Insurance Agency Management System

By

Name:
Student No

Template by Dr. Shazia A Afzal

Contents

1. Details of the chosen business case

Insurance Agency Management System is a web application that is developed to track the details of the insurance policy and customer details. This website is an online insurance tracking and information management system. This system is a platform that provides its user easy access to information regarding consumer and insurance resources. The prime objective of the developed system is to make sure that customers can access all available insurance policies added by the company, check the status of their insurance policy as well as premium, and access the policy details of all their previous policies. It also enables administrative users to add or modify any insurance policy from their end.

This system uses a combined database including various entities such as

1) Insurance Policies which consist of details of the insurance policy(Policy Id, Policy Type, Premium of the policy, Duration of the policy, Benefits, and Drawbacks)
2) The customer plays an important role in this system, and it consists of the Id of the customer, Customer Name, Customer Address, Customer Email Id, Contact No
3) The agent consists of the name of the agent, agent id, contact no, email id, address
4) Payment consists of payment id, payment amount, date, customer details

Hence details mentioned above regarding the database are subject to collection from various insurance companies and their customers which is an end users in this system.

2. Scope

The scope must suggest the range of the data included/not included in the project.

- What is included in the project
- What is not included

2.1. Business Requirements

1) To be able to access the insurance policy details for customers, the company should be able to add various policies from their end. Hence using stored procedures admin from the company can track the policy resource

2) Once the user/customer login to the system they should be able to view all available policies and chose their required policy after comparing them
3) The customer should be able to buy the required policy

4) Customers should be able to buy endorsements

5) Customers should be able to keep a track of their previous policies and ongoing policies on one platform

6) Customers should be able to get regular updates regarding the policy details that are being modified at the admin end.

7) The agent should be able to view the contact details of the customer that they are dealing with.

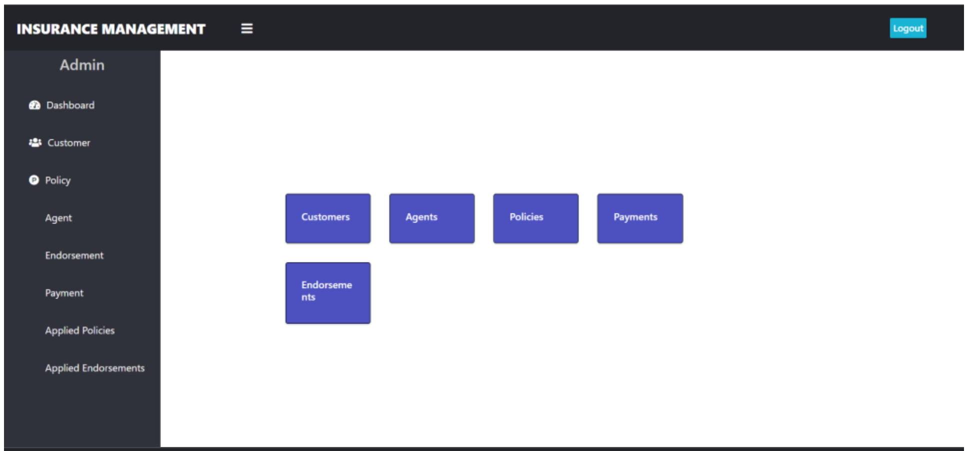
8) There should be direct communication between Company and the Customer. No agent as mediator during a claim.

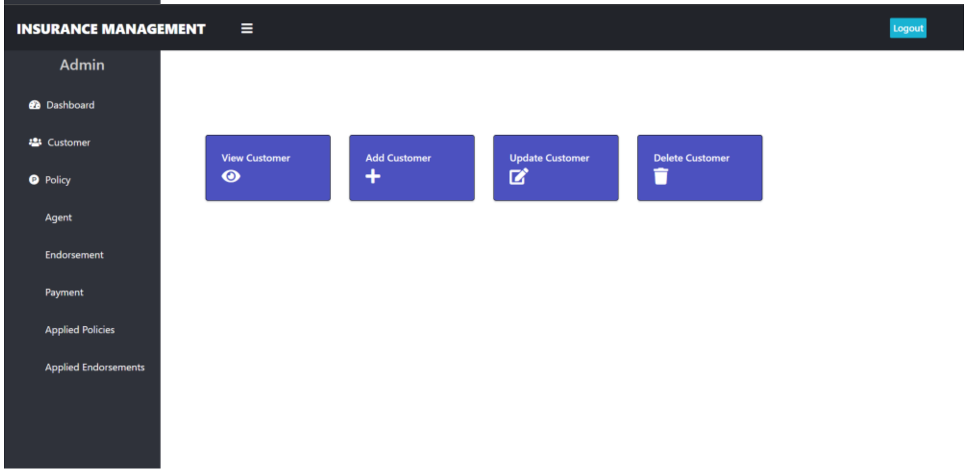
3. Business Rules

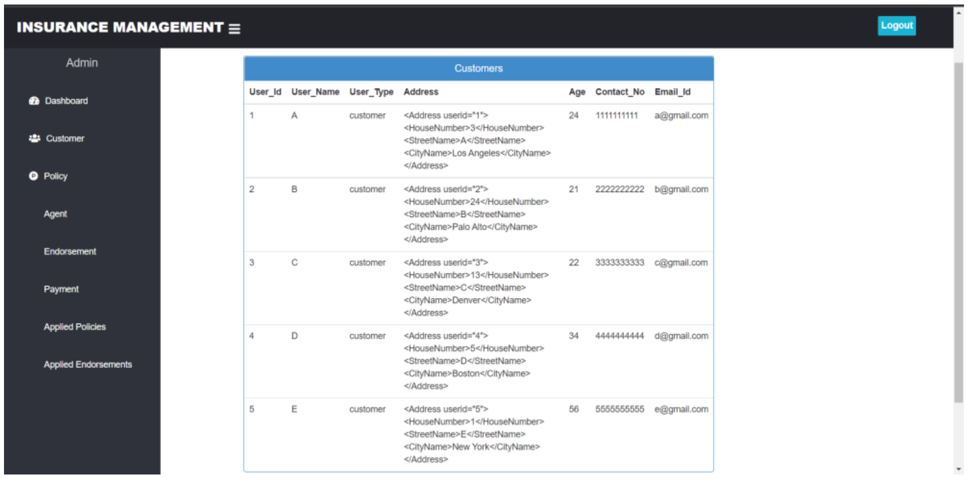
3.1. Implementation of business rules – screen shots

1) This system implements the following conditions using stored procedures and triggers
2) The company can add policies and details
3) Customers can view various policies and compare them
4) Customers can buy a policy

5) Agents can communicate with customers through any media
6) Customers can buy endorsements
7) Notify customers about their premium payment and its penalty if delayed
8) Notify customer whenever a new policy is added by the company
9) The customer can decide in which form he wants the returns or reimbursements (stocks, crypto, etc.).
10) Motivational notifications to pay the advance premium and to not discard the policies in between (discount/benefit).

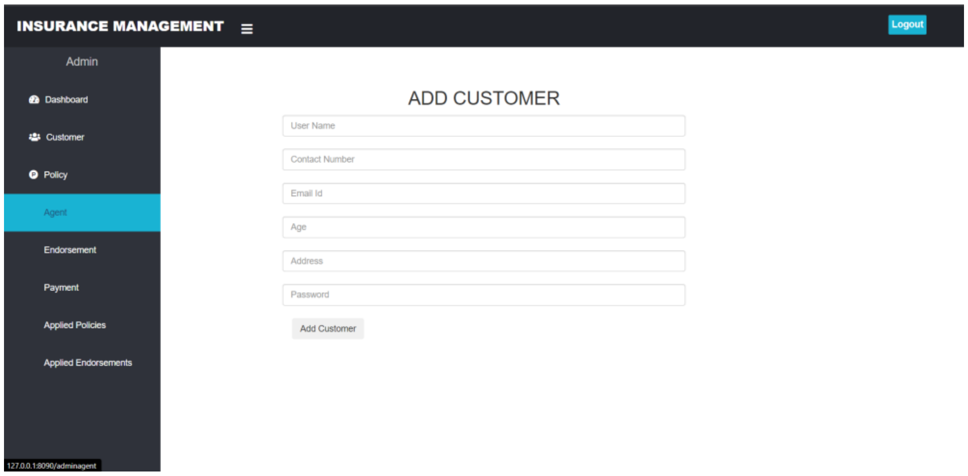






Customers

User_Id	User_Name	User_Type	Address	Age	Contact_No	Email_Id
1	A	customer	<Address userid="1"> <HouseNumber>3</HouseNumber> <StreetName>A</StreetName> <CityName>Los Angeles</CityName> </Address>	24	1111111111	a@gmail.com
2	B	customer	<Address userid="2"> <HouseNumber>24</HouseNumber> <StreetName>B</StreetName> <CityName>Palo Alto</CityName> </Address>	21	2222222222	b@gmail.com
3	C	customer	<Address userid="3"> <HouseNumber>13</HouseNumber> <StreetName>C</StreetName> <CityName>Denver</CityName> </Address>	22	3333333333	c@gmail.com
4	D	customer	<Address userid="4"> <HouseNumber>5</HouseNumber> <StreetName>D</StreetName> <CityName>Boston</CityName> </Address>	34	4444444444	d@gmail.com
5	E	customer	<Address userid="5"> <HouseNumber>1</HouseNumber> <StreetName>E</StreetName> <CityName>New York</CityName> </Address>	56	5555555555	e@gmail.com



ADD CUSTOMER

User Name

Contact Number

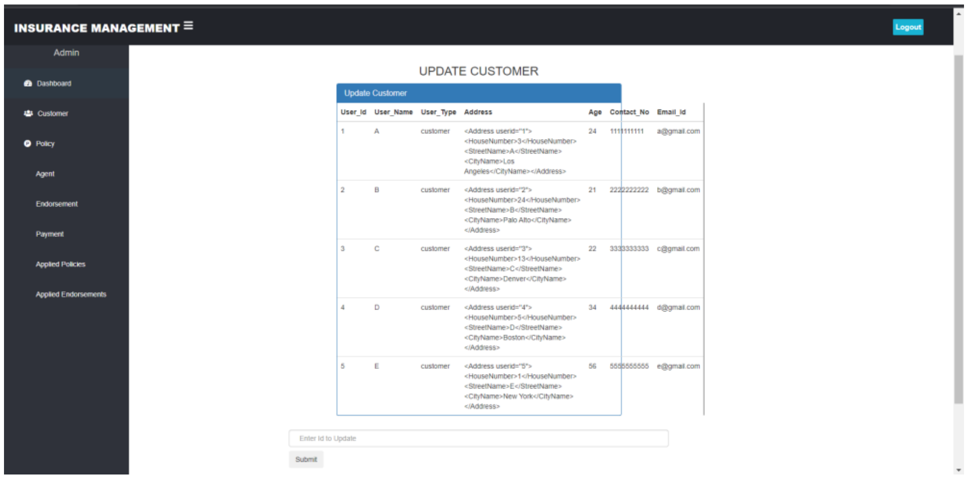
Email Id

Age

Address

Password

Add Customer



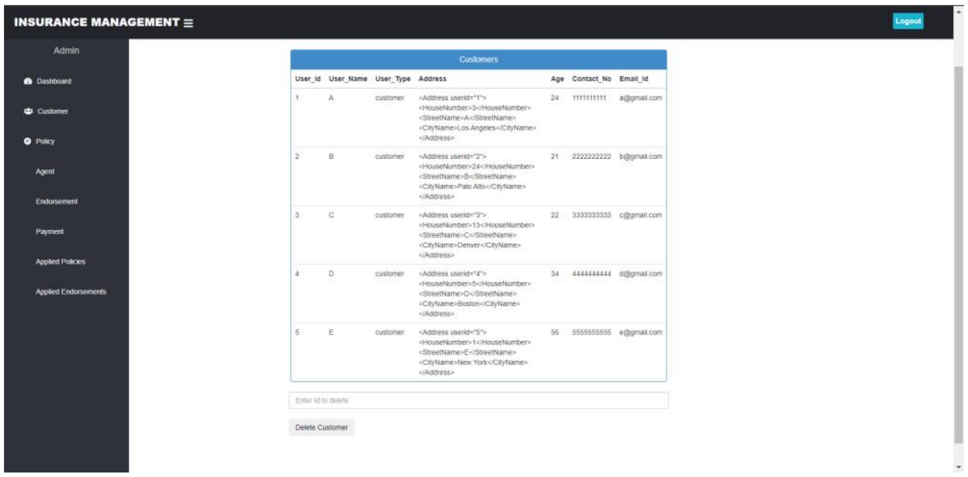
UPDATE CUSTOMER

Update Customer

User_Id	User_Name	User_Type	Address	Age	Contact_No	Email_Id
1	A	customer	<Address userid="1"> <HouseNumber>3</HouseNumber> <StreetName>A</StreetName> <CityName>Los Angeles</CityName></Address>	24	1111111111	a@gmail.com
2	B	customer	<Address userid="2"> <HouseNumber>24</HouseNumber> <StreetName>B</StreetName> <CityName>Palo Alto</CityName> </Address>	21	2222222222	b@gmail.com
3	C	customer	<Address userid="3"> <HouseNumber>13</HouseNumber> <StreetName>C</StreetName> <CityName>Denver</CityName> </Address>	22	3333333333	c@gmail.com
4	D	customer	<Address userid="4"> <HouseNumber>5</HouseNumber> <StreetName>D</StreetName> <CityName>Boston</CityName> </Address>	34	4444444444	d@gmail.com
5	E	customer	<Address userid="5"> <HouseNumber>1</HouseNumber> <StreetName>E</StreetName> <CityName>New York</CityName> </Address>	56	5555555555	e@gmail.com

