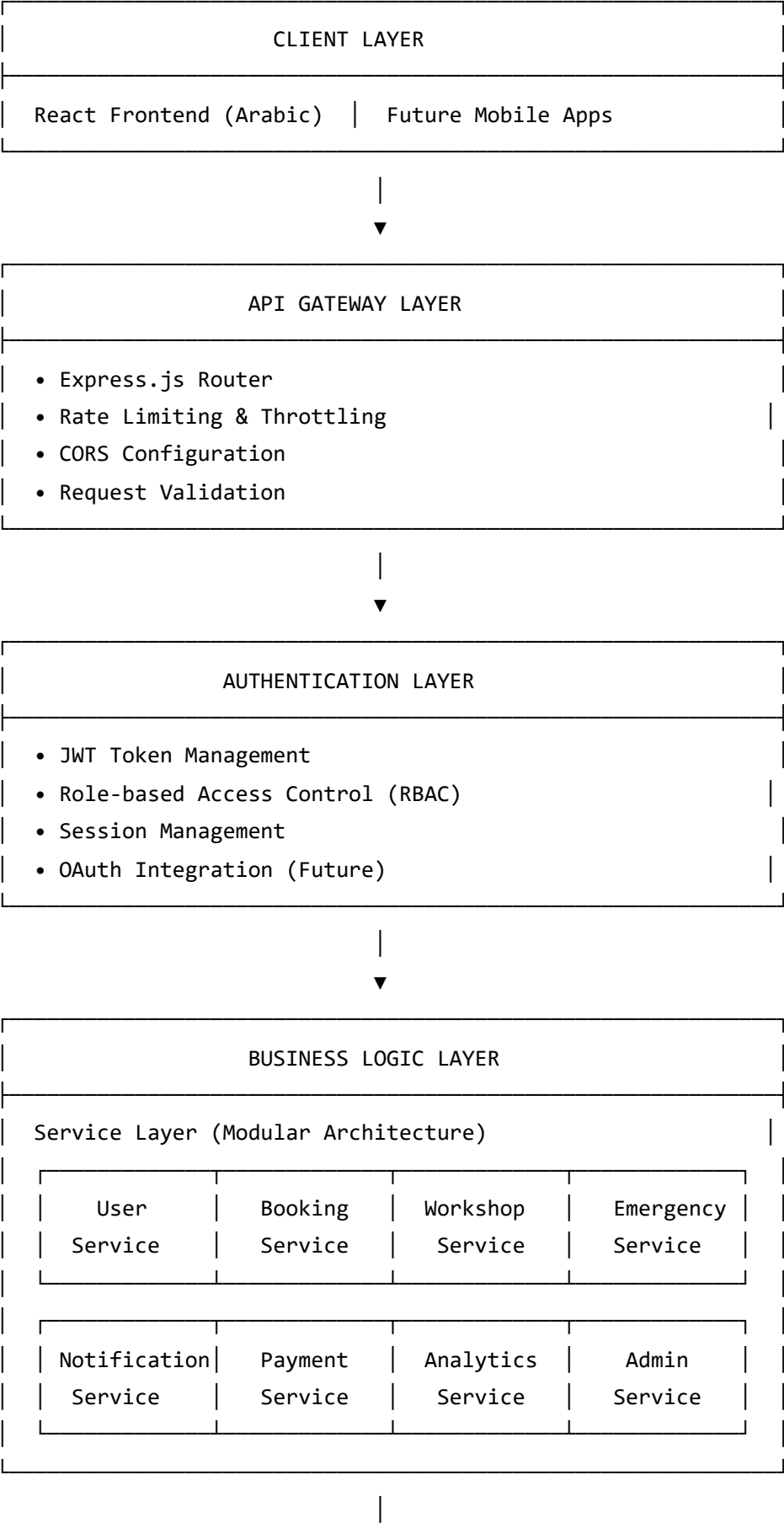


Car Service Marketplace - Main Architecture

System Architecture Overview

Backend Architecture (Node.js + TypeScript +

MongoDB)





DATA ACCESS LAYER

- MongoDB with Mongoose ODM
- Repository Pattern
- Data Validation & Sanitization
- Connection Pooling



