# Store System Documentation

## Table of Contents



---

## Overview

The Store System is a comprehensive e-commerce solution that enables users to browse stores, search products, manage shopping carts, save favorites, and receive notifications. It includes location-based services, category browsing, promotional banners, and a complete shopping experience.

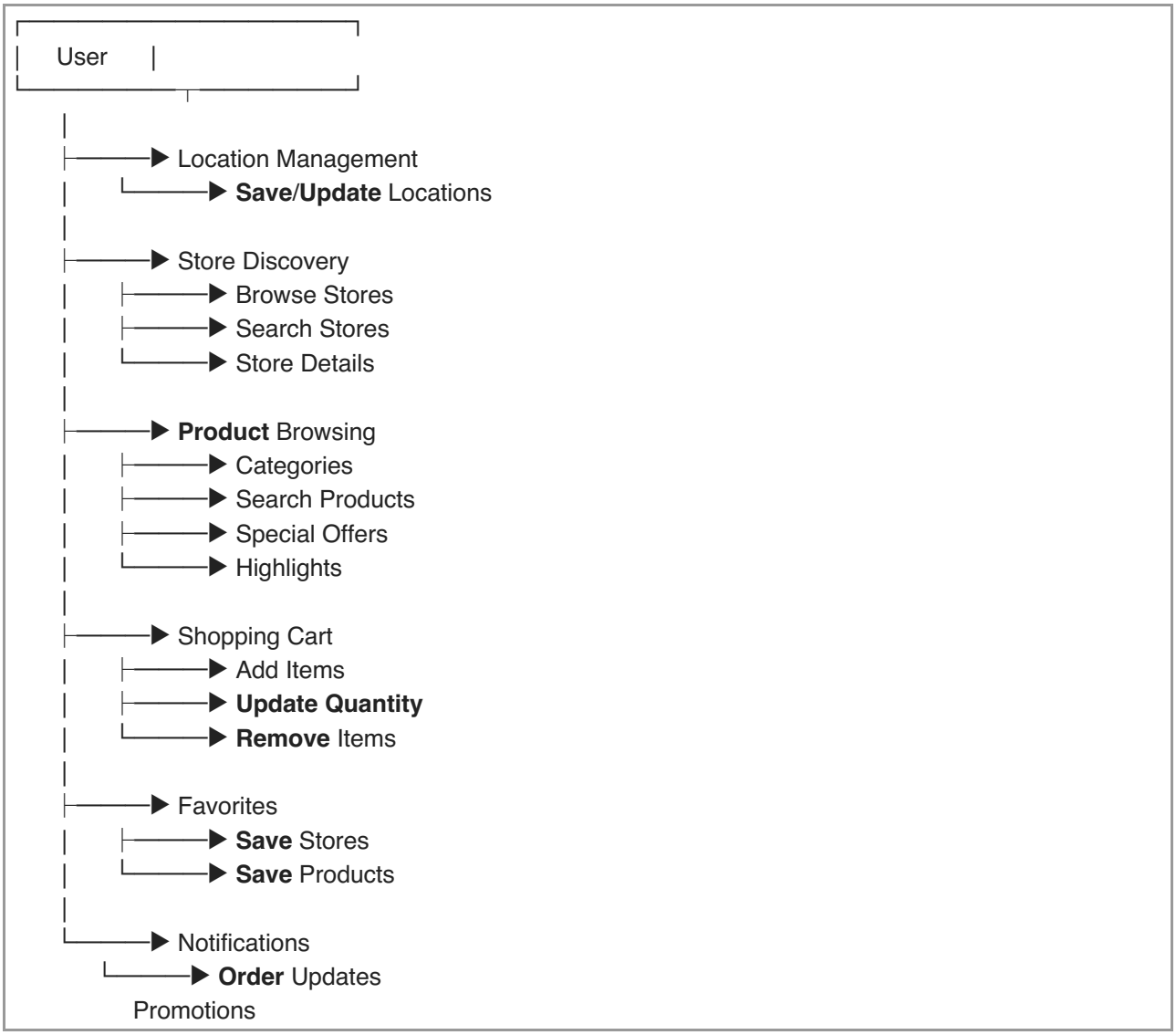
### Key Features

- **Location Management**: Save and manage multiple delivery locations
- **Store Discovery**: Browse and search stores with ratings and reviews
- **Product Catalog**: Search, filter, and browse products by category
- **Shopping Cart**: Add, update, and manage cart items
- **Favorites/Wishlist**: Save favorite stores and products
- **Promotional Banners**: Location-based promotional content
- **Product Reviews**: User reviews and ratings system
- **Notifications**: Real-time notifications for orders and promotions
- **Voice Search**: Voice-based product search capability

