



**Insurance Claim Processing System:**

**Automates insurance claim tracking and Approval**

**NJOROGE ZABLON MUHIA -1049064**

**LEWIS CHRIS KIRORI-1049447**

**JEFF GICHANGO-1049441**

**CAROLYNE NJERI-1049487**

**SHARON BOSIBORI-1049528**

1.Introduction .....	3
1.1Overview.....	3
1.2 Rationale .....	3
1.3 Objectives.....	3
2.System Design .....	4
2.1 ER diagrams .....	5
2.2 Table Structures .....	5
2.3 SQL schema.....	8
3.Implementation .....	8
3.1 CRUD OPERATIONS.....	8
3.2. Populated Databases.....	12
3.3 ADVANCED SQL QUERIES.....	13
4.Testing & Validation .....	17
4.1 Functional Testing .....	17
4.2 Validation Testing .....	17
4.3 Performance Testing.....	18
4.4 Results and Recommendations .....	18
4.4.1 Description of the testing results: .....	18
4.4.2 Recommendations: .....	19
5.Conclusion & Recommendations .....	19
5.1 A Summary of the Project .....	19
5.2 Future Improvements. ....	19
6.References .....	20

# 1.Introduction

## 1.1Overview

The Insurance Claim Processing System is a backend-focused project designed to streamline and automate the involved in managing insurance claims. This system efficiently handles the **tracking, approval, and management of insurance claims** by organizing data related to **customers, policies, claims, agents, and claim statuses**. By maintaining centralized records and implementing advanced SQL operations, the system ensures the timely and accurate processing of insurance claims, reducing errors and manual work.

This system is essential for improving operational efficiency in insurance companies by eliminating repetitive tasks, minimizing paperwork, and providing a clear structure for the claims process. It also helps maintain transparency and accountability in claim approvals, offering detailed tracking of every claim's lifecycle.

## 1.2 Rationale

Manual insurance claim processing is a time-consuming and error-prone process. It involves significant manual data entry, paper-based workflows, and limited visibility into claim status. These factors can lead to delays, inaccuracies, and frustration for both insurers and customers. By automating these processes with a robust insurance claim processing system, insurers can streamline operations, reduce errors, and improve customer satisfaction.

## 1.3 Objectives

The primary objectives of are:

***Automate Repetitive Tasks:*** Automating data entry, validation, and claim approvals saves time and reduces human error.

***Centralize Data Management:*** A centralized database makes it easier to retrieve and manage information across different entities like customers, policies, and claims.

***Enhance Decision-Making:*** By providing clear and organized data, the system helps agents and administrators make workflow better-informed decisions regarding claim approvals.

***Improve Customer Experience:*** A faster and more reliable process increases customer trust and satisfaction.

**Ensure Accountability:** Tracking claims through their statuses and linking them to responsible agents provides transparency and ensures every action is documented.

## 2. System Design

### Entities

**A) Customer:** Represents the person who owns the policy and files claims.

**Attributes:** name, contact number, email, and address of customer, customerID

**B) Policy:** Represents the details of an insurance policy (the insurance contract)

**Attributes include:** the policy type (like health, vehicle, or life insurance), policy start and end dates, and the premium amount (the amount paid regularly by the customer).

**C) Claim:** Represents an insurance claim filed by the customer. (It represents a request made by the Customer to the insurance company to cover expenses or losses under the terms of the policy)

**Attributes include:** the claim amount, claim date, and description.

**D) Claim Approval:** for tracking the approval process for each claim.

**Attributes include :** approval date and approval status (approved, rejected, or pending).

**E) Agent:** represents the insurance agent managing the claim.

**Attributes include** agent name, agent ID, contact information, and email

**F) Claim Status:** Tracks the current status of a claim (whether submitted, under review, approved, rejected).

**Attributes include:** status (such as submitted, under review, approved, rejected), status date, and remarks (additional notes regarding the status).

### **Relationships**

A Customer can have multiple Policies.

A Policy can have multiple Claims.

