Build a modern, responsive Aptitude Preparation Platform (MVP) with the following specifications:

PROJECT OVERVIEW:
Create a web application for Indian students to practice and take aptitude tests for placement preparation. The platform should provide a real exam-like experience with AI-powered analytics.

TECH STACK (SEO & Performance Optimized):
- Frontend: React 18+ with TypeScript
- Framework: Next.js 14 (App Router) - for SSR, SEO optimization, and fast performance
- Styling: Tailwind CSS + shadcn/ui components
- State Management: Zustand (lightweight, fast)
- Form Handling: React Hook Form + Zod validation
- Backend: Supabase (PostgreSQL database, Auth, Storage, Edge Functions)
- Authentication: Supabase Auth with email/password and OAuth (Google)
- Routing: Next.js App Router with dynamic routes
- Icons: Lucide React
- Charts: Recharts for analytics dashboard
- Deployment: Vercel (optimal for Next.js, automatic SEO optimization)

SEO OPTIMIZATION REQUIREMENTS:
- Implement Next.js Metadata API for all pages
- Add structured data (JSON-LD) for educational content
- Generate dynamic sitemap.xml and robots.txt
- Implement Open Graph and Twitter Card meta tags
- Use semantic HTML5 tags throughout
- Add proper heading hierarchy (H1, H2, H3)
- Implement lazy loading for images
- Create descriptive URLs with keywords (e.g., /practice/quantitative-aptitude)
- Add canonical URLs to prevent duplicate content
- Implement breadcrumb navigation with schema markup

CORE FEATURES TO BUILD:

═══

1. HOMEPAGE (/)
Landing page with:
- Hero section with clear value proposition
- Two prominent CTAs: "Start Practice" and "Take Test"
- Feature highlights (cards showing key benefits)
- Statistics counter (total questions, users, companies covered)
- Testimonials section (placeholder for now)

- Footer with links

Design: Modern, clean, gradient backgrounds, glassmorphism cards
Colors: Primary: #6366f1 (Indigo), Secondary: #8b5cf6 (Purple), Accent: #10b981 (Green)

═══

2. AUTHENTICATION (/login, /signup)
- Clean auth forms with email/password
- Google OAuth integration via Supabase
- Form validation with proper error messages
- Password strength indicator
- "Remember me" and "Forgot password" options
- Redirect to dashboard after successful login

═══

3. PRACTICE MODE (/practice)

A. Topic Selection Screen:
Display 5 main topics as interactive cards:
1. Quantitative Aptitude (Arithmetic, Percentages, Profit & Loss)
2. Logical Reasoning (Puzzles, Series, Blood Relations)
3. Verbal Ability (Grammar, Vocabulary, Comprehension)
4. Data Interpretation (Tables, Graphs, Charts)
5. Problem Solving (Mixed logical problems)

Each card shows:
- Topic name and icon
- Number of questions available
- Your accuracy percentage (from past attempts)
- Color-coded difficulty indicator
- "Start Practice" button

B. Practice Configuration Modal:
When topic is selected, show modal with:
- Number of questions: Slider (5, 10, 15, 20, 25)
- Difficulty: Dropdown (Easy, Medium, Hard, Mixed)
- Time limit: Toggle (Timed/Untimed) + time input
- "Start Practice" button

C. Practice Interface:

- Clean question display with:
  * Question number indicator (e.g., "Question 3 of 15")
  * Timer (if enabled) - prominent display
  * Question text (support for LaTeX if math formulas)
  * Four option buttons (A, B, C, D) - single select
  * "Submit Answer" button
  * "Skip" button (marks as skipped)
  * Progress bar showing completed questions

- Immediate Feedback After Submission:
  * Green checkmark for correct, red X for wrong
  * Show correct answer if wrong
  * Detailed explanation (expandable section)
  * "Next Question" button
  * Keep score visible: "12/15 correct"

D. Practice Summary Page:
After completing practice session:
- Overall score (percentage and fraction)
- Time taken (if timed)
- Accuracy breakdown
- List of questions with your answers vs correct answers
- "Review Mistakes" and "Practice Again" buttons
- "Back to Topics" button