---

## System Architecture

### Core Components


User
Location Management
Save/Update Locations
Store Discovery
Browse Stores
Search Stores
Store Details
Product Browsing
Categories
Search Products
Special Offers
Highlights
Shopping Cart
Add Items
Update Quantity
Remove Items
Favorites
Save Stores
Save Products
Notifications
Order Updates
Promotions

## Data Flow


User Request
Authentication Middleware
Validation Middleware
Controller
Database Query
Models (Store, Product, Cart, etc.)
Response
JSON Data

# User Flow & Shopping Lifecycle

## Step 1: Set User Location

**Endpoint**: PUT /api/v1/user/location

**What Happens**:

1. User selects or updates their delivery location
2. System sets the location as default
3. All subsequent requests use this location for filtering

**Request Example**:

```
PUT /api/v1/user/location
Authorization: Bearer <token>

{
  "locationId": "507f1f77bcf86cd799439011"
}
```

**Response Example**:


```
{
  "success": true,
  "message": "Location updated successfully",
  "data": {
    "id": "507f1f77bcf86cd799439011",
    "address": "123 Main St, City, State 12345",
    "latitude": 40.7128,
    "longitude": -74.0060,
    "isDefault": true
  }
}
```


## Step 2: Browse Stores

**Endpoint**: GET /api/v1/store/stores/recommended

**What Happens**:

1. System fetches recommended stores based on location
2. Returns stores sorted by rating and reviews
3. Includes favorite status for each store

**Response Example**:

```
{
  "success": true,
  "data": {
    "stores": [
      {
        "id": "507f1f77bcf86cd799439012",
        "name": "Fresh Groceries",
        "description": "Fresh fruits and vegetables",
        "status": "open",
        "backgroundColor": "#FF6B6B",
        "icon": "https://example.com/icon.png",
        "iconColor": "#FFFFFF",
        "leftImage": "https://example.com/image.png",
        "rating": 4.5,
        "reviews": 120,
        "location": "123 Main St",
        "deliveryTime": "30-45 mins",
        "deliveryFee": 50,
        "minOrder": 200,
        "isFavorite": false
      }
    ]
  }
}
```


### Step 3: Browse Products

**Endpoint**: GET /api/v1/store/products/special-offers

**What Happens**:

1. System fetches products with special offers/discounts
2. Returns products sorted by discount percentage
3. Includes stock availability and favorite status

**Response Example**:

```
{
  "success": true,
  "data": {
    "products": [
      {
        "id": "507f1f77bcf86cd799439013",
        "name": "Organic Apples",
        "image": "https://example.com/apple.jpg",
        "discountPercent": 20,
        "originalPrice": 200,
        "discountedPrice": 160,
        "rating": 4.8,
        "reviews": 45,
        "quantityType": "Kg",
        "stock": 50,
        "storeId": "507f1f77bcf86cd799439012",
        "storeName": "Fresh Groceries",
        "isFavorite": true
      }
    ],
    "total": 25,
    "page": 1,
    "limit": 20
  }
}
```


### Step 4: Add to Cart

**Endpoint**: POST /api/v1/cart/items

**What Happens**:

1. User adds product to cart with quantity
2. System validates product availability and stock
3. Creates or updates cart item
4. Returns cart item details

**Request Example**:

```
POST /api/v1/cart/items
Authorization: Bearer <token>

{
  "productId": "507f1f77bcf86cd799439013",
  "quantity": 2,
  "storeId": "507f1f77bcf86cd799439012"
}
```

**Response Example**:

```
{
  "success": true,
  "message": "Added to cart successfully",
  "data": {
    "id": "507f1f77bcf86cd799439014",
    "productId": "507f1f77bcf86cd799439013",
    "quantity": 2
  }
}
```


