isango! Platform Architecture

CONTENTS

REVISION HISTORY

Version	Date	Description	Author(s)	Reviewer(s)
V.1	22/04/2008	Initial Version	Prashant Kumar	Hara Prasad, Saurabh Chowdhury
V.2	16/10/2011	Updated For LOT family websites	Prashant Kumar	Hara Prasad, Saurabh Chowdhury

This page is intentionally left blank!

INTRODUCTION

This document describes all the architectural components and high level design decisions for the isango! platform.

This document is intended to highlight high level architecture, system components integration workflows.

DESIGN PRINCIPLES

Service-Oriented Architectural approach has been followed in constructing the services exposed to the consumer based delivery channels. Following design principles and best practices have been kept in mind while creating the system design –

LAYERED DESIGN

The isango family of websites have been built using the classical three layered design principle. The layers are divided into Presentation, Business Services and Persistence. All the ASP.NET pages, images, css and scripts are present in the Presentation Layer. All the business logic and the respective workflows are contained in the business services layer. It also does the job of connecting with multiple external systems of Suppliers to aggregate data to abstract data as a service to the presentation layer. In the course of data aggregation and potential transformation, it will also manage any transactions that might be involved. All the logic of interacting with the database to load, find or save is contained in the Persistence layer.

RESPONSIVENESS A.K.A PERFORMANCE

All the system components are built by keeping in mind the responsiveness of the system. The components strive to find the right balance between flexibility and performance. isango.com and other websites employ data caching techniques to make the system more responsive. Cache building employs both reactive and proactive schemes. For example, master data like Countries etc. are built using the proactive scheme where they would only be cached at the start of the application whereas reactive schemes are used to cache all the activities that have been viewed.

COARSE GRAINED A.K.A. CHUNKY INTERFACES

All the services expose coarse grained calls to persist or load data from the persistent store. This avoids chattiness between system boundaries and systems would have to make fewer invocations to realize the desired functionality.

LOOSE COUPLING

All the component interactions are done through interfaces thus promoting loose coupling between system boundaries. This gives us enough flexibility in future to swap in different business implementations of a specific interface. When we developed the justlondontheatrebreaks.com website we had to integrate products sourced from a 3rd party supplier with realtime availability and pricing to minimize the impact on our existing system & code we used the inversion of control pattern through dependency injection.

WELL ENCAPSULATED INTERFACES

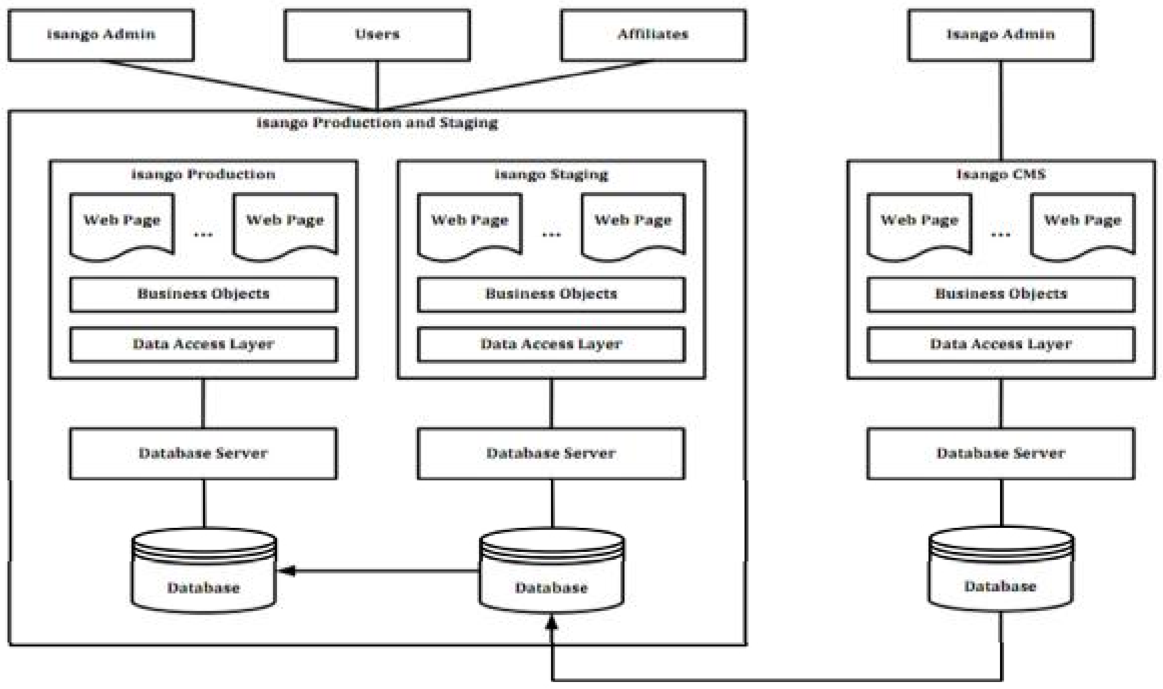
All the service interfaces help the clients realize all their functionality in totality. This makes sure that the clients don't have to consume multiple interfaces to realize a business use case. Furthermore, this approach reduces the number of entry points into the services layer to a minimum, thus making the code more maintainable and easier to debug.

For example, **IManageActivitiesService** exposes all the methods needed for tours and activities and the callers won't have to consume another interface to load tours and activities.

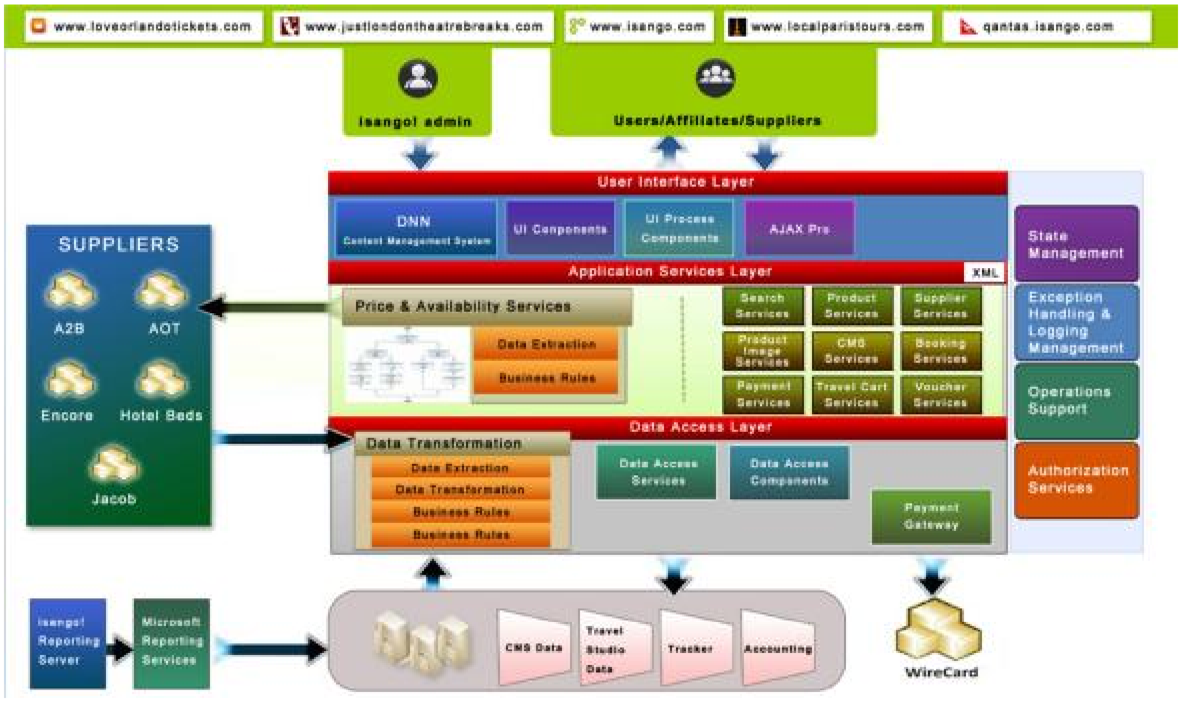
ROBUSTNESS

The design of the system is robust so that it can recover from any type of system exception. All the layers or the sub-systems would employ standard and best practices for exception handling.

