MODERN FRONTEND UI - LEGAL BOOKING PLATFORM

PROJECT CONTEXT

You are building a modern, production-ready frontend for a Legal Booking SaaS platform. The backend is already deployed and functional with all APIs ready. Your task is to create a complete, beautiful, and highly functional UI using Next.js 14, TypeScript, Tailwind CSS, and shadcn/ui components.

Existing Setup:

- Next.js 14 (App Router)
- TypeScript
- Tailwind CSS
- shadcn/ui components (already configured)
- Lucide React icons
- Axios for API calls

Backend API Base URL: `https://your-backend-url.com/api/v1`

DESIGN PHILOSOPHY

Visual Style: Modern SaaS Platform

- **Clean & Professional**: Legal services require trust - use clean layouts, ample whitespace
- **Modern Gradients**: Subtle gradients for hero sections and cards
- **Blue/Purple Color Scheme**: Primary blue (`#3B82F6`), accent purple (`#9333EA`)
- **Micro-interactions**: Smooth hover effects, transitions, loading states
- **Mobile-First**: Fully responsive, works perfectly on all devices
- **Accessibility**: WCAG 2.1 AA compliant

UI Patterns to Follow:

- **Airbnb-style search**: Location filters with cascading dropdowns
- **Stripe-style forms**: Clean, labeled inputs with validation feedback
- **Linear-style navigation**: Minimal, elegant navbar
- **Notion-style cards**: Subtle shadows, hover effects
- **Figma-style modals**: Centered, backdrop blur, smooth animations

COMPLETE PAGE STRUCTURE

1. PUBLIC PAGES (No Authentication Required)

1.1 Homepage (`/app/page.tsx`) - ALREADY EXISTS, ENHANCE IT

Current State: Basic hero + features section exists

Enhancements Needed:

```typescript
```

// Add these sections:

1. **Dynamic Hero with Search**
   - Animated gradient background
   - Search bar with auto-complete (specializations)
   - Real-time stats from backend API (/api/stats)
   - CTA buttons: "Find Lawyer" and "Register as Lawyer"

2. **How It Works Section**
   - 4-step visual process (cards with numbers)
   - Icons: Search → Book → Consult → Review
   - Animated on scroll (use Intersection Observer)

3. **Featured Lawyers Carousel**
   - Fetch top 6 lawyers from API (/api/lawyers/featured)
   - Horizontal scroll on mobile, grid on desktop
   - Lawyer cards: Photo, name, specialization, rating, price
   - "View All" button → /search

4. **Specializations Grid**
   - 8 main specializations (Criminal, Civil, Family, etc.)
   - Icon + name + lawyer count
   - Click → /search?specialization_id=xxx

5. **Testimonials Section**
   - 3 testimonial cards
   - Client photo (anonymized), review, rating
   - Carousel on mobile

6. **Stats Bar**
   - 4 metrics in a row: Verified Lawyers | Happy Clients | Consultations | Average Rating
   - Animated counter (count-up effect)

7. **Lawyer CTA Section** (Already exists, keep it)

8. **FAQ Accordion**
   - 8-10 common questions
   - Smooth expand/collapse animation

9. **Footer** (Already exists, expand it)
   - Add: About, Contact, Privacy Policy, Terms

- Social media links
- Newsletter signup

Key Interactions:

- Search bar → Navigate to `/search` with query params
- Specialization cards → Navigate to `/search?specialization_id=xxx`
- Featured lawyers → Navigate to `/lawyers/[id]`

1.2 Lawyer Search Page (`/app/search/page.tsx`) - CREATE NEW

Layout:

```
┌─────────────────────────────────────────────────┐
│ Filters Sidebar  │ Results Grid                 │
│ (1/4 width)      │ (3/4 width)                  │
└─────────────────────────────────────────────────┘
```

Left Sidebar Filters:

```typescript
interface FilterState {
  state_id: string | null;
  district_id: string | null;
  court_id: string | null;
  police_station_id: string | null;
  specialization_id: string | null;
  sub_specialization_id: string | null;
  min_price: number | null;
  max_price: number | null;
  min_experience: number | null;
  min_rating: number | null;
  languages: string[];
  sort_by: 'price_asc' | 'price_desc' | 'rating_desc' | 'experience_desc';
}
```

Filter Components:

1. **Location Cascade**
   - State dropdown (fetch from `/api/locations/states`)

- District dropdown (loads after state selected)
- Court/Police Station dropdown (toggle between both)

2. **Specialization Cascade**
   - Main specialization dropdown
   - Sub-specialization dropdown (loads after main selected)
3. **Price Range Slider**
   - Dual-handle slider (₹500 - ₹100,000)
   - Display current range
4. **Experience Filter**
   - Checkbox: 0-5 years, 5-10 years, 10-15 years, 15+ years
5. **Rating Filter**
   - Star buttons: All, 4+, 3+
6. **Languages**
   - Multi-select checkboxes: English, Hindi, Bengali, Tamil, etc.
7. **Sort By**
   - Dropdown: Price (Low-High), Price (High-Low), Highest Rated, Most Experienced
8. **Clear All Button**

Right Side Results:

typescript

```
// Lawyer Card Component
<LawyerCard>
  - Profile photo (left)
  - Name, specializations
  - Rating (stars) + review count
  - Experience (years)
  - Languages
  - Courts (first 2, then "+3 more")
  - Price (₹X/consultation)
  - "View Profile" button
</LawyerCard>
```

