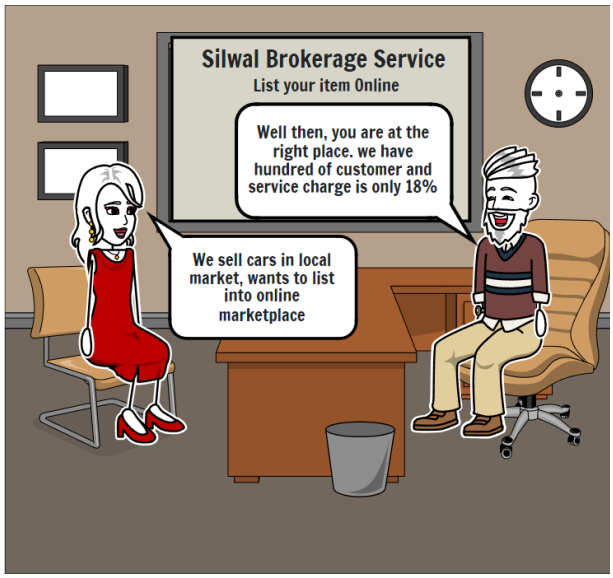

Nabin owns a car company and is thinking about boosting the sales




He organized the meeting, and got numerous suggestions.




Lalita researched online but found no result. So she decided to visit some office.

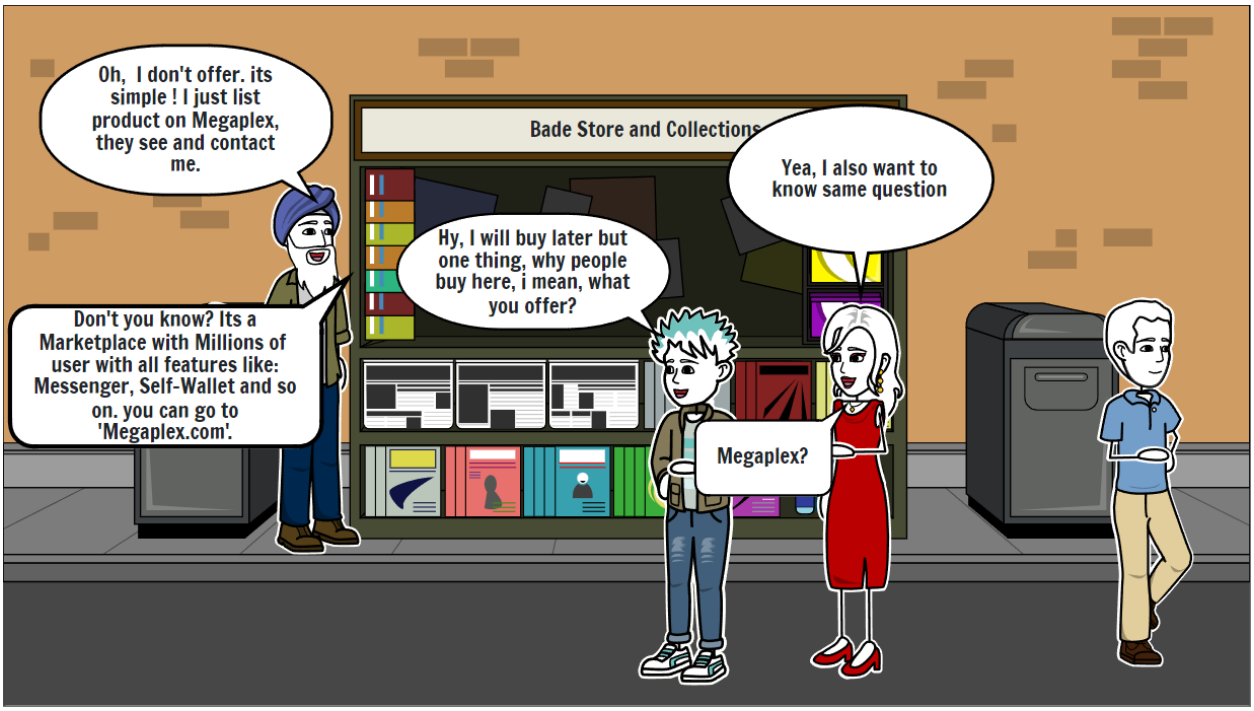



She visited a brokerage service but she doesn't like the offer

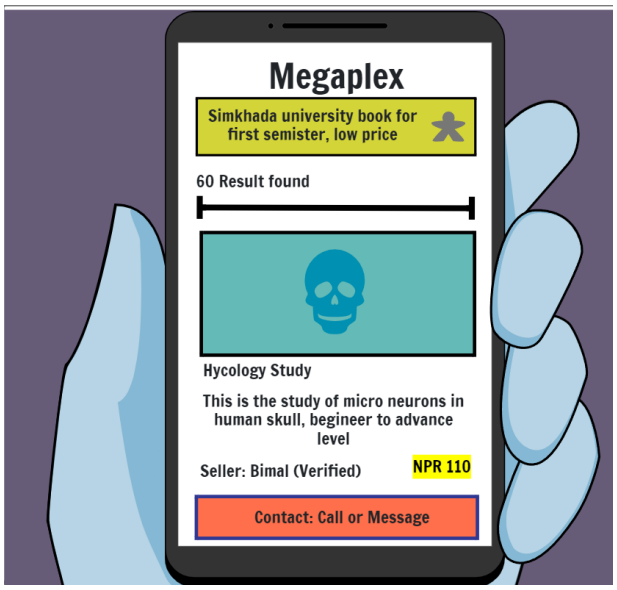
## Solution

Megaplex is one of the biggest problem-solving marketplaces. In essence, a Megaplex allows a company to function almost entirely online, removing the requirement for a physical location with sales and administrative staff. The process is automated, which distinguishes it from other websites that still place a premium on commercial development above customer satisfaction. Users can buy or sell their products through the megaplex easily. The seller has access to display their products by fixing their prtheargin  of products in this system.

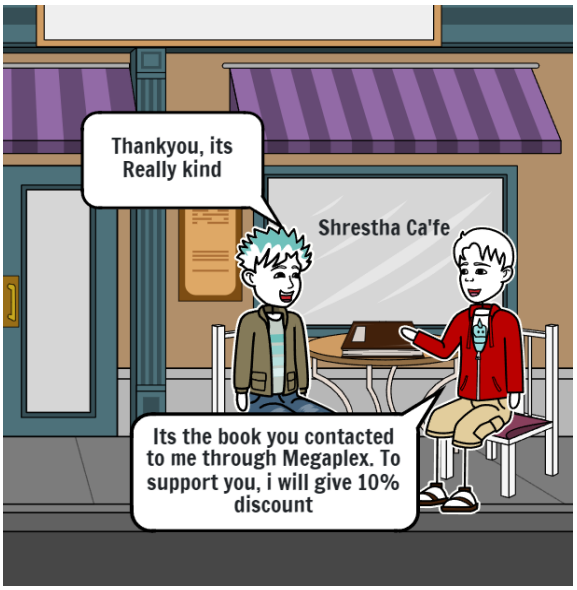

They both reached the shop to buy basic items. Salil is shocked why all are here to buy and Lalita is thinking about how he managed to get lots of buyers.



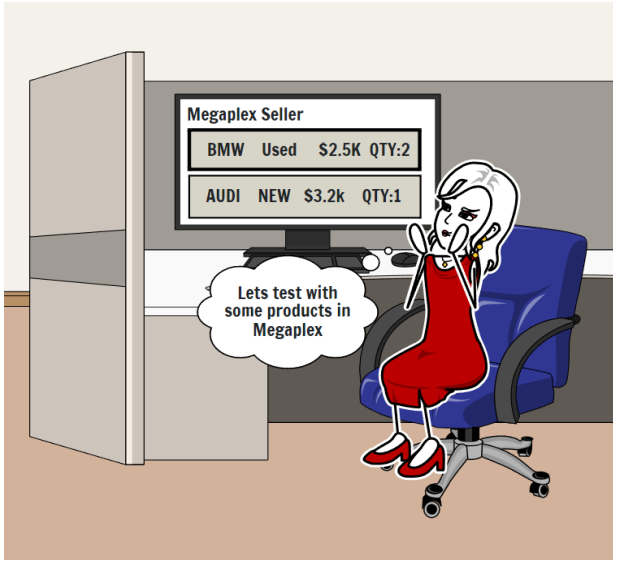

They asked the seller and found all the information about **'Megaplex'.**




Salil then searched for Megaplex and found the price of the book is half that of a brand new one from a Verified seller: Bimal.

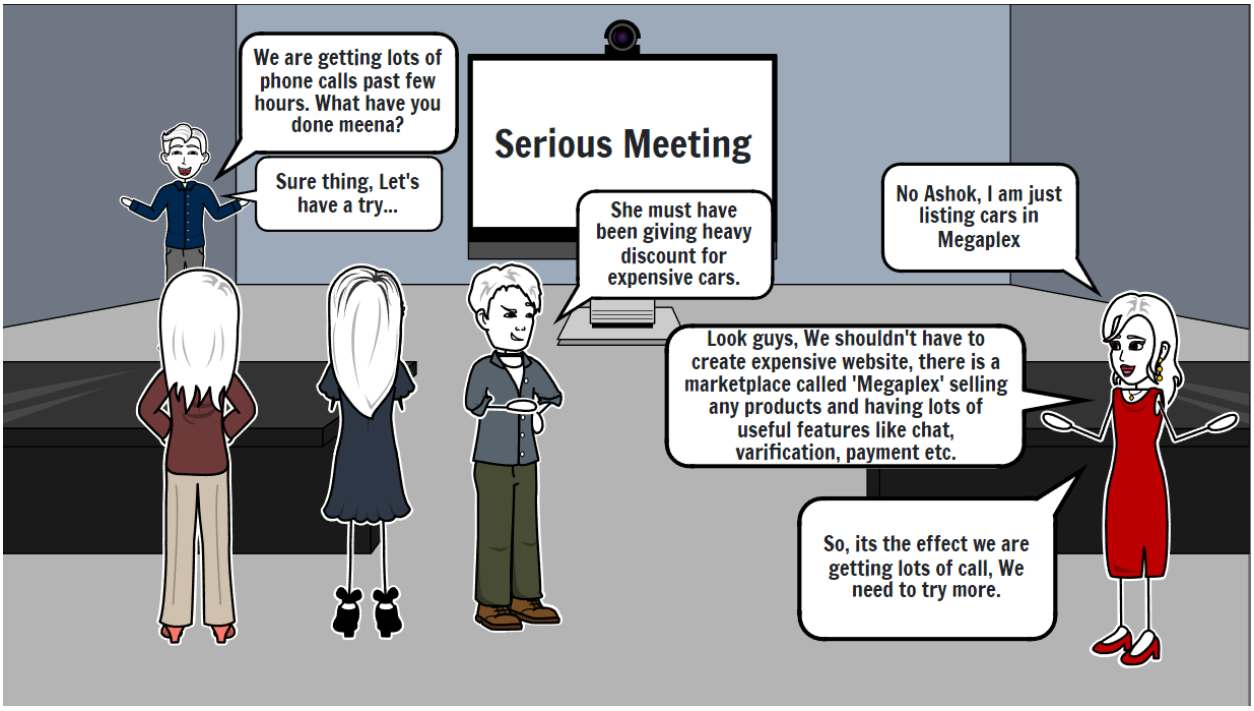



He met the seller and bought the book. Likewise, he gets an instant 10% discount. The problem gets solved.




She got into the office and list some cars with quantity


After that, she got a message from one of the users surfing BMW. She got happy.




In a meeting, she provides detailed information about everything, and the manager gives it a go for improving sales performance.

***

# Scope of Megaplex

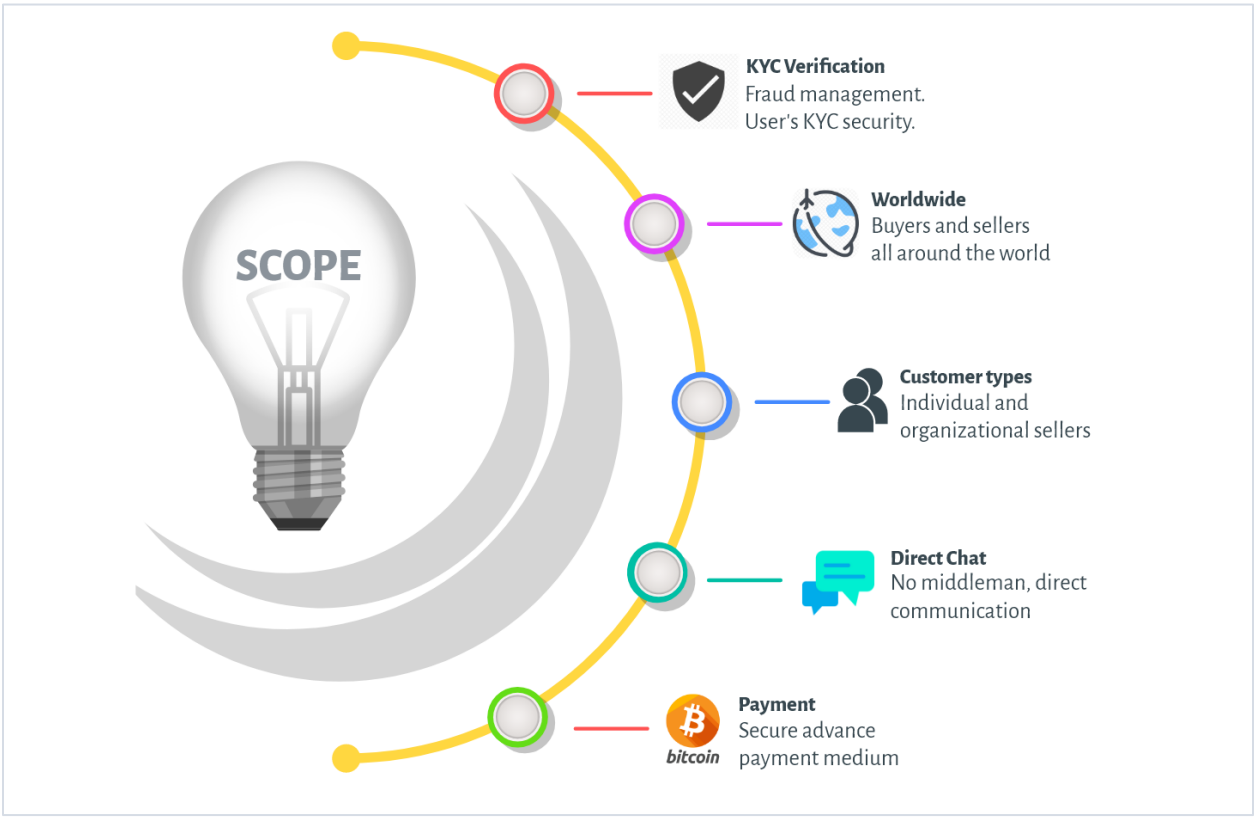



*Figure 5: Scope of Megaplex*

## Stakeholder Analysis


*Figure 6: Stakeholders Analysis of Megaplex*

Internal stakeholders are those who have a direct relationship with an organization, such as programmers, project team members or managers, or shareholders. User groups, government agencies, and the media are examples of external stakeholders, who are not employed by a firm but are influenced by its actions and consequences in some fashion.

**Other key stakeholders include:**

1. Domain and hosting provider
2. Competitors
3. Advertising Agencies

## Business Model Canvas


*Figure 7: Business Model Canvas*

# Ethical Consideration

Ethics is a collection of ideals and norms that should be maintained while dealing with human issues. No one works in a way that is harmful to the community or an organisation for ethical reasons. Ethical implications are crucial, particularly in research.

*Figure 8: Ethical Concerns*

Megaplex website is built followed by ethical and, legalization issues that are in human nature. While using this website users face the permission of privacy, ethical and legal conditions before entering the user's registration process. There is no chance licked user's information and cannot be accessed by any third-party software. User information is stored in a website server full of safety and privacy maintenance. Users have only access to the limited resources which they used on this website.

# Literature Review:

## HamroBazar

Formerly owned by Saakha Group, Hamrobazar is a Nepali customer-to-customer trading system that allows people and businesses to post a wide range of new and used items. (Aryal 2021) In its rise, Hamrobazar offered a unique solution to e-waste recycling and personalized trading. Despite conducting trades worth millions of rupees on daily basis, Hamrobazar is solely dependent on third-party advertisements to earn revenues. This system of revenue resulted in a loss to the Sakha Group which compelled them to sell the company to its highest bidder: MNS Investment. As of April 2021, the system is handled by the MNS group with the same objective of providing a free advertisement hosting platform for Nepali individuals and organizations.

The workflow of the system is simple: let users post advertisements, let others view them, and let them set terms of agreements for trading. This is one of the simplest ways of running a marketplace. This system shows no involvement of the application itself and enables customers to contact their counterparts and arrange a deal. However, with no intervention of the system such as the trader verification, the buyers may find themselves at odd when

distinguishing a genuine seller from a scammer. Moreover, with just the comment section and contact number, the users have limited options for contacting the seller. This limited functionality may drive users towards other trading platforms and the site stakeholders may see a decrease in their revenue.

On the flip side, Megaplex offers a distinct solution to these problems. With the real-time chat system, the users will be able to contact the seller directly and justify their queries. Moreover, with the user verification in place, the system administrator can offer a "verified" badge to genuine sellers and offer users more transparency.

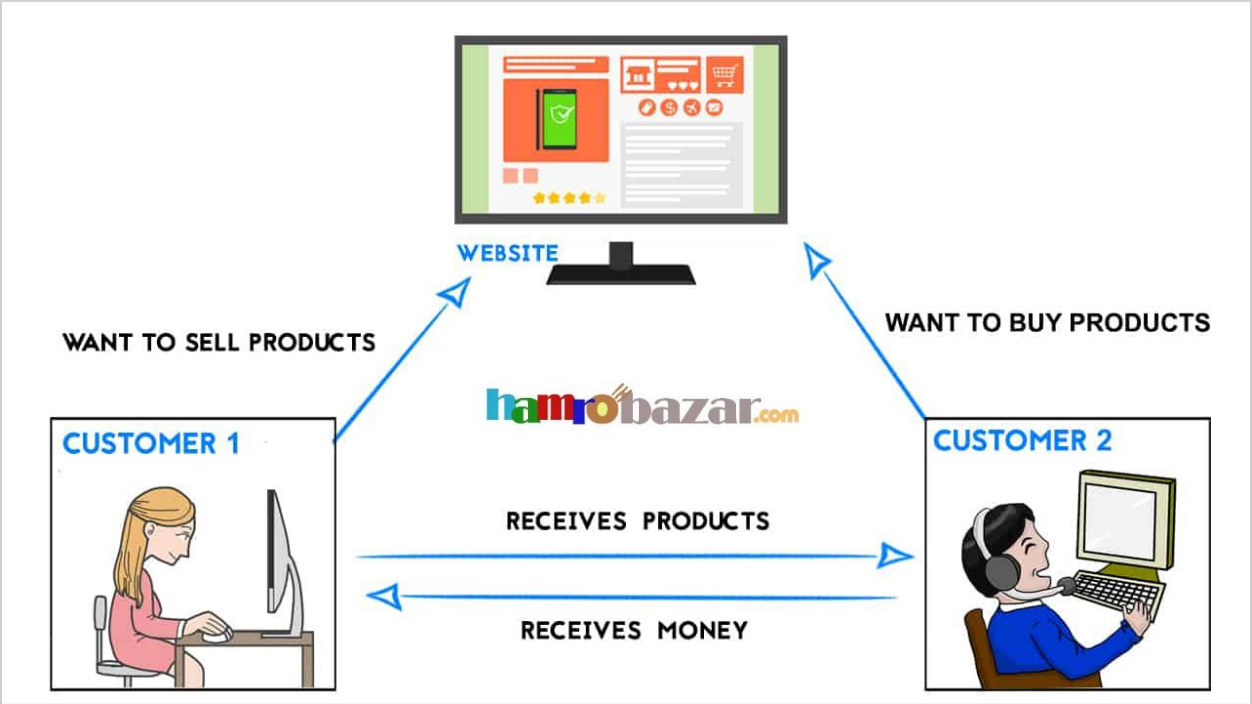



*Figure 9: Hamrobazar*

## eBay:

Based in San Jose California, eBay is an American e-commerce trading platform. With millions of daily users, eBay's mission statement reads, **"pioneer new communities around the world built on commerce, sustained by trust, and inspired by opportunity"**. With its advanced features such as social listings, carts, in-depth search, product reviews, and Wishlist, eBay has been successful at retaining a large number of loyal customers. (Hsiao 2019) It is interesting to notice that this platform is not limited to a single country and its customers span across the globe.

However, even with its considerable resources, eBay has failed to update its interface with the changing time. The UI of the system loses its appeal with the passing day. The button in

this system still resembles its earliest ancestor and the failure to update them is somewhat unsatisfying.

On the other hand, Megaplex is equipped with the latest UI and offers the smoothest user experience to its customers. Similarly, the room feature allows the user to connect with other users or system administrators if they wish to do so. This offers more flexibility to the users.


*Figure 10: eBay*

## Sastoramro:

With headquarters located in TankPradad roundabout, Kathmandu, Sastoramro is a Nepali customer-to-customer trading platform. Having aims and objectives the same as Hamrobazar, Sastoramro offers another advertisement hosting platform to Nepalese. In comparison with Hamrobazar, Sastoramro has considerably lower user activities and revenues. The site offers a basic advertisement hosting platform with a limited number of advertisements and a seriously outdated UI. The method of generating revenue is similar to Hamrobazar which is to rely on third-party advertisements.

Looking at the limitations of this system, it is composed of drawbacks as that of Hamrobazar. Moreover, it even lacks a comment section where user can post their product views. Also,

the lack of advanced filters can be considered an additional drawback of this system. However, its User Interface can be seen as the biggest imperfection among all of its defects.

In contrast, Megaplex offers users a well-maintained comment section to post their queries or reviews along with well-detailed seller information. To combat the bad UI of Sastoramro, Megaplex offers the latest UI designs and visually appealing animations.


*Figure 11: Sastoramro*

## Quikr:

Primarily designed for Indians, Quikr is an Indian advertisement posting platform that can host advertisements from simple household items to job posting from multinational companies. (Anon 2022) With its aim of "empowering every person in the country to independently connect with buyers and sellers online", Quick offers more features than the above-mentioned systems.

On a closer analysis, Quikr fails to limit the number of advertisements on its platform which provides a negative experience to users the buyers/sellers. Moreover, the lack of a comment section in the said platform has provided additional drawbacks to this system. Without a comment section, users are unable to express their views on the application resulting in a lack of transparency in the system.

On the flip side, Megaplex is equipped with proper tools to enable users to express their views on the listed products. Moreover, the advertisements are placed in such a subtle way that it does not distract users from performing their tasks.

## Coutloot:

Describing itself as “India's largest offline to online social commerce app”, Coutloot is another customer-to-customer trading application designed for Indian customers. On its official website, It has stated that Coutloot has been successful in selling more than 60 Million products through its platforms. With its proper logistic management, live package tracing, and lively customer interactions, It has seen considerable customer growth in recent years.

Considering their interface, the work done on their platform is plausible. However, for a company with such a strong customer base, they have failed to maintained customer satisfaction. On closer analysis, the lack of a website-based platform may be one of the reasons. Although this system is performing quite well as a mobile application, It still lacks a website presence. Moreover, even in its app version, most customers are complaining that they are having difficulties in finding categories (Raut 2019).

In Contrast, Megaplex has a website presence that can be accessed from any device and be used for trading. All the steps are designed after a careful usability analysis of this application. Hence, it is unlikely that the users will face the same usability issue such as locating categories page as that of Coutloot.


## Primary Research:

In order to fix the problems with the aforementioned items, the proposed system must contain an effective communication system to facilitate the divulgence of views between the seller and the buyer. Moreover, the users in the case-study products were having difficulty navigating the system. Hence, it is only logical for our system to have an effective workflow to compensate for those design flaws. In the same topic, the users must be able to express their views freely on the respective platforms. Hence, a well-maintained comment section is vital for the marketplace to provide transparency to the users. Finally, most of the products listed above have an outdated Ui which may cause the users to switch to other platforms. Therefore, our proposed system must be equipped with the latest UI to stand out from the crowd.

## Methodology

Megaplex uses a waterfall model for its website's software development life cycle (SDLC). The waterfall model aids in the step-by-step pre-design of the product, allowing this SDLC to be employed for the megaplex development process.

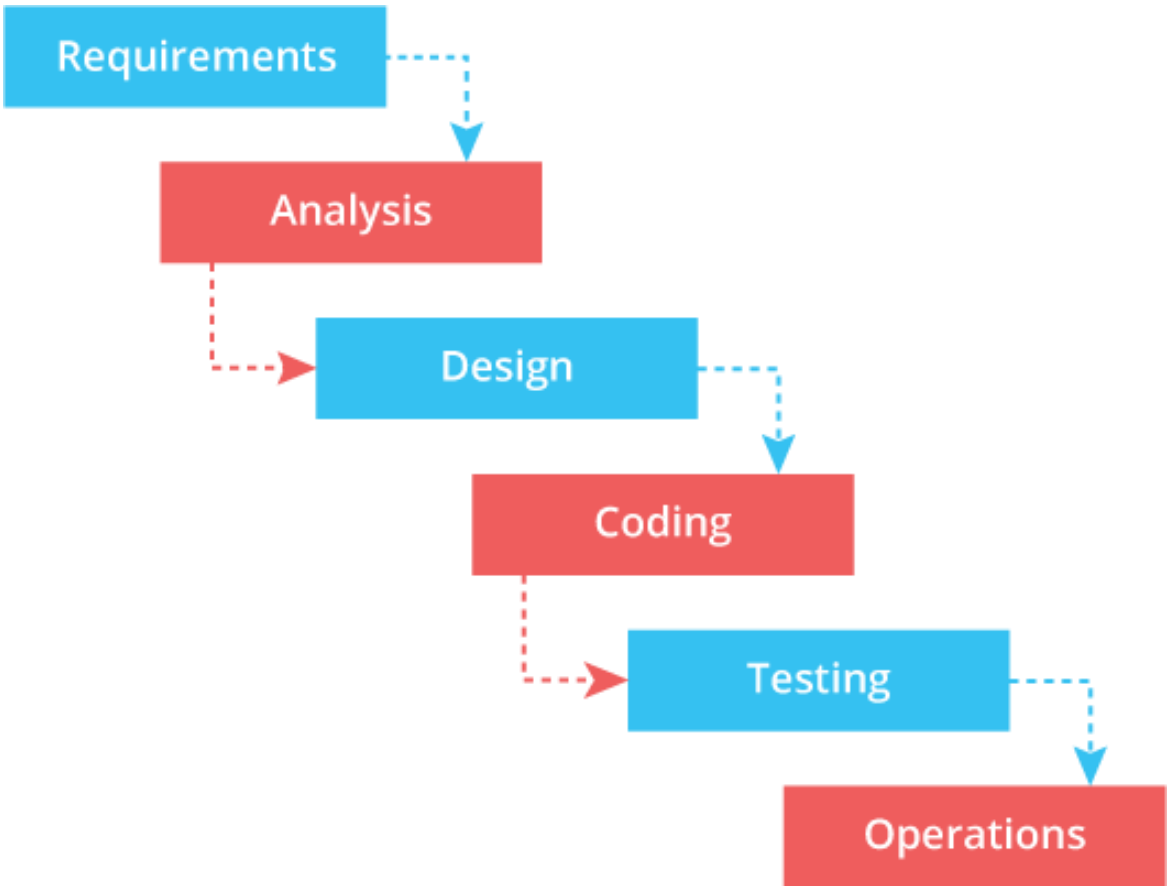



*Figure 12: Waterfall model*

**Requirement analysis:** To begin the megaplex website building process, our team members examine the challenges that people confront in their everyday lives, and we also debate the issues and how to fix them. We planned to establish a megaplex secondary marketplace website as a result of the discussion's outcome, which may assist individuals in solving challenges they confront in their daily lives. Our team members discussed the outcomes while conveying the user's answer during this procedure. Individuals need to be able to sell and buy all types of things that they use in their everyday lives. For the project's flexibility, all needs are analysed by all team members.

**Design:** Following the analysis and requirements evaluation phase, the prototyping phase begins the design process. All aspects of the megaplex website's design are based on the

developers' vision and system operations. We began learning about programming languages and frameworks before developing the system. A high-fidelity prototype, a paper prototype, an activity diagram, a use case diagram, and an ER-Diagram are all drawn to distinguish the seller, the buyer, and the admin by pointing to a user-friendly product.

**Implementation:** We designed the megaplex's homepage to include a navbar that allows users to register/login and view/search all of the products shown on the homepage. Every week, the project commander assigns tasks to team members divided into smaller groups based on the person responsible for developing the front end, the person in charge of maintaining the back end, and the one who is in charge of creating the documentation

**Testing:** After the constructed megaplex system was connected and the complete system was tested for any errors, our team members began testing with users in the age range of 14 to 60 by demonstrating the system's functions. During the testing stage, compliance with the customer's standards is also tested.

**Maintenance:** If a fault is discovered by the end-user, the developer team is accountable for fixing, updating, or modifying the product to ensure its effectiveness.

## People Analysis

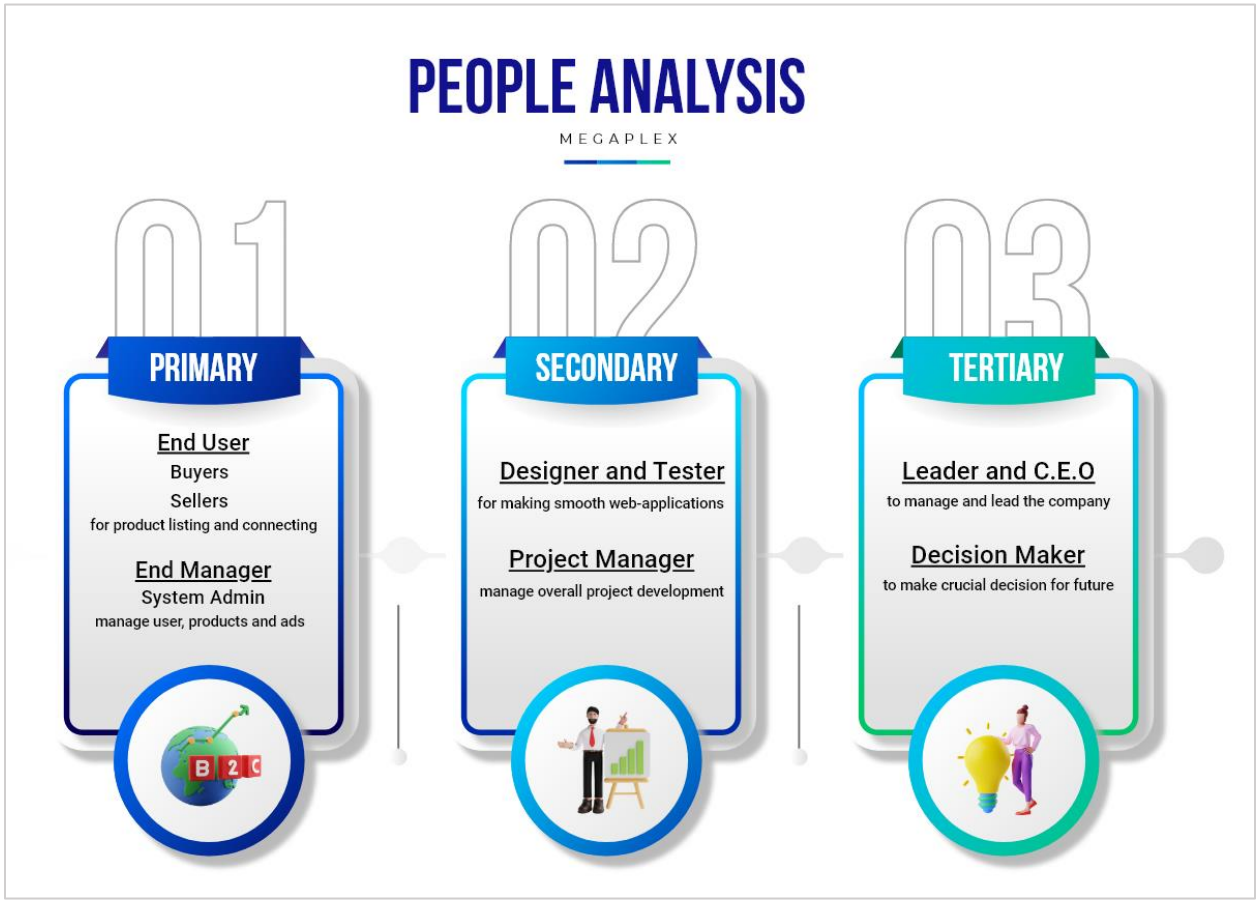



*Figure 13: People Analysis*

## Technology Used

**Systems for search, insights, finding**: Its recommendation is based on the user's ability to locate products and categories that are comparable to what they are looking for, as well as the user's eyesight.

