

“Rentish”

A

Project Report

Submitted in partial fulfillment of the requirement for the award of the degree of



Bachelor of Technology

In

Computer Science & Engineering

Submitted to

**RAJIV GANDHI PROUDHYOGIKI VISHWAVIDYALAYA,
BHOPAL (M.P.)**



Guided by:

Prof. Dharmendra Pathak

Submitted By

Hatim Saifee Press Wala (0832CS191074)

Karun Mourya (0832CS191088)

Khushhal Gupta (0832CS191089)

Murtaza Barwawala (0832CS191110)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHAMELI DEVI GROUP OF INSTITUTIONS

INDORE (M.P.) 452020

2021-22

““RENTISH”

A Minor Project

Software Requirement Specification Report submitted to

Rajiv Gandhi Proudhyogiki Vishwavidyalaya, Bhopal

Bachelor of Technology

in

Computer Science and Engineering

by

Hatim Saifee Press Wala (0832CS191074)

Karun Mourya (0832CS191088)

Khushhal Gupta (0832CS191089)

Murtaza Barwahwala (0832CS191110)

Under the guidance of

Prof. Dharmendra Pathak



Session: 2021-22

Department of Computer Science & Engineering

Chameli Devi Group of Institutions, Indore

452 020 (Madhya Pradesh)

DECLARATION

We certify that the work contained in this report is original and has been done by us under the guidance of my supervisor(s).

- a. The work has not been submitted to any other Institute for any degree or diploma.
- b. We have followed the guidelines provided by the Institute in preparing the report.
- c. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- d. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references.

Name and Signature of Project Team Members:

Sr. No.	Enrollment No.	Name of students	Signature of students
1.	0832CS191074	Hatim Saifree Press Wala	
2.	0832CS191088	Karun Mourya	
3.	0832CS191089	Khushhal Gupta	
4.	0832CS191110	Murtaza Barwawala	

CHAMELI DEVI GROUP OF INSTITUTIONS, INDORE



CERTIFICATE

Certified that the project report entitled, "Rentish" is a bonafide work done under my guidance Prof. Dharmendra Pathak (Hatim Saifee Press Wala, Karun Mourya, Khushhal Gupta, Murtaza Barwahwala) in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science Engineering.

Date:

(Prof. Dharmendra Pathak)

Guide

(Prof. Shailendra Kumar Mishra)

Head of the Department

(Mr. Vikrant Sharma)

Project Coordinator

(Dr. Manish Shrivastava)

(Principal, CDGI)

CHAMELI DEVI GROUP OF INSTITUTIONS, INDORE

ACKNOWLEDGEMENT

We have immense pleasure in expressing our sincerest and deepest sense of gratitude towards our guide Prof. Dharmendra Pathak for the assistance, valuable guidance and co- operation in carrying out this Project work. We are developing this project with the help of Faculty members of our institute and we are extremely grateful to all of them. We also take this opportunity to thank Head of the Department Prof. Shailendra Kumar Mishra, and Principal of Chameli Devi Group of Institutions, Dr. Manish Shrivastava, for providing the required facilities for the project work . We are greatly thankful to our parents, friends and faculty members for their motivation, guidance and help whenever needed.

Name and signature of team Members:





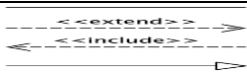












1. Hatim Saif Press Wala (0832CS191074)
2. Karun Mourya (0832CS191088)
3. Khushhal Gupta (0832CS191089)
4. Murtaza Barwawala (0832CS19110)

TABLE OF CONTENTS

CONTENTS	Page No.
Title Page	-
Declaration	I
Certificate by the Supervisor	II
Acknowledgement	III
List of Symbols	IV
List of Figures	V
List of Tables	VI
Abstract	VI
Chapter 1: Introduction	1.1
1.1 Rationale	1.2
1.2 Goal	1.2
1.3 Objective	1.3
1.4 Methodology	1.3
1.5 Role	1.5
1.6 Contribution of Project	1.5
1.6.1 Market Potential	1.5
1.6.2 Innovativeness	1.5
1.6.3 Usefulness	1.6
1.7 Report Organization	1.6
Chapter 2: System Features	2.1
2.1 Functional Requirement Collection	2.1
2.1.1 Distributed Database or Client Server Model	2.2
2.2 Non-Functional Requirement	2.4
2.2.1 Performance Requirement	2.4
2.2.2 Safety Requirement	2.4
2.2.3 Security Requirement	2.4
2.2.4 Software Quality Attributes	2.4
2.3 External Interface Requirement	2.5
2.3.1 User Interface	2.5
2.3.2 Hardware Interface	2.5
2.3.3 Software Interface	2.6

2.3.4 Communication Interface	2.7
Chapter 3: Analysis & Design	3.1
3.1 Use-case Diagrams	3.1
3.2 Activity Diagrams	3.3
3.3 Sequence Diagrams	3.5
3.4 Class Diagrams	3.7
3.5 E-R Diagram	3.8
3.6 System Architecture	3.9
Chapter 4: Construction	4.1
4.1 Implementation	4.1
4.1.1 Implementation Details	4.1
4.1.2 Database Tables	4.4
4.2 Testing	4.9
4.2.1 White Box Testing	4.9
4.2.2 Black Box Testing	4.9
4.2.3 Test Cases	4.9
Chapter 5: Conclusion	5.1
Chapter 6: Future Scope	6.1
References	7.1

List of Symbols

Serial no.	Symbols	Description	UML Diagram
1		Actor	Use case
2		Use cases	Use case
3		Initial state	Activity
4		Final state	Activity
5		Relationships	Use case
6		System Boundary	Use Case
7		Action State	Activity
8		Control flow	Activity
9		Decision Node	Activity
10		Fork	Activity
11		Join	Activity
12		Lifeline (actor)	Sequence
13		Lifeline	Sequence
14		Message	Sequence
15		Execution occurrence	Sequence
16		Self message	Sequence
17		Active class	Class

List of Figures

Figure no.	Description	Page no.
1.1	Agile Development	1.4
2.1	Client Server Model	2.3
3.1	Use Case diagram (Admin)	3.1
3.2	Use Case diagram (Customer)	3.2
3.3	Activity diagram (Admin)	3.3
3.4	Activity diagram (Customer)	3.4
3.5	Sequence diagram (Admin)	3.5
3.6	Sequence diagram (Customer)	3.6
3.7	Class diagram	3.7
3.8	ER diagram	3.8
3.9	System Architecture	3.9
4.1.1	Razorpay payment gateway	4.1
4.1.2	Checkout Address Form	4.1
4.1.3	Checkout Order Summary	4.2
4.1.4	Cart Page	4.2
4.1.5	Profile Page	4.3
4.1.6	Signup Page	4.3
4.2.1	Access Token Schema	4.4
4.2.2	Feedback Schema	4.4
4.2.3	Verification Token Schema	4.5
4.2.4	Product Schema	4.5
4.2.5	User Model Schema	4.6
4.2.6	Reset Token Schema	4.6
4.2.7	Refresh Token Schema	4.7

4.2.8	Query Schema	4.7
4.2.9	Order Schema	4.8

List of Tables

Figure no.	Description	Page no.
1.1	Role	1.5
4.1	Test Cases	4.9

Abstract

Our project is a rental website, a website that will rent you most of the general utility items and more. we named it Rentsih as it gives out the theme and general idea of this website. People need the costly thing for a short time and the best way through it is renting but even renting nowadays is typical. No one wants to go through all that trouble of renting in this technical world everyone accepts everything to happen with a click. Keeping this in mind Rentish is defined as a general website that makes renting things much more easy.

This project is all about helping people rent on the online platform applying much less effort than what is needed at present. It will have all the necessary functions which a renting website should possess.

This website keeps the user in a healthy environment with fewer problems than the outside world. It reduces the hard work user had to do to get something on rent by going to some shop and renting an item. Users will be able to rent an item with just a few clicks.

This is an idea to make user's life easy by giving them everything on a single platform. Here, the user will be able to find anything he wants. He would not have to surf the internet for different websites for different things this character increases its value hundred folds.

If someone goes for a start-up in some field but can't get much investment to buy all the requirements for a start-up. so, he or she can rent the required items.

In rentish users will even get complete packages like they will get whole bedroom furniture package or drawing room package etc. In this way, users won't have to rent everything they need one by one. They will be able to get everything at once.

Chapter - 1

Introduction

RENTISH is a website that will provide rental service for real estate property, vehicles, household appliances, furniture, electrical products etc.

On this website people who want to rent or want to explore the possibility of renting will visit. When they come they will land on the home page. where they will get to explore all the items they can rent. There will be an option to search where the user will get to search items according to their need. There will be a login or register page where new users will register themselves. Users will have to register themselves by filling in all their detail. From the home page, users can go to different sections of the fields which they want to explore. Their user can see all the details about the product with images. When the user finds the product they will be able to click on the rent button. From there they will be redirected to the rent page of that product which will show all product details with product ID and payment summary. On that page, users will have to mention the duration for which they want to rent the item. According to which their payable amount will be shown with deposit.

From there users can go back to exploring more suitable items or can click on confirm button. which will take them to a payment page where they will be able to select the payment method they prefer. After payment confirmation, they will land on confirm page.

After renting the item and using it the person will be able to go to the order menu and will be able to return it by clicking on the return button. If the rent duration ends and the user does not return the item he will be charged per day for it.