### Step 5: View Cart

**Endpoint**: GET /api/v1/cart

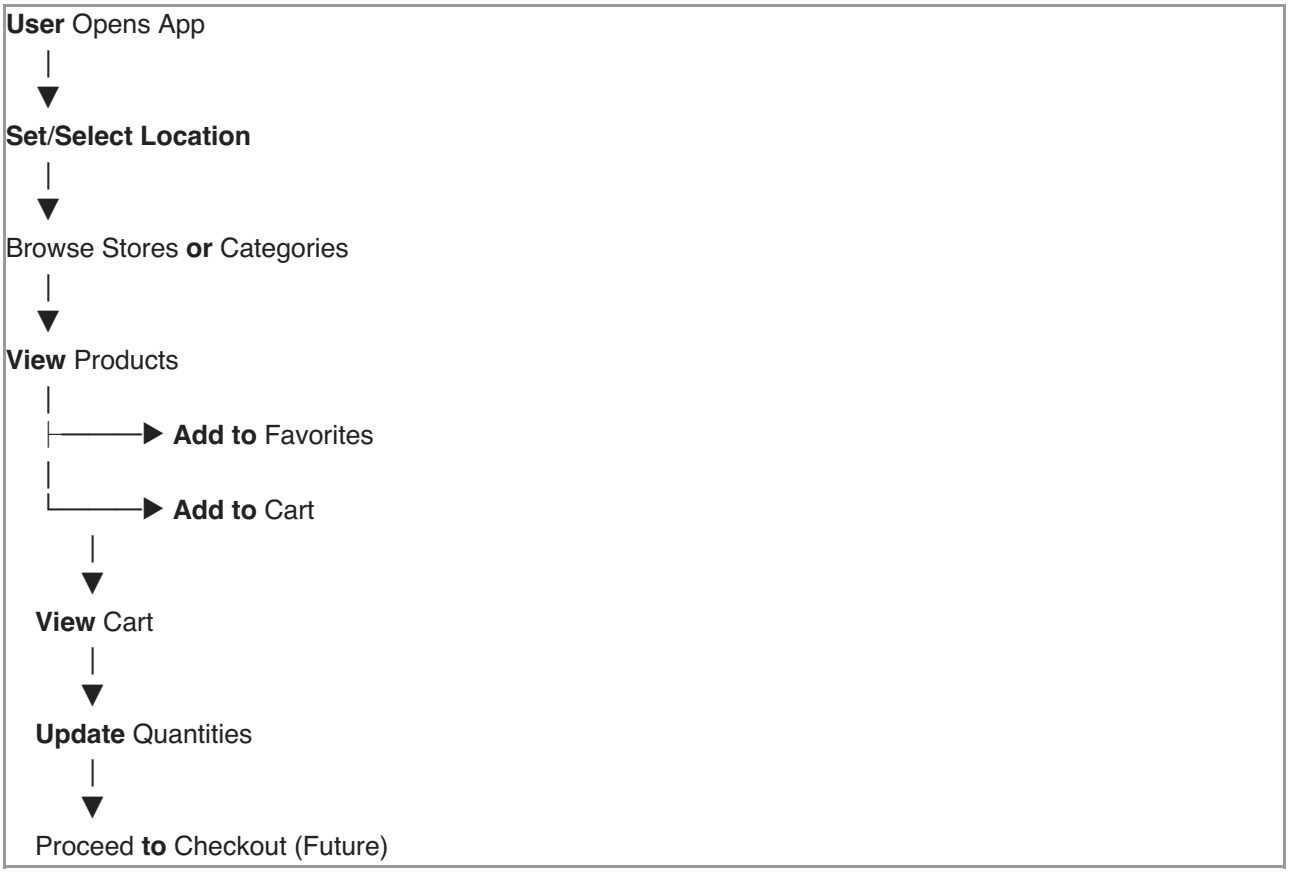
**What Happens**:

1. System fetches all cart items for user
2. Calculates subtotal, delivery fee, and total
3. Returns complete cart summary

**Response Example**:


```
{
  "success": true,
  "data": {
    "items": [
      {
        "id": "507f1f77bcf86cd799439014",
        "productId": "507f1f77bcf86cd799439013",
        "productName": "Organic Apples",
        "productImage": "https://example.com/apple.jpg",
        "price": 160,
        "quantity": 2,
        "storeId": "507f1f77bcf86cd799439012",
        "storeName": "Fresh Groceries"
      }
    ],
    "subtotal": 320,
    "deliveryFee": 50,
    "total": 370
  }
}
```


### Complete Shopping Flow




---

# Data Models

## Location Model

**Collection**: locations

**Schema**:

```
{
  userId: ObjectId (ref: 'User'),
  address: String (required),
  latitude: Number (required),
  longitude: Number (required),
  isDefault: Boolean (default: false),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { userId: 1, isDefault: 1 } - Find default location per user
- { userId: 1 } - Find all locations for user

**Key Features**:

- Only one default location per user (enforced by pre-save hook)
- Location-based filtering for stores and products

## Store Model

**Collection**: stores

**Schema**:

```
{
  name: String (required, indexed),
  description: String (required),
  status: String (enum: ['open', 'closed'], default: 'open'),
  backgroundColor: String (default: '#FFFFFF'),
  icon: String (required),
  iconColor: String (default: '#000000'),
  leftImage: String (required),
  rating: Number (default: 0, min: 0, max: 5),
  reviews: Number (default: 0),
  location: {
    address: String (required),
    latitude: Number (required),
    longitude: Number (required)
  },
  deliveryTime: String (required),
  deliveryFee: Number (default: 0),
  minOrder: Number (default: 0),
  images: [String],
  categories: [String],
  openingHours: {
    monday: { open: String, close: String },
    tuesday: { open: String, close: String },
    // ... other days
  },
  isActive: Boolean (default: true),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { isActive: 1, status: 1 } - Find active open stores
- { name: 'text', description: 'text' } - Text search
- { 'location.latitude': 1, 'location.longitude': 1 } - Geospatial queries

