**DATA DICTIONARY DOCUMENT FOR DINE SWIFT APPLICATION**

**Document Version:** 1.0 **Date:** October 4, 2025 **Project Lead:** Kiyimba Fahad

C:\Users\User\Downloads\WhatsApp Image 2025-09-04 at 08.15.18.jpegC:\Users\User\Downloads\WhatsApp Image 2025-09-04 at 08.48.02 (1).jpeg

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **RegNo** | **EMAIL** | **PHONE** |
| Mushabe Moses | 23/U/12131/EVE | Mosesmushae9@gmail.com | 0752307875 |
| Drate Hillary | 23/U/23611 | dratehillary@gmail.om | 0758235980 |
| Mukyala Dorcus Nandy | 23/U/11911/EVE | mukyaladorcus@gmail.com | 0755011795 |
| Kiyimba Fahad | 23/U/0628 | kiyimbafwitty@gmail.com | 0762938957 |

Contents

[1. INTRODUCTION 2](#_Toc210733115)

[2. SUBSYSTEM 1: USER IDENTITY, ROLES, AND STAFF 3](#_Toc210733116)

[3. SUBSYSTEM 2: ORDERING, INVENTORY, AND LOGISTICS 4](#_Toc210733117)

[4. SUBSYSTEM 3: FINANCIAL, ACCOUNTS, AND PAYMENTS 6](#_Toc210733118)

[5. SUBSYSTEM 4: MENU AND ITEM DEFINITION 7](#_Toc210733119)

[6. SUBSYSTEM 5:MEDIA ASSET TRACKING 8](#_Toc210733120)

[7. SUBSYSTEM 6: COMMUNICATION AND NOTIFICATIONS 9](#_Toc210733121)

[8. USER INTERACTION TRACKING 10](#_Toc210733122)

[9. FINAL AUDIT FIELDS 11](#_Toc210733123)

[10. RELATIONSHIPS 12](#_Toc210733124)

[1. Core Hierarchy & Account Relationships 12](#_Toc210733125)

[2. Polymorphic Content and Media Association 13](#_Toc210733126)

[3. Interaction and Communication 14](#_Toc210733127)

[11. CONCLUSION 14](#_Toc210733128)

# INTRODUCTION

This document clearly describes the data base schematic structure, relationships for the entire Dineswift app depending on the proposed RDD

# SUBSYSTEM 1: USER IDENTITY, ROLES, AND STAFF

This system enforces the **Role-Based Access (RBA)** model, allowing a single user (e.g., a Manager) to simultaneously hold multiple roles (e.g., Customer).

| Entity | Field Name | Data Type | Key/Index | Description & Notes |
| --- | --- | --- | --- | --- |
| **users** | **user\_id** | UUID/String | PK | Unique ID for the person. |
|  | **email** | String | Unique Index | User's primary email. |
|  | **full\_name** | String |  | User's name. |
|  | **phone\_number** | String |  | Contact number. |
|  | **gps\_location** | JSONB |  | Last known or current GPS coordinates. |
| **roles** | **role\_id** | Integer | PK | Unique ID for the role. |
|  | **role\_name** | String | Unique Index | e.g., 'customer', 'manager', 'supplier'. |
| **user\_roles** | **user\_role\_id** | UUID/String | PK | Junction table ID. |
|  | **user\_id** | UUID/String | **FK (users)** | The user holding the role. |
|  | **role\_id** | Integer | **FK (roles)** | The role assigned. |
|  | **restaurant\_id** | UUID/String | **FK (restaurants)** | Optional: Links role to a specific restaurant (e.g., 'Worker at Restaurant A'). |
| **restaurant\_staff** | **staff\_id** | UUID/String | PK | Unique staff record. |
|  | **user\_id** | UUID/String | **FK (users)** | Links to the person in the users table. |
|  | **restaurant\_id** | UUID/String | **FK (restaurants)** | The restaurant they work for. |
|  | **role** | Enum |  | e.g., waiter, chef, delivery. |
|  | **managed\_by\_id** | UUID/String | **FK (staff)** | Direct manager's staff\_id. |
|  | **rating\_avg** | Decimal |  | Average rating from customer feedback. |

