Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Project Report

"AttHire"

Submitted by:

Aakriti Banjara(05)
Sabin Bhattarai (09)
Jiwan Humagain(19)
Anmol Karki(20)
Sudip Subedi (52)

Submitted to:

Assistant Professor

Mr. Santosh Khanal

Department of Computer Science and Engineering

Submission Date: 5th April, 2023

Bona fide Certificate

This project work on

“AttHire”

is the bona fide work of

“Aakriti Banjara,Sabin Bhattarai ,Jiwan Humagain,

Anmol Karki and Sudip Subedi”

who carried out the project work under my supervision.

Project Supervisor

_____

Praynita Karki
Assistant Professor
Department of Computer Science and Engineering

Date: 11/23/2022

ACKNOWLEDGEMENT

We would like to express our sincere appreciation to our supervisor, Mrs. Praynita Karki for her constant guidance, encouragement, understanding and recommendations during the advancement of our project, without her valuable help this project would not have been possible. We were able to elevate our idea due to her guidance. We are highly indebted to him for believing in us and for being a constant source of motivation.

We would also like to thank our project coordinator Dr. Rajani Chulyadyo for instancing different ideas as a semester project, approving our project proposal, and assigning an appropriate supervisor for the project.

We extend our gratitude to Kathmandu University Department of Computer Science and Engineering for providing us a platform where students can work on a topic as a semester-long project and demonstrate our findings.

"Thank You"

Abstract

‘AttHire’ is a web-based rental website.The rise of fashion carried out by the high wave of new trends in social media has been prominent.Considering the poor economic background of Nepal, trends are changing so quickly that our people, especially majority of our women, cannot afford to buy clothes every now and then.

Hence to bridge this economic gap we have developed ‘AttHire’, an e-commerce platform that facilitates both individuals and businesses to rent clothes to customers as well as hire for themselves. This will benefit not only businesses but also individuals, who can earn some cash in the exchange of their clothes for a certain period of time.

JavaScript, CSS, HTML, Material UI, Redux, Stripe has been used to build the websites’s frontend.For the backend, we have used Node Js, Express, MongoDb, JWT. In this website we have basically two modules. The first one is the client module for the customers and the second module is the admin dashboard.

The customer has to register to surf the website. The registered customer can view details of product and hire/rent clothes as per their wish. He/She has to complete the payment to place an order. The admin dashboard contains the access to the admin page on the website. Only the users who have been given admin privilege can add and delete users and products.

Keywords:*NodeJs,JavaScript,MongoDB,CSS,HTMLMaterialUI,Redux,Stripe, Express.*

Table of Contents

ACKNOWLEDGEMENT i

Abstract ii

List of Figures viii

List of Tables ix

Acronyms x

Chapter 1: Introduction 1

1.1 Background 1

1.2 Objective 2

1.3 Motivation and Significance 2

Chapter 2: Related Works 4

Chapter 3: Design and Implementation 5

3.1 Web Interface 5

3.1.1 Design and Interface 5

3.1.2 Routing and dynamic Range Rendering 7

3.1.3 User Access Control and Session Tracking 8

3.1.4 Services Activities and Micro-interactions 8

3.2 API and Microservices 10

3.2.1 User Registration with Authentication 10

3.2.2 Login 10

3.2.3 Product processing 10

3.2.4 Product Retrieval 10

3.3 Database and ER Diagram 11

3.3.1 Users 12

3.3.2 Product 12

3.3.3 Carts 12

3.3.4 Orders 12

3.4 Architectural Design 13

3.4.1 Flowchart 13

Chapter 4: Implementation 14

4.1 Web Application 14

4.1.1 Project directory structure 14

4.1.2 Application Setup 15

4.2 ExpressJS REST API 16

4.2.1 Project directory structure 16

4.2.2 Server Configuration 16

4.2.3 Endpoints 17

4.2.3.1 Authentication Route (/auth/) 17

4.2.3.2 Product Route (/api/product/) 19

4.2.3.3 Orders Route (/api/order/) 22

4.2.3.4 Cart Route (/api/cart/) 23

Chapter 5: System Requirement Specification 24

5.1 Software Specification 24

5.1.1 Web Application technologies used 24

5.1.2 ExpressJS API Dependencies 25

5.1.2 Hardware Specification 26

5.1.2.1 Minimum Requirements 26

5.1.2.2 Services and technologies used 26

Chapter 6: Discussion on the Achievements 27

6.1 Features 27

6.2.1 Login and registration 27

6.2.2 Cart System 27

6.2.3 Admin Dashboard 27

6.2.4 User-friendly Navigation 27

Chapter 7: Conclusion and Recommendation 28

5.1 Limitation 28

5.2 Future Enhancements 28

References 30

Appendix 31

Appendix 1 - Project Account and Journal 31

Appendix 2 - Work Division 33

Appendix 3 – Screenshots 34

List of figures

Figures **Page no.**

Figure 3.1 Main Page of the website display 6
Figure 3.2 Block Diagram showing the page request handling sequence 7
Figure 3.3 Main page of the website showing filtered services 9
Figure 3.4 Events,Requests and Responses 9
Figure 3.5 Entity Relation Diagram 11
Figure 3.6 Application Flowchart 13
Figure 4.1 Project Directory Structure of backend 14
Figure 4.2 Project Directory Structure of backend 16
Figure 9.1 Sign up Page 35
Figure 9.2 Login Page 35
Figure 9.3 Main Page 35
Figure 9.4 Main 36
Figure 9.5 Product page asking the customer number of days to rent 36
Figure 9.6 Cart Page 37
Figure 9.7 Payment Page for Buyer's Info 37
Figure 9.8 Payment Page for Buyer's card information 38
Figure 9.9 Admin Dashboard 38
Figure 9.10 New Product Creating Page 39
Figure 9.11 Product List 40

List of Tables

Tables **Page no.**

Table 4.2.3.1 Project Directory structure of backend 19

Table 4.2.3.2 Product route (\api\product) 22

Table 4.2.3.3 Order route (\api\order) 23

Table 4.2.3.4 Cart route (\api\cart) 24

Table 5.1.1 Web Application Technologies used 25

Table 5.1.2 Express JS API dependencies 26

Table 9.1 GANTT Chart 31

Table 9.2 Legend 32

Table 9.3 Work division of Web Application 33

Table 9.4 Work division of Microservice Backend 34

Acronyms

RAM	**Random Access Memory**
CPU	**Central Processing Unit**
JS	**JavaScript**
CLI	**Command Line Interface**
UI	**User Interface**
API	**Application Programming Interface**
CRUD	**Create Read Update Delete**
URL	**Uniform Resource Locator**
CSS	**Cascading Style Sheets**

Chapter 1 Introduction

1.1 Background

Everyone needs clothing for comfort and staying warm,however,the evolution of fashion has been such that Fashion designers no longer dictate fashion trends,but influencers and stars do.With the rise of tiktok,Instagram and other social media platforms,the fashion industry has taken a swift in a new direction.Often Women consider clothes to be old with just one or two wears. Considering the poor economic background of Nepal, buying a new outfit for one occasion is one of the major hardships Nepali women face at the brink of an event. Considering this essence, ATTHIRE is created to bridge the economic gap Nepali women face by facilitating both individuals and businesses to rent clothes to consumers for a certain time.Despite the rise of these e-commerce businesses and new fashion trends,Nepali People still lag behind in this rapidly changing fashion industry due to their weak economic background and their tendency to not wear a dress after using it once or twice.To tackle this problem we have developed ATTHIRE.

ATTHIRE is a web application that allows people to rent products at a reasonable price and also provides an opportunity for people to earn some cash in return who already have some product that they don't use actively.

With the use of React,Redux and Material UI we aim to develop a user friendly interface to make the shopping experience pleasant for the users. In the main page we will group and exhibit different cards in a pagination system. Each card will contain the trending product, shop now button, which provides a full description about the product in another page.

In the description page all information about the product will be provided alongside with add to cart service. From there we will be able to see information about how many products a person rented .

Cart system will be provided to all the users who are logged in so that they can add their respective products for later. Total price with total units will be shown in cart alongside with the services that the user will be adding. Users will be able either delete the item or move to the payment procedure page form that.

