A screenshot of a computer security system

Description automatically generated

# Abstract

Falcom utilizes a client-server design, mixing client-side and server-side aspects to create a seamless and secure user experience. The front end, developed using React, provides a straightforward and responsive layout that adjusts to multiple devices, ensuring accessibility for all users. The backend, constructed with Node.js and Express, provides extensive server-side capabilities. Advanced security methods, such as HTTPS, JWT for secure authentication, and data encryption, are implemented to offer optimum protection against potential threats. Falcom is designed to provide high performance and scalability, allowing future expansion to meet expanding user demands. The website caters to a varied clientele, offering a complete assortment of automotive products such as bike, automobile, and truck tires, tubes, tire fluid, and accessories. Administrators control the system's correctness and security, merchants update product data, and consumers explore the platform to find solutions tailored to their needs.

# Table of Contents

[Abstract 2](#_Toc188427676)

[Table of Contents 3](#_Toc188427677)

[Table Of Abbreviations 7](#_Toc188427678)

[Introduction 8](#_Toc188427679)

[Software Details 9](#_Toc188427680)

[MONGODB 9](#_Toc188427681)

[EXPRESS.JS 9](#_Toc188427682)

[REACT 10](#_Toc188427683)

[NODE.JS 10](#_Toc188427684)

[SSL Security 10](#_Toc188427685)

[Packages Used 11](#_Toc188427686)

[Strategic Design Principles Of Business Development 12](#_Toc188427687)

[User Experience Design 12](#_Toc188427688)

[Scalable Architecture 12](#_Toc188427689)

[Performance Optimization 12](#_Toc188427690)

[Security Integration 12](#_Toc188427691)

[Security Implemented By Design Principle 12](#_Toc188427692)

[Principle Of Least Privilege 15](#_Toc188427693)

[Secure Data Transmission 15](#_Toc188427694)

[Rate Limiting and Throttling 15](#_Toc188427695)

[Authentication And Session Management 15](#_Toc188427696)

[Input Validation 16](#_Toc188427697)

[Implementation Of Security 16](#_Toc188427698)

[Added Https 16](#_Toc188427699)

[Form Validation 18](#_Toc188427700)

[OTP Verification To Create User Account 19](#_Toc188427701)

[Lock Account After Multiple Failed Logins 20](#_Toc188427702)

[Hashing Password 22](#_Toc188427703)

[Implementation Of Captcha 23](#_Toc188427704)

[Rate Limiting 24](#_Toc188427705)

[Role Based Access Control 26](#_Toc188427706)

[Server-Side 26](#_Toc188427707)

[Client-Side 27](#_Toc188427708)

[UserLog 28](#_Toc188427709)

[Conclusion 28](#_Toc188427710)

[Version Control 29](#_Toc188427711)

[References 29](#_Toc188427712)

[Appendix 30](#_Toc188427713)

Table Of Figure

[Figure 1:HomePage of the System 8](#_Toc188438082)

[Figure 2: Framework used 9](#_Toc188438083)

[Figure 3: Server Started on HTTPS 10](#_Toc188438084)

[Figure 4: Backend Package 11](#_Toc188438085)

[Figure 5: Frontend Package 11](#_Toc188438086)

[Figure 6: Prevent File Upload Vulnerability 13](#_Toc188438087)

[Figure 7: Implementation of File Upload 13](#_Toc188438088)

[Figure 8: Database Validation 14](#_Toc188438089)

[Figure 9: Server-Side logic in search 14](#_Toc188438090)

[Figure 10: Https is enabled for server 15](#_Toc188438091)

[Figure 11: Rate Limit Package 15](#_Toc188438092)

[Figure 12:Jwt Secret Key 15](#_Toc188438093)

[Figure 13:Joi Package 16](#_Toc188438094)

[Figure 14: Server Site Https 16](#_Toc188438095)

[Figure 15: List of Certificate 17](#_Toc188438096)

[Figure 16: Importing package 17](#_Toc188438097)

[Figure 17: Verifying the data transfer with Wireshark 17](#_Toc188438098)

[Figure 18: Server-Side input validation 18](#_Toc188438099)

[Figure 19: Client-Side validation 18](#_Toc188438100)

[Figure 20: Server-Side OTP Validation 19](#_Toc188438101)

[Figure 21: Client-Side OTP in Register 20](#_Toc188438102)

[Figure 22: Server-Side Lock user for multiple attempts 21](#_Toc188438103)

[Figure 23: Client-Side Implementation 22](#_Toc188438104)

[Figure 24:Server-Side Hashed Password 23](#_Toc188438105)

[Figure 25:Hased Password save in database 23](#_Toc188438106)

[Figure 26: Implement Captcha in Client-Side 24](#_Toc188438107)

[Figure 27: Captcha Server-side response 24](#_Toc188438108)

[Figure 28: OTP limitation Server-Side 25](#_Toc188438109)

[Figure 29:Limitation in Frontend 25](#_Toc188438110)

[Figure 30: Guard in route 26](#_Toc188438111)

[Figure 31:Middleware of guard 27](#_Toc188438112)

[Figure 32:Admin Navigation 27](#_Toc188438113)

[Figure 33:User-Log Dashboard 28](#_Toc188438114)

[Figure 34:Client-Side API Endpoint 30](#_Toc188438115)

[Figure 35: Routes of Client-Side 30](#_Toc188438116)

[Figure 36: .Env file 31](#_Toc188438117)

[Figure 37:Nodemailer uses 31](#_Toc188438118)

[Figure 38: UI of View Product by admin 32](#_Toc188438119)

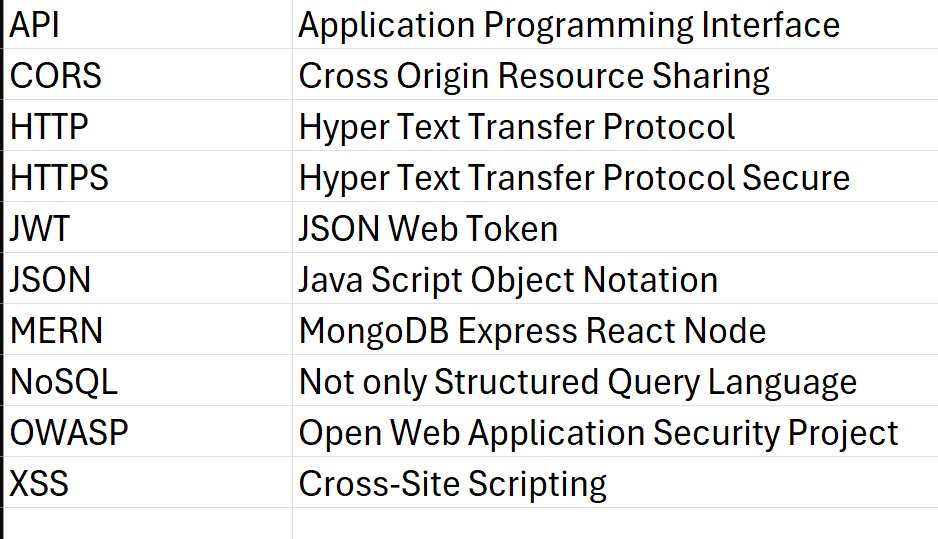
[Figure 39:View Order by admin 32](#_Toc188438120)

