Mazdoor Connect - Frontend Development Specification

Project Brief for React Development Team

---

1. Project Overview

1.1 Product Description

Mazdoor Connect is a marketplace platform connecting homeowners in Pakistan with verified skilled workers (electricians, plumbers, AC mechanics, carpenters, painters). The platform enables users to browse verified worker profiles, view ratings/reviews, and book services with transparent pricing.

1.2 Project Goals

- Build a mobile-first responsive web application using React
- Provide seamless user experience for browsing and booking workers
- Enable workers to manage their profiles and job requests
- Admin dashboard for platform management
- Launch MVP in 6-8 weeks

1.3 Target Users

- **Customers**: Homeowners aged 25-55, middle to upper-middle class, urban Pakistan
- **Workers**: Skilled tradespeople (electricians, plumbers, AC mechanics, etc.)
- **Admin**: Platform owner managing operations

---

2. Technical Requirements

2.1 Technology Stack

Frontend Framework:

- React 18+ (with hooks)
- Next.js 14+ (for SEO and server-side rendering)
- TypeScript (strongly recommended)

Styling:

- Tailwind CSS (primary styling framework)
- Responsive design (mobile-first approach)
- Support for Urdu language (RTL layout)

State Management:

- React Context API or Zustand (for global state)
- React Query (for server state management)

Routing:

- Next.js App Router

Forms:

- React Hook Form
- Zod (for validation)

UI Components:

- Shadcn/ui (recommended) OR custom components
- Lucide React (for icons)

Authentication:

- NextAuth.js or Firebase Auth
- Phone number OTP verification

Maps:

- Google Maps API (for location selection and worker proximity)

Payment Integration:

- JazzCash API
- EasyPaisa API
- (Note: Initially manual, integrate in Phase 2)

Deployment:

- Vercel (recommended for Next.js)
- OR Netlify

2.2 Browser & Device Support

Browsers:

- Chrome (latest 2 versions)
- Firefox (latest 2 versions)
- Safari (latest 2 versions)
- Edge (latest 2 versions)

Devices:

- Mobile: 375px - 768px (priority)
- Tablet: 768px - 1024px
- Desktop: 1024px+

Operating Systems:

- iOS 14+
- Android 10+
- Windows 10+
- macOS 10.15+

---

3. User Roles & Permissions

3.1 Customer (End User)

Can:

- Browse workers by category
- Filter and search workers
- View worker profiles (ratings, reviews, portfolio)
- Book appointments
- Track booking status
- Make payments
- Rate and review workers
- Manage their profile
- View booking history

Cannot:

- Access worker dashboard
- Access admin features

3.2 Worker

Can:

- Create and manage profile
- Upload portfolio photos
- Set availability
- Receive job notifications
- Accept/reject jobs
- Update job status
- View earnings
- View ratings and reviews
- Chat with customers (Phase 2)

Cannot:

- Access other worker profiles (private data)
- Access admin features
- Modify platform settings

3.3 Admin

Can:

- View all users (customers & workers)
- Approve/reject worker applications
- Manage worker verification status

- View all bookings
- Manually assign workers to jobs
- Handle disputes
- View analytics dashboard
- Manage platform settings
- Send notifications
- Generate reports

---

4. Application Structure

4.1 Page Hierarchy

```
/
├── / (Home)
├── /how-it-works
├── /categories
│   ├── /categories/electrician
│   ├── /categories/plumber
│   ├── /categories/ac-mechanic
│   ├── /categories/carpenter
│   └── /categories/painter
├── /workers
│   └── /workers/[workerId] (Worker Profile)
├── /book
│   └── /book/[workerId]
├── /login
├── /signup
├── /verify-otp
├── /customer
│   ├── /customer/dashboard
│   ├── /customer/bookings
│   ├── /customer/profile
│   └── /customer/favorites
├── /worker
│   ├── /worker/dashboard
│   ├── /worker/profile
│   ├── /worker/jobs
│   ├── /worker/earnings
│   └── /worker/reviews
├── /admin
│   ├── /admin/dashboard
│   ├── /admin/workers
│   ├── /admin/customers
│   ├── /admin/bookings
│   ├── /admin/verification
│   ├── /admin/analytics
│   └── /admin/settings
├── /about
├── /contact
├── /privacy-policy
└── /terms-of-service
```

---

5. Detailed Page Specifications

5.1 Home Page (/)

Purpose: Landing page to attract and convert visitors

Sections:

Hero Section:

- Headline (Urdu + English): "محفوظ اور تصدیق شدہ کاریگر | Verified Workers in 1 Hour"
- Subheadline: "Police verified workers • Fixed prices • Damage protection up to Rs. 25,000"
- Primary CTA: "Find a Worker" (button)
- Hero image: Professional worker photo or illustration
- Language toggle: Urdu | English

Category Grid:

- 5 category cards (Electrician, Plumber, AC Mechanic, Carpenter, Painter)
- Each card shows:
  - Icon
  - Category name
  - Available workers count (e.g., "47 verified workers")
  - "View All" link

How It Works:

- 3-step process with icons:
  1. Browse & Select Worker
  2. Book Appointment
  3. Get Work Done
- Each step has brief description

Trust Indicators:

- Stats row:
  - "500+ Happy Customers"
  - "150 Verified Workers"
  - "4.8★ Average Rating"
  - "10,000+ Jobs Completed"
- Verification badges:
  - CNIC Verified
  - Police Verified
  - Skill Tested
  - Damage Protection

Featured Workers:

- Carousel showing 6 top-rated workers
- Each card shows:
  - Photo

    - Name
    - Category
    - Rating
    - Jobs completed
    - "View Profile" button

Customer Testimonials:

- 3-4 testimonials with:
    - Customer photo (optional)
    - Name
    - Review text
    - Rating stars
    - Service used

Footer:

- Links: About, How It Works, Categories, Contact
- Social media icons
- Copyright
- Language selector
- Download app (Phase 2)

Technical Notes:

- Fully responsive
- Lazy load images
- Optimize for Core Web Vitals
- SEO optimized (meta tags, structured data)

5.2 Category Page (/categories/[category])

Example: /categories/electrician

Purpose: Show all workers in specific category with filtering

Components:

Header:

- Breadcrumb: Home > Categories > Electrician
- Page title: "Electricians in Karachi"
- Subtitle: "47 verified electricians available"

Filter Sidebar (Desktop) / Filter Modal (Mobile):

- **Location:**
    - Map view OR dropdown of areas
    - Auto-detect current location (with permission)
- **Availability:**

    - Available Today
    - Available This Week
    - Custom Date Picker
- **Rating:**
    - 4.5+ stars
    - 4.0+ stars
    - All ratings
- **Price Range:**
    - Slider (Rs. 0 - Rs. 10,000)
- **Experience:**
    - 0-2 years
    - 3-5 years
    - 5-10 years
    - 10+ years
- **Specialties:** (for AC Mechanics)
    - Split AC
    - Window AC
    - Inverter AC
    - Installation
    - Repair
    - Gas Refilling

Sort Options:

- Highest Rated
- Most Jobs Completed
- Nearest to Me
- Lowest Price
- Newest

Worker Cards Grid:

- Grid layout: 1 column (mobile), 2 columns (tablet), 3 columns (desktop)
- Each card shows:
    - Profile photo
    - Verification badges (CNIC ✓, Police ✓, Skill ✓)
    - Name, age
    - Category
    - Rating (stars + number, e.g., 4.8 ★ from 47 reviews)
    - Jobs completed (e.g., "127 jobs")
    - Price range (e.g., "Rs. 500-800 per service")
    - Availability (e.g., "Available today")
    - Specialties (tags: Split AC, Window AC)
    - "View Profile" button
    - Heart icon (add to favorites)

Pagination:

- Show 12 workers per page
- Infinite scroll OR numbered pagination

Empty State:

- If no workers match filters:
    - Message: "No workers found. Try adjusting filters."
    - Button: "Reset Filters"

Technical Notes:

- Implement debounced search
- URL params for filters (shareable links)
- Skeleton loaders while fetching

5.3 Worker Profile Page (/workers/[workerId])

Purpose: Detailed worker information to help customers decide

Layout:

Top Section:

- Worker photo (large, professional)
- Verification badges (prominent)
- Name, age
- Category (e.g., "AC Mechanic")
- Rating (large, e.g., 4.8 ★★★★★)
- Number of reviews (e.g., "Based on 47 reviews")
- Jobs completed (e.g., "127 completed jobs")
- Member since (e.g., "Member since Jan 2026")
- Location (e.g., "DHA Phase 5, Karachi")

Action Buttons:

- Primary: "Book Now" (large, prominent)
- Secondary: "Send Message" (Phase 2)
- Icon: Heart (add to favorites)

About Section:

- Years of experience
- Brief bio (e.g., "15 years experience in AC repair and installation. Specialized in inverter ACs and troubleshooting.")
- Specialties (tags)
- Languages spoken (Urdu, English, etc.)