Users will be able to give feedback on the order by going on the feedback option on the item they have already rented. They will find the feedback option when he opens the product he already rented in my orders.

1.1 Rationale

Sometimes people need some item for a limited time.

examples:-

1. Like when they go to some other city for a short period but there they need all the basic utilities like tv fridge furniture etc. but he or she wouldn't want to invest that much amount of money just for a short period
2. When people can't afford to buy some things but they need it anyhow just for a little time. Most people face these kinds of problems.
3. Sometimes people need Fashionable and costly clothes just for a day. They don't want to spend that kind of money just for a day. Many individuals face this situation.

There are hundreds of situations like these for which we need some solution. So to tackle all this type problem here we have rentish. A website to rent Like many big websites which sell everything Like Flipkart and Amazon Rentish will rent everything online.

1.2 Goal

Making sure rentish fits within the S.M.A.R.T acronym helps to make it easier to understand that renting is better, and therefore, easier to make it a reality.

S = Specific

M = Measurable

A = Attainable

R = Realistic

T = Time-Related.

we want satisfaction for renting world. It will be a whole renting package in just a compact form User will get to rent anything he wants from a bicycle to a house from rentish in just a few clicks any item will be home delivered. People will not have to pay extra text if they will rent which will save them money.

1.3 Objective

Renting items have become an important factor in modern society hence the need to have a website for it. In India, everyone tries to find a way to save money By taking the best he can at least cost. Every Indian goes for the most efficient choice. And renting is most efficient for short period.

Study of already available open-source websites developed in metro cities and foreign countries. There are many other renting websites which give cars, bikes and other gives furniture and few gives appliances. we will study all these websites and give an advance combination in rentish. So we will Analyse the foreign website and modify it according to our local needs.

We will try to develop an efficient website that will be helpful to users and will bring out the best in the technical renting world.

1.4 Methodology

In our project, we will be using Agile methodology. In agile methodology, we do development and testing simultaneously. Agile methodologies attempt to produce the proper product through small cross-functional self-organizing teams that produce small pieces of functionality regularly, allowing for frequent customer input and course correction as needed.

Agile's advantages are directly related to its faster, lighter, and more involved mindset.

1. Faster is Better: One of the most significant advantages of Agile Methodology is its speed.
2. Customer satisfaction is rapid, continuous development and delivery of useful software.
3. Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
4. Product is developed fast and frequently delivered (weeks rather than months.)
5. A face-to-face conversation is the best form of communication.
6. It continuously gave attention to technical excellence and good design.

Disadvantages of Agile methodology:

1. It is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. Cost of Agile development methodology is slightly more as compared to other development methodologies.



Fig 1.1 Agile Development

1.5 ROLE

S. no	Name	Role/Working Modules
1	Hatim Saiffee Press Wala	Order/Invoice, Login/Registration Module
2	Karun Mourya	Customer and User Modules
3	Khushhal Gupta	Customer and User Modules
4	Murtaza Barwahwala	Order/Invoice, Login/Registration Module

Table : 1.1

1.6 CONTRIBUTION OF PROJECT

1.6.1 Market potential

Rentish can give out maximum market returns. There are big websites which give houses on rent and there are websites which give us a hotel.

Some websites give cars, bikes, and trucks on rent. Some websites give furniture on rent. There are websites for electrical appliances, and even there are websites for kitchen appliances.

Rentish is a combination of all these websites This is an idea to make user's life easy by giving them everything on a single platform. Here, the user will be able to find anything he wants. He would not have to surf the internet for different websites for different things this character increases its value hundred folds.

This website will even help in building other start-ups by decreasing the investment cost as the user can rent items he needs for a start-up instead of buying them.

1.6.2 Innovativeness

Rentish will be an innovative and unique website that helps the user in renting whatever he needs and whenever he needs.

This website will consist all the items that are rentable. If someone goes for a start-up in some field but can't get much investment to buy all the requirements for a start-up. so, he or she can rent the required items like office furniture or set up for some company start-up or kitchen accessories or dining furniture for restaurant start-ups till they can get stable economically in this way rentish will help build many start-ups and is a great step towards progressive and fulfilled future.

In rentish users will even get complete packages like they will get whole bedroom furniture package or drawing room package etc. In this way, users won't have to rent everything they need one by one. They will be able to get everything at once.

1.6.3 Usefulness

Currently, there are people across the globe who wants to rent something but can't find the proper means to do so. These include someone who wants to rent a piece of cloth to someone who wants to rent a house. people find it very difficult to find different platforms they can trust for renting different things without any other individual support. Already available websites can give you items on rent but a single platform or can't get you all things you need to rent.

Hence there is a need to update those websites according to today's technical world expectations.

We have many reasons to design Rentish firstly, there is no other website which provides all goods and content for rent. Secondly, it is easy to rent everything in one place. So the user will be much more satisfied.

1.7 Report Organization

Chapter 1

Introduction

This chapter comprises the main aim, motive, and introduction of the project. It comprises all the necessary details regarding the basic essence of the project.

Rationale : The set of reasons and the logical course of action along with the principles employed are also mentioned in this section.

Goal : Project goal describes the impact of the project: the long-term effects that should (also) be triggered. This reveals that the intended results at this higher level are in the main not concrete, nor can they be exclusively attributed to this project.

Objective : A summary of research done, review and in-depth analysis of a particular subject and is often used to help the reader quickly the purpose of the project.

Methodology : It introduces the topic of the project. It also contains the entry-level guide of all the functionalities in the project and their use.

Chapter 2

System features

It includes the process of determining user expectations for a new or modified website. These features, called requirements are relevant and detailed . Requirements Analysis define what the website is supposed to do. The software requirements enables users and serve as the basis for all the future design, coding, and testing that will be done on the project rrequirements are further defined through performance, look and feel, and other criteria.

Chapter 3

Analysis and design

The system design chapter is where the software functionalities plays a crucial role. The developer takes the output from the requirements stage, which states what the software is supposed to do, and defines how it should do it. This is a crucial stage for any software project because even the best programmers will have trouble implementing a poor design.

Chapter 4

Construction

The construction chapter is where the implementation and testing plays a crucial role. There is implementation of code that was used in the project and the database table that are used. The testing of funclitites of main modiules of our project.

Chapter 5

Conclusion

This chapter the basic summary of the project. It shows the result of the project and how it will solve our day to day problem. It consist the deduction of all the work done in completeing this project.

Chapter 6

Future Scope

This chapter includes the future scope of our project. What will be the steps we will taking to update this project. Here we will learn about the utilization of the project in the future web technology will overcome all other business module.

Chapter - 2

System Features

2.1 Functional Requirement

A functional requirement is a description of the basic nature of the system and the services which the system should provide. As mentioned in the name, functional requirement defines the functions of the system i.e., on providing a certain input what will be the nature (output, result, behavior) of the system. It describes the functions(components) used in the system manufacture and is the basic step which a system should met for efficient working of the system and if it does not match the functional requirement then the system will not work efficiently. Following are the functional requirements:

1. 3rd Party Package Integration: In our rental website, we will be using third party software such as Payment Gateway, NPM Packages, Google API's and many more. Deciding the right third-party integrations will make the website structured and ready for business in the future.
2. Mobile Responsive: In our website, we will be using Bootstrap which is a CSS Framework and is highly responsive and is widely used framework in corporate world. Therefore, it will be responsive for various width and height of devices.
3. ID Proof Verification: In our website, we will be adding a feature for ID proof verification of customers such that there will be no false customers. This verification will be done by Admin and will be able to remove false(irrational) customers.
4. Shipping and Payment System Integration: In our website, we will be integrating payment gateway which will give customer various choices(methods) in payment and there will be various methods available for delivery of goods as well.
5. Email Newsletter: In our website, we will be using SMTP protocol and node mailer package of NPM to send mails to the customer which is very useful for marketing requirements as well as for verification purposes. Email will be sent to customer on placing the orders, discount offers, new products and discount coupons for individuals as well.
6. Social Proof: In our website, we will be using rating as well as feedback system so that other customers will be able to know about the quality and the usefulness of a particular product. This will help in improving user confidence and satisfy their needs.

7. Relevant and Useful content: In our website, we will be providing various filters according to the category, sub-category, price(low to high) of product and will also add search functionality which will provide only relevant and useful content to the users.
8. Accessibility: Since we will be hosting our website, it will be accessible all around the world uninterruptedly at any time.
9. Easy to use: In our website, we will ensure that there will be no unnecessary steps involved in checkout flow and strive for a one click experience.
10. Speed: We will use the best web hosting services to make sure that the speed of the website is as fast as possible.
11. Social Sharing: In our website, we will allow the customer to share our website as well as products to various social media platform as it will increase the present as well as attract potential customers.

2.1.1 Distributed Database or Client Server Model:

As it is a rental website it will be using distributed database and client server model for various API calls explained below:

Distributed Database: A distributed database is basically a type of database which is not limited to one computer system instead it is distributed(spread) over different sites(computer systems) or over a network of computers. As it is a database which is distributed to various locations and does not share any physical components, it is required when a variety of users is accessing the website globally. We are using distributed database in our website because of following advantages:

- a. Performance is Improved: We will be able to achieve parallelism by executing different queries at different locations by breaking it into number of subqueries which basically leads to improvement in performance.
- b. Expansion is Easier: In distributed environment, expansion of the system in terms of adding more data, increasing database size or increasing computational speed by increasing the number of processors is much easier.
- c. Reliability and Availability is increased: If a data and DBMS software is distributed to various location then if one site fails then the other sites will be able to operate and we will only not able to access the data which was stored on that site and this basically will lead us to improvement in reliability and performance.
- d. Different levels of Transparency: Different levels of transparency means that hiding the details of where each file is physically stored within the database system. It will increase the security of the data stored in database and is reliable.