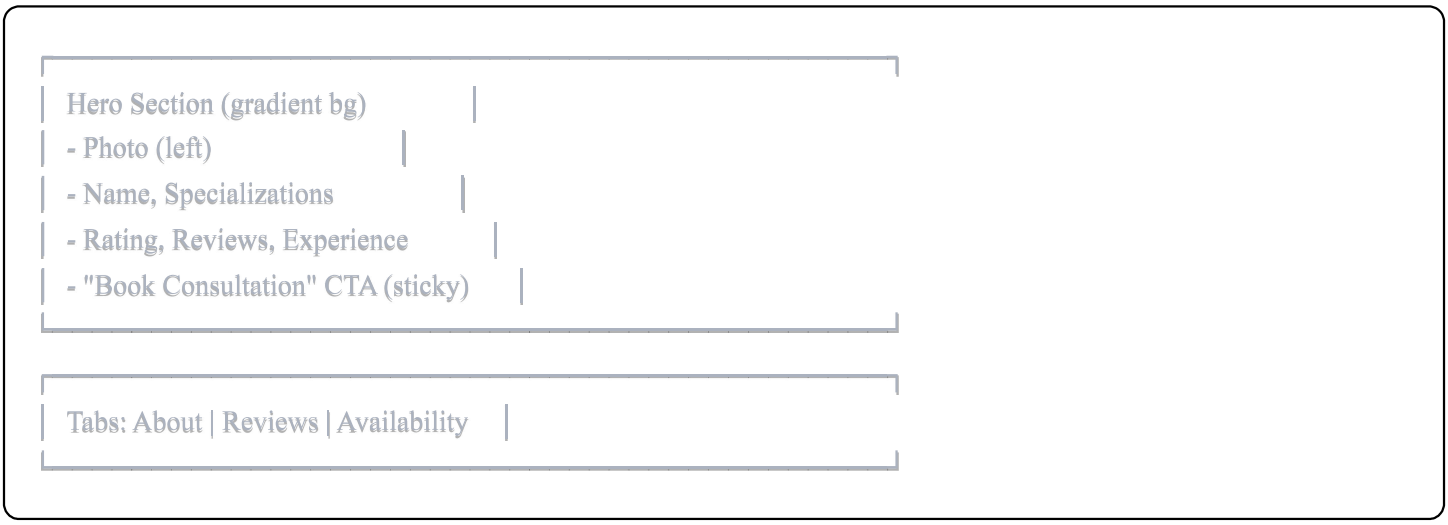
Additional Features:

- Loading skeleton while fetching

- Pagination at bottom (10 per page)
- "No results" state with illustration
- Sticky filter sidebar on desktop
- Filter drawer on mobile (slide from left)

1.3 Lawyer Profile Page (`/app/lawyers/[id]/page.tsx`) - CREATE NEW

Layout:



Hero Section:

- Large profile photo (200x200, rounded)
- Name (h1)
- Specializations (badges)
- Bar Council number (masked: ABC***123)
- Experience (e.g., "8 years practicing")
- Rating (4.5 ★ with 120 reviews)
- Languages spoken (badges)
- Price (₹3,000/consultation, prominent)
- **Book Consultation** button (sticky on mobile)

About Tab:

- Bio (formatted text)
- Education
- Courts practicing in (list with icons)

- Specializations & sub-specializations (visual tags)

Reviews Tab:

- Review cards:
  - Anonymous user name (J***n D***)
  - Rating (stars)
  - Comment
  - Date
- Pagination (10 per page)
- Average rating breakdown (5-star, 4-star, 3-star, etc. with progress bars)

Availability Tab: (For future - show "Currently accepting bookings" message)

Booking Action:

- Click "Book Consultation" → If not logged in, redirect to `/auth/login?redirect=/booking/create?lawyer_id=xxx`
- If logged in → Navigate to `/booking/create?lawyer_id=xxx`

---

2. AUTHENTICATION PAGES

2.1 User Registration (`/app/auth/register/page.tsx`) - CREATE NEW

Design:

- Centered card (max-w-md)
- Logo at top
- Form fields:
  - Full Name
  - Email
  - Phone (+91 prefix, Indian format validation)
  - Password (with strength indicator)
  - User Type: Radio buttons (User / Lawyer)
- Terms checkbox
- "Register" button (full width, loading state)
- Already have account? Login link

After Submission:

- Show OTP verification screen (modal or new page)
- 2 OTP inputs (Email OTP, Phone OTP)
- Resend OTP buttons (with countdown timer)
- Verify button

API Calls:

```typescript
POST /api/v1/auth/register
POST /api/v1/auth/verify-otp
```

2.2 Login Page (`/app/auth/login/page.tsx`) - CREATE NEW

Design:

- Centered card
- Email/Phone input (accepts both)
- Password input (with show/hide toggle)
- "Forgot Password?" link
- "Login" button (full width, loading state)
- Don't have account? Register link

After Login:

- Store tokens in localStorage
- Redirect based on user_type:
  - User → `/dashboard`
  - Lawyer → `/lawyer/dashboard`
  - Admin → `/admin/dashboard`

2.3 Lawyer Registration (`/app/auth/lawyer-register/page.tsx`) - CREATE NEW

Multi-Step Form (Wizard):

Step 1: Basic Account

- Name, Email, Phone, Password (same as user registration)
- "Continue" button

Step 2: Professional Details

- Bar Council Registration Number
- Years of Experience (number input)
- Education (textarea)
- Bio (textarea, max 1000 chars)
- Languages (multi-select)
- "Continue" button

Step 3: Practice Details

- Courts (multi-select dropdown with search)
  - Shows: State → District → Courts
  - Selected courts show as removable badges
- Specializations (multi-select)
  - Main specialization → Sub-specialization
  - Shows as tree checkboxes
- Consultation Fee (₹500 - ₹100,000)
- "Continue" button

Step 4: Documents

- File uploads:
  - Bar Council Certificate (PDF/JPG/PNG, max 5MB)
  - ID Proof - Aadhaar/PAN (PDF/JPG/PNG, max 5MB)
  - Profile Photo (JPG/PNG, max 2MB, optional)
- Preview uploaded files
- "Submit for Verification" button

After Submission:

- Success page: "Your application is under review"

- "We'll verify your documents within 3-5 days"
- "Check your email for updates"
- "Go to Homepage" button

API Calls:

```typescript
POST /api/v1/auth/register (Step 1)
POST /api/v1/lawyers/register (Steps 2-4 combined)
```

---

3. USER DASHBOARD PAGES

3.1 User Dashboard (`/app/dashboard/page.tsx`) - CREATE NEW

Layout:

```
┌─────────────────────────────────────────────────┐
│ Sidebar (nav) │ Main Content                    │
└─────────────────────────────────────────────────┘
```

Sidebar Navigation:

- Dashboard (home icon)
- My Bookings (calendar icon)
- AI Assistant (bot icon)
- Profile (user icon)
- Logout (logout icon)

Main Content - Overview:

- Welcome message: "Welcome back, [Name]!"
- Stats cards (grid):
  - Total Bookings
  - Upcoming Consultations
  - Completed Consultations
  - Pending Reviews
- **Upcoming Consultations Section:**

- Cards showing next 3 bookings
- Each card: Lawyer name, photo, date/time, specialization
- "Join Consultation" button (if within 10 min)
- "View Details" link

- **Recent Activity:**
  - Timeline/list of recent actions
  - "Booking confirmed", "Payment successful", etc.

