Remote Pharmacy Medication Tracker

Complete API Documentation & System Specification

SYSTEM OVERVIEW

The Remote Pharmacy Medication Tracker is a full-stack web application that connects customers with remote pharmacies for medication requests and delivery tracking.

Key Features:

- Real-time medication availability checking
- Location-based pharmacy search
- Comprehensive inventory management
- End-to-end order processing and delivery tracking
- Multi-role authentication and authorization
- Automated notifications via SMS/Email
- Integrated payment processing

CORE SYSTEM COMPONENTS

2.1 MEDICATION REQUEST MANAGEMENT (Tharusha)

Overview:

Handles all customer medication requests, tracking their lifecycle from creation through fulfillment or cancellation.

API Endpoints:

Method	Endpoint	Description
POST	/api/requests	Create new medication request
GET	/api/requests	Get all requests with filters (status, urgency, date range)

Method	Endpoint	Description
GET	/api/requests/:id	Get specific request details
PUT	/api/requests/:id	Update request details (quantity, urgency, notes)
DELETE	/api/requests/:id	Cancel pending request
PATCH	/api/requests/:id/status	Update request status (pharmacy admin only)
GET	/api/requests/user/:userId	Get all requests for a specific user
GET	/api/requests/pharmacy/:pharmacyId	Get all requests for a specific pharmacy
GET	/api/requests/urgent	Get all urgent priority requests

Request Object Schema:

```
{
 _id: ObjectId,
 userId: ObjectId (ref: Users),
 pharmacyId: ObjectId (ref: Pharmacies),
 medicationName: String,
 quantity: Number,
 urgencyLevel: String, // "urgent", "normal", "low"
 status: String, // "pending", "processing", "available", "unavailable", "fulfilled", "cancelled"
 prescriptionRequired: Boolean,
 prescriptionImage: String, // URL to uploaded image
 notes: String,
 requestDate: Date,
 responseDate: Date,
 estimatedAvailability: Date,
 createdAt: Date,
 updatedAt: Date
}
```

Key Features:

- Priority-based request handling (Urgent, Normal, Low)
- Prescription image upload support
- Real-time status tracking
- Automated customer notifications
- Request history and analytics
- Bulk request processing

Third-Party Integration:

- **Service:** Twilio SMS API / SendGrid Email API
- **Purpose:** Send real-time notifications about request status updates, availability confirmations, and alerts

Business Logic:

- Validate medication name against inventory
- Check pharmacy service area before accepting requests
- Auto-assign to nearest pharmacy with stock
- Send notifications on status changes
- Generate request analytics

2.2 PHARMACY INVENTORY MANAGEMENT (Lahiru)

Overview:

Comprehensive inventory management system for pharmacies to track medications, stock levels, expiry dates, and automated alerts.

API Endpoints:

Method	**Endpoint**	**Description**
POST	/api/inventory	Add new medication to inventory
GET	/api/inventory	Get all inventory with pagination and filters
GET	/api/inventory/:id	Get specific medication details
PUT	/api/inventory/:id	Update medication details
DELETE	/api/inventory/:id	Remove medication from inventory
GET	/api/inventory/low-stock	Get medications below threshold
PATCH	/api/inventory/:id/stock	Update stock quantity
GET	/api/inventory/expiring	Get medications expiring within 30 days
GET	/api/inventory/search	Search medications by name or category
GET	/api/inventory/pharmacy/:pharmacyId	Get inventory for specific pharmacy
POST	/api/inventory/bulk-upload	Bulk upload medications via CSV
GET	/api/inventory/categories	Get all medication categories

Inventory Object Schema:

```
{
 _id: ObjectId,
 pharmacyId: ObjectId (ref: Pharmacies),
 medicationName: String,
 genericName: String,
 category: String, // "prescription", "otc", "controlled"
 dosage: String, // "500mg", "10ml", etc.
 form: String, // "tablet", "capsule", "syrup", "injection"
 quantity: Number,
 unitPrice: Number,
 batchNumber: String,
 expiryDate: Date,
 manufacturer: String,
 requiresPrescription: Boolean,
 lowStockThreshold: Number,
 storageConditions: String,
 sideEffects: [String],
 contraindications: [String],
 activeIngredients: [String],
 createdAt: Date,
 updatedAt: Date
}
```

Key Features:

- Comprehensive medication categorization
- Batch and expiry date tracking
- Automated low stock alerts
- Drug validation and information lookup
- Advanced search and filtering
- Inventory valuation reports
- Stock movement history

Third-Party Integration:

- **Service:** FDA Drug Database API / RxNorm API
- **Purpose:** Validate medication names, retrieve drug information, check drug interactions, provide standardized medication data

Business Logic:

- Validate medication data against FDA database

- Auto-generate alerts for low stock (< threshold)
- Alert for medications expiring within 30 days
- Calculate inventory value
- Track stock movements (in/out)
- Generate inventory reports

2.3 ORDER & DELIVERY MANAGEMENT (Rajika)

Overview:

End-to-end order processing system that converts approved requests into orders, manages delivery logistics, handles payments, and provides real-time tracking.

API Endpoints:

Method	**Endpoint**	**Description**
POST	/api/orders	Create order from approved request
GET	/api/orders	Get all orders with filters
GET	/api/orders/:id	Get specific order details
PUT	/api/orders/:id	Update order details
PATCH	/api/orders/:id/status	Update order status
GET	/api/orders/track/:id	Real-time delivery tracking
POST	/api/orders/:id/invoice	Generate order invoice
PATCH	/api/orders/:id/assign-delivery	Assign delivery partner
POST	/api/orders/:id/payment	Process payment
GET	/api/orders/user/:userId	Get user's order history
GET	/api/orders/pharmacy/:pharmacyId	Get pharmacy's orders
PATCH	/api/orders/:id/cancel	Cancel order with refund

Method	Endpoint	Description
GET	/api/orders/delivery-partner/:partnerId	Get assigned deliveries

Order Object Schema:

```
{
 _id: ObjectId,
 orderNumber: String, // Unique (e.g., "ORD-2026-001234")
 requestId: ObjectId (ref: Requests),
 userId: ObjectId (ref: Users),
 pharmacyId: ObjectId (ref: Pharmacies),
 items: [{
  medicationId: ObjectId (ref: Inventory),
  name: String,
  quantity: Number,
  unitPrice: Number,
  totalPrice: Number
 }],
 subtotal: Number,
 deliveryFee: Number,
 tax: Number,
 totalAmount: Number,
 deliveryAddress: {
  street: String,
  city: String,
  postalCode: String,
  phoneNumber: String,
  coordinates: {
     latitude: Number,
     longitude: Number
  }
 },
 status: String, // "confirmed", "packed", "out_for_delivery", "delivered", "cancelled"
 deliveryPartnerId: ObjectId (ref: Users),
 estimatedDelivery: Date,
 actualDelivery: Date,
 paymentStatus: String, // "pending", "paid", "failed", "refunded"
 paymentMethod: String, // "card", "cash", "online"
 paymentIntentId: String, // Stripe payment intent
 trackingUpdates: [{
  status: String,
  timestamp: Date,
  location: String,
  notes: String
 }],
 invoiceUrl: String,
 createdAt: Date,
 updatedAt: Date
}
```

Key Features:

- Seamless request-to-order conversion
- Real-time delivery tracking
- Secure payment processing
- Automated invoice generation

- Delivery partner management
- Order status notifications
- Refund processing

Third-Party Integration:

- **Service:** Stripe Payment Gateway / PayPal API
- **Purpose:** Process secure payments, manage transactions, handle refunds, create payment intents

Business Logic:

- Validate inventory before creating order
- Calculate delivery fee based on distance
- Auto-assign nearest available delivery partner
- Update inventory after order confirmation
- Generate unique order numbers
- Send tracking updates to customers
- Process refunds for cancelled orders

2.4 PHARMACY MANAGEMENT (Thilina)

Overview:

Comprehensive pharmacy profile management system with location-based services, verification workflows, and customer review capabilities.