EXTERNAL SERVICES

- Real-time Communication (Socket.IO)
- Push Notifications (Firebase/OneSignal)
- Email Service (Nodemailer)
- SMS Service (Twilio)
- Maps & Geolocation (Google Maps API)
- File Storage (Cloudinary/AWS S3)

Technology Stack

Backend Dependencies

```
{
  "dependencies": {
    "express": "^4.18.0",
    "typescript": "^5.0.0",
    "mongoose": "^7.0.0",
    "socket.io": "^4.7.0",
    "jsonwebtoken": "^9.0.0",
    "bcryptjs": "^2.4.3",
    "joi": "^17.9.0",
    "nodemailer": "^6.9.0",
    "multer": "^1.4.5",
    "cloudinary": "^1.37.0",
    "express-rate-limit": "^6.7.0",
    "helmet": "^7.0.0",
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "node-cron": "^3.0.0",
    "geolib": "^3.3.4"
  },
  "devDependencies": {
    "@types/node": "^20.0.0",
    "@types/express": "^4.17.0",
    "nodemon": "^2.0.0",
    "ts-node": "^10.9.0"
  }
}
```

Folder Structure

```
backend/
├─ src/
│   ├─ controllers/
│   │   ├─ auth.controller.ts
│   │   ├─ booking.controller.ts
│   │   ├─ workshop.controller.ts
│   │   ├─ user.controller.ts
│   │   └─ admin.controller.ts
│   ├─ services/
│   │   ├─ auth.service.ts
│   │   ├─ booking.service.ts
│   │   ├─ workshop.service.ts
│   │   ├─ notification.service.ts
│   │   ├─ emergency.service.ts
│   │   └─ geolocation.service.ts
│   ├─ models/
│   │   ├─ User.model.ts
│   │   ├─ Workshop.model.ts
│   │   ├─ Booking.model.ts
│   │   └─ Notification.model.ts
│   ├─ middleware/
│   │   ├─ auth.middleware.ts
│   │   ├─ validation.middleware.ts
│   │   ├─ upload.middleware.ts
│   │   └─ rateLimit.middleware.ts
│   ├─ routes/
│   │   ├─ auth.routes.ts
│   │   ├─ booking.routes.ts
│   │   ├─ workshop.routes.ts
│   │   └─ admin.routes.ts
│   ├─ utils/
│   │   ├─ database.ts
│   │   ├─ socketHandlers.ts
│   │   ├─ emailTemplates.ts
│   │   └─ helpers.ts
│   ├─ config/
│   │   ├─ database.config.ts
│   │   ├─ jwt.config.ts
│   │   └─ app.config.ts
│   └─ app.ts
```

└─ package.json
└─ tsconfig.json

Role-Based Access Control (RBAC) Details

1. Guest User

Limited Emergency Access Only

- Can view workshops list without details
- Can access emergency service (limited form)
- Must register to book regular services
- No profile or history access

2. Registered Customer

Full Consumer Access

- Complete service booking system
- Profile and vehicle management
- Booking history and tracking
- Emergency services with full features
- Reviews and ratings system
- Real-time notifications

3. Workshop Owner

Business Management Access

- Workshop profile and service catalog management
- Incoming booking request management
- Staff and technician assignment
- Business analytics and reports
- Customer communication tools
- Service approval/rejection workflow