# SUBSYSTEM 2: ORDERING, INVENTORY, AND LOGISTICS

This handles the core flow from order placement to delivery and stock tracking.

| Entity | Field Name | Data Type | Key/Index | Description & Notes |
| --- | --- | --- | --- | --- |
| **restaurants** | **restaurant\_id** | UUID/String | PK | Unique ID for the restaurant. |
|  | **name** | String |  | Name of the business. |
| **orders** | **order\_id** | UUID/String | PK | Master record for all sales/supply orders. |
|  | **restaurant\_id** | UUID/String | **FK (restaurants)** | The target restaurant. |
|  | **status** | Enum |  | pending, accepted, in\\_delivery, completed. |
|  | **total\_amount** | Decimal |  | Final amount charged. |
| **sales\_order\_details** | **order\_id** | UUID/String | **FK (orders)** | Details for a customer sales order. |
|  | **user\_id** | UUID/String | **FK (users)** | The customer. |
|  | **delivery\_address** | Text |  | Where to deliver. |
|  | **waiter\_staff\_id** | UUID/String | **FK (staff)** | The assigned waiter/server. |
|  | **batch\_id** | UUID/String |  | **De-normalized:** Used to group orders for delivery. |
| **order\_items** | **order\_item\_id** | UUID/String | PK | Specific item ordered. |
|  | **order\_id** | UUID/String | **FK (orders)** | Links to the master order. |
|  | **menu\_or\_inventory\_id** | UUID/String | **Polymorphic** | The ID of the product (Menu Item or Inventory Item). |
|  | **item\_type** | Enum |  | **Crucial:** menu\\_item or inventory\\_item. |
|  | **quantity** | Integer |  | How many units were ordered. |
| **inventory\_items** | **inventory\_item\_id** | UUID/String | PK | Raw ingredient/stock ledger. |
|  | **restaurant\_id** | UUID/String | **FK (restaurants)** | The restaurant owning the stock. |
|  | **name** | String |  | e.g., 'Flour - All Purpose'. |
|  | **current\_stock\_qnty** | Decimal |  | Current stock level. |
|  | **min\_threshold\_qnty** | Decimal |  | Alert trigger level. |
| **bookings** | **booking\_id** | UUID/String | PK | Table reservation or event booking. |
|  | **user\_id** | UUID/String | **FK (users)** | The customer who booked. |
|  | **deposit\_amount** | Decimal |  | **Deposit requirement** (financial transaction links back via transactions.source\_id). |
|  | **table\_id** | UUID/String | **FK (tables)** | The table reserved. |
| **delivery\_log** | **delivery\_log\_id** | UUID/String | PK | Log of delivery events. |
|  | **batch\_id** | UUID/String |  | The group of orders being delivered. |
|  | **worker\_staff\_id** | UUID/String | **FK (staff)** | The driver/delivery person. |
|  | **event\_type** | Enum |  | picked\\_up, route\\_change, delivered. |

# SUBSYSTEM 3: FINANCIAL, ACCOUNTS, AND PAYMENTS

This system uses the **Single Ledger Principle** (transactions table) and handles the Pre-Paid Meal Plan Account.

| Entity | Field Name | Data Type | Key/Index | Description & Notes |
| --- | --- | --- | --- | --- |
| **customer\_accounts** | **account\_id** | UUID/String | PK | ID for the pre-paid meal plan account. |
|  | **user\_id** | UUID/String | **FK (users)** | The account holder. |
|  | **restaurant\_id** | UUID/String | **FK (restaurants)** | **Crucial:** Balance is tied to this restaurant. |
|  | **balance** | Decimal |  | The current non-withdrawable balance. |
|  | **is\_refundable** | Boolean |  | **FIXED:** Set to FALSE by default. |
|  | **crypto\_details** | **JSONB** |  | Flexible storage for network/token info if funded by crypto. |
| **transactions** | **transaction\_id** | UUID/String | PK | Immutable record of all money movement. |
|  | **source\_id** | UUID/String | **Polymorphic** | ID of the originating event (Order, Booking, Account Deposit, etc.). |
|  | **category** | Enum |  | **Crucial:** sales\\_income, account\\_deposit, supplier\\_payment, deposit. |
|  | **amount** | Decimal |  | Monetary value of the transaction. |
|  | **transaction\_type** | Enum |  | debit or credit. |
|  | **payment\_method\_id** | UUID/String | **FK** | How the money was paid/received. |
| **payment\_methods** | **method\_id** | UUID/String | PK | Unique payment method ID. |
|  | **user\_id** | UUID/String | **FK (users)** | The owner of the payment method. |
|  | **method\_type** | Enum |  | card, crypto\\_wallet, cash. |
|  | **last\_four\_digits** | String |  | Masked card/account info. |

