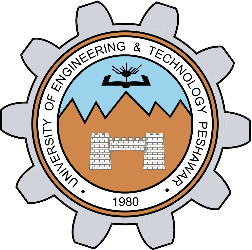
**Project Report**

****

**SPRING 2025**

**Database Management Systems**

Submitted by: **Zarmina Miftah Durrani**

# Noor ul Huda Muhammad Shahbaz Khan

Registration No. : **22PWCSE2126**

# 22PWCSE2205

**22PWCSE2211**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature:

# Submitted to:

Engr.Sumayyae Salahuddin

# Submission Date:

20th June, 2025

Department of Computer Systems Engineering University of Engineering and Technology, Peshawar

## ****TABLE OF CONTENTS****

1. **PROJECT OVERVIEW**   
   1.1 Core Objectives  
   1.2 Project Scope
   1. **TECHNICAL ARCHITECTURE**

1.3.1 Technology Stack  
 1.3.2 System Architecture

1. **DATABASE DESIGN & ANALYSIS**  
     2.1 Database Design Philosophy  
     2.2 Entity Analysis  
     3.3 Advanced Business Rules Implementation
2. **ENTITY-RELATIONSHIP DIAGRAM**
3. **RELATIONAL SCHEMA**
4. **SQL QUERIES**
5. **DATABASE TABLES**
6. **LARAVEL IMPLEMENTATION DETAILS**  
   7.1 Laravel MVC Implementation  
   7.1.1 Model

7.1.2 View

7.1.3 Controller

1. **SYSTEM FUNCTIONALITY**  
   8.1 Role-Based Access Control   
    8.1.1 Admin Functions  
    8.1.2 User Functions
2. **SECURITY IMPLEMENTATION**
3. **FUTURE ENHANCEMENTS**  
   10.1 Scalability Improvements  
   10.2 Feature Expansions
4. **CONCLUSION**   
   11.1 The Project Successfully Demonstrates
5. **REFERENCES**
6. **DEPLOYMENT LINK**
7. **GITHUB LINK**
8. **PROJECT OVERVIEW:**

The **Courier Management System** is a full-stack web application built using the **Laravel 12 framework** with a **MySQL database**. It is designed to streamline the operations of a courier service by digitizing parcel tracking, user management, status updates, and administrative control. The system supports multiple roles, including administrators and general users, and provides a secure interface to manage and monitor courier shipments.

This project demonstrates a practical implementation of database design principles, Laravel’s MVC structure, and real-time courier tracking functionalities. It integrates session management, input validation, and CRUD operations, offering a scalable and responsive solution for courier businesses.

**1.1 CORE OBJECTIVES:**

* **Primary Objective:**To develop a web-based courier management system that allows users to send, track, and manage parcels securely and efficiently.
* **Secondary Objectives:**
* Implement role-based access control for administrators and users.
* Enable real-time parcel tracking and status management.
* Provide interfaces for managing couriers, users, and contact messages.
* Ensure secure authentication and session handling using Laravel features.
* Deploy the project on a live hosting platform (Infinity Free) for accessibility.
* Apply database normalization and maintain data integrity through foreign key constraints.

## ****1.2 PROJECT SCOPE****

The scope of the Courier Management System includes:

* **Admin Module:**
  + Manage users and courier entries.
  + Update shipment status (Received, In Transit, Delivered, etc.).
  + View and respond to contact messages.
* **User Module:**
  + Register and log in securely.
  + Submit courier details for shipment.
  + Track parcel status using unique tracking numbers.
* **Database Management:**
  + Normalized relational schema in 3NF.
  + Implementation of cascading delete/update rules and validations.
* **Deployment:**
  + Fully functional deployment using **InfinityFree** for frontend/backend and **phpMyAdmin** for online database connectivity.

## ****1.3 TECHNICAL ARCHITECTURE****

This section outlines the technologies used in the development, the system architecture design, and the layered structure based on the MVC (Model-View-Controller) paradigm followed by Laravel.

**1.3.1 TECHNOLOGY STACK**

| **Layer** | **Technology Used** |
| --- | --- |
| **Frontend** | HTML5, CSS3, Bootstrap 5, Blade Templating Engine |
| **Backend** | PHP 8.3 with Laravel 12 Framework |
| **Database** | MySQL (hosted via InfinityFree using phpMyAdmin) |
| **Development** | XAMPP (Apache, MySQL, PHP) local environment |
| **Version Control** | Git & GitHub |
| **Deployment** | InfinityFree (Free Web Hosting Provider) |
| **Tools Used** | Composer, Laravel Artisan CLI, VS Code |

### 1.3.2 SYSTEM ****ARCHITECTURE****

The application is built using the Model-View-Controller (MVC) architectural pattern, ensuring a clear separation of concerns and organized code structure.