4. Technician Staff

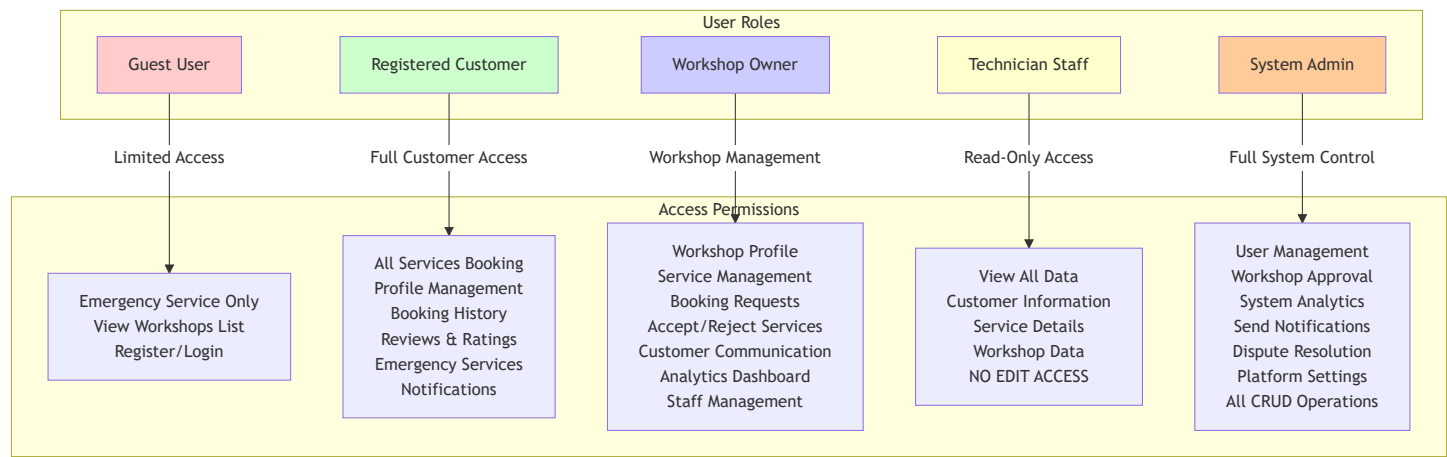
Read-Only System Access

- View all customer information
- Access booking and service details
- Workshop data visibility
- **NO EDIT PERMISSIONS** (Important Security Feature)
- Reports and analytics viewing only

