**Project team 9 - Insurance Company Database**

*Team Members:*

*AKSHAY AYYANCHIRA  
RAGINI RANGANATHAN  
KAVYA POTLURI  
PRATIK PAREKH*

## 

## ***PROJECT PROPOSAL***

**Objective**

The objective is to create insurance management software for the insurance company which provides different types of insurance ranging from Life Insurance, Property, Gadgets, Vehicle insurance. This system will organize and track insurance and different policies bought by customers. Thus,minimizing the workload of the Agent and the company manager.

**Content**

* Users can pick insurance of their need and will be able to store their information and payment info for automatic monthly payments.
* System will keep track of medical/property history.
* Storing profit/commission data - The software will keep track of how the company is making profit. Along with it, if user signs up through any referral or agent, software will keep track of their profit margins/earnings too.
* Basic agent information like name , contact details will be stored in the database.
* There will be option to create new policy and edit existing policies.

**Scope**

* Policy related information of the particular customer can be easily searched in the database .
* The system will calculate the insurance premium for a particular customer based on insurance package they choose.
* This system will automatically inform the agent regarding customer premium due date so that agent will not forget to collect premium from their customers .
* The system will also generate a monthly analysis report for the insurance agent so that they can monitor their monthly business.

***PROJECT ENVIRONMENT***

* The UI of the software will be developed in HTML and styled using CSS.
* For database connection, php version 7 with Jetbrains PHPstorm IDE will be used.
* AWS MYSQL RDS instance with MySQL workbench for database development.

## ***HIGH LEVEL REQUIREMENTS***

## **Initial user roles**

|  |  |
| --- | --- |
| **User Role** | **Description** |
| Guest | Any user who can browse through insurance policies but hasn’t signed up with the system yet. |
| Customer | A guest who signed up with the system and has an account and is intended to buy policies. |
| Agent | User who was employed with the insurance company for selling policies. He can look for customers data enrolled by him and remind them of their expiring policy. |
| Manager | User having administrator role. Primary tasks will be to add new rules and policy, enrolling agents in system etc. |

### **Initial user story descriptions**

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| US1 | As a User\*, I can browse through multiple policies offered by the company. |
| US2 | As a Customer , I should be able to buy multiple policies. |
| US3 | As a Guest, I want to sign up with the system, so that I can buy policies. |
| US4 | As an Agent, I want to enroll customer in the policy. |
| US5 | As a Customer, I should be able to login to the system. |
| US6 | As a Customer, I want to file an insurance. |
| US7 | As a Customer, I want to check the status of claim. |
| US8 | As a Manager, I want to change the status of claim, so that I can either approve or keep it on hold. |
| US9 | As a Customer, I want to view the policies that I am currently enrolled and edit my profile. |
| US10 | As a Manager, I want to change the rules of existing policy so that I can address some existing problem if any. |
| US11 | As a Manager, I want to notify change of policy to policy holders, so that they are informed about the changes. |
| US12 | As an Agent, I want to see details about customers enrolled by me, so that I can keep track of how they are doing. |
| US13 | As a Customer, I want pay premium for my policies. |
| US14 | As a Manager , I want to monitor the salary payments to the agent . |
| US15 | As an Agent, I want to notify customers about policy expiration. |
| US16 | As an Agent, I want to check my commission details. |
| US17 | As a Customer, I want to delete my account from the account. |
| US23 | As a User\*, I should be able to perform advance search by providing combination of keywords and search for the policies offered by the company. |

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| US1 | ~~As a User\*, I can browse through multiple policies offered by the company.~~ |
| US2 | ~~As a Customer , I should be able to buy multiple policies.~~ |
| US3 | ~~As a Guest, I want to sign up with the system, so that I can buy policies.~~ |
| US4 | As an Agent, I want to enroll customer in the policy. |
| US5 | ~~As a Customer, I should be able to login to the system.~~ |
| US6 | ~~As a Customer, I want to file an insurance.~~ |
| US7 | ~~As a Customer, I want to check the status of claim.~~ |
| US8 | ~~As a Manager, I want to change the status of claim, so that I can either approve or keep it on hold.~~ |
| US9 | As a Customer, I want to view the policies that I am currently enrolled and edit my profile. |
| US10 | ~~As a Manager, I want to change the rules of existing policy so that I can address some existing problem if any.~~ |
| US11 | ~~As a Manager, I want to notify change of policy to policy holders, so that they are informed about the changes.~~ |
| US12 | As an Agent, I want to see details about customers enrolled by me, so that I can keep track of how they are doing. |
| US13 | As a Customer, I want pay premium for my policies. |
| US14 | As a Manager , I want to monitor the salary payments to the agent . |
| US15 | As an Agent, I want to notify customers about policy expiration. |
| US16 | As an Agent, I want to check my commission details. |
| US17 | ~~As a Customer, I want to delete my account from the account.~~ |
| US23 | ~~As a User\*, I should be able to perform advance search by providing combination of keywords and search for the policies offered by the company.~~ |
| US24 | Have a trigger which checks if transaction date passed the due date? If so, add late charges to the payable amount |
| US25 | As a manager, I want to see commission earned by particular Agent by providing his employee id. (Stored Procedure) |
| NOTE | Daily Event to calculate the total amount of premium based on duedate, premium, penalty & total and increase the due date by 30 if due date passes. |
| NOTE | Commission: A trigger which will be invoked when CustomerPolicy table has new entry.  This trigger will see if the customer has an agent and will call stored procedure passing the agents employee id to insert a new entry in commission table |
| NOTE | A view for displaying transactions done by all customer(Manager’s View) Customer name, policy id, and wahtever we can add. |
|  | A view for customers to see all their previous transactions (Stored procedure where we will have a text box to get customer’s username) ---- Same thing for claim. |

\*The rolename **User** is used in User Stories that apply to both Customer, guest.

***HIGH LEVEL CONCEPTUAL DESIGN***

Assumption : Guest users of the application are not stored in database,

so we have no entity for guest.

Entities:

Policy

Agent

Manager

Customer

Claim

Transaction

Account

Commission

Relationships:

**Customer** buys **Policy**

**Customer** makes **Transaction** by paying Premium

**Customer** logs into his **Account**

**Agent** logs into his **Account**

**Manager** logs into his **Account**

**Customer** files a **Claim**

**Agent** enrolls **Customer**

**Manager** approves **claim**

**Manager** holds **claim**

**Manager** adds **Policy**

**Agent** checks **Commission**

**Agent** notifies **Customer** about policy expiration

**Sprint 1**

**Refining Requirements:**

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| US3 | As a Guest, I want to sign up with the system by entering my a unique username, first name, last name, birth date, email, and password. |
| US1 | As a User\*, I can browse through multiple policies offered by the company. |
| US19 | As an Agent, I want to add an registered customer in my customer’s list. |
| US4 | As an agent, I want to enroll in my customer’s list a new customer into the system with a unique username, first name, last name, email id, birth date and password. |
| US20 | As a Customer, I can view or edit my profile. |
| US21 | As a Manager, I can view all the agents working for me. |
| US12 | As an Agent, I can view all my customers and number of policies they hold. |
| US22 | As a Manager, I can view all the customers assigned to my agents and the number of policies they hold. |

\*The rolename **User** is used in User Stories that apply to both Customer, guest.

## ***Conceptual design:***

Entity :Customer

Attributes:

Username

First\_name

Last\_name

Date\_of\_birth[composite attribute]

Email\_id

Phone\_no[multi-valued attribute]

Address[composite]

Address\_line1

Address\_line2

City

State

Zip\_code

Entity :Agent

Attributes:

Username

First\_name

Last\_name

Email\_id

Phone\_no [multi-valued attribute]

Address [composite]

Address\_line1

Address\_line2

City

State

Zip\_code

Entity : Policy

Attributes:

Policy\_id

Policy\_Name

Description

Premium

Coverage

Years

Policy\_type

Entity :Manager

Attributes:

Username

First\_name

Last\_name

Email\_id

Phone\_no [multi-valued attribute]

Address [composite]

Address\_line1

Address\_line2

City

State

Zip\_code

Entity :Account

Attributes:

Username

Password

Account\_type

**Relationships:**

Relationship : **Customer** has **Policy**

Cardinality : Many to Many

Participation:

Customer has partial participation

Policy has partial participation

Relationship : **Agent** has **Customer**

Cardinality: One to Many

Participation:

Agent has partial participation

Customer has partial participation

Relationship : **Agent** works for **Manager**

Cardinality: Many to One

Participation:

Agent has total participation

Manager has partial participation

Relationship : **Customer** has **Account**

Cardinality: One to One

Participation:

Agent has total participation

Manager has total participation

Relationship : **Manager** has **Account**

Cardinality: One to One

Participation:

Agent has total participation

Manager has total participation

Relationship : **Agent** has **Account**

Cardinality : One to One

Participation :

Agent has Total participation

Account has partial participation

**Part 3: Logical Design**

Table:Account

Columns:

Username

Password

Account\_type

Table:Customer

Columns:

Username [foreign key; references Username of Account]

First\_name

Last\_name

Date\_of\_birth

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Justification:

Username is considered as primary key in customer table as each customer who signs up with the system chooses a unique username.