═══

4. TEST MODE (/test)

A. Test Type Selection:
Three cards to choose from:
1. All Questions (Mixed Topics)
2. PYQs Only (Company-Specific)
3. Custom Test (Upload Document) - show "Coming Soon" badge for MVP

B. Test Configuration Screen:

For "All Questions":
- Number of questions: 30, 60, 90 (radio buttons)
- Difficulty: Easy, Medium, Hard, Mixed (dropdown)
- Time limit: Auto-calculate based on questions (1.5 min per Q) or Custom
- "Proceed to Instructions" button

For "PYQs Only":
- Company selector: Dropdown with 5 companies (TCS, Infosys, Wipro, Accenture, Cognizant)
- Year filter: 2024, 2023, 2022, All Years (checkboxes)
- Show expected question count
- "Proceed to Instructions" button

C. Pre-Test Instructions Screen:
- Scrollable instruction box with:
  * Test details (questions, time, marking scheme)
  * Important guidelines (bullet points)
  * Question palette legend with color codes
  * Navigation rules

- System check indicators (green checkmarks):
  ✓ Internet Connection
  ✓ Browser Compatibility
  ✓ Screen Resolution

- Checkbox: "I have read and understood all instructions"
- "Start Test" button (disabled until checkbox checked)

D. Test Interface (Full-Screen Mode):
Header:
- Platform logo (small)
- Test name
- Timer (countdown, turns red at <5 mins)
- "Submit Test" button (with confirmation modal)

Main Layout (Split View):
Left Side (70%):
- Question number indicator
- Question text (with image support if needed)
- Four option buttons (A, B, C, D) - radio buttons
- Action buttons row:
  * "Clear Response" button
  * "Mark for Review" toggle (yellow flag icon)
  * "Save & Next" button (primary)
  * "Previous" button (secondary)

Right Side (30%) - Question Palette:
- Numbered grid of all questions (6 columns)
- Color coding:
  * Green: Answered
  * Yellow: Marked for review

  * Red: Not answered but visited
  * White: Not visited
- Section selector (if multiple sections)
- "Submit Test" button (secondary)

Bottom Navigation:
- "< Previous" and "Save & Next >" buttons (always visible)

E. Test Submission Flow:
- Click "Submit Test" → Confirmation Modal:
  * Show answered/unanswered/marked count
  * "Are you sure you want to submit?" message
  * "Review Again" and "Submit" buttons

- After submission → Redirect to Results

F. Test Results & Analytics Dashboard (/test/results/[testId]):

Layout with tabs:
1. Overview Tab:
   - Score card (large, centered):
     * Total score (e.g., 68/100)
     * Percentage and percentile
     * Time taken
     * Rank indicator (if leaderboard exists)

   - Section-wise performance table:
     | Section | Attempted | Correct | Incorrect | Accuracy | Time |

   - Performance comparison chart (bar chart):
     * Your score vs Average score vs Top score

2. Detailed Analysis Tab:
   - Topic-wise accuracy (pie chart or radar chart)
   - Time distribution across questions (line graph)
   - Difficulty-wise breakdown (bar chart)

   - AI Insights Section (cards):
     * Strengths: "You excel in..." (green card)
     * Weaknesses: "Needs improvement in..." (red card)
     * Time Management: "You spent most time on..." (blue card)
     * Recommendations: "Practice these topics..." (purple card)

3. Solutions Tab:

- Scrollable list of all questions with:
  * Question number and text
  * Your answer (highlighted in red if wrong, green if correct)
  * Correct answer
  * Explanation (expandable)
  * "Practice Similar" button (links to practice mode)

- Filter options:
  * Show All / Incorrect Only / Marked for Review / Skipped

Action Buttons (bottom):
- "Download Report" (PDF)
- "Retake Test"
- "Back to Dashboard"

═══

5. USER DASHBOARD (/dashboard)

Sidebar Navigation:
- Dashboard (home icon)
- Practice (brain icon)
- Take Test (clipboard icon)
- My Results (chart icon)
- Profile (user icon)
- Logout

Main Dashboard Content:
- Welcome message: "Welcome back, [Name]!"

