

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

Abstract:	2
Table of Figures	5
Introduction	9
Aim.....	10
Objectives.....	10
Mission and Vision.....	10
Mind Map.....	11
Problem Definition	12
Solution	14
Scope of Megaplex	18
Stakeholder Analysis	19
Business Model Canvas.....	20
Ethical Consideration	21
Literature Review:.....	21
HamroBazar	21
eBay:.....	22
Sastoramro:	23
Quikr:	24
Coutloot:.....	25
Primary Research:.....	25
Methodology	26
PACT Analysis.....	28
Technology Used	28
Tools used in Megaplex.....	28
Use case Diagram	30
Activity Diagram.....	31
Flowchart Diagram.....	32
PESTEL Analysis.....	33
Updated Project Plan	35
Risk Analysis	37
Issue Log.....	38

Workflow Diagram	39
Megaplex Initial Prototype.....	40
Final Product.....	42
Limitations of Megaplex	67
Future Works:	69
Conclusion:	70
References:	70
Appendix	71

Table of Figures

Figure 1: Keywords	8
Figure 2: Mission and Vision of Megaplex	10
Figure 3: Mind map of the project.....	11
Figure 4: Mind map of Megaplex	11
Figure 5: Scope of Megaplex	18
Figure 6: Stakeholders Analysis of Megaplex.....	19
Figure 7: Business Model Canvas	20
Figure 8: Ethical Concerns.....	21
Figure 9: Hamrobazar	22
Figure 10: eBay.....	23
Figure 11: Sastoramro.....	24
Figure 12: Waterfall model	26
Figure 13: People Analysis	28
Figure 14: Tools Used	29
Figure 15: Use Case Diagram of Megaplex	30
Figure 16: Activity Diagram of Megaplex.....	31
Figure 17: Flowchart of Megaplex	32
Figure 18: PESTEL Analysis of Megaplex.....	33
Figure 19: Project Members	35
Figure 20: Project Dashboard	35
Figure 21: Project Dashboard 2	36
Figure 22: Tasks.....	36
Figure 23: Risk Analysis	37
Figure 24: Issue Logs	38
Figure 25: Workflow of Megaplex.....	39
Figure 26: Homepage	40
Figure 27: Profile page	40
Figure 28: Login page	41
Figure 29: Signup Page	41
Figure 30: Homepage	42
Figure 31: Homepage	42
Figure 32: Login page	43
Figure 33: Registration page	43
Figure 34: User profile	44
Figure 35: Edit Profile	44
Figure 36: Adding Product	45
Figure 37: Confirmation Page for Product Additions.....	45
Figure 38: Browse Products	46
Figure 39: View the top products in each category.....	46
Figure 40: Hot Collections	47

Figure 41: Megaplex Built in Chat.....	47
Figure 42: Built-in Wallet	48
Figure 43: Transaction History	48
Figure 44: Chatroom	49
Figure 45: General Chat Room.....	49
Figure 46: Filtering Products	50
Figure 47: Search result	50
Figure 48: Product Overview	51
Figure 49: Product details page	51
Figure 50: User Profile	52
Figure 51: Megaplex Notification	52
Figure 52: News and announcement page	53
Figure 53: User follower and the following page	53
Figure 54: Wishlist page.....	54
Figure 55: Admin dashboard.....	54
Figure 56: Page for sending notifications by Admin.....	55
Figure 57: Admin category viewing page	55
Figure 58: Admin's page for adding categories.....	56
Figure 59: Admin's page for editing categories	56
Figure 60: Admin Subcategory viewing page	57
Figure 61: Admin's page for adding Subcategories.....	57
Figure 62: Contact Form	58
Figure 63: Password reset form	58
Figure 64: Password reset form(Email Sent).....	59
Figure 65: Password reset (Email Sent).....	59
Figure 66: Page for entering a new password	60
Figure 67: Password reset conformation	60
Figure 68: Password change form	61
Figure 69: Game homepage	61
Figure 70: Game one	62
Figure 71: Game two.....	62
Figure 72: Dark mode homepage	63
Figure 73: Dark mode profile.....	63
Figure 74: Dark mode product details	64
Figure 75:Dark mode Wishlist.....	64
Figure 76: About us	65
Figure 77: About Us page 1	65
Figure 78: About Us page 2	66
Figure 79: Frameworks used to develop Megaplex	66
Figure 80: Quality Control.....	67
Figure 81: Payment	68
Figure 82: Possible Frauds.....	68

Figure 83: High Competitions	69
Figure 84: important code of megaplex	71
Figure 85: Installed Apps	72
Figure 86: Code for Layouts	74
Figure 87: Accounts	75
Figure 88: Forms	76
Figure 89: Important code of dashboard passcode for Dashboard	79
Figure 90: Code for Direct Chat	82
Figure 91: Code of Homepage	86
Figure 92: Code of Notification	88
Figure 93: Code of Product Detail app	90
Figure 94: Code of room Chat app	93
Figure 95: Code for admin section	95

KEYWORD

S.E.O for Megaplex



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

1



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

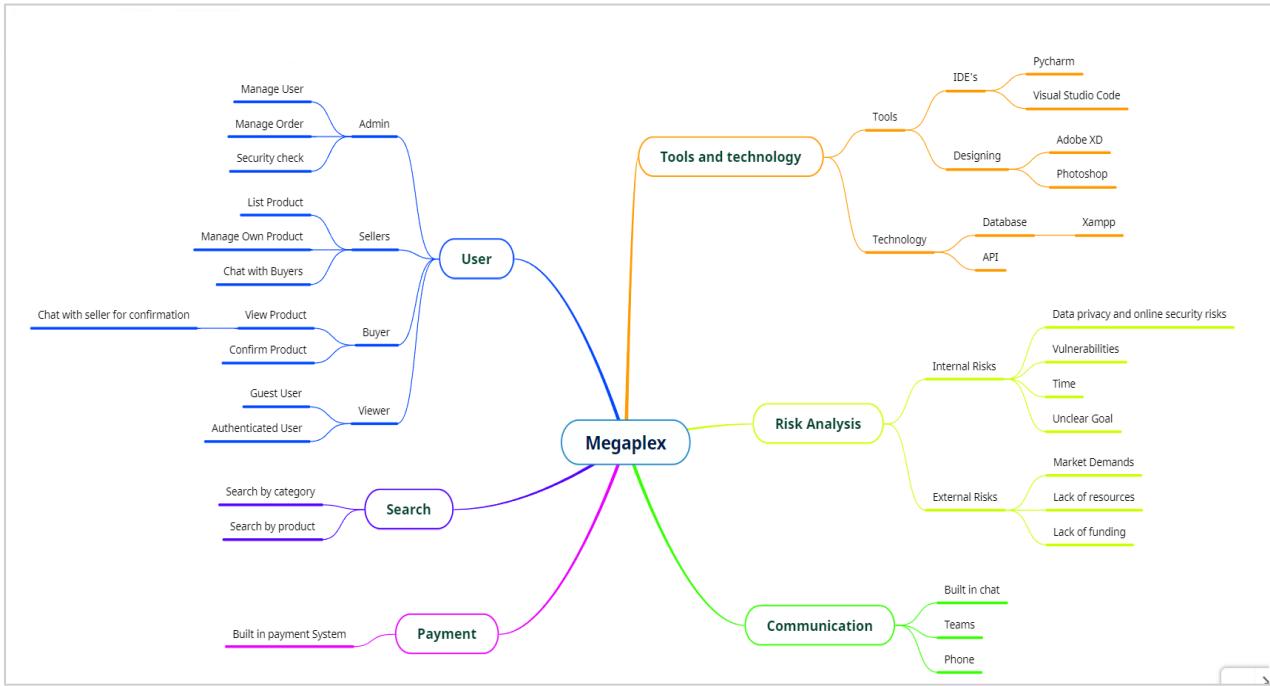


Figure 3: Mind map of the project

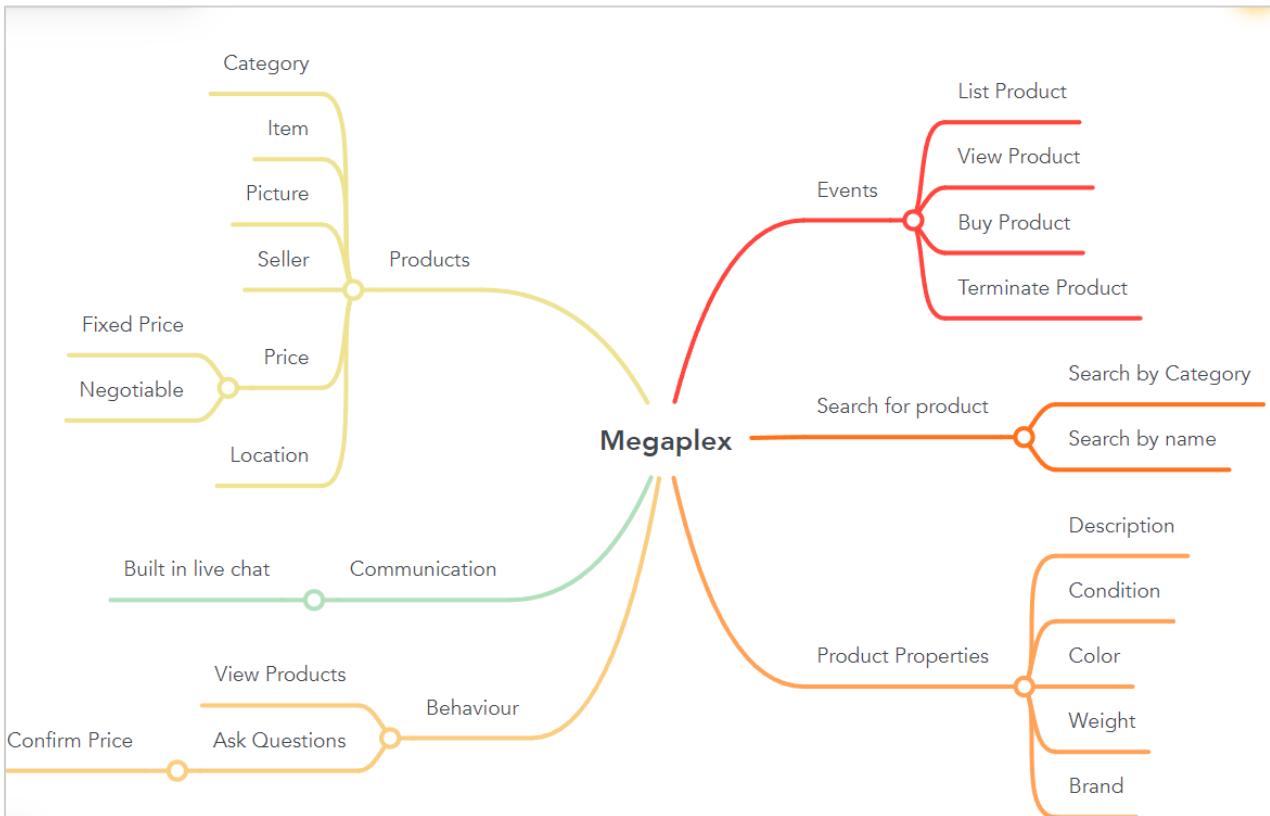
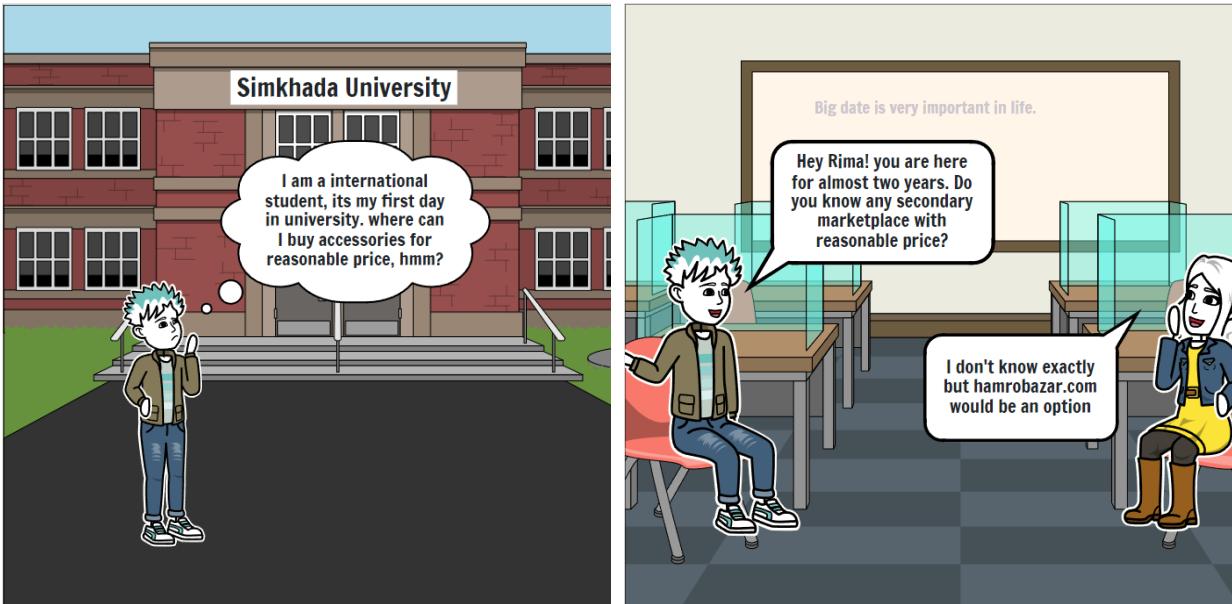


Figure 4: Mind map of Megaplex

Problem Definition

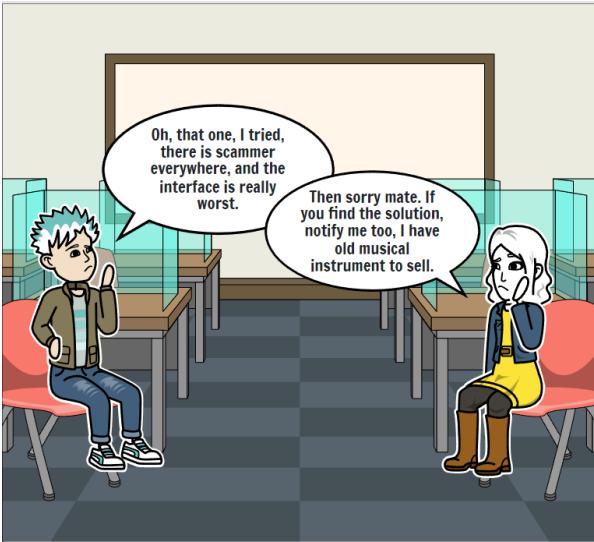
The smallest and largest online businesses that operate on a business-to-consumer (B2C), consumer-to-consumer (C2C), and buyer face these types of problems in their daily life. They face so much confusion about where to sell them and from where to buy their products like automobiles, real state, electronics, musical instruments, furniture 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.

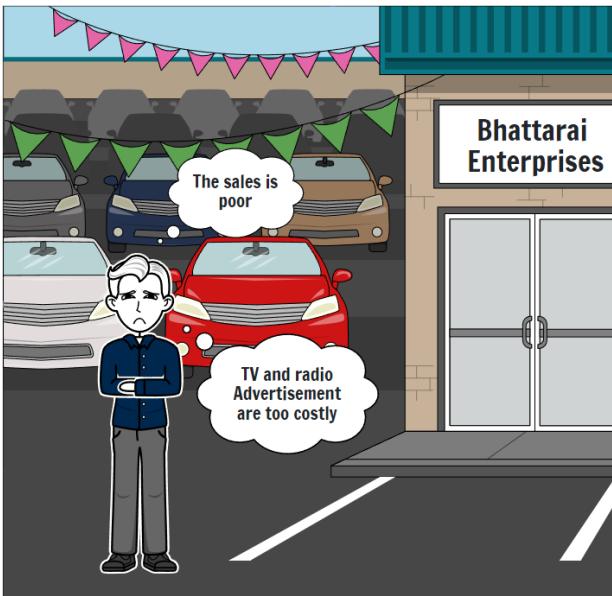


Salil already tried it and found it untrustworthy.

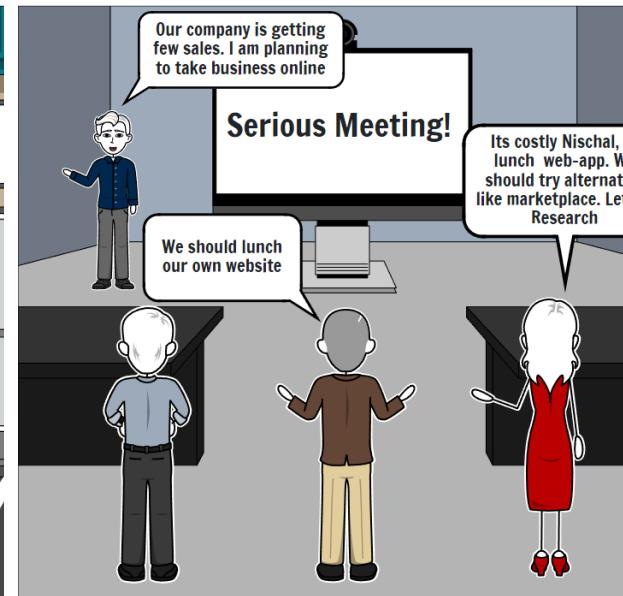


He then searched around a local store but was unable to find a quality good

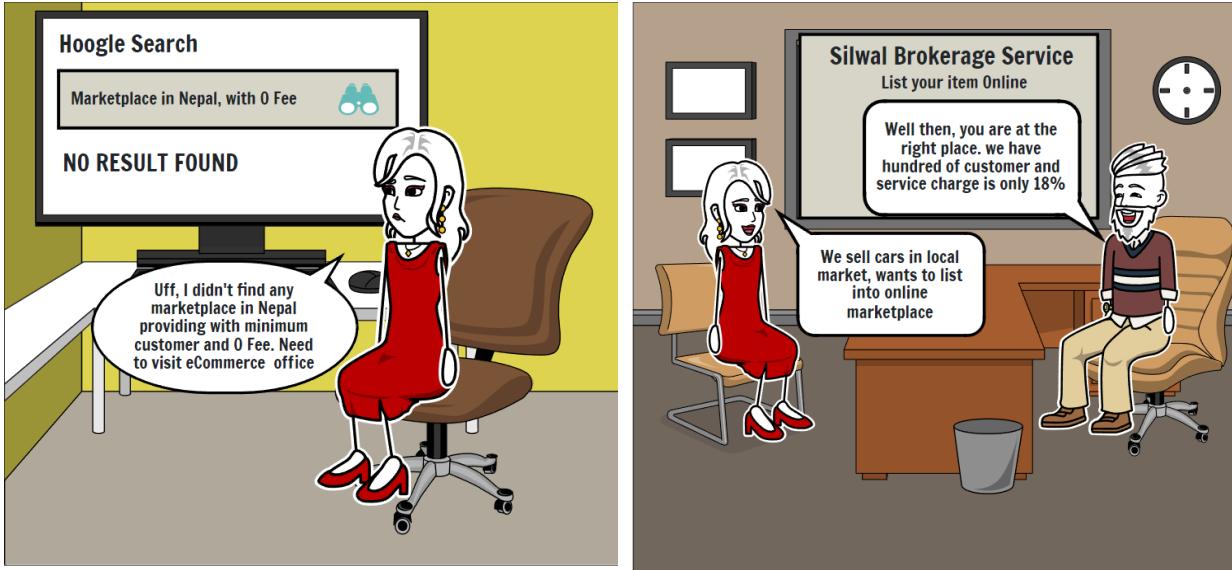
Case TWO



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



He organized the meeting, and got numerous suggestions.



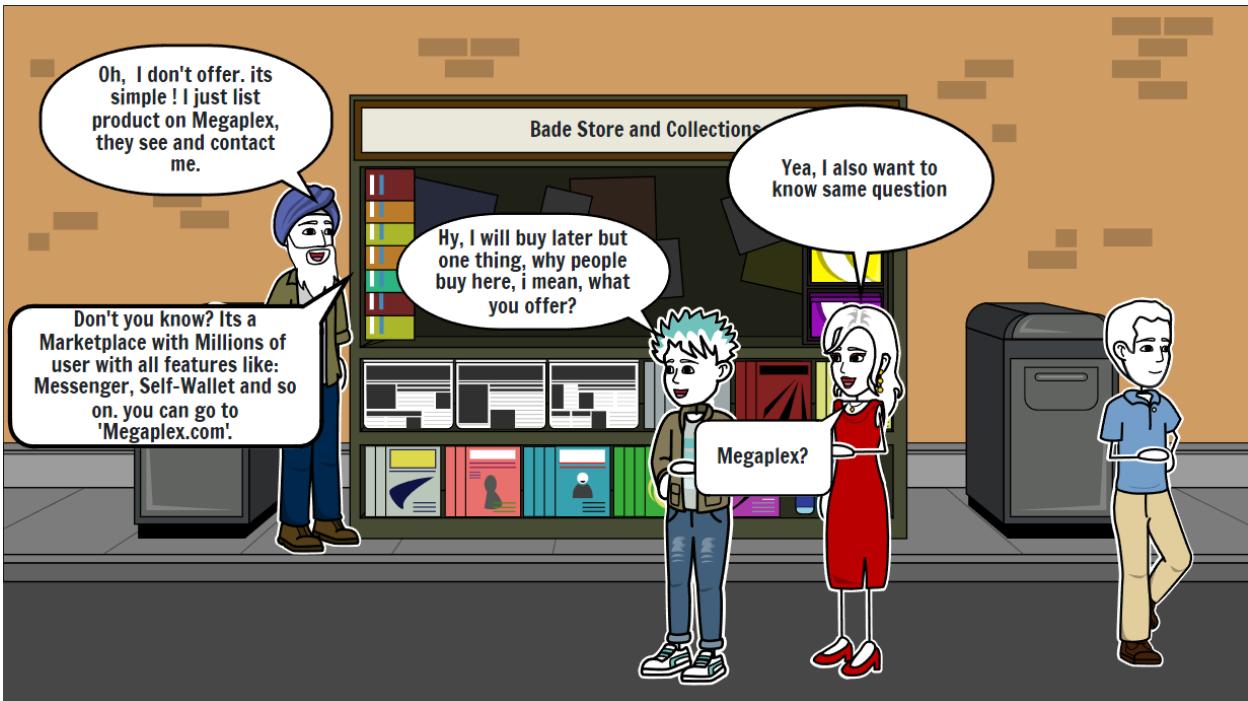
Lalita researched online but found no result. She visited a brokerage service but she doesn't like the offer

Solution

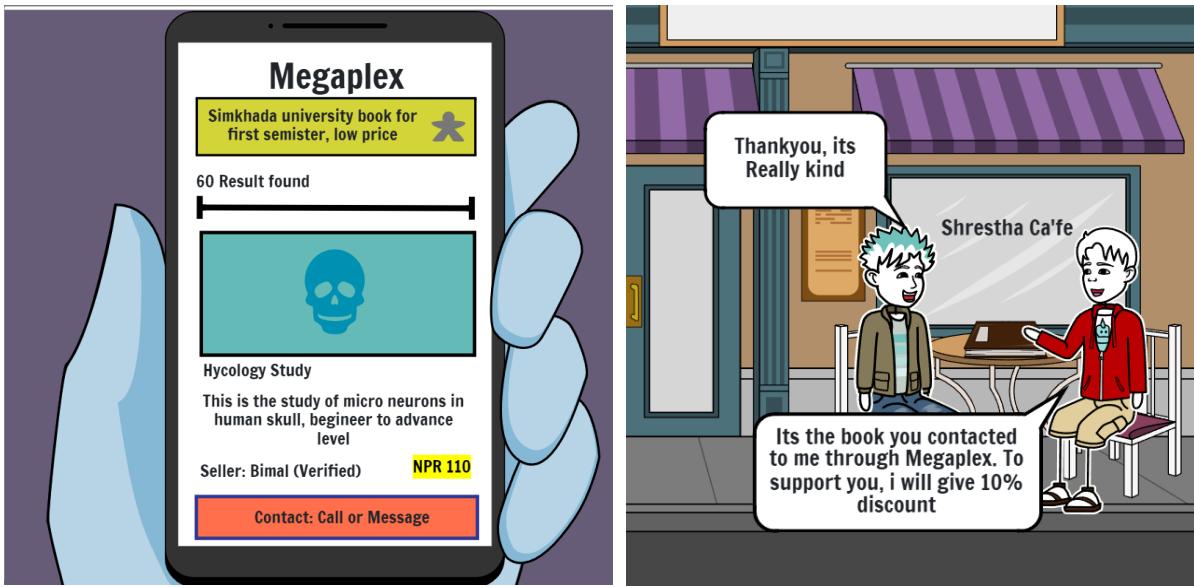
Megaplex is one of the biggest problem-solving marketplaces. In essence, a Megaplex allows a company to function almost entirely online, removing the requirement for a physical location with sales and administrative staff. The process is automated, which distinguishes it from other websites that still 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.

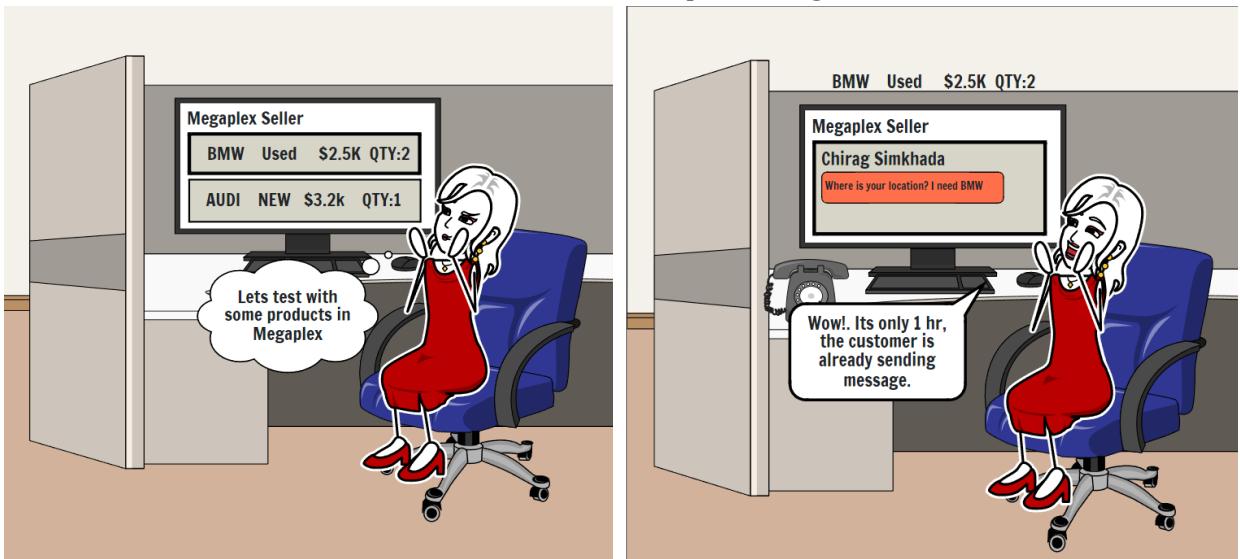


They asked the seller and found all the information about '**Megaplex**'.



