**CSE 404L – Database Management Systems Lab**

****

**Spring 2025**

**Submitted by:**

**Khalil hussain(22pwcse2224)**

**Waqas atta(22pwcse2142)**

**Abdul Muqtadir (22pwcse2186)**

**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.”

Submitted to:

**Engr Sumayyea Salahudin**

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

# Project Insurance Management System

## CONCEPTUAL SCHEMA

**Entity Description Table**

* **USER**

|  |  |
| --- | --- |
| Attribute | Description |
| user-id | INT Primary Key uniquely identifies each user uniquely, usually auto-incremented |
| Name | VARCHAR(100) stores full name of the customer |
| Email | VARCHAR(150) stores unique email for communication |
| Phone | VARCHAR(20) stores contact number, allows international formats |
|  |  |

* **Policy**

|  |  |
| --- | --- |
| Attribute | Description |
| Policy-id | INT Primary Key uniquely identifies each policy |
| user \_id | INT Foreign key referencing User |
| policy name | VARCHAR(100)Name/title of the policy |
| Start-date | DATE policy activation date |
| End-date | DATE policy expiration date |

* **Claims**

|  |  |
| --- | --- |
| Attribute | Description |
| Claim-id | INT Primary Key uniquely identifies each claim |
| Policy-id | INT Foreign Key refers to the associated policy |
| date | DATE when claim was filed |
| Claim-amount | DECIMAL(10,2) requested amount in the claim |
| Status | ENUM('Pending', 'Approved', 'Rejected')  current state of claim |

* **Premiums**

|  |  |
| --- | --- |
| Attribute | Description |
| Premium-id | INT Primary Key uniquely identifies each |

|  |  |
| --- | --- |
|  | premium payment |
| Policy-id | INT Foreign Key refers to the associated policy |
| due date | DATE when for due date |
| Amount | DECIMAL(10,2) the actual paid premium amount |
| paid date | DATE when payment was paid |

* **Renewals**

|  |  |
| --- | --- |
| Attribute | Description |
| Renewal-id | INT Primary Key uniquely identifies each renewal record |
| Policy-id | INT Foreign Key to the policy being renewed |
| Renewal-date | DATE when renewal was done |
| Amount | DECIMAL(10,2) the actual paid premium amount |
| Status | status paid or unpaid |
|  |  |

**. Beneficiary**

|  |  |
| --- | --- |
| Attribute | Description |
| beneficiary-id | INT Primary Key uniquely identifies each renewal record |
| Policy-id | INT Foreign Key to the policy being  renewed |
| Name | VARCHAR(100) stores full name of the beneficiary |
| Relation | ARCHAR(100)Relationship to the policy holder |

* **Agent**

|  |  |
| --- | --- |
| Attribute | Description |
| Agent-id | INT Primary Key uniquely identifies each agent record |
| Name | VARCHAR(100) stores full name of the customer |
| Email | VARCHAR(150) stores unique email for communication |
| Phone | VARCHAR(20) stores contact number,  allows international formats |

* **Payment**

|  |  |
| --- | --- |
| Attribute | Description |
| payment-id | INT Primary Key uniquely identifies each agent record |
| user id | INT Foreign key referencing User |
| Method | Payment method (Cash, Card, etc.) |
| Date | DATE when payment was filed |
| Amount | DECIMAL(10,2) the actual paid premium amount |

* **Feedback**

|  |  |
| --- | --- |
| Attribute | Description |
| Feedback-id | INT Primary Key uniquely identifies each Feedback record |
| User\_Id | INT Foreign key referencing User |
| Date | DATE when payment was filed |
| Message | VARCHAR(100) Text feedback from the user, |
| Rating | rating provided by the user |

* **Entity Relationships**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Table Name | Related Tables & Relationship Types | Description |
| 1 | Customers | Policies (1:M), Claims (1:M) | One customer can have multiple  policies and can file multiple claims. |
| 2 | Policies | Customers (M:1), Claims (1:M), Premiums (1:M), Renewals (1:M) | Each policy belongs to one customer and can have multiple claims, premium payments, and  renewals. |
| 3 | Claims | Policies (M:1) | Each claim is linked to one policy. |
| 4 | Premiums | Policies (M:1) | Each premium is  linked to one policy. |
| 5 | Renewals | Policies (M:1) | Each renewal record  is associated with one policy. |

# Business Rules for Insurance Management System

## User & Policy

* Each user can register for one or more policies.
* Each policy must be associated with exactly one user.