3.2 My Bookings (`/app/dashboard/bookings/page.tsx`) - CREATE NEW

Tabs:

- All
- Pending (waiting for lawyer response)
- Accepted (upcoming consultations)
- Completed
- Cancelled/Refunded

Booking Card:

```
┌─────────────────────────────────────────┐
│ [Lawyer Photo] [Name]                   │
│ Specialization                          │
│ Status Badge (color-coded)              │
│ Date & Time (if scheduled)              │
│ Amount: ₹X                              │
│ [View Details] [Action Button]          │
└─────────────────────────────────────────┘
```

Action Buttons (based on status):

- Pending: "Cancel Booking"
- Accepted: "Join Consultation" (if time is near)
- Completed: "Write Review"

Click "View Details":

- Navigate to `/dashboard/bookings/[id]`

3.3 Booking Details Page (`/app/dashboard/bookings/[id]/page.tsx`) - CREATE NEW

Layout:

- Booking ID, Status badge (top)
- Timeline (visual):
  - Booking Created ✓
  - Payment Successful ✓
  - Lawyer Accepted (if accepted)
  - Consultation Scheduled (if scheduled)
  - Consultation Completed (if completed)
  - Review Submitted (if review done)

Details Section:

- Lawyer Details (photo, name, specialization)
- Your Case Description (original text)
- AI Summary (shown to lawyer)
- Scheduled Time (if accepted)
- Amount Paid: ₹X

Actions:

- If status = pending: "Cancel & Get Refund" button
- If status = accepted: "Join Consultation" button (if within 10 min)
- If status = completed & no review: "Write Review" button
- "Download Receipt" button

Chat Section (if consultation started):

- Show in-app chat messages
- Input to send new message

3.4 Booking Creation (`/app/booking/create/page.tsx`) - CREATE NEW

Pre-requisites: Must be logged in, lawyer_id in query params

Step 1: Case Description

```
┌─────────────────────────────────────────┐
│ Lawyer Info Card (top)                  │
│ - Photo, Name, Specialization           │
│ - Fee: ₹X                               │
└─────────────────────────────────────────┘

Describe your case (10-200 characters):
[Textarea - character counter]

Select Court or Police Station:
○ Court [Dropdown]
○ Police Station [Dropdown]

[Continue Button]
```

Step 2: AI Summary Preview

Your Description:

[Original text displayed]

AI-Generated Summary:

[AI summary displayed]

⚠️ This summary will be sent to the lawyer.
Not accurate? Edit your description:

[Editable Textarea]
[Regenerate Summary Button]

Consultation Fee: ₹3,000

[Confirm & Pay Button]

Step 3: Payment (Razorpay Modal)

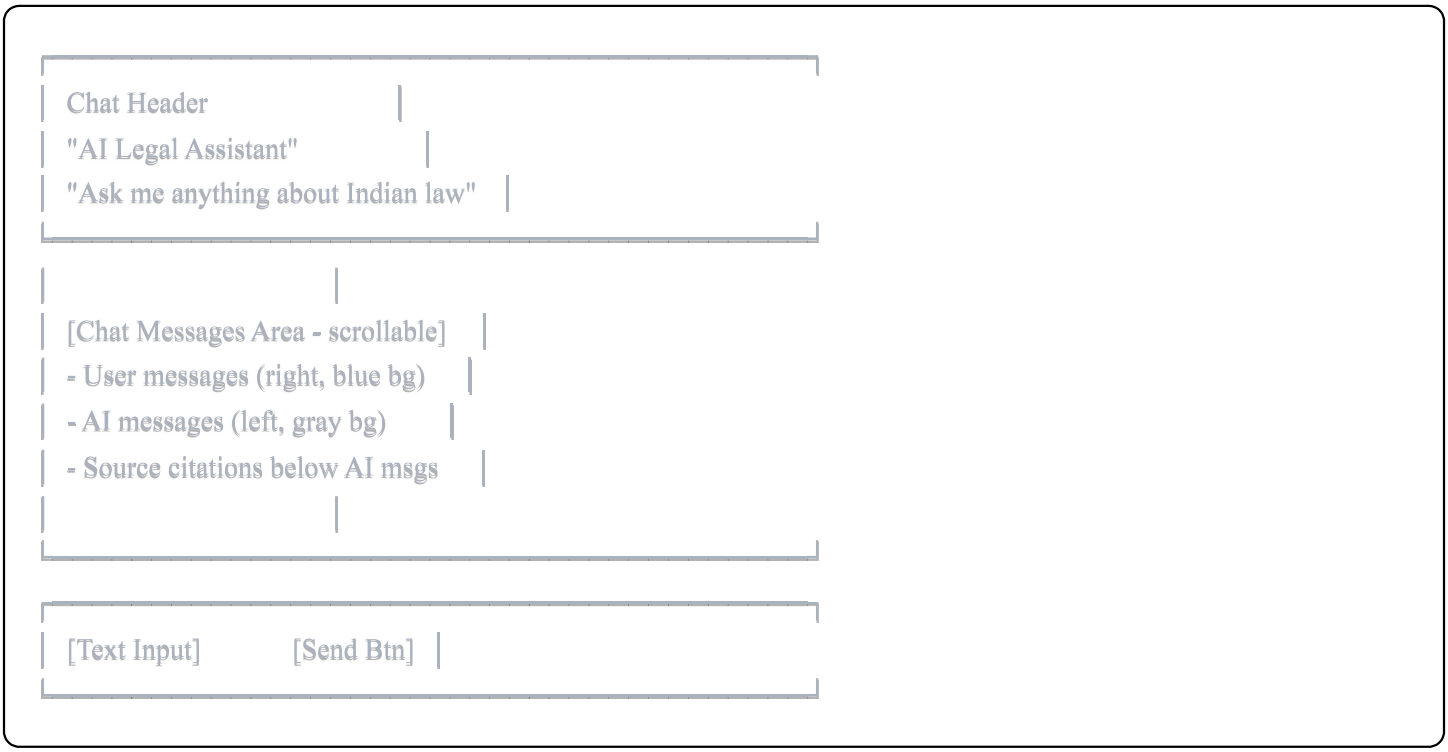
- Razorpay checkout opens
- After successful payment → Redirect to `/dashboard/bookings/[id]`

API Calls:

```typescript
POST /api/v1/bookings/create (Step 1 - generates AI summary)
POST /api/v1/bookings/regenerate-summary (if user edits)
POST /api/v1/bookings/confirm (Step 2 - creates order)
POST /api/v1/payments/verify (after Razorpay callback)
```

