

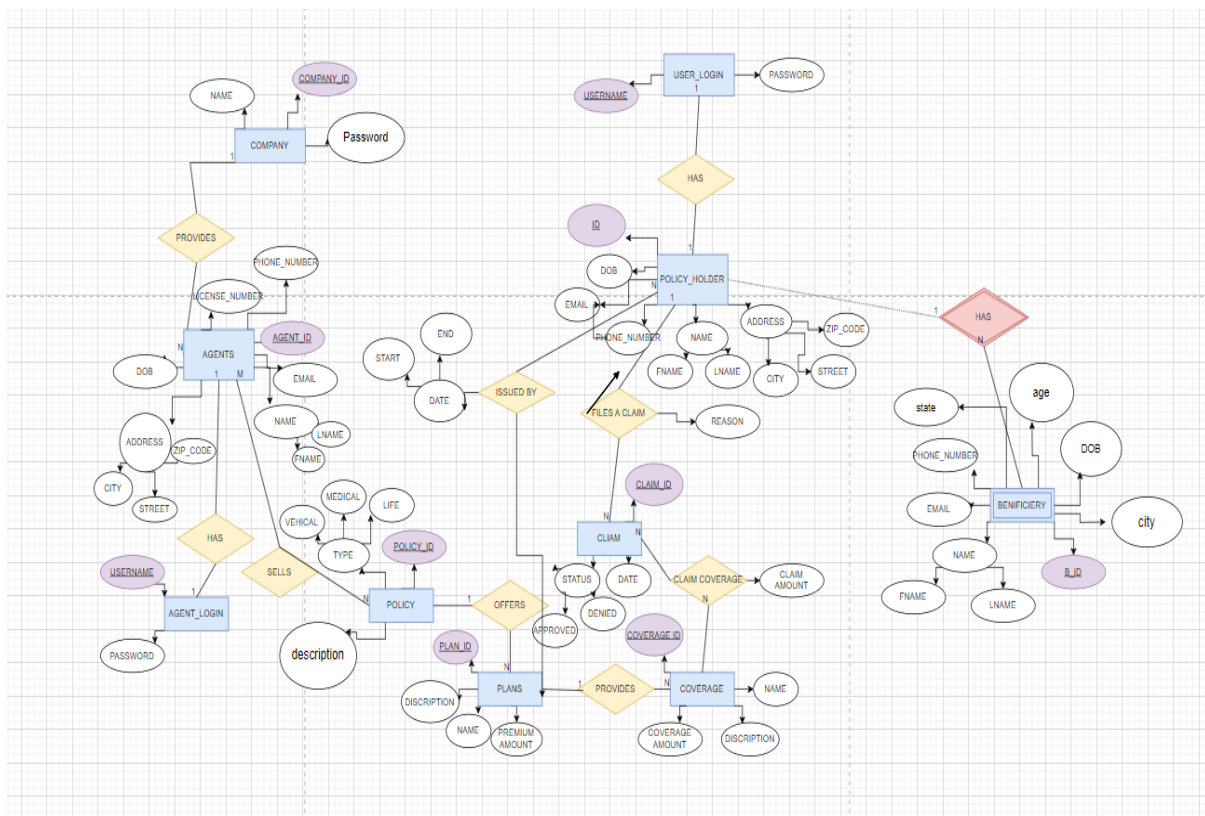
Insurance Company Database Management System

Team Members:

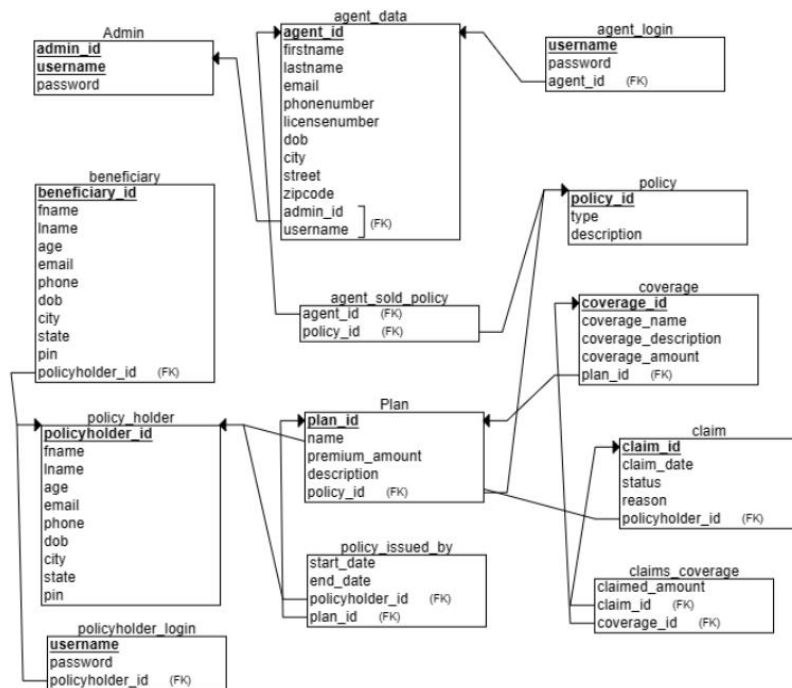
NAME : Sreeja Sahithi Chintalapati
SRN : PES1UG21CS940

NAME: R.Vaishnavi
SRN : PES1UG21CS466

ER Diagram



Relational Schema



MINI PROJECT USER REQUIREMENT SPECIFICATION

Purpose of project:

The purpose of an insurance company database catering to vehicle, medical, and life insurance policies with diverse plans and coverages is to efficiently manage and organise critical data related to policyholders, policies, claims, and financial transactions. It serves as the digital backbone of the insurance operations, supporting functions such as underwriting, premium calculations, claims processing, and customer service. By centralising and structuring data, the database enhances the company's ability to assess risk, customise policies, and provide timely and accurate services to policyholders. Moreover, it plays a crucial role in ensuring regulatory compliance, data security, and effective decision-making, ultimately contributing to the company's mission of offering reliable financial protection to its customers.

Scope of project:

The scope of the insurance company database is comprehensive, covering a wide range of functionalities within the insurance domain. It includes the storage and management of policyholder information, policy details, coverage options, claims data, premium payments, and related financial records. This scope extends to supporting the intricacies of different insurance types, including vehicle, medical, and life insurance, each with its unique parameters and requirements. Additionally, the database must handle various plans and coverages, allowing for flexibility in policy customization. Scalability and adaptability are essential, as the database must accommodate evolving business needs, changing regulations, and advancements in data management technology.

