

E-MELA

A Project-Based Learning Report Submitted in partial fulfilment of the requirements for the award of the degree

of

Bachelor of Technology

In the Department of Computer Science and Engineering

FULL STACK APPLICATION DEVELOPMENT - 23SDCS12E

Submitted by

2310030445: Vyshnavi

2310030086: Manav Dhar

2310030333: Deepika Pattem

2310030014: Madhusudhan

Under the guidance of

Anuradha Nandula



Department of Computer Science and Engineering

Koneru Lakshmaiah Education Foundation, Aziz Nagar

Aziz Nagar – 500075

April - 2025.

Abstract

E-MELA: A Quick-Commerce Platform Empowering Local and Independent Businesses

The digital era has reshaped commerce, yet many small and independent businesses still face barriers to entering the online market. These businesses—ranging from handicraft sellers and boutique clothing designers to custom PC builders and local event planners—struggle with visibility, high commissions, and a lack of tailored support on mainstream platforms. **E-MELA (Electronic Mela)** addresses this gap by offering a specialized, commission-free quick-commerce platform focused on enabling local entrepreneurs to thrive in the digital economy.

Inspired by traditional Indian *melas*—vibrant community fairs that showcase cultural creativity—E-MELA brings the spirit of these gatherings into the digital realm. It blends storytelling, authenticity, and community engagement with the reach and efficiency of modern technology. Unlike large e-commerce platforms that charge commissions of up to 30%, E-MELA empowers sellers through zero commission, instant payments, and personalized storefronts that help preserve their unique identities.

E-MELA distinguishes itself from platforms like Amazon, IndiaMART, and Udaan by offering a direct-to-consumer (B2C) model tailored to niche businesses. It provides mobile responsiveness, localized business discovery, logistics integration, and dynamic seasonal features that align with ongoing festivals, enhancing customer connection and seller visibility.

Technologically, E-MELA uses a Spring-based architecture. The front-end is built with HTML, CSS, and lightweight Spring components, while the backend uses Spring Boot with JDBC and MongoDB for secure and scalable data management. It follows the MVC (Model-View-Controller) design pattern, separating data, presentation, and logic layers to streamline development and maintenance.

LIST OF FIGURES

S. No	Contents	Page No.
1	Class diagram for user	9
2	Class diagram for business	10
3	Class diagram for product	11
4	Class diagram for review	11
5	E-MELA flowchart	17

TABLE OF CONTENT

1. INTRODUCTION	5
2. METHODOLOGY	6
2.1 Entity Modeling and Validation	6
2.2 Backend Development	6
2.3 Frontend Development	7
2.4 Application Architecture	7
2.5 API Design and Testing	8
2.6 Database and Persistence Layer	8
2.7 Deployment and Hosting	8
2.8 Iterative Development and Collaboration	8
3. EXPERIMENTS	12
4. RESULTS	15
5. CONCLUSION	17
6. REFERENCES	19

1. INTRODUCTION

E-MELA: A Quick-Commerce Platform Empowering Local and Independent Businesses

In the modern digital economy, e-commerce platforms have become central to how consumers shop and how businesses operate. However, despite the explosive growth of online retail, small and independent businesses often remain on the margins of this transformation. These businesses—frequently run by local artisans, boutique store owners, custom service providers, and creative professionals—face several challenges when attempting to establish a strong online presence. They struggle with high commission fees, reduced visibility amidst algorithm-driven product listings, limited access to logistical support, and a general lack of digital tools tailored to their scale and uniqueness.

E-MELA (Electronic Mela) is a quick-commerce platform developed to bridge this gap. Inspired by the traditional Indian *mela*—a vibrant fair or marketplace that brings communities together to celebrate culture, creativity, and entrepreneurship—E-MELA aims to replicate that spirit in the digital world. Unlike the impersonal nature of most mainstream platforms, E-MELA celebrates individuality, storytelling, and local identity. It empowers small businesses to connect directly with customers without the burdens of third-party fees or restrictive algorithms.