**API:** A programming interaction is defined as an application programming interface (API). It was developed by the platform's inventor to enable other users to contribute to the development of aspects of the app's functionality without having to write any code. Developers offer endpoints, which are analogous to the application's inputs and outputs in terms of functionality. API keys would be used to restrict access to resources while working with an API. APIs developed by companies such as Facebook, Twitter, and Google for their online services are examples of good APIs. Tools used in Megaplex


*Figure 14: Tools Used*

# Use case Diagram

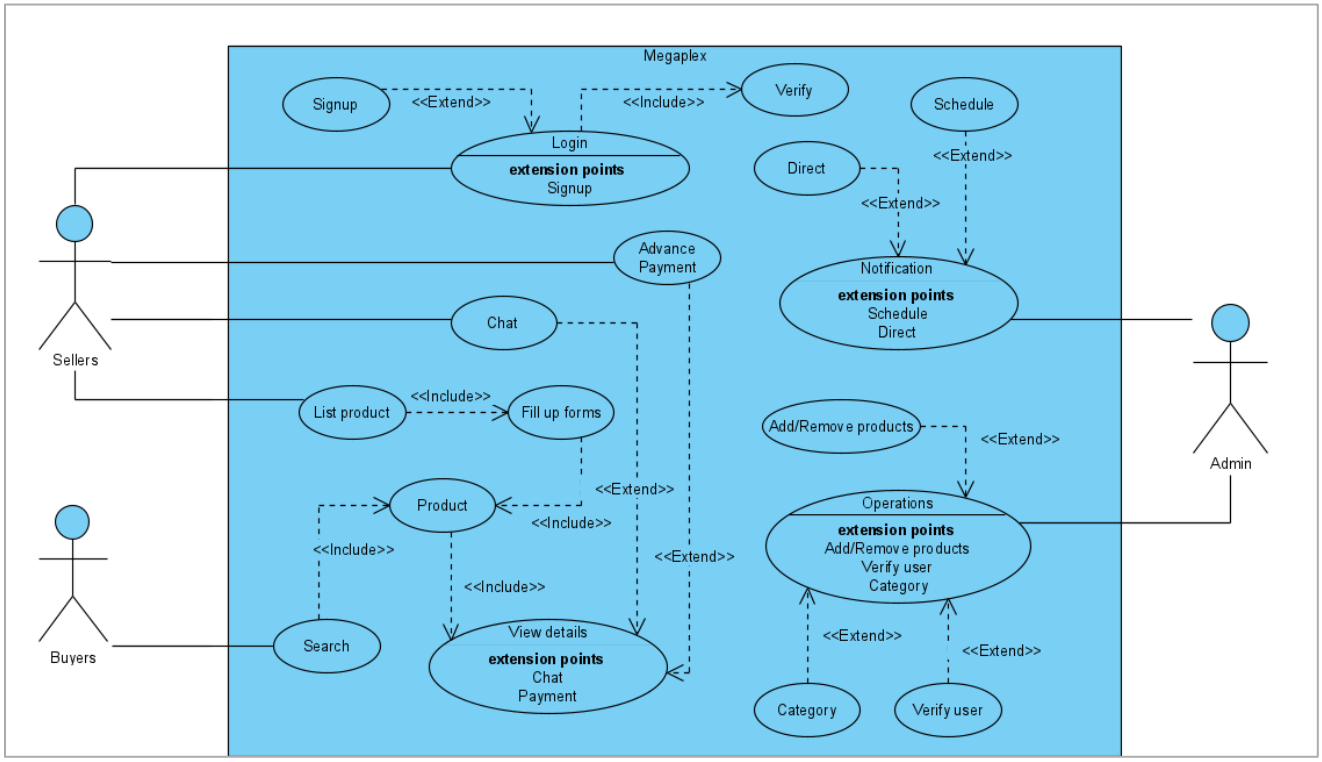



*Figure 15: Use Case Diagram of Megaplex*

# Activity Diagram

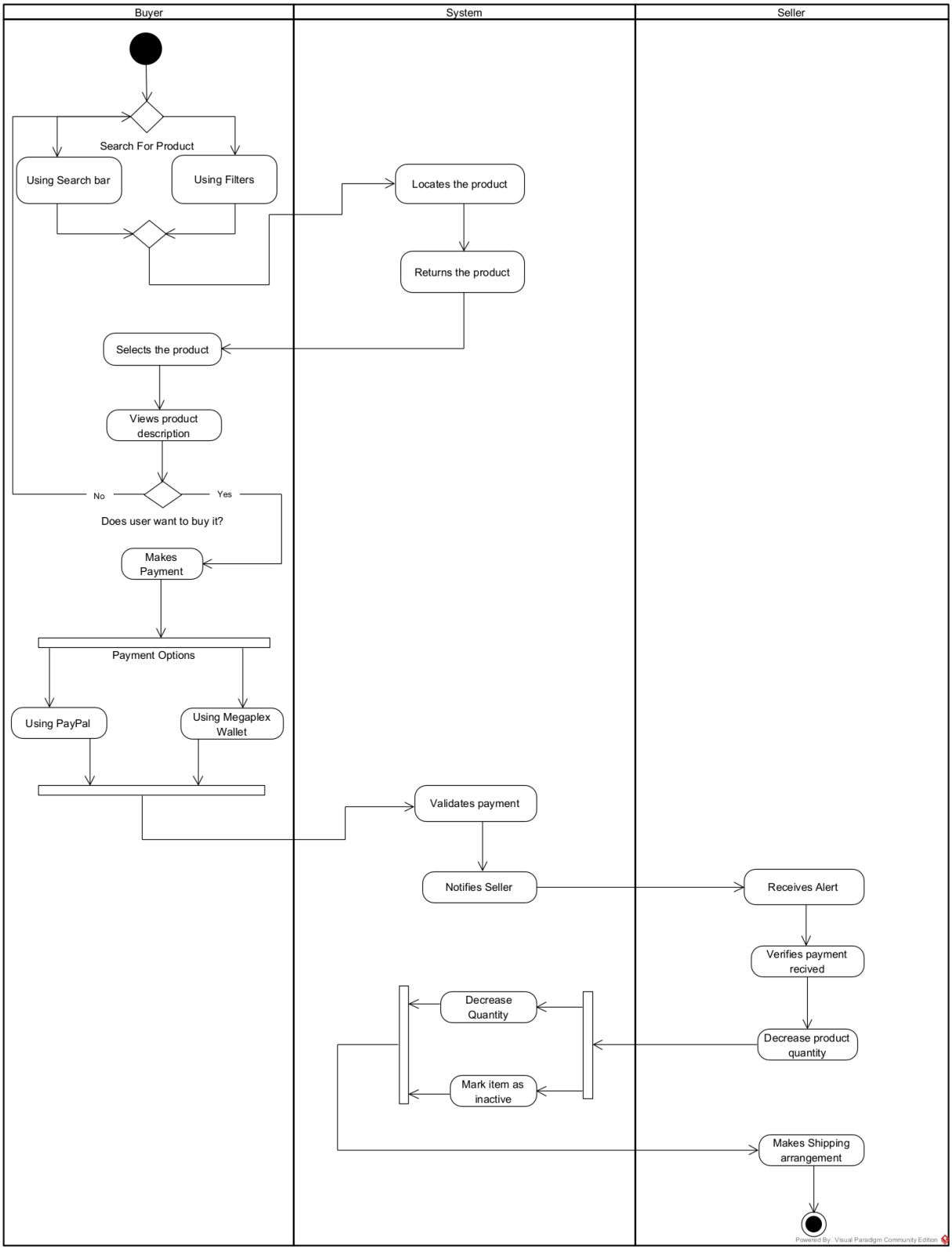



*Figure 16: Activity Diagram of Megaplex*

# Flowchart Diagram

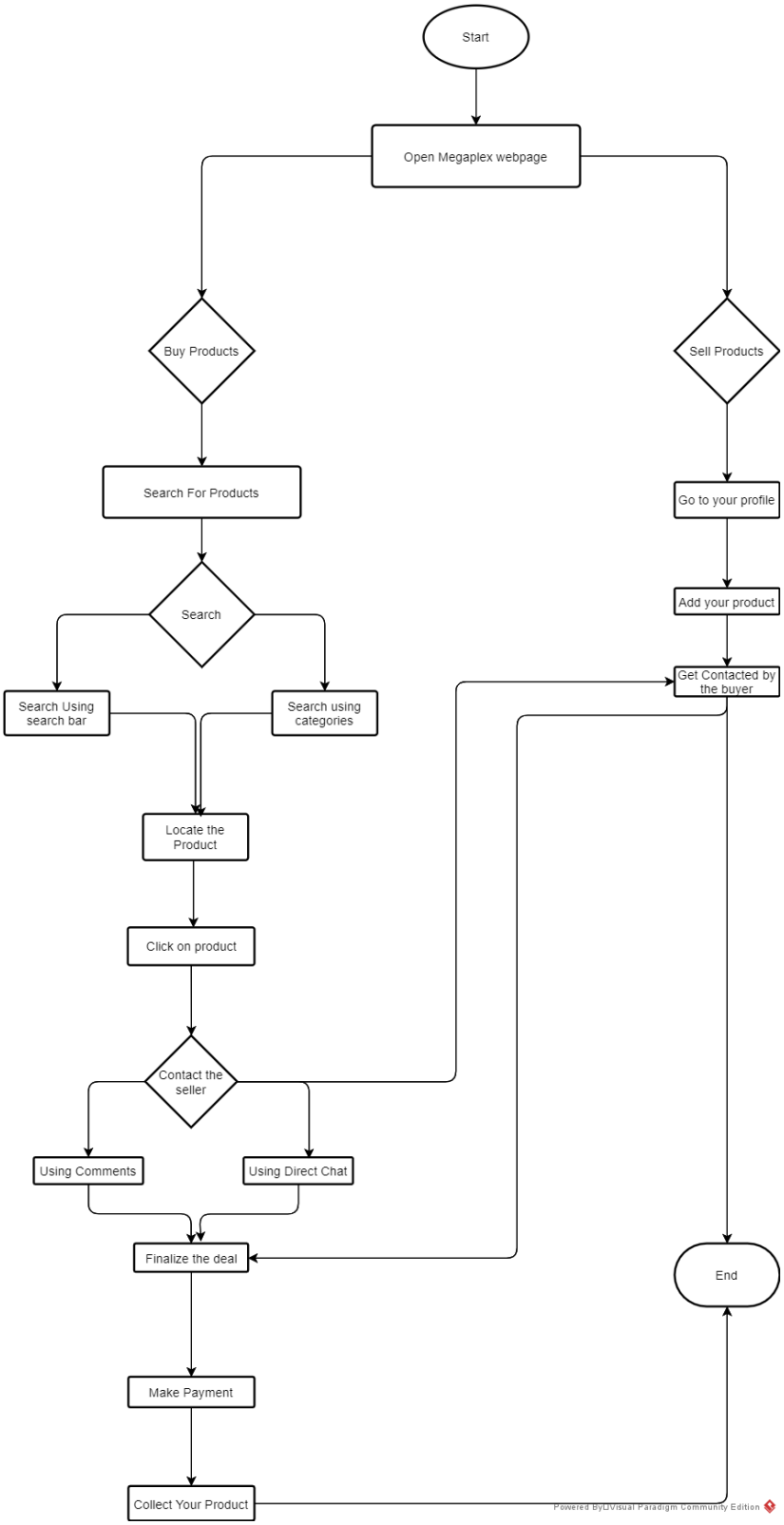



*Figure 17: Flowchart of Megaplex*

## PESTEL Analysis


*Figure 18: PESTEL Analysis of Megaplex*

**Political** factors have a big part in establishing what factors can affect Megaplex's long-term success in each region or area. Megaplex operates in Specialty Retail, Other in over a dozen countries and is exposed to various political environments and political system concerns.

**Environment:** Changing economic standards or environmental criteria, which might influence a company's profitability. (*PESTEL Analysis* n.d.). States might have multiple environmental and legal policies within a country. For example, various responsibility rules exist in Nepal in the event of disasters or environmental disasters, which might have a huge impact in Megaplex.

**Social:** In any setting, the culture of an organisation is affected by the culture of the society in which it operates. The general public's attitudes and beliefs have a big impact in a certain market and the way they craft their marketing messages has severe impacts.

**Technological:** As new technologies emerge, it is important for businesses to keep tabs on how quickly they disrupt existing markets. Faster pace means less time to recuperate and be effective, whereas slower pace means more time to recover and be successful.

**Economical:** Macroeconomic variables such as the rate of inflation, the mortgage rate, the rate of interest, the currency value, and the length of an economy's business cycle influence aggregate demand and aggregate investment in an economy. Microenvironmental factors such as competition norms have an influence on Megaplex's strategic advantage.

**Legal:** The legal and administrative structures in many countries are insufficient to secure a company's intellectual property rights. Megaplex must carefully examine whether or not to join such markets in order to maintain its competitive advantage.

# Updated Project Plan

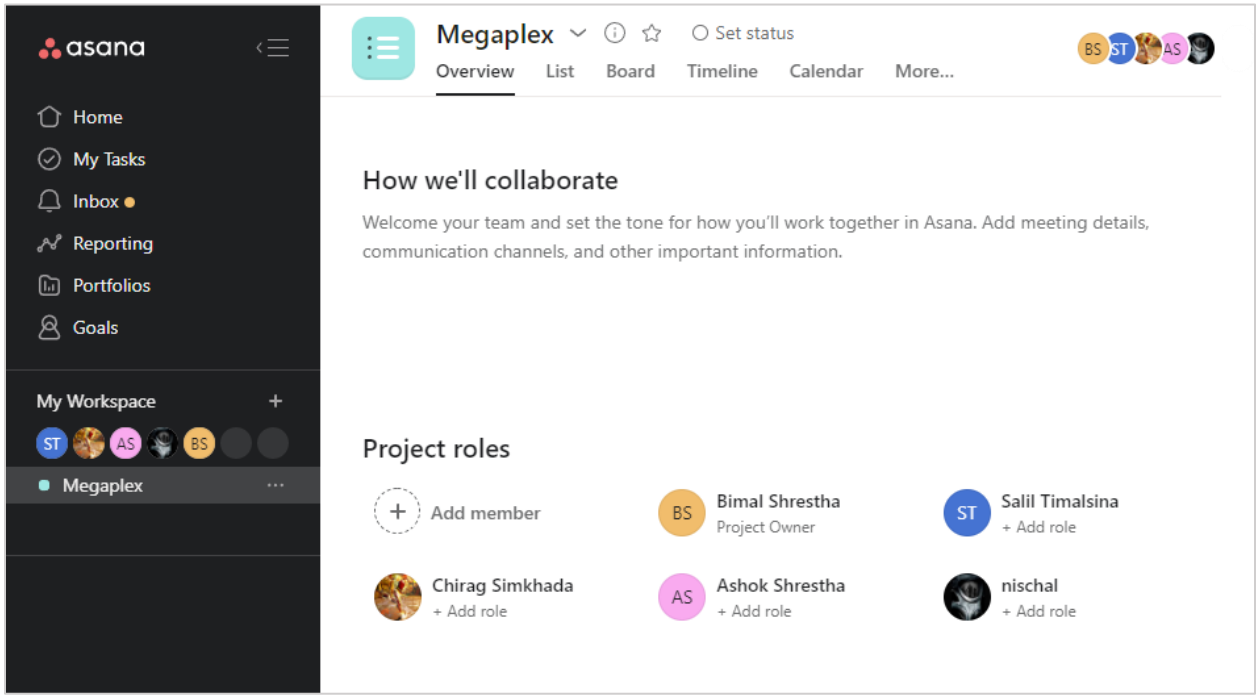



*Figure 19: Project Members*

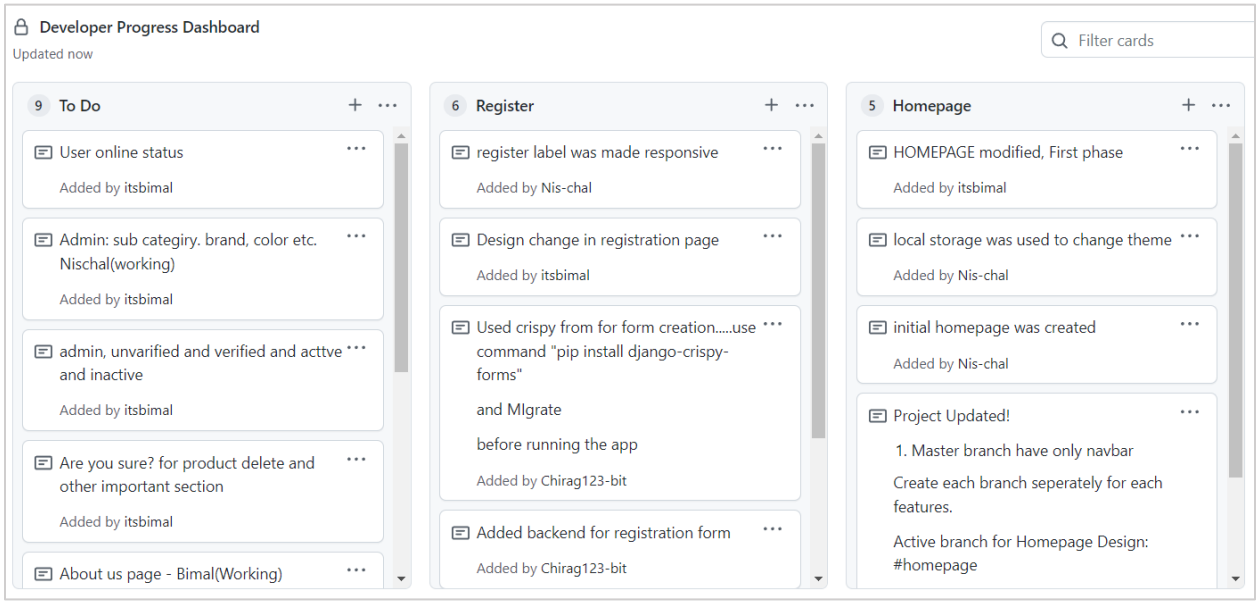



*Figure 20: Project Dashboard*

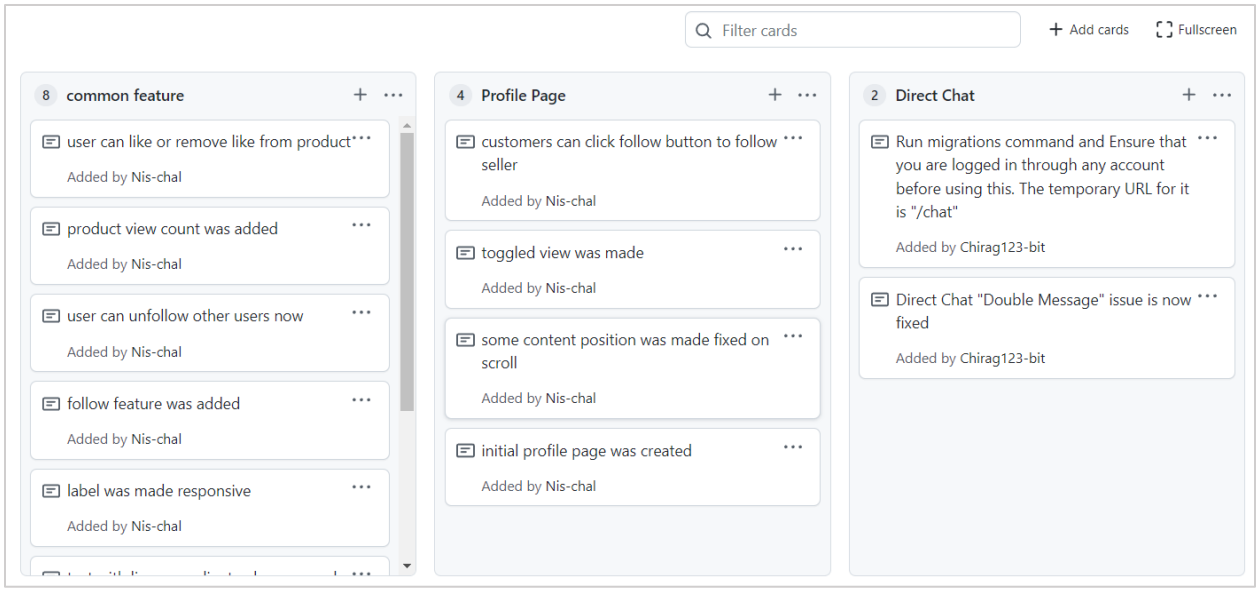
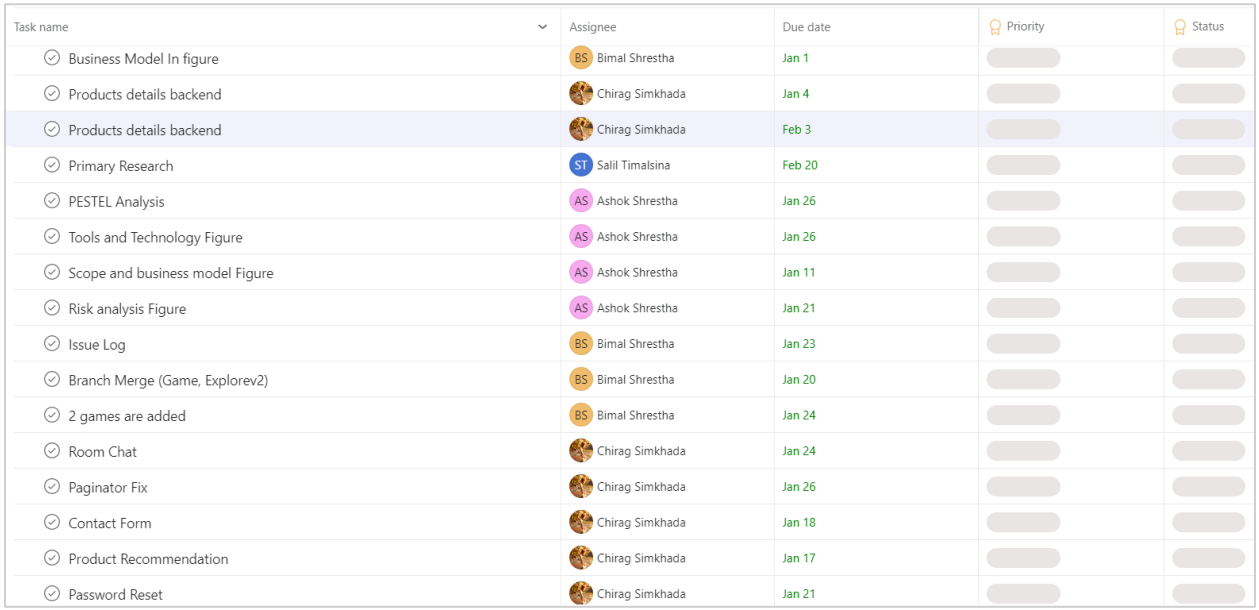



*Figure 21: Project Dashboard 2*



| Task name | Assignee | Due date | Priority | Status |
|---|---|---|---|---|
| Business Model In figure | Bimal Shrestha | Jan 1 | | |
| Products details backend | Chirag Simkhada | Jan 4 | | |
| Products details backend | Chirag Simkhada | Feb 3 | | |
| Primary Research | Salil Timalsina | Feb 20 | | |
| PESTEL Analysis | Ashok Shrestha | Jan 26 | | |
| Tools and Technology Figure | Ashok Shrestha | Jan 26 | | |
| Scope and business model Figure | Ashok Shrestha | Jan 11 | | |
| Risk analysis Figure | Ashok Shrestha | Jan 21 | | |
| Issue Log | Bimal Shrestha | Jan 23 | | |
| Branch Merge (Game, Explorev2) | Bimal Shrestha | Jan 20 | | |
| 2 games are added | Bimal Shrestha | Jan 24 | | |
| Room Chat | Chirag Simkhada | Jan 24 | | |
| Paginator Fix | Chirag Simkhada | Jan 26 | | |
| Contact Form | Chirag Simkhada | Jan 18 | | |
| Product Recommendation | Chirag Simkhada | Jan 17 | | |
| Password Reset | Chirag Simkhada | Jan 21 | | |