# SUBSYSTEM 4: MENU AND ITEM DEFINITION

**Menus** group items; **Menu Items** hold the food details.

| Entity | Field Name | Data Type | Key/Index | Description |
| --- | --- | --- | --- | --- |
| **menus** | **menu\_id** | UUID/String | PK | Unique Menu ID. |
|  | **restaurant\_id** | UUID/String | FK | The restaurant owning the menu. |
|  | **name** | String |  | e.g., 'Lunch Menu', 'Dinner Menu'. |
|  | **description** | Text |  | Brief overview of the menu. |
|  | **price\_range** | Enum |  | low, medium, high (replaces $ signs). |
|  | **allergen\_info** | Text |  | General menu allergen statement. |
|  | **created\_at** | Timestamp |  | Audit field. |
|  | **created\_by\_id** | UUID/String | FK (users) | User ID of the creator. |
| **menu\_items** | **menu\_item\_id** | UUID/String | PK | Unique ID for the sellable product. |
|  | **menu\_id** | UUID/String | FK | Links back to the menus table (Many-to-One). |
|  | **inventory\_item\_id** | UUID/String | FK (Optional) | Links to the primary raw component for stock tracking. |
|  | **price** | Decimal |  | Selling price. |
|  | **category** | String |  | e.g., 'Appetizers', 'Main Courses'. |
|  | **preparation\_time\_min** | Integer |  | Estimated prep time (minutes). |
|  | **portion\_size** | String |  | e.g., '250g', 'Large'. |
|  | **calories** | Integer |  | Caloric content. |
|  | **nutritional\_info** | Text |  | Detailed nutritional description. |
|  | **ingredients** | Text |  | List of ingredients (for display). |
|  | **is\_available** | Boolean |  | Based on stock and time of day. |
|  | **special\_status** | Enum |  | daily\\_special, seasonal, standard. |

# **SUBSYSTEM 5:MEDIA ASSET TRACKING**

To prevent duplicate tables for media, we used a single media\_assets table linked by a polymorphic key.

| Entity | Field Name | Data Type | Key/Index | Description |
| --- | --- | --- | --- | --- |
| **media\_assets** | **media\_id** | UUID/String | PK | Unique media identifier. |
|  | **source\_entity\_id** | UUID/String | Polymorphic | ID of the entity that owns the media. |
|  | **source\_entity\_type** | Enum |  | menu\\_item, announcement, table. |
|  | **media\_type** | Enum |  | image, video. |
|  | **media\_url** | String |  | URL of the media file. |
|  | **thumbnail\_url** | String |  | URL for faster loading. |
|  | **uploaded\_by\_id** | UUID/String | FK (users) | User who uploaded the media. |

# SUBSYSTEM 6: COMMUNICATION AND NOTIFICATIONS

This includes essential group definitions and the notifications that link polymorphically to every event.

