SaltyPadel

API CONTRACT — v1.0

Sprint 4 | PHP + MySQL + IONOS | COMP2003 Team 4

Last updated: February 2026

1. Purpose

This document defines the agreed interface between the SaltyPadel frontend (SPA) and the PHP API hosted on IONOS. It locks endpoint URLs, request/response JSON field names, and error formats to prevent integration mismatch between frontend and backend development.

All team members must read and agree to this contract before writing any implementation code.

2. Sprint 4 Scope

Sprint 4 covers backend foundations only. The following is in scope:

- ✓ **PHP API skeleton + database connection (PDO)**
- ✓ **Admin authentication (JWT token sessions)**
- ✓ **CRUD for Upcoming Events and Past Events**
- ✓ **Basic manual testing for events flow (create, read, update, delete)**
- ✓ **Full CRUD for Partners, Testimonials, and Site Settings**
- ✓ **Image upload and storage framework (IONOS)**
- ✓ **WhatsApp / Instagram / hero text site settings management**

SECTION A — Conventions

3. Conventions

These conventions apply to every endpoint in this contract. Agree on these before writing a single line of PHP or JS.

3.1 Base URL

Development: http://localhost/api/v1/ (local XAMPP or similar)

Production: https://<ionos-domain>/api/v1/ (update when IONOS domain confirmed)

⚠ *IONOS hosting note: Shared hosting often requires .htaccess rewrite rules for clean routing. If not, routing may fall back to /api/index.php?route=upcoming-events.*

3.2 API Versioning

All endpoints use the /v1/ prefix. This means future changes can be made under /v2/ without breaking the existing frontend.

```
Base:  /api/v1/
Example:  GET /api/v1/upcoming-events
```

3.3 JSON Field Naming — camelCase

All JSON field names use camelCase — both in requests and in responses. No snake_case anywhere in the API layer. (Database column names may use snake_case internally — that is fine, PHP maps them before returning.)

Wrong — do not use	Correct — use this
`event_name`	`eventName`
`event_location`	`eventLocation`
`event_date`	`eventDate`
`event_time`	`eventTime`
`booking_status`	`bookingStatus`
`booking_url`	`bookingUrl`

3.4 Date & Time Formats

eventDate: YYYY-MM-DD (ISO 8601 — e.g. 2026-07-15)
eventTime: HH:MM:SS (24-hour — e.g. 10:00:00)

3.5 Standard Response Shapes

Every endpoint returns one of these two shapes — no exceptions:

Success response:

```
{
  "success": true,
  "data": { ... }        // single object or array
}
```

Error response:

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "One or more fields are invalid",
    "details": [          // array — list every problem
      "eventName is required",
      "eventDate must be a future date"
    ]
  }
}
```

3.6 HTTP Status Codes

Code	Meaning	When to use
200	OK	Successful read / update / delete
201	Created	Successful POST — new record created
400	Bad Request	Validation errors — missing or invalid fields
401	Unauthorised	Missing or invalid JWT token
403	Forbidden	Authenticated but not allowed to do this
404	Not Found	Resource with that ID does not exist
409	Conflict	Duplicate record or constraint violation
429	Too Many Requests	Account locked after repeated failed logins
500	Internal Server Error	Unexpected crash — do not expose stack trace

SECTION B — PHP File Structure

4. PHP File Structure

Create this folder structure in the IONOS hosting root:

```
/api/
   /v1/
      /config/
         db.php              ← PDO connection (host, user, pass, dbname)
         config.php          ← JWT secret, environment settings
      /routes/
         auth.php            ← POST /login, POST /logout
         upcoming_events.php ← GET / POST / PUT / DELETE
         past_events.php     ← GET / POST / PUT / DELETE
      /middleware/
         auth_check.php      ← Validates JWT — called by all protected routes
      index.php              ← Router: reads URL, dispatches to correct route file
      test.php               ← Health check: returns { success: true, data: { status: ok
} }
   .htaccess                 ← URL rewrite rules (required for clean routing on IONOS)
```

First step: Hit /api/v1/test before touching any other endpoint. Must return { success: true }.

SECTION C — Authentication

5. Authentication

Admin-only endpoints require a JWT access token returned by the login endpoint, sent in every request header.