Keeping in mind the advantages and disadvantages of distributed database, we will be using MongoDB for the database operations and purposes. MongoDB is an open source, document-based Database Management System(Tool) that stores data in JSON-like format and is highly scalable, flexible and distributed NoSQL database.

Client Server Model: Client-Server denotes that a relationship exists between the program in an application and is composed of clients initiating a request for services and servers providing the result of that request. The client server model or client server architecture is a distributed application framework that divides the task between clients and server which either resides in the same system or communicates through computer network or the internet. The server runs on one or more programs that share resources with and distribute work among clients. Client-server communication typically relies on the TCP/IP protocol suite. TCP protocol maintains a connection until the client and server have completed the message exchange. There are four main categories of client server architecture but we will be using Three-Tier Architecture for our web application which consists of a presentation layer(Client side), Login layer(server side) and Data layer(Database side).

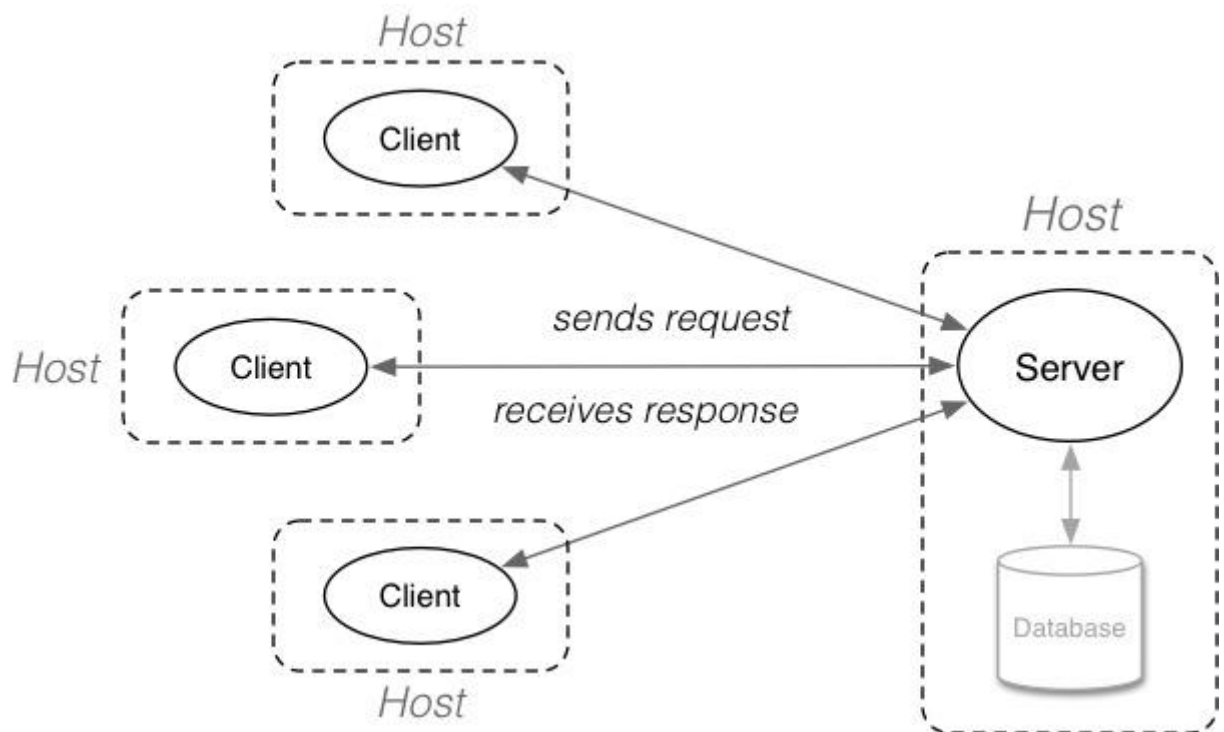


Fig : 2.1 Client Server Model

2.2 Non-Functional Requirement

2.2.1 Performance Requirements:

We will have to make the system respond quickly according to the input given to it.

The speed of the connection will depend on the hardware used rather than the characteristics of this system.

We will have to put the system on a server with high internet speed.

2.2.2 Safety requirements:

A firewall is a hardware or software system that essentially works as a wall or gateway between two or more networks, permitting authorized traffic and blocking unauthorized or potentially malicious traffic from accessing a network or system.

With an application gateway in place, there are two lines of communication: one between your computer and the proxy, then one between the proxy and the destination computer or network. Proxy firewalls are among the most secure.

2.2.3 Security Requirements:

Data integrity will be checked by the system for critical values.

The system will have its own authentication process to prevent unauthorized access.

Users will be sent a user ID and Password on their mail.

Sessions are maintained throughout.

2.2.4 Software Quality Attributes:

Efficiency : This system will have efficiency as it requires a fewer amount of computing resources and will have high throughput.

Reusability : This system will use some Java APIs and Packages, so the concept of reusability will be implemented.

Availability : The software Once uploaded on the web server, can be accessed 24X7 throughout the world.

Adaptability : This system will be able to work on low configured as well as high configured computers, having the old or latest version of the software.

Correctness : Accurate test results and information provided by this system.

Robustness : This system will provide some sort of robustness i.e., if the test module will fail then the searching module will have no effect on it and would be usable.

Flexibility : This system will be flexible enough to extend up to new updated versions and changes.

Maintainability : Backup of data will be taken at a certain time and all the records and database will be maintained properly.

Reliability : Every software module will be well tested against different test cases, so the system will be reliable.

Testability : Proper test cases will be designed to test each and every module of the software, this will show the concept of testability.

Usability : With the user-friendly GUIs, this website will be easy to use by admin and users.

2.3 External Interface Requirement:

This section specifies the hardware, software or database element with which a system or component must interface. This section also provides information to ensure that the system will communicate properly with external components.

2.3.1 User Interface:

The application will be accessed through a web browser Interface and would be viewed best using 1024 x 768 and 800 x 600 pixels resolution. The software would be fully compatible with Microsoft Internet Explorer for version 6 or above and with Google Chrome v36 or above.

2.3.2 Hardware Interface:

Development End

Hardware Specifications (Recommended):

Processor : i3 Processor

Storage : 10 GB

RAM : 4 GB

Software Specifications (Recommended):

Operating System : Windows 10

Browser : Chrome Browser Version 96.0

Designing Tool : VS Code

Front End : React Version 17.0.2

Rentish

Back End : Express.js Version 4.17.1, Node.js Version 16.13.0

Database : MongoDB Version 4.4

Deployment End

Server Side

Hardware Specifications(Recommended):

Processor : Quad Core Processor

Storage : 50 GB

RAM : 4 GB

Software Specifications(Recommended):

Operating System : Windows 10

Browser : Chrome Version 96.0

DBMS : MongoDB Version 4.4

Client Side

Hardware Specifications(Recommended):

Processor : Dual Core Processor

Storage : 10 GB

RAM : 512 MB

Software Specifications(Recommended):

Operating System : Windows 10

Browser : Chrome Browser 96.0

2.3.3 Software Interface:

For Back-end: Express.js, Node.js, MongoDB

For Front-end: React, Bootstrap, VS Code Code Editor v1.62 for project implementation.

Operating System : Window 7 or higher.

Web Browser : Mozilla/Google Chrome/Internet Explorer

2.3.4 Communication Interface

In this SRS describes the interface requirements for the system. This is usually designed to a specific standard that enables one machine to telecommunicate with another machine.

The Customer must connect to the Internet to access the Website: Dialup or Broadband Connection or WIFI (52 kbps min) with Internet Service