**Model Handles:**

* Storing user and admin account information in the database
* Saving courier shipment records and tracking details
* Managing contact messages and customer inquiries
* Enforcing rules such as “only admin can update courier status”
* Handling relationships between users, couriers, and statuses

**View Handles:**

* Displaying web pages like the dashboard, courier status, and contact forms
* Providing interfaces for login, registration, and profile views
* Showing tracking details and courier history to users
* Displaying admin panels for courier management
* Ensuring responsive and clean layout using Blade, Bootstrap, and CSS

**Controller Handles:**

* Processing user and admin requests like login, logout, and form submissions
* Handling courier creation, updates, deletions, and status tracking
* Validating form data and applying business rules
* Retrieving data from the database and passing it to views
* Controlling page flow based on user type (admin vs. regular user)

### ****2. DATABASE DESIGN & ANALYSIS****

#### ****2.1 DATABASE DESIGN PHILOSOPHY****

Our database design adheres to the following principles:

* **Normalization:** All tables are normalized to 3NF to eliminate data redundancy and ensure logical storage
* **Referential Integrity:** Foreign key relationships ensure consistency between related records
* **Scalability:** The schema is structured to support future expansion such as multi-branch courier handling
* **Security:** All queries use prepared statements to prevent SQL injection
* **Performance:** Indexes are applied to frequently queried columns like tracking ID and user email for faster retrieval

#### ****2.2 ENTITY ANALYSIS****

| **ENTITY** | **PRIMARY KEY** | **KEY ATTRIBUTES** | **DESCRIPTION** |
| --- | --- | --- | --- |
| **User** | u\_id | name, email, password | Stores login and identity information of registered users |
| **Admin** | a\_id | email, password | Manages backend operations including courier oversight |
| **Courier** | c\_id | senderName, receiverName, source, destination, status, dateBooked | Stores detailed courier shipment information |
| **Status** | s\_id | courier\_id (FK), current\_status, last\_updated | Tracks the status updates of each courier parcel |
| **Contact** | id | name, email, subject, message, created\_at | Stores user-submitted contact or inquiry forms |
| **ContactMessage** | id | name, email, message, created\_at | Captures messages submitted via frontend contact form |

#### ****2.3 ADVANCED BUSINESS RULES IMPLEMENTATION****

**I. USER REGISTRATION RULE**

* **Rule:** Only valid and unique email addresses can be used for registration.
* **Explanation:** One email can only be used for one account.
* **Example:** If "zubair @gmail.com" is already registered, another user can’t use it.

**II. LOGIN AUTHENTICATION RULE**

* **Rule:** Email and password must match a valid record in the users or admin table.
* **Explanation:** Unauthorized users cannot access system modules.
* **Example:** Wrong password leads to login failure.

**III. COURIER ENTRY RULE**

* **Rule:** Only admins can add, edit, or delete courier records.
* **Explanation:** Users can view tracking info but cannot manipulate courier data.
* **Example:** Only admin can change "In Transit" to "Delivered".

**IV. UNIQUE TRACKING RULE**

* **Rule:** Each courier must have a unique tracking ID (c\_id).
* **Explanation:** Prevents confusion when retrieving status.
* **Example:** No two parcels should share the same ID like #C1001.

**V. STATUS UPDATE RULE**

* **Rule:** Courier status can be: Received, In Transit, Delivered, Delayed, Cancelled
* **Explanation:** All shipments follow specific status paths.
* **Example:** A parcel moves from Received → In Transit → Delivered.

**VI. CONTACT FORM VALIDATION RULE**

* **Rule:** User messages must contain name, valid email, and message content.
* **Explanation:** Incomplete contact forms are rejected.
* **Example:** A blank message or missing email is not accepted.

**VII. AUTOMATIC TIMESTAMP RULE**

* **Rule:** Fields like created\_at and last\_updated are automatically managed by Laravel.
* **Explanation:** Helps admins know when couriers were created or modified.
* **Example:** A courier created today has its date auto-filled as today's date.

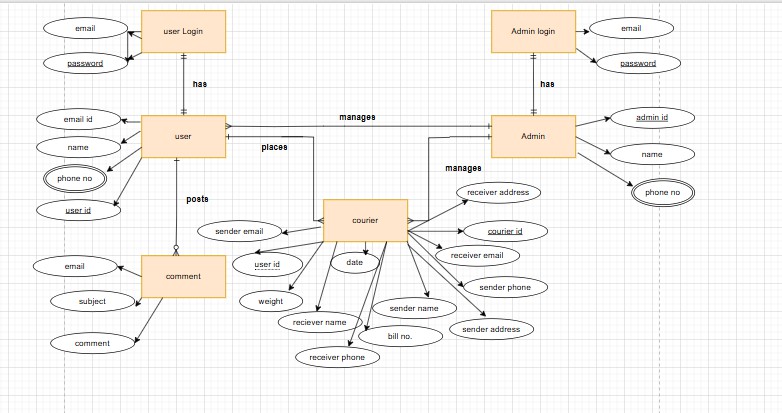
**VIII. CASCADING DELETION RULE**

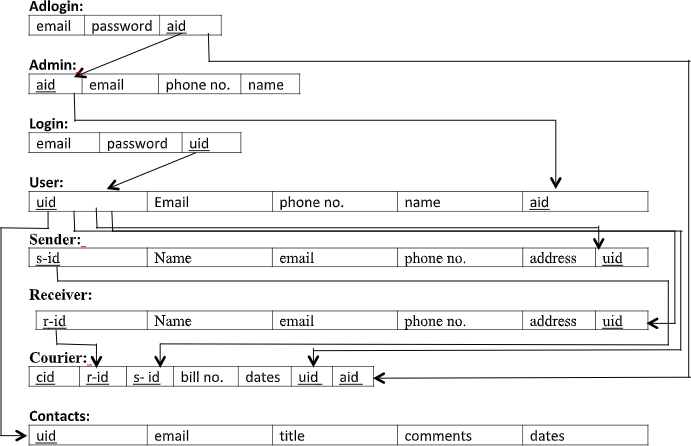
* **Rule:** Deleting a user deletes their associated contact messages; deleting a courier deletes its related status.
* **Explanation:** Ensures clean data without orphan records.
* **Example:** Delete courier #C1002 → all its status logs are also deleted.