*Figure 22: Tasks*

# ejtyj Analysis

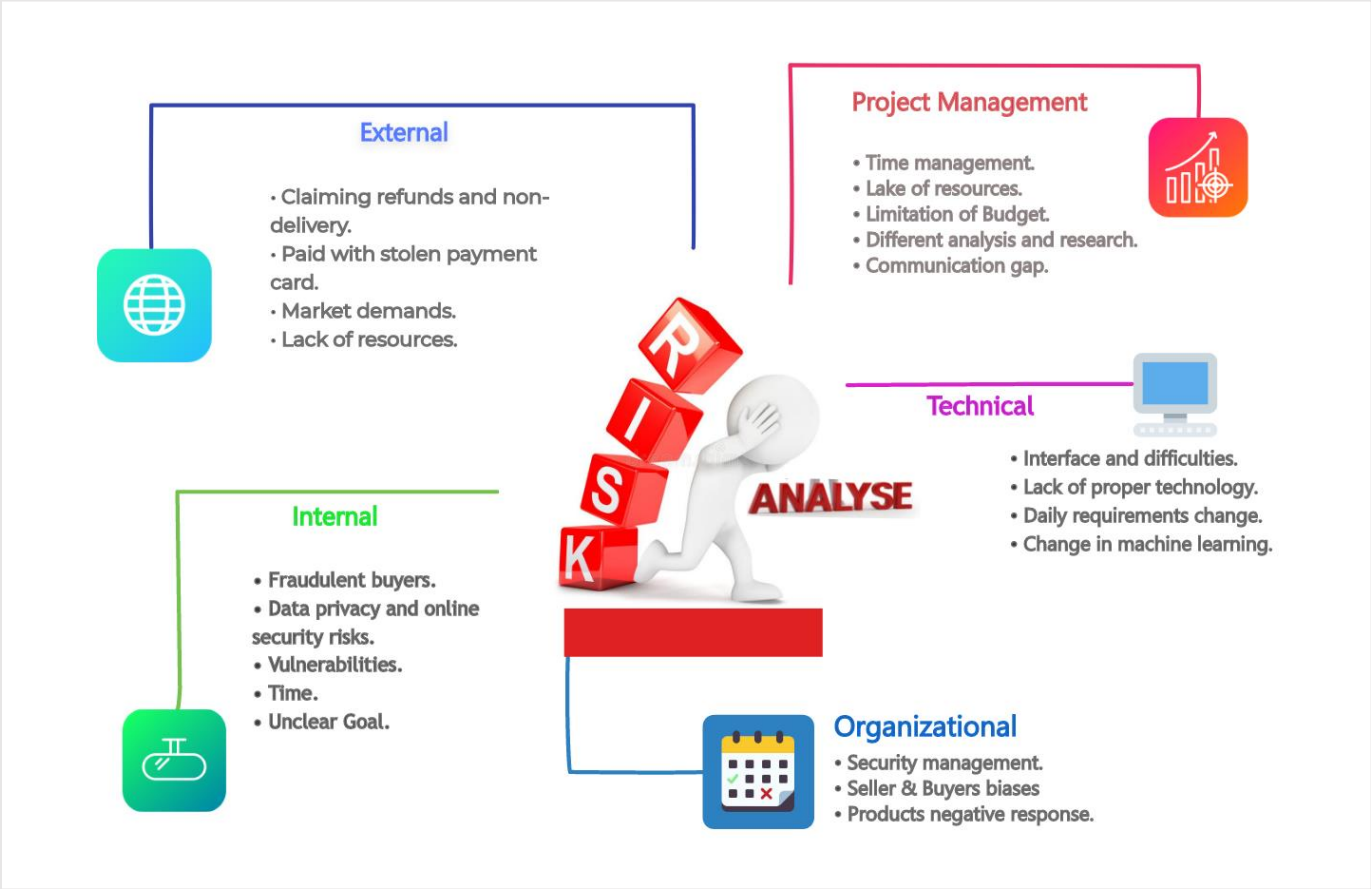



*Figure 23: Risk Analysis*

# Issue Log

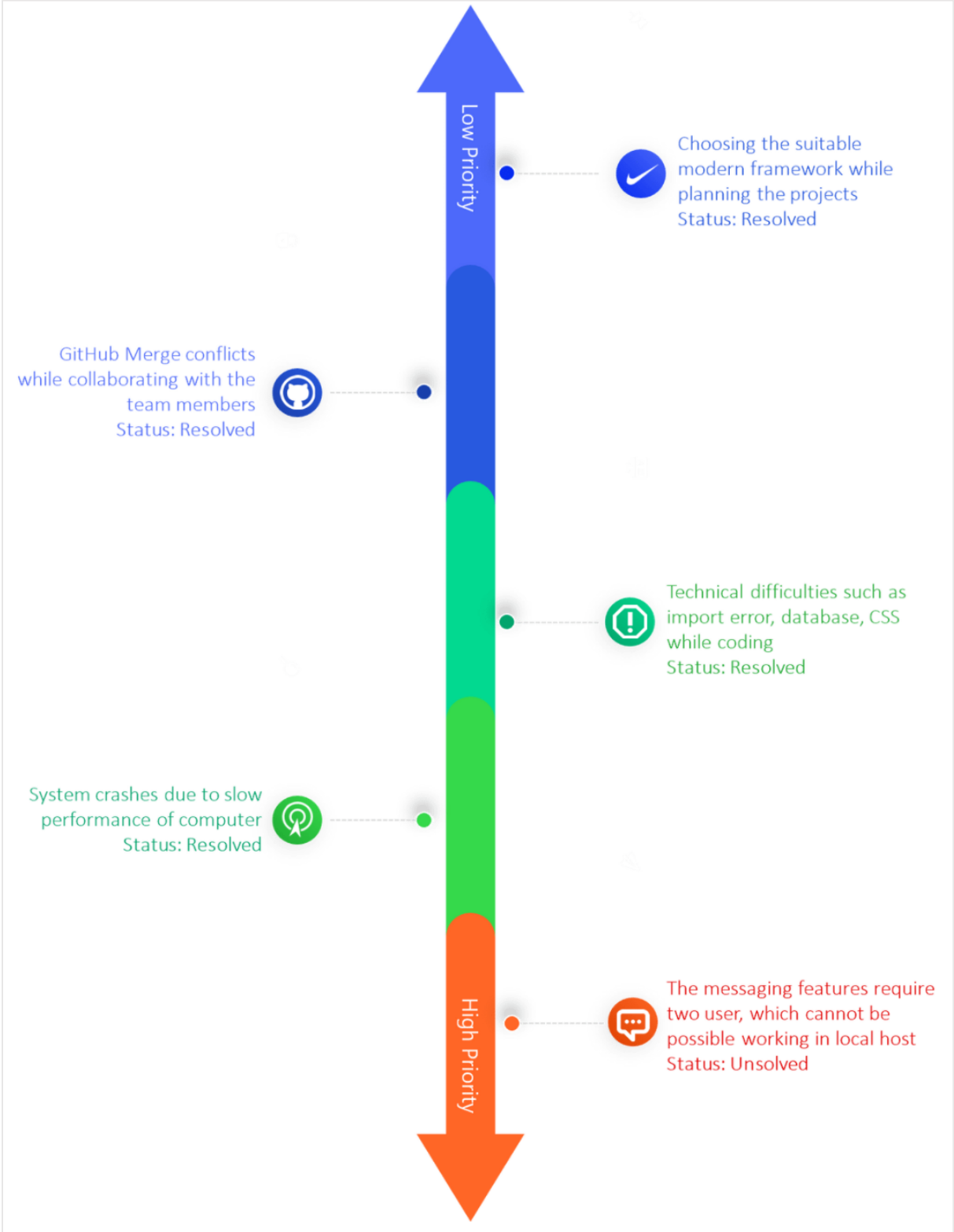



*Figure 24: Issue Logs*

# Workflow Diagram

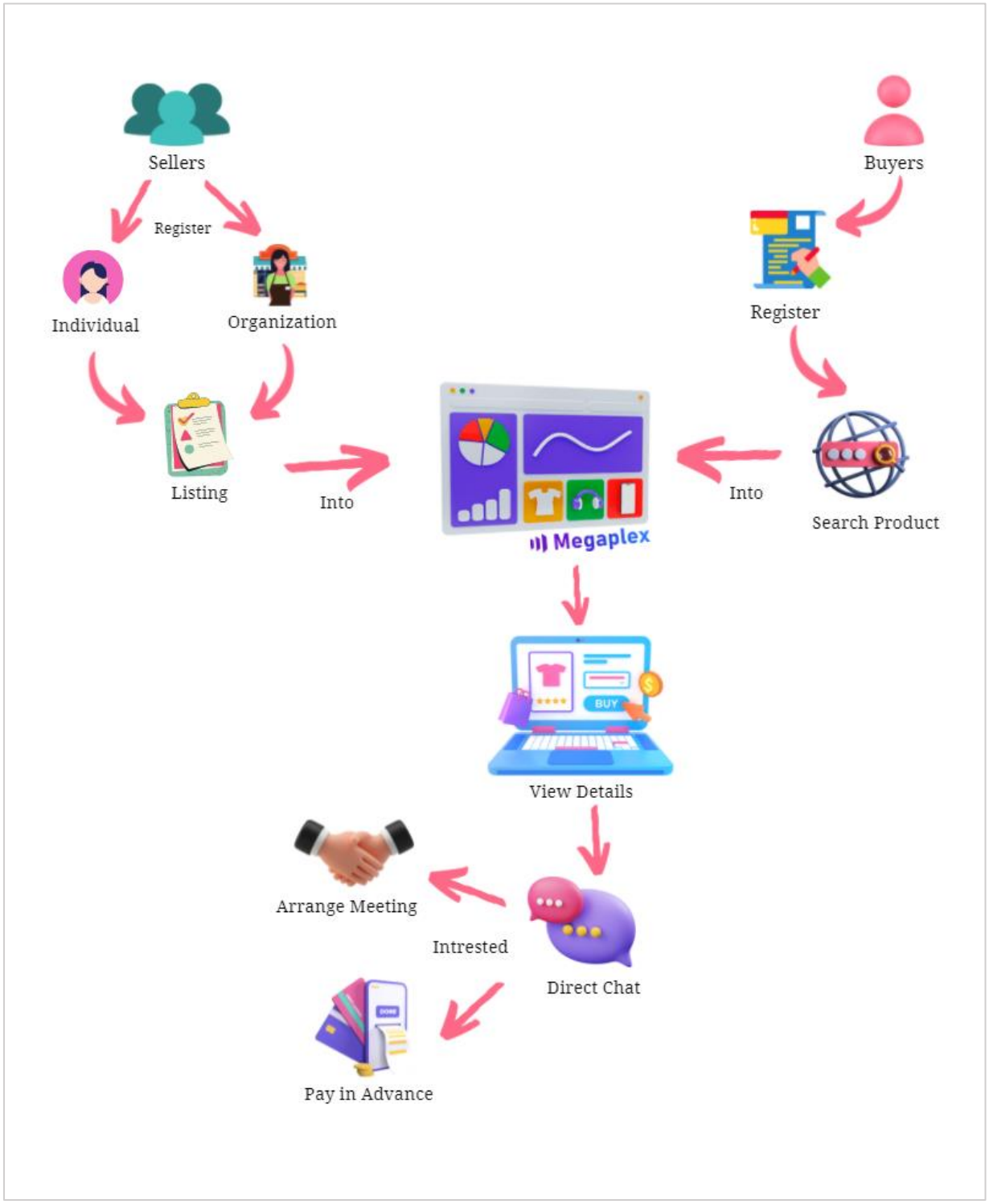



*Figure 25: Workflow of Megaplex*

# Megaplex Initial Prototype (Adobe XD)

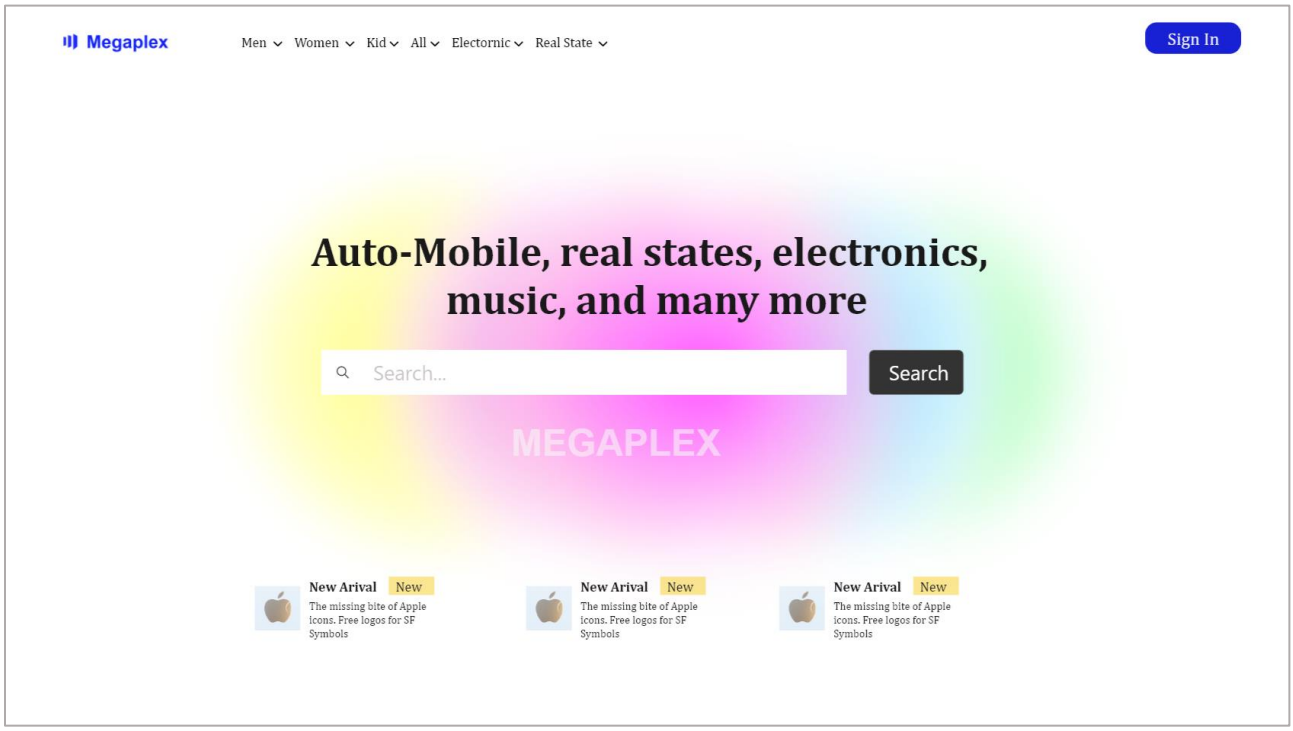



*Figure 26: Homepage*

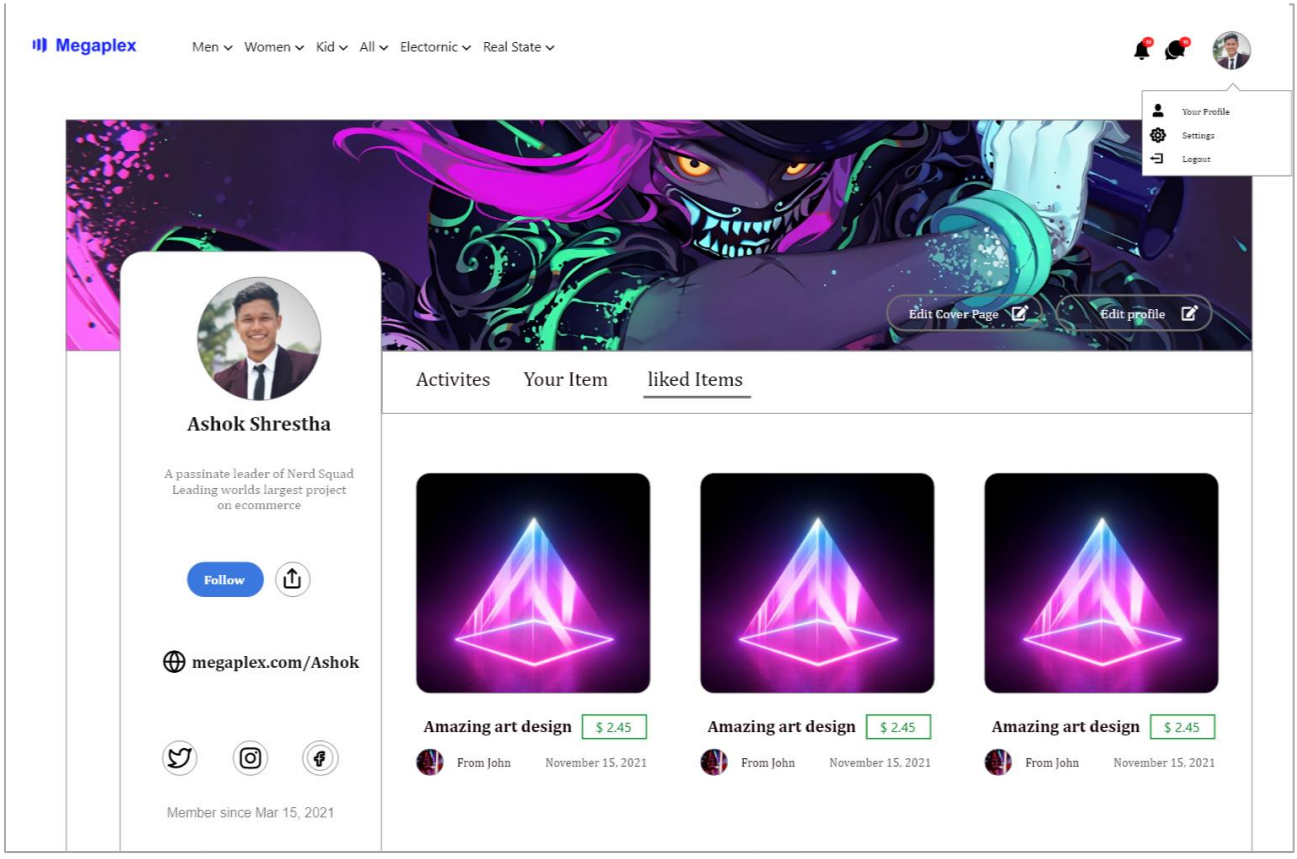



*Figure 27: Profile page*

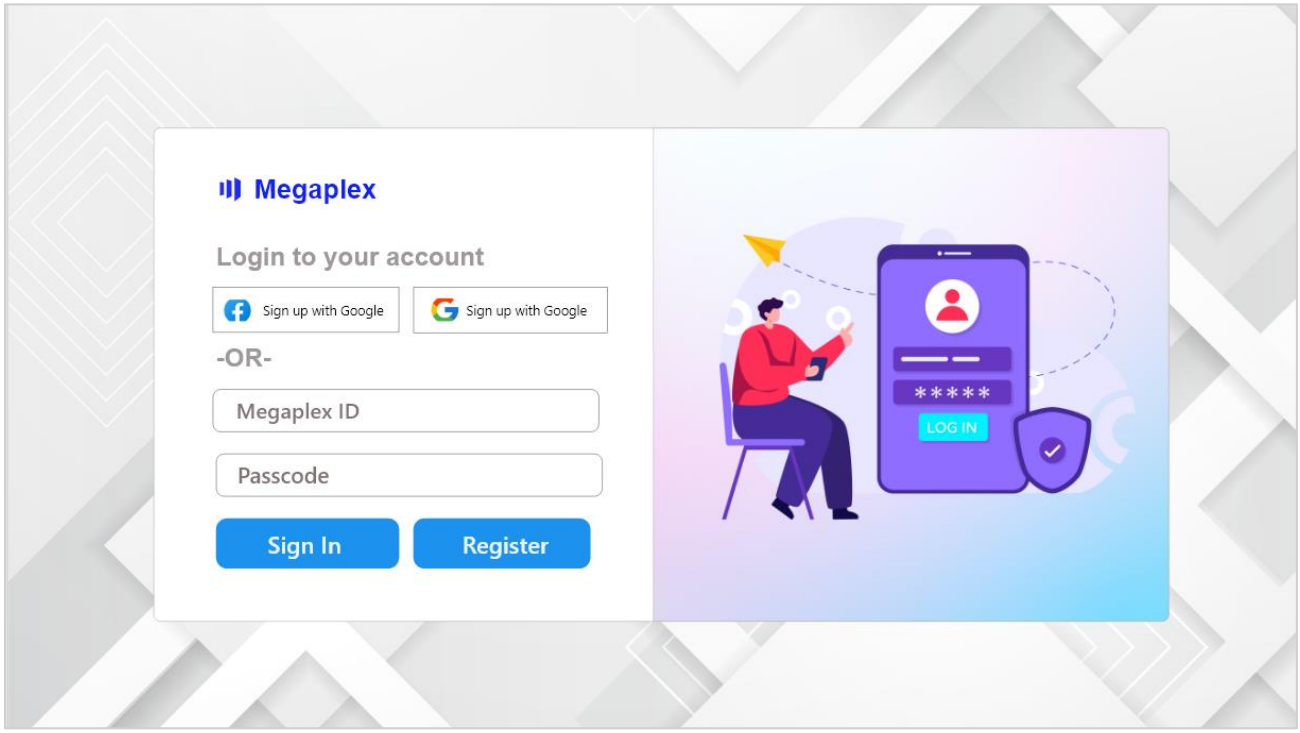



*Figure 28: Login page*

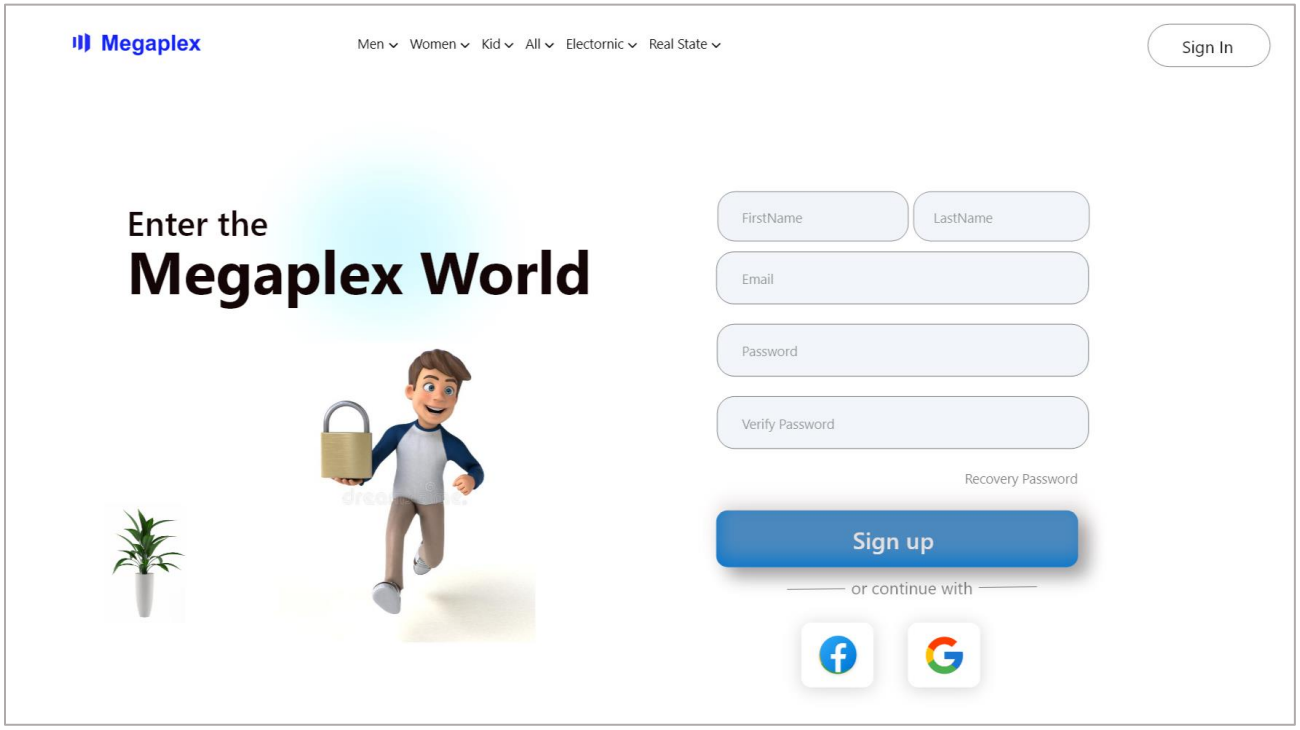



*Figure 29: Signup Page*

# Final Product

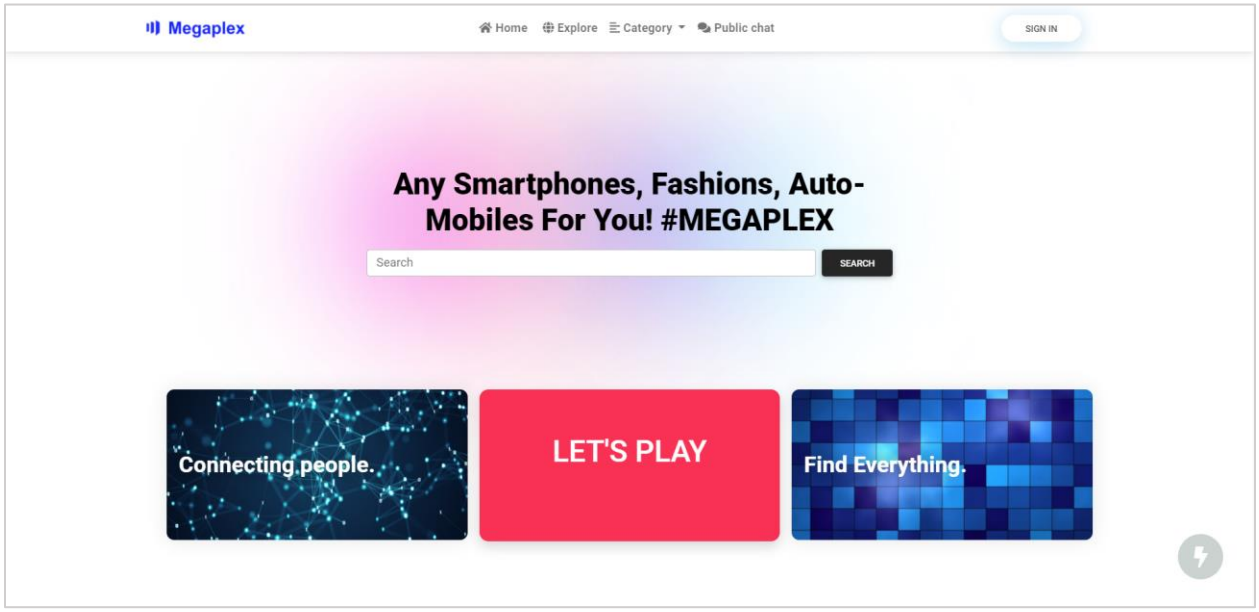



*Figure 30: Homepage*

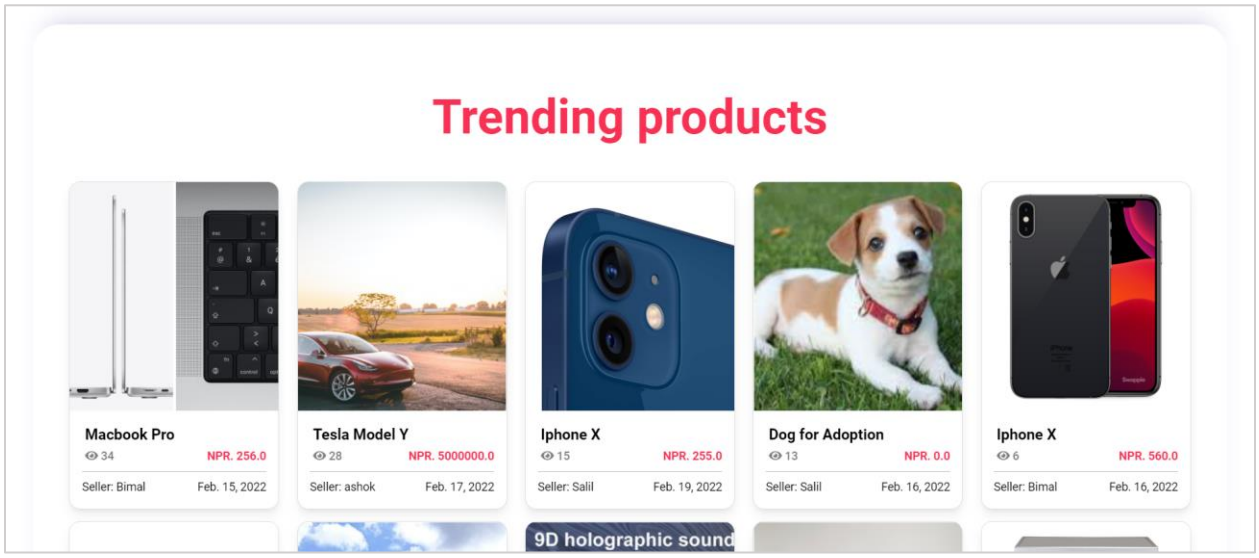



*Figure 31: Homepage*

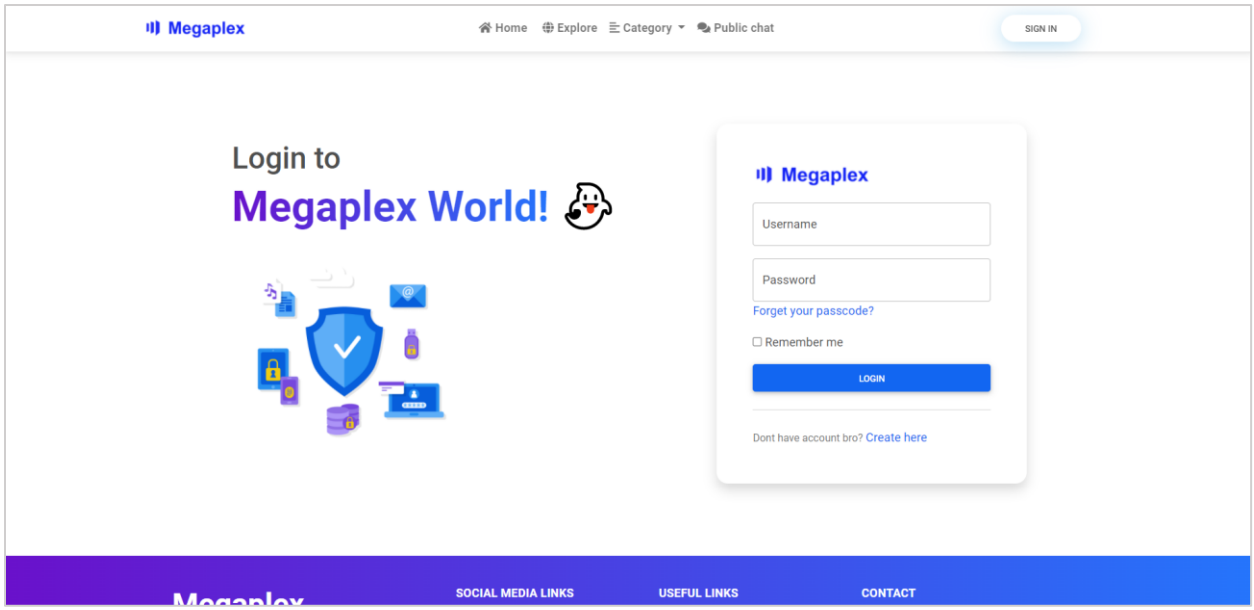



*Figure 32: Login page*

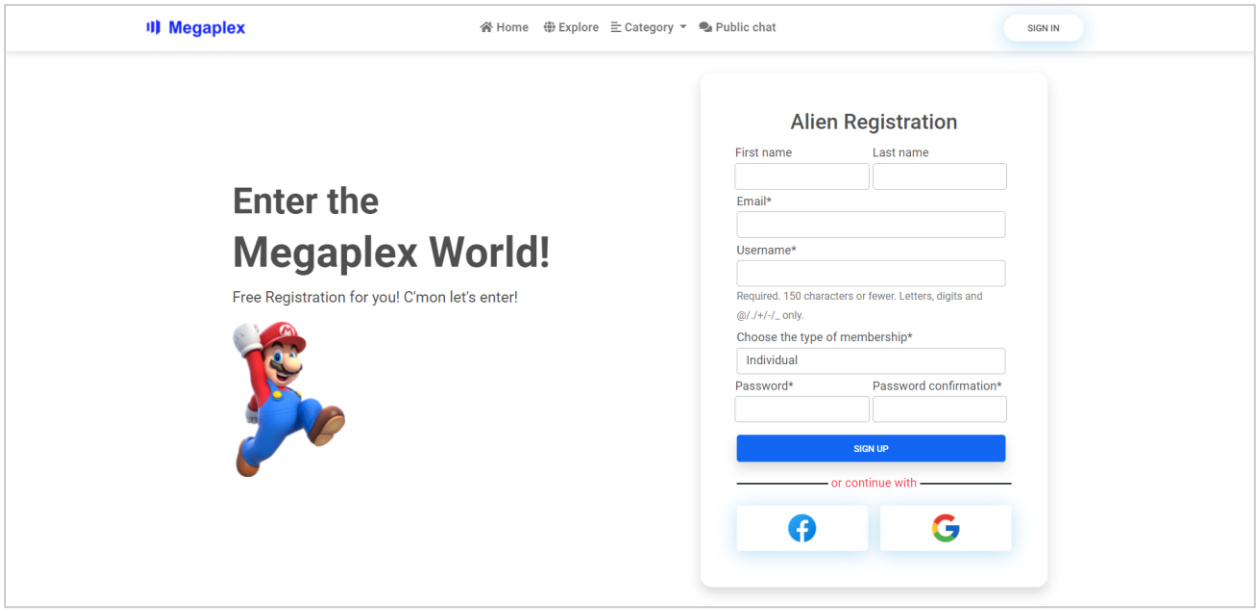



*Figure 33: Registration page*

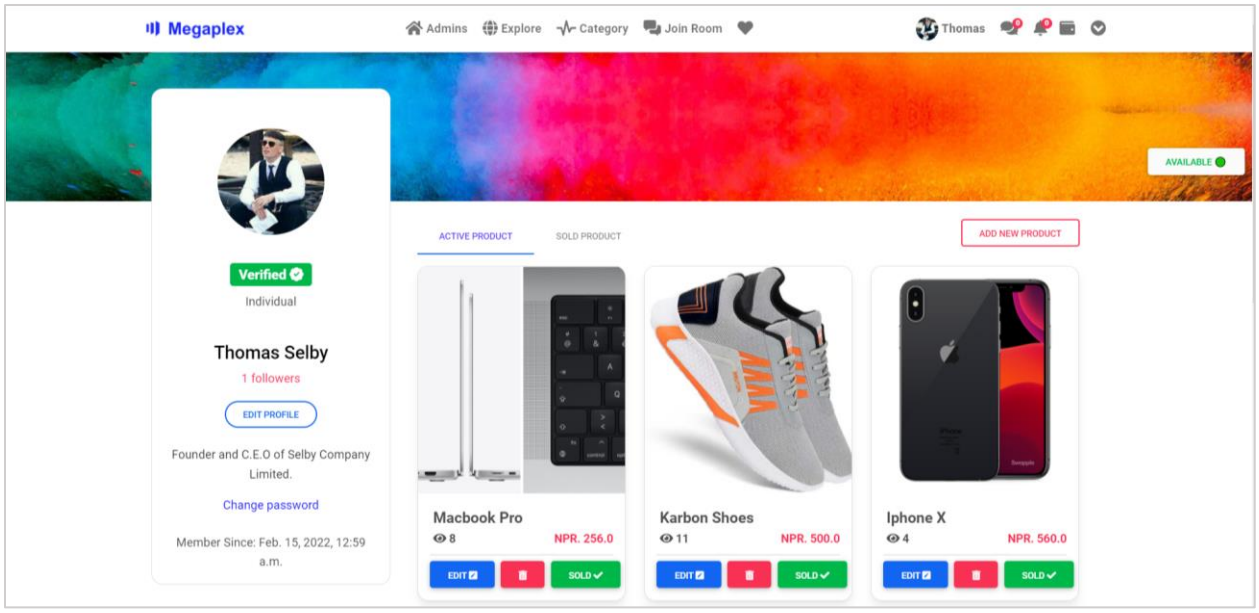



*Figure 34: User profile*

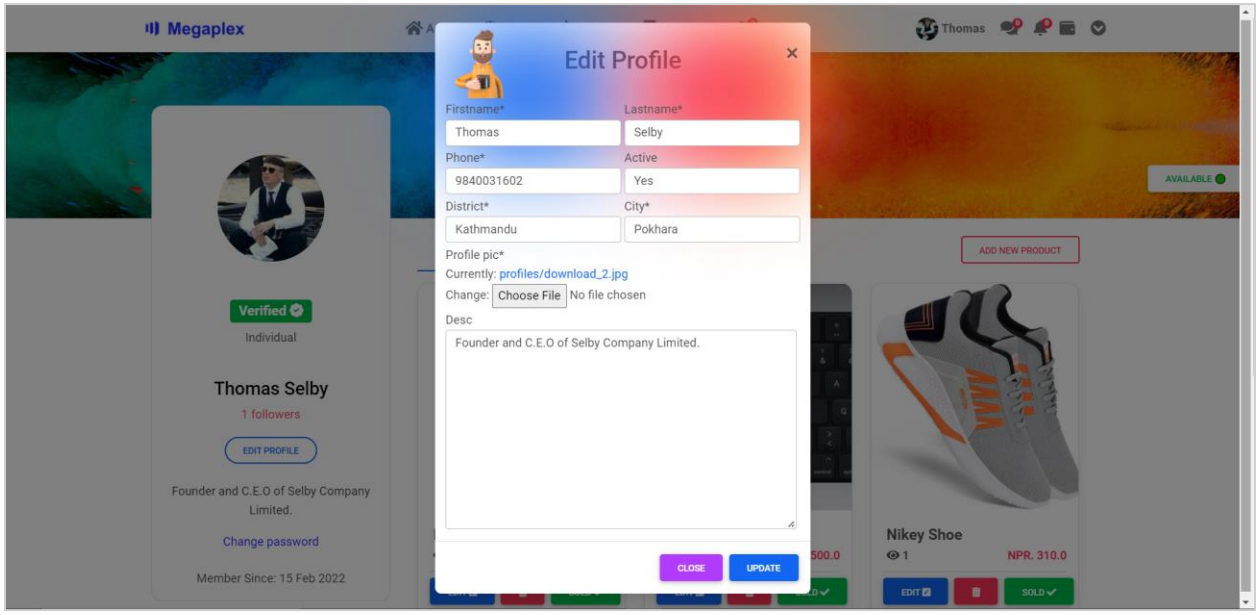



*Figure 35: Edit Profile*

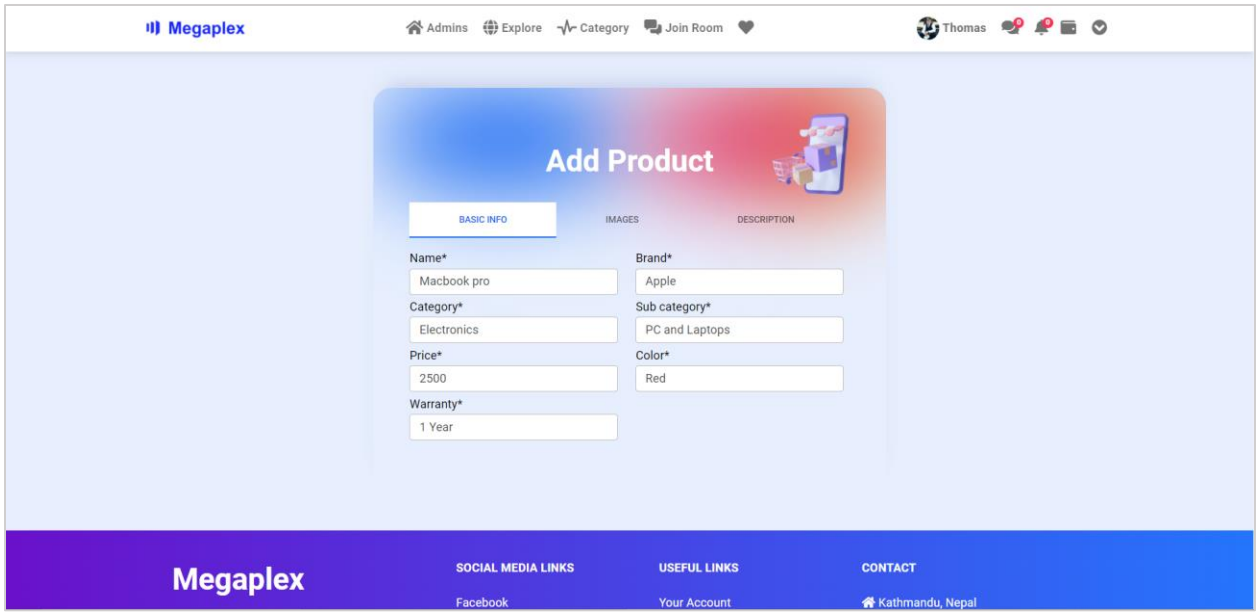



*Figure 36: Adding Product*

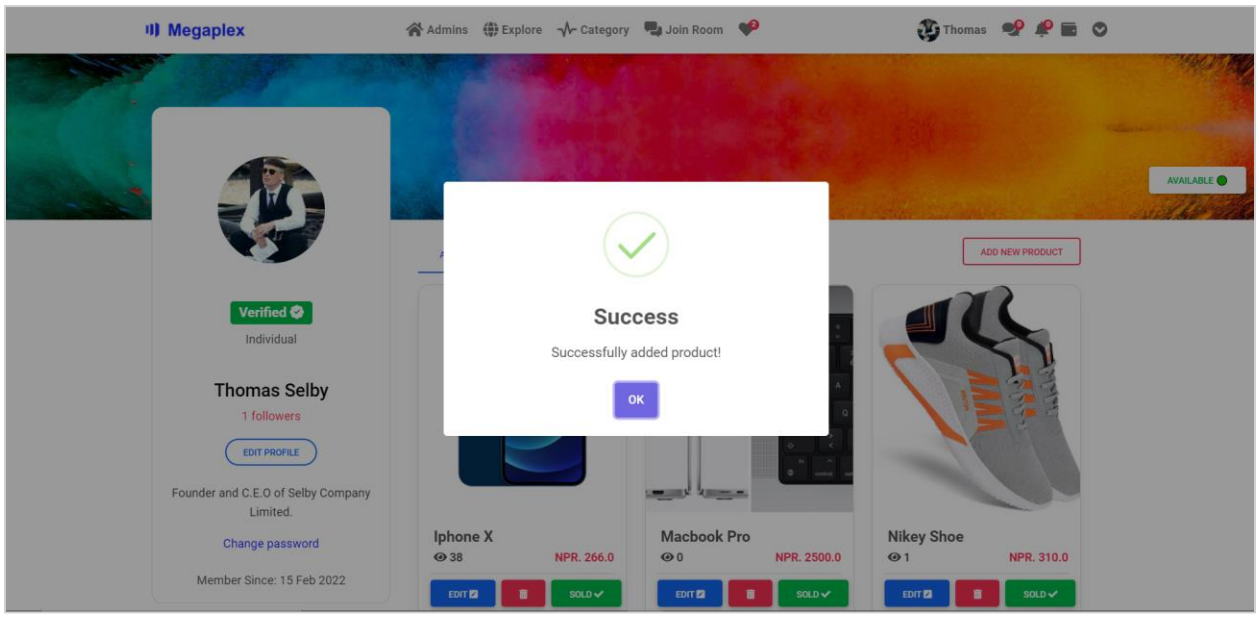



*Figure 37: Confirmation Page for Product Additions*

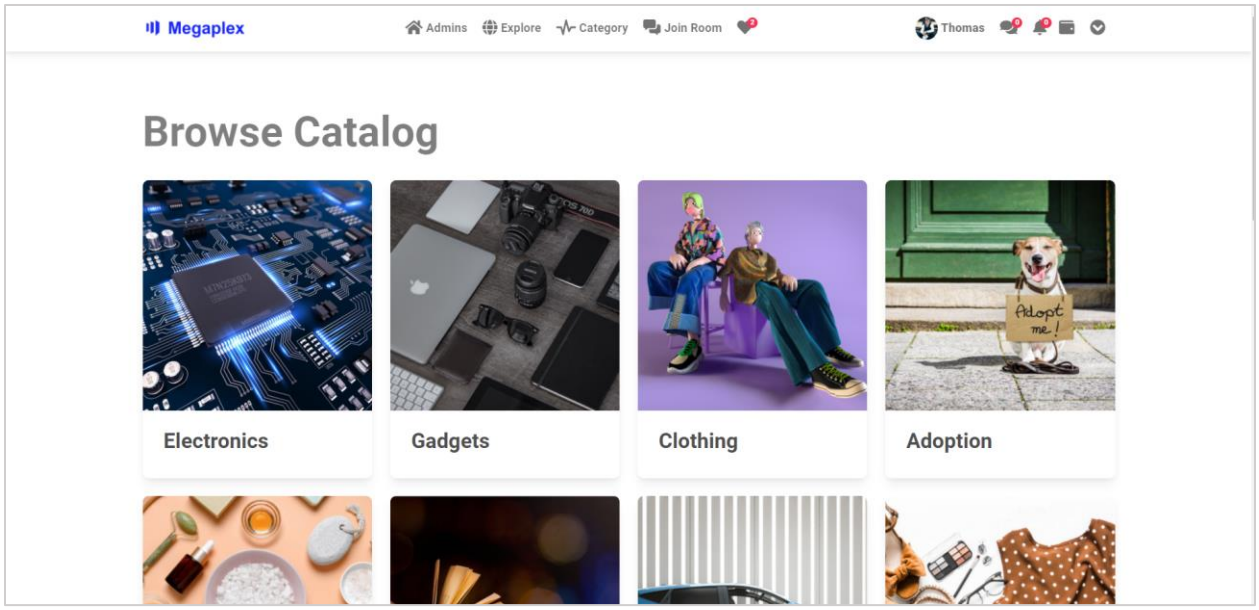



*Figure 38: Browse Products*

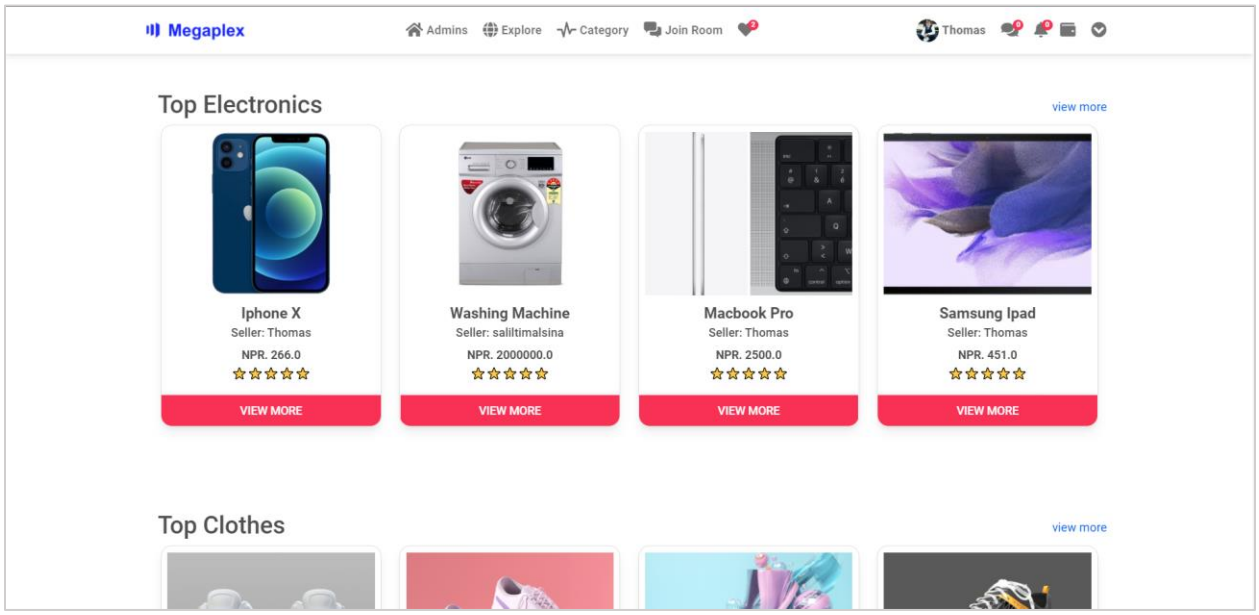



*Figure 39: View the top products in each category.*

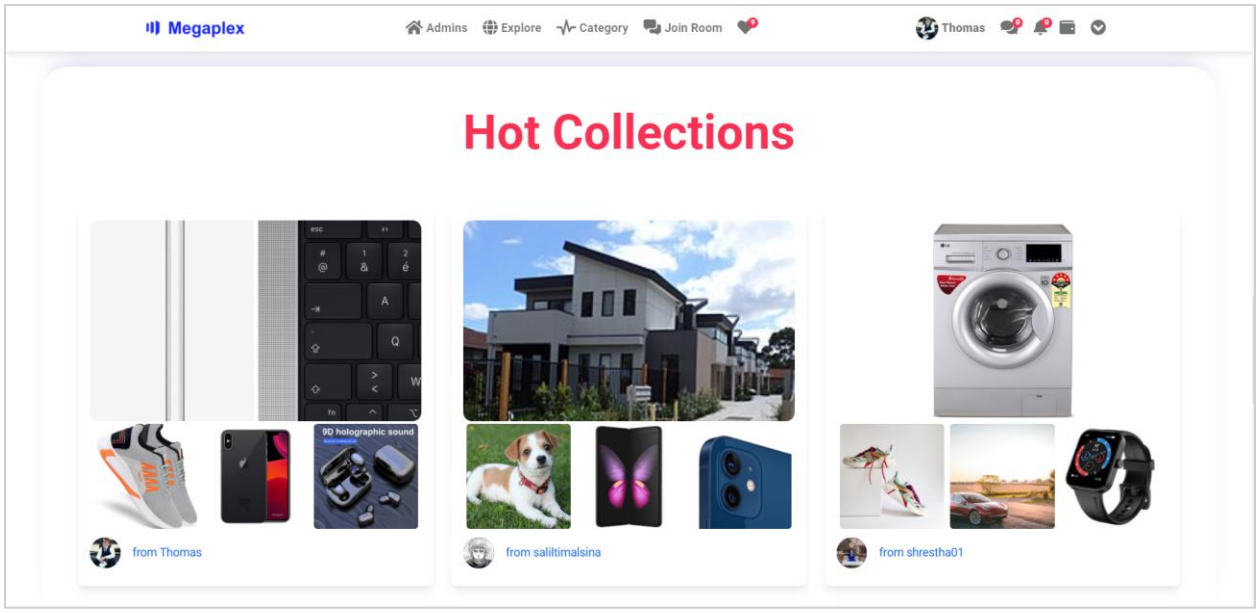



*Figure 40: Hot Collections*

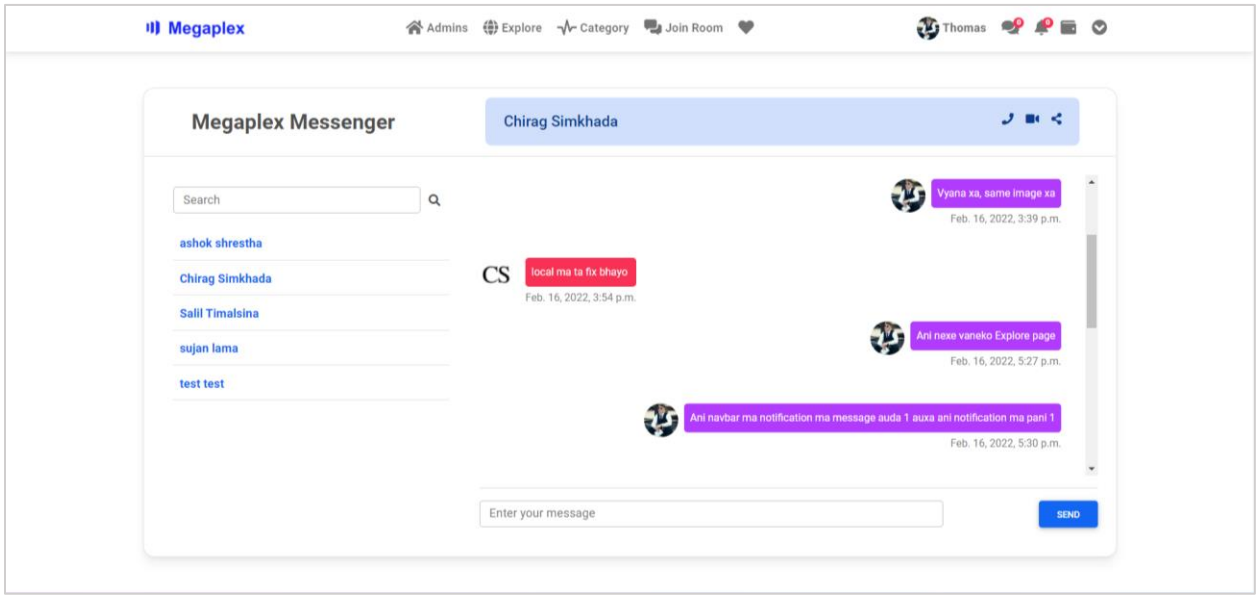



*Figure 41: Megaplex Built in Chat*

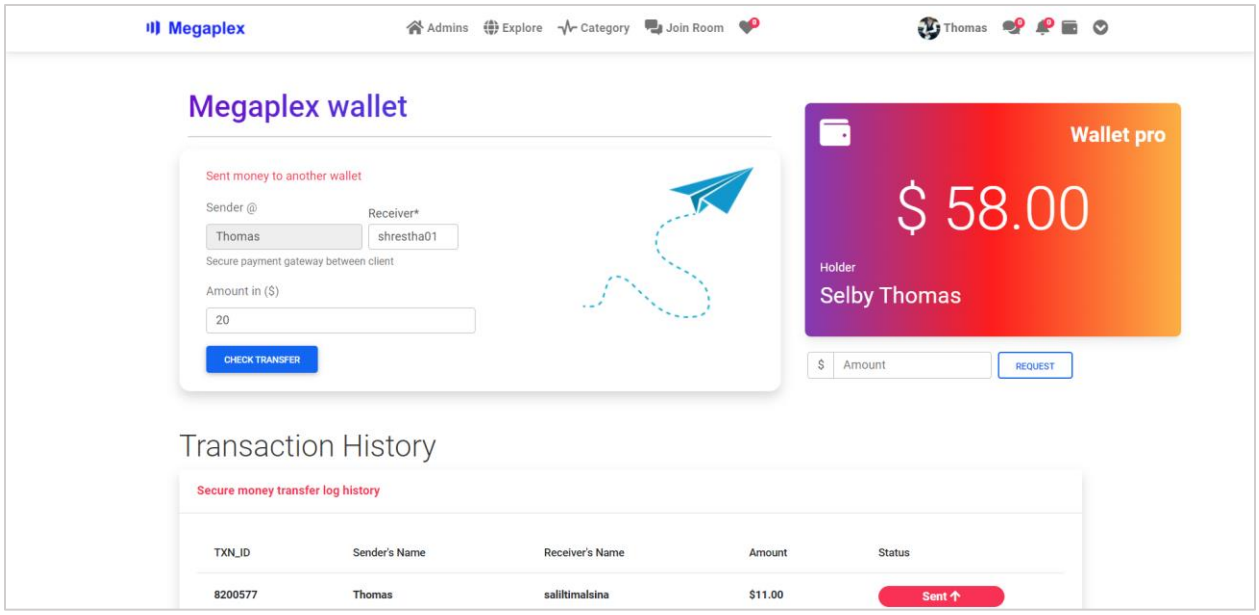



*Figure 42: Built-in Wallet*

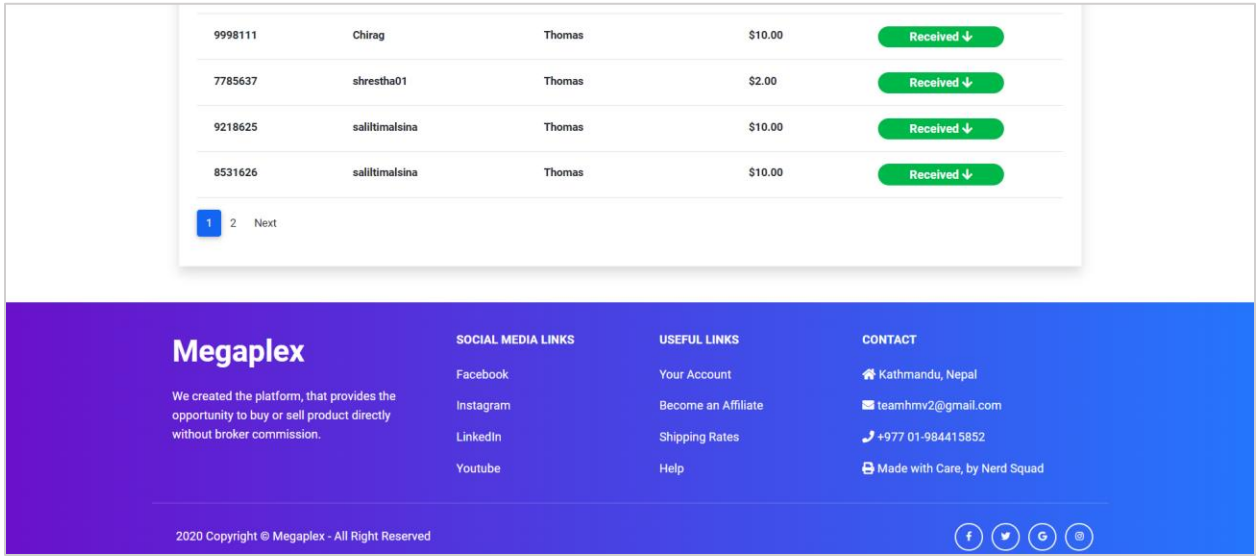



*Figure 43: Transaction History*

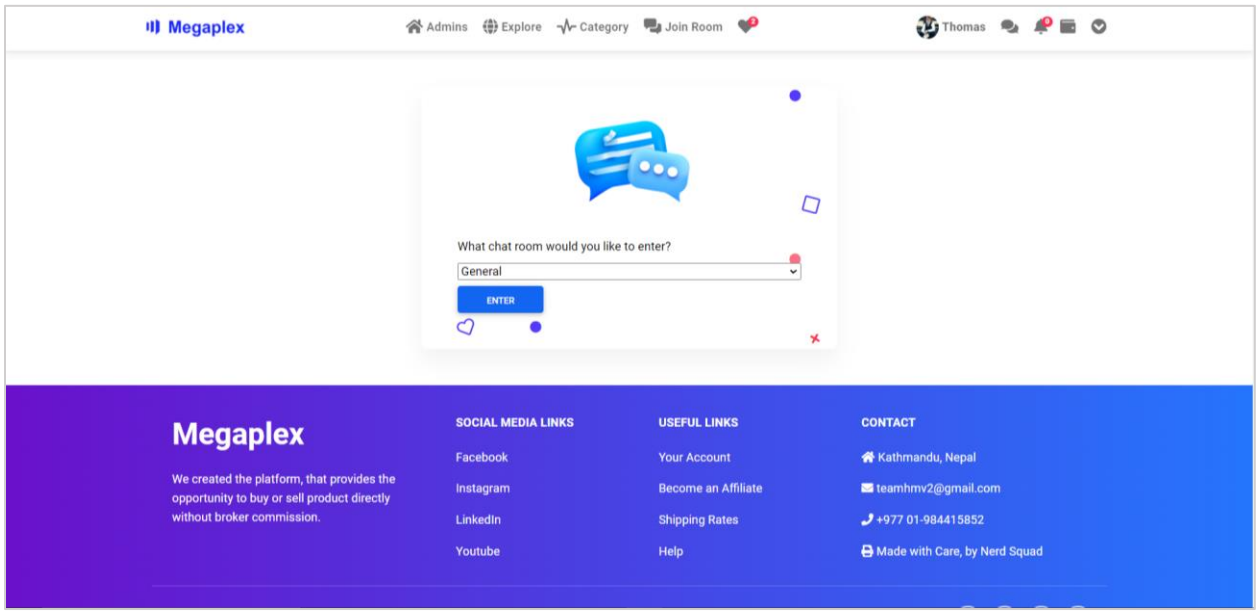



*Figure 44: Chatroom*

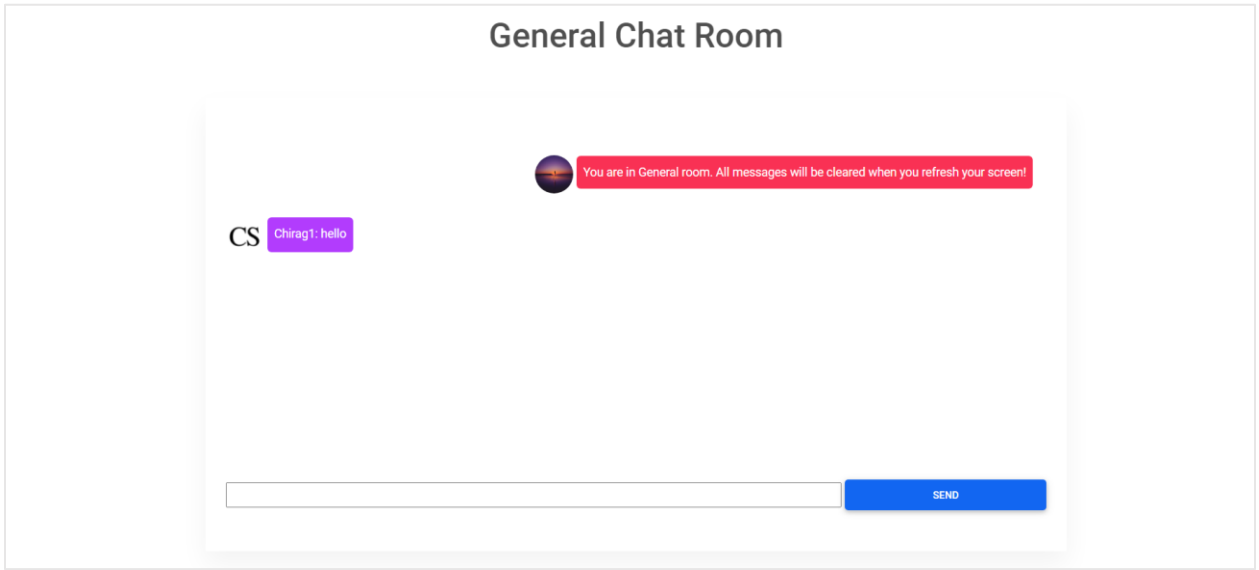



*Figure 45: General Chat Room*

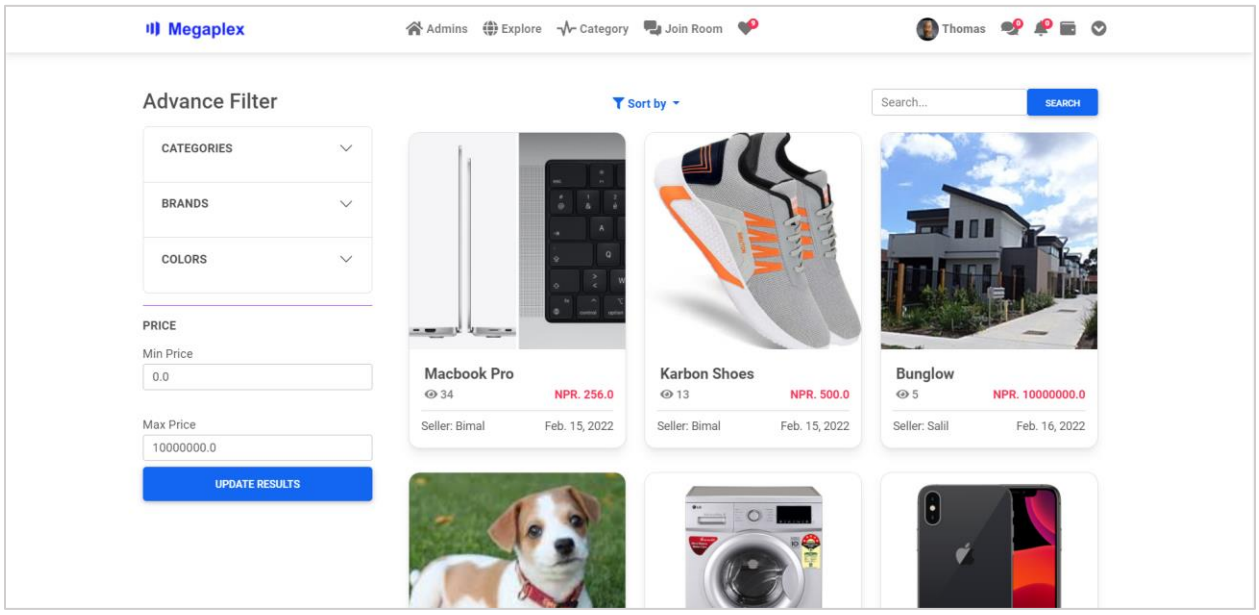



*Figure 46: Filtering Products*

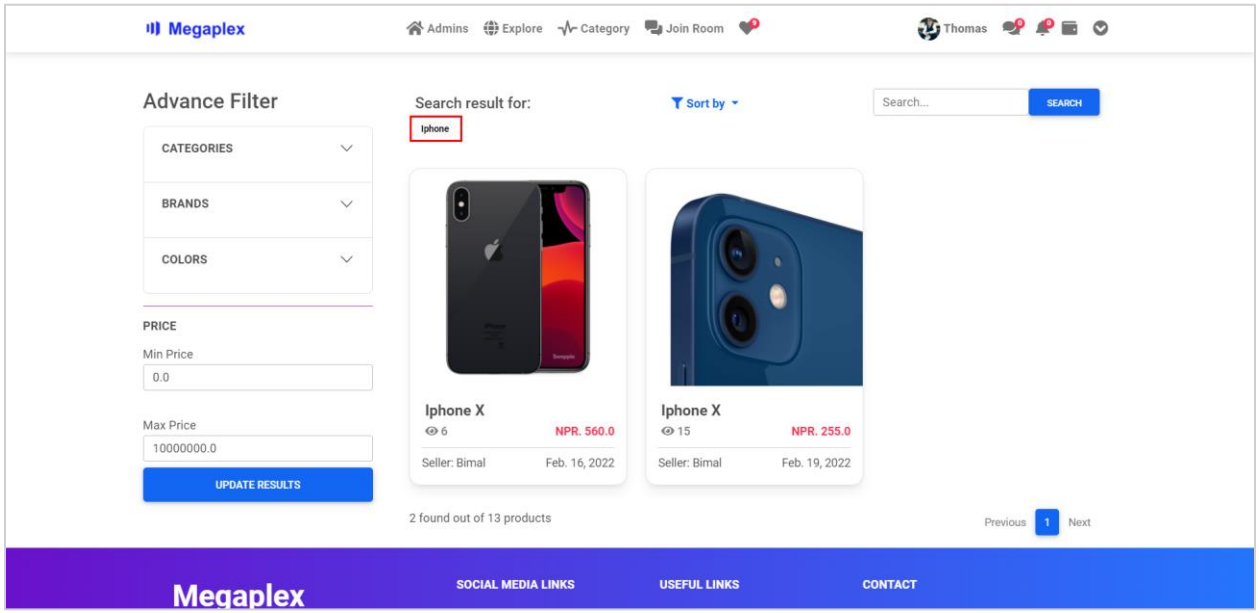



*Figure 47: Search result*

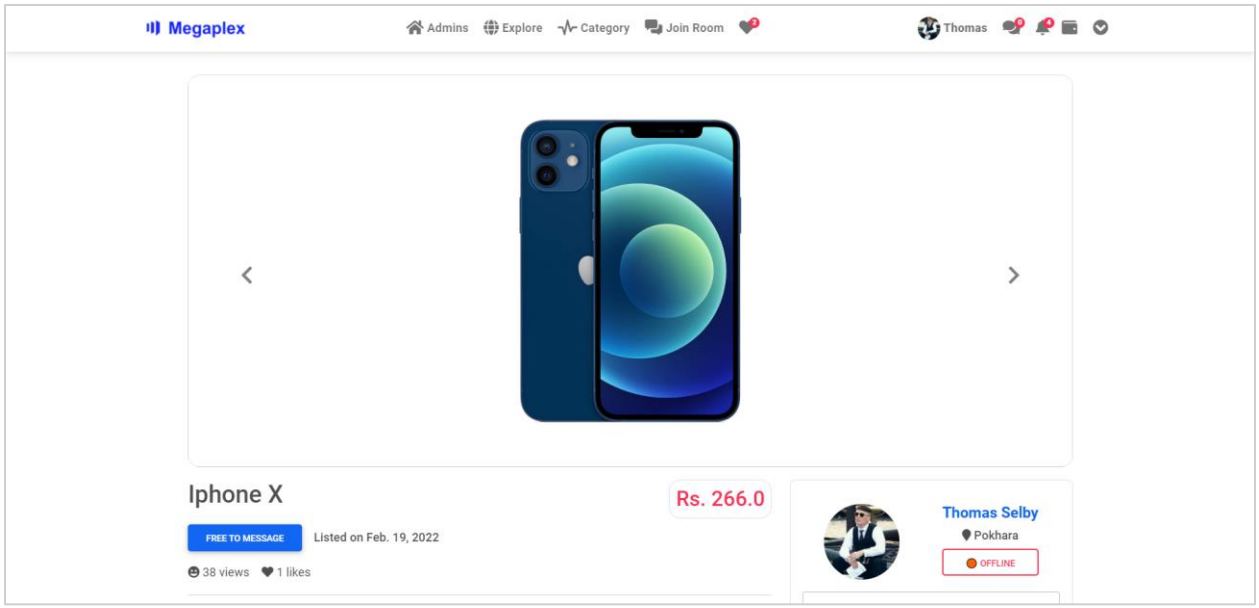



*Figure 48: Product Overview*

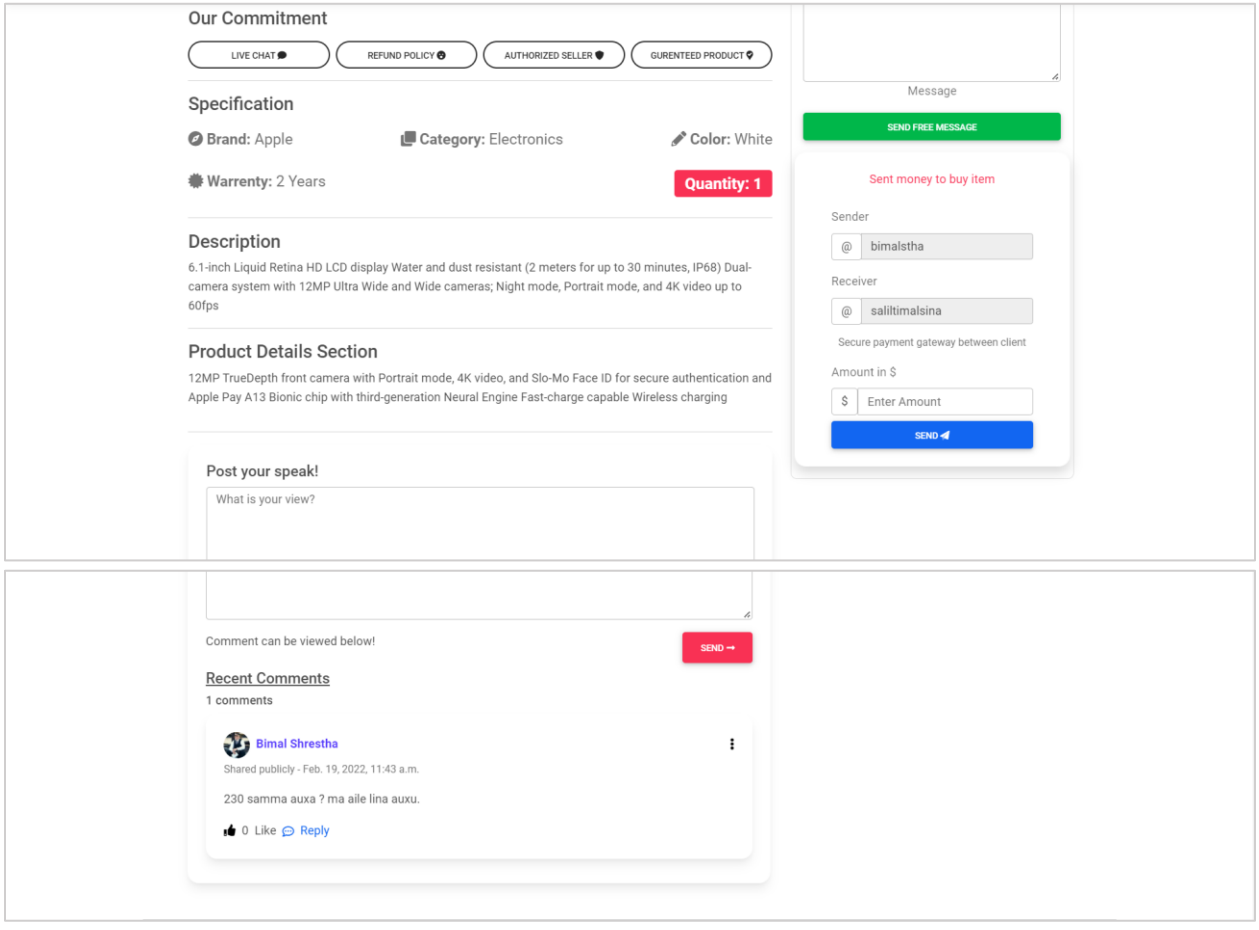



*Figure 49: Product details page*

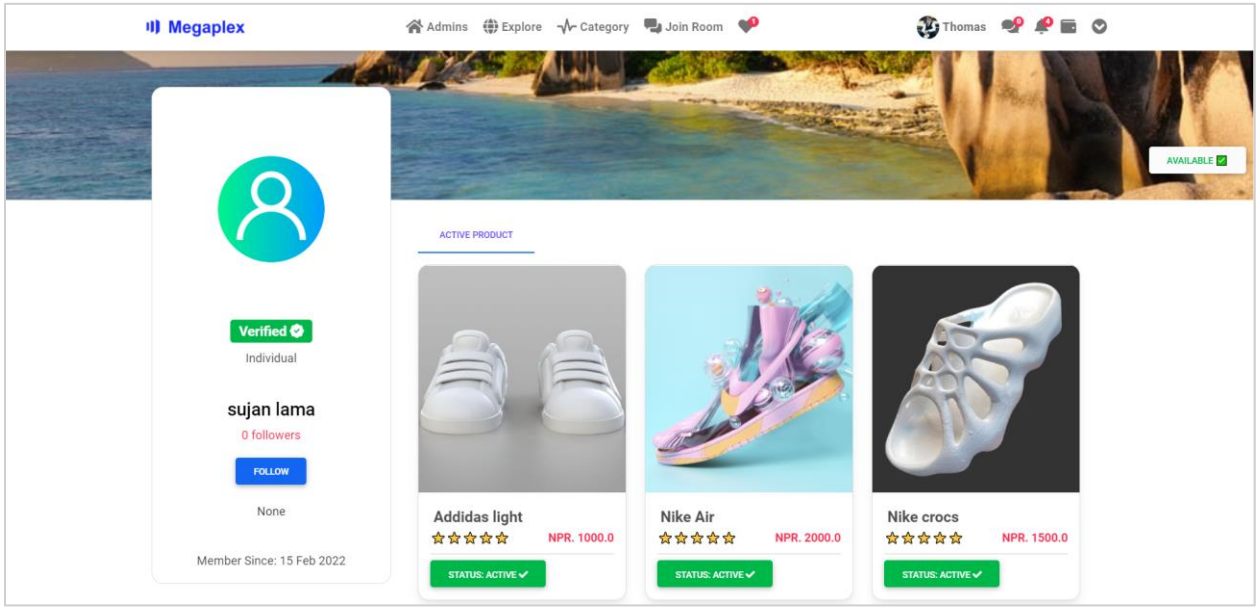



*Figure 50: User Profile*

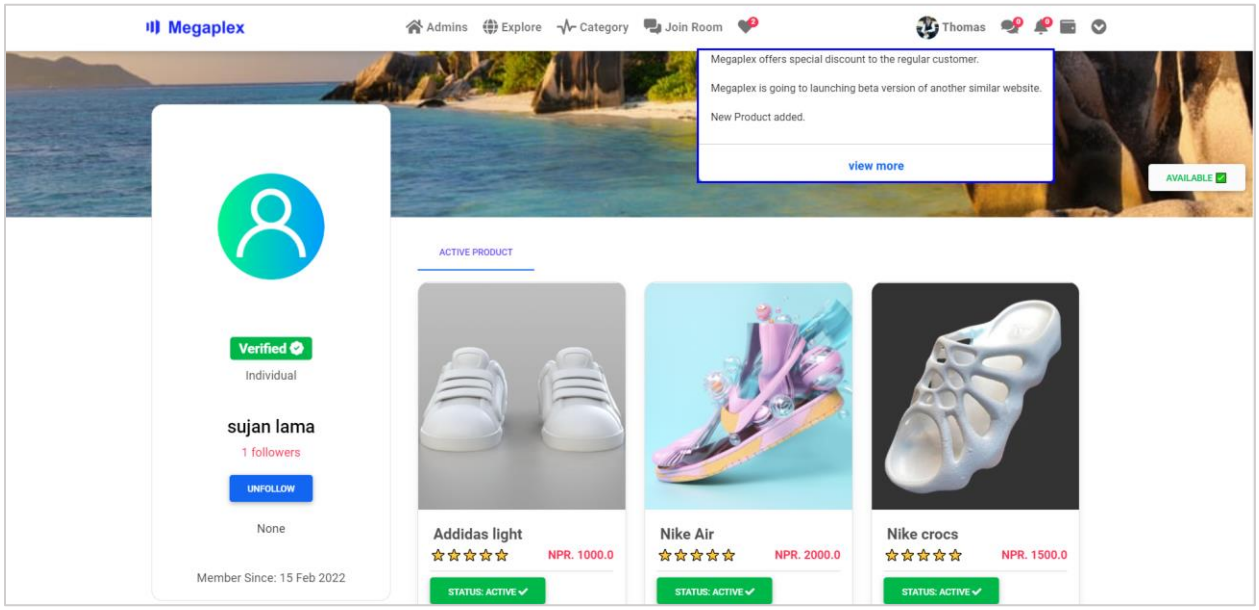



*Figure 51: Megaplex Notification*

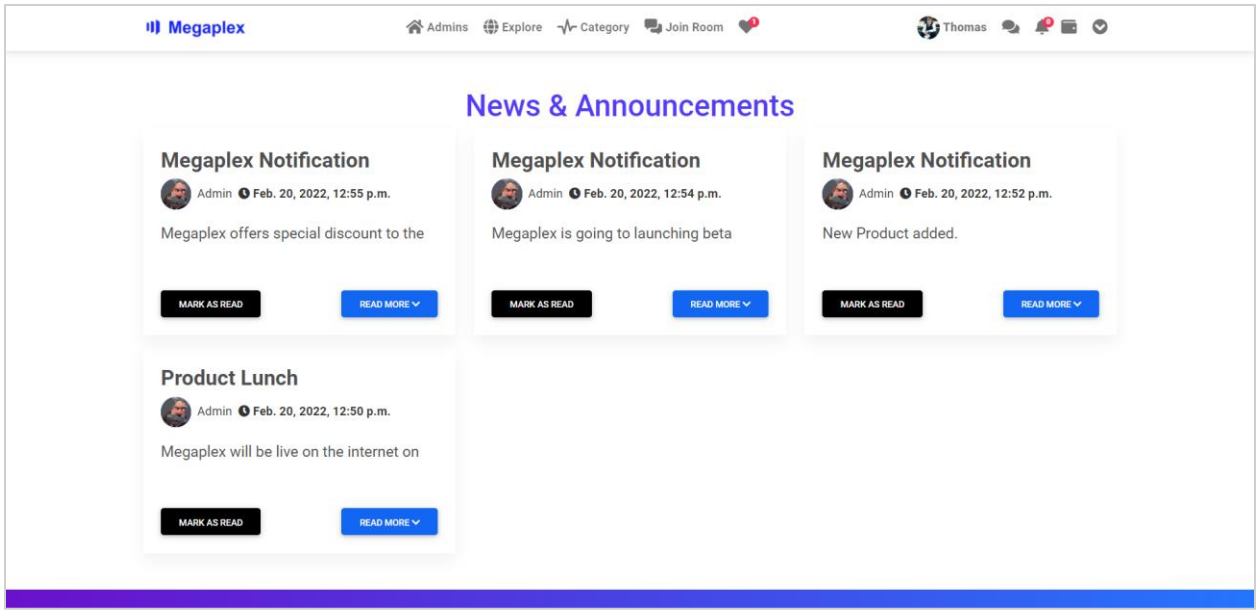



*Figure 52: News and announcement page*

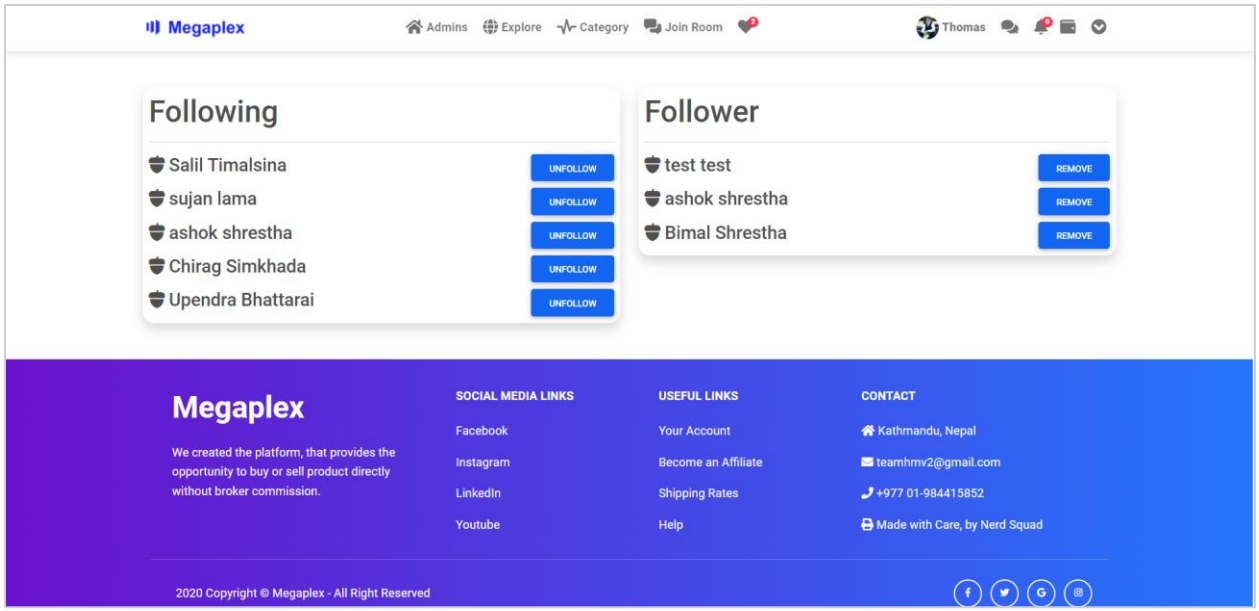



*Figure 53: User follower and the following page*

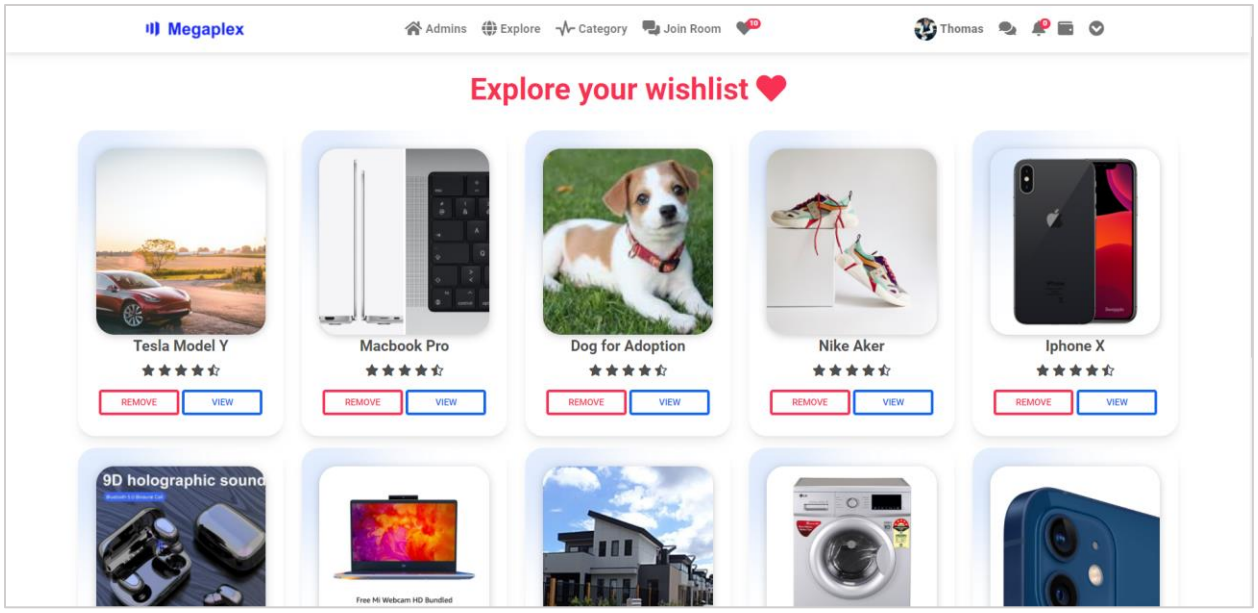



*Figure 54: Wishlist page*

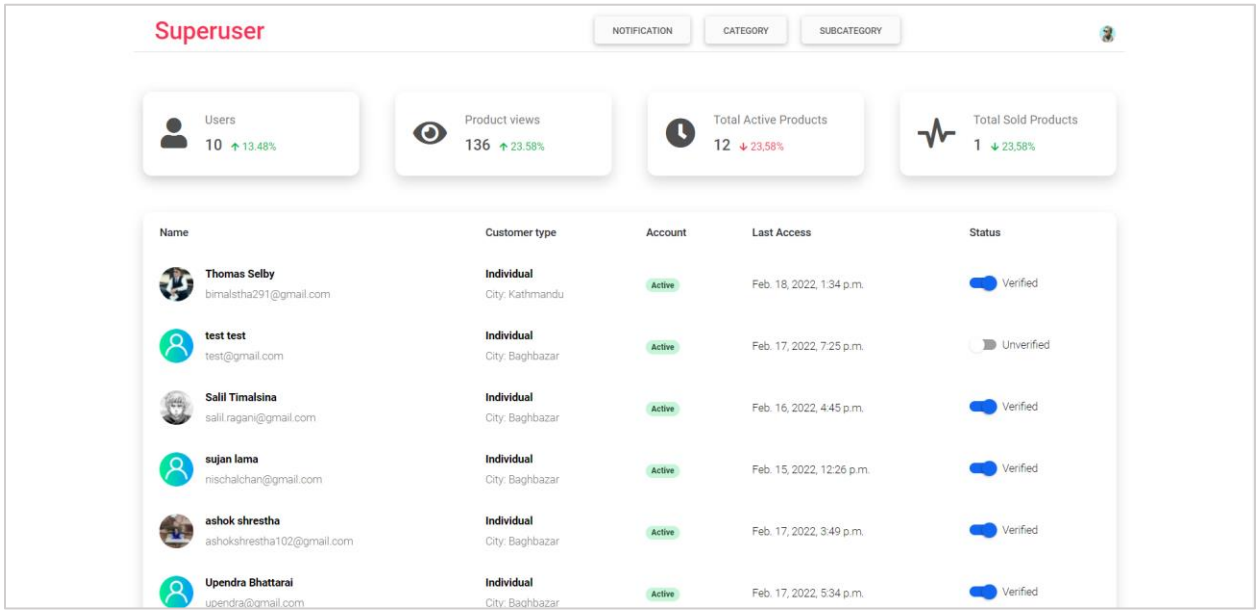



*Figure 55: Admin dashboard*

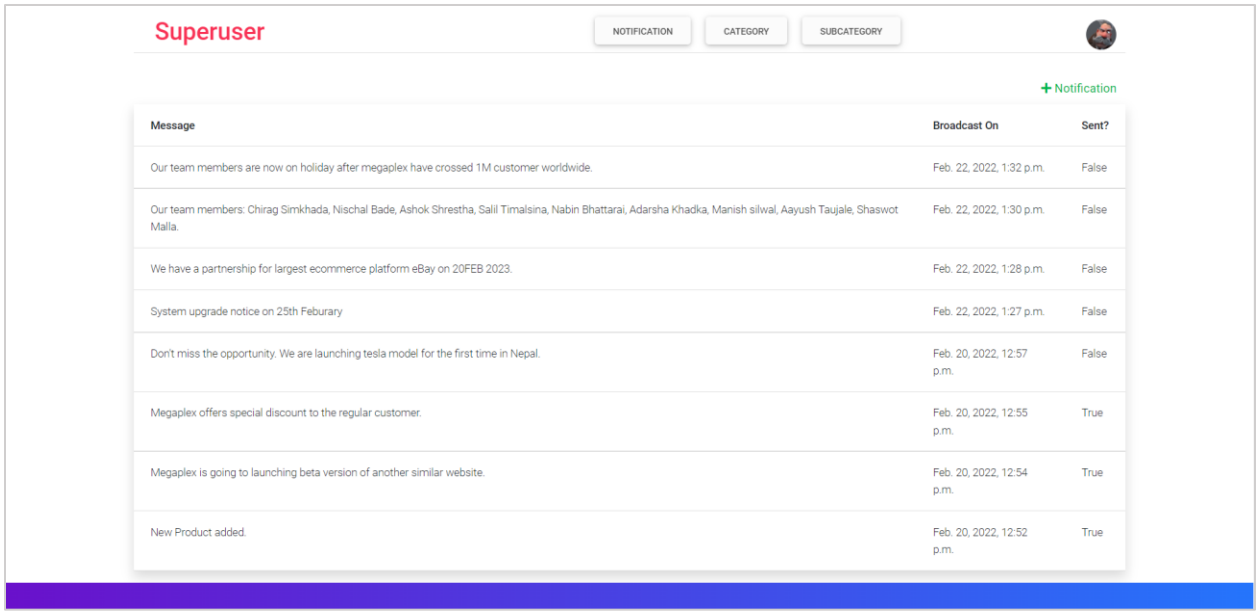



*Figure 56: Page for sending notifications by Admin*

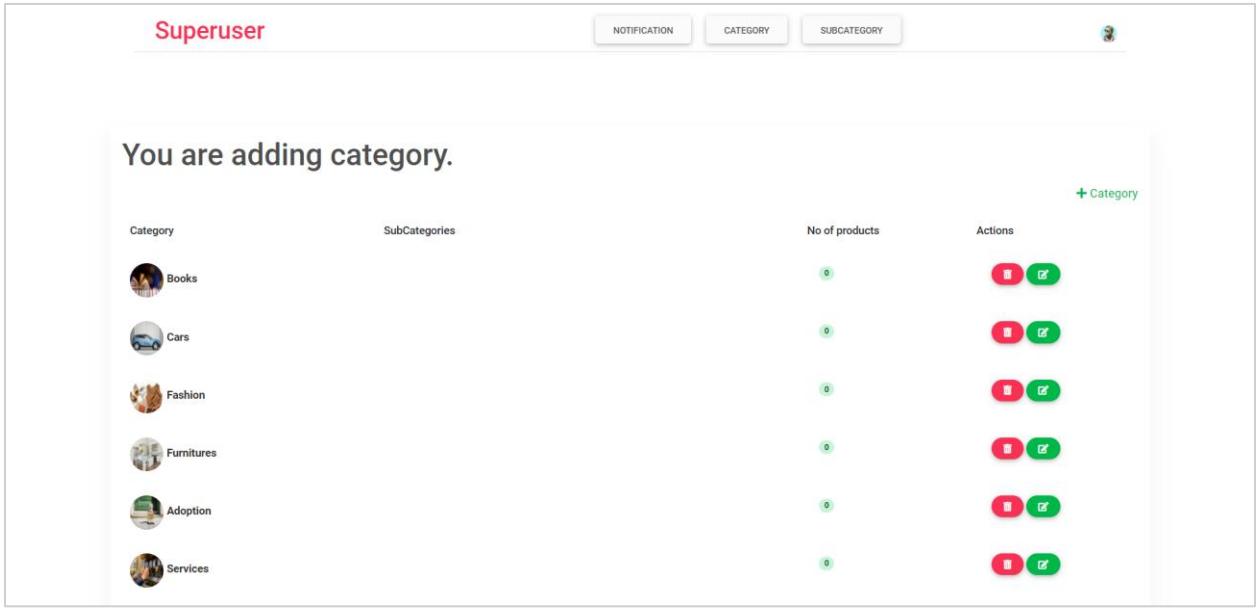



*Figure 57: Admin category viewing page*

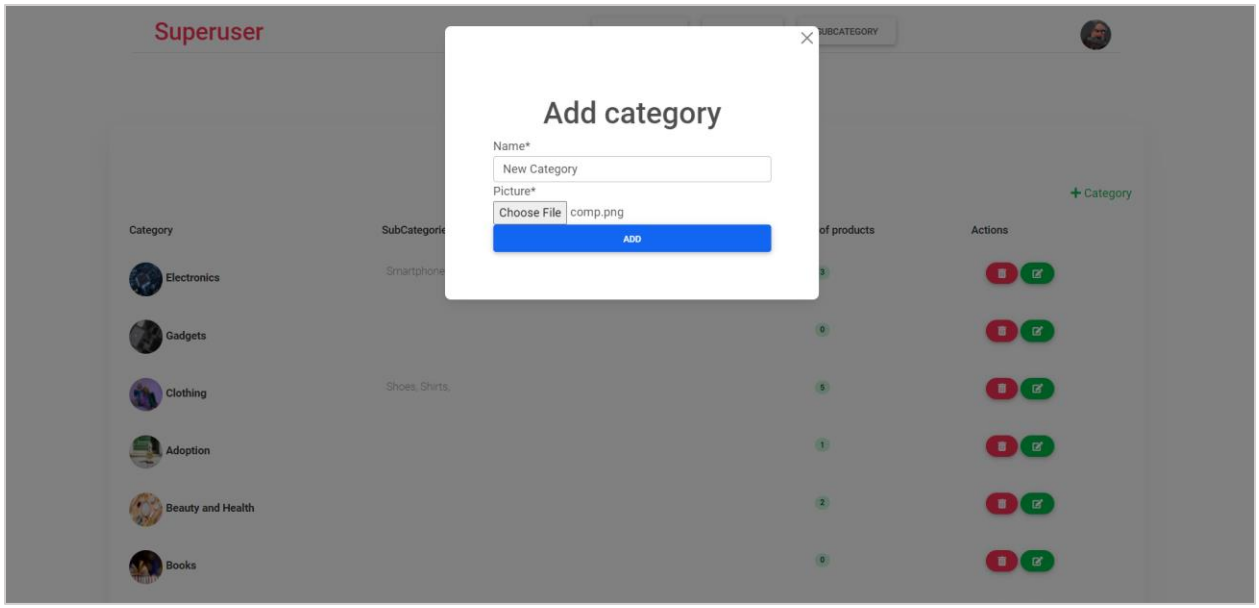



*Figure 58: Admin's page for adding categories*

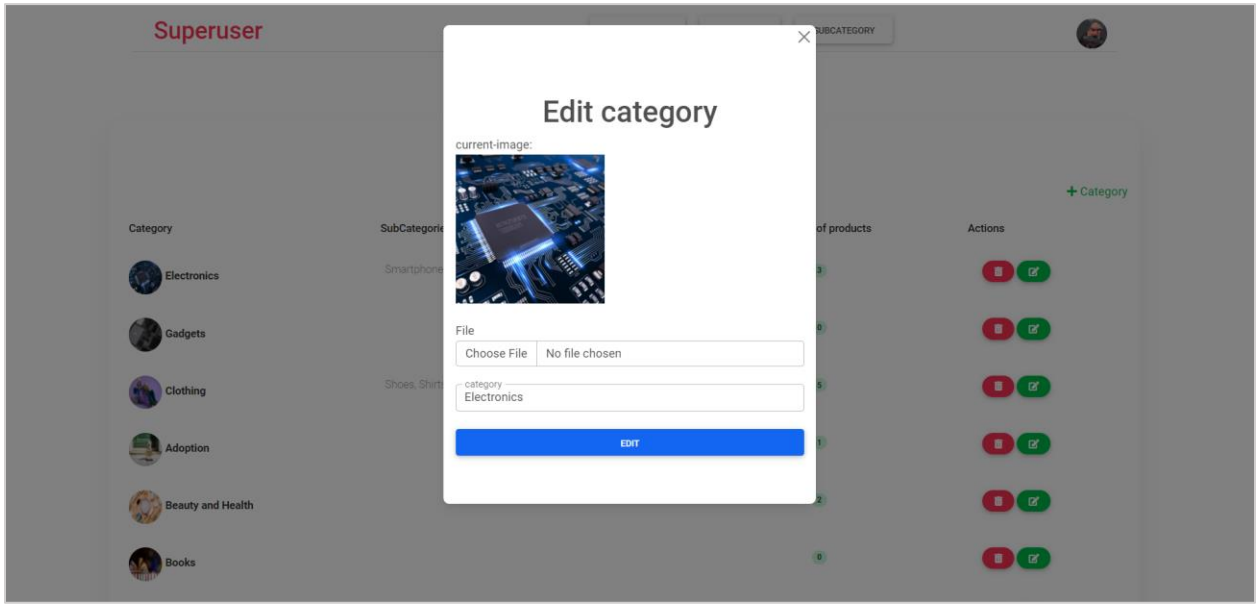



*Figure 59: Admin's page for editing categories*

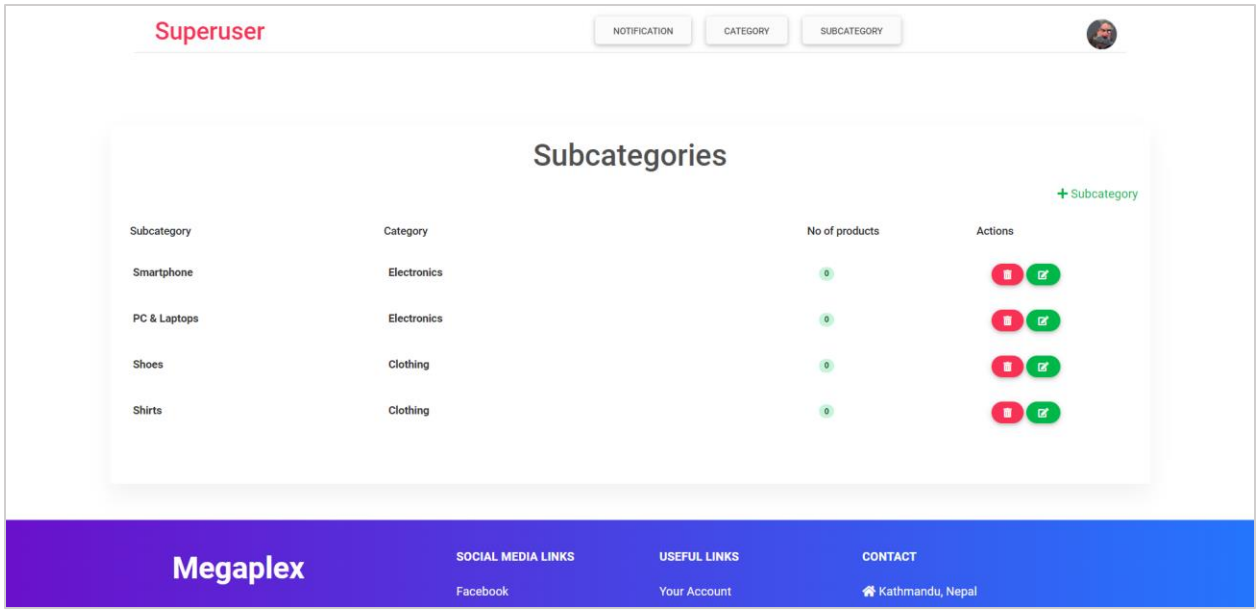



*Figure 60: Admin Subcategory viewing page*

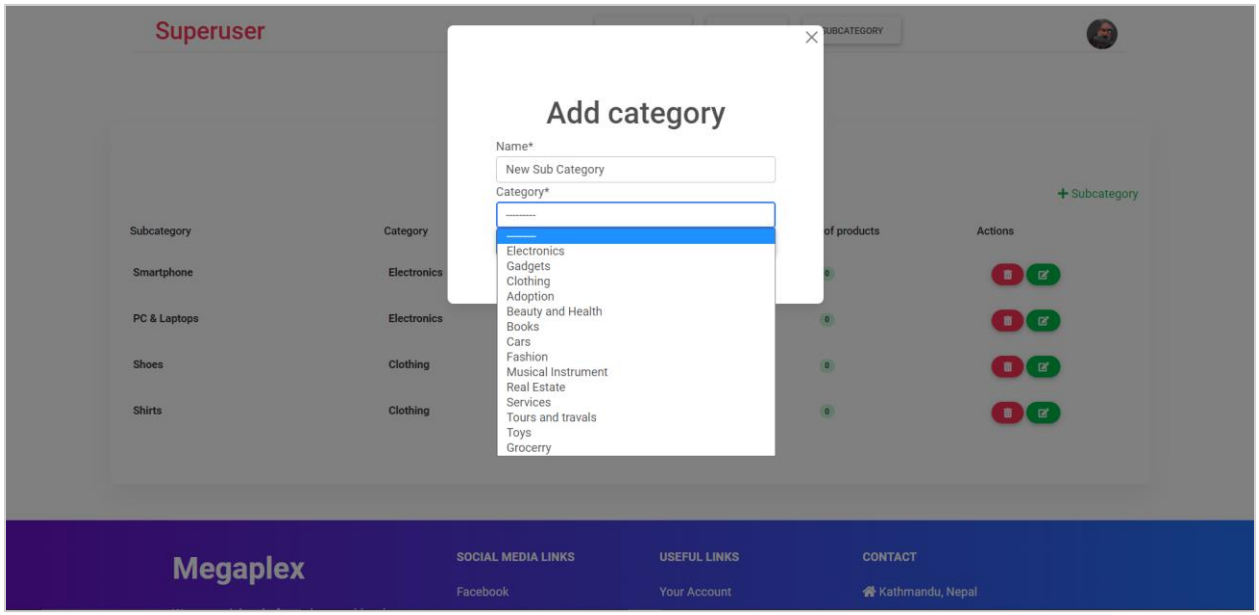



*Figure 61: Admin's page for adding Subcategories*

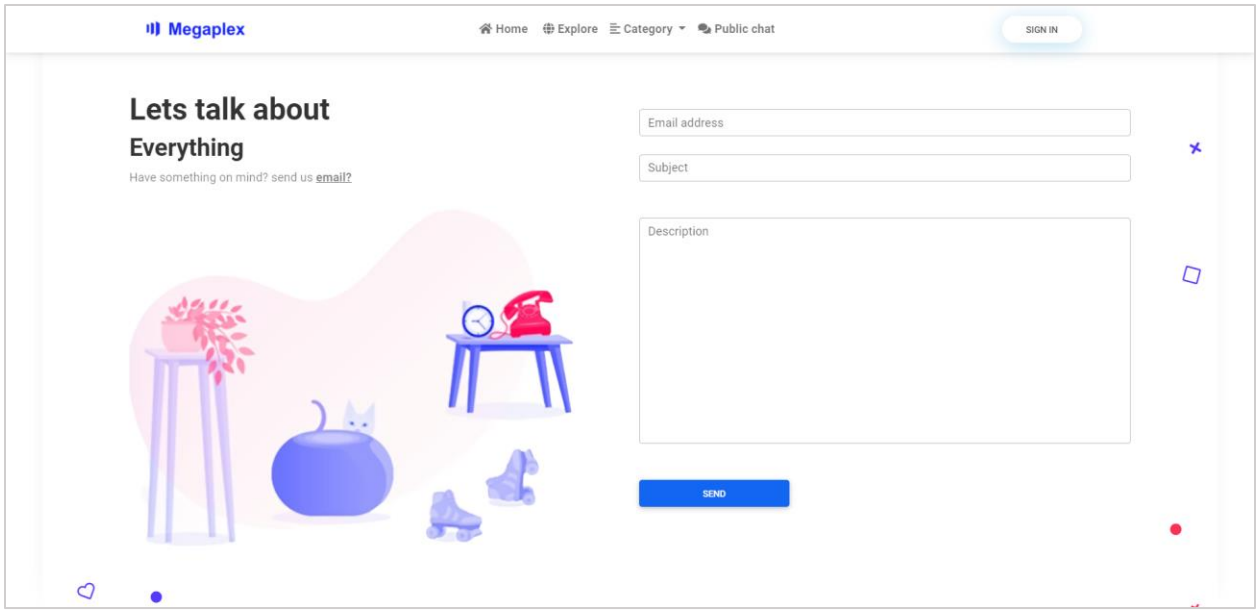



*Figure 62: Contact Form*

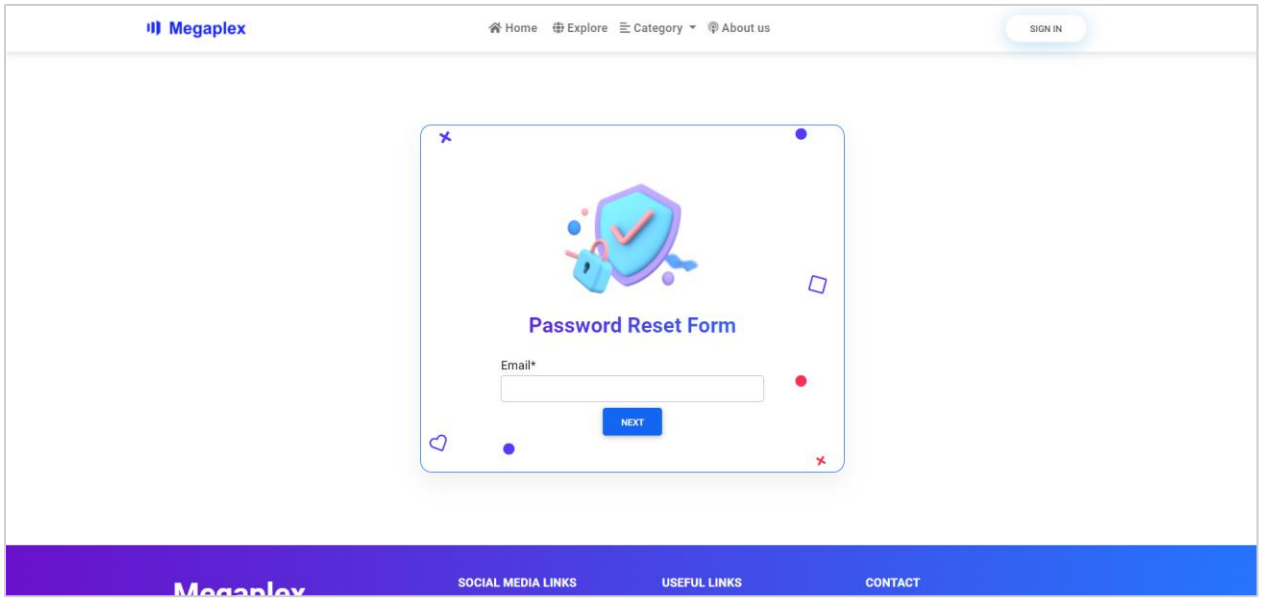



*Figure 63: Password reset form*

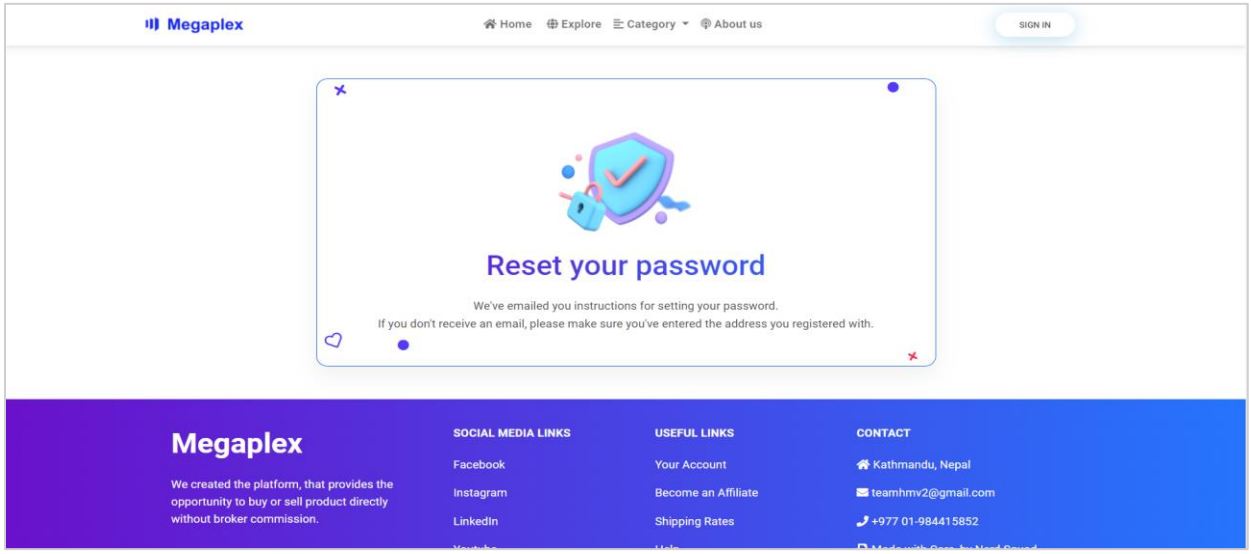



*Figure 64: Password reset form(Email Sent)*

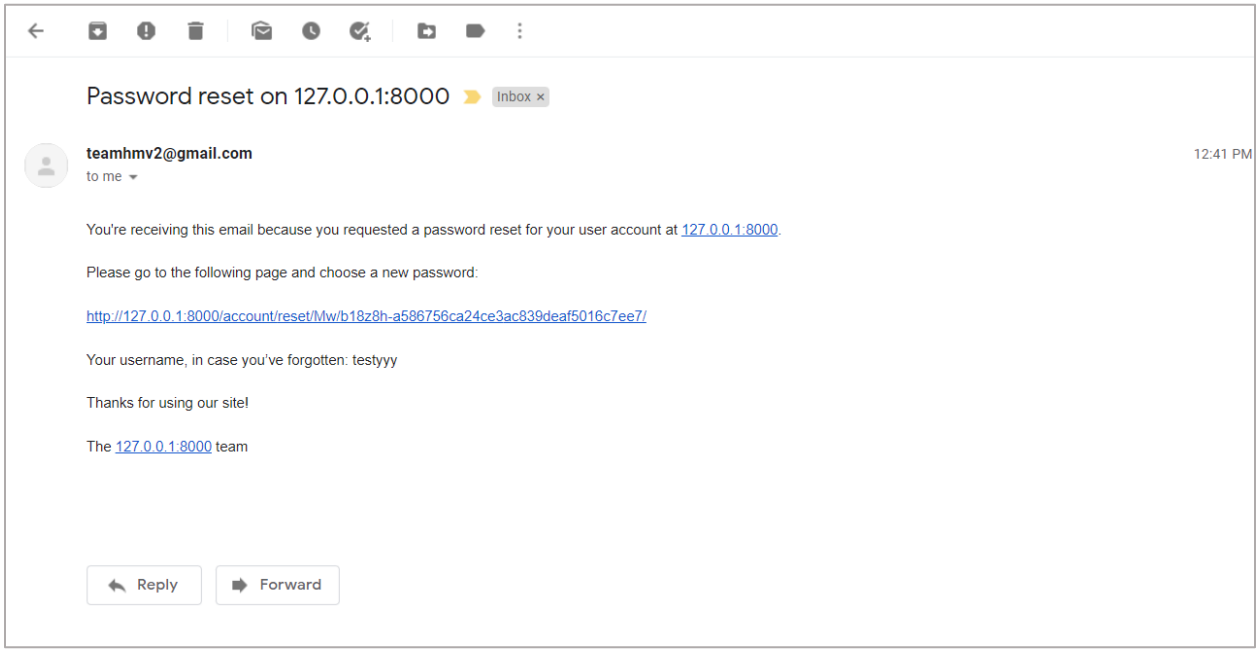



*Figure 65: Password reset (Email Sent)*

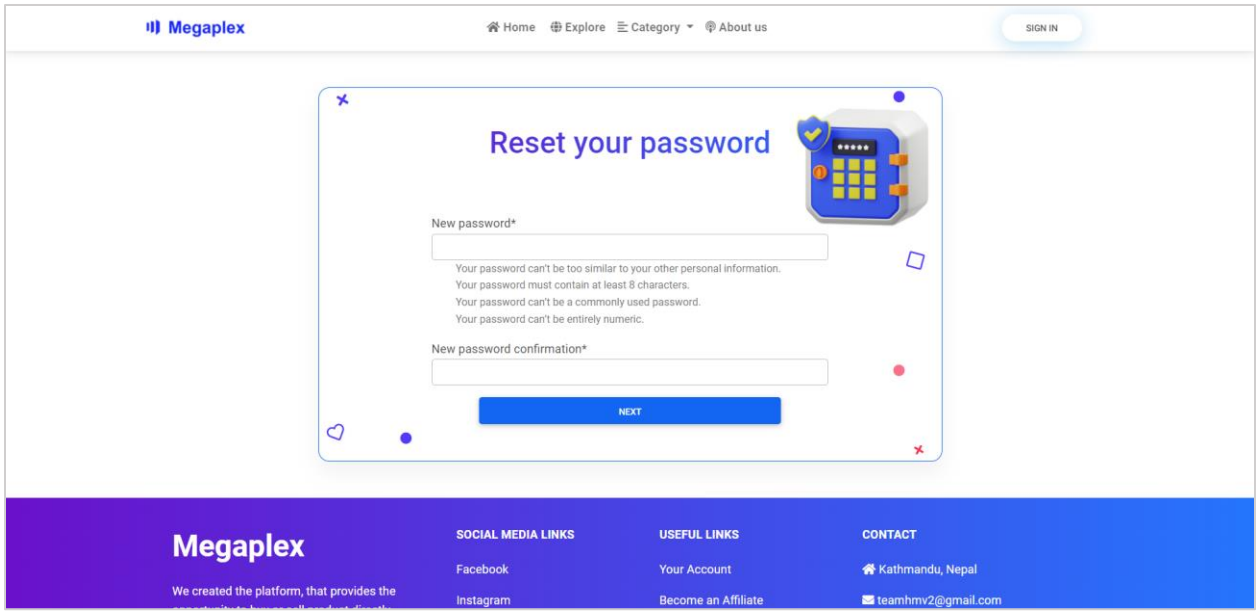



*Figure 66: Page for entering a new password*

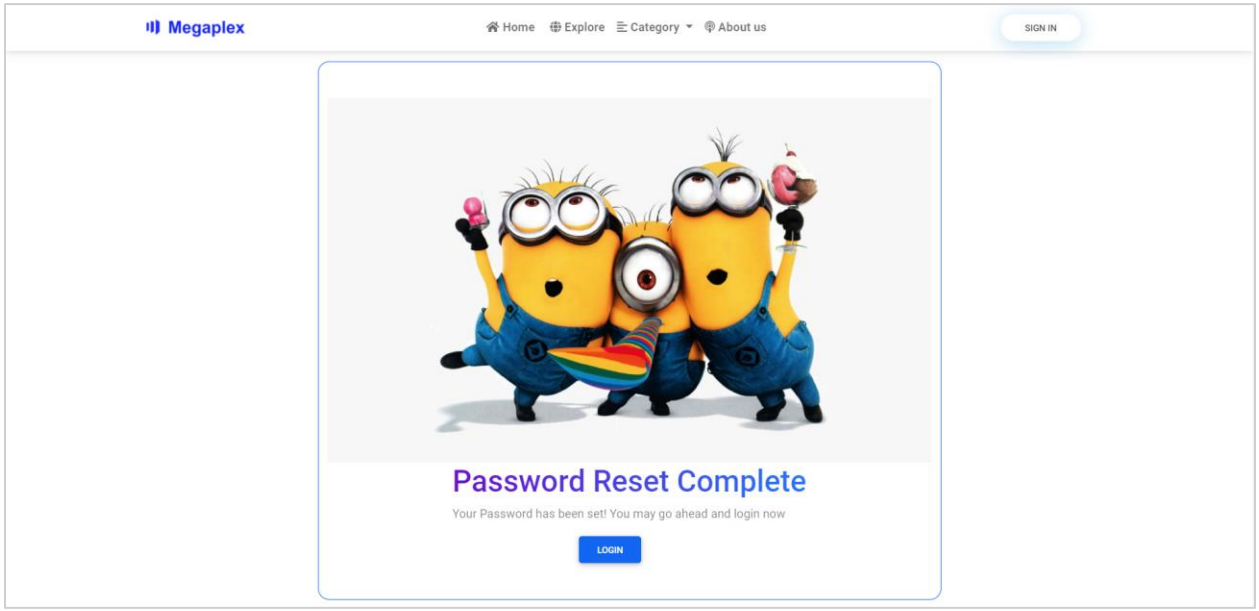



*Figure 67: Password reset conformation*

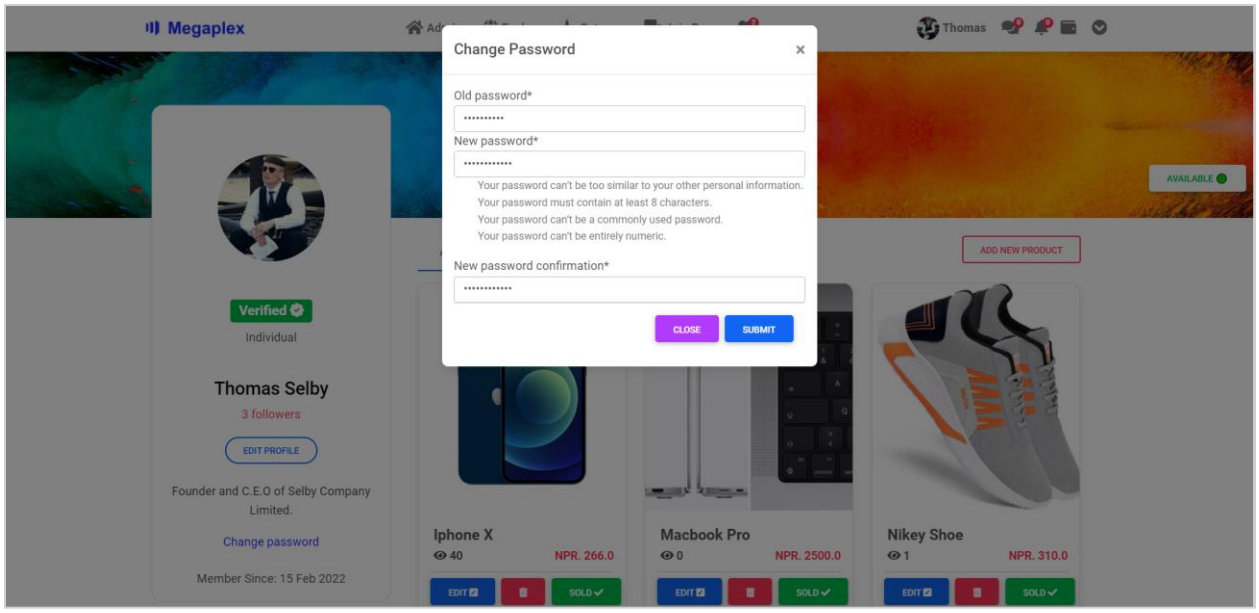



*Figure 68: Password change form*

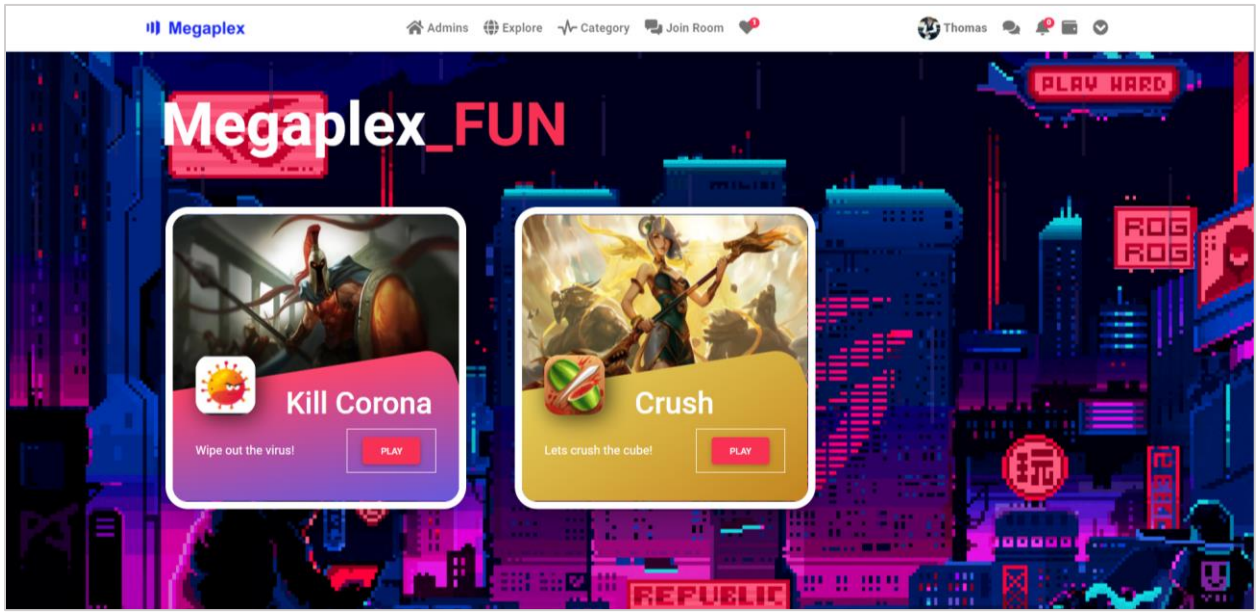



*Figure 69: Game homepage*

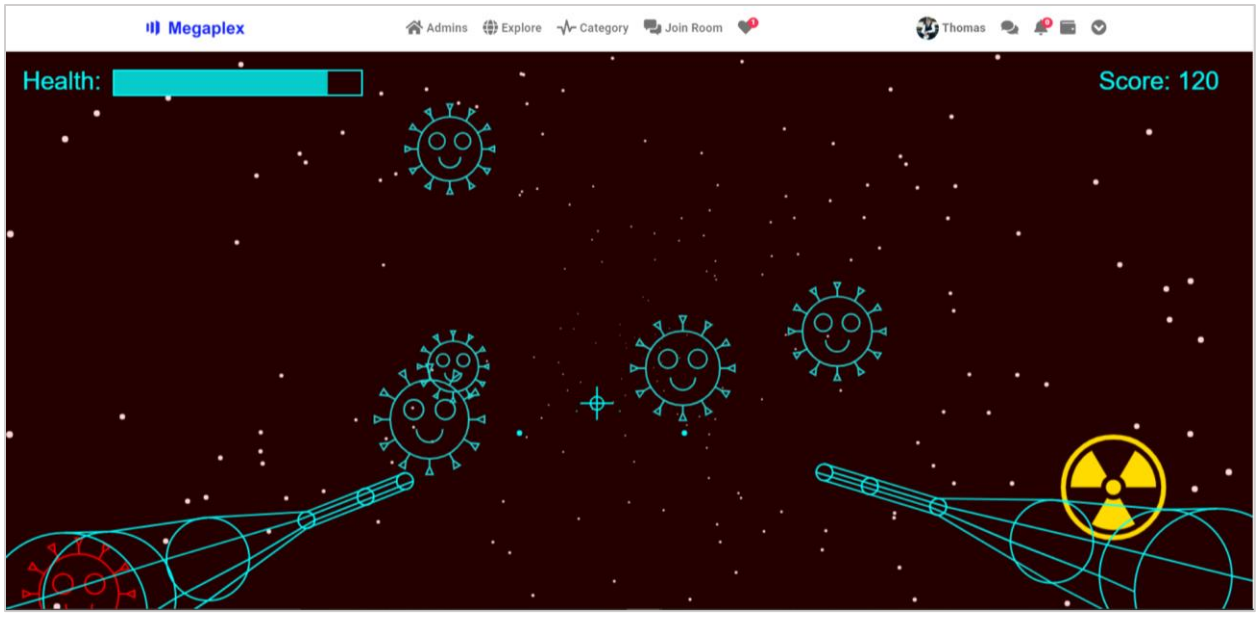



*Figure 70: Game one*

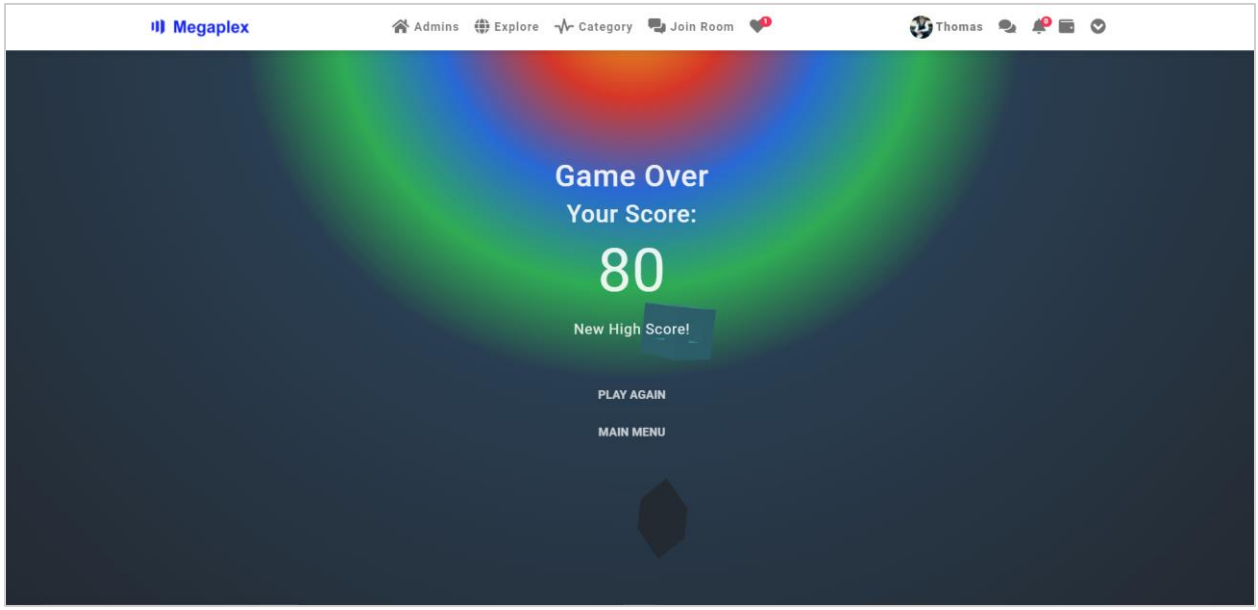



*Figure 71: Game two*

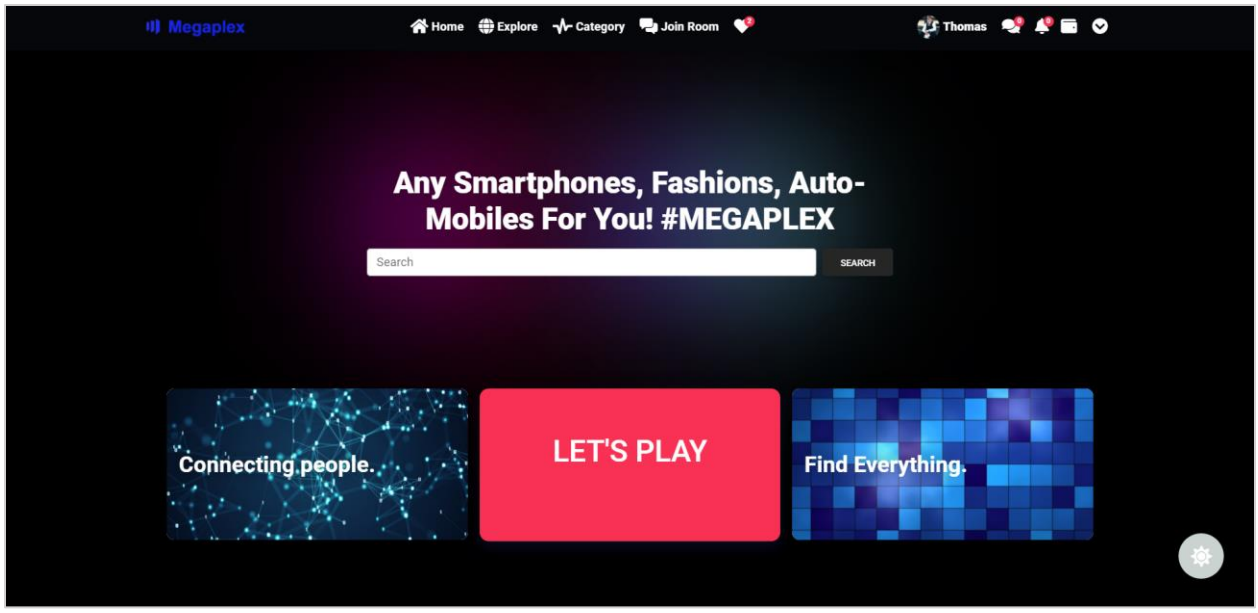



*Figure 72: Dark mode homepage*

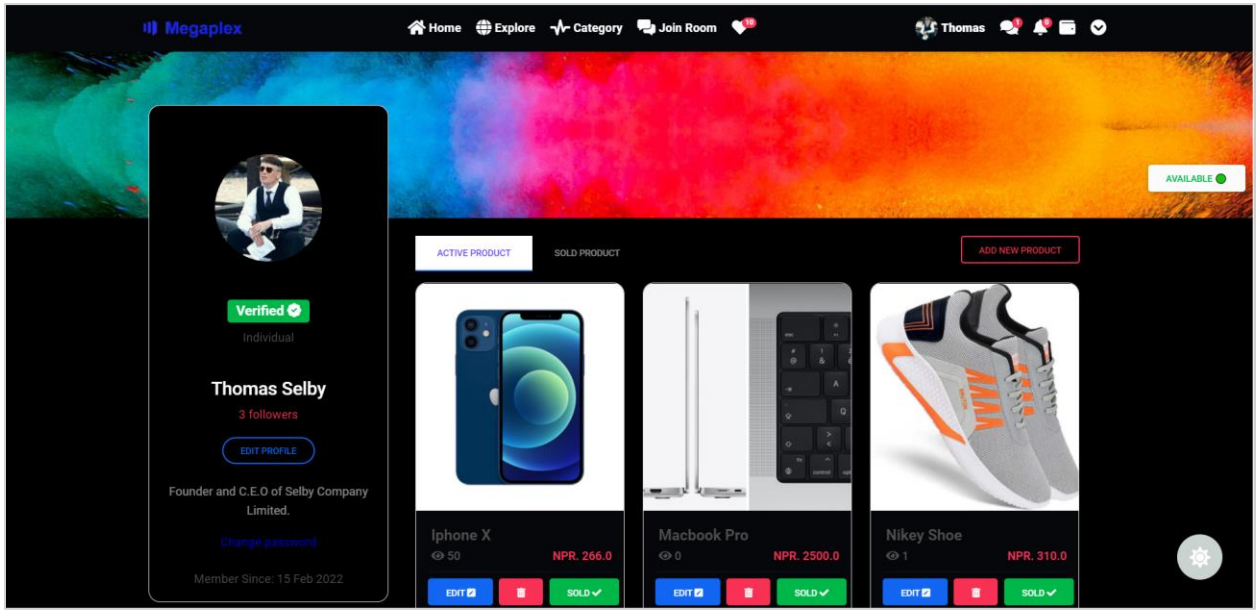



*Figure 73: Dark mode profile*

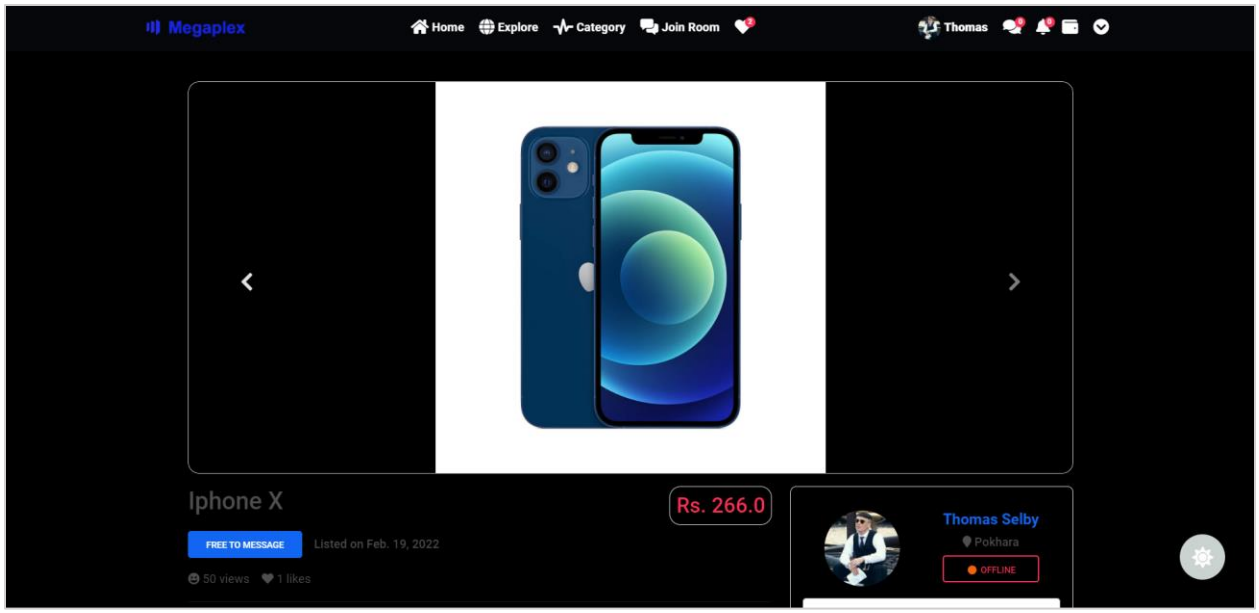



*Figure 74: Dark mode product details*

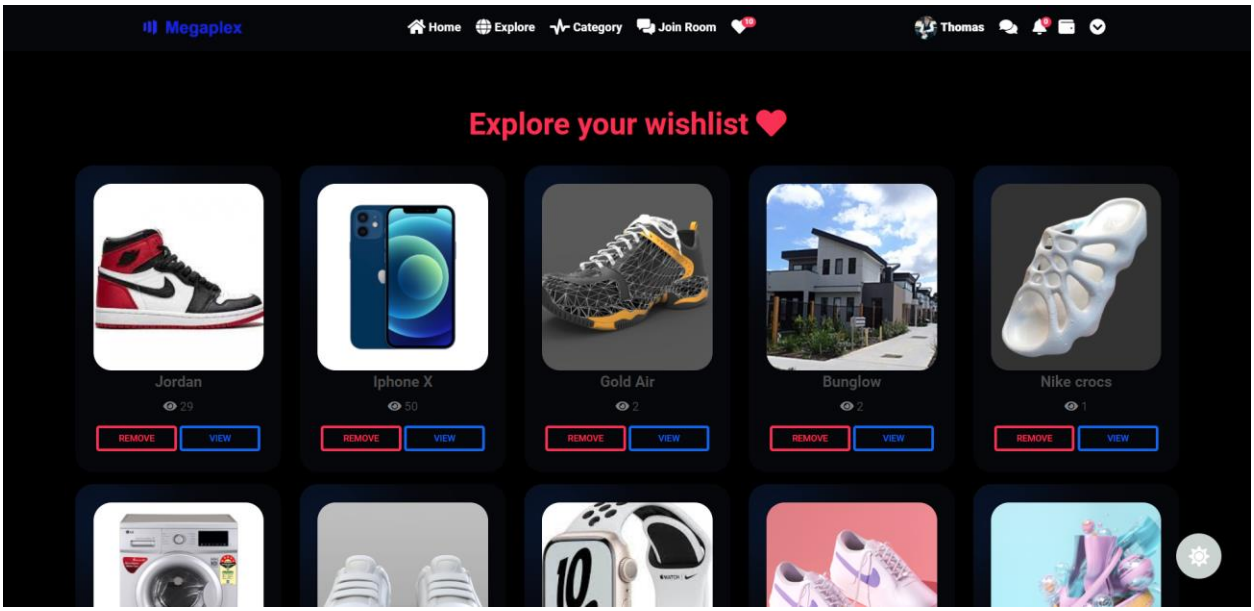



*Figure 75:Dark mode Wishlist*

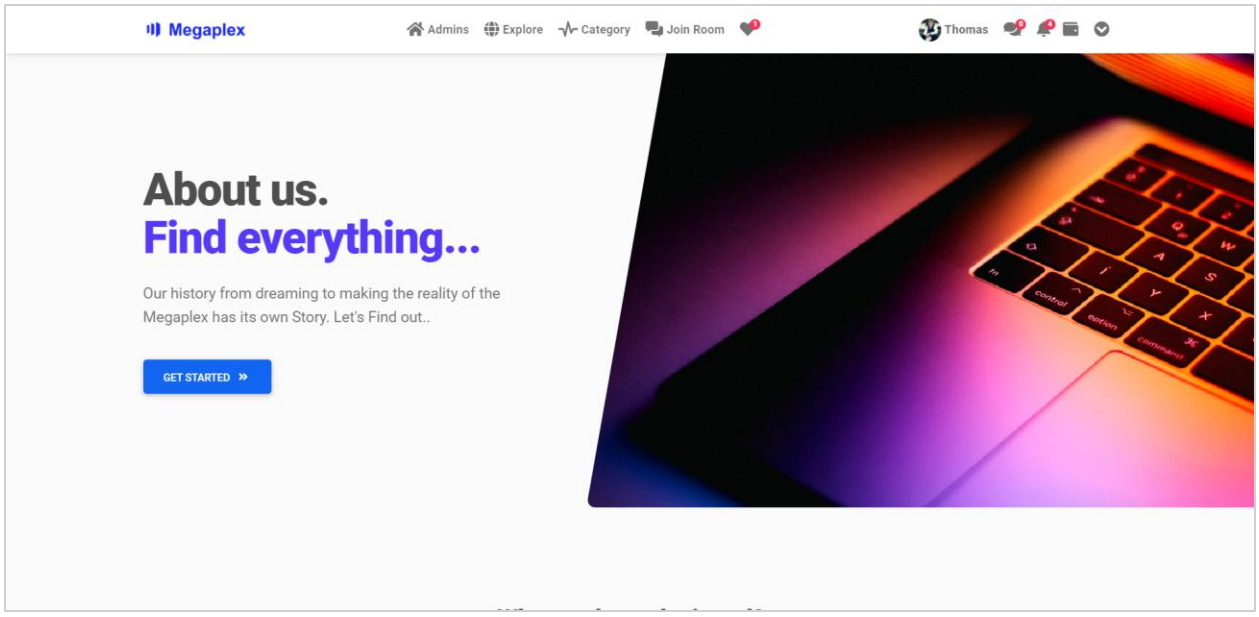



*Figure 76: About us*

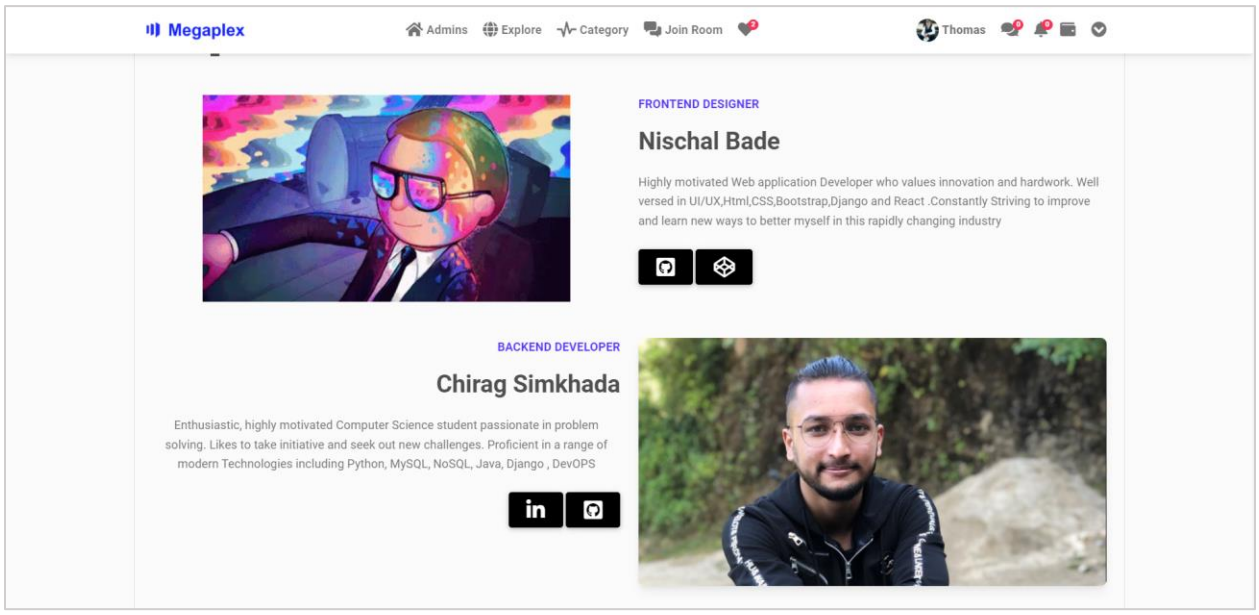



*Figure 77: About Us page 1*

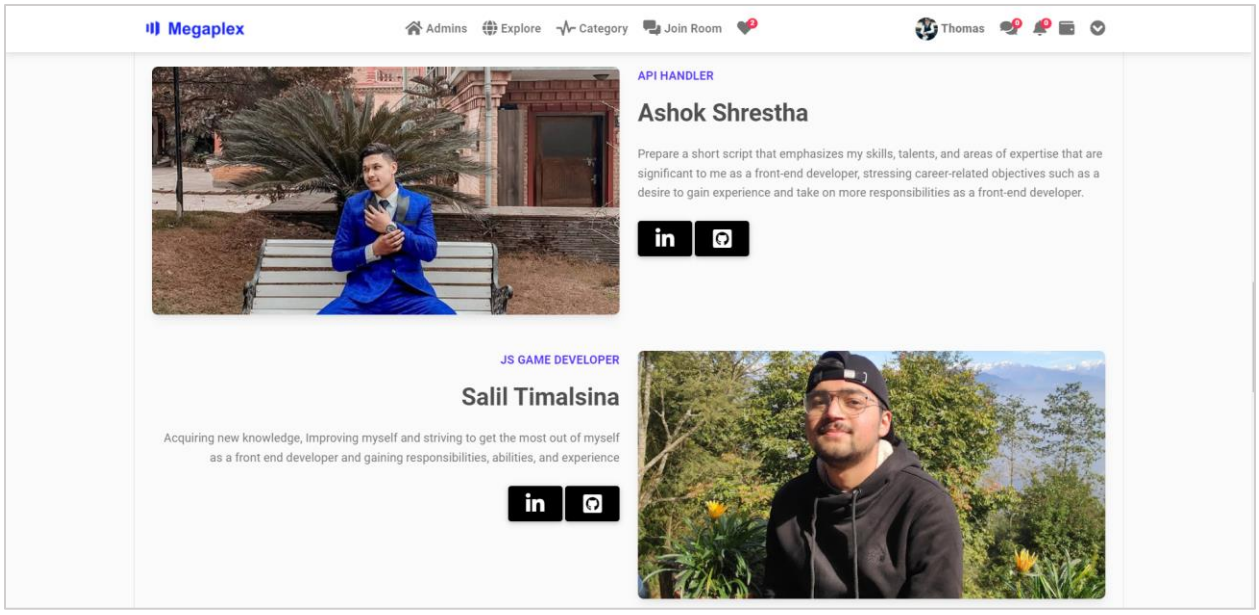



*Figure 78: About Us page 2*

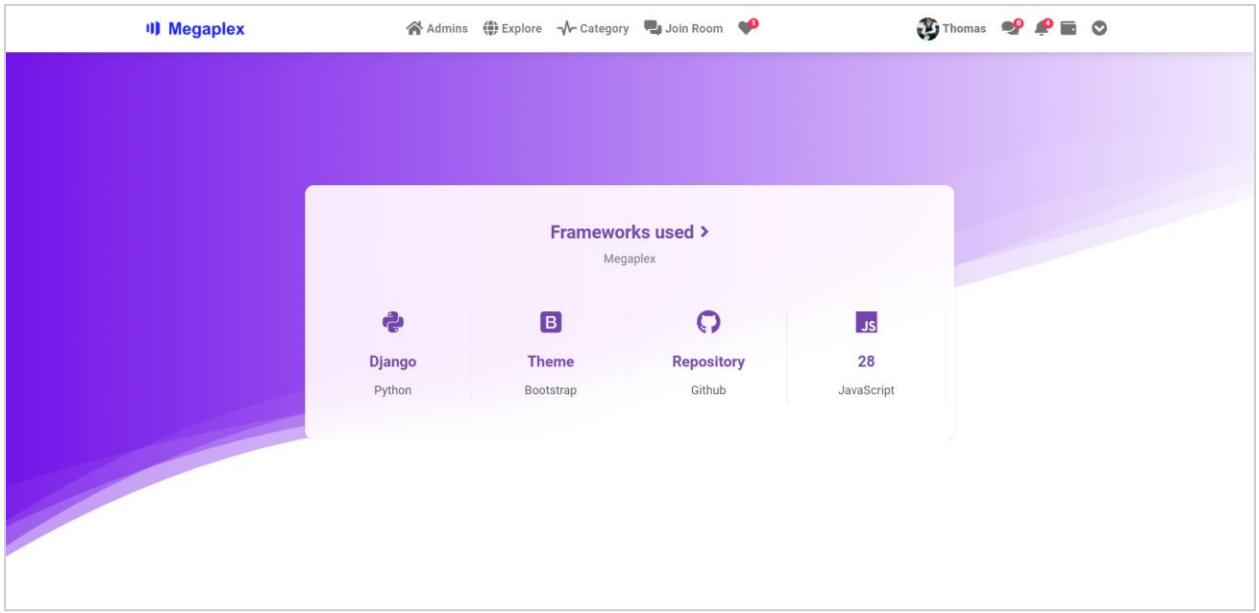



*Figure 79: Frameworks used to develop Megaplex*

# Limitations of Megaplex

### Quality control is less restrictive


*Figure 80: Quality Control*

Megaplex is a customer-to-customer (C2C) and business-to-business (B2B) platform that does not manufacture or sell goods, thus we may not be capable of monitoring the quality of the goods on our sites.

### Difficulty in external payment methods

The absence of external payment solutions at Megaplex necessitates the use of the company's own secure payment service, which means that if customers want external payment systems such as credit or debit card, PayPal, or bank card, they might face the problems.


*Figure 81: Payment*

**Fraudulence is a possibility**

Megaplex may have fraudsters seeking to defraud both customers and sellers without following the rules of the firm. Buyers should be aware of sellers that want unusual payment methods or who refuse to provide thorough information about their listing. Before moving their goods, sellers should always get complete payment.

*Figure 82: Possible Frauds*

**High levels of competitiveness**

Multiple vendors with a similar product category will always exist. Different retailers give discounts and freebies to their customers to promote sales and profitability. This makes establishing a presence in the market considerably more challenging for new enterprises or start-ups. Small enterprises are unable to scale up their revenues due to increased competition.

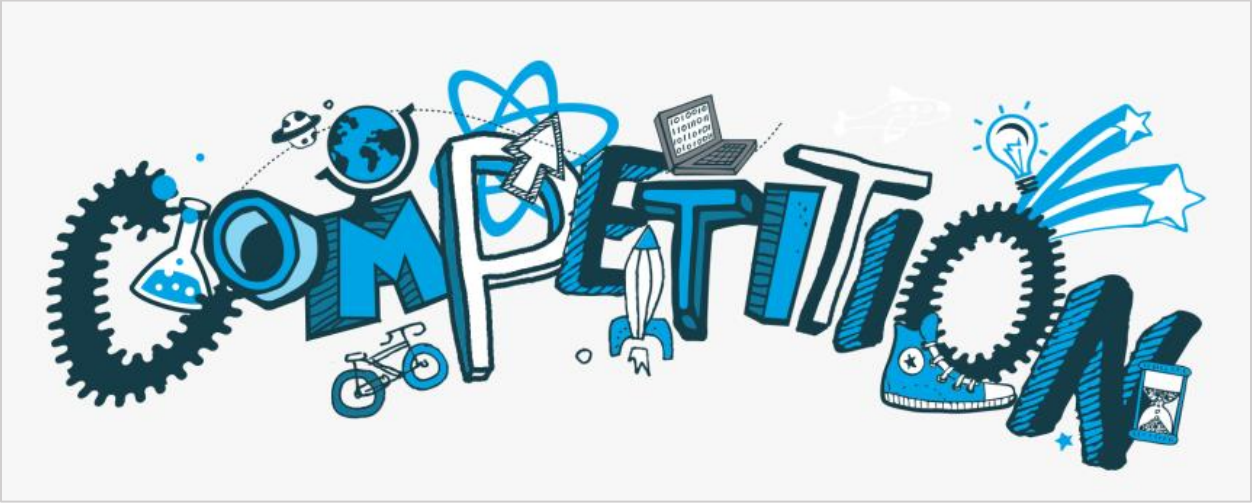


*Figure 83: High Competitions*

## Future Works:

- Integrating an advanced filtering system will provide users with more control over product browsing
- Notification System can be upgraded to include notification message editing feature in admin panel
- Proper systems to report and remove an abusive comment can be added in future releases
- Room chat history can be preserved in the database and be shared with the members of respective chats
- Audio call and voice message system between buyer and seller.
- Machine Learning for listing and searching the products, while user's search the product system will show similar products lies in that searching category.