## Policy & Agent

* A policy can be assigned to one or more agents.
* An agent can be responsible for one or more policies

## Policy & Claims

* A claim can only be made against a registered policy.
* A policy can have multiple claims, but each claim belongs to only one policy.

## Policy & Beneficiary

* A policy may have multiple beneficiaries.
* A beneficiary must be linked to a valid policy.

.

## Policy & Renewal

* A policy can have multiple renewal records.
* Each renewal must record the date, amount, and status (Paid/Unpaid).

## Policy & Premium

* A premium record must be created for each policy payment schedule.
* Each premium entry must have a due date, paid date, and amount.

## User & Payment

* A user may make multiple general payments (e.g., registration fees, charges).
* Each payment must include an amount, date, and payment method (cash, card, etc.).

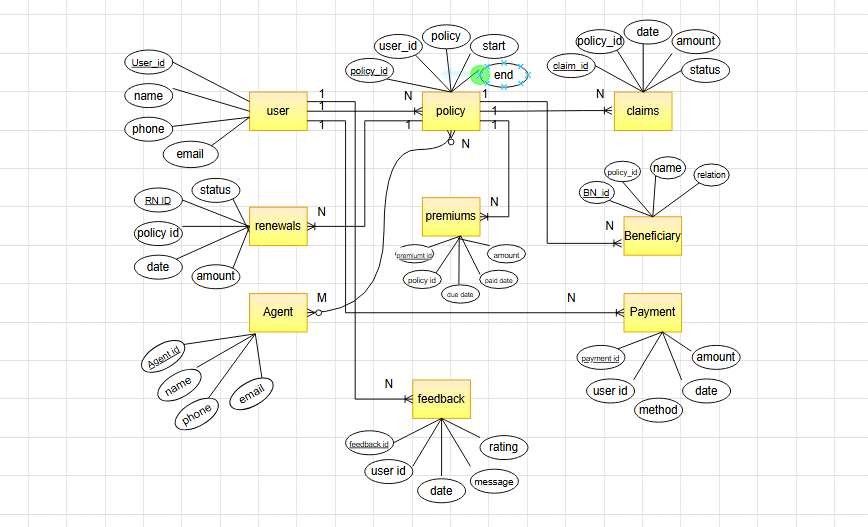
## User & Feedback

* A user may submit multiple feedback entries.
* Each feedback must include a date, a rating (1–5), and an optional message.

1. **Relationships & Cardinality**

|  |  |  |
| --- | --- | --- |
| Relationship | Type | Description |
| User → Policy | 1:M | One user can have multiple policies. |
|  |  |  |
| Policy→ claim | 1:M | Each policy can have multiple claim. |
| Policy → premium | 1:M | Each policy can have multiple premium |
| Policy → Agent | M:N | many agent can have multiple agent |
| policy→ Beneficiary | 1:M | Each policy is linked to one beneficiary. |
| Policy → Renewal | 1:M | Each policy is linked to many REnewal. |
| User→ payment | 1:M | Each user is linked to many payment. |
| User→ feedback | 1:M | Each user give many feedback. |

1. **ER Diagram (Conceptual)**



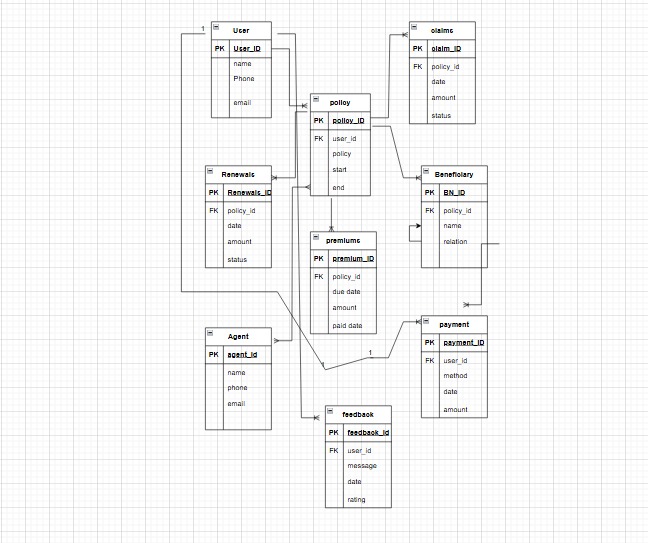
1. **Tools Used**
   * ChatGPT for help
   * Microsoft Word for documentation
   * Draw.io for ER diagram creation