Enter Id to Update

Submit

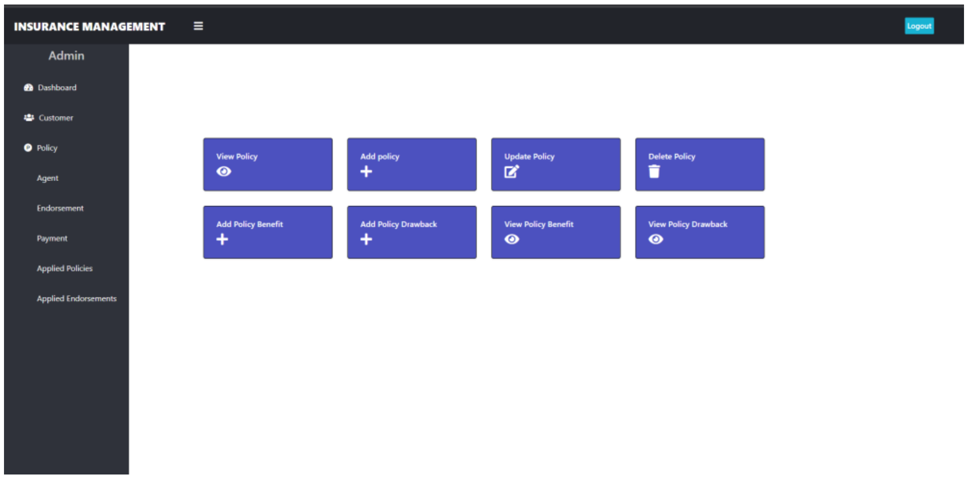


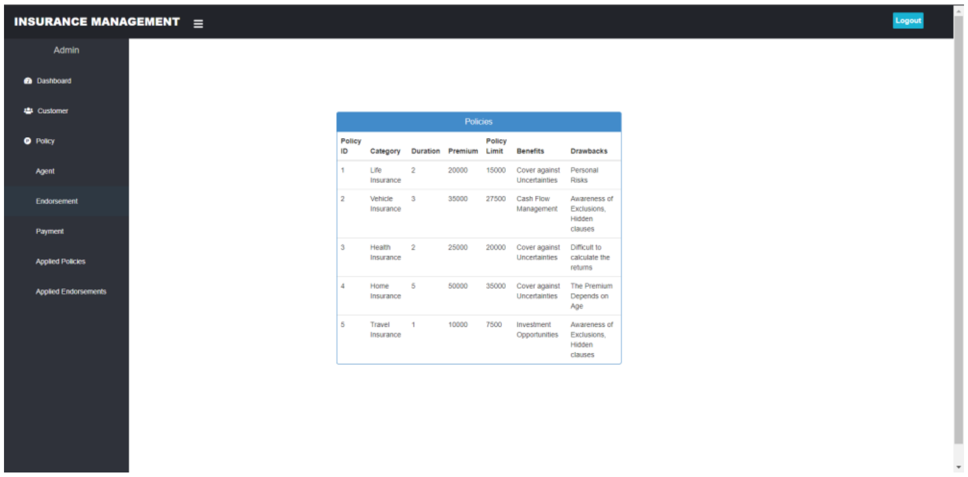
Customers

User_Id	User_Name	User_Type	Address	Age	Contact_No	Email_Id
1	A	customer	<Address userid="1"> <HouseNumber>3</HouseNumber> <StreetName>A</StreetName> <CityName>Los Angeles</CityName> </Address>	24	1111111111	a@gmail.com
2	B	customer	<Address userid="2"> <HouseNumber>24</HouseNumber> <StreetName>B</StreetName> <CityName>Palo Alto</CityName> </Address>	21	2222222222	b@gmail.com
3	C	customer	<Address userid="3"> <HouseNumber>13</HouseNumber> <StreetName>C</StreetName> <CityName>Denver</CityName> </Address>	22	3333333333	c@gmail.com
4	D	customer	<Address userid="4"> <HouseNumber>5</HouseNumber> <StreetName>D</StreetName> <CityName>Boston</CityName> </Address>	34	4444444444	d@gmail.com
5	E	customer	<Address userid="5"> <HouseNumber>1</HouseNumber> <StreetName>E</StreetName> <CityName>New York</CityName> </Address>	56	5555555555	e@gmail.com

Enter Id to delete

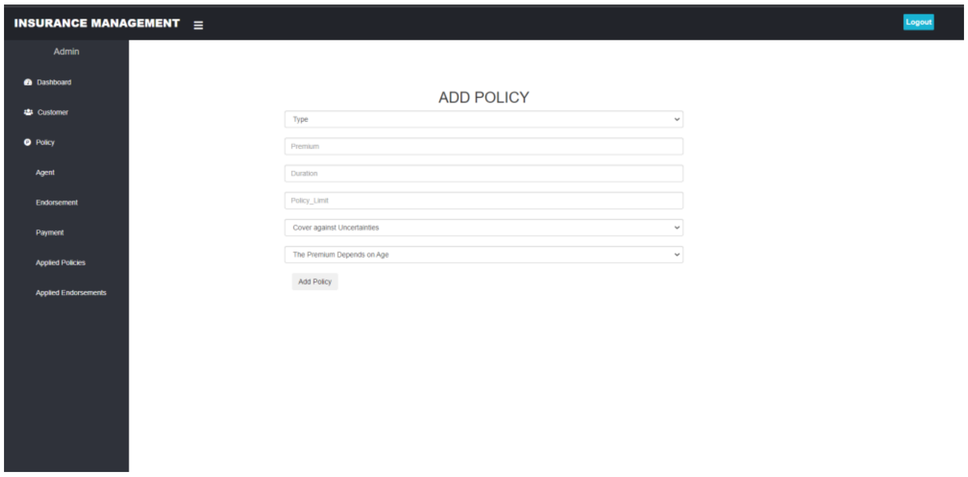
Delete Customer





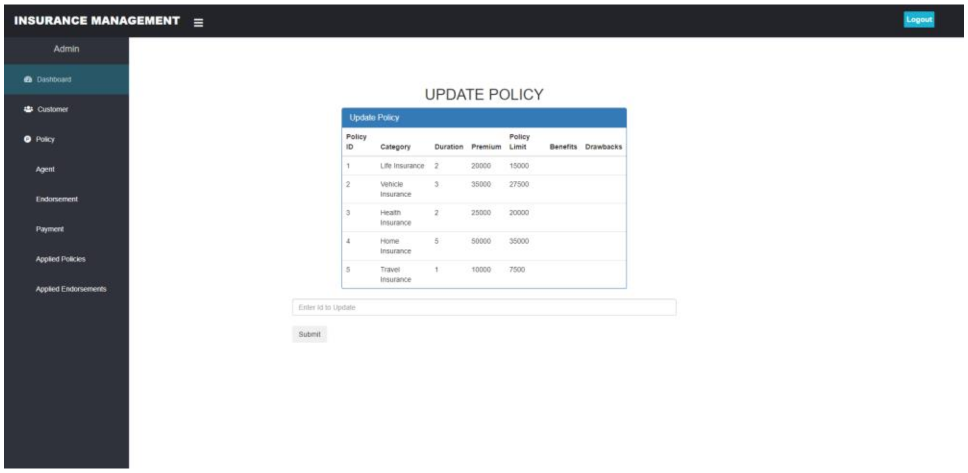
Policies

Policy ID	Category	Duration	Premium	Policy Limit	Benefits	Drawbacks
1	Life Insurance	2	20000	15000	Cover against Uncertainties	Personal Risks
2	Vehicle Insurance	3	35000	27500	Cash Flow Management	Awareness of Exclusions, Hidden clauses
3	Health Insurance	2	25000	20000	Cover against Uncertainties	Difficult to calculate the returns
4	Home Insurance	5	50000	35000	Cover against Uncertainties	The Premium Depends on Age
5	Travel Insurance	1	10000	7500	Investment Opportunities	Awareness of Exclusions, Hidden clauses



ADD POLICY

Type	
Premium	
Duration	
Policy_Limit	
	Cover against Uncertainties
	The Premium Depends on Age



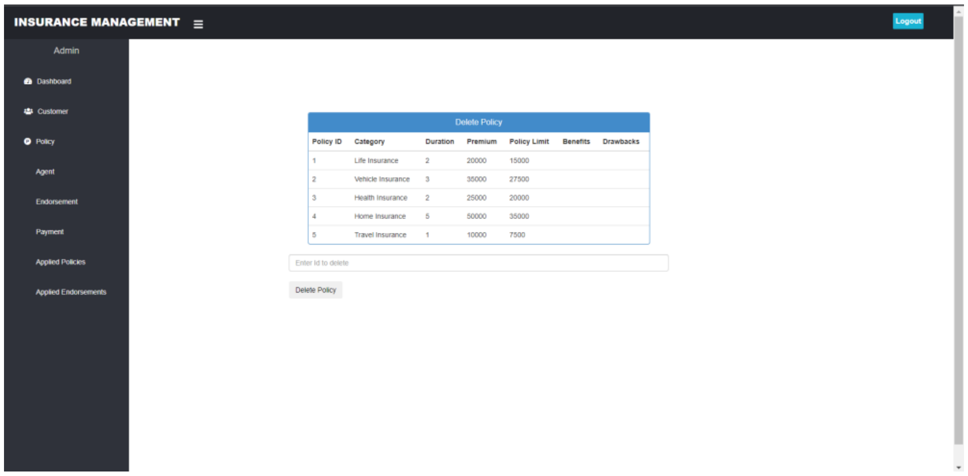
UPDATE POLICY

Update Policy

Policy ID	Category	Duration	Premium	Policy Limit	Benefits	Drawbacks
1	Life Insurance	2	20000	15000		
2	Vehicle Insurance	3	35000	27500		
3	Health Insurance	2	25000	20000		
4	Home Insurance	5	50000	35000		
5	Travel Insurance	1	10000	7500		