Also the multivalued attribute ‘Phone\_no’ is restricted to two values i.e ‘Phone\_no1’ and ‘Phone\_no2’ and two separate columns are made for that.

Table:Policy

Columns:

Policy\_id

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

Justification:

Each policy holds a unique policy\_id, so it is considered as the primary key.

Table:Agent

Columns:

Username [foreign key; references Username of Account]

Manager\_username [foreign key; reference to Username of Manager]

First\_name

Last\_name

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Justification:

Each Username will be unique so we chose to have Username as primary key of Agent.

As Agent works for Manager doesn’t has any special relationship attributes, we chose to include manager\_username as a foreign key to establish the relationship between Agent and Manager.

Table :Manager

Columns:

Username [foreign key; references Username of Account]

First\_name

Last\_name

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Justification:

Each Username will be unique so we chose to have Username as primary key of Manager.

As Agent works for Manager doesn’t has any special relationship attributes, we chose to include the primary key of Agent as a foreign key in Manager to establish the relationship between Agent and Manager.

Table:CustomerAgent

Columns:

Customer\_username [foreign key;references username of Customer]

Agent\_username [foreign key;references username of Agent]

Justification:

Though the relationship is not many to many, we choose to have cross-reference table for Customer has an Agent relationship because of their participation constraint. A customer may or may not have an Agent and this may result in having null values in Customer Table.

Table:CustomerPolicy

Columns:

Customer\_username [foreign key;references Username of Customer]

Policy\_id [foreign;references Policy\_id of Policy]

Purchase\_date

Start\_date

End\_date

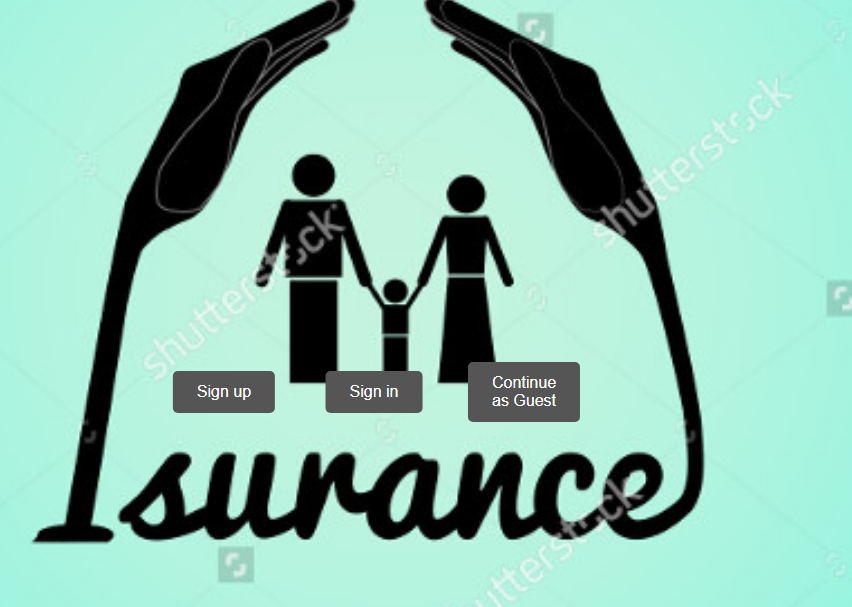
Justification:

The relation between customer and policy is many to many. Therefore having a cross-reference table is the only option.

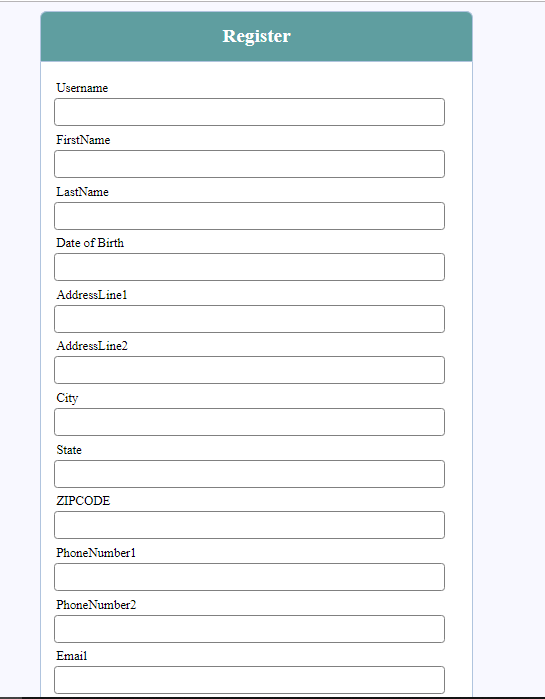
### **Part 5: Demonstrate key SQL queries**

FRONTEND SCREENSHOTS

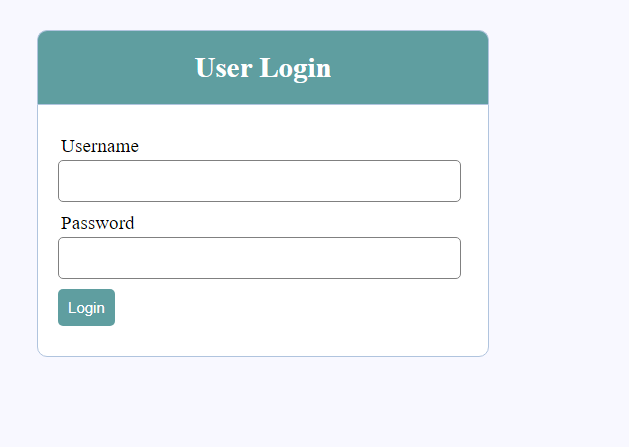
Homepage



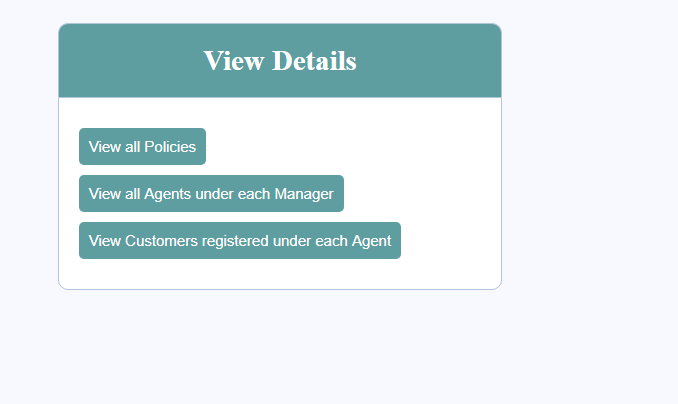
User Registration



User Login



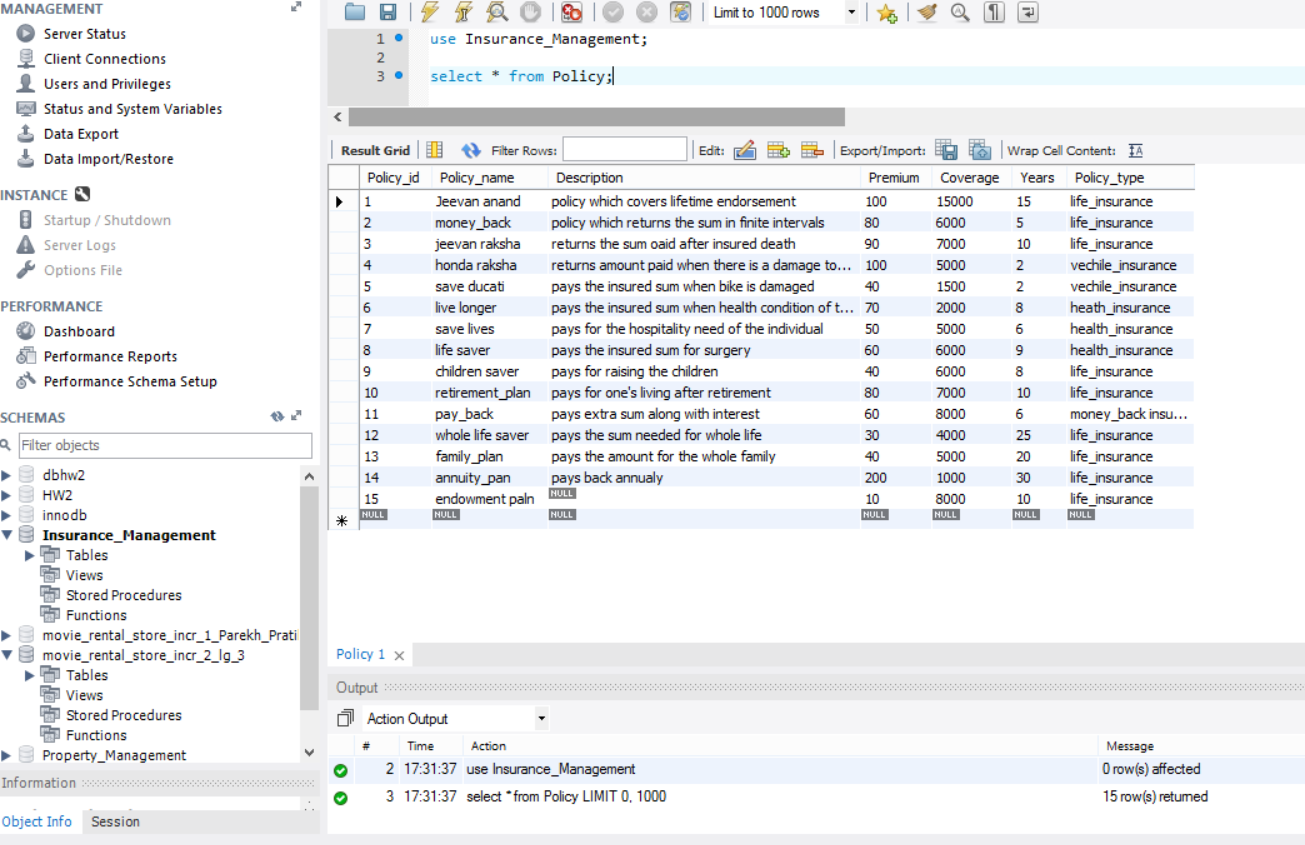
Buttons to display data in UI:



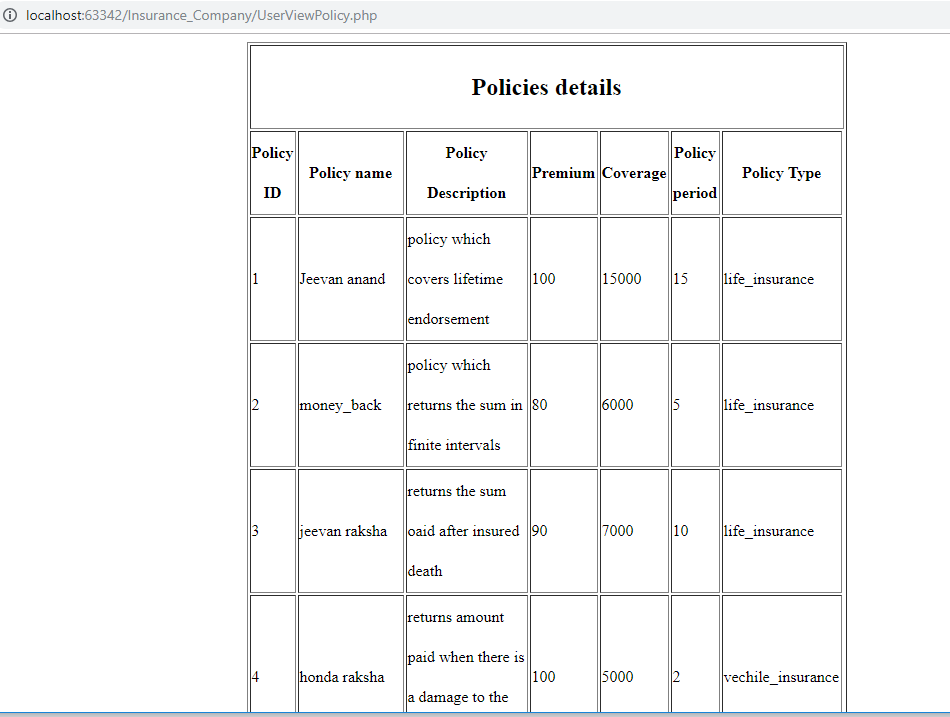
Data for all Policies in the database:

Query :

select \* from Policy;



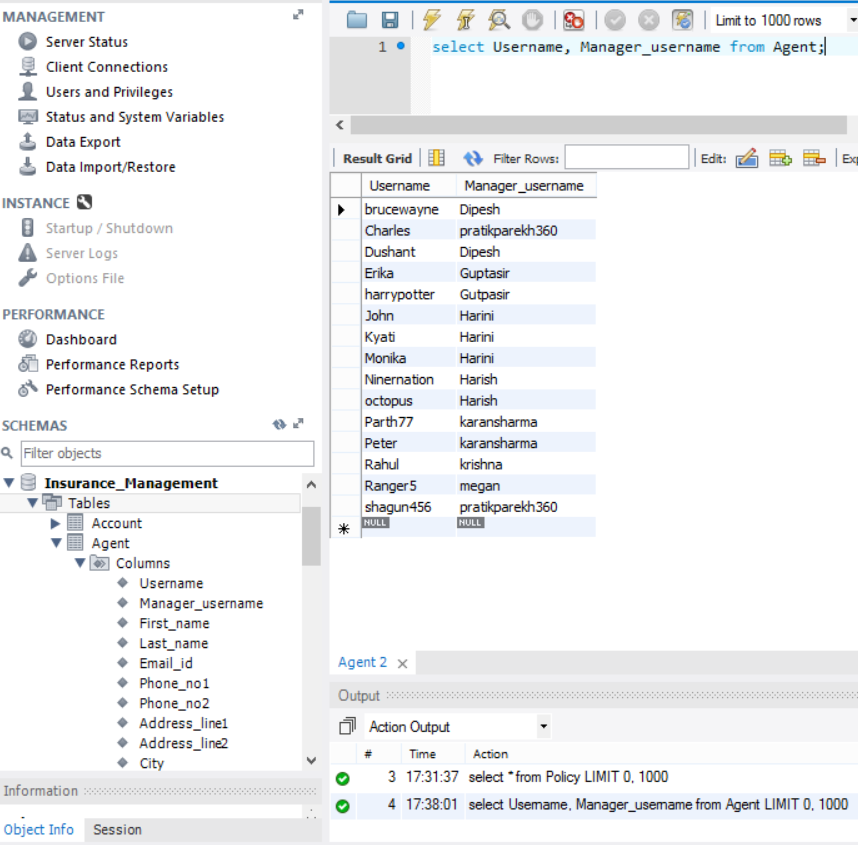
Corresponding UI display :



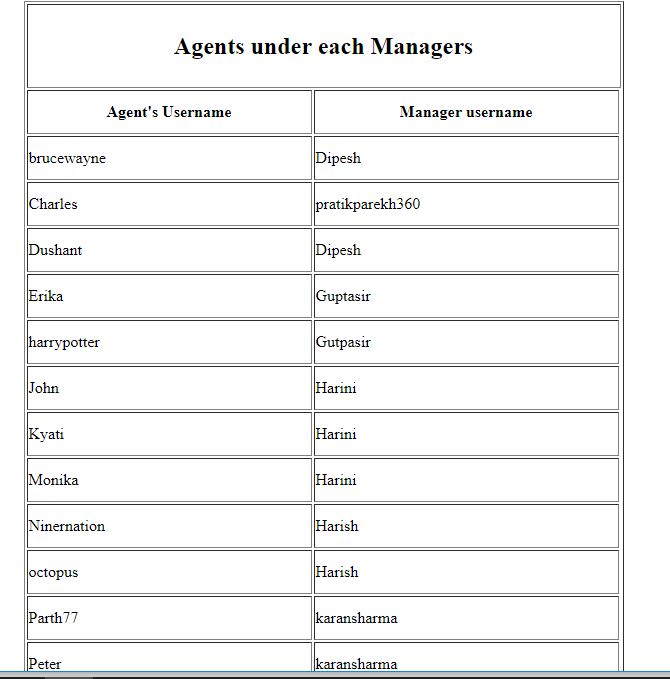
Data for agent username and its corresponding manager’s username :

Query :

select Username, Manager\_username from Agent;



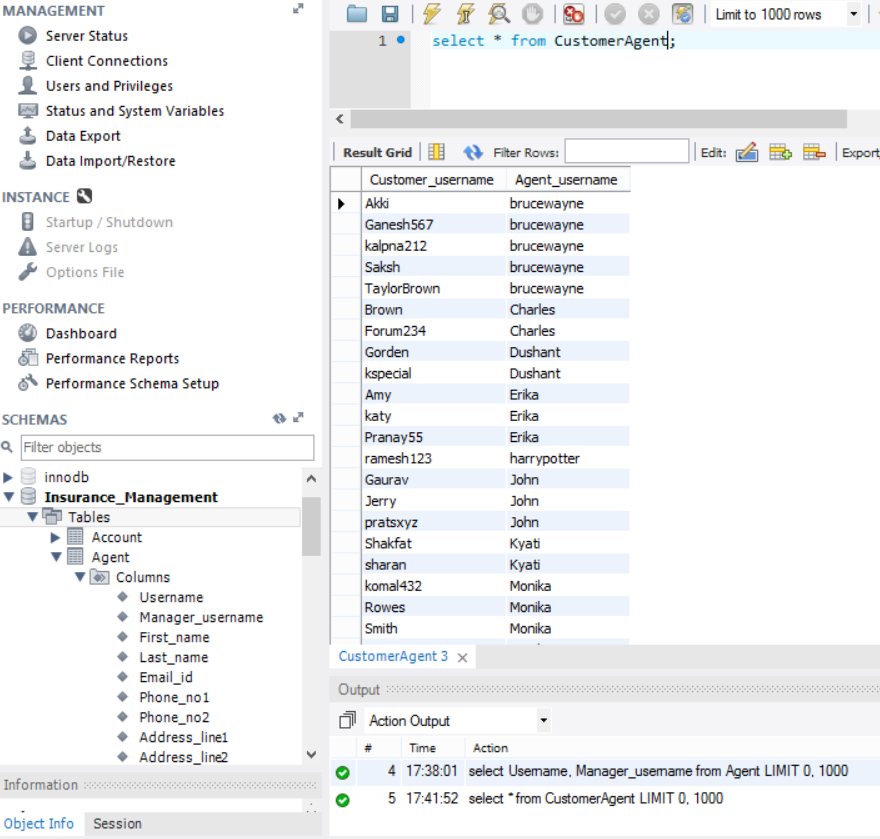
Corresponding UI display:



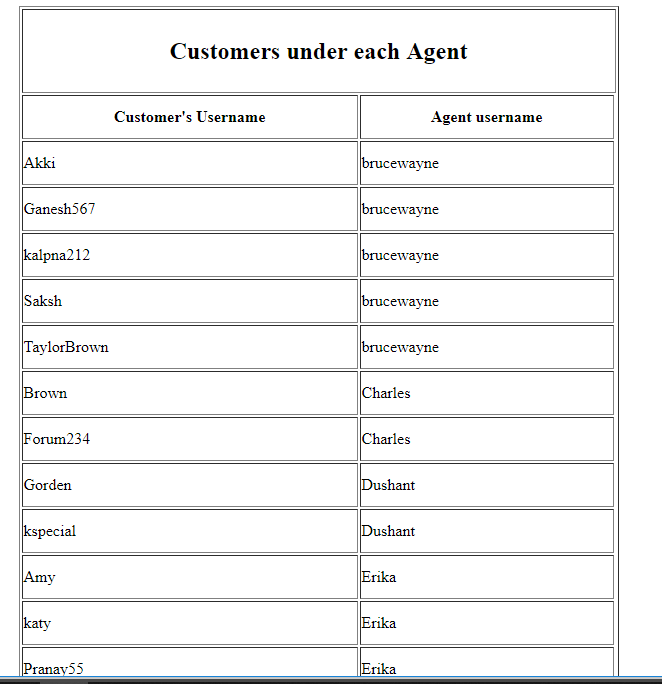
Data for listing customer and its corresponding agent:

Query:

Select \* from CustomerAgent;



Corresponding UI display:



Sprint 2

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| US1 | As a Customer , I should be able to buy multiple policies. |
| US2 | As a Customer, I want to file an insurance. |
| US3 | As a Customer, I want to check the status of claim. |
| US4 | As a Manager, I want to change the status of claim, so that I can either approve or keep it on hold. |
| US5 | As a Customer, I want to view the policies that I am currently enrolled and edit my profile. |
| US6 | As a Customer, I want pay premium for my policies. |

***CONCEPTUAL DESIGN:***

**Entity : User\_personal\_details**

Attributes:

Username

First\_name

Last\_name

Date\_of\_birth [composite attribute]

Email\_id

Phone\_no [multi-valued attribute]

Address [composite]

Address\_line1

Address\_line2

City

State

Zip\_code

**Entity :Customer**

Attributes:

Username

Credit\_card

Expiry\_date

Name\_on\_card

**Entity: Employee**

**Attributes**

Employee\_id

Username

Designation

Manager\_id

**Entity : Policy**

Attributes:

Policy\_id (It will uniquely identify a policy)

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

**Entity :Account**

Attributes:

Username

Password

Account\_type

**Entity :Transaction**

Attributes:

Transaction\_id (It will uniquely identify a transaction)

Policy\_Number

Amount

Description

Date

**Entity :Claims**

Attributes:

Claim\_id (It will uniquely identify a claim)

Policy\_Number

Status

Description

Amount

Approved\_by

**Relationships:**

Relationship : **Customer** buys a **Policy**

Cardinality : Many to Many

Participation:

Customer has partial participation

Policy has partial participation

Relationship : **Employee(Agent)** serves **Customer**

Cardinality: One to Many

Participation:

Employee(Agent) has partial participation

Customer has partial participation

Relationship : **Employee(Agent)** works for **Employee(Manager)**

Cardinality: Many to One

Participation:

Employee (Agent) has total participation

Employee (Manager) has partial participation

Relationship : **Customer** has an **Account**

Cardinality: One to One

Participation:

Customer has total participation

Account has partial participation

Relationship : **Employee** has an **Account**

Cardinality: One to One

Participation:

Employee has total participation

Account has partial participation

Relationship: **Customer** files a **Claim**

Cardinality: One to many

Participation:

Customer has partial participation

Claim has total participation

Relationship: **Customer** makes **Transaction** by paying premium

Cardinality: One to many

Participation:

Customer has partial participation

Transaction has total participation

Relationship: **Employee(Manager)** changes status of **Claim**

Cardinality: One to many

Participation:

Manager has partial participation

Claim has total participation

**Sprint 2 - Logical Design**

Table:Account

Columns:

Username

Password

Account\_type

Justification : Username is unique for customer and employee, so its is chosen as primary key in account table.

Table:User\_personal\_details

Columns:

Username [foreign key; references Username of Account]

First\_name

Last\_name

Date\_of\_birth

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Justification:

Username is considered as primary key in User\_personal\_details as each customer and employee who signs up with the system chooses a unique username.

Also the multivalued attribute ‘Phone\_no’ is restricted to two values i.e ‘Phone\_no1’ and ‘Phone\_no2’ and two separate columns are made for that.

Table:Customer

Columns:

Username [ foreign key ; references Username of User\_personal\_details ]

Credit\_card\_number

Expiry\_date

Name\_on\_card

Justification: Username is the foreign key to the User\_personal\_details table which determines the customer’s personal details

Table:Policy

Columns:

Policy\_id

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

Justification: Each policy holds a unique policy\_id, so it is considered as the primary key.

Table:Employee

Columns:

Employee\_id

Designation

Username [Foreign key; references Username of Account]

Manager\_id [Foreign key; reference to Employee\_id of Employee]

Justification:

Username is chosen as foreign key to associate each employee with a Account.

As Agent works for Manager doesn’t has any special relationship attributes, we chose to include manager\_username as a foreign key to establish the relationship between Agent and Manager.

Table:CustomerAgent

Columns:

Customer\_username [Foreign key;references username of Customer]

Employee\_id(Agent) [Foreign key;references Employee\_id of Employee]

Justification:

Though the relationship is not many to many, we choose to have cross-reference table for Customer has an Agent relationship because of their participation constraint. A customer may or may not have an Agent and this may result in having null values in Customer Table.

Table: CustomerPolicy

Columns:

Policy\_Number

Customer\_username [Foreign key;references Username of Customer]

Policy\_id [Foreign key;references Policy\_id of Policy]

Purchase\_date

Start\_date

End\_date

Justification:

The relation between customer and policy is many to many. Therefore having a cross-reference table is the only option with Policy\_Number as a primary key..Customer\_username is used as foreign key to associate with Customer table.Policy\_id is used as foreign key to associate with policy table.

Table:**Claim**

Columns :

Claim\_id

Status

Policy\_Number [Foreign key reference to Policy\_Number of CustomerPolicy table]

Description

Amount

Approved\_by [Foreign key reference to Employee\_id of Employee table]

Justification: The Policy\_Number links the claim\_id with corresponding policy\_id in the customerpolicy table. Therefore Policy\_Number is set as foreign key to the customerPolicy table.Approved\_by will tell which Employee(Manager) is taking care of the claim.

Table:**Transaction**

Columns:

Transaction\_id

Policy\_Number [Foreign key reference to CustomerPolicy table]

Amount

Description

Date

Justification:

A policy bought by customer will have a policy number. And there will be multiple premium installments for each insurance. To uniquely identify, we need to have a transaction ID which will be a primary key in this table.

policy number, a foreign key to customer policy table will signify for which user’s policy is the premium being paid.

**Sprint 2 - Normalization Logical Design**

Table:Account

Columns:

Username

Password

Account\_type

Normalization Justification : Account is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table:User\_details

Columns:

Username [foreign key; references Username of Account]

First\_name

Last\_name

Date\_of\_birth

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Normalization Justification : 2NF(Same zip\_code can be used by multiple cities. Hence, creating a seperate table for zip codes, cities and states having primary key as zip\_code is not possible. In such case we will need to create a composite key having zip\_code and city and add it as foreign key to User\_table table which is not feasible. Also, zip\_code, city and state are used only by User\_detail table, hence there is no point in creating a seperate table which will not be used by any other table.Creating a seperate table increases unnecessarily joins which slower performance. Hence, we avoided creating seperate table unless its absolute necessary.

Table:Customer

Columns:

Username [ foreign key ; references Username of User\_details ]

Credit\_card\_number

Expiry\_date

Name\_on\_card

Normalization justification: The table is in 2NF. The customer is assumed to have only one credit card, and credit card information is used only by the customer table so there is no point in creating a separate table which will result in using unnecessary joins.

Table:Policy

Columns:

Policy\_id

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

Normalization Justification :Policy is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:Employee

Columns:

Employee\_id

Designation

Username [Foreign key; references Username of Account]

Manager\_id [Foreign key; reference to Employee\_id of Employee]

Normalization Justification:

The table is in 3NF. The table is not in BCNF because the non prime attribute ‘Username’ can determine the primary attribute - Employee\_id.