API Endpoints:

Method	Endpoint	Description
POST	/api/pharmacies	Register new pharmacy
GET	/api/pharmacies	Get all pharmacies with filters
GET	/api/pharmacies/:id	Get specific pharmacy details

```
},
```

Method	Endpoint	Description
PUT	/api/pharmacies/:id	Update pharmacy details
DELETE	/api/pharmacies/:id	Deactivate pharmacy
GET	/api/pharmacies/nearby	Find pharmacies by location (lat/lng, radius)
PATCH	/api/pharmacies/:id/verify	Verify pharmacy (admin only)
POST	/api/pharmacies/:id/reviews	Add pharmacy review
GET	/api/pharmacies/:id/reviews	Get pharmacy reviews
GET	/api/pharmacies/search	Search pharmacies by name or location
PATCH	/api/pharmacies/:id/hours	Update operating hours
GET	/api/pharmacies/:id/stats	Get pharmacy statistics

Pharmacy Object Schema:

```
{
  _id: ObjectId,
  name: String,
  licenseNumber: String, // Unique
  location: {
    address: String,
    city: String,
    province: String,
    postalCode: String,
    coordinates: {
      latitude: Number,
      longitude: Number
    }
  },
  contactInfo: {
    phone: String,
    email: String,
    website: String,
    emergencyContact: String
  },
  operatingHours: {
    monday: { open: String, close: String, isClosed: Boolean },
    tuesday: { open: String, close: String, isClosed: Boolean },
    wednesday: { open: String, close: String, isClosed: Boolean },
    thursday: { open: String, close: String, isClosed: Boolean },
    friday: { open: String, close: String, isClosed: Boolean },
    saturday: { open: String, close: String, isClosed: Boolean },
    sunday: { open: String, close: String, isClosed: Boolean }
  },
  serviceRadius: Number, // in kilometers
  isVerified: Boolean,
  verificationDate: Date,
  rating: Number, // Average (1-5)
  totalReviews: Number,
  ownerId: ObjectId (ref: Users),
  facilityType: String, // "retail", "hospital", "clinic"
  services: [String], // ["home_delivery", "24_hours", "emergency", "consultation"]
  images: [String], // URLs
  certifications: [String],
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Pharmacy Reviews Schema:

```
{
  _id: ObjectId,
  pharmacyId: ObjectId (ref: Pharmacies),
  userId: ObjectId (ref: Users),
  rating: Number, // 1-5
  comment: String,
  serviceQuality: Number, // 1-5
  deliverySpeed: Number, // 1-5
  productAvailability: Number, // 1-5
  isVerifiedPurchase: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Key Features:

- Location-based pharmacy discovery
- Pharmacy verification workflow
- Operating hours management
- Customer reviews and ratings
- Service radius definition
- Multi-location support
- Performance analytics

Third-Party Integration:

- **Service:** Google Maps Geocoding & Distance Matrix API
- **Purpose:** Convert addresses to coordinates, calculate distances, validate addresses, provide location-based search

Business Logic:

- Geocode pharmacy addresses
- Calculate service areas
- Find nearest pharmacies
- Validate operating hours
- Calculate average ratings
- Verify pharmacy credentials
- Generate pharmacy analytics

---

3. ADDITIONAL SYSTEM FEATURES

3.1 USER MANAGEMENT & AUTHENTICATION

API Endpoints:

Method	**Endpoint**	**Description**
POST	/api/auth/register	Register new user
POST	/api/auth/login	User login (returns JWT token)
POST	/api/auth/logout	User logout (invalidate token)
POST	/api/auth/refresh	Refresh JWT token
POST	/api/auth/forgot-password	Send password reset email
POST	/api/auth/reset-password	Reset password with token
POST	/api/auth/verify-email	Verify email address
GET	/api/users/profile	Get current user profile

Method	Endpoint	Description
PUT	/api/users/profile	Update user profile
PATCH	/api/users/change-password	Change password
GET	/api/users/:id	Get user by ID (admin only)
GET	/api/users	Get all users (admin only)

User Schema:

```
{
 _id: ObjectId,
 name: String,
 email: String, // Unique
 password: String, // Hashed with bcrypt
 role: String, // "customer", "pharmacy_admin", "delivery_partner", "system_admin"
 phone: String,
 address: {
  street: String,
  city: String,
  postalCode: String,
  coordinates: {
   latitude: Number,
   longitude: Number
  }
 },
 profileImage: String,
 isEmailVerified: Boolean,
 isActive: Boolean,
 lastLogin: Date,
 pharmacyId: ObjectId, // For pharmacy_admin role
 createdAt: Date,
 updatedAt: Date
}
```

Roles & Permissions:

Customer:

- Create/view/update own medication requests
- View pharmacies and reviews
- Place orders
- Track deliveries
- Add reviews

Pharmacy Admin:

- Manage pharmacy profile
- Manage inventory
- Process medication requests
- View/fulfill orders
- View analytics

Delivery Partner:

- View assigned deliveries
- Update delivery status
- Upload delivery proof
- Track earnings

System Admin:

- Full system access
- Verify pharmacies

- Manage users
- View system analytics
- Configure settings

Authentication Features:

- JWT-based authentication
- Password hashing with bcrypt (10 salt rounds)
- Role-based access control (RBAC)
- Token refresh mechanism
- Password reset via email
- Email verification
- Session management

3.2 NOTIFICATIONS & ALERTS SYSTEM

API Endpoints:

Method	Endpoint	Description
GET	/api/notifications	Get user notifications
GET	/api/notifications/unread	Get unread notifications
PATCH	/api/notifications/:id/read	Mark notification as read
PATCH	/api/notifications/read-all	Mark all as read
DELETE	/api/notifications/:id	Delete notification
GET	/api/notifications/preferences	Get notification preferences
PUT	/api/notifications/preferences	Update notification preferences

Notification Schema:

```
{
  _id: ObjectId,
  userId: ObjectId (ref: Users),
  type: String, // "request_update", "order_update", "low_stock", "expiry_alert", "payment"
  title: String,
  message: String,
  relatedEntity: {
   entityType: String, // "request", "order", "inventory"
   entityId: ObjectId
  },
  isRead: Boolean,
  priority: String, // "high", "normal", "low"
  createdAt: Date
}
```

Notification Types:

1. **Request Status Updates:**
   - Request submitted
   - Request processing
   - Medication available/unavailable
   - Request fulfilled
   - Request cancelled
2. **Order Updates:**
   - Order confirmed
   - Payment received
   - Order packed
   - Out for delivery
   - Delivered
   - Order cancelled

3. **Pharmacy Alerts:**
    - Low stock warning
    - Medication expiring soon
    - New request received
    - Verification approved
4. **Payment Notifications:**
    - Payment successful
    - Payment failed
    - Refund processed

Delivery Channels:

- In-app notifications
- Email notifications (SendGrid)
- SMS notifications (Twilio)
- Push notifications (future)

3.3 ANALYTICS & REPORTING

API Endpoints:

Method	Endpoint	Description
GET	/api/analytics/dashboard	Get dashboard overview
GET	/api/analytics/requests	Request analytics by date range
GET	/api/analytics/revenue	Revenue analytics and trends
GET	/api/analytics/popular-medications	Most requested medications
GET	/api/analytics/pharmacy-performance	Pharmacy performance metrics
GET	/api/analytics/delivery-stats	Delivery statistics

Method	Endpoint	Description
GET	/api/analytics/customer-insights	Customer behavior insights
GET	/api/analytics/inventory-reports	Inventory turnover reports

Analytics Features:

For Customers:

- Request history
- Order history
- Spending analytics

For Pharmacies:

- Request fulfillment rates
- Revenue trends
- Popular medications
- Inventory turnover
- Customer ratings
- Peak hours analysis

For System Admins:

- Platform-wide metrics
- User growth
- Transaction volumes
- Geographic distribution
- Performance KPIs

3.4 SEARCH & FILTER SYSTEM

API Endpoints:

Method	Endpoint	Description
GET	/api/search	Global search (medications, pharmacies)
GET	/api/search/medications	Search medications with filters
GET	/api/search/pharmacies	Search pharmacies with filters
GET	/api/search/autocomplete	Autocomplete suggestions

Filter Capabilities:

Medication Filters:

- Category (prescription, OTC, controlled)
- Price range
- Availability status
- Pharmacy location
- Form (tablet, syrup, injection)

Pharmacy Filters:

- Distance/location
- Rating
- Operating hours
- Services offered
- Verification status

Request/Order Filters:

- Status
- Date range
- Urgency level
- Price range
- Pharmacy

---

4. DATABASE SCHEMA SUMMARY

Collections:

1. **Users** - User accounts and authentication
2. **Pharmacies** - Pharmacy profiles and information
3. **Inventory** - Medication inventory items
4. **Requests** - Medication requests from customers
5. **Orders** - Confirmed orders with delivery details
6. **Reviews** - Pharmacy reviews and ratings
7. **Notifications** - User notifications
8. **DeliveryPartners** - Delivery partner profiles (optional separate collection)