Chapter - 3

Analysis & Design

3.1 Use Case Diagram

a) Admin

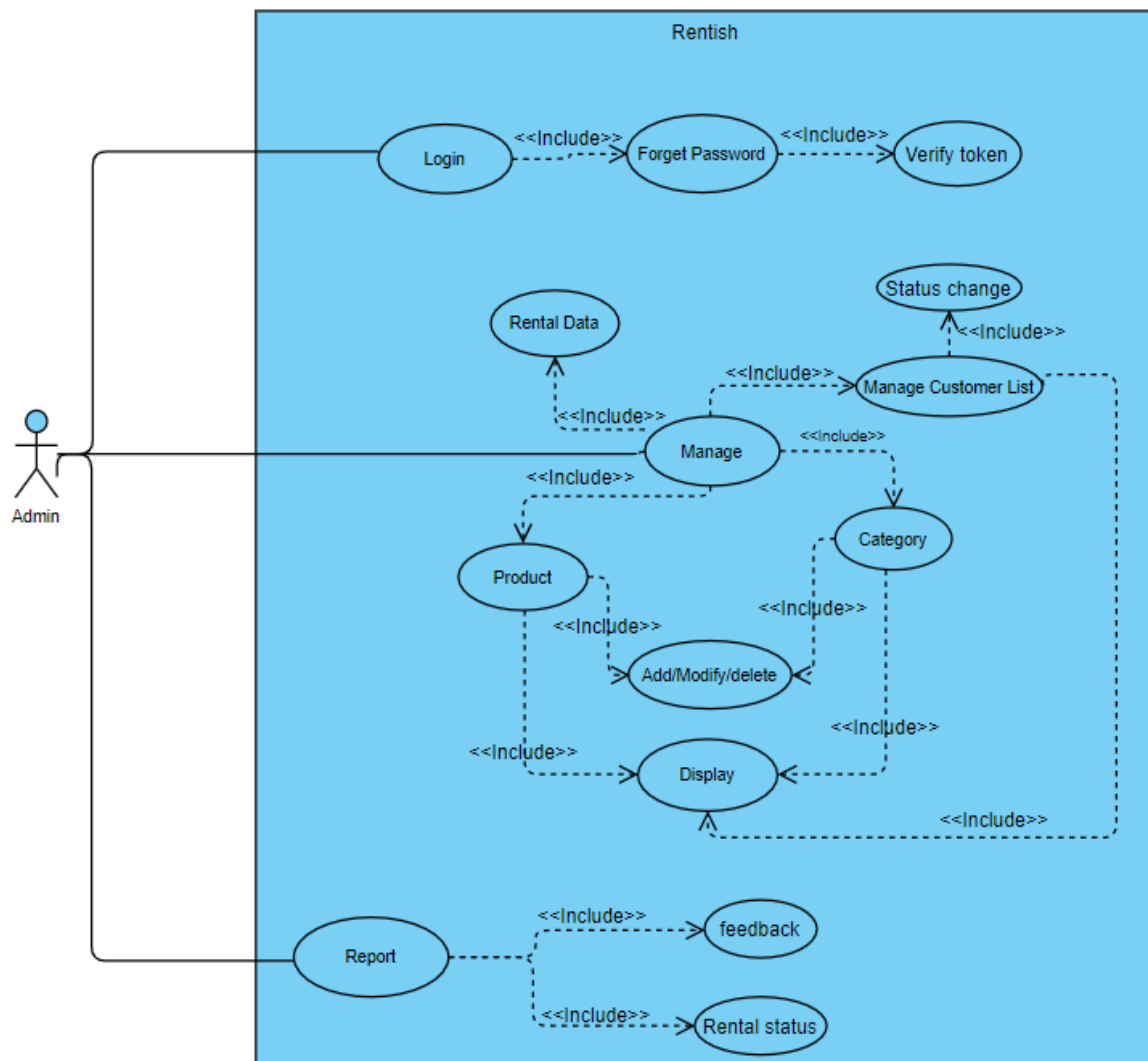


Fig 3.1 : Use case (Admin)

b) Customer

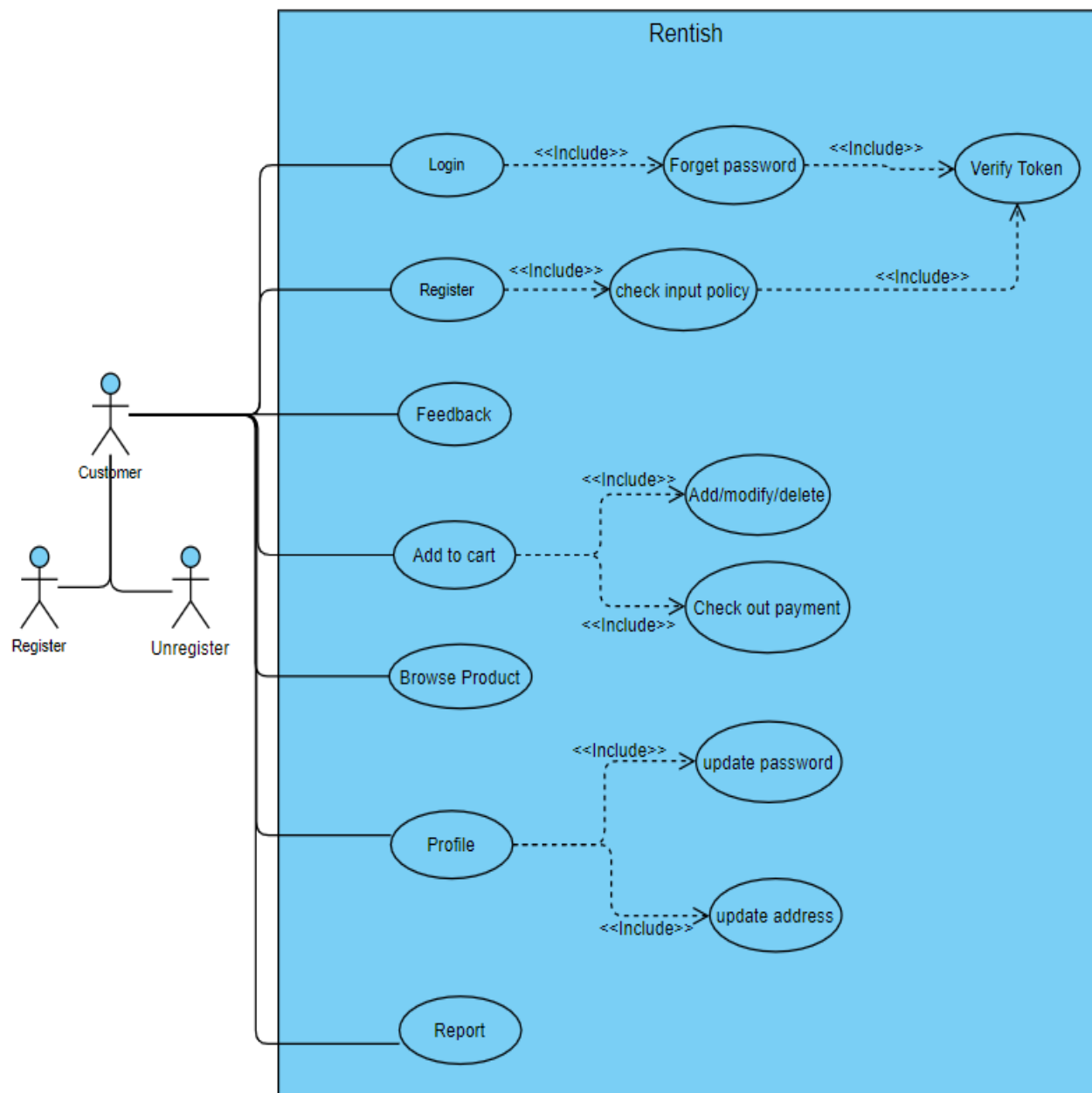


Fig 3.2 : Use case (Customer)

3.2 Activity Diagram

a) Admin

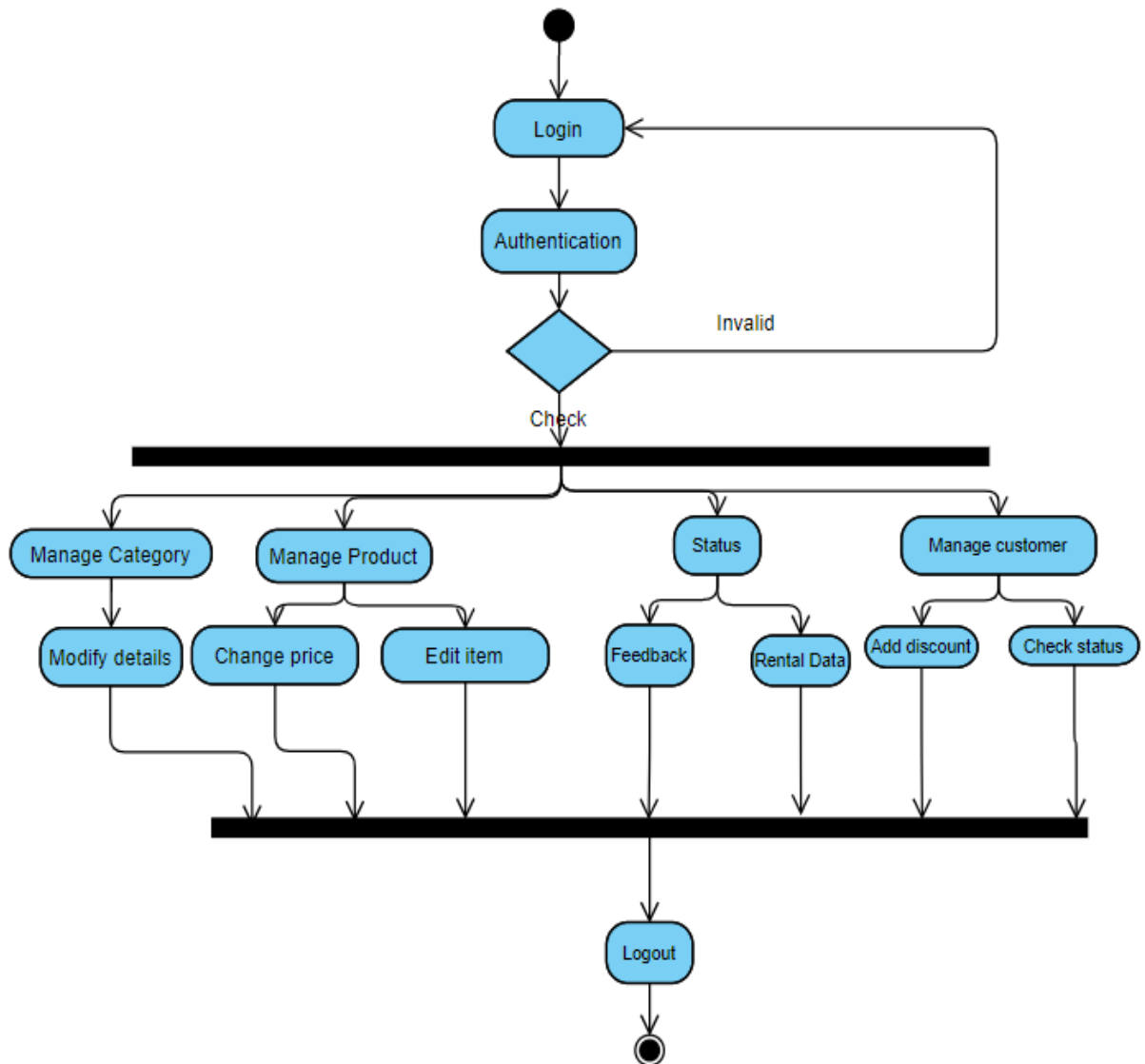


Fig 3.3 : Activity (Admin)