The Employee\_id attribute is introduced and made primary key for easy reference and linking to other table. Further normalization of this table is not beneficial and will result in over normalization.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table:CustomerAgent

Columns:

Customer\_username [Foreign key;references username of Customer]

Employee\_username(Agent) [Foreign key;references username of Employee]

Normalization Justification: 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table:CustomerPolicy

Columns:

Policy\_number

Customer\_username [Foreign key;references Username of Customer]

Policy\_id [Foreign key;references Policy\_id of Policy]

Justification:

The relation between customer and policy is many to many. Therefore having a cross-reference table is the only option with Policy\_Number as a primary key..Customer\_username is used as foreign key to associate with Customer table.Policy\_id is used as foreign key to associate with policy table.

Normalization Justification:4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table:CustomerPolicyDetails

Columns:

Policy\_number

Purchase\_date

Start\_date

End\_date

Justification: The table is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Because the Policy\_id and username could uniquely identify a policy details in previous approach, the table was not satisfying the 3NF requirements. Hence we created this table to achieve 4NF.

Table:**Claim**

Columns :

Claim\_id

Status

Policy\_Number [Foreign key reference to Policy\_Number of CustomerPolicy table]

Description

Amount

Approved\_by [Foreign key reference to Employee\_id of Employee table]

Justification:

The Policy\_Number links the claim\_id with corresponding policy\_id in the customer\_policy table. Therefore Policy\_Number is set as foreign key to the customer\_policy table.

Approved\_by will tell which Employee(Manager) is taking care of the claim.

Normalization Justification :4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:**Transaction**

Columns:

Transaction\_id

Policy\_Number [Foreign key reference to CustomerPolicy table]

Amount

Description

Date

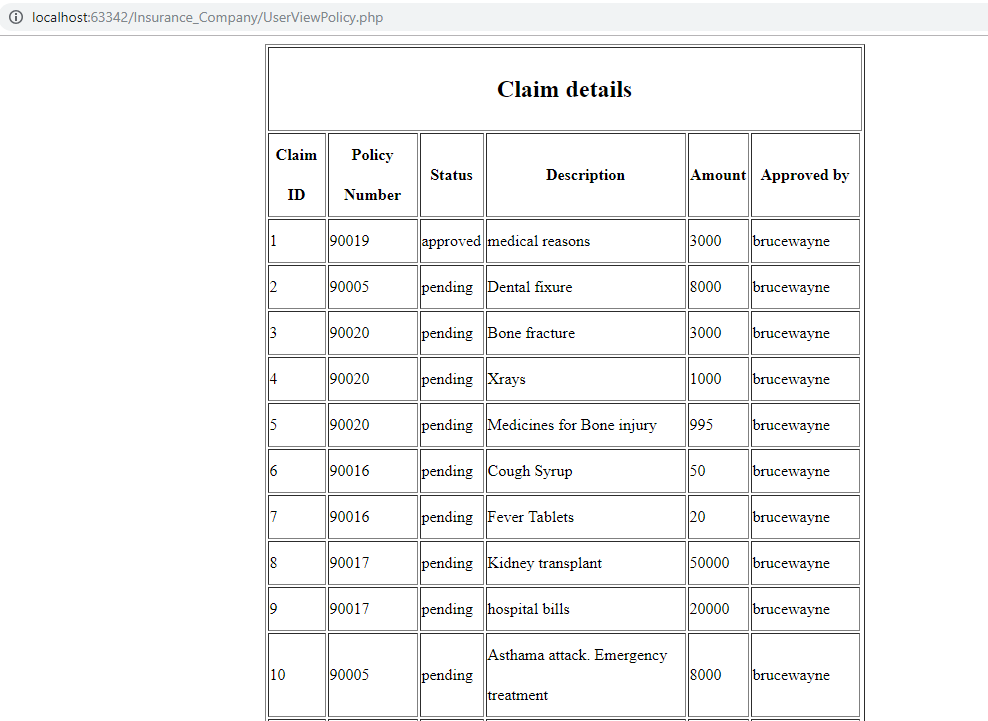
Justification:

A policy bought by customer will have a policy number. And there will be multiple premium installments for each insurance. To uniquely identify, we need to have a transaction ID which will be a primary key in this table.

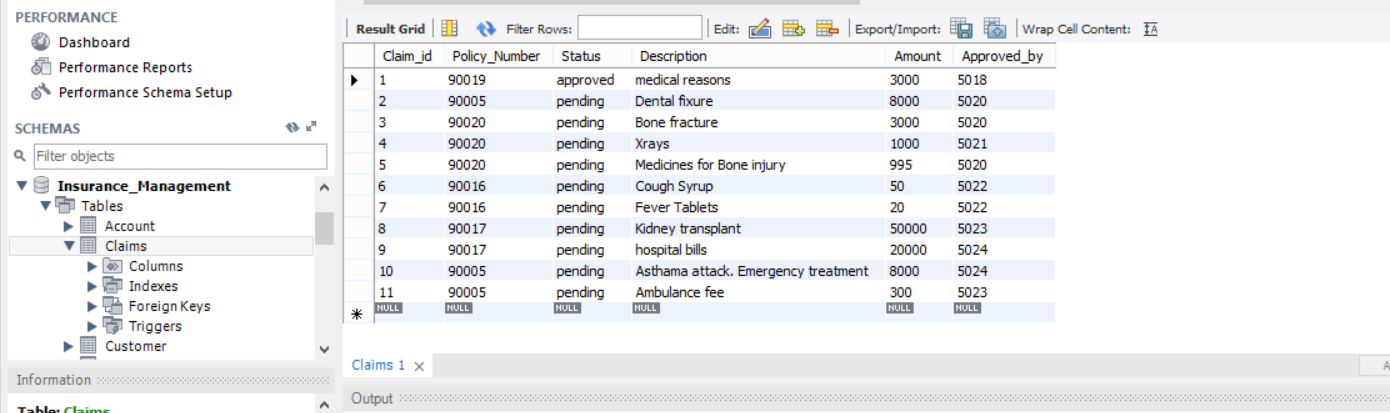
policy number, a foreign key to customer policy table will signify for which user’s policy is the premium being paid.

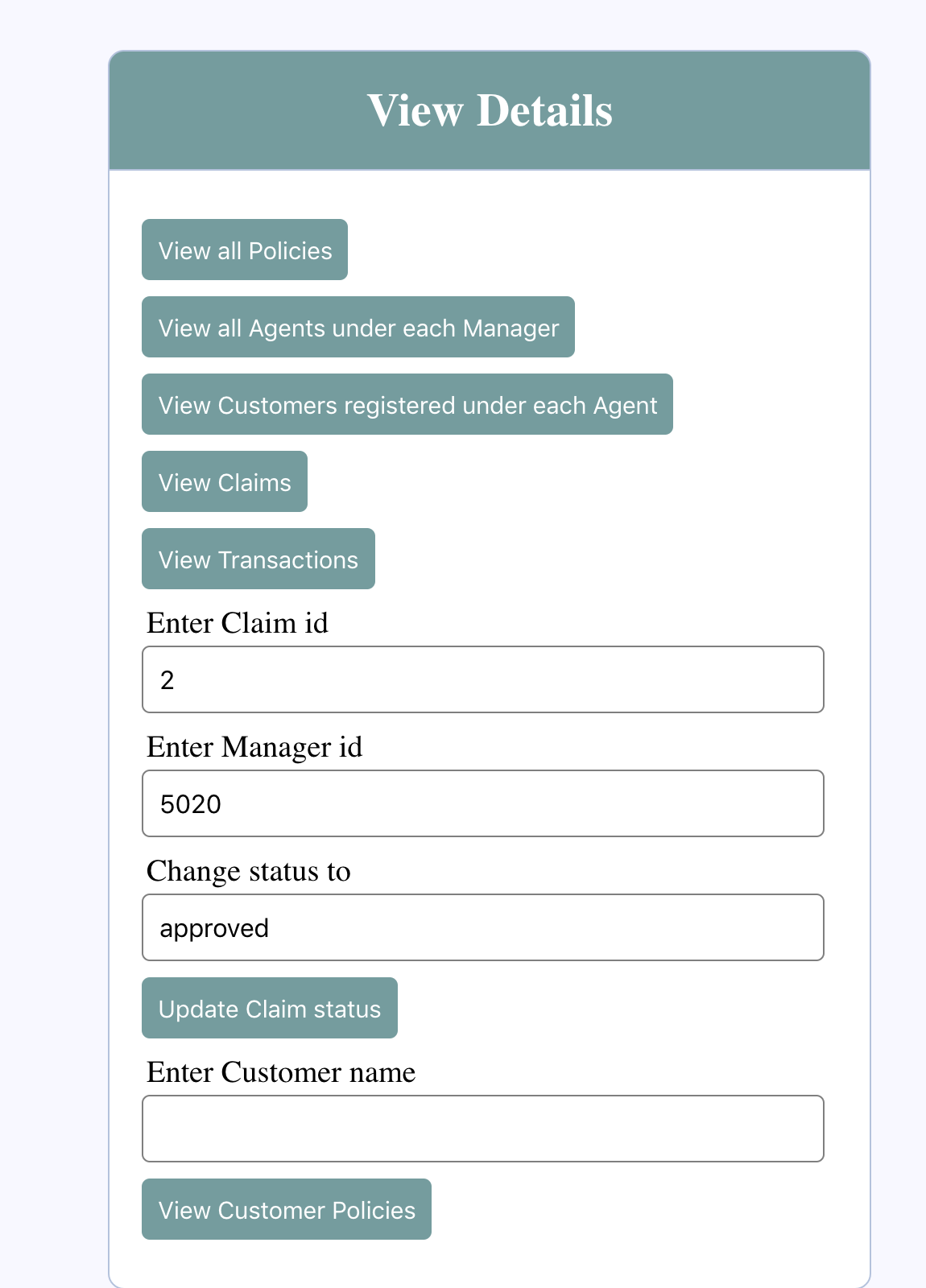
Normalization Justification : This table is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

