

INFS 740 Project Report

Project Members:

1. Name : Sanjana Achan
Gno. : G01378624
2. Name : Pradeep Ranjan
Gno. : G01373552

Project Description:

The Insurance Management System (IMS) is a robust web application for insurance providers. Using MongoDB for secure data storage and Spring Boot for seamless integration, IMS manages customers, policies, payments, claims, and agents. It supports CRUD operations and advanced search queries, ensuring real-time data accuracy. The system's strength lies in its intuitive Angular-based front end, which facilitates easy information manipulation. Dynamic charts powered by Chart.js enhance data visualization, aiding strategic decision-making. IMS strikes a balance between complexity and user-friendliness, offering insurance professionals a comprehensive and straightforward solution.

Technologies Used:

- Front-end: developed using HTML, CSS, JavaScript, and Chart.js within the Angular framework
- Back-end: Spring Boot with Java
- Database: MongoDB, a NoSQL database

Basic Queries:

- View, Add, Update, Delete customers
- View, Add, Update, Delete policies
- View, Add, Update, Delete payments
- View, Add, Update, Delete claims
- View, Add, Update, Delete agents

Collections:

1. Customer: Contains records such as CustomerID, Name, Contact Information.
2. Policies: Stores PolicyID, PolicyType, Terms, and Coverage Details.
3. Payment: Tracks PaymentID, CustomerID, Payment Amount, and Date.
4. Claims: Manages ClaimID, Associated PolicyID, Claim Date, and Status Information.
5. Agents: Holds AgentID, Name, Assigned Customers, and Performance Metrics.

1.Customer Collection

An example of customer collection in MongoDB

```
{
  "id": "C01",
  "firstName": "John",
  "lastName": "Doe",
  "email": "johndoe@example.com",
  "dateOfBirth": "1990-05-15T00:00:00.000+00:00",
  "phoneNumber": "+1234567890",
  "createDate": "1990-11-22T10:30:00",
  "gender": "Male",
  "age": 33
}
```

1.1 Customer data displayed on the front end

Customers								
Customer ID	First Name	Last Name	Age	Gender	Email	Date of Birth	Phone Number	Actions
C01	John	Doe	33	Male	johndoe@example.com	1990-05-15T00:00:00.000+00:00	+1234567890	<button>Edit</button> <button>Delete</button>
C02	Alice	Smith	38	Female	alice.smith@example.com	1985-08-10T00:00:00.000+00:00	+9876543210	<button>Edit</button> <button>Delete</button>
C03	Bob	Johnson	28	Male	bob.johnson@example.com	1995-03-25T00:00:00.000+00:00	+1122334455	<button>Edit</button> <button>Delete</button>
C04	Eva	Brown	43	Female	eva.brown@example.com	1980-12-05T00:00:00.000+00:00	+9988776655	<button>Edit</button> <button>Delete</button>
C05	Michael	Williams	45	Male	michael.williams@example.com	1978-09-20T00:00:00.000+00:00	+1122334455	<button>Edit</button> <button>Delete</button>
C06	Sarah	Davis	31	Female	sarah.davis@example.com	1992-04-30T00:00:00.000+00:00	+9988776655	<button>Edit</button> <button>Delete</button>
C07	David	Martinez	36	Male	david.martinez@example.com	1987-07-15T00:00:00.000+00:00	+1234567890	<button>Edit</button> <button>Delete</button>
C08	Emily	Johnson	33	Female	emily.johnson@example.com	1990-01-18T00:00:00.000+00:00	+1122334455	<button>Edit</button> <button>Delete</button>
C09	Daniel	Lee	40	Male	daniel.lee@example.com	1983-11-12T00:00:00.000+00:00	+9988776655	<button>Edit</button> <button>Delete</button>
C10	Olivia	Anderson	34	Female	olivia.anderson@example.com	1989-06-28T00:00:00.000+00:00	+1234567890	<button>Edit</button> <button>Delete</button>
C11	James	Wilson	25	Male	james.wilson@example.com	1998-02-08T00:00:00.000+00:00	+1122334455	<button>Edit</button> <button>Delete</button>
C12	Sophia	Garcia	30	Female	sophia.garcia@example.com	1993-07-22T00:00:00.000+00:00	+9988776655	<button>Edit</button> <button>Delete</button>
C13	William	Smith	41	Male	william.smith@example.com	1982-10-03T00:00:00.000+00:00	+1234567890	<button>Edit</button> <button>Delete</button>
C14	Mia	Johnson	29	Female	mia.johnson@example.com	1994-05-12T00:00:00.000+00:00	+1122334455	<button>Edit</button> <button>Delete</button>
C15	Matthew	Davis	37	Male	matthew.davis@example.com	1986-09-09T00:00:00.000+00:00	+9988776655	<button>Edit</button> <button>Delete</button>
C16	Bobby	Lord	56	Male	bobby@lord.com	1988-10-17T00:00:00.000+00:00	12223212	<button>Edit</button> <button>Delete</button>
C00	Bob	Geer	23	Male	bob@gmail.com	2000-07-20T00:00:00.000+00:00	1234567898	<button>Edit</button> <button>Delete</button>
<div>Create</div>								