Enter Id to Update

Submit

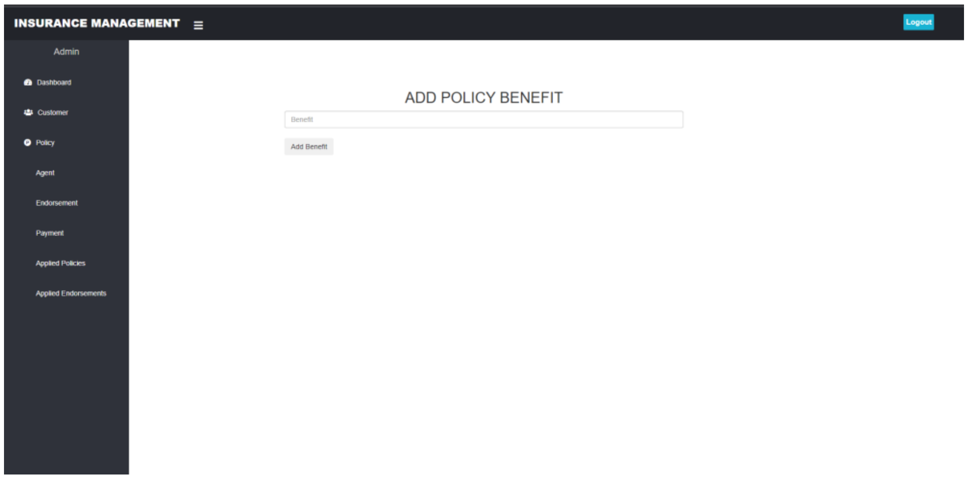


Delete Policy

Policy ID	Category	Duration	Premium	Policy Limit	Benefits	Drawbacks
1	Life Insurance	2	20000	15000		
2	Vehicle Insurance	3	35000	27500		
3	Health Insurance	2	25000	20000		
4	Home Insurance	5	50000	35000		
5	Travel Insurance	1	10000	7500		

Enter Id to delete

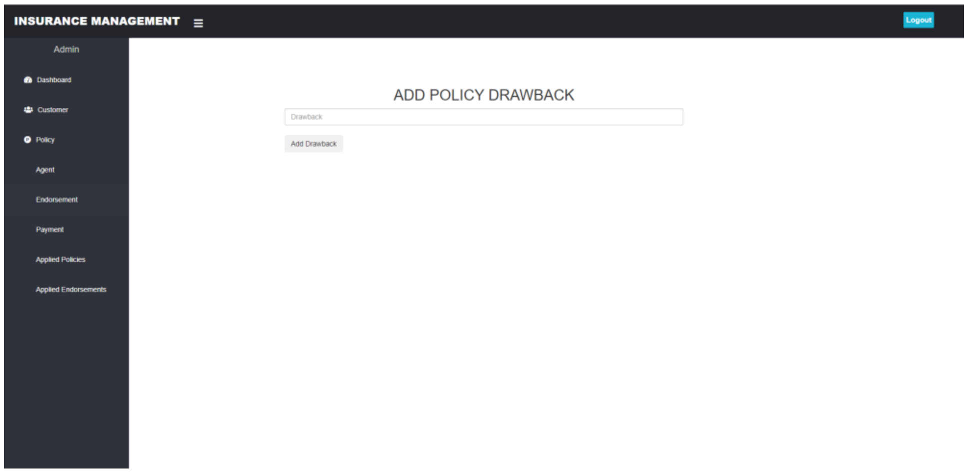
Delete Policy

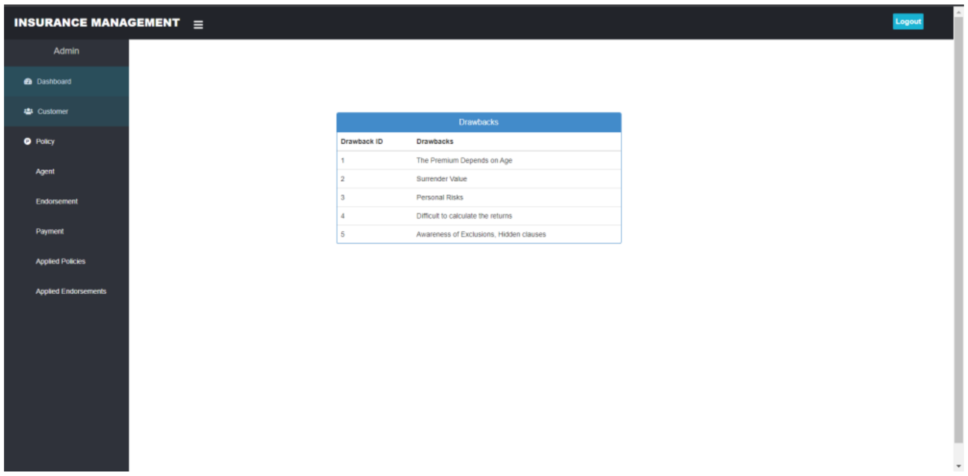


ADD POLICY BENEFIT

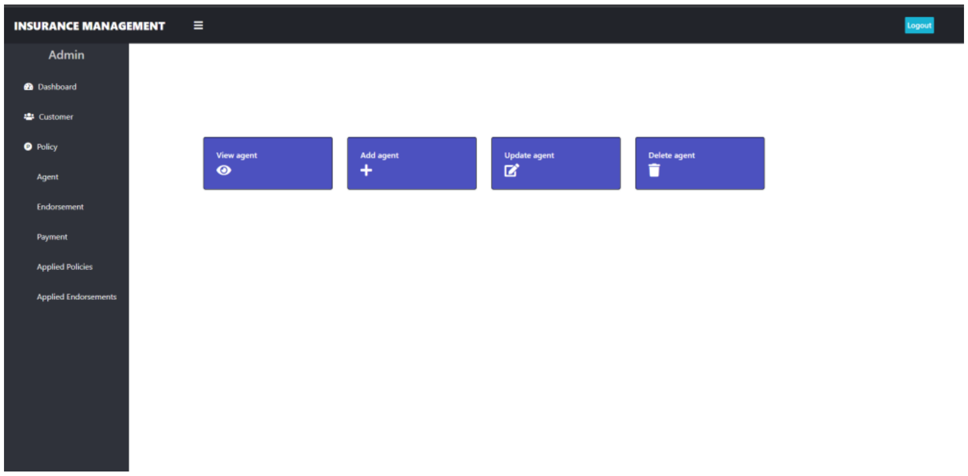
Benefit

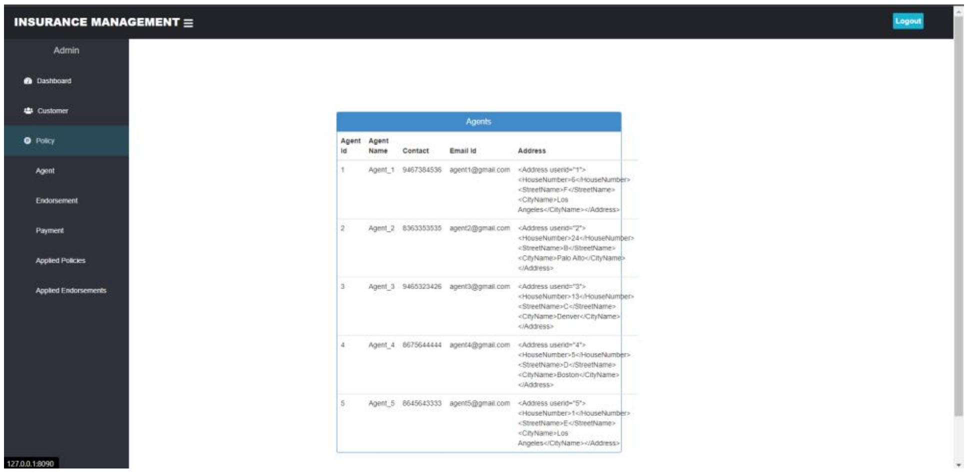
Add Benefit





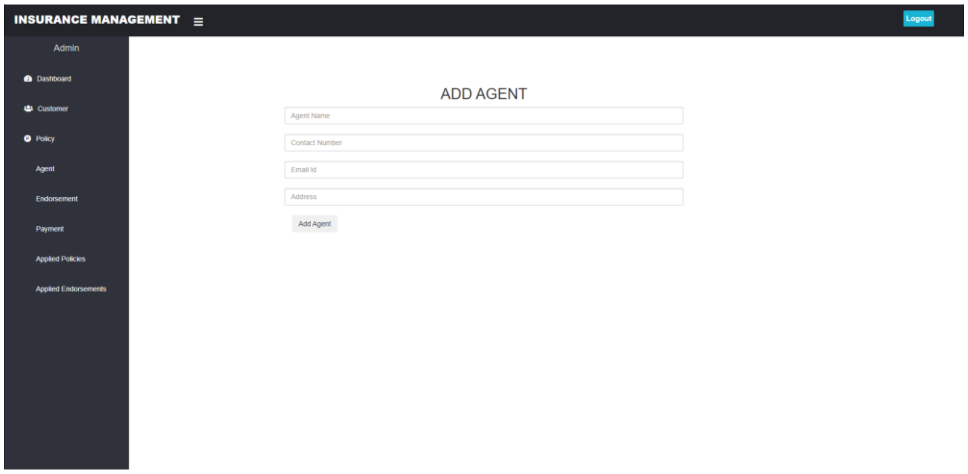
Drawback ID	Drawbacks
1	The Premium Depends on Age
2	Surrender Value
3	Personal Risks
4	Difficult to calculate the returns
5	Awareness of Exclusions, Hidden clauses





Agents

Agent Id	Agent Name	Contact	Email Id	Address
1	Agent_1	9467384536	agent1@gmail.com	<Address userid="1"><HouseNumber>6</HouseNumber><StreetName>F</StreetName><CityName>Los Angeles</CityName></Address>
2	Agent_2	8363353535	agent2@gmail.com	<Address userid="2"><HouseNumber>24</HouseNumber><StreetName>B</StreetName><CityName>Palo Alto</CityName></Address>
3	Agent_3	9465323426	agent3@gmail.com	<Address userid="3"><HouseNumber>13</HouseNumber><StreetName>C</StreetName><CityName>Denver</CityName></Address>
4	Agent_4	8675644444	agent4@gmail.com	<Address userid="4"><HouseNumber>5</HouseNumber><StreetName>D</StreetName><CityName>Boston</CityName></Address>
5	Agent_5	8645643333	agent5@gmail.com	<Address userid="5"><HouseNumber>1</HouseNumber><StreetName>E</StreetName><CityName>Los Angeles</CityName></Address>



