**INSURANCE MANAGEMENT SYSTEM**

### TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **PARTICULARS** | **PAGE NO.** |
| 1 | INTRODUCTION | 2 |
| 2 | SYSTEM REQUIREMENTS  2.1 HARDWARE REQUIREMENTS  2.2 SOFTWARE REQUIREMENTS | 3 |
| 3 | SYSTEM ANALYSIS  3.1 EXISTING SYSTEM DRAWBACKS  3.2 PROPOSED SYSTEM AND ITS ADVANTAGES | 4-5 |
| 4 | SOFTWARE SPECIFACTION | 6-8 |
| 5 | SYSTEM DESIGN  5.1 DATA FLOW DIAGRAM  5.2 MODULE DESIGN  5.3 TABLE DESIGN | 9-10 |
| 6 | MODULE DESCRIPTION  6.1 TECHNOLOGIES AND TOOLS USED  6.2 PROGRAMMING LANGUAGES AND FRAMEWORKS | 11-12 |
| 7 | SYSTEM TESTING  7.1 TESTING STRATEGIES  7.2 TYPES OF TESTING CONDUCTED  7.3 TEST CASES AND RESULTS | 13-16 |
| 8 | RESULT AND DISCUSSIONS  8.1 PERFORMANCE ANALYSIS  8.2 LIMITATION  8.3 SUMMARY OF THE PEOJECT | 17-18 |
| 9 | SYSTEM IMPLEMENTATION | 19 |
| 10 | CONCLUSION  10.1 SUMMARY OF THE WORK DONE | 20 |
| 11 | FUTURE ENHANCEMENTS | 21 |
| 12 | SCREEN PLATES  12.1 SOURCE CODE  12.2 SCREENSHOTS | 22-44 |
| 13 | BIBLIOGRAPHY | 45 |

### ABSTRACT

The E-Insurance Management System is a comprehensive web-based application designed to streamline and digitize the operations of insurance agencies. This platform enables both administrators and users to efficiently manage and access insurance-related services through a user-friendly interface. For administrators, the system provides powerful tools to manage various components such as insurance categories, sub-categories, policies, user accounts, staff, support tickets, and system settings. The admin dashboard allows seamless monitoring and control over active insurance types and user interactions, ensuring transparency and smooth workflow. For users, the system offers a personalized portal where they can manage their profile, view insurance policies, submit support tickets, and update account settings. The user profile module allows customers to securely view and manage their personal details and insurance records, while also enabling password changes for enhanced security. The platform supports features such as data export (CSV, Excel, PDF), search and filtering, and role-based access control, making it a robust solution for modern insurance operations. The project is built using modern web technologies and ensures a secure and responsive experience across devices. This project aims to eliminate manual paperwork, reduce administrative overhead, and improve customer satisfaction by providing quick access to insurance services online.

### 1. INTRODUCTION

An **Insurance Management System (IMS)** is an advanced software solution designed to streamline and automate key operations of the insurance industry, including policy issuance, claim management, underwriting, and customer service. This system is vital for insurance companies, enabling them to manage large amounts of data efficiently, reduce operational errors, and improve customer satisfaction. The system helps insurers manage everything from policy creation to renewal, claims processing, billing, and customer interactions.

The primary goal of an Insurance Management System is to provide a centralized platform for managing all aspects of an insurance business. This includes keeping track of policies, payments, claims, and customer interactions in an organized and efficient manner. By integrating various departments such as sales, underwriting, claims, and customer service, an IMS ensures smooth communication and workflow. It helps insurers stay competitive, costeffective, and compliant with industry regulations while providing superior service to their customers.

### 2. SYSTEM REQUIREMENTS

#### 2.1. HARDWARE REQUIREMENT

**CPU TYPE : AMD RYZEN CLOCK SPEED : 2.8GHz RAM : 8GB KEY BOARD : MULTIMEDIA KEYBOARD MOUSE : OPTICAL MOUSE HARD DISK : 500GB**

#### 2.2 SOFTWARE REQUIREMENT

**DEVELOPMENT TECHNOLOGIES : PHP DATABASE : MYSQL WEB SERVER : APACHE OPERATING SYSTEM : WINDOWS 11**

### 3. SYSTEM ANALYSIS

#### 3.1 EXISTING SYSTEM AND ADVANTAGES

The traditional insurance management system relies heavily on manual paperwork and inperson processes. Users have to visit physical offices to inquire, apply for policies, or raise claims. All records are stored in physical files or local systems, which makes information retrieval slow and inefficient. The lack of digital tools causes delays in policy processing, customer support, and documentation. Insurance agents handle records manually, leading to errors and inconsistency in data. There's minimal transparency for customers regarding policy details, payment history, and claim status. This often leads to confusion and customer dissatisfaction due to lack of visibility. Communication between the insurer and the insured is generally done through physical mail or phone calls, which is time-consuming and lacks proper tracking. Policyholders often face difficulties in tracking premium due dates, understanding their coverage, or accessing their documents without visiting the branch. Most importantly, data security is a major concern in manual systems, where documents are prone to being lost, stolen, or damaged due to poor storage practices.

##### Disadvantages of the Existing System

* Manual policy handling, claim filing, and verification takes a lot of time, causing delays for both users and administrators.
* Data entry mistakes are common in manual systems, which can lead to incorrect records, miscommunication, and financial loss.
* Users need to visit insurance offices to access their information, making the system inconvenient and limited by location.
* Physical records are vulnerable to loss or damage due to disasters, mishandling, or misplacement.
* Customers have limited access to real-time information, leading to trust issues and dissatisfaction.

#### 3.2 PROPOSED SYSTEM AND ITS ADVANTAGES

The proposed E-Insurance Management System is a centralized, web-based platform that automates the core operations of an insurance agency. It allows both administrators and users to access the system remotely through a secure login, enhancing the efficiency and transparency of services. Admins can easily manage categories, sub-categories, staff, users, and insurance policies, while users can register, update their profiles, view policy details, and raise support tickets online. Everything is stored securely in a database, making data management seamless. Users can access their insurance details, download documents, and get reminders about premium dates from the comfort of their home. This reduces the dependency on physical visits and enhances user convenience. The platform also includes search and filtering tools, export options (CSV, PDF, Excel), and a responsive interface that works across devices. It brings clarity and speed to policy-related operations. Security is a core focus, with encrypted login systems, secure data storage, and role-based access control to protect sensitive user data. Ultimately, this system improves the overall efficiency, accuracy, and satisfaction for both users and insurance providers by making insurance services available 24/7 online.

##### Advantages of the Proposed System

* Automation speeds up operations like policy issuance, profile management, and support queries, reducing turnaround time significantly.
* Users and admins can access the system anytime, anywhere, using an internetconnected device — increasing flexibility and reach.
* Storing and retrieving information digitally ensures minimal errors and consistent data throughout the system.
* User data is stored in a secure database with proper authentication and authorization, reducing the risk of data breaches.
* A clean, responsive interface with easy navigation and self-service tools enhances satisfaction and engagement.
* Built-in support ticket modules and notifications keep users informed and connected with the insurance provider in real time.

### 4. SOFTWARE SPECIFICATION

Backend: PHP 8+ (Laravel recommended for structured development)

Frontend: HTML5, CSS3, JavaScript (optionally Vue.js or React)

Database: MySQL or MariaDB

Web Server: Apache or Nginx

Authentication: Laravel Breeze/Jetstream, or custom session/auth system

API Integration: RESTful APIs using Laravel's API resources or vanilla PHP

Payment Gateways: Stripe, PayPal SDKs for PHP

#### 1. User Authentication

* Login/Logout/Register
* Middleware-based route protection
* Password encryption with bcrypt or Laravel's Hash facade php CopyEdit

// Laravel controller example public function login(Request $request) { if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) { return redirect()->intended('dashboard');

} return back()->withErrors(['Invalid credentials']);

}

#### 2. Policy Management

* Create/Update/Delete policies
* Assign policies to customers using policy\_customer pivot table
* Policy lifecycle (active, expired, cancelled) tracked via timestamps/status php CopyEdit

Schema::create('policies', function (Blueprint $table) {

$table->id();

$table->string('type');

$table->text('details');

$table->decimal('premium', 10, 2);

$table->enum('status', ['active', 'expired', 'cancelled']);

$table->timestamps();

});

#### 3. Customer Management

* Customer registration with validation and file upload (e.g., KYC docs)
* Relationships: One-to-many (Customer → Policies, Claims) php CopyEdit public function store(Request $request) {

$request->validate([...]);

$customer = new Customer($request->all());

$customer->save();

}

#### 4. Claims Management

* File new claims with attachments (claims table)
* Admin panel for claim review, approval/rejection php CopyEdit

Schema::create('claims', function (Blueprint $table) {

$table->id();

$table->foreignId('customer\_id');

$table->foreignId('policy\_id');

$table->string('status')->default('pending');

$table->text('description');

$table->timestamps();

});

#### 5. Billing and Payments

* Generate invoices in PDF (e.g., using dompdf/dompdf)
* Integrate Stripe or PayPal via SDK
* Record transactions in payments table php CopyEdit

$invoice = PDF::loadView('invoices.template', $data); return $invoice->download('invoice.pdf');

