Faculty of Engineering & Information Technology

Computer Engineering Department

Software Graduate Project

MatjarCom

Students:

Abdullah Ghassan Sholi

Omar Farouq Quzmar

Supervisors:

Dr.Samer Arandi

Dr.Muhannad Al-Jabi

Acknowledgment

I would like to extend my heartfelt gratitude to everyone who supported and contributed to the development of the MatjarCom application. Special thanks to my academic advisor for their invaluable guidance and feedback throughout this project. I am also deeply grateful to my family and friends for their unwavering support and encouragement. Additionally, I appreciate the open-source community and the developers behind the libraries and tools that made this project possible. Lastly, I thank all the beta testers whose insightful feedback helped refine and improve the application.

Abstract

MatjarCom is a comprehensive multi-vendor e-commerce platform designed to empower merchants by providing a digital marketplace to create and manage their online stores. The application allows merchants to design and customize their stores, manage products and categories, communicate with customers, and track their sales performance. Customers benefit from a unified shopping experience across multiple stores with features such as product browsing, purchasing, and reviewing. The app is built using Flutter for the front end and Node.js with MongoDB for the backend, ensuring a responsive and scalable architecture. MatjarCom also includes robust security measures, real-time communication capabilities, and support for both English and Arabic languages. The application is hosted on Render for backend services, ensuring reliable and scalable deployment.

Contents

List of Figures

tocloft 0.5ex

1 Introduction

In the digital age, e-commerce platforms have become essential for businesses to reach a broader audience and enhance their sales potential. MatjarCom aims to fill the gap in the market for a robust, user-friendly multi-vendor platform that facilitates merchants to create, customize, and manage their online stores efficiently. This application is designed to cater to both merchants and customers, providing a seamless experience for store management and product purchasing. Leveraging modern technologies and best practices, MatjarCom offers a comprehensive solution for businesses to thrive in the competitive online marketplace.

2 Constraints & Earlier Work

2.1 Constraints

2.1.1 Technological Constraints

Compatibility: Ensuring the application is compatible with various devices and operating systems.
Performance: Maintaining optimal performance and responsiveness for both the front end and back end.
Security: Implementing robust security measures to protect user data and transactions.

2.1.2 Resource Constraints

Time: Limited time available for development

2.1.3 Operational Constraints

Scalability: Designing the system to handle increasing numbers of users and transactions.
Localization: Supporting multiple languages and regional settings to cater to a diverse user base.

2.1.4 User Experience Constraints

Usability: Ensuring the application is intuitive and easy to use for both merchants and customers.
Accessibility: Making the app accessible to users with varying levels of technical expertise.

2.2 Constraints

2.2.1 Earlier Work

The concept of multi-vendor marketplaces is not new, and several established platforms like Amazon, eBay, and Etsy have paved the way in this domain. These platforms provide merchants with an opportunity to reach a global audience, offering extensive features for product listing, inventory management, and customer engagement. However, many of these platforms have limitations in terms of customization and flexibility for individual merchants.

Prior to MatjarCom, similar efforts have been made to develop multi-vendor applications, such as Shopify, WooCommerce, and Magento. These platforms offer varying degrees of functionality and customization but often come with a steep learning curve or cost barriers for small to medium-sized businesses.

MatjarCom differentiates itself by providing a more accessible, customizable, and localized solution, especially for markets that require bilingual support. It leverages modern technologies and a modular architecture to offer a user-friendly and flexible platform for both merchants and customers.

3 Literature Review

3.1 E-commerce Platforms and Marketplaces

Research on e-commerce platforms highlights the importance of user experience, customization, and scalability. Platforms like Shopify and Magento provide extensive customization options but may be challenging for less technical users. Studies suggest that platforms offering a balance between ease of use and customization tend to be more successful (Smith, 2020).

3.2 Multi-Vendor Systems

Multi-vendor systems enable multiple sellers to operate within a single marketplace, providing benefits like reduced operational costs and expanded reach. However, these systems also present challenges in terms of managing diverse product catalogs and ensuring fair distribution of resources (Johnson & Wang, 2019).

3.3 Security in E-commerce

Security is a critical concern for e-commerce platforms, with risks ranging from data breaches to fraudulent transactions. Implementing secure authentication methods, data encryption, and robust transaction monitoring are essential strategies for mitigating these risks (Doe, 2021).

3.4 Mobile Commerce Trends

With the rise of smartphones, mobile commerce has become increasingly prevalent. Research indicates that user-friendly mobile interfaces and seamless integration of payment systems are crucial for the success of mobile e-commerce applications (Lee, 2022).

3.5 Customer Engagement and Retention

Effective customer engagement strategies, such as personalized marketing and responsive customer service, significantly impact customer retention and satisfaction. Multi-vendor platforms must provide tools for merchants to interact with their customers and manage their relationships effectively (Patel, 2020).

4 Methodology

4.1 Technology Stack Selection

4.1.1 Frontend (Flutter)

Utilize Flutter SDK for cross-platform mobile app development.
Ensure compatibility with both iOS and Android platforms.

4.1.2 Backend (Node.js with MongoDB)

Node.js for server-side logic and API development with Layered Architecture.

MongoDB for NoSQL database to store dynamic data (e.g., merchant details, store information).

Use packages like bcrypt for password hashing, nodemailer for email communication, and express for RESTful API development.

4.1.3 Security Implementation

Authentication: Implement JWT (Json Web Token) for secure authentication of both merchants and customers.
Use bcrypt for password hashing to protect user credentials.
Authorization: Implement separate authorization logic in each endpoint to validate user type (customer, merchant, admin) based on the JWT payload.˙
Ensure API security with CORS, HTTP headers, and rate-limiting (express-rate-limit) to prevent abuse and protect against common vulnerabilities.

4.1.4 Database Design

MongoDB: Design collections for customers, merchants, admin, products, store information, etc.
Optimize schema for fast data retrieval and scalability.
Maintain data integrity with appropriate indexes and validations.

4.1.5 Frontend Development

Flutter: Design responsive UI components using Flutter widgets.
Implement state management (e.g., setState({})) for managing app state and data flow.
Utilize flutter translate for supporting multiple languages (Arabic and English).

4.1.6 Backend Development

Node.js: Develop RESTful APIs to handle CRUD operations for customers, merchants, admin functionalities.
Integrate third-party APIs (e.g., Lahza for payments) securely.
Implement server-side validation and error handling for robustness.

4.1.7 Admin Dashboard Implementation

Design admin dashboard for managing merchants, customer support, analytics, and content management.
Implement CRUD operations specific to admin functionalities.
Ensure dashboard is secure with authentication for admin access.

4.1.8 Messaging System Integration

Integrate messaging systems for customer-merchant and merchant-admin communications.

4.1.9 Help & Support

implement a support system where customers can contact admin via email for complaints and queries.

Use Node.js with nodemailer for secure email communication.
Provide a user-friendly interface for customers to initiate and track their support requests.

5 Results & Discussion

The results of the study are presented in this section.

5.1 Technologies Used

5.1.1 Mobile Frontend (Flutter)

Flutter is a versatile open-source UI software development kit created by Google, renowned for its capability to build natively compiled applications for mobile, web, and desktop from a single codebase. Launched in 2017, Flutter utilizes the Dart programming language and offers a rich set of pre-designed widgets and tools that expedite the development process while ensuring consistency and high performance across platforms. Its reactive framework enables rapid prototyping and seamless deployment, making it a preferred choice for developers aiming to deliver visually appealing and responsive applications across diverse operating systems. With a growing community and continuous updates enhancing its features, Flutter continues to redefine cross-platform development by empowering developers to create stunning user experiences efficiently

5.1.2 Web Frontend (Flutter Web)

Flutter Web extends the capabilities of the Flutter framework to enable developers to build full-featured, scalable web applications using the same codebase as their mobile apps. Introduced as a technical preview in 2019 and officially released in 2020, Flutter Web leverages the same reactive framework and widget system that Flutter is known for, allowing developers to create responsive, visually rich web applications that can adapt seamlessly across different screen sizes and devices. By compiling Dart code to optimized JavaScript, Flutter Web enables high performance and smooth animations, while also supporting features like hot reload for rapid iteration during development. With its growing ecosystem of plugins and libraries, Flutter Web empowers developers to build modern, engaging web experiences efficiently, bridging the gap between mobile and web platforms.

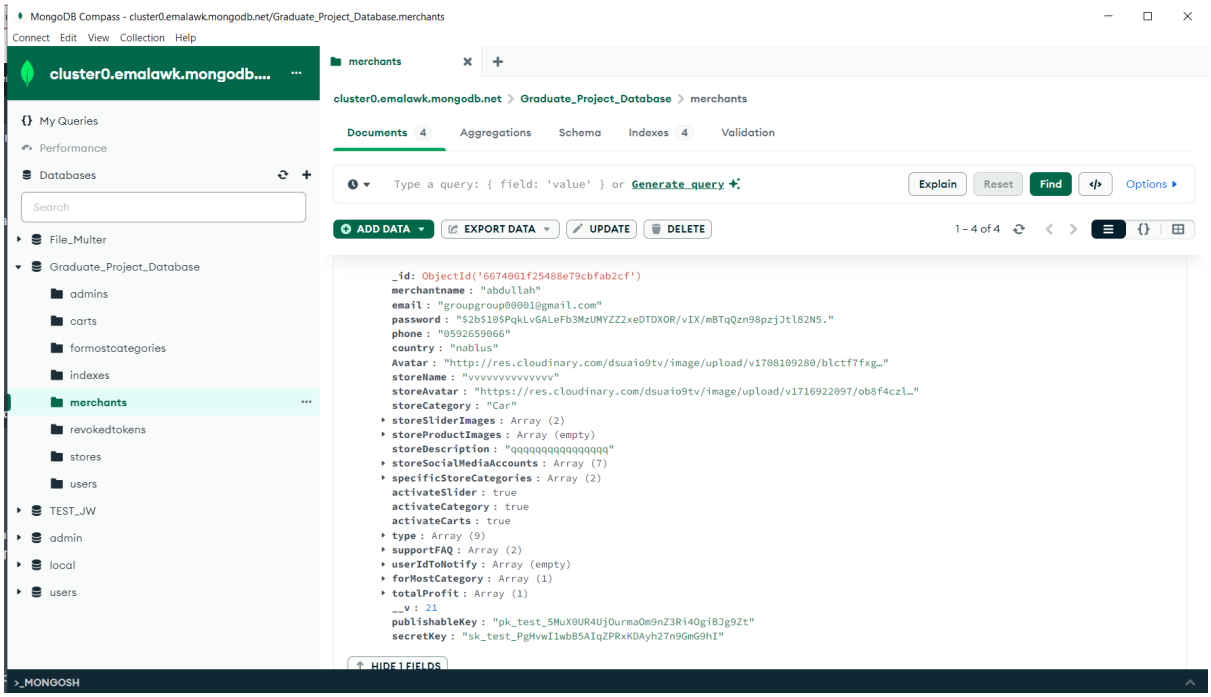
5.1.3 Backend (NodeJS)

Node.js is a powerful runtime environment built on Chrome's V8 JavaScript engine, designed for scalable and efficient server-side applications. It allows developers to use JavaScript both for frontend and backend development, unifying the programming language across the entire stack. Express.js, often referred to simply as Express, is a minimalist and flexible web application framework for Node.js. It provides a robust set of features for building web and mobile applications, including middleware support for handling requests and responses, routing mechanisms to define application endpoints, and a template engine for rendering dynamic HTML content. Express simplifies the process of building APIs and web servers, offering a lightweight and fast solution that is highly extensible through its ecosystem of middleware and third-party modules. Together, Node.js and Express form a potent combination for developing scalable and performant server-side applications, making them popular choices among developers for building modern web applications and APIs.

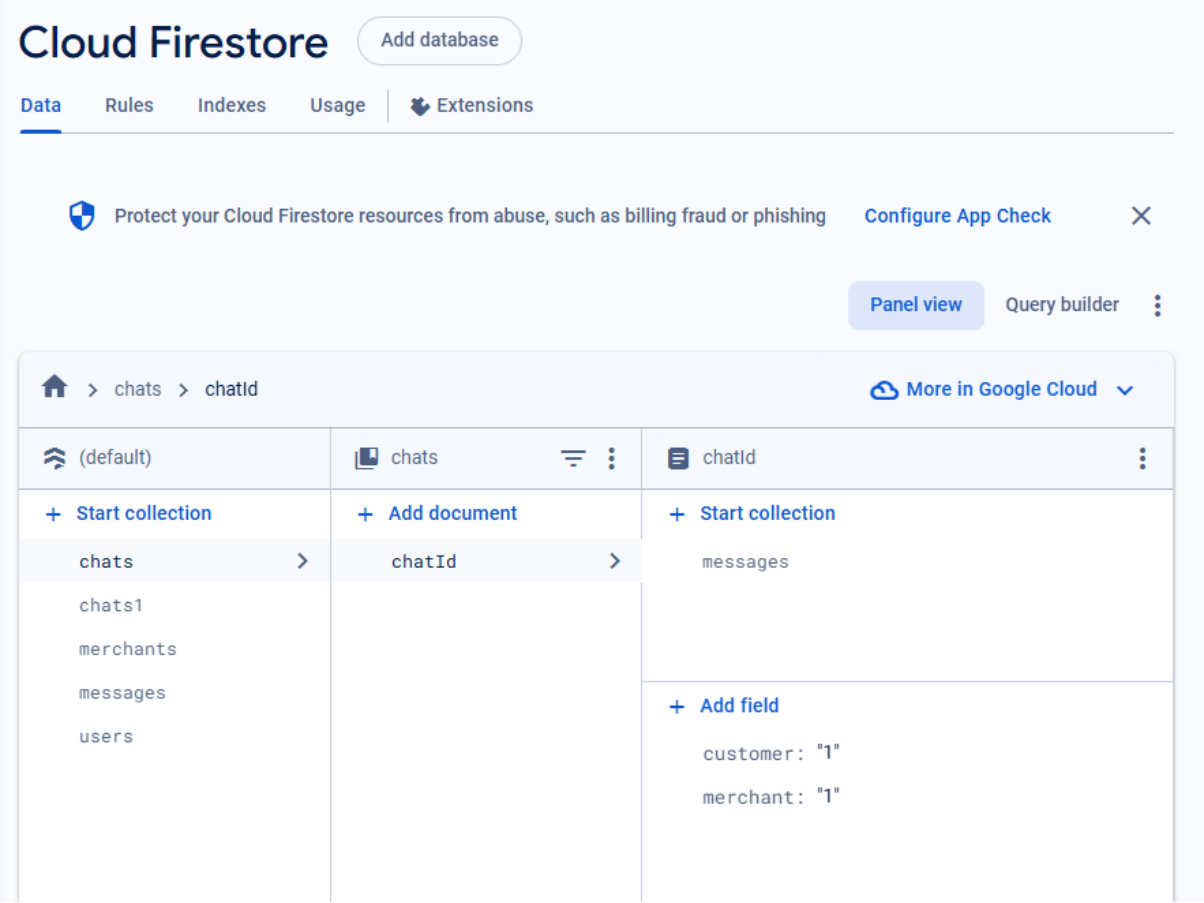
5.1.4 Database (NoSQL MongoDB & Firebase)

MongoDB is a NoSQL database that stores data in a flexible, JSON-like format, making it particularly suitable for handling unstructured or semi-structured data. It is known for its scalability, high performance, and ease of use, especially in applications with large volumes of data or where the schema may evolve over time. The Mongoose package, on the other hand, is an ODM (Object Data Modeling) library for MongoDB and Node.js, providing a straightforward way to model application data and interact with MongoDB databases using JavaScript objects. Mongoose simplifies CRUD operations, schema validation, and relationships between data, offering a schema-based solution on top of MongoDB's schema-less nature.

Our database schemas:

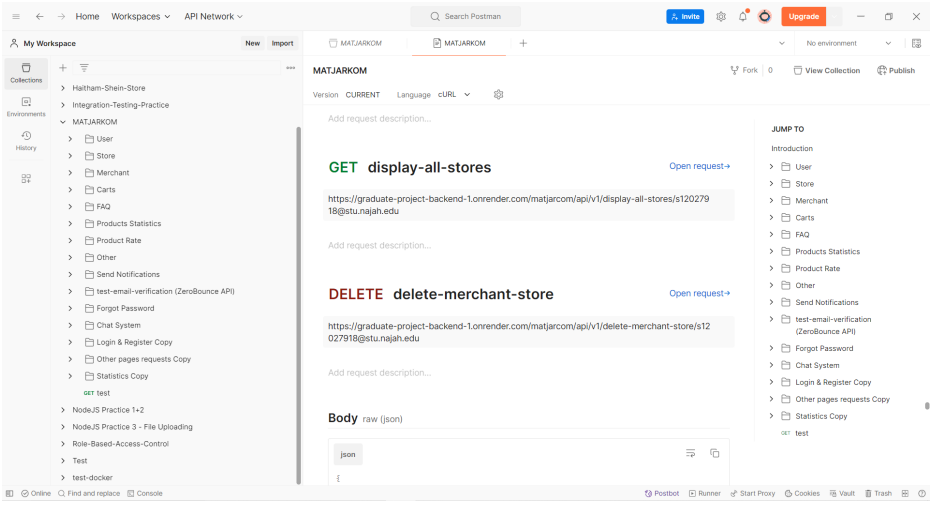


Our firebase for chatting system:



5.1.5 API Testing & Documentation (Postman)

REST APIs Are introduced in the system, using different CRUD operations and methods of HTTP LIKE (PUT,POST,Patch ...)
Postman



5.1.6 Web Application Features:

Authentication MatjarCom prioritizes user security through a dual-layered approach. Utilizing JSON Web Tokens (JWT), the platform generates secure tokens during user authentication, ensuring verified access. User passwords are stored encrypted in the database, enhancing data protection

Authorization Implement separate authorization logic in each endpoint to validate user type (customer, merchant, admin) based on the JWT payload.˙
Ensure API security with CORS, HTTP headers, and rate-limiting (express-rate-limit) to prevent abuse and protect against common vulnerabilities.

6 User-Friendly Interface

6.1 Splash Screen

Figure 1: splash screen is an introductory screen that appears when an application is launched.