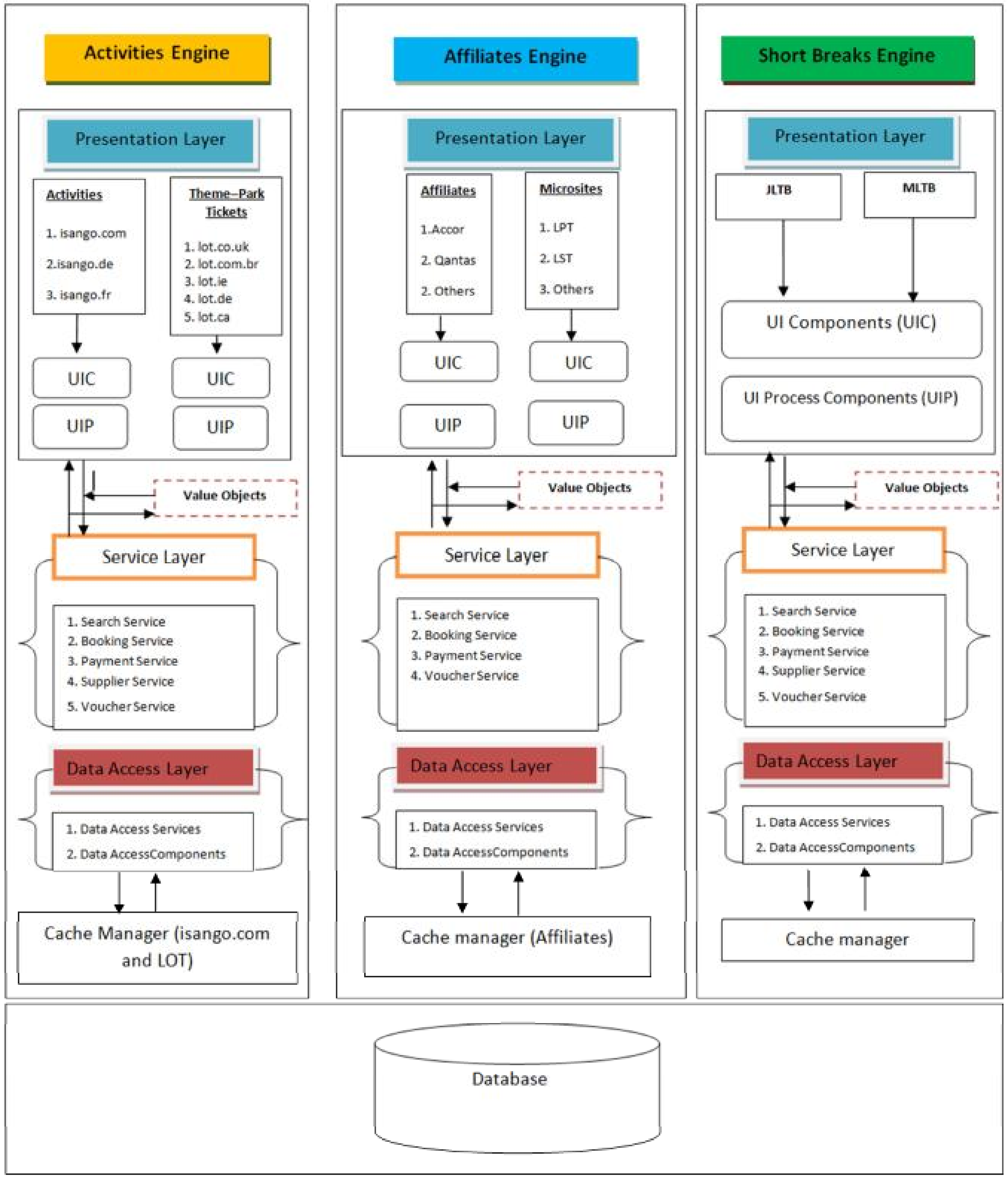
SOLUTION ARCHITECTURE AT A GLANCE



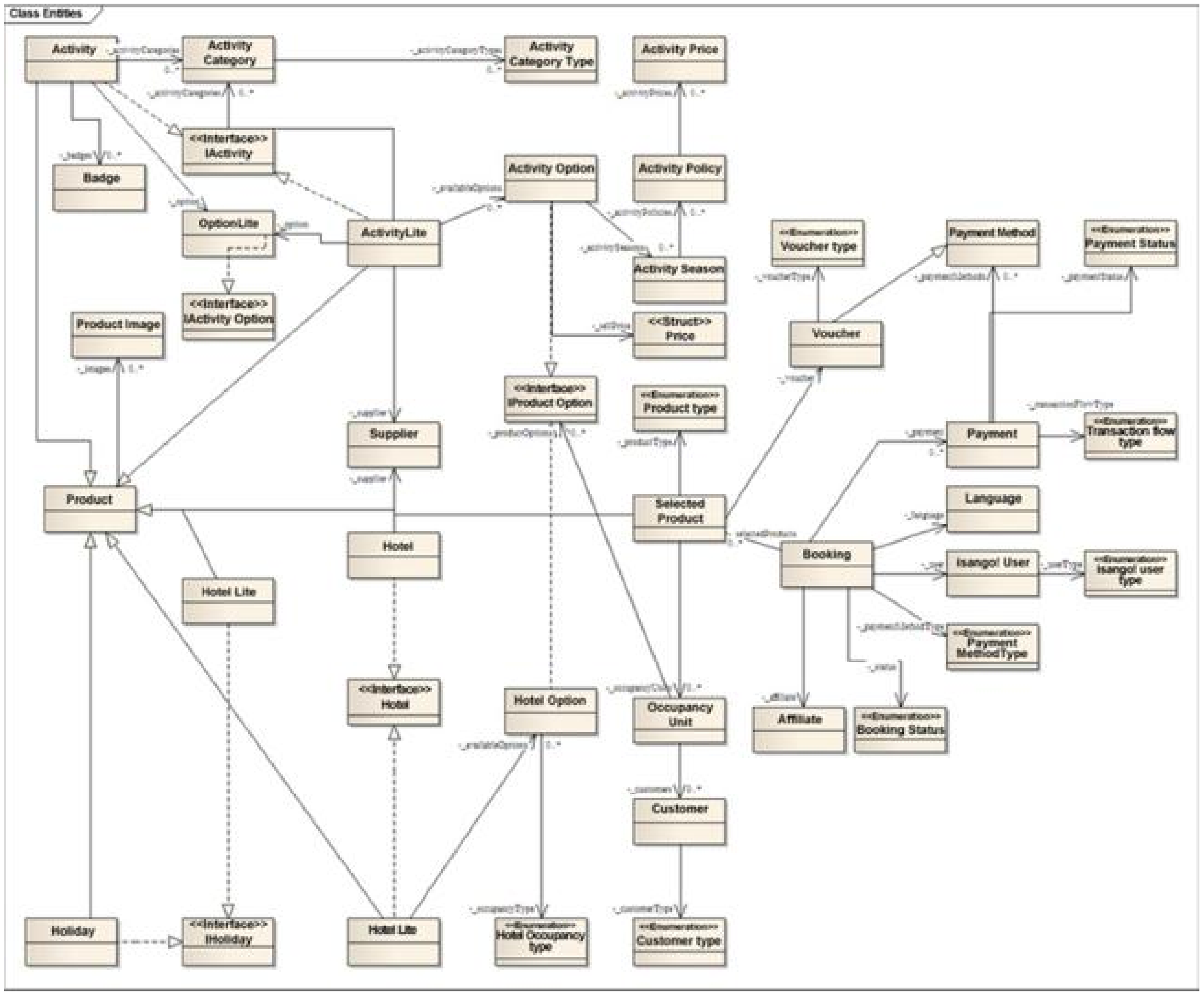
CONCEPTUAL ARCHITECTURE



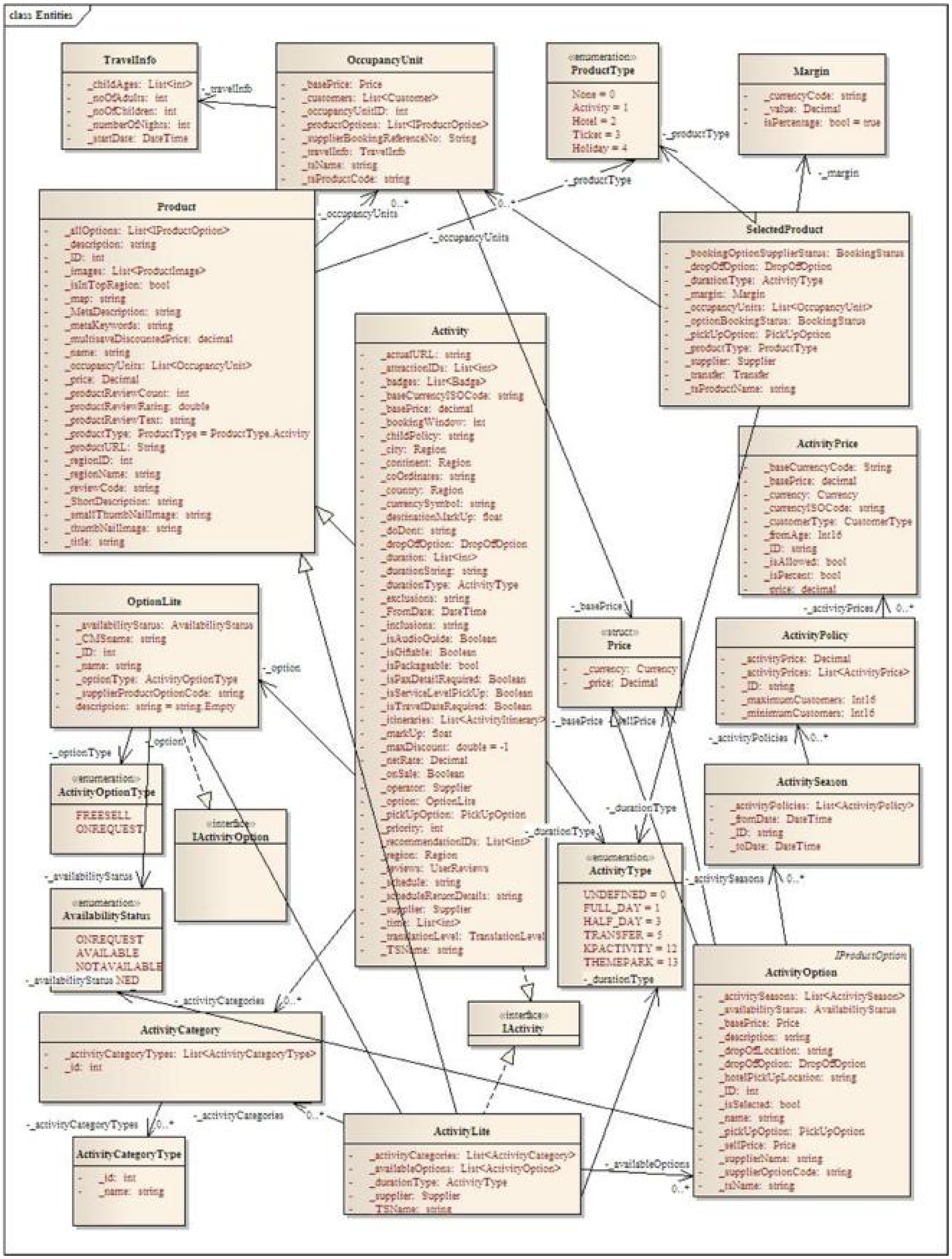
HIGH LEVEL WEBSITE ARCHITECTURE (BY LINE OF BUSINESS)