## StoreProduct Model

**Collection**: storeproducts

**Schema**:

```
{
  name: String (required, indexed),
  description: String (required),
  price: Number (required, min: 0),
  originalPrice: Number (required, min: 0),
  discountPercent: Number (default: 0, min: 0, max: 100),
  image: String (required),
  images: [String],
  category: ObjectId (ref: 'StoreCategory', required, indexed),
  rating: Number (default: 0, min: 0, max: 5),
  reviews: Number (default: 0),
  stock: Number (default: 0, min: 0),
  quantityType: String (required, default: 'Pcs'),
  storeId: ObjectId (ref: 'Store', required, indexed),
  specifications: Mixed,
  isActive: Boolean (default: true),
  isSpecialOffer: Boolean (default: false),
  isHighlight: Boolean (default: false),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { storeId: 1, isActive: 1 } - Find products by store
- { category: 1, isActive: 1 } - Find products by category
- { isSpecialOffer: 1, isActive: 1 } - Find special offers
- { isHighlight: 1, isActive: 1 } - Find highlighted products
- { name: 'text', description: 'text' } - Text search

## StoreCategory Model

**Collection**: storecategories

**Schema**:

```
{
  name: String (required, unique),
  icon: String (required),
  image: String (required),
  backgroundColor: String (default: '#FFFFFF'),
  productCount: Number (default: 0),
  order: Number (default: 0),
  isActive: Boolean (default: true),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { isActive: 1, order: 1 } - Find active categories sorted
- { name: 'text' } - Text search

## Banner Model

**Collection**: banners

**Schema**:

```
{
  title: String (required),
  description: String,
  image: String (required),
  link: String (required),
  linkType: String (enum: ['product', 'store', 'category', 'url'], required),
  order: Number (default: 0),
  isActive: Boolean (default: true),
  locationId: ObjectId (ref: 'Location', optional),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { isActive: 1, order: 1 } - Find active banners sorted
- { locationId: 1, isActive: 1 } - Location-specific banners

## StoreCart Model

**Collection**: storecarts

**Schema**:

```
{
  userId: ObjectId (ref: 'User', required),
  productId: ObjectId (ref: 'StoreProduct', required),
  quantity: Number (required, min: 1),
  storeId: ObjectId (ref: 'Store', required),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { userId: 1, productId: 1 } (unique) - One cart item per product per user
- { userId: 1 } - Find all cart items for user
- { storeId: 1 } - Find cart items by store

**Key Features**:

- Prevents duplicate cart items (unique index)
- Automatically updates quantity if item already exists

## Favorite Model

**Collection**: favorites

**Schema**:

```
{
  userId: ObjectId (ref: 'User', required),
  type: String (enum: ['store', 'product'], required),
  itemId: ObjectId (required, refPath: 'typeModel'),
  typeModel: String (enum: ['Store', 'StoreProduct'], required),
  addedAt: Date (default: Date.now),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { userId: 1, type: 1 } - Find favorites by type
- { userId: 1, itemId: 1, type: 1 } (unique) - One favorite per item per user

**Key Features**:

- Supports both stores and products
- Uses dynamic reference (refPath) for polymorphic relationships
- Prevents duplicate favorites

## Notification Model

**Collection**: notifications

**Schema**:

```
{
  userId: ObjectId (ref: 'User', required),
  title: String (required),
  message: String (required),
  type: String (enum: ['order', 'promotion', 'store', 'product', 'general'], default: 'general'),
  isRead: Boolean (default: false),
  link: String (optional),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { userId: 1, isRead: 1, createdAt: -1 } - Find unread notifications
- { userId: 1, createdAt: -1 } - Find all notifications sorted

## StoreReview Model

**Collection**: storereviews

**Schema**:

```
{
  userId: ObjectId (ref: 'User', required),
  productId: ObjectId (ref: 'StoreProduct', required),
  rating: Number (required, min: 1, max: 5),
  comment: String (required),
  createdAt: Date,
  updatedAt: Date
}
```

**Indexes**:

- { productId: 1, createdAt: -1 } - Find reviews by product
- { userId: 1, productId: 1 } (unique) - One review per user per product

