Bask-it: Technical and User Documentation

Bask-it

June 6, 2025

Contents

1 Introduction

This document provides both technical and user documentation for the Bask-it payment processing application. Bask-it allows users to add items from any website to their basket and pay for all the items at once during checkout.

2 System Overview: Get to know Bask-t

Bask-it allows seamless integration with various merchants and payment gateways. The system consists of several key components including user accounts, baskets, transactions, and third-party integrations.

3 User Documentation: User Guide, Getting Started, Features, Troubleshooting, FAQs.

To get started with Bask-it, follow these steps:

1. 1. Getting Started

1. Create an Account:

1. Go to https://www.bask-it.com/
2. Install the Bask-it browser plug-in or extension compatible with your web browser.
3. Go to https://www.bask-it.com/signup.
4. Enter your details: name, email address, and password.
5. Click "Sign Up."

1. Adding Items to Your Basket:

1. Browse any supported website and add items to your Bask-it basket.
2. Once you are ready, open the Bask-it extension and visit the checkout tab to view your basket.
3. Once you're ready, visit the Bask-it checkout page to view your basket.
4. Select Checkout.

1. 2. Payment Process

1. Choose Your Payment Method:

2. Bask-it supports several payment methods including credit/debit cards, PayPal, and bank transfers.

3. Select Pay.

1. Review Your Order:

1. Review the items in your basket and shipping details.

2. Apply any discounts or promotional codes before proceeding.

3. Select Pay/Checkout.

4. After you confirm your basket, the system securely processes your payment.

4 Technical Documentation: Architecture Overview, API Documentation, Database Schema, Error Codes, Developer Setup.

1. System architecture diagram: A high-level system architecture diagram would greatly enhance the technical documentation. It could show the relationships between the backend, frontend, database, and any third-party services.

1. API documentation: If the system offers an API, include a dedicated section with endpoint documentation, request/response formats, authentication mechanisms, and examples.

1. Endpoint documentation: request/response formats, authentication mechanisms, and examples.

1. Code examples: Include code snippets demonstrating key functionality or common tasks. These examples should be simple yet representative of real-world usage.

1. Error handling and Troubleshooting: List common error messages or issues users may encounter, along with solutions or workarounds.

1. Glossary and Definitions: Include a glossary of terms, especially for technical terms or acronym

1. Faqs: Questions that have already been asked and answered.

1. Database schema and entity relationship diagram (ERD): Make sure the ERD diagram is clear and easy to understand, especially for users unfamiliar with database design. You might want to consider adding descriptions of each entity and the relationships between them.

1. Database table descriptions: Along with the ERD, include a section that describes each database table and its attributes. For instance, what data each column holds, and what relationships exist between tables.

1. Consistency and formatting:

1. Consistent terminology: Ensure that you consistently use the same terms throughout the documentation. For example, decide whether you'll refer to "baskets" or "shopping carts" and stick with one term.

1. Code block formatting: Code blocks should be clearly formatted and distinguished from regular text, and use proper syntax highlighting.

1. User and Developer Personas

1. Audience Awareness: Tailor the language and details based on the user and developer personas.

1. Versioning and Updates

1. Version History: Include a version history section (either in the document footer or in a separate section) to track changes, new features, or bug fixes in each release.

1. Appendices and Resources

1. Appendix with Code samples Appendix with Code Samples: If there are important code samples, include them in the appendix.

1. Additional Resources and References: Provide links to any external resources, tools, or tutorials that might be helpful for users or developers.

1. Feedback and Support

1. Contact Information: Provide clear instructions on how users can get support if they encounter issues (e.g., email, chat support, community forums).

1. Feedback Mechanism: Allow users and developers to provide feedback on the documentation to improve future versions.

1. Accessibility

1. PDF/HTML Versions: Ensure the documentation is available in both PDF and HTML formats, so it's easy to access across devices.

1. Accessibility Considerations: Ensure that the documentation is accessible, with readable fonts, proper contrast, and support for screen readers, especially for the visually impaired.

1. Additional Suggestions

1. Testing Instructions: If your documentation is for developers, include steps to test the system, including unit tests, integration tests, or mock environments.