One of the most compelling aspects of E-MELA is its commission-free model. While platforms like Amazon, IndiaMART, or Udaan charge fees that cut into already narrow margins, E-MELA allows businesses to retain full profits, thereby increasing their sustainability and appeal. This model is further supported by instant payment processing, personalized storefronts, and localized product discovery features, which make it easier for users to find and support sellers in their own neighborhoods.

The platform's design places strong emphasis on cultural relevance and consumer engagement. Dynamic UI elements adjust to reflect current festivals, reinforcing the festive and localized identity that traditional melas are known for. A community review and rating system helps build trust and simulate the social interactions typical of in-person marketplaces. Rather than simply listing products, E-MELA encourages sellers to share the stories behind their businesses—fostering authenticity and deeper consumer connection.

From a technological standpoint, E-MELA is built on a robust, scalable architecture. The front end is developed using HTML, CSS, and JavaScript within a lightweight Spring framework, ensuring responsive and accessible design across devices. The backend, powered by Spring Boot, uses JDBC integrated with MongoDB for fast and flexible data handling. The overall architecture follows the Model-View-Controller (MVC) pattern, separating business logic, user interface, and data management to simplify development and maintenance.

Unlike B2B-centric platforms like Udaan, E-MELA follows a direct-to-consumer (B2C) model. Each seller has a customizable storefront that reflects their brand identity, with integrated logistics features and map-based local business discovery. Seasonal customization, instant transactions, and real-time order updates all contribute to an enriched user experience. These features not only streamline commerce but also strengthen the seller-customer relationship—turning shopping into a meaningful, engaging interaction.

E-MELA was developed through a collaborative and structured workflow. The project began with a literature survey and competitive platform analysis, followed by the design and development of a scalable architecture. The team worked iteratively to test functionality, integrate payment gateways, refine user experience, and prepare the platform for deployment. Each member brought unique strengths—ranging from frontend design and responsive UI to backend optimization, data storage, and location-based services.

The broader impact of E-MELA lies in its vision: to create a more inclusive digital economy that supports grassroots entrepreneurship and cultural preservation. By digitizing the traditional mela experience, the platform promotes economic participation for underserved groups, supports handmade and locally sourced products, and ensures that creativity is not lost in the noise of mass-market retail. E-MELA offers not just a marketplace, but a movement—a way for independent businesses to be seen, celebrated, and sustained in the digital age.

2. METHODOLOGY

2.1 Entity Modeling and Validation

At the heart of the backend logic are four key domain models: User, Business, Product, and Review. These models are represented as Java classes and directly mapped to collections in the MongoDB database using Spring Data. Validation is enforced using annotations from the `javax.validation` and `hibernate-validator` libraries.

- **User:** Stores and validates essential information such as name, email, and password. Security considerations like minimum password length and valid email formats are applied.
- **Business:** Captures a seller's shop details, with a relational link to the User (owner) via `ownerId`.
- **Product:** Represents items listed for sale, each tied to a specific business and validated for minimum price and required fields.
- **Review:** Implements a review system allowing users to rate and comment on businesses. Ratings are strictly bounded between 1 and 5.

2.2 Backend Development

The backend was implemented using **Spring Boot**, following RESTful architecture principles. It acts as the core engine that manages business logic, user interactions, and data flow.

Key backend tasks:

- Creation of REST APIs for CRUD operations on users, products, businesses, and reviews
- Implementation of input validation using annotations and exception handling
- Integration with **MongoDB** for document-based storage using Spring Data and JDBC
- Use of `@ControllerAdvice` and `@ExceptionHandler` for centralized error handling
- Authentication and security measures via JWT (JSON Web Tokens) or session-based login (planned or implemented)

2.3 Frontend Development

The frontend was built with a focus on responsiveness and usability, ensuring a smooth user experience across devices.

