Technical Documentation for Arab-Connect

Overview

This documentation provides a detailed technical overview of the API system developed using Laravel. The system supports user authentication, branch management, contact forms, country data, home page functionalities, and subcategory management. It is designed to facilitate interactions for both users and vendors, with robust validation, authentication, and data management mechanisms.

Table of Contents

System Architecture

Technology Stack

- **Backend**: Laravel 9.x
- **Database**: MySQL

- **Authentication**: JSON Web Tokens (JWT) via `tymon/jwt-auth`
- **Media Handling**: Spatie Media Library
- **Logging**: Spatie Activity Log
- **Validation**: Laravel Validator
- **Notifications**: Laravel Notification System
- **External Services**: WhatsApp integration (via custom service layer)

Directory Structure

```
app/
├── Http/
│   ├── Controllers/
│   │   ├── Api/          # API controllers (AuthController,
BranchController, etc.)
│   │   └── ...
│   ├── ServicesLayer/    # Business logic services (BranchService,
WhatsAppService)
│   └── ...
├── Models/               # Eloquent models (User, Branch, Category,
etc.)
└── ...
```

Authentication

JWT Authentication Flow

- **Registration**: Users or vendors register via `/auth/register`.
- **Login**: Authenticate via `/auth/login` to receive a JWT token.
- **Protected Routes**: Secured with `auth:api` middleware.
- **Token Refresh**: Refresh tokens via `/auth/refresh`.
- **Logout**: Invalidate tokens via `/auth/logout`.

Key Authentication Endpoints

Endpoint	Method	Description	Parameters

/auth/register	POST	Register new user or vendor	email, mobile, country_code, password, name, user_type, country_id
/auth/login	POST	User login	email, password
/auth/email-check	POST	Verify email with code	email, code
/auth/regenerate-code	POST	Regenerate email verification code	email
/auth/send-reset-code	POST	Send password reset code	email
/auth/verify-reset-code	POST	Verify password reset code	email, code
/auth/reset	POST	Reset password	email, code, password, password_confirmation
/auth/change-password	POST	Change password	old_password, new_password, new_password_confirmation
/auth/refresh	GET	Refresh JWT token	Requires auth:api middleware
/auth/logout	POST	Invalidate token	Requires auth:api middleware

User Management

User Model Fields

```
[
    'name' => 'string|max:255',
    'email' => 'string|email|max:255|unique',
    'mobile' => 'string|max:60|unique',
    'country_code' => 'string|max:10',
    'password' => 'string|hashed',
    'user_type' => 'integer|in:1,2', // 1=vendor, 2=user
    'code' => 'string|max:4', // Verification code
    'img' => 'string|null', // Profile photo path
    'country_id' => 'integer|exists:countries,id',
    'email_verified_at' => 'datetime|null',
    'mobile_verified_at' => 'datetime|null',
    'new_email' => 'string|null', // For email change verification
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null', // Soft delete
]
```

Key User Endpoints

Endpoint	Method	Description	Parameters
/auth	GET	Get authenticated user details	Requires auth:api middleware
/auth/user/update-profile	POST	Update user profile	name, mobile, country_code, email, img, country_id
/auth/user/upload-photo	POST	Upload profile photo	img (image file, max 5MB)
/auth/change-mobile-number	POST	Change mobile number	mobile
/auth/update	POST	Update user name	name
/auth/update-email	POST	Initiate email change	email
/auth/verify-email	POST	Verify new email	email, code
/auth/user/verify-new-email	POST	Verify new email after change	code
/auth/delete-account	GET	Soft delete account	Requires auth:api middleware