1. Performance Considerations: If relevant, include sections that highlight performance-related aspects of the application such as scaling, optimization, and caching strategies.

5 Entity Relationship Diagram

5.1 ERD Overview

The ERD below provides a visual representation of the database structure, including entities, their attributes, and relationships. This diagram is crucial for understanding how data is organized and how different entities interact within the Bask-it application. Below the digram is a glossary of what each field in the figure represents.

User creates one or more Baskets. Each Basket contains multiple Basket$_{I}tementries, linkingIt$
Entities: All new and existing entities are included:
User, Basket, Item, Basket$_{I}tem, Checkout, Transaction, andPaymentMethod.Relationships$
User creates Basket. Basket contains Basket$_{I}tem.Basket_{I}temincludesItem.BasketleadstoCh$
Each entity includes key attributes, such as user$_{i}d, basket_{i}d, item_{i}d, quantity, status, etc.Layou$

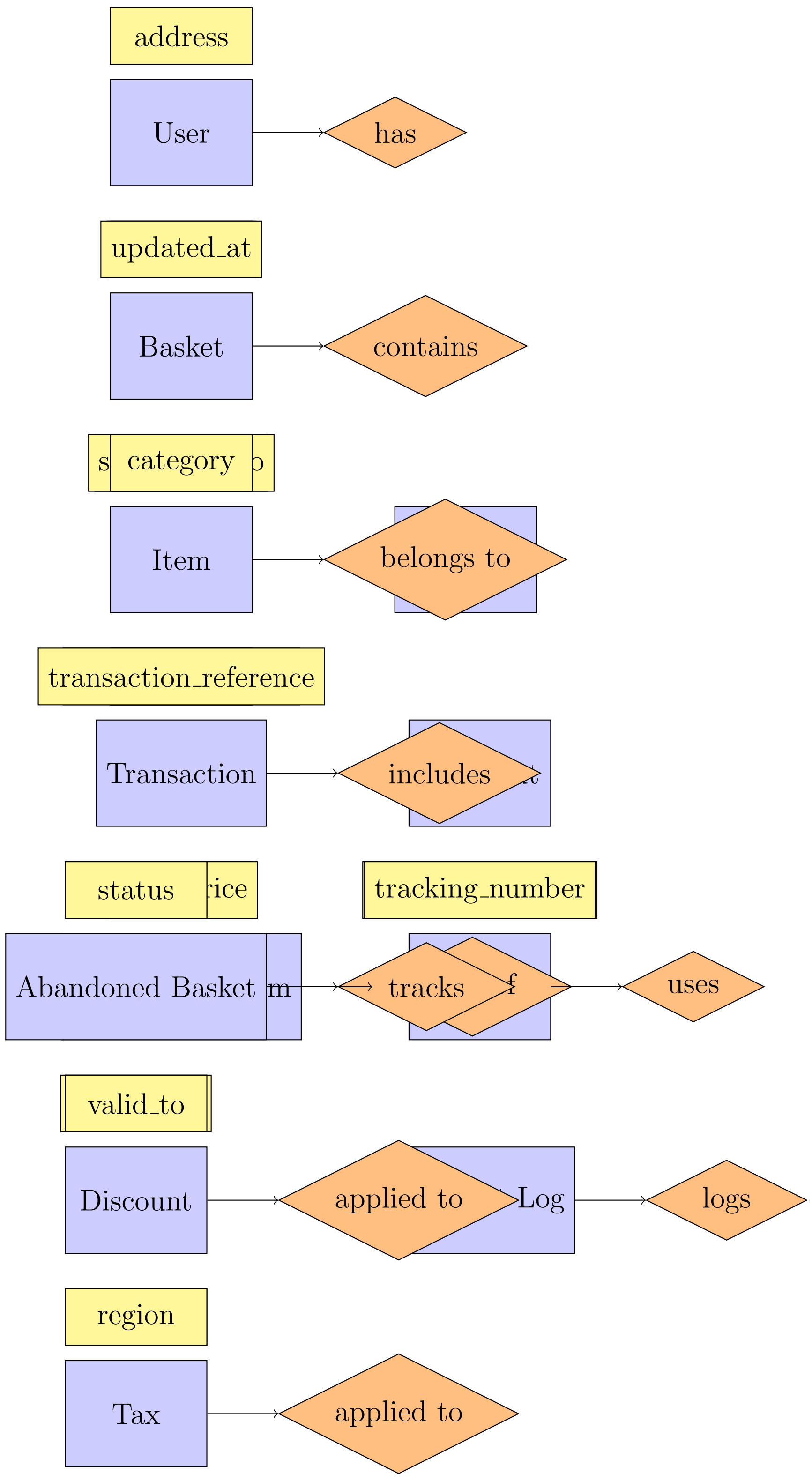

Figure 1: Entity Relationship Diagram for Bask-it System

The entities are spaced and positioned logically for clarity. Attributes are visually linked to their entities.

. User Primary Key: user$_i dAUsercancreatemultipleBaskets(1 - to -$
$Many).AUsercanhavemultiplePaymentMethods(1-to-Many).2.BasketPrimaryKey :$
$basket_idForeignKey : user_id(fromUser)Tracksthetotalnumberofitemsandtotalvalue.Linkedto$
$to-ManyrelationshipswithItems.OneBasketcanlinktooneCheckout(1-to-$
$1).3.ItemPrimaryKey : item_idRepresentsitemsfromanywebsite, withattributeslikeprice, descr$
$to-ManywithBasket).4.Basket_ItemJunctiontablelinkingBasketandItem.Tracksquantityandp$
$checkout_idForeignKey : basket_id(linkstoBasket), user_id(linkstoUser), payment_method_id.Repre$
$transaction_idForeignKey : checkout_id(linkstoCheckout).Recordstheactualpaymentdetailspost-$
$checkout.7.PaymentMethodPrimaryKey : payment_method_idForeignKey :$