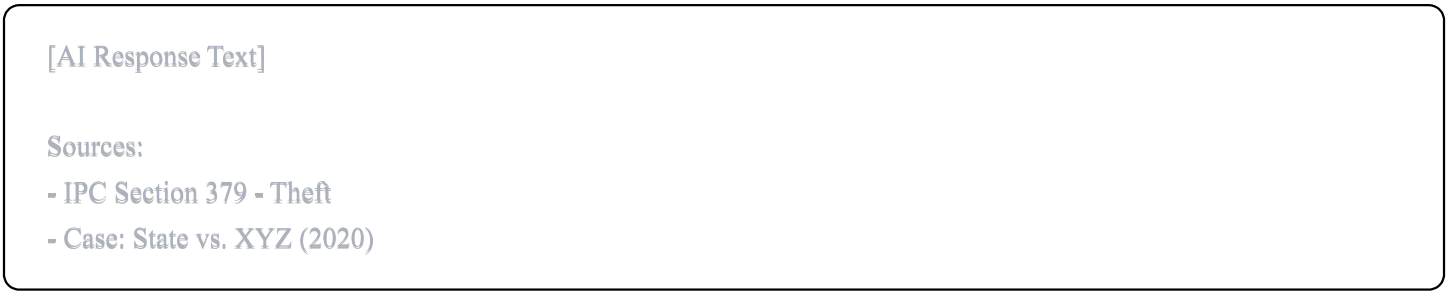
3.5 AI Legal Assistant (`/app/dashboard/ai-assistant/page.tsx`) - CREATE NEW

Design: ChatGPT-style Interface



Features:

- Pre-filled suggested questions (pills):
  - "How to file an FIR?"
  - "What is Section 302 IPC?"
  - "Divorce procedure in India"
  - "Property dispute resolution"
- AI response with citations:



- **Disclaimer** (always shown):
  - "⚠️ This is general information, not legal advice. For your specific case, book a consultation."
- **Smart CTA:** If AI detects user needs lawyer:

API Call:

```
typescript

POST /api/v1/ai/chat
Body: { message: "user query", context: { user_type: "user" } }
```

---

4. LAWYER DASHBOARD PAGES

4.1 Lawyer Dashboard (`/app/lawyer/dashboard/page.tsx`) - CREATE NEW

Layout: Same sidebar structure as user dashboard

Sidebar Navigation:

- Dashboard
- Booking Requests (with badge showing count)
- My Consultations
- Earnings
- Profile
- AI Assistant
- Logout

Main Content - Overview:

- Welcome message
- Verification status banner (if pending):

```
⏳ Your application is under review
We'll verify your documents within 3-5 days.
```

- Stats cards:
  - Pending Requests
  - Upcoming Consultations
  - Total Earnings (this month)
  - Average Rating
- **Pending Booking Requests:**

- List of pending bookings (cards)
- Quick actions: Accept, Reject, Reschedule

- **Today's Consultations:**
  - List of today's scheduled consultations

4.2 Booking Requests (`/app/lawyer/bookings/page.tsx`) - CREATE NEW

Tabs:

- Pending Requests
- Accepted
- Completed
- Rejected

Pending Request Card:

```
┌─────────────────────────────────────────┐
│ User: J***n D*** (anonymized)           │
│ Specialization: Criminal Law            │
│ Requested: 2 hours ago                  │
│                                         │
│ Case Description:                       │
│ [Original 200-char text]                │
│                                         │
│ AI Summary:                             │
│ [AI-generated summary]                  │
│                                         │
│ Court: Kolkata District Court           │
│ Fee: ₹3,000                             │
│                                         │
│ [Accept] [Reject] [Reschedule]          │
└─────────────────────────────────────────┘
```

Accept Modal:

- Date picker (calendar)
- Time picker (dropdown: 9 AM - 6 PM)
- "Confirm" button
- API: `POST /api/v1/lawyers/bookings/[id]/respond`

Reject Modal:

- Reason textarea (required)
- "Confirm Rejection" button
- Message: "User will receive full refund"

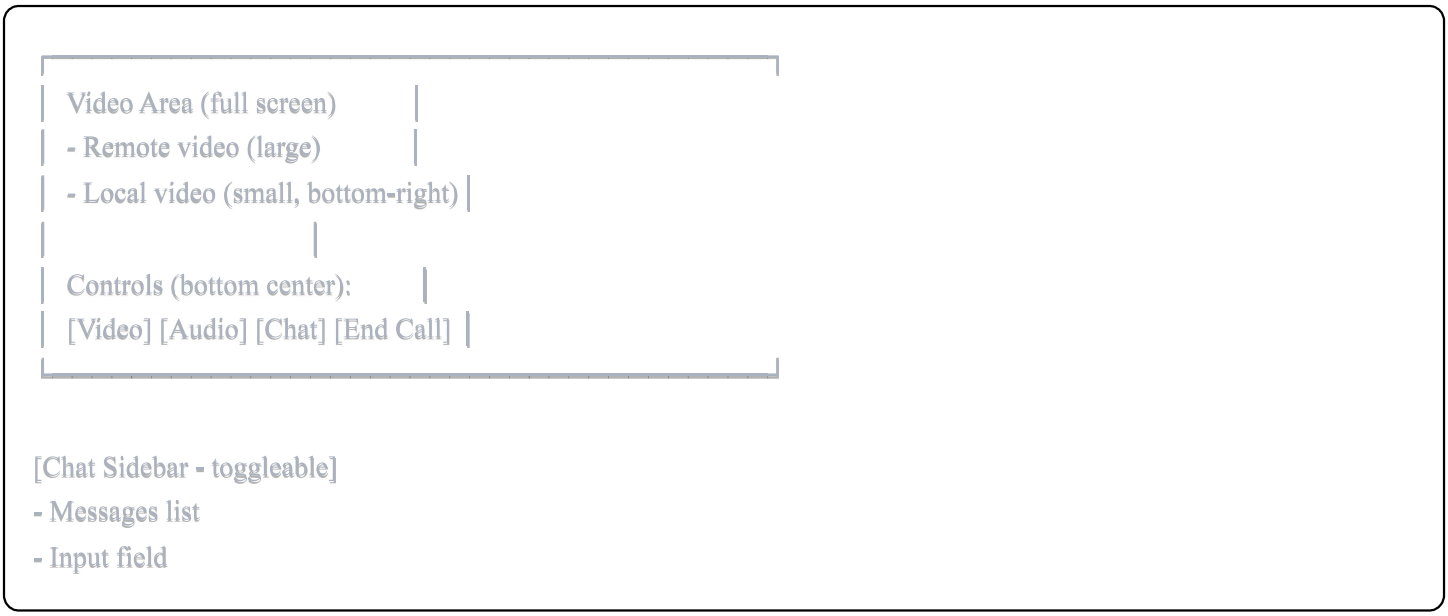
Reschedule Modal:

- Same as Accept modal
- Note: "Max 2 reschedules allowed"

4.3 Consultation Room (`/app/consultation/[booking_id]/page.tsx`) - CREATE NEW

Only accessible when consultation starts (within 10 min of scheduled time)

Layout:



Video Controls:

- Video toggle (camera on/off)
- Audio toggle (mic on/off)
- Chat toggle (show/hide sidebar)
- End Call (red button, confirmation modal)

Chat Sidebar:

- Same as booking details chat
- Real-time messaging

End Call:

- Confirmation modal: "Are you sure? This will mark the consultation as complete."
- On confirm:
  - Stop Agora recording
  - Update booking status
  - Redirect to booking details

Implementation:

```typescript
// Use Agora SDK
import AgoraRTC from 'agora-rtc-sdk-ng';

// Get credentials from API
const { agora_token, channel_name, app_id } = await fetchConsultationCreds(booking_id);

// Initialize Agora client
const client = AgoraRTC.createClient({ mode: 'rtc', codec: 'vp8' });
await client.join(app_id, channel_name, agora_token, 0);

// Create local tracks
const [audioTrack, videoTrack] = await AgoraRTC.createMicrophoneAndCameraTracks();
await client.publish([audioTrack, videoTrack]);

// Handle remote users
client.on('user-published', async (user, mediaType) => {
  await client.subscribe(user, mediaType);
  if (mediaType === 'video') {
    user.videoTrack.play('remote-video-container');
  }
});
```

4.4 Earnings Page (`/app/lawyer/earnings/page.tsx`) - CREATE NEW

Layout:

- Total Earnings (big number at top)
- This Month Earnings (card)
- Earnings chart (line graph - monthly breakdown)
- Recent Payouts Table:
    - Date, Booking ID, Amount, Status, Receipt

---

5. ADMIN PAGES

5.1 Admin Dashboard (`/app/admin/dashboard/page.tsx`) - CREATE NEW

Protected Route: Only accessible by admin users

Stats Overview:

- Total Users
- Total Lawyers
- Pending Verifications (clickable → verification queue)
- Total Bookings
- Total Revenue (commission)

Charts:

- User growth (line chart)
- Bookings by specialization (pie chart)
- Revenue by month (bar chart)

Recent Activity Feed:

- New user registrations
- New lawyer applications
- Bookings created
- Payments processed

5.2 Lawyer Verification Queue (`/app/admin/lawyers/pending/page.tsx`) - CREATE NEW

List of Pending Lawyers:

Each row:

- Lawyer name, email, phone
- Bar Council number
- Applied date
- "Review" button

Click "Review" → Detail Modal:

- Full lawyer details
- View uploaded documents:
  - Bar Council Certificate (open in new tab)
  - ID Proof (open in new tab)
  - Profile Photo
- Verification actions:
  - "Approve" button (green)
  - "Reject" button (red, requires reason)

API Calls:

```typescript
GET /api/v1/admin/lawyers/pending
POST /api/v1/admin/lawyers/[id]/verify
Body: { action: "approve" | "reject", rejection_reason?: string }
```

SHARED COMPONENTS TO CREATE

Component Library (`/components/`)

```
/components
├── /ui (shadcn components - already exist)
├── /shared
│   ├── Header.tsx
│   ├── Footer.tsx
│   ├── Sidebar.tsx
│   ├── PageLoader.tsx
│   ├── EmptyState.tsx
```

```
│   └── ProtectedRoute.tsx
├── /booking
│   ├── BookingCard.tsx
│   ├── BookingStatusBadge.tsx
│   └── BookingTimeline.tsx
├── /lawyer
│   ├── LawyerCard.tsx
│   ├── LawyerAvatar.tsx
│   ├── SpecializationBadge.tsx
│   └── RatingDisplay.tsx
├── /search
│   ├── SearchFilters.tsx
│   ├── LocationCascade.tsx
│   ├── SpecializationCascade.tsx
│   └── PriceRangeSlider.tsx
├── /consultation
│   ├── VideoPlayer.tsx
│   ├── ChatPanel.tsx
│   └── VideoControls.tsx
└── /ai
    ├── ChatMessage.tsx
    ├── ChatInput.tsx
    └── SourceCitation.tsx
```

Key Shared Components

Header.tsx

typescript

```
'use client';

export default function Header() {
  const { user, logout } = useAuth(); // Custom hook

  return (
    <nav className="sticky top-0 z-50 border-b bg-white/95 backdrop-blur">
      <div className="container mx-auto px-4 h-16 flex items-center justify-between">
        <Link href="/">
          <Logo />
        </Link>

        <div className="flex items-center gap-4">
          {user ? (
            <>
              <NotificationBell />
              <UserMenu user={user} onLogout={logout} />
            </>
          ) : (
            <>
              <Link href="/search">
                <Button variant="ghost">Find Lawyers</Button>
              </Link>
              <Link href="/auth/login">
                <Button variant="outline">Login</Button>
              </Link>
              <Link href="/auth/register">
                <Button>Get Started</Button>
              </Link>
            </>
          )}
        </div>
      </div>
    </nav>
  );
}
```