Branch Management

Branch Model Fields

```
[
    'name' => 'string|max:255',
    'email' => 'string|max:255|null',
    'mobile' => 'string|max:255',
    'country_code' => 'string|max:10|null',
    'location' => 'string|max:1550',
    'map_location' => 'string|null', // Generated Google Maps link
    'imgs' => 'string|null', // Comma-separated image paths
    'tax_card' => 'string|null', // Image path
    'commercial_register' => 'string|null', // Image path
    'face' => 'string|max:1550|null', // Facebook URL
    'insta' => 'string|max:1550|null', // Instagram URL
    'tiktok' => 'string|max:1550|null', // TikTok URL
    'website' => 'string|max:1550|null', // Website URL
    'category_id' => 'integer|exists:categories,id|null',
    'uuid' => 'string|unique', // Custom UUID
    'owner_id' => 'integer|exists:users,id',
    'lon' => 'string|max:100|null',
    'lat' => 'string|max:100|null',
    'is_verified' => 'boolean',
    'expire_at' => 'datetime|null', // Subscription expiration
    'all_days' => 'boolean', // Indicates if branch operates all days
    'three_month_email_sent_at' => 'datetime|null',
    'one_month_email_sent_at' => 'datetime|null',
]
```

Key Branch Endpoints

Endpoint	Method	Description	Parameters
/auth/branches/store	POST	Create new branch (vendor only)	name, mobile, country_code, country_id, location, lat, lon, img, category_id, payments, email, face, insta, tiktok, website, tax_card, commercial_register, days, all_days
/branches/store	POST	Create new branch (vendor only)	Same as above
/branches/update/{id}	POST	Update branch	Same as above, id required
/branch/{id}	GET	Get branch details	id (optional)
/branches/details/{id}	GET	Get enriched branch details	id (optional)

/get/branches/{page}	POST	Search/filter branches	search, id, category_id, lat, lon, page
/nearest-branches	POST	Get branches by proximity	lat, lon

Branch Relationships

- **Owner**: Belongs to a User (vendor).
- **Category**: Belongs to a Category.
- **SubCategory**: Belongs to a SubCategory.
- **Payments**: Many-to-many with PaymentMethod via branch_payments.
- **Days**: Has many Day records for working hours.
- **Related Branches**: Has many branches owned by the same user.

Categories & Subcategories

Category Model

```
[
    'name' => 'string|max:255',
    'img' => 'string|null', // Image path
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null', // Soft delete
]
```

- **Relationships**:
  - Has many SubCategory.
  - Has many Branch.

SubCategory Model

```
[
    'category_id' => 'integer|exists:categories,id',
    'name' => 'string|max:255',
    'img' => 'string|null',
]
```

- **Relationships**:
  - Belongs to Category.
  - Has many Branch.

Key Endpoints

Endpoint	Method	Description	Parameters
/categories	GET	List all categories	per_page, search
/sub-categories/{category_id}	GET	Get subcategories for a category	category_id, per_page, page
/sub-category/{id}/branches	GET	Get branches for a subcategory	id, per_page

Favorites System

Favorite Model

```
[
    'branch_id' => 'integer|exists:branches,id',
    'user_id' => 'integer|exists:users,id',
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null',
]
```

- **Relationships**:
  - Junction table between User and Branch.

Key Endpoints

Endpoint	Method	Description	Parameters
/favorites/get	GET	Get user's favorite branches	Requires auth:api middleware

/favorites/add/{id}	POST	Add/remove favorite branch	id (branch ID)

Contact System

Contact Model

```
[
    'name' => 'string|max:255',
    'email' => 'string|max:255',
    'mobile' => 'string|max:255',
    'message' => 'string|max:12000',
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null',
]
```

Endpoint

Endpoint	Method	Description	Parameters
/contacts/store	POST	Submit contact form	name, email, mobile, message

Settings & Static Content

Models

- **About:** [

```
    'content' => 'text',
    'type' => 'integer', // 1=About Us, 2=Why Choose Us, 3=What We Offer
    'img' => 'string|null',
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null',
]
```

- **Blog**:[
```
    'title' => 'string',
    'slug' => 'string',
    'description' => 'text',
    'imgs' => 'string|null',
    'category_id' => 'integer|exists:categories,id',
    'meta_title' => 'string|null',
    'meta_description' => 'string|null',
    'meta_tags' => 'string|null',
    'meta_keywords' => 'string|null',
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null',
]
```