[Figure 40:Add to Cart user side 33](#_Toc188438121)

[Figure 41:Other UI 34](#_Toc188438122)

[Figure 42:SSL Certificate 34](#_Toc188438123)

# Table Of Abbreviations



# Introduction

Falcom is an innovative platform dedicated to providing high-quality automotive products and services that cater to drivers from all walks of life. Designed with seamless navigation, robust security measures, and user-friendly search and filtering options, Falcom makes it easy to locate the appropriate automobile solutions for your needs. More than just a store, Falcom delivers thorough insights into the reliability, safety, and performance of its products, helping customers to make informed selections. With options such as premium tires, lubricants, and accessories obtained from industry-leading suppliers, Falcom ensures longevity, safety, and better vehicle performance.

A screenshot of a computer

Description automatically generated

Figure 1:HomePage of the System

The platform targets a diverse audience, including car enthusiasts, professional drivers, businesses, and new drivers, offering tailored expert guidance for every customer. Falcom also prioritizes the security of its users by implementing advanced data encryption, secure payment gateways, and regular security audits to protect customer information and ensure a safe shopping experience. Additionally, Falcom fosters a sense of community by building long-lasting relationships with its customers based on trust, transparency, and shared passion for the road. Falcom is the ideal destination for anyone seeking to upgrade their driving experience, enhance vehicle performance, or stay connected with the latest automotive trends. By choosing Falcom, customers become part of a dedicated network that prioritizes quality, customer satisfaction, and innovation, ensuring that every journey is safe, efficient, and memorable. With a commitment to excellence and a focus on security, Falcom not only meets but exceeds the expectations of modern drivers, providing peace of mind alongside top-tier automotive solutions. Whether you're looking for performance upgrades, maintenance essentials, or the latest in automotive technology, Falcom is your trusted partner on the road ahead.

# Software Details

The MERN stack is a robust and efficient web development stack combining four key technologies: MongoDB, Express.js, React, and Node.js. This stack allows developers to create full-stack JavaScript applications, with both the frontend and backend written in JavaScript [(*MERN Stack Explained*, n.d.).](#mern) Here’s a detailed breakdown of each component of the MERN stack for developing Falcom:

A diagram of a company

Description automatically generated

Figure 2: Framework used

## MONGODB

* **Type**: NoSQL database
* **Role**: Database layer

MongoDB provides a scalable and flexible database solution for Falcom, capable of storing diverse data types such as product catalogs, customer information, and transaction records. Its schema-less nature supports rapid iteration, allowing for easy updates and changes to the data model as the application grows. MongoDB's high-performance read and write operations ensure efficient data handling, making it ideal for a dynamic e-commerce platform like Falcom.

## EXPRESS.JS

* **Type**: Web application framework
* **Role**: Server-side framework

Express.js simplifies backend development for Falcom by providing a robust set of tools for creating RESTful APIs. It enables efficient routing, middleware integration, and seamless handling of HTTP requests and responses. Its unopinionated nature gives developers the flexibility to design the backend architecture tailored to the platform’s specific needs, such as secure user authentication, product management, and order processing.

## REACT

* **Type**: Frontend library
* **Role**: Client-side framework

React is used to build Falcon’s interactive and responsive user interface. It allows the creation of reusable components for features like product displays, search filters, and customer dashboards. With its virtual DOM, React ensures high performance and smooth user interactions, even in data-intensive scenarios. Its component-based architecture promotes modularity and reusability, ensuring a maintainable and scalable frontend.

## NODE.JS

* **Type**: JavaScript runtime
* **Role**: Server-side runtime environment

Node.js serves as the backbone of the backend, enabling real-time data processing and handling multiple concurrent user requests. Its event-driven, non-blocking I/O model ensures lightweight and efficient server operations. For Falcom, Node.js is essential for managing data transactions, integrating payment gateways, and supporting features like real-time stock updates and notifications.

## SSL Security

The process of security on a website comprises of a series of critical processes, including deployment of specialized tools to prevent the website from growing susceptible. First, one should substitute HTTP with HTTPS to guarantee that the information sent between the client and the server is encrypted and not exposed to interception.

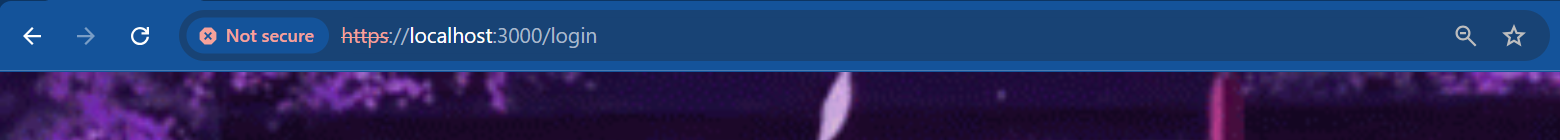


Figure 3: Server Started on HTTPS

Generated SSL certificated was implemented to the system to secure more.

# Packages Used

A screen shot of a computer screen

Description automatically generated

Figure 4: Backend Package

A black screen with orange and white lines

Description automatically generated

Figure 5: Frontend Package

# Strategic Design Principles Of Business Development

## User Experience Design

After careful research of the elements of the planned product, real users’ habits and preferences, a good-looking and easy to navigate interface was built. Convenience is provided with proper layout of the design because it assists the users to conveniently and consecutively search for the best tyre. Appreciation of the employment of graphic elements and the synchronized colors improves the general outlook and makes it engaging for all the users [(Stevens, 2025).](#ui)

## Scalable Architecture

When it came to technological architecture, scalability was vital. It meets growth qualities in so far as future enlargement in terms of user base and functions is concerned. This was done in a procedural manner followed in React, by coding in modules, so each part could be adjusted independently of the others. In addition, this form of architecture significantly accelerates the velocity of development and can is more readily maintainable.

## Performance Optimization

Specific software-related goals were to boost sites’ performance by reducing loading times, as well as organizing data effectively. Features like lazy loading and effective calls to the API were employed. This still is optimized for request processing on the back end, which is necessary for real-time additions such as business data and verification status.