**Front end screenshots**

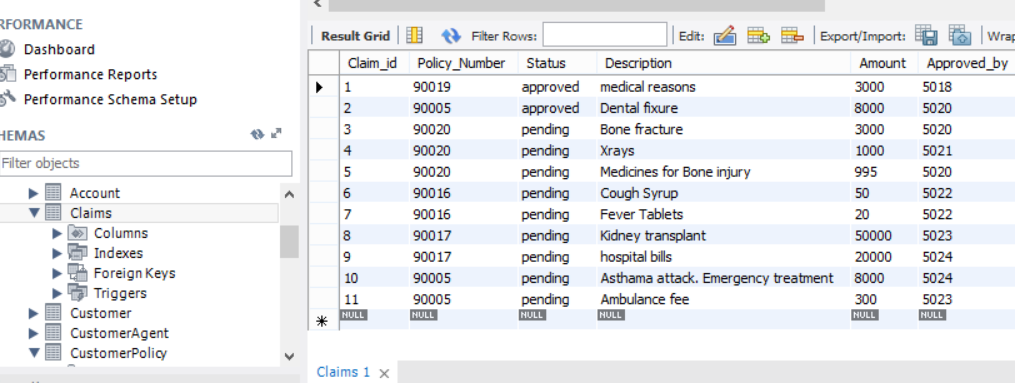
****

**Before updating claim status**

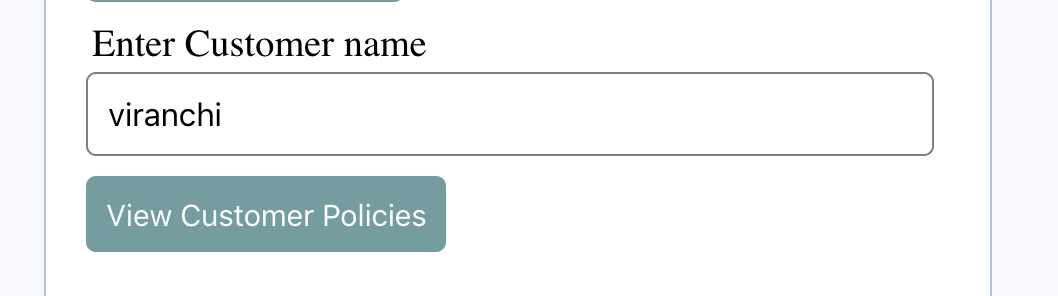
****

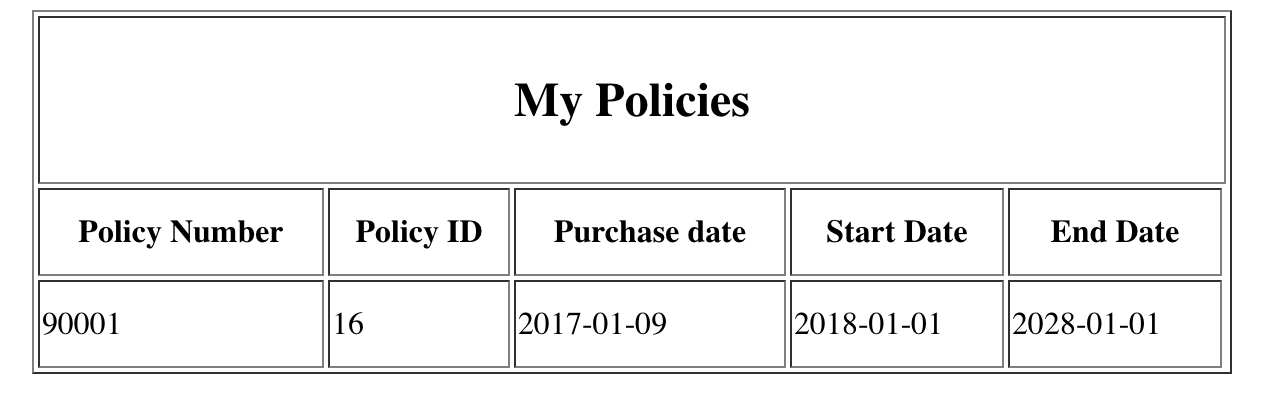
****

**After updating claim status**

****

**User viewing policy details**

****

****

**SPRINT 3**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Sprint 1** | **Sprint 2** | **Sprint3** | **Product Backlog** |

|  |  |
| --- | --- |
| **Story ID** | **Story description** |
| US1 | As a User\*, I can browse through multiple policies offered by the company. |
| US2 | As a Customer , I should be able to buy multiple policies. |
| US3 | As a Guest, I want to sign up with the system, so that I can buy policies. |
| US4 | As an Agent, I want to enroll customer in the policy. |
| US5 | As a Customer, I should be able to login to the system. |
| US6 | As a Customer, I want to file an insurance. |
| US7 | As a Customer, I want to check the status of claim. |
| US8 | As a Manager, I want to change the status of claim, so that I can either approve or keep it on hold. |
| US9 | As a Customer, I want to view the policies that I am currently enrolled. |
| US10 | As a Manager, I want to change the rules of existing policy so that I can address some existing problem if any. |
| US11 | As a Manager, I want to notify change of policy to policy holders, so that they are informed about the changes. |
| US12 | As an Agent, I want to see details about customers enrolled by me, so that I can keep track of how they are doing. |
| US13 | As a Customer, I want pay premium for my policies. |
| US14 | As a Manager, I can view all the agents working for me. |
| US15 | As an Agent, I can view all my customers and number of policies they hold. |
| US16 | As an Agent, I want to check my commission details. |
| US17 | As a Manager, I can view all the customers assigned to my agents and the number of policies they hold. |
| US23 | As a User\*, I should be able to perform advance search by providing combination of keywords and search for the policies offered by the company. |
| US25 | As a manager, I want to see commission earned by particular Agent by providing his employee id. |
| US26 | As a customer, I want to see my bill for the current month with any pending payments and late payment fees |
| US27 | As a customer, I want to see my previous transactions for the current month with any pending payments and late payment fees |

\*The rolename **User** is used in User Stories that apply to both Customer, guest.

**CONCEPTUAL DESIGN:**

**Entity : User\_personal\_details**

Attributes:

Username

First\_name

Last\_name

Date\_of\_birth [composite attribute]

Email\_id

Phone\_no [multi-valued attribute]

Address [composite]

Address\_line1

Address\_line2

City

State

Zip\_code

**Entity :Customer**

Attributes:

Username

Credit\_card

Expiry\_date

Name\_on\_card

**Entity: Employee**

**Attributes**

Username

Designation

Manager\_id

**Entity : Policy**

Attributes:

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

**Entity :Account**

Attributes:

Username

Password

Account\_type

**Entity :Transaction**

Attributes:

Amount

Description

Date

**Entity :Claims**

Attributes:

Status

Description

Amount

**Entity : Commission**

Attributes :

Commision\_amount

**Relationships:**

Relationship : **Customer** buys a **Policy**

Cardinality : Many to Many

Participation:

Customer has partial participation

Policy has partial participation

Relationship : **Employee(Agent)** serves **Customer**

Cardinality: One to Many

Participation:

Employee(Agent) has partial participation

Customer has partial participation

Relationship : **Employee(Agent)** works for **Employee(Manager)**

Cardinality: Many to One

Participation:

Employee (Agent) has total participation

Employee (Manager) has partial participation

Relationship : **Customer** has an **Account**

Cardinality: One to One

Participation:

Customer has total participation

Account has partial participation

Relationship : **Employee** has an **Account**

Cardinality: One to One

Participation:

Employee has total participation

Account has partial participation

Relationship: **Customer** files a **Claim**

Cardinality: One to many

Participation:

Customer has partial participation

Claim has total participation

Relationship: **Customer** makes **Transaction** by paying premium

Cardinality: One to many

Participation:

Customer has partial participation

Transaction has total participation

Relationship: **Employee(Manager)** changes status of **Claim**

Cardinality: One to many

Participation:

Manager has partial participation

Claim has total participation

Relationship: **Employee(Agent)** checks **Commission**

Cardinality: One to many

Participation:

Employee (Agent) has partial participation

Commission has total participation

**Logical Design**

Table:**Account**

Columns:

Username

Password

Account\_type

Justification : Username is unique for customer and employee, so its is chosen as primary key in account table.