**IX. ROLE RESTRICTION RULE**

* **Rule:** Only admins can access the backend dashboard; regular users are restricted.
* **Explanation:** Admin-level permissions are enforced strictly.
* **Example:** A logged-in user cannot view courier management page.

**X. ADMIN VALIDATION RULE**

* **Rule:** Admin login credentials must match exactly with the database records.
* **Explanation:** Prevents unauthorized backend access.
* **Example:** Wrong admin email or password blocks login attempt.
  1. **ER Diagram:**
  2. **Normalized Relational Schema:**

****

* 1. **SQL QUERIES:**

select \* from `users` where `email` = ? and `password` = ? limit 1 [[Maleeha@gmail.com,](mailto:Maleeha@gmail.com) 1234567] select \* from `users` where `u\_id` = ? limit 1 [9]

select \* from `users` where `u\_id` = ? limit 1 [9] select \* from `users` where `u\_id` = ? limit 1 [9]

select \* from `adlogin` where `email` = ? limit 1 [[admin1@gmail.com](mailto:admin1@gmail.com)] select \* from `users` []

select \* from `contacts` order by `created\_at` desc []

select \* from `contact\_messages` order by `created\_at` desc [] select \* from `users` []

select \* from `courier` where `courier`.`c\_id` = ? limit 1 [11] select \* from `users` where `users`.`u\_id` = ? limit 1 [9] select \* from `courier` where `u\_id` = ? [9]

select \* from `users` []

select count(\*) as aggregate from `users` where `email` = ? [[admin1@gmail.com](mailto:admin1@gmail.com)] select count(\*) as aggregate from `users` where `email` = ? [[jabbar@gmail.com](mailto:jabbar@gmail.com)]

insert into `users` (`name`, `email`, `password`) values (?, ?, ?) [jabbar, [jabbar@gmail.com,](mailto:jabbar@gmail.com) 12345678] select \* from `users` order by `u\_id` asc []

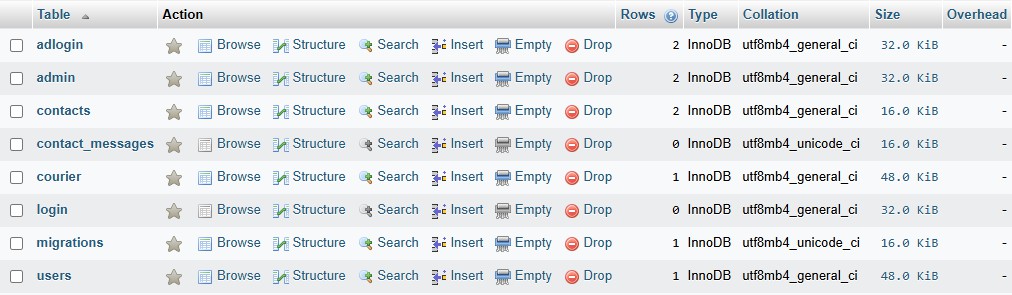
select \* from `users` where `u\_id` = ? limit 1 [10]

select count(\*) as aggregate from `users` where `email` = ? and `u\_id` <> ? [[jabbar@gmail.com,](mailto:jabbar@gmail.com) 10] update `users` set `name` = ?, `email` = ? where `u\_id` = ? [jabbar1, [jabbar@gmail.com,](mailto:jabbar@gmail.com) 10]

select \* from `users` order by `u\_id` asc [] delete from `users` where `u\_id` = ? [10] select \* from `users` order by `u\_id` asc []

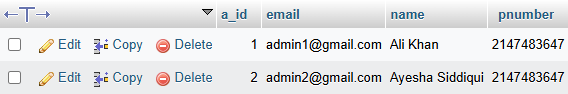
select \* from `contacts` order by `created\_at` desc []

select \* from `contact\_messages` order by `created\_at` desc []