- **PaymentMethod**:[
```
    'name' => 'string',
    'img' => 'string|null',
    'is_activate' => 'boolean',
    'deleted_at' => 'datetime|null',
]
```

Key Endpoints

Endpoint	Method	Description	Parameters
/settings/all	GET	Get system settings	None
/abouts/{type}	GET	Get about content	type (1, 2, or 3)
/blogs	GET	List blogs	page
/blog/{id}	GET	Get blog details	id
/payments	GET	List payment methods	page

Error Handling

Standard Error Responses

```
{
    "status": 400,
    "message": "Bad Request",
    "data": "Validation error message"
}
```

Common HTTP Status Codes

- **200**: Success
- **400**: Bad Request (validation errors)
- **401**: Unauthorized (authentication failure or inactive account)
- **404**: Not Found
- **500**: Internal Server Error

Response Formats

Successful Response

```
{
    "status": 200,
    "message": "success",
    "data": {
        // Response data (e.g., user, branch, or collection)
    }
}
```

Paginated Response

```
{
    "status": 200,
    "message": "success",
    "data": {
        "data": [...], // Array of items
        "current_page": 1,
        "last_page": 5,
        "per_page": 10,
        "total": 50
    }
}
```

Database Schema

Key Tables

- **users**: Stores user data (vendors and regular users).
- **branches**: Stores branch information for vendors.
- **categories**: Stores category data for branches.
- **sub_categories**: Stores subcategory data linked to categories.
- **branch_payments**: Junction table for branch and payment method relationships.
- **days**: Stores working hours for branches.
- **user_favorites**: Junction table for user favorite branches.
- **contacts**: Stores contact form submissions.
- **abouts**: Stores static content for "About" sections.
- **blogs**: Stores blog posts.
- **payment_methods**: Stores available payment methods.
- **countries**: Stores country data (name, flag, mobile code).
- **notifications**: Stores notification records.
- **package_histories**: Stores payment history for packages.

Key Relationships

- **User** has many `Branch` (as `owner_id`).
- **Branch** belongs to `Category` and `SubCategory`.
- **Branch** has many `Day` (working hours).
- **Branch** has many-to-many with `PaymentMethod` via `branch_payments`.
- **User** has many-to-many with `Branch` via `user_favorites`.

- **Category** has many `SubCategory` and `Branch`.
- **SubCategory** has many `Branch`.

Security Considerations

- **JWT Tokens**: Configured with a 20-year expiration for long-lived sessions.
- **Password Hashing**: Uses `bcrypt` for secure password storage.
- **Input Validation**: All endpoints use Laravel's `Validator` to enforce strict input validation.
- **Database Transactions**: Critical operations (e.g., user registration, branch creation) use transactions to ensure data integrity.
- **Middleware Protection**: Sensitive endpoints are protected with `auth:api` and `userActivation` middleware.
- **Soft Deletes**: Implemented for users, branches, and other models to prevent permanent data loss.
- **File Handling**: Images are validated and stored securely; deleted files are removed from the filesystem during model deletion.

Deployment Notes

- **Required Environment Variables**:
  - `APP_URL`: Base URL for image paths.
  - `DB_*`: Database connection settings (host, database, username, password).
  - `JWT_SECRET`: Secret key for JWT token generation.
- **File Storage**:
  - User and branch images are stored in `public/uploads`.
  - Configured in `config/filesystems.php`.
- **Scheduled Tasks**:
  - Branch expiration notifications (3-month and 1-month reminders).
  - Subscription management for branches.
- **Dependencies**:
  - Install `tymon/jwt-auth`, `spatie/laravel-permission`, `spatie/laravel-activitylog`, and `spatie/laravel-medialibrary` via Composer.
- **Database Migrations**:

    - Ensure migrations are run to create tables (`users`, `branches`, `categories`, etc.).
- **External Services**:
    - Configure WhatsApp service for notifications (if used).
    - Configure mail service for email notifications (e.g., `UpdateEmail` notification).

This documentation provides a comprehensive technical overview of the API system. For detailed implementation specifics, refer to the source code and inline comments in the respective controllers and models.