Services & Pricing:

- Table format:

Service	Price Range	Duration
AC Gas Refill	Rs. 2,500-3,500	1-2 hours
AC Deep Cleaning	Rs. 1,500-2,500	1 hour
AC Installation (1-ton)	Rs. 3,500-5,000	2-3 hours

- Note: "Final price depends on complexity and parts required"

Availability Calendar:

- Calendar showing available dates
- Time slots for selected date
- "Available Today" indicator if applicable

Portfolio:

- Photo gallery of previous work
- Grid layout, clickable to expand
- Before/After photos if available

Reviews Section:

- Overall rating breakdown:
  - 5 stars: [progress bar] 78%
  - 4 stars: [progress bar] 15%
  - 3 stars: [progress bar] 5%
  - 2 stars: [progress bar] 2%
  - 1 star: [progress bar] 0%
- Individual reviews:
  - Customer name (first name + initial)
  - Rating (stars)
  - Date
  - Service used
  - Review text
  - Optional: Photo uploaded by customer
- Pagination: Show 10 reviews per page
- Sort: Newest First, Highest Rated, Lowest Rated

Verification Details:

- Expandable accordion:
  - CNIC Verified ✓ (verified on [date])
  - Police Verified ✓ (certificate on file)
  - Skill Tested ✓ (passed practical test on [date])
  - Background Checked ✓

Similar Workers:

- "Other AC Mechanics You Might Like"
- Horizontal scrollable cards
- 6 workers shown

Report Button:

- Small link at bottom: "Report this profile"

Technical Notes:

- Image optimization (lazy load, WebP format)
- Share functionality (WhatsApp, Facebook)
- Print-friendly version
- Structured data for SEO

5.4 Booking Page (/book/[workerId])

Purpose: Customer books appointment with selected worker

Progress Indicator:

- Step 1: Service Details (active)
- Step 2: Date & Time
- Step 3: Contact Info
- Step 4: Confirmation

Step 1: Service Details

- Worker summary card (photo, name, rating)
- Service selection:
  - Dropdown: Select service type (e.g., "AC Gas Refill")
  - Price range shown: Rs. 2,500-3,500
- Problem description:
  - Text area: "Describe the issue" (optional)
  - File upload: "Upload photos" (optional, max 3 photos)
- Address:
  - Auto-fill if user logged in
  - Text input: Full address
  - Map: Pin location (Google Maps integration)
  - Optional: Apartment/floor number
- Next button

Step 2: Date & Time

- Calendar view (available dates highlighted)
- Time slots for selected date:
  - 8:00 AM - 10:00 AM
  - 10:00 AM - 12:00 PM

    - 12:00 PM - 2:00 PM
    - 2:00 PM - 4:00 PM
    - 4:00 PM - 6:00 PM
    - 6:00 PM - 8:00 PM
- Note: "Worker will confirm exact time after reviewing your request"
- Emergency option: "Need urgent service today?" (+30% fee)
- Back button, Next button

Step 3: Contact Info

- If logged in: Auto-fill, allow edit
- If not logged in:
    - Name (required)
    - Phone number (required)
    - Email (optional)
    - "Create account" checkbox (save for future bookings)
- Special instructions:
    - Text area: "Anything else the worker should know?"
- Back button, Next button

Step 4: Confirmation

- Booking summary:
    - Worker: [Photo, Name, Category]
    - Service: [Service name]
    - Date & Time: [Selected date/time]
    - Address: [Full address]
    - Estimated Price: Rs. 2,500-3,500
    - Platform Fee: Rs. 300
    - Total Estimate: Rs. 2,800-3,800
- Terms checkbox: "I agree to Terms of Service and Cancellation Policy"
- Payment info:
    - "Payment to be made after work completion"
    - Payment methods: Cash, JazzCash, EasyPaisa
- Confirm Booking button (large, primary)
- Back button

Success Screen:

- Checkmark animation
- "Booking Confirmed!"
- Booking reference number: #MZ12345
- Message: "We've sent confirmation to your phone and email"
- What's Next:
    - "Worker will call you within 15 minutes to confirm"
    - "You'll receive updates via SMS"

- Action buttons:
  - "View Booking Details"
  - "Back to Home"

Technical Notes:

- Form validation (client + server side)
- Auto-save draft (if user navigates away)
- Loading states
- Error handling
- SMS/Email confirmation

5.5 Customer Dashboard (/customer/dashboard)

Purpose: Customer's main hub for managing bookings and profile

Layout:

Header:

- Welcome message: "Welcome back, [First Name]!"
- Quick stats:
  - Total bookings
  - Upcoming bookings
  - Favorite workers

Upcoming Bookings Section:

- Card for each upcoming booking:
  - Worker photo, name
  - Service type
  - Date & time
  - Address (truncated)
  - Status: "Pending Confirmation" | "Confirmed" | "In Progress"
  - Actions:
    - View Details
    - Cancel Booking
    - Message Worker (Phase 2)
- If no upcoming bookings:
  - Empty state: "No upcoming bookings"
  - CTA: "Find a Worker"

Recent Bookings:

- List of 5 most recent completed bookings
- Each shows:
  - Worker photo, name
  - Service

    - Date
    - Rating (if rated) OR "Leave Review" button
    - "View Details" link
- "View All Bookings" link

Favorite Workers:

- Horizontal scrollable cards
- Each shows:
    - Worker photo
    - Name, category
    - Rating
    - "Book Again" button
- "View All Favorites" link

Quick Actions:

- "Find a Worker" button
- "My Profile" button
- "Help & Support" button

Sidebar (Desktop) / Bottom Nav (Mobile):

- Dashboard (current)
- My Bookings
- Favorites
- Profile
- Logout

Technical Notes:

- Real-time updates (if booking status changes)
- Notifications badge (new messages, booking updates)

5.6 Customer Bookings Page (/customer/bookings)

Purpose: View all past and upcoming bookings

Tabs:

- Upcoming (default)
- In Progress
- Completed
- Cancelled

Each Booking Card Shows:

- Worker photo, name, category
- Service type
- Booking ID (e.g., #MZ12345)

- Date & time
- Address
- Status badge (color-coded)
- Price (for completed)
- Actions (based on status):
  - Upcoming: "Cancel" | "Reschedule" | "View Details"
  - In Progress: "View Details" | "Call Worker"
  - Completed: "View Details" | "Leave Review" (if not reviewed) | "Book Again"
  - Cancelled: "View Details" | "Book Again"

Filters:

- Date range picker
- Service type dropdown
- Worker dropdown (if customer has history)

Search:

- Search by booking ID or worker name

Pagination:

- 10 bookings per page

Empty States:

- "No upcoming bookings"
- "No completed bookings yet"

5.7 Booking Details Page (/customer/bookings/[bookingId])

Purpose: Detailed view of single booking

Sections:

Status Banner:

- Color-coded based on status
- "Booking Confirmed" | "Worker On The Way" | "Work In Progress" | "Completed"
- Estimated completion time (if in progress)

Booking Information:

- Booking ID: #MZ12345
- Booked on: [Date, Time]
- Status: [Current status]
- Service: [Service name]
- Date & Time: [Scheduled date/time]
- Address: [Full address with map]
- Special Instructions: [If any]

Worker Details:

- Photo, name
- Category
- Rating
- Phone: [Clickable to call]
- "Message Worker" button (Phase 2)

Pricing:

- Service estimate: Rs. 2,500-3,500
- Platform fee: Rs. 300
- Total estimate: Rs. 2,800-3,800
- (After completion):
    - Final amount: Rs. 3,000
    - Payment method: Cash/JazzCash/EasyPaisa
    - Payment status: Paid/Pending

Timeline:

- Booking created: [Date, Time]
- Confirmed by worker: [Date, Time]
- Worker arrived: [Date, Time] (if applicable)
- Work started: [Date, Time] (if applicable)
- Work completed: [Date, Time] (if applicable)

Actions:

- If upcoming: "Cancel Booking" | "Reschedule"
- If completed and not reviewed: "Leave Review"
- "Download Invoice" (PDF)
- "Report Issue"

Review Section (if completed):

- Rating given (stars)
- Review text
- Photos uploaded (if any)

5.8 Worker Dashboard (/worker/dashboard)

Purpose: Worker's main hub for managing jobs and profile

Header:

- Welcome message: "Welcome, [Worker Name]!"
- Profile completion: Progress bar (e.g., "80% complete")
- Quick stats:
    - Rating: 4.8 ★
    - Jobs this month: 23

    - Earnings this month: Rs. 57,000

Pending Job Requests:

- Cards for new booking requests:
    - Customer name (first name + initial)
    - Service requested
    - Date & time requested
    - Address (with distance from worker)
    - Customer phone: [Clickable]
    - Problem description
    - Photos (if uploaded)
    - Estimated earnings: Rs. 2,700 (Rs. 3,000 - Rs. 300 platform fee)
    - Actions:
        - "Accept" (green button)
        - "Decline" (red button)
- If no pending requests:
    - Empty state: "No new job requests"
    - Tip: "Make sure your availability is up to date"

Today's Schedule:

- List of confirmed jobs for today:
    - Time slot
    - Customer name
    - Service type
    - Address
    - Phone number
    - Navigation button (opens Google Maps)
    - "Update Status" button:
        - On My Way
        - Started Work
        - Completed
- If no jobs today:
    - "No jobs scheduled for today"

This Week:

- Calendar view showing upcoming jobs
- Click date to see details

Quick Actions:

- "Update Availability"
- "Edit Profile"
- "View Earnings"

Sidebar/Bottom Nav:

- Dashboard (current)
- Jobs
- Profile
- Earnings
- Reviews
- Logout

5.9 Worker Profile Management (/worker/profile)

Purpose: Worker creates and edits their profile

Sections:

Profile Photo:

- Current photo display
- "Change Photo" button
- Guidelines: "Professional photo, clear face, good lighting"
- File upload (max 5MB, JPG/PNG)
- Crop tool

Basic Information:

- Full Name (required)
- Phone Number (required, verified)
- Email (optional)
- Date of Birth (required)
- CNIC Number (required, verified)
- Address (required)

Professional Details:

- Category (dropdown: Electrician, Plumber, AC Mechanic, Carpenter, Painter)
- Years of Experience (number input)
- Specialties (multi-select tags):
  - For AC Mechanic: Split AC, Window AC, Inverter AC, Installation, Repair, Gas Refilling, etc.
- Bio (text area, max 500 characters):
  - Placeholder: "Describe your experience and what makes you great at your job"
- Languages (multi-select: Urdu, English, Punjabi, Sindhi, etc.)

Services & Pricing:

- Add Service button
- Each service:
  - Service name (dropdown + custom)
  - Price range (min - max)
  - Estimated duration

    - Remove button

Availability:

- Weekly schedule:
    - For each day: Available/Not Available toggle
    - If available: Time slots (start time - end time)
- Emergency availability: Yes/No
- "I'm available on weekends": Checkbox

Portfolio:

- Upload work photos
- Grid display of uploaded photos
- Max 20 photos
- Caption for each photo (optional)
- Delete option

Bank Details (for payments):

- Bank name
- Account title
- Account number
- OR JazzCash/EasyPaisa number

Verification Status:

- CNIC: Verified ✓ | Pending | Not Submitted
- Police Verification: Verified ✓ | Pending | Not Submitted
- Skill Test: Passed ✓ | Not Taken
- Upload documents button (if pending)

Save Button:

- "Save Changes" (validates all required fields)
- Success message on save

Technical Notes:

- Real-time validation
- Auto-save draft (every 30 seconds)
- Image compression on upload
- Form state management

5.10 Worker Jobs Page (/worker/jobs)

Purpose: Manage all jobs (pending, upcoming, completed)

Tabs:

- New Requests (default, shows count badge)

- Upcoming
- In Progress
- Completed
- Cancelled

New Requests Tab:

- Each request card:
    - Customer info
    - Service details
    - Date/time
    - Address (distance)
    - Estimated earnings
    - Accept/Decline buttons
    - Timer: "Expires in 2h 15m" (auto-decline after 3 hours)

Upcoming Tab:

- Confirmed jobs
- Each shows:
    - Date/time
    - Customer name, phone
    - Service
    - Address
    - Earnings
    - "Get Directions" button
    - "Update Status" dropdown:
        - On My Way
        - Arrived
        - Started Work

In Progress Tab:

- Active jobs
- Update status to "Completed"
- Timer: How long job has been in progress

Completed Tab:

- Past jobs
- Shows:
    - Date
    - Customer
    - Service
    - Amount earned
    - Rating received
    - View details

Filters:

- Date range
- Service type
- Customer (if repeat customer)

Search:

- By booking ID or customer name

5.11 Worker Earnings Page (/worker/earnings)

Purpose: Track earnings and payment history

Summary Cards:

- This Month:
  - Total Earnings: Rs. 57,000
  - Jobs Completed: 23
  - Average per Job: Rs. 2,478
- Last Month:
  - Total Earnings: Rs. 49,500
  - Jobs Completed: 20
- All Time:
  - Total Earnings: Rs. 287,000
  - Total Jobs: 127

Chart:

- Line/Bar chart showing monthly earnings
- Toggle: Last 6 months | Last 12 months | All time

Transaction History:

- Table format: | Date | Booking ID | Customer | Service | Amount | Platform Fee | Your Earning | Status |
- Filter by:
  - Date range
  - Status: Paid, Pending, Disputed
- Export: Download CSV

Payment Schedule:

- "Next payout on: [Date]"
- Pending amount: Rs. 12,500
- "Payouts processed weekly every Monday"

Bank Details:

- Current bank account: [Masked]
- "Update Bank Details" button

5.12 Worker Reviews Page (/worker/reviews)

Purpose: View all reviews received

Overall Stats:

- Average Rating: 4.8 ★★★★★
- Total Reviews: 47
- Rating Distribution (visual):
  - 5 stars: 78% (37 reviews)
  - 4 stars: 15% (7 reviews)
  - 3 stars: 5% (2 reviews)
  - 2 stars: 2% (1 review)
  - 1 star: 0% (0 reviews)

Reviews List:

- Each review shows:
  - Customer name (first name + initial)
  - Rating (stars)
  - Date
  - Service performed
  - Review text
  - Photos (if customer uploaded)
- Sort by:
  - Newest First
  - Highest Rated
  - Lowest Rated
- Filter by:
  - Rating
  - Service type
  - Date range

Response Feature (Phase 2):

- Worker can reply to reviews
- "Thank you for the kind words!"

Report Feature:

- If review is inappropriate/fake
- "Report this review" link

5.13 Admin Dashboard (/admin/dashboard)

Purpose: High-level overview of platform metrics

Key Metrics (Cards):

- Total Users:
  - Customers: 1,247
  - Workers: 153
  - Growth: +12% this month
- Total Bookings:
  - This Month: 2,134
  - Last Month: 1,899
  - Growth: +12.4%
- Revenue:
  - This Month: Rs. 534,000
  - Last Month: Rs. 475,000
  - Growth: +12.4%
- Average Rating:
  - Platform: 4.7 ★
  - Change: +0.1 from last month

Charts:

- Bookings Over Time:
  - Line chart showing daily bookings (last 30 days)
- Revenue Over Time:
  - Bar chart showing monthly revenue (last 12 months)
- Category Breakdown:
  - Pie chart: Jobs by category
- Geographic Distribution:
  - Map showing bookings by area

Recent Activity:

- Last 10 bookings (live feed)
  - Booking ID
  - Customer
  - Worker
  - Service
  - Status
  - View Details link

Alerts:

- Pending worker verifications: 7
- Unresolved disputes: 2
- Low-rated workers: 3 (below 4.0)
- Failed payments: 1

Quick Actions:

- Verify Pending Workers
- Review Disputes
- Send Platform Announcement
- Generate Report

5.14 Admin Workers Management (/admin/workers)

Purpose: Manage all workers on platform

Tabs:

- All Workers (default)
- Pending Verification
- Active
- Suspended
- Rejected

Worker Table: Columns:

- Photo
- Name
- Category
- Rating
- Jobs Completed
- Status (Active | Pending | Suspended | Rejected)
- Verification Status (CNIC ✓, Police ✓, Skill ✓)
- Member Since
- Actions (dropdown):
    - View Profile
    - Edit
    - Verify
    - Suspend
    - Delete

Filters:

- Category
- Rating
- Location
- Verification status
- Date joined

Search:

- By name, phone, CNIC

Bulk Actions:

- Select multiple workers

- Bulk approve
- Bulk suspend
- Export selected

Pending Verification Tab:

- Workers awaiting verification
- Each row shows:
    - Submitted documents (clickable to view)
    - Skill test status
    - Actions:
        - Approve
        - Request More Info
        - Reject

