IST 659 Data Admin Concepts and Database Management

Project Report

Insurance Claim Management System

Tejas Vishwanath Gawade
Shrish Kiran Vaidya
Vedant Devidas Patil

Introduction:

In this insurance claim management system, where customers can seek new claims and study the details of their existing claims, allows customers and administrators to connect with one another. Logging in, reviewing customer requests, and accepting or deny them. Each customer and Admin have unique login credentials. A new customer can sign up by putting a request which will be accepted by the employee. A customer can put claim request, which will be rejeccted/accepted by the admin, The customer can see his pending request which the customer has made in past. Customer can also see the request which are declined by the admin and he acan also see the reason why the request was declined.

Problem Statement:

There are certain companies in the insurance sector with modern insurance claim portals for clients and employees. However, several new, modestly sized insurance startups do not yet have fully functional claim handling systems. They contain too many features and functions, which makes it more difficult to navigate the website and may result in several errors. As a result, we have made an effort to create a claim management system that is simple and easy to use for both customers and employees.

Overview of Entities, Tables and its attributes

The Insurance database has 2 main entities:

- Customer: Customer is the main entity for which the application is designed and the customer can put an insurance claim requests, request to be the part of the insurance plan, can chosse various policy types and policy sub types. Customer has uniue lohin credentials. If the customer has 1 family member then he considered as non-premium customer and if the customer 1 or many family members then he considered as premium customer.
- Admin: The functionality of the admin is that admin can accept or rejected new customer requests, the admin can accept and reject the insurance calim request made by the customer. Admin also haves unique login credentials.

Tables:

Customers

Attribute Name	**Description**
Customer fname	first name of the customer
Customer lname	last name of the customer
Customer dob	date of birth of the customer
Customer mobile number	contact information of the customer
Customer zipcode	area code where the customer resides
policy type id	Id of the primary policy that customer want
gender	gender information of customer
Customer email	email address of the customer
Login id	id with which customer can login in application
Policy sub type id	id of the secondary policy which customer wants
state_code	state code of the customer lives in

Admin:

Attribute Name	**Description**
employee id	employee id provided by the company
employee fname	first name of the employee
employee lname	last name of the employee
login id	id with which employee can log in the application
employee dob	date of birth of the empolyee

Customer login details:

Attribute Name	**Description**

login id	login id of the customer
password	password for the customer

Admin login details:

Attribute Name	**Description**
login id	login id of the employee
password	password for the employee
employee id	employee id of the employee

pending request:

Attribute Name	**Description**
request id	id of each request made by the customer
amount requested	amount request by the customer
request date	date on which request was made
disease id	id of the disease which customer selects
Customer fname	requested customer first name
Customer lname	requested customer last name
Customer dob	requested customer date of birth
Customer mobile number	requested customer mobile number
Customer login id	login id of the customer
Customer request type	customer request is premium or not
Amount approved	if approved amount approved by the employee
Approval id	if approved request approved or not
Rejected reason	if rejected what is the reason
request reviewed by	employee who reviews the request

approved request:

Attribute Name	**Description**
Approved request id	id of the each request approved by the employee
Requested amount	amount requested by the customer
Amount approved	amount approved by the employee
Requested approved by	id of employee who approved the request
Customer fname	first name of the customer whose request is approved
Customer lname	last name of the customer whose request is approved
Customer dob	date of birth of the customer whose request is approved
Customer gender	gender of the customer whose request is approved
Customer mobile number	mobile number of the customer whose request is approved
Customer login id	customer login id of the customer whose request is approved

policy types:

Attribute Name	**Description**
Policy id	unique id for the policy avaibles
Policy name	name of each policy

decline request:

Attribute Name	**Description**
request id	id of the request declined by the employee
amount requested	amount request by the customer
request declined by	id of employee who declined the request

reason declined	reason why the request was declined
customer fname	first name of the customer whose request is declined
customer lname	last name of the customer whose request is declined
customer dob	date of birth of the customer whose request is declined
customer gender	gender of the customer whose request is declined
customer mobile number	mobile number of the customer whose request is declined

approval status:

Attribute Name	**Description**
status id	id of the approval status
approval status	approval status yes or no

new customer request:

Attribute Name	**Description**
login id	uniuqe login id for the new customer
customer fname	first name of the new customer
customer lname	lname of the new customer
dob	Date of birth of new customer
gender	gender of the new customer
email id	Email I'd of the new customer
state code	State code of the new customer
zip code	Zip code of the new customer
moible number	Contact information of the new customer
policy type	Name of the customer policy type
policy subtype	Name of the customer policy sub type
family members	Number of family members of the customer

approval status id	approved status yes or no of the request
department	premeium or non-permeium department assigned

premium customers:

Attribute Name	**Description**
customer id	id of the premium customer
customer fname	first name of the premium customer
customer lname	last name of the premium customer
customer dob	date of the birth of the premium customer
zipcode	zipcode the premium customer
customer moible number	mobile number of the premium customer
policy type	policy type of selected by premium customer
policy sub type	policy sub type selected by premium customer
gender	gender of the premium customer
customer email	email address of premium customer
state code	state code of premium customer
customer login id	login id of the premium customer
family id	family id of the customer

family:

Attribute Name	**Description**
family id	family id of the family
family lastname	last customer whose family it is
customer login id	login id of the customer
number of member	number of family menbers

state_lookup:

Attribute Name	**Description**
state_code	code of the state
state_name	name of the state

gender_lookup:

Attribute Name	**Description**
gender_id	id of each gender
gender	name of the gender

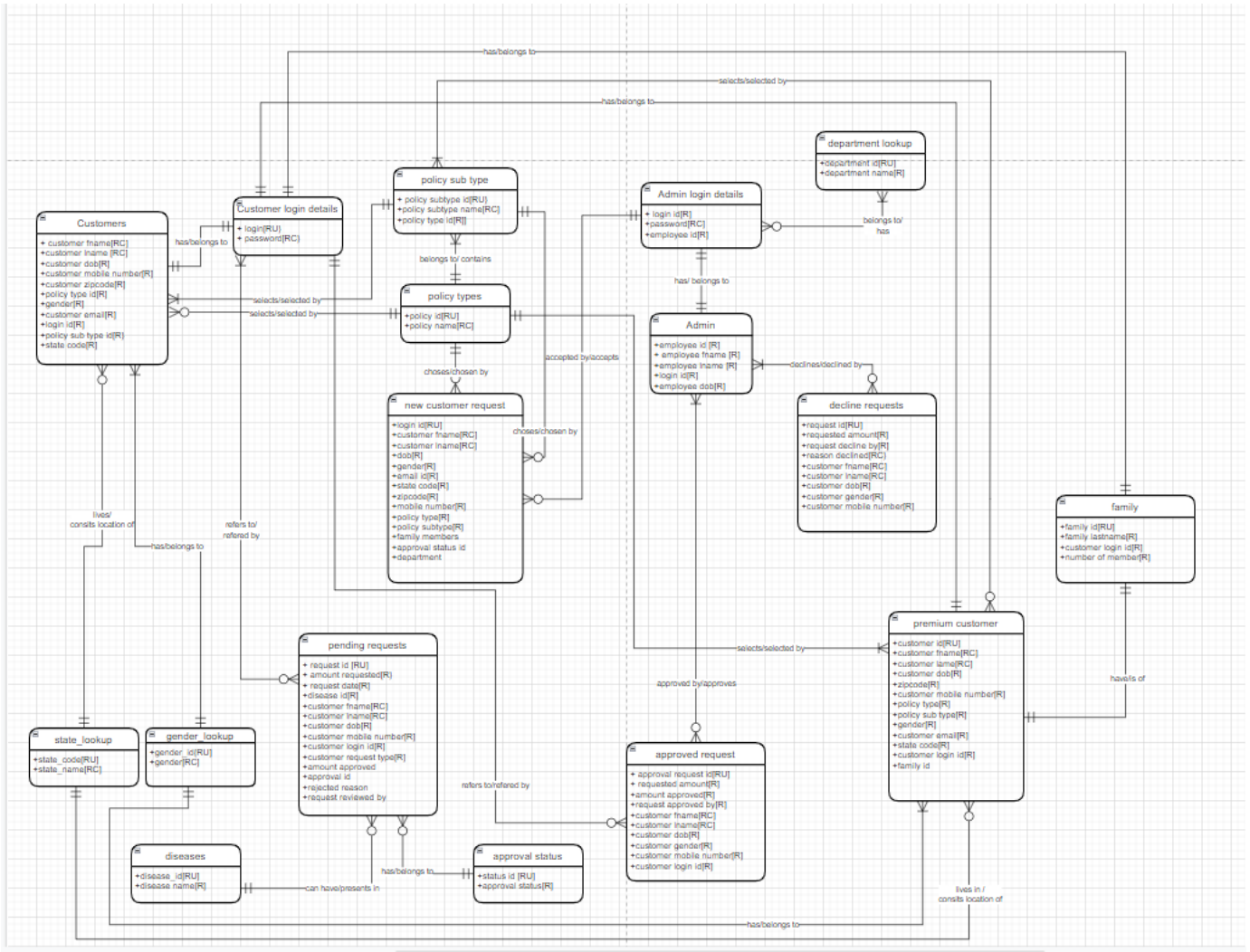
department_lookup:

Attribute Name	**Description**
gender_id	id of each gender
gender	name of the gender

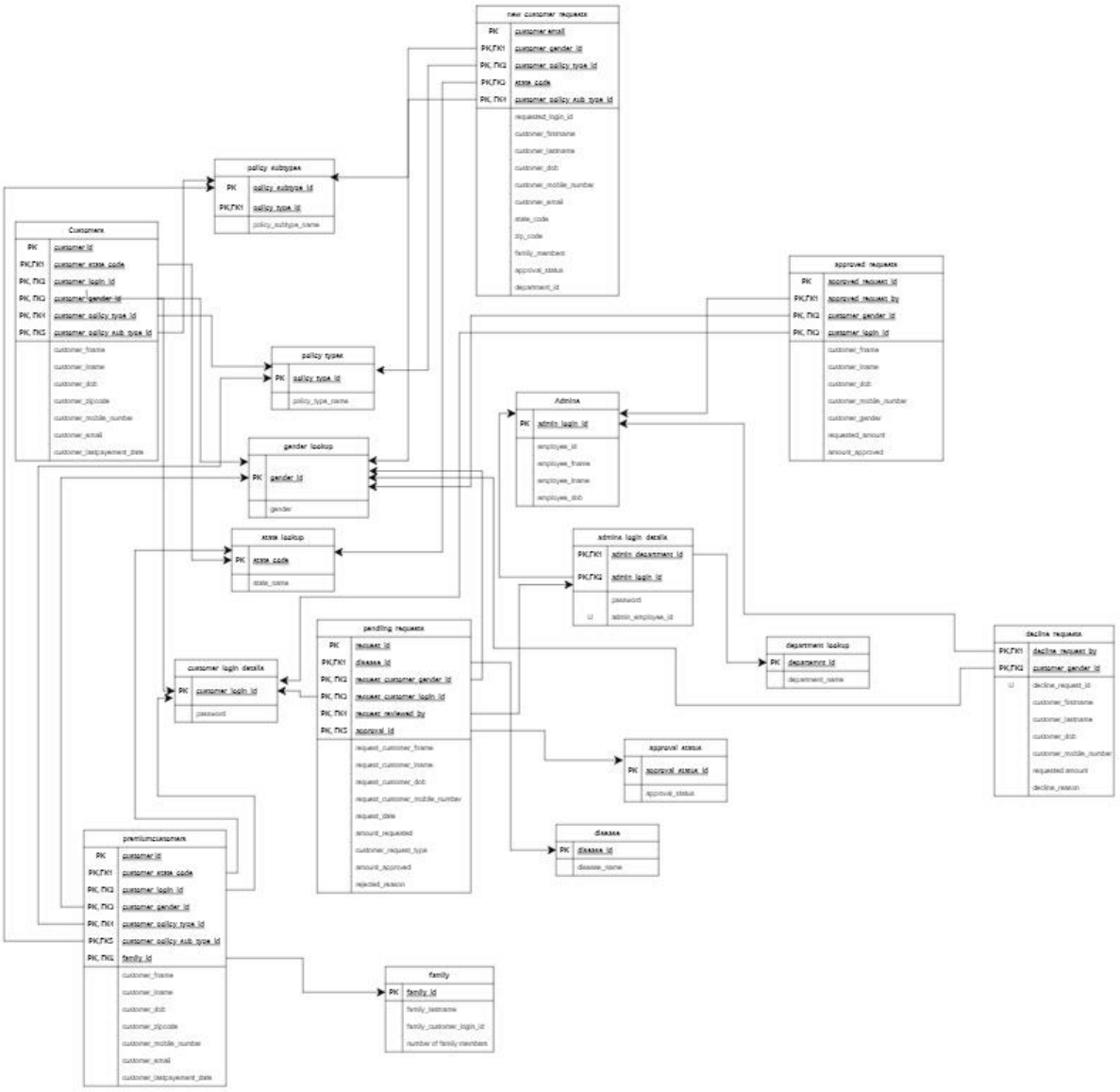
diseaes:

Attribute Name	**Description**
disease id	id of the disease
disease name	name of the disease

Conceptual Model:



Logical Model:



DATABASE DESIGN AND IMPLEMENTATION

Now, let's understand how we designed the tables, the relationship between different entities through below screenshot.

Relationships					
Relationship	**Entity**	**Rule**	**Min**	**Max**	**Entity**
Customers State lookup	**Customers**	lives in	1	1	**State lookup**
	State lookup	consits location of	0	M	**Customers**
Customers Customer login details	**Customers**	has	1	1	**Customer login details**
	Customer login details	belongs to	1	1	**Customers**
Customers gender_lookup	**Customers**	has	1	1	**gender_lookup**
	gender_lookup	belongs to	1	M	**Customers**
Customers policy type	**Customer**	selects	1	1	**policy type**
	policy type	selected by	0	M	**Customers**
Customers policy sub-type	**Customers**	selects	1	1	**policy sub-type**
	policy sub-type	selected by	1	M	**Customers**
policy sub type policy type	**policy sub type**	belongs to	1	1	**policy types**
	policy type	contains	1	M	**policy sub type**
Admin login details departement lookup	**admin login details**	belongs to	1	M	**department lookup**
	department lookup	has	0	M	**admin login details**
Admin Admin login details	**Admin**	has	1	1	**Admin login details**
	Admin login details	belongs to	1	1	**Admins**
Admin decline requests	**Admin**	declines	0	M	**decline requests**
	decline requests	declined by	1	M	**Admin**

new customer request Admin	**new customer request**	accepted by	1	1	**Admin**
	Admin	accepts	0	M	**new customer request**
pending request diseases lookup	**pending request**	can have	1	1	**disease**
	disease	presents in	0	M	**pending requet**
pending request customer login details	**pending request**	refers to	1	M	**customer login details**
	customer login details	reffered by	0	M	**pending request**
pending request Admin	**pending request**	accepted by	1	1	**Admin**
	Admin	accepts	0	M	**pending request**
pending request approval status	**pending request**	has	1	1	**approval status**
	approval status	belongs to	0	M	**pending requests**
approved request Admin	**approved request**	approved by	1	M	**Admin**
	Admin	approves	0	M	**approved request**
approved request customer login details	**approved request**	refers to	1	1	**customer login details**
	customer login details	referred by	0	M	**approved request**
new customer request state lookup	**new customer request**	refers to	1	1	**state lookup**
	state lookup	reffered by	0	M	**new customers request**
new customer request policy type	**new customer request**	chooses	1	1	**policy type**
	policy type	chosen by	0	M	**new customer request**
new customer request policy sub type	**new customer request**	chooses	1	1	**policy sub type**
	policy sub type	chosen by	0	M	**new customer request**
premium customers State lookup	**premium customers**	lives in	1	1	**State lookup**
	State lookup	consits location of	0	M	**premium customers**

Relationships					
Relationship	**Entity**	**Rule**	**Min**	**Max**	**Entity**
premium customers Customer login deta	**premium customers**	has	1	1	**Customer login details**
	Customer login details	belongs to	1	1	**premium customers**
premium customers gender_lookup	**premium customers**	has	1	1	**gender_lookup**
	gender_lookup	belongs to	1	M	**premium customers**
premium customers policy type	**premium customers**	selects	1	1	**policy type**
	policy type	selected by	0	M	**premium customers**
premium customers policy sub-type	**premium customers**	selects	1	1	**policy sub-type**
	policy sub-type	selected by	1	M	**premium customers**
premium customers family	**premium customers**	have	1	1	**family**
	family	is of	1	1	**premium customers**
family customer login details	**family**	has	1	1	**customer login details**
	customer login details	belongs to	1	1	**family**

The first row at the top of the table above represents the Customer State Lookup relationship. It shows that a customer lives in maximum one state and a state consists of multiple of Customers.

The second rule suggest that one Customer has one login details; a particular login detail is owned by one and only one unique customer.

The relationship between Customer and gender portray that one customer belongs to one and only one gender whereas a gender can associated to multiple customers.

Customer Policy Type relationship suggest that one customer selects only one policy type; one policy type can be selected by none or many customers.

These are the few examples of ER relationship that explains us how we read and understand the entity relationship. After defining the relationship between different entities, next important step is to start implementing the database by creating the tables and inserting values using the roadmap that we have created in the previous steps.

To begin, we have created the table “new_customer_requests” and inserted the values in the table. Also, we define the primary key, foreign key, unique key etc in the create table syntax and inserted few data into the table to test the table and its data.

Below are some of the many tables that we have created into the database. These screenshots shows us how to create tables, define relationships and insert values.

1) new_customer_requests

```
CREATE TABLE new_customer_requests
    (
        requested_login_id varchar(20) not null,
        customer_firstname varchar(50) not null,
        customer_lastname varchar(50) not null,
        customer_dob date not null,
        customer_gender_id int not null,
        customer_email varchar(50) not null,
        state_code char(2) not null,
        zip_code int not null,
        mobile_number int not null,
        policy_type_id int not null,
        policy_sub_type_id int not null,
        family_members int not null,
        approval_status_id int,
        department_id int


        CONSTRAINT pk_new_customer_requests_customer_email primary key(customer_email),
        CONSTRAINT fk_new_customer_requests_customer_gender_id foreign key(customer_gender_id)
        REFERENCES genders_lookup(gender_id),
        CONSTRAINT fk_new_customer_requests_policy_type FOREIGN key (policy_type_id)
        REFERENCES policy_types(policy_type_id),
        CONSTRAINT fk_new_customer_requests_policy_sub_type FOREIGN key (policy_sub_type_id)
        REFERENCES policy_subtypes(policy_subtype_id),
        constraint fk_new_customer_requests_state_code FOREIGN key (state_code)
        REFERENCES state_lookup(state_code)
    );
```

Inserting Values:

```
insert into new_customer_requests
values('TomH', 'Tom', 'Hardy', '12/06/1999', 1, 'TharDy@gmail.com', 'NY', 13210, 315886758, 1 , 2, 1, NULL, Null),
    ('PenG', 'Penny', 'Green', '12/08/1993', 2, 'Peng@gmail.com', 'NY', 13210, 315886756, 2, 2, 3, NULL, Null),
    ('MelRa', 'Melissa', 'Rauch', '12/15/1986', 3, 'MelissaR@gmail.com','NY', 13210, 315886759, 1, 1, 4, null, Null),
    ('JarrodS', 'Jarrod', 'Sally', '12/18/1989', 4, 'JarrodS@gmail.com', 'NY', 13210, 315886754, 2, 1, 3, Null, NULL)

update new_customer_requests set approval_status_id = 1, department_id = 1 WHERE requested_login_id = 'PenG'
update new_customer_requests set approval_status_id = 2, department_id = 2 where requested_login_id = 'TomH'
```

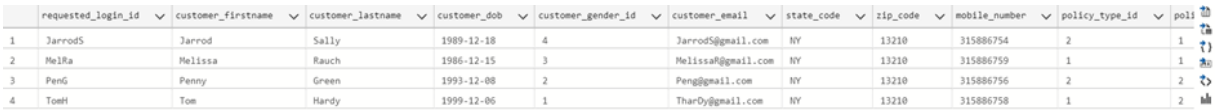
Triggers used:

There is a trigger used on new_customer_requests table, whenever the approval_status_id and department_id is updated in new_customer_requests table the trigger gets fired. So, basically when the trigger is fired then if the approval_status_id is set to 1 and department_id is 1 then the customer is added to premium customer. So, the customer is added to customer or premiumcustomers table is decided based on

department id. If department_id is 1 then it is stored premiumcustomer and when department_id is 2 then the customer is added into customers table. Also, of the approval_status_id is 2 then the request is rejected and request is deleted from new_customer_requests.

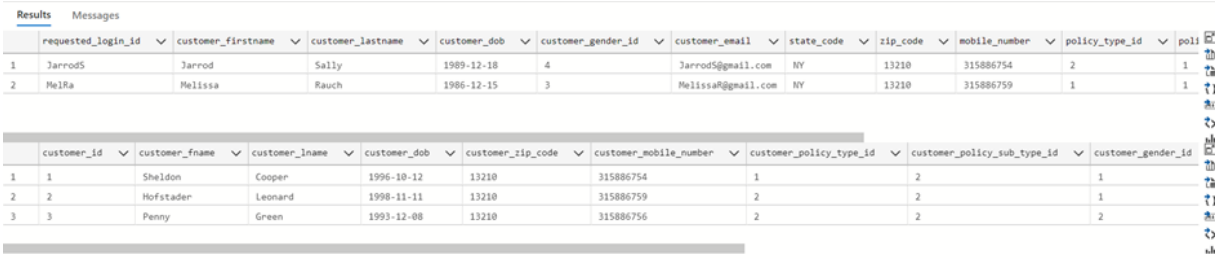
Output:

new_customer_requests:



	requested_login_id	customer_firstname	customer_lastname	customer_dob	customer_gender_id	customer_email	state_code	zip_code	mobile_number	policy_type_id	poli
1	JarrodS	Jarrod	Sally	1989-12-18	4	JarrodS@gmail.com	NY	13210	315886754	2	1
2	MelRa	Melissa	Rauch	1986-12-15	3	MelissaR@gmail.com	NY	13210	315886759	1	1
3	PenG	Penny	Green	1993-12-08	2	Peng@gmail.com	NY	13210	315886756	2	2
4	TomH	Tom	Hardy	1999-12-06	1	TharDy@gmail.com	NY	13210	315886758	1	2

After running the update statements:



Results Messages

	requested_login_id	customer_firstname	customer_lastname	customer_dob	customer_gender_id	customer_email	state_code	zip_code	mobile_number	policy_type_id	poli
1	JarrodS	Jarrod	Sally	1989-12-18	4	JarrodS@gmail.com	NY	13210	315886754	2	1
2	MelRa	Melissa	Rauch	1986-12-15	3	MelissaR@gmail.com	NY	13210	315886759	1	1

	customer_id	customer_fname	customer_lname	customer_dob	customer_zip_code	customer_mobile_number	customer_policy_type_id	customer_policy_sub_type_id	customer_gender_id
1	1	Sheldon	Cooper	1996-10-12	13210	315886754	1	2	1
2	2	Hofstader	Leonard	1998-11-11	13210	315886759	2	2	1
3	3	Penny	Green	1993-12-08	13210	315886756	2	2	2

We can see the requests are deleted from new customer_requests and the request with approval_status_id =1 has been added to premiumcustomers.

