🔧 Build a complete frontend for an online platform called "Online Entrance Exam PYQ Platform"

📱 Pages & Features to Build:

✅ 1. Homepage (`/`)

- Hero section: H1 ("Master Indian Entrance Exams..."), CTA buttons: "Start Practicing Now", "Login"
- "How It Works" section: cards with icon, heading, description
- Featured Exams: use `ExamCard` component with grid layout
- Testimonials (optional): carousel of student reviews
- Pricing and FAQ section (FAQ using `shadcn/ui` Accordion)
- Footer with navigation and social links

🔐 2. Authentication

`/login`, **`/register`**, **`/forgot-password`**

- Forms using `shadcn/ui Form`
- Login: Email, Password + Forgot Password link
- Register: Email, Password, Confirm Password, T&C checkbox
- Reset: Send email → set new password
- Centered in a card container with error/success handling
- Post-login, redirect to `/dashboard`

👨‍🎓 3. Dashboard (`/dashboard`)

- Header with welcome message and Logout button
- Section: “My Exams” (grid of purchased `ExamCard`s)
- Section: “Explore More Exams” (unpurchased exams with "Purchase Access" or "Access Granted")
- Section: Quick Start: buttons to “Start Instant Test”, “My Performance Analytics”

📘 4. Exam Details (`/exams/:examId`)

- Header: Exam title, description
- List of years as `YearCard`s with:
  - “Start Full Exam”
  - “Browse Paper”
  - “Review My Attempt”
- Status: New / Completed / Attempted
- Button: “Instant Test for [Exam]”

📝 5. Exam Simulation (`/exam-simulation/:paperId`)

- Fullscreen, fixed top bar: exam name, timer (color coded), toggle exam/browse mode
- Question area:
  - `QuestionDisplayCard` (Question No, text, options as radio buttons)
  - Buttons: "Clear Response", "Mark for Review", "Show Solution" (only in browse mode)

    - `SolutionDisplayArea`: correct answer, explanation, your answer, status icon
- Bottom nav: Previous/Next buttons
- Right sidebar / drawer: color-coded question palette

📂 6. Browse Paper Mode

- Same as simulation but:
    - No timer
    - Solutions always visible
    - No answer saving

🎯 7. Solve Specific Question

- Single question view with "Check Answer"
- Show whether it’s correct, explanation, next button

⚡ 8. Instant Test Mode

- Random PYQs one by one
- Optional timer
- Show mini-analysis at end (score, time)
- Button: “Review this mini-test”

📊 9. Performance Analytics (`/analysis`)

- Header: "My Performance Analytics", back to dashboard
- Summary: Cards showing exams attempted, total time, accuracy
- Line chart: “Score Trend Over Time” (exam dropdown, Recharts line chart)
- Bar chart: “Topic-wise Accuracy”
- Table: Recent attempts (exam, date, score, time, review button)

🤖 10. AI Doubt Clearing Centre (`/doubt-clearing`)

- Chat layout with:
    - Header: title + intro text
    - Scrollable message area with chat bubbles (user + AI)
    - Input textarea with placeholder and dynamic height
    - Send button (disabled during processing)
    - Loading dots during response
    - Auto-scroll to new message
- Future buttons to trigger AI from:
    - Solutions: "Explain this concept"
    - Analytics: "Why is my Chemistry weak?"

📄 11. Review Attempt Page (`/review/:attemptId`)

- Header: exam, score, time taken
- List of questions:

- Your answer, correct answer, status icon
- Toggle to show/hide solution
- Navigation buttons and optional palette

💄 UI Standards:

- Use Tailwind for styling, follow mobile-first approach
- All components (ExamCard, QuestionDisplayCard, MessageBubble, Timer, etc.) should be reusable
- Use `next/image` for all images
- Use Inter font via `next/font`
- Proper theming setup for light and dark modes
- Show meaningful loading states, error messages, and empty states throughout

---

🧩 State Management:

- **Context API**: auth status, global loading, theme
- **Zustand**: for complex states like exam session, question palette, performance data
- Authentication token in `Authorization` header of all fetch requests

🛠️ Technical Considerations:

- Use `getServerSideProps` for dynamic pages like dashboard
- Use `getStaticProps` for landing/static FAQ

- Use client-side fetching + SWR (optional) for interactive states
- Use dynamic imports (`next/dynamic`) to optimize performance
- Use `next/link`, `useRouter`, and semantic HTML
- Full keyboard navigation, ARIA roles, focus trapping in modals