5.15 Admin Bookings Management (/admin/bookings)

Purpose: View and manage all bookings

Filters:

- Status: All | Pending | Confirmed | In Progress | Completed | Cancelled | Disputed
- Date range
- Category
- Worker
- Customer

Booking Table: Columns:

- Booking ID
- Date & Time
- Customer
- Worker
- Service
- Amount
- Status (color-coded badges)
- Actions:
    - View Details
    - Assign Worker (if pending)
    - Cancel
    - Resolve Dispute (if disputed)

Click Row:

- Opens booking details modal
- Shows full information
- Timeline of events
- Chat history (Phase 2)

- Actions available based on status

Disputed Bookings:

- Separate section
- Shows both customer and worker side
- Admin can:
  - Message both parties
  - Refund customer
  - Penalize worker
  - Mark as resolved

5.16 Admin Analytics (/admin/analytics)

Purpose: Deep dive into platform metrics

Date Range Selector:

- Last 7 days | Last 30 days | Last 90 days | Last 12 months | Custom

Metrics Sections:

1. User Metrics:

- New customer signups (chart)
- Customer retention rate
- New worker signups (chart)
- Worker retention rate
- Active users (daily/monthly)

2. Booking Metrics:

- Total bookings (chart)
- Booking conversion rate (visitors → bookings)
- Average booking value
- Cancellation rate
- Completion rate

3. Revenue Metrics:

- Total revenue (chart)
- Revenue by category (pie chart)
- Revenue by area (map)
- Average platform fee
- Revenue per worker
- Revenue per customer

4. Performance Metrics:

- Average response time (worker acceptance)
- Average job completion time

- Customer satisfaction score
- Worker satisfaction score
- Repeat booking rate

5. Category Insights:

- Most demanded services
- Highest revenue categories
- Fastest growing categories
- Category-wise ratings

Export Options:

- Download reports as PDF/CSV/Excel
- Schedule automated weekly/monthly reports

---

6. Component Specifications

6.1 Reusable Components

WorkerCard Component:

```typescript
interface WorkerCardProps {
  worker: {
    id: string;
    name: string;
    photo: string;
    category: string;
    rating: number;
    reviewCount: number;
    jobsCompleted: number;
    priceRange: { min: number; max: number };
    specialties: string[];
    availableToday: boolean;
    verified: {
      cnic: boolean;
      police: boolean;
      skill: boolean;
    };
  };
  onBook?: () => void;
  onFavorite?: () => void;
  isFavorite?: boolean;
}
```

Design:

- Card with hover effect
- Image with verification badges overlay
- Name, category
- Star rating + review count

- "X jobs completed"
- Price range
- Specialties (max 3 tags)
- "Available today" badge (if applicable)
- "Book Now" button
- Heart icon (favorite)

BookingCard Component:

```typescript
interface BookingCardProps {
  booking: {
    id: string;
    worker: {
      name: string;
      photo: string;
      category: string;
    };
    service: string;
    date: Date;
    time: string;
    address: string;
    status: 'pending' | 'confirmed' | 'in-progress' | 'completed' | 'cancelled';
    price?: number;
  };
  onView?: () => void;
  onCancel?: () => void;
  onReview?: () => void;
}
```

Design:

- Status badge (colored)
- Worker photo + name
- Service type
- Date/time
- Address (truncated)
- Action buttons based on status

RatingDisplay Component:

```typescript
interface RatingDisplayProps {
  rating: number; // 0-5
  reviewCount?: number;
  size?: 'small' | 'medium' | 'large';
  showNumber?: boolean;
}
```

Design:

- Filled/half-filled/empty stars
- Number display (e.g., "4.8")
- Review count (e.g., "from 47 reviews")

ServicePriceCard Component:

typescript

```typescript
interface ServicePriceCardProps {
  service: {
    name: string;
    priceMin: number;
    priceMax: number;
    duration?: string;
  };
}
```

Design:

- Service name
- Price range (Rs. X - Rs. Y)
- Duration (if provided)

ReviewCard Component:

typescript

```typescript
interface ReviewCardProps {
  review: {
    id: string;
    customerName: string;
    rating: number;
    date: Date;
    service: string;
    text: string;
    photos?: string[];
  };
}
```

Design:

- Customer name (masked)
- Star rating
- Date
- Service type
- Review text
- Photos (if any, gallery)

FilterSidebar Component:

```typescript
interface FilterSidebarProps {
  filters: {
    location?: string;
    availability?: string;
    rating?: number;
    priceRange?: { min: number; max: number };
    experience?: string;
    specialties?: string[];
  };
  onFilterChange: (filters: Filters) => void;
  onReset: () => void;
}
```

Design:

- Accordion sections for each filter type
- Apply/Reset buttons
- Mobile: Full-screen modal
- Desktop: Sidebar

StatusBadge Component:

```typescript
interface StatusBadgeProps {
  status: 'pending' | 'confirmed' | 'in-progress' | 'completed' | 'cancelled' | 'disputed';
  size?: 'small' | 'medium' | 'large';
}
```

Design:

- Color-coded badges:
  - Pending: Yellow
  - Confirmed: Blue
  - In Progress: Purple
  - Completed: Green
  - Cancelled: Red
  - Disputed: Orange

VerificationBadge Component:

```typescript
interface VerificationBadgeProps {
  type: 'cnic' | 'police' | 'skill';
  verified: boolean;
  size?: 'small' | 'medium';
}
```

Design:

- Checkmark icon + label
- Green if verified, gray if not

EmptyState Component:

typescript

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

Design:

- Centered content
- Large icon (illustration)
- Title
- Description
- CTA button (if action provided)

LoadingSpinner Component:

typescript

```typescript
interface LoadingSpinnerProps {
  size?: 'small' | 'medium' | 'large';
  fullScreen?: boolean;
}
```

Design:

- Animated spinner
- Optional: Full-screen overlay

Modal Component:

typescript

```typescript
interface ModalProps {
  isOpen: boolean;
  onClose: () => void;
  title?: string;
  children: React.ReactNode;
  size?: 'small' | 'medium' | 'large' | 'full';
}
```

Design:

- Overlay background
- Centered modal
- Close button (X)
- Title bar
- Content area
- Responsive (full-screen on mobile for large modals)

6.2 Form Components

Input Field:

typescript

```typescript
interface InputFieldProps {
  label: string;
  type?: 'text' | 'email' | 'tel' | 'number' | 'password';
  value: string;
  onChange: (value: string) => void;
  error?: string;
  required?: boolean;
  placeholder?: string;
  disabled?: boolean;
  icon?: React.ReactNode;
}
```

Select Dropdown:

typescript

```typescript
interface SelectFieldProps {
  label: string;
  options: { value: string; label: string }[];
  value: string;
  onChange: (value: string) => void;
  error?: string;
  required?: boolean;
  placeholder?: string;
}
```

Text Area:

typescript

```typescript
interface TextAreaProps {
  label: string;
  value: string;
  onChange: (value: string) => void;
  error?: string;
  required?: boolean;
  placeholder?: string;
  rows?: number;
  maxLength?: number;
}
```

File Upload:

typescript

```typescript
interface FileUploadProps {
  label: string;
  accept?: string;
  maxSize?: number; // in MB
  multiple?: boolean;
  onUpload: (files: File[]) => void;
  error?: string;
}
```

Date Picker:

typescript

```typescript
interface DatePickerProps {
  label: string;
  value: Date | null;
  onChange: (date: Date) => void;
  minDate?: Date;
  maxDate?: Date;
  error?: string;
}
```

Time Picker:

typescript

```typescript
interface TimePickerProps {
  label: string;
  value: string; // HH:MM format
  onChange: (time: string) => void;
  error?: string;
}
```

Rating Input:

typescript

```typescript
interface RatingInputProps {
  value: number;
  onChange: (rating: number) => void;
  max?: number; // default 5
}
```

Design:

- Interactive stars
- Hover effect
- Click to select

7. API Integration Requirements

7.1 Backend API Endpoints (To Be Consumed)

Authentication:

- `POST /api/auth/send-otp` - Send OTP to phone
- `POST /api/auth/verify-otp` - Verify OTP and login
- `POST /api/auth/logout` - Logout user
- `GET /api/auth/me` - Get current user info

Workers:

- `GET /api/workers` - Get all workers (with filters, pagination)
- `GET /api/workers/:id` - Get worker by ID
- `POST /api/workers` - Create worker profile
- `PATCH /api/workers/:id` - Update worker profile
- `DELETE /api/workers/:id` - Delete worker
- `GET /api/workers/:id/reviews` - Get worker reviews
- `GET /api/workers/:id/availability` - Get worker availability

