SALON RESERVATION SYSTEM

Backend API Documentation

Version 1.0.0	**Framework** Laravel 12 + MySQL + Sanctum
Base URL `http://your-domain.com/api`	**Auth Method** Bearer Token (Laravel Sanctum)

February 2026

1. System Overview

The Salon Reservation System is a RESTful API built with Laravel 12 and MySQL. It supports three distinct user roles — Super Admin, Admin, and Client — each with scoped access to the system's resources. Authentication is handled via Laravel Sanctum using stateless Bearer tokens.

1.1 Key Features

- Role-based access control (Super Admin, Admin, Client)
- Secure token authentication via Laravel Sanctum
- Reservation system with business logic validation (hours, overlaps, working days)
- Admin dashboard with statistics and KPIs
- Full CRUD for services, reservations, and admin users
- Search, filter, and sort capabilities for admins
- Rate-limited login and registration endpoints

1.2 Technology Stack

Component	Technology
Backend Framework	Laravel 12
Database	MySQL 8+

Component	Technology
Authentication	Laravel Sanctum (Bearer Tokens)
Password Hashing	bcrypt
Validation	Laravel FormRequest classes
PHP Version	8.2+

2. Quick Start

2.1 Installation

```
# 1. Install dependencies
composer install

# 2. Copy environment file
cp .env.example .env

# 3. Generate application key
php artisan key:generate

# 4. Configure database in .env
DB_DATABASE=salon_db
DB_USERNAME=root
DB_PASSWORD=your_password

# 5. Run migrations
php artisan migrate

# 6. Seed the database (Super Admin + Business Settings)
php artisan db:seed

# 7. Start development server
php artisan serve
```

2.2 Default Credentials

Role	Email	Password
Super Admin	superadmin@salon.com	SuperAdmin@123

2.3 Global Request Headers

Header	Value	Required
`Content-Type`	`application/json`	Yes
`Accept`	`application/json`	Yes
`Authorization`	`Bearer {token}`	Protected routes

3. Roles & Permissions

The system implements three roles enforced by RoleMiddleware on all protected routes:

Role	Identifier	Capabilities
Super Admin	`super_admin`	Full system access — manage admins, services, business settings, view all reservations
Admin	`admin`	Manage reservations (create/update/search/filter), view dashboard, phone bookings
Client	`client`	Register, login, view services, book/view/cancel own reservations

4. Database Schema

4.1 users

Column	Type	Nullable	Notes
`id`	bigint (PK)	No	Auto-increment
`name`	varchar(255)	No	
`email`	varchar(255)	No	Unique
`phone`	varchar(20)	Yes	
`password`	varchar(255)	No	bcrypt hashed
`role`	enum	No	super_admin | admin | client
`created_at / updated_at`	timestamp	Yes	

4.2 services

Column	Type	Nullable	Notes
`id`	bigint (PK)	No	Auto-increment

Column	Type	Nullable	Notes
title	varchar(255)	No	
description	text	Yes	
duration	int (unsigned)	No	Duration in minutes
price	decimal(10,2)	No	
is_active	boolean	No	Default: true
created_by	bigint (FK)	No	References users.id

4.3 reservations

Column	Type	Nullable	Notes
id	bigint (PK)	No	Auto-increment
user_id	bigint (FK)	No	References users.id
service_id	bigint (FK)	No	References services.id
reservation_date	date	No	
start_time	time	No	
end_time	time	No	Auto-calculated
status	enum	No	pending | confirmed | cancelled | completed
notes	text	Yes	
created_by_admin	boolean	No	Default: false

4.4 business_settings

Column	Type	Nullable	Notes
id	bigint (PK)	No	
open_time	time	No	Default: 09:00:00
close_time	time	No	Default: 18:00:00
working_days	json	No	Array: 0=Sun … 6=Sat

5. Standard Response Format

All endpoints return consistent JSON envelopes:

5.1 Success Response

```
{
  "success": true,
  "message": "Operation successful.",
  "data": { ... }
}
```

5.2 Error Response

```
{
  "success": false,
  "message": "Descriptive error message.",
  "errors": {
    "field": ["Validation error detail."]
  }
}
```

5.3 HTTP Status Codes

Code	Status	Meaning
200	OK	Request succeeded
201	Created	Resource created successfully
401	Unauthorized	Missing or invalid token
403	Forbidden	Authenticated but insufficient role
404	Not Found	Resource or route not found
409	Conflict	Booking time slot conflict
422	Unprocessable	Validation failed
429	Too Many Requests	Rate limit exceeded

6. API Endpoints

6.1 Route Overview

Method	Endpoint	Description	Auth
POST	/api/register	Client self-registration	Public
POST	/api/login	Login — all roles	Public