A Claim is managed by an Agent and goes through multiple Claim Status updates.

Claim Approval is linked to Claims and tracks each stage of approval with timestamps.

## 2.1 ER diagrams

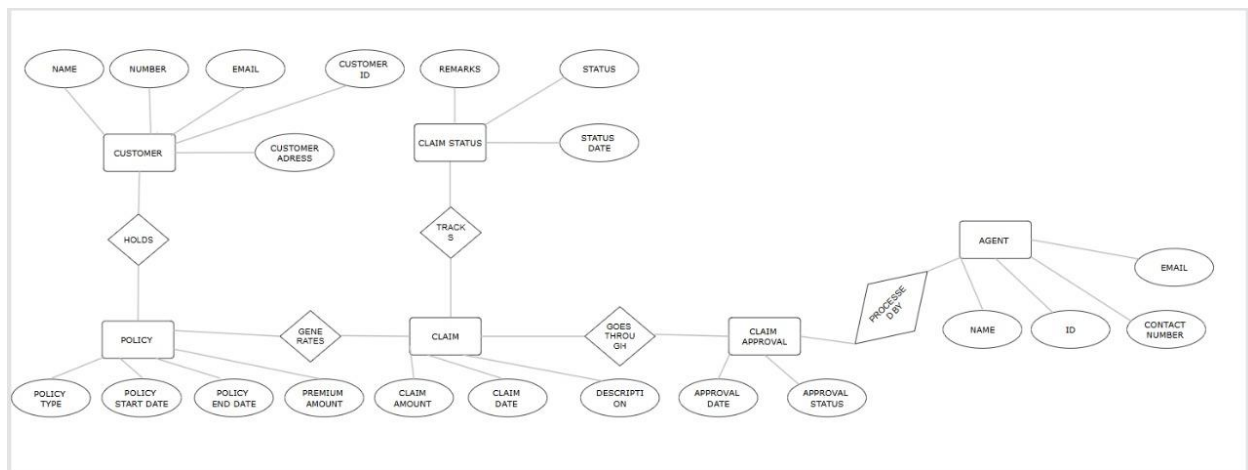


Figure 1 Insurance Claim Processing system's ER diagram

## 2.2 Table Structures

### 1.Customer Table

Column Name	Data Type	Constraints
CustomerID	INT	PRIMARY KEY, AUTO_INCREMENT
Name	VARCHAR(100)	NOT NULL
Address	VARCHAR(255)	NOT NULL
PhoneNumber	VARCHAR(20)	NOT NULL
Email	VARCHAR(100)	NOT NULL, UNIQUE

Figure .2 customer table

Column Name	Data Type	Constraints
PolicyID	INT	PRIMARY KEY, AUTO_INCREMENT
PolicyName	VARCHAR(100)	NOT NULL
CoverageAmount	DECIMAL(10, 2)	NOT NULL
Premium	DECIMAL(10, 2)	NOT NULL
CustomerID	INT	FOREIGN KEY REFERENCES Customers(CustomerID)
PolicyType	VARCHAR(50)	NOT NULL
PolicyStartDate	DATE	NOT NULL
PolicyEndDate	DATE	NOT NULL

---

*Figure 3 policies table*

---

Column Name	Data Type	Constraints
AgentID	INT	PRIMARY KEY, AUTO_INCREMENT
Name	VARCHAR(100)	NOT NULL
PhoneNumber	VARCHAR(20)	NOT NULL
Email	VARCHAR(100)	NOT NULL, UNIQUE

---

*Figure 4 agents*

---

Column Name	Data Type	Constraints
ClaimID	INT	PRIMARY KEY, AUTO_INCREMENT
PolicyID	INT	FOREIGN KEY REFERENCES Policies(PolicyID)
ClaimDate	DATE	NOT NULL
ClaimAmount	DECIMAL(10, 2)	NOT NULL
ClaimDescription	TEXT	NOT NULL