**Milestone 02 – Conceptual Schema and Normalization**

**1. Relationship Table**

| **Relationship** | **Type** | **Rule Explanation** |
| --- | --- | --- |
| Users — Policies | 1 : M | One user (Users.user\_id) can register for multiple policies |
| Policies — Claims | 1 : M | One policy (Policy.policy\_id) can have multiple claims |
| Policies — Premiums | 1 : M | One policy can have many premium payments |
| Policies — Renewals | 1 : M | A policy can be renewed many times |
| Policies — Beneficiaries | 1 : M | One policy may have multiple beneficiaries |
| Policies — Agents | M : N | One policy can be handled by multiple agents; one agent can handle many policies |
| Users — Payments | 1 : M | A user can make many general payments |
| Users — Feedback | 1 : M | One user can submit multiple feedback entries |
| Claims — Policies | M : 1 | Each claim belongs to a registered policy |

**2. Relational Schemax!**



**USER: (1:M )** UserID name email phone

**POLICY: (1:M)** PolicyID UserID policy\_name start\_date end\_date

**CLAIM: (M:1)** ClaimID PolicyID date claim\_amount status

**PREMIUM: (1:M)** PremiumID PolicyID due\_date amount paid\_date

**RENEWAL: (1:M)** RenewalID PolicyID renewal\_date amount status

**BENEFICIARY: (1:M)** BeneficiaryID PolicyID name relation

**AGENT: (M:N)** AgentID name email phone

**PAYMENT: (1:M)** PaymentID UserID method date amount

**FEEDBACK: (1:M)** FeedbackID UserID date message rating

**Normalization Summary**

1NF: Each attribute is atomic (no multi-valued fields like multiple phone numbers or relations).

2NF: All non-key fields depend on the entire primary key.

3NF: No transitive dependencies exist.

, **1. Introduction**

The Insurance and Claim Management System is developed to streamline insurance-related processes, including policy registration, premium tracking, beneficiary handling, renewal management, and claims processing. This document outlines the complete database design, including entities, attributes, relationships, metadata, constraints, keys, and relational integrity.

**2. Entities (Tables)**

* User
* Agent
* Policy
* Claim
* Beneficiary
* Renewals
* Premiums
* Feedback
* Payment

**3. Table Structures and Metadata**

**Table: User**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| user\_id | INT | PRIMARY KEY, NOT NULL |
| name | VARCHAR(50) | NOT NULL |
| phone | VARCHAR(15) | NOT NULL |
| email | VARCHAR(50) | UNIQUE, NOT NULL |
| status | VARCHAR(20) | CHECK (status IN ('Active','Inactive')) |

**Table: Agent**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| agent\_id | INT | PRIMARY KEY, NOT NULL |
| name | VARCHAR(50) | NOT NULL |
| phone | VARCHAR(15) |  |
| email | VARCHAR(50) |  |

**Table: Policy**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| policy\_id | INT | PRIMARY KEY, NOT NULL |
| policy\_name | VARCHAR(50) | NOT NULL |
| start | DATE | NOT NULL |
| end | DATE | NOT NULL |
| user\_id | INT | FOREIGN KEY REFERENCES User(user\_id) |
| agent\_id | INT | FOREIGN KEY REFERENCES Agent(agent\_id) |

**Table: Claim**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| claim\_id | INT | PRIMARY KEY, NOT NULL |
| policy\_id | INT | FOREIGN KEY REFERENCES Policy(policy\_id) |
| date | DATE | NOT NULL |
| amount | DECIMAL(10,2) | NOT NULL |
| status | VARCHAR(20) | CHECK (status IN ('Pending','Approved','Rejected')) |

**Table: Beneficiary**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| BN\_id | INT | PRIMARY KEY, NOT NULL |
| policy\_id | INT | FOREIGN KEY REFERENCES Policy(policy\_id) |
| name | VARCHAR(50) | NOT NULL |
| relation | VARCHAR(30) | NOT NULL |

**Table: Renewals**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| RN\_id | INT | PRIMARY KEY, NOT NULL |
| policy\_id | INT | FOREIGN KEY REFERENCES Policy(policy\_id) |
| date | DATE | NOT NULL |
| amount | DECIMAL(10,2) | NOT NULL |
| status | VARCHAR(20) | CHECK (status IN ('Paid','Pending')) |