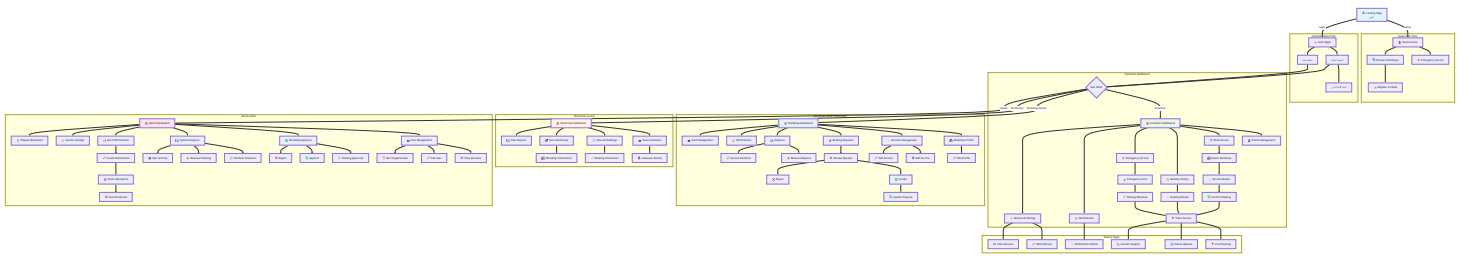
5. System Admin

Full Platform Control

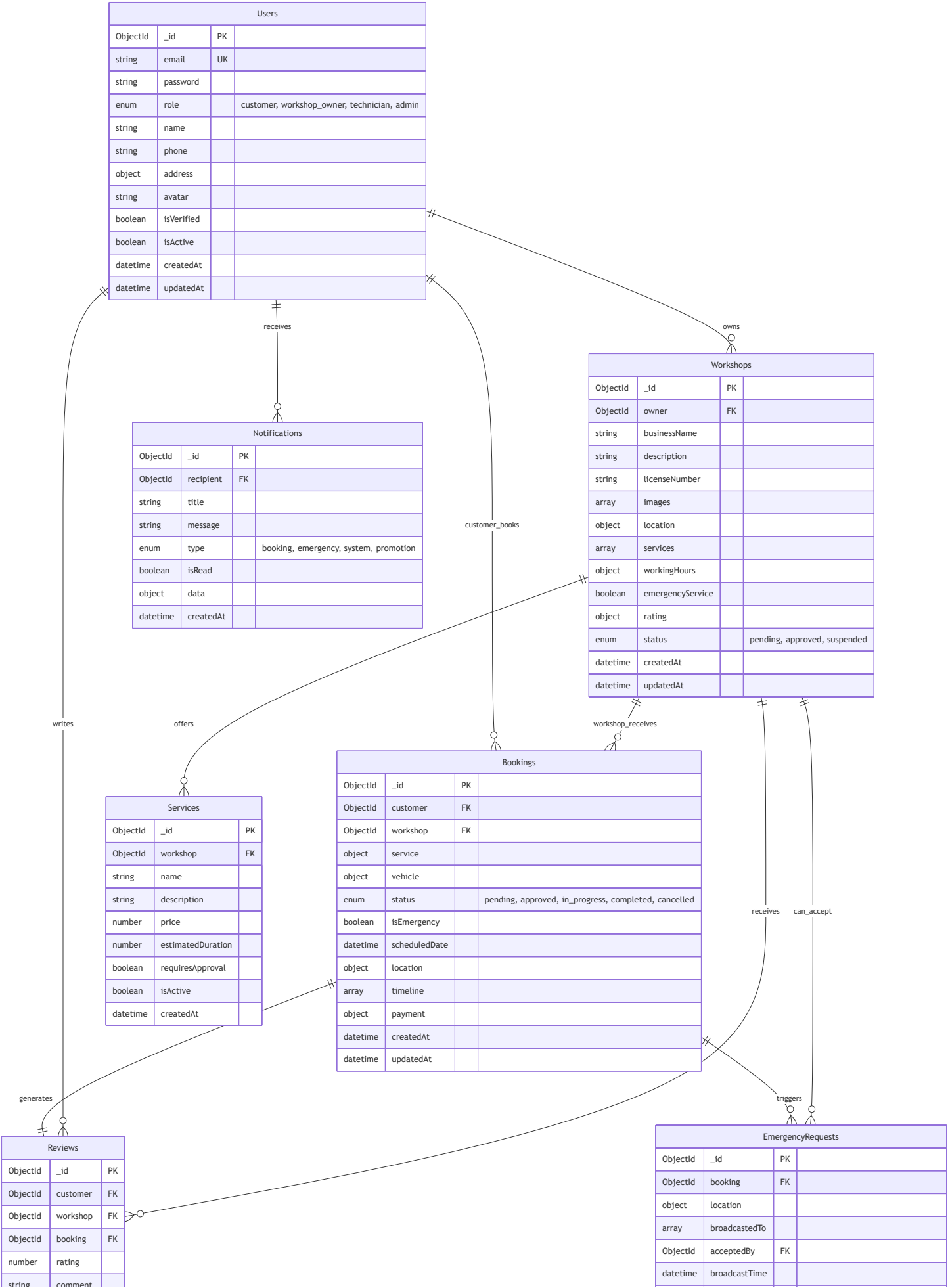
- Complete user management (CRUD)
- Workshop approval workflow
- Platform-wide analytics
- Custom notification broadcasting
- Dispute resolution tools
- System configuration management