Project Overview:

The insurance company database is the backbone of our operations, seamlessly connecting the complex web of policyholders, insurance products, and claims processes. In an era driven by data, it plays a pivotal role in efficiently managing vast amounts of information. From documenting policyholder details and coverage options to tracking claims and financial transactions, this database ensures the accuracy and accessibility of critical data. In the dynamic landscape of vehicle, medical, and life insurance, it's the linchpin that allows us to offer tailored plans with precision. Scalable and adaptable, it evolves with our business, accommodating changing customer needs, regulatory demands, and technological advancements. With robust security measures in place, it safeguards sensitive data and ensures compliance with industry standards. In essence, our insurance company database is the engine that powers our commitment to providing reliable and responsive financial protection to policyholders, solidifying our place as a trusted partner in safeguarding their futures.

Major project functionalities:

Policy Administration: This functionality is the heart of the database system. It allows the insurance company to create, manage, and maintain various insurance policies. For instance, it enables the creation of policies for vehicle, life, and medical insurance. Within each of these insurance types, the system permits the customization of policies to suit individual needs. Policies can have different durations, coverage amounts, and terms, depending on the customer's preferences and requirements. Policy administration also includes features like policy issuance, modification, renewal, and cancellation, providing flexibility for policyholders to manage their coverage effectively.

Plan and Coverage Customization: Within each policy type, there are various plans and coverages available. This functionality allows policyholders to select from a menu of options to build a policy that aligns precisely with their needs. For example, in a medical insurance policy, a policyholder might choose between different plans that offer varying levels of coverage for hospitalization, doctor visits, or prescription medications. The system manages these choices and ensures that the policyholder receives the appropriate coverage based on their selections.

Claims Management: This crucial functionality handles the entire claims process. When a policyholder needs to make a claim, the system guides them through the process, collecting the necessary information and documentation. It then assesses the claim against the policy's terms and conditions, automating much of the evaluation process. Claims can be related to vehicle accidents, medical expenses, or life insurance payouts, each requiring different documentation and procedures. The system ensures that the correct procedures are followed and that legitimate claims are processed promptly.

Premium Payments: To ensure the continuity of coverage, policyholders need to pay their premiums regularly. The premium payment functionality allows policyholders to make payments conveniently through various methods, such as online payments, direct debit, or credit card payments. The system tracks payment histories, reducing the risk of policy lapses due to missed payments.

Beneficiary Management: This feature manages beneficiary information, a critical aspect of life insurance policies. Policyholders can designate one or more beneficiaries who will receive the policy's benefits in case of the policyholder's death. The system securely stores beneficiary details, ensuring that the right individuals receive the intended benefits. It also allows policyholders to update beneficiary information as needed, accommodating life changes like marriages, divorces, or the birth of children.

User-Friendly Interface:

A user-friendly interface is essential for all these functionalities. An intuitive interface ensures that both insurance company staff and policyholders can interact with the system easily. It includes dashboards, forms, and online tools that simplify policy management, claims filing, premium payments, and beneficiary updates.

System Features and Functional Requirements :

1. The Insurance Company offers 3 different types of policies which include
Vehicle insurance , Life insurance , Medical insurance.
The company provides multiple agents who sell the above mentioned insurance policies.
The entities involved are -:
Insurance Company, Agents, Policy.
2. Each policy offers many plans based on different premium amount and benefits each of these provide different forms of coverage .
The entities involved are -:
Policy, Plan, Coverage.
3. Each Policy holder can buy different types of policies and choose a suitable plan for the policy that they buy .
Each policy holder can file a claim based on their need and this claim will be fulfilled if it meets the requirements of the coverage .
The policy holder will get to know if the claim is approved or denied or pending based upon the status of the claim.
Claims can be related to vehicle accidents, medical expenses, or life insurance payouts, each requiring different documentation and procedures

The entities involved are-:
Policy, Plan, Policy holder, Claims, Coverage.
4. Each policy holder can make a premium payment for the policy that they bought using different methods such as net-banking , credit card, debit card , cash.

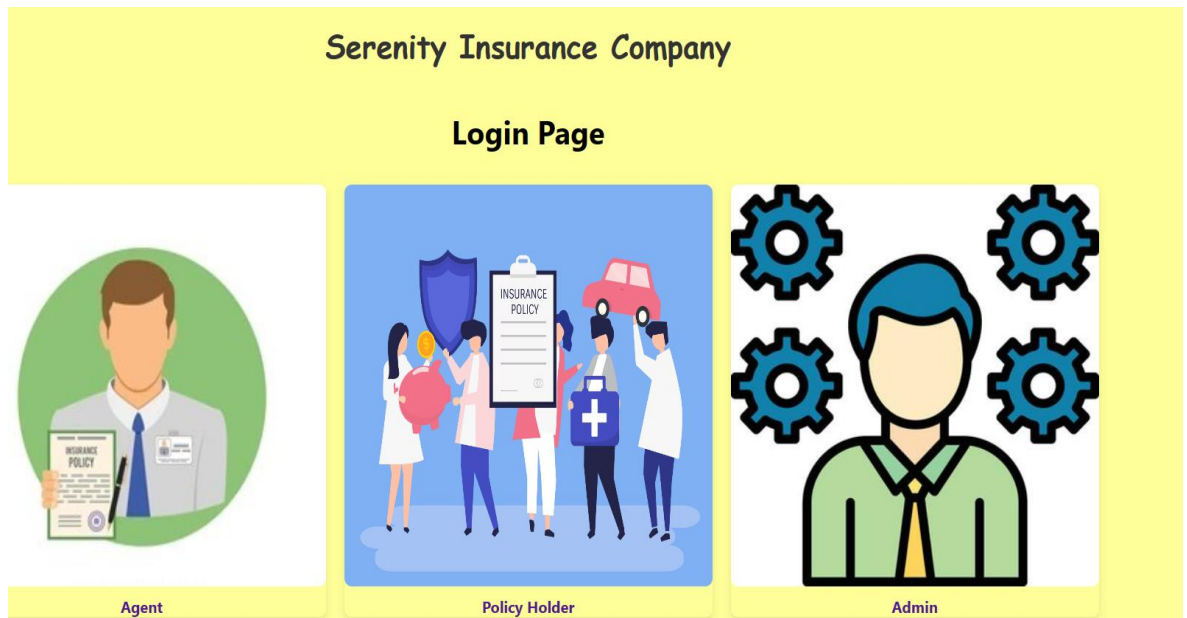
The system tracks payment histories, reducing the risk of policy lapses due to missed payments.
The entities involved are-:
Policy holder, Premium payment
5. Each policy holder can have multiple beneficiaries , these beneficiaries benefit from the claims made by the policy holder.
It also allows policyholders to update beneficiary information as needed, accommodating life changes like marriages, divorces, or the birth of children.
The entities involved are-:
Beneficiary , Policy holder

6. Sign-up/Login -There will be a prompt given where in they can choose if they are an agent or a user who wants to buy a policy .An account is created for user/Agent if they are signing in and their details are stored .

