Interface Specifications

Generated: November 19, 2025
Version: v1.0
Architecture: RESTful API + Livewire Components

1. API Specifications

1.1 General API Standards

Base URL

```
Production: https://erp.example.com/api
Development: http://localhost:8000/api
```

Authentication

```
Header: Authorization: Bearer {token}
Token Type: Laravel Sanctum
Expiry: Configurable (default: 1 year)
```

Content Standards

```
Content-Type: application/json
Accept: application/json
Character Encoding: UTF-8
```

Response Format

```
{
  "success": boolean,
  "data": object|array,
  "message": string,
  "meta": {
    "timestamp": "ISO 8601",
    "version": "string",
    "pagination": object
  },
  "errors": object // Only on validation errors
}
```

1.2 Authentication API Specifications

POST /api/login

Authenticates user and returns access token.

Request Body:

```
{
  "email": "required|string|email|max:255",
  "password": "required|string|min:8",
  "remember": "optional|boolean",
  "device_name": "optional|string|max:255"
}
```

Success Response (200):

```
{
  "success": true,
  "data": {
    "user": {
      "id": 1,
      "name": "John Doe",
      "email": "john@example.com",
      "current_organization_id": 1,
      "organizations": [
        {
          "id": 1,
          "name": "Acme Corp",
          "role": "admin"
        }
      ]
    },
    "token": "1|abcdefghijklmnopqrstuvwxyz123456",
    "abilities": [""],
    "expires_at": "2026-11-19T12:00:00Z"
  },
  "message": "Login successful"
}
```

Error Responses:
- `401`: *Invalid credentials*
- `422`: *Validation errors*
- `429`: *Too many login attempts*

POST /api/logout

Revokes user's authentication token.

Headers: `Authorization: Bearer {token}`
Success Response (204): *No Content*
Error Response (401): *Invalid token*

1.3 Organization API Specifications

GET /api/organizations

List user's organizations with pagination.

Query Parameters:

```
page: integer (default: 1)
per_page: integer (default: 15, max: 100)
search: string (searches in name, description)
status: enum[active,inactive] (default: active)
sort: enum[name,created_at] (default: created_at)
order: enum[asc,desc] (default: desc)
```

Success Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "name": "Acme Corporation",
      "description": "Pharmaceutical distribution company",
      "is_active": true,
      "created_at": "2025-01-01T00:00:00Z",
      "updated_at": "2025-01-01T00:00:00Z",
      "members_count": 25,
      "stores_count": 3,
      "employees_count": 45,
      "monthly_revenue": 150000.00
    }
  ],
  "meta": {
    "pagination": {
      "current_page": 1,
      "from": 1,
      "last_page": 1,
      "per_page": 15,
      "to": 1,
      "total": 1
    }
  }
}
```

POST /api/organizations

Create new organization.

Request Body:

```
{
  "name": "required|string|max:255|unique:organizations",
  "description": "nullable|string|max:1000",
  "is_active": "optional|boolean"
}
```

Success Response (201):

```
{
  "success": true,
  "data": {
    "id": 2,
    "name": "New Company Ltd",
    "description": "Company description",
    "is_active": true,
    "created_at": "2025-11-19T12:00:00Z",
    "updated_at": "2025-11-19T12:00:00Z"
  },
  "message": "Organization created successfully"
}
```

GET /api/organizations/{id}

Retrieve organization details with relationships.

Success Response (200):

```json
{
  "success": true,
  "data": {
    "id": 1,
    "name": "Acme Corporation",
    "description": "Pharmaceutical distribution company",
    "is_active": true,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z",
    "members": [
      {
        "id": 1,
        "user": {
          "id": 1,
          "name": "John Doe",
          "email": "john@example.com"
        },
        "role": "admin",
        "position": "System Administrator",
        "joined_at": "2025-01-01T00:00:00Z"
      }
    ],
    "units": [
      {
        "id": 1,
        "name": "Head Office",
        "type": "department",
        "parent_id": null,
        "members_count": 15,
        "children": [...]
      }
    ],
    "stores": [
      {
        "id": 1,
        "name": "Main Store",
        "code": "STORE001",
        "location": "Building A, Floor 1",
        "manager": {
          "id": 1,
          "name": "Store Manager"
        }
      }
    ]
  }
}
```

1.4 Employee API Specifications

GET /api/employees

GET /api/employees

List employees with advanced filtering.

Query Parameters:

```
page: integer
per_page: integer
search: string (searches first_name, last_name, email, employee_code)
department_id: integer
position_id: integer
shift_id: integer
status: enum[active,inactive]
gender: enum[male,female,other]
date_of_birth_from: date
date_of_birth_to: date
created_at_from: date
created_at_to: date
sort: enum[name,email,created_at,position]
order: enum[asc,desc]
```

Success Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "user_id": 1,
      "employee_code": "EMP001",
      "first_name": "John",
      "last_name": "Doe",
      "middle_name": "William",
      "email": "john.doe@company.com",
      "phone": "+1234567890",
      "date_of_birth": "1990-01-15",
      "gender": "male",
      "address": "123 Main Street, Apt 4B",
      "city": "New York",
      "state": "NY",
      "country": "USA",
      "zip_code": "10001",
      "photo": "https://example.com/storage/photos/john.jpg",
      "is_active": true,
      "is_admin": false,
      "biometric_id": "BIO001",
      "position": {
        "id": 1,
        "title": "Software Developer",
        "code": "DEV001",
        "department": {
          "id": 1,
          "name": "IT Department"
          "name": "IT Department"
        }
      },
      "shift": {
        "id": 1,
        "name": "Regular Shift",
        "code": "REG001",
        "start_time": "09:00:00",
        "end_time": "17:00:00",
        "working_hours": 8
      },
      "organization_unit": {
        "id": 1,
        "name": "IT Department",
        "type": "department"
      },
      "attendance_today": {
        "punch_in": "2025-11-19T09:00:00Z",
        "punch_out": null,
        "total_hours": null,
        "status": "present",
        "late_minutes": 0
      },
      "leave_balances": {
        "annual": 15,
        "sick": 10,
        "personal": 5
      },
      "created_at": "2025-01-01T00:00:00Z",
      "updated_at": "2025-11-19T10:30:00Z"
    }
  ],
  "meta": {
    "pagination": {...},
    "filters": {
      "departments": [...],
      "positions": [...],
      "shifts": [...]
    }
  }
}
```

POST /api/employees

Create new employee record.

Request Body:

```
{
  "first_name": "required|string|max:255",
  "last_name": "required|string|max:255",
  "middle_name": "nullable|string|max:255",
  "email": "required|email|max:255|unique:users,email",
  "phone": "nullable|string|max:20",
  "date_of_birth": "required|date|before:today",
  "gender": "required|in:male,female,other",
  "address": "nullable|string|max:500",
  "city": "nullable|string|max:100",
  "state": "nullable|string|max:100",
  "country": "nullable|string|max:100",
  "zip_code": "nullable|string|max:20",
  "position_id": "required|exists:job_positions,id",
  "shift_id": "nullable|exists:shifts,id",
  "organization_unit_id": "required|exists:organization_units,id",
  "biometric_id": "nullable|string|max:50|unique:employees,biometric_id",
  "is_active": "optional|boolean",
  "is_admin": "optional|boolean",
  "create_user_account": "optional|boolean",
  "user_permissions": "optional|array"
}
```

Success Response (201):

```
{
  "success": true,
  "data": {
    "id": 45,
    "employee_code": "EMP045",
    "first_name": "Jane",
    "last_name": "Smith",
    "email": "jane.smith@company.com",
    "position": {...},
    "organization_unit": {...},
    "created_at": "2025-11-19T12:00:00Z"
  },
  "message": "Employee created successfully"
}
```

1.5 Financial API Specifications

GET /api/accounts

List chart of accounts.

Query Parameters:

```
page: integer
per_page: integer
search: string
type: enum[asset,liability,equity,revenue,expense]
parent_id: integer
is_active: boolean
```

Success Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "code": "1001",
      "name": "Cash Account",
      "type": "asset",
      "description": "Primary cash account",
      "parent_id": null,
      "balance": 50000.00,
      "is_active": true,
      "created_at": "2025-01-01T00:00:00Z",
      "children": [
        {
          "id": 2,
          "code": "1001.01",
          "name": "Main Cash",
          "type": "asset",
          "balance": 30000.00
        }
      ]
    }
  ]
}
```

POST /api/journal-entries

Create journal entry with double-entry validation.

Request Body:

```
{
  "reference_number": "required|string|max:50|unique:journal_entries",
  "entry_date": "required|date",
  "description": "required|string|max:1000",
  "voucher_type": "nullable|in:sales,purchase,expense,salary,general",
  "customer_id": "nullable|exists:customers,id",
  "vendor_id": "nullable|exists:vendors,id",
  "entries": "required|array|min:2",
  "entries..chart_of_account_id": "required|exists:chart_of_accounts,id",
  "entries..type": "required|in:debit,credit",
  "entries..amount": "required|numeric|min:0.01",
  "entries..description": "nullable|string|max:500"
}
```

Business Rules:
- Total debits must equal total credits
- At least one debit and one credit entry required
- Account types must match debit/credit rules
- Cannot debit revenue or equity accounts
- Cannot credit asset or expense accounts

Success Response (201):