1.2 Creating new customers:

Create

Customer ID: COO

First Name: Bob

Last Name: Geer

Age: 23

Gender: Male

Email: bob@gmail.com

Date of Birth: 06/21/2000

Phone Number: 1234567898

Create

Cancel

1.3 Update existing customer details (update Bob's last name to Alle):

Edit Customer

Customer ID: COO

First Name: Bob

Last Name: Alle

Age: 23

Gender: Male

Email: bob@gmail.com

Date of Birth: mm/dd/yyyy

Phone Number: 1234567898

Update

Delete

Cancel

1.4 Deleting an existing customer (deleting customer bob):

Customers								
Customer ID	First Name	Last Name	Age	Gender	Email	Date of Birth	Phone Number	Actions
C01	John	Doe	33	Male	john.doe@example.com	1990-05-15T00:00:00.000+00:00	+1234567890	<div>EditDelete</div>
C02	Alice	Smith	38	Female	alice.smith@example.com	1985-08-10T00:00:00.000+00:00	+9876543210	<div>EditDelete</div>
C03	Bob	Johnson	28	Male	bob.johnson@example.com	1995-03-25T00:00:00.000+00:00	+1122334455	<div>EditDelete</div>
C04	Eva	Brown	43	Female	eva.brown@example.com	1980-12-05T00:00:00.000+00:00	+9988776655	<div>EditDelete</div>
C05	Michael	Williams	45	Male	michael.williams@example.com	1978-09-20T00:00:00.000+00:00	+1122334455	<div>EditDelete</div>
C06	Sarah	Davis	31	Female	sarah.davis@example.com	1992-04-30T00:00:00.000+00:00	+9988776655	<div>EditDelete</div>
C07	David	Martinez	36	Male	david.martinez@example.com	1987-07-15T00:00:00.000+00:00	+1234567890	<div>EditDelete</div>
C08	Emily	Johnson	33	Female	emily.johnson@example.com	1990-01-18T00:00:00.000+00:00	+1122334455	<div>EditDelete</div>
C09	Daniel	Lee	40	Male	daniel.lee@example.com	1983-11-12T00:00:00.000+00:00	+9988776655	<div>EditDelete</div>
C10	Olivia	Anderson	34	Female	olivia.anderson@example.com	1989-06-28T00:00:00.000+00:00	+1234567890	<div>EditDelete</div>
C11	James	Wilson	25	Male	james.wilson@example.com	1998-02-08T00:00:00.000+00:00	+1122334455	<div>EditDelete</div>
C12	Sophia	Garcia	30	Female	sophia.garcia@example.com	1993-07-22T00:00:00.000+00:00	+9988776655	<div>EditDelete</div>
C13	William	Smith	41	Male	william.smith@example.com	1982-10-03T00:00:00.000+00:00	+1234567890	<div>EditDelete</div>
C14	Mia	Johnson	29	Female	mia.johnson@example.com	1994-05-12T00:00:00.000+00:00	+1122334455	<div>EditDelete</div>
C15	Matthew	Davis	37	Male	matthew.davis@example.com	1986-09-09T00:00:00.000+00:00	+9988776655	<div>EditDelete</div>
C16	Bobby	Lord	56	Male	bobby@lord.com	1988-10-17T00:00:00.000+00:00	12223212	<div>EditDelete</div>

2. Policy collection:

An example of policy collection in MongoDB

```
{
  "id": "P01",
  "customerId": "C01",
  "agentId": "A01",
  "policyNumber": "P001",
  "policyAmount": 50000,
  "policyType": "auto",
  "startDate": "2023-01-15T00:00:00.000+00:00",
  "endDate": "2024-01-15T00:00:00.000+00:00",
  "active": true
}
```