* 1. **DATABASE TABLES:**

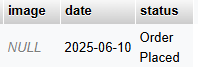
1. **Adlogin:**



1. **Admin:**
2. **Contacts:**

****

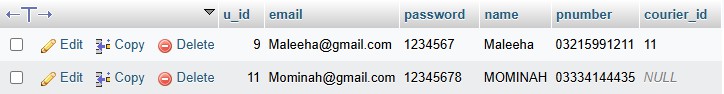
1. **Courier:**



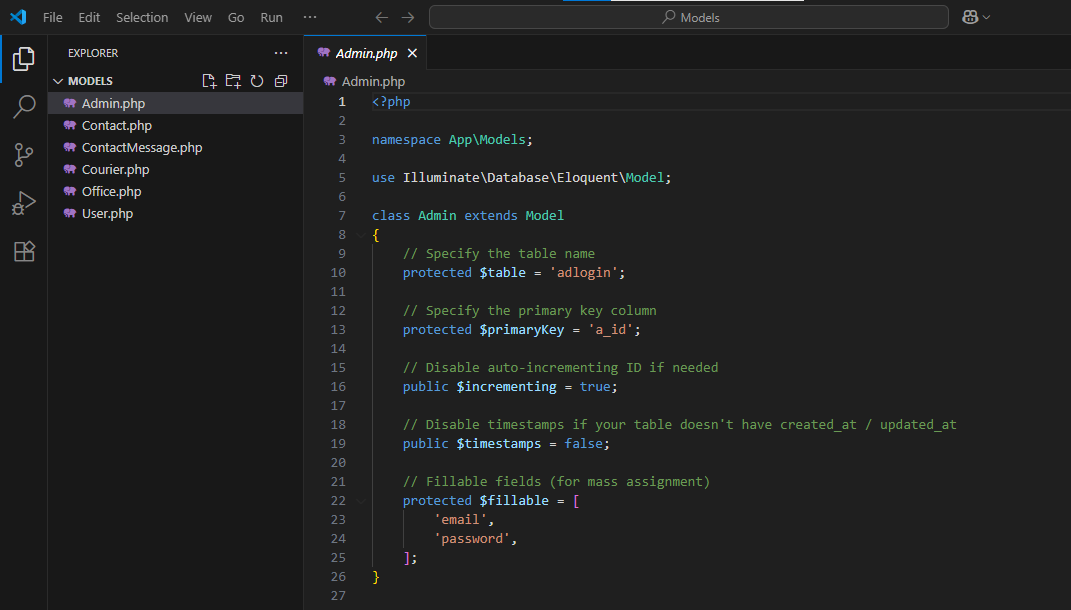
1. **Login:**

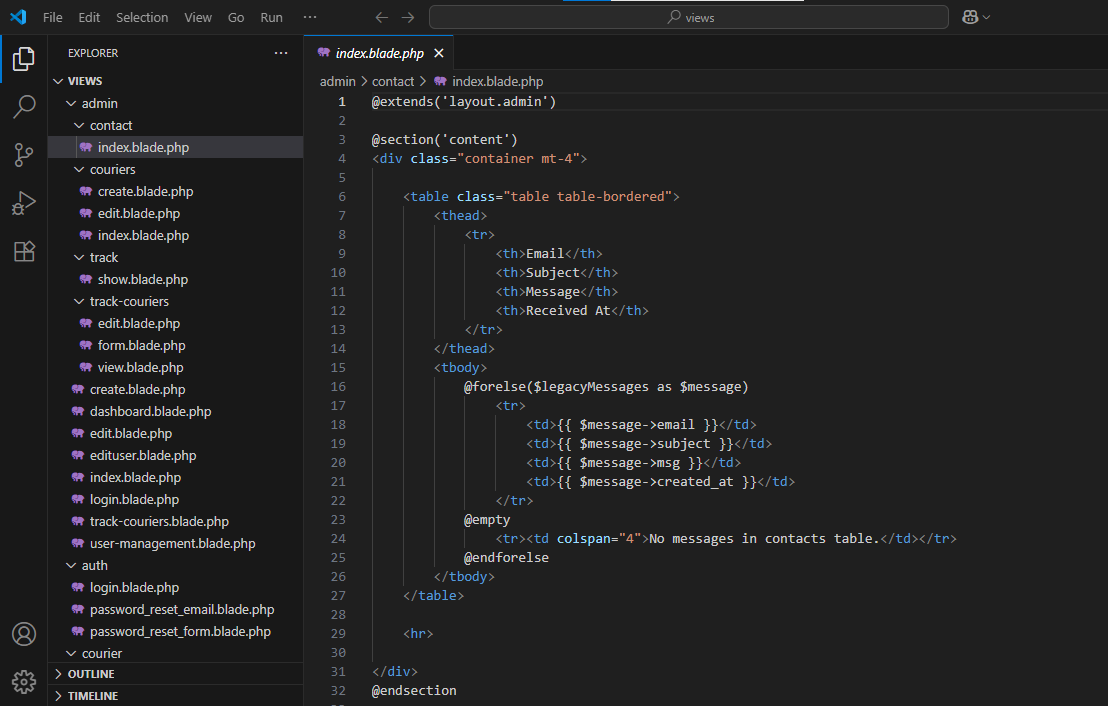
****

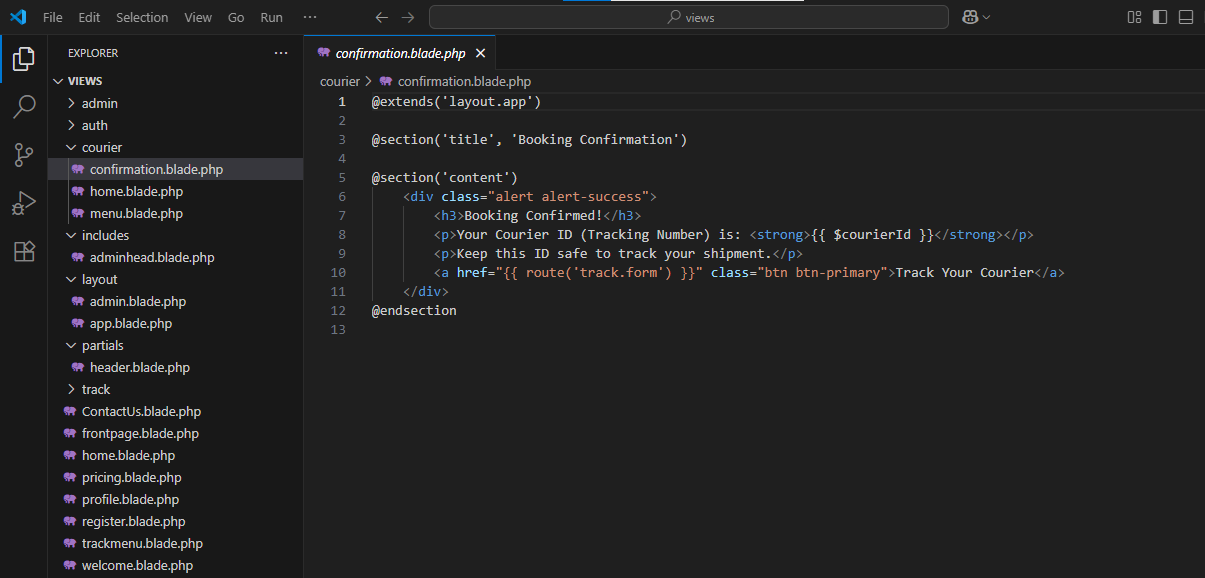
1. **Users:**

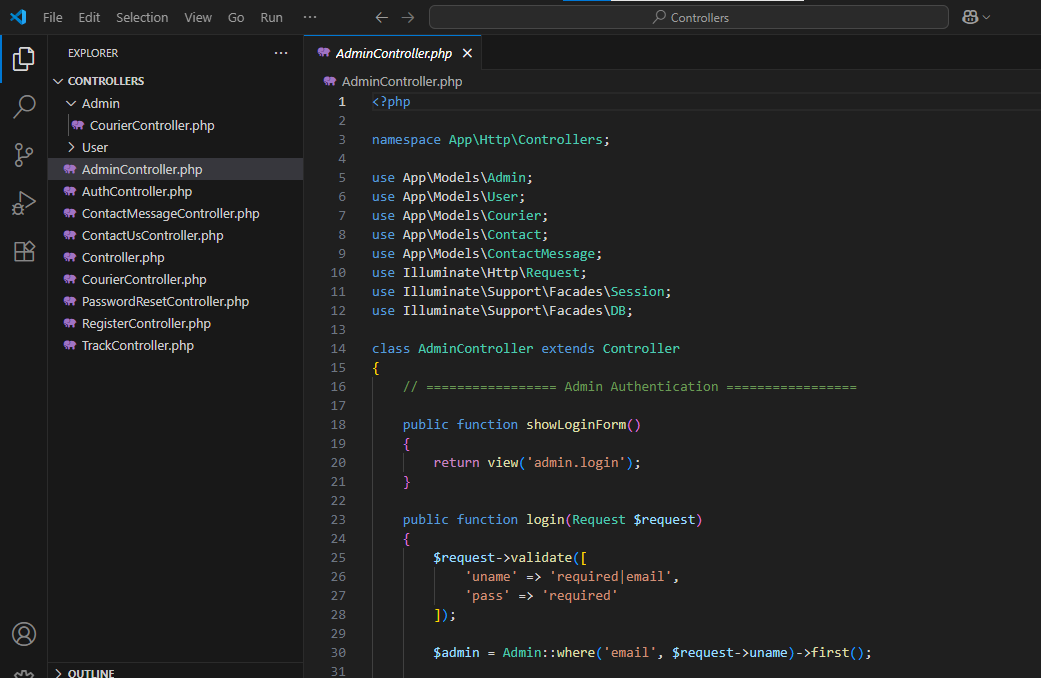
****

* 1. **LARAVEL IMPLEMENTATION DETAILS:**

**7.1 Model **

**7.2 Views:**

****

**7.3 Controllers:**

### ****8. SYSTEM FUNCTIONALITY****