---

# API Endpoints

## Location APIs

### 1. Get User Location

**Endpoint**: GET /api/v1/user/location

**Description**: Get current user's default location

**Authentication**: Required

**Response**:

```
{
  "success": true,
  "data": {
    "id": "string",
    "address": "string",
    "latitude": "number",
    "longitude": "number",
    "isDefault": "boolean"
  }
}
```

### 2. Get User Locations List

**Endpoint**: GET /api/v1/user/locations

**Description**: Get all saved locations for user

**Authentication**: Required

**Response**:

```
{
  "success": true,
  "data": {
    "locations": [
      {
        "id": "string",
        "address": "string",
        "latitude": "number",
        "longitude": "number",
        "isDefault": "boolean"
      }
    ]
  }
}
```


**3. Update User Location**

**Endpoint**: PUT /api/v1/user/location

**Description**: Update user's current default location

**Authentication**: Required

**Request Body**:


```
{
  "locationId": "string"
}
```


**Response**:


```
{
  "success": true,
  "message": "Location updated successfully",
  "data": {
    "id": "string",
    "address": "string",
    "latitude": "number",
    "longitude": "number",
    "isDefault": "boolean"
  }
}
```


**4. Search Locations**

**Endpoint**: GET /api/v1/locations/search?query={query}

**Description**: Search for locations/addresses

**Authentication**: Required

**Query Parameters**:

- query (required): Search term

**Response**:


```
{
  "success": true,
  "data": {
    "locations": [
      {
        "id": "string",
        "address": "string",
        "latitude": "number",
        "longitude": "number"
      }
    ]
  }
}
```


## Search APIs

#### 5. Search Items/Products

**Endpoint**: GET /api/v1/store/search?query={query}&category={category}&page={page}&limit={limit}

**Description**: Search for products/items

**Authentication**: Required

**Query Parameters**:

- query (optional): Search term
- category (optional): Category filter
- page (optional): Page number (default: 1)
- limit (optional): Items per page (default: 20)

**Response**:

```
{
  "success": true,
  "data": {
    "items": [
      {
        "id": "string",
        "name": "string",
        "description": "string",
        "price": "number",
        "originalPrice": "number",
        "discountPercent": "number",
        "image": "string",
        "category": "string",
        "rating": "number",
        "reviews": "number",
        "stock": "number",
        "storeId": "string",
        "storeName": "string",
        "isFavorite": "boolean"
      }
    ],
    "total": "number",
    "page": "number",
    "limit": "number"
  }
}
```


**6. Voice Search**

**Endpoint**: POST /api/v1/store/voice-search

**Description**: Process voice search query

**Authentication**: Required

**Request Body**:


```
{
  "audioFile": "base64_encoded_audio"
}
```


**Response**:


```
{
  "success": true,
  "data": {
    "query": "string",
    "items": []
  }
}
```


**Note**: Currently returns mock data. In production, integrate with speech-to-text service.

## Banner APIs

### 7. Get Banners

**Endpoint**: GET /api/v1/store/banners?locationId={locationId}

**Description**: Get promotional banners for carousel

**Authentication**: Required

**Query Parameters**:

- locationId (optional): Location ID for location-specific banners

**Response**:


```
{
  "success": true,
  "data": {
    "banners": [
      {
        "id": "string",
        "title": "string",
        "description": "string",
        "image": "string",
        "link": "string",
        "linkType": "string",
        "order": "number",
        "isActive": "boolean"
      }
    ]
  }
}
```


## Categories APIs

### 8. Get Categories

**Endpoint**: GET /api/v1/store/categories

**Description**: Get all product categories

**Authentication**: Required

**Response**:

```
{
  "success": true,
  "data": {
    "categories": [
      {
        "id": "string",
        "name": "string",
        "icon": "string",
        "image": "string",
        "backgroundColor": "string",
        "productCount": "number",
        "order": "number"
      }
    ]
  }
}
```


#### 9. Get Category Products

**Endpoint**: GET /api/v1/store/categories/{categoryId}/products?page={page}&limit={limit}&sort={sort}

**Description**: Get products by category

**Authentication**: Required

**Path Parameters**:

- categoryId (required): Category ID

**Query Parameters**:

- page (optional): Page number (default: 1)
- limit (optional): Items per page (default: 20)
- sort (optional): Sort option - price_asc, price_desc, rating, newest (default: newest)

**Response**: Same structure as Search Items response

## Stores APIs

#### 10. Get Recommended Stores

**Endpoint**: GET /api/v1/store/stores/recommended?locationId={locationId}&limit={limit}

**Description**: Get recommended stores for user

**Authentication**: Required

**Query Parameters**:

- locationId (optional): Location ID
- limit (optional): Number of stores (default: 10)