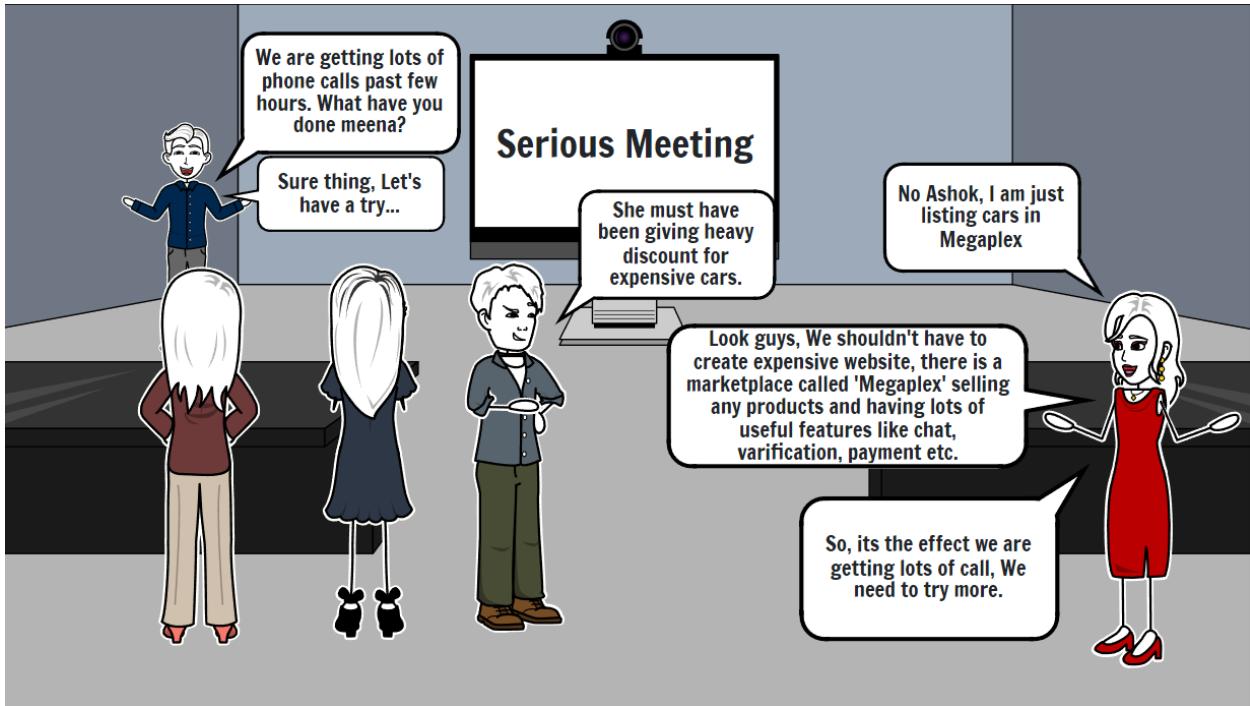
Salil then searched for Megaplex and found the price of the book is half that of a brand new one from a Verified seller: Bimal.

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

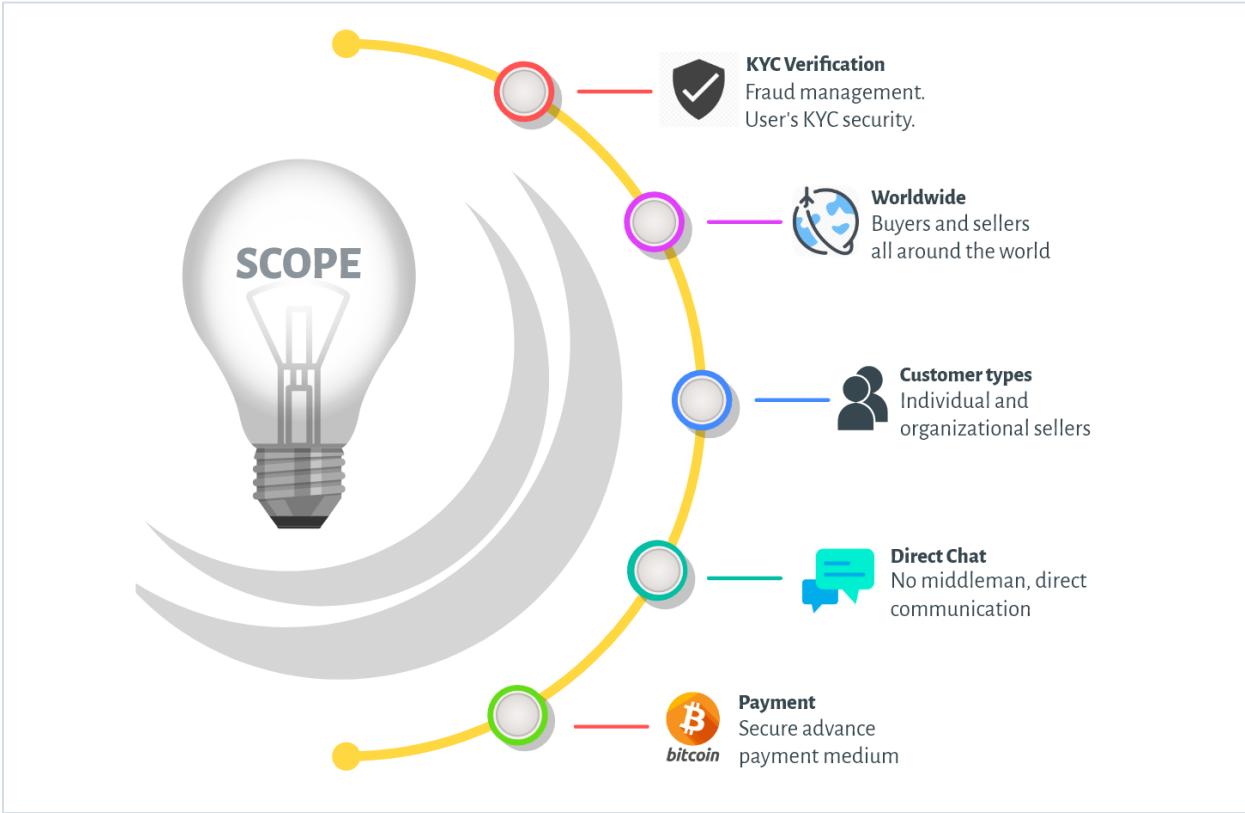


Figure 5: Scope of Megaplex

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

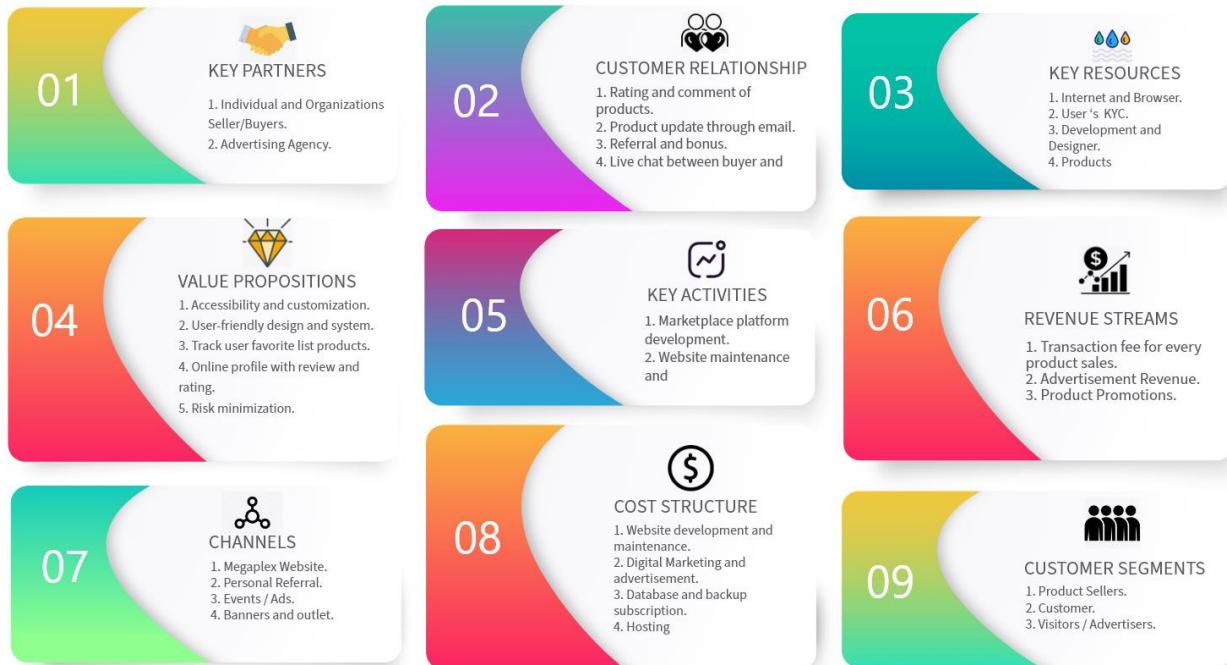


Figure 7: Business Model Canvas

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 leaked 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.

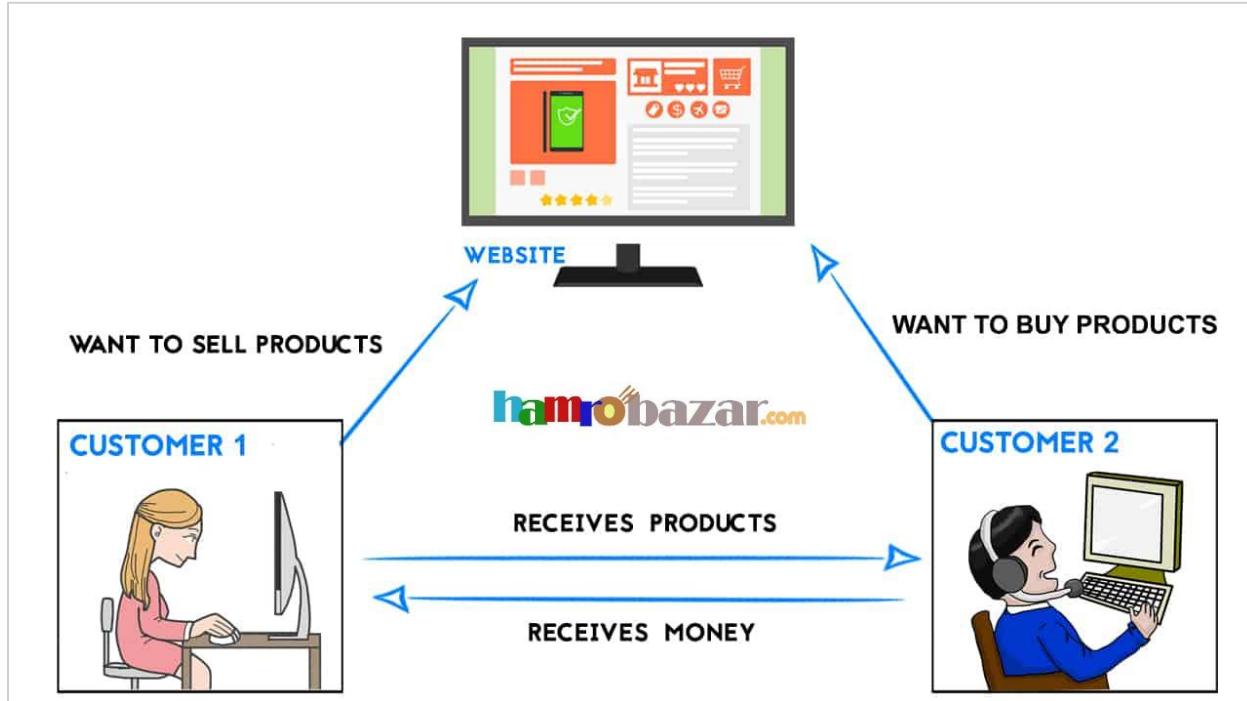


Figure 9: Hamrobazar

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.

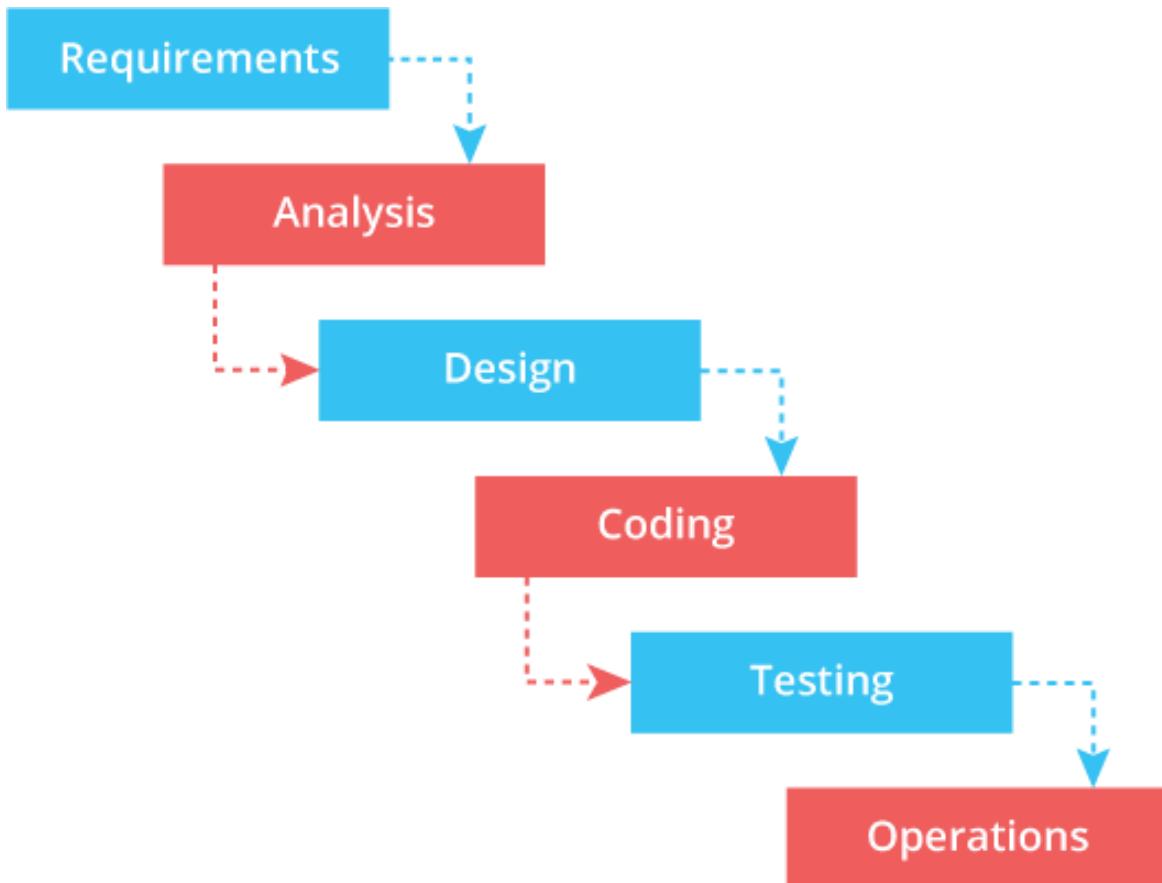


Figure 12: Waterfall model

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

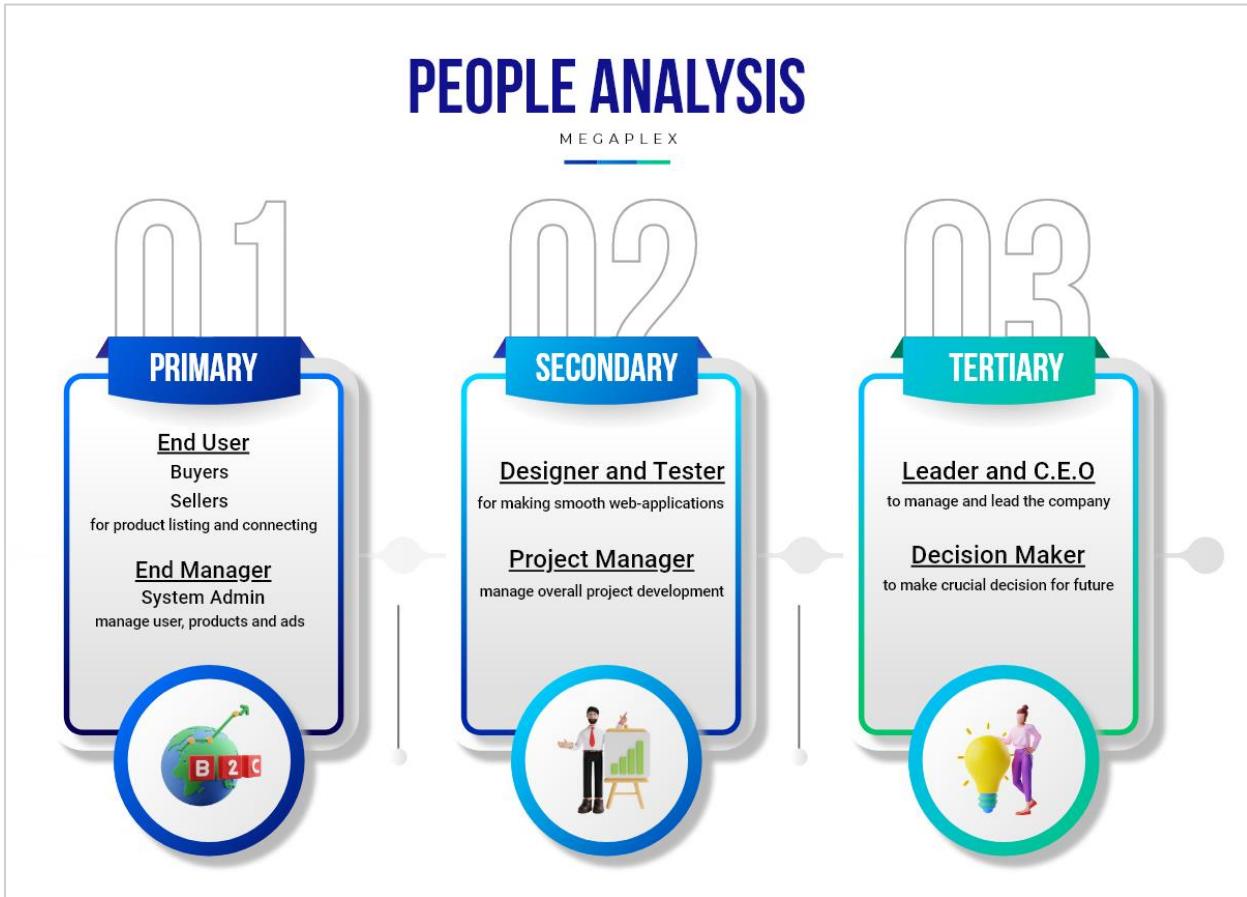


Figure 13: People Analysis

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

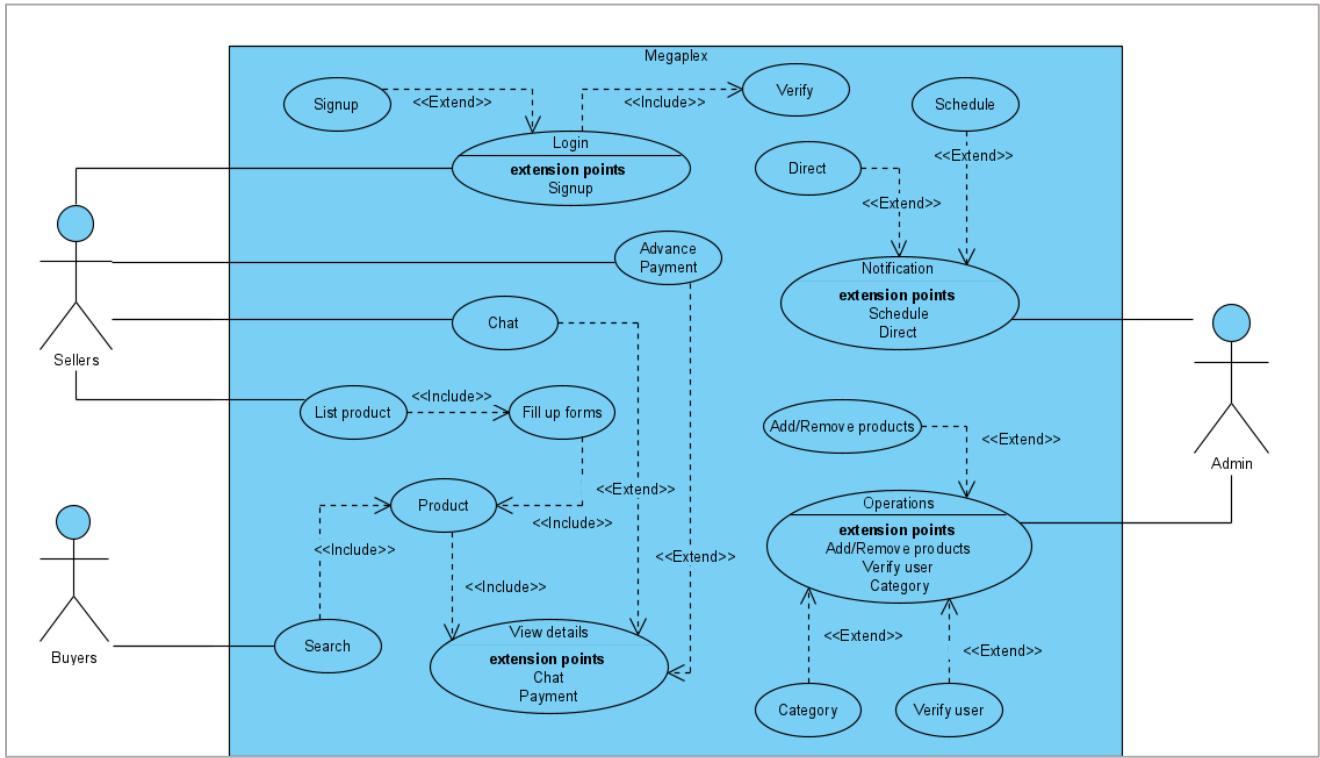


Figure 15: Use Case Diagram of Megaplex

Activity Diagram

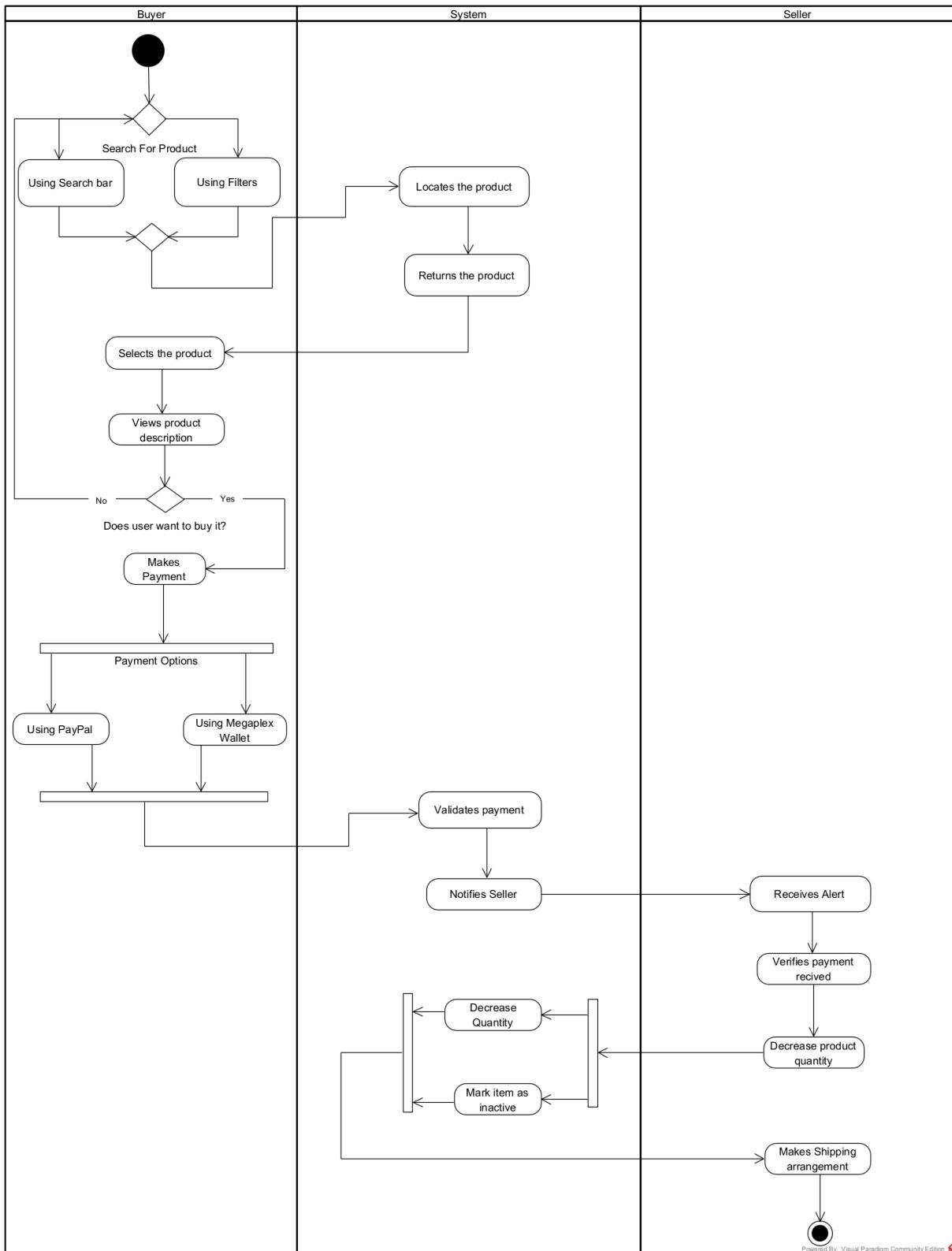


Figure 16: Activity Diagram of Megaplex

Flowchart Diagram

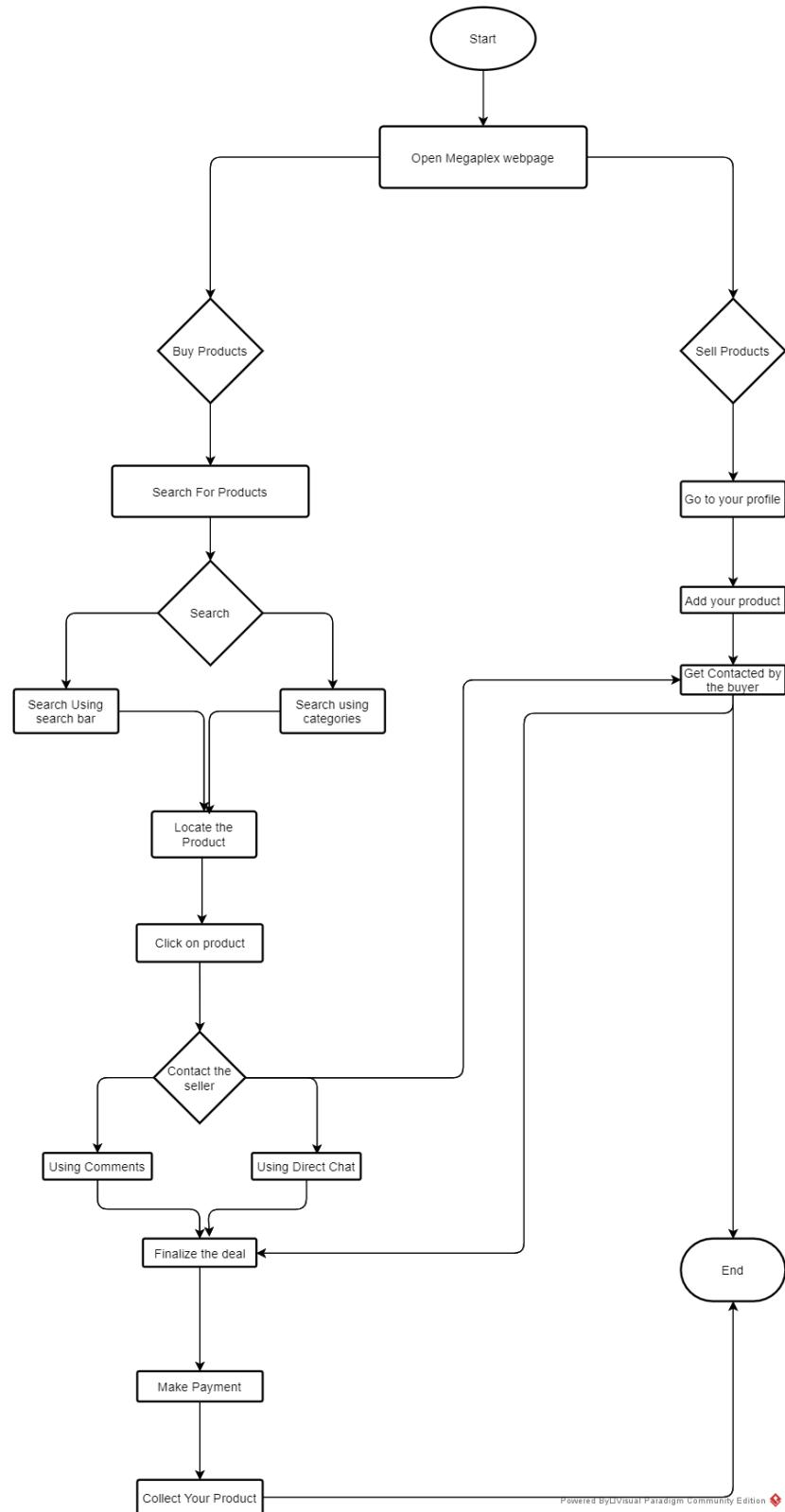


Figure 17: Flowchart of Megaplex

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

The screenshot shows the Asana interface for the 'Megaplex' workspace. On the left, the sidebar includes links for Home, My Tasks, Inbox, Reporting, Portfolios, and Goals. Under 'My Workspace', the 'Megaplex' project is selected. The main area displays the 'How we'll collaborate' section, which includes a welcome message and a list of team members. Below this is the 'Project roles' section, which lists the current members and their assigned roles:

Role	Member	Description
Project Owner	Bimal Shrestha (BS)	Added by itsbimal
Team Member	Chirag Simkhada (AS)	Added by itsbimal
Team Member	Ashok Shrestha (AS)	Added by itsbimal
Team Member	Salil Timalsina (ST)	Added by itsbimal
Team Member	nischal	Added by itsbimal

Figure 19: Project Members

The screenshot shows the Asana project dashboard for the 'Megaplex' workspace. It displays three main sections: 'To Do', 'Register', and 'Homepage'. Each section contains a list of tasks with their descriptions and status.

