✅Complete Implementation Plan for MediChalo MVP (24/7 Medicine Delivery)

Tech Stack

- **Frontend:** React.js
- **Backend:** Node.js, Express.js
- **Database:** MongoDB Atlas
- **Storage:** AWS S3 / Local uploads folder
- **APIs:** Google Cloud Vision API, Google Maps API, Twilio/Fast2SMS, NodeMailer

---

✅Features Implemented

1. **User Authentication** with OTP verification.
2. **Prescription Upload** (image/pdf).
3. **OCR Integration** to extract text from prescription.
4. **Medicine Matching** from extracted text.
5. **Search Nearby Pharmacies** with available stock.
6. **Notify Pharmacies** with available medicines.
7. **Pharmacy Accept/Reject Orders**.
8. **Split Delivery System** for orders across multiple pharmacies.
9. **Route Optimization** using Google Maps API.
10. **OTP Verification** during delivery confirmation.
11. **Secure Storage** of prescription and user data.

✅Folder Structure

```
medichalo-mvp/
├── backend/
│   ├── config/
│   │   ├── db.js
│   │   ├── googleMaps.js
│   │   └── twilio.js
│   ├── controllers/
│   │   ├── authController.js
│   │   ├── medicineController.js
│   │   ├── orderController.js
│   │   ├── pharmacyController.js
│   │   └── prescriptionController.js
│   ├── models/
│   │   ├── user.js
│   │   ├── pharmacy.js
│   │   ├── medicine.js
│   │   ├── order.js
│   │   └── otp.js
```

```
│   ├── routes/
│   │   ├── authRoutes.js
│   │   ├── medicineRoutes.js
│   │   ├── orderRoutes.js
│   │   ├── pharmacyRoutes.js
│   │   └── prescriptionRoutes.js
│   └── server.js
├── frontend/
│   ├── public/
│   │   └── index.html
│   └── src/
│       ├── components/
│       │   ├── SearchBar.jsx
│       │   ├── PrescriptionUpload.jsx
│       │   ├── OTPModal.jsx
│       │   └── OrderSummary.jsx
│       ├── pages/
│       │   ├── Home.jsx
│       │   ├── OrderPage.jsx
│       │   └── PharmacyDetails.jsx
│       ├── utils/
│       │   ├── api.js
│       │   ├── auth.js
│       │   └── location.js
│       ├── App.js
│       └── index.js
├── .env
└── README.md
```

✅Backend Implementation

Database Models

User Model

```
const mongoose = require("mongoose");
const userSchema = new mongoose.Schema({
  name: String,
  phone: String,
  email: String,
  password: String,
  verified: Boolean
});
module.exports = mongoose.model("User", userSchema);
```

Pharmacy Model

```
const mongoose = require("mongoose");
const pharmacySchema = new mongoose.Schema({
  name: String,
  licenseNumber: String,
  location: { type: { type: String, default: "Point" }, coordinates:
[Number] },
  inventory: [{ medicineId: mongoose.Schema.Types.ObjectId, quantity:
Number, price: Number }],
  isVerified: Boolean
});
pharmacySchema.index({ location: "2dsphere" });
module.exports = mongoose.model("Pharmacy", pharmacySchema);
```

Medicine Model

```
const mongoose = require("mongoose");
const medicineSchema = new mongoose.Schema({
  name: String,
  genericName: String,
  description: String,
  alternatives: [{ name: String, pharmacyId: mongoose.Schema.Types.ObjectId,
price: Number }]
});
module.exports = mongoose.model("Medicine", medicineSchema);
```

Order Model

```
const mongoose = require("mongoose");
const orderSchema = new mongoose.Schema({
  patientId: mongoose.Schema.Types.ObjectId,
  splitOrders: [{ pharmacyId: mongoose.Schema.Types.ObjectId, medicines: [{
medicineId: mongoose.Schema.Types.ObjectId, quantity: Number, price:
Number }], status: String }],
  assignedDeliveryPartner: mongoose.Schema.Types.ObjectId,
  status: String,
  prescription: String,
  deliveryOtp: Number,
  createdAt: { type: Date, default: Date.now }
});
module.exports = mongoose.model("Order", orderSchema);
```

OTP Model

```
const mongoose = require("mongoose");
const otpSchema = new mongoose.Schema({
```

```
  phone: String,
  otp: Number,
  expiresAt: Date
});
module.exports = mongoose.model("OTP", otpSchema);
```

Key API Endpoints

Authentication (`authRoutes.js`)

- `/signup` → register and generate OTP
- `/verify-otp` → verify OTP and log in

Prescription Upload (`prescriptionRoutes.js`)

- `/upload` → upload file, extract text, find medicines, notify pharmacies

Medicine Search (`medicineRoutes.js`)

- `/search` → search by name
- `/generic-options` → find generic alternatives

Pharmacy (`pharmacyRoutes.js`)

- `/nearby` → find pharmacies by location and available stock
- `/accept-order` → accept order requests

Orders (`orderRoutes.js`)

- `/create-order` → create split orders
- `/confirm-delivery` → confirm via OTP
- `/route` → optimized delivery route

Prescription Upload Flow Example (`prescriptionController.js`)

```
const multer = require("multer");
const vision = require("@google-cloud/vision");
const Pharmacy = require("../models/pharmacy");
const Medicine = require("../models/medicine");
const Order = require("../models/order");

// Upload setup using multer omitted for brevity

const client = new vision.ImageAnnotatorClient({ keyFilename: "path-to-
json" });

async function uploadPrescription(req, res) {
```

```
  const filePath = req.file.path;
  const orderId = req.body.orderId;

  // OCR extract
  const [result] = await client.textDetection(filePath);
  const text = result.textAnnotations[0].description.toLowerCase();

  // Find medicines
  const medicines = await Medicine.find({});
  const matched = medicines.filter(m => text.includes(m.name.toLowerCase()));

  // Find pharmacies with stock
  const pharmacyIds = await Pharmacy.find({
    'inventory.medicineId': { $in: matched.map(m => m._id) }
  }).select("_id");

  // Notify pharmacies (email/sms omitted for brevity)
  res.json({ matchedMedicines: matched, pharmacies: pharmacyIds });
}

module.exports = { uploadPrescription };
```

✅Frontend Prescription Upload Component (`PrescriptionUpload.jsx`)

```
function PrescriptionUpload({ orderId }) {
  const [file, setFile] = useState(null);

  const handleUpload = async () => {
    const formData = new FormData();
    formData.append("prescription", file);
    formData.append("orderId", orderId);
    await fetch("/api/upload-prescription", { method: "POST", body:
formData });
  };

  return (
    <div>
      <input type="file" accept="image/*,application/pdf" onChange={e =>
setFile(e.target.files[0])} />
      <button onClick={handleUpload}>Upload</button>
    </div>
  );
}
```

✅Notifications Setup

1. **Email using NodeMailer**
2. **SMS using Twilio**
3. **In-app notifications stored in MongoDB**

✅Route Optimization

- Use Google Maps API to calculate distances and order routes.
- Pass patient and pharmacy coordinates to the API for optimized paths.

✅Security Measures

✔️Validate inputs
✔️Encrypt passwords
✔️Limit file uploads
✔️Authenticate requests
✔️Store sensitive data securely

✅Final Notes

This structure covers the full flow from prescription upload, OCR extraction, medicine matching, pharmacy notification, order acceptance, split delivery, and OTP verification.