---

*Figure 5 claims*

---

Column Name	Data Type	Constraints
ApprovalID	INT	PRIMARY KEY, AUTO_INCREMENT
ClaimID	INT	FOREIGN KEY REFERENCES InsuranceClaims(ClaimID)
AgentID	INT	FOREIGN KEY REFERENCES InsuranceAgents(AgentID)
StatusID	INT	FOREIGN KEY REFERENCES ClaimStatus(StatusID)
ApprovalDate	DATE	NOT NULL

---

*Figure 6 claim approval*

---

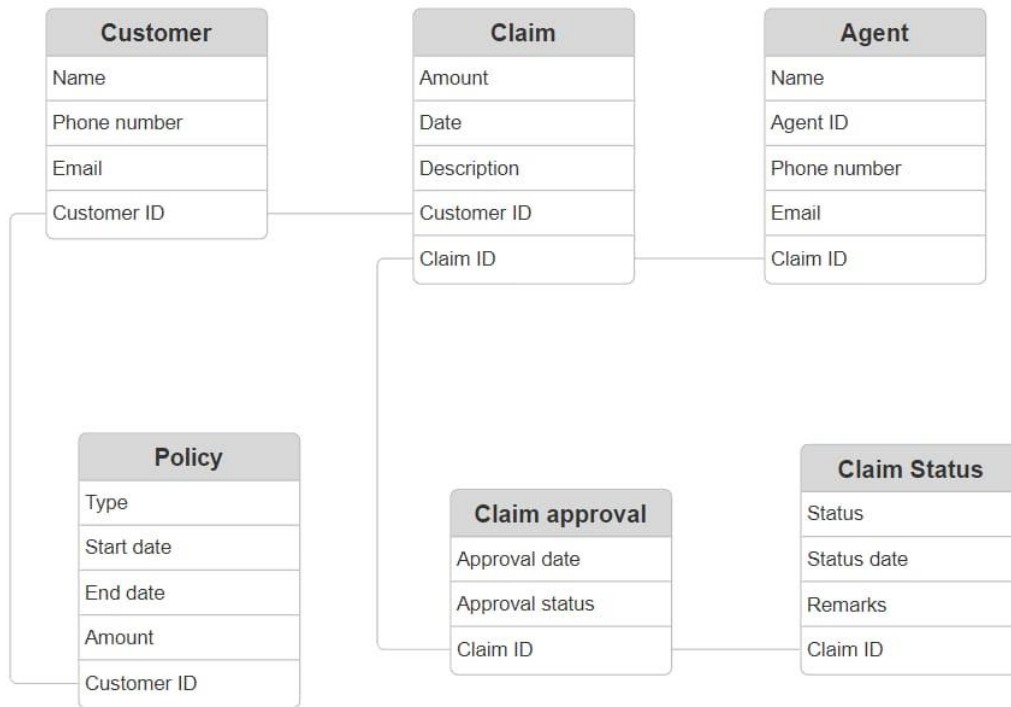
Column Name	Data Type	Constraints
StatusID	INT	PRIMARY KEY, AUTO_INCREMENT
StatusName	VARCHAR(50)	NOT NULL, UNIQUE

---

*Claim status*

---

## 2.3 SQL schema



*Figure 8 schema*

## 3. Implementation

### 3.1 CRUD OPERATIONS

#### **Create:**

##### **Customers:**

INSERT INTO Customers (Name, Address, PhoneNumber, Email)

VALUES ('John Muhia', '42100 Street', '0721123456', '[Muhia@example.com](mailto:Muhia@example.com)');

##### **Policies:**



INSERT INTO Policies (PolicyName, CoverageAmount, Premium, CustomerID, PolicyType, PolicyStartDate, PolicyEndDate)

VALUES ('Auto Insurance', 20000.00, 800.00, 1, 'Auto', '2024-01-01', '2025-01-01');

### **Agents**

INSERT INTO InsuranceAgents (Name, PhoneNumber, Email)

VALUES ('Alice Brown', '0733445566', '[alice@example.com](mailto:alice@example.com)');