6.2 Onboarding Page

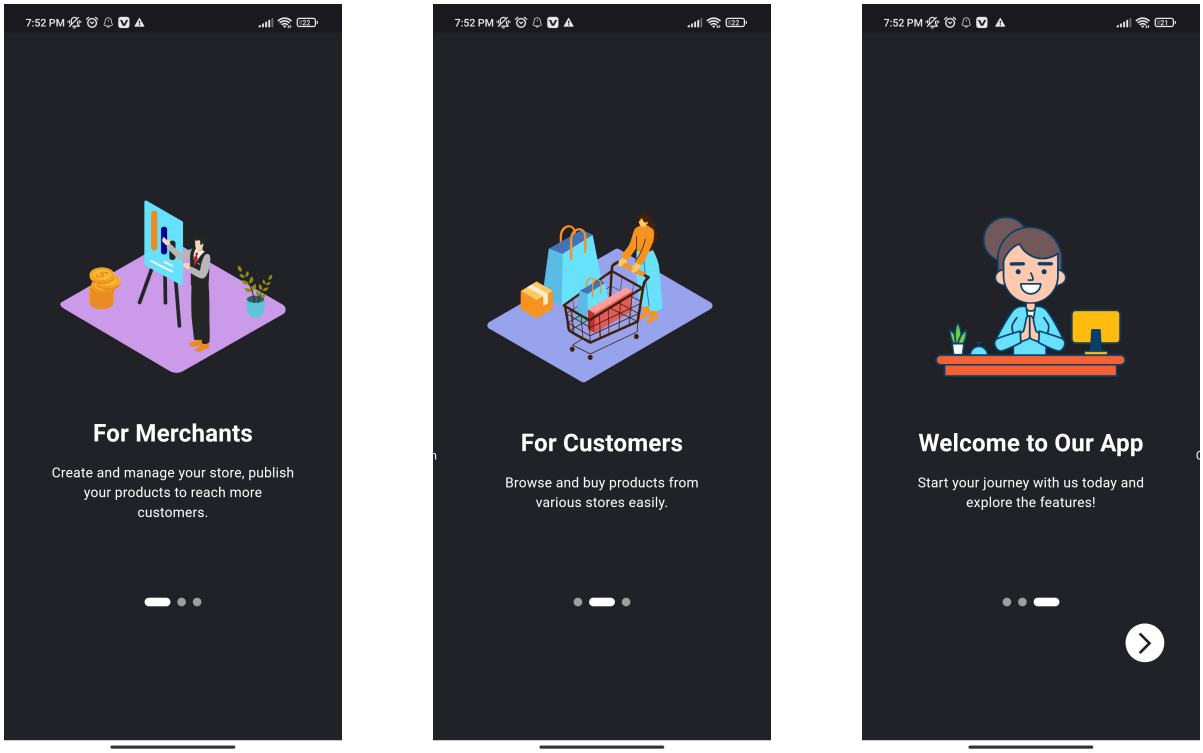


Figure 2: onboarding page (or series of pages) is designed to introduce new users to an app or service. The goal is to guide users through initial setup and demonstrate core features and benefits.

6.3 Login Page

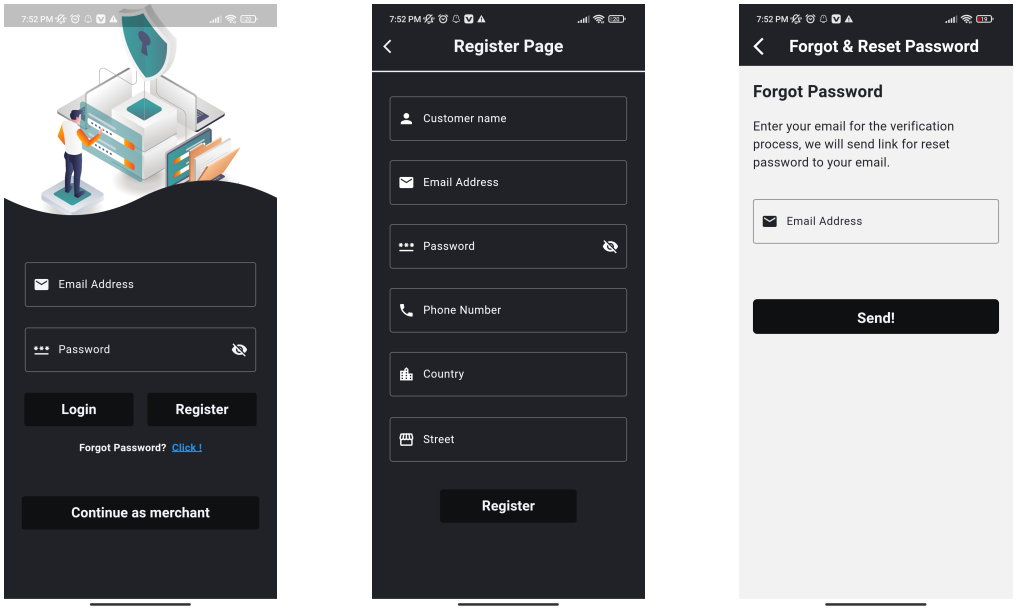


Figure 3: Login, register, and Forgot & Reset Password for Customer

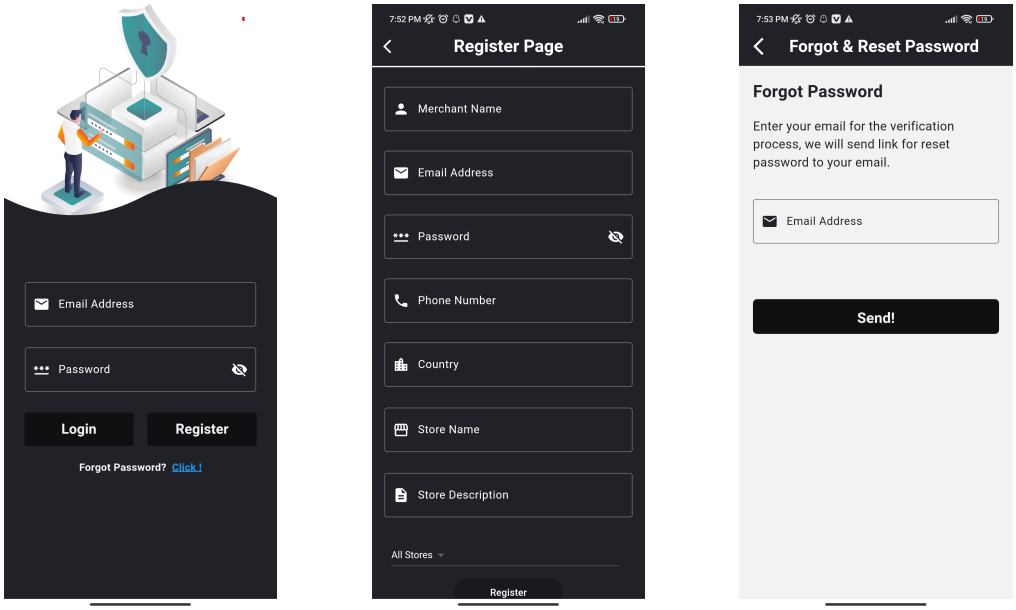


Figure 4: Login, register, and Forgot & Reset Password for Merchant

6.4 Merchant Side

6.4.1 Login Page

* Robust login page, by put fields validators, display message in dialog when merchant insert invalid email or password, display another message in dialog when merchant try to login with large number with wrong data this protect our system from dDOS attack which could hult server.

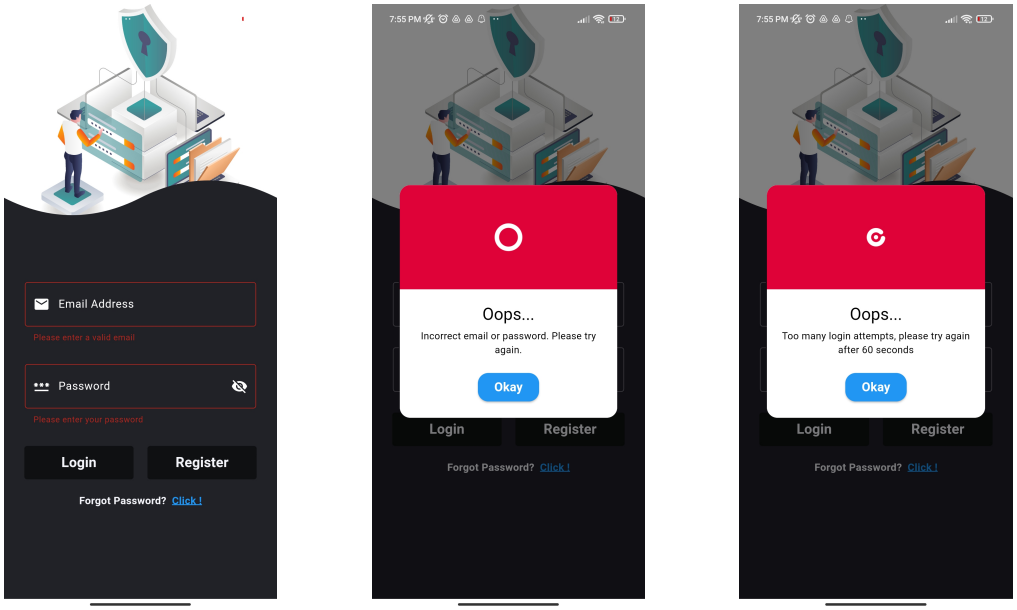


Figure 5: Robust merchant login page

6.4.2 Register Page

* Robust register page, by put fields validators, as phone number must be consist of 10 digits & start with "05", password validator must contain at least 8 characters , at least 1 number, at least 1 small letter, at least 1 capital letter , at least 1 special character .

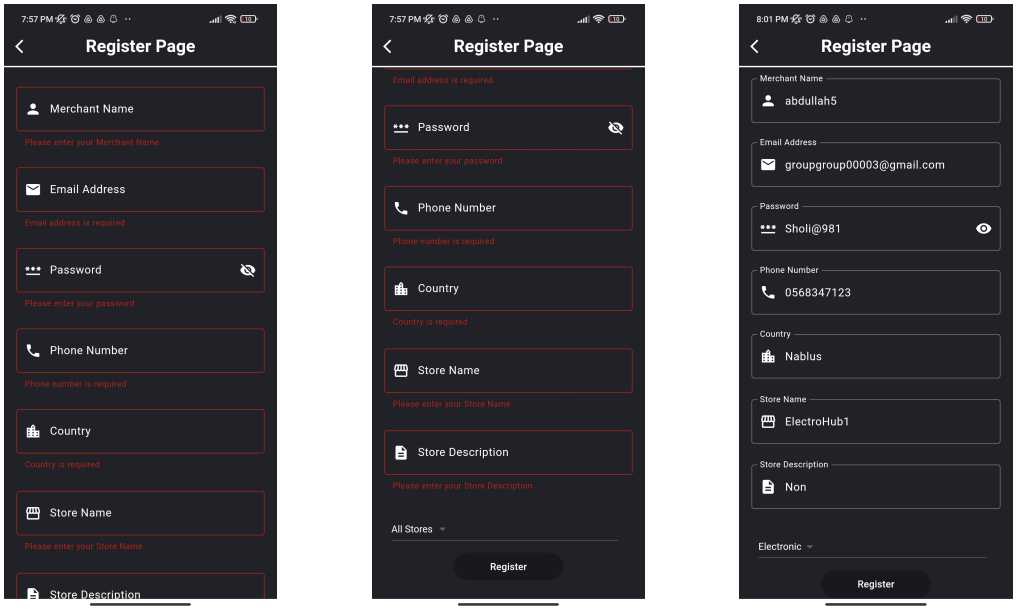


Figure 6: Robust merchant login page

6.4.3 Main Page & Personal Information

* After login or register successfully, redirect merchant to main page, after that merchant could display & update his information, Enter to store management and add payments information.

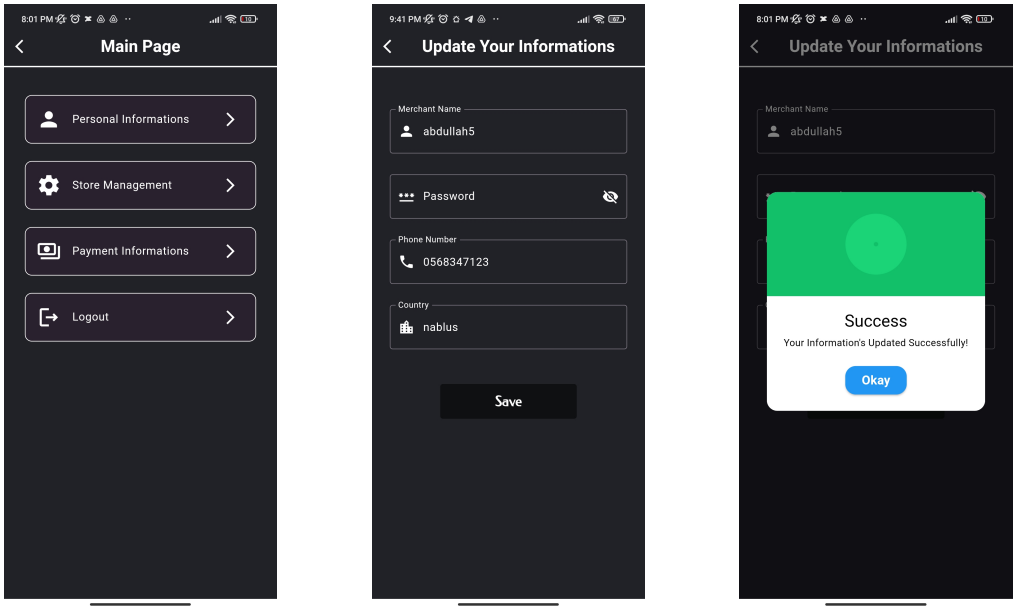


Figure 7: Merchant main page & Merchant display & update information data

6.4.4 Payment Information & Store Management

* Add payment information & Enter to store management page.

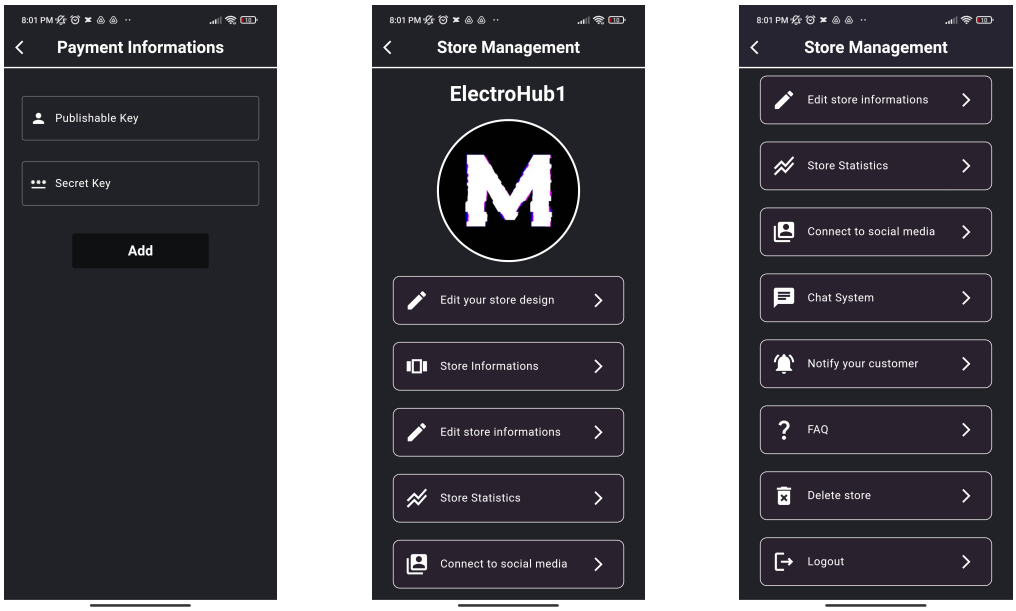


Figure 8: Payment Information & Store Management

6.4.5 Design your store

* Merchant Enter to Edit your store design , this allow merchant to Create & Design his store UI, add image on slider for advertisment, delete images from image slider, allow merchant to create , update & delete category, allow merchant to create product apply discount , put the product in specific category, put quantities & price, update product data also allow merchant to add images to product, allow merchant to activate & de-activate slider, categories & products

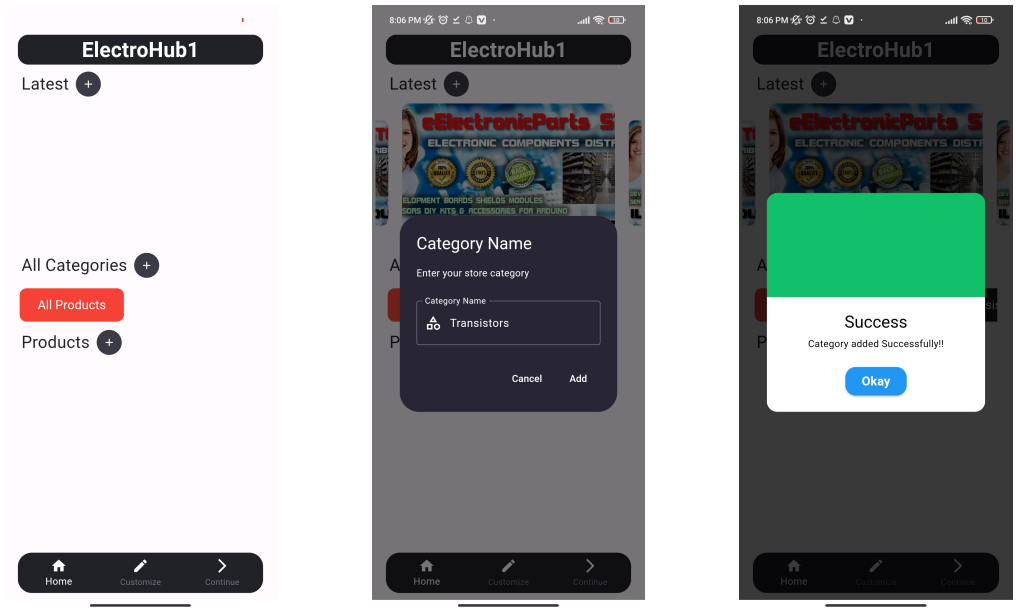


Figure 9: Edit store design, add image to slider, delete image from slider, add, update and delete category