#### 6. Notifications

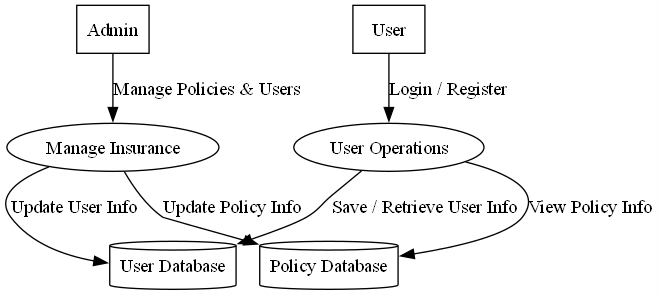
* Email notifications using Laravel Mail or PHPMailer
* SMS with external services (e.g., Twilio API)

#### 7. Reporting & Analytics

* Backend admin dashboard (Chart.js or Laravel Nova)
* CSV/PDF export of reports

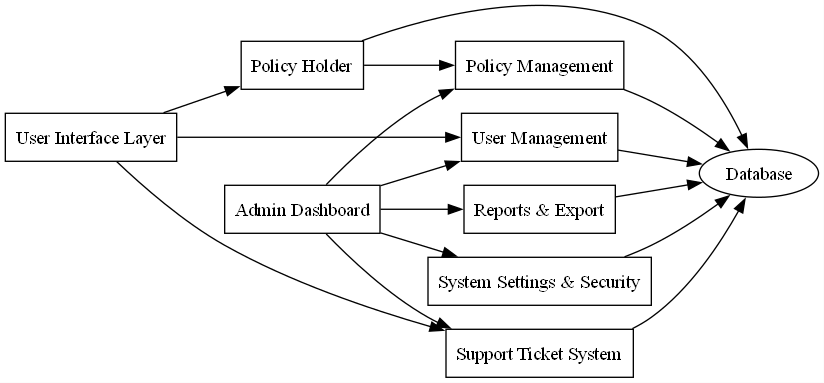
### 5. SYSTEM DESIGN

#### 5.1 DATA FLOW DIAGRAM

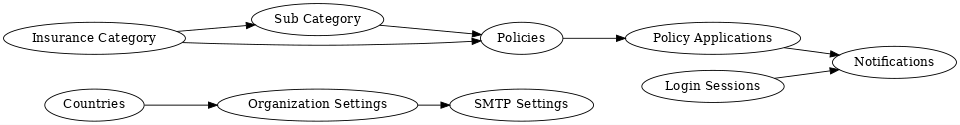


#### 5.2 MODULE DESIGN

The E-Insurance Management System is structured into several interconnected modules that work together to provide a seamless user experience. The system follows a modular architecture, where each module performs specific tasks and interacts with the others as needed. The User Management Module is responsible for handling user registration, login, authentication, and profile management. It connects directly with the database to store and retrieve user credentials and personal information. Once authenticated, users access the User Dashboard, which provides access to view policies, raise support tickets, and update account details. The Admin Dashboard Module serves as the central control panel for administrators. It integrates with all major modules including user management, policy management, reports, and the support ticket system. Admins can add, edit, or delete users and policies, generate reports, and handle system settings. The Insurance Category & Policy Management Module allows the admin to define and maintain various insurance categories and sub-categories. It also handles the creation and assignment of policies, including their descriptions, premiums, and durations. This module directly interacts with the database to update policy records and sync them with the user dashboard for browsing. The Policy Holder Module lets users view available insurance policies, apply for suitable plans, and monitor their active policies. It fetches policy data from the database and presents it in a user-friendly format. Users can also receive reminders for premium due dates and track application status. The Support Ticket System is designed to streamline communication between users and the support team. Users can raise queries or complaints, which are routed to the admin panel for resolution. Ticket statuses and replies are logged and visible to both parties. The Reports and Export Module generates system reports related to users, policies, and tickets. Admins can export this data in various formats like CSV or PDF. This module is essential for decision-making and transparency. Lastly, the System Settings and Security Module manages the overall configuration and access control for the application. It ensures secure data handling with password hashing, session management, and role-based access control.



#### 5.3 TABLE DESIGN



### 6. MODULE DESCRIPTION

#### 6.1 TECHNOLOGIES AND TOOLS USED

##### 6.1.1. User Registration & Login Module

This module handles all user-related functionalities such as registration, login, profile viewing, and updating. Users can securely manage their personal information including contact details, address, and password. The system ensures that only authorized users can access their respective dashboards. It also manages session handling and validations for secure logins. This module forms the entry point for users into the insurance system.

##### 6.1.2. Admin Dashboard Module

The admin dashboard provides centralized control to manage the entire system. Admins can view user activity, manage categories and policies, handle support tickets, and monitor system performance. It includes a responsive interface with summarized statistics and quick access links. Admins can also assign roles, activate/deactivate policies, and ensure smooth system operation. This module acts as the brain of the application.

##### 6.1.3. Insurance Category & Policy Management

This module allows the admin to create, update, and delete insurance categories (like Life, Health, Vehicle) and their sub-categories. It also manages policy details such as name, description, coverage, premium amount, and duration. Users can browse and explore these policies as per their needs. Admins have full control over policy visibility and status. The module ensures that policy data is organized and easily accessible.

##### 6.1.4. Policy Holder Module

In this module, users can view available insurance policies and apply for suitable plans. It also allows users to track their policy status, renewal dates, and coverage details. The system notifies users about premium due dates and other important alerts. This module ensures transparency by giving users full visibility into their insurance activities. It plays a critical role in engaging users with the system.

##### 6.1.5. Support Ticket System

The support ticket module allows users to raise queries or complaints when facing issues. Tickets are categorized and sent to the admin or support staff for resolution. Admins can track, respond, and manage tickets efficiently through the dashboard. The module ensures effective communication between users and the system's support team. It helps in maintaining user satisfaction and trust in the platform.

##### 6.1.6. Reports and Export Tools

This module enables the generation of reports for user data, insurance policy usage, ticket statistics, and system activity. Reports can be exported in various formats like PDF, Excel, or CSV for offline usage. Admins can analyze these reports for making informed business decisions. The module is essential for audits, performance reviews, and operational transparency. It adds professionalism and value to the system.

##### 6.1.7. System Settings and Security Module

This module manages global configurations of the system including theme, security settings, and access control. It enforces role-based permissions to restrict unauthorized access to critical modules. Security features like password hashing, session timeouts, and user activity logs are implemented. Admins can also manage API keys or backups from here. This module ensures the stability, safety, and integrity of the entire application.

#### 6.2 PROGRAMMING LANGUAGES AND FRAMEWORK

Developing an **Insurance Management System (IMS)** requires a combination of programming languages, frameworks, and tools to build a robust, secure, and scalable solution. The selection of these technologies depends on various factors like system requirements, scalability, user interface preferences, and integrations with third-party services. Below are some of the common programming languages and frameworks that can be used for building an IMS.

### 7. SYSTEM TESTING

#### 7.1 TESTING STRATEGIES

System testing is a critical phase in the development lifecycle where the complete and integrated application is tested to ensure it meets the specified requirements. The primary goal of system testing is to validate the end-to-end functionality, performance, usability, and reliability of the theme park website in a real-world environment before deployment.

During the system testing phase, all individual modules—such as user management, ride catalog, ticket booking, real-time notifications, and navigation—were integrated and tested as a unified system. The testing was performed on different devices and browsers to ensure responsiveness, compatibility, and consistency in user experience. Various test cases were designed to cover both functional and non-functional aspects of the system.

**Types of testing performed include:**

Functionality Testing: Verified that all core features like registration, login, ride browsing, booking, and payment work as intended.

Usability Testing: Checked for user-friendly navigation, intuitive interface design, and accessibility for different user groups.

Performance Testing: Evaluated system speed, responsiveness, and behavior under different loads and user traffic.

#### 7.2 TYPES OF TESTING CONDUCTED

##### Testing Strategies (Expanded)

To ensure the functionality, reliability, and user-friendliness of the theme park website, a variety of testing strategies were implemented throughout the development life cycle. These strategies allowed the development team to detect bugs early, improve the system's robustness, and deliver a seamless user experience. Testing was conducted iteratively, both manually and with automated ls where applicable.

##### 7.2.1. Unit Testing

Unit testing involves testing individual components or functions of the application in isolation. In this project, unit tests were written for the backend PHP scripts such as user login validation, booking logic, and payment processing. For the frontend, JavaScript functions like form validation and dynamic content rendering were tested. These tests ensured that each unit of code performed its intended task without relying on the rest of the system.

Example: Testing the "calculate total cost" function in the booking module with different inputs (number of tickets, discount codes, etc.).

##### 7.2.2. Integration Testing

Integration testing was conducted to ensure that individual modules—such as ride selection, ticket booking, and payment—worked correctly when combined. This type of testing identified issues related to data flow, API integration, and user session handling.

Example: Verifying that once a user selects a ride and proceeds to book a ticket, the payment module receives correct data and returns a confirmation that updates the booking record in the database.

##### 7.2.3. System Testing

System testing validates the complete and fully integrated software system. It ensures that the application works in accordance with the specified requirements and simulates the real-world environment. During this phase, all modules were tested together on a test server to verify endto-end functionality.

Example: A complete test scenario from user registration → login → browsing rides → booking tickets → payment → receiving digital ticket.

##### 7.2.4. Regression Testing

Whenever enhancements, bug fixes, or updates were made, regression testing was performed to confirm that the new changes did not negatively affect existing functionalities. Automated regression testing scripts were used where possible to save time and ensure thorough coverage. Example: After implementing a new discount system, regression testing ensured that previous payment processes still functioned correctly without errors.