We will be using Stripe online payment system to deal with monetary transactions. Credit card details will be handled by Stripe payment system. After payment all orders will be visible to the user in their profile.

1.2 Objective

1. To develop an easy to use web based interface where users can rent or sell their products
2. To enable websites with the latest brands giving an attractive and interactive look to online retailers
3. To use a stripe online payment system to deal with monetary transactions
4. To make the website secure, fast and stable for a smoother experience

1.3 Motivation and Significance

Many people today desire to dress differently for various occasions, but they don't want to buy the outfits. We developed an online store where customers could rent clothing in order to address this issue. People want different dresses for various

celebrations, but they don't want to buy them. To address this issue, we have created a website that will offer a range of rental clothing that can be returned once it has been worn for the desired event.Through the system, sellers can deliver the product as per the client request. We hope this website has created the best experience for both the buyers and the sellers and the problem of buying expensive clothes for the single occasion.

Chapter 2 Related Works

Although there have been many E-commerce websites for the sale of used products all over the world, not many exist like this one in Nepal. Some of the open-source projects that we could find and other popular projects are as follows:

GitHub - medusa js/medusas (2020), is an open source project in which an e-commerce website is built using Node.js. It was made as an alternative to Shopify and is very feature rich along with a very good UI. However, it doesn't serve our purpose of serving women and economically backward groups in Nepal to help them wear the clothes they desire without having to buy them to be fashionable.

ThriftStoreNepal (2021) is a popular project which is not actually open source, but the theme is similar to that of ours. It has a good user interface and easier navigation. However, it actually buys used items and sells/rents them by themselves, both virtually and physically via their store, as opposed to the idea of ours, where people and businesses can still be the owners of the product while renting it for cash, where we will act as a mediator.

Chapter 3 Design and Implementation

3.1 Web Interface

The project has an interface in the form of a website for the users to interact with. The website is built with ReactJS on top of NodeJS alongside ExpressJS. The website has ample features for users to register an account, log in, interact with the news posts, change their details in settings and so on. The features of the website are discussed below:

3.1.1 Design and Interface

The website features modern minimal design language. The page layouts are content focused while being responsive to different screen sizes and types across devices for better user experience. The pages were developed following mobile-first design followed by Progressive Advancement [1] which helped building of the pages faster and consistently.

ReactJS was chosen for the frontend of this project because of the seamless integration of programming logic and calculation along with the HTML elements in the form of JSX. ReactJS also allows the features of in-page data management with Props and State which helps keep track of the users' real time activity removing the need to rely on the backend which is time and resources heavy.

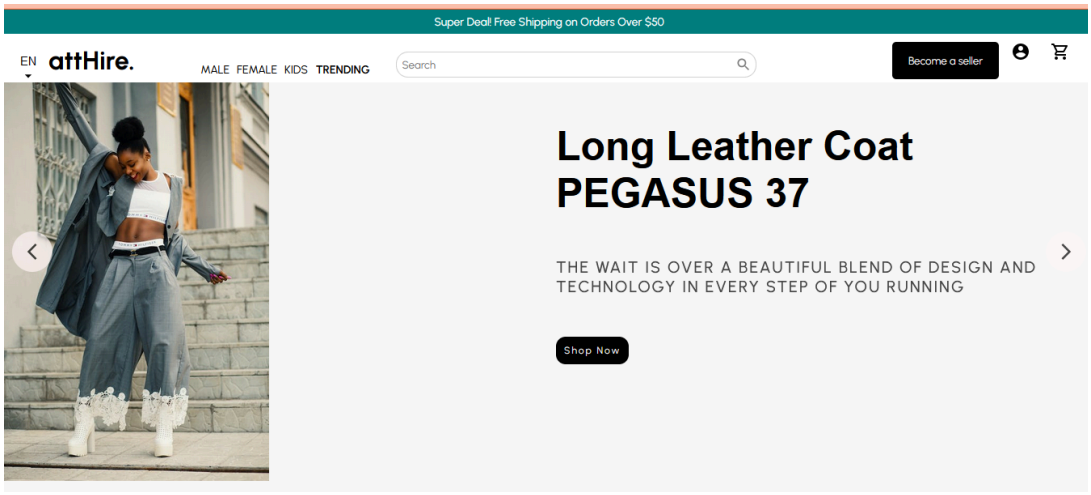


Figure 3.1 Main page of the website display.

All pages are styled with custom CSS and no pre-defined third-party frameworks or libraries were used (except for Google Fonts [2]). All the styles of elements in the pages were designed following the latest CSS3 features for a modern and sleek look while also being minimal.

The CSS are written such that the designs remain consistent across devices with varying widths and orientation. Also, the layouts are developed to adapt itself to the devices' available width so that the user does not need to pan or zoom and have a better experience while going through the web pages.

3.1.2 Routing and dynamic Range Rendering

Each of the pages have their routes defined in the system. The specific pages are invoked and generated dynamically whenever a user requests for a page to arrive at the server. NodeJS has a native module for http server building, but ExpressJS library has been used in this project to simplify and ease up the development process.

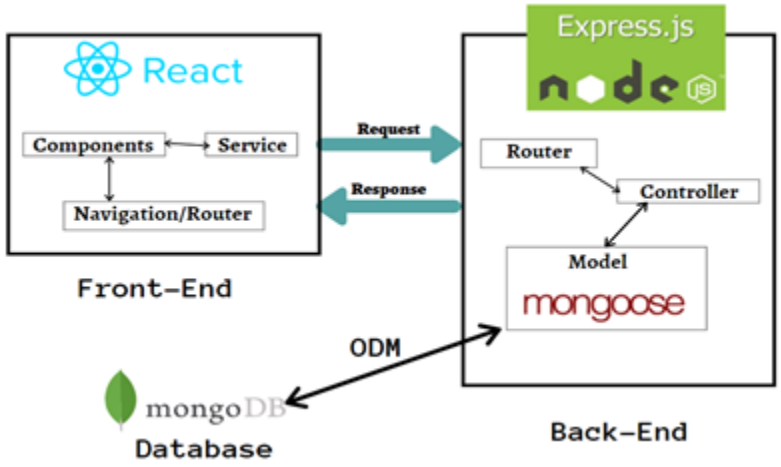


Figure 3.2 Block Diagram showing the page request handling sequence

There are times when the contents of the pages need to be generated in real time with the information related to the specific user or the logged-state of the user. For example, if user "jiwan" is trying to access his cart page, the page related to displaying profile only contains a template for information display and the content needs to be put in for the page to be rendered as expected. This process is called Server-Side Rendering (SSR). The main advantage of the SSR approach is that it removes most of the loads from clients' side and shifts them to the more powerful server's side for page rendering and thus, makes it feasible for the clients with minimum hardware specification to browse and operate the website.

3.1.3 User Access Control and Session Tracking

A web application can have many users accessing the website. It is, thus, important to have an access control mechanism in place to control which user can access what kind of data. One user cannot update or alter data of another user.

For this, the website uses session cookies to track the logged in user. The user-id of the logged in user is encrypted and stored as a cookie in the user's browser. This cookie can be accessed for each request the user makes to the server. The cookie can then be parsed to extract the user-id of the user and filter out the information the user cannot have access to and only the information that the user can access is sent to the user's end. In case, the user tries to access data that is not accessible, they are redirected to the home page indicating that they cannot access the content.

3.1.4 Services Activities and Micro-interactions

The main objective of this project was to create a platform for people to find quality services and for that, the filtration of the services would be done with the user's input or from selecting respective categories as shown in figure below. For more filtration of services users can set product's color ,size ,sort recently added products and services will be shown accordingly.