ADD AGENT

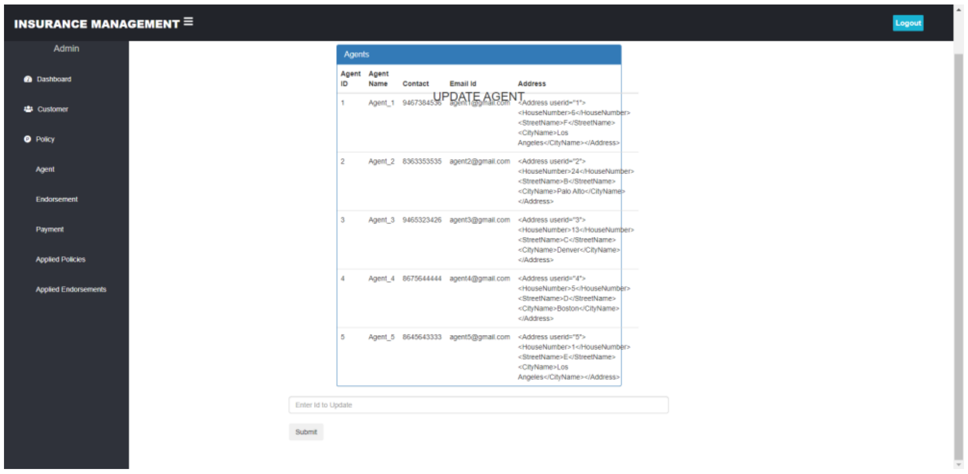
Agent Name

Contact Number

Email Id

Address

Add Agent



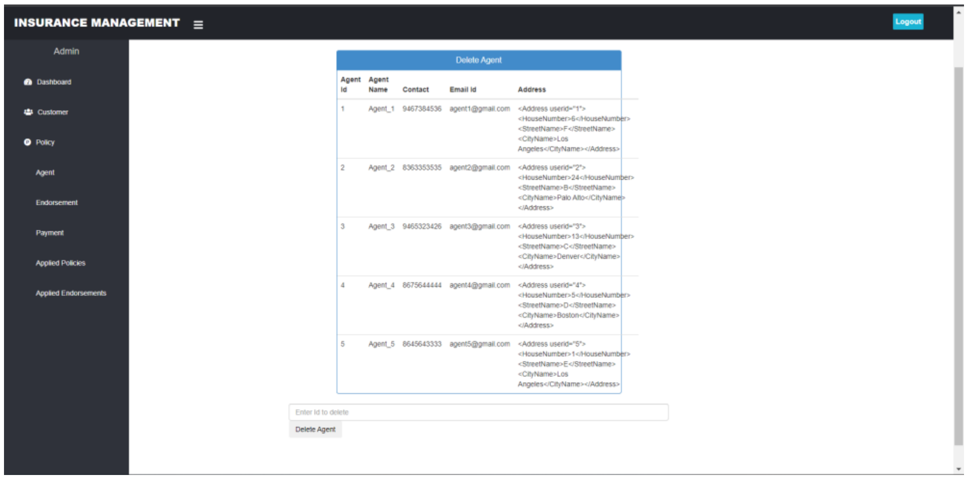
Agents

UPDATE AGENT

Agent ID	Agent Name	Contact	Email Id	Address
1	Agent_1	9467384536	agent1@gmail.com	<Address userid="1"><HouseNumber>6</HouseNumber><StreetName>F</StreetName><CityName>Los Angeles</CityName></Address>
2	Agent_2	8363353535	agent2@gmail.com	<Address userid="2"><HouseNumber>24</HouseNumber><StreetName>B</StreetName><CityName>Palo Alto</CityName></Address>
3	Agent_3	9465323426	agent3@gmail.com	<Address userid="3"><HouseNumber>13</HouseNumber><StreetName>C</StreetName><CityName>Denver</CityName></Address>
4	Agent_4	8675644444	agent4@gmail.com	<Address userid="4"><HouseNumber>5</HouseNumber><StreetName>D</StreetName><CityName>Boston</CityName></Address>
5	Agent_5	8645643333	agent5@gmail.com	<Address userid="5"><HouseNumber>1</HouseNumber><StreetName>E</StreetName><CityName>Los Angeles</CityName></Address>

Enter Id to Update

Submit

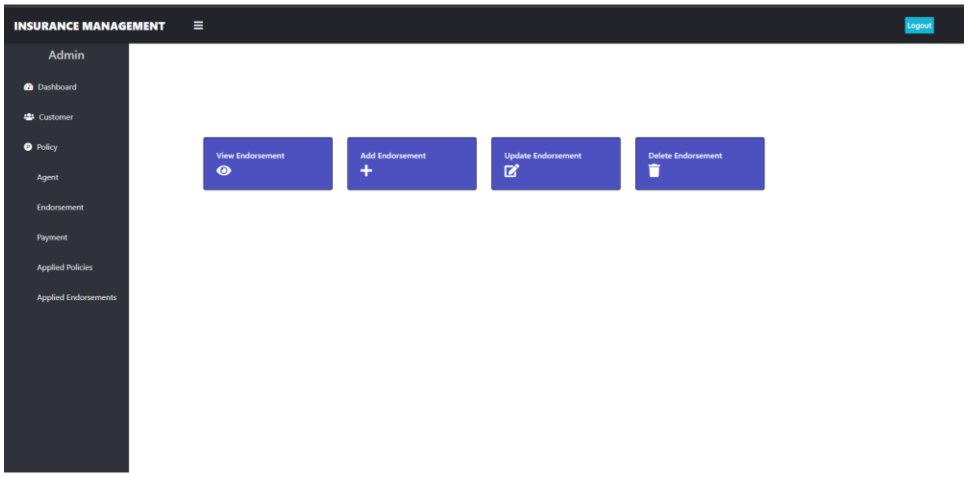


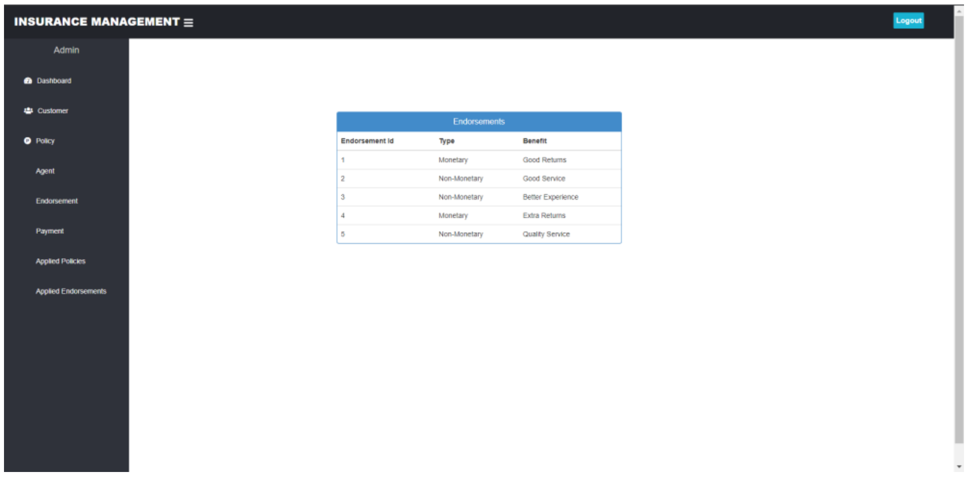
Delete Agent

Agent Id	Agent Name	Contact	Email Id	Address
1	Agent_1	9467384536	agent1@gmail.com	<Address userid="1"><HouseNumber>6</HouseNumber><StreetName>F</StreetName><CityName>Los Angeles</CityName></Address>
2	Agent_2	8363353535	agent2@gmail.com	<Address userid="2"><HouseNumber>24</HouseNumber><StreetName>B</StreetName><CityName>Palo Alto</CityName></Address>
3	Agent_3	9465323426	agent3@gmail.com	<Address userid="3"><HouseNumber>13</HouseNumber><StreetName>C</StreetName><CityName>Denver</CityName></Address>
4	Agent_4	8675644444	agent4@gmail.com	<Address userid="4"><HouseNumber>5</HouseNumber><StreetName>D</StreetName><CityName>Boston</CityName></Address>
5	Agent_5	8645643333	agent5@gmail.com	<Address userid="5"><HouseNumber>1</HouseNumber><StreetName>E</StreetName><CityName>Los Angeles</CityName></Address>

Enter Id to delete

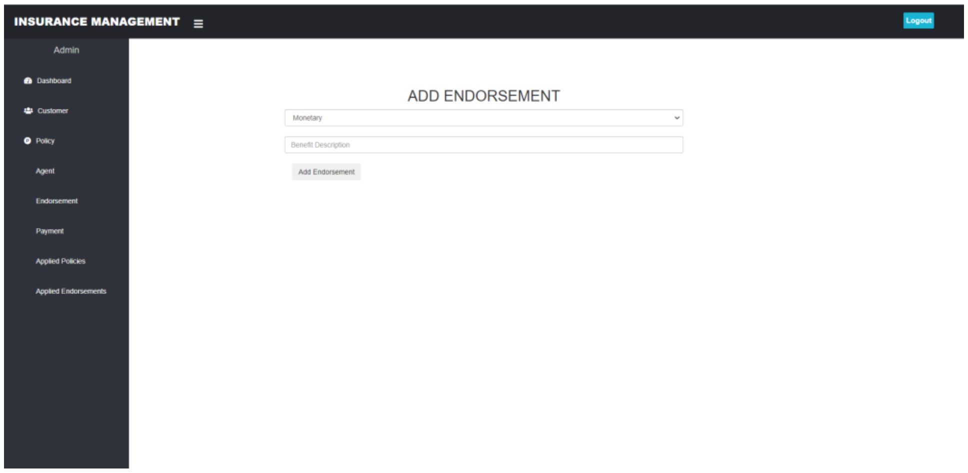
Delete Agent





Endorsements

Endorsement Id	Type	Benefit
1	Monetary	Good Returns
2	Non-Monetary	Good Service
3	Non-Monetary	Better Experience
4	Monetary	Extra Returns
5	Non-Monetary	Quality Service

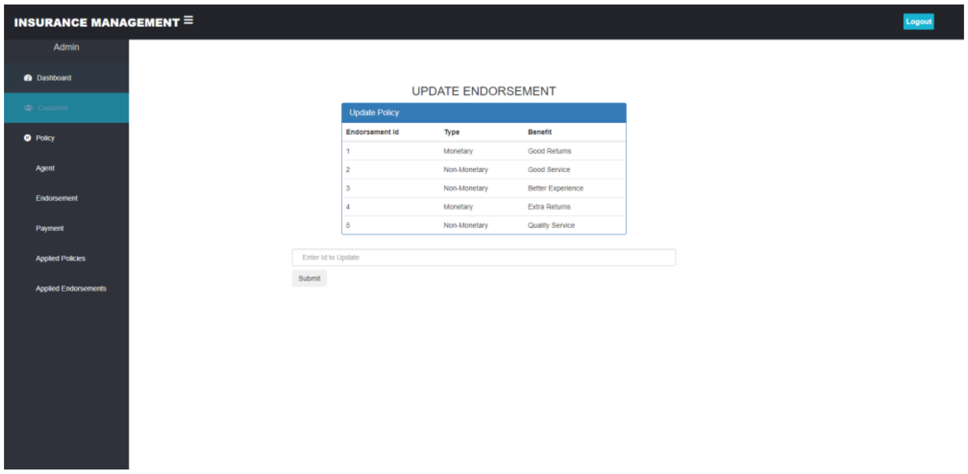


ADD ENDORSEMENT

Monetary

Benefit Description

Add Endorsement



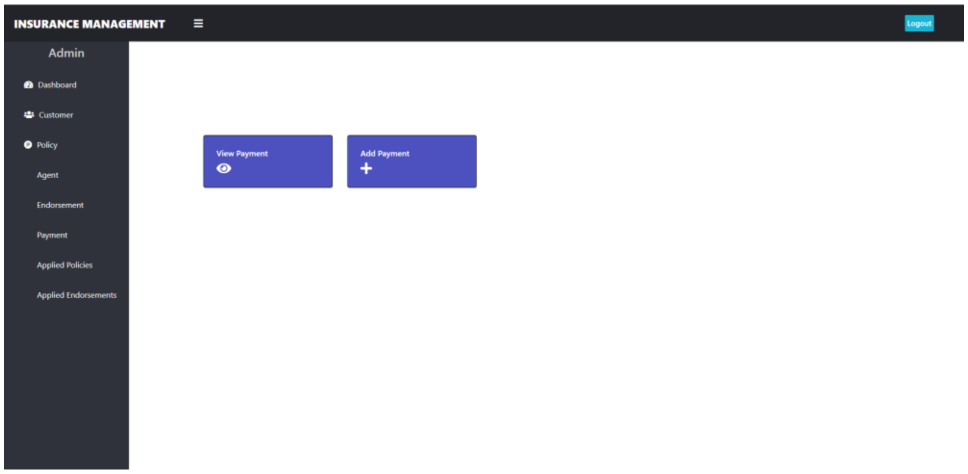
UPDATE ENDORSEMENT

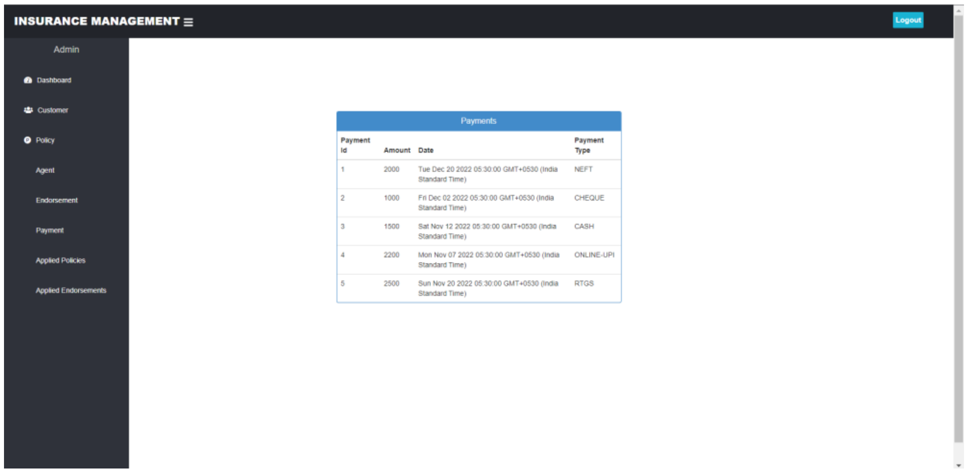
Update Policy

Endorsement Id	Type	Benefit
1	Monetary	Good Returns
2	Non-Monetary	Good Service
3	Non-Monetary	Better Experience
4	Monetary	Extra Returns
5	Non-Monetary	Quality Service