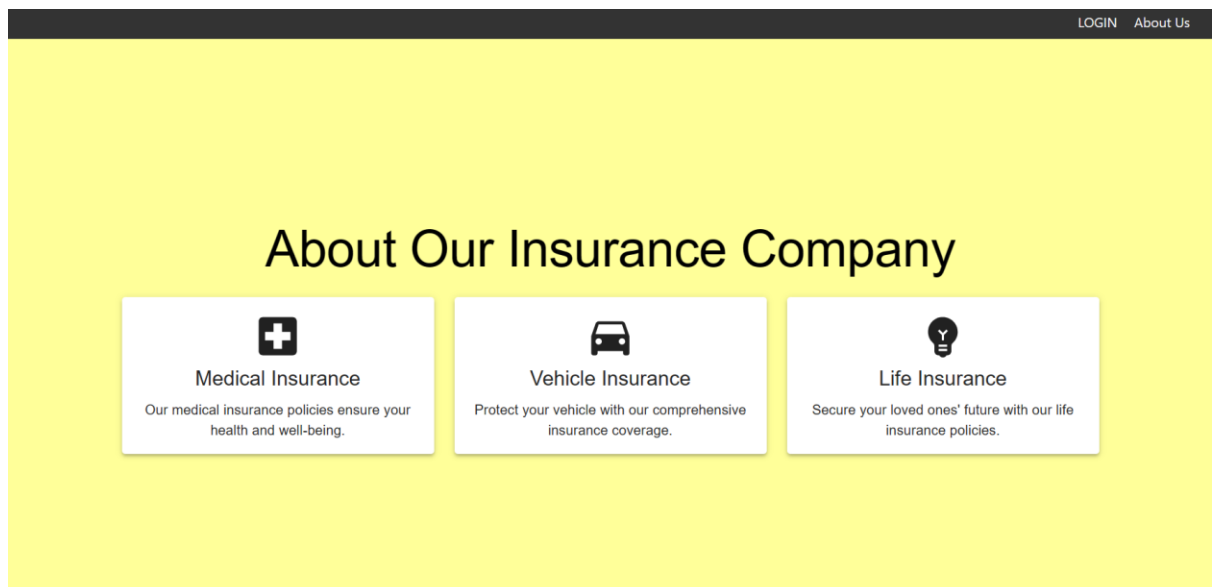
The entities involved are:-

Agent_login, User_login

Home page:-



About us:-

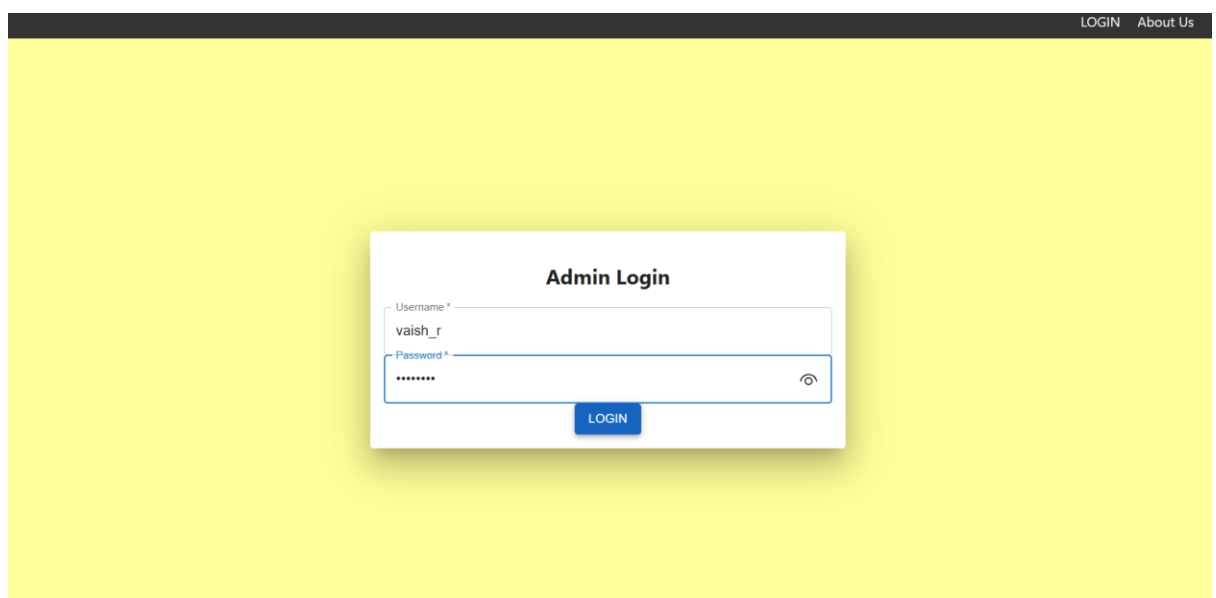


ADMIN- Admin Login:-

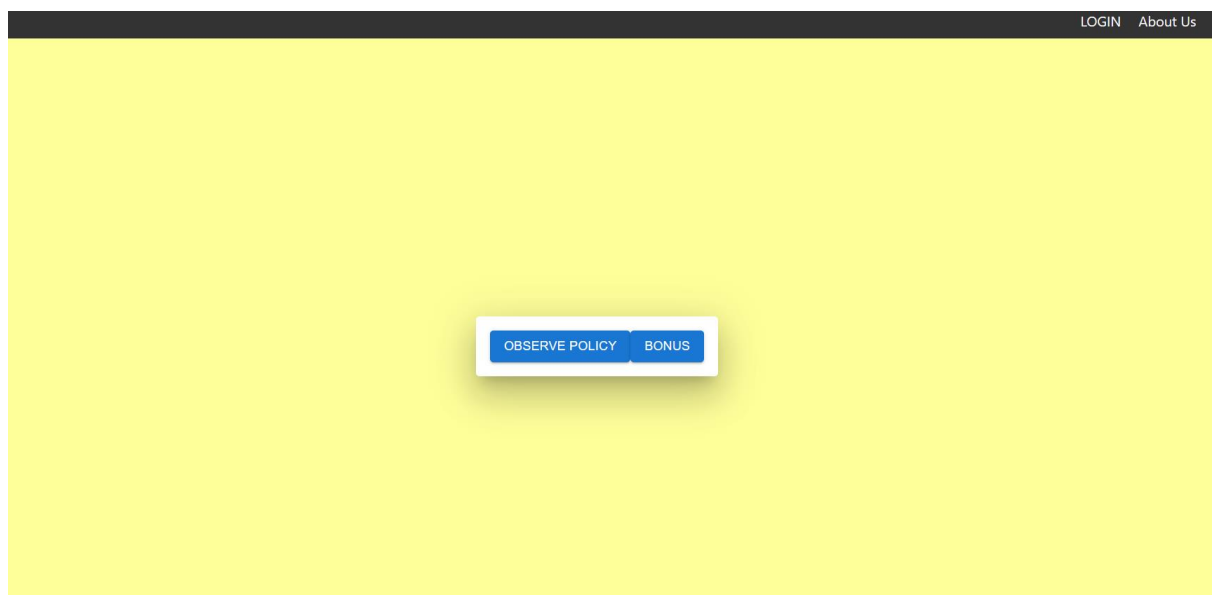
SQL QUERY-

```
CREATE ROLE 'Admin_role';  
GRANT ALL PRIVILEGES ON insurance_db.* TO 'Admin_role';  
CREATE USER 'vaish_r'@'localhost' IDENTIFIED BY 'vaish123';  
GRANT Admin_role TO 'vaish_r'@'localhost';
```

GUI-



The screenshot shows a web application interface with a yellow background. At the top right, there are links for "LOGIN" and "About Us". In the center, there is a white box titled "Admin Login". Inside this box, there are two input fields: "Username *" with the value "vaish_r" and "Password *" with masked characters "*****". A blue "LOGIN" button is positioned below the password field. A small eye icon is visible next to the password field to toggle visibility.



The screenshot shows the bottom section of the web application interface, which has a yellow background. At the top right, there are links for "LOGIN" and "About Us". In the center, there is a white box containing two blue buttons: "OBSERVE POLICY" and "BONUS".

TO OBSERVE THE POLICY SALE ON ADMIN SIDE-

SQL QUERY:

```
`SELECT
    policy_id,
    SUM(count_plan_id) AS total_count_plan_id,
    MAX(test1.policy_type) AS policy_type
FROM
    (