2.1 Read policy data from the front end:

Policies									
Policy ID	Customer ID	Agent ID	Policy Number	Policy Amount	Policy Type	Start Date	End Date	Active	Actions
P01	C01	A01	P001	50000	auto	1/14/23, 7:00 PM	1/14/24, 7:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P02	C02	A02	P002	75000	life	2/19/23, 7:00 PM	2/19/24, 7:00 PM	No	<button>Edit</button> <button>Delete</button>
P03	C03	A03	P003	60000	home	3/24/23, 8:00 PM	3/24/24, 8:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P04	C04	A04	P004	45000	auto	4/9/23, 8:00 PM	4/9/24, 8:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P05	C05	A05	P005	80000	life	5/4/23, 8:00 PM	5/4/24, 8:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P06	C06	A06	P006	55000	home	6/14/23, 8:00 PM	6/14/24, 8:00 PM	No	<button>Edit</button> <button>Delete</button>
P07	C07	A07	P007	70000	auto	7/19/23, 8:00 PM	7/19/24, 8:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P08	C08	A08	P008	90000	life	8/24/23, 8:00 PM	8/24/24, 8:00 PM	No	<button>Edit</button> <button>Delete</button>
P09	C09	A09	P009	48000	home	9/4/23, 8:00 PM	9/4/24, 8:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P10	C10	A10	P010	72000	auto	10/9/23, 8:00 PM	10/9/24, 8:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P11	C11	A11	P011	55000	life	11/14/23, 7:00 PM	11/14/24, 7:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P12	C12	A12	P012	60000	health	12/19/23, 7:00 PM	12/19/24, 7:00 PM	No	<button>Edit</button> <button>Delete</button>
P13	C13	A13	P013	55000	travel	1/24/24, 7:00 PM	1/24/25, 7:00 PM	No	<button>Edit</button> <button>Delete</button>
P14	C14	A14	P014	70000	property	2/9/24, 7:00 PM	2/9/25, 7:00 PM	Yes	<button>Edit</button> <button>Delete</button>
P15	C15	A15	P015	65000	auto	3/4/24, 7:00 PM	3/4/25, 7:00 PM	No	<button>Edit</button> <button>Delete</button>
P16	C00	A16	30000	20000	auto	11/27/23, 7:00 PM	12/8/23, 7:00 PM	Yes	<button>Edit</button> <button>Delete</button>

2.2 Create a new policy:

Create

Policy ID

P16

Customer ID

C00

Agent ID

A16

Policy Number

30000

Policy Amount

20000

Policy Type

auto

Start Date

11/28/2023

End Date

12/08/2023

Active

☒

Create

Cancel

2.3 Update existing policy (updated P1 : policy type as home from auto):

Edit Policy

Policy ID

P16

Customer ID

C00

Agent ID

A16

Policy Number

30000

Policy Amount

20000

Policy Type

auto

Start Date

11/27/2023

End Date

12/08/2023

Active

☒

Update

Delete

2.4 Delete existing policy (deleted P04):

Policies

Policy ID	Customer ID	Agent ID	Policy Number	Policy Amount	Policy Type	Start Date	End Date	Active	Actions
P01	C01	A01	P001	50000	auto	1/14/23, 7:00 PM	1/14/24, 7:00 PM	Yes	<div>EditDelete</div>
P02	C02	A02	P002	75000	life	2/19/23, 7:00 PM	2/19/24, 7:00 PM	No	<div>EditDelete</div>
P03	C03	A03	P003	60000	home	3/24/23, 8:00 PM	3/24/24, 8:00 PM	Yes	<div>EditDelete</div>
P04	C04	A04	P004	45000	auto	4/9/23, 8:00 PM	4/9/24, 8:00 PM	Yes	<div>EditDelete</div>
P05	C05	A05	P005	80000	life	5/4/23, 8:00 PM	5/4/24, 8:00 PM	Yes	<div>EditDelete</div>
P06	C06	A06	P006	55000	home	6/14/23, 8:00 PM	6/14/24, 8:00 PM	No	<div>EditDelete</div>
P07	C07	A07	P007	70000	auto	7/19/23, 8:00 PM	7/19/24, 8:00 PM	Yes	<div>EditDelete</div>
P08	C08	A08	P008	90000	life	8/24/23, 8:00 PM	8/24/24, 8:00 PM	No	<div>EditDelete</div>
P09	C09	A09	P009	48000	home	9/4/23, 8:00 PM	9/4/24, 8:00 PM	Yes	<div>EditDelete</div>
P10	C10	A10	P010	72000	auto	10/9/23, 8:00 PM	10/9/24, 8:00 PM	Yes	<div>EditDelete</div>
P11	C11	A11	P011	55000	life	11/14/23, 7:00 PM	11/14/24, 7:00 PM	Yes	<div>EditDelete</div>
P12	C12	A12	P012	60000	health	12/19/23, 7:00 PM	12/19/24, 7:00 PM	No	<div>EditDelete</div>
P13	C13	A13	P013	55000	travel	1/24/24, 7:00 PM	1/24/25, 7:00 PM	No	<div>EditDelete</div>
P14	C14	A14	P014	70000	property	2/9/24, 7:00 PM	2/9/25, 7:00 PM	Yes	<div>EditDelete</div>
P15	C15	A15	P015	65000	auto	3/4/24, 7:00 PM	3/4/25, 7:00 PM	No	<div>EditDelete</div>

Create

3. Payment collection

An example of payment collection in MongoDB

```
{
  "id": "PAY1",
  "policyId": "P01",
  "customerId": "C01",
  "paymentDate": "2014-01-23",
  "amountPaid": 1000.0,
  "isLatePayment": "false"
}
```