**Response**:

```
{
  "success": true,
  "data": {
    "stores": [
      {
        "id": "string",
        "name": "string",
        "description": "string",
        "status": "string",
        "backgroundColor": "string",
        "icon": "string",
        "iconColor": "string",
        "leftImage": "string",
        "rating": "number",
        "reviews": "number",
        "location": "string",
        "deliveryTime": "string",
        "deliveryFee": "number",
        "minOrder": "number",
        "isFavorite": "boolean"
      }
    ]
  }
}
```


**11. Get All Recommended Stores**

**Endpoint**: GET /api/v1/store/stores/recommended/all?locationId={locationId}&page={page}&limit={limit}

**Description**: Get paginated list of all recommended stores

**Authentication**: Required

**Query Parameters**:

- locationId (optional): Location ID
- page (optional): Page number (default: 1)
- limit (optional): Items per page (default: 20)

**Response**: Same structure as above with pagination metadata

**12. Get Store Details**

**Endpoint**: GET /api/v1/store/stores/{storeId}

**Description**: Get detailed information about a store

**Authentication**: Required

**Path Parameters**:

- storeId (required): Store ID

**Response**:

```
{
  "success": true,
  "data": {
    "id": "string",
    "name": "string",
    "description": "string",
    "status": "string",
    "rating": "number",
    "reviews": "number",
    "location": {
      "address": "string",
      "latitude": "number",
      "longitude": "number"
    },
    "deliveryTime": "string",
    "deliveryFee": "number",
    "minOrder": "number",
    "images": ["string"],
    "categories": ["string"],
    "isFavorite": "boolean",
    "openingHours": {
      "monday": {"open": "string", "close": "string"},
      "tuesday": {"open": "string", "close": "string"}
    }
  }
}
```


#### 13. Search Stores

**Endpoint**: GET /api/v1/store/stores/search?query={query}&locationId={locationId}

**Description**: Search for stores

**Authentication**: Required

**Query Parameters**:

- query (optional): Search term
- locationId (optional): Location ID

**Response**: Same structure as Get Recommended Stores

### Products APIs

#### 14. Get Special Offers

**Endpoint**: GET /api/v1/store/products/special-offers?locationId={locationId}&page={page}&limit={limit}

**Description**: Get products with special offers/discounts

**Authentication**: Required

**Query Parameters**:

- locationId (optional): Location ID
- page (optional): Page number (default: 1)
- limit (optional): Items per page (default: 20)

**Response**:


```
{
  "success": true,
  "data": {
    "products": [
      {
        "id": "string",
        "name": "string",
        "image": "string",
        "discountPercent": "number",
        "originalPrice": "number",
        "discountedPrice": "number",
        "rating": "number",
        "reviews": "number",
        "quantityType": "string",
        "stock": "number",
        "storeId": "string",
        "storeName": "string",
        "isFavorite": "boolean"
      }
    ],
    "total": "number",
    "page": "number",
    "limit": "number"
  }
}
```


**15. Get Highlights/Popular Products**

**Endpoint**: GET /api/v1/store/products/highlights?locationId={locationId}&page={page}&limit={limit}

**Description**: Get most popular/highlighted products

**Authentication**: Required

**Query Parameters**: Same as Special Offers

**Response**: Same structure as Special Offers

**16. Get Product Details**

**Endpoint**: GET /api/v1/store/products/{productId}

**Description**: Get detailed information about a product

**Authentication**: Required

**Path Parameters**:

- productId (required): Product ID

**Response**:


```
{
  "success": true,
  "data": {
    "id": "string",
    "name": "string",
    "description": "string",
    "images": ["string"],
    "price": "number",
    "originalPrice": "number",
    "discountPercent": "number",
    "rating": "number",
    "reviews": [
      {
        "id": "string",
        "userId": "string",
        "userName": "string",
        "rating": "number",
        "comment": "string",
        "date": "string"
      }
    ],
    "quantityType": "string",
    "stock": "number",
    "category": "string",
    "storeId": "string",
    "storeName": "string",
    "isFavorite": "boolean",
    "specifications": {}
  }
}
```


**17. Get Products by Store**

**Endpoint**: GET /api/v1/store/stores/{storeId}/products?page={page}&limit={limit}&category={category}

**Description**: Get all products from a specific store

**Authentication**: Required

**Path Parameters**:

- storeId (required): Store ID

**Query Parameters**:

- page (optional): Page number (default: 1)
- limit (optional): Items per page (default: 20)
- category (optional): Category filter

**Response**: Same structure as Search Items response

## Favorites/Wishlist APIs

### 18. Get User Favorites

**Endpoint**: GET /api/v1/user/favorites?type={type}

**Description**: Get user's favorite stores or products

**Authentication**: Required

**Query Parameters**:

- type (optional): stores or products

**Response**:


```
{
  "success": true,
  "data": {
    "favorites": [
      {
        "id": "string",
        "type": "string",
        "itemId": "string",
        "addedAt": "string"
      }
    ]
  }
}
```