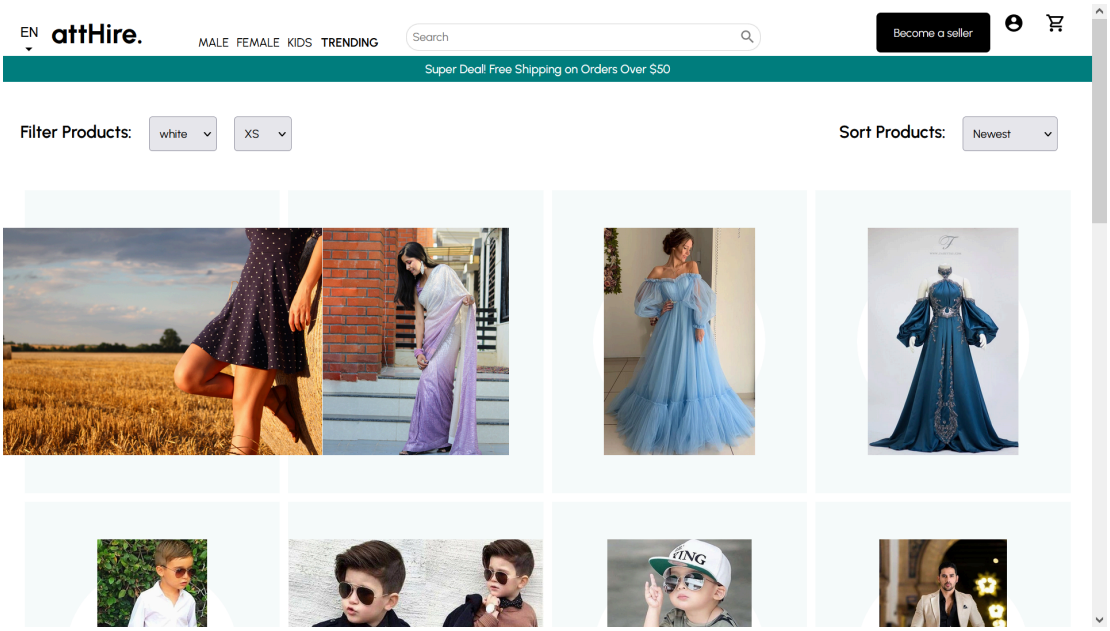


Figure 3.3 Main page of the website showing filtered services

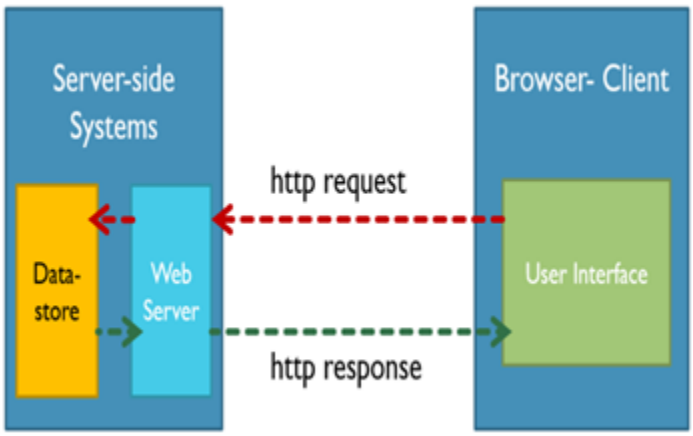


Figure 3.4 Events,Requests and Responses

3.2 API and Microservices

The web application communicates with our node microservice using RESTful requests / responses built on the ExpressJS framework. The services provided by our API are as follows:

3.2.1 User Registration with Authentication

User registration endpoint listens for account registration credentials such as email, password and username. This service authenticates user registration by validating credentials, hashing passwords, checking for duplicate registrations.

3.2.2 Login

Registered users need to be logged-in in order to access and interact with our application. Login-sessions are handled on a web-application site, whose credentials are first validated by this service.

3.2.3 Product processing

This endpoint extracts the data provided or posted by the user, processes them and stores it to the database. This endpoint accepts data as json and stores them in service collection in mongo DB database.

3.2.4 Product Retrieval

A collection of endpoints returns services that have been stored in the database. It provides 3 distinct news extraction methods:

- All Products: It returns all products that are stored in the database.

- Single Product: It returns product when the user clicks on the view details button. This endpoint returns a single product with a unique id stored in the database.
- Filtered product: It returns product according to user input or selection of categories or price range. This endpoint returns a single product with a filtered query in the URL link.

3.3 Database and ER Diagram

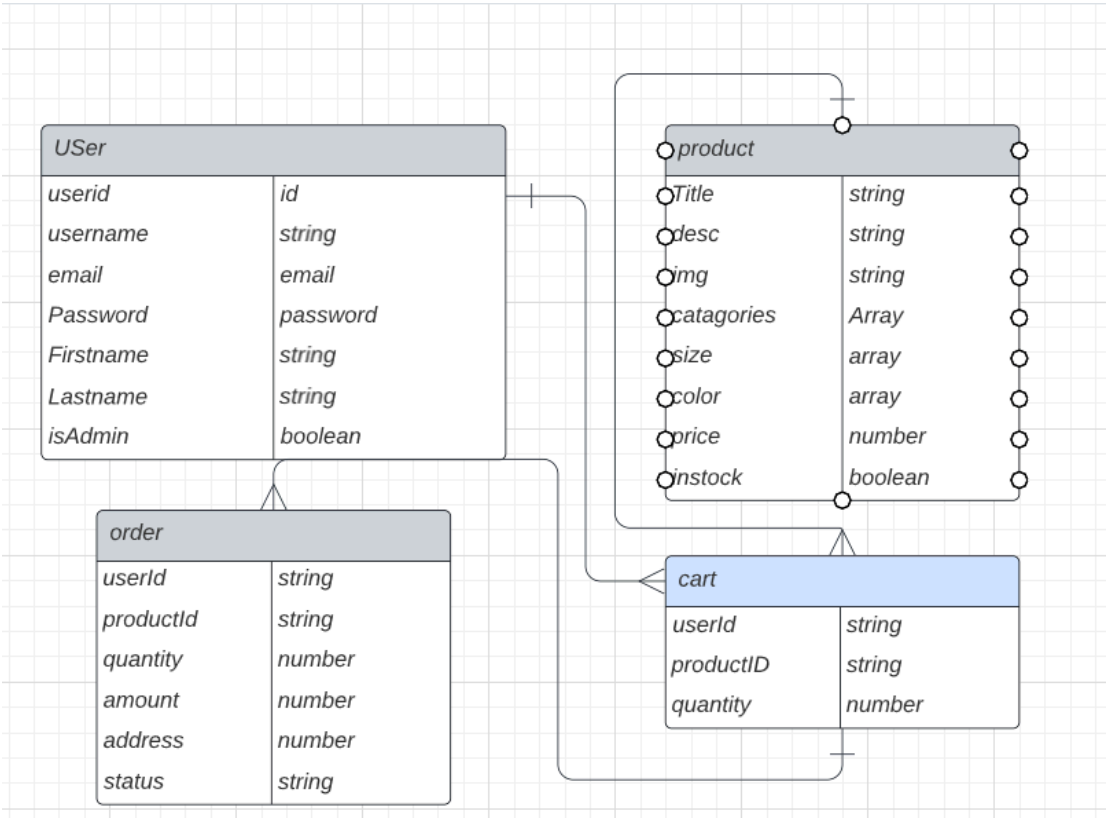


Figure 3.5 Entity Relation Diagram

The data gathered in our project are modeled using a Document-oriented database, MongoDB. Our database design is structured into five collections: users, services, orders, reviews and carts.

3.3.1 Users

Every user in the users collection is unique and indexed, for speeding up query and preventing duplication. The values in the ‘password’ field are encrypted passwords.

3.3.2 Product

The data that is posted by the user is sent to the backend as json format which is processed and stored in this collection. The ‘title’ field of this collection is indexed as a string and it stores the title of the product. ‘Description’ field stores descriptive synopsis of service whereas ‘images’ stores the URL of the data for individual images.

3.3.3 Carts

When the user clicks Add to Cart button the service alongside the user is saved in cart collection in the database. Carts collection only contains two keys in table one ‘user_id’ which stores the information about the user and ‘product_id’ which stores the information added into the Cart.

3.3.5 Orders

After the user proceeds to payment his/her user data alongside with product data and payment all are sent to the backend. Orders collection stores vital information such as payment information, order status information, etc. ‘paymentInfo’ field stores url_id and status given by the stripe system. ‘orderStatus’ field stores where the order is delivered or still is in processing.