b) Customer

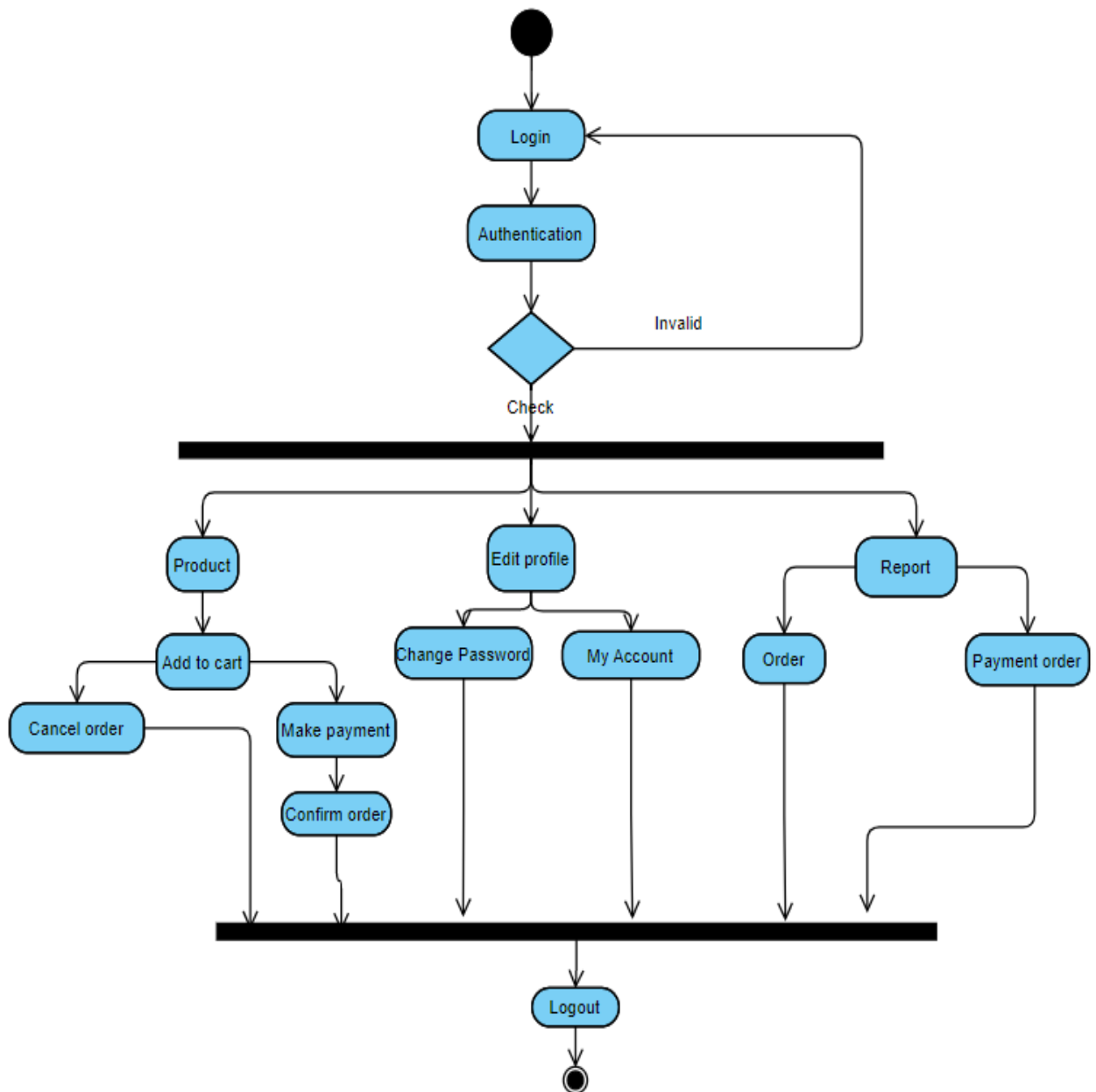


Fig 3.4 : Activity (Customer)

3.3 Sequence Diagram

a) Admin

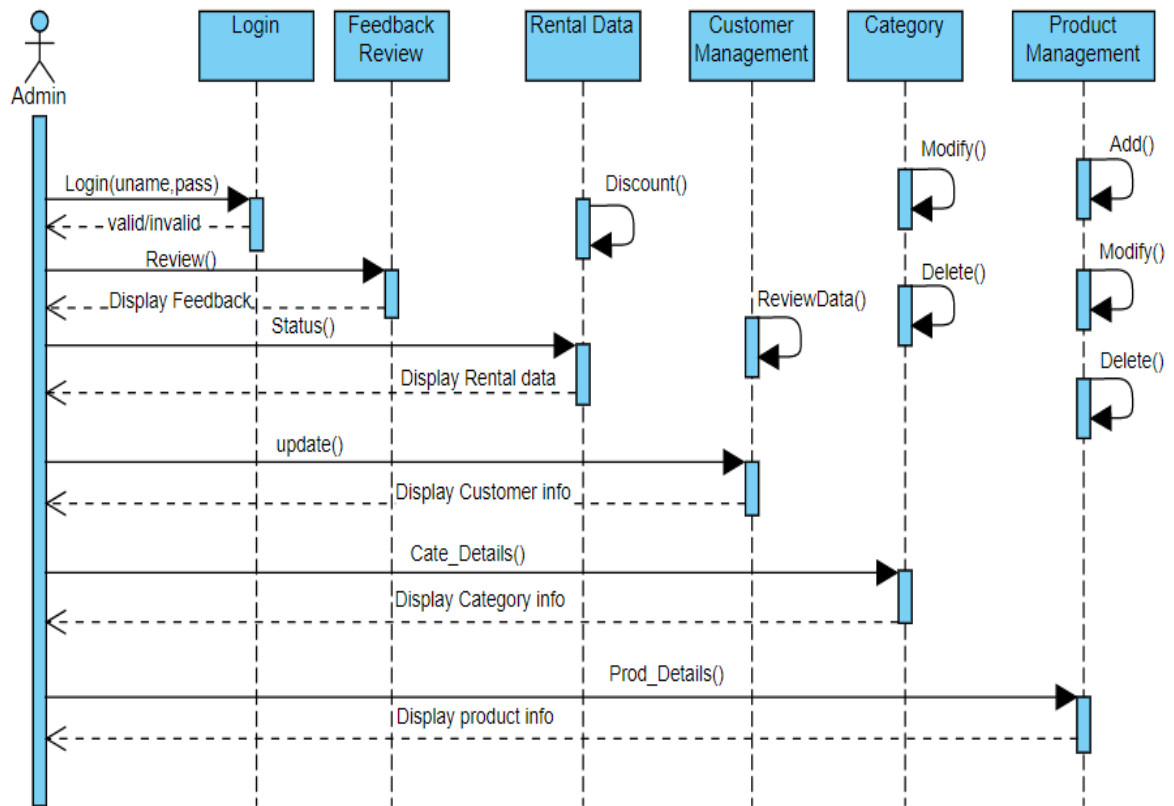


Fig 3.5 : Sequence Diagram (Admin)