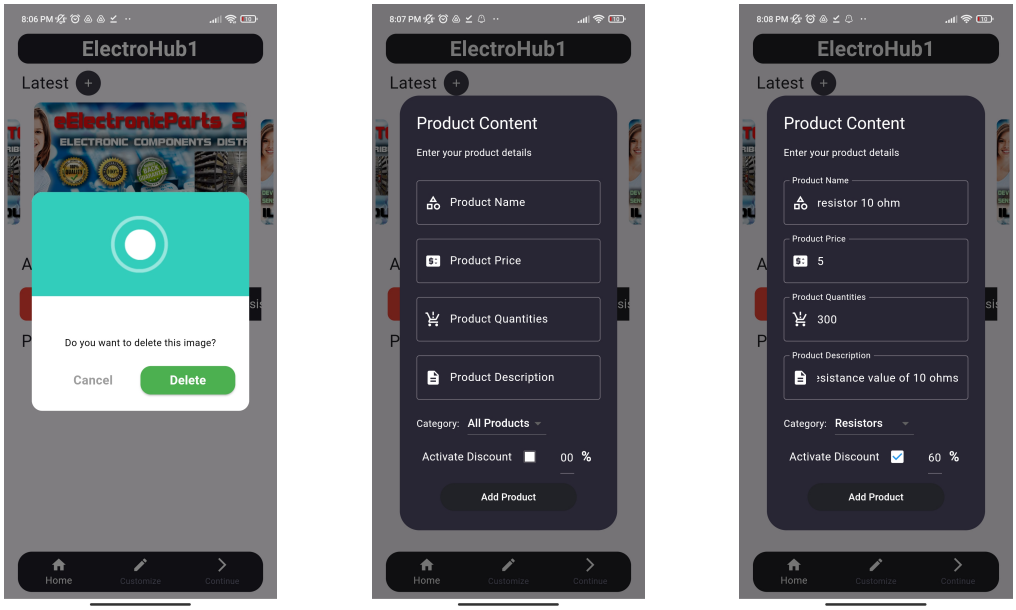


Figure 10: Create new Product

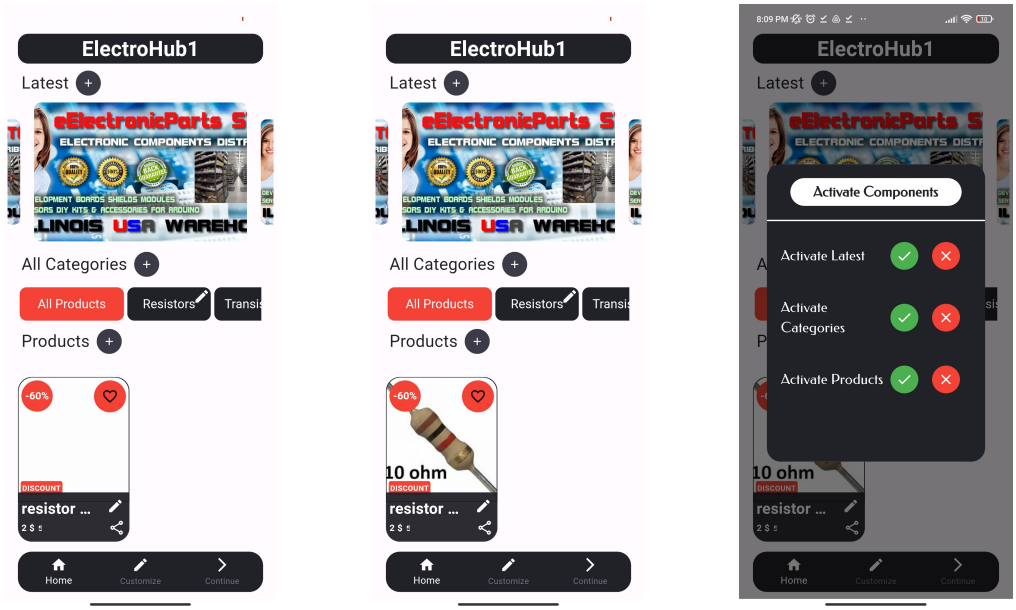


Figure 11: Product added successfully, upload image for product, activate store components

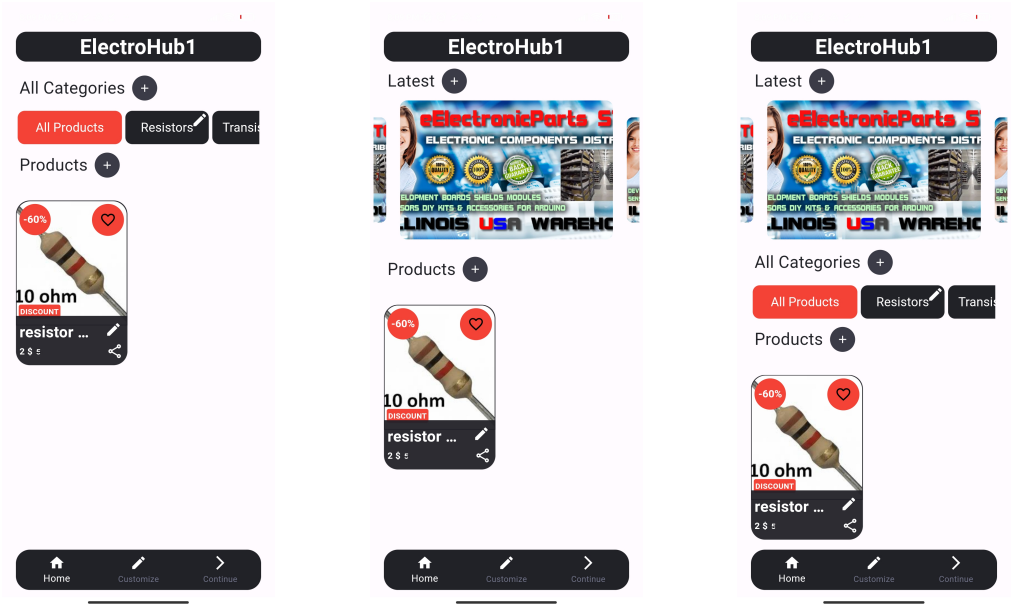


Figure 12: Preview activate & de-activate components

* Select Store design page, this allow merchant to select background, boxes , clipping , primary text, secondary text color,via color picker window, allow merchant to customize components design by put them smooth or solid, give the merchant 3 different design options , provide preview button to allow merchant to preview his design before publish it, provide another button which allow merchant to publish store , then the store will published & user could interact with it.

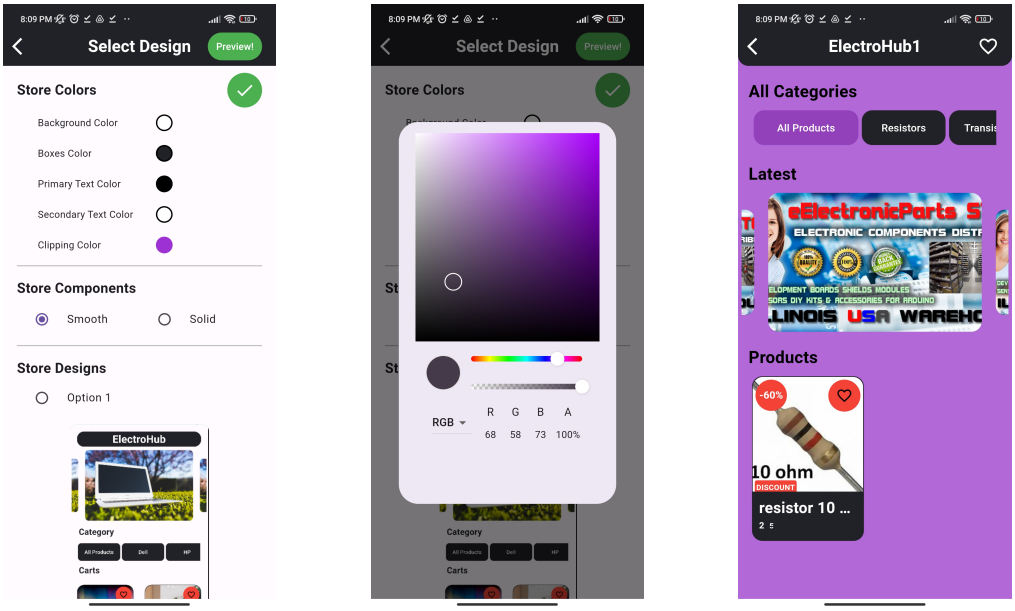


Figure 13: Select & customize your store design

* view all design options

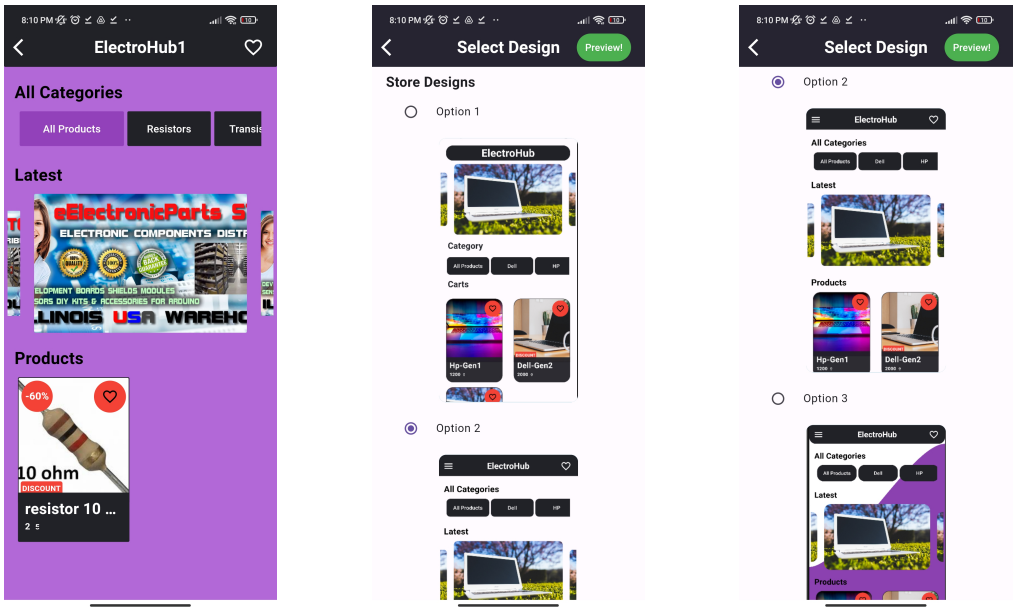


Figure 14: view all design options

* We note that error message displayed , it indicate that merchant doesn't insert his payment information

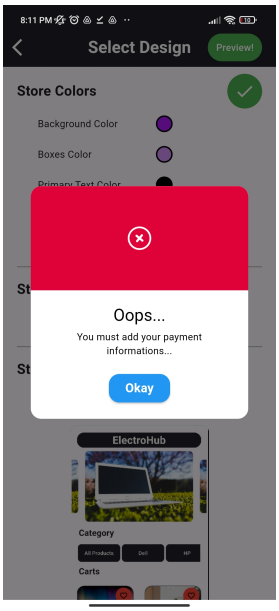


Figure 15: Error message related to payment information

* We go to lahza.io, & add the keys for Trial version

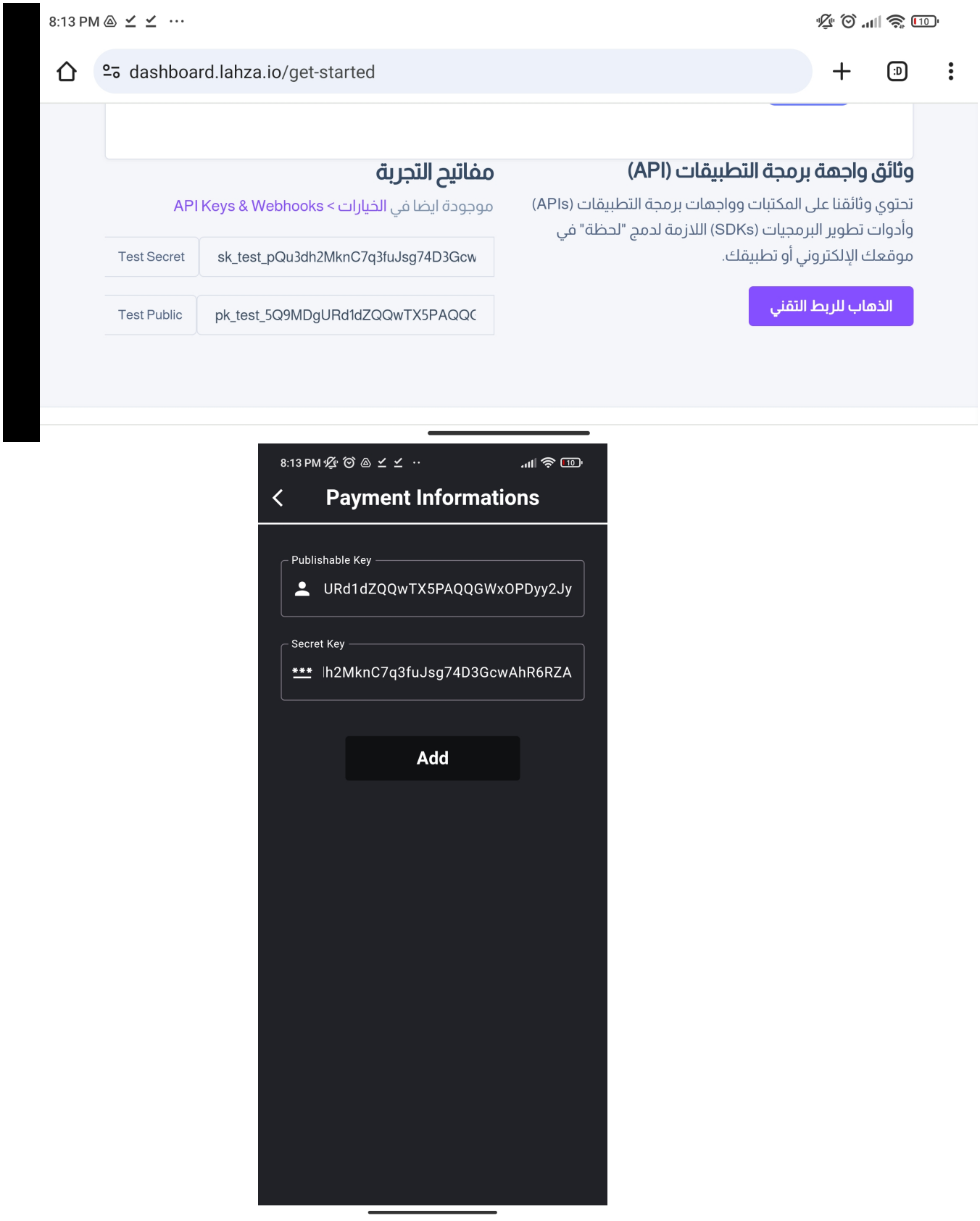


Figure 16: add the keys for Trial version

* we note that keys added successfully & store published successfully! & Display store informations

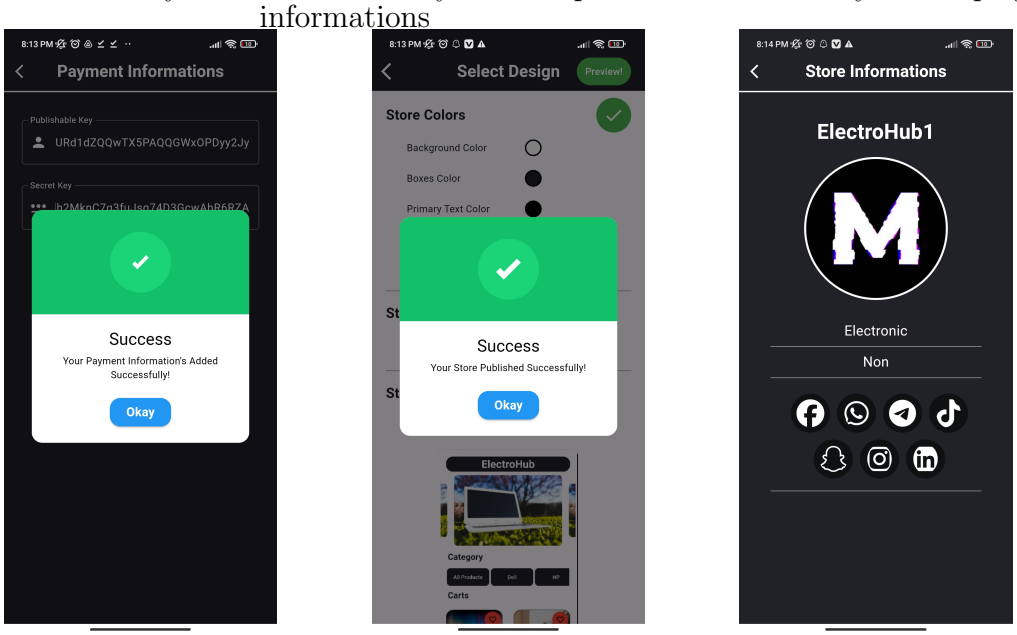


Figure 17: publish store successfully & display store information

6.4.6 display & edit store information

* merchant doesn't add social media account link, display & edit store information

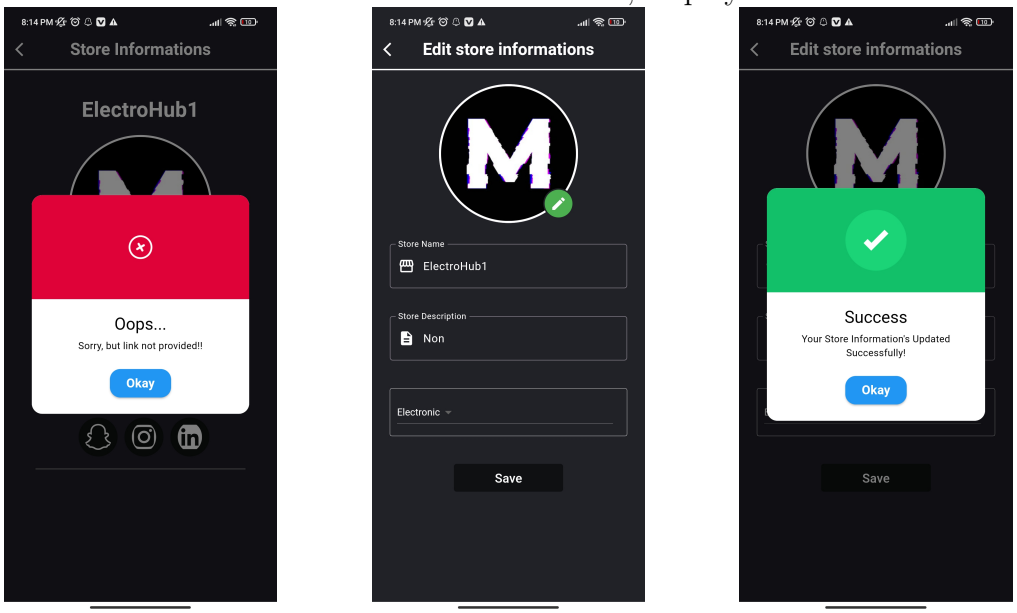


Figure 18: merchant doesn't add social media account link, display & edit store information

6.4.7 Social Media Accounts & Chat System

* allow merchant add his social media accounts, also allow merchant to enter to chat system, it consist of 2 parts, part allow merchant to chat with his customer , another part allow merchant to chat with admin.