3.4 Architectural Design

3.4.1 Application Flowchart

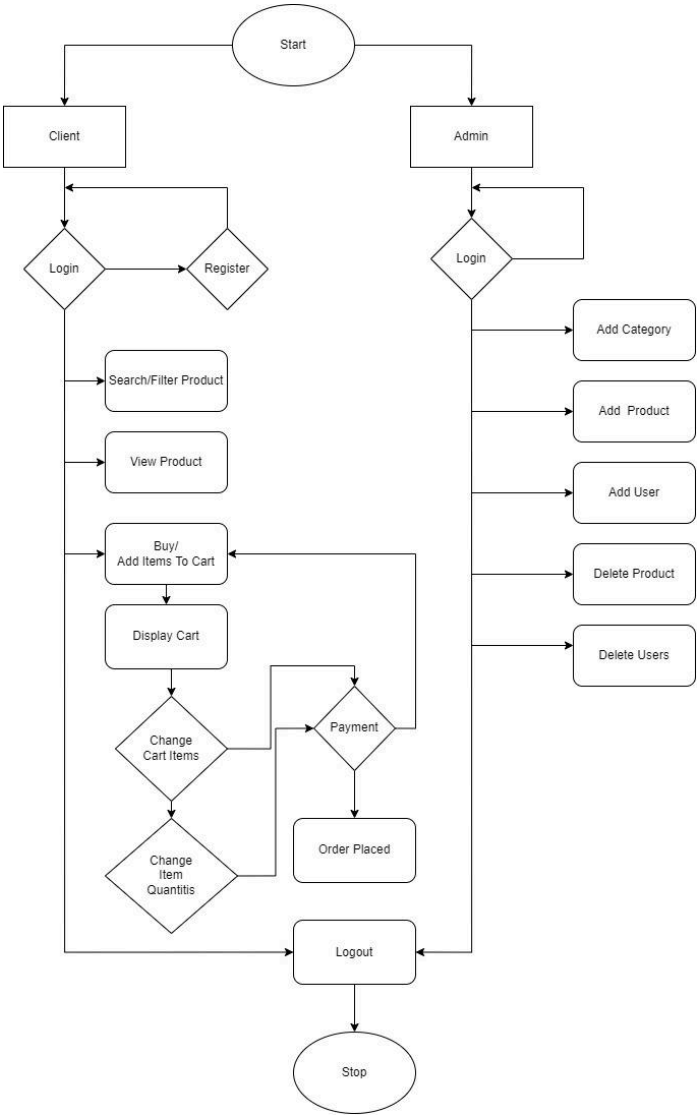


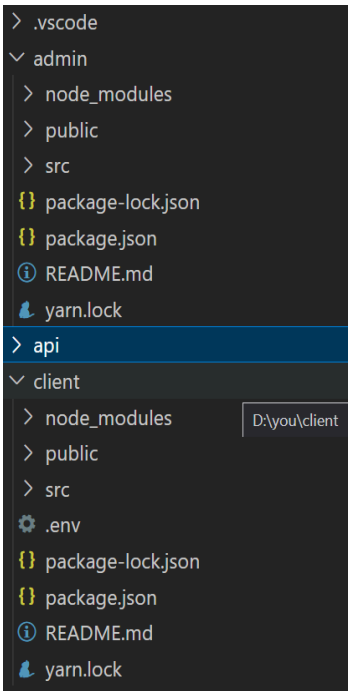
Figure 3.6 Application Flowchart

Chapter 4 Implementation

4.1 Web Application

4.1.1 Project directory structure

The ‘index.js’ in file backend is the entry point for the application startup that initiates a http server built with Express[1]. The ‘package.json’ file contains the general information about the project repository, dependency-list, and start-up commands. The environment variables are stored in a '.env' file held inside file directory names ‘backend’ and loaded into the system on the startup process of the application.



The ‘routes’ folder held inside file directory names ‘backend’ contains a file which has defined functions for API HTTP requests from the users’ end. The models folder inside the api contains the schemas for the mongodb.

Figure 4.1 Project Directory Structure of backend

The ‘Admin’ is a folder that contains the frontend info of our admin panel whereas the ‘client’ folder contains the frontend data of our react application. Inside clients we have a src folder which contains pages and component folders which stores all the required data for our application.

4.1.2 Application Setup

The following are brief discussions on important modules and functionalities that are used in the development of the web application.

a) **Server**

The application features a http server, powered by Express. The server watches for any incoming HTTP request from users, determines its method and process accordingly. The POST requests are routed to the different routes such as '/api/auth', '/api/product', '/api/order'. The GET requests are handled with in-file definitions.

b) **Requests**

The web application manages GET requests with the help of axios HTTP requests. The web application needs to communicate with the expressJS REST API, the other part of this project, mainly for POST requests. The GET requests are routed to the different routes such as '/api/auth', '/api/product/:id', '/api/order/', etc. For this, we have used Axios. Axios makes it easy to send HTTP requests to the APIs.The web pages also use Axios to send POST requests to the local server.

c) **Image Storage**

The users' images are handled by the web application. The images are stored in a firebase. Firebase provides an API to integrate web applications with its own JavaScript package. This web application uses the 'firebase' package configured with API keys stored in environment variables to connect and store images to the Firebase server.

4.2 ExpressJS REST API

4.2.1 Project directory structure

We have an ‘api’ folder to store the data of our backend. Here inside the ‘api’ folder we have two other folders which store the information of our backend. We have models and routes folder here in api. The model folder contains the schema for our database and in the routes folder it contains the routing information. it stores the information of POST , GET, PUT etc. methods. Here .env files store the tokens and other environment variable information.

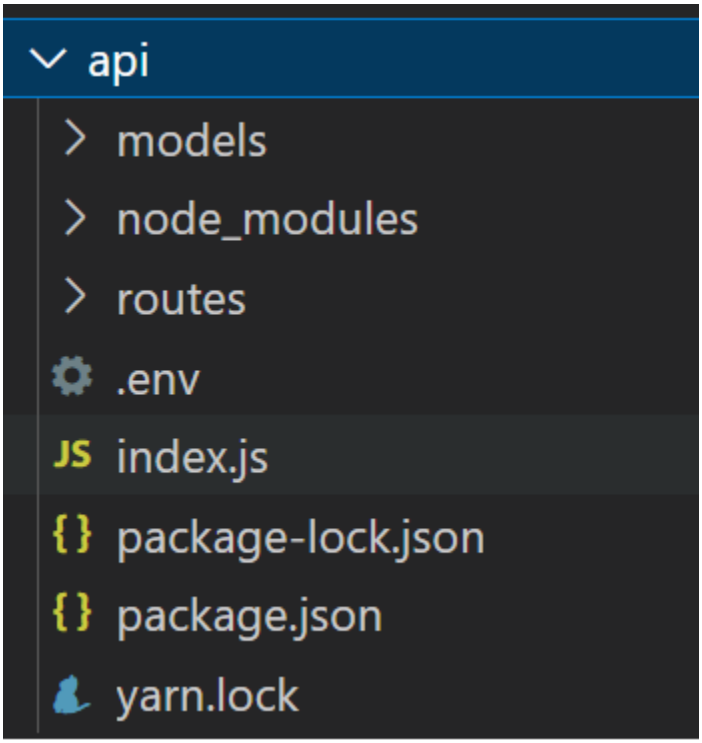


Figure 4.2 Project Directory Structure of backend

4.2.2 Server Configuration

Our REST API is built on an express web framework. This section briefly explains the configuration and extensions used in our application.

a) **MongoDB configuration:** We store our data in MongoDB . We communicate our application to the database using server.js. It allows us to access and query our remote database.
b) **API Authentication:** In order to use services on our server, the web application must be authenticated with a unique API key. The API key is a time-serialized token that is generated using the JWT (json web tokens) package. This ensures that clients must be authorized to send requests on our endpoints.

4.2.3 Endpoints

Our express application listens a total. These endpoints can be categorized into:

4.2.3.1 Authentication Route (/api/auth/):

a) **User Registration:** These endpoints provide user registration.