- Quick Stats Cards (4 cards in a row):
  1. Total Tests Taken
  2. Average Score
  3. Questions Practiced
  4. Current Streak (days)

- Recent Activity Section:
  - List of last 5 tests/practice sessions with:
    * Date and time
    * Type (Practice/Test)
    * Topic/Test name
    * Score
    * "View Details" link

- Performance Over Time:
  - Line chart showing score trend (last 10 attempts)
  - X-axis: Date, Y-axis: Score percentage

- Recommended Actions:
  - Cards suggesting:
    * "Practice [Weak Topic]"
    * "Take a full-length mock test"
    * "Review your mistakes from last test"

- Quick Access Buttons:
  - "Start Practice"
  - "Take New Test"

═══
═══

6. PROFILE PAGE (/profile)
- Profile information form:
  * Name, Email (read-only)
  * College/Institution
  * Graduation Year
  * Target Companies (multi-select)
  * Phone Number
  * "Save Changes" button

- Statistics section showing:
  * Member since date
  * Total time spent practicing
  * Achievements/badges (placeholder for now)

- Settings section:
  * Email notifications toggle
  * Dark mode toggle (implement dark mode support)
  * Language preference (English only for MVP)

- Danger zone:
  * "Delete Account" button (with confirmation)

═══
═══

DATABASE SCHEMA (Supabase/PostgreSQL):

Table: users (extends Supabase auth.users)
- id (uuid, primary key)
- email (text)
- full_name (text)
- college (text)
- graduation_year (integer)
- target_companies (text[])
- phone (text)
- created_at (timestamp)
- updated_at (timestamp)

Table: topics
- id (uuid, primary key)
- name (text) - e.g., "Quantitative Aptitude"
- slug (text) - e.g., "quantitative-aptitude"
- description (text)
- icon (text) - icon name for UI
- total_questions (integer)
- created_at (timestamp)

Table: questions
- id (uuid, primary key)
- topic_id (uuid, foreign key → topics.id)
- question_text (text)
- option_a (text)
- option_b (text)
- option_c (text)
- option_d (text)
- correct_answer (text) - 'A', 'B', 'C', or 'D'
- explanation (text)
- difficulty (text) - 'easy', 'medium', 'hard'
- is_pyq (boolean)
- company_name (text) - if is_pyq = true
- year (integer) - if is_pyq = true
- image_url (text, nullable) - for questions with images
- created_at (timestamp)

Table: practice_sessions
- id (uuid, primary key)
- user_id (uuid, foreign key → users.id)
- topic_id (uuid, foreign key → topics.id)
- total_questions (integer)
- correct_answers (integer)

- time_taken (integer) - in seconds
- difficulty (text)
- is_timed (boolean)
- created_at (timestamp)

Table: practice_attempts
- id (uuid, primary key)
- session_id (uuid, foreign key → practice_sessions.id)
- question_id (uuid, foreign key → questions.id)
- user_answer (text) - 'A', 'B', 'C', 'D', or NULL if skipped
- is_correct (boolean)
- time_taken (integer) - in seconds
- created_at (timestamp)

Table: tests
- id (uuid, primary key)
- user_id (uuid, foreign key → users.id)
- test_type (text) - 'mixed', 'pyq', 'custom'
- company_name (text, nullable) - for PYQ tests
- total_questions (integer)
- correct_answers (integer)
- incorrect_answers (integer)
- skipped_questions (integer)
- score (numeric) - percentage
- time_limit (integer) - in seconds
- time_taken (integer) - in seconds
- started_at (timestamp)
- submitted_at (timestamp)
- created_at (timestamp)

Table: test_attempts
- id (uuid, primary key)
- test_id (uuid, foreign key → tests.id)
- question_id (uuid, foreign key → questions.id)
- user_answer (text) - 'A', 'B', 'C', 'D', or NULL
- is_correct (boolean)
- is_marked_for_review (boolean)
- time_taken (integer) - in seconds
- order (integer) - question order in test
- created_at (timestamp)

SAMPLE DATA REQUIREMENTS:
Create seed data with:
- 5 topics with proper details
- 100 questions per topic (total 500 questions)
- Mix of difficulties (30% easy, 50% medium, 20% hard)
- At least 50 PYQ questions from 5 companies:
  * TCS: 20 questions (mix of all topics)
  * Infosys: 15 questions
  * Wipro: 10 questions
  * Accenture: 10 questions
  * Cognizant: 5 questions

Question examples should be realistic aptitude questions with proper explanations.