## Security Integration

Since its creation, there has been an emphasis on the topic of security. This means that while creating the system architectural framework, one had to add robust security technologies like HTTPS, JWT, and Bcrypt amongst others. More so, the division of the presentation layer from the business layer aids in the prevention of hazards [(Moteria, 2024).](#security)

# Security Implemented By Design Principle

The security of Falcom system architecture is reinforced by adhering to several OWASP (Open Web Application Security Project) design principles, ensuring robust protection against common vulnerabilities and threats.

## Prevent File Upload Vulnerability

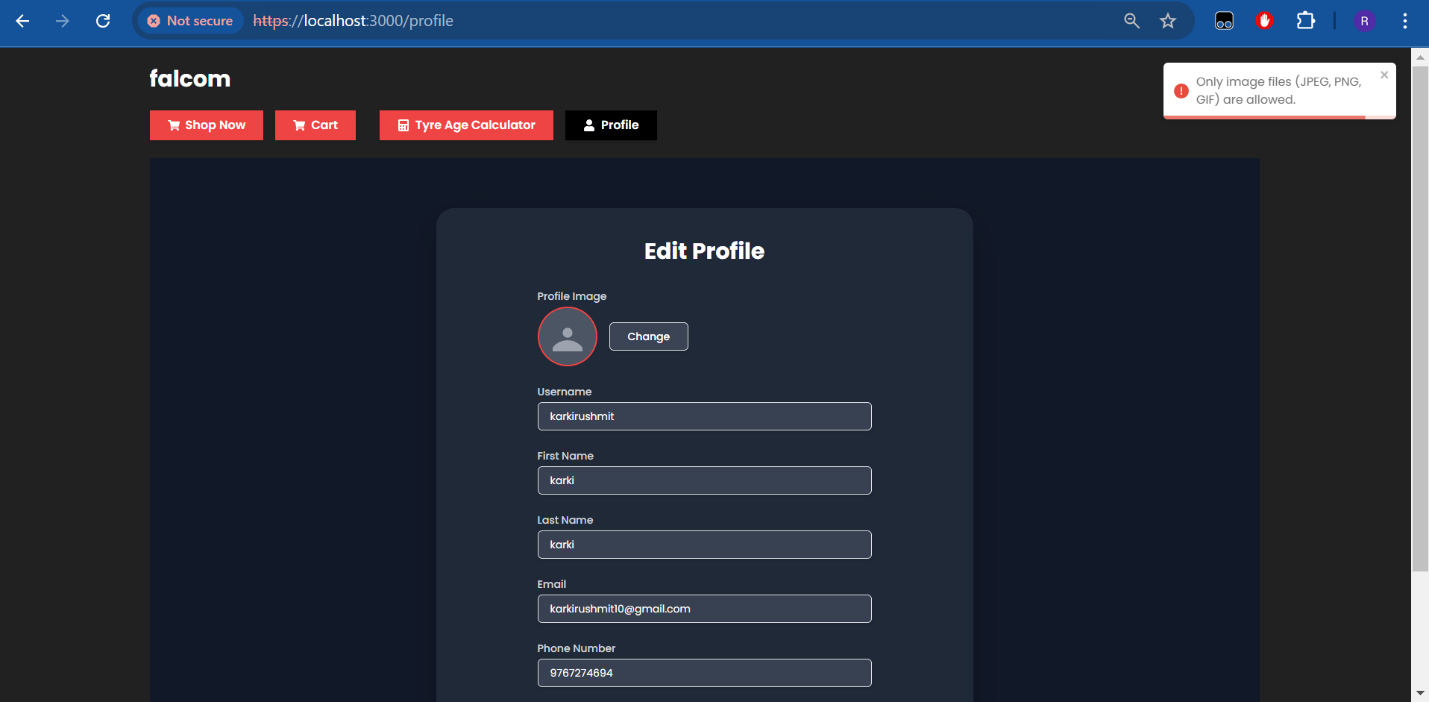


Figure 6: Prevent File Upload Vulnerability

A screen shot of a computer code

Description automatically generated

Figure 7: Implementation of File Upload

# Preventing Command Injection in Search Bar

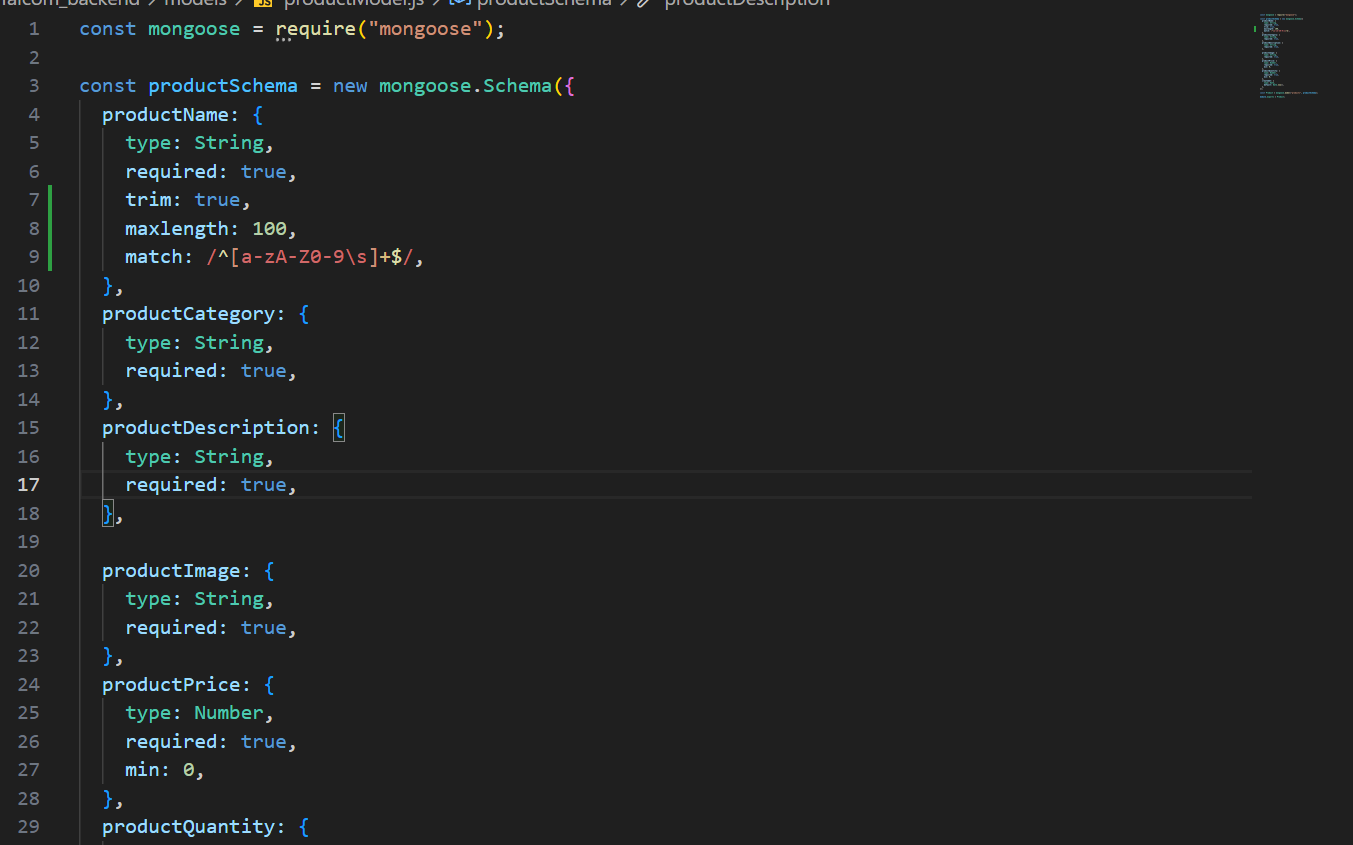


Figure 8: Database Validation

A screenshot of a computer program

Description automatically generated

Figure 9: Server-Side logic in search

## Principle Of Least Privilege

Access constraints are tightly enforced. Users are provided only the rights necessary to perform their roles. Admins have verification and management privileges, firm owners can manage their profiles, and typical users have read-only access to business information.

## Secure Data Transmission

HTTPS is used to encrypt data exchanged between the client and server, guarding against eavesdropping and man-in-the-middle attacks. This ensures that all data, including user credentials and company information, is securely delivered.



Figure 10: Https is enabled for server

## Rate Limiting and Throttling

Express-Rate-Limit is implemented to restrict the number of requests a user may make during a specific timeframe. This helps reduce brute-force assaults and exploitation of the system. If the user makes the multiple requests from the same system, then there should be the capability of limiting the request to the server which eventually stops the server to have a multiple same request from the same system. For this "express rate-limit": "^7.5.0", is employed.

A screen shot of a computer program

Description automatically generated

Figure 11: Rate Limit Package

## Authentication And Session Management

JWT (JSON Web Tokens) is applied for secure authentication and session management. Tokens are securely produced, stored, and verified, guaranteeing that only authenticated users can access protected resources.

A black background with white text