#### ****8.1 ROLE-BASED ACCESS CONTROL****

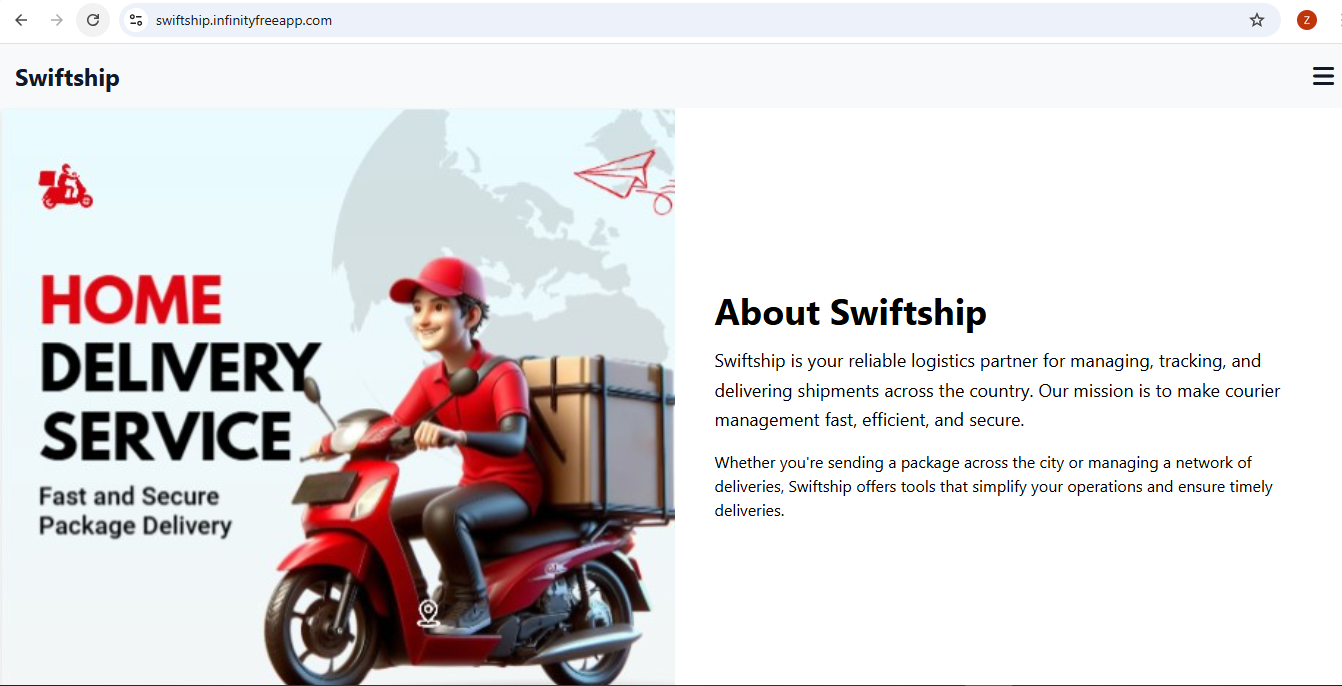
##### **8.1.1 ADMIN FUNCTIONS**

* **User Management:** View, verify, edit, and delete registered user accounts
* **Courier Oversight:** Add new couriers, update tracking details, and manage delivery statuses
* **Status Management:** Assign and update real-time courier delivery status
* **Message Center:** View and respond to contact messages from users
* **System Monitoring:** Access courier statistics, total shipments, and user activity reports