### 19. Add to Favorites

**Endpoint**: POST /api/v1/user/favorites

**Description**: Add store or product to favorites

**Authentication**: Required

**Request Body**:


```
{
  "type": "store",
  "itemId": "string"
}
```


**Response**:

```
{
 "success": true,
 "message": "Added to favorites successfully",
 "data": {
  "id": "string",
  "type": "string",
  "itemId": "string",
  "addedAt": "string"
 }
}
```


#### 20. Remove from Favorites

**Endpoint**: DELETE /api/v1/user/favorites/{favoriteId}

**Description**: Remove item from favorites

**Authentication**: Required

**Path Parameters**:

- favoriteId (required): Favorite ID

**Response**:


```
{
 "success": true,
 "message": "Removed from favorites successfully"
}
```


#### 21. Check Favorite Status

**Endpoint**: GET /api/v1/user/favorites/check?type={type}&itemId={itemId}

**Description**: Check if item is favorited

**Authentication**: Required

**Query Parameters**:

- type (required): stores or products
- itemId (required): Item ID

**Response**:


```
{
 "success": true,
 "data": {
  "isFavorite": "boolean"
 }
}
```


### Cart APIs

#### 22. Get Cart

**Endpoint**: GET /api/v1/cart

**Description**: Get user's shopping cart

**Authentication**: Required

**Response**:


```
{
  "success": true,
  "data": {
    "items": [
      {
        "id": "string",
        "productId": "string",
        "productName": "string",
        "productImage": "string",
        "price": "number",
        "quantity": "number",
        "storeId": "string",
        "storeName": "string"
      }
    ],
    "subtotal": "number",
    "deliveryFee": "number",
    "total": "number"
  }
}
```


**23. Add to Cart**

**Endpoint**: POST /api/v1/cart/items

**Description**: Add product to cart

**Authentication**: Required

**Request Body**:


```
{
  "productId": "string",
  "quantity": "number",
  "storeId": "string"
}
```


**Response**:

```
{
  "success": true,
  "message": "Added to cart successfully",
  "data": {
    "id": "string",
    "productId": "string",
    "quantity": "number"
  }
}
```


**Error Cases**:

- Product not found (404)
- Insufficient stock (400)
- Product not active (400)

**24. Update Cart Item**

**Endpoint**: PUT /api/v1/cart/items/{itemId}

**Description**: Update cart item quantity

**Authentication**: Required

**Path Parameters**:

- itemId (required): Cart item ID

**Request Body**:


```
{
  "quantity": "number"
}
```


**Response**:


```
{
  "success": true,
  "message": "Cart item updated successfully",
  "data": {
    "id": "string",
    "quantity": "number"
  }
}
```


**25. Remove from Cart**

**Endpoint**: DELETE /api/v1/cart/items/{itemId}

**Description**: Remove item from cart

**Authentication**: Required

**Path Parameters**:

- itemId (required): Cart item ID

**Response**:

```
{
  "success": true,
  "message": "Removed from cart successfully"
}
```

#### 26. Clear Cart

**Endpoint**: DELETE /api/v1/cart

**Description**: Clear entire cart

**Authentication**: Required

**Response**:

```
{
  "success": true,
  "message": "Cart cleared successfully"
}
```

## Notifications APIs

#### 27. Get Notifications

**Endpoint**: GET /api/v1/notifications?page={page}&limit={limit}

**Description**: Get user notifications

**Authentication**: Required

**Query Parameters**:

- page (optional): Page number (default: 1)
- limit (optional): Items per page (default: 20)

**Response**:

```
{
  "success": true,
  "data": {
    "notifications": [
      {
        "id": "string",
        "title": "string",
        "message": "string",
        "type": "string",
        "isRead": "boolean",
        "createdAt": "string",
        "link": "string"
      }
    ],
    "unreadCount": "number",
    "total": "number",
    "page": "number",
    "limit": "number"
  }
}
```


#### 28. Mark Notification as Read

**Endpoint**: PUT /api/v1/notifications/{notificationId}/read

**Description**: Mark notification as read

**Authentication**: Required

**Path Parameters**:

- notificationId (required): Notification ID

**Response**:


```
{
  "success": true,
  "message": "Notification marked as read",
  "data": {
    "id": "string",
    "isRead": "boolean"
  }
}
```