LawyerCard.tsx

typescript

```tsx
interface LawyerCardProps {
  lawyer: {
    id: string;
    name: string;
    profile_photo_url?: string;
    specializations: string[];
    average_rating: number;
    total_reviews: number;
    years_experience: number;
    consultation_fee: number;
    languages: string[];
  };
  variant?: 'default' | 'compact';
}

export default function LawyerCard({ lawyer, variant = 'default' }: LawyerCardProps) {
  return (
    <Card className="hover:shadow-lg transition-shadow cursor-pointer">
      <CardContent className="p-6">
        <div className="flex gap-4">
          <Avatar className="h-20 w-20">
            <AvatarImage src={lawyer.profile_photo_url} />
            <AvatarFallback>{lawyer.name[0]}</AvatarFallback>
          </Avatar>

          <div className="flex-1">
            <h3 className="font-semibold text-lg">{lawyer.name}</h3>
            <div className="flex items-center gap-2 mt-1">
              <RatingDisplay rating={lawyer.average_rating} />
              <span className="text-sm text-gray-500">
                ({lawyer.total_reviews} reviews)
              </span>
            </div>

            <div className="flex gap-2 mt-2">
              {lawyer.specializations.slice(0, 2).map(spec => (
                <Badge key={spec} variant="secondary">{spec}</Badge>
              ))}
            </div>

            <div className="flex items-center gap-4 mt-3 text-sm text-gray-600">
              <span>{lawyer.years_experience} years exp.</span>
              <span>{lawyer.languages.join(', ')}</span>
```

```
          </div>
        </div>

        <div className="text-right">
          <div className="text-2xl font-bold text-blue-600">
            ₹{lawyer.consultation_fee}
          </div>
          <div className="text-sm text-gray-500">per consultation</div>

          <Link href={`/lawyers/${lawyer.id}`}>
            <Button className="mt-4 w-full">View Profile</Button>
          </Link>
        </div>
      </div>
    </CardContent>
  </Card>
 );
}
```

API INTEGRATION LAYER

/lib/api.ts (API Client)

typescript

```typescript
import axios, { AxiosInstance } from 'axios';

const API_BASE_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000/api/v1';

class ApiClient {
  private client: AxiosInstance;

  constructor() {
    this.client = axios.create({
      baseURL: API_BASE_URL,
      headers: {
        'Content-Type': 'application/json',
      },
    });

    // Request interceptor (add auth token)
    this.client.interceptors.request.use((config) => {
      if (typeof window !== 'undefined') {
        const token = localStorage.getItem('access_token');
        if (token) {
          config.headers.Authorization = `Bearer ${token}`;
        }
      }
      return config;
    });

    // Response interceptor (handle token refresh)
    this.client.interceptors.response.use(
      (response) => response,
      async (error) => {
        const originalRequest = error.config;

        if (error.response?.status === 401 && !originalRequest._retry) {
          originalRequest._retry = true;

          try {
            const refreshToken = localStorage.getItem('refresh_token');
            const { data } = await axios.post(`${API_BASE_URL}/auth/refresh`, {
              refresh_token: refreshToken,
            });

            localStorage.setItem('access_token', data.data.access_token);
            return this.client(originalRequest);
```

```
        } catch (refreshError) {
          localStorage.clear();
          window.location.href = '/auth/login';
          return Promise.reject(refreshError);
        }
      }

      return Promise.reject(error);
    }
  );
}

// Auth
async register(data: RegisterData) {
  return this.client.post('/auth/register', data);
}

async login(email_or_phone: string, password: string) {
  return this.client.post('/auth/login', { email_or_phone, password });
}

async logout() {
  return this.client.post('/auth/logout');
}

async verifyOTP(user_id: string, email_otp: string, phone_otp: string) {
  return this.client.post('/auth/verify-otp', { user_id, email_otp, phone_otp });
}

// Locations
async getStates() {
  return this.client.get('/locations/states');
}

async getDistricts(state_id: string) {
  return this.client.get(`/locations/states/${state_id}/districts`);
}

async getCourts(district_id: string) {
  return this.client.get(`/locations/districts/${district_id}/courts`);
}

async getSpecializations() {
  return this.client.get('/locations/specializations');
```

```
}

// Lawyers
async searchLawyers(params: SearchParams) {
  return this.client.get('/lawyers/search', { params });
}

async getLawyer(lawyer_id: string) {
  return this.client.get(`/lawyers/${lawyer_id}`);
}

async registerLawyer(formData: FormData) {
  return this.client.post('/lawyers/register', formData, {
    headers: { 'Content-Type': 'multipart/form-data' },
  });
}

// Bookings
async createBooking(data: CreateBookingData) {
  return this.client.post('/bookings/create', data);
}

async regenerateSummary(booking_draft_id: string, updated_description: string) {
  return this.client.post('/bookings/regenerate-summary', {
    booking_draft_id,
    updated_description,
  });
}

async confirmBooking(booking_draft_id: string) {
  return this.client.post('/bookings/confirm', { booking_draft_id });
}

async verifyPayment(razorpay_order_id: string, razorpay_payment_id: string, razorpay_signature: string) {
  return this.client.post('/payments/verify', {
    razorpay_order_id,
    razorpay_payment_id,
    razorpay_signature,
  });
}

async getUserBookings() {
  return this.client.get('/bookings/user');
}
```

```
async getBookingDetails(booking_id: string) {
  return this.client.get(`/bookings/${booking_id}`);
}

// Lawyer Bookings
async getLawyerPendingBookings() {
  return this.client.get('/lawyers/bookings/pending');
}

async respondToBooking(booking_id: string, action: 'accept' | 'reject' | 'reschedule', data: any) {
  return this.client.post(`/lawyers/bookings/${booking_id}/respond`, {
    action,
    ...data,
  });
}

// Consultations
async startConsultation(booking_id: string) {
  return this.client.post(`/consultations/${booking_id}/start`);
}

async endConsultation(booking_id: string) {
  return this.client.post(`/consultations/${booking_id}/end`);
}

async getChatMessages(booking_id: string) {
  return this.client.get(`/consultations/${booking_id}/messages`);
}

async sendChatMessage(booking_id: string, message: string) {
  return this.client.post(`/consultations/${booking_id}/messages`, { message });
}

// AI Assistant
async chatWithAI(message: string, context: any) {
  return this.client.post('/ai/chat', { message, context });
}

// Reviews
async submitReview(booking_id: string, rating: number, comment: string) {
  return this.client.post(`/bookings/${booking_id}/review`, { rating, comment });
}
```

```
  async getLawyerReviews(lawyer_id: string, page: number = 1) {
    return this.client.get(`/lawyers/${lawyer_id}/reviews`, { params: { page } });
  }

  // Admin
  async getPendingLawyers() {
    return this.client.get('/admin/lawyers/pending');
  }

  async verifyLawyer(lawyer_id: string, action: 'approve' | 'reject', rejection_reason?: string) {
    return this.client.post(`/admin/lawyers/${lawyer_id}/verify`, {
      action,
      rejection_reason,
    });
  }
}

export const api = new ApiClient();
export default api;
```