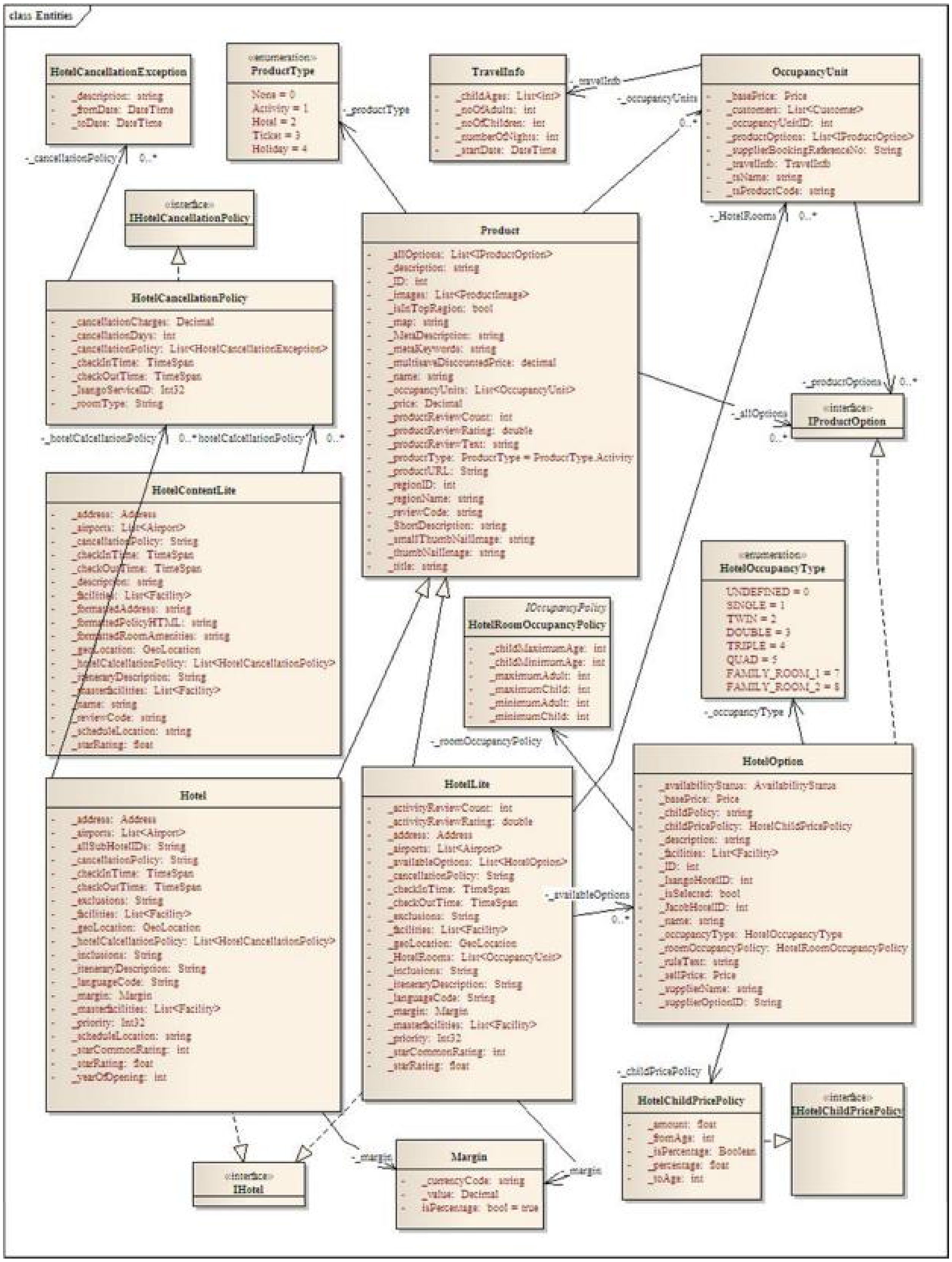
DOMAIN MODEL



ACTIVITY DETAILED CLASS DIAGRAMS

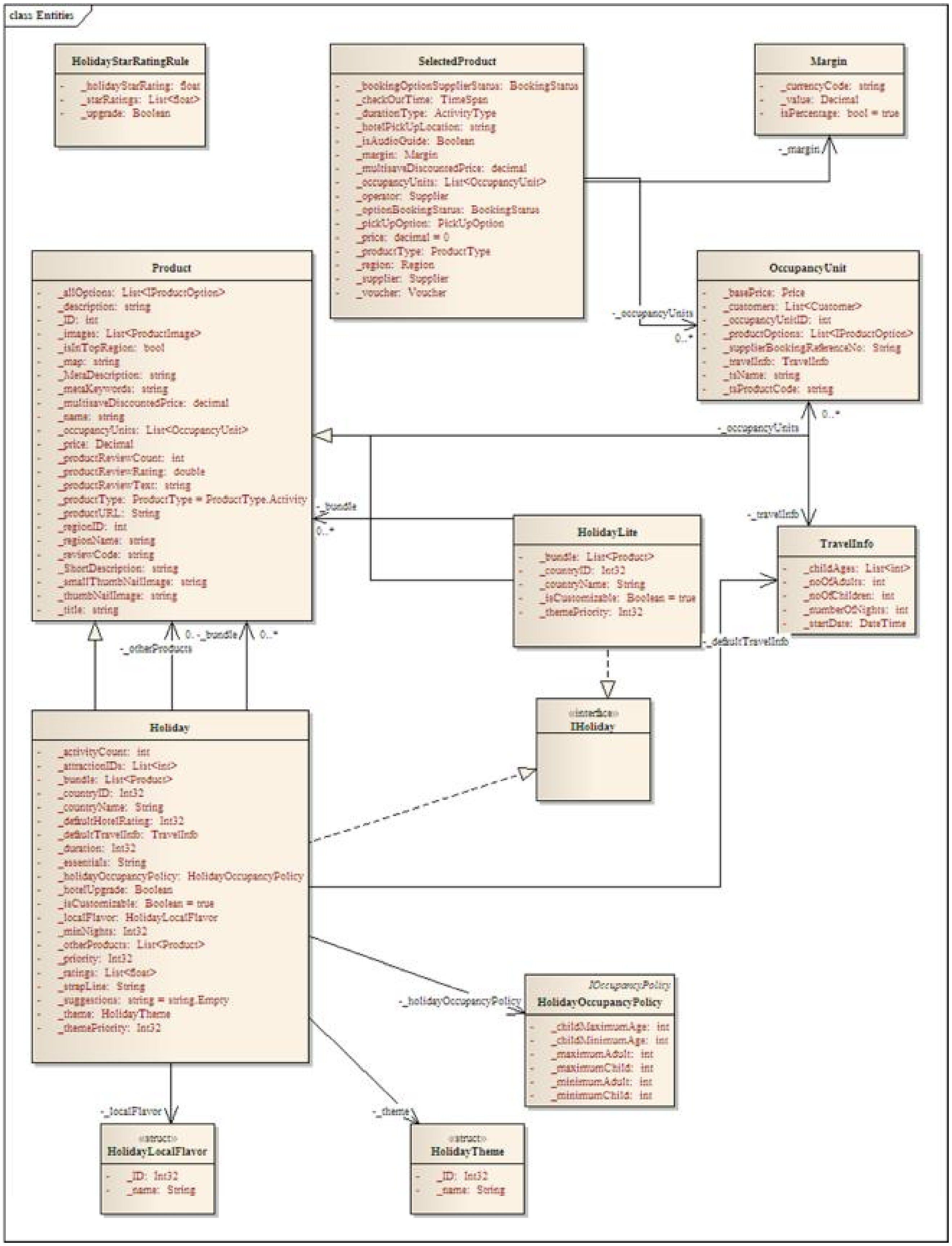


HOTEL DETAILED CLASS DIAGRAMS

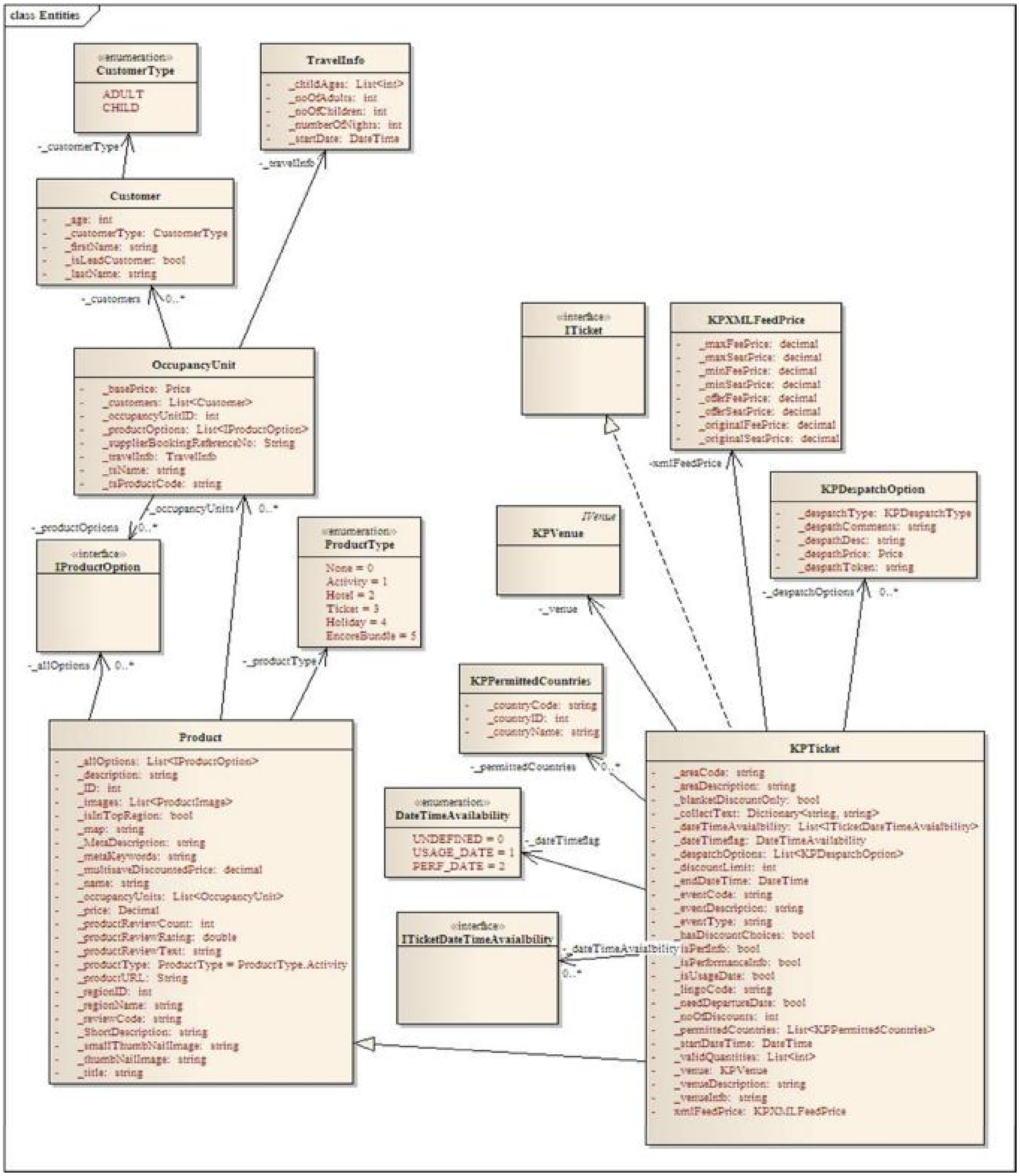


class Entities
HotelCancellationException
- _description: string
- _fromDate: DateTime
- _toDate: DateTime
«enumeration» ProductType
None = 0
Activity = 1
Hotel = 2
Ticket = 3
Holiday = 4
-_productType
TravelInfo
- _childAges: List<int>
- _noOfAdults: int
- _noOfChildren: int
- _numberOfNights: int
- _startDate: DateTime
-_travelInfo
-_occupancyUnits
0..*
OccupancyUnit
- _basePrice: Price
- _customers: List<Customer>
- _occupancyUnitID: int
- _productOptions: List<IProductOption>
- _supplierBookingReferenceNo: String
- _travelInfo: TravelInfo
- _tsName: string
- _tsProductCode: string
-_cancellationPolicy
0..*
«interface» IHotelCancellationPolicy
Product
- _allOptions: List<IProductOption>
- _description: string
- _ID: int
- _images: List<ProductImage>
- _isInTopRegion: bool
- _map: string
- _MetaDescription: string
- _metaKeywords: string
- _multisaveDiscountedPrice: decimal
- _name: string
- _occupancyUnits: List<OccupancyUnit>
- _price: Decimal
- _productReviewCount: int
- _productReviewRating: double
- _productReviewText: string
- _productType: ProductType = ProductType.Activity
- _productURL: String
- _regionID: int
- _regionName: string
- _reviewCode: string
- _ShortDescription: string
- _smallThumbNailImage: string
- _thumbNailImage: string
- _title: string
-_HotelRooms
0..*
HotelCancellationPolicy
- _cancellationCharges: Decimal
- _cancellationDays: int
- _cancellationPolicy: List<HotelCancellationException>
- _checkInTime: TimeSpan
- _checkOutTime: TimeSpan
- _IsangoServiceID: Int32
- _roomType: String
-_productOptions
0..*
-_allOptions
0..*
«interface» IProductOption
-_hotelCalcellationPolicy
0..*
hotelCalcellationPolicy
0..*
HotelContentLite
- _address: Address
- _airports: List<Airport>
- _cancellationPolicy: String
- _checkInTime: TimeSpan
- _checkOutTime: TimeSpan
- _description: string
- _facilities: List<Facility>
- _formattedAddress: string
- _formattedPolicyHTML: string
- _formattedRoomAmenities: string
- _geoLocation: GeoLocation
- _hotelCalcellationPolicy: List<HotelCancellationPolicy>
- _itineraryDescription: String
- _masterfacilities: List<Facility>
- _name: string
- _reviewCode: string
- _scheduleLocation: string
- _starRating: float
«enumeration» HotelOccupancyType
UNDEFINED = 0
SINGLE = 1
TWIN = 2
DOUBLE = 3
TRIPLE = 4
QUAD = 5
FAMILY_ROOM_1 = 7
FAMILY_ROOM_2 = 8
IOccupancyPolicy
HotelRoomOccupancyPolicy
- _childMaximumAge: int
- _childMinimumAge: int
- _maximumAdult: int
- _maximumChild: int
- _minimumAdult: int
- _minimumChild: int
-_occupancyType
-_roomOccupancyPolicy
Hotel
- _address: Address
- _airports: List<Airport>
- _allSubHotelIDs: String
- _cancellationPolicy: String
- _checkInTime: TimeSpan
- _checkOutTime: TimeSpan
- _exclusions: String
- _facilities: List<Facility>
- _geoLocation: GeoLocation
- _hotelCalcellationPolicy: List<HotelCancellationPolicy>
- _inclusions: String
- _itineraryDescription: String
- _languageCode: String
- _margin: Margin
- _masterfacilities: List<Facility>
- _priority: Int32
- _scheduleLocation: string
- _starCommonRating: int
- _starRating: float
- _yearOfOpening: int
HotelLite
- _activityReviewCount: int
- _activityReviewRating: double
- _address: Address
- _airports: List<Airport>
- _availableOptions: List<HotelOption>
- _cancellationPolicy: String
- _checkInTime: TimeSpan
- _checkOutTime: TimeSpan
- _exclusions: String
- _facilities: List<Facility>
- _geoLocation: GeoLocation
- _HotelRooms: List<OccupancyUnit>
- _inclusions: String
- _itineraryDescription: String
- _languageCode: String
- _margin: Margin
- _masterfacilities: List<Facility>
- _priority: Int32
- _starCommonRating: int
- _starRating: float
-_availableOptions
0..*
HotelOption
- _availabilityStatus: AvailabilityStatus
- _basePrice: Price
- _childPolicy: string
- _childPricePolicy: HotelChildPricePolicy
- _description: string
- _facilities: List<Facility>
- _ID: int
- _IsangoHotelID: int
- _isSelected: bool
- _JacobHotelID: int
- _name: string
- _occupancyType: HotelOccupancyType
- _roomOccupancyPolicy: HotelRoomOccupancyPolicy
- _rulsText: string
- _sellPrice: Price
- _supplierName: string
- _supplierOptionID: String
-_childPricePolicy
HotelChildPricePolicy
- _amount: float
- _fromAge: int
- _isPercentage: Boolean
- _percentage: float
- _toAge: int
«interface» IHotelChildPricePolicy
«interface» IHotel
-_margin
Margin
- _currencyCode: string
- _value: Decimal
- isPercentage: bool = true
-_margin