2) Customers

```
CREATE TABLE customers (
    customer_id int IDENTITY NOT NULL,
    customer_fname VARCHAR(20) NOT NULL,
    customer_lname VARCHAR(20) NOT NULL,
    customer_dob DATE NOT NULL,
    customer_zip_code int NOT NULL,
    customer_mobile_number VARCHAR(10) NOT NULL,
    customer_policy_type_id int NOT NULL,
    customer_policy_sub_type_id int Not Null,
    customer_gender_id int NOT NULL,
    customer_email VARCHAR(50) NOT NULL,
    customer_lastpayement_date DATE,
    customer_state_code CHAR(2) NOT NULL,
    customer_customer_login_id VARCHAR(20) NOT NULL

    CONSTRAINT pk_customers_customer_id PRIMARY Key(customer_id)
)
```

Insert:

```
Insert into customers
values ('Shrish', 'Vaidya', '12-31-2000', 13210, 315886754, 1, 2, 1, 'Shrish._Vaidya@gmail.com', NULL, 'NY', 'Shrish'),
('Kevin', 'Weiss', '12-07-2000', 13210, 315887755, 2, 1, 1, 'KWeiss@gmail.com', Null, 'NY', 'Kevin')
```

Output:

	customer_id	customer_fname	customer_lname	customer_dob	customer_zip_code	customer_mobile_number	customer_policy_type_id	customer_policy_sub_type_id	customer_gender_id
1	1	Shrish	Vaidya	2000-12-31	13210	315886754	1	2	1
2	2	Kevin	Weiss	2000-12-07	13210	315887755	2	1	1

3) Admins table

```
CREATE TABLE admins (
    admin_login_id VARCHAR(20) NOT NULL,
    employee_id int not null,
    employee_fname varchar(20) not null,
    employee_lname varchar(20) not null,
    employee_dob date not null,

    CONSTRAINT pk_admins_admin_login_id PRIMARY Key(admin_login_id)
)
```

Insert Syntax:

```
Insert INTO admins
VALUES
('Manager', 101, 'Rutvij', 'Patil', '06/12/1998'),
('Boss', 102, 'Rohan', 'Bhowmick', '07/15/2000')
```

Output:

	admin_login_id	employee_id	employee_fname	employee_lname	employee_dob
1	Boss	102	Rohan	Bhowmick	2000-07-15
2	Manager	101	Rutvij	Patil	1998-06-12

4) Admin Login Details

```
GO
CREATE table admins_login_details
(
    admin_employee_id int NOT null,
    pass varchar(50) not null,
    admin_department_id int not null,
    admin_login_id varchar(20) not null,

)

alter table admins_login_details
    add constraint fk_admins_login_details_admin_department_id foreign KEY (admin_department_id)
        references department_lookup(department_id)


alter table admins_login_details
    add constraint fk_admins_login_details_admin_login_id foreign key (admin_login_id)
        references admins(admin_login_id)

Go
```

Insert:

```
insert INTO admins_login_details values
(101,1001,1,'Boss'),
(102,1002,2,'Manager')
```

Output:

	admin_employee_id	pass	admin_department_id	admin_login_id
1	101	1001	1	Boss
2	102	1002	2	Manager

5) Pending Requests

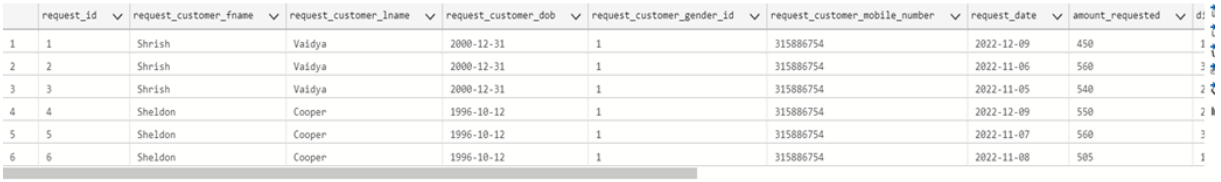
```
Go
CREATE TABLE pending_requests (
    request_id INT IDENTITY NOT NULL,
    request_customer_fname VARCHAR(20) NOT NULL,
    request_customer_lname VARCHAR(20) NOT NULL,
    request_customer_dob DATE NOT NULL,
    request_customer_gender_id int NOT NULL,
    request_customer_mobile_number varchar(10) NOT NULL,
    request_date DATE NOT NULL,
    amount_requested INT not NULL,
    disease_id INT not NULL,
    request_customer_login_id VARCHAR(20) NOT null,
    customer_request_type varchar(20),
    amount_approved int,
    approval_id int,
    rejected_reason varchar(100),
    request_reviewed_by varchar(20),


    CONSTRAINT pk_requests_request_id PRIMARY Key(request_id),
    constraint fk_pending_requests_disease_id foreign key (disease_id) references disease(disease_id),
    CONSTRAINT fk_pending_requests_request_customer_gender_id foreign key(request_customer_gender_id) references genders_lookup(gender_id),
    CONSTRAINT fk_pending_requests_request_customer_login_id FOREIGN key(request_customer_login_id) references customer_login_details(customer_login_id),
    CONSTRAINT fk_pending_requests_reviewed__by foreign key(request_reviewed_by) references admins(admin_login_id),
    constraint fk_pending_requests_approval_request_id foreign key(approval_id) references approval_status(approval_status_id)
)
```

Insert:

```
insert into pending_requests
VALUES('Shrish', 'Vaidya', '12-31-2000', 1, 315886754, '12/09/2022', 450, 1, 'Shrish', 'Non-Premium', Null, Null, Null, NULL),
('Shrish', 'Vaidya', '12-31-2000', 1, 315886754, '11/06/2022', 560, 3, 'Shrish', 'Non-Premium', Null, Null, Null, NULL),
('Shrish', 'Vaidya', '12-31-2000', 1, 315886754, '11/05/2022', 540, 2, 'Shrish', 'Non-Premium', Null, Null, Null, NULL),
('Sheldon', 'Cooper', '10-12-1996', 1, 315886754, '12/09/2022', 550, 2, 'Coop', 'Premium', NULL, Null, Null, NULL),
('Sheldon', 'Cooper', '10-12-1996', 1, 315886754, '11/07/2022', 560, 3, 'Coop', 'Premium', NULL, Null, Null, NULL),
('Sheldon', 'Cooper', '10-12-1996', 1, 315886754, '11/08/2022', 505, 1, 'Coop', 'Premium', NULL, Null, Null, NULL)
```

Output:



	request_id	request_customer_fname	request_customer_lname	request_customer_dob	request_customer_gender_id	request_customer_mobile_number	request_date	amount_requested	d
1	1	Shrish	Vaidya	2000-12-31	1	315886754	2022-12-09	450	1
2	2	Shrish	Vaidya	2000-12-31	1	315886754	2022-11-06	560	3
3	3	Shrish	Vaidya	2000-12-31	1	315886754	2022-11-05	540	2
4	4	Sheldon	Cooper	1996-10-12	1	315886754	2022-12-09	550	2
5	5	Sheldon	Cooper	1996-10-12	1	315886754	2022-11-07	560	3
6	6	Sheldon	Cooper	1996-10-12	1	315886754	2022-11-08	505	1

6) Approved Request

```
CREATE TABLE approved_requests
(
    approved_request_id INT NOT NULL,
    customer_firstname VARCHAR(50) NOT NULL,
    customer_last_name VARCHAR(50) NOT NULL,
    customer_dob date not null,
    customer_gender_id int not null,
    customer_mobile_number varchar(10) not null,
    requested_amount int not NULL,
    amount_approved int not null,
    approved_request_by varchar(20) not null,
    customer_login_id varchar(20) not null,

    CONSTRAINT pk_approved_request_request_id primary key (approved_request_id)
);

alter table approved_requests
    add CONSTRAINT fk_approved_requests_approved_request_by foreign key (approved_request_by)
        references admins(admin_login_id)


alter table approved_requests
    add CONSTRAINT fk_approved_requests_customer_gender_id foreign key (customer_gender_id)
        REFERENCES genders_lookup(gender_id)

alter table approved_requests
    add CONSTRAINT fk_approved_requests_customer_login_id foreign key (customer_login_id)
        REFERENCES customer_login_details(customer_login_id)
```

Insert:

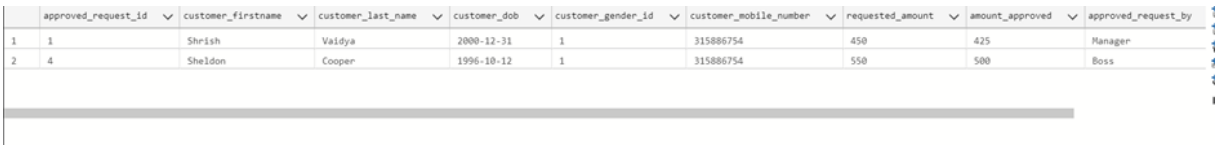
Used Trigger to insert values

```
--For insertion off approved and declined request we use the trigger that will add into approve and disaprove request according to what the admin does on the front end
update pending_requests set amount_approved = 425, approval_id = 1, request_reviewed_by = 'Manager' where request_customer_login_id = 'Shrish' and request_id = 1
update pending_requests set approval_id = 2, request_reviewed_by = 'Manager', rejected_reason = 'xyz' where request_customer_login_id = 'Shrish' and request_id = 2
update pending_requests set amount_approved = 500, approval_id = 1, request_reviewed_by = 'Boss' where request_customer_login_id = 'Coop' and request_id = 4
update pending_requests set approval_id = 2, request_reviewed_by = 'Boss', rejected_reason = 'abc' where request_customer_login_id = 'Coop' and request_id = 5
```

The first and third update statement will set approval_status to 1 which means the request will be added into the approved_requests.

approved_requests before update:

approved_requests after update:



	approved_request_id	customer_firstname	customer_last_name	customer_dob	customer_gender_id	customer_mobile_number	requested_amount	amount_approved	approved_request_by
1	1	Shrish	Vaidya	2000-12-31	1	315886754	450	425	Manager
2	4	Sheldon	Cooper	1996-10-12	1	315886754	550	500	Boss

7) Decline Requests

```
CREATE TABLE decline_requests
    (
        decline_request_id int not null,
        customer_firstname varchar(50) not null,
        customer_lastname varchar(50) not null,
        customer_dob date not null,
        customer_gender_id int not null,
        customer_mobile_number VARCHAR(10) not null,
        requested_amount int not null,
        decline_request_by VARCHAR(20) not null,
        decline_reason varchar(50),
    );
alter table decline_requests
    add CONSTRAINT fk_decline_requests_decline_request_by foreign key (decline_request_by)
        references admins(admin_login_id)


alter table decline_requests
    add constraint fk_decline_requests_customer_gender_id foreign key (customer_gender_id)
        references genders_lookup(gender_id)
```

Insert:

```
--For insertion off approved and declined request we use the trigger that will add into approve and disaprove request according to what the admin does on the front end
update pending_requests set amount_approved = 425, approval_id = 1, request_reviewed_by = 'Manager' where request_customer_login_id = 'Shrish' and request_id = 1
update pending_requests set approval_id = 2, request_reviewed_by = 'Manager', rejected_reason = 'xyz' where request_customer_login_id = 'Shrish' and request_id = 2
update pending_requests set amount_approved = 500, approval_id = 1, request_reviewed_by = 'Boss' where request_customer_login_id = 'Coop' and request_id = 4
update pending_requests set approval_id = 2, request_reviewed_by = 'Boss', rejected_reason = 'abc' where request_customer_login_id = 'Coop' and request_id = 5
```

Second and fourth update query will add the request to decline request as approval_status_id is 2. This will happen when the triggers will get fire after the update statement.