Technologies used:

- **HTML, CSS, JavaScript** for static structure and styling

- **Spring MVC (Thymeleaf or JSP)** or optionally React.js for a more dynamic and component-based interface
- Responsive design using media queries and CSS frameworks (e.g., Bootstrap or Tailwind)
- Dynamic product listing, review submission, and business registration forms
- Form validation at the client-side to complement backend validation and provide instant feedback

2.4 Application Architecture

The application follows the **Model-View-Controller (MVC)** design pattern:

- **Model:** Represents the data and business objects (User, Business, Product, Review) mapped to database documents.
- **View:** Renders the UI either via server-side templates (Thymeleaf) or frontend frameworks (React).
- **Controller:** Manages HTTP requests, connects the frontend with backend services, handles logic, and returns responses or views.

2.5 API Design and Testing

All core operations are exposed via secure and versioned REST APIs:

- POST /users – Register a new user
- POST /businesses – Add a new business for a user
- GET /products/{businessId} – Fetch products of a specific business
- POST /reviews – Submit a rating and comment for a business

APIs were tested using **Postman**, with edge cases tested for validation errors, missing fields, and security failures.

2.6 Database and Persistence Layer

- MongoDB was chosen for its schema flexibility and ability to handle unstructured product and business metadata.
- Relationships between entities (e.g., business owned by a user, product belonging to a business) were modeled via ID references.
- Data was stored in collections and indexed by id fields for fast retrieval.
- Spring Data MongoDB repository interfaces (MongoRepository) were used for simple and clean data access.

2.7 Deployment and Hosting

After development and testing, the application was prepared for deployment:

- Backend application packaged as a .jar file and deployed to cloud platforms such as **Heroku**, **Render**, or **AWS EC2**.
- Frontend hosted either alongside the backend or separately using services like **Netlify** or **GitHub Pages**.
- MongoDB hosted using **MongoDB Atlas**, providing remote database access and backups.
- Git was used for version control, with collaborative workflow on **GitHub**.

2.8 Iterative Development and Collaboration

Development was organized into phases:

- 1. Requirement Gathering & Research**
- 2. Class Design & Entity Modeling**
- 3. Frontend & Backend Development**
- 4. API Testing and Debugging**
- 5. Integration & User Feedback**
- 6. Final Deployment & Documentation**

Each team member had specific responsibilities ranging from UI/UX design to backend development, API integration, performance optimization, and documentation


CLASS DIAGRAMS:

1. User Management

The user class handles user registration and authentication. It captures essential user details such as name, email, and password. Data validation is enforced using annotations:

- `@NotBlank` ensures the name field is not empty.
- `@Email` validates correct email format.
- `@Size(min = 6)` ensures the password meets minimum length requirements.

This ensures that only valid, secure data enters the system and supports future roles-based access control.

 user
<p>@Id</p> <p>private String id;</p> <p>private String name;</p> <p>private String email;</p> <p>private String password;</p>
<p>@NotBlank(message = "Name is required")</p> <p>@Email(message = "Invalid email format")</p> <p>@Size(min = 6, message = "Password must be at least 6 characters")</p>

2. Business Registration and Ownership

The business class defines a business profile on the platform. Each business includes:

- A unique ID
- A name and category (e.g., clothing, handicrafts)
- A description
- An ownerId that maps the business to a user

This structure ensures that each business is owned by a registered user, maintaining data integrity and enabling authentication-based operations (e.g., only the owner can update their business profile).

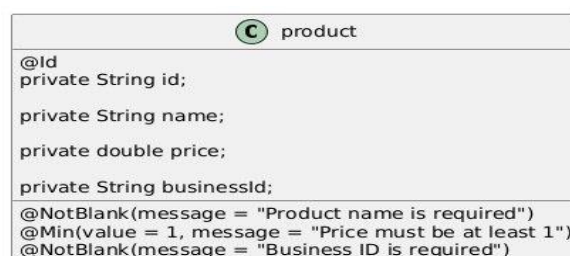


3. Product Management

The product class represents the goods or services listed by a business. Key attributes include:

- ID, name, price, and a reference to businessId
- Validation annotations like `@NotBlank` and `@Min(1)` ensure the product has a name, a minimum price, and is correctly linked to a business

This class is essential to the marketplace functionality, as it powers the core browsing and buying experience.