Enter Id to Update

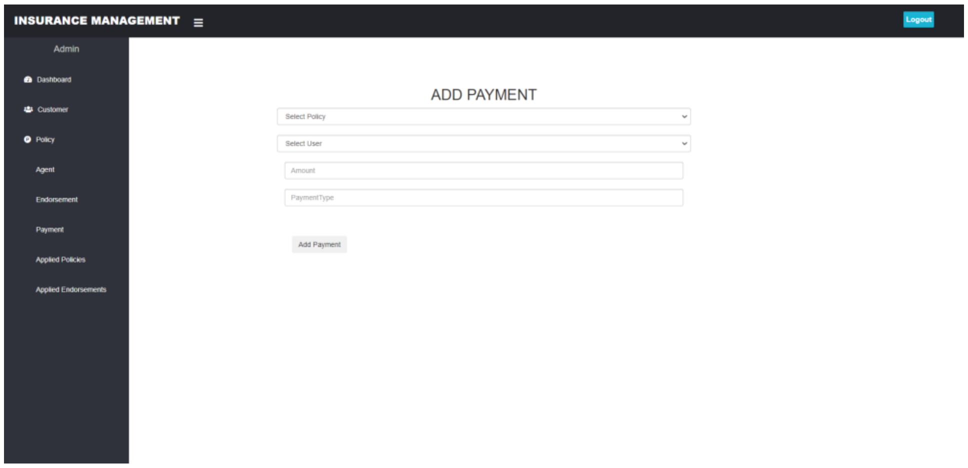
Submit

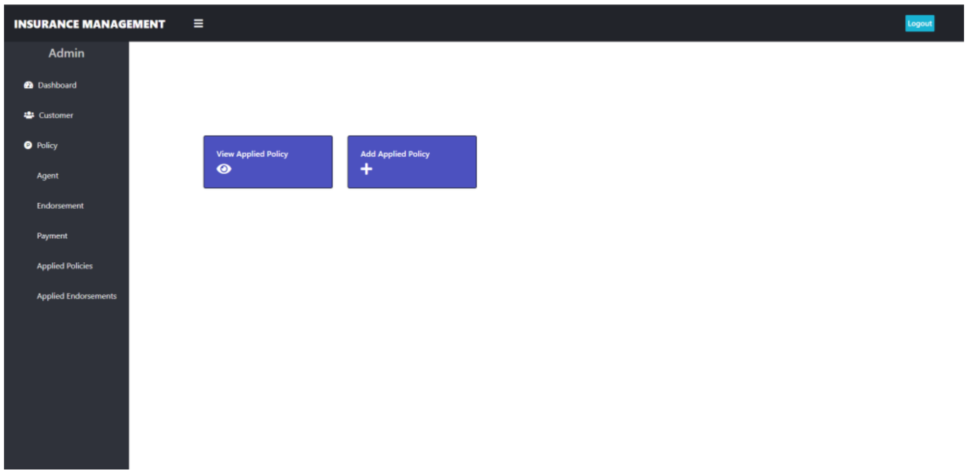


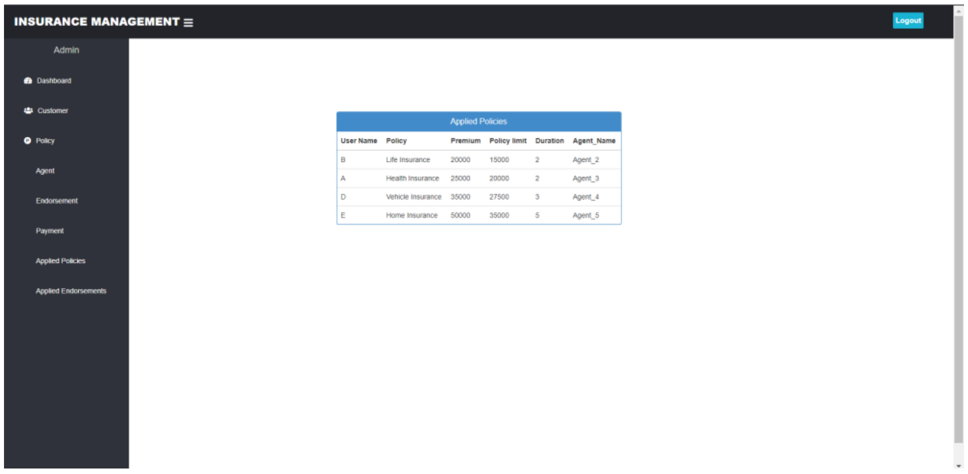


Payments

Payment Id	Amount	Date	Payment Type
1	2000	Tue Dec 20 2022 05:30:00 GMT+0530 (India Standard Time)	NEFT
2	1000	Fri Dec 02 2022 05:30:00 GMT+0530 (India Standard Time)	CHEQUE
3	1500	Sat Nov 12 2022 05:30:00 GMT+0530 (India Standard Time)	CASH
4	2200	Mon Nov 07 2022 05:30:00 GMT+0530 (India Standard Time)	ONLINE-UPI
5	2500	Sun Nov 20 2022 05:30:00 GMT+0530 (India Standard Time)	RTGS

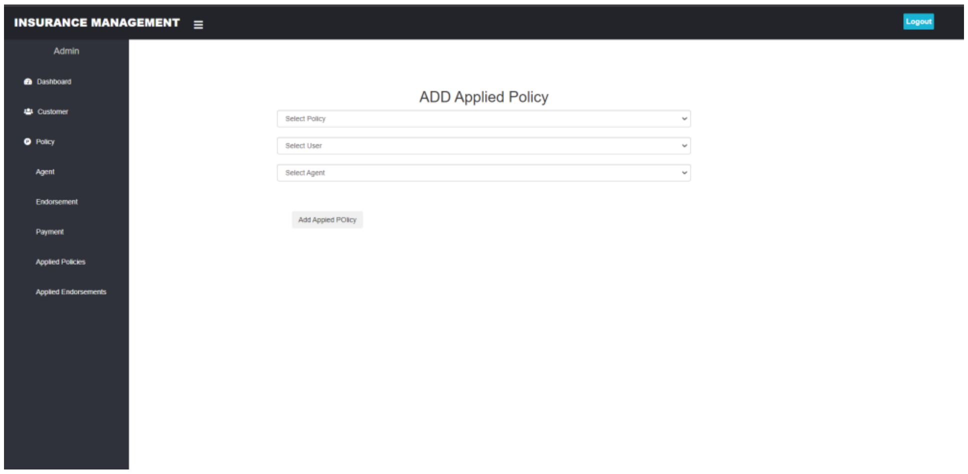


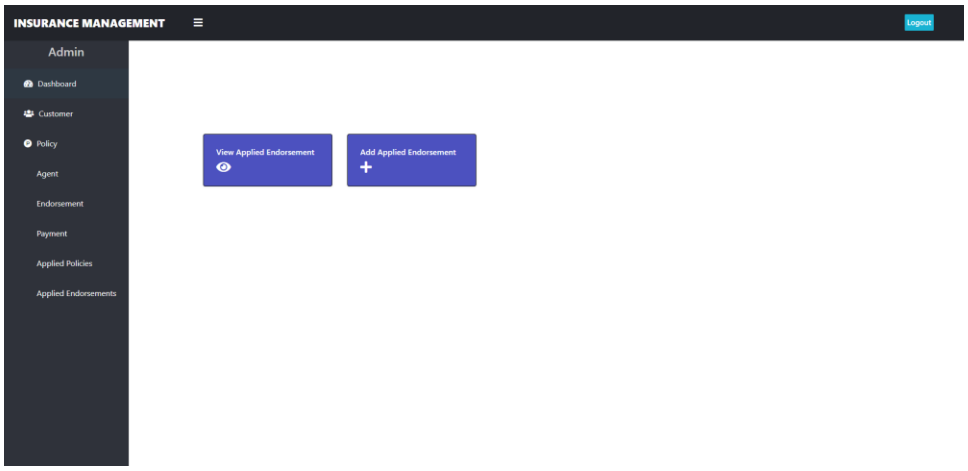


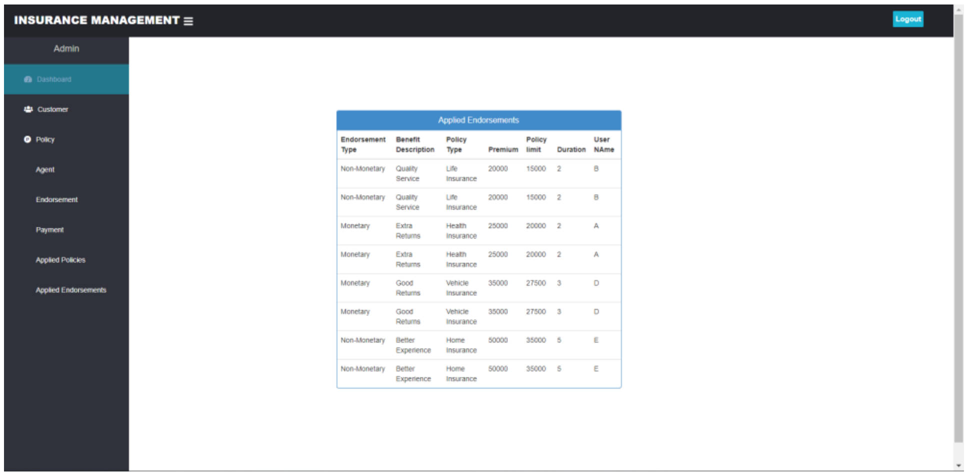


Applied Policies

User Name	Policy	Premium	Policy limit	Duration	Agent_Name
B	Life Insurance	20000	15000	2	Agent_2
A	Health Insurance	25000	20000	2	Agent_3
D	Vehicle Insurance	35000	27500	3	Agent_4
E	Home Insurance	50000	35000	5	Agent_5

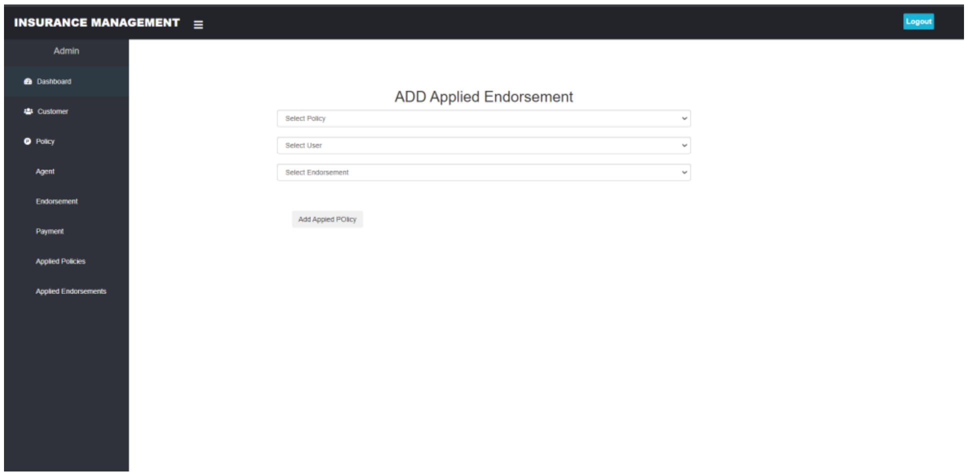






Applied Endorsements

Endorsement Type	Benefit Description	Policy Type	Premium	Policy limit	Duration	User NAme
Non-Monetary	Quality Service	Life Insurance	20000	15000	2	B
Non-Monetary	Quality Service	Life Insurance	20000	15000	2	B
Monetary	Extra Returns	Health Insurance	25000	20000	2	A
Monetary	Extra Returns	Health Insurance	25000	20000	2	A
Monetary	Good Returns	Vehicle Insurance	35000	27500	3	D
Monetary	Good Returns	Vehicle Insurance	35000	27500	3	D
Non-Monetary	Better Experience	Home Insurance	50000	35000	5	E
Non-Monetary	Better Experience	Home Insurance	50000	35000	5	E