b) Customer

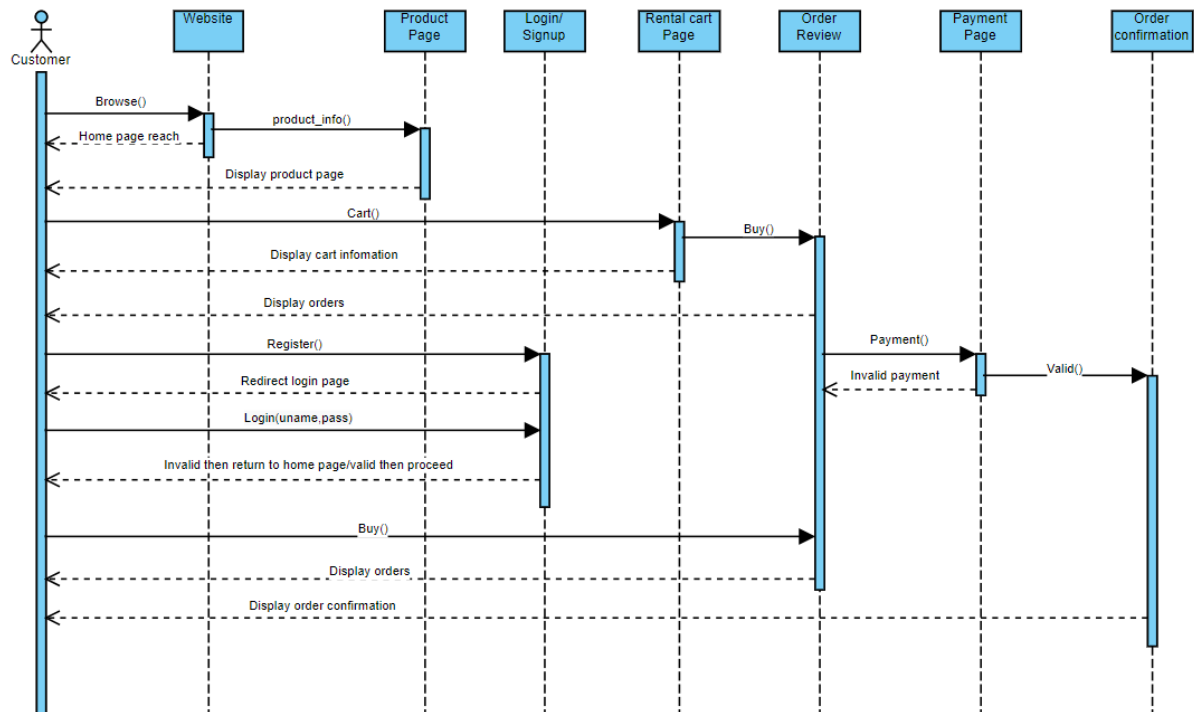


Fig 3.6 : Sequence Diagram (Customer)