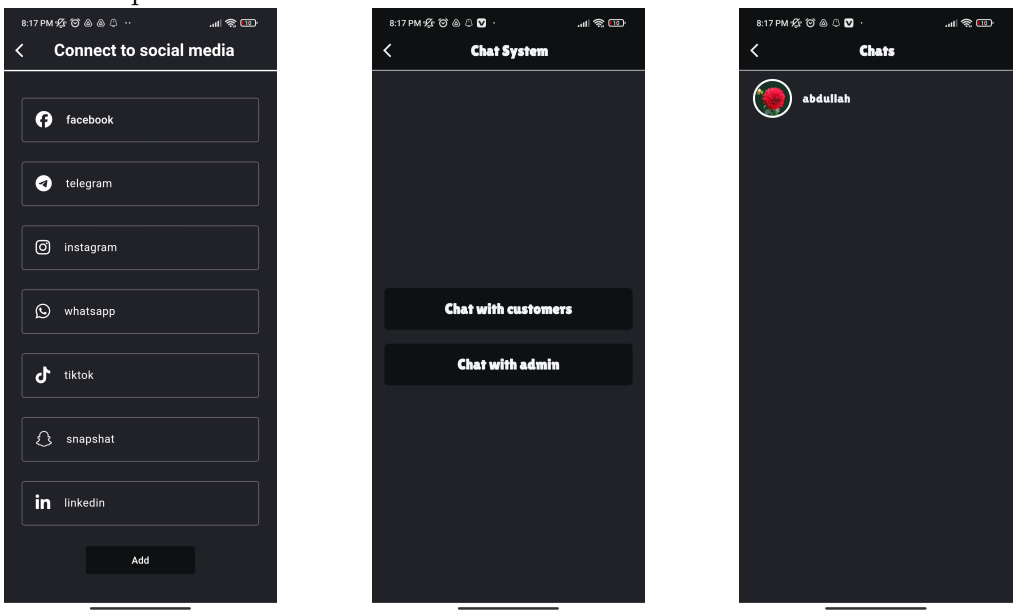


Figure 19: add social media account & chat system

6.4.8 Chating pages & Custom Notifications

* Open chat page with specific customer, open chat page with admin, also allow merchant to create custom notification

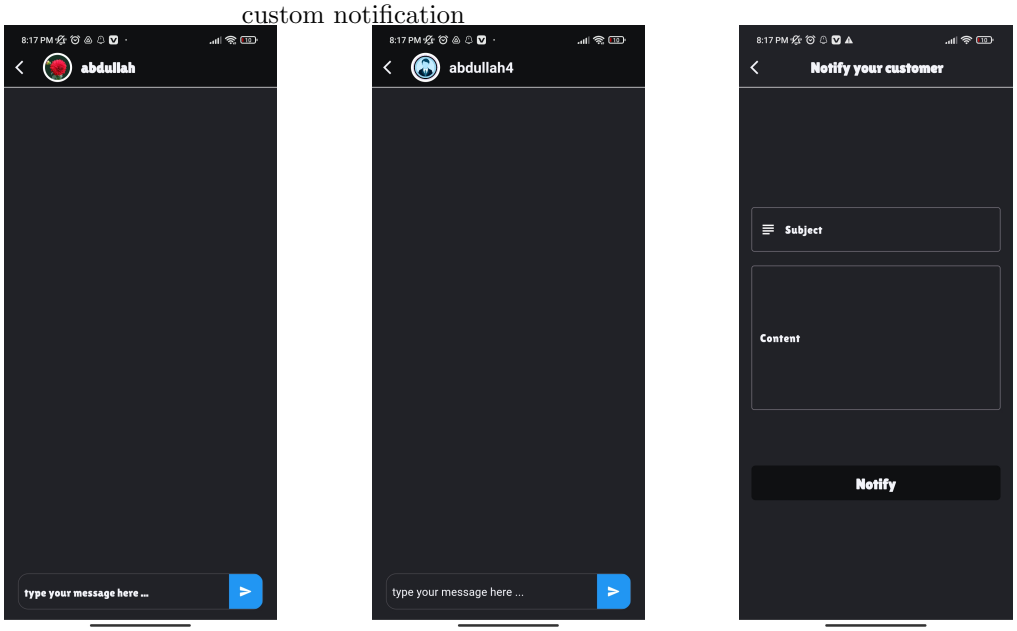


Figure 20: add social media account & chat system

6.4.9 FAQ Page, delete store & Logout

* FAQ (Frequently Asked Questions) page , delete store & logout

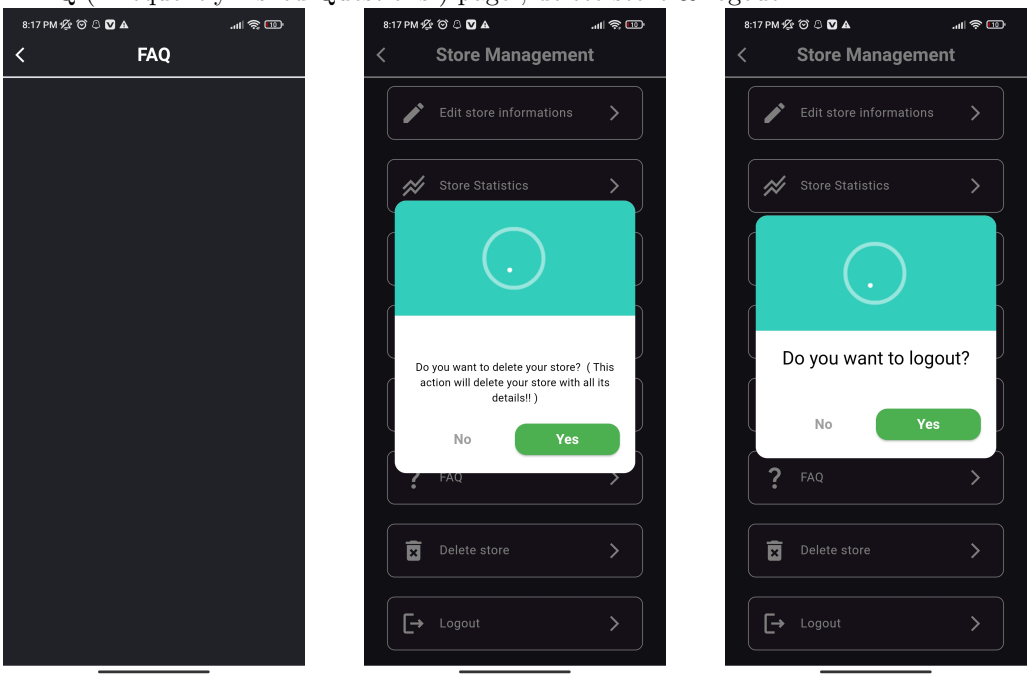


Figure 21: FAQ page , delete store & logout

6.4.10 Store Statistics Page

* Store Statistics page

Figure 22: Store Statistics Page

6.5 Customer Side

6.5.1 Customer Register Page, main page & Drawer

* Inside main page allow customer to browse different stores & Filter stores depend on their categories.

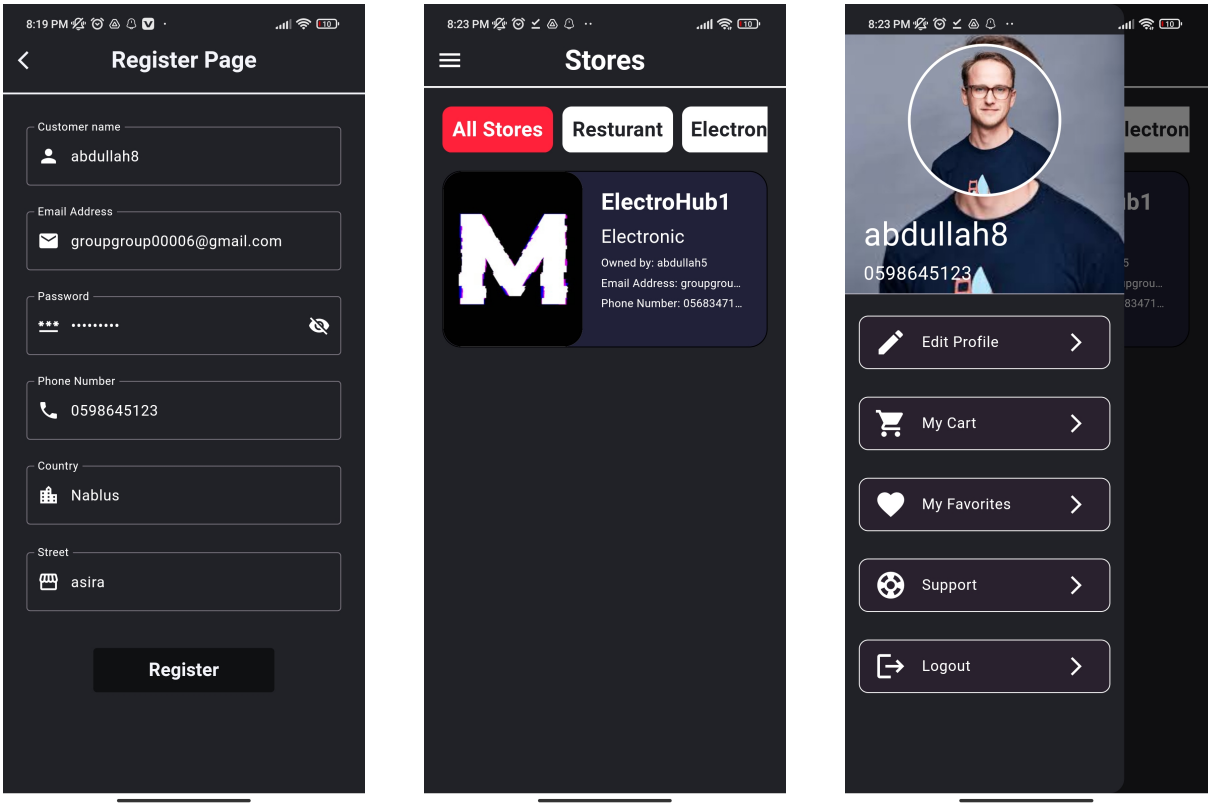


Figure 23: Customer Register Page, Main Page & Drawer

6.5.2 Display & Edit Customer Profile, Enter to Shopping cart and favorite products

* allow customer to display & update his information , enter to cart, inside cart allow user but products from different stores & the money will redirect into each merchant card independently, also Favourite products page allow customers put products inside favorite list from different stores

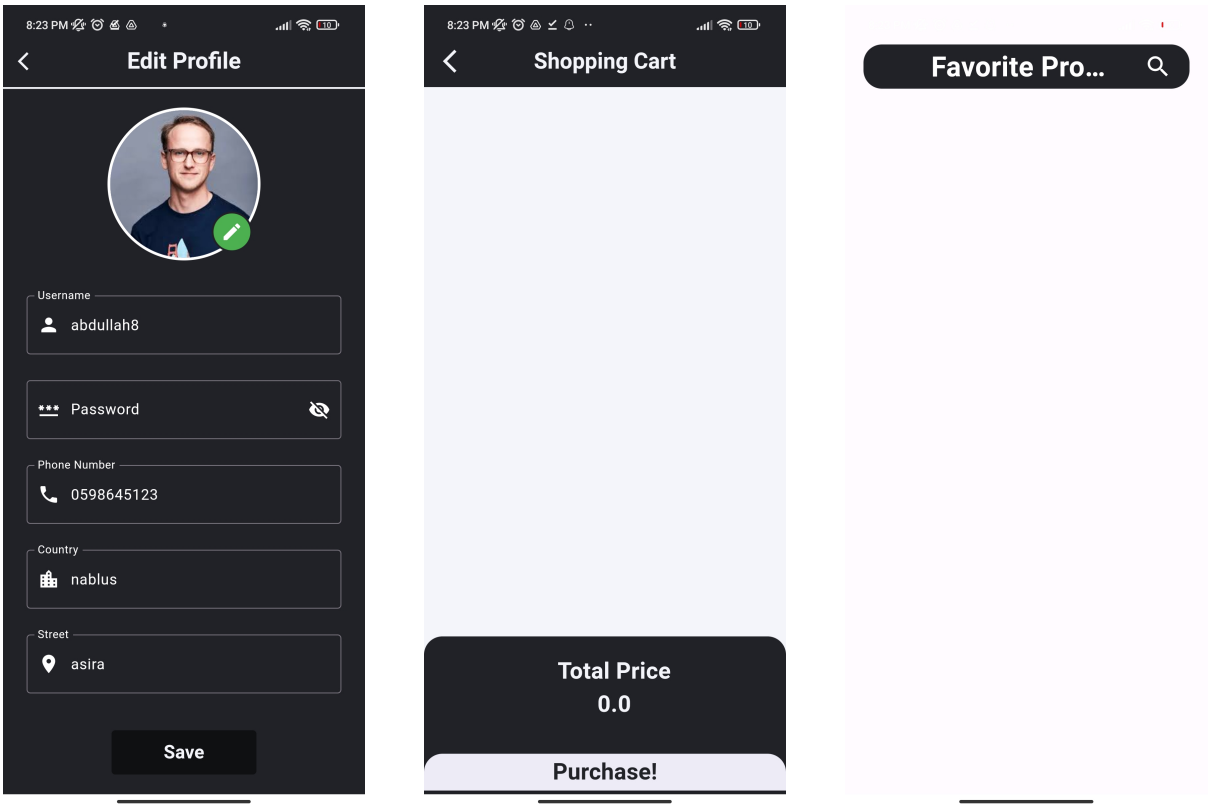


Figure 24: Display & Edit Customer Profile, Enter to Shopping cart and favorite products

6.5.3 Help & Support and specific store

* Help & Support allow customers to send email to admin, also enter to specific store .

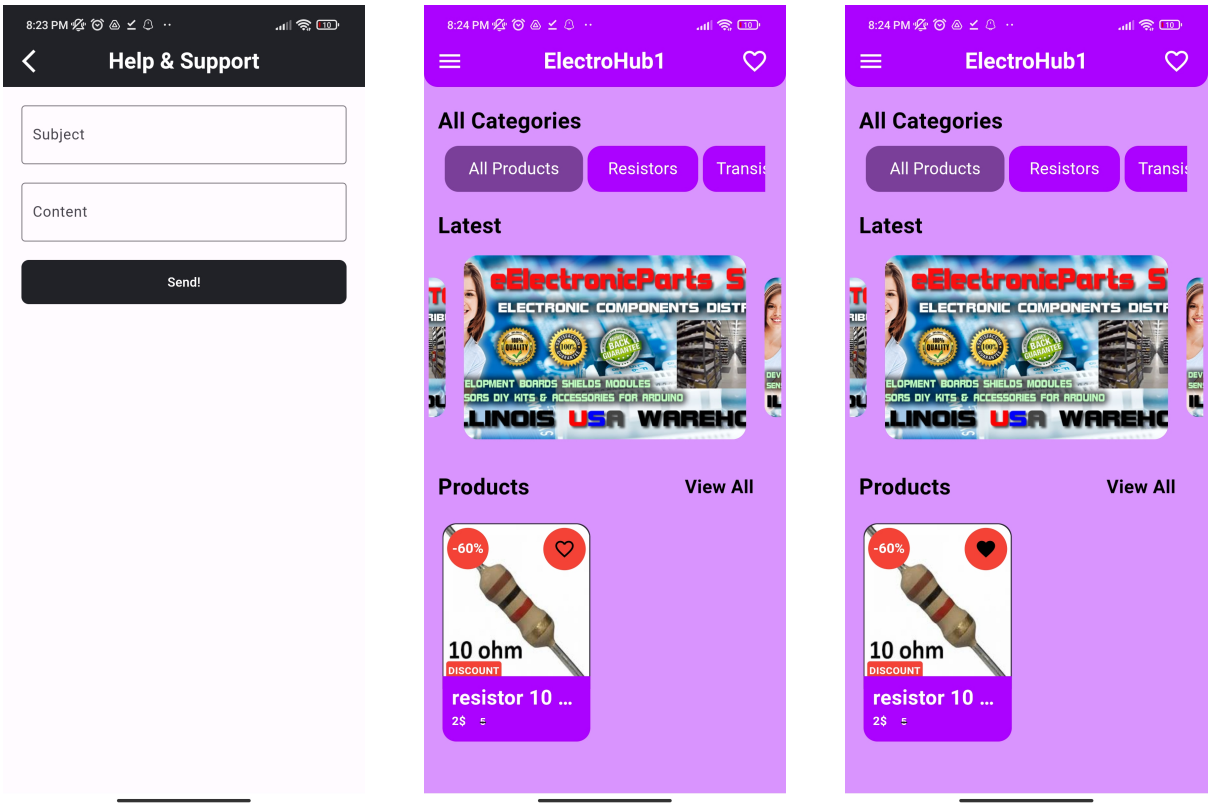


Figure 25: Help & Support and specific store

6.5.4 Favorite Products, Enter to specific product & add product to cart successfully!

* Inside favorite product list allow customer to display all favorite products & search box allow customer to search about specific product, with modern search bar which filtered each inserted character, another page for display product & enter quantity of product to buy