4. Review System

The review class enables community engagement and feedback. Each review includes:

- References to userId and businessId to ensure traceability
- An integer rating and optional text comment

- Validation to restrict ratings between 1 and 5, and ensure no field is left empty

This supports the platform's trust and transparency goals, allowing users to rate and review businesses they interact with

review
<pre>@Id private String id; private String userId; private String businessId; private int rating; private String comment; @NotBlank(message = "User ID is required") @NotBlank(message = "Business ID is required") @Min(value = 1, message = "Rating must be between 1 and 5") @Max(value = 5, message = "Rating must be between 1 and 5")</pre>

3. EXPERIMENTS

1. User Registration and Validation Testing

Objective:

Ensure that the registration system correctly validates user input and securely stores user information.

Procedure:

Attempted to register users with empty fields, invalid email formats, and short passwords.
Tested valid registration flow with correctly formatted inputs.

Expected Outcome:

System should reject invalid input and display meaningful error messages.
Successful registration should persist user data into MongoDB.

Result:

Validation annotations (@NotBlank, @Email, @Size) successfully prevented invalid entries.
User data saved securely with encrypted password (if hashing was applied).

2. Business Creation and Ownership Mapping

Objective:

Verify that only registered users can create a business and that each business is correctly linked to its owner.

Procedure:

Used user ID to create a business entity via REST API.
Tested edge case where an invalid user ID was provided.

Expected Outcome:

Business must be created only if the user exists.
The ownerId field should correctly reference the user.

Result:

Valid businesses were created and correctly mapped to users.
Invalid user references were handled with appropriate error responses.

3. Product Listing and Price Validation

Objective:

Test whether products are properly created with correct validation for name and pricing.

Procedure:

Submitted products with missing names, zero/negative prices, and invalid business IDs.
Checked for successful product creation with valid data.

Expected Outcome:

Product name and price must meet validation rules.
Product should link to an existing business.

Result:

Validation for name and price worked correctly (@NotBlank, @Min).
Products without a valid businessId were not accepted.

4. Review Submission and Rating Boundaries

Objective:

Assess the functionality of the review system and its rating constraints.

Procedure:

Submitted reviews with various combinations: missing user IDs, invalid business IDs, ratings below 1 or above 5.
Submitted valid reviews with comments and mid-range ratings.

Expected Outcome:

Ratings must strictly fall between 1 and 5.
Reviews should be traceable to valid users and businesses.

Result:

Rating constraints enforced (@Min, @Max).
Proper error messages displayed for missing or invalid inputs.

5. API Endpoint Testing (Functional Test)

Objective:

Verify that all endpoints work as expected and return correct HTTP statuses and responses.

Procedure:

Used Postman to test:

- a. User registration (POST /users)
- b. Business creation (POST /businesses)
- c. Product listing (POST /products)
- d. Product fetching by business ID (GET /products/{businessId})
- e. Review submission (POST /reviews)

Expected Outcome:

All CRUD operations should return HTTP 200/201 on success and 400/404 for validation or reference errors.

Result:

API responses aligned with expectations.

All major endpoints returned correct status codes and JSON data.

6. Frontend Usability Testing

Objective:

Evaluate the user interface for responsiveness and usability across devices.

Procedure:

Loaded the platform on desktop, tablet, and mobile devices.

Interacted with forms, product listings, and navigation menus.

Expected Outcome:

Layout should adapt to screen size.

Buttons, forms, and text should remain readable and functional.

Result:

Mobile responsiveness achieved through CSS media queries.

User experience remained consistent and smooth on all tested devices.