Register (" /api/auth/register")	
Method	POST

Request	{
	"username": String,
	"email": String,
	"password": String,
	"firstname": String,
	"lastname": String,
	}
		
		Required, Alphanumeric
		Required, regex: str@str.str
		Required, 8-50 characters

Response	Success	Fail

	{
	"success": true,
	"user": Object,
	"token": String,
	}
		"invalid data"
		Validation errors

Table 4.2.3.1 Project Directory Structure of backend

b) **User Login endpoints:** These endpoints validate user logins

Login (“api/auth/login”)		
Method	POST	

Request	{
	"email": String,
	"password": String,
	}
		
		(Either name OR
		password Required)
		Required

Response	Success	Fail

	{
	"success": true,
	"user": Object,
	"token": String,
	}
		"Invalid Email or Password"

Table 4.2.3.1 Project Directory Structure of backend

4.2.3.2 Product Route (/api/product/)

Product routes provide endpoints for different services retrieval methods.

Get all services (“api/product?keyword=<string> &category=<string>&price[gte]=<number>&&price[lte]= <number>”)		
Method	GET	
Response	Success	Fail
	Returns all products	
Get single Product (“api/product/:id”)		
Method	GET	

Response	Success	Fail
	get single product its full information	

CRUD route for one user parameterized by id:

Read:

Service By ID (“/api/product/:id”)		
Method	GET	
Parameter	_id	
Response	Success	Fail
	Returns service that matches given _id in parameter.	500

Post/write:

Method	POST
Parameter	_id

Request	{
	"title": String,
	"price": String,
	"description": String,
	"category": String,
	“images”: String
	}
		(All are optional)
		
		Firebase image reference string

Response	Success	Fail

Update:

Method	PUT	
Parameter	_id	

Request	{
	"title": String,
	"price": String,
	"description": String,
	"category": String,
	“images”: String
	}
		(All are optional)
		
		Firebase image reference string

Response	Success	Fail
	“service updated”	Validation errors

Delete:

Method	DELETE	
Parameter	_id	
Response	Success	Fail
	"Product deleted"	

Table 4.2.3.2 Product route(/api/product)

4.2.3.3 Orders Route (/api/order/)

Orders routes provide endpoints for different orders retrieval methods.

CRUD route for one user parameterized by id:

Read:

Service By ID (“/api/order/:id”)		
Method	GET	
Parameter	_id	
Response	Success	Fail
	Returns order that matches given _id in parameter.	500

Post/write(“/api/order/”):

Method	POST	
Parameter	_id	

Request	{
	"productid": String,
	"quantity": number,
	"amount": number,
	"status": String,
	"address": string,
	
	}
		(All are required)

Response	Success	Fail

Update:

Method	PUT	
Parameter	_id	

Request	{
	"order Status": String,
	}
		(All are required)

Response	Success	Fail
	“order updated”	Validation errors

Table 4.2.3.3 Order route(/api/order)

4.2.3.4 Cart Route (/api/cart/)

Cart routes provide endpoints for different cart items retrieval methods.

update my cart Items (“/cart/:id”)		
Method	PUT	
Response	Success	Fail
	Updateusers cart items.	
Delete cart item(“/cart/:id”)		
Method	DELETE	
Response	Success	Fail
	Item deleted	
Post new cart item(“/api/cart/”)		
Method	POST	
Response	Success	Fail
	Item added	

Table 4.2.3.4 Cart route(/api/cart)

Chapter 5 System Requirement Specification

5.1 Software Specification

5.1.1 Web Application technologies used

Name	Version
axios	^0.21.4
firebase	^9.1.0
express	^4.17.1
react	^17.0.2
redux	^7.2.5
@stripe/react-stripe-js	^2.6.3
recharts	^2.0.9

Table 5.1.1 Web Application technologies used

5.1.2 ExpressJS API Dependencies

ExpressJS Library	Version

express	^4.17.1
crypto-js	^4.1.1
nodemon	^2.0.12
Jsonwebtoken	^8.5.1
Mongoose	^6.0.5
cors	^2.8.5
Stripe	^8.174.0
dotenv	^13.0.0

Table 5.1.2 ExpressJS API Dependencies

5.1.2 Hardware Specification

5.1.2.1 Minimum Requirements

- **CPU:** Entry level SOC
- **RAM:** 2GB of RAM Recommended

5.1.2.2 Services and technologies used

- **Database Storage:** MongoDB

Chapter 6: Discussion on the Achievements

6.1 Features

6.1.1 Login and registration

Registration and login systems have been implemented.A user that has not been registered yet cannot login to the system.

6.1.2 Cart System

This service allows users to add services to cart so that they save it for later or order the service from there. It also allows users to delete the items in the cart. Summary of total cart items with total price is visible there.

6.1.3 Admin Dashboard

Only the accounts that are given admin access can access this admin panel.Here,the admins can add/edit or delete products on the store as well as add/edit or delete user profiles.

6.1.4 User-friendly Navigation

UI that is easy on the eye has been designed for easy navigation.Users can easily navigate through our e-store without any assistance.

Chapter 7: Conclusion and Recommendation

AttHire is an ecommerce website that brings our idea of creating an alternative and sustainable financial approach for Nepali people to wear the cloth they desire to express themselves.Be it a grand occasion or some small fest,here,the users can hire the clothes they want to wear for a certain period of time as well as rent out the ones they already have for some cash in exchange. The major goal of our project was to create a responsive and easy to navigate website for users as well as businesses.More features can still be added to make this website even better. However,this website provides a platform for users to rent clothes easily.

5.1 Limitation

AttHire has some limitations and shall be solved shortly:

- Admins cannot check individual user order details.
- All transaction records and product sale/buy reports are not developed.
- Analytics section can be developed to help admins track their products' performance.
- The sellers cannot communicate with the users.
- Rating of products cannot be done.

5.2 Future Enhancements

There is always room for improvement. Being a project with huge possibilities but relatively small number of team members there are many areas where enhancements can be made.

- Add extensive documentation for peoples wishing to derive from this open source project

- Email can be sent when registration to verify the new email address or reset forgotten password
- More Categories can be added
- Direct message system within the website using JavaScript libraries like socket IO
- Product video can be added to depict current product condition to the customers
- Upgrade Newsletter service so that customers can get update from their favorite products

References

[1] Shopify (2022). Retrieved July 15, 2022, from https://www.shopify.com/

[2] MongoDb Documentation(2022), Retrieved July 20, 2022, from https://www.mongodb.com/docs/

[3] Node Js Documentation(2022), Retrieved July 20, 2022, from https://nodejs.org/en/docs/

[4] Express Documentation(2022), Retrieved July 20, 2022, from https://expressjs.com/en/5x/api.html

[5] React Documentation(2022), Retrieved July 26, 2022, from https://reactjs.org/docs/getting-started.html

[6] Material UI(2022), Retrieved July 28, 2022, from https://mui.com/material-ui/getting-started/overview/

[7] Mohamedsamara/mern-e commerce(2022), Retrieved July 29, 2022, from https://github.com/mohamedsamara/mern-ecommerce

[8] Jigar-sable/flipkart-mern(2022), Retrieved July 31, 2022, from https://github.com/jigar-sable/flipkart-mern