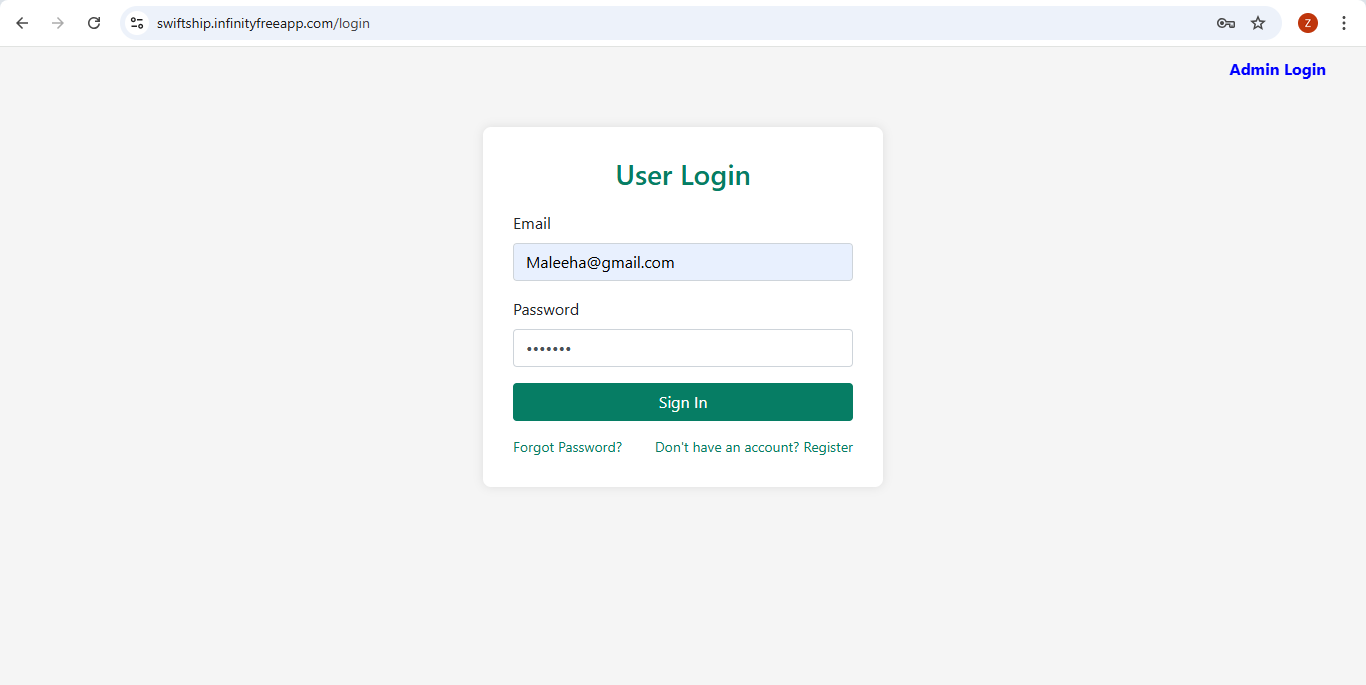
##### **8.1.2 USER FUNCTIONS**

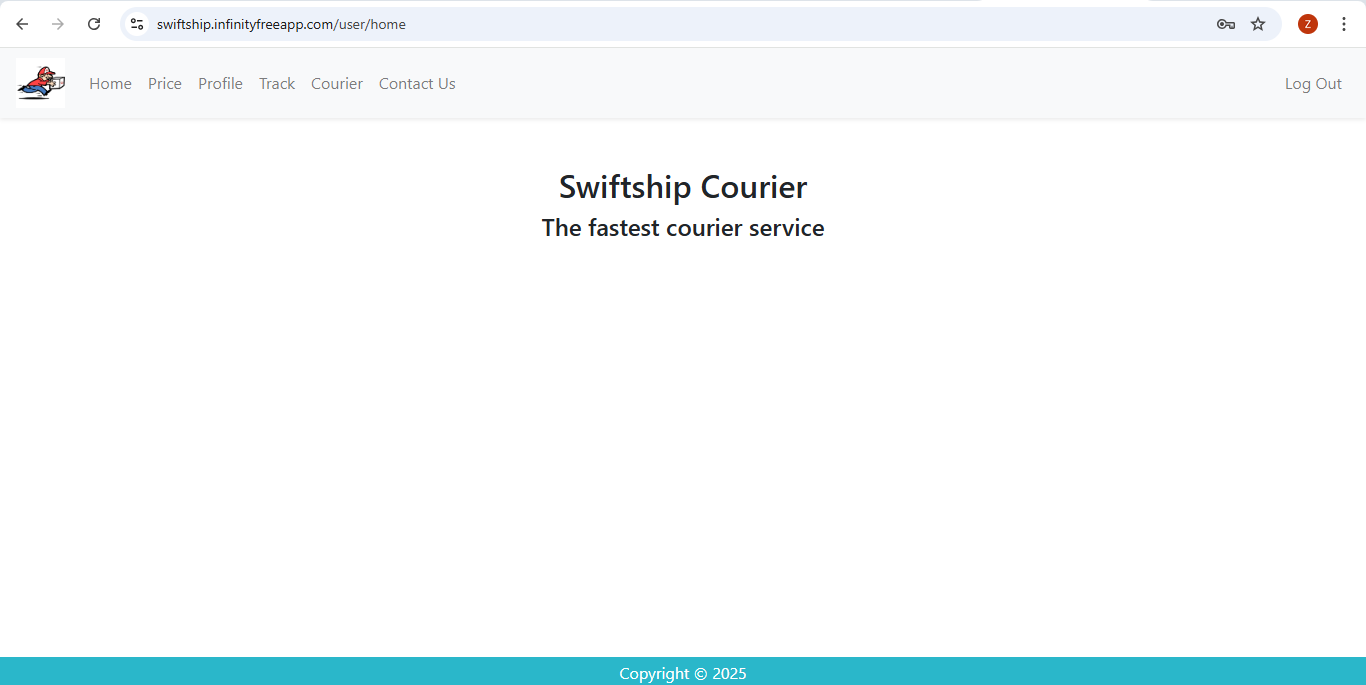
* **Courier Tracking:** Track the status of submitted shipments using unique tracking IDs
* **Profile Management:** Update personal details such as name, email, and password
* **Courier History:** View all courier records submitted by the user
* **Contact Support:** Submit queries or concerns via the contact form
* **Notifications:** Get real-time updates on courier status changes (e.g., "In Transit", "Delivered")