## Additional Utility APIs

#### 29. Get Delivery Time Estimate

**Endpoint**: GET /api/v1/store/delivery-time?storeId={storeId}&locationId={locationId}

**Description**: Get estimated delivery time

**Authentication**: Required

**Query Parameters**:

- storeId (required): Store ID
- locationId (optional): Location ID

**Response**:

```
{
  "success": true,
  "data": {
    "estimatedTime": "string",
    "deliveryFee": "number"
  }
}
```

**30. Get Store Status**

**Endpoint**: GET /api/v1/store/stores/{storeId}/status

**Description**: Check if store is open/closed

**Authentication**: Required

**Path Parameters**:

- storeId (required): Store ID

**Response**:

```
{
  "success": true,
  "data": {
    "status": "string",
    "nextOpenTime": "string"
  }
}
```

**Status Calculation**:

- Checks current time against store's opening hours
- Returns open or closed based on current day and time
- Returns next opening time if currently closed

---

# Workflow Examples

## Example 1: Complete Shopping Flow

**Scenario**: User wants to buy groceries

1. **Set Location**

```
PUT /api/v1/user/location
{ "locationId": "loc123" }
```

2. **Browse Stores**

```
GET /api/v1/store/stores/recommended?locationId=loc123
```

3. **View Store Products**

```
GET /api/v1/store/stores/store123/products
```

4. **Add to Cart**

```
POST /api/v1/cart/items
{
  "productId": "prod123",
  "quantity": 2,
  "storeId": "store123"
}
```

5. **View Cart**

```
GET /api/v1/cart
```

6. **Add Favorite Store**

```
POST /api/v1/user/favorites
{
  "type": "store",
  "itemId": "store123"
}
```

## Example 2: Search and Filter Products

**Scenario**: User searches for "organic apples"

1. **Search Products**

```
GET /api/v1/store/search?query=organic%20apples
```

2. **Filter by Category**

```
GET /api/v1/store/search?query=organic%20apples&category=Fruits
```

3. **View Product Details**

```
GET /api/v1/store/products/prod123
```

4. **Add to Favorites**

```
POST /api/v1/user/favorites
{
  "type": "product",
  "itemId": "prod123"
}
```

## Example 3: Browse Special Offers

**Scenario**: User wants to see discounted products

1. **Get Special Offers**

```
GET /api/v1/store/products/special-offers?page=1&limit=20
```

2. **Add Multiple Items to Cart**

```
POST /api/v1/cart/items
{ "productId": "prod1", "quantity": 1, "storeId": "store1" }

POST /api/v1/cart/items
{ "productId": "prod2", "quantity": 2, "storeId": "store1" }
```

3. **Update Cart Item**

```
PUT /api/v1/cart/items/cartItem123
{ "quantity": 3 }
```

---

# Configuration

## Environment Variables

No additional environment variables required. Uses existing database and authentication configuration.

## Database Indexes

All models include optimized indexes for:

- User-based queries
- Text search
- Geospatial queries (stores)
- Compound queries (cart, favorites)

## Rate Limiting

Store endpoints follow standard rate limiting:

- **Public Endpoints**: 100 requests per 15 minutes per IP
- **Authenticated Endpoints**: 1000 requests per 15 minutes per user

## Error Handling

All endpoints return consistent error format:

```
{
  "success": false,
  "message": "Error message",
  "errors": [
    {
      "field": "productId",
      "message": "Product ID is required"
    }
  ],
  "code": "VALIDATION_ERROR"
}
```

### Common Error Codes

- NOT_FOUND - Resource not found
- VALIDATION_ERROR - Input validation failed
- DUPLICATE_ENTRY - Item already exists (e.g., already in favorites)
- INSUFFICIENT_STOCK - Product stock insufficient
- INVALID_STATE - Product/store not active
- UNAUTHORIZED - Authentication required

### Pagination

All list endpoints support pagination:

- page: Page number (default: 1)
- limit: Items per page (default: 20)

Response includes:

- total: Total number of items
- page: Current page
- limit: Items per page

### Sorting

Product endpoints support sorting:

- price_asc: Price low to high
- price_desc: Price high to low
- rating: Highest rating first
- newest: Most recently added first

---

## Summary

**Total API Endpoints**: 30+

### By Category:

- **Location**: 4 endpoints
- **Search**: 2 endpoints
- **Banners**: 1 endpoint

- **Categories**: 2 endpoints
- **Stores**: 4 endpoints
- **Products**: 4 endpoints
- **Favorites**: 4 endpoints
- **Cart**: 5 endpoints
- **Notifications**: 2 endpoints
- **Utilities**: 2 endpoints

## Authentication:

Most endpoints require JWT authentication token:

```
Authorization: Bearer <token>
```

## Base URL:

```
http://localhost:3000/api/v1
```

---

**Document Version**: 1.0
**Last Updated**: December 2024