Appendix

Appendix 1 - Project Account and Journal

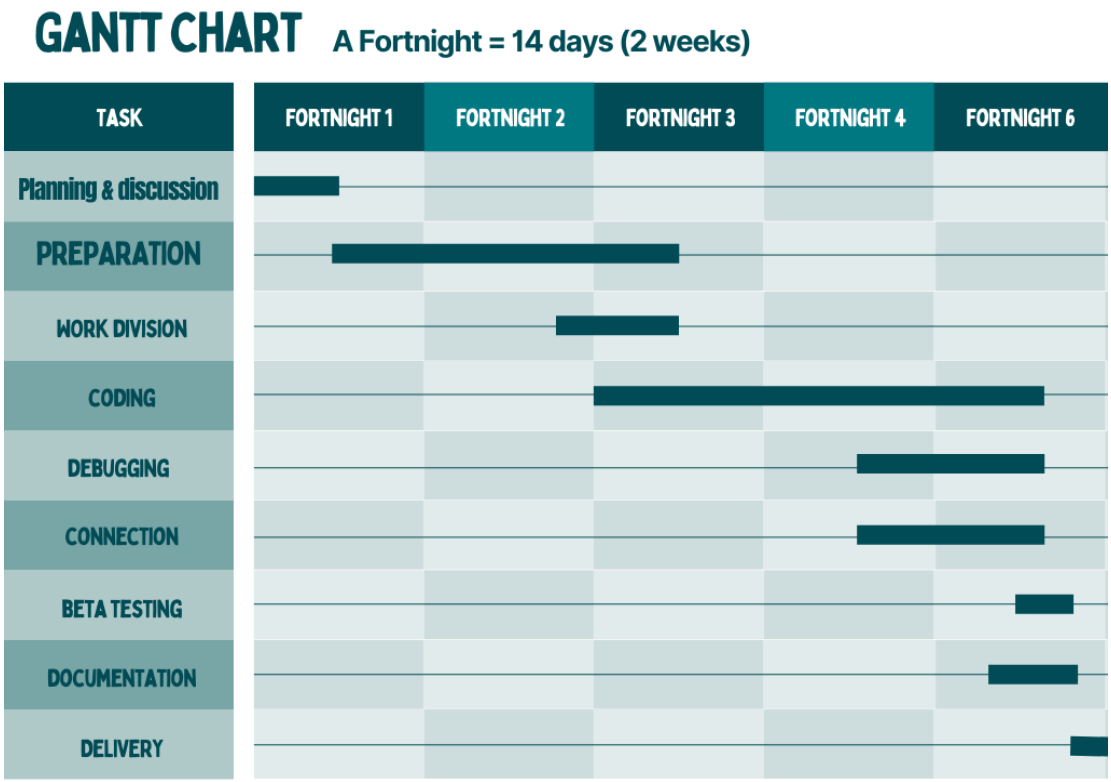


Table 9.1 GANTT Chart

Legend	
Tasks	**Sub-categories**
Project Proposal	Proposal Writing
Frontend	Login System
	Cart Design
	Web Application
Backend	Login System
	API and Users data Extraction.
Documentation	Report Writing
Further improvements	Information processing

Table 9.2 Legend

Appendix 2 - Work Division

Web Application:

Members	Landing Page	Login/ Profile	User interaction	Responsive UI	Product display page	Cart Design
Aakriti Banjara						
Sabin Bhattarai						
Sudip Subedi						

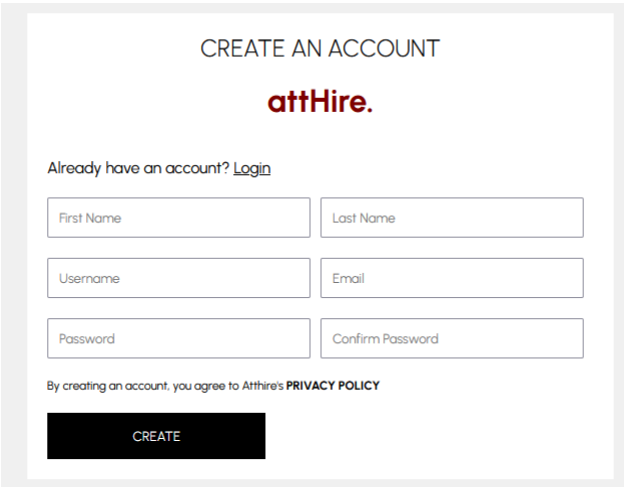
Table 9.3 Work division of Web Application

Node and ExpressJS API:

Member	Authentication	Services Processing	Filter	Product API	User API
Jiwan Humagain					
Anmol Karki					