═══

API ROUTES (Next.js API Routes / Supabase Edge Functions):
- /api/auth/* - Authentication endpoints (handled by Supabase)
- /api/topics - GET all topics
- /api/questions/[topicId] - GET questions by topic
- /api/practice/start - POST to create practice session
- /api/practice/submit - POST to submit practice attempt
- /api/test/start - POST to create test
- /api/test/submit - POST to submit test
- /api/results/[testId] - GET test results
- /api/dashboard/stats - GET user statistics
- /api/user/profile - GET/PUT user profile

═══

KEY FUNCTIONAL REQUIREMENTS:

1. RESPONSIVE DESIGN:
   - Mobile-first approach
   - Breakpoints: sm (640px), md (768px), lg (1024px), xl (1280px)
   - Test interface should work on tablets (landscape mode recommended)
   - Touch-friendly buttons (minimum 44x44px)

2. PERFORMANCE:
   - Lazy load components where possible
   - Image optimization with Next.js Image component
   - Implement skeleton loaders for data fetching

- Target Lighthouse score: 90+ on all metrics

3. ACCESSIBILITY:
- Proper ARIA labels
- Keyboard navigation support
- Focus indicators on interactive elements
- Color contrast ratio WCAG AA compliant
- Alt text for all images

4. STATE MANAGEMENT:
- Use Zustand for:
  * User authentication state
  * Current test/practice session data
  * Timer state
  * Question navigation state

5. ERROR HANDLING:
- Graceful error messages for API failures
- Form validation with clear error messages
- Network connectivity checks during tests
- Auto-save functionality (save progress every 30 seconds)

6. SECURITY:
- Implement Row Level Security (RLS) in Supabase
- Protect all API routes with authentication
- Sanitize user inputs
- Rate limiting on API endpoints
- Secure cookie handling

SEO IMPLEMENTATION CHECKLIST:

1. META TAGS (use Next.js Metadata API):
- Title: "Aptitude Test Preparation | Practice for Placements - [YourPlatformName]"
- Description: "Best platform for aptitude test preparation with 500+ questions, company-specific PYQs from TCS, Infosys, Wipro. Practice & take mock tests with AI-powered analytics."
- Keywords: "aptitude test, placement preparation, TCS aptitude, Infosys test, quantitative aptitude, logical reasoning, online mock test"

2. STRUCTURED DATA (JSON-LD):
Implement for:

- Organization schema
- Course/Educational schema for practice modules
- FAQPage schema (add FAQ section on homepage)
- BreadcrumbList schema

3. CONTENT OPTIMIZATION:
- H1 on every page (unique, keyword-rich)
- Descriptive H2 and H3 subheadings
- Alt text for all images: "Quantitative Aptitude Practice Questions", "Logical Reasoning Test Interface"
- Internal linking between practice topics and test modes
- Create a /blog route (placeholder for future content)

4. TECHNICAL SEO:
- Generate sitemap.xml with all routes
- Create robots.txt (allow all, reference sitemap)
- Implement canonical URLs
- Add hreflang tags (for future multi-language support)
- 404 page with helpful navigation
- Implement proper redirects (301 for permanent, 302 for temporary)

5. PAGE SPEED:
- Optimize images (WebP format, responsive sizes)
- Minimize JavaScript bundles
- Implement code splitting
- Use Next.js font optimization
- Enable Vercel Edge Caching

6. MOBILE SEO:
- Viewport meta tag configured
- Touch-friendly elements
- No horizontal scrolling
- Fast mobile page load (<3 seconds)

═══

ADDITIONAL FEATURES:

1. Dark Mode Support:
- Toggle in user profile
- Use CSS variables or Tailwind dark mode
- Save preference in localStorage and database

2. Loading States:
   - Skeleton loaders for data fetching
   - Spinner for async actions
   - Progress indicators during test submission

3. Notifications:
   - Toast notifications for success/error messages
   - Use a library like react-hot-toast

4. Analytics Integration:
   - Add Google Analytics 4
   - Track events: test_started, test_completed, practice_session
   - Conversion tracking for sign-ups

5. PWA Features (Progressive Web App):
   - Add manifest.json
   - Service worker for offline support (basic caching)
   - Install prompt for mobile users

DEPLOYMENT & ENVIRONMENT:

1. Environment Variables (.env.local):
   NEXT_PUBLIC_SUPABASE_URL=your_supabase_url
   NEXT_PUBLIC_SUPABASE_ANON_KEY=your_anon_key
   SUPABASE_SERVICE_ROLE_KEY=your_service_role_key
   NEXT_PUBLIC_SITE_URL=https://yourplatform.com

2. Deployment Steps:
   - Deploy to Vercel (automatic from GitHub)
   - Connect Supabase project
   - Set up environment variables in Vercel
   - Configure custom domain
   - Enable Vercel Analytics

3. Monitoring:
   - Set up Vercel Analytics
   - Configure error tracking (Sentry optional)
   - Database query performance monitoring in Supabase

DESIGN GUIDELINES:

Color Palette:
- Primary: Indigo (#6366f1)
- Secondary: Purple (#8b5cf6)
- Success: Green (#10b981)
- Error: Red (#ef4444)
- Warning: Yellow (#f59e0b)
- Background: White (#ffffff) / Dark (#0f172a)
- Text: Gray-900 (#111827) / Gray-100 (#f3f4f6)

Typography:
- Font: Inter (Google Fonts)
- Headings: font-bold
- Body: font-normal
- Sizes: text-sm, text-base, text-lg, text-xl, text-2xl, text-3xl, text-4xl

Spacing:
- Use Tailwind spacing scale (p-4, m-6, gap-4, etc.)
- Consistent padding in sections (py-12, px-4)

Components:
- Buttons: rounded-lg, shadow-sm, hover effects
- Cards: rounded-xl, shadow-md, border
- Forms: rounded-md, focus:ring-2
- Modals: backdrop-blur, rounded-xl

Animations:
- Smooth transitions (transition-all duration-300)
- Hover effects on buttons and cards
- Loading spinners with animation-spin
- Page transitions (use Framer Motion if needed)

═══

FILE STRUCTURE:

```
your-app/
├── app/
│   ├── (auth)/
│   │   ├── login/
│   │   │   └── page.tsx
```

```
│   │   └── signup/
│   │       └── page.tsx
│   ├── (dashboard)/
│   │   ├── dashboard/
│   │   │   └── page.tsx
│   │   ├── practice/
│   │   │   ├── page.tsx
│   │   │   ├── [topicId]/
│   │   │   │   └── page.tsx
│   │   │   └── session/
│   │   │       └── [sessionId]/
│   │   │           └── page.tsx
│   │   ├── test/
│   │   │   ├── page.tsx
│   │   │   ├── configure/
│   │   │   │   └── page.tsx
│   │   │   ├── instructions/
│   │   │   │   └── [testId]/
│   │   │   │       └── page.tsx
│   │   │   ├── active/
│   │   │   │   └── [testId]/
│   │   │   │       └── page.tsx
│   │   │   └── results/
│   │   │       └── [testId]/
│   │   │           └── page.tsx
│   │   └── profile/
│   │       └── page.tsx
│   ├── api/
│   │   ├── auth/
│   │   ├── topics/
│   │   ├── questions/
│   │   ├── practice/
│   │   ├── test/
│   │   └── dashboard/
│   ├── layout.tsx
│   ├── page.tsx (homepage)
│   ├── loading.tsx
│   ├── error.tsx
│   └── not-found.tsx
├── components/
│   ├── ui/ (shadcn components)
│   ├── auth/
│   ├── dashboard/
│   ├── practice/
```

```
│   ├── test/
│   ├── common/ (Header, Footer, etc.)
│   └── analytics/
├── lib/
│   ├── supabase/
│   │   ├── client.ts
│   │   ├── server.ts
│   │   └── middleware.ts
│   ├── hooks/
│   ├── utils/
│   └── validations/
├── store/ (Zustand stores)
├── types/
├── public/
│   ├── images/
│   └── icons/
├── styles/
│   └── globals.css
├── .env.local
├── next.config.js
├── tailwind.config.ts
├── tsconfig.json
└── package.json
```

