# Constellation POC Insights – Modernizing Enterprise Solutions

Constellation in **Pega** is the **latest UI architecture** introduced from **Pega Infinity '23 and beyond**. It replaces the older **section-based UI (App Studio + Theme Cosmos)** with a **design system-driven UI** that gives a modern, responsive, and performance-optimized experience.

Here's a breakdown:

---

🔹 **What is Constellation?**

- A **design system–based UI framework** in Pega.
- Provides **ready-to-use UX templates** (like forms, case views, and dashboards).
- Focuses on **low-code configuration** rather than heavy customizations.
- Built with **ReactJS micro-frontends** for high performance.

---

🔹 **Why Constellation?**

1. **Consistency** – Same UI patterns across all applications.
2. **Speed** – Lightweight, faster rendering compared to section-based UI.
3. **Upgrade friendly** – Since customization is minimal, upgrades won't break the UI.
4. **Omnichannel** – Works across desktop, mobile, portal with responsive design.
5. **Integration-ready** – Works smoothly with **DX API** for headless deployment.

---

🔹 **Key Features**

- **Case view templates** → Summary panel, primary/secondary stages, actions.
- **Declarative UI** → You define "what" should show, not "how" it should render.
- **Simplified authoring** → Business users can configure case views directly.
- **No heavy sections / layouts** → Most are replaced with auto-generated React components.
- **Widgets & Cards** → Replace traditional harness & section-based portals.
- **Separation of UI & Logic** → Logic remains in case rules, UI handled by Constellation.

---

🔹 **Constellation vs Theme Cosmos (Old UI)**

| Feature | Theme Cosmos | Constellation |
|---|---|---|
| Technology | Section-based UI (server-side rendering) | React-based UI (client-side rendering) |
| Customization | High (can modify sections, harnesses) | Low (uses templates, minimal overrides) |

| Feature | Theme Cosmos | Constellation |
|---|---|---|
| Performance | Slower (large DOM, reloads often) | Faster (lightweight, API-driven) |
| Upgrade effort | High (custom UI often breaks) | Low (since UI is standardized) |
| DX API | Optional | Core to architecture |

---

- **When to Use Constellation?**
  - New applications on **Pega Infinity 23+** → recommended.
  - Applications that need **headless deployment** or **modern UX**.
  - When upgrade effort & maintainability are a priority.
  - Not ideal if you need **highly customized UI** (Cosmos may still be used).

forvandle tech bytes

# Blueprint in Pega Constellation

---

◆ **What is Blueprint?**

- A **visual design tool** in **Pega Constellation**.
- Helps **business & IT collaborate** to design applications quickly.
- Captures **case types, personas, data, and channels** at a **conceptual level**.
- Auto-generates a **working application skeleton** with Constellation UI.

---

◆ **Key Capabilities**

- Define **Case Types** (workflows, stages, steps).
- Add **Personas** (users & roles).
- Model **Data Objects** (e.g., Employee, Loan, Account).
- Select **Channels** (Web, Mobile, Email, Chat).
- Automatically generate:
  - Case structures
  - Personas & roles
  - Navigation & dashboards
  - Data relationships

---

◆ **Why Use Blueprint?**

⬛ **Faster Application Delivery** – avoids manual setup.
⬛ **Business-IT Collaboration** – SMEs define processes visually.
⬛ **Best Practices by Default** – consistent design system.
⬛ **Upgrade Friendly** – aligned with Constellation's architecture.

---

◆ **Example: Employee Onboarding**

- Case: **Onboarding**
- Stages: **Collect Docs → HR Review → Manager Approval → Provision Access**
- Personas: **Employee, HR, Manager**
- Data: **Employee Profile, Asset Request**

– Pega auto-generates case type, Constellation UI templates, and navigation.

◆ **Blueprint = Accelerator**

"Blueprint acts as a **sketchpad for application design**, turning ideas into a working Constellation app in minutes."

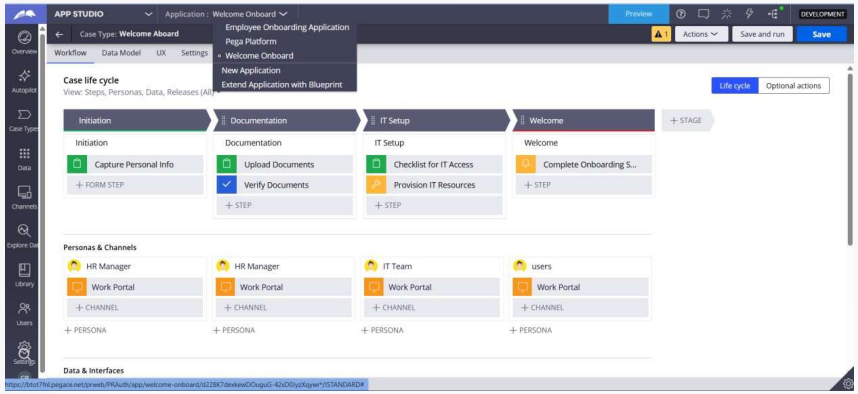
) Suggestion: For your presentation, you can also add a **simple diagram** like this:

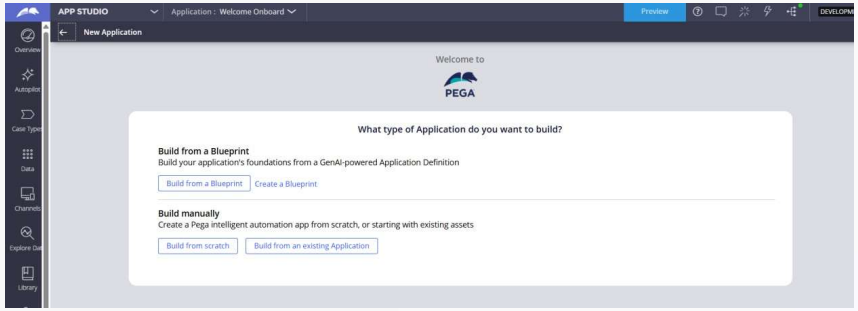
**Personas → Case Types → Data Objects → Channels → Auto-Generated App**

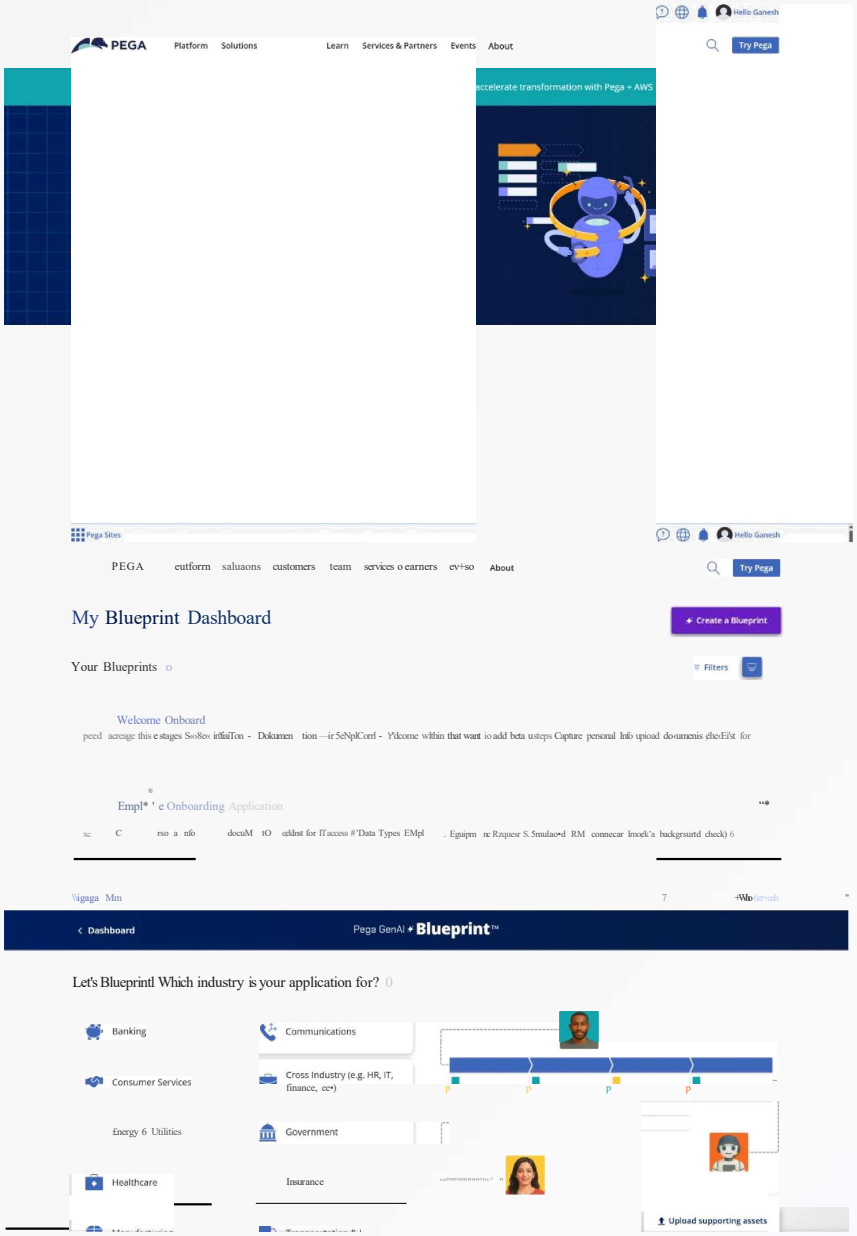
How to create Blueprint?

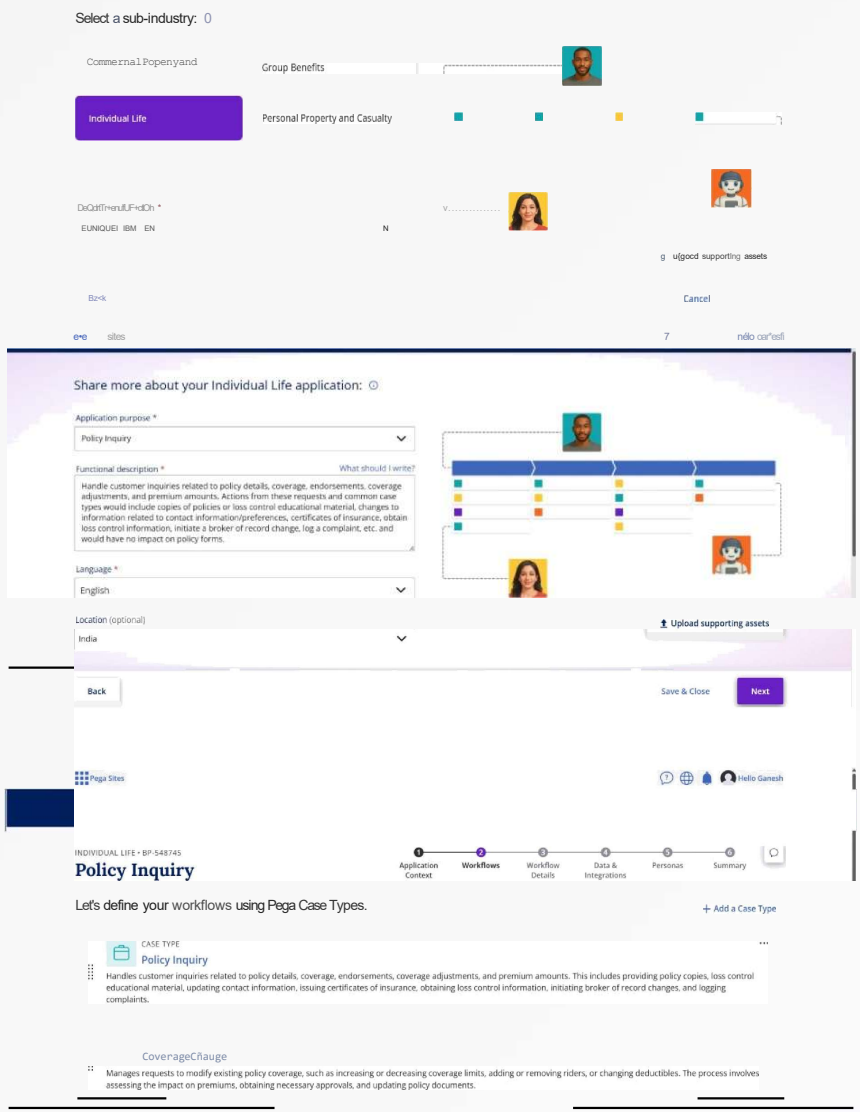
Design: -

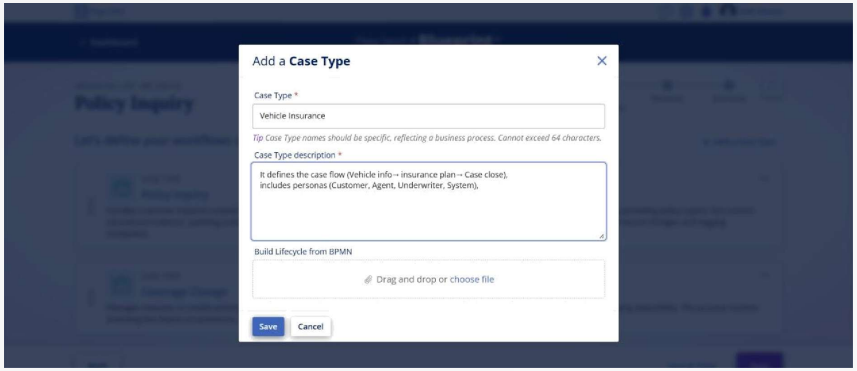
For new Blueprint→Click on create a blueprint

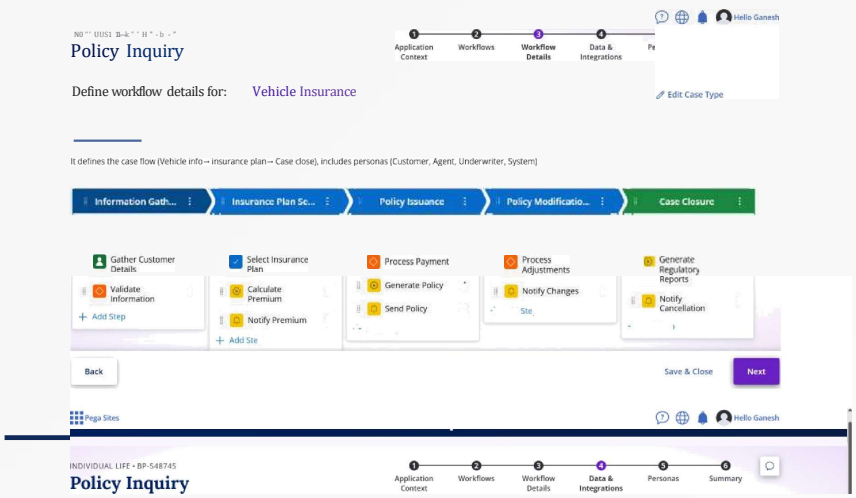


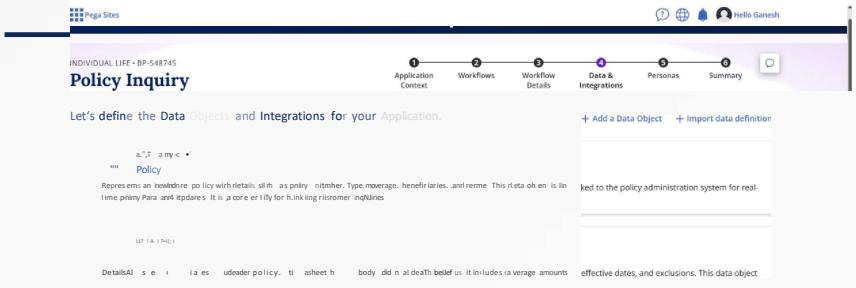




PEGA
Platform
Solutions
Learn
Services & Partners
Events
About
Hello Ganesh
Try Pega
accelerate transformation with Pega + AWS
Pega Sites
PEGA
eutform
saluaons
customers
team
services o earners
ev+so
About
Hello Ganesh
Try Pega
My Blueprint Dashboard
Create a Blueprint
Your Blueprints
Filters
Welcome Onboard
Empl* ' e Onboarding Application
Dashboard
Pega GenAI Blueprint™
Let's Blueprintl Which industry is your application for?
Banking
Consumer Services
Energy 6 Utilities
Healthcare
Communications
Cross Industry (e.g. HR, IT, finance, cc•)
Government
Insurance
Upload supporting assets

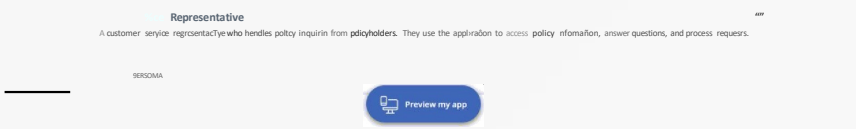

Select a sub-industry: 0
Commernal Popenyand
Group Benefits
Individual Life
Personal Property and Casualty
EUNIQUEI IBM EN
N
g u(gocd supporting assets
Bz<k
Cancel
sites
7
nélo car"esfi
Share more about your Individual Life application:
Application purpose *
Policy Inquiry
Functional description *
What should I write?
Handle customer inquiries related to policy details, coverage, endorsements, coverage adjustments, and premium amounts. Actions from these requests and common case types would include copies of policies or loss control educational material, changes to information related to contact information/preferences, certificates of insurance, obtain loss control information, initiate a broker of record change, log a complaint, etc. and would have no impact on policy forms.
Language *
English
Location (optional)
India
Upload supporting assets
Back
Save & Close
Next
Pega Sites
Hello Ganesh
INDIVIDUAL LIFE - BP-548745
Policy Inquiry
1 Application Context
2 Workflows
3 Workflow Details
4 Data & Integrations
5 Personas
6 Summary
Let's define your workflows using Pega Case Types.
+ Add a Case Type
CASE TYPE
Policy Inquiry
Handles customer inquiries related to policy details, coverage, endorsements, coverage adjustments, and premium amounts. This includes providing policy copies, loss control educational material, updating contact information, issuing certificates of insurance, obtaining loss control information, initiating broker of record changes, and logging complaints.
CoverageCñauge
Manages requests to modify existing policy coverage, such as increasing or decreasing coverage limits, adding or removing riders, or changing deductibles. The process involves assessing the impact on premiums, obtaining necessary approvals, and updating policy documents.