3.1 Read payment data from the front end:

Payment						
Payment ID	Policy ID	Customer ID	Payment Date	Amount Paid	Is Late Payment	Actions
PAY1	P01	C01	1/23/14, 12:00 AM	1000	false	Edit Delete
PAY2	P02	C02	2/19/23, 12:00 AM	1500	true	Edit Delete
PAY3	P03	C03	3/24/23, 12:00 AM	1200	false	Edit Delete
PAY4	P04	C04	4/9/23, 12:00 AM	900	true	Edit Delete
PAY5	P05	C05	5/4/23, 12:00 AM	1600	false	Edit Delete
PAY6	P06	C06	6/14/23, 12:00 AM	1100	true	Edit Delete
PAY7	P07	C07	7/19/23, 12:00 AM	1400	false	Edit Delete
PAY8	P08	C08	8/24/23, 12:00 AM	1800	true	Edit Delete
PAY9	P09	C09	9/4/23, 12:00 AM	950	false	Edit Delete
PAY10	P10	C10	10/9/23, 12:00 AM	1450	true	Edit Delete
PAY11	P11	C11	11/14/23, 12:00 AM	1100	false	Edit Delete
PAY12	P12	C12	12/19/23, 12:00 AM	1250	true	Edit Delete
PAY13	P13	C13	1/24/24, 12:00 AM	1100	false	Edit Delete
PAY14	P14	C14	2/9/24, 12:00 AM	1400	true	Edit Delete
PAY20	P16	C00	11/9/23, 12:00 AM	200	true	Edit Delete
Create						

3.2 Create a new payment:

3.3 Update existing payment (updated PAY20 : Amount paid is \$500 from \$200):

3.4 Delete of Payment (deleted PAY20):

Payment

Payment ID	Policy ID	Customer ID	Payment Date	Amount Paid	Is Late Payment	Actions
PAY1	P01	C01	1/23/14, 12:00 AM	1000	false	<button>Edit</button> <button>Delete</button>
PAY2	P02	C02	2/19/23, 12:00 AM	1500	true	<button>Edit</button> <button>Delete</button>
PAY3	P03	C03	3/24/23, 12:00 AM	1200	false	<button>Edit</button> <button>Delete</button>
PAY4	P04	C04	4/9/23, 12:00 AM	900	true	<button>Edit</button> <button>Delete</button>
PAY5	P05	C05	5/4/23, 12:00 AM	1600	false	<button>Edit</button> <button>Delete</button>
PAY6	P06	C06	6/14/23, 12:00 AM	1100	true	<button>Edit</button> <button>Delete</button>
PAY7	P07	C07	7/19/23, 12:00 AM	1400	false	<button>Edit</button> <button>Delete</button>
PAY8	P08	C08	8/24/23, 12:00 AM	1800	true	<button>Edit</button> <button>Delete</button>
PAY9	P09	C09	9/4/23, 12:00 AM	950	false	<button>Edit</button> <button>Delete</button>
PAY10	P10	C10	10/9/23, 12:00 AM	1450	true	<button>Edit</button> <button>Delete</button>
PAY11	P11	C11	11/14/23, 12:00 AM	1100	false	<button>Edit</button> <button>Delete</button>
PAY12	P12	C12	12/19/23, 12:00 AM	1250	true	<button>Edit</button> <button>Delete</button>
PAY13	P13	C13	1/24/24, 12:00 AM	1100	false	<button>Edit</button> <button>Delete</button>
PAY14	P14	C14	2/9/24, 12:00 AM	1400	true	<button>Edit</button> <button>Delete</button>

Create

4. Agent collection

An example of Agent collection in MongoDB

```
{
  "id": "A1",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@example.com",
  "gender": "Male",
  "policyIds": [
    "P01",
    "P04",
    "P07"
  ]
}
```

4.1 Read agent data from the front end:

Agents					
Agent Id	First Name	Last Name	Email	Policy IDs	Actions
A1	Serena	Frost	Frost@example.com	P01, P04, P07	Edit Delete
A2	Ryan	Spectre	Spectre@example.com	P02, P06	Edit Delete
A3	Olivia	Nightshade	Nightshade@example.com	P03, P04	Edit Delete
A5	Mike	Johnson	mike.johnson@example.com	P05	Edit Delete
A6	Lisa	Brown	lisa.brown@example.com		Edit Delete
A7	Zoe	Miller	Zoe.miller@example.com	P02	Edit Delete
A8	Jackson	Frost	Frost@example.com	P11, P12, P04	Edit Delete
A10	Pradeep	Ranjan	pradeep.ranjan7837@gmail.com	P11	Edit Delete
A11	Rahul	Alle	pradeep.Alle7837@gmail.com	P01, P11, P12	Edit Delete
A121	Aurora	Steele	Steele@gmail.com	P09, P01	Edit Delete
A22	Sable	Veil	Veil7837@gmail.com	P01	Edit Delete
A26	Kai	Zenith	Zenith7837@gmail.com	P05, P07, P09	Edit Delete
A25	Anuj	Karpten	Anuj@gmail.com		Edit Delete
Create					