```
{
  "success": true,
  "data": {
    "id": 123,
    "reference_number": "JE001",
    "entry_date": "2025-11-19",
    "description": "Office supplies purchase",
    "status": "draft",
    "total_debits": 1000.00,
    "total_credits": 1000.00,
    "is_balanced": true,
    "entries": [
      {
        "id": 1,
        "chart_of_account": {
          "id": 1,
          "code": "5001",
          "name": "Office Expenses"
        },
        "type": "debit",
        "amount": 1000.00
      },
      {
        "id": 2,
        "chart_of_account": {
          "id": 2,
          "code": "1001",
          "name": "Cash Account"
        },
        "type": "credit",
        "amount": 1000.00
      }
    ],
    "created_at": "2025-11-19T12:00:00Z"
  },
  "message": "Journal entry created successfully"
}
```

1.6 Inventory API Specifications

GET /api/inventory/items

List inventory items with stock information.

Query Parameters:

```
page: integer
per_page: integer
search: string (searches name, code, description)
category: string
store_id: integer
stock_status: enum[in_stock,low_stock,out_of_stock,overstock]
min_price: decimal
max_price: decimal
has_expiry: boolean
sort: enum[name,code,price,stock_level]
order: enum[asc,desc]
```

Success Response (200):

```json
{
  "success": true,
  "data": [
    {
      "id": 1,
      "name": "Paracetamol 500mg",
      "code": "PAR001",
      "description": "Paracetamol tablets 500mg",
      "category": "Medicines",
      "unit": "strips",
      "cost_price": 25.50,
      "selling_price": 35.00,
      "min_stock_level": 50,
      "max_stock_level": 500,
      "total_stock": 150,
      "stock_status": "adequate",
      "stock_value": 3825.00,
      "stores": [
        {
          "store_id": 1,
          "store_name": "Main Store",
          "quantity": 100,
          "location": "Aisle 3, Shelf B",
          "last_updated": "2025-11-19T10:00:00Z"
        },
        {
          "store_id": 2,
          "store_name": "Branch Store",
          "quantity": 50,
          "location": "Storage Area A",
          "last_updated": "2025-11-18T15:30:00Z"
        }
      ],
      "batch_info": {
        "total_batches": 3,
        "earliest_expiry": "2026-06-30",
        "latest_expiry": "2026-12-31"
      },
      "is_active": true,
      "created_at": "2025-01-01T00:00:00Z",
      "updated_at": "2025-11-19T10:00:00Z"
    }
  ],
  "meta": {
    "pagination": {...},
    "filters": {
      "categories": ["Medicines", "Equipment", "Supplies"],
      "stores": [...],
      "stock_statuses": ["in_stock", "low_stock", "out_of_stock"]
    }
  }
}
```

POST /api/inventory/transactions

Create inventory transaction.

Request Body:

```
{
  "store_id": "required|exists:inventory_stores,id",
  "transaction_type": "required|in:IN,OUT,TRANSFER,ADJUST",
  "reference_number": "required|string|max:50",
  "transaction_date": "required|date",
  "notes": "nullable|string|max:1000",
  "transfer_to_store_id": "required_if:transaction_type,TRANSFER|exists:inventory_stores,id",
  "items": "required|array|min:1",
  "items..item_id": "required|exists:inventory_items,id",
 "items..quantity": "required|numeric|min:0.01",
  "items..unit_cost": "nullable|numeric|min:0",
 "items..batch_number": "nullable|string|max:50",
  "items..expiry_date": "nullable|date",
 "items..notes": "nullable|string|max:500"
}
```

Business Rules:
- OUT transactions require sufficient stock
- TRANSFER transactions require source and destination stores
- ADJUST transactions can increase or decrease stock
- Batch numbers required for pharmaceutical items
- Expiry dates required for items with expiry tracking

Success Response (201):