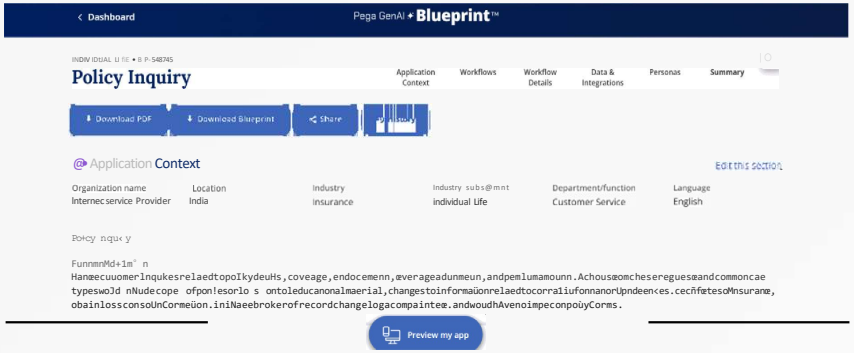

Add a Case Type
Case Type *
Vehicle Insurance
Tip Case Type names should be specific, reflecting a business process. Cannot exceed 64 characters.
Case Type description *
It defines the case flow (Vehicle info→ insurance plan→ Case close),
includes personas (Customer, Agent, Underwriter, System),
Build Lifecycle from BPMN
Drag and drop or choose file
Save
Cancel

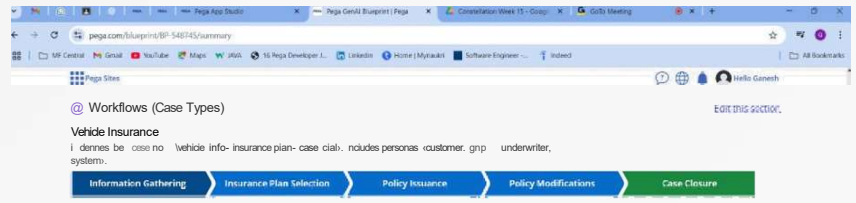

Policy Inquiry
Application Context
Workflows
Workflow Details
Data & Integrations
Hello Ganesh
Edit Case Type
Define workflow details for: Vehicle Insurance
It defines the case flow (Vehicle info→ insurance plan→ Case close), includes personas (Customer, Agent, Underwriter, System)
Information Gath...
Insurance Plan Sc...
Policy Issuance
Policy Modificatio...
Case Closure
Gather Customer Details
Validate Information
Add Step
Select Insurance Plan
Calculate Premium
Notify Premium
Add Ste
Process Payment
Generate Policy
Send Policy
Process Adjustments
Notify Changes
Generate Regulatory Reports
Notify Cancellation
Back
Save & Close
Next

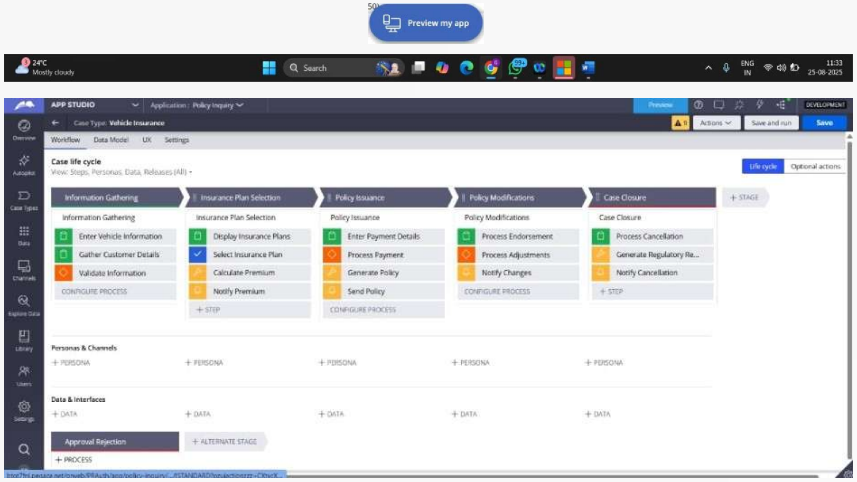

Pega Sites
Hello Ganesh
INDIVIDUAL LIFE • BP-548745
Policy Inquiry
Application Context
Workflows
Workflow Details
Data & Integrations
Personas
Summary
Let's define the Data Objects and Integrations for your Application.
+ Add a Data Object
+ Import data definition
Policy
ked to the policy administration system for real-
effective dates, and exclusions. This data object

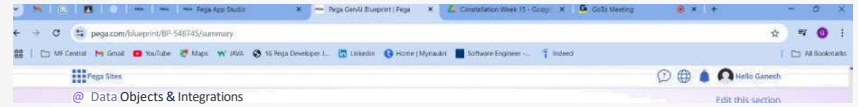

Pega Sites
Hello Ganesh
Dashboard
Pega GenAI + Blueprint™
INDIVIDUAL LIFE • BP-548745
Policy Inquiry
1 Application Context
2 Workflows
3 Workflow Details
4 Data & Integrations
5 Personas
6 Summary
Who are the Personas involved in your workflows?
+ Add a Persona
PERSONA
Policyholder
A policyholder who needs to inquire about their policy details, coverage, or premium amounts. They may also request policy documents or updates to their contact information.


Representative
A customer serviœ regrœsentacTye who hendles poltcy inquirin from pdicyholders. They use the applıvraöon to access policy nfomañon, answer questions, and process requesrs.
9ERSOMA
Preview my app


Dashboard
Pega GenAI + Blueprint™
Policy Inquiry
Application Context
Workflows
Workflow Details
Data & Integrations
Personas
Summary
Download PDF
Download Blueprint
Share
Application Context
Edit this section
Organization name
Internec service Provider
Location
India
Industry
insurance
Industry subs@mnt
individual Life
Department/function
Customer Service
Language
English
Policy nqux y
FunnmnMd+1m˚ n
HanœecuuomerlnqukesrelaedtopolkydeuHs,coveage,endocemenn,œverageadunmeun,andpemlumamounn.Achousœomcheseregueswandcommoncae typeswold nNudecope ofponlesorlo s ontoleducanonalmaerial,changestoinformaüonrelaedtocorraliufonnanorUpndeen‹es.cecñfœtesoMnsuranæ, obainlossconsoUnCormeüon.iniNaeebrokerofrecordchangelogacompainteœ.andwoudhAvenoimpeconpoüyCorms.
Preview my app


@ Workflows (Case Types)
Edit this section.
Vehicle Insurance
Information Gathering
Insurance Plan Selection
Policy Issuance
Policy Modifications
Case Closure


Preview my app
APP STUDIO
Case Type: Vehicle Insurance
Workflow
Data Model
UX
Settings
Case life cycle
Information Gathering
Insurance Plan Selection
Policy Issuance
Policy Modifications
Case Closure
Enter Vehicle Information
Gather Customer Details
Validate Information
Display Insurance Plans
Select Insurance Plan
Calculate Premium
Notify Premium
Enter Payment Details
Process Payment
Generate Policy
Send Policy
Process Endorsement
Process Adjustments
Notify Changes
Process Cancellation
Generate Regulatory Re...
Notify Cancellation
Personas & Channels
Data & Interfaces
Approval Rejection



@ Data Objects & Integrations

Edit this section

No data objects were found for your Blueprint.

9 Personas

Edit this section

Policyholder

A policyholder who neeos la inquire about their policy details, coverage, or premium amount They may also regVies:potty documents or updates to their contact information.

Service Representative

A cuscomer service representativewho handles policy inquiries from policyholders. They use the application co access policy informacon, answer queszions, and process

Team Lead

A team lead or supervisor who oversees the customer service team. They use the application to monitor performance, track trends, and identify areas for improvement.

Insurance Agent


**Share Blueprint**

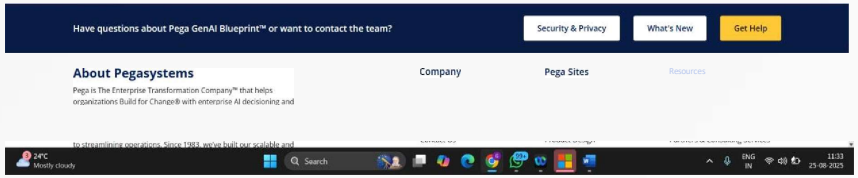
Save as a PDFto send around and share your vision toyour team!

Download PDF

D•pley yeur Bluaprin4

Download your Blueprint which can be Imported toyour Pega Platfom as a en a .

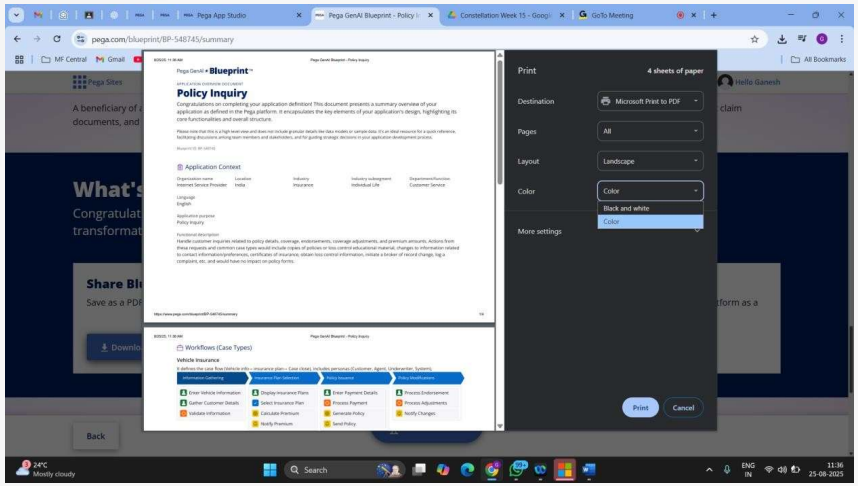
Download Blueprint

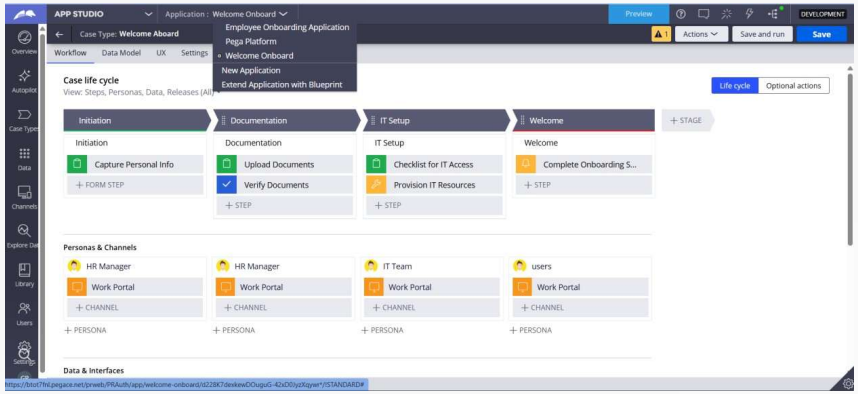



Policy Inquiry 20250B25T060352371 GMT.blueprint

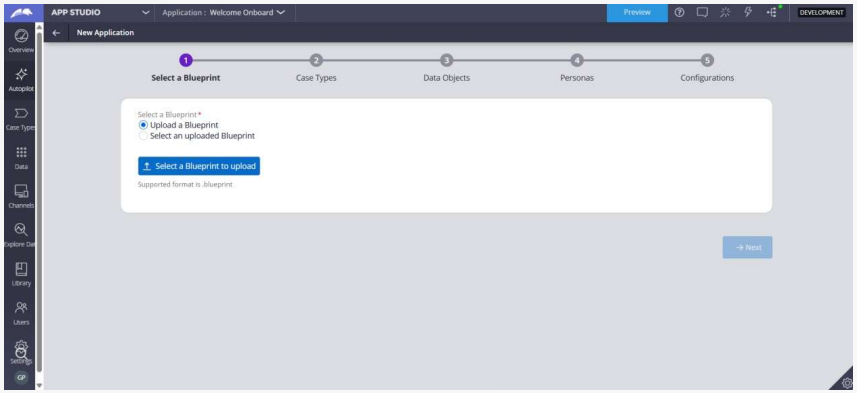
65.5 KB • Dane

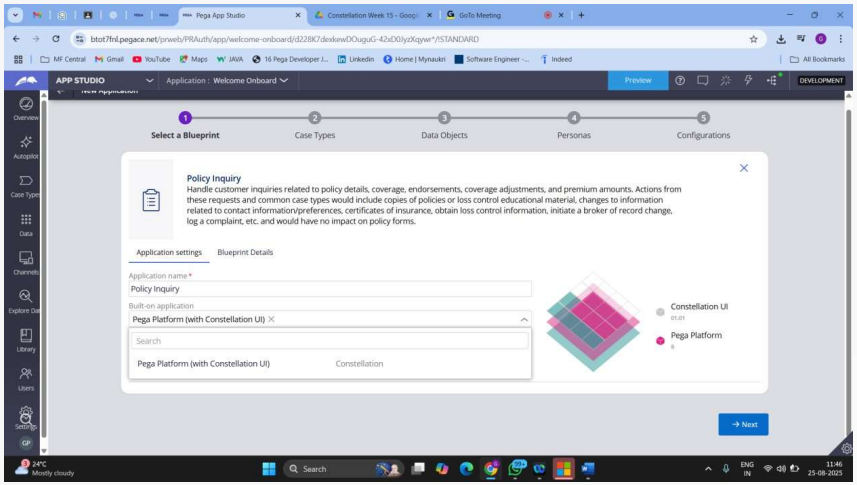
forvandle
tech bytes

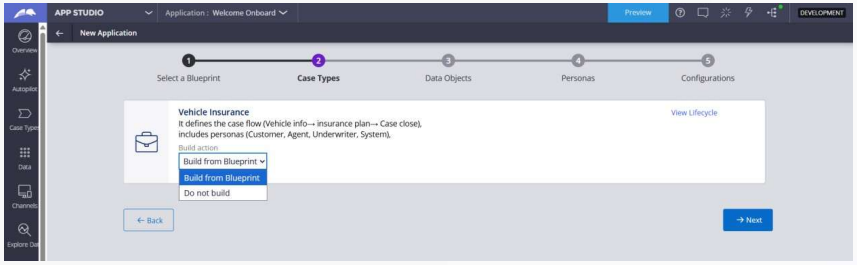


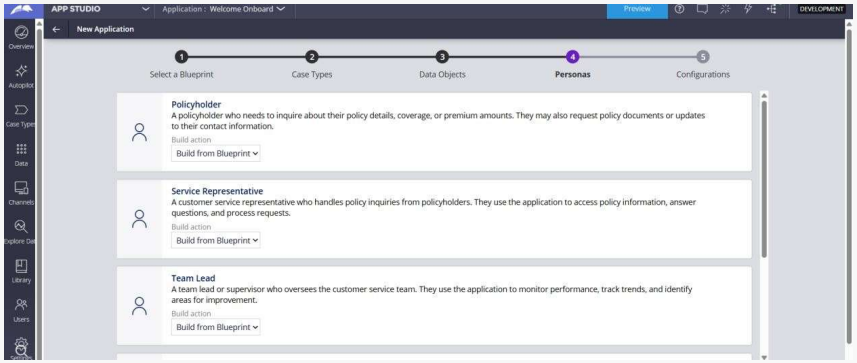



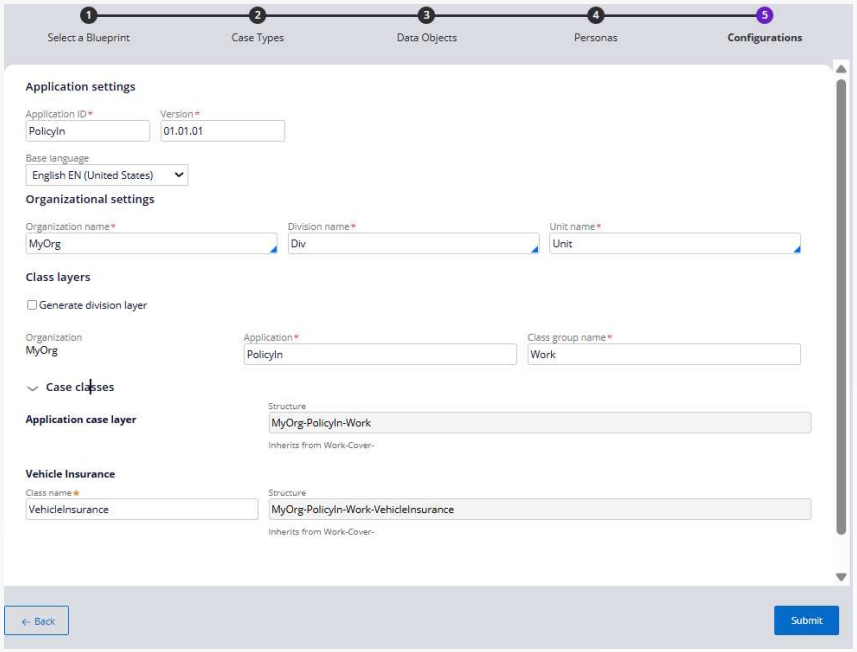
forvandle tech bytes



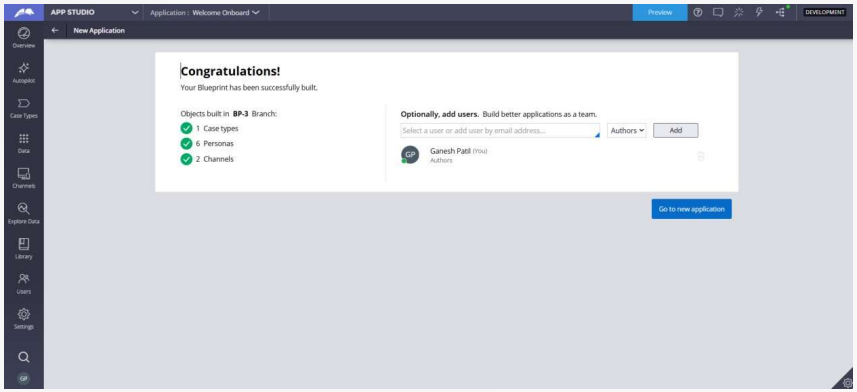



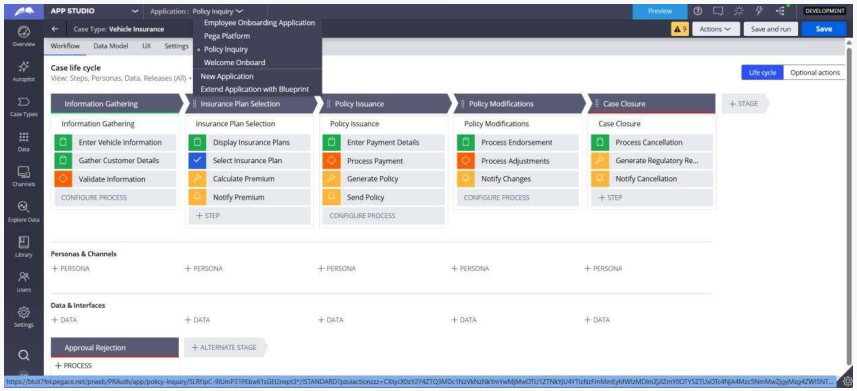



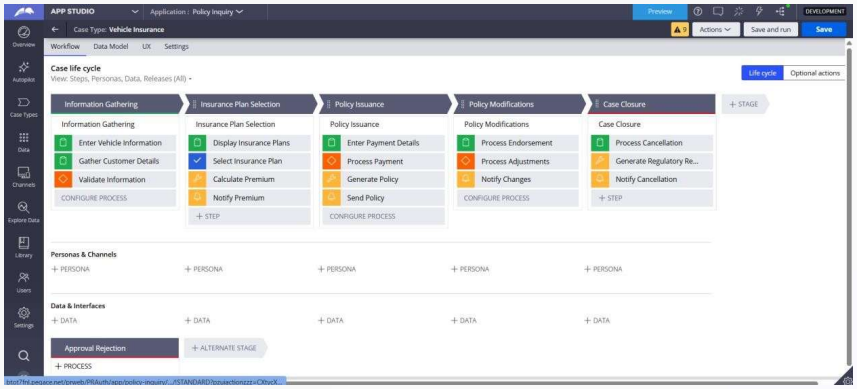

APP STUDIO
Application : Welcome Onboard
Preview
DEVELOPMENT
New Application
Select a Blueprint
Case Types
Data Objects
Personas
Configurations
Policyholder
A policyholder who needs to inquire about their policy details, coverage, or premium amounts. They may also request policy documents or updates to their contact information.
Build action
Build from Blueprint
Service Representative
A customer service representative who handles policy inquiries from policyholders. They use the application to access policy information, answer questions, and process requests.
Build action
Build from Blueprint
Team Lead
A team lead or supervisor who oversees the customer service team. They use the application to monitor performance, track trends, and identify areas for improvement.
Build action
Build from Blueprint


Select a Blueprint
Case Types
Data Objects
Personas
Configurations
Application settings
Application ID*
PolicyIn
Version*
01.01.01
Base language
English EN (United States)
Organizational settings
Organization name*
MyOrg
Division name*
Div
Unit name*
Unit
Class layers
Generate division layer
Organization
MyOrg
Application*
PolicyIn
Class group name*
Work
Case classes
Application case layer
Structure
MyOrg-PolicyIn-Work
Inherits from Work-Cover-
Vehicle Insurance
Class name*
VehicleInsurance
Structure
MyOrg-PolicyIn-Work-VehicleInsurance
Inherits from Work-Cover-
← Back
Submit

forvandle
tech bytes



**Autopilot in Pega Constellation**

- **Autopilot** is Pega's **AI-assisted build feature** that helps developers and business users configure applications faster.
- It uses **generative AI + natural language prompts** to create case types, stages, data objects, and even UI views automatically.
- Acts like a **co-pilot inside App Studio**, reducing manual effort and accelerating app development while ensuring Constellation best practices.
- While opening Constellation, you will be able to see two additional rulesets added: **PegaGenAI** and **PegaNLP**. After selecting the option, the final prompt will be passed, and based on that, it will be processed.

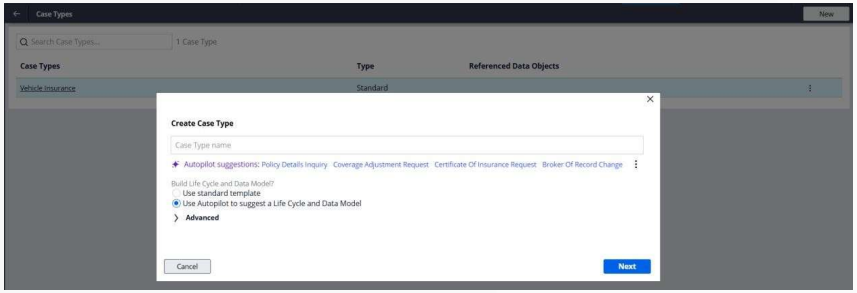
In **Pega Constellation**, the **Autopilot feature** is available inside **App Studio** (Infinity '23+).

