UrbanFleet Delivery Services

Application Specification Document

Version: 1.0
Date: January 23, 2026
Platform: Node.js + Express

Table of Contents

---

1. Executive Summary

UrbanFleet Delivery Services is a comprehensive web platform designed for a UAE-based delivery workforce solutions company. The application serves three primary user segments:

- **Businesses** seeking professional delivery rider workforce
- **Contractors** offering fleet vehicle partnerships
- **Riders** looking for career opportunities with visa sponsorship

The platform features a modern, responsive design with the company's navy blue (#1a2744) and orange (#F56A07) branding, professional imagery, and streamlined form submissions via email.

---

2. System Overview

2.1 Purpose

The application provides a digital presence for UrbanFleet Delivery Services, enabling:

- Online rider job applications
- Contractor fleet registration

- Business partnership inquiries
- Company information and service showcase

2.2 Key Features

- Responsive single-page website design
- Three specialized application/inquiry forms
- EmailJS integration for form submissions
- PostgreSQL database for data persistence
- Static HTML export capability for external hosting

2.3 Target Users

User Type	Description
Delivery Riders	Individuals seeking employment with visa sponsorship
Fleet Contractors	Companies with vehicle fleets for partnership
Business Clients	Companies needing delivery workforce solutions
Administrators	Internal staff managing applications

---

3. Technology Stack

3.1 Backend

Component	Technology	Version
Runtime	Node.js	20.x
Framework	Express.js	4.x
Language	TypeScript	5.x
ORM	Drizzle ORM	Latest
Database	PostgreSQL	15.x
Validation	Zod	3.x

3.2 Frontend

Component	Technology	Version
Framework	React	18.x
Build Tool	Vite	5.x
Routing	Wouter	3.x
State Management	TanStack React Query	5.x
Styling	Tailwind CSS	3.x
UI Components	shadcn/ui + Radix UI	Latest
Animations	Framer Motion	11.x
Forms	React Hook Form + Zod	Latest
Icons	Lucide React	Latest

3.3 External Services

Service	Purpose
EmailJS	Form submission email delivery
PostgreSQL (Neon)	Cloud database hosting

---

4. Architecture

4.1 Directory Structure

```
urbanfleet/
├── client/                    # Frontend React application
│   ├── src/
│   │   ├── components/ui/     # Reusable UI components
│   │   ├── hooks/             # Custom React hooks
│   │   ├── lib/               # Utilities and services
│   │   │   └── emailService.ts
│   │   └── pages/             # Page components
│   │       ├── home.tsx
│   │       ├── rider-application.tsx
│   │       ├── contractor-application.tsx
│   │       └── business-inquiry.tsx
│   └── index.html
├── server/                    # Backend Express application
│   ├── index.ts               # Server entry point
│   ├── routes.ts              # API route definitions
│   ├── storage.ts             # Database operations
│   └── vite.ts                # Vite middleware
├── shared/                    # Shared code
│   └── schema.ts              # Database schema + types
├── html-export/               # Static site build output
└── attached_assets/           # Images and media
```

4.2 Data Flow

```
User Input → React Form → Validation (Zod) → EmailJS API → Email Delivery
                                                  ↓
                                        Express API → Storage Layer → PostgreSQL
```

---

5. Database Schema

5.1 Users Table

```sql
CREATE TABLE users (
    id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
    username TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL
);
```

5.2 Rider Applications Table

```
CREATE TABLE rider_applications (
    id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
    full_name TEXT NOT NULL,
    email TEXT NOT NULL,
    phone TEXT NOT NULL,
    nationality TEXT NOT NULL,
    current_location TEXT NOT NULL,
    visa_status TEXT NOT NULL,
    has_uae_driving_license TEXT NOT NULL,
    license_type TEXT,
    years_of_experience INTEGER NOT NULL,
    vehicle_type TEXT NOT NULL,
    owns_vehicle TEXT NOT NULL,
    available_to_start TEXT NOT NULL,
    preferred_work_area TEXT NOT NULL,
    english_proficiency TEXT NOT NULL,
    additional_notes TEXT,
    created_at TIMESTAMP DEFAULT NOW() NOT NULL
);
```

5.3 Contractor Applications Table

```
CREATE TABLE contractor_applications (
    id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
    company_name TEXT NOT NULL,
    contact_person TEXT NOT NULL,
    email TEXT NOT NULL,
    phone TEXT NOT NULL,
    trade_license TEXT NOT NULL,
    emirate TEXT NOT NULL,
    fleet_motorcycles INTEGER DEFAULT 0,
    fleet_cars INTEGER DEFAULT 0,
    fleet_vans INTEGER DEFAULT 0,
    fleet_trucks INTEGER DEFAULT 0,
    fleet_bicycles INTEGER DEFAULT 0,
    total_drivers INTEGER NOT NULL,
    years_in_business INTEGER NOT NULL,
    current_clients TEXT,
    insurance_coverage TEXT NOT NULL,
    additional_services TEXT,
    additional_notes TEXT,
    created_at TIMESTAMP DEFAULT NOW() NOT NULL
);
```

5.4 Business Inquiries Table

```
CREATE TABLE business_inquiries (
    id VARCHAR PRIMARY KEY DEFAULT gen_random_uuid(),
    company_name TEXT NOT NULL,
    contact_person TEXT NOT NULL,
    email TEXT NOT NULL,
    phone TEXT NOT NULL,
    industry TEXT NOT NULL,
    company_size TEXT NOT NULL,
    emirate TEXT NOT NULL,
    delivery_volume TEXT NOT NULL,
    vehicle_types_needed TEXT NOT NULL,
    riders_needed INTEGER NOT NULL,
```

```
    start_date TEXT NOT NULL,
    contract_duration TEXT NOT NULL,
    special_requirements TEXT,
    additional_notes TEXT,
    created_at TIMESTAMP DEFAULT NOW() NOT NULL
);
```

---

6. API Endpoints

6.1 Rider Applications

POST `/api/rider-applications`

Creates a new rider job application.