Relationships:

Users (1) -----> (N) Requests

Users (1) -----> (N) Orders

Users (1) -----> (1) Pharmacies (for pharmacy_admin)

Pharmacies (1) -----> (N) Inventory

Pharmacies (1) -----> (N) Requests

Pharmacies (1) -----> (N) Orders

Requests (1) -----> (1) Orders

Inventory (1) -----> (N) Order Items

---

5. TECHNOLOGY STACK

Backend:

- **Runtime:** Node.js v18+
- **Framework:** Express.js v4.18+
- **Database:** MongoDB v6+ with Mongoose ODM
- **Authentication:** JWT (jsonwebtoken)

- **Password Hashing:** bcrypt
- **Validation:** express-validator / Joi
- **File Upload:** Multer
- **Email:** Nodemailer with SendGrid
- **SMS:** Twilio SDK
- **Payment:** Stripe SDK
- **Maps:** Google Maps API SDK

Frontend:

- **Library:** React 18+
- **Routing:** React Router v6
- **State Management:** Redux Toolkit / Context API
- **HTTP Client:** Axios
- **UI Framework:** Tailwind CSS / Material-UI
- **Forms:** React Hook Form
- **Maps:** React Google Maps / Leaflet
- **Charts:** Recharts / Chart.js
- **Notifications:** React Toastify

Development Tools:

- **API Testing:** Postman / Thunder Client
- **API Documentation:** Swagger / Postman Documentation
- **Version Control:** Git
- **Code Quality:** ESLint, Prettier
- **Testing:** Jest, Supertest (backend), React Testing Library (frontend)
- **Performance Testing:** Artillery.io

Deployment:

- **Backend:** Render / Railway / Heroku

- **Frontend:** Vercel / Netlify
- **Database:** MongoDB Atlas
- **File Storage:** AWS S3 / Cloudinary (for images)

---

6. THIRD-PARTY INTEGRATIONS SUMMARY

Component	Service	Purpose
Medication Requests	Twilio / SendGrid	SMS & Email notifications
Inventory Management	FDA API / RxNorm	Drug validation & information
Order & Delivery	Stripe / PayPal	Payment processing
Pharmacy Management	Google Maps API	Geocoding & distance calculation

---

7. SECURITY FEATURES

Authentication & Authorization:

- JWT tokens with expiration
- Refresh token rotation
- Password hashing (bcrypt, 10 rounds)
- Role-based access control
- Protected routes middleware

Data Security:

- Input validation and sanitization
- SQL injection prevention (MongoDB)
- XSS protection
- CORS configuration
- Rate limiting
- Helmet.js security headers

API Security:

- HTTPS only
- API key authentication for third-party services
- Environment variables for secrets
- Request size limits
- File upload restrictions

---

8. VALIDATION & ERROR HANDLING

Input Validation:

- Email format validation
- Phone number validation
- Required fields validation
- Data type validation
- String length limits
- Number range validation

Error Response Format:

```
{
  success: false,
  error: {
    code: "VALIDATION_ERROR",
    message: "Invalid input data",
    details: [
      {
        field: "email",
        message: "Invalid email format"
      }
  ]
 }
}
```

HTTP Status Codes:

- 200: Success
- 201: Created
- 400: Bad Request
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 409: Conflict
- 500: Internal Server Error

---

9. API DOCUMENTATION

All APIs must be documented using **Swagger/OpenAPI** or **Postman Collections** including:

- Endpoint descriptions
- Request/response examples
- Authentication requirements
- Query parameters
- Request body schemas
- Response schemas
- Error responses

---

10. TESTING REQUIREMENTS

Unit Testing:

- Test individual controller functions

- Test service layer logic
- Test utility functions
- Test validation schemas
- **Target Coverage:** 70%+

Integration Testing:

- Test API endpoints with Supertest
- Test database operations
- Test authentication middleware
- Test third-party integrations
- **Test Scenarios:** Success, validation errors, authentication errors

Performance Testing:

- Use Artillery.io for load testing
- Test concurrent requests (100+ users)
- Test response times (< 200ms for simple queries)
- Test database query optimization

This completes the comprehensive documentation for the Remote Pharmacy Medication Tracker system. Each team member has clear responsibilities, well-defined APIs, and specific third-party integrations to implement.