**Table: Premiums**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| premium\_id | INT | PRIMARY KEY, NOT NULL |
| policy\_id | INT | FOREIGN KEY REFERENCES Policy(policy\_id) |
| due\_date | DATE | NOT NULL |
| paid\_date | DATE |  |
| amount | DECIMAL(10,2) | NOT NULL |

**Table: Feedback**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| feedback\_id | INT | PRIMARY KEY, NOT NULL |
| user\_id | INT | FOREIGN KEY REFERENCES User(user\_id) |
| date | DATE | NOT NULL |
| rating | INT | CHECK (rating BETWEEN 1 AND 5) |
| message | TEXT |  |

**Table: Payment**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Constraints |
| payment\_id | INT | PRIMARY KEY, NOT NULL |
| user\_id | INT | FOREIGN KEY REFERENCES User(user\_id) |
| amount | DECIMAL(10,2) | NOT NULL |
| date | DATE | NOT NULL |
| method | VARCHAR(30) | CHECK (method IN ('Cash','Card','Online')) NOT NULL |

**4. Keys and Relationships**

**Primary Keys**

* user\_id for User
* agent\_id for Agent
* policy\_id for Policy
* claim\_id for Claim
* BN\_id for Beneficiary
* RN\_id for Renewals
* premium\_id for Premiums
* feedback\_id for Feedback
* payment\_id for Payment

**Foreign Keys**

* policy.user\_id → User(user\_id)
* policy.agent\_id → Agent(agent\_id)
* claim.policy\_id → Policy(policy\_id)
* beneficiary.policy\_id → Policy(policy\_id)
* renewals.policy\_id → Policy(policy\_id)
* premium.policy\_id → Policy(policy\_id)
* feedback.user\_id → User(user\_id)
* payment.user\_id → User(user\_id)

**Relationships**

* One User → Many Policies 1:N
* One Agent → Many Policies 1:N
* One Policy → Many Claims 1:N
* One Policy → Many Premiums 1:N
* One Policy → Many Beneficiaries 1:N
* One Policy → Many Renewals 1:N
* One User → Many Payments 1:N
* One User → Many Feedbacks 1:N

**5. Sample Data:**











**Insurance Management System**

(Built with Laravel, MySQL, Tailwind CSS)

**Introduction:**

This project is a web-based Insurance Management System developed using the Laravel PHP framework.

It allows an insurance company to manage users, agents, policies, claims, payments, feedback, and more, with a secure admin panel and user-friendly interface.

**Objectives**

* To automate and digitize insurance operations.
* To provide a secure platform for managing policies, claims, and payments.
* To enable admin control over user roles and data.
* To offer a professional, responsive, and modern user experience.

**Technologies Used**

|  |  |
| --- | --- |
| Layer | Technology/Language |
| Frontend | Blade (Laravel), Tailwind CSS, JavaScript |
| Backend | PHP (Laravel 12 |
| Database | MySQL |
| Assets | Vite, npm |
| Hosting | Hostinger (shared hosting) |

**System Features**

**A. Authentication**

* User registration and login (Laravel Breeze)
* Role-based access: admin and user

**B. Admin Panel**

* Dashboard with statistics and charts
* User management (edit, delete, promote/demote to admin)
* Approve/reject admin access requests

**C. Entities Managed**

* “**Users:**” Register, login, request admin access
* “**Agents**:” CRUD (Create, Read, Update, Delete)
* “**Policies**:” CRUD, assign to users/agents
* “**Premiums**:” CRUD, link to policies
* “**Claims**:” CRUD, status management
* “**Payments**:” CRUD, link to users
* “**Feedback**:” Users can submit feedback
* “**Beneficiaries**:” CRUD, link to policies
* “**Renewals**:” CRUD, link to policies

**D. Request Admin Access**

* Users can request admin rights
* Head admin can approve/reject requests

**E. Dashboard**

* Summary cards for all entities
* Interactive charts (e.g., claims by status, policies per agent)

**F. Professional UI**

* Responsive design with Tailwind CSS
* Clean navigation bar with role-based links
* Custom login/register pages with background image

**7. Key Code Features**

* **“Resource Controllers”** for all entities (CRUD)
* **“Middleware”** for authentication and admin protection
* **“Blade Components**” for layouts and navigation
* **“.env”** for environment configuration
* **“.htaccess”** for routing and security on shared hosting