Description automatically generated

Figure 12:Jwt Secret Key

## Input Validation

All user inputs undergo extensive validation and sanitization using tools like Express-Validator and Joi. This protects against typical attacks such as NoSQL injection, cross-site scripting (XSS), and other injection flaws. By utilizing these principles, Falcom offers a secure platform that efficiently protects user data and maintains the integrity of its operations. This focus to security by design helps in developing trust with users and provides a firm foundation for handling company information safely.

A black background with white text and numbers

Description automatically generated

Figure 13:Joi Package

# Implementation Of Security

## Added Https

Transitioning from HTTP to HTTPS in the local environment was a key step, ensuring encrypted data transfer. This change enhanced the security of user transactions and data, simulating real-world secure transfer protocols and showcasing the platform's capability to handle secure transactions and protect user information [(*What Is HTTPS? A Definition and How to Switch to HTTPS? | Fortinet*, n.d.).](#https)

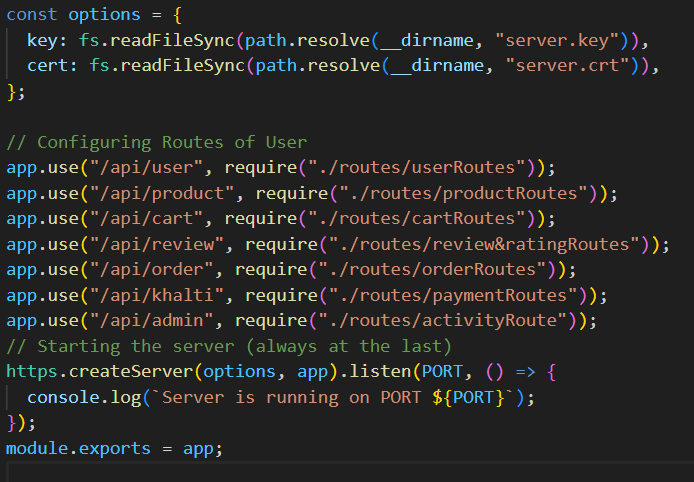


Figure 14: Server Site Https

A screenshot of a computer

Description automatically generated

Figure 15: List of Certificate

A black background with white text

Description automatically generated

Figure 16: Importing package

A screenshot of a computer

Description automatically generated

Figure 17: Verifying the data transfer with Wireshark

## Form Validation

Every input undergoes validation to minimize human input errors. This guarantees that all data submitted by users is correct and consistent, lowering the likelihood of errors and boosting the overall reliability and security of the platform. To further strengthen security, data sanitization is conducted on all inputs to avoid harmful data and ensure clean, safe data handling.

A computer screen with colorful text

Description automatically generated

Figure 18: Server-Side input validation

A screenshot of a login form

Description automatically generated

Figure 19: Client-Side validation

## OTP Verification To Create User Account

Additionally, user accounts are generated only after verification. The verification procedure occurs immediately after the user data is saved in an unverified state, ensuring that only verified users may use the site. After saving the user info the OTP is sent to the email address and if the OTP is verified then the user is verified and can only login to the system.