        SELECT
            plan_id,
            COUNT(plan_id) AS count_plan_id,
            MAX(test.policy_id) AS policy_id,
            MAX(test.policy_type) AS policy_type
        FROM
            (
                SELECT
                    pib.plan_id,
                    pib.policyholder_id,
                    p.type AS policy_type,
                    p.policy_id,
                    pl.name AS plan_name
                FROM
                    policy_issued_by AS pib
                JOIN plan AS pl ON pib.plan_id = pl.plan_id
                JOIN policy AS p ON pl.policy_id = p.policy_id
            ) AS test
        GROUP BY
            plan_id
    ) AS test1
GROUP BY
    policy_id`
```


GUI:



AGENT BONUS -

SQL QUERY-

```
CREATE PROCEDURE CalculateBonus(IN agent_username VARCHAR(255), OUT  
bonus_amount INT)
```

```
BEGIN
```

```
    DECLARE agent_id INT;
```

```
    DECLARE sold_policies_count INT;
```

```
    SELECT agent_id INTO agent_id
```

```
    FROM agent_data
```

```
    WHERE agent_data.username = agent_username;
```

```
    SELECT COUNT(*) INTO sold_policies_count
```

```
    FROM agent_sold_policy
```

```
    WHERE agent_id = agent_id;
```

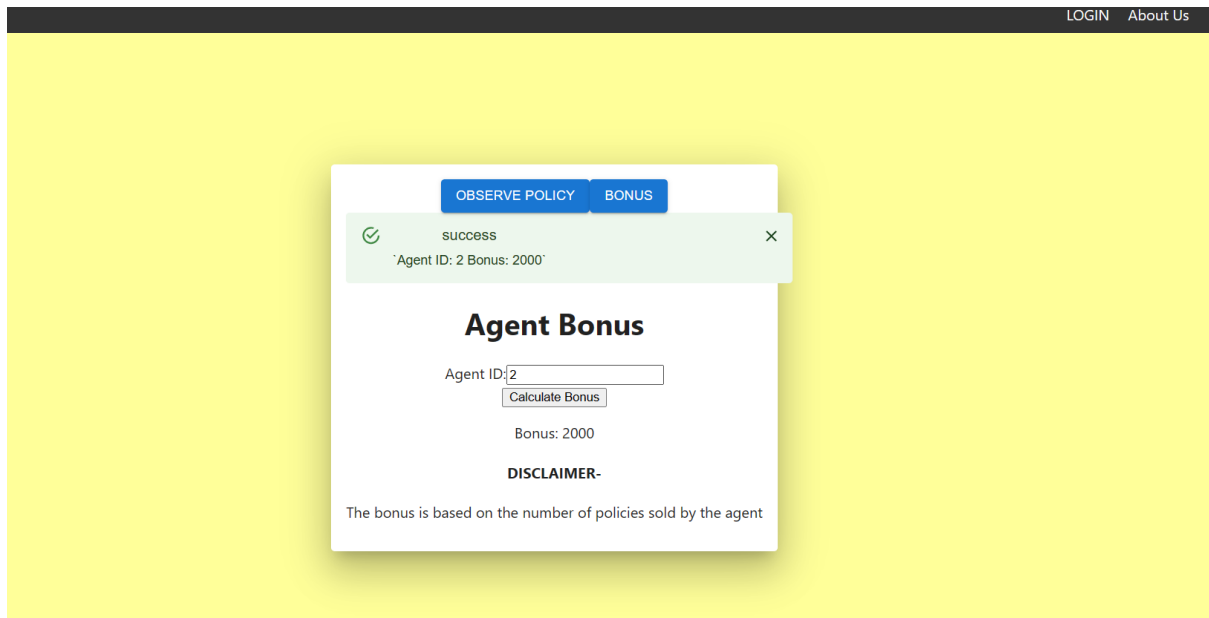
```
    SET bonus_amount = sold_policies_count * 1000;
```

```
END //
```

```
DELIMITER ;
```

```
CALL CalculateBonusForAgent(2);
```

