**Phone Verification Implementation Prompts**

**Phase 1: Backend Infrastructure**

**Prompt 1: Database Schema Design**

Create a database schema for phone verification that includes:

- User phone numbers (with country code support)

- Verification codes with expiration timestamps

- Attempt tracking and rate limiting

- Status tracking (pending, verified, expired, failed)

- Support for both SMS and fallback methods

Include migration scripts and consider security best practices for storing sensitive data.

**Prompt 2: SMS Service Integration**

Implement SMS service integration using [Twilio/AWS SNS/your preferred provider] that:

- Generates 6-digit verification codes

- Sends SMS with branded messaging

- Handles international phone numbers

- Implements rate limiting (max 3 attempts per 10 minutes)

- Logs delivery status and failures

- Includes cost optimization strategies

Provide error handling for failed SMS deliveries and fallback options.

**Prompt 3: Verification API Endpoints**

Create RESTful API endpoints for phone verification:

POST /api/verification/send

- Validates phone number format

- Generates and sends verification code

- Returns success/error status

- Implements rate limiting

POST /api/verification/verify

- Validates submitted code

- Updates user verification status

- Handles expired/invalid codes

- Returns JWT token on success

POST /api/verification/resend

- Resends verification code

- Tracks resend attempts

- Implements progressive delays

Include comprehensive error responses and status codes.

**Phase 2: Frontend Components**

**Prompt 4: Phone Input Component**

Design a React component for phone number input that:

- Uses domestic phone number formatting (react-phone-number-input)

- Validates phone numbers in real-time

- Provides clear error messaging

- Supports keyboard navigation

- Is screen reader friendly

- Has a clean, modern design consistent with the Mingus theme

Include loading states and disabled states.

**Prompt 5: Verification Code Input**

Create a verification code input component that:

- Has 6 individual input fields for digits

- Auto-advances to next field on input

- Allows paste functionality (splits pasted code across fields)

- Auto-submits when all fields are filled

- Shows loading state during verification

- Handles backspace navigation

- Is accessible and mobile-friendly

- Includes haptic feedback on mobile

Add visual feedback for success/error states.

**Prompt 6: Verification Flow Container**

Build a verification flow container component that:

- Manages state between phone input and code verification steps

- Shows progress indicators

- Handles timer countdown for resend functionality

- Displays appropriate success/error messages

- Includes smooth transitions between steps

- Supports going back to edit phone number

- Handles network errors gracefully

Include loading skeletons and empty states.

**Phase 3: User Experience Enhancements**

**Prompt 7: Resend Functionality**

Implement smart resend functionality that:

- Shows countdown timer (starts at 60 seconds)

- Increases delay progressively (60s, 120s, 300s)

- Limits total resend attempts (max 3 per session)

- Offers alternative contact methods after failed attempts

- Tracks and displays attempt history

- Provides different messaging for each attempt

- Includes option to change phone number

Add analytics tracking for resend patterns.

**Prompt 8: Support Integration**

Create support contact integration that:

- Shows contextual help based on verification step

- Includes "Contact Support" button with pre-filled context

- Opens support chat/email with verification details

- Provides common troubleshooting tips in a help modal

- Includes FAQ section for verification issues

- Offers phone support callback option

- Tracks support interaction analytics

Include escalation paths for urgent issues.

**Prompt 9: Error Handling & Feedback**

Design comprehensive error handling that:

- Shows specific error messages for different failure types

- Provides actionable solutions for each error

- Includes retry mechanisms with exponential backoff

- Offers alternative verification methods (email, authenticator)

- Shows network status and connectivity issues

- Includes graceful degradation for slow connections

- Provides offline mode messaging

Add user-friendly language and avoid technical jargon.

**Phase 4: Security & Compliance**

**Prompt 10: Security Implementation**

Implement security measures for phone verification:

- Rate limiting on both frontend and backend

- CAPTCHA integration after multiple failed attempts

- Protection against SIM swapping attacks

- Secure code generation (cryptographically random)

- Code expiration and single-use enforcement

- Input sanitization and validation

- Audit logging for security events

- GDPR/privacy compliance for phone number storage

Include monitoring and alerting for suspicious activities.

**Prompt 11: Analytics & Monitoring**

Set up analytics and monitoring for verification flow:

- Track conversion rates at each step

- Monitor SMS delivery success rates

- Measure verification completion time

- Track support contact frequency and reasons

- Monitor error rates and types

- Set up alerts for system issues

- Create dashboard for verification metrics

- A/B test different UI variations

Include user journey mapping and drop-off analysis.

**Phase 5: Testing & Quality Assurance**

**Prompt 12: Test Suite Development**

Create comprehensive test suite covering:

- Unit tests for all verification logic

- Integration tests for SMS service

- E2E tests for complete verification flow

- Load testing for high-volume scenarios

- Security testing for common vulnerabilities

- Accessibility testing (WCAG compliance)

- Mobile responsiveness testing

- Cross-browser compatibility testing

Include test data management and mock services.

**Prompt 13: User Acceptance Testing**

Design user acceptance testing scenarios:

- Happy path verification flow

- Error scenarios and recovery

- Resend functionality testing

- Support contact testing

- Accessibility testing with screen readers

- Mobile device testing (iOS/Android)

- International phone number testing

- Edge case handling (network issues, timeouts)

Include user feedback collection and iteration planning.

**Implementation Notes**

**Priority Order:**

1. **Phase 1 (Backend)** - Core infrastructure first
2. **Phase 2 (Frontend)** - Basic UI components
3. **Phase 3 (UX)** - Enhanced user experience
4. **Phase 4 (Security)** - Security hardening
5. **Phase 5 (Testing)** - Quality assurance

**Key Success Metrics:**

* **Verification completion rate > 85%**
* **SMS delivery success rate > 95%**
* **Support contact rate < 5%**
* **Average verification time < 2 minutes**
* **User satisfaction score > 4.0/5.0**

**Technology Recommendations:**

* **SMS Provider**: Twilio (reliability) or AWS SNS (cost)
* **Phone Input**: react-phone-number-input
* **Styling**: Tailwind CSS or styled-components
* **State Management**: React Context or Zustand
* **Form Validation**: react-hook-form + yup
* **Analytics**: PostHog or Mixpanel