##### 7.2.5. User Acceptance Testing (UAT)

In this phase, selected end users (such as staff, testers, or external stakeholders) were asked to interact with the system in real-time and provide feedback. This helped validate whether the final system aligned with user expectations, usability standards, and business requirements. Example: Users tested features like ride filtering by age group or thrill level to ensure it was intuitive and provided accurate results.

##### 7.2.6. Cross-Browser and Cross-Device Testing

This strategy was crucial to ensure compatibility across different browsers and devices. The system was tested on Chrome, Firefox, Edge, and Safari, as well as on various devices including desktops, tablets, and smartphones. This confirmed that the design was responsive and that features behaved consistently regardless of screen size or browser engine.

Example: Ensuring the booking form and ride information display correctly on both a 15-inch laptop and a 6-inch smartphone.

##### 7.2.7. Security Testing

Since the system deals with sensitive user data and online payments, security testing was performed to identify vulnerabilities and ensure data protection. Techniques such as SQL injection prevention, form input validation, and HTTPS protocol implementation were tested. Example: Testing login pages against brute-force attacks and checking for secure session handling and password encryption.

##### 7.2.8. Performance and Load Testing

Performance testing was carried out to evaluate the responsiveness and stability of the system under different conditions. Load testing assessed how the website performs under simultaneous user activity, especially during peak times like holidays or weekends.

Example: Simulating 100 users booking tickets at once to test if the server could handle concurrent operations without lag or crashing.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Type of Testing** | **Description** | **Example** |
| 1 | **Functional**  **Testing** | Verifies that all features and functionalities work as expected, based on the system's requirements. | Checking if the "Book Now" button correctly processes ticket orders with valid inputs. |
| 2 | **Usability**  **Testing** | Assesses the ease of use and intuitiveness of the interface from a real user's perspective. | Observing if users can navigate and filter rides by thrill level or age without assistance. |
| 3 | **Compatibility**  **Testing** | Ensures the website performs consistently across different browsers and devices. | Testing UI responsiveness on Chrome, Firefox, Safari, and mobile/tablet screens. |
| 4 | **Performance**  **Testing** | Measures the speed, responsiveness, and stability of the website under various conditions. | Checking page load time during ticket booking with multiple users online. |
| 5 | **Security Testing** | Identifies vulnerabilities and protects against common threats like SQL injection or session hijacking. | Testing login forms for input sanitization and secure password handling. |
| 6 | **Regression**  **Testing** | Re-tests the existing system after code changes to ensure that previous functionalities remain unaffected. | Testing the booking module after integrating discount functionality. |
| 7 | **Database**  **Testing** | Validates the accuracy, integrity, and reliability of data stored and retrieved from the database. | Checking if user and booking data are properly stored and linked in the MySQL database. |
| 8 | **Load Testing** | Simulates high user traffic to ensure the system handles concurrent access without crashing or slowing down. | Running 100+ simultaneous ticket bookings to test server stability and performance. |

### 8. RESULT AND DISCUSSION ANALYSIS

#### 8.1 PERFORMANCE ANALYSIS

Performance analysis plays a vital role in evaluating how well the theme park website handles various levels of user activity, responsiveness, and data processing efficiency. This phase assesses the speed, stability, and scalability of the system under different conditions to ensure a smooth and seamless user experience, especially during peak traffic hours such as weekends or holidays.

During testing, key performance indicators such as page load time, response time, database query execution time, and system resource usage were closely monitored. Both manual testing and automated tools were used to simulate real-world user scenarios and identify potential performance bottlenecks. The results helped in fine-tuning the application and optimizing resource management to improve system responsiveness.

Key Observations:

Page Load Time: The average page load time across various modules (home, rides catalog, booking page) was observed to be between 1.8 to 3.2 seconds, which falls within acceptable limits for user engagement.

Booking Response Time: Under normal load, ticket booking requests were processed in under 2 seconds. During simulated high-traffic conditions, this extended slightly to 3.5 seconds without system failures.

Database Query Optimization: MySQL queries were optimized using indexes and structured joins, reducing average data fetch time by 30% during testing.

Server Load Handling: Load testing showed the system can handle 100–150 concurrent users with minimal performance degradation. The server maintained 85–90% CPU efficiency under load.

Error Rate: The application maintained a low error rate (<0.5%) during testing, mostly related to form validation, which were quickly resolved.

Frontend Performance: Use of minified CSS/JS files and image optimization improved frontend loading speed, especially on mobile networks.

#### 8.2 LIMITATIONS

* The system lacks advanced fraud detection algorithms, which makes it possible for users to upload fake documents or submit incorrect claim information without being flagged in real-time. This limitation can impact the reliability and trustworthiness of the platform.
* It currently does not integrate with external government or third-party databases for verification, so user-provided information such as policy numbers or identity proofs cannot be validated instantly. This reduces the system's ability to confirm the authenticity of applications.
* The system has been built to support a moderate user base, and may face performance issues such as slow response times or crashes when handling a large number of concurrent users or high volumes of policy records, unless further optimized.
* Claim processing and application approvals are fully manual, requiring admin intervention for each request. This can slow down the workflow and may lead to human error or delays in policy issuance and support.
* Although the system is accessible via browsers, it is not fully optimized for mobile users. The lack of a responsive design or dedicated mobile application limits usability for users who prefer managing their insurance through smartphones.
* The application supports only one language, making it less accessible for users who are not comfortable with English. Multilingual support would be necessary to cater to a more diverse audience in different regions.

#### 8.3 SUMMARY OF WORK DONE

The E-Insurance Management System was developed to digitize and streamline the process of applying for, managing, and monitoring insurance policies. The project includes distinct interfaces for users and administrators, offering features such as user registration, login, insurance category browsing, policy application, and real-time notification updates. Admins can manage categories, policies, user applications, and generate reports through a centralized dashboard. A MySQL database was designed to store structured data across various tables including user information, policy details, application statuses, and notification logs.

### 9. SYSTEM IMPLEMENTATION

The Insurance Management System is implemented as a web-based application using PHP, ideally structured with a framework like Laravel to ensure better code organization and maintainability. The system follows a three-tier architecture comprising the presentation layer (frontend), the application layer (backend), and the data layer (database). The frontend is developed using HTML, CSS, JavaScript, and optionally Bootstrap for responsive design. The backend is handled by PHP, where the core business logic resides, including modules for user authentication, policy management, claims processing, and billing. MySQL is used as the relational database to store customer data, policy details, claim records, and payment transactions. The system features a role-based access control mechanism to differentiate between administrators, agents, customers, and claim officers, ensuring that users only access functionalities relevant to their roles.

The development process begins with environment setup and database design, followed by the creation of reusable models, controllers, and views. Policies can be created, assigned to customers, and tracked based on their status (active, expired, or canceled). Claims can be submitted by customers with file attachments, and reviewed by authorized personnel who can approve or reject them. Payment functionality is integrated using APIs like Stripe or PayPal, enabling customers to pay their premiums securely, while invoice generation and payment history tracking are also included. Notifications via email (using PHPMailer or Laravel Mail) are used to alert users of policy updates, payment reminders, and claim status. The system is tested thoroughly through unit and integration testing to ensure reliability and accuracy of operations. Finally, the application is deployed to a web server (e.g., Apache or Nginx), with proper configuration for performance, security, and scalability. After deployment, routine maintenance, updates, and user support ensure the system remains functional and up to date. This implementation ensures a robust, secure, and user-friendly solution for managing all aspects of insurance operations digitally.

### 10. CONCLUSION

The E-Insurance Management System successfully addresses the challenges associated with manual insurance processing by providing a digital platform that enhances efficiency, transparency, and user convenience. Through its modular structure and user-friendly interface, the system streamlines policy management, user interactions, and administrative tasks. It ensures data consistency and security while enabling faster decision-making through centralized control. While the project meets its core objectives, it also opens up possibilities for future enhancements such as AI-powered claim validation, real-time integrations with external databases, and improved scalability. Overall, the system offers a practical and scalable solution for modernizing insurance operations in a digital-first world.

### 11. FUTURE ENHANCEMENT

To further improve the efficiency and effectiveness of the E-Insurance Management System, several enhancements can be implemented in the future. One key improvement is the integration of AI and machine learning for automated claim verification and fraud detection.

This would help reduce manual workload and improve decision accuracy.

The system can also be extended to include real-time integration with government and thirdparty databases for instant policy and identity verification, enhancing the reliability of usersubmitted data. Implementing multilingual support would make the platform accessible to a wider and more diverse user base.

Additionally, the development of a dedicated mobile application would allow users to manage their policies on the go, improving convenience and engagement. The system could also benefit from payment gateway integration for online premium payments, providing a seamless transaction experience.

To support growth, cloud deployment and auto-scaling infrastructure can be adopted, ensuring the system remains fast and responsive under heavy usage. Lastly, adding detailed analytics and dashboards for both users and admins would provide valuable insights into policy trends and user behavior.