$user_id(linkstoUser).Tracksvariouspaymentmethodslikecreditcards, PayPal, orbankaccounts.8$
$gateway_idRepresentsexternalpaymentserviceslikeStripeorPayPal.9.PaymentLogPrimaryKe$
$log_idTrackseachinteractionbetweentheplatformandaPaymentGateway.Logsresponses, status,$

User to Basket:

A user can have multiple baskets (1-to-Many). Each basket belongs to one user. Basket to Item:

A basket can contain multiple items (Many-to-Many via Basket$_Item).BaskettoCheckout :$

A basket can have only one checkout (1-to-1). Checkout to Transaction:

Each checkout links to one transaction (1-to-1). Transaction to Payment Gateway:

Transactions use payment gateways for processing. User to Payment Method:

A user can have multiple payment methods. Transaction to Payment Log:

Each transaction can generate multiple payment logs.

Historical Pricing:

price$_at_additioninBasket_Itemallowstrackingitempricesatthetimeofaddition, ensuringconsis$
$CurrencyandMulti - GatewaySupport :$

currency fields and integration with Payment Gateway ensure flexibility for global users. Payment Status Tracking:

The Transaction entity links to Payment Logs, providing a complete audit trail for payments and gateway interactions. Shipping and Billing:

Checkout includes shipping$_addressandbilling_addressfordetailedhandlingofcustomerdeliveri$

The model supports abandoned baskets, multiple payment methods, and flexible item tracking across various websites.

User Role Customization:

While "customer" and "merchant" are good starting points for user roles, consider adding a separate table for User$_Roles : Thiswouldallowtheapplicationtoscalemoreeasily$

While the stock$_status attributeisuseful, youmaywanttoaddaStock_Itementitytotrackinventor$
$Attributes : item_id, warehouse_id, available_quantity.Thisisusefulforbusinessesthatoperateacro$

The Checkout table includes discount$_{a}pplied, buttheremightbeaneedforaseparateDiscountsta$
$Attributescouldincludediscount_{i}d, code, description, amount_{o}r_{p}ercentage, valid_{f}rom, valid_{t}o, us$