**Front Page:**

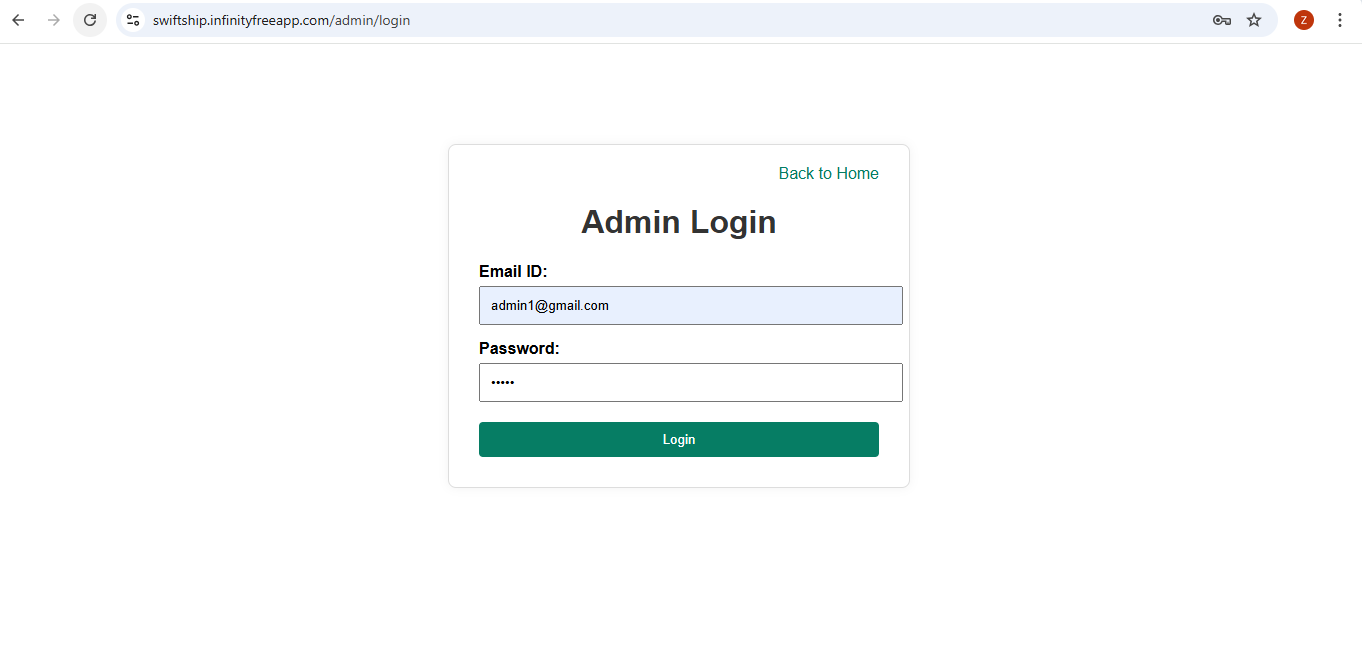
****

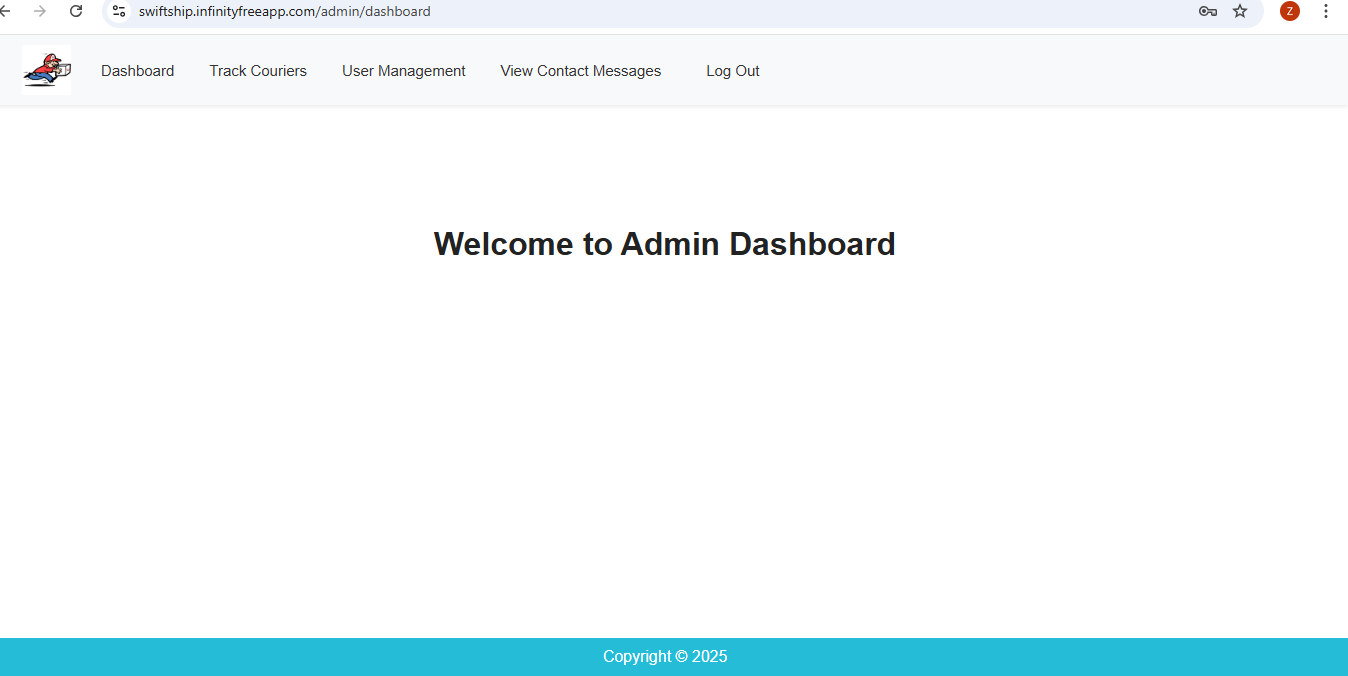
**User End:**

****

****

**Admin End:**

****

****

* 1. **SECURITY IMPLEMENTATIONS:**
* **Authentication:** Login required for both admin and user accounts
* **Authorization:** Admins have access to backend panels; users are restricted to personal functionality
* **Input Validation:** All user inputs are validated using Laravel’s built-in validation rules
* **File Security:** No file uploads allowed from frontend, reducing upload-based threats
* **Session Security:** Laravel session-based login ensures secure access
* **Query Safety:** Use of prepared statements prevents SQL injection attacks

### 10. ****FUTURE ENHANCEMENTS****

#### ****10.1 SCALABILITY IMPROVEMENTS****

* **Multi-Branch Support:** Add functionality for managing couriers across different cities or branches
* **API Integration:** Provide RESTful APIs for mobile apps and external logistics platforms
* **Real-Time Tracking:** Integration with GPS modules for live location tracking
* **Automated Email Alerts:** Notify users via email at each status update
* **Admin Roles:** Different admin levels for shipment handlers, super admins, and moderators

#### ****10.2 FEATURE EXPANSIONS****

* **Barcode/QR Code Generator:** For each shipment to ease physical handling and tracking.
* **SMS Notification Integration:** Inform users via SMS when courier status changes.