Section	Task Description	Added By
To Do	User online status	Added by itsbimal
To Do	Admin: sub category, brand, color etc.	Added by itsbimal
To Do	admin, unverified and verified and active and inactive	Added by itsbimal
To Do	Are you sure? for product delete and other important section	Added by itsbimal
To Do	About us page - Bimal(Working)	Added by itsbimal
Register	register label was made responsive	Added by Nis-chal
Register	Design change in registration page	Added by itsbimal
Register	Used crispy from for form creation....use command "pip install django-crispy-forms" and Migrate	Added by Chirag123-bit
Register	Added backend for registration form	Added by Chirag123-bit
Homepage	HOMEPAGE modified, First phase	Added by itsbimal
Homepage	local storage was used to change theme	Added by Nis-chal
Homepage	initial homepage was created	Added by Nis-chal
Homepage	Project Updated!	
Homepage	1. Master branch have only navbar Create each branch separately for each features. Active branch for Homepage Design: #homepage	

Figure 20: Project Dashboard

The dashboard displays three columns of cards:

- common feature** (8 cards):
 - user can like or remove like from product***
 - product view count was added
 - user can unfollow other users now
 - follow feature was added
 - label was made responsive
- Profile Page** (4 cards):
 - customers can click follow button to follow *** seller
 - toggled view was made
 - some content position was made fixed on scroll
 - initial profile page was created
- Direct Chat** (2 cards):
 - Run migrations command and Ensure that *** you are logged in through any account before using this. The temporary URL for it is "/chat"
 - Direct Chat "Double Message" issue is now *** fixed

Figure 21: Project Dashboard 2

Task name	Assignee	Due date	Priority	Status
Business Model In figure	BS Bimal Shrestha	Jan 1	Medium	Not Started
Products details backend	CS Chirag Simkhada	Jan 4	Medium	Not Started
Products details backend	CS Chirag Simkhada	Feb 3	High	In Progress
Primary Research	ST Salil Timalsina	Feb 20	Medium	Not Started
PESTEL Analysis	AS Ashok Shrestha	Jan 26	Medium	Not Started
Tools and Technology Figure	AS Ashok Shrestha	Jan 26	Medium	Not Started
Scope and business model Figure	AS Ashok Shrestha	Jan 11	Medium	Not Started
Risk analysis Figure	AS Ashok Shrestha	Jan 21	Medium	Not Started
Issue Log	BS Bimal Shrestha	Jan 23	Medium	Not Started
Branch Merge (Game, Explorev2)	BS Bimal Shrestha	Jan 20	Medium	Not Started
2 games are added	BS Bimal Shrestha	Jan 24	Medium	Not Started
Room Chat	CS Chirag Simkhada	Jan 24	Medium	Not Started
Paginator Fix	CS Chirag Simkhada	Jan 26	Medium	Not Started
Contact Form	CS Chirag Simkhada	Jan 18	Medium	Not Started
Product Recommendation	CS Chirag Simkhada	Jan 17	Medium	Not Started
Password Reset	CS Chirag Simkhada	Jan 21	Medium	Not Started

Figure 22: Tasks

Risk Analysis

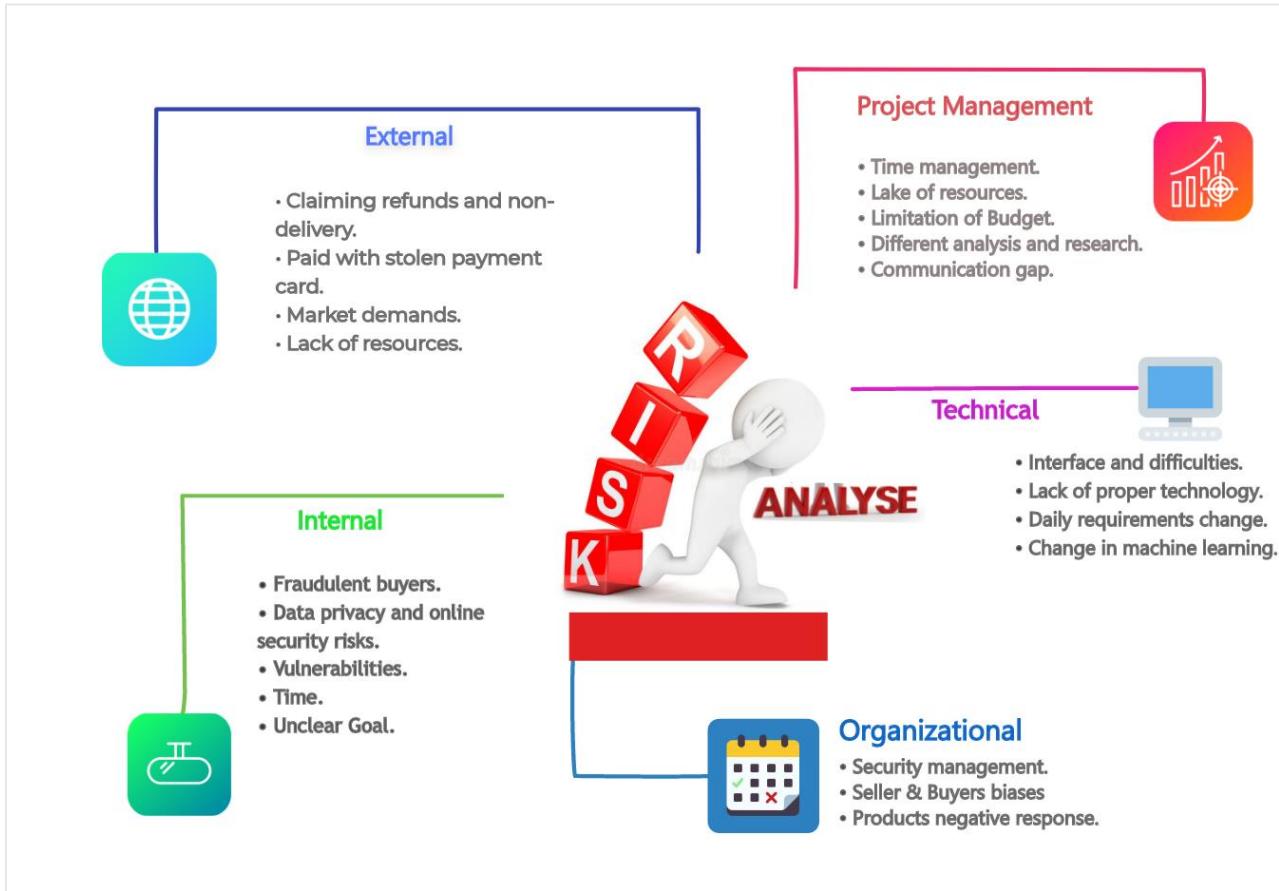


Figure 23: Risk Analysis

Issue Log

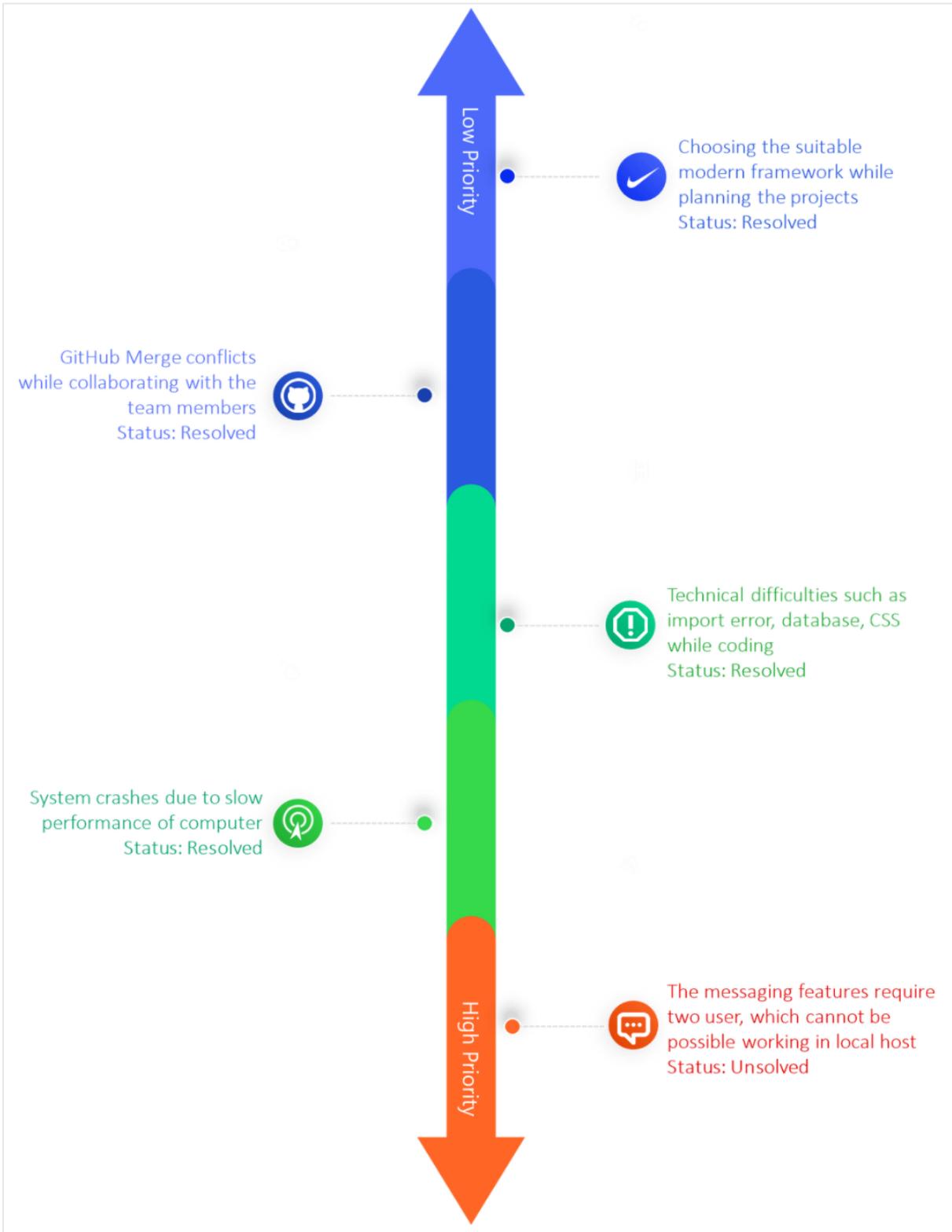


Figure 24: Issue Logs

Workflow Diagram

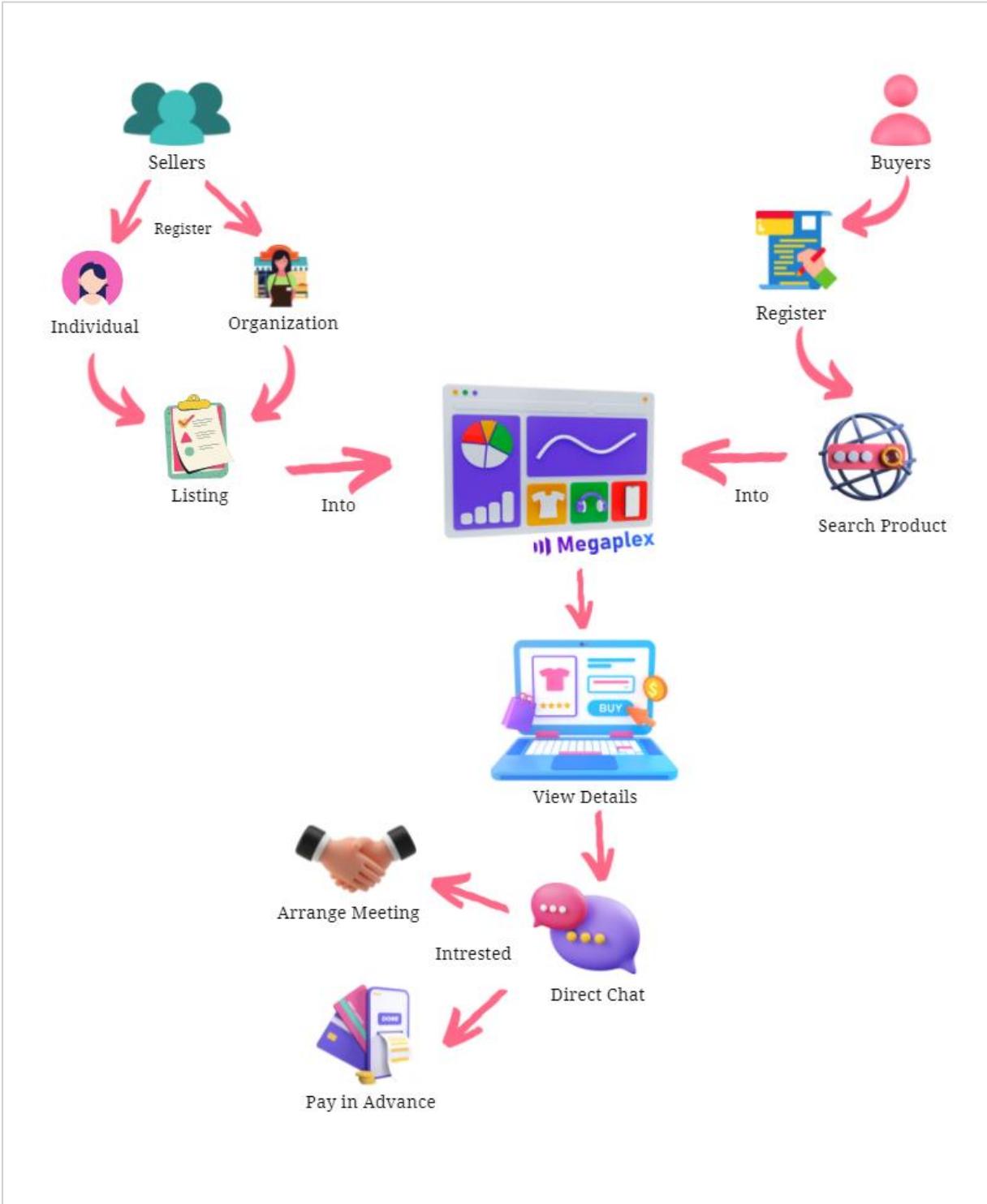


Figure 25: Workflow of Megaplex

Megaplex Initial Prototype (Adobe XD)

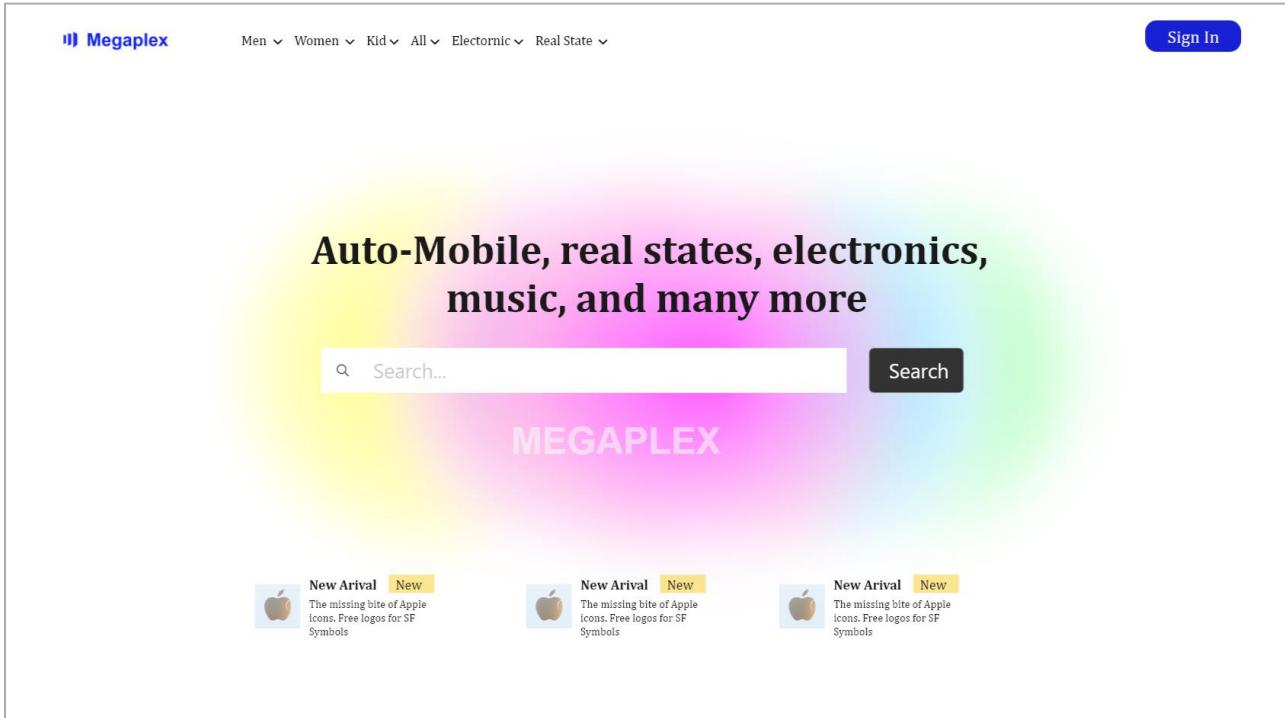


Figure 26: Homepage

The screenshot shows the profile page for a user named Ashok Shrestha. The top navigation bar includes a logo, a search bar, and a sign-in button. On the right side, there is a sidebar with options for "Your Profile", "Settings", and "Logout". The main content area features a large, vibrant background image of a stylized, glowing dragon-like creature. On the left, there is a user profile card with a photo of Ashok Shrestha, his name, and a bio: "A passinate leader of Nerd Squad Leading worlds largest project on ecommerce". Below the bio are "Follow" and "Edit profile" buttons. The user has a link to their website: "@ megaplex.com/Ashok". There are also social media icons for Twitter, Instagram, and Facebook. A message at the bottom indicates the user joined "Member since Mar 15, 2021". The main content area below the profile card shows three items from the user's collection, each labeled "Amazing art design" and priced at "\$ 2.45". Each item has a small profile picture, the name "From John", and the date "November 15, 2021".

Figure 27: Profile page

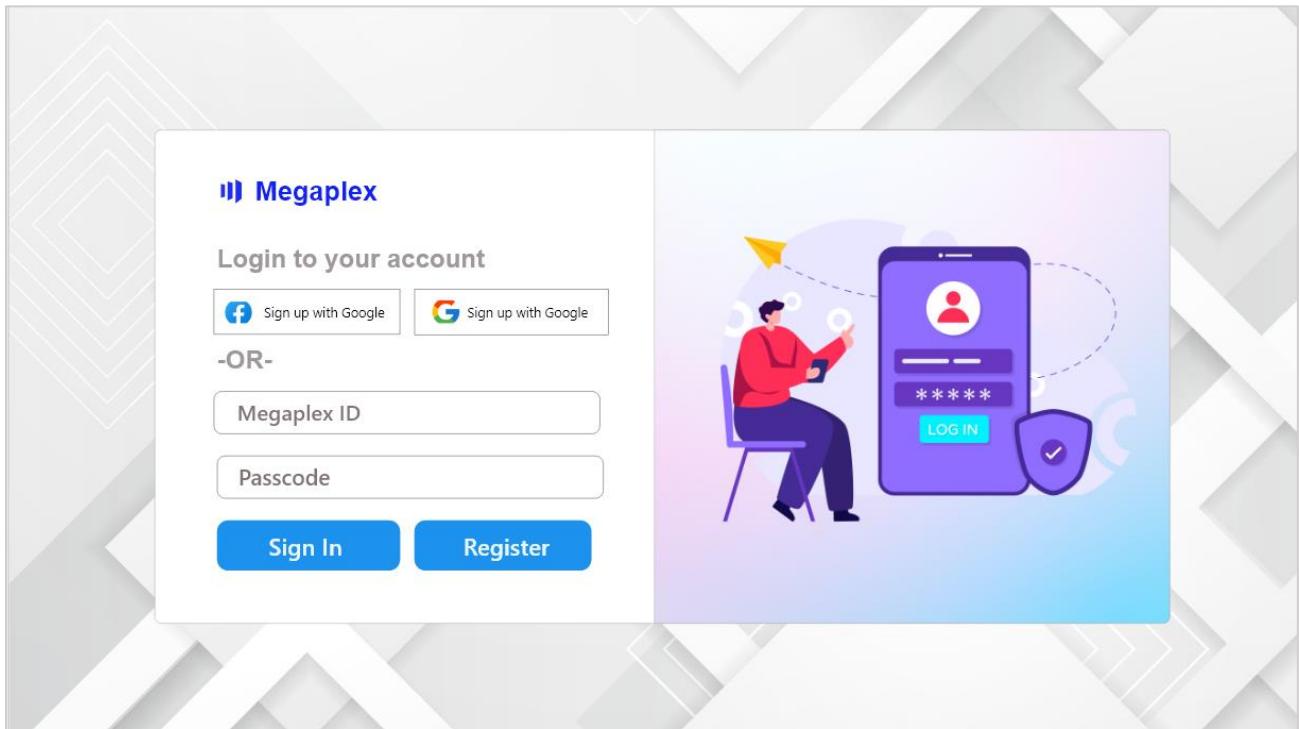


Figure 28: Login page

The signup page for Megaplex starts with the Megaplex logo and a navigation bar with dropdown menus for Men, Women, Kid, All, Electronic, and Real State. On the right is a "Sign In" button. The main area features the text "Enter the Megaplex World" above a cartoon character running with a padlock. To the left is a potted plant. To the right are input fields for FirstName, LastName, Email, Password, Verify Password, and a "Recovery Password" link. A large blue "Sign up" button is at the bottom, with "or continue with" and social media icons for Facebook and Google below it.