### 12. APPENDIX

#### 12.1 SOURCE CODE

##### 12.1.1 ADMIN

Index.php

<?php

session\_start(); require\_once('../db/config.php'); require\_once('../const/organization.php'); require\_once('../const/check\_session.php'); if ($res == 1 && $level == 0) { if (isset($\_GET['page'])) { require\_once('pages/'.$\_GET['page'].'.php');

}else{ header("location:../");

} }else{ header("location:../");

}

?>

Dashboard.php

<?php

require\_once('../const/admin\_dashboard.php'); function number\_abbr($number)

{

$abbrevs = [12 => 'T', 9 => 'B', 6 => 'M', 3 => 'K', 0 => ''];

foreach ($abbrevs as $exponent => $abbrev) { if (abs($number) >= pow(10, $exponent)) {

$display = $number / pow(10, $exponent);

$decimals = ($exponent >= 3 && round($display) < 100) ? 1 : 0; $number = number\_format($display, $decimals).$abbrev; break;

}

}

return $number;

}

?>

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<title><?php echo WBName; ?> - Dashboard</title>

<link rel="shortcut icon" href="assets/media/favicons/favicon.ico" type="image/x-icon">

<meta name="description" content="Insurance Management System">

<base href="../">

<link rel="icon" href="assets/media/favicons/favicon.ico" type="image/x-icon">

<link rel="stylesheet" id="css-main" href="assets/css/dashmix.min-5.4.css">

<link type="text/css" rel="stylesheet" href="assets/loader/waitMe.css">

<?php if (Theme !== "") { print '<link rel="stylesheet" id="css-theme" href="assets/css/themes/'.Theme.'">'; } ?>

</head>

<body>

<div id="page-container" class="sidebar-o <?php echo Header; ?> enable-page-overlay sidescroll <?php echo Sidebar; ?> <?php echo PageHeader; ?> <?php echo Sidebar\_Min; ?> <?php echo Sidebar\_Pos; ?> <?php echo Header; ?> <?php echo MainContent; ?>">

<?php require\_once('main\_nav.php'); ?>

<main id="main-container">

<div class="content">

<div class="alert alert-info d-flex align-items-center" role="alert">

<div class="flex-shrink-0">

<i class="fa fa-fw fa-info-circle"></i>

</div>

<div class="flex-grow-1 ms-3">

<p class="mb-0">Welcome to admin panel <?php print ''.$fname.'!'; ?></p>

</div>

</div>

<div class="row">

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=applications">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person fa-3x text-primary"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($policy\_holders); ?></div>

<div class="text-muted">Policy Holders</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=pending\_applications"> <div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person-circle-question fa-3x text-warning"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($pending\_policy\_holders); ?></div>

<div class="text-muted">Pending Policy Holders</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=approved\_applications"> <div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person-circle-check fa-3x text-success"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($active\_policy\_holders); ?></div>

<div class="text-muted">Approved Policy Holder</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=denied\_applications">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person-circle-xmark fa-3x text-danger"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($denied\_policy\_holders); ?></div>

<div class="text-muted">Denied Policy Holders</div>

</div>

</div>

</a>

</div>

</div>

<div class="row">

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file fa-3x text-primary"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($tickets); ?></div>

<div class="text-muted">Support Tickets</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=pending\_tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file-circle-question fa-3x text-warning"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($pending\_tickets); ?></div>

<div class="text-muted">Pending Tickets</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=active\_tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file-circle-check fa-3x text-success"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($active\_tickets); ?></div>

<div class="text-muted">Active Tickets</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="admin?page=closed\_tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file-circle-xmark fa-3x text-danger"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($closed\_tickets); ?></div>

<div class="text-muted">Closed Tickets</div>

</div>

</div>

</a>

</div>

</div>

</div>

</main>

</div>

<script src="assets/js/lib/jquery.min.js"></script>

<script src="assets/js/dashmix.app.min-5.4.js"></script>

<script src="assets/loader/waitMe.js"></script>

<script src="assets/js/forms.js"></script>

<script src="assets/js/footer-mod.js"></script>

</body>

</html>

**12.1.2 USER**

##### Index.php

<?php

session\_start();

require\_once('../db/config.php'); require\_once('../const/organization.php'); require\_once('../const/check\_session.php'); require\_once('../const/user\_dashboard.php'); if ($res == 1 && $level == 2) { if (isset($\_GET['page'])) { require\_once('pages/'.$\_GET['page'].'.php');

}else{ header("location:../");

} }else{ header("location:../");

}

?>

<script src="assets/js/footer-mod.js"></script>

Dashboard.php

<?php require\_once('../const/user\_dashboard.php'); function number\_abbr($number)

{

$abbrevs = [12 => 'T', 9 => 'B', 6 => 'M', 3 => 'K', 0 => '']; foreach ($abbrevs as $exponent => $abbrev) { if (abs($number) >= pow(10, $exponent)) {

$display = $number / pow(10, $exponent);

$decimals = ($exponent >= 3 && round($display) < 100) ? 1 : 0; $number = number\_format($display, $decimals).$abbrev; break; } } return $number;

}

?>

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<title><?php echo WBName; ?> - Dashboard</title>

<meta name="description" content="Insurance Management System">

<meta name="author" content="Bwire Mashauri">

<base href="../">

<link rel="icon" href="assets/media/favicons/favicon.ico" type="image/x-icon">

<link rel="stylesheet" id="css-main" href="assets/css/dashmix.min-5.4.css">

<link type="text/css" rel="stylesheet" href="assets/loader/waitMe.css">

<?php if (Theme !== "") { print '<link rel="stylesheet" id="css-theme" href="assets/css/themes/'.Theme.'">'; } ?>

</head>

<body>

<div id="page-container" class="<?php echo Header; ?> <?php echo PageHeader; ?> <?php echo Header; ?> <?php echo MainContent; ?>">

<?php require\_once('pages/main\_nav.php'); ?>

</div>

</div>

<div class="content">

<div class="alert alert-info d-flex align-items-center" role="alert">

<div class="flex-shrink-0">

<i class="fa fa-fw fa-info-circle"></i>

</div>

<div class="flex-grow-1 ms-3">

<p class="mb-0">

<?php

$h = date('G'); if($h>=5 && $h<=11)

{ echo "Good morning ".$fname."";

} else if($h>=12 && $h<=15)

{

echo "Good afternoon ".$fname."";

} else { echo "Good evening ".$fname."";

}

?>! Welcome back.

</p>

</div>

</div>

<div class="row">

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=history">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person fa-3x text-primary"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($policy\_holders); ?></div>

<div class="text-muted">My Policy Applications</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=pending\_applications">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person-circle-question fa-3x text-warning"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($pending\_policy\_holders); ?></div>

<div class="text-muted">My Pending Policy Applications</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=approved\_applications"> <div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person-circle-check fa-3x text-success"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($active\_policy\_holders); ?></div>

<div class="text-muted">My Approved Policy Applications</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=denied\_applications">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-person-circle-xmark fa-3x text-danger"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($denied\_policy\_holders); ?></div>

<div class="text-muted">My Denied Policy Applications</div>

</div>

</div>

</a>

</div>

</div>

<div class="row">

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=history2">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file fa-3x text-primary"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($tickets); ?></div>

<div class="text-muted">My Support Tickets</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=pending\_tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file-circle-question fa-3x text-warning"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($pending\_tickets); ?></div>

<div class="text-muted">My Submitted Tickets</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=active\_tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file-circle-check fa-3x text-success"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($active\_tickets); ?></div>

<div class="text-muted">My Active Tickets</div>

</div>

</div>

</a>

</div>

<div class="col-md-3">

<a class="block block-rounded block-link-shadow" href="user?page=closed\_tickets">

<div class="block-content block-content-full">

<div class="py-4 text-center">

<div class="mb-3">

<i class="fa fa-file-circle-xmark fa-3x text-danger"></i>

</div>

<div class="fs-4 "><?php echo number\_abbr($closed\_tickets); ?></div>

<div class="text-muted">My Closed Tickets</div>

</div>

</div>

</a>

</div>

</div>

</div>

</main>

</div>

<script src="assets/js/lib/jquery.min.js"></script>

<script src="assets/js/dashmix.app.min-5.4.js"></script>

<script src="assets/loader/waitMe.js"></script>

<script src="assets/js/forms.js"></script>

<script src="assets/js/footer-mod.js"></script>

</body>

</html>

##### Apply.php

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<title><?php echo WBName; ?> - Apply Insurance Policy</title>

<meta name="description" content="Insurance Management System">

<meta name="author" content="Bwire Mashauri">

<base href="../">

<link rel="icon" href="assets/media/favicons/favicon.ico" type="image/x-icon">

<link rel="stylesheet" id="css-main" href="assets/css/dashmix.min-5.4.css">

<link rel="stylesheet" href="assets/js/plugins/datatablesbs5/css/dataTables.bootstrap5.min.css">

<link rel="stylesheet" href="assets/js/plugins/datatables-buttonsbs5/css/buttons.bootstrap5.min.css">

<link rel="stylesheet" href="assets/js/plugins/datatables-responsivebs5/css/responsive.bootstrap5.min.css">

<link type="text/css" rel="stylesheet" href="assets/loader/waitMe.css">

<?php if (Theme !== "") { print '<link rel="stylesheet" id="css-theme" href="assets/css/themes/'.Theme.'">'; } ?>

</head>

<body>

<div id="page-container" class="<?php echo Header; ?> <?php echo PageHeader; ?> <?php echo Header; ?> <?php echo MainContent; ?>">

<?php require\_once('pages/main\_nav.php'); ?>

</div>

</div>

<div class="content">

<?php require\_once('../const/check-reply.php'); ?>

<div class="block block-rounded">

<div class="block block-rounded">

<div class="block-header block-header-default">

<h3 class="block-title">Apply Insurance Policy</h3>

</div>

<div class="block-content">

<table class="table table-bordered table-vcenter js-dataTable-buttons">

<thead>

<tr>

<th>Name</th>

<th>Category</th>

<th>Sub-Category</th>

<th>Sum Assured</th>

<th>Premium</th>

<th>Tenture</th>

<th style="width: 5%;">Apply</th>

</tr>

</thead>

<tbody>

<?php

try {

$conn = new

PDO('mysql:host='.DBHost.';dbname='.DBName.';charset='.DBCharset.';collation='.DBColla tion.';prefix='.DBPrefix.'', DBUser, DBPass);