* **Payment Integration:** Add support for cash-on-delivery or prepaid courier billing.
* **Mobile App:** Native Android/iOS app for easier courier tracking.
* **Feedback System:** Let users’ rate courier services for quality assurance.

### ****11**.** ****CONCLUSION****

The Courier Management System demonstrates the practical application of database management and Laravel development in solving a real-world logistics challenge. The system provides a centralized platform for managing, tracking, and updating courier shipments while ensuring role-based access, usability, and efficiency.

#### ****11.1 THE PROJECT SUCCESSFULLY DEMONSTRATES:****

* **Efficient Database Design:** Fully normalized schema with clear relationships and constraints
* **Role-Based Functionality:** Distinct modules for admin and user with RBAC implementation
* **Web Development Skills:** End-to-end Laravel MVC integration with Blade templating
* **Secure Deployment:** Application successfully hosted on InfinityFree with database integration
* **System Usability:** Clean UI, easy tracking, and real-time courier updates ensure good user experience

This project has provided essential experience in building scalable, secure, and practical web applications, preparing us for future endeavors in full-stack software development.

* 1. **REFERENCES**:

Chatgpt

* 1. **DEPLOYMENT LINK:**

<https://swiftship.infinityfreeapp.com/>

**14. GITHUB LINK:**

https://github.com/Zarmina2311/Courier-Management-System