3.4 Class Diagram

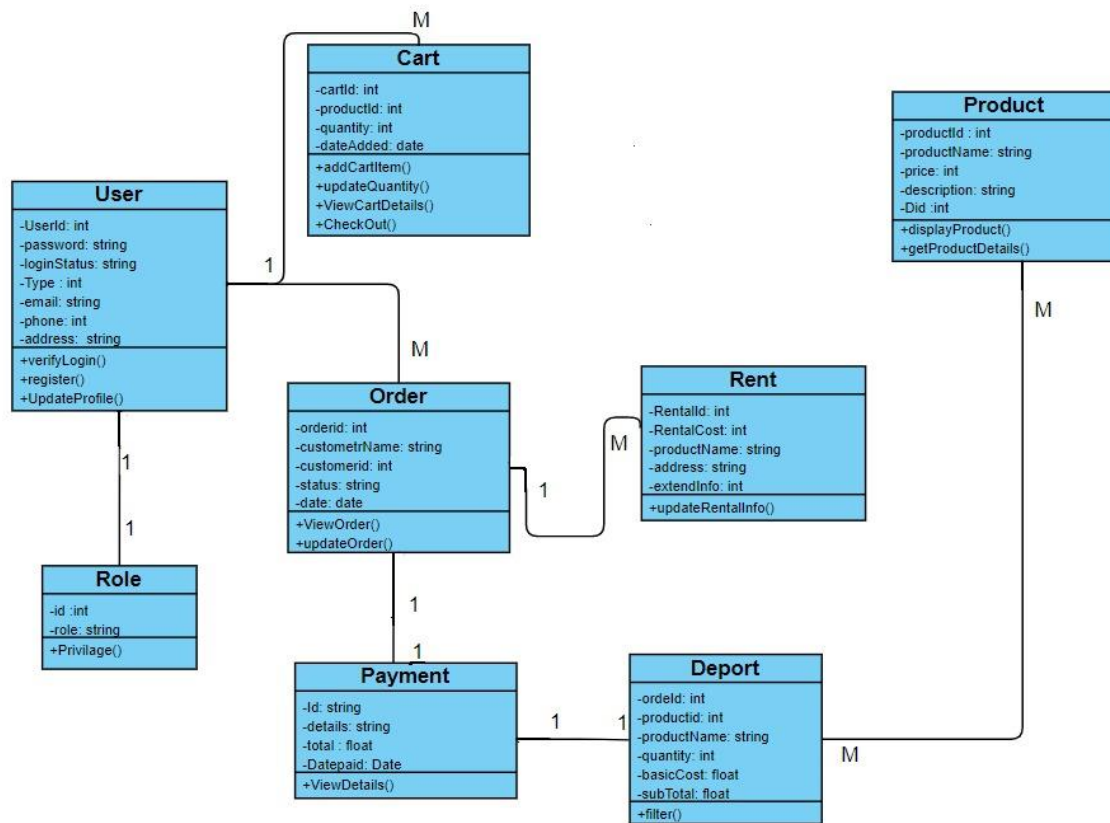


Fig 3.7 : Class Diagram

3.5 ER Diagram

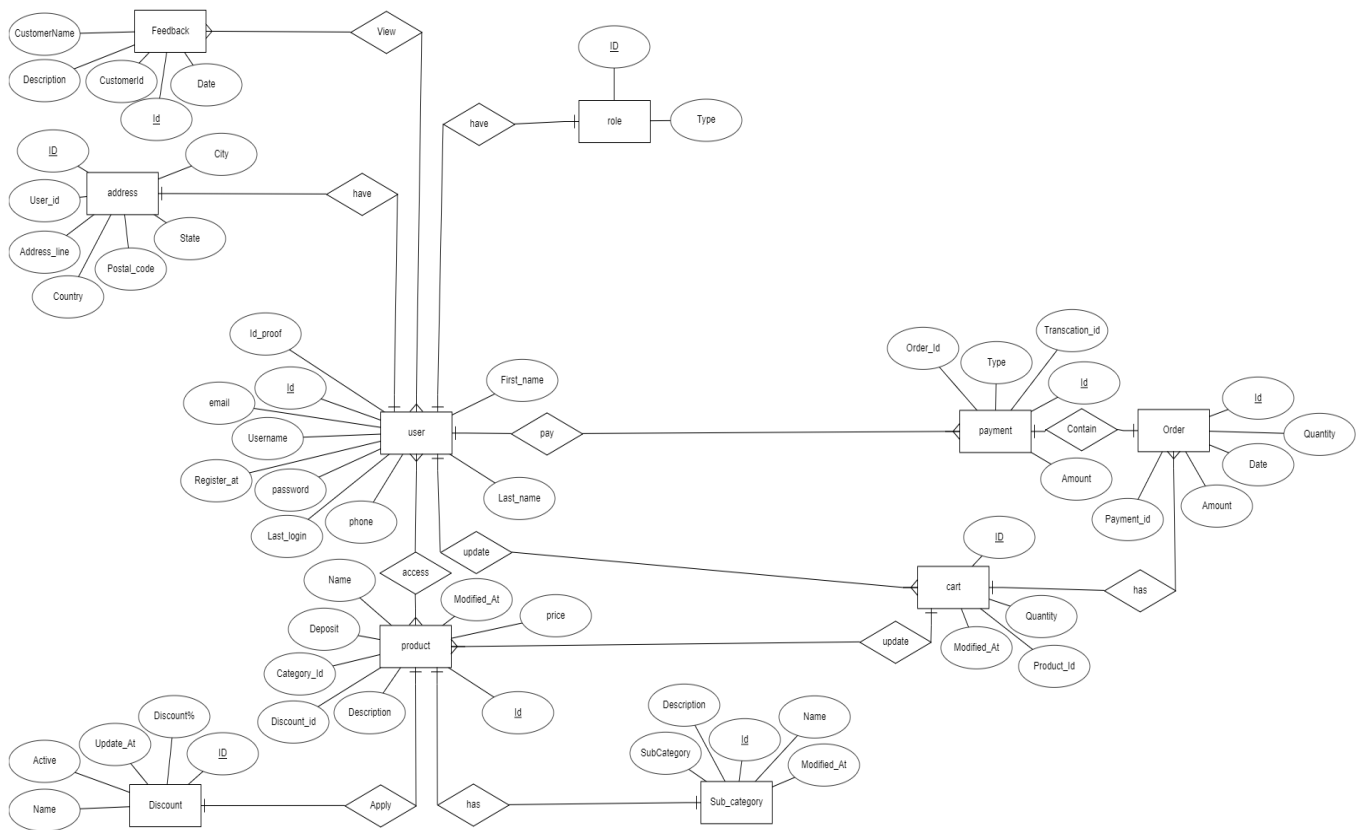


Fig 3.8 : ER Diagram

3.6 System Architecture

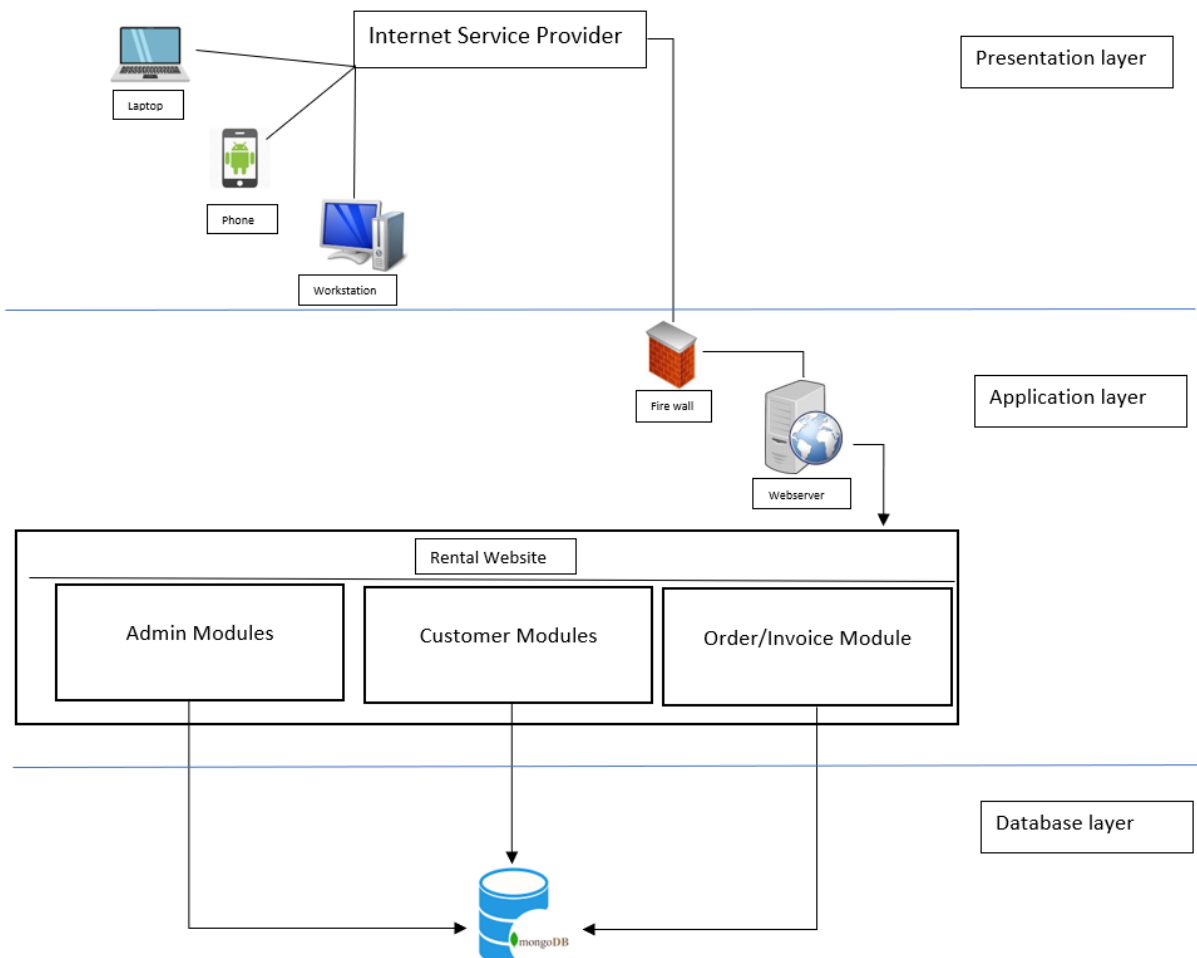


Fig 3.9 : System Architecture Diagram

Chapter - 4

Construction

4.1 Implementation

4.1.1 Implemented Project

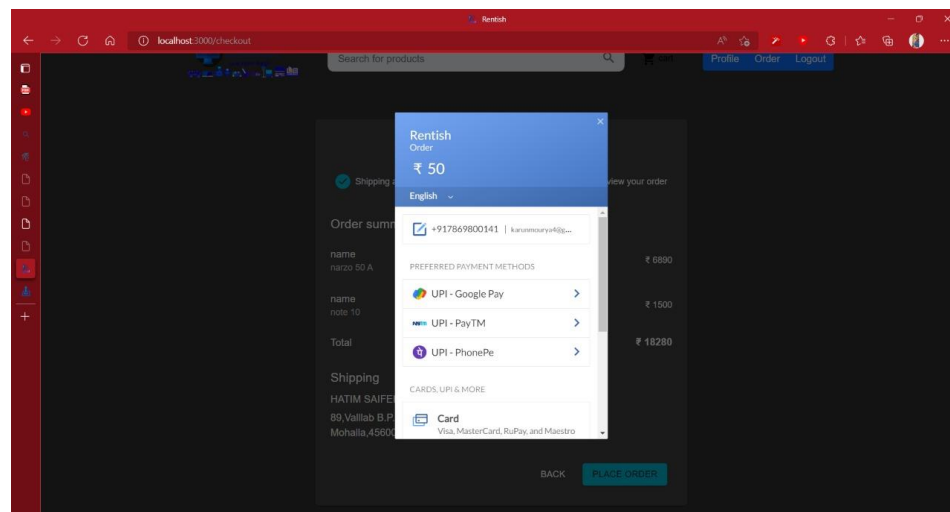


Fig : 4.1.1 Razorpay payment gateway

A screenshot of the 'Checkout' page on the 'Rentish' website. The browser address bar shows 'localhost:3000/checkout'. The page has a dark theme with a red sidebar. The 'Checkout' section is active, showing a progress bar with 'Shipping address' as the first step and 'Review your order' as the second. Below the progress bar, the 'Shipping address' form is displayed. It includes fields for 'First name*', 'Last name*', 'Address Line 1*', 'Address Line 2', 'City*', 'State/Province/Region' (a dropdown menu), 'Zipcode*', and 'Country*' (a dropdown menu). At the bottom of the form, there is a checkbox labeled 'Use this address for payment details'.

Fig : 4.1.2 Checkout Address Form

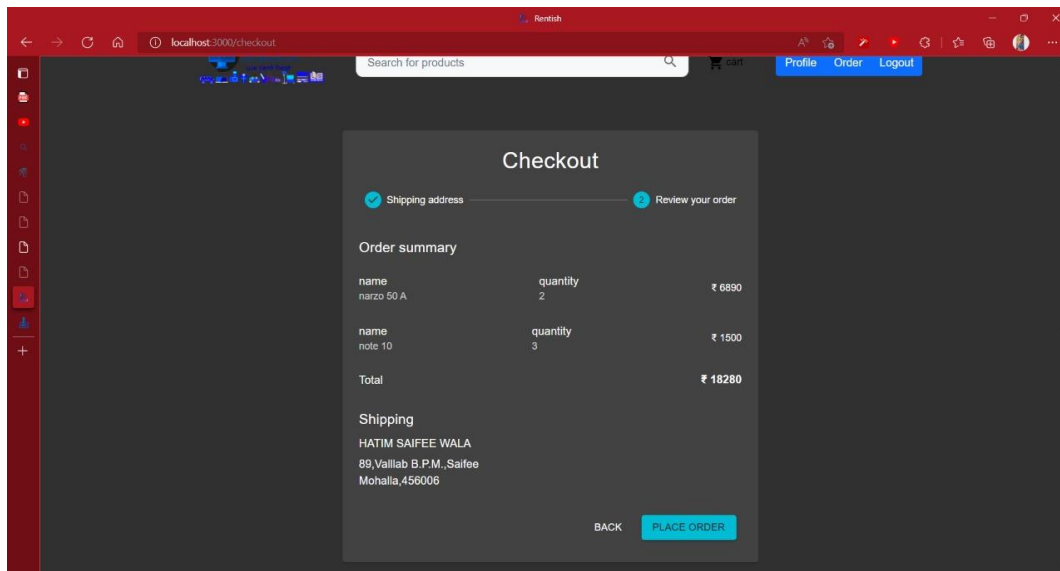


Fig : 4.1.3 Checkuot Order summary page

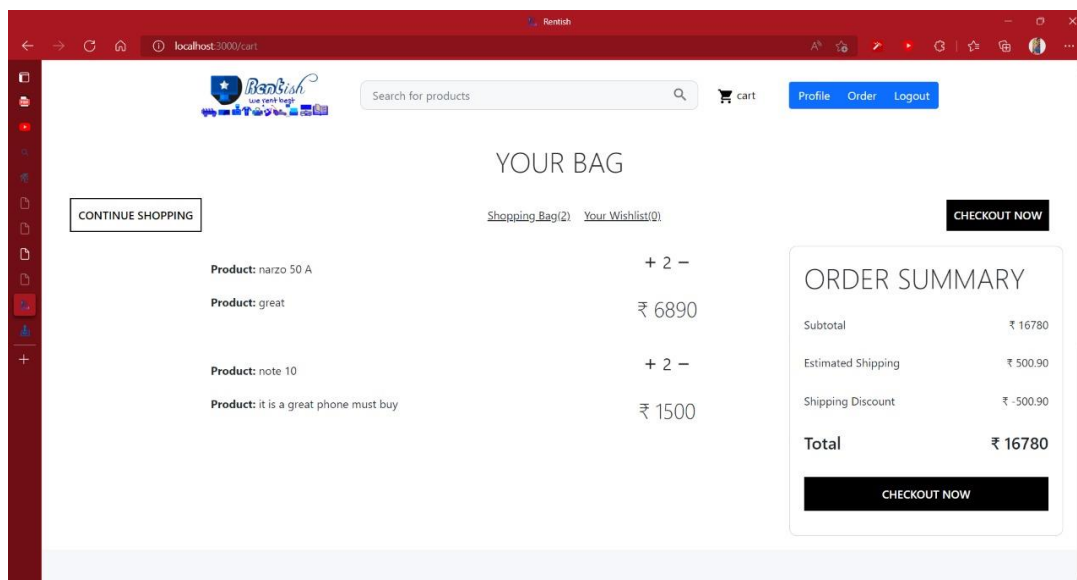


Fig : 4.1.4 Cart Page

The screenshot shows a web browser window with the URL `localhost:3000/profile`. The page title is "Your profile". It features a sidebar on the left with various icons. The main content area has two sections: "Personal Details" and "Change Password".