Figure 29: Signup Page

Final Product

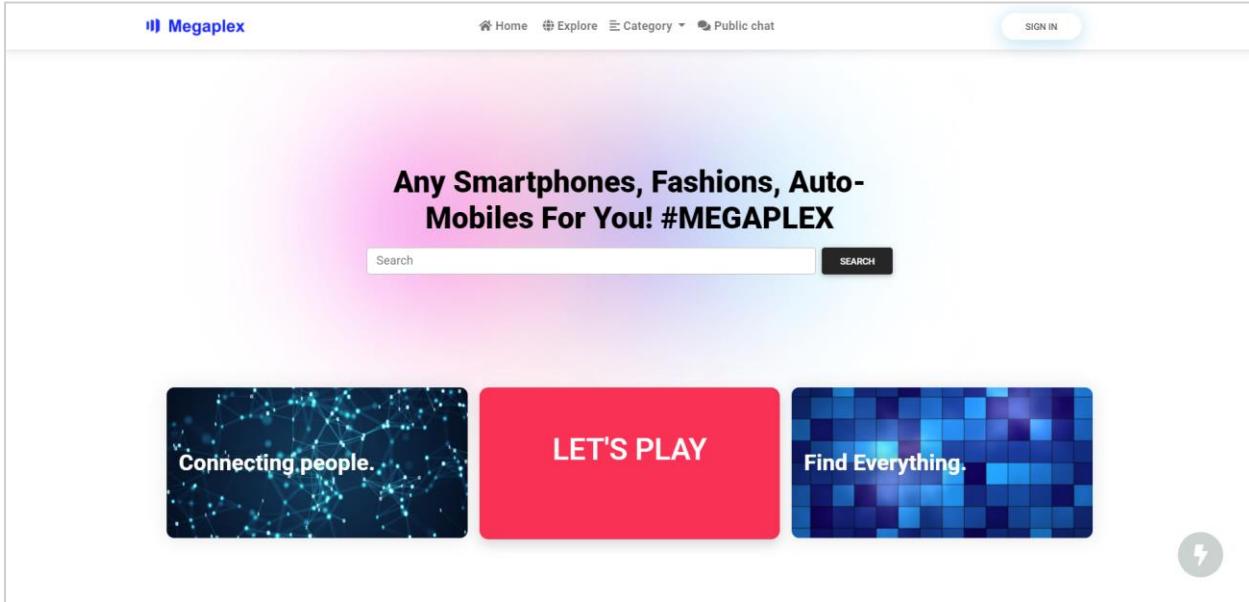


Figure 30: Homepage

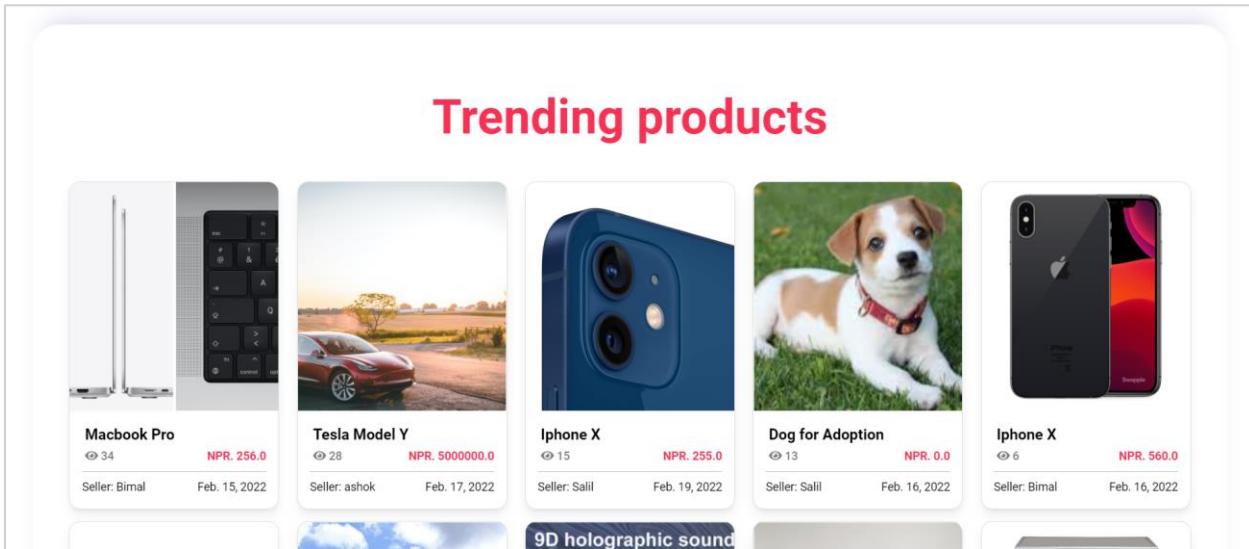


Figure 31: Homepage

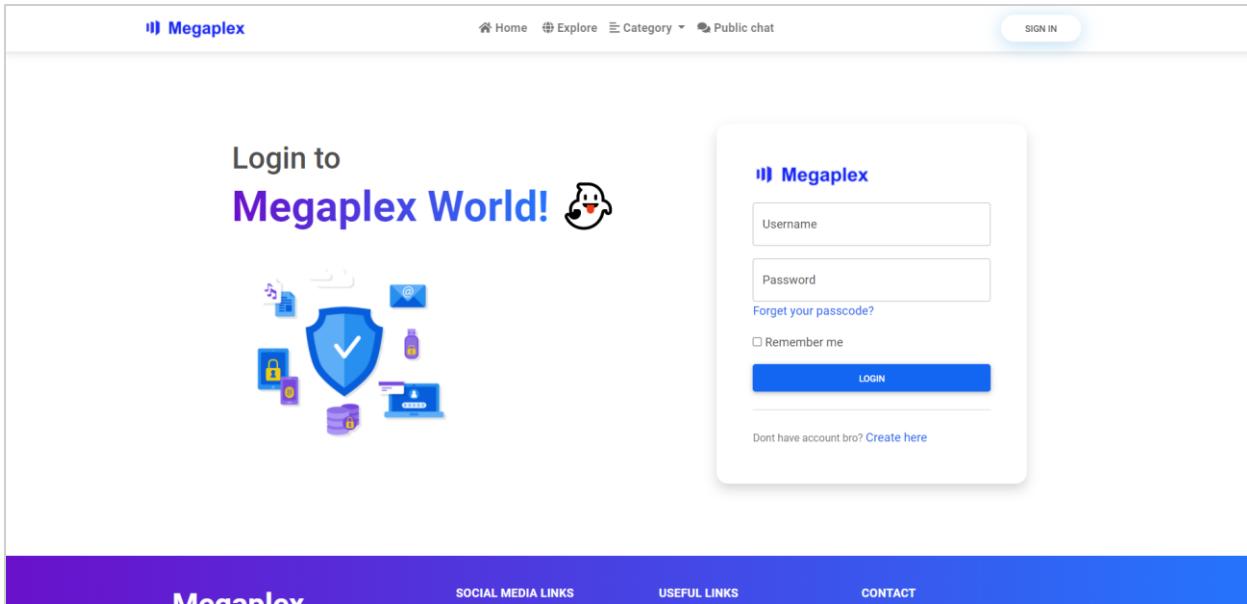


Figure 32: Login page

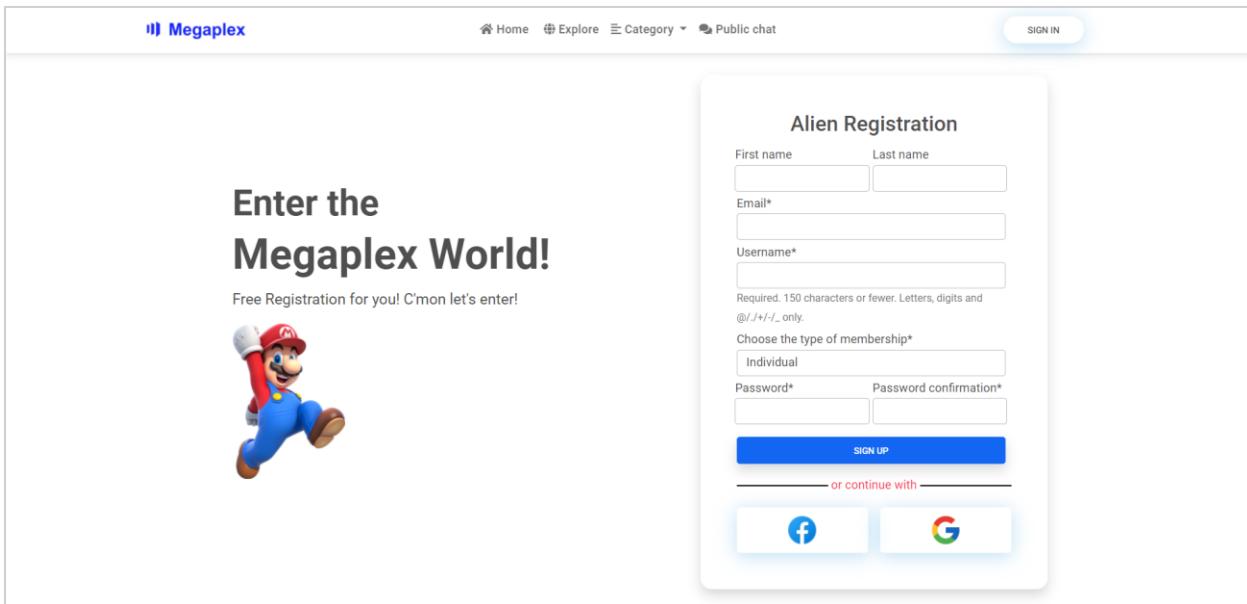


Figure 33: Registration page

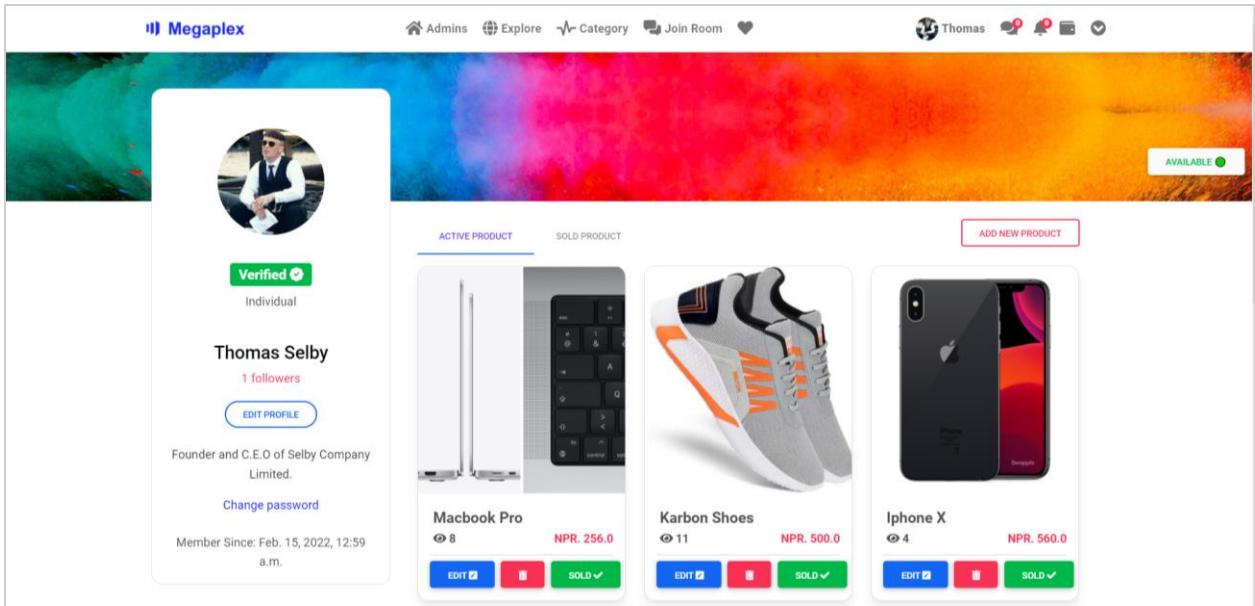


Figure 34: User profile

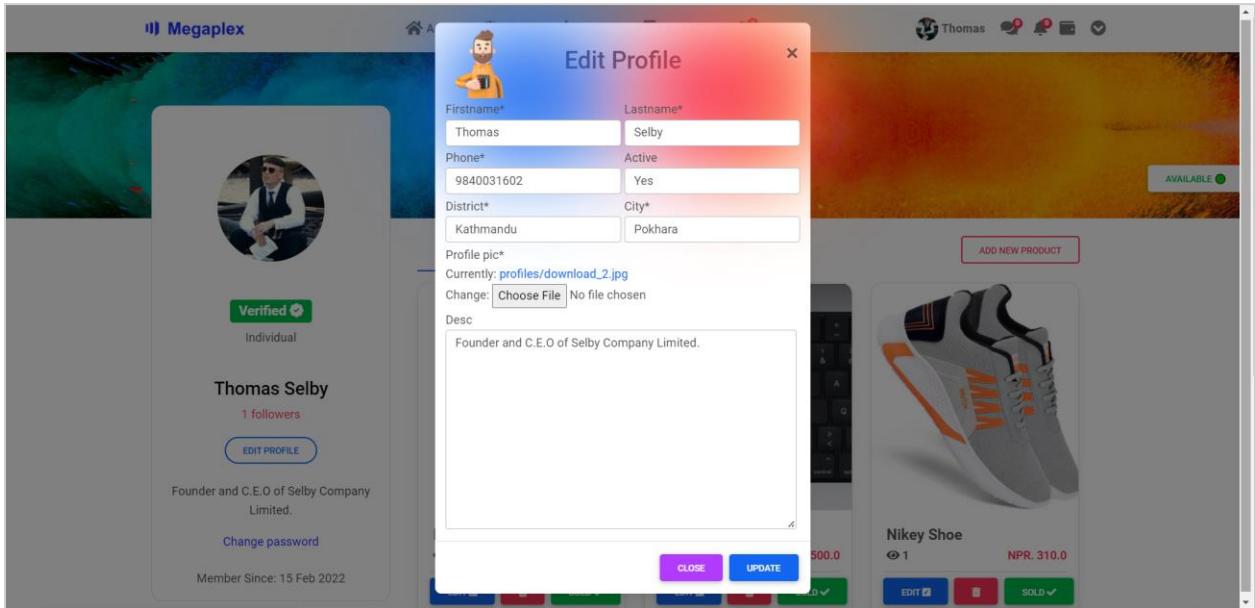


Figure 35: Edit Profile

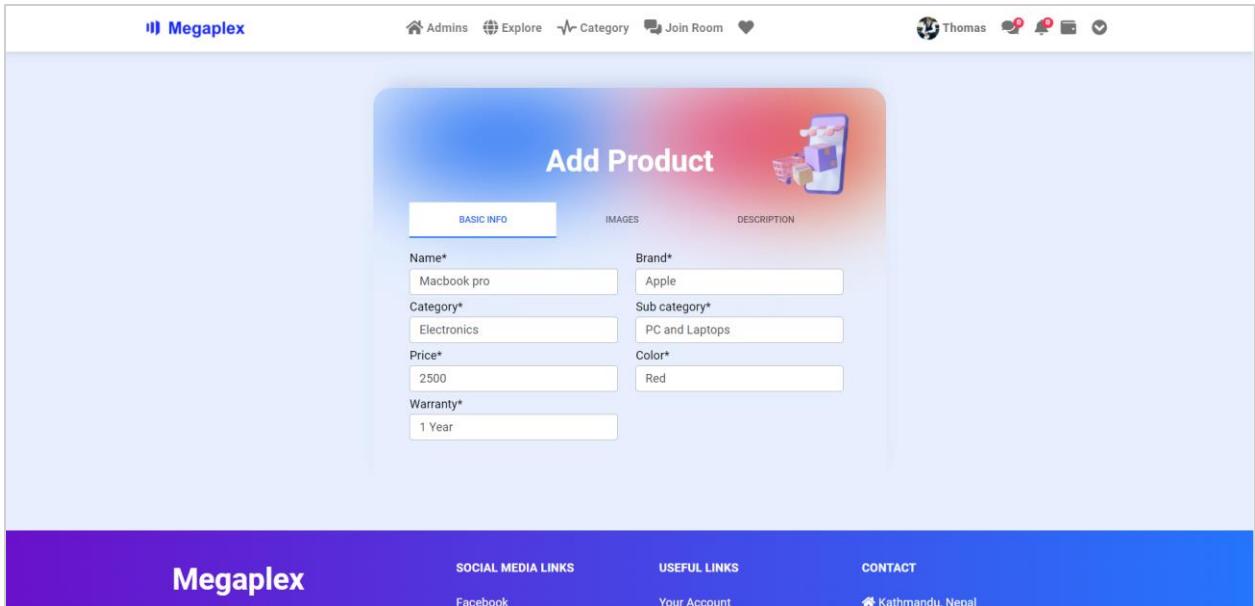


Figure 36: Adding Product

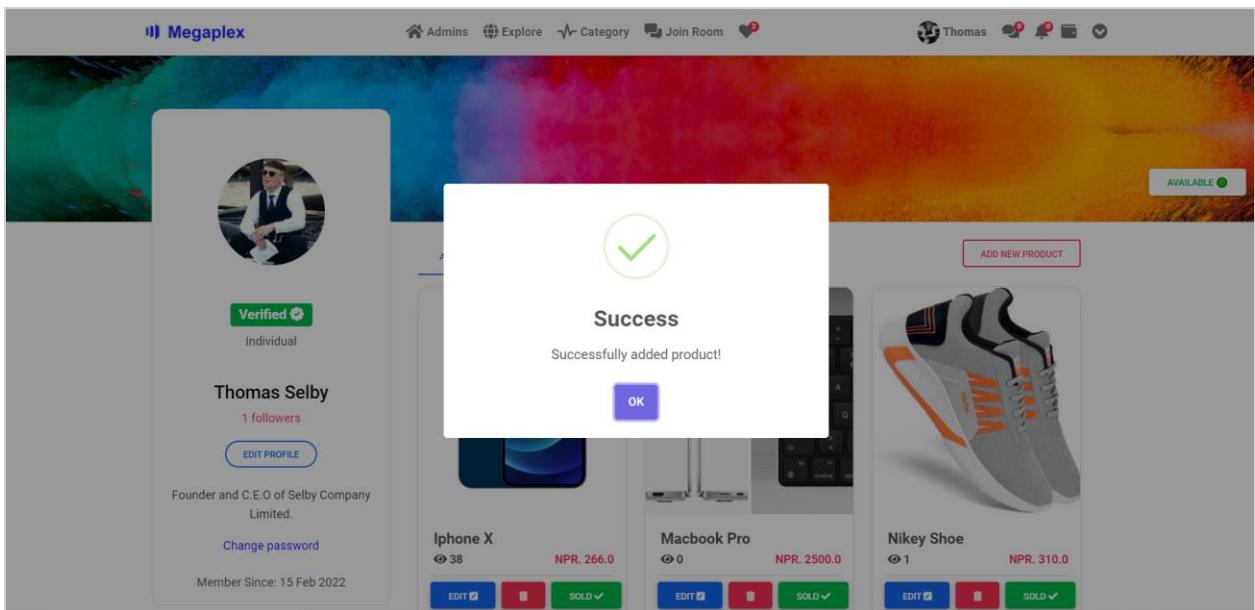


Figure 37: Confirmation Page for Product Additions

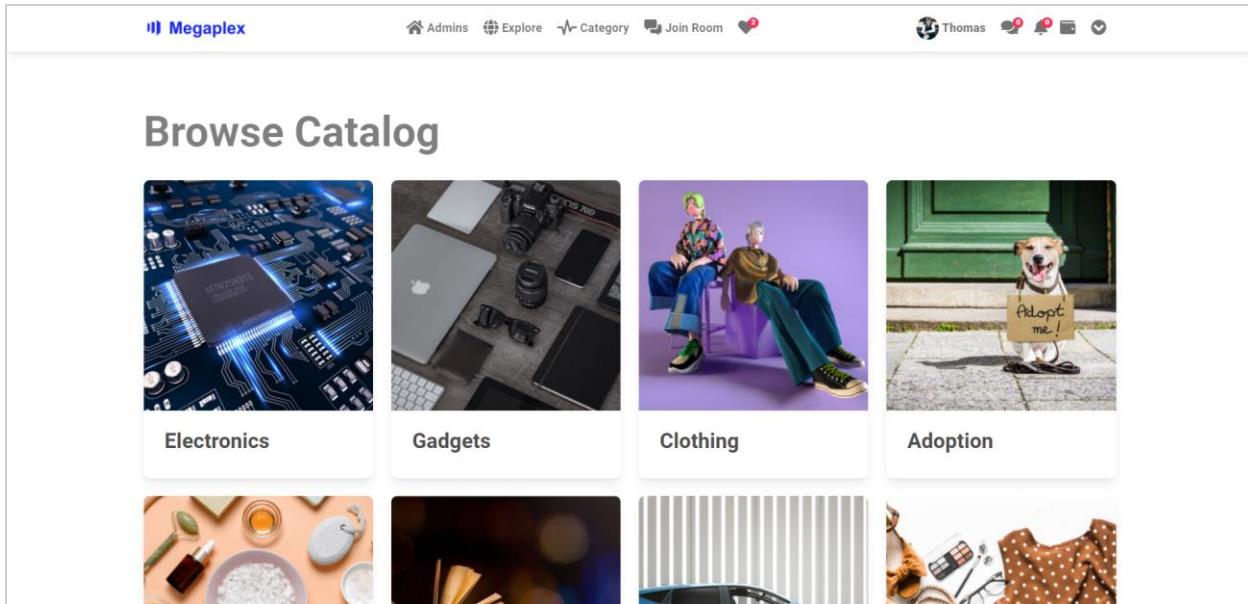


Figure 38: Browse Products

Category	Product	Seller	Price	Rating
Top Electronics	iPhone X	Thomas	NPR. 266.0	★★★★★
	Washing Machine	salitimalsina	NPR. 2000000.0	★★★★★
	Macbook Pro	Thomas	NPR. 2500.0	★★★★★
	Samsung Ipad	Thomas	NPR. 451.0	★★★★★
view more				
Top Clothes				

Figure 39: View the top products in each category.

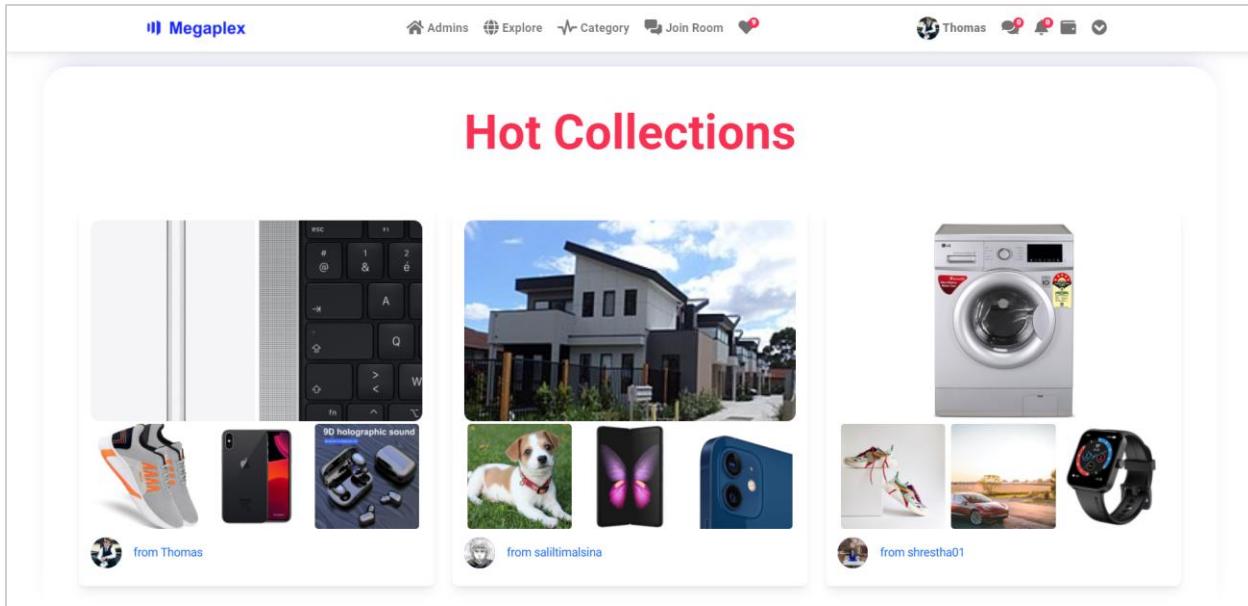


Figure 40: Hot Collections

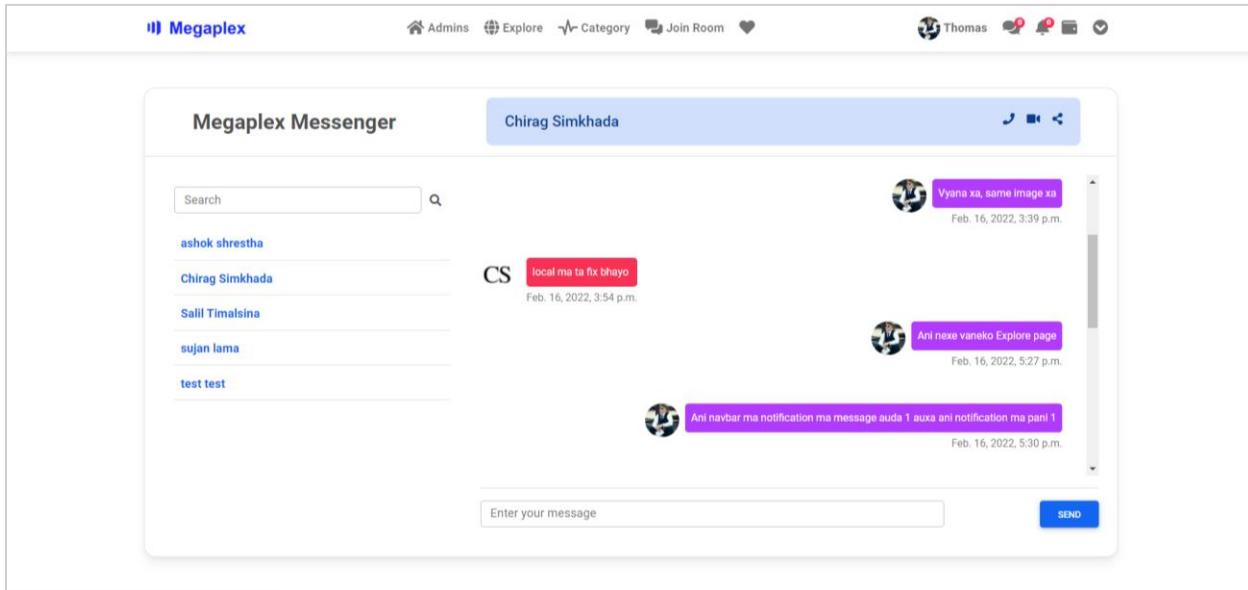


Figure 41: Megaplex Built in Chat

The screenshot shows the Megaplex wallet interface. At the top, there are navigation links: Admins, Explore, Category, Join Room, and a user profile for Thomas. Below the header, the title "Megaplex wallet" is displayed. A form for sending money to another wallet is shown, with fields for Sender (@ Thomas) and Receiver (@ shrestha01). It includes a note about a secure payment gateway between client and server, an amount field (Amount in \$ 20), and a "CHECK TRANSFER" button. To the right is a digital wallet card for "Holder Selby Thomas" with a balance of "\$ 58.00". The card has a purple-to-orange gradient background and a paper airplane icon. Below the wallet card is a "REQUEST" button.

Figure 42: Built-in Wallet

The screenshot shows the transaction history page. At the top, there is a heading "Secure money transfer log history". Below it is a table with columns: TXN_ID, Sender's Name, Receiver's Name, Amount, and Status. The data in the table is as follows:

TXN_ID	Sender's Name	Receiver's Name	Amount	Status
8200577	Thomas	salitimalsina	\$11.00	Sent ↑

Below the table, there is a pagination indicator showing "1 2 Next".

Megaplex

We created the platform, that provides the opportunity to buy or sell product directly without broker commission.

SOCIAL MEDIA LINKS

- Facebook
- Instagram
- LinkedIn
- Youtube

USEFUL LINKS

- Your Account
- Become an Affiliate
- Shipping Rates
- Help

CONTACT

- Kathmandu, Nepal
- teamhmrv2@gmail.com
- +977 01-984415852
- Made with Care, by Nerd Squad

2020 Copyright © Megaplex - All Right Reserved

Figure 43: Transaction History

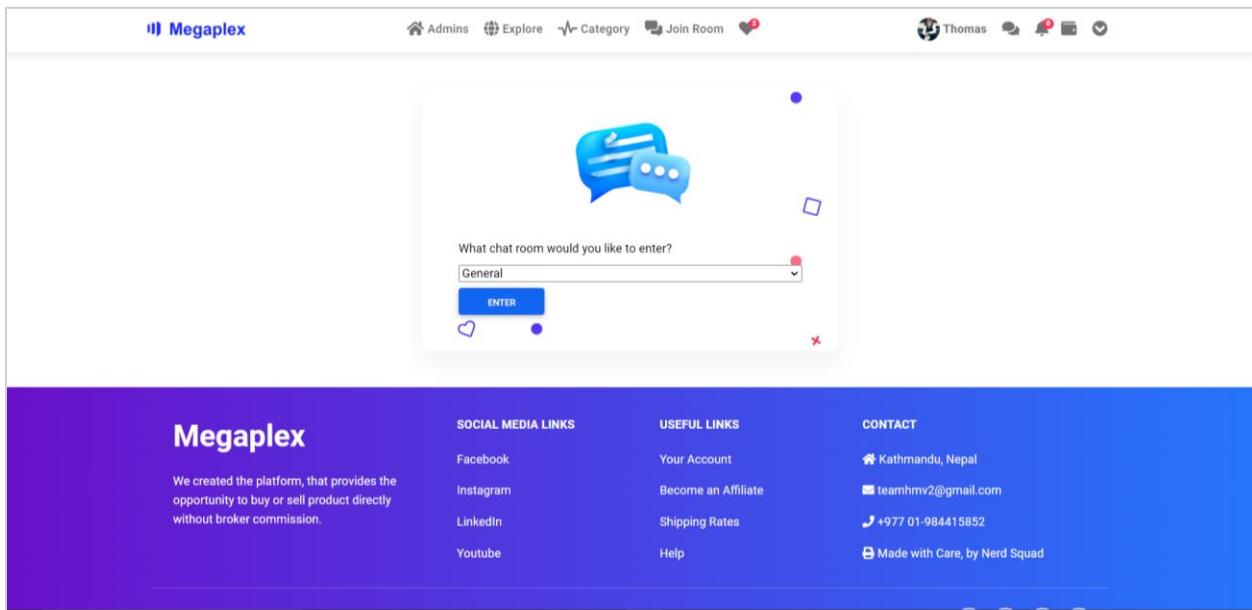


Figure 44: Chatroom

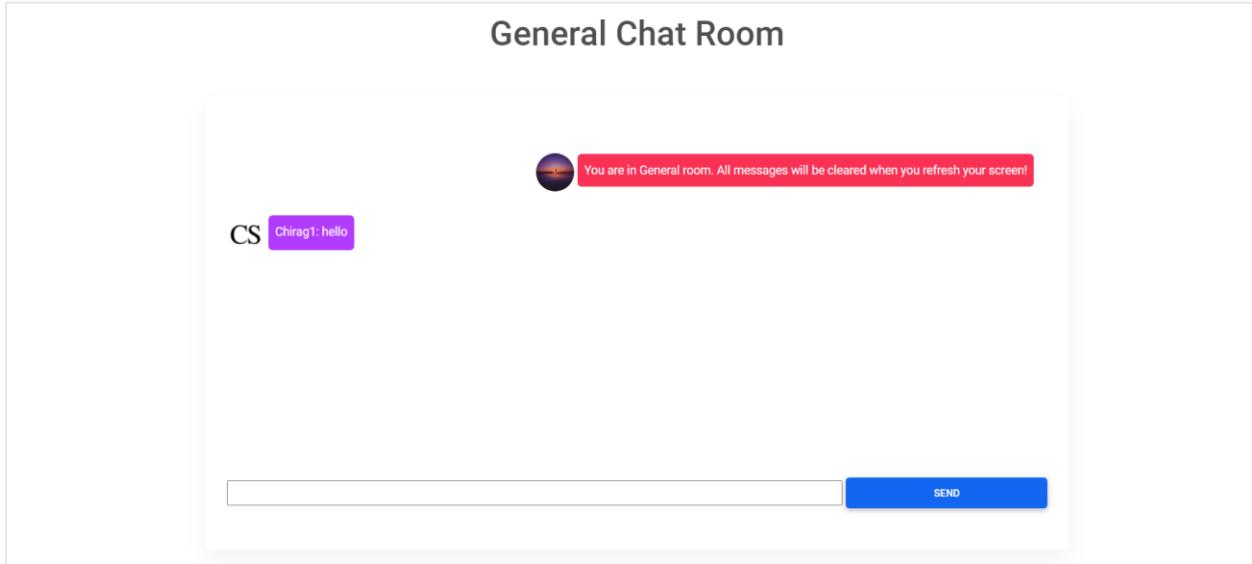


Figure 45: General Chat Room

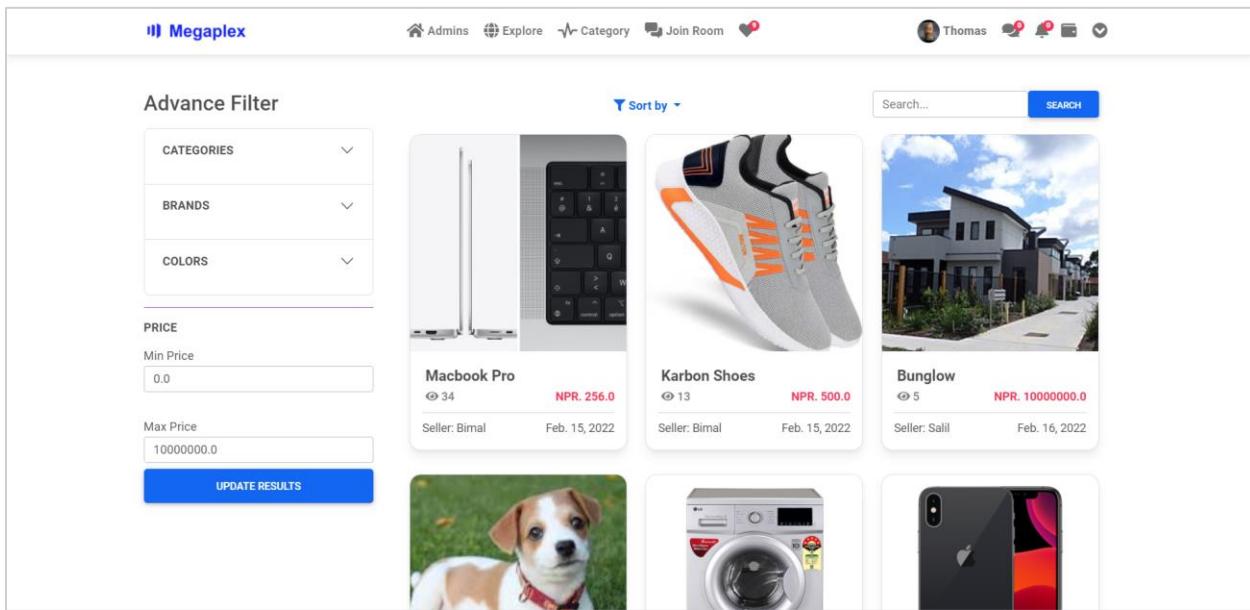


Figure 46: Filtering Products

The screenshot shows the Megaplex interface with the following elements:

- Header:** Megaplex logo, Admins, Explore, Category, Join Room, Thomas.
- Left Sidebar (Advance Filter):**
 - CATEGORIES dropdown
 - BRANDS dropdown
 - COLORS dropdown
 - PRICE section with Min Price (0.0) and Max Price (10000000.0) inputs, and an UPDATE RESULTS button.
- Search Result:**
 - Search result for: Iphone (highlighted in red).
 - Sort by: A dropdown menu currently set to "Sort by".
 - Search Bar: Search... and a blue SEARCH button.
- Product Grid:** A 2x2 grid of iPhone X products:
 - Top-left: iPhone X (6 reviews, NPR. 560.0), Seller: Bimal, Feb. 16, 2022.
 - Top-right: iPhone X (15 reviews, NPR. 255.0), Seller: Bimal, Feb. 19, 2022.
 - Bottom-left: Text: 2 found out of 13 products.
 - Bottom-right: Navigation buttons: Previous, Next.
- Footer:** Megaplex logo, SOCIAL MEDIA LINKS, USEFUL LINKS, CONTACT.