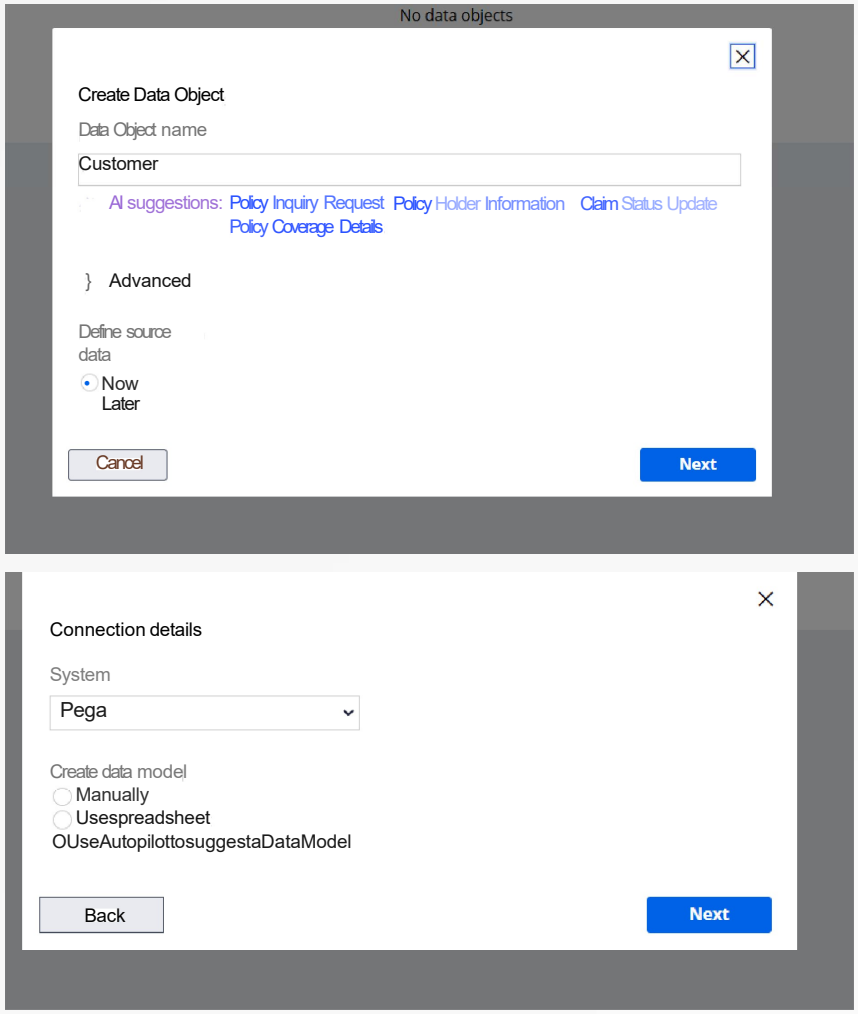
🔹 You can see it in these places:

1. **When creating a new application** – Autopilot suggests case types, personas, and data objects from natural language input.
2. **Within App Studio (Case Type Designer)** – You'll find an **Autopilot panel / "Ask Autopilot" option**, where you can describe what you want (e.g., *"Create a Vehicle Insurance case with stages for Application, Verification, and Issuance"*), and it will auto-generate the flow.
3. **In Data & Personas setup** – Autopilot helps define **data objects, personas, and roles** based on prompts.

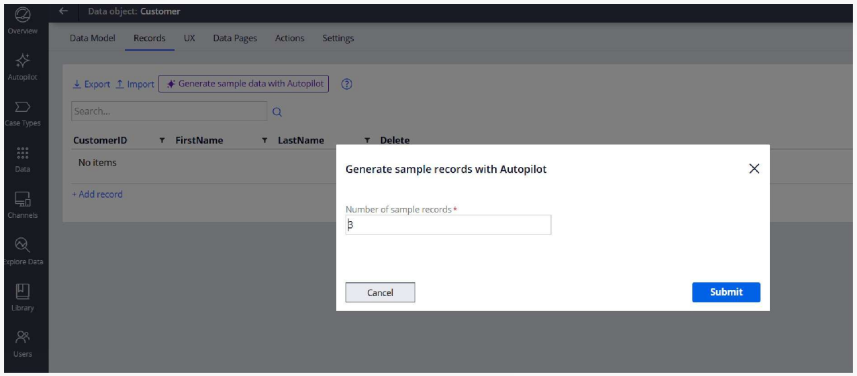
---

⬛ In short: You see **Autopilot embedded in App Studio**, mainly in **New App creation** and **Case Type configuration screens**.

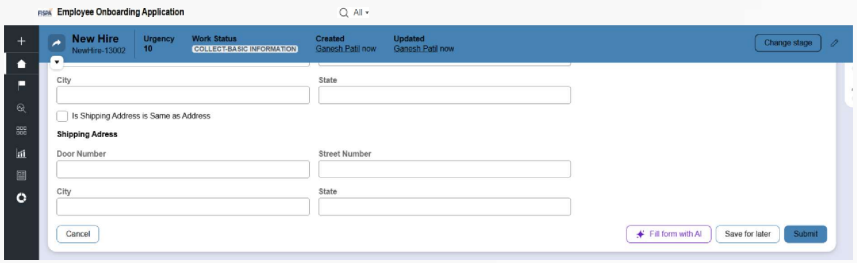



No data objects
Create Data Object
Data Object name
Customer
AI suggestions: Policy Inquiry Request Policy Holder Information Claim Status Update Policy Coverage Details
Advanced
Define source data
Now
Later
Cancel
Next
Connection details
System
Pega
Create data model
Manually
Usespreadsheet
OUseAutopilottosuggestaDataModel
Back
Next

forvandle tech bytes







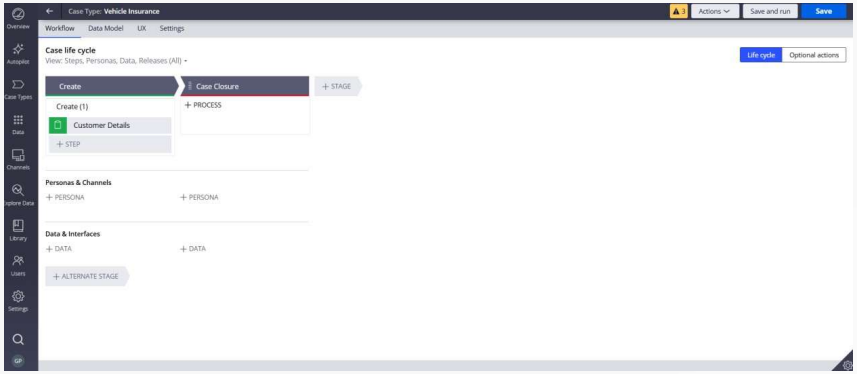
OOTB Rulesets related to GenAI

- PEGA -NLP
- PEGA-GEN AI
- PEGA -BLUEPRINT
- PEGA-BIGDATA

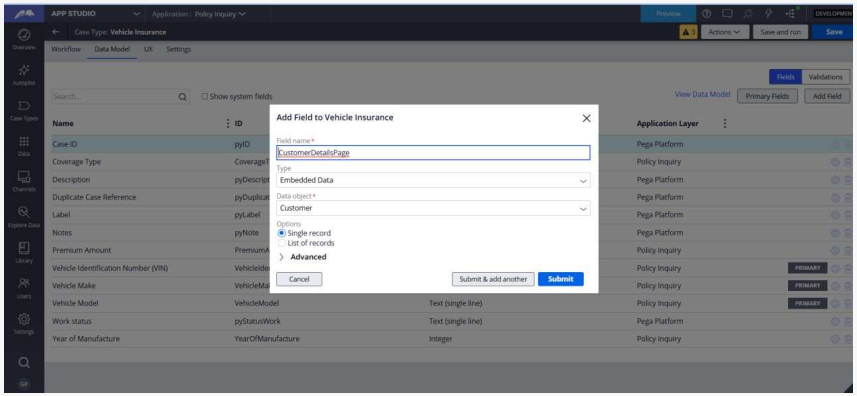
OOTB activities

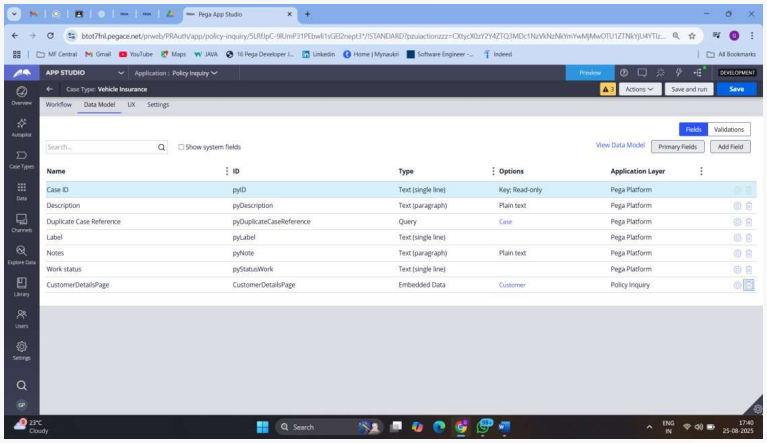
- pzGetGenerativeAIDefinitions: Core OOTB activtiy
- pxConnectToGenerativeAI: Makes a call to connector rule of calling GEN AI Rest Service
- pzGetAICaseTypes: Getting case type suggested names
- pyPostGetGenerativeAIResponse :Get the Response

- ✓ **Events and actions are not supported in Pega Constellation, and on-change refresh is not possible.**
- ✓ **We cannot add custom buttons in Constellation.**
- ✓ **There is no concept of sections; instead, Constellation uses views.**
- ✓ We are done with the case type creation; now let's focus on configuring the view.

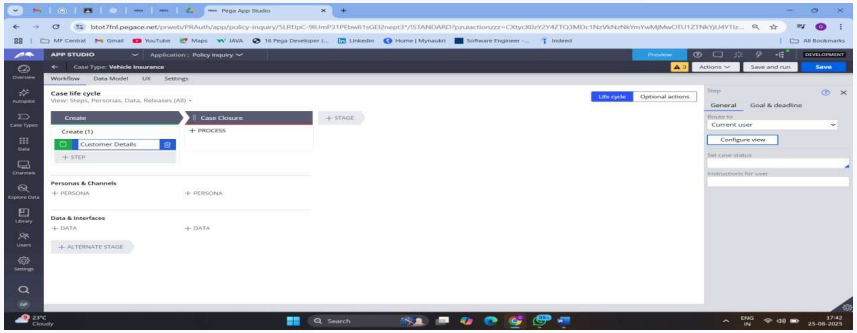


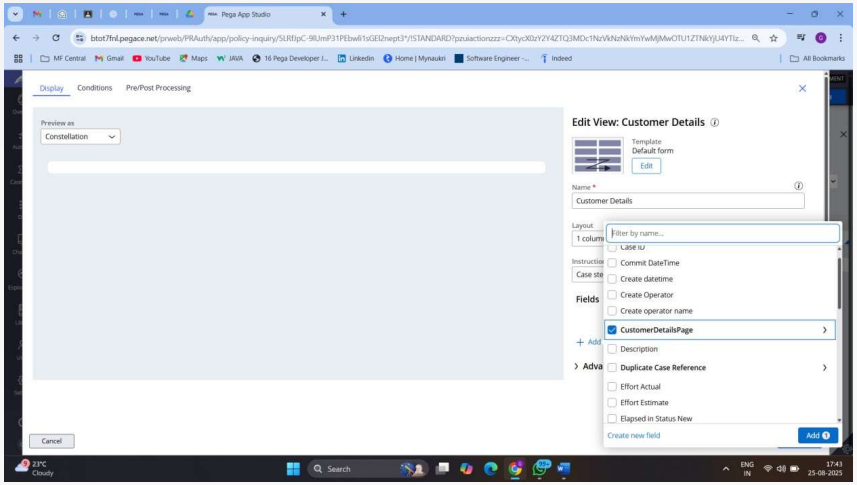

Before configuring the view, I will create a page property from the data model of the case type. Since want to collect data, I am selecting **Embedded Data**.



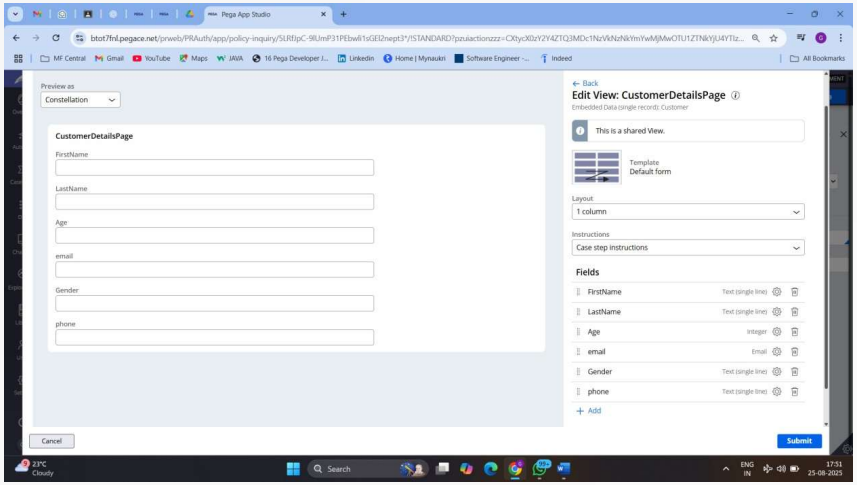




Then, open the **Workflow** tab, select the assignment, and configure the view.

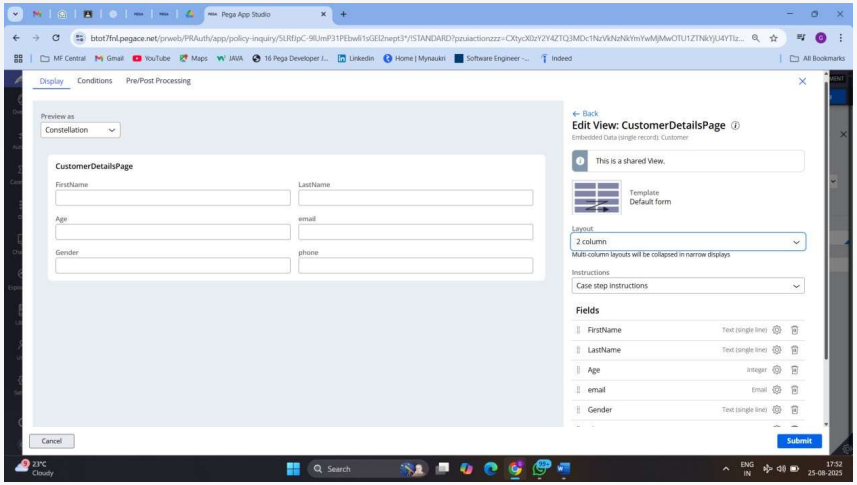




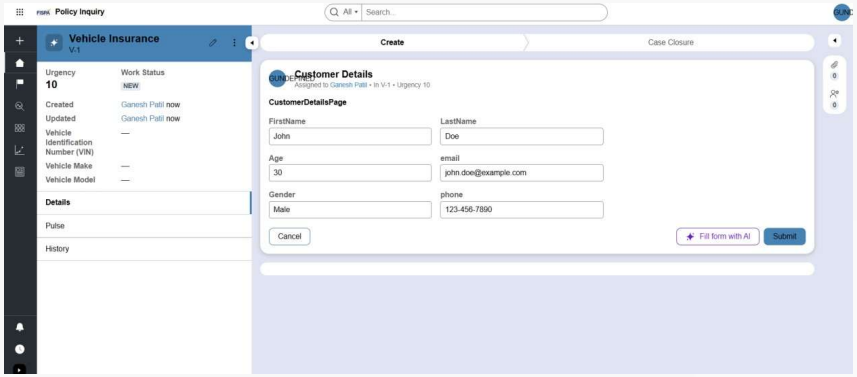

Click on the page name, and then add the properties (fields) that you want to use as input fields.




Select the desired layout from the Layout dropdown. You can change it from 1 to 2 or 3, as shown in the screenshot.




Save your changes, run the case, and check whether the flow is working as expected.



Working with Views –

Data Class Views

Case Type Views

Primary Fields

Embedding Data Class Views into Case Type Class Views

'' z Requirement

Create a new screen to collect Billing Address and Shipping Address details in a page format.

Add a checkbox between them. When the checkbox is selected, the Shipping Address should be copied to the Billing Address.

'' z Design

Create a Data Object – Address

Fields:

Street

City

State

Postal Code

Country

Add Two Page Properties in the case type data model:

BillingAddress → Type: Address (Page)

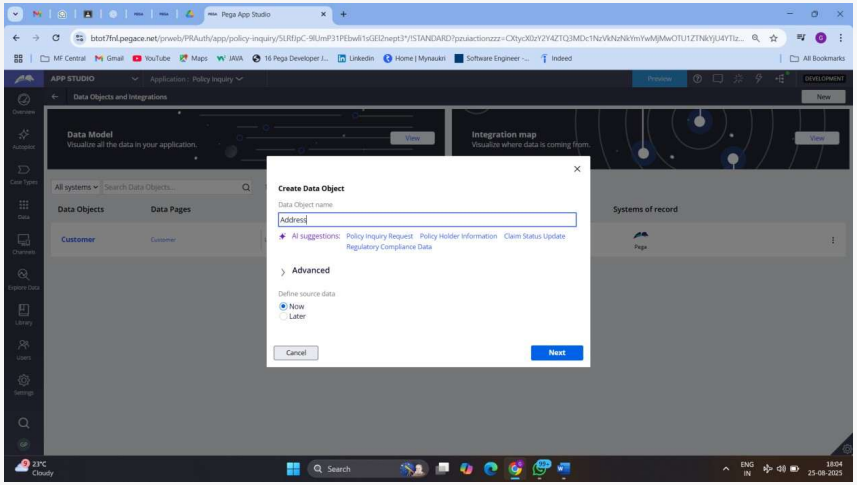
ShippingAddress → Type: Address (Page)

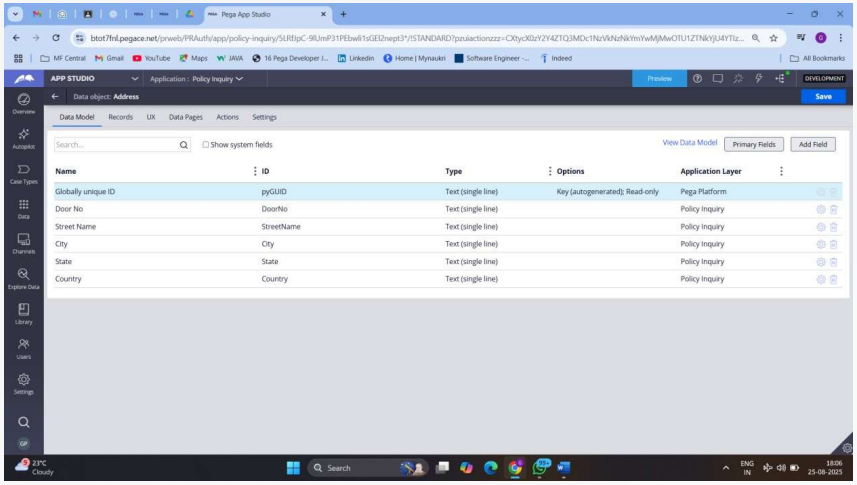
Configure the View

Display both Billing Address and Shipping Address sections.

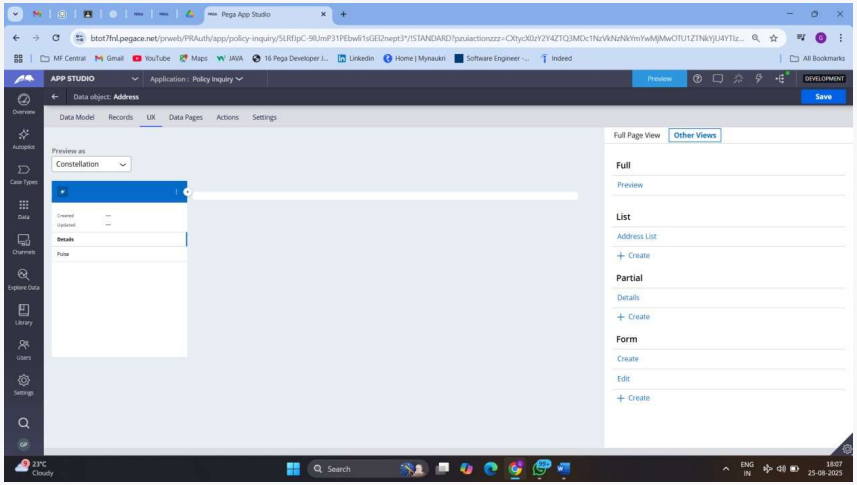
Add a Checkbox (e.g., “Same as Shipping Address”).