Output:

decline_requests table before the Update statement:

decline_requests table after the update statement:

	decline_request_id	customer_firstname	customer_lastname	customer_dob	customer_gender_id	customer_mobile_number	requested_amount	decline_request_by	decline_reason
1	2	Shrish	Vaidya	2000-12-31	1	315886754	560	Manager	xyz
2	5	Sheldon	Cooper	1996-10-12	1	315886754	560	Boss	abc

TRIGGERS, STORED PROCEDURES, AND USER DEFINED FUNCTIONS:

Triggers:

1) Trigger on new_customer_requests:

```
create trigger t_new_customer_requests_trigger
on new_customer_requests
After UPDATE
AS BEGIN
    declare @approval_status int = (select approval_status_id as approval_status from inserted)
    declare @login varchar(20) =  (select requested_login_id as login from inserted)
    declare @pass varchar(10) = (select mobile_number as pass from inserted)
    declare @department int =  (select department_id as department from inserted)
    if @approval_status = 2
    Begin
        Delete from new_customer_requests where requested_login_id = @login
    END
    if @department = 1 and @approval_status = 1
    Begin
        Insert into customer_login_details VALUES (@login, @pass)
        insert into family(family_lastname,family_customer_login_id,number_of_family_members)
        select inserted.customer_lastname, inserted.requested_login_id, inserted.family_members from inserted
        declare @family_id int = (select family_id from family where family_customer_login_id = @login)
        Insert into premiumcustomers(customer_fname,customer_lname,customer_dob,customer_zip_code,customer_mobile_number,customer_policy_type_id,customer_policy_sub_type_id,customer_gender_id,customer_email,customer_
        select inserted.customer_firstname,customer_lastname,inserted.customer_dob,inserted.zip_code,inserted.mobile_number,inserted.policy_type_id,inserted.policy_sub_type_id,inserted.customer_gender_id,inserted.cus
        update premiumcustomers set family_id = @family_id where customer_customer_login_id = @login
        Delete from new_customer_requests where requested_login_id = @login
    END
    if @department = 2 and @approval_status = 1
    BEGIN
        Insert into customer_login_details VALUES (@login, @pass)
        Insert into customers(customer_fname,customer_lname,customer_dob,customer_zip_code,customer_mobile_number,customer_policy_type_id,customer_policy_sub_type_id,customer_gender_id,customer_email,customer_state_c
        select inserted.customer_firstname,customer_lastname,inserted.customer_dob,inserted.zip_code,inserted.mobile_number,inserted.policy_type_id,inserted.policy_sub_type_id,inserted.customer_gender_id,inserted.cus
        delete from new_customer_requests where requested_login_id = @login
    END
END
Go
```

This trigger will run after an update statement on the new_customer_requests. When the approval_status is 2 then the request is rejected and then the request is deleted from the new_customers_requests table and if the approval_status is 1 then according to the department_id the customer is added to premiumcustomers table or customers table. If the department_id is 1 then customer is added to premiumcustomers table and if it is 2 then into customers table. Also if the approval_status is 1 then the trigger also inserts into the customer_login_details table so that the login credentials are created.

2) Trigger on pending_requests:

```
create TRIGGER t_requests_trigger
on pending_requests
after UPDATE
as BEGIN
    declare @approval_status int = (select inserted.approval_id as approval_status from inserted)
    declare @customer_loginid varchar(20) = (select inserted.request_customer_login_id as customer_loginid from inserted)
    if @approval_status = 1
    BEGIN
        insert into approved_requests(approved_request_id, customer_firstname, customer_last_name, customer_dob, customer_gender_id,customer_mobile_number, requested_amount,amount_approved, approved_request_by, custo
        select inserted.request_id,inserted.request_customer_fname,inserted.request_customer_lname,inserted.request_customer_dob,inserted.request_customer_gender_id,inserted.request_customer_mobile_number,inserted.am
        delete from pending_requests where request_customer_login_id = @customer_loginid and approval_id = @approval_status
    END
    if @approval_status = 2
    BEGIN
        Insert into decline_requests(decline_request_id,customer_firstname,customer_lastname,customer_dob,customer_gender_id,customer_mobile_number,requested_amount,decline_reason,decline_request_by)
        select inserted.request_id, inserted.request_customer_fname, inserted.request_customer_lname, inserted.request_customer_dob, inserted.request_customer_gender_id, inserted.request_customer_mobile_number, inser
        delete from pending_requests where request_customer_login_id = @customer_loginid and approval_id = @approval_status
    END
END

update pending_requests set amount_approved = 500, approval_id = 2, rejected_reason = 'NA', request_reviewed_by = 'Manager' where request_id = 5

GO
```

This trigger is fired after an update statement is executed. When approval_status is 1 then the request is added into the approved_requests table and then if the approval_status is 2 then the request is added to the decline_requests an then it is deleted from pending request.

Procedures:

1) Procedure to add a new admin:

```
create procedure p_upsert_admin
(
@admin_login_id as varchar(20),
@employee_fname as varchar(20),
@employee_lname as varchar(20),
@employee_dob as date,
@psd as varchar(20),
@employee_dept as int

) AS
BEGIN
    declare @next_employee_id as int
    set @next_employee_id =  (select TOP 1 employee_id from admins order by employee_id DESC)
    set @next_employee_id = dbo.f_number(@next_employee_id)
    if not exists (select * from admins where admin_login_id = @admin_login_id)
    BEGIN
        Insert into admins(admin_login_id,employee_id,employee_dob,employee_fname,employee_lname) values(@admin_login_id,@next_employee_id,@employee_dob,@employee_fname,@employee_lname)
        Insert into admins_login_details(admin_employee_id,admin_department_id,admin_login_id,pass)values(@next_employee_id,@employee_dept,@admin_login_id,@psd)
    END
    ELSE
    BEGIN
        declare @existing_employee_id as int
        set @existing_employee_id = (select employee_id from admins where admin_login_id = @admin_login_id)
        INSERT into admins_login_details(admin_employee_id,admin_department_id,admin_login_id,pass) values(@existing_employee_id,@employee_dept,@admin_login_id,@psd)
    END
END
```

So, here if a new admin has to be added or the old admins needs to be assigned multiple departments then we can use this procedure.

Here if the admin is present in the admins table then it will just insert into admin_login_details table the new department_id and credentials are added for the new department assigned to that employee.

Now, if a new admin has to be added then first all the details are added to the Admins table at first then his credentials are inserted in the admins_login_credentials table.

2) Procedure to add new policies to the policy_types and policy sub_types table:

```
create PROCEDURE p_policy_add
(
@policy_type_name as varchar(20)
) as
Begin
    declare @policy_type_id int
    if not exists ( select * from policy_types where policy_type_name = @policy_type_name)
    Begin
    Insert into policy_types values(@policy_type_name)
    set @policy_type_id = (select policy_type_id from policy_types where policy_type_name = @policy_type_name)
    Insert into policy_subtypes(policy_type_id, policy_subtype_name) values (@policy_type_id, dbo.f_concat(@policy_type_name, 'max'))
    INSERT into policy_subtypes(policy_type_id, policy_subtype_name) values (@policy_type_id, dbo.f_concat(@policy_type_name, 'pro'))
    END
    ELSE
        THROW 500001, 'Error: Duplicate policy name inserted', 1

END
```

This procedure is used to add new policies. This takes new policy name as input. Now further if this policy name already exists in a policy_types table then it will throw an error that “Duplicate policy name inserted”. If the policy name is new then it will first insert it into the policy_types table

and it's two subtypes "policy_name max" and "policy_name pro" are made are added into the policy_subtypes table.

User Defined Functions:

1) Function to concatinate two strings:

```
create function f_concat(
    @policy_type_name as varchar(20),
    @policy_sub_type_name as varchar(20)
)returns varchar(20) AS
BEGIN
    return CONCAT(@policy_type_name,' ',@policy_sub_type_name)
END
```

This function is mainly used in procedure to add policy type to create policy subtypes.

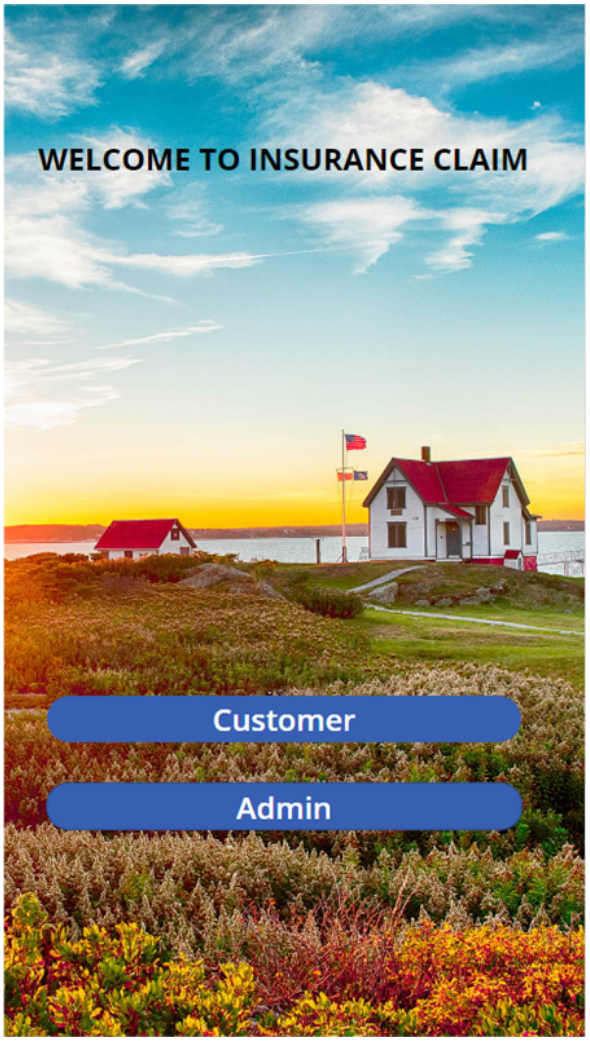
2) Function to find next number:

```
504    create function f_number(
505        @next_num as int
506    )returns int AS
507    BEGIN
508        set @next_num += 1
509        return @next_num
510    END
511
512    GO
```

This function is main;ly used in add admins procedure to find the next employee_id.