## Conclusion:

For a country like Nepal where e-commerce is booming, it is profitable for an organization to provide alternatives to the current primary marketplace. With this vision, Megaplex was developed to reduce monopoly by current primary marketplaces in Nepal. In addition, normal people will be able to sell their products or used products. This, in turn, helps to reduce digital waste and promote local talent. Although a normal person may not be able to sell his/her product through distributors, they will get a proper platform to showcase their products. In conclusion, Megaplex will help Nepal to reduce its digital waste and promote local talent.

## References:


Aryal, M. (2021) *Who Bought The Online Platform Hamrobazar.Com Sales?* [online] available from <https://ictframe.com/online-platform-hamrobazar-com-sales/> [8 February 2022]

Hsiao, A. (2019) *Tap Into These Ebay Features For Better Shopping And Selling* [online] available from <https://www.thebalancesmb.com/ten-ebay-website-features-you-may-have-missed-1140587> [8 February 2022]

Anon (2022) *About Quikr* [online] available from <https://www.quikr.com/html/about.php> [8 February 2022]

Anon (2020) *Positive & Negative Reviews: Coutloot|Local Online Shopping - By Coutloot - Shopping Category - 7 Similar Apps, 6 Review Highlights & 26,244 Reviews - Appgrooves: Save Money On Android & Iphone Apps* [online] available from <https://appgrooves.com/app/coutloot-click-sell-chat-buy-by-jasmeet-thind/negative> [8 February 2022]

Raut, J. (2019) *UX Case Study Of Coutloot: C2C (Buy & Sell) Platform* [online] available from <https://medium.muz.li/ux-case-study-of-coutloot-c2c-buy-sell-platform-fcb4370a524> [8 February 2022]

*PESTEL Analysis* (n.d.) available from <https://corporatefinanceinstitute.com/resources/knowledge/strategy/pestel-analysis/> [24 February 2022]

*Selling on Amazon vs an Ecommerce Website: Pros & Cons* (2020) available from <https://www.vaimo.com/pros-cons-amazon-ecommerce/> [24 February 2022]

# Appendix

**Video:** https://www.youtube.com/watch?v=jTa4f1fDHHE

**GitHub Link**: https://github.com/HMV2/Megaplex

## Important Source Code:

## Megaplex

Megaplex > settings.py > ...

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Megaplex.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'homepage.custom_context_processors.notifications',
            ],
        },
    },
]

WSGI_APPLICATION = 'Megaplex.wsgi.application'
ASGI_APPLICATION = 'Megaplex.asgi.application'


CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("localhost", 6379)],
        },
    },
}
```

*Figure 84: important code of megaplex*

Megaplex > settings.py > ...

```
35  INSTALLED_APPS = [
36      'django.contrib.admin',
37      'django.contrib.auth',
38      'django.contrib.contenttypes',
39      'django.contrib.sessions',
40      'django.contrib.messages',
41      'django.contrib.staticfiles',
42      'account.apps.AccountConfig',
43      'homepage.apps.HomepageConfig',
44      'crispy_forms',
45      'dashboard',
46      'directChat',
47      'admins',
48      'product',
49      'channels',
50      'notification',
51      'roomChat',
52      'django_celery_results',
53      'django_celery_beat',
54  ]
```

*Figure 85: Installed Apps*

Megaplex > celery.py > debug_task

```python

from __future__ import absolute_import, unicode_literals
import os

from celery import Celery
from django.conf import settings
from celery.schedules import crontab
from datetime import datetime, timedelta

# set the default Django settings module for the 'celery' program.
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Megaplex.settings')

app = Celery('Megaplex')
app.conf.enable_utc = False
app.conf.update(timezone='Asia/Kathmandu')

app.config_from_object(settings, namespace='CELERY')

app.conf.beat_schedule = {
}

#app.conf.timezone = 'UTC'

# Load task modules from all registered Django app configs.
app.autodiscover_tasks()


@app.task(bind=True)
def debug_task(self):
    print(f'Request: {self.request!r}')
```

Megaplex > urls.py > ...

```python

from django.contrib import admin
from django.urls import path
from django.conf.urls import include
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('homepage.urls')),
    path('account/',include('account.urls')),
    path('chat/',include('directChat.urls')),
    path('dashboard/',include('dashboard.urls')),
    path('profile/', include('dashboard.urls')),
    path('admins/', include('admins.urls')),
    path('product/',include('product.urls')),
    path('profile/', include('dashboard.urls')),
    path('room/', include('roomChat.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

templates > layout.html > html > head > script

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    {% block title %} {{ section.title }} {% endblock title %}
    {% block css_files %} {% endblock css_files %}

    <!-- ======================bootstrap-cdn================================= -->
    <!-- Font Awesome -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css" rel="stylesheet" />
    <!-- Google Fonts -->
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" rel="stylesheet" />
    <!-- MDB -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.css" rel="stylesheet" />
    <!-- ==========================fontawesome-cdn================================= -->
    <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
        integrity="sha384-AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGroGkvsg+p" crossorigin="anonymous" />
    <link rel="stylesheet" href="{% static 'css/homepage.css' %}">
    <link rel="stylesheet" href="{% static 'css/navbar.css' %}">
    <link rel="stylesheet" href="{% static 'css/main.css' %}">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi
</head>

<body>
    {% include 'navbar.html' %}
    <div class="main">{% block main_content %} {% endblock %}</div>
    {% block games %} {% endblock %}
    {% block js_files %} {% endblock js_files %}
    {% include 'footer.html' %}
    <script src="{% static 'javascript/theme-toggle.js'%}"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymou
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.js"></script>
    <script src="{% static 'javascript/onscroll.js' %}"></script>

```

*Figure 86: Code for Layouts*

templates > admin_layout.html > html > body > script > "icon"

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    {% block title %} {{ section.title }} {% endblock title %}
    {% block css_files %} {% endblock css_files %}
    <!-- ======================bootstrap-cdn================================= -->
    <!-- Font Awesome -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css" rel="stylesheet" />
    <!-- Google Fonts -->
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" rel="stylesheet" />
    <!-- MDB -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.css" rel="stylesheet" />
    <!-- ============================fontawesome-cdn================================== -->
    <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
        integrity="sha384-AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGroGkvsg+p" crossorigin="anonymous" />
    <link rel="stylesheet" href="{% static 'css/homepage.css' %}">
    <link rel="stylesheet" href="{% static 'css/navbar.css' %}">
    <link rel="stylesheet" href="{% static 'css/main.css' %}">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