Table 9.4 Work division of Microservice Backend

Appendix 3 – Screenshots



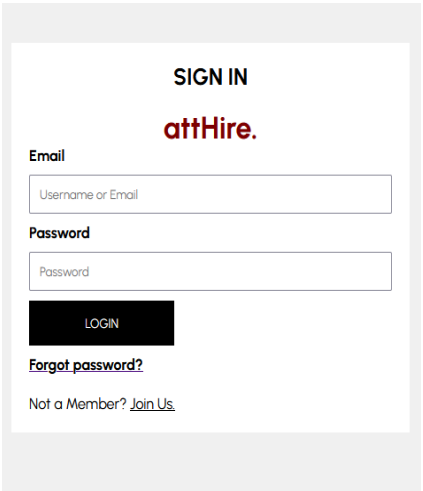


Figure 9.1 Sign up Page

Figure 9.2 Login Page

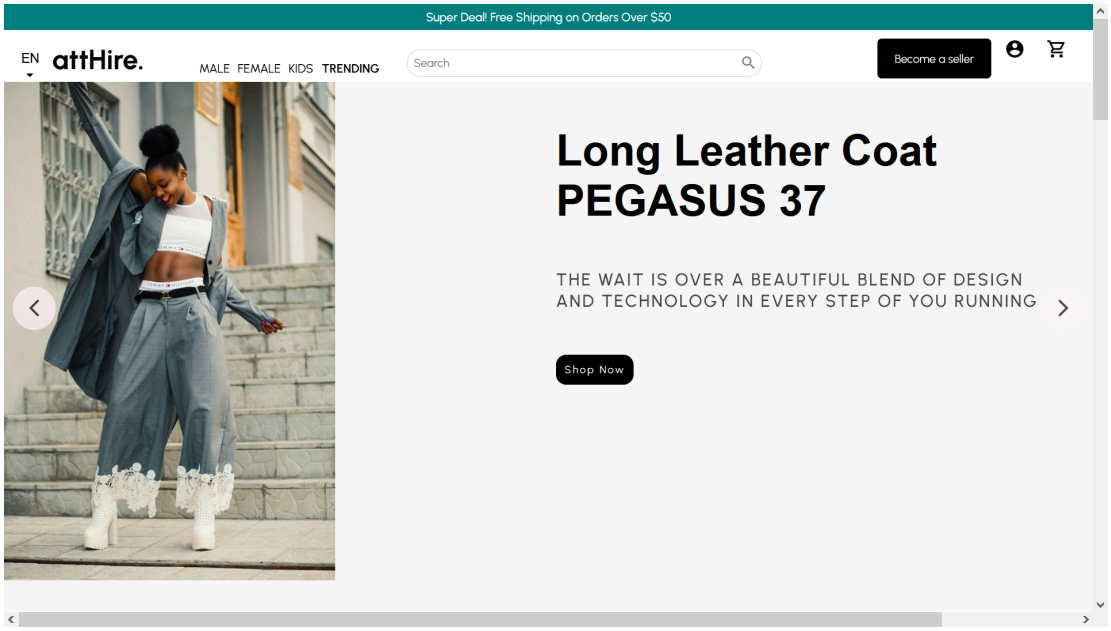


Figure 9.3 Main Page

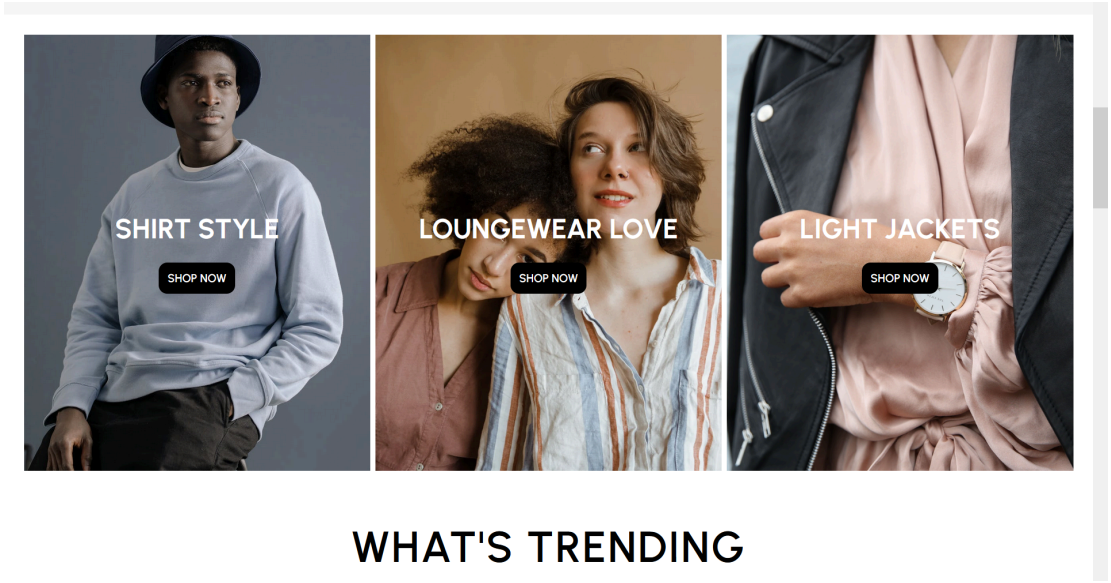


Figure 9.4 Main Page

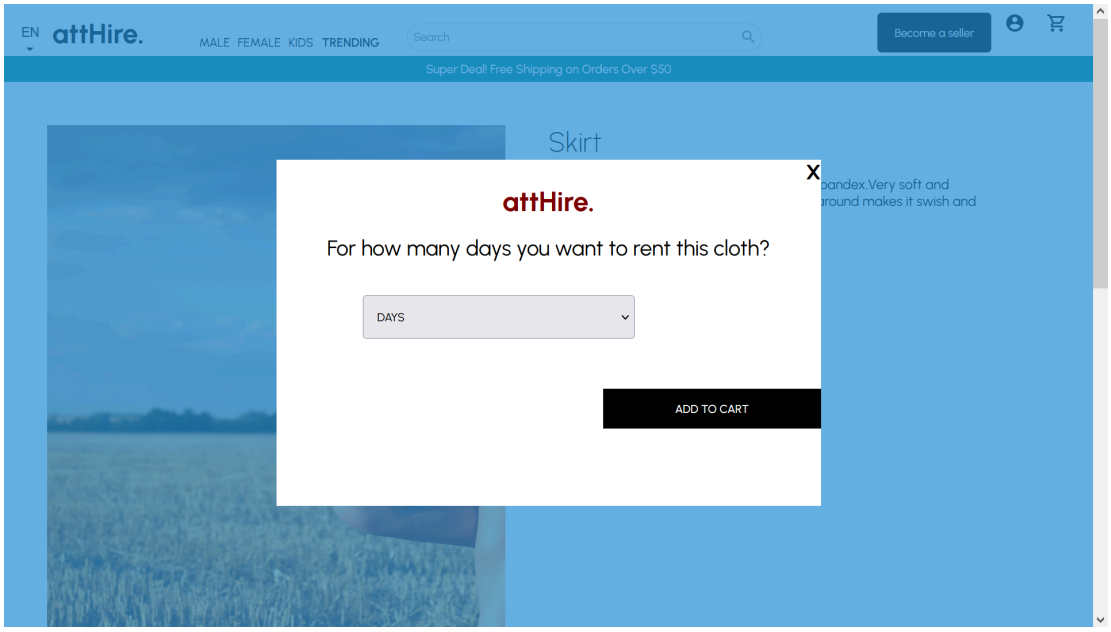


Figure 9.5 Product page asking the customer number of days to rent

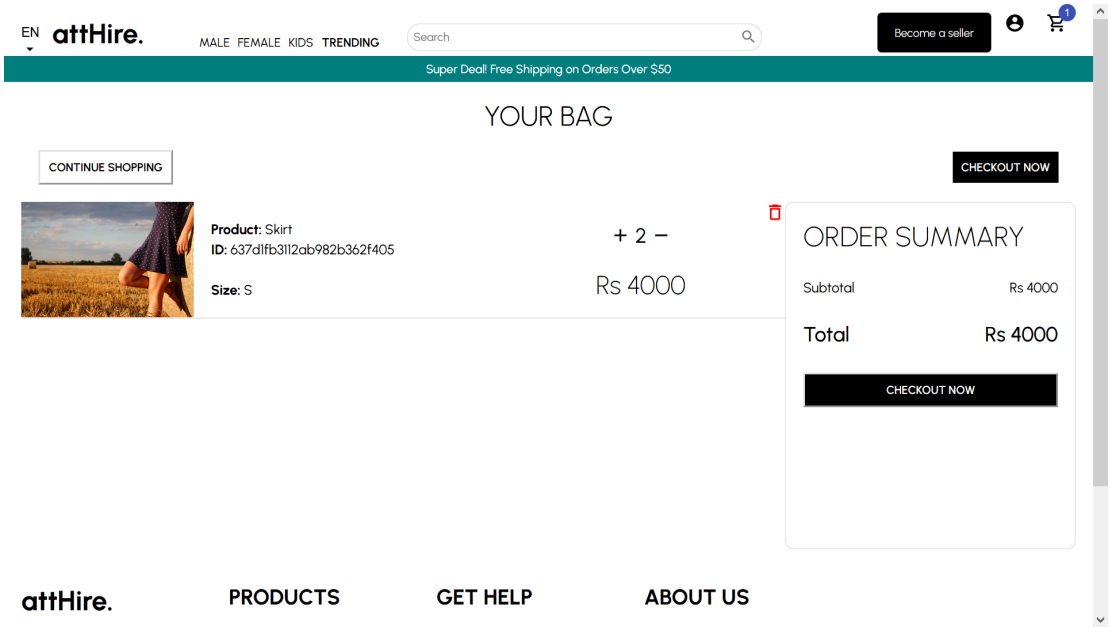