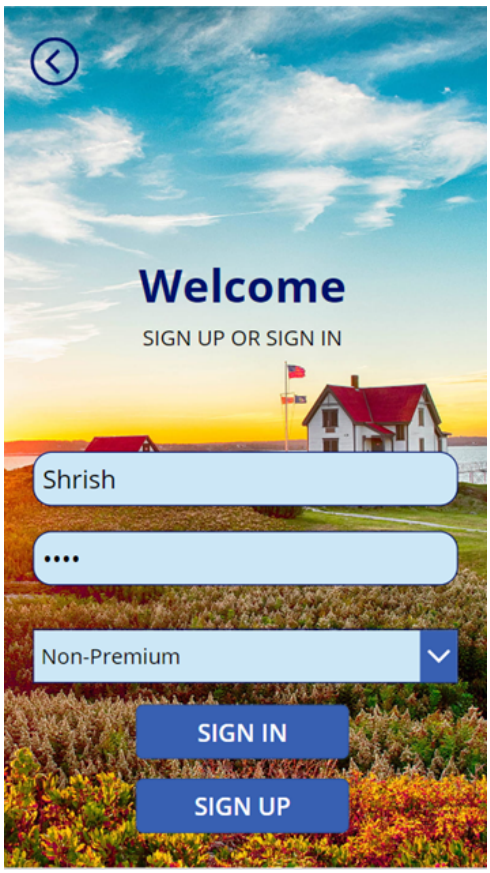
Demo of UI:

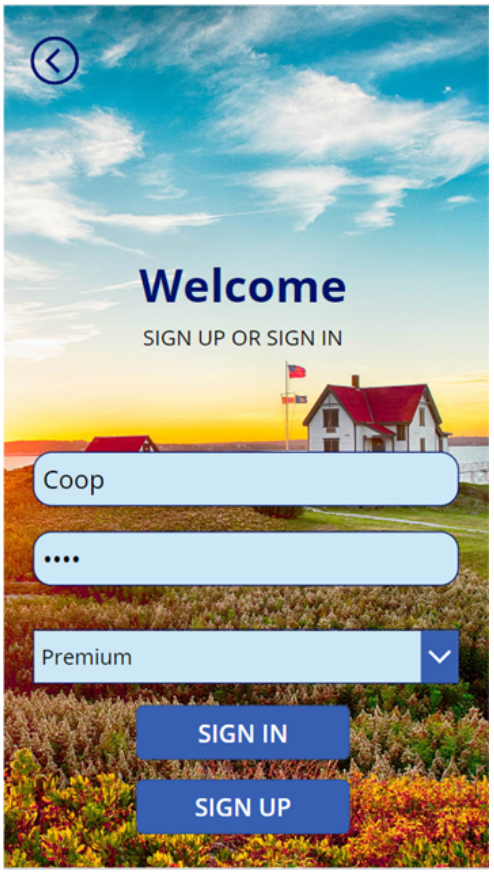
The UI consists of two sides the customer and admin and further it gets divided into two Premium Customer and Non-Premium customers and also the admin they have two departments Premium and Non-Premium.
Starting with the first page:
Main Page:
Description:
On this page you choose if you are a customer or admin:



Let’s say we choose customer at first:
Customer Login:

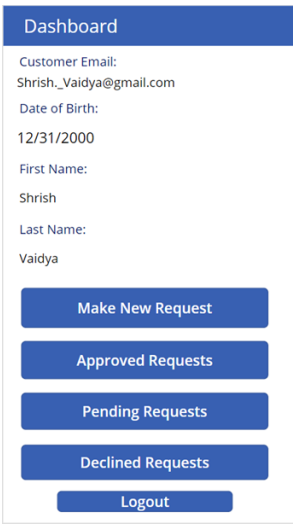
Page 1: Customer Login

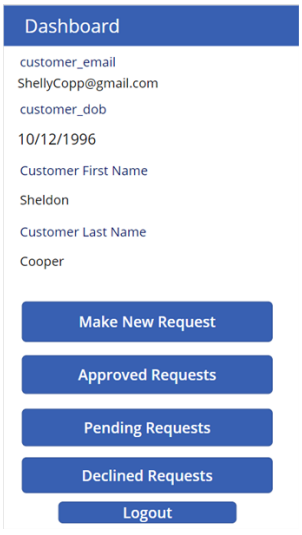




This will check the login and password and the department and then login

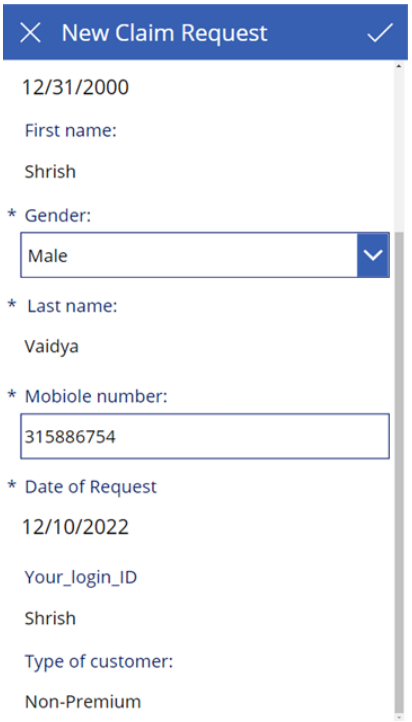
Page 1a: Non-Premium customer dashboard and the Premium customer dashboard





The logout button at the bottom of both the pages will logout the customers from the dashboard page.
From this page the pages for Premium and Non-Premium customers are the same and also the functionality is same so I will go further with Non-Premium customer to explain the functionality.

Page 1c: New Claim Request

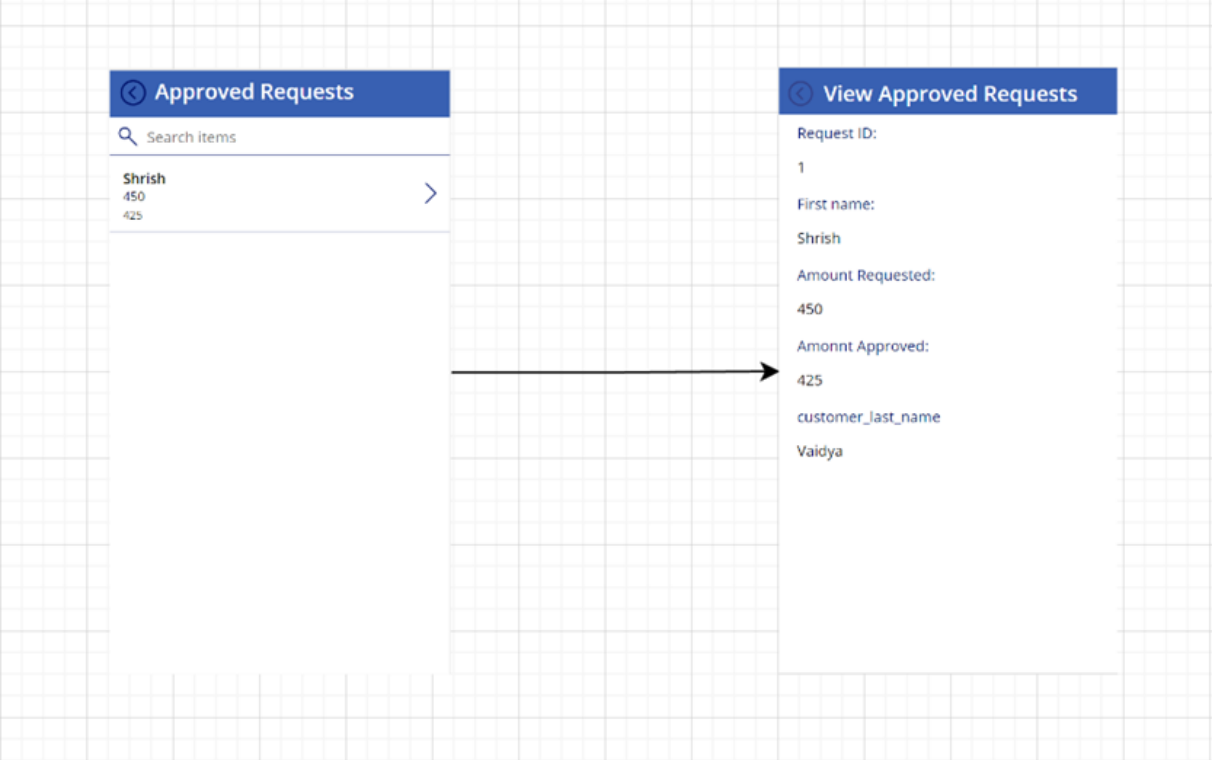


This is how the customer will add a New Claim Request
How the pending requests look after the new claim request:

request_customer_fname	request_customer_lname	request_customer_dob	request_customer_gender_id	request_customer_mobile_number	request_date
Shrish	Vaidya	2000-12-31	1	315886754	2022-11-05
Sheldon	Cooper	1996-10-12	1	315886754	2022-11-08
Shrish	Vaidya	2000-12-31	1	315886754	2022-12-10

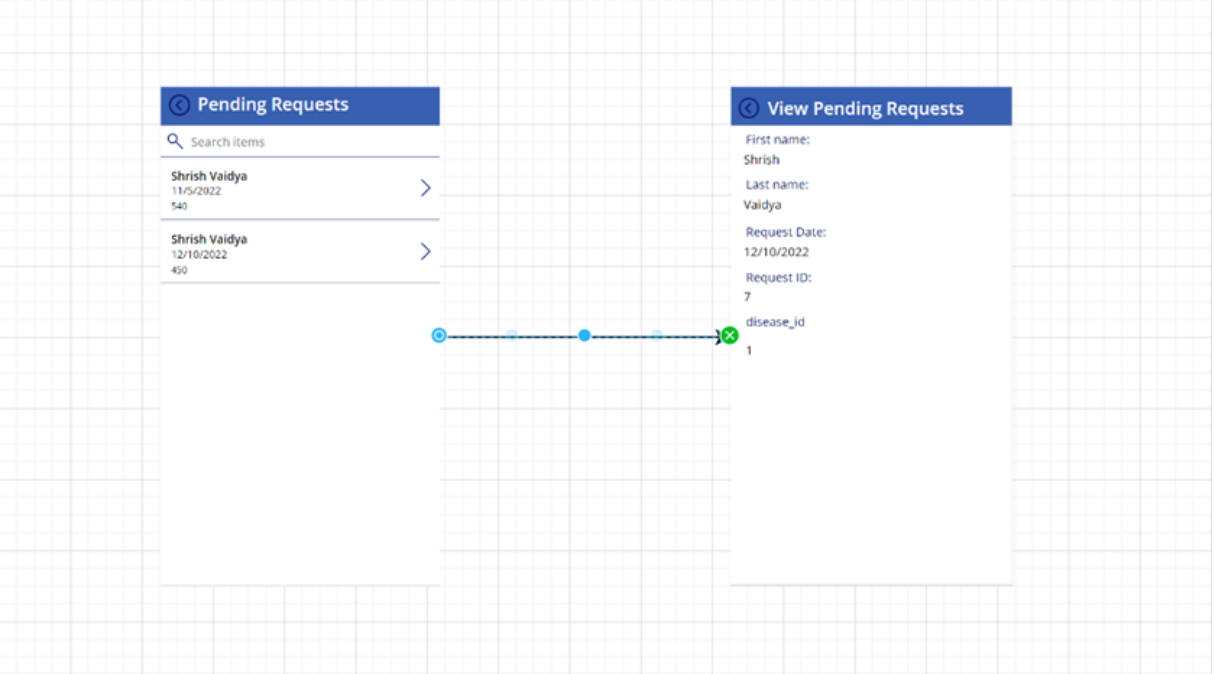
We can see that the last request that we recently added.