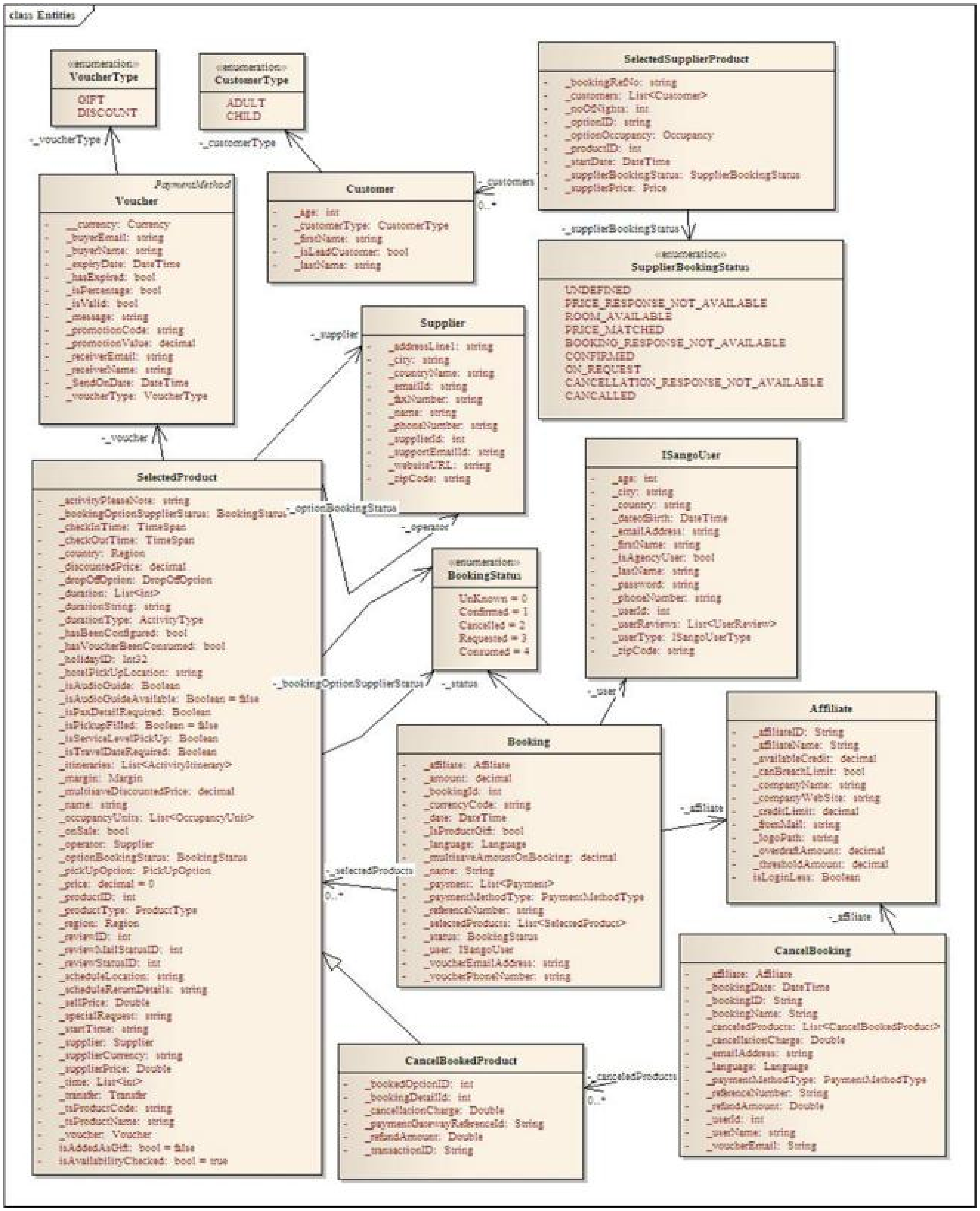
HOLIDAY DETAILED CLASS DIAGRAMS



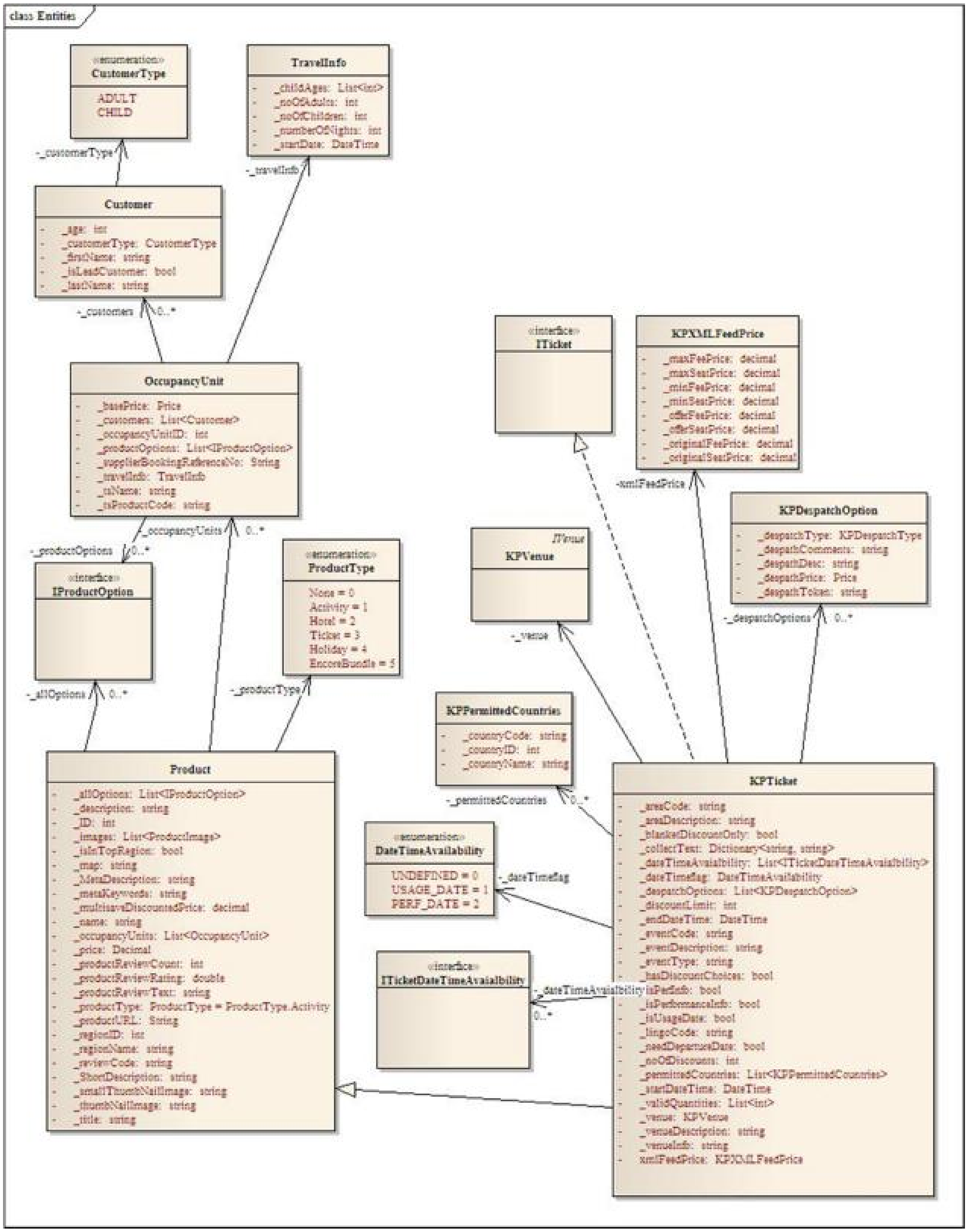
THEATE TICKET DETAILED CLASS DIAGRAMS



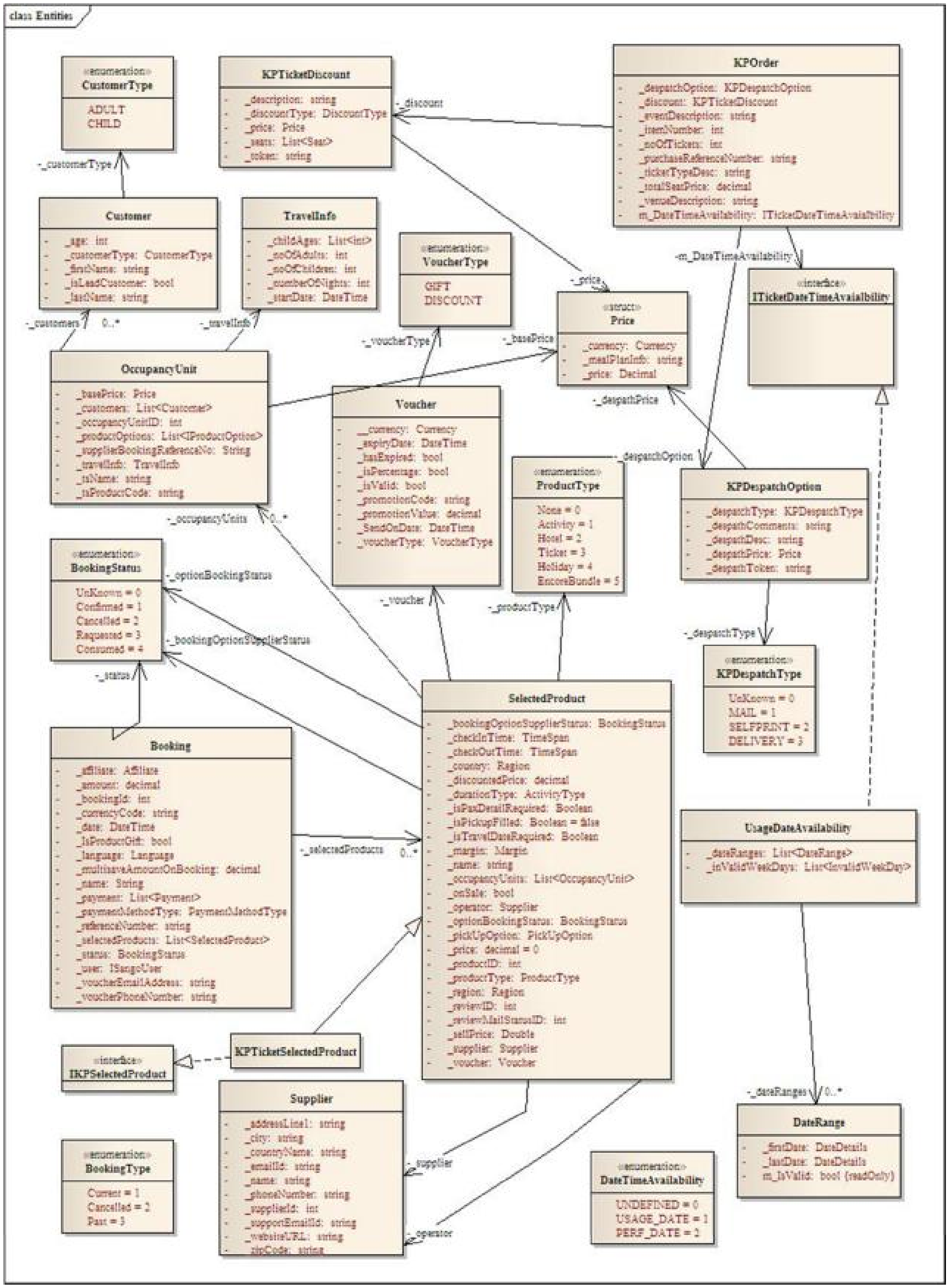
BOOKING DETAILED CLASS DIAGRAMS



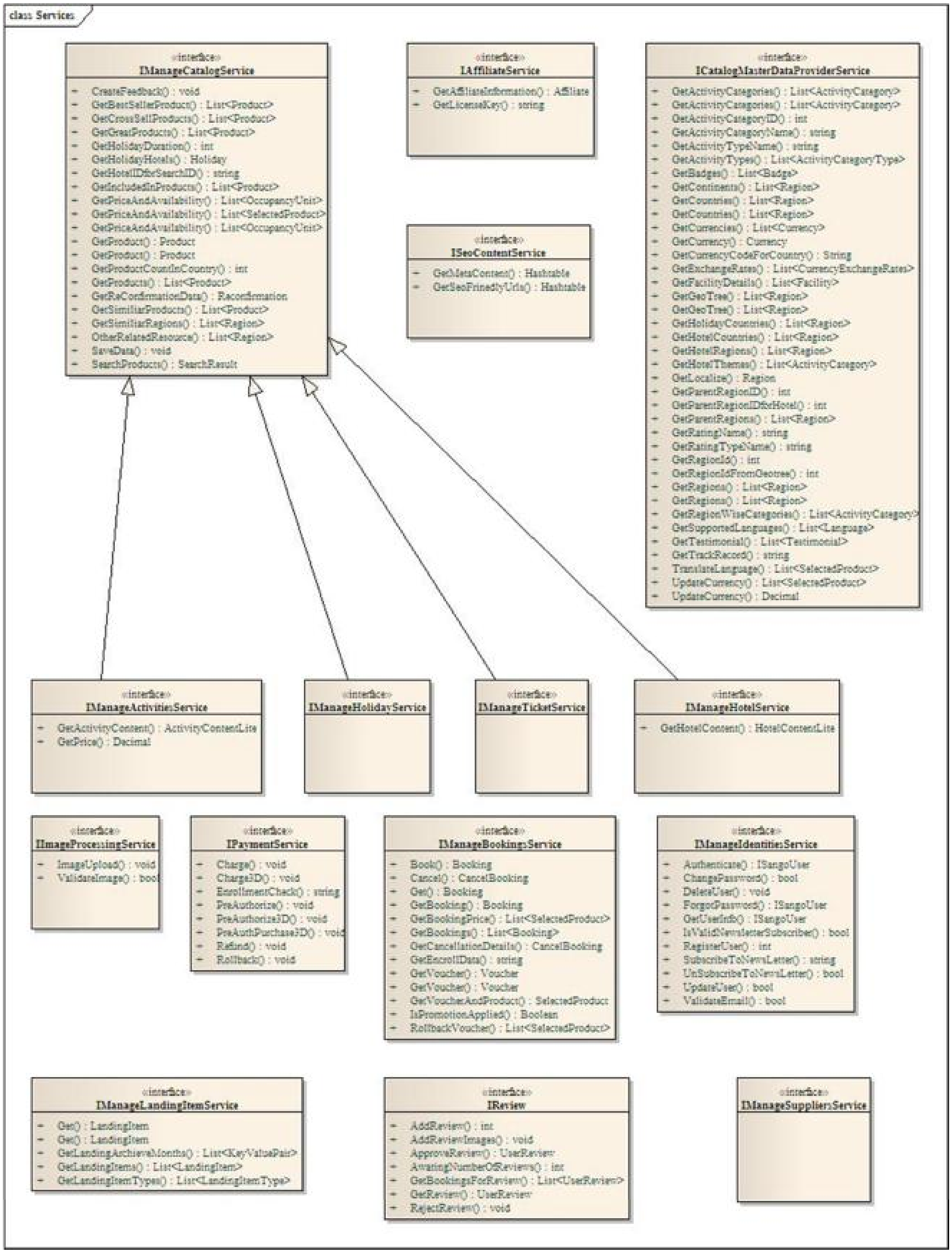
THEATE TICKET BOOKING



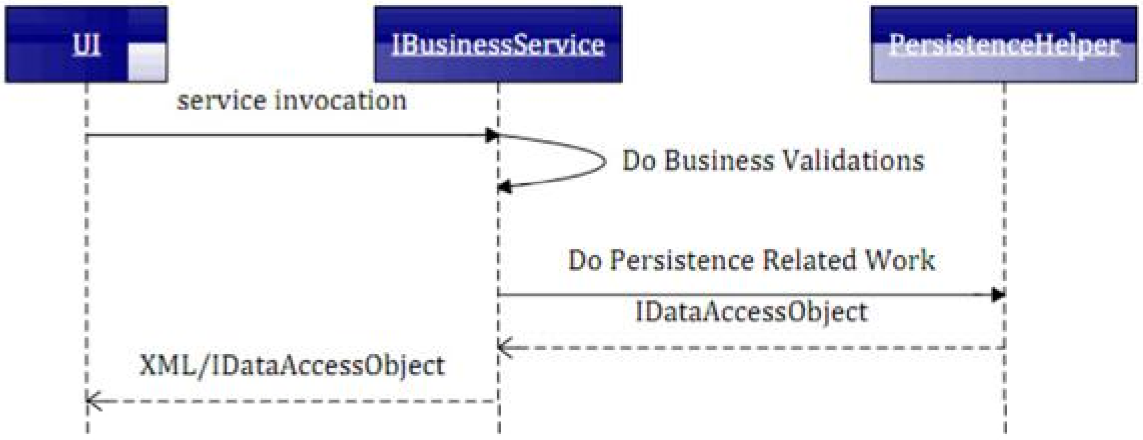
THEATRE TICKET BOOKING – SUPPLIER



BUSINESS SERVICES

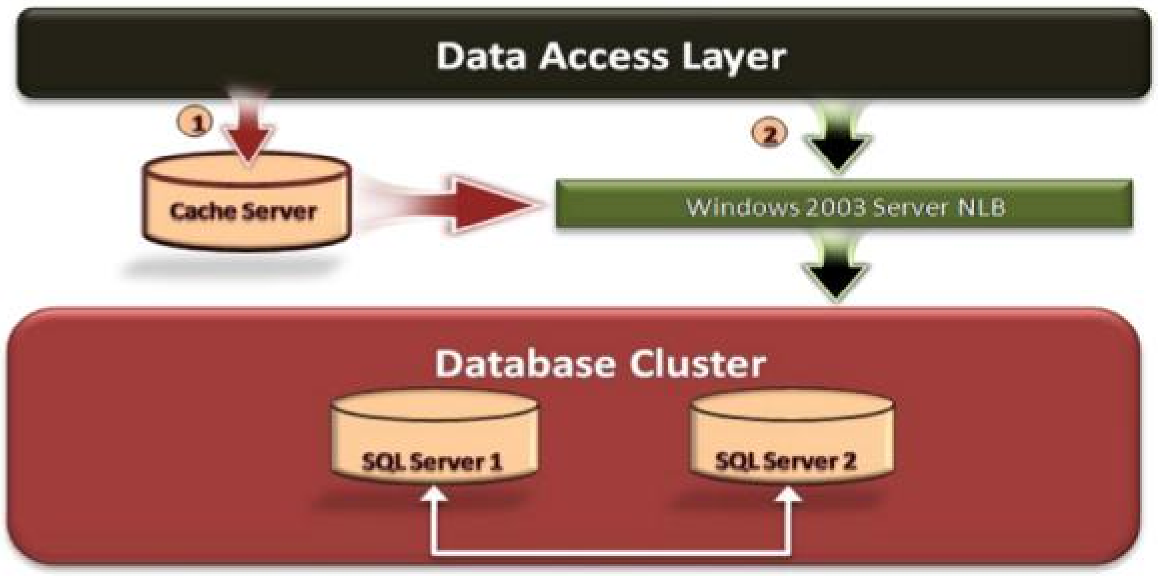

class Services
«interface» IManageCatalogService
- CreateFeedback() : void
- GetBestSellerProduct() : List<Product>
- GetCrossSellProducts() : List<Product>
- GetGreatProducts() : List<Product>
- GetHolidayDuration() : int
- GetHolidayHotels() : Holiday
- GetHotelIDforSearchID() : string
- GetIncludedInProducts() : List<Product>
- GetPriceAndAvailability() : List<OccupancyUnit>
- GetPriceAndAvailability() : List<SelectedProduct>
- GetPriceAndAvailability() : List<OccupancyUnit>
- GetProduct() : Product
- GetProduct() : Product
- GetProductCountInCountry() : int
- GetProducts() : List<Product>
- GetReConfirmationData() : Reconfirmation
- GetSimiliarProducts() : List<Product>
- GetSimiliarRegions() : List<Region>
- OtherRelatedResource() : List<Region>
- SaveData() : void
- SearchProducts() : SearchResult
«interface» IAffiliateService
- GetAffiliateInformation() : Affiliate
- GetLicenseKey() : string
«interface» ISeoContentService
- GetMetaContent() : Hashtable
- GetSeoFriendlyUrls() : Hashtable
«interface» ICatalogMasterDataProviderService
- GetActivityCategories() : List<ActivityCategory>
- GetActivityCategories() : List<ActivityCategory>
- GetActivityCategoryID() : int
- GetActivityCategoryName() : string
- GetActivityTypeName() : string
- GetActivityTypes() : List<ActivityCategoryType>
- GetBadges() : List<Badge>
- GetContinents() : List<Region>
- GetCountries() : List<Region>
- GetCountries() : List<Region>
- GetCurrencies() : List<Currency>
- GetCurrency() : Currency
- GetCurrencyCodeForCountry() : String
- GetExchangeRates() : List<CurrencyExchangeRates>
- GetFacilityDetails() : List<Facility>
- GetGeoTree() : List<Region>
- GetGeoTree() : List<Region>
- GetHolidayCountries() : List<Region>
- GetHotelCountries() : List<Region>
- GetHotelRegions() : List<Region>
- GetHotelThemes() : List<ActivityCategory>
- GetLocalize() : Region
- GetParentRegionID() : int
- GetParentRegionIDforHotel() : int
- GetParentRegions() : List<Region>
- GetRatingName() : string
- GetRatingTypeName() : string
- GetRegionId() : int
- GetRegionIdFromGeotree() : int
- GetRegions() : List<Region>
- GetRegions() : List<Region>
- GetRegionWiseCategories() : List<ActivityCategory>
- GetSupportedLanguages() : List<Language>
- GetTestimonial() : List<Testimonial>
- GetTrackRecord() : string
- TranslateLanguage() : List<SelectedProduct>
- UpdateCurrency() : List<SelectedProduct>
- UpdateCurrency() : Decimal
«interface» IManageActivitiesService
- GetActivityContent() : ActivityContentLite
- GetPrice() : Decimal
«interface» IManageHolidayService
«interface» IManageTicketService
«interface» IManageHotelService
- GetHotelContent() : HotelContentLite
«interface» IImageProcessingService
- ImageUpload() : void