ADD Applied Endorsement

Select Policy

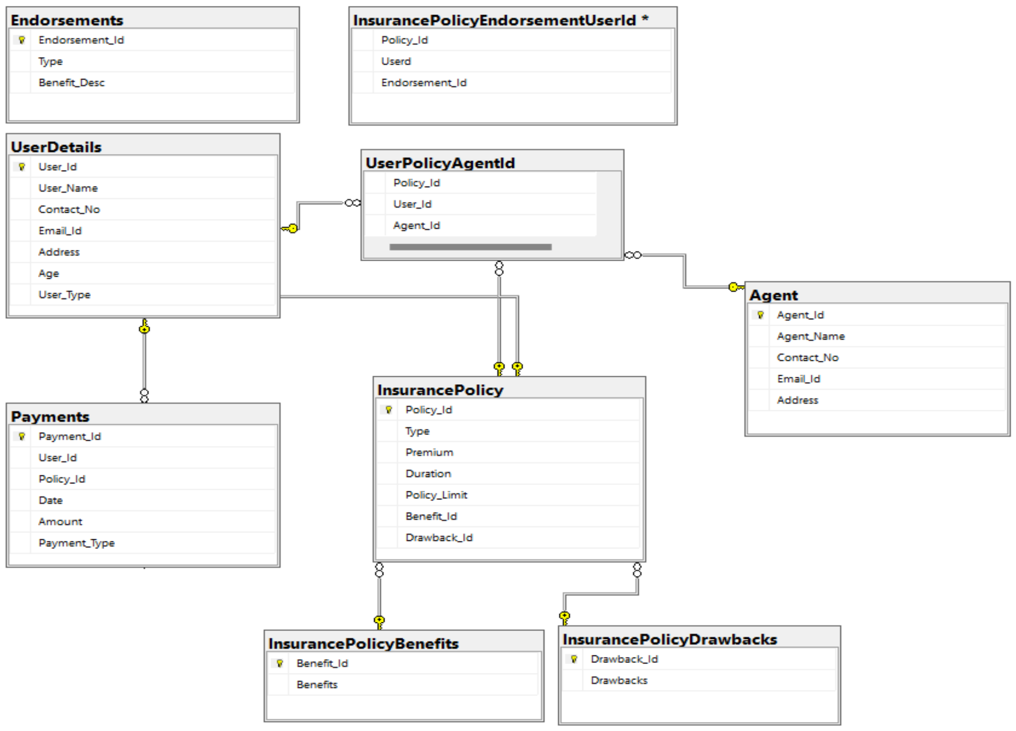
Select User

Select Endorsement

Add Applied POlicy

4. Relational Schema in 3NF – showing the scope

Schema:



Explain it is in 3NF:

3NF: A given relation is said to be in its third normal form when it's in 2NF but has no transitive partial dependency. Meaning, when no transitive dependency exists for the attributes that are non-prime, then the relationship can be said to be in 3NF.

In our Schema:

Policy_Id -> Benefit_Id

Benefit_Id -> Benefit

Super Key: {Policy_id}, {Policy_id, Type}, {Policy_id,Type, Premium}, so on..

Candidiate Key: Policy_Id

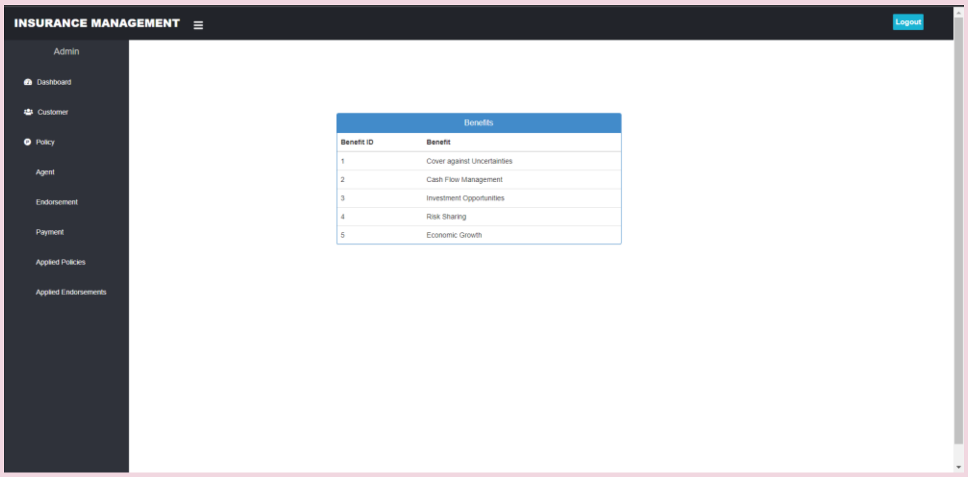
Here Benefit and Drawback are dependent on Benefit_Id and Drawback_Id, which violates 3NF form. Therefore, we create a new table (Benefit_Id, Benefit) and (Drawback_Id and Drawback). Hence, achieving the state of no transitive dependency.

4.1. XML in Schema

XML type is used to store unstructured data in relational database systems. It validates the information inside the file with tags and its values using DTD (Document Type Definition) and schema. It can also be used with the input parameter in a function or stored procedure. The XML type is used here for the purpose of storing the address of Users and Agents. Since, the address contains multiple fields like house number, street number, city, etc. so it is appropriate to use XML type.

5. Implementation in SQL Server

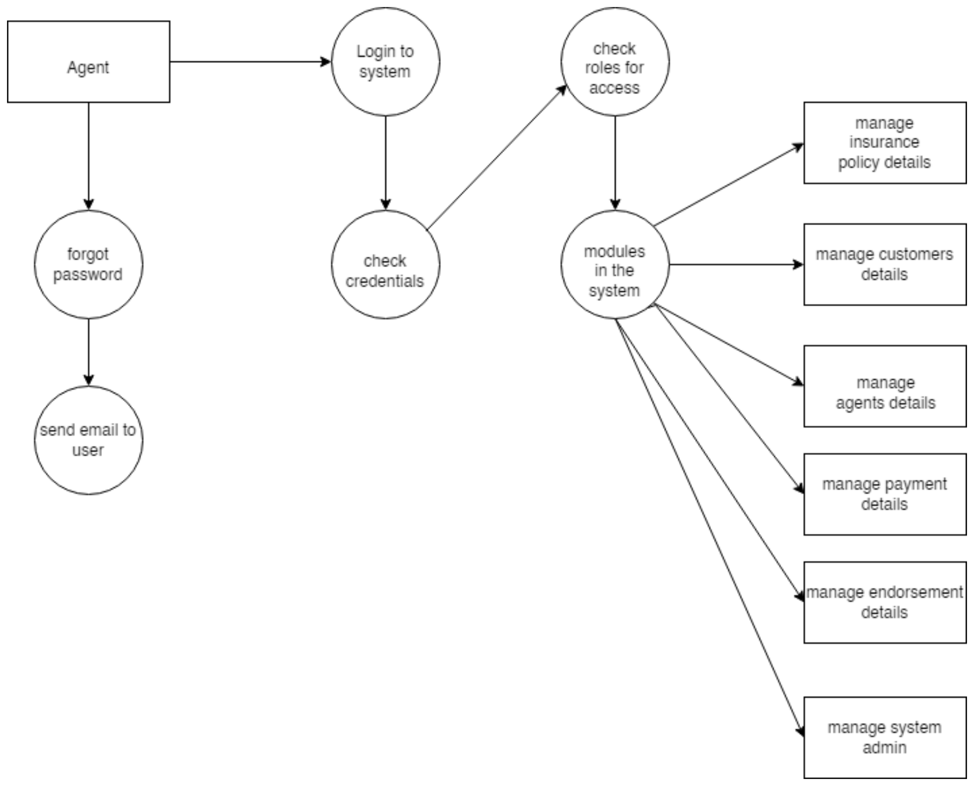
5.1. Tables with Data Diagram



Benefits

Benefit ID	Benefit
1	Cover against Uncertainties
2	Cash Flow Management
3	Investment Opportunities
4	Risk Sharing
5	Economic Growth

Data Diagram:



1) Customer and Payment have a relationship referred to as one-to-many since both have customer id as a common key
2) The customer and agent have a one-to-one relationship since both have a common key as the customer id

5.2. Stored PROCEDURES

A. <u>Insert Procedure</u>

1) To insert customer details

```
CREATE PROCEDURE dbo.insertUserDetails

        (

        @User_Name VARCHAR(30),

        @Contact_No VARCHAR(50),

        @Email_Id VARCHAR(30),
        @Address XML,

        @Age INT,

        @User_Type VARCHAR(30),

        @Password VARCHAR(30)

        )

AS

BEGIN

 INSERT INTO db1.dbo.UserDetails
VALUES(@User_Name,@Contact_No,@Email_Id,@Address,@Age,@User_Type,@P
assword);

END;
```

2) To insert agent details such as name, contact no, Email Id, Address

```
CREATE PROCEDURE dbo.insertAgent

        (

        @Agent_Name varchar(30),

        @Contact_No VARCHAR(50),

        @Email_Id VARCHAR(30),

        @Address XML)

AS

BEGIN

        INSERT INTO db1.dbo.Agent
VALUES(@Agent_Name,@Contact_No,@Email_Id,@Address);

END;
```

3) To insert insurance policy benefits

```
CREATE PROCEDURE dbo.insertInsurancePolicyBenefits(
        @Benefits text)
AS
BEGIN
        INSERT INTO db1.dbo.InsurancePolicyBenefits VALUES(@Benefits);
END;
```

4) To insert insurance policy drawbacks

```
CREATE PROCEDURE dbo.insertInsurancePolicyDrawbacks(
        @Drawbacks text)
AS
BEGIN
        INSERT INTO db1.dbo.InsurancePolicyDrawbacks VALUES(@Drawbacks);
END;
```

5) To insert as well as create an insurance policy

```
CREATE PROCEDURE dbo.insertInsurancePolicy
        (
        @Type VARCHAR(50),
        @Premium INT,
        @Duration INT,
        @Policy_Limit INT,
        @Benefit_Id INT,
        @Drawback_Id INT
        )
AS
BEGIN
        INSERT INTO db1.dbo.InsurancePolicy VALUES
(@Type,@Premium,@Duration,@Policy_Limit,@Benefit_Id,@Drawback_Id);
END;
```

6) To insert endorsements

```
CREATE PROCEDURE dbo.insertEndorsements(
        @Type VARCHAR(50),
        @Benefit_Desc TEXT)
AS
BEGIN
        INSERT INTO db1.dbo.Endorsements VALUES(@Type,@Benefit_Desc);
END;
```

7) To insert payments details

```
CREATE PROCEDURE dbo.insertPayments(
        @Customer_Id INT,
        @Policy_Id INT,
        @Date DATE,
        @Amount INT,
        @Payment_Type VARCHAR(30))
AS
```

```
BEGIN

        INSERT INTO db1.dbo.Payments
VALUES(@Customer_Id,@Policy_Id,@Date,@Amount,@Payment_Type);

END;
```