GUI -



Agent Register:

SQL Query:

```
INSERT INTO agent_data
(firstname,lastname,email,phonenumber,licensenummer,dob,city,street,zipcode) VALUES
(max,mec,max@gmail.com,45698234,3456,08-06-1989,Bengaluru,jpnagar,560078);
```

```
INSERT INTO agent_login (username, password,agent_id) VALUES
(max_m,max123,5)
```

GUI:

Agent Registration form

First Name *	Last Name *
max	mec
Email *	
max@gmail.com	
Phone Number *	
45698234	
License Number *	
3456	
Date of Birth *	
08-06-1989	
City *	
Bengaluru	
Street *	Zip Code *
jpnagar	560078
Username *	
max_m	
Password *	

REGISTER

Registration Successful

```
mysql> select * from agent_data;
```

agent_id	firstname	lastname	email	phonenummer	licensenummer	dob	city	street	zipcode
1	jhon	brown	jhon@gmail.com	85768586	568456	1989-10-24	Bengaluru	jpnagar	560078
2	prarthana	bhat	bhat@gmail.com	56798	4567	1997-06-19	Bengaluru	jpnagar	560078
3	srinika	sai	sai@gmail.com	34567845	3456	1989-06-15	Bengaluru	jpnagar	560078
4	sirisha	c	siri@gmail.com	56789034	4567	1986-06-12	Bengaluru	jpnagar	560078
5	max	mec	max@gmail.com	45698234	3456	1989-06-08	Bengaluru	jpnagar	560078

5 rows in set (0.00 sec)

```
mysql> select * from agent_login;
+-----+-----+-----+
| agent_id | username | password |
+-----+-----+-----+
| 1 | jhon_brown | brown123 |
| 5 | max_m | max123 |
| 2 | prarth | p123 |
| 4 | sirisha_c | siri123 |
| 3 | sri_sai | sri123 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Agent Login:

SQL Query:

```
SELECT * FROM agent_login WHERE username =john_brown;
```

GUI-

[LOGIN](#)
[About Us](#)

Agent Login

Username *

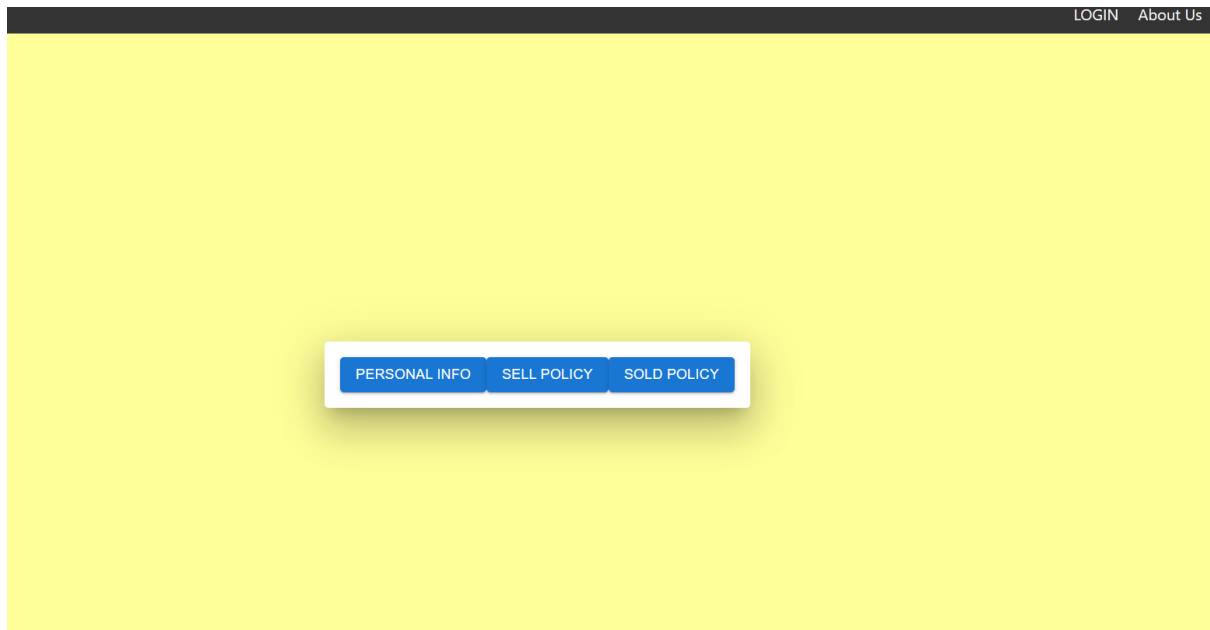
jhon_brown

Password *

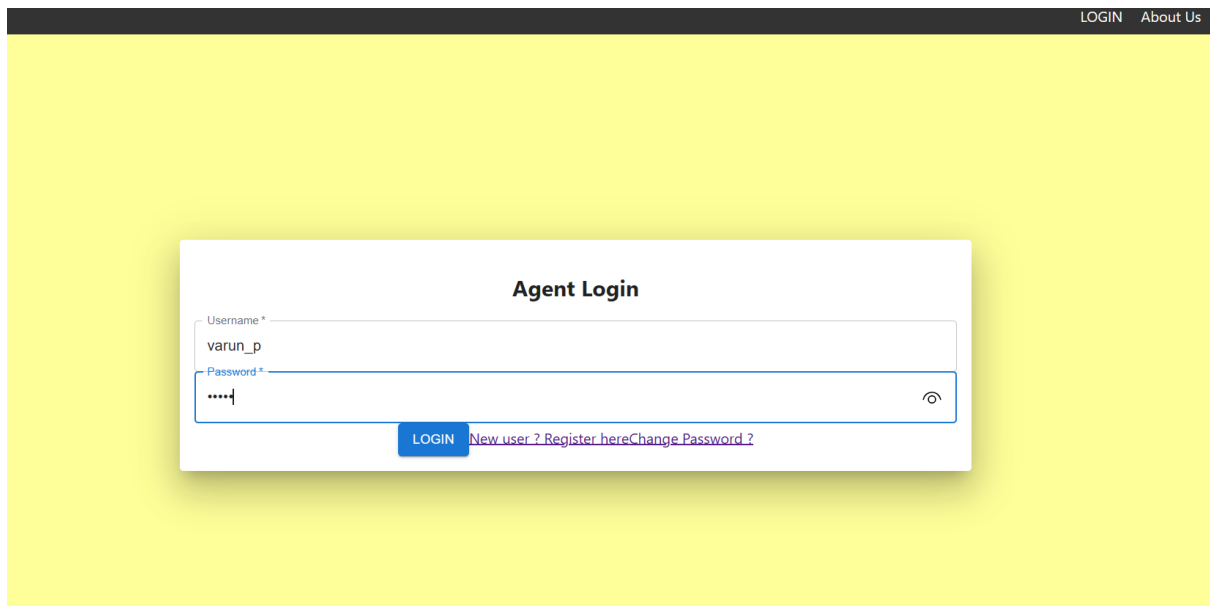
.....

LOGIN

[New user ? Register here](#)
[Change Password ?](#)



INVALID LOGIN-



Invalid Username or Password

[New user ? Register here](#)

CHANGE PASSWORD-

SQL QUERY-

UPDATE agent_login SET password = "sirisha_c" WHERE username = s123;

GUI -

Change Password

Username *
sirisha_c

New_Password *



CHANGE

Before change-