Figure 47: Search result

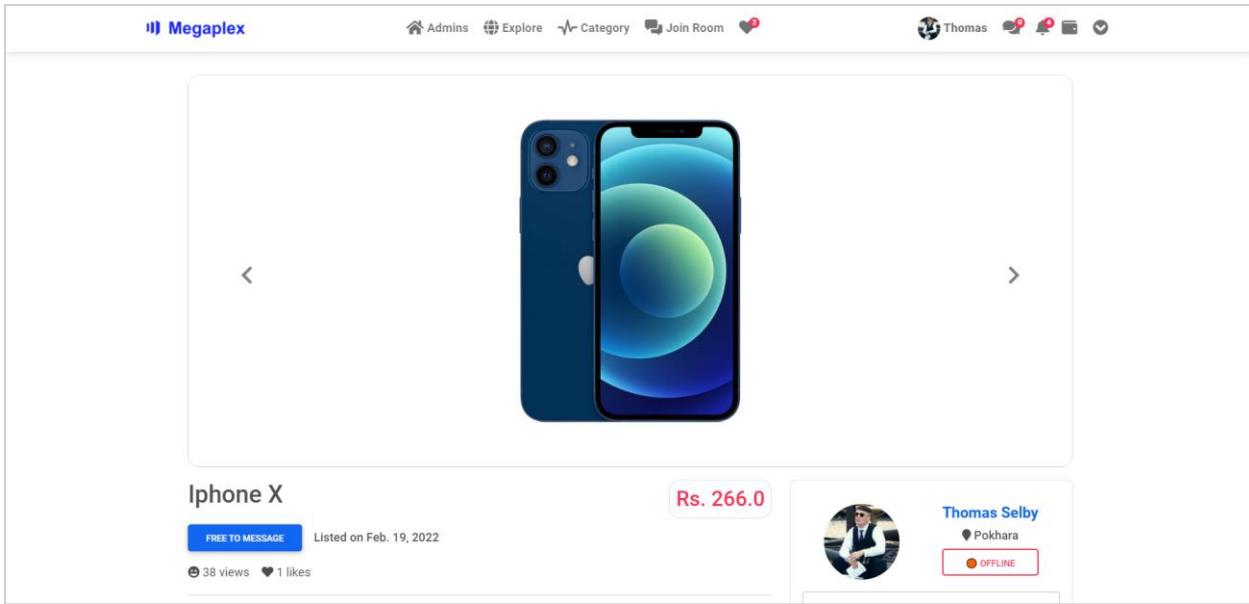


Figure 48: Product Overview

Our Commitment

[LIVE CHAT](#) [REFUND POLICY](#) [AUTHORIZED SELLER](#) [GUARANTEED PRODUCT](#)

Specification

Brand: Apple Category: Electronics Color: White

Warranty: 2 Years Quantity: 1

Description

6.1-inch Liquid Retina HD LCD display Water and dust resistant (2 meters for up to 30 minutes, IP68) Dual-camera system with 12MP Ultra Wide and Wide cameras; Night mode, Portrait mode, and 4K video up to 60fps

Product Details Section

12MP TrueDepth front camera with Portrait mode, 4K video, and Slo-Mo Face ID for secure authentication and Apple Pay A13 Bionic chip with third-generation Neural Engine Fast-charge capable Wireless charging

Post your speak!

What is your view?

Comment can be viewed below!

Recent Comments

1 comments

Bimal Shrestha
Shared publicly - Feb. 19, 2022, 11:43 a.m.
230 samma auxa ? ma aile lina auxu.
[Like](#) 0 Like [Reply](#)

Message

SEND FREE MESSAGE

Sent money to buy item

Sender:

Receiver:

Secure payment gateway between client

Amount in \$ **SEND**

Figure 49: Product details page

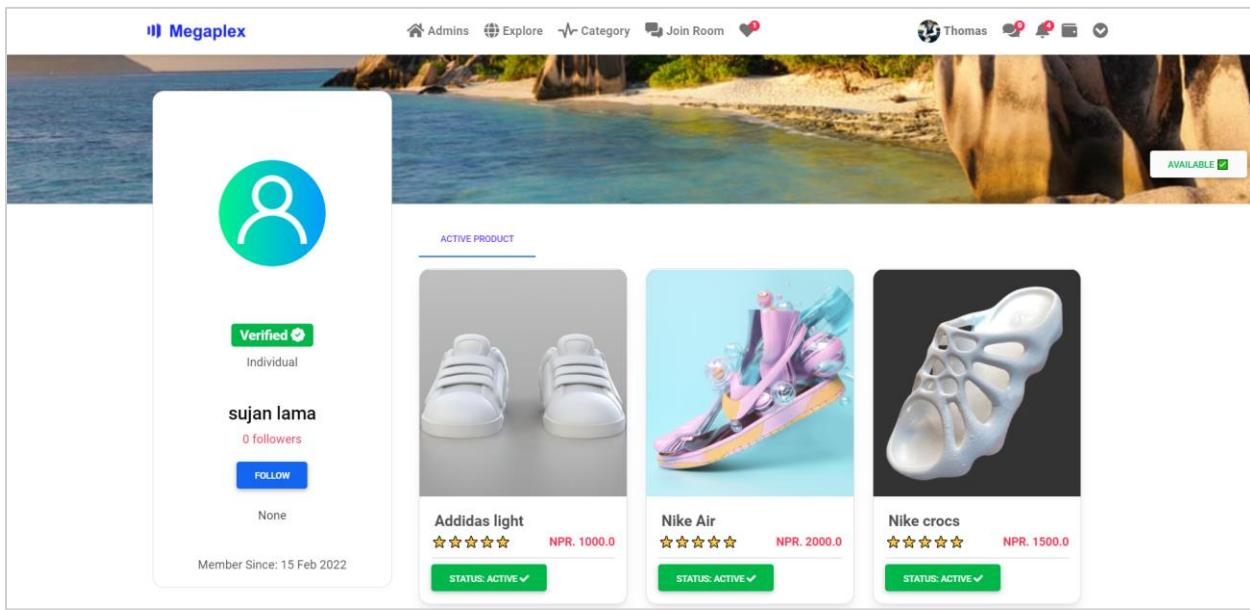


Figure 50: User Profile

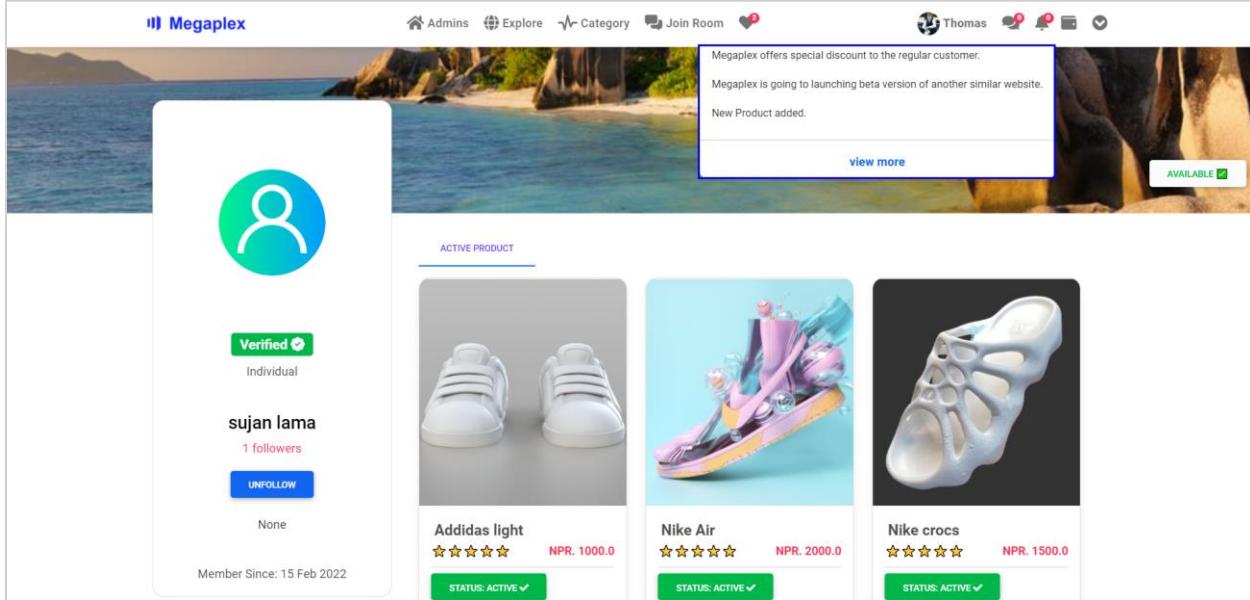


Figure 51: Megaplex Notification

The screenshot shows the 'News & Announcements' section of the Megaplex platform. It displays four notifications and one product launch entry:

- Megaplex Notification**: Admin posted on Feb. 20, 2022, at 12:55 p.m. Megaplex offers special discount to the user. Buttons: MARK AS READ, READ MORE ▾.
- Megaplex Notification**: Admin posted on Feb. 20, 2022, at 12:54 p.m. Megaplex is going to launching beta. Buttons: MARK AS READ, READ MORE ▾.
- Megaplex Notification**: Admin posted on Feb. 20, 2022, at 12:52 p.m. New Product added. Buttons: MARK AS READ, READ MORE ▾.
- Product Launch**: Admin posted on Feb. 20, 2022, at 12:50 p.m. Megaplex will be live on the internet on [redacted]. Buttons: MARK AS READ, READ MORE ▾.

Figure 52: News and announcement page

The screenshot shows the 'Following' and 'Follower' sections of the Megaplex platform:

- Following**: A list of users being followed:
 - Salil Timalsina
 - sujan lama
 - ashok shrestha
 - Chirag Simkhada
 - Upendra Bhattarai
 Each entry has a 'UNFOLLOW' button to its right.
- Follower**: A list of users who follow the current user:
 - test test
 - ashok shrestha
 - Bimal Shrestha
 Each entry has a 'REMOVE' button to its right.

Megaplex

We created the platform, that provides the opportunity to buy or sell product directly without broker commission.

SOCIAL MEDIA LINKS

- Facebook
- Instagram
- LinkedIn
- Youtube

USEFUL LINKS

- Your Account
- Become an Affiliate
- Shipping Rates
- Help

CONTACT

- Kathmandu, Nepal
- teamhmv2@gmail.com
- +977 01-984415852
- Made with Care, by Nerd Squad

2020 Copyright © Megaplex - All Right Reserved

Figure 53: User follower and the following page

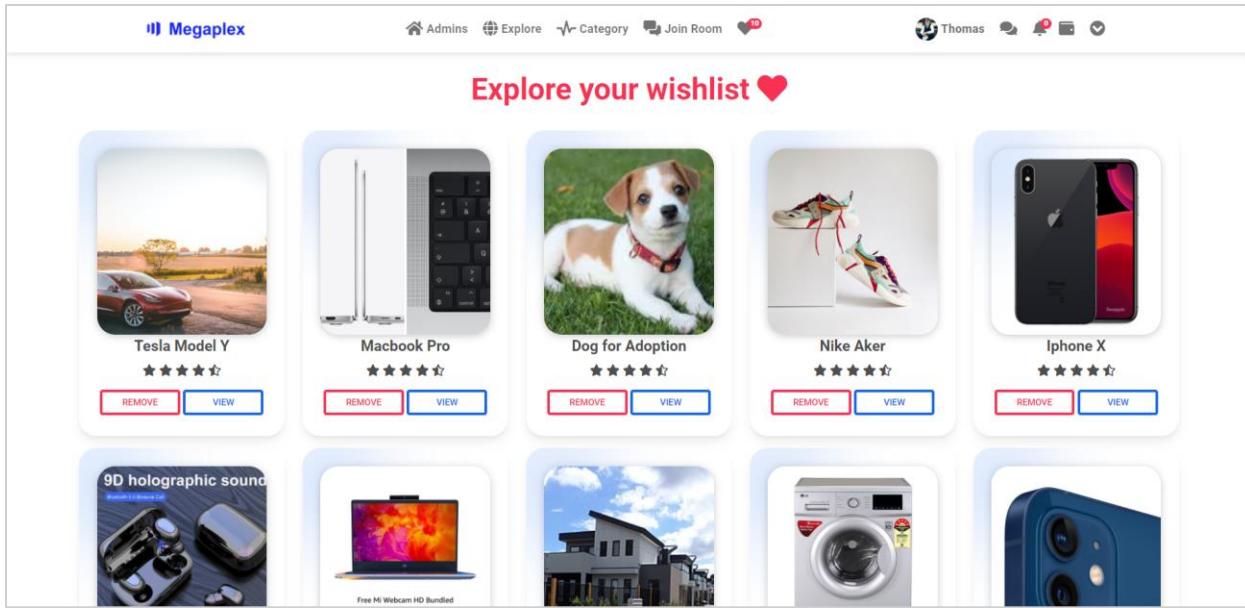


Figure 54: Wishlist page

Name	Customer type	Account	Last Access	Status
Thomas Selby bimalstha291@gmail.com	Individual City: Kathmandu	Active	Feb. 18, 2022, 1:34 p.m.	Verified
test test test@gmail.com	Individual City: Baghbazar	Active	Feb. 17, 2022, 7:25 p.m.	Unverified
Salil Timalsina salil.ragani@gmail.com	Individual City: Baghbazar	Active	Feb. 16, 2022, 4:45 p.m.	Verified
sujan lama nischalchan@gmail.com	Individual City: Baghbazar	Active	Feb. 15, 2022, 12:26 p.m.	Verified
ashok shrestha ashokshrestha102@gmail.com	Individual City: Baghbazar	Active	Feb. 17, 2022, 8:49 p.m.	Verified
Upendra Bhattarai upendra@gmail.com	Individual City: Baghbazar	Active	Feb. 17, 2022, 5:34 p.m.	Verified

Figure 55: Admin dashboard

Superuser			
	NOTIFICATION	CATEGORY	SUBCATEGORY
+ Notification			
Message		Broadcast On	Sent?
Our team members are now on holiday after megaplex have crossed 1M customer worldwide.		Feb. 22, 2022, 1:32 p.m.	False
Our team members: Chirag Simkhada, Nischal Bade, Ashok Shrestha, Sall Timalsina, Nabin Bhattarai, Adarsha Khadka, Manish silwal, Aayush Tujale, Shaswot Malla.		Feb. 22, 2022, 1:30 p.m.	False
We have a partnership for largest ecommerce platform eBay on 20FEB 2023.		Feb. 22, 2022, 1:28 p.m.	False
System upgrade notice on 25th February		Feb. 22, 2022, 1:27 p.m.	False
Don't miss the opportunity. We are launching tesla model for the first time in Nepal.		Feb. 20, 2022, 12:57 p.m.	False
Megaplex offers special discount to the regular customer.		Feb. 20, 2022, 12:55 p.m.	True
Megaplex is going to launching beta version of another similar website.		Feb. 20, 2022, 12:54 p.m.	True
New Product added.		Feb. 20, 2022, 12:52 p.m.	True

Figure 56: Page for sending notifications by Admin

Superuser			
	NOTIFICATION	CATEGORY	SUBCATEGORY
You are adding category.			
+ Category			
Category	SubCategories	No of products	Actions
 Books		0	 
 Cars		0	 
 Fashion		0	 
 Furnitures		0	 
 Adoption		0	 
 Services		0	 

Figure 57: Admin category viewing page

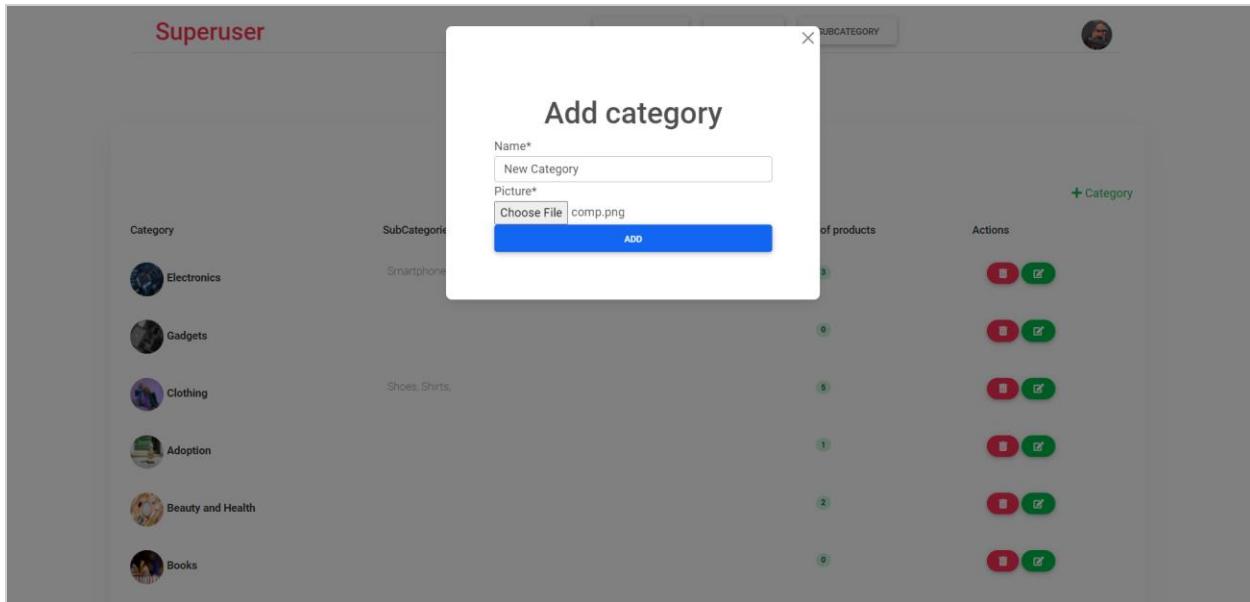


Figure 58: Admin's page for adding categories

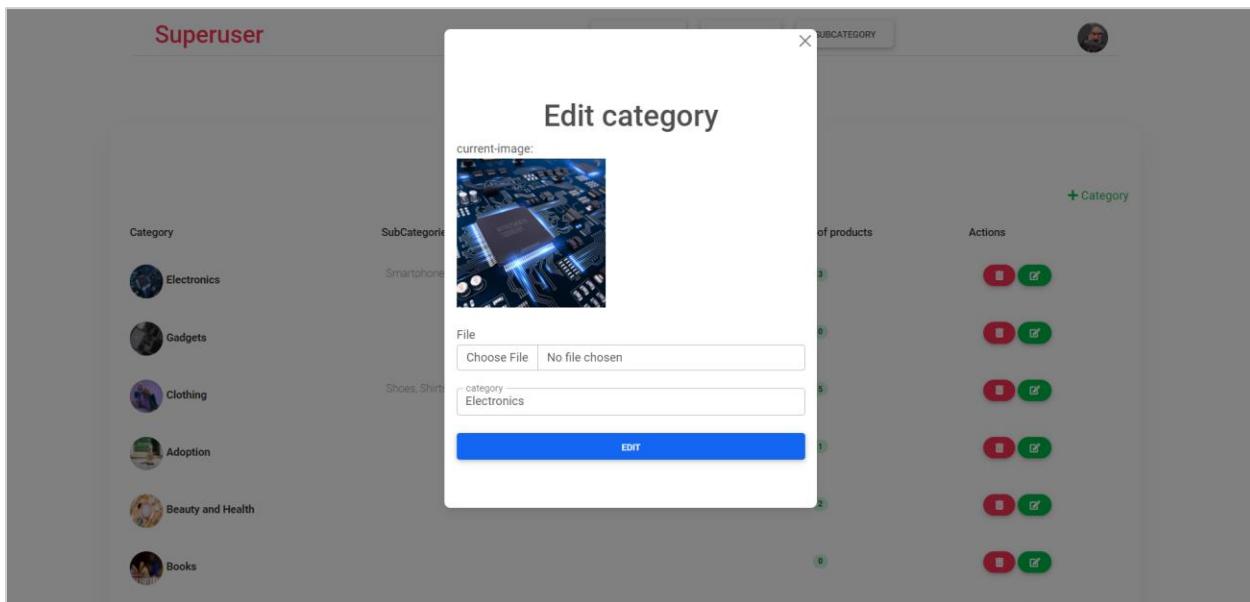


Figure 59: Admin's page for editing categories

The screenshot shows the 'Subcategories' section of the Megaplex admin interface. At the top, there are three buttons: 'NOTIFICATION', 'CATEGORY', and 'SUBCATEGORY'. A user profile icon is on the right. Below the header, the title 'Subcategories' is centered. To its right is a green button labeled '+ Subcategory'. The main content area displays a table with five rows of subcategory data:

Subcategory	Category	No of products	Actions
Smartphone	Electronics	0	Delete Edit
PC & Laptops	Electronics	0	Delete Edit
Shoes	Clothing	0	Delete Edit
Shirts	Clothing	0	Delete Edit

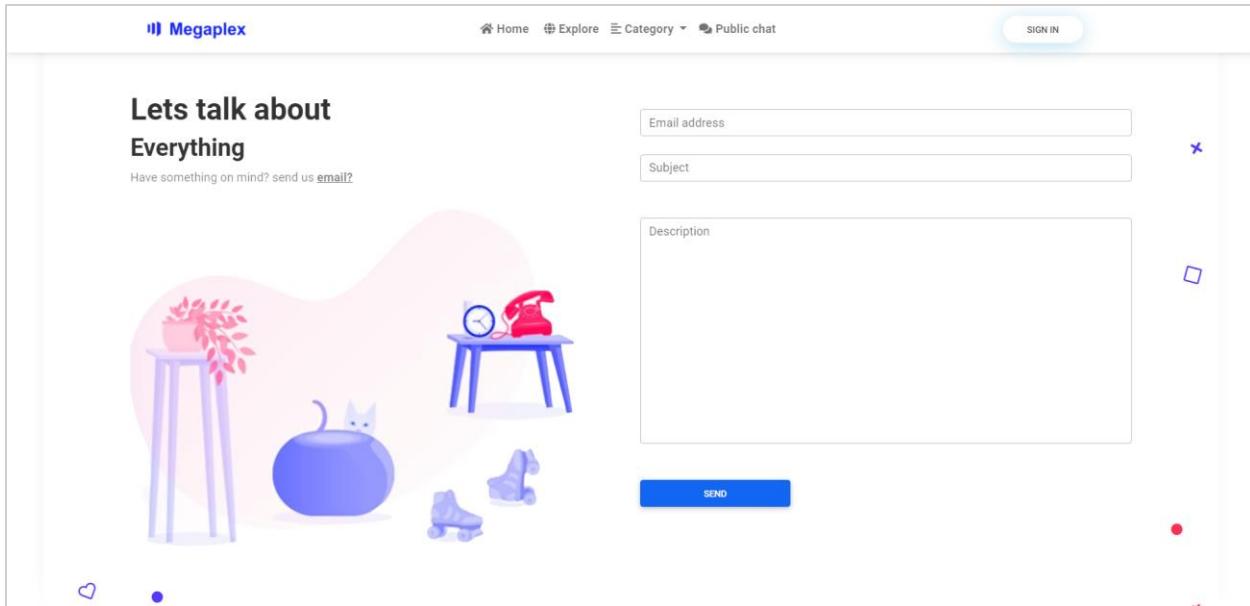
At the bottom of the page is a purple footer bar with the Megaplex logo and links to 'SOCIAL MEDIA LINKS' (Facebook), 'USEFUL LINKS' (Your Account), and 'CONTACT' (Kathmandu, Nepal).

Figure 60: Admin Subcategory viewing page

The screenshot shows the 'Add category' modal window. The title 'Add category' is at the top. It has two input fields: 'Name*' with 'New Sub Category' and 'Category*' with a dropdown menu. The dropdown menu lists various categories: Electronics, Gadgets, Clothing, Adoption, Beauty and Health, Books, Cars, Fashion, Musical Instrument, Real Estate, Services, Tours and travals, Toys, and Grocery. The background of the modal is semi-transparent, showing the 'Subcategories' table from Figure 60.

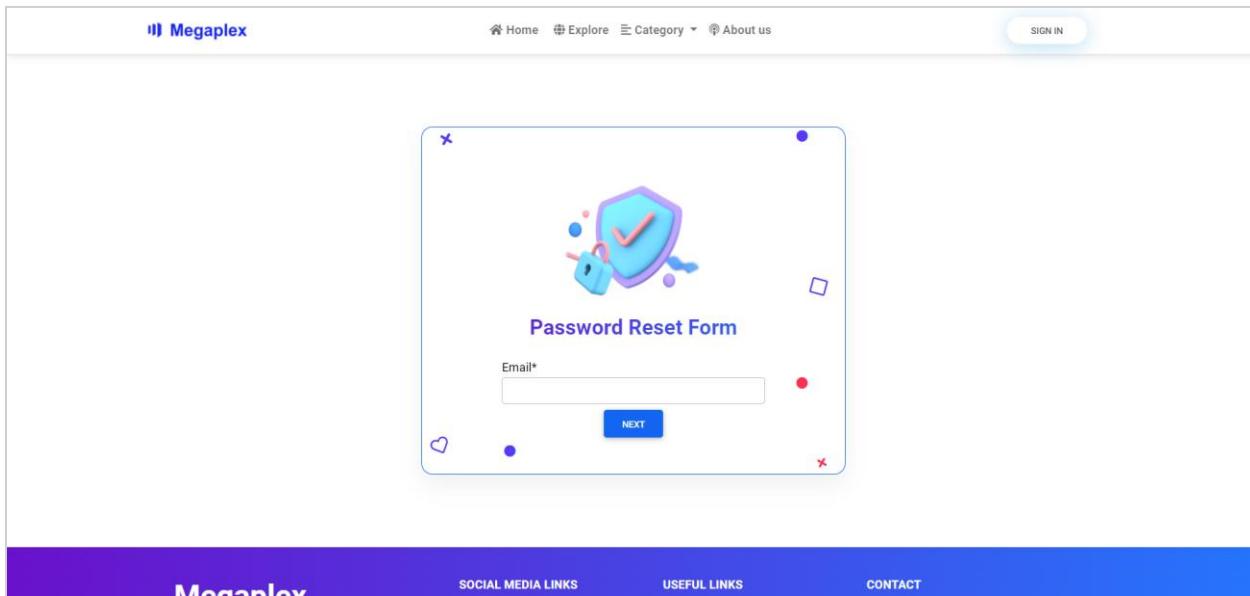
At the bottom of the page is a dark blue footer bar with the Megaplex logo and links to 'SOCIAL MEDIA LINKS' (Facebook), 'USEFUL LINKS' (Your Account), and 'CONTACT' (Kathmandu, Nepal).