Page 1d: Approved Requests page for customers & View_Approved_Requests



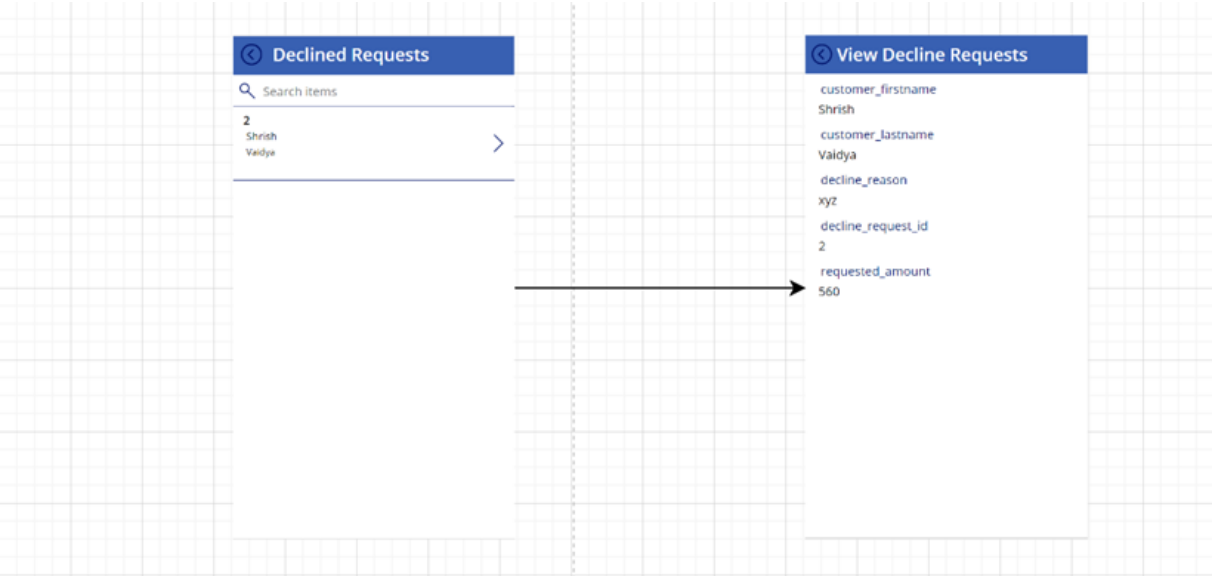
This page can be accessed from customer dashboard.
So, on approved request page you can see the previously approved request and also on next page you can see the description of this approval request.

Page 1e: Pending Requests page for customers & View_Pending_Requests



This page can be accessed from the customer dashboard.
On this page you can find the pending requests made by the customer and also view that request in the next page. Here, you can see the request that I made in the previous new claim requests page.

Page 1f: Declined Requests page for customers & View_Declined_Requests

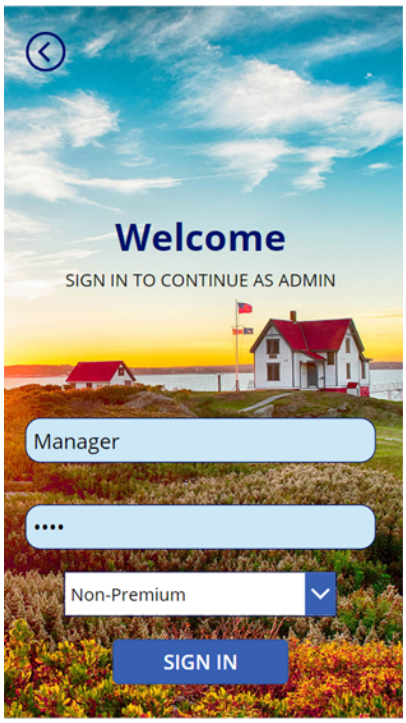


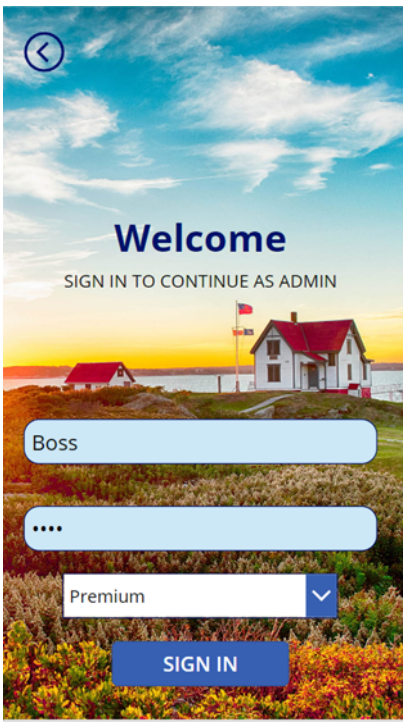
This page is accessed from customer dashboard.
On this page you can view the previously declined requests and also you can view the declined request in next page.

Admin Login:

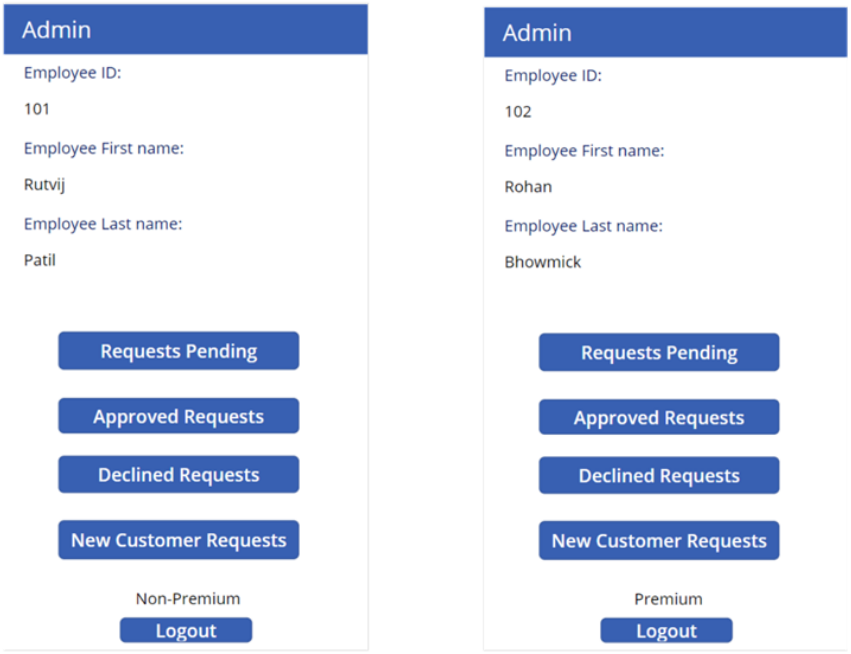
Page 2a: Admin Login

These are the login pages for the admin and here the admin will login according to the department assigned



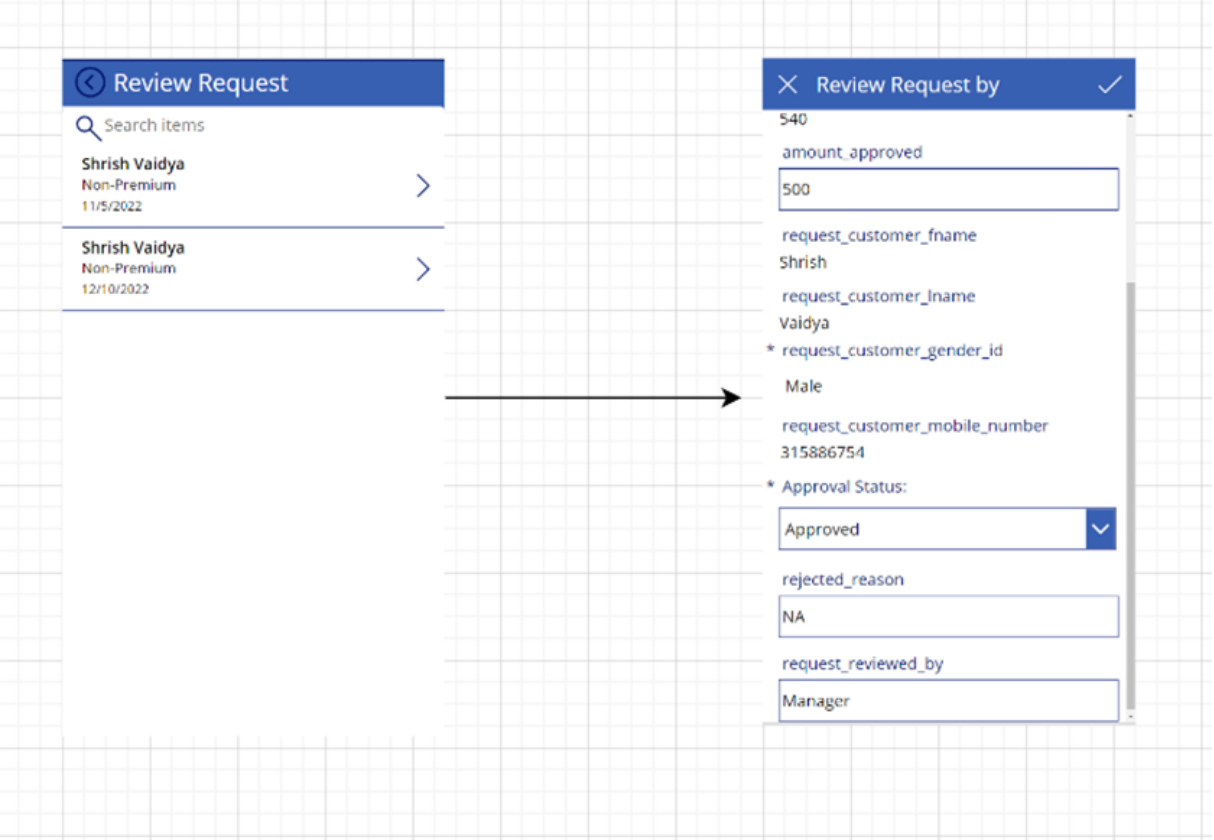


Page 2b: Admin Dashboard



These are the dashboards for the Premium and Non-Premium admins. And the logout button at the bottom is used to logout from the dashboard.
From here the pages are same for both departments only difference between the functionality is that the Premium department employee can only review claim pending requests of premium customers and the non-premium customers will review the non-premium customers claim pending requests. But both department employees can review the new customer requests.
So, from here I will go ahead with the Non-Premium Department employee and explain the functionality of the admin side.

Page 2c: pending Requests page for admin & Review a pending request



So once the admin accepts or declines the request then the trigger runs.