Table:**User\_personal\_details**

Columns:

Username [foreign key; references Username of Account]

First\_name

Last\_name

Date\_of\_birth

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Justification:

Username is considered as primary key in User\_personal\_details as each customer and employee who signs up with the system chooses a unique username.

Also the multivalued attribute ‘Phone\_no’ is restricted to two values i.e ‘Phone\_no1’ and ‘Phone\_no2’ and two separate columns are made for that.

Table: **Customer**

Columns:

Username [ foreign key ; references Username of User\_personal\_details ]

Credit\_card\_number

Expiry\_date

Name\_on\_card

Justification: Username is the foreign key to the User\_personal\_details table which determines the customer’s personal details

Table:**Policy**

Columns:

Policy\_id

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

Justification: Each policy holds a unique policy\_id, so it is considered as the primary key.

Table:**Employee**

Columns:

Employee\_id

Designation

Username [Foreign key; references Username of Account]

Manager\_id [Foreign key; reference to Employee\_id of Employee]

Justification:

Username is chosen as foreign key to associate each employee with a Account.

As Agent works for Manager doesn’t has any special relationship attributes, we chose to include manager\_username as a foreign key to establish the relationship between Agent and Manager.

Table:**CustomerAgent**

Columns:

Customer\_username [Foreign key;references username of Customer]

Employee\_id(Agent) [Foreign key;references Employee\_id of Employee]

Justification:

Though the relationship is not many to many, we choose to have cross-reference table for Customer has an Agent relationship because of their participation constraint. A customer may or may not have an Agent and this may result in having null values in Customer Table.

Table: **CustomerPolicy**

Columns:

Policy\_Number

Customer\_username [Foreign key;references Username of Customer]

Policy\_id [Foreign key;references Policy\_id of Policy]

Purchase\_date

Start\_date

End\_date

Due\_date

Total\_amount

Justification:

The relation between customer and policy is many to many. Therefore having a cross-reference table is the only option with Policy\_Number as a primary key..Customer\_username is used as foreign key to associate with Customer table.Policy\_id is used as foreign key to associate with policy table.

Table:**Claim**

Columns :

Claim\_id

Status

Policy\_Number [Foreign key reference to Policy\_Number of CustomerPolicy table]

Description

Amount

Approved\_by [Foreign key reference to Employee\_id of Employee table]

Justification: The Policy\_Number links the claim\_id with corresponding policy\_id in the customerpolicy table. Therefore Policy\_Number is set as foreign key to the customerPolicy table.Approved\_by will tell which Employee(Manager) is taking care of the claim.

Table:**Transaction**

Columns:

Transaction\_id

Policy\_Number [Foreign key reference to CustomerPolicy table]

Amount

Description

Date

Justification:

A policy bought by customer will have a policy number. And there will be multiple premium installments for each insurance. To uniquely identify, we need to have a transaction ID which will be a primary key in this table.

policy number, a foreign key to customer policy table will signify for which user’s policy is the premium being paid.

Table:**Commission**

Columns:

Policy\_number [Foreign key reference to CustomerPolicy table]

Employee \_id [Foreign key references Employee table ]

Commission\_amount

Justification: Primary key is policy number because the agent will get commission only once for a policy bought by customer under him. This will be unique.

**Normalization Logical Design**

Table:Account

Columns:

Username

Password

Account\_type

Normalization Justification : Account is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:User\_personal\_details

Columns:

Username [foreign key; references Username of Account]

First\_name

Last\_name

Date\_of\_birth

Email\_id

Phone\_no1

Phone\_no2

Address\_line1

Address\_line2

City

State

Zip\_code

Normalization Justification : 2NF(Same zip\_code can be used by multiple cities. Hence, creating a seperate table for zip codes, cities and states having primary key as zip\_code is not possible. In such case we will need to create a composite key having zip\_code and city and add it as foreign key to User\_table table which is not feasible. Also, zip\_code, city and state are used only by User\_detail table, hence there is no point in creating a seperate table which will not be used by any other table.Creating a seperate table increases unnecessarily joins which slower performance. Hence, we avoided creating seperate table unless its absolute necessary.

Table:Customer

Columns:

Username [ foreign key ; references Username of User\_details ]

Credit\_card\_number

Expiry\_date

Name\_on\_card

Normalization justification: The table is in 2NF. The customer is assumed to have only one credit card, and credit card information is used only by the customer table so there is no point in creating a separate table which will result in using unnecessary joins.

Table:Policy

Columns:

Policy\_id

Policy\_name

Description

Premium

Coverage

Years

Policy\_type

Normalization Justification :Policy is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:Employee

Columns:

Employee\_id

Designation

Username [Foreign key; references Username of Account]

Manager\_id [Foreign key; reference to Employee\_id of Employee]

Normalization Justification:

The table is in 3NF. The table is not in BCNF because the non prime attribute ‘Username’ can determine the primary attribute - Employee\_id.

The Employee\_id attribute is introduced and made primary key for easy reference and linking to other table. Further normalization of this table is not beneficial and will result in over normalization.

Table:CustomerAgent

Columns:

Customer\_username [Foreign key;references username of Customer]

Agent\_Employee\_ID [Foreign key;references Employee\_id of Employee]

Normalization Justification: 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:CustomerPolicy

Columns:

Policy\_number

Customer\_username [Foreign key;references Username of Customer]

Policy\_id [Foreign key;references Policy\_id of Policy]

Justification:

The relation between customer and policy is many to many. Therefore having a cross-reference table is the only option with Policy\_Number as a primary key..Customer\_username is used as foreign key to associate with Customer table.Policy\_id is used as foreign key to associate with policy table.

Normalization Justification:4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:CustomerPolicyDetails

Columns:

Policy\_number

Purchase\_date

Start\_date

End\_date

Due\_date

Current\_bill\_amount

Justification: The table is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Because the Policy\_id and username could uniquely identify a policy details in previous approach, the table was not satisfying the 3NF requirements. Hence we created this table to achieve 4NF.

Table:**Claim**

Columns :

Claim\_id

Status

Policy\_Number [Foreign key reference to Policy\_Number of CustomerPolicy table]

Description

Amount

Approved\_by [Foreign key reference to Employee\_id of Employee table]

Justification:

The Policy\_Number links the claim\_id with corresponding policy\_id in the customer\_policy table. Therefore Policy\_Number is set as foreign key to the customer\_policy table.

Approved\_by will tell which Employee(Manager) is taking care of the claim.

Normalization Justification :4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:**Transaction**

Columns:

Transaction\_id

Policy\_Number [Foreign key reference to CustomerPolicy table]

Amount

Description

Date

Justification:

A policy bought by customer will have a policy number. And there will be multiple premium installments for each insurance. To uniquely identify, we need to have a transaction ID which will be a primary key in this table.

policy number, a foreign key to customer policy table will signify for which user’s policy is the premium being paid.

Normalization Justification : This table is in 4NF(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Table:**Commission**

Columns:

Policy\_number[Foreign key reference to CustomerPolicy Table]

Employee\_id[Foreign Key reference to Employee table]

Commission\_amount

The table is in 4NF

(There are no transitive or partial dependencies among the attributes. Since the attributes are dependent on the primary key, there will be no insertion, updation or deletion anomalies for this entity, so it is in 3NF. Also,there are no multivalued dependencies among the attributes.So,it is in 4NF.)

Justification: The policy number in this table determines both the agent and the commision amount that the agent will get.

**Indexes**

Table : Account

Clustered Index : Username

Justification : It is a default index as username is the primary key of Account table.

Also Most of the queries searches and updation are performed with the help of primary key which is Username. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Table : User\_personal\_details :

Clustered Index : Username

Justification :It is a default index as username is the primary key of User\_personal\_details table.Also Most of the queries searches and updation are performed with the help of primary key which is Username. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non - Clustered Composite Index : last\_name , first\_name

Justification : Many searches and most of the queries are based on last\_name and also most of queries will have first\_name in the select clause along with last\_name so we chose to make it a composite index.

Non - Clustered Index : City

Justification : Many queries will have city in the group by as well as select clause for the analysis purpose and for searching.

Table : Customer

Clustered Index : Username

Justification : It is a default index as username is the primary key of Customer table.

Also Most of the queries searches and updation are performed with the help of primary key which is Username. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Table:Policy

Clustered Index : Policy\_id

Justification : It is a default index as username is the primary key of Policy table. Also Most of the queries searches and updation are performed with the help of primary key which is Policy\_id.And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non - Clustered Index : Policy\_type

Justification: Most of the queries and searches made by the customers will be based on type of policy. Also managers might need policy type in group by queries for analysis purpose.

Non - Clustered Index : Coverage

Justification : Customers will frequently search policies based on coverage and this event will be very frequent.

Table : Employee

Clustered Index :Employee\_id

Justification : It is a default index as it is the primary key of CustomerAgent table. Also Most of the queries searches and updation are performed with the help of primary key. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non Clustered Index :Username

Justification: It is default index as it the foreign key and most queries will need this index for searching or updating values.

Non Clustered Index: Manager\_id

Justification: It is default index as it the foreign key and most queries will need this index for searching or updating values.