AUTHENTICATION CONTEXT

/contexts/AuthContext.tsx

typescript

```tsx
'use client';

import React, { createContext, useContext, useState, useEffect, ReactNode } from 'react';
import { useRouter } from 'next/navigation';
import api from '@/lib/api';

interface User {
  id: string;
  name: string;
  email: string;
  phone: string;
  user_type: 'user' | 'lawyer' | 'admin';
}

interface AuthContextType {
  user: User | null;
  loading: boolean;
  login: (email_or_phone: string, password: string) => Promise<void>;
  logout: () => Promise<void>;
  register: (data: any) => Promise<string>;
}

const AuthContext = createContext<AuthContextType | undefined>(undefined);

export function AuthProvider({ children }: { children: ReactNode }) {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);
  const router = useRouter();

  useEffect(() => {
    checkAuth();
  }, []);

  const checkAuth = async () => {
    try {
      const token = localStorage.getItem('access_token');
      if (token) {
        const { data } = await api.client.get('/auth/me');
        setUser(data.data);
      }
    } catch (error) {
      localStorage.clear();
    } finally {
```

```
      setLoading(false);
    }
  };

  const login = async (email_or_phone: string, password: string) => {
    const { data } = await api.login(email_or_phone, password);
    localStorage.setItem('access_token', data.data.access_token);
    localStorage.setItem('refresh_token', data.data.refresh_token);
    setUser(data.data.user);

    // Redirect based on user type
    if (data.data.user.user_type === 'admin') {
      router.push('/admin/dashboard');
    } else if (data.data.user.user_type === 'lawyer') {
      router.push('/lawyer/dashboard');
    } else {
      router.push('/dashboard');
    }
  };

  const logout = async () => {
    await api.logout();
    localStorage.clear();
    setUser(null);
    router.push('/');
  };

  const register = async (registerData: any) => {
    const { data } = await api.register(registerData);
    localStorage.setItem('access_token', data.data.access_token);
    localStorage.setItem('refresh_token', data.data.refresh_token);
    return data.data.user_id;
  };

  return (
    <AuthContext.Provider value={{ user, loading, login, logout, register }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) {
```

```
    throw new Error('useAuth must be used within AuthProvider');
  }
  return context;
}
```

STYLING GUIDELINES

Color Palette

```css
/* /app/globals.css */

:root {
  /* Primary Colors */
  --primary-blue: #3B82F6;
  --primary-blue-dark: #2563EB;
  --primary-blue-light: #60A5FA;

  /* Accent Colors */
  --accent-purple: #9333EA;
  --accent-purple-dark: #7E22CE;

  /* Neutral Colors */
  --gray-50: #F9FAFB;
  --gray-100: #F3F4F6;
  --gray-200: #E5E7EB;
  --gray-300: #D1D5DB;
  --gray-500: #6B7280;
  --gray-700: #374151;
  --gray-900: #111827;

  /* Semantic Colors */
  --success: #10B981;
  --warning: #F59E0B;
  --error: #EF4444;
  --info: #3B82F6;
}
```

Tailwind Custom Classes

javascript

```
// tailwind.config.js

module.exports = {
  theme: {
    extend: {
      colors: {
        primary: {
          DEFAULT: '#3B82F6',
          dark: '#2563EB',
          light: '#60A5FA',
        },
        accent: {
          DEFAULT: '#9333EA',
          dark: '#7E22CE',
        },
      },
      boxShadow: {
        'card': '0 1px 3px 0 rgba(0, 0, 0, 0.1), 0 1px 2px 0 rgba(0, 0, 0, 0.06)',
        'card-hover': '0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05)',
      },
      animation: {
        'fade-in': 'fadeIn 0.3s ease-in-out',
        'slide-up': 'slideUp 0.3s ease-out',
        'scale-in': 'scaleIn 0.2s ease-out',
      },
      keyframes: {
        fadeIn: {
          '0%': { opacity: '0' },
          '100%': { opacity: '1' },
        },
        slideUp: {
          '0%': { transform: 'translateY(10px)', opacity: '0' },
          '100%': { transform: 'translateY(0)', opacity: '1' },
        },
        scaleIn: {
          '0%': { transform: 'scale(0.95)', opacity: '0' },
          '100%': { transform: 'scale(1)', opacity: '1' },
        },
      },
    },
  },
};
```

RESPONSIVE DESIGN BREAKPOINTS

typescript

```
// Use these breakpoints consistently:

Mobile: < 640px (sm)
Tablet: 640px - 1024px (md, lg)
Desktop: > 1024px (xl, 2xl)

// Examples:
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
  {/* Cards */}
</div>

<div className="hidden md:block">
  {/* Desktop-only sidebar */}
</div>

<div className="md:hidden">
  {/* Mobile-only drawer */}
</div>
```

LOADING STATES & SKELETONS

Skeleton Components

typescript

```
// /components/shared/SkeletonCard.tsx

export function SkeletonCard() {
  return (
    <div className="bg-white rounded-lg shadow p-6 animate-pulse">
      <div className="flex gap-4">
        <div className="h-20 w-20 bg-gray-200 rounded-full" />
        <div className="flex-1 space-y-3">
          <div className="h-4 bg-gray-200 rounded w-3/4" />
          <div className="h-4 bg-gray-200 rounded w-1/2" />
          <div className="h-4 bg-gray-200 rounded w-full" />
        </div>
      </div>
    </div>
  );
}

// Usage:
{loading ? (
  <div className="space-y-4">
    {[1,2,3].map(i => <SkeletonCard key={i} />)}
  </div>
) : (
  lawyers.map(lawyer => <LawyerCard key={lawyer.id} lawyer={lawyer} />)
)}
```

ERROR HANDLING & EMPTY STATES

Empty State Component

typescript

```tsx
// /components/shared/EmptyState.tsx

interface EmptyStateProps {
  icon: React.ReactNode;
  title: string;
  description: string;
  action?: {
    label: string;
    onClick: () => void;
  };
}

export function EmptyState({ icon, title, description, action }: EmptyStateProps) {
  return (
    <div className="flex flex-col items-center justify-center py-12 px-4">
      <div className="text-gray-400 mb-4">
        {icon}
      </div>
      <h3 className="text-lg font-semibold text-gray-900 mb-2">{title}</h3>
      <p className="text-gray-500 text-center mb-6 max-w-md">{description}</p>
      {action && (
        <Button onClick={action.onClick}>{action.label}</Button>
      )}
    </div>
  );
}

// Usage:
{lawyers.length === 0 && (
  <EmptyState
    icon={<Search className="h-16 w-16" />}
    title="No lawyers found"
    description="Try adjusting your filters or search in a different location"
    action={{
      label: "Clear Filters",
      onClick: handleClearFilters
    }}
  />
)}
```