Customers:

- `GET /api/customers/:id` - Get customer profile
- `PATCH /api/customers/:id` - Update customer profile
- `GET /api/customers/:id/bookings` - Get customer bookings
- `GET /api/customers/:id/favorites` - Get favorite workers
- `POST /api/customers/:id/favorites/:workerId` - Add to favorites
- `DELETE /api/customers/:id/favorites/:workerId` - Remove from favorites

Bookings:

- `GET /api/bookings` - Get all bookings (admin)
- `GET /api/bookings/:id` - Get booking by ID
- `POST /api/bookings` - Create new booking
- `PATCH /api/bookings/:id` - Update booking
- `DELETE /api/bookings/:id` - Cancel booking
- `POST /api/bookings/:id/review` - Submit review

Categories:

- `GET /api/categories` - Get all service categories
- `GET /api/categories/:slug/workers` - Get workers by category

Reviews:

- `POST /api/reviews` - Create review
- `GET /api/reviews/:id` - Get review
- `DELETE /api/reviews/:id` - Delete review (admin)

Payments:

- `POST /api/payments/initiate` - Initiate payment
- `POST /api/payments/confirm` - Confirm payment
- `GET /api/payments/:id` - Get payment details

Analytics (Admin):

- `GET /api/analytics/dashboard` - Dashboard metrics
- `GET /api/analytics/revenue` - Revenue analytics
- `GET /api/analytics/bookings` - Booking analytics
- `GET /api/analytics/users` - User analytics

Notifications:

- `GET /api/notifications` - Get user notifications
- `PATCH /api/notifications/:id/read` - Mark as read

Admin:

- `GET /api/admin/workers/pending` - Get workers pending verification
- `PATCH /api/admin/workers/:id/verify` - Verify worker
- `PATCH /api/admin/workers/:id/suspend` - Suspend worker
- `GET /api/admin/disputes` - Get disputed bookings
- `PATCH /api/admin/disputes/:id/resolve` - Resolve dispute

7.2 API Response Formats

Success Response:

```json
{
  "success": true,
  "data": { ... },
  "message": "Operation successful"
}
```

Error Response:

```json
{
  "success": false,
  "error": {
    "code": "INVALID_INPUT",
    "message": "Validation failed",
    "details": [
      {
        "field": "phone",
        "message": "Invalid phone number format"
      }
    ]
  }
}
```

Pagination Response:

json

```json
{
  "success": true,
  "data": [...],
  "pagination": {
    "page": 1,
    "limit": 12,
    "total": 47,
    "totalPages": 4,
    "hasNext": true,
    "hasPrev": false
  }
}
```

7.3 Data Models (TypeScript Interfaces)

User:

typescript

```typescript
interface User {
  id: string;
  phone: string;
  email?: string;
  role: 'customer' | 'worker' | 'admin';
  createdAt: Date;
  updatedAt: Date;
}
```

Customer:

typescript

```typescript
interface Customer extends User {
  firstName: string;
  lastName: string;
  addresses: Address[];
  favoriteWorkers: string[]; // worker IDs
  totalBookings: number;
}
```

Worker:

typescript

```
interface Worker extends User {
  name: string;
  photo: string;
  category: 'electrician' | 'plumber' | 'ac-mechanic' | 'carpenter' | 'painter';
  bio: string;
  yearsOfExperience: number;
  specialties: string[];
  languages: string[];
  services: Service[];
  availability: Availability;
  location: {
    address: string;
    coordinates: { lat: number; lng: number };
  };
  rating: number;
  reviewCount: number;
  jobsCompleted: number;
  verification: {
    cnic: { verified: boolean; number: string; verifiedAt?: Date };
    police: { verified: boolean; verifiedAt?: Date };
    skill: { verified: boolean; verifiedAt?: Date };
  };
  bankDetails: {
    accountTitle: string;
    accountNumber: string;
    bankName: string;
  } | {
    jazzCashNumber: string;
  };
  status: 'pending' | 'active' | 'suspended' | 'rejected';
  portfolio: { url: string; caption?: string }[];
  memberSince: Date;
}
```

Service:

```typescript
interface Service {
  name: string;
  priceMin: number;
  priceMax: number;
  duration?: string;
}
```

Availability:

```typescript
```

```
interface Availability {
  monday: TimeSlot[];
  tuesday: TimeSlot[];
  wednesday: TimeSlot[];
  thursday: TimeSlot[];
  friday: TimeSlot[];
  saturday: TimeSlot[];
  sunday: TimeSlot[];
  emergencyAvailable: boolean;
}

interface TimeSlot {
  start: string; // HH:MM
  end: string; // HH:MM
}
```

Booking:

```
typescript
```

```
interface Booking {
  id: string;
  bookingNumber: string; // MZ12345
  customer: {
    id: string;
    name: string;
    phone: string;
  };
  worker: {
    id: string;
    name: string;
    photo: string;
    category: string;
  };
  service: {
    name: string;
    description?: string;
    photos?: string[];
  };
  scheduledDate: Date;
  scheduledTime: string;
  address: {
    full: string;
    coordinates: { lat: number; lng: number };
  };
  priceEstimate: { min: number; max: number };
  finalPrice?: number;
  platformFee: number;
  status: 'pending' | 'confirmed' | 'in-progress' | 'completed' | 'cancelled' | 'disputed';
  timeline: {
    created: Date;
    confirmed?: Date;
    workerArrived?: Date;
    workStarted?: Date;
    completed?: Date;
    cancelled?: Date;
  };
  payment: {
    method?: 'cash' | 'jazzcash' | 'easypaisa';
    status: 'pending' | 'paid';
    paidAt?: Date;
  };
  review?: Review;
  cancellationReason?: string;
  specialInstructions?: string;
  createdAt: Date;
  updatedAt: Date;
}
```

Review:

```
typescript
```

```
interface Review {
  id: string;
  booking: string; // booking ID
  worker: string; // worker ID
  customer: {
    name: string; // masked
  };
  rating: number; // 1-5
  ratings: {
    punctuality: number;
    quality: number;
    pricing: number;
    cleanliness: number;
    professionalism: number;
  };
  text: string;
  photos?: string[];
  workerResponse?: {
    text: string;
    respondedAt: Date;
  };
  createdAt: Date;
}
```

Address:

typescript

```typescript
interface Address {
  id: string;
  label: string; // "Home", "Office"
  fullAddress: string;
  apartmentNumber?: string;
  coordinates: { lat: number; lng: number };
  isDefault: boolean;
}
```

---

8. Design Guidelines

8.1 Color Palette

Primary Colors:

- Primary: 🟦 #2563EB (Blue) - Trust, reliability
- Primary Dark: 🟦 #1E40AF
- Primary Light: ⬜ #DBEAFE

Secondary Colors:

- Secondary: 🟧 #F59E0B (Orange) - Energy, action
- Secondary Dark: 🟧 #D97706
- Secondary Light: 🟨 #FEF3C7

Status Colors:

- Success: `#10B981` (Green)
- Warning: `#F59E0B` (Orange)
- Error: `#EF4444` (Red)
- Info: `#3B82F6` (Blue)

Neutral Colors:

- Gray 50: `#F9FAFB`
- Gray 100: `#F3F4F6`
- Gray 200: `#E5E7EB`
- Gray 300: `#D1D5DB`
- Gray 400: `#9CA3AF`
- Gray 500: `#6B7280`
- Gray 600: `#4B5563`
- Gray 700: `#374151`
- Gray 800: `#1F2937`
- Gray 900: `#111827`

Background:

- White: `#FFFFFF`
- Light Background: `#F9FAFB`

8.2 Typography

Font Family:

- Primary (English): `Inter, sans-serif`
- Secondary (Urdu): `Noto Nastaliq Urdu, serif`

Font Sizes:

- H1: 36px / 2.25rem (Desktop), 28px / 1.75rem (Mobile)
- H2: 30px / 1.875rem (Desktop), 24px / 1.5rem (Mobile)
- H3: 24px / 1.5rem (Desktop), 20px / 1.25rem (Mobile)
- H4: 20px / 1.25rem (Desktop), 18px / 1.125rem (Mobile)
- Body Large: 18px / 1.125rem
- Body: 16px / 1rem
- Body Small: 14px / 0.875rem
- Caption: 12px / 0.75rem

Font Weights:

- Light: 300
- Regular: 400
- Medium: 500
- Semibold: 600
- Bold: 700

Line Height:

- Tight: 1.25
- Normal: 1.5
- Relaxed: 1.75

8.3 Spacing Scale (Tailwind Default)

- 0: 0px
- 1: 4px
- 2: 8px
- 3: 12px
- 4: 16px
- 5: 20px
- 6: 24px
- 8: 32px
- 10: 40px
- 12: 48px
- 16: 64px
- 20: 80px
- 24: 96px

8.4 Border Radius

- Small: 4px
- Medium: 8px
- Large: 12px
- XL: 16px
- Full: 9999px (for pills/circles)

8.5 Shadows

```css
/* Small */
box-shadow: 0 1px 2px 0 rgb(0 0 0 / 0.05);

/* Medium */
box-shadow: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0 0 0 / 0.1);

/* Large */
box-shadow: 0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px -4px rgb(0 0 0 / 0.1);

/* XL */
box-shadow: 0 20px 25px -5px rgb(0 0 0 / 0.1), 0 8px 10px -6px rgb(0 0 0 / 0.1);
```

8.6 Buttons

Primary Button:

- Background: Primary color
- Text: White
- Hover: Primary Dark
- Padding: 12px 24px
- Border Radius: 8px
- Font Weight: 600

Secondary Button:

- Background: Transparent
- Border: 2px solid Primary
- Text: Primary
- Hover: Light primary background
- Padding: 12px 24px
- Border Radius: 8px

Outline Button:

- Background: Transparent
- Border: 1px solid Gray 300
- Text: Gray 700
- Hover: Gray 50 background

Danger Button:

- Background: Error color
- Text: White
- Hover: Darker red

Disabled State:

- Background: Gray 200
- Text: Gray 400
- Cursor: not-allowed

8.7 Cards

Default Card:

```
css
```

```
background: white;
border: 1px solid gray-200;
border-radius: 12px;
padding: 20px;
box-shadow: medium;
transition: shadow 0.2s;

&:hover {
  box-shadow: large;
}
```

8.8 Input Fields

Default Input:

```css
border: 1px solid gray-300;
border-radius: 8px;
padding: 10px 16px;
font-size: 16px;
transition: border-color 0.2s;

&:focus {
  border-color: primary;
  outline: none;
  ring: 2px solid primary-light;
}

&:error {
  border-color: error;
}

&:disabled {
  background: gray-100;
  cursor: not-allowed;
}
```

With Icon:

- Icon positioned on left
- Padding-left increased to accommodate icon

Error State:

- Red border
- Error message below in red text

8.9 Accessibility Requirements

Color Contrast:

- All text must meet WCAG AA standards (4.5:1 for normal text)
- Important UI elements: AAA standards (7:1)

Keyboard Navigation:

- All interactive elements must be keyboard accessible
- Visible focus states required
- Logical tab order

Screen Readers:

- Proper ARIA labels
- Alt text for all images
- Semantic HTML

Touch Targets:

- Minimum 44x44px on mobile
- Adequate spacing between interactive elements

Form Validation:

- Clear error messages
- Error prevention where possible
- Success feedback

---

9. Responsive Breakpoints

Mobile:

- Small: 320px - 374px
- Medium: 375px - 424px
- Large: 425px - 767px

Tablet:

- 768px - 1023px

Desktop:

- Small: 1024px - 1279px
- Medium: 1280px - 1535px
- Large: 1536px+

Tailwind Breakpoints:

```css
sm: 640px
md: 768px
lg: 1024px
xl: 1280px
2xl: 1536px
```

---

10. Performance Requirements

10.1 Loading Performance

Target Metrics:

- First Contentful Paint (FCP): < 1.8s
- Largest Contentful Paint (LCP): < 2.5s
- Time to Interactive (TTI): < 3.5s
- Cumulative Layout Shift (CLS): < 0.1
- First Input Delay (FID): < 100ms

Optimization Strategies:

- Code splitting (lazy load routes)
- Image optimization (WebP format, lazy loading)
- Font optimization (variable fonts, font-display: swap)
- Bundle size optimization (tree shaking, minification)
- Caching strategies

10.2 Image Handling

Requirements:

- All images served in WebP format (with JPEG/PNG fallback)
- Responsive images (different sizes for different viewports)
- Lazy loading (except above-the-fold images)
- Blur placeholder while loading
- Maximum file size: 500KB

Image CDN:

- Use Cloudinary or similar
- Automatic format conversion
- Automatic quality optimization
- Responsive image delivery

10.3 Bundle Size

Target:

- Initial bundle: < 200KB (gzipped)
- Total bundle: < 1MB (gzipped)

Strategies:

- Dynamic imports for routes
- Lazy load modals, charts, heavy components
- Tree-shake unused code

- Optimize dependencies

---

11. SEO Requirements

11.1 Meta Tags (Per Page)

Required on All Pages:

```html
<title>Page Title | Mazdoor Connect</title>
<meta name="description" content="...">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="canonical" href="https://mazdoorconnect.pk/current-page">

<!-- Open Graph -->
<meta property="og:title" content="...">
<meta property="og:description" content="...">
<meta property="og:image" content="...">
<meta property="og:url" content="...">
<meta property="og:type" content="website">

<!-- Twitter Card -->
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:title" content="...">
<meta name="twitter:description" content="...">
<meta name="twitter:image" content="...">
```

11.2 Structured Data (JSON-LD)

Home Page:

```json
{
  "@context": "https://schema.org",
  "@type": "LocalBusiness",
  "name": "Mazdoor Connect",
  "description": "...",
  "url": "https://mazdoorconnect.pk",
  "telephone": "+92-XXX-XXXXXXX",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Karachi",
    "addressCountry": "PK"
  }
}
```

Worker Profile:

```json
```

```json
{
  "@context": "https://schema.org",
  "@type": "Person",
  "name": "Ahmed Khan",
  "jobTitle": "AC Mechanic",
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "4.8",
    "reviewCount": "47"
  }
}
```

Review:

json

```json
{
  "@context": "https://schema.org",
  "@type": "Review",
  "author": {
    "@type": "Person",
    "name": "Sara K."
  },
  "reviewRating": {
    "@type": "Rating",
    "ratingValue": "5"
  },
  "reviewBody": "...",
  "datePublished": "2026-01-15"
}
```

11.3 Sitemap & Robots.txt

Sitemap:

- Generate dynamic sitemap.xml
- Include all public pages
- Update automatically when content changes

Robots.txt:

```
User-agent: *
Allow: /
Disallow: /customer/
Disallow: /worker/
Disallow: /admin/
Sitemap: https://mazdoorconnect.pk/sitemap.xml
```

---

12. Security Requirements

12.1 Authentication & Authorization

Implementation:

- JWT tokens stored in httpOnly cookies
- Refresh token rotation
- CSRF protection
- Rate limiting on login attempts

Role-Based Access Control:

- Customer can only access customer routes
- Worker can only access worker routes
- Admin has full access
- Middleware to verify permissions on every protected route

12.2 Input Validation

Client-Side:

- All forms validated before submission
- Use Zod for schema validation
- Sanitize user input

Server-Side:

- Never trust client-side validation
- Re-validate all inputs on server
- Sanitize to prevent XSS/SQL injection

12.3 Data Protection

Sensitive Data:

- Phone numbers: Mask in UI (03XX-XXX-1234)
- CNIC: Never display full number in frontend
- Bank details: Masked (last 4 digits only)

HTTPS:

- Enforce HTTPS in production
- HSTS headers

Content Security Policy:

```
default-src 'self';
script-src 'self' 'unsafe-inline' https://maps.googleapis.com;
img-src 'self' data: https:;
```

---

13. Testing Requirements

13.1 Unit Testing

Framework: Jest + React Testing Library

Coverage Target: 80%+

Test:

- All utility functions
- Form validation logic
- Business logic
- Custom hooks

13.2 Integration Testing

Test:

- User flows (signup, booking, review)
- API integration
- Form submissions
- Navigation

13.3 E2E Testing

Framework: Playwright or Cypress

Critical Flows:

- User signup/login
- Worker profile creation
- Service booking (end-to-end)
- Review submission
- Admin verification workflow

13.4 Accessibility Testing

Tools:

- axe DevTools
- Lighthouse
- Manual keyboard navigation testing
- Screen reader testing (NVDA/JAWS)

---

14. Internationalization (i18n)

14.1 Language Support

Phase 1:

- English (primary)

- Urdu (secondary)

Implementation:

- Use next-i18next or react-i18next
- Language switcher in header
- URL structure: /en/... or /ur/...
- Detect browser language on first visit

14.2 RTL Support

Requirements:

- Full RTL layout for Urdu
- Mirror layouts (sidebar on right, etc.)
- RTL-friendly icons
- Test all UI components in both directions