```
mysql> select * from agent_login;
+-----+-----+-----+
| agent_id | username | password |
+-----+-----+-----+
| 1 | jhon_brown | brown123 |
| 5 | max_m | max123 |
| 2 | prarth | p123 |
| 4 | sirisha_c | siri123 |
| 3 | sri_sai | sri123 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

After change-

```
mysql> select * from agent_login;
+-----+-----+-----+
| agent_id | username | password |
+-----+-----+-----+
| 1 | jhon_brown | brown123 |
| 5 | max_m | max123 |
| 2 | prarth | p123 |
| 4 | sirisha_c | s123 |
| 3 | sri_sai | sri123 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

PERSONAL INFO:

SQL QUERY:

```
SELECT * FROM agent_data NATURAL JOIN agent_login WHERE username
=john_brown;
```

GUI:

LOGINAbout Us

PERSONAL INFOSELL POLICYSOLD POLICY

PERSONAL INFO:--

First name: jhon
Last name: brown
License Number: 568456
Email: jhon@gmail.com
Phone Number: 85768586
Date of birth: 1989-10-23
City: Bengaluru
Street: jpnagar
Zipcode: 560078
Your ID is : 1

SELL POLICY:

SQL QUERY:

```
SELECT agent_id FROM agent_login WHERE username = "jhon_brown";  
INSERT INTO agent_sold_policy (agent_id, policy_id) values(1,123);
```

GUI:

PERSONAL INFOSELL POLICYSOLD POLICY

SELL POLICY:--

- Policy Type: Medical
Policy ID: 123
- Policy Type: Vehical
Policy ID: 124
- Policy Type: Life
Policy ID: 125

Policy_ID *
123

SELL

SOLD POLICY:

SQL QUERY:(Nested query)

```
SELECT ph.fname AS policyholder_name, ph.phone, po.name AS plan_name, pl.type AS
policy_type
FROM (
    SELECT pib.policyholder_id, pib.plan_id, p.policy_id
    FROM policy_issued_by AS pib
    JOIN plan AS p ON pib.plan_id = p.plan_id
    JOIN agent_sold_policy AS asp ON p.policy_id = asp.policy_id
    WHERE asp.agent_id = ?
) AS policy_ids
JOIN policy_holder AS ph ON policy_ids.policyholder_id = ph.policyholder_id
JOIN plan AS po ON policy_ids.plan_id = po.plan_id
JOIN policy AS pl ON policy_ids.policy_id = pl.policy_id;
```

GUI -

•	Policy Holder Name: karl Phone Number: 64557465 Policy Type:Medical Plan Name: Catastrophic Health Insurance
•	Policy Holder Name: Ram Phone Number: 567583 Policy Type:Medical Plan Name: Catastrophic Health Insurance
•	Policy Holder Name: karl Phone Number: 64557465 Policy Type:Medical Plan Name: Children's Health Insurance
•	Policy Holder Name: Akash Phone Number: 468484 Policy Type:Medical Plan Name: Children's Health Insurance
•	Policy Holder Name: karl Phone Number: 64557465 Policy Type:Life

POLICY HOLDER:

Policy holder Register is same as Agent Register.

GUI:

Policy Holder Registration

First Name

Rohan

Last Name

singh

Age

45

Email

rohan@gmail.com

Phone Number

67890456

22-06-1978

City

Hyderabad

State

Telangana

Pincode

5678

Username *

rohan_s

Password *

.....|

Submit

Policy holder Change Password is same as Agent Change Password.

LOGIN About Us

Change Password

Username *

karl_smith

New Password *

CHANGE

POLICY HOLDER LOGIN:

SQL QUERY:

SELECT * FROM policyholder_login WHERE username = karl_smith;

GUI-

LOGIN About Us

Policy Holder Login

Username *

karl_smith

Password *

LOGIN [New user ? Register here](#)[Change Password ?](#)



BUY POLICY:

SQL QUERY:

```
SELECT plan.* FROM policy JOIN plan ON policy.policy_id =
plan.policy_id WHERE policy.type = "Vehicle";
```

```
SELECT plan_id from plan where name="Commercial Auto Insurance";
```

```
INSERT INTO policy_issued_by (plan_id,policyholder_id,start_date,end_date) VALUES
(262,13,23-11-2023,22-11-2025);
```

Function to calculate the end - date:

```
CREATE FUNCTION end_date_plan(start_date DATE)
  RETURNS DATE
  DETERMINISTIC
BEGIN
  DECLARE result_date DATE;
  SET result_date = DATE_ADD(start_date, INTERVAL 2 YEAR);
  RETURN result_date;
END;
```

Internally the queries are like as shown bellow