| Entity | Field Name | Data Type | Key/Index | Description & Notes |
| --- | --- | --- | --- | --- |
| **communication\_groups** | **group\_id** | UUID/String | PK | Master ID for all groups. |
|  | **restaurant\_id** | UUID/String | **FK (restaurants)** | The owner. |
|  | **type** | Enum |  | **FIXED:** internal\\_staff or public\\_community (for security). |
|  | **name** | String |  | e.g., 'Kitchen Crew' or 'Gold Tier Members'. |
| **group\_members** | **member\_id** | UUID/String | PK | Junction table ID. |
|  | **group\_id** | UUID/String | **FK** | The group being joined. |
|  | **user\_id** | UUID/String | **FK (users)** | The member. |
| **chat\_messages** | **message\_id** | UUID/String | PK | Unique message ID. |
|  | **sender\_id** | UUID/String | **FK (users)** | The user who sent the message. |
|  | **recipient\_id** | UUID/String | **FK (users)** | Optional: For 1:1 chat. |
|  | **group\_id** | UUID/String | **FK (groups)** | Optional: For group chat. |
| **notifications** | **notification\_id** | UUID/String | PK | Unique notification instance. |
|  | **recipient\_id** | UUID/String | **FK (users)** | Who receives the alert. |
|  | **source\_entity\_id** | UUID/String | **Polymorphic** | ID of the trigger event (Order, Message, Booking, Deposit). |
|  | **source\_entity\_type** | Enum |  | order, message, booking, batch. |
|  | **message** | Text |  | The notification text itself. |

# USER INTERACTION TRACKING

We will use a generic source\_id and source\_type for *every* interaction table, simplifying the entire relationship structure.

**A. comments and replies**

| Entity | Field Name | Data Type | Key/Index | Description |
| --- | --- | --- | --- | --- |
| **comments** | **comment\_id** | UUID/String | PK | Unique comment ID. |
|  | **source\_entity\_id** | UUID/String | Polymorphic | ID of the item being commented on (e.g., a media\_id or menu\_item\_id, message). |
|  | **source\_entity\_type** | Enum |  | media, menu\\_item. |
|  | **comment\_text** | Text |  | The content of the comment. |
|  | **user\_id** | UUID/String | FK (users) | The user who posted the comment. |
|  | **updated\_at** | Timestamp |  | Used to enforce the "16-minute edit window" rule in the application logic. |
| **replies** | **reply\_id** | UUID/String | PK | Unique reply ID. |
|  | **comment\_id** | UUID/String | FK | The comment being replied to. |
|  | **reply\_text** | Text |  | The content of the reply. |
|  | **user\_id** | UUID/String | FK (users) | The user who posted the reply. |

**B. likes**

The likes table will track likes on any entity (Media, Comment, Reply, Menu Item).

| Entity | Field Name | Data Type | Key/Index | Description |
| --- | --- | --- | --- | --- |
| **likes** | **like\_id** | UUID/String | PK | Surrogate key for the like action. |
|  | **source\_entity\_id** | UUID/String | Polymorphic | ID of the item being liked. |
|  | **source\_entity\_type** | Enum |  | media, comment, reply, menu\\_item. |
|  | **user\_id** | UUID/String | FK (users) | The user who performed the like action. |
|  | **created\_at** | Timestamp |  | Time of the like action. |

**C. views**

| Entity | Field Name | Data Type | Key/Index | Description |
| --- | --- | --- | --- | --- |
| **views** | **view\_id** | UUID/String | PK | Surrogate key for the view action. |
|  | **source\_entity\_id** | UUID/String | Polymorphic | ID of the item being viewed. |
|  | **source\_entity\_type** | Enum |  | media, menu\\_item, comment, reply. |
|  | **user\_id** | UUID/String | FK (users) | The user who viewed the item. |
|  | **created\_at** | Timestamp |  | Time of the view action. |

# FINAL AUDIT FIELDS

Finally, we ensure every table uses a consistent, normalized set of audit fields, always referencing the **users.user\_id** (UUID) for integrity.

| Field Name | Data Type | Key/Index | Notes |
| --- | --- | --- | --- |
| **created\_at** | Timestamp |  | Date/Time of creation. |
| **updated\_at** | Timestamp |  | Date/Time of last modification. |
| **deleted\_at** | Timestamp |  | Soft delete marker (preferred over hard delete). |
| **created\_by\_id** | UUID/String | FK (users) | The person who created the record. |
| **updated\_by\_id** | UUID/String | FK (users) | The person who last updated the record. |
| **deleted\_by\_id** | UUID/String | FK (users) | The person who deleted the record. |