**8. Deployment Steps**

1. Zip project (except `node\_modules`)

2. Upload to Hostinger and extract in `public\_html/your\_project\_folder`

3. Set document root to `/public`

4. Set up `.env` with Hostinger DB credentials

5. Import database via phpMyAdmin

6. Run `composer install` and `npm run build`

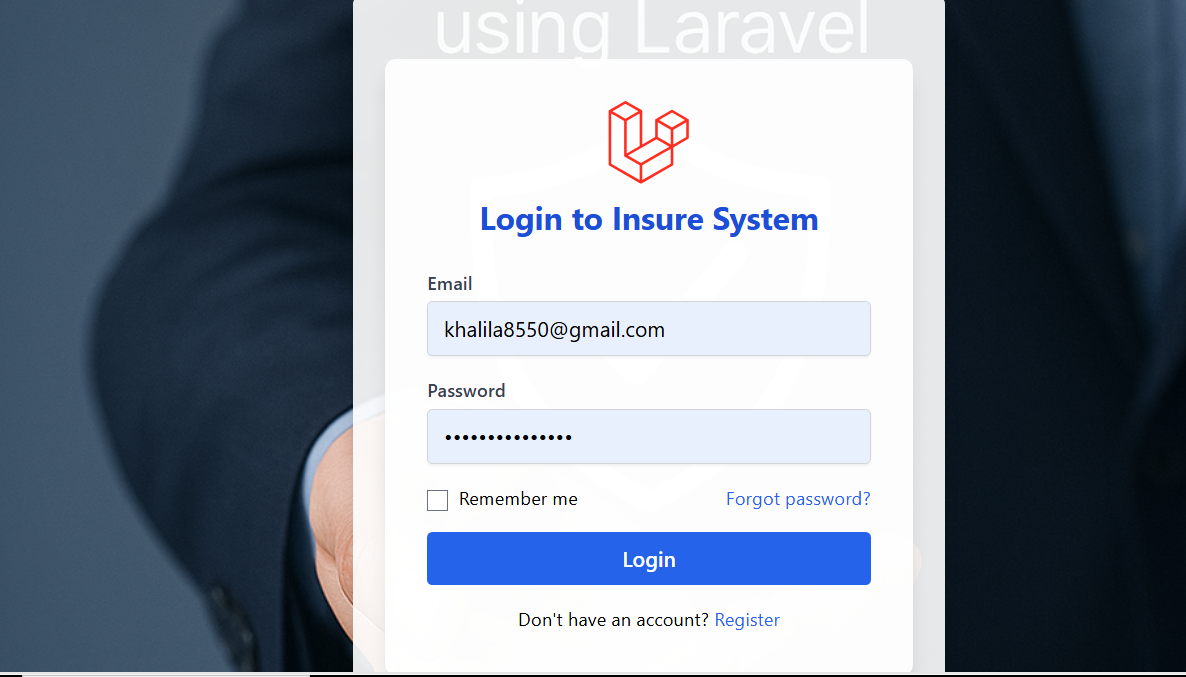
7. Set permissions for `storage` and `bootstrap/cache`