```
//NESTED QUERY
app.post('/choose_plan', (req, res) => {
  const { policy_type } = req.body;

  if (!policy_type) {
    return res.status(400).json({ error: 'Missing policy_type' });
  }

  const selectQuery = `
    SELECT * FROM plan
    WHERE policy_id IN (
      SELECT policy_id FROM policy
      WHERE type = ?
    )`;

  db.query(selectQuery, [policy_type], (err, result) => {
    if (err) {
      res.status(500).json({ error: 'Database error' });
    } else {
      console.log('JSON Object sent to the client:', result);
      res.status(200).json(result);
    }
  });
});

app.post('/chosenpolicy_plan', (req, res) => {
  const {selectedType,plantype,startDate,endDate,policyholder_id} = req.body;

  const get_plan_id='SELECT plan_id from plan where name=?';
  db.query(get_plan_id, [plantype], (err, result) => {
    if (err) {
      console.error('Error finding data from the database:', err);
      res.status(500).send(err);
    } else {
      console.log('Data found:', result);
      res.status(200).json({ message: ' successfully' });

      planid=result[0].plan_id;
      const plan_issued = 'INSERT INTO policy_issued_by (plan_id,policyholder_id,start_date,end_date) VALUES (?, ?, ?,?)';
      db.query(plan_issued, [planid,policyholder_id,startDate,endDate], (err, Result) => {
        if (err) {
          console.error('Error inserting data:', err);
          res.status(500).send('Error ');
        } else {
          console.log("ok");
        }
      });
    }
  });
});
```

GUI:

PERSONAL INFOBUY POLICYBOUGHT POLICYBENEFICIARYFILE A CLAIMDELETE A BENEFICIARY

•

Policy holder ID: 13

Type: Medical

Description: Medical insurance is a financial arrangement that individuals.The insurer commits to covering a portion of healthcare expenses, such as doctor visits, hospital stays, medications,and preventive care.

Type: Vehical

Description: Vehicle insurance offers financial protection against accidents, theft, and damage to vehicles.

Type: Life

Description: Life insurance offers peace of mind with a tax-free payout to beneficiaries upon the policyholder's death, ensuring financial security and covering expenses and future needs.

PERSONAL INFOBUY POLICYBOUGHT POLICYBENEFICIARYFILE A CLAIMDELETE A BENEFICIARY

Selected Type is: Vehical

Available Plans are:

Plan Name: Liability Insurance

Premium Amount:

Plan Description: It covers bodily injury and property damage liability,ensuring that if you're at fault in an accident,the other party's medical expenses and vehicle repairs are covered.

Plan Name: Commercial Auto Insurance

Premium Amount:

Plan Description: It covers vehicles used for business purposes.It helps businesses maintain financial stability by ensuring that vehicles are safe.

PERSONAL INFOBUY POLICYBOUGHT POLICYBENEFICIARYFILE A CLAIMDELETE A BENEFICIARY

SUMMARY :

Selected Policy is: Vehical

Selected Plan is: Commercial Auto Insurance

Start Date: 2023-11-23

End Date: 2025-11-22

CLOSE

Choose Start Date:

23-11-2023Submit

AFTER THE POLICY IS BOUGHT -

PERSONAL INFO
BUY POLICY
BOUGHT POLICY
BENEFICIARY
FILE A CLAIM
DELETE A BENEFICIARY

- policy: Medical
plan: Children's Health Insurance
start date: 2023-11-21T18:30:00.000Z
end date: 2025-11-20
- policy: Vehical
plan: Commercial Auto Insurance
start date: 2023-11-22T18:30:00.000Z
end date: 2025-11-21

BOUGHT POLICY:

SQL QUERY:

```
SELECT pib.*, p.type AS policy_type, pl.name AS plan_name FROM policy_issued_by AS
pib JOIN plan AS pl ON pib.plan_id = pl.plan_id JOIN policy AS p ON pl.policy_id =
p.policy_id WHERE pib.policyholder_id = 1;
```

GUI-

PERSONAL INFO
BUY POLICY
BOUGHT POLICY
BENEFICIARY
FILE A CLAIM
DELETE A BENEFICIARY

- policy: Medical
plan: Catastrophic Health Insurance
start date: 2023-11-22T18:30:00.000Z
end date: 2025-06-18
- policy: Medical
plan: Children's Health Insurance
start date: 2023-11-10T18:30:00.000Z
end date: 2025-11-09
- policy: Vehical
plan: Commercial Auto Insurance
start date: 2023-11-08T18:30:00.000Z
end date: 2025-05-13
- policy: Vehical
plan: Gap Insurance
start date: 2023-11-23T18:30:00.000Z
end date: 2026-10-12
- policy: Life
plan: Whole Life Insurance
start date: 2023-11-15T18:30:00.000Z
end date: 2025-07-07
- policy: Life
plan: Joint Life Insurance
start date: 2023-11-20T18:30:00.000Z
end date: 1969-12-31

BENEFICIARY:

SQL QUERY:

```
INSERT INTO BENEFICIARY (policyholder_id, fname, lname, age, email,
phone, dob, city, state, pin) VALUES
(Rahul,rao,45,rahul@gmail.com,45623987,17-10-
1978,Hyderabad,Telangana,5678);
```

GUI:

Beneficiary Registration

First Name

Rahul

Last Name

rao

Age

45

Email

rahul@gaim.com

Phone Number

45623987

17-10-1978

City

Hyderabad

State

Telangana

Pincode

5678

SUBMIT

```
mysql> select * from beneficiary;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| beneficiary_id | policyholder_id | fname | lname | age | email | phone | dob | city | state | pin |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | 2 | Akash | sharma | 35 | akash@gmail.com | 468484 | 2023-11-09 | Bengaluru | NULL | 560078 |
| 3 | 2 | abcd | abcd | 56 | abc@gmail.com | 35678798 | 2023-11-09 | Bengaluru | NULL | 560078 |
| 4 | 2 | pri | sha | 76 | pri@gmail.com | 8758375 | 2023-11-09 | Bengaluru | NULL | 560078 |
| 5 | 2 | pri | Nile | 45 | pri@gmail.com | 8758 | 2023-11-23 | Bengaluru | NULL | 560078 |
| 7 | 2 | Madhvi | Sharma | 34 | madhvi@gmail.com | 7568785 | 1989-06-22 | Bengaluru | NULL | 560078 |
| 8 | 13 | shreya | pal | 34 | pal@gmail.com | 45567 | 1989-06-15 | Bengaluru | NULL | 560078 |
| 9 | 1 | Rahul | rao | 45 | rahul@gaim.com | 45623987 | 1978-10-17 | Hyderabad | NULL | 5678 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

WHEN THE POLICY HOLDER HAS 5 BENEFICIARY ALREADY
(this is for policy holder with the id 2)

SQL QUERY:

```
select count(beneficiary_id) from BENEFICIARY where  
policyholder_id=2;
```