4.2 Create a new agent:

Create

Agent ID: A25

First Name: Anuj

Last Name: Karpfen

Email: Anuj@gmail.com

Policy IDs (comma-separated): P16,P09

CreateCancel

4.3 Update the existing agent (update agent ID from A25 to A30):

Edit Agent

Agent ID: A30

First Name: Anuj

Last Name: Karpfen

Email: Anuj@gmail.com

Policy IDs (comma-separated):

UpdateDeleteCancel

4.4 Delete existing agent (A30 has been deleted)

Agents					
Agent Id	First Name	Last Name	Email	Policy IDs	Actions
A1	Serena	Frost	Frost@example.com	P01, P04, P07	<div>EditDelete</div>
A2	Ryan	Spectre	Spectre@example.com	P02, P06	<div>EditDelete</div>
A3	Olivia	Nightshade	Nightshade@example.com	P03, P04	<div>EditDelete</div>
A5	Mike	Johnson	mike.johnson@example.com	P05	<div>EditDelete</div>
A6	Lisa	Brown	lisa.brown@example.com		<div>EditDelete</div>
A7	Zoe	Miller	Zoe.miller@example.com	P02	<div>EditDelete</div>
A8	Jackson	Frost	Frost@example.com	P11, P12, P04	<div>EditDelete</div>
A10	Pradeep	Ranjan	pradeep.ranjan7837@gmail.com	P11	<div>EditDelete</div>
A11	Rahul	Alle	pradeep.Alle7837@gmail.com	P01, P11, P12	<div>EditDelete</div>
A121	Aurora	Steele	Steele@gmail.com	P09, P01	<div>EditDelete</div>
A22	Sable	Veil	Veil7837@gmail.com	P01	<div>EditDelete</div>
A26	Kai	Zenith	Zenith7837@gmail.com	P05, P07, P09	<div>EditDelete</div>
<div>Create</div>					

5. Claims collection:

An example of claim collection in MongoDB

```
{
  "id": "CLM01",
  "policyId": "P01",
  "customerId": "C01",
  "dateOfClaim": "2023-03-05T00:00:00.000+00:00",
  "claimAmount": 2500.0,
  "status": "filed"
}
```

5.1 Read claim data from the front end:

Claims						
Claim ID	Policy ID	Customer ID	Date of Claim	Claim Amount	Status	Actions
CLM01	P01	C01	Mar 4, 2023	2500	paid	<button>Edit</button> <button>Delete</button>
CLM02	P02	C02	May 14, 2023	3000	in_review	<button>Edit</button> <button>Delete</button>
CLM03	P03	C03	Jul 24, 2023	2200	paid	<button>Edit</button> <button>Delete</button>
CLM04	P04	C04	Sep 9, 2023	1800	denied	<button>Edit</button> <button>Delete</button>
CLM05	P05	C05	Nov 4, 2023	3200	filed	<button>Edit</button> <button>Delete</button>
CLM06	P06	C06	Jan 19, 2024	2400	in_review	<button>Edit</button> <button>Delete</button>
CLM07	P07	C07	Mar 14, 2024	2800	paid	<button>Edit</button> <button>Delete</button>
CLM08	P08	C08	May 9, 2024	2100	denied	<button>Edit</button> <button>Delete</button>
CLM09	P09	C09	Jul 4, 2024	2600	filed	<button>Edit</button> <button>Delete</button>
CLM10	P10	C10	Sep 14, 2024	3500	in_review	<button>Edit</button> <button>Delete</button>
CLM11	P11	C11	Nov 19, 2024	2700	paid	<button>Edit</button> <button>Delete</button>
CLM12	P12	C12	Jan 24, 2025	2000	denied	<button>Edit</button> <button>Delete</button>
CLM13	P13	C13	Mar 9, 2025	2900	filed	<button>Edit</button> <button>Delete</button>
CLM14	P11	C05	Nov 22, 2023	40000	paid	<button>Edit</button> <button>Delete</button>
CLM15	P15	C00	Nov 27, 2023	50000	paid	<button>Edit</button> <button>Delete</button>
<div>Create</div>						