```
Authorization: Bearer <jwt_token>
Content-Type: application/json
```

5.1 JWT Strategy — Stateless, No Blacklist

For Sprint 4 simplicity, JWT is stateless only. No server-side blacklist is implemented. Logout is handled client-side by deleting the token from storage. Token expiry is set to 24 hours — long enough for Phil to manage events in one session without being forced to log in again.

Token expiry: 86400 seconds (24 hours)
Storage: Frontend stores token in memory (JS variable) — not localStorage for security
Logout strategy: Client deletes token. Server does not need to track it in Sprint 4.

5.2 Lockout Logic

The admins table stores failed_attempts and locked_until. The following behaviour must be implemented exactly:

Failed login: Increment failed_attempts by 1
5th failed attempt: Set locked_until = NOW() + 15 minutes
Login attempt while locked: Return 429 — do not increment counter further
Successful login: Reset failed_attempts to 0 AND set locked_until to NULL
Lock expires: locked_until < NOW() — allow login attempt again

5.3 Health Check

[GET] /api/v1/test | *Auth: Public — no auth*	
Success Response	`{ "success": true, "data": { "status": "ok" } }`
Notes	Run this first to confirm PHP + PDO + IONOS connection is working before any other work

5.4 Admin Login

[POST] /api/v1/login | *Auth: Public — no auth*	

Request Body	`{`
	`  "username": "admin",`
	`  "password": "plaintextPassword"`
	`}`

Success Response	`// 200 OK:`
	`{`
	`  "success": true,`
	`  "data": {`
	`  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",`
	`  "expiresInSeconds": 86400`
	`  }`
	`}`

Error Response	`// 401 — wrong credentials:`
	`{ "success": false, "error": { "code": "INVALID_CREDENTIALS",`
	`  "message": "Invalid username or password", "details": [] } }`
	
	`// 429 — account locked:`
	`{ "success": false, "error": { "code": "ACCOUNT_LOCKED",`

	`"message": "Account locked. Try again in 15 minutes.", "details": [] } }`

Notes	`Use password_verify() against bcrypt hash stored in DB`
	`On success: reset failed_attempts = 0 and locked_until = NULL`
	`Frontend function to replace: button_verify_login() in SaltyPadelJava.js`

5.5 Admin Logout

[POST] `/api/v1/logout` | *Auth: Protected — JWT required*	
Request Body	`No body — token in Authorization header`
Success Response	`{ "success": true, "data": { "message": "Logged out" } }`
Error Response	`{ "success": false, "error": { "code": "UNAUTHORISED", "message": "Invalid token", "details": [] } }`

Notes	`Sprint 4: server returns 200, client deletes token from memory`
	`No server-side blacklist needed in Sprint 4`
	`Frontend function: confirm_logout() in SaltyPadelJava.js`

SECTION D — Data Models

6. Data Models

6.1 Event Object (returned by API)

Both upcoming and past events return this shape. Past events omit bookingStatus and bookingUrl.

Field (camelCase)	Type	Notes
`eventId`	number	AUTO_INCREMENT primary key
`eventName`	string	Required. Max 100 chars.
`eventLocation`	string	Required. Max 255 chars. Upcoming events only.
`eventDate`	string (YYYY-MM-DD)	Required. ISO 8601.
`eventTime`	string (HH:MM:SS)	Required. 24-hour. Upcoming events only.
`bookingStatus`	string	OPEN / FULL / CANCELLED. Upcoming events only. Default: OPEN.
`bookingUrl`	string (URL)	Optional. External ticketing link. Upcoming events only. No prices.
`imagePath`	string	Optional. File path on IONOS server. Sprint 5.

6.2 Database Design Note

For Sprint 4 simplicity and clear separation of responsibilities, upcoming_events and past_events are stored in separate MySQL tables. This makes queries straightforward and maps directly to the two admin UI screens. Future versions may unify them into a single events table using a date comparison or a status flag — but that refactor belongs to Sprint 5 or later when the scope justifies it.

SECTION E — MySQL Schema (Sprint 4 Only)

7. MySQL Tables