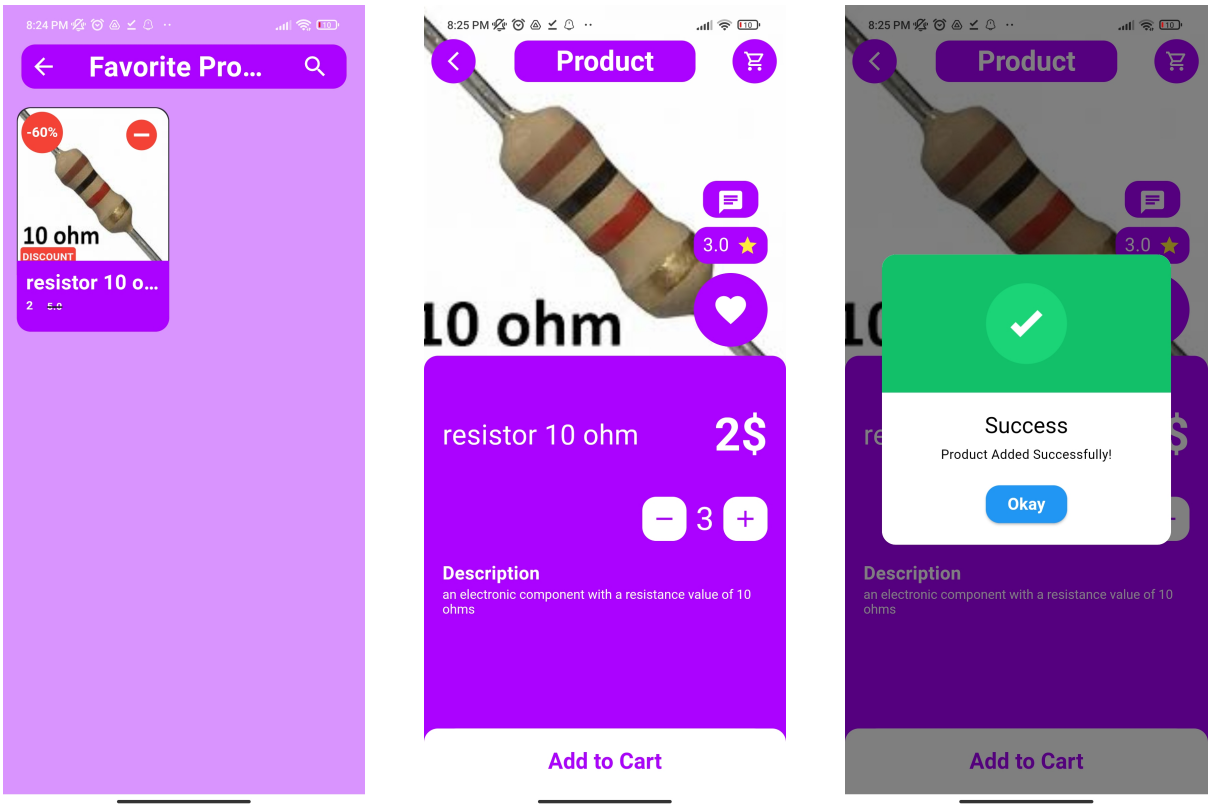


Figure 26: Favorite Products, Enter to specific product & add product to cart successfully!

6.5.5 Shopping Cart

* Now enter to shopping cart, for checkout the product add payment informations & buy the product.

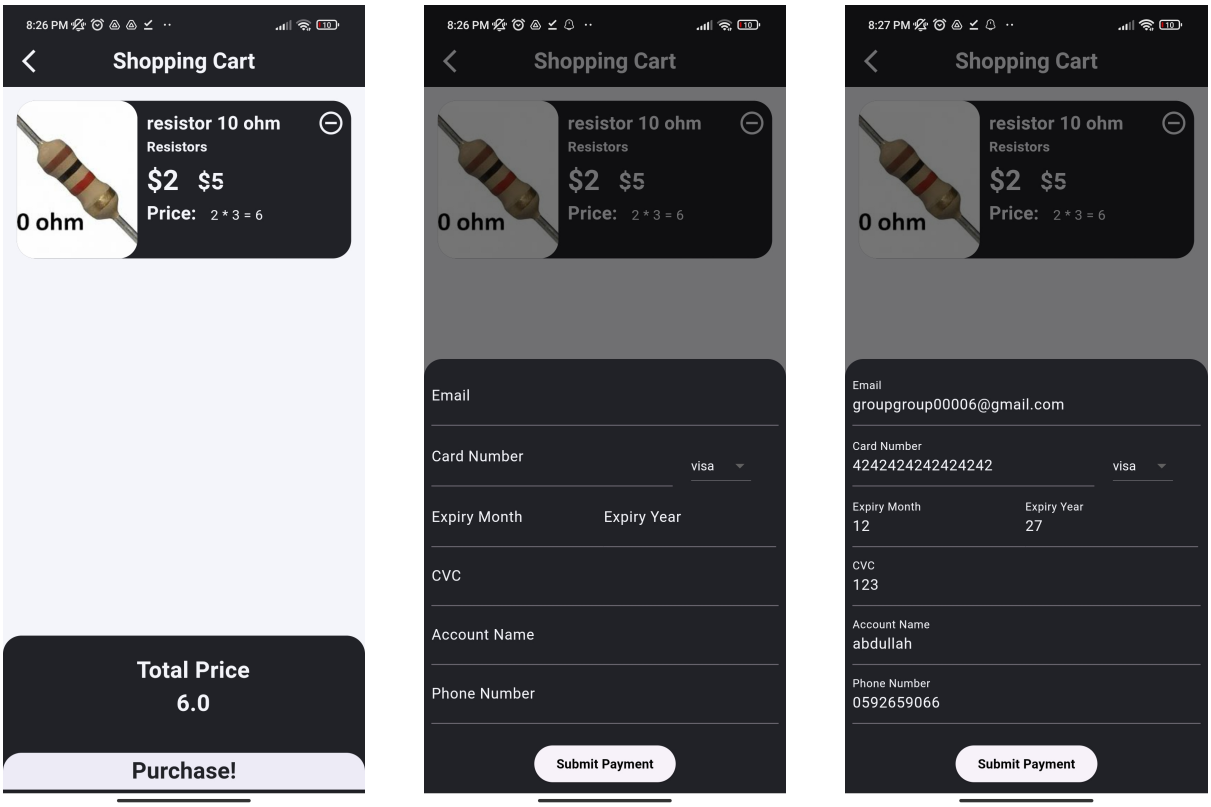


Figure 27: Shopping Cart

6.5.6 Rating & Reviews

* Allow customer to review the product rate , & allow customer to rate specific product & add his comment

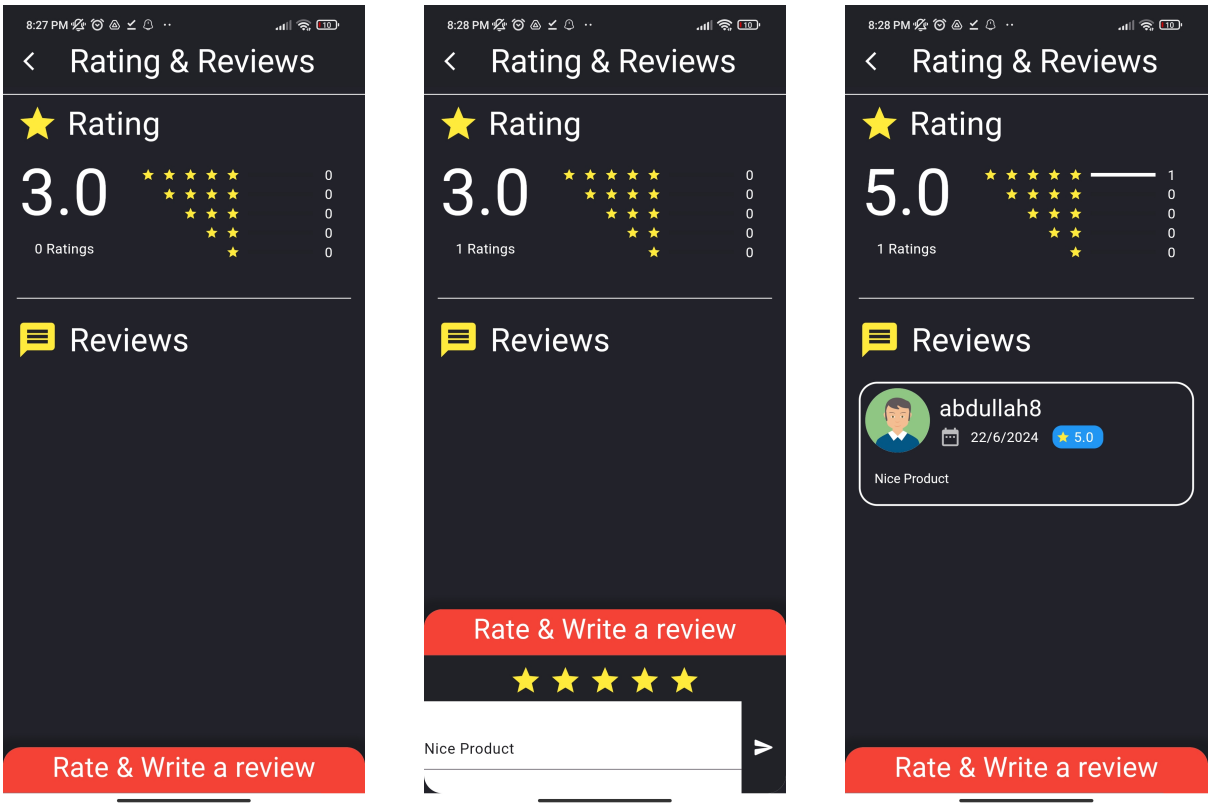


Figure 28: Rating & Reviews

6.5.7 Notifications

* Allow customer to turn on or turn off notification, when turn off the customer doesn't receive any notification from this store merchant.