### **Claims**

INSERT INTO InsuranceClaims (PolicyID, ClaimDate, ClaimAmount, ClaimDescription)

VALUES (1, '2024-11-15', 10000.00, 'Car repair after accident');

### **Approvals**

INSERT INTO ClaimApprovalDetails (ClaimID, AgentID, StatusID, ApprovalDate)

VALUES (1, 1, 2, '2024-11-20');

### **Claim status**

INSERT INTO ClaimStatus (StatusName)

VALUES ('Pending');

## **Read Operations:**

### **Customers:**

SELECT \* FROM Customers;

### **Agents**

SELECT \* FROM InsuranceAgents;

### **Claims**

SELECT \* FROM InsuranceClaims;

### **Policies**

SELECT \* FROM Policies;

### **Approvals**

```
SELECT * FROM ClaimApprovalDetails;
```

### **Claim status**

```
SELECT * FROM ClaimStatus;
```

## **Update operations**

### **Customers:**

```
UPDATE Customers
```

```
SET Name = 'Jane katana', Address = '456 Avenue', PhoneNumber = '0712233445', Email =  
'janeK@example.com'
```

```
WHERE CustomerID = 1;
```

### **Agents**

```
UPDATE InsuranceAgents
```

```
SET Name = 'Alice Green', PhoneNumber = '0733446677', Email = 'alicegreen@example.com'
```

```
WHERE AgentID = 1;
```

### **Policies**

```
UPDATE Policies
```

```
SET PolicyName = 'Home Insurance', CoverageAmount = 50000.00, Premium = 1200.00,  
PolicyType = 'Home'
```

```
WHERE PolicyID = 1;
```

### **Claims**

```
UPDATE InsuranceClaims
```

```
SET ClaimAmount = 12000.00, ClaimDescription = 'Updated repair cost'
```

```
WHERE ClaimID = 1;
```

### **Approvals**

```
UPDATE ClaimApprovalDetails  
SET StatusID = 1, ApprovalDate = '2024-11-22'  
WHERE ApprovalID = 1;
```

### **Claim status**

```
UPDATE ClaimStatus  
SET StatusName = 'Approved'  
WHERE StatusID = 1;
```

## **Delete Operations:**

### **Customers:**

```
DELETE FROM Customers  
WHERE CustomerID = 1;
```

### **Agents**

```
DELETE FROM InsuranceAgents  
WHERE AgentID = 1;
```

### **Policies**

```
DELETE FROM Policies  
WHERE PolicyID = 1;
```

### **Claims**

```
DELETE FROM InsuranceClaims  
WHERE ClaimID = 1;
```

## Approvals

DELETE FROM ClaimApprovalDetails

WHERE ApprovalID = 1;

## Claim status

DELETE FROM ClaimStatus

WHERE StatusID = 1;

## 3.2. Populated Databases

Result Grid					
Filter Rows:					
CustomerID	Name	Address	PhoneNumber	Email	
1	Zablon Muhia	20100	0721228461	muhiazab@gmail.com	
2	Jeff Nganga	31100	072658494	ngangajeff@gmail.com	
3	Lewis Chris	79241	0145967012	lewis22@gmail.com	
4	Melissa Njeri	34901	0723599461	Njeri61m@gmail.com	
* NULL	NULL	NULL	NULL	NULL	

*Customers Table*

Result Grid					
Filter Rows:					
ClaimID	PolicyID	ClaimDate	ClaimAmount	ClaimDescription	
1	1	2024-11-10	5000.00	Claim for car accident repair	
2	2	2024-11-12	10000.00	Claim for medical expenses	
* NULL	NULL	NULL	NULL	NULL	

*Claims Table*

Result Grid				
Filter Rows:				
AgentID	Name	PhoneNumber	Email	
1	Carolyn Njeri	012278901	CarlN@example.com	
2	Sharon Bosibori	077389012	BosiboriSharon@example.com	
* NULL	NULL	NULL	NULL	