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$stmt = $conn->prepare("SELECT \* FROM tbl\_insuarance\_policy LEFT JOIN tbl\_insuarance\_category ON tbl\_insuarance\_policy.category = tbl\_insuarance\_category.id LEFT JOIN tbl\_insuarance\_sub\_category ON tbl\_insuarance\_policy.sub\_category = tbl\_insuarance\_sub\_category.id WHERE tbl\_insuarance\_policy.status = '1' AND tbl\_insuarance\_category.status = '1' AND tbl\_insuarance\_sub\_category.status = '1' ORDER BY tbl\_insuarance\_policy.name");

$stmt->execute();

$result = $stmt->fetchAll(); foreach($result as $row)

{

?>

<tr>

<td class="">

<?php echo $row[1]; ?>

</td>

<td class="">

<?php echo $row[9]; ?>

</td>

<td class="">

<?php echo $row[13]; ?>

</td>

<td class="" align="right">

<?php echo number\_format($row[4]); ?> <?php echo WBCurrency; ?>

</td>

<td class="" align="right">

<?php echo number\_format($row[5]); ?> <?php echo WBCurrency; ?>

</td>

<td class="">

<?php echo $row[6]; ?> Month(s)

</td>

<td class="text-center">

<div class="btn-group">

<a class="btn btn-sm btn-alt-secondary js-bs-tooltip-enabled" onclick="return confirm('Are you sure you want to apply?');" href="user/core/apply-policy?id=<?php echo $row[0]; ?>"> Apply

</a>

</div>

</td>

</tr>

<?php

}

}catch(PDOException $e)

{ echo "Connection failed: " . $e->getMessage();

}

?>

</tbody>

</table>

</div>

</div>

</div>

</div>

</main>

</div>

<script src="assets/js/lib/jquery.min.js"></script>

<script src="assets/js/dashmix.app.min-5.4.js"></script>

<script src="assets/loader/waitMe.js"></script>

<script src="assets/js/forms.js"></script>

<script src="assets/js/plugins/datatables/jquery.dataTables.min.js"></script>

<script src="assets/js/plugins/datatables-bs5/js/dataTables.bootstrap5.min.js"></script>

<script src="assets/js/plugins/datatablesresponsive/js/dataTables.responsive.min.js"></script>

<script src="assets/js/plugins/datatables-responsivebs5/js/responsive.bootstrap5.min.js"></script>

<script src="assets/js/plugins/datatables-buttons/dataTables.buttons.min.js"></script>

<script src="assets/js/plugins/datatables-buttons-bs5/js/buttons.bootstrap5.min.js"></script>

<script src="assets/js/plugins/datatables-buttons-jszip/jszip.min.js"></script>

<script src="assets/js/plugins/datatables-buttons-pdfmake/pdfmake.min.js"></script>

<script src="assets/js/plugins/datatables-buttons-pdfmake/vfs\_fonts.js"></script>

<script src="assets/js/plugins/datatables-buttons/buttons.print.min.js"></script>

<script src="assets/js/plugins/datatables-buttons/buttons.html5.min.js"></script>

<script src="assets/js/pages/be\_tables\_datatables.min.js"></script>

<script src="assets/js/footer-mod.js"></script>

</body>

</html>

##### Login.php

<?php

session\_start(); require\_once('db/config.php'); require\_once('const/organization.php'); require\_once('const/check\_session.php'); if ($res == 1) { switch ($level) { case '0':

header("location:admin?page=dashboard"); break; case '1':

header("location:staff?page=dashboard"); break; default:

header("location:user?page=dashboard");

}

}

?>

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<title><?php echo WBName; ?> - Authentication</title>

<meta name="description" content="Insurance Management System">

<link rel="icon" href="assets/media/favicons/favicon.ico" type="image/x-icon">

<link rel="stylesheet" id="css-main" href="assets/css/dashmix.min-5.4.css">

<?php if (Theme !== "") { print '<link rel="stylesheet" id="css-theme" href="assets/css/themes/'.Theme.'">'; } ?>

<link type="text/css" rel="stylesheet" href="assets/loader/waitMe.css">

</head>

<body>

<div id="page-container">

<main id="main-container">

<div class="bg-image">

<div class="row g-0 justify-content-center bg-primary-dark-op">

<div class="hero-static col-sm-8 col-md-6 col-xl-4 d-flex align-items-center p-2 px-sm-0">

<div class="block block-transparent block-rounded w-100 mb-0 overflow-hidden"> <div class="block-content block-content-full px-lg-5 px-xl-6 py-4 py-md-5 py-lg-6 bg-bodyextra-light">

<div class="mb-2 text-center">

<img style="height:70px; object-fit: cover !important; margin-bottom:20px;" src="assets/media/logo/<?php echo WBLogo; ?>" alt=""><br>

<a class="link-fx fw-bold fs-2" href="./">

<span class="text-primary"><?php echo WBName; ?></span>

</a>

<p class="text-uppercase fw-bold fs-sm text-muted">Insurance Management System</p> </div>

<?php require\_once('const/check-reply.php'); ?>

<form id="app\_frm" method="POST" autocomplete="OFF" action="core/auth">

<div class="mb-4">

<div class="input-group input-group-lg">

<input type="email" class="form-control form-control-alt" required name="username" placeholder="Enter your email"> <span class="input-group-text">

<i class="fa fa-user-circle"></i>

</span>

</div>

</div>

<div class="mb-4">

<div class="input-group input-group-lg">

<input type="password" class="form-control form-control-alt" required name="password" placeholder="Enter your password">

<span class="input-group-text">

<i class="fa fa-asterisk"></i>

</span>

</div>

</div>

<div class="d-sm-flex justify-content-sm-between align-items-sm-center text-sm-start mb-4">

<div class="form-check">

<input type="checkbox" class="form-check-input" name="remember">

<label class="form-check-label" for="login-remember-me">Remember me for two weeks</label>

</div></div>

<input type="hidden" name="submit" value="1">

<div class="text-center mb-2">

<button id="sub\_btn" type="submit" class="btn btn-primary">

<i class="fa fa-fw fa-sign-in-alt opacity-50 me-1"></i> Sign In

</button>

<p class="mt-2 mb-0 d-lg-flex justify-content-lg-between">

<a class="btn btn-sm btn-alt-secondary d-block d-lg-inline-block mb-1" href="?page=forgotpw">

<i class="fa fa-exclamation-triangle opacity-50 me-1"></i> Forgot password</a>

<a class="btn btn-sm btn-alt-secondary d-block d-lg-inline-block mb-1" href="?page=onboard">

<i class="fa fa-plus opacity-50 me-1"></i> New Account

</a></p>

</div>

</form>

</div></div>

</div></div>

</div>

</main>

</div>

<script src="assets/js/dashmix.app.min-5.4.js"></script>

<script src="assets/js/lib/jquery.min.js"></script>

<script src="assets/loader/waitMe.js"></script>

<script src="assets/js/forms.js"></script>

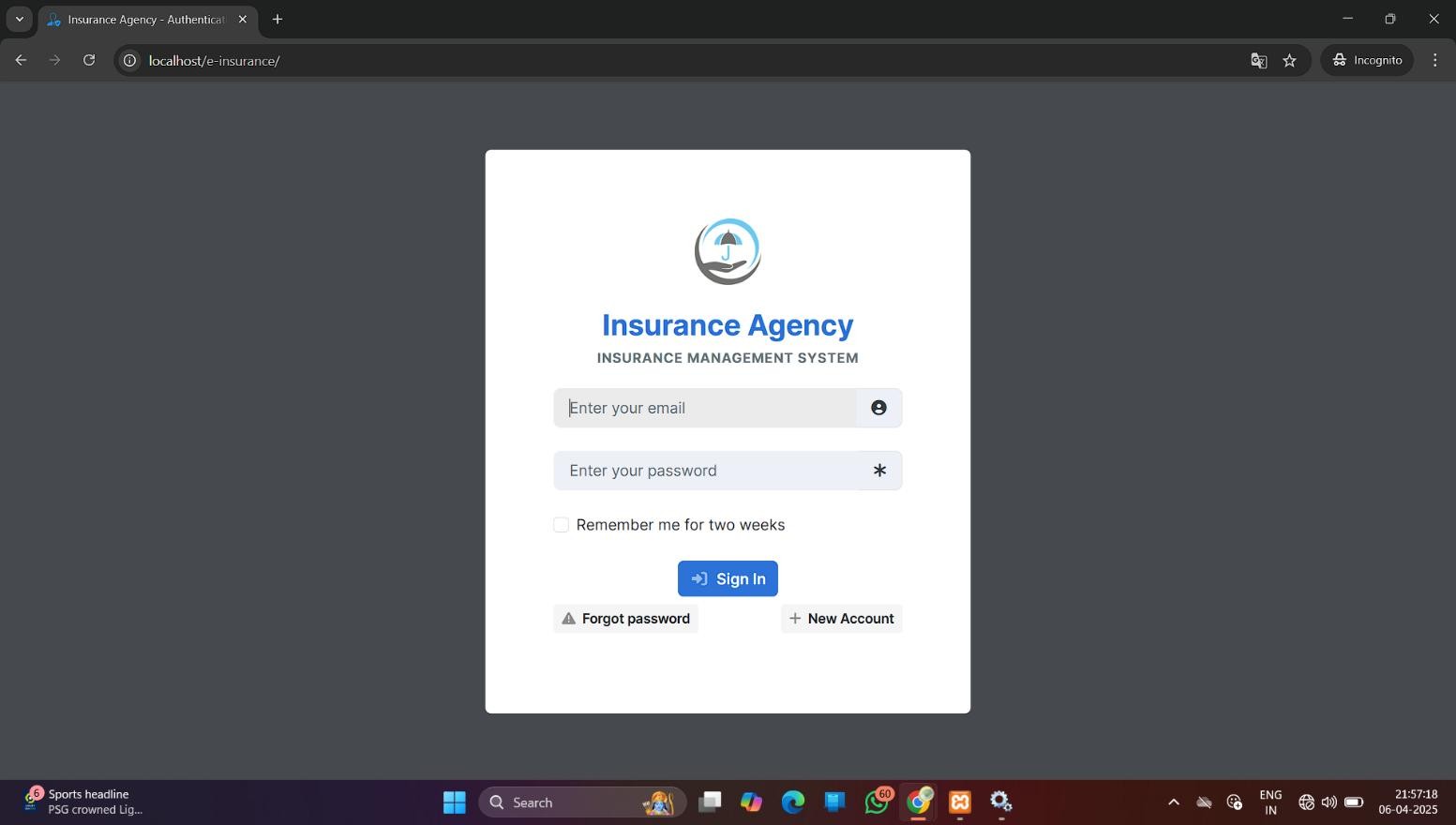
<script src="assets/js/cstm.js"></script>