5.2 Create a new claim:

Create

Claim ID:CLM15

Policy ID:P15

Customer ID:C00

Date of Claim:11/28/2023

Claim Amount:50000

Status:paid

5.3 Update existing claims (changing the status from paid to denied):

Edit Claim

Claim ID:CLM15

Policy ID:P15

Customer ID:C00

Date of Claim:mm/dd/yyyy

Claim Amount:50000

Status:denied

Update

Delete

Cancel

5.4 Delete an existing claim (CLM15 has been deleted):

Claims						
Claim ID	Policy ID	Customer ID	Date of Claim	Claim Amount	Status	Actions
CLM01	P01	C01	Mar 4, 2023	2500	paid	<button>Edit</button> <button>Delete</button>
CLM02	P02	C02	May 14, 2023	3000	in_review	<button>Edit</button> <button>Delete</button>
CLM03	P03	C03	Jul 24, 2023	2200	paid	<button>Edit</button> <button>Delete</button>
CLM04	P04	C04	Sep 9, 2023	1800	denied	<button>Edit</button> <button>Delete</button>
CLM05	P05	C05	Nov 4, 2023	3200	filed	<button>Edit</button> <button>Delete</button>
CLM06	P06	C06	Jan 19, 2024	2400	in_review	<button>Edit</button> <button>Delete</button>
CLM07	P07	C07	Mar 14, 2024	2800	paid	<button>Edit</button> <button>Delete</button>
CLM08	P08	C08	May 9, 2024	2100	denied	<button>Edit</button> <button>Delete</button>
CLM09	P09	C09	Jul 4, 2024	2600	filed	<button>Edit</button> <button>Delete</button>
CLM10	P10	C10	Sep 14, 2024	3500	in_review	<button>Edit</button> <button>Delete</button>
CLM11	P11	C11	Nov 19, 2024	2700	paid	<button>Edit</button> <button>Delete</button>
CLM12	P12	C12	Jan 24, 2025	2000	denied	<button>Edit</button> <button>Delete</button>
CLM13	P13	C13	Mar 9, 2025	2900	filed	<button>Edit</button> <button>Delete</button>
CLM14	P11	C05	Nov 22, 2023	40000	paid	<button>Edit</button> <button>Delete</button>
<div>Create</div>						

Complex Queries:

Query1: Retrieve customer and policy details for claims exceeding a specific limit.

Collections : three collections: claims, policy, and customer.

Purpose and Logic: Filtering the claims collection for those exceeding a specified threshold amount. It then enriches these claims with additional details by joining data from the policy and customer collections.

The function further processes this data by unwinding the joined details for proper alignment and projecting the necessary fields, resulting in a structured list of high-value claim details.

Output: The final output is a list of objects, each representing a high-value claim with detailed information about the policy and the customer associated with it.

Search Queries

Query1:

Query2:

Query3:

Retrieve customer and policy details for claims exceeding a specified limit.

Choose Claim Amount:

Sno	CustomerFullName	Policy Number	Policy Amount	Policy Type	Claim Status	Claim Amount
1	Olivia Anderson	P010	72000	auto	in_review	3500
2	Michael Williams	P011	55000	life	paid	40000


Query2: Retrieve customer information about late payments associated with specific insurance policy types.

Collections : three collections: payment, policy, and customer.

Purpose and Logic: Its primary goal is to identify and analyze late payments for specific insurance policy types. It achieves this by joining relevant customer and policy details with the payment collection data.

Data Processing: The function processes the data by first unwinding the joined customer and policy details to align them correctly. Then, it applies filtering based on policy type and late payment status and projects key information into a structured format.

Output: The output is a collection of LatePaymentDTO objects, each providing comprehensive details about the customer, their policy, and the specifics of the late payment incident.



- Dashboard
- Customer
- Policies
- Payment
- agent
- Claims
- Search Queries

Query1:

Query2:

Query3:

Retrieve customer and policy data based on a specific insurance policy type, considering whether late payments are present or not.

Policy Type:
auto

Late Payment:
True

Search

Sno	customerID	CustomerFullName	policyId	policyStartDate	policyEndDate	Latepayment	amountPaid
1	C04	Eva Brown	P04	2023-04-10T00:00:00.000+00:00	2024-04-10T00:00:00.000+00:00	true	900
2	C10	Olivia Anderson	P10	2023-10-10T00:00:00.000+00:00	2024-10-10T00:00:00.000+00:00	true	1450

Query3: Retrieve Agent details based on customer gender and policy status.

Collections: This code operates on three collections: "agents," "policy," and "customer."

Purpose and Logic: The code's primary objective is to find agents with active policies and male customers associated with those policies. It does this by performing a series of aggregation operations that involve joining relevant data from these collections.

Data Processing: The code starts by joining "agents" with "policy" and "customer" collections. It then unwinds the joined data, applies filters to select active policies and male customers, and finally projects the necessary fields into a structured format.

Output: The result is a list of "ClaimByAgentDTO" objects, each containing detailed information about agents, policies, and associated male customers meeting the specified criteria.

Search Queries

Query1:

Query2:

Query3:

Retrieve Agent details based on customer gender and policy status.