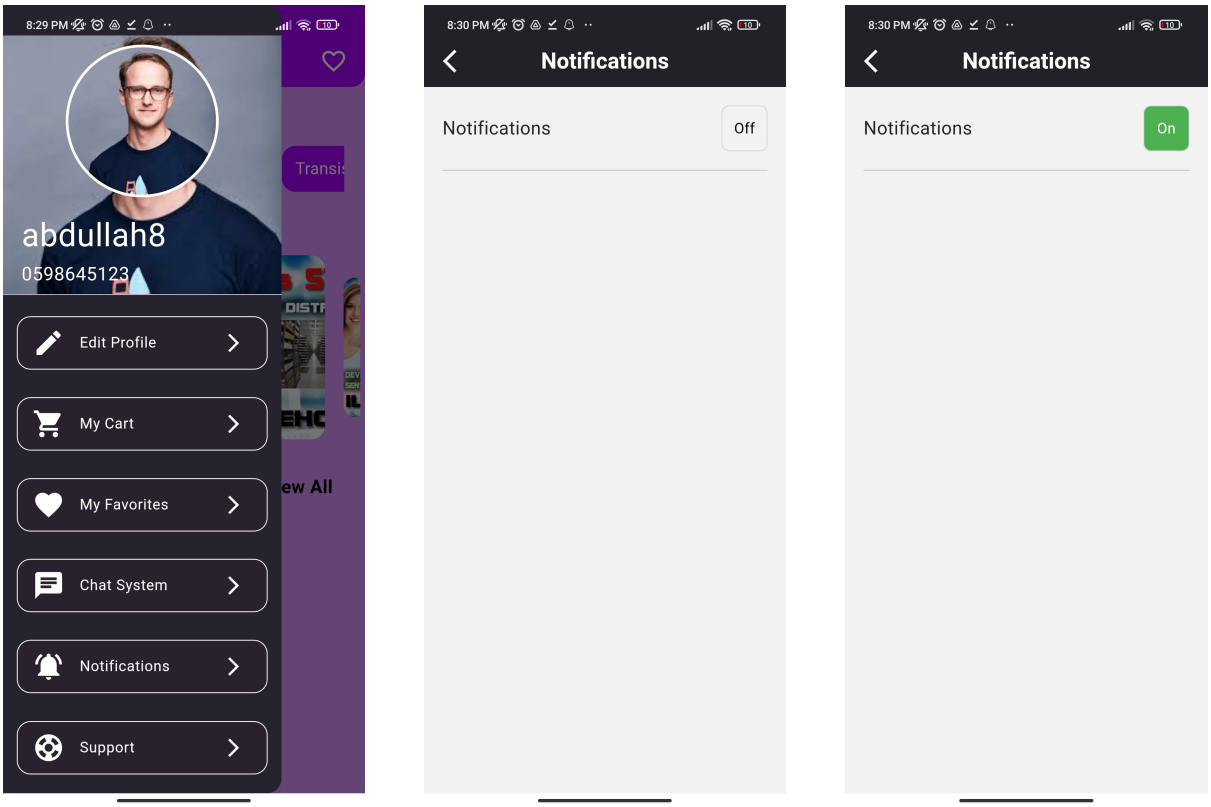


Figure 29: Notifications

6.5.8 Help & Support

* allow customers to send his questions to the merchant

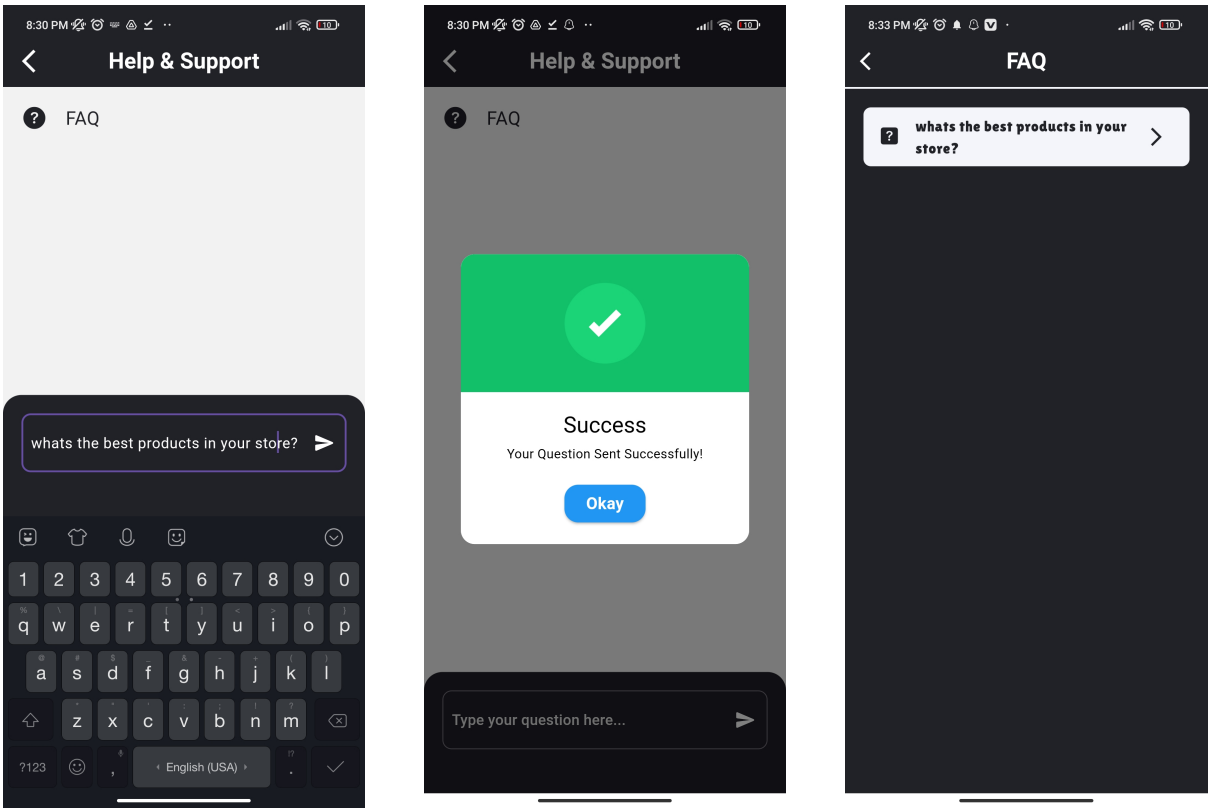


Figure 30: Help & Support

6.5.9 Help & Support

* allow customers to send his questions to the merchant

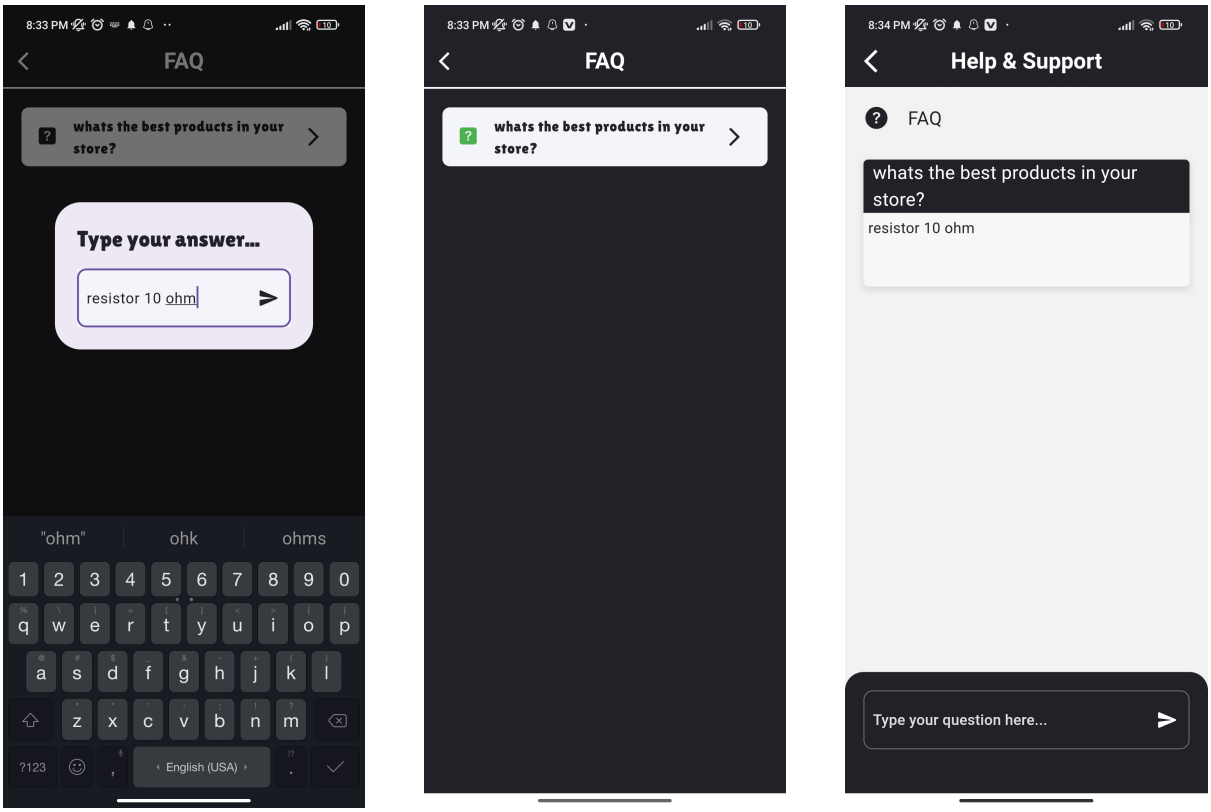


Figure 31: Help & Support

6.5.10 Chatting Page

* allow customers chat with merchant

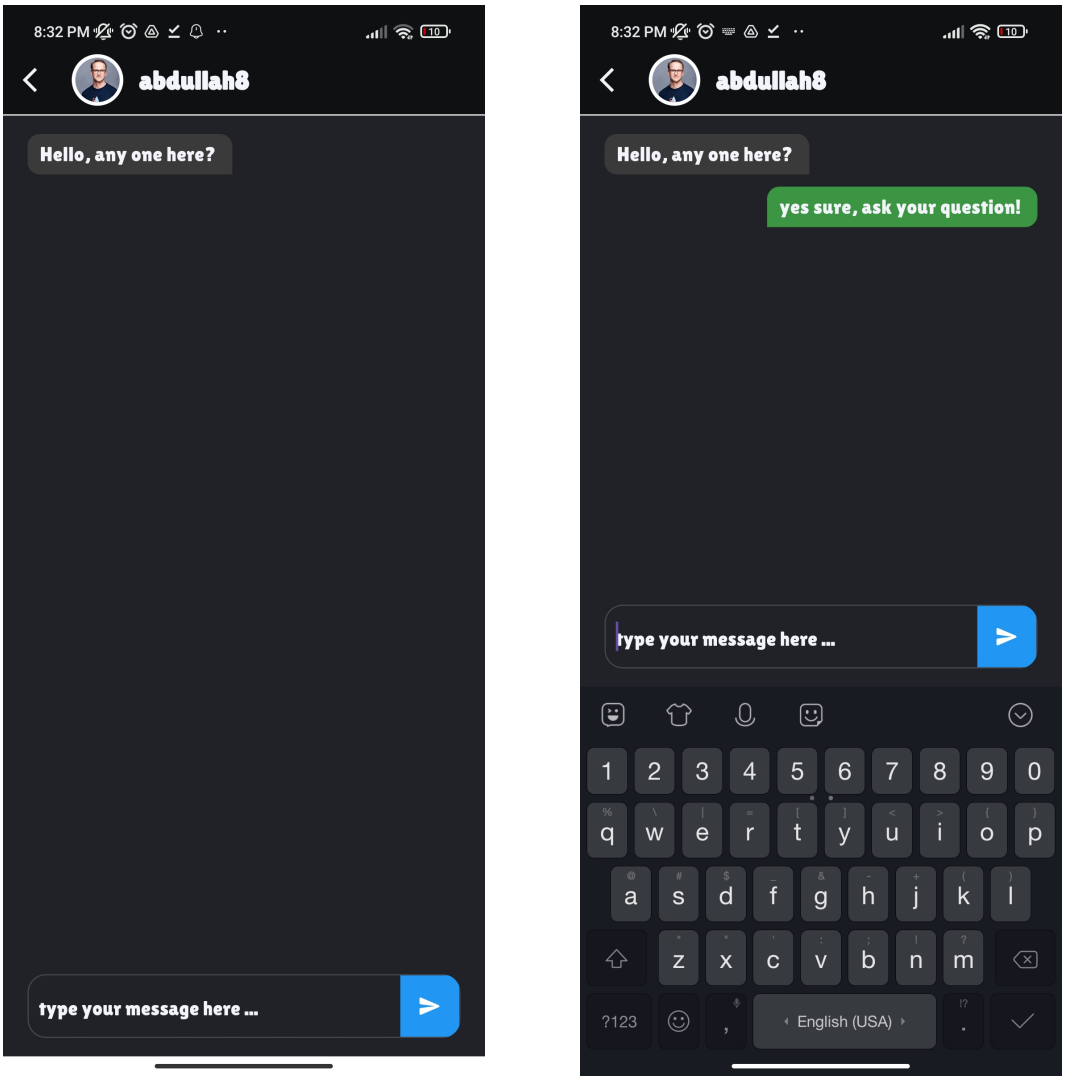


Figure 32: Chatting Page

6.5.11 Create a Custom Notification

* allow customers to create & send notification to his customers

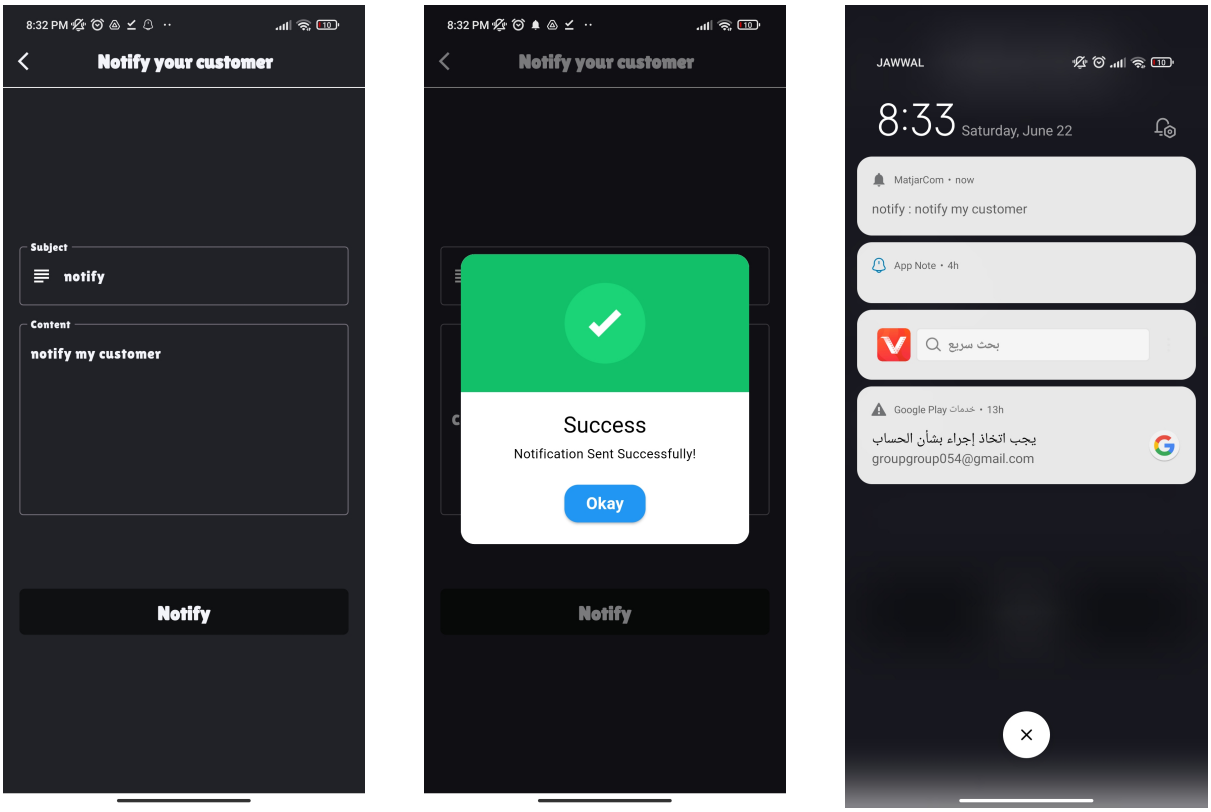


Figure 33: Create a Custom Notification

6.5.12 Forgot & Reset Password

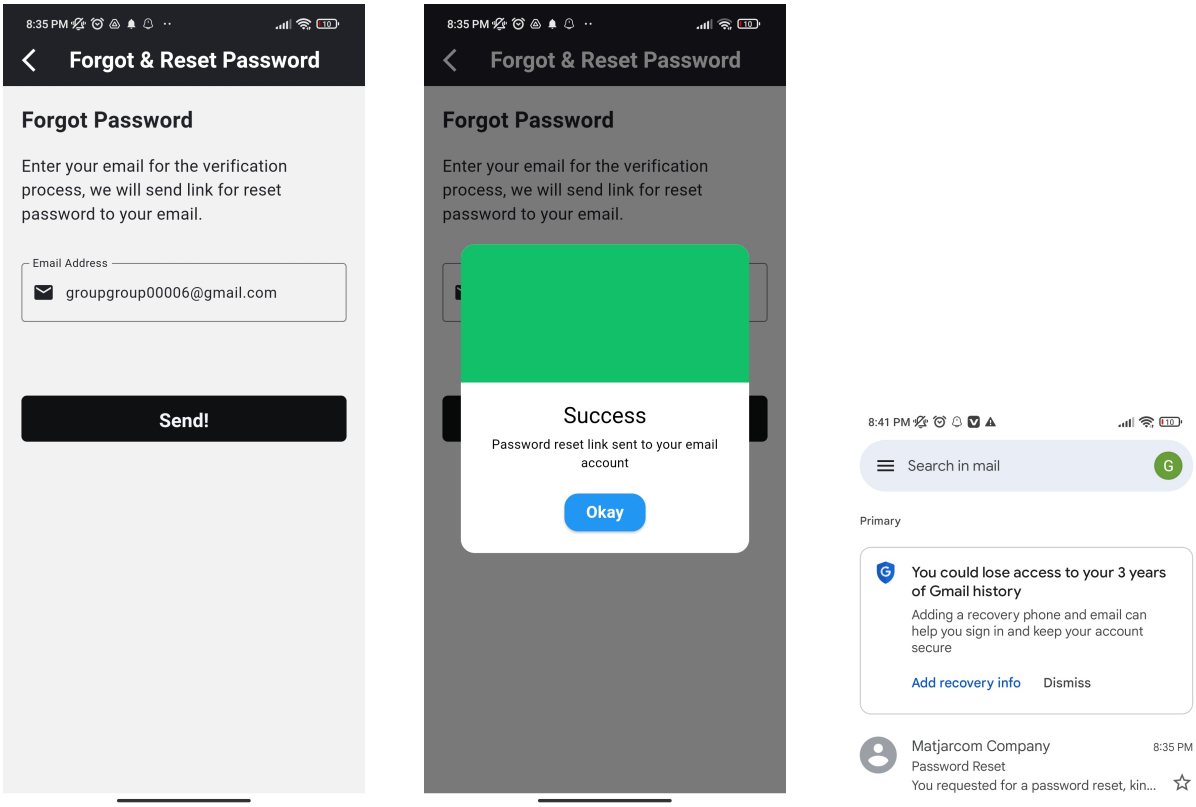


Figure 34: Forgot & Reset Password

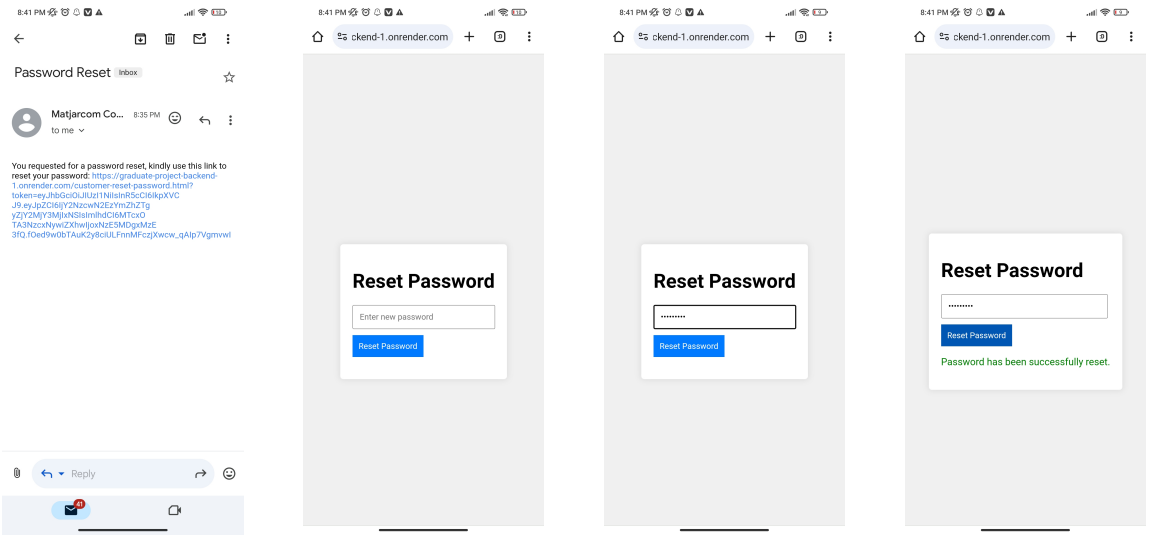


Figure 35: Forgot & Reset Password

6.6 Admin Dashboard Side

6.6.1 Login Page

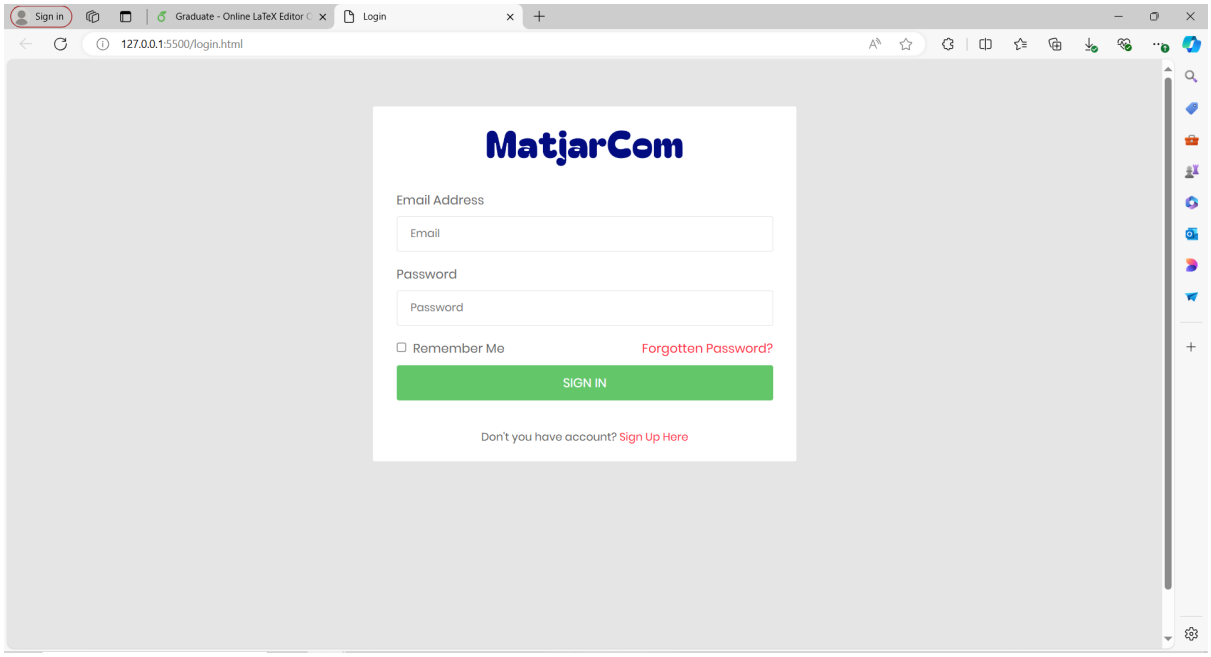


Figure 36: Forgot & Reset Password

6.6.2 Login Page

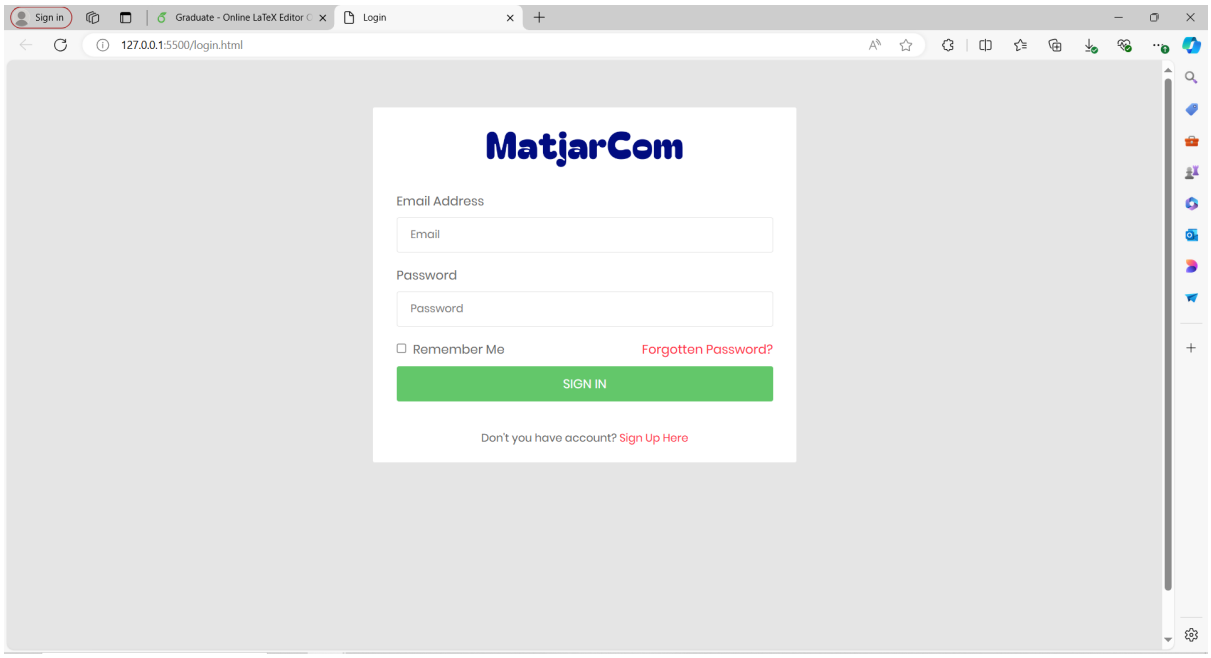


Figure 37: Login Page

6.6.3 Register Page

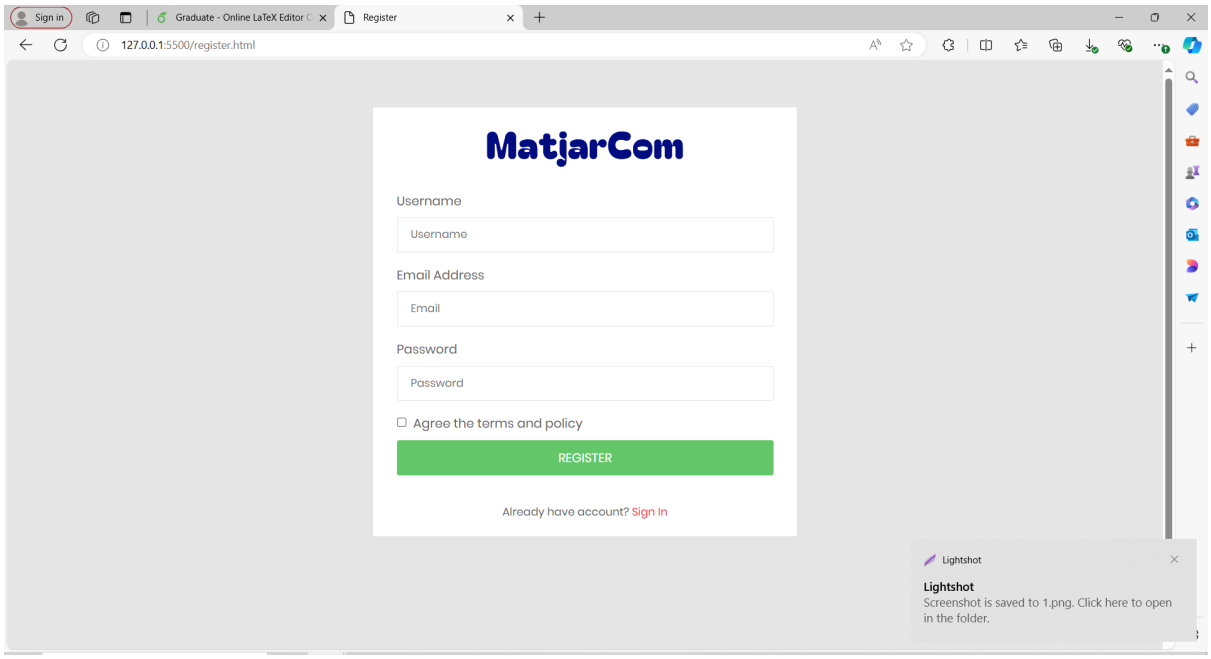


Figure 38: Register Page

6.6.4 Forgot & Reset Password Page

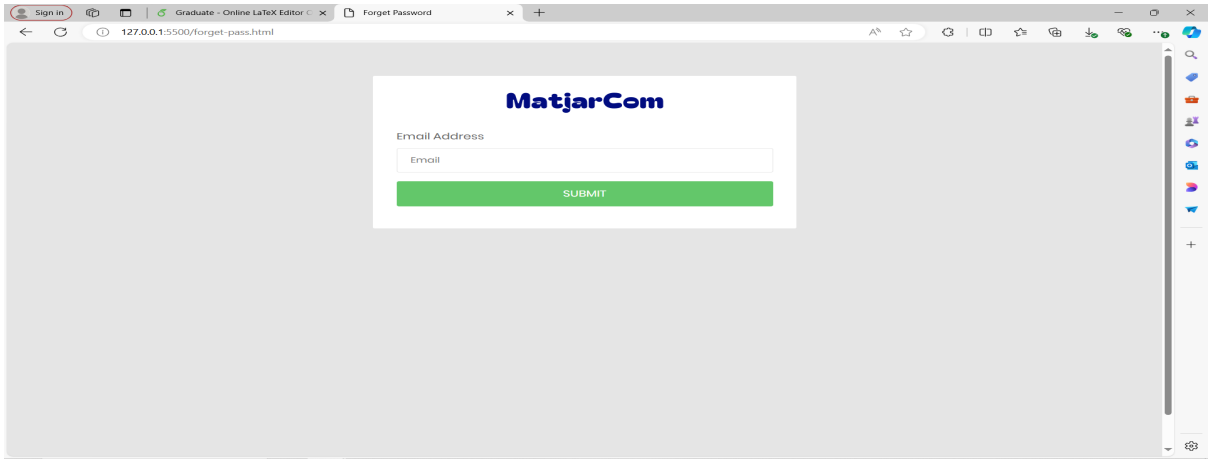


Figure 39: Forgot & Reset Password Page

6.6.5 Admin Dashboard

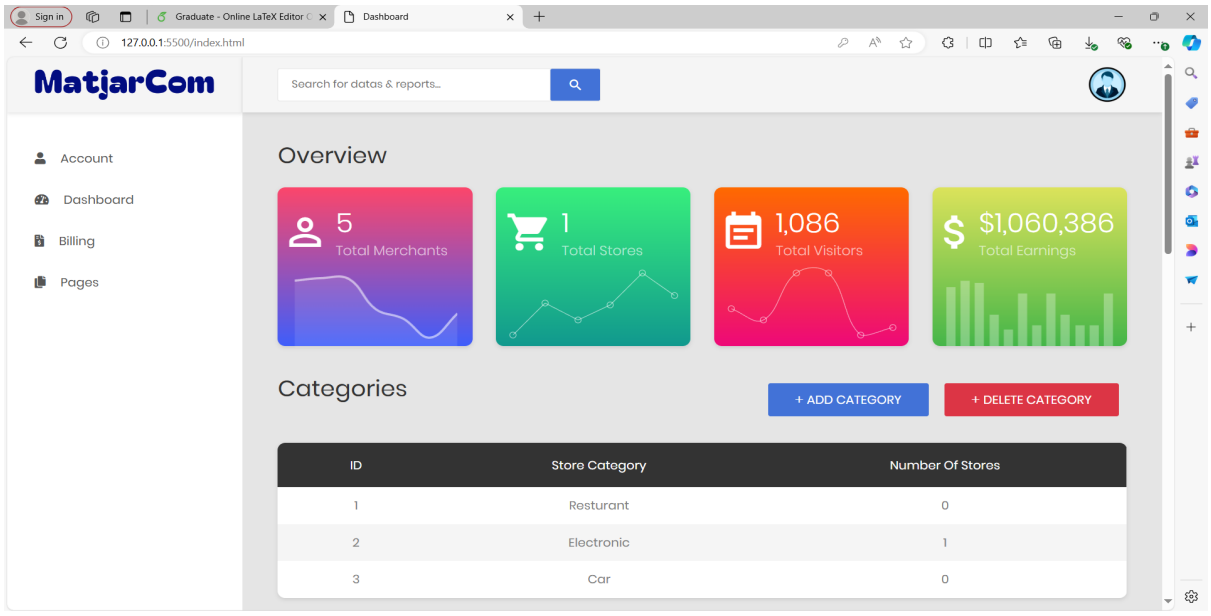


Figure 40: Admin Dashboard

6.6.6 Add & Delete Category

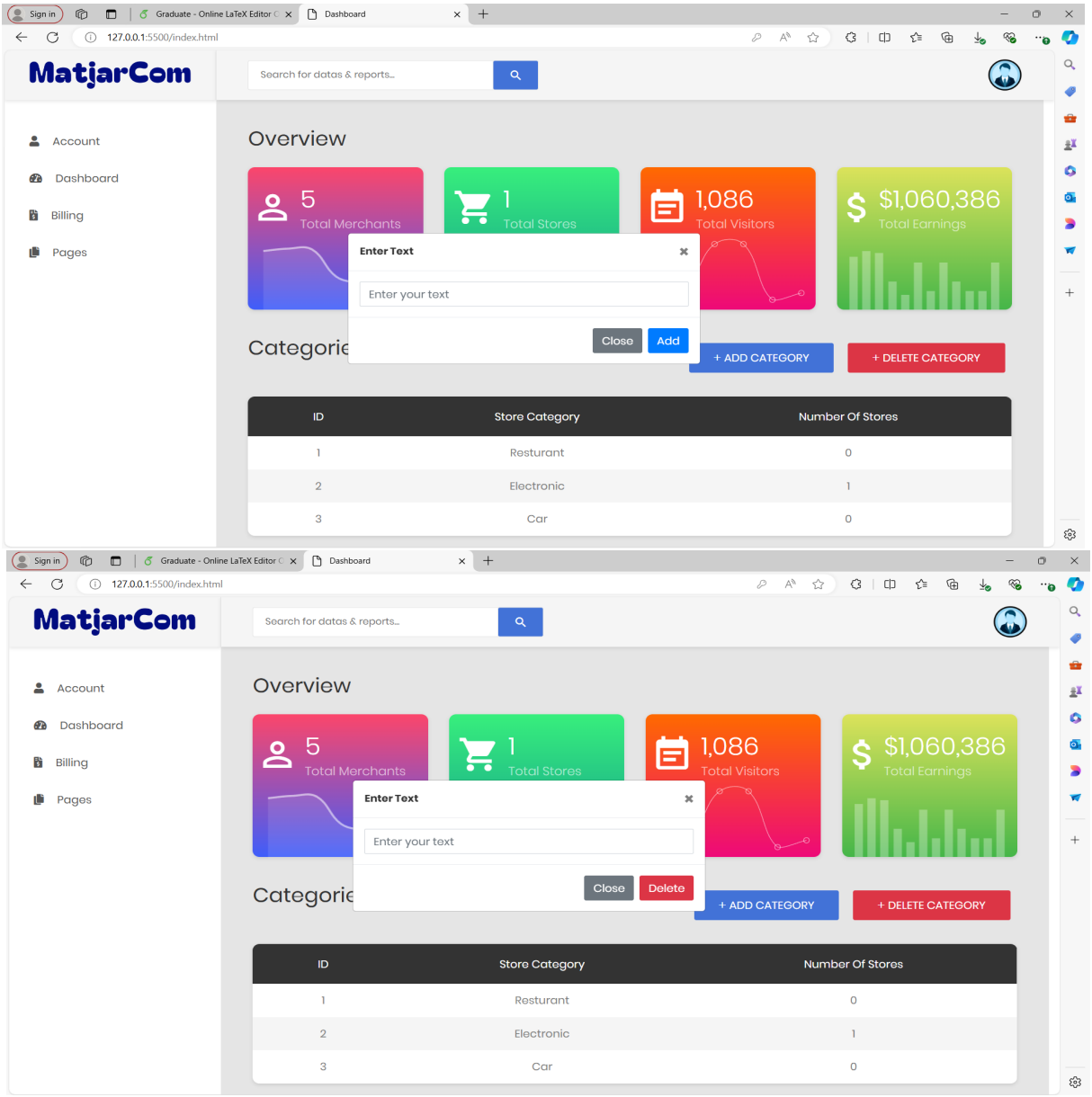


Figure 41: Add & Delete Category

6.6.7 Store & merchants Tables

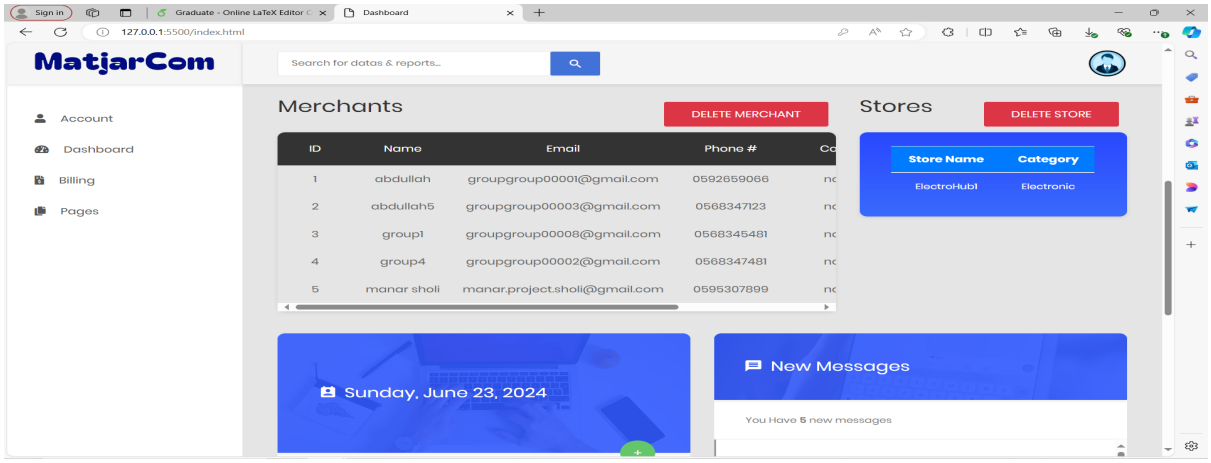


Figure 42: Store & merchants Tables

6.6.8 Tasks & Messages

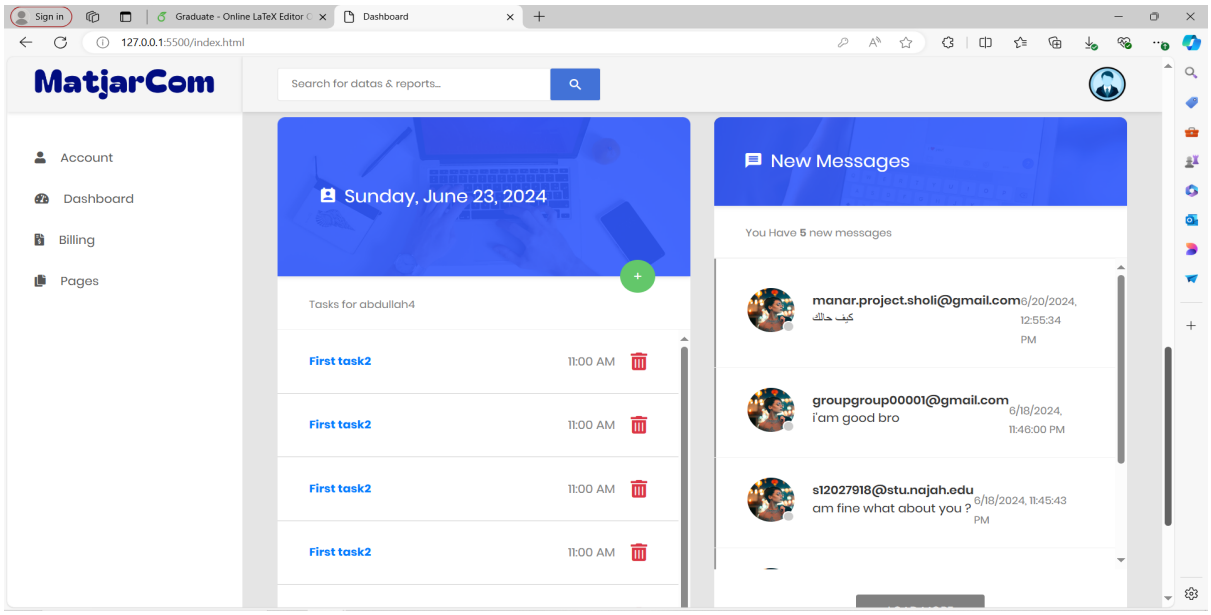


Figure 43: Tasks & Messages

6.6.9 Create new task

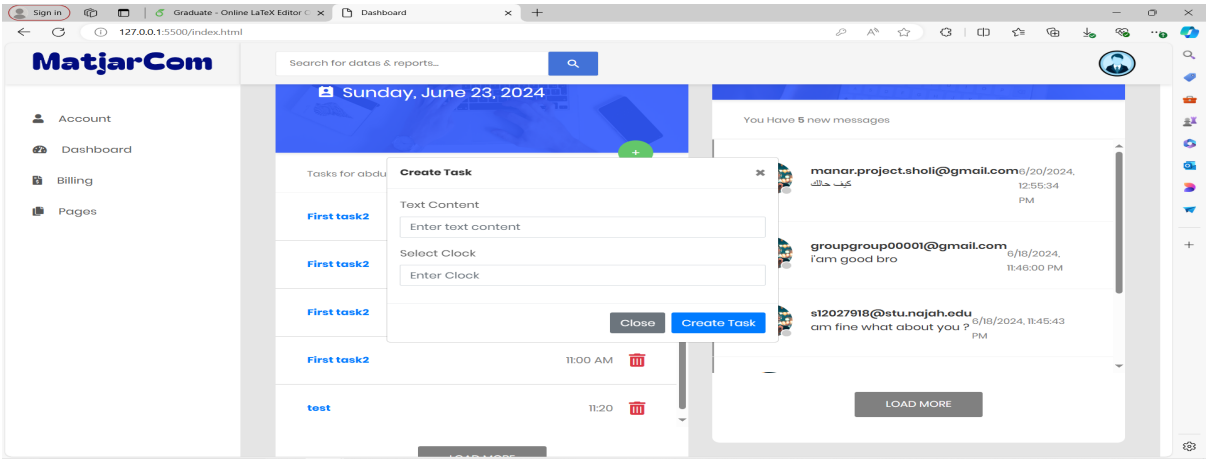


Figure 44: Create new task

6.6.10 Delete Task

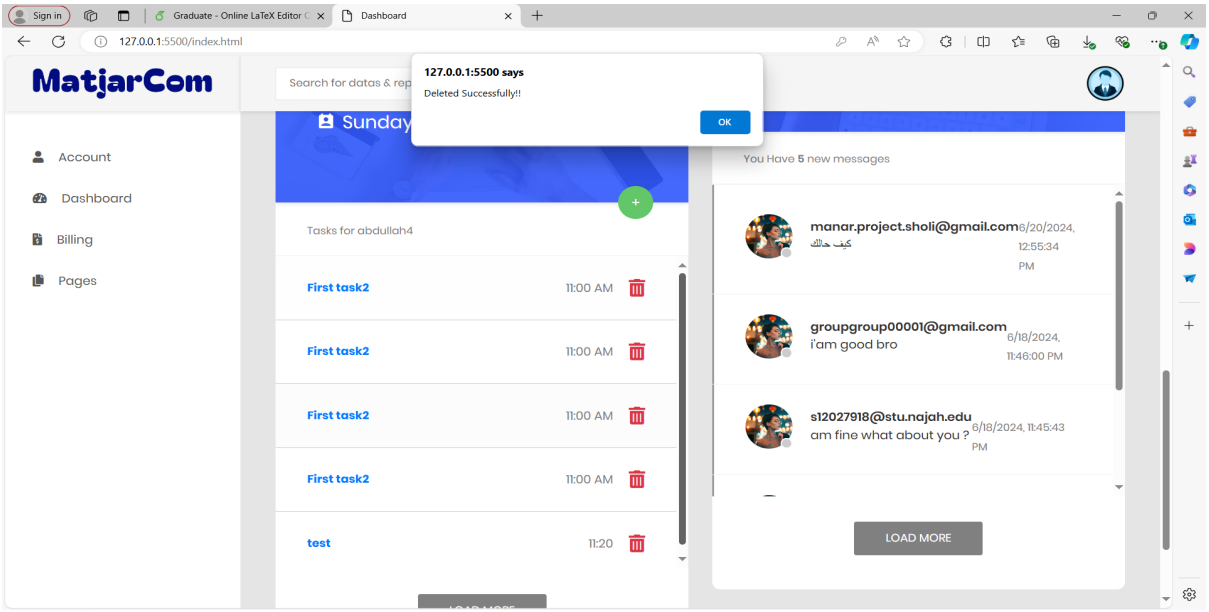


Figure 45: Delete Task

6.6.11 Enter to chat with specific merchant

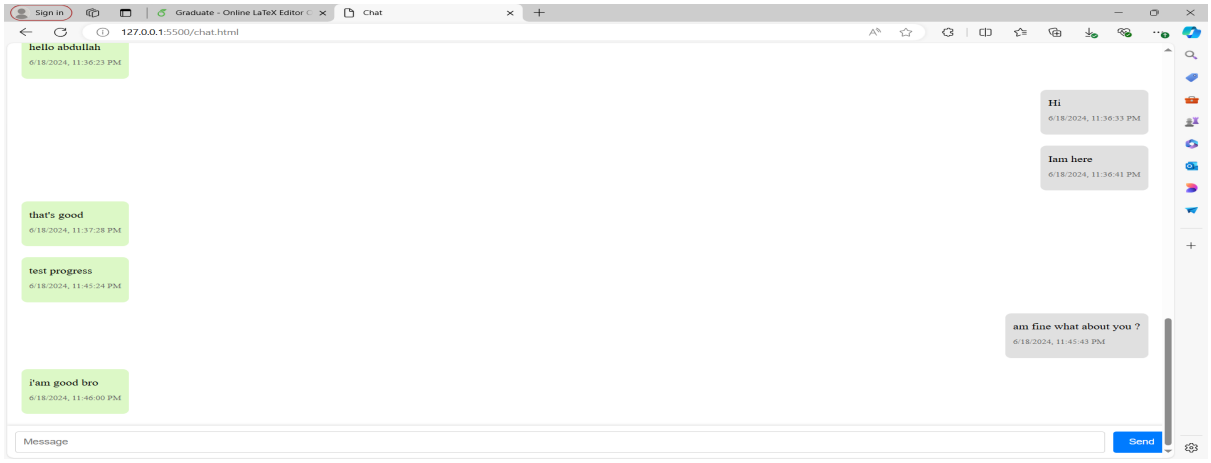


Figure 46: Enter to chat with specific merchant

6.6.12 Delete Merchant or Store

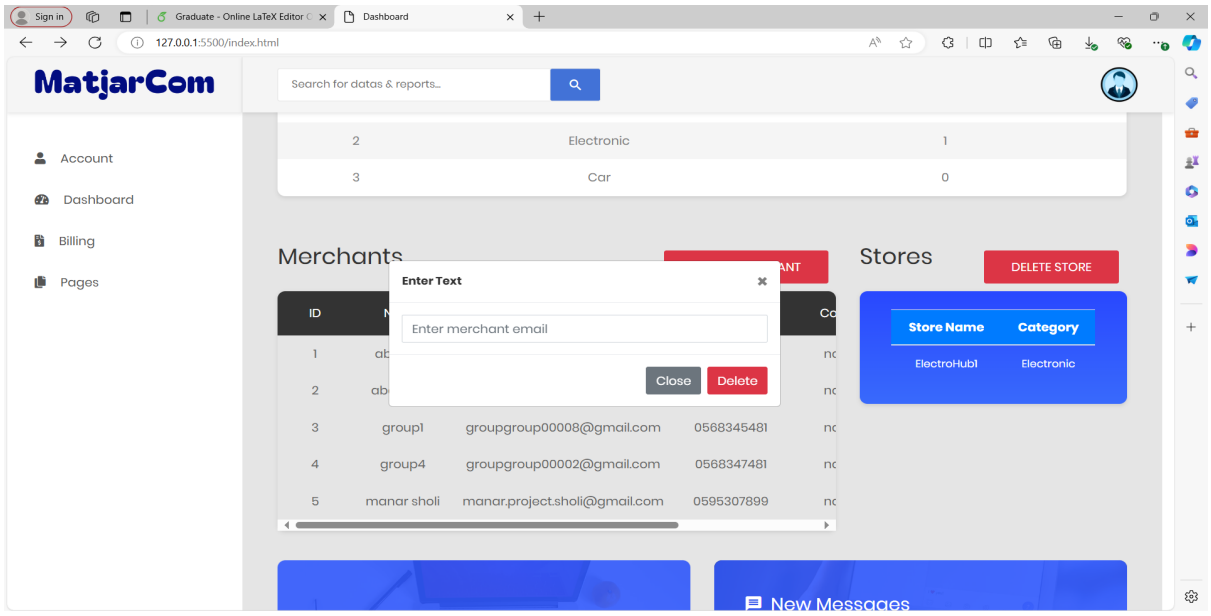


Figure 47: Delete Merchant or Store

6.6.13 Admin account

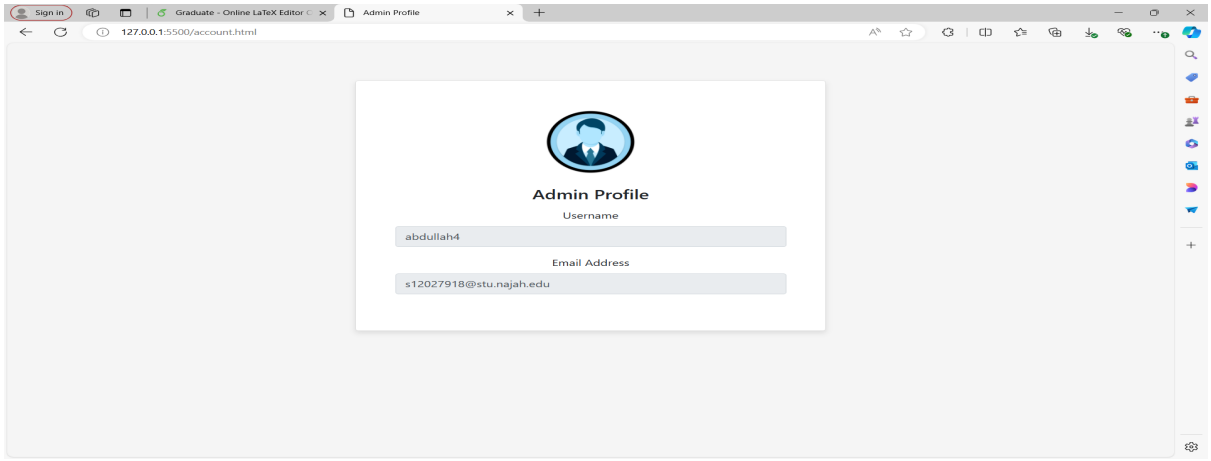


Figure 48: Admin Account

6.6.14 Billing Information

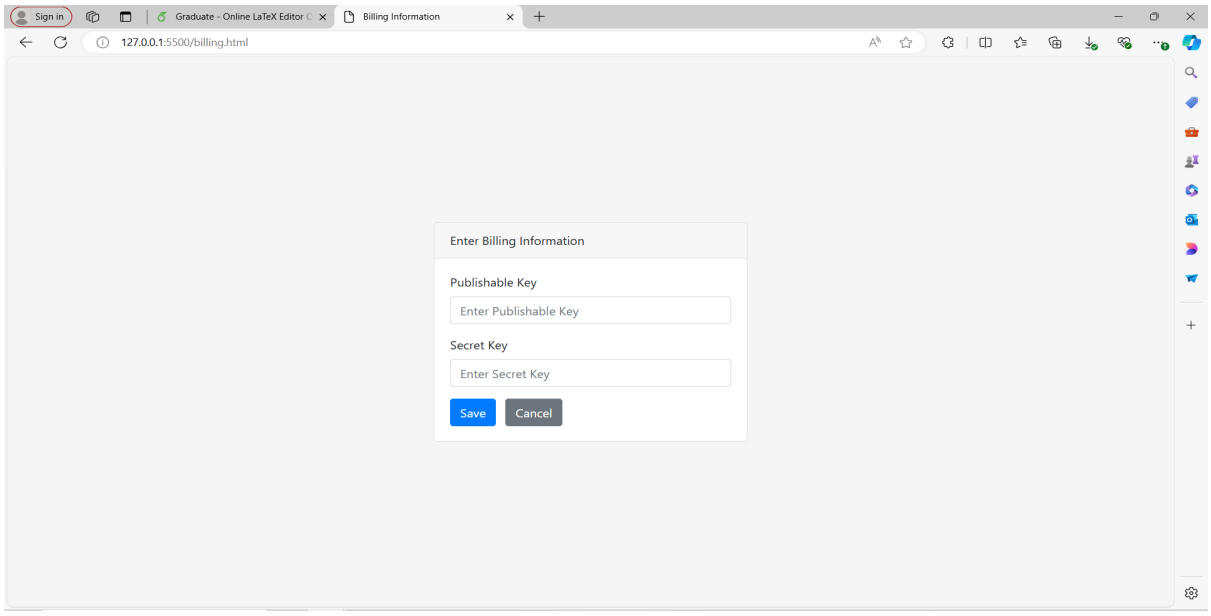


Figure 49: Billing Information

6.6.15 Admin account

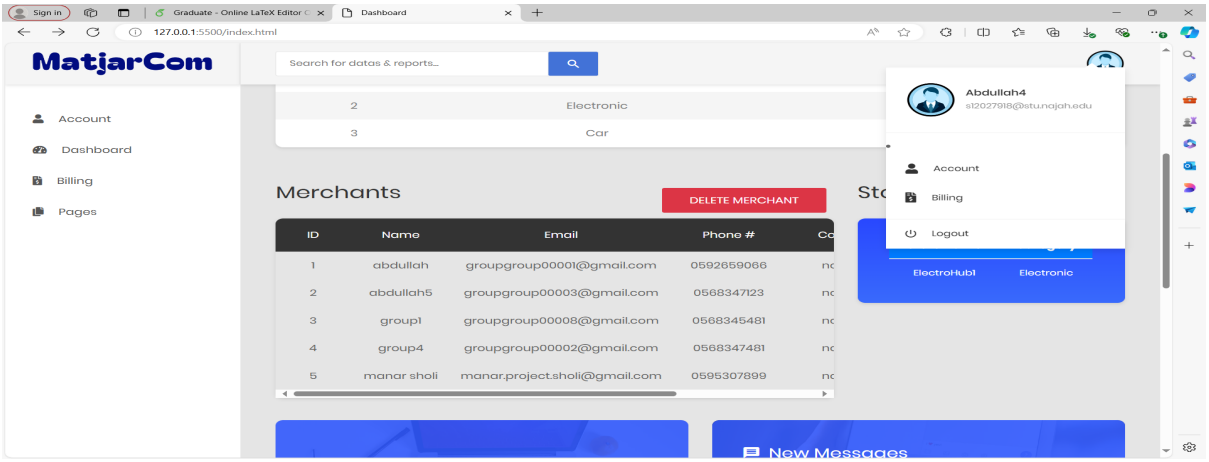


Figure 50: Admin Account

7 Result & Discussion

MatjarCom has emerged as a robust and versatile multi-vendor platform, demonstrating significant potential in streamlining and enhancing the e-commerce experience for both merchants and customers. The application effectively addresses the complexities involved in managing and operating online stores while providing an intuitive and user-friendly interface for end-users.