</body></html>

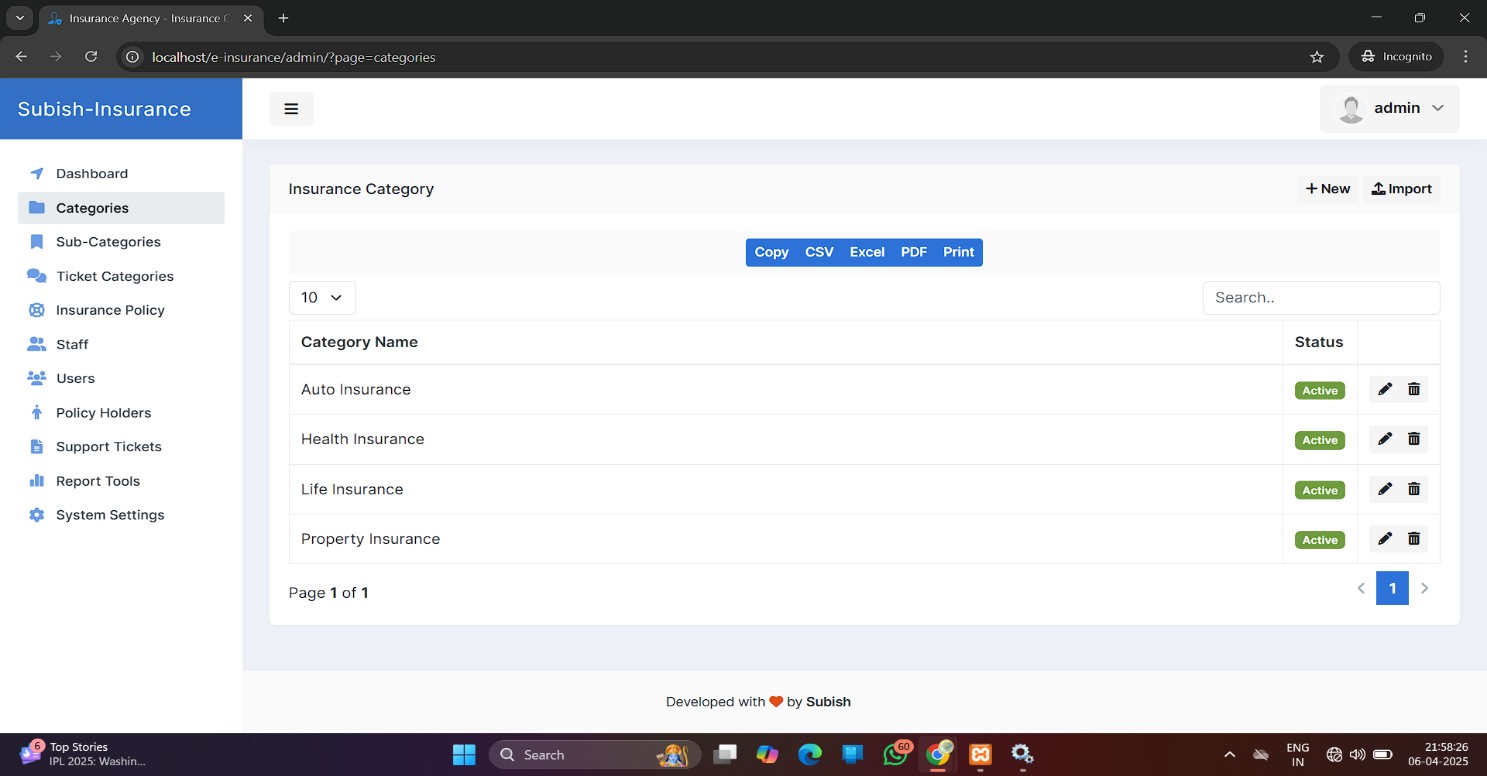
#### 12.2 SCREENSHOTS

**12.2.1 ADMIN PAGES :**

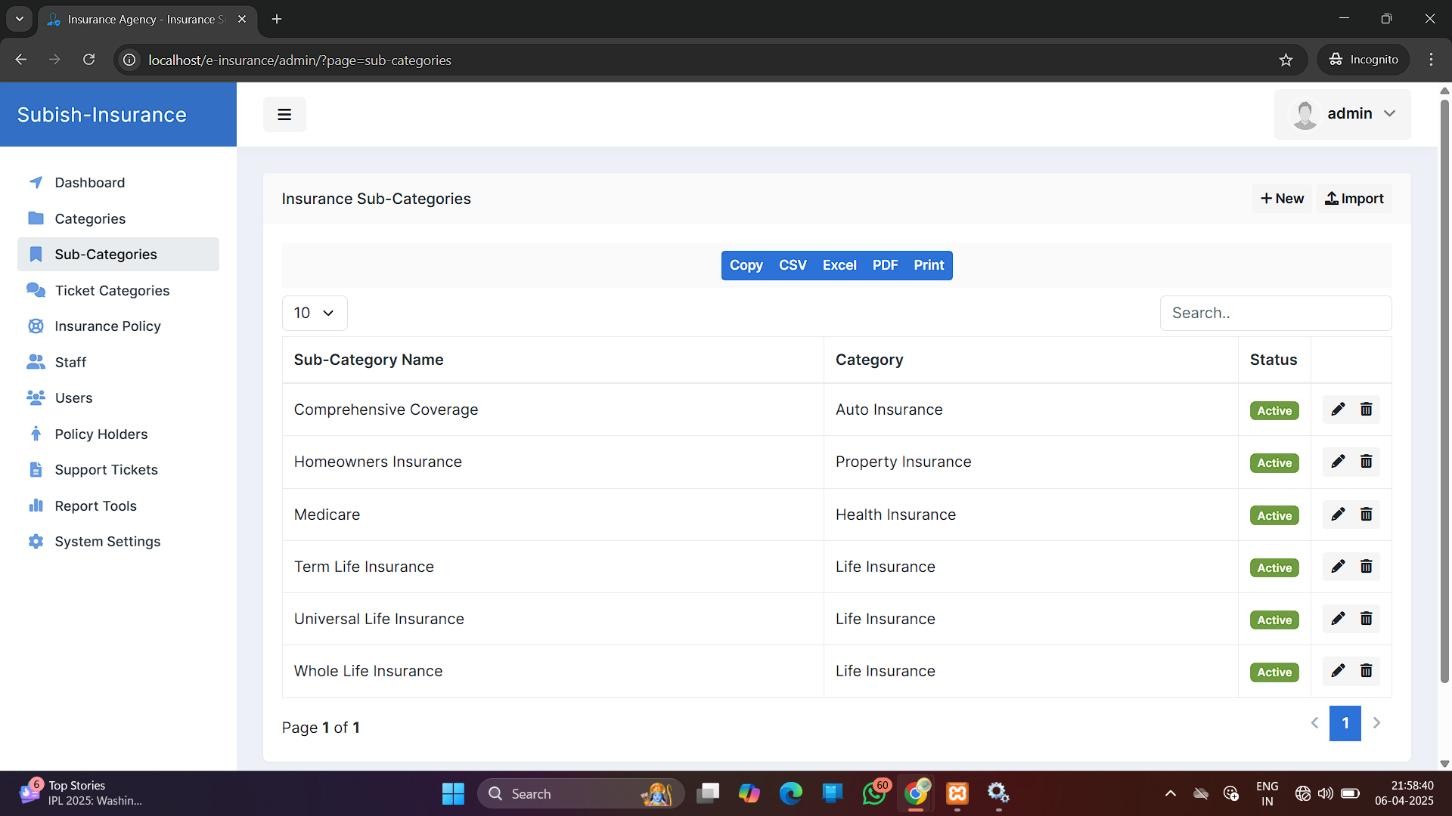
LOGINPAGE



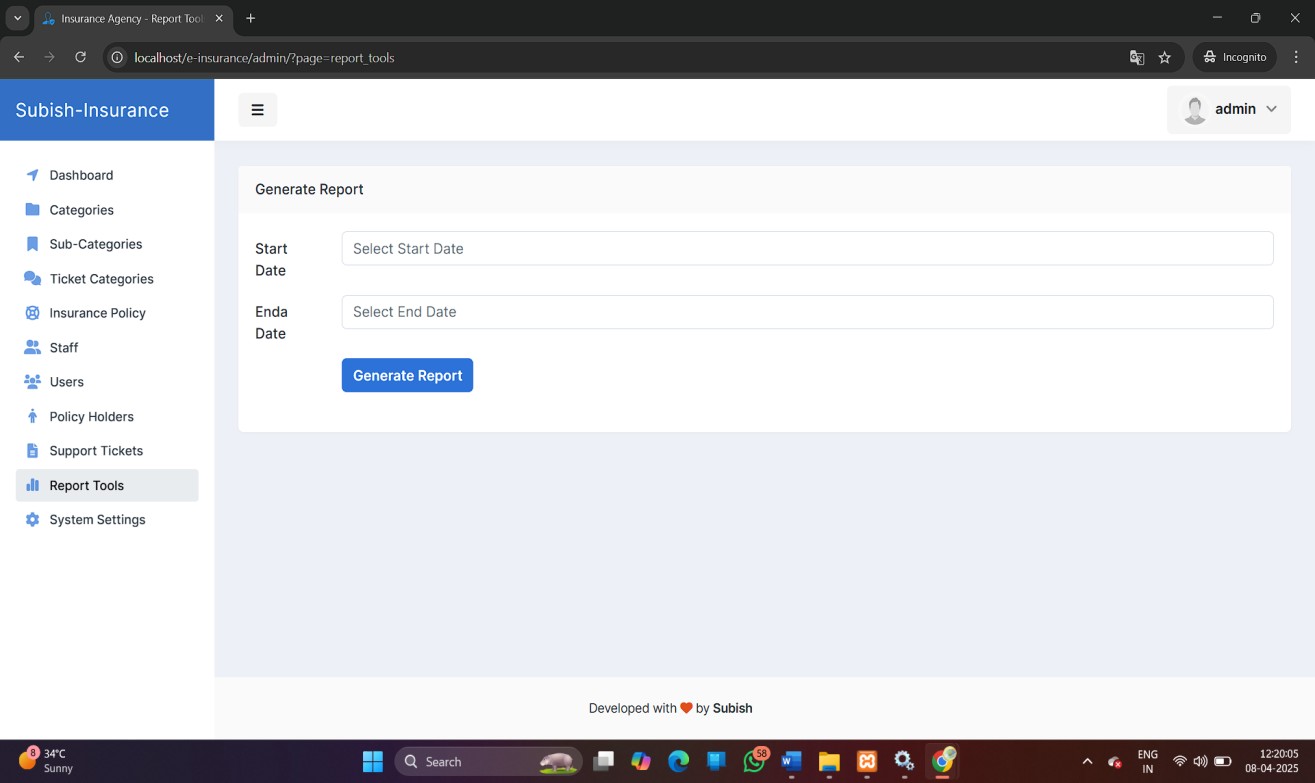
CATEGORIES



SUB-CATEGORIES

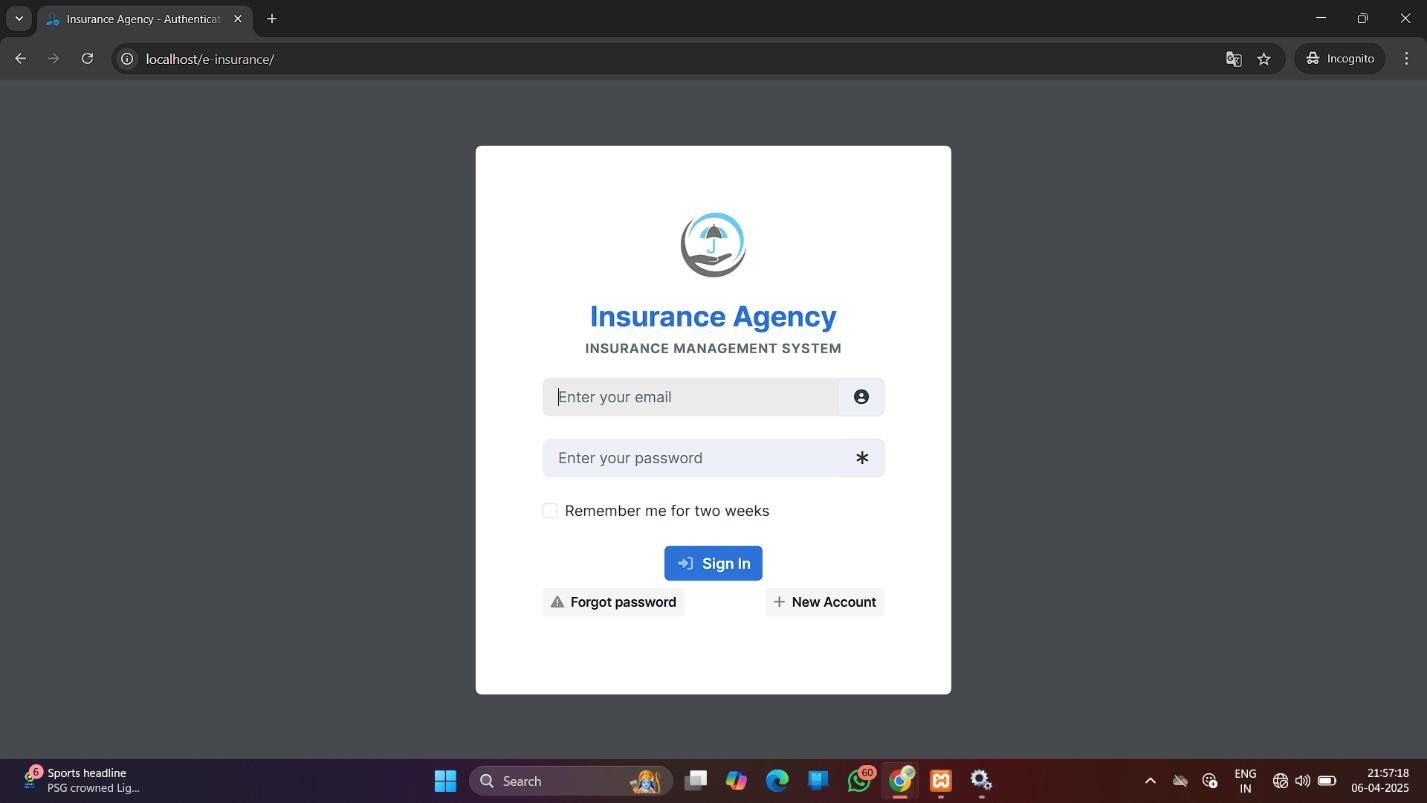


REPORT TOOLS

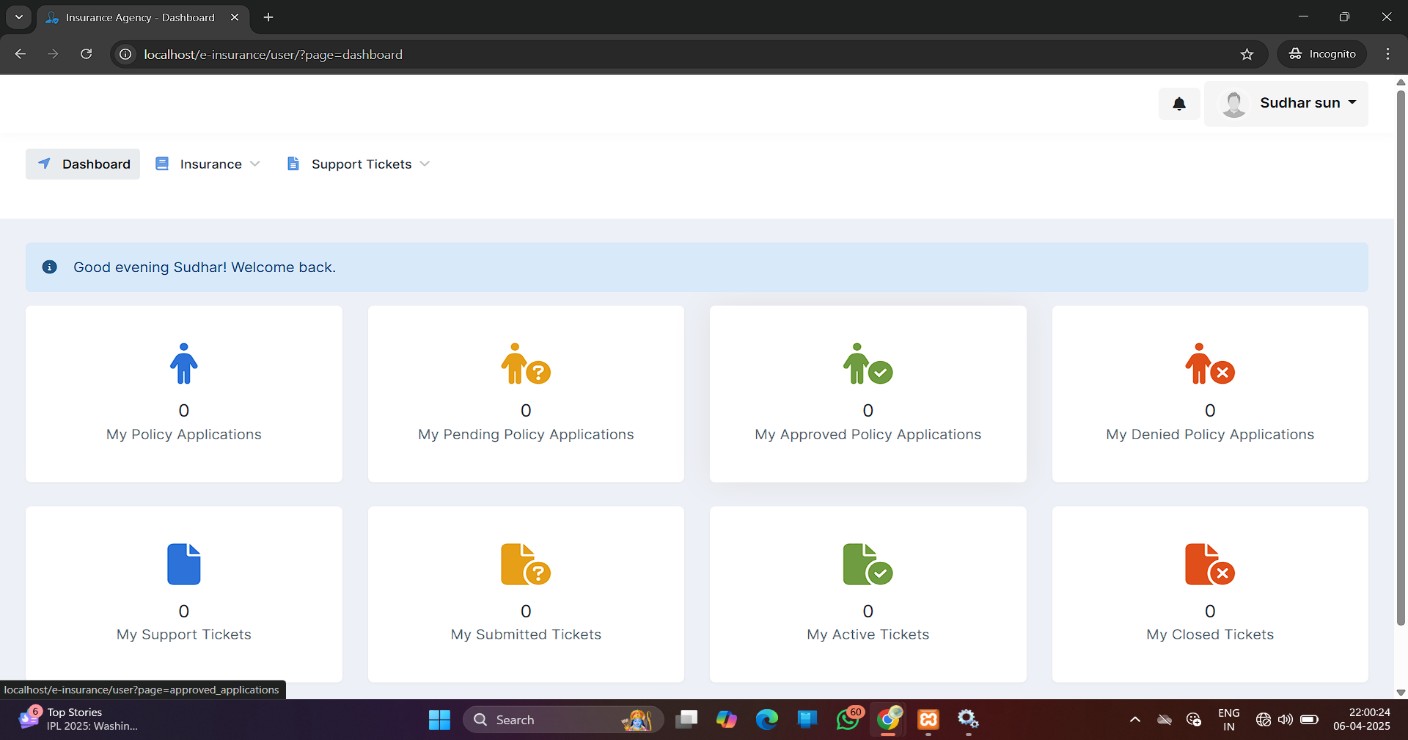


##### 13.2.2 USER PAGES

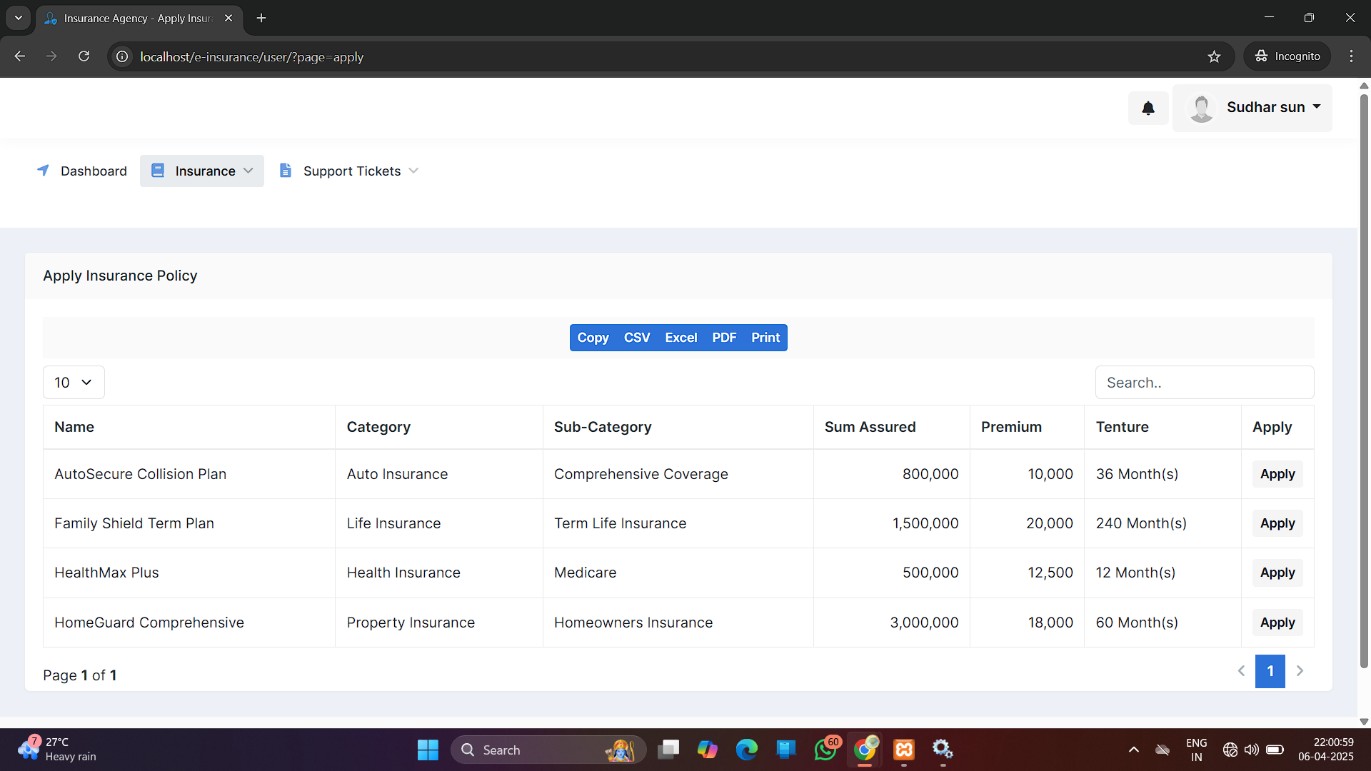
LOGIN PAGE



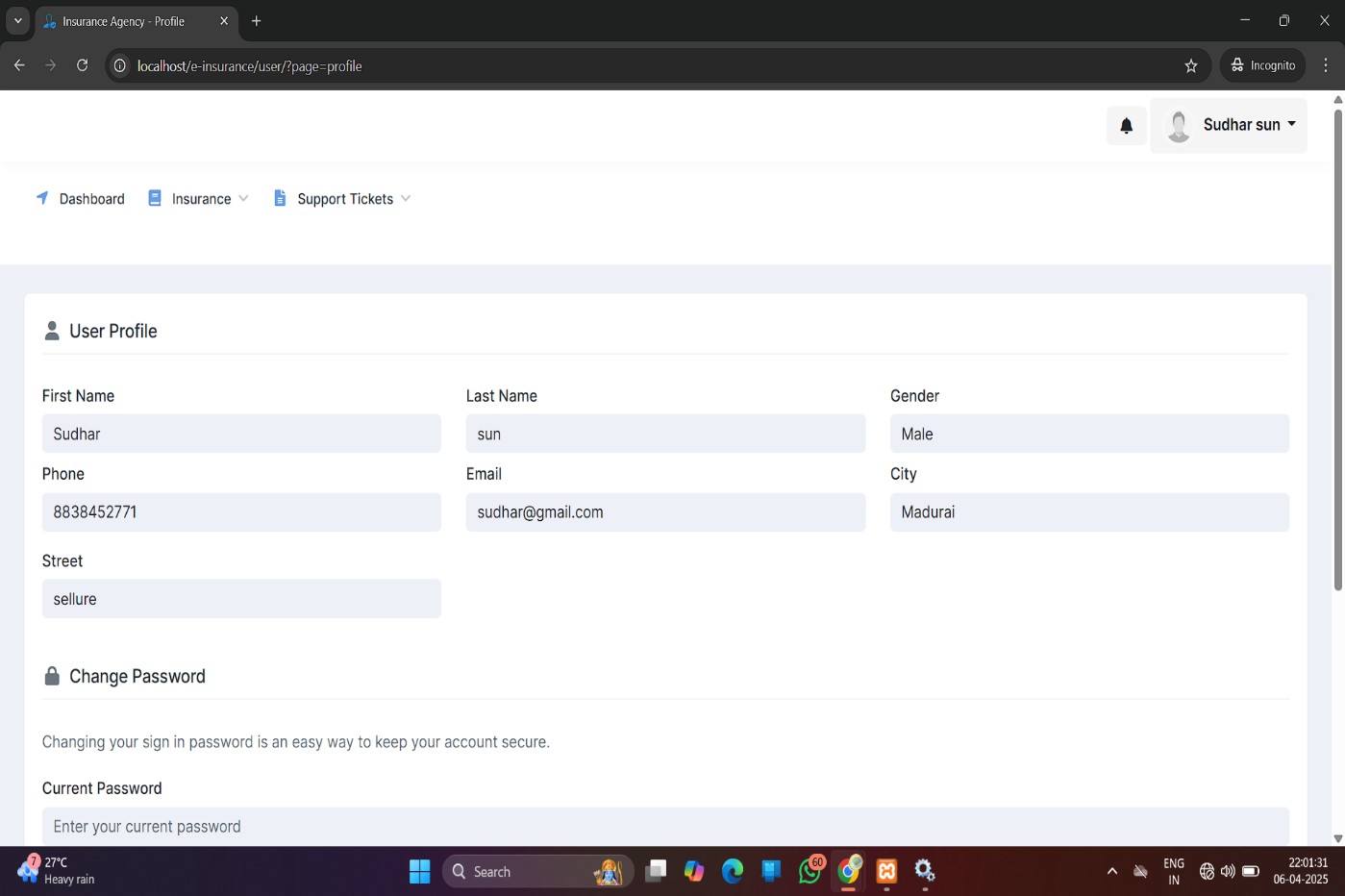
DASBOARD PAGE



INSURANCE -APPLY PAGE



USER PROFILE