      <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js" integrity="sha256-u7e5khyithlIdTpu22PHhENmPcRdFiHRjhAuHcs05RI=" crossorigin="
</head>

<body>
    {% include 'admin_navbar.html' %}
    <div class="main">{% block main_content %} {% endblock %}</div>
    {% block js_files %} {% endblock js_files %}
    {% include 'footer.html' %}
    <!-- MDB -->
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.js"></script>
    <script src="{% static 'javascript/theme-toggle.js' %}"></script>
    <script src="{% static 'javascript/onscroll.js' %}"></script>
    <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
    {% if messages %}
        {% for m in messages %}
            {% if m.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
            <script>
                Swal.fire({
```

templates > navbar.html > nav.navbar.navbar-expand-lg.navbar-light.sticky-top. > div.container > div#navbarButtonsExample.collapse.navbar-collapse > ul.navbar-nav.flex-row.align-items-

```
{% load static %}

<nav class="navbar navbar-expand-lg navbar-light sticky-top  " style="background-color: #fff;" >
    <div class="container">
        <a class="navbar-brand" href="/" style="width: 10%">
            <img src="{% static 'images/logo1.PNG' %}" alt="" class="" style="width: min(140px,15vw);" />
        </a>
        <button class="navbar-toggler" type="button" data-mdb-toggle="collapse" data-mdb-target="#navbarButtonsExample"
            aria-controls="navbarButtonsExample" aria-expanded="false" aria-label="Toggle navigation">
            <i class="fas fa-bars"></i>
        </button>
        <div class="collapse navbar-collapse" id="navbarButtonsExample">
            <ul class="navbar-nav mx-auto mb-2 mb-lg-0">
                {% if request.user.is_authenticated %}
                <ul class="navbar-nav flex-row d-none d-md-flex">
                    <li class="nav-item me-3 me-lg-1 active ">
                        <a class="nav-link " href="/admins/dashboard">
                            <span><i class="fas fa-home fa-lg"></i></span>
                            Admins
                        </a>
                    </li>

                    <li class="nav-item me-3 me-lg-1">
                        <a class="nav-link " href="/product/explore">
                            <span><i class="fas fa-globe fa-lg"></i></span>
                            Explore
                        </a>
                    </li>

                    <li class="nav-item me-3 me-lg-1">
                        <a class="nav-link " href="/category">
                            <span><i class="fas fa-heart-rate fa-lg"></i></span>
                            Category
                        </a>
                    </li>
```

*Figure 87: Accounts*

account > forms.py > PartialProfileForm > Meta

```python
from django.core import validators
from .models import Profile
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserChangeForm, UserCreationForm
from django import forms
from django.forms import ModelForm

USER_CHOICES = [
    ("Individual","Individual"),
    ("Organization", "Organization")
    ]

class UserForm(UserCreationForm):
    """Form for creating Users"""
    email = forms.EmailField()
    plan = forms.CharField(label="Choose the type of membership", widget=forms.Select(choices=USER_CHOICES))
    class Meta:
        model = User
        fields = ['first_name', 'last_name','email','username', "plan",'password1','password2']


class LoginForm(forms.Form):
    """Form to allow users to log in"""
    username = forms.CharField(validators=[validators.MinLengthValidator(3)])
    password = forms.CharField(widget=forms.PasswordInput)


class ProfileForm(ModelForm):
    """Form for creating User profile"""
    desc = forms.CharField(
        label='desc',
        widget=forms.Textarea(attrs={
            'rows':'3',
            'placeholder':''
        })
    )
    class Meta:
        model = Profile
        fields = "__all__"
        exclude = ['user', 'email', 'username']
```

*Figure 88: Forms*

account > views.py > logout_user

```python
from django.contrib.auth import logout, login, authenticate
from django.shortcuts import redirect, render
from .forms import ProfileForm, UserForm
from django.contrib import messages, auth
from .models import Profile
from django.contrib import messages
from django.contrib.auth.models import User
from directChat.views import get_unread


def login(request):
    if request.method == 'POST':
        uname = request.POST['username']
        passwd = request.POST['password']

        user = auth.authenticate(username=uname, password=passwd)

        if user is not None:
            if not user.is_staff:
                auth.login(request, user)
                messages.success(request,"Welcome to Megaplex World")
                return redirect("/")

            elif user.is_staff:
                auth.login(request, user)
                return redirect('/admin')

        else:
            messages.add_message(request, messages.ERROR, "Invalid Username and Password!")
            return render(request, 'account/login.html',{'room_name':"broadcast",'get_unread':get_unread(request)})

    else:
        return render(request, 'account/login.html',{'room_name':"broadcast",'get_unread':get_unread(request)})


def register(request):
    form = UserForm()
    if request.method == "POST":
        form = UserForm(request.POST)
        if form.is_valid():
```

```python
class Profile(models.Model): #Model to create profile for users
    user = models.OneToOneField(User,null=True,on_delete=models.CASCADE,related_name='profile')
    username = models.CharField(max_length=20)
    firstname = models.CharField(max_length=50)
    lastname = models.CharField(max_length=50)
    email = models.EmailField()
    phone = models.CharField(max_length=10)
    plan = models.CharField(max_length=20)
    district = models.CharField(max_length=30, default="Kathmandu")
    city = models.CharField(max_length=30, default="Baghbazar")
    profile_pic = models.FileField(upload_to='profiles', default='default.png')
    created_date = models.DateTimeField(auto_now_add=True)
    followers = models.ManyToManyField(User,blank=True,related_name='followers',default=[0])
    following = models.ManyToManyField(User,blank=True ,related_name ='following',default=[0])
    cover_pic = models.FileField(upload_to='covers', default='cover.jpg', blank=True, null=True)
    verified = models.BooleanField(default=False)
    desc = models.TextField(max_length=10000, blank=True, null=True)
    product_sold = models.IntegerField(default=0, blank=True, null=True)
    active = models.BooleanField(default=True, blank=True, null=True)
    balance = models.DecimalField(
        default=50,
        max_digits=12,
        decimal_places=2
    )
    def __str__(self):
        return self.firstname + " "+ self.lastname

def txn():
    return str(random.randint(1000000, 9999999))

class transaction(models.Model):