STEP-BY-STEP IMPLEMENTATION GUIDE:

Phase 1: Setup (Day 1-2)
1. Create Next.js project with TypeScript
2. Install dependencies (Tailwind, shadcn, Supabase, etc.)
3. Set up Supabase project and database schema
4. Configure authentication
5. Set up basic routing structure

Phase 2: Authentication (Day 3-4)
1. Build login/signup pages
2. Implement Supabase auth integration
3. Create protected route middleware
4. Build user profile page

Phase 3: Homepage & Navigation (Day 5-6)
1. Build landing page

2. Create header/footer components
3. Implement navigation
4. Add responsive design

Phase 4: Practice Mode (Day 7-12)
1. Build topic selection page
2. Create practice configuration modal
3. Implement practice interface
4. Add immediate feedback system
5. Build practice summary page
6. Integrate with Supabase

Phase 5: Test Mode (Day 13-20)
1. Build test type selection
2. Create test configuration page
3. Implement instruction screen with checks
4. Build full test interface with question palette
5. Add timer and auto-save functionality
6. Implement test submission flow
7. Create results and analytics dashboard

Phase 6: Dashboard (Day 21-24)
1. Build main dashboard
2. Create statistics cards
3. Implement charts (performance over time)
4. Add recent activity section
5. Build recommendations system

Phase 7: Data & Testing (Day 25-28)
1. Create seed data (500 questions)
2. Test all user flows
3. Fix bugs and polish UI
4. Optimize performance

Phase 8: SEO & Deployment (Day 29-30)
1. Implement all SEO optimizations
2. Add structured data
3. Test on multiple devices
4. Deploy to Vercel
5. Configure custom domain
6. Set up analytics

TESTING CHECKLIST:
- [ ] User can sign up and log in successfully
- [ ] User can select a topic and start practice
- [ ] Questions display correctly with all options
- [ ] Immediate feedback works (correct/incorrect)
- [ ] Practice summary calculates scores correctly
- [ ] User can configure and start a test
- [ ] Instruction screen displays all information
- [ ] Test interface works smoothly (timer, navigation, palette)
- [ ] Question palette reflects correct states
- [ ] Test submission saves data to database
- [ ] Results page displays accurate analytics
- [ ] Charts render correctly
- [ ] Dashboard shows correct statistics
- [ ] Profile updates save successfully
- [ ] Dark mode toggle works
- [ ] Responsive design works on mobile/tablet
- [ ] SEO meta tags are present on all pages
- [ ] Sitemap generates correctly
- [ ] All API routes are secured with authentication

═══

IMPORTANT NOTES FOR BOLT.NEW:
1. Generate complete, production-ready code (not placeholders)
2. Include proper TypeScript types throughout
3. Implement error boundaries for React components
4. Add comprehensive comments in complex logic
5. Follow Next.js 14 best practices (App Router)
6. Use server components where possible for better performance
7. Implement proper loading and error states
8. Make all components reusable
9. Follow accessibility standards (WCAG)
10. Optimize for Core Web Vitals

═══

START BUILDING with this priority order:
1. Set up project structure and dependencies
2. Configure Supabase with database schema
3. Implement authentication flow

4. Build homepage
5. Create practice mode (complete flow)
6. Build test mode (complete flow)
7. Implement dashboard and analytics
8. Add SEO optimizations
9. Polish UI/UX
10. Test and deploy

Build this as a fully functional MVP that can be immediately deployed and used by students. Focus on clean code, good UX, and performance.