```
{
  "success": true,
  "data": {
    "id": 789,
    "reference_number": "GRN001",
    "transaction_type": "IN",
    "status": "draft",
    "store": {
      "id": 1,
      "name": "Main Store"
    },
    "total_items": 5,
    "total_quantity": 500,
    "total_value": 12750.00,
    "items": [
      {
        "id": 1,
        "item": {
          "id": 1,
          "name": "Paracetamol 500mg",
          "code": "PAR001"
        },
        "quantity": 100,
        "unit_cost": 25.50,
        "total_cost": 2550.00,
        "batch_number": "BATCH001",
        "expiry_date": "2026-12-31"
      }
    ],
    "created_at": "2025-11-19T12:00:00Z"
  },
  "message": "Transaction created successfully"
}
```

---

2. Livewire Component Specifications

2.1 Component Architecture

Component Structure

```
class ExampleComponent extends Component
{
    // Public properties (available to Blade)
    public $property;

    // Protected properties (component internal)
    protected $listeners = ['eventName' => 'handleMethod'];

    // Component mount
    public function mount($parameter = null)
    {
        // Initialization logic
    }

    // Public methods (callable from Blade)
    public function methodName()
    {
        // Component logic
    }

    // Rendering
    public function render()
    {
        return view('components.example');
    }
}
```

Blade Integration

Submit

2.2 Dashboard Component Specifications

Dashboard Component

Real-time dashboard with KPIs and quick actions.

Public Properties:

```
public $stats = [
    'total_employees' => 0,
    'active_employees' => 0,
    'total_stores' => 0,
    'low_stock_items' => 0,
    'today_transactions' => 0,
    'pending_leave_requests' => 0,
    'monthly_revenue' => 0.00,
    'monthly_expenses' => 0.00
];


public $recentActivities = [];
public $lowStockItems = [];
public $upcomingEvents = [];
public $quickActions = [];
```

Methods:

```
public function mount()
{
    $this->loadDashboardData();
}


public function loadDashboardData()
{
    $this->stats = $this->calculateStats();
    $this->recentActivities = $this->getRecentActivities();
    $this->lowStockItems = $this->getLowStockItems();
}


public function refreshStats()
{
    $this->loadDashboardData();
    $this->dispatch('stats-refreshed');
}
```

Events:
- `stats-refreshed` - Fired when stats are refreshed
- `quick-action-triggered` - Fired when quick action is clicked

Real-time Features:
- Auto-refresh every 30 seconds
- Live updates via WebSocket (planned)
- Loading states during data fetch

2.3 Employee Management Component Specifications

EmployeeManagement Component

Complete employee CRUD with advanced filtering.

Public Properties:

```
public $employees = [];
public $search = '';
public $filters = [
    'department' => '',
    'position' => '',
    'status' => 'active',
    'gender' => ''
];
public $selectedEmployee = null;
public $showCreateModal = false;
public $showEditModal = false;
public $pagination = 15;
```

Methods:

```
public function updatedSearch()
{
    $this->resetPage();
    $this->loadEmployees();
}


public function updatedFilters()
{
    $this->resetPage();
    $this->loadEmployees();
}


public function createEmployee()
{
    $this->showCreateModal = true;
}


public function editEmployee($employeeId)
{
    $this->selectedEmployee = Employee::find($employeeId);
    $this->showEditModal = true;
}


public function deleteEmployee($employeeId)
{
    Employee::find($employeeId)->delete();
    $this->loadEmployees();
    $this->dispatch('employee-deleted', $employeeId);
}
```

Validation Rules:

```
protected $rules = [
    'first_name' => 'required|string|max:255',
    'last_name' => 'required|string|max:255',
    'email' => 'required|email|unique:users,email',
    'position_id' => 'required|exists:job_positions,id',
    'organization_unit_id' => 'required|exists:organization_units,id'
];
```

2.4 Inventory Management Component Specifications

TransactionWizard Component

Multi-step transaction creation wizard.

Public Properties:

```
public $currentStep = 1;
public $totalSteps = 4;
public $transactionData = [
    'transaction_type' => '',
    'store_id' => '',
    'reference_number' => '',
    'notes' => '',
    'items' => []
];
public $availableItems = [];
public $selectedStore = null;
```

Methods:

```
public function nextStep()
{
    if ($this->validateCurrentStep()) {
        $this->currentStep++;
        $this->loadStepData();
    }
}


public function previousStep()
{
    $this->currentStep--;
}


public function finalizeTransaction()
{
    $this->validate($this->getFinalValidationRules());

    $transaction = InventoryTransaction::create([
        'organization_id' => auth()->user()->current_organization_id,
        // ... other fields
    ]);

    $this->dispatch('transaction-created', $transaction->id);
    $this->resetWizard();
}
```

Step Validation:
- Step 1: Transaction type selection
- Step 2: Store and reference details
- Step 3: Item selection and quantities
- Step 4: Review and confirmation

---

3. Error Handling Specifications

3.1 API Error Responses

Validation Error (422)