A screen shot of a computer program

Description automatically generated

Figure 20: Server-Side OTP Validation

A screen shot of a computer program

Description automatically generated

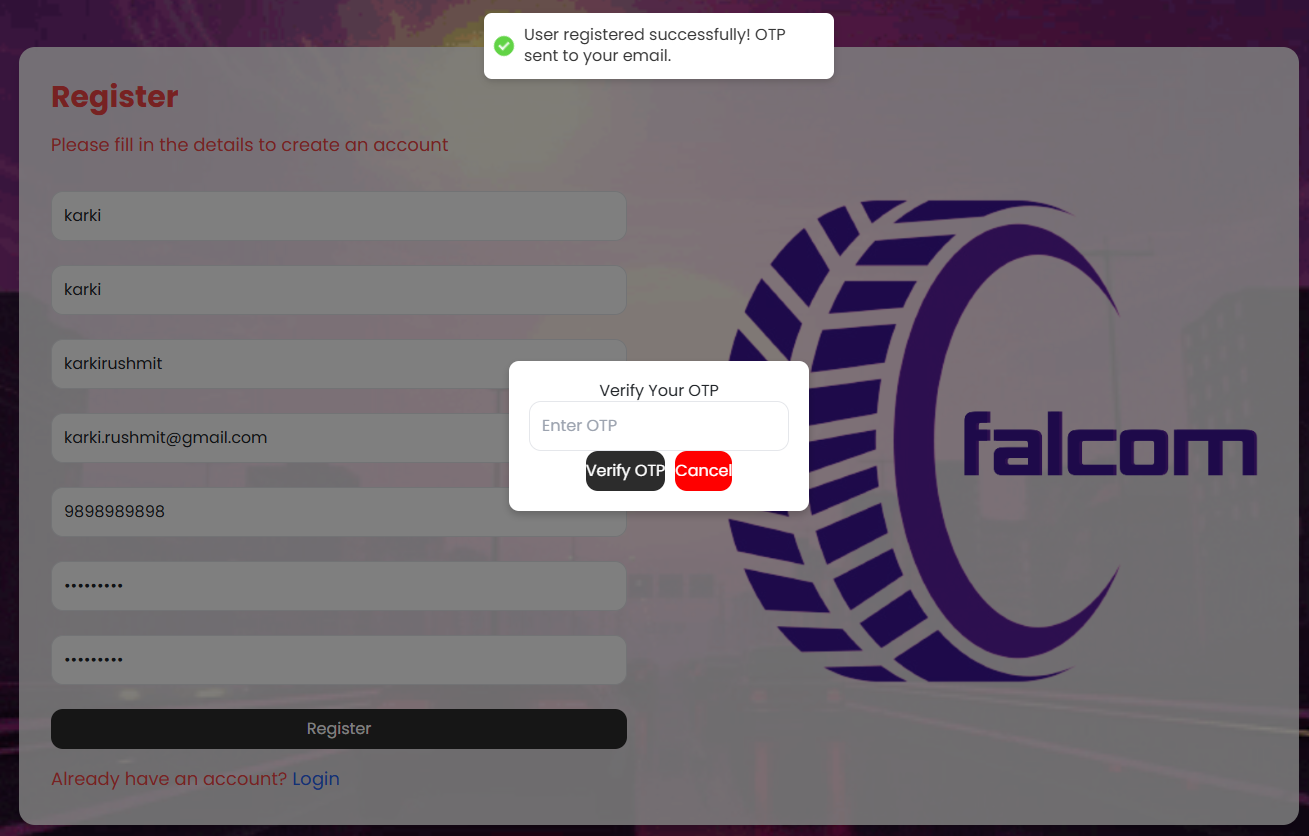


Figure 21: Client-Side OTP in Register

## Lock Account After Multiple Failed Logins

To increase security and coding safeguards against bad givers or guessing, it is recommended to freeze one’s account if repeated failed attempts are made. This strategy entails determination of a lockout policy that is a measure of the number of tries a person can make before their account is locked in a particular amount of time. The standard would be to set the authorized number of attempts to be five and the time lock would be set to half an hour. To this aim, further fields are created in the user model to keep track of the failed login attempts and the lockout situation. In the event of the login process, if the number of failures that arise from improper entries is beyond the authorized limit, the account is closed for the given time. This is done by comparing and changing the login credentials and lockout property in the user database. Through this, the system disallows any unauthorized attempts to log in through different erroneous user ID and password by locking the account to protect the user accounts and the overall application.