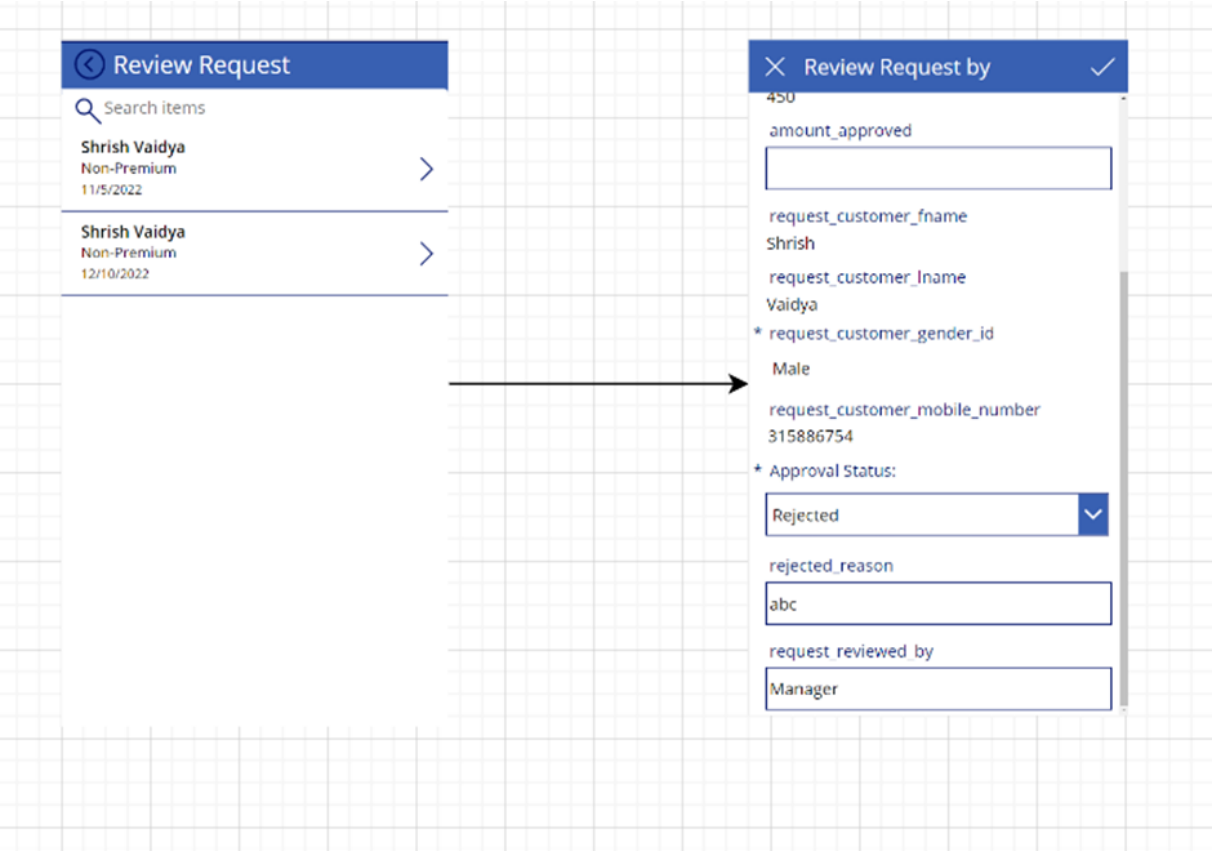
Here, in this example the request was accepted so it generated the following output:

Here you can see the request is deleted from pending and is added to approved requests table.

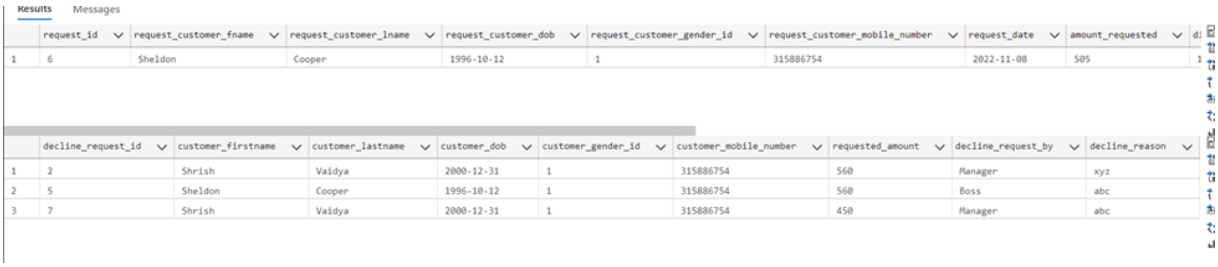
	request_id	request_customer_fname	request_customer_lname	request_customer_dob	request_customer_gender_id	request_customer_mobile_number
1	6	Sheldon	Cooper	1996-10-12	1	315886754
2	7	Shrish	Vaidya	2000-12-31	1	315886754

customer_firstname	customer_last_name	customer_dob	customer_gender_id	customer_mobile_number	requested_amount	amount_approved	appro
Shrish	Vaidya	2000-12-31	1	315886754	450	425	Mana
Shrish	Vaidya	2000-12-31	1	315886754	540	500	Mana
Sheldon	Cooper	1996-10-12	1	315886754	550	500	Boss

Now for another example I will decline a request.



So, again the trigger gets activated.
This will give the output as below:
The request is deleted from pending request and then it is added into declined requests table.

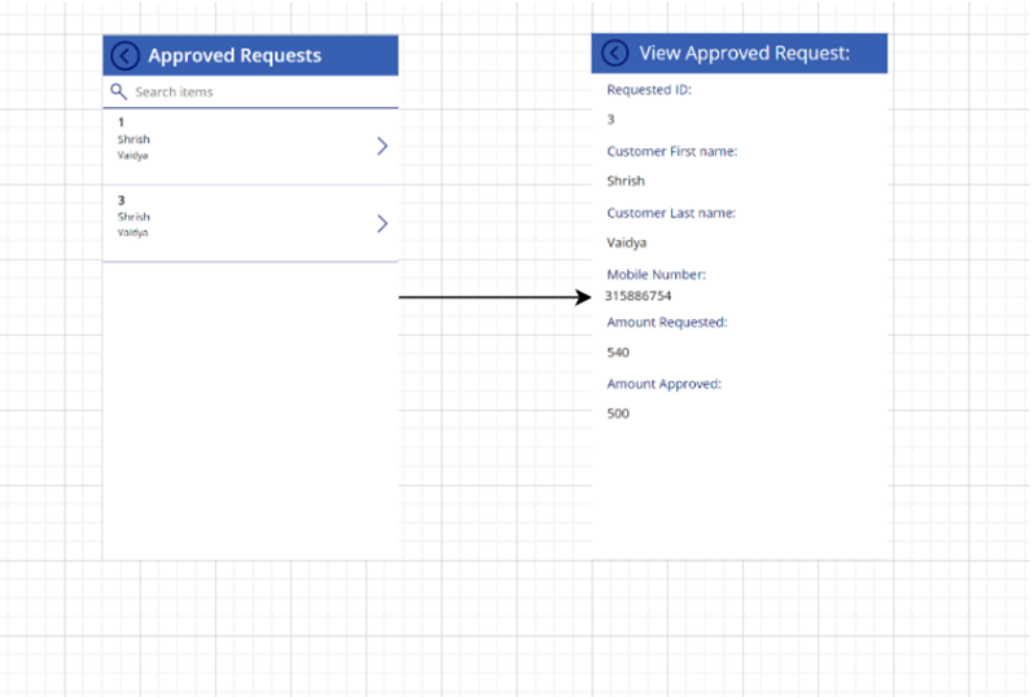


	request_id	request_customer_fname	request_customer_lname	request_customer_dob	request_customer_gender_id	request_customer_mobile_number	request_date	amount_requested
1	6	Sheldon	Cooper	1996-10-12	1	315886754	2022-11-08	505

	decline_request_id	customer_firstname	customer_lastname	customer_dob	customer_gender_id	customer_mobile_number	requested_amount	decline_request_by	decline_reason
1	2	Shrish	Vaidya	2000-12-31	1	315886754	560	Manager	xyz
2	5	Sheldon	Cooper	1996-10-12	1	315886754	560	Boss	abc
3	7	Shrish	Vaidya	2000-12-31	1	315886754	450	Manager	abc

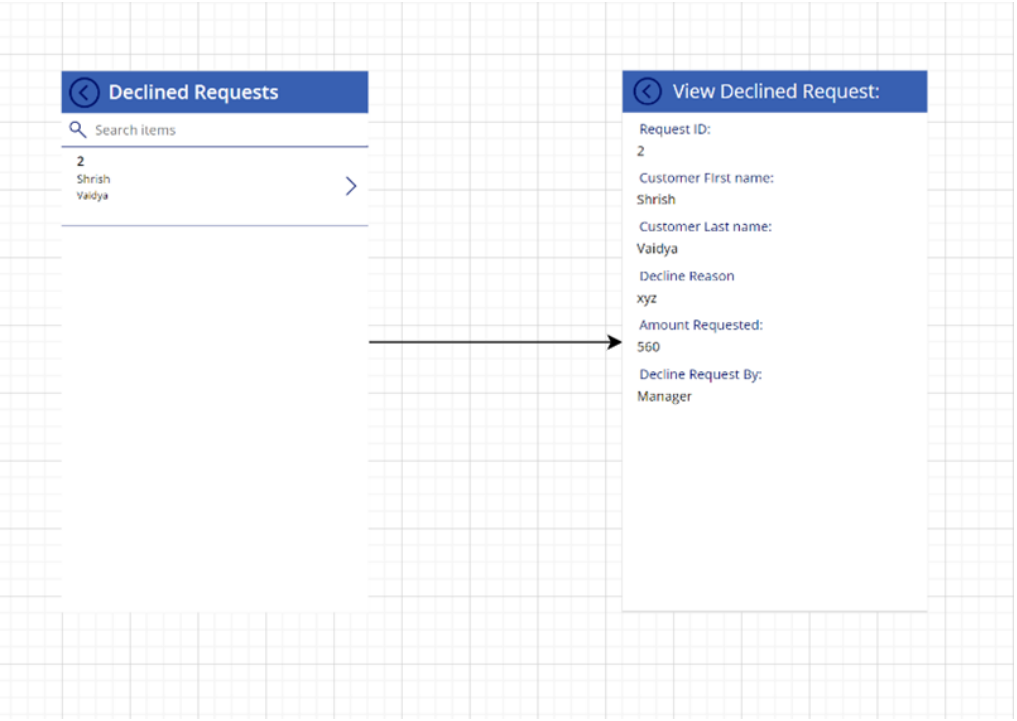
NOTE: TRIGGERS ARE NOT WORKING TJORUGH THE POWER APPS SO I WENT TO THE BACKEND TO EXECUTE THEM. THIS IS JUST AN EXAMPLE TO SHOWCASE HOW THE APP SHOULD WORK.

Page 2d: Approved Requests for employees & View Approved Requests for Employees



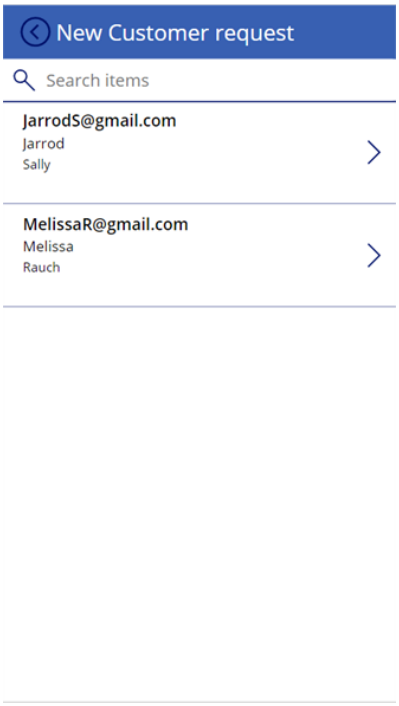
These pages are accessible from the admin dashboard. An admin can see how many requests he have approved till now and also, he can visit details of those details on further page.

Page 2e: Declined Requests for employees & View Declined Requests for Employees



These pages are accessible through the Admin Dashboard. An admin can anytime visit and see the requests that he denied earlier and also, he can view the details of it on next page.

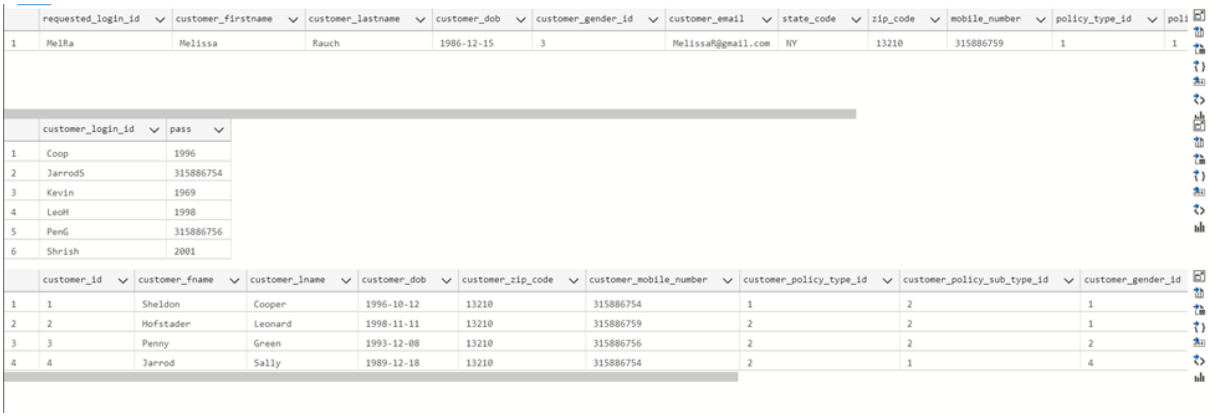
Page 2f: New customer Request Admin



Here on this page the admin can view many requests that were made by new customers to join or take up the policies.