- ValidateImage() : bool
«interface» IPaymentService
- Charge() : void
- Charge3D() : void
- EnrollmentCheck() : string
- PreAuthorize() : void
- PreAuthorize3D() : void
- PreAuthPurchase3D() : void
- Refund() : void
- Rollback() : void
«interface» IManageBookingsService
- Book() : Booking
- Cancel() : CancelBooking
- Get() : Booking
- GetBooking() : Booking
- GetBookingPrice() : List<SelectedProduct>
- GetBookings() : List<Booking>
- GetCancellationDetails() : CancelBooking
- GetEncrollData() : string
- GetVoucher() : Voucher
- GetVoucher() : Voucher
- GetVoucherAndProduct() : SelectedProduct
- IsPromotionApplied() : Boolean
- RollbackVoucher() : List<SelectedProduct>
«interface» IManageIdentitiesService
- Authenticate() : ISangoUser
- ChangePassword() : bool
- DeleteUser() : void
- ForgotPassword() : ISangoUser
- GetUserInfo() : ISangoUser
- IsValidNewsletterSubscriber() : bool
- RegisterUser() : int
- SubscribeToNewsLetter() : string
- UnSubscribeToNewsLetter() : bool
- UpdateUser() : bool
- ValidateEmail() : bool
«interface» IManageLandingItemService
- Get() : LandingItem
- Get() : LandingItem
- GetLandingArchieveMonths() : List<KeyValuePair>
- GetLandingItems() : List<LandingItem>
- GetLandingItemTypes() : List<LandingItemType>
«interface» IReview
- AddReview() : int
- AddReviewImages() : void
- ApproveReview() : UserReview
- AwaitingNumberOfReviews() : int
- GetBookingsForReview() : List<UserReview>
- GetReview() : UserReview
- RejectReview() : void
«interface» IManageSuppliersService

The services would follow the principles of Well-Encapsulated Interfaces. The UI would invoke the business services through .NET API or HTTP-GET/POST verbs to realize a user scenario. Following sequence diagram depicts the overall design paradigm which would be used for the entire system



PERSISTENCE LAYER

PRODUCT CATALOG CACHING+CLUSTERING



Since the isango DB interaction is more read intensive, we have decided to pursue a caching option considering the immense performance boost it would provide to the overall system. So, we leveraged an out of process product catalogue cache. This product catalogue cache would store all the products in the memory so that the system doesn't have to invoke expensive database calls to retrieve the product information from TravelStudio and CMS database. The updates would still be invoked on the database directly.

Pros –

High Performance Retrieval: The data access layer would save on making two expensive calls to load the product information. Cache Server would periodically retrieve changes from the product databases. Thus, in effect, the data access layer would only deal with in-memory data for product.

Cons –

Increased Complexity: The data access layer would have to deal with two data sources – cache and the database. But, it's insignificant when compared to overall system efficiency.

PRODUCT CACHE SERVER

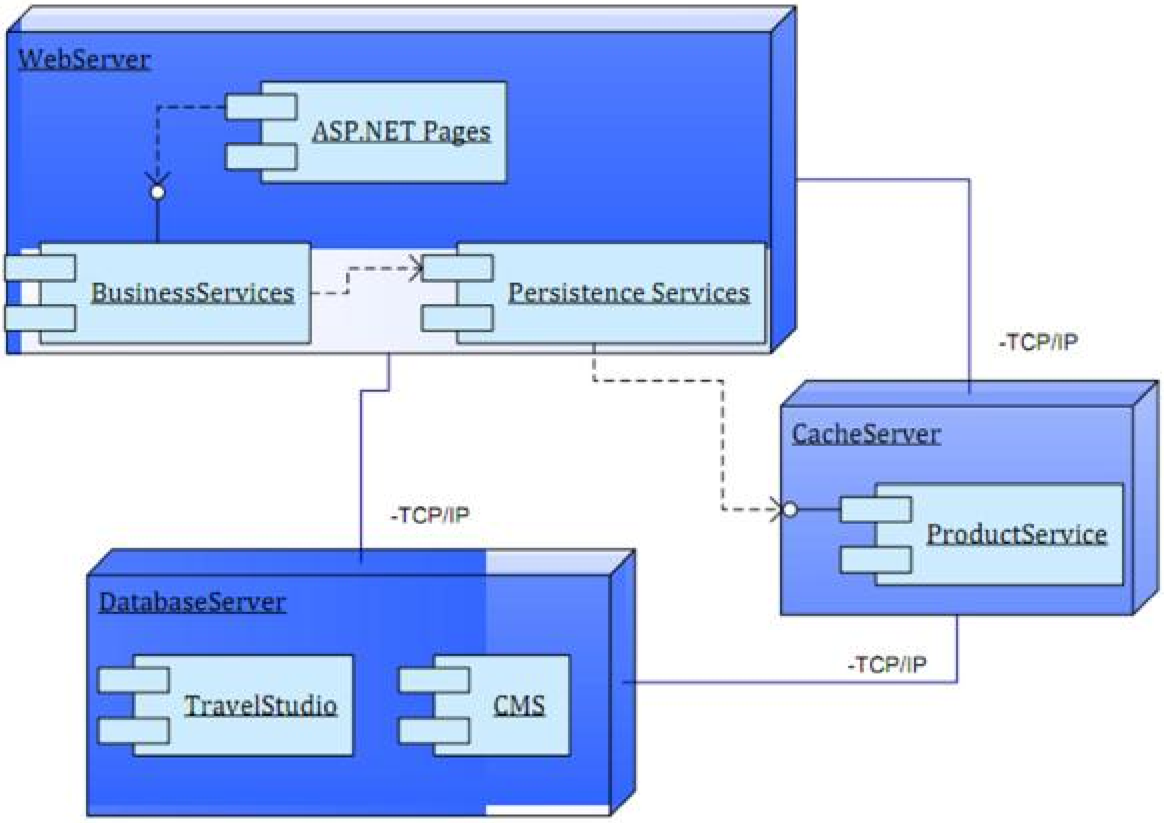
The client applications always look at the product cache and retrieve it from the database if it is not able to find the product in the cache. It also saves the product in the product cache after retrieving it from the database.