    txn_id = models.IntegerField(default=txn)
    # sender = models.CharField(max_length=50)
    sender = models.ForeignKey(User, on_delete=models.DO_NOTHING, related_name="sender")
    # receiver = models.CharField(max_length=50)
    receiver = models.ForeignKey(User, on_delete=models.DO_NOTHING, related_name="reciver")
    amount = models.DecimalField(default=0,max_digits=11,decimal_places=2)
```

account > urls.py > ...

```python
from django.contrib.auth import login
from django.contrib.auth import views as auth_views
from django.urls import path
from . import views

#Account's URL
urlpatterns = [
    path('register/',views.register, name="register_user"),
    path('login/', views.login, name="login_user"),
    path('logout/', views.logout_user),
    path('password_reset/', auth_views.PasswordResetView.as_view(template_name="account/password_reset_form.html"), name="password_rese
    path('password_reset_sent/', auth_views.PasswordResetDoneView.as_view(template_name="account/password_reset_sent.html"), name="pass
    path('reset/<uidb64>/<token>/', auth_views.PasswordResetConfirmView.as_view(template_name="account/password_reset_confirm.html"), n
    path('reset/done', auth_views.PasswordResetCompleteView.as_view(template_name="account/password_reset_complete.html"), name="passwo
]
```

account > templates > account > login.html > ...

```
{% extends 'layout.html' %}
{% load static %}

{% block title %}<title>Login to Megaplex</title>{% endblock title %}

{% block main_content %}

    <div class="container col-xl-10 col-xxl-8 px-4 py-5">
        <div class="row align-items-center g-lg-5 py-5">
            <div class="col-lg-7 text-center text-lg-start">
                <h2 class="fs-1 lh-1 mb-3">Login to</h2>
                <h1 class="display-4 fw-bold lh-1 mb-3 "><span class="M-logo">Megaplex World!</span> </h1>
                <img class="w-50" src="{% static 'images/login_sec.png' %}" alt="">
            </div>
            <div class="col-md-10 mx-auto col-lg-5 " >
                <form class="p-4 p-md-5 shadow bg-white" action="#" method="POST" style="border-radius: 15px">
                    {% csrf_token %}
                    <img src="{% static 'images/logo.PNG' %}" alt="" class="w-50 mb-3 text-center"/>
                    <div class="form-floating mb-3">
                        <input type="text" class="form-control" id="floatingInput" placeholder="Username" name="username">
                        <label for="floatingInput"> Username</label>
                    </div>
                    <div class="form-floating mb-3">
                        <input type="password" class="form-control" id="floatingPassword" placeholder="Password" name="password">
                        <label for="floatingPassword">Password</label>
                        <a href="/account/password_reset/">Forget your passcode?</a>
                    </div>
                    <div class="checkbox mb-3">
                        <label>
                            <input type="checkbox" value="remember-me"> Remember me
                        </label>
                    </div>
                    <button class="w-100 btn btn-primary" type="submit">Login</button>
                    <hr class="my-4">
                    <small class="text-muted">Dont have account bro?</small> <a href="/account/register">Create here</a>
                </form>
            </div>
        </div>
    </div>

{% endblock main_content %}
```

dashboard > views.py > User_Profile

```
from .forms import ProductForm, txnForm
from django.contrib import messages
from django.contrib.auth import update_session_auth_hash
from directChat.views import get_unread
from django.core.paginator import Paginator
from django.contrib.auth.models import User
from django.db.models import Q
from notification.models import BroadcastNotification

# Create your views here.
@login_required
def profile(request):
    user = Profile.objects.get(user=request.user)
    form = ProfileForm(instance=user)

    pform = PasswordChangeForm(user=request.user)
    try:
        active_products = Product.objects.filter(seller=request.user, is_active=True)
    except:
        active_products = None
    try:
        inactive_products = Product.objects.filter(seller=request.user, is_active=False)
    except:
        inactive_products = None
    context={
        'room_name':"broadcast",
        'profile':user,
        'active_products':active_products,
        'inactive_products':inactive_products,
```

*Figure 89: Important code of dashboard passcode for Dashboard*

dashboard > views.py > ...

```python
@login_required
def AddFollowers(request,profile_id):
    logged_in_user = request.user.profile.id
    profile = Profile.objects.get(id = profile_id)
    user_profile = Profile.objects.get(id = logged_in_user)
    user_id = profile.user
    user_profile_id = user_profile.user
    profile.followers.add(user_profile_id)
    user_profile.following.add(user_id)
    return redirect('/dashboard/profile/{}'.format(profile.id))


#when user click un-follow to un-follow other user  the user will be remove and no longer be follower
#the other user will be removed  and won't be following the other user
#if is following
@login_required
def RemoveFollowers(request,profile_id):
    logged_in_user = request.user.profile.id
    profile = Profile.objects.get(id = profile_id)
    user_profile = Profile.objects.get(id = logged_in_user)
    user_id = profile.user
    user_profile_id = user_profile.user
    profile.followers.remove(user_profile_id)
    user_profile.following.remove(user_id)
    return redirect('/dashboard/profile/{}'.format(profile.id))


@login_required
def follower_list(request,profile_id):
    profile = Profile.objects.get(id = profile_id)
    user_profile = Profile.objects.filter()
    user = profile.user
    followings=profile.following.all()
    followers=profile.followers.all()
    context = {
        'user':user,
        'profile':profile,
        'user_profiles':user_profile,
        'followings':followings,
        'followers':followers,
        'room_name':"broadcast"
```

dashboard > urls.py > ...

```python
from django.contrib.auth import views as auth_views
from django.urls import path

from . import views

urlpatterns = [
    path('profile/', views.profile, name='profile'),
    path('profile/<int:profile_id>',views.User_Profile,name='user_profile'),
    path('follow_user/<int:profile_id>',views.AddFollowers,name ='follow'),
    path('unfollow_user/<int:profile_id>',views.RemoveFollowers,name='unfollow'),
    path('follower_list/<int:profile_id>',views.follower_list,name='following_list'),
    path('remove_follower/<int:follower_id>',views.Delete_Follower,name='Remove_following_user'),
    path('toggle_following/<int:following_id>',views.togglefollowing,name='toggle_following_user'),
    path('addProduct/', views.addProduct, name="addProduct"),
    path('editProduct/<int:product_id>', views.editProduct, name="editProduct"),
    path('removeProduct/<int:product_id>', views.remove_product, name="removeProduct"),