Figure 61: Admin's page for adding Subcategories



The contact form is titled "Lets talk about Everything". It features a decorative background illustration of a room with a pink sofa, a blue chair, a red telephone, a purple ball, and a small cat. The form includes fields for "Email address", "Subject", and "Description", along with a "SEND" button.

Figure 62: Contact Form



The password reset form is titled "Password Reset Form". It features a decorative background illustration of a shield with a checkmark and a lock. The form includes a field for "Email*" and a "NEXT" button.

Figure 63: Password reset form

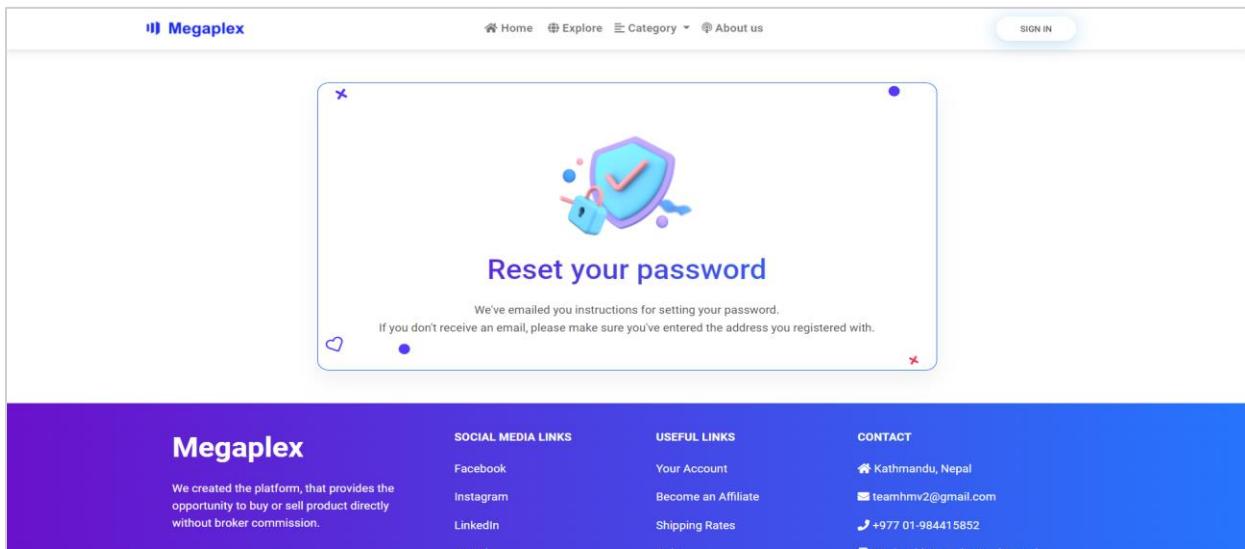


Figure 64: Password reset form(Email Sent)

The screenshot shows an email inbox with a single message from "teamhmhv2@gmail.com" to the user. The subject is "Password reset on 127.0.0.1:8000". The email body contains instructions for password reset, including a link (<http://127.0.0.1:8000/account/reset/Mw/b18z8h-a586756ca24ce3ac839deaf5016c7ee7/>), the user's username ("testyee"), and thanks from the "127.0.0.1:8000" team. At the bottom, there are "Reply" and "Forward" buttons.

Figure 65: Password reset (Email Sent)

Megaplex

New password*

Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

New password confirmation*

NEXT

SOCIAL MEDIA LINKS

- Facebook
- Instagram

USEFUL LINKS

- Your Account
- Become an Affiliate

CONTACT

- Kathmandu, Nepal
- teamhmv2@gmail.com

Figure 66: Page for entering a new password

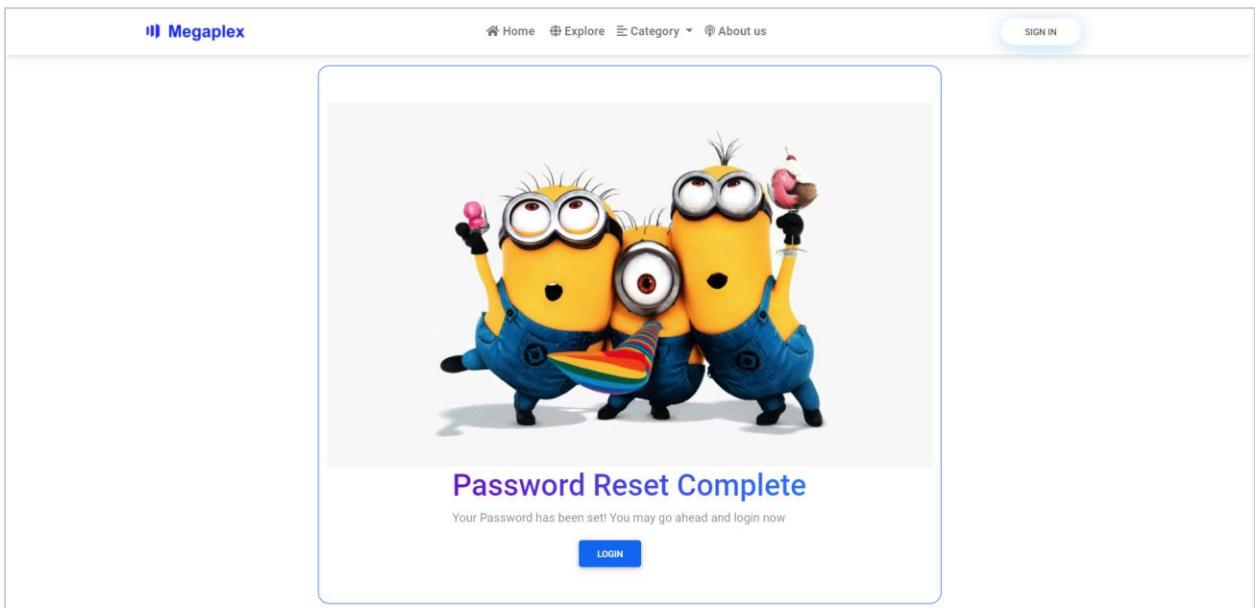


Figure 67: Password reset conformation

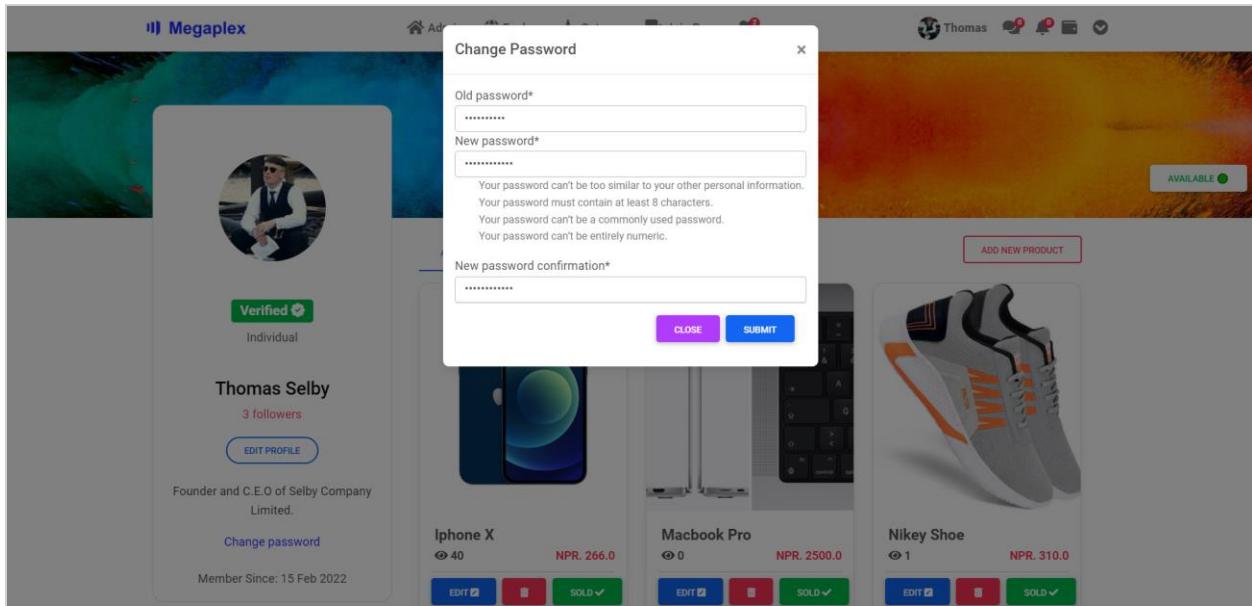


Figure 68: Password change form

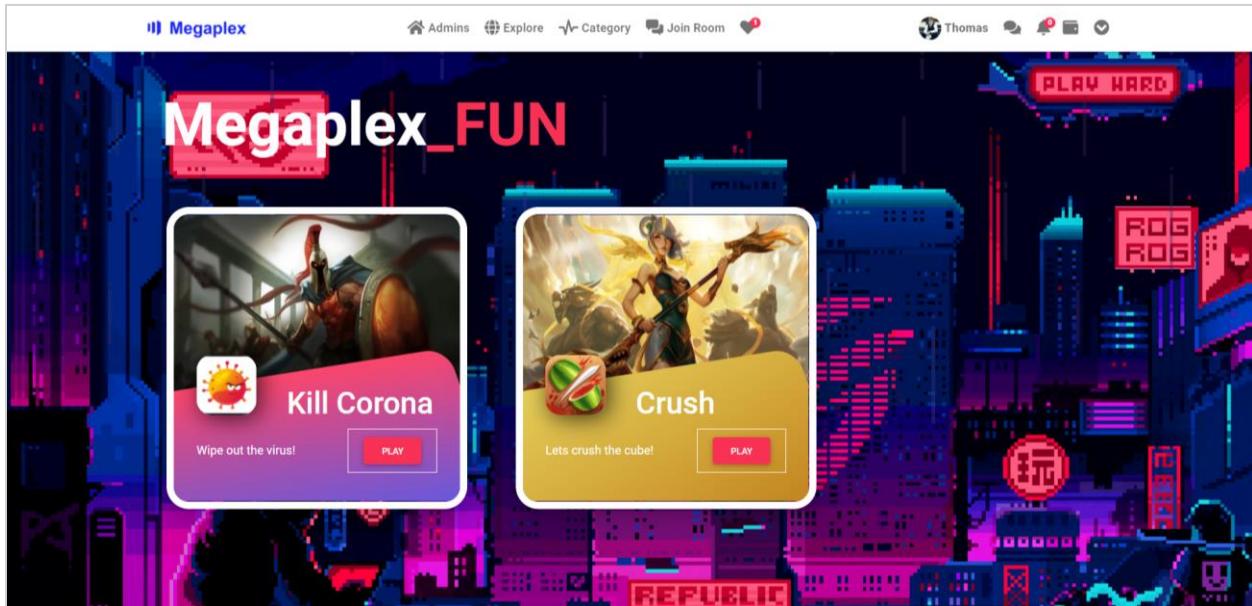


Figure 69: Game homepage

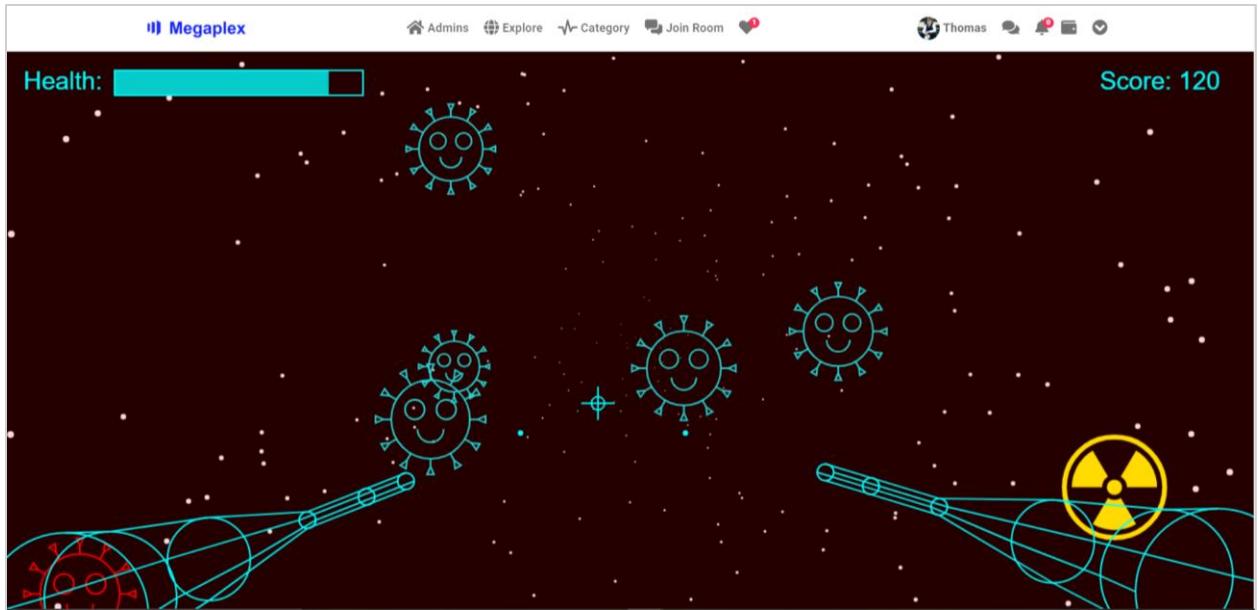


Figure 70: Game one

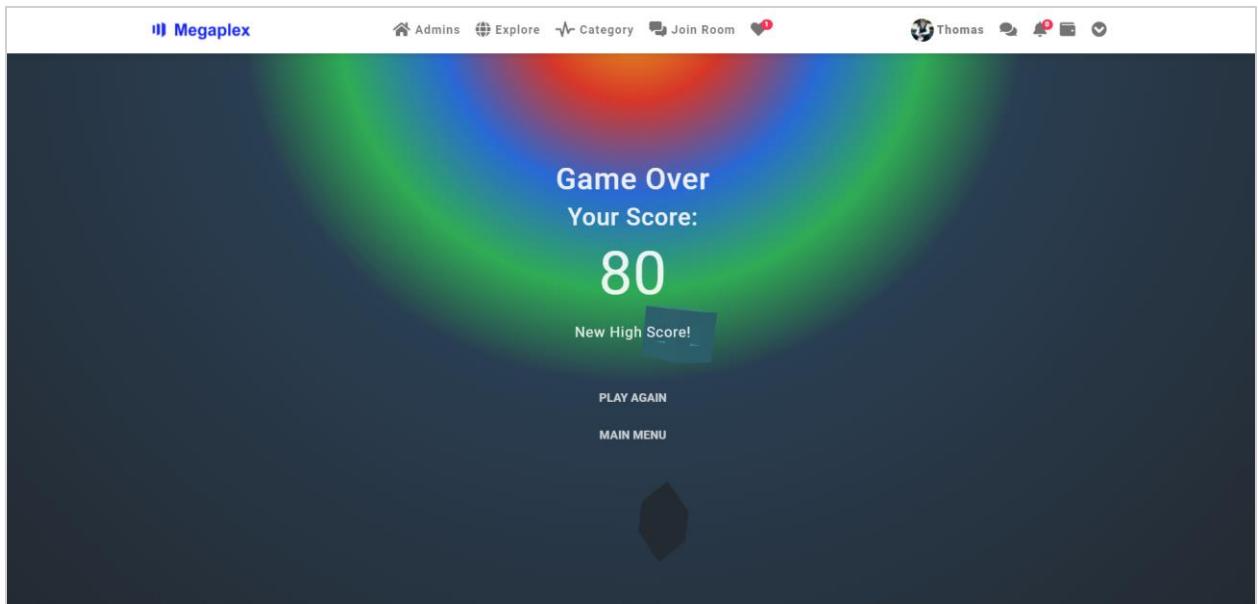


Figure 71: Game two

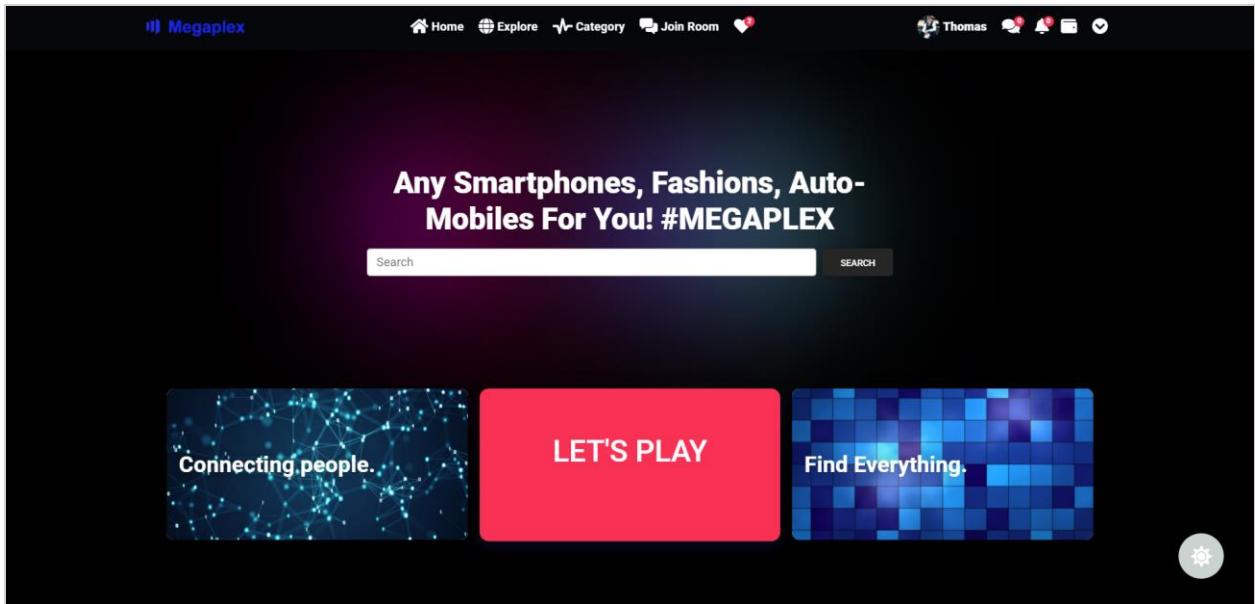


Figure 72: Dark mode homepage

The image shows the dark mode profile page for a user named Thomas Selby. On the left, there's a profile card with a circular photo of a man, a "Verified" badge, and the name "Thomas Selby" with "3 followers". It also includes a bio: "Founder and C.E.O of Selby Company Limited.", a "Change password" link, and a "Member Since: 15 Feb 2022". To the right, there are three product cards: "ACTIVE PRODUCT" showing two iPhone X phones, "SOLD PRODUCT" showing a Macbook Pro keyboard, and another "SOLD PRODUCT" showing a Nike shoe. There are also "EDIT" and "SOLD" buttons for each item. A "ADD NEW PRODUCT" button is located at the top right of the product section.

Figure 73: Dark mode profile

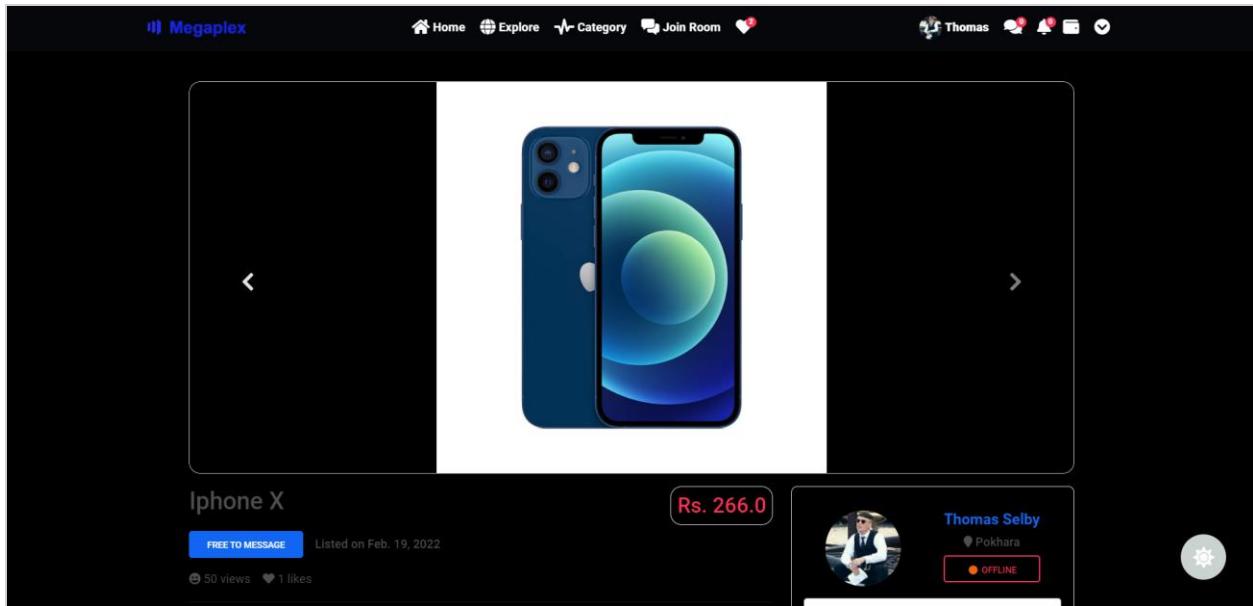


Figure 74: Dark mode product details

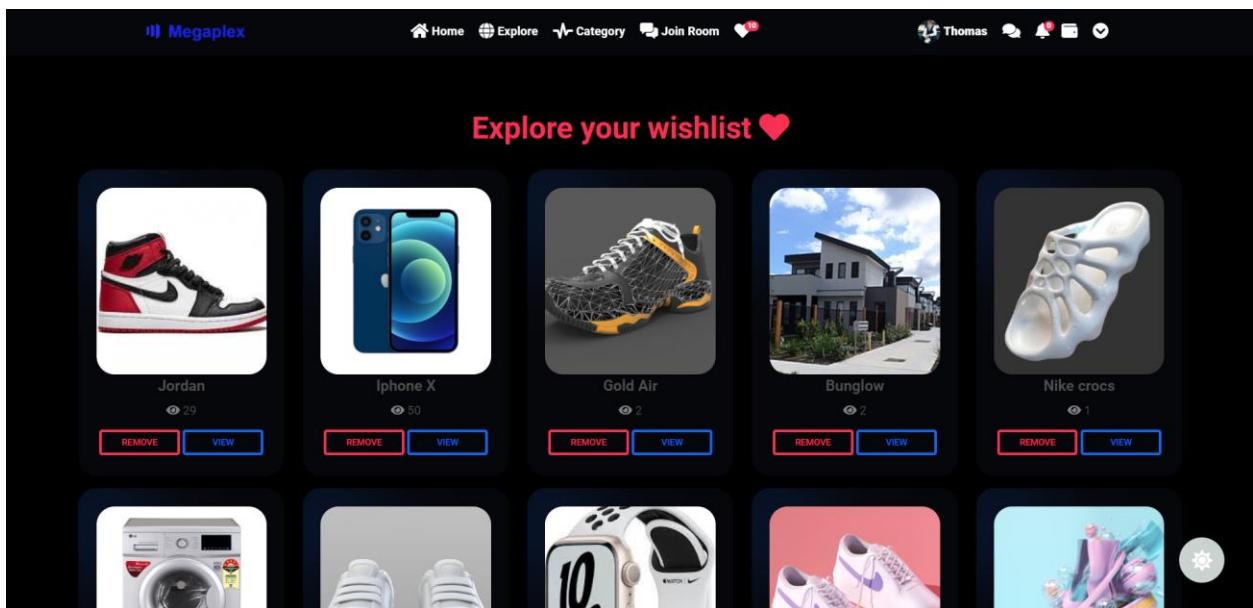


Figure 75:Dark mode Wishlist

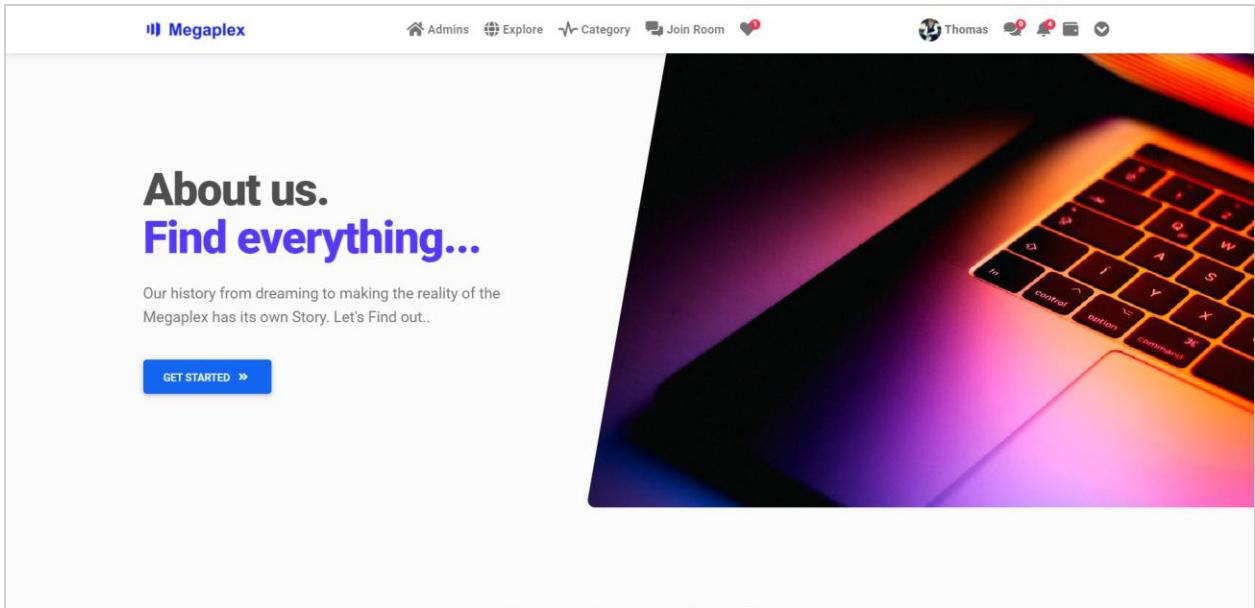


Figure 76: About us

A screenshot of the Megaplex platform showing two user profiles. The first profile on the left is for "Nischal Bade", labeled as a "FRONTEND DESIGNER". It features a colorful, abstract portrait of a person wearing glasses. Below the portrait is a brief bio: "Highly motivated Web application Developer who values innovation and hardwork. Well versed in UI/UX,HTML,CSS,Bootstrap,Django and React .Constantly Striving to improve and learn new ways to better myself in this rapidly changing industry". Two dark social media icons are shown below the bio. The second profile on the right is for "Chirag Simkhada", labeled as a "BACKEND DEVELOPER". It features a photograph of a man with a beard and glasses. Below the photo is a brief bio: "Enthusiastic, highly motivated Computer Science student passionate in problem solving. Likes to take initiative and seek out new challenges. Proficient in a range of modern Technologies including Python, MySQL, NoSQL, Java, Django , DevOPS". Two dark social media icons are shown below the bio.

Figure 77: About Us page 1

The screenshot shows the 'About Us' section of the Megaplex website. At the top, there's a navigation bar with links for 'Admins', 'Explore', 'Category', 'Join Room', and a user icon for 'Thomas'. Below the navigation, there are two profile cards.

Ashok Shrestha
API HANDLER
Prepare a short script that emphasizes my skills, talents, and areas of expertise that are significant to me as a front-end developer, stressing career-related objectives such as a desire to gain experience and take on more responsibilities as a front-end developer.

Salil Timalsina
JS GAME DEVELOPER
Acquiring new knowledge, Improving myself and striving to get the most out of myself as a front end developer and gaining responsibilities, abilities, and experience

Each profile includes a photo, a title, a brief bio, and social media links for LinkedIn and GitHub.

Figure 78: About Us page 2

The screenshot shows the 'Frameworks used' section of the Megaplex website. The background features a purple and white abstract design. The section title is 'Frameworks used >' followed by 'Megaplex'.

Framework	Technology
Django	Python
Theme	Bootstrap
Repository	Github
JS	JavaScript

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.