```
CREATE TABLE admins (
  id              INT AUTO_INCREMENT PRIMARY KEY,
  username        VARCHAR(50)  NOT NULL UNIQUE,
  password        VARCHAR(255) NOT NULL,   -- bcrypt hash ONLY, never plaintext
  failed_attempts INT          DEFAULT 0,
  locked_until    DATETIME     NULL,       -- NULL = not locked
  created_at      TIMESTAMP    DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

```
CREATE TABLE upcoming_events (
  id              INT AUTO_INCREMENT PRIMARY KEY,
  event_name      VARCHAR(100) NOT NULL,
  event_location  VARCHAR(255) NOT NULL,
  event_date      DATE         NOT NULL,
  event_time      TIME         NOT NULL,
  booking_status  ENUM('OPEN','FULL','CANCELLED') DEFAULT 'OPEN',
  booking_url     VARCHAR(255) NULL,
  image_path      VARCHAR(255) NULL,
  created_at      TIMESTAMP    DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

```
CREATE TABLE past_events (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  event_name  VARCHAR(100) NOT NULL,
  event_date  DATE         NOT NULL,
  image_path  VARCHAR(255) NULL,
  created_at  TIMESTAMP    DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

Test insert: Insert 1 dummy admin with a bcrypt hashed password and test login before wiring up the frontend.

SECTION F — Endpoints

8. Upcoming Events Endpoints

[GET] `/api/v1/upcoming-events` | *Auth: Public — no auth*

Success Response

```
// 200 OK:
{ "success": true, "data": [
  {
    "eventId": 1,
    "eventName": "Summer Padel Open",
    "eventLocation": "Plymouth Sports Centre",
```

	<pre> "eventDate": "2026-07-15",
	    "eventTime": "10:00:00",
	    "bookingStatus": "OPEN",
	    "bookingUrl": "https://tickettailor.com/..."
	  }
	] }</pre>

Notes	Order by eventDate ASC (soonest first) No prices in response — ever Maps to upcoming_events div in frontend

[POST] /api/v1/upcoming-events | *Auth: Protected — JWT required*	

Request Body	<pre>{
	  "eventName": "Spring Tournament",
	  "eventLocation": "Plymouth Padel Club",
	  "eventDate": "2026-04-20",
	  "eventTime": "09:00:00",
	  "bookingStatus": "OPEN",
	  "bookingUrl": "https://tickettailor.com/..."
	}</pre>

Success Response	<pre>// 201 Created:
	{ "success": true, "data": { "eventId": 5, "message": "Event created" } }</pre>

Error Response	<pre>// 400 Validation error:
	{ "success": false, "error": { "code": "VALIDATION_ERROR",
	  "message": "Invalid fields",
	  "details": ["eventName is required", "eventDate must be a future date"] }
	}</pre>

Notes	PDO prepared statements — no raw SQL concatenation eventDate must be a future date — validate server-side Image upload not included — Sprint 5

[PUT] /api/v1/upcoming-events/{eventId} | *Auth: Protected — JWT required*	

Request Body	<pre>{
	  "eventName": "Updated Name",
	  "eventLocation": "New Venue",
	  "eventDate": "2026-05-01",
	  "eventTime": "11:00:00",
	  "bookingStatus": "FULL",
	  "bookingUrl": "https://tickettailor.com/..."
	}</pre>

Success Response	<pre>{ "success": true, "data": { "message": "Event updated" } }</pre>

Error Response	<pre>{ "success": false, "error": { "code": "NOT_FOUND",
	  "message": "Event not found", "details": [] } }</pre>

Notes	Partial updates allowed — only update fields that are present in request body

[DELETE] /api/v1/upcoming-events/{eventId} | *Auth: Protected — JWT required*	
Request Body	No body — eventId in URL path
Success Response	<pre>{ "success": true, "data": { "message": "Event deleted" } }</pre>