    path('wishlist/',views.wishlist,name='wishlist'),
    path('wallet/',views.wallet,name='wallet'),
    path('mark_sold/<int:product_id>',views.mark_sold),
    path('mark_unsold/<int:product_id>',views.mark_unsold),
    path('set_online/<int:id>',views.set_online),
    path('set_offline/<int:id>',views.set_offline),
    path('notification',views.user_notification)
]
```

dashboard > forms.py > ...

```python
from product.models import Product
from django.forms import ModelForm
from django import forms
from account.models import transaction

class ProductForm(ModelForm):
    """Form for creating Product"""

    description = forms.CharField(
        label='description',
        widget=forms.Textarea(attrs={
            'rows':'3',
            'placeholder':''
        })
    )

    specifications = forms.CharField(
        label='specifications',
        widget=forms.Textarea(attrs={
            'rows':'3',
            'placeholder':''
        })
    )
    class Meta:
        model = Product
        exclude = ['is_active','view_count','product_likes']

class txnForm(ModelForm):
    class Meta:
        model = transaction
        fields = '__all__'
```

rd > templates > dashboard > profile.html > div.profile-page.position-relative > div.profile.container-sm.row.m-auto.fixed > div.profile-info.d-flex.flex-column.text-center.col-lg-3.border-15.borde

```
{% extends 'layout.html' %} {% load static %} {% load crispy_forms_tags %} {% block title %}
<title>{{user.first_name}}'s Profile</title>
{% endblock title %} {% block css_files %}

<link rel="stylesheet" href="{% static '/css/main.css' %}" />
<link rel="stylesheet" href="{% static '/css/homepage.css' %}" />
<link rel="stylesheet" href="{% static '/css/profile.css' %}" />

{% endblock css_files %} {% block main_content %}

<div class="profile-page position-relative">
    <div class="cover position-relative">
        <img src="https://wallpapercave.com/wp/wp6941067.jpg" alt="" class="cover-img" />
        <div class="edit-profile-cover d-flex position-absolute theme-color gap-2">
            {% if user.profile.active %}
            <button type="button" class="btn btn-light text-success" data-mdb-ripple-color="dark"><a class="unlink"
                    href="/dashboard/set_offline/{{user.id}}">Available 🟢</a></button>
            {% else %}
            <button type="button" class="btn btn-light text-danger" data-mdb-ripple-color="dark"> <a class="unlink"
                    href="/dashboard/set_online/{{user.id}}"> Offline ❌ </a> </button>
            {% endif %}
        </div>
    </div>

    <div class="profile container-sm row m-auto fixed">
        <div class="profile-info d-flex flex-column text-center col-lg-3 border-15 border p-3 bg_fix ">

            <span class="m-auto w-50"><img class="rounded-circle w-100 h-100  " src="{{profile.profile_pic.url}}"
                    style="object-fit: cover; aspect-ratio: 1/1;" alt="" /></span>
            {% if user.profile.verified %}
            <h4><span class="badge bg-success">Verified <i class="fa fa-badge-check"></i></span></h4>
            {% else %}
            <h4><span class="badge bg-danger">Unverified <i class="fa fa-badge-cross"></i></span></h4>
            {% endif %}
            <p>{{profile.plan}}</p>
            <h4 class="pt-4 theme-color">{{profile.firstname}} {{profile.lastname}}</h4>
            <a href="/dashboard/follower_list/{{profile.id}}" class="text-danger">{{profile.followers.all.count}}
                followers</a>

            <div class=" mt-3">
                {% if request.user == user %}
                <a data-toggle="modal" data-target="#exampleModal" class="text-decoration-none"><button
```

```
nd-9.col.col-lg-8.col-xl-6.col-xxl-5.theme-bg.p-5.position-relative.bg-opacity-50.rounded-9.overflow-hidden.min-vh-75. > form.p-md-0.pe-md-2 > div#ex2-content.tab-content > div#ex2-tabs-1.tab-pane.
30  {% block main_content %}
31  <section class="container-fluid bg-primary  bg-opacity-10 pt-5 pb-5">
32      <div class=" mx-auto col-sm-10 col-md-9  col col-lg-8 col-xl-6 col-xxl-5 theme-bg p-5 position-relative bg-opacity-50  rounded-9 overflow-hidden m
33          <h1 class=" text-center display-6 fw-bolder text-white p-4">Add Product</h1>
34          <span class="w-50 h-25 bg-primary position-absolute rounded-circle top-0  start-0 blur" style="z-index: -1;"></span>
35          <span class="w-25 h-25 bg-warning position-absolute rounded-circle top-0  end-0 blur" style="z-index: -1;"></span>
36          <span class="w-50 h-25 bg-danger position-absolute rounded-circle top-0  start-50 blur" style="z-index: -1;"></span>
37          <span class=" position-absolute rounded-circle top-0  end-0 w-25 h-25 " style="z-index: -1;"> <img src="https://cdn3d.iconscout.com/3d/premium
38          <form action="" class="p-md-0 pe-md-2" method="POST" enctype="multipart/form-data">
39              {% csrf_token %}
40
41              <!-- Tabs navs -->
42              <ul class="nav nav-tabs nav-fill mb-3" id="ex1" role="tablist">
43                  <li class="nav-item " role="presentation">
44                      <a class="nav-link active" id="ex2-tab-1" data-mdb-toggle="tab" href="#ex2-tabs-1" role="tab"
45                          aria-controls="ex2-tabs-1" aria-selected="true">Basic Info</a>
46                  </li>
47                  <li class="nav-item" role="presentation">
48                      <a class="nav-link" id="ex2-tab-2" data-mdb-toggle="tab" href="#ex2-tabs-2" role="tab"
49                          aria-controls="ex2-tabs-2" aria-selected="false">Images</a>
50                  </li>
51                  <li class="nav-item" role="presentation">
52                      <a class="nav-link" id="ex2-tab-3" data-mdb-toggle="tab" href="#ex2-tabs-3" role="tab"
53                          aria-controls="ex2-tabs-3" aria-selected="false">Description</a>
54                  </li>
55              </ul>
56              <!-- Tabs navs -->
57
58              <!-- Tabs content -->
59              <div class="tab-content" id="ex2-content">
60                  <div class="tab-pane fade show active pb-4" id="ex2-tabs-1" role="tabpanel" aria-labelledby="ex2-tab-1">
61                      <div class="row row-cols-2 mt-1" >
62                          <div class="form-group ">
63                              {{ form.name | as_crispy_field }}
64                          </div>
65                          <div class="form-group  ">
66                              {{ form.brand | as_crispy_field }}
67                          </div>
68                      </div>
69                      <div class="row row-cols-2 mt-1" >
70                          <div class="form-group ">
```

```
directChat > urls.py > ...
1  from django.urls import path
2  from directChat.views import Inbox, Directs, SendDirect
3  urlpatterns = [
4      path('', Inbox, name='inbox'),
5      path('directs/<username>',Directs, name='directs'),
6      path('send/',SendDirect, name='send_direct'),
7  ]
```