GUI :

❗ Error ×

The limit for your Beneficiary is exceeded

Beneficiary Registration

First Name

liya

Last Name

sharma

Age

34

Email

liya@gmail.com

Phone Number

98763546

15-06-1989

📅

City

Hyderbad

DELETE BENEFICIARY:

SQL QUERY:

```
DELETE FROM beneficiary WHERE fname = "abcd" AND lname = "abcd" AND  
phone=35678798 AND policyholder_id = 2;
```

GUI-

PERSONAL INFO BUY POLICY BOUGHT POLICY BENEFICIARY FILE A CLAIM DELETE A BENEFICIARY

First Name
abcd

Last Name
abcd

Phone Number
35678798

DELETE BENEFICIARY

McAfee Security Dell Gmail YouTube localhost:3000 says
Beneficiary deleted successfully OK LOGIN About Us

PERSONAL INFO BUY POLICY BOUGHT POLICY BENEFICIARY FILE A CLAIM DELETE A BENEFICIARY

First Name
abcd

Last Name
abcd

Phone Number
35678798

DELETE BENEFICIARY

FILE A CLAIM:

SQL QUERY:

```
SELECT policyholder_id from policyholder_login where username="akash_sharma";
select coverage_id from coverage where (plan_id in (SELECT plan_id from
policy_issued_by where policyholder_id=13)) and (coverage_name="Property Damage
Liability") and (coverage_amount>="4250")
```

```
INSERT into claim (claim_date, status, reason, policyholder_id) values("22-11-
2023","Accepted","car accident",13);
```

```
INSERT INTO claims_coverage (claim_id, coverage_id, claimed_amount) VALUES (21,118,
4250);
```

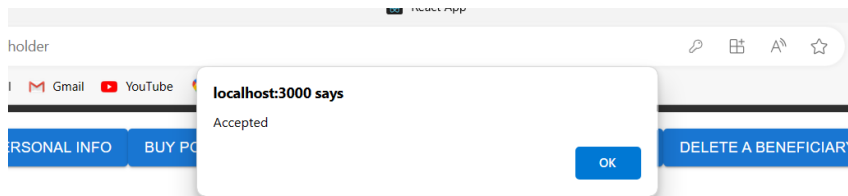
Internally Queries work like this:

```
JS server1.js X JS Buypolicy.js U JS Beneficiary.js U
backend > JS server1.js > app.post('/make_claim') callback
727 app.post('/make_claim', (req, res) => {
728   const { reason, date, claimed_amount, username, category } = req.body;
729
730   const findQuery = 'SELECT policyholder_id from policyholder_login where username=?';
731   db.query(findQuery, [username], (err, result) => {
732     if (err) {
733       console.error('Error querying the database:', err);
734       //res.status(500).json({ error: 'Database error' });
735     } else {
736       console.log('Data queried from the database:', result);
737
738       if (result.length === 0) {
739         //res.status(404).json({ error: 'User not found' });
740       } else {
741         const policyholderId = result[0].policyholder_id;
742         const selectSQL = 'SELECT plan_id from policy_issued_by where policyholder_id=?';
743         db.query(selectSQL, [policyholderId], (err, result1) => {
744           if (err) {
745             console.error('Error querying policy data:', err);
746             //res.status(500).json({ error: 'Database error' });
747           } else {
748             const planID = result1.map(item => item.plan_id);
749             console.log(planID)
750             const coveragesql = 'select coverage_id from coverage where (plan_id in (?) and (coverage_name=?) and (coverage_amount>=?));'
751             db.query(coveragesql, [planID, category, claimed_amount], (err, result2) => {
752               if (err) {
753                 console.error('Error querying the database:', err);
754                 //res.status(500).json({ error: 'Database error' });
755               } else {
756                 const coverageID = result2.map(item => item.coverage_id);
757                 if (result2.length === 0) {
758                   const status = 'Rejected';
759                   insertClaimAndCoverage(status, policyholderId, date, reason, claimed_amount, coverageID);
760                 } else {
761                   const status = 'Accepted';
762                   insertClaimAndCoverage(status, policyholderId, date, reason, claimed_amount, coverageID);
763                 }
764               }
765             });
766           }
767         });
768       }
769     }
770   });
771 }
```

```
JS server1.js X JS Buypolicy.js U JS Beneficiary.js U
backend > JS server1.js > app.post('/make_claim') callback
761   const status = 'Accepted';
762   insertClaimAndCoverage(status, policyholderId, date, reason, claimed_amount, coverageID);
763 }
764 }
765 });
766 }
767 });
768 }
769 }
770 });
771
772 function insertClaimAndCoverage(status, policyholderId, date, reason, claimed_amount, coverageID) {
773   const insertClaim = 'INSERT into claim (claim_date, status, reason, policyholder_id) values(?,?,?,?)';
774   db.query(insertClaim, [date, status, reason, policyholderId], (err, result3) => {
775     if (err) {
776       console.error('Error querying the database:', err);
777       //res.status(500).json({ error: 'Database error' });
778     } else {
779       console.log('Data inserted from the database:', result3);
780
781       const claimID = result3.insertId;
782
783       const claimrelation = 'INSERT INTO claims_coverage (claim_id, coverage_id, claimed_amount) VALUES (?, ?, ?)';
784       db.query(claimrelation, [claimID, coverageID, claimed_amount], (err, result4) => {
785         if (err) {
786           console.error('Error querying the database:', err);
787           // res.status(500).json({ error: 'Database error' });
788         } else {
789           console.log('Data inserted from the database:', result4);
790         }
791       });
792     }
793   });
794
795   // Send the response outside the nested callbacks
796   res.status(200).json({ success: status });
797 }
```

GUI:

Valid claim-



Make a Claim

Claim Category
Property Damage Liability

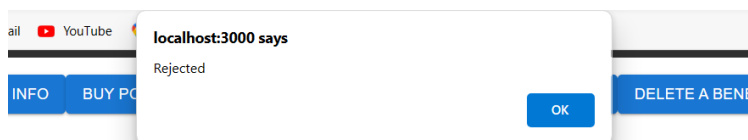
Claim Reason
car accident

Claim Amount
4250

Date
22-11-2023

SUBMIT CLAIM

Invalid claim-



Make a Claim

Claim Category
Property Damage Liability

Claim Reason
car accident

Claim Amount
6000

Date
22-11-2023

SUBMIT CLAIM