8) To insert agent details assigned to customers with their insurance policy

```
CREATE PROCEDURE dbo.insertUserPolicyAgentId(

        @Policy_Id INT,

        @User_Id INT,

        @Agent_Id INT

)

AS

BEGIN

        INSERT INTO db1.dbo.UserPolicyAgentId
VALUES(@Policy_Id,@User_Id,@Agent_Id);

END;
```

9) To insert insurance policy endorsement of customer

```
CREATE PROCEDURE dbo.insertInsurancePolicyEndorsementUserId(

        @Policy_Id INT,

        @User_Id INT,

        @Endorsement_Id INT

)

AS

BEGIN
```

```
        INSERT INTO db1.dbo.InsurancePolicyEndorsementUserId
VALUES(@Policy_Id,@User_Id,@Endorsement_Id);

END;
```

B. Delete Procedure

1) To delete a customer account

```
CREATE PROCEDURE dbo.deleteUser(@User_Id INT)

AS

BEGIN

        DELETE FROM db1.dbo.UserDetails WHERE User_Id = @User_Id;

END;


EXEC deleteUser 1;

SELECT * FROM db1.dbo.UserDetails;

SELECT * FROM db1.dbo.InsurancePolicyEndorsementUserId;
```

2) To delete the agent account

```
CREATE PROCEDURE dbo.deleteAgent(@Agent_Id INT)

AS

BEGIN

        DELETE FROM db1.dbo.Agent WHERE Agent_Id = @Agent_Id;

END;

EXEC deleteAgent 1;
```

3) To delete the insurance policy

```
CREATE PROCEDURE dbo.deleteInsurancePolicy(@Policy_Id INT)
AS
BEGIN
        DELETE FROM db1.dbo.InsurancePolicy WHERE Policy_Id = @Policy_Id;
END;
EXEC deleteInsurancePolicy 1;
```

C. Update Procedure

1) Customers should be able to update their password

```
--Procedure to update Users Password
CREATE PROCEDURE dbo.updatePassword(
        @User_Id INT,
        @Password VARCHAR(30))
AS
BEGIN
        UPDATE db1.dbo.UserDetails SET Password = @Password WHERE User_Id =
@User_Id;
END;

EXEC updatePassword 1,'abcd';
```

2) Customers should be able to update their contact no.

```
--PROCEDURE TO UPDATE User'S CONTACT NUMBER
CREATE PROCEDURE dbo.updateUserContact(
        @User_Id INT,
        @Contact_No VARCHAR(50))
AS
BEGIN
        UPDATE db1.dbo.UserDetails SET Contact_No = @Contact_No WHERE User_Id = @User_Id;
END;


EXEC updateUserContact 1,2222222222;
SELECT * FROM db1.dbo.UserDetails;
```

3) Customer Should be able to update their address

```
--PROCEDURE TO UPDATE User'S  Address
CREATE PROCEDURE dbo.updateUserAddress(
        @User_Id INT,
        @Address XML)
AS
BEGIN
        UPDATE db1.dbo.UserDetails SET Address= @Address where User_Id = @User_Id;
END;


EXEC updateUserAddress 1,'<Address Address="2" />';
SELECT * FROM db1.dbo.UserDetails;
```

4) Agent should be able to update their address

```
--PROCEDURE TOM UPDATE AGENT'S ADDRESS

CREATE PROCEDURE dbo.updateAgentAddress(

        @Agent_Id INT,

        @Address XML)

AS

BEGIN

        UPDATE db1.dbo.Agent SET Address= @Address where Agent_Id = @Agent_Id;

END;


EXEC updateAgentAddress 1,'<Address Address="3" />';

SELECT * FROM db1.dbo.Agent;
```

5) Agent should be able to update their contact no.

```
--PROCEDURE TOM UPDATE AGENT'S CONTACT NUMBER

CREATE PROCEDURE dbo.updateAgentContact(

        @Agent_Id INT,

        @Contact_No VARCHAR(50))

AS

BEGIN

        UPDATE db1.dbo.Agent SET Contact_No = @Contact_No WHERE Agent_Id = @Agent_Id;

END;


EXEC updateAgentContact 1,2222222222;
```

```
SELECT * FROM db1.dbo.Agent;
```

6) To update the insurance policy premium

```
--PROCEDURE TO UPDATE INSURANCE_POLICY PREMIUM

CREATE PROCEDURE dbo.updatePolicyPremium(

        @Policy_Id INT,

        @Premium INT)

AS

BEGIN

        UPDATE db1.dbo.InsurancePolicy SET Premium = @Premium WHERE
Policy_Id = @Policy_Id;

END;


EXEC updatePolicyPremium 1,20000;

SELECT * FROM db1.dbo.InsurancePolicy;
```

7) To update general information about the insurance policy such as policy duration and limit

```
--PROCEDURE TO UPDATE INSURANCE_POLICY DURATION

CREATE PROCEDURE dbo.updatePolicyDuration(

        @Policy_Id INT,

        @Duration INT)

AS

BEGIN
```

```
        UPDATE db1.dbo.InsurancePolicy SET Duration = @Duration WHERE
Policy_Id = @Policy_Id;
END;


EXEC updatePolicyDuration 1,3;
SELECT * FROM db1.dbo.InsurancePolicy;


--PROCEDURE TO UPDATE INSURANCE_POLICY POLICY_LIMIT
CREATE PROCEDURE dbo.updatePolicy_Limit(
        @Policy_Id INT,
        @Policy_Limit INT)
AS
BEGIN
        UPDATE db1.dbo.InsurancePolicy SET Policy_Limit = @Policy_Limit
WHERE Policy_Id = @Policy_Id;
END;


EXEC updatePolicy_Limit 1,3000;
SELECT * FROM db1.dbo.InsurancePolicy;
SELECT * FROM db1.dbo.logInsurancePolicy;
```

5.3. Triggers

```
CREATE TRIGGER dbo.IPI ON db1.dbo.InsurancePolicy
FOR INSERT
```

```
AS

BEGIN

        DECLARE @Policy_Id INT;

        DECLARE @Type VARCHAR(50);

        DECLARE @Premium INT;

        DECLARE @Duration INT;

        DECLARE @Policy_Limit INT;

        DECLARE @Benefit_Id INT;

        DECLARE @Drawback_Id INT;

        SELECT @Policy_Id = Policy_Id from inserted;

        SELECT @Type = Type from inserted;

        SELECT @Premium = Premium from inserted;

        SELECT @Duration = Duration from inserted;

        SELECT @Policy_Limit = Policy_Limit from inserted;

        SELECT @Benefit_Id = Benefit_Id from inserted;

        SELECT @Drawback_Id = Drawback_Id from inserted;


        INSERT INTO db1.dbo.logInsurancePolicy
VALUES(@Policy_Id,@Type,@Premium,@Duration,@Policy_Limit,@Benefit_Id,@Drawbac
k_Id);

END;


CREATE TRIGGER dbo.IPU ON db1.dbo.InsurancePolicy

FOR UPDATE

AS
```

```
BEGIN
	DECLARE @Policy_Id INT;
	DECLARE @Type VARCHAR(50);
	DECLARE @Premium INT;
	DECLARE @Duration INT;
	DECLARE @Policy_Limit INT;
	DECLARE @Benefit_Id INT;
	DECLARE @Drawback_Id INT;
	SELECT @Policy_Id = Policy_Id from inserted;
	SELECT @Type = Type from inserted;
	SELECT @Premium = Premium from inserted;
	SELECT @Duration = Duration from inserted;
	SELECT @Policy_Limit = Policy_Limit from inserted;
	SELECT @Benefit_Id = Benefit_Id from inserted;
	SELECT @Drawback_Id = Drawback_Id from inserted;

	INSERT INTO db1.dbo.logInsurancePolicy
VALUES(@Policy_Id,@Type,@Premium,@Duration,@Policy_Limit,@Benefit_Id,@Drawbac
k_Id);
END;

CREATE TRIGGER dbo.ci ON db1.dbo.UserDetails
FOR INSERT,UPDATE
AS
BEGIN
```

```
        DECLARE @User_Id INT;

        DECLARE @User_Name VARCHAR(30);

        DECLARE     @Address XML;

        DECLARE     @Age INT;

        DECLARE     @Contact_No VARCHAR(50);

        DECLARE     @Email_Id VARCHAR(30);

        DECLARE     @User_Type VARCHAR(30);

        DECLARE @Password VARCHAR(30);

        SELECT @User_Id = User_Id from inserted;

        SELECT @User_Name =User_Name  from inserted;

        SELECT @Address = Address from inserted;

        SELECT @Age  = Age from inserted;

        SELECT @Contact_No = Contact_No from inserted;

        SELECT @Email_Id = Email_Id from inserted;

        SELECT @User_Type = User_Type from inserted;

        SELECT @Password = Password from inserted;

        INSERT INTO db1.dbo.logUserDetails
VALUES(@User_Id,@User_Name,@Contact_No,@Email_Id,@Address,@Age,@User_Type,
@Password);


END;


CREATE TRIGGER dbo.ai ON db1.dbo.Agent

FOR INSERT,UPDATE

AS
```

```
BEGIN

        DECLARE @Agent_Id INT;

        DECLARE @Agent_Name VARCHAR(30);

        DECLARE     @Address XML;

        DECLARE     @Contact_No VARCHAR(50);

        DECLARE     @Email_Id VARCHAR(30);

        SELECT @Agent_Id =Agent_Id from inserted;

        SELECT @Agent_Name =Agent_Name   from inserted;

        SELECT @Contact_No = Contact_No from inserted;

        SELECT @Email_Id = Email_Id from inserted;

        SELECT @Address = Address from inserted;


        INSERT INTO db1.dbo.logAgent VALUES(@Agent_Id,@Agent_Name
,@Contact_No,@Email_Id,@Address);


END;
```

5.4. Views and benefits of using views

Views:

A view is a virtual table whose contents are defined by a query. Like a table, a view consists of a set of named columns and rows of data. Unless indexed, a view does not exist as a stored set of data values in a database. The rows and columns of data come from tables referenced in the query defining the view and are produced dynamically when the view is referenced.

Benefits of Using Views:

Views are generally used to focus, simplify, and customize the perception each user has of the database. Views can be used as security mechanisms by letting users access data through the view, without granting the users permissions to directly access the underlying base tables of the view. Views can be used to provide a backward compatible interface to emulate a table that used to exist but whose schema has changed. Views can also be used when you copy data to and from SQL Server to improve performance and to partition data.

```
CREATE VIEW dbo.IP

AS

SELECT dbo.InsurancePolicy.Policy_Id, dbo.InsurancePolicy.Premium,
dbo.InsurancePolicy.Type,

dbo.InsurancePolicy.Duration, dbo.InsurancePolicy.Policy_Limit,
dbo.InsurancePolicyBenefits.Benefits,

dbo.InsurancePolicyDrawbacks.Drawbacks

FROM dbo.InsurancePolicy INNER JOIN dbo.InsurancePolicyBenefits ON

dbo.InsurancePolicy.Benefit_Id = dbo.InsurancePolicyBenefits.Benefit_Id INNER JOIN

dbo.InsurancePolicyDrawbacks ON

dbo.InsurancePolicy.Drawback_Id = dbo.InsurancePolicyDrawbacks.Drawback_Id;


SELECT * from dbo.IP;


SELECT a.Type,SUM(a.Premium) FROM db1.dbo.InsurancePolicy a GROUP BY a.Type
HAVING a.Type = 'Health Insurance';

SELECT a.Type,AVG(a.Premium) FROM db1.dbo.InsurancePolicy a GROUP BY a.Type
HAVING a.Type = 'Health Insurance';

SELECT a.Type,MAX(a.Premium) FROM db1.dbo.InsurancePolicy a GROUP BY a.Type
HAVING a.Type = 'Health Insurance';

SELECT a.Type,MIN(a.Premium) FROM db1.dbo.InsurancePolicy a GROUP BY a.Type
HAVING a.Type = 'Health Insurance';


EXEC insertUserDetails 3,'abc','1111111111','abc@gmail.com','<Address userid="3">
```

```
                    <HouseNumber>3</HouseNumber>

                    <StreetName>ABCD</StreetName>

                    <CityName>ABCD</CityName>

                    </Address>',20,'customer';


SELECT Address.query('/Address') FROM db1.dbo.UserDetails;

DECLARE @xmldata XML;

SET @xmldata = '<Address userid="2">

        <HouseNumber>@xml</HouseNumber>

        <StreetName>efgh</StreetName>

        <CityName>efgh</CityName>

</Address>';

UPDATE db1.dbo.UserDetails

SET Address = @xmldata WHERE User_Id = 3;


CREATE VIEW dbo.view1

AS

SELECT db1.dbo.Agent.Agent_Name, db1.dbo.InsurancePolicy.Type,
db1.dbo.InsurancePolicy.Premium,

db1.dbo.InsurancePolicy.Duration, db1.dbo.InsurancePolicy.Policy_Limit,
db1.dbo.UserDetails.User_Name

FROM    db1.dbo.Agent CROSS JOIN

                db1.dbo.InsurancePolicy CROSS JOIN
```

db1.dbo.UserDetails;

6. Conclusions

A computerized method for managing insurance has been created and tested using test data. The advantages of a computer system over an existing one are significant in terms of the time and effort that may be saved by human labor.
The system has the ability to add, update, and delete customers, agents, policies, payments, and endorsements, enabling companies to obtain accurate and timely information systems. It facilitates effective system activity monitoring, which improves decision-making.