If the system will include customer reviews for items, an Item$_{R}eviewtablemightbeuseful:$
$Attributes: review_{i}d, item_{i}d, user_{i}d, rating, comment, created_{a}t, updated_{a}t.PaymentMethodSec$

For the Payment$_{M}ethodentity: Ensuresensitivefieldslikemethod_{d}etailsareencryptedorstore$

In the Checkout table, shipping can vary based on items. A separate Shipping$_{D}etailstablecouldbeuseful: Attributes: shipping_{i}d, checkout_{i}d, item_{i}d, shipping_{m}ethod,$

If the system needs strong audit trails, additional fields like created$_{b}y, updated_{b}y, oranAudit_{L}og$
$Attributes: log_{i}d, entity_{n}ame, entity_{i}d, change_{d}escription, changed_{b}y, timestamp.$

Payment Gateway Integration:

Should the Payment$_{G}atewaytabletrackfeesorcostspertransaction?Ifso, addattributesfortransa$

Multi-Currency Support:

The Transaction and Item tables include currency, but will exchange rates be handled within the system? If so: A separate Exchange$_{R}atestablemightbenecessary.ItemSourcea$

The website$_{u}rlintheItementityisagreatadditionfortrackingsources.Ifthere'saneedtogroupitem$
$AddaVendortable: vendor_{i}d, name, website_{u}rl.BasketLifecycle:$

Are abandoned baskets going to be analyzed? If so: Consider adding fields like last$_{a}ccessedtoBasketoradedicatedBasket_{L}ogforlifecycletracking.$

User Roles:

A new User$_{R}olestableallowsflexiblerolemanagement(e.g., customer, merchant, admin).2.StockM$

A Stock$_{I}tementitytracksitemquantitiesacrossdifferentwarehouses.3.Discounts:$

Added a Discounts table to manage coupon codes, seasonal offers, and other promotions.

4. Shipping Details:

A Shipping$_{D}etailstableallowstrackingofshippingmethods, costs, andexpecteddeliverydatesfore$

A Vendor table tracks information about the website or platform where items are sourced.

6. Item Reviews:

An Item$_{R}eviewtablesupportscustomerreviewsandratingsforproducts.$

Basket:

Added a last$_{a}ccessedattributetotrackwhenabasketwaslastinteractedwith.Checkout:$

Supports discount application via the Discounts table and integrates shipping information via Shipping$_{D}etails.Payment_{M}ethod:$

Attributes like verification$_{s}tatuscanbederivedaspartofthestatusfield.Transaction:$

Added multi-currency support with potential integration into an Exchange$_{R}atesservice.$

Item:

Attributes added: sku, shipping$_{i}nfo, website_{u}rl, andimage_{u}rltosupportthird-partyanddetailedproductinformation.BasketandBasket_{I}tem:$

Extended to include price$_{a}t_{a}dditionandtotal_{p}ricefordynamicpricingandbasketcalculations.Aban$

Tracks abandoned or saved baskets with timestamps and status (saved, abandoned). Checkout:

Includes tax$_{a}mount, discount_{a}pplied, coupon_{c}ode, andaddressesforshippingandbilling.Transact$

Transaction now integrates tax$_{a}mount, discount_{a}mount, andcoupon_{c}odefields.Transaction_{I}temt$ $specificshippingcosts, andtaxes.Merchant:$

New entity to manage merchant-specific details for multi-vendor setups.

Shipping: Supports full shipping workflows: addresses, methods, costs, tracking numbers, and delivery dates. Discount and Tax:

Discount handles promotions with attributes like type (percentage or fixed) and value. Tax tracks tax jurisdictions and rates, tied to applicable regions.

Position Adjustments:

Optimized node spacing to avoid overlaps. Positioned attributes in a clean radial format around the entities. Reduced Complexity:

Relationships like logs, tracks, and applied to are visually placed in cleaner arrangements.

Arrows: Enhanced clarity by explicitly defining the flow between entities and relationships using arrows. Flexibility:

Attributes like username, sku, price, and category are linked to their respective entities to emphasize clarity.

Multi-vendor transactions.

Item shipping and tax handling.

Coupon and discount management.

Tracking saved and abandoned baskets.