Method	Endpoint	Description	Auth
GET	/api/services	List active services	Public
GET	/api/business-settings	View business hours	Public
POST	/api/logout	Logout current token	All Auth
GET	/api/me	Get authenticated user	All Auth
POST	/api/reservations	Book a reservation	Client
GET	/api/my-reservations	View own reservations	Client
DELETE	/api/reservation/{id}	Cancel a reservation	Client
GET	/api/admin/dashboard	Dashboard statistics	Admin/SA
GET	/api/admin/reservations	List all reservations	Admin/SA
POST	/api/admin/reservations	Manual phone booking	Admin/SA
PUT	/api/admin/reservations/{id}	Update reservation status	Admin/SA
POST	/api/services	Create a service	Super Admin
PUT	/api/services/{id}	Update a service	Super Admin
DELETE	/api/services/{id}	Delete a service	Super Admin
PUT	/api/business-settings	Update business hours	Super Admin
POST	/api/create-admin	Create admin user	Super Admin

6.2 Authentication Endpoints

POST /api/register

Registers a new client. Rate limited to 10 requests per minute.

Request Body:

```
{
  "name": "Jane Doe",
  "email": "jane@example.com",
  "phone": "+1234567890",
  "password": "password123",
  "password_confirmation": "password123"
}
```

Success Response (201):

```
{
  "success": true,
  "message": "Registered successfully.",
  "data": {
```

```
    "user": { "id": 5, "name": "Jane Doe", "email": "jane@example.com", "role": "client"
},
    "token": "1|abc123..."
  }
}
```

POST /api/login

Authenticates any user and returns a Bearer token. Rate limited to 5 requests per minute. All previous tokens are revoked on successful login.

Request Body:

```
{
  "email": "jane@example.com",
  "password": "password123"
}
```

Success Response (200):

```
{
  "success": true,
  "message": "Logged in successfully.",
  "data": {
    "user": { "id": 5, "name": "Jane Doe", "role": "client" },
    "token": "1|abc123..."
  }
}
```

POST /api/logout

Revokes the current access token. Requires Bearer token.

GET /api/me

Returns the currently authenticated user's profile. Works for all roles.

6.3 Client Endpoints

POST /api/reservations

Creates a new reservation. Performs full business logic validation. The end_time is auto-calculated from the service duration.

Request Body:

```
{
  "service_id": 1,
  "reservation_date": "2026-03-15",
  "start_time": "10:00",
  "notes": "First visit — prefer quiet corner"
}
```

Validation Rules:

- service_id — Must exist and be active
- reservation_date — Must be today or in the future
- start_time — Format HH:MM (24-hour)
- notes — Optional, max 1000 characters

Business Logic Checks:

- Reservation date must fall on a configured working day
- start_time and calculated end_time must be within open_time–close_time
- No overlap with any non-cancelled reservation in the same time window

Success Response (201):

```
{
  "success": true,
  "message": "Reservation created successfully.",
  "data": {
    "id": 42,
    "service_id": 1,
    "reservation_date": "2026-03-15",
    "start_time": "10:00:00",
    "end_time": "10:45:00",
    "status": "pending",
    "created_by_admin": false
  }
}
```

GET /api/my-reservations

Returns all reservations belonging to the authenticated client, ordered by date descending. Includes the related service object.

DELETE /api/reservation/{id}

Cancels a reservation. Sets status to 'cancelled'. Cannot be used on already-completed or already-cancelled reservations. Only the owner can cancel their own reservation.

6.4 Admin Endpoints

GET /api/admin/dashboard

Returns a comprehensive statistics object for the dashboard.

Response Data:

```
{
  "total_reservations": 248,
  "today_reservations": 7,
  "today_details": [ ... ],
  "total_revenue": "12450.00",
  "most_booked_services": [
    { "id": 1, "title": "Haircut", "reservations_count": 89 }
  ],
  "reservations_by_status": {
    "pending": 12,
    "confirmed": 45,
    "cancelled": 8,
    "completed": 183
  }
}
```

GET /api/admin/reservations

Returns a paginated list of all reservations with support for search, filtering, and sorting via query parameters.

Query Parameters:

Parameter	Type	Description
client_name	string	Partial match on client name
phone	string	Partial match on client phone
date	date (Y-m-d)	Filter by exact reservation date
status	enum	pending | confirmed | cancelled | completed
sort	asc | desc	Sort by date (default: desc)

Example:

```
GET /api/admin/reservations?client_name=Jane&status=pending&date=2026-03-15&sort=asc
```

POST /api/admin/reservations

Creates a reservation manually on behalf of a client (e.g., phone booking). Same business logic validation applies as client bookings, but date can be any date and created_by_admin is set to true.

Request Body:

```
{
  "user_id": 5,
  "service_id": 2,
  "reservation_date": "2026-03-15",
  "start_time": "14:00",
  "notes": "Phone booking — client requested window seat"
}
```

PUT /api/admin/reservations/{id}

Updates the status and/or notes of an existing reservation.

Request Body:

```
{
  "status": "confirmed",
  "notes": "Client confirmed by phone"
}
```

6.5 Super Admin Endpoints

POST /api/services

Creates a new salon service. The authenticated super admin is automatically recorded as the creator.