Gender:

Female

Active: ☒

search

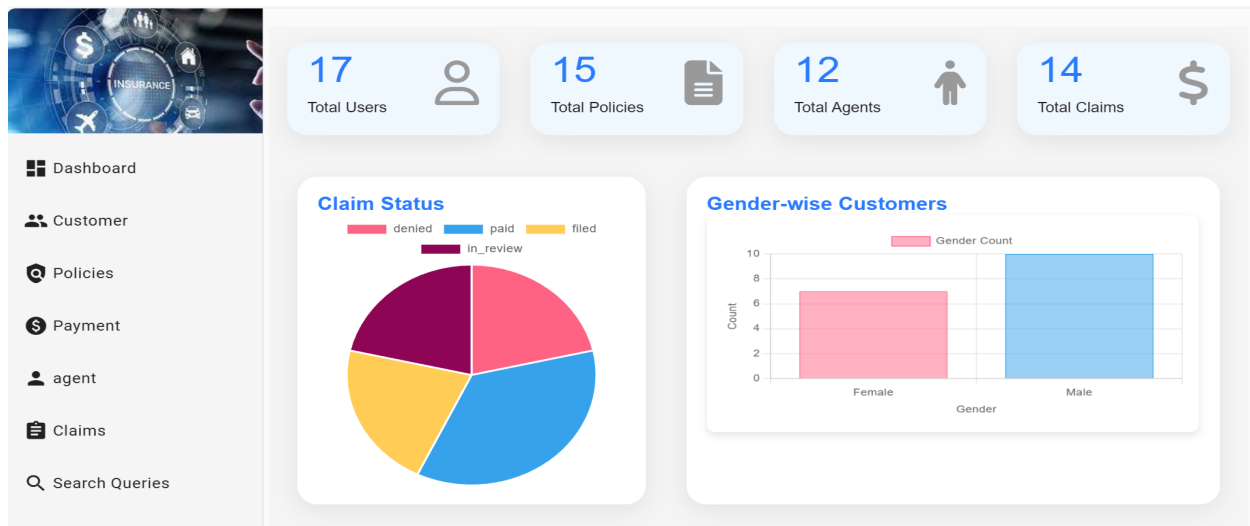
Sno	AgentFullName	AgentID	agentEmail	CustomerFullName	customerGender	policyNumber	policyactive
1	Serena Frost	A1	Frost@example.com	Eva Brown	Female	P004	true
2	Olivia Nightshade	A3	Nightshade@example.com	Eva Brown	Female	P004	true
3	Jackson Frost	A8	Frost@example.com	Eva Brown	Female	P004	true

Visualization:

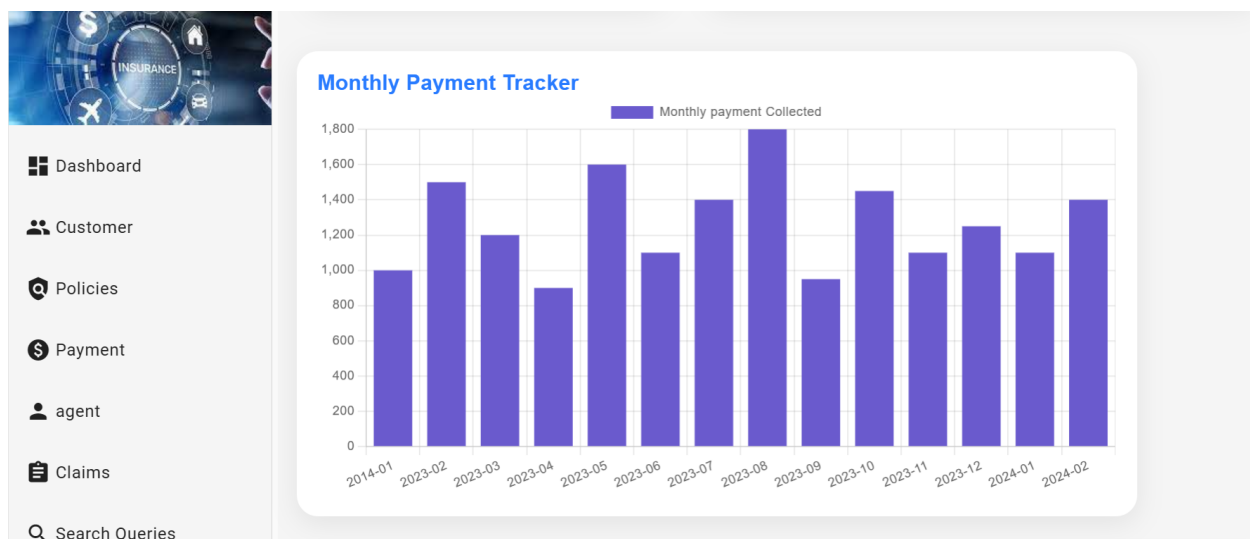
Flash cards: Flash cards display the counts of customers, policies, agents, and claims from their respective collections.

Pie chart (Claim Status): The pie chart visually represents the status of claims, showcasing four states—denied, paid, filed, and in review—in relation to customers.

Bar Chart (Gender-wise Customers): This chart displays the total number of male and female customers separately, utilizing distinct bars for each gender category.



Bar Chart (Monthly Payment tracker) : The second chart calculates and visualizes the total monthly payments, representing the sum of 'amount paid' for each month.