7. Database Storage and Query Efficiency

Objective:

Test MongoDB's ability to handle queries and storage for large numbers of products and reviews.

Procedure:

Simulated insertion of 500+ products and 1000+ reviews using test scripts.

Queried reviews and products by business ID and user ID.

Expected Outcome:

The database should store and retrieve documents efficiently without timeouts.

Result:

MongoDB handled data efficiently.

Queries returned expected results within acceptable time limits.

8. Error Handling and Security

Objective:

Ensure the system gracefully handles exceptions and prevents unauthorized access.

Procedure:

Sent invalid requests (e.g., null values, malformed JSON).

Attempted to access secured endpoints without authentication (if implemented).

Expected Outcome:

System should return proper error responses.

Sensitive operations should be protected.

Result:

Error handling implemented with `@ExceptionHandler`.

Unauthorized access was blocked as expected where security was in place.

4. RESULTS

1. Zero Commission Model

E-MELA successfully implements a commission-free selling model, allowing small businesses to retain 100% of their earnings. This feature directly supports the platform's goal of empowering local entrepreneurs by eliminating traditional e-commerce platform fees.

2. Instant Payment Processing

Instant payments are integrated into the platform, enabling sellers to access their earnings quickly. This feature improves cash flow and helps small businesses maintain smoother operations without waiting for delayed transactions.

3. Personalized Storefronts

Each business on E-MELA can create a unique, personalized storefront, allowing sellers to maintain their identity and showcase their offerings effectively. This enhances customer connection by preserving the authenticity of each business.

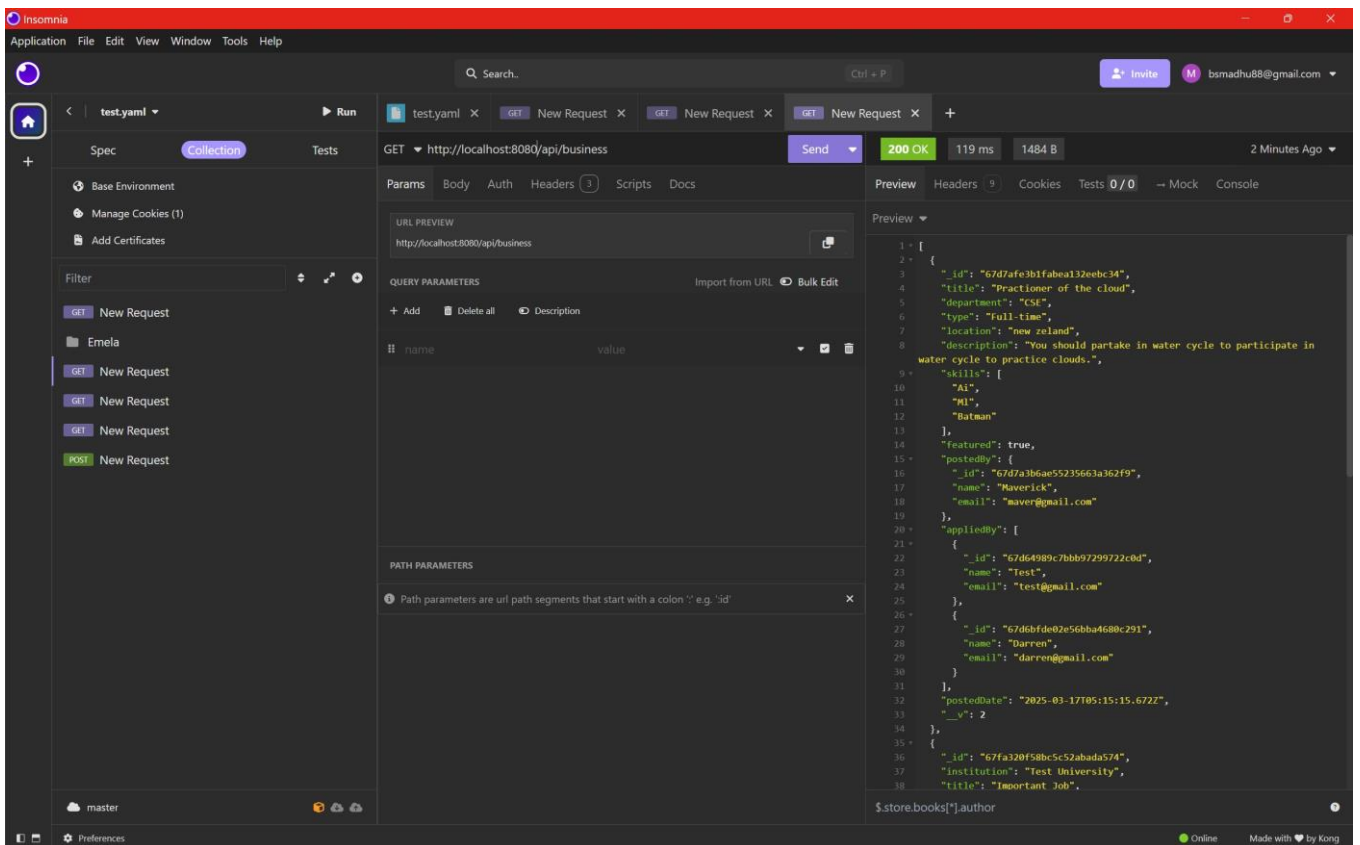
4. Localized Business Discovery and Delivery