Figure 9.6 Cart Page

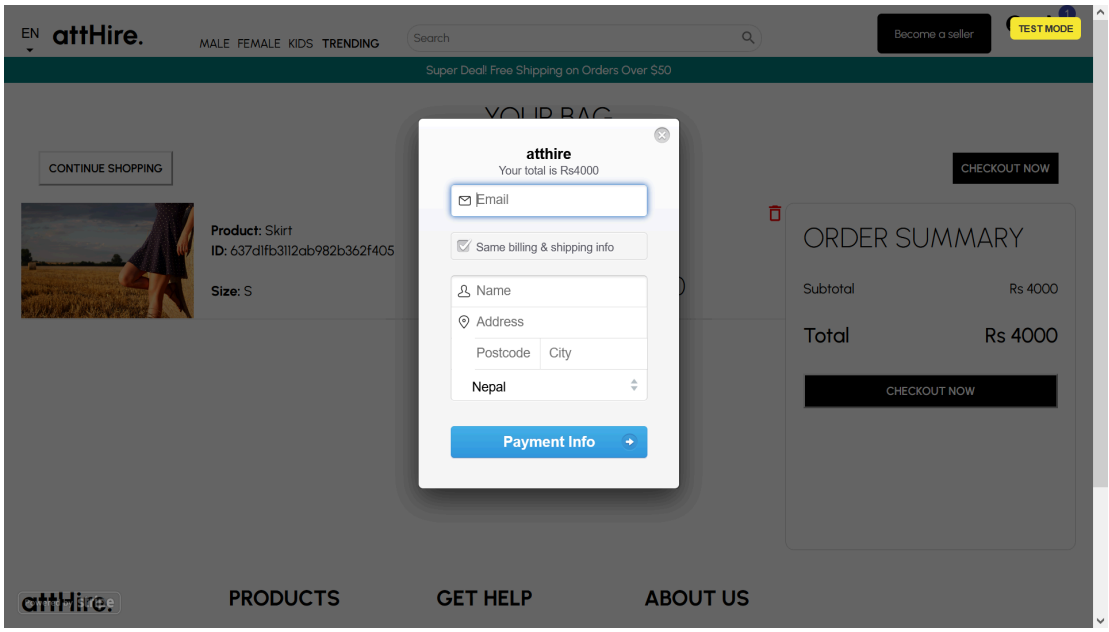


Figure 9.7 Payment Page for Buyer's info

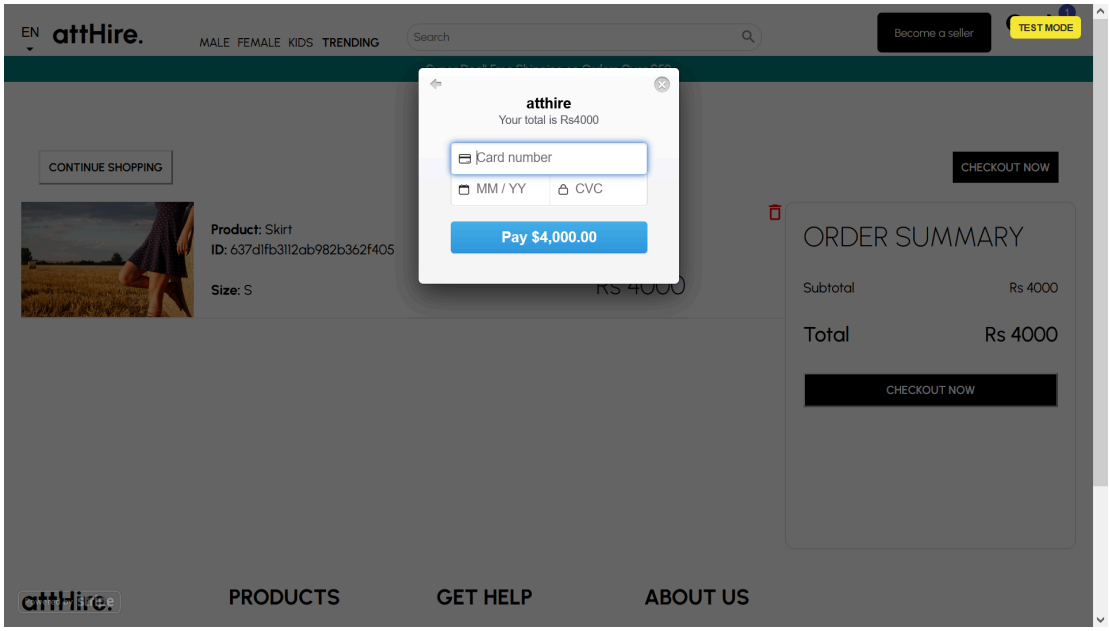


Figure 9.8 Payment Page for Buyer's card information

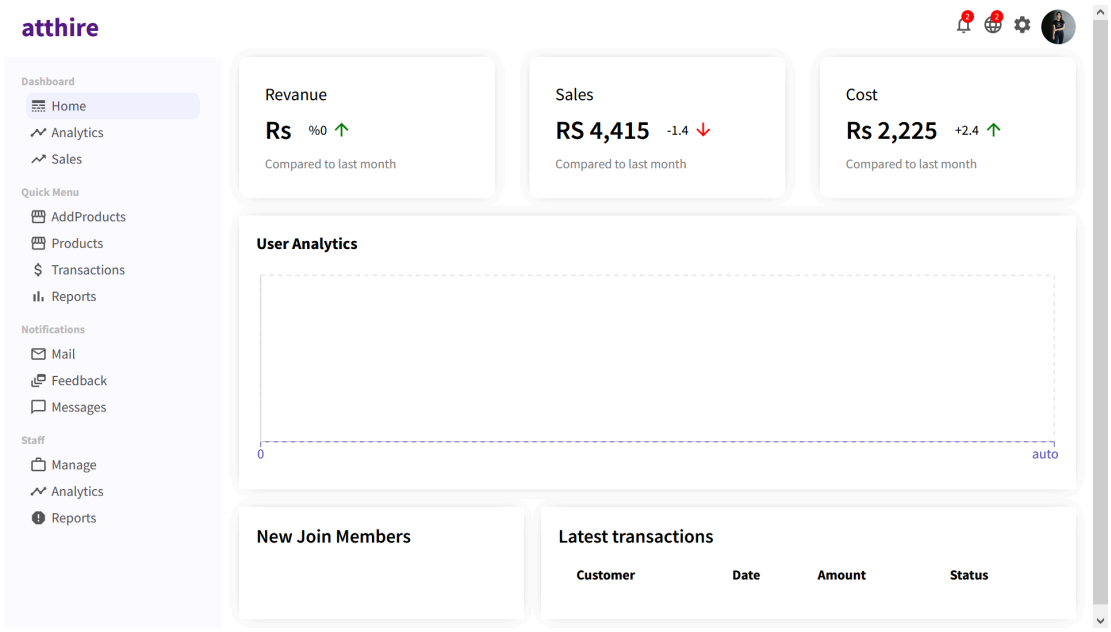


Figure 9.9 Admin Dashboard

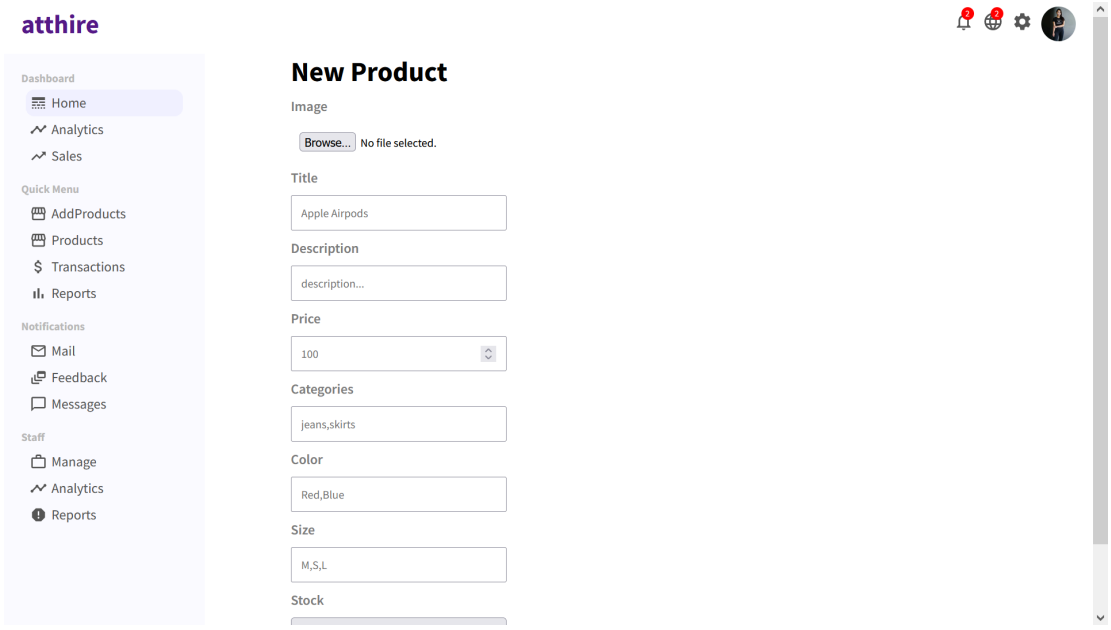


Figure 9.10 New product creating page

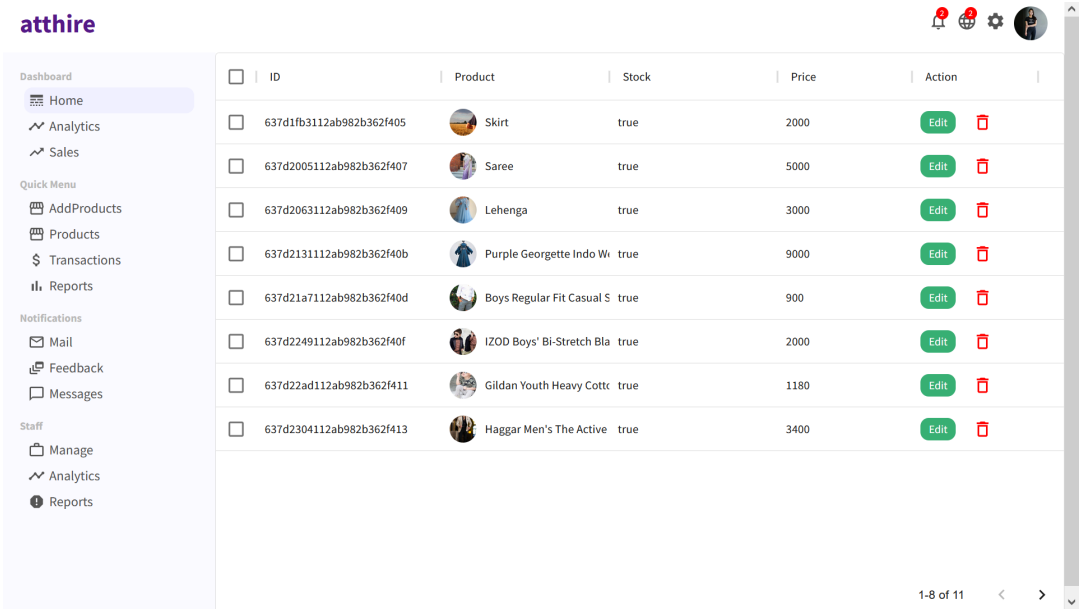


Figure 9.11 Product List