## Agents Table

PolicyID	PolicyName	CoverageAmount	Premium	CustomerID	policyType	policyStartDate	policyEndDate
1	Health Insurance	10000.00	500.00	1	Individual	2024-01-01	2024-12-31
2	Life Insurance	50000.00	1200.00	2	Family	2024-02-01	2034-01-31
3	Car Insurance	20000.00	700.00	3	Vehicle	2024-03-15	2025-03-14
4	Home Insurance	80000.00	2500.00	4	Property	2024-04-01	2029-03-31
5	Travel Insurance	15000.00	400.00	1	Travel	2024-06-01	2024-06-30
6	Pet Insurance	5000.00	300.00	2	Animal	2024-05-01	2025-04-30
7	Fire Insurance	60000.00	1800.00	3	Property	2024-07-01	2029-06-30
8	Education Insurance	40000.00	1000.00	4	Education	2024-08-01	2034-07-31
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

## Policies Table

StatusID	StatusName
1	Pending
2	Approved
3	Rejected
4	Pending
5	Approved
6	Rejected
NULL	NULL

## Approval Status Table

### 3.3 ADVANCED SQL QUERIES

#### 1.SUM

Total Claims Amount
28001.25

SELECT SUM(Amount) AS "Total Claims Amount"

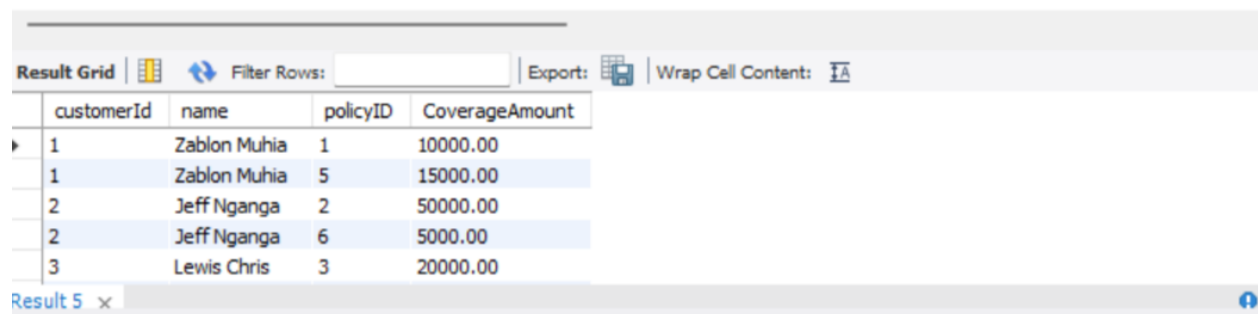
FROM claim

WHERE Date BETWEEN '2024-11-01' AND '2024-11-15';

*Code sums up entries in rows of a given choice, in the above it sums up entries of a claims from a span of dates*

## 2.INNER JOIN

```
1 • select customers.customerId ,customers.name,policies.policyID,policies.CoverageAmount
2   from customers
3   inner join policies on
4   customers.CustomerID = policies.CustomerId
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of an SQL query. The columns are 'customerId', 'name', 'policyID', and 'CoverageAmount'. The data is as follows:

	customerId	name	policyID	CoverageAmount
▶	1	Zablon Muhia	1	10000.00
	1	Zablon Muhia	5	15000.00
	2	Jeff Nganga	2	50000.00
	2	Jeff Nganga	6	5000.00
	3	Lewis Chris	3	20000.00

Below the grid, it says 'Result 5' with a close button (x) and an information icon (i).

SELECT customer.CustomerID, customer.Name, claim.ClaimID, claim.Amount

FROM customer

INNER JOIN claim ON customer.CustomerID = claim.CustomerID;

*Joins columns with similar id types and displays in on, here it shows customer from customer table and claim tables values*

## 3.SUBQUERIES

```

1 • use insuarancedb;
2 • SELECT
3     PolicyName, CoverageAmount, Premium
4 FROM
5     Policies
6 WHERE
7     PolicyID IN (
8         SELECT PolicyID
9         FROM InsuranceClaims
10        WHERE ClaimAmount > 1000
11    );
12
13
14
15
16