8 Conclusion

MatjarCom successfully bridges the gap between merchants and customers by providing a comprehensive multi-vendor platform that is both powerful and accessible. The app's well-thought-out design and functionality cater to the diverse needs of online store management and customer interaction. Merchants benefit from a wide range of customization and management tools, while customers enjoy a seamless and enriched shopping experience.

The backend's focus on security, scalability, and performance, combined with a dynamic and responsive Flutter-based frontend, underscores the app's capability to handle the demands of modern e-commerce. MatjarCom sets a high standard for multi-vendor applications by integrating advanced features, offering a bilingual interface, and ensuring secure and efficient operations.

In conclusion, MatjarCom not only meets but exceeds the expectations of a multi-vendor e-commerce solution, paving the way for future enhancements and broader adoption in the digital marketplace. With continuous updates and improvements, it is poised to become a leading platform in the multi-vendor e-commerce sector.

References

[1] NodeJS Course. (n.d.). Retrieved from https://www.youtube.com/watch?v=Bzzp0q0T5oMlist=PLQtNtS-WfRa8OF9juY3k6WUWayMfDKHK2

[2] Flutter Course 1. (n.d.). Retrieved from https://www.udemy.com/course/complete-flutter-arabic/?couponCode=LETSLEARNNOW

[3] Flutter Course 2. (n.d.). Retrieved from https://www.youtube.com/@Code2Start

[4] Stackoverflow. (n.d.). Retrieved from https://stackoverflow.com/.

[5] Flutter. (n.d.). Retrieved from https://flutter.dev/

[6] Node.js. (n.d.). Retrieved from https://nodejs.org/

[7] MongoDB. (n.d.). Retrieved from https://www.mongodb.com/

[8] JSON Web Token. (n.d.). Retrieved from https://jwt.io/

[9] bcrypt. (n.d.). Retrieved from https://www.npmjs.com/package/bcrypt

[10] Cloudinary. (n.d.). Retrieved from https://cloudinary.com/

[11] CORS (Cross-Origin Resource Sharing). (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[12] dotenv. (n.d.). Retrieved from https://www.npmjs.com/package/dotenv

[13] multer. (n.d.). Retrieved from https://www.npmjs.com/package/multer

[14] Nodemailer. (n.d.). Retrieved from https://nodemailer.com/

[15] Nodemon. (n.d.). Retrieved from https://nodemon.io/

[16] Express Rate Limit. (n.d.). Retrieved from https://www.npmjs.com/package/express-rate-limit

[17] Lahza API. (n.d.). Retrieved from https://lahza.com/

[18] ZeroBounce. (n.d.). Retrieved from https://www.zerobounce.net/

[19] Render. (n.d.). Retrieved from https://render.com/