Personal Details

Name	Email
<input type="text"/>	<input type="text" value="hatimsaifee12345@gmail.com"/>
Phone Number	Address
<input type="text"/>	<input type="text" value="89,Valllab B.P.M.,Saifee Mohalla,456006"/>

[Update Details](#)

Change Password

New Password	Confirm New Password
<input type="password"/>	<input type="password"/>

[Update Password](#)

Email Verification

Fig : 4.1.5 Profile Page

The screenshot shows a web browser window with the URL `localhost:3000/login`. The page features a large orange banner on the left with the text "Welcome Back!" and "To keep connected with us please login with your personal info". A "Sign In" button is located below the banner. On the right, there is a "Create Account" form with fields for Full Name, Email, Password, Phone Number, and Address. A "SIGN UP" button is at the bottom of the form.

Welcome Back!
To keep connected with us please login with your personal info
[Sign In](#)

Create Account

or use your email for registration

Full Name
Email
Password
Phone Number
Address

[SIGN UP](#)

Fig : 4.1.6 Signup page

4.1.2 Database table

```
import mongoose from "mongoose";
let { Types, model, Schema } = mongoose;
AccessSchema = new Schema({
  accessToken: {
    type: String,
    required: true
  },
  expiry: {
    type: Date,
    required: true,
    default: Date.now,
    expires: 60 * 60 * 1
  },
  userId: {
    type: Types.ObjectId,
    required: true
  }
}, {timestamps: true});
AccessSchema.index({"lastModifiedDate": 1 }, { expireAfterSeconds: 60 * 60 });
export default model('accessToken', AccessSchema);
```

Fig 4.2.1 Access Token

```
import mongoose from "mongoose";
let { Types, model, Schema } = mongoose;
const feedbackSchema = Schema({
  user: {
    type: Schema.Types.ObjectId,
    ref: "User",
  },
  querymessage: {
    type: String,
    require: true
  },
  email: {
    type: String,
    require: true
  },
  date: [
    type: Date,
    default: Date.now
  ]
});
export default model("feedback", feedbackSchema);
```

Fig 4.2.2 Feedback Schema

```
import mongoose from "mongoose";
const Schema = mongoose.Schema;

const verificationtokenSchema = Schema({
  owner: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'user',
    required: true,
  },
  token: {
    type: String,
    required: true,
  },
  createdAt: {
    type: Date,
    expires: 3600,
    default: Date.now()
  }
});

export default mongoose.model('verificationtoken', verificationtokenSchema);
```

Fig 4.2.3 Verification token schema

```
import mongoose from "mongoose";
let { Types, model, Schema } = mongoose;
ProductSchema = new Schema({
  name: {
    required: true, type: String,
  }, rentalprice: {
    type: Number
  }, quantity: {
    type: Number,
    required: true,
  }, costprice: {
    type: String, required: true,
  }, onrent: {
    type: Boolean, default: false, required: true,
  }, description: {
    type: String,
  }, deposit: {
    type: String, required: true,
  }, coupon: {
    type: Array,
  }, categoryy: {
    type: String, require: true
  }, image: {
    type: Array, required: true,
  }, manufacturer: {
    type: String, required: true
  }, createdAt: {
    type: Date,
    default: Date.now,
  }, userId: {
    type: Types.ObjectId, ref: 'user'
  }
}, {timestamps: true});

export default model("product", ProductSchema);
```

Fig 4.2.4 Product Schema


```

import mongoose from "mongoose";
const Schema = mongoose.Schema;
const userSchema = Schema({
  username: {
    type: String, require: true,
  }, email: { value:{
    type: String, require: true,
  }, isVer:{
    type: Boolean, require: true,
  }
  }, password: {
    type: String, require: true,
  }, gender: {
    type: String, require: true,
  }, phonenumber: { value:{
    type: String, require: true,
  }, isVer:{
    type:Boolean, require:true,
  }
  }, address: {
    type: String, require: true,
  }, validation: { documenttype: {
    type: String,
  }, documentnumber: {
    type: String,
    path: {
    type: String,
  }
  }
  }, DOB:{
    type:Date, require:true
  }, scope: {
    type: String, required: true,
  }, cart: { items: [
    {
      productId: {
        type: Schema.Types.ObjectId, ref: 'Product', require:
      }, quantity: { type: Number, required: true }
    }
  ]
  }
});

```

Fig 4.2.5 User Model Schema

```

import mongoose from "mongoose";
const Schema = mongoose.Schema;

const resettokenSchema = Schema({
  owner: {
    type: mongoose.Schema.Types.ObjectId,
    ref:'user',
    required: true,
  },
  token: {
    type: String,
    required: true,
  },
  createdAt:{
    type:Date,
    expires:3600,
    default:Date.now()
  }
});

export default mongoose.model('resettoken', resettokenSchema);

```

Fig 4.2.6 Reset Token Schema

```
import mongoose from "mongoose";
let { Types, model, Schema } = mongoose;
RefreshSchema = new Schema({
  refreshToken: {
    type: String,
    required: true
  },
  expiry: {
    type: Date,
    required: true,
    default: Date.now,
    expires: 60 * 60 * 24 * 7
  },
  userId: {
    type: Types.ObjectId,
    required: true,
    ref: 'user'
  }
}, {timestamps: true});
RefreshSchema.index({"lastModifiedDate": 1},{ expireAfterSeconds: 60 * 60 * 24 * 7 });
export default model('refreshToken', RefreshSchema);
```

Fig 4.2.7 Refresh Token Schema

```
import mongoose from "mongoose";
let { Types, model, Schema } = mongoose;
const querySchema = Schema({
  user: {
    type: Schema.Types.ObjectId,
    ref: "User",
  },
  querymessage:{
    type:String,
    require:true
  },
  email:{
    type:String,
    require: true
  },
  date:{
    type:Date,
    default: Date.now
  }
})
export default model("query",querySchema);
```

Fig 4.2.8 Query Schema

```
import mongoose from "mongoose";
const Schema = mongoose.Schema;
const orderSchema = Schema({
  user: {
    type: Schema.Types.ObjectId, ref: "User",
  }, cart: {
    totalQty: {
      type: Number, default: 0, required: true,
    }, totalCost: {
      type: Number, default: 0, required: true,
    }, items: [
      {
        productId: {
          type: mongoose.Schema.Types.ObjectId, ref: "Product",
        },
        qty: {
          type: Number, default: 0,
        },
        price: {
          type: Number, default: 0,
        },
        title: {
          type: String,
        },
      },
    ],
  }, address: {
    type: String, required: true,
  }, paymentId: {
    type: String, required: true,
  }, createdAt: {
    type: Date, default: Date.now,
  }, Delivered: {
    type: Boolean, default: false,
  },
});
export default mongoose.model("order", orderSchema);
```

Fig 4.2.9 Order Schema

4.2 Testing

4.2.1 White Box Testing

White box testing is an approach that allows testers to inspect and verify the inner workings of a software system its code, infrastructure, and integrations with external systems.

4.2.2 Black Box Testing

The Black Box Test is a test that only considers the external behavior of the system; the internal workings of the software is not taken into account.

4.2.3 Test Cases

Test Case Id	Test Case Name	Module	Expected Output	Actual Output	Pre-condition
Rent_001	Rent_Login	Login Module	Redirect to dashboard, with authenticated user functionalities.	Redirecting to dashboard, with authenticated user functionalities.	Already been registered.
Rent_002	Add product	Admin Module	Product added to database and visible on admin product list page	Properly added product in database and visible on list to.	Product should not already exist.
Rent_003	Checkout	Checkout Module	Order created.	Order created.	Product has to be in cart.

Table No. 4.1

Chapter - 5

Conclusion

With the continued evolution in technology, maintaining the budget is becoming more and more essential now-a-days, where user can freely rent things that they need in a particular period, this is where Rentish come in picture ,which provided a platform for all users to rent daily product like electronics, furniture and many more.

Rentish also solves the problem very precisely and dedicatedly for each and every user .The design of this site is very simple and user-friendly too, which make it more efficiently. Thus we can say that Rentish has a great scope in future for all the users.

Chapter 6

Future Scope

Rentish, as we know, is a rental website, here we have a user module and admin module.

In the user module, the user can view a product, add the product to a cart and rent a product from the cart or directly rent a product. Users can also give feedback on the service and post any queries in that matter. they can also verify their number and email by requesting the OTP and can also change their password by requesting a password generate link.

In the admin module, the admin can add products, delete a product, edit a product and update the product .admin can also see the queries and answer them and can also see feedback.

In the future, we will add a delivery module to this project by which users will be able to track their orders. Here the delivery company will get the order details and will be able to update the delivery status of the order.

References

Book:

- [1] Henry Chan (Author), Raymond Lee (Author), Tharam Dillon (Author) and Elizabeth Chang (Author). "E-Commerce: Fundamentals and Applications".

Research sites:

- [2] Kieraya Furnishing Solutions Pvt. Ltd. "<https://www.furlenco.com>".
- [3] NoBroker.in "<https://furniture.nobroker.in/>".
- [4] RentMacha "<https://www.rentmacha.com>".

Reference Youtube channel:

- [5] Lama Dev (Youtube) "React Node.js E-Commerce App Full Tutorial (REDUX - Stripe - JWT) - MERN Stack Shopping App".
- [6] Thapa Technical (Youtube) "MERN Stack Tutorial 2021".