Implementation:

```css
html[dir="rtl"] {
  /* RTL-specific styles */
}
```

14.3 Currency & Number Formatting

Currency:

- Always display as "Rs. X,XXX"
- Use Intl.NumberFormat for proper formatting

Dates:

- Format based on locale
- English: Jan 15, 2026
- Urdu: ۱۵ جنوری ۲۰۲۶

---

15. Analytics & Tracking

15.1 Events to Track

User Events:

- Page views
- Signup/Login
- Worker profile views
- Category browsing
- Search queries

- Filter usage
- Booking initiated
- Booking completed
- Review submitted
- Favorites added

Performance Events:

- Page load time
- API response time
- Error occurrences

Business Metrics:

- Conversion rate (visitor → booking)
- Average booking value
- Customer lifetime value

15.2 Analytics Tools

Google Analytics 4:

- Track all user interactions
- E-commerce tracking for bookings
- Custom events

Hotjar or Microsoft Clarity:

- Session recordings
- Heatmaps
- User behavior analysis

Custom Dashboard:

- Real-time analytics in admin panel
- Track key business metrics

---

16. Third-Party Integrations

16.1 Google Maps API

Usage:

- Worker location display
- Customer address selection
- Distance calculation
- Navigation (directions to customer)

Features Needed:

- Places Autocomplete (address search)
- Geocoding (address → coordinates)
- Distance Matrix (calculate distance)
- Maps JavaScript API (display map)

API Key:

- Restrict to domain in production
- Set usage limits

16.2 SMS Gateway

Provider: Twilio or local Pakistani provider (e.g., SMS.to, Unifonic)

Use Cases:

- OTP verification
- Booking confirmations
- Worker notifications
- Status updates
- Payment confirmations

Template Examples:

```
OTP: Your Mazdoor Connect verification code is {code}. Valid for 5 minutes.

Booking Confirmed: Your booking #MZ12345 with Ahmed is confirmed for {date} at {time}. Call: {phone}

Job Request: New job request! Service: {service}, Location: {area}. Accept/Reject: {link}
```

16.3 Payment Gateways

Phase 1 (Manual):

- Worker shares JazzCash/EasyPaisa number
- Customer transfers directly
- Worker confirms payment

Phase 2 (Integrated):

- JazzCash API
- EasyPaisa API
- Credit/Debit card (optional)

Requirements:

- PCI compliance
- Secure payment flow
- Transaction tracking
- Refund capability

16.4 Firebase Services

Authentication:

- Phone number OTP authentication
- Session management

Cloud Storage:

- Worker photos
- Portfolio images
- Customer-uploaded photos
- Review images

Cloud Messaging:

- Push notifications (Phase 2)
- Worker job alerts
- Customer updates

Firestore (Optional):

- Real-time chat (Phase 2)
- Notifications

16.5 Email Service

Provider: SendGrid, AWS SES, or Mailgun

Email Types:

- Welcome email
- Booking confirmations
- Payment receipts
- Password reset (if using email auth later)
- Weekly summaries
- Promotional emails

Templates:

- Responsive HTML templates
- Plain text fallback
- Unsubscribe link

---

17. Error Handling & Logging

17.1 Error Handling Strategy

User-Facing Errors:

- Clear, actionable error messages
- Avoid technical jargon
- Suggest next steps

Example:

```
❌ Bad: "500 Internal Server Error"
✅ Good: "Something went wrong. Please try again or contact support if the issue persists."

❌ Bad: "Validation failed"
✅ Good: "Please enter a valid phone number (03XX-XXX-XXXX)"
```

Error UI Components:

- Toast notifications (temporary errors)
- Inline form errors
- Full-page error state (for critical failures)
- Error boundaries (catch React errors)

17.2 Logging

Client-Side:

- Console errors in development
- Error tracking service in production (Sentry)

Server-Side:

- Log all API errors
- Track failed requests
- Monitor performance

What to Log:

- API errors
- Authentication failures
- Payment failures
- User actions (for debugging)
- Performance metrics

What NOT to Log:

- Passwords
- Full CNIC numbers
- Payment card details
- Personal phone numbers (mask them)

17.3 Error Monitoring

Tool: Sentry or similar

Features:

- Real-time error alerts
- Error grouping
- Stack traces
- User context (what they were doing)
- Release tracking

---

18. Deployment & DevOps

18.1 Environments

Development:

- Local development
- Hot reload
- Mock APIs (optional)

Staging:

- Production-like environment
- For testing before release
- Uses staging backend API

Production:

- Live site
- Optimized builds
- CDN for static assets

18.2 Deployment Platform

Recommended: Vercel

Why:

- Optimized for Next.js
- Automatic deployments from Git
- Preview deployments for PRs
- Edge network (fast globally)
- Free tier available

Alternative: Netlify, AWS Amplify

18.3 CI/CD Pipeline

On Every Commit:

- Run linting (ESLint)
- Run type checking (TypeScript)
- Run tests (Jest)
- Build check

On Pull Request:

- Run all above
- Deploy preview environment
- Run E2E tests

On Merge to Main:

- Deploy to staging
- Run smoke tests
- Manual approval
- Deploy to production

18.4 Environment Variables

Required Variables:

```env
# API
NEXT_PUBLIC_API_URL=https://api.mazdoorconnect.pk
NEXT_PUBLIC_API_KEY=...

# Firebase
NEXT_PUBLIC_FIREBASE_API_KEY=...
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=...
NEXT_PUBLIC_FIREBASE_PROJECT_ID=...

# Google Maps
NEXT_PUBLIC_GOOGLE_MAPS_API_KEY=...

# Analytics
NEXT_PUBLIC_GA_MEASUREMENT_ID=...

# Environment
NEXT_PUBLIC_ENVIRONMENT=production
```

Security:

- Never commit .env files
- Use platform's environment variable manager
- Rotate keys regularly

---

19. Browser Storage Strategy

19.1 LocalStorage

Use For:

- User preferences (language, theme)
- Draft form data (auto-save bookings)
- Recently viewed workers
- Search history

Do NOT Store:

- Authentication tokens (use cookies)
- Sensitive user data

19.2 Cookies

Use For:

- Authentication tokens (httpOnly, secure)
- Session management

Settings:

```javascript
{
  httpOnly: true,
  secure: true, // production only
  sameSite: 'strict',
  maxAge: 7 * 24 * 60 * 60 // 7 days
}
```

19.3 Session Storage

Use For:

- Multi-step form progress (booking flow)
- Temporary filters/search state

---

20. Progressive Web App (PWA) - Phase 2

20.1 PWA Features

Installability:

- Web app manifest
- Install prompt
- Home screen icon

Offline Support:

- Service worker
- Cache critical assets
- Offline page

Push Notifications:

- Job notifications for workers
- Booking updates for customers
- Marketing messages (opt-in)

20.2 App Manifest

json

```json
{
  "name": "Mazdoor Connect",
  "short_name": "Mazdoor",
  "description": "Find verified workers for home repairs",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#2563EB",
  "icons": [
    {
      "src": "/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

---

21. Development Best Practices

21.1 Code Structure

Folder Structure:

```
src/
├── app/                    # Next.js app directory
│   ├── (auth)/
│   │   ├── login/
│   │   └── signup/
│   ├── (customer)/
│   │   ├── dashboard/
│   │   └── bookings/
│   ├── (worker)/
│   │   ├── dashboard/
│   │   └── profile/
│   ├── (admin)/
│   ├── categories/
│   ├── workers/
│   └── layout.tsx
├── components/
│   ├── ui/                 # Reusable UI components
│   │   ├── Button.tsx
│   │   ├── Input.tsx
│   │   ├── Modal.tsx
│   │   └── ...
│   ├── features/           # Feature-specific components
│   │   ├── worker/
│   │   ├── booking/
│   │   └── review/
│   └── layout/             # Layout components
│       ├── Header.tsx
│       ├── Footer.tsx
│       └── Sidebar.tsx
├── lib/                    # Utility functions
│   ├── api.ts              # API client
│   ├── auth.ts             # Auth helpers
│   ├── utils.ts            # General utils
│   └── constants.ts        # Constants
├── hooks/                  # Custom React hooks
│   ├── useAuth.ts
│   ├── useBooking.ts
│   └── useWorkers.ts
├── types/                  # TypeScript types
│   ├── user.ts
│   ├── worker.ts
│   ├── booking.ts
│   └── ...
├── contexts/               # React contexts
│   ├── AuthContext.tsx
│   └── ThemeContext.tsx
├── styles/
│   └── globals.css
└── public/
    ├── images/
    ├── icons/
    └── ...
```

21.2 Naming Conventions

Files:

- Components: PascalCase (e.g., `WorkerCard.tsx`)
- Utilities: camelCase (e.g., `formatPrice.ts`)
- Hooks: camelCase with "use" prefix (e.g., `useAuth.ts`)

Variables:

- camelCase for variables and functions
- PascalCase for components and classes
- UPPER_SNAKE_CASE for constants

CSS Classes:

- Use Tailwind utility classes primarily
- Custom classes: kebab-case (e.g., `custom-button`)

21.3 Code Quality

Linting:

- ESLint with recommended configs
- Prettier for formatting
- Pre-commit hooks (Husky + lint-staged)

TypeScript:

- Strict mode enabled
- No implicit any
- All props typed

Comments:

- Use JSDoc for complex functions
- Explain "why" not "what"
- Keep comments up to date

21.4 Git Workflow

Branches:

- `main` - Production
- `staging` - Staging environment
- `develop` - Development
- `feature/feature-name` - Feature branches
- `bugfix/bug-description` - Bug fixes

Commit Messages:

```
feat: Add worker booking flow
fix: Resolve rating display issue
refactor: Optimize worker card component
docs: Update API integration guide
style: Format code with Prettier
test: Add unit tests for booking logic
```

Pull Requests:

- Descriptive title and description
- Link to issue/task
- Screenshots for UI changes
- Request review before merging

---

22. Documentation Requirements

22.1 Code Documentation

Required:

- README.md with setup instructions
- Component documentation (Storybook recommended)
- API integration guide
- Environment setup guide
- Deployment guide

22.2 User Documentation (Phase 2)

For Customers:

- How to book a worker
- How to leave reviews
- How to manage bookings
- FAQ

For Workers:

- How to create profile
- How to accept jobs
- How to manage earnings
- Best practices guide

---

23. Deliverables & Timeline

23.1 Phase 1 - MVP (6-8 Weeks)

Week 1-2: Foundation

- Project setup
- Design system implementation
- Reusable components
- Authentication flow

Week 3-4: Customer Features

- Home page
- Category pages
- Worker profiles
- Booking flow
- Customer dashboard

Week 5-6: Worker Features

- Worker dashboard
- Profile management
- Job management
- Earnings tracking

Week 7: Admin Features

- Admin dashboard
- Worker verification
- Booking management

Week 8: Testing & Launch

- Bug fixes
- Performance optimization
- User testing
- Deployment

23.2 Phase 2 - Enhancements (4-6 Weeks)

Features:

- In-app chat
- Push notifications
- Payment gateway integration
- Advanced analytics
- PWA features
- Mobile apps (React Native)

---

24. Budget Estimate

24.1 Development Costs

If Hiring Agency/Freelancers:

Item	Hours	Rate ($/hr)	Total
UI/UX Design	80	$30	$2,400
Frontend Development	320	$40	$12,800
Testing & QA	40	$25	$1,000
Project Management	40	$35	$1,400
Total	**480**		**$17,600**

Pakistan Rates (Estimated):

- Total cost: Rs. 1,500,000 - 2,500,000 for complete MVP

24.2 Ongoing Costs (Monthly)

Service	Cost (Monthly)
Hosting (Vercel Pro)	$20 (Rs. 5,500)
Firebase (Blaze)	$50-100 (Rs. 14,000-28,000)
Google Maps API	$50-200 (Rs. 14,000-55,000)
SMS Gateway	$100-300 (Rs. 28,000-83,000)
Error Tracking (Sentry)	$29 (Rs. 8,000)
Domain	$15/year (Rs. 4,000/year)
Total	**$250-700/month**

---

25. Support & Maintenance

25.1 Post-Launch Support

First 3 Months:

- Bug fixing (highest priority)
- Performance monitoring
- User feedback implementation
- Security patches

Ongoing:

- Monthly updates

- Feature additions based on user feedback
- Performance optimization
- Security updates

25.2 Maintenance Checklist

Weekly:

- Monitor error logs
- Check performance metrics
- Review user feedback
- Address critical bugs

Monthly:

- Update dependencies
- Security audit
- Performance review
- Analytics review

Quarterly:

- Major feature releases
- User testing
- Competitive analysis
- Technology stack review

---

26. Success Metrics

26.1 Technical Metrics

Performance:

- Lighthouse score: 90+ (all categories)
- Page load time: < 2s
- Time to Interactive: < 3s
- Zero critical errors in production

Quality:

- Code coverage: 80%+
- Zero accessibility violations (critical)
- Mobile responsiveness: 100%

26.2 Business Metrics

User Engagement:

- Conversion rate: 5%+ (visitor → signup)

- Booking completion rate: 70%+
- User retention: 40%+ (30-day)
- Average session duration: 3+ minutes

Worker Metrics:

- Worker activation rate: 80%+
- Jobs per worker per week: 3+
- Worker retention: 60%+ (monthly)

---

27. Contact & Communication

27.1 Project Communication

Primary Contact:

- Name: [Your Name]
- Email: [Your Email]
- Phone: [Your Phone]

Preferred Communication:

- Slack/Discord for daily updates
- Weekly video calls for progress review
- Email for formal communications
- GitHub issues for bug tracking

27.2 Feedback & Reviews

Regular Reviews:

- Daily standups (15 min)
- Weekly demos (30 min)
- Bi-weekly sprint reviews (1 hour)

Deliverable Reviews:

- Review designs before development
- Review major features before deployment
- Approve before production deployment

---

28. Questions for Development Team

Before starting, please clarify:

Technical:

1. Do you have experience with Next.js 14+ App Router?
2. Are you comfortable with TypeScript?

3. Have you implemented Urdu/RTL layouts before?
4. Experience with Google Maps API integration?
5. Experience with payment gateway integrations?

Design: 6. Will you provide design files or work from this spec? 7. Do you have a UI/UX designer on the team? 8. Can you provide design mockups before development?

Process: 9. What is your development methodology? (Agile/Scrum/Kanban) 10. How do you handle change requests during development? 11. What is your testing process? 12. How do you handle post-launch support?

Timeline: 13. Can you commit to 6-8 week timeline for MVP? 14. What is your availability (hours per day)? 15. How do you handle delays or blockers?

---

29. Final Notes

29.1 Priorities

Must-Have (MVP):

- Working authentication
- Worker browsing and profiles
- Booking flow (even if partially manual)
- Customer and worker dashboards
- Mobile-responsive design
- Urdu language support

Nice-to-Have (Can be Phase 2):

- Advanced filters
- In-app chat
- Payment integration
- Push notifications
- Analytics dashboard
- Mobile apps

29.2 Flexibility

This specification is:

- A comprehensive guide, not a rigid requirement
- Open to suggestions and improvements
- Subject to change based on user feedback
- Focused on MVP first, then iteration

We encourage:

- Developer input on technical decisions
- UX improvements
- Performance optimizations
- Better approaches to solving problems

30. Appendix

30.1 Useful Resources

Design Inspiration:

- TaskRabbit (US)
- Urban Company (India)
- Thumbtack (US)
- Bark (UK)

Technical Documentation:

- Next.js: https://nextjs.org/docs
- Tailwind CSS: https://tailwindcss.com/docs
- Shadcn/ui: https://ui.shadcn.com
- React Hook Form: https://react-hook-form.com
- Google Maps API: https://developers.google.com/maps/documentation

Pakistani Market Research:

- Pakistan Telecommunication Authority (PTA) reports
- State Bank of Pakistan digital payments data
- Local competitor analysis

30.2 Glossary

Terms Used:

- **Worker**: Skilled tradesperson (electrician, plumber, etc.)
- **Customer**: Homeowner booking services
- **Booking**: Service appointment
- **Category**: Type of service (electrician, plumber, etc.)
- **Platform Fee**: Commission charged per booking
- **Verification**: Background check process
- **Portfolio**: Worker's previous work photos
- **Rating**: 1-5 star customer review
- **Availability**: Worker's schedule

Contact for Clarifications

If you have any questions about this specification:

Email: [your-email@example.com] **Phone:** [Your Phone Number] **Available:** Monday-Friday, 9 AM - 6 PM PKT

Document Version: 1.0 **Last Updated:** January 2026 **Prepared By:** [Your Name] **For:** Antigrativity Development Team

---

END OF SPECIFICATION

Next Steps:

1. Review this specification thoroughly
2. Ask any questions or clarifications needed
3. Provide time and cost estimate
4. Create detailed project plan
5. Begin development upon approval

Thank you for your interest in building Mazdoor Connect. We're excited to work together on this impactful project!