[20] flutter_localizations. (n.d.). Retrieved from https://pub.dev/packages/flutter$_{l}ocalizations$

[21] cupertino_icons. (n.d.). Retrieved from https://pub.dev/packages/cupertino$_{i}cons$

[22] http. (n.d.). Retrieved from https://pub.dev/packages/http

[23] animated_background. (n.d.). Retrieved from https://pub.dev/packages/animated$_{b}ackground$

[24] image_picker. (n.d.). Retrieved from https://pub.dev/packages/image$_{p}icker$

[25] dio. (n.d.). Retrieved from https://pub.dev/packages/dio

[26] flutter_staggered_grid_view. (n.d.). Retrieved from https://pub.dev/packages/flutter$_{s}taggered_{g}rid_{v}iew$

[27] dotted_border. (n.d.). Retrieved from https://pub.dev/packages/dotted$_{b}order$

[28] social_media_flutter. (n.d.). Retrieved from https://pub.dev/packages/social$_{m}edia_{f}lutter$

[29] font_awesome_flutter. (n.d.). Retrieved from https://pub.dev/packages/font$_{a}wesome_{f}lutter$

[30] google_fonts. (n.d.). Retrieved from https://pub.dev/packages/google$_{f}onts$

[31] url_launcher. (n.d.). Retrieved from https://pub.dev/packages/url$_{l}auncher$

[32] carousel_slider. (n.d.). Retrieved from https://pub.dev/packages/carousel$_{s}lider$

[33] cached_network_image. (n.d.). Retrieved from https://pub.dev/packages/cached$_{n}etwork_{i}mage$

[34] simple_circular_progress_bar. (n.d.). Retrieved from https://pub.dev/packages/simple$_{c}ircular_{p}rogress_{b}ar$

[35] favorite_button. (n.d.). Retrieved from https://pub.dev/packages/favorite$_{b}utton$

[36] file_picker. (n.d.). Retrieved from https://pub.dev/packages/file$_p$*icker*

[37] like_button. (n.d.). Retrieved from https://pub.dev/packages/like$_b$*utton*

[38] image_preview. (n.d.). Retrieved from https://pub.dev/packages/image$_p$*review*

[39] flutter_expandable_text. (n.d.). Retrieved from https://pub.dev/packages/flutter$_e$*xpandable*$_t$*ext*

[40] get. (n.d.). Retrieved from https://pub.dev/packages/get

[41] fl_chart. (n.d.). Retrieved from https://pub.dev/packages/fl$_c$*hart*

[42] onesignal_flutter. (n.d.). Retrieved from https://pub.dev/packages/onesignal$_f$*lutter*

[43] firebase_core. (n.d.). Retrieved from https://pub.dev/packages/firebase$_c$*ore*

[44] permission_handler. (n.d.). Retrieved from https://pub.dev/packages/permission$_h$*andler*

[45] cloud_firestore. (n.d.). Retrieved from https://pub.dev/packages/cloud$_f$*irestore*

[46] quickalert. (n.d.). Retrieved from https://pub.dev/packages/quickalert

[47] smooth_page_indicator. (n.d.). Retrieved from https://pub.dev/packages/smooth$_p$*age*$_i$*ndicator*

[48] share_plus. (n.d.). Retrieved from https://pub.dev/packages/share$_p$*lus*

[49] flutter_colorpicker. (n.d.). Retrieved from https://pub.dev/packages/flutter$_c$*olorpicker*