*Figure 90: Code for Direct Chat*

directChat > views.py > ...

```python
from django.http.response import HttpResponse, HttpResponseBadRequest
from django.shortcuts import redirect
from directChat.models import Chat_Message
from django.contrib.auth.decorators import login_required
from django.template import loader
from django.contrib.auth.models import User
from django.urls import reverse

# @login_required
def Inbox(request):
    messages = Chat_Message.get_messages(user=request.user)
    active_direct = None
    directs = None
    if messages:
        message = messages[0]
        active_direct = message['user'].username
        directs = Chat_Message.objects.filter(user=request.user, recipient=message['user'])
        directs.update(is_read=True)
        for message in messages:
            if message['user'].username == active_direct:
                message['unread'] = 0
    context = {
        'room_name':"broadcast",
        'directs': directs,
        'messages': messages,
        'active_direct': active_direct,
        'get_unread':get_unread(request)
    }
    template = loader.get_template('directChat/direct.html')

    return HttpResponse(template.render(context, request))


def get_name(username):
    try:
        user =  User.objects.get(id=username)
    except:
        user = User.objects.get(username=username)
    full_name = user.first_name + " "+ user.last_name
    return full_name
```

```python
@login_required
def Directs(request, username):
    user = request.user.id
    current_user = Chat_Message.objects.all()
    messages = Chat_Message.get_messages(user=user)
    active_direct = username
    try:
        directs = Chat_Message.objects.filter(user=user, recipient=username)
    except:
        directs = Chat_Message.objects.filter(user=user, recipient=user)

    directs.update(is_read=True)

    for message in messages:
        if message['user'].username == username:
            message['unread'] = 0
        name = get_name(username)
        context = {
            'room_name':"broadcast",
            'directs': directs,
            'messages': messages,
            'active_direct': active_direct,
            'current_user': current_user,
            'user_full':name,
            'get_unread':get_unread(request)
        }

    template = loader.get_template('directChat/direct.html')
    return HttpResponse(template.render(context, request))
```

directChat > models.py > ...

```python
from itertools import count
from django.db import models

# Create your models here.
from django.db import models
from django.contrib.auth.models import User
from django.db.models import Max

# Create your models here.
class Chat_Message(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='user')
    sender = models.ForeignKey(User, on_delete=models.CASCADE, related_name='from_user')
    recipient = models.ForeignKey(User, on_delete=models.CASCADE, related_name='to_user')
    body = models.TextField(max_length=1000, blank=True, null=True)
    date = models.DateTimeField(auto_now_add=True)
    is_read = models.BooleanField(default=False)

    def send_message(from_user, to_user, body):
        sender_message = Chat_Message(
            user=from_user,
            sender=from_user,
            recipient=to_user,
            body=body,
            is_read=True)
        sender_message.save()

        recipient_message = Chat_Message(
            user=to_user,
            sender=from_user,
            body=body,
            recipient=from_user)
        recipient_message.save()
        return sender_message

    def get_messages(user):
        messages = Chat_Message.objects.filter(user=user).values('recipient').annotate(last=Max('date')).order_by('-last')
        users = []
        for message in messages:
            users.append({
                'user': User.objects.get(pk=message['recipient']),
                'last': message['last'],
```

```
> section > div.container.py-5 > div.row > div.col-md-12 > div.card.border-15.border > div.card-body > div.row > div.col-md-6.col-lg-5.col-xl-4.mb-4.mb-md-0 > div.p-3 > div
{% extends 'layout.html' %}
{% load static %}
{% block title %}<title>Megaplex Messanger</title>{% endblock title %}
{% block main_content %}
<section>
  <div class="container py-5">
    <div class="row">
      <div class="col-md-12">
        <div class="card border-15 border">
          <div class="card-header d-flex justify-content-between align-items-center">
            <div class="container overflow-hidden">
              <div class="row">
                <div class="col-4">
                  <div class="p-3">
                    <h5 class="mb-0 fs-3 fw-bold">Megaplex Messenger</h5>
                  </div>
                </div>
                <div class="col-8">
                  {% if user_full != None %}
                  <div class="alert alert-primary mb-0" role="alert">
                    <div class="d-flex justify-content-between">
                      <h5 class="mb-0 me-3">{{user_full}}</h5>
                      <div class="d-flex">
                        <i class="fa fa-phone me-3" aria-hidden="true"></i>
                        <i class="fa fa-video me-3" aria-hidden="true"></i>
                        <i class="fa fa-share-alt" aria-hidden="true"></i>
                      </div>
                    </div>
                  </div>
                  {% endif %}
                </div>
              </div>
            </div>
          </div>
          <div class="card-body">
            <div class="row">
              <div class="col-md-6 col-lg-5 col-xl-4 mb-4 mb-md-0">
                <div class="p-3">
                  <div class="input-group rounded mb-3">
                    <input type="search" class="form-control rounded" placeholder="Search" aria-label="Search"
                      aria-describedby="search-addon" />
```

```
rectChat > direct.html > section > div.container.py-5 > div.row > div.col-md-12 > div.card.border-15.border > div.card-body > div.row > div.col-md-6.col-lg-7.col-xl
                <div class="p-3">
                  <div class="input-group rounded mb-3">
                    <input type="search" class="form-control rounded" placeholder="Search" aria-label="Search"
                      aria-describedby="search-addon" />
                    <span class="input-group-text border-0" id="search-addon">
                      <i class="fas fa-search"></i>
                    </span>
                  </div>

                  <div data-mdb-perfect-scrollbar="true" style="position: relative; height: 400px">
                    <ul class="list-unstyled mb-0">

                      {% for message in messages %}

                      <li class="p-2 border-bottom">
                        <a class="{% if active_direct == message.user.username %}is-active{% endif %}"
                          href="{% url 'directs' message.user.id %}">
                          <div class="d-flex flex-row">
                            <div class="pt-1 d-flex">
                              <p class="fw-bold mb-0">{{message.user.first_name}} {{message.user.last_name}}
                              </p>
                              {% if message.unread > 0 %}
                              <p class="badge bg-danger ms-2">{{message.unread}}</p>
                              {% endif %}
```

homepage > views.py > index_page

```python
from django import views
from django.http.response import HttpResponse
from django.shortcuts import redirect, render
from django.contrib.auth import logout
from channels.layers import get_channel_layer
from asgiref.sync import async_to_sync
from product.models import Category
from product.models import Product
from django.core.mail import send_mail
from django.conf import settings
from django.contrib import messages
from directChat.views import get_unread
from django.db import connection

def index_page(request):
    products = Product.objects.filter( is_active=True ).order_by('-view_count')[:10]
    user_list = getUserCount()
    pro_list = ""
    if len(user_list)>0:
        pro_list = [0]*len(user_list)
        ln = 0
        for i in user_list:
            pro_list[ln] = [Product.objects.filter(seller__id = i)[:4]]
            ln +=1
    context={
        'room_name':"broadcast",
        'products':products,
        'collection':pro_list,
        'get_unread':get_unread(request)
    }
    if request.method == 'POST':
        type = request.POST.get('type')
        if type=='search':
            item = request.POST.get('item')
            if item !="":
                return redirect('/product/filter/'+item)
        elif type=="contact":
            subject = request.POST.get('subject')
            email = request.POST.get('email')
            desc = request.POST.get('desc') + "\n" + "Reply Email: "+email
            test = send_mail(subject, desc, settings.EMAIL_HOST_USER, [settings.EMAIL_HOST_USER], fail_silently=False)
```

*Figure 91: Code of Homepage*

homepage > views.py > index_page

```python
def category(request):
    category = Category.objects.all()
    context={
        'products':category,
        'category':category,
        'room_name':"broadcast",
        'get_unread':get_unread(request)
    }
    return render(request, 'homepage/category.html', context)


def getUserCount():
    products = Product.objects.all()
    users = {}
    for i in products:
        if i.seller.id in users:
            users[i.seller.id] += 1
        else:
            users[i.seller.id] = 1
    users_with_valid_numbers = []
    for key in users:
        if users[key]>3:
            users_with_valid_numbers.append(key)

    return users_with_valid_numbers
```

homepage > urls.py > ...

```python
from django.contrib.auth import views as auth_views
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index_page, name='index_page'),
    path('fun/',views.fun),
    path('fun2/',views.fun2),
    path('games/',views.games),
    path('category/', views.category, name='category'),
    path('logout/', views.logout_view),
    path('about/', views.about),
]
```

age.html > section.container-fluid.search-section.position-relative.theme-color.overflow-hidden > div.d-flex.justify-content-center.gap-3 > a.w-100.align-items-center.country-card.bg-danger.btn.btn-

```
{% extends 'layout.html' %} {% load static %} {% block title %}
<title>Home</title>
{% endblock title %} {% block css_files %}
<link rel="stylesheet" href="{% static '/css/homepage.css' %}" />
<link rel="stylesheet" href="{% static '/css/main.css' %}" />

{% endblock css_files %}

{% block main_content %}
<section class="container-fluid search-section position-relative theme-color overflow-hidden">
    <div class="content-search m-auto relative">
        <p class="display-6 bolder w-100 text-center">
            Any Smartphones, Fashions, Auto-Mobiles For You! #MEGAPLEX
        </p>
        <form class="d-flex w-100 m-auto flex-column flex-md-row" method="POST">
            {% csrf_token %}
            <input type="hidden" name="type" value="search">
            <input class="form-control me-2" name="item" type="search" placeholder="Search" aria-label="Search" />
            <button type="submit" class="btn btn-dark" type="submit">Search</button>
        </form>

        <div class="circles position-absolute d-flex top-0">
            <div class="circle-one circle"></div>
            <div class="circle-two circle"></div>
            <div class="circle-three circle"></div>
            <div class="circle-four circle"></div>
        </div>
    </div>

    <div class="d-flex justify-content-center gap-3" style="margin-top: 150px;">

            <a href="/product/filter/?category={{i.id}}" class="">
                    <div class="country-card position-relative">
                        <img src="https://image.freepik.com/free-photo/abstract-futuristic-molecules-technology-lines-dots-connect-dark-blue-backg
                                                    class="img img-responsive" style="aspect-ratio: 1/0.5; object-fit: cover; object-position: cent
                        <div class=" h-100 position-absolute top-0  d-flex align-items-center country-bg"><span class="country-name ms-3">Connecti
                    </div>
            </a>
            <a class="w-100 align-items-center country-card bg-danger btn btn-danger shadow rounded-6 "
                href="/games">
                    <div class="text-white ">
```

homepage > templates > Homepage > category.html > ...

```
{% extends 'layout.html' %} {% load static %} {% block title %}
    <title>Home</title>
{% endblock title %}


{% block main_content %}
    <section id="featured-box">
        <div class="container-xl my-5">
            <div class="intro">
                <h1 class="display-4 fw-semibold py-4 fw-bold text-black-50">Browse Catalog</h1>
            </div>
            <div class="row row-cols-md-3  gx-5  gy-4 row-cols-2  row-cols-lg-4 ">
                {% for i in category %}

                <div class="card col-auto   ">
                  <div class="bg-image hover-overlay ripple" data-mdb-ripple-color="light">
                    <img src="{{i.picture.url}}" class="img-fluid"  style="object-fit: cover; aspect-ratio:1/1"/>
                    <a href="/product/filter/?category={{i.id}}">
                      <div class="mask" style="background-color: rgba(251, 251, 251, 0.15)"></div>
                    </a>
                  </div>

                  <div class="card-body ps-1">
                    <h5 class="card-title fw-bolder fs-3">{{i.name}}</h5>

                  </div>
                </div>
                {% endfor %}
            </div>
        </div>
    </section>
{% endblock main_content %}
```

notification > tasks.py > ...

```
from celery import shared_task
from channels.layers import get_channel_layer
from asgiref.sync import async_to_sync
from .models import BroadcastNotification
import json
from celery import Celery, states
from celery.exceptions import Ignore
import asyncio
@shared_task(bind = True)
def broadcast_notification(self, data):
    try:
        notification = BroadcastNotification.objects.filter(id = int(data))
        if len(notification)>0:
            notification = notification.first()
            channel_layer = get_channel_layer()
            loop = asyncio.new_event_loop()
            asyncio.set_event_loop(loop)
            loop.run_until_complete(channel_layer.group_send(
                "notification_broadcast",
                {
                    'type': 'send_notification',
                    'message': json.dumps(notification.message),
                }))
            notification.sent = True
            notification.save()
            return 'Done'

        else:
            self.update_state(
                state = 'FAILURE',
                meta = {'exe': "Not Found"}
            )

            raise Ignore()
```

*Figure 92: Code of Notification*

notification > models.py > ...

```python
import json
from tokenize import String
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver
from django_celery_beat.models import PeriodicTask, CrontabSchedule
from pyparsing import Char
# Create your models here.


class BroadcastNotification(models.Model):
    title = models.CharField(default="Megaplex Notification", max_length=120)
    message = models.TextField()
    broadcast_on = models.DateTimeField()
    sent = models.BooleanField(default=False)

    class Meta:
        ordering = ['-broadcast_on']


@receiver(post_save, sender=BroadcastNotification)
def notification_handler(sender, instance, created, **kwargs):
    # call group_send function directly to send notificatoions or you can create a dynamic task in celery beat
    if created:
        schedule, created = CrontabSchedule.objects.get_or_create(hour = instance.broadcast_on.hour, minute = instance.broadcast_on.min
        task = PeriodicTask.objects.create(crontab=schedule, name="broadcast-notification-"+str(instance.id), task="notification.tasks.
```

notification > consumers.py > ...

```python
# chat/consumers.py
import json
from channels.generic.websocket import AsyncWebsocketConsumer

class NotificationConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        self.room_name = self.scope['url_route']['kwargs']['room_name']
        self.room_group_name = 'notification_%s' % self.room_name

        # Join room group
        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )

        await self.accept()

    async def disconnect(self, close_code):
        # Leave room group
        await self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    # Receive message from room group
    async def send_notification(self, event):
        message = event['message']

        # Send message to WebSocket
        await self.send(text_data=json.dumps(message))
```

notification > routing.py > ...

```python
from django.urls import re_path
from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/notification/(?P<room_name>\w+)/$', consumers.NotificationConsumer.as_asgi()),
]
```

product > views.py > product_details

```python
from directChat.views import get_unread
from django.http import HttpResponse
from django.db import connection
from account.models import  transaction

def product_details(request,product_id):
    product = Product.objects.get(id=product_id)
    comments = Comment.objects.filter(product = product, parent=None)
    replies = Comment.objects.filter(product = product).exclude(parent=None)
    comment_count = comments.count()
    product_category = product.category
    no_of_random_items = 5
    recommendations = Product.objects.filter(category=product_category)
    possible_ids = list(recommendations.values_list('id', flat=True))
    possible_ids = random.choices(possible_ids, k=no_of_random_items)
    recommended_products = recommendations.filter(pk__in=possible_ids)
    product.view_count += 1
    product.save()
    repDict = {}
    for reply in replies:
        if reply.parent.sno not in repDict.keys():
            repDict[reply.parent.sno] = [reply]
        else:
            repDict[reply.parent.sno].append(reply)

    context={
        'room_name':"broadcast",
        'product':product,
        'comments':comments,
        'count':comment_count,
        'reply':repDict,
        'recommended_products':recommended_products,
        'get_unread':get_unread(request),
        'room_name':"broadcast",
    }
    if request.method == 'POST':
        formType = request.POST.get('formType')
        if formType == "message" and request.POST.get('body')!="":
            from_user = request.user
            to_user_username = request.POST.get('to_user')
            body = request.POST.get('body')
```

*Figure 93: Code of Product Detail app*

product > urls.py > ...

```python
from unicodedata import name
from django.contrib.auth import views as auth_views
from django.urls import path
from . import views

urlpatterns = [
    path('details/<int:product_id>', views.product_details, name='Details_page'),
    path('filter/', views.filter_page),
    path('filter/<str:item>', views.searchProduct),
    path('user/<int:user>', views.searchUserProduct),
    path('<int:product_id>/like',views.ToggleProductlike,name ='like'),
    path('<int:product_id>/removeliked',views.RemoveFromLikedList,name='removefromlikelist'),
    path('<int:comment_id>/comment_like',views.like_toggle,name ='like'),
    path('explore',views.explorepage,name='explore')
]
```

product > views.py > product_details

```python
from django.http import HttpResponse
from django.db import connection
from account.models import  transaction

def product_details(request,product_id):
    product = Product.objects.get(id=product_id)
    comments = Comment.objects.filter(product = product, parent=None)
    replies = Comment.objects.filter(product = product).exclude(parent=None)
    comment_count = comments.count()
    product_category = product.category
    no_of_random_items = 5
    recommendations = Product.objects.filter(category=product_category)
    possible_ids = list(recommendations.values_list('id', flat=True))
    possible_ids = random.choices(possible_ids, k=no_of_random_items)
    recommended_products = recommendations.filter(pk__in=possible_ids)
    product.view_count += 1
    product.save()
    repDict = {}
    for reply in replies:
        if reply.parent.sno not in repDict.keys():
            repDict[reply.parent.sno] = [reply]
        else:
            repDict[reply.parent.sno].append(reply)

    context={
        'room_name':"broadcast",
        'product':product,
        'comments':comments,
        'count':comment_count,
        'reply':repDict,
        'recommended_products':recommended_products,
        'get_unread':get_unread(request),
        'room_name':"broadcast",
    }
    if request.method == 'POST':
        formType = request.POST.get('formType')
        if formType == "message" and request.POST.get('body')!="":
            from_user = request.user
            to_user_username = request.POST.get('to_user')
            body = request.POST.get('body')
            try:
```

product > models.py > ...

```python
from django.db.models.base import Model
from django.contrib.auth.models import User
from django.utils.timezone import now

WARRANTY_CHOICES = (
    ("None", "None"),
    ("1 Year","1 Year"),
    ("2 Years","2 Years"),
    ("3 Years","3 Years"),
    ("4 Years","4 Years"),
    ("5 Years","5 Years")
)

class Category(models.Model):
    name = models.CharField(max_length=200)
    picture = models.ImageField(upload_to = 'category_images', default="def.jpeg")

    def __str__(self) -> str:
        return self.name

class Sub_Category(models.Model):
    name = models.CharField(max_length=200)
    category = models.ForeignKey(Category,on_delete=models.CASCADE)


    def __str__(self) -> str:
        return self.name

class Brand(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self) -> str:
        return self.name
```

product > templates > product > details.html > div.container.text-center. > div.row.m-5 > div > div.row.mt-3 > div.col-lg-8.text-center.text-lg-start > div.d-flex.justify-content-betwe

```
{% extends 'layout.html' %} {% load static %} {% load extras %} {% block title %}
<title>{{product.name}}</title>{% endblock title %}

{% block main_content %}

<div class="container text-center ">
    <div class="row m-5">
        <div id="carouselExampleIndicators" class="carousel slide" data-mdb-ride="carousel">
            <div class="carousel-indicators">
                <button type="button" data-mdb-target="#carouselExampleIndicators" data-mdb-slide-to="0" class="active"
                    aria-current="true" aria-label="Slide 1"></button>
                <button type="button" data-mdb-target="#carouselExampleIndicators" data-mdb-slide-to="1"
                    aria-label="Slide 2"></button>
                <button type="button" data-mdb-target="#carouselExampleIndicators" data-mdb-slide-to="2"
                    aria-label="Slide 3"></button>
            </div>
            <div class="carousel-inner border-15">
                <div class="carousel-item active border border-15 border">
                    <img src="{{product.image1.url}}" class="d-block w-100 shadow shadow-lg-3" alt="Product Image 1"
                        style="object-fit: contain; aspect-ratio: 1/0.44;" />
                </div>
                <div class="carousel-item border border-15  border">
                    <img src="{{product.image2.url}}" class="d-block w-100" alt="Product Image 2"
                        style="object-fit: contain; aspect-ratio: 1/0.44;" />
                </div>
                <div class="carousel-item border border-15  border ">
                    <img src="{{product.image3.url}}" class="d-block w-100" alt="Product Image 3"
                        style="object-fit: contain; aspect-ratio: 1/0.44;" />
                </div>
            </div>
            <button class="carousel-control-prev " type="button" data-mdb-target="#carouselExampleIndicators"
                data-mdb-slide="prev">
                <span class="carousel-control-prev-icon text-black " aria-hidden="true"></span>
                <span class="visually-hidden">Previous</span>
            </button>
            <button class="carousel-control-next" type="button" data-mdb-target="#carouselExampleIndicators"
                data-mdb-slide="next">
                <span class="carousel-control-next-icon text-black" aria-hidden="true"></span>
                <span class="visually-hidden">Next</span>
            </button>
        </div>
```

product > templates > product > filter.html > ...

```
{% extends 'layout.html' %} {% load static %} {% block title %}
<title>Filter</title>{% endblock title %}

{% block main_content %}
<div class="container pt-5">
  <div class="row">
    <div class="col-md-8 order-md-2 col-lg-9">
      <div class="container-fluid">
        <div class=" d-flex justify-content-between">
          <li class="m-2">
            {% if search_item or min_tag %}
            <h5>Search result for:</h5>
            {% endif %}
            <div class="d-flex">
              {% if search_item %}
              <p><span class="badge bg-light text-black">{{search_item}}</span></p>
              {% endif %}
              {% if min_tag %}
              <p><span class="badge bg-light text-black">Minimum Price: {{min_price1}}</span></p>
              <p><span class="badge bg-light text-black">Maximum Price: {{max_price1}}</span></p>
              {% endif %}
            </div>
          </li>
```

roomChat > views.py > ...

```python
from django.shortcuts import render
from directChat.views import get_unread

def index(request):
    return render(request, 'roomChat/chat.html')

def room(request, room_name):
    return render(request, 'roomChat/room.html', {
        'room_name1': room_name,
        'room_name':"broadcast",
        'get_unread':get_unread(request)
    })
```

*Figure 94: Code of room Chat app*

roomChat > urls.py > ...

```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('<str:room_name>/', views.room, name='room'),
]
```

roomChat > consumers.py > ...

```python
# chat/consumers.py
import json
from asgiref.sync import async_to_sync
from channels.generic.websocket import WebsocketConsumer

class ChatConsumer(WebsocketConsumer):
    def connect(self):
        self.room_name = self.scope['url_route']['kwargs']['room_name']
        self.room_group_name = 'room_%s' % self.room_name

        # Join room group
        async_to_sync(self.channel_layer.group_add)(
            self.room_group_name,
            self.channel_name
        )

        self.accept()

    def disconnect(self, close_code):
        # Leave room group
        async_to_sync(self.channel_layer.group_discard)(
            self.room_group_name,
            self.channel_name
        )
```

roomChat > templates > roomChat > chat.html > ...

```
{% extends 'layout.html' %} {% load static %} {% block title %}
<title>Room Selection </title>
{% endblock title %}


{% block main_content %}
<div
    class="m-auto d-flex flex-column container col-xl-4 col-lg-7 pb-5 my-5 px-lg-5 col-10 col-sm-8  shadow-5-soft position-relative overflow-hidde
    <img src="{% static 'images/chatbg.png' %}" class="m-auto " alt="no image found" style="width: 200px; height:200px; object-fit: contain;">
    <span class=" text-black">What chat room would you like to enter?</span>


    <select id="room-name-input" class="my-2">
        <option value="General">General</option>
        <option value="Q & A">Q & A</option>
        <option value="Marketing">Marketing</option>
        <option value="Technical">Technical</option>
    </select>
    <input id="room-name-submit" class="btn btn-primary col-lg-3" type="button" value="Enter">


    <span class="position-absolute top-0  me-5 m-auto end-0 fs-6"><i class="fas fa-circle text-primary ms-5"></i></span>
    <span class="position-absolute bottom-0  me-5 mb-5  end-0 fs-6 " style="z-index: -1;"><i
            class="fas fa-circle text-danger ms-5 mb-7  "></i></span>
    <span class="position-absolute bottom-0  fs-4 mb-2 "><i class="far fa-heart text-primary "
            style="transform: rotate(335deg);"></i></span>
    <span class="position-absolute bottom-0  end-0   me-4 fs-6 "><i class="fas fa-times text-danger "
            style="transform: rotate(75deg);"></i></span>
    <span class="position-absolute bottom-50  end-0 mb-5  me-4 fs-5 "><i class="fas fa-times text-primary mb-10"
            style="transform: rotate(75deg);"></i></span>
    <span class="position-absolute bottom-50  end-0  me-4 fs-4 fa-flip-vertical "><i
            class="far fa-square text-primary fa-rotate-by" style="transform: rotate(75deg);"></i></span>
    <span class="position-absolute bottom-0  ms-5 m-auto mb-3 "><i class="fas fa-circle text-primary ms-5 "></i></span>
</div>
```

```
{{ room_name|json_script:"room-name" }}
<script>
    const loggedUser = JSON.parse(document.getElementById('user_id').textContent);

    const room = JSON.parse(document.getElementById('room-name').textContent);

    const chatSocket = new WebSocket(
        'ws://' +
        window.location.host +
        '/ws/room/' +
        room +
        '/'
    );

    chatSocket.onmessage = function (e) {
        const data = JSON.parse(e.data);

        if (loggedUser != data.username) {
            document.querySelector('#chat-log').innerHTML +=
                "<div class='d-flex flex-row justify-content-end mb-4 pt-1'>" +
                "<img src=" + data.profile_pic + " class = 'rounded-circle' alt='avatar 1' style='width: 45px; height: 45px;'>" +
                "<p class= 'small p-2 mb-1 text-white rounded-3 bg-danger ms-1' >" + data.username + ": " +
                data.message + "</p>" +
                "</div>"
        } else {
            document.querySelector('#chat-log').innerHTML +=
                "<div class='d-flex flex-row justify-content-start mb-4 pt-1'>" +
                 "<img src=" + data.profile_pic + " class = 'rounded-circle' alt='avatar 1'  style='width: 45px; height: 45px;'>" +
                "<p class= 'small p-2 mb-1 text-white rounded-3 bg-secondary ms-1' >" + data.username + ": " +
                data.message + "</p>" +
                "</div>"
        }
    };

    chatSocket.onclose = function (e) {
        console.error('Chat socket closed unexpectedly');
    };
```

admins > views.py > send_notification

```python
from django.shortcuts import redirect
import os
from django.contrib.auth import logout
from notification.models import BroadcastNotification

# Create your views here.
def dashboard(request):
    users = User.objects.all()
    user_info = users.filter(is_staff=0)
    admin_info = users.filter(is_staff=1)
    context={
        'admin_info':admin_info,
        'user_info':user_info,
    }
    user_count = users.count()
    user_info = User.objects.filter(is_staff=0)
    admin_info = User.objects.filter(is_staff=1)
    product = Product.objects.all()
    view_count = 0
    profiles = Profile.objects.all()
    top_sellers = profiles.order_by('-product_sold')
    top_products = product.order_by('-view_count')[:5]
    active_product_count = product.filter(is_active=True).count()
    inactive_product_count = product.filter(is_active=False).count()
    for i in product:
        view_count += i.view_count

    context={
        'admins':admin_info,
        'users': user_info,
        'users_count':user_count,
        'view_count':view_count,
        'top_seller':top_sellers,
        'top_products':top_products,
        'active_product_count':active_product_count,
        'inactive_product_count':inactive_product_count
    }
    return render(request, 'admins/dashboard.html',context)
```

*Figure 95: Code for admin section*

admins > urls.py > ...

```python
from django.contrib.auth import views as auth_views
from django.urls import path

from . import views

urlpatterns = [
    path('dashboard/', views.dashboard, name='dashboard'),
    path('send_notification/',views.send_notification,name='send_notification'),
    path('add_category/',views.add_category,name='add_category'),
    path('add_subcategory/',views.add_subcategory,name='add_subcategory'),
    path('<int:category_id>/edit_category',views.edit_category,name='edit_category'),
    path('activate/<int:id>', views.active),
    path('deactivate/<int:id>', views.deactive),
    path('verify/<int:id>', views.verify),
    path('unverify/<int:id>', views.unverify),
    path('deleteSub/<int:id>', views.deleteSub),
    path('deleteCat/<int:id>', views.deleteCat),
    path('logout', views.logout_user)
]
```

*Figure 96: Admin pages' URLs*

admins > forms.py > ...

```python
from django.forms import DateTimeInput, ModelForm
from notification.models import BroadcastNotification
from product.models import Category, Sub_Category

class NotificationForm(ModelForm):
    class Meta:
        model = BroadcastNotification
        fields = ['message','broadcast_on','title']
        widgets = {
            'broadcast_on': DateTimeInput(attrs={'type': 'datetime-local'}),
        }

class CategoryForm(ModelForm):
    class Meta:
        model = Category
        fields = "__all__"

class SubcategoryForm(ModelForm):
    class Meta:
        model = Sub_Category
        fields = "__all__"
```

*Figure 97: Admin page's forms*

s > admins > dashboard.html > div.container.py-5 > section.mb-5 > div.row.gx-xl-5 > div.col-xl-3.col-md-6.mb-4.mb-xl-0 > a..d-flex.align-items-center.p-4.bg-glass.shadow.rounded-6.te

```html
{% extends 'admin_layout.html' %} {% load static %} {% block title %}
<title>profile</title>
{% endblock title %} {% block css_files %}

<link rel="stylesheet" href="{% static '/css/main.css' %}" />
<link rel="stylesheet" href="{% static '/css/homepage.css' %}" />
<link rel="stylesheet" href="{% static '/css/profile.css' %}" />

{% endblock css_files %} {% block main_content %}


<!-- Main content -->
<div class="container py-5">
    <!-- Section: Summary -->
    <section class="mb-5">
        <div class="row gx-xl-5">
            <div class="col-xl-3 col-md-6 mb-4 mb-xl-0">
                <!-- Card -->
                <a class="
                  d-flex
                  align-items-center
                  p-4
                  bg-glass
                  shadow-3-strong
                  rounded-6
                  text-reset
                  ripple
                " href="#" data-ripple-color="hsl(0, 0%, 75%)">
                  <div class="rounded-4 fs-1">
                      <i class="fa fa-user"></i>
                  </div>

                  <div class="ms-4">
                      <p class="text-muted mb-2">Users</p>
                      <p class="mb-0">
                          <span class="h5 me-2">{{users_count}}</span>
                          <small class="text-success text-sm"><i class="fas fa-arrow-up fa-sm me-1"></i>13.48%</small>
                      </p>
                  </div>
                </a>
```

*Figure 98: Admin dashboard code*

admins > templates > admins > send_notification.html > div#exampleModal.modal.fade

```
<table class="table container shadow">
  <thead>
    <tr>
      <th scope="col">Message</th>
      <th scope="col">Broadcast On</th>
      <th scope="col">Sent?</th>
    </tr>
  </thead>
  <tbody>
    {% for i in nots %}
    <tr>
      <td>{{i.message}}</td>
      <td>{{i.broadcast_on}}</td>
      <td>{{i.sent}}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>
  <div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
      <div class="modal-dialog">
          <div class="modal-content">
              <div class=" border-0 d-flex justify-content-end ">

                  <button type="button" class="btn-close me-1 mt-1" data-mdb-dismiss="modal"
                      aria-label="Close"></button>
              </div>
              <div class="modal-body">
                  <div class=" m-auto  p-5 ">
                      <h1 class="text-center">Add Notification</h1>
                      <form method = "POST" enctype="multipart/form-data">
                          {% csrf_token %}
                          {{ form.title | as_crispy_field }}
          {{ form.message | as_crispy_field }}
          {{ form.broadcast_on | as_crispy_field }}
          <button type="submit" class="btn btn-primary btn-block">Send</button>

                      </form>
                  </div>
              </div>

              </div>
```

*Figure 99: Admin Notification Control Page's Code*

# Thankyou!