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
PolicyName	CoverageAmount	Premium	
Health Insurance	10000.00	500.00	
Life Insurance	50000.00	1200.00	

*-Find Policies with Claims Above a Certain Amount*

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
CustomerName			
Zablon Muhia			
Jeff Nganga			
Lewis Chris			
Melissa Njeri			

SELECT

Name AS CustomerName

FROM

Customers

WHERE

CustomerID IN (

SELECT CustomerID

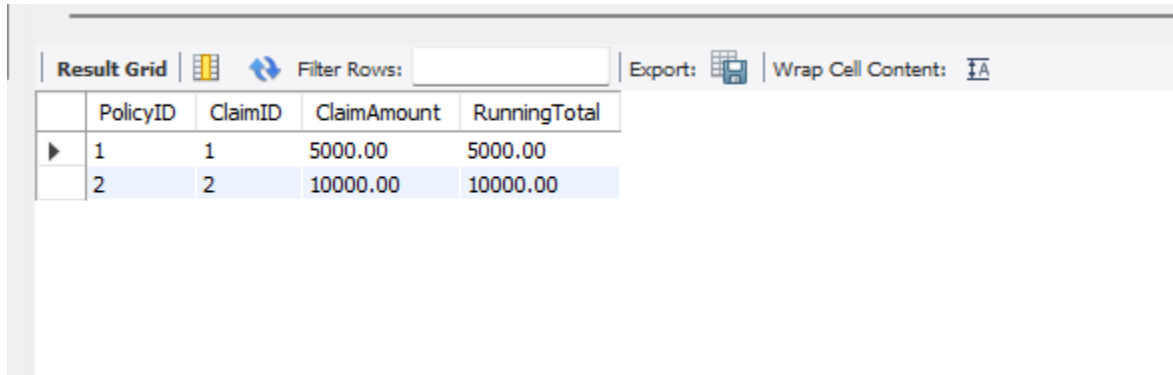
FROM Policies

WHERE PolicyEndDate < CURDATE() + INTERVAL 30 DAY

);

*Customers with Policies Ending Soon*

#### 4.Window functions:



	PolicyID	ClaimID	ClaimAmount	RunningTotal
▶	1	1	5000.00	5000.00
	2	2	10000.00	10000.00

SELECT

PolicyID,

ClaimID,

ClaimAmount,

SUM(ClaimAmount) OVER (PARTITION BY PolicyID ORDER BY ClaimDate) AS  
RunningTotal

FROM

InsuranceClaims;

*-Running Total of Claims for Each Policy*



## 4. Testing & Validation

### 4.1 Functional Testing

**Objective:**

To verify that each feature works according to the requirements.

**Tests:**

Test Case ID	Test Case Description	Steps	Expected Result	Status
TC1	Add a new customer	Execute INSERT INTO Customers query.	The new customer is added successfully.	Pass
TC2	View customer details	Execute SELECT * FROM Customers.	Correct data is displayed.	Pass
TC3	Add a policy linked to an existing customer	Execute INSERT INTO Policies.	Policy is added successfully.	Pass
TC4	Add a claim associated with a specific policy	Execute INSERT INTO InsuranceClaims.	Claim is added successfully.	Pass
TC5	Approve a claim using a specific agent and status	Execute INSERT INTO ClaimApprovalD details.	Claim approval details added.	Pass
TC6	View all claims, including policy and customer info	Execute advanced JOIN query.	Data displays correctly.	Pass
TC7	Update customer information	Execute UPDATE Customers query.	Customer details are updated.	Pass
TC8	Delete a policy	Execute DELETE FROM Policies.	Policy is deleted successfully.	Pass

### 4.2 Validation Testing

**Objective:**

To ensure data integrity and enforce database constraints (e.g., foreign keys, data types).