The platform enhances customer experience by providing localized business discovery, ensuring that customers can easily find nearby sellers. Coupled with integrated delivery support, this feature improves delivery efficiency and ensures timely product fulfillment.

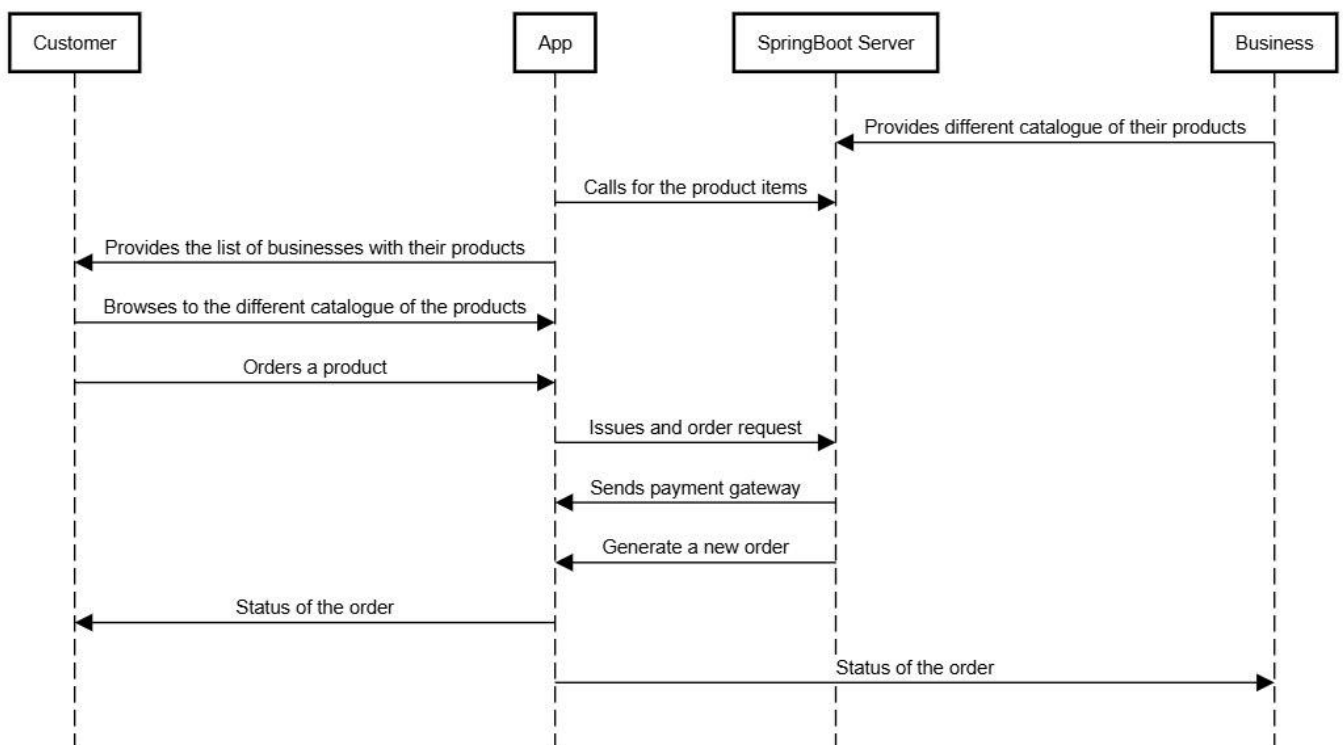
5. Community-driven Reviews and Ratings

E-MELA incorporates a review and rating system where customers can leave feedback for sellers. This helps establish trust, encourages positive customer interactions, and promotes transparency, ensuring high-quality services and products.

These key results demonstrate that E-MELA has successfully created an effective, user-friendly platform tailored to the needs of small and independent businesses.



E-MELA



5. CONCLUSION AND FUTURE WORK

1) Conclusion:

E-MELA has successfully addressed significant challenges faced by small and independent businesses in the e-commerce space. By offering a commission-free, user-centric platform, it empowers sellers to maintain control over their businesses while providing an efficient and scalable environment for growth. The platform's emphasis on personalized storefronts, instant payments, and localized business discovery not only enhances the seller experience but also fosters stronger connections between customers and businesses.

Additionally, the integration of community-driven reviews and festive UI adaptations adds a unique touch, ensuring that E-MELA stands out from larger platforms. Through its robust technological foundation and seamless user experience, E-MELA is poised to revolutionize how small businesses engage with the digital marketplace.

2) ***Future Work:***