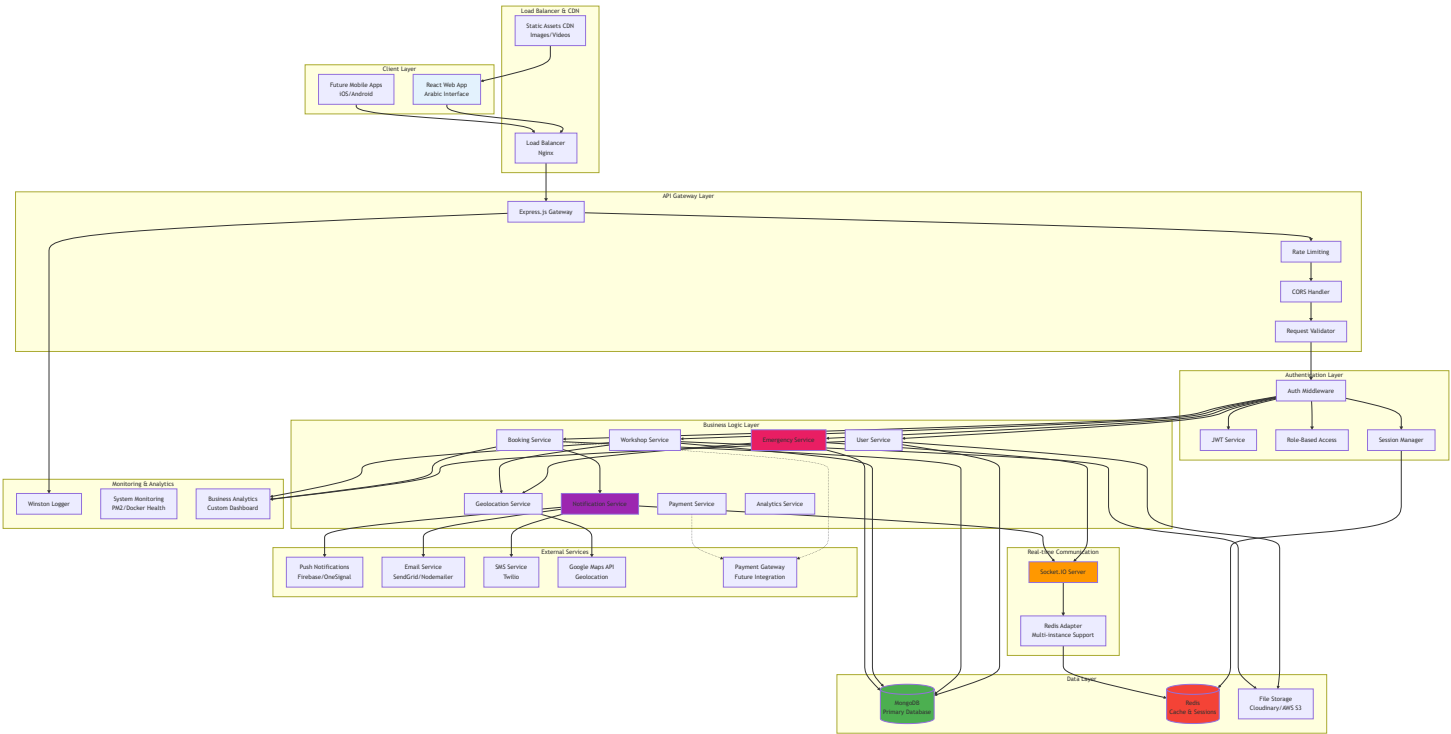
Critical Page Flows



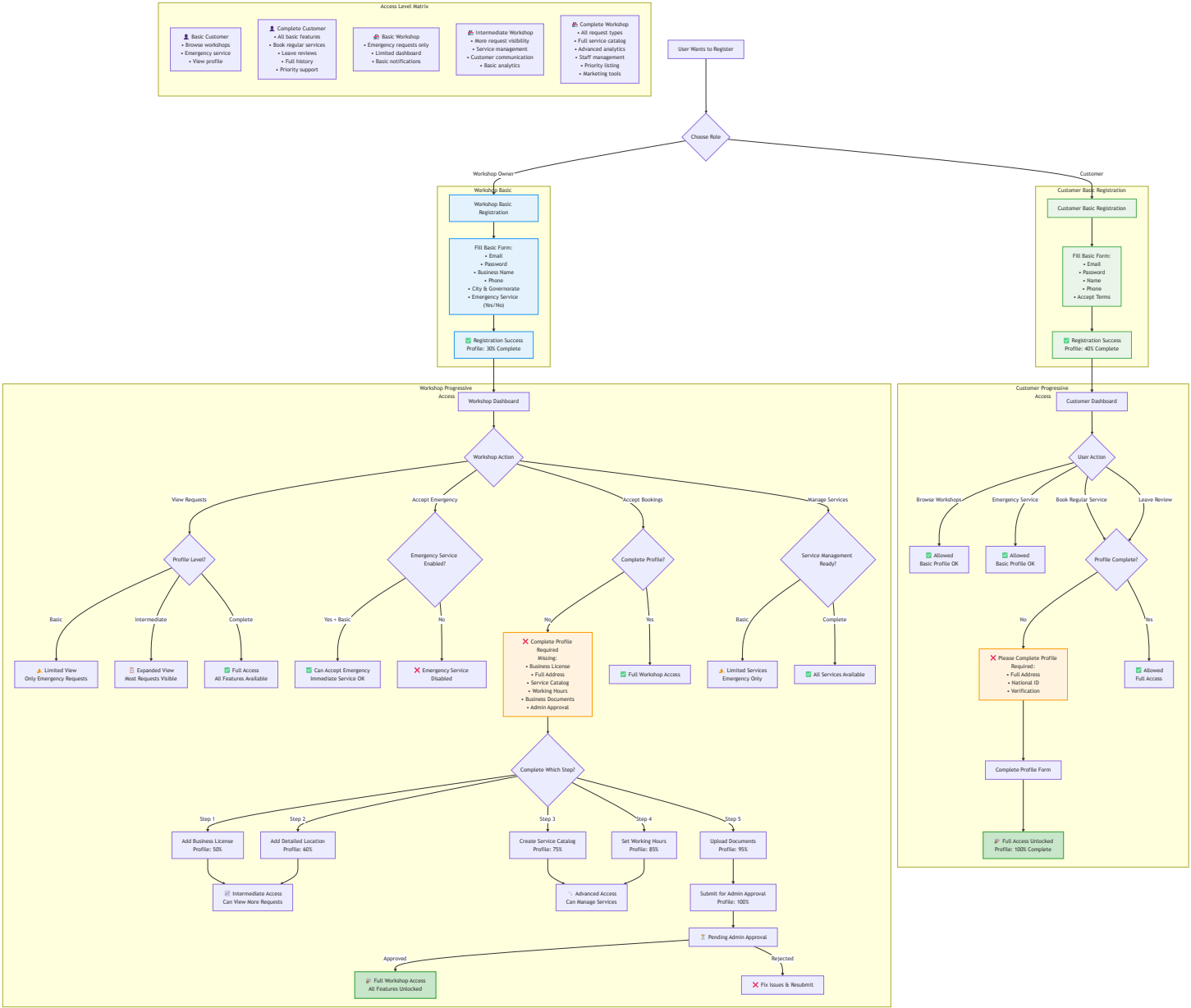
Database Schema & Relationships



Detailed System Architecture



User Register and Login Archticure



API Endpoints Structure

Authentication Endpoints

POST /api/auth/register
POST /api/auth/login
POST /api/auth/logout
POST /api/auth/refresh-token
POST /api/auth/forgot-password
POST /api/auth/reset-password
GET /api/auth/verify-email/:token

User Management

GET /api/users/profile
PUT /api/users/profile
PUT /api/users/change-password
POST /api/users/upload-avatar
DELETE /api/users/account

Workshop Management

GET /api/workshops # Public: Browse workshops
GET /api/workshops/:id # Public: Workshop details
POST /api/workshops # Workshop Owner: Create workshop
PUT /api/workshops/:id # Workshop Owner: Update workshop
DELETE /api/workshops/:id # Workshop Owner: Delete workshop
GET /api/workshops/nearby # Public: Find nearby workshops
POST /api/workshops/:id/services # Workshop Owner: Add service
PUT /api/workshops/:id/services/:serviceId # Workshop Owner: Update service
DELETE /api/workshops/:id/services/:serviceId # Workshop Owner: Delete service

Booking Management

GET	/api/bookings	# Customer: My bookings
POST	/api/bookings	# Customer: Create booking
GET	/api/bookings/:id	# Customer/Workshop: Booking details
PUT	/api/bookings/:id/status	# Workshop: Update booking status
DELETE	/api/bookings/:id	# Customer: Cancel booking
GET	/api/bookings/workshop/:workshopId	# Workshop Owner: Workshop bookings

Emergency Service

POST	/api/emergency/request	# Customer/Guest: Create emergency request
GET	/api/emergency/nearby-workshops	# System: Find emergency providers
POST	/api/emergency/:id/accept	# Workshop: Accept emergency
PUT	/api/emergency/:id/status	# Workshop: Update emergency status
GET	/api/emergency/:id/track	# Customer: Track emergency service