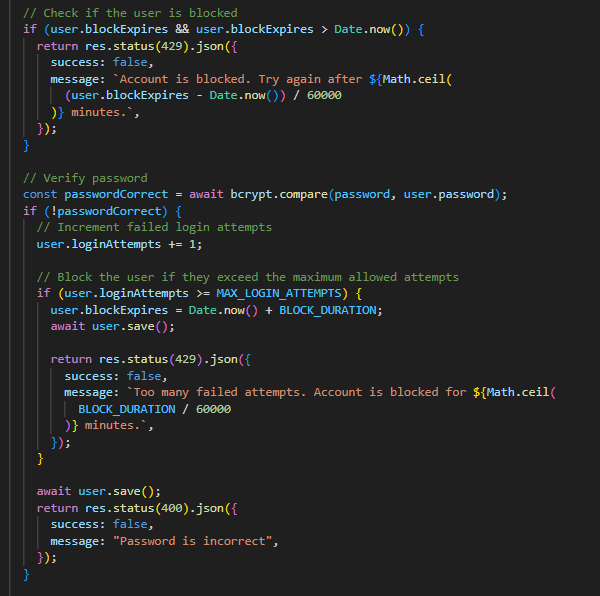


Figure 22: Server-Side Lock user for multiple attempts

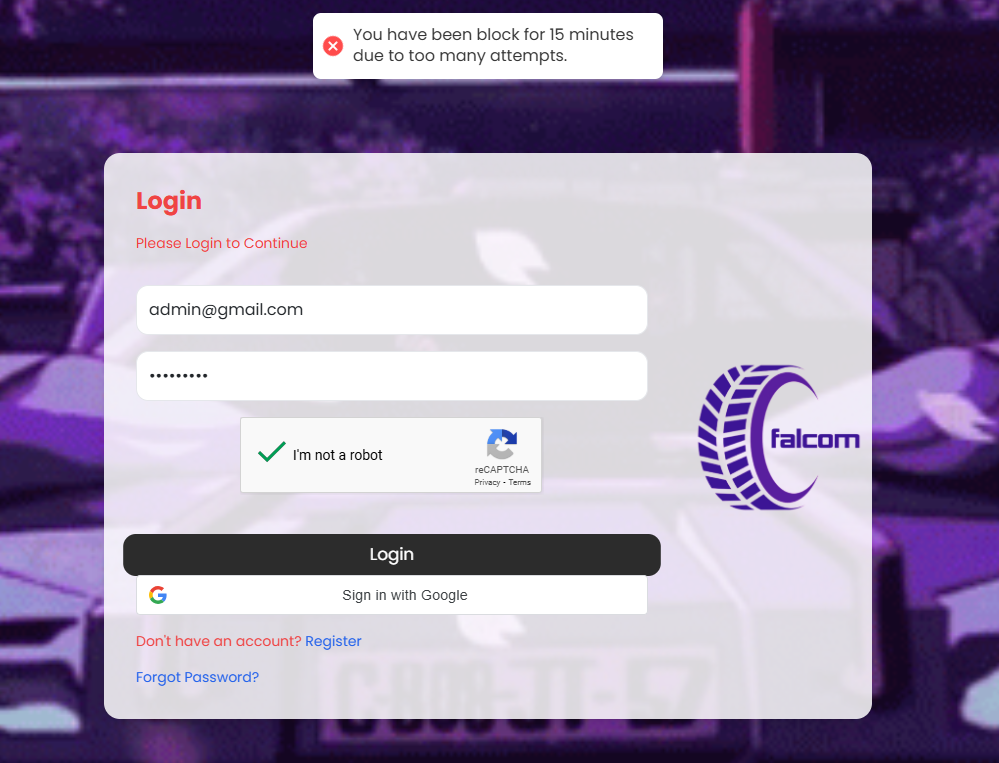


Figure 23: Client-Side Implementation

## Hashing Password

Password hashing is something that lays in the basics of security for keeping the users’ login details in a database. Instead of maintaining the passwords as plain-text it is encrypted using a cryptographic hash function so in case the database is penetrated the passwords cannot be simply assumed. They produce a string of characters of a specified size that seems random and does not contain any identifiable information with the original password. Also, another salt of say random string is generally applied to the passwords to increase security since, for the same value of password, a different value of hash string is produced. This keeps the hackers from using tables they had pre-computed known as rainbow tables and reverse computer for passwords. By maintaining only password hashes as well as the salt added to them, the database is made more secure as even if the hashed data is somehow exposed the actual passwords are a lot tougher for the attackers to access. This strategy considerably increases the general security of the application and keeps the users’ data from being hacked [(*How to Hash Passwords: One-Way Road to Enhanced Security*, n.d.).](#hashed)

A screen shot of a computer code

Description automatically generated

A computer screen with text

Description automatically generated

Figure 24:Server-Side Hashed Password

A screenshot of a computer

Description automatically generated

Figure 25:Hased Password save in database

## Implementation Of Captcha

The integration of CAPTCHA before login in the **Falcom App** enhances the security of the platform, ensuring a safe and reliable user experience. This feature protects against malicious activities, such as brute-force attacks, credential stuffing, and automated bots, by requiring users to complete a CAPTCHA challenge before submitting their login credentials.CAPTCHA stands for Completely Automated Public Turing test to tell Computers and Humans Apart [(Balankdharan, 2023).](#captcha)

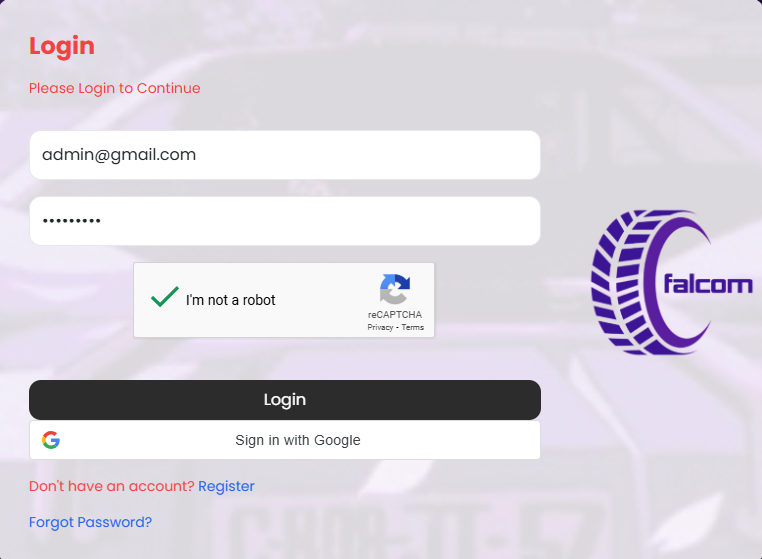


Figure 26: Implement Captcha in Client-Side

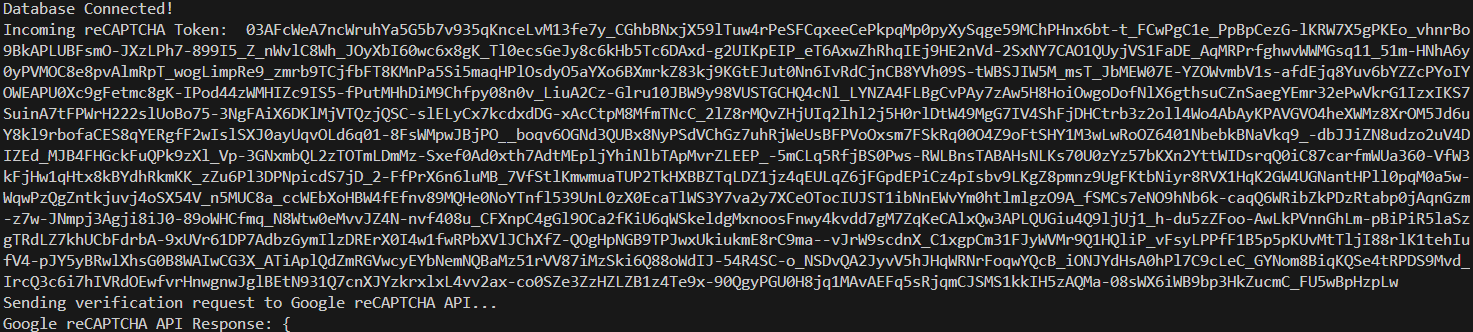


Figure 27: Captcha Server-side response

## Rate Limiting

The common security feature known as rate limiting prevents a user from making too many requests within a specific timeframe to a web application. This is good as a measure against abusive access and DoS attacks where the attacker churns through requests much faster than the server can handle, and the result is a slowing down of the server or complete unavailability to regular users Because of this, they also limit the rate of requests so that it also reduces brute force attacks; a method in which attackers use several combinations of username and password to access a system a program. This precaution prevents a given user from utilizing most of the application’s resources, thus preventing inefficiency and making the system more stable. Also, rate limiting aids in moderating traffic flow on the server; if traffic is allowed to rush in within a short time, this puts a lot of pressure on the server, and therefore affects everyone that must use the server side.

A screen shot of a computer program

Description automatically generated

Figure 28: OTP limitation Server-Side

A white rectangular sign with black text

Description automatically generatedA screenshot of a login page

Description automatically generated

Figure 29:Limitation in Frontend

## Role Based Access Control

RBAC can be defined as a security model that regulates system’s access to users with roles in the organization. Every role relates to privileges, which determines which actions the end user authorized to certain role can make [(*What Is Role-Based Access Control (RBAC)? Examples, Benefits, and More*, n.d.).](#role) The approach guarantees that user has only the specific access level required to perform his job profile and thus discourages any unnamed activity [(Team, n.d.).](#role2)

### Server-Side

A screen shot of a computer program

Description automatically generated

Figure 30: Guard in route

A screen shot of a computer screen

Description automatically generated

Figure 31:Middleware of guard

### Client-Side

A screen shot of a computer program

Description automatically generated

Figure 32:Admin Navigation

## UserLog

Admin can access all the user log from admin dashboard such as user ip, username, user email and even the time when user login or user failed to login and if the user has attempts to login for many times, then user will be blocked by admin.