Page 2g: View New Customer request

Here on these pages the customer request that was made on the customer sign up page are displayed and further now admin can make a decision to accept it or decline it.
Let's say admin accepts this request in the above screenshot. Admin always assigns the customer to premium if the family members are more than 1. A trigger gets fired up in the backend after the admin submits and then the trigger creates the login credentials of the customer in customer login details table and then if the customer request is has premium in it then the customer is added to premium customers table or else it gets added into the customers table.
This will give output as below:



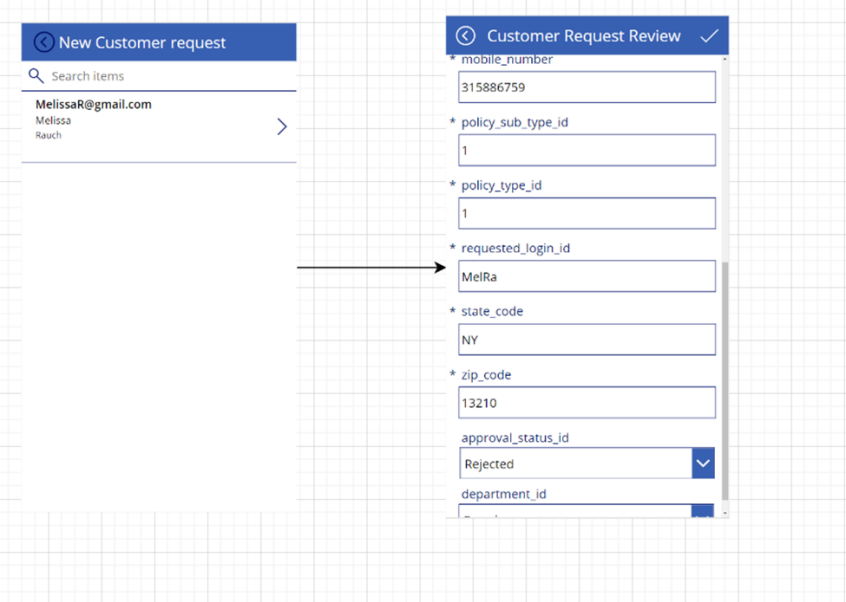
	requested_login_id	customer_firstname	customer_lastname	customer_dob	customer_gender_id	customer_email	state_code	zip_code	mobile_number	policy_type_id	poli
1	MelRa	Melissa	Rauch	1986-12-15	3	MelissaR@gmail.com	NY	13210	315886759	1	1

	customer_login_id	pass
1	Coop	1996
2	JarrodS	315886754
3	Kevin	1969
4	LeoH	1998
5	PenG	315886756
6	Shrish	2001

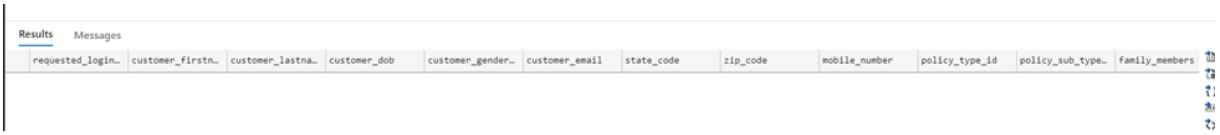
	customer_id	customer_fname	customer_lname	customer_dob	customer_zip_code	customer_mobile_number	customer_policy_type_id	customer_policy_sub_type_id	customer_gender_id
1	1	Sheldon	Cooper	1996-10-12	13210	315886754	1	2	1
2	2	Hofstader	Leonard	1998-11-11	13210	315886759	2	2	1
3	3	Penny	Green	1993-12-08	13210	315886756	2	2	2
4	4	Jarrod	Sally	1989-12-18	13210	315886754	2	1	4

Here you can see the customer request in removed from the new customers requests table and then the login credentials are created in customer login details and finally the customer is added in premium customers table.

Now, for next example I will decline the requests:
Whenever a request is declined then the request is deleted from the new customer requests.
I will decline this request:

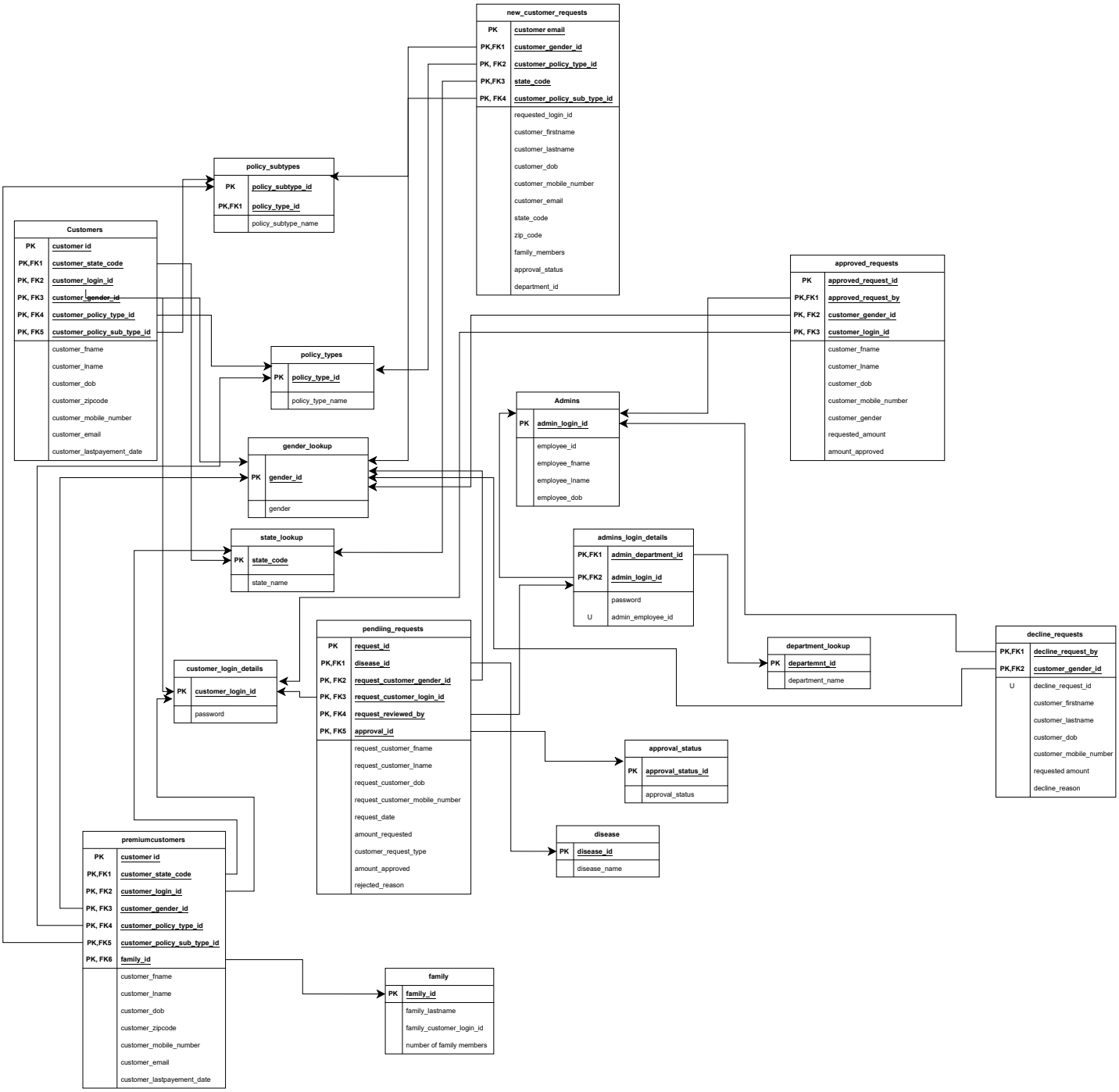


Output after the request was rejected:



Here you can see that after the request was rejected the trigger it deleted the request from the new customer request table.

NOTE: TRIGGERS ARE NOT WORKING TJORUGH THE POWER APPS SO I WENT TO THE BACKEND TO EXECUTE THEM. THIS IS JUST AN EXAMPLE TO SHOWCASE HOW THE APP SHOULD WORK.


new_customer_requests
PK customer email
PK,FK1 customer_gender_id
PK, FK2 customer_policy_type_id
PK,FK3 state_code
PK, FK4 customer_policy_sub_type_id
requested_login_id
customer_firstname
customer_lastname
customer_dob
customer_mobile_number
customer_email
state_code
zip_code
family_members
approval_status
department_id
policy_subtypes
PK policy_subtype_id
PK,FK1 policy_type_id
policy_subtype_name
Customers
PK customer id
PK,FK1 customer_state_code
PK, FK2 customer_login_id
PK, FK3 customer_gender_id
PK, FK4 customer_policy_type_id
PK, FK5 customer_policy_sub_type_id
customer_fname
customer_lname
customer_dob
customer_zipcode
customer_mobile_number
customer_email
customer_lastpayement_date
approved_requests
PK approved_request_id
PK,FK1 approved_request_by
PK, FK2 customer_gender_id
PK, FK3 customer_login_id
customer_fname
customer_lname
customer_dob
customer_mobile_number
customer_gender
requested_amount
amount_approved
policy_types
PK policy_type_id
policy_type_name
Admins
PK admin_login_id
employee_id
employee_fname
employee_lname
employee_dob
gender_lookup
PK gender_id
gender
state_lookup
PK state_code
state_name
admins_login_details
PK,FK1 admin_department_id
PK,FK2 admin_login_id
password
U admin_employee_id
pendiing_requests
PK request_id
PK,FK1 disease_id
PK, FK2 request_customer_gender_id
PK, FK3 request_customer_login_id
PK, FK4 request_reviewed_by
PK, FK5 approval_id
request_customer_fname
request_customer_lname
request_customer_dob
request_customer_mobile_number
request_date
amount_requested
customer_request_type
amount_approved
rejected_reason
department_lookup
PK departemnt_id
department_name
decline_requests
PK,FK1 decline_request_by
PK,FK2 customer_gender_id
U decline_request_id
customer_firstname
customer_lastname
customer_dob
customer_mobile_number
requested amount
decline_reason
customer_login_details
PK customer_login_id
password
approval_status
PK approval_status_id
approval_status
disease
PK disease_id
disease_name
premiumcustomers
PK customer id
PK,FK1 customer_state_code
PK, FK2 customer_login_id
PK, FK3 customer_gender_id
PK, FK4 customer_policy_type_id
PK,FK5 customer_policy_sub_type_id
PK, FK6 family_id
customer_fname
customer_lname
customer_dob
customer_zipcode
customer_mobile_number
customer_email
customer_lastpayement_date
family
PK family_id
family_lastname
family_customer_login_id
number of family members