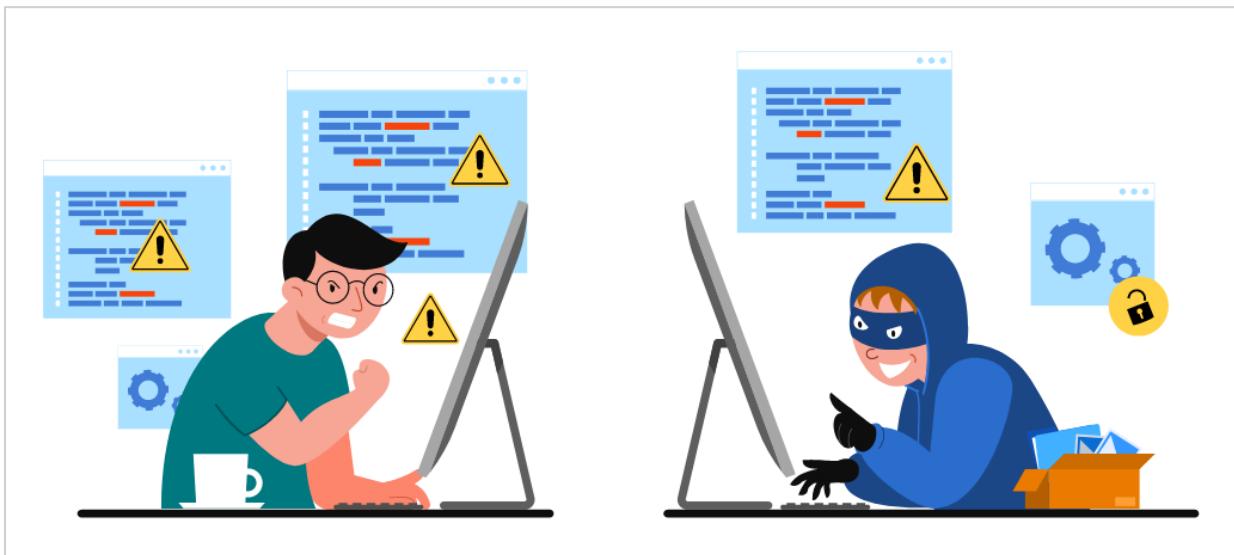


Figure 82: Possible Frauds

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 > ...
58 MIDDLEWARE = [
59     'django.middleware.security.SecurityMiddleware',
60     'django.contrib.sessions.middleware.SessionMiddleware',
61     'django.middleware.common.CommonMiddleware',
62     'django.middleware.csrf.CsrfViewMiddleware',
63     'django.contrib.auth.middleware.AuthenticationMiddleware',
64     'django.contrib.messages.middleware.MessageMiddleware',
65     'django.middleware.clickjacking.XFrameOptionsMiddleware',
66 ]
67
68 ROOT_URLCONF = 'Megaplex.urls'
69
70 TEMPLATES = [
71     {
72         'BACKEND': 'django.template.backends.django.DjangoTemplates',
73         'DIRS': [os.path.join(BASE_DIR, 'templates')],
74         'APP_DIRS': True,
75         'OPTIONS': {
76             'context_processors': [
77                 'django.template.context_processors.debug',
78                 'django.template.context_processors.request',
79                 'django.contrib.auth.context_processors.auth',
80                 'django.contrib.messages.context_processors.messages',
81                 'homepage.custom_context_processors.notifications',
82             ],
83         },
84     },
85 ]
86
87 WSGI_APPLICATION = 'Megaplex.wsgi.application'
88 ASGI_APPLICATION = 'Megaplex.asgi.application'
89
90
91 CHANNEL_LAYERS = {
92     "default": {
93         "BACKEND": "channels_redis.core.RedisChannelLayer",
94         "CONFIG": {
95             "hosts": [{"localhost", 6379}],
96         },
97     },
98 }
```

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
1
2   from __future__ import absolute_import, unicode_literals
3   import os
4
5   from celery import Celery
6   from django.conf import settings
7   from celery.schedules import crontab
8   from datetime import datetime, timedelta
9
10 # set the default Django settings module for the 'celery' program.
11 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Megaplex.settings')
12
13 app = Celery('Megaplex')
14 app.conf.enable_utc = False
15 app.conf.update(timezone='Asia/Kathmandu')
16
17 app.config_from_object(settings, namespace='CELERY')
18
19 app.conf.beat_schedule = {
20 }
21
22 #app.conf.timezone = 'UTC'
23
24 # Load task modules from all registered Django app configs.
25 app.autodiscover_tasks()
26
27
28 @app.task(bind=True)
29 def debug_task(self):
30     print(f'Request: {self.request!r}')

```

```

Megaplex > 🗂️ urls.py > ...
1
2   from django.contrib import admin
3   from django.urls import path
4   from django.conf.urls import include
5   from django.conf.urls.static import static
6   from django.conf import settings
7
8   urlpatterns = [
9       path('admin/', admin.site.urls),
10      path('', include('homepage.urls')),
11      path('account/', include('account.urls')),
12      path('chat/', include('directChat.urls')),
13      path('dashboard/', include('dashboard.urls')),
14      path('profile/', include('dashboard.urls')),
15      path('admins/', include('admins.urls')),
16      path('product/', include('product.urls')),
17      path('profile/', include('dashboard.urls')),
18      path('room/', include('roomChat.urls')),
19  ]
20
21  if settings.DEBUG:
22      urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
23      urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

```

templates > <> layout.html > <> html > <> head > <> script
  1  {% load static %}
  2  !DOCTYPE html
  3  <html lang="en">
  4
  5  <head>
  6    <meta charset="utf-8" />
  7    <meta name="viewport" content="width=device-width, initial-scale=1" />
  8    {% block title %} {{ section.title }} {% endblock title %}
  9    {% block css_files %} {% endblock css_files %}
10
11    <!-- =====bootstrap-cdn===== -->
12    <!-- Font Awesome -->
13    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css" rel="stylesheet" />
14    <!-- Google Fonts -->
15    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" rel="stylesheet" />
16    <!-- MDB -->
17    <link href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.css" rel="stylesheet" />
18    <!-- =====fontawesome-cdn===== -->
19    <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
20      | integrity="sha384-AYmEC3Yw5cVb3ZcuHt0A93w35dYTsvhLPvNys9eStHFGJvOvKxVfELGroGkvsg+p" crossorigin="anonymous" />
21    <link rel="stylesheet" href="{% static 'css/homepage.css' %}">
22    <link rel="stylesheet" href="{% static 'css/navbar.css' %}">
23    <link rel="stylesheet" href="{% static 'css/main.css' %}">
24    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
25    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENmM7KCKrR/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5I" crossorigin="anonymous">
26    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+YQKtv3Rn7W3mgPxhU9K/ScQsAP" crossorigin="anonymous">
27    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8j0t6vLEHfe/JQGiRR5QQxSffWpi" crossorigin="anonymous">
28  </head>
29
30  <body>
31    {% include 'navbar.html' %}
32    <div class="main">% block main_content %} {% endblock %}</div>
33    {% block games %} {% endblock %}>
34    {% block js_files %} {% endblock js_files %}>
35    {% include 'footer.html' %}
36    <script src="{% static 'javascript/theme-toggle.js' %}"></script>
37    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous">
38    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.js"></script>
39    <script src="{% static 'javascript/onscroll.js' %}"></script>
40

```

Figure 86: Code for Layouts

```

templates > admin.layout.html > html > body > script > icon
1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="utf-8" />
6      <meta name="viewport" content="width=device-width, initial-scale=1" />
7      {% block title %} {{ section.title }} {% endblock title %}
8      {% block css_files %} {% endblock css_files %}
9      <!-- =====bootstrap-cdn===== -->
10     <!-- Font Awesome -->
11     <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css" rel="stylesheet" />
12     <!-- Google Fonts -->
13     <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" rel="stylesheet" />
14     <!-- MDB -->
15     <link href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.css" rel="stylesheet" />
16     <!-- =====fontawesome-cdn===== -->
17     <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
|      integrity="sha384-AYmEC3YU5cvBz2cuHtOA9w35dYtsvhLPvhs9eStHfgJvOvKxVfELGroGkvsg+p" crossorigin="anonymous" />
18     <link rel="stylesheet" href="{{% static 'css/homepage.css' %}}>
19     <link rel="stylesheet" href="{{% static 'css/navbar.css' %}}>
20     <link rel="stylesheet" href="{{% static 'css/main.css' %}}>
21     <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
22         rel="stylesheet">
23     <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js" integrity="sha256-u7e5khyithlIdTpU22PHhENmPcRdFiHRjhAuHcs05RI=" crossorigin="anonymous" />
24     <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js" integrity="sha256-u7e5khyithlIdTpU22PHhENmPcRdFiHRjhAuHcs05RI=" crossorigin="anonymous" />
25   </head>
26
27 <body>
28     {% include 'admin-navbar.html' %}
29     <div class="main">{% block main_content %} {% endblock %}</div>
30     {% block js_files %} {% endblock js_files %}
31     {% include 'footer.html' %}
32     <!-- MDB -->
33     <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/3.10.1/mdb.min.js"></script>
34     <script src="{{% static 'javascript/theme-toggle.js' %}}></script>
35     <script src="{{% static 'javascript/onscroll.js' %}}></script>
36     <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
37     {% if messages %}
38         {% for m in messages %}
39             {% if m.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
40                 <script>
41                     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-stretch
1  {% load static %}
2
3  <nav class="navbar navbar-expand-lg navbar-light sticky-top" style="background-color: #fff;">
4      <div class="container">
5          <a class="navbar-brand" href="/" style="width: 10%">
6              
7          </a>
8          <button class="navbar-toggler" type="button" data-mdb-toggle="collapse" data-mdb-target="#navbarButtonsExample"
9                  aria-controls="navbarButtonsExample" aria-expanded="false" aria-label="Toggle navigation">
10             <i class="fas fa-bars"></i>
11         </button>
12         <div class="collapse navbar-collapse" id="navbarButtonsExample">
13             <ul class="navbar-nav mx-auto mb-2 mb-lg-0">
14                 {% if request.user.is_authenticated %}
15                     <ul class="navbar-nav flex-row d-none d-md-flex">
16                         <li class="nav-item me-3 me-lg-1 active ">
17                             <a class="nav-link" href="/admins/dashboard">
18                                 <span><i class="fas fa-home fa-lg"></i></span>
19                                 Admins
20                             </a>
21                         </li>
22
23                         <li class="nav-item me-3 me-lg-1">
24                             <a class="nav-link" href="/product/explore">
25                                 <span><i class="fas fa-globe fa-lg"></i></span>
26                                 Explore
27                             </a>
28                         </li>
29
30                         <li class="nav-item me-3 me-lg-1">
31                             <a class="nav-link" href="/category">
32                                 <span><i class="fas fa-heart-rate fa-lg"></i></span>
33                                 Category
34                             </a>
35                         </li>

```

Figure 87: Accounts

```

account > forms.py > PartialProfileForm > Meta
 1  from django.core import validators
 2  from .models import Profile
 3  from django.contrib.auth.models import User
 4  from django.contrib.auth.forms import UserChangeForm, UserCreationForm
 5  from django import forms
 6  from django.forms import ModelForm
 7
 8  USER_CHOICES = [
 9      ("Individual", "Individual"),
10      ("Organization", "Organization")
11  ]
12
13 class UserForm(UserCreationForm):
14     """Form for creating Users"""
15     email = forms.EmailField()
16     plan = forms.CharField(label="Choose the type of membership", widget=forms.Select(choices=USER_CHOICES))
17     class Meta:
18         model = User
19         fields = ['first_name', 'last_name', 'email', 'username', 'plan', 'password1', 'password2']
20
21
22 class LoginForm(forms.Form):
23     """Form to allow users to log in"""
24     username = forms.CharField(validators=[validators.MinLengthValidator(3)])
25     password = forms.CharField(widget=forms.PasswordInput)
26
27
28 class ProfileForm(ModelForm):
29     """Form for creating User profile"""
30     desc = forms.CharField(
31         label='desc',
32         widget=forms.Textarea(attrs={
33             'rows': '3',
34             'placeholder': ''
35         }))
36     class Meta:
37         model = Profile
38         fields = "__all__"
39         exclude = ['user', 'email', 'username']
40

```

Figure 88: Forms

```

account > 🏠 views.py > ✎ logout_user
 1  from django.contrib.auth import logout, login, authenticate
 2  from django.shortcuts import redirect, render
 3  from .forms import ProfileForm, UserForm
 4  from django.contrib import messages, auth
 5  from .models import Profile
 6  from django.contrib import messages
 7  from django.contrib.auth.models import User
 8  from directChat.views import get_unread
 9
10
11 def login(request):
12     if request.method == 'POST':
13         uname = request.POST['username']
14         passwd = request.POST['password']
15
16         user = auth.authenticate(username=uname, password=passwd)
17
18         if user is not None:
19             if not user.is_staff:
20                 auth.login(request, user)
21                 messages.success(request, "Welcome to Megaplex World")
22                 return redirect("/")
23
24             elif user.is_staff:
25                 auth.login(request, user)
26                 return redirect('/admin')
27
28         else:
29             messages.add_message(request, messages.ERROR, "Invalid Username and Password!")
30             return render(request, 'account/login.html',{'room_name':'broadcast','get_unread':get_unread(request)})
31
32     else:
33         return render(request, 'account/login.html',{'room_name':'broadcast','get_unread':get_unread(request)})
34
35
36
37 def register(request):
38     form = UserForm()
39     if request.method == "POST":
40         form = UserForm(request.POST)
41         if form.is_valid():

```

```

13 class Profile(models.Model): #Model to create profile for users
14     user = models.OneToOneField(User,null=True,on_delete=models.CASCADE,related_name='profile')
15     username = models.CharField(max_length=20)
16     firstname = models.CharField(max_length=50)
17     lastname = models.CharField(max_length=50)
18     email = models.EmailField()
19     phone = models.CharField(max_length=10)
20     plan = models.CharField(max_length=20)
21     district = models.CharField(max_length=30, default="Kathmandu")
22     city = models.CharField(max_length=30, default="Baghbazar")
23     profile_pic = models.FileField(upload_to='profiles', default='default.png')
24     created_date = models.DateTimeField(auto_now_add=True)
25     followers = models.ManyToManyField(User,blank=True,related_name='followers',default=[0])
26     following = models.ManyToManyField(User,blank=True ,related_name = 'following',default=[0])
27     cover_pic = models.FileField(upload_to='covers', default='cover.jpg', blank=True, null=True)
28     verified = models.BooleanField(default=False)
29     desc = models.TextField(max_length=10000, blank=True, null=True)
30     product_sold = models.IntegerField(default=0, blank=True, null=True)
31     active = models.BooleanField(default=True, blank=True, null=True)
32     balance = models.DecimalField(
33         default=50,
34         max_digits=12,
35         decimal_places=2
36     )
37     def __str__(self):
38         return self.firstname + " " + self.lastname
39
40     def txn():
41         return str(random.randint(1000000, 9999999))
42
43 class transaction(models.Model):
44     txn_id = models.IntegerField(default=txnid)
45     # sender = models.CharField(max_length=50)
46     sender = models.ForeignKey(User, on_delete=models.DO_NOTHING, related_name="sender")
47     # receiver = models.CharField(max_length=50)
48     receiver = models.ForeignKey(User, on_delete=models.DO_NOTHING, related_name="receiver")
49     amount = models.DecimalField(default=0,max_digits=11,decimal_places=2)

```

```

account > ✎ urls.py > ...
1  from django.contrib.auth import login
2  from django.contrib.auth import views as auth_views
3  from django.urls import path
4  from . import views
5
6  #Account's URL
7  urlpatterns = [
8      path('register/',views.register, name="register_user"),
9      path('login/', views.login, name="login_user"),
10     path('logout/', views.logout_user),
11     path('password_reset/', auth_views.PasswordResetView.as_view(template_name="account/password_reset_form.html"), name="password_rese
12     path('password_reset_sent/', auth_views.PasswordResetDoneView.as_view(template_name="account/password_reset_sent.html"), name="pass
13     path('reset/<uidb64>/<token>/', auth_views.PasswordResetConfirmView.as_view(template_name="account/password_reset_confirm.html"), n
14     path('reset/done', auth_views.PasswordResetCompleteView.as_view(template_name="account/password_reset_complete.html"), name="passwo
15 ]

```

```

account > templates > account > login.html > ...
1  {% extends 'layout.html' %}
2  {% load static %}
3
4  {% block title %}<title>Login to Megaplex</title>{% endblock title %}
5
6  {% block main_content %}
7
8    <div class="container col-xl-10 col-xxl-8 px-4 py-5">
9      <div class="row align-items-center g-lg-5 py-5">
10        <div class="col-lg-7 text-center text-lg-start">
11          <h2 class="fs-1 lh-1 mb-3">Login to</h2>
12          <h1 class="display-4 fw-bold lh-1 mb-3 "><span class="M-logo">Megaplex World!</span> </h1>
13          
14        </div>
15        <div class="col-md-10 mx-auto col-lg-5 " >
16          <form class="p-4 p-md-5 shadow bg-white" action="#" method="POST" style="border-radius: 15px">
17            {% csrf_token %}
18            
19            <div class="form-floating mb-3">
20              <input type="text" class="form-control" id="floatingInput" placeholder="Username" name="username">
21              <label for="floatingInput"> Username</label>
22            </div>
23            <div class="form-floating mb-3">
24              <input type="password" class="form-control" id="floatingPassword" placeholder="Password" name="password">
25              <label for="floatingPassword"> Password</label>
26              <a href="/account/password_reset/">Forget your passcode?</a>
27            </div>
28            <div class="checkbox mb-3">
29              <label>
30                <input type="checkbox" value="remember-me"> Remember me
31              </label>
32            </div>
33            <button class="w-100 btn btn-primary" type="submit">Login</button>
34            <hr class="my-4">
35            <small class="text-muted">Dont have account bro?</small> <a href="/account/register">Create here</a>
36          </form>
37        </div>
38      </div>
39    </div>
40
41  {% endblock main_content %}

```

```

dashboard > views.py > User_Profile
13  from .forms import ProductForm, txnForm
14  from django.contrib import messages
15  from django.contrib.auth import update_session_auth_hash
16  from directChat.views import get_unread
17  from django.core.paginator import Paginator
18  from django.contrib.auth.models import User
19  from django.db.models import Q
20  from notification.models import BroadcastNotification
21
22  # Create your views here.
23  @login_required
24  def profile(request):
25      user = Profile.objects.get(user=request.user)
26      form = ProfileForm(instance=user)
27
28      pform = PasswordChangeForm(user=request.user)
29      try:
30          active_products = Product.objects.filter(seller=request.user, is_active=True)
31      except:
32          active_products = None
33      try:
34          inactive_products = Product.objects.filter(seller=request.user, is_active=False)
35      except:
36          inactive_products = None
37      context={
38          'room_name':'broadcast',
39          'profile':user,
40          'active_products':active_products,
41          'inactive_products':inactive_products,

```

Figure 89: Important code of dashboard passcode for Dashboard

```

dashboard > views.py > ...
158 @login_required
159 def AddFollowers(request,profile_id):
160     logged_in_user = request.user.profile.id
161     profile = Profile.objects.get(id = profile_id)
162     user_profile = Profile.objects.get(id = logged_in_user)
163     user_id = profile.user
164     user_profile_id = user_profile.user
165     profile.followers.add(user_profile_id)
166     user_profile.following.add(user_id)
167     return redirect('/dashboard/profile/{}'.format(profile.id))
168
169
170 #when user click un-follow to un-follow other user the user will be remove and no longer be follower
171 #the other user will be removed and won't be following the other user
172 #if is following
173 @login_required
174 def RemoveFollowers(request,profile_id):
175     logged_in_user = request.user.profile.id
176     profile = Profile.objects.get(id = profile_id)
177     user_profile = Profile.objects.get(id = logged_in_user)
178     user_id = profile.user
179     user_profile_id = user_profile.user
180     profile.followers.remove(user_profile_id)
181     user_profile.following.remove(user_id)
182     return redirect('/dashboard/profile/{}'.format(profile.id))
183
184
185 @login_required
186 def follower_list(request,profile_id):
187     profile = Profile.objects.get(id = profile_id)
188     user_profile = Profile.objects.filter()
189     user = profile.user
190     followings=profile.following.all()
191     followers=profile.followers.all()
192     context = {
193         'user':user,
194         'profile':profile,
195         'user_profiles':user_profile,
196         'followings':followings,
197         'followers':followers,
198         'room_name':'broadcast'

```

```

dashboard > urls.py > ...
1  from django.contrib.auth import views as auth_views
2  from django.urls import path
3
4  from . import views
5
6  urlpatterns = [
7      path('profile/', views.profile, name='profile'),
8      path('profile/int:profile_id',views.User_Profile,name='user_profile'),
9      path('follow_user/int:profile_id',views.AddFollowers,name ='follow'),
10     path('unfollow_user/int:profile_id',views.RemoveFollowers,name='unfollow'),
11     path('follower_list/int:profile_id',views.follower_list,name='following_list'),
12     path('remove_follower/int:follower_id',views.Delete_Follower,name='Remove_following_user'),
13     path('toggle_following/int:following_id',views.togglefollowing,name='toggle_following_user'),
14     path('addProduct', views.addProduct, name="addProduct"),
15     path('editProduct/int:product_id', views.editProduct, name="editProduct"),
16     path('removeProduct/int:product_id', views.remove_product, name="removeProduct"),
17     path('wishlist/',views.wishlist,name='wishlist'),
18     path('wallet/',views.wallet,name='wallet'),
19     path('mark_sold/int:product_id',views.mark_sold),
20     path('mark_unsold/int:product_id',views.mark_unsold),
21     path('set_online/int:id',views.set_online),
22     path('set_offline/int:id',views.set_offline),
23     path('notification',views.user_notification)
24 ]

```

```

dashboard > forms.py > ...
1  from product.models import Product
2  from django.forms import ModelForm
3  from django import forms
4  from account.models import transaction
5
6  class ProductForm(ModelForm):
7      """Form for creating Product"""
8
9      description = forms.CharField(
10         label='description',
11         widget=forms.Textarea(attrs={
12             'rows':3,
13             'placeholder':''
14         }))
15
16
17      specifications = forms.CharField(
18         label='specifications',
19         widget=forms.Textarea(attrs={
20             'rows':3,
21             'placeholder':''
22         }))
23
24  class Meta:
25      model = Product
26      exclude = ['is_active','view_count','product_likes']
27
28  class txnForm(ModelForm):
29      class Meta:
30          model = transaction
31          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
1  {% extends 'layout.html' %} {% load static %} {% load crispy_forms_tags %} {% block title %}
2  <title>{{user.first_name}}'s Profile</title>
3  {% endblock title %} {% block css_files %}
4
5  <link rel="stylesheet" href="{% static '/css/main.css' %}" />
6  <link rel="stylesheet" href="{% static '/css/homepage.css' %}" />
7  <link rel="stylesheet" href="{% static '/css/profile.css' %}" />
8
9  {% endblock css_files %} {% block main_content %}
10
11 <div class="profile-page position-relative">
12     <div class="cover position-relative">
13         
14         <div class="edit-profile-cover d-flex position-absolute theme-color gap-2">
15             {% if user.profile.active %}
16                 <button type="button" class="btn btn-light text-success" data-mdb-ripple-color="dark"><a class="unlink"
17                     href="/dashboard/set_offline/{{user.id}}">Available ●</a></button>
18             {% else %}
19                 <button type="button" class="btn btn-light text-danger" data-mdb-ripple-color="dark"><a class="unlink"
20                     href="/dashboard/set_online/{{user.id}}"> Offline ✘ </a></button>
21             {% endif %}
22         </div>
23     </div>
24
25     <div class="profile container-sm row m-auto fixed">
26         <div class="profile-info d-flex flex-column text-center col-lg-3 border-15 border p-3 bg_fix ">
27
28             <span class="m-auto w-50"></span>
30             {% if user.profile.verified %}
31                 <h4><span class="badge bg-success">Verified <i class="fa fa-badge-check"></i></span></h4>
32             {% else %}
33                 <h4><span class="badge bg-danger">Unverified <i class="fa fa-badge-cross"></i></span></h4>
34             {% endif %}
35             <p>{{profile.plan}}</p>
36             <h4 class="pt-4 theme-color">{{profile.firstname}} {{profile.lastname}}</h4>
37             <a href="/dashboard/follower_list/{{profile.id}}" class="text-danger">{{profile.followers.all.count}}
38                         followers</a>
39
40             <div class="mt-3">
41                 {% if request.user == user %}
42                     <a data-toggle="modal" data-target="#exampleModal" class="text-decoration-none"><button

```

```

nd-9.col.col-lg-8.col-xl-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;">
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 > ...
1  from django.http.response import HttpResponseRedirect
2  from django.shortcuts import redirect
3  from directChat.models import Chat_Message
4  from django.contrib.auth.decorators import login_required
5  from django.template import loader
6  from django.contrib.auth.models import User
7  from django.urls import reverse
8
9  # @login_required
10 def Inbox(request):
11     messages = Chat_Message.get_messages(user=request.user)
12     active_direct = None
13     directs = None
14     if messages:
15         message = messages[0]
16         active_direct = message['user'].username
17         directs = Chat_Message.objects.filter(user=request.user, recipient=message['user'])
18         directs.update(is_read=True)
19         for message in messages:
20             if message['user'].username == active_direct:
21                 message['unread'] = 0
22     context = {
23         'room_name': "broadcast",
24         'directs': directs,
25         'messages': messages,
26         'active_direct': active_direct,
27         'get_unread': get_unread(request)
28     }
29     template = loader.get_template('directChat/direct.html')
30
31     return HttpResponseRedirect(template.render(context, request))
32
33
34 def get_name(username):
35     try:
36         user = User.objects.get(id=username)
37     except:
38         user = User.objects.get(username=username)
39     full_name = user.first_name + " " + user.last_name
40     return full_name

```

```

44  @login_required
45  def Directs(request, username):
46      user = request.user.id
47      current_user = Chat_Message.objects.all()
48      messages = Chat_Message.get_messages(user=user)
49      active_direct = username
50      try:
51          directs = Chat_Message.objects.filter(user=user, recipient=username)
52      except:
53          directs = Chat_Message.objects.filter(user=user, recipient=user)
54
55      directs.update(is_read=True)
56
57      for message in messages:
58          if message['user'].username == username:
59              message['unread'] = 0
60          name = get_name(username)
61          context = {
62              'room_name': "broadcast",
63              'directs': directs,
64              'messages': messages,
65              'active_direct': active_direct,
66              'current_user': current_user,
67              'user_full': name,
68              'get_unread': get_unread(request)
69          }
70
71      template = loader.get_template('directChat/direct.html')
72      return HttpResponseRedirect(template.render(context, request))

```

```

directChat > 🗂 models.py > ...
1   from itertools import count
2   from django.db import models
3
4   # Create your models here.
5   from django.db import models
6   from django.contrib.auth.models import User
7   from django.db.models import Max
8
9   # Create your models here.
10  class Chat_Message(models.Model):
11      user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='user')
12      sender = models.ForeignKey(User, on_delete=models.CASCADE, related_name='from_user')
13      recipient = models.ForeignKey(User, on_delete=models.CASCADE, related_name='to_user')
14      body = models.TextField(max_length=1000, blank=True, null=True)
15      date = models.DateTimeField(auto_now_add=True)
16      is_read = models.BooleanField(default=False)
17
18      def send_message(from_user, to_user, body):
19          sender_message = Chat_Message(
20              user=from_user,
21              sender=from_user,
22              recipient=to_user,
23              body=body,
24              is_read=True)
25          sender_message.save()
26
27          recipient_message = Chat_Message(
28              user=to_user,
29              sender=from_user,
30              body=body,
31              recipient=from_user)
32          recipient_message.save()
33
34      return sender_message
35
36  def get_messages(user):
37      messages = Chat_Message.objects.filter(user=user).values('recipient').annotate(last=Max('date')).order_by('-last')
38      users = []
39      for message in messages:
40          users.append({
41              'user': User.objects.get(pk=message['recipient']),
42              'last': message['last'],
43          }

```