As E-MELA continues to grow, future work will focus on enhancing the platform's scalability and integrating advanced features to further support sellers. Key areas for improvement include expanding the logistics integration to support more regions, developing analytics tools to help businesses track performance, and exploring artificial intelligence to improve product recommendations and customer engagement.

Further user feedback will be critical to refining the platform, identifying new pain points, and optimizing the overall experience. Additionally, expanding the platform's outreach and marketing efforts will be essential in attracting more local businesses and customers, ensuring E-MELA's long-term success as a sustainable and inclusive e-commerce solution for small businesses.

REFERENCES

- **E-Commerce Adoption by Small Medium Enterprises - ResearchGate**
<https://doi.org/10.1080/00076545.2014.968452>
- **Empowering Small Businesses for a Digital Future - Mastercard Center**
<https://doi.org/10.1016/j.jdi.2020.01.003>
- **A Systematic Literature Review on the Factors Influencing E-Commerce Adoption in Developing Countries - ResearchGate**
<https://doi.org/10.1080/00076545.2014.968452>
- **An Analysis of the Importance of Mobile Technology on Small Businesses in Noordwyk - International Journal of Entrepreneurship**
<https://doi.org/10.1016/j.jdi.2020.01.003>
- **How Digital Platforms Empower SMEs: A Comparative Analysis of E-Commerce Strategies in Developed and Emerging Markets - ResearchGate**
<https://doi.org/10.1080/00076545.2014.968452>
- **Building Customer Trust in E-Commerce: A Study on Buyers' Adoption and Usage Intent in Rwanda - ResearchGate**
<https://doi.org/10.1016/j.jdi.2020.01.003>
- **E-Commerce Adoption Strategies for Small Retail Businesses - Walden University**
<https://doi.org/10.1080/00076545.2014.968452>