The cache would have an upper bound of the products that it could cache thus it would always remove the least used products from the cache.

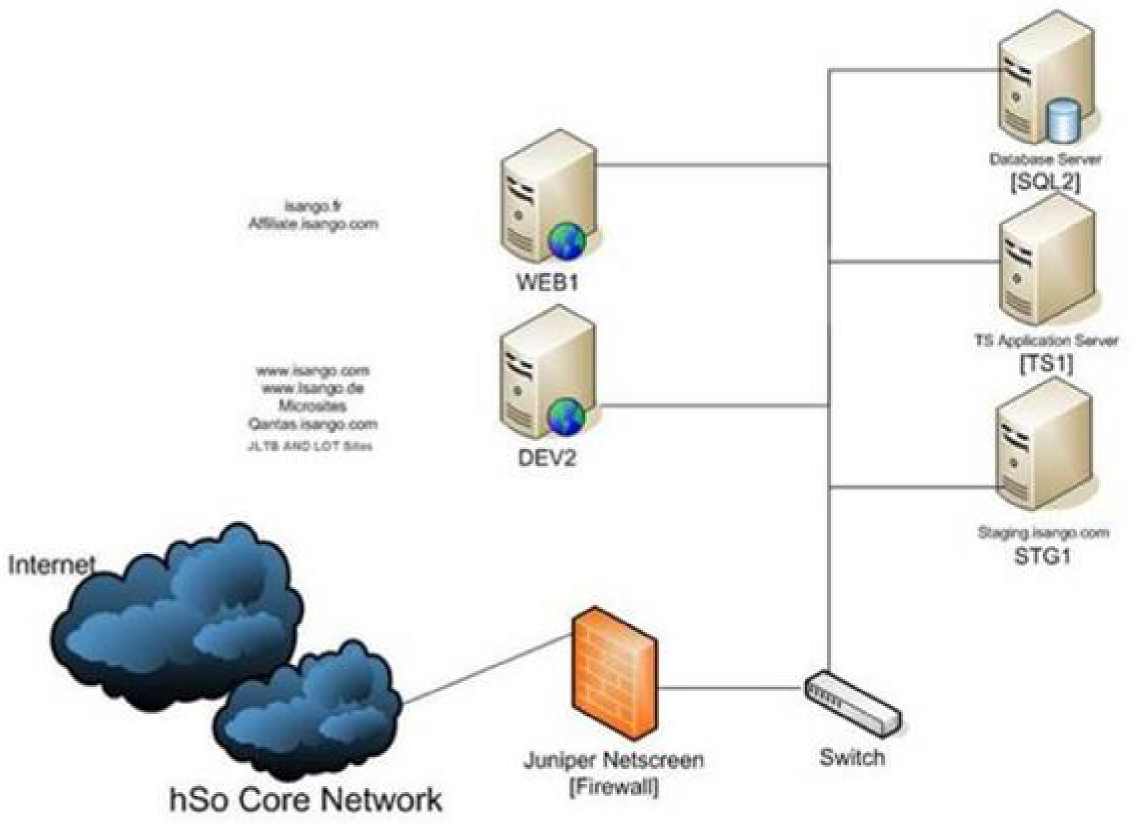
Every time the isango operations team makes a change to the product catalog, we would log the product id and the change operation, like Insert, Update or Delete, that was performed on it.

A windows service would pick up the changes and re-inject the new modified product into the cache.