```

> < 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
1  {%- extends 'layout.html' %} 
2  {%- load static %} 
3  {%- block title %}<title>Megaplex Messanger</title>{%- endblock title %} 
4  {%- block main_content %} 
5  <section>
6    <div class="container py-5">
7      <div class="row">
8        <div class="col-md-12">
9          <div class="card border-15 border">
10         <div class="card-header d-flex justify-content-between align-items-center">
11           <div class="container overflow-hidden">
12             <div class="row">
13               <div class="col-4">
14                 <div class="p-3">
15                   <h5 class="mb-0 fs-3 fw-bold">Megaplex Messenger</h5>
16                 </div>
17               </div>
18               <div class="col-8">
19                 {% if user_full != None %}
20                   <div class="alert alert-primary mb-0" role="alert">
21                     <div class="d-flex justify-content-between">
22                       <h5 class="mb-0 me-3">{{user_full}}</h5>
23                       <div class="d-flex">
24                         <i class="fa fa-phone me-3" aria-hidden="true"></i>
25                         <i class="fa fa-video me-3" aria-hidden="true"></i>
26                         <i class="fa fa-share-alt" aria-hidden="true"></i>
27                       </div>
28                     </div>
29                   </div>
30                 {% endif %}
31               </div>
32             </div>
33           </div>
34         </div>
35       <div class="card-body">
36         <div class="row">
37           <div class="col-md-6 col-lg-5 col-xl-4 mb-4 mb-md-0">
38             <div class="p-3">
39               <div class="input-group rounded mb-3">
40                 <input type="search" class="form-control rounded" placeholder="Search" aria-label="Search"
41                   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-5 >
38   <div class="p-3">
39     <div class="input-group rounded mb-3">
40       <input type="search" class="form-control rounded" placeholder="Search" aria-label="Search"
41         aria-describedby="search-addon" />
42       <span class="input-group-text border-0" id="search-addon">
43         <i class="fas fa-search"></i>
44       </span>
45     </div>
46
47     <div data-mdb-perfect-scrollbar="true" style="position: relative; height: 400px">
48       <ul class="list-unstyled mb-0">
49
50         {% for message in messages %}
51
52           <li class="p-2 border-bottom">
53             <a class="{% if active_direct == message.user.username %}is-active{% endif %}" href="{% url 'directs' message.user.id %}">
54               <div class="d-flex flex-row">
55                 <div class="pt-1 d-flex">
56                   <p class="fw-bold mb-0">{{message.user.first_name}} {{message.user.last_name}}</p>
57
58                 {% if message.unread > 0 %}
59                   <p class="badge bg-danger ms-2">{{message.unread}}</p>
60
61                 {% endif %}

```

```

homepage > 🏷 views.py > ⚒ index_page
 1  from django import views
 2  from django.http.response import HttpResponseRedirect
 3  from django.shortcuts import redirect, render
 4  from django.contrib.auth import logout
 5  from channels.layers import get_channel_layer
 6  from asgiref.sync import async_to_sync
 7  from product.models import Category
 8  from product.models import Product
 9  from django.core.mail import send_mail
10  from django.conf import settings
11  from django.contrib import messages
12  from directChat.views import get_unread
13  from django.db import connection
14
15 def index_page(request):
16     products = Product.objects.filter( is_active=True ).order_by('-view_count')[:10]
17     user_list = getUserCount()
18     pro_list = ""
19     if len(user_list)>0:
20         pro_list = [0]*len(user_list)
21         ln = 0
22         for i in user_list:
23             pro_list[ln] = [Product.objects.filter(seller__id = i)[:4]]
24             ln +=1
25     context={
26         'room_name':"broadcast",
27         'products':products,
28         'collection':pro_list,
29         'get_unread':get_unread(request)
30     }
31     if request.method == 'POST':
32         type = request.POST.get('type')
33         if type=='search':
34             item = request.POST.get('item')
35             if item !="":
36                 return redirect('/product/filter/'+item)
37         elif type=="contact":
38             subject = request.POST.get('subject')
39             email = request.POST.get('email')
40             desc = request.POST.get('desc') + "\n" + "Reply Email: "+email
41             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
62 def category(request):
63     category = Category.objects.all()
64     context={
65         'products':category,
66         'category':category,
67         'room_name':"broadcast",
68         'get_unread':get_unread(request)
69     }
70     return render(request, 'homepage/category.html', context)
71
72
73
74 def getUserCount():
75     products = Product.objects.all()
76     users = {}
77     for i in products:
78         if i.seller.id in users:
79             users[i.seller.id] += 1
80         else:
81             users[i.seller.id] = 1
82     users_with_valid_numbers = []
83     for key in users:
84         if users[key]>3:
85             users_with_valid_numbers.append(key)
86
87     return users_with_valid_numbers

```

```

homepage > 🗃 urls.py > ...
1  from django.contrib.auth import views as auth_views
2  from django.urls import path
3
4  from . import views
5
6  urlpatterns = [
7      path('', views.index_page, name='index_page'),
8      path('fun/',views.fun),
9      path('fun2/',views.fun2),
10     path('games/',views.games),
11     path('category/', views.category, name='category'),
12     path('logout/', views.logout_view),
13     path('about/', views.about),
14 ]

```

```

page.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-
1  {% extends 'layout.html' %} {% load static %} {% block title %}
2  <title>Home</title>
3  {% endblock title %} {% block css_files %}
4  <link rel="stylesheet" href="{% static '/css/homepage.css' %}" />
5  <link rel="stylesheet" href="{% static '/css/main.css' %}" />
6
7  {% endblock css_files %}
8
9  {% block main_content %}
10 <section class="container-fluid search-section position-relative theme-color overflow-hidden">
11   <div class="content-search m-auto relative">
12     <p class="display-6 bolder w-100 text-center">
13       Any Smartphones, Fashions, Auto-Mobiles For You! #MEGAPLEX
14     </p>
15     <form class="d-flex w-100 m-auto flex-column flex-md-row" method="POST">
16       {% csrf_token %}
17       <input type="hidden" name="type" value="search">
18       <input class="form-control me-2" name="item" type="search" placeholder="Search" aria-label="Search" />
19       <button type="submit" class="btn btn-dark" type="submit">Search</button>
20     </form>
21
22     <div class="circles position-absolute d-flex top-0">
23       <div class="circle-one circle"></div>
24       <div class="circle-two circle"></div>
25       <div class="circle-three circle"></div>
26       <div class="circle-four circle"></div>
27     </div>
28   </div>
29
30   <div class="d-flex justify-content-center gap-3" style="margin-top: 150px;">
31
32     <a href="/product/filter/?category={{i.id}}" class="">
33       <div class="country-card position-relative">
34         
35         <div class=" h-100 position-absolute top-0 d-flex align-items-center country-bg"><span class="country-name ms-3">Connecti
36       </div>
37     </a>
38
39     <a class="w-100 align-items-center country-card bg-danger btn btn-danger shadow rounded-6 " href="/games">
40       <div class="text-white ">

```

```
1  {% extends 'layout.html' %} {% load static %} {% block title %}
2  | | <title>Home</title>
3  {% endblock title %}
4
5
6  {% block main_content %}
7      <section id="featured-box">
8          <div class="container-xl my-5">
9              <div class="intro">
10                 | | <h1 class="display-4 fw-semibold py-4 fw-bold text-black-50">Browse Catalog</h1>
11             </div>
12             <div class="row row-cols-md-3 gx-5 gy-4 row-cols-2 row-cols-lg-4 ">
13                 {% for i in category %}
14
15                     <div class="card col-auto  ">
16                         <div class="bg-image hover-overlay ripple" data-mdb-ripple-color="light">
17                             
18                             <a href="/product/filter/?category={{i.id}}">
19                                 <div class="mask" style="background-color: rgba(251, 251, 251, 0.15)"></div>
20                             </a>
21                         </div>
22
23                         <div class="card-body ps-1">
24                             <h5 class="card-title fw-bolder fs-3">{{i.name}}</h5>
25
26                         </div>
27                     </div>
28                 {% endfor %}
29             </div>
30         </div>
31     </section>
32 {% endblock main_content %}
```

```
notification > tasks.py > ...
1  from celery import shared_task
2  from channels.layers import get_channel_layer
3  from asgiref.sync import async_to_sync
4  from .models import BroadcastNotification
5  import json
6  from celery import Celery, states
7  from celery.exceptions import Ignore
8  import asyncio
9  @shared_task(bind = True)
10 def broadcast_notification(self, data):
11     try:
12         notification = BroadcastNotification.objects.filter(id = int(data))
13         if len(notification)>0:
14             notification = notification.first()
15             channel_layer = get_channel_layer()
16             loop = asyncio.new_event_loop()
17             asyncio.set_event_loop(loop)
18             loop.run_until_complete(channel_layer.group_send(
19                 "notification_broadcast",
20                 {
21                     'type': 'send_notification',
22                     'message': json.dumps(notification.message),
23                 }))
24             notification.sent = True
25             notification.save()
26             return 'Done'
27
28     else:
29         self.update_state(
30             state = 'FAILURE',
31             meta = {'exe': "Not Found"})
32
33
34     raise Ignore()
```

Figure 92: Code of Notification

```

notification > 🗂 models.py > ...
1 import json
2 from tokenize import String
3 from django.db import models
4 from django.db.models.signals import post_save
5 from django.dispatch import receiver
6 from django_celery_beat.models import PeriodicTask, CrontabSchedule
7 from pyParsing import Char
8 # Create your models here.
9
10
11 class BroadcastNotification(models.Model):
12     title = models.CharField(default="Megaplex Notification", max_length=120)
13     message = models.TextField()
14     broadcast_on = models.DateTimeField()
15     sent = models.BooleanField(default=False)
16
17     class Meta:
18         ordering = ['-broadcast_on']
19
20
21 @receiver(post_save, sender=BroadcastNotification)
22 def notification_handler(sender, instance, created, **kwargs):
23     # call group_send function directly to send notifications or you can create a dynamic task in celery beat
24     if created:
25         schedule, created = CrontabSchedule.objects.get_or_create(hour = instance.broadcast_on.hour, minute = instance.broadcast_on.minute)
26         task = PeriodicTask.objects.create(crontab=schedule, name="broadcast-notification-"+str(instance.id), task="notification.tasks.send_notification")
27

```

```

notification > 🗂 consumers.py > ...
1 # chat/consumers.py
2 import json
3 from channels.generic.websocket import AsyncWebSocketConsumer
4
5 class NotificationConsumer(AsyncWebSocketConsumer):
6     async def connect(self):
7         self.room_name = self.scope['url_route']['kwargs']['room_name']
8         self.room_group_name = 'notification_%s' % self.room_name
9
10     # Join room group
11     await self.channel_layer.group_add(
12         self.room_group_name,
13         self.channel_name
14     )
15
16     await self.accept()
17
18     async def disconnect(self, close_code):
19         # Leave room group
20         await self.channel_layer.group_discard(
21             self.room_group_name,
22             self.channel_name
23         )
24
25     # Receive message from room group
26     async def send_notification(self, event):
27         message = event['message']
28
29         # Send message to WebSocket
30         await self.send(text_data=json.dumps(message))

```

```

notification > 🗂 routing.py > ...
1 from django.urls import re_path
2 from . import consumers
3
4 websocket_urlpatterns = [
5     re_path(r'^ws/notification/(?P<room_name>\w+)/$', consumers.NotificationConsumer.as_asgi()),
6 ]

```

```

product > 🏷 views.py > ⚑ product_details
14   from directChat.views import get_unread
15   from django.http import HttpResponseRedirect
16   from django.db import connection
17   from account.models import transaction
18
19   def product_details(request,product_id):
20       product = Product.objects.get(id=product_id)
21       comments = Comment.objects.filter(product = product, parent=None)
22       replies = Comment.objects.filter(product = product).exclude(parent=None)
23       comment_count = comments.count()
24       product_category = product.category
25       no_of_random_items = 5
26       recommendations = Product.objects.filter(category=product_category)
27       possible_ids = list(recommendations.values_list('id', flat=True))
28       possible_ids = random.choices(possible_ids, k=no_of_random_items)
29       recommended_products = recommendations.filter(pk__in=possible_ids)
30       product.view_count += 1
31       product.save()
32       repDict = {}
33       for reply in replies:
34           if reply.parent.sno not in repDict.keys():
35               repDict[reply.parent.sno] = [reply]
36           else:
37               repDict[reply.parent.sno].append(reply)
38
39       context={
40           'room_name':'broadcast',
41           'product':product,
42           'comments':comments,
43           'count':comment_count,
44           'reply':repDict,
45           'recommended_products':recommended_products,
46           'get_unread':get_unread(request),
47           'room_name':'broadcast',
48       }
49       if request.method == 'POST':
50           formType = request.POST.get('formType')
51           if formType == "message" and request.POST.get('body')!="":
52               from_user = request.user
53               to_user_username = request.POST.get('to_user')
54               body = request.POST.get('body')

```

Figure 93: Code of Product Detail app

```

product > 🏷 urls.py > ...
1   from unicodedata import name
2   from django.contrib.auth import views as auth_views
3   from django.urls import path
4   from . import views
5
6   urlpatterns = [
7       path('details/<int:product_id>', views.product_details, name='Details_page'),
8       path('filter/', views.filter_page),
9       path('filter/<str:item>', views.searchProduct),
10      path('user/<int:user>', views.searchUserProduct),
11      path('<int:product_id>/like',views.ToggleProductlike,name ='like'),
12      path('<int:product_id>/removeliked',views.RemoveFromLikedList,name='removefromlikelist'),
13      path('<int:comment_id>/comment_like',views.like_toggle,name ='like'),
14      path('explore',views.explorepage,name='explore')
15   ]

```

```

product > views.py > product_details
15  from django.http import HttpResponseRedirect
16  from django.db import connection
17  from account.models import transaction
18
19 def product_details(request,product_id):
20     product = Product.objects.get(id=product_id)
21     comments = Comment.objects.filter(product = product, parent=None)
22     replies = Comment.objects.filter(product = product).exclude(parent=None)
23     comment_count = comments.count()
24     product_category = product.category
25     no_of_random_items = 5
26     recommendations = Product.objects.filter(category=product_category)
27     possible_ids = list(recommendations.values_list('id', flat=True))
28     possible_ids = random.choices(possible_ids, k=no_of_random_items)
29     recommended_products = recommendations.filter(pk__in=possible_ids)
30     product.view_count += 1
31     product.save()
32     repDict = {}
33     for reply in replies:
34         if reply.parent.sno not in repDict.keys():
35             repDict[reply.parent.sno] = [reply]
36         else:
37             repDict[reply.parent.sno].append(reply)
38
39 context={
40     'room_name':"broadcast",
41     'product':product,
42     'comments':comments,
43     'count':comment_count,
44     'reply':repDict,
45     'recommended_products':recommended_products,
46     'get_unread':get_unread(request),
47     'room_name':"broadcast",
48 }
49 if request.method == 'POST':
50     formType = request.POST.get('formType')
51     if formType == "message" and request.POST.get('body')!="":
52         from_user = request.user
53         to_user_username = request.POST.get('to_user')
54         body = request.POST.get('body')
55         try:

```

```

product > models.py > ...
2  from django.db.models.base import Model
3  from django.contrib.auth.models import User
4  from django.utils.timezone import now
5
6  WARRANTY_CHOICES = (
7      ("None", "None"),
8      ("1 Year","1 Year"),
9      ("2 Years","2 Years"),
10     ("3 Years","3 Years"),
11     ("4 Years","4 Years"),
12     ("5 Years","5 Years")
13 )
14
15 class Category(models.Model):
16     name = models.CharField(max_length=200)
17     picture = models.ImageField(upload_to = 'category_images', default="def.jpeg")
18
19     def __str__(self) -> str:
20         return self.name
21
22 class Sub_Category(models.Model):
23     name = models.CharField(max_length=200)
24     category = models.ForeignKey(Category,on_delete=models.CASCADE)
25
26
27     def __str__(self) -> str:
28         return self.name
29
30 class Brand(models.Model):
31     name = models.CharField(max_length=200)
32
33     def __str__(self) -> str:
34         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-between
1  {% extends 'layout.html' %} {% load static %} {% load extras %} {% block title %}
2  <title>{{product.name}}</title>{% endblock title %}
3
4  {% block main_content %}
5
6  <div class="container text-center">
7    <div class="row m-5">
8      <div id="carouselExampleIndicators" class="carousel slide" data-mdb-ride="carousel">
9        <div class="carousel-indicators">
10          <button type="button" data-mdb-target="#carouselExampleIndicators" data-mdb-slide-to="0" class="active"
11            aria-current="true" aria-label="Slide 1"></button>
12          <button type="button" data-mdb-target="#carouselExampleIndicators" data-mdb-slide-to="1"
13            aria-label="Slide 2"></button>
14          <button type="button" data-mdb-target="#carouselExampleIndicators" data-mdb-slide-to="2"
15            aria-label="Slide 3"></button>
16        </div>
17        <div class="carousel-inner border-15">
18          <div class="carousel-item active border border-15 border">
19            
21          </div>
22          <div class="carousel-item border border-15 border">
23            
25          </div>
26          <div class="carousel-item border border-15 border">
27            
29          </div>
30        </div>
31        <button class="carousel-control-prev" type="button" data-mdb-target="#carouselExampleIndicators"
32          data-mdb-slide="prev">
33          <span class="carousel-control-prev-icon text-black" aria-hidden="true"></span>
34          <span class="visually-hidden">Previous</span>
35        </button>
36        <button class="carousel-control-next" type="button" data-mdb-target="#carouselExampleIndicators"
37          data-mdb-slide="next">
38          <span class="carousel-control-next-icon text-black" aria-hidden="true"></span>
39          <span class="visually-hidden">Next</span>
40        </button>
41      </div>

```

```

product > templates > product > filter.html > ...
1  {% extends 'layout.html' %} {% load static %} {% block title %}
2  <title>Filter</title>{% endblock title %}
3
4  {% block main_content %}
5  <div class="container pt-5">
6    <div class="row">
7      <div class="col-md-8 order-md-2 col-lg-9">
8        <div class="container-fluid">
9          <div class="d-flex justify-content-between">
10            <li class="m-2">
11              {% if search_item or min_tag %}
12                <h5>Search result for:</h5>
13                {% endif %}
14                <div class="d-flex">
15                  {% if search_item %}
16                    <p><span class="badge bg-light text-black">{{search_item}}</span></p>
17                  {% endif %}
18                  {% if min_tag %}
19                    <p><span class="badge bg-light text-black">Minimum Price: {{min_price1}}</span></p>
20                    <p><span class="badge bg-light text-black">Maximum Price: {{max_price1}}</span></p>
21                  {% endif %}
22                </div>
23            </li>

```

```

roomChat > 🗂 views.py > ...
1  from django.shortcuts import render
2  from directChat.views import get_unread
3
4  def index(request):
5      return render(request, 'roomChat/chat.html')
6
7  def room(request, room_name):
8      return render(request, 'roomChat/room.html', {
9          'room_name1': room_name,
10         'room_name':"broadcast",
11         'get_unread':get_unread(request)
12     })

```

Figure 94: Code of room Chat app

```

roomChat > 🗂 urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('', views.index, name='index'),
6      path('<str:room_name>', views.room, name='room'),
7 ]

```

```

roomChat > 🗂 consumers.py > ...
1  # chat/consumers.py
2  import json
3  from asgiref.sync import async_to_sync
4  from channels.generic.websocket import WebsocketConsumer
5
6  class ChatConsumer(WebsocketConsumer):
7      def connect(self):
8          self.room_name = self.scope['url_route']['kwargs']['room_name']
9          self.room_group_name = 'room_%s' % self.room_name
10
11         # Join room group
12         async_to_sync(self.channel_layer.group_add)(
13             self.room_group_name,
14             self.channel_name
15         )
16
17         self.accept()
18
19     def disconnect(self, close_code):
20         # Leave room group
21         async_to_sync(self.channel_layer.group_discard)(
22             self.room_group_name,
23             self.channel_name
24         )

```

```

roomChat > templates > roomChat > chat.html > ...
1  {% extends 'layout.html' %} {% load static %} {% block title %}
2  <title>Room Selection </title>
3  {% endblock title %}
4
5
6  {% block main_content %}
7  <div
8      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-hidden"
9      
10     <span class="text-black">What chat room would you like to enter?</span>
11
12
13     <select id="room-name-input" class="my-2">
14         <option value="General">General</option>
15         <option value="Q & A">Q & A</option>
16         <option value="Marketing">Marketing</option>
17         <option value="Technical">Technical</option>
18     </select>
19     <input id="room-name-submit" class="btn btn-primary col-lg-3" type="button" value="Enter">
20
21
22     <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>
23     <span class="position-absolute bottom-0 me-5 mb-5 end-0 fs-6 " style="z-index: -1;"><i
24         class="fas fa-circle text-danger ms-5 mb-7 "></i></span>
25     <span class="position-absolute bottom-0 fs-4 mb-2 "><i class="far fa-heart text-primary "
26         style="transform: rotate(335deg);"></i></span>
27     <span class="position-absolute bottom-0 end-0 me-4 fs-6 "><i class="fas fa-times text-danger "
28         style="transform: rotate(75deg);"></i></span>
29     <span class="position-absolute bottom-50 end-0 me-4 fs-5 "><i class="fas fa-times text-primary mb-10"
30         style="transform: rotate(75deg);"></i></span>
31     <span class="position-absolute bottom-50 end-0 me-4 fs-4 fa-flip-vertical "><i
32         class="far fa-square text-primary fa-rotate-by" style="transform: rotate(75deg);"></i></span>
33     <span class="position-absolute bottom-0 ms-5 m-auto mb-3 "><i class="fas fa-circle text-primary ms-5 "></i></span>
34 </div>

```

```

38 {{ room_name|json_script:"room-name" }}
39 <script>
40     const loggedUser = JSON.parse(document.getElementById('user_id').textContent);
41
42     const room = JSON.parse(document.getElementById('room-name').textContent);
43
44     const chatSocket = new WebSocket(
45         'ws://' +
46         window.location.host +
47         '/ws/room/' +
48         room +
49         '/'
50     );
51
52     chatSocket.onmessage = function (e) {
53         const data = JSON.parse(e.data);
54
55         if (loggedUser != data.username) {
56             document.querySelector('#chat-log').innerHTML +=
57                 "<div class='d-flex flex-row justify-content-end mb-4 pt-1'>" +
58                 "" +
59                 "<p class= 'small p-2 mb-1 text-white rounded-3 bg-danger ms-1' >" + data.username + ":" +
60                 data.message + "</p>" +
61                 "</div>" +
62         } else {
63             document.querySelector('#chat-log').innerHTML +=
64                 "<div class='d-flex flex-row justify-content-start mb-4 pt-1'>" +
65                 "" +
66                 "<p class= 'small p-2 mb-1 text-white rounded-3 bg-secondary ms-1' >" + data.username + ":" +
67                 data.message + "</p>" +
68                 "</div>" +
69         }
70     };
71
72     chatSocket.onclose = function (e) {
73         console.error('Chat socket closed unexpectedly');
74     };

```

```

admins > 🐍 views.py > ⌂ send_notification
13  from django.shortcuts import redirect
14  import os
15  from django.contrib.auth import logout
16  from notification.models import BroadcastNotification
17
18 # Create your views here.
19 def dashboard(request):
20     users = User.objects.all()
21     user_info = users.filter(is_staff=0)
22     admin_info = users.filter(is_staff=1)
23     context={
24         'admin_info':admin_info,
25         'user_info':user_info,
26     }
27     user_count = users.count()
28     user_info = User.objects.filter(is_staff=0)
29     admin_info = User.objects.filter(is_staff=1)
30     product = Product.objects.all()
31     view_count = 0
32     profiles = Profile.objects.all()
33     top_sellers = profiles.order_by('-product_sold')
34     top_products = product.order_by('-view_count')[:5]
35     active_product_count = product.filter(is_active=True).count()
36     inactive_product_count = product.filter(is_active=False).count()
37     for i in product:
38         view_count += i.view_count
39
40     context={
41         'admins':admin_info,
42         'users': user_info,
43         'users_count':user_count,
44         'view_count':view_count,
45         'top_seller':top_sellers,
46         'top_products':top_products,
47         'active_product_count':active_product_count,
48         'inactive_product_count':inactive_product_count
49     }
50
51     return render(request, 'admins/dashboard.html',context)

```

Figure 95: Code for admin section

```

admins > 🐍 urls.py > ...
1  from django.contrib.auth import views as auth_views
2  from django.urls import path
3
4  from . import views
5
6  urlpatterns = [
7      path('dashboard/', views.dashboard, name='dashboard'),
8      path('send_notification/',views.send_notification,name='send_notification'),
9      path('add_category/',views.add_category,name='add_category'),
10     path('add_subcategory/',views.add_subcategory,name='add_subcategory'),
11     path('<int:category_id>/edit_category',views.edit_category,name='edit_category'),
12     path('activate/<int:id>', views.active),
13     path('deactivate/<int:id>', views.deactive),
14     path('verify/<int:id>', views.verify),
15     path('unverify/<int:id>', views.unverify),
16     path('deleteSub/<int:id>', views.deleteSub),
17     path('deleteCat/<int:id>', views.deleteCat),
18     path('logout', views.logout_user)
19 ]

```

Figure 96: Admin pages' URLs

```

admins > forms.py > ...
1  from django.forms import DateTimeInput, ModelForm
2  from notification.models import BroadcastNotification
3  from product.models import Category, Sub_Category
4
5  class NotificationForm(ModelForm):
6      class Meta:
7          model = BroadcastNotification
8          fields = ['message', 'broadcast_on', 'title']
9          widgets = {
10              'broadcast_on': DateTimeInput(attrs={'type': 'datetime-local'}),
11          }
12
13  class CategoryForm(ModelForm):
14      class Meta:
15          model = Category
16          fields = "__all__"
17
18  class SubcategoryForm(ModelForm):
19      class Meta:
20          model = Sub_Category
21          fields = "__all__"

```

Figure 97: Admin page's forms

```

$ > admins > dashboard.html > section.mb-5 > div.row.gx-5 > div.col-3.col-md-6.mb-4.mb-xl-0 > a.d-flex.align-items-center.p-4.bg-glass.shadow.rounded-6.te
1  {% extends 'admin_layout.html' %} {% load static %} {% block title %}
2  <title>profile</title>
3  {% endblock title %} {% block css_files %}
4
5  <link rel="stylesheet" href="{% static '/css/main.css' %}" />
6  <link rel="stylesheet" href="{% static '/css/homepage.css' %}" />
7  <link rel="stylesheet" href="{% static '/css/profile.css' %}" />
8
9  {% endblock css_files %} {% block main_content %}
10
11
12 <!-- Main content -->
13 <div class="container py-5">
14     <!-- Section: Summary -->
15     <section class="mb-5">
16         <div class="row gx-xl-5">
17             <div class="col-xl-3 col-md-6 mb-4 mb-xl-0">
18                 <!-- Card -->
19                 <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%)>
20                     <div class="rounded-4 fs-1">
21                         <i class="fa fa-user"></i>
22                     </div>
23
24                     <div class="ms-4">
25                         <p class="text-muted mb-2">Users</p>
26                         <p class="mb-0">
27                             <span class="h5 me-2">{{users_count}}</span>
28                             <small class="text-success text-sm"><i class="fas fa-arrow-up fa-sm me-1"></i>13.48%</small>
29                         </p>
30                     </div>
31                 </a>
32             </div>
33         </div>
34     </section>
35 </div>
36
37
38
39
40

```

Figure 98: Admin dashboard code

```

admins > templates > admins > send_notification.html > div#exampleModal.modal.fade
16  <table class="table container shadow">
17    <thead>
18      <tr>
19        <th scope="col">Message</th>
20        <th scope="col">Broadcast On</th>
21        <th scope="col">Sent?</th>
22      </tr>
23    </thead>
24    <tbody>
25      {% for i in nots %}
26      <tr>
27        <td>{{i.message}}</td>
28        <td>{{i.broadcast_on}}</td>
29        <td>{{i.sent}}</td>
30      </tr>
31      {% endfor %}
32    </tbody>
33  </table>
34  <div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
35    <div class="modal-dialog">
36      <div class="modal-content">
37        <div class="border-0 d-flex justify-content-end ">
38
39          <button type="button" class="btn-close me-1 mt-1" data-mdb-dismiss="modal"
40            | aria-label="Close"></button>
41        </div>
42        <div class="modal-body">
43          <div class="m-auto p-5 ">
44            <h1 class="text-center">Add Notification</h1>
45            <form method = "POST" enctype="multipart/form-data">
46              {% csrf_token %}
47              {{ form.title | as_crispy_field }}
48              {{ form.message | as_crispy_field }}
49              {{ form.broadcast_on | as_crispy_field }}
50            <button type="submit" class="btn btn-primary btn-block">Send</button>
51
52          </div>
53        </div>
54      </div>
55    </div>
56  </div>

```

Figure 99: Admin Notification Control Page's Code

Thankyou!