Request Body:

```
{
  "title": "Keratin Treatment",
  "description": "Smoothing treatment for frizzy hair.",
  "duration": 90,
  "price": 120.00,
  "is_active": true
}
```

PUT /api/services/{id}

Updates any field of an existing service. All fields are optional (partial update).

DELETE /api/services/{id}

Deletes a service. Will return a 422 error if the service has any active (non-cancelled / non-completed) reservations to prevent data integrity issues.

PUT /api/business-settings

Updates the salon's operating hours and working days. Working days are provided as an array of integers (0 = Sunday, 1 = Monday, … 6 = Saturday).

Request Body:

```
{
  "open_time": "09:00",
  "close_time": "19:00",
  "working_days": [1, 2, 3, 4, 5, 6]
}
```

POST /api/create-admin

Creates a new admin user. The new user receives the 'admin' role and can immediately log in with the provided credentials.

Request Body:

```
{
  "name": "Sarah Manager",
  "email": "sarah@salon.com",
  "phone": "+9876543210",
  "password": "Admin@Secure99",
  "password_confirmation": "Admin@Secure99"
}
```

7. Reservation Business Logic

All reservation creation — whether by a client or admin — passes through ReservationService, which enforces the following checks in order:

#	Check	Rule	Error Code
1	Working Day	reservation_date day-of-week must be in business_settings.working_days	422
2	Business Hours	start_time and auto-calculated end_time must be within open_time – close_time	422
3	Overlap Check	No non-cancelled reservation may share any time with the new window	409
4	End Time Calc	end_time = start_time + service.duration (minutes)	N/A

Overlap detection SQL logic:

```
WHERE reservation_date = '{date}'
  AND status NOT IN ('cancelled')
  AND start_time < '{new_end_time}'
  AND end_time   > '{new_start_time}'
```

Note: Clients cannot book past dates (enforced by after_or_equal:today validation rule). Admins can book any date.

8. Project Structure

```
app/
├── Http/
│   ├── Controllers/
│   │   ├── Auth/
│   │   │   └── AuthController.php          # Register, Login, Logout, Me
│   │   ├── Client/
│   │   │   ├── ServiceController.php        # Public service listing
│   │   │   └── ReservationController.php    # Book, list, cancel
│   │   ├── Admin/
│   │   │   ├── DashboardController.php      # Statistics
│   │   │   └── ReservationController.php    # CRUD + search/filter
│   │   └── SuperAdmin/
│   │       ├── ServiceController.php        # Service CRUD
│   │       ├── AdminController.php          # Create admin users
│   │       └── BusinessSettingsController.php
│   ├── Middleware/
│   │   └── RoleMiddleware.php              # role:super_admin, role:admin
│   └── Requests/
│       ├── Auth/                           # RegisterRequest, LoginRequest
│       ├── Client/                         # CreateReservationRequest
│       ├── Admin/                          # Create/UpdateReservationRequest
│       └── SuperAdmin/                     # Service, Admin, BusinessSettings
├── Models/
│   ├── User.php
│   ├── Service.php
│   ├── Reservation.php
│   └── BusinessSetting.php
├── Services/
│   └── ReservationService.php              # Core booking business logic
└── Traits/
    └── ApiResponse.php                     # Consistent JSON responses

database/
├── migrations/                             # 5 migration files
└── seeders/
    └── DatabaseSeeder.php                  # Super Admin + Business Settings
```

```
routes/
└── api.php                                  # All route definitions

bootstrap/
└── app.php                                  # Laravel 12 app config + exception handling
```

9. Security

Security Measure	Implementation
Password Hashing	bcrypt via Laravel's 'hashed' cast on User model
Token Authentication	Laravel Sanctum stateless Bearer tokens, revoked on logout and re-login
Role Enforcement	RoleMiddleware checks user.role on every protected route
Input Validation	Dedicated FormRequest class per endpoint; all fail with consistent 422 JSON
Mass Assignment	$fillable defined on all models; no $guarded = []
Rate Limiting	Login: 5 req/min | Register: 10 req/min via Laravel throttle middleware
Error Handling	Global exception handler returns JSON 401/403/404/429 — no stack traces exposed
SQL Injection	Eloquent ORM with parameterized queries throughout

10. Error Reference

HTTP Code	Trigger	Example Message
401	No/invalid token	Unauthenticated. Please login.
403	Wrong role	Unauthorized. Insufficient permissions.
404	Resource missing	Reservation not found.
409	Time conflict	This time slot is already booked. Please choose another time.
422	Validation error	Validation failed. [errors object included]
422	Non-working day	The selected date is not a working day.
422	Outside hours	Reservation must be between 09:00:00 and 18:00:00.
422	Active service	Cannot delete a service with active reservations.
422	Invalid cancel	Cannot cancel a reservation with status: completed.
429	Rate limit hit	Too many requests. Please try again later.