### 13. BIBLIOGRAPHY

Deitel, P. J., & Deitel, H. M. (2013). *PHP Programming with MySQL*. Pearson Education. Welling, L., & Thomson, L. (2009). *PHP and MySQL Web Development* (4th ed.). Addison- Wesley.

Nixon, R. (2018). *Learning PHP, MySQL & JavaScript* (5th ed.). O’Reilly Media.

ISO. (2020). *ISO 27001 Information Security Management*. Retrieved from [https://www.iso.org](https://www.iso.org/)

PHP Manual. (n.d.). *PHP Documentation*. Retrieved from <https://www.php.net/manual/en/>

W3Schools. (n.d.). *PHP, MySQL, and HTML Tutorials*. Retrieved from <https://www.w3schools.com/>

GeeksforGeeks. (n.d.). *PHP Projects – Beginner to Advanced*. Retrieved from <https://www.geeksforgeeks.org/>

MySQL Documentation. (n.d.). Retrieved from <https://dev.mysql.com/doc/>

Stack Overflow. (n.d.). *Community Discussions and Solutions on PHP and MySQL*. Retrieved from <https://stackoverflow.com/>

TutorialsPoint. (n.d.). *PHP and MySQL Tutorials*. Retrieved from https://www.tutorialspoint.com/php/