Request Body:

```
{
    "fullName": "string (required)",
    "email": "string (required)",
    "phone": "string (required)",
    "nationality": "string (required)",
    "currentLocation": "string (required)",
    "visaStatus": "string (required)",
    "hasUaeDrivingLicense": "string (required)",
    "licenseType": "string (optional)",
    "yearsOfExperience": "integer (required)",
    "vehicleType": "string (required)",
    "ownsVehicle": "string (required)",
    "availableToStart": "string (required)",
    "preferredWorkArea": "string (required)",
    "englishProficiency": "string (required)",
    "additionalNotes": "string (optional)"
}
```

Response (201 Created):

```
{
    "success": true,
    "data": {
        "id": "uuid",
        "fullName": "string",
        ...
        "createdAt": "timestamp"
    }
}
```

Response (400 Bad Request):

```
{
    "success": false,
    "error": "Invalid application data"
}
```

6.2 Contractor Applications

POST `/api/contractor-applications`

Creates a new contractor fleet registration.

Request Body:

```
{
    "companyName": "string (required)",
    "contactPerson": "string (required)",
    "email": "string (required)",
    "phone": "string (required)",
    "tradeLicense": "string (required)",
    "emirate": "string (required)",
    "fleetMotorcycles": "integer (optional, default: 0)",
    "fleetCars": "integer (optional, default: 0)",
    "fleetVans": "integer (optional, default: 0)",
    "fleetTrucks": "integer (optional, default: 0)",
    "fleetBicycles": "integer (optional, default: 0)",
    "totalDrivers": "integer (required)",
    "yearsInBusiness": "integer (required)",
    "currentClients": "string (optional)",
    "insuranceCoverage": "string (required)",
    "additionalServices": "string (optional)",
    "additionalNotes": "string (optional)"
}
```

Response: Same structure as rider applications.

6.3 Business Inquiries

POST `/api/business-inquiries`

Creates a new business partnership inquiry.

Request Body:

```
{
    "companyName": "string (required)",
    "contactPerson": "string (required)",
    "email": "string (required)",
    "phone": "string (required)",
    "industry": "string (required)",
    "companySize": "string (required)",
    "emirate": "string (required)",
    "deliveryVolume": "string (required)",
    "vehicleTypesNeeded": "string (required)",
    "ridersNeeded": "integer (required)",
    "startDate": "string (required)",
    "contractDuration": "string (required)",
    "specialRequirements": "string (optional)",
    "additionalNotes": "string (optional)"
}
```

Response: Same structure as rider applications.

---

7. Frontend Pages

7.1 Home Page (`/`)

- Hero section with animated elements
- Service overview (Businesses, Contractors, Riders)
- Fleet showcase carousel (Bicycles, Motorcycles, Cars, Vans, Trucks)
- Statistics and compliance information
- Join the Team section
- Contact information and footer

7.2 Rider Application (`/rider-application`)

- Multi-field application form
- Personal information collection
- License and experience verification
- Vehicle preference selection
- EmailJS submission integration

7.3 Contractor Application (`/contractor-application`)

- Company information form
- Fleet inventory details (5 vehicle types)
- Driver count and experience
- Insurance and licensing verification
- EmailJS submission integration

7.4 Business Inquiry (`/business-inquiry`)

- Company details form
- Delivery requirements specification
- Vehicle type and rider quantity needs
- Contract duration preferences
- EmailJS submission integration

---

8. Email Integration

8.1 EmailJS Configuration

Parameter	Value
Service ID	service_ee9x1tw
Template ID	template_6wdb2sv
Public Key	o0pB4SDymxzdZV06g
Recipient	info@urbanfleetdelivery.ae

8.2 Email Service Implementation

Location: `client/src/lib/emailService.ts`

The service provides three formatting functions:

- `formatRiderApplication()` - Formats rider application data
- `formatContractorApplication()` - Formats contractor data with fleet breakdown
- `formatBusinessInquiry()` - Formats business inquiry details

Each form submission triggers:

1. Client-side validation (Zod)
2. EmailJS API call with formatted data
3. Optional database storage via Express API

---

9. Security Considerations

9.1 Data Validation

- All form inputs validated using Zod schemas
- Server-side validation before database insertion
- Type-safe data handling with TypeScript

9.2 Database Security

- UUIDs for primary keys (non-sequential)
- Parameterized queries via Drizzle ORM
- Environment variable for database credentials

9.3 API Security

- Input sanitization on all endpoints
- Error messages don't expose internal details
- CORS configured for production domains

---

10. Deployment

10.1 Development

```
npm run dev
```

Starts Vite dev server on port 5000.

10.2 Production Build

```
npm run build
```

Outputs:

- `dist/public/` - Static frontend assets
- `dist/index.cjs` - Bundled server

10.3 Static Export

The `html-export/` directory contains a complete static build suitable for deployment to any web server without Node.js backend requirements. Forms function via EmailJS client-side integration.

10.4 Environment Variables

Variable	Description
DATABASE_URL	PostgreSQL connection string
NODE_ENV	development / production

Document Information

Prepared for: UrbanFleet Delivery Services
Document Type: Technical Specification
Classification: Internal Use

End of Document