Error Response	<pre>{ "success": false, "error": { "code": "NOT_FOUND",
	  "message": "Event not found", "details": [] } }</pre>

Notes	Frontend confirmDelete() modal must fire before this call is made Hard delete — removes record from DB permanently

9. Past Events Endpoints

[GET] /api/v1/past-events | *Auth: Public — no auth*

Success Response	// 200 OK:
	{ "success": true, "data": [
	{
	"eventId": 3,
	"eventName": "Winter Championship",
	"eventDate": "2025-12-10"
	}
	] }

Notes	Order by eventDate DESC (most recent first) Past events only return: eventId, eventName, eventDate — no booking fields Maps to past_events div in frontend

[POST] /api/v1/past-events | *Auth: Protected — JWT required*

Request Body	{
	"eventName": "Autumn League Finals",
	"eventDate": "2025-11-20"
	}

Success Response	{ "success": true, "data": { "eventId": 8, "message": "Past event created" } }
Error Response	{ "success": false, "error": { "code": "VALIDATION_ERROR", "message": "Invalid fields", "details": ["eventDate must be in the past"] } }
Notes	eventDate must be in the past — validate server-side

[DELETE] /api/v1/past-events/{eventId} | *Auth: Protected — JWT required*

Request Body	No body — eventId in URL path
Success Response	{ "success": true, "data": { "message": "Past event deleted" } }
Error Response	{ "success": false, "error": { "code": "NOT_FOUND", "message": "Event not found", "details": [] } }
Notes	Frontend confirmDelete() modal must fire before this call is made

SECTION G — Frontend Integration

10. Frontend JS Functions to Update

These are the specific functions in SaltyPadelJava.js that need updating to call real API endpoints. All other JS stays the same for Sprint 4.

JS Function	What changes	Endpoint
`button_verify_login()`	Replace hardcoded check with fetch()	`POST /api/v1/login`
`confirm_logout()`	Client deletes token	`POST /api/v1/logout`
`button_upcoming_events()`	Load from API, not placeholder HTML	`GET /api/v1/upcoming-events`
`button_past_events()`	Load from API, not placeholder HTML	`GET /api/v1/past-events`
`button_manage_upcoming_events()`	Load + CRUD via fetch()	`All /upcoming-events endpoints`
`button_manage_past_events()`	Load + CRUD via fetch()	`All /past-events endpoints`

SECTION H — Security Requirements

11. Security Checklist

bcrypt passwords: Never store plaintext. Use PHP password_hash() and password_verify().

Parameterised queries: PDO bindParam() on every query. No string concatenation in SQL — ever.

JWT on all protected routes: auth_check.php middleware validates token before any admin action.

Account lockout: 5 failed logins = 15 minute lock. Reset counters on successful login.

Input validation: Validate and sanitise all POST fields server-side. Strip tags.

HTTPS only: Production: IONOS SSL enabled. Reject HTTP requests in production.

Logging: Log all admin actions (login, create, delete). Never log passwords or tokens.

No stack traces: Return generic 500 message to client. Log detail server-side only.

SECTION I — Open Decisions

12. Open Decisions — Team Must Confirm

These items are not yet decided. No implementation should begin on any of these until the team agrees. Document the decision here when it is made.

Open Question	Suggested Default	Team Decision
Exact IONOS domain / base URL for production	TBC once domain confirmed	_____
IONOS routing — does clean URL /api/v1/upcoming-events work or need .htaccess?	Add .htaccess rewrite rules	_____
Lockout threshold — 5 attempts / 15 min?	5 attempts / 15 min	_____
JWT expiry — 24h or shorter?	86400 seconds (24h)	_____
PUT partial update or full object required?	Partial updates allowed	_____
Image storage on IONOS — SFTP, file manager or object storage?	TBC Sprint 5	_____

SECTION J — Partners CRUD

14. Partners Endpoints

Three permanent partners only: VX3, The Padel Directory, Express Padel. Phil cannot add or remove partners — admin can only update logos. Section title: Our Proud Partners.