Recent transaction query:

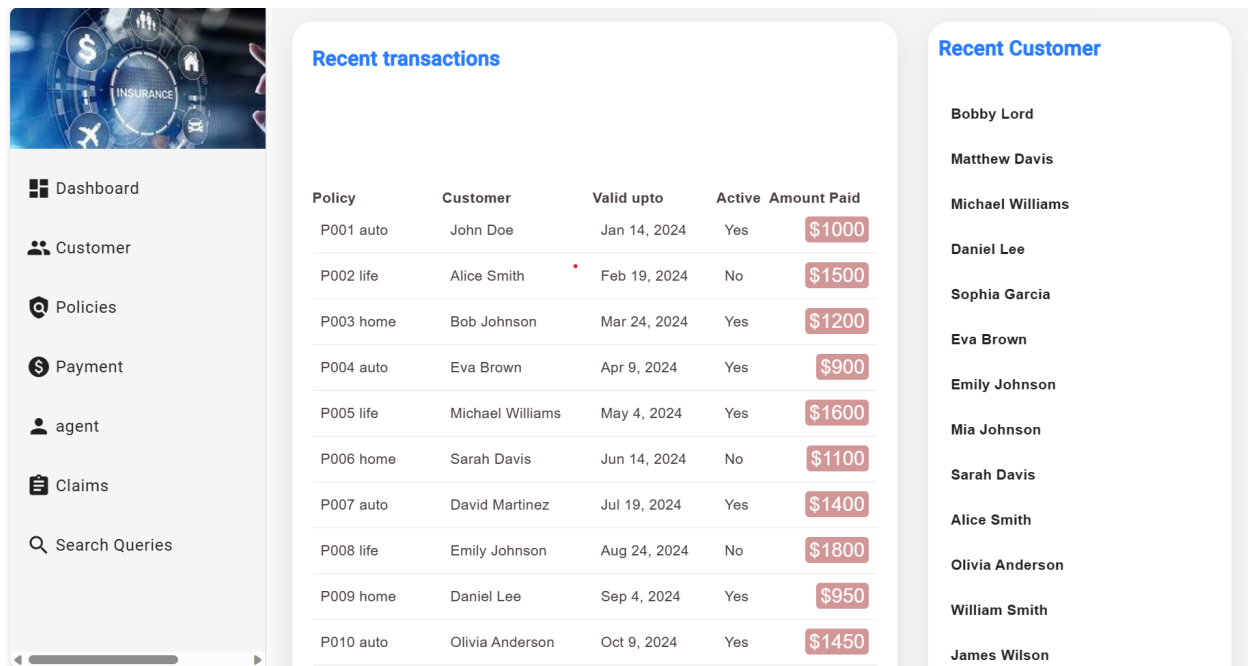
Collections: This code operates on three collections: "payment," "policy," and "customer."

Purpose and Logic: The primary objective of this code is to retrieve data related to customer payments and their associated policies. It achieves this by performing a series of aggregation operations that involve joining relevant data from these collections.

Data Processing: The code starts by looking up customer and policy details based on their respective IDs. It then unwinds the joined customer and policy data for alignment. After that, it projects the necessary fields into a structured format.

Output: The result is a list of "CustomerPaymentPolicyDTO" objects, each containing comprehensive information about customers, their associated policies, and payment details.

Recent customer tab: This tab shows the list of customers who were recently added.



The screenshot displays a web application dashboard with a sidebar on the left and two main content areas. The sidebar contains a navigation menu with the following items: Dashboard, Customer, Policies, Payment, agent, Claims, and Search Queries. The top of the dashboard features a header image with the word 'INSURANCE' and various icons. The main content area is divided into two panels. The left panel, titled 'Recent transactions', contains a table with the following data:

Policy	Customer	Valid upto	Active	Amount Paid
P001 auto	John Doe	Jan 14, 2024	Yes	\$1000
P002 life	Alice Smith	Feb 19, 2024	No	\$1500
P003 home	Bob Johnson	Mar 24, 2024	Yes	\$1200
P004 auto	Eva Brown	Apr 9, 2024	Yes	\$900
P005 life	Michael Williams	May 4, 2024	Yes	\$1600
P006 home	Sarah Davis	Jun 14, 2024	No	\$1100
P007 auto	David Martinez	Jul 19, 2024	Yes	\$1400
P008 life	Emily Johnson	Aug 24, 2024	No	\$1800
P009 home	Daniel Lee	Sep 4, 2024	Yes	\$950
P010 auto	Olivia Anderson	Oct 9, 2024	Yes	\$1450

The right panel, titled 'Recent Customer', displays a list of customer names: Bobby Lord, Matthew Davis, Michael Williams, Daniel Lee, Sophia Garcia, Eva Brown, Emily Johnson, Mia Johnson, Sarah Davis, Alice Smith, Olivia Anderson, William Smith, and James Wilson.

Team Member Accomplishments:

- Sanjana Achan - Developed Front-end application, visualizing the data and wrote the queries for MongoDB.
- Pradeep Ranjan – Developed Back-end, created the No-sql database and wrote the queries for MongoDB.