A screenshot of a computer

Description automatically generated

Figure 33:User-Log Dashboard

# Conclusion

In conclusion, Falcom is constructed with a comprehensive security framework that complies with industry standards and best practices, guaranteeing substantial protection against potential threats and weaknesses. The system integrates many security protocols, including the prevention of unauthorized multiple login attempts, user activity tracking, and the implementation of XSS mitigation, alongside other sophisticated techniques. These safeguards together protect the site, its users, and their information.  
Moreover, Falcom's security framework adheres to secure design principles, encompassing least privilege access, encrypted data transport, and robust authentication procedures. Consistent input validation, rate limitation, and periodic security evaluations enhance the platform, guaranteeing a secure and dependable environment for users.

By emphasizing security, Falcom ensures a reliable and uninterrupted experience for customers in search of premium bike, car, and truck tires, tubes, tire fluid, and additional products. Falcom's commitment to security and reliability positions it as a dependable platform that addresses the varied requirements of its users, while instilling confidence in its sustainable and customer-centric solutions.

# Version Control

GitHub Link: <https://github.com/Rushmitkarki/220046_security_cw2_final>

YouTube Link:

# References

*MERN Stack explained*. (n.d.). MongoDB. <https://www.mongodb.com/resources/languages/mern-stack>

Stevens, E. (2025, January 8). What is user experience (UX) design? Everything you need to know. *CareerFoundry*. <https://careerfoundry.com/en/blog/ux-design/what-is-user-experience-ux-design-everything-you-need-to-know-to-get-started/>

Moteria, D. (2024, September 24). *MERN Stack Security Best Practices: How to keep your Applications safe*. Elluminati Inc. <https://www.elluminatiinc.com/mern-stack-security/>

*What is HTTPS? A Definition and How to Switch to HTTPS? | Fortinet*. (n.d.). Fortinet. <https://www.fortinet.com/resources/cyberglossary/what-is-https>

*How to Hash Passwords: One-Way Road to Enhanced Security*. (n.d.). Auth0 - Blog. <https://auth0.com/blog/hashing-passwords-one-way-road-to-security/>

Balankdharan. (2023, October 29). Adding ReCAPTCHA in MERN application - Balankdharan - Medium. *Medium*. <https://medium.com/@balankdharan/adding-recaptcha-in-mern-application-10391bf10f6d>

*What is Role-Based Access Control (RBAC)? Examples, Benefits, and More*. (n.d.). <https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more>

Team, M. D. (n.d.). *Role-Based access control in Self-Managed deployments*. MongoDB Manual v8.0. <https://www.mongodb.com/docs/manual/core/authorization/>