[GET] /api/v1/partners | *Auth: Public — no auth*	

Success Response	`// 200 OK:`
	`{ "success": true, "data": [`
	`  {`
	`  "partnerId": 1,`
	`  "partnerName": "VX3",`
	`  "logoPath": "/images/partners/vx3.png",`
	`  "websiteUrl": "https://vx3.co.uk"`
	`  }`
	`] }`

Notes	Always returns exactly 3 partners — fixed set, no add/remove

[PUT] /api/v1/partners/{partnerId}/logo | *Auth: Protected — JWT required*	
Request Body	multipart/form-data — file field: "logo"
Success Response	{ "success": true, "data": { "logoPath": "/images/partners/vx3-new.png" } }
Error Response	{ "success": false, "error": { "code": "NOT_FOUND", "message": "Partner not found", "details": [] } }
Notes	Admin can ONLY update the logo image — not name or website Accepted formats: JPG, PNG, SVG. Max 2MB. Old logo file deleted from IONOS storage on successful upload

SECTION K — Testimonials CRUD

15. Testimonials Endpoints

[GET] /api/v1/testimonials | *Auth: Public — no auth*	

Success Response	`// 200 OK:`
	`{ "success": true, "data": [`
	`  {`
	`  "testimonialId": 1,`
	`  "quoteText": "Best padel community in Plymouth!",`
	`  "authorName": "John D.",`
	`  "authorRole": "Club Member",`
	`  "isVisible": true`
	`  }`
	`] }`

Notes	Return only where isVisible = true for public page Admin GET returns all including hidden ones

[POST] /api/v1/testimonials | *Auth: Protected — JWT required*	
Request Body	`{ "quoteText": "Amazing coaching sessions!", "authorName": "Sarah M.", "authorRole": "Beginner Player", "isVisible": true }`
Success Response	`{ "success": true, "data": { "testimonialId": 4, "message": "Testimonial created" } }`
Error Response	`{ "success": false, "error": { "code": "VALIDATION_ERROR", "message": "Invalid fields", "details": ["quoteText is required"] } }`
Notes	quoteText and authorName are required isVisible defaults to true

[PUT] /api/v1/testimonials/{testimonialId} | *Auth: Protected — JWT required*	
Request Body	`{ "quoteText": "Updated quote text", "authorName": "Sarah M.", "authorRole": "Intermediate Player", "isVisible": false }`
Success Response	`{ "success": true, "data": { "message": "Testimonial updated" } }`
Error Response	`{ "success": false, "error": { "code": "NOT_FOUND", "message": "Testimonial not found", "details": [] } }`
Notes	Setting isVisible to false hides it from public page without deleting it

[DELETE] /api/v1/testimonials/{testimonialId} | *Auth: Protected — JWT required*	
Request Body	No body — testimonialId in URL path
Success Response	`{ "success": true, "data": { "message": "Testimonial deleted" } }`
Error Response	`{ "success": false, "error": { "code": "NOT_FOUND", "message": "Testimonial not found", "details": [] } }`
Notes	Frontend confirmDelete() modal must fire before this call

SECTION L — Site Settings

16. Site Settings Endpoints

Site settings allow Phil to update key links and content without touching code. All settings are key-value pairs stored in a site_settings table.

[GET] /api/v1/settings | *Auth: Public — no auth*	
Success Response	`// 200 OK: { "success": true, "data": {`

	<pre> "whatsappUrl": "https://wa.me/447700000000",
	  "instagramUrl": "https://instagram.com/saltypadel",
	  "merchandiseUrl": "https://vx3.co.uk/saltypadel",
	  "heroHeading": "Welcome to SaltyPadel",
	  "heroSubtext": "Plymouth padel community"
	} }</pre>

Notes	Frontend loads this on page init and populates WhatsApp link, Instagram link, merch link, hero text