SOFTWARE DEPLOYMENT

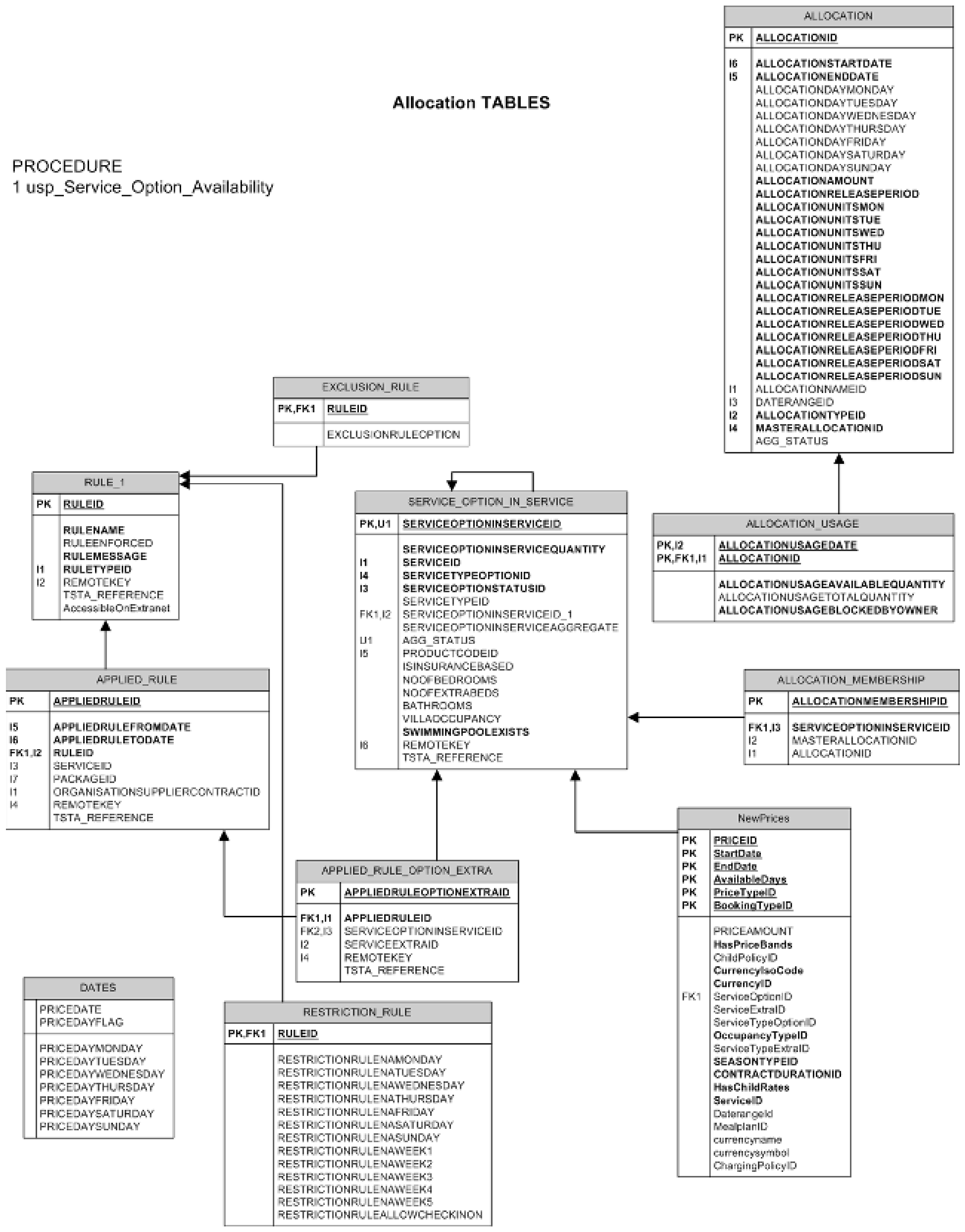


NETWORK DIAGRAM

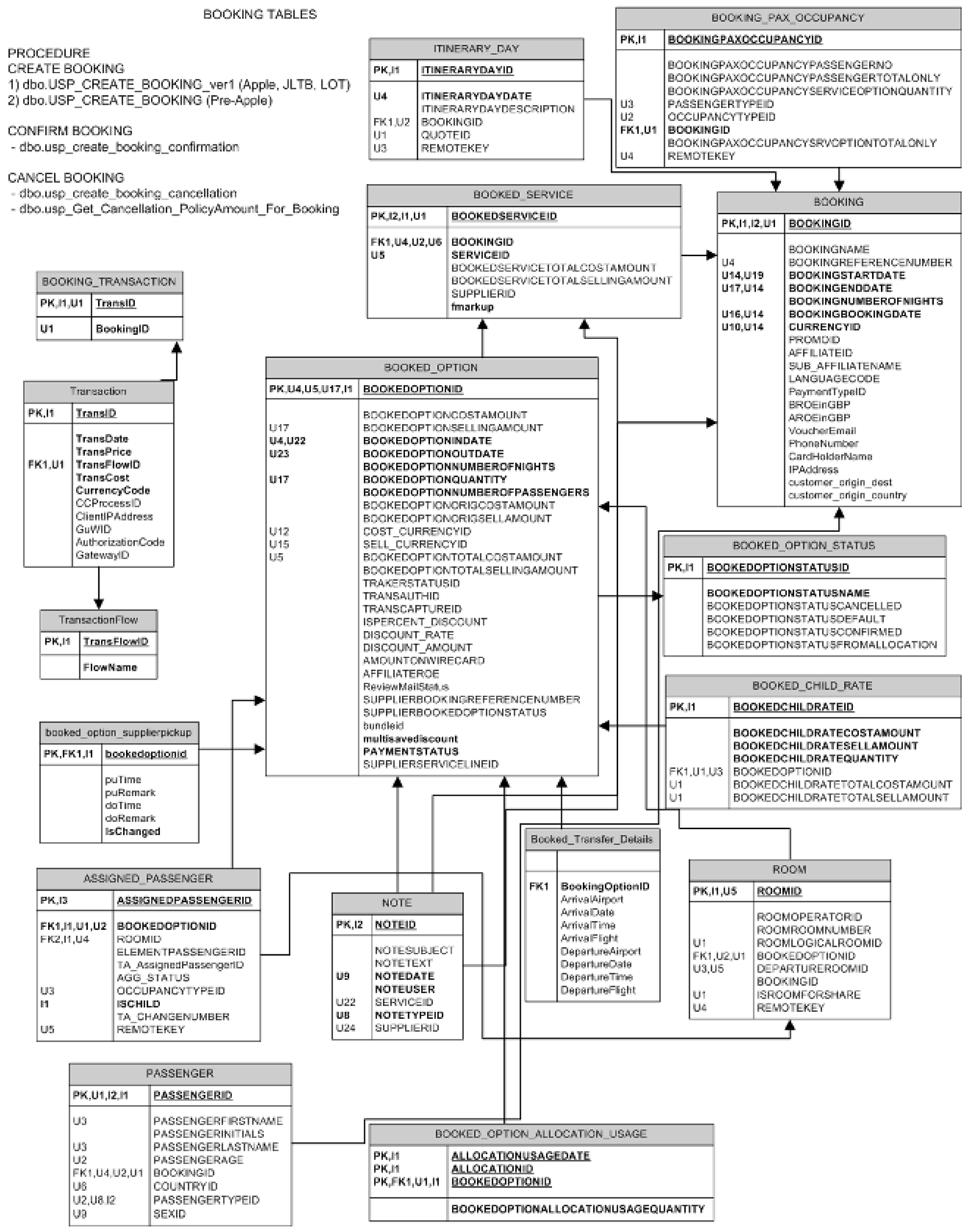


ER DIAGRAMS

PRODUCT AVAILABILITY



BOOKING


BOOKING TABLES

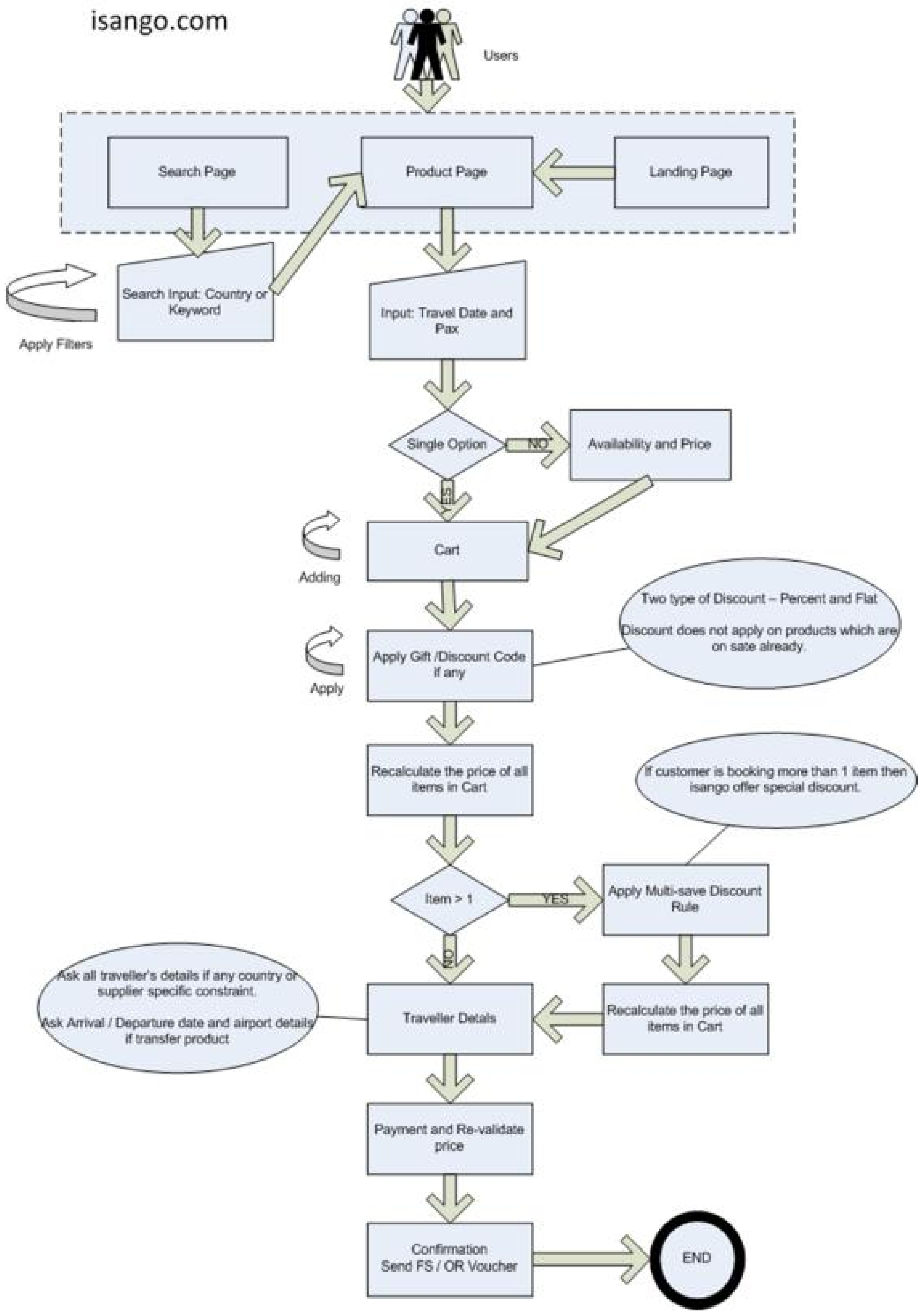
PROCEDURE
CREATE BOOKING
1) dbo.USP_CREATE_BOOKING_ver1 (Apple, JLTB, LOT)
2) dbo.USP_CREATE_BOOKING (Pre-Apple)

CONFIRM BOOKING
- dbo.usp_create_booking_confirmation

CANCEL BOOKING
- dbo.usp_create_booking_cancellation
- dbo.usp_Get_Cancellation_PolicyAmount_For_Booking

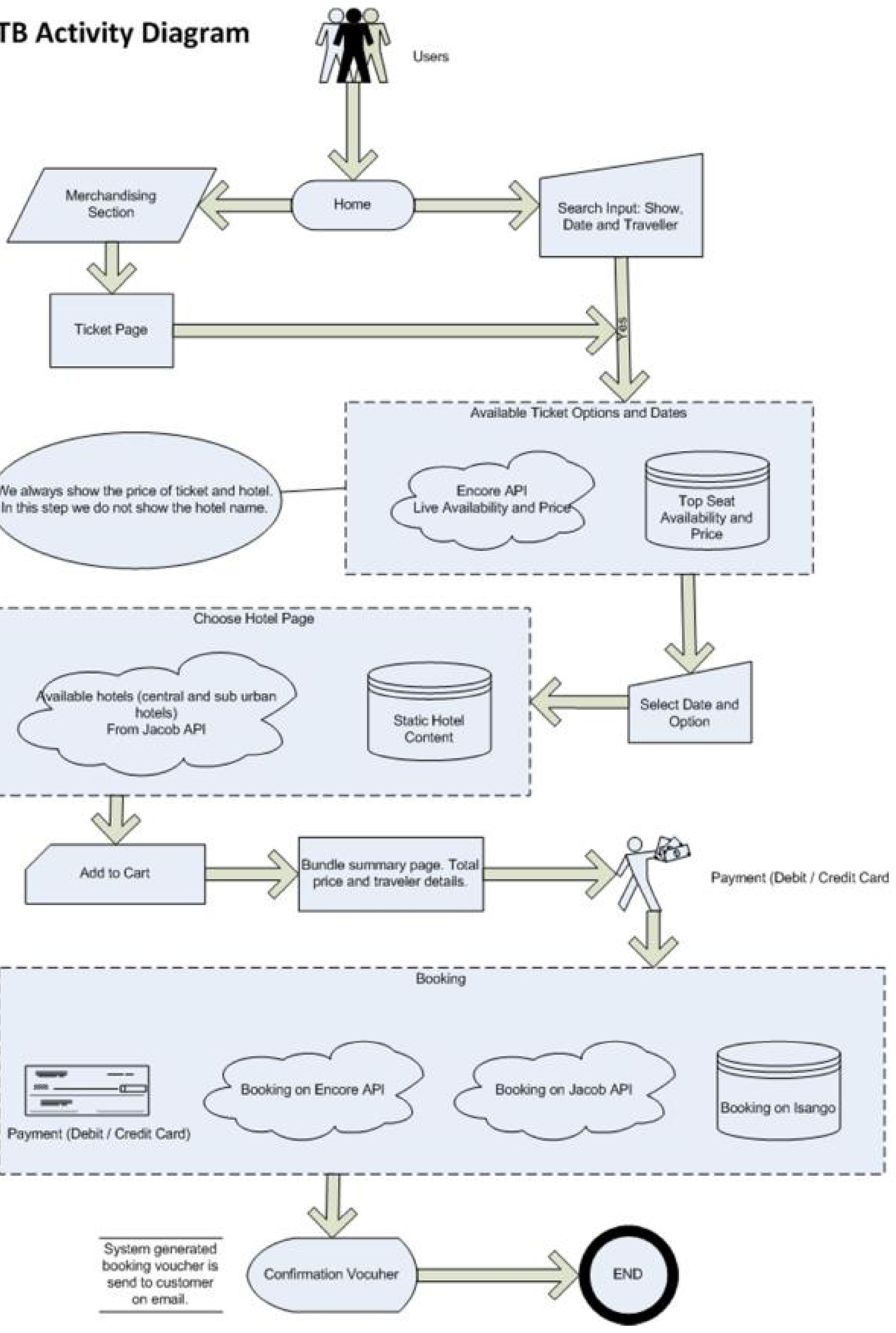
FLOWCHARTS

ISANGO.COM

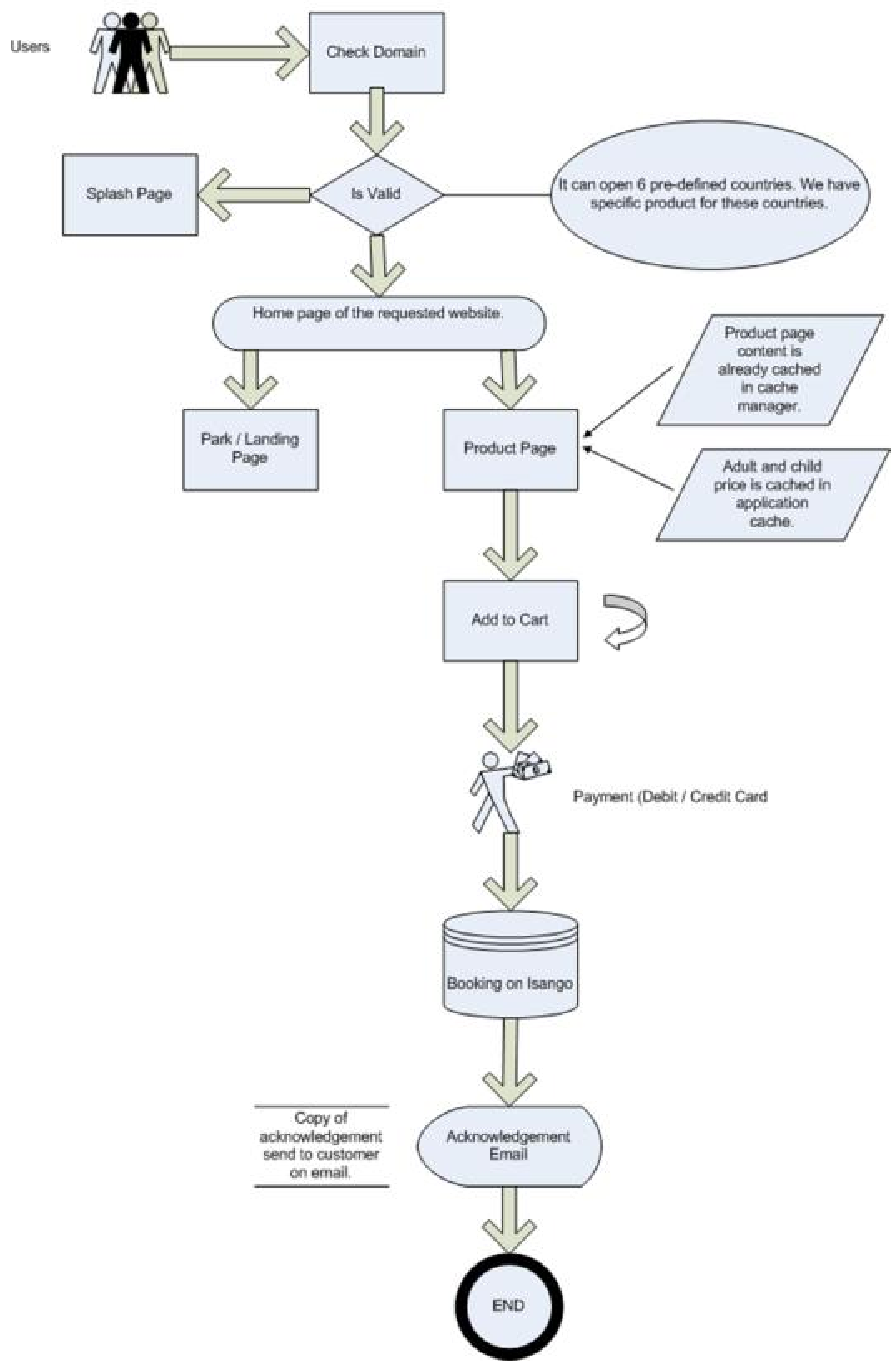


isango.com
Users
Search Page
Product Page
Landing Page
Search Input: Country or Keyword
Apply Filters
Input: Travel Date and Pax
Single Option
NO
Availability and Price
YES
Cart
Adding
Two type of Discount – Percent and Flat
Discount does not apply on products which are on sale already.
Apply Gift /Discount Code if any
Apply
Recalculate the price of all items in Cart
If customer is booking more than 1 item then isango offer special discount.
Item > 1
YES
Apply Multi-save Discount Rule
NO
Ask all traveller's details if any country or supplier specific constraint.
Ask Arrival / Departure date and airport details if transfer product
Traveller Details
Recalculate the price of all items in Cart
Payment and Re-validate price
Confirmation
Send FS / OR Voucher
END