FORM VALIDATION

Use React Hook Form + Zod

typescript

```typescript
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import * as z from 'zod';

const registerSchema = z.object({
  name: z.string().min(3, "Name must be at least 3 characters"),
  email: z.string().email("Invalid email address"),
  phone: z.string().regex(/^\+91[0-9]{10}$/, "Invalid Indian phone number"),
  password: z.string()
    .min(8, "Password must be at least 8 characters")
    .regex(/[A-Z]/, "Must contain uppercase letter")
    .regex(/[0-9]/, "Must contain number")
    .regex(/[^A-Za-z0-9]/, "Must contain special character"),
});

export function RegisterForm() {
  const { register, handleSubmit, formState: { errors } } = useForm({
    resolver: zodResolver(registerSchema),
  });

  const onSubmit = async (data) => {
    // Submit to API
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <div>
        <Input {...register('name')} placeholder="Full Name" />
        {errors.name && <p className="text-red-500 text-sm mt-1">{errors.name.message}</p>}
      </div>
      {/* Other fields */}
    </form>
  );
}
```

ACCESSIBILITY REQUIREMENTS

1. **Semantic HTML**: Use proper heading hierarchy (h1, h2, h3)
2. **ARIA Labels**: Add aria-label to icon-only buttons
3. **Keyboard Navigation**: All interactive elements accessible via Tab
4. **Focus Indicators**: Visible focus rings on all focusable elements
5. **Alt Text**: All images must have descriptive alt text
6. **Color Contrast**: Minimum 4.5:1 for text, 3:1 for large text
7. **Form Labels**: All inputs must have associated labels

```typescript
// Good example:
<button
  aria-label="Search for lawyers"
  className="focus:ring-2 focus:ring-blue-500"
>
  <Search className="h-5 w-5" />
</button>

// Bad example:
<button>
  <Search />
</button>
```

PERFORMANCE OPTIMIZATION

1. Image Optimization

```typescript
```

```
import Image from 'next/image';

// Use Next.js Image component
<Image
  src={lawyer.profile_photo_url}
  alt={lawyer.name}
  width={200}
  height={200}
  className="rounded-full"
  priority={false} // Set to true for above-the-fold images
/>
```

2. Code Splitting

typescript

```
// Lazy load heavy components
import dynamic from 'next/dynamic';

const VideoConsultation = dynamic(
  () => import('@/components/consultation/VideoConsultation'),
  { ssr: false, loading: () => <PageLoader /> }
);
```

3. Memoization

typescript

```
import { useMemo, useCallback } from 'react';

const filteredLawyers = useMemo(() => {
  return lawyers.filter(lawyer => /* filter logic */);
}, [lawyers, filters]);

const handleFilterChange = useCallback((newFilters) => {
  setFilters(newFilters);
}, []);
```

DEPLOYMENT CHECKLIST

Environment Variables (.env.local)

```bash
NEXT_PUBLIC_API_URL=https://your-backend.onrender.com/api/v1
NEXT_PUBLIC_RAZORPAY_KEY_ID=rzp_live_xxx
NEXT_PUBLIC_AGORA_APP_ID=xxx
```

Build & Deploy to Vercel

```bash
npm run build  # Test locally
# Push to GitHub
# Vercel auto-deploys from main branch
```

Post-Deployment Testing

- [ ] All pages load without errors
- [ ] Authentication flow works
- [ ] Payment integration works (Razorpay)
- [ ] Video consultation works (Agora)
- [ ] Mobile responsive on all pages
- [ ] SEO meta tags present
- [ ] Analytics tracking works

FINAL DELIVERABLES

When you complete this UI build, you should have:

1. ✅ **18 Complete Pages**:
    - Homepage (enhanced)
    - Search Page
    - Lawyer Profile Page
    - User Registration
    - Login
    - Lawyer Registration (4-step wizard)

- OTP Verification
- User Dashboard (5 sub-pages)
- Lawyer Dashboard (4 sub-pages)
- Booking Creation (3 steps)
- Consultation Room
- AI Assistant
- Admin Dashboard (2 pages)

2. ✅ **30+ Reusable Components**:
   - All UI components from shadcn/ui
   - Custom shared components
   - Feature-specific components
3. ✅ **Complete API Integration**:
   - Centralized API client
   - Auth context with token refresh
   - Error handling
   - Loading states
4. ✅ **Responsive Design**:
   - Works on mobile, tablet, desktop
   - Touch-friendly on mobile
   - Optimized for all screen sizes
5. ✅ **Production-Ready**:
   - Environment variables configured
   - Error boundaries
   - Loading states
   - Empty states
   - SEO optimization

IMPLEMENTATION PRIORITY

Phase 1: Core Flow (Week 1)

1. Enhance Homepage

2. Search Page
3. Lawyer Profile Page
4. User Registration + Login
5. Booking Creation (Steps 1-3)

Phase 2: Dashboards (Week 2)

1. User Dashboard + My Bookings
2. Lawyer Dashboard + Booking Requests
3. Review System

Phase 3: Advanced Features (Week 3)

1. Consultation Room (Video/Audio)
2. AI Assistant
3. Lawyer Registration (4-step)
4. Admin Pages

Phase 4: Polish (Week 4)

1. Animations & transitions
2. Loading states everywhere
3. Error handling
4. Mobile optimization
5. Performance optimization
6. Accessibility audit

SUCCESS CRITERIA

Your frontend is complete when:

- ✅ A user can sign up, search lawyers, book consultation, pay, and join video call
- ✅ A lawyer can register, get verified, accept bookings, conduct consultations
- ✅ An admin can verify lawyers and view analytics
- ✅ All pages are mobile-responsive
- ✅ No console errors in production

- ✅ Page load time < 3 seconds
- ✅ Works in Chrome, Firefox, Safari, Edge

ADDITIONAL NOTES

1. **Use TypeScript strictly** - No `any` types except where absolutely necessary
2. **Follow Next.js 14 conventions** - Use App Router, Server Components where appropriate
3. **Keep components small** - Max 200 lines per component
4. **Write clean code** - Use ESLint, Prettier
5. **Test on real devices** - iOS Safari, Android Chrome
6. **Optimize images** - Use WebP format, lazy loading
7. **Add meta tags** - For SEO and social sharing
8. **Handle edge cases** - Network errors, empty states, loading states
9. **Add analytics** - Google Analytics or Mixpanel
10. **Monitor errors** - Sentry integration

Now build this beautiful, modern, production-ready frontend! 🚀