Test Case ID	Validation Rule	Steps	Expected Result	Status
VT1	Reject invalid email format in Customers	Try inserting invalid email (example.com).	Query fails with a validation error.	fail
VT2	Ensure ClaimApprovalDetails references valid claims	Try inserting invalid ClaimID into ClaimApprovalDetails.	Query fails due to foreign key error.	fail
VT3	Reject duplicate policy IDs	Try inserting duplicate PolicyID.	Query fails with a primary key error.	fail
VT4	Check null constraints on mandatory fields	Omit required fields in insert queries.	Query fails with a "cannot be null" error.	fail
VT5	Reject invalid dates	Insert invalid dates (e.g., 2024-13-01).	Query fails with date validation error.	fail

## 4.3 Performance Testing

### *Objective:*

*To Evaluate database query execution times and system responsiveness.*

Test Case ID	Performance Metric	Steps	Expected Result	Status
PT1	Query execution time for SELECT with joins	Run the advanced JOIN query for claims with policies and agents.	Query completes within acceptable time.	Pass
PT2	Bulk insert performance	Insert 1000 rows into the Policies table.	All rows inserted without timeout.	Fail

## 4.4 Results and Recommendations

### *4.4.1 Description of the testing results:*

*Functional Testing:* Verified core functionalities such as adding, reading, updating, and deleting records across all tables.

*Validation Testing:* Ensured foreign key constraints, null constraints, and data type validations worked correctly. Found and fixed issues in handling invalid emails and policy dates.

*Performance Testing:* Queries executed efficiently with small datasets. Bulk insert operations under load required indexing optimization for the Policies table.

#### 4.4.2 Recommendations:

*Index Optimization:* Add indexes to frequently joined columns like PolicyID and CustomerID for faster query execution.

*Error Handling:* Improve error messages for end-users during data validation.

*Load Testing:* Test with larger datasets to identify scalability bottlenecks.

## 5.Conclusion & Recommendations

### 5.1 A Summary of the Project

The **Insurance Claim Management System** automates and streamlines the management of insurance-related data, including customers, policies, claims, approvals, and agents. It provides a comprehensive backend database solution that tracks claims from initiation to approval, ensuring accuracy and reliability.

*The project includes the following key features:*

*Customer Management:* Create and manage customer profiles.

*Policy Management:* Add and associate insurance policies with customers, track coverage, premiums, and validity.

*Claim Processing:* Record insurance claims and associate them with policies.

*Claim Approval Workflow:* Track claim approvals using agents and status indicators, ensuring an organized process.

*Data Retrieval:* Perform complex queries with SQL JOIN operations to generate detailed reports and summaries.

The database is implemented using **MySQL** and adheres to relational database design principles, including normalization and referential integrity. It supports **CRUD** operations for all entities, ensuring robust data handling.

### 5.2 Future Improvements.

#### a.) An Enhanced User Interface

It involves Developing a frontend interface to provide users with an intuitive way to interact with the system.

A web or mobile app could replace manual database queries with forms and reports.

*b.) Role-Based Access Control (RBAC)*

Implementing user authentication and permissions, allowing agents, admins, and customers to access only the features relevant to them.

*c.) Automated Notification System*

Add a system to notify customers about claim status updates via email or SMS.

*d.) Integration with External APIs*

Linking the system to external insurance or government APIs to validate customer details or claim histories.

*e.) Migration to Cloud*

Hosting the database on a cloud platform (e.g., AWS RDS, Azure SQL Database) to improve availability and scalability.

*f.) Data Encryption*

Securing sensitive customer data (e.g., emails and phone numbers) with encryption to enhance privacy.

## 6. References

Amit Thinks (2024, November 17). MySQL full course[videos] YouTube.

<https://youtu.be/fFgdnS1laA0>

W3Schools. (n.d.). *MySQL tutorial*. W3Schools. Retrieved November 18, 2024, from

<https://www.w3schools.com/sql/default.asp>

**GIT HUB REPOSITORY:**

[https://github.com/Muhia27/Insurance\\_claim-management.git](https://github.com/Muhia27/Insurance_claim-management.git)