7. INNOVATION

Insurance Management System is an absolute system which allows insurance companies to not only manage its policies and customers but we introduce management of Premium payments done by customers for specific policies. We also initiate the management of Agents who work for the company and connect to people to sell the policy.
An endorsement is a small change or benefit added to the existing policies.
The customers can opt for endorsement of interest considering the limit of policy. We allow the company to handle endorsements.
We have used the Auto-increment feature for primary keys of every table in the Insurance management system to prevent any ambiguity or error while inserting a new tuple to the table. We have restricted the insertion of null values in the mandatory columns by using the NOT NULL constraint.
We also ensure that all values are different for columns such as username, contact number and email Id to avoid any errors.

8. Bibliography

1. markingmyname (n.d.). SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS). [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16.
2. npm. (n.d.). mssql. [online] Available at: https://www.npmjs.com/package/mssql.
3. Allstate. (2021). What Is an Insurance Endorsement? | Allstate. [online] Available at: https://www.allstate.com/resources/what-is-an-insurance-endorsement [Accessed 25 Dec. 2022].
4.
5.

Appendix A – Create table Queries

All CREATE TABLE queries can be given as follows:

```
CREATE TABLE db1.dbo.UserDetails(
        User_Id INT IDENTITY(1,1) PRIMARY KEY,
        User_Name VARCHAR(30) UNIQUE NOT NULL,
        Contact_No VARCHAR(50) UNIQUE NOT NULL,
        Email_Id VARCHAR(30) UNIQUE NOT NULL,
        Address XML NOT NULL,
        Age INT NOT NULL,
        User_Type VARCHAR(30) NOT NULL,
        Password VARCHAR(30) NOT NULL
);

CREATE TABLE db1.dbo.UserType(User_Type VARCHAR(30) PRIMARY KEY);
ALTER TABLE db1.dbo.UserDetails
ADD FOREIGN KEY (User_Type) REFERENCES UserType(User_Type);
INSERT INTO db1.dbo.UserType VALUES('admin');
INSERT INTO db1.dbo.UserType VALUES('customer');
```

```
CREATE TABLE db1.dbo.Agent(
        Agent_Id INT IDENTITY(1,1) PRIMARY KEY,
        Agent_Name VARCHAR(30) UNIQUE NOT NULL,
        Contact_No VARCHAR(50) UNIQUE NOT NULL,
        Email_Id VARCHAR(30) UNIQUE NOT NULL,
        Address XML
);

CREATE TABLE db1.dbo.InsurancePolicyBenefits(
        Benefit_Id INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
        Benefits text NOT NULL
);


CREATE TABLE db1.dbo.InsurancePolicyDrawbacks(
        Drawback_Id INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
        Drawbacks text NOT NULL
);
CREATE TABLE db1.dbo.InsurancePolicy(
        Policy_Id INT IDENTITY(1,1) PRIMARY KEY,
        Type VARCHAR(50) NOT NULL,
        Premium INT NOT NULL,
        Duration INT NOT NULL,
        Policy_Limit INT NOT NULL,
        Benefit_Id INT,

        Drawback_Id INT

);


CREATE TABLE db1.dbo.PolicyType(Type VARCHAR(50) PRIMARY KEY);

INSERT INTO db1.dbo.PolicyType VALUES('Life Insurance');

INSERT INTO db1.dbo.PolicyType VALUES('Vehicle Insurance');

INSERT INTO db1.dbo.PolicyType VALUES('Health Insurance');

INSERT INTO db1.dbo.PolicyType VALUES('Home Insurance');

INSERT INTO db1.dbo.PolicyType VALUES('Fire Insurance');

INSERT INTO db1.dbo.PolicyType VALUES('Travel Insurance');


SELECT * FROM db1.dbo.PolicyType;


ALTER TABLE       db1.dbo.InsurancePolicy

ADD FOREIGN KEY (Type) REFERENCES PolicyType(Type) ON DELETE CASCADE;


ALTER TABLE db1.dbo.InsurancePolicy

ADD FOREIGN KEY (Benefit_Id) REFERENCES InsurancePolicyBenefits(Benefit_Id) ON
DELETE CASCADE;


ALTER TABLE db1.dbo.InsurancePolicy

ADD FOREIGN KEY (Drawback_Id) REFERENCES
InsurancePolicyDrawbacks(Drawback_Id) ON DELETE CASCADE;
```

```
CREATE TABLE db1.dbo.Payments(
	Payment_Id INT IDENTITY(1,1) PRIMARY KEY,
	User_Id INT NOT NULL,
	Policy_Id INT NOT NULL,
	Date DATE NOT NULL,
	Amount INT NOT NULL,
	Payment_Type VARCHAR(30) NOT NULL
);

ALTER TABLE db1.dbo.Payments
ADD FOREIGN KEY (User_Id) REFERENCES UserDetails(User_Id) ON DELETE
CASCADE;

ALTER TABLE db1.dbo.Payments
ADD FOREIGN KEY (Policy_Id) REFERENCES InsurancePolicy(Policy_Id) ON DELETE
CASCADE;

CREATE TABLE db1.dbo.UserPolicyAgentId(
	Policy_Id INT NOT NULL,
	User_Id INT NOT NULL,
	Agent_Id INT NOT NULL,
);
ALTER TABLE db1.dbo.UserPolicyAgentId
ADD FOREIGN KEY (Policy_Id) REFERENCES InsurancePolicy(Policy_Id) ON DELETE
CASCADE;
```

```
ALTER TABLE db1.dbo.UserPolicyAgentId

ADD FOREIGN KEY (User_Id) REFERENCES UserDetails(User_Id) ON DELETE
CASCADE;


ALTER TABLE db1.dbo.UserPolicyAgentId

ADD FOREIGN KEY (Agent_Id) REFERENCES Agent(Agent_Id) ON DELETE CASCADE;


CREATE TABLE db1.dbo.Endorsements(

        Endorsement_Id INT IDENTITY(1,1) PRIMARY KEY,

        Type VARCHAR(50) NOT NULL,

        Benefit_Desc TEXT NOT NULL

);


CREATE TABLE db1.dbo.EndorsementType(Type VARCHAR(50));

INSERT INTO db1.dbo.EndorsementType VALUES('Monetary');

INSERT INTO db1.dbo.EndorsementType VALUES('Non-Monetary');


CREATE TABLE db1.dbo.InsurancePolicyEndorsementUserId(

        Policy_Id INT NOT NULL,

        User_Id INT NOT NULL,

        Endorsement_Id INT,

);


ALTER TABLE db1.dbo.InsurancePolicyEndorsementUserId
```

```
ADD FOREIGN KEY (Policy_Id) REFERENCES InsurancePolicy(Policy_Id) ON DELETE CASCADE;

ALTER TABLE db1.dbo.InsurancePolicyEndorsementUserId

ADD FOREIGN KEY (User_Id) REFERENCES UserDetails(User_Id) ON DELETE CASCADE;

ALTER TABLE db1.dbo.InsurancePolicyEndorsementUserId

ADD FOREIGN KEY (Endorsement_Id) REFERENCES Endorsements(Endorsement_Id);


CREATE TABLE db1.dbo.logInsurancePolicy(

        Policy_Id INT,

        Type VARCHAR(50) NOT NULL,

        Premium INT NOT NULL,

        Duration INT NOT NULL,

        Policy_Limit INT NOT NULL,

        Benefit_Id INT,

        Drawback_Id INT

);



CREATE TABLE db1.dbo.logUserDetails(

        User_Id INT,

        User_Name VARCHAR(30) NOT NULL,

        Contact_No VARCHAR(50) NOT NULL,

        Email_Id VARCHAR(30) NOT NULL,

        Address XML NOT NULL,
        Age INT NOT NULL,
        User_Type VARCHAR(30) NOT NULL,
        Password VARCHAR(30) NOT NULL
);


CREATE TABLE db1.dbo.logAgent(
        Agent_Id INT,
        Agent_Name varchar(30),
        Contact_No VARCHAR(50) ,
        Email_Id VARCHAR(30),
        Address XML
);
```

Appendix B – INSERT INTO

INSERT INTO queries can be given as follows:

```
EXEC insertUserDetails 'admin','1111111111','admin@gmail.com','<Address userid="1">

                        <HouseNumber>3</HouseNumber>
                        <StreetName>ABCD</StreetName>

                        <CityName>ABCD</CityName>

                        </Address>',40,'admin','admin';


EXEC insertUserDetails 'A','2222222222','A@gmail.com','<Address userid="2">

                <HouseNumber>27</HouseNumber>

                <StreetName>EFGH</StreetName>

                <CityName>EFGH</CityName>

                </Address>',20,'customer','A';
SELECT * FROM db1.dbo.UserDetails;



EXEC insertAgent 'Agent_1','3333333333','Agent1@gmail.com','<Address agentid="2">

                <HouseNumber>30</HouseNumber>

                <StreetName>EFGH</StreetName>

                <CityName>EFGH</CityName>

                </Address>';
```

```
SELECT * FROM db1.dbo.Agent;


EXEC insertInsurancePolicyBenefits 'really good';


SELECT * FROM db1.dbo.InsurancePolicyBenefits;


EXEC insertInsurancePolicyDrawbacks 'really bad';

SELECT * FROM  db1.dbo.InsurancePolicyDrawbacks;




EXEC insertInsurancePolicy 'Health Insurance',10000,2,1234,1,1;

EXEC insertInsurancePolicy 'Life Insurance',20000,3,10000,1,1;

EXEC insertInsurancePolicy 'Vehicle Insurance',30000,4,15000,1,1;

EXEC insertInsurancePolicy 'Life Insurance',40000,5,20000,1,1;

EXEC insertInsurancePolicy 'Health Insurance',50000,6,25000,1,1;

EXEC insertInsurancePolicy 'Life Insurance',60000,7,30000,1,1;


SELECT * FROM  db1.dbo.InsurancePolicy;

SELECT * FROM db1.dbo.logInsurancePolicy;


EXEC insertEndorsements 'Monetary','1';


SELECT * FROM db1.dbo.Endorsements;
```

```
EXEC insertPayments 1,1,'11-11-2011',1200,'OnlineUPI';

SELECT * FROM db1.dbo.Payments;

EXEC insertUserPolicyAgentId 1,1,1;

SELECT * FROM db1.dbo.UserPolicyAgentId;

EXEC insertInsurancePolicyEndorsementUserId 1,1,1;
SELECT * FROM db1.dbo.InsurancePolicyEndorsementUserId;
```