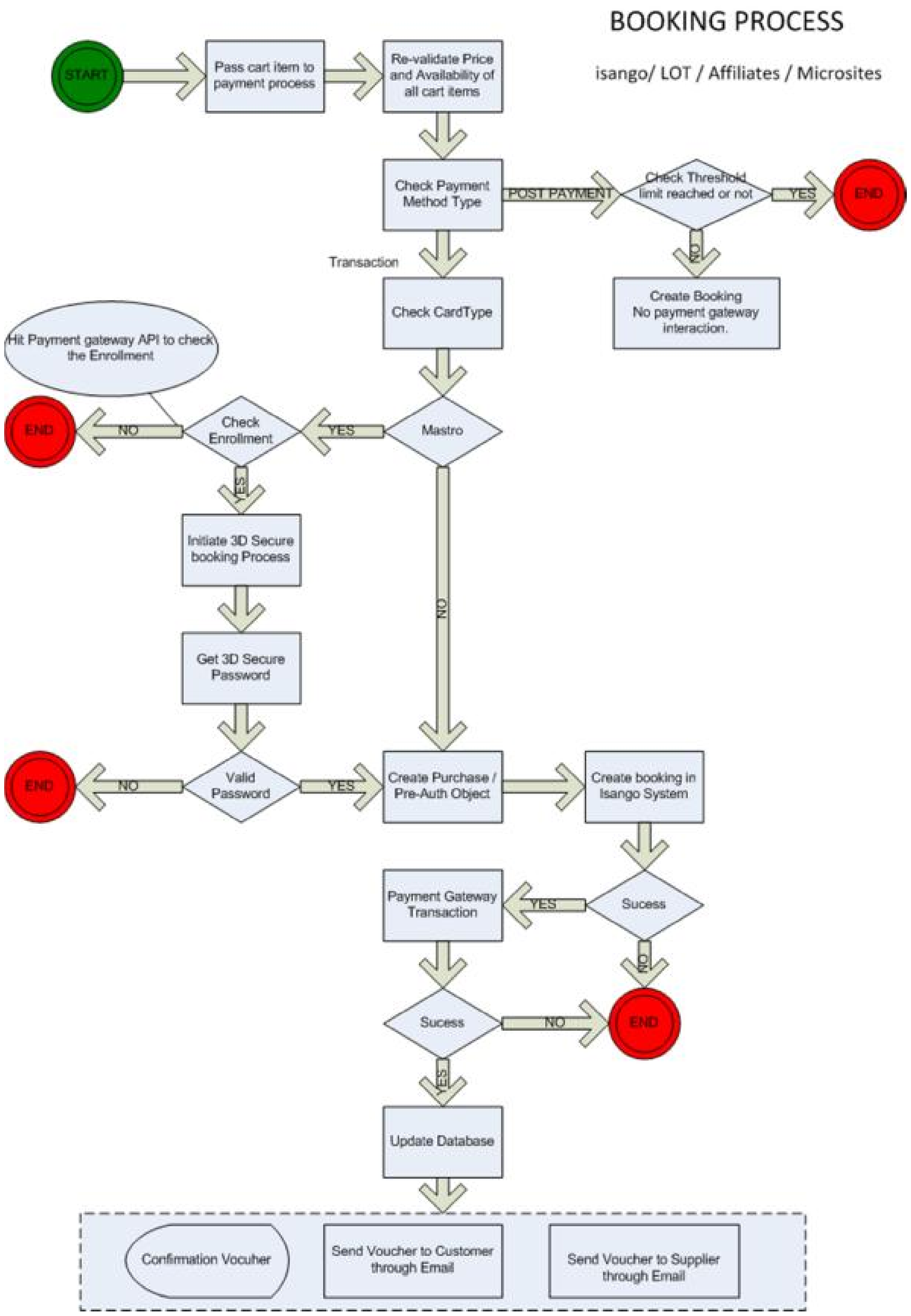
JLTB Activity Diagram


Users
Home
Merchandising Section
Search Input: Show, Date and Traveller
Ticket Page
Yes
Available Ticket Options and Dates
We always show the price of ticket and hotel. In this step we do not show the hotel name.
Encore API Live Availability and Price
Top Seat Availability and Price
Choose Hotel Page
Available hotels (central and sub urban hotels) From Jacob API
Static Hotel Content
Select Date and Option
Add to Cart
Bundle summary page. Total price and traveler details.
Payment (Debit / Credit Card
Booking
Payment (Debit / Credit Card)
Booking on Encore API
Booking on Jacob API
Booking on Isango
System generated booking voucher is send to customer on email.
Confirmation Vocuher
END

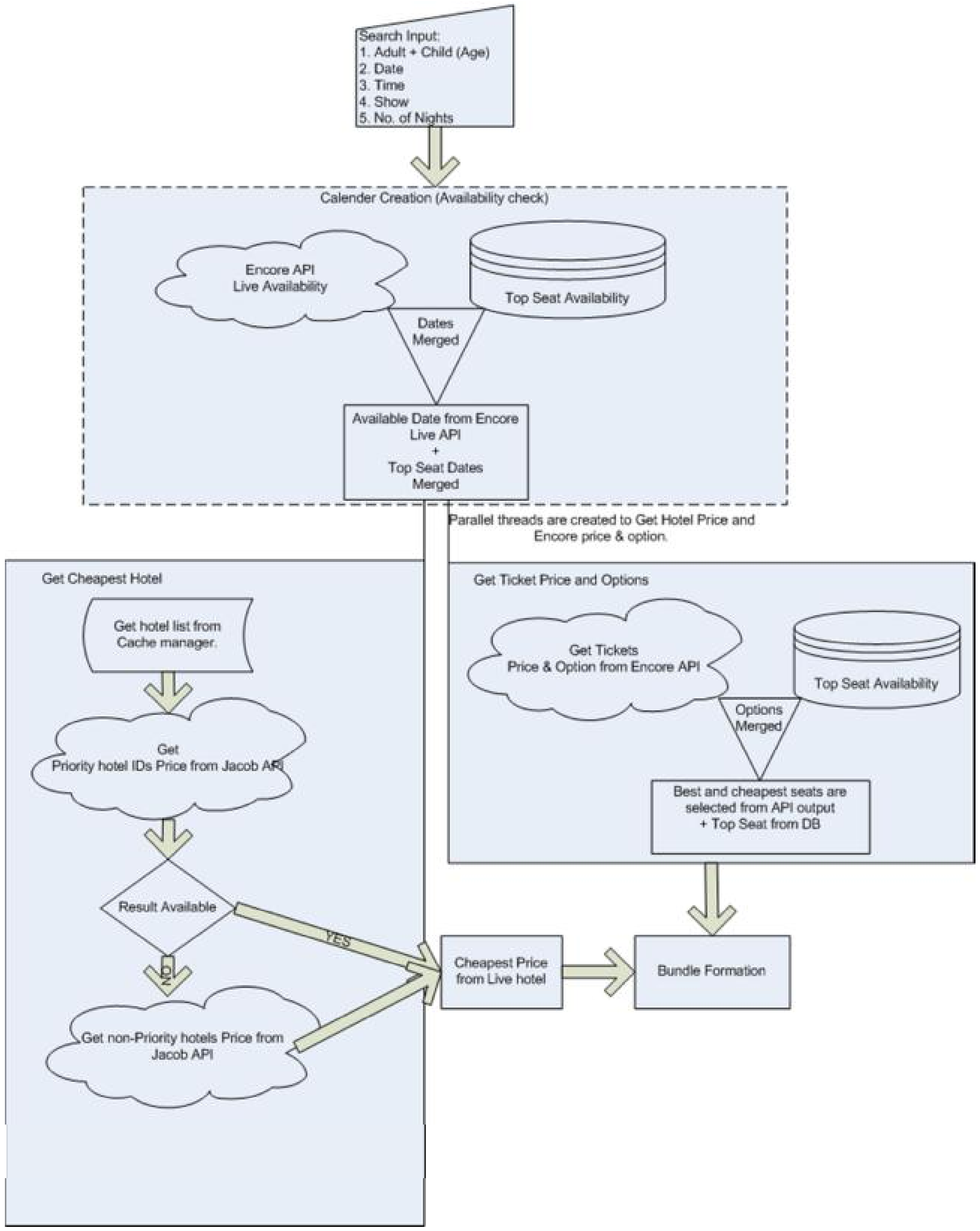
LOVE ORLANDO TICKETS


Users
Check Domain
Splash Page
Is Valid
It can open 6 pre-defined countries. We have specific product for these countries.
Home page of the requested website.
Product page content is already cached in cache manager.
Park / Landing Page
Product Page
Adult and child price is cached in application cache.
Add to Cart
Payment (Debit / Credit Card
Booking on Isango
Copy of acknowledgement send to customer on email.
Acknowledgement Email
END

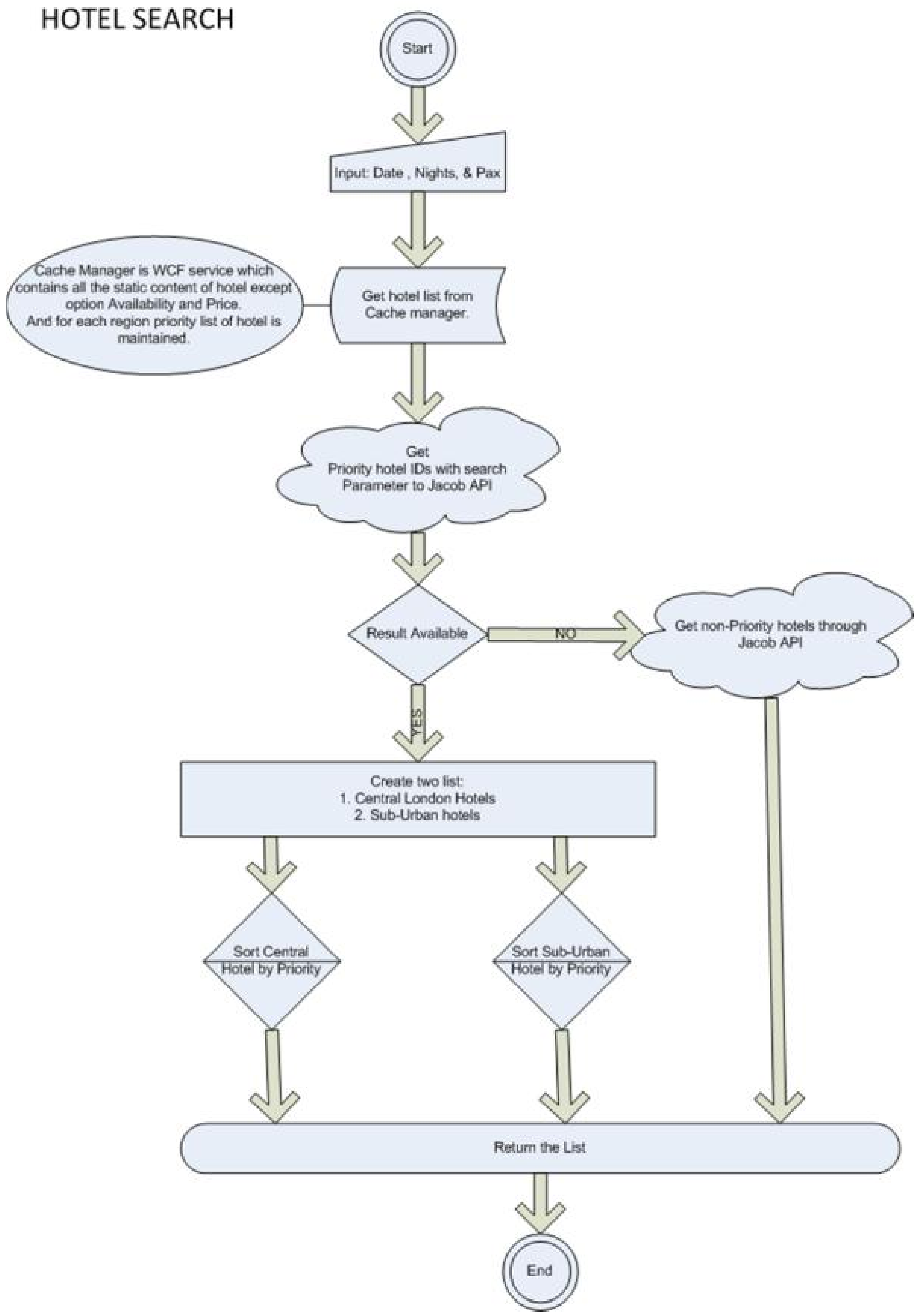
BOOKING


BOOKING PROCESS
isango/ LOT / Affiliates / Microsites
START
Pass cart item to payment process
Re-validate Price and Availability of all cart items
Check Payment Method Type
POST PAYMENT
Check Threshold limit reached or not
YES
END
NO
Create Booking No payment gateway interaction.
Transaction
Check CardType
Hit Payment gateway API to check the Enrollment
END
NO
Check Enrollment
YES
Mastro
YES
Initiate 3D Secure booking Process
NO
Get 3D Secure Password
END
NO
Valid Password
YES
Create Purchase / Pre-Auth Object
Create booking in Isango System
Payment Gateway Transaction
YES
Sucess
NO
Sucess
NO
END
YES
Update Database
Confirmation Vocuher
Send Voucher to Customer through Email
Send Voucher to Supplier through Email

RUNTIME BUNDLE CREATION


Search Input:
1. Adult + Child (Age)
2. Date
3. Time
4. Show
5. No. of Nights
Calender Creation (Availability check)
Encore API Live Availability
Top Seat Availability
Dates Merged
Available Date from Encore Live API + Top Seat Dates Merged
Parallel threads are created to Get Hotel Price and Encore price & option.
Get Cheapest Hotel
Get hotel list from Cache manager.
Get Priority hotel IDs Price from Jacob API
Result Available
YES
NO
Get non-Priority hotels Price from Jacob API
Get Ticket Price and Options
Get Tickets Price & Option from Encore API
Top Seat Availability
Options Merged
Best and cheapest seats are selected from API output + Top Seat from DB
Cheapest Price from Live hotel
Bundle Formation

HOTEL SEARCH


HOTEL SEARCH
Start
Input: Date , Nights, & Pax
Cache Manager is WCF service which contains all the static content of hotel except option Availability and Price.
And for each region priority list of hotel is maintained.
Get hotel list from Cache manager.
Get
Priority hotel IDs with search Parameter to Jacob API
Result Available
NO
Get non-Priority hotels through Jacob API
YES
Create two list:
1. Central London Hotels
2. Sub-Urban hotels
Sort Central Hotel by Priority
Sort Sub-Urban Hotel by Priority
Return the List
End