8. Clear Laravel cache

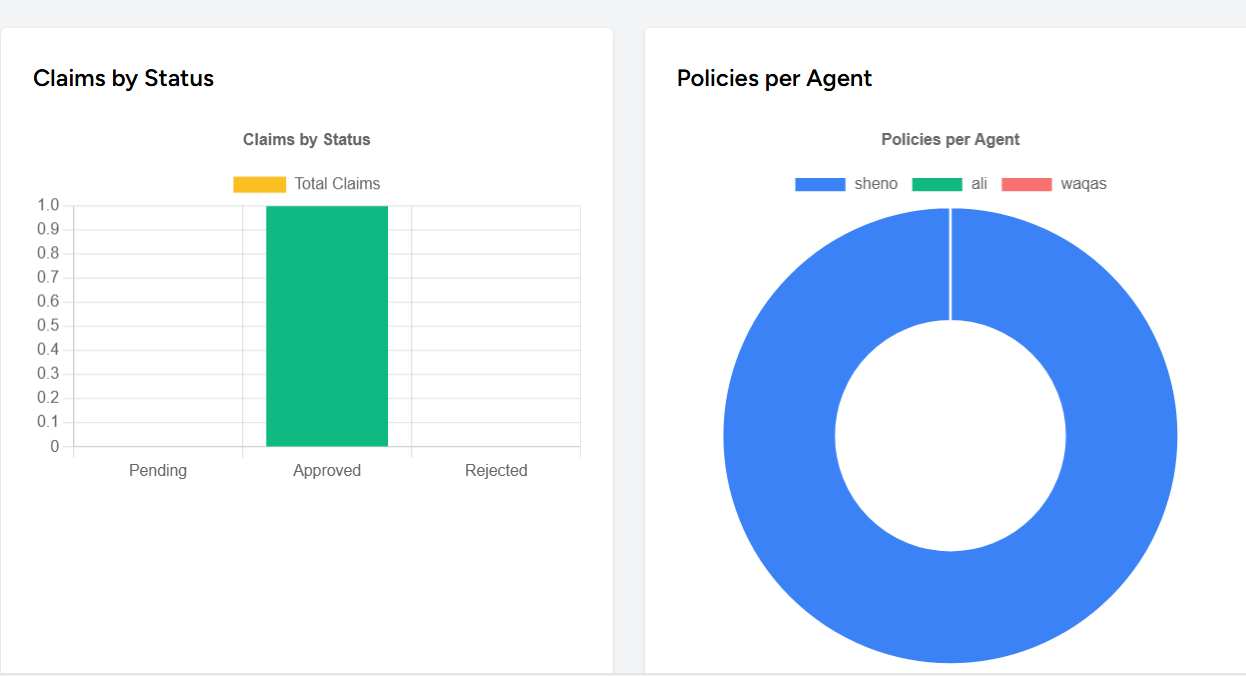
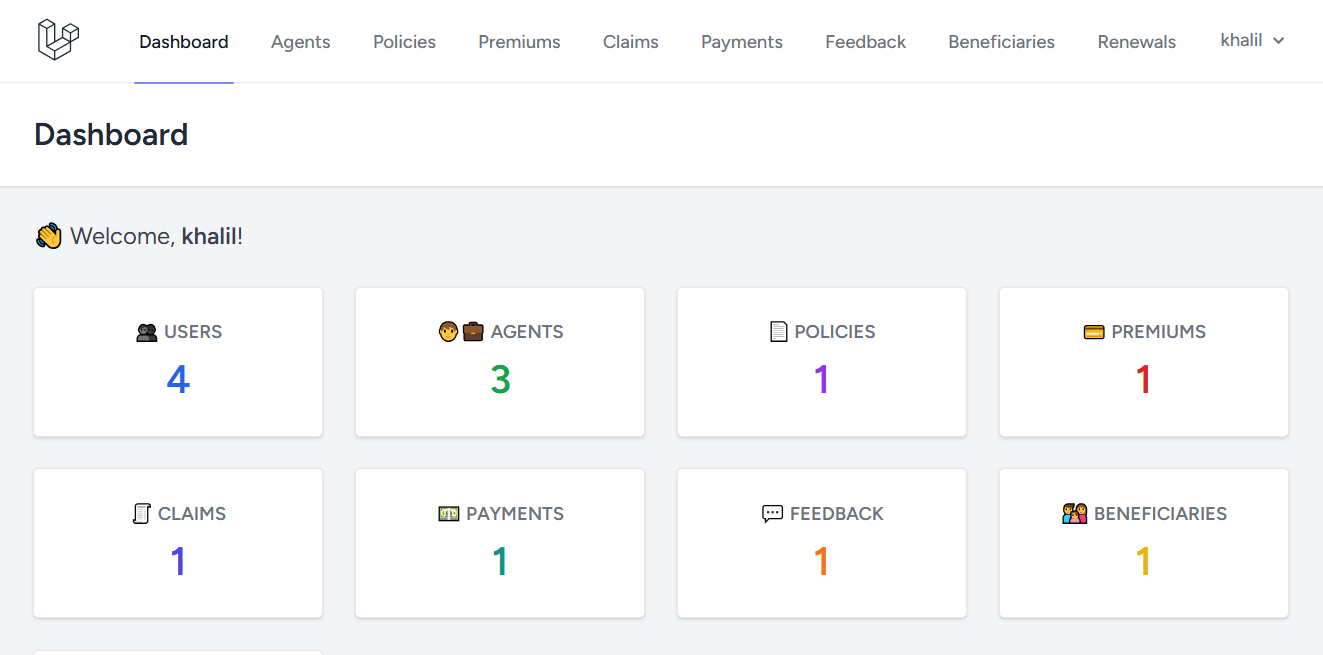
9. Visit your domain to access the app

