**PEGA**

**Pega Overview**

* Pega is a **low-code platform** that provides an application development environment which allows us to create digital solutions to achieve business outcomes.
* **Low-code application development** can be treated as a type of application development that uses visual, declarative techniques instead of programming.
* Low-code platforms provide us a **visual development environment** with **drag-and-drop** capabilities.
* Pega is a well-known **Business Process Management (BPM)** tool.
* It helps organizations to build applications that capture data and initiate business processes.

**Pega History**

* Founded in **1983 by Alan Trefler**, Pegasystems Inc. (Pega) is a global software company based in Cambridge, Massachusetts
* Initially focused on providing case management solutions, Pega has evolved to offer a comprehensive platform for workflow automation and AI-powered decisioning.
* Pegasystems develops software for **customer relationship management (CRM) and business process management (BPM).**

**Why Pega?**

* **Efficiency**: Pega’s low-code platform allows for rapid development and deployment of applications, reducing time-to-market.
* **Flexibility**: The platform supports a wide range of use cases across different industries, from customer service to supply chain management.
* **Integration**: Pega seamlessly integrates with existing systems, enhancing their capabilities through automation and AI.

**Jio Use Case: Transforming SIM Approval Process**

**Problem**

* **Inefficient SIM approval process** with multiple manual steps causing delays and errors.

**Solution**

1. **Digital Process Automation (DPA)**:
   * Automated end-to-end workflow.
   * Integrated with existing systems for real-time updates.
2. **Robotic Process Automation (RPA)**:
   * Automated repetitive tasks like data entry and document verification.
3. **Business Process Management (BPM)**:
   * Optimized and monitored the approval process for compliance and efficiency.
4. **Artificial Intelligence (AI)**:
   * Verified documents and predicted approval outcomes.

**Outcome**

* **Faster approvals**, improved accuracy, and reduced operational costs.

**Tying It Back to Pega**

* **Pega DPA**: Automates workflows and integrates systems.
* **Pega RPA**: Handles repetitive tasks efficiently.
* **Pega BPM**: Manages and optimizes processes.
* **Pega AI**: Enhances decision-making and personalization.

Pega's platform can deliver similar benefits as Jio's transformation, ensuring efficiency and customer satisfaction.

**Pega Studios**

Pega Platform provides four role-based authoring workspaces, known as studios. Each studio is designed to enhance productivity by providing tools and features tailored to specific roles within an organization:

***App Studio***

A low-code workspace for business users and developers to design applications quickly.

*Example Use Case:* A business analyst uses App Studio to create a customer service application with drag-and-drop tools, allowing customer service representatives to track and resolve customer issues efficiently.

***Dev Studio***

An advanced workspace for technical developers to build complex and customized applications.

*Example Use Case:* A system architect uses Dev Studio to integrate the customer service application with the company's existing CRM system, adding custom logic and data flows.

***Prediction Studio***

A workspace for data scientists to create and manage predictive models and analytics.

*Example Use Case:* A data scientist uses Prediction Studio to develop a model that predicts customer churn, helping the company identify at-risk customers and take proactive measures to retain them.

***Admin Studio***

A workspace for IT administrators to manage and monitor the Pega environment.

*Example Use Case:* An IT administrator uses Admin Studio to monitor system performance, manage user access, and ensure the application runs smoothly without any downtime.

**Generative AI in Pega**

**Generative AI in Pega** refers to the use of AI technologies to enhance application development and decision-making processes. Pega GenAI integrates AI-powered features across the Pega Platform to boost productivity and creativity.

* **Key Capabilities**:
  + **AI-Generated Low-Code**: Automatically generate workflows and integrations.
  + **Conversational Suggestions**: Provide real-time development tips and guidance.
  + **Test Data Generation**: Create test data on the fly to validate applications.

**Integrating with External Applications**

**Integrating with external applications** in Pega involves connecting Pega applications with other systems to exchange data and functionality. Pega supports various integration methods, including REST APIs, SOAP services, and robotic process automation (RPA).

* **Integration Methods**:
  + **REST APIs**: Connect to external systems using RESTful web services.
  + **SOAP Services**: Use SOAP protocols for data exchange.
  + **RPA**: Automate interactions with systems that lack APIs.

**Pega Constellation Design System**

**Pega Constellation Design System** is a comprehensive design framework that provides a consistent user experience across Pega applications. It includes a set of design rules, patterns, and components that help streamline the development process.

A design system is a builder kit that includes a shared library of:

* UI elements
* Styles
* UX patterns comprised of components
* Themes
* Guidelines

**Core Features**:

* + - **UI Elements**: Predefined user interface elements like buttons, forms, and modals.
    - **UX Patterns**: Standardized user experience patterns to ensure consistency.
    - **Themes**: Customizable themes to match branding requirements.

***Prescribed Design in Constellation***

**Prescribed Design in Constellation** refers to a set of informed default designs and templates with preset configurations. This approach reduces design time and ensures consistency across applications.

**Key Concepts**:

* + - **Design Tokens**: Standardized design elements like colors, spacing, and typography.
    - **Templates**: Prebuilt page layouts and components.
    - **Guidelines**: Best practices for using design elements effectively.

**Microjourneys**

* A **Microjourney** is a small part of the overall customer journey and focuses on accomplishing a specific goal.
* Microjourneys are designed to be delivered quickly, often within 60-90 days, making them ideal for iterative development and continuous improvement.
* ***Example:*** A customer wants to change their address. The Microjourney includes steps like verifying identity, updating the address in the system, and sending a confirmation notification.

**Personas and Channels**

* **Personas** determine who interacts with the application. **Channels** determine how a Persona interacts with the application.
* Defining Personas helps tailor the user experience to specific roles, ensuring that each user interacts with the application in a way that meets their needs and preferences
* ***Example:*** A Persona is a customer or a company employee, and a Channel is a Web Portal or a chatbot.

**Data and Interfaces**

* **Data** is the information that the Microjourney interacts with to accomplish the customer's goal, and the interface defines where the data comes from or where it has persisted.
* Effective data management and integration with external systems are crucial for providing a seamless and efficient customer experience.
* ***Example:*** In a customer service application, data might include customer details and service history, while interfaces could be APIs that connect to external CRM systems

**Case Types and Cases**

* A **Case Type** is an abstract model of a business transaction. A **Case** is a specific transaction instance.
* **Case Types** help standardize processes, making it easier to manage and track business transactions.
* ***Example:*** In an online order system, the Case Type could be "Order Processing," and each order placed by a customer would be a Case.

**Case Life Cycle**

The **Case Life Cycle** provides a clear visual representation of the workflow, making it easier to manage and optimize business process.

You define the **Case Life Cycle** for a Case Type to help you visualize the work that must be completed as part of the desired business transaction. The major building blocks of the Case Life Cycle are **Stages, Processes, Steps**:

***Stages:*** Represent the Case transfer from one caseworker to another or a significant change in the status of the Case. When you design a Case Life Cycle, you begin by organizing work into Stages.

***Processes:*** Contain a series of Tasks, or Steps, that users complete as they work on the case. Each Stage can contain one or more Processes.

***Steps:*** A Step within a Process is either a user Action or an automated Action. Automated Actions, also referred to as Automation Steps, are performed by the system

***Example:*** In an employee onboarding process, Stages might include "Application Review," "Interview," and "Hiring." Each Stage would have Processes like "Schedule Interview" and "Send Offer Letter," with Steps such as "Review Application" and "Send Email".

**Data Modeling**

* A **Data Model** is the visual representation of all the data elements of an organization and the connections between them.
* The main purpose of the Data Model is to define the data that your application requires to achieve the business outcome.
* **Components**:
  + **Fields**: Properties that store and format data.
  + **Data Objects**: Categories of data that include fields, field mappings, and connections to data sources.
* **Example**: In a library management system, data objects might include "Books," "Authors," and "Patrons," with fields like "Title," "Author Name," and "Library Card Number."

**Types of Data Models in Pega**

**Conceptual Data Model**

* The conceptual data model is a high-level representation of the data entities and their relationships. It focuses on the business perspective and is used for initial discussions with stakeholders.
* It is used to understand the data requirements and relationships at a high level.
* **Components:** Entities and relationships.
* ***Example:*** In a library management system, the conceptual data model might include entities like "Books," "Authors," and "Members," with relationships such as "Books are written by Authors" and "Members borrow Books."
* ***When to Use:*** Use the conceptual data model during the early stages of a project to gather requirements and ensure all stakeholders have a common understanding of the data needs.

**Logical Data Model**

* The logical data model refines the conceptual data model by adding more detail. It defines the data objects, attributes, and relationships in a way that is independent of any specific database or technology.
* It provides a detailed blueprint of the data structure that can be used for database design.
* **Components:** Data objects, attributes, and relationships.
* ***Example:*** Continuing with the library management system, the logical data model might define the "Books" data object with attributes like "Title," "ISBN," and "Publication Date," and the "Authors" data object with attributes like "Name" and "Biography."
* ***When to Use:*** Use the logical data model after the conceptual model to provide a detailed and technology-agnostic view of the data structure. This model is useful for database designers and developers.

**Physical Data Model**

* The physical data model translates the logical data model into a specific database implementation. It includes details about how data is stored, indexed, and accessed.
* It defines the actual database schema, including tables, columns, data types, and constraints.
* ***Components:*** Tables, columns, data types, indexes, and constraints.
* ***Example:*** For the library management system, the physical data model might define a "Books" table with columns like "Title" (VARCHAR), "ISBN" (VARCHAR), and "Publication\_Date" (DATE), and an "Authors" table with columns like "Name" (VARCHAR) and "Biography" (TEXT).
* ***When to Use:*** Use the physical data model during the implementation phase to create the actual database schema. This model is essential for database administrators and developers who will build and maintain the database.

**Data Records**

**Data Records** are instances of data objects that contain specific values for each field.

* **Definition**: A unique collection of fields and values that describe an instance of a data object.
* **Storage**: When stored locally, each data record corresponds to a row in a database table.
* **Usage**: Data records can be referenced in forms or views to provide consistent and valid data options.
* **Example**: In a customer service application, a "Customer" data object might have data records with fields like "First Name," "Last Name," "Email," and "Phone Number." Each customer record contains specific values for these fields.

**Data Pages and the Visual Data Model**

**Data Pages** in Pega provide links between data records and the application, defining how data is sourced and used.

* **Definition**: Data pages are used to retrieve and store data for use in the application.
* **Types**: Data pages can be read-only, editable, or list-based, depending on the use case.

**Visual Data Model** is a diagram that shows the relationships between data objects, data pages, and the application.

* **Features**:
  + **Visualization**: Provides a clear view of how data entities are related.
  + **Interaction**: Allows users to update, add, and delete fields directly from the visual model.
* **Example**: In an online order system, a data page might retrieve customer information from an external CRM system. The visual data model would show the connection between the "Customer" data object and the external system.

**Integration Designer**

**Integration Designer** in Pega provides a centralized view of all data objects, data pages, object dependencies, and external systems in an application.

* **Features**:
  + **Integration Map**: Visual representation of data objects, cases, and systems of record.
  + **No-Code Approach**: Allows users to set up, visualize, map, and manage data sources and integrations using a drag-and-drop interface.
  + **Real-Time Updates**: Displays integration and data updates in real-time as the application runs.
* **Example**: In a financial application, the Integration Designer might show how the "Account" data object is connected to an external banking system via REST APIs. Users can easily manage these integrations and update connection details as needed.

**Field Groups**

**Field Groups** are clusters of individual fields that present related data inside a view.

* **Definition**: A field group is a single entity that contains multiple fields, organizing related data together.
* **Usage**: Field groups help organize fields under a single header and can be collapsible for better user experience.
* **Example**: In a car loan application, a field group might include fields for "Car Make," "Model," "Year," and "Price"

**Fields**

**Fields** are individual pieces of data that are displayed and collected in Pega applications.

* **Definition**: Named, reusable UI components that store unique values associated with a case.
* A **field**, when used to collect a piece of information, becomes a **property**. A combination of properties forms a **data element.**
* Fields are used to collect individual pieces of information. When these fields are defined in the system, they are referred to as properties. A combination of these properties, when grouped together, forms a data element.
* **Types**: ***Simple, Fancy, Complex, Primary***

**Simple Field Types**

**Simple Field Types** are basic data types used to capture straightforward information.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Fancy Field Types**

**Fancy Field Types** are more advanced data types that provide enhanced functionality and user experience.

A screenshot of a white sheet

AI-generated content may be incorrect.

**Complex Field Types**

**Complex Field Types** are used to capture more intricate data structures and relationships.

A screenshot of a computer

AI-generated content may be incorrect.

**Primary Fields**

**Primary Fields** are key fields that uniquely identify records and are essential for data integrity and relationships.

**Calculated Values**

**Calculated Values** are fielding whose values are derived from other fields through calculations.

* **Definition**: A calculation expresses a relationship between fields by setting the value of a calculated field based on one or more input fields.
* **Types**:
  + **Functions**: Iterate over items in a list (e.g., Sum, Average, Maximum, Minimum).
  + **Expressions**: Calculate field values by referencing other fields and data relationships.
  + **Decision Tables**: Use conditions to test property values and return appropriate responses
* **Example**: In an online shopping cart, the total cost is calculated by multiplying the unit price by the quantity ordered and summing up the costs of all items.

**Data Relationships**

**Data Relationships** in Pega are used to connect data objects and cases, allowing for organized and efficient data management.

* **Definition**: A data relationship is a container that associates a set of related fields. It does not store data itself but links data objects.
* **Types**:
  + **Single Record Relationship**: Connects a single set of values (e.g., a customer’s name and email).
  + **Multiple Record Relationship**: Connects a list of grouped values (e.g., a list of customers).
* **Example**: In a video streaming service, a "Customer" data relationship might include fields like "First Name," "Last Name," and "Email." This relationship ensures that all customer-related data is linked together

**Case Statuses**

**Case Statuses** in Pega indicate the progress of a case towards resolution. They provide users with information about the current state of a business process.

* **Definition**: Case statuses are labels that describe the state of a case at any given point in its life cycle.
* **Standard Statuses**: Pega includes standard statuses like "New," "Open," "Pending-Approval," and "Resolved-Completed."
* **Custom Statuses**: You can add custom statuses to fit specific business needs.
* **Example**: In an online shopping application, case statuses might include:
  + **New**: When an order is first placed.
  + **Pending-Fulfillment**: When the order is being prepared for shipment.
  + **Resolved-Completed**: When the order has been shipped and delivered

By using data relationships and case statuses effectively, you can ensure that your Pega applications are well-organized and provide clear, actionable information to users.

**Routing in Pega**

**Routing** determines how work assignments are directed to the appropriate user or work queue. It ensures that tasks are assigned to the most suitable person or team based on criteria such as skills, availability, and workload.

* **Worklist:** A list of assignments for a specific user.
* **Work Queue:** A list of assignments for a group of users.

**Types of Routing**

1. **Push Routing**:
   * **Definition**: The system automatically assigns work to a user or work queue based on predefined criteria.
   * **Usage**: Commonly used during case flow processing to determine the next assignment.
   * **Example**: Assigning a loan application to a specific loan officer based on their availability and workload
2. **Pull Routing**:
   * **Definition**: Users select the next assignment from a queue or list of available tasks.
   * **Usage**: Allows users to choose the most urgent or suitable task to work on.
   * **Example**: Customer service agents pulling the next customer query from a shared work queue

**Routing Options**

1. **Route to Current User**:
   * **Definition**: Assigns the task to the user who completed the previous assignment.
   * **Example**: If an employee submits an expense report, the same employee might need to provide additional information
2. **Route to Specific User**:
   * **Definition**: Assigns the task to a specific individual.
   * **Example**: Routing an expense report to a manager for approval
3. **Route to Work Queue**:
   * **Definition**: Assigns the task to a group of users, allowing any member of the group to pick it up.
   * **Example**: Routing customer service tickets to a team of support agents
4. **Business Logic Routing**:
   * **Definition**: Uses conditions and rules to determine the routing path.
   * **Example**: Routing high-value loan applications to senior loan officers for review

**SLA (Service Level Agreement)**

An **SLA** in Pega defines the expected time frames for completing tasks or cases. It includes three main intervals: **Goal**, **Deadline**, and **Passed Deadline**. Each interval can trigger escalation actions to ensure timely completion.

* **Goal**: The ideal time to complete the task.
* **Deadline**: The maximum time allowed to complete the task.
* **Passed Deadline**: The time after the deadline has passed, which can trigger repeated escalation actions.

**Escalation Actions**: These actions can include sending notifications, increasing urgency, reassigning tasks, or automatically resolving cases. SLAs help prioritize work and ensure that critical tasks are completed on time

**Example**: In a customer support application, an SLA might specify that a support ticket should be responded to within 2 hours (goal) and resolved within 24 hours (deadline). If the deadline is missed, the system might escalate the ticket to a manager.

**Agile Workbench**

It is a powerful tool within the Pega platform that facilitates real-time collaboration and feedback during application development.

It helps to report and manage bugs, provide feedback directly within the application.

Using agile workbench we can connect with project management tools like Jira, agile studio for enhanced tracking and reporting.

**Key Features**

* *Real-Time Collaboration*
* *Bug Reporting and Feedback Management*
* *Integration with Project Management Tools*

**Sizing a Pega Platform**

It involves estimating resources, efforts and time required to complete the project.

It is crucial for planning, budgeting and ensuring successful completion of project on time.

**Benefits:** *Accurate Planning, Effective Budgeting, Successful Project Delivery*

**Optional Actions**

**Optional Actions** can be defined as the actions that can be performed by users when the case is not in their control without affecting the main flow.

**Types of Optional Actions**

**1. Optional User Actions**

**Optional User Actions** are single-step actions that users can perform at any time during a case or within a specific stage.

**Example**: Updating a customer's phone number during a customer service case

**2. Optional Processes**

**Optional Processes** are multi-step actions that users can perform at any time during a case or within a specific stage.

**Example**: Cancelling an order in an online shopping case, which involves multiple steps like verifying the order status, processing the cancellation, and notifying the customer.

**Case-Wide Actions**

**Case-Wide Actions** are optional actions that can be invoked at any time during the entire case life cycle.

**Stage-Only Actions**

**Stage-Only Actions** are optional actions that can be invoked only within a specific stage of the case life cycle.

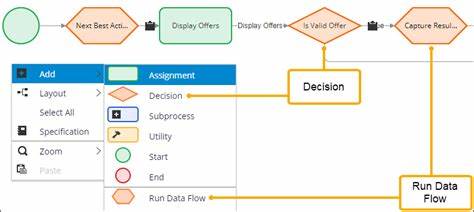
**Automating decisions**

Automating decisions based on business logic, decision shapes allowing application users to focus on decisions that require human expertise.

**Flow**

Flow is a sequence of events that models a business Process. You create flows to support the different paths that users can take through a Case Life Cycle.

You can define paths in a flow by using graphical shapes and connectors.



**Child Cases**

**Child cases** are used to model complex business processes that require multiple related tasks to be handled separately but in parallel.

Each child case represents a series of steps that must be completed to resolve the parent case.

A sub case that doesn’t have any dependency with parent case is called as **spin-off case.**

**Duplicate Case Identification**

It is a process in which we identify a case that is being very similar to another existing case.

It is done based on two parameters:

* + ***Base Condition***
* ***Weighted Condition***

**Base Condition**

* A base condition is a basic rule that must be met for a case to be considered a possible duplicate.
* For example, if two cases have the same email address, they might be flagged as duplicates.

**Weighted Conditions**

* Weighted conditions are additional rules that help identify duplicates by giving different importance levels to each rule.
* Each rule gets a score, and if the total score is high enough, the cases are considered duplicates.
* For example, matching names might get a score of 30, and matching phone numbers might get a score of 50.
* If the total score is above a certain number, the cases are flagged as duplicates.

**Data Validation**

* Data Validation ensures that the correct data is provided and processed.
* Valid data is required so that the system can process the information without error.
* ***Business Logic Validation, Validation on a form and a stage.***

**Validation Types**

There are 3 types of validations in Pega:

**UI Validation in Pega:**

**UI validation**ensures that the data entered by users is correct and meets the required format before submission. This can be achieved using various controls like dropdowns, radio buttons, and date pickers, which restrict user input to valid values

UI Validations use fields that are processed on client-side and are evaluated their itself.

**Validate Rule:**

**A validate rule**in Pega checks that the data entered by users meets specific conditions. It is typically used in flow actions to prevent users from submitting invalid data.

Validate Rule is processed on server, once submit is clicked using properties.

For example, a validate rule can ensure that a date of birth is not in the future

**Edit Validate Rule:**

**An edit validate rule**ensures that the data entered is in the correct format. This rule is applied at the property level and can be used to check formats like email addresses or phone numbers.

Edit validate rule is performed on patterns can be implemented only in Dev Studio.

For example, an edit validate rule can ensure that a phone number contains only digits and is of a specific length.

**Data Transforms**

Data Transforms are used to manipulate data within an application. They can:

* Copy data from one field to another.
* Convert data from one type to another.
* Move data between objects.
* Calculate values.
* Change data formats.
* Set default values.

**Run Data Transform Automation**: Runs the selected Data Transform at runtime or allows you to define a new one.

**Pre/Post-Processing Actions**: Configure a step to run a Data Transform before (pre-processing) or after (post-processing) the step runs.

**Insights**

The Explore Data landing page in Pega allows users to quickly explore and analyze data within their application.

To generate insights, we use KPI (Key Performance Indicators) which are quantitative data.

**Insights** can be generated by anyone who is having access to and are dynamic that are used to handle unplanned situations.

**Reports** are generated by developers by using a static query and we have more control over the data.

**Form Views**

**Form Views** help users complete their work efficiently by providing intuitive interfaces.

**Form View Configuration**

You can configure form views in two ways: on the Workflow tab or the UX tab of a Case Type or data object.

**Workflow Tab**

On the Workflow tab, when you configure a view for a step in the Case Life Cycle, the system creates a form view. This method is used to create forms directly associated with specific steps in the workflow.

**UX Tab**

On the UX tab, in the Other Views pane, you can create different types of views:

* **List View**: Displays data in a table format.
* **Partial View:** Various detailed layouts (e.g., one column, two columns, three columns, sub tabs, narrow-wide, wide-narrow, dynamic tabs(list))
* **Form View**: Displays a form for data entry.
* **Read-Only:** Displays fields in a non-editable format

**Views versus Forms**

**Views** and **Forms** are related but distinct concepts in Pega.

* **Views**:
  + **Definition**: Reusable configurations of UI elements used to interact with a case.
  + **Usage**: Can be used in various contexts, not limited to forms.
* **Forms**:
  + **Definition**: Interfaces for collecting data from users and processing work.
  + **Usage**: A form can have one or more views, but a view is not always a form
* **Example**: A form for a loan application might include views for entering applicant details, loan information, and financial data. Each view is a part of the form but can also be reused in other contexts

**User Interface Elements**

User interface (UI) elements, such as text inputs and layouts, present information and actionable items on the user interface. In the Constellation architecture, Pega automatically pairs fields with the optimal out-of-the-box control. The basic UI elements in Pega Platform include:

* **Autocomplete controls**
* **Charts**
* **Check boxes**
* **Drop-down lists**
* **Links**
* **Rich-text editors**
* **Text input**

**Styling User Interface Elements**

In the Constellation architecture, you can use themes to style UI elements like buttons, links, or headers. Custom themes are particularly useful for meeting branding requirements. Changes to a theme affect all UI elements across the application, ensuring a consistent look and feel. This consistency saves development time, provides a predictable user interface, and makes it easier for users to complete forms

The out-of-the-box Pegasus theme is an example of a default theme provided by Pega.

**Channel Interfaces**

Channels are various platforms through which customers interact with Pega applications, such as messaging services, voice services, web portals, or mobile portals. Pega provides options to configure interfaces for each channel, allowing you to create tailored and interactive user interfaces. Channels are created from templates with predefined layouts and navigation.

Conversational Channels: Conversational channels enable users to interact with Pega applications through digital platforms like:

* Facebook Messenger
* WhatsApp Messenger
* Apple Messages for Business
* SMS/MMS (Twilio)
* Embedded chat window
* Amazon Alexa
* Email

These channels support Pega Intelligent Virtual Assistant (IVA) and Pega Email Bot, which understand user intent and automate responses.

**Configuring Channels:** The Channels landing page allows you to create, view, and edit all types of channel interfaces. You can also download new components from Pega Marketplace. A lock icon indicates that a channel component is not yet added to the application.

**Channel Configuration on Case Types:** You can create channel-specific IVA processes within a case life cycle. For example, you can add a process for users interacting through Facebook Messenger, allowing specific commands to open cases in your application.

**Dashboards**

A dashboard is a landing page accessed by users in their portal. It consists of widgets that consolidate summary information and key performance indicators (KPIs) from different sources, presenting them in an actionable format. Dashboards are tailored to specific personas to help increase productivity.

**Persona-Based Interfaces:** Pega uses persona-based interfaces, meaning each user sees a dashboard tailored to their role. For example, a manager might see the number of open cases and overdue assignments to manage their team's workload effectively.

**Organization:** A dashboard template defines the layout and divides the work area into regions. Regions are containers for widgets, which display information about the application and KPIs. For example, a two-column template might have Region A on the left and Region B on the right.

**Sharing Dashboards:** Dashboards can be created for individual, team, or all application users. Visibility options include:

* Private: Only you can see it.
* Shared: Users with access can see it.
* Public: All users can see it.

**Users and Personas**

Applications have different types of users who perform various tasks. For example, Service Representatives and Clients access different interfaces and features. Pega Platform provides a User Management feature to organize users into Personas, which share common security and assignments.

**Personas:** A Persona represents a type of user with unique needs. For example, a technician might need a specific user interface to perform service tasks. In App Studio, you can add a new Persona by navigating to Users > User Management and adding a Persona in the Persona tab. Once created, you can assign it to users and configure the application to meet the Persona's needs.

**Default and Custom Personas:** When you create an application, Pega provides a default Persona called "Users," who are responsible for creating and resolving cases. You can create additional Personas when a unique combination of Channel interface, responsibilities, permissions, and work routing is needed. Each new Persona comes with default permissions and a Work Queue, which you can customize.

**User Management:** A user is an individual who interacts with the application. In App Studio, you can create a new user with an email address and assign them a Persona or developer role by navigating to Users > User Management > People. Each Persona created in App Studio has an associated role, but not every role has an associated Persona.