Reviews & Ratings

GET	/api/reviews/workshop/:workshopId	# Public: Workshop reviews
POST	/api/reviews	# Customer: Create review
PUT	/api/reviews/:id	# Customer: Update own review
DELETE	/api/reviews/:id	# Customer: Delete own review
GET	/api/reviews/customer/:customerId	# Customer: My reviews

Notifications

GET	/api/notifications	# User: Get notifications
PUT	/api/notifications/:id/read	# User: Mark as read
PUT	/api/notifications/read-all	# User: Mark all as read
DELETE	/api/notifications/:id	# User: Delete notification
POST	/api/notifications/broadcast	# Admin: Send broadcast notification

Analytics & Reports

GET	/api/analytics/dashboard	# Role-based dashboard data
GET	/api/analytics/bookings	# Booking statistics
GET	/api/analytics/revenue	# Revenue reports (Workshop/Admin)
GET	/api/analytics/users	# User statistics (Admin)
GET	/api/analytics/workshops	# Workshop statistics (Admin)

Admin Management

GET	/api/admin/users	# Admin: List all users
PUT	/api/admin/users/:id/status	# Admin: Ban/activate user
GET	/api/admin/workshops/pending	# Admin: Pending workshop approvals
PUT	/api/admin/workshops/:id/approve	# Admin: Approve workshop
PUT	/api/admin/workshops/:id/reject	# Admin: Reject workshop
GET	/api/admin/system-stats	# Admin: System statistics
POST	/api/admin/notifications	# Admin: Send system notifications

File Upload

POST	/api/upload/image	# Upload single image
POST	/api/upload/images	# Upload multiple images
POST	/api/upload/document	# Upload document
DELETE	/api/upload/:fileId	# Delete uploaded file

Geolocation Services

GET	/api/geo/nearby-workshops	# Find workshops by location
POST	/api/geo/calculate-distance	# Calculate distance between points
GET	/api/geo/geocode	# Convert address to coordinates
GET	/api/geo/reverse-geocode	# Convert coordinates to address

Real-time Socket Events

```
// Customer Events
socket.emit('join_customer', customerId);
socket.on('booking_status_updated', (data) => {});
socket.on('emergency_accepted', (data) => {});
socket.on('technician_location', (data) => {});
socket.on('new_notification', (data) => {});

// Workshop Events
socket.emit('join_workshop', workshopId);
socket.on('emergency_request', (data) => {});
socket.on('new_booking', (data) => {});
socket.on('booking_cancelled', (data) => {});
socket.on('emergency_taken', (data) => {});

// Admin Events
socket.emit('join_admin');
socket.on('new_workshop_registration', (data) => {});
socket.on('system_alert', (data) => {});
```

Middleware Stack

```
// Applied to all routes
app.use(helmet());           // Security headers
app.use(cors());             // CORS handling
app.use(rateLimit());        // Rate limiting
app.use(compression());      // Response compression
app.use(morgan());           // Request logging

// Authentication required
app.use('/api/users', authenticate);
app.use('/api/bookings', authenticate);
app.use('/api/workshops', authenticate); // Except GET requests

// Role-based access
app.use('/api/admin/*', requireRole(['admin']));
app.use('/api/workshops/manage', requireRole(['workshop_owner']));
```

Response Format Standards

```
// Success Response
{
  "success": true,
  "data": { ... },
  "message": "Operation completed successfully",
  "timestamp": "2025-05-29T10:30:00Z"
}
```

```
// Error Response
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid input data",
    "details": { ... }
  },
  "timestamp": "2025-05-29T10:30:00Z"
}
```

```
// Paginated Response
{
  "success": true,
  "data": [...],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 150,
    "totalPages": 8
  }
}
```

Status Codes Used

- 200 - Success
- 201 - Created
- 400 - Bad Request
- 401 - Unauthorized
- 403 - Forbidden
- 404 - Not Found

- 409 - Conflict
- 422 - Validation Error
- 429 - Rate Limit Exceeded
- 500 - Internal Server Error