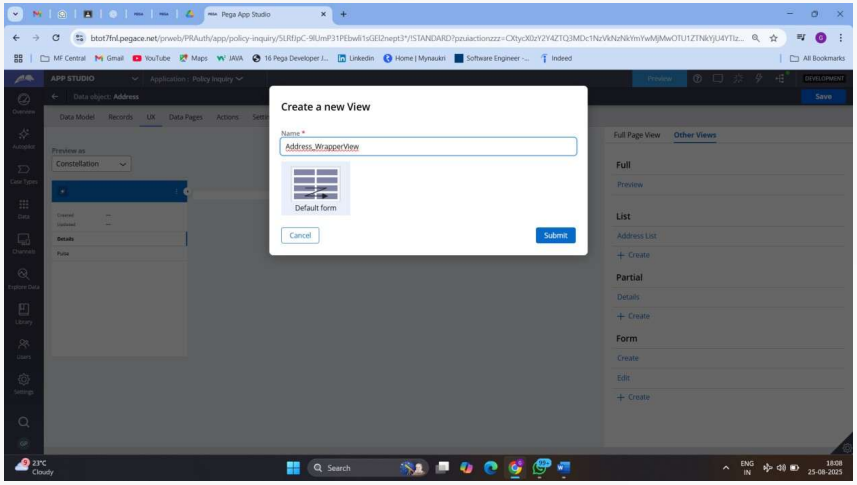
On selection of the checkbox → Use Data Transform / Copy action to copy values from ShippingAddress → BillingAddress.



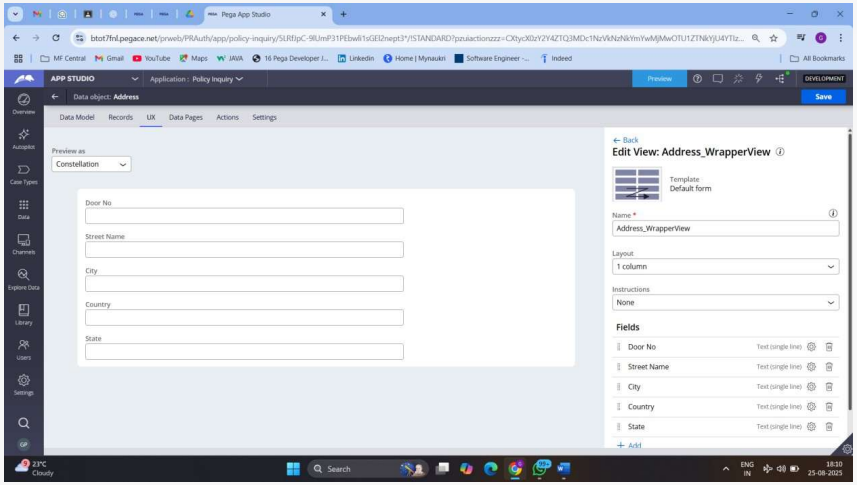




After adding the property, create the view. To do this, go to the UX tab, click on Other Views, and then add a Form view.



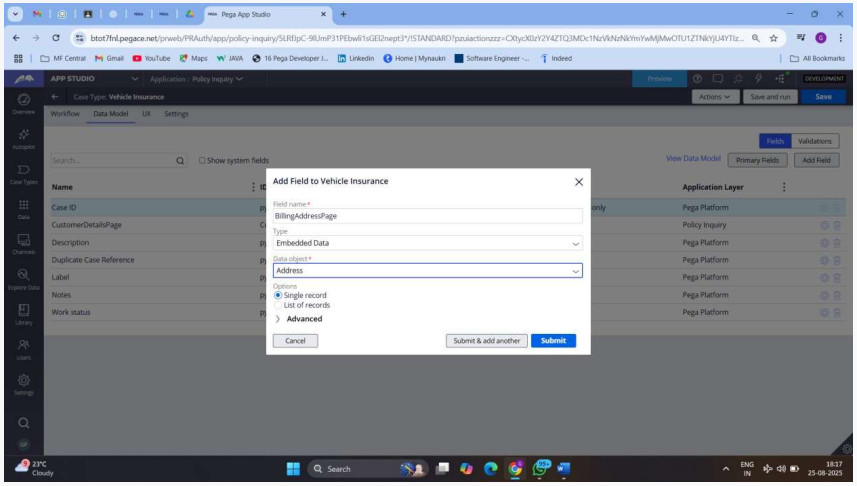

Click on the newly created view, add the required properties, and rearrange them in the desired order.

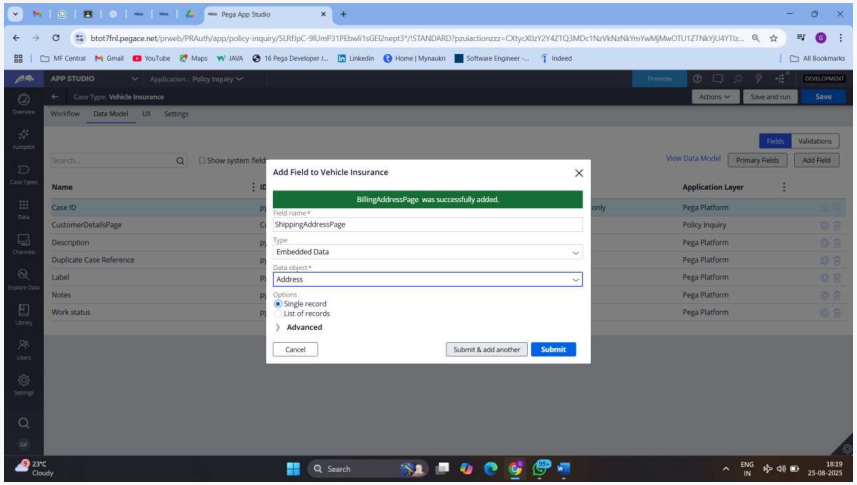


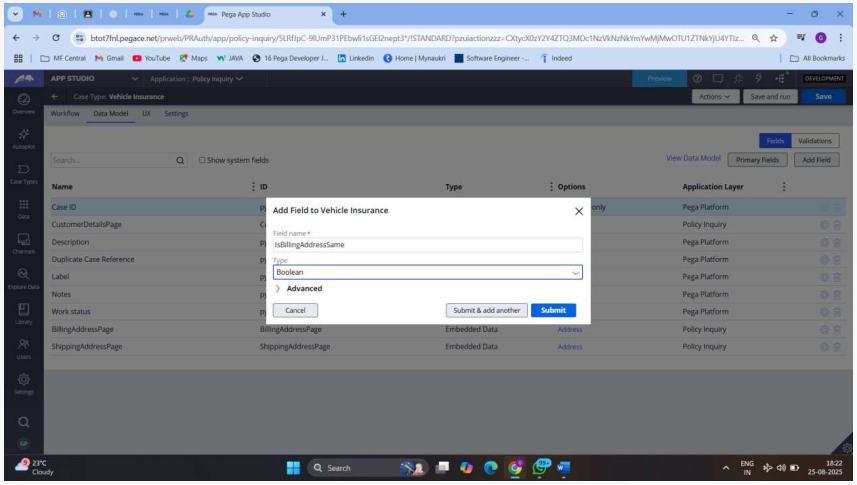

This is the Data Class View, which we created within the Data Type.

If I want to refer to the Data View inside a Case Type View, I need to establish a relationship. To do this, I create a Single Page property in the Case Type class and reference the Address Data Type.

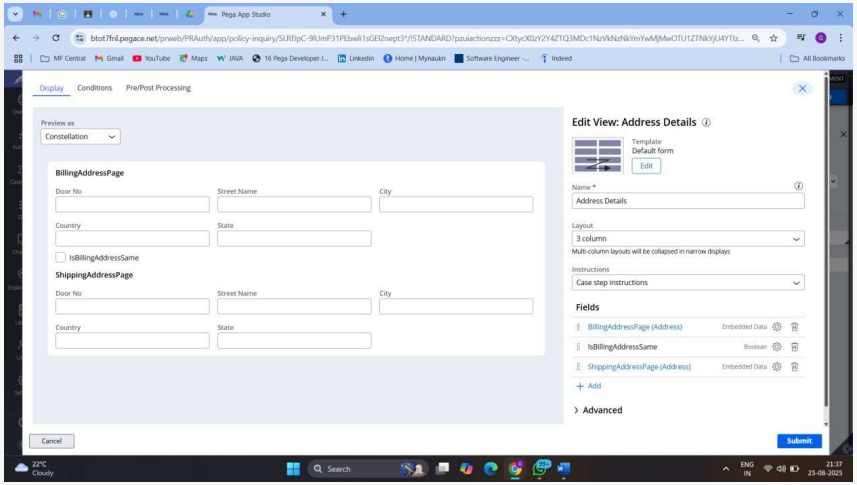
Since my requirement involves two addresses, I am creating two single-page properties accordingly.

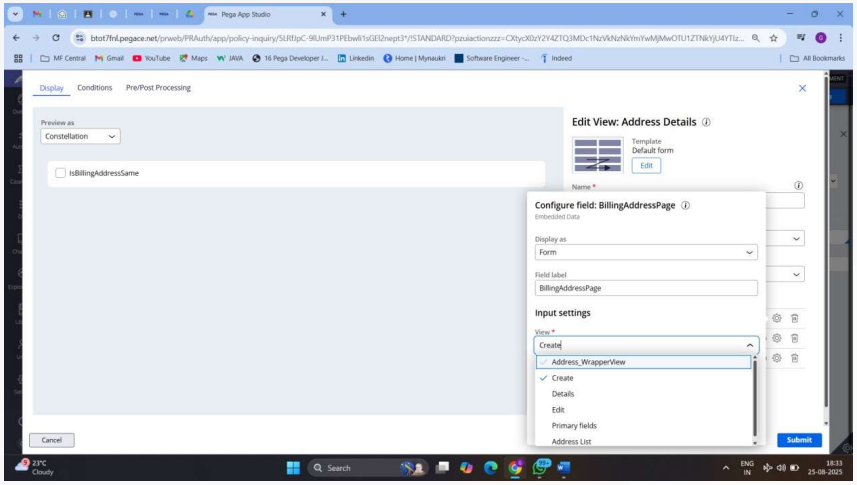





Step: Add an assignment named Address Details and configure the view.



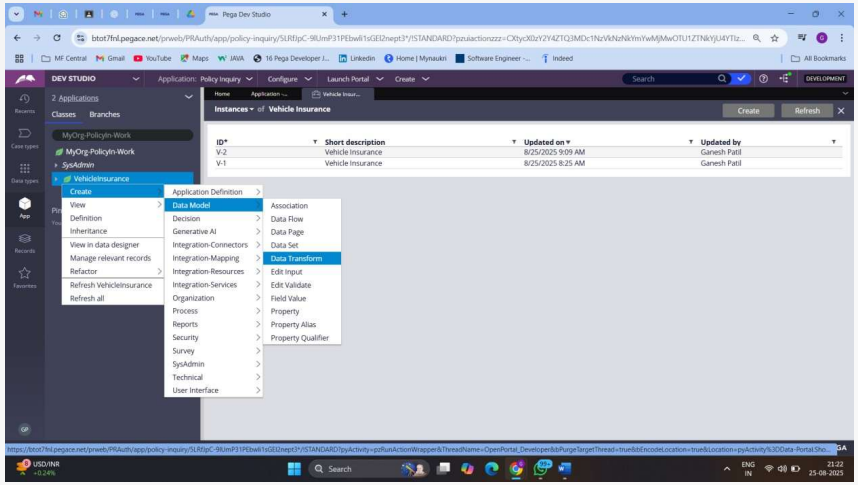

## Remaining Development

Implement functionality to copy the Billing Address to the Shipping Address when the radio button value is set to Yes.

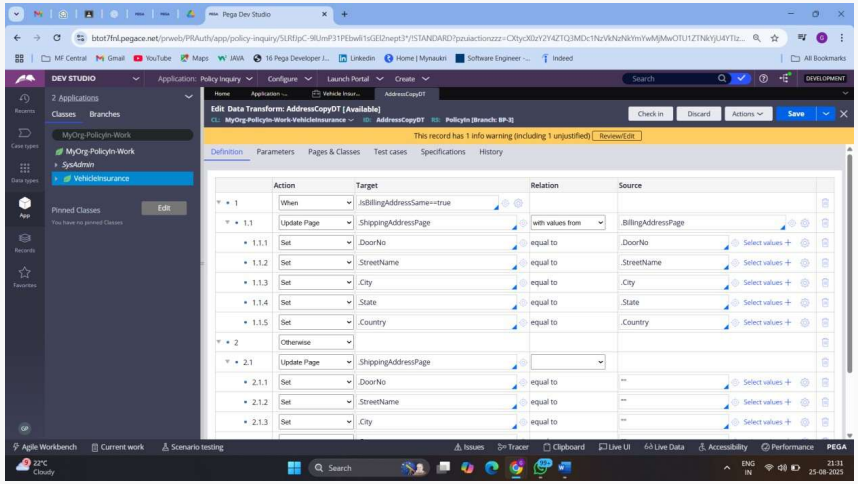
## Design

Use Dev Studio to create a Data Transform that copies values from the BillingAddress page to the ShippingAddress page.

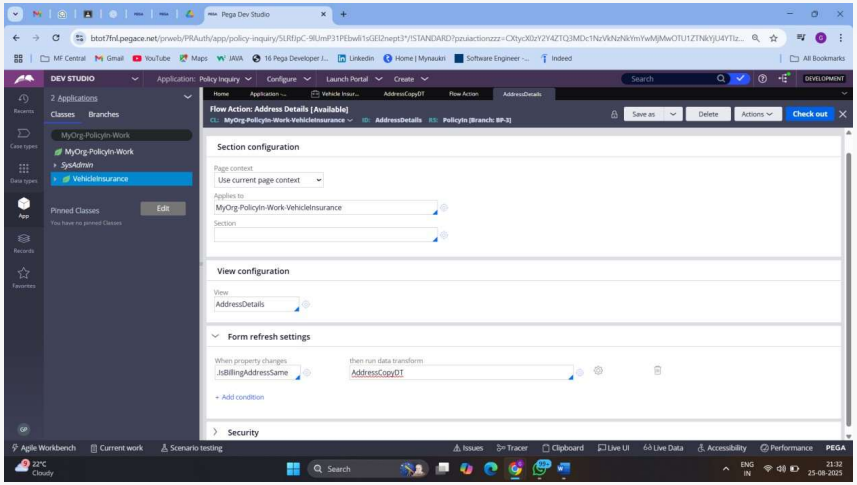
Configure the radio button's action to call this Data Transform when the value is Yes.



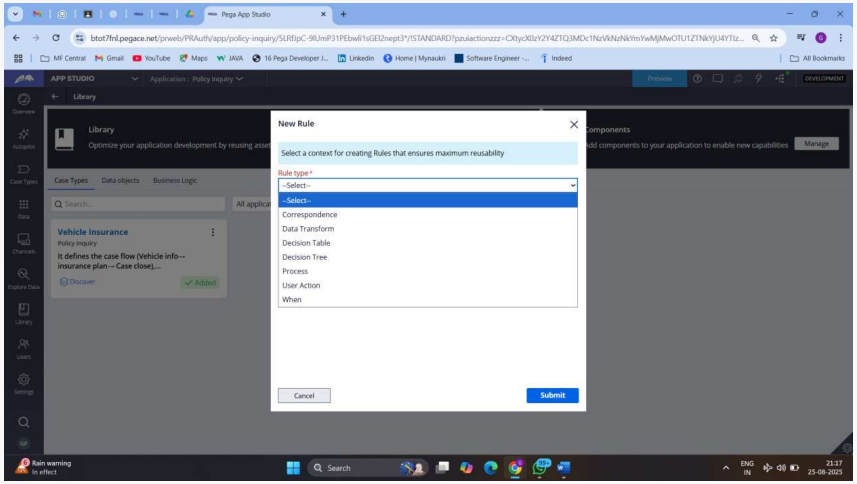

\




We can call this Data Transform from the Flow Action.




What are the rules we can create from App studio:Library→Create new



# Primary Fields in Constellation

**Definition: Key attributes of a case or data object that represent it in the UI.**

**Purpose: Used to display records in lists, search results, and cards for clear identification.**

**Example (Vehicle Insurance): Policy Number, Vehicle Registration, Customer Name, Status.**

- **In short: Primary fields make cases and data objects easily recognizable in Constellation.**

**Purpose of Primary Fields in**

**Constellation Uniquely identify a case or data**

**object.**

**Summarize key details in lists, cards, and search results.**

**Provide a consistent view across the Constellation UI.**

- **They help users quickly recognize and work with cases or data without opening full details.**

**Note: When fields are marked as Primary Fields, Pega automatically creates the pyPrimaryView.**

**Requirement**

**Create a new screen to collect single vehicle data.**

**Create a Data Object named Vehicle with fields like:**

**VIN (Vehicle Identification Number)**

**VRN (Vehicle Registration Number)**

**Manufacturing Company**

**(Add more as required…)**

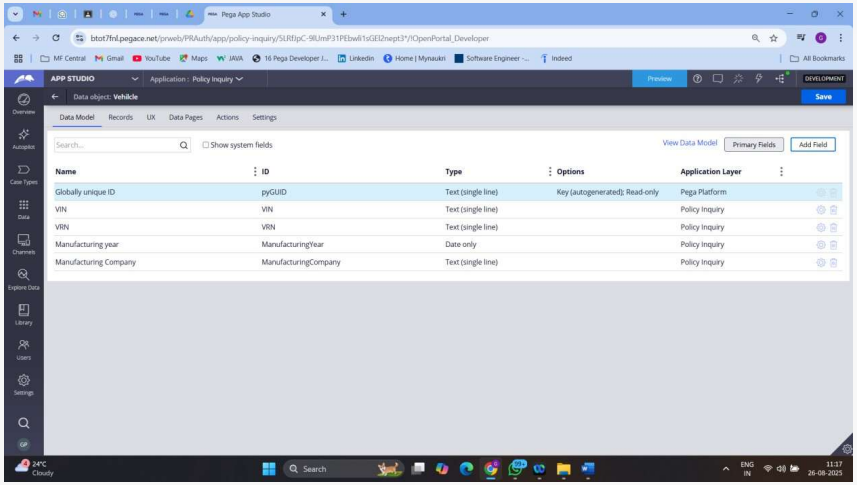
**Design**

**Create a Data Object Vehicle in App Studio → Library.**

**Add the fields (VIN, VRN, Manufacturing Company, etc.).**

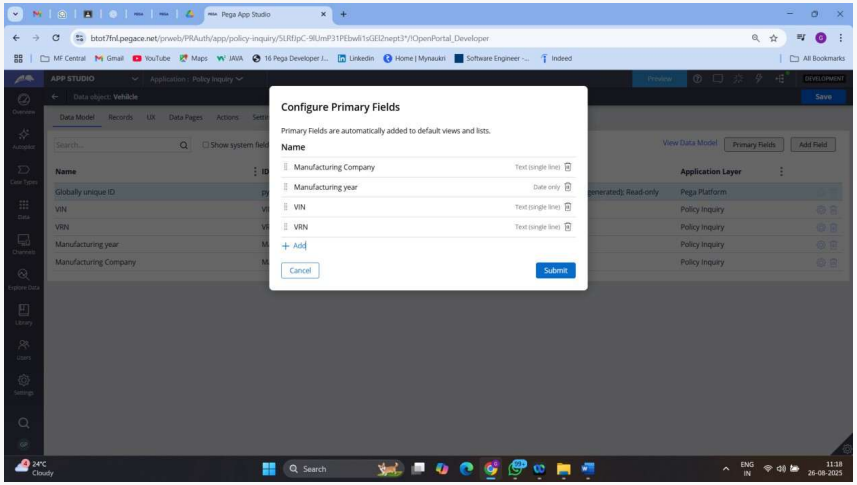
**Mark important fields (like VIN and VRN) as Primary Fields.**

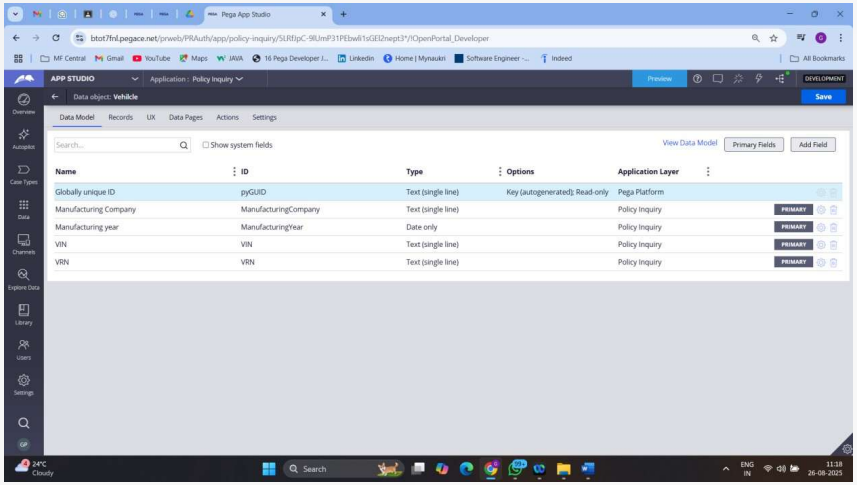
**Pega will auto-generate the pyPrimaryView for consistent display.**



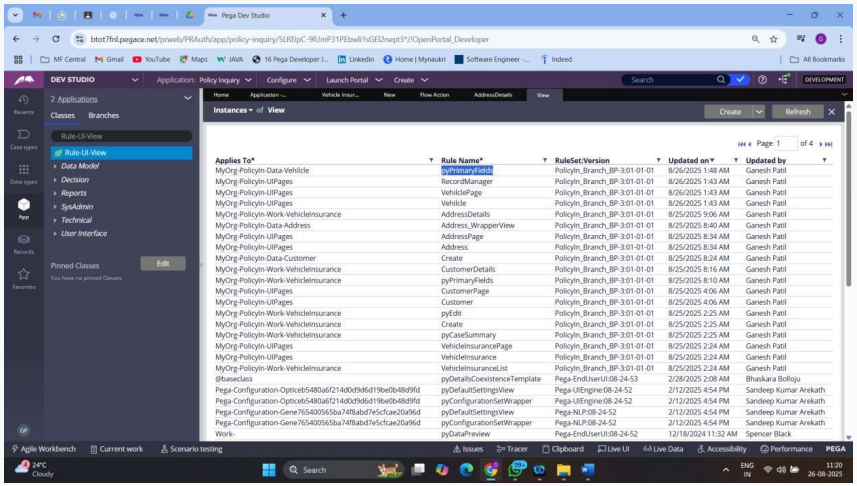

Mark these four fields as Primary Fields.

Go to Primary Fields option and click to select them.

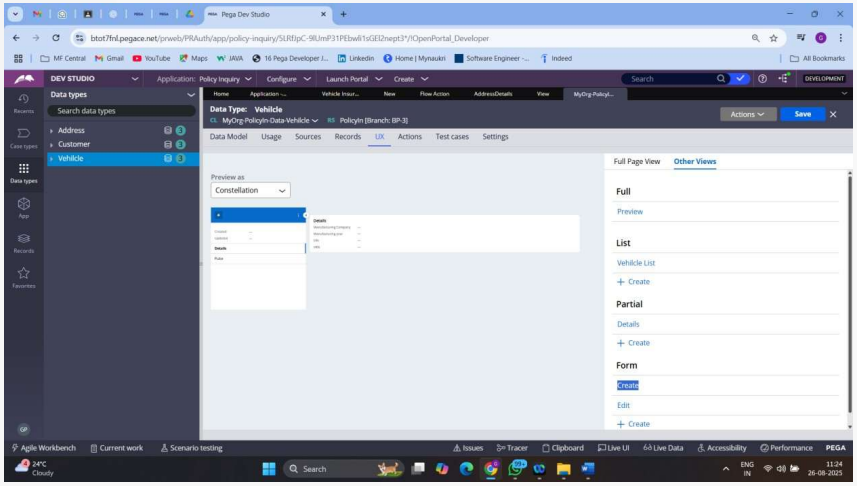


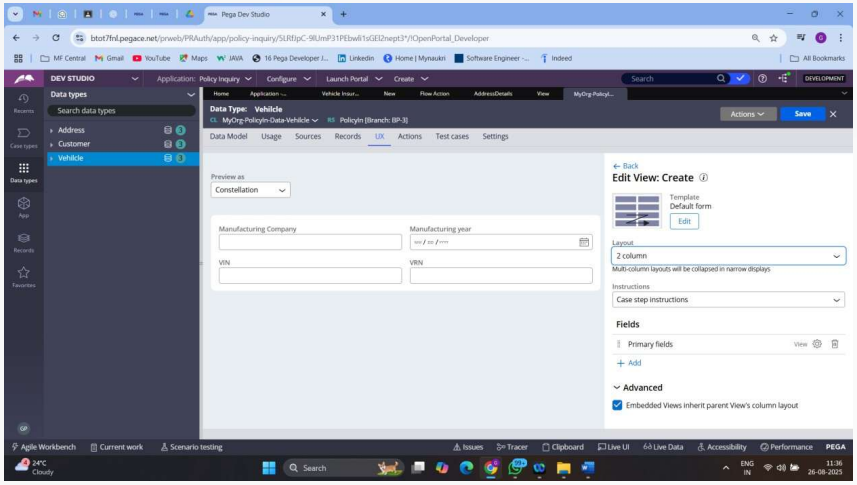



If we mark the fields as Primary, then the pyPrimaryView gets generated automatically, and we can view this under Rule-UI-View.

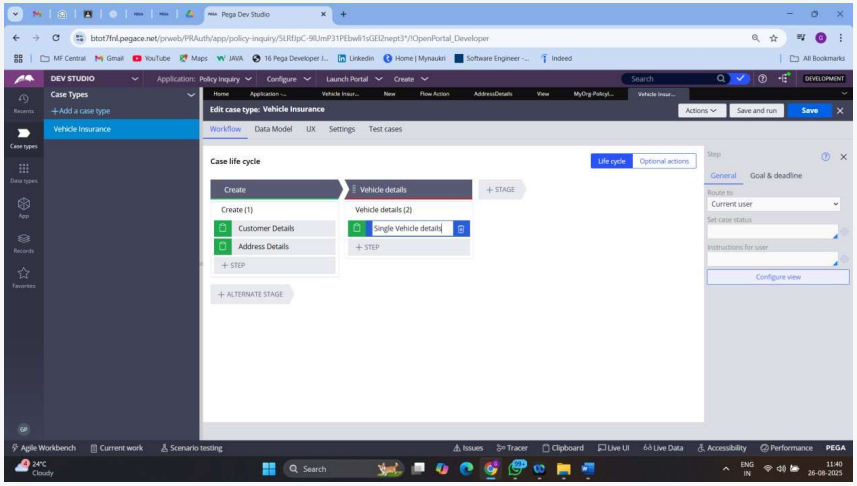


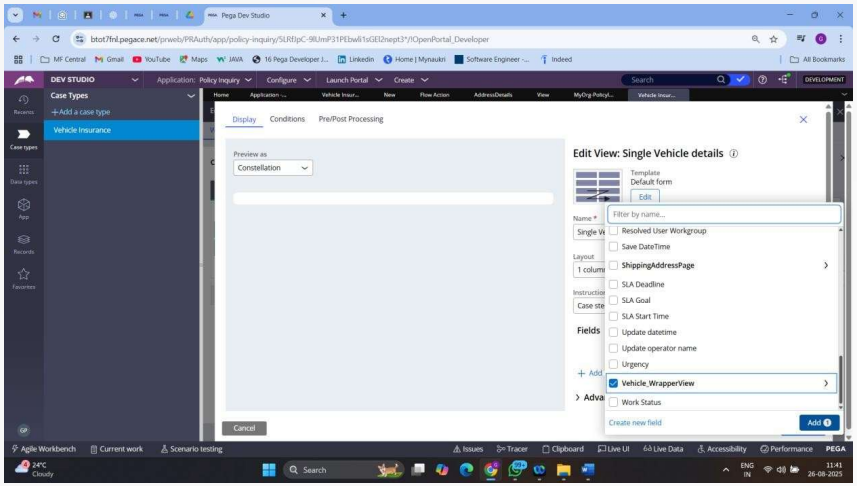
Navigate to Data Type → UX tab → Other Views → Form View → Click on Create.

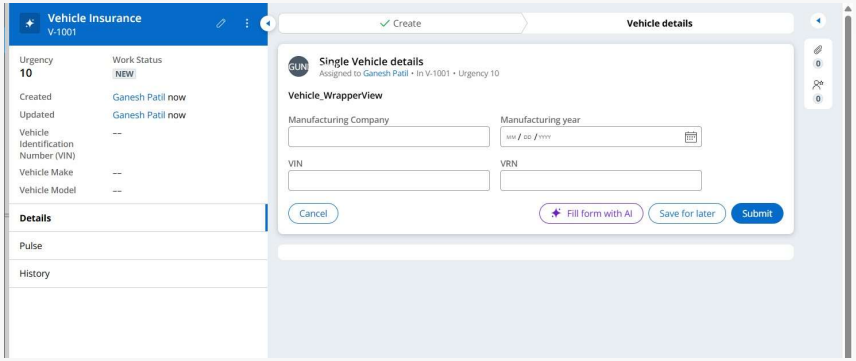


Save the changes and run the case to check whether it is working as expected or not.






## Types of Views in Constellation

**Form View** – Used to collect or display data in fields.

Example: Application Form

**List View** – Displays a list of cases or data records.

Example: All Vehicle Insurance Policies

**Details View** – Provides a full summary of a single case or record.

Summary Panel View – Shows key case information alongside the workflow.

**Dashboard View** – Combines widgets/cards for reports, charts, and quick actions.

Example: Customer Profile Page

Definition of a View in Pega Constellation

A View is an instance of the class Rule-UI-View, which stores component details in JSON format.

Views are reusable and help in structuring the UI efficiently.

Classic Pega (Section-based UI) – Full Case View / Full Page View

Equivalent to the Work Object Form.

Contains:

Heading

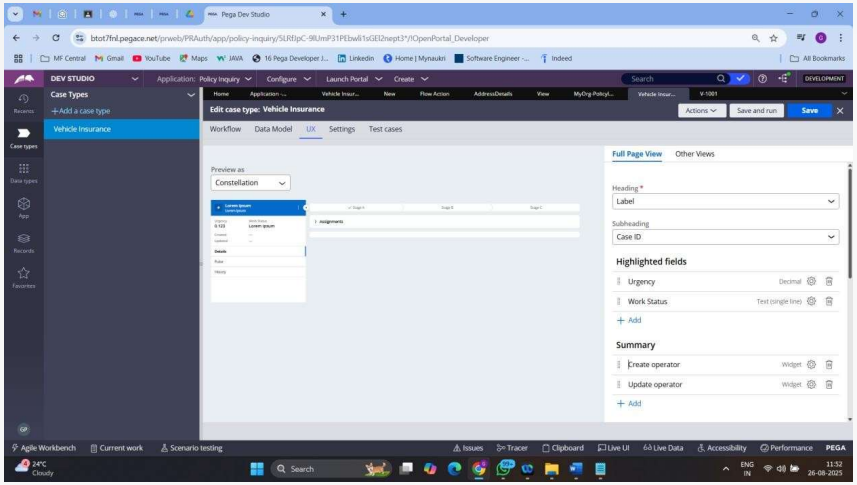
Subheading

Highlighted Fields

Summary

Tabs

Utilities and – Flow Actions and Buttons get loaded into this Full Page View.




Constellation Views – Key Types

Preview

Any changes made in the UX tab will be reflected here.

It provides a live preview of UI changes before running the case.

Details View (falls under Partial Views category)

Read-only representation of the case (similar to Confirm Harness in PRPC).

Typically used at the end of a transaction to display the entire case data.

All views related to the case can be added into this Details View.

Primary Fields View (auto-generated)

Created automatically when we mark fields as Primary Fields.

Helps uniquely identify cases/data in lists, search results, and summaries.

- In short:

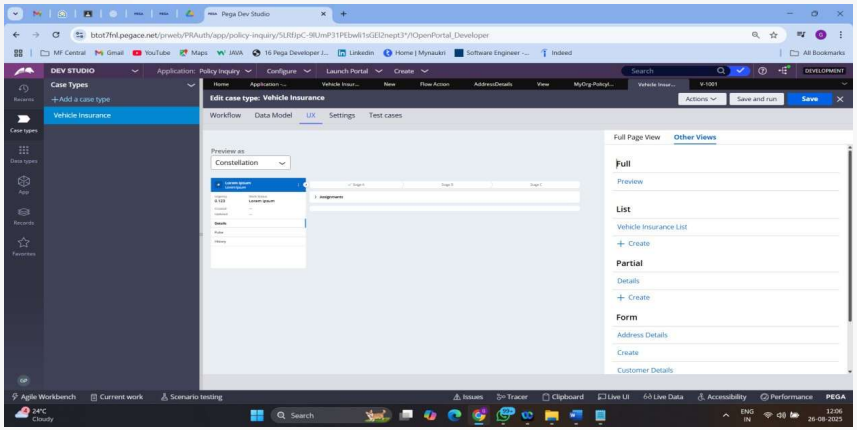
Form View → Collect input data

Details View → Display full case summary (read-only)

List View → Display multiple records (tabular)

Primary Fields View → Auto-generated identification view

Preview → Live check of UI changes




Single Select – Dropdown & Customer Search (Auto Complete + Dropdown)

Requirement:

Create a Customer table with fields like:

Customer ID

First Name

Last Name

Email

Mobile Number

Address …

Create a Customer Search screen:

Use Auto Complete and Dropdown controls.

Display the selected customer's data in a page format.

Design Steps:

Create Customer Data Object

Go to App Studio → Data → Create Data Object.

Name: Customer

Add fields: Customer ID, First Name, Last Name, Email, Mobile, Address, etc.

Mark Customer ID + First Name as Primary Fields for easy identification.

Create a Single Page Property in Case Type

In the Case Type → Data Model, add a Single Page property.

Reference the Customer Data Object.

This allows the case to store one selected customer's details.

Configure Customer Search View

Go to UX → Views → Add Form View (e.g., CustomerSearch).

Add Customer ID / Name as Auto Complete control:

Source: Data Page (D_CustomerList).

On selection → populate customer details into the single page property.

Add Dropdown control for another field (e.g., Customer Type or Location).

Display Customer Details in Page Format

Once the customer is selected, bind the Single Page (Customer) property.

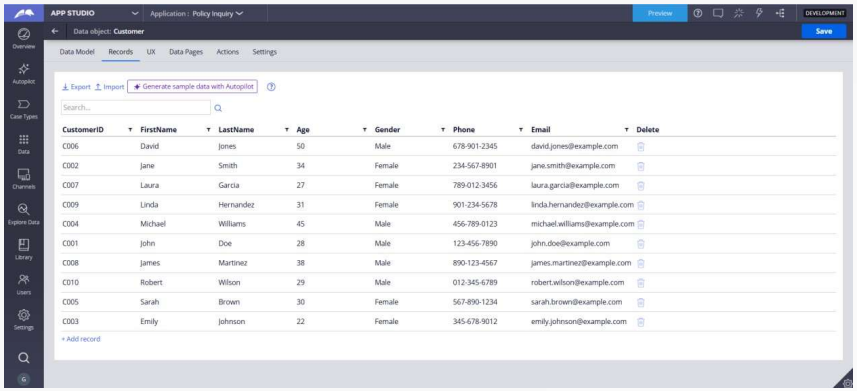
Show fields like Customer ID, Name, Email, Mobile, Address in a form view.

End Result:

User searches for a customer either by Auto Complete or Dropdown.

Selected record gets stored in the Single Page (Customer)

property. Customer details automatically display in a page format.




Step: Create a Single Page Property

Go to Case Type → Data Model.

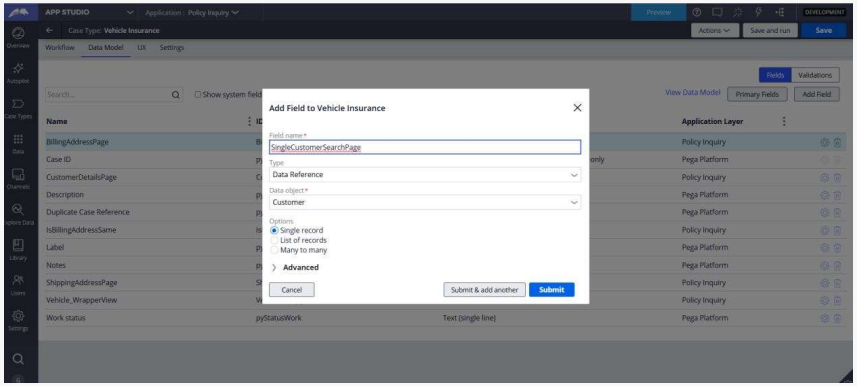
Click + Add Field → choose Single Page.

Provide:

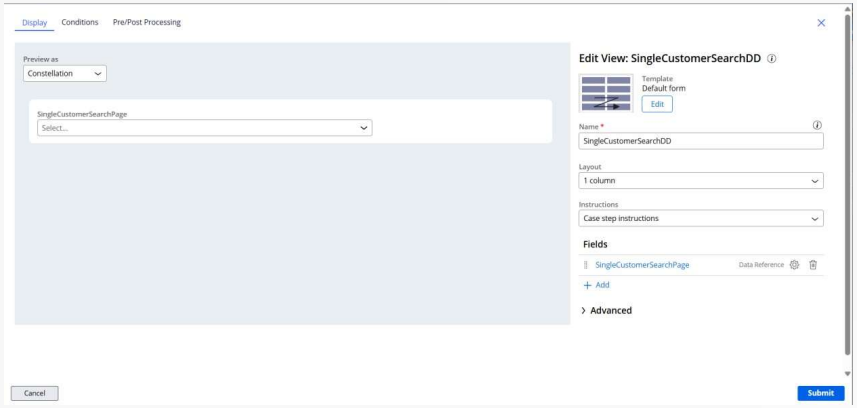
Field Name: Customer

Data object reference: Customer (the Data Type we already created).

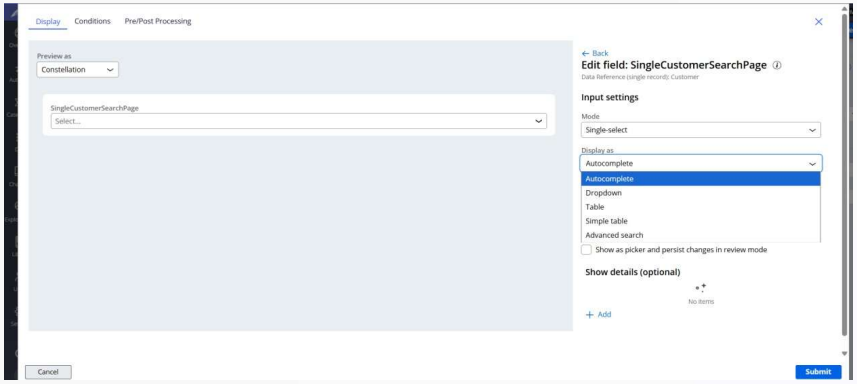
Save the Case Type.



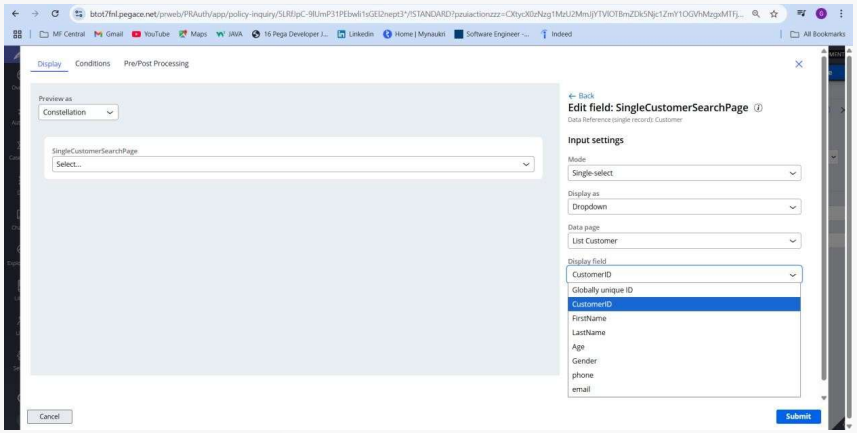

Add an Assignment → Configure the view → Add the page that we created.



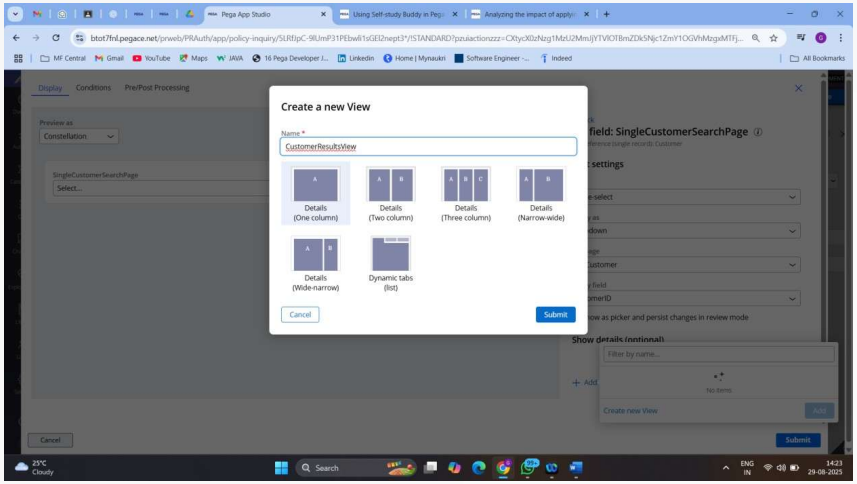

Click on the page → then select the control you want to create, such as Dropdown, Autocomplete, etc.



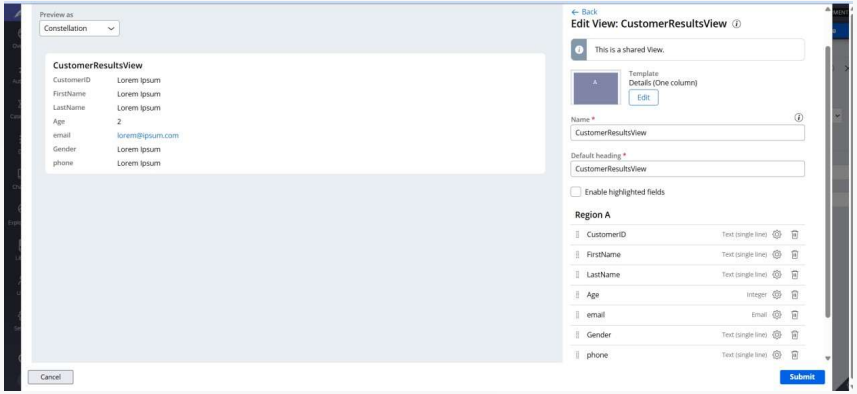
By default, the source of the table is always a List-type Data Page, and it cannot be modified. However, by using the Display Field option, we can choose which field to show. For now, I have selected Customer ID.



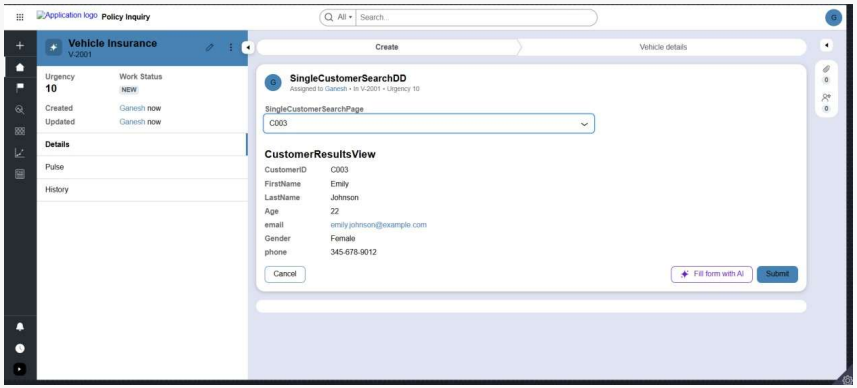

I want to display additional details when a Case ID is selected, so I am adding the view accordingly




On clicking the view, we get the option to add fields that we want to display as part of the Customer ID. I have added the required properties such as First Name, Last Name, etc.




After saving and running, I can see the dropdown where I can select a single value, and based on that, the system retrieves the corresponding details.




Single Select – Table

Use Cases:

Search by only one field – User can search records based on a single field.

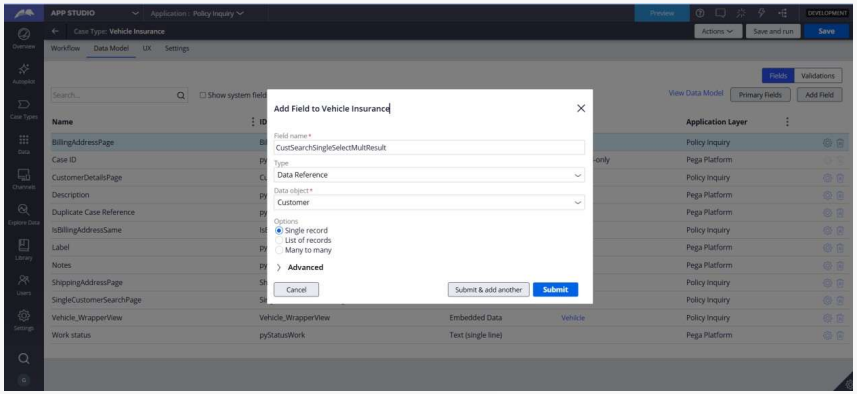
Display all records in a table format – Show all rows in a tabular layout.

Requirement:

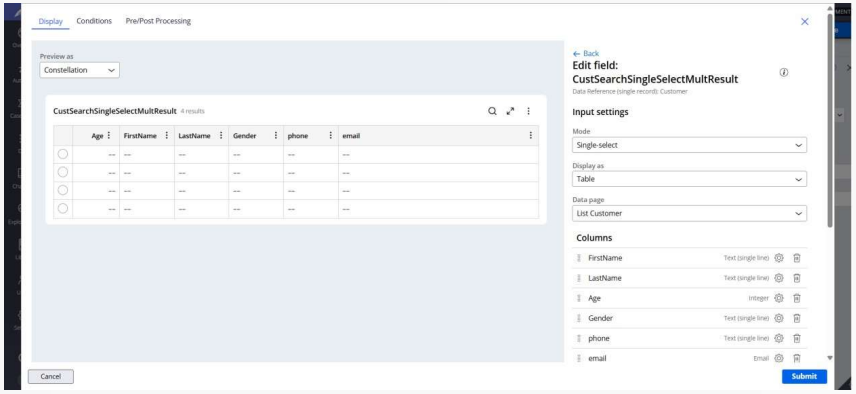
Users should be able to select one row out of the multiple rows displayed in the table.

Design Approach:

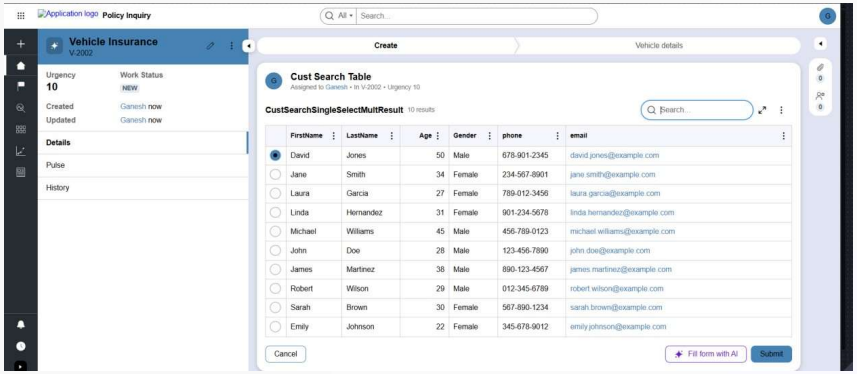
Since the Data Type is already created, the next step is to create a Single Page property in the Case Type to reference the Data Object.



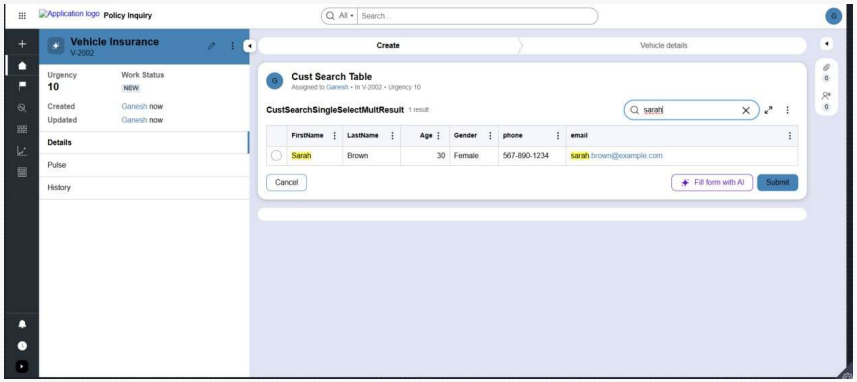

Add Assignment → Configure → Add field → Select the page → Click on the page → Select table → Add columns.



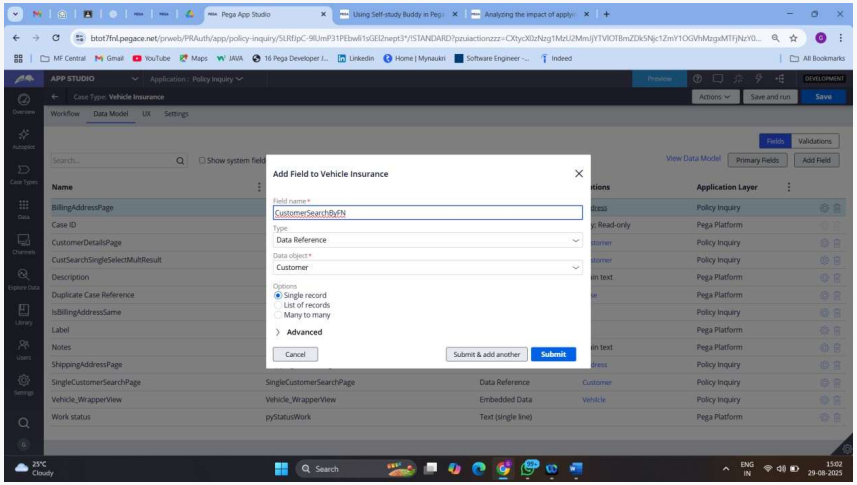

Save and Run



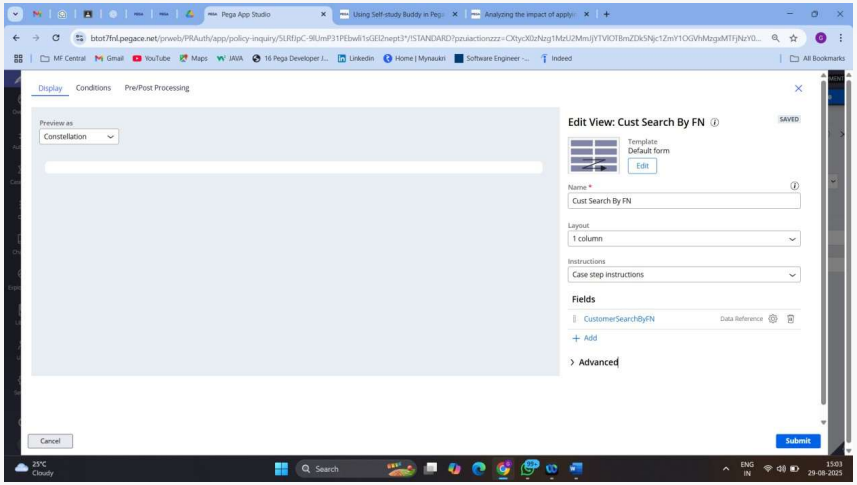
**We can search for any customer by using the search feature.**



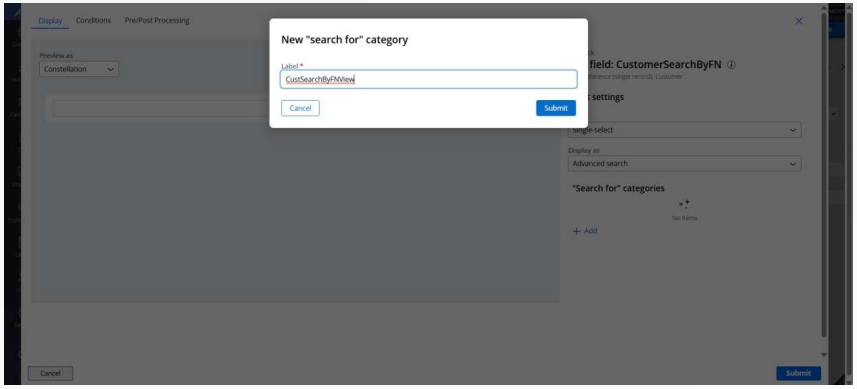

Single Select – Search by one field, display multiple results in a table format, allow the user to select one record, and display the selected record on the next screen.



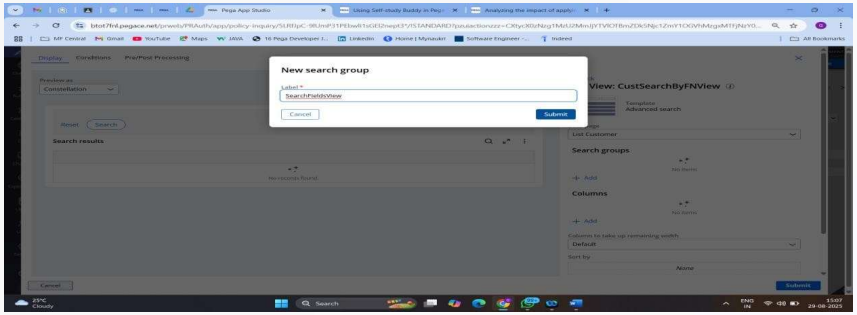

Add Assignment → Configure → Add field → Select the page → Click on the page → Select Advanced Search → Add columns.

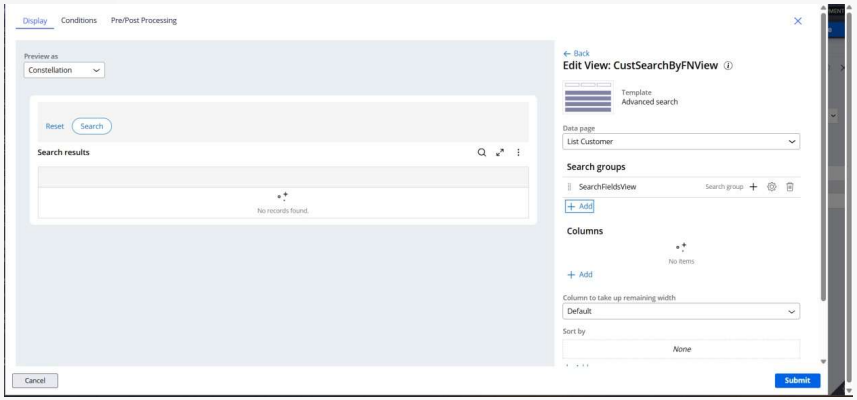



**Click on the page → Select "Display as Advanced Search" → Add search for category.**

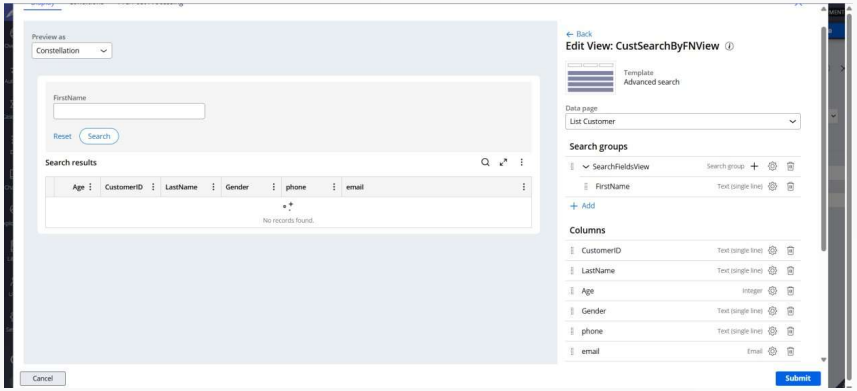



**After adding the view, click on the view.**

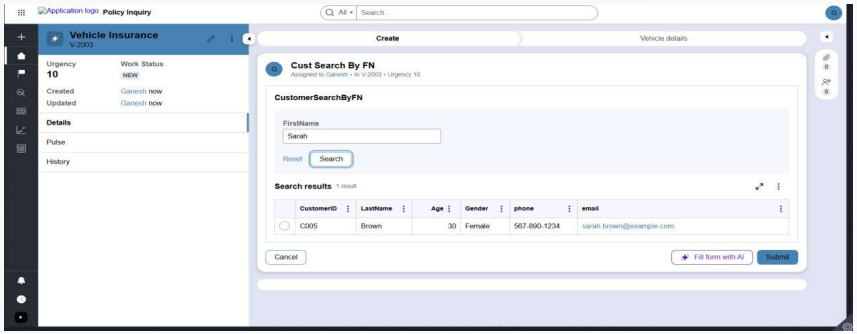





Click on the + icon to select the properties to use for the search. Also, add the columns you want to display after searching by First Name. Configure the view accordingly, as shown in the screenshot.




Save and run.



**Data Collection – Table Format**

**Requirement:**

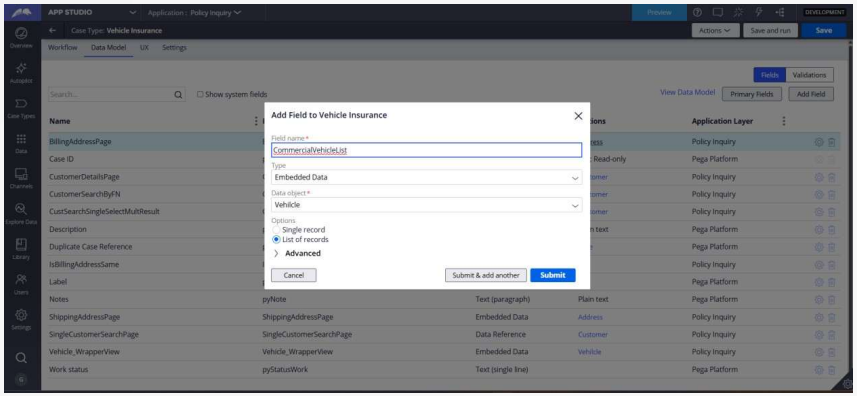
1. Create a new **Data Object** named Vehicle with fields:
    - VIN
    - Registration Number
    - Model
    - Make Year
    - Company
    - …etc.
2. Create a **new screen** to collect details of **multiple vehicles** in a **table format**.
3. Include options to **Add Vehicle** and **Delete Vehicle**.
4. Display the data using **Table View** or **Repeating View**.

---

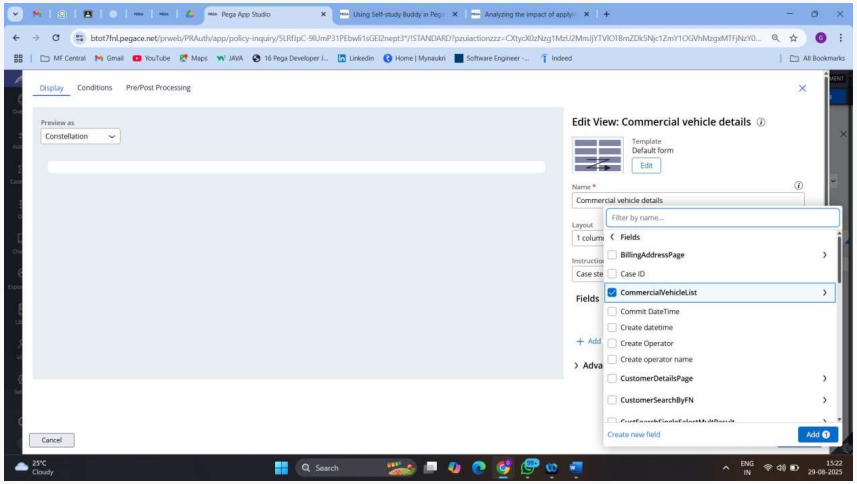
**Design:**

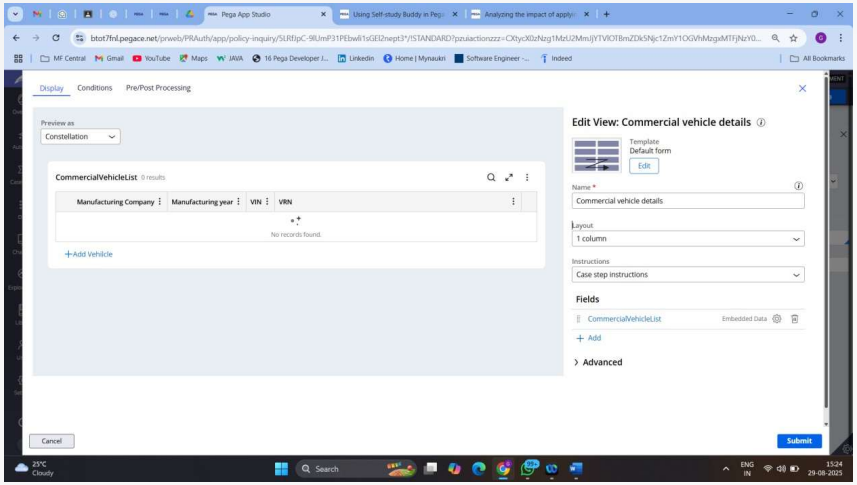
- The **Vehicle Data Object** already exists.
- Use this Data Object as the source for the **Single Page or Page List property** in your case type.
- Configure a **View** in the assignment to display vehicles in **table/repeating view** format.
- Add **actions** for adding and deleting rows as per the requirement.

Open Case Type → Data Model → Add Page List.

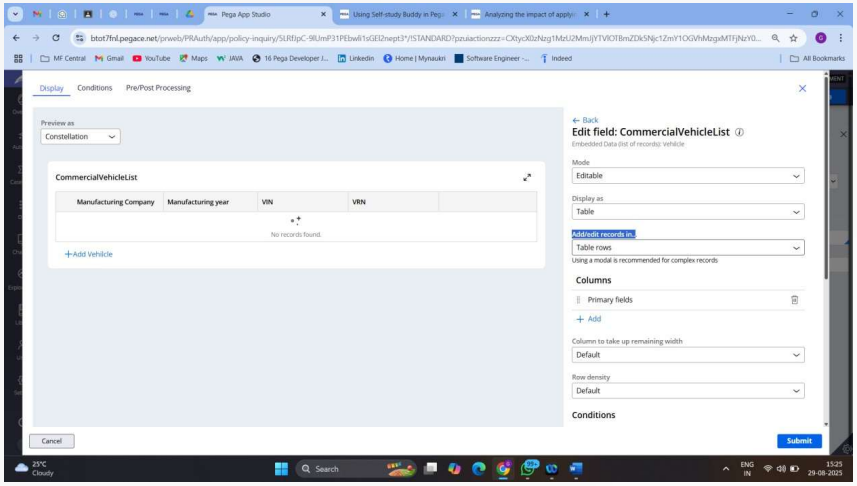



Add Assignment→Configure view→Select page list

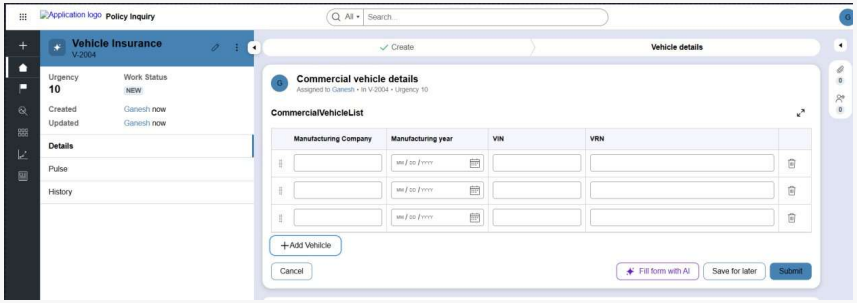





Click on the Page List → Select "Add/Edit Records in" → Choose "Table Rows".




Save and Run




**Working with Page List – Data Reference (Multi-Select)**

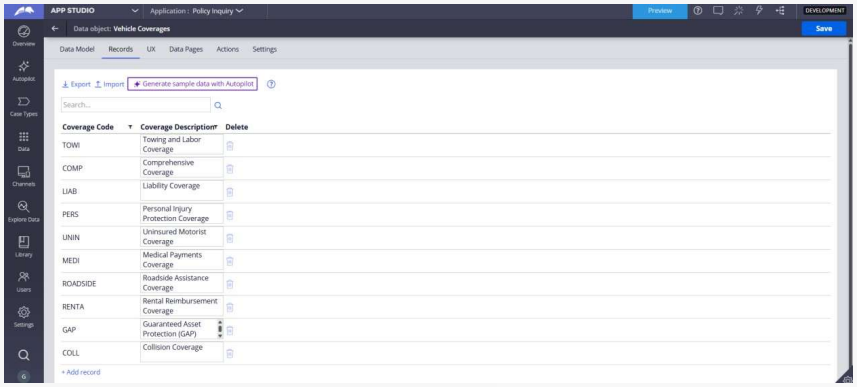
**Requirement:**

1. Create a new **Data Object** named Vehicle Coverages.
2. Add some sample records into the Data Object.
3. Create a **screen** to load Vehicle Coverage records in a **table format**.
    - Include a **select option** for each record.
    - Include a **Select All** option to select multiple records at once.

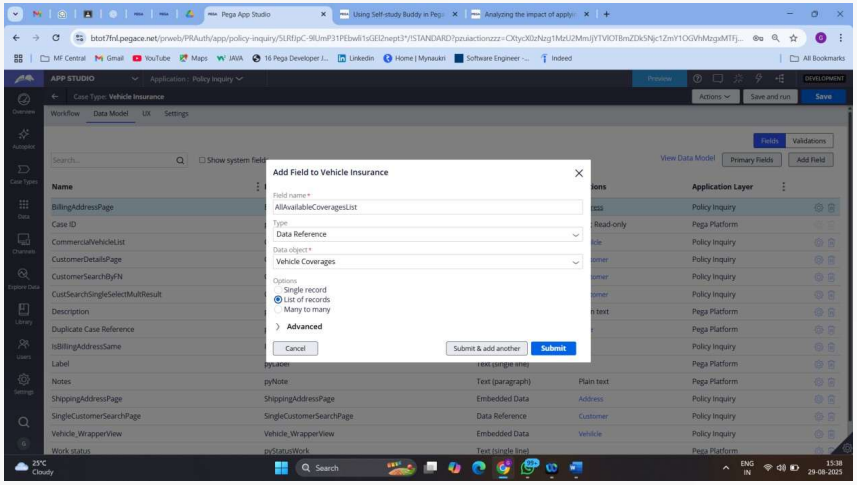
---

**Design:**

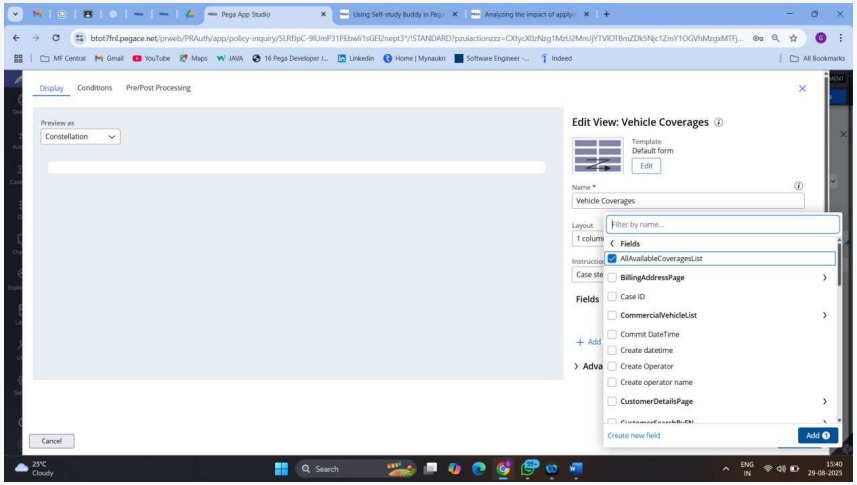
- The **Data Object** Vehicle Coverages has been created.
- Properties and sample data have been added.
- Next step: Use a **Page List property** in the Case Type to reference multiple Vehicle Coverage records.
- Configure the **View** to display the data in a **table format** with multi-select functionality.



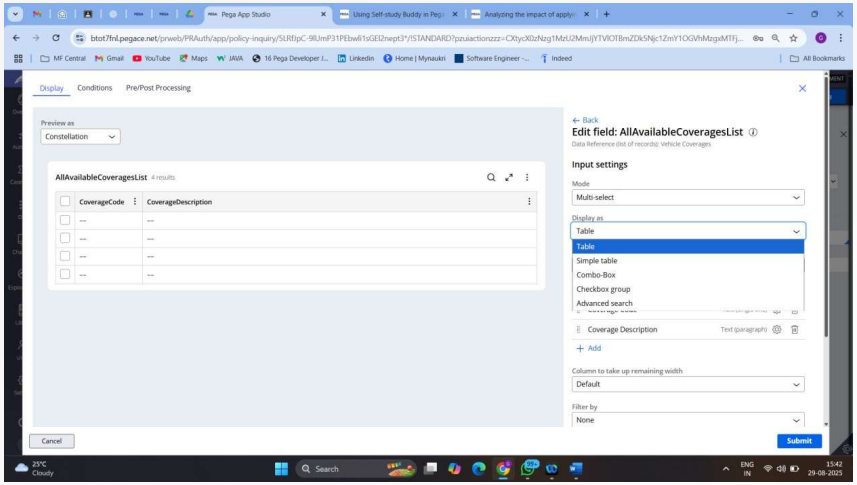

Create Page list

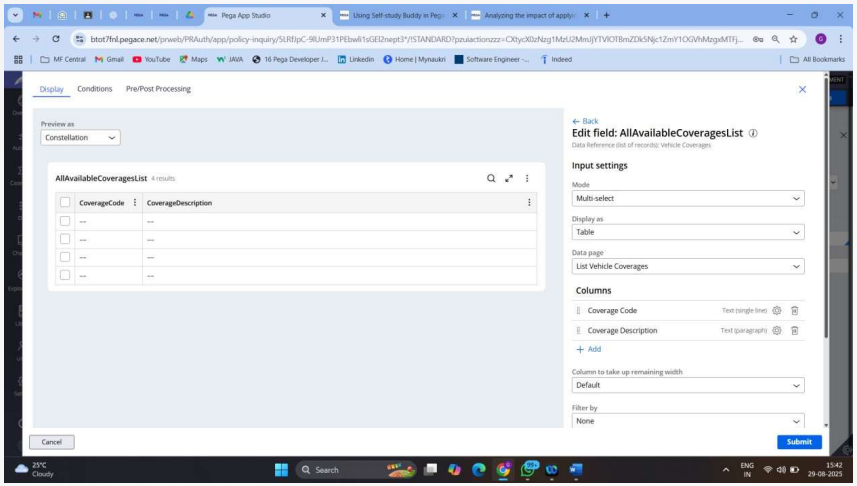



Add step→Configure view→Add page list

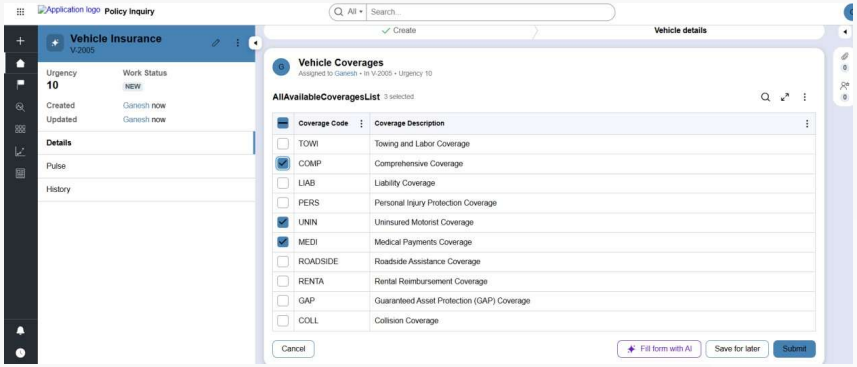



**Click on the list → Add the columns you want to display. You can see the structure as well.**




Display
Conditions
Pre/Post Processing
Preview as
Constellation
AllAvailableCoveragesList 4 results
CoverageCode
CoverageDescription
Back
Edit field: AllAvailableCoveragesList
Input settings
Mode
Multi-select
Display as
Table
Data page
List Vehicle Coverages
Columns
Coverage Code
Coverage Description
Add
Column to take up remaining width
Default
Filter by
None
Cancel
Submit

Save and Run


Application logo
Policy Inquiry
Search...
Vehicle Insurance
V-2005
Urgency
10
Work Status
NEW
Created
Ganesh now
Updated
Ganesh now
Details
Pulse
History
Create
Vehicle details
Vehicle Coverages
AllAvailableCoveragesList 3 selected
Coverage Code
Coverage Description
TOWI
Towing and Labor Coverage
COMP
Comprehensive Coverage
LIAB
Liability Coverage
PERS
Personal Injury Protection Coverage
UNIN
Uninsured Motorist Coverage
MEDI
Medical Payments Coverage
ROADSIDE
Roadside Assistance Coverage
RENTA
Rental Reimbursement Coverage
GAP
Guaranteed Asset Protection (GAP) Coverage
COLL
Collision Coverage
Cancel
Fill form with AI
Save for later
Submit

# Working with Attachments

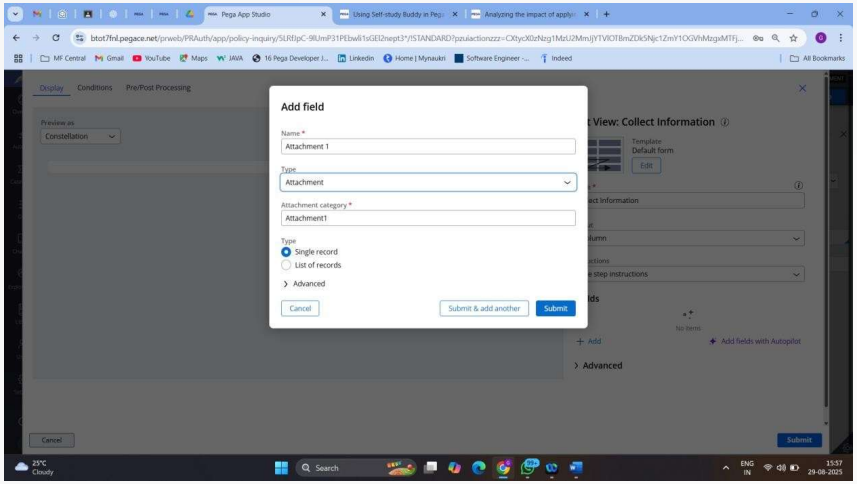
**Requirement:**

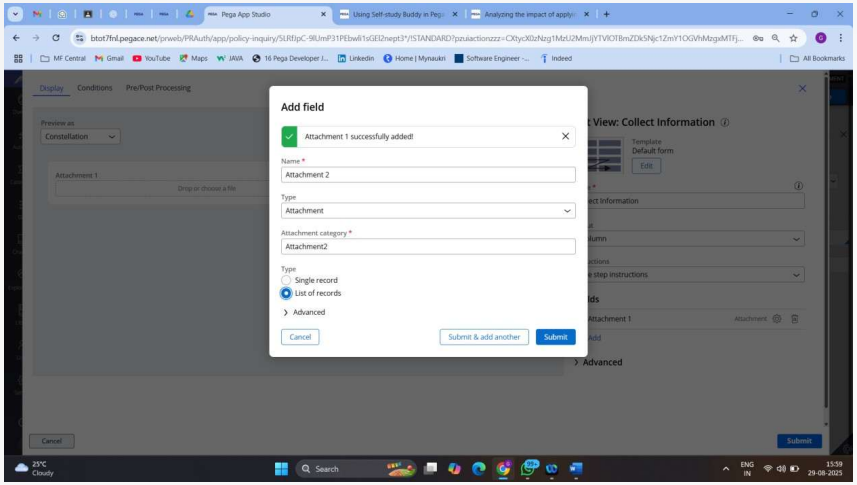
Create a new screen that allows users to attach documents to a Work Object.

---

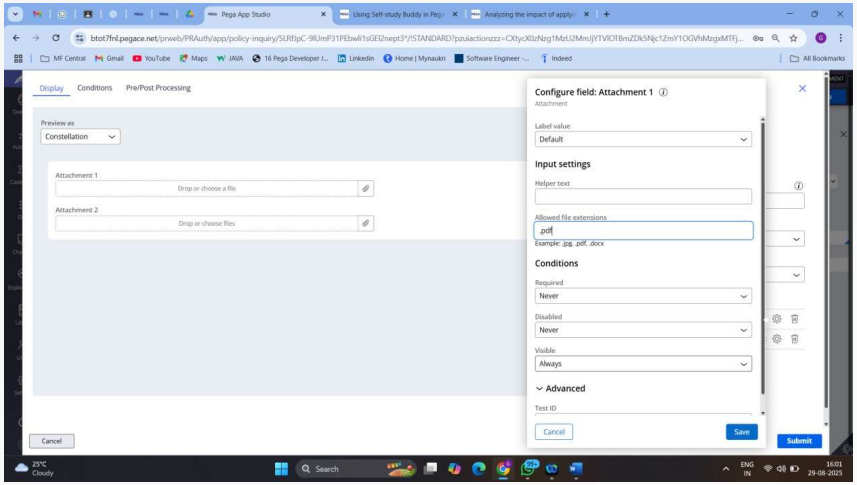
**Design:**

- Attachments can be implemented as:
    1. **Single File** → Use a **Single Page property**
    2. **Multiple Files** → Use a **Page List property**
- Steps Taken:
    1. Added a **new stage** in the case type.
    2. Configured the **View** for the stage.
    3. Added two properties:
        - **Single Page** (for single file attachment)
        - **Page List** (for multiple file attachments)

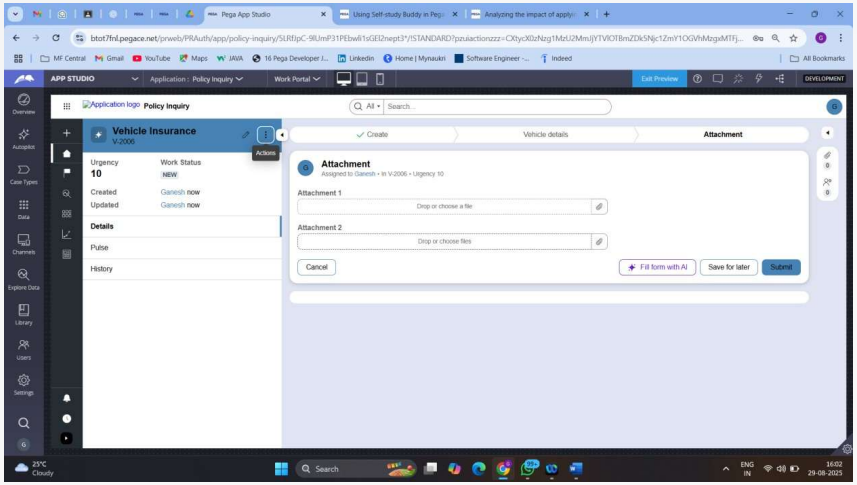




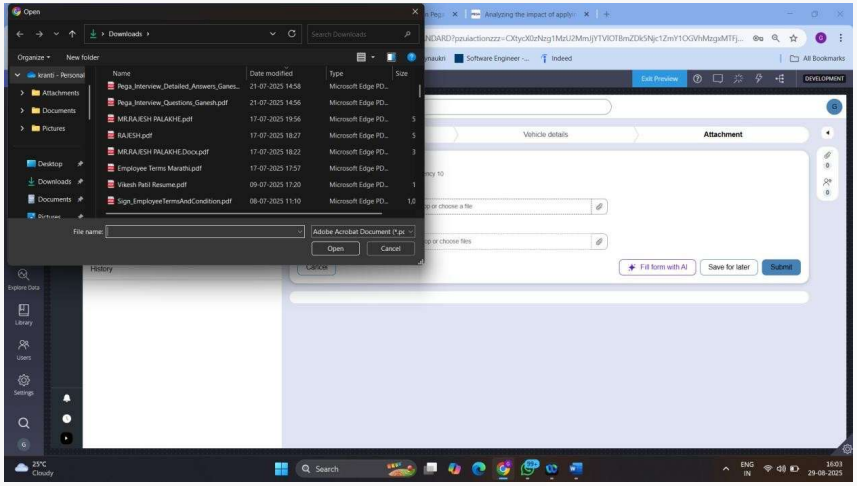

Click on the gear icon



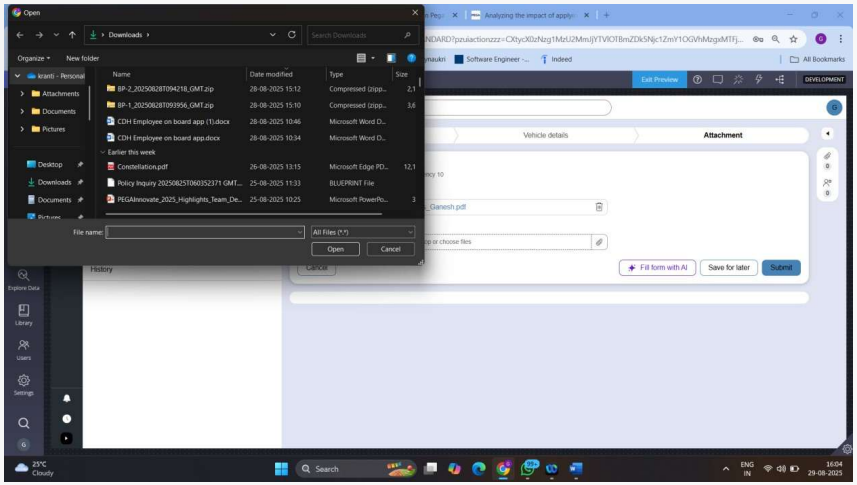

Save and run



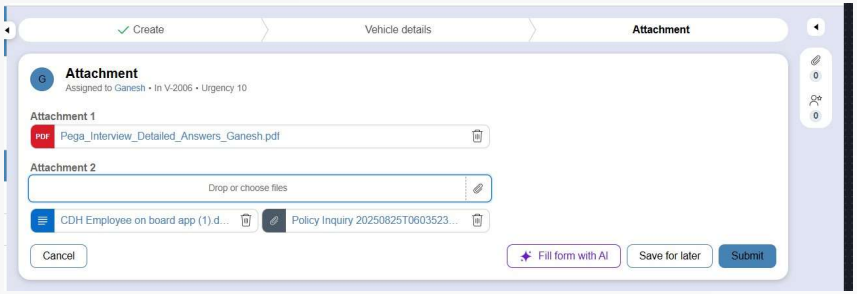

**"Since I have selected PDF as the allowed format, I am able to see only PDF files."**



Let’s check the Attachment Page List — we can see all file formats since we haven’t restricted it like we did for the single attachment.




As shown in the screenshot, after adding an attachment for the Single Page, we will not see the option to add another file. However, this option is available for Page List type attachments.



# Working with Theme

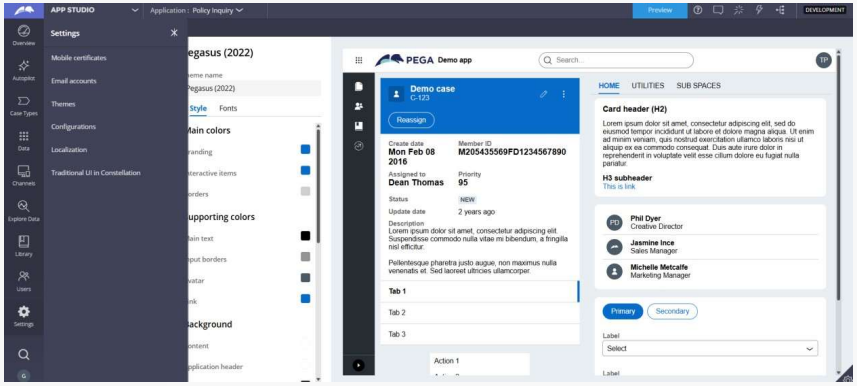
**Requirement:**

Create a new theme and apply it to the Case Type.

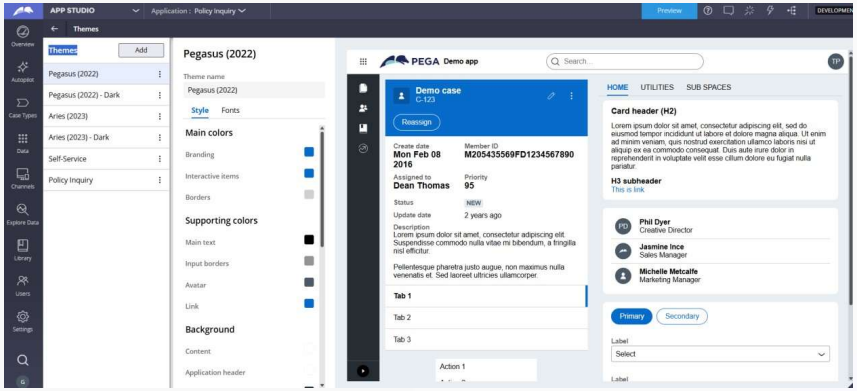
---

**Design:**

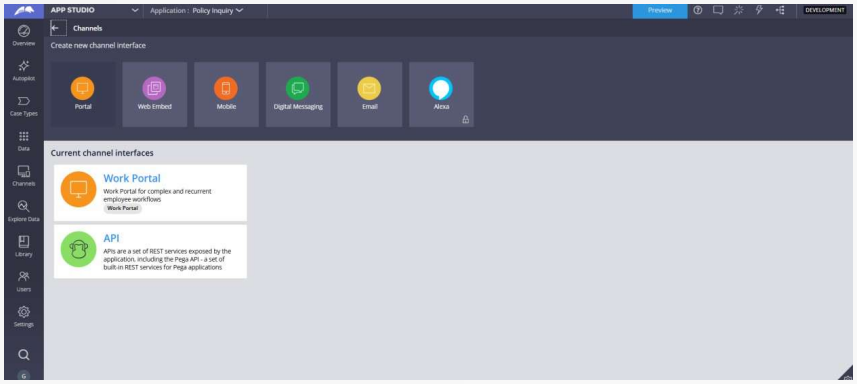
- Navigate to **App Studio → Settings → Themes**.
- Create a **new theme** and customize colours, fonts, and styling as needed.
- Apply the theme to the desired **Case Type** to reflect the changes in the UI.



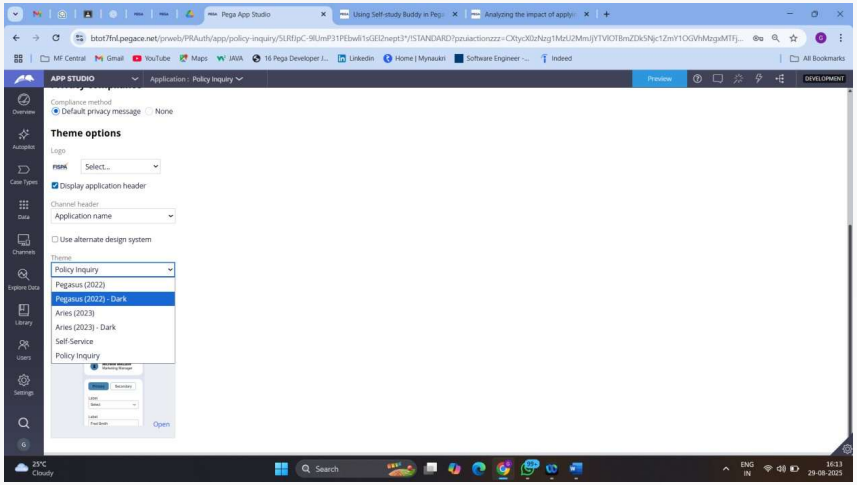

If we want, we can add a new theme, or we can view and use the current theme, which is displayed by its theme name.



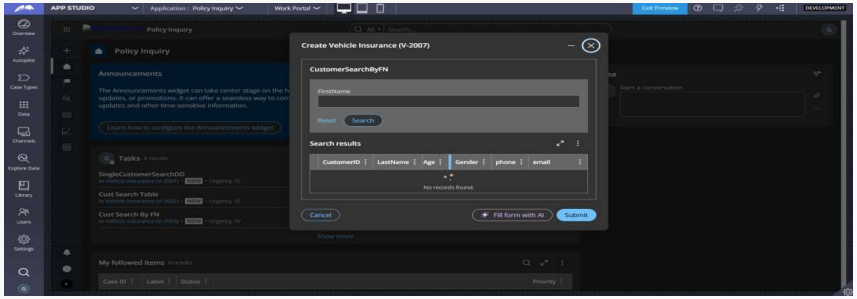
Let’s explore more → Left Navigation → Channels → Work Portal.




From the Theme option, we can change the theme.




Since I have selected the Dark theme, let’s check how it behaves.



# DX API

**DX API Overview**

**What is DX API?**

- **DX API (Digital Experience API)** is a Pega framework that allows external systems or applications to **interact with Pega applications** programmatically.
- It provides **RESTful endpoints** to **create, update, retrieve, and manage case data** or execute other actions in Pega.

---

**Key Features:**

1. **Access Pega Cases** – Start new cases, update existing cases, and retrieve case information.
2. **Integration Ready** – Supports **JSON payloads** for easy integration with web or mobile applications.
3. **Security & Authentication** – Uses **OAuth 2.0, API keys, or JWT tokens** to secure endpoints.
4. **Extensible** – Can expose custom rules or actions as APIs for external consumption.
5. **Monitoring & Logging** – Built-in logging to track API requests and responses.

---

**Use Cases:**

- **Mobile apps** can create or update cases directly in Pega.
- **Third-party systems** can query customer or employee data stored in Pega.
- **Automated workflows** that require data synchronization across systems.

---

**Navigation in Pega:**

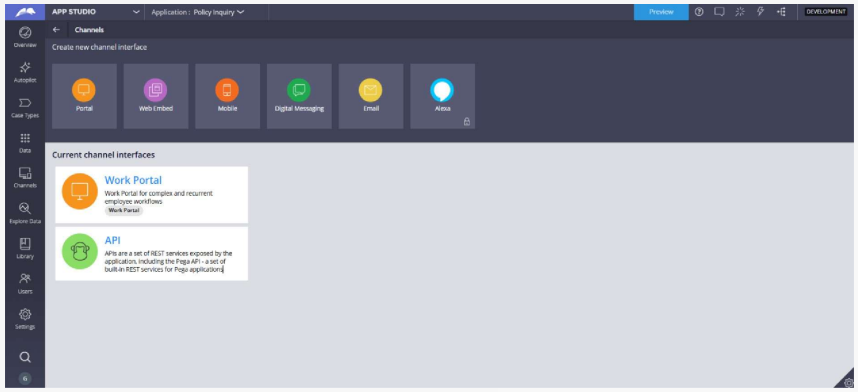
- **App Studio / Dev Studio → Channels → API**
- From here, you can **explore existing DX APIs**, create new endpoints, and test them.

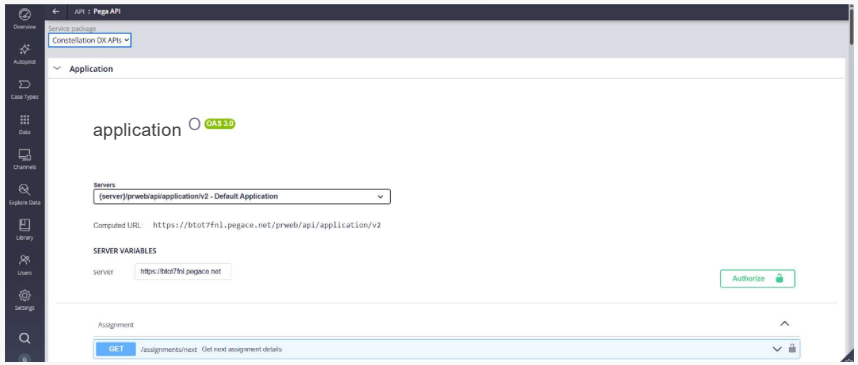
---

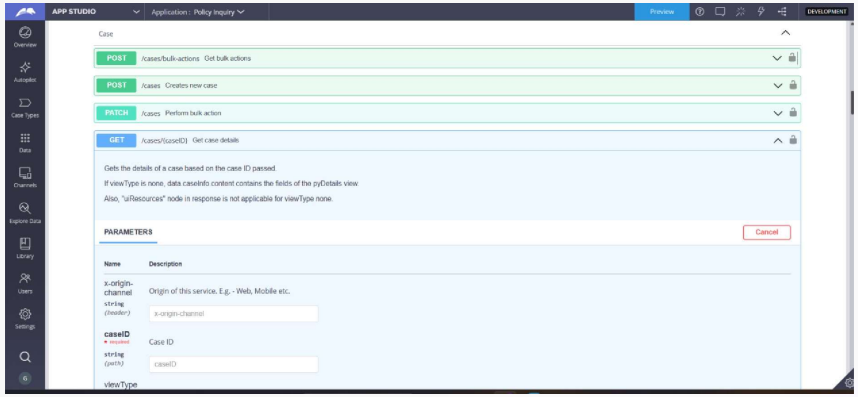
**Key Takeaway:**

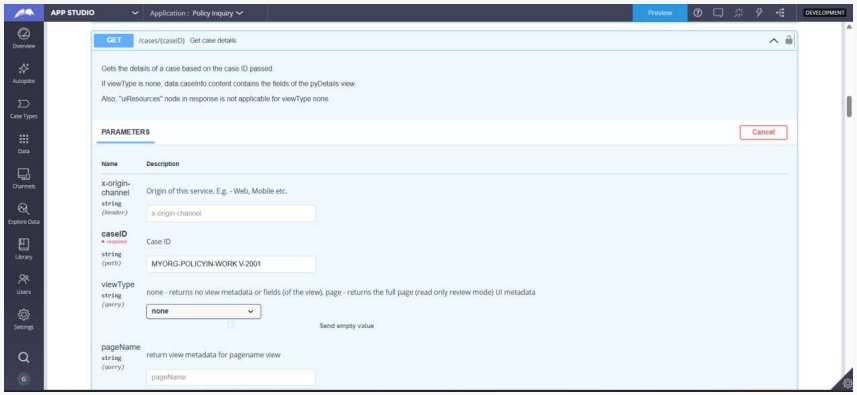
DX API enables **seamless integration** between Pega and external applications, making processes **more efficient, automated, and accessible**.

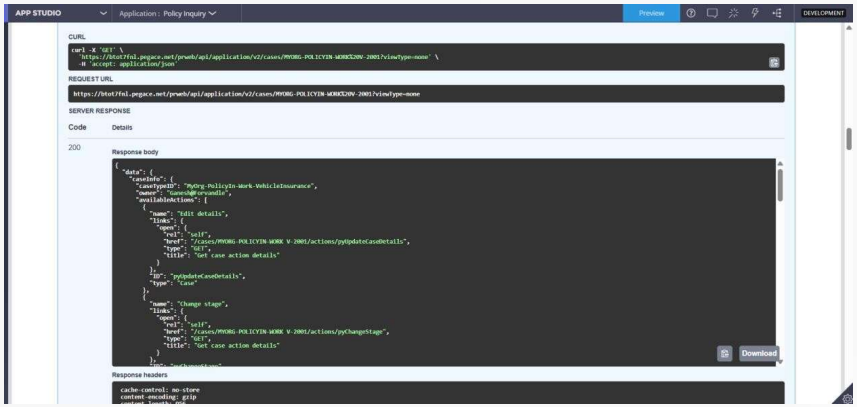
forvandle tech bytes







```
{
 "data":{
  "caseInfo": {
   "caseTypeID": "MyOrg-PolicyIn-Work-VehicleInsurance",
   "owner": "Ganesh@Forvandle",
   "availableActions": [
    {
     "name": "Edit details",
     "links": {
      "open": {
```

```
      "rel": "self",
      "href": "/cases/MYORG-POLICYIN-WORK V-2001/actions/pyUpdateCaseDetails",
      "type": "GET",
      "title": "Get case action details"
     }
    },
    "ID": "pyUpdateCaseDetails",
    "type": "Case"
   },
   {
    "name": "Change stage",
    "links": {
     "open": {
      "rel": "self",
      "href": "/cases/MYORG-POLICYIN-WORK V-2001/actions/pyChangeStage",
      "type": "GET",
      "title": "Get case action details"
     }
    },
    "ID": "pyChangeStage",
    "type": "Case"
   }
  ],
  "associations": {},
  "lastUpdatedBy": "Ganesh@Forvandle",
  "assignments": [
   {
    "instructions": "",
    "canPerform": "true",
    "assigneeInfo": {
     "name": "Ganesh",
```

```
  "ID": "Ganesh@Forvandle",
  "type": "worklist"
 },
 "processID": "Create_Flow_1",
 "urgency": "10",
 "processName": "Create (1)",
 "isMultiStep": "false",
 "name": "Address Details",
 "context": "",
 "links": {
  "open": {
   "rel": "self",
   "href": "/assignments/ASSIGN-WORKLIST MYORG-POLICYIN-WORK V-2001!CREATE_FLOW_1",
   "type": "GET",
   "title": "Get assignment details"
  }
 },
 "ID": "ASSIGN-WORKLIST MYORG-POLICYIN-WORK V-2001!CREATE_FLOW_1",
 "actions": [
  {
   "name": "Address Details",
   "links": {
    "save": {
     "rel": "self",
     "href": "/assignments/ASSIGN-WORKLIST MYORG-POLICYIN-WORK V-2001!CREATE_FLOW_1/actions/AddressDetails/save",
     "type": "PATCH",
     "title": "Save assignment action "
    },
    "open": {
     "rel": "self",
```

```
            "href": "/assignments/ASSIGN-WORKLIST MYORG-POLICYIN-WORK V-2001!CREATE_FLOW_1/actions/AddressDetails",
            "type": "GET",
            "title": "Get assignment action details"
          }
        },
        "ID": "AddressDetails",
        "type": "FlowAction"
      }
    ]
  }
],
"hasNewAttachments": false,
"businessID": "V-2001",
"sla": {
  "goal": "",
  "deadline": ""
},
"WidgetsToRefresh": [],
"caseTypeName": "Vehicle Insurance",
"urgency": "10",
"createTime": "2025-08-29T08:58:10.352Z",
"createdBy": "Ganesh@Forvandle",
"name": "Vehicle Insurance",
"stages": [
 {
   "entryTime": "2025-08-29T08:58:10.484Z",
   "name": "Create",
   "links": {
     "open": {
       "rel": "self",
```

```
      "href": "/cases/MYORG-POLICYIN-WORK V-2001/stages/PRIM0",
      "type": "PUT",
      "title": "Jump to this stage"
    }
  },
  "visited_status": "active",
  "ID": "PRIM0",
  "type": "Primary",
  "transitionType": "create"
},
{
  "name": "Vehicle details",
  "links": {
    "open": {
      "rel": "self",
      "href": "/cases/MYORG-POLICYIN-WORK V-2001/stages/PRIM4",
      "type": "PUT",
      "title": "Jump to this stage"
    }
  },
  "visited_status": "future",
  "ID": "PRIM4",
  "type": "Primary",
  "transitionType": "resolution"
},
{
  "name": "Attachment",
  "links": {
    "open": {
      "rel": "self",
      "href": "/cases/MYORG-POLICYIN-WORK V-2001/stages/PRIM6",
```

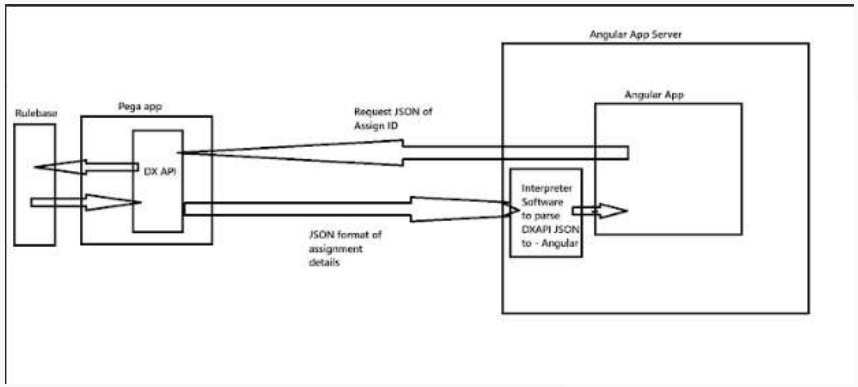
```
      "type": "PUT",
      "title": "Jump to this stage"
     }
    },
    "visited_status": "future",
    "ID": "PRIM6",
    "type": "Primary",
    "transitionType": "automatic"
  }
],
"ID": "MYORG-POLICYIN-WORK V-2001",
"caseTypeIcon": "cmicons/pycase.svg",
"status": "New",
"stageID": "PRIM0",
"stageLabel": "Create",
"lastUpdateTime": "2025-08-29T11:03:31.298Z",
"content": {
  "classID": "MyOrg-PolicyIn-Work-VehicleInsurance",
  "pxObjClass": "MyOrg-PolicyIn-Work-VehicleInsurance",
  "pyLabel": "Vehicle Insurance",
  "pyID": "V-2001",
  "pyViewName": "",
  "pyViewContext": "",
  "pxUrgencyWork": 10,
  "pxCreateOperator": "Ganesh@Forvandle",
  "pxUpdateDateTime": "2025-08-29T11:03:31.298Z",
  "pxUpdateOperator": "Ganesh@Forvandle",
  "pyStatusWork": "New",
  "pxCreateDateTime": "2025-08-29T08:58:10.352Z",
  "pyCaseLinks": []
}
```

```
  },
  "referencedUsers": {
    "Ganesh@Forvandle": {
      "UserName": "Ganesh"
    }
  }
 }
}
```


Flow diagram of communicating external system with DX API

**Explanation of Steps:**

1. **External System** sends a request to Pega DX API with required data.
2. **DX API Endpoint** authenticates the request using security tokens.
3. Pega executes the **case, data update, or any configured rule** based on the API call.
4. The API returns a **JSON response** to the external system with success/failure or requested data.



forvandle
tech bytes

# THANK YOU!

www.forvandle.com