Table:CustomerAgent

Clustered Composite Index : Customer\_username,Agent\_Employee\_id

Justification : It is a default index as it is the primary key of CustomerAgent table. Also Most of the queries searches and updation are performed with the help of primary key. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non - Clustered Index : Agent\_Employee\_id

Justification : It is default index as it the foreign key and most queries will need this index for searching or updating values.

Table : CustomerPolicy

Clustered Index : Policy\_number

Justification : It is a default index as it is the primary key of CustomerAgent table. Also Most of the queries searches and updation are performed with the help of primary key. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non Clustered Index : Customer\_username

It is default index as it the foreign key and most queries will need this index for searching or updating values.

Non Clustered Index : Policy\_id

It is default index as it the foreign key and most queries will need this index for searching or updating values.

Table CustomerPolicyDetails:

Clustered Index : Policy\_number

Justification : It is a default index as Policy\_number is the primary key of CustomerPolicyDetails table. Also Most of the queries searches and updation are performed with the help of primary key which is Policy\_number.And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Table : Claim

Clustered Index: Claim\_id

Justification: It is a default index as is the primary key of Claim table. Also Most of the queries searches and updation are performed with the help of primary key. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non - Clustered Index: Policy\_Number

Justification: It is default index as it the foreign key and most queries will need this index for searching or updating values.

Non - Clustered Index: Approved\_by

Justification: It is default index as it the foreign key and most queries will need this index for searching or updating values.

Non - Clustered Index : Status

Justification : Status will be used frequently by employee to find out claim which are pending, approved or declined.

Table : Transaction

Clustered Index :Transaction\_id

Justification : It is a default index as is the primary key of Transaction table. Also Most of the queries searches and updation are performed with the help of primary key. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key.

Non - Clustered Index : Policy\_Number

Justification: It is default index as it the foreign key and most queries will need this index for searching or updating values.

Non - Clustered Index : Date

Justification: Date will be used in group by queries for employee to search transaction of particular month or week.

Table:Commission

Clustered Index :Policy\_number

Justification: It is a default index as is the primary key of Transaction table. Also Most of the queries searches and updation are performed with the help of primary key. And the strong reason to make this the index because data is of variable length so to avoid full file scan,we create index of primary key

Non - Clustered Index : Employee\_id

Justification: It is default index as it the foreign key and most queries will need this index for searching or updating values.

**Sprint 3:**

**Stored Programs and Views:**

**Views:**

**1. Customer\_Transactions\_View**

Goal:

This view will help eliminate multiple joins for getting transaction records for user.

This view will be used by stored procedure as it will just require a username to get all the details about the transaction of that specific username.

**2.Customer\_claim\_view.**

Goal:

This view will be used when a customer will want to see claims he/she has made by entering his/her username, so joins will be eliminated and results will be given faster. Also this view will be used by manager to view claims under him.

**3. Commission\_view**

Goal:

This view is used when agent wants to see his/her commission details.

**4. BILL View**

Goal:

This view is used by customer to see their bill information i.e. their current bill amount with any previous pending amount if any for a particular policy she/he owns.

**5.Transaction view for managers.**

Goal :

This view will be used more frequently by managers to see customer and their transactions.

**TODO:**

1. **A view for commision and Bills**
2. **Stored procedure making use of views that we created above.**
3. **Commission and Bill automation using events**
4. **And any unnoticed functionalities**

**Trigger**

1. setTotalToZero trigger:

Goal:

Where there an entry will be inserted into transaction table, this trigger will make the Current\_bill\_amount to 0 for the corresponding Policy\_Number in CustomerPolicyDetails table

**Event**

1.BILLUPDATER

Goal :

This event will run daily.The main purpose of this event to check each policy number’ s due date and current\_bill\_amount in CustomerPolicyDetails and if the date has passed the due\_date it will call a stored procedure to make updation accordingly i.e if bill not pay and due-date passed then it should apply additional charges along with new premium to total amount or if premium is paid on time then only new premium amount should get updated along with new due\_date.

Delimiter |

CREATE EVENT BILLUPDATER

ON SCHEDULE

EVERY 1 DAY

DO

BEGIN

-- Get count of rows in customer policy details

DECLARE numberOfPolicies INT;

DECLARE counter INT;

DECLARE total INT;

DECLARE due\_date DATETIME;

DECLARE aaj DATETIME;

DECLARE premium INT;

DECLARE policy\_number INT;

DECLARE somethingToDelete INT;

SET aaj = NOW();

SET counter = 0;

SELECT COUNT(\*) INTO numberOfPolicies FROM CustomerPolicyDetails;

WHILE counter < numberOfPolicies DO

Select cp.Policy\_Number, cp.due\_date, p.premium ,cp.Current\_bill\_amount INTO policy\_number, due\_date, premium, total

from Policy p inner join CustomerPolicy c using (Policy\_id) inner join CustomerPolicyDetails cp

using (Policy\_Number) order by Policy\_Number

LIMIT counter,1;

IF(aaj > due\_date) THEN

IF total = 0 THEN

SET total = premium;

SET due\_date = DATE\_ADD(due\_date, INTERVAL 30 DAY);

ELSE

SET total = total + premium + (0.05 \* premium);

SET due\_date = DATE\_ADD(due\_date, INTERVAL 30 DAY);

END IF;

CALL UpdateBill(total,due\_date,policy\_number);

END IF;

SET counter = counter + 1;

END WHILE;

END |

Delimiter ;

**Stored Procedures**

**1.CommissionDetails**

Parameters:IN username varchar(50)

Goal: This stored procedure gives details of commission for each agent.This takes Username as input parameter.This procedure lists out commission earned for each policy sold to customer by a agent.

**2.GetMyTransactions**

Parameters: IN username varchar(50)

Goal:This Stored procedure gives details of each transaction done by each customer for each policy taken by him.This procedure lists out policy number,policy id,policy name,transaction amount,transaction date for each customer.

**3.ShowMyClaims**

Parameters:IN Username varchar(11)

Goal:This Stored procedure gives details of each claim made by a customer.This procedure lists out policy details of each claim,claim status,claim amount and manager associated with it.

**4.UpdateBill**

Parameters:IN total\_param FLOAT, IN due\_date\_param DATE, IN policy\_number\_param INT

Goal:This procedure is called by event when certain due date is greater than current date.This procedure update CustomerPolicyDetails table with amount due to be paid by a customer for each policy he has taken and due date for next payment.

**5.BuyPolicy**

Parameters:IN username VARCHAR(30), IN policy\_id INT

Goal:This procedure inserts details of new policy purchased by a customer into CustomerPolicy table and CustomerPolicyDetails.This Procedures updates the tables with new policy number ,premium,maturity period,date when the policy is taken.

**Functions :**

1. Stored function: **GetPolicyMaturity**

Parameters: Policy\_id (INT)

Goal: This function returns the maturity of the policy. It is used when we are calculating the end date of the policy whenever a customer buys a policy.

1. Stored function: **GetPolicyNumber**

Parameters: Username VARCHAR(30), Policy\_id (INT)

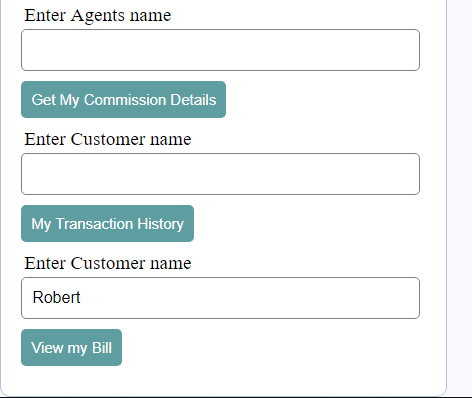
Goal: This function returns the unique policynumber assigned to a unique combination of policy id and username. Used inside Buy policy stored procedure.

1. Stored function: **GetPolicyPremium**

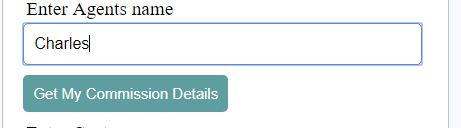
Parameters: Policy\_id (INT)

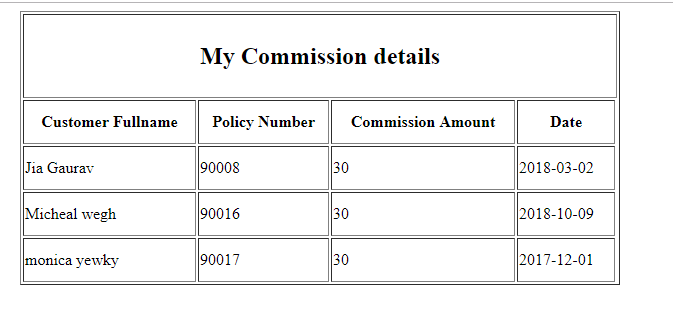
Goal: This function returns the premium amount of the policy. It is used when we are calculating the bill of the policy whenever a customer buys a policy.

Frontend screenshots:

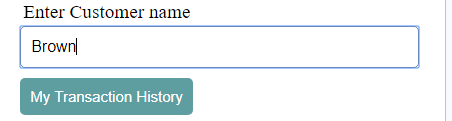


Agent can check for commission details by providing his name:





Customer/User can his previous transactions :



Customer can check his bill statement:

