**Overview**

MDC is a B2B organization that provides product delivery services. To facilitate business expansion, MDC intends to automate essential procedures through the development of a Pega application. The program should be simple to use and easily expandable to enable future upgrades, as stated below. List of processes

The need is to build a delivery solution competent of automating the tasks efficiently:

1. Business partner Registration and Renewal
2. Shipment Request
3. Delivery Slot Request / Demand
4. Truck procurement
5. Invoicing

**Identified Process/Functions**

* Business Partner Registration – Business Partners are facilitated to register on the MDC website.
* Business Partner Renewal – Business partners will renew their subscriptions using the pega application. Additionally, business partners may modify the plan during the renewal term.
* Business Partner Cancellation – Business partners will invoke cancellations using the pega application.
* Business Partner Shipping Request – Allow Business Partners to make and cancel shipping requests in accordance with their registered plan.
* Notify Membership Renewal – The system is intended to notify reminders to business partners regarding subscription extensions.
* Truck Vendor Registration – Truck vendors are facilitated to register on the MDC website
* Registration Approval – The executive manager analyses the registration and then accepts or rejects it.
* Payment – Pega application to allow Business Partners and Truck Vendors to pay for Renewal and Registration
* Truck Vendor Ranking – The city manager must rank the trucking businesses that will serve the city.
* Truck Assignment – Pega application to enable city managers to handle truck assignments originally by automated means, followed by manual assignment.
* Manage Delivery Price Plan – Managed delivery price plans should be able to be configured or extended based on business requirements.
* Manage Pricing Model and discounts
* Manage Shipping Category – system to maintain the shipping category
* Manage Membership Plans – Executive Manager should be able to configure(Add/Update) plans
* Invoicing – Pega application that facilitates Business Partners and Vendors to view the invoices associated with services requested/provided.

**Assumption**

1. At any moment, registered business partners are limited to a single active registration.
2. There will be no cancellation of agreements between Truck Vendors and MDC.
3. There will be no shortage of trucks for assignment.
4. The design does not consider proximity when allocating delivery trucks.
5. The maximum length of a package is 60 inches, and the maximum weight is 250 pounds..
6. MDC only ships ‘Same Day' and ‘Intra-City' packages.
7. There will be no truck allocation if the Gold membership partner cancels shipping requests and if the corresponding truck does not carry goods from any other partners. The vehicle seller will not be compensated in any way.

**Design Discussion for the identified processes: - To be filled**

**Business Partner Registration/Renewal**

According to the requirement, the registration process has a life cycle of its own. This procedure entails several participants in order to complete the registration of business partners. As a result, a Case type is the optimal solution for solving this problem.

The Business Partner initiates the registration process using the Pega Mashup, and Pega gains authentication in order to create cases via the **Model operator**. The business partner presents the case for approval to Executive Manager by providing necessary registration information such as business partner information, membership plan, and payment. By adopting a case match rule, the system can detect duplicate registrations. Duplication can be averted by validating the combinations of registered business name and city attributes. An out-of-the-box approval flow would be used to administer the approval process. To establish an operator, the OOTB AddNewOperatorRecord method will be used; an email notice will be sent to the business partner with the needed credentials and required information. A case is resolved following the Executive manager's approval or rejection. In the event of rejection, the case will trigger the refund procedure, which is believed to be an alternate path to the business.

**Renewal** is performed in the same manner as registration. Between renewal and new registration, the only distinction is the kind of event. As a result, the registration case type can be utilized for Renewal transactions.

**Renewal Reminder** 15 days Prior notification – A job scheduler is used to filter the data type for active business partners. The logic is as follows: activity to determine the current date and add 15 days to obtain the range for filtering the data in the business partner table. As a result, the system will send an email notification to any business partners that fall within this range.

Dashboard Reminder notification – A Label with a conditional display is configured on user dashboard. Condition is evaluated with Thread level data page, Logic on the Data Page to check if the current operator is inside the range(CurrentDate + 15 Days) a post-transformation of the data to set the flag(***IsAlert***) to true. Example – D\_DefineRenewalAlert.IsAlert = true

**Cancellation -** Cancellation of a business partner registration requires approval from the City Manager / Executive Manager and/or a refund procedure to be launched, a case type approach would be appropriate. A separate case type is introduced in the

Alternatively, a data type is used to record registration information. Harness is provided to business partners in order to gather registration data. A flag *(ApprovalFlag)* is set to distinguish between approved and rejected instances. The Executive Manager will be presented with the information and will be asked to capture his decision. Emails will be sent in response to approval/rejection. This design, however, is not optimum for consideration as a solution. It is regarded to be an old-fashioned method; each extra activity, operation, or piece of data that has to be captured results in additional data types and overhead.

**Truck Registration**

Truck vendor registration may have been regarded a data type change only if the information was being captured and the activity was being performed by a single user. Due to the addition of approval to the process, a case type is defined that is best suited for the design.

Through MDC website, vendors register for truck services in specific cities for products shipping and delivery. While registering, each truck vendor registration procedure begins with the collection of vendor, Truck, service type and contract information, followed by an approval workflow by executive manager and rating process for city managers. As with Business Partners, truck vendors gain authentication through the Model operator in order to create cases. The primary difference between the truck registration and Business partners is that; business partners are provided Pega application access unlike Truck vendors

**Truck Vendor Ranking:**

Although truck rating is an essential part of the business process, it is performed by the city manager and an external system in a single step. Ratings are gathered during the registration process and post delivery. This therefore, can be handled as single sequential steps (Asisgnment/DataPage*(externalSourec)*) The designing a case to accomplish this functionality is discarded. City Manager will be assigned a task in his worklist to capture the truck rating.

**Business Partner Shipment Request**

Once their registration is approved, Business partners can initiate shipment requests. There are reasonable grounds to believe that the method should be case-based, as shipping requests go through a number of stages, including initiation, reservation of slots, delivery, and invoicing. Furthermore, once a status changes, a notification should be sent to all stakeholders.

The purpose of a business partner's shipping request is to move packages from source to destination inside a city. Each shipping request refers to the gathering of the following information

* No of units
* Pickup
* Delivery locations
* Date of dispatch.

The process entails the City manager updating the status of shipment requests following shipment delivery, updating truck rankings, and eventually initiating invoicing for a business partner. Due to the fact that this requires several actors to process the journey, a case type is the ideal option.

**Control initiation – timelines**

When a business partner initiates a shipping request via a logged-in user, ABAC is used to govern the shipping request. *(Access control policy condition - Update - The user can create a case that meets the policy conditions or update data for such a case)* condition to determine the time and partner member type (*Gold,Sliver,Bronze*) evaluation. Rules for case creation partners as per the requirement\*\*\*

**Truck Computation logic**

After a Shipment request case is created and the relevant information is gathered, a single instance is generated in the "DaliyShipmentLoad" data type for that particular date (one record per day). This record will be updated when another shipment request is received from another business partner or same business

* **Truck Capacity – 64 units (Reference Data Type/Application Settings)**
* No of Truck calculation is updated when a records gets updated
* Total No Requested units *(Gold Customers)* / Truck Capacity (example - 290/64 = 4.5 truck rounded to 5 truck ) Left out units for other partners to accommodate the shipping request is (Silver and Bronze) is 30 units (i.e. total capacity for 5 truck = 320, consumed capacity 290, balance 30 units.

Once a request is created is a case is parked in wait shape until the delivery is completed.

**Shipment Delivery**

The delivery process can potentially be performed with a data type or Subprocess but this method does not provide the necessary flexibility to fully construct and evolve the process. Furthermore, the truck allocation is not tied to a single request; rather, it is anticipated to deliver based on the truck load. Therefore a case is best suited for this requirement. Shipment delivery cases is triggered by a job scheduler based on the “DaliyShipmentLoad” data type instance for that respective date *(Please refer to the truck Calculation section)*

*Job Scheduler time 1.00 PM logic to initiate the Shipment delivery case*

If the “DaliyShipmentLoad” data type contains an instance for that specific date, A shipment Delivery cases will be triggered according on the number of trucks computed based on the requests initiatedfor that day. For instance the No of truck calculated as 5 trucks consolidation all the request. Job scheduler will run For Loop based on No of trucks property. In this case 5 cases are created and assigned to City Manager. An automated truck allocation is performed prior to assigning to the city manager ***(Priroty Calculation)***. Trucks are decided from the vendors first in first out depending on the vendor's ranking (if both vendors have the same ranking, a random vendor will be chosen to determine the truck).

**Invoicing**

An invoice's purpose is to collect data, generate an invoice, obtain approvals, and send the invoice. An invoice may be represented by a case type, depending on the nature of the business process. The distinct case-type method adheres to the ‘SOC principle' and ‘subject to being held to account' strategies, which can aid in the development of modular applications, which is an important area to focus on because this procedure will be reused/extended for different business sectors/Cities in the future. As a result, a distinct case type is advised for the Invoice process.

The following processes require use of invoicing.

* Business Partner Invoice
* Truck Vendor Payment
* Registration payments

Alternatively, a separate case type for each process might be used to meet the need. Security and reporting obligations will grow more difficult. It also violates the fundamental design principle of reusability.

**Manage Membership Plan**

1. Data type can be implemented to store and retrieve this information. Given that the criteria expressly says that alternative pricing plans may be established in the future, it is recommended to choose data type as selection. The executive manager may be tasked with the responsibility of managing this sort of data.
2. Alternatively, Field values are an alternative technique for specifying permitted property values. Field values help control the list of permitted values independently of the property. It can reuse a single property and change the permitted values based on the context of the property by managing the authorized values independently from the property. A filed value can be used to present the user with a list of possible options. When a new plan is introduced, a field value rule should be promoted or maintained in the production rule set.

**Manage Shipment Categories**

As of now, we have no requirements for modifications to the shipment category; nevertheless, there is a possibility that other categories could be added in the future to accommodate new enterprises. As a result, it seems logical to implement this as a data type so that future modifications may be readily handled.

**Manage pricing models and discounts**

The data can be set up as a data type or a decision table, depending on preferences. Data types are a possibility if security and operational flexibility are taken into consideration.

**Case Design**

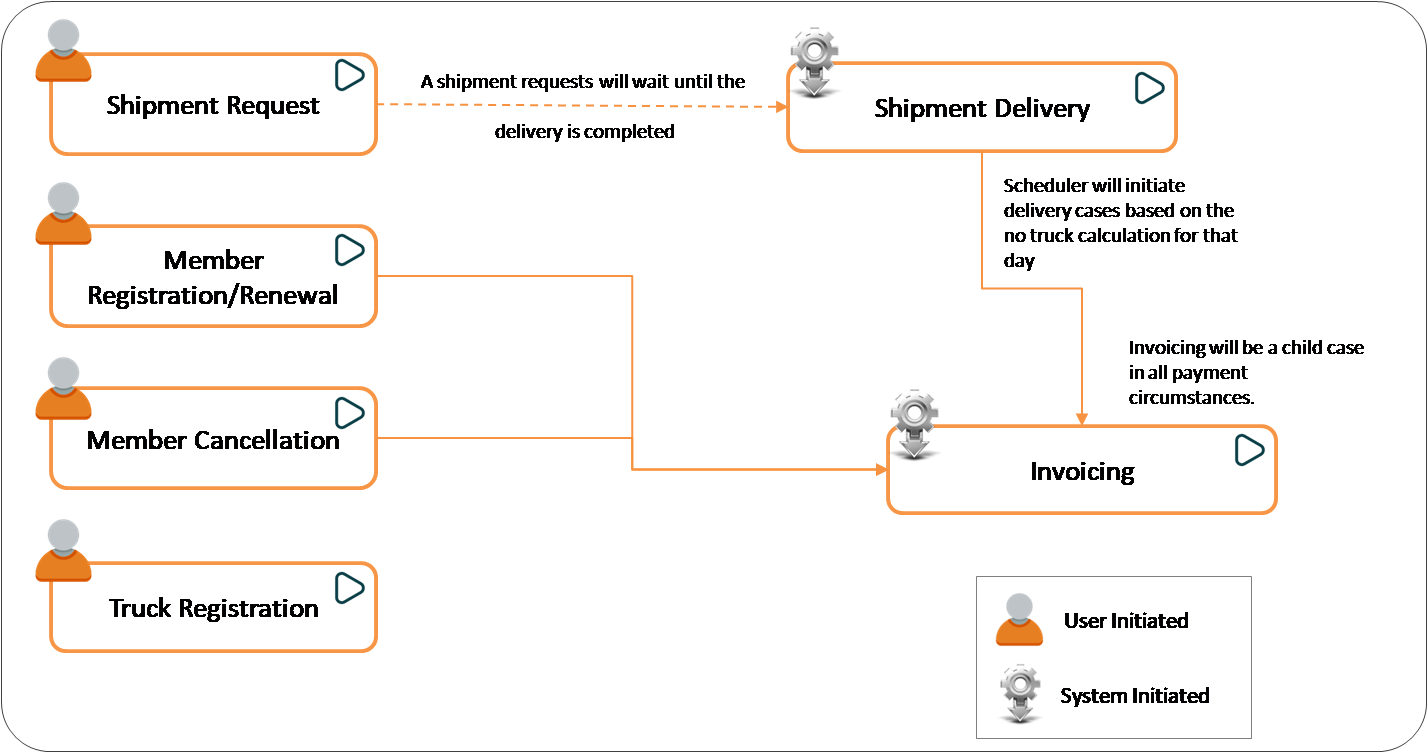
The following table summarizes the case types identified throughout the preceding discussion.

|  |  |  |
| --- | --- | --- |
| **Features** | **Case Type** | **Data Type** |
| Business Partner Registration/Renewal | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |  |
| Member Cancellation | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |  |
| Truck Vendor Registration | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |  |
| Shipment Request | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |  |
| Registered Business Partner |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Registered Truck Vendor |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Vehicle Info |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Invoice | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |  |
| Manage Membership Plans |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Manage Shipping Category |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Manage Delivery Price Plan, Discounts |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Manage Pricing Model |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Truck Ranking |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |
| Truck Allocation |  | C:\Users\santosh\AppData\Local\Microsoft\Windows\INetCache\IE\UD41WHSA\1200px-Check_mark.svg[1].png |

Below are the case hierarchy options:

|  |  |  |
| --- | --- | --- |
| **Case Hierarchy Design 1** | **Case Hierarchy Design 2** | **Case Hierarchy Design 3** |
| * Customer Registration (Parent)   + Invoicing (child) * Member Cancellation (Parent)   + Invoicing (child) * Vendor Registration (Parent) * Shipping Request (Parent) * Shipping Delivery (Parent)   + Invoicing (child) | * Shipping Request (top level) * Truck Request (child case) * Invoicing (child case) * Registration (top level)   + Registration (Specialisation)   + Renewal (Specialisation)   + Cancellation (Specialisation)     - Invoice | * Shipping Request (top level) * Invoicing (top level) * Truck Request (top level) * Vendor Registration (top level) |

Primary Case design



|  |  |  |
| --- | --- | --- |
| **Design Approach** | **Pros** | **Cons** |
| Case Hierarchy Design 1  **(Recommended)** | * All processes are presented as independent component applications this brings the scope to extend the reusability for any other LOB’s and future extension. * **Registration Process** Because registrations are self-contained, they are simple to manage. It's simple to create reports. * **Invoice:** Invoicing is considered a self-contained process and thus can be reused across all business functions, with the potential for expansion based on unique requirements. | For similar Or identical workflows, duplicate rules are kept. Each registration type must be configured and maintained individually and exposed to web site as Mashup. |
| Case Hierarchy Design 2 | Reusability is accomplished by the use of a single registration case. Specialization is used to carry out the intended based process. | * It is difficult to maintain the business partner and truck vendor processes separate in a single flow when there is huge deviation in execution of the flow. Adding to that is the city level process will make it more complicated. * It is challenging to distinguish Registration-related reports unless a differentiating criteria is used to extract reports. |
| Case Hierarchy Design 3 | * Independent cases are easy to implement. | * Truck allotment posses a difficult challenge. Also since the cases are not coupled the data flow and dependencies can also become issue. Although we can update using a nightly job or Case updates need to handle the locking mechanism and data is never a real time. |

Case Type Life cycle for the recommended case hierarchy

**Customer Registration/Renewal**

**Primary Stages (Recommended)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Primary Stage 1**  Status – New  Actor - | **Primary Stage 2**  Status – Pending Approval | **Primary Stage 2**  Status – Pending Invoice | **Resolve** |
| * **Register Info** *(Capture register info UI from)* * **Case match** *(Duplicate Based on Org ID and City )* * **Payment** *(Payment Gateway)* * BusinessPartner-Renewal | * **Approval Flow** *(PEGA OOTB approval for executive manager)* * **Change Stage** *(Conditional to alternate stage based on rejection)* | * **Create Case** Invoice (Child Case) * **Wait** * **Create Business Operator** | * **Send Email** * **Resolve** (Resolve-Completed) |

**Alternate Stages (Recommended)**

|  |
| --- |
| **Alternate Stage 1**  Status – Pending Refund |
| * **Refund** *(Payment refund)* * **Resolve (Resolved Cancelled)** |

**Approach discussion**

Using the Web Mashup interface, the case type will be exposed to the MDC website. The user is presented with a UI form to registration information. Post submitting, the system will use the case match rule to identify the duplicate request and resolve the case else will be presented with payment. Once a payment is done, a payment gateway is linked into the system. The OOTB approval flow would be used to route a case to the Executive manager for approval. A child case is initiated invoice processing (refer to the invoice processing case) , email is sent to the business partner on successful registration. All registered members are captured in a Data type “MDC-Data-BusinessPartner” that will serve as a source for reporting. Rejection cases are processed with a refund transaction and perhaps an email notification is triggered from alternate stage. Furthermore, a renewal process will reuse the same flow; the distinction is dependent on the Model (Registration Type).

**Member Cancellation**

**Primary Stages (Recommended)**

|  |  |
| --- | --- |
| **Primary Stage 1**  Status – Pending Approval | **Resolve** |
| * **Approval** | * Send Email * **Resolve (Resolve-Completed)** |

**Approach discussion**

When a user logs in to Dashboard, the option to request cancellation is available, and the reason for the request is captured. D\_GetMemberPendingShipment (lookup on shipment case, Filter member Org ID and Status) If the system cannot locate a pending request, a case is triggered to cancel and is routed for approval; otherwise, a notice to the user is shown. Upon the cancellation, system to mark the status of business partner to inactive “MDC-Data-BusinessPartner” based on the Member Org ID.

**Vendor Registration**

**Primary Stages (Recommended)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Primary Stage 1**  Status – New | **Primary Stage 2**  Status – Pending Approval | **Primary Stage 3**  Status – Pending Rating | **Primary Stage 4** |
| * **Collect Information** | * **Approval** | * **Rating** (External System simulation) * **Rate Vendors** * City Manager | * **Notify** * **Resolution** |

**Alternate Stages (Recommended)**

|  |
| --- |
| **Alternate Stage 1** |
| * **Rejection** * Resolved-Rejected |

Similar to Member registration, vendor registration will be identical to Member registration with the exception of user creation. Using the Web Mashup interface, the case type will be exposed to the MDC website. The user is presented with a UI form to registration information. Vendor data is persisted in the data type “MDC-Data-Vendors”.

**Shipping Request**

**Primary Stages (Recommended)**

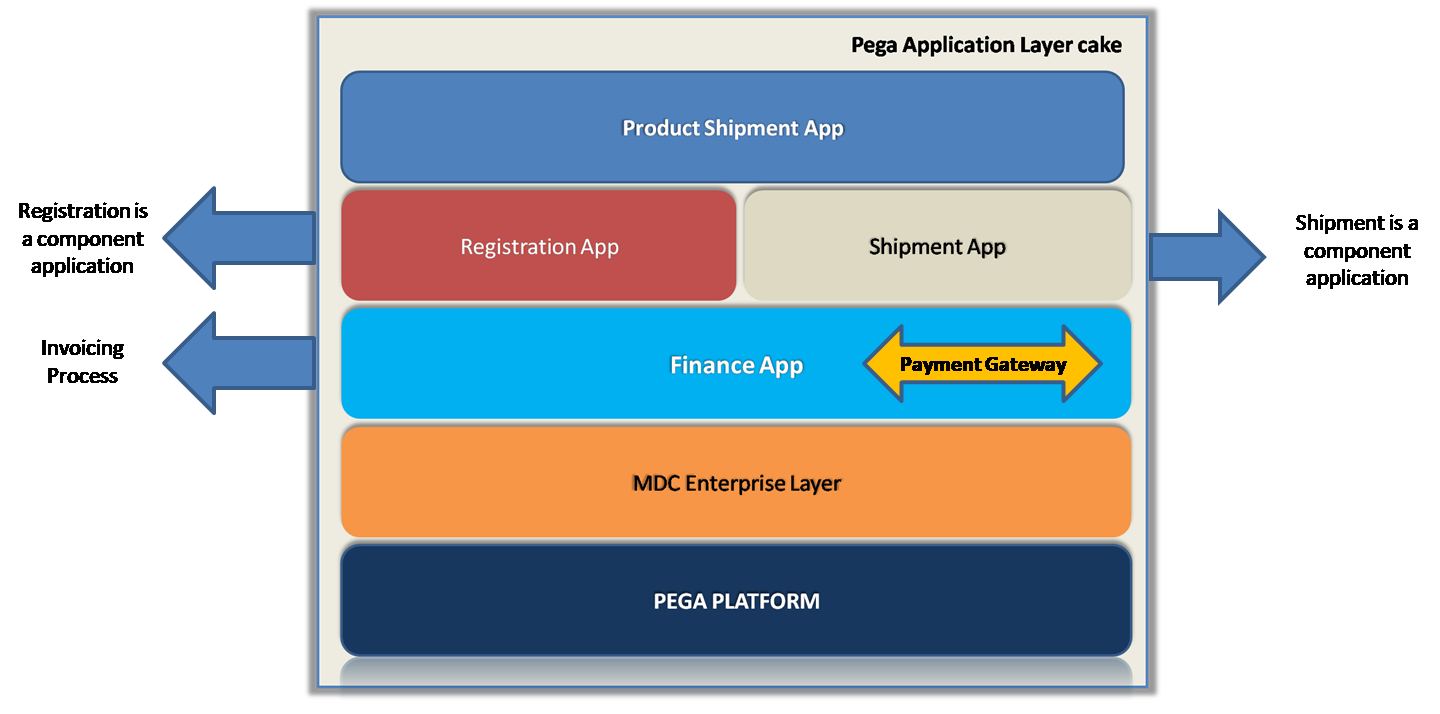
|  |  |  |  |
| --- | --- | --- | --- |
| **Primary Stage 1**  Status – New | **Primary Stage 2**  Status – Pending Scheduling | **Primary Stage 3**  Status – Pending Invoice | **Primary Stage 4** |
| * Initiate Shipment request | * Reserve Slots   + Gold Partner First   + Later Others * Add/Update TruckAllotment * Wait | * Invoice (Child case) | * Resolve * Trigger Invoicing, Notify Partner |

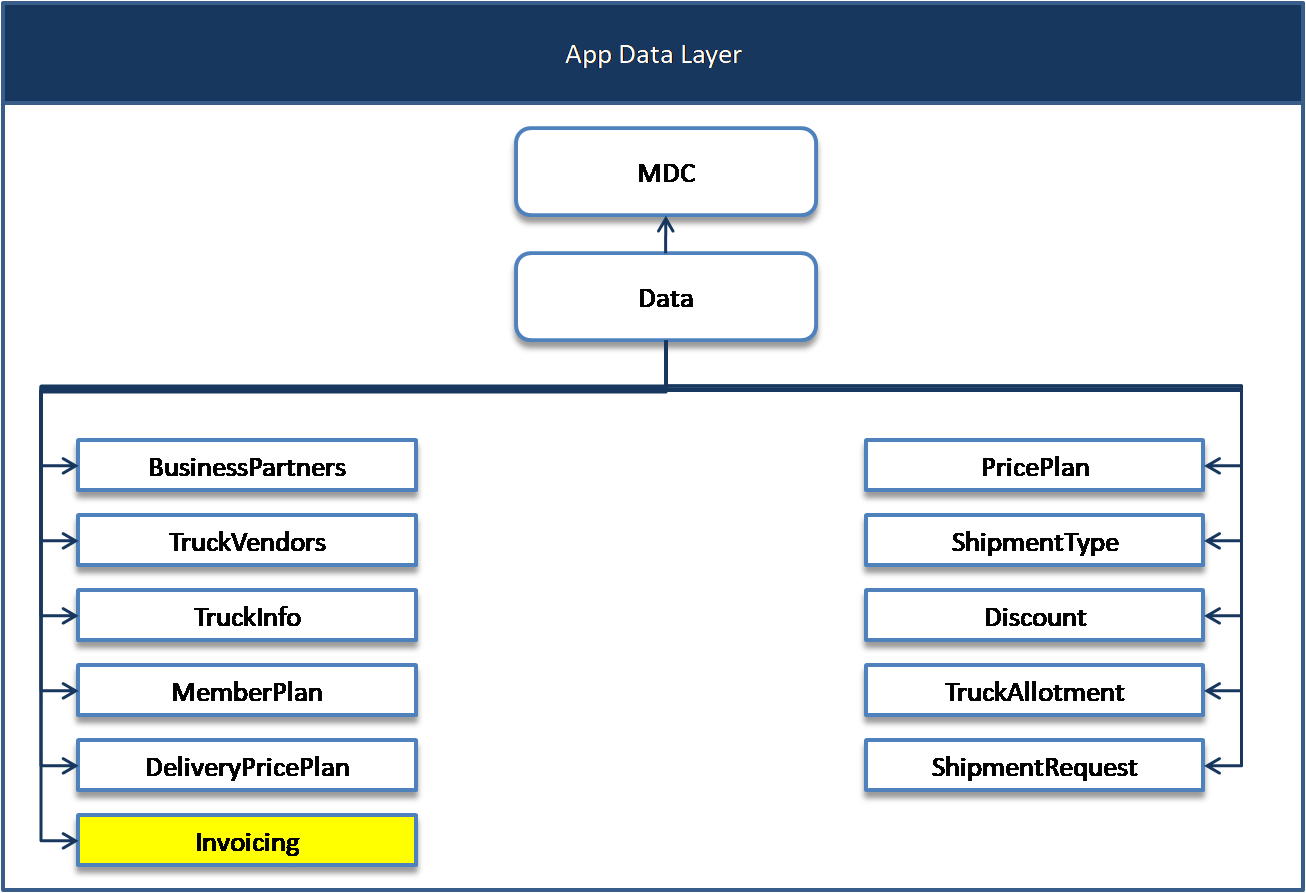
**Alternate Stages (Recommended)**

|  |
| --- |
| **Alternate Stage 1** |
| * Resolved-Cancelled |

Application Class Structure

|  |  |  |
| --- | --- | --- |
| **Approach** | **Pros** | **Cons** |
| Approach#2 – Multiple Applications with direct inheritance & ruleset specialization | * Development and rule maintenance will be simple as city specific rules are defined in city specific classes and rulesets. | * Executive manager needs to switch back between multiple applications for approvals which will be a time consuming task. * Current reporting requirements will get complicated. * More number of applications need to be created when business introduces its operations to new cities. |
| Approach#1 – Single Application with Pattern Inheritance specialization | * Single application, maintenance will be easy * Rules navigation would be easy as they are grouped under different classes. * Executing manager can work approvals without switching between applications * Reporting requirements can be implemented easily. | * There may be additional work required whenlarge number of city specific business requirements are added. * Packaging for city specific changes may be little difficult. |



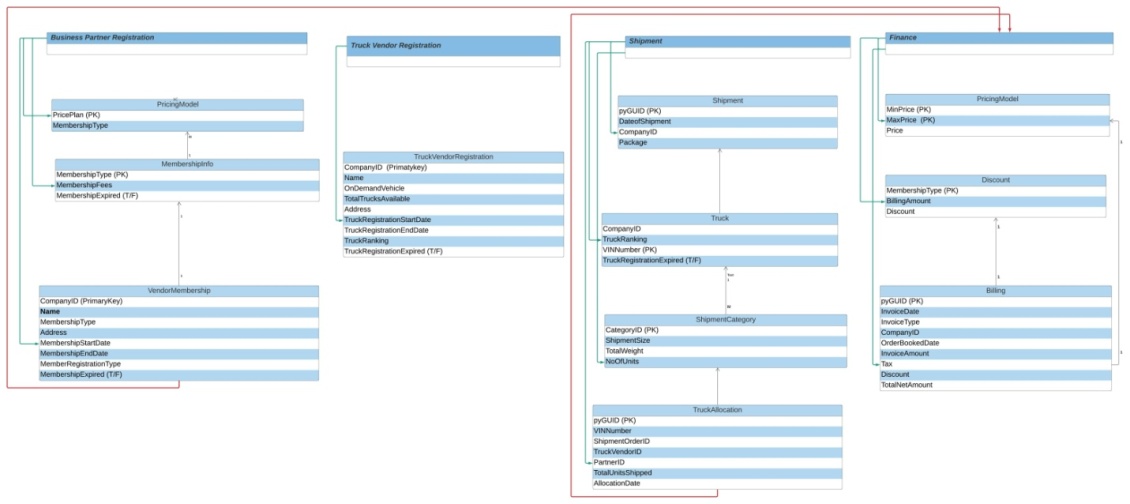


Classes may be extended to any state by adding the state name to the end of the class name through class specialization. If any particular modifications are made to any of the case types, that class will be extended. This method would be taken when each state undergoes significant change.

Data Model design

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Type** | **Keys** | **Significant Properties** | **Inheritance** | **Description** | **Schema** |
| MDC-Data-BusinessPartner | **Key:** OrgID |  | MDC-Data | Each business partner registers his or her company/enterprise and is assigned a unique ID that will serve as the table's key. | CustomerData |
| MDC-Data-Vendors | **Key:**OrgID |  | MDC-Data | Every Truck Vendor registers his company and is provided with a unique ID which will be used as a key for this table. | CustomerData |
| MDC-Data-VehicleInfo | **Key:** VINNumber |  | MDC-Data | Each vehicle is identified by a unique VIN number, which will serve as the table's key. | CustomerData |
| MDC-Data-MemberPlan | **Key:** MembershipType |  | MDC-Data | Membership Type is always unique to associated business attributes. | CustomerData |
| MDC-Data-DeliveryPricePlan | **Key:** PricePlan |  | MDC-Data | PricePlan is always unique to associated business attributes. | CustomerData |
| MDC-Data-PricingPlan | **Key:** RangeFrom+RangeTo | * Member ship plan | MDC-Data | Pricing range is unique. | CustomerData |
| MDC-Data-ShipmentCategory | Key:**CategoryID** |  | MDC-Data | CategoryID is unique and is allocated based on unique combination of dimension and weight. | CustomerData |
| MDC-Data-Discount | **Key:**PartnerType+InvoiceAmount |  | MDC-Data | Discount may vary for a partner type based on invoice amount. | CustomerData |
| MDC-PDS-Data-TruckAllotment | **Key:** TruckVINNumber+ShipmentRequestID+ShipmentDate |  | MDC-Data | This table is to store the truck allocation and its capacity by its ranking. | PegaData |
| MDC-Data-DaliyShipmentLoad | **Key:** City+Date |  | MDC-Data | To leverage BOLB capability pega data is chosen over customer data | PegaData |
| MDC-Data-BusinessPartner-Renewal | **Key** OrgID |  |  |  |  |
| MDC-Data-Invoicing | **Key:** pyGUID  **Description**: InvoiceID+OtherCombination would be a unique key but not necessary as OOTB pyGUID will suffice to meet the requirement. |  | MDC-Data |  | CustomerData |

ER Diagram



**Security Design**

The goal behind security features in an application is to restrict access to certain users rather than allowing everyone access to the entire program. In Pega, this is done via Access groups, roles, and privileges. In the application, this helps restrict case access. For the case journey across the whole process, access groups and privileges as well as ABAC and RBAC features are utilized in designed to safeguard the application.

**Actor’s security overview**

|  |  |
| --- | --- |
| *Executive Managers approve registrations, therefore they have approval privileges.* | **C:\Users\santosh\Desktop\Exam_Overview\Executive Manager.png** |
| *The City Manager rates Truck firms after registration approval and assigns trucks manually.* | **C:\Users\santosh\Desktop\Exam_Overview\City Manager.png** |
| *Accountants process the invoices post shipment for that respective transaction* | C:\Users\santosh\Desktop\Exam_Overview\Accountant.png |
| *Business Partners initiate the registration and Also, initiates Shipment request* | C:\Users\santosh\Desktop\Exam_Overview\Business Partners.png |
| **Note:**   * *A system Service access group will be intoduced to perform system related tasks* * *A Guest/Model Operator will be added to oversee the Mashup user registration process for business (Guest User Box to updated) Partners. The Pega User Role will be assigned to this Operator.* * *An administrator operator will be introduced to manage all administrative tasks across the application. Administartor will be designated with dependent roles from pega (PegaRULES:SysAdm4, PegaRULES:SecurityAdministrator)*   *Open ID – Client key and Client URL – Need to more details* | |

**Security Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Persona** | **Case Access** | **Data Access** | **Channels** | **Security Rules** |
| Executive Manager | * Business Partner Registration * Business Partner Renewal * Truck Vendor Registration |  | Web | *Restriction, through the use of privileges at the flow action level* |
| Business Partner Manager | * Business Partner Registration * Business Partner Renewal * Shipment Request |  | Web |  |
| City Manager | Truck registration |  | Web |  |
| Accountant | Invoice |  | Web | *Assuming invoicing is a distinct process, only accountants with the appropriate access group may use it.  Access level 5 for Invoicing, and 0 for all other classes.* |
| Guest | * Business Partner Registration * Truck Vendor Registration |  | Web |  |

|  |  |  |
| --- | --- | --- |
| **Security Requirement** | **Primary/Recommended approach** | **Alternate approach** |
| Only Executive Manager can approve registrations and renewals | *Privilege at flow action* | *When Rule based on the Access Group at Flow action* |
| Business partners can view only their requests and reports. | *Report definition on Shipment Case Type, filter instances by* ***pxCreateOperator*** *using current operator page.* ***(Exclude Resolved Cases)*** |  |
| Accountant can access only invoicing process. | *Assuming invoicing is a distinct process, only accountants with the appropriate access group may use it.  Access level 5 for Invoicing, and 0 for all other classes.* | *Views of the Worklist/workbasket may be filtered to display only invoice cases associated with the Accountant workgroup / access group.* |
| Partner Type and Pricing Plan can be updated only by Executive Manager | *Partnering Type and Pricing Plan are considered data types and will therefore be delegated to the Executive management access group..* | *Access control policy of action type “Update” will be created for the invoice class with access control policy condition comparing current user access group with invoice application specific access group.* |
| City Manager can see requests only pertaining to their city | *Opeartor Skills can be leveraged.* *A city manager is assigned the city name as a skill, which enables the report to gather relevant city-operated instances depending on the currently logged-in Operator (OperatorID Page)* | *Alternatively, a data type is defined that stores Managers and their associated city operations; a report may then retrieve city information depending on the Logged in user, therefore retrieving city-specific instances from the system.* |
| Truck vendor registers on MDC website and provides the required details, they don’t have access to the application | *Through mashup, only that Vendor Registration case type is accessible, limiting their access to other activities.* |  |
| Profit related information should be only seen by Executive Manager and city managers can view their respective city profit information | *To control the view, attribute-based access control to be used; a property-level masking will be implemented depending on the access policy condition. (Access Group and operator Skill vs. the city in the case)* | *.* |

**Handling Application Security Vulnerabilities**

**Content Security Policies, CSRF & CORS:**

* 1. For the mashup to function properly, the same site cookie property should be set to none.
  2. To strengthen the application's security, the CSP will use a policy called Allowed websites of Frame-Ancestors with the value of the host application's URI to ensure that the mashup loads without difficulty. Additionally, the trustworthy origin will provide a host URI value that informs the system about trusted sources from which the application may be accessed. CORS policies must also be configured for future accesses through AJAX, if necessary.

**Extensibility & Specialization**

The application stack is structured in such a manner that all reusable and common assets may be expanded at the top layer.

Case types and data objects can be imported and extended to new component applications.

Specialization can be used to extend implementation specific to cities in case of taxation or any specific business flow.

**Reusable assets**

--Please add--

1) Data objects are defined at organization layer to enable reusability. Mention few data classes here\*

2) Component applications have been built which would help in reusability.

3) Registration, Finance will have assets that at the moment are used by product delivery services and can be used by moving services as well.

**User Interface**

All personas *(Except Truck Vendors)* would have access to user portals. The portal will be redesigned to accommodate the addition of new navigation options.

* UI Pages will be utilized to manage a few business requirements, such as cancellation of registrations.
* Widgets will be used to operate on delegated data types. They will be added in accordance with the persona.
* Business partner and vendor registration would be made part of the MDC portal through mash up.
* Guest users do not need to input any access credentials since when the mashup link is clicked, the mashup user credentials are passed through.
* App studio would be used to construct the application, with screens being created using design templates. Even citizen developers would find it simple to sustain development via app studio.

**Reports**

**Registration and Renewals Reports:**

Following registration, the design document specifies that a data type “MDC-Data-BusinessPartner” is loaded with the newly registered users. On this class, a report definition will be defined that can be scheduled to extract the registrations that occurred in the previous week/month/year.

During renewal, the entry in the business partner table is updated by extending the expiry date and also an entry is made into the MDC-Data-BusinessPartner-Renewal table. This table holds all the renewals. The key for this table would be a combination of BusinessPartnerID and Start Date and End Date. A report needs to be written on this table and to be scheduled which will give all the renewal details.

**Products delivered per partner:**

As when a shipment request is created the shipment details are stored in a data type MDC-Data-“DaliyShipmentLoad” This table would hold the business partner ID and shipment request ID and the corresponding units he has raised request for. A report would be written against this class which would pull business partner grouped results. This data will be extracted from the Index table.

**Executive Manager Profit report**:

Profitability may be defined as follows based on existing business operations. Amount received as part of ((business partner registration+shipment request) - Amount paid to the vendor for truck services utilized). This can be accomplished in a variety of ways.

**Recommended approach**

As soon as a payment is made a declare trigger needs to be configured and should update the profit value in a data type(MDC-Data-Profit). This payment could be to MDC or from MDC.

**Alternate approach**

Run a period job using a job scheduler which keeps hitting the payments data type(Mention data type) and calculates as per the above logic and updates a table.

Design questions 1

Moving services will be built in parallel to product delivery services. This will be a production application build on other component applications like product delivery services. This application if needed can extends case types corresponding to registration and finance and truck management.

This application can be placed either on top of product delivery services or in parallel to product delivery services. With the limited requirements of Moving services, it certainly seems to be an application which will have its own business processes and would not inherit anything from product delivery. Product deliver is also built in such a way that all reusable components are placed in the lower application stack.

MDC enterprise layer has all the data types which can be reused or extended from moving services.

Finance application will provide all the payments, refund and invoicing capabilities with the suggested design.

Registration may or may not be needed for moving services but it is made available for reuses. Also moving services would definitely need vehicle assignment and procurement. At the moment the logic is very specific to product delivery this can be updated to a common phrase like vehicle and can be resued by moving services.

----PLACE the MOVING SERVICES INCLUDED APPLICATION STACK PICTURE HERE----

**Design Question 2:**

With the current design during invoicing taxes are applied without considering any state specific taxes. To consider state specific taxation during invoicing multiple approaches can be considered. Few of them are suggested below.

\* Data Type Approach: Creata a data type and have state names as a column and specific tax details in corresponding columns and retrieve this information using a data page and use the same during invoicing. This approach works when its only meant to calculate the final invoicing amount.

\* Decision Table Approach: This is similar to above approach but is maintained in a decision table. To maintain the same this rule has to be delegated to users and the same is not that easy comparitative to approach 1.

\* Circumstancing Approach: This approach can be followed when there are very specific rules pertaining to the city. Few calculations are not very straight forward that you can acheive through approach 1 and 2. There could be different logic while calculation apart from tax numbers. This approach would be suitable in that case.

\* Specialization Approach: This approach is similar to approach 3 but is more extendable and vast. This approach would mean creating multiple classes for state and define rules in that class. While invoicing, Page Class should be point to pick the right rules.

Conclusion: With the limited requirements mention, maintaining a data type with state specific details is the recommended approach. It can be easily delegated for any updation.

**Design Question 3:**

In the current design while a shipment request is raised, the business partner can chose only one day. The shipment request form needs to be updated to have a check box asking if they want to raise recurring shipment requests. When clicked another section would be loaded where the user would furnish date from and date to and package details. Once submitted these details are stored in the shipmemt requests data type. one record is inserted per day in this table. The job scheduler which runs at 1PM takes account of such recurring shipment requests in the table and creates shipment cases for them.

Nee to introduce Agnet to perform recurring shipment load also a harness to capture the recurring shipment

---Review and add the data type details--

**Design Question 4:**

To accommodate additional cities and their associated managers and accountant below is the suggested organization chart.

Organisation would be MDC

Divisions would be Finance, SERVICES

Units would be under services product delivery and under finance would be Payments

City will be added as Skill for all operators

Workgroups would be maintained as per divisions

if a city contains one or more managers they would be made part of a work queue with a manager among them.

As accountants are considered as external users, it may not be required for them to be part of the organisation structure.

-- CAn you represent this as a picture--