```
{
  "success": false,
  "message": "The given data was invalid.",
  "errors": {
    "email": [
      "The email field is required.",
      "The email must be a valid email address."
    ],
    "password": [
      "The password field is required.",
      "The password must be at least 8 characters."
    ]
  }
}
```

Authorization Error (403)

```
{
  "success": false,
  "message": "You do not have permission to perform this action.",
  "error": "INSUFFICIENT_PERMISSIONS",
  "required_permission": "employees.create",
  "available_permissions": ["employees.view", "attendance.view"]
}
```

Not Found Error (404)

```
{
  "success": false,
  "message": "The requested resource was not found.",
  "error": "RESOURCE_NOT_FOUND",
  "resource_type": "Employee",
  "resource_id": 999
}
```

Server Error (500)

```
{
  "success": false,
  "message": "An internal server error occurred.",
  "error": "INTERNAL_SERVER_ERROR",
  "timestamp": "2025-11-19T12:00:00Z",
  "request_id": "req_abc123def456"
}
```

3.2 Livewire Error Handling

Validation Errors

```
public function updated($propertyName)
{
    $this->validateOnly($propertyName);
}


public function save()
{
    try {
        $this->validate();
        // Save logic
    } catch (ValidationException $e) {
        $this->dispatch('validation-error', $e->getMessage());
    }
}
```

Exception Handling

```
public function deleteRecord($id)
{
    try {
        $record = Model::findOrFail($id);
        $record->delete();
        $this->dispatch('record-deleted', $id);
    } catch (ModelNotFoundException $e) {
        $this->dispatch('error', 'Record not found');
    } catch (Exception $e) {
        Log::error('Delete failed: ' . $e->getMessage());
        $this->dispatch('error', 'Delete operation failed');
    }
}
```

---

4. Performance Specifications

4.1 API Performance Requirements

Response Time Targets

- Simple GET requests: <200ms
- Complex filtered queries: <500ms
- POST/PUT requests: <1000ms
- Bulk operations: <3000ms

Pagination Limits

- Default page size: 15 records
- Maximum page size: 100 records
- Maximum depth for nested includes: 3 levels

Rate Limiting

- Authenticated users: 1000 requests/hour
- Unauthenticated users: 100 requests/hour
- Bulk operations: 10 requests/hour

4.2 Livewire Performance Requirements

Component Load Time

- Simple components: <500ms
- Complex components: <1500ms
- Data-heavy components: <3000ms

Memory Usage

- Simple components: <10MB
- Complex components: <50MB
- Maximum per request: 128MB

---

5. Security Specifications

5.1 Authentication Security

Password Requirements

- Minimum length: 8 characters
- Must contain: uppercase, lowercase, number
- Optional: special character
- Password history: Last 5 passwords not allowed

Token Security

- Token length: 64 characters minimum
- Token expiry: Configurable (default 1 year)
- Token rotation: Available on demand
- Device tracking: Multiple device support

5.2 Data Security

Encryption

- Data at rest: AES-256 encryption
- Data in transit: TLS 1.3
- Sensitive fields: Additional encryption layer
- Key rotation: Automated annually

Access Control

- Role-based permissions: Granular control
- Organization isolation: Complete data separation
- API key management: Scoped permissions
- Audit logging: All access attempts logged

---

6. Integration Specifications

6.1 Webhook Specifications

Event Webhooks

```
{
  "event": "employee.created",
  "timestamp": "2025-11-19T12:00:00Z",
  "organization_id": 1,
  "data": {
    "employee_id": 45,
    "name": "Jane Smith",
    "email": "jane.smith@company.com"
  },
  "signature": "sha256_hash_of_payload"
}
```

Supported Events

- `employee.created`, `employee.updated`, `employee.deleted`
- `transaction.created`, `transaction.finalized`
- `journal_entry.posted`, `journal_entry.voided`
- `leave.requested`, `leave.approved`, `leave.rejected`

6.2 File Upload Specifications

Upload Endpoints

```
POST /api/upload/employees/photos
POST /api/upload/documents/attachments
POST /api/upload/reports/exports
```

File Specifications

- Maximum file size: 10MB
- Allowed formats: JPG, PNG, PDF, DOC, DOCX, XLS, XLSX
- Storage: Local or S3 compatible
- Retention: Configurable per file type

This interface specification document provides detailed technical specifications for all system interfa
as of November 19, 2025.*