**Screenshots**

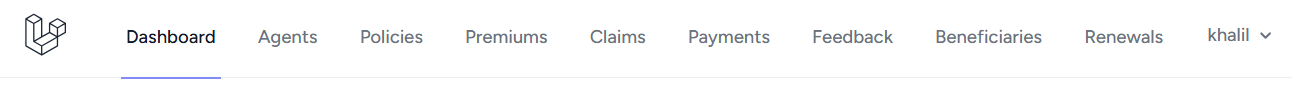
Login page



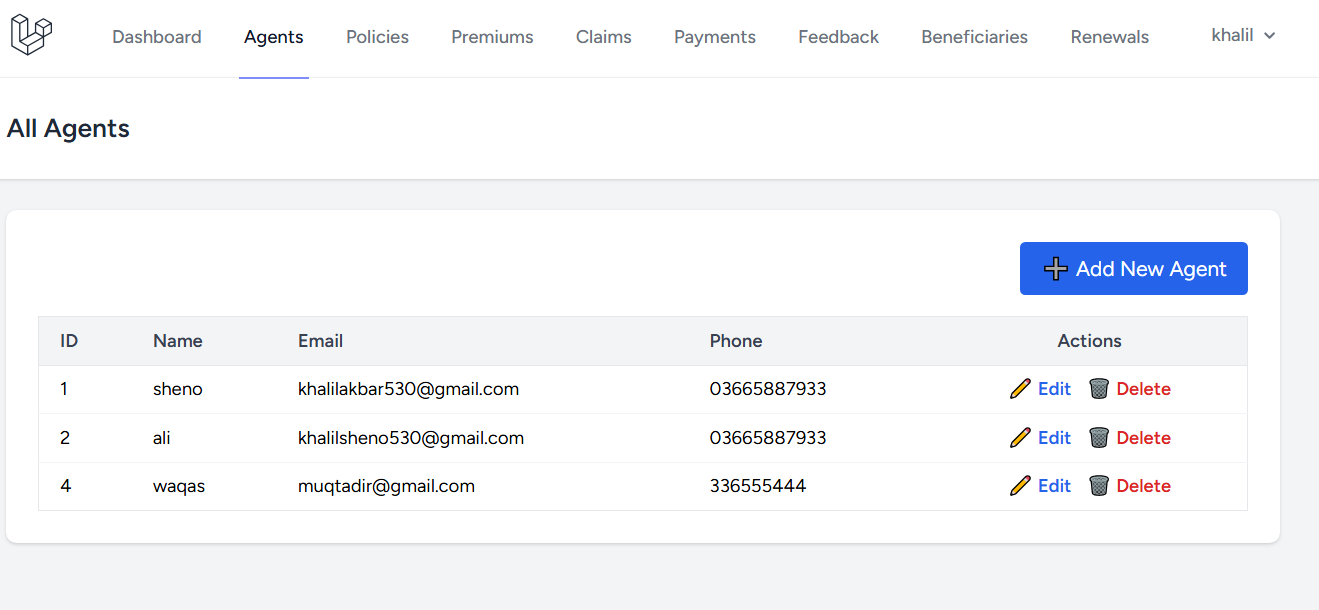
- Dashboard with stats and charts



- User management page



- Entity CRUD pages (agents, policies, etc.)



**Conclusion**

This Laravel-based Insurance Management System provides a robust, secure, and user-friendly platform for managing all aspects of an insurance business.

With role-based access, admin controls, and a modern UI, it is ready for real-world deployment and further expansion.

**Future Enhancements**

* Email notifications for admin requests and claim status
* Export data to PDF/Excel
* Soft delete and trash bin for records
* API endpoints for mobile app integration
* Two-factor authentication for admins

**References**

* [Laravel Documentation](https://laravel.com/docs/)
* [Tailwind CSS](https://tailwindcss.com/)
* [Hostinger Knowledge Base](https://www.hostinger.com/tutorials/)
* [Chart.js](https://www.chartjs.org/)

**Sql data base access**:

DB\_DATABASE=u288075992\_toolbuddy

DB\_USERNAME=u288075992\_tooladmin

DB\_PASSWORD=Khalil123$@

**Project link**

### **indigo-badger-534082.hostingersite.com**