# RELATIONSHIPS

Finalized Entity Relationships and Integrity Mechanisms for **DINESWIFT**

## Core Hierarchy & Account Relationships

These relationships form the backbone of the system's identity, access, and account structure.

| Relationship | Entities Involved | Mechanism | Description |
| --- | --- | --- | --- |
| **User Role Assignment** | users ↔ roles | **Many-to-Many via Junction Table** | A single **users** record (person) can possess multiple **roles** (customer, manager, worker) through the intermediate **user\_roles** table. This is the core of the RBA model. |
| **Staff Employment** | users ← restaurant\\_staff | **Direct Foreign Key (FK)** | Every record in **restaurant\_staff** points back to one **user** who holds the staff position. |
| **Meal Account Ownership** | users ← customer\\_accounts | **Direct Foreign Key (FK)** | A **user** can have multiple pre-paid **customer\_accounts**, but each account belongs to exactly one user and is tied to one specific restaurant\\_id. |
| **Menu Ownership** | restaurants ← menus | **Direct Foreign Key (FK)** | A **restaurant** can manage multiple **menus**, but each menu belongs to only one restaurant. |
| **Menu to Menu Item** | menus ← menu\\_items | **Direct Foreign Key (FK)** | A **menu** contains many **menu\\_items**, but in this structure, a menu item is directly attached to one specific menu (simplifying management). |

## Polymorphic Content and Media Association

This pattern is used when one piece of content (like a photo) can be owned by or associated with multiple different types of master tables.

| Relationship | Entities Involved | Mechanism | Description |
| --- | --- | --- | --- |
| **Media Ownership** | media\\_assets ↔ menu\\_items, announcements, etc. | **Polymorphic Key** | A **media\\_asset** uses the pair (source\\_entity\\_id + source\\_entity\\_type) to point to *any* single owner (e.g., a specific **menu\\_item** or an **announcement**). |
| **Order Item Association** | order\\_items ↔ menu\\_items OR inventory\\_items | **Polymorphic Key** | An **order\\_item** uses the pair (menu\\_or\\_inventory\\_id + item\\_type) to link to either a sellable product or a raw stock item, ensuring all order types are tracked in one table. |
| **Financial Source** | transactions ↔ orders, bookings, accounts | **Polymorphic Key** | Every entry in the **transactions** ledger links back to the specific event that triggered it using the pair (source\\_id + category). |
| **Notification Source** | notifications ↔ orders, messages, bookings, etc. | **Polymorphic Key** | A **notification** links to the event that generated it (e.g., a new **message** or a completed **order**) via the pair (source\\_entity\\_id + source\\_entity\\_type). |

***Note on Polymorphic Keys:*** *The database* ***will not*** *automatically enforce the integrity of these keys. The* ***Python application layer*** *is responsible for validating that the UUID in the ID field exists in the table specified by the Type field.*

## Interaction and Communication

This section details the complex relationships for user feedback and communication threads.

| Relationship | Entities Involved | Mechanism | Description |
| --- | --- | --- | --- |
| **Interaction Target** | likes, views, comments ↔ ANY content | **Polymorphic Key** | **likes** and **views** use a polymorphic key to link to *any* piece of content: a **menu\\_item**, **media\\_asset**, **comment**, or **reply**. |
| **Comment Threading** | comments ← replies | **Direct Foreign Key (FK)** | A **comment** can receive many **replies**, and each reply points directly back to its parent comment. |
| **Direct Messaging** | users ↔ chat\\_messages | **Direct Foreign Key (FK)** | A **chat\\_message** has a **sender\\_id** and an optional **recipient\\_id**, both pointing directly to the unified **users** table. |
| **Group Communication** | communication\\_groups ← chat\\_messages | **Direct Foreign Key (FK)** | A **chat\\_message** can optionally point to a **group\\_id** to broadcast the message to a defined community or staff group. |
| **Group Membership** | communication\\_groups ↔ users | **Many-to-Many via Junction Table** | The **group\\_members** table links any number of **users** to any **communication\\_group** (enforced by application logic based on the group's type). |

# CONCLUSION

This complete map confirms the entire structure depending on the proposed RDD. The schema is fully finalized and ready for app development and implementation planning.