# Appendix

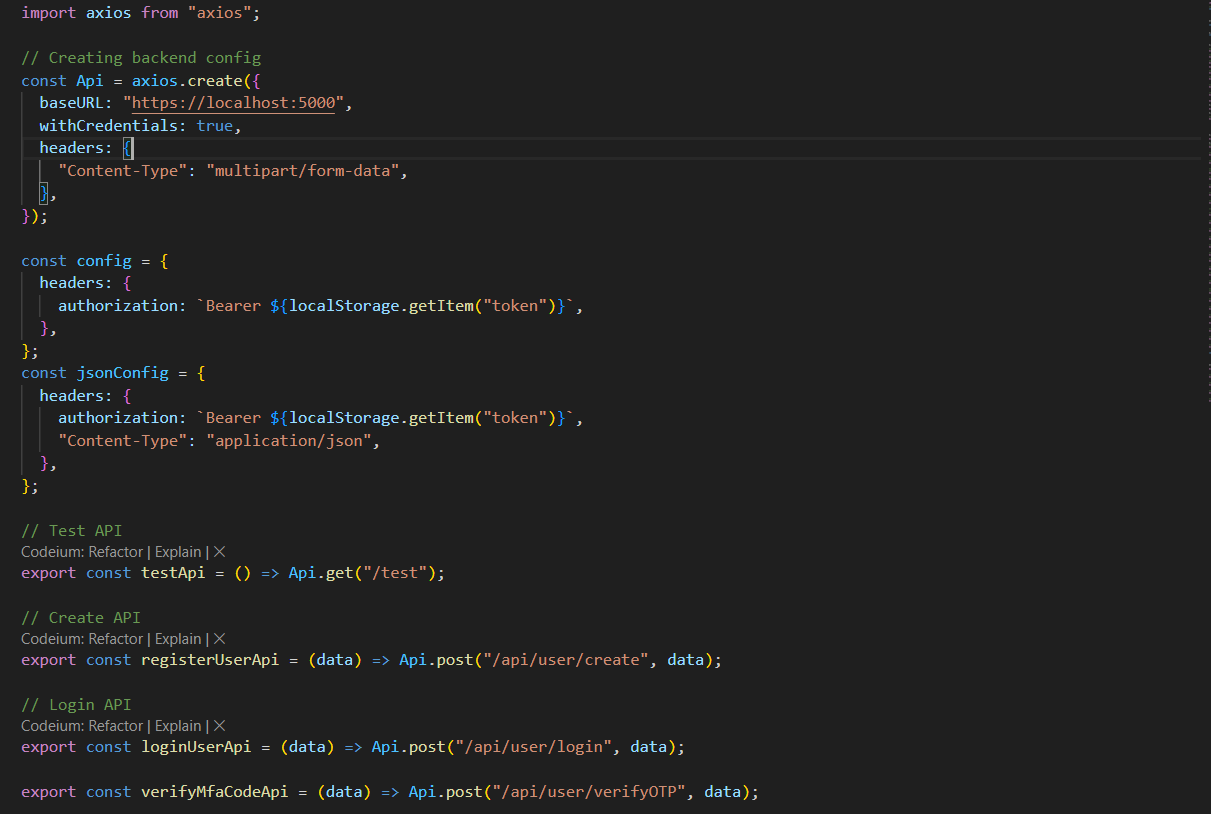


Figure 34:Client-Side API Endpoint

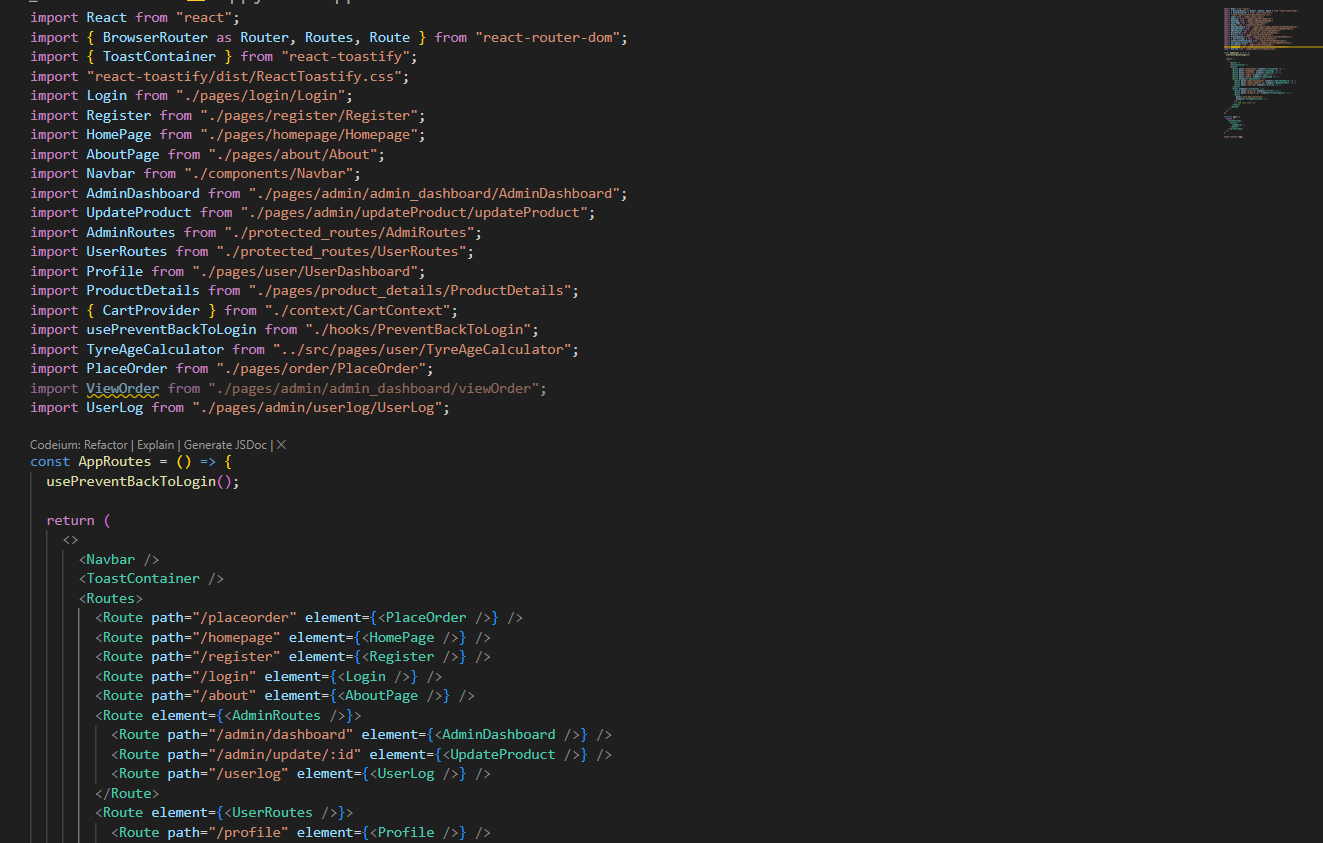


Figure 35: Routes of Client-Side

A screen shot of a computer

Description automatically generated

Figure 36: .Env file

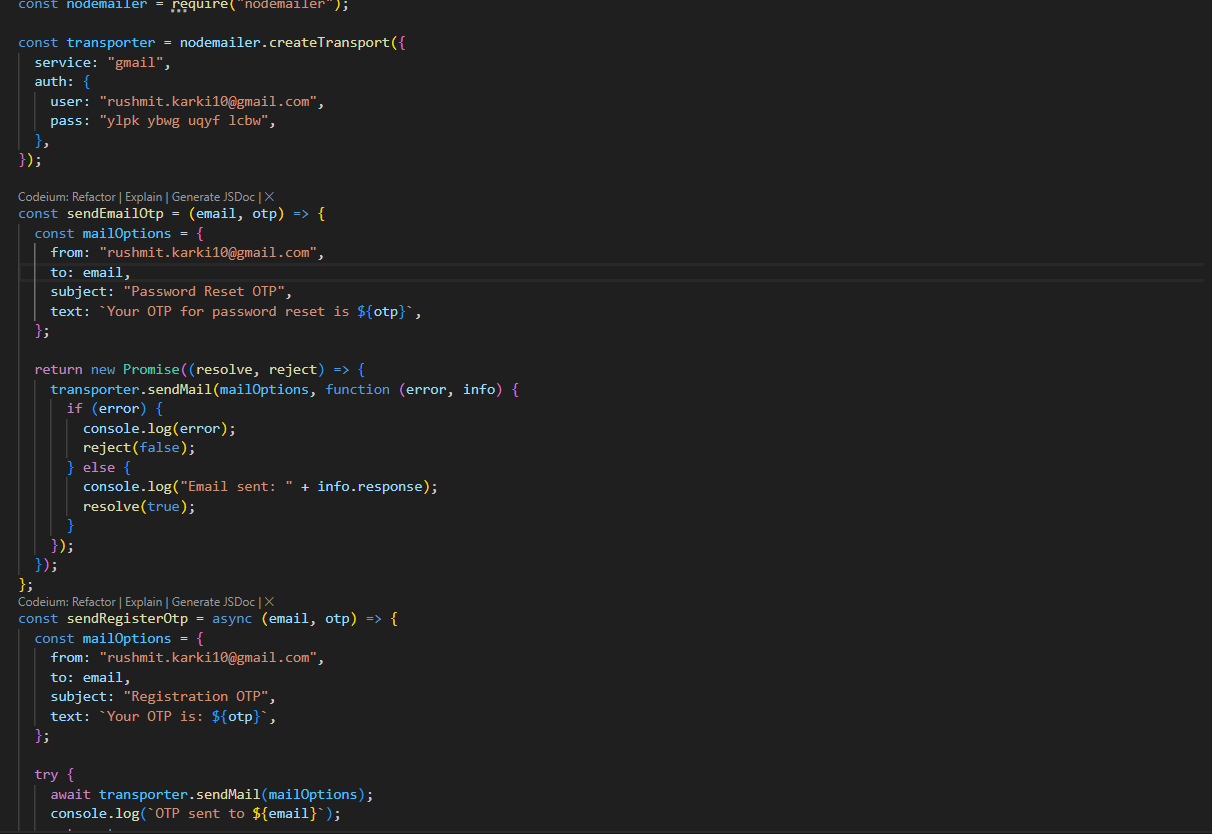


Figure 37:Nodemailer uses

A screenshot of a computer

Description automatically generated

Figure 38: UI of View Product by admin

A screenshot of a computer

Description automatically generated

Figure 39:View Order by admin

A screenshot of a computer

Description automatically generated

Figure 40:Add to Cart user side

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Figure 41:Other UI

A screenshot of a computer

Description automatically generated

Figure 42:SSL Certificate