[PUT] /api/v1/settings | *Auth: Protected — JWT required*	

Request Body	<pre>{
	  "whatsappUrl": "https://wa.me/447700900000",
	  "instagramUrl": "https://instagram.com/saltypadel",
	  "merchandiseUrl": "https://vx3.co.uk/saltypadel",
	  "heroHeading": "Welcome to SaltyPadel",
	  "heroSubtext": "Plymouth number 1 padel community"
	}</pre>

Success Response	<pre>{ "success": true, "data": { "message": "Settings updated" } }</pre>
Error Response	<pre>{ "success": false, "error": { "code": "VALIDATION_ERROR", "message": "Invalid fields", "details": ["whatsappUrl must be a valid URL"] } }</pre>
Notes	Partial update allowed — only send fields that changed Validate all URLs are valid format Phil uses this to update his WhatsApp number when it changes

SECTION M — Image Storage & Upload

17. Image Storage Framework

Images are stored as files on the IONOS server. The database stores only the file path string. IONOS provides two access methods: SFTP (file transfer tool) and File Manager (browser-based). The team must confirm which is available on Phil's plan.

Storage location: /images/events/ — event photos
/images/partners/ — partner logos
/images/gallery/ — gallery images (if added)
Accepted formats: JPG, PNG, WebP. Max file size: 5MB per image.
DB field: imagePath stores relative path only e.g. /images/events/summer-open.jpg
Access: Public URL = https://<ionos-domain>/images/events/filename.jpg

17.1 Image Upload — Events

[POST] /api/v1/upcoming-events/{eventId}/image | *Auth: Protected — JWT required*	
Request Body	multipart/form-data — file field: "image"

Success Response	<pre>{ "success": true, "data": {
	  "imagePath": "/images/events/summer-open.jpg"
	} }</pre>

Error Response	<pre>{ "success": false, "error": { "code": "INVALID_FILE",
	  "message": "File too large or wrong format", "details": [] } }</pre>

Notes	Saves file to IONOS /images/events/ folder via PHP move_uploaded_file() Updates imagePath field in upcoming_events table for that eventId Old image file deleted on replacement Same pattern applies for /api/v1/past-events/{eventId}/image

17.2 MySQL Schema Additions for Sprint 4

Add these tables alongside the events and admins tables already defined:

```
CREATE TABLE partners (
  id           INT AUTO_INCREMENT PRIMARY KEY,
  partner_name VARCHAR(100) NOT NULL,
  logo_path    VARCHAR(255) NULL,
  website_url  VARCHAR(255) NULL,
  created_at   TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

```
-- Insert the 3 permanent partners on setup:
INSERT INTO partners (partner_name, website_url) VALUES
  ('VX3', 'https://vx3.co.uk'),
  ('The Padel Directory', 'https://thepadeldirectory.co.uk'),
  ('Express Padel', 'https://expresspadel.co.uk');
```

```
CREATE TABLE testimonials (
  id          INT AUTO_INCREMENT PRIMARY KEY,
  quote_text  TEXT         NOT NULL,
  author_name VARCHAR(100) NOT NULL,
  author_role VARCHAR(100) NULL,
  is_visible  TINYINT(1)   DEFAULT 1,
  created_at  TIMESTAMP    DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

```
CREATE TABLE site_settings (
  setting_key   VARCHAR(100) PRIMARY KEY,
  setting_value TEXT         NOT NULL,
  updated_at    TIMESTAMP    DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

```
-- Insert default settings on setup:
INSERT INTO site_settings (setting_key, setting_value) VALUES
  ('whatsappUrl',     'https://wa.me/447700000000'),
  ('instagramUrl',    'https://instagram.com/saltypadel'),
  ('merchandiseUrl',  'https://vx3.co.uk/saltypadel'),
  ('heroHeading',     'Welcome to SaltyPadel'),
  ('heroSubtext',     'Plymouth padel community');
```

13. Future Additions — Sprint 5 / 6

These are confirmed future endpoints. Do not build in Sprint 4.

→ UAT and client sign-off

→ Production deployment and domain finalisation

→ Performance optimisation and final testing